

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)

магістра

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДІВ

(назва теми)

ПРОГНОЗУВАННЯ ПІДПИСОК НА ЦИФРОВИЙ КОНТЕНТ

RESEARCH AND SOFTWARE IMPLEMENTATION OF METHODS

TO FORECAST DIGITAL CONTENT SUBSCRIPTIONS

Виконав(ла): студент(ка) 2 курсу, групи КНТ-142м

Спеціальності 121 Інженерія програмного

(код і найменування спеціальності)

забезпечення

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

БІРЮК В.О.

(ПРІЗВИЩЕ та ініціали)

Керівник ЛЬОВКІН В.М.

(ПРІЗВИЩЕ та ініціали)

Рецензент СКРУПСЬКИЙ С.Ю.

(ПРІЗВИЩЕ та ініціали)

2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет _____ ФКНТ _____
Кафедра _____ програмних засобів _____
Ступінь вищої освіти _____ магістр _____
Спеціальність _____ 121 Інженерія програмного забезпечення _____
(код і найменування)
Освітня програма (спеціалізація) _____ Інженерія програмного забезпечення _____
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф. _____
Сергій СУББОТІН _____
« _____ » _____ 2023 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

БІРЮК Владислави Олександрівни

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження та програмна реалізація методів
прогнозування підписок на цифровий контент. Research and Software
Implementation of Methods to Forecast Digital Content Subscriptions

керівник проєкту (роботи) к.т.н., доцент ЛЬОВКІН Валерій Миколайович _____,
(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від « 14 » листопада 2023 року № 437

2. Строк подання студентом проєкту (роботи) 01 грудня 2023 року

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

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно
розробити) 1. Аналіз проблеми та постановка завдань дослідження. 2. Матеріали
і методи. 3. Проєктування програмного забезпечення. 4. Основні рішення
щодо реалізації програмного забезпечення. 5. Експлуатація, тестування та
експериментальне дослідження програми.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, кількість
слайдів, плакатів)

Слайди презентації _____

6. Консультанти розділів проєкту (роботи)

Розділ	ПРІЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-5 Основна частина	ЛЬОВКІН В.М., доцент		
Нормоконтроль	КАМІНСЬКА Ж.К., асистент		

7. Дата видачі завдання « 04 » вересня 2023 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Срок виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області	2 тиждень	Розділ 1
3	Дослідження проблеми	3-5 тижні	Розділ 2
4	Розробка архітектури програми	6 тиждень	Розділ 3
5	Розробка програми	7-8 тижні	Розділ 4
6	Тестування та експериментальне дослідження програми	9-10 тижні	Розділ 5
7	Оформлення пояснювальної записки та документів до неї. Нормоконтроль та рецензування	11 тиждень	Додатки
8	Захист роботи	12 тиждень	

Студент(ка)


(підпис)

Владислава БІРЮК

(Ім'я ПРІЗВИЩЕ)

Керівник проєкту (роботи)


(підпис)

Валерій ЛЬОВКІН

(Ім'я ПРІЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи магістра:
117 с., 43 рис., 4 табл., 3 дод., 14 джерел.

ЦИФРОВИЙ КОНТЕНТ, ПІДПИСКА, ПРОГНОЗУВАННЯ,
ЧАСОВИЙ РЯД, LSTM.

Об'єкт роботи – процес прогнозування підписок на цифровий контент.
Предмет роботи – методи прогнозування підписок на цифровий контент.
Мета роботи – розробити програмне забезпечення прогнозування підписок на цифровий контент для надання і контролю доступу до цифрового контенту та короткострокового прийняття рішень на основі динаміки чисельності груп підписників.

Матеріали, методи та технічні засоби: мови Python, HTML, CSS, JavaScript, фреймворк Django, бібліотеки Tensorflow, Keras, Pandas, система керування базами даних PostgreSQL.

Результати. Запропонований метод прогнозування підписок на цифровий контент, розроблене програмне забезпечення надання доступу та керування доступом до цифрового контенту, яке інтегрує цей метод у процес роботи з цифровим контентом, та проведене експериментальне дослідження є основними результатами роботи.

Висновки. Наукова новизна роботи визначається методом прогнозування підписок на цифровий контент, який дозволяє створювати моделі прогнозування окремо для різних груп користувачів, виконуючи прогнозування кількості підписок у кожній групі користувачів на наступні дні на основі даних за попередні дні за допомогою використання нейронних мереж на основі LSTM.

Галузь використання. Індустрія цифрового контенту, надання доступу користувачам до цифрового контенту.

ABSTRACT

Explanatory note to the diploma qualifying work of the master: 117 pages, 43 figures, 4 tables, 3 appendixes, 14 sources.

DIGITAL CONTENT, SUBSCRIPTION, PREDICTION, TIME SERIES, LSTM.

The object of the work is the process of prediction of digital content subscriptions. The subject of the work is methods of prediction of digital content subscriptions. The goal of the work is to develop software for predicting prediction of digital content subscriptions for providing and managing access to digital content and short-term decision-making based on the dynamics of the number of subscribers in groups.

Materials, methods and technical means: Python, HTML, CSS, JavaScript languages, Django framework, Tensorflow, Keras, Pandas libraries, PostgreSQL database management system.

The results. The proposed method of prediction of digital content subscriptions, the developed software for providing access and managing access to digital content, which integrates this method into the process of working with digital content, and the conducted experimental research are the main results of the work.

Conclusions. The scientific novelty of the work is determined by the method of prediction of digital content subscriptions, which allows to create prediction models separately for different groups of users, making predictions of the number of subscriptions in this group of users for the following days based on data from previous days using neural networks based on LSTM.

Field of use. Digital content industry, providing users with access to digital content.

ЗМІСТ

	С.
Перелік скорочень та умовних познач.....	8
Вступ.....	9
1 Аналіз проблеми та постановка завдань дослідження	10
1.1 Огляд програмного забезпечення надання і керування доступом до цифрового контенту	10
1.2 Порівняння аналогів	14
1.3 Висновки за розділом 1	15
2 Матеріали і методи.....	16
2.1 Задача прогнозування підписок на цифровий контент	16
2.2 Нейронні мережі на основі архітектури LSTM.....	17
2.3 Метод прогнозування підписок на цифровий контент	19
2.4 Висновки за розділом 2	21
3 Проєктування програмного забезпечення	23
3.1 Вибір засобів розробки	23
3.2 Моделювання програмного забезпечення на основі функціональних вимог	24
3.3 Проєктування структури бази даних.....	30
3.4 Висновки за розділом 3	38
4 Основні рішення щодо реалізації програмного забезпечення	39
4.1 Порядок роботи програмного забезпечення	39
4.2 Рішення з реалізації вебзастосунку	41
4.3 Висновки за розділом 4	57
5 Експлуатація, тестування та експериментальне дослідження програми	58
5.1 Опис програми.....	58
5.2 Експлуатація програми.....	58
5.3 Експериментальне дослідження	65
5.4 Висновки за розділом 5	73
Висновки	74

Перелік джерел посилання	75
Додаток А Технічне завдання	77
Додаток Б Текст програми	84
Додаток В Слайди презентації.....	109

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

ARIMA – AutoRegressive Integrated Moving Average;

LSTM – Long Short-Term Memory;

БД – база даних.

ВСТУП

Цифровий контент є частиною життя майже кожної сучасної людини. Фактично він є навіть певним маркером розвитку сучасної цивілізації. У підсумку впливаючи на величезну кількість процесів життя сучасної людини. Автори цифрового контенту при цьому мають цілий ряд проблем. З одного боку вони обумовлені тим, що автором може бути будь-яка людина. І не тільки може, але і фактично є. Але з іншого боку, якщо людина здатна створювати оригінальний контент, то далі виникає проблема знаходження адекватної платформи для представлення власних результатів. За наявності відповідних вмінь автори шукають адекватну платформу. А з цим нерідко виникають проблеми. Платформи, що збирають кошти, нерідко вводять значні обмеження щодо контенту, що впливає, наприклад, на авторів з України в поточних умовах.

Разом з доступом до контенту виникає проблема забезпечення прав авторів. Фактично для цього необхідно забезпечити платний доступ до таких матеріалів. Однак, при забезпеченні такого доступу виникають різні варіанти, як це зробити.

Нерідко застосунки створюються для забезпечення роботи окремих сервісів. У свою чергу окремі автори фактично мають використовувати платформи, подібні до платформ збору коштів. Виникає фрагментованість, коли є багато платформ для підтримки доступу до централізованих сервісів та платформ для збору коштів для авторів. Багато користувачів фактично не знають або не звертають уваги на такі платформи. Об'єднання ж цих ідей може принести вигоду як окремому сервісу, так і авторам цифрового контенту. Однак, окрім наявності такого інструменту важливим є також і наявність інструментів аналізу та прогнозування підписок на цифровий контент, щоб можна було приймати рішення про те, як розвивати сервіс, каналі надалі, на яку аудиторію орієнтуватися.

1 АНАЛІЗ ПРОБЛЕМИ ТА ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ

1.1 Огляд програмного забезпечення надання і керування доступом до цифрового контенту

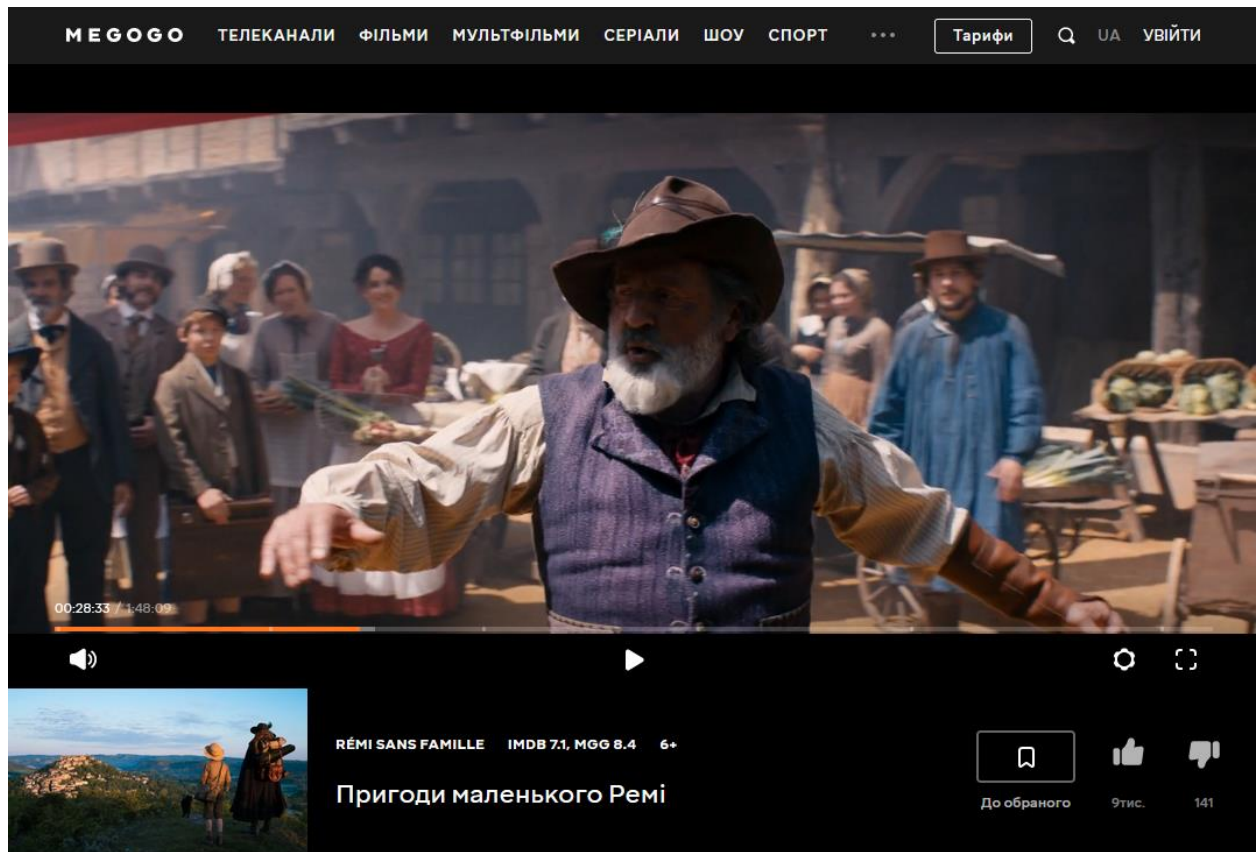
Оскільки, як було описано у вступі, існує дві течії щодо роботи з цифровим контентом, коли фактично сервіси представляють інтереси широковідомих авторів, а ті в свою чергу отримують певну загальну плату за надання їхнього цифрового контенту, та коли автори самі звертаються до користувачів, надаючи їм свій цифровий контент за певну сплату. Відповідно у даному огляді потрібно розглянути обидві течії, а в підсумку виділити важливі можливості для реалізації у програмному забезпеченні, яке має об'єднати їх разом.

Відносно першого напрямку існує ряд сервісів зокрема в Україні, які зазвичай забезпечують перегляд телеканалів, фільмів, мультфільмів. Однак, не всі вони є однаковими за особливостями та функціональністю. В якості базового прикладу було обрано сервіс MEGOGO [1] (рис. 1.1), адже він має найбільш широкі можливості серед розглянутих. По-перше, він містить аудіоконтент, а до того ж він надає доступ до матеріалів блогерів.

За функціональністю MEGOGO надається багато різних засобів, включаючи пошук контенту, в тому числі різного типу. Частина матеріалів є безкоштовною, частина потребує підписки на сервіс. Підписки можуть бути на різний період часу, а також на конкретний контент. Враховуючи, що велика частина цифрового контенту є відомими фільмами, мультфільмами, такий варіант є зрозумілим. Відносно контенту користувачі можуть залишати власні уподобання, створювати відгуки, визначаючи оцінки, що в свою чергу дозволяє створювати власний рейтинг контенту. Також користувачі можуть зберігати контент для подальшого перегляду, отримувати доступ до історії переглядів.

Проте стосовно роботи блогерів варто відзначити, що вони не можуть збирати кошти за свій контент самотійно, тобто надається можливість

переглянути контент цих авторів на сайті або в застосунку. Далі ж вони можуть отримувати від самого сервісу певну сплату, але конкретний користувач не може сплатити за конкретний контент блогерів. Тобто підписка фактично в цьому відношенні є загальною.



ОПИС ПЕРСОНІ І КОМАНДИ ТРЕЙЛЕРИ І КАДРИ 64 ВІДГУКИ

2019, Пригоди, Драми, Сімейні, Франція

Full HD 108 хвилин

Історія десятирічного хлопчика Ремі, що втік від своєї названої матері в пошуках рідної сім'ї, і приєднався у своїй нелегкій подорожі до компанії мандрівного музиканта, разом

Рисунок 1.1 – Інтерфейс вебсервісу MEGOGO

Друга течія серед існуючого програмного забезпечення представлена сервісом Patreon [2] (рис. 1.2). Patreon направлений на використання користувачами, здатними створювати цифровий контент. Цим користувачам надається можливість за рахунок різноманітних підписок отримати монетизацію стосовно власного контенту.

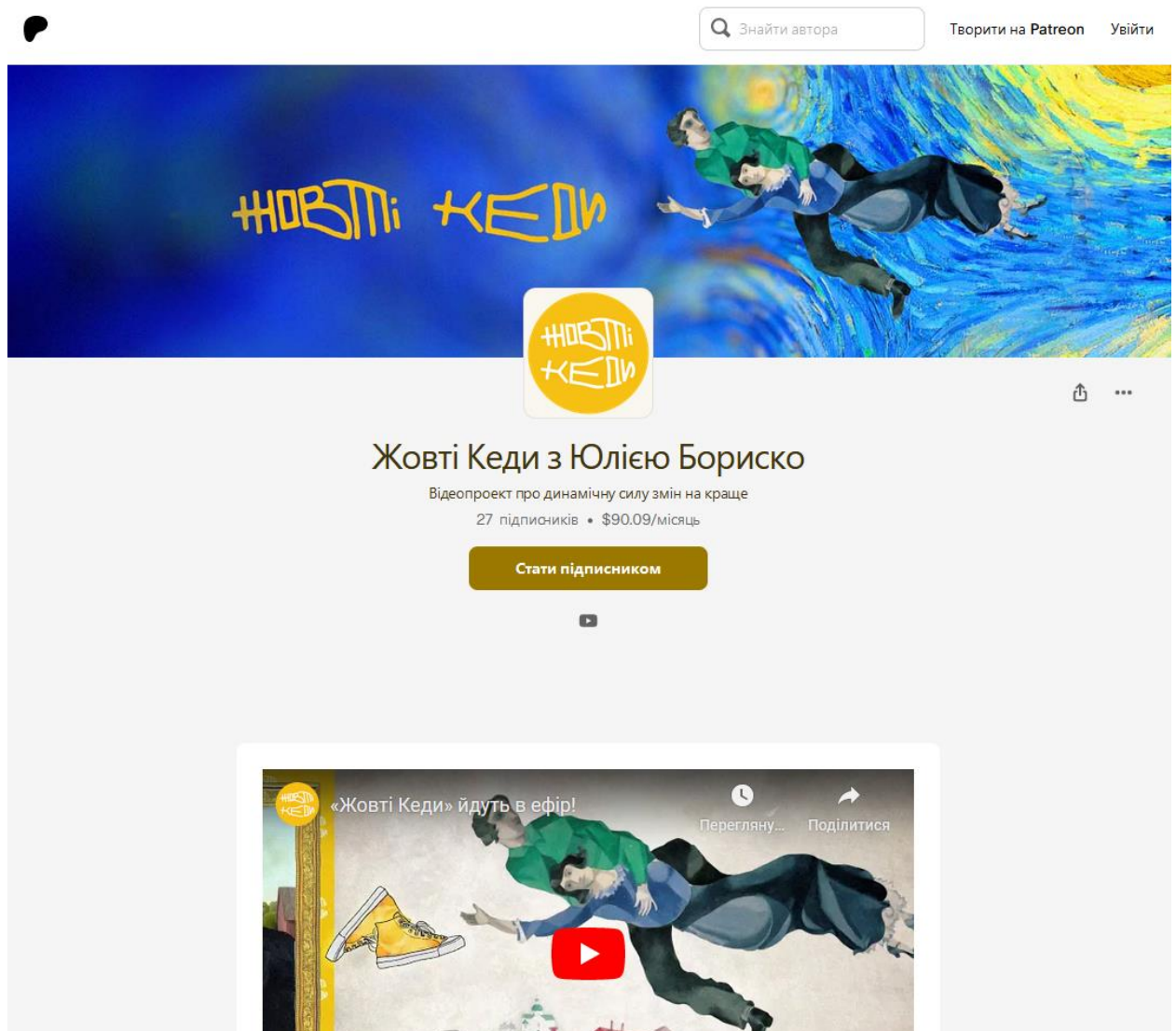


Рисунок 1.2 – Інтерфейс вебсервісу Patreon

Підписки на Patreon бувають різного типу, але серед матеріалів також можуть виставлятися і безкоштовні. Для платного доступу підписка може визначати, до якого контенту надається доступ, та відповідно змінюється сума за підписку. За певних підписок може надаватися можливість спілкуватися з представниками авторських каналів. Це є значною відмінністю від попередньої течії, проте варто вказати також на те, що загальної підписки не існує, тобто контент може бути або безкоштовним, або вимагати підписки на конкретний канал.

Сервіс має дуже обмежений пошук. Фактично потрібно задати конкретну назву каналу. Тобто знайти нових авторів за таких умов фактично

неможливо, адже тоді потрібно вгадати назву. Можливості шукати за категоріями або ще якимось чином немає. Відповідно потрібні спочатку інші канали комунікації з авторами, щоби потім перейти на Patreon. А сам сервіс тоді використовується саме для оформлення підписок і для перегляду ексклюзивного контенту фактично, тобто контенту, відсутнього на інших платформах.

Подібним сервісом є Вuymeascoffee [3] (рис. 1.3), який в основі передбачає сплату невеликої суми для авторів. Це може не передбачати наявність доступу до певного контенту, а лише використовується як точка доступу до саме монетизації. Тобто таким чином користувачі можуть віддячити автору.

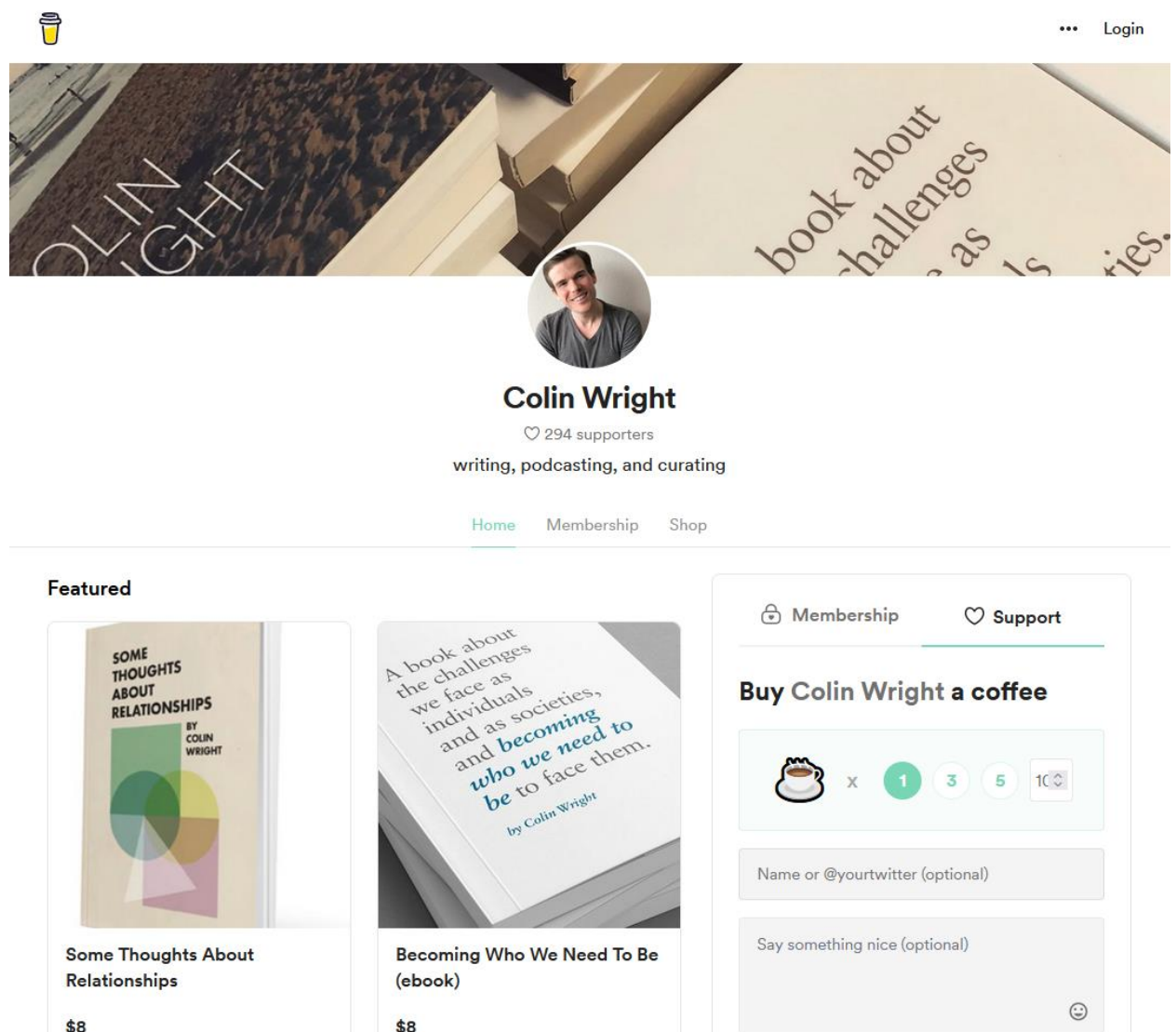


Рисунок 1.3 – Інтерфейс вебсервісу Вuymeascoffee

Таким чином, Буymeasoffee є достатньо особливим стосовно функціональності. Сплата може бути або одноразовою, або періодичною. Також надається можливість продати певний контент користувачам.

1.2 Порівняння аналогів

Як основні представники двох течій у даній сфері, були обрані MEGOGO та Patreon (табл. 1.1). Можливості цих сервісів були порівняні для того, щоб визначити можливості підсумкового вебзастосунку, а також щоби визначити його особливості. MEGOGO та Patreon представляють різні варіанти підписки, але в розробці, що виконується, ці ідеї мають бути об'єднані. Окрім того розробка повинна дозволити прогнозувати підписки на цифровий контент. Також було помічено, що в Patreon відсутні варіанти оцінювання (окрім уподобань) цифрового контенту, а також і повноцінний пошук, тому знаходити новий контент у самій програмі фактично неможливо.

Таблиця 1.1 – Результати аналізу вебзастосунків для надання доступу та керування доступом до цифрового контенту

Критерій порівняння	MEGOGO	Patreon	Розробка
Підписка на авторський канал	—	+	+
Загальна підписка	+	—	+
Можливість поставити запитання автору	—	+	+
Багатокритеріальний пошук	+	—	+
Оцінювання цифрового контенту	+	—	+
Прогнозування кількості підписників	—	—	+

1.3 Висновки за розділом 1

Завдання, які відповідають меті роботи та мають бути виконані:

- проаналізувати існуючі вебзастосунки для надання доступу та керування доступом до цифрового контенту;
- розробити метод прогнозування підписок на цифровий контент;
- проектування програмного забезпечення надання доступу та керування доступом до цифрового контенту;
- реалізувати вебзастосунок для надання доступу та керування доступом до цифрового контенту;
- виконати експериментальне дослідження прогнозування підписок на цифровий контент.

У розділі 1 виконано завдання проаналізувати існуючі вебзастосунки для надання доступу та керування доступом до цифрового контенту. Розглянуто застосунки MEGOGO, Patreon, Буymeacoffee, виокремлено необхідні функції для реалізації в розробці цієї роботи, визначено, що важливою є організація пошуку цифрового контенту, розділення доступу та забезпечення різних варіантів підписки, надання інструментів аналізу та прогнозування підписок.

2 МАТЕРІАЛИ І МЕТОДИ

2.1 Задача прогнозування підписок на цифровий контент

Прогнозування підписок на цифровий контент передбачає визначення кількості підписок на цифровий контент на відповідний день. Це означає кількість підписок, які є активними на задану дату.

Сама задача прогнозування підписок на цифровий контент є встановленням кількості підписок на наступний день або дні за відомими даними попередніх днів. Таким чином, це задача прогнозування часового ряду. Тобто якщо відомо, що у минулі Q днів за кожен день встановлено кількість підписок, то потрібно виконати прогнозування на наступні Q або менше днів, щоб встановити, як буде змінюватися кількість підписників у майбутньому.

Розв'язання задачі прогнозування підписок на цифровий контент є основою короткострокового прийняття рішень, що стосуються динаміки чисельності підписників або їх окремих груп. Якщо прогнозування відбувається для окремих груп, то це дозволяє зрозуміти співвідношення між цими групами в найближчий момент. Окрім прийняття стратегічних рішень на тривалий період потрібні також і короткострокові рішення. Це дозволяє зрозуміти, чи варто змінювати рекламну кампанію найближчим часом або в який саме момент це краще зробити. Таке прогнозування дозволяє зрозуміти перспективи на тиждень. Це в свою чергу допомагає зрозуміти, як краще рекламувати сервіс, окремий контент, який саме контент краще рекламувати, розуміючи, користувачів яких саме груп стає більше або менше. Зміни в групах призводять до розуміння необхідних змін у підтримці користувачів, адже різні групи генерують різну кількість звернень.

Для розв'язання такого типу задач на даний момент існує ряд алгоритмів. Це можуть бути алгоритми на основі лінійної регресії, AutoRegressive Integrated Moving Average (ARIMA), нейронні мережі. Серед нейронних мереж найбільш придатними для прогнозування часових рядів є

рекурентні нейронні мережі, серед яких такими, що дозволяють враховувати довгострокові залежності є нейронні мережі на основі архітектури Long Short-Term Memory (LSTM) [4]-[7].

2.2 Нейронні мережі на основі архітектури LSTM

Особливістю LSTM є внутрішня структура її чарунок (рис. 2.1), яка дозволяє запам'ятовувати довготривалі в часі залежності. Це і дозволило в підсумку LSTM проявити себе відносно підвищення точності прогнозування в різних задачах прогнозування часових рядів.

