Хмельницький національний університет
Факультет інформаційних технологій
Кафедра інженерії програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА

Веб-платформа "LabHub" для обміну інформацією на базі фреймворка Laravel

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»

Шифр КвРІПЗ.200123.20.04.ПЗ

Виконав студент III курсу група ІПЗс-20-1 _____ В.К. Варук
Підпис Ініціали, прізвище

Керівник д-р фіз.мат. наук, професор _____ Л.П. Бедратюк
Науковий ступінь, звання Підпис Ініціали, прізвище

Нормоконтролер канд. тех. наук, доцент _____ І.В. Гурман
Підпис Ініціали, прізвище

До захисту допускаю:
Завідувач кафедри інженерії
програмного забезпечення _____ Л. П. Бедратюк
Підпис Ініціали, прізвище

7 червня 2023 р.

Хмельницький 2023

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
05 02 2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Варуку Валентину Костянтиновичу
Прізвище, ім'я, по батькові студента

1. Тема проекту (роботи) Веб-платформа "LabHub" для обміну інформацією на базі фреймворка Laravel

Керівник проекту (роботи) Бедратюк Леонід Петрович, д-р фіз.-мат. наук, професор
Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджена наказом ректора університету від 05.02.2023 р. № 11
2. Строк подання студентом проекту (роботи) на кафедру 01.06.2023 р.
3. Вихідні дані до проекту (роботи) Матеріали переддипломної практики
4. Зміст пояснювальної записки (перелік питань, які потрібно розробити) Дослідження предметної області та постановка задачі, проектування програмного забезпечення, програмна реалізація, тестування програмного забезпечення
5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень) Презентаційні матеріали (слайди, 19 шт.)
Креслення

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
Л. П. Бедратюк
05 02 2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Варуку Валентину Костянтиновичу
Прізвище, ім'я, по батькові студента

1. Тема проекту (роботи) Веб-платформа "LabHub" для обміну інформацією на базі фреймворка Laravel

Керівник проекту (роботи) Бедратюк Леонід Петрович, д-р фіз.-мат. наук, професор
Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджена наказом ректора університету від 05.02.2023 р. № 11
2. Строк подання студентом проекту (роботи) на кафедру 01.06.2023 р.
3. Вихідні дані до проекту (роботи) Матеріали переддипломної практики
4. Зміст пояснювальної записки (перелік питань, які потрібно розробити) Дослідження предметної області та постановка задачі, проектування програмного забезпечення, програмна реалізація, тестування програмного забезпечення
5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень) Презентаційні матеріали (слайди, 19 шт.)
Креслення

АНОТАЦІЯ

Тема кваліфікаційної роботи «Веб-платформа “LabHub” для обміну інформацією на базі фреймворка Laravel».

Автор роботи: Варук Валентин Костянтинович.

Керівник роботи: Бедратюк Леонід Петрович.

Пояснювальна записка: 55 с., 41 рис., 1 табл., 4 дод., 25 джерел.

Графічна частина: 3 креслення ф. А3.

ВЕБ-ПЛАТФОРМА, БАЗА ДАНИХ, PHP, LARAVEL, BOOTSTRAP.

Мета кваліфікаційної роботи: розробка веб-платформи для поліпшення пошуку та обміну інформацією, заснованого на бартерному принципі між користувачами.

У кваліфікаційній роботі проведено аналіз предметної області та її інформаційного забезпечення, визначені функціональні вимоги до програмної системи, розроблена загальна архітектура додатку, спроектована структура бази даних бібліотеки та структура додатку. Для розробки програмного продукту використано мову розмітки HTML, Bootstrap та фреймворк Laravel. За допомогою цих засобів розроблено програмне забезпечення для поліпшення пошуку та обміну інформації.

Впровадження розробленого програмного продукту дозволяє автоматизувати ведення та використання бази даних платформи та значно полегшити пошук потрібної інформації.

01.06.23
Дата

Підпис

ВІДОМІСТЬ ДОКУМЕНТІВ

№ рядка	Формат	Позначення документа	Найменування документа	К-сть аркушів	№ екз.	Примітка
			Текстові документи			
1	А4	КвРІПЗ.200123.20.04.ПЗ	Пояснювальна записка	105		
2	А4		Завдання на дипломний	1		
			проект			
3	А4		Анотація	1		
			Графічні документи			
4	А4		Презентаційні матеріали	19		

Змн.	Арк.	№ докум.	Підпис	Дата	КвРІПЗ.200123.01.04.ВД		
Виконав		Ворук В.К.		6.06	Веб-платформа "Lab-hub" для обміну інформацією на базі фреймворка Laravel Відомість документів	Літ.	Арк. 1 / Аркушів 1
Керівник		Бебратюк Л.П.		7.06			
Н. Контр.		Гуржин І.В.		5.06		ХНУ, ІПЗс-20-1	
Зав. Каф.		Бебратюк Л.П.		7.06			

ЗМІСТ

Змн.	Арк.	№ докум.	Підпис	Дата	КвРІПЗ.200123.01.04.ПЗ		
Виконав		Варук В.К.		01.06	Веб-плафторма "LabHub" для обміну інформацією на базі фреймворка Laravel	Літ.	Арк. 4 / Акрушів 61
Керівник		Бедратюк Л.П.		7.06			
Н. Контр.		Гурман І.В.		5.06		ХНУ, ІПЗс-20-1	
Зав. Каф.		Бедратюк Л.П.		7.06	Пояснювальна записка		

ВСТУП

З кожним роком серед студентів спостерігається зростання кількості інтелектуальної праці, яка реалізується в різних напрямках та сферах навчання. Вони займаються розробкою різноманітних цікавих проектів, вкладаючи в них різну кількість часу і зусиль. Ці проекти можуть бути збережені у вигляді коду на платформі GitHub, але існує ймовірність, що вони можуть бути випадково видалені або втрачені через технічні проблеми, такі як форматування переносного носія або видалення з корзини на комп'ютері.

Ці творчі зусилля студентів часто залишаються лише в рамках позитивних оцінок від викладачів та важливого професійного досвіду. Однак, інформація має велике значення у розвитку інтелектуальних здібностей людини. З появою Інтернету зникла необхідність шукати документацію у книгах або газетах, переглядати новини та здійснювати інші пошуки. Зараз достатньо мати підключення до Wi-Fi або мобільного Інтернету, і усю необхідну інформацію можна знайти онлайн.

Однак, є ситуації, коли пошук певної інформації може бути викликаним труднощами. Наприклад, коли потрібно знайти розв'язок певної задачі, проекту або технічного завдання. Це може вимагати великих зусиль, включаючи скріншоти, фотографії низької якості та інші незручні методи збереження інформації.

У світлі цих проблем було створено веб-платформу під назвою "LabHub" з метою полегшення пошуку та обміну інформацією. Ця платформа надає зручний інтерфейс для користувачів, де вони можуть обмінюватися своїми роботами, дослідженнями, експериментами та іншими проектами. "LabHub" пропонує можливість обміну ексклюзивною інформацією, що демонструється в зручному інтерфейсі, де користувачі можуть ознайомитися з тим, яку інформацію пропонують іншим користувачам.

Головною метою кваліфікаційної роботи є розробка веб-платформи для поліпшення пошуку та обміну інформацією, заснованого на бартерному

принципі між користувачами. Це означає, що користувачі можуть знайти потрібну інформацію, а також опублікувати свої роботи для обміну. Використовуючи дану платформу, користувач може значно заощадити час та зусилля, які раніше витрачалися на самостійний пошук інформації в безлічі джерел.

Для досягнення поставленої мети було сформульовано ряд завдань:

– дослідження предметної області веб-платформ для обміну інформацією з метою визначення потреб потенційних користувачів.

– аналіз існуючих рішень на ринку.

– розробити технічне завдання, в якому будуть визначені вимоги до функціональності та характеристик платформи.

– розробити архітектура веб-платформи та бази даних, враховуючи потреби користувачів.

– вибрати відповідні технології для розробки платформи.

– розробити зручного та привітного інтерфейсу для користувачів, який забезпечує зручний доступ до функціональності платформи.

– реалізувати веб-застосунок на основі розробленої архітектури.

– провести тестування готового веб-застосунку для перевірки його працездатності та коректності роботи.

1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Змістовний аналіз предметної області, її структурних та функціональних особливостей

Фреймворк Laravel є одним з найпопулярніших PHP-фреймворків, який пропонує безліч позитивних переваг, що робить його популярним вибором серед розробників для розробки веб-застосунків. Ось деякі з його помітних позитивних переваг:

– елегантний синтаксис: Laravel має чистий і виразний синтаксис, який дозволяє розробникам писати код більш читабельним і зручним для підтримки способом. Фреймворк робить акцент на простоті та дружніх до розробників угодах, що полегшує його розуміння та роботу з ним.

Система маршрутизації в Laravel є ключовою особливістю, яка сприяє його надійності та гнучкості. Ось деякі подробиці про систему маршрутизації в Laravel:

– параметри маршруту та підстановочні знаки: Система маршрутизації Laravel підтримує параметри маршруту, що дозволяє вам визначати динамічні сегменти в ваших URL-адресах. Ці параметри можуть бути використані для перехоплення значень з URL і передачі їх як аргументи методам або закриттям контролерів. Крім того, Laravel надає можливості маршрутизації за допомогою підстановочних знаків, що дозволяє вам визначати універсальні маршрути або обробляти певні шаблони у ваших URL-адресах;

– іменування маршрутів та генерація URL-адрес: Laravel дозволяє присвоювати імена вашим маршрутам, що полегшує посилання на них у вашому додатку. Імена маршрутів забезпечують зручний спосіб генерувати URL-адреси, використовуючи іменовані маршрути замість жорсткого кодування URL-адрес у кодовій базі вашого додатку. Це покращує зручність супроводу і дозволяє гнучко змінювати URL-адреси;

– групи маршрутів і проміжне програмне забезпечення: Система маршрутизації Laravel підтримує групування маршрутів, що дозволяє вам групувати пов'язані маршрути і застосовувати до них спільні атрибути або проміжне програмне забезпечення. Це корисно для застосування автентифікації, авторизації або іншого проміжного програмного забезпечення до групи маршрутів. Групування маршрутів допомагає впорядкувати ваш код і зменшити кількість повторюваних елементів;

– кешування маршрутів: Laravel надає механізм кешування маршрутів, який значно покращує продуктивність вашого додатку. Кешуючи ваші маршрути, Laravel уникає накладних витрат на динамічне вирішення визначень маршрутів при кожному запиті, що призводить до прискорення часу відгуку;

– іменовані параметри та обмеження маршруту: Laravel дозволяє вам визначати іменовані параметри у ваших маршрутах, що полегшує створення URL-адрес, які включають певні значення параметрів. Ви також можете застосовувати обмеження до параметрів маршруту, визначаючи шаблони, яким повинен відповідати параметр. Це допомагає гарантувати, що ваші маршрути обробляють лише очікувані значення, підвищуючи безпеку і запобігаючи ненавмисним збігам маршрутів;

– прив'язка моделі маршруту: Система маршрутизації Laravel включає потужні можливості прив'язки моделей. Ця функція дозволяє вам автоматично вставляти екземпляри моделі безпосередньо в методи вашого контролера на основі параметрів маршруту. Це спрощує процес отримання та роботи з конкретними моделями у вашому додатку;

– загалом, система маршрутизації Laravel забезпечує гнучкий та інтуїтивно зрозумілий спосіб визначення та керування кінцевими точками вашого додатку. Вона пропонує такі функції, як чіткі визначення маршрутів, захоплення параметрів, іменування маршрутів, застосування проміжного програмного забезпечення, кешування маршрутів, іменовані параметри маршрутів, обмеження та прив'язка моделей. Ці функції полегшують обробку

різних типів запитів, покращують організацію коду та підвищують загальну продуктивність і зручність супроводу вашого додатку на Laravel;

– Laravel Eloquent ORM: Eloquent - це вбудована в Laravel система об'єктно-реляційного відображення (ORM). Вона спрощує роботу з базами даних, дозволяючи розробникам працювати з базами даних за допомогою інтуїтивно зрозумілого та виразного об'єктно-орієнтованого синтаксису. Eloquent забезпечує зручний спосіб взаємодії з базою даних, визначення зв'язків між моделями та виконання загальних завдань, пов'язаних з базами даних;

– модульна та розширювана архітектура: Laravel слідує архітектурному шаблону Model-View-Controller (MVC), який сприяє організації коду та розділенню завдань. Фреймворк розроблений за модульним принципом і дозволяє розробникам легко розширювати його функціональність за допомогою пакетів і бібліотек. Екосистема пакетів Laravel, відома як "Laravel Ecosystem", пропонує широкий спектр готових компонентів та інтеграцій, що економить час розробки;

– Laravel Forge та Envoyer: Laravel Forge - це потужний інструмент для управління серверами та розгортання, в той час як Laravel Envoyer спрощує процес розгортання додатків Laravel. Ці інструменти легко інтегруються з Laravel, пропонуючи зручні та ефективні робочі процеси налаштування, конфігурації та розгортання серверів;

– інструменти тестування та налагодження: Laravel надає надійний набір інструментів тестування, включаючи PHPUnit, для полегшення модульного тестування та забезпечення якості кодової бази вашого додатку. Крім того, фреймворк пропонує функції налагодження та обробки помилок, що полегшує виявлення та вирішення проблем під час розробки.

Загалом, позитивні сторони Laravel полягають в елегантному синтаксисі, надійній системі маршрутизації, можливостях ORM, модульній архітектурі, потужній підтримці спільноти, зручних інструментах розгортання та функціях тестування. Ці сильні сторони сприяють популярності Laravel і роблять його чудовим вибором для створення масштабованих і підтримуваних веб-додатків.

1.2 Аналіз наявного програмно-технічного забезпечення предметної області

Laravel – це PHP-фреймворк, тому доречніше порівнювати його з іншими PHP-фреймворками, а не з самими мовами програмування. Проте, я можу надати коротке порівняння PHP (мови, що використовується в Laravel) з деякими іншими популярними мовами програмування:

PHP та Python:

– PHP в першу чергу призначений для веб-розробки і широко використовується для написання сценаріїв на стороні сервера. Python, з іншого боку, є універсальною мовою, яка використовується в різних сферах, включаючи веб-розробку, аналіз даних, машинне навчання та написання сценаріїв;

– PHP має більшу частку у веб-розробці завдяки широкому використанню в популярних системах управління контентом (CMS), таких як WordPress. Python відома своєю простотою та читабельністю, що робить її кращим вибором для завдань, які вимагають чистого та лаконічного коду.

Обидві мови мають багату екосистему фреймворків та бібліотек. Laravel – це високо оцінений фреймворк для PHP, в той час як Django і Flask – популярні фреймворки для веб-розробки на Python.

PHP та Ruby:

– PHP та Ruby є популярними мовами для веб-розробки. PHP відома своїм широким розповсюдженням і зазвичай використовується для створення динамічних веб-додатків. Ruby, з іншого боку, високо цінується за свій елегантний синтаксис і зручні для розробників домовленості;

– PHP має більшу спільноту і величезну кількість доступних бібліотек та фреймворків. Laravel є відомим фреймворком PHP, тоді як Ruby on Rails (RoR) є широко використовуваним веб-фреймворком, побудованим на Ruby;

– Ruby часто надають перевагу за його акцент на щастя розробника, оскільки він заохочує до чистого та виразного коду. PHP має більшу частку ринку і довгу історію в індустрії веб-розробки.

