

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЗАПОРІЗЬКА ПОЛІТЕХНІКА»

Факультет комп'ютерних наук та технологій
Кафедра комп'ютерних систем та мереж

Пояснювальна записка
до дипломного проекту (роботи)
магістра

(ступінь вищої освіти)

на тему КОМП'ЮТЕРНА СИСТЕМА ДЛЯ ДИСТАНЦІЙНОГО
НАВЧАННЯ З ІНТЕГРОВАНИМ ШТУЧНИМ ІНТЕЛЕКТОМ

Виконав: студент 2 курсу, групи КНТ-622м
спеціальності

123 Комп'ютерна інженерія

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Спеціалізовані комп'ютерні системи

КОСТЕЦЬКИЙ Д.В

(ПРИЗВИЩЕ та ініціали)

Керівник ТЯГУНОВА М.Ю.

(ПРИЗВИЩЕ та ініціали)

Рецензент КОЗІНА Г. Л.

(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет Комп'ютерних наук і технологій
Кафедра «Комп'ютерні системи та мережі»
Ступінь вищої освіти магістерський
Спеціальність 123 Комп'ютерна інженерія
(код і найменування)
Освітня програма (спеціалізація) «Спеціалізовані комп'ютерні системи»
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Зав. кафедри Кудерметов Р.К.

“24” жовтня 2023 року

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

КОСТЕЦЬКОГО Дмитра Віталійовича

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Комп'ютерна система для дистанційного навчання
з інтегрованим штучним інтелектом

керівник проєкту (роботи) к. т. н., доцент, ТЯГУНОВА Марія Юріївна
(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом вищого навчального закладу від “24” жовтня 2023 року № 400

2. Строк подання студентом проєкту (роботи) 10 грудня 2023 року

3. Вихідні дані до проєкту (роботи) аналіз вимог, специфікації завдань,
визначення ключових функцій, що формують основу для розробки та
реалізації системи.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1) Аналіз предметної області;

2) Проектування та моделювання системи;

3) Реалізація системи;

4) Тестування та впровадження проєкту.

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

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-5	ТЯГУНОВА М.Ю., доцент		
нормоконтроль	ЩЕРБАК Н.В., ст. викл.		

7. Дата видачі завдання 01.10.2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Ознайомлення з вимогами та завданнями проєкту, визначення ключових функціональностей, вибір інструментів розробки та архітектур.	12.10.2023 р.	
2	Дослідження впливу дистанційного навчання в порівнянні з звичайним.	17.10.2023 р.	
3	Проектування та моделювання системи	25.10.2023 р.	
4	Налаштування середовища та створення бекенду	10.11.2023 р.	
5	Реалізація фронтенду та тестування системи	22.11.2023 р.	
6	Оформлення пояснювальної записки	29.11.2023 р.	

Студент


(підпис)

Дмитро КОСТЕЦЬКИЙ
(ім'я, ПРИЗВИЩЕ)

Керівник проєкту (роботи)


(підпис)

Марія ТЯГУНОВА
(ім'я, ПРИЗВИЩЕ)

РЕФЕРАТ

ПЗ: 100 с., 78 рис., 1 табл., 38 джерел.

API, JS, LARAVEL, MYSQL, PHP, REACT, UML, БІБЛІОТЕКА,
ДИСТАНЦІЙНЕ НАВЧАННЯ, ДИСЦИПЛІНИ, КОМП'ЮТЕРНА СИСТЕМА,
НАВЧАННЯ, ТЕСТУВАННЯ, ФРЕЙМВОРК, ШТУЧНИЙ ІНТЕЛЕКТ

Об'єкт дослідження – комп'ютерні системи дистанційного навчання з інтегрованим штучним інтелектом.

Предмет дослідження – інструменти штучного інтелекту для автоматичного створення тестів.

Мета роботи – вдосконалення навчального процесу та підвищення зручності проведення освітнього процесу викладачам і студентам шляхом розроблення системи дистанційної освіти з можливістю створення тестів штучним інтелектом.

У першій частині проведено аналіз предметної області, який включає опис термінології системи, розгляд переваг та недоліків дистанційної освіти. Визначено специфічні вимоги до системи, функції користувачів, програмного продукту. Проведено вибір інструментів розробки, а саме фреймворку Laravel, бібліотеки React та проведено аналіз інструментів штучного інтелекту.

У другій частині виконано проєктування та моделювання системи, яке включало визначення архітектури системи, створення діаграм варіантів використання та послідовності. Також розглянуто вибір архітектури та проєктування серверної частини, а також розроблено ескізи інтерфейсу.

У третій частині описано реалізацію системи, як бекенду, так і фронтенду.

У четвертій частині було описано валідацію коду, тестування та розрахунок прискорення створення тестів, впровадження проєкту, оптимізацію та просування системи в мережі Інтернет.

ABSTRACT

Explanatory note to the master's work: 100 p., 78 figures, 1 table, 38 sources.

API, JS, LARAVEL, MYSQL, PHP, REACT, UML, LIBRARY, DISTANCE LEARNING, DISCIPLINES, COMPUTER SYSTEM, LEARNING, TESTING, FRAMEWORK, ARTIFICIAL INTELLIGENCE

Object of research - computer-based distance learning systems with integrated artificial intelligence.

The subject of research is artificial intelligence tools for automatic test creation.

The purpose of the study is to improve the educational process and increase the convenience of the educational process for teachers and students by developing a distance education system with the ability to create tests by artificial intelligence.

The first part of the paper analyzes the subject area, including a description of the system terminology, and a consideration of the advantages and disadvantages of distance education. Specific requirements for the system, user functions, and software product are identified. The selection of development tools, namely the Laravel framework, the React library, and an analysis of artificial intelligence tools were made.

In the second part, the system was designed and modeled, which included defining the system architecture, creating use case diagrams, and sequencing. The choice of architecture and design of the server part is also considered, as well as the development of interface sketches.

The third part describes the implementation of the system, both backend and frontend.

The fourth part describes code validation, testing, and calculation of test acceleration, project implementation, optimization, and promotion of the system on the Internet.

ЗМІСТ

Скорочення та умовні позначки	8
Вступ.....	9
1 Аналіз предметної області.....	11
1.1 Опис предметної області	11
1.1.1 Термінологія системи	11
1.1.2 Переваги та недоліки дистанційної освіти	13
1.2 Визначення вимог до системи	14
1.2.1 Визначення функцій користувачів системи	15
1.2.2 Вимоги до програмного продукту	16
1.2.3 Специфічні вимоги.....	18
1.3 Вибір інструментів розробки	21
1.3.1 Фреймворк Laravel	24
1.3.2 Бібліотека React	26
1.3.3 Аналіз інструментів штучного інтелекту	27
1.4 Аналіз інтеграції дистанційного навчання в освітні процеси	31
1.5 Висновки за розділом 1	35
2 Проєктування та моделювання системи	36
2.1 Проєктування системи.....	36
2.2 Моделювання системи.....	39
2.2.1 Діаграма варіантів використання	40
2.2.2 Діаграми послідовності	42
2.3 Вибір архітектури та проєктування серверної частини	46

	7
2.4 Проєктування інтерфейсу користувача та ескізів.....	49
2.5 Висновки за розділом 2.....	53
3 Реалізація системи.....	54
3.1 Створення бекенду	54
3.2 Створення фронтенду	64
3.3 Функціонування реалізованої системи	68
3.4 Висновки за розділом 3.....	75
4 Тестування та впровадження проєкту.....	76
4.1 Валідація коду.....	76
4.2 Функціональне тестування системи.....	78
4.3 Розрахунок економії часу при створенні тесту.....	83
4.4 Порівняння розробленої системи з Moodle	87
4.5 Оптимізація роботи системи	90
4.6 Просування системи в мережі Інтернет.....	92
4.7 Висновки за розділом 4.....	93
Висновки	94
Перелік джерел посилання	96

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

Бекенд	–	Частина системи яка відповідає за серверну обробку
Фронтенд	–	Клієнтська частина системи
ШІ	–	Штучний інтелект
API	–	Application Programming Interface
CORS	–	Cross-Origin Resource Sharing
CRUD	–	Create, read, update, delete
DOM	–	Document Object Model
GPT-	–	Generative Pre-trained Transformer
JSON	–	JavaScript Object Notation
MVC	–	Model-View-Controller
ORM	–	Object-Relational Mapping

ВСТУП

У наш неспокійний час, коли світ стикається з різними військовими конфліктами та епідеміями, дистанційне навчання стає першочерговим варіантом розв'язання проблеми освіти, оскільки воно надає можливість навчатися в укритті, або в будь-якому безпечному місці, де є інтернет, оминаючи відвідування фізичних занять у навчальних закладах, де може бути небезпечно або знаходиться потенційно хвора людина. Завдяки швидкому росту технологій та кількості комп'ютерів і телефонів, дистанційна освіта стає дедалі актуальнішою. У теперішній час одним із найвідоміших інструментів, який допомагає здобувати освіту в такий спосіб, є система Moodle. Це система керування дистанційним навчанням, з відкритим програмним кодом, який може використовувати будь-хто, безкоштовно.

Дистанційне навчання розвивається вже багато десятиліть, але найбільшу увагу було привернуто до нього, з початку поширення пандемії COVID-19, коли у всьому світі було введено карантинні обмеження. Завдяки цьому, процес інтеграції цього типу навчання у звичне очне було прискорено. Адже саме дистанційне навчання, завдяки його гнучкості, допомогло продовжити виконувати навчальний план та навчатись майже за графіком, який створюєш сам собі. Але перелічуючи переваги дистанційного навчання, в нього також є і негативні впливи на організм, а саме: погіршення зору, постави, психологічні аспекти.

Дистанційне навчання має багато переваг над звичайним відвідуванням аудиторій тим, що вивчати матеріали можна будь-де, а якщо щось було не зрозуміло, то можна переглянути його ще раз, також лекції й завдання можна завантажити на пристрій і тим самим навчатися без світла або інтернету. До переваг можна віднести те, що студентам не потрібно витратити час та кошти на те, щоб добратись до навчального закладу, і тим самим зменшується кількість викидів в атмосферу.

Економічно, віддалене навчання більш вигідне ніж традиційне, оскільки зменшує витрати на поїздки, проживання та обігрівання гуртожитку і закладу. Крім того, через це, багато програм дистанційного навчання більш доступні за ціною, ніж традиційні. Але недоліки теж присутні, у студентів зменшується комунікація з іншими людьми, псується зір через те, що впродовж великого відрізка часу перебуває перед екраном, хоча для запобігання цього, офтальмологи рекомендують робити спеціальні вправи.

Отже, перевагами дистанційного навчання можна назвати: гнучкий графік, можливість навчитись без світла та інтернету в укритті, доступність, бо для цього потрібен мінімум смартфон і економія коштів. Таким чином, обрана тема магістерської роботи є актуальною.

Мета роботи – вдосконалення навчального процесу та підвищення його зручності викладачам і студентам, шляхом розроблення системи дистанційної освіти з можливістю створення тестів за допомогою штучного інтелекту. Для досягнення цієї мети необхідно виконати наступні завдання:

- проаналізувати наявні аналогічні системи для дистанційної освіти, розглянути їх переваги та недоліки;
- визначити вимоги до розроблюваної системи;
- розробити архітектуру системи та описати взаємодію її складових;
- вибрати необхідне програмне забезпечення для реалізації системи;
- реалізувати систему з урахуванням визначених вимог і обраного програмного забезпечення;
- інтегрувати передові технології штучного інтелекту до системи, які нададуть новий функціонал;
- протестувати роботу системи та проаналізувати отримані результати.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис предметної області

У даній роботі, описано створений прототип системи для дистанційного навчання. Для цього, в першу чергу, необхідно провести виявлення цілей дистанційного навчання, існуючих робочих процесів і видів діяльності користувачів таких систем, діючих осіб та їх потреб, виконавців бізнес-процесів, визначити доцільність використання штучного інтелекту в такій системі.

Розроблювальна система буде поширюватися під назвою – Educational Site (Освітній сайт). Дана назва була обрана для того, щоб користувачам було легше знайти сайт, та для нових користувачів, одразу зрозуміло, для чого він використовується.

Даний програмний засіб створюється для того, щоб полегшити та популяризувати дистанційне навчання. Викладачі легко створюють тести та проводять лекції, учні мають до них доступ з пристроїв, користувачі деканату слідкують за успішністю студентів та можуть редагувати розклад.

Головною перевагою створеного програмного додатка буде те, що ним можна користуватись будь-де, через те, що його можна відкрити на всіх пристроях з інтернетом, у вільний час та не важливо, в якій частині світу.

1.1.1 Термінологія системи

Для початку визначимо терміни, які будуть використовуватися при створенні цієї комп'ютерної системи:

- посади: ректор, директор, декан, заступник декана, персонал деканату, завідувач кафедри, викладач і так далі [1];
- форми навчання: денна, заочна, скорочена, дистанційна;
- організаційна одиниця: факультет, кафедра;
- ступінь освіти: бакалавр, магістр, аспірант, аспірант, докторант [2];
- курс, староста курсу, група, куратор групи, староста групи, студент, абітурієнт [3];

- календарні періоди: семестр, модуль, кредити, часи, пари, залікова тиждень, екзаменаційна сесія [4];

- види навчальних робіт/занять: лекція, лабораторна робота, практична робота, консультація, курсовий проєкт, практика, наукова робота, дипломний проєкт, магістерська робота [5];

- документи студента: особиста справа студента, студентський білет, залікова книжка, диплом, виписка з диплома [6];

- основні документи: навчальний план, навчальна програма дисципліни, робоча програма, розклад занять, розпис екзаменів, відомість, зведена таблиця оцінок [7];

- види контролю: екзамен, залік, контрольна робота, тест, захист курсових/випускних робіт і так далі [8].

Розглянемо діючих осіб, які забезпечують виконання бізнес-процесів і будуть користувачами розроблюваної системи.

Викладач – це людина, яка проводить практичні, лабораторні або лекційні роботи, приймає курсові і дипломні проєкти.

Функції викладача:

- складання робочої програми;
- проведення лекцій і лабораторних робіт;
- проведення індивідуальної роботи зі студентом;
- навчання студентів за заздалегідь складеною робочою програмою і згідно з розкладом зайняття;

- введення відомостей відвідувань студентів, і заповнення зведеної таблиці оцінок;

- не допускати до іспиту або заліку студента із заборгованістю;

- проводити перевірку знань кожен модуль, і якщо предмет екзаменаційний, то з настанням сесії, провести іспит або виставити автомат за поточними оцінками.

Студент – це учень навчального закладу, який здобуває освіту.

Функції студента:

- виконання лабораторних робіт;
- відвідування лекцій;
- вивчення матеріалу наданого викладачем;
- демонстрація отриманих умінь і навичок на практиці;
- перевірка статистики успішності;
- стеження за графіком заліків, іспитів і консультацій;
- складання іспитів і заліків згідно зі складеним планом;
- можливість перездати предмет у випадку, якщо він не задоволений оцінкою;
- після закінчення навчання, повинен отримати диплом і виписку з диплома.

1.1.2 Переваги та недоліки дистанційної освіти

При дистанційному навчанні, комунікація між викладачем і студентом відбувається на відстані і ґрунтується майже за принципом самостійного навчання, цей вид дозволяє уникнути мінусів звичних форм навчання, людина почуває себе впевненіше, має тісніше спілкування з учителем, позбавляється від більшості психологічних бар'єрів в процесі комунікації і дає можливість вчитися в незалежності від часу або місце положення [8]. Ця форма може використовуватися при підвищенні діючої кваліфікації, за станом здоров'я або для осіб із фізичними обмеженнями.

При такому виді навчання, спілкування та обмін інформацією виконується за допомогою інтернету. Лекції викладаються віртуально, лабораторні виконуються в спеціальних програмах-симуляторах [9]. Ще одним з варіантів, лекції можуть бути збережені в відео форматі і можуть бути переглянутими у будь-який слухний час, так само як і тестування отриманих знань, може бути проведено в будь-який момент.

До привілей віддаленого навчання відносяться [10]:

- гнучкість навчатися;
- можливість поєднання навчання з професійною діяльністю;
- може бути не обмежена кількість слухачів при читанні лекції;

- оскільки навчання відбувається дистанційно, зникає потреба у використанні навчальних закладів, тим самим економляться кошти на їх опалювання і догляд, завдяки цьому зменшується плата за навчання і більша кількість людей може собі його дозволити;

- розвиваються самостійність, уміння критично мислити, можливість самодисципліни, відповідальності і наполегливості в досягненні мети, здатність приймати конструктивні рішення;

- можливість навчатися людям з обмеженими можливостями, або тим хто живе у віддалених місцях від міста.

Недоліками дистанційної освіти є:

- обмежені технічні можливості у зв'язку із застарілим обладнанням;
- менший контроль за учнями, тим самим вони можуть відволікатися в процесі навчання;
- недостатній рівень навичок користування комп'ютером;
- не можливість отримання знань при відсутності електроенергії або інтернету;
- нестача "живого" спілкування з викладачем і іншими студентами.

1.2 Визначення вимог до системи

Перед тим, як розпочинати реалізацію програмного забезпечення, спочатку потрібно визначити вимоги, які висуваються до розроблюваної системи.

Розроблювальна система повинна мати сторінки реєстрації та авторизації, де можна створювати нових користувачів (представників деканату, викладачів, студентів та адміністраторів), або увійти до облікового запису.

1.2.1 Визначення функцій користувачів системи

Користувачам присвоюються різні ролі, за якими вони мають доступ до певних функцій сайту. У системі існує чотири класи користувачів – студент, викладач, працівник деканату та адміністратор [11]. Студент може лише переглядати дані курсів, розклад і проходження тестів. Викладач та працівник деканату мають можливості редагування, додавання та видалення тих даних, до яких мають доступ. Адміністратор своєю чергою має всі права на роботу з будь-яким типом даних у базі даних.

Система повинна мати в собі такі сутності: користувач, предмет, розклад, група, тест, курс.

Також на сайті повинна бути інтеграція штучного інтелекту за допомогою якого викладач може генерувати питання для студентів в один клік.

Визначення вимог для студента. Студент повинен мати можливість:

- приєднання до курсу;
- перегляд теорії, або посилання для приєднання до онлайн лекції;
- проходження тестування;
- перегляд свого розкладу;
- перегляд своїх поточних результатів за складеними тестами;
- перегляд всіх учасників курсу;
- перегляд переліку вивчаючих курсів.

Визначення вимог для працівника деканату. Працівник деканату повинен мати можливість:

- вносити унікальний розклад для груп;
- редагувати існуючий розклад;
- переглядати курси;
- переглядати оцінки та результати тестування студентів.

Визначення вимог для адміністратора. Адміністратор повинен мати можливість:

- видавати ролі користувачам;

- слідувати за обліковими записами користувачів(видаляти тих хто зареєструвався з некоректними даними);
- мати доступ до всіх функцій (додавання/редагування/видалення), які мають учасники інших ролей.

Визначення вимог для викладача. Викладач повинен мати можливість:

- видаляти студентів з курсу;
- створювати теоретичний матеріал, або посилання на онлайн лекції;
- створювати тести;
- перевіряти результати за тестуванням;
- дивитись повну інформацію про оцінки студентів;
- перегляд розкладу за групами;
- перегляд учасників курсу;
- додавати свою фотографію та переглядати чужі.

1.2.2 Вимоги до програмного продукту

Розроблювальний програмний продукт є повністю незалежним та автономним.

Інтерфейс повинен мати адаптивну розмітку, для того, щоб користувачі могли користуватися сайтом з пристроїв, які мають різний розмір екрана, від маленького смартфона до великого телевізора чи проектора, який своєю чергою має підключення до глобальної мережі «Інтернет» [12]. Бути інтуїтивно зрозумілим, для того, щоб нові користувачі не витрачали час на пошук інформації, яка їм потрібна, бо вона повинна бути структурованою на сайті.

Розроблювальний продукт повинен бути оптимізованим, щоб не викликати довге завантаження на не потужних пристроях [13]. Все що потрібно від користувача – це встановлений браузер та якісний доступ до інтернету.

Від адміністраторів потрібно поповнювати рахунок до API якщо буде активно використовуватись штучний інтелект.

Система повинна виконувати наступні завдання:

- відображення матеріалів, або лекцій;
- відображення сторінки з проходженням тестування;

- відображення розкладу занять;
- відображення результатів тестів;
- внесення та редагування розкладу для груп;
- видавання ролей користувачам;
- додавання та видалення студентів з курсу;
- створення теоретичного матеріалу, або посилання для проведення онлайн лекції;
- створення та перегляд результатів тестувань;
- генерування тестів за допомогою ШІ.

Користуватися даним продуктом може кожен охочий, який має хоча б мінімальні навички користування комп'ютером або смартфоном. Завдяки інтуїтивному інтерфейсу, потенційні користувачі будуть зацікавлені в користуванні цим продуктом.

Розроблювальний продукт не обмежується майже нічим, окрім якісного зв'язку з інтернетом.

Доступ до головного функціоналу сайту надається лише зареєстрованим користувачам, а також тим кому був виданий доступ, зроблено з метою того, щоб не було вчинено ніяких поганих дій, у зв'язку з якими могли б постраждати інші користувачі.

Дані облікових засобів повинні зберігатися в зашифрованому вигляді, щоб у випадку отримання доступу до бази даних третіми особами, не було доступу до персональних даних користувачів.

Функціонування програмного засобу повністю залежить від роботи сервера на якому він розміщений, тому потрібно його розміщувати на серверах, які мають високий рейтинг серед інших його клієнтів [14].

Також залежність від доступності або зміни документації сайту, на якому використовується штучний інтелект.

Фактори які можуть вплинути на вимоги:

- збільшення кількості клієнтів;
- додавання функціонала, який не було обговорено на стадії проєктування;

- збільшення даних у базі даних;
- збільшення кількості користувачів;
- редагування існуючого функціоналу;
- розміщення на мало потужному сервері;
- закінчення токенів на арі штучного інтелекту.

Але навіть при проєктуванні програмного засобу неможливо передбачити все, тому при створенні прототипу можуть бути виявлені неточності, що дозволить доповнити новою інформацією вже існуючий опис.

1.2.3 Специфічні вимоги

Вхідні дані у розроблювальному програмному засобі формуються за допомогою викладачів, робітників деканату та адміністраторів. Вони мають спеціальний графічний інтерфейс, за допомогою якого вводять певну інформацію, наприклад викладач створює тест та вносить питання та варіанти відповідей до нього. Після збереження ці дані будуть відправлені до бази даних де надалі записані в певні таблиці. Після чого вони будуть відображені студентів під час тестування.

Розроблювальний програмний засіб будується за принципом клієнт-серверної технології, тому для взаємодії між ними, потрібно мати якісне з'єднання з інтернетом [15].

Функціональність інтерфейсу для різних груп відображається по різному, тому потрібно розглянути про кожну з них більш детально.

При першому візиті, користувача вітає сторінка, де на передньому плані відображаються курси. У верхній частині знаходиться вся навігація, що містить дві кнопки: "Авторизація" та "Реєстрація". Ці кнопки використовуються для відкриття сторінок, де розташовані форми введення даних. Для першого входу на сайт, користувач повинен спочатку здійснити реєстрацію, після якої виконати авторизацію, і йому відобразиться нова навігація, яка відповідає відповідній ролі у системі.

Для студента відобразиться поле, яке містить наступні сторінки:

- перегляду лекції – де відображатиметься посилання за допомогою якого буде можливість під'єднатися до онлайн лекції чи лабораторної роботи, або посилання на теорію у вигляді методичних вказівок;

- перегляду розкладу, на якій студент зможе побачити розклад групи на тиждень;

- проходження тестування, де студент продемонструє отримані знання за допомогою проходження тесту. При переході на тест, на сторінці відображено перелік питань та варіантів відповідей з яких потрібно вибрати вірний. Після повного проходження тесту, натиснути кнопку відправити, тим самим завершити тест та отримати оцінку за нього;

- перехід на головну сторінку, де Студент може переглянути, перелік предметів до яких він долучився.

Для цієї ролі на головній сторінці відображатимуться курси, до яких його було долучено.

Для викладача показано поле з наступними для переходу сторінками:

- створення нового предмету, де будуть знаходитись поля, за допомогою яких можна ввести назву предмета та обрати курс, до якого він відноситься;

- перегляд розкладу, де викладач матиме змогу подивитись свій розклад в інших групах за тижнями;

- перегляд оцінок, на якій викладач має змогу переглянути поточні результати проходження тестів студентами;

- створення теоретичної частини;

- створення тесту, як вручну, так і за допомогою ШІ.

Для цього користувача головна сторінка буде містити перелік створених ним курсів, при переході на які, відображатиметься меню з можливістю додавання нової інформації.

При переході на певний предмет, зможе додати теоретичний матеріал, посилання на онлайн лекцію, або нове тестування.

Для робітника деканату є можливість перейти на сторінку:

- створення розкладу, що являє собою форму, за допомогою якої буде можливо вносити розклад по групах;
- управління групами, де виведена інформація з усіма групами у вигляді таблиці. Також можливість редагувати, видаляти або додавати певну групу;
- управління курсами, де будуть знаходитись необхідні поля для створення, зміни або видалення.

Для цих користувачів на початковій сторінці буде відображено список курсів. При переході на які відображено перелік студентів долучених до нього.

Адміністратор буде бачити всі вище перераховані види сторінок, а також матиме додаткову:

- редагування, де може змінювати майже всю інформацію збережену в базі даних, а саме: групи, факультету, студента, викладача, предмету, розкладу і т. д. З цими даними можна проводити будь-які маніпуляції.

Ну і взагалі даний користувач матиме доступ до всіх функцій, які не мають інші ролі.

Система має витримувати великі навантаження, бо до неї одночасно може звернутися велика кількість різних користувачів. Та зі збільшенням даних, що зберігається, швидкодія системи не повинна сильно знижуватись.

База даних повинна бути повністю нормалізованою на стадії проектування, це допоможе у майбутньому зберегти пам'ять та швидкодію, бо у таблицях не буде повторюваних даних. Усі зв'язки повинні бути пов'язані між собою, щоб при видаленні, наприклад цілого курсу, за ним було видалено всі лекції та тести. Вона також повинна швидко обробляти запити, що надходять до неї шляхом створення спеціальних індексів [16].

