Хмельницький національний університет
Факультет інформаційних технологій
Кафедра інженерії програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА

Гурного Романа Вікторовича
Прізвище, ім'я, по-батькові студента(ки)

на здобуття ступеня вищої освіти Бакалавра

Вебзастосунок для менеджменту навчання та автоматизованого контролю знань студентів
Назва теми

Рівень вищої освіти Перший (бакалаврський)

Галузь знань 12 «Інформаційні технології»

Спеціальність 121 «Інженерія програмного забезпечення»

Освітня програма Освітньо-професійна програма «Інженерія програмного забезпечення»

КвРІПЗ.2201123.01.02.ПЗ

Виконав студент ІІІ курсу група ІПЗс-22-1 _____ Роман ГУРНИЙ
Підпис Ініціали, прізвище

Керівник канд. пед. наук, доцент _____ Оксана ОНИШКО
Науковий ступінь, звання Підпис Ініціали, прізвище

Нормоконтролер канд. тех. наук _____ Оксана ЯШИНА
Науковий ступінь, звання Підпис Ініціали, прізвище

До захисту допускаю:
Завідувач кафедри інженерії програмного забезпечення _____ Леонід БЕДРАТЮК
Підпис Ініціали, прізвище

10 06 2025 р.

Хмельницький 2025

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет Інформаційних технологій
Кафедра Інженерії програмного забезпечення
Рівень вищої освіти Перший (бакалаврський)
Галузь знань 12 «Інформаційні технології»
Спеціальність 121 «Інженерія програмного забезпечення»
Освітня програма Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри ІПЗ
Л. П. Бедратюк
02 01 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Гурному Роману Вікторовичу
Прізвище, ім'я, по батькові студента

1. Тема роботи Вебзастосунок для менеджменту навчання та автоматизованого контролю знань студентів

Керівник роботи Онишко О.Г. канд. пед. наук, доцент
Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджена наказом ректора університету від 07.02.2025 р. №23
2. Строк подання студентом роботи на кафедру 01.06.2025 р.
3. Вихідні дані до роботи Матеріали переддипломної практики

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити) Вступ, дослідження предметної області та постановка задачі, проектування програмного забезпечення, програмна реалізація та тестування застосунку, висновки

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень) Презентаційні матеріали (слайди, 15 шт.)

6. Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Яшина О.М., канд. тех. наук		
Антиплагіат	Форкун Ю.В., доцент		

7. Дата видачі завдання « 02 » січня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів (розділів) кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Ознайомлення з тематикою дипломного проектування, визначення та узгодження індивідуальних тем кваліфікаційних робіт (КвР)	01.12 – 31.12.2024	
2	Збір матеріалу за темою КвР; дослідження предметної області, в якій планується використання програмного забезпечення (ПЗ), визначення задач та вимог, розробка технічного завдання	01.01 – 20.02.2025	
3	Проектування програмного забезпечення	21.02 – 20.03.2025	
4	Програмна реалізація з використанням відповідних засобів розробки. Тестування ПЗ	21.03 – 30.04.2025	
5	Написання вступу, загальних висновків, оформлення переліку джерел посилання та додатків. Оформлення пояснювальної записки КвР згідно вимог	01.05 – 25.05.2025	
6	Попередній захист КвР	Травень 2025	Згідно графіка
7	Перевірка КвР на плагіат, нормоконтроль, отримання відгуків, рецензій та інших супровідних документів. Брошурування (зшиття) пояснювальної записки	26.05 – 30.05.2025	
8	Здача КвР на кафедру; підготовка КвР для розміщення у репозитарії ХНУ; підготовка до захисту та захист КвР	з 01.06.2025	

Студент _____ Роман ГУРНИЙ
Підпис Ім'я, ПРІЗВИЩЕ

Керівник роботи _____ Оксана ОНИШКО
Підпис Ім'я, ПРІЗВИЩЕ

АНОТАЦІЯ

Тема кваліфікаційної роботи «Вебзастосунок для менеджменту навчання та автоматизованого контролю знань студентів».

Автор роботи: Гурний Роман Вікторович.

Керівник роботи: Онишко Оксана Григорівна.

Пояснювальна записка: 93 с., 49 рис., 3 дод., 40 джерел.

Графічна частина: 15 презентаційних слайдів.

ВЕБЗАСТОСУНОК, СУБД MySQL, UML, COMPOSER, CSS, DOCKER, HEROKU, HTML, JAVASCRIPT, LARAVEL, MVC, PHP, SQL, VUE JS 3.

Мета кваліфікаційної роботи: створення веб-застосунку для студентів та викладачів щоб полегшити і систематизувати процес менеджменту та контролю успішності студентів.

У кваліфікаційній роботі проведено аналіз предметної області та її інформаційного забезпечення, визначені функціональні та нефункціональні вимоги до програмної системи, розроблена загальна архітектура застосунку, спроєктована база даних та структура застосунку.

Для реалізації програмного продукту було використано різні програмні інструменти і технології, включаючи мови PHP та JavaScript у середовищі PhpStorm та WebStorm. Клієнтська частина побудована з використанням VUE JS 3 та VUEX. Для контейнеризації застосовано Docker, як СУБД використано MySQL. Збірка проекту здійснювалася через Webpack.

Backend частину розроблено на основі фреймворку Laravel, що передбачає легку масштабованість і підтримку в подальших ітераціях розвитку проекту.

Практичне значення результатів роботи полягає в тому, що впровадження розробленого застосунку дозволяє автоматизувати дистанційне навчання шляхом впровадження груп, додавання теорії, створення завдань та отримання звітів.

10.06.2025
Дата

Підпис

ВІДОМІСТЬ ДОКУМЕНТІВ

№ рядка	Формат	Позначення документа	Найменування документа	К-сть аркушів	№ екз.	Примітка
			Текстові документи			
1	А4	КвРІПЗ.2201123.01.02.ПЗ	Пояснювальна записка	93		
2	А4		Завдання на кваліфіка-	1		
			ційну роботу			
3	А4		Анотація	1		
			Графічні документи			
4	А4		Презентаційні матеріали	15		
5	А3	КвРІПЗ.2201123.01.02.Е1	Діаграма використання	1		
6	А3	КвРІПЗ.2201123.01.02.Е2	Діаграма станів	1		
7	А3	КвРІПЗ.2201123.01.02.Е3	Діаграма взаємодії	1		
8	А3	КвРІПЗ.2201123.01.02.Е4	Діаграма класів	1		
9	А3	КвРІПЗ.2201123.01.02.Е5	Діаграма розгортання	1		
10	А3	КвРІПЗ.2201123.01.02.Е6	Діаграма схеми даних	1		

Змн.	Арк.	№ докум.	Підпис	Дат	КвРІПЗ.2201123.01.02.ВД		
Виконав		Гурний Р.В.		10.06	Вебзастосунок для менеджменту навчання та автоматизованого контролю знань студентів	Літ.	Арк. 1 / Аркушів 1
Керівник		Онишко О.Г.		10.06			
Н. контр.		Яшина О.М.		10.06	Відомість документів	ХНУ, ІПЗс-22-1	
Зав. каф.		Бедратюк Л.П.		10.06			

ЗМІСТ

ВСТУП

На сьогоднішній день вебсайти все більше доводять, що вони є потужним кросплатформеним засобом, які дозволяють користувачу за допомогою браузера працювати з більшістю популярних програм онлайн, без необхідності завантаження та інсталяції програмного засобу. Більше того, це можна зробити як на стаціонарному комп'ютері, так і на мобільному смартфоні.

Все більше розробників намагаються зробити своє програмне забезпечення доступним онлайн, адже це зменшує витрати на розробку на окремі платформи та дозволяє залучити більше користувачів. Сучасна розробка вебсайту вже не являє собою стилізовані статті з картинками на якісь теми, вона максимально наближена до створення професійних онлайн інструментів.

На розвиток вебсайт індустрії велику роль відіграли нові технології та засоби розробки. До таких засобів можна віднести популярні фреймворки [1] з підтримкою реактивності, мікросервісну архітектуру та безголову розробку. Всі вони дозволяють за короткий термін створити потужний програмний засіб, що буде доступний онлайн будь-де.

Багато предметних галузей вже є автоматизованими, в більшості з них є велика кількість аналогів, тому перед визначенням теми дипломного проекту була проведена велика робота з пошуку предметної галузі, що потребує іншого аналога, або іншого бачення на реалізацію вже готових програмних продуктів.

Після пошуку предметної області було обрано автоматизацію перевірки навчальних здобутків учнів та студентів, адже зараз онлайн навчання посідає не менш важливу роль ніж очне. Оскільки попит на дистанційне навчання став великий, почались використовуватись монопольні програми-гіганти як от Microsoft Teams чи Google Classroom. Це потужні програмні засоби, але їх основна ціль не є дистанційне навчання, до того ж там досить складний і перенасичений функціонал, що може перешкоджати комфортному використанню цих продуктів для дистанційного навчання в українських школах. Хоч дані

програми і справляються з вимогами, було прийнято рішення створити український аналог з своїм баченням реалізації автоматизованої перевірки знань.

Щоб більшість навчальних закладів могли використовувати майбутній український аналог, його розповсюдження буде абсолютно безкоштовне і вихідний код буде опублікований в відкритому доступі з ліцензією GPL, що дозволить українській спільності впроваджувати новий функціонал та виправляти наявні проблеми.

Об’єктом дослідження є сервіси, які надають змогу автоматизованої перевірки знань та можливість для онлайн навчання.

Предметом дослідження даної кваліфікаційної роботи є програмне забезпечення, яке забезпечує автоматизований процес перевірки знань та онлайн навчання.

Головною метою даної кваліфікаційної роботи є створення вебзастосунку Edelways – платформи для дистанційного навчання, створеної спеціально для українських навчальних закладів, та підтримуваної своєю спільнотою. Щоб цю платформу могла вдосконалювати спільнота, на розробника лягає велика відповідальність в розробці гнучкої архітектури та легкості додавання нового функціоналу. Платформа названа на честь благородної квітки едельвейс, що росте на вершині холодних гір, і щоб отримати її, треба пройти через багато випробовувань. Так само в здобуванні освіти: необхідно пройти доволі тернистий шлях щоб здобути професійний рівень в певній галузі.

Отже, сучасні засоби веброзробки дозволяють створювати професійні програмі засоби, що будуть доступні за допомогою браузера [2]. З допомогою цих засобів буде розроблена українська платформа для онлайн навчання Edelways, що матиме своє бачення на автоматизацію цієї предметної області.

1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Змістовний аналіз предметної області, її структурних та функціональних особливостей

Предметною областю дипломного проекту є автоматизація перевірки знань учнів та студентів. Дана область охоплює широкий спектр діяльності, включаючи створення завдань різних типів з ручною або автоматичною перевіркою, ведення обліку результатів та генерацію систематизованих звітів. Тобто предметна область являє собою повноцінний віртуальний простір, що дозволяє повністю реалізувати процес дистанційного навчання.

Онлайн навчання зазвичай реалізовується за допомогою створення віртуальних завдань, поділ учасників на групі, можливості проведення групових дзвінків та ведення статистики. Основний акцент робиться на підтримку освітнього процесу у віртуальному середовищі, де викладач має змогу створювати навчальні групи, додавати до них учасників, готувати завдання з різними критеріями перевірки, надавати доступ до теоретичних матеріалів, здійснювати оцінювання поданих робіт, залишати коментарі та, за необхідності, дозволяти повторну перевірку. Водночас студент або учень може приєднатися до певної навчальної групи за спеціальним кодом, ознайомитися з теорією, переглядати список активних завдань, бачити терміни їх виконання, надсилати власні роботи у вигляді файлів та отримувати зворотний зв’язок у вигляді оцінок і коментарів.

В даній області можна виділити наступні сутності: користувач, адміністратор, завдання, відгук, теорія. Для цих сутностей буде реалізовано ряд функцій, які забезпечать виконання основної мети проекту.

Функціонально предметна область охоплює низку ключових елементів, таких як користувачі (студенти), адміністратори (викладачі), навчальні групи, завдання, теоретичні матеріали, а також оцінювання й коментарі. Ці елементи

тісно взаємодіють між собою, утворюючи структуровану модель онлайн освіти. Для зручності та ефективності роботи кожен користувач має персональний обліковий запис. Викладач може створити власну навчальну групу, куди надсилає запрошення студентам, додає навчальні матеріали, формує індивідуальні або групові завдання, встановлює дедлайни, максимальну кількість балів і додає додаткові пояснення у вигляді файлів.

Участь студента в навчальному процесі починається з приєднання до групи за унікальним кодом, після чого відкривається доступ до теоретичних матеріалів та актуальних завдань. Після виконання завдання студент завантажує файл із роботою, який потім оцінюється викладачем. Результати перевірки відображаються у вигляді балів та коментарів. У випадку незадовільної оцінки або потреби в доопрацюванні студент має можливість повторно надіслати виправлену роботу на перевірку..

В групі можна переглядати активні завдання, їх кінцевий термін та максимальний бал. Також присутня окрема вкладка, де міститься корисна теорія, яка допоможе із реалізацією завдань.

Виконати завдання означатиме завантажити файл відповідного формату та надіслати на перевірку. Після перевірки адміністратором групи користувач буде бачити свою оцінку та коментар. При необхідності роботу можна завантажити ще раз та запросити повторну перевірку.

Створення проекту має здійснюватися, використовуючи необхідні мови програмування та технології, щоб вихідний програмний продукт можна було легко підтримувати та додати новий функціонал. Щоб досягти такого результату, необхідно багато зусиль витратити на проектування.

Програмний продукт буде розроблено за «безголовим» принципом [3]. Тобто сервер та інтерфейсна частина являтимуть собою два окремих проекти, що дозволить поставити явне розділення між цими частинами та поліпшить підтримку програми. Серверна та інтерфейсна частини в свою чергу теж будуть спроектовані на основі модульної архітектури. Передача даних між серверною та

інтерфейсною частиною буде відбуватись за допомогою АПІ.

Платформа для дистанційного навчання Edelways має містити вікна:

– реєстрації;

– авторизації;

– перегляду основних та поточних груп;

– приєднання до групи;

– створення та перегляд теорії;

– створення та перегляд завдань;

– запрошення та перегляд користувачів;

– оцінки виконаного завдання.

Для зручності використання додатку на різних пристроях, необхідно передбачити адаптивну верстку інтерфейсної частини та забезпечити комфортний та мінімалістичний дизайн. Адаптивне верстання [4] являє собою адаптацію вмісту сторінки в залежності від розміру екрану.

Аналіз предметної області, сфери застосування та визначення стеку технологій дозволяють остаточно визначитись із технічним завданням. Для цього необхідно провести аналітичну роботу, визначивши основні функціональні можливості та використати порівняльну характеристику з схожими застосунками.

1.2 Аналіз наявного програмно-технічного забезпечення предметної області

До найбільш популярних наявних програмних продуктів належать Microsoft Teams [5] та Google Classroom [6]. Це дві програми-монополісти, які захопили майже весь ринок в даній предметній області. В них є багато переваг. Основною є те, що вони безкоштовні. Але основним недоліком є те, що вони не створювались виключно для дистанційного навчання, тому там не вистачає якогось функціоналу і навпаки багато можливостей є зайвими.

Google Classrom – програмний продукт, зображений на рисунку 1.1, розроблений компанією Google для створення та перевірки виконання завдань.

Перевагами є те, що цей сервіс має тісну інтеграцію з іншими сервісами цієї ж компанії, але основним недоліком є відсутність почуття цілісності програми, не вистачає ряду функцій і навпаки є багато зайвих можливостей. Також даний продукт є безкоштовним, що робить його популярним рішенням для дистанційного навчання. Недоліком є обмежений набір функцій, що не дозволяє мати повноцінний електронний журнал або вести статистику відвідувань учнів. Також відсутня історія коментарів, якщо робота змінювалась і перевірялась декілька разів. Наступним недоліком є те, що немає логічного поділу між завданнями та теорією. Це призводить до плутанини під час використання.

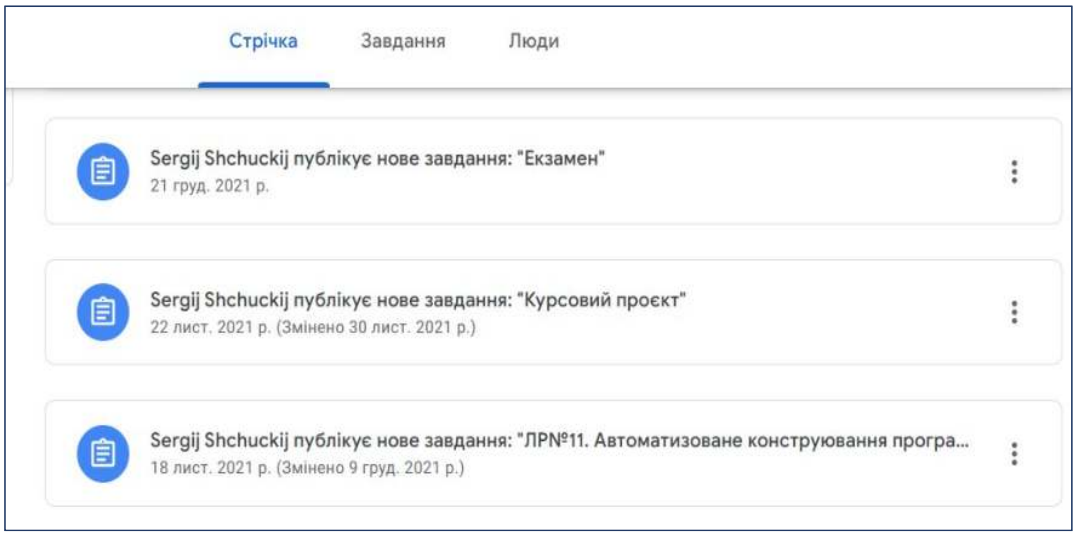


Рисунок 1.1 – Google Classroom

Натомість проект «Edelways» дозволить додавати теоретичний матеріал в групу, щоб здобувачі освіти могли просто та доступно підготуватись до виконання завдання. Також відкритість програмного коду проекту та АПІ дозволить перейняти переваги наявних рішень та впровадити їх згідно з потребами українських закладів освіти. В цьому стануть в пригоді різні модулі, які будуть додавати новий функціонал і підтримуватись спільнотами. Такі модулі можна буде встановлювати за потреби.

Таким чином, «Edelways» не має такого об’ємного функціоналу, але своєю відкритістю сприяє простій інтеграції як відео конференцій, так і автоматичного

тестування чи створення платформи для олімпіадного програмування з автоматичною перевіркою результату виконання програми.

Microsoft Teams – кросплатформений програмний продукт, зображений на рисунку 1.2, розроблений для взаємодії людей в командах та менеджменту роботи команди. Має дуже велику кількість функціональних можливостей та інтеграцію з іншими сервісами одної екосистеми. Найчастіше використовується в Україні для реалізації дистанційного навчання. Даний програмний продукт має місцями складний інтерфейс та закритий програмний код, що унеможливлює додавання якихось нових функцій незалежними спільнотами.

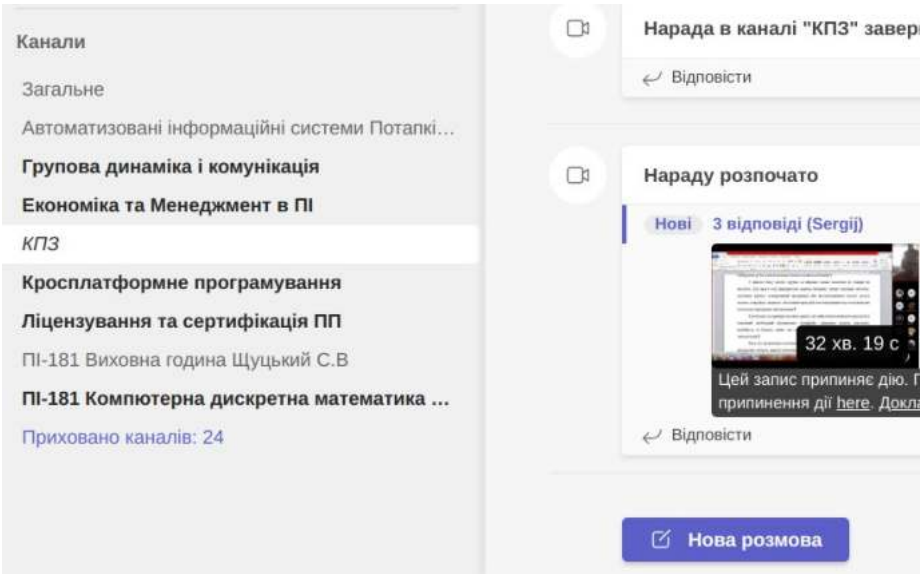


Рисунок 1.2 – Microsoft Teams

Перевагою даного продукту є доступність з різних пристроїв. Існує веб версія, мобільний застосунок для програма для комп’ютерів з різними операційними системами. Наступною перевагою є наявність офісного пакету, який інтегрований в даний продукт. Це дозволяє переглядати, створювати та редагувати файли прямо в застосунку. Також слід зазначити наявність можливості групових дзвінків із підтримкою великої кількості відвідувачів.

Основним недоліком є те, що даний програмний продукт є платним для комерційного використання, тому це зменшує попит серед державних установ.

Також є визначений список країн, які можуть використовувати дане ПЗ.

Отже, було проведено аналітичний огляд популярних наявних засобів для розробки та розроблених програмних продуктів для обраної предметної області. Зваживши всі переваги та недоліки було обрано Laravel для серверної розробки та VueJS для інтерфейсної. Також були проаналізовані функціональні можливості наявних програмних продуктів, виділені їх основні переваги та недоліки.

1.3 Визначення вимог до програмного забезпечення та технічне завдання

Сервіс для дистанційного навчання має бути зручний, із зрозумілим та простим інтерфейсом та швидкодією. Крім того, він має правильно виконувати запрограмовані можливості та коректно реагувати на будь-які непередбачувані дії користувача.

Даний продукт має реалізувати мережеву клієнт-серверну архітектуру. Кожен клієнт буде підключить до одного й того ж серверу та взаємодіяти з єдиною базою даних. Тобто він складатиметься з двох окремих частин – серверної та клієнтської. Оскільки між цими частинами є чітка межа то варто розписати вимоги для кожної з них окремо.

Вимоги до серверної частини:

– реалізувати можливість реєстрації (користувач вводить електронну пошту, пароль та ім’я. У випадку коректних даних створюється запис в базі даних. Критерієм успішності є помилка при повторному використанні тієї ж електронної адреси);

– додати можливість авторизації (користувач вводить електрону пошту та пароль. Після успішної перевірки надсилається токен доступу. Критерієм успішності при невірному паролі є повідомлення про помилку входу);

– реалізувати створення навчальних груп (користувач із правами викладача створює групу з унікальною назвою. Критерієм успішності є наявність нової групи у списку після створення);

– реалізувати можливість додавання та видалення користувачів з груп (адміністратор групи має змогу надсилати запрошення за email, видаляти учасників. Критерієм успішності є зміна списку учасників відповідно до дій адміністратора);

– забезпечити можливість створення завдань (викладач може створити завдання з текстом, дедлайном та файлами. Критерієм успішності є наявність завдання у групі після створення);

– реалізувати можливість завантаження файлів (користувачі мають змогу прикріплювати файли до завдань. Критерієм успішності є збереження файлів і можливість їх подальшого перегляду або завантаження);

– реалізувати можливість додавання теоретичних матеріалів (викладач створює теоретичний блок, який стає доступним членам групи. Критерієм успішності є відображення теорії для учасників відповідної групи);

– реалізувати можливість отримання списків теорій, завдань і груп (користувач може переглянути лише ті дані, до яких має доступ. Критерієм успішності є відсутність доступу до чужої інформації);

– реалізувати відкриту документацію API (усі доступні запити мають бути задокументовані у вигляді графічного інтерфейсу, наприклад Swagger. Критерієм успішності є доступність опису всіх ендпоінтів).

Вимоги до інтерфейсної частини:

– реалізувати інтерфейс реєстрації (користувач заповнює форму з перевіркою правильності введених даних. Критерієм успішності є перехід до входу після успішної реєстрації);

– реалізувати інтерфейс авторизації (користувач входить в систему за допомогою електронної пошти і пароля. Критерієм успішності є відображення основного екрану після успішного входу);

– реалізувати інтерфейс перегляду та створення груп (користувач бачить список доступних груп та може створити нову. Критерієм успішності є поява нової групи у списку після її створення);

– реалізувати перегляд конкретної групи (користувач відкриває групу та переглядає її вміст: учасників, завдання, теорію. Критерієм успішності є коректне відображення даних згідно з правами доступу);

– реалізувати адміністрування групи для власника (власник може редагувати завдання, додавати або видаляти учасників, публікувати теорію. Критерієм успішності є зміна вмісту групи відповідно до дій адміністратора);

– реалізувати приєднання до групи (користувач приєднується до групи за кодом або посиланням. Критерієм успішності є відображення нової групи у списку після приєднання);

– реалізувати взаємодію з сервером через API (усі дії користувача ініціюють відповідні запити до серверу. Критерієм успішності є правильна реакція інтерфейсу на відповіді сервера).

Для зв’язку між інтерфейсною та серверною частинами використаємо мережу Інтернет та безкоштовний хостинг Heroku [7]. Щоб реалізувати поставлені вимоги необхідно приділити багато уваги проектуванню програмного засобу.

Для успішної реалізації продукту слід визначити його основні етапи проєктування. На етапі проектування відбувається надзвичайно важливий процес визначення архітектури системи, від цього кроку залежить подальший процес розробки усієї системи. Якщо цих систем декілька то важливо розробити правильну архітектуру для кожної з них. Етап проектування структури бази даних можна поділити на наступні кроки:

– аналіз вхідних даних;

– аналіз вихідних даних

– визначення основних сутностей;

– визначення основних характеристик сутностей;

– визначення зв’язків між сутностями;

– визначення типів зв’язків;

– нормалізація структури даних.

Проектування та розробка програмної частини:

– визначити архітектуру системи з чітким поділом клієнтської та серверної частин (критерієм успішності є можливість розробляти та тестувати кожну частину окремо);

– спроєктувати базу даних (виконати аналіз сутностей, зв’язків та провести нормалізацію. Критерієм успішності є коректна структура для збереження всіх необхідних даних);

– визначити стек технологій та інструментів (вибрати мову програмування, фреймворки, бібліотеки. Критерієм успішності є відповідність вибраного стеку поставленим вимогам);

– створити прототип інтерфейсу [8] (макети з основними сценаріями взаємодії. Критерієм успішності є відповідність прототипу майбутньому дизайну);

– реалізувати архітектуру клієнтської та серверної частин (створити відповідні модулі та сервіси. Критерієм успішності є можливість масштабування та тестування окремих частин);

– розгортання програмного продукту на сервері (розгорнути серверну та клієнтську часини на сервері, встановити зв’язок між ними. Критерієм успішності є глобальний доступ до програми).

Після проектування та конструювання слід переходити до етапу тестування.

Тестування надзвичайно важливий етап, який проводиться протягом усього життєвого циклу програмного забезпечення. Рівень тестування вказує на те, над чим саме здійснюється тестування: чи то уся система загалом, чи група модулів, а може й окремий модуль. Тестування – можливість уберегти кінцевого клієнта від негативного досвіду.