PHP та JavaScript:

– PHP - це мова сценаріїв на стороні сервера, в той час як JavaScript в основному використовується для написання сценаріїв на стороні клієнта у веб-браузерах. JavaScript також набув популярності на стороні сервера з появою таких фреймворків, як Node.js;

– PHP і JavaScript мають різні варіанти використання. PHP зазвичай використовується для обробки серверної логіки, взаємодії з базами даних і генерації динамічного веб-контенту. JavaScript, з іншого боку, використовується для підвищення інтерактивності на стороні клієнта і створення складних веб-додатків;

– Laravel, як PHP-фреймворк, забезпечує надійну екосистему для серверної веб-розробки. JavaScript має власні фреймворки, такі як Express.js та React.js для серверної та клієнтської розробки відповідно.

Важливо зазначити, що вибір мови програмування та фреймворку залежить від таких факторів, як вимоги проекту, досвід розробника, підтримка спільноти та зрілість екосистеми. Кожна мова та фреймворк мають свої сильні та слабкі сторони, і вибір повинен ґрунтуватися на конкретних потребах проекту.

На сьогоднішній день існує багато схожих платформ для обміну інформаціює, такі як:

FEX.NET – це сервіс обміну файлами, який дозволяє зберігати та обмінюватись файлами (файлообмінник). Ви можете безкоштовно зберігати файли на платформі протягом 7 днів з моменту їх завантаження без реєстрації, після чого файли видаляються системою автоматично і не підлягають відновленню.

Серед переваг:

– швидко передавати файли або папки;

– отримати доступ до всіх ваших файлів з різних пристроїв;

– збереження анонімності під час передачі файлів.

Серед недоліків:

– для того, щоб збільшити об'єм пам'яті та зберігати файли довше ніж 7 днів, необхідно зареєструвати обліковий запис і оплатити послуги зберігання.

GitHub.com – розподілена система контролю версій, яка дає можливість розробникам відстежувати зміни в файлах і працювати спільно з іншими розробниками. Звичайно ця система основний потенціал має як система контролю версій (СКВ, VCS, Version Control Systems), але у даному випадку ми її будемо розглядати як звичайний файлообмінник.

Серед переваг:

– невеликий і швидкий;

– він виконує всі операції локально, що збільшує його швидкість;

– крім того, Git локально зберігає весь репозиторій в невеликий файл без втрати якості даних.

Резервне копіювання:

– Git ефективний в зберіганні бекапів, тому відомо мало випадків, коли хтось втрачав дані при використанні Git.

1.3 Визначення вимог до програмного продукту

За підсумками детального достідження предметної області та існуючих схожих додатків, було розроблене технічне завдання та список фукціональних та не функціональних вимог.

Завданням кваліфікаційної роботи є створення веб-платформи “LabHub” для обміну інформацією на базі фреймворка Laravel. Основною метою кваліфікаційної роботи є створення веб-платформи для покращеного пошуку та обміну інформацією, реалізованою за допомогою бартеру між користувачами.

Користуючись застосунком можливо не лише знайти потрібну та корисну інформацію, а також опублікувати для обміну й свої роботи, дослідження, експеременти, творчість та багато іншого. За допомогою даної платформи користувач зможе заощадити велику кількість часу та сил, яку б він витратив для пошуку потрібної йому інформації перевіряючи безліч інших джерел самостійно.

Для досягнення цих цілей необхідно побудувати послідовну систему в якій користувачі матимуть змогу зберігати, переглядати, купувати та продавати інформацію, потрібно забезпечити зберігання даних про інформацію та користувачів в базу даних. Також потрібно реалізувати декілька рівнів адміністрування користувачів:

– гість (не авторизований користувач або guest);

– авторизований користувач;

– заблокований користувач;

– адміністратор.

Функціонал адміністратора передбачатиме:

– перегляд та пошук за певними критеріями опублікованих файлів користувачів з нищим рівнем доступом ніж у адміністатора;

– видалення опублікованих файлів користувачів;

– перегляд та пошук за датою проведення загального журналу транзакцій;

– перегляд та пошук авторизованих користувачів, які пройшли повний етап авторизації, можливість надання даним користувачам найнищих прав доступу, а саме «Ban», але залишивши їм лише можливість авторизуватись та зв’язатися з адміністрацією веб-платформи. Перегляд деальної інформації про вибраного користувача, а саме список проведених ним транзакцій за весь період часу, а також начислення або списання коштів з рахунку вибраного користувача;

– створення та видалення нових навчальних предметів;

– створення та видалення нових категорій, які будуть використовуватись під час опублікування, авторизованим, користувачем його інформації;

– перегляд отриманих повідомлень від авторизованих користувачів, з можливістю надання цим повідомлення відмітки про виконання, а також видалення повідомлень, які вже були відміченні як «Виконаний».

Функціонал авторизованого користувача передбачатиме:

– перегляд наявних в нього умовних одиниць коштів;

– перегляд та фільтрація опублікованих файлів іншими користувачами;

– створення та відправлення повідомлення адміністраторам;

– створення та видалення власних публікацій з певною інформацією, фільтрація та пошук серед списку опублікованих матеріалів даного авторизованого користувача, перегляд замовлень до кожного опублікованого файлу, з подальшою можливістю підтвредженням або відхиленням їх;

– створення та видалення власних замовлень, прикріплених до публікацій інших авторизованих користувачів, пошук та фільтрація серед загального списку доданих замовлень даного користувача;

– перегляд та пошук за датою проведення журналу транзакцій, даного авторизованого користувача.

Функціонал авторизованого користувача (заблокований користувач), який отримав «Ban» (блок доступу), за порушення прав користування, має передбачати:

– зв’язок з адміністрацією.

Функціонал для не авторизованого користувача (гість) має передбачати:

– можливість авторизації або регістрації.

Для того, щоб чітко сформувати вимоги до програмного забезпечення використовується уніфікована мова моделювання UML.

Уніфікована мова моделювання (UML) використовується для чіткого визначення вимог до програмного забезпечення. Діаграми UML створюються з метою візуального представлення системи з усіма її компонентами, такими як ролі, актори, класи та дії, для забезпечення повного розуміння або

документування системи. Вони використовують графічні символи для зображення різних елементів системи та їх взаємозв'язків.

Діаграми UML є стандартом в галузі розробки програмного забезпечення і дозволяють зрозуміти, аналізувати та проектувати системи з використанням єдиної мови. Вони включають різні типи діаграм, такі як діаграми класів, послідовностей, діяльності, станів, компонентів і багато інших. Кожен тип діаграми використовується для моделювання певних аспектів системи і надає зручний спосіб візуалізації та аналізу системи.

Діаграми UML є корисним інструментом для комунікації між розробниками, архітекторами та іншими зацікавленими сторонами під час розробки програмного забезпечення. Вони допомагають уникнути непорозумінь, уточнити вимоги до системи та забезпечити ефективну розробку і впровадження.

Складемо описи акторів (користувачів системи) та усіх можливих їхніх дій (варіантів використання).

У таблиці 1.1 наведені актори розроблюваного програмного засобу.

Таблиця 1.1 – Опис рівнів адміністрування розроблюваного ПЗ

Актор	Короткий опис
Незареєстрований користувач (гість)	Зможе зареєструватись або авторизуватись.
Зареєстрований користувач	Зможе переглядати, зробити пошук по певним критеріям, створювати, купувати та продавати інформацію.
Адміністратор	Зможе створювати нові критерії для пошуку, нараховувати грошові одиниці, видаляти публікації користувачів, блокувати користувачів.
Заблокований користувач	Зможе зв’язатися з адміністрацією.

Визначивши акторів системи та варіанти використання, було побудовано діаграму варіантів використання (рисунок 1.1).

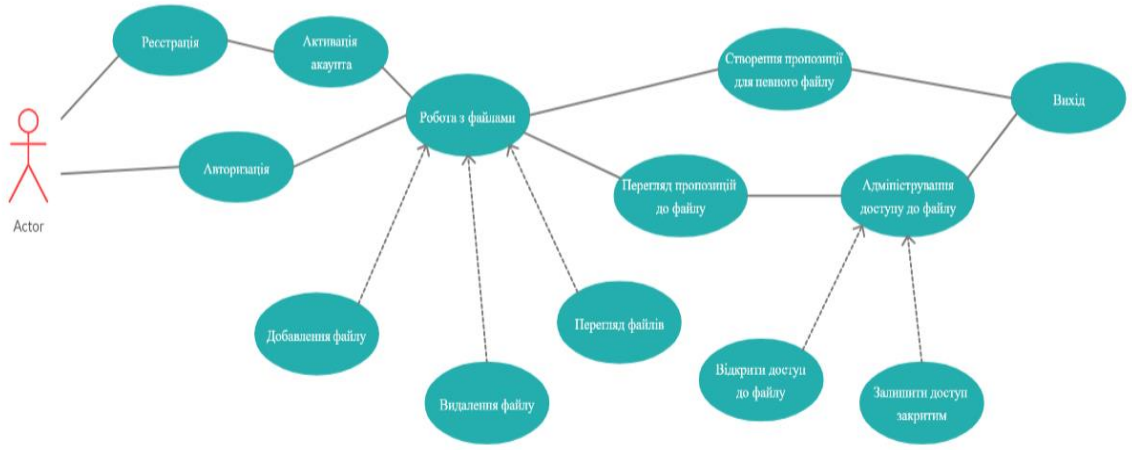


Рисунок 1.1 – Діаграма використання

Після завершення аналізу вимог до програмного продукту, було розроблено технічне завдання, яке можна знайти у додатку А. Це технічне завдання служило основою для подальшої розробки веб-платформи.

У цьому розділі був проведений детальний аналіз особливостей предметної області. Під час аналізу були виявлені проблеми, що існують у даній сфері, а також запропоновані шляхи їх вирішення. Виявлення цих проблем є важливим кроком у процесі розробки, оскільки воно дозволяє зрозуміти потреби та вимоги користувачів, а також виявити можливі обмеження та виклики, з якими може зіткнутися проект.

Після аналізу проблем був проведений огляд наявних програмних засобів, що використовуються в даній області. Цей огляд спрямовувався на виявлення та оцінку існуючих рішень, а також їх переваг і недоліків. Цей етап допоміг з'ясувати, які функції вже реалізовані і які проблеми можуть виникнути при використанні існуючих програмних засобів.

Крім того, у даному розділі був складений список функціональних та нефункціональних вимог до розроблюваного проекту. Функціональні вимоги

визначають, які функції та можливості повинен мати проект, щоб задовольнити потреби користувачів. Нефункціональні вимоги стосуються якості та характеристик проекту, таких як продуктивність, безпека, надійність тощо. Ці вимоги грають важливу роль у розробці, оскільки вони визначають критерії, за якими буде оцінюватися успішність та якість розробленого програмного продукту.

1.4 Постановка задачі

Після аналізу вимог до програмного продукту було створено технічне завдання, яке можна знайти у додатку А. Це технічне завдання буде використовуватись як основа для подальшої розробки кваліфікаційної роботи “Веб-платформа ‘LabHub’ для обміну інформацією на базі фреймворку Laravel”.

Під час розробки цієї веб-платформи потрібно вирішити наступні завдання:

– створити простий та привабливий інтерфейс користувача, що забезпечує зручне використання веб-застосунку;

– розробити функціонал, що дозволить заповнювати веб-платформу даними про користувачів та інформацію, яка може бути продана або придбана;

– забезпечити можливість зворотного зв'язку з адміністратором через веб-платформу;

– розробити функціонал, що дозволить керувати фільтрами для поліпшеного пошуку інформації.

2 ПРОЕКТУВАННЯ ВЕБ-ПЛАТФОРМИ

2.1 Аналіз та вибір архітектури веб-платформи

Аналіз та вибір архітектури є важливою складовою процесу розробки веб-застосунків. При проектуванні веб-застосунків виникає потреба у виборі оптимальної архітектури, яка задовольняє вимоги проекту, забезпечує потрібну функціональність та масштабованість та потрібно визначити оптимальні засоби комунікації між клієнтом та сервером, а також інфраструктуру, на якій буде працювати веб-застосунок. Також це дозволяє розробникам визначити найкращу архітектуру для свого проекту, враховуючи його вимоги, функціональність, масштабованість та безпеку. Цей процес є ключовим у створенні успішного веб-застосунка, який задовольнятиме потреби користувачів та бізнесу.

Аналіз та вибір архітектури веб-застосунка передбачає наступні кроки:

– збір вимог – перший крок - це збір та аналіз вимог до веб-застосунка. Розробник повинен зрозуміти, які функції та можливості має мати додаток, які є його основні цілі та вимоги до продуктивності;

– аналіз даних – необхідно проаналізувати типи даних, з якими буде працювати веб-застосунок, а також оцінити їх обсяг та швидкодію доступу. Це допоможе визначити, які технології та бази даних будуть найбільш підходящими для проекту;

– вибір архітектурного стилю – необхідно здійснити вибір архітектурного стилю для веб-застосунка. Існує багато різних стилів, таких як клієнт-сервер, шарова архітектура, мікросервісна архітектура тощо. Вибір стилю залежить від вимог проекту та його масштабу;

– розподілені системи – якщо веб-застосунок передбачає використання розподілених систем, необхідно вирішити питання їх організації та взаємодії. Це можуть бути мікросервіси, контейнеризація, оркестрація контейнерів, обробка повідомлень тощо;

– протоколи – веб-застосунки використовують різні протоколи для обміну даними між клієнтом та сервером. Найбільш поширеним протоколом є HTTP (Hypertext Transfer Protocol), який використовується для передачі даних у веб-середовищі. Зазвичай використовується протокол HTTP або його безпечна версія HTTPS для забезпечення шифрування та безпеки під час передачі даних;

– серверна інфраструктура – вибір серверної інфраструктури залежить від потреб вашого веб-застосунка. Одна з основних роздільних ліній - це вибір між традиційними фізичними серверами або хмарними платформами. Фізичні сервери можуть бути розташовані в вашому власному дата-центрі або в орендованому обліковому записі. З іншого боку, хмарні платформи, такі як Amazon Web Services (AWS), Google Cloud або Microsoft Azure, надають інфраструктуру в хмарі, що дозволяє масштабувати ресурси веб-застосунку залежно від його потреб.

При виборі серверної інфраструктури також слід враховувати такі фактори, як масштабованість, надійність, продуктивність та безпека. Наприклад, великі веб-застосунки з високим навантаженням можуть вимагати розподіленої системи з використанням балансування навантаження та масштабування горизонтально. З іншого боку, менші проекти можуть бути задоволені використанням віртуальних серверів або контейнеризації для забезпечення ізоляції ресурсів та швидкого впровадження.

Крім того, варто розглянути аспекти безпеки, такі як захист від DDoS-атак, файрволи та механізми автентифікації, коли розглядається вибір серверної інфраструктури. Важливо пам'ятати, що безпека є одним з найважливіших аспектів веб-застосунків.

Остаточний вибір архітектури, протоколів та серверної інфраструктури залежить від конкретних вимог вашого проекту, його масштабу, обсягу даних та очікуваної кількості користувачів. Це процес, що вимагає виважених рішень та збалансованого підходу для досягнення найкращих результатів.