Доступ до бази виконується за допомогою сервера, яким в свою чергу керують за допомогою вебдодатка.

Під час взаємодії між вебклієнтом та сервером передається унікальний ключ, який формується після авторизації та надсилається при кожному запиті до сервера. За його допомогою підвищується надійність системи та знижується шанс вторгнення до її роботи третіми особами.

Сервер повинен бути обладнаний системою, яка робить резервну копію бази через деякий час, це потрібно для того, щоб якщо в роботі сайту виникне якась непередбачена помилка, було реальним повернутись на стабільну версію та не втратити дані.

Також, у разі відмови одного з модулів, інші повинні продовжувати свою роботу.

При реєстрації дані користувача повинні бути зашифрованими, та зберігатися у базі даних у зашифрованому вигляді. Це робиться для того, щоб при отриманні доступу третіми особами, вони нічого не змогли б з ними зробити.

Після повної розробки даної системи, вона повинна легко піддаватись редагуванню існуючого функціоналу, або додавання нового.

Також можливість створення додаткового API, щоб деякі модулі даної системи можна було б підключити й використовувати в інших системах. Наприклад створення чат-бота, який би зміг сповіщати про новий тест, або те, що він скоро закривається, про подібну реалізацію було описано в моїй статті [17].

Для зручності, код програми може бути доповнений коментарями, які будуть лаконічно описувати функціональні можливості певних ділянок коду, реалізація яких не буде зрозуміла при поверхневому огляді.

1.3 Вибір інструментів розробки

Після аналізу предметної області та розробки технічного завдання, необхідно обрати інструменти, якими можна легко реалізувати цілі проєкту. Щоб зробити правильний вибір, важливо визначити критерії, за допомогою яких виконуватиметься їх оцінка. Наприклад, генерація клієнтської частини, яка може бути на сервері чи доповнюватись за допомогою асинхронних запитів, специфіка проєкту, мови програмування, безпека та рівень досвіду їх використання [18].

Без аналізу цих аспектів, складно зробити вірний вибір, оскільки ринок інструментів досить багатий. Але все ж він дуже важливий, і якщо зробити його неправильно, можна зіткнутися з проблемами в майбутньому, при розробці або додаванні нового унікального функціонала. Особливо коли вони схожі і виконують однакові функції у своїй категорії, і мають як переваги, так і недоліки в порівнянні з конкурентами що розглядаються.

На цей момент існує три популярні мови програмування для створення бекенду: PHP, JavaScript, та Python. Розглянемо коротко про кожну з них:

- PHP – має досить великий потенціал зі спільнотою розробників яка постійно збільшується, простий синтаксис, що полегшує її вивчення і легко дається новачкам [19]. Фреймворки, наприклад Laravel, Symfony, Yii, забезпечують швидку розробку та зручність використання. Однак у порівнянні з розглянутими мовами, PHP може виявитися менш продуктивним в окремих випадках, але у маленьких проєктах це не помітно;

- JavaScript – найпопулярніший інструмент, для написання фронтенду та бекенду. Для останнього використовується в модифікації Node.js, її асинхронність дозволяє ефективно виконувати операції з багатьма завданнями. Однак асинхронний код може виглядати складним, а відсутність єдиних стандартів викликає розбіжності у структурі проєкту;

- Python – мова програмування, яка за останній час набрала велику популярність завдяки великій кількості бібліотек, простому синтаксису та використанням у різних галузях, особливо в big data. Однак, якщо порівняти з і іншими мовами програмування, то вона поступається продуктивністю.

Проаналізувавши мови програмування, стало очевидним, що кращим вибором є PHP, бо вона сучасна, актуальна, має зрозумілий синтаксис, купу документації, та й в університеті було отримано досвід роботи з нею.

Тому обравши мову програмування, потрібно обрати до неї ефективний та потужний фреймворк, на якому буде вестись розробка, оскільки він визначає структуру, організацію проєкту, надає інструменти для ефективної розробки, і

грає важливу роль у розробці вебдодатка. Перед вибором, потрібно розглянути найпопулярніші з них:

- Laravel – є простим у використанні, має широкий спектр готових рішень, включаючи: маршрутизацію, ORM, шаблонізатор Blade, управління базою даних за допомогою міграцій та інше [20]. Його перевагами є: зручний та читабельний синтаксис, який полегшує роботу розробникам, активну спільноту, що гарантує підтримку та розвиток, а до недоліків можна віднести його меншу продуктивність в порівнянні з "легшими" фреймворками, але й вони позбавлені його переваг;

- Symfony – це високопродуктивний фреймворк, Використовуючи який, можна створювати вебзастосунки різної складності. Серед розробників прийнято, що Laravel використовується для додатків середньої складності, тоді як Symfony для великих і складних систем. Він базується на принципах компонентної архітектури і є ускладненою версією Laravel, через що для початківців є складнішим у вивченні [21];

- Yii – орієнтований на простоту використання, швидкодію та безпеку. Має ефективні механізми кешування та високий рівень безпеки. Але є вагомий недолік – менш гнучкий для нового функціоналу.

Провівши аналіз, виявилось, що для реалізації всіх поставлених завдань у цьому проєкті, бекенд частину найкраще реалізовувати за допомогою фреймворку Laravel, який використовує мову програмування PHP. Тепер розглянемо іншу частину розробки, а саме – фронтенд, бо вибір бібліотеки чи фреймворку для розробки клієнтської частини є також важливим, тому проведемо аналіз наступних існуючих інструментів:

- Angular – це фреймворк для розробки масштабних односторінкових додатків мовою JavaScript. Має велику функціональність та корпоративну підтримку з боку Google, що забезпечує надійність та активний розвиток, які зробили його одним із популярніших інструментів для фронтенд розробки [22]. Але іншого боку, вагомим недоліком Angular є його великий розмір, що впливає на швидкість завантаження сторінки для простих проєктів;

- Vue.js – це бібліотека для розробки клієнтської частини на мові JavaScript, надає простий інтерфейс для розробників будь-якого рівня кваліфікації, тому легкий та простий у вивченні. До її позитивних сторін, можна віднести простоту, легкість інтеграції в існуючі проєкти, проте в порівнянні з React та Angular, має меншу екосистему через що і меншу кількість бібліотек;

- React – це досить популярна бібліотека, за допомогою якої виконується розробка інтерфейсів мовою JavaScript. Через те, що вона працює з віртуальним DOM та компонентами, які дають можливість їх пере використання, ці два підходи – роблять її ефективним інструментом для створення швидких та динамічних інтерфейсів.

Тому, для фронтенду, було вирішено використовувати бібліотеку React, бо за її допомогою легко створюються продуктивні користувацькі інтерфейси, які розбиваються на компоненти для полегшення розробки та підтримки.

Такий комплексний підхід до вибору інструментів дозволить забезпечити оптимальну реалізацію поставлених завдань.

1.3.1 Фреймворк Laravel

При виборі мови програмування головним чинником було обрати ту, за допомогою якої буде не складно реалізувати систему, та легко її підтримувати у майбутньому. Порівнявши їх, було прийнято рішення зупинитись на PHP, бо ця мова є відносно легкою при розробці системи з нуля, часто оновлюється, і ми вивчали цю мову на минулих курсах, тому вже набуто практичні навички.

PHP – серверна інтерпретована мова програмування, тобто та, яка виконується відразу, без попередньої компіляції, й розроблялась для створення вебсторінок [23]. Перша версія вийшла у 1994 році й складалась з набору скриптів, які були написані мовою C, і використовувалась розробником для того, щоб виконувати підрахунок користувачів на сайті. Після посиленого доопрацювання і тестування, у 1997 році вийшла друга версія, якою були написані один відсоток сайтів. І вже у 2015 році була представлена 7 версія, яка була розроблена на основі експериментальної версії, яка розроблялась з нахилом

у сторону продуктивності. Найпопулярніша версія 7.4 була розроблена і представлена у 2019 році, в неї додали типізацію і стрілкові функції.

Також через її популярність, більшість хостинг-провайдерів за замовчуванням підтримують цю мову, тому це викликає великих проблем, при розміщенні сайту на сервері і його впровадженні.

Але на даний час, писати “чистою” мовою програмування недоцільно, бо для неї було створено різні фреймворки, під різні задачі.

Для мови PHP, початківцям рекомендують починати з фреймворку Laravel, бо він найлегший в вивченні, і в основному використовується для невеликих проєктів.

Laravel – це перший за популярністю фреймворк написаний мовою PHP [24]. Він використовується для написання бекенду і в собі вміщує в собі велику кількість механізмів, які користувач може використовувати протягом розробки, не встановлюючи ніякі додаткові бібліотеки.

Порівнявши Laravel з іншими фреймворками стало явним, що це краще рішення, бо він:

- надає купу готових інструментів, які б потрібно було самому реалізовувати при використанні “чистої” мови, наприклад: Eloquent ORM, який використовується для функціонального будування запитів які будуть надсилатись до бази даних, готові файли для маршрутизації, аутентифікації, шифрування, створення API;

- має чудову документацію, багато бібліотек та активну підтримку від інших розробників, тому в інтернеті можна знайти відповіді на всі питання;

- дотримується сучасних практик та використовує найпопулярніший архітектурний патерн для написання логіки;

- містить спеціальну мову шаблонізації Blade, за допомогою якої зазвичай роблять адміністративну панель, або навіть весь сайт.

За основу написання проєкту взято патерн проєктування MVC, бо за його допомогою розробникам легко зрозуміти, як працює той чи інший модуль, бо він написаний структуровано.

Цей патерн складається з наступних частин [25]:

- Model – це спеціальні класи які успадковуються від глобального класу Model, і відповідає за дані, тобто їх відображення або зберігання;
- View – це частина коду яка відповідає за візуалізацію даних отриманих з моделі і відображення клієнту;
- Controller – це такий собі набір правил. При отриманні запиту на сервер, спочатку переглядається список маршрутів і контролерів прикріплених за ними, після вибору останнього, він вже вирішує які моделі потрібно використовувати і за допомогою якого виду повинно бути виведено.

Таким чином за допомогою цих частин, код ділиться на логічні частини, які легко підтримувати та розробляти.

1.3.2 Бібліотека React

Перед тим, як перейти до розгляду і розуміння, що таке бібліотека React і де вона застосовується, потрібно спочатку зрозуміти, що таке взагалі фронтенд і з чого він може складатись.

Фронтенд – це клієнтська частина, тобто та, що відображається в браузері у вигляді користувацького інтерфейсу і в основному складається з наступних технологій [26]:

- HTML – використовується для створення структури вебсторінок;
- CSS – використовується для стилізування елементів, визначених у HTML;
- JS – використовується для створення динамічності на вебсторінках, валідації форм які використовуються для відправки даних на бекенд, або анімації.

JavaScript – мова програмування, яка була створена у 1995 році для розробки вебсторінок та додатків на яких присутні динамічні елементи, з якими може взаємодіяти користувач. З часом JS став стандартом для браузерів і отримав велику купу бібліотек створених розробниками [27]. З часом з мови для браузерів почав використовуватись для створення бекенд додатків за допомогою фреймворків.

Але розробляти складні сайти на “чистій” мові можливо, проте це складніше, займе більше часу та наступним розробникам буде складніше додавати щось нове, тому компанія Facebook створила свою власну бібліотеку React, яка дуже сильно полегшує розробку, бо складається з компонентів які можна легко пере використовувати. Вона має велику продуктивність завдяки тому, що працює з не статичним DOM, тому може швидко змінювати стани компонентів.

Через те, що React працює з копією DOM, виконується дуже потужна оптимізація рендеру сторінок, тому її використовують лише для розробки односторінкових сайтів, це ті, що підтримують зв'язок с бекендом за допомогою асинхронних AJAX викликів, без перезавантаження всієї сторінки, тоді як це зроблено в тих же шаблонах Blade, що сторінка генерується на бекенді і відправляється з нуля клієнту [28].

1.3.3 Аналіз інструментів штучного інтелекту

Розглянувши те, які з існуючих сервісів можна використовувати для розв'язання цієї проблеми, було з'ясовано, що більшість з них працюють на базі Chat GPT, яка використовує модель GPT-3 і розроблена компанією OpenAI.

Оскільки розроблювальна система буде складатись з клієнтської та серверної частини, важливою частиною роботи є правильний вибір варіанту взаємодії між ними. Нині у веброботці існує велика купа різних варіацій API, тому перед вибором розглянемо детально про кожний з них:

- restful API – є одним з найпопулярніших і широко застосованим методом взаємодії. Він заснований на принципах HTTP/HTTPS, і надає певний інтерфейс для обміну інформації між клієнтом і сервером. В ньому використовуються стандартні HTTP методи GET, POST, PUT, DELETE для виконання операцій над даними;

- GraphQL – являє собою мову запитів, яка дозволяє клієнтам отримувати лише необхідні дані й запобігати їх надмірності;

- WebSocket – дозволяє встановити постійний зв'язок між фронтендом і бекендом забезпечуючи миттєвий двосторонній зв'язок, але до його мінусів відносяться: високі витрати на інтернет, та навантаження на сервер.

Проаналізувавши ці методи взаємодії, було обрано RESTful API, бо зараз він найпопулярніший. Наступним був вибір структури даних, тобто в якому вигляді буде виконуватись передача інформації, розглянемо дві з найпопулярніших:

- XML – використовується для структурування інформації і використовується для її передачі між додатками. Раніше була дуже популярною, але на заміну їй прийшов більш сучасний формат під назвою JSON;

- JSON – це відносно новий формат, який прийшов на зміну XML та надає більш зручну структуру в використанні та в легкому представленні для читання людьми.

У системі дистанційного навчання, API відіграє важливу роль у забезпеченні зв'язку між сервером та клієнтом, а також надає можливість взаємодіяти з сервісами, такими як Chat GPT.

Функціональність API в дистанційному навчанні, наприклад:

- через неї можна передавати інформацію про користувачів, їхній прогрес, результати тестів та інші дані, як між платформою для навчання, так й інтегрувати роботу з месенджерами або поштовими сервісами;

- для створення тестів використовується штучний інтелект, який підтримує зв'язок за її допомогою, і може використовуватися для передавання запитів до моделі Chat GPT і отримання відповідей.

Коротко про те як виконується робота в API:

- перший етап взаємодії – це забезпечення безпеки запитів, шляхом аутентифікації та авторизації. Користувач або застосунок отримують доступ до API через унікальний ключ, що надається при успішній авторизації;

- після отримання ключа, застосунок надсилає запити, що містять дані або інструкції. Наприклад, для генерації тест III, додаток може надіслати запит до API з певними параметрами і ключем;

- після того, як сервер отримує запит, він буде оброблений, виконуються відповідні дії та повернеться відповіді, що містить запитану інформацію, або результати. Наприклад, для створення тесту, API може повернути згенерований текст тестових запитань у вигляді JSON.

API відіграє ключову роль при роботі між додатком і зовнішніми сервісами, забезпечуючи обмін даних і виконання функцій.

Chat GPT навчається з використанням глибокого навчання, що означає, що модель має багато шарів нейронів, які мають можливість будувати великі залежності з даних. Основою моделі є архітектура під назвою трансформер – це новий вид обробки тексту, при якому модель навчилася генерувати відповіді відштовхуючись від минулих відповідей. Моделі навчаються за допомогою великих масивів книжок, сторінок і статей взятих з відкритого доступу в інтернеті. Бо вона може розуміти контекст розмови, й коли приймає запит від людини, аналізує попередні повідомлення й формує відповідь.

Розглянемо більш детально про архітектуру трансформера, її використовують в різних задачах, наприклад для перекладу тексту, виділення головного з написаного, генерації тексту або сценарію і т. п. Через те, що дана архітектура вміє «пам'ятати» контекст, зробило її найпопулярнішим напрямом в розробці штучного інтелекту, бо завдяки цьому людина може вести бесіду, і навіть не розуміти, що це не людина. Розглянемо основні моменти, з чого вона складається:

- основна ідея – це увага, як і вже писалось вище, бо такий механізм дозволяє не втрачати ланцюг розмови;

- енкодер, який отримує на вхід векторну послідовність позиційної інформації, яка вказує на елементи початкової послідовності, і після чого застосовує до неї кілька шарів самоусвідомлення і повно зв'язних шарів, щоб згенерувати скрите уявлення послідовності. Декодер в свою чергу отримує це уявлення та вихід з енкодера, та також застосовує декілька попередньо описаних шарів, але до них додається механізм уваги, який допомагає декодеру, зосередитись на результаті;

- шар мережі – це основні блоки які виконують певні операції, які забезпечують навчання моделі і її роботу. До них можна віднести наступні основні типи: повно зв'язні – всі нейрони шару з'єднуються з всіма наступними нейронами, уваги – рахує ваги важливості кожного вхідного елементу, випадіння – нейрони вимикаються випадково, щоб не сталося перенавчання;

- через те, що трансформер не враховує порядок слів, потрібно виконувати спеціальне позиційне кодування для передавання індексу кожного елементу.

Розглянемо поверхнево роботу архітектури трансформера, моделі уваги.

Енкодер:

- кожне слово яке надходить з речення, перетворюється на числовий вектор;

- для кожного з них знаходиться подібність, і з цього створюється зважена копія;

- ці зважені копії підсумовуються, після чого виходить результуючий вектор для кожного слова. Цей процес повторюється з різними вагами, після чого надається різниця між критеріями схожості слів;

- вектор з результатом проходить через шар, який займається лінійним перетворенням, для того, щоб зменшити його розмірність;

- ці кроки повторюються декілька разів, щоб було створено декілька векторів для кожного слова.

Декодер:

- розпочинається все зі спеціального токена «start»;

- декодер звертається до попередньо закодованих слів, щоб зрозуміти контекст розмови;

- отримані дані нормалізуються та розраховується ймовірність наступного слова, після чого обране використовується як точка початку для наступного кроку;

- після чого, для кожного наступного слова, ці етапи виконуються у циклі, поки не буде отримано токен завершення «end of sentence».

Отже, розглянувши поверхнево цю архітектуру, було зрозуміло, що вона охоплює велику кількість областей де її можна застосувати. Головною перевагою є її механізм уваги, роблячи трансформер однією з головних архітектур нашого часу.

Повернемося до GPT-3, він має в собі інформацію з усього інтернету, до періоду 2022 року, бо в той час її навчили великим масивом даних, і з них вона може генерувати відповіді, за підрахунками її база знань складає 120 Гб даних.

GPT-4 є аналогічною, але її база знань більше через те, що вона зараз підключена до інтернету і навчається у реальному часі, проте ця версія моделі є платною і має певні ліміти на кількість запитів від користувача на хвилину. Зазвичай зараз дуже поширено використовують GPT-3 при створенні текстів для постів, книг, вона показує задовільні результати при інтеграції у чат-боти або текстового обслуговування клієнтів, бо інколи складно відрізнити, хто насправді веде з тобою розмову.

Також сервіс, на якому розміщено цей штучний інтелект, надає спеціальні API, за допомогою яких можна легко підключити його до будь-якої системи, що і планується зробити для автоматичної генерації тестів за темою й описом.

1.4 Аналіз інтеграції дистанційного навчання в освітні процеси

В Україні існує спеціальна державна служба, яка займається оцінкою якості освіти й існує вона з 2017 року. Оскільки дистанційна освіта активно розвивається останні 3 роки, протягом цього часу, служба провела велику кількість опитувань та досліджень. Опитування є анонімними та проводяться за допомогою онлайн-форм, які розраховані на людей, які беруть участь у навчальному процесі з різних куточків країни. За їх допомогою аналізується зібрана інформація та надаються пропозиції для покращення навчального процесу [29].

У зв'язку з початком карантину і переведенням всіх навчальних закладів зі звичного очного на повноцінне дистанційне навчання, було стресом для всіх, хто пов'язаний з навчальним процесом. У квітні 2021 року, після завершення першого повноцінного навчального року, проведеного у змішаній формі, відбулось опитування серед представників адміністрації, персоналу та здобувачів освіти за 2020-2021 навчальний рік, усього їх було 9184 осіб. Темою опитування було оцінити рівень організації дистанційного формату за допомогою шкали від 1 до 5. Його результат зображено на рисунку 1.1 [30]. Видно, що середньому від дистанційного навчання немає негативного впливу на рівень освіти.

Оцінка респондентами результатів навчання

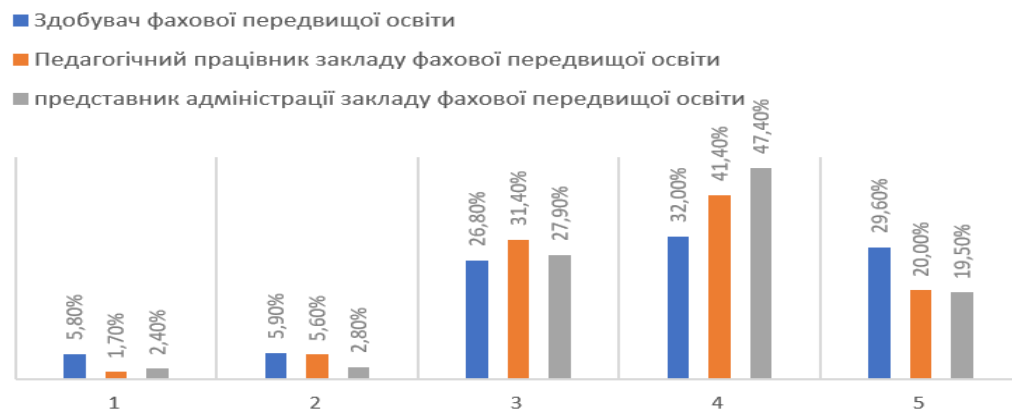


Рисунок 1.1 – Оцінка дистанційного навчання

У травні 2022 було проведено нове опитування в вищих навчальних закладах, і отримано 27 тисяч анкет за підсумками 2021-2022 року навчання, в яких було оцінено якість змішаного навчання на дистанційного та під час воєнного стану, на рисунку 1.2 зображено результат [31]. Проаналізувавши його можна зрозуміти, що переважна кількість опитаних задоволені навчанням, навіть в такий нелегкий для країни час. Все це було досягнуто внаслідок того, що навчальні заклади вже мали досвід дистанційного навчання і це не було додатковим стресом для людей.



Рисунок 1.2 – Рівень задоволення здобувачів освіти та викладачів в дистанційному режимі 2022 році.

Протягом 2023 року було проведено велику кількість різних опитувань в яких взяло участь 40 тисяч людей. Серед них головними були про: формати навчання, оцінку форм навчання, оцінку знань в порівнянні з 2022 роком.

Отже, першим розглянемо те як змінювались форми навчання по всій Україні. На рисунку 1.3 зображено, що через те, що ситуація в країні в другому кварталі відносно стабілізувалась в порівнянні з першим, але, як бачимо, навчальні заклади віддають перевагу змішаному формату, бо він відносно безпечний через дію надзвичайного стану в країні, і через це частка очної форми знижується [32].

Наступним було проведено опитування викладачів, для того, щоб вони порівняли рівень знань учнів II півріччя 2022-2023 року з показниками попереднього року. На рисунку 1.4 показано, що 30% викладачів відзначили, що рівень знань погіршився, і 20% вказали що покращилось. Але можна вказати на те, що зниження рівня може бути через те, що в здобувачів освіти, з'явилась купа чинників, які перешкоджають навчанню, серед них: тривоги, під час яких зупиняється робота більшості закладів, та те, що треба перебувати в цей час в укритті, вимкнення світла, через атаки важливої інфраструктури росією. Тому спеціалісти кажуть, що попри всі складнощі, заклади функціонують та надають знання відносно стабільно.

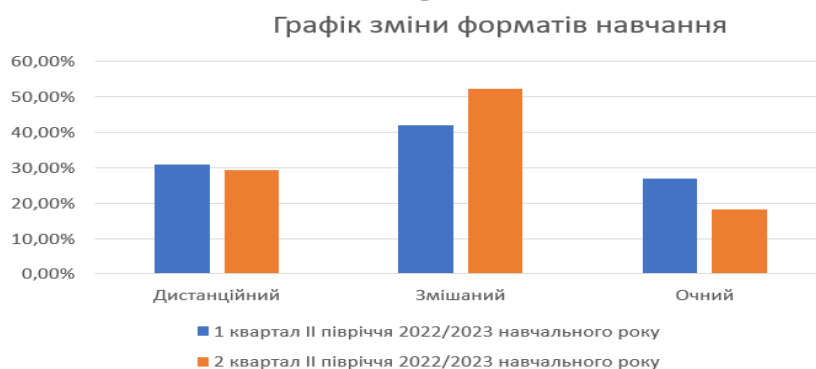


Рисунок 1.3 – Графік зміни формату навчання



Рисунок 1.4 – Рівень знань учнів в порівнянні з попереднім навчальним роком

Ну і так само як проводили опитування у 2021 та 2022 на тему оцінки форм навчання, 2023 рік не є особливим. На рисунку 1.5 зображено результат, на якому бачимо, що для здобувачів та працівників освіти різниці між форматами проведення майже немає, бо наші навчальні заклади пристосувалися до різних форм навчання та виконують свою роботу максимально професійно.

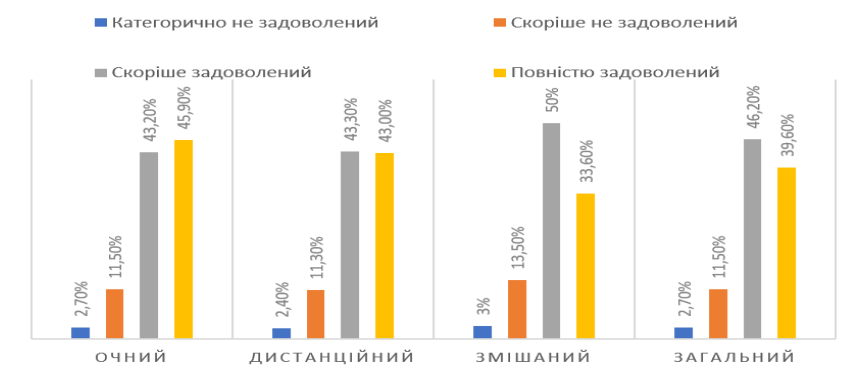


Рисунок 1.5 – Рівень задоволеності в дистанційному форматі 2022-2023 році

1.5 Висновки за розділом 1

Підбиваючи підсумок розділу, було проведено дослідження предметної області, пов'язаної із темою магістерської роботи “Комп’ютерна система для дистанційного навчання з інтегрованим штучним інтелектом”.