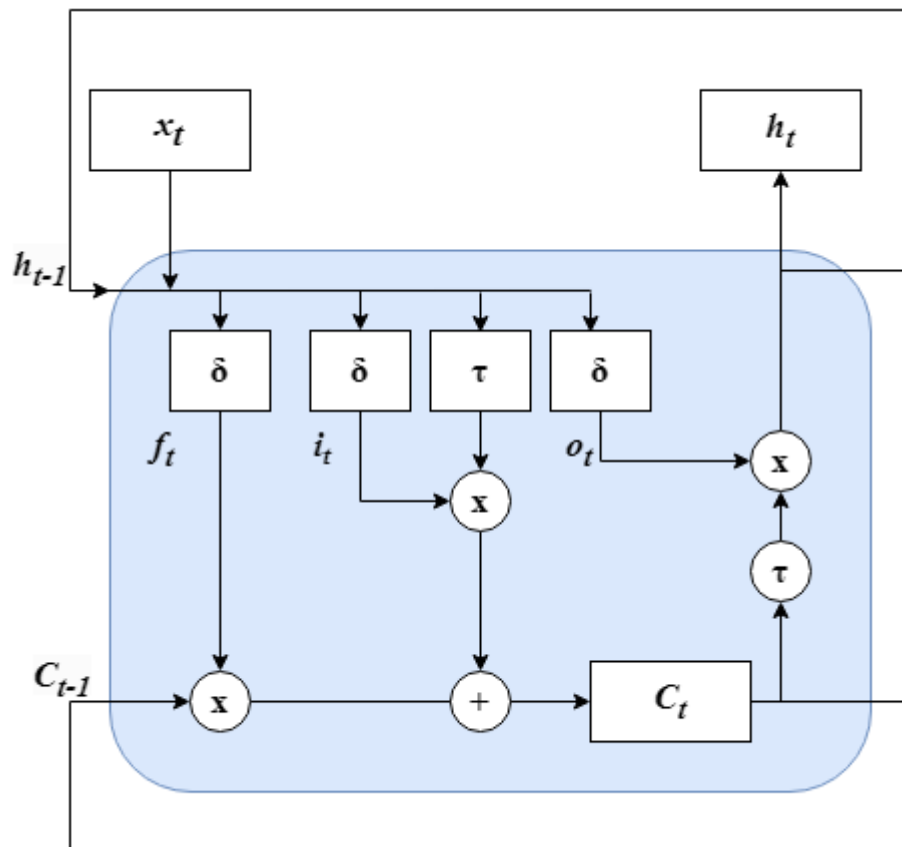


Рисунок 2.1 – Архітектура LSTM

Чарунка LSTM має в якості входів h_{t-1} і C_{t-1} , що мають значення, які відповідають моменту часу $t-1$. Всі дії всередині чарунки відповідають моменту часу t .

Стан чарунки представляється C_t , при чому це поточний стан, а відповідно попередній стан визначається C_{t-1} . Вихідне значення представляється h_t , при чому це вихідне значення, яке обчислюється в поточний момент чарункою, а відповідно попереднє вихідне значення визначається h_{t-1} .

На рис. 2.1 показано активаційні функції, які використовуються в обчисленнях у кожному відповідному вентилі. Позначкою δ визначено логістичну сигмоїдну функцію активації, а τ – активаційна функція гіперболічний тангенс.

LSTM містить всередині 4 різні типи вентилів, які впливають на обчислення вихідного результату:

- вентиль забуття, який представлений значенням f_t , що обчислюється самим вентилям забуття: значення на виході вентиля визначається значенням від 0 до 1, яке впливає на те, на скільки попередній стан має зберегтися або забути;

- вхідний вентиль, який представлений значенням i_t , що обчислюється самим вхідним вентилям: працює також з тими ж даними, що і вентиль забуття, але результат роботи цього вентиля потрапляє до стану LSTM, використовуючи при цьому спочатку множення, а потім додавання до попереднього стану;

- вхідний модуляційний вентиль, який представлений значенням, отриманим через функцію активації τ , що обчислюється самим вхідним модуляційним вентилям;

- вихідний вентиль, який представлений значенням o_t , що обчислюється самим вихідним вентилям та дозволяє сформувати вихідний стан [7].

Позначкою x_t вказано вхідний вектор, яким фактично визначаються вхідні дані.

2.3 Метод прогнозування підписок на цифровий контент

Метод прогнозування підписок на цифровий контент направлений на прогнозування кількості підписок, таким чином вирішуючи задачу прогнозування підписок на цифровий контент, але в ракурсі саме груп користувачів. Враховуючи логіку проведення маркетингових досліджень, користувачі були розділені на групи у відповідності з віком та статтю. Це дозволяє потім приймати рішення про те, який контент варто радити такій групі користувачів.

Тож у підсумку користувачі були розділені на наступні групи:

- чоловіки до 24 років включно;
- чоловіки від 25 до 34 років включно;
- чоловіки від 35 до 44 років включно;
- чоловіки від 45 до 54 років включно;
- чоловіки від 55 до 64 років включно;
- чоловіки віком від 65 років;
- жінки до 24 років включно;
- жінки від 25 до 34 років включно;
- жінки від 35 до 44 років включно;
- жінки від 45 до 54 років включно;
- жінки від 55 до 64 років включно;
- жінки віком від 65 років.

Для кожної такої групи модель прогнозування підписок на цифровий контент будується окремо. Але для цього необхідно спочатку витягти, обробити дані з БД, а тоді вже, підготувавши дані для окремої групи, створити відповідну модель.

Схематичне представлення методу прогнозування підписок на цифровий контент представлено на рис. 2.2.

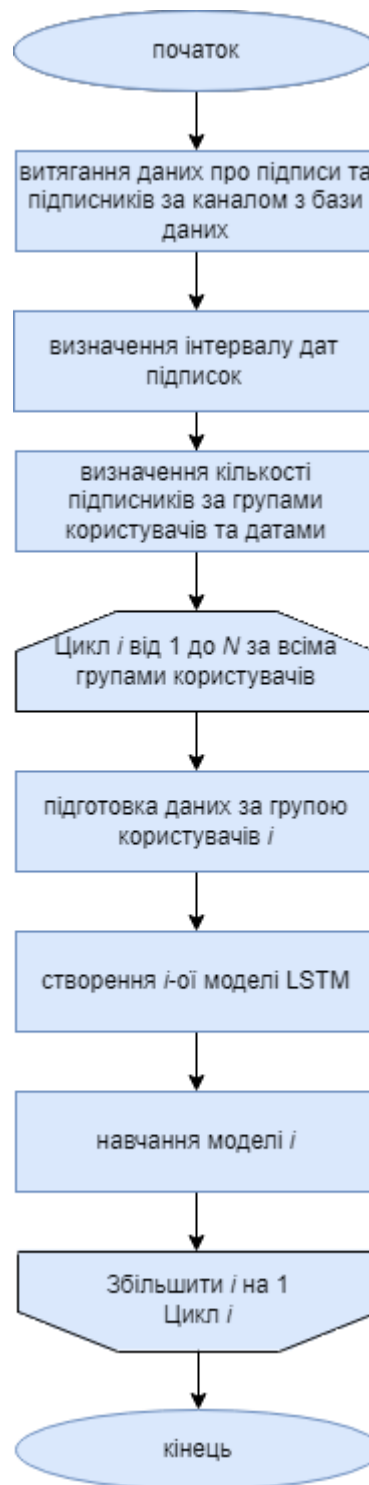


Рисунок 2.2 – Метод прогнозування підписок на цифровий контент

Аналізуючи дані в БД, необхідно визначити мінімальну та максимальну доступні дати, за які були зафіксовані підписки на цифровий контент. Тоді для кожної дати в цьому проміжку має бути проаналізовано, яка кількість активних підписок була на цю дату та яка кількість до якої групи користувачів належить. Для цього за кожною підпискою з даних

користувача має бути визначено, до якої вікової групи належить цей користувач та яку стать має. У результаті визначається за кожною датою кількість підписок за кожною групою користувачів. У цих обчисленнях стандартно припускається, що якщо підписка закінчується у задану дату, то вона вважається активною.

Значення за всім доступним набором дат мають бути сформовані для кожної групи окремо. Тоді далі для кожної такої групи за визначеними таким чином даними створюються окремі моделі. Кожна модель базується на архітектурі LSTM.

Запропоновано використовувати модель з 2 шарів. Кількість днів, яка подається на вхід, має враховувати розмір всієї вибірки даних. Стандартно кількість днів на вхід дорівнює кількості днів на виході. Кількість днів, на яку відбувається прогнозування, також має враховувати розмір вибірки. У даній роботі виконано прогнозування на 7 наступних днів за значеннями 7 попередніх днів.

Після того, як модель створено, має бути її навчено, для чого і застосовуються дані, виділені для відповідної групи користувачів. Критерієм навчання є середньоквадратична похибка.

Під час експериментального дослідження було виконано після цього тестування, для якого виділялось 20 % даних. За реального використання відповідно модель використовується для визначення значень на наступні 7 днів за даними 7 попередніх.

2.4 Висновки за розділом 2

У розділі 2 описано задачу прогнозування підписок на цифровий контент, розглянуто архітектуру LSTM, описано запропонований метод прогнозування підписок на цифровий контент.

Наукова новизна роботи визначається методом прогнозування підписок на цифровий контент, який дозволяє створювати моделі

прогнозування окремо для різних груп користувачів, виконуючи прогнозування кількості підписок у кожній групі користувачів на наступні дні на основі даних за попередні дні за допомогою використання нейронних мереж на основі LSTM.

3 ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Вибір засобів розробки

З програмою мають працювати як клієнти, яким потрібно надати доступ до цифрового контенту, так і відповідальні за ведення авторських каналів, адміністратори, яким потрібно керувати цим доступом. Всі вони потребують доступу до самого програмного забезпечення, знаходячись в різних місцях, а тому логічним є створення вебзастосунку. Для реалізації вебзастосунку, враховуючи необхідність реалізації методу прогнозування підписок на цифровий контент, було визначено мови Python і C# (табл. 3.1).

Таблиця 3.1 – Порівняння мов програмування для реалізації програмного забезпечення надання і керування доступом до цифрового контенту

Критерій порівняння / мова програмування	Python	C#
Система типізації	Динамічна	Статична
Ліцензія	Відкрита	Відкрита
Фреймворк розробки вебзастосунків	Django	ASP.NET
Простота імплементції існуючих рішень	++	+
Інструменти для роботи з даними	++	+—
Інтерфейс для роботи з нейронними мережами	Keras, Tensorflow	Keras.NET, Accord.NET, ML.NET
Підтримка в сфері машинного навчання	++	+—

І C#, і Python мають потужні засоби для реалізації вебзастосунків, але в сфері машинного навчання Python виявився більш придатним варіантом, адже має кращу підтримку (набагато більше матеріалів про виконання машинного навчання, обробку даних тощо), забезпечує багато інструментів для роботи з даними, а також має більш потужні засоби та інтерфейси для роботи з нейронними мережами.

3.2 Моделювання програмного забезпечення на основі функціональних вимог

Клієнт використовує програмне забезпечення за допомогою наступних сценаріїв:

- відтворення цифрового контенту;
- перегляд переліку цифрового контенту за проєктом;
- перегляд авторського каналу;
- перегляд профілю клієнта;
- оновлення власного профілю;
- перегляд усього цифрового контенту за авторським каналом;
- пошук авторських каналів;
- пошук цифрового контенту;
- пошук проєктів цифрового контенту;
- збереження цифрового контенту, до якого альтернативним сценарієм є вилучення цифрового контенту зі збереженого, що здійснюється за умов, що певний цифровий контент вже був збережений, а тоді повторна дія має призводити до вилучення його зі збереженого;
- перегляд збереженого цифрового контенту;
- уподобання цифрового контенту, до якого альтернативним сценарієм є зняття уподобання з цифрового контенту, що здійснюється за умов, що певний цифровий контент вже був уподобаний, а тоді повторна дія має призводити до вилучення його з уподобаного;

– виставлення антипатії цифровому контенту, до якого альтернативним сценарієм є зняття антипатії з цифрового контенту, що здійснюється за умов, що певний цифровий контент вже був співвіднесений за антипатією, а тоді повторна дія має призводити до вилучення його з антипатії;

- перегляд уподобаного цифрового контенту;
- перегляд цифрового контенту в антипатіях;
- коментування цифрового контенту;
- перегляд історії відтворення цифрового контенту;
- перегляд створених коментарів;
- перегляд оцінок за авторським каналом;
- перегляд оцінок за цифровим контентом;
- виставлення оцінки авторському каналу;
- виставлення оцінки цифровому контенту;
- перегляд найкращих авторських каналів;
- перегляд найкращого цифрового контенту;
- перегляд виставлених оцінок авторським каналам;
- перегляд виставлених оцінок цифровому контенту;

– авторизація облікового запису, до якого альтернативним сценарієм є реєстрація облікового запису, що здійснюється за умов, що клієнт не має власного облікового запису у вебзастосунку, включаючи обліковий запис відповідального за ведення авторського каналу, або якщо він має другий тип облікового запису, то не бажає його використовувати за цим принципом;

- оформлення запиту на технічну підтримку;
- перегляд створених запитів на технічну підтримку;
- створення спілкування щодо авторського каналу;
- перегляд створених спілкувань за авторськими каналами;
- перегляд варіантів загальної підписки;
- перегляд варіантів підписки на авторський канал;
- оформлення загальної підписки (рис. 3.1).

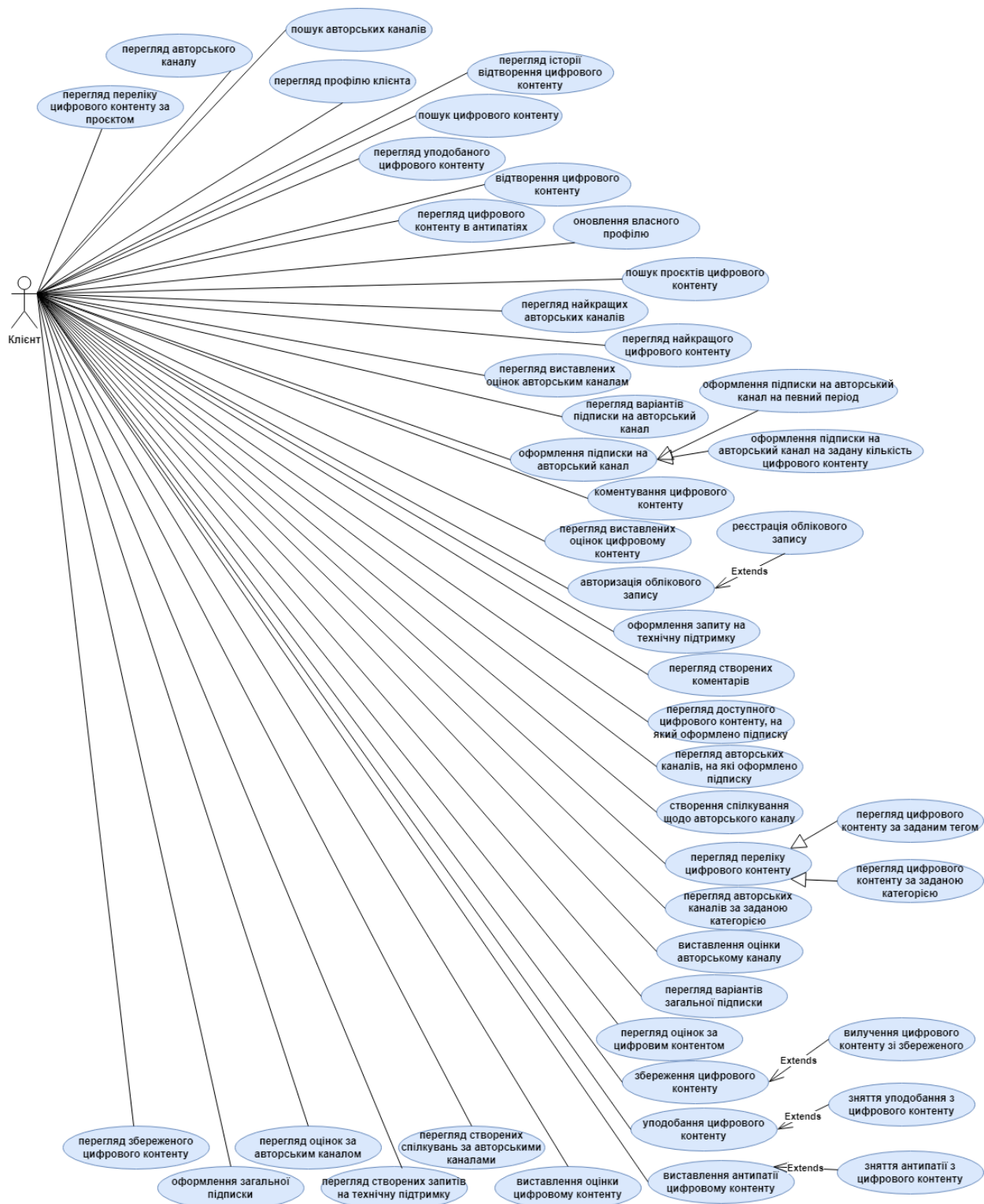


Рисунок 3.1 – Діаграма прецедентів клієнта

Інші сценарії клієнта:

- оформлення підписки на авторський канал на певний період;
- оформлення підписки на авторський канал на задану кількість цифрового контенту;

- перегляд доступного цифрового контенту, на який оформлено підписку;

- перегляд авторських каналів, на які оформлено підписку;

- перегляд цифрового контенту за заданим тегом;

- перегляд цифрового контенту за заданою категорією;

- перегляд авторських каналів за заданою категорією.

Відповідальний за ведення авторського каналу використовує програмне забезпечення за допомогою наступних сценаріїв:

- створення авторського каналу, до якого альтернативним сценарієм є редагування опису авторського каналу, яке виконується за умов наявності авторського каналу і необхідності внести зміни в опис;

- додавання відповідального за ведення контенту, до якого альтернативним сценарієм є вилучення відповідального за ведення контенту за умов, коли дані були внесені некоректно;

- додавання категорії до авторського каналу, до якого альтернативним сценарієм є вилучення категорії з авторського каналу за умов, коли дані були внесені некоректно;

- додавання проєкту цифрового контенту до авторського каналу, до якого альтернативними сценаріями є редагування опису проєкту цифрового контенту за умов, коли проєкт вже був доданий і дані необхідно оновити, та вилучення проєкту цифрового контенту з авторського каналу за умов, коли проєкт вже був доданий, але необхідно витерти всі внесені дані;

- додавання цифрового контенту до проєкту, до якого альтернативними сценаріями є редагування цифрового контенту за умов, коли він був уже доданий, але дані застарілі або внесені помилково, та вилучення цифрового контенту з проєкту, коли він вже був доданий, але це було зроблено помилково;

- додавання категорії до цифрового контенту, до якого альтернативним сценарієм є вилучення категорії з цифрового контенту за умов, коли категорію було додано, але помилково;

- додавання тегу до цифрового контенту, до якого альтернативним сценарієм є видалення тегу з цифрового контенту за умов, коли тег було додано, але помилково;

- опублікування цифрового контенту, до якого альтернативним сценарієм є архівування цифрового контенту за умов, коли контент вже не повинен демонструватися клієнтам, але був внесений раніше;

- додавання матеріалу до цифрового контенту, до якого альтернативними сценаріями є редагування матеріалу в складі цифрового контенту, коли матеріал був доданий, але дані некоретні, та видалення матеріалу з цифрового контенту, коли він був доданий, але більше не має бути присутнім у цифровому контенті;

- авторизація відповідального за ведення авторського каналу, до якого альтернативним сценарієм є реєстрація відповідального за ведення авторського каналу за умов, коли в користувача ще немає облікового запису;

- додавання інформації про людину, яка може бути учасником створення цифрового контенту;

- додавання учасника створення авторського каналу, до якого альтернативними сценаріями є редагування інформації про учасника створення авторського каналу за умов, коли учасник вже був доданий до каналу, але опис є некоректним, та видалення учасника створення авторського каналу за умов, коли учасник був доданий, але зроблено це помилково;

- додавання учасника створення цифрового контенту, до якого альтернативними сценаріями є редагування інформації про учасника створення цифрового контенту за умов, коли учасник вже був доданий до контенту, але опис є некоректним, та видалення учасника створення цифрового контенту за умов, коли учасник був доданий, але зроблено це помилково;

- відправлення повідомлення до спілкування, до якого альтернативним сценарієм є вилучення повідомлення зі спілкування за умов, коли повідомлення відправлено, але зроблено це помилково;
- перегляд повідомлень, які надійшли за авторським каналом (рис. 3.2).

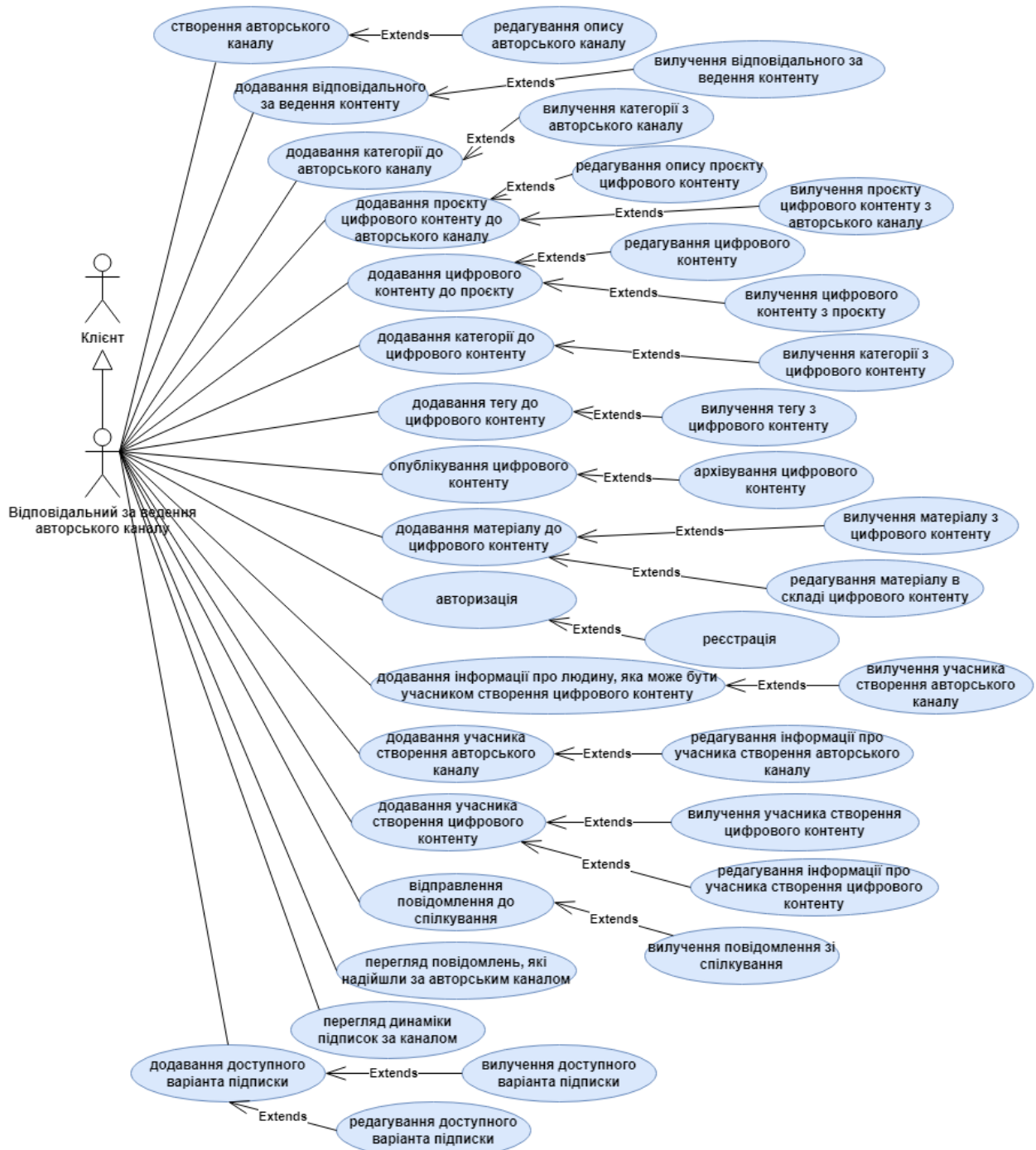


Рисунок 3.2 – Діаграма прецедентів відповідального за ведення авторського каналу

Інші сценарії відповідального за ведення авторського каналу:

- перегляд динаміки підписок за каналом;
- додавання доступного варіанта підписки, до якого альтернативними сценаріями є редагування доступного варіанта підписки за умов, коли варіант підписки внесений, але його параметри змінились, та видалення доступного варіанта підписки за умов, коли раніше доданий варіант вже не є актуальним і більше не буде використовуватися.

Відповідальний за ведення загального каналу використовує програмне забезпечення за допомогою наступних сценаріїв (рис. 3.3):

- перегляд станів загальних підписок;
- перегляд запитів на технічну підтримку за сервісом;
- фіксування запиту на технічну підтримку;
- внесення результату оброблення запиту на технічну підтримку;
- оновлення результату оброблення запиту на технічну підтримку.



Рисунок 3.3 – Діаграма прецедентів відповідального за ведення загального каналу

3.3 Проєктування структури бази даних

Від визначення структури роботи з даними фактично залежить не тільки робота всього створеного в результаті вебзастосунку в цілому, але і зокрема робота запропонованого методу прогнозування підписок на

цифровий контент, який в підсумку до складу цього вебзастосунку входить. Це пов'язано з тим, що для організації роботи методу необхідна вибірка даних. Для накопичення вибірки даних необхідно, щоби всі потрібні для роботи дані були накопичені в базі даних (БД), а потім з неї в потрібний момент витягнуті. Відповідно всі потрібні характеристики для кожної пов'язаної ситуації мають бути враховані. Тому під час проєктування структури БД були враховані одночасно і функціональні вимоги за результатами моделювання програмного забезпечення, і вимоги методу, який було запропоновано.

Спочатку було вирішено питання структуризації цифрового контенту, що є центральним поняттям всієї предметної області. До відповідних сутностей і в подальшому таблиць БД належать:

- авторський канал, на якому і демонструється цифровий контент, що представлений таблицею БД `digsoapp_authorhood`: при організації вебзастосунку фактично заплановано, що один канал представляє загальний цифровий контент, який надається самим сервісом (якщо це необхідно, то відповідно може бути не створений), а інші є авторськими, які ведуть самі автори або їхні колеги;

- контакти, відповідальні за ведення контенту, якими можуть бути або самі автори і відповідно власники авторських каналів, або їхні колеги (для ведення окремого каналу потрібно мати окремий обліковий запис): представлені таблицею `digsoapp_contact`;

- проєкти, які дозволяють згрупувати цифровий контент, і представлені таблицею `digsoapp_project`: один авторський канал повинен мати хоча би один проєкт, а той у свою чергу може об'єднувати різний цифровий контент;

- цифровий контент, який представляє собою різні матеріали, включаючи потенційно відеоматеріали, аудіо, текстові матеріали, фотографії, що знаходяться в таблиці `digsoapp_content`, де фактично представлені

посилання на матеріали, а самі матеріали представлені окремими завантаженими файлами, що знаходяться на сервері;

- категорії, які дозволяють об'єднати різний цифровий контент або авторські канали та представлені таблицею `digsoapp_category`;

- теги, які дозволяють додатково об'єднати саме цифровий контент за різними позначками, кількість яких відповідно є більшою ніж кількість категорій: представлені таблицею `digsoapp_tag`;

- додаткові матеріали до цифрового контенту, які розширюють основний центровий контент, представлені в таблиці `digsoapp_stuff`: цифровий матеріал містить основний матеріал, але він може розширюватися, якщо це потрібно (наприклад, відео може не бути необхідності розширювати, але можливо до відео додано фотографії зі зйомок або якісь ілюстративні матеріали, відповідно основне відео тоді є цифровим контентом з таблиці `digsoapp_content`, а фотографії, які розширюють відео, відповідно знаходяться в цій таблиці);

- ілюстративний матеріал або обкладинка до цифрового контенту, представлений таблицею `digsoapp_leadstuff`: фактично може бути основним зображенням, яким відображується цифровий контент до того, як було відкрито основну сторінку.

Клієнти сервісу, які переглядають цифровий контент, оформлюють підписки на цифровий контент, є одним з двох типів користувачів та представлені в таблиці `digsoapp_customer`. Другий тип користувачів є користувачами, пов'язаними з роботою з авторськими каналами (помічники). Дані про них знаходяться в таблиці `digsoapp_contact`.