– вибір технологій – після визначення архітектури веб-застосунка необхідно вибрати підходящі технології та інструменти для його реалізації. Це

можуть бути мови програмування, фреймворки, бази даних, сервери додатків та інші компоненти;

– масштабованість – розробник повинен врахувати можливості масштабування веб-застосунка. Наприклад, якщо очікується зростання навантаження, може бути доцільно використовувати горизонтальне масштабування або розподілені системи для підвищення продуктивності;

– безпека – при проектуванні архітектури веб-застосунка необхідно приділити увагу питанням безпеки. Захист від атак, обробка даних, автентифікація та авторизація користувачів - це лише кілька аспектів, які варто врахувати;

– тестування та оцінка – вибрану архітектуру потрібно протестувати та оцінити її ефективність. Це можна зробити шляхом проведення тестів продуктивності, навантаження та інших видів тестування.

Для розробки автоматизованого веб-сайту використаємо клієнт-серверну архітектуру ПЗ. Архітектура клієнт-сервер є одним із архітектурних шаблонів програмного забезпечення та є домінуючою концепцією у створенні розподілених мережних застосунків і передбачає взаємодію та обмін даними між ними.

Серверна частина системи може розміщуватись на хмарному хостингу, або на фізичному сервері. Клієнтами виступатимуть веб-переглядачі, браузери, за допомогою яких менеджери та викладачі взаємодіятимуть з системою.

Для забезпечення зручності розробки, тестування та подальшої підтримки, покращення та легкого масштабування ПЗ, слід розділити програмну реалізацію на шари, які будуть виконувати певні функції.

Існує декілька підходів та архітектурних шаблонів для вирішення проблеми поділу системи на частини. Одним із таких підходів є архітектурне рішення, яке складається з таких частин:

– модель;

– представлення;

– сервіс;

– репозиторій.

Як можна зрозуміти по переліку – це архітектурний шаблон Spring MVC або Model View Controller, де Model – це сервіс, модель і репозиторій, View – представлення (наші майбутні сторінки) і Controller – контроллер.

Модель – це компонент, який містить дані об’єкта предметної області. Цей компонент часто слугує відображенням таблиць БД в об’єктну модель. Модель не виконує ніякої обробки даних, роботи з БД чи відображенням.

Сервіс – це компонент, який відповідає за бізнес-логіку сайту. В ньому виконується основна обробка та перетворення даних, містяться правила, принципи та логіка роботи з даними (моделлю).

Репозиторій – це шар роботи з базою данних. Він відповідає за збереження, оновлення, видалення та діставання інформації з бази даних. Інкапсулює реалізацію бази даних і виступає інтерфейсом для зв’язку з нею. Виконує перетворення таблиць БД в об’єкти та навпаки.

Представлення – це компонент програми, який відповідає за візуальне відображення даних, клієнтську частину сайту. Одночасно можуть співіснувати кілька виглядів (представлень) однієї і тієї ж інформації, наприклад розклад конкретного викладача, та загальний розклад для менеджера.

Представлення є сторінкою. Візуальний елемент - це сторінка, що відмальовує себе на екрані користувача, а керуючий елемент реагує на взаємодію користувача з візуальною частиною, його можна назвати простіше - контроллер.

Згідно цієї архітектури, користувач взаємодіє з візуальною частиною представлення, наприклад натискаючи на графічні елементи інтерфейсу. Контроллер реагує на ці події, за допомогою шару сервісів обробляє необхідні дані, та оновлює графічний інтерфейс. Сервіси виконують основні обчислення та логічні рішення в системі, використовуючи репозиторії для роботи з БД. Моделі використовуються для передачі даних між іншими архітектурними шарами.

Схема наведеної архітектури зображена на рисунку 2.1.

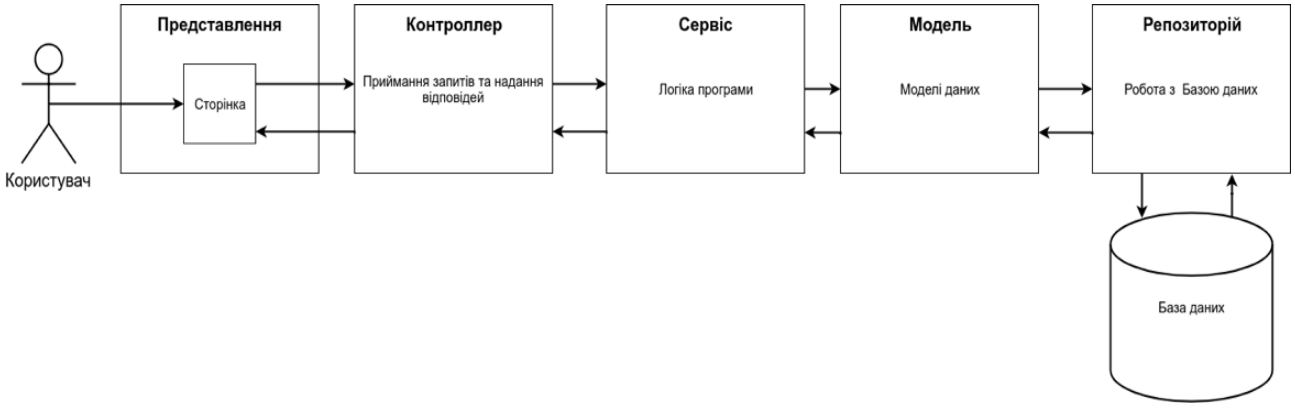


Рисунок 2.1 – Архітектура програми

2.2 Опис структури даних та моделі бази даних

База даних (БД) - це структурована колекція даних, яка організована з метою ефективного зберігання, керування та доступу до інформації. Бази даних використовуються для обробки великих обсягів даних, дозволяючи легко здійснювати їх оновлення, модифікацію та вилучення за допомогою відповідних запитів.

Існує багато різноманітних типів баз даних, кожен з яких має свої унікальні особливості та використання, але для кваліфікаційної роботи було розглянуто декілька: реляційна база даних та не реляційна база даних.

Реляційна база даних (РБД), ця база даних є найпоширенішим типом баз даних. У РБД дані організовані у вигляді таблиць з рядками і стовпцями. Кожен рядок в таблиці представляє кортеж або запис, а стовпці відображають атрибути або характеристики цих записів. Реляційні бази даних використовують мову структурованих запитів (SQL) для керування даними. Вони забезпечують гнучкість, надійність та ефективність при зберіганні та обробці даних. Основні переваги РБД включають:

– стандартизація – реляційні бази даних підтримують загальноприйняті стандарти, такі як SQL, що спрощує розробку та обмін даними між різними системами;

– інтегрітет даних – реляційні бази даних надають можливість встановлювати обмеження цілісності даних, що дозволяє забезпечити правильність та унікальність даних;

– запити – SQL дозволяє виконувати складні запити до бази даних, що дає змогу отримувати потрібну інформацію швидко та ефективно;

– нормалізація – реляційні бази даних можуть бути нормалізовані, що сприяє ефективному зберіганню та управлінню даними, зменшуючи дублікацію та забезпечуючи консистентність.

Нормалізація є процесом проектування реляційних баз даних з метою зменшення дублікації даних та забезпечення їх консистентності. Цей процес включає розбиття бази даних на менші, більш організовані таблиці, які пов'язані між собою за допомогою зв'язків. Нормалізація бази даних допомагає забезпечити цілісність даних, спрощує їх зміну та підтримку, а також покращує ефективність запитів.

У нормалізації використовуються так звані нормальні форми, які визначають рівень організації та нормалізації даних. Існує кілька рівнів нормальних форм, від першої нормальної форми (1NF) до п'ятої нормальної форми (5NF). Кожна наступна нормальна форма вимагає додаткових умов для забезпечення нормалізованості бази даних.

Основні нормальні форми включають:

– перша нормальна форма (1NF) – вимагає, щоб кожен атрибут у таблиці мав атомарні значення, тобто значення не можуть бути розбиті на менші частини. Кожен стовпчик повинен містити лише одне значення;

– друга нормальна форма (2NF) – вимагає, щоб кожен неключовий атрибут у таблиці пов'язаний лише з унікальними значеннями первинного ключа. Таким чином, усі атрибути, що не залежать від всього первинного ключа, виносяться в окрему таблицю;

– третя нормальна форма (3NF) – Вимагає, щоб кожен неключовий атрибут у таблиці пов'язаний лише з первинним ключем, а не з іншими

неключовими атрибутами. Таким чином, виносяться атрибути, що залежать від інших неключових атрибутів, в окремі таблиці;

– четверта нормальна форма (4NF) – вимагає, щоб у таблиці не було множинних залежностей, коли один неключовий атрибут залежить від групи інших неключових атрибутів. Це досягається розбиттям таких залежностей на окремі таблиці;

– п'ята нормальна форма (5NF) – вимагає, щоб в таблиці не було залежностей між множинними зв'язками. Це досягається розбиттям таблиць, що мають зв'язки між собою, на окремі таблиці.

Кожна наступна нормальна форма дозволяє забезпечити більшу організованість та незалежність даних у базі даних, але може вимагати більше ресурсів для зберігання та обробки даних. При проектуванні бази даних важливо враховувати вимоги до даних та бізнес-логіку для вибору найвідповідніших нормальних форм для конкретного випадку.

Однак, реляційні бази даних також мають деякі недоліки:

– складність схеми – проектування структури реляційної бази даних може бути складним завданням, особливо для великих проектів;

– висока накладність – реляційні бази даних можуть бути менш ефективними при опрацюванні деяких типів даних або виконанні складних операцій.

Не реляційна база даних (NoSQL): Не реляційна база даних, також відома як NoSQL, використовується для зберігання та керування великими обсягами структурованих та неструктурованих даних. У NoSQL базах даних використовуються різні моделі, такі як ключ-значення, стовпчаста, документоорієнтована або графова модель, замість традиційних таблиць. NoSQL бази даних надають високу швидкодію, масштабованість та гнучкість в опрацюванні даних. Деякі переваги NoSQL баз даних включають:

– гнучкість – NoSQL бази даних дозволяють зберігати інформацію в різних форматах, що спрощує роботу з неструктурованими даними;

– масштабованість – NoSQL бази даних можуть легко масштабуватись горизонтально, що дозволяє обробляти великі обсяги даних та забезпечувати високу швидкодію;

– простота розробки – у NoSQL базах даних не потрібно строго дотримуватись заздалегідь заданої схеми, що дозволяє швидше розробляти та впроваджувати додатки.

Однак, NoSQL бази даних також мають свої недоліки:

– відсутність стандартів – у NoSQL базах даних відсутні загальноприйняті стандарти, що може ускладнити розробку та обмін даними між різними системами;

– обмеженість запитів – в порівнянні з реляційними базами даних, NoSQL бази даних можуть мати обмежені можливості для виконання складних запитів та аналітики.

Після аналізування переваг і недоліків різних типів баз даних, було прийнято рішення використовувати реляційну базу даних MySQL для розробки кваліфікаційної роботи. З урахуванням попередньо створеної діаграми варіантів використання (див. рисунок 1.1), ми визначили набір даних, які будуть зберігатися в базі даних. Це призвело до створення багатьох колекцій даних і побудови структури бази даних (див. рисунок 2.2).

В результаті, було отримано значну кількість полів, які є необхідними для належної функціональності розроблюваного веб-сайту. Кожна таблиця має основний ключ, атрибути є атомарними, повторення груп даних не виявлено, а отже спроектована БД відповідає I нормальній формі. Таким чином, була створена логічна модель бази даних, і на основі цієї моделі можна перейти до проектування фізичної моделі.

Рисунок 2.2 – Структура бази даних

2.3 Проектування серверної частини веб-платформи

PHP – це одна з найпопулярніших і вимаганих мов програмування. Вона широко використовується у великих проектах, таких як Facebook, а також є основою для популярних систем управління контентом, зокрема WordPress. Близько третини всіх сайтів в Інтернеті і приблизно 60% сайтів на системах управління контентом працюють на PHP.

Незважаючи на стабільність PHP, в більшості випадків розробка ведеться з використанням фреймворків з декількох причин:

– фреймворки прискорюють розробку, оскільки надають базові функції для роботи з базами даних, такі як створення, читання, оновлення та видалення записів. Це усуває необхідність вручну писати SQL-запити до бази даних;

– додатки, написані на фреймворках, легко масштабуються, тобто їх можна легко розширювати та адаптувати до зростаючих потреб;

– розробка проектів на фреймворках є зручнішою, ніж на чистому PHP, оскільки код фреймворків зазвичай має лаконічну структуру, що полегшує роботу розробникам;

– PHP-фреймворки використовують модель MVC (Model-View-Controller), яка спрощує розробку, розділяючи логіку додатку на модель (дані), представлення (вигляд) та контролер (логіка обробки запитів);

– додатки, створені на фреймворках, зазвичай краще захищені від потенційних загроз безпеки порівняно з додатками, написаними на чистому PHP;

– фреймворки використовують принцип DRY (Don't Repeat Yourself), що дозволяє розробникам писати менше коду, оскільки часто використовувані функції та компоненти можна перевикористовувати.

Для створення веб-сервісу "LabHub" був використаний один з найпопулярніших фреймворків PHP - Laravel. Вибір цього фреймворка був обґрунтований численними перевагами:

– Laravel має шаблонізатор Blade, який спрощує створення фронтенду додатки шляхом автоматичного екранування HTML і дозволяє використовувати код PHP прямо в шаблонах. Шаблони зберігаються в окремих файлах з розширенням .blade.php і перетворюються в нативний PHP, що майже не впливає на швидкість роботи додатків;

– Laravel має різні варіанти установки і може бути легко встановлений за допомогою пакетного менеджера Composer;

– фреймворк підтримує різні бази даних, такі як Microsoft BI, MySQL, MongoDB, PostgreSQL, Redis та SQLite;

– Laravel має широкі можливості розширення через використання пакетів. Існує близько 9000 пакетів для Laravel у каталозі Packalyst, що дозволяє розширювати функціональність фреймворка.

2.4 Проектування інтерфейсу користувача

Інтерфейс системи повинен бути простим, легким у сприйнятті, зручним у використанні, функціональним і привабливим з естетично приємним виглядом. Користувач повинен легко зорієнтуватись у програмі і знати, де знаходиться кожна функція системи і як її викликати.

Для створення графічного інтерфейсу ми використовуватимемо спеціально розроблений дизайн сторінки входу. При відкритті програми, спочатку з'явиться форма привітання, де користувач буде мати можливість обрати між формою реєстрації або авторизації.

Початкову веб-сторінку, яку користувач має побачити після переходу по посиланню на веб-платформу – це сторінка Welcome (рис. 2.3) яка має містити наступні елементи:

– логотип;
– кнопка “Login” (кнопка авторизації);
– кнопка “Register” (кнопка реєстрації).

Рисунок 2.3 – Вигляд сторінки Welcome

Після того як користувач авторизувався або створив новий акаунт, він має перейти на сторінку Home (рис. 2.4), яка має містити наступні елементи:

– верхня частину веб-сторінки (header);

– кнопка для фільтрації інформації.

