

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут інформатики та радіoeлектроніки,
Факультет комп'ютерних наук і технологій
(повне найменування інституту, назва факультету)

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)

магістр

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ
РЕКОМЕНДАЦІЙНОЇ СИСТЕМИ ДЛЯ РОЗШИРЕНОЇ
ФІЛЬТРАЦІЇ ВІДЕО
RESEARCH AND SOFTWARE IMPLEMENTATION OF THE
RECOMMENDATION SYSTEM FOR THE ENHANCED
FILTRATION OF VIDEO

Виконав: студент(ка) 2 курсу, групи КНТ-219м
Спеціальності 122 Комп'ютерні
науки

(код і найменування спеціальності)

Освітня програма (спеціалізація)
Системи штучного інтелекту

Куш А.В.

(прізвище та ініціали)

Керівник Гладкова О.М.

(прізвище та ініціали)

Рецензент Морщавка С.В.

(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
 (повне найменування закладу вищої освіти)

Інститут, факультет ІРЕ, ФКНТ
 Кафедра програмних засобів
 Ступінь вищої освіти магістр
 Спеціальність 122 Комп'ютерні науки
 (код і найменування)

Освітня програма (спеціалізація) Системи штучного інтелекту
 (назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
С.О. Субботін
 “ ” 20 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Куц Анастасія Володимирівна

(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження та програмна реалізація рекомендаційної системи для розширеної фільтрації відео. Research and Software Implementation of the Recommendation System for the Enhanced Filtration of Video

керівник проєкту (роботи) Гладкова Ольга Миколаївна, к.т.н.,
 (прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від “03” листопада 2020 року № 301

2. Строк подання студентом проєкту (роботи) 5 грудня 2020 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області та постановка завдання. 2. Дослідження особливостей розробки рекомендаційних систем. 3. Проектування системи. 4. Вибір інструментарію розробки. 5. Конструювання рекомендаційної системи. 6. Керівництво з використання програмного продукту. 7. Розробка організаційно-економічної частини. 8. Розробка заходів з охорони праці та цивільної безпеки. 9. Додатки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Слайди презентації

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-6 Основна частина	ГЛАДКОВА О.М., доцент		
Організаціо-економічна частина	Пожуєва Т.О., професор		
Охорона праці та цивільна безпека	Коробко О.В., ст. викладач		
Нормоконтролер	Камінська Ж.К., асистент		

7. Дата видачі завдання “ 4 ” вересня 2020 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області та постановка завдання.	2–3 тижні	Розділ 1
3	Дослідження особливостей розробки рекомендаційних систем.	4 тиждень	Розділ 2
4	Проектування системи.	5 тиждень	Розділ 3
5	Вибір інструментарію розробки.	6–7 тижні	Розділ 4
6	Конструювання рекомендаційної системи.	8 тиждень	Розділ 5
7	Керівництво з використання програмного продукту.	9 тиждень	Розділ 6
8	Розробка організаційно-економічної частини.	10 тиждень	Розділ 7
9	Розробка заходів з охорони праці та цивільної безпеки.	11 тиждень	Розділ 8
10	Оформлення пояснювальної записки та документів до неї.	12-13 тиждень	Додатки
11	Нормоконтроль та рецензування.	14–15 тижні	
12	Захист роботи.	16 тиждень	

Студент(ка)

_____ Куш А.В.
(підпис) (прізвище та ініціали)

Керівник проекту (роботи)

_____ Гладкова О.М.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи магістра:
195 с., 42 табл., 95 рис., 10 формул, 2 дод., 105 джерел.

МЕДИЦИНА, ЯКІСТЬ, ВІДЕО, YOUTUBE, РЕКОМЕНДАЦІЙНА СИСТЕМА.

Об'єкт дослідження – програмні засоби для отримання якісного відео.

Предмет дослідження – вебзастосунок для надання рекомендацій, що являють собою список якісного відео.

Мета роботи – розробити рекомендаційну систему, яка дозволить експертам в медичній сфері швидко знаходити якісний відеоконтент та забезпечить їх механізмами для його оцінювання.

Матеріали, методи та технічні засоби: СКБД MySQL, мови Python та JavaScript, бібліотеки Vue.js і Vuex, вебсервер Nginx та вебсервіс Gunicorn, фреймворк Django, система контейнеризації Docker-compose.

Результати. Створено рекомендаційну систему у вигляді клієнт-серверного програмного продукту для експертів в сфері медицини, що дозволяє під час пошуку звузити список відео, що отримуються з системи YouTube, до найбільш якісних, а також виконати їх оцінювання.

Наукова новизна отриманих результатів. Отримано подальший розвиток гібридного алгоритму надання рекомендацій для обробки списку відео з метою надати більш релевантні результати пошуку.

Практична значимість результатів. Розроблена рекомендаційна система забезпечує додаткову фільтрацію і сортування списку відео, одже користувач отримує більш релевантний та якісний список відео, оскільки враховуються наявні виставлені експертами оцінки відео взяті з бази даних.

Галузь використання – в медичних закладах, для навчання в медичних університетах або для самоосвіти.

ABSTRACT

Explanatory note to the diploma qualifying work of the master: 195 pages, 42 tables, 95 figures, 10 formulas, 2 appendixes, 105 sources.

MEDICINE, QUALITY, VIDEO, YOUTUBE, RECOMMENDATION SYSTEM.

The object of study – software for getting the qualitative video.

The subject of the research – is a web application for providing recommendations, which is a list of qualitative video.

The purpose of the work is to develop a recommendation system that will allow medical experts to quickly find qualitative video content and provide them mechanisms for their evaluation.

Materials, methods and technical means: MySQL database, Python and JavaScript languages, Vue.js and Vuex libraries, Nginx web server and Gunicorn web service, Django framework, Docker-compose containerization system.

Results. A referral system has been created in the form of a client-server software product for medical experts, which allows to narrow the list of videos received from the YouTube system to the highest quality, as well as perform evaluation.

The scientific novelty of the obtained results is that the further development of a hybrid algorithm for providing recommendations for processing the video list in order to provide more relevant search results.

The practical significance of the results. The developed recommendation system provides additional filtering and sorting of the video list, so the user gets a more relevant and high-quality video list, as the existing video ratings taken by experts are taken from the database.

Field of use – in medical institutions, for study at medical universities or for self-education.

ЗМІСТ

Перелік скорочень та умовних позначок	9
Вступ.....	10
1 Аналіз предметної області та постановка завдання	12
1.1 Огляд популярних інформаційних ресурсів для навчання.....	12
1.2 Аналіз платформи YouTube	17
1.2.1 Аналіз YouTube як платформи для навчання	17
1.2.2 Дослідження роботи алгоритмів YouTube.....	19
1.3 Огляд критеріїв, яким повинен відповідати якісний контент	24
1.4 Дослідження медичного відео на YouTube	25
1.5 Визначення параметрів для оцінки	28
1.6 Постановка задачі	30
2 Дослідження особливостей розробки рекомендаційних систем	33
2.1 Дослідження рекомендаційних систем та ресурсів, що їх використовують.....	33
2.2 Аналіз рекомендаційних методів	36
2.3 Аналіз вимог	41
2.4 Висновки до розділу 2	42
3 Проєктування системи.....	44
3.1 Проєктування баз знань.....	44
3.2 Проєктування загальної архітектури системи	46
3.3 Проєктування структури СКБД.....	53
3.4 Проєктування прототипу графічного інтерфейсу	58
3.5 Висновки до розділу 3	64
4 Вибір інструментарію для розробки	66
4.1 Порівняння та вибір СКБД	66
4.2 Порівняння та вибір мови програмування для розробки серверної частини програмного продукту	68

4.3	Обґрунтування вибору мови програмування для клієнтської частини програмного продукту	70
4.4	Вибір фреймворку або бібліотек для розробки клієнтської частини	71
4.5	Вибір вебсерверу для клієнтської частини.....	73
4.6	Вибір фреймворку або бібліотек для розробки серверної частини ...	74
4.7	Вибір вебсервісу для серверної частини	77
4.8	Вибір фреймворку або бібліотек для розробки модулю асинхронних задач	79
4.9	Вибір засобів для розгортання проєкту	80
4.10	Вибір інтегрованого середовища	81
4.11	Висновки до розділу 4	83
5	Конструювання рекомендаційної системи	86
5.1	Конструювання серверної частини	86
5.2	Конструювання авторизації в системі YouTube	93
5.3	Конструювання модулю асинхронних задач	95
5.4	Конструювання клієнтської частини	104
5.5	Конструювання засобів розгортання проєкту.....	108
5.6	Висновки до розділу 5	111
6	Керівництво з використання програмного продукту	113
6.1	Призначення та умови використання програмного продукту	113
6.2	Функції системи	113
6.3	Вхідні та вихідні дані	114
6.4	Налаштування програмного продукту.....	114
6.5	Взаємодія із програмним продуктом	115
6.6	Висновки до розділу 6	125
7	Розробка організаційно-економічної частини.....	126
7.1	Оцінювання актуальності розробки.....	126
7.2	Аналіз ринкових можливостей запуску проєкту	127
7.3	Розробка ринкової стратегії проєкту	130
7.4	Планування етапів розробки програмного продукту.....	131

7.5	Розрахунок витрат на розробку та впровадження проєкту	133
7.6	Висновки до розділу 7	138
8	Розробка заходів з охорони праці та цивільної безпеки	140
8.1	Аналіз потенційних небезпек	140
8.2	Заходи по забезпеченню безпеки	141
8.3	Заходи по забезпеченню виробничої санітарії та гігієни праці	143
8.4	Заходи з забезпечення безпеки у надзвичайних ситуаціях	151
8.4.1	Заходи з пожежної безпеки	151
8.4.2	Заходи з цивільного захисту. Укриття населення у захисних спорудах цивільного захисту	154
	Висновки	158
	Перелік джерел посилання	161
	Додаток А Текст програми.....	172
	Додаток Б Слайди презентації	187

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

API	– Application Programming Interface
CSS	– Cascading Style Sheets
DDoS	– Distributed Denial of Service
HTML	– HyperText Markup Language
HTTP	– HyperText Transfer Protocol (протокол передачі гіпертексту)
JSON	– JavaScript Object Notation
ORM	– Object-Relational Mapping (об'єктно-реляційне відображення)
TCP/IP	– Transmission Control Protocol / Internet Protocol
WSGI	– Web Server Gateway Interface
БД	– База даних
СКБД	– системи керування базами даних

ВСТУП

Проблема неякісного контенту є досить масштабною та є чи не однією з найбільших в Інтернеті. Адже з розвитком Всесвітньої павутини та появою підключених до Інтернету пристроїв в кожному будинку, ризик натрапити на неякісний контент збільшився в рази. Щосекунди на просторах Інтернету публікується нова інформація, при чому часто її якість та актуальність не перевіряється. Особливо гостро дана проблема постає під час використання додаткових матеріалів у навчанні.

Залучення відео матеріалів, презентацій під час лекцій, при самостійному чи дистанційному навчанні давно знайома та актуальна практика. Такий підхід дозволяє спросити важкий матеріал та сприяє його кращому засвоєнню. Часто в якості додаткових матеріалів використовують відео саме з сервісу – YouTube.

В даній роботі розглянуто аргументи для використання відео з платформи YouTube, в якості додаткового матеріалу для навчання, до яких можна віднести безкоштовність відео та можливість збереження та перегляд відео в будь-який час на зручній швидкості. А до основних проблем відносяться: відсутність визначання якості та правильності контенту в відео, необхідність перегляду експертом великої кількості відео для визначення якісного, необхідність перевірки каналів на предмет, того, що їх ведуть експерти.

Саме тому в даній роботі розглянуто проблему наявності неякісного контенту на YouTube, а також описано спосіб для вирішення даної проблеми. Така розробка є актуальною та необхідною, оскільки механізми контролю, що оцінюють відео щодо медичних навичок за змістом, якістю, точністю та достовірністю інформації, нажалі на платформі не передбачені.

Розроблювана система орієнтована на експертів в сфері медицини та покликана допомогти їм під час пошуку необхідного відео. Завдяки можливостям, які YouTube надає розробникам можна отримати інформацію,

що дозволить розробити спеціальне програмне забезпечення, що здатне допомогти в оцінці якості відео. Така розробка може стати прототипом та поштовхом для власників платформи до створення необхідних алгоритмів, а згодом можливо і до розвитку платформи в напрямку освіти.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1 Огляд популярних інформаційних ресурсів для навчання

Поява Всесвітньої павутини, подарувала можливість знаходити та використовувати інформацію будь-де та будь-коли. Таким чином зникає необхідність походу до бібліотеки та переручуванні купи книжок в надії знайти потрібну інформацію, адже Інтернет надає доступ до величезних обсягів інформації та можливості її фільтрації, наприклад наявність пошуку за словом на вебсайті.

На рисунку 1.1 наведено деякі популярні інформаційні ресурси світу, список може бути набагато більшим, оскільки число соціальних мереж, платформ, месенджерів та інших інформаційних ресурсів стрімко зростає [1].

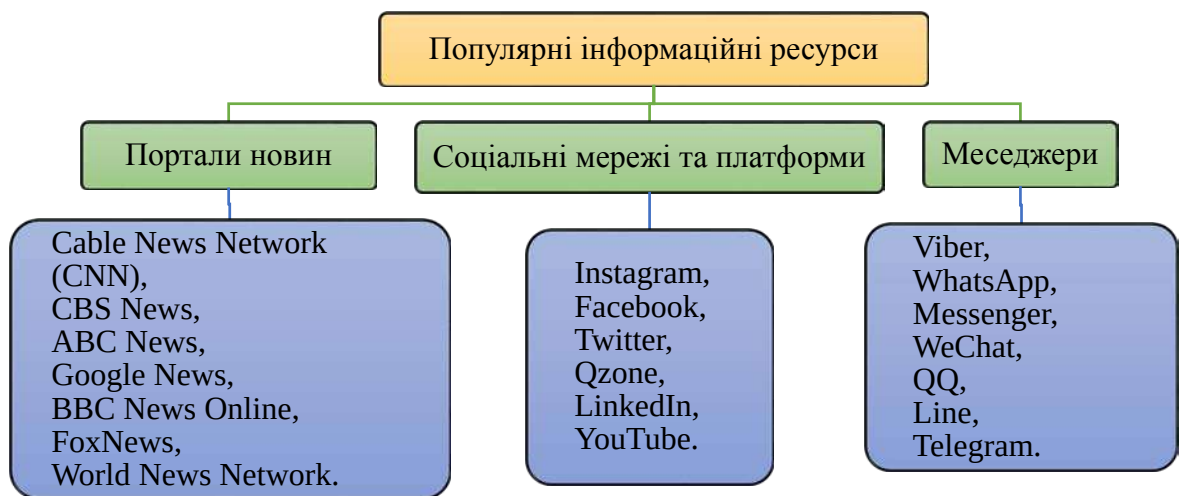


Рисунок 1.1 – Популярні інформаційні ресурси світу

Інтернет допомагає не тільки в поширенні новин, файлів чи повідомлень, він також надає широкі можливості для навчання. Його популярність, як інформаційного ресурсу для студентів полягає в швидкості та легкості знаходження потрібної інформації. Завдяки інтеграції мультимедійних засобів в навчання у студентів підвищується інтерес до предмету та покращується навчання.

Всесвітня павутина пропонує безліч сайтів та платформ, як спеціалізованих для дистанційного навчання в різних університетах, школах, коледжах, на онлайн-курсах, так і тих, що пропонують користувачам різний контент. На рисунку 1.2 представлено деякі приклади популярних платформ.

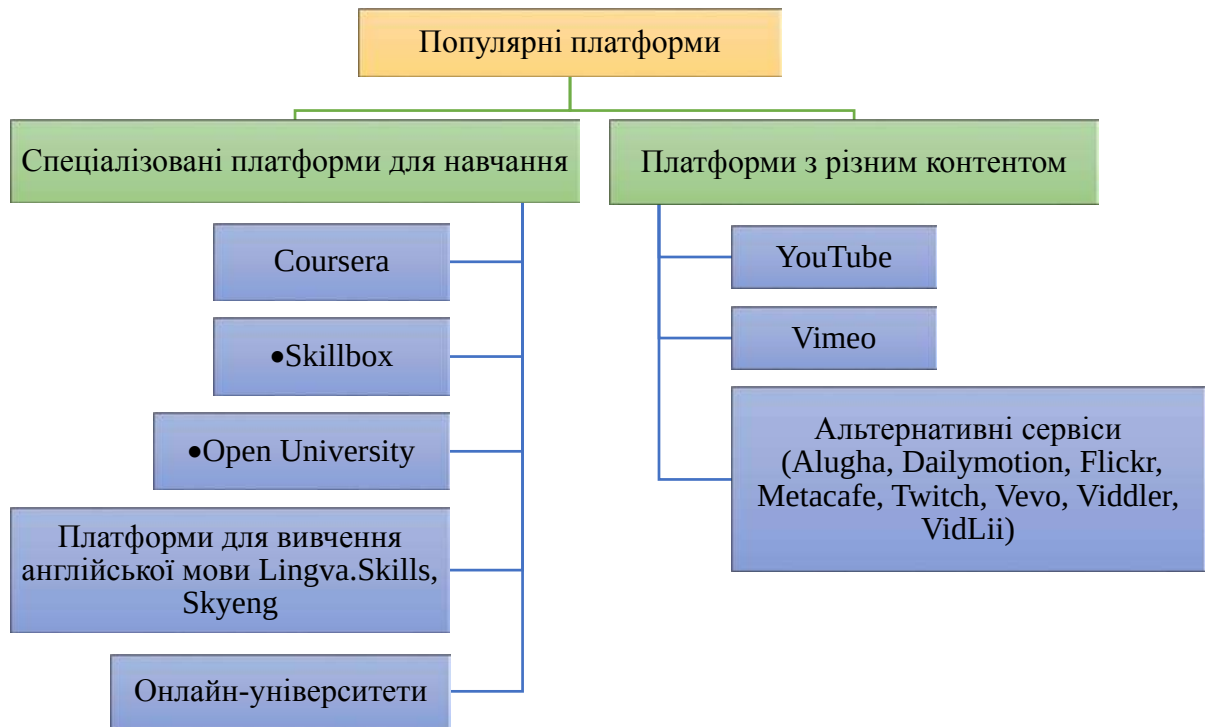


Рисунок 1.2 – Популярні платформи світу

Імовірність, що інформація наявна на спеціалізованих платформах чи сайтах буде більш коректною, якісною та правдивою, ніж на платформах з різним контентом, є значно вищою. Наприклад, на сайті [2] представлено перелік найкращих онлайн-університетів США, що дозволяють студентам вивчати найкращі програми дистанційно. Перелік деяких популярних спеціалізованих на навчанні онлайн-платформ наведено в таблиці 1.1 [3].

Найбільш популярною формою представлення інформації в Інтернеті стає відео формат. Сьогодні найвідомішими сервісами, що забезпечують користувачів відеоконтентом є YouTube та Vimeo.

Таблиця 1.1 – Опис популярних спеціалізованих онлайн-платформ [3]

Назва ресурсу	Опис ресурсу
Coursera	Платформа надає можливість отримати знання з різноманітних дисциплін. З системою співпрацюють 180 навчальних закладів з 27 країн, а також такі корпорації як Google та Intel. По закінченні навчання студент отримує диплом, який визнається іншими країнами світу.
Skillbox	Студенти вивчають найбільш актуальні області знань, та мають можливість отримати пропозиції про стажування чи працевлаштування одразу після завершення навчання, оскільки платформа співпрацює з найбільшими компаніями-роботодавцями. В якості викладачів виступають провідні спеціалісти в своїй сфері.
Open University	Британський онлайн-університет з високим рівнем якості, що спеціалізується на дистанційній освіті. На платформі представлено 180 напрямків освіти як загальноосвітніх так і спеціалізованих.
Lingva.Skills	Безкоштовний соціальний проєкт для вивчення англійської мови, розроблений за підтримки уряду України.
Skyeng	Онлайн-школа для вивчення англійської мови. Школа має власну інтерактивну платформу, заняття на якій з викладачем поводяться цілодобово, мобільний застосунок та навчальну розсилку.

Vimeo – платформа, що дозволяє завантажувати, переглядати та скачувати відео. Якість відеоконтенту досягається завдяки суворим правилам щодо розміщення відео. Головною особливістю платформи є відсутність на ній реклами. Користувач може створити безкоштовний аккаунт, однак для отримання повноцінного функціоналу необхідно придбати один з пакетів [4]. YouTube – є найпопулярнішим відеохостингом, що надає користувачам послуги для роботи з відео. З даним [5] він став другим сайтом у світі за кількістю відвідувачів.

В таблиці 1.2 представлено порівняння YouTube та Vimeo [6]. YouTube та Vimeo пропагують різні підходи для роботи та функціонування своїх платформ. Через свої суворі правила Vimeo є менш популярним ніж YouTube. Це можна пояснити, що оплата на сервісі проводиться за якісний, а не за популярний контент. Крім того користувачі більш лояльні та залишають конструктивні та доречні коментарі. YouTube не накладає на користувачів практично жодних обмежень, можна створити та розвивати власний канал безкоштовно. Монетизація на сервісі присутня, однак на відміну від Vimeo,

вона реалізована за рахунок реклами, партнерських програм, платних оглядів, тощо. YouTube надає більше безкоштовних можливостей з розвитку власного каналу ніж Vimeo, серед них відсутні обмеження в зберіганні та завантаженні відео, широкі можливості для редагування відео, безкоштовна базова та детальна аналітика по каналу [6].

Таблиця 1.2 – Порівняння YouTube та Vimeo [6]

Параметр порівняння	YouTube	Vimeo
Зручність пошуку	Пошуковий запит автоматично відображається в системі пошуку Google. Іноді складно знайти потрібний контент.	Контент розділений на 12 категорій, а відео автоматично завантажується до групи за певною темою. Такий підхід спрощує пошук корисної інформації.
Мобільність	Високий рівень адаптації та багато оновлень.	Швидко адаптується до будь-якого мобільного пристрою.
Вартість	Використання платформи разом з її функціями в будь-яких цілях безкоштовна.	Можна використовувати безкоштовно, але за стандартний пакет з розширеними можливостями доведеться платити.
Технічна підтримка	Підтримка в основному полягає в публікації безлічі гайдів та інструкцій.	При оплаті стандартного пакету платформа надає розширену персональну підтримку.
Зберігання інформації	Відсутні обмеження, тобто користувач може безкоштовно завантажити будь-яку кількість відео.	Обмежене безкоштовне зберігання відео, користувачу надається тільки 500 МБ на тиждень. Для отримання 5ТВ для зберігання доведеться заплатити.
Реклама	Велика кількість реклами.	На платформі відсутні рекламні ролики.
Можливості редагування відео	Вбудований відео-редактор або ресурс Google Фото. Додавання до відео посилань, анотацій, титрів. Проведення прямих ефірів.	Безкоштовно можна додати на відео титри, назву, теги. Назву ролика можна змінювати без зміни URL-адреси.
Аналітика	Безкоштовна. Можна проаналізувати тривалість перегляду відео, кількість підписок чи відписок, тощо.	Безкоштовно доступна тільки базова статистика, за детальну аналітику доведеться платити.

Окрім YouTube та Vimeo існують ще й альтернативні сервіси, такі як Alugha, що зосереджена на створенні та поширенні багатомовного контенту, Viddler – на корпоративних відео, Twitch – трансляція відеоігор і кібер-

спортивних турнірів, Vevo – на музичних кліпах. Metacafe – це сервіс для коротких відео, а Flickr є платформою для обміну фото, де також можна розміщувати короткі відео. Однак на відміну від YouTube, який об’єднує в собі безліч різноманітних тем відеоконтенту, альтернативні сервіси більше орієнтовані на певну конкретну тему [4].

YouTube є вже не просто розважальним сервісом, він містить велику кількість навчального контенту. В таблиці 1.3 наведено порівняння платформи з спеціалізованих систем для навчання [3].

Таблиця 1.3 – Переваги та недоліки платформи YouTube та деяких спеціалізованих систем для навчання [3]

Системи		Параметри	
		Переваги	Недоліки
Системи з різним контентом	YouTube	Безкоштовне використання. Різноманіття контенту. Загальнодоступність. Багатомовність. Популярність. Можливість використання субтитрів.	Наявність вбудованої реклами, за відсутності якої необхідно оформити платні підписку. Відсутність якісної перевірки контенту.
Спеціалізовані системи	Coursera	Дипломи, що визнаються іншими країнами. Є безкоштовні курси. Велика кількість навчальних програм.	Практично відсутні дипломи на безкоштовних курсах. Курси проходять в онлайн режимі, на які необхідно реєструватись заздалегідь.
	Skillbox	Якісні та актуальні знання. Можливість влаштуватись на роботу/стажування після навчання. Викладачі є спеціалістами в своїй сфері.	Висока вартість курсів. Мала кількість безкоштовних матеріалів. Вивчення певного напрямку займає багато часу (до року).
	Open University	Високий рейтинг освіти. Велика кількість доступних курсів. Диплом MBA.	Мова викладання тільки англійська. Всі курси платні. Складнощі з вступом для користувачів не з Великобританії.

Якщо розглянути переваги та недоліки YouTube та деяких спеціалізованих систем для навчання (табл.1.3) стає зрозумілим, що сервіс не

здатний повноцінно замінити системи дистанційного навчання, однак може запропонувати миттєву та безкоштовну допомогу навчанні. Популярність платформи забезпечила її різноманіттю відеоконтенту з багатьох тем, тож не дивно, що користувачі так часто звертаються до неї по допомогу в пошуках необхідної інформації, адже імовірність знайти саме те, що цікавить досить висока. Він чудово підходить для саморозвитку або в якості зручної подачі додаткового матеріалу студентам.

1.2 Аналіз платформи YouTube

1.2.1 Аналіз YouTube як платформи для навчання

Доказом переваг використання відеоматеріалу з YouTube для покращення навчання студентів, стали результати досліджень представлених в статті [7]. Вони показали, що студенти запам'ятовують та розуміють складні поняття набагато краще, коли вони подаються в візуальній формі, тобто у вигляді відео. Також до переваг використання YouTube в навчальних цілях можна віднести [8]:

- отримувати нові знання можна за допомогою будь-якого гаджету від комп'ютера до телефону, в зручний для користувача час;
- коментарі надають можливість діалогу між користувачами;
- різноманітня контенту, дозволяє знайти відео з будь-якої теми;
- платформа надає можливість до мікронавчання, тобто користувач може сам керувати вивченням матеріалу, наприклад, переглядати складні моменти повторно, повертатись до певного матеріалу за потреби, тощо;
- проведення вебінарів, лекцій, тощо в прямих-ефірах на платформі з можливістю спілкування з аудиторією за допомогою чату.

Автори YouTube каналів мають повну свободу дії, у той же час така свобода та відсутність механізмів перевірки якості відео, що завантажуються, призводять до великої кількості неякісного контенту. Очевидно, що імовірність виникнення ризику особливо висока для студентів, що

намагаються знайти інформацію потрібну для навчання, оскільки їм може бракувати інформаційної грамотності, щоб оцінити якість контенту.

З метою оцінки ефективності використання YouTube, як платформи для вивчення студентами анатомії, в коледжі медицини Шардського університету протягом навчального року було проведено дослідження, в якому прийняли участь студенти-медики другого курсу. Студенти заздалегідь отримували необхідні матеріали та відповідні посилання на YouTube, що використовувався як запропонований ресурс, під час кожного заняття. В рамках дослідження було запущено канал HumanAnatomy Education (HAE) [9].

Результати дослідження показали, що 98% студентів використовували YouTube як інформаційний ресурс, 92% зазначили, що канал допоміг їм вивчити анатомію. На основі отриманих результатів дослідження можна зробити висновок, що YouTube може розглядатись як ефективний інструмент для вивчення, якщо відеоролики будуть різноманітними, спрямованими на досягнення цілей курсу та обов'язково ретельно перевірятимуться [9].

В таблиці 1.4 наведено приклади перевірених каналів на YouTube з медицини [10].

Таблиця 1.4 – Медичні канали в сфері медицини [10]

Назва	Опис
Осмос	Англомовний канал для вивчення біології, хімії та медицини. Інформація про різні розділи медицини детально представлена в коротких, візуальних відео. Канал корисний для студентів-медиків та школярів.
Strong medicine	Канал пізнавальних навчальних відео з різних тем медицини, що розроблені Еріком Стронгом, який є клінічним асистентом професора медицини в Стенфордському університеті і практикуючим госпіталем.
Alila Medical Media	На каналі представлені прості та зрозумілі мінілекції з різних тем медицини, що ідеально підходять для студентів, викладачів і пацієнтів.
School of surgery	На каналі представлені відео з практичної хірургії і не тільки, зроблені експертами для студентів-медиків і лікарів.
Lecturio Medical Education	Освітній канал з медицини, на якому лекції читають досвідчені фахівці, експерти USMLE, лауреати навчальних курсів і професора світового рівня. Канал буде корисним для лікарів, викладачів, студентів-медиків, медсестр і будь-яких інших медичних працівників, а також для тих просто цікавиться медициною.

Керівництво YouTube розуміючи весь потенціал платформи в 2018 році заявило про виділення коштів та співпрацю з авторами, що створюють навчальний та практичний контент. Так, з'явився новий канал YouTube Learning з перевіреним контентом, де представлені освітні відеозаняття з різних тем. Однак, окрім цього каналу на платформі також безліч інших освітніх каналів, які користувачі використовують в навчальних цілях, на яких проблема якості контенту все ще залишається невирішеною.

1.2.2 Дослідження роботи алгоритмів YouTube

YouTube постійно намагається створити якісні персоналізовані рекомендації для безлічі користувачів, однак є три основні аспекти: масштаб, свіжість відео, шум, що ускладнюють цей процес [11].

Масштаб – обробка великого масиву відео. Проблема полягає тому, що для роботи з великою базою користувачів та відеоматеріалами, що зберігаються та постійно додаються на платформу, необхідно використовувати високоспеціалізовані алгоритми розподіленого навчання та ефективні системи обслуговування [11].

Свіжість відео – висока чутливість та динамічність моделі рекомендацій, які повинні бути точними для глядачів. Платформа є надзвичайно динамічною, оскільки щосекунди на її сервери завантажують безліч відеоматеріалу. Тому система рекомендацій повинна бути досить чутливою до нового завантаженого контенту та до останніх дій користувача. Тож головною проблемою є необхідність встановлення балансу між новим та добре зарекомендованими відео [11].

Шум. Поведінку користувачів на платформі досить складно передбачити через рідкість та різноманітність зовнішніх факторів, що не простежуються. Платформа не завжди отримує правдиві дані про вподобання користувача, тому моделює шуми та неявні сигнали зворотного зв'язку. Окрім того метадані пов'язані з контентом погано структуровані [11].

Платформа зацікавлена в поверненні глядачів, тому намагається знизити ризики, які завдають перераховані вище критерії. Однак, окрім цього на якість рекомендацій впливають ще й такий фактор, як можливості серверів їх обчислювальна потужність. Тому YouTube повинен знаходити баланс між точністю рекомендацій та прийнятною вартістю обчислювальної потужності.

Ресурс використовує свої алгоритми для надання найбільш персоналізованого контенту, на основі аналізу поведінки користувачів, історії переглянутих ними відео, час затрачений на користування системою, рейтинг кліків, вподобань та коментарі [12]. За допомогою такого аналізу YouTube надає відео своїм користувачам на п'яти розділах платформи: пошук, домашнє відео, рекомендовані відео, тренди, підписки. В таблиці 1.5 наведено опис кожного з розділу [12-14].

Таблиця 1.5– Опис розділів рекомендацій платформи YouTube [12-14]

Розділ	Опис
Пошук	Виконується порівняння запиту користувача з наявними даними про відео. Для цього враховуються ключові слова, назва (заголовок), опис ролику та кількість відео, які користувач переглянув на конкретному каналі, а також інші відео, які були переглянуті на одну і ту саму тему.
Домашнє відео	Головна сторінка, яку бачить користувач у додатку чи сайті YouTube. Для відображення відео на сторінці виконується аналіз історії активності користувачів на її основі знаходяться нові відео, що можуть бути актуальними. Вони класифікуються за тим, наскільки відео задовольняє та приваблює користувачів, як часто кожен глядач переглядає відео з кожного каналу чи відео, що мають подібну тему.
Рекомендовані відео	Розділ відображається на домашній сторінці. Тут представлені рекомендації з різних каналів, що дозволяє урізноманітнити основну сторінку.
Тренди	Сторінка, де відображаються нові та популярні відео у країні користувача. Завдяки цій сторінці YouTube прагне забезпечити баланс між новими та популярними відео. Для класифікації відео в розділі враховують кількість переглядів та їх темпи зростання для кожного з роликів.
Підписки	Розділ на якому користувачі можуть переглядати нові відео завантажені на канали, на які вони підписані. Для визначення рейтингу використовується показник – швидкість перегляду, що вимірює кількість підписників, які переглянули відео одразу після публікації. Чим вище значення цього показника, тим вище рейтинг відео.

Також для ранжування та рекомендації відео в кожному з розділів платформи алгоритм YouTube використовує різні сигнали та метрики. Тим самим система допомагає користувачам знайти відео, що їх зацікавить, а також дозволяє ресурсу утримувати користувачів та забезпечує їх регулярне повернення [13]. На роботу алгоритму з вибору відео для будь-якої людини, впливають низка різних факторів, такі як [15]:

- тип контенту, який користувач регулярно або нечасто переглядає;
- час, який люди витрачають на перегляд певного відео (переглядають до кінця чи вимикають тільки на декілька секунд);
- швидкість з якою відео набирає популярність;
- як часто канал завантажує нові відео;
- подобається, не подобається, кількість коментарів, чи діляться відео;
- будь-які негативні відгуки.

За словами головного директор з продуктів YouTube Ніла Мохана більше 70% часу проведеного на платформі, витрачається на перегляд рекомендованого контенту, а середній час сеансу з мобільного пристрою складає годину, що говорить про ефективність розробленого алгоритму [16]. Інформація про роботу та функціонування YouTube є досить секретною, у 2016 році платформа надала обмежену інформацію, щодо роботи їх алгоритмів у опублікованій роботі «Deep Neural Networks for YouTube Recommendations» [11,16].

Система YouTube працює за п'ятьма основними принципами [17]:

- принцип подвійної фільтрації (використовується комбінація з двох нейронних мереж);
- А/В тестування;
- присвоєння ранг/рейтинг кожному відео з набору варіантів, що призначені для конкретного користувача;

- поправочні статичні коефіцієнти (precision, recall, ranking loss і інші);
- результат оцінювання отриманий опираючись на стабільні зміни ряду показників залученості глядача, наприклад CTR (Click-Through rate) та watch time – час перегляду.

Використання двох нейронних мереж дозволяє YouTube надавати своїм користувачам більш якісні рекомендації. В документі [11] представлено загальну архітектуру системи рекомендацій (рис. 1.3), яка складається з двох нейронних мереж: одна для генерації кандидатів, а інша для оцінювання.

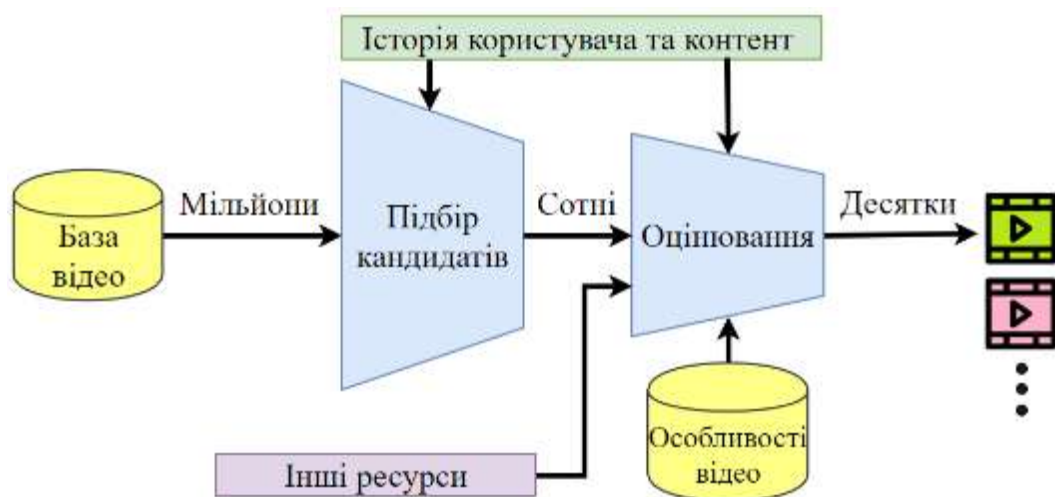


Рисунок 1.3 – Архітектура рекомендаційної системи [11]

Мережа генерації кандидатів опираючись на історію активності користувача (наприклад, ідентифікатори відео, які переглядаються, історію пошуку та напрямок переглядів, демографічні показники користувача), як на вхідні дані, виконує фільтрацію наявного відеоматеріалу та отримує множину з кілька сотень відео, які можуть бути цікавими користувачу [11, 17, 18]. Мережа забезпечує широку персоналізацію за допомогою спільної фільтрації. Тут подібність між користувачами виражається в грубих характеристиках, таких як ідентифікатори відео-годин, пошукові маркери та демографічні показники [11].

Рейтингова нейронна мережа отримує сформовану в першій нейронній мережі множину та виконує ранжирування – це процес, в результаті якого отримана вибірка з потенційно цікавим відеоконтентом розміщується у відповідності до пріоритетів. Мережа присвоює кожному відео бали відповідно до бажаної функції, використовуючи набір функцій, що описують відео та користувача. Відео, що отримали найвищі бали відображаються користувачу за рейтингом [11, 17].

В обох мережах автори використовують різні хитрощі та офлайн-метрики (точність, відкликання, втрата рейтингу тощо) для керівництва ітераційними вдосконаленням системи [11]. По-перше, підготовка підмережі до перетворення розрізнених функцій (таких як ідентифікатори відео, пошукові маркери та ідентифікатори користувачів) у щільні функції та вивчення вбудовування цих функцій разом з іншими параметрами моделі за допомогою градієнтного спуску. По-друге, щоб впоратись з проблемою експлуатації/розвідки, вік навчального прикладу визначається як особливість. Це допомагає подолати неявну упередженість у моделях, що рекомендують несвіжий вміст, як результат середньої ймовірності перегляду під час навчання. Тому вік прикладу встановлюється нуль, що дозволяє компенсувати цей факт [18].

Для остаточного визначення ефективності алгоритму чи моделі використовують тестування A/B за допомогою експериментів в реальному часі, які дозволяють виміряти незначні зміни швидкості кліків, часу перегляду та багатьох інших показників, що вимірюють залучення користувачів. Оскільки, результати живих переходів не завжди співвідносяться з автономними експериментами, тому такий підхід є досить важливим [11].

Оскільки обчислювальна потужність серверів платформи в основному направлена на надання персоналізованих рекомендації та обробку нових годин відео, що постійно завантажуються, то відповідно, статті витрат на процес перевірки якості відеоконтенту мінімальні.

1.3 Огляд критеріїв, яким повинен відповідати якісний контент

Користувачі стають все більш вимогливими до якості, актуальності та унікальності інформації, яку надає Всесвітня павутина. І тут головними проблемами є по-перше не пропорційність зростання кількості інформації та наявності (розробки) алгоритмів її оцінки та фільтрації для забезпечення якості, а по-друге відсутність чіткого, так би мовити словникового, задокументованого визначення терміну якість контенту, що є одним з основних факторів для розвитку будь-якого ресурсу.

Одним з прикладів визначення терміну контент можуть слугувати наступні формулювання [19]:

- контентом називають збірний термін, що характеризує будь-яку інформацію на сторінках сайту;
- контентом можна назвати тексти, аудіо- і відеофайли, графічні зображення, анімацію – все, що користувач може прочитати, побачити, почути на сайті [19].

Отже, для будь-якого вебресурсу контентом вважається весь інформаційний вміст, що представлений на його сторінках в різних видах. Вимоги до його якості будуть відрізнятися в залежності від платформи, на якій розміщується інформація та від самого виду контенту. Тим не менш можна виділити п'ять спільних критеріїв для будь-якого виду контенту [20]:

- користь – несе певну практичну цінність для користувача;
- тематичність – зберігає єдину тематику та розкриває поставлену тему протягом всієї статті чи відео;
- довіра – наданий матеріал підкріплений посиланнями на використану інформацію, що дозволяє викликати у користувача довіру до контенту;
- унікальність та оптимізація контенту під ключові запити, допомагають пошуковим системам та враховуються при формуванні релевантних результатів пошуку [21];

- актуальність;
- віральність – це здатність контенту поширюватись самостійно, тобто він викликає у користувача бажання ділитися ним з іншими користувачами [22].

В залежності від виду контенту та ресурсу його публікації вимоги до якості будуть змінюватись, окрім того користувачі також вносять свої правки.

Проблема неякісного контенту є досить масштабною та є чи не однією з найбільших в Інтернеті. Адже з розвитком Всесвітньої павутини та появою підключених до Інтернету пристроїв в кожному будинку, ризик натрапити на неякісний контент збільшився в рази.

1.4 Дослідження медичного відео на YouTube

Нажаль на сьогодні проблема якості відеоконтенту на YouTube досить масштабна. Так в статті [7], де представлені результати дослідження використання YouTube для покращення навчання, окрім того, що говориться про зручність та актуальність використання платформи для навчання, також зазначається, що головною проблемою такого формату навчання є відсутність перевірки якості контенту, тому вибором відео повинен займатись експерт.

Особливо гостро дана проблема стоїть саме в сфері медицини, адже дезінформація працівників медичних закладів, може коштувати комусь життя. Це підтверджують результати наведені в статті [23], де зазначено, що під час перевірки відео пов'язаних з процедурою імобілізації перелому дистального радіусу (фіксація після перелому зап'ястя) з 68000 відео тільки 16 відповідали поставленим критеріям якості. При цьому з 16 отриманих відео тільки 6 отримали оцінку з якості освітніх цінностей рівну 4 або 5, а решта 10 отримали оцінку нижче або рівну 3 [23]. Про необхідність перевірки відео говорить і хірург Стенфордського госпіталю Олівер Ааламі [24]. Про сумнівну якість контенту в сфері медицини на YouTube говорять і результати досліджень наведені в статтях за посиланнями [25, 26].

В статті [25] представлено результати та висновки щодо медичних відео на YouTube, на основі раніше опублікованих робіт з подібною тематикою. В результаті дослідження було виявлено, що 65% статей зробили висновок, що медичні відеозаписи на YouTube містять неправдиву інформацію, у той час коли 35% навпаки рішуче рекомендують їх як джерело освіти для пацієнтів. Також визначено, що найпопулярнішими та часто використовуваними інструментами для оцінювання надійності відео на платформі стали DISCERN, журнал американської медичної асоціації (JAMAS) та глобальні показники якості (GQS). За результатними дослідження визначено, що з 2675 оцінених медичних відеозаписів 1589 (59%) класифікуються як такі, що мають корисний вміст, який може позитивно впливати на освіту пацієнтів, тоді як 1086 (41%) класифікуються як оманливі [25]. Також результати представлені на рисунку 1.4 у вигляді діаграм.

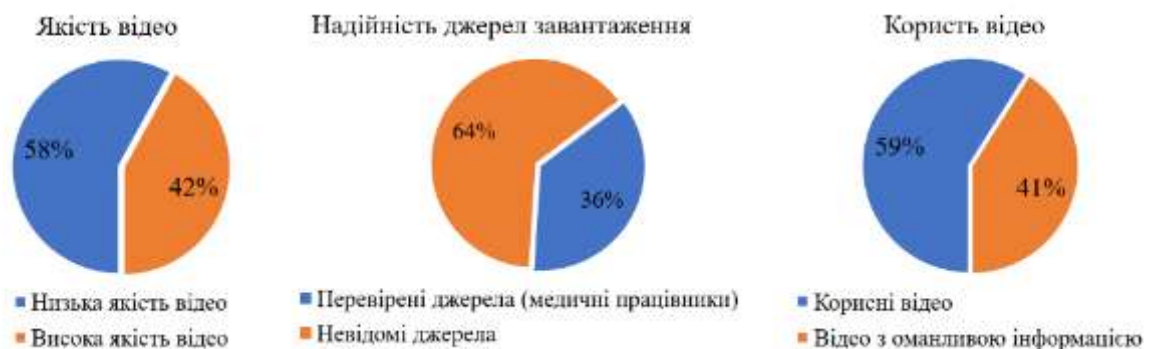


Рисунок 1.4 – Результати аналізу відео на YouTube

В статті [26] представлено результати оцінки змісту, точності, надійності та якості відео на YouTube щодо інтубації, яка є однією з найважливіших навичок для медичних працівників. З 54000 знайдених відеороликів перші 50, які були на англійській мові, були включені в дослідження для оцінки після фільтрації. Вибрані відео розділили на три категорії на основі джерел, які створили або завантажили відео на платформу:

– академічні (академічні лікарем або особи, що є частиною академічного закладу);

- медичні працівники (лікарі, медсестри, фельдшери тощо);
- медичні (із вебсайтів, пов'язаних зі здоров'ям).

Також були виділені основні характеристики обраних відео, які можна розділити на дві групи. До першої відносяться ті, що безпосередньо пов'язані з процедурою проведення інтубації медиками [26]:

- призначення відео (медичні працівники чи пацієнти);
- суб'єкт для демонстрації (манекен, людина, анімація);
- наявність фактичної сцени з клінічної практики, конкретні проблеми пов'язанні з інтубацією;
- тип інтубації.

До другої групи відносяться характеристики відео пов'язані з роботою платформи YouTube: дата публікації, тривалість, кількість переглядів, коментарів, кількість користувачів яким сподобалось чи не сподобалось відео.

Результати аналізу обраного відео показали, що більшість (92%, $n=46$) досліджуваного відео матеріалу були завантажені першою та другою групою, тобто медичними працівниками (50%, $n=25$) та академіками (42%, $n=21$) відповідно. Та тільки 8% тобто всього 4 відеоролика було завантажено третьою групою, тобто медичною, до якої відносились відео з вебсайтів, що стосуються охорони здоров'я. Також 54% відеороликів ($n=27$) були призначені як навчальні, а решта демонстрували лише щоденну практику [26]. Результати щодо джерел, що завантажили відео представлені на рисунку 1.5.

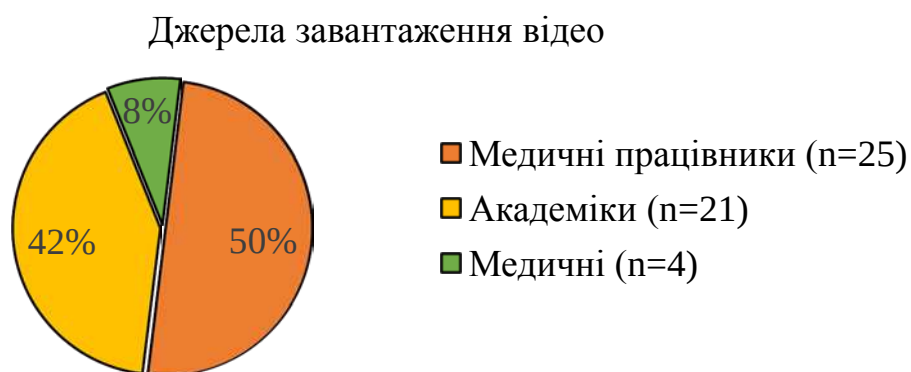


Рисунок 1.5 – Результати щодо джерел, які завантажили відео

Майже всі з обраних відеороликів були орієнтовані на медичних працівників (98%, $n=49$) відеороликів, та лише один (2%) із відеороликів був присвячений освіті пацієнтів. На більше ніж половині відео (72%, $n=36$) продемонстрували фактичну клінічну сцену з щоденних практик, а в якості суб'єкту для демонстрації використовували – людину [26]. На рисунку 1.6 зображено більш детальні результати.



Рисунок 1.6 – Результати отримані в статті

Надійність даних наданих відеозаписів, оцінювали на основі критеріїв журналу американської медичної асоціації (JAMA). Оцінка JAMA була значно вищою у відеороликах академічних джерел та у відеороликах з навчальною метою порівняно з показниками інших груп.

Поєднання письмової та візуальної інформації про будь-які медичні навички дозволяє краще уявити, засвоїти та зрозуміти матеріал студентами, що призведе до покращення результатів навчання. Однак такі відеоролики несуть відповідальність за точність наданої інформації, тому її якість та перевірка є надзвичайно важливими [23].

1.5 Визначення параметрів для оцінки

Оскільки платформа YouTube на сьогодні не передбачає повноцінну перевірку медичного контенту, то система, яка б виконувала оцінку якості медичного відео за змістом, якістю, точністю та достовірністю інформації, є актуальними та необхідними.

Для розробки такої системи надзвичайно важливим є параметри, за якими вона повинна працювати. Оскільки тематика відео пов'язана з медициною, то потрібно також враховувати і основні характеристики кожного відеоролику. Потрібні характеристики відео, що будуть використатись в якості параметрів для оцінки якості, можна розділити на дві групи: що стосуються інформації наданої в ролику та реакції на нього.

До першої групи відносяться такі параметри відео ролику як:

- джерела, що завантажили відео (медичний університет, лікарні, медичні працівники, інші);
- категорії відео (лекції, інструктаж, анімований відеоряд, короткотривалі відео, тощо);
- ключові слова;
- наявність опису відео.

До другої групи відносяться характеристики відео пов'язані з реакцією користувачів на переглянуте відео: якість наданої інформації в відео, якість медичної практики, якість опису, реакція користувачів, яким сподобалось чи не сподобалось відео.

Отримати характеристики відео, що завантажуються на платформу YouTube можна за допомогою API YouTube [27]. Для цього необхідно виконати три основні кроки:

- перший корок – реєстрація на сервісі YouTube, у випадку якщо канал все наявний, то можна використати логін та пароль від нього;
- другий крок – отримання спеціального ключа розробника (Developer Key);
- третій крок – виконання запитів.

Виконання третього кроку, а сам відправлення запитів в систему для отримання необхідної інформації, складається з двох основних запитів:

- перший запит – отримання списку відео;
- другий запит – отримання коментарів, тегів та опису відео.

Для отримання інформації по відео при виконанні другого запиту в його тілі необхідно вказати спеціальний ідентифікатор конкретного відео, який надходить в першому запиту.

Таким чином за допомогою можливостей, які YouTube надає розробникам можна отримати інформацію, що дозволить розробити спеціальне програмне забезпечення, що буде здатне допомогти в оцінці якості відео. Така розробка є актуальною та необхідною, оскільки механізми контролю, що оцінюють відео щодо медичних навичок за змістом, якістю, точністю та достовірністю інформації, нажалі на платформі не передбачені. Розроблювана система може стати прототипом та поштовхом для власників платформи до створення необхідних алгоритмів, а згодом можливо і до розвитку платформи в напрямку освіти.

1.6 Постановка задачі

В результаті аналізу предметної області було розглянуто проблему визначення терміну якісного контенту та проблему якості відео на популярній платформі YouTube, а також переваги та недоліки її використання в якості навчального ресурсу, доведена актуальність обраної теми.

YouTube є динамічним ресурсом, на сервери якого щосекунди завантажують безліч годин відео матеріалу. Популярність платформи забезпечила її різноманіттю відеоконтенту з багатьох тем, тож не дивно, що користувачі так часто звертаються до неї по допомогу в пошуках необхідної інформації, адже імовірність знайти саме те, що цікавить досить висока. Однак використання YouTube в якості повноцінної онлайн заміни сучасних навчальних закладів на сьогодні неможлива у зв'язку з рядом причин, а от варіант використання платформи в якості ресурсу, що надає допомогу в навчанні – є чудовим.