Клієнти можуть реагувати на цифровий контент наступними способами за допомогою відповідних сутностей:

- коментарі на цифровий контент знаходяться в таблиці `digsoapp_comment`;

- оцінки, виставлені всьому авторському каналу знаходяться в таблиці `digsoapp_authorassessment`;

- оцінки, виставлені цифровому контенту знаходяться в таблиці `digsoapp_contentassessment`;

- уподобання, виставлені цифровому контенту знаходяться в таблиці `digsoapp_contentlike`: уподобання відрізняються від оцінок тим, що вони не передбачають виставлення балів, тобто це не рецензування, а тільки висловлення загальної позитивної або негативної (у випадку подальшої антипатії) думки;

- антипатії до цифрового контенту знаходяться в таблиці `digsoapp_contentdislike`;

- збереження цифрового контенту знаходяться в таблиці `digsoapp_save`;

- перегляди цифрового контенту знаходяться в таблиці `digsoapp_look`.

Підписки на цифровий контент можуть оформлюватися клієнтами двома способами:

- як підписка на загальний цифровий контент, доступний за підпискою, тобто від різних авторів (загальні за сервісом та контент з авторських каналів, автори яких дозволили доступ всім клієнтам з цим загальним варіантом підписки);

- як підписка на цифровий контент від конкретного автора, тобто на його авторський канал: такий варіант підписки визначає сплату або за перегляд певного набору цифрового контенту (окремих матеріалів), або за перегляд цифрового контенту на вказаний період (у такому випадку, якщо строк дії підписки вийшов, то доступ до матеріалів не доступний, а у першому випадку доступ продовжується, але тільки до заданої кількості матеріалів, які за цей час вийшли).

Перший варіант підписки зберігається в таблиці `digsoapp_subscription`, визначаючи для конкретного клієнта, який такий варіант підписки оформив, фактично поточний стан цієї підписки. Фіксація сплати за таку підписку знаходиться в таблиці `digsoapp_action`. Друга таблиця необхідна для того, щоб зберегти всі випадки сплати за цю підписку. Перша таблиця необхідна,

щоб зберегти інформацію про те, в кого є не даний момент доступ до цифрового контенту на основі цього варіанта підписки.

Цей варіант підписки базується на пропозиціях сервісу, які визначають тривалість та вартість такої підписки. Варіанти цієї загальної підписки знаходяться в таблиці `digsoapp_product`.

Другий варіант підписки зберігається в таблиці БД `digsoapp_timerpayment`, що визначає конкретний факт сплати за доступ до авторського каналу на певний строк, і таблиці БД `digsoapp_contentpayment`, що визначає конкретний факт сплати за доступ до заданої кількості цифрового контенту авторського каналу, опублікованого з моменту сплати за доступ.

Для підтримки другого варіанта спочатку за авторським каналом потрібно визначити варіанти доступних підписок. Оскільки є два варіанти сплат за підписку, то вони виконуються на основі двох варіантів доступних підписок:

- у таблиці `digsoapp_authortimepayment` представлені доступні варіанти підписки на платний цифровий контент на певний період: у такому варіанті вказується кількість місяців, протягом якої буде доступ до контенту авторського каналу;

- у таблиці `digsoapp_authorcontentpayment` представлені доступні варіанти підписки на задану кількість публікацій цифрового контенту: тоді такий клієнт зможе бачити весь раніше опублікований контент авторського каналу та задану кількість нових публікацій цифрового контенту (після останньої такої публікації буде доступ тільки до цих публікацій, які входять у задану в підписці кількість).

Однак, ці варіанти визначаються для авторського каналу тоді, коли обрано саме такий варіант підтримки авторського каналу. Авторський канал може містити взагалі тільки безкоштовний цифровий контент. В іншому випадку це може бути сплата за доступ до всього сервісу. Але тоді не

надається доступ до тих авторських каналів, які для себе визначили таку персональну сплату.

Для варіанта персональної підписки на авторський канал передбачено також окрім перегляду цифрового контенту можливість поставити запитання авторам та отримати відповідь на запитання. Самі запитання, які розпочинають відповідне спілкування, знаходяться в таблиці `digsoapp_conversation`. Далі на запитання слідує відповідь від представників каналу, а тоді клієнт може знову ставити запитання, уточнювати, писати якесь повідомлення. Таке подальше спілкування знаходиться в таблиці `digsoapp_personal`.

Оскільки сам цифровий контент, авторські канали можуть викликати інтерес зокрема і учасниками, людьми, задіяними при виробництві цього контенту. Для цього потрібно визначити для окремого цифрового контенту або для всього каналу, хто саме став учасником окремого відео, епізоду, займається створенням авторського каналу. Самі учасники представлені відповідними своїми даними. Ці дані знаходяться в таблиці БД `digsoapp_member`.

Безпосередньо дані про участь цих учасників у створенні цифрового контенту (конкретного фактично матеріалу) знаходяться в таблиці `digsoapp_contentmember`.

Дані про загальних учасників у створенні всього авторського каналу представлені в таблиці `digsoapp_authorhoodmember`. Учасники беруть участь у створенні цифрового контенту або авторського каналу, визначаючи для них одну зі стандартних ролей. Ролі знаходяться в таблиці `digsoapp_role` і мають визначатися централізовано.

На рис. 3.4 представлено сформовану в результаті схему реляційної БД.

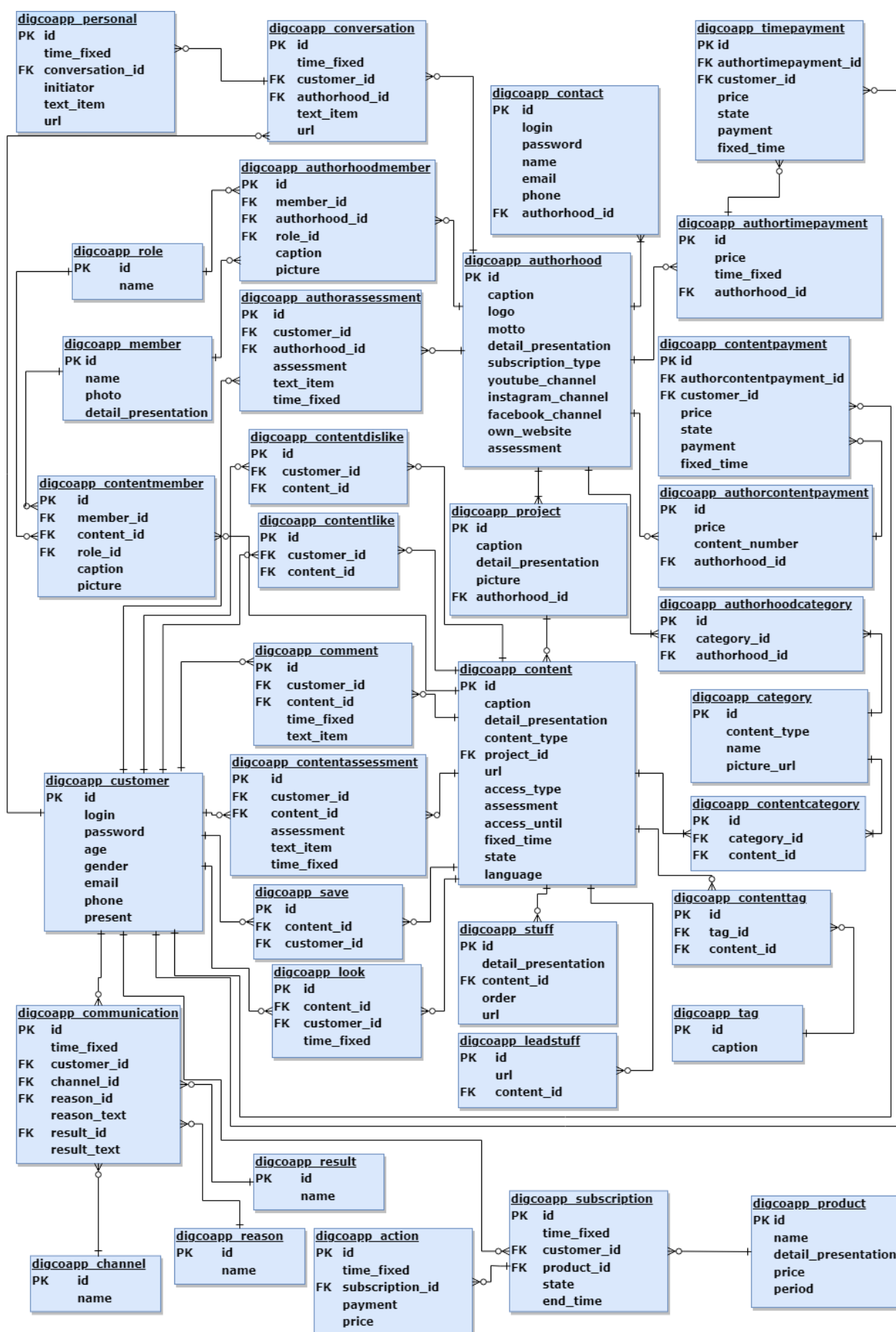


Рисунок 3.4 – Структура реляційної БД

Для клієнтів здійснюється також загальна технічна підтримка. Результати цієї підтримки на рівні відповідних звернень зберігаються в таблиці `digsoapp_communication`. Для внесення цих даних виділяються стандартні результати, що мають зберігатися в таблиці `digsoapp_result`, причини звернень, що мають зберігатися в таблиці `digsoapp_reason`, та канали комунікації, за якими було зафіксовано звернення, що мають зберігатися в таблиці `digsoapp_channel`.

Оскільки категорії відносяться до авторських каналів, а кожен авторський канал може відноситися до різних категорій, то зв'язок має тип багато-до-багатьох, то для його реалізації має бути створено таблицю `digsoapp_authorhoodcategory`. Аналогічно категорії можуть відноситися до цифрового контенту за тими ж самими правилами, тому має бути створено таблицю `digsoapp_contentcategory`.

Теги використовуються для цифрового контенту, тому кожен матеріал цифрового контенту може мати різні теги, а тег використовується не один раз, тому має бути створено таблицю `digsoapp_contenttag`.

Інші типи зв'язків між таблицями в БД представлені зв'язками один-до-багатьох, які або допускають можливість відсутності запису в таблиці з іншого боку, або ні. Більша частина зв'язків допускає, що з іншого боку дані можуть бути відсутні (наприклад, цифровий контент може не містити тегів, а на початку створення нового тегу відповідно жодного цифрового контенту за ним немає). Так авторський канал, як і цифровий контент, мають бути віднесені хоча би до однієї категорії обов'язково. Тому під час їхнього створення одразу має бути вказано відповідну категорію. Це стосується також зв'язку між авторським каналом і представником, який хоча би один має бути вказаний.

Для організації роботи з БД має бути використано систему PostgreSQL, враховуючи при цьому виборі її потужність, відкритість, відповідно і безкоштовність цієї системи, адже це відкрите програмне забезпечення, а також кросплатформність.

3.4 Висновки за розділом 3

У розділі 3 виконано роботи, пов'язані з проєктуванням програмного забезпечення, та представлено результати, охоплюючи вибір мови python, системи PostgreSQL, визначення функціональних вимог і моделювання роботи програмного забезпечення, проєктування структури БД.

4 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Порядок роботи програмного забезпечення

Основний порядок роботи програмного забезпечення визначається діями, які ініціює користувач, основою для яких є меню, склад його залежить від прав користувача. Якщо користувач є клієнтом, а таким розглядається будь-який базовий користувач, який про себе нічого не вказав, або користувач після стандартної авторизації клієнтського облікового запису, то йому дозволяється виконувати будь-які дії в межах представлених у розділі 3 сценаріїв для клієнта. Проте при виконанні частини цих дій відбувається перевірка прав доступу. Це пов'язано з тим, що наявності облікового запису клієнта недостатньо для того, щоб переглядати весь цифровий контент. Для деякого контенту необхідна відповідна підписка. Проте оскільки підписки існують різні, то перевірка прав доступу передбачає виконання деякої процедури (рис. 4.1).

Процедура розпочинається з отримання значень характеристик цифрового контенту. Серед цих характеристик є і інформація про права доступу. Далі ці права співставляються зі сторони контенту та клієнта.

Якщо доступ до об'єкта цифрового контенту безкоштовний, то він надається без перевірки прав зі сторони клієнта. Якщо ні, то перевіряється, чи не є авторський канал, який містить цей цифровий контент, загальним. Якщо це так, то перевіряється, чи має клієнт діючу загальну підписку. Якщо так, то надається доступ. Якщо ні, то доступ скасовується.

Якщо авторський канал не загальний, тобто він вимагає оформлення підписки, то перевіряється, чи має клієнт діючу підписку на деякий час або на деяку кількість цифрового контенту. Якщо така діюча підписка є, то надається доступ. Оскільки для швидкості доступу для кожної підписки визначається, чи вона є діючою, то при закінченні кожної доби запускається

скрипт, який визначає за кожною діючою підпискою, чи вона продовжує діяти.

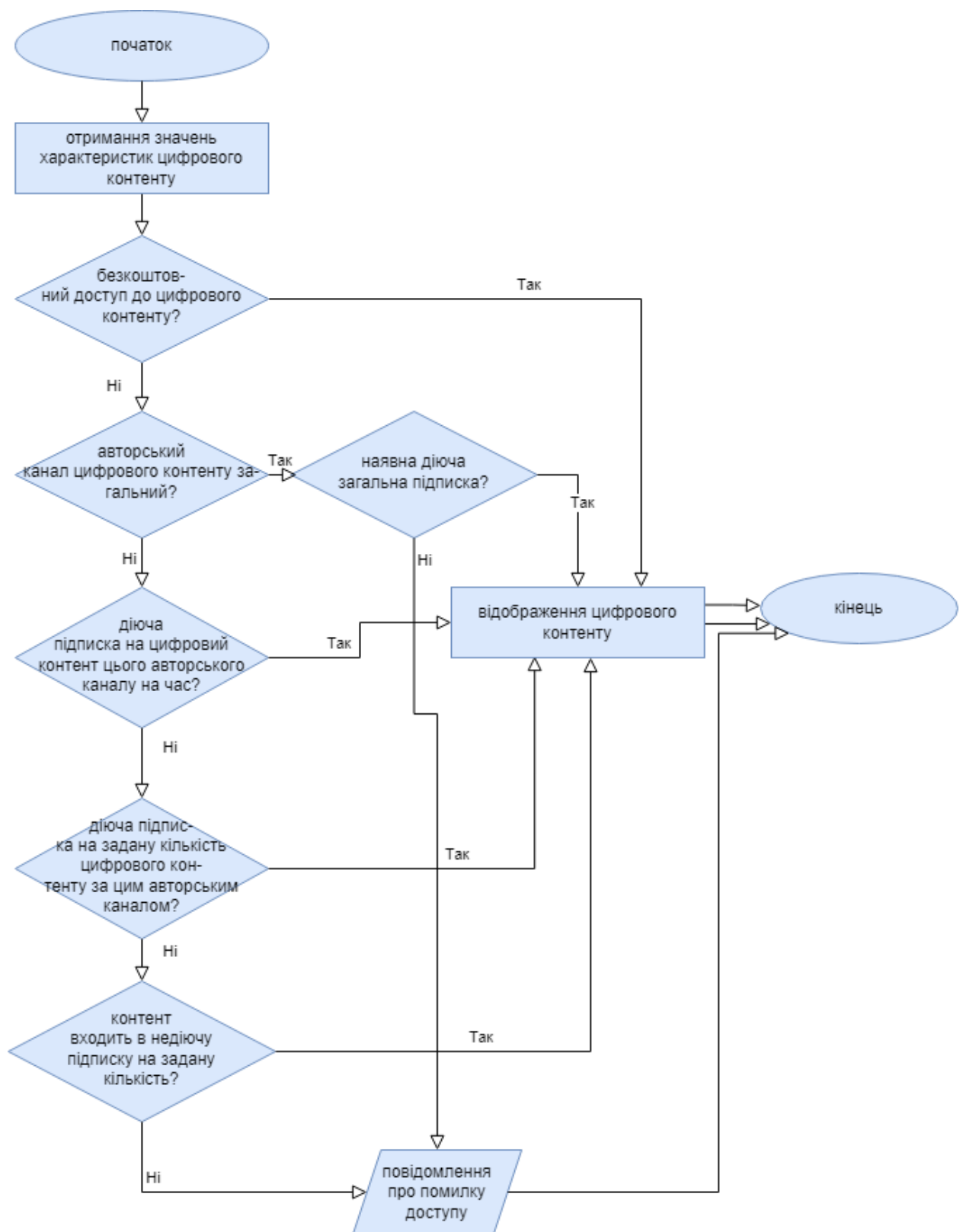


Рисунок 4.1 – Процедура перевірки прав доступу до цифрового контенту

Якщо діючих підписок немає, то перевіряється, чи має клієнт недіючу підписку на задану кількість цифрового контенту, а якщо така підписка на цей авторський канал є, то перевіряється, чи входить саме цей контент у цю кількість. Якщо так, то доступ надається. Тобто підписка на задану кількість продовжує діяти для цього контенту і після її завершення. Проте після її завершення новий цифровий контент вже за нею не переглянути.

4.2 Рішення з реалізації вебзастосунку

Реалізовано вебзастосунок за допомогою фреймворку Django, що визначило його стандартну структуру, які деталізовано і описано нижче.

Модель Authorhood реалізована як клас, що представляє авторський канал, на основі таких полів:

- id – ідентифікатор авторського каналу (тип BigAutoField);
- caption – назва авторського каналу (тип CharField);
- logo – посилання на зображення або логотип авторського каналу (тип URLField);
- motto – девіз авторського каналу (тип CharField);
- detail_presentation – детальний опис авторського каналу (тип TextField);
- subscription_type – тип підписки (тип CharField), який використовується для доступу до платних матеріалів каналу: загальний або персональний (тобто сплата вимагається за конкретним каналом);
- youtube_channel – посилання на канал youtube авторського каналу (тип URLField);
- instagram_channel – посилання на сторінку instagram авторського каналу (тип URLField);
- facebook_channel – посилання на сторінку facebook авторського каналу (тип URLField);

- own_website – посилання на власний вебсайт авторського каналу (тип URLField);
- assessment – середня оцінка, яку було поставлено авторському каналу (тип DecimalField);
- contents – категорії, до яких належить авторський канал (поле ManyToManyField).

Модель Contact реалізована як клас, що представляє користувачів, пов'язаних з роботою з авторським каналом (помічників), через наслідування класу User на основі перевизначення або визначення (де вказано тип) таких полів:

- id – ідентифікатор помічника;
- login – логін помічника;
- password – пароль помічника;
- name – ім'я помічника;
- email – електронна пошта помічника;
- phone – телефон помічника;
- authorhood – авторський канал, з яким може працювати цей помічник (тип ForeignKey).

Модель Authortimepayment реалізована як клас, що представляє підписки на платний цифровий контент авторського каналу на певний період, на основі таких полів:

- id – ідентифікатор підписки на платний цифровий контент авторського каналу на певний період (тип BigAutoField);
- authorhood – авторський канал, за яким доступна підписка (тип ForeignKey);
- price – вартість підписки (тип DecimalField);
- time_fixed – кількість місяців, протягом якої діє підписка (тип PositiveSmallIntegerField).

Модель `Authorcontentpayment` реалізована як клас, що представляє підписки на платний цифровий контент авторського каналу на задану кількість публікацій цифрового контенту, на основі таких полів:

- `id` – ідентифікатор підписки на платний цифровий контент авторського каналу на задану кількість публікацій цифрового контенту (тип `BigAutoField`);
- `authorhood` – авторський канал, за яким доступна підписка (тип `ForeignKey`);
- `price` – вартість підписки (тип `DecimalField`);
- `content_number` – кількість публікацій цифрового контенту, яка доступна за підпискою (тип `PositiveSmallIntegerField`).

Модель `Project` реалізована як клас, що представляє проекти цифрового контенту, на основі таких полів:

- `id` – ідентифікатор проекту цифрового контенту (тип `BigAutoField`);
- `authorhood` – авторський канал, за яким створено цей проект (тип `ForeignKey`);
- `caption` – назва проекту цифрового контенту (тип `CharField`);
- `detail_presentation` – детальний опис проекту цифрового контенту (тип `TextField`);
- `picture` – посилання на файл зображення, що відповідає проекту цифрового контенту (поле `URLField`).

Модель `Content` реалізована як клас, що представляє цифровий контент, на основі таких полів:

- `id` – ідентифікатор цифрового контенту (тип `BigAutoField`);
- `project` – проект, до якого належить цей цифровий контент (поле `ForeignKey`);
- `caption` – назва цифрового контенту (тип `CharField`);
- `detail_presentation` – детальний опис цифрового контенту (тип `TextField`);

- content_type – тип цифрового контенту, тобто відео, аудіо, текст, фотографії (тип CharField);
- url – адреса основного файлу цифрового контенту (тип URLField);
- access_type – тип доступу до цифрового контенту, тобто безкоштовний або потребує підписки (поле BooleanField);
- assessment – середня оцінка, яку було поставлено цифровому контенту (тип DecimalField);
- access_until – дата і час, до яких доступний цей цифровий контент (поле DateTimeField);
- state – стан цифрового контенту (поле PositiveSmallIntegerField): передпублікація, доступний, заархівований;
- language – мова цифрового контенту (поле CharField);
- categories – категорії, до яких належить цифровий контент (поле ManyToManyField);
- tags – теги, до яких належить цифровий контент (поле ManyToManyField).

Категорії авторських каналів та цифрового контенту створені моделлю Category на основі таких полів:

- id – ідентифікатор категорії (тип BigAutoField);
- content_type – тип цифрового контенту, тобто це відео, аудіо, текст чи фотографія (поле CharField);
- name – назва категорії (поле CharField);
- picture_url – адреса зображення, що ілюструє категорію (тип URLField).

Модель тегів Tag реалізована як клас на основі полів id як ідентифікатора тегу (поле BigAutoField) та його назви caption (поле CharField).

Модель Stuff реалізована як клас, що представляє матеріал до цифрового контенту, на основі таких полів:

- id – ідентифікатор матеріала до цифрового контенту (тип BigAutoField);
- content – цифровий контент, якого стосується матеріал (тип ForeignKey);
- detail_presentation – детальний опис матеріалу цифрового контенту (тип TextField);
- url – адреса файлу матеріалу цифрового контенту (тип URLField);
- order – порядковий номер відображення матеріалу серед всіх матеріалів цього цифрового контенту (тип PositiveSmallIntegerField).

Модель Leadstuff реалізована як клас, що представляє ілюстративний матеріал до цифрового контенту, на основі таких полів:

- id – ідентифікатор ілюстративного матеріала до цифрового контенту (тип BigAutoField);
- content – цифровий контент, якого стосується матеріал (тип ForeignKey);
- url – адреса файлу матеріалу цифрового контенту (тип URLField).

Модель Comment реалізована як клас, що представляє коментар на цифровий контент, на основі таких полів:

- id – ідентифікатор коментаря на цифровий контент (тип BigAutoField);
- content – цифровий контент, якого стосується коментар (тип ForeignKey);
- customer – клієнт, який залишив цей коментар до цифрового контенту (тип ForeignKey);
- time_fixed – час створення коментаря (поле DateTimeField);
- text_item – текст коментаря (поле CharField).

Модель Authorassessment реалізована як клас, що представляє оцінку, виставлену авторському каналу, на основі таких полів:

- id – ідентифікатор оцінки, виставленої авторському каналу (тип BigAutoField);

- authorhood – авторський канал, якому виставлено оцінку (тип ForeignKey);
- customer – клієнт, який виставив цю оцінку авторському каналу (тип ForeignKey);
- assessment – виставлена оцінка (тип PositiveSmallIntegerField);
- text_item – текст, доданий до оцінки (тип CharField);
- time_fixed – час створення оцінки (поле DateTimeField).

Модель Contentassessment реалізована як клас, що представляє оцінку, виставлену цифровому контенту, на основі таких полів:

- id – ідентифікатор оцінки, виставленої цифровому контенту (тип BigAutoField);
- content – цифровий контент, якому виставлена оцінка (тип ForeignKey);
- customer – клієнт, який виставив цю оцінку цифровому контенту (тип ForeignKey);
- assessment – виставлена оцінка (тип PositiveSmallIntegerField);
- text_item – текст, доданий до оцінки (тип CharField);
- time_fixed – час створення оцінки (поле DateTimeField).

Модель Contentlike реалізована як клас, що представляє уподобання, виставлене цифровому контенту, на основі таких полів:

- id – ідентифікатор уподобання, виставленого цифровому контенту (тип BigAutoField);
- content – цифровий контент, якому виставлене уподобання (тип ForeignKey);
- customer – клієнт, який уподобав цей цифровий контент (тип ForeignKey).

Модель Contentdislike реалізована як клас, що представляє антипатію, виставлену цифровому контенту, на основі таких полів:

- id – ідентифікатор антипатії, виставленої цифровому контенту (тип BigAutoField);

- content – цифровий контент, якому виставлена антипатія (тип ForeignKey);

- customer – клієнт, який визначив антипатію для цього цифрового контенту (тип ForeignKey).

Модель Authorhoodmember реалізована як клас, що представляє загальних учасників у створенні авторського каналу, на основі таких полів:

- id – ідентифікатор учасника в створенні авторського каналу (тип BigAutoField);

- authorhood – авторський канал, у створенні якого приймав участь учасник (тип ForeignKey);

- member – учасник, щодо якого визначено його участь у створенні авторського каналу (тип ForeignKey);

- role – роль, яку на даному авторському каналі виконував учасник (тип ForeignKey);

- caption – назва конкретної ролі (тип CharField);

- picture – адреса приєднаного зображення учасника (тип URLField).

Модель Contentmember реалізована як клас, що представляє учасника у створенні цифрового контенту, на основі таких полів:

- id – ідентифікатор учасника в створенні цифрового контенту (тип BigAutoField);

- content – цифровий контент, у створенні якого приймав участь учасник (тип ForeignKey);

- member – учасник, щодо якого визначено його участь у створенні цифрового контенту (тип ForeignKey);

- role – роль, яку у створенні даного цифрового контенту виконував учасник (тип ForeignKey);

- caption – назва конкретної ролі (тип CharField);

- picture – адреса приєднаного зображення учасника (тип URLField).

Безпосередньо ролі створені на основі класу моделі Role за допомогою полів id як ідентифікатора тегу (поле BigAutoField) та його назви name (поле CharField).

Модель Member реалізована як клас, що представляє учасника, який може брати участь у створенні авторського каналу або окремого цифрового контенту, на основі таких полів:

- id – ідентифікатор учасника в створенні цифрового контенту (тип BigAutoField);
- photo – адреса зображення учасника (тип URLField);
- name – ім'я учасника (тип CharField);
- detail_presentation – детальний опис учасника (тип TextField).

Модель Save реалізована як клас, що представляє збереження цифрового контенту, на основі таких полів:

- id – ідентифікатор збереження цифрового контенту (тип BigAutoField);
- content – цифровий контент, який було збережено (тип ForeignKey);
- customer – клієнт, який зберіг цифровий контент собі (тип ForeignKey).

Модель Look реалізована як клас, що представляє перегляд цифрового контенту, на основі таких полів:

- id – ідентифікатор перегляду цифрового контенту (тип BigAutoField);
- content – цифровий контент, який було переглянуто (тип ForeignKey);
- customer – клієнт, який переглянув цифровий контент (тип ForeignKey);
- time_fixed – час перегляду цифрового контенту (поле DateTimeField).

Модель Customer реалізована як клас, що представляє клієнтів через наслідування класу User на основі перевизначення або визначення (де вказано тип) таких полів:

- id – ідентифікатор клієнта;
- login – логін клієнта;
- password – пароль клієнта;
- age – вік клієнта (поле PositiveIntegerField);
- gender – стать клієнта (поле BooleanField);
- email – електронна пошта клієнта;
- phone – телефон клієнта;
- present – опис клієнтом себе (тип CharField).

Канали комунікації створені на основі класу моделі Channel за допомогою полів id як ідентифікатора каналу комунікації (поле BigAutoField) та його назви name (поле CharField).

Стандартні результати технічної підтримки створені на основі класу моделі Result за допомогою полів id як ідентифікатора результату (поле BigAutoField) та його назви name (поле CharField).

Причини звернень до технічної підтримки створені на основі класу моделі Reason за допомогою полів id як ідентифікатора результату (поле BigAutoField) та його назви name (поле CharField).

Модель Communication реалізована як клас, що представляє звернення до технічної підтримки, на основі таких полів:

- id – ідентифікатор звернення до технічної підтримки (тип BigAutoField);
- time_fixed – час, коли відбулось звернення (поле DateTimeField);
- customer – клієнт, який виконав звернення (поле ForeignKey);
- channel – канал комунікації, за яким відбулось звернення (поле ForeignKey);
- reason – причина звернення (поле ForeignKey);
- reason_text – повний текст, що описує причину звернення;

- result – результат звернення (поле ForeignKey);
- result_text – повний текст, що описує результат опрацювання звернення.