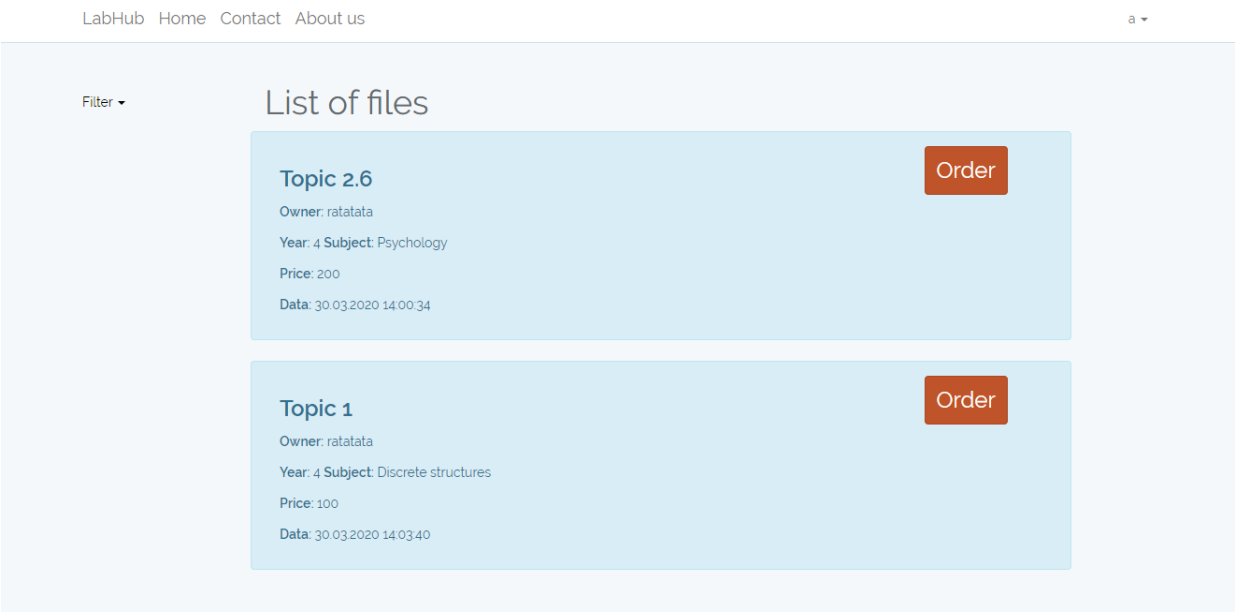


Рисунок 2.4 – Вигляд сторінки Home

Також потрібно описати верхню частину веб-сторінки (рис. 2.5), яка має складатися з наступних елементів:

– кнопка для переходу на сторінку Home;

– кнопка для переходу на сторінку Welcome;

– кнопка для переходу на сторінку Contact;

– кнопка для переходу на сторінку About us.

– кнопка для керування профілем.

Рисунок 2.5 – Вигляд верхньої частини веб-сторінки

Профіль користувача (рис. 2.6) має складатися з наступних елементів:

– верхня частина веб-сторінки (header);
– кнопка для переходу на сторінку My Files;
– кнопка для переходу на сторінку My orders;
– кнопка для переходу на сторінку MyTransactions;
– список інформації, яку користувач вже опублікував;
– кнопка для фільтрації інформації по статусу;
– поле для пошуку інформації.

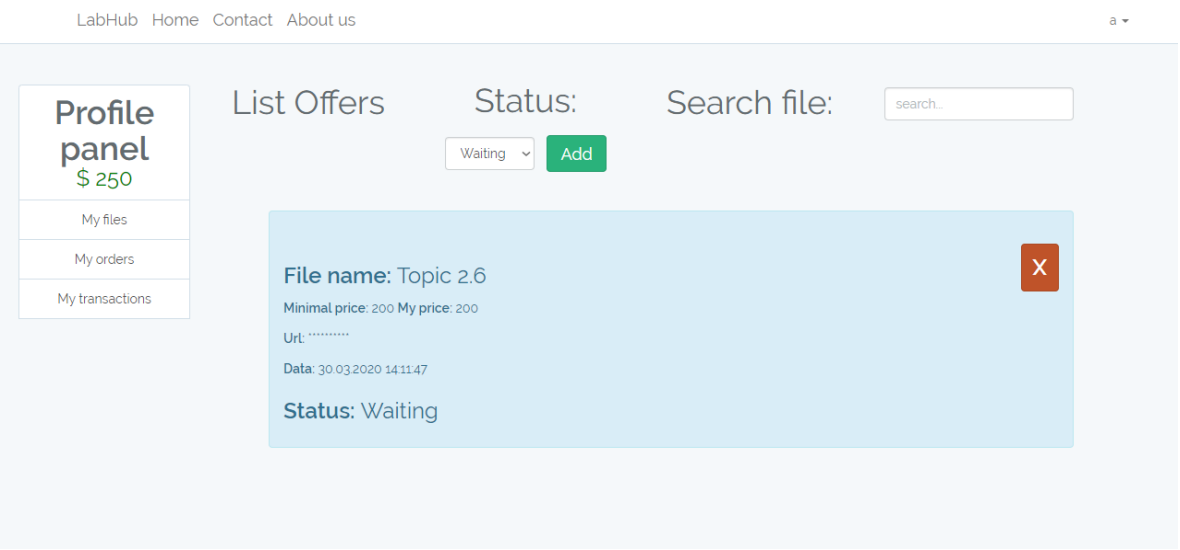


Рисунок 2.6 – Вигляд сторінки My Orders

Користувач в разі необхідності може звернутися до адміністраторів натиснувши на вкладку Contact (рис. 2.7), яка має містити наступні елементи:

– верхня частина веб-сторінки (header);
– поле для заголовку повідомлення;
– поле для повідомлення;
– кнопка для відправлення повідомлення.

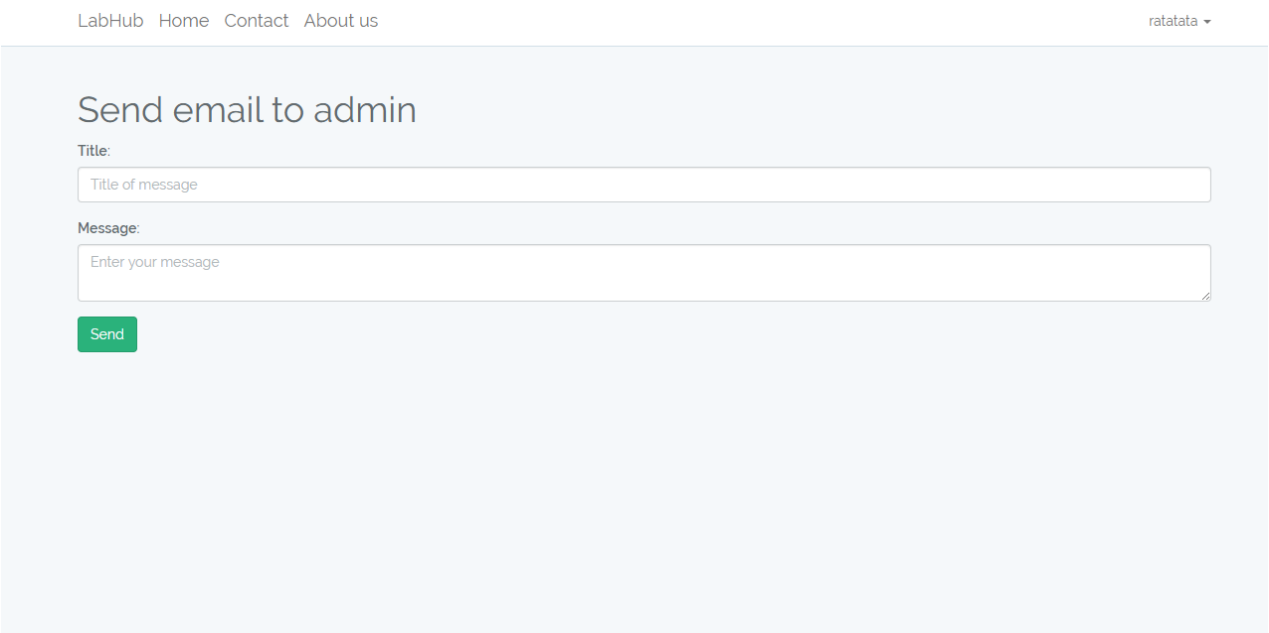


Рисунок 2.7 – Вигляд сторінки Contact

Профіль адміністратор (рис. 2.8) має містити наступну елементи:

– верхня частина веб-сторінки (header);

– кнопка для переходу на сторінку Files, яка має містити список всієї інформації, яка була опублікована;

– кнопка для переходу на сторінку Transactions, яка має містити список всіх транзакцій;

– кнопка для переходу на сторінку Users, яка має містити список всіх користувачів;

– кнопка для переходу на сторінку Subjects, яка має містити список всіх предметів, які використовуватимуться для фільтрації інформації;

– кнопка для переходу на сторінку Categories, яка має містити список всіх категорій які використуватимуться для фільтріції інформації;

– кнопка для переходу на сторінку Message, яка має містити список всіх повідомлень, які користувачі відправили адміністратору;

– кнопка для переходу на сторінку Ban List, яка має містити список всіх заблокованих користувчів.

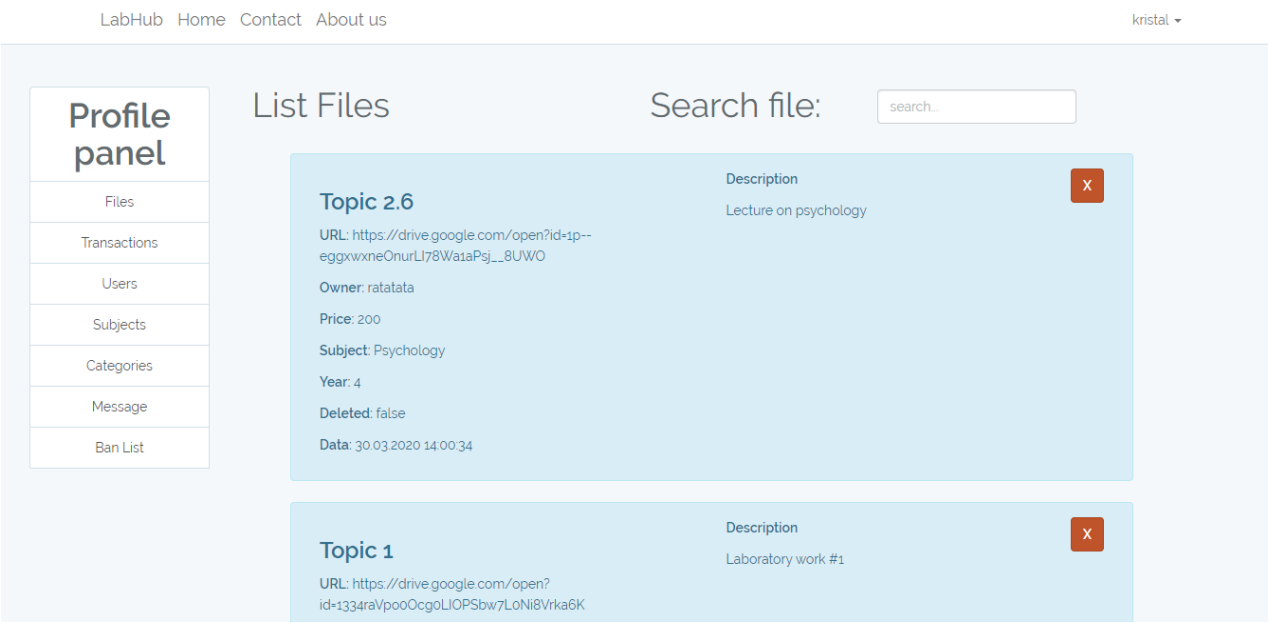


Рисунок 2.8 – Вигляд сторінки Files

2.5 Аналіз та вибір технологій і методів реалізації веб-платформи

У сучасній розробці веб-застосунків необхідно використовувати сучасні методи програмування. Однією з основних мов програмування для розробки веб-застосунків є PHP, яка має найбільшу популярність у всьому світі. Загалом використовуються різні фреймворки в яких за основу взята мова програмування PHP.

Для реалізації кваліфікаційної роботи був обраний фреймворк Laravel, HTML та Bootstrap.

HTML (HyperText Markup Language) є основною мовою розмітки для створення веб-сторінок. Вона використовується для визначення структури та семантики контенту на веб-сторінці. HTML складається з тегів, які вкладаються один в одного і визначають різні елементи сторінки, такі як заголовки, абзаци, списки, таблиці, посилання, зображення та багато іншого.

Переваги HTML:

– простота вивчення. HTML має простий синтаксис, і вивчити його основи досить легко. Це дозволяє швидко створювати прості веб-сторінки;

– семантика. HTML надає широкий спектр тегів, які дозволяють визначати семантику елементів сторінки. Наприклад, ви можете використовувати теги `<header>`, `<nav>`, `<article>` тощо для кращого розуміння структури сторінки пошуковими системами та іншими програмами;

– кросс-платформена підтримка. Браузери підтримують виконання HTML-коду на різних платформах, що дозволяє створювати веб-сторінки, які будуть працювати на різних пристроях, таких як комп'ютери, смартфони та планшети.

Bootstrap – це потужний фреймворк для розробки веб-інтерфейсів. Він забезпечує набір готових стилів, компонентів та скриптів JavaScript, які допомагають швидше та ефективніше створювати сучасні та респонсивні веб-сторінки.

Переваги Bootstrap:

– респонсивний дизайн. Bootstrap має вбудовану підтримку для респонсивного дизайну, що дозволяє сторінці адаптуватися до різних розмірів екранів. Це особливо важливо в епоху мобільних пристроїв, коли люди відвідують веб-сайти з різних пристроїв;

– масштабованість. Bootstrap дозволяє легко масштабувати ваші стилі та компоненти. Ви можете вибрати лише ті компоненти, які вам потрібні, або використовувати весь набір інструментів. Це спрощує розробку та підтримку веб-сайтів;

– готові стилі та компоненти. Bootstrap надає велику кількість готових стилів, таких як кнопки, форми, таблиці, навігація та багато іншого. Це дозволяє розробникам швидше створювати стильні веб-сторінки без необхідності писати багато власного CSS-коду.

Недоліки HTML та Bootstrap:

– однообразність. Часте використання Bootstrap може призвести до того, що ваші веб-сайти можуть виглядати схожими на інші сайти, які також використовують цей фреймворк. Це може зменшити унікальність вашого дизайну;

– зависимість від фреймворка. Використання Bootstrap може створити залежність від фреймворка, особливо якщо ви використовуєте його для всіх аспектів дизайну свого веб-сайту. Це може ускладнити перехід до іншого фреймворку або власного дизайну в майбутньому;

– розмір. Bootstrap - це великий фреймворк, і включення всього коду може призвести до збільшення розміру веб-сторінки та затримок при завантаженні. Це особливо важливо на мобільних пристроях з обмеженими швидкостями Інтернету.

Висновки до розділу

У цьому розділі було здійснено проектування системи, вибрано оптимальний варіант архітектури, що відповідає предметній області, і обрано технології та методи для реалізації веб-платформи.