Доказом переваг використання відеоматеріалу з YouTube для покращення навчання студентів, стали результати досліджень, які

представлені в статтях [8, 10]. Згідно з якими студенти, що використовували YouTube в навчанні, засвоювали матеріал набагато краще, оскільки мали доступ до відеозаписів в будь-який зручний для них час та з будь-якого місця, а також мали можливість повторного перегляду в разі необхідності, через певний час для підтримання своїх знань.

Однак, окрім переваг існують також і недоліки – це не достовірність інформації, низька якість та відсутність перевірки відеоматеріалів на YouTube, складність пошуку якісних відео. Що збільшує ризик натрапити на оманливу інформацію, при чому імовірність виникнення цього ризику особливо висока для студентів, що намагаються знайти інформацію потрібну для навчання, оскільки їм може бракувати інформаційної грамотності, щоб оцінити якість знайденого контенту. Ці недоліки призводять також і до того, що викладачам, які прагнуть покращити навчання своїх студентів доводиться переглядати великий обсяг відео перш ніж знайти дійсно якісний контент.

Проаналізувавши та оцінивши переваги та недоліки платформи YouTube можна зробити висновок, на використання ресурсу як сервісу для навчання є підстави, однак суттєві недоліки не дають можливості на її повноцінне використання. Таким чином для забезпечення балансу у використанні доцільно розробити програмний продукт, який би використовуючи ресурси YouTube, дозволив спростити пошук відео для користувачів та забезпечити його якість. Для цього потрібно виконати ряд завдань:

- обрати модель представлення знань для рекомендаційної системи;
- спроектувати базу знань;
- проаналізувати та обрати СКБД;
- спроектувати архітектуру БД;
- розробити загальну архітектуру системи;
- розробити архітектуру та структуру користувацького інтерфейсу;
- обрати засоби контейнеризації.

В результаті отримаємо рекомендаційну систему у вигляді клієнт-серверного програмного продукту, який надаватиме спеціальний графічний інтерфейс для взаємодії із ним. Посилання на відео будуть зберігатися в базі даних та братись з платформи YouTube завдяки спеціальному API, що вона надає. Наповнення відеом базу даних програмного продукту на початковому етапі буде здійснюватися за рахунок пошуків користувачів, шляхом збереження посилань на відео у базі даних під час кожного пошуку. В якості користувачів будуть експерти в сфері медицини (лікарі, академіки, викладачі). Щоб впевнитись у тому, що користувач є дійсно фахівцем його реєстрація в системі здійснюється за допомогою адміністратора платформи. Для цього користувач повинен відправити на пошту необхідні дані, після чого в результаті успішного проходження перевірки він отримає доступ до системи.

Зареєстровані користувачі-експерти матимуть можливість оцінювати відео даного ресурсу, а саме визначати якісність, наявність практичних сцен, достовірність матеріалу відео тощо. Усі ці оцінки будуть збережені у базі даних. При наступних запитах, якщо посилання на відео, що прийде у відповіді від YouTube вже наявне в базі даних розроблюваного продукту, тоді братимуться дані з бази даних, з усіма попередніми оцінками. Програмний продукт надаватиме можливість звузити список відео під час пошуку до найбільш якісних. А рішення дати можливість оцінювати відео користувачам, надає можливість в майбутньому задіяти нейромодуль, що на основі вже оціненого відео, буде перевіряти нове відео, що надходитиме з системи YouTube та фільтрувати його.

Впровадження в розроблювальний продукт рекомендаційних алгоритмів дозволить надати користувачу додатковий функціонал у вигляді рекомендацій, що пришвидшать та спростять пошук необхідного відео матеріалу.

2 ДОСЛІДЖЕННЯ ОСОБЛИВОСТЕЙ РОЗРОБКИ РЕКОМЕНДАЦІЙНИХ СИСТЕМ

2.1 Дослідження рекомендаційних систем та ресурсів, що їх використовують

Рекомендаційними системами називають системи, що виконують фільтрацію інформації та будують з неї перелік об'єктів, яким може віддати перевагу користувач на основі рейтингу. Для формування рейтингу використовуються дані з профілю користувача, дані про його покупки та пошукові запити, дані про інших користувачів, а також інформацію про самі товари чи послуги [28].

Системи рекомендацій присутні практично на будь-якому сучасному вебсайті чи платформі, в електронній комерції та інтернет рекламі [29]. Розробка та впровадження рекомендаційних систем для більшості компаній в першу чергу є способом вирішення двох проблем [30]:

- версія прогнозу, тобто прогнозування рейтингового значення для комбінації елементів користувача, тобто на основі даних про оцінки, що виставив користувач, виконується прогнозування рейтингів елементів, з якими користувач не взаємодівав;
- версія для ранжування, тобто знаходження переліку найкращих пропозицій, у яких ймовірність сподобатись користувачу досить висока.

Системи рекомендацій використовують такі компанії, як Amazon, Netflix, Youtube, LinkedIn, Facebook, Pandora та інші з метою допомогти своїм користувачам у відкритті нових та актуальних елементів, які вони пропонують (відео, продукти, робота, музика) [31]. Використання рекомендаційних механізмів здебільшого обумовлено бажанням компаній у збільшенні їх прибутків, популярності, кількості зацікавленості аудиторії та затребуванні їх продукту [30]. Для компаній, що мають суттєву кількість продукції, яку вони пропонують, це ще й спосіб вирішити проблему, коли більша частина їх продукції може залишитись непоміченою користувачами [32]. Таким чином

завдяки ефективному використанню рекомендаційних систем інтернет ресурсам вдається якісніше забезпечувати потреби користувачів та створювати приємне користувацьке враження.

В таблиці 2.1 наведено приклади використання рекомендаційних систем на різних ресурсах [30, 32, 33].

Таблиця 2.1– Приклади використання рекомендаційних систем [30, 32, 33]

Назва	Ресурс
Amazon	Рекомендації на Amazon.com надаються на основі чітко наданих рейтингів, поведінки щодо покупки та історії перегляду, а також даних про клієнтів, що дозволяють визначити, які предмети зазвичай купуються разом. Для рекомендацій система використовує гібридні методи фільтрації.
Netflix	Близько 75% всіх фільмів на платформі обираються користувачами для перегляду в результаті роботи рекомендаційних систем. Для надання рекомендацій система використовує дані щодо виконаних користувачем налаштувань на Netflix, виставлених ним оцінки фільмам, а також подібні дані інших передплатників.
Youtube	Платформа використовує системи рекомендацій з метою допомоги користувачам в пошуках цікавого відео та підвищення рівня зацікавленості аудиторії. Для надання рекомендацій система аналізує безліч параметрів, які можна розділити на дві групи: дані про контент (зміст, назва, опис, теги) та показник залучення аудиторії (перегляди, час та відсоток перегляду ролику, кількість відміток «Мені подобається» чи «Мені не подобається», коментарі, відгуки користувачів) [33].
Соціальні мережі	Сервіси соціальних мереж (Linkedin, Facebook) використовують рекомендаційні механізми з метою запропонувати користувачам нові групи чи профілі. Рекомендації надаються на основі вивчення мережі зв'язків між користувачем та його друзями. Linkedin використовує дані з минулого досвіду користувача, поточні назви та затвердження, щоб запропонувати йому можливі вакансії.
Pandora	Музична база рекомендацій створюється на основі подібності з піснями та виконавцями обраними користувачем. Система є прикладом змістового підходу, оскільки, використовує зворотній зв'язок з користувачем для покращення роботи станції, що ґрунтується на основі його оцінок виставленим композиціям.

Окрім цього, Spotify, IMDB, Google News та багато інших платформ постійно дають рекомендації та пропозиції відповідно до наших потреб [30].

Всі рекомендаційні системи працюють практично однаково. Тому їх роботу можна розглянути на прикладі надання рекомендацій щодо книг системою Amazon (рис. 2.1). Спочатку система використовує дані про користувача, щоб визначити індивідуальні смаки та інтереси. Потім вона об'єднує отримані дані про конкретного користувача з колективною поведінкою всіх інших схожих з ним користувачів, і рекомендує речі, які можуть сподобатися [30].



Рисунок 2.1– Потік даних у механізмі рекомендацій [30]

Для надання якісних рекомендацій система використовує всю наявну інформацію, для її отримання виділяють два способи для збору даних: явний та неявний [30]. В таблиці 2.2 наведено опис даних способів [30, 34].

Таблиця 2.2 – Способи збору даних [30, 34].

Спосіб	Опис	Недоліки
Явний	Передбачає оцінювання користувачем предмету, вмісту, послуги, тощо, за допомогою виставлення оцінки, вибором відмітки «мені подобається» чи «мені не подобається», заповнення анкети, або іншим передбаченим способом.	Важко змусити користувача виконати оцінювання чи залишити відгук. У випадку отримання значення рейтингу складно визначити його характер. Оскільки, оцінка 3 може бути чудовим для одного, але середнім для іншого.
Неявний	Виконується протоколювання взаємодії користувача з сайтом, а отримані дані інтерпретуються як зацікавленість чи ні. Спосіб надає набагато більше даних, на основі яких відбувається автоматичне створення рейтингу.	Невизначеність, тобто неможливо визначити сподобався товар чи ні, корисною була отримана інформація чи ні, причину відмови від покупки, тощо.

Прикладами явного способу збору даних може стати інформація про минулу діяльність на сайті, рейтинги, залишені відгуки, або ж інформація зазначена і профілі користувача, наприклад, його стать, вік, тощо. До прикладів неявного збору даних, можна віднести перехід по посиланням, місцезнаходження, тощо.

На рисунку 2.2 схематично зображено приклад надання рекомендацій на основі отриманої інформації про користувача [35].

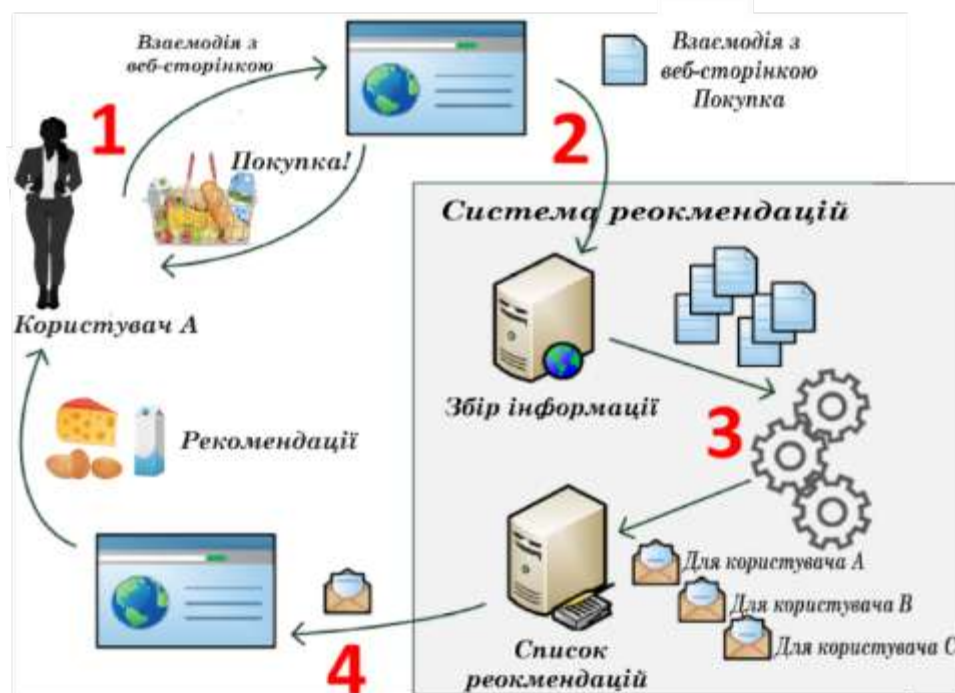


Рисунок 2.2 – Приклад надання рекомендацій користувачу [35]

На сьогодні існує багато видів рекомендацій, для створення яких важливо визначити ціль їх впровадження та тип системи, що дозволить її реалізувати. Для якісних рекомендацій бажано, щоб механізми відповідали таким критеріям як: актуальність, новизна та різноманітність.

2.2 Аналіз рекомендаційних методів

Рекомендаційні системи використовують два види інформації [36]:

- характерна інформація – це інформація про елементи (ключові слова, категорії, тощо) та користувачів (налаштування, профілі, тощо);
- взаємодія між елементами користувача, тобто інформація про кількість покупок, вподобань, рейтинги, тощо [36].

Виходячи з цього можна виділити алгоритми, що використовуються в рекомендаційних системах [36]:

- content-based filtering, що використовує характерну інформацію;
- collaborative filtering, яка базується на взаємодії елементів користувача;
- hybrid поєднує в собі content-based filtering та collaborative filtering, що дозволяє уникнути проблем, які виникають при роботі тільки з одним видом систем [36].

На рисунку 2.3 схематично зображено дані алгоритми, а їх опис, переваги та недоліки наведено в таблиці 2.3 [30, 34].

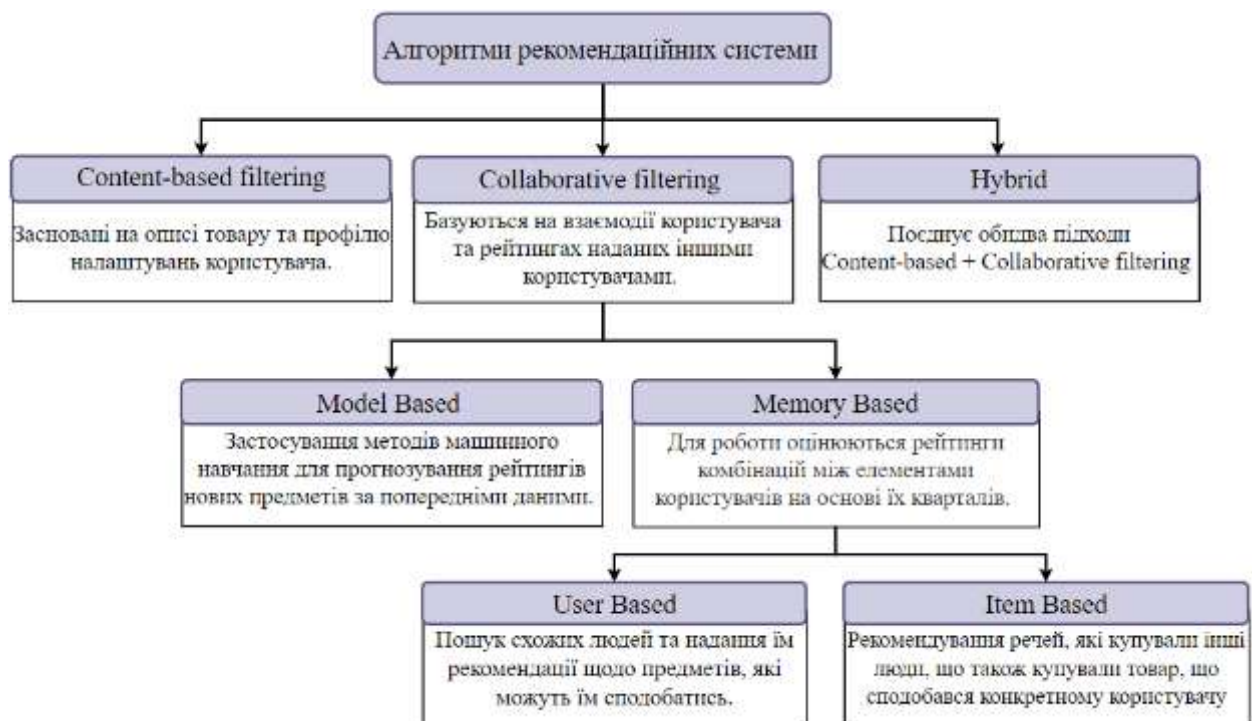


Рисунок 2.3 – Алгоритми, що використовуються в рекомендаційних системах

Таблиця 2.3 – Опис алгоритмів рекомендаційних систем [30, 34]

Назва	Опис	Переваги	Недоліки
Content-based filtering	Рекомендації засновані на даних, зібраних про кожний елемент, який пропонує ресурс та історичної інформації про користувача, що відображає його вибір. В результаті користувачу рекомендуються об'єкти, що схожі на ті, якими він раніше цікавився або купував. Схожість оцінюється виходячи з змісту об'єктів.	Можливість зацікавити користувача відразу, оскільки необхідна інформація тільки про певний елемент. Рекомендація навіть тих об'єктів, що не отримали оцінок від користувачів.	Сильна залежність від предметної області. Зниження точності та кількості корисних рекомендацій. Зниження продуктивності з часом, в зв'язку з тим, що модель не навчається.
Collaborative filtering	Ґрунтується на зборі та аналізі інформації про поведінку, діяльність та на прогнозах, що може сподобатися користувачу. Рекомендації надаються на основі аналізу спільної поведінки та історії оцінок користувача та інших користувачів.	Врахування поведінки інших користувачів з схожими характеристиками, дає хороші та ефективні результати в різних сферах.	Проблема холодного старту (відсутність початкової бази оцінок), розрідженість матриці, масштабованість [36].
Hybrid	Поєднання content-based filtering та collaborative filtering.	Дозволяє об'єднати сильні сторони обох систем та позбавитись більшості недоліків.	Складність розробки.

Алгоритм collaborative filtering ґрунтуються на минулих зафіксованих взаємодіях між користувачами та предметами, такий підхід в більшості випадків надає кращі результати ніж рекомендації, що створені за допомогою алгоритму content-based filtering (рис. 2.4) [29, 36]. Існує два підходи collaborative filtering (рис. 2.3):

- model-based – методи засновані на машинному навчанні та техніці видобутку даних. Перевага – рекомендація більшої кількості елементів для більшої кількості користувачів порівняно з іншими методами [36].

- memory-based, метод в якому оцінюються рейтинги комбінацій між елементами користувачів на основі їх кварталів, для цього існує два підходи: user-based (пошук схожих людей та надання їм рекомендації щодо предметів, які можуть їм сподобатись) та item-based (рекомендування речей, який купували інші люди, що також купували товар, що сподобався конкретному користувачу) [30].



Рисунок 2.4– Робота алгоритму collaborative filtering [30]

Рекомендації отримані за допомогою content-based filtering шукають схожість між предметами, які подобались, купувались, переглядались користувачем в минулому, для надання нових рекомендацій (рис. 2.5).



Рисунок 2.5– Робота алгоритму content-based filtering [30]

Для надання рекомендацій hybrid systems використовують дані про взаємодію елементів користувача та характеристики користувача та елементів (рис. 2.6). Прикладом використання такого підходу може бути Netflix, де рекомендації базуються як на звичаях перегляду та пошуку користувачів (collaborative filtering), так і на рекомендаціях фільмів, що мають схожі між собою характеристики (content-based filtering) [30].

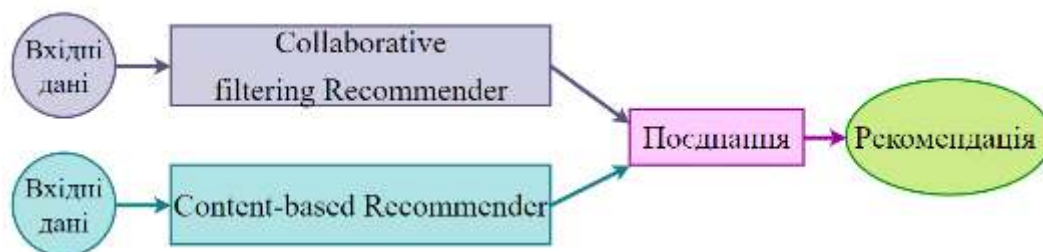


Рисунок 2.6– Робота алгоритму hybrid [30]

Використання даних алгоритмів дозволяють рекомендаційним системам надавати користувачам актуальні, цікаві та персоналізовані рекомендації. Вони також є частиною загальної архітектури рекомендаційних систем, що складається з трьох основних компонентів [37]:

- покоління кандидата – в якості вхідних даних система отримує інформацію про попередню активність користувача та отримує невелику підмножину відео з загальної бази. Існує два загальних підходи до генерації кандидатів: content-based filtering та collaborative filtering;
- зарахування балів – обрані кандидати додатково класифікуються та оцінюються іншою моделлю, наприклад, за певною шкалою;
- перерахунок рейтингу – система враховує додаткові обмеження для забезпечення актуальності та різноманітності рекомендацій. Наприклад, система може видаляти вміст, що точно не сподобався користувачу раніше.

Рекомендаційна система YouTube, також створена на основі загальної архітектури. Так, наприклад, мережа рейтингу, яка зараховує бали, присвоює бал відео відповідно до бажаної цільової функції, використовуючи набір функцій, що описують відео та користувача. В результаті таких дій відео з найбільшою кількістю балів відображається користувачеві, за рейтингом [37].

Отже, зазвичай на початку створення та впровадження рекомендаційних системи за основу береться їх загальна архітектура, що складається з трьох основних компонентів (покоління кандидата, зарахування балів, перерахунок рейтингу) та обирається алгоритм для надання рекомендацій. Таким чином, маючи в якості основи загальну структуру

систем, компанії, що впроваджують в свій програмний продукт систему рекомендацій, починають розвивати, доповнювати, а також адаптувати рекомендаційні системи під конкретні вимоги, що висуваються до системи.

В ході проведеного аналізу алгоритмів рекомендаційних систем для реалізації роботи даного програмного продукту було вирішено використовувати hybrid алгоритм. Дане рішення обумовлено тим, що система буде виконувати дві різні фільтрації, що використовують різні дані. У першому випадку фільтрація буде відбуватись на основі даних, що надходять з YouTube, тому для роботи з ними підходить content-based filtering. А у другому фільтрація опирається на дані отримані в результаті оцінювання якості кожного відео користувачами розроблюваної платформи, саме тому для її реалізації підходить використання collaborative filtering. Тож для виконання поставлених задач було вирішено використовувати hybrid алгоритм, який поєднує в собі content-based filtering та collaborative filtering.

2.3 Аналіз вимог

Розроблювана рекомендаційна система повинна надавати користувачам допомогу під час пошуку необхідного їм відео, надавати можливості для визначення його якості шляхом оцінювання.

Для виконання поставлених завдань система повинна виконувати такі функції:

- зареєструвати чи авторизувати користувача в системі;
- надати користувачу інформацію для реєстрації в системі;
- надати функціонал для підтвердження використання даних профілю YouTube;
- надати користувачу доступ до інформації в його особистому кабінеті та можливість її редагування;
- надати користувачу список відео, який сформований в результаті його пошукового запиту;

- надати можливість обрати відео з отриманого списку та переглянути його;
- надати користувачу можливість переглядати список обраного відео, а також додавати та видаляти відео з нього;
- забезпечити користувача засобами для оцінювання переглянутого відео.

Вхідними даними для роботи системи є оцінки користувача та дані про кожне відео, що надходять з системи YouTube.

Вихідними даними системи є список відео, який користувач отримує в результаті виконання пошуку та власне саме відео, що містить необхідну користувачеві інформацію.

Використання програмного продукту здійснюється відповідно до розробленої для нього експлуатаційної документації, яка відповідає стандартам і містить всю необхідну інформацію для освоєння та експлуатації програмного продукту.

2.4 Висновки до розділу 2

На сьогодні практично всі як великі так і малі компанії впроваджують та використовують на своїх вебсайтах та платформах рекомендаційні системи. Їх використання обумовлено прагненням затримати користувача на сервісі та надати йому послуги краще ніж конкуренти, запропонувати саме те, що потрібно. В результаті чого компанія зможе примножити свою аудиторію, продати більше послуг, а значить і заробити коштів більше.

В даному розділі розглянуто причини впровадження рекомендаційних систем, наведено приклади відомих компаній, що їх використовують, таких як Amazon, Netflix, Youtube, Linkedin, Facebook. Описано базовий шаблон роботи рекомендаційних систем, розглянуто способи збору інформації для роботи системи. А також проаналізовано алгоритми, що обробляють отримані дані з

метою надати якісні рекомендації, наведено переваги та недоліки їх використання. В результаті чого обрано hybrid systems алгоритм для роботи.

Також в розділі було визначено основні вимоги до майбутньої системи її вхідні та вихідні дані. Такий аналіз дозволив чітко визначити масштаби розробки даного програмного продукту.

3 ПРОЄКТУВАННЯ СИСТЕМИ

3.1 Проєктування баз знань

Базою знань (БЗ) це певний набір закономірностей, що встановлюються між інформацією про досвід та знання людини в певній переметній області, що вводиться та виводиться [38]. БЗ призначена для керування знаннями та для зберігання даних про предметну область в структурованому вигляді. В базі знань також зберігаються правила та набори інструкцій, завдяки яким можна отримати нові факти на основі отриманих знань [39].

Знаннями називають закономірності, принципи, зв'язки чи закони, що отримуються в результаті практичної діяльності, власного досвіду в межах предметної області [38]. А даними – інформацію, що отримана в результаті спостережень чи змін окремих властивостей, вона також характеризує об'єкти (процеси, явища) предметної області, а також їх властивості (тобто факти) [39].

В розроблюваній системі всі правила було вирішено розділити на дві бази знань: база знань фільтрації та база знань сортування відео. Оскільки частина правил відноситься до тих, що містять інформацію, яка допоможе користувачу знайти необхідне відео, а друга – виконати сортування отриманого списку та надати першими відео з найвищою якістю.

До першої бази знань входять правила для фільтрації списку отриманого відео: джерела завантаження відео (медичний університет, лікарні, медичні працівники, інші), категорії відео (лекції, інструктаж, анімований відеоряд, короткотривалі відео, тощо), ключові слова.

До другої бази знань відносяться правила, на основі яких відбувається сортування списку відео: опис відео (наявний, відсутній), дані YouTube (число позитивних та негативних відмоток, число переглядів та коментарів) а також правила, що засновані та оцінках користувачів (якість медичної практики, якість наданої інформації, якість опису та наявність практичного використання) та оцінюються за шкалою від 1 до 5.

На рисунках 3.1 та 3.2 в ієрархічному вигляді представлено отримані бази знань.

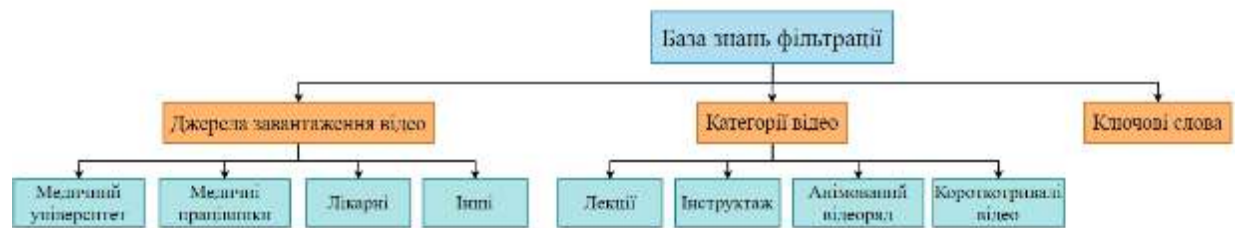


Рисунок 3.1 – База знань фільтрації відео



Рисунок 3.2 – База знань сортування відео

Використання двох окремих баз знань, дозволяє надавати користувачам більш якісне відео. На рисунку 3.3 зображено загальну архітектуру рекомендаційної системи.

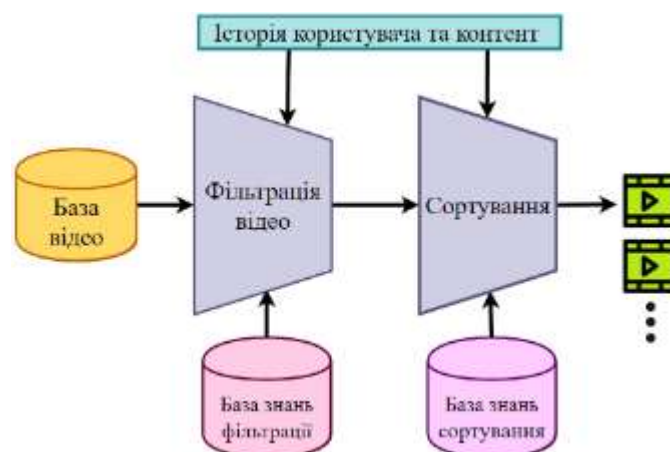


Рисунок 3.3 – Архітектура рекомендаційної системи

Розроблена архітектура рекомендаційної системи складається з шести основних компонентів, опис яких наведено в таблиці 3.1.

Таблиця 3.1 – Опис основних компонентів архітектури рекомендаційної системи

Назва компоненту	Опис
База відео	Список відео, який сформовано на основі отриманої інформації з платформи YouTube та на історіях пошуку користувачів на ній.
Історія користувача та контент	Інформація про дії користувача на розроблюваній платформі (оцінки якості, реакція на відео) та дані про відео отримані з платформи YouTube.
База знань фільтрації	Містить правила, щодо фільтрації даних про відео отриманих з YouTube.
База знань сортування	Містить правила, для сортування інформації про відео.
Фільтрація відео	Етап, який використовує алгоритм content-based filtering. В якості вхідних даних інформацію про відео, яка зберігається відповідно до правила описаних в базі знань фільтрації.
Сортування	Етап, який використовує алгоритм collaborative filtering, який опирається на правила прописані в базі знань сортування. В якості вхідних даних використовує список відфільтрованих відео на попередньому етапі.

Таким чином для надання якісних рекомендацій розроблювана система використовує дві бази знань, в яких міститься перелік правил для збереження даної інформації та роботи алгоритму hybrid, який поєднує в собі content-based filtering та collaborative filtering, що використовуються для виконання етапів фільтрації та сортування відповідно.

3.2 Проектування загальної архітектури системи

Більшість вебсайтів на сьогодні базуються на моделі клієнт-серверної архітектури, в якій вебсервер розміщує, доставляє та керує ресурсами, які споживає клієнт. Клієнтами вважають пристрої кінцевих користувачів, що підключені до сервера через інтернет з'єднання [40, 41].

В залежності від типу сайту розрізняють архітектуру вебсервера для динамічного або для статичного сайту. Статичним називають сайт, що на запит користувача повертає той же самий чітко визначений та закодований вміст із серверу. Наприклад, коли користувач переходить на певну сторінку, то його браузер відправить HTTP "GET" запит із значенням URL-адреси. Сервер обробить запит, та поверне HTTP-відповідь. На рисунку 3.4 зображено архітектуру такого вебсервера [42].

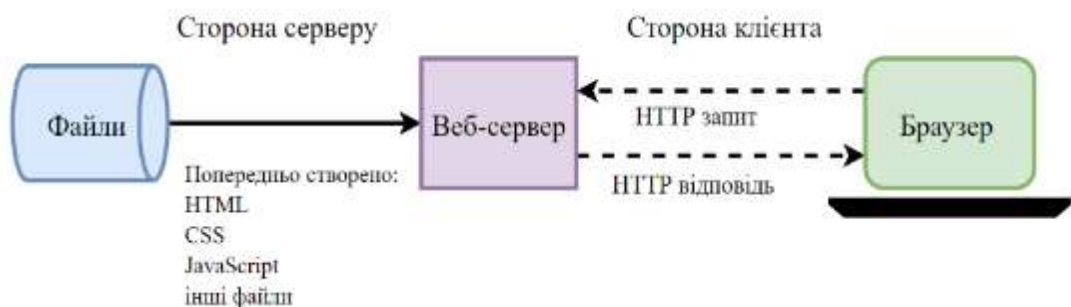


Рисунок 3.4 – Архітектура вебсервера для статичного сайту [42]

Динамічним сайтом називають сайт чи застосунок, де певний вміст відповіді генерується динамічно та лише за потреби. HTML-сторінки на таких сайтах, зазвичай, створюються шляхом вставки потрібних даних з БД в HTML шаблон. В архітектурі вебсервера для такого типу сайту обробка запитів до динамічних ресурсів здійснюється, як показано на рисунку 3.5 [42]. Всі запити пересилаються на обробку до серверу (показано на схемі як вебдодаток) (2). У випадку динамічного запиту сервер інтерпретує запит, зчитує необхідну інформацію з бази даних (3), поєднує отримані дані з HTML-шаблонами (4) і повертає відповідь, що містить сформований HTML (5, 6) [42].

Використання динамічних сайтів є набагато ефективнішим при зберіганні, обробці та роботі з великою кількістю контенту, ніж у випадку коли для цих цілей використовують статичні вебсайти [42].

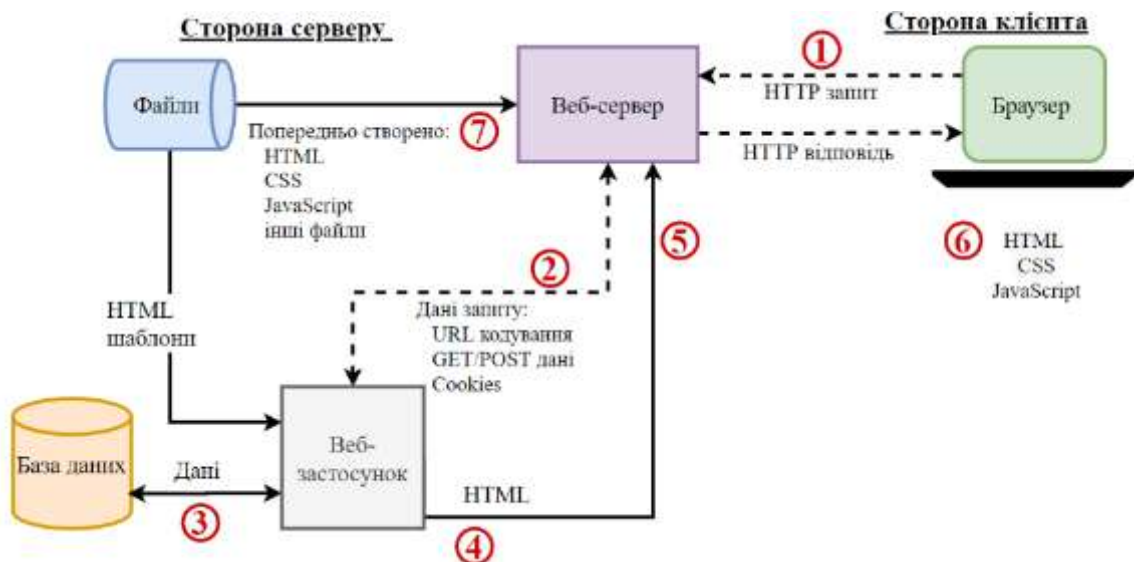


Рисунок 3.5 – Архітектура вебсервера для динамічного сайту [42]

Таким чином для виконання поставлених задач розроблювати саме вебзастосунок динамічного типу є найкращим варіантом. Розроблюваний продукт складається з двох основних частин: серверної та клієнтської. Серверна частина міститиме досить складний та відносно довгий у плані часу виконання функціонал, тому було прийнято рішення розділити відповідальність та створити окремі сервіси для серверної та клієнтської частин. Схема взаємодії між частинами програмного продукту зображено на рисунку 3.6.



Рисунок 3.6 – Схема взаємодії компонентів системи.

Компоненти зображені на рисунку умовно можна розділити на основні групи, а саме за клієнтською частиною, серверною частиною, рекомендаційною системою та модулем з асинхронними задачами. Рекомендаційну систему було поміщено в модуль асинхронних задач, оскільки її функціонал використовується в даному модулі. Розділення програмного продукту на частини, обумовлено логічним розділенням відповідальності, що надає гнучкості у оновленні компонентів системи.

В розроблюваному продукті наявні складні обчислення, які вимагають великих часових та ресурсних затрат. Для забезпечення можливості витримати високе навантаження та в цілях збереження даних під час довгих операцій було прийнято рішення розбити ці обчислення на окремі модулі, що виконуватимуться паралельно один від одного, тобто асинхронно, та не будуть блокувати основний процес. Саме тому додано спеціальний модуль з асинхронними задачами. На рисунку 3.7 зображено більш детальна схема взаємодії компонентів системи.

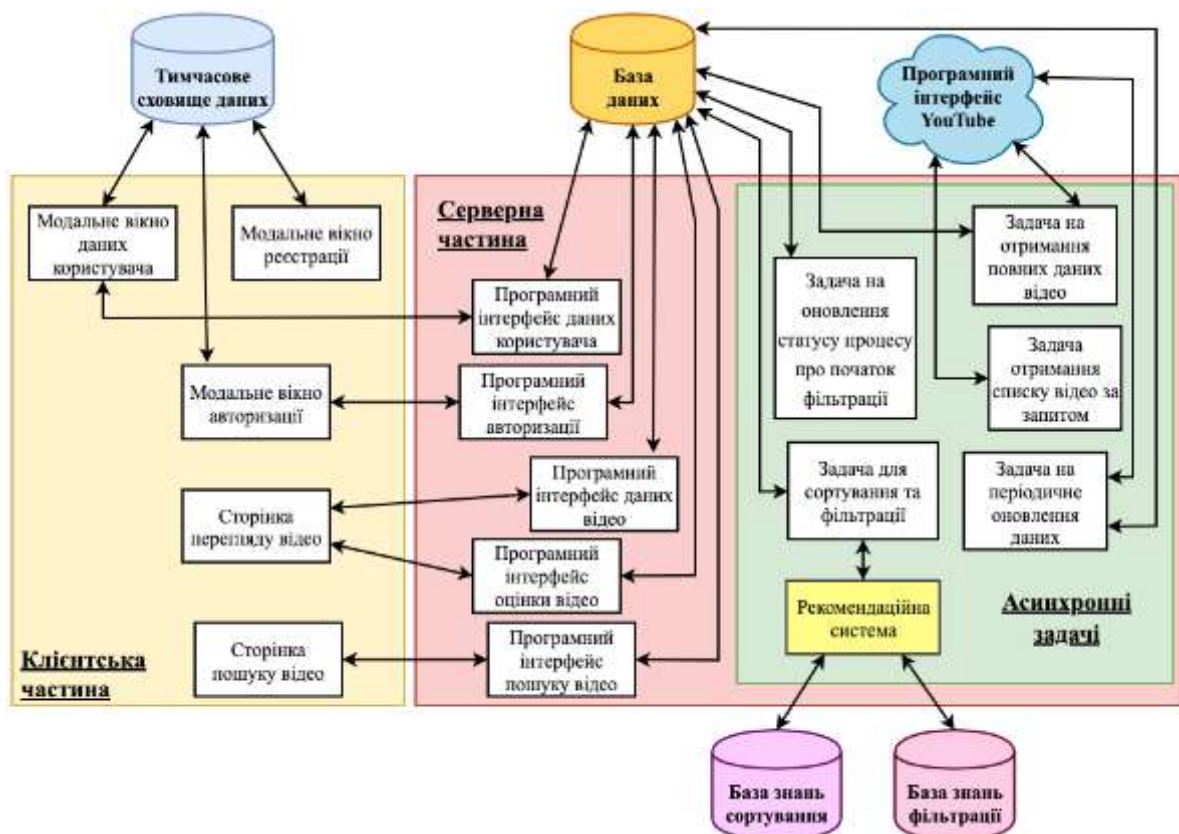


Рисунок 3.7 – Розширена схема взаємодії компонентів.

Взаємодія між клієнтською частиною та серверною відбувається шляхом передачі даних за допомогою протоколів зв'язку TCP/IP оскільки для розроблюваного продукту, клієнтська частина завжди повинна отримувати відповідь від серверної частини.

Взаємодія між серверною частиною та асинхронними задачами відбувається за допомогою бази даних. Асинхронні задачі будуть постійно перевіряти базу даних та оновлювати дані в ній. Дане рішення дозволить забезпечити витримку високого навантаження та запобігти втратам даних під час непередбачуваних ситуацій. Опис компонентів системи наведено в таблицях 3.2 – 3.4.

Таблиця 3.2 – Опис компонентів серверної частини

Назва компоненту	Опис компоненту
Програмний інтерфейс авторизації	Програмний інтерфейс, що відповідає за авторизацію в системі існуючих користувачів.
Програмний інтерфейс даних користувача	Програмний інтерфейс, що відповідає за дані користувачів, а саме: отримання актуальних, або їх оновлення.
Програмний інтерфейс даних відео	Програмний інтерфейс, що надає певні дані про конкретне відео за запитами
Програмний інтерфейс оцінки відео	Програмний інтерфейс, що надає можливість користувачам розроблюваної платформи зробити оцінки обраного відео.
Програмний інтерфейс пошуку відео	Програмний інтерфейс, що запускає механізм пошуку рекомендацій за запитом, що надходить. Також відповідає за фільтрацію відео на основі правил з баз знань.

Таблиця 3.3 – Опис компонентів клієнтської частини

Назва компоненту	Опис компоненту
Модальне вікно даних користувача	Модальне вікно, що зберігатиме дані користувача з розроблюваної системи.
Модальне вікно реєстрації	Модальне вікно, що надсилатиме запити на реєстрацію користувачів до серверу.
Модальне вікно авторизації	Модальне вікно, що надсилатиме запити на авторизацію до серверу.
Сторінка перегляду відео	Сторінка, де відображатиметься відео отримане за запитом з серверної частини.
Сторінка пошуку відео	Головна сторінка, на якій буде поле для пошуку відео. Вона надсилатиме запит на пошук відео до серверної частини.

На даному рисунку зображено кроки для отримання рекомендацій відео за запитом користувача. Спочатку надсилається запит на пошук списку відео з клієнтської на серверну частину (програмний інтерфейс пошуку відео). Програмний інтерфейс зберігає дані процесу в БД, такі як унікальний ідентифікатор процесу пошуку, текст запиту, вхідні дані для фільтрацій, що надійшли з клієнтської частини, та надсилає відповідь до клієнтської частини. Відповідь містить оповіщення, що процес пошуку розпочато та надсилає унікальний ідентифікатор поточного пошуку. Задача отримання списку відео за запитом постійно перевіряє БД на наявність нових даних вхідних запитів. Після отримання даних запитів, формує запит та надсилає його до програмних інтерфейсів YouTube. Відповідь від програмних інтерфейсів YouTube містить лише частину повного списку даних про відео та посилання на наступну частину та попередню, якщо така існує. Тому в БД надсилається посилання на отримання наступної частини відео.

На наступному кроці задача на отримання повних даних відео перевіряє БД на наявність порожніх даних та нових відео, після чого надсилає запит до програмних інтерфейсів YouTube на отримання даних, що не вистачає. Далі задача на оновлення статусу процесу про початок фільтрації перевіряє статуси відео, згрупованих за процесами. Якщо всі відео обраного процесу перевірені, тобто усі дані для них було зібрано з YouTube, тоді даний процес маркується як готовий для фільтрації та сортування. На наступному кроці задача фільтрує та сортує дані для кожного унікального процесу згідно з правилами з БЗ фільтрації та сортувань.

Після, процес пошуку буде помічено як завершений, і клієнтська частина зможе отримати список відео надіславши запит до програмного інтерфейсу пошуку відео. Завдяки даному рішенню кожна частина міститиме свою окрему логіку, що в майбутньому дозволить облегшити оновлення системи. Також з'являється можливість у майбутньому горизонтально та вертикально масштабувати серверну частину, щоб зменшити навантаження з неї та тим самим зменшити час отримання відповіді.

3.3 Проектування структури СКБД

Опис сутностей та атрибутів бази даних наведений в таблиці 3.5.

Таблиця 3.5 – Опис сутностей та атрибутів бази даних

Сутність	Атрибут	Назва атрибуту в БД	Опис сутності
1	2	3	4
Відео videos_video	id	id	Містить всю наявну інформацію про відео.
	назва	title	
	посилання на сутність категорії	category_id	
	канал	channel_id	
	опис	description	
	дата оновлення	updated_at	
	дата створення	created_at	
	статус	status	
Дані YouTube videos_youtubedata	посилання на сутність дані youtube	youtube_data_id	Дані про відео, що стосуються системи YouTube
	id	id	
	спеціальний хеш з системи YouTube	etag	
	дата публікації	pub_date	
	індекс відео в системі YouTube	video_hash	
	посилання на сутність превью зображення	image_preview_id	
	число негативних оцінок	negative_mark_number	
	число позитивних оцінок	positive_mark_number	
	число коментарів	comment_count	
Прев'ю зображення videos_imagepreview	число переглядів	view_count	Містить зображення для відео.
	id	id	
	посилання	link	
	ширина	width	
Канал videos_channel	висота	height	Інформація про канал на YouTube, звідки взяте відео.
	id	id	
	індекс каналу в системі YouTube	youtube_id	
	назва	name	
	країна	country	
	опис	description	
Джерело videos_channelsource	ключові слова	keywords	Ресурс, який завантажив відео на свій канал YouTube
	id	id	
	канал	channel_id	
	джерело	source_id	

Продовження таблиці 3.5

1	2	3	4
Оцінки користувачів videos_usermark	id	id	Містить інформацію про оцінки користувачів виставлені на розроблюваній платформі. Може містити значення, що дорівнюють 0. У такому випадку вони не будуть враховуватися в середньому значенні.
	якість інформації	information_quality	
	якість медичної практики	medical_practice_quality	
	якість опису відео	description_quality	
	якість практичного застосування	practical_usage_availability	
	посилання на стуність відео	video_id	
	посилання на сутність користувача	user_id	
Тег filters_tag	id	id	Сутність для збереження даних про теги.
	ім'я	name	
Тег-відео videos_tagvideo	id	id	Являє собою реалізацію зв'язку багато-до-багатьох між сутностями тег та відео
	посилання на сутність тег	tag_id	
	посилання на сутність	video_id	
Категорія filters_category	id	id	Сутність для збереження даних про категорії
	ім'я	name	
	ключ з системи YouTube	youtube_id	
	спеціальний хеш з системи YouTube	etag	
Категорія - відео videos_categoryvideo	id	id	Являє собою реалізацію зв'язку багато-до-багатьох між сутностями категорія та відео
	посилання на сутність категорія	category_id	
	посилання на сутність	video_id	
Обрані відео videos_featured	id	id	Сутність, що зберігатиме дані про відео, що було додано до списку обраних для конкретного користувача.
	дата оновлення	updated_at	
	дата створення	created_at	
	посилання на відео	video_id	
	посилання на користувача	user_id	
Користувач auth_user	id	id	Містить детальну інформацію про користувача
	ім'я	first_name	
	прізвище	last_name	
	псевдонім	username	
	роль	is_superuser	
	статус	is_staff	
	дата створення	date_joined	
	дата останнього входу	last_login	
	пошта	email	
	пароль	password	
	активованість користувача	is_active	

Продовження таблиці 3.5

1	2	3	4
Дані авторизації в системі YouTube accounts_youtubecredentials	id	id	Містить у собі інформацію про авторизацію користувача у системі YouTube
	токен для аутентифікації в YouTube	token	
	токен для оновлення аутентифікації	refresh_token	
	посилання для генерації токenu	token_uri	
	ключ користувача під час процесу авторизації в системі YouTube	client_id	
	секретні дані користувача	client_secret	
	дата до якої діє токен	expiry	
	посилання на сутність користувача	user_id	
Токен knox_authtoken	digest	digest	Містить зашифровану інформацію про дані користувача та використовується для авторизації в системі.
	salt	salt	
	час створення	created	
	посилання на користувача	user_id	
	дата до якої діє токен	expiry	
	токен для авторизації в системі	token_key	
Процес process_process	id	id	Являє собою сутність запиту, для пошуку відео за критеріями. Процес повинен бути активованим, щоб по ньому розпочався пошук. Може мати статуси: «очікує на отримання базових даних», «очікує на отримання повних даних», «очікує на фільтрацію» «дані успішно сформовані» та «помилка в процесі виконання».
	статус	status	
	посилання на поточну групу відео з YouTube	youtube_video_group	
	активованість процесу	active	
	текст пошуку	search_data	
	деталі можливих помилок системи	invalid_msg	
	посилання на наступний процес	next_process	
	посилання на попередній процес	prev_process	
	дата оновлення даних	updated_at	
	дата створення	created_at	
	посилання на сутність користувача	user	
Процес-відео process_processvideo	id	id	Додаткова таблиця для реалізації зв'язку багато-до-багатьох. Завдяки цій сутності можемо створювати групи відео для обраних процесів.
	порядковий номер відео у групі для обраного процесу	video_order	
	посилання на сутність процесу	process_id	
	посилання на сутність відео	video_id	

Продовження таблиці 3.5

1	2	3	4
Процес-категорія process_processcategory	id	id	Являє собою реалізацію зв'язку багато-до-багатьох між сутностями тег та відео
	посилання на сутність процес	process_id	
	посилання на сутність категорії	category_id	
Процес-джерело process_ProcessSource	id	id	Являє собою реалізацію зв'язку багато-до-багатьох між процес тег та джерело
	посилання на сутність процес	process_id	
	посилання на сутність джерело	source_id	
Процес-тег process_processtag	id	id	Являє собою реалізацію зв'язку багато-до-багатьох між сутностями процес та тег
	посилання на сутність процес	process_id	
	посилання на сутність тег	tag_id	
Джерело filters_source	id	id	Сутність для збереження даних про джерела
	ім'я	name	

В структурі даних наявні ситуації, коли для одного процесу може бути декілька відео, категорій, джерел, тегів, так само як і для відео, категорій, джерел та тегів може бути кілька процесів. Така сама ситуація між сутностями відео та джерело, категорія та тег. Тому вирішено використовувати зв'язок багато-до-багатьох за допомогою допоміжної таблиці в БД, як зображено на рисунку 3.9. А на рисунку 3.10 представлено схему бази даних.

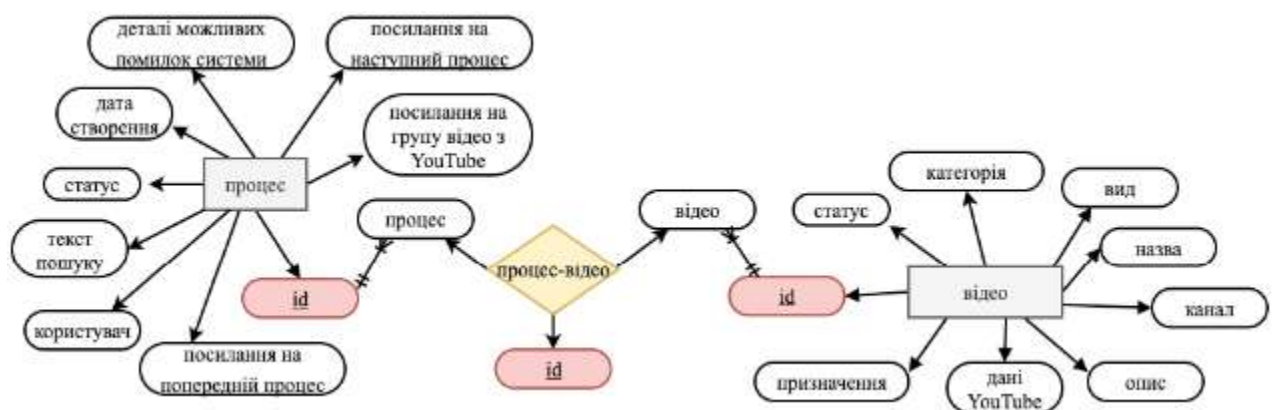


Рисунок 3.9 – Зв'язок багато-до-багатьох між сутностями процес та відео.