Модель Conversation реалізована як клас, що представляє спілкування між представниками авторського каналу та клієнтом, на основі таких полів:

- id – ідентифікатор спілкування (тип BigAutoField);
- authorhood – авторський канал, представники якого беруть участь у спілкуванні (поле ForeignKey);
- customer – клієнт, який розпочав спілкування (поле ForeignKey);
- time_fixed – час, коли розпочалось спілкування звернення (поле DateTimeField);
- text_item – повний текст запитання (поле CharField);
- url – адреса файлу, який був доданий до запитання (тип URLField).

Модель Personal реалізована як клас, що представляє подальші повідомлення у створеному спілкуванні, на основі таких полів:

- id – ідентифікатор повідомлення спілкування (тип BigAutoField);
- conversation – спілкування, до якого належить повідомлення (поле ForeignKey);
- initiator – сторона, що створила повідомлення (клієнт, який створив саме спілкування, або представник авторського каналу), що представляє поле BooleanField;
- time_fixed – час, коли було створено повідомлення (поле DateTimeField);
- text_item – повний текст повідомлення (поле CharField);
- url – адреса файлу, який був доданий до повідомлення (тип URLField).

Модель Subscription реалізована як клас, що представляє загальну підписку клієнта, на основі таких полів:

- id – ідентифікатор загальної підписки клієнта (тип BigAutoField);
- customer – клієнт, який оформив цю підписку (поле ForeignKey);

- product – варіант загальної підписки, яку оформив клієнт (поле ForeignKey);
- time_fixed – час, коли було оформлено цю підписку (поле DateTimeField);
- state – поточний стан підписки (поле BooleanField);
- end_time – час, коли було закінчено дію підписки або планується закінчити, якщо вона ще не закінчилась (поле DateTimeField).

Модель Action реалізована як клас, що представляє сплату за загальну підписку, на основі таких полів:

- id – ідентифікатор сплати за загальну підписку клієнта (тип BigAutoField);
- subscription – загальна підписка клієнта, за яку було виконано сплату (поле ForeignKey);
- payment – номер платіжу в PayPal (поле CharField);
- time_fixed – час, коли було сплачено за підписку (поле DateTimeField);
- price – сума, яку було сплачено (поле DecimalField).

Модель Product реалізована як клас, що представляє варіант загальної підписки на сервіс, на основі таких полів:

- id – ідентифікатор варіанта загальної підписки на сервіс (тип BigAutoField);
- name – назва варіанта загальної підписки на сервіс (поле CharField);
- detail_presentation – детальний опис варіанта загальної підписки на сервіс (тип TextField);
- price – вартість підписки (поле DecimalField);
- period – кількість місяців, протягом якої діє підписка (поле PositiveSmallIntegerField).

Модель Timerpayment реалізована як клас, що представляє конкретну сплату за підписку клієнта на авторський канал на певний період, на основі таких полів:

- id – ідентифікатор сплати за підписку клієнта на авторський канал на певний період (тип BigAutoField);
- authortimepayment – підписка на платний цифровий контент авторського каналу на певний період, за якою було виконано сплату (поле ForeignKey);
- customer – клієнт, який оформив підписку (поле ForeignKey);
- state – поточний стан підписки (поле BooleanField);
- price – сплачена сума за підписку (поле DecimalField);
- payment – номер платіжу в PayPal (поле CharField);
- fixed_time – час, коли було оформлено підписку (поле DateTimeField).

Модель Contentpayment реалізована як клас, що представляє конкретний факт сплати за доступ до заданої кількості цифрового контенту авторського каналу, на основі таких полів:

- id – ідентифікатор сплати за доступ до заданої кількості цифрового контенту авторського каналу (тип BigAutoField);
- authorcontentpayment – варіант підписки на задану кількість публікацій цифрового контенту, за який було сплачено (поле ForeignKey);
- customer – клієнт, який оформив підписку (поле ForeignKey);
- state – поточний стан підписки (поле BooleanField);
- price – сплачена сума за підписку (поле DecimalField);
- payment – номер платіжу в PayPal (поле CharField);
- fixed_time – час, коли було оформлено підписку (поле DateTimeField).

Для кожного сценарію, визначеного на діаграмах прецедентів у розділі 3, було визначено представлення для витягання даних і формування їх у необхідному вигляді з заповненням відповідного шаблону цими даними. Шаблони сторінок створені за допомогою шаблону [8], який є вільним і безкоштовним. Лише деякі з цих сценаріїв не призвели до створення окремих представлень. Наприклад, реалізацію коментування цифрового контенту

було об'єднано зі сценарієм перегляду створених коментарів. Це пояснюється тим, що користувачу зручно залишати коментар на тій же сторінці, де ці коментарі відображаються, тому витягання відповідних коментарів було об'єднано з формою створення коментаря. Реалізовано це було в представленні `digital_content_look`, яке відповідає за відтворення цифрового контенту. Сам цифровий контент визначається параметром `digital_content` відповідної функції. Цей параметр має визначати ідентифікатор цифрового контенту, який передається через посилання на сторінку цифрового контенту. Цей номер виділяється при обробці посилання в менеджері, що визначений у файлі `urls.py`, а тоді в даній функції витягають всі дані за цим цифровим контентом, включаючи теги, категорії, коментарі, уподобання, антипатії, дані проєкту, до його віднесено цифровий контент, дані про авторський канал, дані про учасників створення цифрового контенту. Далі виконується перевірка прав доступу для клієнта на основі процедури перевірки прав доступу до цифрового контенту (рис. 4.1). Якщо права наявні, то вся ця інформація про цифровий контент передається в шаблон сторінки. Якщо права відсутні, то дані взагалі не передаються. У випадку, якщо права наявні, то визначається, чи зберіг авторизований клієнт цей цифровий контент. Відповідна позначка передається в шаблон (збережено чи ні). На останньому кроці формується форма створення коментаря. Форма створення коментаря реалізована через клас `CommentForm`, який створений на основі моделі коментаря. На формі представлена відповідна позиція для введення тексту. У той же час авторство коментаря, дані про цифровий контент, до якого відноситься коментар, вносяться програмно.

Для зручності реалізації та звернень усі представлення розділені на блоки:

- `digital_content_views.py` – представлення для підтримки роботи клієнта, які стосуються самого цифрового контенту, авторських каналів, тобто фактично цифрового контенту і безпосередньо пов'язаних даних зі

структуризацією цього контенту та діями користувача (коментарі, уподобання, антипатії, збереження), що представлено в додатку Б;

- `subscription_views.py` – представлення, які стосуються роботи клієнта та відповідальних за ведення каналів стосовно підписок на цифровий контент, включаючи внесення варіантів підписок, відображення їх, оформлення підписок, в тому числі сплату за підписку, що виконується за допомогою впровадження платіжної системи PayPal, що було реалізовано за допомогою модуля `django-paypal` [9]-[10];

- `data_editor_views.py` – представлення, які використовуються відповідальними за ведення авторських та загальних каналів, надаючи для цього засоби додавання, редагування, вилучення відповідних даних.

Для простоти підтримки остання група представлень реалізована на основі використання класів `CreateView`, `UpdateView`, `DeleteView`, що дозволило значно скоротити процес написання необхідного коду.

Представлення підтримки користувацького пошуку реалізовані 3 окремими функціями:

- пошук авторських каналів реалізовано функцією `authorhood_search`, у яку самі параметри пошуку передаються через параметри GET-запитів, звідки витягаються назва каналу (`caption`), потрібний тип підписки (`subscription`), якщо це було встановлено, мінімальна прийнятна середня оцінка (`assessment1`), максимальна прийнятна середня оцінка (`assessment2`), перелік категорій авторського каналу, вказаних через кому (`categories`), перелік учасників, що приймають участь у створенні авторського каналу (`members`), а отримані відповідні результати відсортовуються за оцінкою;

- пошук цифрового контенту реалізовано функцією `digital_content_search`, особливістю якої є те, що пошук здійснюється з визначенням прав доступу до цифрового контенту, тому в підсумку повертаються в шаблон два переліки, що включають той цифровий контент, до якого має доступ користувач (`digital_content_results`) та той, до якого користувач доступ не має (`digital_content_unavailable`);

– пошук проєктів цифрового контенту реалізовано функцією `content_project_search`, у яку самі параметри пошуку передаються через параметри GET-запитів, звідки витягаються назва каналу (`caption`), мінімальна прийнятна середня оцінка цифрового каналу, на якому було створено проєкт, (`assessment1`), максимальна прийнятна середня оцінка цифрового каналу, на якому було створено проєкт, (`assessment2`).

Збереження цифрового контенту виконується за допомогою представлення `DigitalContentSaveView`. Його особливістю є те, що це представлення діє без шаблону сторінки. Тобто в підсумку не повертається відповідна сторінка, а тільки виконуються необхідні дії, а далі відбувається інформування про ці дії. Цим зокрема зумовлена і реалізація цього класу за допомогою наслідування класів `LoginRequiredMixin`, що необхідно для забезпечення авторизації, та `View`. Відповідно неавторизований користувач не зможе виконати цю дію. Однак особливістю є також і те, що може виявитися, що користувач вже раніше зберіг цей цифровий контент. Ідентифікатор цифрового контенту передається в метод `post`, який підтримує основну роботу, через параметр `pk`. При цьому спочатку перевіряється, чи не було раніше створено таке збереження за допомогою об'єкту класу `Save`. Якщо ні, то зберігається цей цифровий контент для авторизованого користувача. Якщо збереження відбулось, то воно вилучається.

Подібно влаштовано і вподобання цифрового контенту. Для цього створено клас `DigitalContentLikeView`. Цей клас діє подібно, але має свої особливості. Особливості пов'язані з тим, що уподобання не просто має бути вилучено, якщо воно вже було зроблено, як це відбувалось зі збереженням. Справа в тому, що окрім уподобання ще існують антипатії. І одночасно існувати щодо одного й того самого контенту і користувача і антипатія, і уподобання не можуть. Тому якщо виявляється, що існує антипатія, то спочатку вона має бути вилучена, далі перевірено, що такого уподобання не існує, а тоді уподобання збережено. Якщо антипатія існує, вона вилучається,

а далі виконується спроба зберегти уподобання. Якщо уподобання існує, воно вилучається. Якщо ні, то створюється.

У цьому розділі створені класи і функції, які реалізують представлення, описані за принципом внесення додаткових особливостей і рішень. Тобто враховуючи кількість створених одиниць, було вирішено описати основні особливості деяких представлень, а не перелічувати взагалі всі існуючі. Це зокрема пояснюється і тим, що ці представлення автоматично підтримують роботу користувача на основі описаних у розділі 3 сценаріїв. Тобто не передбачається їхній виклик як окремих функцій. Вони викликаються для оброблення відповідного сценарію. А обробка реалізується за допомогою витягання даних і внесення їх у шаблон, щоб відобразити результуючу сторінку користувачу.

Для реалізації методу прогнозування підписок на цифровий контент застосовано пакети Tensorflow [11], який забезпечує бекенд для роботи з машинним навчанням, Keras [12], який забезпечує засоби безпосередньо створення моделей прогнозування, та Pandas [13], який необхідний для читання даних, що створюють вибірку даних, застосовану для створення моделей, оброблення цих даних. Процес оброблення даних, створення, навчання моделей було винесено у файл `subscription_forecast.py`, де виділено відповідні функції.

Оскільки метод прогнозування підписок на цифровий контент в основі використовує модель на основі LSTM, то серед основних використаних для реалізації моделі засобів було застосовано клас LSTM з підпаketу `layers` пакету `keras`. З цього ж підпаketу було використано клас `Dense` для забезпечення повнозв'язного шару. Клас LSTM потребує під час використання конструктору завдання кількості чарунок в LSTM-шарі, визначення розмірності, що складається зокрема з кількості вхідних ознак та тривалості часового інтервалу. Для навчання використовується метод `fit` створеної моделі, у якому в параметрах задається кількість ітерацій (використовувалось 100), дані для проведення навчання.

4.3 Висновки за розділом 4

У розділі 4 виконано роботи, пов'язані з забезпеченням реалізації програмного забезпечення, та представлено результати, охоплюючи перевірку прав доступу до цифрового контенту, реалізовані класи в складі вебзастосунку Django.

5 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

5.1 Опис програми

Розроблена програма призначається для забезпечення роботи сервісів надання доступу до цифрового контенту централізованим чином та на основі створення окремих авторських каналів з можливістю надання цифрового контенту, забезпечення платних підписок на цей контент та контролю за станом підписок.

Фактичні вимоги до технічних засобів мають враховувати кількість користувачів, підтримка роботи з якими має забезпечуватися. Як тільки кількість користувачів збільшується, вимоги мають зростати. Мінімальні вимоги, які визначені нижче, враховують можливість одночасної роботи з сервером 25 користувачів. У розрахунках враховано, що очікувана кількість користувачів вебзастосунку – 100. За такої кількості максимально 25 з них можуть звернутися до вебзастосунку одночасно. Кількість авторських каналів – 10. На кожний канал має бути виділено мінімум 10 Гб для збереження даних. Кожен користувач повинен мати 100 Мб для збереження вкладень до повідомлень. Тобто сумарно необхідно 110 Гб та 5 Гб на роботу БД. Самі вимоги до організації сервера також включають мінімально процесор Intel Xeon E5440, який має частоту 2,83 ГГц, та 4 Гб доступної оперативної пам'яті.

5.2 Експлуатація програми

Після звернення до вебзастосунку відкривається сторінка перегляду найкращого цифрового контенту (рис. 5.1). Усі користувачі спочатку є клієнтами. Якщо користувач неавторизований, то при спробі доступу до сторінки, де користувач має бути авторизованим, його перенаправляють на сторінку авторизації. З цієї сторінки (до неї також можна отримати доступ з

пункту Авторизація розділу Обслуговування) можна перейти до авторизації автора каналу.

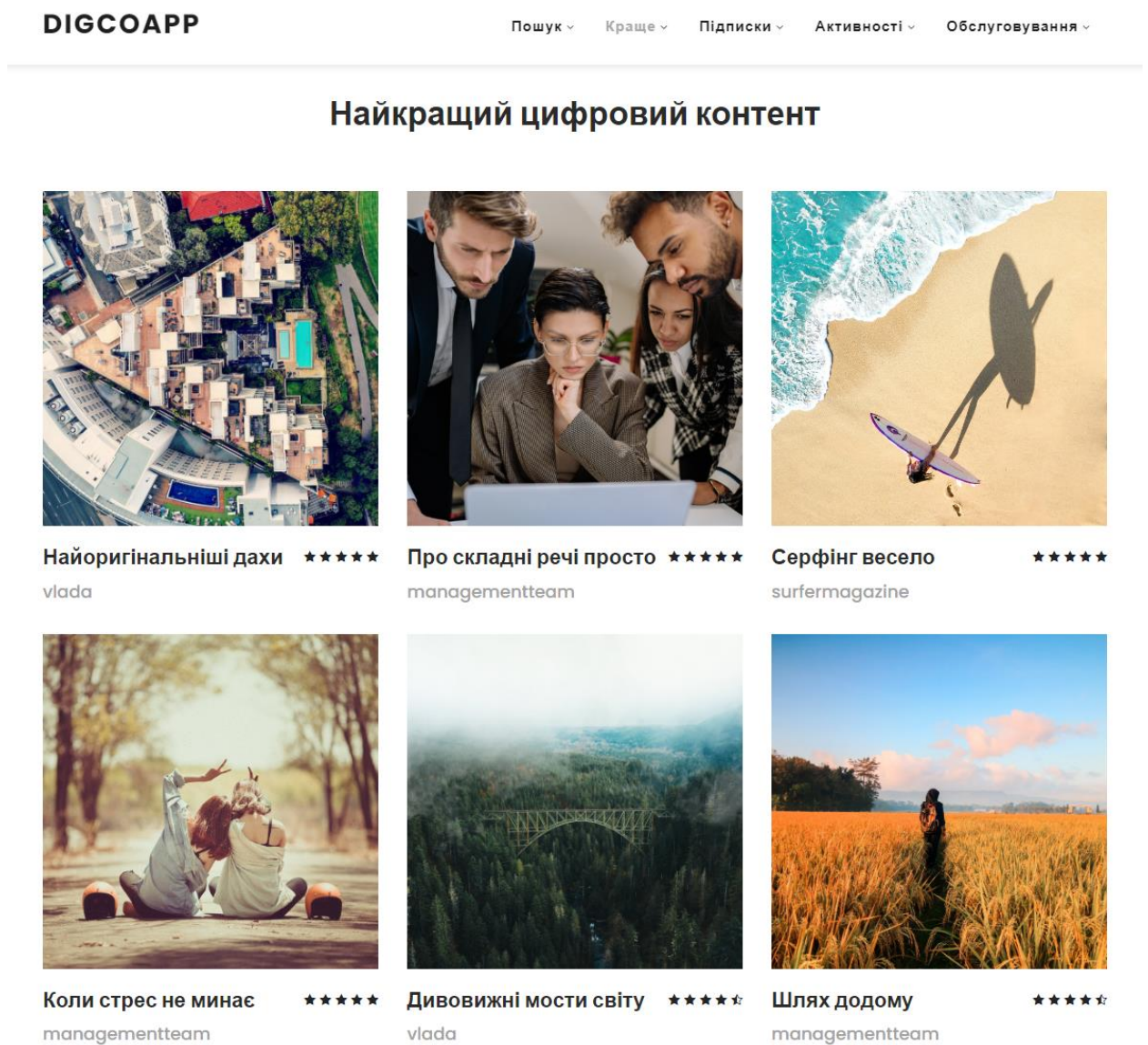


Рисунок 5.1 – Відображення найкращого цифрового контенту в програмі

Усі розділи меню клієнта розділена на Пошук (пункти Канали, Проекти, Контент для пошуку відповідних даних), Краще (пункти Канали та Контент для перегляду найкращих каналів та цифрового контенту), Підписки, Активності (пункти Збережене, Уподобане, Антипатії, Коментарі, Оцінки, Історія для перегляду відповідних результатів зі збереженим цифровим контентом, уподобаним, контентом, якому виставлено антипатію,

переліком коментарів, які створив авторизований користувач, виставленими оцінками та переглянутим цифровим контентом) та Обслуговування.

Зі сторінок найкращих авторських каналів, пошуку авторських каналів можна перейти до сторінки авторського каналу (рис. 5.2).

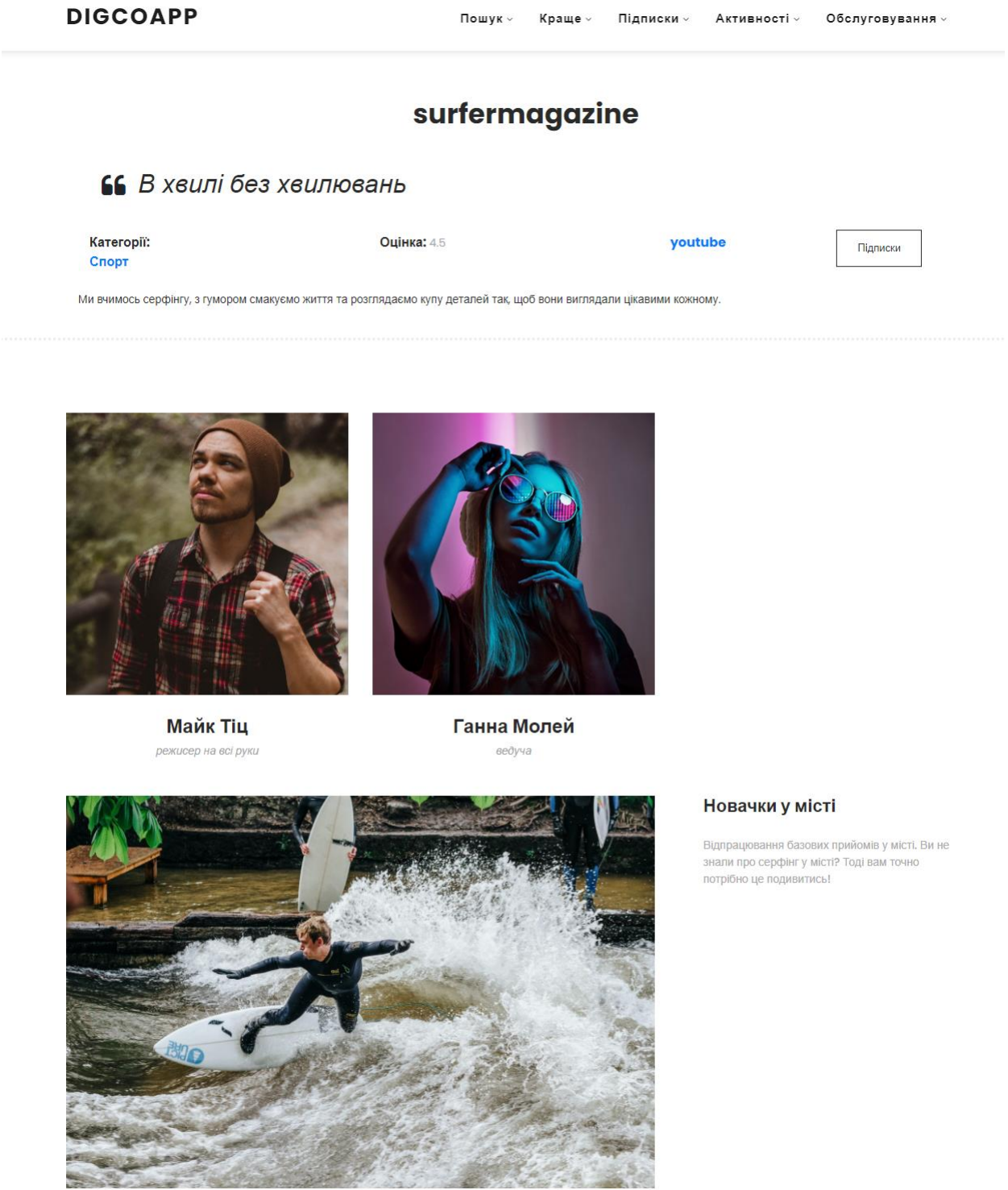


Рисунок 5.2 – Перегляд сторінки авторського каналу

На сторінці авторського каналу згори вказується назва каналу, його девіз біля цитати, перелік категорій, до яких належить канал (натискаючи на назву, можна перейти до переліку авторських каналів за категорією), середня оцінка, яку було виставлено клієнтами, які підписались на канал раніше, посилання на соціальні мережі, опис авторського каналу. Кнопка Підписки дозволяє перейти до переліку підписок, які можна оформити на канал (рис. 5.3). Якщо натиснути на один з варіантів підписки, то відкриється сторінка для сплати за підписку через PayPal.

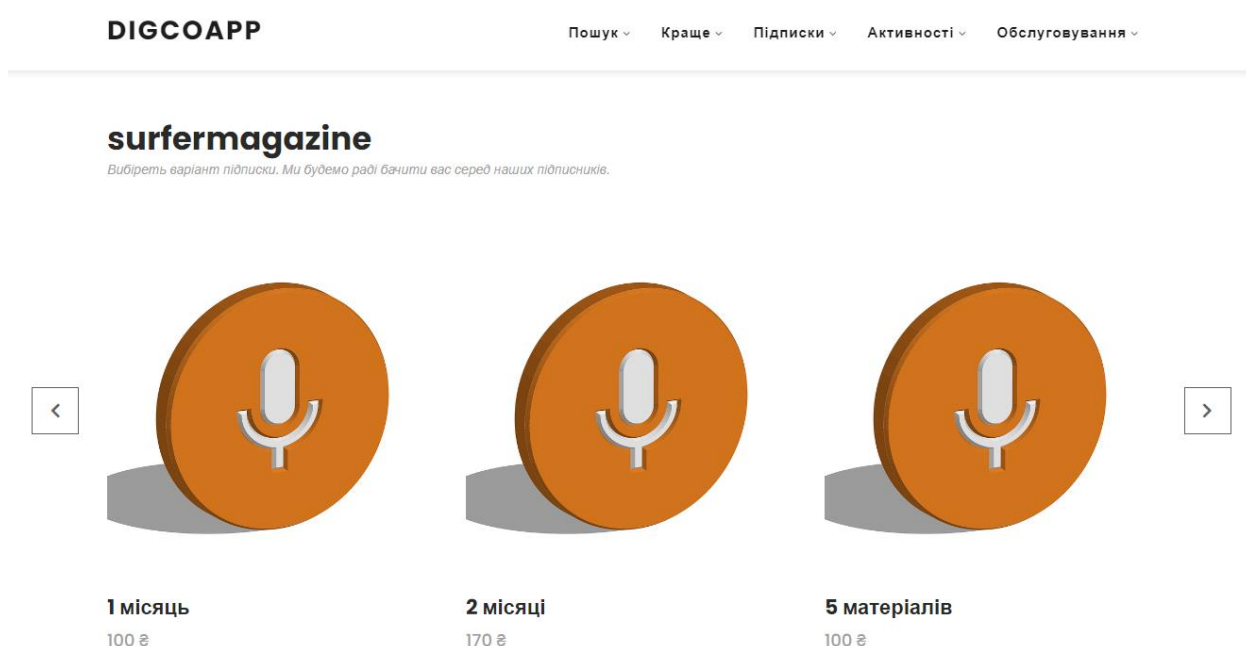


Рисунок 5.3 – Варіанти підписок на авторський канал

На сторінці авторського каналу також знаходиться перелік учасників створення даного авторського каналу. При натисканні на оцінку можна перейти до переліку всіх оцінок авторського каналу. Посилання оцінити доступно тоді, коли користувач підписався на канал. Якщо він підписався, то кнопка Підписки не доступна.

Нижче на сторінці знаходиться перелік проєктів. При натисканні на фотографію або назву проєкту відкривається сторінка перегляду цифрового контенту за проєктом, а зі сторінки проєкту можна перейти сторінки

конкретного контенту (рис. 5.4). На тій сторінці знаходиться назва контенту, назва проєкту, до якого належить контент (біля цитати), назва авторського каналу (під назвою проєкту), та сам основний файл контенту (відео, фотографія, аудіо або текст).

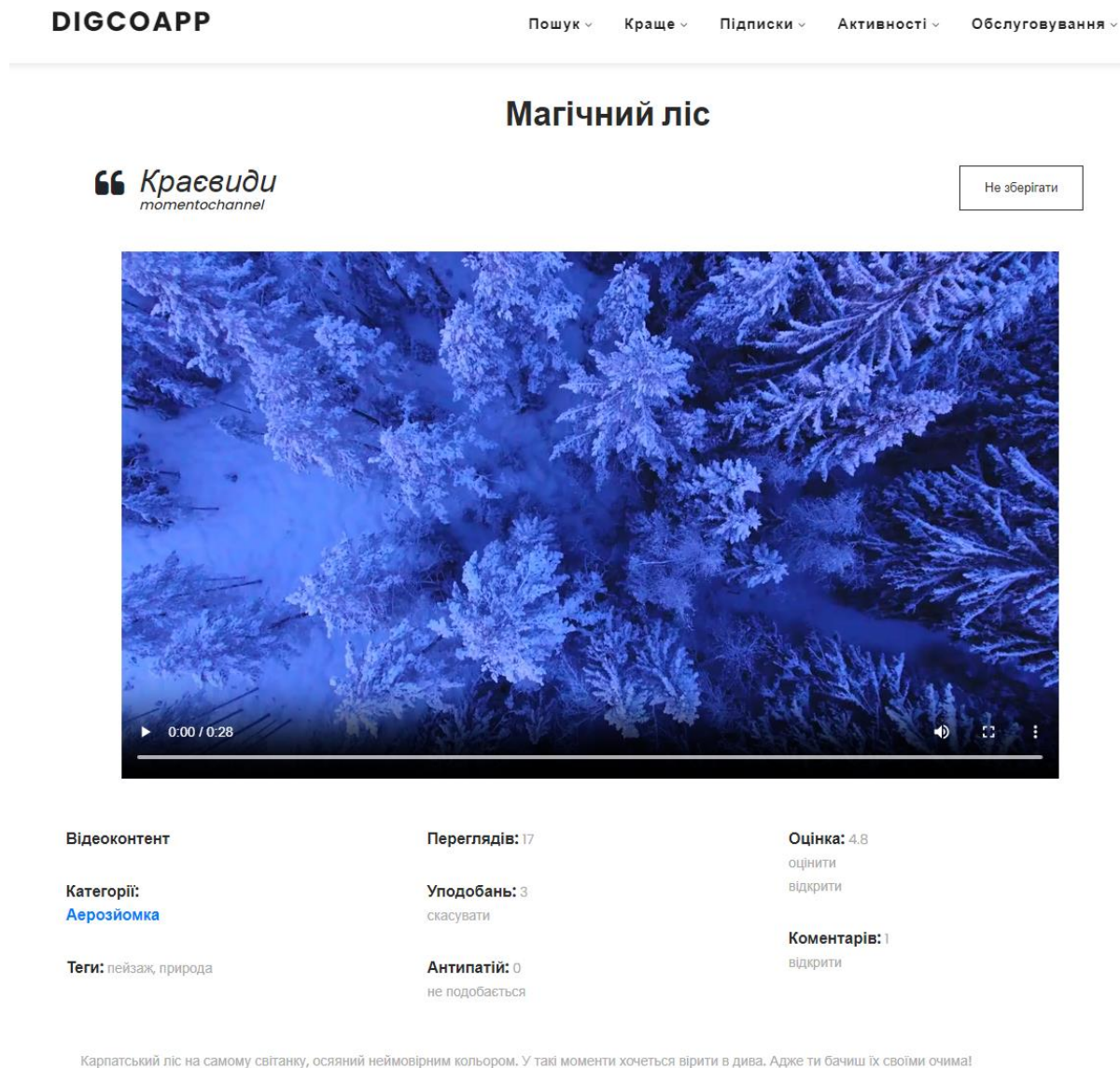


Рисунок 5.4 – Перегляд цифрового контенту

Під назвою контенту нижче знаходиться кнопка Зберегти для збереження контенту або Не зберігати, якщо контент був раніше збережений.