Виділимо наступні вимоги до тестування:

– реалізувати модульне тестування (перевіряти окремі функції та класи на правильність виконання. Критерієм успішності є правильні результати при різних вхідних даних, також програма не має вибивати чи входити в некоректний стан);

– реалізувати інтеграційне тестування (перевіряти взаємодію кількох модулів між собою. Критерієм успішності є коректна передача даних між компонентами);

– реалізувати системне тестування (перевірити систему загалом, включаючи UI, безпеку та стабільність. Критерієм успішності є повна відповідність системи вимогам);

– реалізувати тестування безпеки (запобігати доступу до чужих даних, навіть при спробі ручного запиту. Критерієм успішності є повернення помилок доступу при несанкціонованих діях);

– організувати тестування кожної частини окремо (модульність дозволяє ізолювати помилки. Критерієм успішності є мінімізація дефектів на етапі інтеграції);

– провести тести для усіх можливих сценаріїв використання (включаючи граничні випадки. Критерієм успішності є стабільна робота системи у всіх умовах).

Даний проект включатиме наступні рівні тестування:

– модульне тестування [9] - тестування, яке перевіряє працездатність окремої функції, класу, загалом якоїсь сутності. Даний підхід полягає в тому, щоб викликати саму функцію і перевірити її працездатність на різних вхідних даних. Усі знайдені помилки виправляються без формального опису;

– інтеграційне тестування [10] - тестування, яке призначене для перевірки взаємодії різних компонентів між собою в межах однієї або різних систем;

– системне тестування [11] – тестування, яке полягає в тому, щоб перевірити систему у її цілісності, як згідно формальних вимог, так і не формальних, які не були вказані. В цілому, це тест на працездатність усією системи разом.

Загалом, знайдені проблеми, дефекти слід проаналізувати, вирішити й задокументувати.

Тести слід проводити для найрізноманітніших станів й випадків програмної системи.

Програмний продукт складається з двох окремих проектів: інтерфейсного та серверного, тому тестування кожного з них окремо дозволить зменшити кількість помилок перед системним тестуванням.

Якщо при проектуванні буде обрана модульна архітектура, то це дозволить тестувати кожен маленький модуль окремо що дозволить ще краще локалізувати усі логічні помилки під час модульного тестування.

В цілому можна переконатись, що складність тестування напряму залежить від архітектури проекту, адже якщо створювати монолітний проект, який б вміщував в собі інтерфейсну та серверну частини без поділу на модульність, то тестування та відлагодження такого проекту могло б зайняти більше часу ніж створення, що було б як економічно та і професійно погано.

Також варто пам’ятати, що підхід з розподілом інтерфейсу та серверу на окремі проекти вимагає додаткового тестування в безпековому плані, адже сторонній користувач не повинен отримати доступ до даних іншого користувача, виконавши аналогічний запит.

Тестування є важливим етапом в життєвому циклі створення програмного продукту. Саме завдяки йому можна повноцінно відлагодити проект та знайти основні неявні помилки. Модульне тестування слід робити після створення кожного нового модуля, інтеграційне після написання декількох логічно пов’язаних модулів і системне при готовності всіх компонентів програмного продукту.

У підрозділі наведено повний перелік функціональних та нефункціональних вимог до програмного забезпечення, які визначають основні принципи його побудови, логіку взаємодії між користувачем і системою, а також критерії успішності кожної операції. Чітке формулювання вимог забезпечує можливість системної реалізації, ефективного тестування та подальшого масштабування продукту. Виконання цих вимог є необхідною умовою для створення надійного, безпечного та зручного у користуванні застосунку у сфері автоматизованого контролю знань.

Визначивши основні вимоги можна перейти до формування технічного завдання. Найменування програмного продукту – вебзастосунок «Edelways». Галузями застосування є автоматизація навчального процесу для будь-якого навчального закладу: шкіл, коледжів, університетів, курсів підвищення кваліфікації тощо. До автоматизації входять створення завдань, автоматична перевірка їх виконання, збереження теоретичних матеріалів, комунікація між учасниками освітнього процесу, а також централізоване зберігання всієї навчальної інформації в одному місці.

Основними задачами, які вирішує вебзастосунок Edelways є:

– створення та збереження теоретичних матеріалів (лекції, конспекти, відео, презентації);

– створення та перевірка практичних завдань, тестів, індивідуальних і групових проєктів;

– організація навчальних груп і курсів із різними правами доступу для учнів і викладачів;

– підтримка комунікації між учасниками освітнього процесу за допомогою повідомлень та коментарів;

– надання зворотного зв’язку від викладача до студента (в тому числі за допомогою коментарів до виконаних завдань);

– централізоване зберігання навчальної інформації у захищеній базі даних;

– забезпечення доступу до даних з будь-якого пристрою, підключеного до Інтернету.

Цільовою аудиторією даного проекту є:

– учні та студенти, які використовують систему для доступу до матеріалів, виконання завдань, отримання оцінок і коментарів;

– викладачі, які створюють навчальний контент, перевіряють роботи, комунікують зі студентами та контролюють їхній прогрес;

– репетитори, які можуть використовувати платформу для індивідуальних занять;

– адміністрація навчальних закладів, яка може моніторити статистику успішності, активність студентів і викладачів.

Завдяки простому та інтуїтивно зрозумілому інтерфейсу користувачі не будуть відчувати дискомфорту при користуванні додатком, навіть за відсутності високого рівня цифрової грамотності. Застосування сучасних веб-технологій дозволяє адаптувати інтерфейс під різні пристрої - комп’ютери, планшети, смартфони.

Даний проект заслуговує на увагу, адже дозволяє значно зекономити час викладачів і студентів, автоматизуючи рутинні процеси перевірки знань та надання зворотного зв’язку. Завдяки можливості централізовано зберігати теорію, завдання, оцінки та коментарі, знижується ризик втрати важливої інформації та спрощується контроль за прогресом навчання. Крім того, функціонал залишення коментарів після перевірки роботи дозволяє більш якісно взаємодіяти між викладачем і студентом, вказувати на помилки та давати рекомендації для подальшого покращення.

Особливо корисною є можливість повторної здачі завдання, що мотивує здобувачів освіти до самовдосконалення. Вся інформація зберігається у надійній централізованій базі даних, що гарантує стабільний доступ до неї з будь-якої точки світу за наявності підключення до мережі Інтернет. Такий підхід є надзвичайно актуальним в умовах дистанційного або змішаного навчання.

Перевагами вебзастосунку Edelways є:

– інтуїтивно зрозумілий інтерфейс, який дозволяє користуватися системою навіть особам із низьким рівнем цифрової грамотності;

– адаптивний дизайн, що забезпечує коректну роботу на різних пристроях: комп’ютерах, планшетах, смартфонах;

– автоматизація рутинних процесів, яка дозволяє зекономити час викладачам і забезпечити швидкий зворотний зв’язок студентам;

– можливість повторної здачі завдань із урахуванням отриманих коментарів, що стимулює до навчання та саморозвитку;

– зменшення навантаження на викладача завдяки автоматизованій перевірці тестів і системі виставлення оцінок;

– підвищення якості організації навчального процесу завдяки чіткій структурі, журналу оцінювання та прозорій системі контролю.

Безпека даних – ще один ключовий аспект, на який робиться акцент у системі Edelways. Вся інформація зберігається в централізованій базі даних, з використанням сучасних методів шифрування та авторизації користувачів. Це забезпечує конфіденційність персональних даних і знижує ризик несанкціонованого доступу.

1.4 Висновки дослідження предметної області та постановки задачі

Отже, аналіз предметної області показав, що створення програмних рішень для реалізації автоматизованої перевірки знань є актуальним. Предметна область розвивається і потребує автоматизації.

Аналіз вимог та формування технічного завдання допомогли краще зрозуміти які саме аспекти предметної області мають бути автоматизовані і яким чином. Також було з’ясовано основні сутності предметної області та описано їх взаємодію за допомогою функціональних особливостей системи.

Вебзастосунок для автоматизації контролю знань студентів є актуальним і практичним рішенням для модернізації освітнього процесу в умовах стрімкого розвитку цифрових технологій. Його розробка обумовлена потребою в ефективному інструменті для автоматизації навчання, забезпечення зручного доступу до матеріалів, організації зворотного зв’язку між учасниками освітнього процесу та збереження всієї важливої інформації в єдиному безпечному середовищі. Завдяки широкому функціоналу та зручності у використанні, система здатна суттєво підвищити якість освітніх послуг та оптимізувати навчальні процеси як для викладачів, так і для здобувачів освіти.

2 ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Проектування архітектури програмного продукту

Програмний продукт повинен бути чітко спроектований перед початком його реалізації, проте це не завжди вдається та вимагає багато досвіду від фахівця. З цієї причини, внесення несуттєвих архітектурних рішень в ході розробки дозволяється, але при цьому вже написаний програмний код має бути змінено відповідно до нових вимог.

Щоб не допустити чисельних змін в архітектурі, необхідно її спроектувати так, щоб вона була максимально мобільна і гнучка, а її компоненти не мали б сильної залежності між ними. Для цього на практиці використовуюся доволі різні підходи, але найпопулярнішими є розділення серверної та інтерфейсної частини на два окремих проекти. Такий підхід ще називають «безголовим», тому, що користувач може використовувати функціональні можливості серверу без інтерфейсної частини, посилаючи напряму спеціальні запити до нього, але цим зазвичай користуються розробники. Даний підхід зображено на рисунку 2.1.

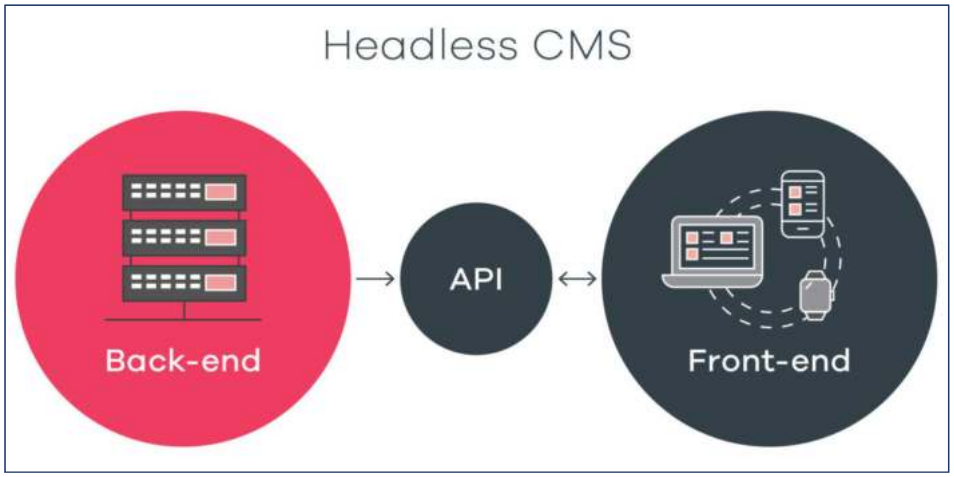


Рисунок 2.1 - «Безголова» архітектура

Основними перевагами є те, що інтерфейсна частина є збірною в функціональному плані і може використовувати не один сервер. Якщо інтерфейсна частина використовує більше одного сервера то такий підхід

називається мікросервісним, де для кожного функціонального рішення створюється окремий серверний проект.

Розбиття одного сервера на декілька дає багато переваг, але при цьому деякі дані необхідно буде дублювати. Загалом, децентралізація даних в різних базах забезпечує стабільність та захищеність, але в той же час архітектура має бути близька до ідеальної щоб правильно оперувати даними в різних базах в одній збірній інтерфейсній частині.

Оскільки мікросервісна архітектура потребує потужних навичок в проектуванні, було прийнято рішення обрати «безголову» архітектуру як основу для всієї системи в цілому.

Вибраний підхід проектування дозволяє обрати абсолютно різні технології для кожної з частин. В аналітичному розділі вже було оглянути сучасні провідні технології та обраний стек як для серверної, так і для інтерфейсної частин.

Перед початком проектування архітектури, потрібно чітко знати які функціональні можливості має виконувати майбутня система. Для цього буде використано діаграму Use Case, яка є в списку популярних UML діаграм. UML - уніфікована мова моделювання, що дозволяє робити чітке моделювання майбутнього програмного продукту перед початком його реалізації.

Діаграма прецедентів [12] (Use Case), що представлена у додатку відображає сутності (акторів), що зображуються у вигляді чоловічків та варіанти використання, що зображуються у вигляді овалів. Актор – це людина чи будь-яка інша система, що має змогу взаємодіяти з розроблюваною системою. Кожен актор має свій варіант взаємодії – на це вказують стрілки на діаграмі, що ведуть від актора до овалу з назвою варіанта, однак, яким чином буде реалізована ця дія не вказується.

Так ми можемо бачити, що Анонімний користувач має змогу або зареєструватися і продовжити роботу в якості Користувача, або вийти з додатку. Актор Користувач був приєднаний до групи її адміністратором за допомогою спеціального коду.

Він має доступ до більшості функцій системи і може взаємодіяти з нею через такі варіанти:

– переглянути теорію ;

– здати завдання;

– приєднатися до групи;

– вийти з групи;

– створити власну групу;

– переглянути результати.

Актор Адміністратор – це той користувач, що створив свою групу і в нього з попереднім типом користувача є більш розширені можливості, такі як:

– видалення учасника з групи;

– приєднання учасника в групу;

– перевірка завдання;

– написання коментаря до завдання;

– перегляд результатів всіх учасників;

– додавання теорії та створення нових завдань.

Після визначення всіх функціональних можливостей потрібно зрозуміти як система буде взаємодіяти з користувачем в цілому. Для цього використаємо діаграму послідовності. Для прикладу буде достатньо спроектувати хоча б одну функціональну можливість системи щоб зрозуміти як вона працюватиме в цілому.

Діаграма послідовності [13] (Sequence), що представлена у додатку відображає взаємодії об'єктів впорядкованих за часом. У проекті були створені та використані такі об'єкти:

– користувач;

– інтерфейс;

– сервер;

– база даних.

Діаграма кооперації (Cooperation), що представлена у додатку є іншим варіантом діаграми послідовностей. Тобто всі об'єкти та повідомлення

дублюються. Вона більш компактна, на ній зручніше розташовувати компоненти і відстежувати потоки подій. Так, на діаграмі в додатку можна побачити, що вона повністю повторює зв'язки зображені на діаграмі послідовності.

Проаналізувавши діаграму, можна зрозуміти, що будь-яка дія користувача має оброблятись інтерфейсною частиною, яка в свою чергу надсилає запит на сервер, який його опрацьовує та дістає необхідні з бази дані, відсилаючи назад на інтерфейсну частину. В цей час користувач бачить анімацію завантаження. Інтерфейсна частина отримує відповідь, зберігає дані в тимчасове сховище та показує результат користувачу.

Після загального проектування системи в цілому та визначення її функціональних можливостей за допомогою спеціальних діаграм почнемо детальне проектування з серверної частини. Фреймворк Laravel дозволяє створювати програмний продукт на основі модульної архітектури з використанням ООП та популярних шаблонів програмування, з можливістю підключення сторонніх модулів. Кожен новий модуль повинен мати однакову архітектуру.

Всі класи, що описують модель даних мають знаходитись в директорії Models та містити в собі виключно методи доступу до даних з бази та обраховуваних даних, що стосуються цієї моделі. Також по замовчуванню кожна модель наслідує клас, який дозволяє взаємодіяти з базою даних. Можна сказати, що в кожній моделі вже реалізований шаблон програмування репозиторій, але такий підхід, коли клас керує своїм станом не є хорошим, проте в нашому випадку це буде краще ніж створення власних репозиторіїв, що по суті дублюватимуть код.

Класи, що обробляють запит на сервер мають знаходитись в директорії Controllers та відповідати за запити тільки одної сутності. В цих класах має бути набір методів в залежності від кількості запитів. В кожному методі може бути присутня лише перевірка даних на коректність, виклик сервісу для обробки логіки та повернення результату. В даних методах не має бути реалізації логіки.

Вся логіка, яка пов’язана з обробкою даних повинна бути винесена в спеціальні сервіси, що мають знаходитись в директорії Services. До цієї логіки відноситься виклик різних функцій для роботи з даними через репозиторій в залежності від різних ситуацій, а також повертання помилки при її виникненні до класу, який відповідає за обробку запитів.

Фреймворк Laravel дозволяє динамічно створювати структуру бази даних в спеціальних файлах міграції, тому всі розмітки таблиць та створення ключів мають знаходитись в цих файлах. Жодні зміни в базу даних не мають бути можливі без попередньої декларації їх в цих спеціальних файлах.

Декларація всіх можливих запитів та прив’язка їх до відповідного методу, що оброблюватиме його має бути в спеціальному файлі з назвою api.php. В цьому ж файлі мають помічатись запити, які є захищені та не мають загального доступу.

Якщо кожен модуль буде створений з дотриманням наведених вище архітектурних правил, то така система зможе називатись модульною і буде легкою як в тестування так і в підтримці. Схему модульної архітектури серверної частини представлено на рисунку 2.2.

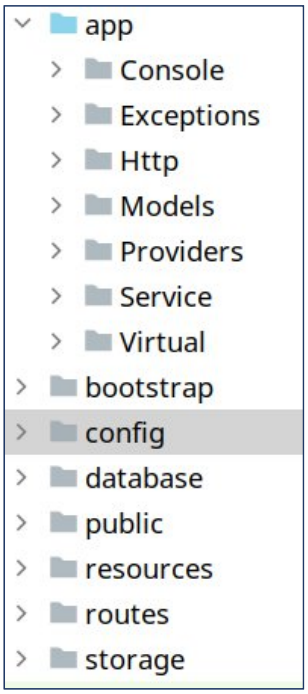


Рисунок 2.2 – Серверна архітектура

Після детального проектування архітектури директорій та модулів, необхідно створити діаграму класів, яка визначатиме основні класи, поля методи та взаємодію між ними. Ця діаграма є дуже важливою, тому її реалізація може зайняти багато часу. Діаграма класів [14] представлена у додатку.

Наступним кроком буде детальне проектування інтерфейсної частини. В основу буде також покладена модульна архітектура [15]. Для кожної сутності має бути розроблений окремий модуль. Кожен модуль повинен мати однакову структуру.

Вся логіка роботи з АПІ [16] повинна бути винесена в окремий файл в директорії store. Усі отримані дані повинні бути збережені в загальному сховищі. До цих даних слід реалізувати методи доступу. Також в цьому файлі слід реалізувати логіку обробки помилок та повертання результату виконання запиту в компонент.

Вся інтерфейсна частина являтиме собою набір компонентів. Можуть існувати як дочірні так і батьківські компоненти. Передача даних між компонентами, якщо вони залежні, дозволена, але тільки в одному напрямку, якщо це не службовий компонент. Компонент [17] – реалізація якоїсь маленької функціональної частинки,що інкапсулює в собі як відображення і стилі, так і логіку роботи, яка в свою чергу взаємодіє з загальним сховищем та сервісами для роботи з АПІ.

Об’єднання групи компонентів має бути в спеціальному компоненті, що має знаходитись в директорії views і являти собою відображення конкретної веб сторінки. Доступ до цієї сторінки має бути за допомогою спеціального посилання, яке задекларовані в файлі з шляхами конкретного модуля.

Файл, де будуть прописані всі шляхи та прикріплені до кожного з них конкретні відображення має знаходитись в директорії routes, та слугувати розміткою всіх існуючих шляхів в поточному модулі.

З поміж всіх модулів інтерфейсної частини буде виділятись модуль з назвою Framework, де буде реалізована базова функціональність, головні конфігурації та

підключення всіх компонент в єдину робочу систему. Схему модульної архітектури інтерфейсної частини представлено на рисунку 2.3.

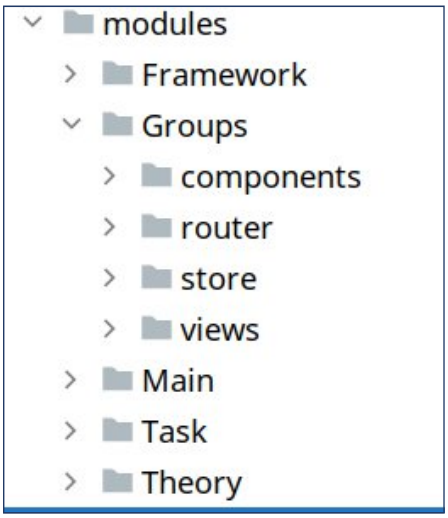


Рисунок 2.3 – Інтерфейсна архітектура

Фінальним етап буде створення діаграми, що відповідатиме за розгортання майбутнього проекту.

Діаграма розгортання у додатку призначена для візуалізації елементів і компонентів програми, що існують лише на етапі її виконання (runtime). Розробка діаграми розгортання, як правило, є останнім етапом специфікації моделі програмної системи.

Діаграма розгортання [18] містить графічні зображення процесорів, пристроїв, процесів і зв'язків між ними. На відміну від діаграм логічного уявлення, діаграма розгортання є єдиною для системи в цілому, оскільки повинна цілком відображати особливості її реалізації

Отже, в цьому розділі було проведено визначення всіх функціональних можливостей, проектування роботи системи в цілому та детальний розбір архітектури та правил побудови кожної з підсистем. Система складається із серверної та інтерфейсної частин. Вони являтимусь собою дві окремі системи, які будуть взаємодіяти за допомогою АПІ.

2.2 Проектування структур даних

В якості основної структури даних проекту використовується СУБД MySQL [19]. Вона надає сховище, що використовуються в традиційних реляційних СУБД (система управління базою даних) [20] з доступом до даних засобами мови SQL (структурована мова запитів).

Реляційні бази даних [21] зберігають дані у вигляді таблиць, що є дуже зручно та структуровано. Кожна таблиця має свій первинний ключ, що слугує ідентифікатором для кожного запису в таблиці. Також задля уникнення дублювання даних в таблицях, використовуються зовнішні ключі, які вказують ідентифікатор сутності в іншій таблиці.

MySQL дозволяє робити різні залежності за допомогою зовнішніх ключів. Наприклад якщо декілька сутностей залежні від однієї, то при видалені батьківської, автоматично видаляться і дочірні записи. Також це додає додаткову перевірку на існування сутності, значення зовнішнього ключа якого буде занесено в таблицю.

Реляційні бази, на відмінну від не реляційних мають чітку структуру кожної запити в таблиці з можливістю перевірки на коректність кожного поля в таблиці. Такий підхід буде більш надійним та правильним для великих додатків із складною структурою даних. Також такі бази даних дозволяють створювати представлення, в яких місяться розраховуванні дані, які оновлюються під час кожного запиту, що дозволяє винести частину логіки написану за допомогою коду в представлення. Такий підхід може суттєво оптимізувати систему.

Як показує практика реляційні СУБД частіше використовуються у великих проектах через свою сталість та структурованість збережених даних, а також через можливість робити чіткий зв’язок між ними.

Для коректної побудови структури бази даних необхідно визначитись із вхідними та вихідними даними. Знаючи більшість вхідних даних можна буде побудувати початкову структуру, щоб коректно їх зберегти та побудувати зв’язки

між ними. Знаючи вихідні дані можна продумати які запити до даних будуть існувати та яким чином вони будуть вибиратись з бази даних. Такий підхід дозволяє значно полегшити проектування структур даних.

Вхідними даними у нашому випадку будуть слугувати дані від користувача при створенні нового користувача, групи, завдання чи теоретичного матеріалу. А також адміністратор групи зможе оцінювати виконані завдання та залишати відгук про виконану роботу. Приклад введення вхідних даних в форму створення нового клієнту показано на рисунку 2.4.

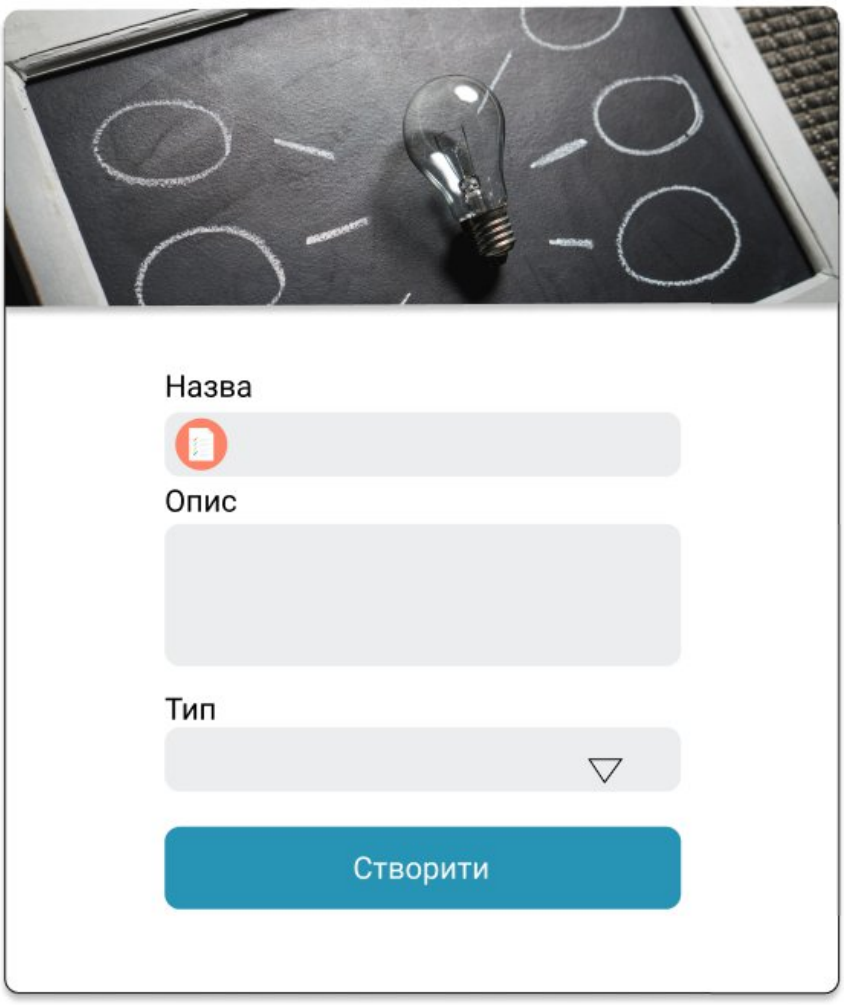


Рисунок 2.4 – Форма створення нового користувача

Вихідними даними нашої системи буде занесення нововведених даних у внутрішню базу даних, завантаження прикріплених файлів на сервер та

відображення їх користувачу при успішній перевірці на правильність форматування. Серед них дані про користувачів, їх групи, створені завдання та теоретичні матеріали. Приклад вихідних даних зображено на рисунку 2.5. При реєстрації користувача чи входу в особистий кабінет повинна відобразитись помилка при введенні некоректних даних. Після перевірки завдання результат перевірки має бути доступний виконавцю завдання включно з відгуком про роботу. Прикріплений до завдання документ повинен бути доступний для відповідальної за перевірку завдання особи. Особисті дані профілю мають бути доступні для редагування. Особистий кабінет користувача може бути видалений за вимогою.