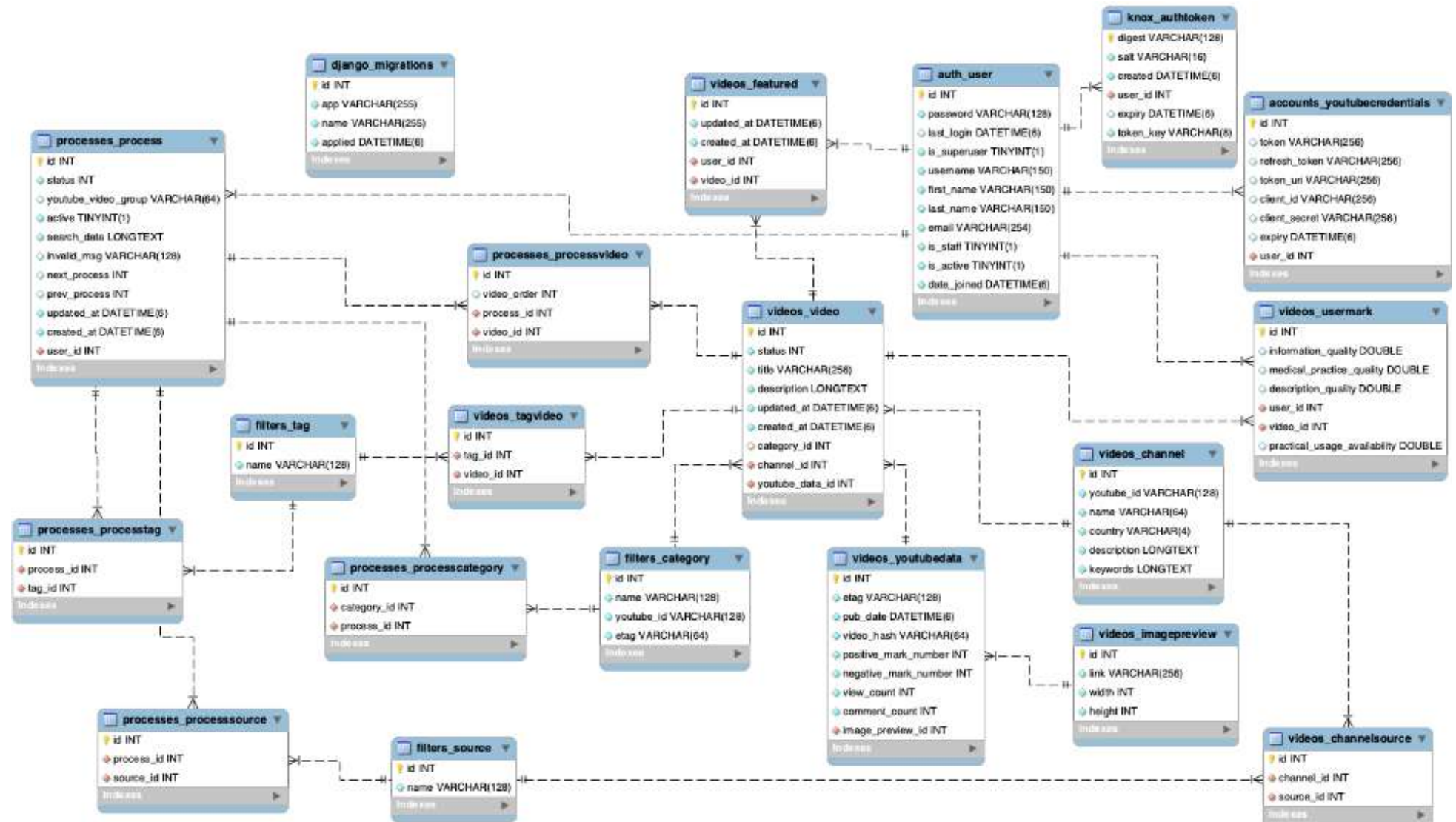


Рисунок 3.10 – Схема бази даних

Було спроектовано структуру СКБД, що дозволить легко та зручно маніпулювати даними, підтримувати їх актуальність. Дана структура була спроектована за стандартами нормальних форм.

3.4 Проектування прототипу графічного інтерфейсу

Клієнтська частина – це все що відображається та виконується на екрані користувача. Тобто вона складається з зображень, таблиць, тексту, форм тощо, а її розробка полягає в розміщенні всіх необхідних елементів на сторінці та надання їм необхідних форм та властивостей, а також забезпечення виконання ними потрібних функцій. Зазвичай процедура розробки клієнтської частини вебзастосунку складається з таких основних етапів, як створення прототипу зовнішнього вигляду сторінок їх верстання та програмування.

Прототипом сайту називають базовий макет сайту, який наглядно показує розташування всіх його основних елементів та функцій. Таки підхід дозволяє виконати синхронізацію між уявленням замовника та розробників про те, як саме повинен виглядати кінцевий результат. Головною метою його створення є економія часу, тому він розробляється на ранніх стадіях проєкту. Такий підхід дозволяє визначити кінцеву ціль розробки та виключити частково або повністю потребу в переробці проєкту в середині або кінці роботи. Також це надає можливість більш точно оцінити об'єм та термін розробки проєкту, його бюджет. В ході розробки протопопу складається представлення про майбутнє проєкту та розуміння його розвитку та перспектив в майбутньому [43, 44].

Створення прототипу графічного інтерфейсу користувача вебзастосунку зазвичай може складатись з трьох основних етапів. Спершу необхідно визначити кількість сторінок, які буде мати розроблюваний вебзастосунок та продумати шлях користувача між ними. Далі створити модульні сітки для сторінок сайту, які представляє собою візуальне розбиття сторінки на певні зони, такі дії дозволяють структурувати зміст сайту. Потім

необхідно визначити основні компоненти, що будуть відображатись на сторінці, їх функції та місце на модульній сітці.

Даний розроблюваний програмний продукт буде складатись з трьох сторінок:

- головна сторінка – призначена для пошуку відео та відображення його результатів;
- сторінка обраного відео – відображає список відео, які користувач додав до категорії «Обране», на яку можна перейти за посиланням в профілі користувача;
- сторінка перегляду – призначена для перегляду та оцінювання обраного відео, для переходу на яку необхідно натиснути на обране відео.

Наступним кроком є розробка модульних сіток для кожної сторінки. На рисунках 3.11-3.13 зображено результат їх проектування.



Рисунок 3.11 – Модульна сітка головної сторінки

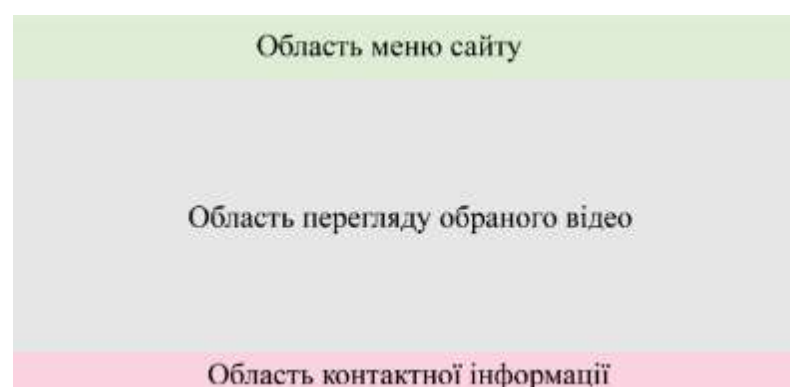


Рисунок 3.12 – Модульна сітка для сторінки з обраним відео



Рисунок 3.13 – Модульна сітка для сторінки перегляду відео

Області на модульних сітках на сторінках для наочності зафарбовані кожна в свій колір, які дублюються на рисунках 3.14-3.16, де представлено розміщення компонентів на сторінках.

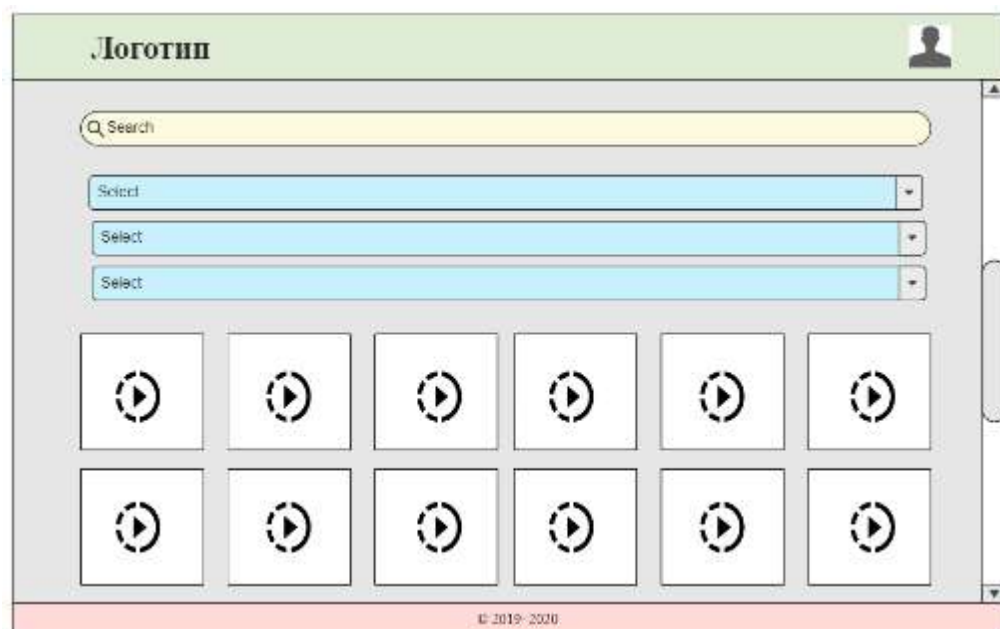


Рисунок 3.14 –Розміщення основних компонентів на головній сторінці



Рисунок 3.15 –Розміщення основних компонентів на сторінці перегляду відео

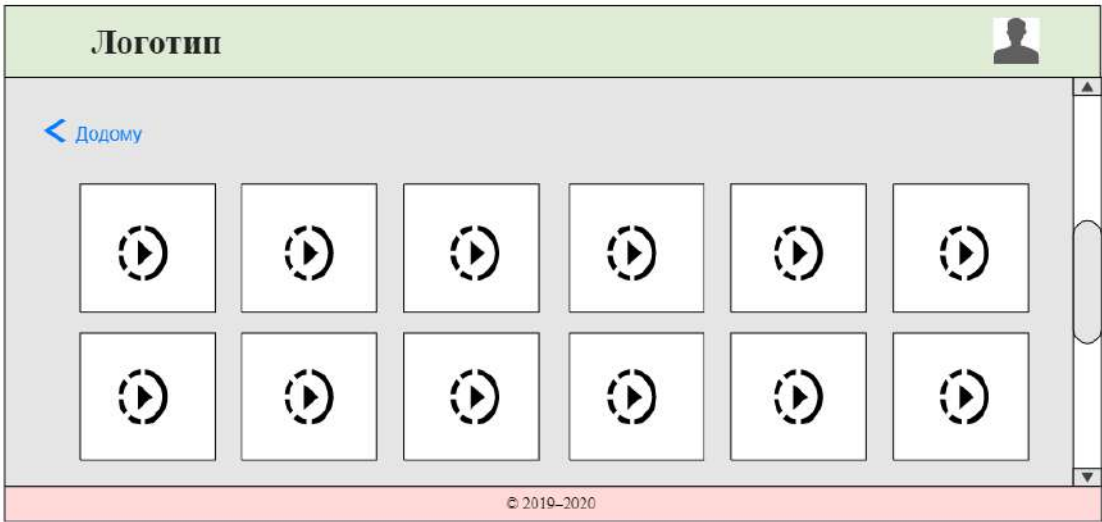


Рисунок 3.16 –Розміщення основних компонентів на сторінці обраного відео

В таблиці 3.6 наведено опис основних компонентів та функції, які вони виконують, а також їх розташування на модульній сітці.

Таблиця 3.6 – Опис основних компонентів на сторінці

Назва компоненту	Функція	Сторінка сайту
Логотип	Відображає назву сервісу, та дозволяє повернутись на головну сторінку.	Наявні на всіх сторінках
Кнопка профілю користувача	Викликає модальне вікно, що надає можливості авторизації і реєстрації в системі та перегляду інформації з профілю користувача.	
Поле пошуку	Призначене для пошуку необхідного відео.	Головна
Іконки відео	Призначені для відображення на сторінці знайдених відео та їх вибору для перегляду.	Головна, сторінка обраного
Список фільтрів	Призначений для швидкого пошуку необхідного відео.	Головна
Список параметрів оцінки якості	Список призначений для оцінки якості обраного користувачем відео.	Перегляду відео
Відео програвач	Призначений для відтворення обраного користувачем відео.	
Таблиця	Містить наявну інформацію про обране відео.	
Кнопка додому	Для швидкого переходу з сторінки з обраним відео на головну сторінку сайту.	Сторінка обраного

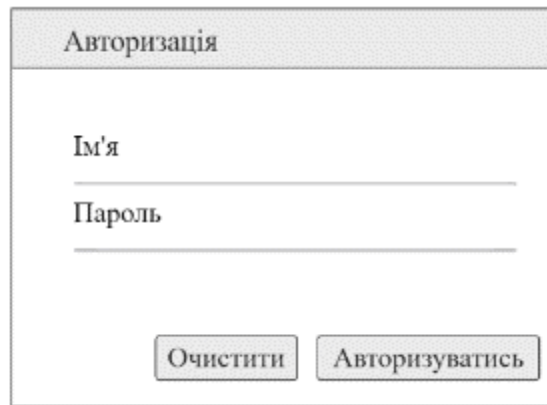
Реєстрація та авторизація користувачів на сайті здійснюється за допомогою модульних вікон, попередній схематичний вигляд яких разом з основними компонентами зображено на рисунках 3.17 та 3.18.

Реєстрація

Інформація для реєстрації на платформі

Зрозуміло

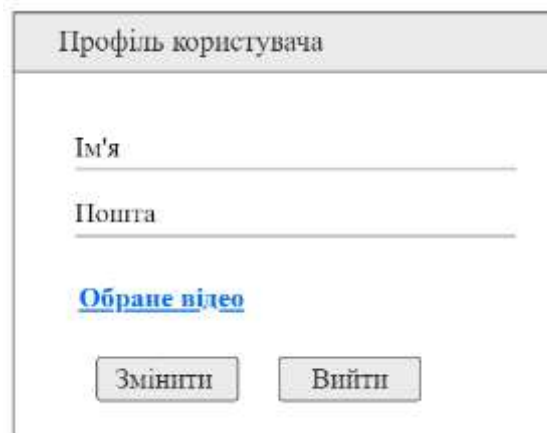
Рисунок 3.17 – Модальне вікно для реєстрації користувача



The image shows a modal window titled 'Авторизація' (Authorization). It contains two input fields: 'Ім'я' (Name) and 'Пароль' (Password). Below the input fields are two buttons: 'Очистити' (Clear) and 'Авторизуватись' (Authorize).

Рисунок 3.18 – Модальне вікно для авторизації користувача

Після вдалої авторизації в системі користувач може перейти до власного профілю, який відображається також у вигляді модального вікна (рис. 3.19). В відкритому вікні можна переглянути чи змінити власну інформацію або ж за допомогою посилання перейти на сторінку для перегляду відео, яке користувач додав до списку «Обране».



The image shows a modal window titled 'Профіль користувача' (User Profile). It contains two input fields: 'Ім'я' (Name) and 'Пошта' (Email). Below the input fields is a blue link labeled 'Обране відео' (Selected video). At the bottom are two buttons: 'Змінити' (Change) and 'Вийти' (Exit).

Рисунок 3.19 – Модальне вікно профілю користувача

Розробка прототипу дозволяє структурувати зміст сторінок вебзастосунку, визначити основні компоненти їх розміщення та функції. Розробка такого прототипу є певним скелетом, який наглядно візуалізує структуру інтерфейсу, а також дозволяє, за потреби, ввести правки з мінімальними витратами на ресурсах та зусиллях.

3.5 Висновки до розділу 3

Важливим етапом розробки будь-якого програмного продукту є проєктування всієї системи, саме тому його детально розглянуто в даному розділі. Для розроблюваного програмного продукту на даному етапі було спроектовано: дві бази знань, структуру бази даних, загальну архітектуру, прототип графічного інтерфейсу.

В розроблюваній системі всі правила було вирішено розділити на дві бази знань: база знань фільтрації та база знань сортування відео. Оскільки частина правил відноситься до тих, що містять інформацію, яка допоможе користувачу знайти необхідне відео, а друга – виконати сортування отриманого списку та надати першими відео з найвищою якістю.

До першої бази знань входять правила для фільтрації списку отриманого відео: джерела завантаження відео (медичний університет, лікарні, медичні працівники, інші), категорії відео (лекції, інструктаж, анімований відеоряд, короткотривалі відео, тощо), ключові слова.

До другої – відносяться правила, на основі яких відбувається сортування списку відео: опис відео (наявний, відсутній), дані YouTube (число позитивних та негативних відмоток, число переглядів та коментарів), а також правила, що засновані та оцінках користувачів (якість медичної практики, якість наданої інформації, якість опису та наявність практичного використання) та оцінюються за шкалою від 1 до 5. Використання двох окремих баз знань, дозволяє надавати користувачам більш якісне відео.

Для надання якісних рекомендацій розроблювана система використовує алгоритм hybrid, який поєднує в собі content-based filtering та collaborative filtering, що використовуються для фільтрування та сортування відео відповідно.

Загальна архітектура системи розділена на дві основні частини: серверну та клієнтську, також для кожної з частин створено окремий сервер,

оскільки серверна частина міститиме досить складний та відносно довгий у плані часу виконання функціонал.

Архітектура складається з десяти основних компонентів, які умовно можна розділити на основні групи за основними компонентами, а саме за клієнтською частиною, серверною частиною, рекомендаційною системою та модулем з асинхронними задачами. Також у розділі наведено їх опис, а оскільки програмний продукт міститиме багато запитів, то наведено схему взаємодії компонентів в запиті на отримання списку відео.

Для зберігання всієї необхідної для роботи інформації спроектовано структуру СКБД, що дозволить легко та зручно маніпулювати даними, підтримувати їх актуальність.

Спроектовано базовий макет сайту, що дозволило наглядно візуалізувати структуру інтерфейсу, показати розташування основних елементів, а також визначити їх функцій.

4 ВИБІР ІНСТРУМЕНТАРІЮ ДЛЯ РОЗРОБКИ

4.1 Порівняння та вибір СКБД

Системи рекомендацій збирають дані про користувачів, товари, відео, тощо і чим більше даних, тим краще та точніше будуть працювати алгоритми з надання рекомендацій. У свою чергу велика кількість даних призводить до проблеми їх зберігання, саме тому для її вирішення використовують бази даних (БД). БД розділяються на два основних типи: реляційні та нереляційні.

В реляційних базах даних дані є структурованими та їх можна візуально відобразити у вигляді таблиць, що складаються з стовпців та рядків. Де стовпці – атрибути даних, а рядок таблиці – це запис із унікальним ідентифікатором. Такий підхід дозволяє легко отримати інформацію за допомогою запитів та організовувати дані простим способом [45, 46].

В нереляційних базах даних дані в них є неструктурованими, що робить БД такого типу більш гнучкими. Дані можуть зберігатися у вигляді пари ключ/значення або як документи JSON. Такий тип БД також є інтуїтивно зрозумілими, швидкими та ефективними, це дозволяє виконувати запити не використовуючи SQL. Нереляційні БД підходять якщо необхідно зберігати великі об'єми даних, які потрібно масштабувати [46, 47].

Системи керування базами даних (СКБД) називають комплексом програмного забезпечення, що надає можливості створення, збереження, оновлення та пошуку інформації в БД з контролем доступу до даних. Обираючи СКБД слід враховувати фінансові витрати на придбання системи, необхідного обладнання, розробку необхідного програмного забезпечення на основі цієї системи, а також навчання персоналу.

За даними видавництва DB-Engines станом на червень 2020 року в п'ятірку найпопулярніших СКБД входять: Oracle, MySQL, Microsoft SQL Server, PostgreSQL, MongoDB (рис. 4.1) [48]. Дана статистика створюється на базі методу, який для її створення збирає та усереднює такі дані як [49]: загальний інтерес до системи та кількість згадок про неї на вебсайтах; частота

технічних дискусій щодо системи; кількість пропозицій про роботу, в яких згадується система; кількість профілів у професійних мережах, в яких згадується система; актуальність у соціальних мережах.

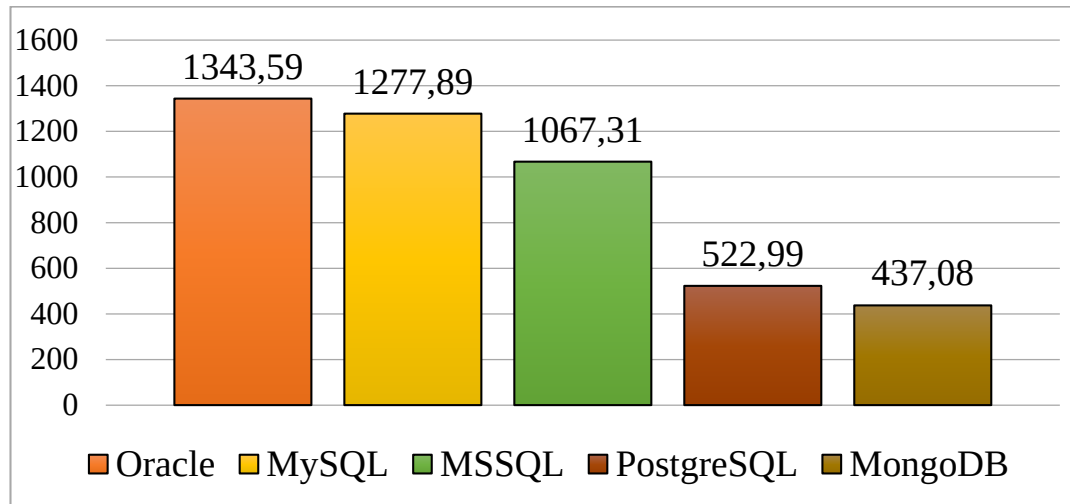


Рисунок 4.1 – Рейтинг популярності СКБД

В таблиці 4.1 представлено порівняння п'ятірки найпопулярніших СКБД. Оцінювання параметрів якості документації, підтримки виробником та суспільством відбувалось за шкалою від 1 до 10, де 1 – погано, а 10 – добре.

Таблиця 4.1 – Порівняння СКБД

Параметр	Oracle	MySQL	Microsoft SQL	PostgreSQL	MongoDB
Тип БД	Реляційна	Реляційна	Реляційна	Реляційна	Нереляційна
Підтримка ОС	Linux, Microsoft Windows	Linux, Microsoft Windows, MacOS	Linux, Microsoft Windows	Linux, Microsoft Windows, MacOS,	Linux, Microsoft Windows, MacOS,
Використання ресурсів	Дуже високе	Економічне	Високе	Середнє	Середнє
Технічна підтримка	Платна	Безкоштовна	Платна	Безкоштовна	Безкоштовна
Підтримка суспільством	9	10	8	8	10
Підтримка виробником	9	8	9	7	8
Документація	8	8	7	7	10

Згідно з результатами дослідження, які наведені в таблиці 4.1, для роботи з такими СКБД як Oracle та Microsoft SQL потрібно мати досить великі ресурсні можливості, окрім того вони є платними. А для розроблюваного програмного продукту необхідна безкоштовна система керування базою даних. Також всі дані, що зберігатимуться будуть мати чітку структуру, тобто СКБД повинна бути реляційною, тому безкоштовна СКБД MongoDB також не підходить, оскільки її модель БД є нереляційною. PostgreSQL має менші вимоги до використання ресурсів ніж попередні СКБД, однак вона має досить незрозумілу документацію.

Таким чином виходячи з проведеного аналізу в якості СКБД було обрано популярну на сьогодні – MySQL. По-перше вона має безкоштовну версію для роботи та здатна забезпечити розроблюваний продукт потужним функціоналом з простим та зрозумілим інтерфейсом. А по-друге для розроблюваного продукту важливо економічне використання ресурсів та швидкість вирішення проблем, що можуть виникнути під час роботи з СКБД. Окрім того функції безпеки MySQL надають надійний захист доступу та використання даних. Також більша частина вебсайтів та додатків може без перешкод працювати з цією СКБД.

4.2 Порівняння та вибір мови програмування для розробки серверної частини програмного продукту

Для розробки серверної частини використовують набір інструментів та мов програмування, які використовуються для проєктування, побудови та обслуговування операцій вебдодатків на стороні сервера. Код для серверної частини системи може бути написаний практично на будь-якій мові програмування. За даними сервісу PYPL [50] прикладами п'ятірки популярних мов на сьогодні є Python, JavaScript, Java, C#, PHP. На рисунку 4.2 зображено статистику популярності мов програмування станом на червень 2020 року.

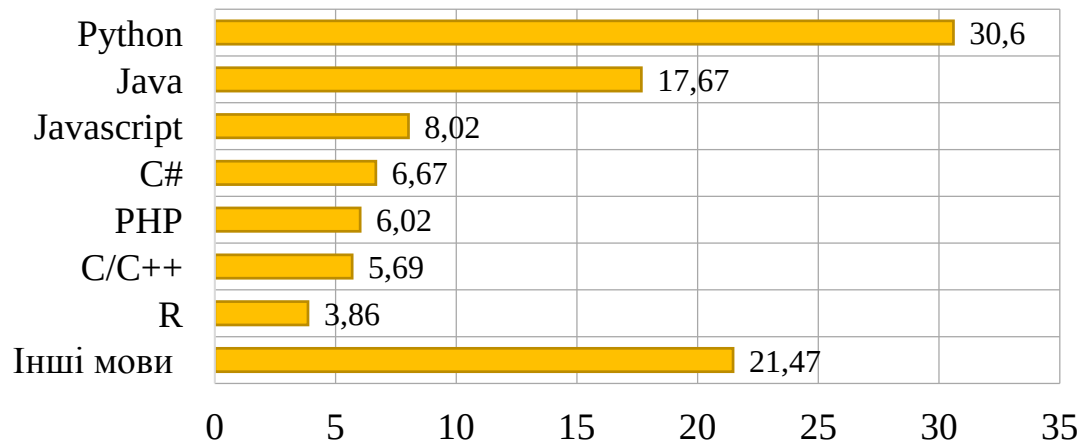


Рисунок 4.2 – Статистика частоти використання мов програмування [50]

Кожна мова програмування має свої сильні та слабкі сторони, тому при виборі важливо розуміти, які задачі та функціонал необхідно реалізувати. Найпопулярнішими мовами, які мають потужний функціонал є Python, PHP та JavaScript, в таблиці 4.2 наведено їх порівняння [51-53].

Таблиця 4.2 – Порівняльна характеристика мов програмування [51-53]

Параметр	Python	PHP	JavaScript
Об'єктно-орієнтована	Так	Так	Так
Виконується на стороні сервера	Так	Так	Так
Мова сценарію на стороні клієнта	Ні	Ні	Так
Виконується в вікні браузера	Ні	Ні	Так
Підтримка БД	Так	Так	Так
Менеджер пакетів	PIP – система управління пакетами, що використовується для установки і управління програмними пакетами.	Composer – це пакетний менеджер, який надає засоби з управління залежностями в PHP-додатках.	NPM – менеджер пакетів для мови програмування JavaScript.
Підтримка спільноти	Має велику спільноту підтримки, що зростає.	Широка підтримка з боку спільноти.	Широка підтримка з боку спільноти.
Документація	Гарно прописана документація для всіх версій мови.	Мова PHP добре документована.	Добре документована, з прикладами.

На основі отриманих результатів порівнянь, в якості мови програмування серверної частини було обрано – Python. Таке рішення обумовлено рядом переваг даної мови перед іншими. По-перше вона має якісну та широку підтримку суспільства, яке за потреби здатне швидко допомогти при вирішенні проблем, що можуть виникнути під час розробки. По-друге мова має легкий, лаконічний та зрозумілий синтаксис, є кросплатформеною. По-третє велика кількість бібліотек дозволяють не створювати власний продукт з нуля, а скористатись вже існуючим функціоналом.

Окрім того, ще одним важливим фактором при виборі мови стало застосування мови в багатьох сферах, що дозволить в майбутньому за потреби без проблем розширювати можливості розроблюваного продукту.

4.3 Обґрунтування вибору мови програмування для клієнтської частини програмного продукту

Для створення графічного інтерфейсу користувача в більшості випадків використовують три основні компоненти HTML, CSS, JavaScript, що інтерпретуються браузером на стороні клієнта.

Мову гіпертекстової розмітки HTML використовують для створення структури зовнішнього вигляду вебсторінки. Каскадні таблиці стилів CSS – для встановлення стилів до HTML-вмісту [54]. Мова сценаріїв JavaScript дозволяє реалізувати складні та динамічні сценаріїв на сторінці браузера, які взаємодіють з елементами HTML на сторінці [54].

Використання JavaScript – дозволяє контролювати роботу браузера та динамічно створювати вміст вебсторінки в залежності від дій користувача, ще однією її перевагою є менша взаємодія з сервером. Така властивість дозволяє виконати перевірку введених користувачем даних перед відправкою їх на сервер, що у свою чергу забезпечує менше навантаження на сервер та використання меншої кількості трафіку на ньому [52].

Отже в якості мови програмування для клієнтської частини вирішено обрати JavaScript, яка дозволить забезпечити контроль роботи браузера, динамічне оновлення вмісту вебсторінки, зменшити навантаження як на сам сервер так і на його трафік.

4.4 Вибір фреймворку або бібліотек для розробки клієнтської частини

Мова JavaScript має багато фреймворків та технологій, що дозволяє скоротити час та витрати на розробку, оскільки зазвичай вони надають безліч вбудованих компонентів та функцій керування користувачем, які не потрібно писати з нуля. Трійкою найпопулярніших технологій мови JavaScript – Vue.js, Angular.js та React, їх порівняння наведено в таблиці 4.3 [55].

Таблиця 4.3 – Порівняння технологій Vue.js, Angular та React [55]

Параметри	Angular.js	React	Vue.js
Тип технологій	Фреймворк	Бібліотека для створення інтерфейсу користувача	Бібліотека
Тип застосунків	Застосунки для мобільних пристроїв (native apps), вебзастосунки (web apps), гібридні (native apps + web apps)	Односторінкові застосунки, мобільні додатки.	Односторінкові застосунки, застосунки для мобільних пристроїв (native apps)
Ідеально підходить для	Якщо ви хочете зосередитись на масштабних, багатофункціональних додатках	Підходить як для сучасної веброзробки так і для власних додатків на iOS та Android	Ідеально підходить для веброзробки та односторінкових застосунків
Розмір	500+ KB	100 KB	80 KB
Популярність	Широко популярний серед розробників	За рік понад 27 000 запитів	Більше 40 000 запитів на GitHub протягом року

Angular – це повноцінний фреймворк, який пропонує всі необхідні інструменти від розробки до тестування під час створення вебзастосунків, тоді як React – це гнучка бібліотека, яка потребує підтримки інших бібліотек для розробки. Vue.js – це наймолодша бібліотека, що поєднує у собі Angular та React, який додає швидкості, ефективності та простоти в розробці вебдодатків [56]. Також Vue.js надає користувачу детальну документацію, що допомагає розробникам при створенні проєкту чи вивченні технології. Створені додатки легко інтегрується в існуючу інфраструктуру. Завдяки невеликим розмірам, його швидкість та гнучкість є оптимальними, що дозволяє покращити продуктивність в порівнянні з іншими технологіями [57].

В результаті аналізу, у якості допоміжної технології було обрано бібліотеку Vue.js, яка є чудовим варіантом для легкої веброзробки, адже для створення застосунку достатньо знань HTML та JavaScript.

Однак згідно з архітектурою системи, клієнтська частина повинна взаємодіяти із тимчасовим сховищем даних. Тимчасове сховище даних зберігатиме дані користувача після реєстрації, задля реалізації механізмів взаємодії авторизованих на неавторизованих користувачів. Саме тому було прийнято рішення використовувати також для Vue.js спеціальну бібліотеку-розширення Vuex.

Vuex – це шаблон управління станами та бібліотека для програм Vue.js. Він служить централізованим сховищем для всіх компонентів програми, з правилами, що гарантують, що стан може мутувати лише передбачувано. Він також інтегрується з офіційним розширенням Vue.js. В якості стану мається на увазі тимчасове сховище даних, дані якого можуть зберігатись в локальному сховищі даних, cookies або в оперативній пам'яті браузера [58].

Vue.js та Vuex дозволить розробити дуже якісну та сучасну клієнтську частину завдяки великому та дружньому суспільству, гарно прописаною документацією та оптимізованим використанням ресурсів браузерів користувачів.

4.5 Вибір вебсерверу для клієнтської частини

Розроблюваний програмний продукт потребує у вебсервері, тому що він обробляє вхідні мережеві запити через HTTP та кілька інших пов'язаних протоколів. Найбільш популярними вебсерверами на сьогодні є Apache та Nginx. Обидва є надійним та безпечним програмним забезпеченням, що використовуються для обробки HTTP – запитів, однак кожен з них має власний набір можливостей та функцій. В таблиці 4.4 наведено порівняння даних вебсерверів [59, 60].

Таблиця 4.4 – Порівняння вебсерверів Apache та Nginx [59, 60]

Параметр	Apache	Nginx
Метод обробки з'єднань з клієнтами	Використовується модель «одне з'єднання на процес». Для кожного запита клієнта створюється окремий процес. Такий підхід не підходить для високонавантажених систем, оскільки велика кількість запитів сповільнює обробку з'єднань та потребує більше пам'яті та ресурсів.	Використовується архітектура для обробки декількох з'єднань. Складається з master-процесу, який створює дочірні процеси, що обслуговують всі з'єднання з клієнтами. Nginx використовує безкінечний цикл, який проходить по всім з'єднанням та відповідає на них, а коли з'єднання закривається, то видаляє.
Видача контенту	Генерує статичний та динамічний контент без використання проксування та налаштування додаткових інструментів для роботи з динамікою.	В 2,5 рази швидше за Apache генерує статичний контент та використовує менше пам'яті. Однак не здатний генерувати динамічний контент без додаткових налаштувань.
Підтримка операційної системи	Підтримує всі операційні системи як Unix, так і Windows	Підтримує всі операційні системи як Unix, так і Windows.
Наявність модулів	Має 60 офіційних модулів та безліч любительських. Вони не потребують збирання та завантажуються відразу.	Існує 130 офіційних модулів, що потребують збирання. Підхід є складнішим, однак забезпечує більшу безпеку.
Підтримка та документація	Доступна офіційна документація.	Доступна офіційна документація
Підходить для	Для малих проєктів, в яких невелика кількість з'єднань.	Для проєктів де обслуговуються велика кількість з'єднань одночасно. При рівномірній завантаженості ресурсів та пам'яті системи.

В результаті порівняння двох досить популярних вебсерверів було прийнято рішення використовувати в якості серверу для клієнтської частини – Nginx. Оскільки він займає менше місця на диску є масштабованою та високопродуктивною реалізацією, у якій кожен робочий процес здатний одночасно обробляти тисячі HTTP – з'єднань.

4.6 Вибір фреймворку або бібліотек для розробки серверної частини

Використання фреймворків під час створення будь-якого проєкту забезпечує розробників великою кількістю готових інструментів, що дозволяють полегшити та пришвидшити роботу. Вони пропонують початкову структуру для майбутнього проєкту, а також забезпечують автоматизацію реалізації загальних рішень тим самим скорочуючи час розробки та надає можливість розробникам зосередитись на створенні логіки проєкту, а не на рутинних елементах [61]. Одними з найбільш популярних фреймворків мови програмування Python є – Django та Flask.

Django є безкоштовним фреймворком веброботи з відкритим кодом Python, який намагається надати розробникам всі необхідні функції за замовчуванням, а не пропонувати їх в якості окремих бібліотек. Так, наприклад, завдяки можливості об'єктно-реляційного відображення (ORM), що дозволяє відобразити об'єкти у таблицях бази даних, один і той же код може працювати з різними базами даних та його не складно перенести з однієї бази в іншу [61].

Flask це фреймворк, який підходить як для простих так і для складних в розробці проєктів, а також є сумісним зі сторонніми бібліотеками та має велику кількість розширень [62]. Фреймворк складається з двох компонентів: Werkzeug та Jinja 2. Werkzeug – це простий маршрутизатор, що забезпечує обробку запитів та відправку відповідей, а також забезпечує сумісність з

WSGI, що дозволяє використовувати фреймворк з вебсервером. Jinja 2 – це шаблонізатор, що поставляє згенеровані на сервері дані в HTML-сторінку [63].

Flask та Django повноцінні, самостійні фреймворки, які пропонують широкий та схожий функціонал для розробки проєкту. В таблиці 4.5 наведено порівняння цих фреймворків [64, 65].

Таблиця 4.5 – Порівняння фреймворків Flask та Django [64, 65]

Параметр порівняння	Django	Flask
База даних	Має вбудовану ORM та система міграцій, що може керувати базами даних. Розробники можуть використовувати будь-яку БД та виконувати загальні завдання БД без необхідності писати довгі запити.	Розробники можуть працювати з різними БД, використовуючи системи ORM для Python та SQLAlchemy, що необхідно додатково встановлювати, як набір інструментів SQL. Для загальних завдань слід писати запити SQL.
Панель адміністратора	Наявний повністю інтегрований адмін-інтерфейс для керування застосунком, який можна налаштувати.	Функції для обробки адміністративних завдань за замовчуванням відсутні. Для їх отримання необхідно встановити розширення Flask-Admin.
Аутентифікація та переваги користувачів	Надає застосунок для аутентифікації, що за замовчуванням надає реалізацію для управління користувачем та привілей.	Надає захищені cookie в якості інструменту для власної реалізації.
Основні особливості функціоналу	Контроль версій для баз даних (міграції). Власний движок шаблонів. Об'єктно-реляційні відображення (ORM). Маршрутизація URL. Підтримка вебсерверів. Вбудовані механізми аутентифікації. Підтримка інтернаціоналізації	Швидкий вбудований дебагер. Вбудований dev-сервер. Підтримка REST-запитів. Сумісний з будь-якою ORM. Безпечні cookie для управління сесією на клієнті. Підтримка юніт-тестів. Відповідає WSGI 1.0.
Використовується в	Instagram, Pinterest, Udemy, Coursera, Zapier.	Netflix, Lyft, Reddit, Zillow, MailGun

Django є більш популярним, оскільки надає велику кількість готових функцій і скорочує час на створення складних додатків. Він підходить для як для простих так і більш складних та значних проєктів [65]. За допомогою Flask можна створити простий додаток, який згодом можна розширити шляхом додання нових більш функціональних можливостей та зробити його складнішим. Завдяки чому фреймворк здатний забезпечити гнучкість для швидкості розширення проєкту [65].

Django має власну інтерпретацію шаблону проєктування Model-View-Controller (MVC), що зветься Model-Template-View (MTV) [66]. Архітектуру даного шаблону проєктування зображено на рисунку 4.3.

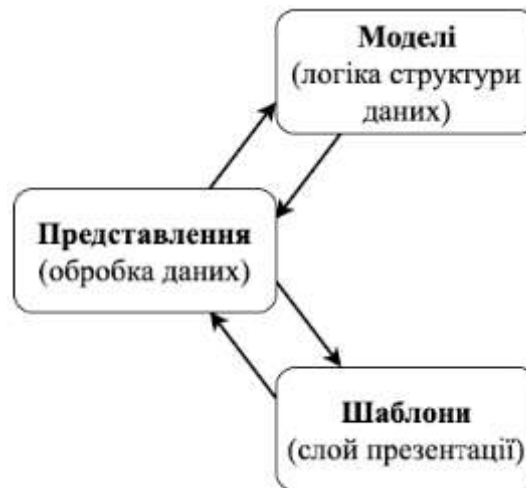


Рисунок 4.3 – Архітектура шаблону проєктування MTV.

Представлення в архітектурі MTV може виглядати як контролер, але це не так. Представлення в цій архітектурі MTV форматує дані за допомогою моделі. У свою чергу, він передає в базу даних і ті дані, які передаються в шаблон для перегляду.

Головна мета шаблону – зберегти все, що відображає браузер. Він використовує данні з моделей, що надходять на сервер з різних частин та об'єднує під час взаємодії з користувачем. Таким чином шаблони в Django більше схожі на представлення у шаблоні MVC. Цей шар є більш цілеспрямованим, і завдяки фреймворку Django він забезпечує набагато

більшу розширюваність для розробників інтерфейсу, ніж те, що надавала архітектура MVC [67].

Таким чином в ході порівнянь двох фреймворків було прийнято рішення для розроблюваного проєкту використовувати Django фреймворк. Таке рішення обумовлено наявністю в Django таких можливостей як аутентифікація, маршрутизація URL-адрес, механізм шаблонів, об'єктно-реляційне відображення (ORM) та міграція схеми бази даних, що роблять фреймворк надзвичайно масштабованим, швидким та універсальним [61].

Наявність готових рішень в фреймворку під час розробки дозволить пришвидшити та спростити розробку, або наприклад, завдяки таким можливостям як ORM та міграціям, а у випадку зростання кількості даних в базі даних можна буде спокійно перейти на іншу за потреби.

Окрім того широка користувачька база тільки зростає, оскільки велика кількість розробників вважають даний фреймворк найкращим вибором для нових технологій, таких як машинне навчання. А у 2020 році планується випуск нових бібліотек для задоволення зростаючого попиту [61].

4.7 Вибір вебсервісу для серверної частини

Для серверної частини розроблюваного продукту потрібен графічний інтерфейс, оскільки згідно зі стандартом мови програмування PEP-3333 для взаємодії вебсерверів із кодом на мові Python використовується стандарти Web-Service Gateway Interface (WSGI) або Asynchronous Server Gateway Interface (ASGI), що гратимуть роль мосту між ними, як зображено на рисунку 4.4 [68].

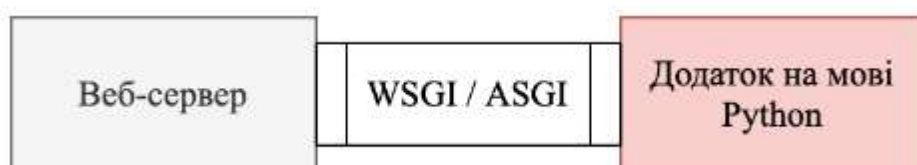


Рисунок 4.4 – Взаємодія між вебсервером та додатком на мові Python.

В обраний фреймворк Django вже вбудована взаємодія із WSGI та ASGI, але взаємодія із останнім була додана нещодавно, тому технологія ще погано протестована та сира. Саме тому кращим рішенням буде використання технології WSGI.

Сьогодні існує велика кількість бібліотек, що реалізують дану взаємодію. Порівняння найпопулярніших з них зображено в таблиці 4.6 [69].

Таблиця 4.6 – Порівняння бібліотек для реалізації WSGI

Назва	Bjoern	uWSGI	CherryPy	Gunicorn
Середнє число запитів на секунду (в одно поточному режимі більше – краще)	75000	500	4000	1500
Займання оперативної пам'яті системи (МБ, менше – краще)	9	60	30	15
Навантаження центрального процесору серверу в багато поточному режимі (максимум 200%, тому що використано 2 ядра ЦПУ, більше – краще)	100	50	140	160

В результаті порівняння можемо зробити висновок, що Bjoern може оброблювати більше запитів на секунду, займає менше оперативної пам'яті, але його недоліком є те, що він використовує тільки одно поточну логіку та важко визначати, яку саме кількість ресурсів буд використовуватись. Також він складний у налаштуванні.

Тож, найкращим рішенням буде використати саме Gunicorn, тому що, в ньому наявна можливість «з коробки» налаштування максимального часу на очікування відповіді, визначання кількості запущених процесів водночас та багато інших налаштувань. Також в ньому наявна можливість збирати дані під час роботи, а саме кількість запитів, які статуси відповідей надходять, з яких IP адрес надходять запити і так далі, які можна відстежувати в режимі реального часу, та у випадку критичних ситуацій на сервері легше та швидше знаходити рішення.

4.8 Вибір фреймворку або бібліотек для розробки модулю асинхронних задач

Для паралельного виконання задач в системі необхідно розробити модуль асинхронних задач, що дозволить економити час на очікуванні для користувачів та максимально ефективно використовувати ресурси системи.

Модуль асинхронних задач можна сконструювати за допомогою бібліотек або фреймворків таких як Celery [70], Dramatiq [71] та Argo[72] або за допомогою розробки власних скриптів.

Celery та Dramatiq дуже схожі за принципом роботи один на одного. Це реалізація черги завдань для вебдодатків Python, яка використовується для асинхронного виконання роботи поза циклом запиту-відповіді HTTP [73]. Вони для реалізації роботи з чергами використовують брокерів. Різниця між Celery та Dramatiq у тому, що Dramatiq в якості брокера може використовувати оперативну пам'ять системи, а в Celery реалізований функціонал для задач, що викликаються у зазначений в конфігураціях час, тобто використовує crontab із систем Linux.

Для даного програмного продукту модуль асинхронних задач повинен вміти періодично витягувати дані з бази даних, та проводити з ними певні маніпуляції, також повинен мати можливість масштабуватися вертикально, тобто використовувати максимальну кількість виділених ресурсів, та вміти ними керувати.

Найкраще для даної системи підходить фреймворк Celery, але він потребує спеціального брокера для коректного функціонування та в ньому відсутня можливість для вертикального масштабування. Тому було прийнято рішення розробити власні скрипти для кожної з задач.

4.9 Вибір засобів для розгортання проєкту

Для того щоб мінімізувати можливість виникнення проблем під час розгортання системи на новій робочій машині, при зміні середовища чи операційної системи – можна скористатись технологією контейнеризації.

Контейнеризація – це легковагова віртуалізація та ізоляція ресурсів на рівні операційної системи, яка дозволяє запускати застосунки та інший необхідний мінімум системних бібліотек в повністю стандартизованому контейнері. Завдяки такому підходу всі необхідні для запуску застосунку, упаковані в один образ та можуть бути використані повторно [74]. Також застосунки з різних контейнерів не можуть взаємодіяти одне з одним, оскільки забезпечується повна ізоляція контейнерів [75].

Контейнером називають – стандартну одиницю програмного забезпечення, що пакує код та всі його залежності, завдяки чому програма працює швидко та надійно від одного обчислюваного середовища до іншого [76]. В непрацюючому стані контейнер – це файл на диску, а під час використання його ядро розпаковує образ контейнеру з потрібними файлами та метаданими, та передає їх до ядра операційної системи [75].

Програмне забезпечення, що використовує технологію контейнеризації, доступне як для Linux, так і для Windows, при цьому воно завжди працюватиме однаково, незалежно від інфраструктури. Контейнери ізолюють програмне забезпечення від навколишнього середовища та забезпечують його рівномірну роботу [76].

На сьогодні існують такі популярні системи контейнеризації Docker, Docker-compose, Kubernetes. Docker – це технологія, що дозволяє огорнути розроблені застосунки в контейнер та виконувати їх розгортання швидко та легко в необхідному середовищі. Docker-compose дозволяє налаштувати та запустити декілька контейнерів Docker. Він використовує файл YAML для налаштування служб застосунку, а потім за допомогою однієї команди дозволяє створити та запустити всі служби з конфігурації [77].

Kubernetes – це платформа з відкритим вихідним кодом для управління контейнеризованими робочими навантаженнями та супутніми службами. Його основні характеристики – кросплатформенність, розширюваність, успішне використання декларативної конфігурації та автоматизації. Послуги, підтримка та інструменти Kubernetes широко доступні. Він дозволяє утримувати баланс в навантаженні між окремими кластерами або віртуальними машинами. Kubernetes базується на системі контейнерів Docker. [78].

В якості системи контейнеризації для розроблюваної системи було прийнято рішення використовувати Docker-compose, оскільки наявна необхідність розгортання декількох контейнерів. А в порівнянні з Kubernetes розгортання контейнерів у Docker-compose у більшості випадках відбувається набагато швидше.

4.10 Вибір інтегрованого середовища

Інтегрованим середовищем розробки (або IDE) – це програмне забезпечення, яке полегшує розробку проєкту та поєднує в собі загальні інструменти для розробки в єдиний графічний інтерфейс користувача [79]. Зазвичай IDE складається з редактора, який призначений для роботи з кодом, наприклад автодоповнення чи підсвітка синтаксису; інструментів для збирання проєкту, виконання та відладки [80].

Обране інтегроване середовище повинно забезпечити розробника набором базових функцій, що дозволять спростити та прискорити процес розробки, до таких функцій відносяться [80]:

- збереження файлів – середовище надає можливість збереження процесу роботи та відновлення його пізніше в такому ж стані як і до закриття IDE;
- можливість виконати запуск коду з середовища розробки;
- підтримка відладчика – можливість покрокового виконання коду є однією з базових функцій IDE;

– перевірка синтаксису – надає можливість виділити змінні, ключові слова тощо, що у свою чергу пришвидшує пошук в коді, а також робить його читання та розуміння простішим;

– автоматичне форматування коду, якісне середовище автоматично розпізнає та робить необхідні відступи чи переноси на наступний рядок після введення конструкцій, синтаксис написання яких цього вимагає.

Окрім наведеного базового набору функцій існує також безліч інших, які можна встановити в якості розширень в інтегроване середовище розробки.

Існує велика кількість середовищ для написання, редагування та відлагодження коду, кожний з яких мають різний набір функцій. В таблиці 4.7 наведено порівняння редакторів коду Visual Studio Code (VS Code), Vim, Atom [81].

Таблиця 4.7 – Порівняння редакторів коду [81]

Параметр	Vim	Visual Studio Code	Atom
Мови, що підтримуються	PHP, Python, Ruby, Perl, Java, Go, SQL, JavaScript, XML, HTML, CSS та інші.	PHP, Python, Ruby, Perl, Java, Go, SQL, JavaScript, XML, HTML, CSS та інші.	PHP, Python, Ruby, Perl, Java, Go, SQL, JavaScript, XML, HTML, CSS та інші.
Перевірка синтаксису	+	+	+
Комбінації клавіш	+	+	+
Встановлення плагінів	+	+	+
Перевірка орфографії	+	+	+
Підтримка Git	+	+	+

Vim є розширеною версією Vi, текстового редактора для UNIX, який не був створений спеціально для редагування коду, однак встановлення спеціальних плагінів здатне це виправити [81]. Робота з Vim в основному відбувається за рахунок клавіатури та команд, що з неї вводяться.

Atom є текстовим редактором з множиною налаштувань, який написаний з використанням фреймворку Electron, що забезпечує підтримку

середовища на всіх платформах. Відсутність вбудованої підтримки збирання та відлагодження проекту можна віднести до недоліків IDE, що можна виправити за допомогою встановлених розширень [80].

Visual Studio Code – повноцінним функціональним кросплатформним редактором коду, який можна налаштувати під будь-яку задачу. Система так само як і Atom побудована на Electron, тому вона має всі ті самі можливості. Налаштування плагінів для середовища виконується досить просто, наприклад для забезпечення підтримки Python, необхідно знайти та встановити необхідний плагін. Після чого VS Code автоматично визначить інтерпретатор Python та встановить бібліотеки [80].

Середовища Atom та Visual Studio Code є досить схожими, оскільки в їх основі лежить один і той самий фреймворк, однак в продуктивності та популярності та швидкості роботи Atom програє VS Code. Така різниця полягає в якості оптимізації розширень (плагінів) для обох середовищ розробки. Для Atom та Visual Studio Code існує безліч сторонніх плагінів, тем та інструментів для налаштування середовища. Однак додаткові розширення для Visual Studio Code використовують менше ресурсів ніж для Atom [82].