Під самим контентом знаходиться інформація про тип основного контенту, про кількість переглядів, про виставлену середню оцінку (посилання оцінити дозволяє перейти до створення оцінки, а посилання

відкрити дозволяє відкрити перелік всіх оцінок цифрового контенту), про кількість уподобань (посилання скасувати дозволяє скасувати уподобання, а посилання уподобати – відповідно його створити), кількість антипатій (посилання скасувати дозволяє скасувати антипатію, а посилання не подобається – відповідно його створити), перелік категорій (за назвою можна перейти до сторінки з переліком цифрового контенту за категорією), перелік тегів (за назвою можна перейти до сторінки з переліком цифрового контенту за тегом), кількість коментарів (посилання відкрити дозволяє перейти вниз сторінки, де містяться коментарі та форма для створення нового коментаря). Нижче знаходиться основний текстовий підпис до цифрового контенту та перелік матеріалів, які було додано до контенту.

Розділ меню Підписки містить пункти Загальні з варіантами загальних підписок та інформацією про виконані підписки, Індивідуальні з варіантами оформлених раніше підписок, Канали з переліком авторських каналів, на які діє підписка, Контент з переліком цифрового контенту, на який було оформлено підписку (кнопки дозволяють перейти між контентом загальних підписок та підписок на авторські канали).

Розділ Обслуговування містить пункти Спілкування для перегляду спілкувань з представниками авторських каналів, Підтримка для перегляду повідомлень з представниками загального каналу (менеджерами), Профіль, Вихід або Авторизація.

Якщо в програмі авторизувався відповідальний за ведення каналу, то йому доступний розділ меню Канал, що включає пункти Створити (якщо користувач вже створив канал або є відповідальним за його ведення, то відповідна форма не буде створена), Динаміка, Проєкти, Відповідальні, Спілкування. Перший пункт дозволяє створити авторський канал. Всі наступні пункти стосуються конкретного авторського каналу, відповідальним за ведення якого є авторизований користувач. Зі сторінки проєктів можна додати новий проєкт, редагувати існуючий, додати новий цифровий контент до проєкту (рис. 5.5).

managementteam

Успішне керування людьми

Відеоконтент ▾

Безкоштовний доступ ▾

Стан

Доступ до

Публікація ▾

00.мм.rrrr

Мова

Файл

українська ▾

Додати

Опублікувати

Матеріал

Внесіть додаткові матеріали або обкладинку

Порядок

Файл

1 ▾

Додати

☐ обкладинка

Додати

Рисунок 5.5 – Додавання цифрового контенту

Під час додавання нового цифрового контенту потрібно задати назву цього контенту, обрати тип основного контенту, варіант доступу (безкоштовний, загальна підписка або підписка на канал), стан контенту (готовий до публікації, публікація – тільки в цьому випадку контент стає доступним клієнтам, архівований), дата, до якої доступний контент, мова контенту, сам основний файл та опис. У кінці потрібно натиснути на кнопку Створити (цифровий контент буде збережений, але не буде відображатися клієнту) або Опублікувати. Назва кнопки залежить від обраного з випадаючого переліку стану. Якщо окрім основного контенту потрібно додати ще файли, то для цього використовується секція Матеріал. У цій секції можна, наприклад, до відео додати фотографії, описати ці фотографії, а одній з них встановити прапорець, що вона є обкладинкою. Тоді, наприклад, на сторінці пошуку буде відображатися фотографія, а вже при відкритті контенту буде відображатися відео.

При відображенні динаміки (рис. 5.6) за каналом, відповідальним за ведення якого є авторизований користувач, надається ряд параметрів. Спочатку потрібно обрати, чи це режим реальної роботи (задається місяць, відображається кількість підписок на кожен день місяця або фактичні дані та прогноз на наступні 7 днів, якщо це поточний місяць), чи режим тестування. Прогнозування відбувається на наступні 7 днів в кінці кожного дня за допомогою автоматичного скрипту на основі навченої моделі. Результати записуються у файл прогнозування. У файлі фактичних даних міститься обчислена фактична кількість. Попередньо за минулими датами у файлі прогнозування зберігаються результати на кожні 7 днів (для тестування).

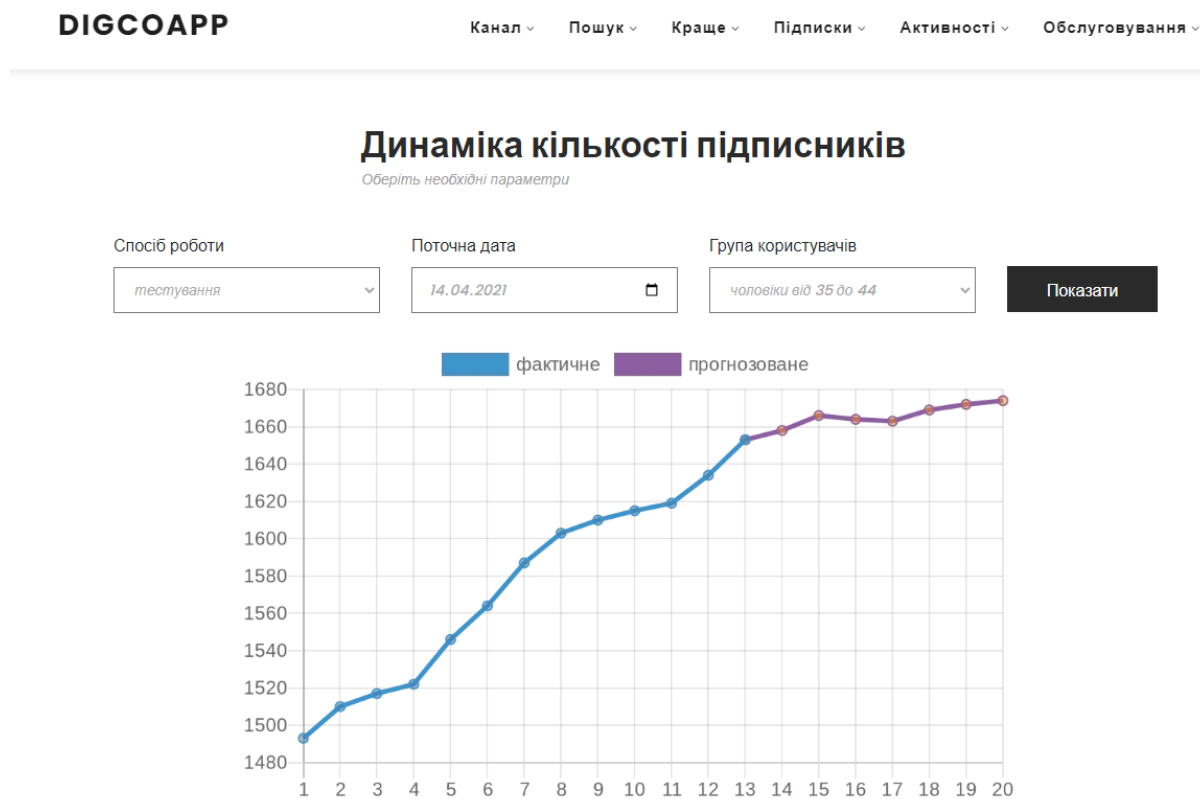


Рисунок 5.6 – Перегляд динаміки підписок

5.3 Експериментальне дослідження

Для проведення експериментального дослідження була використана вибірка даних [14], яка розповсюджується за однією з вільних ліцензій.

За денною кількістю підписок користувачів була розрахована середня кількість на кожен місяць для різних груп людей: зокрема для користувачів до 24 років (рис. 5.7) та від 25 до 34 років (рис. 5.8).

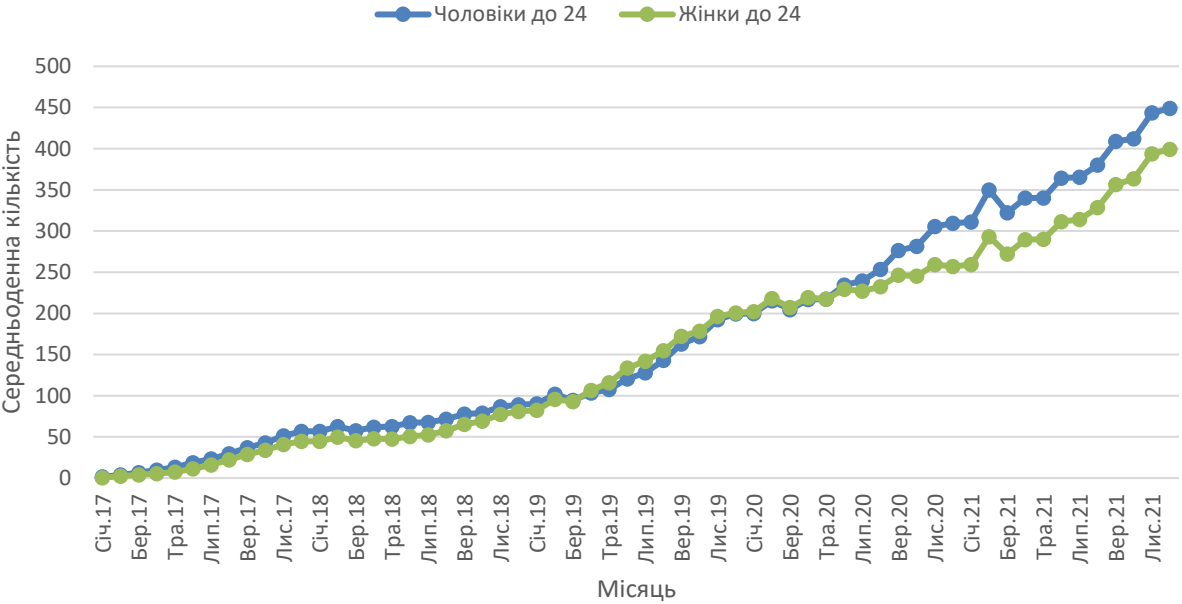


Рисунок 5.7 – Середньоденна кількість підписок користувачів до 24 років за місяцями

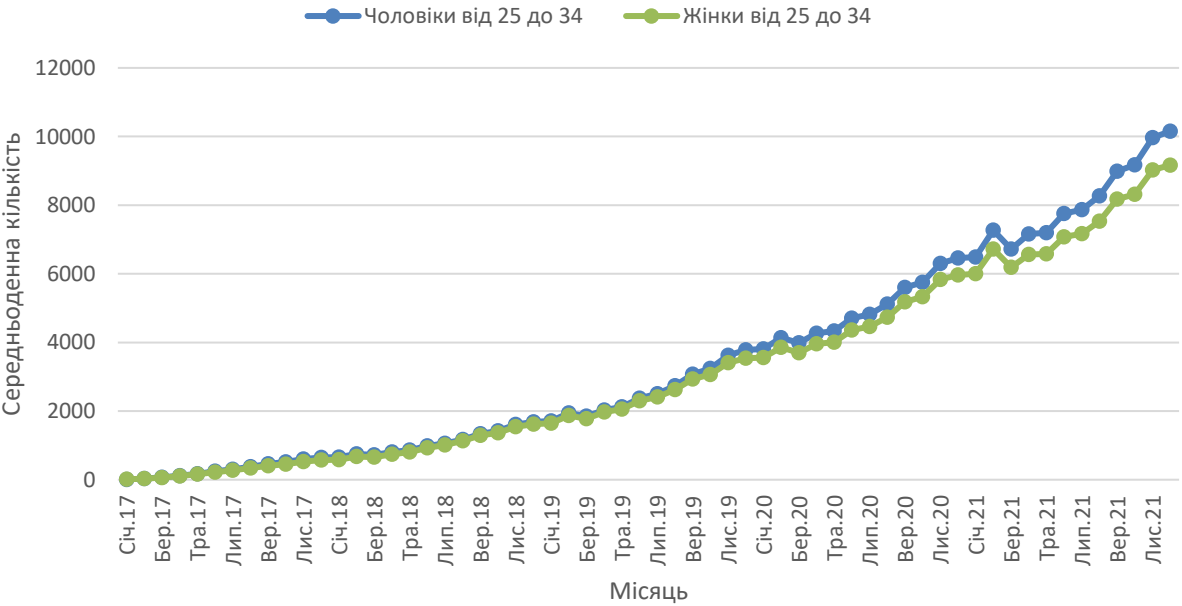


Рисунок 5.8 – Середньоденна кількість підписок користувачів від 25 до 34 років за місяцями

Середня кількість підписок була розрахована за місяцями для користувачів від 35 до 44 років (рис. 5.9) та від 45 до 54 років (рис. 5.10).

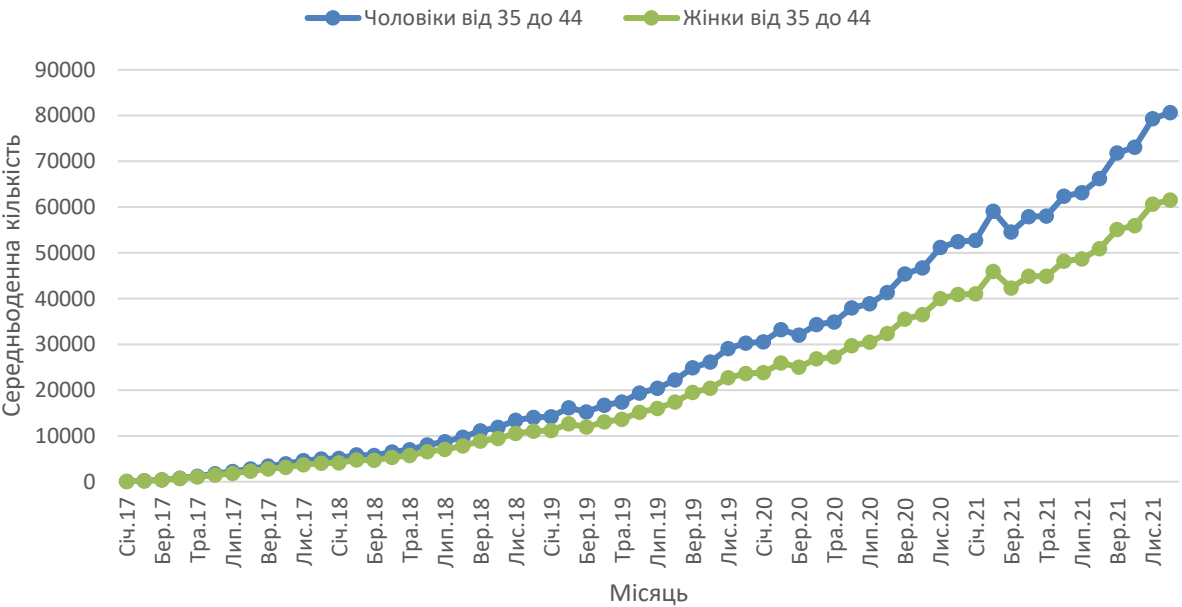


Рисунок 5.9 – Середньоденна кількість підписок користувачів від 35 до 44 років за місяцями

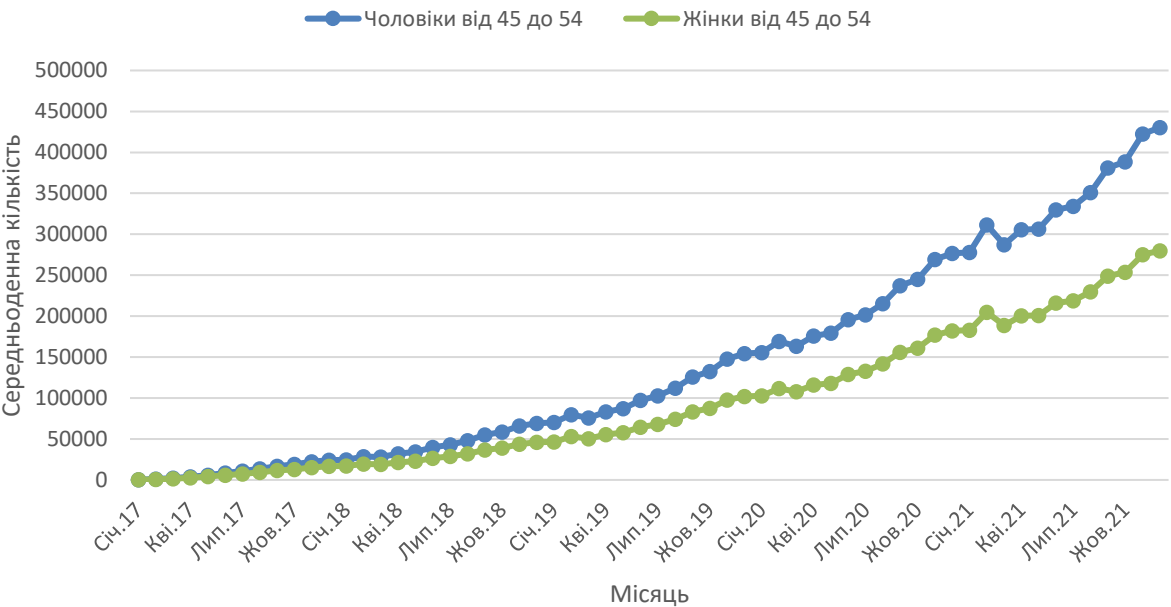


Рисунок 5.10 – Середньоденна кількість підписок користувачів від 45 до 54 років за місяцями

Середня кількість підписок була розрахована за місяцями для користувачів від 55 до 64 років (рис. 5.11) та від 65 років (рис. 5.12).

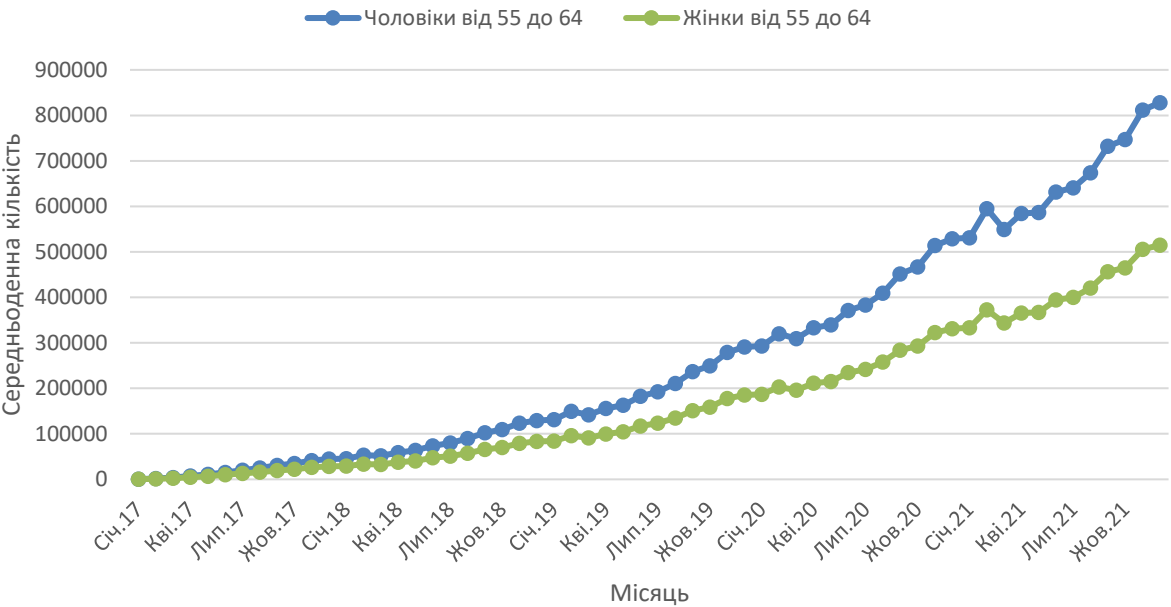


Рисунок 5.11 – Середньоденна кількість підписок користувачів від 55 до 64 років за місяцями

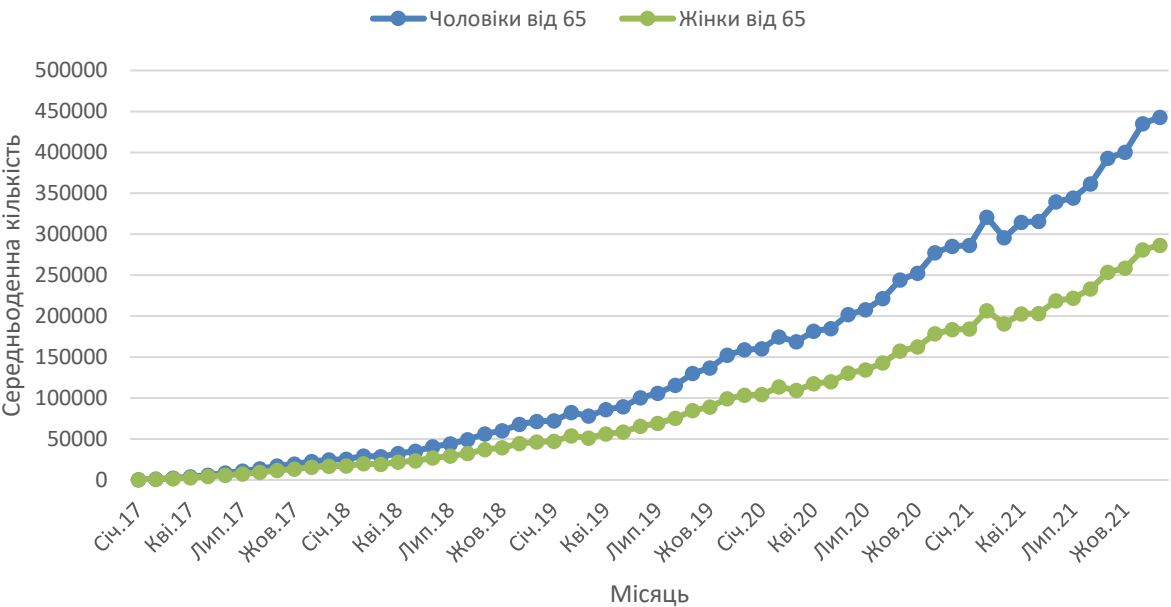


Рисунок 5.12 – Середньоденна кількість підписок користувачів від 65 років за місяцями

Середня кількість звернень у підтримку була розрахована за місяцями для користувачів до 24 років (рис. 5.13) та від 25 до 34 років (рис. 5.14).

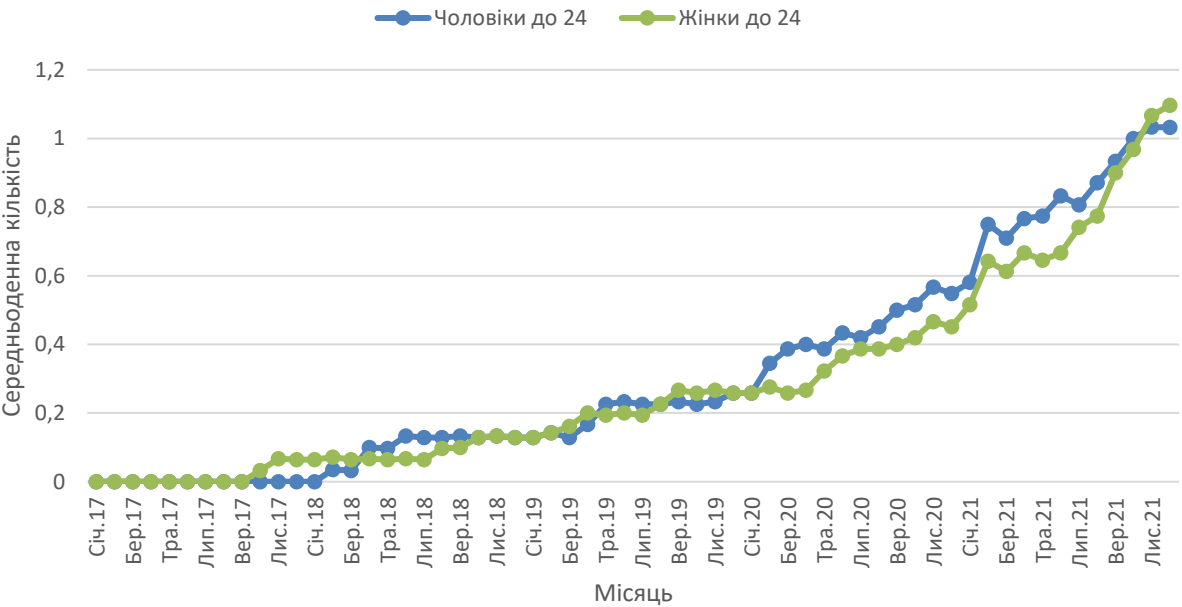


Рисунок 5.13 – Середньоденна кількість звернень у підтримку користувачів до 24 років за місяцями

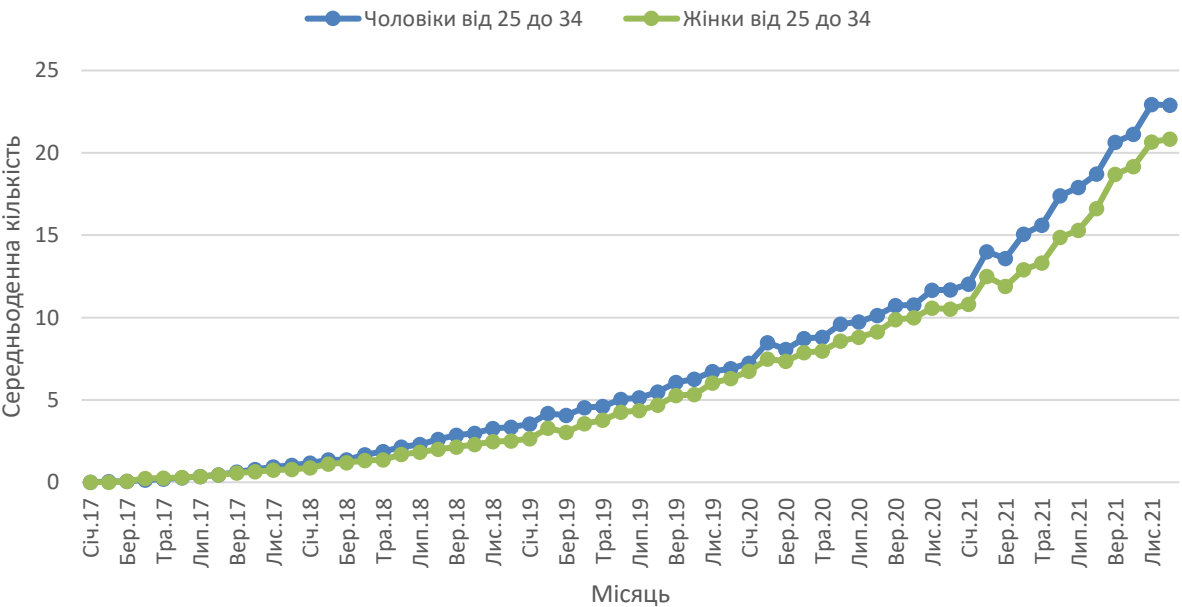


Рисунок 5.14 – Середньоденна кількість звернень у підтримку користувачів від 25 до 34 років за місяцями

Середня кількість звернень у підтримку була розрахована за місяцями для користувачів від 35 до 44 років (рис. 5.15) та від 45 до 54 років (рис. 5.16).