Задовольняючи вимоги та поставлені завдання, був розроблений інтерфейс користувача, спроектована структура бази даних, а також розроблено архітектуру системи та її компоненти для подальшої розробки модулів, таких як:

– система наповнення веб-платформи контентом, заповнення бази даних інформацією;

– система управління зворотнього зв`язку з адміністратором;

– система управління обміну інформації між користувачами.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Розробка бази даних

Для створення нової бази даних на MySQL потрібно виконати такі кроки:

– потрібно встановити MySQL – перед створенням бази даних переконайтеся, що на вашому комп'ютері або сервері встановлений MySQL сервер. Завантажте MySQL Community Edition з офіційного веб-сайту MySQL і дотримуйтесь наданих інструкцій для встановлення;

– потрібно підключитися до сервера MySQL – після встановлення підключіться до вашого MySQL сервера за допомогою клієнта, наприклад, командного рядка MySQL або графічного інтерфейсу, наприклад, MySQL Workbench. Потрібно ввести облікові дані (ім'я користувача, пароль і, можливо, ім'я хоста) для підключення до сервера;

– потрібно створити нову базу даних – після успішного підключення ви можете створити нову базу даних за допомогою SQL-запиту. Наприклад, виконайте таку SQL-команду для створення бази даних з назвою "labhub":

```
CREATE DATABASE labhub;
```

– потрібно перевірити, чи була створена база даних: Ви можете перевірити, чи була база даних створена, виконавши SQL-запит, який відобразить список всіх наявних баз даних на сервері:

```
SHOW DATABASES;
```

Після виконання цього запиту ви повинні побачити свою нову базу даних у списку.

Це основні кроки для створення нової бази даних на MySQL. Після створення бази даних ви зможете створювати таблиці і здійснювати різні операції з даними.

3.2 Розробка програмних модулів

Планується створення системи, яка складатиметься з різних модулів, що втілюватимуть вказану архітектуру (модель, сервіс, репозиторій, контролер, представлення). Кожен модуль відповідатиме за роботу з окремим об'єктом або групою об'єктів в межах певної предметної області. Нижче наведено список модулів, на які буде розподілена програма:

- модуль керування повідомленнями;
- модуль керування авторизацією;
- модуль керування реєстрацією;
- модуль керування користувачами;
- модуль керування транзакціями;
- модуль керування статусами;
- модуль керування замовленнями;
- модуль керування категоріями;
- модуль керування профілем користувача;
- модуль керування профілем адміністратора;
- модуль керування головною сторінкою;
- модуль керування стрінкою інформація про нас;
- модуль керування сторінкою зв’язок з адміністрацією;
- модуль керування сторінкою файлів;
- модуль керування сторінкою замовлень;
- модуль керування сторінкою транзакцій;
- модуль керування сторінкою користувачів;
- модуль керування сторінкою пердметів;
- модуль керування сторінкою категорій;
- модуль керування сторінкою авторизації;
- модуль керування сторінкою реєстрації.

Крім основних модулів, планується використання додаткового компонента для забезпечення безпеки, який відповідатиме за захист сайту, реєстрацію та авторизацію користувачів. Модуль авторизації і реєстрації буде використовувати функціонал безпеки для реалізації процесів реєстрації та авторизації.

Усі компоненти системи будуть взаємодіяти з модулем бази даних і матимуть графічний інтерфейс для зміни, додавання, видалення та перегляду інформації.

3.3 Керівництво користувача

При вході на клієнтську частину користувач потрапляє на головну сторінку (рис. 3.1), де може авторизуватися або зареєструватися.

Рисунок 3.1 – Вигляд форми Welcome

Коли користувач нажимає на кнопку авторизації, він переходить на сторінку Login (рис. 3.2).

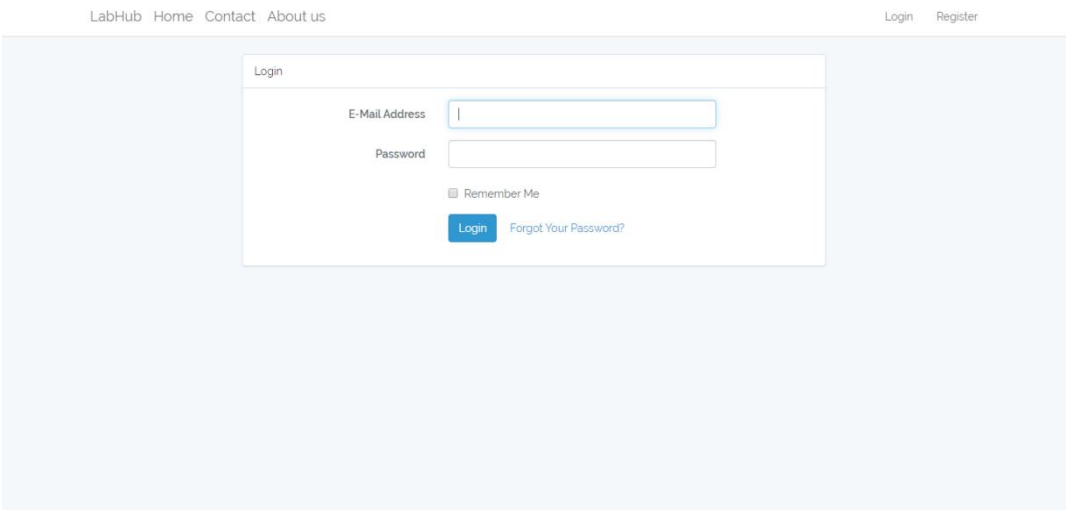


Рисунок 3.2 – Вигляд сторінки Login

Якщо користувач нажимає на кнопку реєстрації, він переходить на сторінку Register (рис. 3.3).

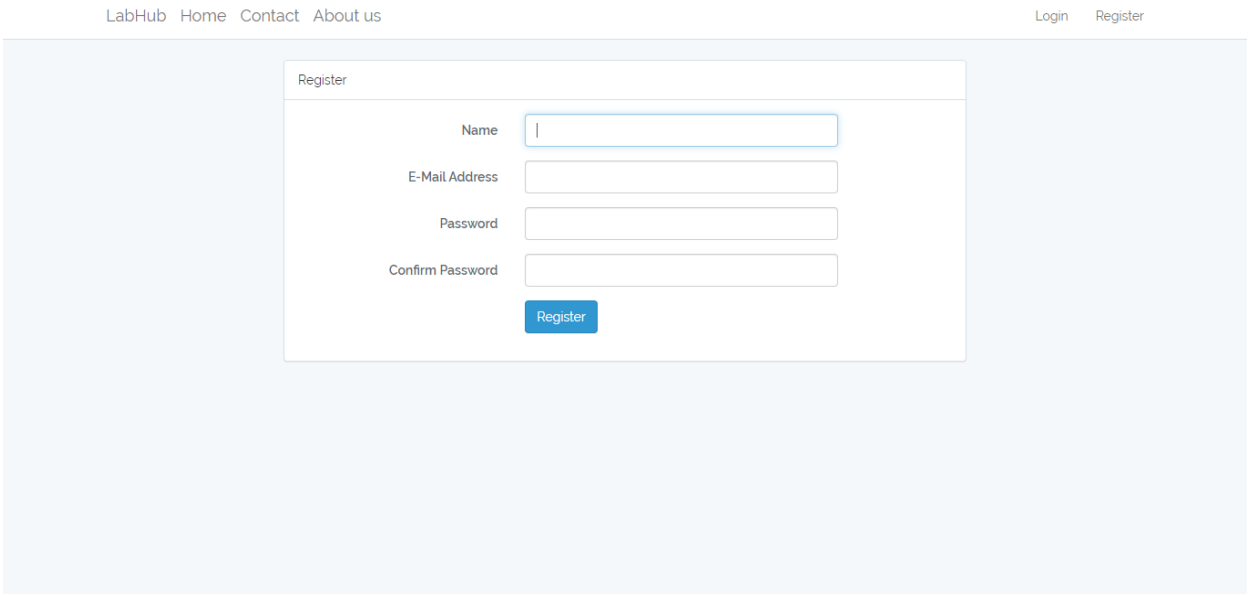


Рисунок 3.3 – Вигляд сторінки Register

Користувачу потрібно заповнити поля (рис. 3.4): name, email, password та confirm password.

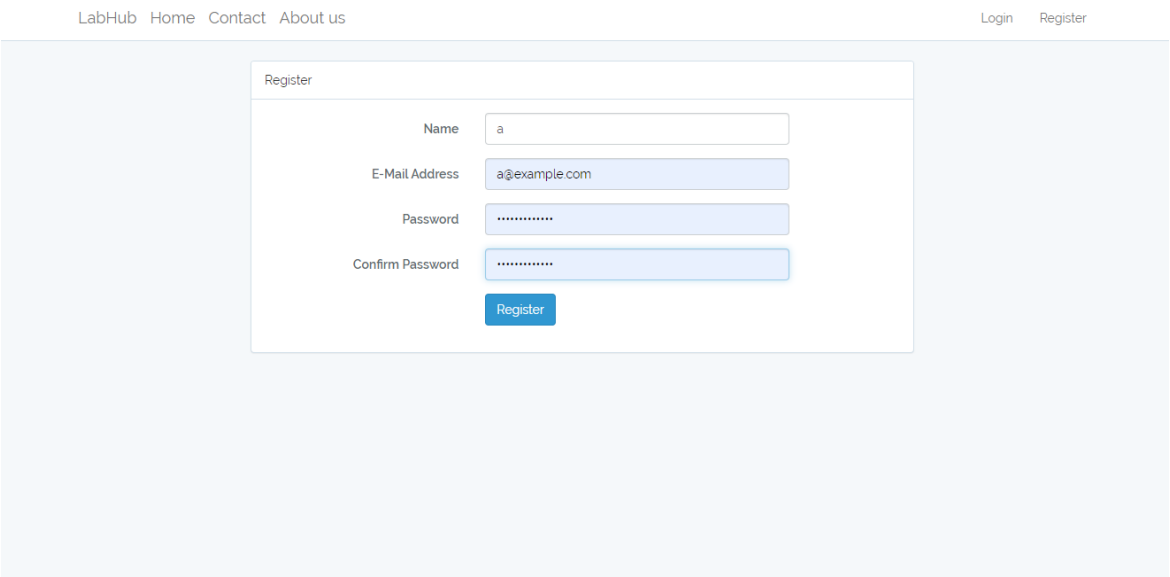


Рисунок 3.4 – Заповнення полів на формі реєстрації

Коли поля всі заповнені тиснемо на кнопку “Register” і відбувається перехід на сторінку Home (рис. 3.5).

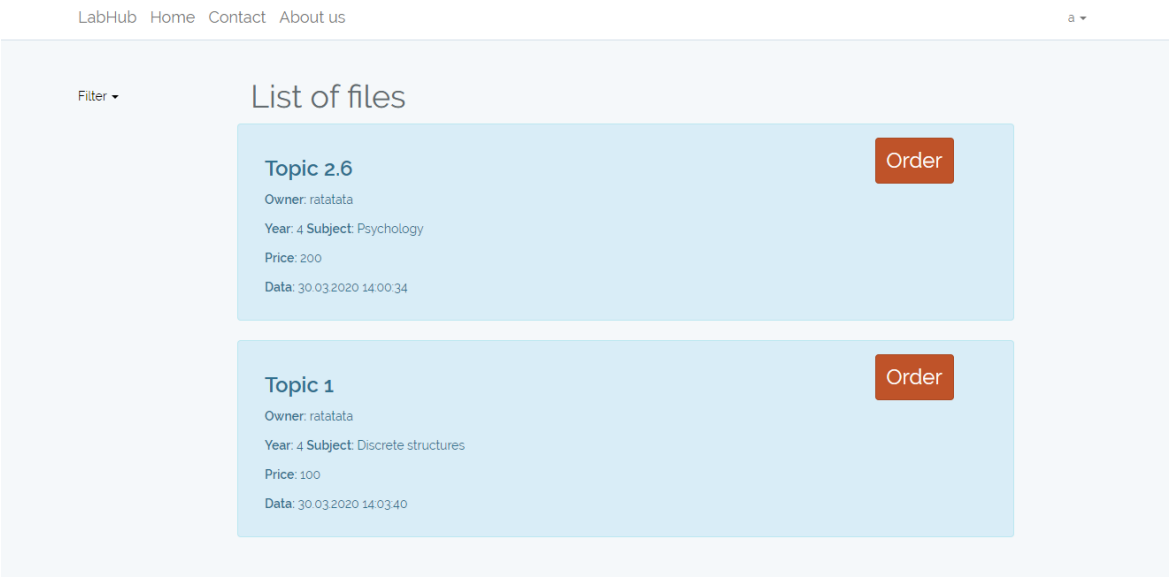


Рисунок 3.5 – Вигляд сторінки Home

В користувача є можливість но сторінці home зробити пропозицію до файлу, для цього потрібно натиснути на кнопку “Order” і він перейде на наступну сторінку (рис. 3.6).

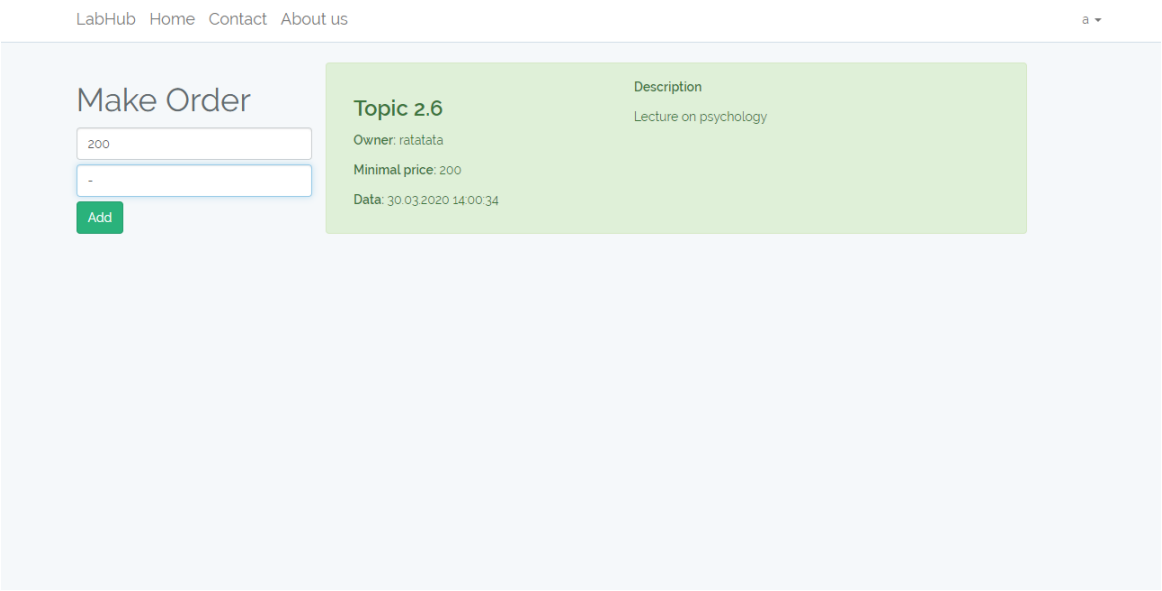


Рисунок 3.6 – Вигляд сторінки Make Order

Натиснувши на кнопку Add користувач бачить наступну сторінку (рис. 3.7).