Проаналізувавши предметну область системи дистанційного навчання та дослідивши її функціональність, можна зробити кілька ключових висновків. Перш за все, система повинна використовувати передові технології, які сприятимуть розвитку нового типу навчання, а саме дистанційної освіти. Надасть можливість безперешкодного доступу до навчальних матеріалів з пристрою за умови наявного доступу до мережі Інтернет. Дослідивши, як здійснюється дистанційна освіта, було виявлено переваги у вигляді гнучкості графіку та доступності навчання, але є й недоліки у вигляді обмеження "живого" спілкування між студентами та викладачами.

Було визначено термінологію системи, ролі та процеси, які відбуваються в середовищі дистанційної освіти, чітко розмежовано можливості викладачів та студентів.

При аналізі вимог до програмного продукту було визначено ключовий функціонал та особливості системи. Були розглянуті вимоги до користувачів з різними ролями, визначено базовий функціонал, необхідність адаптивного інтерфейсу, оптимізації продукту, безпеки даних та можливості розширення в майбутньому.

При проєктуванні необхідно буде врахувати можливі зміни вимог при зростанні кількості користувачів та можливим розширенням функціонала, адже проєктування системи – це безперервний процес, оскільки під час її розробки можуть легко з'явитися нові вимоги.

При аналізі вибору інструментів для розробки системи максимальну увагу було приділено наступним критеріям: специфіка проєкту, мова програмування, безпека та досвід. Результатом цього процесу став вибір наступних технологій:

фреймворк Laravel для бекенду та бібліотека React для фронтенду. Обрані інструменти мають чудовий потенціал для ефективної розробки проєкт та сприятимуть подальшому розвитку системи завдяки своїм перевагам та підтримці. Розглянуто сервіси, які надають зручне API, за допомогою якого можна легко взаємодіяти зі штучним інтелектом просто надсилаючи йому певні команди.

Провівши детальний аналіз впливу дистанційного навчання за роками, було зрозуміло, що це перспективний напрям, бо за останні роки воно зарекомендувало себе, як гарна альтернатива фізичному відвідуванню занять.

Отже, розроблена система повинна мати високий потенціал для покращення дистанційної освіти та доступу до структурованих навчальних матеріалів у віртуальному навчальному середовищі. Оскільки її метою є створення безпечної та ефективної системи з використанням передових технологій.

2 ПРОЄКТУВАННЯ ТА МОДЕЛЮВАННЯ СИСТЕМИ

2.1 Проєктування системи

В процесі проєктування програмного засобу не завжди можна передбачити всі аспекти, тому під час створення прототипу можуть виявитися неточності. Це надає можливість доповнити наявний опис новою інформацією та позбутися від них при створенні повноцінної системи.

В розроблювальній системі вхідні дані наповнюються всіма учасниками, бо у них є спеціальні інтерфейси, через які вони вносять інформацію. Наприклад, викладач може створити тест, вручну або використовуючи штучний інтелект, який буде передано до бази даних, з наступним збереженням в таблицях, дотримуючись правил нормалізації. Під час складання тесту студент зможе переглядати ці дані, які будуть відображатися після обробки бекендом.

Розроблювальна система ґрунтується на принципі клієнт-серверної технології, що передбачає обмін даними між клієнтом та сервером. З цього випливає необхідність якісного інтернет-з'єднання для роботи програми.

При першому вході на сайт, користувачеві відкривається головна сторінка сайту, де на передньому плані відображається перелік наявних курсів. У лівій частині розташоване навігаційне меню з двома кнопками: авторизація та реєстрація, за їх допомогою виконується перенаправлення на сторінки, які матимуть свої форми. Для того, щоб зайти до сайту, першим ділом потрібно зареєструватись. Зареєструвавшись, і виконавши авторизацію, відповідно до ролі, буде відображено навігаційне поле.

Для студента буде доступне навігаційне поле з посиланнями на сторінки:

- перегляд курсу, де представлено посилання, за допомогою яких виконується приєднання до онлайн лекцій, лабораторних робіт або перегляду теоретичного матеріалу;

- перегляд розкладу, який зображено на весь тиждень і доступний для перегляду, надаючи інформацію про графік занять;

- проходження тестування, де є можливість перевірити свої знання, запустивши тест, в якому питання відображаються по черзі, і після завершення тестування показується фінальна оцінка;

- перегляд переліку курсів, які він вивчає.

Для цієї ролі, початкова сторінка покаже перелік курсів, до яких студент приєднаний, і перехід на кожен курс виведе перелік методичних вказівок та тестів, які можна пройти.

Для викладача відображено буде доступно:

- створення предмета, на якій доступні поля з такими назвами: назва предмета, опис;

- перегляд розкладу, де є можливість переглядати розклад груп на тиждень, що надає повний огляд графіка навчальних заходів;

- перегляд оцінок, де можна переглядати оцінки, отримані студентами після складання тестування;

- створення тесту, на якій є можливість створювати тести вручну або генерувати їх з використанням ШІ.

На головній сторінці викладачу представлений список курсів, до яких він приєднався. При переході на будь-який створений ним курс, буде доступний функціонал, де він матиме можливість додавати освітній матеріал, запрошення на онлайн лекцію або створити тест.

Для працівника деканату можливість перейти на:

- створення розкладу, де можна створити розклад, додаючи інформацію відповідно до груп;
- управління групами, яка містить інформацію про всі групи, а також опції редагування, видалення або додавання нових;
- управління курсами, де можна керувати курсами, включаючи створення, редагування та видалення.

Для адміністратора доступні всі сторінки, а також:

- редагування, де доступна зміна всієї інформації, яка зберігається у базі даних, такої як: група, студент, викладач, предмет, розклад і т. д. Виконуючи: редагування, додавання та видалення, що забезпечує повний контроль над інформацією в системі.

Адміністратор має повний доступ до всіх функціональних можливостей, включаючи ті, до яких для звичайних користувачів доступ закритий.

Система повинна бути готовою до збільшення користувачів та обробки значної кількості запитів в короткий період часу. З ростом аудиторії, обсяг збереженої інформації також збільшується. При правильному проєктуванні має забезпечуватись стійкість та швидкодія системи при зростанні навантаження.

Для збереження пам'яті та швидкодії у майбутньому, на стадії проєктування бази даних, її потрібно нормалізувати, бо у таблицях не повинні містити повторювані дані, і оскільки всі зв'язки між ними взаємопов'язані, при видаленні курсу, автоматично буде видалено пов'язані з ним записи.

За керування базою даних відповідає бекенд, на який впливають за допомогою фронтенду. Тобто база не обов'язково повинна бути на тому ж

самому сервері, але якщо це так, то швидкість запитів буде довше на кілька сотих секунди.

Для захищеного зв'язку фронтенду з бекендом, в запитах додається спеціальний ключ, який автоматично генерується після кожної авторизації й використовується в якості ідентифікації користувача при зверненні до сервера, що забезпечує високий рівень надійності та зменшує ймовірність несанкціонованого доступу.

Сервер повинен мати функціональність для регулярного створення резервних копій даних. Це необхідно для того, щоб в разі фатальної помилки, була можливість відновити цілісність даних і уникнути їх втрати.

Також потрібно приділити велику увагу даним користувача які він вносить при реєстрації, зокрема його пароллю. Забезпечення безпеки здійснюється шляхом зберігання пароля в зашифрованому вигляді. Це важливий захід, призначений для того, щоб при зломі бази даних, ніхто не міг користуватись даними користувачів.

Система містить в собі наступні типи користувачів: студент, викладач, працівник деканату та адміністратор. Студент має право на перегляд: існуючих курсів, та тих на які виконана підписка, розкладу, теоретичних матеріалів та проходження тестів. Викладач та працівник деканату мають можливості створювати, змінювати та видаляти та дані, які відповідають їхнім ролям. Адміністратор має повні права, та може взаємодіяти з будь-яким даними, які зберігаються у системі.

2.2 Моделювання системи

Моделювання системи – це один з найголовніших етапів при проєктуванні програмного забезпечення. На цьому етапі виконується аналіз того, як майбутній продукт використовуватиметься, з детальним описом дій. Також в цьому розділі

визначаються всі користувачі (актори), та дії які вони будуть виконувати (прецеденти).

Побудова цих діаграм виконується за допомогою спеціальної мови моделювання – UML. Дана мова була обраною через те, що вона стандартизована, і схему описану за правилами зрозуміє кожен, хто з нею працює.

Результатом даного моделювання буде отримано діаграми варіантів використання та послідовності. За отриманими результатами можна спокійно переходити до проектування архітектури програмного засобу і його реалізації.

2.2.1 Діаграма варіантів використання

Опишемо спочатку розроблювану систему на концептуальному рівні за допомогою діаграми варіантів використання. Вона дозволить відобразити взаємодію сутностей та акторів проекту з прецедентами [33]. Діаграма дій для модельованої системи зображена на рисунку 2.1.

З даної діаграми можна виділити наступне:

- користувачі: адміністратор, працівник деканату, студент, викладач та для того, щоб було більш зручно, було створено ще одного актора під назвою користувач, який буде об'єднувати в собі працівника деканату, студента та викладача. Також на діаграмі присутній актор з назвою система, який приймає запити від інших;

- прецеденти: вхід до системи, CRUD операції над даними, перегляд оцінок, перегляд розкладу, управління курсом, управління предметом, вивчення курсу.

Перейдемо до короткого опису кожного з описаного вище прецеденту:

- вхід до системи – описує авторизацію, і також передбачається можливість реєстрації або виведення помилки у разі введення некоректних даних;

- CRUD операції над даними – описує те, як адміністратор виконує роботу над інформацією, а саме редагування, створення та видалення записів;

- перегляд оцінок – описує роботу того, як користувач може переглядати інформацію пов'язану з оцінками;

- перегляд розкладу – користувач може ознайомитись з розкладом своєї групи;
- управління курсом – описує те, як працівник деканату може керувати курсом, також розширюється можливістю керування групами, які доповнюються керуванням розкладу;
- управління предметом – описує те, як викладач може керувати предметом, який розширюється можливостями управління теорією та тестами;
- вивчення курсу – описує те, як студент може вивчати курс, теорію та виконання тестів.

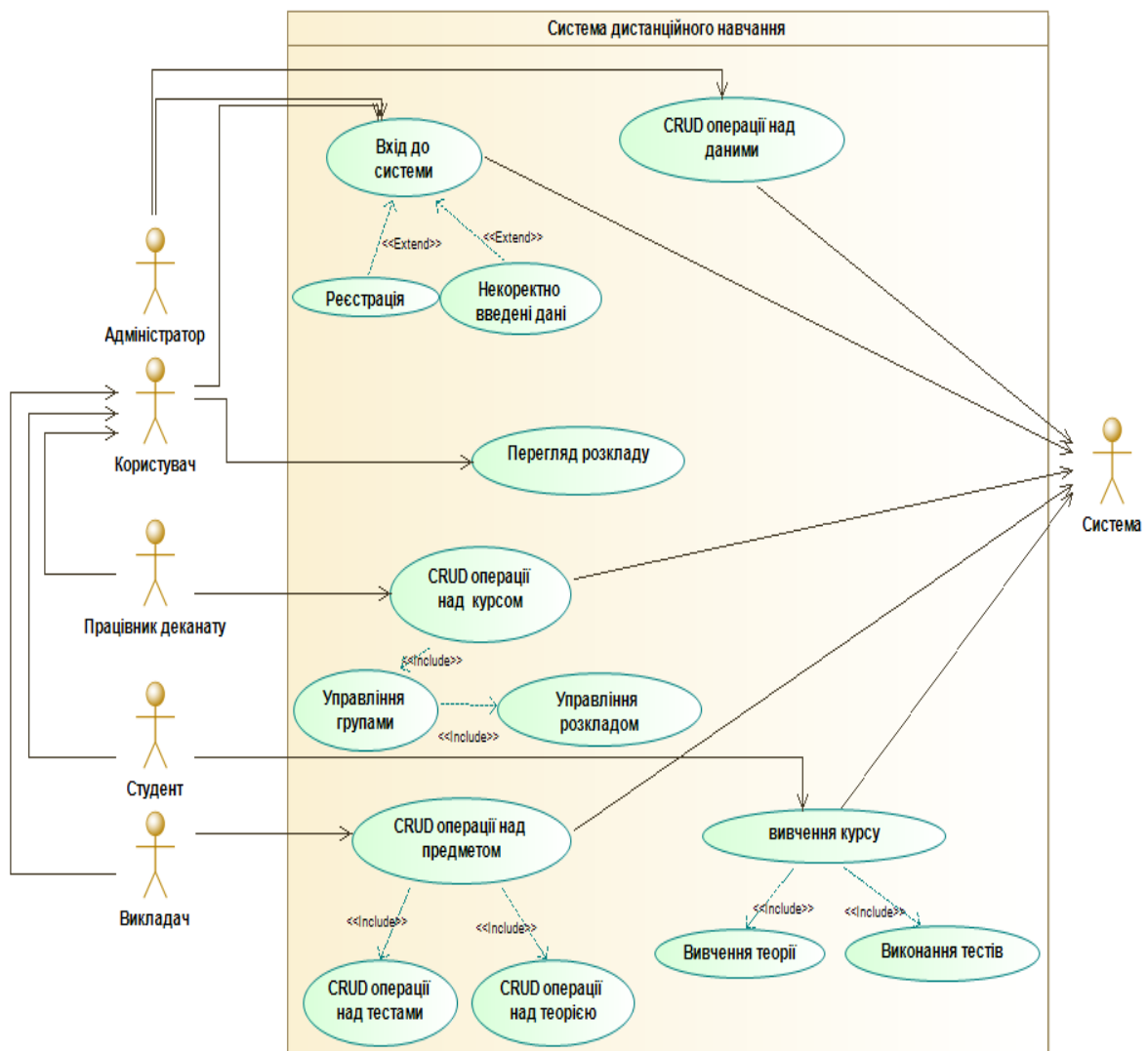


Рисунок 2.1 – Діаграма прецедентів системи дистанційного навчання

2.2.2 Діаграми послідовності

Діаграма послідовності – за її допомогою зображують те, як об'єкти взаємодіють впорядковано за часом, та їх аспекти. За допомогою таких діаграм відображають життєвий цикл під час якого виконується взаємодія актора з системою в рамках поточного прецеденту [34]. Її мета – це відображення об'єктів в динаміці, інформація надається у вигляді повідомлень, і взаємодія передбачає обмін ними в рамках сценарію.

Кожна дія представляється стрілкою, яка розміщується між життєвими циклами об'єктів і з'являються вони у зображеному порядку. Приклади діаграм зображено на рисунках 2.2 – 2.8.

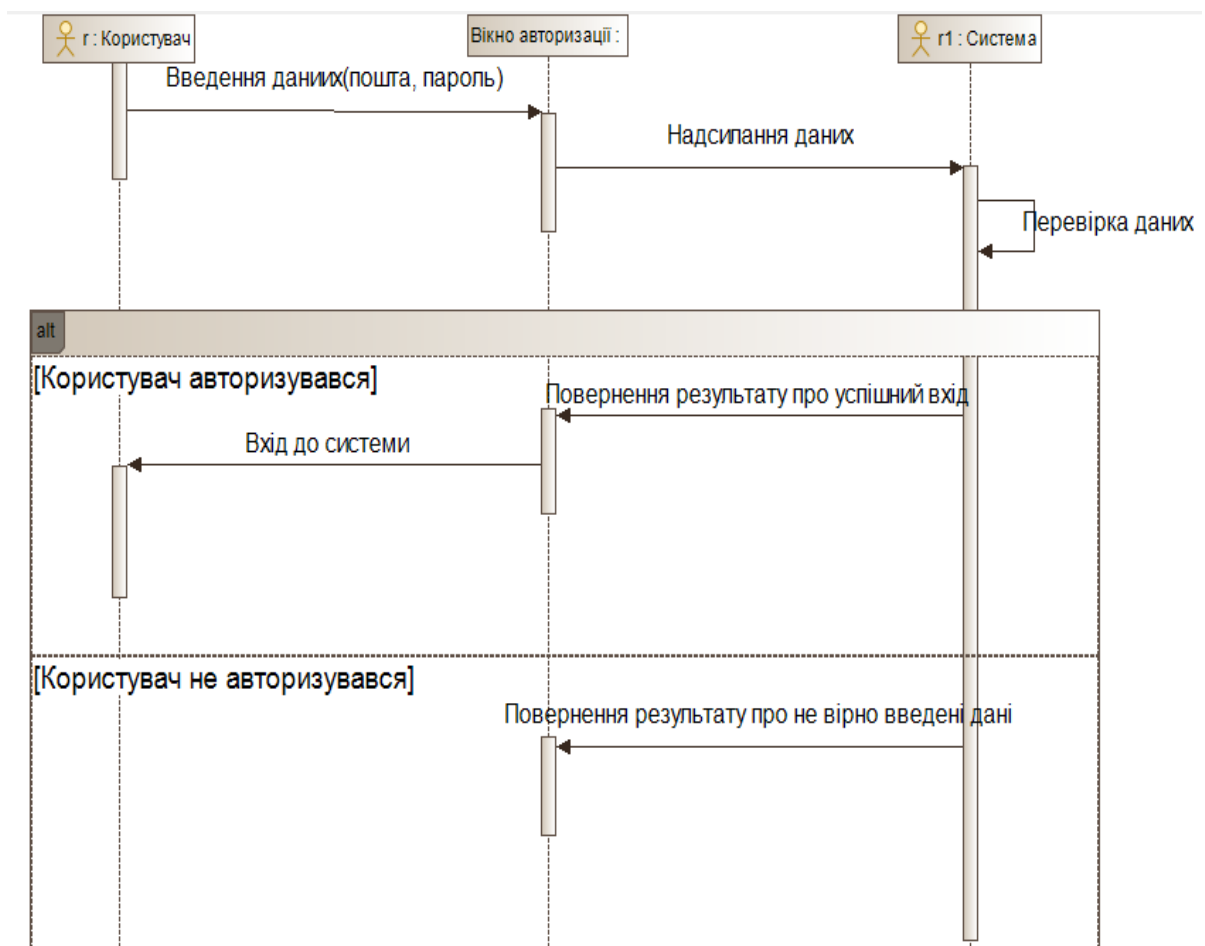


Рисунок 2.2 – Вхід до системи



Рисунок 2.3 – CRUD операції над предметом

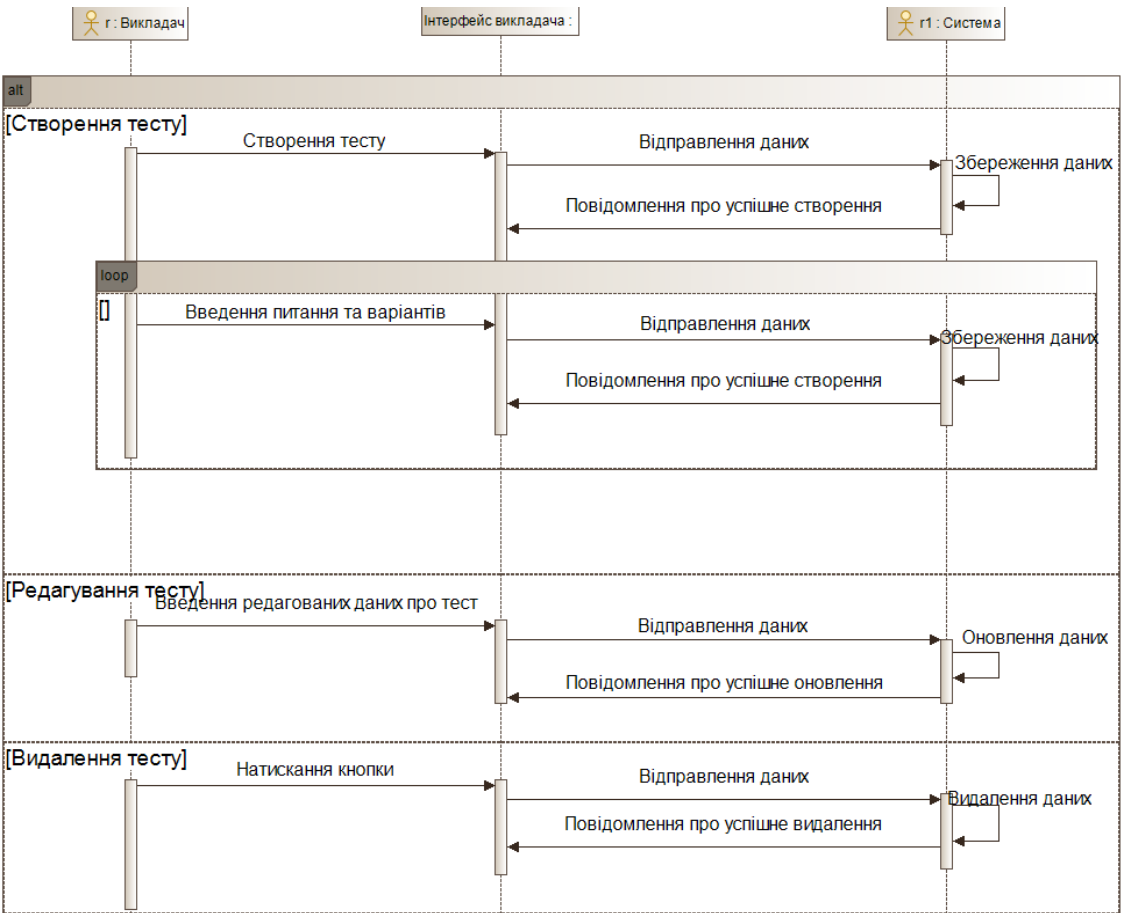


Рисунок 2.4 – CRUD операції над тестами



Рисунок 2.5 – CRUD операції над теорією

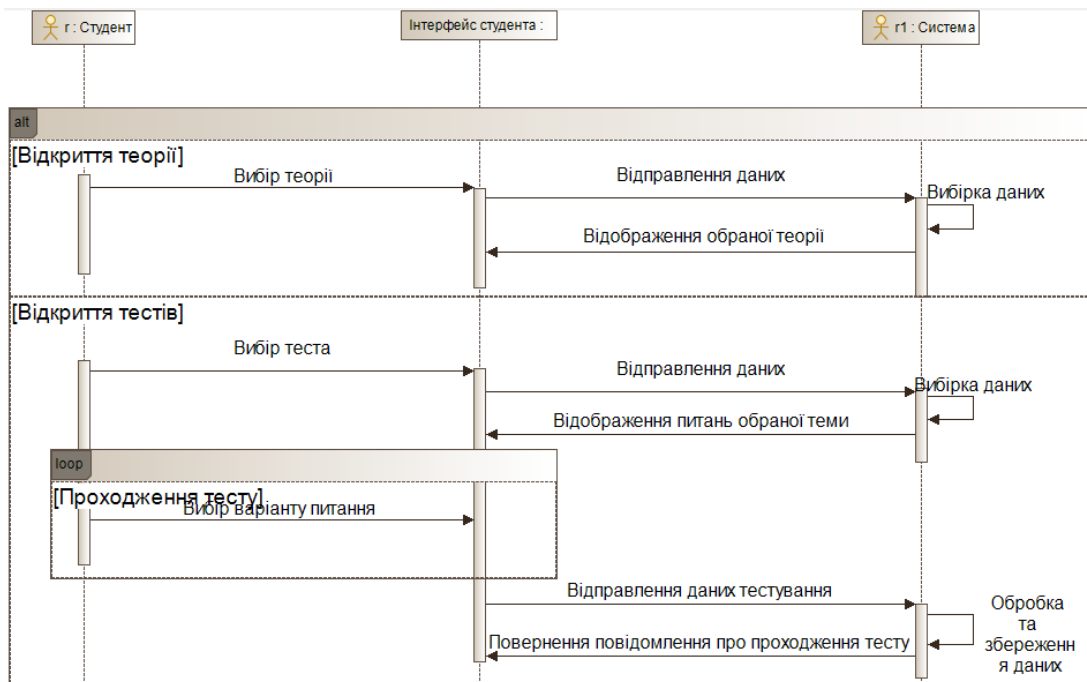


Рисунок 2.6 – Вивчення курсу



Рисунок 2.7 – Перегляд розкладу

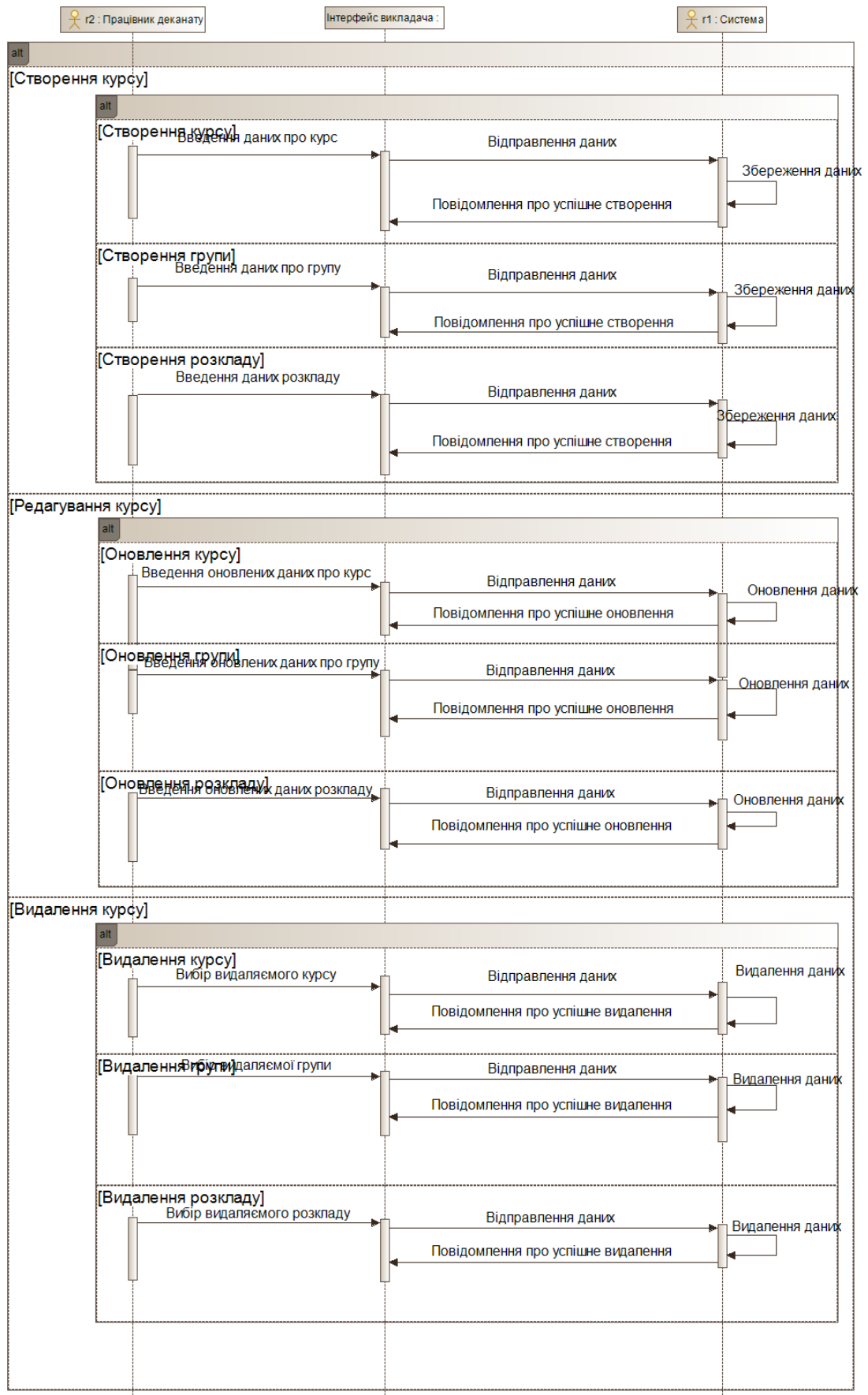


Рисунок 2.8 – CRUD операції над курсом

2.3 Вибір архітектури та проєктування серверної частини

Перед проєктуванням прототипу системи, потрібно правильно вибрати архітектуру, бо від цього залежить робота та існування розроблювальної системи.