В результаті проведеного аналізу в якості інтегрованого середовища розробки було обрано Visual Studio Code. Оскільки дане середовище в має зручний та інтуїтивно зрозумілий інтерфейс, в порівнянні з Vim, та здатне забезпечити розробника базовим функціоналом розробки, а за потреби розширити функціональні можливості за рахунок великого списку плагінів. А в порівнянні з Atom, VS Code є швидшим та вимагає менше ресурсів для коректної роботи плагінів.

4.11 Висновки до розділу 4

В даному розділі розглянуто аналіз порівняння та вибір необхідного інструментарію для розробки програмного продукту. Таким чином в ході проведеного аналізу в якості СКБД було обрано популярну на сьогодні –

MySQL, з якою працюють більша частина вебсайтів та додатків. Вона здатна забезпечити розроблюваний продукт потужним функціоналом, надійним захистом даних, економічним використанням ресурсів та швидким вирішенням проблем.

На основі отриманих результатів порівнянь мов програмування, в якості мови для серверної частини проєкту обрано популярну мову – Python. Оскільки вона має якісну та широку підтримку суспільства, легкий, лаконічний та зрозумілий синтаксис, є кросплатформенною, а також має велику кількість бібліотек, що дозволяють використовувати для розробки вже існуючий функціонал.

В якості мови програмування для клієнтської частини обрано JavaScript, що дозволить забезпечити контроль роботи браузера, динамічне оновлення вмісту вебсторінки та зменшить навантаження як на сам сервер так і на його трафік. Також в якості допоміжної технології для розробки клієнтської частини обрано бібліотеку Vue.js, яка є чудовим варіантом для легкої веброзробки, адже для створення застосунку достатньо знань HTML та JavaScript.

В якості тимчасового сховища, яке взаємодіє з клієнтською частиною, зберігає дані користувача після реєстрації та реалізує механізми взаємодії між авторизованими на неавторизованими користувачами, обрано спеціальну бібліотеку-розширення для Vue.js під назвою VueX.

В результаті порівняння двох досить популярних вебсерверів було прийнято рішення використовувати в якості серверу для клієнтської частини – Nginx. Оскільки він займає менше місця на диску є масштабованою та високопродуктивною реалізацією, у якої кожен робочий процес здатний одночасно обробляти тисячі HTTP – з'єднань.

В якості фреймворку для серверної частини обрано Django. Таке рішення обумовлено наявністю в нього таких можливостей як аутентифікація, маршрутизація URL-адрес, механізм шаблонів, об'єктно-реляційне відображення (ORM) та міграція схеми бази даних, що роблять фреймворк

надзвичайно масштабованим, швидким та універсальним. Наявність готових рішень дозволить пришвидшити та спростити розробку.

В якості вебсервісу було обрано Unicorn, тому що, в ньому наявна можливість «з коробки» налаштування максимального часу на очікування відповіді, визначання кількості запущених процесів водночас та багато інших налаштувань. Також в ньому наявна можливість збирати дані під час роботи, а саме кількість запитів, які статуси відповідей надходять, з яких IP адрес надходять запити і так далі, які можна відстежувати в режимі реального часу, та у випадку критичних ситуацій на сервері легше та швидше знаходити рішення.

Для паралельного виконання задач в системі необхідно розробити модуль асинхронних задач, що дозволить економити час на очікуванні для користувачів та максимально ефективно використовувати ресурси системи. В ході аналізу було виявлено, що більш менш може підійти фреймворк Celery, але він потребує спеціального брокера для коректного функціонування та в ньому відсутня можливість для вертикального масштабування. Тому було прийнято рішення розробити власні скрипти для кожної з задач.

В якості системи контейнеризації для розроблюваної системи було прийнято рішення використовувати Docker-compose, оскільки наявна необхідність розгортання декількох контейнерів. Використання даної технології дозволить мінімізувати можливість виникнення проблем під час розгортання системи на новій робочій машині, при зміні середовища чи операційної системи.

В результаті проведеного аналізу в якості інтегрованого середовища розробки було обрано Visual Studio Code. Оскільки дане середовище в має зручний та інтуїтивно зрозумілий інтерфейс, в порівнянні з Vim, та здатне забезпечити розробника базовим функціоналом розробки, а за потреби розширити функціональні можливості за рахунок великого списку плагінів. А в порівнянні з Atom, VS Code є швидшим та вимагає менше ресурсів для коректної роботи плагінів.

5 КОНСТРУЮВАННЯ РЕКОМЕНДАЦІЙНОЇ СИСТЕМИ

Розроблюваний програмний продукт складається з двох основних частин, а саме з клієнтської та серверної частини. Оскільки використовувався єдиний репозиторій для зберігання проєкту, щоб полегшити розробку, було прийнято рішення створити відповідно два сховища даних «backend» та «frontend» як зображено на рисунку 5.1.

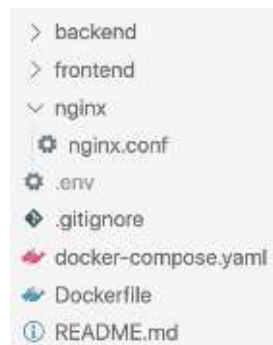


Рисунок 5.1 – Базова структура проєкту

На даному рисунку також зображено інші файли та директорії, які використовуються для розгортання проєкту та розглядатимуться у розділі 5.5.

5.1 Конструювання серверної частини

Для розробки серверної частини обрано фреймворк Django, тому структура проєкту створена за стандартами обраного фреймворку.

Для більш зручного створення програмного інтерфейсу було використано додаткову бібліотеку Django-REST-Framework (DRF). Вона необхідна для серіалізації та десеріалізації, тобто конвертації із об'єктів мови програмування Python в JSON та навпаки. Цей функціонал необхідний для реалізації надсилання відповідей до клієнтської частини у форматі JSON та для обробки запитів з клієнтської частини у даному форматі. Це дозволить стандартизувати формат даних, що передаються між сервісами.

Весь функціонал проєкту можна поділити на умовні групи, зображені в таблиці 5.1.

Таблиця 5.1 – Умовні групи функціоналу проєкту

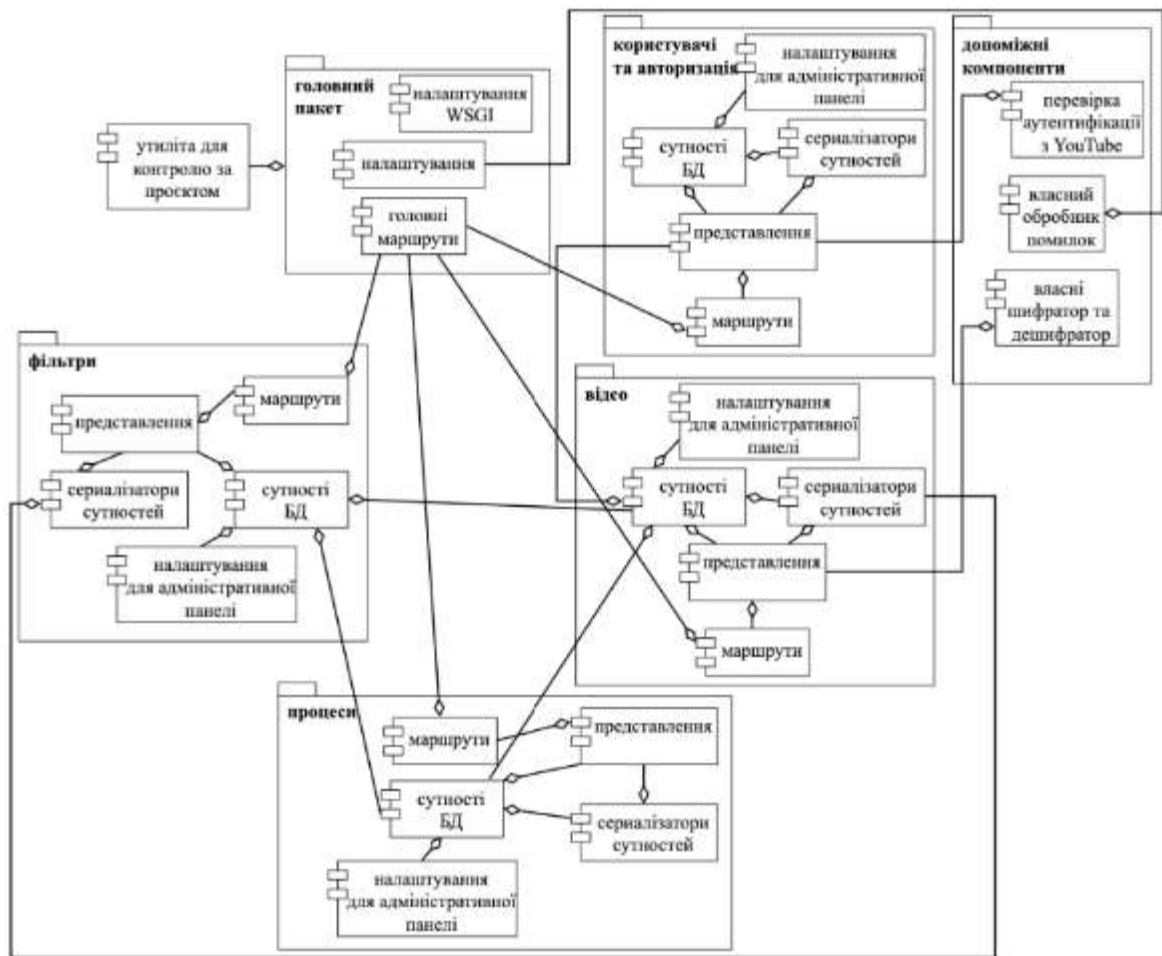
Назва групи	Опис
Головна	Містить файли для конфігурації та запуску проєкту.
Користувачі та авторизація	Містить авторизацію та реєстрацію користувачів у системі, а також авторизацію у платформі YouTube.
Процеси	Містить функціонал для взаємодії із процесами, а саме їх створення, отримання та оновлення.
Відео	Містить функціонал для отримання даних про відео, додавання відео до списку «Обране» окреме для кожного користувача, функціонал для оцінювання відео за критеріями.
Фільтри	Містить функціонал для отримання даних фільтрів.

Оскільки використовуються фреймворки Django та DRF, то головна група являтиме собою результат генерації проєкту за допомогою консольної утиліти Django. Усі інші компоненти являтимуть собою «додатки» [83], які матимуть схожі назви модулів, але різний функціонал, що в них зберігається. Усі використані модулі для серверної частини наведено в таблиці 5.2.

Таблиця 5.2 – Опис списку використаних модулів в серверній частині

Назва модулю	Опис модулю
Налаштування WSGI	Точка входу для вебсерверів, сумісних з WSGI, для обслуговування проєкту.
Налаштування	Налаштування та конфігурації для проєкту.
Головні маршрути	Оголошення URL-адрес для цього проєкту.
Налаштування для адміністративної панелі	Спеціальні налаштування для сутностей, над якими можна виконувати певні дії в панелі адміністратора.
Сутності БД	Сутності для реалізації ORM в додатку.
Сериалізатори сутностей	Сериалізатори, що дозволяють легко конвертувати ORM-об'єкти в JSON.
Представлення	Модуль, що містить функціонал для обробки запитів.
Маршрути	URL-адреса для поточного додатку.
Перевірка аутентифікації з YouTube	Модуль, для перевірки результату авторизації в системі YouTube.
Власний обробник помилок	Оброблює помилки та надсилає їх до клієнта у разі їх виникнення.
Власні шифратор та дешифратор	Модуль, що містить функціонал для шифрації та дешифрації даних, що не входять до UTF-8 кодування.

Загальну файлову структуру наведено на рисунку 5.2.



Рисунки 5.2 – Розширена архітектура серверної частини

Згідно з даним рисунком утиліта для контролю за проєктом взаємодіє із головним пакетом системи. Компоненти головного пакету використовують маршрути кожного із інших пакетів та формують із них повний список маршрутів для взаємодії із системою. Також модуль налаштувань в головному пакеті використовує власний обробник помилок із пакету допоміжних компонентів.

Пакет користувачі та авторизація використовує сутності з пакету відео, так само як пакети фільтри та процеси. Також пакет користувачі та авторизація використовує модуль перевірки аутентифікації з системи YouTube. Пакет процеси використовує сутності з пакету фільтрів. Пакет відео використовує власні шифратор та дешифратор у модулі представлень.

Усі основні пакети мають схожу структуру модулів. На рисунку 5.3 зображено класова взаємодії між модулями у пакеті відео.

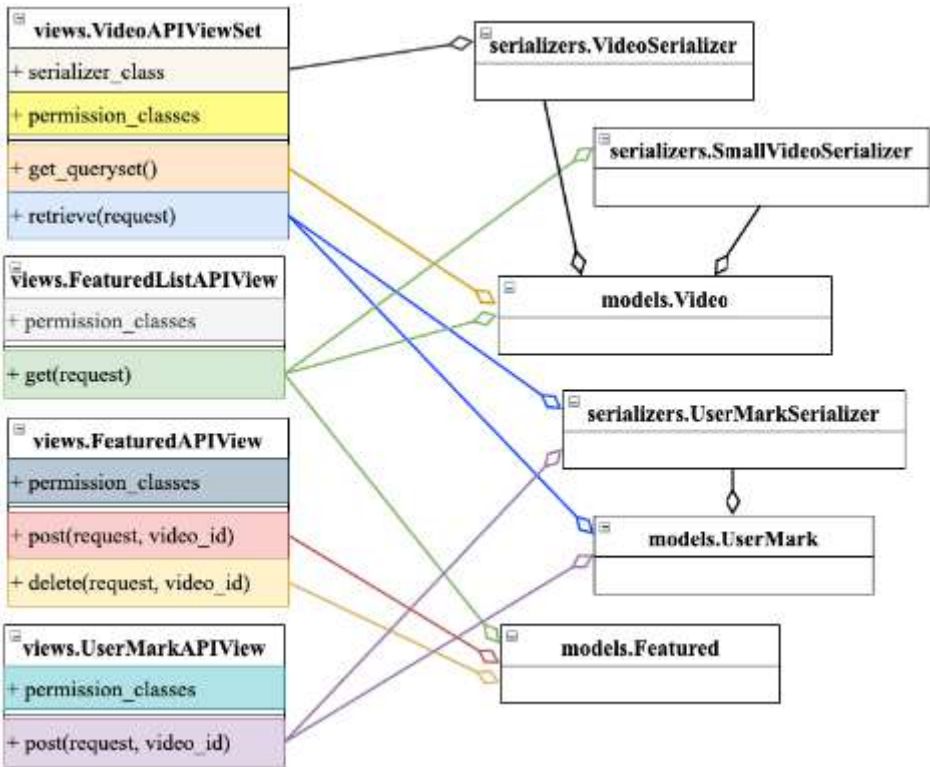


Рисунок 5.3 – Класова взаємодія на прикладі пакету відео

На даному рисунку клас VideoAPIViewSet використовує класи VideoSerializer, UserMarkSerializer та UserMark models.Video. Класи FeaturedListAPIView, FeaturedAPIView та UserMarkAPIView використовують UserMarkSerializer, SmallVideoSerializer, UserMark, Video та Featured відповідно до рисунку. Опис та призначення класів наведено в таблиці 5.3.

Таблиця 5.3 – Опис класів пакету відео

Назва	Опис
1	2
views.VideoAPIViewSet	Представлення, що надає дані по конкретному відео, спираючись на його id.
views.FeaturedListAPIView	Представлення, що надає список обраних користувачем відео.
views.FeaturedAPIView	Представлення, що надає можливість додавання або видалення відео зі списку обраних.

Продовження таблиці 5.3

1	2
views.UserMarkAPIView	Представлення, що надає можливість зберегти оцінку відео користувачів в розроблюваному програмному продукті.
serializers.VideoSerializer	Конвертує об'єкт відео з БД у JSON, при цьому включає усі дані про відео та повні дані з YouTube.
serializers.SmallVideoSerializer	Конвертує об'єкт відео з БД у JSON, при цьому включає частину даних про відео та посилання на зображення з YouTube.
models.Video	Являє собою об'єкт для взаємодії ORM з БД із таблицею відео.
serializers.UserMarkSerializer	Конвертує об'єкт оцінок користувачів з БД у JSON.
models.UserMark	Являє собою об'єкт для взаємодії ORM з БД із таблицею оцінок користувачів системи.
models.Featured	Являє собою об'єкт для взаємодії ORM з БД із таблицею обрані відео.

На рисунку 5.4 зображена класова взаємодія модулів в пакеті фільтрів.

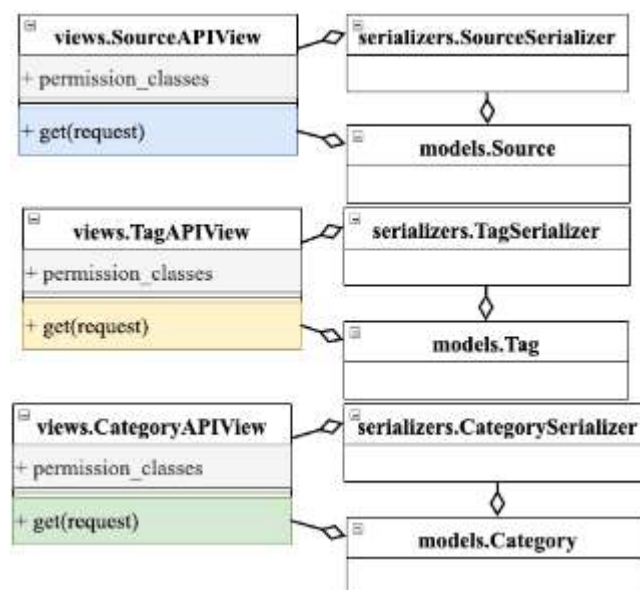


Рисунок 5.4 – Класова взаємодія модулів в пакеті фільтрів

На даному рисунку класи SourceAPIView, TagAPIView та CategoryAPIView використовують SourceSerializer, TagSerializer, CategorySerializer, Tag, Source та Category відповідно. В таблиці 5.4 наведено опис кожного з класів.

Таблиця 5.4 – Опис класів пакету фільтрацій

Назва	Опис
SourceAPIView	Представлення, що надає список джерел, що містять у назві задане користувачем слово або фразу.
TagAPIView	Представлення, що надає список тегів, що містять у назві задане користувачем слово або фразу.
CategoryAPIView	Представлення, що надає список категорій, що містять у назві задане користувачем слово або фразу.
SourceSerializer	Конвертує об'єкт джерело з БД у JSON.
TagSerializer	Конвертує об'єкт тег з БД у JSON.
CategorySerializer	Конвертує об'єкт категорія з БД у JSON.
Tag	Являє собою об'єкт для взаємодії ORM з БД із таблицею тег.
Source	Являє собою об'єкт для взаємодії ORM з БД із таблицею джерело.
Category	Являє собою об'єкт для взаємодії ORM з БД із таблицею категорія.

На рисунку 5.5 зображено класову взаємодію модулів пакету процесів

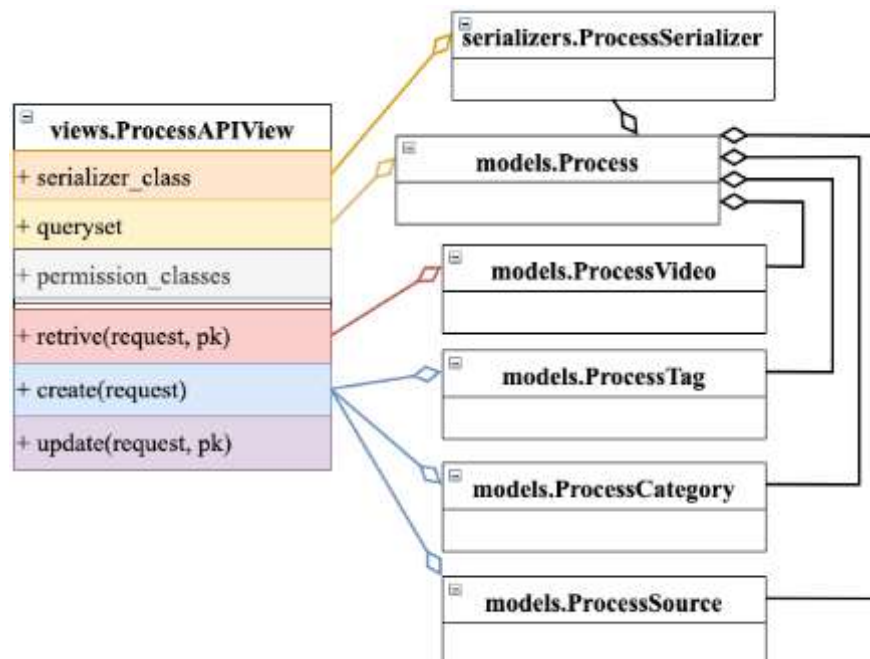


Рисунок 5.5 – Класова взаємодія модулів пакету процесів

На даному рисунку ProcessAPIView використовує ProcessSerializer, Process, ProcessVideo, ProcessTag, ProcessCategory, ProcessSource, які у свою чергу пов'язані між собою. В таблиці 5.5 наведено опис даних класів.

Таблиця 5.5 – Опис класів пакету процесів

Назва	Опис
ProcessAPIView	Представлення, що надає дані по обраному процесу, включаючи список відео, якщо такий існує.
ProcessSerializer	Конвертує об'єкт категорія з БД у JSON.
Process	Являє собою об'єкт для взаємодії ORM з БД із таблицею процес.
ProcessVideo	Являє собою об'єкт для взаємодії ORM з БД із таблицею процес відео.
ProcessTag	Являє собою об'єкт для взаємодії ORM з БД із таблицею процес тег.
ProcessCategory	Являє собою об'єкт для взаємодії ORM з БД із таблицею процес категорія.
ProcessSource	Являє собою об'єкт для взаємодії ORM з БД із таблицею процес джерело.

На рисунку 5.6 зображено класову взаємодію модулів пакету користувачів та авторизації.

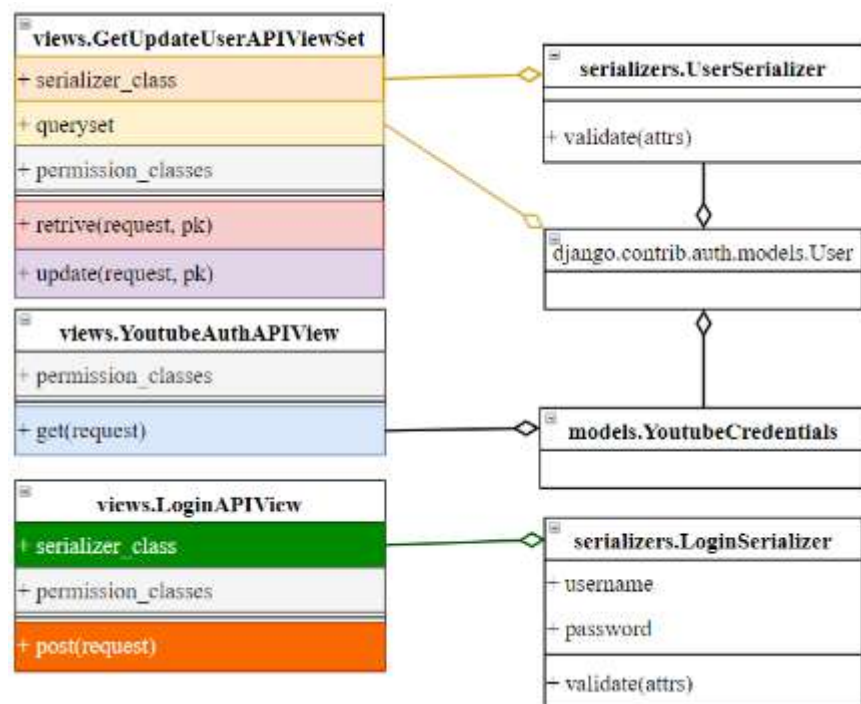


Рисунок 5.6 – Класова взаємодія модулів пакету користувачів та авторизації

На даному рисунку GetUpdateUserAPIView, YoutubeAuthAPIView та LoginAPIView використовують UserSerializer, YouTubeCredentials та LoginSerializer відповідно. В таблиці 5.6 наведено детальний опис даних класів.

Таблиця 5.6 –Опис класів пакету користувачів та авторизації

Назва	Опис
GetUpdateUserAPIView	Представлення, що використовується для отримання або оновлення даних користувачів.
YoutubeAuthAPIView	Представлення, що використовується для обробки даних авторизації системи YouTube.
LoginAPIView	Представлення, що використовується для авторизації користувачів у системі.
UserSerializer	Конвертує об'єкт користувач з БД у JSON.
YouTubeCredentials	Являє собою об'єкт для взаємодії ORM з БД із таблицею дані аутентифікації YouTube.
LoginSerializer	Сериалізує та перевіряє правильність вхідних даних користувачів для авторизації.

Завдяки використанню Django та DRF було створено зрозумілу для розуміння та швидку для оновлення систему. Усю логіку було розбито на певні групи та реалізовано в якості «додатків», які зберігаються в окремих пакетах. Це дозволить запобігти плутанині в файловій системі серверної частини. Приклади коду наведено в додатку А.

5.2 Конструювання авторизації в системі YouTube

Для забезпечення кращої якості результатів пошуку було прийнято використовувати додатково дані з YouTube профілю користувача, що покращить рекомендації, оскільки будуть використані додаткові алгоритми YouTube, на основі історії переглядів. Google дозволяє зареєструвати свій проєкт в консолі розробників, та обрати технології, що будуть використовуватися, в нашому випадку це програмний інтерфейс YouTube [84].

Для забезпечення безпеки профілів користувачів на YouTube, система налаштована на отримання доступів, що дають можливість тільки переглядати дані, та забороняють оновлення або видалення персональних даних завдяки спискам обсягу доступів системи Google [85].

Для реалізації можна використати бібліотеку «google_auth_oauthlib». Але у даній реалізації авторизація користувачів у системі YouTube виконана

інтерфейсу розроблюваного продукту, запит містить ті самі дані, що надав користувач під час кроку 6.

Далі система перевіряє унікальні ідентифікатори, той що прийшов, та той, що зберігається в БД. У випадку співпадіння даних, користувачу надсилається відповідь із статус кодом 301, та посиланням на переадресацію на сторінку із повідомленням про успішну авторизацію.

Таким чином в розроблюваній системі розроблено авторизацію в системі YouTube, що дозволить покращити рекомендації користувачам.

5.3 Конструювання модулю асинхронних задач

Розробка модулю асинхронних задач дозволить знизити навантаження на систему, пришвидшити пошук, фільтрацію та сортування даних. Тому було спроектовано наступні задачі:

- періодичне оновлення відео;
- отримання базових даних по поточному запиту;
- отримання повних даних по поточному запиту;
- підготовка для фільтрації та сортування;
- фільтрація та сортування відео.

На етапі проєктування було прийнято рішення створити окремі модулі для асинхронних задач. Оскільки асинхронні задачі будуть використовувати сутності БД, що наведено в серверної частини, тому щоб уникнути дублювання коду, вирішено дотримуватись принципів Don't Repeat Yourself (DRY) та прийнято рішення створити для них додаток в основному проєкті.

В Django наявна можливість створення власних команд [86], що дозволять запускати асинхронні задачі за допомогою консольної утиліти `manage.py`. Тож в окремому додатку було створено файли (в директорії `management - commands`) відповідно до назви кожної з асинхронних задач:

- `fetchbasedata.py`;
- `fetchfulldata.py`;

- `filtervideos.py`;
- `periodicdataupdate.py`;
- `prepareforfiltering.py`.

Для реалізації їх безкінечного виконання, ключовий метод кожної задачі був запущен в безкінечному циклі. Щоб не надсилати забагато запитів до БД або до сторонніх програмних інтерфейсів, було прийнято рішення призупиняти поточне виконання задачі, після виконання усіх кроків на певний час, що вказаний у константі.

Для того щоб передавати дані між задачами було створено окрему сутність процес, що містить вхідні дані користувача. Вона зв'язана з обраними категоріями, тегами або джерелами, та зі списком відео. Було створено спеціальні статуси для процесів та відео, щоб асинхронні задачі взаємодіяли тільки з необхідними даними. Так, наприклад, якщо процесу належить статус «очікує фільтрації», то дані для цього процесу будуть використовуватися тільки у задачі для фільтрації, так само й для відео. Усі можливі статуси для процесів та відео наведено в таблиці опису сутностей в БД (таблиця 3.5).

Логіка роботи асинхронної задачі зображено на рисунку 5.8 на прикладі реалізації задачі на періодичне оновлення відео.



Рисунок 5.8 – Принцип роботи задачі на періодичне оновлення даних відео

Задача на періодичне оновлення виконує задачу маркування неперевіреного відео, щоб задача збирання повних даних про відео оновила їх дані. Дана задача виконується раз на день. Цей параметр було винесено у константу, щоб було легко конфігурувати періодичність даної задачі. Усі інші задачі формують механізм пошуку списку відео за даними, що ввів користувач.

Коли користувач починає пошук, то створюється процес, що має статус очікування отримання повних даних. Далі створений процес викликається відповідною задачею, а саме отримання базових даних, принцип якої зображено на рисунку 5.9.



Рисунок 5.9 – Принцип роботи задачі на отримання базових даних

Задача на отримання базових даних працює за наступним принципом: отримується група відео, розмір якої винесено в константу, потім формується запит до системи YouTube та оброблюється результат, а саме створюються

сутності відео, канал, дані YouTube та створюється зв'язок між процесом та відео завдяки сутності процес відео, що реалізує зв'язок багато до багатьох. Якщо в базі даних вже наявні схожі сутності, то дублікати не створюються, а просто пропускаються, бо в кожній із сутностей наявні унікальні атрибути.

Після отримання базових даних по процесу, йому присвоюється новий статус та цей процес буде викликатися в задачі на отримання повних даних відео, принцип дії якої зображено на рисунку 5.10.



Рисунок 5.10 – Принцип роботи задачі на отримання повних даних по запиту

Задача на отримання повних даних збирає групи відео, розміри яких винесено в окрему константну. Кожна група відео містить у собі дані по відео, по створеному каналу, та дані користувача з платформи YouTube, щоб використати їх для пошуку. Наступним кроком є надсилання запитів до

платформи YouTube на отримання необхідних даних за правилами, що описані в документації до програмному інтерфейсу [27].

Після отримання невистачаючих даних про теги, джерела, категорії та статистичні дані про кожне відео з групи вони надсилаються до БД на створення або оновлення існуючих даних.

Задача для підготовки даних для фільтрації перевіряє в БД наявні процеси та працює на принципом наведеним на рисунку 5.11.

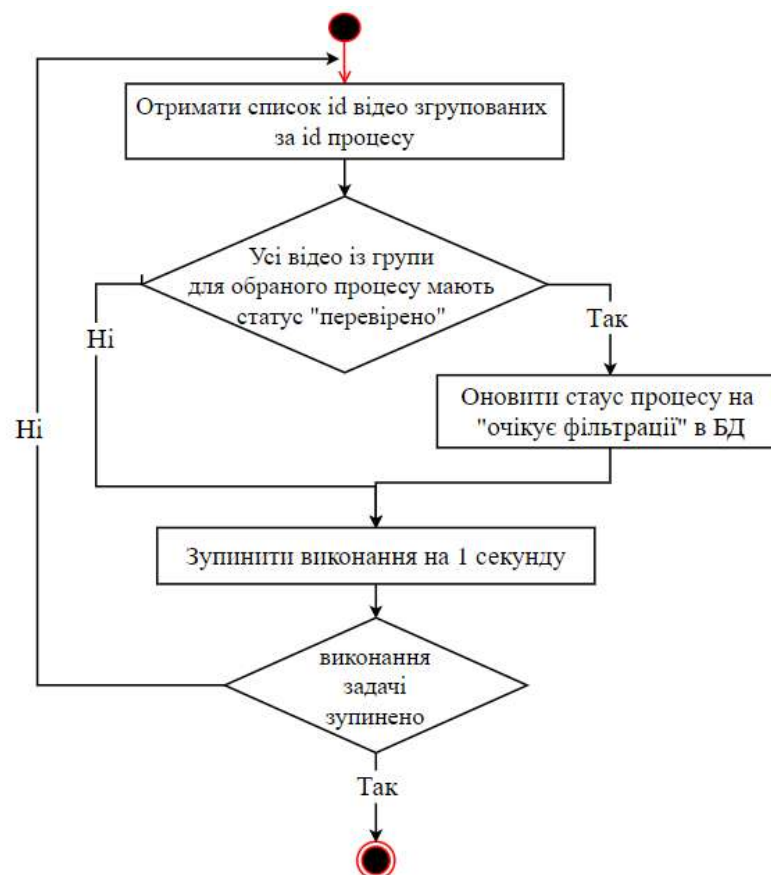


Рисунок 5.11 – Принцип роботи задачі з підготовки даних для фільтрації

Дана задача витягує з БД усі відео згруповані за процесом. Далі вона перевіряє якщо усі відео для поточного процесу були перевірені, тобто всі дані для них було зібрано у попередній задачі, тоді процесу надається відповідний статус та даний процес викликатиметься в задачі фільтрації та сортування даних відео, принцип роботи якої зображений на рисунку 5.12.



Рисунок 5.12 – Принцип роботи задачі сортування та фільтрації відео

Першим кроком є витягування даних відео, даних YouTube, канал, категорії, джерел, тегів для процесу, та джерел, тегів та категорій для відео, пов'язаних з процесом з БД. В якості дій користувача використовуються спеціальні оцінки користувачів з розроблюваної платформи, що також зберігаються в БД. Щоб зменшити число запитів до БД та знизити швидкість очікування усіх даних з БД, було використано налаштування Django annotate, prefetch_related та select_related, що використовують Inner Join для того щоб витягнути дані єдиним запитом та сформувати з них хеш таблицю. Далі ці дані надсилаються до рекомендаційної системи, яка складається з компонентів: обробка та групування даних; фільтрування даних; сортування даних.

Дані до рекомендаційної системи прийдуть у вигляді хеш таблиці, для того щоб було зручніше працювати з даними, було створено спеціальний шаблон структури даних. Тому перший крок в рекомендаційній системі – групування даних за процесом та за відео у спеціальний список, що потім буде використаний у фільтрації та сортуванні.

Оскільки дані з БД надходять із використанням join, тому із даних з різних таблиць формуються усі можливі комбінації та надсилаються у відповідь користувачу. Тому було розроблено групування за унікальним

ідентифікатором відео, унікальним ідентифікатором процесу та унікальним ідентифікатором користувача (для оцінок користувачів з розроблюваної платформи). Процес групування зображено на рисунку 5.13.

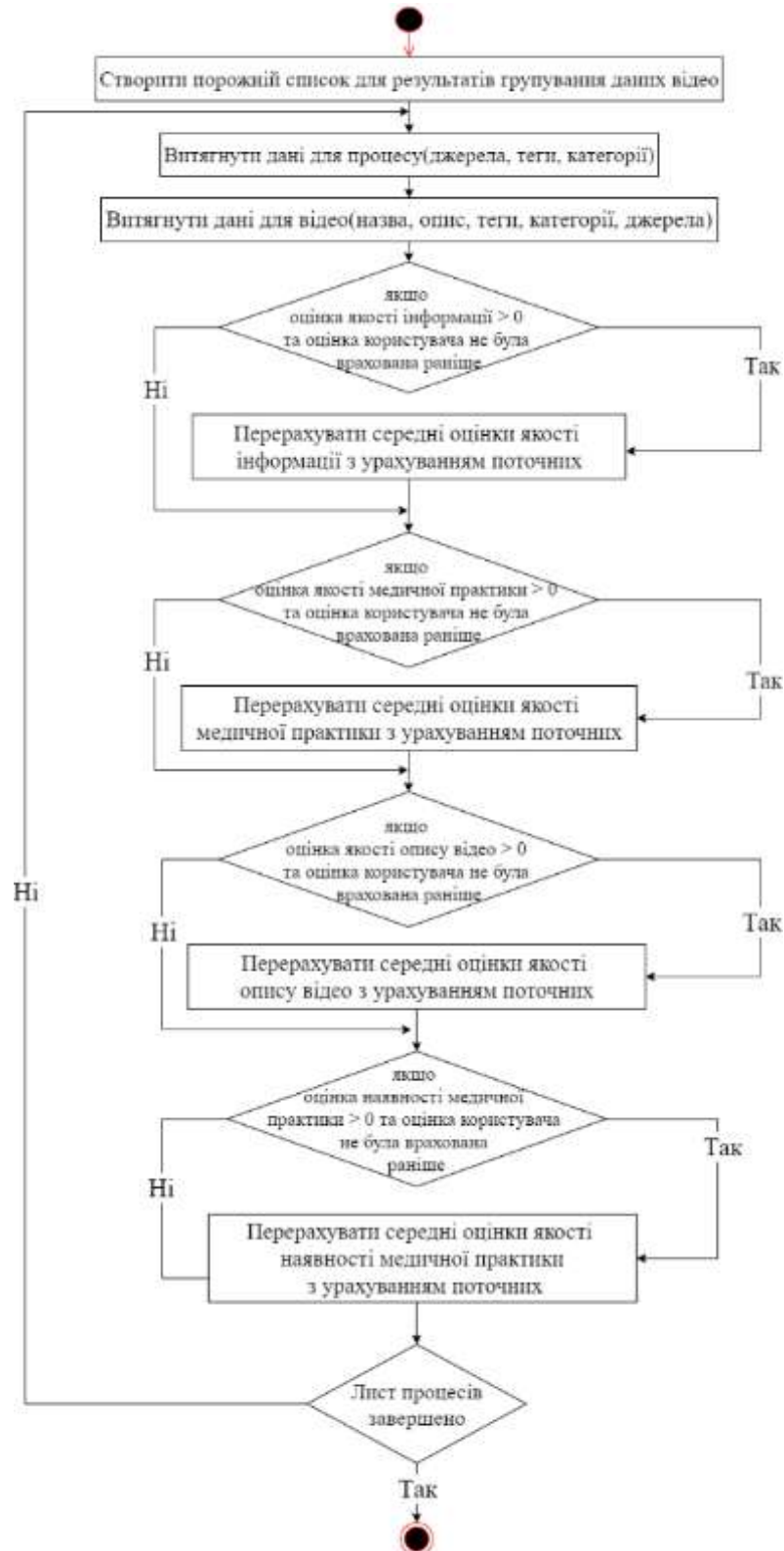


Рисунок 5.13 – Процес групування даних з БД у розроблений тип даних

Наступним кроком є процес фільтрації списку відео за процесом. Для процесів фільтрації нема потреби зберігати усі оцінки користувачів, бо це займатиме зайве місце в оперативній пам'яті та для сортування буде використано середнє значення за кожної з оцінок. Але постає проблема у випадках коли користувач виставляє не всі оцінки для відео, тому інші значення можуть дорівнюватись 0. Для цього було розроблено додаткову перевірку значень, тобто щоб порахувати середнє значення треба суми усіх оцінок певних груп поділити на число значень більших за 0. Принцип дії зображено на рисунку 5.14.

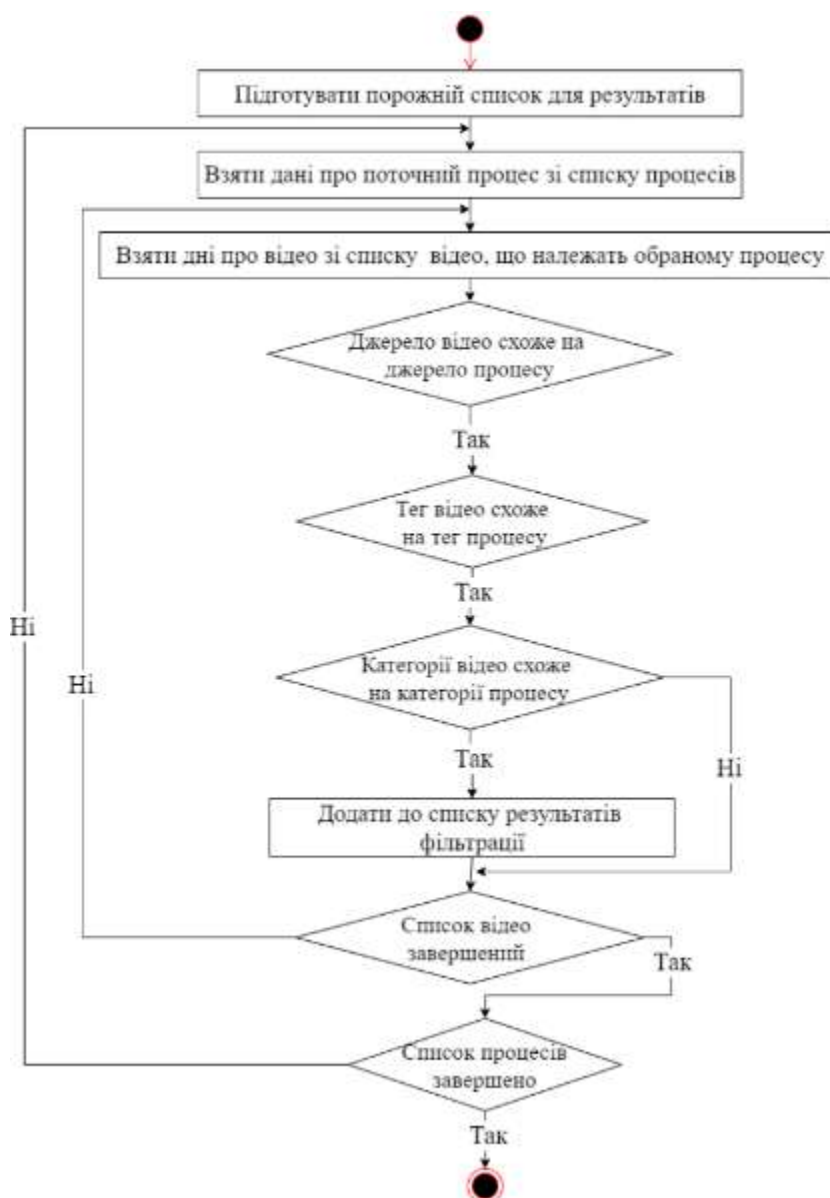


Рисунок 5.14 – Процес фільтрації списку відео за процесом

Для фільтрації було розроблено перевірку для кожного з джерел, категорій, тегів, для кожного відео та процесу. Якщо назва однієї з перелічених категорій відео містить або знаходиться в одній із назв категорій процесу, то це значить, що відео підходить під задані дані процесу пошуку.

Далі йде сортування відео, процес дії якої зображено на рисунку 5.15.



Рисунок 5.15 – Процес сортування списку відео за процесом

Для сортування використовується сума значень усіх обраних в розділі 3.1 критеріїв, а саме якість інформації, медичної практики, опису, наявність медичної практики, число позитивних та негативних оцінок з YouTube, число коментарів та переглядів з YouTube. Також наявність опису рахується за 1,

його відсутність за 0 для загальної суми. Такі самі умови для належності відео до обраних відео користувача, що створив процес.

Після завершення сортування процесу надається новий статус, що каже про успішне завершення пошуку, якщо була виявлена помилка, вона буде збережена у спеціальному атрибуті процесу. Також для кожного з відео надається порядок у групі, тобто оновлюється сутність процес-відео у відповідності із порядковим номером після сортування. Якщо відео не пройшло фільтрацію, то йому надається порядковий номер, що дорівнює 0 та не включається до списку коли йде запит на отримання даних за процесом.

Дана реалізація модулю асинхронних задач надає можливість швидко, та із використанням оптимальних ресурсів серверу знаходити відео за вхідними критеріями. В майбутньому можливо масштабувати кожну з задач, щоб ще краще використовувати ресурси системи.

5.4 Конструювання клієнтської частини

Для розробки клієнтської частини було використано фреймворк Vue.js. В даного фреймворку наявна спеціальна консольна утиліта `vue-cli`, тому клієнтську частину було згенеровано за допомогою неї. В якості параметрів було обрано `router` для створення маршрутів для компонентів, `vuex` в якості тимчасового сховища даних та `webpack` для контролю ресурсами. Загальна файлова система клієнтської частини складається з трьох основних пакетів, що наведено в таблиці 5.7.

Таблиця 5.7 – Опис головних частин клієнтської частини

Назва	Опис
build	Містить у собі файли для конфігурації проєкту для компіляції компонентів проєкту у код, що розпізнається браузером.
config	Містить у собі необхідні для проєкту налаштування, такі як адреса серверної частини, або назву директорії, куди буде компілюватися проєкт.
src	Головний пакет проєкту, містить у собі код проєкту, тобто усі компоненти, сторінки тощо.

Структуру головного пакету «src» зображено на рисунку 5.16.

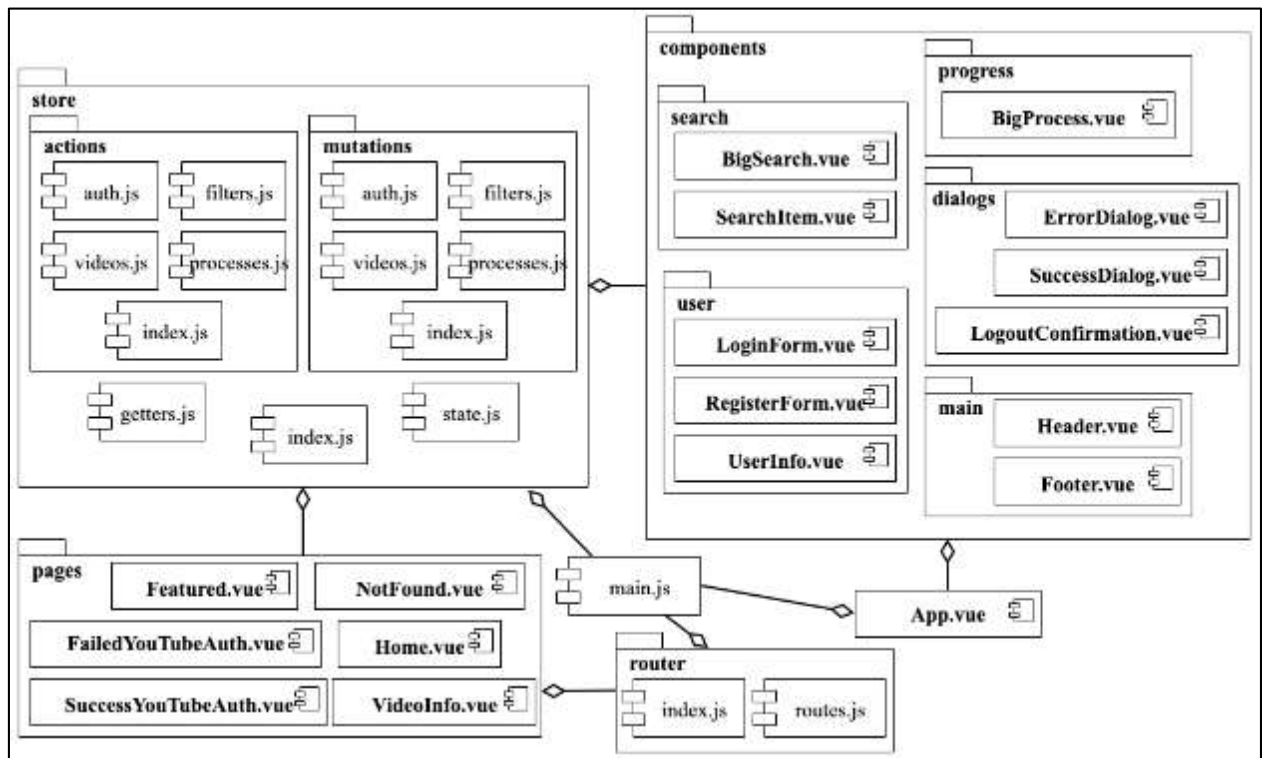


Рисунок 5.16 – Файлова структура клієнтської частини

Модуль «main.js» являється головним для клієнтської частини. В ньому підключені головний компонент App.vue, пакет router, що відповідає за реалізацію вебсторінок, та пакет store, що використовується для реалізації Vuex.js. В App.vue ініціалізовані компоненти, що знаходяться в пакеті components, що надає можливість використання їх у компонентах, що знаходяться в пакеті pages. Опис компонентів, що знаходяться в пакеті components наведено в таблиці 5.8.

Таблиця 5.8 – Опис компонентів пакету components

Назва	Опис компоненту
1	2
BigSearch.vue	Містить у собі поле для вводу тексту для пошуку та поля для обирання критеріїв пошуку.
SearchItem.vue	Являє собою елемент, що містить у собі базові дані про відео. Використовується у списках результатів відео.
LoginForm.vue	Модальна форма для авторизації на платформі.
RegisterForm.vue	Модальна форма, що містить у собі інструкції для реєстрації.

Продовження таблиці 5.8

1	2
UserInfo.vue	Модальна форма, що містить у собі інформацію про авторизованого користувача, з можливістю оновити ці дані.
BigProcess.vue	Спеціальне анімоване колесо, що використовується для зображення системи, що зараз вона опрацьовує якісь дані.
ErrorDialog.vue	Модальне вікно, що містить у собі інформацію про помилки зі сторони користувача. Використовується коли користувач намагається виконати дії авторизованого користувача, попередньо не авторизувавшись у системі.
SuccessDialog.vue	Модальне вікно про інформування користувача, про успішність виконання його операцій.
LogoutConfirmation.vue	Модальне вікно, що використовується для підтвердження виходу користувача з платформи.
Header.vue	Компонент, що знаходиться зверху кожної зі сторінок.
Footer.vue	Компонент, що знаходиться знизу кожної зі сторінок.

В клієнтській частині було створено три основні сторінки та три додаткові, в яких вказується тільки текст, що інформує користувача про результат виконаних дій. В таблиці 5.9 наведено детальний опис даних сторінок.

Таблиця 5.9 – Опис сторінок клієнтської частини

Назва сторінки	Назва компоненту	Опис
Головна	Home.vue	Основна сторінка. Містить у собі поле для пошуку та обирання критеріїв, статус пошуку відео, та список відео в разі успішного пошуку
Обране	Featured.vue	Сторінка для відображення списку обраних відео.
Дані відео	VideoInfo.vue	Сторінка для відображення даних відео.
Успішна авторизація в YouTube	SuccessYouTubeAuth.vue	Сторінка для інформування користувача, що авторизація в YouTube пройшла успішно.
Неуспішна авторизація в YouTube	FailedYouTubeAuth.vue	Сторінка для інформування користувача, що авторизація в YouTube не пройшла успішно.
Сторінку не знайдено	NotFound.vue	Сторінка для інформування користувача, про те, що даний продукт не має сторінки за обраним посиланням.

Реалізація сторінок можлива завдяки офіційному розширенню для Vue.js, що зветься vue-router [87]. Функціонал для даної реалізації було розміщено в пакеті router (рис. 5.17).

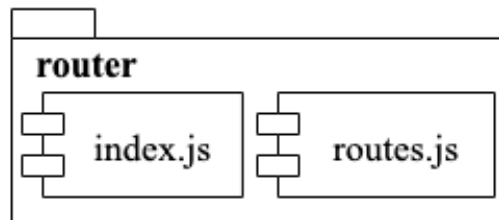


Рисунок 5.17 – Файлова структура пакету router

В модулі index.js відбувається ініціалізація об'єкту класу Router, що буде використаний в App.js. Об'єкт класу Router використовує маршрути, що вказані в routes.js, які використовують компоненти з пакету pages.

Для передачі даних між компонентами було обрано Vuex. Реалізація даного функціоналу було розміщено в пакеті store (рис 5.18).