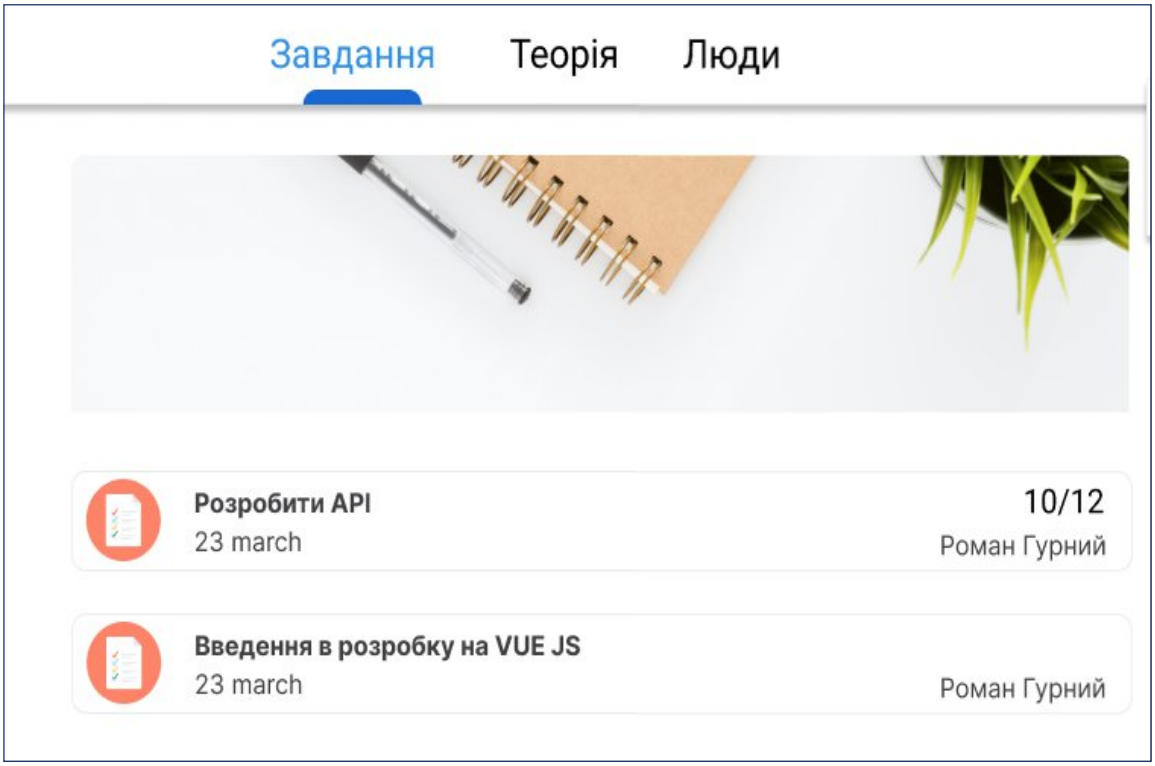


Рисунок 2.5 – Список поточних завдань

Після визначення вхідних та вихідних даних можна приступити до їх структурування та проектування бази даних. Основними сутностями в проекті будуть: завдання, теорія, користувач, група. Отже для кожної сутності має бути створена як мінімум одна таблиця в базі та реалізовані правильні зв’язки.

Для сутності користувач виділимо такі основні поля як ім’я, електронна скринька, пароль, шлях до зображення аватара і звісно ж головний ключ для ідентифікації.

Для сутності завдання такими полями будуть назва, опис, шлях до прикріпленого файлу з поясненнями (опціонально), максимальна кількість балів та посилання на користувача, який створив це завдання. Також буде створено ще дві допоміжні таблиці, які вміщуватимуть в собі дані про здану роботу та результат перевірки роботи.

Що стосується сутності групи, то тут все максимально просто: назва групи та її автор – для головної таблиці. Також буде створена ще одна допоміжна таблиця, де буде посилання на групу та посилання до приєднаного до цієї групи користувача.

Ще однією сутністю буде теорія. До неї входитимуть такі поля як назва, шлях до прикріпленого документу та автор. Сам документ буде завантажено на сервер в спеціальну директорію щоб всі користувачі, які належатимуть до тієї групи, мали до нього доступ.

Вище наведено приблизну структуру даних майбутнього додатку. Звісно в процесі розробки структура може бути змінена в залежності від потреб, але ці зміни не будуть кардинальними. Повна схема структури бази даних зображена на додатку.

2.3 Проектування інтерфейсу

Інтерфейс програми – одна з найбільш важливих його частин, адже від зрозумілості та зручності його використання залежить наскільки багато користувачі захочуть користуватись таким програмним продуктом. Тому є помилкою, коли під час проектування на інтерфейсну частину не звертають належної уваги, а надають перевагу його розробці динамічно, в залежності від розроблених функціональних можливостей.

При такому підході проект не буде мати якийсь спільний дизайн і буде мати максимально хаотичний, в той час простий, але не зрозумілий інтерфейс. Звісно можна зробити спочатку такий, а отім його оптимізувати і покращити, але з економічної точки зору це буде вартувати як створення ще однієї інтерфейсної частини.

На сьогоднішній день існує чудовий вихід з такого положення. Багато програмних засобів дозволяють створювати дизайн без залучення жодного програміста чи верстальника. Час створення такого дизайну як і затрати на нього будуть мінімальні. Також є низька ціна помилки і можливість створення багатьох варіантів одного й того ж рішення на вибір.

Такий підхід безумовно надає багато переваг. Одною і дуже важливою з них є можливість розробки дизайну як для великих так і для малих мобільних екранів. Проектування дизайну означає вибір основних стилів проекту, створення власних рішень для вже знайомих функцій.

Також не менш важливу роль потрібно приділити UX (показник, що означає зрозумілість інтерфейсу) показнику, адже важливо запитати думку рядових користувачів про зрозумілість розробленого дизайну, перш ніж почати його розробку.

Для проектування дизайну був обраний безкоштовний онлайн додаток Figma [22]. Цей додаток дозволяє в максимально швидко створити дизайн за допомогою векторних примітив та зручному набору інструментів.

Перш за все пройдемо реєстрацію в цьому програмному продукті та створимо новий проект. Приклад створення нового проекту показано на рисунку 2.6. Після чого необхідно вказати розміри робочої області. Всі зміни в подальшому будуть автоматично збережені.

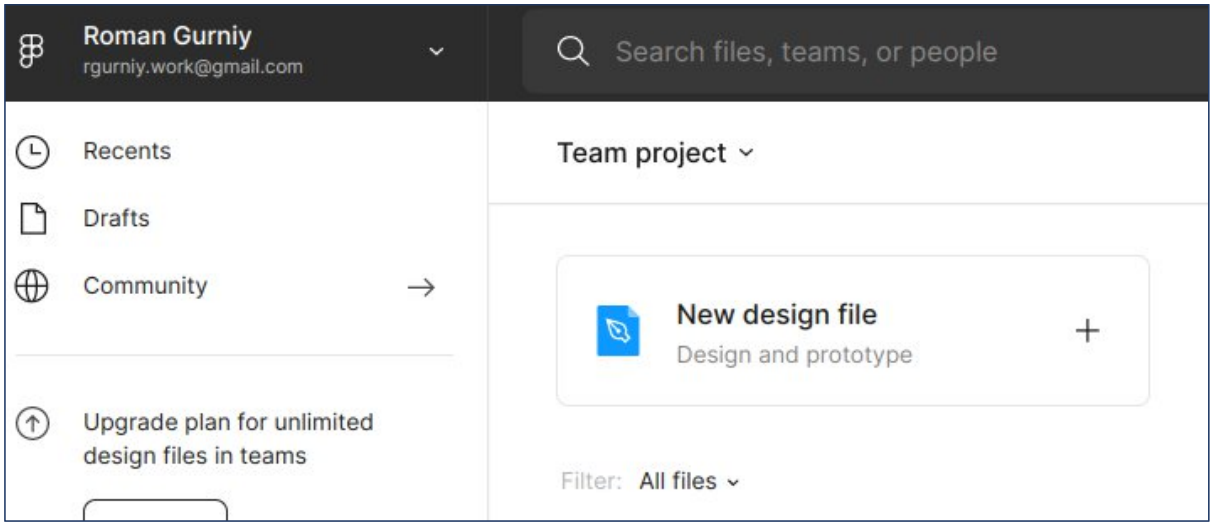


Рисунок 2.6 – Створення нового проекту в Figma

Наступним кроком буде початок проектування інтерфейсу. Для початку створимо форму для входу в систему, обравши основний колір додатку, логотип та стиль дизайну. Приклад дизайну форми входу показано на рисунку 2.7.

Основний колір додатку визначає який колір використовуватиметься в основі всіх функціональних компонентів. До таких компонентів належать кнопки, панелі, вкладки та інше. Після аналізу кольорів, для проекту було обрано світло синій колір як основний.

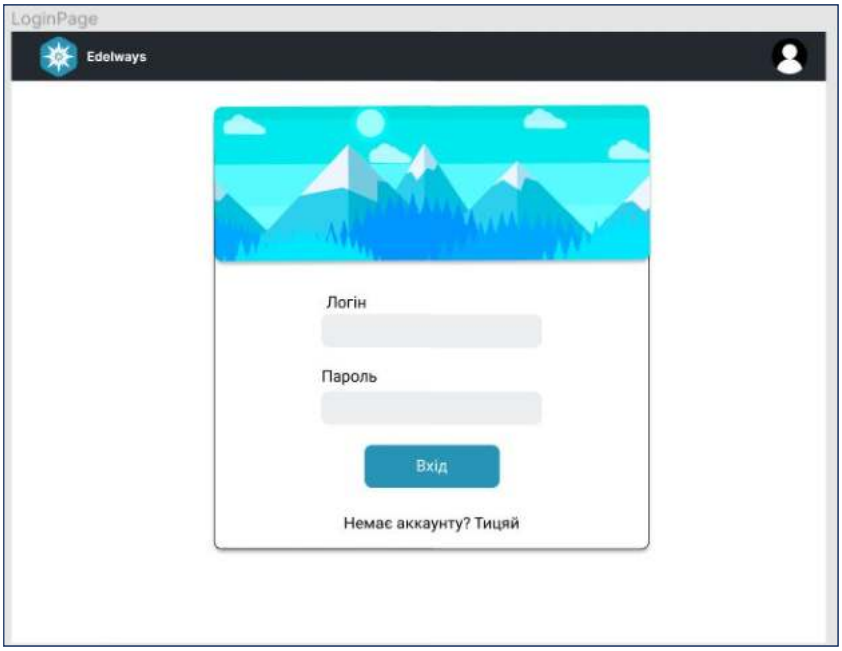


Рисунок 2.7 – Форма входу в систему

Далі необхідно розробити весь дизайн, що покривав б усі описані функціоналі можливості проекту. Для цього створюватимемо вікно за вікном, добираючи необхідні стилі.

Наприклад, створимо головне вікно для програми, що дозволятиме перемикати вміст за допомогою зручних вкладок у верхній частині, а панель адміністратора буде розташовано у випадному списку, який буде видно при натисканні маленької кнопки. Ця кнопка буде відображатись тільки якщо поточний користувач буде адміністратором групи. Також слід додати ліве меню для навігації між групами, що буде з’являтись при натиску на спеціальну кнопку. Таке рішення буде чудово виглядати на мобільному пристрої і дозволить легко перемикатись між групами.

Слід зазначити, що інтерфейс без жодних дизайнерських фото виглядає досить нудно і не дає відчуття спокою та затишку, тому для проекту «Edelweys» спеціально були підібрані кращі дизайнерські варіанти. Результат проектування головного вікна показано на рисунку 2.8.

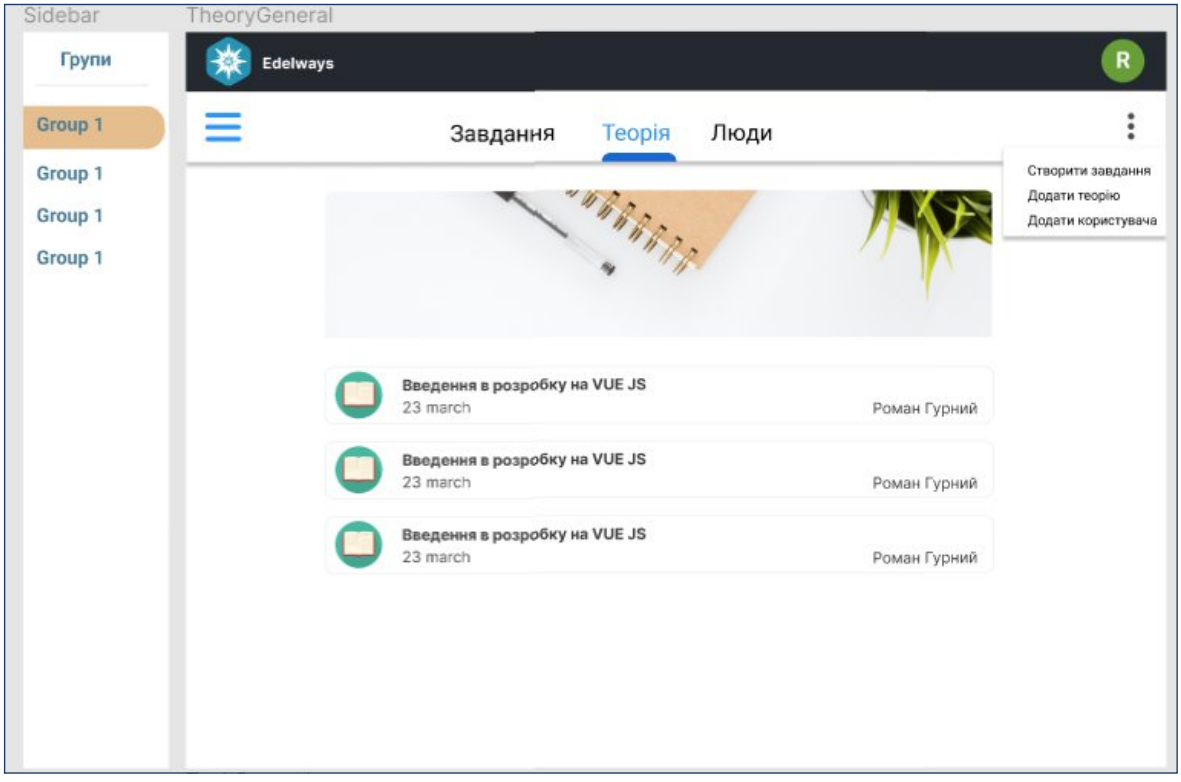


Рисунок 2.8 – Дизайн головного вікна

Отже, у цьому розділі було проаналізовано всі популярні архітектурні рішення та обрано одне з них. Таким архітектурним рішенням стала «безголова» розробка. Після чого було детально спроектовано як серверну, так і інтерфейсну частини. Наступним кроком було проектування структури даних та її оптимізація. Фінальним кроком стало проектування інтерфейсу для всіх функціональних можливостей проекту.

2.4 Аналіз та вибір технологій і методів реалізації

На сьогоднішній день існує велика кількість різноманітних методів та рішень створення програмного забезпечення для веб індустрії. Розробка є складним процесом, тому зміна технологічного стеку в процесі розробки може поставити хрест на проекті в цілому. Тож необхідно провести огляд наявних рішень та методів, звернути увагу на переваги та недоліки та підібрати бажаний стек технологій для розробки.

Спочатку необхідно визначитись із технологіями та засобами для серверної частини. Існує багато мов програмування, які дозволяють писати серверні рішення. Розглянемо популярні мови для серверного програмування більш детально.

Java – універсальна мова програмування, надає велику кількість бібліотек для написання серверної логіки, перевірена часом, на ній успішно написано багато велетенських програм, актуальна і в сьогоднішній час. На цій мові написаний популярний фреймворк Spring [23], що полегшує в декілька разів повноцінну розробку вебсайтів. Програмуючи на java пишеться набагато більше слів, ніж на інших мовах для однакових задач, тому можна вважати цю мову надто багатослівною. Також ця мова має середній поріг входу, тому перед початком розробки потрібно буде багато часу приділити вивченню цієї технології.

Python [24] – теж універсальна мова програмування, вона включає в себе бібліотеки для вирішення будь-якого завдання, має приємний мінімалістичний синтаксис, динамічну типізацію та може похизуватися своєю

кросплатформеністю. На цій мові написаний фреймворк Jango [25] спеціально розроблений для комфортної веб розробки. Має середній поріг входу і хоч і потребує менше часу на вивчення ніж java, проте теж не дозволить зайнятись одразу швидкою розробкою.

PHP [26] – серверна мова програмування, створена спеціально для написання серверної логіки. Має приємний синтаксис, багато готових методів, що написані на C і дають велику швидкодію. Містить як статичну так і динамічну типізацію. На цій мові написані такі популярні фреймворки для Laravel [27] та Simfony [28]. PHP має низький поріг входу, тому розробка програмного продукту займе менше часу.

Отже, для розробки серверної частини дипломного проекту було вибрано мову програмування PHP, тому що в ній звичний для багатьох синтаксис, проста робота з ООП, легкість в імплементації патернів, за допомогою спеціальних службових коментарів можна забезпечити статичну типізацію в тих місцях програми, де це необхідно, містить багато вбудованих методів для роботи з файлами, базами даних та масивів.

Обравши мову програмування, наступним кроком буде вибір фреймворку, щоб пришвидшити розробку та не придумувати велосипед вкотре. PHP пропонує два найбільш популярних фреймворки: Simfony та Laravel. Перший є багатофункціональний, але є доволі низькорівневим і потребує багато знань для роботи з ним. Laravel – фреймворк, що по суті є надбудовою над Simfony, де було піднято на рівень абстракції вище всі засоби для розробки серверної логіки, взаємодії з базою даних, тощо. Даний фреймворк є дуже простий у вивченні, містить багато готових рішень та має дуже зручний інструмент для взаємодії з базою даних. Цей фреймворк буде використаний для розробки серверної частини.

Наступним кроком буде вибір засобів розробки для інтерфейсної частини. На сьогоднішній день мало сайтів створюються за допомогою чистої мови програмування JavaScript, і все частіше використовуються якісь бібліотеки чи фреймворки для роботи з нею. Найпопулярнішими фреймворками є Angular та

Vue Js, а найпопулярнішою бібліотекою є React.

React [29] - відкрита JavaScript бібліотека для створення інтерфейсів користувача, яка покликана вирішувати проблеми часткового оновлення вмісту вебсторінки, з якими стикаються в розробці односторінкових застосунків. Розробляється Facebook, Instagram і спільнотою індивідуальних розробників.

React дозволяє розробникам створювати великі вебзастосунки, які використовують дані, котрі змінюються з часом, без перезавантаження сторінки. Його мета полягає в тому, щоб бути швидким, простим, масштабованим. React обробляє тільки користувацький інтерфейс у застосунках. Це відповідає видові у шаблоні модель-вид-контролер, і може бути використане у поєднанні з іншими JavaScript бібліотеками або в великих фреймворках MVC [30], таких як AngularJS. Як бібліотеку інтерфейсу користувача React найчастіше використовують разом з іншими бібліотеками, такими як Redux.

Angular [31] - фронтенд-фреймворк з відкритим кодом, написаний на TypeScript, який розробляють Google, під керівництвом Angular Team, спільнота приватних розробників та інші компанії. Angular - це AngularJS (2010 р.), який переосмислили та який був повністю переписаний тією ж командою розробників у 2016 році та отримав назву Angular 2. На цей час остання стабільна версія - Angular 11. Має доволі високий поріг входу, та потребує багатьох навичків в роботі з основними засобами веб розробки.

Vue [32] – це прогресивний фреймворк для створення користувацьких інтерфейсів. На відміну від фреймворків-монолітів, Vue створено придатним для поступового впровадження. Його ядро в першу чергу вирішує завдання рівня представлення, що спрощує інтеграцію з іншими бібліотеками, та існуючими проектами. З іншого боку, Vue повністю підходить і для створення складних односторонніх додатків, якщо використовувати його спільно з сучасними інструментами та додатковими бібліотеками.

Загальною і основною перевагою всіх перелічених вище фреймворків та бібліотек є реактивність. Реактивне програмування [33] – це парадигма

програмування, побудована на потоках даних і розповсюдженні змін. Це означає, що у мовах програмування має бути можливість легко виразити статичні чи динамічні потоки даних, а реалізована модель виконання буде автоматично розсилати зміни через потік даних.

Реактивне програмування спочатку пропонувалося як засіб простого створення інтерфейсів користувача, анімацій у системах реального часу, але стало загальною парадигмою програмування.

Отже, для реалізації інтерфейсної частини було обрано фреймворк VueJS з доповненнями. До доповнень належать робота з переходом на інші сторінки, робота з загальним сховищем та інше. Vue має відносно низький поріг входу і інтуїтивно зрозумілий синтаксис, тому розробка за допомогою цього засобу має бути проста і швидка.

2.5 Висновки проектування програмного забезпечення

У процесі проектування проекту було обрано клієнт-серверну архітектуру, де сервер та клієнт будуть реалізовуватись окремими незалежними модулями, а їх взаємодія буде реалізована за допомогою АПІ. Під час проектування були представлені різні діаграми, які відображають основні сутності та їх взаємозв’язки.

Також було розглянуто наявні засоби для розробки ПЗ та обрано стек технологій для інтерфейсної та серверної частин. Під час проектування структур даних була проведена їх нормалізація та вибудувані чіткі зв’язки і визначені поля для всіх сутностей системи. На даному етапі є зрозумілим які функціональні можливості будуть реалізовані та яким чином.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ЗАСТОСУНКУ

3.1 Програмування структур даних

Правильна реалізація спроектованої структури потребує особливої уваги та професіоналізму, адже допустивши якусь незначну помилку, в майбутньому для її виправлення може знадобитись пожертвувати даними якоїсь таблиці чи переробці якоїсь функціональної частини.

Для реалізації структури бази даних спочатку її необхідно встановити та підключити до проекту. Для безпечного встановлення та використання бази даних без впливу на операційну систему буде використовуватись Docker [34] (сервіс віртуалізації), що дозволить помістити базу даних в окремий незалежний контейнер, доступ до якого буде здійснюватись за допомогою спеціального порту. Щоб створити контейнер виконаємо конфігурацію спеціального файлу як на рисунку 3.1.

```
  magento2_mysql:
    image: mysql:5.7
#    container_name: magento2_mysql
    volumes:
      - mysql:/var/lib/mysql:delegated
      - ./config/mysql/my.cnf:/etc/mysql/conf.d/my.cnf
      - ./dumps:/dumps
    ports:
      - 3307:3306
    expose:
      - 3306
    env_file:
      - ./env/mysql.env
```

Рисунок 3.1 – Конфігурація контейнера бази даних

Після успішного запуску контейнера створимо нову базу даних в ньому за допомогою команди, що зображена на рисунку 3.2. Після створення бази даних до неї можна підключитись ввівши всі необхідні дані.

```
create database edelways;
```

Рисунок 3.2 – Створення бази даних

Наступним кроком буде підключення бази даних до серверної частини та перевірка наявності з'єднання. Для цього вкажемо такі дані як порт, назву бази даних, ім'я адміністратора та його пароль, а також в ролі хосту буде виступати назва нашого контейнера. Після введення даних натиснемо на відповідну кнопку щоб переконатись що з'єднання працює коректно.

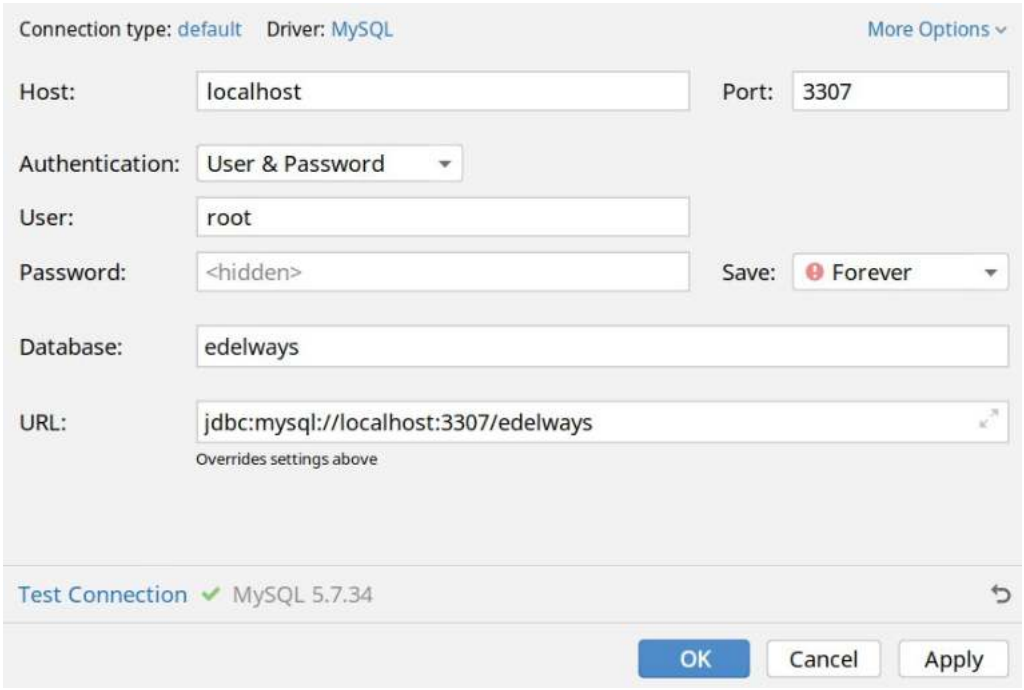


Рисунок 3.3 – Підключення до бази даних

Створення структури таблиць в цьому проекті не буде відбуватись за допомогою SQL коду, натомість буде використаний наявний функціонал фреймворку Laravel, що слугує додатковим рівнем абстракції та простішої взаємодії з базою даних.

Кожна зміна в базу даних документується міграцією. Міграція – спеціальний клас що повинен містити реалізацію двох функцій. Перша – запуск міграції на виконання, друга – видалення всіх змін і повернення до початкового

стану, що був до моменту запуску міграції.

Додатковий рівень абстракції дозволяє автоматично реалізовувати описану в ньому структуру даних для багатьох популярних реляційних баз даних, таких як MariaDB, PostgressSQL, MySQL, OracleSQL та інші, адже якби реалізація структури даних була написана за допомогою SQL скрипту, то його б довелось перероблювати під кожну з перерахованих баз даних, адже у кожній є своє правила та вимоги до синтаксису.

Першим ділом створимо сутність користувача та групи згідно спроектованої схеми. Наступним кроком створимо сутність завдання та правильно запрограмуємо звя’зки з іншими таблицями за допомогою зовнішніх ключів. Приклад реалізації міграції для сутності завдання наведено на рисунку 3.4.

```
Schema::create( table: 'tasks', function (Blueprint $table) {
    $table->increments( column: 'id');
    $table->string( column: 'name');
    $table->unsignedInteger( column: 'group_id')->nullable( value: false);
    $table->unsignedInteger( column: 'user_id')->nullable( value: false);
    $table->string( column: 'description');
    $table->unsignedInteger( column: 'mark');
    $table->string( column: 'document_path')->nullable( value: true);
    $table->timestamps();
    $table->foreign( columns: 'group_id')->references( columns: 'id')->on( table: 'groups')
        ->cascadeOnDelete();
    $table->foreign( columns: 'user_id')->references( columns: 'id')->on( table: 'users');
```

Рисунок 3.4 – Міграція для сутності

Сутність була створена таким чином, що при видаленні групи, всі завдання, які належали їй будуть видалені автоматично. Такий підхід дозволить не зберігати в базі не актуальні дані, що може пришвидшити роботу проекту. Результат виконання міграції показано на рисунку 3.5.

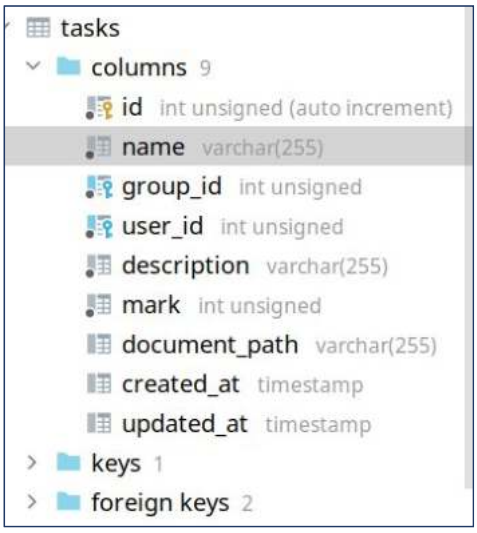


Рисунок 3.5 – Результат реалізації структури даних

Після реалізації структури даних у вигляді відповідних таблиць та зв’язків між ними у базі даних, необхідно створити додатковий рівень абстракції, щоб могли ними апелювати і писати логіку програми абстраговано, та не прив’язуючись до джерела даних. Робота з абстракціями це хороший тон, який був перевірений на багатьох проектах, хоча мінусом є те, що втрачається його швидкодія.