Вибір архітектури програмного забезпечення – це одна з найважливіших ролей і є найголовнішою при розробці програмного забезпечення. Вона містить в собі компоненти, їх властивості та відносини між ними. Тобто описує принцип організації компонентів у системі. Документування даної архітектури спрощує процес комунікації, дозволяє передбачити недоліки та дає можливість використовувати шаблони повторно в інших проєктах. Правильно спроектована архітектура може зберегти час і гроші, а також збільшити швидкодію розроблювального продукту.

Зараз лідерами в архітектурі є: монолітна та мікросервісна, тому розглянемо про них детальніше.

Монолітна – це архітектура, яка розробляється як один пакет, тобто програмний засіб буде упакований в один моноліт та розгорнутий таким же чином. В архітектурі даного типу використовується тільки одна технологія, через те, що програма зібрана в одному місці, в неї легко додавати нові можливості та запускатися в одному місці. Вона незалежна ні від чого, так що її можна вважати автономною. У неї велика швидкість обміну між модулями, бо вони знаходяться поряд в одному коді й працюють в рамках одного програмного засобу [35].

Мікросервісна – це архітектура, яка слідує модульному виду розробки, тобто вона складається з незалежних частин, під назвою модулі, які можуть використовуватися у різних додатках, але водночас, вони обмінюються інформацією між собою, за допомогою методів (інтерфейсів) [36]. В архітектурі даного виду кожен компонент не залежить від інших, та можуть бути реалізованими на різних мовах програмування.

У цьому дипломному проєкті систему буде спроектовано за архітектурою моноліту, бо програмні засоби даного типу легко розгортаються в рамках одного сервера, легко розробляється та має високу швидкодію, шляхом того що компоненти знаходяться поруч в межах одного коду. Але мінусом даного підходу може те, що з часом, код системи модернізується та збільшується, що в кінцевому результаті, може призвести до того, що код стане перевантаженим і важким для сприйняття розробником.

Для кращого розуміння того, як буде працювати програма на фізичному обладнанні використовують спеціальну діаграму розгортання – це спеціальна діаграма, за допомогою якої показуються компоненти: вебсервери, сервери баз даних або сервер додатка, а також те, як вони взаємодіють між собою.

На рисунку 2.9 зображено діаграму розгортання, за допомогою якої зображено те, як розгортатиметься розроблювальний додаток. Клієнт за допомогою веббраузера відкриває сторінку та відправляє до сервера різні команди навіть не розуміючи цього. На сервері встановлено спеціальне програмне забезпечення, яке дає можливість запустити сайт написаний мовою програмування php, а також MySQL базу даних. Зв'язок клієнта та сервера виконується за допомогою протоколу HTTPS.

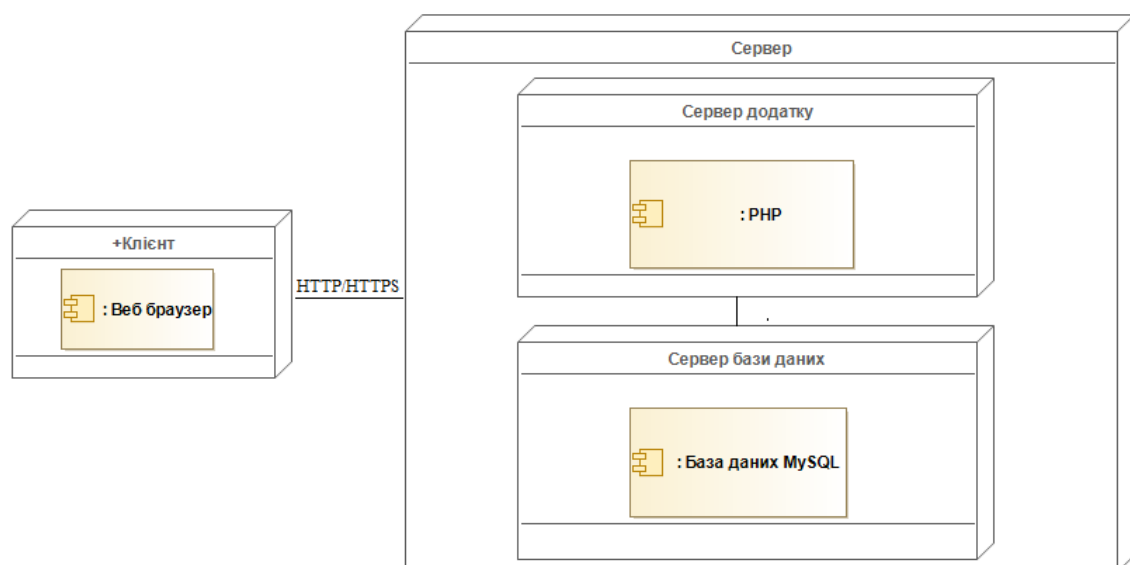


Рисунок 2.9 – Діаграма розгортання

Під час доступу з веб браузера клієнта, він посилає спеціальні запити до Laravel-додатка, який приймає їх та передає в маршрутизатор. Після того, як маршрутизатор обробив цей запит, він викликає спеціальний клас-контролер. Цей клас приймає весь запит, і якщо потрібно отримує дані, з якими надалі буде працювати, або одразу поверне відповідь у вигляді JSON коду.

Дані дії виконуються за правилами шаблону проєктування MVC, бо за цим шаблоном побудовано весь фреймворк Laravel. Він створений для полегшення роботи розробників так, якщо розшифрувати цю аббревіатуру отримаємо три слова «Модель-Вид-Контролер». Модель – це клас за допомогою якого виконуються будь-які види зв'язку з базою даних, вид – це те, що відображається користувачу, контролер – це вся бізнес-логіка.

Якщо розглянути більш детально, то маршрутизатор викликає контролер, який обирається зі списку усіх маршрутів, і може взаємодіяти з моделлю і виконувати певну бізнес-логіку. Після всіх взаємодій він починає генерувати відповідь до веб браузера клієнта у вигляді JSON, яка відображається користувачеві.

Для зручної розробки та редагування, Laravel в ієрархії файлів моделі, види та контролери зберігаються в різних каталогах, згідно зі структурою Laravel.

На рисунку 2.10 – зображено діаграму класів, вона описує моделі, контролери та їх методи. На даній діаграмі присутня головна Model як керує всією базою і від неї успадковуються моделі: UserModel, ScheduleModel та SystemModel, вони мають основний набір даних, за які відповідають. Цими моделями керують три контролери UserController, ScheduleController та SystemController, вони будуть працювати з їх даними в залежності що потрібно отримати у кінцевому результаті, будь-то створення нових записів, видалення, редагування або відображення.

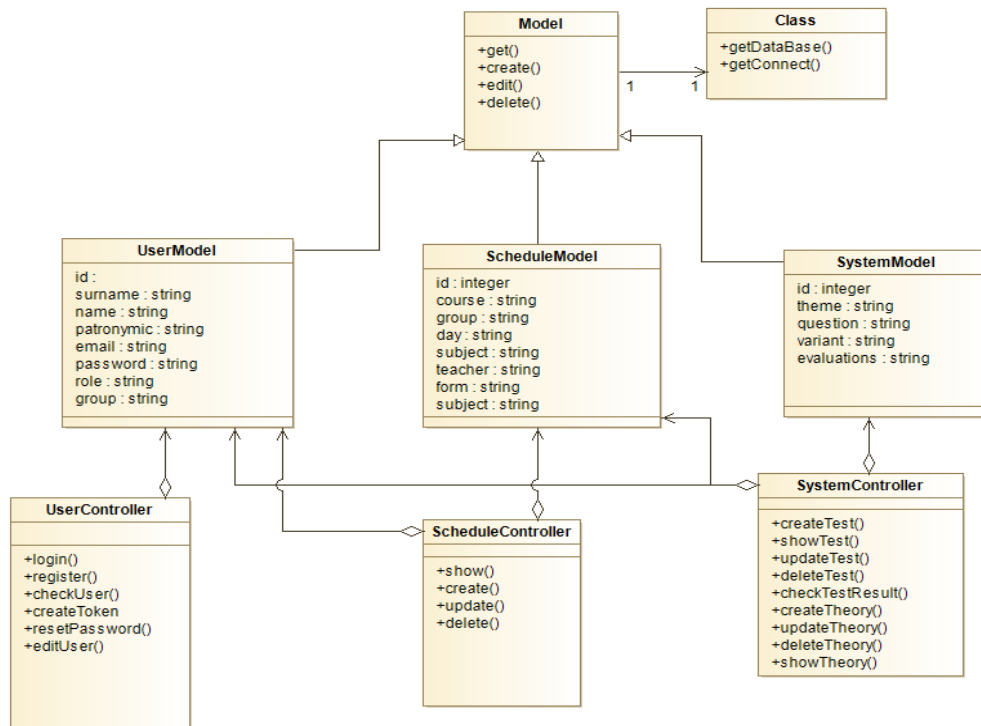


Рисунок 2.10 – Діаграма класів

2.4 Проєктування інтерфейсу користувача та ескізів

Для реалізації клієнтської частини сайту було обрано спеціальну бібліотеку React яка поставляється разом з фреймворком Laravel.

Ця бібліотека використовується для розробки користувацьких за стилем SPA, що в перекладі означає односторінковий додаток, тобто ви один раз відкриваєте сайт і далі користуєтесь без перезавантажень при переході між сторінками, а дані автоматично запитуються у браузера та обробляються бібліотекою [37].

Дана бібліотека підтримує підключення SCSS стилів – це спеціальний препроцесор, який поліпшує роботу розробнику та на стадії компіляції перетворюється у CSS, який розуміє браузер. Також JS коду, за допомогою якої з'являється можливість додати інтерактивність до сайту, наприклад в голові

сайту можна зробити список, який випадає при наведенні, чи обмінюватись даними з сервером не перезавантажуючи сторінку.

Перед початком розробки сайту, бажано спроектувати ескізи майбутніх сторінок, бо робити правки або додавати нові функції, легше на стадії створення ескізів, чим після завершення розробки сайту, бо є ймовірність того, що прийдеться перероблювати сайт з нуля.

На рисунку 2.11 зображено ескіз головної сторінки, вона містить наступні блоки: навігація, курси, новини, допоміжна інформація та блок дати.

У блоці курси будуть відображатись всі курси, до яких було долучено користувача, у блоці новини відображено актуальні новини, у допоміжній інформації відображено контакти адміністрації, за якими можна звернутися при некоректній роботі сайту, у блоці дата виводиться поточна інформація про число місяць та рік.

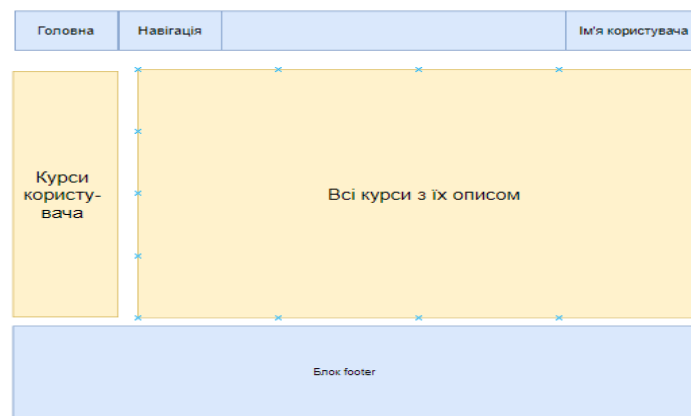


Рисунок 2.11 – Вигляд головної сторінки

Сайт поділяється на три частини: header(голова), body(тіло), footer(підвал). У даному сайті header може бути чотирьох видів, бо в системі присутні чотири ролі та footer який однаковий для всіх. Оскільки вони повинні бути присутніми на кожній сторінці сайту, та бути незмінним, було прийнято рішення описати їх один раз на окремому ескізі. Ескіз наведено на рисунку 2.12.

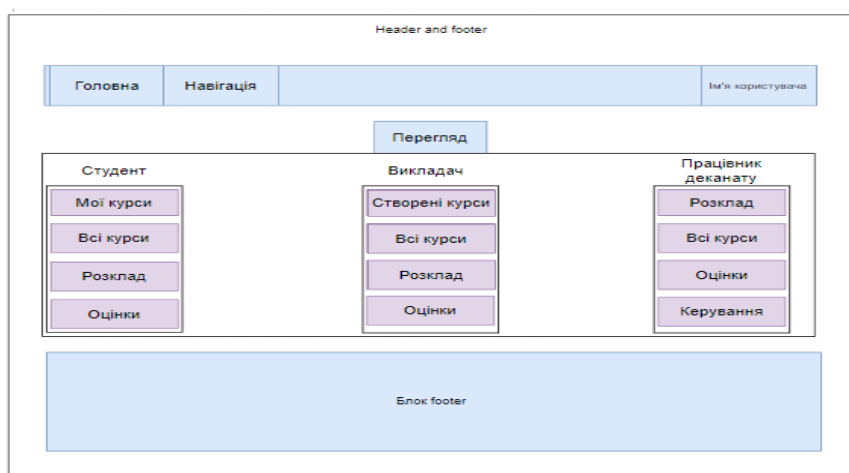


Рисунок 2.12 – Вигляд голови та підвалу сайту

На рисунку 2.13 зображено ескіз перегляду курсу. На рисунках 2.14 – 2.15 зображено форми реєстрації та авторизації, з відповідними полями для заповнення.

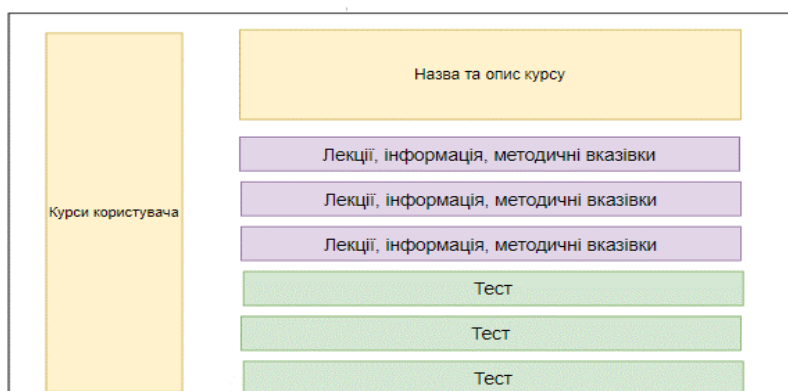


Рисунок 2.15 – Перегляд курсу

The diagram shows a login form titled "Login". It contains two input fields: "Пошта" (Email) and "Пароль" (Password). Below these fields is a button labeled "Авторизуватись" (Log in).

Рисунок 2.13 – Форма авторизації

Register

Прізвище

Ім'я

По батькові

Пошта

Пароль

Повторно
пароль

Група

Зареєструватись

Рисунок 2.14 – Форма реєстрації

На рисунку 2.16 зображено ескіз основних форм залучених у редагуванні, створені або видалені записи з бази даних системи.

Підставляється одне з обраних видів ескізів описаних нижче

Відображення таблиці з даними та можливістю редагування або видалення

Додати групу

Групи

Додати групу

Додати користувача

Прізвище

Ім'я

По батькові

Пошта

Пароль

Додати користувача

Додати викладача

Прізвище

Ім'я

По батькові

Пошта

Додати викладача

Додати предмет

Назва предмету

Додати предмет

Рисунок 2.16 – Ескіз основних форм

На рисунку 2.17 зображено те, як повинна виглядати форма додавання питання до тестування.

Назва питання

1 варіант

2 варіант

3 варіант

Додати ще один варіант

Додати питання

Рисунок 2.17 – Форма додавання питання

На рисунку 2.18 зображено ескіз форми створення розкладу, за допомогою якого працівник деканату зможе вносити розклад до груп.

Вибір групи						
	№	Тип	Предмет	Викладач	Редагувати	Видалити
П о н е д і л о к	1	-----	-----	-----	Додати	
	2	-----	-----	-----	Додати	
	3	-----	-----	-----	Додати	
	4	-----	-----	-----	Додати	

Рисунок 2.18 – Форма створення розкладу

2.5 Висновки за розділом 2

В даному розділі було проведено дослідження, яке охоплює наступні етапи: проектування, моделювання, розробка діаграм використання та послідовності, вибір архітектури серверної частини та ескізів інтерфейсу користувача.

При проектуванні визначено, та описано те, що повинні бачити користувачі, та їх функції, розділення функціонала на ролі, продуктивність сервера, та його можливість виконувати резервне збереження інформації з певною періодичністю.

Моделювання дозволило визначити взаємодії компонентів та оптимізувати їх роботу. Використання діаграм варіантів та діаграм послідовності надало деталізований огляд функціонала та взаємодій користувачів із системою. За допомогою діаграми послідовності було візуалізовано процеси та взаємодію між компонентами системи. Це сприяло оптимізації послідовності подій та уточненню зв'язків між елементами, що допоможе при розробці системи.

При виборі архітектури, було важливо обрати ту, яка б змогла забезпечити ефективність, масштабованості, та легкість при створенні системи. Провівши ретельний аналіз існуючих варіантів було обрано оптимальне рішення для поставлених задач.

Заключенням, було проєктування інтерфейсу користувача та створення ескізів, з яких в майбутньому буде розроблено клієнтську частину. Було визначено ключові аспекти користувацького досвіду та вигляду системи, які забезпечують зручність використання та мінімалістичний дизайн.

У цілому, вивчення та розгляд різноманітних аспектів проєктування та моделювання системи підкреслили важливість інтегрованого підходу до розробки, враховуючи як технічні, так і дизайнерські аспекти. Це дослідження стало основою для подальшої розробки та вдосконалення системи з урахуванням виявлених особливостей та вимог користувачів.

3 РЕАЛІЗАЦІЯ СИСТЕМИ

Для створення системи було прийнято рішення використовувати найсучасніші версії мов програмування, пакетів та фреймворків, з метою забезпечення високої ефективності, безпеки та функціональності. Дану систему розроблено з використанням фреймворку Laravel у поєднанні з бібліотекою React під операційною системою Windows. Розглянемо ключові моменти її створення.

3.1 Створення бекенду

Для початку розробки бекенду, потрібно налаштувати середовище та створити проєкт.

Для цього спочатку було встановлено програмний засіб під назвою OpenServer – це досить популярний інструмент серед розробників вебдодатків, бо надає можливість їх запусків локально без необхідності публікації їх в

Інтернеті для тестування та розробки. І цим самим економить велику купу часу. Дана програма містить в собі наступні серверні програми:

- Apache – вебсервер який завдяки спеціальному модулю може обробляти PHP-скрипти. Коли браузер відправляє запит, сервер Apache приймає цей запит і бере на себе його обробку та відправляє його до PHP-інтерпретатору, який обробляє код та генерує HTML-код або JSON, який потім надсилається назад серверу Apache, який повертає його браузеру;

- Mysql – є сервером баз даних, який використовується для зберігання, управління та доступу до даних з коду PHP. Це дозволяє створювати та працювати з базами даних, без підключення до віддалених серверів які знаходяться в інтернеті.

Таким чином, даний програмний засіб має все для того, щоб за кілька натискань створити всі необхідні локальні вебсервери. Аналогами цієї програми є docker – це інший засіб для розгортання програмного забезпечення у вигляді «контейнерів». На відміну від OpenServer, який надає готове середовище вебсервера для розробки з встановленими усіма версіями модулів, docker надає можливість запускати сайти в ізолювано. Найбільша різниця між ними полягає у тому, що OpenServer орієнтований на розгортання вебсерверів в операційній системі Windows, в той час, як Docker є універсальним засобом в якому спочатку потрібно розробити інструкції запуску, які написані одного разу, і використовуються на інших операційних системах (Windows, IOS, Linux) для запуску проєкту. Але для того, щоб docker працював на windows, потрібно попередньо налаштувати візуалізацію в цій операційній системі, але при цьому деякі інші програми перестають працювати, тому в нашому випадку він не підходить для розробки.

Після запуску вебсерверів, потрібно створити проєкт за допомогою спеціального пакетного менеджера залежностей для PHP. Виконавши команди, які зображено на рисунку 3.1, ми отримуємо проєкт Laravel, який вже є сконфігурованим і з встановленими основними бібліотеками, які будуть використовуватись протягом розробки.

Далі необхідно створити базу даних і виконати міграцію, тобто створити таблиці та структуру бази даних відповідно до моделі даних, які є початковими й з можливістю простого їх створення в майбутньому. Цей крок є важливим, оскільки він готує середовище для зберігання та роботи з даними, що будуть використані в вашому проєкті. На рисунку 3.2 зображено те, як відкривається спеціальний менеджер, використовуючи який, створюється база даних, і виконання міграції, яка створює потрібні таблиці в ній.

```

wwwmi@ c:\OpenServer\OpenServer
# composer -v
Warning: This development build of Composer is over 60 days old. It is recommended
to update it by running "c:\opensever\opensever\modules\php\PHP_8.0\..\..\us
erdata\composer\composer.phar self-update" to get the latest version.

Composer version 2.3-dev+f1f013edde6528e58a47dfd94af527f5b9af801c (2.3-dev) 2022-0
4-29 13:02:24

wwwmi@ c:\OpenServer\OpenServer
# composer create-project --prefer-dist laravel/laravel EducationalSite
Warning: This development build of Composer is over 60 days old. It is recommended
to update it by running "c:\opensever\opensever\modules\php\PHP_8.0\..\..\us
erdata\composer\composer.phar self-update" to get the latest version.
Creating a "laravel/laravel" project at "../EducationalSite"
Installing laravel/laravel (v9.5.2)
- Downloading laravel/laravel (v9.5.2)
- Installing laravel/laravel (v9.5.2): Extracting archive
Created project in C:\OpenServer\OpenServer\EducationalSite
> @php -r "file_exists('.env') || copy('.env.example', '.env');"
Loading composer repositories with package information
  laravel/sail ..... DONE
  laravel/sanctum ..... DONE
  laravel/tinker ..... DONE
  nesbot/carbon ..... DONE
  nunomaduro/collision ..... DONE
  nunomaduro/termwind ..... DONE
  spatie/laravel-ignition ..... DONE

84 packages you are looking for funding.
Use the "composer fund" command to find out more!
> @php artisan vendor:publish --tag=laravel-assets --ansi --force

  INFO No publishable resources for tag [laravel-assets].

> @php artisan key:generate --ansi

  INFO Application key set successfully.

wwwmi@ c:\OpenServer\OpenServer

```

Рисунок 3.1 – Створення проєкту Laravel

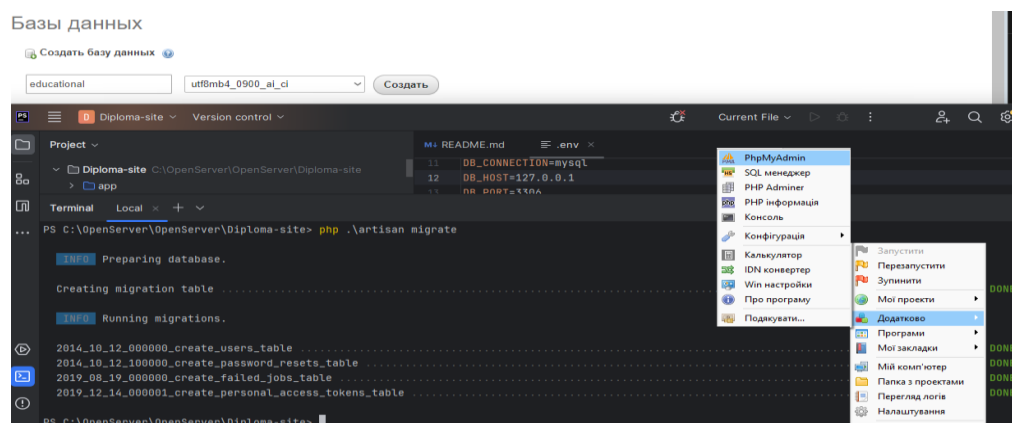


Рисунок 3.2 – Створення бази даних та заповнення її початковими таблицями

При створенні нового Laravel додатку, створюється певна структура каталогів та файлів, яка допомагає організувати робоче середовище для розробки, яка зображена на рисунку 3.3. Коротко про кожен з цих каталогів:

- `app` – містить в собі основний код програми, який складається з контролерів, моделей, сервісів та інших допоміжних класів, що відповідають за бізнес-логіку;
- `bootstrap` – містить файли, які завантажують середовище проєкту та ініціалізують його перед запуском;
- `config` – містить в собі конфігураційні файли для налаштування додатка, наприклад для підключення до сторонніх сервісів, налаштування кешування тощо;
- `database` – містить файли, пов'язані з оновленням бази даних, які складаються з: файлів міграцій, які дозволяють створювати й оновлювати структуру та спеціальних сідерів, які допомагають заповнити таблиці тестовими даними;
- `public` – містить всі публічні файли проєкту, такі як CSS, скомпільований JavaScript та зображення, доступ до яких може отримати будь-який користувач;
- `resources` – містить в собі шаблони Blade для виглядів, стилі CSS, та не скомпільований JavaScript;
- `routes` – містить файли в яких визначаються маршрути для всього проєкту, основними є файли `web` та `api`, які містять маршрути для вебінтерфейсу і взаємодії з додатком через API;
- `storage` – містить в собі підкаталоги для зберігання логів, кешу, завантажень та іншої тимчасової інформації;
- `tests` – містить тести які створюються розробником для автоматичного тестування роботи додатка та перевірки на помилки;
- `vendor` – містить у собі усі сторонні бібліотеки, які були встановлені за допомогою `composer` і використовуються у проєкті.