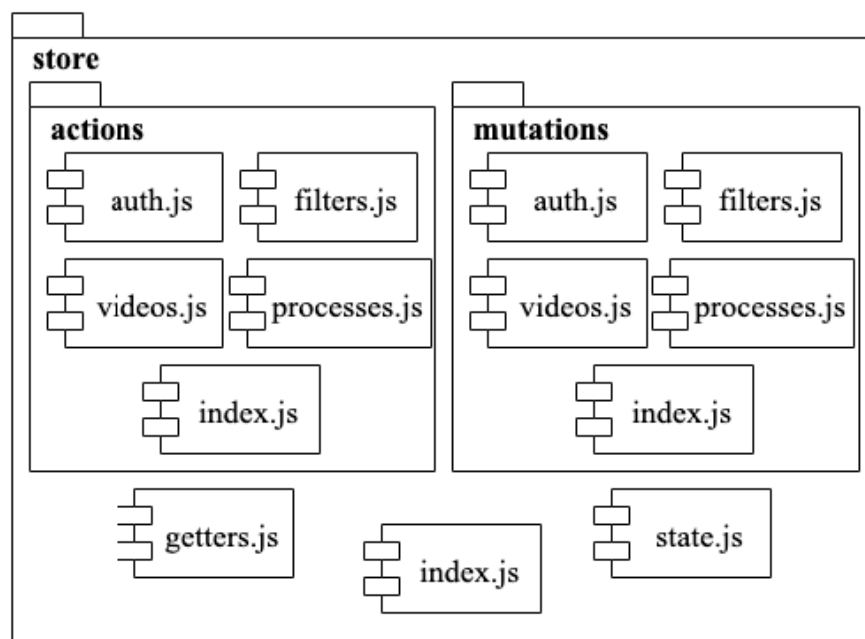


Рисунок 5.18 – Файлова структура пакету store

В пакеті actions було реалізовано логіку для взаємодії із програмними інтерфейсами серверної частини для управління акаунтами, відео, фільтрами та процесами та розміщено у окремі модулі за назвами auth.js, videos.js, filters.js та processes.js відповідно.

В пакеті mutations було реалізовано логіку для взаємодії із даними, що використовуються у клієнтській частині для управління даними про акаунти, відео, фільтри та процеси та розміщено у окремі модулі за назвами auth.js, videos.js, filters.js та processes.js відповідно.

В модулі state.js створено базову структуру даних, що використовувалася в клієнтській частині.

В модулі getters.js реалізовано методи, що дозволяють отримати дані із сховища Vuex.

В модулі index.js відбувається ініціалізація об'єкту класу VueX, для подальшого використання в App.vue. Завдяки цьому усі компоненти системи можуть взаємодіяти із усім функціоналом VueX.

В результаті було розроблено клієнтську частину, яка містить грамотно спроектовану файлову структуру, що забезпечить легкість її оновлення у майбутньому. Завдяки бібліотеці Vue.js розроблена клієнтська частина працюватиме швидко та не буде навантажувати веббраузер користувача. Також дана система протестована в різних веббраузерах та працює без візуальних проблем. Приклади коду наведено в додатку А

5.5 Конструювання засобів розгортання проєкту

В модулі асинхронних задач було прийнято рішення призупиняти поточне виконання задач для запобігання перенавантаженню сторонніх програмних інтерфейсів та блокування з їх сторін як наслідок. Але тимчасова зупинка задач зупинятиме поточний процес, що може призвести до нестабільної роботи системи. Щоб запобігти описаній ситуації було прийнято рішення запускати усі асинхронні задачі у окремих процесах.

Було обрано Docker-compose в якості системи контейнеризації, який використовує для створення та взаємодії із зображеннями систему Docker. Тому відповідно було створено два файли із налаштуваннями – Dockerfile та Docker-compose.yaml. Дані файли містять у собі опис та структуру системи, що буде запускатися на сервері. Завдяки даному рішенню усі частини системи працюють в окремих процесах, а саме в контейнерах, та можуть бути перезавантаженні у випадку проблем або неполадок на сервері.

Було розроблено Dockerfile, що містить опис для створення чотирьох зображень, що були використані в програмному продукті. Список наведено в таблиці 5.10.

Таблиця 5.10 – Опис зображень для контейнеризації проєкту

Назва	Опис
backend-base	Відповідає за збірку серверної частини, налаштування необхідних пакетів та підготування для використання в інших зображеннях, або одразу в контейнерах. Використовує за основу стандартне зображення python:3.8.
backend-gunicorn	Використовує за основу зображення backend-base та додає пакет gunicorn та команду для запуску серверної частини із використанням бібліотеки gunicorn.
frontend-build	Використовується для компіляції клієнтської частини для подальшого її використання в зображенні prod-nginx. Використовує за основу стандартне зображення node:12.2.0-alpine.
prod-nginx	Використовується для запуску вебсерверу. Використовує за основу nginx:1.16.0-alpine. Завдяки команді COPY --from зберігає скомпільовані файли клієнтської частини, для відображення їх користувачеві та використовує статичні дані (стили, зображення тощо), що знаходяться в контейнері backend для використання їх в панелі адміністратора.

Кожен із цих зображень було використано для конфігурації контейнерів у файлі docker-compose.yaml. БД також «підіймається» в окремому контейнері, для цього було використано стандартне зображення для MySQL версії «8.0». Список усіх восьми контейнерів наведено в таблиці 5.11.

Таблиця 5.11 – Опис контейнерів в системі контейнеризації програмного продукту

Назва	Використане зображення	Опис
mysql	mysql:8.0	Використовується в системі в якості СКБД.
fetch_base_data	backend-base	Використовується для задачі отримання базових даних відео.
fetch_full_data	backend-base	Використовується для задачі отримання повних даних про відео.
filter_videos	backend-base	Використовується для задачі фільтрації відео.
periodic_data_update	backend-base	Використовується для задачі для періодичного оновлення даних про відео.
prepare_for_filtering	backend-base	Використовується для задачі підготування даних до фільтрації.
backend	backend-gunicorn	Використовується для запуску програмних інтерфейсів серверної частини.
nginx	prod-nginx	Використовується для надання користувачу сторінок із клієнтською частиною для взаємодії із системою.

Усі контейнері підключені до єдиної мережі, що було створено в `docker-compose.yaml`. Це дозволить без додаткових налаштувань надіслати запити до інших контейнерів, використовуючи замість імені хосту ім'я контейнеру.

Також для серверної та клієнтської частини було налаштовано взаємодію завдяки налаштуванню `nginx`, що було винесено в окремий файл. В даному файлі описано усі маршрути для переадресації зовнішніх запитів на внутрішні.

Так наприклад для взаємодії із програмними інтерфейсами серверної частини треба надіслати запит до хосту, де налаштований програмний продукт та із використання піддомену `«/api/»`. У свою чергу сервер із `nginx` надсилатиме запит до контейнеру із серверною частиною. Це дозволить зробити єдину точку для взаємодії із системою, що дозволить відстежувати трафік та збирати необхідні метрики для системи.

Таким чином система контейнеризація налаштована для витримки непередбачуваних ситуацій від серверу, наділена можливістю для швидкого

налаштування та запуску на сервері та в разі необхідності перезавантаження усієї системи або окремих компонентів завдяки стандартним командам Docker та Docker-compose. Приклади коду наведено в додатку А.

5.6 Висновки до розділу 5

В даному розділі розглянуто конструювання розроблюваного програмного продукту, який складається з двох основних частин, а саме з клієнтської та серверної частини.

Для розробки серверної частини обрано фреймворк Django, тому структура проєкту створена за стандартами обраного фреймворку. Також для більш зручного створення програмного інтерфейсу було використано додаткову бібліотеку Django-REST-Framework (DRF). Вона необхідна для серіалізації та десеріалізації, тобто конвертації із об'єктів мови програмування Python в JSON та навпаки. Цей функціонал необхідний для реалізації надсилання відповідей до клієнтської частини у форматі JSON та для обробки запитів з клієнтської частини у даному форматі. Це дозволить стандартизувати формат даних, що передаються між сервісами.

У розділі наведено також опис використаних модулів в серверній частині, загальну файлову структуру, описи класів пакетів відео, фільтрації, процесів, а також пакету користувачів та авторизації.

Для забезпечення кращої якості результатів пошуку використовуються додаткові дані з профілю користувача на YouTube, для чого в системі розроблено авторизацію в системі YouTube.

Розробка модулю асинхронних задач дозволить знизити навантаження на систему, пришвидшити пошук, фільтрацію та сортування даних. Тому було спроектовано наступні задачі: періодичне оновлення відео; отримання базових даних по поточному запиту; отримання повних даних по поточному запиту; підготовка для фільтрації та сортування; фільтрація та сортування відео.

Оскільки асинхронні задачі будуть використовувати сутності БД, що

наведено в серверній частині, тому щоб уникнути дублювання коду, а саме повторного створення однакових сутностей, вирішено дотримуватись принципів Don't Repeat Yourself (DRY) та прийнято рішення створити для них додаток в основному проєкті.

Створено спеціальні статуси для процесів та відео, щоб асинхронні задачі взаємодіяли тільки з необхідними даними.

Реалізація модулю асинхронних задач надає можливість швидко, та із використанням оптимальних ресурсів серверу знаходити відео за вхідними критеріями. В майбутньому можливо масштабувати кожну з задач, щоб ще краще використовувати ресурси системи.

Для розробки клієнтської частини було використано фреймворк Vue.js. В даного фреймворку наявна спеціальна консольна утиліта `vue-cli`, тому клієнтську частину було згенеровано за допомогою неї. В якості параметрів було обрано `router` для створення маршрутів для компонентів, `VueX` в якості тимчасового сховища даних та `webpack` для контролю ресурсами.

В результаті було розроблено клієнтську частину, яка містить грамотно спроектовану файлову структуру, що забезпечить легкість її оновлення у майбутньому. Завдяки бібліотеці `Vue.js` розроблена клієнтська частина працюватиме швидко та не буде навантажувати веббраузер користувача. Також дана система протестована в різних веббраузерах та працює без візуальних проблем.

В якості системи контейнеризації обрано `Docker-compose`, що використовує для створення та взаємодії із зображеннями систему `Docker`. Тому відповідно було створено два файли із налаштуваннями – `Dockerfile` та `Docker-compose.yml`. Дані файли містять у собі опис та структуру системи, що буде запускатися на сервері. Завдяки даному рішенню усі частини системи працюють в окремих процесах, а саме в контейнерах, та можуть бути перезавантаженні у випадку проблем або неполадок на сервері.

6 КЕРІВНИЦТВО З ВИКОРИСТАННЯ ПРОГРАМНОГО ПРОДУКТУ

6.1 Призначення та умови використання програмного продукту

Програмний продукт являє собою вебдодаток, який призначений для користувачів, які є експертами в сфері медицини та намагаються знайти необхідне якісне відео на платформі YouTube. Розроблювана система покликана допомогти користувачам в вирішенні цієї проблеми. Так як завдяки даному продукту всі відео, що відповідають поставленим вимогам користувача проходять додатковий рівень фільтрації та сортування, що дозволяє в результаті надати користувачам відео максимально наближене до вимог пошуку.

Програмний продукт може бути використаний на будь-яких пристроях, де встановлений веббраузер та в наявності вихід до мережі інтернет.

6.2 Функції системи

Програмний продукт є вебдодатком, який повинен виконувати такі функції:

- зареєструвати чи авторизувати користувача в системі;
- надати користувачу інформацію для реєстрації в системі;
- надати функціонал для підтвердження використання даних профілю YouTube;
- надати користувачу доступ до інформації в його особистому кабінеті та можливість її редагування;
- надати користувачу список відео, який сформовано в результаті його пошукового запиту;
- надати можливість обрати відео з отриманого списку та переглянути його;

- надати користувачу можливість переглядати список обраного відео, а також додавати та видаляти відео з нього;
- забезпечити користувача засобами для оцінювання переглянутого відео.

6.3 Вхідні та вихідні дані

Вхідні дані – текст пошуку та обрані категорії, джерела та теги вказані за допомогою графічного інтерфейсу програмного продукту. Для пошуку користувач повинен бути авторизованим в системі. Після введення даних та початку пошуку програма інформує про поточний процес пошуку користувачу повідомленням на екрані.

У якості вихідних даних користувач отримає список відео, що максимально відповідає його бажанням та фільтрам, що було введено перед початком пошуку.

6.4 Налаштування програмного продукту

Для роботи з програмним продуктом, необхідно виконати ряд інструкцій з його запуску та налаштувань.

Спочатку потрібно впевнитись, що на комп'ютері встановлене програмне забезпечення для роботи з системою контролю версій Git. У випадку якщо воно відсутнє потрібно скористатись інструкціями з його встановлення наданими на офіційній сторінці Git [88].

Потім необхідно в консолі перейти в папку, в якій буде зберігатись проєкт та клонувати його з репозиторію. Після чого перейти за допомогою команди `cd` в необхідну директорію та встановити змінні оточення, що наведено в таблиці 6.1.

Таблиця 6.1 – Опис змінних оточення

Назва	Опис
YOUTUBE_CLIENT_SECRET	Секретний ключ для використання програмного інтерфейсу YouTube.
APP_SECRET_KEY	Секретний ключ, для налаштування безпеки системи.
DB_USER	Ім'я користувача для використання СКБД.
DB_PASSWORD	Пароль користувача для використання СКБД.
DB_NAME	Назва бази даних в СКБД
APP_HOST	ір адреса хосту або доменна назва. Використовується для налаштувань безпеки взаємодії серверної та клієнтської частини.

Наступним кроком є запуск системи командою: `docker-compose up --build`, попередньо упевнившись, що в системі встановлені Docker та Docker-compose.

Після цього треба підключитись до контейнера `backend` та в консолі створити користувача із правами адміністратора як наведено на рисунку 6.1.

```

→ youmed git:(develop) * docker exec -it backend_youmed python manage.py createsuperuser
[DEBUG] 2020-10-31 15:21:03,104 ::: Using selector: EpollSelector
Username (leave blank to use 'root'): administrator
Email address: admin@youmed.com
Password:
Password (again):
Superuser created successfully.
→ youmed git:(develop) * █

```

Рисунок 6.1 – Команди для створення адміністратора

Після виконання усіх кроків система налаштована та готова для виконання задач.

6.5 Взаємодія із програмним продуктом

Для використання програмного продукту необхідно авторизуватися в системі. Якщо спробувати почати пошук не авторизувавшись в системі, система проінформує користувача відповідним модальним вікно, про потребу в авторизації як на рисунку 6.2.



Рисунок 6.2 – Модальне вікно із проханням авторизуватися в системі

Для авторизації в системі треба натиснути відповідну кнопку в верхній частині сторінки, як наведено на рисунку 6.3.



Рисунок 6.3 – Верхня частина сторінки

Після натискання на кнопку відкриється модальне вікно як наведено на рисунку 6.4.

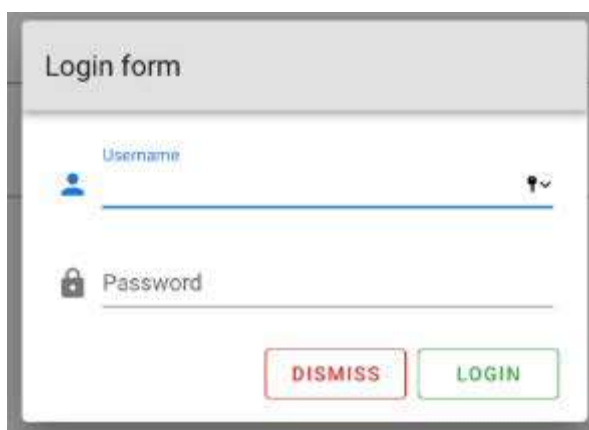
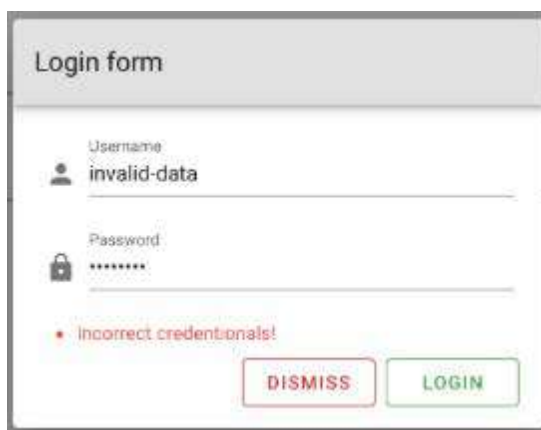


Рисунок 6.4 – Модальне вікно для авторизації

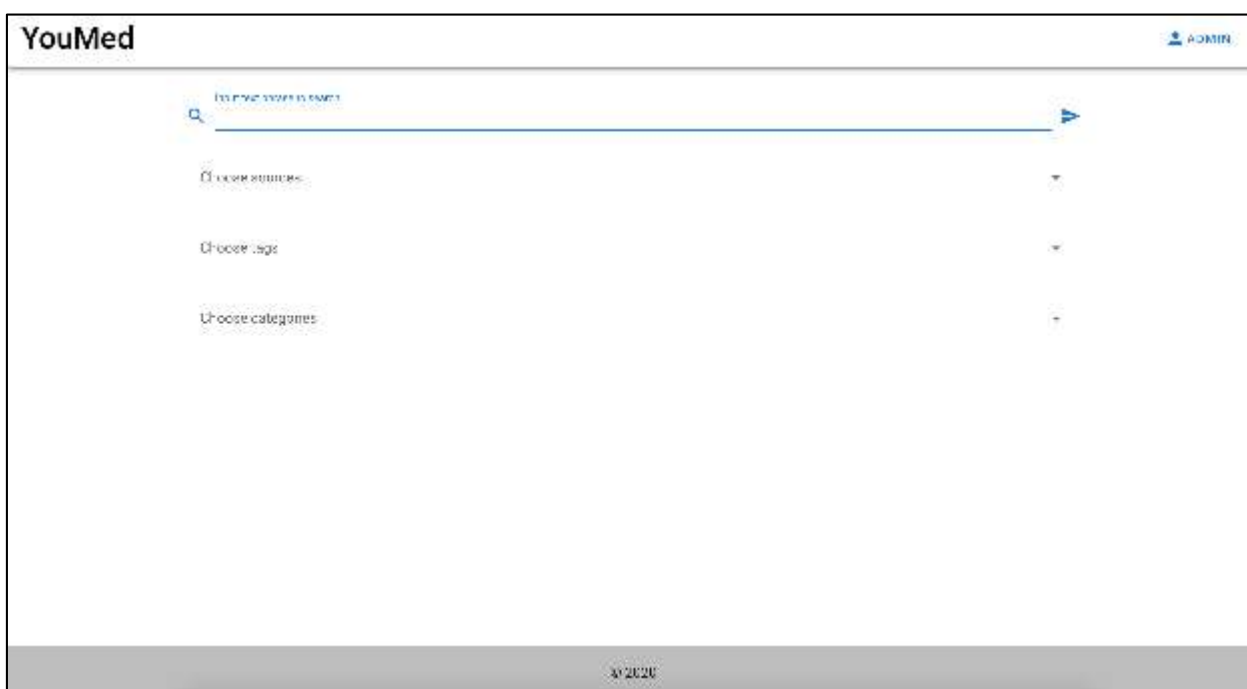
Для авторизації треба заповнити відповідні поля. Якщо ввести невірні дані користувача, то внизу форми буде зображено відповідну помилку як на рисунку 6.5.



The image shows a login form titled "Login form". It has two input fields: "Username" with the text "invalid-data" and "Password" with masked characters "*****". Below the fields is a red error message: "Incorrect credentials!". At the bottom right are two buttons: "DISMISS" (red) and "LOGIN" (green).

Рисунок 6.5 – Інформація про неправильні дані для авторизації

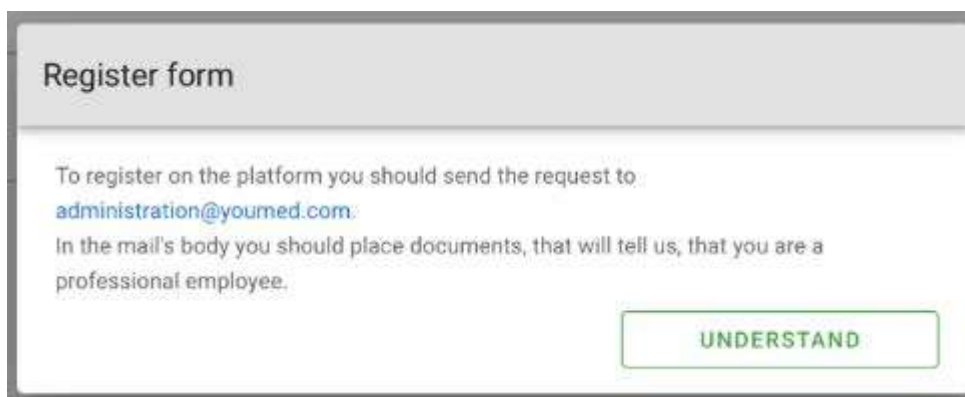
Після успішної авторизації користувача переадресує на головну сторінку, яку наведено на рисунку 6.6.



The image shows the main page of the "YouMed" system. The header includes the "YouMed" logo and a user profile icon labeled "ADMIN". Below the header is a search bar with the placeholder text "Your new search is empty". Under the search bar are three filter sections: "Choose medicines", "Choose tags", and "Choose categories", each with a dropdown arrow. The footer is a grey bar with the text "© 2020".

Рисунок 6.6 – Головна сторінка системи

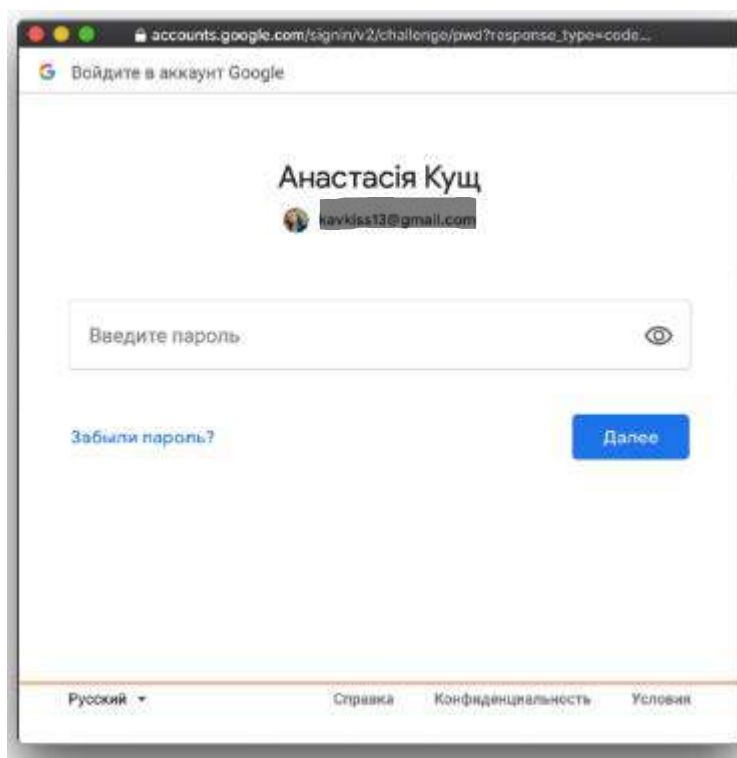
Якщо профіль користувача ще не створено в системі, то він повинен натиснути на кнопку «Register» на головній сторінці. Система надасть користувачеві список дій, у вигляді модального вікна, що треба виконати для створення його профілю, як наведено на рисунку 6.7.



The image shows a modal window titled "Register form". It contains the following text: "To register on the platform you should send the request to administration@youmed.com. In the mail's body you should place documents, that will tell us, that you are a professional employee." At the bottom right, there is a green button labeled "UNDERSTAND".

Рисунок 6.7 – Модальне вікно із інструкцією для реєстрації

Якщо користувач авторизується в системі вперше, то вона створює окреме вікно, де користувачеві треба ввести дані з його профілю на YouTube, щоб система могла використати ці дані для пошуку як на рисунку 6.8 та 6.9.



The image shows a Google login page in Russian. The browser address bar shows "accounts.google.com/signin/v2/challenge/pwd?response_type=code...". The page title is "Войдите в аккаунт Google". The user's name "Анастасия Куш" and email "kaykiss13@gmail.com" are displayed. There is a password input field with the placeholder "Введите пароль" and an eye icon. Below the password field is a link "Забыли пароль?". A blue button labeled "Далее" is on the right. At the bottom, there is a language selector set to "Русский" and links for "Справка", "Конфиденциальность", and "Условия".

Рисунок 6.8 – Авторизация в Google

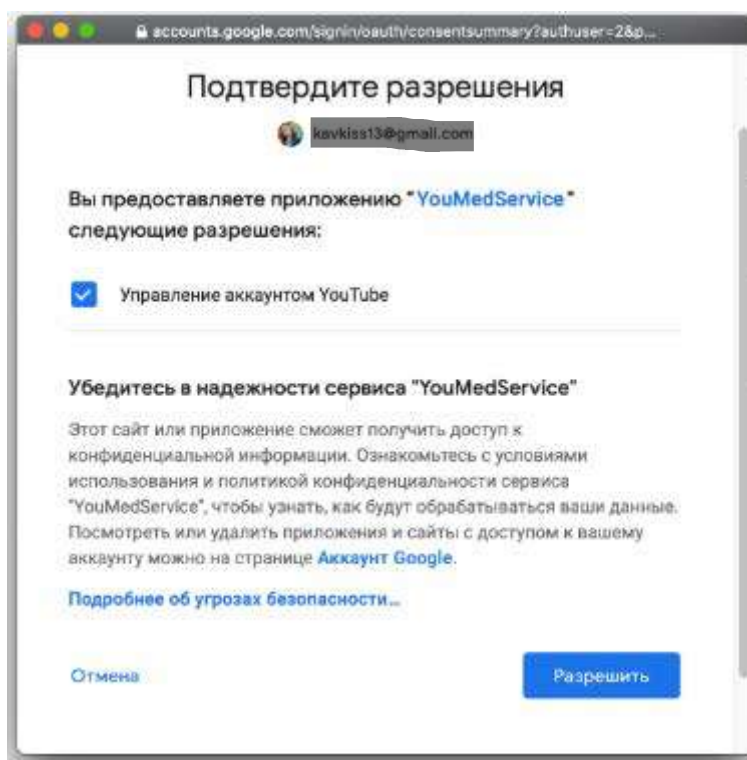


Рисунок 6.9 – Надання даних про YouTube профіль

У випадку якщо авторизація пройшла успішно, користувач отримає відповідне повідомлення яке наведено на рисунку 6.10. В іншому випадку – повідомлення про неуспішну авторизацію як на рисунку 6.11.

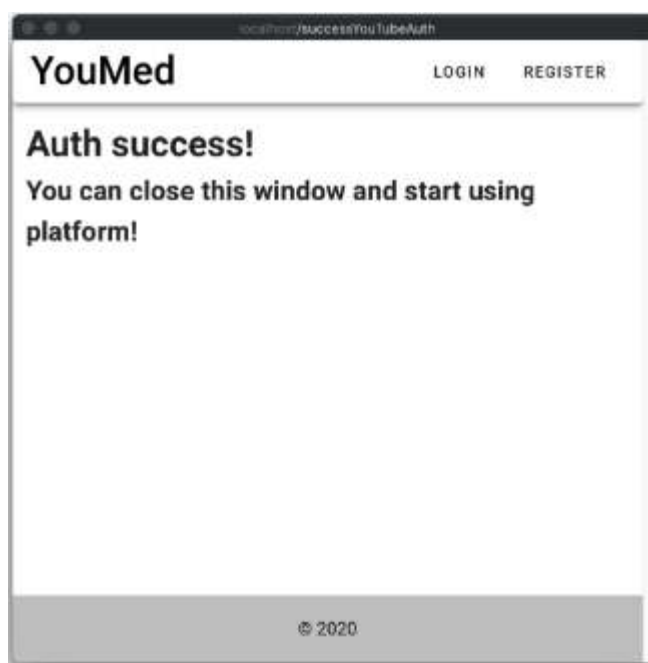


Рисунок 6.10 – Повідомлення про успішну авторизацію в системі YouTube

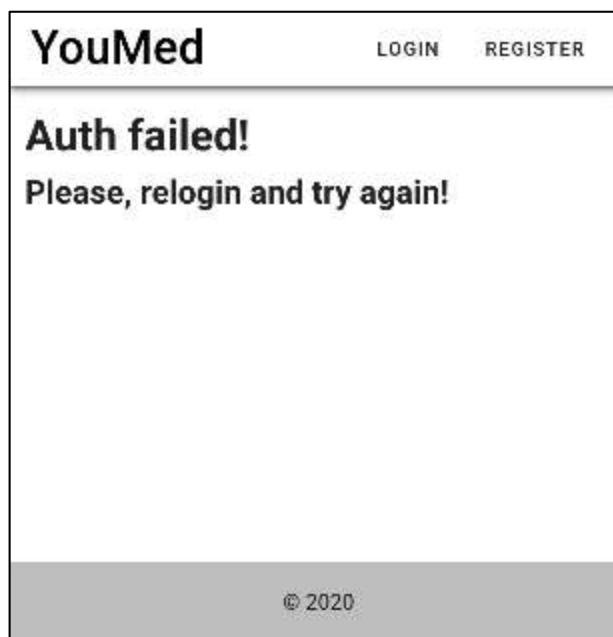


Рисунок 6.11 – Повідомлення про неуспішну авторизацію в системі YouTube

Після успішної авторизації користувач може переглянути інформацію у власному профілі (рис.6.12). Де він може перейти на сторінку з відео, яке він додав до списку обраного, за допомогою посилання, виконати редагування даних за допомогою кнопки Edit, а також вийти з системи – Log out.

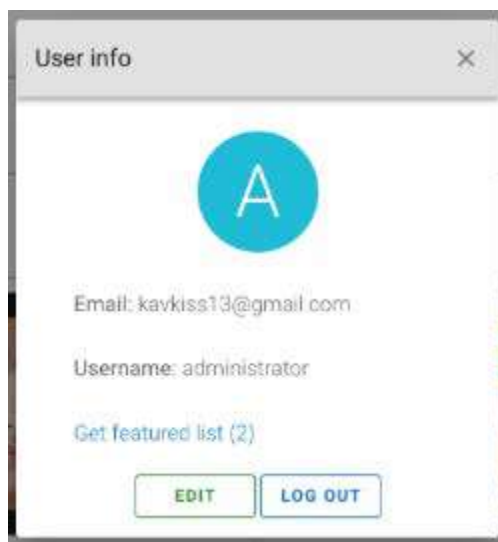


Рисунок 6.12 – Профіль користувача

Після успішної авторизації користувач може розпочати пошук за необхідними йому критеріями. Після введення певної інформації у поле вводу

розпочинається пошук відео як показано на рисунку 6.13.

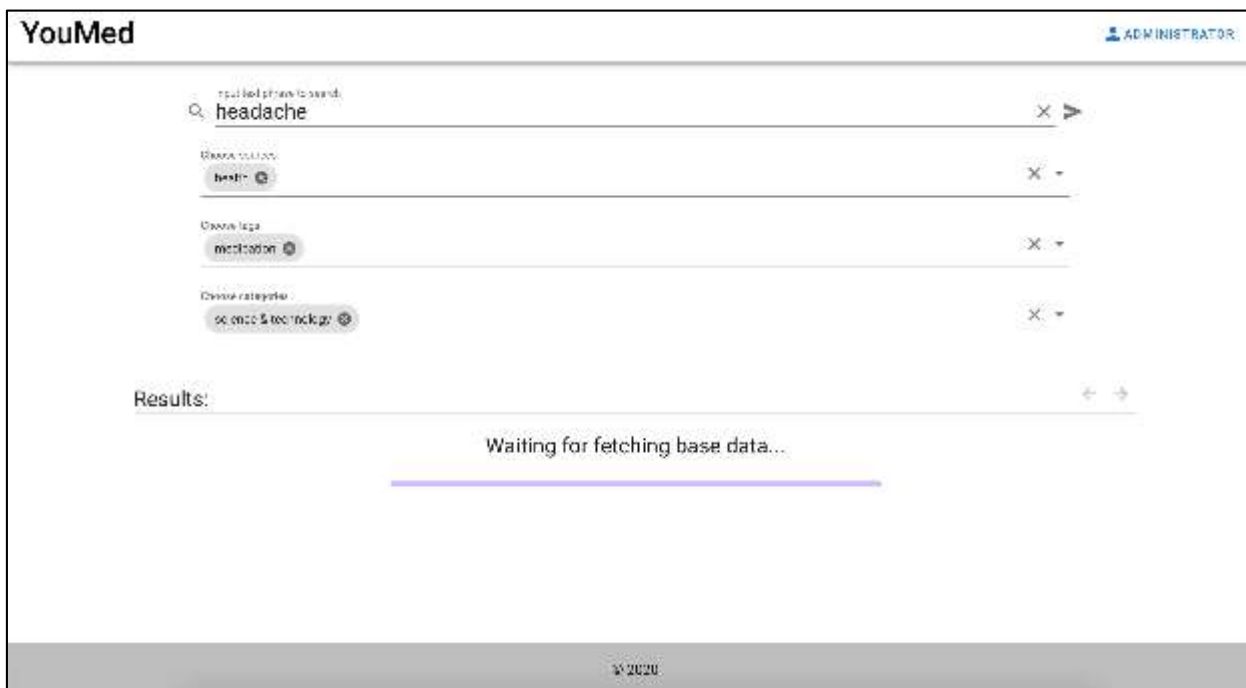


Рисунок 6.13 – Процес пошуку відео

Під час очікування результатів пошуку, користувач отримує інформацію про поточний статус пошуку у вигляді коротких повідомлень, під лінією Results. Він може отримати наступні повідомлення:

- «Waiting for fetching base data» – очікування на отримання базових даних;
- «Waiting for fetching full data» – очікування на отримання повних даних;
- «Waiting for filtering data» – очікування фільтрації.

Після завершення пошуку користувач отримає результат пошуку, а саме список відео, як наведено на рисунку 6.14.

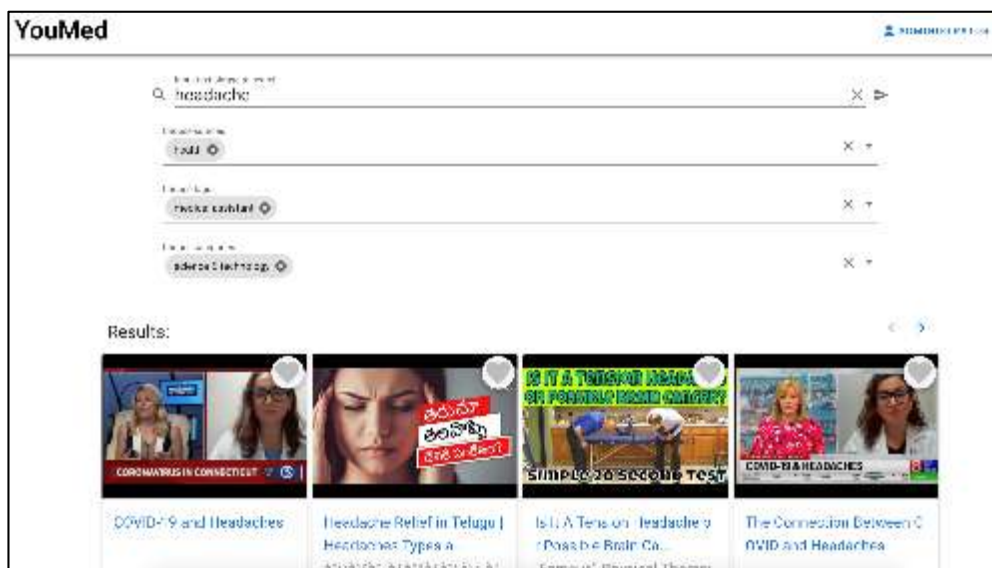


Рисунок 6.14 – Результат процесу пошуку відео

Після отримання результатів пошуку користувач має можливість обрати необхідне відео для перегляду. В результаті чого відкриється нова сторінка з детальною інформацією про нього, а також з системою оцінювання (рис. 6.15).

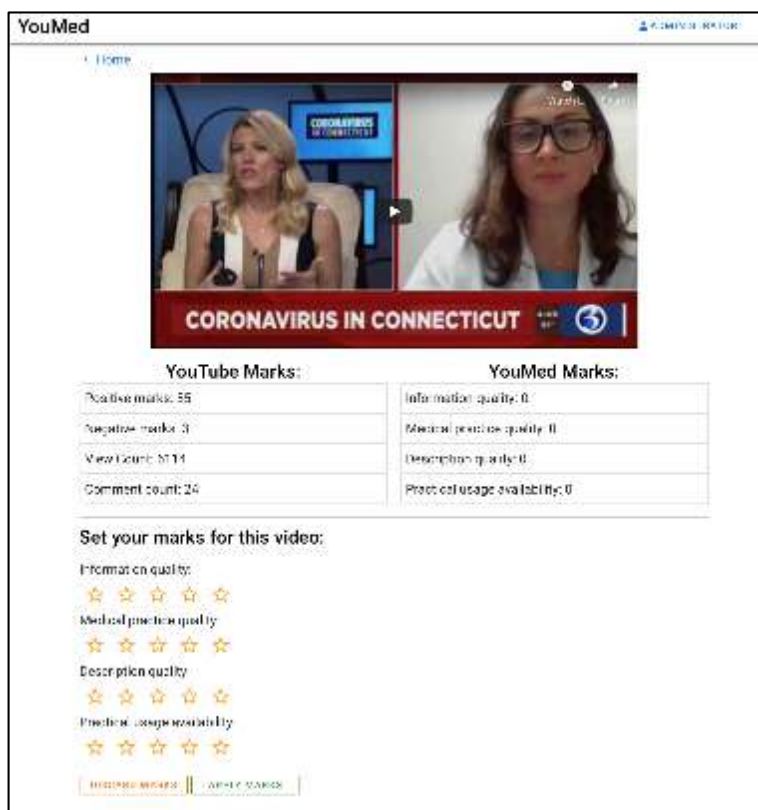


Рисунок 6.15 – Сторінка перегляду відео

Користувач має можливість оцінити відео, використавши спеціальну форму внизу сторінки, що наведено на рисунку 6.16.

Set your marks for this video:

Information quality:
☆☆☆☆☆

Medical practice quality:
☆☆☆☆☆

Description quality:
☆☆☆☆☆

Practical usage availability:
☆☆☆☆☆

DISCARD MARKS APPLY MARKS

Рисунок 6.16 – Оцінювання відео

Після оцінки користувачем відео, середня оцінка про відео одразу оновлюється, як наведено на рисунку 6.17.

YouTube Marks:

Positive marks: 55
Negative marks: 3
View Count: 6114
Comment count: 24

YouMed Marks:

Information quality: 0
Medical practice quality: 3.5
Description quality: 0
Practical usage availability: 0

Set your marks for this video:

Information quality:
☆☆☆☆☆

Рисунок 6.17 – Результат оцінки відео

Повернувшись на головну сторінку користувач має можливість додати відео до власного списку обраних відео, натиснувши на сердечко в правому верхньому куті на прев'ю відео, після натискання на яке воно стане червоним (рис. 6.18).

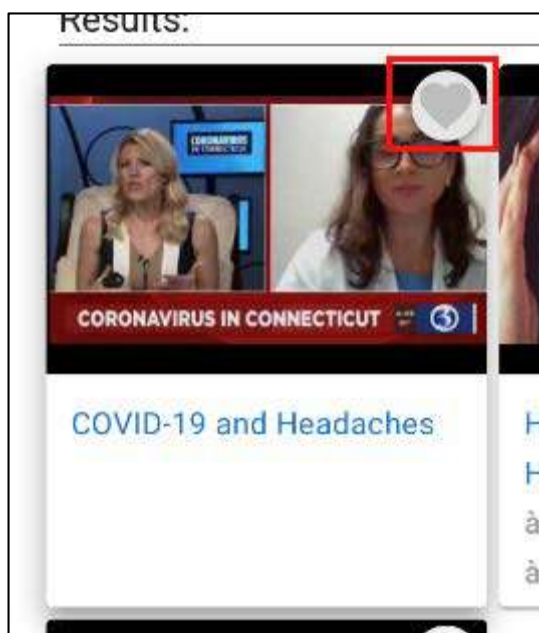


Рисунок 6.18 – Додавання відео до списку обраних

Для отримання списку обраних користувач повинен зайти до власного профілю та натиснути на посилання наведене на рисунку 6.19.

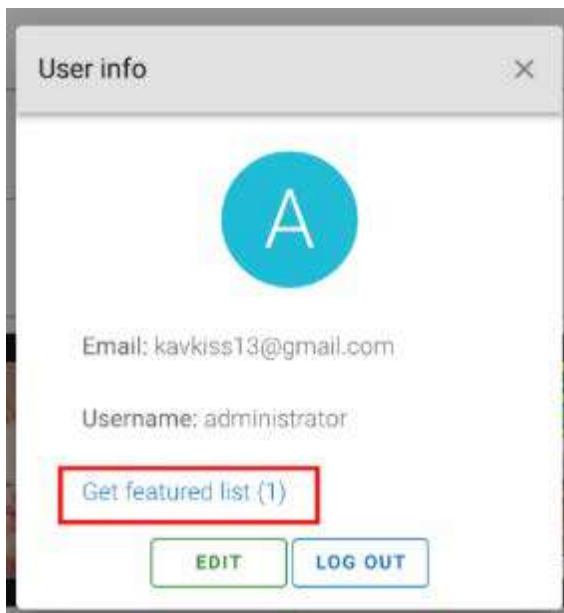


Рисунок 6.19 – Інструкція для відкриття сторінки зі списком обраного

Сторінку зі списком обраного відео користувача наведено на рисунку 6.20.

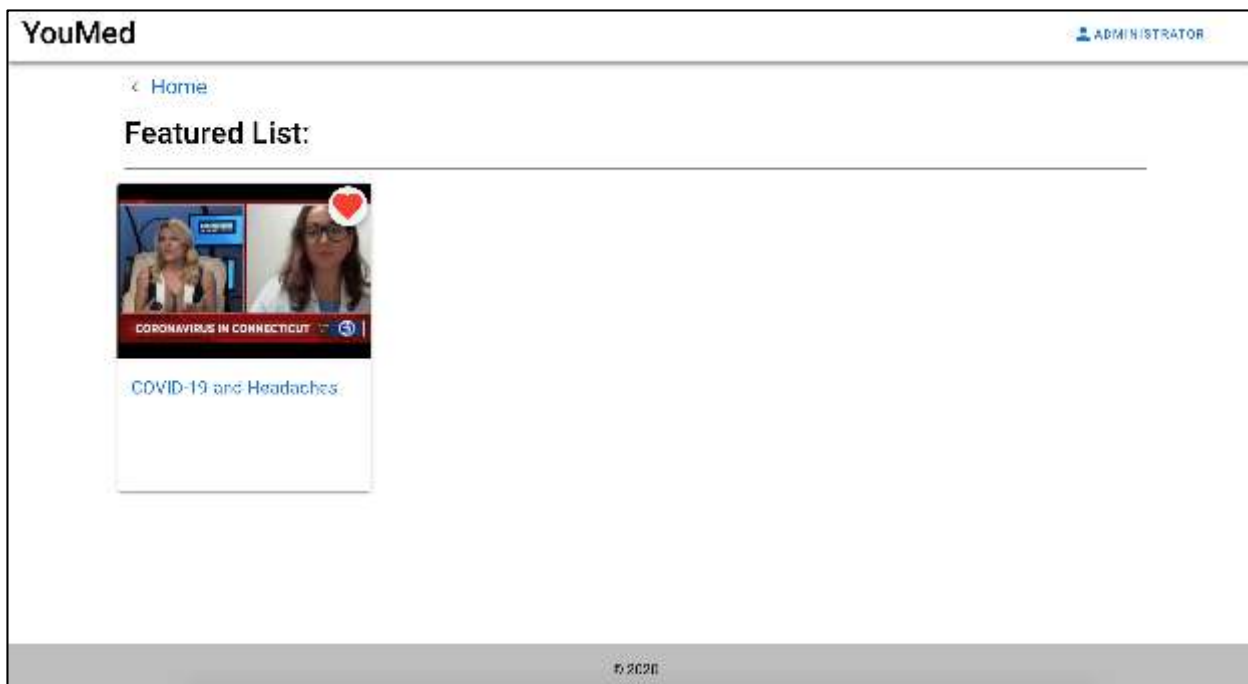


Рисунок 6.20 – Перегляд списку обраного відео

6.6 Висновки до розділу 6

У розділі розглянуто призначення та умови використання програмного продукту, вхідні та вихідні дані та функції системи. Також наведено детальну інструкцію для користувачів з встановлення та використання програмного продукту.

7 РОЗРОБКА ОРГАНІЗАЦІЙНО-ЕКОНОМІЧНОЇ ЧАСТИНИ

7.1 Оцінювання актуальності розробки

Перед початком створення будь-якого програмного продукту важливо провести аналіз ринку. Це дозволить зрозуміти чи є попит на даний продукт, визначити кількість потенційних клієнтів. Окрім того попередній аналіз дозволить прорахувати початковий грошовий капітал для старту, а також час за який проєкт окупиться та почне приносити прибуток.

Під час проведення оцінювання окремо фокусуються на виявленні доцільності втілення нових ідей технологій та їх здійсненності в промисловому масштабі. Зазвичай такі оцінки включають розгляд таких важливих питань як: характеристики ринку, переваги для споживачів; основні конкуренти; здійсненність ідеї.

Передумовами для розробки продукту на тему «Дослідження та програмна реалізація рекомендаційної системи для розширеної фільтрації відео» став стрімкий та неконтрольований зріст наявної інформації в Інтернеті. На сьогодні люди постійно звертаються до всесвітньої мережі за допомогою. Вони шукають необхідну інформацію, спілкуються, окрім того популярне онлайн навчання. В результаті ми отримали велику мережу для пошуку інформації, однак без необхідних алгоритмів для контролю та фільтрації її якості. Особливо гостро дана проблема виражене в сфері медицини. В таблиці 7.1 наведено опис ідеї даного програмного продукту.

Таблиця 7.1 – Опис програмного продукту

Зміст ідеї	Напрямки застосування	Вигоди для споживачів (користувачів)
Рекомендаційна система у вигляді клієнт-серверного програмного продукту для експертів в сфері медицини, що надає можливість додаткової фільтрації відео отриманого в результаті пошуку з платформи YouTube.	Медицина	Лікарі та інші працівники медичних закладів, викладачі вузів зможуть швидко знаходити якісне відео та використовувати його для наглядного пояснення діагнозу та саморозвитку.
	Заклади медичної освіти	
	Самоосвіта	

Таким чином вирішено розробити рекомендаційну систему у вигляді клієнт-серверного програмного продукту, який надаватиме спеціальний графічний інтерфейс для взаємодії із ним. Посилання на відео будуть зберігатися в базі даних та братись з платформи YouTube. Наповнення бази даних відео на початковому етапі буде здійснюватиметься за рахунок пошуків користувачів, які будуть експертами в сфері медицини (лікарі, академіки, викладачі). Вони матимуть можливість оцінювати відео даного ресурсу, а саме визначати якісність, наявність практичних сцен, достовірність матеріалу відео тощо. Програмний продукт надаватиме можливість звузити список відео під час пошуку до найбільш якісних. Таким чином вдасться створити систему для людей, які є експертами в сфері медицини, та прагнуть використовувати тільки якісні відео для своїх цілей.

Використовувати програмний продукт можна для навчання, наприклад, викладачі в медичних закладах освіти можуть знаходити необхідне відео та використовувати його в якості додаткових відеоматеріалів для пояснення теми. Лікарі можуть використовувати для пояснення пацієнтам їх діагнозу, щоб заспокоїти їх, роз'яснити важливість лікування, правильність виконання тих чи інших дій, підвищити рівень свідомості пацієнтів. Або ж будь-який експерт може просто використовувати для самоосвіти чи підтримки знань в актуальному стані.

7.2 Аналіз ринкових можливостей запуску проєкту

Актуальність створення даного програмного продукту, що здатний стати зручною платформою для багатьох спеціалістів, є очевидним фактом. Проте надзвичайно важливо, щоб умови ринку також були сприятливими для запуску реалізації ідеї продукту. В таблиці 7.2 наведено попередню оцінку ринку.

Таблиця 7.2 – Попередня характеристика потенційного ринку

№	Показники стану ринку (найменування)	Характеристика
1	Головні конкуренти	Vimeo
		YouTube
2	Динаміка ринку (якісна оцінка)	Зростає
3	Наявність обмежень для входу (вказати характер обмежень)	Відсутні. Оскільки продукт має відмінну від конкурентів нішу на ринку.
4	Специфічні вимоги до стандартизації та сертифікації	Відсутні

На сьогодні таких спеціалізованих аналогів продукту немає. Однак в якості конкурентів можна назвати дві популярних платформи – YouTube та Vimeo, які надають користувачам можливості роботи з відео, однак пропагують різні підходи для роботи та функціонування своїх платформ. Конкретність програмного продукту з даними платформами полягає в тому, що розробці необхідно перевершити популярність даних сервісів та надати якісніший контент.

Vimeo накладає на користувача суворі правила з користування платформою, тим самим забезпечуючи їх якісним контентом. Користувачі більш лояльні та залишають конструктивні та доречні коментарі. Платформа націлена більше на творчі напрямки.

YouTube став не просто розважальним сервісом, а ще й платформою на якій люди діляться своїми знаннями та використовують для навчання. В результаті, на його просторах можна знайти велику кількість навчального контенту, що представлений в різних формах. Нажаль, головним ризиком під час його використання для навчання є авторство, оскільки YouTube не накладає на користувачів практично жодних обмежень. Така свобода дій та відсутність механізмів перевірки якості відео, призвели до великої кількості неякісного контенту на платформі. Таким чином імовірність натрапити саме на такий контент висока для користувачів, яким може бракувати інформаційної грамотності, щоб оцінити якість знайденого контенту.

Програмний продукт є об'єднаною та покращеною версією даних платформ, оскільки він так само як Vimeo на перше місце ставить якість контенту, направлений на конкретну галузь та користувачів. Та завдяки YouTube має доступ до великої кількості відеоконтенту. В таблиці 7.3 наведено порівняння проєкту з конкурентами.

Таблиця 7.3 – Порівняння характеристик проєкту та конкурентів

№	Характеристики порівняння	Продукція конкурентів		
		Дипломний проєкт	Vimeo	YouTube
1	Швидкість роботи	висока	середня	висока
2	Зручність використання	висока	середня	висока
3	Точність результатів	висока	середня	середня
4	Кросплатформність	наявна	наявна	наявна

Проблема пов'язана з неякісним контентом особливо гостро стоїть в сфері медицини, оскільки дезінформація може призвести до негативних наслідків. В таблиці 7.4 наведено попередню характеристику потенційних клієнтів.

Таблиця 7.4 – Попередня характеристика потенційних клієнтів

№	Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до продукту (користувачів)
1	Потреба використання якісного відео в якості додаткового відео матеріалу під час навчального процесу.	Працівники медичних закладів освіти	Використовують для пошуку відео матеріалів для отримання нових знань.	Зручність у використанні. Швидка робота системи. Інтуїтивно-зрозумілий Інтерфейс. Надавати якісне відео.
2	Використання якісного відео для підтримки актуальності знань.	Працівники медичних закладів	Надання можливість користувачам системи поліпшити вже існуючі знання в сфері медицини.	

В таблиці 7.5 наведено SWOT-аналіз середовища реалізації інноваційного проєкту. За допомогою якого можна оцінити всі чинники, що можуть впливати на прийняття рішень, а також визначити можливості для розвитку.