Для нового шару абстракції використаємо шаблон програмування, що зветься модель. Згідно переднього проектування цей шаблон міститиме тільки перелік полів, методи доступу до них та методи що повертатимуть поля, що розраховуються. Взаємодія моделі даних з конкретною таблицею в базі даних реалізовуватиметься через шаблон репозиторій.

Фреймворк Laravel постачає базовий клас моделі, що вже містить весь функціонал, що мав би виконувати репозиторій, тому немає сенсу його створювати ще раз. До його можливостей належать вибірка з бази за умовою та порядком, оновлення даних, видалення, зберігання та інше.

Спочатку створимо модель для сутності група. За допомогою спеціальних службових коментарів помічаємо всі доступні поля, завдяки цьому методи доступу до них формуються магічним чином. Також вказуємо за допомогою присвоєння до спеціальних службових змін базового класу які поля можна

редагувати і які мають бути приховані при вибірці.

Наступним кроком буде створення методів, то відповідатимуть за постачання розрахункових полів. Такі дані немає сенсу зберігати в базі даних, адже їх завжди можна розрахувати в будь-який момент. Прикладом такого поля може бути повне ім’я користувача чи відформатована в правильному форматі дата його народження.

Динамічні поля також дозволяють завантажити батьківську сутність чи список дочірніх сутностей, що зменшує необхідність написання цього функціоналу в шарі логічної реалізації. Такий підхід збільшить зрозумілість та читабельність коду. Приклад реалізації моделі зображено на рисунку 3.6.

```php
/**
 * @property $name
 * @property $code
 * @property $user_id
 * @property $id
 * @property $canAccess
 * @property $isAdmin
 */
class Group extends Model
{
    use HasFactory;

    const ID = 'id';
    const NAME = 'name';
    const CODE = 'code';
    const USER_ID = 'user_id';

    protected $table = 'groups';
    protected $primaryKey = 'id';
    public $timestamps = true;

    protected $fillable = [
        self::NAME,
        self::CODE,
        self::USER_ID
    ];

    protected $hidden = [
        'user'
    ];
```

Рисунок 3.6 – Приклад моделі сутності

Отже, у цьому розділі було реалізовано структуру даних проекту у вигляді створення відповідних таблиць за допомогою реляційного СУБД MySQL. Також були реалізовані коректні зв’язки між таблицями для запобіганню засмічення

бази даних та правильної і швидкої вибірки даних. Фінальним етапом було створення нового рівня абстракції для роботи з цими даними на рівні логіки, а не джерела даних.

3.2 Програмування логіки роботи серверної частини

Згідно описаної розділами вище архітектури серверної частини, вона являтиме собою набір запитів, що мають бути задокументовані і доступні всім охочим. Для кожного запиту повинен бути окремий метод, що його обслуговує. Для кожної сутності може існувати декілька запитів, але реалізація запитів для іншої сутності має бути в окремому класі. Такі класи називаються контролери. Згідно проектування, в них не повинно міститись багато логіки, а вся вона має бути винесена в спеціальні сервіси. Принцип роботи сервера зображено на рисунку 3.7.

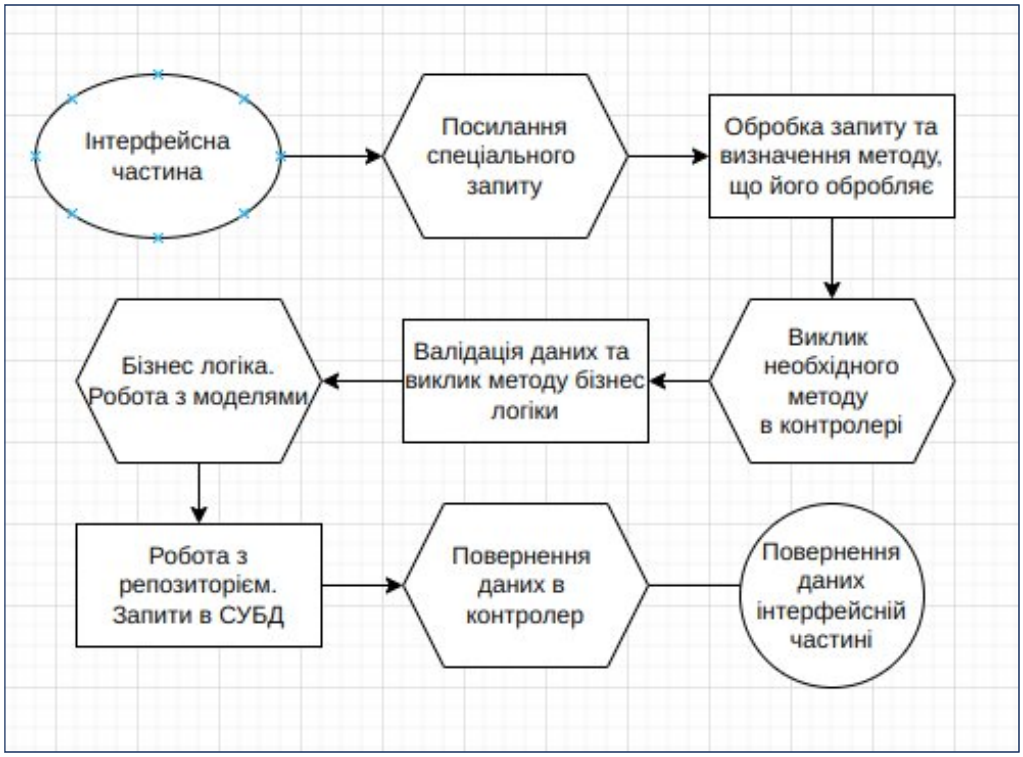


Рисунок 3.7 – Схема роботи сервера

Першим кроком задекларуємо новий запит в спеціальному файлі api.php, обравши тип доступу для нього. Для кожного запиту є два типи доступу: для всіх користувачів чи тільки для авторизованих. Щоб створити запит, необхідно спочатку створити клас контролера та метод що буде обробляти цей запит. Фінальним кроком буде декларація запиту у вигляді його назви, шляху до контролера, та метода, що його оброблятиме. Приклад декларації запиту представлено на рисунку 3.8

```
Route::group(['middleware' => ['auth:sanctum']], function () {
    Route::get( uri: '/profile', [AuthController::class, 'userData']);
```

Рисунок 3.8 – приклад декларації запиту

Кожен метод, що обробляє запит повинен бути задокументований за допомогою спеціальних службових коментарів. Це дозволить скористатись модулем Swagger [35] (сервіс для графічного подання АПІ) для формування графічного відображення АПІ та можливості його зручного тестування і використання сторонніми користувачами.

Даний модуль будує загальний файл з описом АПІ на основі цих спеціальних коментарів, що мають бути побудовані за спеціальними правилами, які можна знайти на сайті цього модулю. Такий підхід дозволить іншим розробникам не маючи доступу до реалізації серверної частини будувати інтеграцію з ним. Також не потрібно вручну створювати документацію функціональних можливостей. Автоматична генерація документації на основі коду гарантує її актуальність і точність. Приклад службового коментаря зображено на рисунку 3.9.

```php
/**
 * @OA\Post(
 *      path="/api/sign-out",
 *      operationId="logoutUser",
 *      tags={"Users"},
 *      summary="Logout user",
 *      description="Returns success message",
 *      security={{"bearer_token":{}}},
 *   @OA\Response(
 *      response=200,
 *       description="Success",
 *      @OA\MediaType(
 *           mediaType="application/json",
 *      )
 *   ),
 *      @OA\Response(
 *          response=400,
 *          description="Bad Request"
 *      ),
 *      @OA\Response(
 *          response=401,
 *          description="Unauthenticated",
 *      ),
 * )
 */
public function logout()
{
    $this->authService->removeToken();

    return response()->json( data: 'Success');
}
```

Рисунок 3.9 – Приклад службового коментаря контролера

Наступним кроком буде створення спеціальних сервісів, в яких буде описана основна логіка роботи сервера та буде відбуватись взаємодія з даними за допомогою моделей, що в свою чергу будуть взаємодіяти з СУБД.

Згідно спроектованого раніше рішення для сервісів, кожна сутність повинна мати свій окремий сервіс. Якщо сутність матиме батьківську сутність то вона може використовувати функціонал її сервісу. Напишемо сервіс для роботи із

завданнями, а саме метод, що буде повертати виконану роботу за її ідентифікатором. Для цього спочатку знайдемо відповідний запис за допомогою репозиторію і перевіримо чи він існує. Якщо запис не існує в базі даних, то повертаємо помилка на контролер з відповідним тестом. Якщо запис існує, то отримуємо батьківську сутність і перевіряємо чи має доступ поточний користувач до неї. В випадку відсутності доступу буде повернено відповідно помилку. Якщо доступ є, то повертаємо завантажені дані на контролер, звідки вони серіалізуються і відправляться клієнту. Приклад реалізації сервісу зображено на рисунку 3.10.

```php
/**
 * @throws Exception
 */
public function getComplete($id)
{
    $complete = TaskComplete::query()->find($id);
    if (!$complete) {
        throw new Exception( message: 'Роботу не знайдено');
    }
    $task = Task::query()->find($complete->task_id);
    $this->validatНапишкеTask($task);
    return $complete;
}
```

Рисунок 3.10 – Приклад реалізації сервісу

Ще однією важливою особливістю фреймворку Laravel є використання шаблону програмування, який полягає в створенні класу, що буде відповідати за створення і надання об’єктів інших класів. Такий підхід дозволяє існуючий повертати екземпляр класу при потребі, не створюючи новий, якщо в цьому немає потреби. Це може значно економити ресурси сервера, що покращить швидкодію.

Для використання об’єкту іншого класу в поточному, необхідно задекларувати таку необхідність в конструкторі класі. Тоді при створені такого класу, йому автоматично будуть надані необхідні йому об’єкти. Приклад

реалізації такого конструктора зображено на рисунку 3.11.

```php
/**
 * @var GroupService
 */
private $groupService;

public function __construct(GroupService $groupService)
{
    $this->groupService = $groupService;
}
```

Рисунок 3.11 – Конструктор класу

Отже, в цьому розділі було реалізовану логіку роботи сервера у вигляді декларування можливих запитів за створення обробників для них. Всі запити були задокументовані та доступні в їх графічному зображенні для використання та тестування.

3.3 Програмування інтерфейсної частини

Згідно обраного засобу розробки, інтерфейсна частина буде реалізована за допомогою фреймворку VueJS. Даний фреймворк має компонентну структуру та надає всі переваги реактивності. Також, для цього фреймворку доступні багато доповнень та вже готових функціональних рішень. Перелічені переваги дозволяють встановлювати в проект тільки необхідні модулі, скрадаючи систему як конструктор.

Доповнення являють собою два типи. Перший це доповнення пов’язанні з розширенням функціональних можливостей фреймворку, другий – готові функціональні рішення, зроблені у вигляді готових компонент, використовуючи стандартні можливості фреймворку.

Основний акцент слід звернути на перший тип доповнень, адже згідно наших потреб нам необхідно буде мати можливість переходу на інші сторінки,

взаємодіяти з серверною частиною та створити загальне джерело даних, щоб інкапсулювати роботу з серверною частиною в окремих файлах, виносячи цю логіку з компонентів.

Провівши пошук необхідних для потреб проекту доповнень було обрано такі доповнення:

– Router (для реалізації переходів між сторінками);

– Vuex (для реалізації загального сховища даних);

– Axios (для взаємодії із серверною частиною).

Наступним кроком стане пошук готових функціональних можливостей. Згідно правил хорошого тону, необхідно використовувати якнайменше готових рішень від сторонніх розробників, адже немає ніякої гарантії що описані можливості будуть працювати правильно і в повній мірі відповідати потребам. Після пошуку доповнень, було обрано доповнення Toaster, що відповідає за вивід короткострокових повідомлень. На цьому пошук доповнень закінчено. Список підключених доповнень зображено на рисунку 3.12.

```
const app = createApp(App)

app.use(store)
app.use(router)
app.use(Toaster, options: {
    position: 'top'
})
app.use(VueAxios, axios)
app.component( name: 'icon', icon)
app.mount( rootContainer: '#app')
```

Рисунок 3.12 – Підключення доповнень

Після встановлення фреймворку та всіх необхідних доповнень можна починати реалізацію інтерфейсної частини. Першим ділом створимо модуль Framework, в якому будуть підключатись доповнення та реалізовані якісь базовані універсальні механіки.

До таких механік можна віднести підключення лістингу сторінок для модуля маршрутизації та підключення всіх маленьких сховищ в одне загальне сховище даних, шляхом об’єднання цих даних. Приклад об’єднання сховищ показано на рисунку 3.13.

```
const store = createStore( options: {
    modules: {
        auth,
        sidebar,
        groups,
        theories,
        tasks

    }
})
```

Рисунок 3.13 - Об’єднання сховищ

Наступним кроком стане створення основного модулю, що міститиме в собі універсальні компоненти, що можна буде використовувати безліч раз. Ці компоненти мають бути динамічними, тобто мати змогу налаштовуватись при ініціалізації та мати базові стилі, що відповідають загальній темі. До таких компонентів будуть належати поля вводу даних, кнопки, контейнери для виводу вмісту, вкладки, списки, повідомлення. З таких компонентів необхідно створити також більш складні універсальні компоненти. До прикладу, використовуючи базові компоненти, можна створити конструктор форми, що дозволить відобразити форму з необхідними полями в пару рядків коду. Приклад реалізації простого універсального компоненту для вводу тесту зображено на рисунку 3.14.

```
<template>
  <div class="input">
    <input :type="inputType" class="input-field" required
           :value='modelValue'
           @input='$emit("update:modelValue", $event.target.value)'
    />
    <label class="input-label">{{label}}</label>
  </div>
</template>
```

Рисунок 3.14 – Базовий компонент для вводу тексту

Маючи готову бібліотеку простих універсальних компонентів, можна реалізовувати основні компоненти, що будуть відповідати за відображення якогось конкретних даних. Такі компоненти не можуть бути використані ще десь, адже вони виконують чітку поставлену задачу.

Створимо новий модуль для сутності користувача та реалізуємо компонент, що буде виводити круг з фотографією користувача або універсальною фотографією, якщо користувач не зареєстрований в системі. При наведенні курсору на цей круг має появитись випадаючий список або з привітанням для зареєстрованого користувача або з посиланням на вхід чи реєстрацію для гостя. Приклад реалізації такого компонента зображено на рисунку 3.15.

```
<template>
  <div class="dropdown">
    <template v-if="!getToken">
       <img src="@/assets/images/user.svg" alt="logo" width="52" height="55">
    </template>
    <template v-else>
      <img :src="getImagePath()" alt="logo" width="52" height="55">
    </template>
    <dropdown>
      <template v-if="!getToken">
        <custom-link title="Вхід" to="/login"></custom-link>
        <custom-link to="/registration" title="Реєстрація"></custom-link>
      </template>
      <template v-else>
        <div class="profile">
        <h1>Вітаємо,<br> {{getName}}</h1>
        <div class="button-container">
          <submit-button title="Профіль"></submit-button>
        </div>
        <div class="button-container">
          <submit-button @click="logout" title="Вийти"></submit-button>
        </div>
        </div>
      </template>
    </dropdown>
  </div>
</template>
```

Рисунок 3.15 – Компонент аватару користувача

Щоб вивести дані про поточного користувача, необхідно їх отримати з серверної частини. Для цього стануть в нагоді можливості загального сховища, адже по замовчуванню в файлі сховища є методи для доступу даних, для встановлення даних та перелік методів для роботи з зовнішнім АПІ.

Створимо файл сховища та пропишемо відповідні методи для отримання даних про користувача, їх присвоєння та вибірку. Приклад завантаження даних про користувача з серверної частини зображено на рисунку 3.16.

```
    async loadUserData({commit, rootGetters}){
        const token = rootGetters.getUserToken
        const configs = {
            headers: { Authorization: `Bearer ${token}` }
        };
        await axios.get( url: config.hostname+'/api/profile', configs)
            .then((res :AxiosResponse<any> )=>{
                commit('setUserName', res.data.name)
                commit('setUserEmail', res.data.email)
                commit('setUserImage', res.data.image)
            })
    }
},
```

Рисунок 3.16 – Завантаження даних користувача

Для відображення сторінки в цілому, необхідно створити спеціальні компоненти, які використовуватимуть вже готові компоненти, що реалізовують якусь одну функціональну можливість. Об’єднання таких компонентів надасть повний експерт необхідних можливостей на конкретній сторінці. Для прикладу створимо за допомогою конструктора форми форму входу користувача в систему. Приклад реалізації такого компоненту зображено на рисунку 3.17.

```
<template>
  <div class="content">
    <form-general
        path="images/register-form.png"
        submitTitle="Вхід"
        @submitForm="loginAction"
    >
      <input-text v-model="email" input-type="email" label="Електронна пошта"></input-text>
      <input-text v-model="password" input-type="password" label="Пароль"></input-text>
    </form-general>
  </div>
</template>
```

Рисунок 3.17 – Реалізація форми входу в систему

Для доступу до написаної сторінки за допомогою браузеру її необхідно задекларувати для модуля маршрутизації в окремому файлі. Кожна декларація містить в собі назву, компонент, що буде відображено, шлях до базової розмітки сторінки і назву шляху сторінки в пошуковій стрічці браузера. Приклад декларації маршруту сторінки зображено на рисунку 3.18.

```
{
    path: '/login',
    name: 'Login',
    component: Login,
    meta:{
        layout: "main"
    }
},
```

Рисунок 3.18 – Декларація маршруту сторінки

Такий поступовий підхід до розробки інтерфейсної частини дозволяє пришвидшити терміни його розробки та не плутатись в коді в майбутньому, коли його стане надто багато. Після реалізації описаних вище дій скомпілюємо проект та перейдемо на відповідну сторінку в браузері. Сторінка входу в систему зображена на рисунку 3.19.

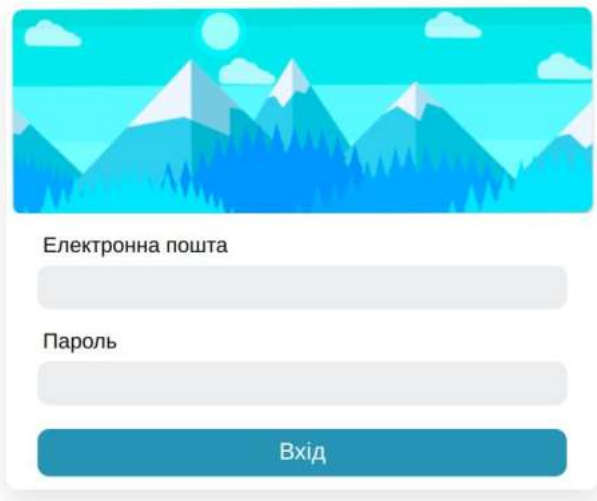


Рисунок 3.19 – Форма входу в систему

Отже, в цьому розділі було реалізовано серверну та інтерфейсну частини. Серверна реалізація дозволяє використання АПІ всіх охочим за допомогою графічного відображення всіх запитів, інтерфейсна частина дозволяє використання можливостей сервера рядовими користувачами.

3.4 Тестування проекту

Тестування проекту є важливою частиною в процесі його розробки. Саме завдяки тестуванню можна зрозуміти чи готовий програмний продукт для користування ним рядовими користувачами.

Під час тестування має перевірятись всі функціональні можливості. Тестування слід проводити по мірі наростання. На приклад спочатку слід почати з серверної частини та модульного тестування.

Оскільки модуль являє собою набір запитів, згрупованих згідно кожної сутності і більше ніяких функціональних можливостей в ньому немає, то слід тестувати запити, один за один по кожній групі. Причому вхідні параметри слід підставляти найрізноманітніші, щоб перевірити працездатність коду в нештатних ситуаціях та виявити логічні помилки.

До таких ситуацій можна віднести або недостачу даних в тілі запиту, або відсутність доступу користувача до запитаних даних. Також важливо перевірити безпеку. Чи може третя особа отримати дані, до яких не мала б мати доступ. Тому слід провести тестування на вразливість до відомих методів отримання небажаного доступу до системи.

Почнемо з тестування запиту на вхід в систему [36]. Цей запит не потребує токену авторизації та очікує електрону пошту і пароль як вхідні параметри. Дані мають передаватись захищеним чином використовуючи протокол HTTPS.

Для цього відкриємо спеціальну сторінку з графічним інтерфейсом запитів та знайдемо необхідний запит в групі Users. Можна також використати спеціальну програму Postman [37]. Дана сторінка зображена на рисунку 3.20.

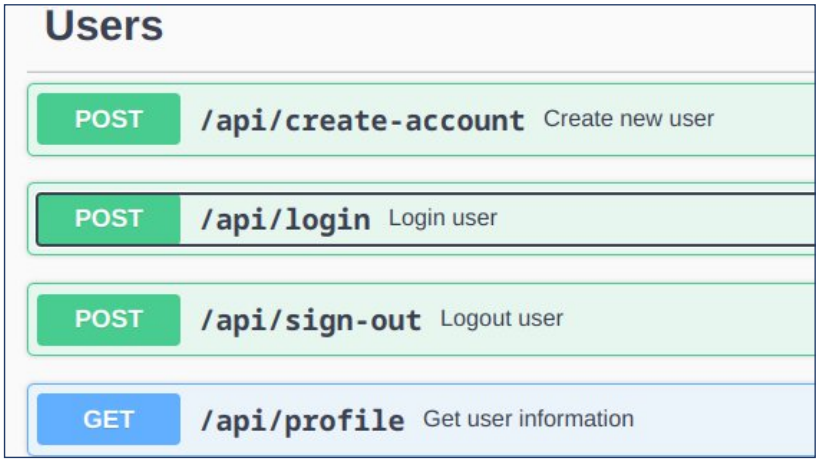


Рисунок 3.20 – Список запитів

Оберемо запит з назвою login та натиснемо на нього. Перед нами буде зображено інформацію про запит, включаючи вхідні параметри та різні варіанти відповіді сервера. Приклад інформації про запит зображено на рисунку 3.21.

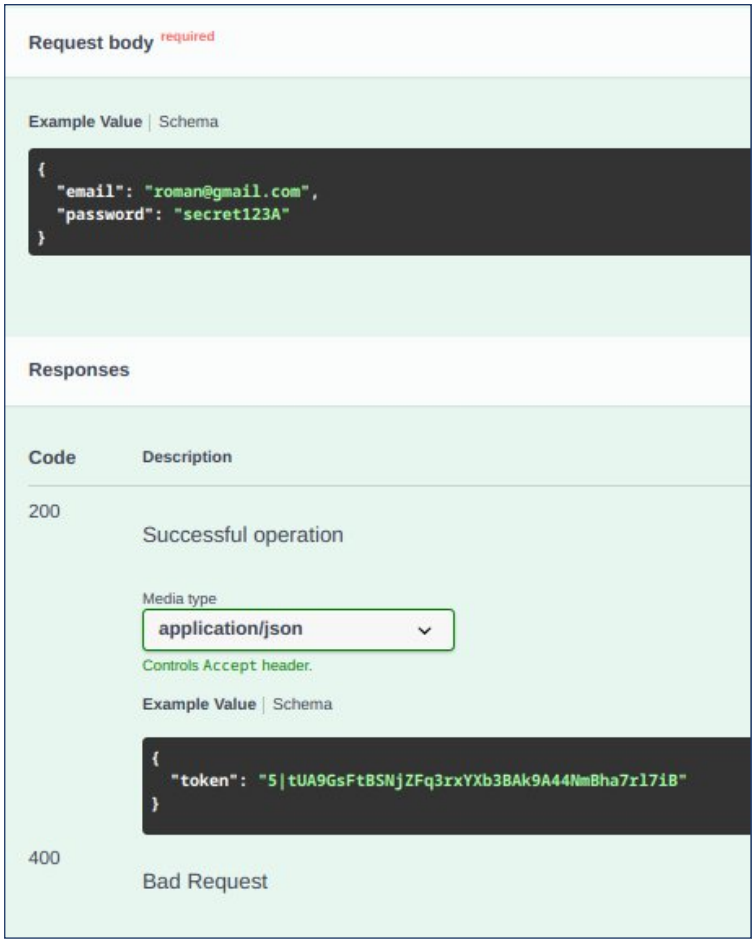


Рисунок 3.21 – Інформація про запит

Як бачимо на рисунку 3.21 зображено як вхідні параметри з їх прикладами та описом, там і вихідні з прикладом вихідних даних. Це дуже сильно полегшує використання серверної частини іншими розробниками.

Натиснемо кнопку та виконаємо запит. В результаті отримаємо ідентифікатор доступу як на рисунку 3.21. Така відповідь сервера можлива тільки коли правильні всі введені дані і такий користувач дійсно існує. Тепер введемо не коректні дані і перевіримо роботу запиту. Сервер повернув помилку з текстом що дані для входу в систему не коректні та код помилки 401. Тож можна зробити висновок що даний запит працює коректно. Приклад відповіді серверу зображено на рисунку 3.22.

Рисунок 3.22 – Відповідь сервера

Тепер протестуємо запит отримання даних про групу, яка належить користувачу, або він входить в ню як учасник. Щоб виконати авторизований запит в графічному інтерфейсі серверу, необхідно вставити отриманий ідентифікатор доступу в текстове поле, що з’явиться після натиснення кнопки входу в верхній частині. Приклад збереження ідентифікатора доступу зображено на рисунку 3.23.

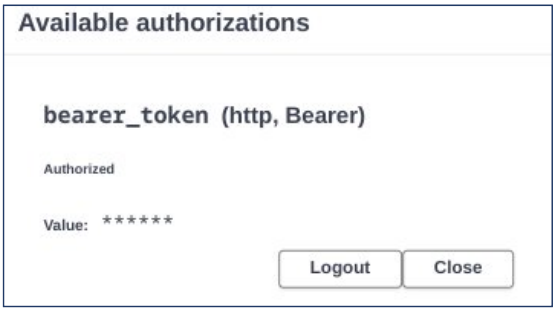


Рисунок 3.23 – Збереження ідентифікатора доступу

Після збереження ідентифікатора доступу оберемо групу запитів з назвою Groups та запит, що відповідає за отримання даних про конкретну групу. Натиснемо кнопку для виконання запиту, попередньо ввівши ідентифікатор потрібної групи. Бачимо, що сервер повернув необхідні дані, адже група з номером 7 була створена користувачем і він має до неї доступ. Приклад відповіді серверу на запит витягу даних про групу зображено на рисунку 3.24.

В такій реалізації є один великий мінус. Сервер завжди поверне дані про групу за її номером, не перевіряючи чи має користувач до неї доступ. Це можна віднести до серйозної безпекової помилки, яку слід усунути в логіці роботи серверу.