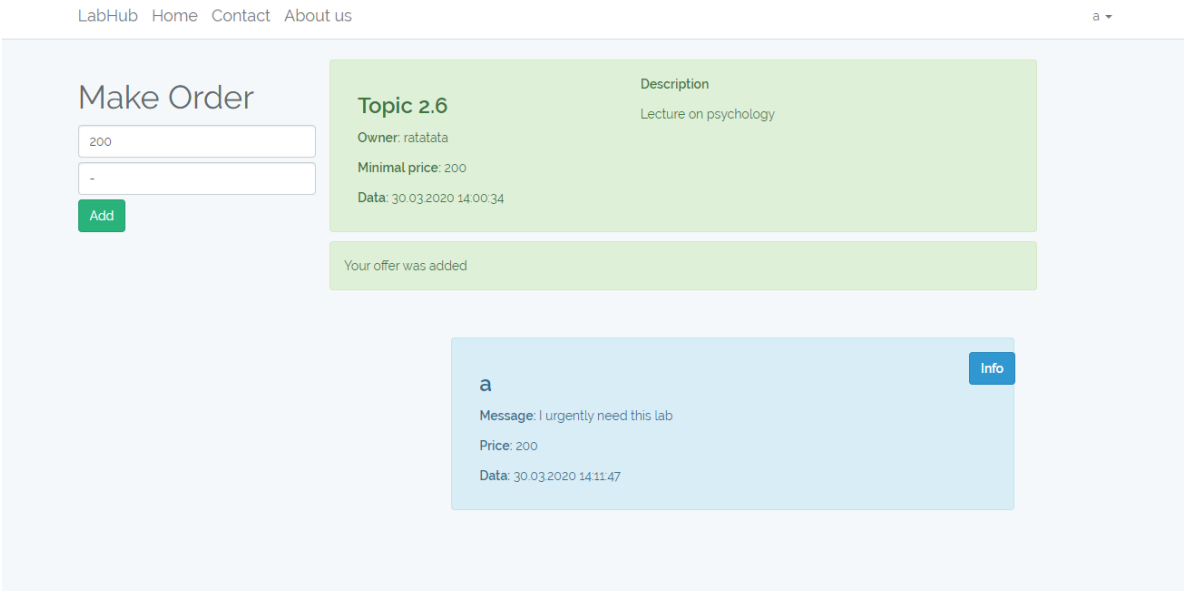


Рисунок 3.7 – Результат натиснення кнопки Add

На сторінці My Orders (рис. 3.8) користувач може здійснювати пошук, фільтр по статусу пропозиції та переглядати свої пропозиції.

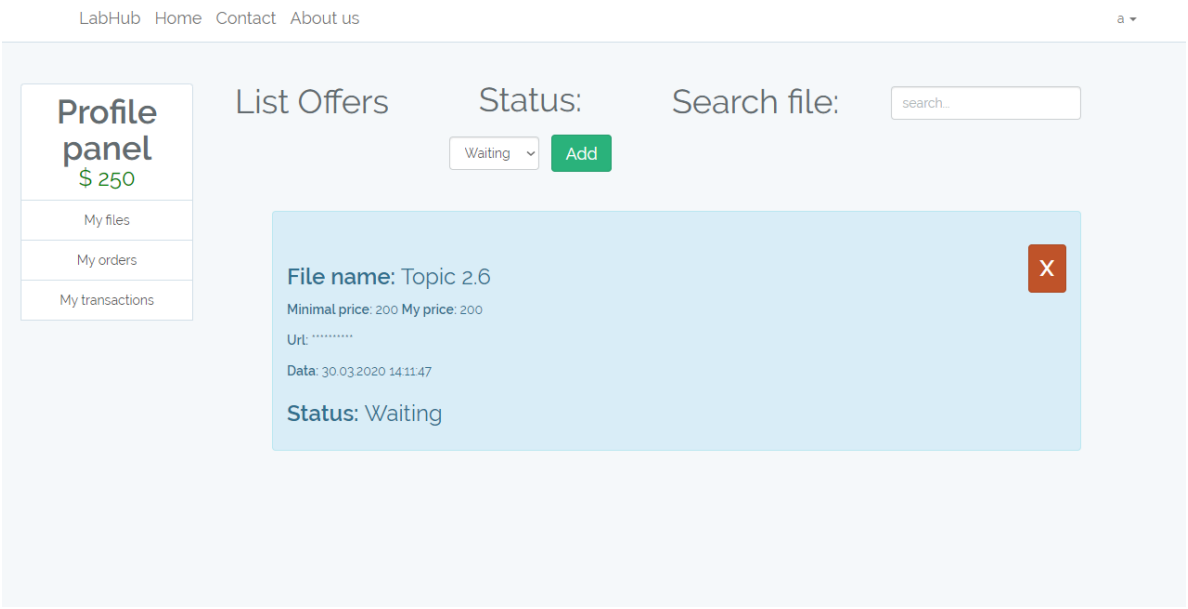


Рисунок 3.8 – Вигляд сторінки My Orders

На сторінці My Files (рис. 3.9) користувач може здійснювати пошук, додавати нові файли, переглядати свої пости та приймати або відхиляти пропозиції, які йому надходять від інших користувачів.

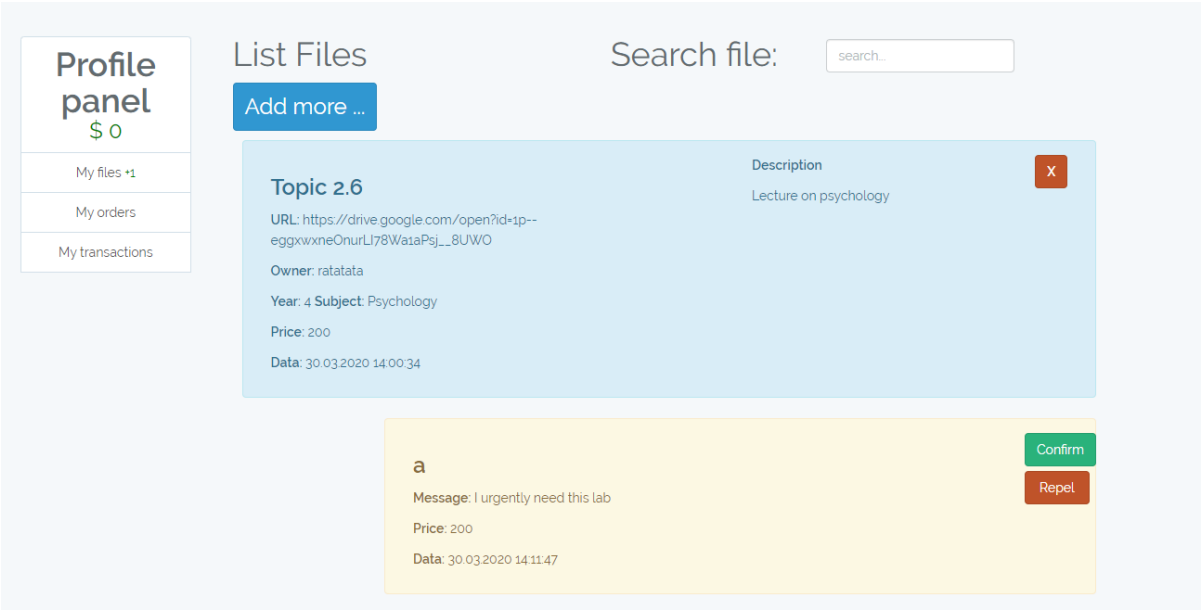


Рисунок 3.9 – Вигляд стоірнки My Files

На сторінці My Transactions (рис 3.10) користувач може переглядати його транзакції та здійснювати пошук по даті.

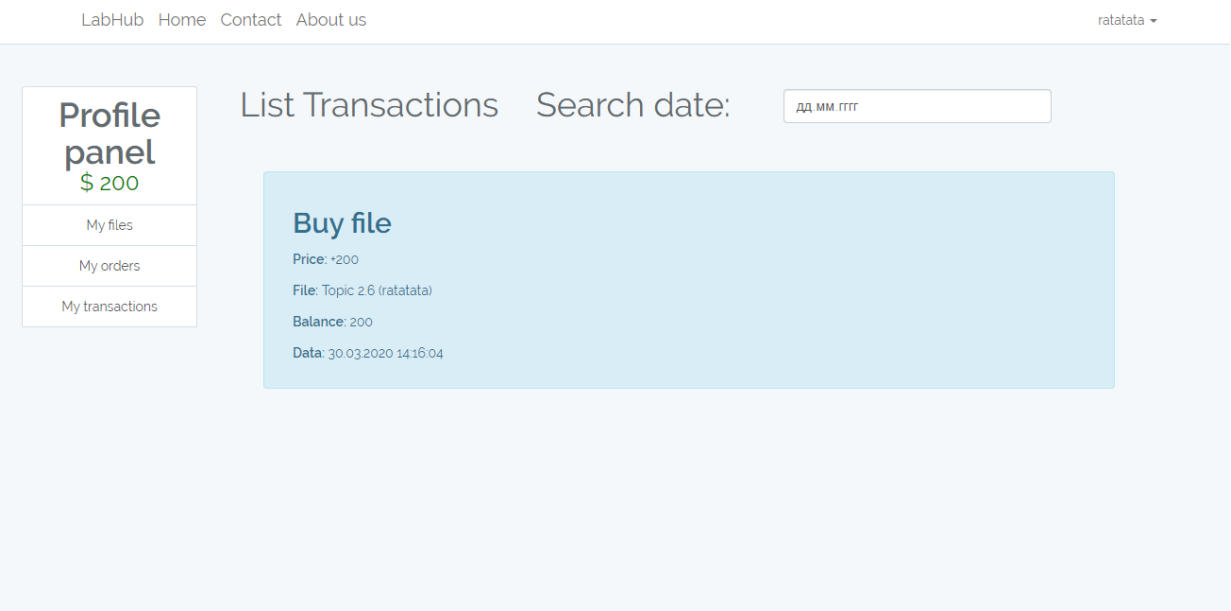


Рисунок 3.10 – Вигляд сторінки My transactions

Користувач в разі необхідності може звернутися до адміністраторів натиснувши на кнопку Contact (рис. 3.11), у верхній частині веб-сторінки (header).

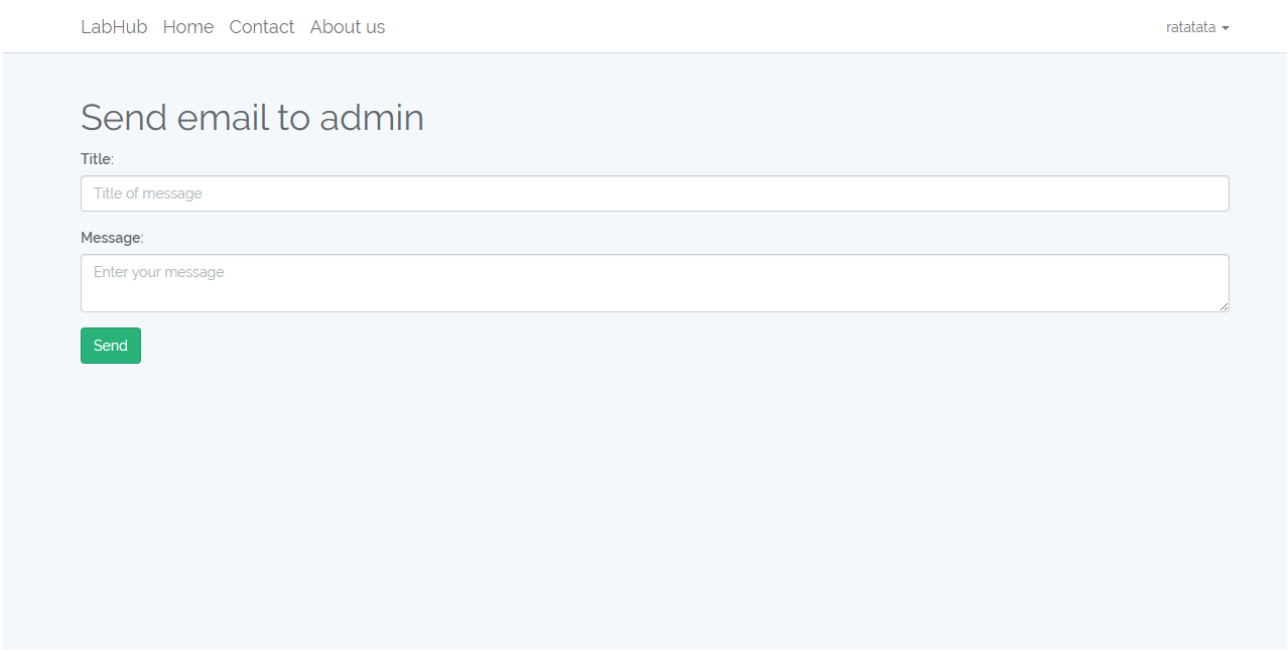


Рисунок 3.11 – Вигляд сторінки Contact

Користувач може додавати нові файли в своєму профілі натиснувши на кнопку Add more … (рис. 3.12-3.13).