Таблиця 7.5 – SWOT-аналіз

Сильні сторони:	Слабкі сторони:
Іноваційність. Актуальність. Легкість використання системи. Інтуїтивно зрозумілий інтерфейс. Швидкість роботи системи.	Довге виконання деяких запитів в БД, через частину неоптимізованих запитів. Неякісно розроблена масштабованість проєкту.
Можливості:	Загрози:
Впровадження іноваційних рішень. Використання нейронних алгоритмів. Розширення сфер використання	DDoS-атаки. Велика кількість користувачів онлайн. Символи у назві відео або каналу або у їх описі, що не відповідають кодуванню UTF-8.

В якості перспектив розвитку даної системи, можна назвати розширення меж системи, тобто перетворити її в систему, на якій можна буде знайти не тільки якісне відео в медичній сфері, але й в інших сферах. Розширення сфер стане можливим тільки за рахунок наявності експертів в тій чи іншій сфері, тому на майбутнє необхідно провезти детальний аналіз актуальних сфер та залучити спеціалістів.

7.3 Розробка ринкової стратегії проєкту

Розробка ринкової стратегії передбачає визначення стратегії охоплення ринку: опис цільових груп потенційних споживачів (табл. 7.6). Головною ціллю даного проєкту є забезпечення користувачів якісними відео матеріалами, за рахунок використання додаткової фільтрації та сортування списку отриманого відео з платформи YouTube. Таким чином можемо зробити висновок, що програмний продукт є спеціалізованим та направленим на знаходження та використання відео в навчальних цілях в сфері медицини.

Таблиця 7.6 – Вибір цільових груп потенційних споживачів

№	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1	Викладчі, що працюють в закладах медичної освіти	Висока	Середній	Низька	Висока
2	Працівники медичних закладів	Середня	Високий	Низька	Висока
Які цільові групи обрано: викладчі, що працюють в закладах медичної освіти, працівники медичних закладів.					

Відповідно до результатів отриманих в ході проведеного аналізу можна зробити висновок, що підходящою цільовою групою для розповсюдження даного програмного продукту є викладачі в закладах медичної освіти та працівники медичних закладів. Використання даного програмного продукту здатне полегшити роботу обраних цільових груп. Оскільки дозволить викладачам економити час на пошук потрібних матеріалів, а їх використання підвищить рівень засвоєння нових знань у студентів, крім того вони зможуть використовувати дані матеріали будь-де та будь-коли. Для працівників медичних закладів даний продукт стане корисним як для навчання так і для підтримки знань в актуальному стані, або ж у випадках коли необхідно пояснити діагноз або інші аспекти роботи в лікарні студентам, що проходять практику.

Таким чином в якості стратегії охоплення ринку збуту програмного продукту обрано стратегію масового маркетингу, оскільки для масової аудиторії в цілому надається стандартизований продукт з можливістю розширення функціональності.

7.4 Планування етапів розробки програмного продукту

Весь комплекс розробки програмного продукту можна розділити на етапи, для кожного з яких вказується кількість виконавців, трудомісткість та

тривалість робіт (табл. 7.7). У розробці даного проєкту беруть участь два спеціалісти:

- консультант, протягом 12 днів;
- програміст, протягом всього проєкту.

Терміни розробки програного продукту є початок – 2 вересня 2020 року, завершення – 23 листопада 2020 року. Тривалість розробки становить 60 днів, а кількість робочих днів на двох працівників 72, оскільки на певних етапах вони можуть працювати одночасно. Робочий день складає 8 годин.

Процес розробки продукту складається з восьми етапів, які детально описано в таблиці 7.7. Також за даними цієї таблиці побудовано діаграму Ганта, яка дозволяє візуалізувати процеси розробки програмного продукту (рис. 7.1).

Таблиця 7.7 – Етапи розробки продукту

Назва роботи	Трудомісткість		Виконавці	Тривалість, днів
	люд.-дні	% до підсумку		
1. Аналіз предметної області	10	13,89	Програміст, консультант	5
2. Визначення вимог до програмного продукту	10	13,89	Програміст, консультант	5
3. Дослідження особливостей розробки рекомендаційних систем	5	6,94	Програміст	5
4. Проектування системи	7	9,72	Програміст	7
5. Вибір інструментарію розробки	7	9,72	Програміст	7
6. Конструювання рекомендаційної системи.	27	37,5	Програміст	27
7. Тестування і налагодження програмного продукту	4	5,56	Програміст, консультант	2
8. Складання програмної документації	2	2,78	Програміст	2
Разом	72	100	X	60

Таблиця 7.8 – Розрахунок заробітної плати працівникам

Категорії працівників	Кількість осіб	Тарифна ставка за розрядом виконуваних робіт, грн/годину	Ефективний фонд робочого часу, годин	Тарифний заробіток, грн.
Основні працівники (програміст)	1	150	480	72000
Допоміжні працівники (консультант)	1	250	96	24000
Загальні витрати на виплати компенсації працівникам ($B_{зп}$)				96000

Значення для тарифної ставки попередньо обговорюється безпосередньо із персоною, що буде виконувати певний спектр робіт.

Обчислення ефективного фонду робочого часу (ЕФР) виконується за формулою 7.2:

$$\text{ЕФР} = K_{рд} * K_{рг}, \quad (7.2)$$

де $K_{рд}$ – тривалість роботи, роб. день;

$K_{рг}$ – тривалість одного робочого дня, год.

Оскільки тривалість роботи основних працівників 60 робочих днів, а допоміжних – 12 робочих днів, то завдяки даній формулі отримали значення ЕФР для даних категорій, що дорівнюють 480 та 96 відповідно.

Отримання тарифного заробітку (ТЗ) для кожної категорії виконується за формулою 7.3:

$$\text{ТЗ} = K_{пр} * \text{ЕФР} * \text{ТС}, \quad (7.3)$$

де $K_{пр}$ – кількість працівників певної категорії, люд.;

ТС – тарифна ставка за розрядом виконуваних робіт, грн/год.

Тому згідно з формулами ТЗ для основних працівників та допоміжних працівників дорівнює 72000грн та 24000грн відповідно.

Для отримання значення витрат на виплати компенсацій для працівників ($B_{зп}$) необхідно використати формулу 7.4:

$$B_{зп} = TЗ_1 + TЗ_2, \quad (7.4)$$

де $TЗ_1$ – тарифний заробіток для категорії основних працівників, грн.;

$TЗ_2$ – тарифний заробіток для категорії допоміжних працівників, грн.

В результаті $B_{зп} = 72000 + 24000 = 96000$ грн.

Для надання користувачам можливості для взаємодії із даним програмним продуктом, він потребує у віддаленому сервері (хостингу), де повинен бути запущений. В якості серверу використовується хостингова платформа Linode [89**Ошибка! Источник ссылки не найден.**].

Помісячна оплата за оренду ресурсів серверу (2 ядра ЦПУ, 4 ГБ оперативної пам'яті та 80 ГБ внутрішнього сховища) становить 600 грн. Річний фонд часу роботи ЕОМ (сервера) становить 12 місяців. Обчислення витрат на оренду серверу ($B_{серв.}$) виконується за формулою 7.5.

$$B_{серв.} = РФЧ * ПО, \quad (7.5)$$

де РФЧ – річний фонд часу роботи ЕОМ, місяців;

ПО – помісячна оплата за оренду серверу, грн/міс.

В результаті $B_{серв.} = 600 * 12 = 7200$ грн.

Витрати на матеріали ($B_{мат.}$) розраховуються за формулою 7.6:

$$B_{мат.} = \sum_{i=1}^n (Ц_i * N_i * (1 + K_{т.з.}) - Ц_{i0} * N_{i0}), \quad (7.6)$$

де $K_{т.з.}$ – коефіцієнт, що враховує транспортно-заготівельні витрати;

C_i – ціна i -го найменування чи комплектуючого, грн.;

N_i – потреба в i -му матеріалі, напівфабрикаті, комплектуючих;

C_{io} – ціна зворотних відходів i -го найменування матеріалу, грн.;

N_{io} – кількість зворотних відходів i -го найменування;

n – кількість найменувань матеріалів, напівфабрикатів та комплектуючих.

Витрати на матеріали для розробки наведені в таблиці 7.9.

Таблиця 7.9 – Витрати на допоміжні матеріали

Найменування, виміри, тирма, марка	Одиниця виміру		
		Ціна за одиницю вим., грн.	Кількість од. вим.
			Сума витрат, грн.

К а н ц е л я р і я	Н а б і р			
П а п і р ф о р м а т у А 4 д л я п р и н т е р а	Л и с т і в			
Всього				

З урахуванням можливих транспортних витрат (5% від вартості матеріалів), сумарна вартість матеріалів становить

$$B_{\text{мат.}} = 1,05 * 685 \text{ грн.} = 719,25 \text{ грн.}$$

В результаті значення витрат на розробку та впровадження програмного рішення $V_{\text{заг.}} = 96000 \text{ грн} + 7200 \text{ грн} + 719,25 \text{ грн} = 103919,25 \text{ грн}$. Дана сума не перевищує середнє значення вартості розробки подібних продуктів.

7.6 Висновки до розділу 7

У цьому розділі наведено економічну оцінку даного проєктного рішення. В результаті аналізу найближчих конкурентів та ринкових можливостей виявлено, що даний програмний продукт є конкурентоспроможним, оскільки він поєднує в собі переваги конкурентів та виключає недоліки, що дозволяє йому забезпечити користувачів якісним відеоконтентом, попит та актуальність якого на ринку на сьогодні висока.

Розробка ринкової стратегії для просування даного програмного продукту на ринку, дозволила виділити дві цільові групи споживачів: викладці, що працюють в закладах медичної освіти, працівники медичних закладів. З чого можна зробити висновок, що програмний продукт є спеціалізованим та направленим на забезпечення інформаційних потреб в сфері медицини.

Весь комплекс розробки програмного продукту було розділено на вісім етапів, що дозволило розділити задачі між виконавцями та визначити час на їх виконання. Для зручності виконання всіх процесів було розроблено діаграму Ганта, яка дозволила їх візуалізувати. Також інформації, по кожному з робітників, про кількість витраченого часу на розробку була використана для подальших розрахунків.

Було розраховано витрати на розробку та впровадження програмного продукту, в результаті отримали 103919,25 грн. Дана сума не перевищує середнє значення вартості розробки подібних продуктів або стартапів. Окрім того, завдяки якісно спроектованій та гнучкій архітектурі даного програмного продукту у майбутньому є можливість швидко та легко змінити сферу його використання з медицини на іншу. Такі маніпуляції дозволять створити на базі

вже існуючого продукту новий, що у свою чергу збільшить прибуток отримуваний від розробки. Враховуючи дані аспекти можна зробити висновок, що його розробка є економічно вигідною з перспективами для розвитку в майбутньому.

8 РОЗРОБКА ЗАХОДІВ З ОХОРОНИ ПРАЦІ ТА ЦИВІЛЬНОЇ БЕЗПЕКИ

8.1 Аналіз потенційних небезпек

У сучасному світі будь-яка галузь діяльності людини не обходиться без використання комп'ютерної техніки. Саме тому, питання поліпшення умов праці користувачів комп'ютерів сьогодні не тільки регулюється державою, а й створює конкуренцію між роботодавцями в питанні залучення кадрів. В Україні законодавством врегульовані норми та вимоги з використання комп'ютерної техніки на підприємстві, безпосередньо й охорона праці при роботі з комп'ютером [90].

Під час розробки програмістом будь-якого проєкту важливо забезпечити його необхідними заходами з охорони праці, протягом усього життєвого циклу розробки проєкту від проєктування до продакшену. Також важливо забезпечити раціональне розташування елементів робочого місця, що дозволить створити сприятливі та комфортні умови праці, уникнути стресових ситуацій, покращати продуктивність та забезпечити підтримку оптимальної робочої пози у працівника [91].

До потенційних небезпек, що можуть виникнути під час роботи за комп'ютером можна віднести [92, 93]:

- підвищення рівню шуму, за рахунок роботи вентиляційних компонентів комп'ютерної техніки, пристроїв введення/виведення, ксероксів, принтерів, кондиціонерів та інших електричних пристроїв;
- погано організоване, захаращене робоче місце, нераціональне розміщення обладнання;
- небезпека ураження електричним струмом внаслідок порушення правил експлуатації електроприладів або їх несправності;
- підвищення рівнів електромагнітних полів;
- підвищення рівню іонізуючого випромінювання;
- підвищений рівень статичної електрики;

- підвищення або зниження температури або вологості повітря, а також його надмірна загазованість або запиленість;
- погане або недостатнє освітлення робочого місця природнім або штучним джерелом світла;
- надмірно великий об'єм інформації, що необхідно обробити за одиницю часу;
- тривале сидяче положення;
- навантаження на органи зору;
- інтелектуальне та нервово-емоційним навантаженням;
- кістково-м'язове напруження, що виникає через необхідність працівників тривалий час знаходитись в позах, що вимагають статичного напруження м'язів спини, шиї, рук, ніг, що призводить до їх стомлення.

Для зменшення шансів виникнення представлених небезпек, важливо дотримуватись правильного режиму праці та відпочинку. Оскільки у разі їх порушення спостерігається незадоволеність роботою, роздратованості, збільшення скарг на часті головні болі, порушення сну, втому, больові відчуття в попереку, очах, руках та області шиї у працівників [91].

8.2 Заходи по забезпеченню безпеки

В офісних приміщеннях для роботи використовується велика кількість різноманітних електроприладів: принтери, ксерокси, сканери, комп'ютери, освітлювальні прилади, кондиціонери, обігрівачі, тощо, які живляться від мережі з напругою 220 В. Саме тому небезпека ураження електричним струмом є однією з найбільших небезпек, яка може виникнути в разі недотримання правил користування ними.

Щоб запобігти випадкам ураження електричним струмом відповідно до НПАОП 40.1-1.21-98 «Правила безпечної експлуатації електроустановок споживачів» частини обладнання, що проводять електричний струм та можуть стати джерелами небезпеки, надійно ізольовані та розміщені в недоступному

для співробітників місці. У випадку, якщо людина доторкнулась до неструмопровідних частин, які в наслідок пошкодженої ізоляції опинились під напругою, для забезпечення її безпеки, у приміщенні офісу використовується занулення та заземлення обладнання відповідно до ДСТУ 7237:2011.

В усіх приміщеннях, де розміщені робочі місця, заземлені конструкції (батареї опалення, водопровідні труби кабелі із заземленим відкритим екраном), надійно ізольовані діелектричними щиткам або сітками, щоб уникнути небезпечної ситуації потрапляння людини під напругу [95].

Відповідно до НПАОП 0.00-4.12-05, затвердженого наказом голови Держкомітету України з нагляду за охороною праці 26.01.2005 № 15, всі працівники проходять навчання та перевірку знань з питань охорони праці, як під час прийняття на роботу так і періодично, один раз на три роки.

До роботи допускаються тільки особи, які пройшли затверджений на підприємстві курс охорони праці для роботи з комп'ютером.

Для захисту працівників від ураження електричним струмом проводяться організаційні та технічні заходи захисту [96].

Організаційні заходи включають в себе [96]:

- призначення осіб, що будуть відповідати за організацію безпеки;
- навчання персоналу правилам безпеки та експлуатації обладнання;
- проведення інструктажів;
- використання попереджувальних плакатів і знаків (ДСТУ ISO 7010:2009).

До технічних заходів забезпечення безпеки працівників належать [96]:

- захисне заземлення, занулення, відмикання;
- електричний поділ мережі;
- вирівнювання потенціалів (ДСТУ 2815-94);
- обмеження сили струму;
- забезпечення недоступності неізольованих частин.

Для забезпечення безпеки працівників на підприємствах, де використовують комп'ютери забороняється [90, 94]:

- проводити ремонт та технічне обслуговування комп'ютера на робочому місці працівника;
- самостійно ремонтувати або намагатись виконати технічне налагодження комп'ютера без залучення компетентних спеціалістів;
- розміщувати на робочому місці зайві документи, деталі та предмети, що не потрібні для роботи;
- використовувати монітори з нечітким зображенням та монітори, у яких наявні поламки екрану;
- працювати з матричним принтером без антивібраційного покриття та зі знятою кришкою;
- експлуатація кабелів та проводів з пошкодженою ізоляцією або такою, що втратила захисні властивості в ході експлуатації;
- використання саморобних подовжувачів або нестандартного (саморобного) електронагрівального обладнання для опалення приміщення;
- користуватись пошкодженими розетками.

Також експлуатація електроустановок і електроустаткування проводиться згідно з НПАОП 40.1-1.01-97 «Правила безпечної експлуатації електроустановок» та НПАОП 40.1-1.21-98 «Правила безпечної експлуатації електроустановок споживачів», а також рекомендацій, що надавав виробник.

8.3 Заходи по забезпеченню виробничої санітарії та гігієни праці

Робочі місця офісних працівників, обладнані персональними комп'ютерами, відповідають вимогам «Правил охорони праці під час експлуатації електронно – обчислювальних машин», що затверджені Наказом Державного комітету України з промислової безпеки, охорони праці та гірничого нагляду від 26.03.2010 року №65 та «Державних санітарних правил і норм роботи з візуальними дисплейними терміналами електронно – обчислювальних машин», затверджених постановою Головного державного санітарного лікаря України від 10.12.1998 року №7 (ДСанПіН 3.3.2-007-98).

Дані правила поширюються на всіх суб'єктів господарювання незалежно від форм власності, які у своїй діяльності здійснюють роботу, пов'язану з персональними комп'ютерами [95].

Площу приміщень, в яких розташовуються персональні комп'ютери визначають у відповідності до чинних нормативних документів. Згідно з ДСанПіН 3.3.2.007-98 розмір одного робочого місця обладнаного персональним комп'ютером становить не менше 6м² [94]. Робочі місця розташовані на відстані не менше ніж 1м від стіни з вікном, а також 1,4 м від стіни, відстань між бічними поверхнями комп'ютера є не меншою за 1,2 м [97]. Суміжні робочі місця працівників, у зв'язку з потребою особливої концентрації уваги, розділені перегородками висотою 1,5 – 2 м [95].

Робочі місця заборонено створювати у підвальних або цокольних приміщення будинків. У приміщеннях, де працюють з комп'ютерами, щодня здійснюється вологе прибирання, з метою запобігання запиленості підлоги та меблів. Покриття підлоги є матовим з антистатичними властивостями, а сама поверхня рівною та неслизькою.

Правильна кольорова гама також забезпечує підвищення продуктивності та психологічному комфорту. Найбільш сприятливими для нервової системи є світлі та пастельні тони – зеленувато-блакитний, ясно-сірий, золотавий. У той час як яскраві, контрастні поєднання (синій і жовтогарячий, червоний і фіолетовий) будуть викликати втому та роздратування у працівників [97].

За потреби та з урахуванням площі приміщення в ньому можуть бути встановлені додатково шафи, полиці, тумби чи інші системами для зберігання документів або особистих речей співробітників [94].

Конструкція робочого місця користувача комп'ютером повинна забезпечити йому підтримку оптимальної пози для роботи та відповідати сучасним вимогам до ергономіка [94].

Стіл, за яким працює спеціаліст забезпечує оптимальне розміщення всього необхідного обладнання та документів на робочій поверхні. Для цього

він має висоту поверхні 680 – 800 мм, ширину 600 – 1400 мм і глибину 800 – 1000 мм. Такі параметри забезпечують можливість виконання операцій в зоні досяжності працівника [94, 95].

Робочий стілець працівника відповідно до п. 4.8 ДСанПіН 3.3.2.007-98 є підйомно-поворотним, з можливістю регулювання висоти та куту нахилу сидіння та спинки. Регулювання параметрів стільця здійснюватися легко та надійно фіксується. Поверхня для сидіння та спинка крісла вироблені з напівм'якого, нековзкого та повітронепроникного покриття, що легко чиститься та не електризується. Поверхня для сидіння є плоскою з заокругленим краєм [94, 95].

Для того, щоб знизити навантаження на органи зору між екраном комп'ютера та користувачем зберігається оптимальна відстань, що становить 600 – 700 мм, але не менше 600 мм з урахуванням літерно – цифрових знаків і символів. Також контролюється, щоб персональний комп'ютер та його комплектуючі не потрапляли під дію джерел тепла (батареї опалення, тощо) або під прямі сонячні промені [94, 95]. На рисунку 8.1 зображено приклад рекомендованого оформлення робочого місця [90].



Рисунок 8.1 – Рекомендовані параметри для робочого місця [90]

Тривала робота за комп'ютером може негативно вплинути на організм людини. Тому щоб мінімізувати ризики важливо мати правильно спроектоване робоче місце та виконувати ряд рекомендацій, що стосуються режиму праці та відпочинку. Саме тому працівникам рекомендується кожну годину робити

перерву мінімум на 15 хвилин, під час якої виходити на свіже повітря, виконувати не складні гімнастичні вправи. Під час же самої роботи кожні 10-15 хвилин намагатись змінювати позу, рухати плечами, ногами та руками [98].

Всі приміщення обладнанні системами опалення та кондиціонування повітря, або припливно-витяжною вентиляцією, відповідно до норм та правил ДБН В.2.5-67:2013 «Опалення, вентиляція та кондиціонування». У приміщеннях забезпечується контроль параметрів мікроклімату та виконується його нормування, відповідно до встановлених санітарно-гігієнічних вимог державних санітарних норми мікроклімату виробничих приміщень ДСН 3.3.6.042-99 [94]. В усіх на робочих місцях працівників повинні забезпечуються оптимальні параметри мікроклімату [94, 97, 98]:

- температура повітря в теплий період року становить 22–24°C, відносна вологість повітря — 40–60%, швидкість руху повітря — не вище за 0,2 м/с;

- температура повітря в холодний період року становить – 21-23°C, відносна вологість повітря — 40–60%, швидкість руху повітря — не більше 0,1 м/с.

Виконується регулярне очищення повітря, шляхом видалення забрудненого, нагрітого, іонізованого повітря з приміщення [98]. Оскільки здатність обчислювальної техніки виділяти суттєву кількість тепла в середовище, призводить до підвищення температури та до зниження рівня вологості в приміщенні [91]. Щоб мати змогу підтримувати необхідний об'єм чистого повітря в робочому просторі передбачено кондиціонування повітря. При недотриманні вказаних показників мікроклімату в офісних приміщеннях робочий день для робітників скорочується мінімум на 10% [94, 97].

Для комфортної роботи співробітників кожне приміщення забезпечено елементами природного та штучного освітлення, відповідно до ДБН В. 2.5-28-2018 «Інженерне обладнання будинків і споруд. Природне і штучне освітлення» та ДБН В.2.5-28:2018 «Природне і штучне освітлення». Оскільки

тривала робота за комп'ютером чи документами при поганому рівні освітленості може призвести до перенапруження органів зору [94].

Зазвичай використовують природнє бокове одностороннє освітлення в день, а у ввечері – загальне рівномірне штучне освітлення. Робоче місце розміщене так, щоб природнє світло було з лівого боку (п. 4.3 ДСанПіН 3.3.2.007-98) та уникати потрапляння прямого світла в очі [94]. Також у працівників є можливість для корегування рівня природнього освітлення за допомогою жалюзі або штор [90, 98]. Оскільки яскраве світло в районі периферійного зору може збільшити навантаження на органи зору та призвести до швидкої втоми, освітлення є рівномірним, тобто ступінь освітленості приміщення та яскравість монітору комп'ютера повинні бути приблизно однаковими [91].

Коефіцієнт пульсації не перевищує 5 % (п. 3.2.14 ДСанПіН 3.3.2.007-98). Рівень освітлення робочого столу в зоні розташування документів знаходиться в межах 300 – 500 лк. Освітлювальні прилади приміщення не повинні створювати відблисків на поверхні екрана, а освітлення екрана має не перевищувати 300 лк [94].

Правильно спроектоване та встановлене освітлення здатне покращити умови праці працівників, полегшити роботу органів зору, знизити втому, підвищити продуктивності та швидкості роботи, а недостатнє освітлення приміщення призводить до настання передчасної втоми, послабленню уваги та напруженню зору [91].

Підвищення рівню шуму погіршує умови праці та негативно впливає на організм людини. В результаті у робітників спостерігаються підвищений рівень стомленості, роздратованість, сильні головні болі, зниження концентрації уваги, нервово-психічне навантаження. Тривала дія інтенсивного шуму (вище 80 дБА) на слух людини призводить до його часткової або повної втрати [91].

Основними джерелами шуму на робочих місцях в офісах є принтери, кондиціонери, системи вентиляції та комп'ютери, в яких джерелами шуму є

вентилятори, що охолоджують систему. Рівень шуму та вібрацій на робочих місцях осіб, що працюють за персональним комп'ютером, визначають відповідно до ДСанПіН 3.3.2.007-98. Для забезпечення дотримання допустимих рівнів шуму на робочих місцях застосовуються засоби звукопоглинання, вибір яких обґрунтовується спеціальними інженерно-акустичними розрахунками (п. 3.3.3 ДСанПіН 3.3.2.007-98) [94, 98].

Також в офісних приміщеннях дотримуються вимог, що прописані у ДСанПіН 3.3.2.007-98 та ДСанПіН 3.3.6.096-2002 щодо рівня неіонізуючих електромагнітних випромінювань, електростатичних і магнітних полів, а також підтримують в нормі рівень інтенсивності потоків інфрачервоного та ультрафіолетового випромінювання [97].

Розрахунок освітленості офісного приміщення. Офісне приміщення знаходиться на четвертому поверсі п'ятиповерхового цегляного будинку, що відповідає нормам ДСанПіН 3.3.2-007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин».

Розміри приміщення становлять: довжина $A = 4$ м, ширина $B = 6$ м, висота $H = 2,7$ м. Площа дорівнює $S = 24$ м², а об'єм $V = 64,8$ м³.

Оскільки в кімнаті працює 2 особи, то на одного працівника припадає площа – 12 м² та об'єм – 32,4 м³, такі значення параметрів відповідають допустимим нормам вказаних у ДСанПіН 3.3.2-007-98.

В приміщенні наявне як природне так і штучне освітлення, що відповідає нормам ДБН В.2.5-28:2018 «Природне і штучне освітлення». Природне освітлення використовується в день, а у ввечері використовувати загальне рівномірне штучне освітлення.

Робоче місце розміщене таким чином, що природне світло знаходиться з лівого боку (п. 4.3 ДСанПіН 3.3.2.007-98), що дозволяє уникнути потрапляння прямого світла в очі. Джерелом природнього освітлення в приміщенні є вікна, на яких для корегування рівня природнього світла наявні жалюзі. Світильники загального і місцевого освітлення дозволяють

створюють нормальні умови освітленості і відповідний контраст між екраном і навколишнім оточенням з урахуванням виду роботи та вимог видимості з боку працівника.

В приміщенні наявні два вікна, через які відбувається проходження сонячного світла, їх розміри становлять 2 м в ширину та 1,5 м в висоту. Відповідно до даних розмірів площа кожного з них становить 3 м².

Щоб визначити коефіцієнт природної освітленості (КПО) необхідно скористатись наступною формулою:

$$K = \frac{S_B}{S_{\Pi}}, \quad (8.1)$$

де S_B – площа вікон,

S_{Π} – площа підлоги.

Для розрахунку коефіцієнта визначаємо площу двох вікон:

$$S_B = (2 * 1,5) * 2 = 6 \text{ м}^2.$$

Площа підлоги дорівнює: $S_{\Pi} = 6 * 4 = 24 \text{ м}^2$.

Тоді КПО дорівнює: $K = \frac{6}{24} = 0,25$.

Відповідно до вимог вказаних у ДБН В.2.5-28:2018 значення коефіцієнту природної освітленості повинно бути не менше 0,2. Отримане значення є більшим за мінімальну норму, що забезпечує достатнє природне освітлення в приміщенні.

Щоб перевірити чи відповідає рівень освітленості в приміщенні нормативним вимогам зазначених у ДБН В.2.5-28:2018 необхідно виконати розрахунок оцінки штучного освітлення в приміщенні. Для цього використовується метод коефіцієнта використання світлового потоку, що використовують при розрахунку загального рівномірного освітлення горизонтальних поверхонь з урахуванням відбитих від стін, підлоги і стелі світлових потоків.

Таким чином фактичне освітлення приміщення визначається за наступною формулою:

$$E_{\phi} = \frac{N * F_{\text{л}} * n * \eta}{S * z * k_3}, \quad (8.2)$$

де E_{ϕ} – фактичне освітлення в приміщенні, лк;

N – кількість світильників, яка дорівнює 4;

$F_{\text{л}}$ – світловий потік лампи, дорівнює 2340 лм;

n – кількість ламп у світильнику, дорівнює 4;

S – площа приміщення, що освітлюється, дорівнює 24 м²;

z – коефіцієнт нерівномірності освітлення для люмінесцентних ламп, дорівнює 1,2;

k_3 – коефіцієнт запасу, що враховує зниження освітлення через забруднення та старіння лампи, приймаємо, що він дорівнює 1,3;

η – коефіцієнт використання освітлювальної установки, 0,43.

Щоб визначити η потрібно знайти індекс приміщення i за наступним виразом:

$$i = \frac{A * B}{h_n * (A + B)}, \quad (8.3)$$

де A – довжина приміщення, м;

B – ширина приміщення, м;

h_n – висота підвісу світильників, м.

Тоді як висота підвісу h_n визначається наступним чином:

$$H_n = H - h_c - h_p, \quad (8.4)$$

де H – висота приміщення, м;

h_c – відстань від світильника до стелі, м;

h_p – висота умовної робочої поверхні, м.

Підставивши значення в формули (8.3) та (8.4), отримаємо наступні показники:

$$h_n = 2,7 - 0,3 - 0,7 = 1,7 \text{ м},$$

$$i = \frac{6 * 4}{1,7 * (6 + 4)} = 1,41.$$

Враховуючи, те що кольорова гамма приміщення складається з світлих відтінків, тип світильника стельової установки та отримане значення індексу приміщення $I = 1,41$ розраховуємо, що коефіцієнт використання освітлювальної установки становить $\eta = 0,43$.

Підставивши всі відомі значення до формули (8.2), розраховуємо фактичне освітлення приміщення:

$$E_f = \frac{4 * 2340 * 4 * 0,43}{24 * 1,2 * 1,3} = 430 \text{ лк.}$$

Отриманий результат порівнюємо з ДБН В.2.5-28:2018, де вказано, що рівень освітленості в приміщенні не повинен бути нижчим за 330 лк, виходячи з чого можемо зробити висновок, що рівень штучного освітлення у приміщенні є достатнім.

8.4 Заходи з забезпечення безпеки у надзвичайних ситуаціях

8.4.1 Заходи з пожежної безпеки

Згідно ГОСТ 27331-87 (СТ СЭВ 5637-86) «Пожарная техника. Классификация пожаров» у офісних приміщеннях, де знаходяться легкозаймисті матеріали (папір, текстиль, пластик, дерево) та електрообладнання (комп'ютери співробітників, мобільні телефони, кухонна техніка), можлива пожежа класів – А (горіння твердих речовин) та Е (горіння електроустановок, що перебувають під напругою) [99]. Саме тому заходи по забезпеченню пожежної безпеки є невід'ємною частиною трудового процесу

в офісних приміщеннях. Цю сферу регламентують Правила пожежної безпеки в Україні, затверджені наказом Міністерства внутрішніх справ України, зі змінами, які періодично вносяться відповідними наказами [100].

Для зменшення вірогідності виникнення пожеж впроваджено правила пожежної безпеки та встановлені пожежні сигналізації [99]. Кожен співробітник повинне неухильно дотримуватись всіх встановлених та прописаних вимог до пожежної безпеки. За дотриманням працівниками встановлених вимог, слідкують уповноважені та назначені керівництвом спеціалісти. До організаційних заходів по забезпеченню пожежної безпеки входять [100]:

- створення умов безпечної праці;
- мінімізація ризику виникнення пожеж;
- своєчасне і повноцінне забезпечення технічними засобами, для запобігання та усунення пожеж;
- контроль дотримання вимог та норм законодавства для попередження пожеж;
- розробка і впровадження регламентів по гасінню пожеж, евакуації та порятунку з місць пожежі й задимлення людей і майна (матеріальних цінностей);
- внутрішнє і зовнішнє навчання співробітників та регулярна перевірка знань працівників.

Всі офісні приміщення відповідають вимогам пожежної безпеки та є відповідно обладнаними [9999].

Усі наявні в офісі лінії електромережі є захищеними від виникнення короткого замикання та перепадів мережевої напруги, яка здатна спричинити збій в роботі техніки. Під час монтажу та експлуатації ліній електромережі повністю виключили виникнення електричного джерела загоряння внаслідок короткого замикання та перевантаження проводів. Обмежили використання проводів з легкозаймистою ізоляцією та намагаються використовувати їх з негорючою ізоляцією. Аварійний резервний вимикач розміщений на видному

та доступному місці, що при необхідності мати можливість повністю вимкнути електричне живлення приміщення, окрім світла [95].

Документи, папір та інші горючі матеріали розміщуються на відстані, не менше як 1,0 м — від електрощитів, електрокабелів, проводів; 0,5 м — від світильників; 0,25 м — від приладів опалення. Меблі та обладнання у приміщеннях розміщені так, щоб забезпечити вільний евакуаційний прохід, який завширшки не менше 1,0 м. Всі евакуаційні шляхи та виходи знаходяться в належному стані є вільними та нічим не заставленими [101].

Евакуаційні виходи наявні у кожному приміщенні, а їх кількість залежить від кількості працівників та площі приміщення. Всі вони обладнані надписом (таблом) «Вихід», а по маршруту евакуації розташовувані вказівні стрілки, які виведуть назовні. Двері на шляхах евакуації відкриваються у напрямку виходу з будівлі, що дозволить не перешкоджати виходу людей. [99].

Приміщення (окрім тих, де розташовуються сервери) оснащені системою автоматичної пожежної сигналізації та вогнегасниками [95]. Вогнегасники встановлені в помітних та легкодоступних місцях, а також в тих де є імовірність виникнення пожежі. При цьому важливо забезпечити їх захист від потрапляння прямих сонячних променів та дії опалювальних та нагрівальних приладів. Вибір типу та необхідна кількість вогнегасників визначається відповідно до вимог НАПБ Б.03.002-2004 «Типові норми належності вогнегасників» [102].

У офісних приміщеннях, що обладнанні електротехнікою та належать до класу пожежі А (Е), передбачені засоби пожежогасіння. Відповідно до ДСТУ 4297:2004 «Пожежна техніка. Технічне обслуговування вогнегасників. Загальні технічні вимоги» для гасіння електроустановок, що знаходяться під напругою, передбачені вуглекислотні вогнегасники типу ВВК-3,5 у кількості 2 штук. Відстань між вогнегасниками та місцями можливих загорянь не перевищує 10 м

Крім того, в офісах з оргтехнікою на 20 м² площі підлоги, передбачено

по одному газовому вогнегаснику з величиною заряду вогнегасної речовини 3 кг і більше. Також додатково приміщення оснащують пристроєм вогнегасним водонапірним аерозольним (ВВПА), з масою заряду вогнегасної речовини 400 г і більше [103].

Щоб знизити ймовірність виникнення пожеж в офісних приміщеннях необхідно дотримуватись всіх заходів та правил знаходження в приміщенні, а також проводити регулярне навчання та контроль якості знань співробітників. Важливо регулярно перевіряти та підтримувати в належному стані засоби пожежогасіння. Виконання всіх рекомендацій дозволить запобігти пожежі, а у разі її виникнення створити всі умови для її гасіння, захисту та успішної евакуації людей.

8.4.2 Заходи з цивільного захисту. Укриття населення у захисних спорудах цивільного захисту

Укриття населення в захисних спорудах є комплексом заходів із їх завчасним будівництвом, обладнанням приміщень та підтримкою їх у готовності до використання. Вони здатні захистити населення від наслідків дії ядерної, хімічної, бактеріологічної та звичайної зброї, від аварій та катастроф, деяких стихійних лих, військових дій або терористичних актів.

В разі виникнення надзвичайних ситуацій усе населення України повинно укритись в захисних спорудах, що входять до фонду захисних споруд. Створення фонду відбувається шляхом обстеження та обліку наземних та підземних будівель та споруд, що відповідають поставленим вимогам захисту, оцінюється необхідність їх дообладнання або переобладнання, за потреби проводиться будівництво нових сховищ та протирадіаційних укриттів [104].

За своїм призначенням та захисними властивостями захисні споруди поділяються на сховища, протирадіаційні укриття та найпростіші укриття.

Сховища є надійним захистом від небезпечних чинників, що виникають в результаті надзвичайних ситуацій. Вони представляють собою

комплекс інженерних споруд, що забезпечують необхідні умови життєдіяльності протягом певного часу [105].

За захисними властивостями від дії вибухової сили сховища поділяються на п'ять класів. За часом побудови на: побудовані завчасно і швидко споруджені. За наявністю фільтровентиляційного обладнання бувають промислового виготовлення та спрощеного (виготовлені з підручних матеріалів) [104]. За місцем заходження: вбудовані, ті що знаходяться в підвальних приміщеннях будинків, та відокремлені, які знаходяться поза будинками. Зазвичай їх будують заздалегідь та підтримують у належному стані, однак вони також можуть будуватись і в період загрози нападу або під час воєнних дій (швидко зведені). Сховища також класифікують за місткістю: малі здатні вміщувати 150-300 чоловік, середні – 300-600 чол. та великі – понад 600 чол. [105].

Сховища мають забезпечувати надійний захист людей від усіх уражаючих факторів та їх безперервне перебування в них не менше двох діб. Вони розташовуються на місцевості, що знаходиться на відстані від ліній водостоку та каналізації, щоб уникнути їх затоплення. Всі входи та виходи мають бути з тим же ступенем захисту, що й основні приміщення, а на випадок їх завалу, повинні бути передбачені аварійні виходи. Висота основних приміщень укриття не менше 2,2 м, а рівень підлоги розташований вище ґрунтових води не менш як на 20 см [104].

Протирадіаційними укриття дозволяють захистити людей від опромінення та впливу радіоактивних речовин в зонах радіаційного забруднення, від біологічних засобів у краплинно-рідинному вигляді та світлового випромінювання ядерного вибуху [104]. Вони можуть створюватись у підвальних поверхах будинків та споруд.

При їх виборі та підготовці обов'язково враховують захисні властивості будівельних матеріалів та окремих конструкцій, оскільки певна щільність та товщина шару матеріалу здатні ослаблювати та зменшити потік радіоактивних речовин. Що дозволяє людині, що знаходиться в приміщенні

отримати меншу дозу опромінення. Так підвали в дерев'яних одноповерхових будинках здатні ослабити дозу радіації в 7 разів, а в житлових одноповерхових кам'яних (цегляних) будинках – у 40, у двоповерхових – у 100, середня частина підвалу кількоповерхового кам'яного будинку – у 800 – 1000 разів. У випадку коли рівень радіації не високий для захисту від бактеріальних засобів, парів отруйних і сильнодіючих ядучих речовин можна використовувати кам'яні (цегляні) або дерев'яні будівлі [104].

Після оцінки захисних властивостей наявних приміщень та опираючись на реальні потреби в кожному окремому випадку, розраховують та завчасно готують необхідну кількість укриттів, для захисту людей в умовах сильного радіаційного забруднення місцевості [104].

Під час підготовки обраних приміщень, в якості протирадіаційних укриттів важливо вжити заходів для виключення можливості потрапляння радіоактивного пилу в приміщення, а також посилити захист населення, збільшити обсяг робіт для пристосування укриттів, а також враховувати потреби і наявність необхідних матеріалів [104].

Сховища та протирадіаційні укриття складаються з основних та допоміжних приміщень. Основними є приміщення для розміщення людей, пункти управління, медичні пункти, а до допоміжних відносяться приміщення, де розташовані фільтровентиляційне обладнання, системами водопостачання, опалення, санітарні вузли, приміщення для зберігання продовольства, тамбур-шлюзи (тамбури) [104].

Найпростіші укриття для захисту або щілини є захисними спорудами, які можуть бути створеними населенням у найкоротший проміжок часу. Такі укриття здатні певною мірою захистити від ударної хвилі, радіаційного ураження та світлового випромінювання. Для будівництва щілин вибирають сухі місця, що знаходяться на підвищенні [104, 105].

Найпростіші укриття типу щілини бувають двох видів відкритими та перекритими. Відкриті щілини дозволяють зменшити в 1,2–2 рази ймовірність ураження ударною хвилею, проникаючою радіацією та світловим

випромінюванням. Перекриті щілини є більш надійним захистом населення, оскільки вони здатні повністю захистити від світлового випромінювання, від ударної хвилі – у 1,5–3 рази, від проникаючої радіації і радіоактивного випромінювання – у 200-300 разів, а також надійно захищає від осколкових і кулькових бомб, від запалювальних засобів [105].

Будівництво укриття починається з розмітки та визначення її плану на місцевості. Щілини роблять у вигляді прямолінійних ділянок з такими розмірами: довжина до 15 м, глибина 1,8–2,0 м; ширина: зверху – 1,1-1,3 м, а на дні – 0,8 м [105]. Стіни укриття укріплюють дошками, хмизом, жердинами або іншим підручним матеріалом. Перекриття щілини роблять з накату колод, за яким йде шар глини товщиною 10—15 см, що дозволяє захистити укриття від потрапляння дощових вод. Потім шар ґрунту 20—40 см, а зверху все вкривають дерном. Також такі укриття можна будувати зі збірних залізобетонних конструкцій. Входи до щілини роблять східчасті, під прямим кутом до осі укриття [104]. У перекритій щілині вхід може бути з однієї або двох сторін з дверима та тамбуром [105]. В нормі найпростіші укриття типу щілина повинна забезпечити захист від 10 до 15 чоловік [105].

Під час перебування у захисних спорудах населення повинно виконувати правила поведінки в них. Заходити у приміщення укриття потрібно спокійно, організовано, швидко та без паніки. Розміщуватись зручніше групами, наприклад ті хто разом працюють або живуть в одному будинку. У кожній групі призначають головного. Людей з дітьми розміщують в окремих приміщеннях або у спеціально відведених місцях. Хворих та людей похилого віку варто розмістити ближче до вентиляційних труб. Дозволяється приходити з власними засобами індивідуального захисту дихання, продуктами харчування та документами. До укриттів забороняється приносити громіздкі, легкозаймисті речі, не дозволяється шуміти та курити, виходити на зовні без дозволу. Всі у захисній споруді повинні виконувати розпорядження чергового в укритті [105].

ВИСНОВКИ

В ході розробки дипломної кваліфікаційної роботи магістра було проаналізовано використання додаткових відео матеріалів під час навчання. В результаті, чого визначено, що їх використання позитивно впливає на процес навчання. Однак через відсутність на платформі спеціальних алгоритмів для оцінювання якості таких відео є ризик натрапити на неякісний контент. Дана проблема збільшує час на пошук необхідного матеріалу та як результат зменшує його використання під час навчання. Таким чином в результаті було вирішено розробити систему, яка вирішує дану проблему.

Для розроблюваного програмного продукту було спроектовано: базу знань фільтрації, базу знань сортування відео, структуру бази даних, загальну архітектуру, прототип графічного інтерфейсу.

В якості моделі представлення знань в рекомендаційній системі обрано алгоритм hybrid, який поєднує в собі content-based filtering та collaborative filtering, що використовуються для фільтрування та сортування відео відповідно. Загальну архітектуру системи було розділено на дві основні частини: серверну та клієнтську, та створено для кожної з них окремий сервер.

Спроектовано базовий макет вебзастосунку, що дозволило наглядно візуалізувати структуру інтерфейсу, показати розташування основних елементів, а також визначити їх функцій.

Для зберігання всієї необхідної інформації було спроектовано структуру СКБД, в якості якої обрано популярну – MySQL. Оскільки вона здатна забезпечити продукт потужним функціоналом, надійним захистом даних, економічним використанням ресурсів, швидким вирішенням проблем.

В якості мови для серверної частини проєкту обрано популярну мову – Python. Оскільки вона має широку підтримку суспільства, легкий, лаконічний та зрозумілий синтаксис, є кросплатформенною, а також має велику кількість бібліотек, що дозволило використовувати для розробки вже існуючий функціонал. В якості мови програмування для клієнтської частини обрано

JavaScript, що забезпечило контроль роботи браузера, динамічне оновлення вмісту вебсторінки та зменшило навантаження як на сам сервер так і на його трафік. Також в якості допоміжної технології для розробки клієнтської частини було обрано бібліотеку Vue.js.

В якості тимчасового сховища, яке взаємодіє з клієнтською частиною, зберігає дані користувача після реєстрації та реалізує механізми взаємодії між авторизованими на неавторизованими користувачами, було обрано спеціальну бібліотеку-розширення для Vue.js під назвою Vuex.

В результаті порівняння вебсерверів було для клієнтської частини було обрано – Nginx. Оскільки він займає менше місця на диску є масштабованою та високопродуктивною реалізацією, у якої кожен робочий процес здатний одночасно обробляти тисячі HTTP – з'єднань.

Фреймворком для серверної частини обрано Django. Таке рішення обумовлено наявністю в нього таких можливостей як аутентифікація, маршрутизація URL-адрес, механізм шаблонів, об'єктно-реляційне відображення (ORM) та міграція схеми бази даних, що роблять його надзвичайно масштабованим, швидким та універсальним. Наявність готових рішень в фреймворку дозволило пришвидшити та спростити розробку.

В якості вебсервісу було обрано Gunicorn, оскільки в ньому наявна можливість «з коробки» налаштування максимального часу на очікування відповіді, визначання кількості запущених процесів водночас, збирати дані під час роботи, а саме кількість запитів, які статуси відповідей надходять, з яких IP адрес надходять запити тощо, які можна відстежувати в режимі реального часу, та у випадку критичних ситуацій на сервері легше знаходити рішення.

Для паралельного виконання задач в системі розроблено модуль асинхронних задач, що дозволило економити час на очікуванні для користувачів та максимально ефективно використовувати ресурси системи. Для реалізації модулю було розроблено власні скрипти для кожної з задач

В якості системи контейнеризації було використано Docker-compose, оскільки необхідно розгортати декілька контейнерів.

Таким чином в результаті розробки отримали клієнт-серверний програмний продукт. У якому посилання на відео беруться з YouTube завдяки спеціальному API, що вона надає та зберігаються в базі даних. Якщо посилання вже наявне в БД продукту, тоді дані про нього разом з усіма попередніми оцінками беруться з БД.

Наповнення бази даних відео на початковому етапі здійснюється за рахунок користувачів, експертів в сфері медицини (лікарі, академіки, викладачі), шляхом збереження посилань на відео у базі даних під час кожного пошуку. Щоб впевнитись у тому, що користувач є дійсно фахівцем його реєстрація в системі здійснюється за допомогою адміністратора платформи. Для цього користувач відправляє на пошту необхідні дані, після чого в результаті успішного проходження перевірки він отримає доступ до системи.

Авторизовані користувачі-експерти мають можливість оцінювати відео даного ресурсу, а саме визначати якість, наявність практичних сцен, достовірність матеріалу відео тощо. Усі ці оцінки зберігаються у базі даних та формують середню оцінку.

Наукова новизна отриманих результатів полягає у тому, що отримано подальший розвиток гібридного алгоритму надання рекомендацій для обробки списку відео з метою надати більш релевантні результати пошуку.

Практична значимість результатів полягає у тому, що розроблена рекомендаційна система, яка забезпечує додаткову фільтрацію та сортування списку відео, таким чином, користувач отримує більш релевантний та якісний список відео, оскільки враховуються наявні виставлені експертами оцінки відео взяті з бази даних. Завдяки наявній можливості оцінювати відео користувачами з'являється можливість в майбутньому задіяти нейромодуль, що на основі вже оціненого відео, буде перевіряти нове відео, що надходитиме з системи YouTube та фільтрувати його.

Результати даної магістерської роботи доповідалися на НПК Тиждень науки-2020 та на X Міжнародній НПК «Сучасні проблеми і досягнення в галузі радіотехніки, телекомунікацій та інформаційних технологій».

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

- 1 Самые популярные социальные сети мира (2019) и рейтинг Украины. Думали их 5? [Электрон. ресурс]. – Режим доступа: <https://futurenow.com.ua/ru/samye-populyarnye-sotsyalnye-sety-myra-2019-y-rejtyng-ukrayny-dumaly-yh-5/>.
- 2 The Best Online Colleges & Universities [Electronic source]. – Access mode: <https://www.learnhowtobecome.org/online-universities-and-colleges/best/>.
- 3 11 Лучших онлайн-университетов и школ [Электрон. ресурс]. – Режим доступа: <https://expertology.ru/11-luchshikh-onlayn-universitetov-i-shkol/#q-product-2>.
- 4 9 лучших альтернатив YouTube в 2019 году [Электрон. ресурс]. – Режим доступа: <https://alugha.com/article/9093b250-0ffe-11e9-b6e9-6dade42308d1?lang=rus>.
- 5 The top 500 sites on the web [Electronic source]. – Access mode: <https://www.alexa.com/topsites>.
- 6 YouTube или Vimeo — что лучше? [Электрон. ресурс]. – Режим доступа: <http://seoquicktop.blogspot.com/2018/04/youtube-vimeo.html>.
- 7 Y. Chtouki, H. Harroud, M. Khalidi and S. Bennani, "The impact of YouTube videos on the student's learning," 2012 International Conference on Information Technology Based Higher Education and Training (ITHET), Istanbul, 2012, pp. 1-4.
- 8 Канал влияния: как YouTube можно использовать для обучения [Электрон. ресурс]. – Режим доступа: <https://www.rbc.ru/trends/education/5e2af0c19a79478e600b3d99>.
- 9 YouTube: An emerging tool in anatomy education [Electronic source]. – Access mode: https://www.academia.edu/1468308/YouTube_An_emerging_tool_in_anatomy_education.

10 15 популярных Ютуб каналов по медицине [Электрон. ресурс]. – Режим доступа: <https://infoselection.ru/zdorove2/item/728-obrazovatelnye-i-nauchnye-kanaly-po-medsine>.

11 Covington P. Deep Neural Networks for YouTube Recommendations / P. Covington, J. Adams, E. Sargin // RecSys '16: Proceedings of the 10th ACM Conference on Recommender Systems / P. Covington, J. Adams, E. Sargin.. – С. 191–198.

12 Understanding YouTube's Algorithm in 2019 [Electronic source]. – Access mode: <http://social.colostate.edu/2019/04/02/youtube-algorithm/>.

13 YouTube Algorithm: The Constantly Updated Guide to YouTube's Updates & Changes [Electronic source]. – 2019. – Access mode: <https://blog.hubspot.com/marketing/youtube-algorithm>.

14 How the YouTube Algorithm Works in 2020 [Electronic source]. – Access mode: <https://www.authorsguilds.com/how-youtube-algorithm-works/>.

15 How Does the YouTube Algorithm Work: A Peek into YouTube's Algorithm Changes [Electronic source]. – 2019. – Access mode: <https://influencermarketinghub.com/how-does-the-youtube-algorithm-work/>.

16 How Does the YouTube Algorithm Work? A Guide to Getting More Views [Electronic source]. – 2019. – Access mode: <https://blog.hootsuite.com/how-the-youtube-algorithm-works/>.

17 Как работают новые алгоритмы YouTube? Две искусственные нейронные сети [Электрон. ресурс]. – 2019. – Режим доступа: <http://konoden.ru/kak-rabotaut-novye-algoritmy-youtube-dve-iskusstvennye-nejronnye-seti/>.

18 How YouTube Recommends Videos [Electronic source]. – 2018. – Access mode: <https://towardsdatascience.com/how-youtube-recommends-videos-b6e003a5ab2f>.