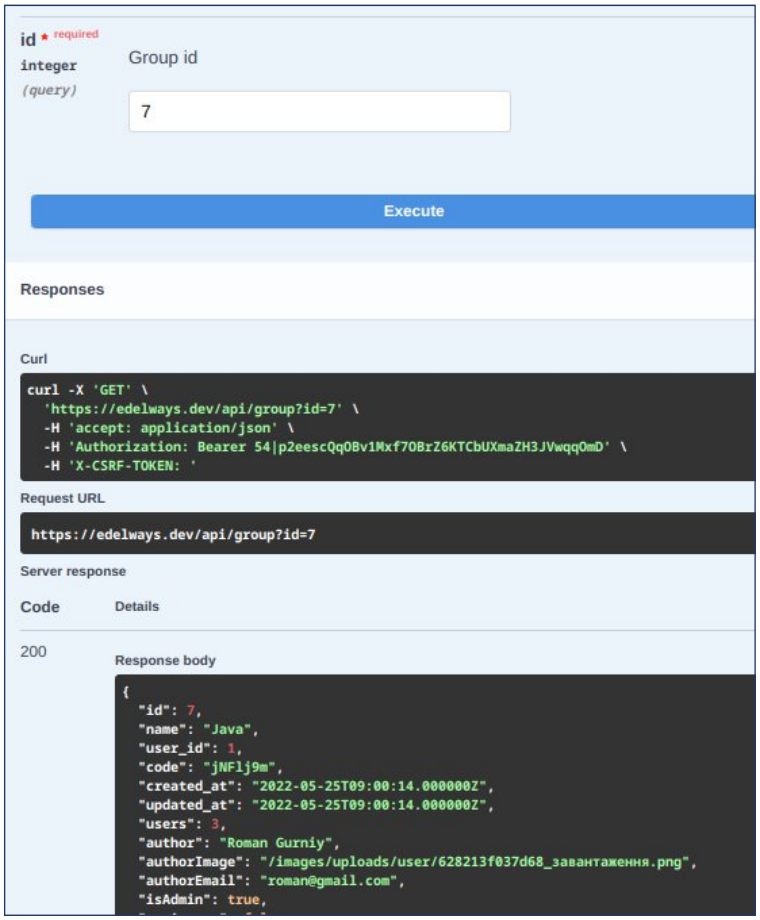


Рисунок 3.24 – Отримання даних про групу

Для вирішення описано проблеми слід додати додаткову перевірку в сервісі групи, який завантажує дані про групу. Якщо користувач не створював цю групу

та не є її учасником, то слід повернути помилку з відповідним текстом. Приклад відповіді серверу з помилкою про відсутність доступу до групи зображено на рисунку 3.25.

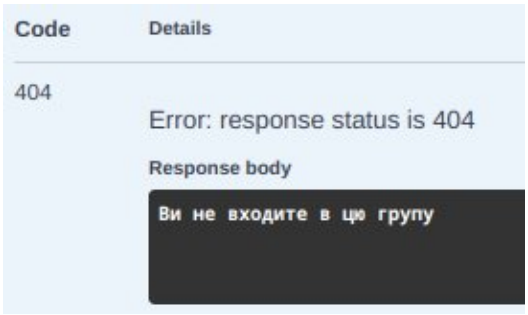


Рисунок 3.25 – Помилка про відсутність доступу

Наступним етапом буде інтеграційне тестування щоб перевірити коректність взаємодії інтерфейсної та серверної частин та знайти всі логічні помилки, що були упущені в процесі модульного тестування.

Спробуємо зареєструватись, ввійти в систему за допомогою графічного інтерфейсу [38]. Тепер створимо нову групу і перейдемо до неї. Тут поки що немає ні активних завдань, ні теорії, але пусте вікно виглядає моторошно. Приклад відображення групи зображено на рисунку 3.26.

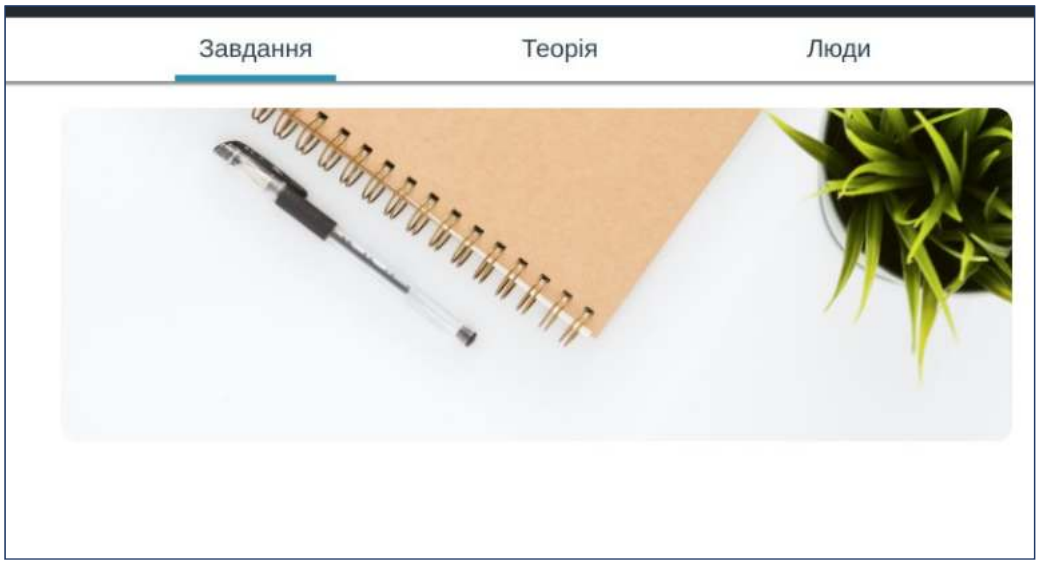


Рисунок 3.26 – Вигляд сторінки групи без вмісту

Пусте вікно не можна віднести до логічної помилки в роботі, адже дійсно в цій групі ще не створено нових завдань. Проте, користувач не одразу може це зрозуміти без відповідного надпису, тому це можна віднести до помилки в проектуванні інтерфейсної частини, адже такий функціонал не був продуманий заздалегідь.

Виправимо цю помилку та додамо відповідний надпис якщо до групи не додано жодного завдання, щоб користувач розумів, що все насправді працює коректно. Приклад вигляду сторінки відображення групи після виправлення помилки зображено на рисунку 3.27.

В загальному в реалізованому інтерфейсі можна знайти багато подібних помилок, що можуть збивати користувача та заважати комфортній роботі, тому більшість з них має бути усунена перед випуском програмного продукту.

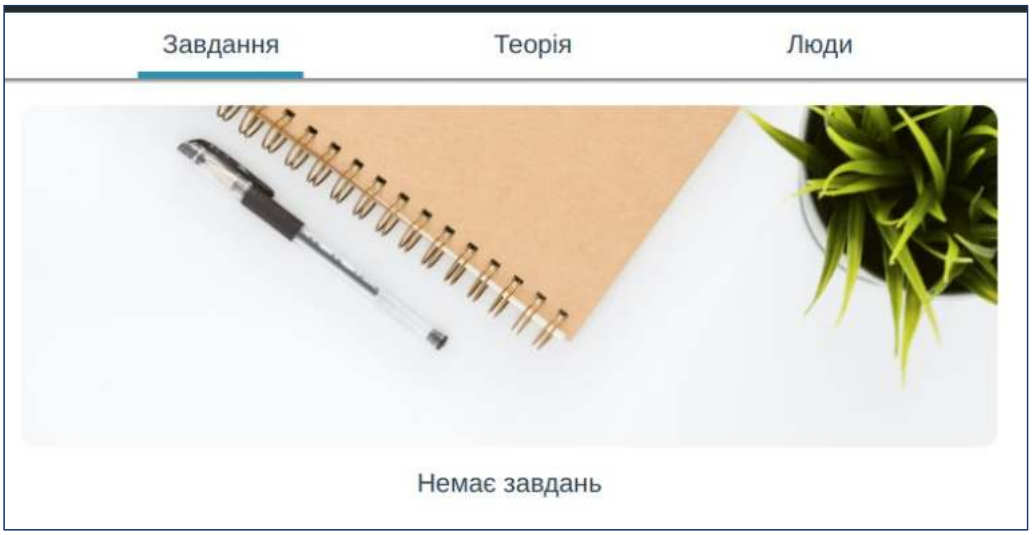


Рисунок 3.27 – Вигляд сторінки групи після виправлень

Також слід перевірити як буде поводити себе інтерфейсна частина якщо не зможе підключитись до серверу, або сервер поверне помилку. Якщо сервер поверне помилку її необхідно показати користувачу, щоб він розумів чому додаток не працює. Для цього було використано спеціальне доповнення. Приклад відображення помилки з серверу показано на рисунку 3.28.

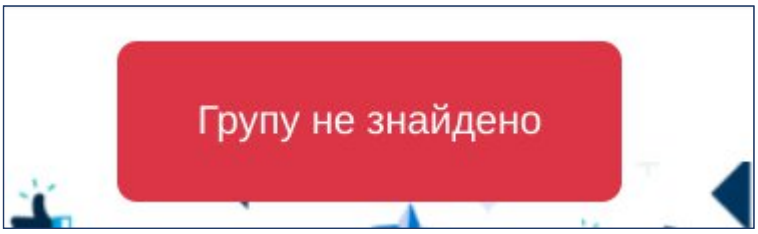


Рисунок 3.28 – Приклад відображення помилки

Отже, у цьому розділі було проведено ряд тестувань та виявлено помилки що стосувались логіки роботи програми, безпеки та в користуванні інтерфейсом. Всі вони були виявленні за допомогою модульного та в подальшому інтеграційного тестування. Після документування більшості помилок, вони буди успішно виправлені та протестовані ще раз.

3.5 Встановлення та розгортання застосунку

Для запуску проекту рядовим користувачам достатньо відкрити браузер та перейти на відповідну адресу, де буде розміщено інтерфейс програми.

Для інсталяції та запуску інтерфейсної частини, необхідно завантажити її вихідний код, встановити менеджер пакетів npm та виконати команди на рисунку 3.29 одна за одною.

```
npm install
npm run build
npm run serve
```

Рисунок 3.29 – Команди для встановлення інтерфейсу

Після запуску цих команд інтерфейсна частина буде доступна в браузері за портом 8080.

Для інсталяції та запуску серверної частини необхідно встановити MySQL, PHP, Nginx утиліти та налаштувати локальний сервер. Після чого необхідно

встановити менеджер пакетів Composer [39] та отримати код серверної частини. Наступним кроком буде виконання команд по черзі на рисунку 3.30.

```
composer self-update --2
composer install
php artisan serve
php artisan migrate
php artisan key:generate
```

Рисунок 3.30 – Команди для встановлення серверу

Після виконання цих команд графічна оболонка серверної частини буде доступна в браузері за портом 443. Фінальним кроком буде вказання порту серверної частини в файлі конфігурації інтерфейсної для їх зв’язку.

Отже, в цьому розділі був описаний процес інсталяції програмного продукту як для рядових користувачів, так і для майбутніх розробників чи тестувальників, яким необхідно мати локальну копію наявного проекту для розробки нового функціоналу та виправлення наявних в проекті помилок.

3.6 Інструкція з використання для користувача

Щоб отримати від програмного продукту максимальний результат, який він може надати, необхідно знати як правильно ним користуватись і які можливості в нього є.

Щоб почати використання програми, спочатку необхідно створити обліковий запис за допомогою реєстрації. Для цього наведіть курсор миші на іконку користувача в верхній лівій частині та натисніть на надпис «Реєстрація». Після чого, заповніть форму особистими даними. Приклад заповнення форми реєстрації зображено на рисунку 3.31.

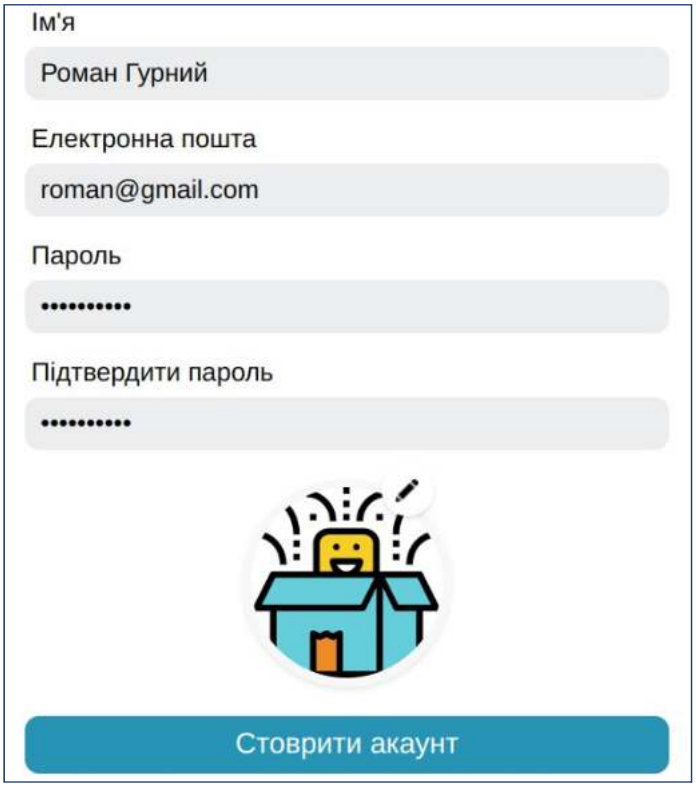


Рисунок 3.31 – Створення особистого кабінету

Наступним кроком буде приєднання до групи. Для цього отримайте код групи в її власника чи адміністратора. Потім натисніть на іконку меню в лівій верхній частині. Перед вами відкриється бокове меню. Натисніть на кнопку приєднатись та введіть код групи в діалогове вікно як на рисунку 3.32. Після приєднання група появиться в списку на головному екрані. Щоб переглянути вміст групи натисніть на кнопку «перегляд» для відповідної групи.

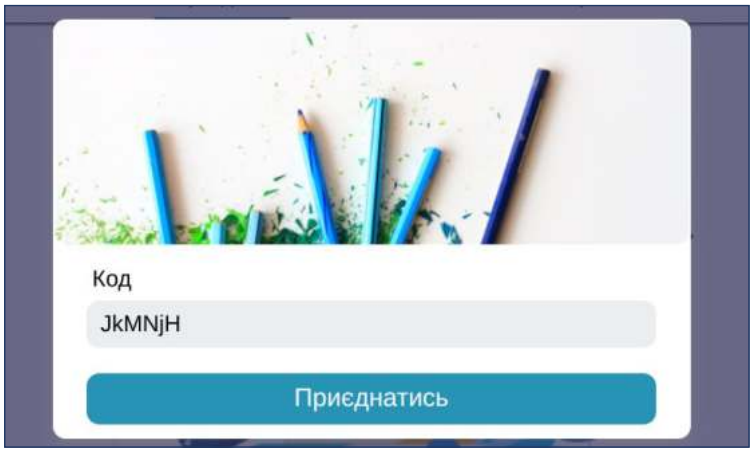


Рисунок 3.32 – Діалогове вікно

На сторінці перегляду групи доступні три вкладки: завдання, теорія та люди. Кожна з вкладок відповідає за відображення власного змісту. Щоб перейти на іншу вкладку необхідно клацнути на ню. Дизайн вкладок зображено на рисунку 3.33.

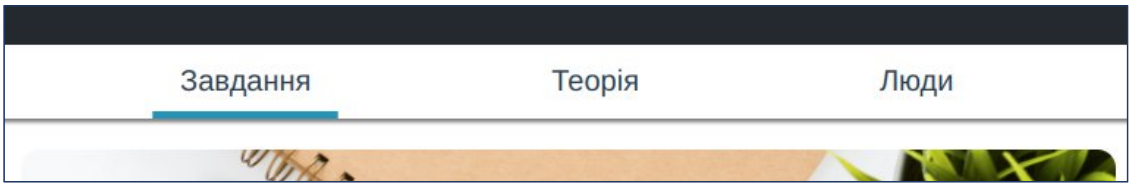


Рисунок 3.33 – дизайн вкладок

В залежності від обраної вкладки можна перейти на сторінку необхідної сутності. Наприклад якщо натиснути на вкладку «Теорія», то з’явиться весь список теоретичних матеріалів, зображених на рисунку 3.34

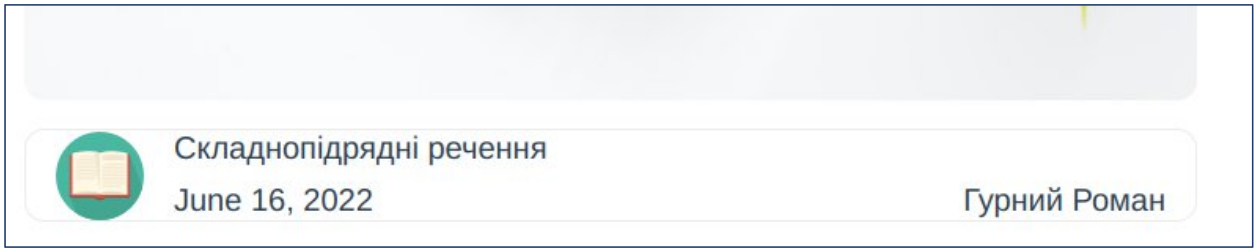


Рисунок 3.34 – Список теорії

Для того щоб ознайомитись із вмістом теоретичного матеріалу, необхідно клацнути по цьому елементу списку, тоді користувача буде переміщено на сторінку перегляду теорії, зображену на рисунку 3.35.

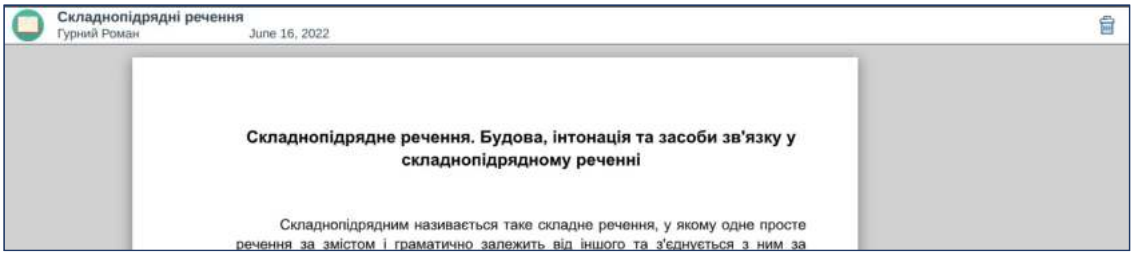


Рисунок 3.35 – Сторінка перегляду теорії

Для того, щоб ознайомитись із тестом завдання та здати роботу необхідно перейти на відповідне завдання, клацнувши по ньому. Після чого користувача буде переадресовано на сторінку перегляду завдання, зображену на рисунку 3.36.

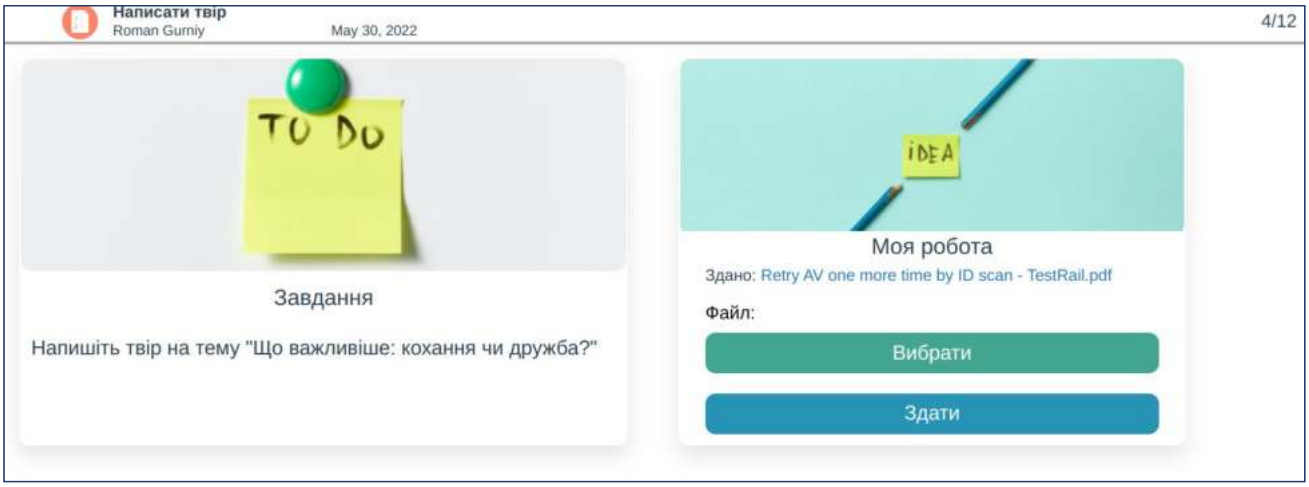


Рисунок 3.36 – Сторінка перегляду завдання

Для здачі роботи необхідно натиснути на кнопку «Вибрати» та завантажити необхідний файл. Якщо робота є перевіреною, то в правому верхньому куту є оцінка. Якщо необхідно здати роботу ще раз, то знову натискаємо на відповідну кнопку та завантажуємо файл. Щоб не втратити даних, всі зміни документуються в історії, що доступна на тій ж сторінці і зображена на рисунку 3.37.

Документування не дозволить маніпулювати оцінками і завжди можна буде дізнати скільки разів перевірялась робота, який результат кожної з перевірок та який коментар залишив перевіряючий для кожної з робіт.

Результат виконання роботи також відображається на головній сторінці групи навпроти кожного з завдань, якщо воно є перевіреним.

Такий підхід дозволяє декілька разів здавати роботу на перевірку і кожного разу отримувати коментар від викладача. Також студент може залишати власні коментарі, що допоможе викладачу краще зрозуміти результат виконаної роботи студента.

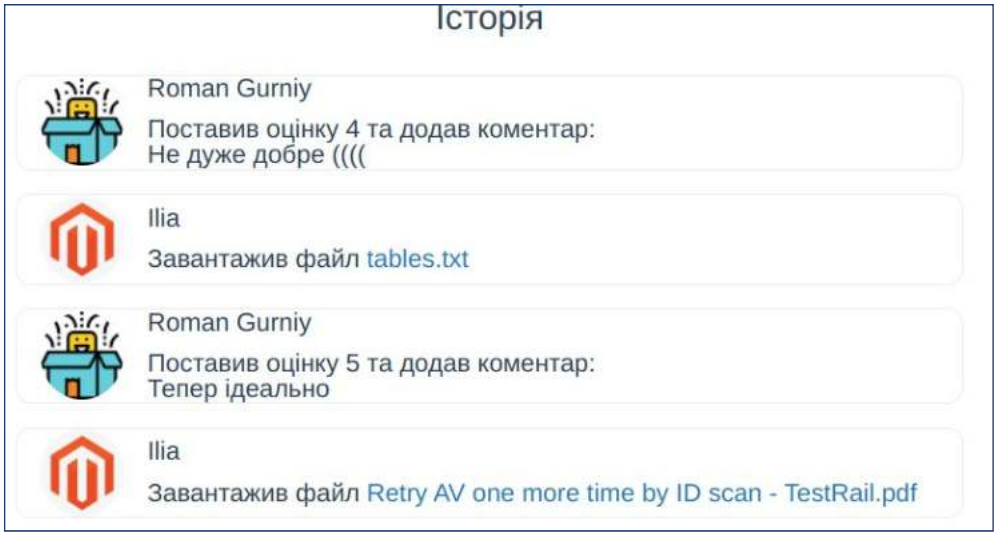


Рисунок 3.37 – Історія виконання завдання

Отже, в цьому розділі було описано інструкцію для використання програмного продукту рядовим користувачем. Процес описано від самого початку реєстрації до перегляду результатів зданої роботи.

3.7 Інструкція з використання адміністратора

Якщо користувач створив власну групу, то цей розділ міститиме інструкцію як правильного адміністрування групи.

Адміністратор точно так само як звичайний користувач може переглянути вміст групи, переглядати списки теорії та завдань, а також теоретичний матеріал. Основної відмінністю є те, що адміністратор може повністю видалити групу натиснувши на відповідну іконку, зображену на рисунку 3.38.

Рисунок 3.38 – Видалення групи

Якщо користувач захоче видалити групу, то перед ним з’явиться відповідне попереджувальне вікно, де йому необхідно буде підтвердити дію. Окрім видалення групи, адміністратор може також видаляти теоретичний матеріал та завдання, а також від’єднувати людей з групи.

Адміністратор окрім видалення існуючого вмісту може додавати новий. Для цього необхідно натиснути на три крапки на сторінці перегляду групи в лівій верхній частин. Після чого з’явиться випадаючий список, зображений на рисунку 3.39.

Рисунок 3.39 – Список можливостей адміністратора

Основною роботою адміністратора є перевірка зданих робіт. Для цього необхідно перейти на сторінку із списком завдань та обрати якесь завдання. Після чого адміністратору відобразиться список людей, які здали це завдання. Також можна побачити які роботи вже були перевірені і коли ці роботи були здані. Список зданих робіт зображено на рисунку 3.40.

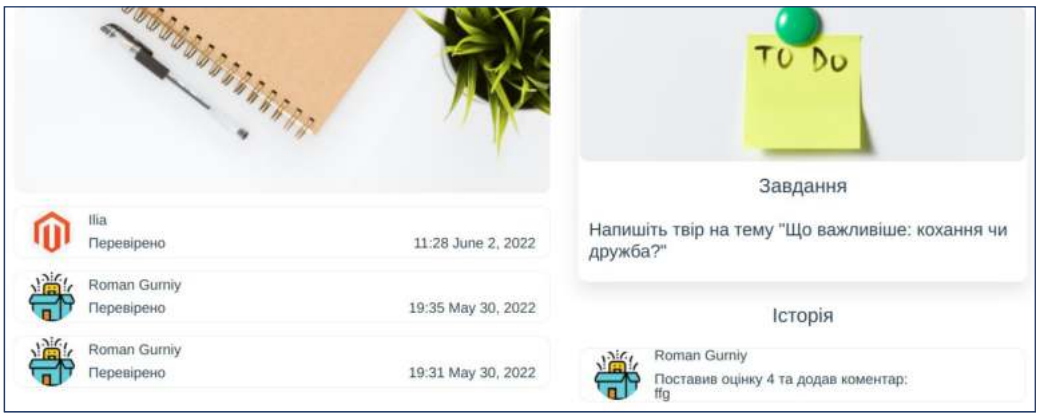


Рисунок 3.40 – Список зданих робіт

Адміністратору також доступна історія всіх змін, що стосуються цього завдання. Вона включає в себе зміни всіх користувачів, що здавали це завдання.

Щоб перевірити завдання чи переглянути перевірене, необхідно клацнути по користувачу, якого необхідно перевірити, після чого адміністратора буде переадресовано на форму перевірки завдання, зображену на рисунку 3.41.

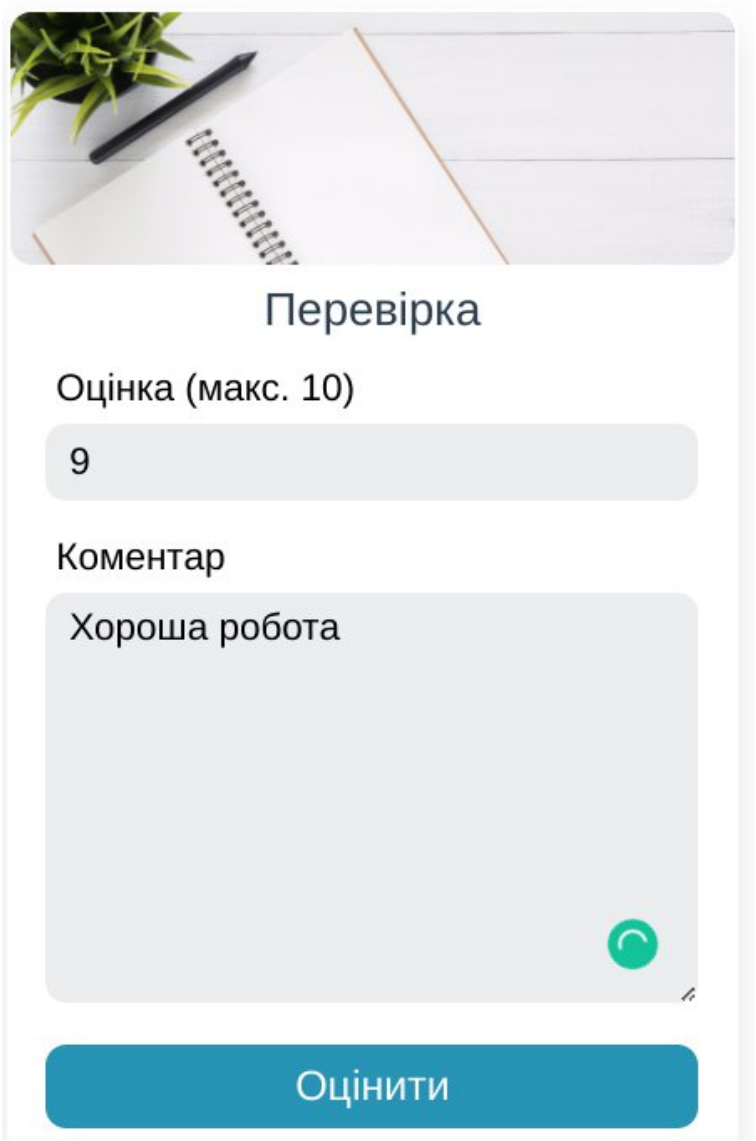


Рисунок 3.41 – Форма перевірки завдання

Отже, в цьому розділі була описана інструкція для правильного адміністрування групою. В інструкції були описані всі можливості та права адміністратора, а також описано як ними можна скористатись.