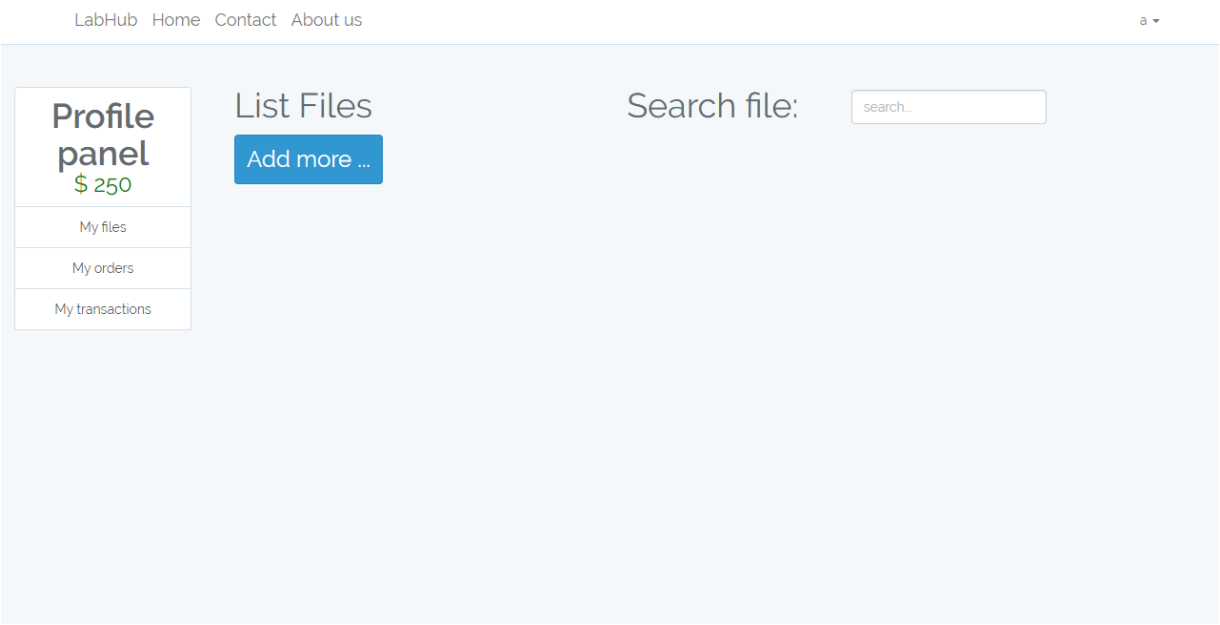


Рисунок 3.12 – Вигляд сторінки My Files

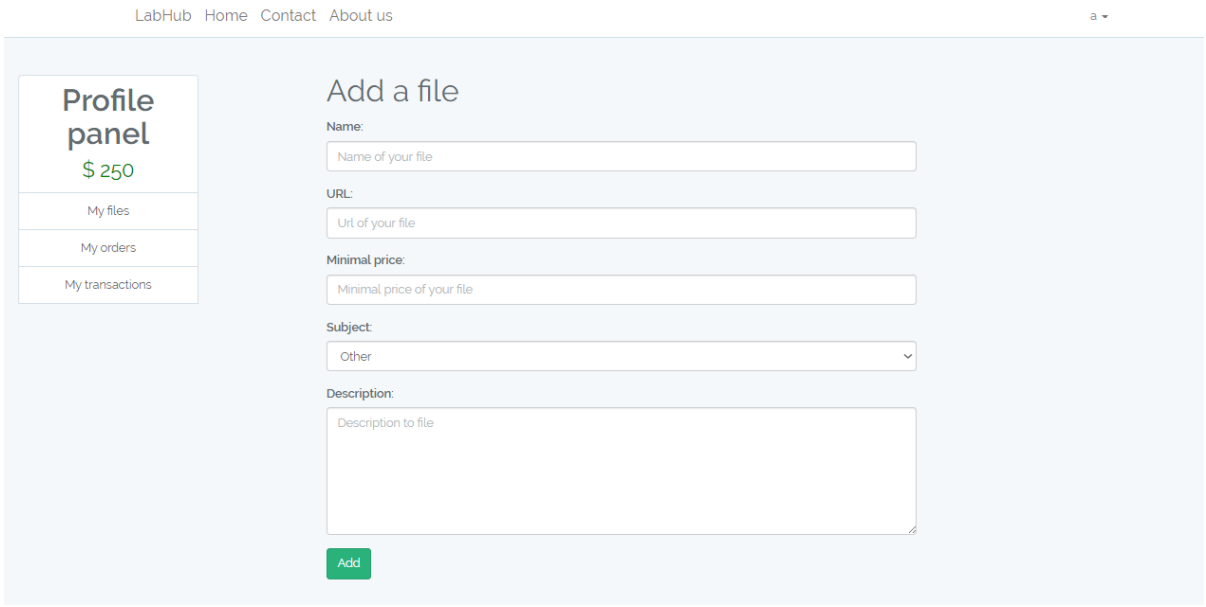


Рисунок 3.13 – Вигляд форми Add Files

На сторінці Files адміністратор бачить всі файли, видаляти, при цьому всі пропозиції в яких статус “Waiting” відхиляються та здійснюється повернення коштів всім користувачам які робили до цього файлу бартер, і здійснювати пошук (рис. 3.14).

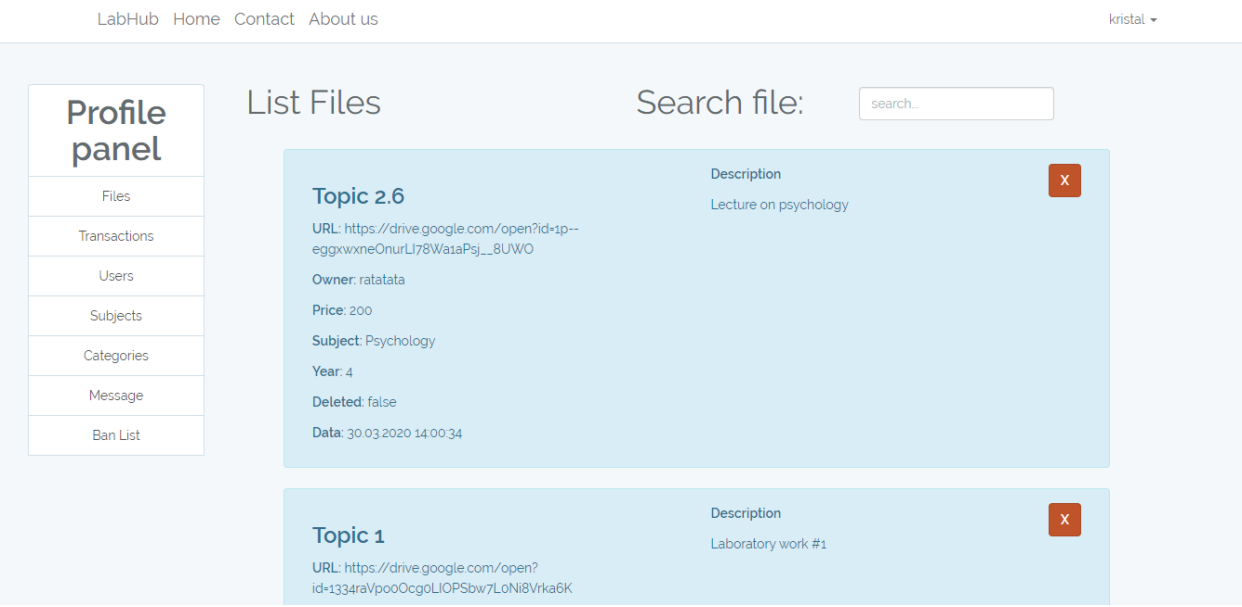


Рисунок 3.14 – Вигляд сторінки Files

На сторінці Transactions (рис. 3.15) адміністратор може переглядати транзакції всіх користувачів та здійснювати пошук по даті.

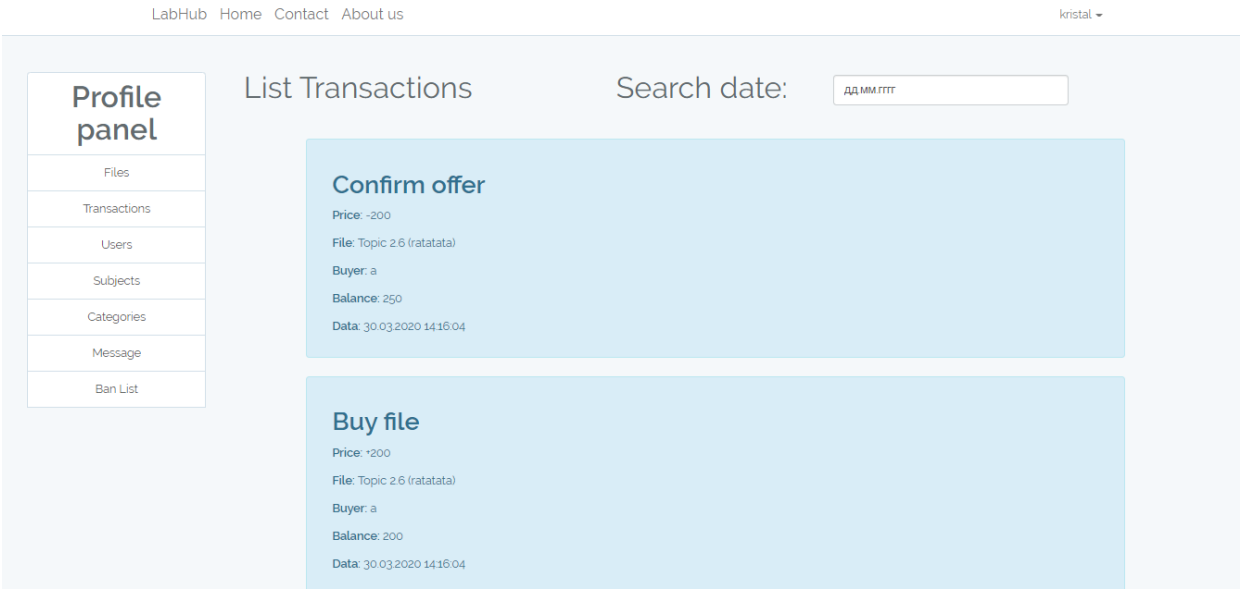


Рисунок 3.15 – Вигляд сторінки Transactions

На сторінці Users (рис. 3.16) адміністратор бачить список всіх користувачів з можливостю нараховувати їм кошти, банити або розбанити та здійснювати пошук.

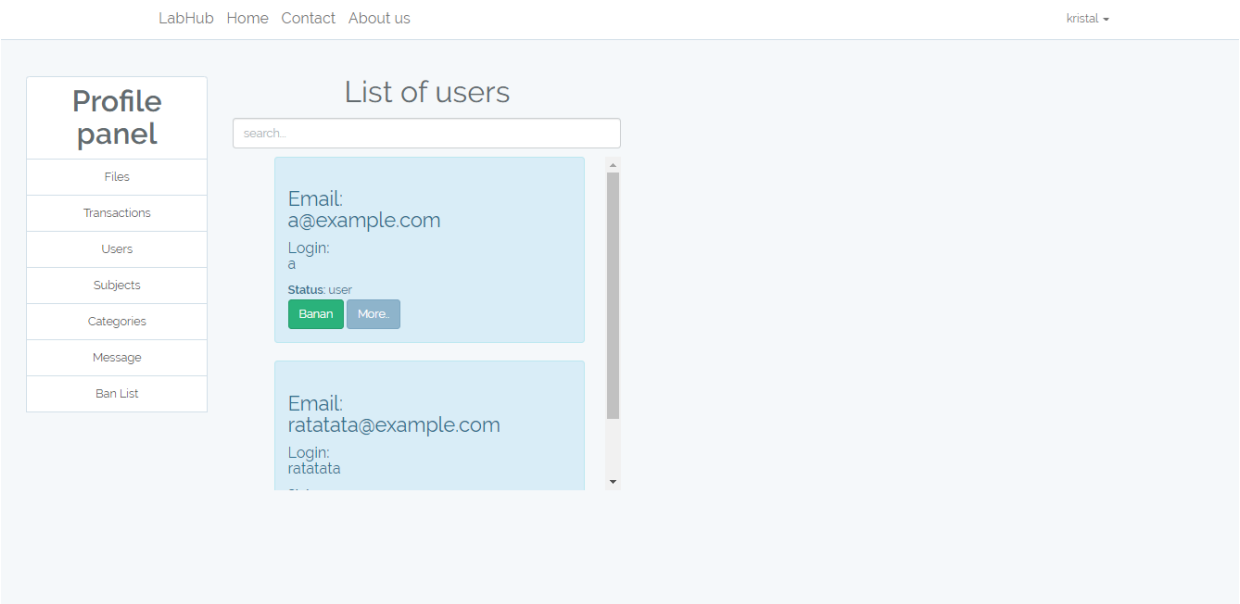


Рисунок 3.16 – Вигляд сторінки Users

На сторінці Subjects (3.17) адміністратор бачить список предметів з можливістю додавати нові.

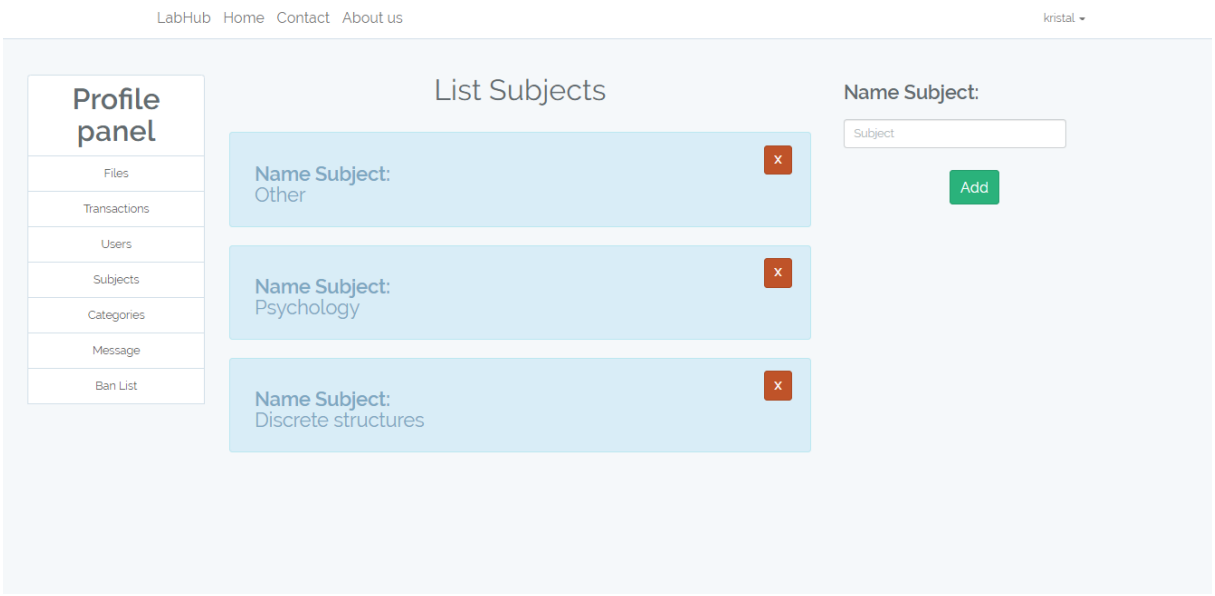


Рисунок 3.17 – Вигляд форми Subjects

На сторінці Categories (рис. 3.18) адміністратор бачить список всіх категогій з можливістю додавати нові.

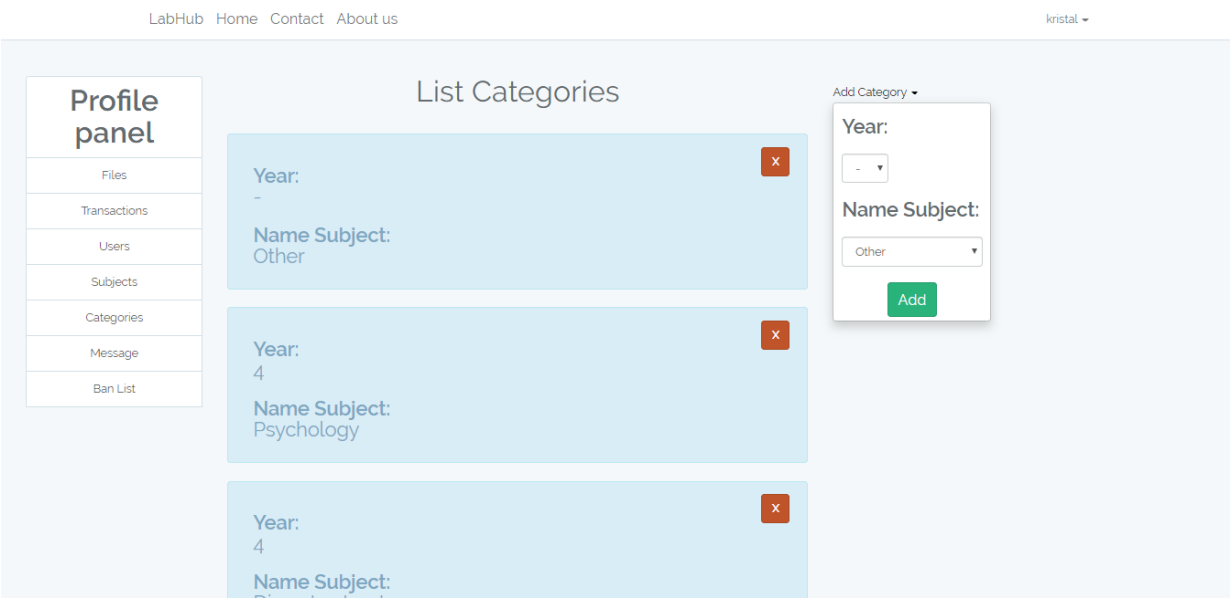


Рисунок 3.18 – Вигляд форми Categories

На сторінці Messages (рис. 3.19-3.20) адміністратор бачить список всіх повідомлень.