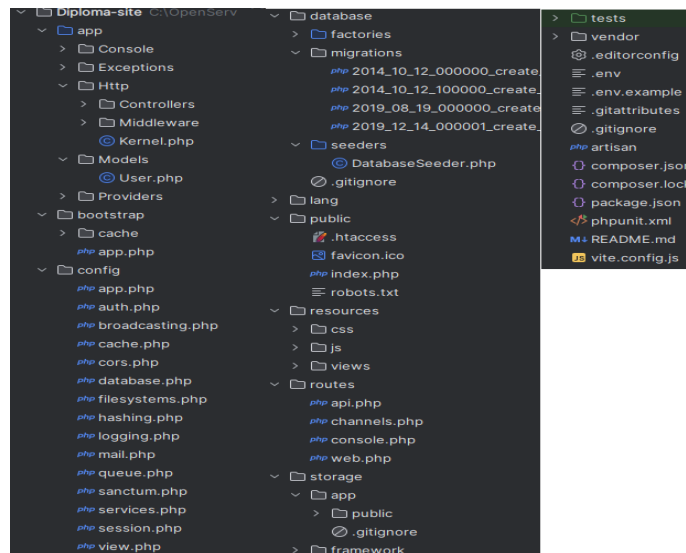


Рисунок 3.3 – Структура проєкту

Розглянувши структуру проєкту, перейдемо до опису його основних частин коду. На рисунку 3.4 представлено логіку авторизації. В систему надходять від користувача його пошта та пароль, вони передаються в функцію «attempt», яка хешує пароль, бо в базі даних для забезпечення безпеки він зберігається саме в такому вигляді, і виконується пошук в таблиці користувачів з такими даними. У випадку, якщо такого не існує, або дані некоректні, буде відображено помилку «Введіть коректні дані», а у випадку успіху – створено токен, із застосуванням якого, користувач зможе ідентифікувати себе в системі.

```

65 public function login(Request $request)
66 {
67     try {
68         if (Auth::attempt(['email' => $request->email, 'password' => $request->password])) {
69             $user = Auth::user();
70
71             return response()->json([
72                 'status' => true,
73                 'message' => 'User Created Successfully',
74                 'token' => $user->createToken( 'name: API TOKEN')->plainTextToken,
75                 'user' => $user->toJson()
76             ], [status: 200]);
77         } else {
78             return response()->json([
79                 'status' => false,
80                 'message' => 'Введіть коректні дані'
81             ], [status: 500]);
82         }
83     } catch
84     (\Throwable $th) {
85         return response()->json([
86             'status' => false,
87             'message' => $th->getMessage()
88         ], [status: 500]);
89     }
90 }
91

```

Рисунок 3.4 – Логіка авторизації

Для подальшого опису був обраний клас, який відповідає за створення викладачів. Основний вибір спрямований на той факт, що в цьому класі вже реалізовані всі методи CRUD, які впливають на дані в базі даних. Аналогічна логіка описана і в інших класах, але з використанням відповідних назв полів, характерних для кожного з них. На рисунку 3.5 зображено те, як методи в класі отримують інформацію у вигляді спеціального об'єкта, в якому виконується перевірка вхідних даних і повернення помилки у випадку їх невідповідності. На рисунку 3.6 зображено те, як перевіряються за допомогою спеціальних правил, і використовуються методи для отримання даних з валідованої вхідної інформації.

```

10 class TeacherRepository
11 {
12     ▲ DimaK0st +1
13     public function create(CreateTeacherRequest $request, Teacher $teacher)
14     {
15         $teacher->surname = $request->getSurname();
16         $teacher->name = $request->getName();
17         $teacher->patronymic = $request->getPatronymic();
18         $teacher->email = $request->getEmail();
19         $teacher->save();
20
21         return $teacher;
22     }
23
24     ▲ DimaK0st
25     public function update(UpdateTeacherRequest $request)
26     {
27         $teacher = Teacher::findOrFail($request->getId());
28
29         $teacher->surname = $request->getSurname();
30         $teacher->name = $request->getName();
31         $teacher->patronymic = $request->getPatronymic();
32         $teacher->email = $request->getEmail();
33         $teacher->save();
34
35         return $teacher;
36     }
37
38     ▲ DimaK0st +1
39     public function delete(DeleteTeacherRequest $request)
40     {
41         $teacher = Teacher::findOrFail($request->getId());
42         $teacher->delete();
43     }
44 }

```

Рисунок 3.5 – Основний принцип реалізації класів CRUD

```

7  class CreateTeacherRequest extends FormRequest
8  {
9      /**
10       * Get the validation rules that apply to the request.
11       *
12       * @return array
13       */
14       public function rules()
15       {
16           return [
17               'surname' => 'required|string',
18               'name' => 'required|string',
19               'patronymic' => 'required|string',
20               'email' => 'nullable|string',
21           ];
22       }
23
24       /**
25       * @return string
26       */
27       public function getSurname(): string
28       {
29           return $this->input('key: ' . 'surname');
30       }
31

```

Рисунок 3.6 – Приклад валідації вхідної інформації

На рисунках 3.7-3.8 зображено трішки ускладнену логіку, в якій виконується дві дії при створенні питання для тесту. В якості параметрів передається номер тесту, до якого додається питання, його назва, а також варіанти відповідей у вигляді масиву. Першою дією створюється запис в базі даних, в якій зберігається назва питання, після чого береться його ідентифікатор, який використовується для ідентифікації при додаванні питань в базу даних.

```

3 usages  DimaK0st +1 *
13  class QuestionService
14  {
15      no usages  DimaK0st
16      public function __construct(private VariantService $variantService, private QuestionRepository $questionRepository)
17      {
18      }
19
20      DimaK0st +1
21      public function create(CreateQuestionRequest $request)
22      {
23          $question = $this->questionRepository->create($request->getTestId(), $request->getTitle());
24          $this->variantService->create($request->getVariants(), $question->id);
25
26          return $question;
27      }
28

```

Рисунок 3.7 – Приклад логіки створення питання

```

13 class QuestionRepository
14 {
15     public function create(int $testId, string $title)
16     {
17         $question = new Question();
18         $question->test_id = $testId;
19         $question->title = $title;
20         $question->save();
21
22         return $question;
23     }

```

```

13 public function create(array $variants, $id)
14 {
15     foreach ($variants as $variant) {
16         $this->variantRepository->create($variant, $id);
17     }
18 }

```

```

10 public function create(array $variant, $id){
11     $variantModel = new Variant();
12     $variantModel->question_id = $id;
13     $variantModel->text = $variant['text'];
14     $variantModel->correct = $variant['correct'];
15     $variantModel->save();
16 }

```

Рисунок 3.8 – Приклад створення питання

Приклад моделі питання відображено на рисунку 3.9. Зазвичай в Laravel моделі використовуються для обміну інформацією з базою даних. У попередніх прикладах було показано те, як в них передаються дані, й за допомогою методу save вони зберігаються. Однак моделі також дозволяють визначати відносини між таблицями. Це означає, що при належному налаштуванні, при виконанні запиту до бази даних, варіанти відповідей, автоматично додаються до результуючого об'єкту. Використання відносин також ефективно скорочує час, оскільки дані отримуються за допомогою одного запиту. Наприклад, один тест може мати десять питань, і кожне питання може мати три варіанти відповідей. Без відносин це призвело б до 41 запиту до бази даних ($1 + 10 + 10 * 3$), але з їх використанням це обмежиться одним запитом.

```

15 class Question extends Model
16 {
17     use HasFactory;
18
19     protected $with = ['variants'];
20
21     public function test()
22     {
23         return $this->belongsTo('related: Test::class, foreignKey: 'test_id', ownerKey: 'id');
24     }
25
26     public function variants()
27     {
28         return $this->hasMany('related: Variant::class, foreignKey: 'question_id', localKey: 'id');
29     }
30
31 }

```

Рисунок 3.9 – Приклад моделі питання

Наступним розглянемо роботу зі штучним інтелектом, для цього було встановлено бібліотеку «jumbaeric/laragpt», яка має зрозумілу документацію, та надає функції для легкої роботи з ChatGpt Api. На рисунку 3.10 зображено, як функція отримує дані від користувача, а саме назву тестування, його опис та кількість питань для генерування. Оскільки запит займає певний час, в Laravel можна зробити спеціальний запит, який виконується асинхронно, тобто користувач лише надсилає дані, і після того, як код їх опрацює, тест буде додано до курсу.

```

Dima Kostetskiy
public function generateWithAi(CreateTestRequest $request)
{
    $createTestDto = CreateTestDto::fromRequest($request);

    GetDataFromAi::dispatch($createTestDto);

    return true;
}

```

Рисунок 3.10 – Логіка запуску асинхронного створення тесту за допомогою штучного інтелекту

На рисунку 3.11 зображено логіку асинхронної команди, яка задає початкові налаштування, і описує те, як повинен функціонувати штучний інтелект, після генерується запит від користувача, в якому всі дані збираються в масив і передаються статичній функції бібліотеки, яка відправить запит на ChatGpt до API, і поверне результат.

```

34  /**
35   * Execute the job.
36   *
37   * @return void
38   */
39   * Dilma Kostetskyl *
40   public function handle()
41   {
42       $result = Laragpt::chat([
43           'model' => 'gpt-3.5-turbo', // required
44           'messages' => [ // required
45               [
46                   'role' => "system",
47                   'content' => "Ви помічник, який створює тести за темою і описом, які передаються і повертаються у вигляді JSON.
48                   з наступною структурою:
49                   {\title\: \"\", \description\: \"\", \course_id\: 1, \count\: 1, \questions\:
50                   [{\question\: \"\", \options\: [{\answer\: \"\", \correct\: 0}]}]}."
51               ],
52               [
53                   'role' => "user",
54                   'content' => "Тема тесту - " . $this->createTestDTO->title . ", опис - " .
55                   $this->createTestDTO->description . ", кількість питань - " . $this->createTestDTO->count . "."
56               ],
57           ],
58       );
59       $questionsDTO = new GeneratedAiQuestionsDTO($result);
60       return $this->testService->createFromAi($this->createTestDTO, $questionsDTO);
61   }
62 }
63

```

Рисунок 3.11 – Логіка асинхронної команди відправки запиту

Далі відповідь штучного інтелекту буде конвертовано за допомогою спеціальних класів DTO, які використовуються для зручної передачі даних між контролерами й сервісами проєкту. На рисунку 4.12 зображено те, як інформацію, спочатку представлено у вигляді звичайного рядку JSON. Ця відповідь передається в клас GeneratedAiQuestionsDTO, де рядок розділяється на об'єкти, і на рисунку 3.13 зображено логіку перетворення цієї інформації. На рисунку 3.14 зображено результуючий об'єкт, який передається по проєкту й з нього інформація зберігається в базу даних.

```

{ "questions": [ { "question": "Яка основна функція програми PocketTracer?", "options": [ { "answer": "Графічний дизайн", "correct": 0 }, { "answer": "Моделювання мереж", "correct": 1 }, { "answer": "Музичне створення", "correct": 0 }, { "answer": "Текстова переписка", "correct": 0 } ] }, { "question": "Яке призначення має PocketTracer?", "options": [ { "answer": "Фільмова монтаж", "correct": 0 }, { "answer": "Кулінарія", "correct": 0 }, { "answer": "Моделювання та симуляція мережі", "correct": 1 }, { "answer": "Малюнок на воді", "correct": 0 } ] }, { "question": "Які можливості надає програма PocketTracer?", "options": [ { "answer": "Спортивні прогнози", "correct": 0 }, { "answer": "Ігри віртуальної реальності", "correct": 0 }, { "answer": "Симуляція мережі та аналіз", "correct": 1 }, { "answer": "Геологічні дослідження", "correct": 0 } ] }, { "question": "Які дані можна аналізувати за допомогою PocketTracer?", "options": [ { "answer": "Тренди моди", "correct": 0 }, { "answer": "Економічні індикатори", "correct": 0 }, { "answer": "Поведінку мережі", "correct": 1 }, { "answer": "Астрономічні спостереження", "correct": 0 } ] }, { "question": "Які інструменти доступні в PocketTracer для редагування зображень?", "options": [ { "answer": "Мозаїка та колажі", "correct": 1 }, { "answer": "Графічні пензли", "correct": 0 }, { "answer": "Текстові стилі", "correct": 0 }, { "answer": "Музичні семпли", "correct": 0 } ] }, { "question": "Чому приділяється основна увага в PocketTracer?", "options": [ { "answer": "Ігрові фічі", "correct": 0 }, { "answer": "Дизайн інтерфейсу", "correct": 0 }, { "answer": "Моделювання мереж", "correct": 1 }, { "answer": "Формування новин", "correct": 0 } ] }, { "question": "Які області використання програми PocketTracer особливо популярні?", "options": [ { "answer": "Гастрономія", "correct": 0 }, { "answer": "Медицина", "correct": 0 }, { "answer": "Інформаційні технології", "correct": 1 }, { "answer": "Астрофізика", "correct": 0 } ] } ] }

```

Рисунок 3.12 – Інформація яка приходить з до API

```

class GeneratedAiQuestionsDTO {
    public array $questions = []; // Масив об'єктів QuestionDTO

    1 usage  ▲ Dima Kostetskiy
    public function __construct(array|string $data)
    {
        10 if (is_string($data)) {
            11     $data = json_decode($data, associative: true);
        }
        12 $this->fill($data);
        13 return $this->questions;
    }

    1 usage  ▲ Dima Kostetskiy
    private function fill(array $data)
    {
        17 foreach ($data['questions'] as $questionData) {
            18     $this->questions[] = new QuestionDTO($questionData);
        }
    }

    5 class OptionDTO {
        6     public $variant;
        7     public $correct;
        8
        1 usage  ▲ Dima Kostetskiy
        public function __construct($data)
        {
            10 {
                11     $this->variant = $data['answer'];
                12     $this->correct = $data['correct'];
            }
        }
    }
}

```

```

5 usage  ▲ Dima Kostetskiy
class QuestionDTO {
    6     public $question = '';
    7     public $options = [];
    8
    1 usage  ▲ Dima Kostetskiy
    public function __construct(array|string $data)
    {
        10 if (is_string($data)) {
            11     $data = json_decode($data, associative: true);
        }
        12 $this->fill($data);
        13
        14 return [
            15     'question' => $this->question,
            16     'options' => $this->options
        ];
    }

    1 usage  ▲ Dima Kostetskiy
    private function fill(array $data)
    {
        22 $this->question = $data['question'];
        23
        24 foreach ($data['options'] as $optionData) {
            25     $this->options[] = new OptionDTO($optionData);
        }
    }
}

```

Рисунок 3.13 – Логіка перетворення інформації

```

App\Http\Dtos\GeneratedAiQuestionsDTO [#539 // app\Jobs\GetDataFromAi.php:
+questions: array:7 [
    0 => App\Http\QuestionDTO [#540
        +question: "Яка основна функція програми PocketTracer?"
        +options: array:4 [
            0 => App\Http\OptionDTO [#541
                +variant: "Графічний дизайн"
                +correct: 0
            ]
            1 => App\Http\OptionDTO [#542
                +variant: "Моделювання мереж"
                +correct: 1
            ]
            2 => App\Http\OptionDTO [#543
                +variant: "Музичне створення"
                +correct: 0
            ]
            3 => App\Http\OptionDTO [#544
                +variant: "Текстова переписка"
                +correct: 0
            ]
        ]
    ]
    1 => App\Http\QuestionDTO [#545
        +question: "Яке призначення має PocketTracer?"
        +options: array:4 [
            0 => App\Http\OptionDTO [#546
                +variant: "Фільмова монтаж"
                +correct: 0
            ]
            1 => App\Http\OptionDTO [#547
                +variant: "Кулінарія"
                +correct: 0
            ]
            2 => App\Http\OptionDTO [#548
                +variant: "Моделювання та симуляція мережі"
                +correct: 1
            ]
        ]
    ]
]

```

Рисунок 3.14 – Результуючий оброблений об'єкт

3.2 Створення фронтенду

Проаналізувавши існуючі інструменти, було вирішено, що для розробки фронтенду, найкраще підходить бібліотека React, бо вона сумісна з Laravel. Для того, щоб додати її в проєкт, потрібно просто виконати команду «npm install react react-dom», після чого створити стартовий файл в каталозі ресурсів Laravel за

маршрутом «resources/js/» з назвою «app.jsx». На рисунку 3.15 зображено те, як потрібно визначити маршрут, який буде повертати основний шаблон з підключеним React.

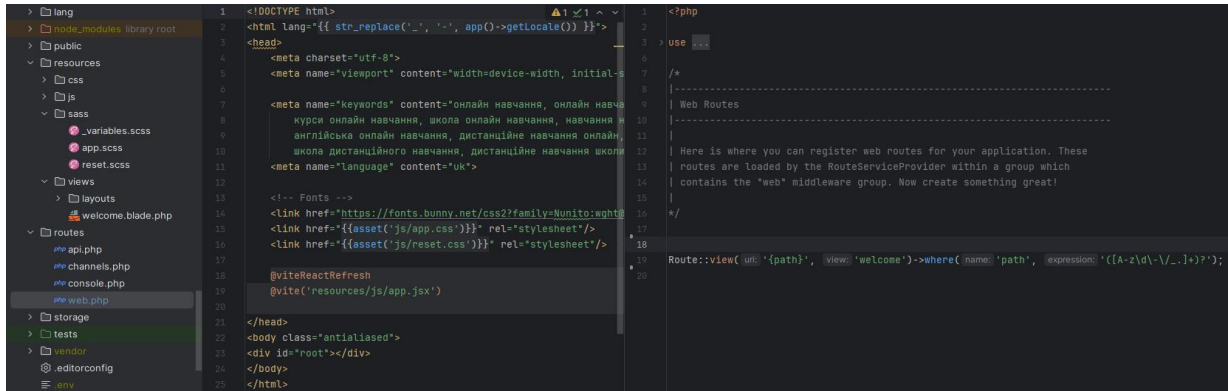


Рисунок 3.15 – Визначення маршруту для відображення фронкенду

На рисунку 3.16 зображено приклад коду, який відповідає за посилення запитів на бекенд. Змінна `headers` містить в собі HTTP-заголовок, який містить інформацію про те, як сервер і клієнт повинні обробляти запит і відповідь.

Заголовок містить наступні поля:

- `content-Type` – вказує тип вмісту, у цьому випадку `application/json`, що означає, що запит або відповідь представлена у форматі JSON;
- `ассерт` – вказує тип вмісту, який клієнт очікує отримати у відповідь на запит. У цьому випадку, клієнт очікує отримати дані у форматі JSON;
- `access-Control-Allow-Headers` – вказує список допустимих заголовків, які клієнт може надсилати в запиті;
- `access-Control-Allow-Origin` – вказує список доменів, які мають можливість отримувати доступ до ресурсу;
- `mode` – вказує, як обробляти запит. У цьому разі значення `no-cors` означає, що запит має бути виконано без використання CORS;
- `credentials` – вказує, чи повинні надсилатися куки й заголовки авторизації разом із запитом. У цьому разі значення `same-origin` означає, що куки та заголовки авторизації буде надіслано лише тоді, запит надсилають на той самий домен, що й сторінку, на якій виконується скрипт;

- authorization – містить токен, який використовується для аутентифікації користувача при виконанні запиту.

Дана зміна використовується при будь-якому запиті від клієнта до бекенду, тому її й було створено, щоб не дублювати код.

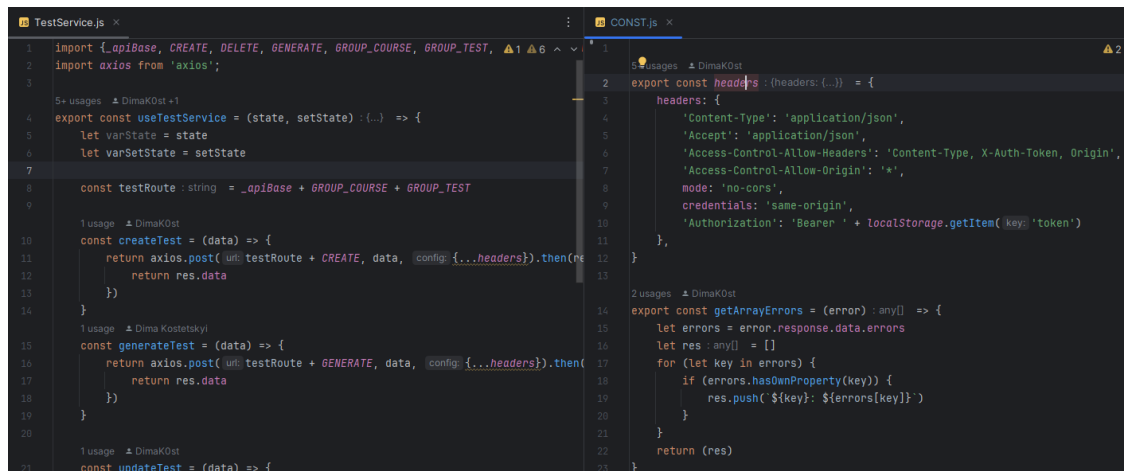


Рисунок 3.16 – Приклад методів, які надсилають запити на бекенд

На рисунку 3.17 наведено те, як працює роутинг в клієнті. В більшості роутів використовується спеціальний компонент під назвою RequireAuth, він перевіряє, чи авторизований користувач, якщо ні, то виконується перенаправлення на сторінку авторизації, а якщо так, то буде виконано той компонент, який було передано до нього. Завдяки такому підходу організації маршрутів, можна досягти навігації на сайті без перезавантаження сторінки.

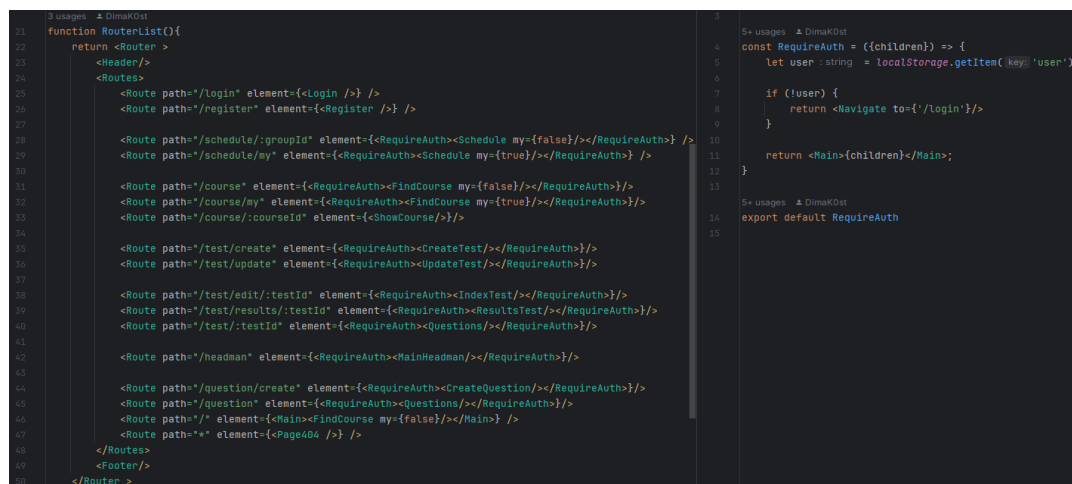


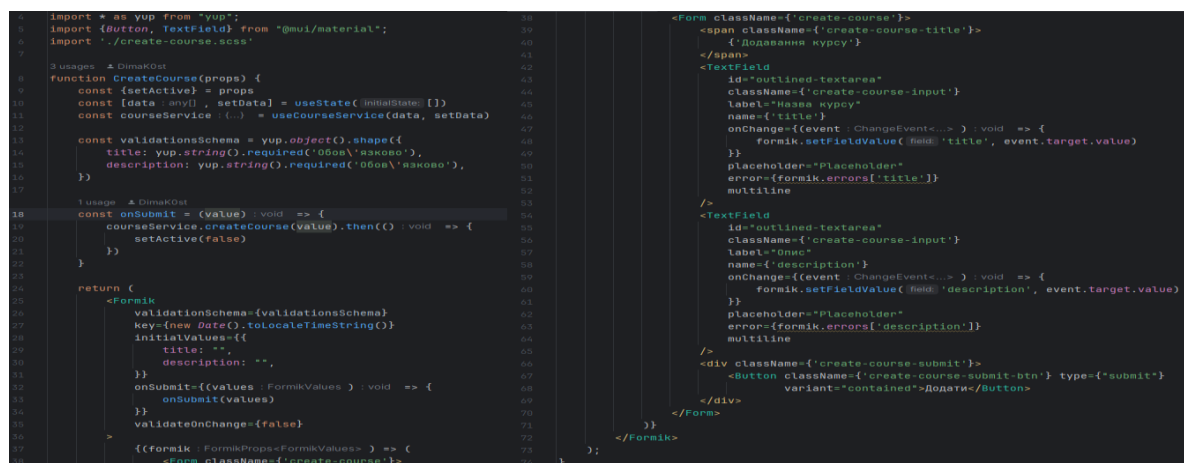
Рисунок 3.17 – Зображення того, як працює роутинг

На рисунку 3.18 зображено код, який використовується схожим чином у формах які займаються створенням або редагуванням даних. В даному підході використано додаткові бібліотеки під назвою «Yup» і «Formik».