3.5 Висновки програмної реалізації та тестування програмного забезпечення

В цьому розділі було реалізовано програмну реалізацію серверної та інтерфейсної частини проекту згідно з вимогами проектування.

Серверна частина реалізована за допомогою фреймворку Laravel та засобом автоматичної генерації документації API Swagger.

Інтерфейсна частина реалізована за допомогою сучасного фреймворку VueJs 3 та використанням бібліотек для спільного сховища VueX та реалізації маршрутизації за допомогою VueRoute.

Після реалізації обох частин вони були розміщені та налаштовані на хостингу Heroku. Це дозволило мати глобальний доступ до вебзастосунку. Також було налаштовано інтеграцію з репозиторієм коду та встановлено автоматичне розгортання після внесення змін.

Фінальним етап стало тестування обох частин системи. Сервер був протестований на відповідність функціональних вимог та безпеки. В ході тестування були знайдені та виправлені певні логічні помилки.

Під час тестування інтерфейсної частини було знайдено помилки в графічній реалізації певних компонентів, що ускладнювало використання продукту. Ці помилки були виправлені та задокументовані.

Також було реалізовано покрокову інструкцію для рядового користувача та адміністратора. В інструкції висвітлені основні можливості проекту для користувача певної ролі.

ВИСНОВКИ

Розробка дипломного проєкту складалася з реалізації серверної та інтерфейсної частин як двох окремих проєктів. Для серверної частини був використаний фреймворк Laravel та мова програмування PHP, для інтерфейсної частини – фреймворк VueJS.

Дипломний проєкт розроблявся поетапно. Спочатку було сформоване технічне завдання та всі вимоги, потім проведено аналіз вибору необхідних засобів розробки. Після цього майбутній програмний продукт був детально спроєктований за допомогою діаграм. Основним етапом була його реалізація, а фінальним – тестування.

Такий підхід дозволив використання функціональних можливостей сервера всім охочим у вигляді відкритого АПІ з графічним інтерфейсом.

Програмний продукт виконує поставлене перед ним технічне завдання та відповідає його проєктній моделі. Завдяки розбиттю серверної та інтерфейсної частин підтримка та розробка нового функціоналу буде максимально простою.

Переваги даного проєкту:

– наявність відкритого АПІ з графічним інтерфейсом;

– відкритий код обох частин програмного продукту;

– використання сучасних технологій та засобів розробки;

– продукт створений для реалізації дистанційного навчання;

– продуманий та спроєктований інтерфейс.

Недоліки програмного продукту:

– наявність некритичних помилок;

– повільніша робота програми порівняно з монолітною архітектурою;

– надає меншу кількість функціоналу.

Інтерфейс програмного продукту був розроблений з підтримкою різних типів дисплеїв. В економічному розділі було проведено аналіз доцільності та вартості розробки даного програмного забезпечення, також визначено основну аудиторію та сферу його застосування.

У ході виконання проєкту було здобуто цінний практичний досвід у використанні сучасних підходів до проєктування та розробки вебзастосунків. Розділення на інтерфейсну та бекенд частини дало змогу побудувати гнучку та масштабовану архітектуру, що легко адаптується до майбутніх змін або інтеграцій.

Особливу увагу було приділено безпеці, зручності користувача та адаптивності інтерфейсу, що є критично важливим у контексті дистанційного навчання.

Завдяки використаним технологіям та структурованому підходу до розробки, проєкт можна легко масштабувати, додавати нові модулі, інтегрувати сторонні сервіси або адаптувати під інші освітні потреби.

У перспективі, програмний продукт має потенціал до розвитку у повноцінну платформу для дистанційного навчання з розширеним функціоналом, що включає тестування, ведення електронних журналів, аналітику успішності та інше.

Таким чином, дипломний проєкт не лише продемонстрував набуті знання та навички, а й має реальне прикладне значення та перспективу подальшого розвитку.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Поняття фреймворк. URL: https://brainlab.com.ua/uk/blog-uk/shho-take-frejmvork-poyasnyuyemo-prostymy-slovamy (дата звернення 10.02.2025).

2. Основи HTML-розмітки. URL: https://www.css.in.ua/ (дата звернення: 22.05.2025).

3. «Безголова» архітектура. URL: https://www.accenture.com/nl-en/blogs/insights/headless-architecture-why-its-becoming-the-new-normal (дата звернення: 12.02.2025).

4. Адаптивне верстання. URL: https://sendpulse.ua/blog/responsive-and-adaptive-design/ (дата звернення: 20.02.2025).

5. Сервіс для менеджменту команд Microsoft Teams. URL: https://teams.microsoft.com/ (дата звернення: 15.04.2025).

6. Сервіс для онлайн навчання Google Classroom. URL: https://classroom.google.com/ (дата звернення: 16.04.2025).

7. Сервіс для хостингу Heroku. URL: https://www.heroku.com/ (дата звернення: 22.04.2025).

8. Основи HTML-розмітки. URL: https://www.css.in.ua/ (дата звернення: 22.05.2025).

9. Модульне тестування. URL: https://qalight.ua/baza-znaniy/modulne-testuvannya/ (дата звернення: 22.04.2025).

10. Інтеграційне тестування. URL: https://qalight.ua/baza-znaniy/integratsijne-testuvannya/ (дата звернення: 25.04.2025).

11. Системне тестування. URL: https://qalight.ua/baza-znaniy/sistemne-testuvannya/ (дата звернення: 28.04.2025).

12. Архітектура мікросервісів. URL: https://microservices.io/ (дата звернення: 20.05.2025).

13. Діаграма потоків даних. URL: https://www.maxzosim.com/data-flow-diagrams/ (дата звернення: 28.04.2025).

14. Діаграма класів. URL: https://www.mindonmap.com/uk/blog/what-is-uml-

class-diagram/ (дата звернення: 02.05.2025).

15. Модульна архітектура. URL: http://uk.homagic.com/blog/what-is-modular-architecture/ (дата звернення: 02.05.2025).

16. What is a REST API. URL: https://www.ibm.com/topics/rest-apis/ (дата звернення: 03.05.2025).

17. Основи компонентів VueJs. URL: https://ua.vuejs.org/guide/essentials/component-basics.html (дата звернення: 04.05.2025).

18. Діаграма розгортання. URL: https://www.guru99.com/uk/deployment-diagram-uml-example.html/ (дата звернення: 05.05.2025).

19. СУБД MySQL. URL: https://www.mysql.com/ (дата звернення: 07.05.2025).

20. Системи управління базами даних. URL: https://hub.kyivstar.ua/articles/sistemi-upravlinnya-bazami-danih-dlya-malogo-ta-serednogo-biznesu-yak-vibrati (дата звернення: 07.05.2025).

21. Основи проектування баз даних. URL: https://ela.kpi.ua/items/fdbe4913-738c-47b3-a9ce-bca8b63b333c (дата звернення: 10.05.2025).

22. Сервіс для створення дизайну Figma. URL: https://www.figma.com/ (дата звернення: 10.05.2025).

23. Фреймворк Spring. URL: https://spring.io/ (дата звернення: 05.03.2025).

24. Мова програмування Python. URL: https://www.python.org/ (дата звернення: 22.03.2025).

25. Фреймворк Jango. URL: https://www.jango.com/ (дата звернення: 22.03.2025).

26. Мова програмування PHP. URL: https://www.php.net/manual/ru/intro-whatis.php (дата звернення: 22.03.2025).

27. Фреймворк Laravel. URL: https://laravel.com/ (дата звернення: 23.03.2025).

28. Фреймворк Simfony. URL: https://symfony.com/ (дата звернення: 23.03.2025).

29. Бібліотека React. URL: https://ua.reactjs.org/ (дата звернення: 04.04.2025).

30. Поняття MVC. URL: https://highload.tech/uk/blogs/shho-take-mvc-ta-mvp-patterni/ (дата звернення: 05.04.2025).

31. Фреймворк Angular. URL: https://angular.io/ (дата звернення: 05.04.2025).

32. Фреймворк VueJS. URL: https://vuejs.org/ (дата звернення: 08.04.2025).

33. Основи CSS-стилів. URL: https://www.css.in.ua/ (дата звернення: 23.05.2025)

34. Сервіс віртуалізації Docker. URL: https://www.docker.com/ (дата звернення: 11.05.2025).

35. Сервіс для графічного оформлення АПІ Swagger URL: https://swagger.io/ (дата звернення: 12.05.2025).

36. Тестування з PHPUnit. URL: https://phpunit.de/ (дата звернення: 17.05.2025).

37. API-розробка з Postman. URL: https://learning.postman.com/ (дата звернення: 17.05.2025).

38. UX/UI Принципи. URL: https://www.nngroup.com/articles/definition-user-experience/ (дата звернення: 20.05.2025).

39. Пакетний менеджер Composer URL: https://getcomposer.org/ (дата звернення: 14.05.2025).

40. Система контролю версій Git. URL: https://git-scm.com/ (дата звернення: 21.05.2025).

ДОДАТОК А

(обов’язковий)

КОД (ЛІСТИНГ) ПРОГРАМИ

User.php

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Foundation\Auth\User as Authenticatable;
use Illuminate\Notifications\Notifiable;
use Laravel\Sanctum\HasApiTokens;

/**
 * @property $id
 */
class User extends Authenticatable
{
    use HasApiTokens, HasFactory, Notifiable;

    /**
     * The attributes that are mass assignable.
     *
     * @var array<int, string>
     */
    protected $fillable = [
        'name',
        'email',
        'password',
        'image'
    ];

    /**
     * The attributes that should be hidden for serialization.
     *
     * @var array<int, string>
     */
    protected $hidden = [
        'password',
        'remember_token',
    ];

    /**
     * The attributes that should be cast.
     *
     * @var array<string, string>
     */
    protected $casts = [
        'email_verified_at' => 'datetime',
    ];
}
```

Group.php

```php
<?php

namespace App\Models;

use Exception;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

/**
 * @property $name
 * @property $code
 * @property $user_id
 * @property $id
 * @property $canAccess
 * @property $isAdmin
 */
class Group extends Model
{
    use HasFactory;

    const ID = 'id';
    const NAME = 'name';
    const CODE = 'code';
    const USER_ID = 'user_id';

    protected $table = 'groups';
    protected $primaryKey = 'id';
    public $timestamps = true;

    protected $fillable = [
        self::NAME,
        self::CODE,
        self::USER_ID
    ];

    protected $hidden = [
        'user'
    ];

    protected $appends = [
        'users',
        'author',
        'authorImage',
        'authorEmail',
        'isAdmin',
        'canAccess',
        'userList',
        'date'
    ];

    public function user()
    {
        return $this->belongsTo(User::class);
    }

    public function getUsersAttribute(): int
    {
        return GroupUsers::query()
            ->where(GroupUsers::GROUP_ID, '=', $this->id)
            ->count() + 1;
    }

    public function getAuthorAttribute()
    {
        return $this->user->name;
    }

    public function getAuthorEmailAttribute()
    {
        return $this->user->email;
    }

    public function getAuthorImageAttribute()
    {
        return $this->user->image;
    }

    public function getIsAdminAttribute(): bool
    {
        $isAdmin = false;
        if($this->user_id === auth()->user()->id){
            $isAdmin = true;
        }
        return $isAdmin;
    }

    public function getDateAttribute()
    {
        return $this->updated_at->format('H:i F j, Y');
    }

    public function getCanAccessAttribute(): bool
    {
        $canAccess = false;
        $groupUser = GroupUsers::query()
```

```
            ->where(GroupUsers::GROUP_ID,'=',$this->id)
            ->where(GroupUsers::USER_ID, '=', auth()->user()->id)
            ->first();
        if($groupUser){
            $canAccess = true;
        }
        return $canAccess;
    }

    public function getUserListAttribute()
    {
        return GroupUsers::query()
        ->where(GroupUsers::GROUP_ID, '=', $this->id)
        ->get();
    }
}
```

Task.php

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use function Symfony\Component\Translation\t;

/**
 * @property $id
 * @property $user_id
 * @property $group_id
 * @property $name
 * @property $document_path
 * @property $description
 * @property $mark
 */
class Task extends Model
{
    use HasFactory;

    const ID = 'id';
    const USER_ID = 'user_id';
    const GROUP_ID = 'group_id';
    const NAME = 'name';
    const DESCRIPTION = 'description';
    const DOCUMENT_PATH = 'document_path';
    const MARK = 'mark';

    protected $fillable = [
        self::GROUP_ID,
        self::USER_ID,
        self::DOCUMENT_PATH,
        self::NAME,
        self::DESCRIPTION,
        self::MARK
    ];

    protected $appends = [
        'author',
        'date',
        'isAdmin',
        'complete',
        'comments'
    ];

    protected $hidden = [
        'user',
        'group',
        'taskCompletes'
    ];

    public function user()
    {
        return $this->belongsTo(User::class);
    }

    public function group()
    {
        return $this->belongsTo(Group::class);
    }

    public function completes()
    {
        return $this->hasMany(TaskComplete::class);
    }

    public function getCompleteAttribute()
    {
        return $this->completes()
            ->where(TaskComplete::USER_ID, '=', auth()->id())
            ->orderBy('created_at', 'desc')
            ->first();
    }

    public function getAuthorAttribute()
    {
        return $this->user->name;
    }

    public function getDateAttribute()
    {
        return $this->created_at->format('F j, Y');
    }

    public function getIsAdminAttribute(): bool
    {
        $group = Group::query()->find($this->group_id);
        if($group->isAdmin){
            return true;
        } else {
            return false;
        }
    }

    public function getCommentsAttribute()
    {
        $key = 1;
        if(!$this->complete && !$this->group->isAdmin){
            return [];
        }
        $comments = [];
        if($this->group->isAdmin){
            $completes = $this->completes()
                ->orderBy('created_at', 'desc')
                ->get();
        } else {
            $completes = $this->completes()
                ->where(TaskComplete::USER_ID, '=', auth()->id())
                ->orderBy('created_at', 'desc')
                ->get();
        }
        foreach ($completes as $complete){
            $review = TaskReview::query()->
                where(TaskReview::COMPLETE_ID, '=',
$complete->id)
                ->first();
            if($review){
                $comment = [
                    'type' => 'review',
                    'key' => $key,
                    'data' => $review
                ];
                $comments[] = $comment;
                $key++;
            }
            $comment = [
                'type' => 'complete',
                'key' => $key,
                'data' => $complete,
```

```
            ];
            $comments[] = $comment;
            $key++;
        }
        return $comments;
    }
}
```

AuthController.php

```
<?php

namespace App\Http\Controllers;

use App\Service\AuthService;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;

class AuthController extends Controller
{
    /**
     * @var AuthService
     */
    private $authService;

    public function __construct(AuthService $authService)
    {
        $this->authService = $authService;
    }

    /**
     * @OA\Post(
     *     path="/api/create-account",
     *     operationId="createUser",
     *     tags={"Users"},
     *     summary="Create new user",
     *     description="Returns user data",
     *     @OA\RequestBody(
     *         required=true,
     *
@OA\JsonContent(ref="#/components/schemas/CreateUserRequest")
     *     ),
     *     @OA\Response(
     *         response=201,
     *         description="Successful operation",
     *
@OA\JsonContent(ref="#/components/schemas/User")
     *      ),
     *     @OA\Response(
     *         response=400,
     *         description="Bad Request"
     *     ),
     * )
     */
    public function createAccount(Request $request)
    {
        $data = $request->validate([
            'name' => 'required|string|max:255',
            'email' => 'required|string|email|unique:users,email',
            'password' => 'required|string|min:6|confirmed',
            'image' => 'required|string'
        ]);

        $this->authService->createUser(
            $data['name'],
            $data['email'],
            $data['password'],
            $data['image']);
        return response()->json([
            'token' => $this->authService->generateToken()
        ]);
    }

    /**
     * @OA\Post(
     *     path="/api/login",
     *     operationId="loginUser",
     *     tags={"Users"},
     *     summary="Login user",
     *     description="Returns user tocken",
     *     @OA\RequestBody(
     *         required=true,
     *
@OA\JsonContent(ref="#/components/schemas/LoginUserRequest")
     *     ),
     *     @OA\Response(
     *         response=200,
     *         description="Successful operation",
     *
@OA\JsonContent(ref="#/components/schemas/UserToken")
     *      ),
     *     @OA\Response(
     *         response=400,
     *         description="Bad Request"
     *     ),
     * )
     */
    public function login(Request $request)
    {
        $attr = $request->validate([
            'email' => 'required|string|email|',
            'password' => 'required|string|min:6'
        ]);

        if (!Auth::attempt($attr)) {
            return response()->json('Credentials not match', 401);
        }

        return response()->json([
            'token' => $this->authService->generateToken(),
        ]);
    }

    /**
     * @OA\Post(
     *     path="/api/sign-out",
     *     operationId="logoutUser",
     *     tags={"Users"},
     *     summary="Logout user",
     *     description="Returns success message",
     *     security={{"bearer_token":{}}},
     *   @OA\Response(
     *     response=200,
     *      description="Success",
     *     @OA\MediaType(
     *          mediaType="application/json",
     *     )
     *   ),
     *     @OA\Response(
     *         response=400,
     *         description="Bad Request"
     *     ),
     *     @OA\Response(
     *         response=401,
     *         description="Unauthenticated",
     *     ),
     * )
     */
    public function logout()
    {
        $this->authService->removeToken();

        return response()->json('Success');
    }

    /**
     * @OA\Get(
```

```
     *      path="/api/profile",
     *      operationId="getUserData",
     *      tags={"Users"},
     *      summary="Get user information",
     *      description="Returns user data",
     *      security={{"bearer_token":{}}},
     *      @OA\Response(
     *          response=200,
     *          description="Successful operation",
     *
@OA\JsonContent(ref="#/components/schemas/User")
     *      ),
     *      @OA\Response(
     *          response=400,
     *          description="Bad Request"
     *      ),
     *      @OA\Response(
     *          response=401,
     *          description="Unauthenticated",
     *      ),
     * )
     */
    public function userData()
    {
        return auth()->user();
    }
}
```

GroupController.php

```
<?php

namespace App\Http\Controllers;

use App\Service\GroupService;
use Exception;
use Illuminate\Http\Request;

class GroupController extends Controller
{
    /**
     * @var GroupService
     */
    private $groupService;

    public function __construct(GroupService $groupService)
    {
        $this->groupService = $groupService;
    }

    /**
     * @OA\Post(
     *      path="/api/create-group",
     *      operationId="createGroup",
     *      tags={"Groups"},
     *      summary="Create new group",
     *      description="Returns group data",
     *      security={{"bearer_token":{}}},
     *      @OA\RequestBody(
     *          required=true,
     *
@OA\JsonContent(ref="#/components/schemas/CreateGroup
Request")
     *      ),
     *      @OA\Response(
     *          response=201,
     *          description="Successful operation",
     *
@OA\JsonContent(ref="#/components/schemas/Group")
     *      ),
     *      @OA\Response(
     *          response=400,
     *          description="Bad Request"
     *      ),
     * )
     */
    public function create(Request $request)
    {
        $data = $request->validate([
            'name' => 'required|string|max:255|min:3',
        ]);
        return $this->groupService->create($data['name']);
    }

    /**
     * @OA\Get(
     *      path="/api/groups",
     *      operationId="getGroupsData",
     *      tags={"Groups"},
     *      summary="Get group information",
     *      description="Returns groups data",
     *      security={{"bearer_token":{}}},
     *        @OA\Parameter(
     *          name="filter",
     *          description="Filter all or my groups",
     *          required=true,
     *          in="query",
     *          @OA\Schema(
     *              type="string"
     *          )
     *      ),
     *      @OA\Response(
     *          response=200,
     *          description="Successful operation",
     *
@OA\JsonContent(ref="#/components/schemas/Group")
     *      ),
     *      @OA\Response(
     *          response=400,
     *          description="Bad Request"
     *      ),
     *      @OA\Response(
     *          response=401,
     *          description="Unauthenticated",
     *      ),
     * )
     */
    public function list(Request $request)
    {
        return
$this->grou
pService->get
GroupByFilter($request->get('filter'));
    }

    /**
     * @OA\Post(
     *      path="/api/join-group",
     *      operationId="joinGroup",
     *      tags={"Groups"},
     *      summary="Join to group",
     *      description="Returns success or not",
     *      security={{"bearer_token":{}}},
     *      @OA\RequestBody(
     *          required=true,
     *
@OA\JsonContent(ref="#/components/schemas/JoinGroupRe
quest")
     *      ),
     *      @OA\Response(
     *          response=200,
     *          description="Successful operation",
     *
@OA\JsonContent(ref="#/components/schemas/Group")
     *      ),
     *      @OA\Response(
     *          response=400,
     *          description="Bad Request"
     *      ),
```

```
     *      @OA\Response(
     *          response=404,
     *          description="Group not found"
     *      ),
     *  )
     */
    public function join(Request $request)
    {
        $data = $request->validate([
            'code' => 'required|string|min:7',
        ]);
        try{
            return $this->groupService->joinGroup($data['code']);
        }catch (Exception $ex){
            return response( $ex->getMessage(), 404);
        }
    }

    /**
     * @OA\Post(
     *      path="/api/delete-group",
     *      operationId="deleteGroup",
     *      tags={"Groups"},
     *      summary="Delete group",
     *      description="Returns success",
     *      security={{"bearer_token":{}}},
     *      @OA\RequestBody(
     *          required=true,
     *
@OA\JsonContent(ref="#/components/schemas/DeleteGroup
Request")
     *      ),
     *      @OA\Response(
     *          response=200,
     *          description="Success",
     *      ),
     *      @OA\Response(
     *          response=400,
     *          description="Bad Request"
     *      ),
     *      @OA\Response(
     *          response=404,
     *          description="Group not found"
     *      ),
     *  )
     */
    public function delete(Request $request)
    {
        $data = $request->validate([
            'id' => 'required|integer|min:1',
        ]);
        try{
            $this->groupService->deleteGroup($data['id']);
        } catch (Exception $ex){
            return response( $ex->getMessage(), 404);
        }
        return response('success');
    }

    /**
     * @OA\Post(
     *      path="/api/left-group",
     *      operationId="leftGroup",
     *      tags={"Groups"},
     *      summary="Left group",
     *      description="Returns success",
     *      security={{"bearer_token":{}}},
     *      @OA\RequestBody(
     *          required=true,
     *
@OA\JsonContent(ref="#/components/schemas/DeleteGroup
Request")
     *      ),
     *      @OA\Response(
     *          response=200,
     *          description="Success",
     *      ),
     *      @OA\Response(
     *          response=400,
     *          description="Bad Request"
     *      ),
     *      @OA\Response(
     *          response=404,
     *          description="Group not found"
     *      ),
     *  )
     */
    public function left(Request $request)
    {
        $data = $request->validate([
            'id' => 'required|integer|min:1',
        ]);
        try{
            $this->groupService->leftGroup($data['id']);
        } catch (Exception $ex){
            return response( $ex->getMessage(), 404);
        }
        return response('success');
    }

    /**
     * @OA\Get(
     *      path="/api/group",
     *      operationId="getGroupData",
     *      tags={"Groups"},
     *      summary="Get group information",
     *      description="Returns group data",
     *      security={{"bearer_token":{}}},
     *          @OA\Parameter(
     *              name="id",
     *              description="Group id",
     *              required=true,
     *              in="query",
     *              @OA\Schema(
     *                  type="integer"
     *              )
     *          ),
     *      @OA\Response(
     *          response=200,
     *          description="Successful operation",
     *
@OA\JsonContent(ref="#/components/schemas/Group")
     *       ),
     *      @OA\Response(
     *          response=404,
     *          description="Group not found"
     *      ),
     *      @OA\Response(
     *          response=401,
     *          description="Unauthenticated",
     *      ),
     *  )
     */
    public function get(Request $request)
    {
        $data = $request->validate([
            'id' => 'required|integer|min:1',
        ]);
        try{
            return $this->groupService->get($data['id']);
        } catch (Exception $ex){
            return response( $ex->getMessage(), 404);
        }
    }

    /**
     * @OA\Post(
     *      path="/api/delete-user",
     *      operationId="deleteUser",
     *      tags={"Groups"},
```

```
     *      summary="Delete user from the group",
     *      description="Returns success",
     *      security={{"bearer_token":{}}},
     *      @OA\RequestBody(
     *          required=true,
     *
@OA\JsonContent(ref="#/components/schemas/DeleteUserRequest")
     *      ),
     *      @OA\Response(
     *          response=200,
     *          description="Success",
     *      ),
     *      @OA\Response(
     *          response=400,
     *          description="Bad Request"
     *      ),
     *      @OA\Response(
     *          response=404,
     *          description="User not found in that group"
     *      ),
     * )
     */
    public function deleteUser(Request $request)
    {
        $data = $request->validate([
            'user_id' => 'required|integer|min:1',
            'group_id' => 'required|integer|min:1',
        ]);
        try{

$this->groupService->leftUserFromGroup($data['user_id'],
$data['group_id']);
        } catch (Exception $ex){
            return response( $ex->getMessage(), 404);
        }
        return response('success');
    }
}
```