19 Термин: Контент [Электрон. ресурс]. – Режим доступа: <https://promopult.ru/library/Контент>.

20 Качественный контент: что это значит + как это сделать [Электрон. ресурс]. – Режим доступа: <https://in-scale.ru/blog/kachestvennyj-kontent>.

21 Что такое контент [[Электрон. ресурс]. – 2016. – Режим доступа: <https://seo-akademiya.com/seo-wiki/kontent/>.

22 Віральність контенту [Електрон. ресурс]. – Режим доступу: <https://alexsmokinof.lviv.ua/віральність-контенту/>.

23 Assessment of “YouTube” Content for Distal Radius Fracture Immobilization [Electronic source]. – Access mode: <https://www.sciencedirect.com/science/article/abs/pii/S1931720417300168>.

24 Хирурги рассказали, что смотрят видео на YouTube для самообразования. Однако есть проблема с проверкой контента [Электрон. ресурс]. – 2019. – Режим доступа: <https://m.habr.com/ru/news/t/478520/>.

25 Review on the Reliability of Medical Contents on YouTube [Electronic source]. – Access mode: <https://online-journals.org/index.php/i-joe/article/view/11558>.

26 Evaluation of the content, quality, reliability and accuracy of YouTube videos regarding endotracheal intubation techniques [Electronic source]. – Access mode: <http://www.njcponline.com/article.asp?issn=1119-3077;year=2018;volume=21;issue=12;spage=1651;epage=1655;aulast=ОСАК>.

27 YouTube documentation [Electronic source]. – Access mode: <https://developers.google.com/youtube/v3/docs>.

28 Рекомендательные системы [Электрон. ресурс]. – Режим доступа: <https://vas3k.ru/blog/355/>.

29 Baptiste R. Introduction to recommender systems [Electronic source] / Rocca Baptiste. – 2019. – Access mode: <https://towardsdatascience.com/introduction-to-recommender-systems-6c66cf15ada>.

30 Pandey P. The Remarkable world of Recommender Systems An overview of the Recommendation systems and how they provide an effective form of targeted marketing. [Electronic source] / Parul Pandey. – 2019. – Access mode:

<https://towardsdatascience.com/the-remarkable-world-of-recommender-systems-bff4b9cbe6a7>.

31 Deng H. Recommender Systems in Practice [Electronic source] / Houtao Deng. – 2019. – Access mode: <https://towardsdatascience.com/recommender-systems-in-practice-cef9033bb23a>.

32 Turban E. Information Technology for Management: On-Demand Strategies for Performance, Growth and Sustainability [Electronic source] / E. Turban, P. Carol, G. Wood – Access mode: <https://books.google.com.ua/books?id=hJZXDwAAQBAJ&lpg=PP1&hl=ru&pg=PA192#v=onepage&q&f=false>.

33 Поширені запитання про алгоритми для створення рекомендацій і монетизацію [Електрон. ресурс]. – Режим доступу: <https://support.google.com/youtube/answer/9269747?hl=uk>.

34 Рекомендательные системы сегодня – необходимость для бизнеса [Электрон. ресурс]. – Режим доступа: <https://blog.heyml.com/рекомендательные-системы-сегодня-необходимость-для-бизнеса-41c1c9109988>.

35 A simple way to explain the Recommendation Engine in AI [Electronic source]. – 2019. – Access mode: <https://medium.com/voice-tech-podcast/a-simple-way-to-explain-the-recommendation-engine-in-ai-d1a609f59d97>.

36 Introduction to Recommender Systems in 2019 Videos [Electronic source]. – 2019. – Access mode: <https://tryolabs.com/blog/introduction-to-recommender-systems/>.

37 Pandey P. Recommendation Systems in the Real world [Electronic source] / Parul Pandey. – 2019. – Access mode: <https://towardsdatascience.com/recommendation-systems-in-the-real-world-51e3948772f3>.

38 Представления знаний в интеллектуальных системах, экспертные системы [Электрон. ресурс]. – 2018. – Режим доступа: <https://habr.com/ru/post/346236/>.

39 Системы, основанные на знаниях [Электрон. ресурс]. – 2014. – Режим доступа: <https://all4study.ru/iskusstvennyj-intellekt-v-aviacii/sistemy-osnovannye-na-znaniyah.html>.

40 Client/Server Architecture [Electronic source]. – 2018. – Access mode: <https://www.techopedia.com/definition/438/clientserver-architecture>.

41 Client-Side vs Server-Side Learning Objectives [Electronic source] – Access mode: <https://www.cloudflare.com/learning/serverless/glossary/client-side-vs-server-side/>.

42 Introduction to the server side [Electronic source]. – 2019. – Access mode: https://developer.mozilla.org/en-US/docs/Learn/Server-side/First_steps/Introduction.

43 Прототип сайта: зачем он нужен и как создать его самостоятельно [Электрон. ресурс]. – 2019. – Режим доступа: <https://serpstat.com/ru/blog/prototip-sajta-zachem-on-nuzhen-i-kak-sozdat/>.

44 Как сделать прототип сайта [Электрон. ресурс]. – 2019. – Режим доступа: <https://blog.ringostat.com/ru/kak-sdelat-prototip-sayta/>.

45 What a Relational Database Is [Electronic source]. – Access mode: <https://www.oracle.com/database/what-is-a-relational-database/>.

46 Relational vs Non-Relational Databases: What's the Difference? [Electronic source]. – Access mode: <https://www.arkware.com/2019/04/20/relational-vs-non-relational-databases-whats-the-difference/>.

47 Non-relational data and NoSQL [Electronic source]. – Access mode: <https://docs.microsoft.com/en-us/azure/architecture/data-guide/big-data/non-relational-data>.

48 DB-Engines Ranking [Electronic source]. – Access mode: <https://db-engines.com/en/ranking>.

49 Method of calculating the scores of the DB-Engines Ranking [Electronic source]. – Access mode: https://db-engines.com/en/ranking_definition.

50 PYPL PopularitY of Programming Language [Electronic source] – Access mode: <http://pypl.github.io/PYPL.html>.

51 Python Vs PHP: What's the Difference? [Electronic source] – Access mode: <https://www.guru99.com/python-vs-php.html>.

52 PHP vs JavaScript: Must Know Differences [Electronic source] – Access mode: <https://www.guru99.com/php-vs-javascript.html>.

53 Stackoverflow [Electronic source] – Access mode: <https://stackoverflow.com/tags>.

54 What is JavaScript ? [Electronic source] – Access mode: https://developer.mozilla.org/uk/docs/Learn/JavaScript/First_steps/What_is_JavaScript.

55 Vue vs. Angular vs. React comparison [Electronic source] Access mode: <https://theappsolutions.com/blog/development/vue-angular-react-comparison/>.

56 Angular vs React vs Vue: Which Is the Best Choice for 2020? [Electronic source]. – 2019. – Access mode: <https://levelup.gitconnected.com/angular-vs-react-vs-vue-which-is-the-best-choice-for-2020-81f577697c7e>.

57 Angular vs Vue vs React: choosing the best framework in 2020 [Electronic source]. – 2020. – Access mode: https://www.educative.io/blog/react-angular-vue-comparison?aid=5082902844932096&utm_source=google&utm_medium=cpc&utm_campaign=blog-dynamic&gclid=Cj0KCQjw6ar4BRDnARIsAITGzlBWg58w1pLFfJKt59Q3CoBma7GOjl0EusFkY-XC85F45ILDPLUabGYaArXbEALw_wcB.

58 What is Vuex? [Electronic source] Access mode: <https://vuex.vuejs.org/>.

59 Apache vs nginx – сравнение и преимущества [Электрон. ресурс]. – 2020. – Режим доступа: <https://wiki.merionet.ru/servernye-resheniya/34/apache-vs-nginx-sravnenie-i-preimushhestva/>.

60 10 отличий Apache от Nginx [Электрон. ресурс]. – 2017. – Режим доступа: <https://ekaterinagoltsova.github.io/posts/apache-vs-nginx/>.

61 Top 13 Python Web Frameworks to Learn in 2020 [Electronic source] – Access mode: <https://steelkiwi.com/blog/top-10-python-web-frameworks-to-learn/>.

62 Лучшие фреймворки Python для вебразработки, которые подойдут начинающим [Электрон. ресурс]. – Режим доступа: <https://tproger.ru/articles/best-python-frameworks-for-beginners/#4>.

63 Топ-5 вебфреймворков Python в 2020 году [Электрон. ресурс]. – 2020. – Режим доступа: <https://pythonist.ru/top-5-python-veb-frejmvorki-v-2020-godu/>.

64 Flask или Django? Выбираем Python вебфреймворк [Электрон. ресурс]. – Режим доступа: <https://python-scripts.com/flask-or-django#flask-vs-django-differences>.

65 Flask vs Django in 2020: Which Framework to Choose? [Electronic source]. – 2020. – Access mode: <https://hackr.io/blog/flask-vs-django>.

66 Documentation Django [Electronic source] – Access mode: <https://docs.djangoproject.com/en/3.1/faq/general/#djangoappears-to-be-a-mvc-framework-but-you-call-the-controller-the-view-and-the-view-the-template-how-come-you-don-t-use-the-standard-names>.

67 Working Structure of Django MTV Architecture [Electronic source]. – 2019. – Access mode: <https://towardsdatascience.com/working-structure-of-django-mtv-architecture-a741c8c64082>.

68 PEP 3333 -- Python Web Server Gateway Interface v1.0.1 [Electronic source] – Access mode: <https://www.python.org/dev/peps/pep-3333/>.

69 A Performance Analysis of Python WSGI Servers: Part 2 [Electronic source]. – 2016. – Access mode: <https://www.appdynamics.com/blog/engineering/a-performance-analysis-of-python-wsgi-servers-part-2/>.

70 Celery 5.0.0 documentation [Electronic source] – Access mode: <https://docs.celeryproject.org/>.

71 Dramatiq documentation [Electronic source] – Access mode: <https://dramatiq.io/>.

72 Argo documentation [Electronic source] – Access mode: <https://argoproj.github.io/>.

73 Why is Celery useful? [Electronic source] – Access mode: <https://www.fullstackpython.com/celery.html>.

74 Что такое контейнер? Обзор технологии контейнеризации [Электрон. ресурс]. – Режим доступа: <https://www.cloud4y.ru/about/news/obzor-tekhnologii-konteynerizatsii/>.

75 Что такое контейнеризация приложений и как это работает [Электрон. ресурс]. – 2019. – Режим доступа: <https://www.bigdataschool.ru/blog/containers-big-data.html>.

76 What is a Container? [E Electronic source] – Access mode: <https://www.docker.com/resources/what-container>.

77 В чем разница между Docker Compose и Kubernetes? [Электрон. ресурс]. – Режим доступа: <https://qastack.ru/programming/47536536/whats-the-difference-between-docker-compose-and-kubernetes>.

78 Що таке Kubernetes? [Электрон. ресурс]. – Режим доступа: <https://kubernetes.io/uk/docs/concepts/overview/what-is-kubernetes/>.

79 Що таке ІДЕ? [Электрон. ресурс]. – Режим доступа: <https://www.redhat.com/en/topics/middleware/what-is-ide>.

80 Лучшие IDE и редакторы кода для Python [Электрон. ресурс]. – 2018. – Режим доступа: <https://tproger.ru/translations/python-ide/>.

81 Подборка бесплатных кроссплатформенных сред для вебразработки [Электрон. ресурс]. – Режим доступа: <https://tproger.ru/digest/free-webdev-editors/>.

82 Выбираем редактор кода: VS Code или Atom? [Электрон. ресурс]. – 2019. – Режим доступа: <https://techrocks.ru/2019/06/24/vs-code-vs-atom-how-to-choose-between-them/>.

83 Writing your first Django app, part 1¶ [Electronic source]. – Access mode: <https://docs.djangoproject.com/en/3.1/intro/tutorial01/#creating-the-polls-app>.

84 API і сервіси [Електронний ресурс] – Режим доступу до ресурсу: <https://console.developers.google.com/>.

85 Identify access scopes [Electronic source]. – Access mode: <https://developers.google.com/youtube/v3/guides/auth/installed-apps#identify-access-scopes>.

86 Writing custom django-admin commands [Electronic source]. – Access mode: <https://docs.djangoproject.com/en/3.1/howto/custom-management-commands/>.

87 Vue Router documentation [Electronic source]. – Access mode: <https://router.vuejs.org/>.

88 Getting Started - Installing Git [Electronic source]. – Access mode: <https://git-scm.com/book/en/v2/Getting-Started-Installing-Git>.

89 Hosting platform "Linode" [Electronic source]. – Access mode: <https://www.linode.com/>.

90 Основні правила дотримання охорони праці при роботі на персональних ЕОМ [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: <https://www.victorija.ua/dovidnik/osnovni-pravyla-dotrymannya-ohorony-pratsi-pri-roboti-na-personalnyh-eom.html>.

91 Безопасность жизнедеятельности [Электрон. ресурс]. – Режим доступа: <https://www.bestreferat.ru/referat-33994.html>.

92 Вредные и опасные факторы при работе с компьютером [Электрон. ресурс]. – Режим доступа: <http://www.grandars.ru/shkola/bezopasnost-zhiznedeyatelnosti/vrednye-factory-pri-rabote-na-pk.html>.

93 Опасные и вредные производственные факторы при работе с компьютером [Електрон. ресурс]. – Режим доступа: <https://studfile.net/preview/6794424/page:24/>.

94 Охорона праці офісних працівників: організація та обладнання робочого місця [Електронний ресурс] – Режим доступу до ресурсу: <http://oppb.com.ua/news/ohorona-praci-ofisnyh-pracivnykiv-organizaciya-ta-obladnannya-robochogo-miscya>.

95 Вимоги охорони праці при виконанні робіт на персональному комп'ютері [Електронний ресурс] – Режим доступу до ресурсу: <https://selidovorada.gov.ua/novini/vimogi-okhoroni-pratsi-pri-vikonanni-robit-na-personalnomu-komp-yuteri>.

96 Засоби індивідуального і колективного захисту від ураження електричним струмом [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: <https://www.sop.com.ua/article/96-zasobi-ndividualnogo-kolektivnogo-zahistu-vd-urajennya-elektrichnim-strumom>.

97 Робота в офісі: основні санітарно-гігієнічні вимоги [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: <https://te.dsp.gov.ua/robota-v-ofisi-osnovni-sanitarno-gigiyenichni-vymogy/>.

98 Условия труда при работе с компьютерной техникой [Електрон. ресурс]. – Режим доступа: <https://studfile.net/preview/6794424/page:25/>.

99 Небезпека пожежі в офісі: вимоги, правила, інструкції, відповідальність керівника [Електронний ресурс] – Режим доступу до ресурсу: <https://ohorona24.com.ua/uk/news/opasnost-pozhara-v-ofise-trebovaniya-pravila-instruktsii-otvetstvennost-rukovoditelya/>.

100 Пожежна безпека на підприємстві: правила та організація [Електронний ресурс] – Режим доступу до ресурсу: <https://profiteh.ua/pozhezhna-bezpeka-na-pidpriemstvi-pravyla-ta-orhanizatsiia/>.

101 Інструкція про заходи пожежної безпеки у офісі на підприємстві [Електронний ресурс] – Режим доступу до ресурсу: <http://trudova->

ohrana.ru/primery-dokumentov/zrazki-nstrukcj-z-pozhezhno-bezpeki/4839-nstrukcja-pro-zahodi-pozhezhno-bezpeki-u-ofs-na-pdprimstv-.html.

102 Пожежна безпека в офісі [Електронний ресурс] – Режим доступу до ресурсу: <http://nkc-ekspert.com.ua/uk/news/pozhezhna-bezpeka-v-ofisi>.

103 Оснащення офісних приміщень первинними засобами пожежогасіння [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: <http://oppb.com.ua/articles/osnashchennya-ofisnyh-prymishchen-pervynnymy-zasobamy-pozhezhogasinnya>.

104 Укриття населення в захисних спорудах [Електронний ресурс] – Режим доступу до ресурсу: https://pidru4niki.com/13731120/bzhd/ukrittya_naselennya_zahisnih_sporudah.

105 Укриття населення в захисних спорудах: типи, сховища, правила перебування. Реферат [Електронний ресурс] – Режим доступу до ресурсу: <https://ru.osvita.ua/vnz/reports/dpju/24297/>.

ДОДАТОК А
Текст програми

A1 docker-compose.yaml

```

version: "3.7"
services:
  mysql:
    image: mysql:8.0
    container_name: mysql_youmed
    volumes:
      - ./data:/var/lib/mysql
    environment:
      - MYSQL_ROOT_HOST=%
      - MYSQL_DATABASE=${DB_NAME}
      - MYSQL_USER=${DB_USER}
      - MYSQL_PASSWORD=${DB_PASSWORD}
      - MYSQL_ROOT_PASSWORD=${DB_PASSWORD}
    networks:
      - youmed
    ports:
      - "3306:3306"
    command:
      [
        "mysqld",
        "--character-set-server=utf8mb4",
        "--collation-server=utf8mb4_unicode_ci",
        "--default-authentication-plugin=mysql_native_password",
      ]
  fetchbasedata:
    container_name: fetchbasedata_youmed
    image: backend-base
    build:
      context: .
      dockerfile: Dockerfile
      target: backend-base
    environment:
      -
      YOUTUBE_CLIENT_SECRET=${YOUTUBE_CLIENT_SECRET}
      - APP_SECRET_KEY=${APP_SECRET_KEY}
      - DB_USER=${DB_USER}
      - DB_PASSWORD=${DB_PASSWORD}
      - DB_NAME=${DB_NAME}
      - DB_HOST=mysql
      - APP_HOST=${APP_HOST}
    networks:
      - youmed
    depends_on:
      - mysql
      - backend
    restart: on-failure
    entrypoint: python manage.py fetchbasedata
  fetchfulldata:
    container_name: fetchfulldata_youmed
    image: backend-base
    build:
      context: .
      dockerfile: Dockerfile
      target: backend-base
    environment:
      -
      YOUTUBE_CLIENT_SECRET=${YOUTUBE_CLIENT_SECRET}
      - APP_SECRET_KEY=${APP_SECRET_KEY}
      - DB_USER=${DB_USER}
      - DB_PASSWORD=${DB_PASSWORD}
      - DB_NAME=${DB_NAME}
      - DB_HOST=mysql
      - APP_HOST=${APP_HOST}
    networks:
      - youmed
    depends_on:
      - mysql
      - backend
    restart: on-failure
    entrypoint: python manage.py fetchfulldata
  filtervideos:
    container_name: filtervideos_youmed
    image: backend-base
    build:
      context: .
      dockerfile: Dockerfile
      target: backend-base
    environment:
      -
      YOUTUBE_CLIENT_SECRET=${YOUTUBE_CLIENT_SECRET}
      - APP_SECRET_KEY=${APP_SECRET_KEY}
      - DB_USER=${DB_USER}
      - DB_PASSWORD=${DB_PASSWORD}
      - DB_NAME=${DB_NAME}
      - DB_HOST=mysql
      - APP_HOST=${APP_HOST}
    networks:
      - youmed
    depends_on:
      - mysql
      - backend
    restart: on-failure
    entrypoint: python manage.py filtervideos
  backend:
    container_name: backend_youmed
    image: backend-gunicorn
    build:
      context: .
      dockerfile: Dockerfile
      target: backend-gunicorn
    environment:
      -
      YOUTUBE_CLIENT_SECRET=${YOUTUBE_CLIENT_SECRET}
      - APP_SECRET_KEY=${APP_SECRET_KEY}
      - DB_USER=${DB_USER}
      - DB_PASSWORD=${DB_PASSWORD}
      - DB_NAME=${DB_NAME}
      - DB_HOST=mysql
      - APP_HOST=${APP_HOST}
    networks:
      - youmed
    depends_on:
      - mysql
      - backend
    restart: on-failure
    entrypoint: python manage.py backend
  backend-base:
    container_name: backend-base_youmed
    image: backend-base
    build:
      context: .
      dockerfile: Dockerfile
      target: backend-base
    environment:
      -
      YOUTUBE_CLIENT_SECRET=${YOUTUBE_CLIENT_SECRET}
      - APP_SECRET_KEY=${APP_SECRET_KEY}
      - DB_USER=${DB_USER}
      - DB_PASSWORD=${DB_PASSWORD}
      - DB_NAME=${DB_NAME}
      - DB_HOST=mysql
      - APP_HOST=${APP_HOST}
    networks:
      - youmed
    depends_on:
      - mysql
      - backend
    restart: on-failure
    entrypoint: python manage.py filtervideos
    periodicdataupdate:
      container_name: periodicdataupdate_youmed
      image: backend-base
      build:
        context: .
        dockerfile: Dockerfile
        target: backend-base
        environment:
          -
          YOUTUBE_CLIENT_SECRET=${YOUTUBE_CLIENT_SECRET}
          - APP_SECRET_KEY=${APP_SECRET_KEY}
          - DB_USER=${DB_USER}
          - DB_PASSWORD=${DB_PASSWORD}
          - DB_NAME=${DB_NAME}
          - DB_HOST=mysql
          - APP_HOST=${APP_HOST}
        networks:
          - youmed
        depends_on:
          - mysql
          - backend
        restart: on-failure
        entrypoint: python manage.py periodicdataupdate
      prepareforfiltering:
        container_name: prepareforfiltering_youmed
        image: backend-base
        build:
          context: .
          dockerfile: Dockerfile
          target: backend-base
          environment:
            -
            YOUTUBE_CLIENT_SECRET=${YOUTUBE_CLIENT_SECRET}
            - APP_SECRET_KEY=${APP_SECRET_KEY}
            - DB_USER=${DB_USER}
            - DB_PASSWORD=${DB_PASSWORD}
            - DB_NAME=${DB_NAME}
            - DB_HOST=mysql
            - APP_HOST=${APP_HOST}
          networks:
            - youmed
          depends_on:
            - mysql
            - backend
          restart: on-failure
          entrypoint: python manage.py prepareforfiltering

```

```

-
YOUTUBE_CLIENT_SECRET=${YOUTUBE_CLIENT_SECRET}
- APP_SECRET_KEY=${APP_SECRET_KEY}
- DB_USER=${DB_USER}
- DB_PASSWORD=${DB_PASSWORD}
- DB_NAME=${DB_NAME}
- DB_HOST=mysql
- APP_HOST=${APP_HOST}
networks:
- youmed
depends_on:
- mysql
restart: on-failure
nginx:
container_name: nginx_youmed
image: prod-nginx
tty: true
build:

```

```

context: .
dockerfile: Dockerfile
target: prod-nginx
ports:
- "80:80"
networks:
- youmed
depends_on:
- backend
restart: on-failure

```

```

networks:
youmed:
driver: bridge

```

```

volumes:
mysql:

```

A2 Dockerfile

```

FROM python:3.8 AS backend-base
COPY ./backend/requirements.txt /tmp/requirements.txt
RUN pip3 install --upgrade pip && \
    pip3 install --no-cache-dir -r /tmp/requirements.txt
RUN mkdir -p /usr/src/app
WORKDIR /usr/src/app
COPY ./backend /usr/src/app
ENV PYTHONPATH=.
# TODO:::add wait-for

FROM backend-base AS backend-gunicorn
EXPOSE 8000
RUN pip install gunicorn==20.0.4
ENTRYPOINT python manage.py migrate && \
    gunicorn -b 0.0.0.0:8000 -w 4 --timeout 480 --max-requests 1000 --
log-level debug main.wsgi

```

```

FROM node:12.2.0-alpine AS frontend-build
WORKDIR /app
ENV PATH /app/node_modules/.bin:$PATH
COPY ./frontend/package.json /app/package.json
RUN npm install --silent
RUN npm install @vue/cli@3.7.0 -g
COPY ./frontend /app
RUN npm run build

FROM nginx:1.16.0-alpine AS prod-nginx
COPY --from=frontend-build /app/dist /usr/share/nginx/html
COPY --from=backend-gunicorn /usr/src/app/static
/usr/share/nginx/static
RUN rm /etc/nginx/conf.d/default.conf
COPY nginx/nginx.conf /etc/nginx/conf.d
EXPOSE 80
CMD ["nginx", "-g", "daemon off;"]

```

A3 nginx/nginx.conf

```

server {
    listen 80;
    server_name nginx;

    error_log /var/log/nginx/error.log;
    access_log /var/log/nginx/access.log;

    location /api/ {
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For
$proxy_add_x_forwarded_for;
        proxy_set_header X-NginX-Proxy true;
        proxy_set_header Host $http_host;
        proxy_pass http://backend:8000/api;
    }
    location /admin/ {

```

```

        proxy_pass http://backend:8000/admin/;
    }
    location / {
        root /usr/share/nginx/html;
        index index.html index.htm;
        try_files $uri $uri/ /index.html;
    }
    location /admin-static/ {
        autoindex on;
        alias /usr/share/nginx/static/;
    }
    error_page 500 502 503 504 /50x.html;

    location = /50x.html {
        root /usr/share/nginx/html;
    }
}

```

A4 backend/requirements.txt

```

django==3.1.2
django-rest-knox==4.1.0
django-cors-headers==3.5.0
requests==2.24.0
mysqlclient==2.0.1

```

```

google-auth==1.22.1
google-auth-oauthlib==0.4.1
google-api-python-client==1.12.3
aniso8601==8.0.0
django-extensions==3.0.9

```

A5 backend/main/settings.py

```

import os
from pathlib import Path

BASE_DIR = Path(__file__).resolve().parent.parent
SECRET_KEY = os.getenv('APP_SECRET_KEY')
DEBUG = True

```

```

ALLOWED_HOSTS = ['*']
APPEND_SLASH = True

```

```

# Application definition
INSTALLED_APPS = [

```

```

'django.contrib.admin',
'django.contrib.auth',
'django.contrib.contenttypes',
'django.contrib.sessions',
'django.contrib.messages',
'django.contrib.staticfiles',
'django_extensions',
'rest_framework',
'knox',
'corsheaders',
'filters',
'accounts',
'processes',
'videos',
'async_tasks'
]
MIDDLEWARE = [
'django.middleware.security.SecurityMiddleware',
'django.contrib.sessions.middleware.SessionMiddleware',
'corsheaders.middleware.CorsMiddleware',
'django.middleware.common.CommonMiddleware',
'django.middleware.csrf.CsrfViewMiddleware',
'django.contrib.auth.middleware.AuthenticationMiddleware',
'django.contrib.messages.middleware.MessageMiddleware',
'django.middleware.clickjacking.XFrameOptionsMiddleware',
]
ROOT_URLCONF = 'main.urls'

TEMPLATES = [
{
'BACKEND':
'django.template.backends.django.DjangoTemplates',
'DIRS': [],
'APP_DIRS': True,
'OPTIONS': {
'context_processors': [
'django.template.context_processors.debug',
'django.template.context_processors.request',
'django.contrib.auth.context_processors.auth',
'django.contrib.messages.context_processors.messages',
],
},
},
]

WSGI_APPLICATION = 'main.wsgi.application'
# Database
DATABASES = {
'default': {
'ENGINE': 'django.db.backends.mysql',
'NAME': os.getenv('DB_NAME'),
'USER': os.getenv('DB_USER'),
'PASSWORD': os.getenv('DB_PASSWORD'),
'HOST': os.getenv('DB_HOST'),
'PORT': 3306,
},
}
# Password validation
AUTH_PASSWORD_VALIDATORS = [
{
'NAME':
'django.contrib.auth.password_validation.UserAttributeSimilarityValid
ator',
},
{
'NAME':
'django.contrib.auth.password_validation.MinimumLengthValidator',
},
{
'NAME':
'django.contrib.auth.password_validation.CommonPasswordValidator'
,
},
{
'NAME':
'django.contrib.auth.password_validation.NumericPasswordValidator',
},

```

```

]
# Logging configuration
LOGGING = {
'version': 1,
'disable_existing_loggers': False,
'formatters': {
'verbose': {
'format': '[{levelname}] {asctime} :: {message}',
'style': '{',
},
},
'handlers': {
'file': {
'level': 'DEBUG',
'class': 'logging.FileHandler',
'filename': './logs/debug.log',
'formatter': 'verbose'
},
'console': {
'level': 'DEBUG',
'class': 'logging.StreamHandler',
'formatter': 'verbose'
},
},
'root': {
'handlers': ['console'],
'level': 'DEBUG',
},
'loggers': {
'django': {
'handlers': ['console'],
'level': 'INFO',
'propagate': False,
},
},
}
# Internationalization
LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'UTC'

USE_I18N = True
USE_L10N = True
USE_TZ = True

# Static files (CSS, JavaScript, Images)
STATIC_URL = '/admin-static/'
STATIC_ROOT = 'static'

# DRF
REST_FRAMEWORK = {
'DEFAULT_AUTHENTICATION_CLASSES':
('knox.auth.TokenAuthentication',),
'EXCEPTION_HANDLER': 'helpers.drf_custom_exception_handler'
}

APP_HOST = os.getenv('APP_HOST')

FRONTEND_SUCCESS_REDIRECT =
f'{APP_HOST}/successYouTubeAuth'
FRONTEND_FAILED_REDIRECT =
f'{APP_HOST}/failedYouTubeAuth'
YOUTUBE_AUTH_REDIRECT =
f'{APP_HOST}/api/auth/youtubeAuth/'

# CORS
CORS_ORIGIN_ALLOW_ALL = False
CORS_ALLOWED_ORIGINS = (os.getenv('APP_HOST'),)

# YouTube
YOUTUBE_SCOPES = [
'https://www.googleapis.com/auth/youtube.readonly',
]
YOUTUBE_SECRET = {
'web': {

```

```

    # "client_id": '203276610183-
ekd2kk6mv1jbbnq4160hs4erbu9ouj6g.apps.googleusercontent.com',
    # youmed2
    "client_id": "920610067512-
b3pn3hufg4s2utel50132rh2nr2sph04.apps.googleusercontent.com",
    # youmed
    "project_id": "youmed-290421",
    "auth_uri": "https://accounts.google.com/o/oauth2/auth",
    "token_uri": "https://oauth2.googleapis.com/token",
    "auth_provider_x509_cert_url":
"https://www.googleapis.com/oauth2/v1/certs",
    "client_secret": os.getenv("YOUTUBE_CLIENT_SECRET"),
    "redirect_uris": [
        f"{APP_HOST}/api/auth/youtubeAuth/",
    ]
}
} import os
from pathlib import Path

BASE_DIR = Path(__file__).resolve().parent.parent

SECRET_KEY = os.getenv('APP_SECRET_KEY')
DEBUG = True

ALLOWED_HOSTS = ["*"]
APPEND_SLASH = True

# Application definition
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',

    'django_extensions',

    'rest_framework',
    'knox',
    'corsheaders',

    'filters',
    'accounts',
    'processes',
    'videos',
    'async_tasks'
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'corsheaders.middleware.CorsMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'main.urls'

TEMPLATES = [
    {
        'BACKEND':
'django.template.backends.django.DjangoTemplates',
        'DIRS': [],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]

```

```

]

WSGI_APPLICATION = 'main.wsgi.application'

# Database
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': os.getenv('DB_NAME'),
        'USER': os.getenv('DB_USER'),
        'PASSWORD': os.getenv('DB_PASSWORD'),
        'HOST': os.getenv('DB_HOST'),
        'PORT': 3306,
    },
}

# Password validation
AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME':
'django.contrib.auth.password_validation.UserAttributeSimilarityValid
ator',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.CommonPasswordValidator'
,
    },
    {
        'NAME':
'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]

# Logging configuration
LOGGING = {
    'version': 1,
    'disable_existing_loggers': False,
    'formatters': {
        'verbose': {
            'format': '[{levelname}] {asctime} :: {message}',
            'style': '{',
        }
    },
    'handlers': {
        'file': {
            'level': 'DEBUG',
            'class': 'logging.FileHandler',
            'filename': './logs/debug.log',
            'formatter': 'verbose'
        },
        'console': {
            'level': 'DEBUG',
            'class': 'logging.StreamHandler',
            'formatter': 'verbose'
        },
    },
    'root': {
        'handlers': ['console'],
        'level': 'DEBUG',
    },
    'loggers': {
        'django': {
            'handlers': ['console'],
            'level': 'INFO',
            'propagate': False,
        },
    },
}

# Internationalization
LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'UTC'

```

```

USE_I18N = True

USE_L10N = True

USE_TZ = True

# Static files (CSS, JavaScript, Images)
STATIC_URL = '/admin-static/'
STATIC_ROOT = 'static'

# DRF
REST_FRAMEWORK = {
    'DEFAULT_AUTHENTICATION_CLASSES':
        ('knox.auth.TokenAuthentication',),
    'EXCEPTION_HANDLER': 'helpers.drf_custom_exception_handler'
}

APP_HOST = os.getenv('APP_HOST')

FRONTEND_SUCCESS_REDIRECT =
f'{APP_HOST}/successYouTubeAuth'
FRONTEND_FAILED_REDIRECT =
f'{APP_HOST}/failedYouTubeAuth'
YOUTUBE_AUTH_REDIRECT =
f'{APP_HOST}/api/auth/youtubeAuth/'

# CORS

```

A6 backend/helpers/custom_encoders.py

```

def encode_str(data: str) -> str:
    return data.encode('utf-8').decode('iso-8859-1')
def decode_str(data: str) -> str:
    try:
        return data.encode('iso-8859-1').decode('utf-8')
    except (UnicodeEncodeError, UnicodeDecodeError,
AttributeError):
        return data
def decode_dict(data: dict) -> dict:
    keys_to_decode = [

```

A7 backend/helpers/user_auth_validation.py

```

import pytz
from typing import Union
from google.auth.transport.requests import Request
from google.oauth2.credentials import Credentials

def is_user_youtube_auth_valid(credentials_from_db: Union[dict,
Credentials]) -> Union[Credentials, bool]:
    if isinstance(credentials_from_db, dict):
        # TODO:::add saving after token refreshing
        if not credentials_from_db['token']:
            return False

        if 'id' in credentials_from_db:
            del credentials_from_db['id']
        if 'user' in credentials_from_db:
            del credentials_from_db['user']

```

A8 backend/accounts/models.py

```

from django.db import models
from django.contrib.auth.models import User

class YoutubeCredentials(models.Model):
    token = models.CharField(max_length=256, null=True)
    refresh_token = models.CharField(max_length=256, null=True)
    token_uri = models.CharField(max_length=256, null=True)
    client_id = models.CharField(max_length=256, null=True)

```

```

CORS_ORIGIN_ALLOW_ALL = False
CORS_ALLOWED_ORIGINS = (os.getenv('APP_HOST'),)

# YouTube
YOUTUBE_SCOPES = [
    'https://www.googleapis.com/auth/youtube.readonly',
]
YOUTUBE_SECRET = {
    "web": {
        # "client_id": '203276610183-
ekd2kk6mv1jbbnq4160hs4erbu9ouj6g.apps.googleusercontent.com',
        # youmed2
        "client_id": "920610067512-
b3pn3hufg4s2utel50132rh2nr2sph04.apps.googleusercontent.com",
        # youmed
        "project_id": "youmed-290421",
        "auth_uri": "https://accounts.google.com/o/oauth2/auth",
        "token_uri": "https://oauth2.googleapis.com/token",
        "auth_provider_x509_cert_url":
            "https://www.googleapis.com/oauth2/v1/certs",
        "client_secret": os.getenv("YOUTUBE_CLIENT_SECRET"),
        "redirect_uris": [
            f'{APP_HOST}/api/auth/youtubeAuth/',
        ]
    }
}

```

```

'title', 'description', 'name', 'source',
'tag', 'category'
]
for key, value in data.items():
    if key in keys_to_decode:
        try:
            data[key] = decode_str(value)
        except (UnicodeEncodeError, UnicodeDecodeError):
            continue
return data

```

```

credentials_from_db['expiry'] =
credentials_from_db['expiry'].astimezone(
    pytz.UTC).replace(tzinfo=None)

credentials = Credentials(**credentials_from_db)
else:
    credentials = credentials_from_db

if credentials.valid:
    return credentials

if credentials.expired and credentials.refresh_token:
    credentials.refresh(Request())
if credentials.valid:
    return credentials

return False

```

```

client_secret = models.CharField(max_length=256, null=True)
expiry = models.DateTimeField(null=True)

user = models.OneToOneField(
    to=User, on_delete=models.CASCADE
)

```

A9 backend/accounts/serializers.py

```
from rest_framework import serializers
from django.contrib.auth.models import User
from django.contrib.auth import authenticate
from django.contrib.auth.hashers import make_password

class LoginSerializer(serializers.Serializer):
    username = serializers.CharField(max_length=64)
    password = serializers.CharField(max_length=64)

    def validate(self, attrs):
        user = authenticate(**attrs)
        if user and user.is_active:
            return user
        raise serializers.ValidationError("Incorrect credentials!")

class RegisterSerializer(serializers.ModelSerializer):

    class Meta:
        model = User
        fields = (
            'id', 'username', 'email', 'password'
        )
        extra_kwargs = {
            'password': {
                'write_only': True
            },
            'username': {
                'required': True
            },
            'email': {
                'required': True
            },
        }
```

A10 backend/accounts/urls.py

```
from django.urls import path, include
from rest_framework.routers import DefaultRouter

from accounts.views import RegisterAPIView, LoginAPIView,
GetUpdateUserAPIViewSet, YoutubeAuthAPIView

urls_router = DefaultRouter()
urls_router.register(
    r'', GetUpdateUserAPIViewSet, basename='user'
)
```

A11 backend/accounts/views.py

```
import json
import pytz
from typing import Union
from urllib.parse import urlparse, parse_qs

import aniso8601
from django.conf import settings
from django.contrib.auth.signals import user_logged_out
from django.contrib.auth.models import User
from django.forms.models import model_to_dict
from django.shortcuts import redirect
from django.urls import reverse
from google_auth_oauthlib.flow import InstalledAppFlow
from oauthlib.oauth2.rfc6749.errors import InvalidGrantError
from knox import views
from knox.models import AuthToken
from rest_framework import generics, permissions, viewsets, mixins,
serializers
from rest_framework import status
from rest_framework.response import Response

from accounts.serializers import UserSerializer, LoginSerializer,
RegisterSerializer
from accounts.models import YoutubeCredentials
from videos.models import Featured
from helpers import is_user_youtube_auth_valid
```

```
def create(self, validated_data):
    user = User.objects.create_user(
        validated_data['username'], validated_data['email'],
        validated_data['password'])

    return user
```

```
class UserSerializer(serializers.ModelSerializer):
    class Meta:
        model = User
        fields = ('id', 'username', 'email', 'password')
        extra_kwargs = {
            'password': {
                'write_only': True,
                'required': False
            },
            'username': {
                'required': False
            },
            'id': {
                'required': False
            },
            'email': {
                'required': False
            },
        }

    def validate(self, attrs):
        if 'password' in attrs:
            attrs['password'] = make_password(attrs['password'])
        return super(UserSerializer, self).validate(attrs)
```

```
urlpatterns = [
    # path('register/', RegisterAPIView.as_view()),
    path('login/', LoginAPIView.as_view()),
    path('auth', include('knox.urls')),
    path('user/', include(urls_router.urls)),
    path('youtubeAuth/', YoutubeAuthAPIView.as_view(),
        name='confirm-youtube-auth')
]
```

```
class GetUpdateUserAPIViewSet(mixins.RetrieveModelMixin,
                              mixins.UpdateModelMixin,
                              viewsets.GenericViewSet):
    permission_classes = [
        permissions.IsAuthenticated,
    ]
    serializer_class = UserSerializer
    queryset = User.objects.all()

    def retrieve(self, request, *args, **kwargs):
        response = super(GetUpdateUserAPIView,
self).retrieve(request, *args, **kwargs)

        featured_list = [
            x.video_id for x in
Featured.objects.filter(user=request.user.id).all()
        ]

        return Response({
            'data': {
                'user': response.data,
                'links': featured_list
            },
            status=status.HTTP_200_OK
        })

    def update(self, request, *args, **kwargs):
        response = super().update(request, *args, **kwargs)
```

```

        return Response({
            'data': {
                'user': response.data,
            }
        }, status=status.HTTP_200_OK)

class YoutubeAuthAPIView(generics.GenericAPIView):
    """ This class is for using for storing the youtube auth keys """
    permission_classes = [
        permissions.IsAuthenticatedOrReadOnly
    ]

    def _parse_credentials(self, credentials: str) -> dict:
        credentials = json.loads(credentials)
        credentials['expiry'] = aniso8601.parse_datetime(
            credentials['expiry']
        ).astimezone(
            pytz.UTC
        ).replace(
            tzinfo=None
        )
        del credentials['scopes']

        return credentials

    def get(self, request):
        flow = InstalledAppFlow.from_client_config(
            settings.YOUTUBE_SECRET,
            scopes=settings.YOUTUBE_SCOPES
        )
        flow.redirect_uri = settings.YOUTUBE_AUTH_REDIRECT
        authorization_response =
        request.build_absolute_uri().replace('http', 'https')

        try:
            flow.fetch_token(authorization_response=authorization_response)
            credentials = flow.credentials.to_json()
            credentials = self._parse_credentials(credentials)
        except InvalidGrantError:
            credentials = None

        if credentials:
            YoutubeCredentials.objects.filter(
                client_id=credentials['client_id']
            ).update(**credentials)
            return redirect(settings.FRONTEND_SUCCESS_REDIRECT)

        return redirect(settings.FRONTEND_FAILED_REDIRECT)

    def youtube_link_or_none(request, user) -> Union[str, None]:
        try:
            credentials_from_db =
            YoutubeCredentials.objects.get(user_id=user.id)
        except YoutubeCredentials.DoesNotExist:
            credentials_from_db = None

        if credentials_from_db is not None and \
            is_user_youtube_auth_valid(model_to_dict(credentials_from_db)):
            return None

        flow = InstalledAppFlow.from_client_config(
            settings.YOUTUBE_SECRET,
            scopes=settings.YOUTUBE_SCOPES
        )

```

A12 backend/videos/models.py

```

from django.db import models
from django.contrib.auth.models import User
from filters.models import Tag, Category, Source

class Channel(models.Model):
    youtube_id = models.CharField(max_length=128)
    name = models.CharField(max_length=256)
    country = models.CharField(max_length=4)

```

```

flow.redirect_uri = settings.YOUTUBE_AUTH_REDIRECT
youtube_link = flow.authorization_url()
client_id = parse_qs(urlparse(youtube_link[0]).query)['client_id'][0]

```

```

if credentials_from_db is not None:
    credentials_from_db.client_id = client_id
    credentials_from_db.save()
else:
    YoutubeCredentials(
        user=user,
        client_id=client_id
    ).save()

return youtube_link[0]

```

```

class LoginAPIView(generics.GenericAPIView):
    serializer_class = LoginSerializer
    permission_classes = [
        permissions.AllowAny
    ]

    def post(self, request):
        serializer = self.get_serializer(data=request.data)
        serializer.is_valid(raise_exception=True)
        user = serializer.validated_data

        featured_list = [
            x.video_id for x in Featured.objects.filter(user=user.id).all()
        ]

        return Response({
            "data": {
                "user": UserSerializer(user,
                    context=self.get_serializer_context()).data,
                "token": f'{AuthToken.objects.create(user)[1]}',
                "links": featured_list,
                "youtube_link": youtube_link_or_none(request, user)
            }
        })

```

```

class RegisterAPIView(generics.GenericAPIView):
    serializer_class = RegisterSerializer
    permission_classes = [
        permissions.AllowAny
    ]

    def post(self, request):
        serializer = self.get_serializer(data=request.data)
        serializer.is_valid(raise_exception=True)
        user = serializer.save()

        return Response({
            "data": {
                "user": UserSerializer(user,
                    context=self.get_serializer_context()).data,
                "token": f'{AuthToken.objects.create(user)[1]}',
                "links": [],
                "youtube_link": youtube_link_or_none(request, user)
            }
        })

```

```

description = models.TextField()
keywords = models.TextField()

```

```

class ImagePreview(models.Model):
    link = models.CharField(max_length=256)
    width = models.IntegerField(default=0)
    height = models.IntegerField(default=0)

```

```

class YoutubeData(models.Model):
    etag = models.CharField(max_length=128)
    pub_date = models.DateTimeField()
    video_hash = models.CharField(max_length=64, unique=True)

    positive_mark_number = models.IntegerField(default=0)
    negative_mark_number = models.IntegerField(default=0)
    view_count = models.IntegerField(default=0)
    comment_count = models.IntegerField(default=0)

    image_preview = models.OneToOneField(
        to=ImagePreview, on_delete=models.CASCADE
    )

class Video(models.Model):
    class VideoStatus(models.IntegerChoices):
        # TODO:::add in progress status
        NOT_CHECKED = 0
        CHECKED = 1

    status = models.IntegerField(
        choices=VideoStatus.choices,
        default=VideoStatus.NOT_CHECKED
    )

    title = models.CharField(max_length=256)
    description = models.TextField()

    channel = models.ForeignKey(
        to=Channel, on_delete=models.CASCADE
    )
    youtube_data = models.OneToOneField(
        to=YoutubeData, on_delete=models.CASCADE
    )
    category = models.ForeignKey(
        to=Category, on_delete=models.CASCADE, null=True
    )

    updated_at = models.DateTimeField(db_index=True,
auto_now=True)
    created_at = models.DateTimeField(db_index=True,
auto_now_add=True)

    # def __str__(self):
    #     return self.title

# TODO:::move to accounts
class Featured(models.Model):
    class Meta:
        unique_together = ['video', 'user']
    video = models.ForeignKey(
        to=Video, on_delete=models.CASCADE
    )
    user = models.ForeignKey(
        to=User, on_delete=models.CASCADE
    )

```

A13 backend/videos/views.py

```

from django.db.utils import IntegrityError
from rest_framework import generics, status, permissions, mixins,
viewsets
from rest_framework.response import Response
from videos.models import Video, Featured, UserMark
from videos.serializers import UserMarkSerializer, VideoSerializer,
SmallVideoSerializer

def _get_quality_value(field_key, user_marks):
    curr_len = len(
        [x[field_key] for x in user_marks if x[field_key] > 0]
    ) or 1
    return round(
        sum([x[field_key] for x in user_marks]) / curr_len,
        2
    )

```

```

    updated_at = models.DateTimeField(db_index=True,
auto_now=True)
    created_at = models.DateTimeField(db_index=True,
auto_now_add=True)

    # def __str__(self):
    #     return f'{self.user.id} -> {self.video.title}'

# TODO:::move to accounts
class UserMark(models.Model):
    class Meta:
        unique_together = ['video', 'user']
    video = models.ForeignKey(
        to=Video, on_delete=models.CASCADE,
related_name='um_videos'
    )
    user = models.ForeignKey(
        to=User, on_delete=models.CASCADE,
related_name='um_user'
    )
    information_quality = models.FloatField(default=0, null=True)
    medical_practice_quality = models.FloatField(default=0, null=True)
    description_quality = models.FloatField(default=0, null=True)
    practical_usage_availability = models.FloatField(default=0,
null=True)

    # def __str__(self):
    #     return f'{self.user.id} -> {self.video.title}'

class TagVideo(models.Model):
    class Meta:
        unique_together = ['video', 'tag']

    tag = models.ForeignKey(
        to=Tag, on_delete=models.CASCADE, related_name='tv_tags'
    )
    video = models.ForeignKey(
        to=Video, on_delete=models.CASCADE,
related_name='tv_videos'
    )

class ChannelSource(models.Model):
    class Meta:
        unique_together = ['source', 'channel']

    source = models.ForeignKey(
        to=Source, on_delete=models.CASCADE,
related_name='cs_sources'
    )
    channel = models.ForeignKey(
        to=Channel, on_delete=models.CASCADE,
related_name='cs_channel'
    )

```

```

class VideoAPIViewSet(mixins.RetrieveModelMixin,
viewsets.GenericViewSet):
    serializer_class = VideoSerializer
    permission_classes = [
        permissions.IsAuthenticated
    ]

    def get_queryset(self):
        return Video.objects.select_related('youtube_data').all()

    def retrieve(self, request, *args, **kwargs):
        video = self.get_object()
        serialized_data = self.get_serializer(video).data

        # TODO:::compute middle sum value in the DB
        user_marks =
UserMark.objects.filter(video_id=video.id).values().all()

```

```

try:
    # TODO:::reduce number of requests to the DB
    current_mark = UserMark.objects.get(user_id=request.user.id,
video_id=video.id)
    current_mark = UserMarkSerializer(current_mark).data
except UserMark.DoesNotExist:
    current_mark = {}

return Response({
    'data': {
        'video': serialized_data,
        'current_mark': current_mark,
        'user_marks': {
            'information_quality':
_get_quality_value('information_quality', user_marks),
            'medical_practice_quality':
_get_quality_value('medical_practice_quality', user_marks),
            'description_quality':
_get_quality_value('description_quality', user_marks),
            'practical_usage_availability':
_get_quality_value('practical_usage_availability', user_marks),
        },
        'youtube_marks': {
            'positive_mark_number':
serialized_data['youtube_data']['positive_mark_number'],
            'negative_mark_number':
serialized_data['youtube_data']['negative_mark_number'],
            'view_count':
serialized_data['youtube_data']['view_count'],
            'comment_count':
serialized_data['youtube_data']['comment_count'],
        },
    }
})

class UserMarkAPIView(generics.GenericAPIView):
    permission_classes = [
        permissions.IsAuthenticated
    ]

    def post(self, request, video_id):
        user_mark_data = request.data
        user_mark_data['video_id'] = video_id
        user_mark_data['user_id'] = request.user.id

        if 'id' in user_mark_data and not isinstance(user_mark_data['id'],
int):
            del user_mark_data['id']

        if user_mark_data.get('id'):
            UserMark.objects.filter(id=user_mark_data['id']).update(
                **user_mark_data
            )
        else:
            try:
                UserMark.objects.create(
                    **user_mark_data
                )
            except IntegrityError:
                UserMark.objects.filter(
                    user_id=user_mark_data['user_id'],
                    video_id=user_mark_data['video_id']
                ).update(
                    **user_mark_data
                )

        current_mark = UserMark.objects.get(
            user_id=user_mark_data['user_id'],
            video_id=user_mark_data['video_id']

```

```

)
    user_marks =
UserMark.objects.filter(video_id=video_id).values().all()

return Response({
    'data': {
        'updated_mark': UserMarkSerializer(current_mark).data,
        'user_marks': {
            'information_quality':
_get_quality_value('information_quality', user_marks),
            'medical_practice_quality':
_get_quality_value('medical_practice_quality', user_marks),
            'description_quality':
_get_quality_value('description_quality', user_marks),
            'practical_usage_availability':
_get_quality_value('practical_usage_availability', user_marks),
        },
    }, status=status.HTTP_202_ACCEPTED)

class FeaturedListAPIView(generics.GenericAPIView):
    permission_classes = [
        permissions.IsAuthenticated
    ]

    def get(self, request, *args, **kwargs):
        videos = Video.objects.filter(
            id__in=Featured.objects.filter(
                user_id=request.user.id,
            ).values('video_id')
        ).all()

        return Response({
            'data': {
                'videos': SmallVideoSerializer(videos, many=True).data
            }
        })

class FeaturedAPIView(generics.GenericAPIView):
    permission_classes = [
        permissions.IsAuthenticated
    ]

    def post(self, request, video_id, *args, **kwargs):
        created, _ = Featured.objects.get_or_create(
            video_id=video_id,
            user_id=request.user.id
        )

        return Response({
            'data': {
                'featured': {
                    'id': created.id,
                    'video_id': video_id,
                    'user_id': request.user.id
                }
            }, status.HTTP_201_CREATED)

    def delete(self, request, video_id, *args, **kwargs):
        Featured.objects.filter(
            video_id=video_id,
            user_id=request.user.id
        ).delete()

        return Response({
            'data': {}
        }, status=status.HTTP_204_NO_CONTENT)