Yup – це бібліотека для валідації даних у JavaScript. Вона дає змогу визначати схеми даних і перевіряти їхню відповідність певним умовам. Зазвичай використовується в поєднанні з Formik для валідації даних, які надходять від користувача, також вона може відображати підказки біля тих полів, які введено некоректно.

Formik – це бібліотека для керування формами в React. Вона надає простий інтерфейс для керувати станом форми, обробкою введеної інформації, також вона вмє легко працювати з Yup, що значно полегшує валідацію, та захист від небажаних запитів на бекенд.

При успішному введенні даних користувачем, Formik за допомогою Yup перевіряє дані, і у випадку їх коректності, виконує функцію onSubmit, як відправляє дані до бекенду.



```

import * as yup from 'yup';
import {Button, TextField} from '@mui/material';
import './create-course.scss';

// usage: <DimakKost
function createCourse(props) {
  const {setActive} = props;
  const [data, setData] = useState({});
  const courseService = useCourseService(data, setData);

  const validationSchema = yup.object().shape({
    title: yup.string().required('Обов\'язково'),
    description: yup.string().required('Обов\'язково'),
  });

  // usage: <DimakKost
  const onSubmit = (value) => {
    courseService.createCourse(value).then(() => {
      setActive(false);
    });
  };

  return (
    <Formik
      validationSchema={validationSchema}
      key={new Date().toLocaleTimeString()}
      initialValues={{
        title: '',
        description: '',
      }}
      onSubmit={onSubmit}
      validateOnChange={false}
    >{
      <Formik className='create-course'>
        <div className='create-course-title'>
          <span className='create-course-title'>
            {Додавання курсу}
          </span>
        </div>
        <TextField
          id='outlined-textarea'
          className='create-course-input'
          label='Назва курсу'
          name='title'
          onChange={(event) => {
            formik.setFieldValue('title', event.target.value);
          }}
          placeholder='Placeholder'
          error={formik.errors['title']}
          multiline
        </TextField>
        <TextField
          id='outlined-textarea'
          className='create-course-input'
          label='Опис'
          name='description'
          onChange={(event) => {
            formik.setFieldValue('description', event.target.value);
          }}
          placeholder='Placeholder'
          error={formik.errors['description']}
          multiline
        </TextField>
        <div className='create-course-submit'>
          <Button className='create-course-submit-btn' type='submit'
            variant='contained'>Додати</Button>
        </div>
      </Formik>
    }
  );
}
  
```

Рисунок 3.18 – Приклад коду для створення форми

Також використовується спеціальна бібліотека за допомогою якої можна зробити динамічну назву для вкладок в залежності від відкритої сторінки, та важливого для SEO опису сторінки. На рисунку 3.19 зображено те, як це використовується в проєкті.

```

return (
  <div className={'course'}>
    <Helmet>
      <title>`${data?.data?.title} - EducationalSite`</title>
      <meta name="description" content={data?.data?.description}/>
    </Helmet>
  </div>
)

```

Рисунок 3.19 – Використання бібліотеки Helmet

3.3 Функціонування реалізованої системи

При першому вході на сайт користувач бачить сторінку, зображену на рисунку 3.20. Вона містить перелік всіх курсів та пошук, використовуючи який, можна легко знайти потрібний курс після авторизації. При натисканні на будь-яку кнопку чи посилання, не авторизованого користувача буде відправлено на сторінку, де він зможе авторизуватись, або зареєструватись в системі.

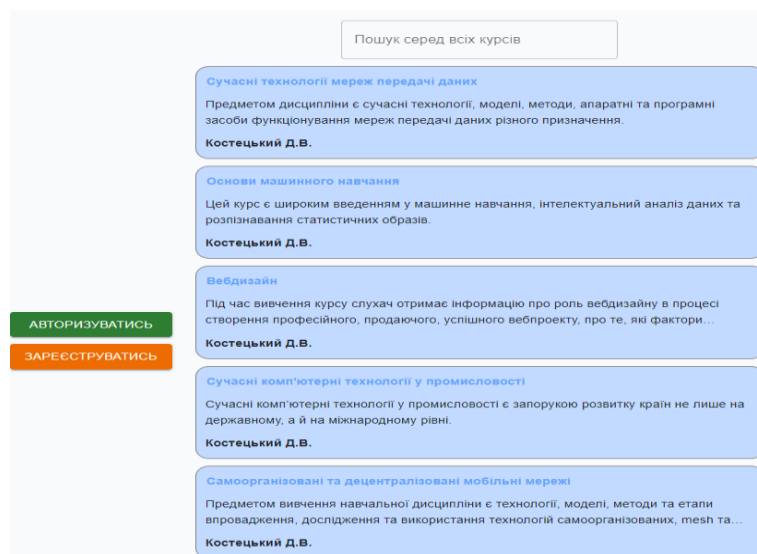


Рисунок 3.20 – Домашня сторінка

На рисунку 3.21 зображено форми авторизації та реєстрації. За їх допомогою можна отримати доступ до основного функціонала, бо після коректно введених даних, браузер отримає токен за яким виконується ідентифікація користувача при будь-якому запиті на сервер, щоб користувач міг зручно користуватись системою, було зроблено спеціальну валідацію за

допомогою якої відображаються некоректно введені дані. Такі перевірки були зроблені на фронтенді, для того, щоб людина моментально отримала інформацію про невірно введену інформацію, без очікування відповіді від сервера, але і на бекенді також було зроблено її, бо зломисник може спробувати за допомогою спеціальних програм в обхід фронтенду відправити зламані дані та порушити відображення на сайті, але з перевіркою, в нього це не вийде і він просто отримає помилку валідації.

Рисунок 3.21 – Форма авторизації та реєстрації

Після успішної авторизації бачимо сторінку, яку продемонстровану на рисунку 3.22. На ній відображаються курси за датою додавання, також реалізовано відображення порціями під назвою пагінація, тобто завантажуються не всі курси за один раз, а спочатку 20, і при гортанні до низу, виконується завантаження наступної двадцятки, і так, поки вони не закінчаться. Також присутній зручний динамічний пошук, який виглядає наче список що випадає, але краще, бо в ньому є пошук, та при початку введення тексту, пропонуються варіанти, які підходять під опис, тому завдяки йому, можна дуже легко відсортувати курси, після чого натиснувши на певний, переглянути його зміст чи підписатись. В лівій секції головної сторінки, знаходиться перелік курсів на які було виконано підписку, для того, щоб не захаращувати екран текстом, було прийнято рішення виводити перші літери кожного зі слів курсу.

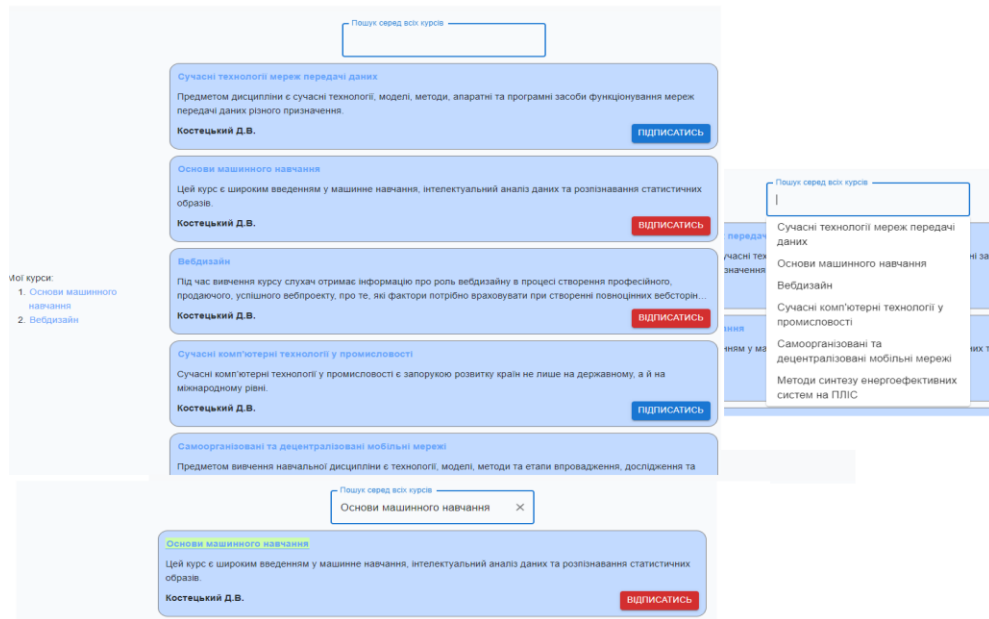
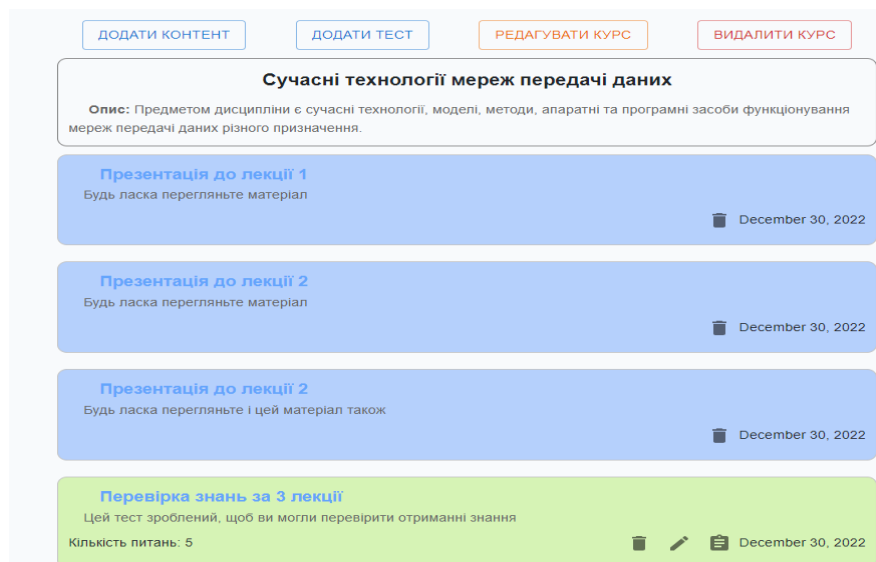


Рисунок 3.22 – Відображення головної сторінки для авторизованого користувача

На рисунку 3.23 зображено те, як відображається курс при переході до нього. Синім кольором відображаються лекційні матеріали, посилання на літературу чи просто інформацію, зеленим кольором відображаються тести.



На рисунку 3.23 – Перегляд курсу

На рисунку 3.24 зображено форми, які викладач може відкрити за допомогою кнопок на сторінці курсу. При створенні тесту можна обрати

кількість питань, які будуть відображатись випадково з усіх створених питань. Також присутня кнопка «Згенерувати за допомогою ШІ», ця кнопка відповідає за те, що вебсистема відправить до API запит з темою та описом тесту, штучний інтелект автоматично згенерує їх, та створить новий тест в системі з питаннями, які надалі можна легко відредагувати, якщо в цьому є необхідність.

The image shows a web interface for an instructor with four distinct forms arranged in a 2x2 grid. Each form has a title at the top, input fields for data entry, and a blue action button at the bottom.

- Додавання матеріалу (Top Left):** Contains three input fields: 'Назва матеріалу', 'Опис', and 'Посилання на матеріал'. The 'ДОДАТИ' button is at the bottom.
- Додавання тесту (Top Right):** Contains three input fields: 'Назва тесту', 'Опис', and 'Кількість питань' (with a dropdown arrow). It features two blue buttons: 'ЗГЕНЕРУВАТИ ШІ' and 'СТВОРИТИ'.
- Редагування курсу (Bottom Left):** Contains two input fields: 'Назва курсу' (with the text 'Сучасні технології мереж передачі даних') and 'Опис' (with a paragraph of text). The 'РЕДАГУВАТИ' button is at the bottom.
- Редагування тесту (Bottom Right):** Contains three input fields: 'Назва курсу' (with the text 'Перевірка знань за 3 лекції'), 'Опис' (with a paragraph of text), and 'Кількість виведення' (with the value '5'). The 'РЕДАГУВАТИ' button is at the bottom.

Рисунку 3.24 – Форми викладача

На рисунку 3.25 зображено створення тестування та додавання до нього питань. Першим етапом йде відкриття курсу й натискання на кнопку «Додати тест», де за допомогою форми потрібно внести наступні дані: назва, опис та кількість питань. Після чого, потрібно обрати тип наповнення питаннями, є два варіанти: згенерувати за допомогою ШІ, або натиснути кнопку «створити» й заповнювати питання вручну. При виборі генерування штучним інтелектом, виконується запит на сервіс Chat Gpt, з якого повертається відповідь, з якої в надалі автоматично створюються питання для тестування. Після успішного додавання, викладач може переглянути створений тест, переглянувши: тему, опис та додані питання з поміченими правильними відповідями.

Додавання тесту

Назва тесту
Перевірка знань за 3 лекції

Опис
Цей тест зроблений, щоб ви могли перевірити отримані знання

Кількість питань
5

ЗГЕНЕРУВАТИ ШІ

СТВОРИТИ

Додавання питання

Назва питання
Який з цих кодів є п'ятирівневим?

Назва питання *
3B/2T

Назва питання *
2B1Q

Назва питання *
RAM5

Назва питання *
MLT-3

ДОДАТИ ВАРІАНТ

СТВОРИТИ ПИТАННЯ

Перевірка знань за 3 лекції

Опис: Цей тест зроблений, щоб ви могли перевірити отримані знання

Який з цих кодів є п'ятирівневим?

- 3B/2T
- 2B1Q
- RAM5**
- MLT-3

Який опис п'яти етапів перетворення даних в процесі інкапсуляції при відправці поштового повідомлення одним комп'ютером іншому є правильним?

- Дані, сегменти, пакети, кадри, біти
- Сегменти, пакети, кадри, біти, дані
- Біти, кадри, пакети, сегменти, дані**
- Сегменти, дані, пакети, кадри, біти

Що з наведеного нижче найкраще описує функцію розширеної команди ping?

- Використовується для відстеження**
- Використовується для діагностики ring-пакету
- Використовується для завдання рамок

Курс цікавий?

- Так**
- Ні

Який тип кабелю кращий

- оптоволоконний кабель**
- коаксимальний кабель
- STP

Рисунок 3.25 – Робота з тестом

На рисунку 3.26 зображено проходження тестування студентом. Студент переходить на сторінку курсу, де обирає певний тест, після чого, відкривається сторінка з питаннями, які обираються випадково з усіх попередньо доданих та з тією кількістю, яку задав викладач, в нашому випадку 5. На рисунку зображено, що потрібно відповісти на 5 випадкових запитань, хоча в самому тесті було створено 8. Після проходження тесту, студентові відображається кількість правильних відповідей, та виконується перенаправлення на головну сторінку курсу.

Питання 1 / 5

Що з наведеного нижче найкраще описує функцію розширеної команди ping?

Використовується для відстеження

Використовується для діагностики ring-пакету

Використовується для завдання рамок

Питання 5 / 5

Курс цікавий?

Так

Ні

Вірних відповідей 2 з 5

Рисунок 3.26 – Проходження тесту

Для того, щоб вносити дані до розкладу, працівник повинен спочатку внести їх до бази даних, форми зазначено на рисунку 3.27. На рисунку 3.28 зображено роботу з розкладом. Працівник деканату може вносити до нього нові дані, видаляти та редагувати. Також зображено форми, які допомагають при роботі з ним.

Рисунок 3.27 – Робота з формами

Група	Предмет	Викладач	Оцінювання	Дія	
KNT-612m	Комп'ютерне 3D-моделювання	Касьян К.М.	Залік	✓ ✕	
KNT-622m	Методи синтезу енергоефективних систем на ПЛІС	Зеленьова І.Я.	Екзамен	✓ ✕	
2	Чисельник	Єфименко М.В.	Залік	✓ ✕	
3	Чисельник і знаменник	Луценко Н.В.	Диференційний залік	✓ ✕	
4	Архітектура і технології вебсервісів			+	
5	Знаменник	Програмування		+	
6				+	
7				+	
8				+	
№	Тип	Предмет	Викладач	Оцінювання	Дія
1	Чисельник	Програмування	Грушко С.С.	Екзамен	✓ ✕
					+

Рисунок 3.28 – Форми для наповнення розкладу

На рисунку 3.29-3.30 зображено те, як сайт відображається на мобільних пристроях, як бачимо, інтерфейс адаптовано до ширини екрана в 300 пікселів, бажано розробляти до 320, але у даному дипломному проєкті вийшло адаптувати ще краще.

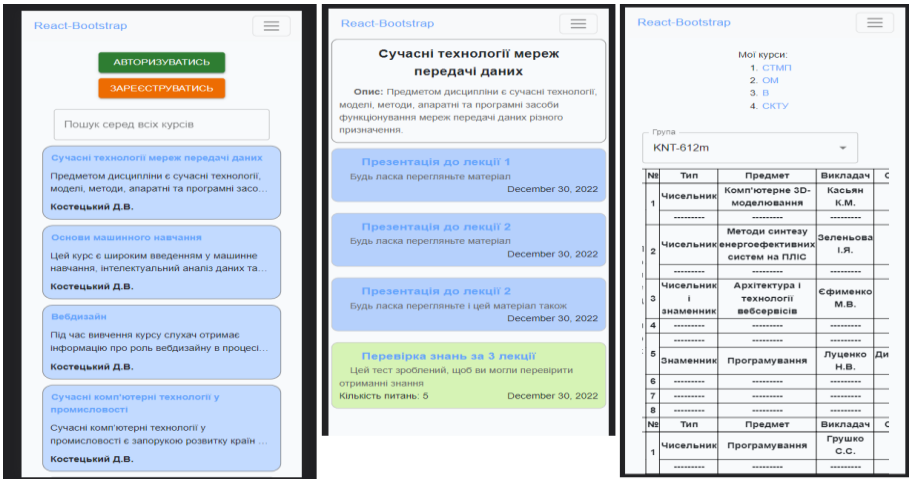


Рисунок 3.29 – Зображення адаптивності сайту

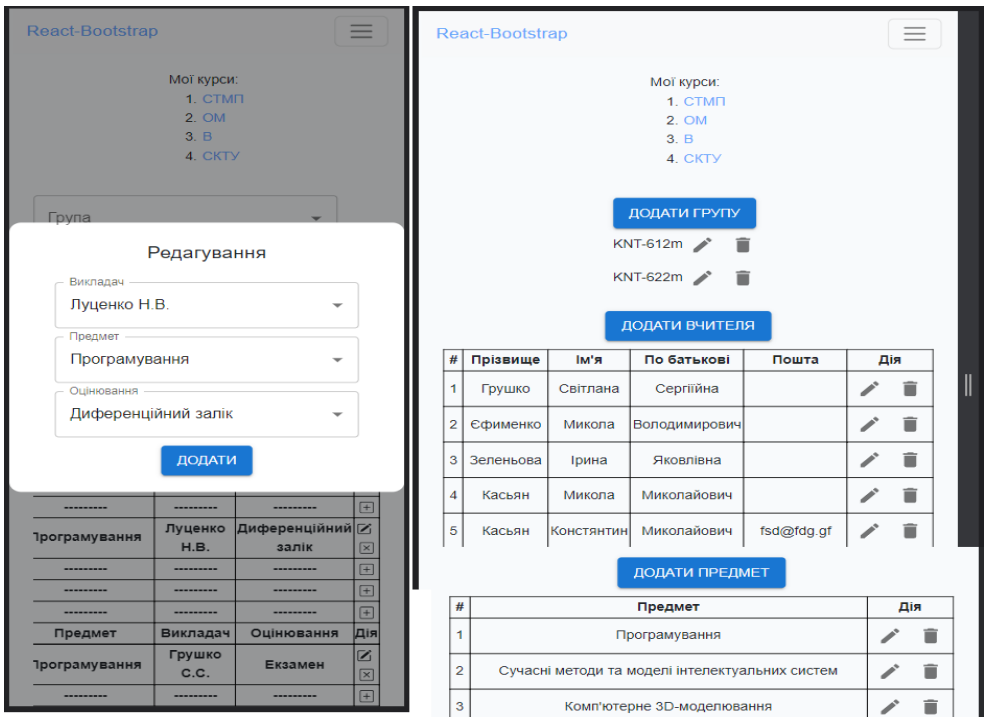


Рисунок 3.30 – Зображення адаптивності сайту

На рисунку 3.31 зображено таблиці, які було створено на сервері для збереження отриманих даних від користувачів.

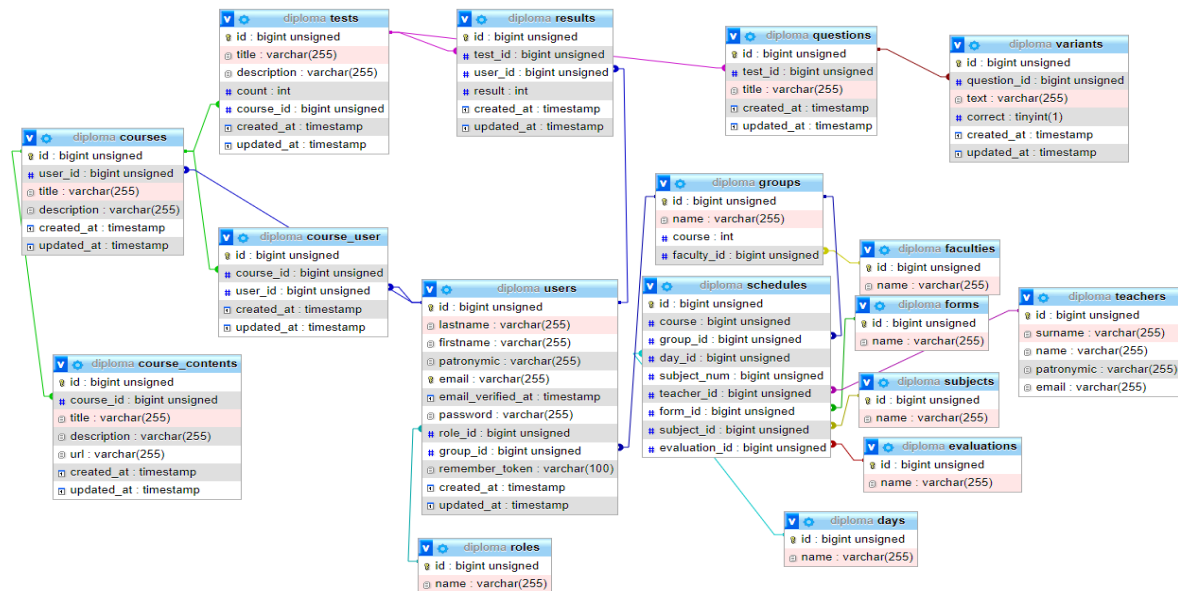


Рисунок 3.31 – Зображення таблиць бази даних

3.4 Висновки за розділом 3

Даний розділ присвячено детальному огляду процесу розробки системи включаючи створення бекенду, фронтенду, реалізацію системи, тестування та валідацію коду.

Для створення бекенду, спочатку було розглянуто налаштування середовища розробки за допомогою Open Server на Windows, після чого описано основні моменти створення проєкту та початкові його налаштування. Наступним було описано важливіші блоки коду та додано пояснення того, як воно працює, розглянуто те, як за допомогою асинхронних черг бекенд відправляє запити на сервіс штучного інтелекту та з отриманої відповіді створює автоматично тести.

Наступним кроком було додавання фронтенду до наявного проєкту за допомогою спеціальних команд та пакетного менеджера, який автоматично налаштовує всі залежності й встановлює допоміжні бібліотеки. Наступним було описано методи за допомогою яких виконується обмін інформацією між клієнтом та сервером, те як генеруються заголовки, роботу з перевіркою даних які вносяться до форм, і те як генерується відображення компонентів.

Розглянуто роботу валідації, компіляції та стиснення коду. Результатом було отримано Повністю перевірений код, який може зламатись лише при помилці логіки, а не синтаксису, також його було стиснуто, що дає приріст по швидкості відкривання сайту, оскільки його вага зменшилася.

При тестуванні системи, було перевірено її захист від можливого нештатного підключення, роботу перевірки форм, головних функціональностей та виявлення можливих помилок. Його проведення дозволило забезпечити високу якість системи та її відповідність визначеним вимогам.

Підбиваючи підсумок цього розділу, можна зробити висновок, що було успішно завершено етап розробки, валідацію коду та тестування. Реалізована система готова до впровадження та використання в реальних умовах.

4 ТЕСТУВАННЯ ТА ВПРОВАДЖЕННЯ ПРОЄКТУ

4.1 Валідація коду

При розробці проєкту для його компіляції і валідації було обрано Vite (це інструмент збірки, який значно покращує інтерфейс розробки, він відразу показує місця, де може бути допущена синтаксична помилка. Тому не валідний код просто не зможе запуститись.

Для забезпечення плавного і швидкого розроблення з мінімальними попередніми налаштуваннями, Vite використовує найдосконаліші функції сучасного JavaScript, такі як модулі ES, за допомогою якого, код мінімізується та доповнюється спеціальними поясненнями для браузера, щоб зібраний код міг відкриватись на старих версіях. Такі принципи дають змогу робити як компіляцію фреймворка, так і використовувати вже готові. Для встановлення не потрібна наявність якихось додаткових плагінів, однак, залежно від специфіки проєкту. Зазвичай встановлення проводиться спільно з одним-двома плагінами, плюс, деяке налаштування під поточний проєкт.