TaskService.php

```
<?php

namespace App\Service;

use App\Models\Group;
use App\Models\Task;
use App\Models\TaskComplete;
use App\Models\TaskReview;
use Exception;

class TaskService
{
    /**
     * @var GroupService
     */
    private $groupService;

    public function __construct(GroupService $groupService)
    {
        $this->groupService = $groupService;
    }

    /**
     * @throws Exception
     */
    public function getTasksList($groupId)
    {
        $group = Group::query()->find($groupId);
        $this->groupService->validate($group);
        return Task::query()
            ->where(Task::GROUP_ID, '=', $groupId)
            ->get();
    }

    /**
     * @throws Exception
     */
    public function createTask($name, $groupId, $description,
$documentPath, $mark)
    {
        $userId = auth()->user()->id;
        $group = Group::query()->find($groupId);
        $this->groupService->validate($group);
        return Task::query()->create([
            Task::NAME => $name,
            Task::GROUP_ID => $groupId,
            Task::DESCRIPTION => $description,
            Task::DOCUMENT_PATH => $documentPath,
            Task::USER_ID => $userId,
            Task::MARK => $mark
        ]);
    }

    /**
     * @throws Exception
     */
    public function get($id)
    {
        $task = Task::query()->find($id);
        $this->validateTask($task);
        return $task;
    }

    /**
     * @throws Exception
     */
    public function completeTask($task_id, $document_path)
    {
        $task = Task::query()->find($task_id);
        $this->validateTask($task);
        if(!$task->complete || $task->complete->mark != ""){
            return TaskComplete::query()->create([
                TaskComplete::TASK_ID => $task_id,
                TaskComplete::USER_ID => auth()->id(),
                TaskComplete::DOCUMENT_PATH =>
$document_path
            ]);
        } else{
            $task->complete->update([
                TaskComplete::DOCUMENT_PATH =>
$document_path
            ]);
            return $task->complete;
        }

    }

    /**
     * @throws Exception
     */
    public function getCompleteList($task_id): array
    {
        $task = Task::query()->find($task_id);
        $this->validateTask($task);
        $userList = [];
        $completes = [];
        $allCompletes =
$task->completes()->orderBy('created_at', 'desc')->get();
        foreach ($allCompletes as $complete) {
            if (!key_exists($complete->user_id, $userList)) {
                $userList[$complete->user_id] = 1;
                $completes[] = $complete;
            }
        }
        return $completes;
    }
```

```
    /**
     * @throws Exception
     */
    public function getComplete($id)
    {
        $complete = TaskComplete::query()->find($id);
        if (!$complete) {
            throw new Exception('Роботу не знайдено');
        }
        $task = Task::query()->find($complete->task_id);
        $this->validateTask($task);
        return $complete;
    }

    /**
     * @throws Exception
     */
    public function reviewTask($complete_id, $mark,
$comment)
    {
        $complete = TaskComplete::query()->find($complete_id);
        if (!$complete) {
            throw new Exception('Роботу не знайдено');
        }
        $this->validateTask($complete->task);
        $review =
TaskReview::query()->where(TaskReview::COMPLETE_ID, '=',
$complete_id)
            ->first();
        if (!$review) {
            return TaskReview::query()->create([
                TaskReview::USER_ID => auth()->id(),
                TaskReview::COMPLETE_ID => $complete_id,
                TaskReview::COMMENT => $comment,
                TaskReview::MARK => $mark
            ]);
        } else {
             $review->update([
                TaskReview::COMMENT => $comment,
                TaskReview::MARK => $mark
            ]);
            return $review;
        }
    }

    /**
     * @throws Exception
     */
    public function validateTask($task)
    {
        if (!$task) {
            throw new Exception('Завдання не знайдено');
        }
        $group = Group::query()->find($task->group_id);
        $this->groupService->validate($group);
    }

    public function deleteTask($id)
    {
        $task = Task::query()->find($id);
        $this->validateTask($task);
        $task->delete();
    }

}
```

AuthService.php

```
<?php

namespace App\Service;
use App\Models\User;
use Illuminate\Database\Eloquent\Builder;
use Illuminate\Database\Eloquent\Model;
use Illuminate\Support\Facades\Auth;

class AuthService
{
    /**
     * @return string
     */
    public function generateToken():string
    {
        return
auth()->user()->createToken('tokens')->plainTextToken;
    }

    public function removeToken()
    {
        auth()->user()->tokens()->delete();
    }

    /**
     * @param string $name
     * @param string $email
     * @param string $password
     * @param string $image
     * @return Builder|Model
     */
    public function createUser(
        string $name,
        string $email,
        string $password,
        string $image
    ) {
        $user = User::query()->create([
            'name' => $name,
            'password' => bcrypt($password),
            'email' => $email,
            'image' => $image
        ]);
        auth()->setUser($user);
        return $user;
    }
}
```

GroupService.php

```
<?php

namespace App\Service;

use App\Models\Group;
use App\Models\GroupUsers;
use Exception;
use Illuminate\Database\Eloquent\Builder;
use Illuminate\Database\Eloquent\Model;

class GroupService
{
    /**
     * @param string $name
     * @return Builder|Model
     */
    public function create(string $name)
    {
        $userId = auth()->user()->id;
        $code = $this->generateRandomString();
        return Group::query()->create([
            Group::NAME => $name,
            Group::USER_ID => $userId,
            Group::CODE => $code
        ]);
    }

    public function generateRandomString($length = 7) {
```

```
        $characters =
'0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKL
MNOPQRSTUVWXYZ';
        $charactersLength = strlen($characters);
        $randomString = '';
        for ($i = 0; $i < $length; $i++) {
            $randomString .= $characters[rand(0,
$charactersLength - 1)];
        }
        return $randomString;
    }

    public function getGroupByFilter($filter)
    {
        $groups = [];
        $userId = auth()->user()->id;
        switch ($filter){
            case "my": {
                $groups = Group::query()
                    ->where(Group::USER_ID, '=', $userId)
                    ->get();
                break;
            }
            case "all": {
                $groupUsers = GroupUsers::query()
                    ->where(GroupUsers::USER_ID, '=',$userId)
                    ->get();

                $groupIds = [];
                foreach ($groupUsers as $groupUser){
                    $groupIds[] = $groupUser->group_id;
                }
                $groups = Group::query()
                    ->whereIn(Group::ID, $groupIds)
                    ->get();
                break;
            }
            default: {

            }

        }
        return $groups;
    }

    /**
     * @throws Exception
     */
    public function joinGroup($groupCode)
    {
        $userId = auth()->user()->id;
        $group = Group::query()
            ->where('code', '=', $groupCode)
            ->first();
        if(is_null($group)){
            throw new Exception('Не вдалось знайти групу');
        }
        $groupUser = GroupUsers::query()
            ->where(GroupUsers::USER_ID, '=',$userId)
            ->where(GroupUsers::GROUP_ID, '=', $group->id)
            ->first();
        if(!is_null($groupUser)) {
            throw new Exception('Користувач вже приєднаний
до цієї групи');
        }
        return GroupUsers::query()
            ->create([
                GroupUsers::GROUP_ID => $group->id,
                GroupUsers::USER_ID => $userId
            ]);
    }

    /**
     * @throws Exception
     */
    public function deleteGroup(int $id)
    {
        $group = Group::query()->find($id);
        if(!$group){
            throw new Exception("Дана група не існує");
        }
        $group->delete();
    }

    /**
     * @throws Exception
     */
    public function leftGroup(int $id)
    {
        $group = Group::query()->find($id);
        $this->validate($group);

        $groupUser = GroupUsers::query()
            ->where(GroupUsers::GROUP_ID, '=', $id)
            ->where(GroupUsers::USER_ID,'=', auth()->user()->id)
            ->first();
        $groupUser->delete();
    }

    /**
     * @throws Exception
     */
    public function leftUserFromGroup($user_id, $group_id)
    {
        $group = Group::query()->find($group_id);
        $this->validate($group);
        $groupUser = GroupUsers::query()
            ->where(GroupUsers::GROUP_ID, '=', $group_id)
            ->where(GroupUsers::USER_ID,'=', $user_id)
            ->first();
        $groupUser->delete();
    }

    /**
     * @throws Exception
     */
    public function get($id)
    {
        $group = Group::query()->find($id);
        $this->validate($group);
        return $group;
    }

    /**
     * @throws Exception
     */
    public function validate($group)
    {
        if(!$group){
            throw new Exception("Групу не знайдено");
        }
        if(!$group->canAccess && !$group->isAdmin){
            throw new Exception("Ви не входите в цю групу");
        }
    }
}
```

ДОДАТОК Б

(обов’язковий)

ПРЕЗЕНТАЦІЙНИЙ МАТЕРІАЛ

Веб-додаток для менеджменту навчання та автоматизованого контролю знань студентів

Хмельницький національний університет

Кафедра інженерії програмного забезпечення

Виконав: Гурний Р.В.

Керівник: кандидат педагогічних наук, доцент Онишко О.Г.

Рисунок Б.1 – Слайд №1

Актуальність теми

- Зростання популярності дистанційного навчання
- Сприяння цифровізації освіти в Україні
- Використання монопольних систем (Google Classroom, Microsoft Teams)

2

Рисунок Б.2 – Слайд №2

Мета та завдання

Створення веб-додатку для організації дистанційного навчання та контролю знань

- Дослідити предметну область та аналоги;
- Визначити вимоги;
- Розробити архітектуру програмного забезпечення;
- Реалізувати систему на основі обраного стеку технологій;
- Провести тестування та оцінити результати.

3

Рисунок Б.3 – Слайд №3

Змістовний аналіз предметної області

Структури даних

Користувач, Завдання, Відгук, Теорія, Клас, Адміністратор, Група

Функціональні особливості

Створення груп, завдань, перегляд теорії, оцінювання, відеодзвінки, нагадування, журнал

Предметна область

Автоматизація перевірки знань та онлайн-навчання

4

Рисунок Б.4 – Слайд №4

Рисунок Б.5 – Слайд №5

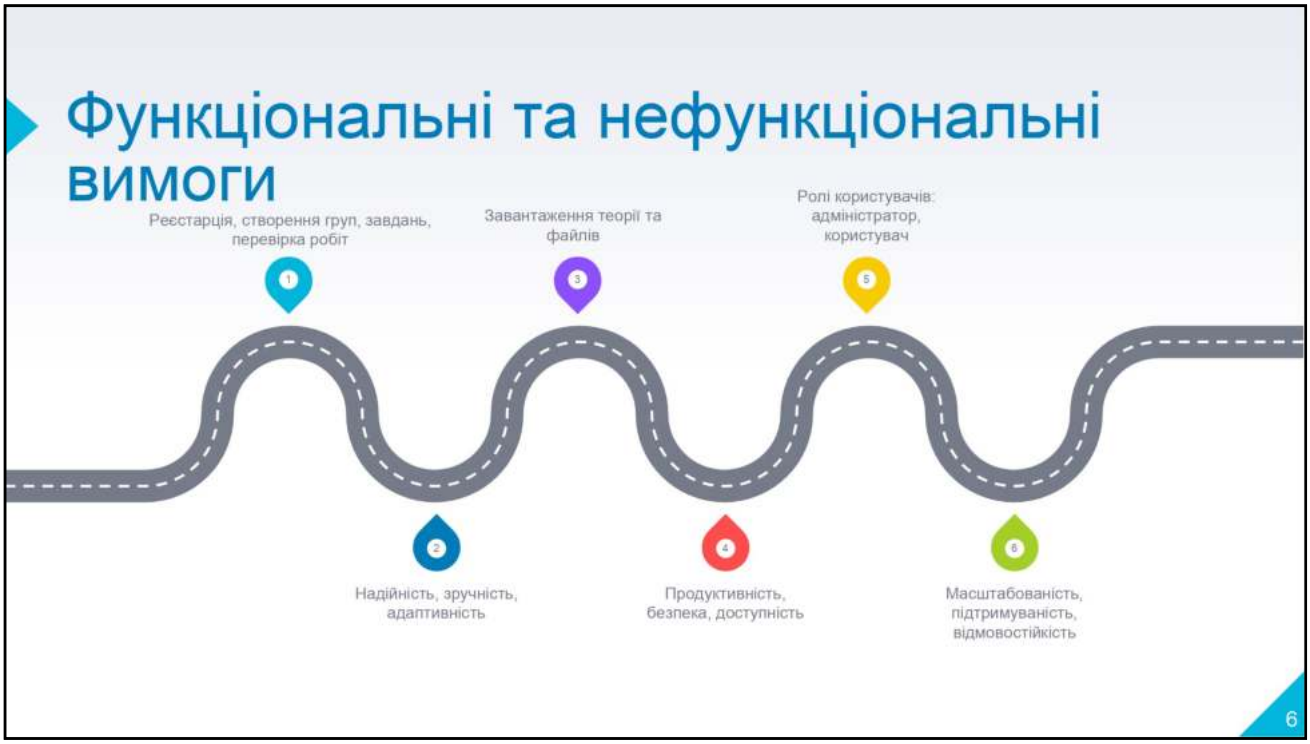


Рисунок Б.6 – Слайд №6

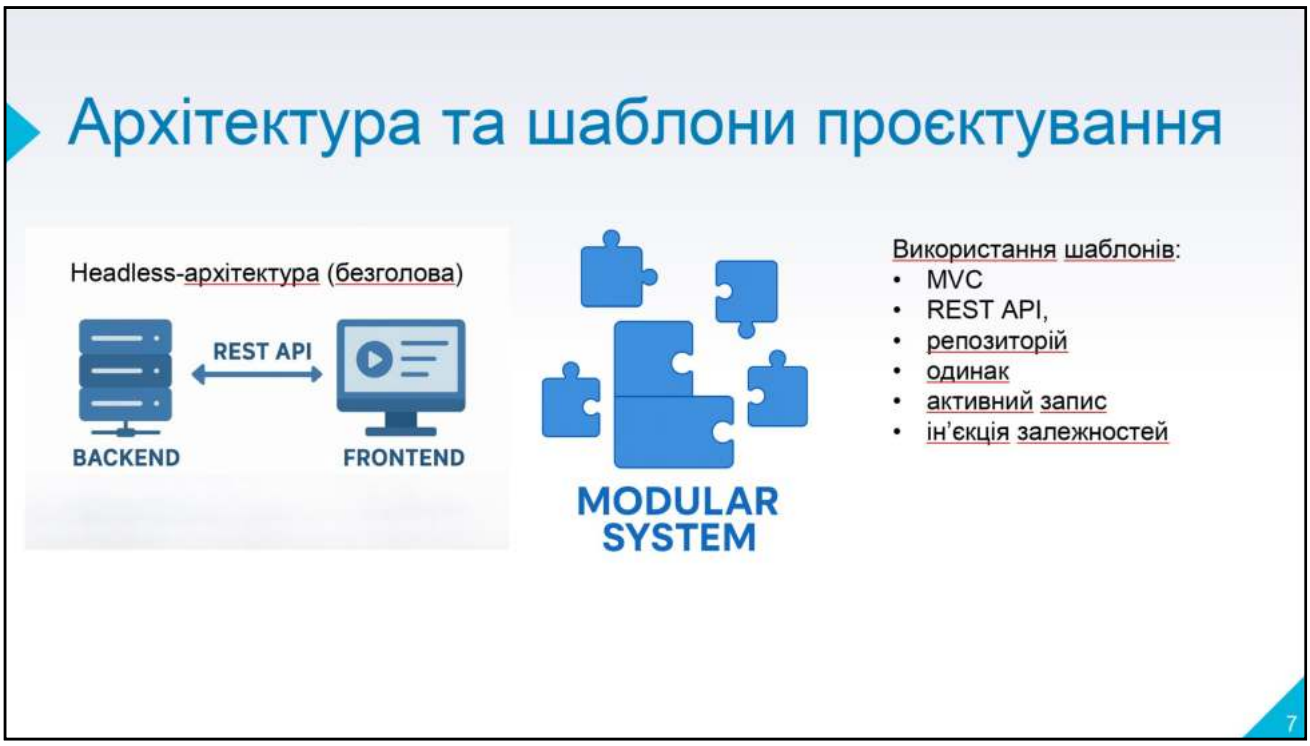
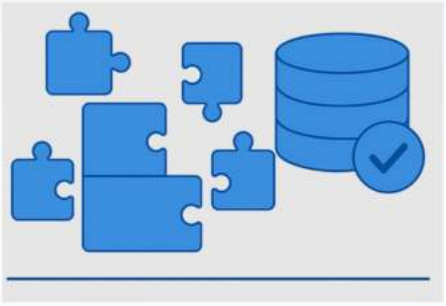


Рисунок Б.7 – Слайд №7

Рисунок Б.8 – Слайд №8

Проєктування модулів і даних

- Основні сутності: Користувач, Група, Завдання, Теорія.
- Реляційна структура бази даних
- Нормалізація та оптимізація структури даних
- Зв'язки між таблицями, зовнішні ключі



9

Рисунок Б.9 – Слайд №9

Аналіз та вибір технологій

VUE JS

- Реактивність
- Фреймворк, що радить
- Компонентна структура

Займає 3 місце серед відомих засобів для інтерфейсної розробки

Laravel

- Простота
- Швидкість
- Компактність

Дозволяє обробляти запити для всіх різновидів API та має потужні інструменти для роботи з даними