```

A14 backend/async_tasks/management/commands/filtervideos.py

```

import time
from collections import defaultdict
from django.core.management.base import BaseCommand
from django.db.models import F

```

```

from django.db import transaction
from processes.models import Process, ProcessVideo
from async_tasks.recommendations import get_recommendations

```

```

PACK_SIZE = 1 # Pack size for fetching processes

class Command(BaseCommand):
    help = 'Command for filtering videos per process'

    def _fetch_data_from_db(self):
        process_ids = Process.objects.filter(
            status=Process.ProcessStatus.WAITING_FOR_FILTERING_DATA
        ).values(
            'id'
        ).all()[0:PACK_SIZE]
        process_ids = [p['id'] for p in process_ids]
        # TODO:::add fetching featured list
        featured_links = []

        data_for_fetching = dict(
            video_title=F('video__title'),
            video_description=F('video__description'),

            video_source=F('video__channel__cs_channel__source_name'),
            video_tag=F('video__tv_videos__tag_name'),
            video_category=F('video__category__name'),

            youtube_positive_mark_number=F('video__youtube_data__positive__mark_number'),

            youtube_negative_mark_number=F('video__youtube_data__negative__mark_number'),
            youtube_view_count=F('video__youtube_data__view_count'),

            youtube_comment_count=F('video__youtube_data__comment_count')
        ),

        search_data=F('process__search_data'),

        process_source=F('process__pc_processes__category_name'),
        process_tag=F('process__ps_processes__source_name'),
        process_category=F('process__pt_processes__tag_name'),

        mark_information_quality=F('video__um_videos__information_quality'),
        mark_medical_practice_quality=F('video__um_videos__medical_practice_quality'),

        mark_description_quality=F('video__um_videos__description_quality'),

```

A15 backend/async_tasks/recommendations.py

```

from collections import defaultdict
from helpers.custom_encoders import decode_str

# TODO:::move to class-based objects
BASE_DATA_STRUCTURE = {
    'videos': defaultdict(lambda: {
        'title': str(),
        'description': str(),

        'sources': set(),
        'tags': set(),
        'categories': set(),

        'marks': {
            'information_quality': float(),
            'medical_practice_quality': float(),
            'description_quality': float(),
            'practical_usage_availability': float()
        },

        "youtube_marks": {
            "positive_mark_number": int(),
            "negative_mark_number": int(),
            "view_count": int(),
            "comment_count": int(),
        }
    }),
}

```

```

mark_practical_usage_availability=F('video__um_videos__practical__usage_availability'),
mark_user_id=F('video__um_videos__user__id')
)
# TODO:::add fetching with parts via the limit / offset
# TODO:::use group by
processes = ProcessVideo.objects.annotate(
    **data_for_fetching
).values(
    'process_id', 'video_id', *data_for_fetching.keys()
).filter(
    process_id__in=process_ids
).all()

return processes, featured_links

def _update_data(self, sorted_data: dict) -> None:
    with transaction.atomic():
        for process_id, ordered_videos in sorted_data.items():
            for video_order, video_id in ordered_videos.items():
                ProcessVideo.objects.filter(
                    process_id=process_id,
                    video_id=video_id
                ).update(
                    video_order=video_order
                )

                Process.objects.filter(
                    id=process_id
                ).update(
                    status=Process.ProcessStatus.SUCCESS
                )

def _handle(self):
    processes, featured_links = self._fetch_data_from_db()

    result_data = get_recommendations(processes, featured_links)

    self._update_data(result_data)

def handle(self, *args, **options):
    while True:
        self._handle()
        time.sleep(1)

'sources': set(),
'tags': set(),
'categories': set(),
'search_data': str()
}

def get_recommendations(unformatted_process_data, featured_links):
    formatted_data = _format_data(unformatted_process_data)
    filtered_data = _filter_data(formatted_data)
    sorted_data = _sort_data(filtered_data, featured_links)

    return sorted_data

def _format_data(processes):
    process_video_grouped = defaultdict(lambda:
        BASE_DATA_STRUCTURE

    information_quality_set = defaultdict(set)
    medical_practice_quality_set = defaultdict(set)
    description_quality_set = defaultdict(set)
    practical_usage_availability_set = defaultdict(set)

    for process_data in processes:
        process_id = process_data['process_id']
        process_video_grouped[process_id]['search_data'] =
            process_data['search_data']

        process_video_grouped[process_id]['sources'].add(

```

```

        decode_str(process_data['process_source']))
    process_video_grouped[process_id]['tags'].add(
        decode_str(process_data['process_tag']))
    process_video_grouped[process_id]['categories'].add(
        decode_str(process_data['process_category']))

    video_id = process_data['video_id']

    process_video_grouped[process_id]['videos'][video_id]['title'] = \
        decode_str(process_data['video_title'])

    process_video_grouped[process_id]['videos'][video_id]['description'] = \
        decode_str(process_data['video_description'])

    process_video_grouped[process_id]['videos'][video_id]['sources'].add(
        decode_str(process_data['video_source']))

    process_video_grouped[process_id]['videos'][video_id]['tags'].add(
        decode_str(process_data['video_tag']))

    process_video_grouped[process_id]['videos'][video_id]['categories'].add(
        decode_str(process_data['video_category']))

    process_video_grouped[process_id]['videos'][video_id]['youtube_marks'] = {
        "positive_mark_number":
        process_data['youtube_positive_mark_number'],
        "negative_mark_number": -1 *
        process_data['youtube_negative_mark_number'],
        "view_count": process_data['youtube_view_count'],
        "comment_count": process_data['youtube_comment_count']
    }

    if process_data['mark_information_quality'] is not None and \
        process_data['mark_information_quality'] > 0 and \
        process_data['mark_user_id'] not in
    information_quality_set[video_id]:

    process_video_grouped[process_id]['videos'][video_id]['marks']['information_quality'] = \

    (process_video_grouped[process_id]['videos'][video_id]['marks']['information_quality'] *
        len(information_quality_set[video_id]) +
        process_data['mark_information_quality']
        ) / float(len(information_quality_set[video_id]) + 1)

    information_quality_set[video_id].add(process_data['mark_user_id'])

    if process_data['mark_medical_practice_quality'] is not None and \
        process_data['mark_medical_practice_quality'] and \
        process_data['mark_user_id'] not in
    medical_practice_quality_set[video_id]:

    process_video_grouped[process_id]['videos'][video_id]['marks']['medical_practice_quality'] = \

    (process_video_grouped[process_id]['videos'][video_id]['marks']['medical_practice_quality'] *
        len(medical_practice_quality_set[video_id]) +
        process_data['mark_medical_practice_quality']
        ) / float(len(medical_practice_quality_set[video_id]) + 1)

    medical_practice_quality_set[video_id].add(process_data['mark_user_id'])

    if process_data['mark_description_quality'] is not None and \
        process_data['mark_description_quality'] > 0 and \
        process_data['mark_user_id'] not in description_quality_set:

```

```

    process_video_grouped[process_id]['videos'][video_id]['marks']['description_quality'] = \

    (process_video_grouped[process_id]['videos'][video_id]['marks']['description_quality'] *
        len(description_quality_set[video_id]) +
        process_data['mark_description_quality']
        ) / float(len(description_quality_set[video_id]) + 1)

    description_quality_set[video_id].add(process_data['mark_user_id'])

    if process_data['mark_practical_usage_availability'] is not None
    and \
        process_data['mark_practical_usage_availability'] > 0 and \
        process_data['mark_user_id'] not in
    practical_usage_availability_set:

    process_video_grouped[process_id]['videos'][video_id]['marks']['practical_usage_availability'] = \

    (process_video_grouped[process_id]['videos'][video_id]['marks']['practical_usage_availability'] *
        len(practical_usage_availability_set[video_id]) +
        process_data['practical_usage_availability']
        ) / float(len(practical_usage_availability_set[video_id]) + 1)

    practical_usage_availability_set[video_id].add(process_data['mark_user_id'])

    return process_video_grouped

def compare_criteria_lists(process_criteria: set, video_criteria: set) ->
bool:
    """ Dummy search through the two lists because total number of
    elements
    will not be bigger then 100.
    """
    process_criteria = list(process_criteria)
    video_criteria = list(video_criteria)

    if process_criteria[0] is None or video_criteria[0] is None:
        # For cases when no criteria were chosen
        return True

    for pc in process_criteria:
        if pc is None:
            continue

        pc = pc.lower()
        for vc in video_criteria:
            if vc is None:
                continue

            vc = vc.lower()
            if pc.find(vc) >= 0 or vc.find(pc) >= 0:
                return True

    return False

def filter_data(formated_data: BASE_DATA_STRUCTURE) -> dict:
    """
    Criteria are:
    * sources
    * tags
    * categories
    """
    videos_for_filtering = defaultdict(set)

    for process_id, process_data in formated_data.items():
        for video_id, video_data in process_data['videos'].items():
            if not compare_criteria_lists(process_data['sources'],
            video_data['sources']) and \

```

```

        not_compare_criteria_lists(process_data['tags'],
video_data['tags']) and \
        not_compare_criteria_lists(process_data['categories'],
video_data['categories']):
    videos_for_filtering[process_id].add(video_id)
    continue

for process_id, video_ids in videos_for_filtering.items():
    for video_id in video_ids:
        del formatted_data[process_id]['videos'][video_id]

return formatted_data
def _sort_data(filtered_data: BASE_DATA_STRUCTURE,
featured_links: list) -> dict:
    """
    Criteria are:
    * description availability
    * is in featured
    * user marks
    * information quality
    * medical practice quality
    * description quality
    * practical usage availability
    * youtube marks
    * positive marks count
    * negative marks count
    * number of views

```

A16 frontend/router/routes.js

```

import Home from "@pages/Home";
import Featured from "@pages/Featured";
import VideoInfo from "@pages/VideoInfo";
import NotFound from "@pages/NotFound";
import FailedYouTubeAuth from "@pages/FailedYouTubeAuth";
import SuccessYouTubeAuth from "@pages/SuccessYouTubeAuth";

export default [
{
    path: "/",
    name: "home",
    component: Home
},
{
    path: "/featured",
    name: "featured",
    component: Featured
},
},
];

```

A17 frontend/store/actions/auth.js

```

const { backendHost } = require("../../config");
import axios from "axios";

const urlEntrypoint = backendHost;

const defaultHeaders = {
    "Content-Type": "application/json"
};

export default {
    getUser({ dispatch, commit, state }, token) {
        let headers = {
            Authorization: `Token ${token}`,
            ...defaultHeaders
        };

        return axios({
            method: "get",
            url: `${urlEntrypoint}/auth/user/`,
            headers: headers
        });
    },
    updateUser({ dispatch, commit, state }, payload) {
        const token = state.user.token;
        const user_id = state.user.id;

```

```

        * number of comments
        """
        process_video_order = defaultdict(dict)

        for process_id, process_data in filtered_data.items():
            videos = [
                {
                    'video_id': v_id,
                    'marks_sum': sum(
                        list(v_data['marks'].values()) +
                        list(v_data['youtube_marks'].values())
                    ) + (1 if v_data['description'] else 0) + int(v_id in
featured_links)
                } for v_id, v_data in process_data['videos'].items()
            ]
            videos = sorted(
                videos,
                key=lambda video: video['marks_sum'],
                reverse=True
            )

            process_video_order[process_id] = {
                (index + 1): video['video_id'] for index, video in
enumerate(videos)
            }

        return process_video_order

```

```

{
    path: "/videos/:id",
    name: "videoInfo",
    component: VideoInfo
},
{
    path: "/successYouTubeAuth",
    component: SuccessYouTubeAuth
},
{
    path: "/failedYouTubeAuth",
    component: FailedYouTubeAuth
},
{
    path: "*",
    component: NotFound
}
];

```

```

let headers = {
    Authorization: `Token ${token}`,
    ...defaultHeaders
};

return axios({
    method: "put",
    url: `${urlEntrypoint}/auth/user/${user_id}/`,
    headers: headers,
    data: payload
});

registerAction({ dispatch, commit, state }, payload) {
    commit("setLoadingStatus", true);
    return axios({
        method: "post",
        url: `${urlEntrypoint}/auth/register/`,
        data: payload,
        headers: defaultHeaders
    }).then(res => {
        dispatch("confirmYouTubeAuth", res.data.data.youtube_link);
        return res;
    });
},
loginAction({ dispatch, commit, state }, payload) {
    console.log(process.env.VUE_APP_BACKEND_HOST);

```

```

commit("setLoadingStatus", true);
return axios({
  method: "post",
  url: `${urlEntrypoint}/auth/login/`,
  data: payload,
  headers: defaultHeaders
}).then(res => {
  dispatch("confirmYoutubeAuth", res.data.data.youtube_link);
  return res;
});
},
logoutAction({ dispatch, commit, state }) {
  commit("setLoadingStatus", true);

  const token = state.user.token;
  let headers = {
    Authorization: `Token ${token}`,
    ...defaultHeaders
  };

```

```

return;

return axios({
  method: "post",
  url: `${urlEntrypoint}/auth/logout/`,
  data: {},
  headers: headers
});
},
confirmYoutubeAuth({ dispatch, commit, state }, youtube_link) {
  if (youtube_link) {
    window.open(youtube_link, "_blank", "width=400, height=450");
  }
  return;
}
};

```

A18 frontend/pages/Home.vue

```

<template>
<div>
  <v-container>
    {{{}}}
    <big-search v-on:startSearch="startSearch" ref="bigSearch" />
    <ul>
      <li class="red--text subheading" v-for="err in errors">
        {{{err}}}
      </li>
    </ul>
    <v-container v-show="process.status != undefined">
      <v-layout flex class="pt-0">
        <div class="headline pt-2">
          Results:
        </div>
        <v-spacer />
        <v-btn
          icon color="primary"
          :disabled="process.prev_process == undefined"
          @click="getOtherProcess(process.prev_process)"
        >
          <v-icon>mdi-arrow-left</v-icon>
        </v-btn>
        <v-btn
          icon color="primary"
          :disabled="process.next_process == undefined"
          @click="getOtherProcess(process.next_process)"
        >
          <v-icon>mdi-arrow-right</v-icon>
        </v-btn>
      </v-layout>
      <v-divider color="grey" class="mb-2"/>
      <v-row
        class="fill-height"
        align-content="center"
        justify="center"
        v-show="!successStatuses.includes(process.status)"
      >
        <v-col
          class="headline text-center"
          cols="12"
        >
          {{{processStatusesLabels[process.status]}}}
        </v-col>
        <v-col cols="6">
          <v-progress-linear
            color="deep-purple accent-4"
            indeterminate
            rounded
            height="6"
          ></v-progress-linear>
        </v-col>
      </v-row>

```

```

    <div v-show="successStatuses.includes(process.status)">
      <v-layout row wrap v-if="this.videos.length > 0">
        <search-item v-for="video in this.videos"
          :key="video.video_id"
          :video="video"/>
      </v-layout>
      <v-layout row wrap justify-center v-else class="headline pt-2">
        No media were found!
      </v-layout>
    </div>
  </v-container>
</div>
</template>

<script>
import BigSearch from '@components/search/BigSearch';
import SearchItem from '@components/search/SearchItem';

export default {
  data: () => {
    return {
      videos: [],
      errors: [],

      pollingInterval: 2000, // 2 seconds

      process: {},
      processStatusesLabels: {
        0: "Waiting for fetching base data...",
        1: "Waiting for fetching full data...",
        2: "Waiting for filtering...",
        3: "Success!",
        4: "Failed!"
      },
      successStatuses: [3, 4],
    }
  },
  methods: {
    setProcessFromAPI(process_id) {
      this.$store.dispatch('getProcess', process_id)
        .then(response => {
          this.process = response.data.data.process;
          this.$store.commit('setProcess', this.process);

          this.videos = response.data.data.videos;

          if (!this.process.active) {
            this.$store.dispatch('setProcessActive', process_id)
              .catch(err => {
                this.process = {}
                this.errors = err.response.data.errors;
              });

```

```

    }

    if (!this.successStatuses.includes(this.process.status))
  {
    this.executePolling();
  } else if (this.videos.length == 0 &&
this.process.next_process != undefined) {
    // this.getOtherProcess(this.process.next_process);
  }
})
.catch(err => {
  this.process = {}
  this.errors = err.response.data.errors;
});
},
getOtherProcess(process_id) {
  this.setProcessFromAPI(process_id);
},
executePolling() {
  setTimeout(() => {
    this.setProcessFromAPI(this.process.id)
  }, this.pollingInterval);
},
startSearch(payload){
  if (this.$store.state.user.token) {
    this.$root.$children[0].$refs.errorDialog.show("You must
be authorized to perform this action!");
    return
  }

  this.$root.$children[0].$refs.bigProcess.show();

  this.$store.dispatch('startProcess', payload)
  .then((response) => {
    this.$root.$children[0].$refs.bigProcess.close();
    this.process = response.data.data.process;
    this.executePolling();
  })
}

```

```

  })
  .catch((err) => {
    this.$root.$children[0].$refs.bigProcess.close();

    this.errors = err.response.data.errors;
  })
},
components: {
  'big-search':    BigSearch,
  'search-item':  SearchItem,
},
mounted() {
  if (this.$store.state.user.token &&
this.$store.state.process.id){
    this.process = this.$store.state.process;

    if (!this.successStatuses.includes(this.process.status)) {
      this.executePolling();
    } else if (this.videos.length == 0) {
      this.getOtherProcess(this.process.id);
    } else if (this.videos.length == 0 &&
this.process.next_process != undefined) {
      this.getOtherProcess(this.process.next_process);
    }
  }
},
computed:
{
  ttt(){
    console.log(process.env.VUE_APP_BACKEND_HOST);
    return process.env.VUE_APP_BACKEND_HOST
  }
}
}
</script>

```

A19 frontend/main.js

```

import Vue from "vue";
import Vuetify from "vuetify";
import VueYouTubeEmbed from "vue-youtube-embed";
import App from "@App";
import store from "@store";
import router from "@router";
import "vuetify/dist/vuetify.min.css";
import "./styles/main.scss";

Vue.use(Vuetify);
Vue.use(VueYouTubeEmbed);

```

A20 frontend/App.vue

```

<template>
  <v-app>
    <main-header/>
    <v-main>
      <router-view />
    </v-main>
    <big-process ref="bigProcess" />
    <login-form ref="loginForm" />
    <register-form ref="registerForm" />
    <user-info ref="userInfo" />
    <error-dialog ref="errorDialog" />
    <success-dialog ref="successDialog" />

    <main-footer/>
  </v-app>
</template>

<script>
import Header from '@components/main/Header';
import Footer from '@components/main/Footer';
import LoginForm from '@components/user/LoginForm';

```

```

const opts = {};

const vuetify = new Vuetify(opts);

new Vue({
  el: "#app",
  render: h => h(App),
  router,
  store,
  vuetify });

import RegisterForm from "@components/user/RegisterForm";
import UserInfo from "@components/user/UserInfo";
import BigProcess from "@components/progress/BigProcess";
import ErrorDialog from '@components/dialogs/ErrorDialog';
import SuccessDialog from '@components/dialogs/SuccessDialog';

export default {

  name: "app",
  components: {
    'main-footer':    Footer,
    'main-header':    Header,
    'login-form':     LoginForm,
    'register-form':   RegisterForm,
    'user-info':      UserInfo,
    'big-process':    BigProcess,
    'error-dialog':   ErrorDialog,
    'success-dialog': SuccessDialog
  },
}
</script>

```

ДОДАТОК Б
Слайди презентації

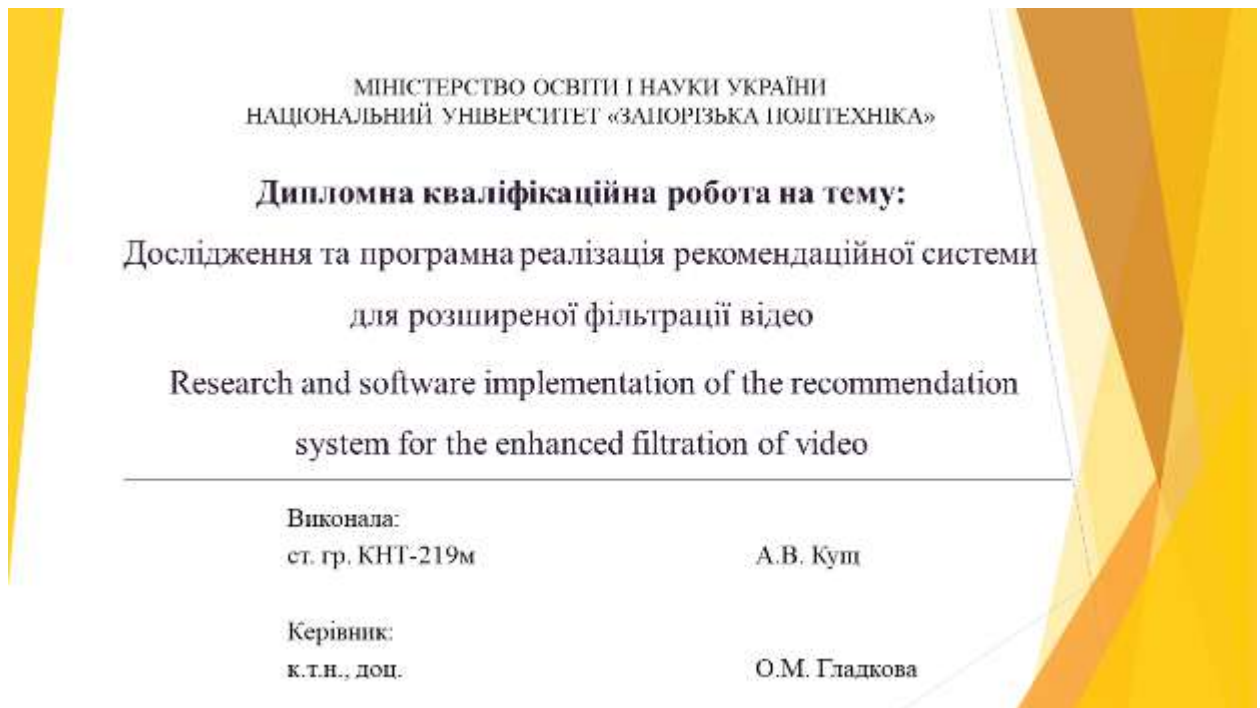


Рисунок Б.1 – Титульний слайд

Мета та завдання

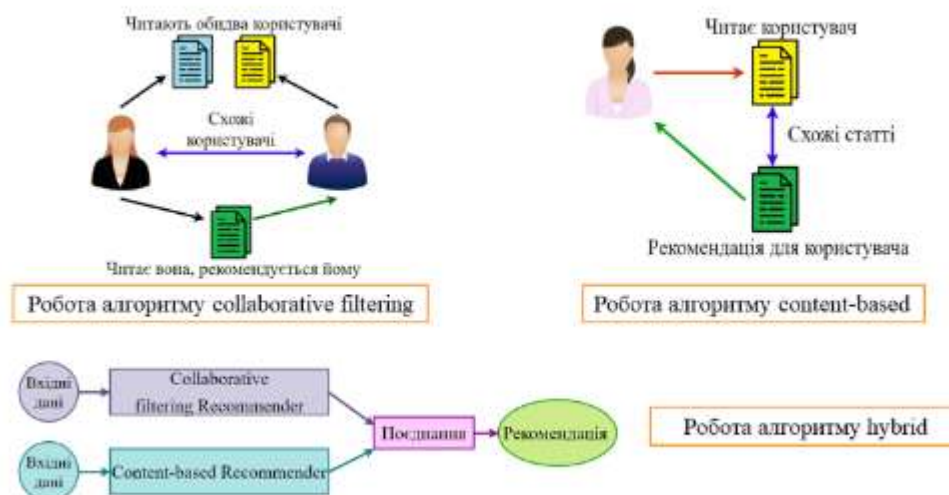
Мета роботи – розробити рекомендаційну систему, яка дозволить експертам в медичній сфері швидко знаходити якісний відео контент та забезпечить їх механізмами для його оцінювання.

Завдання:

- ▶ обрати модель представлення знань для рекомендаційної системи;
- ▶ спроектувати базу знань;
- ▶ спроектувати архітектуру БД;
- ▶ проаналізувати та обрати СКБД;
- ▶ розробити загальну архітектуру системи;
- ▶ розробити архітектуру та структуру користувацького інтерфейсу;
- ▶ обрати засоби контейнеризації.

Рисунок Б.2 – Мета та завдання роботи

Рекомендаційні системи: вибір алгоритму



3

Рисунок Б.3 – Рекомендаційні системи: вибір алгоритму

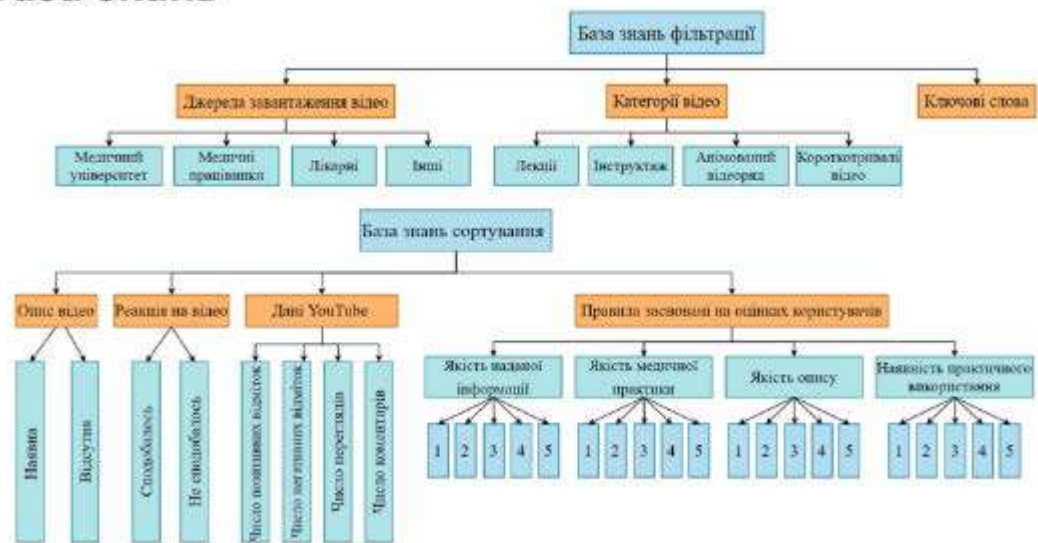
Параметри для оцінки

- ▶ Параметри відео ролику :
 - ▶ джерела, що завантажили відео (медичний університет, лікарні, медичні працівники, інші);
 - ▶ категорії відео (лекції, інструктаж, анімований відеоряд, короткотривалі відео, тощо);
 - ▶ ключові слова;
 - ▶ наявність опису відео.
- ▶ Характеристики відео пов'язані з реакцією користувачів на переглянуте відео:
 - ▶ якість наданої інформації в відео,
 - ▶ якість медичної практики,
 - ▶ якість опису,
 - ▶ реакція користувачів, яким сподобалось чи не сподобалось відео.

4

Рисунок Б.4 – Параметри для оцінки

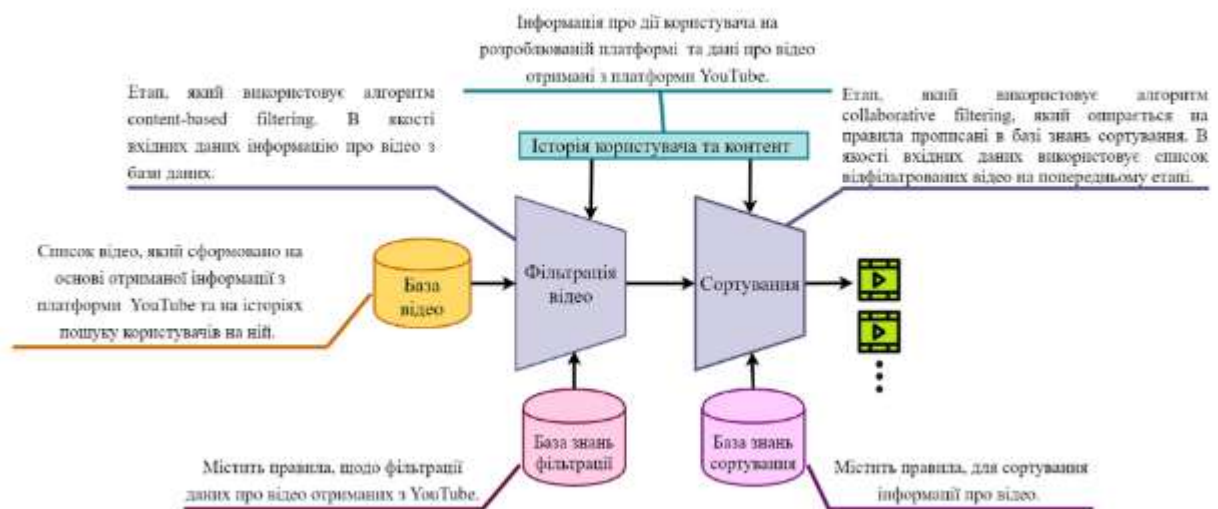
База знань



5

Рисунок Б.5 – База знань

Загальна архітектура рекомендаційної системи



6

Рисунок Б.6 – Загальна архітектура рекомендаційної системи

База даних

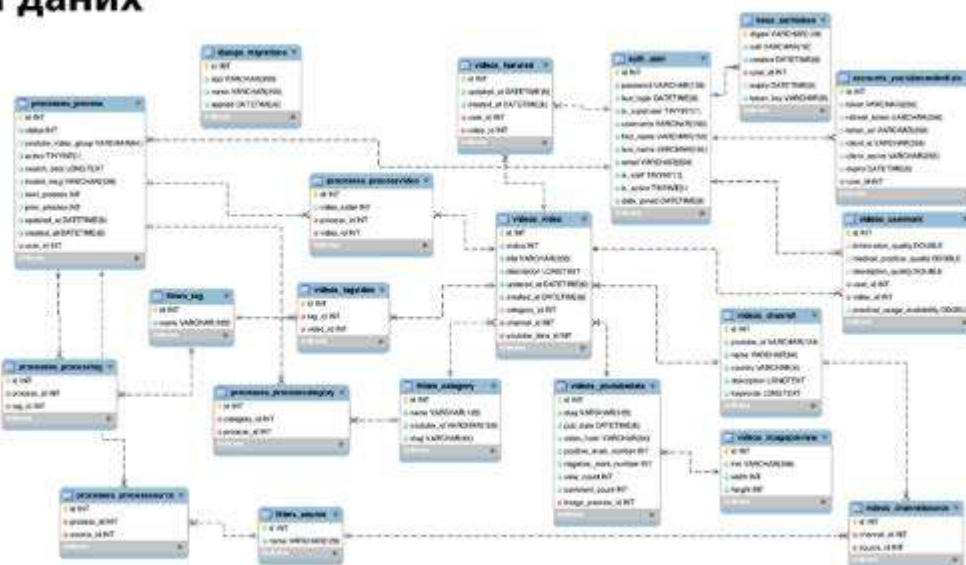
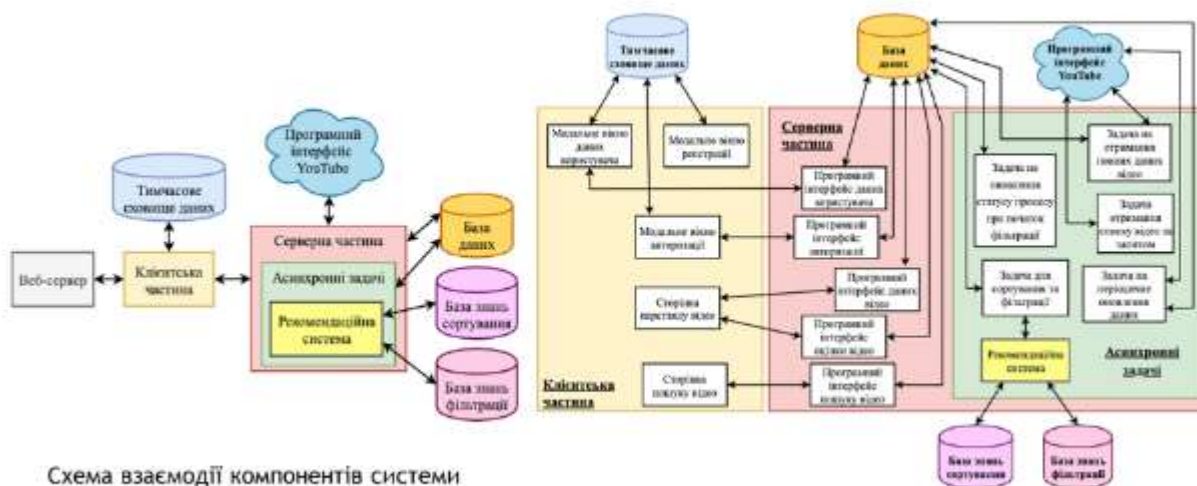


Рисунок Б.7 – База даних

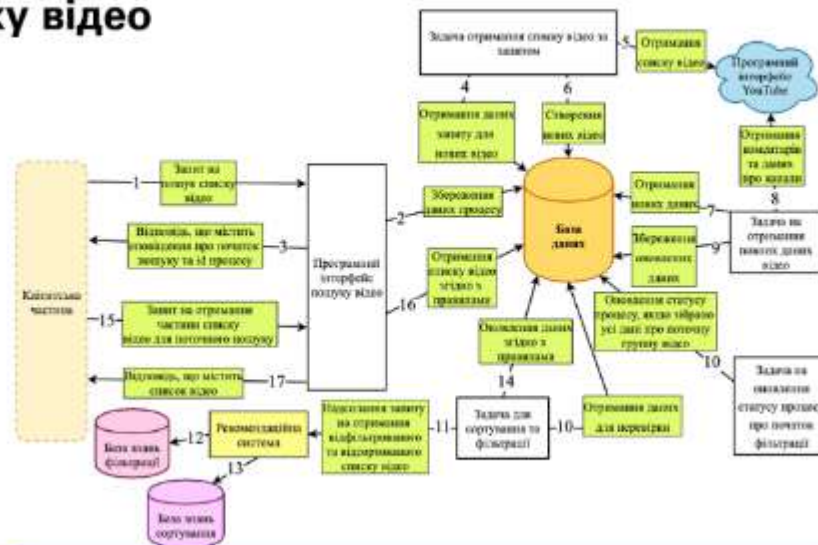
Взаємодія між частинами програмного продукту



Розширена схема взаємодії компонентів

Рисунок Б.8 – Взаємодія між частинами програмного продукту

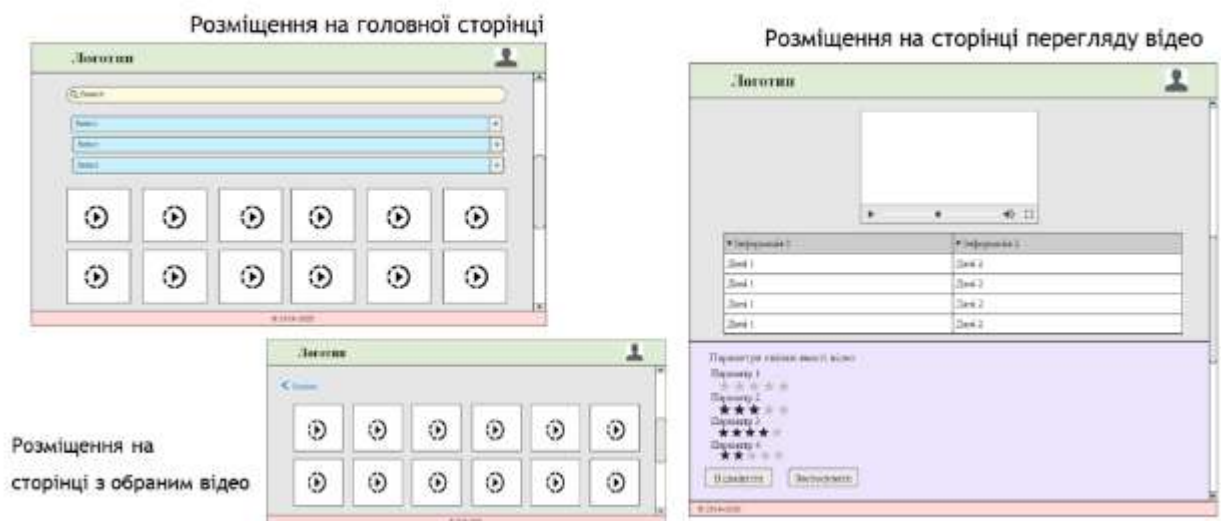
Схема взаємодії компонентів в запиті на отримання списку відео



9

Рисунок Б.9 – Схема взаємодії компонентів в запиті на отримання списку відео

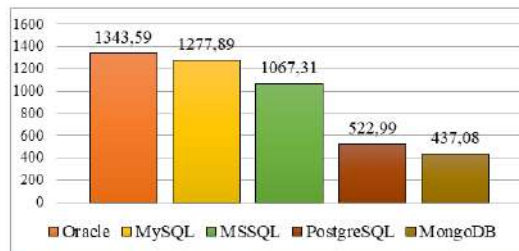
Розміщення компонентів на сторінках



10

Рисунок Б.10 – Розміщення компонентів на сторінках

Вибір СКБД



Рейтинг популярності СКБД

Порівняння СКБД

Параметр	Oracle	MySQL	Microsoft SQL	PostgreSQL	MongoDB
Тип БД	Реляційна	Реляційна	Реляційна	Реляційна	Нереляційна
Підтримка ОС	Linux, Microsoft Windows	Linux, Microsoft Windows, MacOS	Linux, Microsoft Windows	Linux, Microsoft Windows, MacOS	Linux, Microsoft Windows, MacOS
Використання ресурсів	Дуже високе	Економічне	Високе	Середнє	Середнє
Технічна підтримка	Платна	Безкоштовна	Платна	Безкоштовна	Безкоштовна
Підтримка суспільством	9	10	8	8	10
Підтримка виробником	9	8	9	7	8
Документація	8	8	7	7	10

11

Рисунок Б.11 – Вибір СКБД

Вибір мов програмування

Статистика частоти використання мов програмування



Порівняльна характеристика мов програмування

Параметр	Python	PHP	JavaScript
Об'єктно-орієнтована	Так	Так	Так
Виконується на стороні сервера	Так	Так	Так
Мітка списаріння на стороні клієнта	Ні	Ні	Так
Виконується в вікні браузера	Ні	Ні	Так
Підтримка БД	Так	Так	Так
Менеджер пакетів	PIP – система управління пакетами, що контролюється для установки і управління програмними пакетами.	Composer – це пакетний менеджер, який надає засоби управління залежностями в PHP-додатках.	NPM – менеджер пакетів для мови JavaScript.
Підтримка спільноти	Мак. велику спільноту підтримки, що росте.	Широка підтримка з боку спільноти.	Широка підтримка з боку спільноти.
Документація	Гарно прописана документація для всіх версій мови.	Мова PHP добре документована.	Добре документована, з прикладами.

12

Рисунок Б.12 – Вибір мов програмування

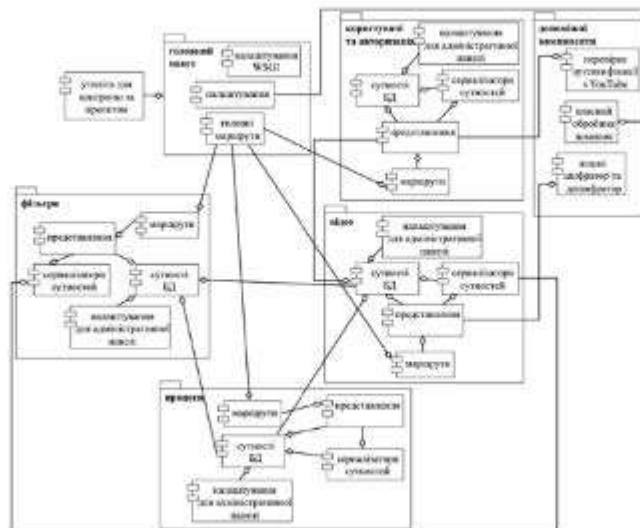
Додаткові інструменти для розробки

- ▶ Також обрані наступні технології для розробки
 - ▶ Фреймворк Django – для розробки серверної частини.
 - ▶ Бібліотеки Vue.JS та Vuex – допоміжні технології для розробки клієнтської частини.
 - ▶ Nginx – вебсервер для програмного продукту.
 - ▶ Gunicorn – вебсервіс для серверної частини.
 - ▶ Docker-compose – система контейнеризації.

13

Рисунок Б.13 – Додаткові інструменти для розробки

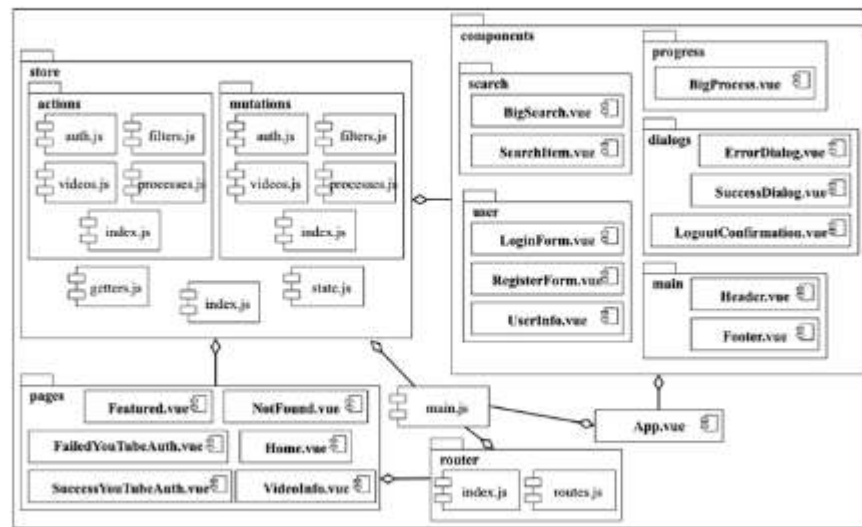
Загальна файлова структура серверної частини



14

Рисунок Б.14 – Загальна файлова структура серверної частини

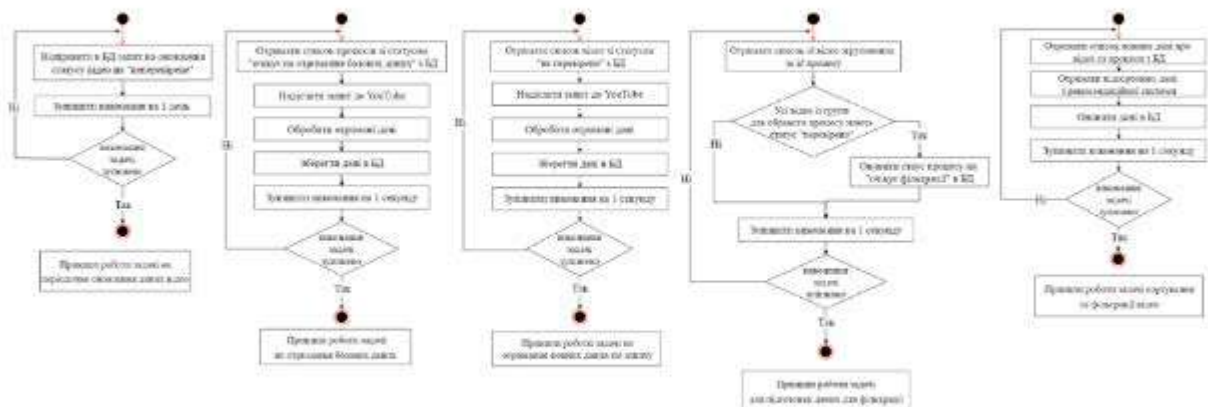
Файлова структура клієнтської частини



15

Рисунок Б.15 – Файлова структура клієнтської частини

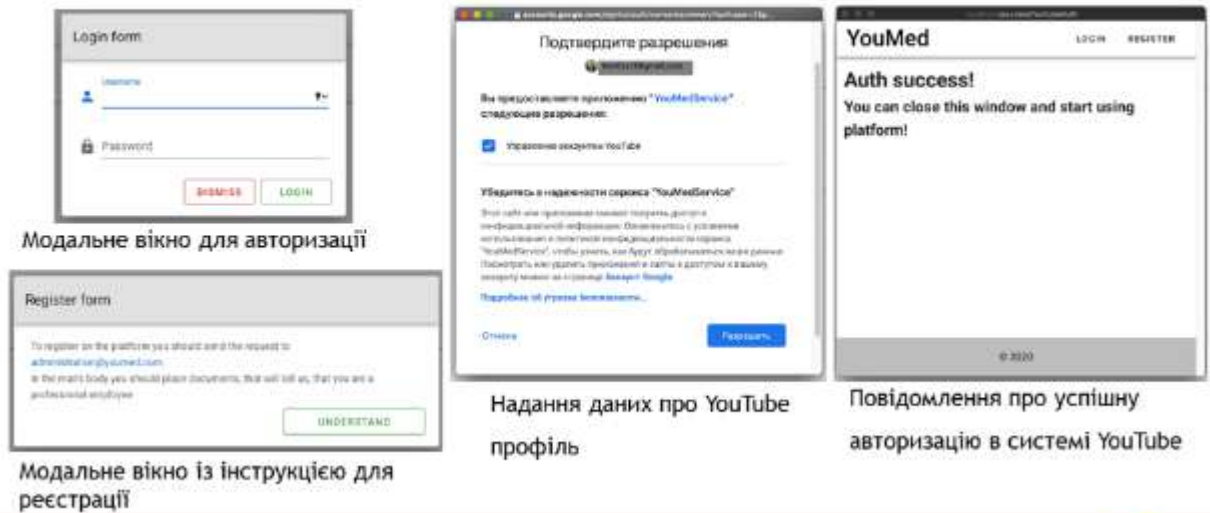
Логіка роботи асинхронних задач



16

Рисунок Б.16 – Логіка роботи асинхронних задач

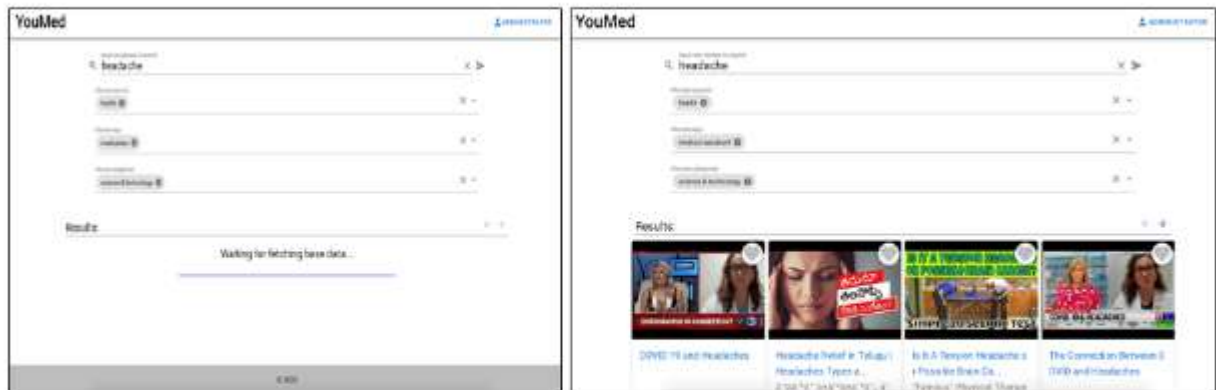
Інтерфейс програмного продукту



17

Рисунок Б.17 – Интерфейс програмного продукту

Процес пошуку відео



Повідомлення

«Waiting for fetching base data» – очікування на отримання базових даних.

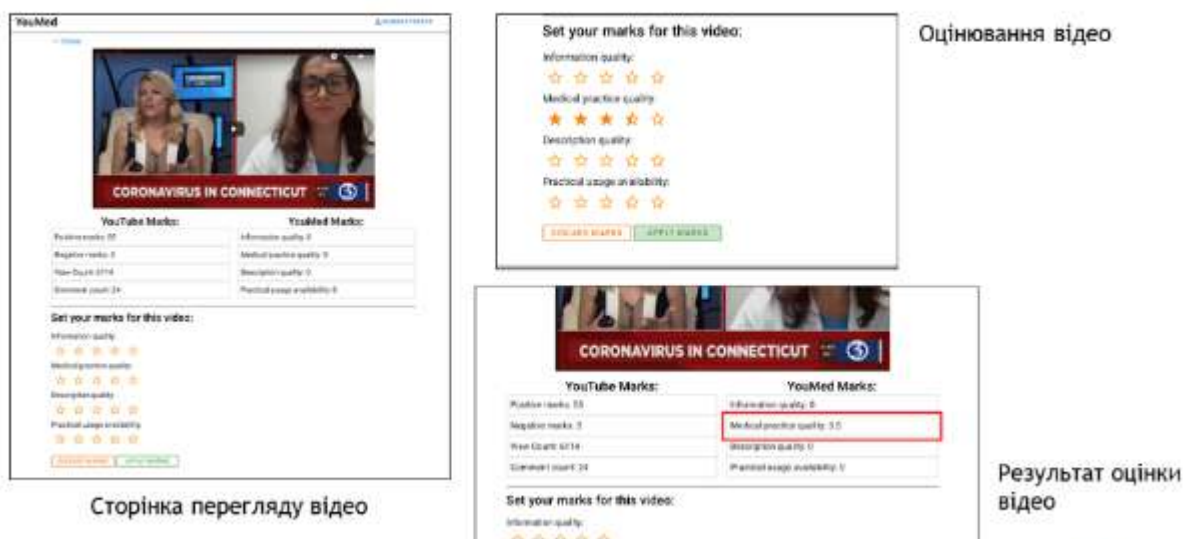
«Waiting for fetching full data» – очікування на отримання повних даних.

«Waiting for filtering data» – очікування фільтрації.

18

Рисунок Б.18 – Процес пошуку відео

Перегляд та оцінювання відео



19

Рисунок Б.19 – Перегляд та оцінювання відео

Висновки

- ▶ Алгоритм hybrid обрано як модель представлення знань для рекомендаційної системи.
- ▶ Спроектовано та розроблено: бази знань для фільтрації, базу знань сортування відео, структуру бази даних, загальну архітектуру, графічний інтерфейс.
- ▶ Проаналізовано та обрано технології для розробки:
 - ▶ для серверної частини: мова – Python, фреймворк – Django, вебсервіс – Unicorn;
 - ▶ для клієнтської частини: мова – JavaScript, бібліотеки – Vue.JS та VueX;
 - ▶ в якості СКБД – MySQL, як вебсервер – Nginx та як систему контейнеризації – Docker-compose.
- ▶ Створено рекомендаційну систему у вигляді клієнт-серверного програмного продукту для експертів в сфері медицини.
- ▶ Наукова новизна – отримано подальший розвиток гібридного алгоритму надання рекомендацій для обробки списку відео з метою надати більш релевантні результати пошуку.
- ▶ Практична значимість результатів – розроблена РС забезпечує додаткову фільтрацію та сортування списку відео, отже користувач отримує релевантний та якісний список відео, оскільки враховуються виставлені експертами оцінки відео взяті з БД.
- ▶ Результати даної магістерської роботи доповідалися на НПК Тиждень науки-2020 та на X Міжнародній НПК «Сучасні проблеми і досягнення в галузі радіотехніки, телекомунікацій та інформаційних технологій».

20

Рисунок Б.20 – Висновки