Наприклад, є плагін мінімізації, який стискає код та ділить його на маленькі шматки. Ця мінімізація робиться завдяки тому, що повні назви функцій та змінних змінюється на представлення у вигляді одного або декількох символів, прибираються всі відступи та перенесення на новий рядок, і завдяки цьому ми отримуємо файли меншої ваги та захищений від втручання в його роботу з браузера, бо код перестає бути легким для читання людьми. Наприклад, проект без цієї мінімізації зображено на рисунку 4.1. Він складається з великої купи файлів, які мають велику вагу (якщо були точним, то 144 файли вагою 10 Мб), тому, що знаходиться на етапі розробки, і код ніяк не змінюється при передачі з сервера до браузера, щоб можна було зрозуміти де помилка, а проект з мінімізацією зображено на рисунку 4.2, на ньому видно, що вага та кількість файлів стала менше, а саме 5 файлів вагою приблизно 2.5 Мб.

Название	Статус	Тип	Инициатор	Размер	Время	Каскад загрузки
127.0.0.1	200	docum...	Другое	2.6 kB	125 мс	
css2?family=Nunito:wght@400;...	200	stylesh...	(индекс)	1.4 kB	105 мс	
app.css	200	stylesh...	(индекс)	2.6 kB	143 мс	
reset.css	200	stylesh...	(индекс)	2.6 kB	313 мс	
client	200	script	(индекс)	78.5 kB	11 мс	
app.jsx	200	script	(индекс)	4.6 kB	11 мс	
webrtc-patch.js	200	script	webrtc.js:23	796 B	183 мс	
installHook.js	200	script	prepareinjecti...	427 kB	181 мс	
@react-refresh	200	script	(индекс):19	20.9 kB	3 мс	
react_devtools_backend.js	200	script	prepareinjecti...	534 kB	48 мс	
react.js?v=89748e98	200	script	app.jsx:2	498 B	9 мс	
react-dom_client.js?v=89748e98	200	script	app.jsx:1	1.4 kB	8 мс	
RouterList.jsx?t=1674995382463	200	script	app.jsx:2	22.5 kB	10 мс	
reset.scss	200	script	app.jsx:4	2.5 kB	10 мс	
app.scss	200	script	app.jsx:5	245 kB	12 мс	
react-helmet.js?v=89748e98	200	script	app.jsx:6	38.3 kB	12 мс	
jsx-dev-runtime	200	script	app.jsx:7	518 B	12 мс	
env.mjs	200	script	client:1	3.5 kB	14 мс	
chunk-YP55OVAM.js?v=89748e98	200	script	react.js?v=897...	73.5 kB	5 мс	
chunk-S5KM4IGW.js?v=89748e98	200	script	react.js?v=897...	2.2 kB	4 мс	
chunk-ZRYYW4C.js?v=89748e98	200	script	react-dom clie...	906 kB	18 мс	
react-router-dom.js?v=89748e98	200	script	RouterList.jsx:1	128 kB	17 мс	
Header.jsx	200	script	RouterList.jsx:2	11.7 kB	22 мс	
Page404.jsx	200	script	RouterList.jsx:3	3.5 kB	24 мс	
Footer.jsx	200	script	RouterList.jsx:4	3.6 kB	25 мс	
Register.jsx?t=1674995382463	200	script	RouterList.jsx:5	25.2 kB	25 мс	
Login.jsx?t=1674990189095	200	script	RouterList.jsx:6	12.7 kB	28 мс	

Рисунок 4.1 – Проект без мінімізації

Название	Статус	Тип	Инициатор	Размер	Время	Каскад загрузки
6	200	docum...	Другое	1.6 kB	1.18 ...	
css2?family=Nunito:wght@400;...	200	stylesh...	6	1.4 kB	64 мс	
app.css	200	stylesh...	6	1.6 kB	329 мс	
reset.css	200	stylesh...	6	1.6 kB	948 мс	
app.cb121d54.css	200	stylesh...	6	34.3 kB	877 мс	
app.6e0ef8d2.js	200	script	6	318 kB	2.04 ...	

Рисунок 4.2 – Мінімізований проект

4.2 Функціональне тестування системи

Розроблена система має спеціальні захисні механізми, які використовуються при її користуванні. Наприклад, при першому відкритті сайту, користувач бачить перелік курсів, до яких він може доєднатись, та при натисканні на певний курс він може переглянути його наповнення, але не може з ним взаємодіяти, бо при натисканню буде автоматичне перенаправлення на сторінку, де він зможе авторизуватися та почати користуватися всіма функціями – це можна назвати першою лінією захисту.

Першим було протестовано те, як працює авторизація, якщо користувач намагається зламати сайт. Він зіткнеться з тим, що тут використовується авторизація за допомогою токена, тому щоб отримати доступ до скритих функцій сайту, потрібно у будь-якому випадку авторизуватись, на рисунку 4.3 зображено це.

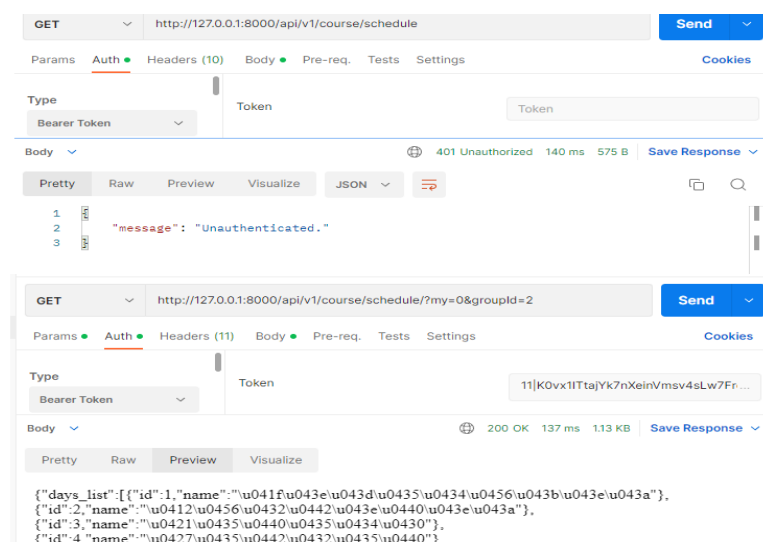


Рисунок 4.3 – Тестування роботи токена

Також на сайтах бувають незручні форми, де при внесенні даних і після підтвердження форма оновлювалась і виводились помилки з видалення попередніх даних. У розроблювальній системі присутня подвійна валідація:

перша – це ще перед надсиланням даних до сервера, дані перевіряються браузером, і у випадку помилки одразу її відображають, друга – це безпосередньо на сервері виконується перевірка, щоб позбавитись спеціальних так званих уколів, за допомогою яких виконується злам сайту. У розробленій системі кожна форма має ці дві валідації. Приклад зображено на рисунку 4.4.

Рисунок 4.4 – Тестування форм

Наприклад, на рисунку 4.5 у лівій частині зображено валідацію у браузері, а у правій частині вона вимкнена, і тепер сервер повідомляє нас про помилки.

Рисунок 4.5 – Тестування форм

Зробимо тестування роботи основного функціоналу сайту. Спочатку створюємо новий курс, в який в подальшому буде додаватись корисна інформація, як зображено на рисунку 4.6.

Рисунок 4.6 – Створення курсу

Після успішного створення, він одразу відобразиться у списку, бо надсилається асинхронний запит до серверу, і користувачу навіть не потрібно оновлювати сторінку. На рисунку 4.7 зображено щойно створений курс, на який можна підписатись, якщо потрібно взаємодіяти з інформацією, яка є в ньому. На рисунку 4.8 зображено те, як він відображається при підписці, і в списку підписок.

Рисунок 4.7 – Перевірка створення курсу

Рисунок 4.8 – Перевірка функціоналу підписки

На рисунку 4.9 виконується заповнення форми, і підготовка запиту для відправки, за допомогою кнопки «Згенерувати ШІ», надсилаються дані на бекенд, які буде конвертовано і відправлено до сервісу Chat Gpt. На рисунку 4.10 зображено інформацію з файлу який займається логуванням роботи, і як бачимо отриманий результат від штучного інтелекту, було успішно оброблено на нормалізовано бекендом. З цих даних наступним кроком буде створення саме тесту.

Рисунок 4.9 – Заповнення форми для створення тесту з використанням ШІ

```
App\Http\Dtos\GeneratedAiQuestionsDTO {#536 // app\Jobs\GetDataFromAi.php:43
+questions: array:8 [
  0 => App\Http\QuestionDTO {#537
+question: "Яким чином ви можете використовувати міграції в Laravel?"
+options: array:4 [
    0 => App\Http\OptionDTO {#538
+variant: "Запуск команди php artisan migrate"
+correct: 1
    ]
    1 => App\Http\OptionDTO {#539
+variant: "Вручну змінювати базу даних"
+correct: 0
    ]
    2 => App\Http\OptionDTO {#540
+variant: "Використовувати сторонні плагіни"
+correct: 0
    ]
    3 => App\Http\OptionDTO {#541
+variant: "Не існує можливості використовувати міграції в Laravel"
+correct: 0
    ]
  ]
}
  1 => App\Http\QuestionDTO {#542
+question: "Як створити маршрут в Laravel?"
+options: array:4 [
    0 => App\Http\OptionDTO {#543
+variant: "В файлі routes/web.php"
+correct: 1
    ]
    1 => App\Http\OptionDTO {#544
+variant: "В файлі config/routes.php"
+correct: 0
    ]
  ]
}
```

Рисунок 4.10 – Зображення обробленої відповіді від до API

На рисунку 4.11 зображено роботу бекенду, тобто все пройшло успішно, перевіривши тест було зрозуміло, що він коректний, на все це пішло близько 5 хвилин. На рисунку 4.12 показано головну сторінку, на якій з'явився створений тест, при натисканні на нього, він безпосередньо запускається що зображено нижче. Після завершення викладач може переглянути всі результати на спеціальні сторінці, де показана оцінка в 5 та 100 бальній системі, та час складання тесту.

ДОДАТИ ПИТАННЯ

Перевірка закріплених знань про Laravel
Опис: Тест в якому студенти можуть закріпити свої навички після вивчення курсу Laravel та мови програмування php, тести середньої складності.

Яким чином ви можете використовувати міграції в Laravel?

- 1. Запуск команди `php artisan migrate`
- 2. Вручну змінювати базу даних
- 3. Використовувати сторонні плагіни
- 4. Не існує можливості використовувати міграції в Laravel

Як створити маршрут в Laravel?

- 1. В файлі `routes/web.php`
- 2. В файлі `config/routes.php`
- 3. Використовуючи `artisan` команду
- 4. Змінюючи файли ядра

Що таке Eloquent в Laravel?

- 1. Один із шаблонів проектування
- 2. ORM (Object-Relational Mapping) для роботи з базою даних
- 3. Мова програмування в Laravel
- 4. Контроль версій для Laravel

Яким чином можна валідувати дані в Laravel?

- 1. За допомогою вбудованих функцій PHP
- 2. Використовуючи `middleware`
- 3. За допомогою класу `Validator`
- 4. Вручну перевіряючи дані в контролері

Рисунок 4.11 – Перевірка створеного тесту

ДОДАТИ КОНТЕНТ ДОДАТИ ТЕСТ РЕДАГУВАТИ КУРС ВИДАЛИТИ КУРС

Сучасні інформаційні технології
Опис: Цей курс є гарним стартом, якщо є бажання вивчити веб-розробку

Перевірка закріплених знань про Laravel
Тест в якому студенти можуть закріпити свої навички після вивчення курсу Laravel та мови програмування php, тести середньої складності.
Кількість питань: 8

Питання 1 / 8
Як встановити новий пакет в Laravel за допомогою Composer?

- composer install назва-пакету
- php artisan install назва-пакету
- composer install назва-пакету
- php artisan add package назва-пакету

Питання 7 / 8
Як створити маршрут в Laravel?

- В файлі `routes/web.php`
- В файлі `config/routes.php`
- Використовуючи `artisan` команду
- Змінюючи файли ядра

Вірних відповідей 4 з 8

Тест: Перевірка закріплених знань про Laravel

#	Прізвище	Ім'я	По батькові	Група	Оцінка	Час
1	Костецький	Дмитро	Віталійович	KNT-612m	2/50%	November 22, 2023 2:22 AM

Рисунок 4.12 – Перевірка повної роботи тестування

4.3 Розрахунок економії часу при створенні тесту

Якщо розглядати навчання в загальному, то створення тестів є важливою частиною цього процесу, бо за їх допомогою можна зрозуміти, наскільки добре засвоїли матеріал студенти, і що потрібно пропрацювати більш детально. У звичному представленні створення тесту, викладачу потрібно витратити час на те, щоб створити питання і підібрати до нього варіанти відповідей, цей процес окрім витрати часу, також сильно втомлює. Але на заміну цьому, є кращий варіант – це генерування тестів за допомогою ШІ. Розглянемо про кожен з цих варіантів:

- коли викладач працює над тестом, він витрачає велику купу часу, бо йому потрібно опираючись на текст його лекцій вигадати питання та варіанти відповідей до нього. При такому варіанті роботи, присутній індивідуальний підхід, бо викладач може регулювати складність питань, та і тест буде особливим. Проте цей процес також має певні обмеження, у вигляді робочого часу, за який потрібно зробити велику купу іншої роботи, і його може не вистачити на написання якісного тесту, особливо коли у викладача велика кількість дисциплін;

- коли за цю роботу береться штучний інтелект, викладачу потрібно лише написати коректну назву, і докладно описати, що повинен перевірити тест: що буде створено, якої складності, і вказати кількість питань. Після чого цей тест буде згенеровано автоматично за лічені секунди, і після перевірки некоректні або не цікаві питання можна просто видалити.

Якщо порівняти ці два підходи, то можна зрозуміти, що при автоматичній генерації, витрачання часу викладача знижується, також не з'являється втома, що добре позначається на його продуктивності. У таблиці 1.1 показано порівняння цих двох підходів.

Таблиця 4.1 – Порівняння створення тестів двома підходами

Варіант	Створення викладачем	Автоматична генерація
Час	Повільно	Швидко
Точність запитань	Залежить від кваліфікації викладача і уважності	Залежить від опису запиту
Складність створення	Залежить від досвіду викладача та його підготовки	Миттєва генерація, але потрібен контроль якості
Зворотній зв'язок	Детальний і контекстний зворотний зв'язок	Просто генерується без пояснень
Різноманітність запитань	Залежить від уяви викладача	Максимально унікальні питання
Інновації та технології	Залежить від інтересу викладача в нових методах і технологіях	Використовує передові технології

Таким чином, як у викладачів, так і у ШІ є свої переваги та обмеження у створенні тестів.

Розрахуємо за розробленою математичною моделлю економію часу при створенні тесту викладачем з використанням ШІ і без.

Час, який витрачає викладач на те, щоб розробити тест, залежить від того, який у нього стаж, якої складності потрібно зробити.

Припустимо, що T_v – це час створення одного запитання тесту викладачем, розрахований у хвилинах, $N_{\text{заг}}$ – загальна кількість питань у тесті. Нехай час створення запиту для створення автоматичного тесту ШІ задамо як $T_{\text{зап}}$. При цьому вважатимемо, що $T_{\text{зап}} = T_v$. Нехай час, витрачений викладачем на перевірку автоматично створеного ШІ одного запитання задано $T_{\text{пер}}$. Припустимо, що $T_{\text{пер}} = 0,2 * T_v$. Нехай час, який буде витрачено викладачем на корегування цих запитань, задано $T_{\text{кор}}$. Припустимо, що $T_{\text{кор}} = 0,5 * T_v$, а кількість запитань, які потребують корегування $N_{\text{кор}}$, дорівнюватиме $N_{\text{кор}} = 0,35 * N_{\text{заг}}$.

У такому випадку загальний час, витрачений викладачем на розробку $N_{\text{заг}}$ питань тесту самостійно без використання ШІ розраховується за формулою 4.1:

$$T_{в_заг} = T_{в} * N_{заг} \quad (4.1)$$

А цей же час, витрачений на розробку $N_{заг}$ питань тесту з використанням штучного інтелекту розраховується за формулою 4.2:

$$T_{ші_заг} = T_{зап} + T_{кор} * N_{кор} + T_{пер} * N_{заг} \quad (4.2)$$

Оскільки час на генерацію запитань ШІ дуже малий, то не враховуємо це у загальному розрахунку.

Якщо загальний витрачений час $N_{заг}$, переписати з урахуванням наведених вище припущень, то отримаємо, що вираз за формулою 4.2 перетворюється у вираз за формулою 4.3:

$$T_{ші_заг} = T_{в} + 0,2 * T_{в} * N_{заг} + 0,5 * T_{в} * 0,35 * N_{заг} \quad (4.3)$$

Спростивши вираз за формулою 4.3, отримаємо:

$$\begin{aligned} T_{ші_заг} &= T_{в} + 0,2 * T_{в} * N_{заг} + 0,175 * T_{в} * N_{заг} = T_{в} + 0,375 * T_{в} * N_{заг} = \\ &= \left(\frac{T_{в}}{N_{заг}} + 0,375 * T_{в} \right) * N_{заг} \end{aligned} \quad (4.4)$$

Це означає, що чим більше буде створено завдань у тесті, тим меншим буде вплив часу на створення запиту до ШІ у загально витраченому часі для створення усіх завдань для тесту.

Розрахуємо відсоток зекономленого часу E при використанні ШІ для створення тестів за формулою 4.5.

$$E = \frac{T_{в_заг} - T_{ші_заг}}{T_{в_заг}} * 100\% \quad (4.5)$$

Перетворимо формулу 4.5 з урахування формул 4.1 та 4.4, отримаємо

результат, представлений формулою 4.6:

$$E = \frac{T_{в_заг} - T_{ші_заг}}{T_{в_заг}} = \frac{T_{в} * N_{заг} - T_{в} + 0,375 * T_{в} * N_{заг}}{T_{в} * N_{заг}} = (0,625 - \frac{1}{N_{заг}}) * 100\% \quad (4.6)$$

Таким чином, із формули 4.6 видно, що чим більшою буде кількість запитань у тесті, тим ефективність застосування ШІ буде вище. При цьому, у будь-якому разі вона не перевищуватиме 62,5 %.

Розглянемо це на прикладі. Припустимо, що на створення одного запитання викладач витрачає в середньому 5 хвилин, і тест складається із 25 запитань. Тоді економія часу у відсотках при застосуванні ШІ складатиме $(0,625 - 0,04) * 100\% = 58,5\%$, що означає, що викладач витратить на створення тесту на 58,5% менше часу.

Перерахуємо ці ж значення за формулами 4.1 та 4.4. Час, який викладач витратив на створення 25 питань тесту самостійно, без використання штучного інтелекту дорівнюватиме $T_{в_заг} = 5 * 25 = 125$ хв.. Загальний час, витрачений на створення 25 питань тесту з використанням ШІ дорівнюватиме $T_{ші_заг} = 5 + 0,375 * 5 * 25 = 51,875$ хв. Тоді час, який зекономить викладач дорівнюватиме $125 - 51,875 = 73,125$ хв., що і є 58,5 % від витраченого часу без використання ШІ. На рисунку 4.13 за допомогою графіка зображено порівняння витрат часу цих двох підходів.

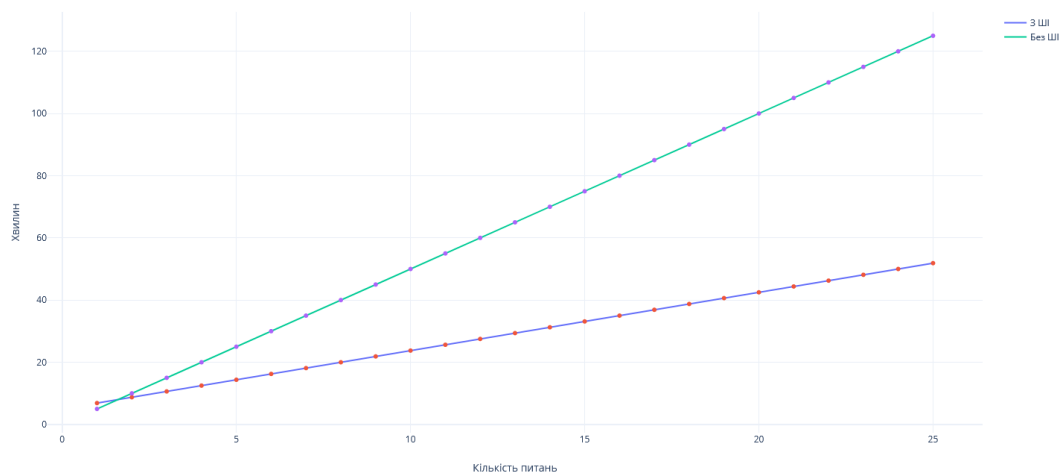


Рисунок 4.13 – Залежність часу від кількості питань

Таким чином, розроблена математична модель є адекватною. Слід також зазначити, що припущення, застосовані у даній моделі базуються на опитуванні викладачів ВНЗ спеціальності 123 Комп'ютерна інженерія.

Проведене практичне тестування функціоналу виявило, що система створила тести з ідеальною якістю, які не потребують жодних корекцій варіантів відповідей. Загалом, для генерації тестів та перевірки відповідей на 25 питань знадобилося всього близько 20 хвилин. Отримані результати свідчать про великий потенціал цієї інновації для практичного використання.

4.4 Порівняння розробленої системи з Moodle

Проаналізувавши існуючі аналоги в інтернеті, для порівняння з розробленою системою, було обрано одну з найпопулярніших під назвою Moodle. В порівнянні проаналізуємо те, як багато вони споживають інтернет трафіку при переходах між сторінками.

При першому відкритті розробленої системи, виконується завантаження всієї клієнтської частини, на рисунку 4.14 зображено вагу головної сторінки, яка складає 2.8 Мб.

Name	Status	Type	Initiator	Size	Time	Waterf
css2?family=Nunitow...	200	styles...	(index)	1.4 kB	19 ms	
app.css	200	styles...	(index)	39.8 kB	152 ms	
reset.css	200	styles...	(index)	39.9 kB	293 ms	
app.258883ea.css	200	styles...	(index)	202 kB	516 ms	
app.f1a49afc.js	200	script	(index)	1.5 MB	520 ms	
stylesheets?v=165749...	200	styles...	(index)	243 kB	407 ms	
javascript?v=16574956...	200	script	(index)	366 kB	514 ms	
webrtc-patch.js	200	script	webrtc.js:23	796 B	197 ms	
css?family=Nunito	200	styles...	app.25888...	1.3 kB	12 ms	
data:image/png;base...	200	png	stylesheets...	(me...	0 ms	
bootstrap.6d55511b.js	200	script	app.f1a49a...	135 kB	9 ms	
search?my=1&search=	200	xhr	app.f1a49a...	5.1 kB	191 ms	
search?my=0&search=	200	xhr	app.f1a49a...	7.1 kB	406 ms	
open?op=get&id=X35...	200	xhr	javascript?...	32.3 kB	271 ms	
open?op=get&id=X78...	200	xhr	javascript?...	33.4 kB	188 ms	
nunito-latin-400-norm...	200	font	css2?family...	14.8 kB	36 ms	
data:image/svg+xml...	200	svg+...	app.25888...	(me...	0 ms	
22 requests 2.8 MB transferred 2.8 MB resources Finish: 19.32 s						

Рисунок 4.14 – Інформація про головну сторінку розробленої системи

Розглянемо те, як швидко завантажується конкурентна система Moodle. На рисунку 4.15 зображено, що при завантаженні головної сторінки, вона займає 2.6 Мб, що в порівнянні з розробленою системою майже однаково.

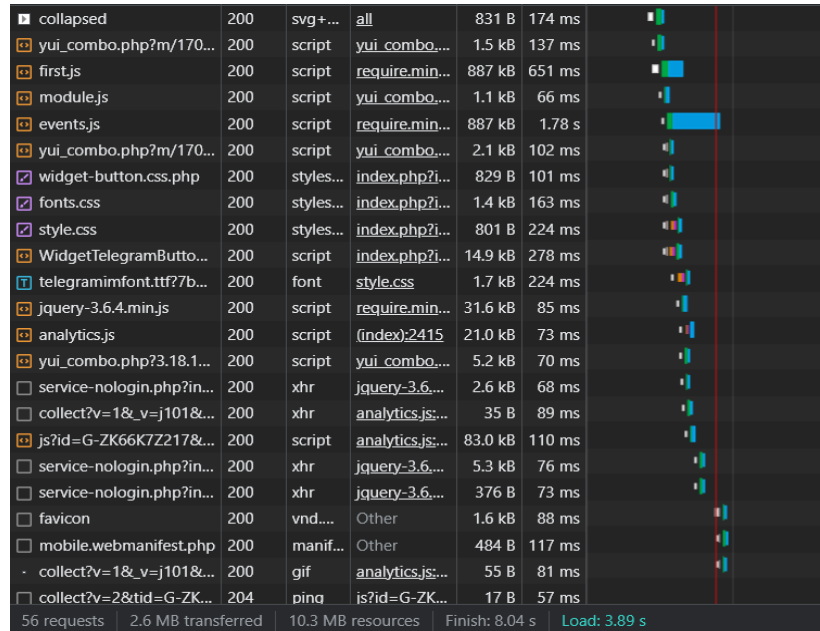


Рисунок 4.15 – Вага початкової сторінки Moodle

На вагу сайту впливає багато чинників, але головним є те, що система Moodle використовує серверну генерацію. Тобто вона отримує з бекенду повне представлення сторінки, а це досить ресурсний запит, бо передається велика купа різних допоміжних HTML тегів. І так аналогічно відбувається кожного разу, коли відкривається кожна наступна сторінка. При поганому з'єднанні з інтернетом, такий сайт буде вантажитись дуже довго, або навіть не зможе завантажитись взагалі. В той же час розроблювальна система, використовує зв'язок, в якому надсилаються асинхронні запити, які передаються в форматі JSON, і відповідно не мають зайвої інформації, тому така система витрачає менше пам'яті за рахунок того, що не вантажить при кожному переході сторінки з нуля, бо вони всі були завантажені при першому візиті сайту.