10

Рисунок Б.10 – Слайд №10

Рисунок Б.11 – Слайд №11

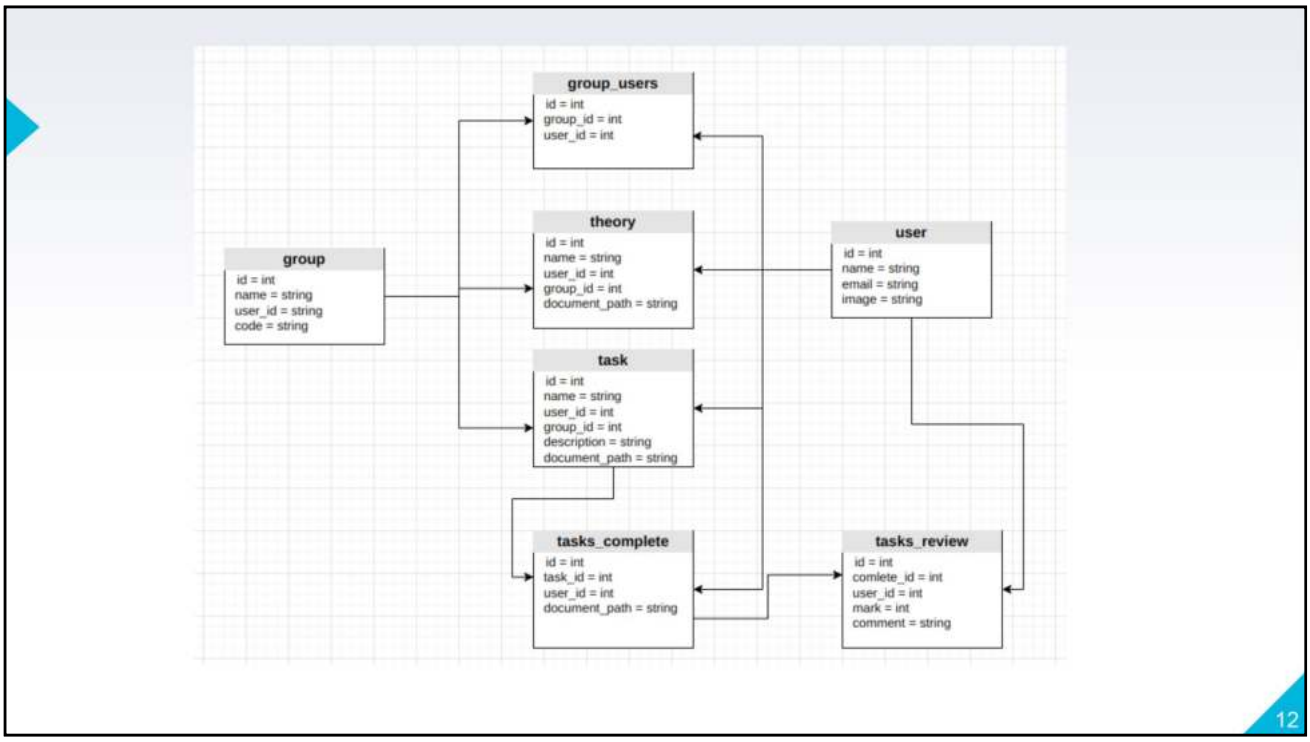


Рисунок Б.12 – Слайд №12

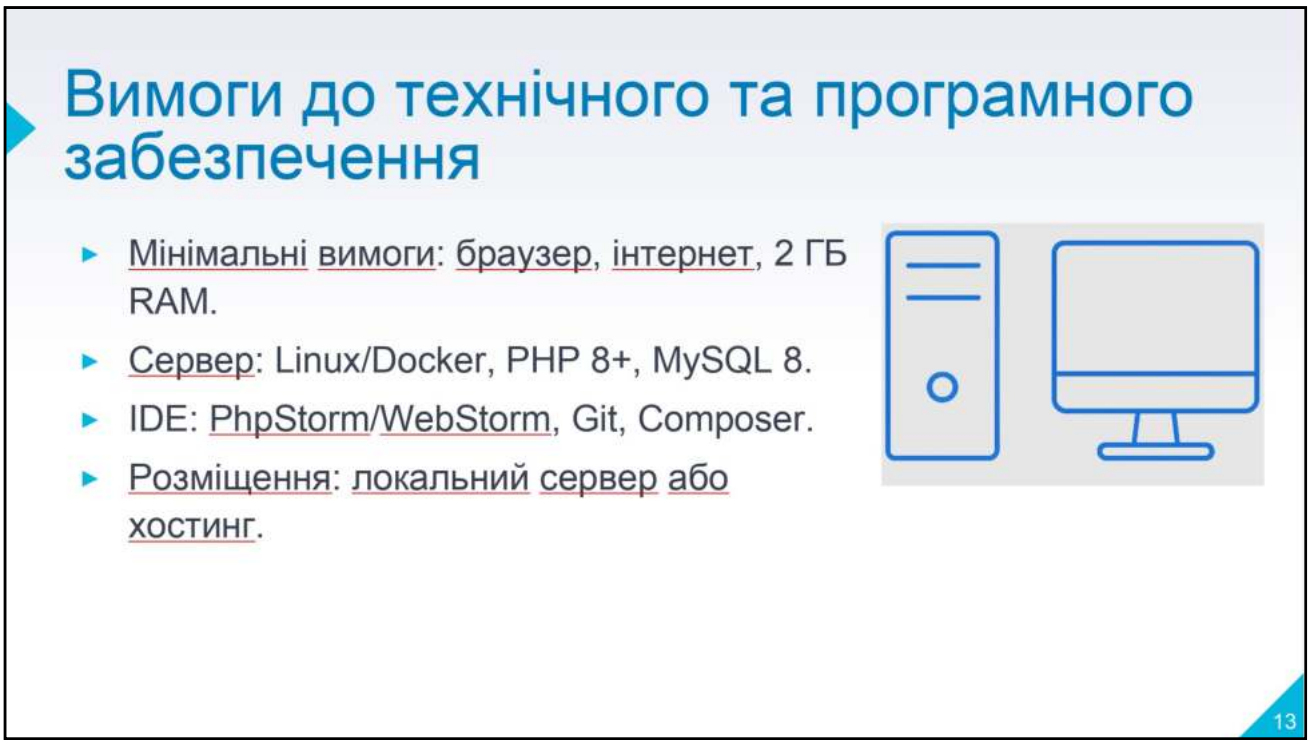


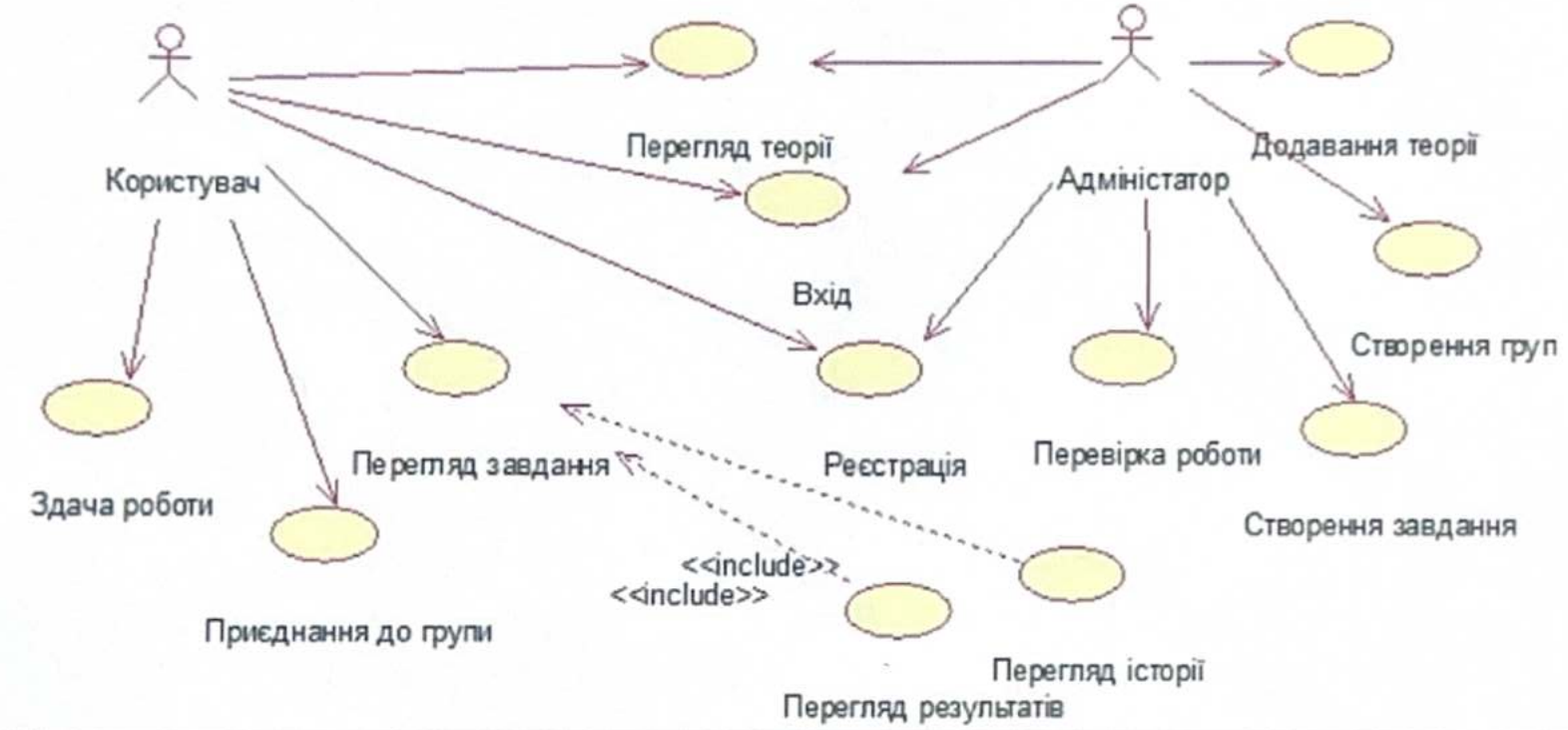
Рисунок Б.13 – Слайд №13

Рисунок Б.14 – Слайд №14

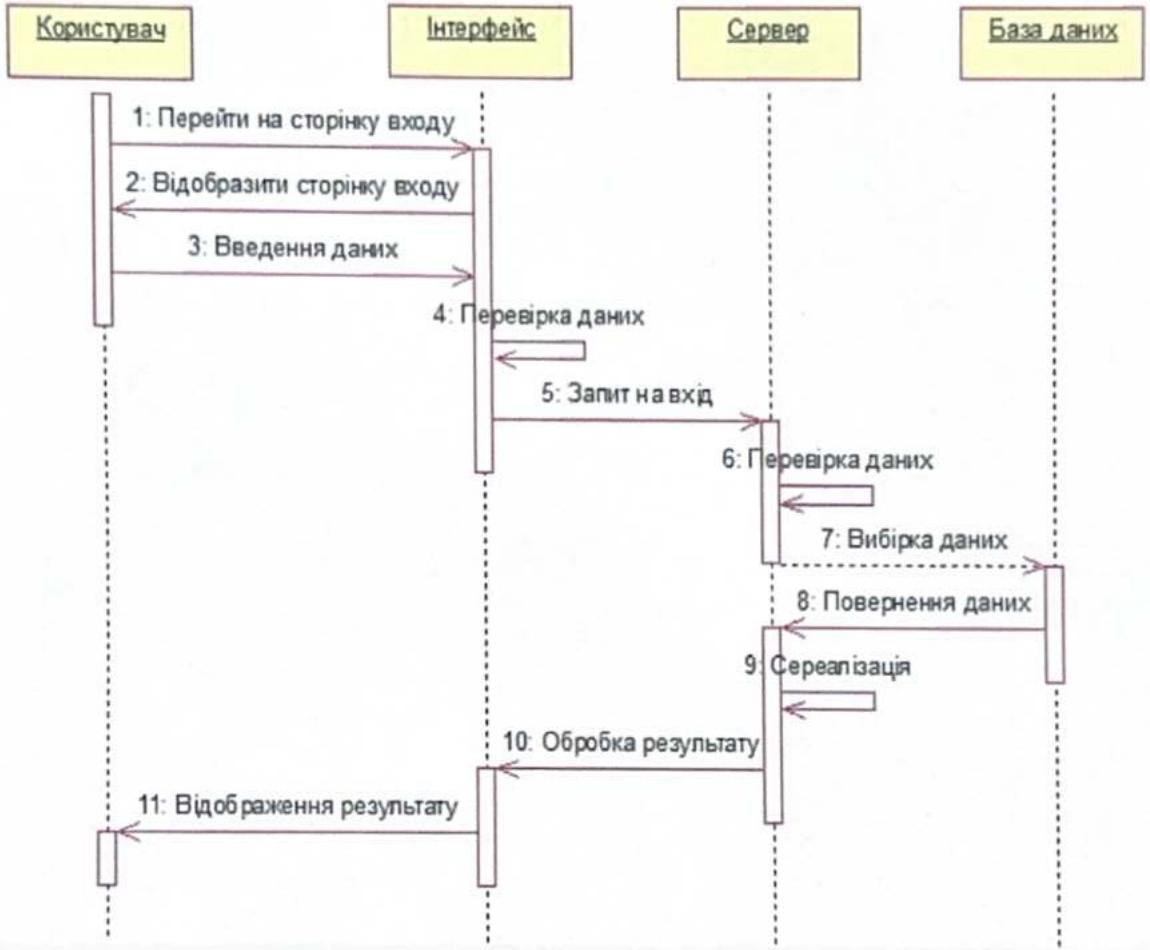
Рисунок Б.15 – Слайд №15

Рисунок Б.16 – Слайд №16

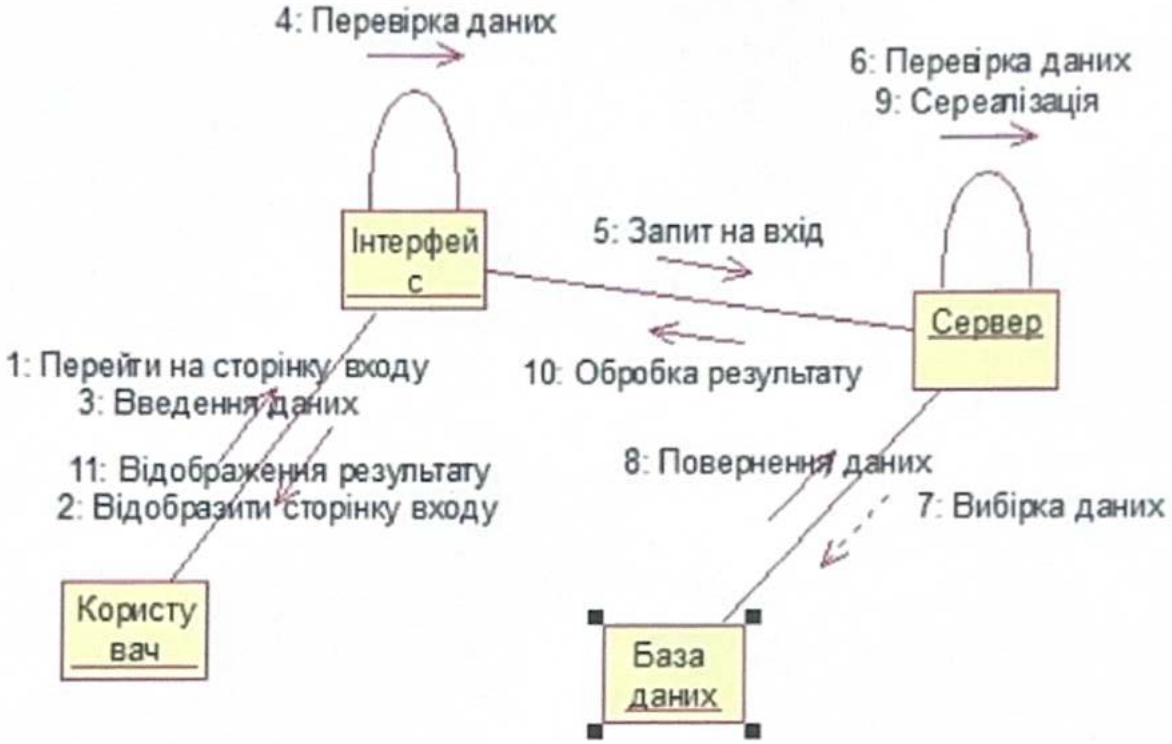
ГРАФІЧНА ЧАСТИНА



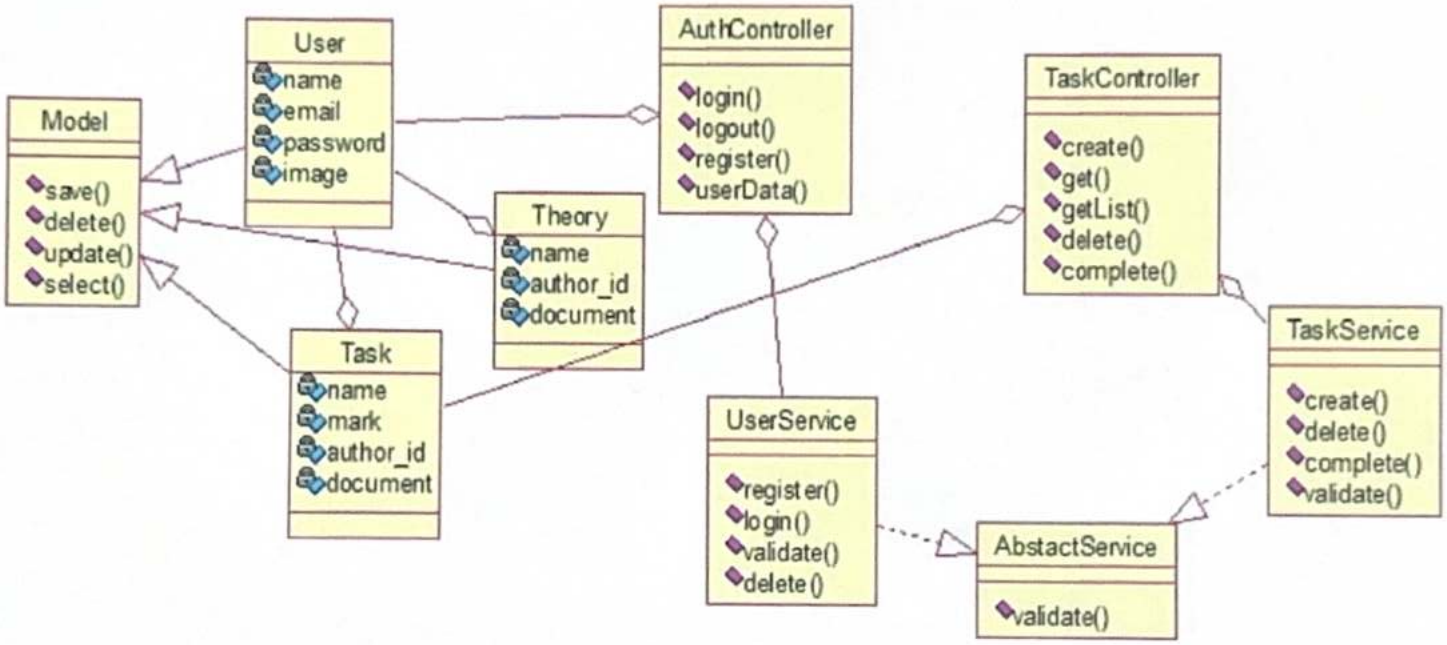
Змін.	Аркуш	№ докум.	Підпис	Дата	КвРІПЗ.2201123.01.02.Е1		
Виконав		Гурний Р.В.			Діаграма використання	Літ.	Аркуш 1 / Аркушів 6
Керівник		Онишко О.Г.				ХНУ, ІПЗс-22-1	
Рецензент							
Н. контр.		Яшина О.М.					
Зав. каф.		Бедратюк Л.П.					



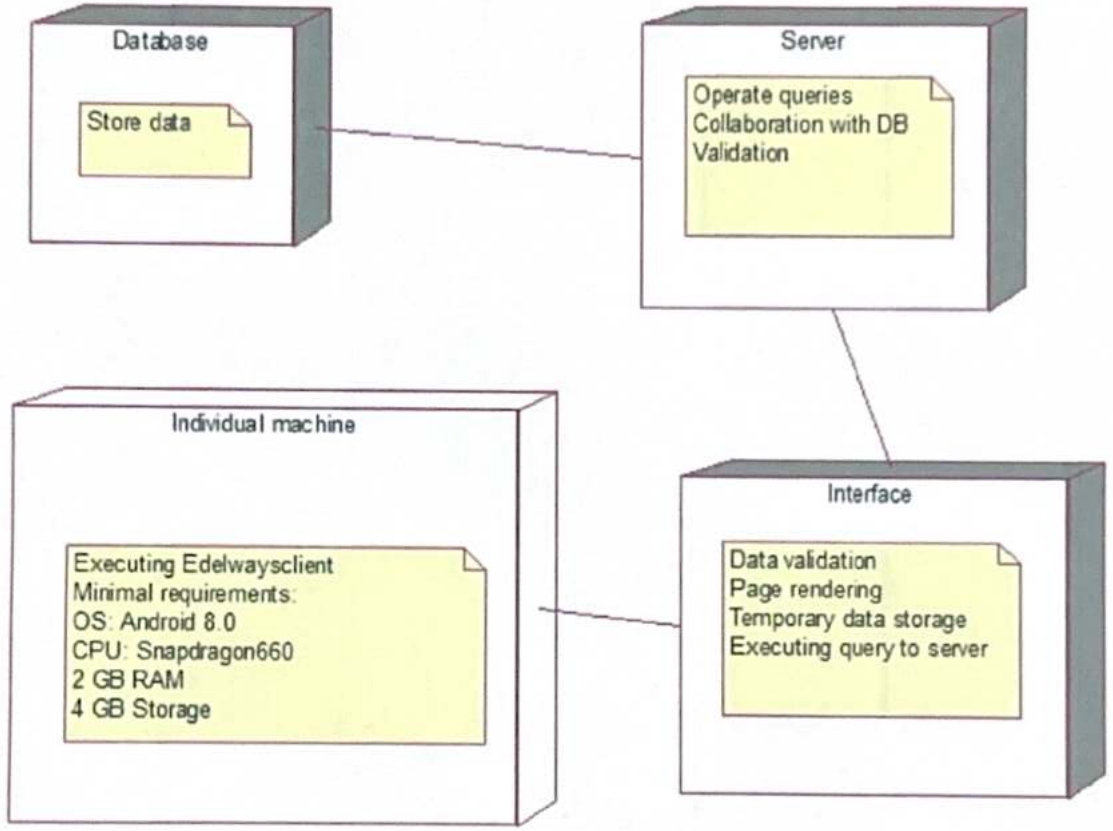
Змін.	Аркуш	№ докум.	Підпис	Дата	КвРІПЗ.2201123.01.02.Е2		
Виконав		Гурний Р.В.			Діаграма послідовності	Літ	Аркуш 2 / Аркушів 6
Керівник		Онишко О.Г.					
Рецензент						ХНУ, ІПЗс-22-1	
Н. контр.		Яшина О.М.					
Зав. каф.		Бедратюк Л.П.					



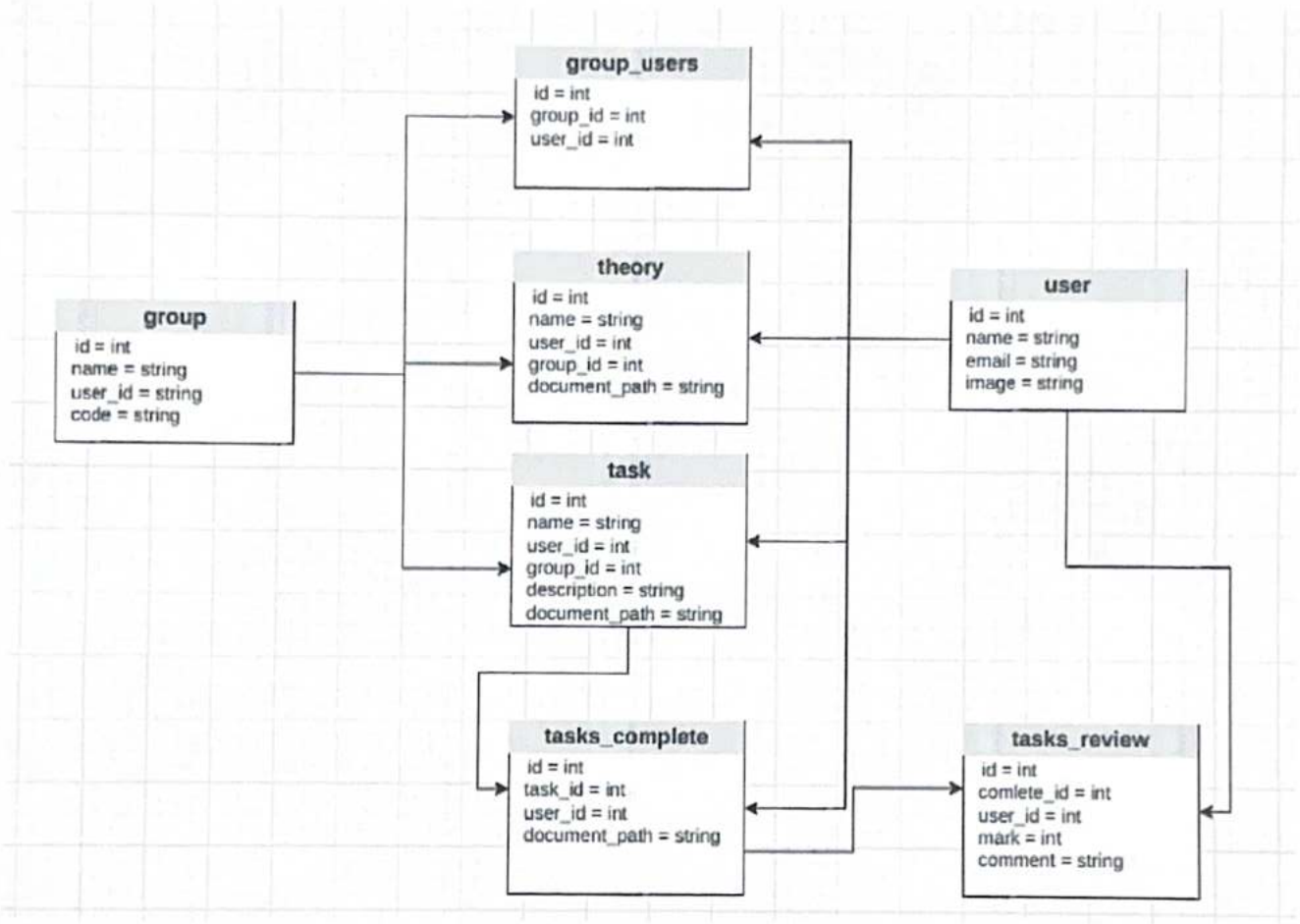
Змін.	Аркуш	№ докум.	Підпис	Дата	КвРІПЗ.2201123.01.02.Е3		
Виконав		Гурний Р.В.			Діаграма взаємодії	Літ	Аркуш 3 / Аркушів 6
Керівник		Онишко О.Г.					
Рецензент						ХНУ, ІПЗс-22-1	
Н. контр.		Яшина О.М.					
Зав. каф.		Бедратюк Л.П.					



Змін.	Аркуш	№ докум.	Підпис	Дата	КвРІПЗ.2201123.01.02.Е4		
Виконав		Гурний Р.В.			Діаграма класів	Літ	Аркуш 4 / Аркушів 6
Керівник		Онишко О.Г.					
Рецензент						ХНУ, ІПЗс-22-1	
Н. контр.		Яшина О.М.					
Зав. каф.		Бедратюк Л.П.					



Змін.	Аркуш	№ докум.	Підпис	Дата	КвРІПЗ.2201123.01.02.Е5		
Виконав		Гурний Р.В.			Діаграма компонентів	Літ	Аркуш 5 / Аркушів 6
Керівник		Онишко О.Г.					
Рецензент						ХНУ, ІПЗс-22-1	
Н. контр.		Яшина О.М.					
Зав. каф.		Бедратюк Л.П.					



Змін.	Аркуш	№ докум.	Підпис	Дата	КвРІПЗ.2201123.01.02.Е6		
Виконав		Гурний Р.В.			Схема бази даних	Літ	Аркуш 6 / Аркушів 6
Керівник		Онишко О.Г.					
Рецензент						ХНУ, ІПЗс-22-1	
Н. контр.		Яшина О.М.					
Зав. каф.		Бедратюк Л.П.					

СУПРОВІДНІ ДОКУМЕНТИ

Завідувачу кафедри інженерії програмного
забезпечення проф. Леоніду БЕДРАТЮКУ
здобувача вищої освіти
Гурного Романа Вікторовича
факультет ІТ, ІІІ курс, група ІПЗс-22-1

ЗАЯВА

З правилами чинного Положення про систему забезпечення академічної доброчесності в Хмельницькому національному університеті, згідно з яким виявлення академічного плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту і застосування заходів дисциплінарної та академічної відповідальності, ознайомлений. Про використання програмно-технічних засобів для перевірки кваліфікаційних робіт здобувачів вищої освіти на наявність академічного плагіату оповіщений та надаю свою згоду на обробку й збереження університетом моєї роботи в інституційному репозитарії Хмельницького національного університету.

Також надаю університету право на передачу моєї роботи для обробки та збереження в базах даних програмно-обчислювального комплексу StrikePlagiarism та/або програмно-технічного засобу AntiPlagiarism і використання роботи для виявлення академічного плагіату в інших роботах, які перевіряються програмно-технічними засобами та користувачами, що мають доступ до цих програмно-технічних засобів, виключно в обмежених цілях для виявлення текстових збігів у роботах.

Робота надається для перевірки в електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

01.06.2025
дата


підпис

Anti-Plagiarism (UA) v-15.281 Educational

The maximum coincidence with one document 1.0%

Dictionaries check: en_US, ru_RU, ua_UA. **Errors in the documents: 8%**

ID: 243712 Title: БКР_Веб-додаток для менеджменту навчання та автоматизованого контролю знань студентів Added in a DB: 2025-06-05 Authors: Роман ГУРНИЙ Heads: ОНИШКО Оксана Consultants: Opponents:	Document		Sum coincidence on the DB	
	Symbols	Lexemes	Symbols	Lexemes
	81208	783	3503 (4%)	42 (5%)

Plagiarism sources

ID	Description	Plagiarism presence in the document	
		Symbols	Lexemes

Протокол аналізу звіту подібності експертом

Заявляю, що я ознайомився (-лась) з Повним звітом подібності, який був згенерований Системою виявлення і запобігання плагіату щодо роботи:

Автор: Гурнии Роман

Співавтор:

Назва: БКР Веб-додаток для менеджменту навчання та автоматизованого контролю знань

Експерт:

Підрозділ: Кафедра інженерії програмного забезпечення

Коефіцієнт подібності 1:16.7%

Коефіцієнт подібності 2:8.9%

Мікропробіли: 0

Заміна букв: 4

Інтервали: 0

Білі знаки: 0

Дата створення звіту: 2025-06-04 14:27:33.0

Після аналізу Звіту подібності констатую наступне:

☑ **Запозичення, виявлені в роботі є законними і не є плагіатом. Рівень подібності не перевищує допустимої межі. Таким чином робота незалежна і приймається.**

☐ **Запозичення не є плагіатом, але перевищено граничне значення рівня подібностей. Таким чином робота повертається на доопрацювання.**

☐ **Виявлено запозичення і плагіат або навмисні текстові спотворення (маніпуляції), як передбачувані спроби укриття плагіату, які роблять роботу невідповідною вимогам законодавства (Ст. 32. ЗУ Про вищу освіту, пункт 3.1, Ст. 42. ЗУ Про освіту) та вимог НАЗЯВО (Критерій 5), а також кодексу етики і процедурам. Таким чином робота не приймається.**

Обгрунтування:

10.06.2025
Дата

експерт

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ
освітнього ступеня «Бакалавр»

Дипломник Гурний Роман Вікторович

Тема Веб додаток для менеджменту навчання та автоматизованого контролю знань студентів

Спеціальність 121 – Інженерія програмного забезпечення

Обсяг кваліфікаційної роботи:

Кількість листів креслень 6; кількість сторінок записки 93

1.Короткий зміст пояснювальної записки та прийнятих рішень У кваліфікаційній роботі проведено аналіз предметної області систем для навчання, досліджено актуальні вимоги до програмного забезпечення такого типу. Визначено функціональні та нефункціональні вимоги до автоматизованої системи перевірки знань студентів. Було проаналізовано існуючі рішення на ринку, виявлено їхні переваги та недоліки. На основі цього обґрунтовано необхідність розробки власного рішення. Вибрано сучасні інструменти та архітектуру для реалізації серверної та інтерфейсної частин, виконано програмну реалізацію обох модулів. Розроблену систему протестовано, результати тестування підтвердили працездатність та відповідність функціональним вимогам.

2. Висновок про відповідність роботи поставленому завданню Кваліфікаційна робота виконана відповідно до поставленого завдання, охоплює всі етапи розробки серверної та інтерфейсної частин системи та відповідає технічному завданню.

3. Характеристика виконання кожного розділу роботи, ступінь використання останніх досягнень науки і техніки та передових методів роботи У вступі обґрунтовано актуальність теми, сформульовано мету, задачі та об'єкт дослідження. У першому розділі здійснено огляд предметної області, проведено аналіз існуючих рішень та сформовано вимоги до розробки. У другому розділі досліджено сучасні архітектурні підходи, обґрунтовано вибір клієнт-серверної архітектури з REST API та базою даних. У третьому розділі реалізовано серверну та інтерфейсну частини: описано структуру коду, реалізовано основні API-ендпоінти, налаштовано взаємодію з базою даних, проведено модульне тестування та оцінено результат. Робота базується на сучасних технологіях, таких як Laravel, MySQL, VueJs 3 тощо.

4. Позитивні сторони роботи Тематика є актуальною, адже системи для реалізації автоматичної перевірки залишаються важливою складовою процесу навчання . У роботі

реалізовано сучасні технічні рішення, дотримано принципів безпеки, масштабованості та структурованості коду. Документація до API є зрозумілою та відповідає сучасним стандартам розробки.

5. Негативні сторони роботи Робота містить тільки базовий набір функцій для перевірки знань та реалізації дистанційного навчання і потребує доповнення та оптимізації. Також було б доцільно реалізувати систему для створення завдань у вигляді тестів з різними типами завдань.

6. Оцінка графічного оформлення та пояснювальної записки Графічний матеріал представлений у вигляді діаграм архітектури, моделей бази даних та схем взаємодії компонентів. Пояснювальна записка оформлена відповідно до діючих стандартів, має логічну структуру та достатню глибину розкриття теми.

7. Відгук про кваліфікаційну роботу в цілому Кваліфікаційна робота виконана на достатньо високому рівні. Вона структурована, змістовна, відображає глибоке розуміння предметної області. Технічна реалізація відповідає сучасним вимогам до розробки серверного програмного забезпечення.

8. Інші зауваження Було б доцільно опублікувати код системи та зробити його публічним. Це дозволило б зацікавленим розробникам вносити більше функцій та покращень.

9. Оцінка кваліфікаційної роботи Кваліфікаційна робота виконана у повному обсязі, відповідає поставленій задачі та заслуговує на оцінку «добре».

РЕЦЕНЗЕНТ (ПІБ, посада, місце роботи) Клюєв Юрій К.Т.Н., зав. кафедри кібербезпеки, ХНУ

"4" 06 2025 р. _____
(підпис)

РІШЕННЯ ЕКСПЕРТНОЇ КОМІСІЇ
КАФЕДРИ ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ
ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатами звіту/звітів перевірки роботи, продукованими програмно-технічним засобом (ами), на наявність текстових збігів.
Назва кваліфікаційної роботи: «Веб-додаток для менеджменту навчання та автоматизованого контролю знань студентів»
Автор: Гурний Роман Вікторович
Освітня програма: Освітньо-професійна програма «Інженерія програмного забезпечення»
Спеціальність: 121 – Інженерія програмного забезпечення
Науковий керівник: Онишко Оксана Григорівна, кандидат педагогічних наук, доцент
Після аналізу звіту/звітів зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі є законними і не є плагіатом. Рівень подібності не перевищує допустимої межі. Таким чином робота незалежна і приймається до захисту.	**відповідає**
2	Виявлені запозичення не є академічним плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи. Робота приймається до захисту, але має бути відкоригована.	
3	Виявлені запозичення не є плагіатом, але перевищено граничне значення рівня подібностей. Виявлені запозичення частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнуті. Робота може бути допущена до захисту після того, як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття текстових запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	
5	Інше:	

Підтвердження:

Запозичення, виявлені у роботі, є законними і не є плагіатом, оскільки:

1) у тексті кваліфікаційної роботи системою виявлення і запобігання плагіату Anti-Plagiarism виявлено схожість з деякими документами у частині загальновживаних обов'язкових словосполучень, що зустрічаються у типовій технічній документації, а також у структурі змісту, назвах розділів/підрозділів, що є характерним для робіт даного напряму. Дані свідчать про використання поширених фраз, технічної термінології та елементів, які не становлять предмету авторського права;

2) запозичення, виявлені у тексті роботи, є фрагментарними.

Максимальний обсяг запозичень, визначений системою Anti-Plagiarism, становить 1.0% з одного джерела. Загальна сумарна подібність у базі даних складає 4% за символами та 5% за лексемами. Крім того, за результатами додаткового аналізу коефіцієнт подібності 1 становить 16.7%, коефіцієнт подібності 2 - 8.9%. Не виявлено мікропробілів, зайвих білих знаків або маніпуляцій з інтервалами, а зафіксована заміна символів (4 випадків) не впливає на змістовну унікальність тексту.

Дата 10.06.2025

Завідувач кафедри _____ Леонід БЕДРАТЮК

Гарант освітньої програми _____ Леонід БЕДРАТЮК

Керівник кваліфікаційної роботи _____ Оксана ОНИШКО

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ДЕКЛАРАЦІЯ УЧАСНИКА ОСВІТНЬОГО ПРОЦЕСУ

щодо дотримання академічної доброчесності

Цією декларацією я, Гурний Роман Вікторович,

студент III курсу спеціальності 121 – Інженерія програмного забезпечення, група ІПЗс-22-1

здобувач вищої освіти (шифр та назва спец-ті, курс, академічна група)

підтверджую, що ознайомився (-лась) з Положенням про систему забезпечення академічної доброчесності у Хмельницькому національному університеті та Кодексом академічної доброчесності і **зобов'язуюсь** дотримуватися їх вимог під час освітнього процесу, проведення наукової діяльності, виконання організаційно-адміністративних функцій тощо.

Усвідомлюю, що у разі порушення мною принципів академічної доброчесності нестиму відповідальність перед академічною спільнотою ХНУ згідно з нормами, визначеними Положенням про систему забезпечення академічної доброчесності у Хмельницькому національному університеті, законодавства України.

01.06. 2025 р. _____

Підпис

Завідувачу кафедри
інженерії програмного забезпечення
проф. Бедратюку Л. П.
студента групи ІПЗс-22-1
Гурного Р. В.
Прізвище, ініціали

ЗАЯВА

Прошу закріпити за мною тему кваліфікаційної роботи освітнього ступеня «бакалавр» за спеціальністю 121 «Інженерія програмного забезпечення»: Веб-додаток для менеджменту навчання та автоматизованого контролю знань студентів

(керівник роботи – Онишко Оксана Григорівна)
Прізвище, ім'я, по батькові

01.06.2025
Дата

Підпис студента