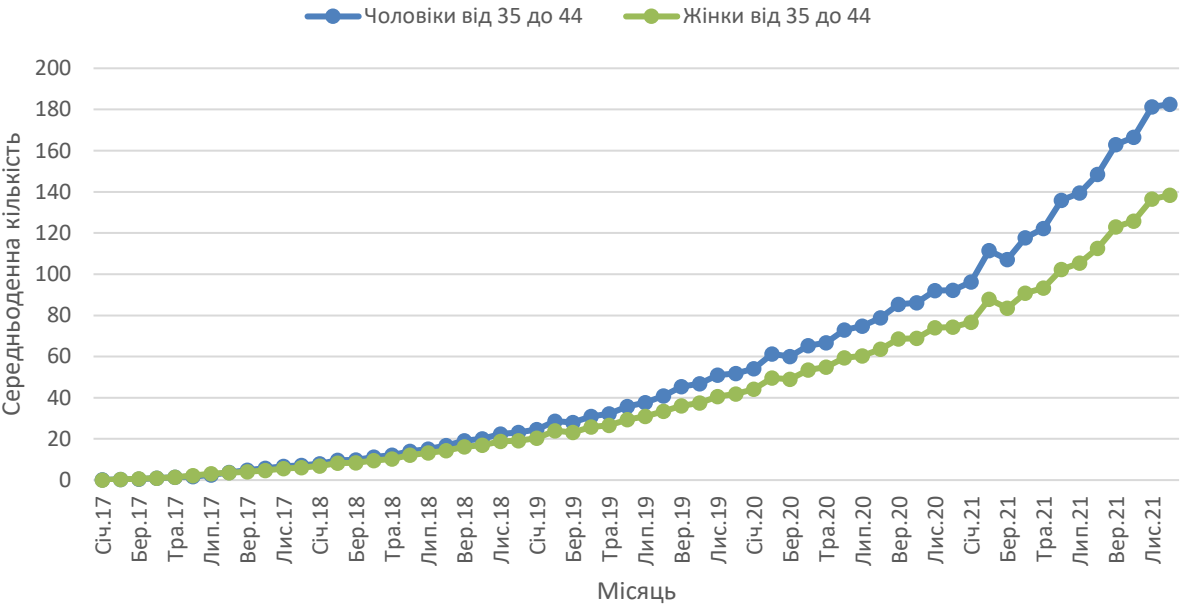


Рисунок 5.15 – Середньоденна кількість звернень у підтримку користувачів від 35 до 44 років за місяцями

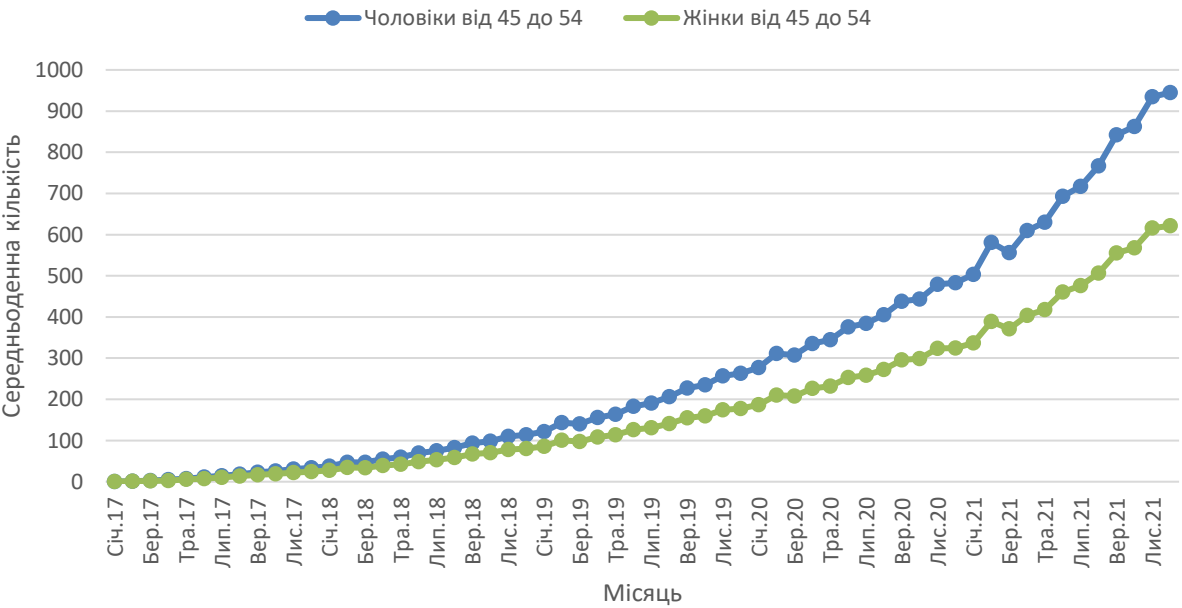


Рисунок 5.16 – Середньоденна кількість звернень у підтримку користувачів від 45 до 54 років за місяцями

Середня кількість звернень у підтримку була розрахована за місяцями для користувачів від 55 до 64 років (рис. 5.17) та від 65 років (рис. 5.18).

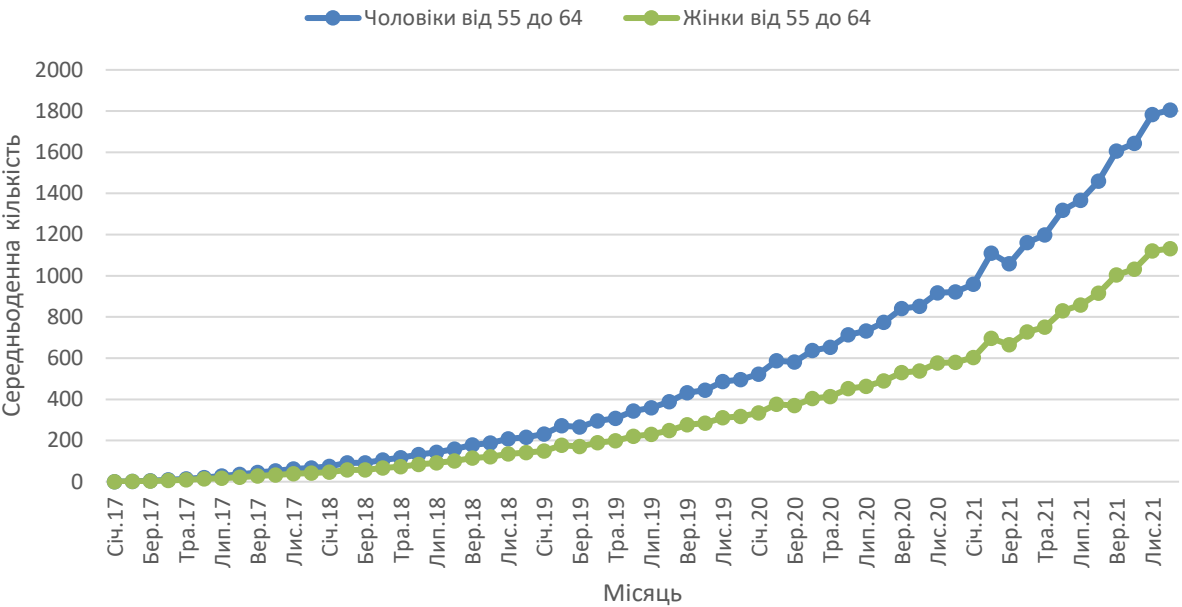


Рисунок 5.17 – Середньоденна кількість звернень у підтримку користувачів від 55 до 64 років за місяцями

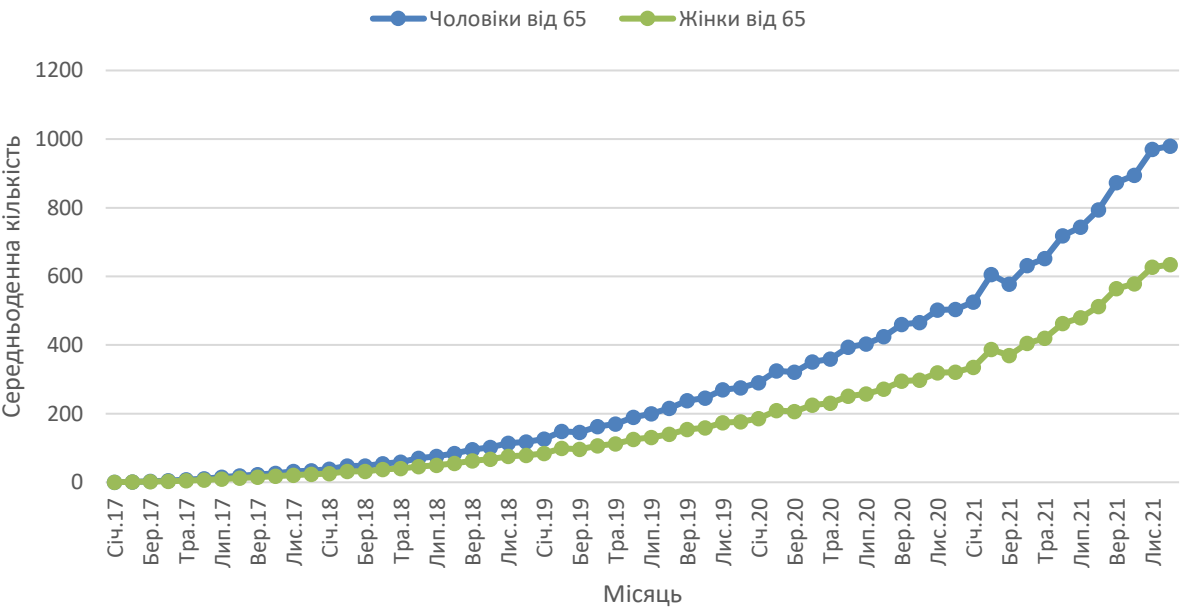


Рисунок 5.18 – Середньоденна кількість звернень у підтримку користувачів від 65 років за місяцями

Аналізуючи середню кількість звернень у підтримку, помітно, що значення кількості звернень дуже корелюють з кількістю підписок для різних груп користувачів. Тому використовувати ці дані для входів моделі не варто, бо вони дублюють дані кількості підписок.

Прогнозування підписок на цифровий контент відбувалось за допомогою запропонованого методу, створюючи моделі для кожної групи користувачів (табл. 5.1). Моделі на основі LSTM складались з 2 шарів (12 і 6 чарунок). Прогнозування підписок відбувалось на наступні 7 днів за минулі 7 днів. Результати оцінювались на основі середньоквадратичної похибки.

Таблиця 5.1 – Експериментальне дослідження прогнозування підписок на цифровий контент за допомогою запропонованого методу

Вікова група користувачів	Користувачі-чоловіки	Користувачі-жінки
До 24	0,0184	0,01977
Від 25 до 34	0,05167	0,04046
Від 35 до 44	0,04005	0,03563
Від 45 до 54	0,04825	0,04811
Від 55 до 64	0,05441	0,04466
Від 65	0,05007	0,05887

Окремо також було проведено те саме дослідження на основі ARIMA (табл. 5.2).

Основний набір груп користувачів дозволив отримати для LSTM зменшення значення середньоквадратичної похибки в 2-3 рази. Для користувачів у групі до 24 років покращення становило 7,22 рази для жінок та 8,19 % для чоловіків. Тож запропонований метод виявився у порівнянні ефективним.

Таблиця 5.2 – Експериментальне дослідження прогнозування підписок на цифровий контент за допомогою моделей на основі ARIMA

Вікова група користувачів	Користувачі-чоловіки	Користувачі-жінки
До 24	0,15061	0,14264
Від 25 до 34	0,12152	0,11681
Від 35 до 44	0,1074	0,10855
Від 45 до 54	0,11142	0,10559
Від 55 до 64	0,11264	0,11151
Від 65	0,11038	0,11701

5.4 Висновки за розділом 5

У розділі 5 показані і описані основні варіанти роботи з програмою та результати експериментального дослідження прогнозування підписок на цифровий контент за допомогою нейронних мереж LSTM та моделей ARIMA. Моделі LSTM, сформовані запропонованим методом, мали меншу похибку в результаті.

ВИСНОВКИ

У підсумку виконання роботи було досягнуто мету роботи розробити програмне забезпечення прогнозування підписок на цифровий контент для надання і контролю доступу до цифрового контенту та короткострокового прийняття рішень на основі динаміки чисельності груп підписників.

Розроблена програма в результаті виконання огляду аналогів, серед яких було виділено такі, що забезпечують платний доступ до цифрового контенту відомих компаній за допомогою відповідного єдиного сервісу, та такі, що забезпечують підписки на цифровий контент окремих авторів. Створене програмне забезпечення об'єднує ці ідеї та призначається для забезпечення роботи сервісів надання доступу до цифрового контенту централізованим чином і на основі створення окремих авторських каналів з можливістю надання цифрового контенту, забезпечення платних підписок на цей контент та контролю за станом підписок.

Реалізоване програмне забезпечення є вебзастосунком, для створення якого використано як основну мову програмування Python та мови забезпечення вебзасобів HTML, CSS, JavaScript, фреймворк Django, бібліотеки Tensorflow, Keras, Pandas, систему керування базами даних PostgreSQL.

Наукова новизна роботи визначається методом прогнозування підписок на цифровий контент, який дозволяє створювати моделі прогнозування окремо для різних груп користувачів, виконуючи прогнозування кількості підписок у кожній групі користувачів на наступні дні на основі даних за попередні дні за допомогою використання нейронних мереж на основі LSTM. Окрім безпосередньо створення методу було також проведено експериментальне дослідження, в якому порівняно створені за допомогою запропонованого методу моделі для різних груп користувачів та моделі на основі ARIMA.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. MEGOGO.NET Дивитися фільми та телеканали онлайн [Електрон. ресурс]. – Режим доступу : <https://megogo.net/ua>.
2. Головна – Patreon [Електрон. ресурс]. – Режим доступу : <https://www.patreon.com/>.
3. Buy Me a Coffee [Electronic resource]. – Access mode : <https://www.buymeacoffee.com/>.
4. McCaffrey, J. Understanding LSTM Cells Using C# [Electronic resource] / J. McCaffrey. – Access mode : <https://learn.microsoft.com/en-us/archive/msdn-magazine/2018/april/test-run-understanding-lstm-cells-using-csharp>.
5. Aggarwal, C. Neural Networks and Deep Learning [Text] : A Text-book / C. Aggarwal. – Chsm, Switzerland : Springer, 2018. – 520 p.
6. Brownlee, J. A Gentle Introduction to Long Short-Term Memory Networks by the Experts [Electronic resource] / J. Brownlee. – Access mode : <https://machinelearningmastery.com/gentle-introduction-long-short-term-memory-networks-experts/>.
7. Long Short Term Memory Networks Explanation [Electronic resource]. – Access mode : <https://www.geeksforgeeks.org/long-short-term-memory-networks-explanation/>.
8. Hexashop – Free Responsive Bootstrap 4 Ecommerce Website Template [Electronic resource]. – Access mode : <https://themewagon.com/themes/hexashop-free-responsive-bootstrap-4-ecommerce-website-template/>.
9. GitHub – spookykey/django-paypal [Electronic resource] : A pluggable Django application for integrating PayPal Payments Standard or Payments Pro. – Access mode : <https://github.com/spookykey/django-paypal>.
10. Welcome to django-paypal's documentation! [Electronic resource]. – Access mode : <https://django-paypal.readthedocs.io/en/latest/>.
11. TensorFlow [Electronic resource]. – Access mode :

<https://www.tensorflow.org/>.

12. Keras [Electronic resource] : Deep Learning for Humans. – Access mode : <https://keras.io/>.

13. Pandas – Python Data Analysis Library [Electronic resource]. – Access mode : <https://pandas.pydata.org/>.

14. Customer Subscription Data [Electronic resource]. – Access mode : https://www.kaggle.com/datasets/gsagar12/dspp1?select=product_info.csv.

ДОДАТОК А
Технічне завдання

Вступ

Цифровий контент є частиною життя майже кожної сучасної людини. Разом з доступом до контенту виникає проблема забезпечення прав авторів. Фактично для цього необхідно забезпечити платний доступ до таких матеріалів. Однак, при забезпеченні такого доступу виникають різні варіанти, як це зробити. Нерідко додатки створюються для забезпечення роботи окремих сервісів. У свою чергу окремі автори фактично мають використовувати платформи, подібні до платформ збору коштів. Виникає фрагментованість, коли є багато платформ для підтримки доступу до централізованих сервісів та платформ для збору коштів для авторів. Багато користувачів фактично не знають або не звертають уваги на такі платформи. Об'єднання ж цих ідей може принести вигоду як окремому сервісу, так і авторам цифрового контенту. Однак, окрім наявності такого інструменту важливим є також і наявність інструментів аналізу та прогнозування підписок на цифровий контент, щоб можна було приймати рішення про те, як розвивати сервіс, каналі надалі, на яку аудиторію орієнтуватися.

A.1 Підстави для розробки

Наказ № 437 від 14 листопада 2023 р. за Національним університетом «Запорізька політехніка» з зазначенням теми роботи «Дослідження та програмна реалізація методів прогнозування підписок на цифровий контент» та завдання на роботу є підставами для розробки.

A.2 Призначення розробки

Розробка призначається для забезпечення роботи сервісів надання доступу до цифрового контенту централізованим чином та на основі створення окремих авторських каналів з можливістю надання цифрового

контенту, забезпечення платних підписок на цей контент та контролю за станом підписок.

А.3 Основні вимоги до програми

А.3.1 Вимоги до функціональних характеристик

Забезпечення функціональних характеристик повинно вкладатися в наступний набір функцій:

- створити авторський канал;
- редагувати опис авторського каналу;
- додати відповідального за ведення контенту;
- вилучити відповідального за ведення контенту;
- додати або вилучити категорію до авторського каналу;
- додати проєкт цифрового контенту до авторського каналу;
- редагувати опис проєкту цифрового контенту;
- додати цифровий контент до проєкту;
- редагувати цифровий контент;
- додати або вилучити категорію до цифрового контенту;
- додати або вилучити тег з цифрового контенту;
- вилучити цифровий контент з проєкту;
- вилучити проєкт цифрового контенту з авторського каналу;
- опублікувати цифровий контент;
- архівувати цифровий контент;
- додати матеріал до цифрового контенту;
- редагувати матеріал у складі цифрового контенту;
- вилучити матеріал з цифрового контенту;
- відтворити цифровий контент;
- переглянути перелік цифрового контенту за проєктом;
- переглянути проєкти авторського каналу;
- переглянути профіль клієнта;

- оновити власний профіль;
- шукати авторські канали;
- шукати цифровий контент;
- шукати проєкти цифрового контенту;
- зберегти цифровий контент;
- переглянути цифровий контент, збережений авторизованим користувачем;
- вилучити цифровий контент зі збереженого;
- уподобати або зняти уподобання з цифрового контенту;
- виставити антипатію або зняти її з цифрового контенту;
- переглянути цифровий контент, уподобаний авторизованим користувачем;
- переглянути цифровий контент, якому було виставлено антипатію авторизованим користувачем;
- коментувати цифровий контент;
- переглянути історію відтворення цифрового контенту;
- переглянути коментарі, створені авторизованим користувачем;
- переглянути оцінки за авторським каналом;
- переглянути оцінки за цифровим контентом;
- виставити оцінку авторському каналу;
- виставити оцінку цифровому контенту;
- переглянути оцінки авторським каналам, виставлені авторизованим користувачем;
- переглянути оцінки цифровому контенту, виставлені авторизованим користувачем;
- авторизувати, зареєструвати обліковий запис клієнта або відповідального за ведення авторського каналу;
- додати інформацію про людину, яка може бути учасником створення цифрового контенту;
- додати учасника створення авторського каналу;

- редагувати інформацію про учасника створення авторського каналу;
- вилучити учасника створення авторського каналу;
- додати учасника створення цифрового контенту;
- редагувати інформацію про учасника створення цифрового контенту;
- вилучити учасника створення цифрового контенту;
- фіксувати запит на технічну підтримку;
- оформити запит на технічну підтримку;
- створити або оновити результат оброблення запиту на технічну підтримку;
- переглянути запити на технічну підтримку, оформлені авторизованим користувачем;
- переглянути запити на технічну підтримку за сервісом;
- створити спілкування щодо авторського каналу;
- переглянути створені спілкування за авторськими каналами авторизованим користувачем;
- переглянути увесь цифровий контент за авторським каналом;
- відправити або вилучити повідомлення зі спілкування;
- переглянути повідомлення, які надійшли за авторським каналом;
- визначити доступний варіант загальної підписки на цифровий контент;
- переглянути варіанти загальної підписки;
- оформити загальну підписку;
- переглянути стан загальних підписок;
- переглянути динаміку загальних підписок;
- переглянути динаміку підписок за авторським каналом;
- визначити доступний варіант підписки на авторський канал на певний період;
- переглянути варіанти підписки на авторський канал на певний період або на задану кількість цифрового контенту;

- визначити доступний варіант підписки на авторський канал на задану кількість цифрового контенту;
- оформити підписку на авторський канал на певний період або на задану кількість цифрового контенту;
- переглянути доступний цифровий контент, на який оформлено підписку;
- переглянути авторські канали, на які оформлено підписку;
- переглянути цифровий контент за заданим тегом;
- переглянути цифровий контент за заданою категорією;
- переглянути авторські канали за заданою категорією;
- переглянути найкращі авторські канали;
- переглянути найкращий цифровий контент.

A.3.2 Умови експлуатації

Програмне забезпечення в підсумку має підтримуватись через організацію хостингу для роботи з сервером, розміщення на сервері вебзастосунку з базовим налаштуванням Django, наявності інтерпретатора python 3, встановлення пакетів Tensorflow, Keras. Або у складі цього серверу, або як окремий сервер БД має бути використано і налаштовано систему PostgreSQL.

A.3.3 Вимоги до складу та параметрів технічних засобів

Фактичні вимоги до технічних засобів мають враховувати кількість користувачів, підтримка роботи з якими має забезпечуватися. Як тільки кількість користувачів збільшується, вимоги мають зростати. Мінімальні вимоги, які визначені нижче, враховують можливість одночасної роботи з сервером 25 користувачів. У розрахунках враховано, що очікувана кількість користувачів вебзастосунку – 100. За такої кількості максимально 25 з них

можуть звернутися до вебзастосунку одночасно. Кількість авторських каналів – 10. На кожний канал має бути виділено мінімум 10 Гб для збереження даних. Кожен користувач повинен мати 100 Мб для збереження вкладень до повідомлень. Тобто сумарно необхідно 110 Гб та 5 Гб на роботу БД. Самі вимоги до організації сервера також включають мінімально процесор Intel Xeon E5440, який має частоту 2,83 ГГц, та 4 Гб доступної оперативної пам'яті.

А.4 Порядок контролю та приймання

Строки виконання дипломної кваліфікаційної роботи магістра визначені в завданні на роботу, включаючи строк подання роботи, календарний план виконання роботи з зазначеними етапами, результатами та строками. Захист роботи відбувається за оголошеним графіком захисту дипломних кваліфікаційних робіт і відповідно роботи екзаменаційної комісії.

ДОДАТОК Б
Текст програми

```

    from .models import Authorhood, Category, Member,
    Authorhoodmember, Content, Tag, Contentmember, Subscription, Customer,
    Contentpayment, Timepayment, Project, Save, Contentlike,
    Contentdislike, Tag, Category, Contentassessment, Comment,
    Authorassessment, Look, Authorcontentpayment, Authortimepayment
    from .forms import CommentForm, ContentassessmentForm,
    AuthorassessmentForm
    from django.utils.decorators import method_decorator
    from django.shortcuts import render
    import datetime
    from django.db.models import Q, Count
    from django.views.decorators.csrf import csrf_exempt
    from django.db.utils import IntegrityError
    from django.http import HttpResponse
    from django.contrib.auth.mixins import LoginRequiredMixin
    from django.views import View
    from django.shortcuts import get_object_or_404

    def authorhood_search(request):
        authorhood_caption = self.request.GET.get('caption')
        authorhoods = None
        if authorhood_caption:
            authorhoods =
    Authorhood.objects.filter(caption__icontains =
    authorhood_caption).prefetch_related("categories").prefetch_related("a
    uthorhoodmembers").prefetch_related("authorhoodmembers__member")
            authorhood_subscription =
    self.request.GET.get('subscription')
            authorhood_subscription = int(authorhood_subscription)
            if authorhood_subscription and authorhoods:
                authorhoods = authorhoods.filter(subscription_type =
    authorhood_subscription)
            elif authorhood_subscription:
                authorhoods =
    Authorhood.objects.filter(subscription_type =
    authorhood_subscription).prefetch_related("categories").prefetch_relat
    ed("authorhoodmembers").prefetch_related("authorhoodmembers__member")
            authorhood_minimum = self.request.GET.get('assessment1')
            authorhood_minimum = int(authorhood_minimum)
            if authorhood_minimum and authorhoods:
                authorhoods = authorhoods.filter(assessment__gte =
    authorhood_minimum)
            elif authorhood_minimum:
                authorhoods = Authorhood.objects.filter(assessment__gte
    =

```

```

authorhood_minimum).prefetch_related("categories").prefetch_related("a
uthorhoodmembers").prefetch_related("authorhoodmembers__member")
    authorhood_maximum = self.request.GET.get('assessment2')
    authorhood_maximum = int(authorhood_maximum)
    if authorhood_maximum and authorhoods:
        authorhoods = authorhoods.filter(assessment__lte =
authorhood_maximum)
    elif authorhood_maximum:
        authorhoods = Authorhood.objects.filter(assessment__lte
=
authorhood_maximum).prefetch_related("categories").prefetch_related("a
uthorhoodmembers").prefetch_related("authorhoodmembers__member")
        categories = self.request.GET.get('categories')
        categories_split = categories.split(',')
    if categories_split:
        categories_condition = []
        for category in categories_split:
            category = category.lower()
            category = Category.objects.get(name = category)
            categories_condition.append(category)
        if authorhoods:
            authorhoods = authorhoods.filter(categories__in =
categories_condition)
        else:
            authorhoods =
Authorhoods.objects.filter(categories__in =
categories_condition).prefetch_related("categories").prefetch_related("
authorhoodmembers").prefetch_related("authorhoodmembers__member")
            members = self.request.GET.get('members')
            members_split = members.split(',')
            if members_split:
                members_condition = []
                for member in members_split:
                    member = member.lower()
                    member = Member.objects.get(name = member)
                    members_condition.append(member)
                members_condition =
Authorhoodmember.objects.filter(member__in = members_condition)
                if authorhoods:
                    authorhoods =
authorhoods.filter(authorhoodmembers__in = members_condition)
                else:
                    authorhoods =
Authorhoods.objects.filter(authorhoodmembers__in =
members_condition).prefetch_related("categories").prefetch_related("aut
horhoodmembers").prefetch_related("authorhoodmembers__member")

```

```

        if authorhoods:
            authorhoods = authorhoods.order_by('-
assessment').values()
            return render(request,
"digcoapp/authorhood_search.html", {"authorhoods": authorhoods })
        else:
            authorhoods =
Authorhoods.objects.all().prefetch_related("categories").prefetch_rela
ted("authorhoodmembers").prefetch_related("authorhoodmembers__member")
.order_by('-assessment').values()
            return render(request,
"digcoapp/authorhood_search.html", {"authorhoods": authorhoods })

def digital_content_search(request):
    content_caption = self.request.GET.get('caption')
    digital_contents = None
    if content_caption:
        digital_contents =
Content.objects.filter(caption__icontains =
content_caption).select_related("project").select_related("project__au
thorhood").prefetch_related("categories").prefetch_related("contentmem
bers").prefetch_related("contentmembers__member").prefetch_related("le
adstuff").prefetch_related("saves").annotate(Count("likes",
distinct=True)).annotate(Count("dislikes",
distinct=True)).annotate(Count("looks", distinct=True))
        authorhood_caption = self.request.GET.get('authcaption')
        if authorhood_caption and digital_contents:
            digital_contents =
digital_contents.filter(project__authorhood__caption__icontains =
authorhood_caption)
        else:
            digital_contents =
Content.objects.filter(project__authorhood__caption__icontains =
authorhood_caption).select_related("project").select_related("project_
authorhood").prefetch_related("categories").prefetch_related("content
members").prefetch_related("contentmembers__member").prefetch_related(
"leadstuff").prefetch_related("saves").annotate(Count("likes",
distinct=True)).annotate(Count("dislikes",
distinct=True)).annotate(Count("looks", distinct=True))
            content_type = self.request.GET.get('contenttype')
            content_type = int(content_type)
            if content_type and digital_contents:
                digital_contents = digital_contents.filter(content_type
= content_type)
            elif content_type:

```

```

        digital_contents = Content.objects.filter(content_type =
content_type).select_related("project").select_related("project__authorhood").prefetch_related("categories").prefetch_related("contentmembers").prefetch_related("contentmembers__member").prefetch_related("leadstuff").prefetch_related("saves").annotate(Count("likes", distinct=True)).annotate(Count("dislikes", distinct=True)).annotate(Count("looks", distinct=True))
        authorhood_minimum = self.request.GET.get('authassessment1')
        authorhood_minimum = int(authorhood_minimum)
        if authorhood_minimum and digital_contents:
            digital_contents =
digital_contents.filter(project__authorhood__assessment__gte =
authorhood_minimum)
            elif authorhood_minimum:
                digital_contents =
Content.objects.filter(project__authorhood__assessment__gte =
authorhood_minimum).select_related("project").select_related("project__authorhood").prefetch_related("categories").prefetch_related("contentmembers").prefetch_related("contentmembers__member").prefetch_related("leadstuff").prefetch_related("saves").annotate(Count("likes", distinct=True)).annotate(Count("dislikes", distinct=True)).annotate(Count("looks", distinct=True))
                authorhood_maximum = self.request.GET.get('authassessment2')
                authorhood_maximum = int(authorhood_maximum)
                if authorhood_maximum and digital_contents:
                    digital_contents =
digital_contents.filter(project__authorhood__assessment__lte =
authorhood_maximum)
                    elif authorhood_maximum:
                        digital_contents =
Content.objects.filter(project__authorhood__assessment__lte =
authorhood_maximum).select_related("project").select_related("project__authorhood").prefetch_related("categories").prefetch_related("contentmembers").prefetch_related("contentmembers__member").prefetch_related("leadstuff").prefetch_related("saves").annotate(Count("likes", distinct=True)).annotate(Count("dislikes", distinct=True)).annotate(Count("looks", distinct=True))
                        content_minimum = self.request.GET.get('assessment1')
                        content_minimum = int(content_minimum)
                        if content_minimum and digital_contents:
                            digital_contents =
digital_contents.filter(assessment__gte = content_minimum)
                            elif content_minimum:
                                digital_contents =
Content.objects.filter(assessment__gte =

```

```

content_minimum).select_related("project").select_related("project__authorhood").prefetch_related("categories").prefetch_related("contentmembers").prefetch_related("contentmembers__member").prefetch_related("leadstuff").prefetch_related("saves").annotate(Count("likes", distinct=True)).annotate(Count("dislikes", distinct=True)).annotate(Count("looks", distinct=True))
        content_maximum = self.request.GET.get('authassessment2')
        content_maximum = int(content_maximum)
        if content_maximum and digital_contents:
            digital_contents =
digital_contents.filter(assessment__lte = content_maximum)
        elif content_maximum:
            digital_contents =
Content.objects.filter(assessment__lte =
content_maximum).select_related("project").select_related("project__authorhood").prefetch_related("categories").prefetch_related("contentmembers").prefetch_related("contentmembers__member").prefetch_related("leadstuff").prefetch_related("saves").annotate(Count("likes", distinct=True)).annotate(Count("dislikes", distinct=True)).annotate(Count("looks", distinct=True))
            categories = self.request.GET.get('categories')
            categories_split = categories.split(',')
            if categories_split:
                categories_condition = []
                for category in categories_split:
                    category = category.lower()
                    category = Category.objects.get(name = category)
                    categories_condition.append(category)
            if digital_contents:
                digital_contents =
digital_contents.filter(categories__in = categories_condition)
        else:
            digital_contents =
Content.objects.filter(categories__in =
categories_condition).select_related("project").select_related("project__authorhood").prefetch_related("categories").prefetch_related("contentmembers").prefetch_related("contentmembers__member").prefetch_related("leadstuff").prefetch_related("saves").annotate(Count("likes", distinct=True)).annotate(Count("dislikes", distinct=True)).annotate(Count("looks", distinct=True))
            tags = self.request.GET.get('tags')
            tags_split = tags.split(',')
            if tags_split:
                tags_condition = []
                for tag in tags_split:
                    tag = tag.lower()

```

```

        tag = Tag.objects.get(caption = tag)
        tags_condition.append(tag)
    if digital_contents:
        digital_contents = digital_contents.filter(tags__in
= tags_condition)
    else:
        digital_contents = Content.objects.filter(tags__in =
tags_condition).select_related("project").select_related("project__auth
orhood").prefetch_related("categories").prefetch_related("contentmembe
rs").prefetch_related("contentmembers__member").prefetch_related("lead
stuff").prefetch_related("saves").annotate(Count("likes",
distinct=True)).annotate(Count("dislikes",
distinct=True)).annotate(Count("looks", distinct=True))
        members = self.request.GET.get('members')
        members_split = members.split(',')
    if members_split:
        members_condition = []
        for member in members_split:
            member = member.lower()
            member = Member.objects.get(name = member)
            members_condition.append(member)
        members_condition = Contentmember.objects.filter(member__in
= members_condition)
        if digital_contents:
            digital_contents =
digital_contents.filter(contentmembers__in = members_condition)
        else:
            digital_contents =
Content.objects.filter(contentmembers__in =
members_condition).select_related("project").select_related("project__a
uthorhood").prefetch_related("categories").prefetch_related("contentme
mbers").prefetch_related("contentmembers__member").prefetch_related("l
eadstuff").prefetch_related("saves").annotate(Count("likes",
distinct=True)).annotate(Count("dislikes",
distinct=True)).annotate(Count("looks", distinct=True))
            if not digital_contents:
                digital_contents =
Content.objects.filter(contentmembers__in =
members_condition).select_related("project").select_related("project__a
uthorhood").prefetch_related("categories").prefetch_related("contentme
mbers").prefetch_related("contentmembers__member").prefetch_related("l
eadstuff").prefetch_related("saves").annotate(Count("likes",
distinct=True)).annotate(Count("dislikes",
distinct=True)).annotate(Count("looks", distinct=True))
            content_language = self.request.GET.get('language')
            if content_language and digital_contents:

```

```

        digital_contents = digital_contents.filter(language =
content_language)
    else:
        digital_contents = Content.objects.filter(language =
content_language).select_related("project").select_related("project__a
uthorhood").prefetch_related("categories").prefetch_related("contentme
mbers").prefetch_related("contentmembers__member").prefetch_related("l
eadstuff").prefetch_related("saves").annotate(Count("likes",
distinct=True)).annotate(Count("dislikes",
distinct=True)).annotate(Count("looks", distinct=True))
        search_time = datetime.datetime.now()
        digital_contents = digital_contents.filter(access_until__lte
= search_time, state = 1)
        digital_contents = digital_contents.filter(state = 1)
        digital_contents = digital_contents.order_by('-assessment')
        digital_content_results = []
        digital_content_unavailable = []
        customer = Customer.objects.get(user = request.user)
        subscription = None
        done = False
        if customer:
            subscription = Subscription.objects.filter(state = True,
customer = customer)
        for digital_content in digital_contents:
            if digital_content.access_type == 0:
                digital_content_results.append(digital_content)
                done = True
            elif digital_content.access_type == 1 and subscription:
                digital_content_results.append(digital_content)
                done = True
            elif digital_content.access_type == 2 and
digital_content.project.authorhood.subscription_type == 2:
                active = Timepayment.objects.filter(customer =
customer, state = True, authortimepayment__authorhood =
digital_content.project.authorhood)
                if active:
                    digital_content_results.append(digital_content)
                    done = True
                else:
                    active_subscriptions =
Contentpayment.objects.filter(customer = customer,
authorcontentpayment__authorhood = digital_content.project.authorhood,
state = True)
                    if active_subscriptions:
digital_content_results.append(digital_content)

```

```

        done = True
    else:
        active_subscriptions =
Contentpayment.objects.filter(customer = customer,
authorcontentpayment__authorhood = digital_content.project.authorhood)
        for active_subscription in
active_subscriptions:
            contents =
Content.objects.filter(fixed_time__gte =
active_subscription.fixed_time).order_by("fixed_time")[
0:active_subscription.authorcontentpayment.content_number]
            if digital_content in contents:

digital_content_results.append(digital_content)
                done = True
                break

        if not done:
            digital_content_unavailable.append(digital_content)
    return render(request,
"digcoapp/digital_content_search.html", {"digital_contents":
digital_content_results, "digital_unavailables":
digital_content_unavailable })

def content_project_search(request):
    project_caption = self.request.GET.get('caption')
    projects = None
    if project_caption:
        projects = Project.objects.filter(caption__icontains =
project_caption).select_related("authorhood").annotate(Count("contents
", distinct=True))
        authorhood_caption = self.request.GET.get('authcaption')
        if authorhood_caption and digital_contents:
            projects =
projects.filter(authorhood__caption__icontains = authorhood_caption)
    else:
        projects =
Project.objects.filter(authorhood__caption__icontains =
authorhood_caption).select_related("authorhood").annotate(Count("conte
nts", distinct=True))
        project_minimum = self.request.GET.get('assessment1')
        project_minimum = int(project_minimum)
        if project_minimum and authorhoods:
            projects = projects.filter(authorhood__assessment__gte =
project_minimum)
        elif project_minimum:

```

```

        projects =
Project.objects.filter(authorhood__assessment__gte =
project_minimum).select_related("authorhood").annotate(Count("contents
", distinct=True))
        project_maximum = self.request.GET.get('assessment2')
        project_maximum = int(project_maximum)
        if project_maximum and projects:
            projects = projects.filter(authorhood__assessment__lte =
project_maximum)
        elif project_maximum:
            projects =
Project.objects.filter(authorhood__assessment__lte =
project_maximum).select_related("authorhood").annotate(Count("contents
", distinct=True))
        if not projects:
            projects =
Project.objects.all().select_related("authorhood").annotate(Count("con
tents", distinct=True))
        return render(request,
"digcoapp/content_project_search.html", {"projects": projects })

```

```

@method_decorator(csrf_exempt,name='dispatch')
class DigitalContentSaveView(LoginRequiredMixin,View):
    def post(self, request, pk):
        try:
            digital_content = Content.objects.get(id = pk)
        except Content.DoesNotExist:
            digital_content = None
        try:
            current_customer = Customer.objects.get(user =
request.user)
        except Customer.DoesNotExist:
            current_customer = None
        if digital_content and current_customer:
            digital_content_save = Save(customer =
current_customer, content = digital_content)
            try:
                digital_content_save.save()
            except IntegrityError:
                digital_content_save =
Save.objects.filter(customer = current_customer, content =
digital_content).delete()
            return HttpResponse()

```

```

@method_decorator(csrf_exempt,name='dispatch')

```

```

class DigitalContentLikeView(LoginRequiredMixin,View):
    def post(self, request, pk):
        try:
            digital_content = Content.objects.get(id = pk)
        except Content.DoesNotExist:
            digital_content = None
        try:
            current_customer = Customer.objects.get(user =
request.user)
        except Customer.DoesNotExist:
            current_customer = None
        if digital_content and current_customer:
            try:
                digital_content_dislike =
Contentdislike(customer = current_customer, content = digital_content)
            except Contentdislike.DoesNotExist:
                digital_content_dislike = None
            if digital_content_dislike:
                digital_content_dislike =
Contentdislike.objects.filter(customer = current_customer, content =
digital_content).delete()
                digital_content_like = Contentlike(customer =
current_customer, content = digital_content)
            try:
                digital_content_like.save()
            except IntegrityError:
                digital_content_like =
Contentlike.objects.filter(customer = current_customer, content =
digital_content).delete()
            return HttpResponse()

    @method_decorator(csrf_exempt,name='dispatch')
class DigitalContentDislikeView(LoginRequiredMixin,View):
    def post(self, request, pk):
        try:
            digital_content = Content.objects.get(id = pk)
        except Content.DoesNotExist:
            digital_content = None
        try:
            current_customer = Customer.objects.get(user =
request.user)
        except Customer.DoesNotExist:
            current_customer = None
        if digital_content and current_customer:
            try:

```

```

        digital_content_like = Contentlike(customer =
current_customer, content = digital_content)
    except Contentlike.DoesNotExist:
        digital_content_like = None
    if digital_content_like:
        digital_content_like =
Contentlike.objects.filter(customer = current_customer, content =
digital_content).delete()
        digital_content_dislike = Contentdislike (customer =
current_customer, content = digital_content)
    try:
        digital_content_dislike.save()
    except IntegrityError:
        digital_content_dislike =
Contentdislike.objects.filter(customer = current_customer, content =
digital_content).delete()
    return HttpResponse()

```

```

def digital_content_look(request, digital_content):
    digital_content = get_object_or_404(Content, pk =
digital_content).prefetch_related("tags").prefetch_related("categories
").prefetch_related("comments").prefetch_related("likes").prefetch_rel
ated("dislikes").select_related("project").select_related("project__au
thorhood").prefetch_related("contentmembers").prefetch_related("conte
ntmembers__member").prefetch_related("contentmembers__role")
    try:
        current_customer = Customer.objects.get(user =
request.user)
    except Customer.DoesNotExist:
        current_customer = None
    digital_content_result = False
    subscription = None
    if current_customer:
        subscription = Subscription.objects.filter(state = True,
customer = current_customer)
    if digital_content.access_type == 0:
        digital_content_result = True
    elif digital_content.access_type == 1 and subscription:
        digital_content_result = True
    elif digital_content.access_type == 2 and
digital_content.project.authorhood.subscription_type == 2:
        active = Timepayment.objects.filter(customer =
current_customer, state = True, authortimepayment__authorhood =
digital_content.project.authorhood)
        if active:

```

```

        digital_content_result = True
    else:
        active_subscriptions =
Contentpayment.objects.filter(customer = current_customer,
authorcontentpayment__authorhood = digital_content.project.authorhood,
state = True)
        if active_subscriptions:
            digital_content_result = True
        else:
            active_subscriptions =
Contentpayment.objects.filter(customer = current_customer,
authorcontentpayment__authorhood = digital_content.project.authorhood)
            for active_subscription in active_subscriptions:
                contents =
Content.objects.filter(fixed_time__gte =
active_subscription.fixed_time).order_by("fixed_time")[
0:active_subscription.authorcontentpayment.content_number]
                if digital_content in contents:
                    digital_content_result = True
                    break
    if digital_content_result:
        digital_content_save = None
        if current_customer:
            try:
                digital_content_save = Save.objects.get(customer
= current_customer, content = digital_content)
            except Save.DoesNotExist:
                digital_content_save = None
        if digital_content_save:
            save_state = True
        else:
            save_state = False
        if current_customer:
            digital_content_look = Look(content =
digital_content, customer = current_customer).save()
            if request.method == "POST":
                digital_content_comment_form =
CommentForm(request.POST)
                if digital_content_comment_form.is_valid():
                    digital_content_comment =
digital_content_comment_form.save(commit = False)
                    digital_content_comment.content =
digital_content
                    digital_content_comment.customer=
current_customer
                    digital_content_comment.save()

```

```

        return
    HttpResponseRedirect(digital_content.get_absolute_url())
    else:
        digital_content_comment_form = None
        return render(request, "digcoapp/digital_content.html",
{"content": digital_content, "saved": save_state, "form":
digital_content_comment_form })
    else:
        return render(request, "digcoapp/digital_content.html",
{"content": None, "saved": False, "form": None })

def project_digital_contents(request, digital_project):
    try:
        digital_project = Project.objects.get(id =
digital_project).select_related("authorhood")
    except Project.DoesNotExist:
        digital_project = None
    if digital_project:
        digital_contents = Content.objects.filter(project =
digital_content, state =
1).prefetch_related("tags").prefetch_related("categories").annotate(Count("likes", distinct=True)).annotate(Count("dislikes",
distinct=True)).annotate(Count("looks",
distinct=True)).prefetch_related("contentmembers").prefetch_related("contentmembers__member").prefetch_related("contentmembers__role").order
by("-fixed_time")
        return render(request,
"digcoapp/digital_project_contents.html", {"contents":
digital_contents, "project": digital_project })
    else:
        return render(request,
"digcoapp/digital_project_contents.html", {"contents": [], "project":
None })

def saved_digital_contents(request):
    try:
        current_customer = Customer.objects.get(user =
request.user)
    except Customer.DoesNotExist:
        current_customer = None
    if current_customer:
        digital_content_saves = Save.objects.filter(customer =
current_customer).select_related("content").order_by("-
id").values_list("content__id")

```

```

        digital_content_saves = Content.objects.filter(id__in =
digital_content_saves).select_related("project").select_related("proje
ct__authorhood").prefetch_related("categories").prefetch_related("cont
entmembers").prefetch_related("contentmembers__member").prefetch_relat
ed("leadstuff").prefetch_related("saves").annotate(Count("likes",
distinct=True)).annotate(Count("dislikes",
distinct=True)).annotate(Count("looks", distinct=True))
        return render(request,
"digcoapp/saved_digital_contents.html", {"contents":
digital_content_saves })
    else:
        return render(request,
"digcoapp/saved_digital_contents.html", {"contents": []})

def liked_digital_contents(request):
    try:
        current_customer = Customer.objects.get(user =
request.user)
    except Customer.DoesNotExist:
        current_customer = None
    if current_customer:
        digital_content_likes =
Contentlike.objects.filter(customer =
current_customer).select_related("content").order_by("-
id").values_list("content__id")
        digital_content_likes = Content.objects.filter(id__in =
digital_content_likes).select_related("project").select_related("proje
ct__authorhood").prefetch_related("categories").prefetch_related("cont
entmembers").prefetch_related("contentmembers__member").prefetch_relat
ed("leadstuff").prefetch_related("saves").annotate(Count("likes",
distinct=True)).annotate(Count("dislikes",
distinct=True)).annotate(Count("looks", distinct=True))
        return render(request,
"digcoapp/liked_digital_contents.html", {"contents":
digital_content_likes })
    else:
        return render(request,
"digcoapp/liked_digital_contents.html", {"contents": []})

def disliked_digital_contents(request):
    try:
        current_customer = Customer.objects.get(user =
request.user)
    except Customer.DoesNotExist:
        current_customer = None

```

```

        if current_customer:
            digital_content_dislikes =
Contentdislike.objects.filter(customer =
current_customer).select_related("content").order_by("-
id").values_list("content__id")
            digital_content_dislikes = Content.objects.filter(id__in
=
digital_content_dislikes).select_related("project").select_related("pr
oject__authorhood").prefetch_related("categories").prefetch_related("c
ontentmembers").prefetch_related("contentmembers__member").prefetch_re
lated("leadstuff").prefetch_related("saves").annotate(Count("likes",
distinct=True)).annotate(Count("dislikes",
distinct=True)).annotate(Count("looks", distinct=True))
            return render(request,
"digcoapp/liked_digital_contents.html", {"contents":
digital_content_dislikes })
        else:
            return render(request,
"digcoapp/disliked_digital_contents.html", {"contents": []})

def looked_digital_contents(request):
    try:
        current_customer = Customer.objects.get(user =
request.user)
    except Customer.DoesNotExist:
        current_customer = None
    if current_customer:
        digital_content_looks = Look.objects.filter(customer =
current_customer).select_related("content").order_by("-
time_fixed").values_list("content__id")
        digital_content_looks = Content.objects.filter(id__in =
digital_content_looks).select_related("project").select_related("proje
ct__authorhood").prefetch_related("categories").prefetch_related("cont
entmembers").prefetch_related("contentmembers__member").prefetch_relat
ed("leadstuff").prefetch_related("saves").annotate(Count("likes",
distinct=True)).annotate(Count("dislikes",
distinct=True)).annotate(Count("looks", distinct=True))
        return render(request,
"digcoapp/looked_digital_contents.html", {"contents":
digital_content_looks })
    else:
        return render(request,
"digcoapp/looked_digital_contents.html", {"contents": []})

def tag_digital_contents(request, tag):

```

```

    try:
        tag = Tag.objects.get(caption = tag)
    except Tag.DoesNotExist:
        tag _project = None
    digital_contents = Content.objects.filter(tags__contains =
tag, state =
1).prefetch_related("tags").prefetch_related("categories").annotate(Count("likes", distinct=True)).annotate(Count("dislikes",
distinct=True)).annotate(Count("looks",
distinct=True)).prefetch_related("contentmembers").prefetch_related("contentmembers__member").prefetch_related("contentmembers__role").order_by("-assessment")
        return render(request, "digcoapp/tag_digital_contents.html",
{"contents": digital_contents })

def category_digital_contents(request, category, content_type):
    try:
        category = Category.objects.get(name = category,
content_type = content_type)
    except Category.DoesNotExist:
        category = None
    digital_contents =
Content.objects.filter(categories__contains = category, state =
1).prefetch_related("tags").prefetch_related("categories").annotate(Count("likes", distinct=True)).annotate(Count("dislikes",
distinct=True)).annotate(Count("looks",
distinct=True)).prefetch_related("contentmembers").prefetch_related("contentmembers__member").prefetch_related("contentmembers__role").order_by("-assessment")
        return render(request,
"digcoapp/category_digital_contents.html", {"contents":
digital_contents })

def digital_content_assessments(request, digital_content):
    try:
        digital_content = Content.objects.get(id =
digital_content)
    except Content.DoesNotExist:
        digital_content = None
    if digital_content:
        assessments = Contentassessment.objects.filter(content =
digital_content).prefetch_related("customer").order_by("-time_fixed")
        return render(request,
"digcoapp/digital_content_assessments.html", {"assessments":
assessments })

```

```

def best_digital_content(request):
    digital_contents =
Content.objects.all().prefetch_related("leadstuff").prefetch_related("
saves").annotate(Count("likes",
distinct=True)).annotate(Count("dislikes",
distinct=True)).annotate(Count("looks", distinct=True)).order_by("-
assessment").values()[:100]
    return render(request, "digcoapp/best_digital_content.html",
{"contents": digital_contents })

def created_comments(request):
    try:
        current_customer = Customer.objects.get(user =
request.user)
    except Customer.DoesNotExist:
        current_customer = None
    if current_customer:
        created_comments = Comment.objects.filter(customer =
current_customer).select_related("content").select_related("content__p
roject").select_related("content__project__authorhood").order_by("-
time_fixed")
        return render(request, "digcoapp/created_comments.html",
{"comments": created_comments })
    else:
        return render(request, "digcoapp/created_comments.html",
{"comments": []})

def subscribed_digital_content(request):
    try:
        current_customer = Customer.objects.get(user =
request.user)
    except Customer.DoesNotExist:
        current_customer = None
    general_variant = self.request.GET.get('general')
    personal_variant = self.request.GET.get('personal')
    if general_variant and personal_variant:
        general_variant = False
    if current_customer and general_variant:
        subscription = Subscription.objects.filter(state = True,
customer = current_customer)
        if subscription:
            current_moment = datetime.datetime.now()
            digital_contents =
Content.objects.filter(access_type = 1, state = 1, access_until__gte =

```

```

current_moment).select_related("project").select_related("project__authorhood").prefetch_related("categories").prefetch_related("contentmembers").prefetch_related("contentmembers__member").prefetch_related("leadstuff").prefetch_related("saves").annotate(Count("likes", distinct=True)).annotate(Count("dislikes", distinct=True)).annotate(Count("looks", distinct=True)).order_by("-fixed_time")

        return render(request,
            "digcoapp/subscribed_digital_content.html", {"digital_contents":
digital_contents })
    else:
        return render(request,
            "digcoapp/subscribed_digital_content.html", {"digital_contents": []})
    elif current_customer:
        active_subscriptions =
Timepayment.objects.filter(customer = current_customer, state =
True).select_related("authortimepayment").select_related("authortimepayment__authorhood")
        subscribed_authorhoods = []
        for active_subscription in active_subscriptions:
            active_authorhood =
active_subscription.authortimepayment.authorhood
            subscribed_authorhoods.append(active_authorhood)
        active_content_subscriptions =
Contentpayment.objects.filter(customer = current_customer, state =
True).select_related("authorcontentpayment").select_related("authorcontentpayment__authorhood")
        for active_subscription in active_content_subscriptions:
            active_authorhood =
active_subscription.authortimepayment.authorhood
            subscribed_authorhoods.append(active_authorhood)
        nonactive_subscriptions =
Contentpayment.objects.filter(customer = current_customer, state =
False)
        subscribed_content = []
        for nonactive_subscription in nonactive_subscriptions:
            contents = Content.objects.filter(fixed_time__gte =
nonactive_subscription.fixed_time).order_by("fixed_time").values_list(
"id")[0:nonactive_subscription.authorcontentpayment.content_number]
            subscribed_content = subscribed_content + contents
        digital_contents = Content.objects.filter(state = 1,
access_until__gte = current_moment, Q(project__authorhood__in =
subscribed_authorhoods)|Q(id__in =
subscribed_content)).select_related("project").select_related("project__authorhood").prefetch_related("categories").prefetch_related("contentmembers").prefetch_related("contentmembers__member").prefetch_related("leadstuff").prefetch_related("saves").annotate(Count("likes", distinct=True)).annotate(Count("dislikes", distinct=True)).annotate(Count("looks", distinct=True)).order_by("-fixed_time")

```

```

tmembers").prefetch_related("contentmembers__member").prefetch_related
("leadstuff").prefetch_related("saves").annotate(Count("likes",
distinct=True)).annotate(Count("dislikes",
distinct=True)).annotate(Count("looks", distinct=True)).order_by("-
fixed_time")
        return render(request,
"digcoapp/subscribed_digital_content.html", {"digital_contents":
digital_contents })

    def best_authorhoods(request):
        authorhoods =
Authorhood.objects.all().prefetch_related("categories").prefetch_relat
ed("authorhoodmembers").prefetch_related("authorhoodmembers__member").
order_by("-assessment").values()[:100]
        return render(request, "digcoapp/best_authorhoods.html",
{"authorhoods": authorhoods })

    def category_authorhoods(request, category, content_type):
        try:
            category = Category.objects.get(name = category,
content_type = content_type)
        except Category.DoesNotExist:
            category = None
        authorhoods = Authorhood.objects.filter(categories__contains
=
category).prefetch_related("categories").prefetch_related("authorhoodm
embers").prefetch_related("authorhoodmembers__member").order_by("-
assessment")
        return render(request, "digcoapp/category_authorhoods.html",
{"authorhoods": authorhoods })

    def authorhood_assessments(request, authorhood):
        try:
            authorhood = Authorhood.objects.get(id = authorhood)
        except Authorhood.DoesNotExist:
            authorhood = None
        if authorhood:
            assessments = Authorassessment.objects.filter(authorhood
= authorhood).prefetch_related("customer").order_by("-time_fixed")
            return render(request,
"digcoapp/authorhood_assessments.html", {"assessments": assessments })

    def own_authorhood_assessments(request):
        try:

```

```

        current_customer = Customer.objects.get(user =
request.user)
    except Customer.DoesNotExist:
        current_customer = None
    if current_customer:
        assessments = Authorassessment.objects.filter(customer =
current_customer).order_by("-time_fixed")
        return render(request,
"digcoapp/own_authorhood_assessments.html", {"assessments":
assessments })

def own_content_assessments(request):
    try:
        current_customer = Customer.objects.get(user =
request.user)
    except Customer.DoesNotExist:
        current_customer = None
    if current_customer:
        assessments = Contentassessment.objects.filter(customer
= current_customer).order_by("-time_fixed")
        return render(request,
"digcoapp/own_content_assessments.html", {"assessments": assessments
})

def authorhood_projects(request, authorhood):
    try:
        authorhood = Authorhood.objects.get(id =
authorhood).prefetch_related("authorhoodmembers")
    except Authorhood.DoesNotExist:
        authorhood = None
    if authorhood:
        projects = Project.objects.filter(authorhood =
authorhood).order_by("-id")
        times = Authortimepayment.objects.filter(authorhood =
authorhood).values()[0:2]
        contents =
Authorcontentpayment.objects.filter(authorhood =
authorhood).values()[0:2]
        if projects.count() > 1:
            return render(request, "digcoapp/authorhood.html",
{"authorhood": authorhood, "projects": projects, "timepayments":
times, "contentpayments": contents })
        elif projects.count() == 1:

    else:

```

```

        return render(request, "digcoapp/authorhood.html",
{"authorhood": [], "projects": None, "timepayments": [],
"contentpayments": [] })

def subscribed_authorhoods(request):
    try:
        current_customer = Customer.objects.get(user =
request.user)
    except Customer.DoesNotExist:
        current_customer = None
    if current_customer:
        active_subscriptions =
Timepayment.objects.filter(customer = current_customer, state =
True).select_related("authortimepayment").select_related("authortimepa
yment__authorhood")
        subscribed_authorhoods = []
        for active_subscription in active_subscriptions:
            active_authorhood =
active_subscription.authortimepayment.authorhood
            if active_authorhood not in subscribed_authorhoods:
                subscribed_authorhoods.append(active_authorhood)
        active_content_subscriptions =
Contentpayment.objects.filter(customer = current_customer, state =
True).select_related("authorcontentpayment").select_related("authorcon
tentpayment__authorhood")
        for active_subscription in active_content_subscriptions:
            active_authorhood =
active_subscription.authortimepayment.authorhood
            if active_authorhood not in
                subscribed_authorhoods.append(active_authorhood)
        return render(request,
"digcoapp/subscribed_authorhoods.html", {"authorhoods":
subscribed_authorhoods })
    return render(request,
"digcoapp/subscribed_authorhoods.html", {"authorhoods":
subscribed_authorhoods })

def contentassessment_create(request, digital_content):
    digital_content = get_object_or_404(Content, pk =
digital_content).select_related("project").select_related("project__au
thorhood")
    try:
        current_customer = Customer.objects.get(user =
request.user)
    except Customer.DoesNotExist:

```

```

        current_customer = None
        digital_content_result = False
        subscription = None
        if current_customer:
            subscription = Subscription.objects.filter(state = True,
customer = current_customer)
            if digital_content.access_type == 0:
                digital_content_result = True
            elif digital_content.access_type == 1 and subscription:
                digital_content_result = True
            elif digital_content.access_type == 2 and
digital_content.project.authorhood.subscription_type == 2:
                active = Timepayment.objects.filter(customer =
current_customer, state = True, authortimepayment__authorhood =
digital_content.project.authorhood)
                if active:
                    digital_content_result = True
                else:
                    active_subscriptions =
Contentpayment.objects.filter(customer = current_customer,
authorcontentpayment__authorhood = digital_content.project.authorhood,
state = True)
                    if active_subscriptions:
                        digital_content_result = True
                    else:
                        active_subscriptions =
Contentpayment.objects.filter(customer = current_customer,
authorcontentpayment__authorhood = digital_content.project.authorhood)
                        for active_subscription in active_subscriptions:
                            contents =
Content.objects.filter(fixed_time__gte =
active_subscription.fixed_time).order_by("fixed_time")[
0:active_subscription.authorcontentpayment.content_number]
                            if digital_content in contents:
                                digital_content_result = True
                                break
            if current_customer and digital_content_result:
                if request.method == "POST":
                    assessment_form =
ContentassessmentForm(request.POST)
                    if assessment_form.is_valid():
                        assessment = assessment_form.save(commit =
False)

                        assessment.content = digital_content
                        assessment.customer = current_customer
                        assessment.comment.save()

```

```

        return
    HttpResponseRedirect(digital_content.get_absolute_url())
    else:
        assessment_form = None
        return render(request,
            "digcoapp/digital_content_assessment.html", {"form": assessment_form
        })
    else:
        return render(request,
            "digcoapp/digital_content_assessment.html", {"form": None })

    def authorassessment_create(request, digital_content):
        digital_content = get_object_or_404(Content, pk =
            digital_content).select_related("project").select_related("project__au
            thorhood")
        try:
            current_customer = Customer.objects.get(user =
            request.user)
        except Customer.DoesNotExist:
            current_customer = None
        if current_customer:
            active_subscriptions =
            Timepayment.objects.filter(customer = current_customer, state =
            True).select_related("authortimepayment").select_related("authortimepa
            yment__authorhood")
            subscribed_authorhoods = []
            for active_subscription in active_subscriptions:
                active_authorhood =
                active_subscription.authortimepayment.authorhood
                if active_authorhood not in subscribed_authorhoods:
                    subscribed_authorhoods.append(active_authorhood)
            active_content_subscriptions =
            Contentpayment.objects.filter(customer = current_customer, state =
            True).select_related("authorcontentpayment").select_related("authorcon
            tentpayment__authorhood")
            for active_subscription in active_content_subscriptions:
                active_authorhood =
                active_subscription.authortimepayment.authorhood
                if active_authorhood not in
                    subscribed_authorhoods.append(active_authorhood)
            if digital_content.project.authorhood in
            subscribed_authorhoods:
                if request.method == "POST":
                    assessment_form = AuthorassessmentForm(request.POST)
                    if assessment_form.is_valid():

```

```

        assessment = assessment_form.save(commit =
False)

        assessment.content = digital_content
        assessment.customer = current_customer
        assessment_comment.save()
        return
HttpResponseRedirect(digital_content.get_absolute_url())
    else:
        assessment_form = None
        return render(request,
"digcoapp/digital_author_assessment.html", {"form": assessment_form })
    else:
        return render(request,
"digcoapp/digital_author_assessment.html", {"form": None })

```

ДОДАТОК В
Слайди презентації

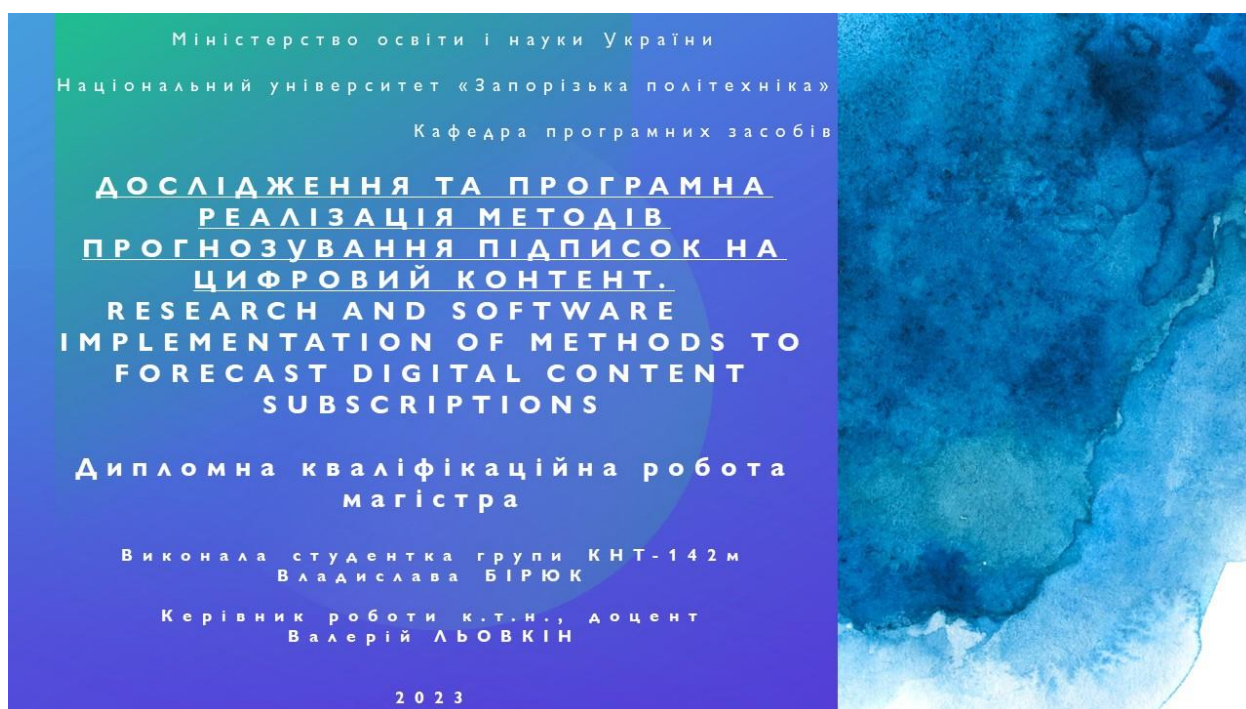


Рисунок В.1 – Слайд № 1

ОБ'ЄКТ, ПРЕДМЕТ І МЕТА ДИПЛОМНОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ

- Об'єкт роботи – процес прогнозування підписок на цифровий контент.
- Предмет роботи – методи прогнозування підписок на цифровий контент.
- Мета роботи – розробити програмне забезпечення прогнозування підписок на цифровий контент для надання і контролю доступу до цифрового контенту та короткострокового прийняття рішень на основі динаміки чисельності груп підписників.



Рисунок В.2 – Слайд № 2 (Об'єкт, предмет і мета дипломної кваліфікаційної роботи)

ЗАВДАННЯ ДИПЛОМНОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ

- проаналізувати існуючі вебзастосунки для надання доступу та керування доступом до цифрового контенту;
- розробити метод прогнозування підписок на цифровий контент;
- проєктування програмного забезпечення надання доступу та керування доступом до цифрового контенту;
- реалізувати вебзастосунок для надання доступу та керування доступом до цифрового контенту;
- виконати експериментальне дослідження прогнозування підписок на цифровий контент.

Рисунок В.3 – Слайд № 3 (Завдання дипломної кваліфікаційної роботи)

РЕЗУЛЬТАТИ АНАЛІЗУ ВЕБЗАСТОСУНКІВ ДЛЯ НАДАННЯ ДОСТУПУ ТА КЕРУВАННЯ ДОСТУПОМ ДО ЦИФРОВОГО КОНТЕНТУ

Критерій порівняння	MEGOGO	Patreon	Розробка
Підписка на авторський канал	–	+	+
Загальна підписка	+	–	+
Можливість поставити запитання автору	–	+	+
Багатокритеріальний пошук	+	–	+
Оцінювання цифрового контенту	+	–	+
Прогнозування кількості підписників	–	–	+

Рисунок В.4 – Слайд № 4 (Результати аналізу вебзастосунків для надання доступу та керування доступом до цифрового контенту)

МЕТОД ПРОГНОЗУВАННЯ ПІДПИСОК НА ЦИФРОВИЙ КОНТЕНТ

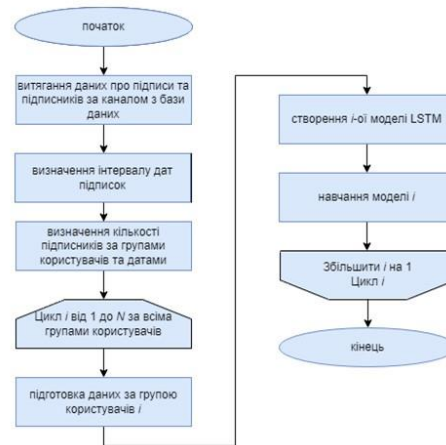


Рисунок В.5 – Слайд № 5 (Метод прогнозування підписок на цифровий контент)

ПОРІВНЯННЯ МОВ ПРОГРАМУВАННЯ ДЛЯ РЕАЛІЗАЦІЇ ПРОГРАМИ НАДАННЯ І КЕРУВАННЯ ДОСТУПОМ ДО ЦИФРОВОГО КОНТЕНТУ

Критерій порівняння / мова програмування	Python	C#
Система типізації	Динамічна	Статична
Ліцензія	Відкрита	Відкрита
Фреймворк розробки вебзастосунків	Django	ASP.NET
Простота імплементації існуючих рішень	++	+
Інструменти для роботи з даними	++	+—
Інтерфейс для роботи з нейронними мережами	Keras, Tensorflow	Keras.NET, Accord.NET, ML.NET
Підтримка в сфері машинного навчання	++	+—

Рисунок В.6 – Слайд № 6 (Порівняння мов програмування для реалізації програми надання і керування доступом до цифрового контенту)

ДІАГРАМИ ПРЕЦЕДЕНТІВ

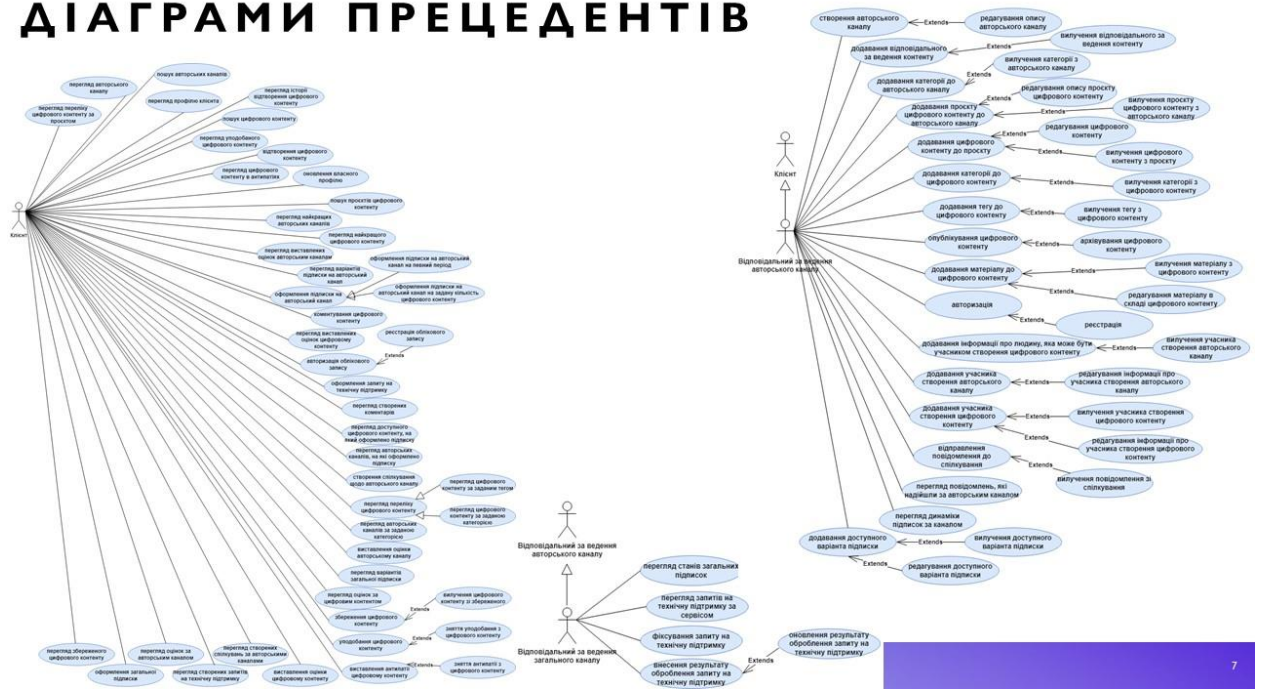


Рисунок В.7 – Слайд № 7 (Діаграми прецедентів)

СТРУКТУРА РЕЛЯЦІЙНОЇ БАЗИ ДАНИХ

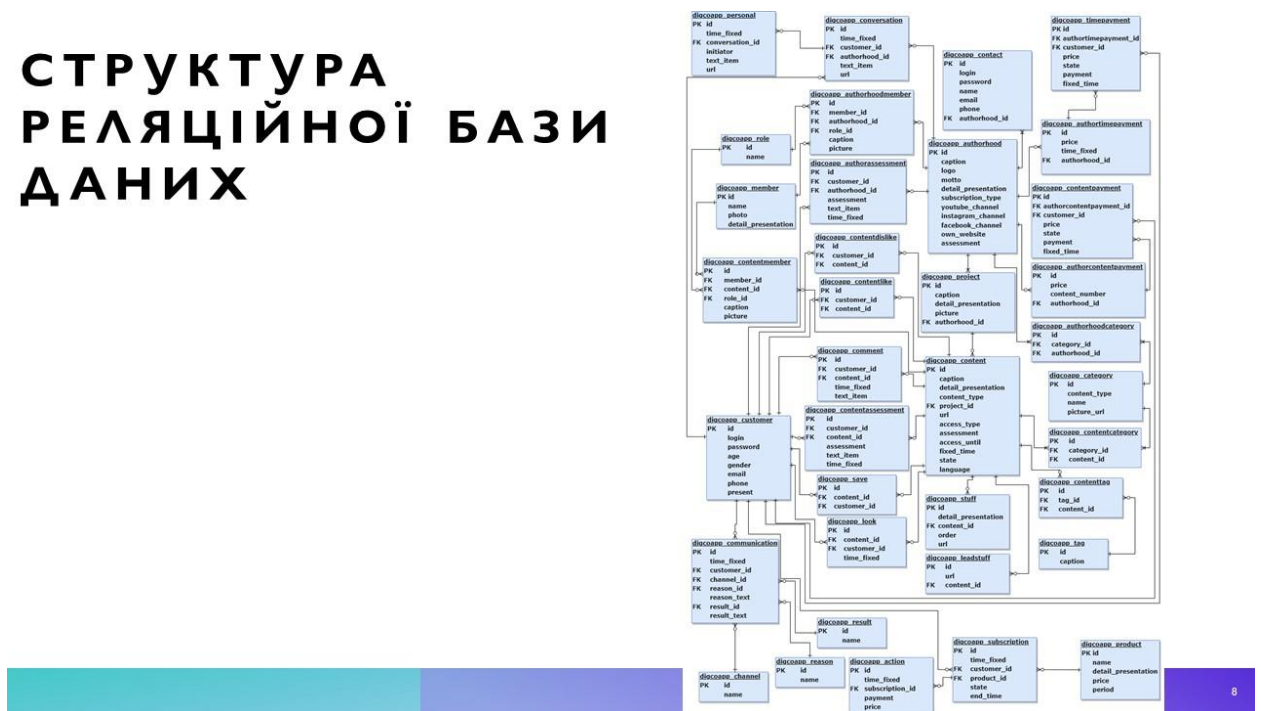


Рисунок В.8 – Слайд № 8 (Структура реляційної бази даних)

ПРОЦЕДУРА ПЕРЕВІРКИ ПРАВ ДОСТУПУ ДО ЦИФРОВОГО КОНТЕНТУ

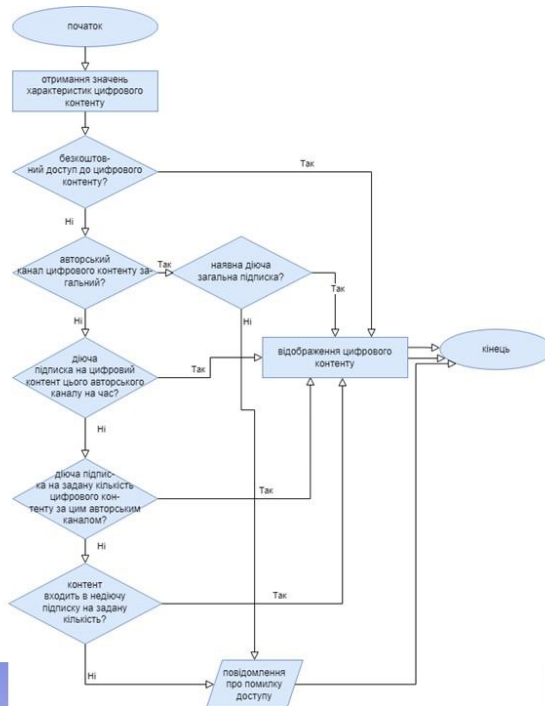


Рисунок В.9 – Слайд № 9 (Процедура перевірки прав доступу до цифрового контенту)

ІНТЕРФЕЙСИ ПРОГРАМИ

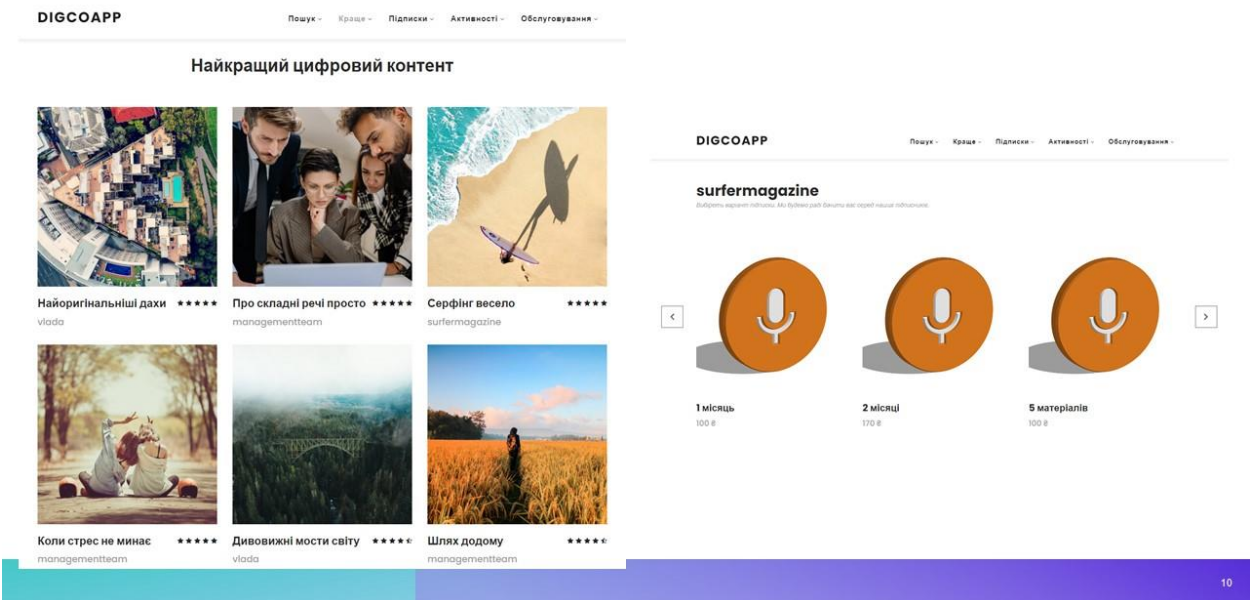


Рисунок В.10 – Слайд № 10 (Інтерфейси програми)

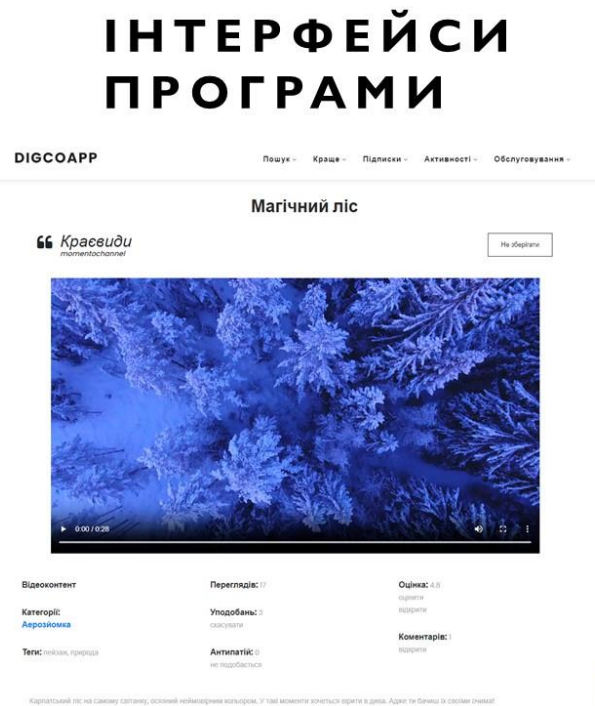
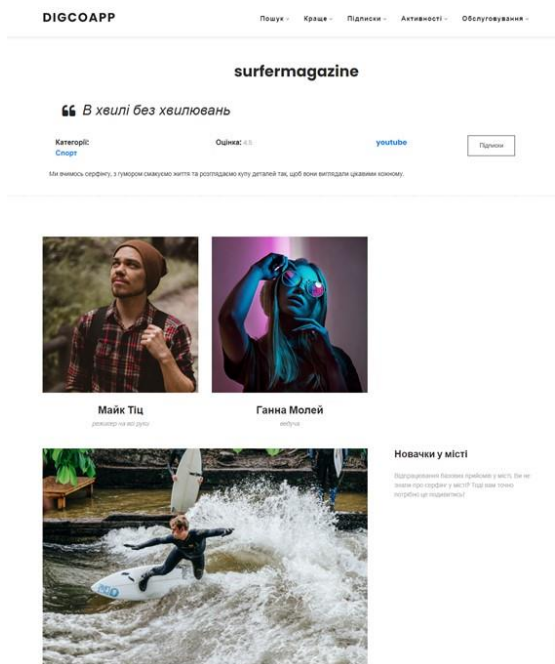


Рисунок В.11 – Слайд № 11 (Інтерфейси програми)

ІНТЕРФЕЙСИ ПРОГРАМИ

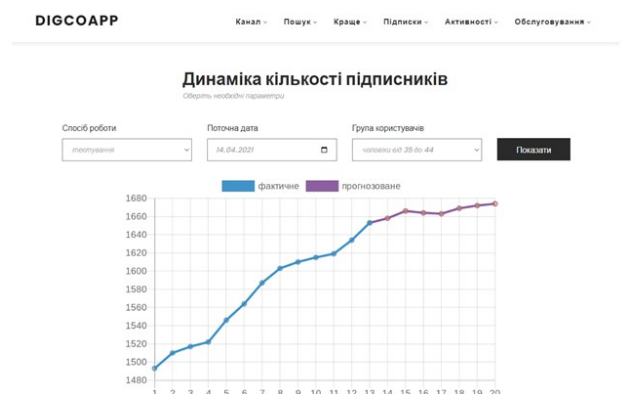
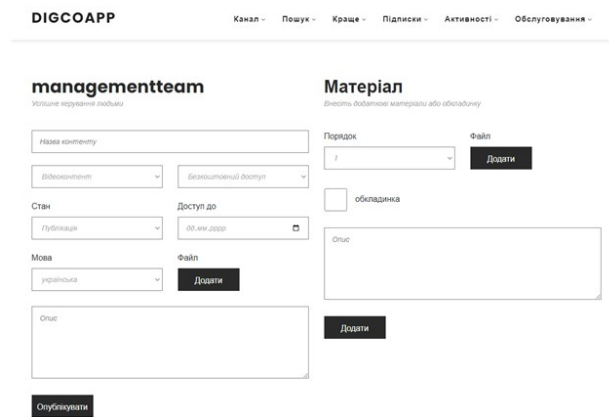


Рисунок В.12 – Слайд № 12 (Інтерфейси програми)

ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ

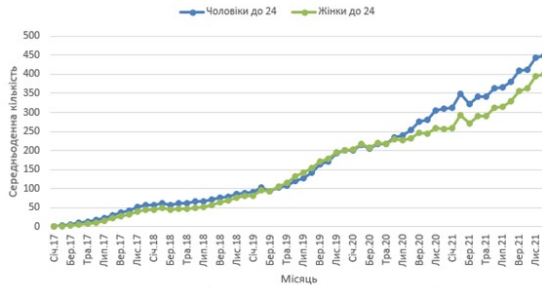


Рисунок 1 - Середньоденна кількість підписок користувачів до 24 років

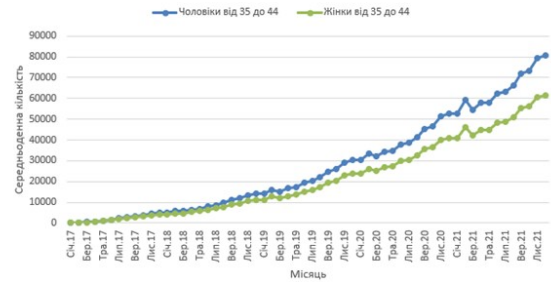


Рисунок 3 - Середньоденна кількість підписок користувачів від 35 до 44 років

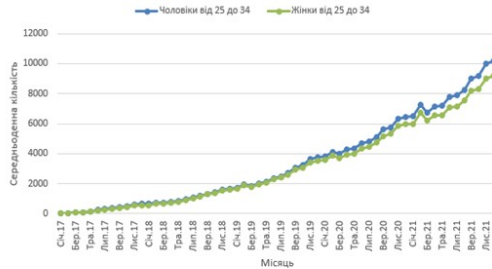


Рисунок 2 - Середньоденна кількість підписок користувачів від 25 до 34 років

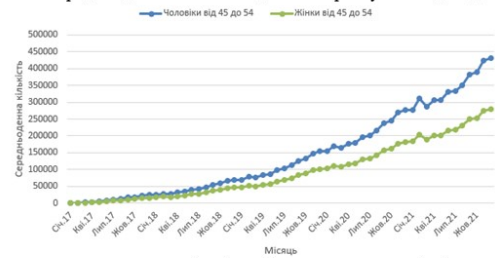


Рисунок 4 - Середньоденна кількість підписок користувачів від 45 до 54 років

13

Рисунок В.13 – Слайд № 13 (Експериментальне дослідження)

ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ

Модель та вікова група користувачів	Користувачі-чоловіки	Користувачі-жінки
LSTM (до 24 років)	0,0184	0,01977
LSTM (від 25 до 34 років)	0,05167	0,04046
LSTM (від 35 до 44 років)	0,04005	0,03563
LSTM (від 45 до 54 років)	0,04825	0,04811
LSTM (від 55 до 64 років)	0,05441	0,04466
LSTM (від 65 років)	0,05007	0,05887
ARIMA (до 24 років)	0,15061	0,14264
ARIMA (від 25 до 34 років)	0,12152	0,11681
ARIMA (від 35 до 44 років)	0,1074	0,10855
ARIMA (від 45 до 54 років)	0,11142	0,10559
ARIMA (від 55 до 64 років)	0,11264	0,11151
ARIMA (від 65 років)	0,11038	0,11701

14

Рисунок В.14 – Слайд № 14 (Експериментальне дослідження)

ВИСНОВКИ

- створене програмне забезпечення призначається для забезпечення роботи сервісів надання доступу до цифрового контенту централізованим чином і на основі створення окремих авторських каналів з можливістю надання цифрового контенту, забезпечення платних підписок на цей контент та контролю за станом підписок;
- наукова новизна роботи визначається методом прогнозування підписок на цифровий контент, який дозволяє створювати моделі прогнозування окремо для різних груп користувачів, виконуючи прогнозування кількості підписок у кожній групі користувачів на наступні дні на основі даних за попередні дні за допомогою використання нейронних мереж на основі LSTM;
- проведено експериментальне дослідження, в якому порівняно створені за допомогою запропонованого методу моделі для різних груп користувачів та моделі на основі ARIMA.

Рисунок В.15 – Слайд № 15 (Висновки)