Для більш наглядного прикладу, було виконано 5 переходів між різними сторінками в цих двох системах, і результат зображено на рисунках 4.16-4.17.

	51 запит	3,23 МБ / 3,25 МБ передано	Передано за: 50,46 с	DOMContentLoaded: 680 мс	load: 832 мс
---	----------	----------------------------	----------------------	--------------------------	--------------

Рисунок 4.16– П'ять запитів в розробленій системі

	36 запитів	9,73 МБ / 2,39 МБ передано	Передано за: 1,99 с	DOMContentLoaded: 518 мс	load: 2,00 с
	56 запитів	10,71 МБ / 2,77 МБ передано	Передано за: 3,63 с	DOMContentLoaded: 1,57 с	load: 2,93 с
	37 запитів	10,78 МБ / 2,92 МБ передано	Передано за: 3,34 с	DOMContentLoaded: 739 мс	load: 1,98 с
	31 запит	9,86 МБ / 2,41 МБ передано	Передано за: 2,55 с	DOMContentLoaded: 705 мс	load: 1,96 с
	26 запитів	9,77 МБ / 2,38 МБ передано	Передано за: 2,42 с	DOMContentLoaded: 699 мс	load: 2,13 с

Рисунок 4.17 – Виконання п'яти переходів на сайті Moodle

З рисунків видно, що так як розроблена система обмінюється з бекендом інформацією через AJAX, то сайт завантажується одного разу, і при переході просто надсилаються запити, які додають вагу до першого відкриття, а в Moodle, так як серверна генерація, при переході на нову сторінку, запити повторюються та додається нова інформація. На рисунку 4.18 продемонстровано графік апроксимованого витраченого трафіку розробленою системою, а на рисунку 4.19 зображено порівняння їх обох, щоб можна було побачити різницю.

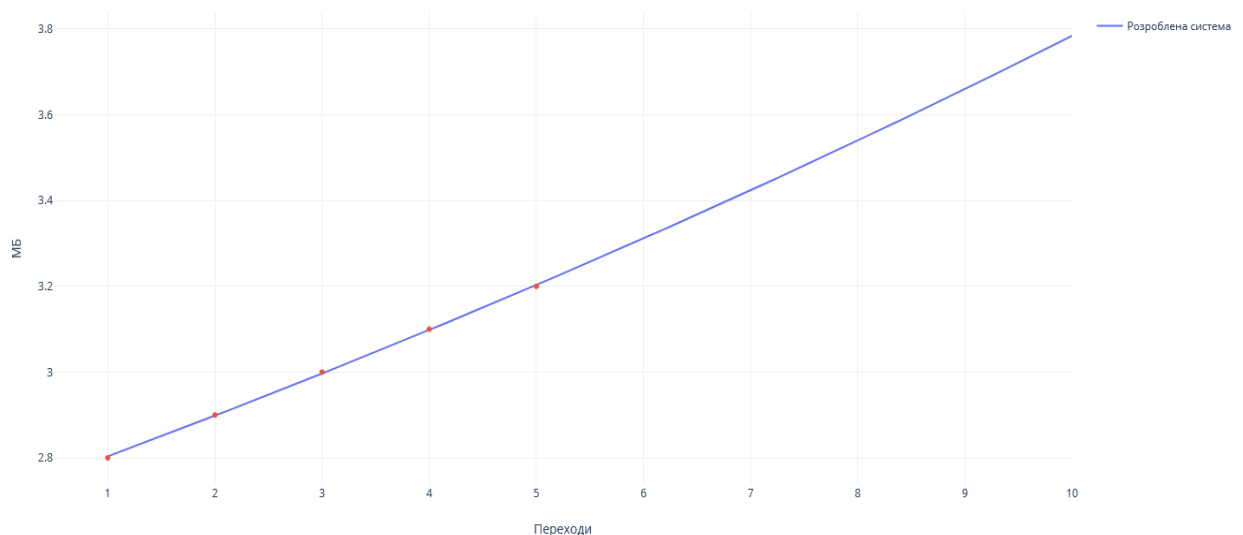


Рисунок 4.18 – Кількість витрачених Мб інтернету розробленою системою

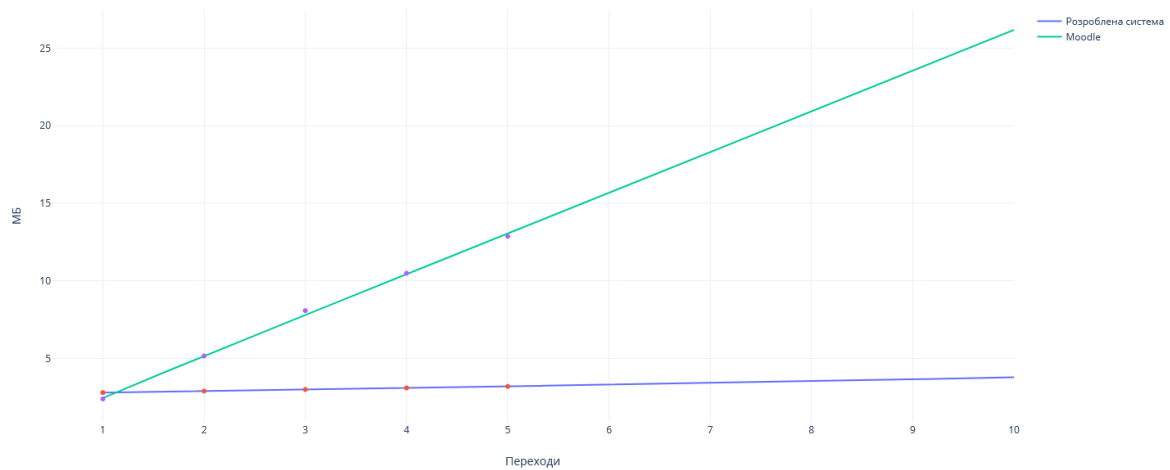


Рисунок 4.19 – Порівняння витрат Мб інтернету

В даному порівнянні було взято вагу лише запитів, оскільки на ці заміри могли вплинути зображення на сайті, тому вони були виключені для точності. Тому можна зробити висновок, що розроблена система є більш оптимізованою, для користувачів з поганим інтернетом ніж Moodle.

4.5 Оптимізація роботи системи

Семантичне ядро – це спеціально відібрані слова або їх сполучення, на основі яких виконується пошук необхідної інформації серед всіх сайтів, і на їх основі обираються найбільш схожі до запиту. Їх також називають ключові слова, фрази або коротко, ключі [38].

Використовуючи спеціальні сайти, за допомогою яких можна подивитись популярність тих чи інших ключових запитів, було виділено ті, які краще всього описують дану систему, тому створення ключових слів було створено відштовхуючись від цих запитів і результат зображено на рисунку 4.20.

онлайн навчання	157 309	навчання мови онлайн безкоштовно	3 099
онлайн навчання безкоштовно	37 608	онлайн навчання манікюру	3 032
курси онлайн навчання	13 252	онлайн навчання +для дітей	2 908
школа онлайн навчання	10 632	навчання онлайн професіям	2 792
навчання нуля онлайн безкоштовно	7 928	обучение онлайн год	2 714
навчання мов онлайн	7 781	дистанційне навчання	839
англійська онлайн навчання	7 174	дистанційному навчанні	839
дистанційне навчання онлайн	4 741	школа дистанційного навчання	96
англійська мова онлайн навчання	4 163	дистанційне навчання школи	81
курси онлайн навчання безкоштовно	4 032	школи дистанційного навчання	81
платформи +для онлайн навчання	3 894	про дистанційне навчання	75
клас онлайн навчання	3 789	дистанційне навчання +в школі	70
самостійне онлайн навчання	3 692	дистанційна форма навчання	48
безкоштовне навчання онлайн +для початківців	3 561	технологія дистанційного навчання	45
гра онлайн навчання	3 411	дистанційне навчання +в Україні	39
самостійне безкоштовне навчання	3 200	організація дистанційного навчання	38
онлайн		сайт дистанційного навчання	32
		дистанційне навчання сайт	32

Рисунок 4.20 – Семантичне ядро сайту

У розробленій системі використовується спеціальна бібліотека, за допомогою якої кожна сторінка має свої певні ключові слова, заголовки та опис. Наприклад, для головної сторінки було обрано наступні ключові слова та опис, вони зображені на рисунку 4.21.

```
<meta name="keywords" content="онлайн навчання, онлайн навчання безкоштовно, курси онлайн навчання, школа онлайн навчання, навчання нуля онлайн безкоштовно, навчання мов онлайн, англійська онлайн навчання, дистанційне навчання онлайн, дистанційне навчання, дистанційне, навчання, школа дистанційного навчання, дистанційне навчання школи, школи дистанційного навчання, школа, університет, технікум, EducationalSite, Educational, educational"/>
<meta name="description" content="Безкоштовний сайт за допомогою якого можна виконувати дистанційне навчання у будь-якій точці світу та для будь-якої людини." />
```

Рисунок 4.21 – Ключові слова та опис

На рисунку 4.22 зображено, як сайт представляється іншим, якщо його відправити повідомленням у месенджері Telegram.

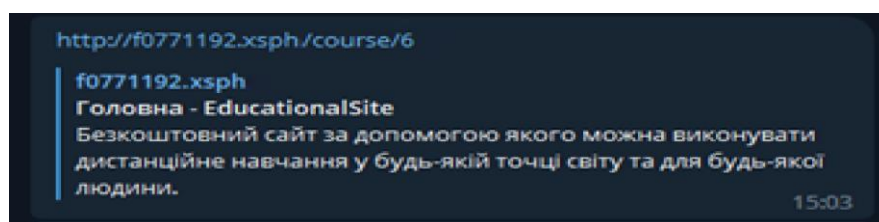


Рисунок 4.22 – Відображення інформації про сайт у месенджері Telegram

4.6 Просування системи в мережі Інтернет

Сайт був проіндексований та доданий до пошукової системи Google за допомогою сайту Search Console. Індексція сторінки – це процес збору, під час якого пошукові системи перевіряють, додають або змінюють наявну інформацію про контент який розміщено на сайті. Вона виконується в автоматичному режимі за допомогою спеціальних роботів, але для того, щоб вони правильно зробили свою роботу, потрібно створити наступні 2 файли:

- sitemap.xml – це файл в якому зберігаються посиланнями на сторінки сайту, і завдяки ньому, пошукові системи можуть отримати актуальну структуру (карту) сайту. В основному його використовують, якщо на сайті: велика кількість сторінок, окремі сторінки, на які можна перейти тише без навігації, глибока вкладеність. Цей файл створюється автоматично за допомогою сайтів або бібліотек. Приклад цього файлу представлено на рисунку 4.23;

```
<url>
  <loc>http://f0771192.xsph/</loc>
  <lastmod>2020-12-08T12:08:25+00:00</lastmod>
  <priority>1.00</priority>
</url>
<url>
  <loc>http://f0771192.xsph/course/6</loc>
  <lastmod>2023-01-27T12:08:25+00:00</lastmod>
  <priority>0.80</priority>
</url>
<url>
  <loc>http://f0771192.xsph/course/5</loc>
  <lastmod>2023-01-27T12:08:25+00:00</lastmod>
  <priority>0.80</priority>
</url>
<url>
  <loc>http://f0771192.xsph/course/4</loc>
  <lastmod>2023-01-27T12:08:25+00:00</lastmod>
  <priority>0.80</priority>
</url>
```

Рисунок 4.23 – Приклад файлу sitemap.xml

- robots.txt містить в собі певні інструкції, які використовують роботи пошукових систем, і в них описано, які сторінки можна відображати в результатах пошуку, а які потрібно приховати. В першу чергу, це інструмент, який дозволяє скоротити кількість запитів від сторонніх сервісів до сайту і тим самим знизити навантаження на нього.

Перевірити індексацію сайту можна за допомогою гугл-консолі у вкладці «Перевірка URL», приклад зображено на рисунку 4.24.

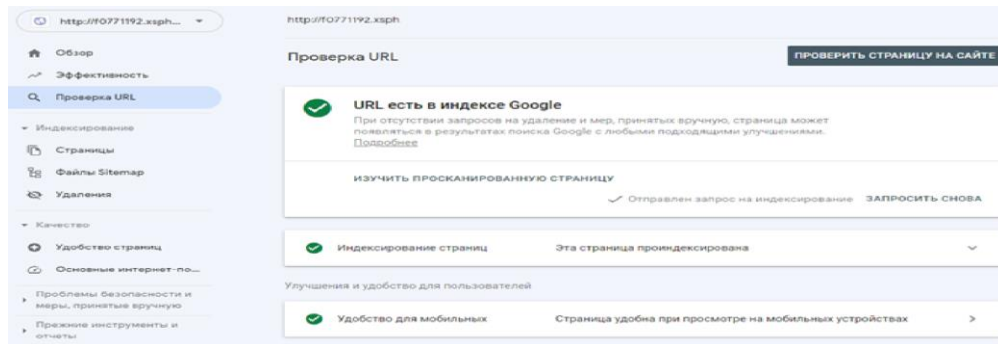


Рисунок 4.24 – Індексція сайту

4.7 Висновки за розділом 4

В даному розділі описується про оптимізацію в використанні семантичного ядра та генерації ключових слів для сайту які впливають на підвищення його рейтингу та відповідності пошуковим запитам. Завдяки використанню семантичного ядра, можна налаштувати привабливий попередній перегляд сайту, при пересиланні його посилання в тому ж Telegram, і це допомагає створити не лише ефективний, але й доступний і зручний для користувачів ресурс.

Наступним кроком було розглянуто просування системи в мережі Інтернет, завдяки тому, що було налаштовано індексацію сайту використовуючи sitemap.xml та robots.txt, а також виконано перевірку її роботи через Google Console, де відображається стабільність та видимість системи у мережі.

Підсумком можна сказати, що ефективне впровадження проєкту не закінчується на етапі розробки коду, а вимагає систематичного підходу до оптимізації та просування для створення стійкої та популярної системи в Інтернеті, готової відповідати вимогам користувачів та досягати поставлених цілей.

ВИСНОВКИ

Таким чином, у дипломному проєкті розроблено систему дистанційної освіти, яка дозволяє вдосконалити навчальний процес та підвищення зручність проведення освітнього процесу викладачам і студентам шляхом інтеграції у систему штучного інтелекту для створення тестів.

Науковою новизною є те, що система має унікальний інноваційний функціонал, який дозволяє легко створювати тестування автоматично, використовуючи штучний інтелект. Завдяки розробленій математичній моделі можна розрахувати, що економія часу при використанні ШІ для створення тесту може сягати майже 62,5%. Це допомагає значно зменшити витрату часу викладача. Завдяки такій можливості, можна створювати тести різних рівнів складності, що дозволяє виявляти слабкі місця в знаннях та корегувати навчальний процес більш ефективно.

Під час написання дипломного проєкту було проведено аналіз предметної області, визначено термінологію, ролі та процеси в середовищі дистанційної освіти. Чітко розмежовано можливості та розглянуто вимоги до користувачів з різними ролями, визначено основний функціонал системи, враховано можливе легке розширення функціоналу у майбутньому.

Перед розробкою проведено, було проаналізовано різні сучасні інструменти та обрано фаворитів, а саме фреймворк Laravel, бібліотеку React та сервіс ChatGpt для користування штучним інтелектом. Обрано монолітну архітектуру, що найкращим чином підходить для системи такого типу. Результатом використання цих інструментів, було реалізовано повно функціональну систему дистанційного навчання з інтеграцією штучного інтелекту, яка повністю відповідає завданням, які були поставленні на початку проєкту.

Після розробки виконано комплексне тестування, валідацію та мінімізацію коду, покращивши вагу та швидкість завантаження сайту. Експеримент зі

створенням тесту за допомогою ШІ, підтвердив припущення та ефективність системи, а це означає, що поставлену мету роботи досягнуто.

Тестове впровадження проєкту в інтернет включало опис інструкцій для пошукових роботів, активацію індексації у пошуку Google та використання семантичного ядра для підбору ключових слів для покращення пошукової видимості сайту.

Отриманий результат свідчить, що система готова до впровадження, відповідає поставленим завданням, має потенціал для подальшого розвитку, сприяючи полегшенню дистанційного навчання для отримувачів освіти та викладачів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Жабенко О. Управлінський персонал університету середньої та нижчої ланки: процедура та критерії добору – 2020. – Т. 31. – №. 4. – С. 30-47.
2. Яцишин А. Інформатизація освіти України та підготовка наукових кадрів вищої кваліфікації до її здійснення //Педагогічні інновації: ідеї, реалії, перспективи. – 2016. – №. 1. – С. 54-62.
3. Гончаров Д. О. Організація студентського самоврядування в академічній групі (з досвіду роботи) – 2011. – №. 28. – С. 39-46.
4. Третяк О. С., Чебоненко С. О. Академічна мобільність та сучасні підходи до забезпечення якості вищої освіти в Україні. – 2015.
5. Кузьома Т. Б. Участь студентів ВНЗ України у науково-дослідній діяльності //Наука і освіта. – 2012. – №. 8. – С. 83-85.
6. Чжан Ювень, Ткачова Н. О. Нормативно-правова база для формування полікультурної компетентності студентів в університетах КНР. Науковий вісник Ужгородського університету. Серія: Педагогіка. Соціальна робота. Вип. 2 (45). Ужгород, 2019. С. 196-199.
7. Корнійчук В. Процес впровадження інформаційно – комунікаційних технологій у навчальний процес: проблеми та перспективи. 2020. URL: <https://doi.org/10.36074/20.11.2020.v3.21> (дата звернення: 30.10.2023).
8. Ткаченко Л. В., Хмельницька О. С. Особливості впровадження дистанційного навчання в освітній процес закладу вищої освіти. 2021. Т. 3, № 75. С. 91–96. URL: <https://doi.org/10.32840/1992-5786.2021.75-3.18> (дата звернення: 30.10.2023).
9. Бабакова Л. М. Шляхи вдосконалення зворотного зв'язку з метою покращення процесу навчання та ефективності контролю – 2022. – Т. 27. – С. 251.
10. Ляшенко І. В. Переваги дистанційного навчання: thesis. 2015. URL: <http://essuir.sumdu.edu.ua/handle/123456789/72530> (дата звернення: 30.10.2023).

11. Зозуляк-Случик Р. Модель формування професійної етики майбутніх соціальних працівників в університетах. Молодь і ринок. 2019. № 6 (173), черв. С. 123–127.

12. Башовий в. М., стаценко в. В., Стаценко Д. В. Визначення швидкості роботи сучасних фреймворків для створення web-інтерфейсів. Technologies and Engineering. 2023. № 4. С. 9–16. URL: <https://doi.org/10.30857/2786-5371.2022.4.1> (дата звернення: 30.10.2023).

13. Шевченко М. М. Інтернет-просування товарів як інструмент маркетингу: thesis. 2018. URL: <http://repository.kpi.kharkov.ua/handle/KhPI-Press/45436> (дата звернення: 30.10.2023).

14. Ешану А. А., Казіяхмедов Т. Б. Відмінності вебдодатка від вебсайту. III Міжнародна науково-практична конференція. 2021. URL: <https://doi.org/10.36906/ap-2020/07> (дата звернення: 30.10.2023).

15. Volodymyr A. Клієнт-серверна модель інформаційної системи масового оповіщення населення. Technical sciences and technologies. 2018. № 4 (14). С. 136–144. URL: [https://doi.org/10.25140/2411-5363-2018-4\(14\)-136-144](https://doi.org/10.25140/2411-5363-2018-4(14)-136-144) (дата звернення: 30.10.2023).

16. Мікула М., Коцюк Ю., Мікула О. Організація баз даних та знань. Вид-во Нац. ун-ту «Острозька акад.», 2021. URL: <https://doi.org/10.25264/978-617-8041-08-3> (дата звернення: 30.10.2023).

17. Куликовська Н. А., Тіменко А. В., Костецький Д. В. Автоматизована система керування температурою приміщень. Вчені записки Таврійського національного університету ім. В.І. Вернадського. Серія: Технічні науки. 2023. Т. 1, № 3. С. 163-167. URL: <https://doi.org/10.32782/2663-5941/2023.3.1/26> (дата звернення: 22.11.2023).

18. Костецький Д. В., Тягунова М. Ю. Розробка системи дистанційного навчання. Національний університет «запорізька політехніка». URL: https://zp.edu.ua/uploads/dept_s&r/2023/conf/4.1/TN-2023-FKNT.pdf – С. 37-38. (дата звернення: 22.11.2023).

19. Academy C. P. Php: The Ultimate Crash Course to Learn PHP with Practical Computer Coding Exercises. Independently Published, 2019. 227 с.
20. Subecz Z. Web-development with Laravel framework. Gradus. 2021. Т. 8, № 1. С. 211–218. URL: <https://doi.org/10.47833/2021.1.csc.006> (дата звернення: 22.11.2023).
21. Sawłuk K., Miłosz M. Comparison of object-relational data mapping technology in Symfony 3 framework. Journal of Computer Sciences Institute. 2018. Т. 8. С. 235–240. URL: <https://doi.org/10.35784/jcsi.687> (дата звернення: 22.11.2023).
22. Purohit R., Jain S., Gupta S. Role of Angular Framework in Web Development. International Journal of Advanced Research in Science, Communication and Technology. 2023. С. 114–127. URL: <https://doi.org/10.48175/ijarsct-10731> (дата звернення: 22.11.2023).
23. Zandstra M. Objects. PHP Obejcts, Patterns, and Practice. Berkeley, CA, 2013. С. 7–10. URL: https://doi.org/10.1007/978-1-4302-6032-5_2 (дата звернення: 30.10.2023).
24. Sinha S. Introduction to Laravel. Beginning Laravel. Berkeley, CA, 2019. С. 1–10. URL: https://doi.org/10.1007/978-1-4842-4991-8_1 (дата звернення: 30.10.2023).
25. Model-View-Controller. Flexible, Reliable Software. 2010. С. 367–372. URL: <https://doi.org/10.1201/9781439882726-37> (дата звернення: 30.10.2023).
26. Uzayr S. b. HTML5. Frontend Development. Boca Raton, 2022. С. 37–76. URL: <https://doi.org/10.1201/9781003309062-2> (дата звернення: 30.10.2023).
27. Wirfs-Brock A., Eich B. JavaScript: the first 20 years. Proceedings of the ACM on Programming Languages. 2020. Т. 4, HOPL. С. 1–189. URL: <https://doi.org/10.1145/3386327> (дата звернення: 30.10.2023).
28. Сугаїпов С. А., Геріханов З. А. Оптимізація веброзробки за допомогою React, Angular і Vue. 2023. Т. 98, № 10. С. 141-143. URL: <https://doi.org/10.18411/trnio-06-2023-565> (дата звернення: 30.10.2023).

29. Tantsiura L. D., Kyselova I. V., Biliaiev A. V. Аудит оцінки лікарями дистанційної форми навчання на етапі післядипломної освіти. Медична освіта. 2021. № 4. С. 100–104. URL: <https://doi.org/10.11603/me.2414-5998.2020.4.11369> (дата звернення: 17.11.2023).

30. Аналітична довідка щодо тенденцій організації дистанційного навчання у закладах фахової передвищої та вищої освіти в умовах карантину у 2020/2021 навчальному році. URL: https://sqe.gov.ua/wp-content/uploads/2021/05/ANALITICHNA_DOVIDKA_Opitivannya_FPO_ZVO_DSYAO_05.2021.pdf (дата звернення: 17.11.2023)

31. Аналітична довідка щодо тенденцій організації дистанційного навчання у закладах вищої освіти у 2021/2022 навчальному році в умовах воєнного стану. URL: https://sqe.gov.ua/wp-content/uploads/2023/01/ANALITICHNA-DOVIDKA-ZVO_12.06.pdf (дата звернення: 17.11.2023).

32. Інформаційно-аналітична довідка щодо організації освітнього процесу в закладах фахової та вищої освіти України в умовах воєнного стану. URL: https://sqe.gov.ua/wp-content/uploads/2023/08/IAD_II_kvartal_FPO_ZVO_SQE-2023.pdf (дата звернення: 17.11.2023)

33. Yu C.-J., Jeong S.-Y. Use Case Diagram Extraction Technique from Requirements Specification. The KIPS Transactions:PartD. 2002. Т. 9D, № 4. С. 639–650. URL: <https://doi.org/10.3745/kipstd.2002.9d.4.639> (дата звернення: 22.11.2023).

34. Махум Z. Діаграма послідовності (Sequence Diagrams). Махум Zosym. URL: <https://www.maxzosim.com/sequence-diagrams/> (дата звернення: 22.11.2023).

35. Lamouchi N. Introduction to the Monolithic Architecture. Pro Java Microservices with Quarkus and Kubernetes. Berkeley, CA, 2021. С. 35–38. URL: https://doi.org/10.1007/978-1-4842-7170-4_2 (дата звернення: 22.11.2023).

36. Булах Б., Харченко К. МІКРОСЕРВІСНА КОНТЕЙНЕРНА АРХІТЕКТУРА СИСТЕМИ КЕРУВАННЯ ПОТОКАМИ ДАНИХ.

Інфокомунікаційні та комп'ютерні технології. 2022. Т. 2, № 02. URL: <https://doi.org/10.36994/2788-5518-2021-02-02-17> (дата звернення: 22.11.2023).

37. Sun Y. Single-Page Applications. Practical Application Development with AppRun. Berkeley, CA, 2019. С. 141–162. URL: https://doi.org/10.1007/978-1-4842-4069-4_7 (дата звернення: 22.11.2023).

38. Ivanechko N., Okrepkyi R., Pavelko V. SEO OPTIMIZATION: THE SEMANTIC CORE. PROBLEMS OF SYSTEMIC APPROACH IN THE ECONOMY. 2022. № 1(87). URL: <https://doi.org/10.32782/2520-2200/2022-1-16> (дата звернення: 22.11.2023).