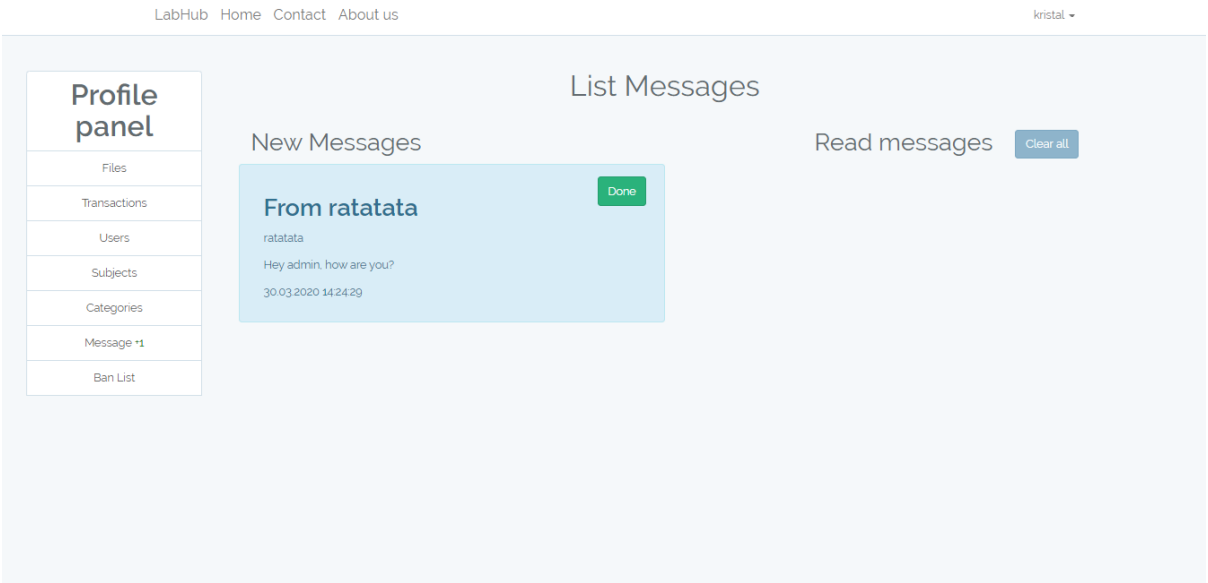


Рисунок 3.19 – Вигляд сторінки Messages (New Messages)

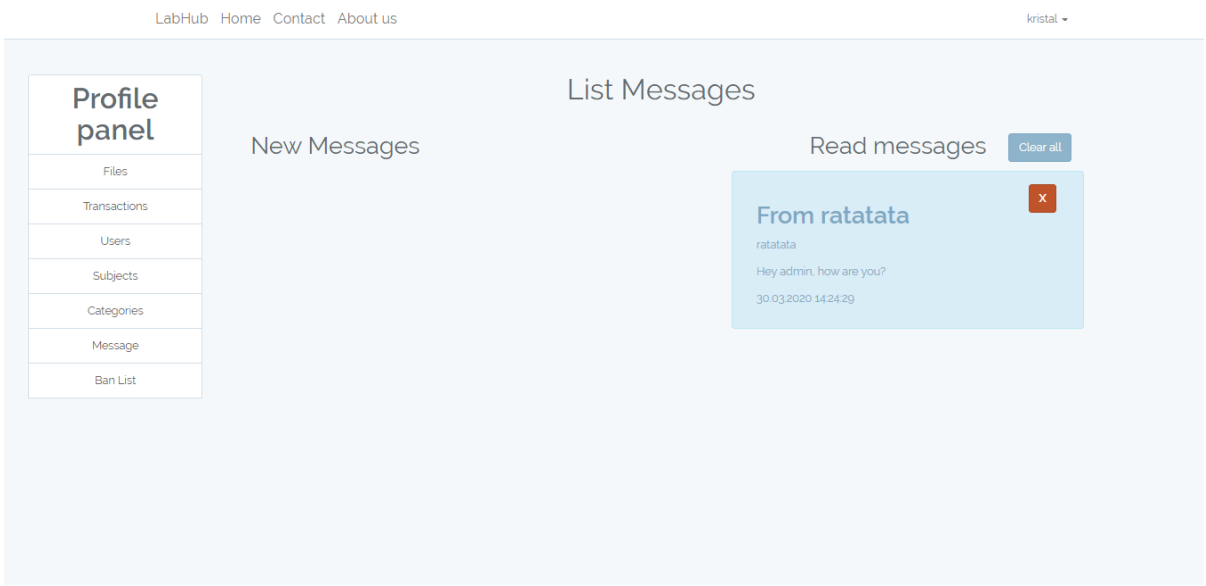


Рисунок 3.20 – Вигляд стоірнки Messages (Read Messages)

На сторінці Ban List (рис. 3.21) адміністратор бачить список всіх заблокованих користувачів з можливістю розблокувати.

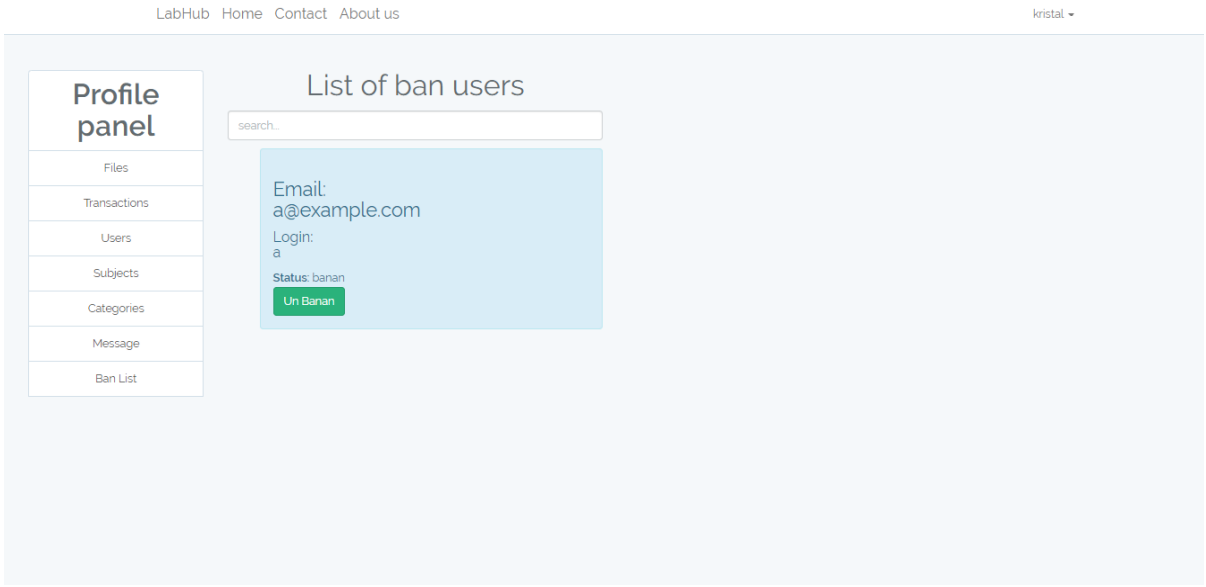


Рисунок 3.21 – Вигляд форми Ban List

3.4 Технічні характеристики інтернет-платформи

Автоматизована система реалізована у вигляді клієнт-серверного веб сайту. Клієнтом виступатиме браузер будь-якого пристрою, який має доступ до інтернету, або власне серверу, на який ми становимо наш сайт.

Для встановлення серверної частини необхідно мати фізичний або хмарний сервер, який відповідатиме таким вимогам:

– рекомендована операційна система – Linux Ubuntu 16.4 і вище, Windows 8-11;

– об’єм вільного простору – від 400 Мб;

– мінімум 4 Гб оперативної памяті;

– мінімум 2 ядерний процесор.

– доступ до мережі «Інтернет» зі швидкістю мінімум 20Мбіт/с.

3.5 Розгортання та встановлення системи

В залежності від обраної операційної системи деякі етапи встановлення проєкту будуть змінюватись. Розглянемо одну з найпопулярніших безкоштовних операційних систем Linux Ubuntu 16.04 і вище та етапи встановлення проекту саме для користувачів цієї ОС.

Щоб запутити проект необхідно встановити:

– Heroku – хмарна PaaS-платформа, що підтримує ряд мов програмування. З 2010 року є дочірньою компанією Salesforce.com. Heroku, одна з перших хмарних платформ, з'явилася в червні 2007 року і спочатку підтримувала тільки мову програмування Ruby, але на даний момент список підтримуваних мов також включає в себе Java, Node.js, Scala, Clojure, Python, Go, Ruby і PHP. На серверах Heroku використовуються операційні системи Debian або Ubuntu;

– Composer – менеджер пакетів прикладного рівня для мови програмування PHP що забезпечує стандартний формат для управління залежностями у програмному забезпеченні та необхідними бібліотеками;

– встановлення глобально на комп’ютер самого фреймворка Laravel;

– створення проекту на фреймворці Laravel за допомогою Composer.

Встановлення Composer:

Ви не можете встановити програму з офіційних репозиторіїв. Потрібно завантажити скрипт з офіційного сайту і помістити його в папку з вашим проектом. Але спочатку обновіть систему і встановіть залежності:

sudo apt update

sudo apt install curl php-cli php-mbstring git unzip

Ви можете завантажити файл установщика в будь-яку папку, наприклад, домашню:

curl -sS https://getcomposer.org/installer -o composer-setup.php

Тільки команда установки буде відрізнятися, в ній ми вказуємо папку, куди потрібно встановити скрипт (рис. 3.22):

sudo php composer-setup.php --install-dir=/usr/local/bin --filename=composer

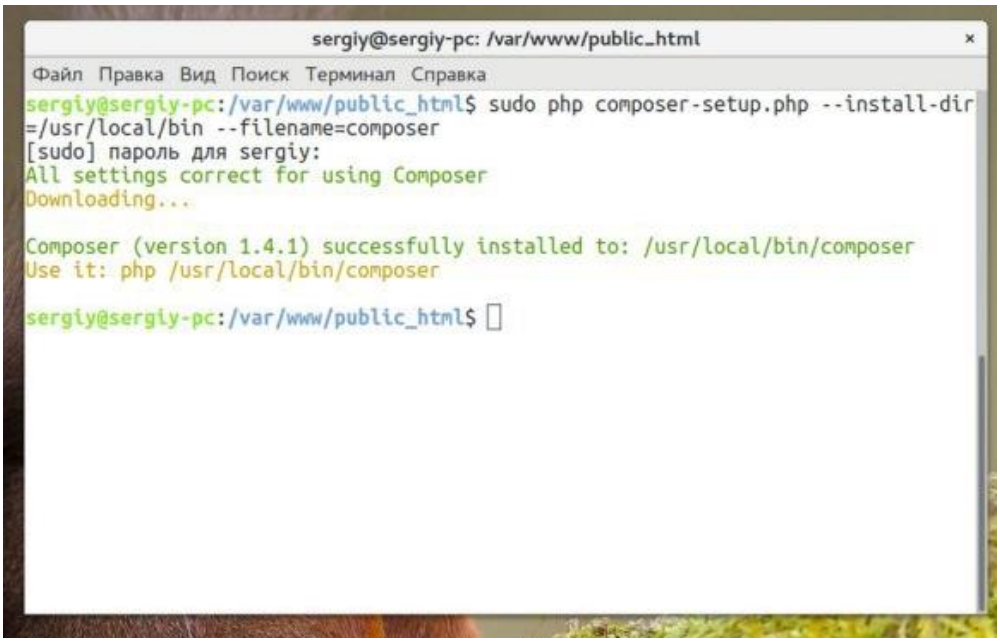


Рисунок 3.22 – Процес встановлення composer

Для перевірки роботи, ви можете виконати команду (рис. 3.23):

composer

Рисунок 3.23 – Процес перевірки роботи команди

Встановлення Laravel:

Спочатку потрібно завантажити установщик Laravel за допомогою Composer. Для цього досить виконати одну команду:

composer global require "laravel/installer"

Далі встановіть проект у вибрану папку, замість зірочок потрібно вказати папку, в яку потрібно встановити проект (рис. 3.24):

sudo composer create-project --prefer-dist laravel/laravel *****

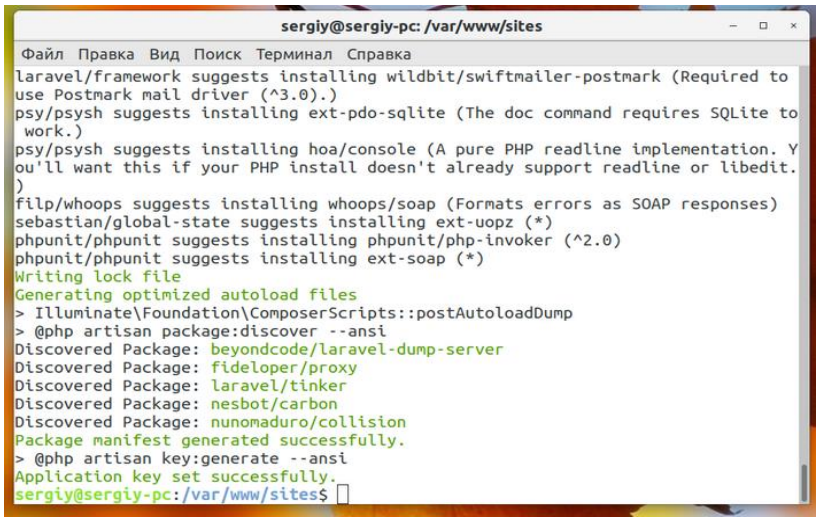


Рисунок 3.24 – Процес встановлення прєкту

Встановлення Heroku:

Спочатку потрібно встановити Heroku CLI на комп’ютер. Heroku підтримує різні операційні системи.

Завантажте і встановіть базу інсталятора Heroku на своїй операційній системі.Після завершення встановлення Heroku ви зможете повернутись до свого терміналу і попросити отримати доступ до Heroku.

heroku login

Спливаюче вікно входу в систему буде виглядати так (рис. 3.25).

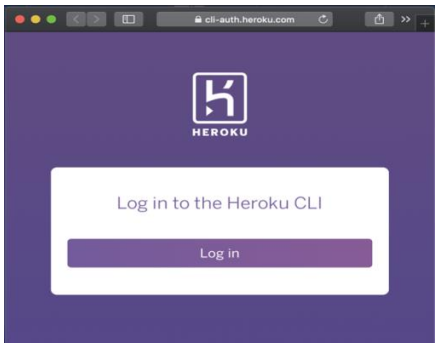


Рисунок 3.25 – Авторизація в Heroku

Після успішного входу в систему потрібно повернутися в термінал, в якому відбулася авторизація користувача.

На цьому етапі вам потрібно створити Procfile (конфігураційний файл Heroku) всередині вашого основного каталогу проекту Laravel.

Після цього ви можете зберегти свій Procfile, а потім повернутися в свій термінал і ввести.

git init

heroku create

Ви отримаєте своє ім'я додатки, створене Heroku, і своє ім'я сховища.

Після цього вам потрібно помістити цей проект на в Heroku:

git add .

git commit -m “Initial Commit”

git push heroku master

Тепер додаток Laravel вже розгорнуто в Heroku, панель моніторингу Heroku (рис. 3.26).

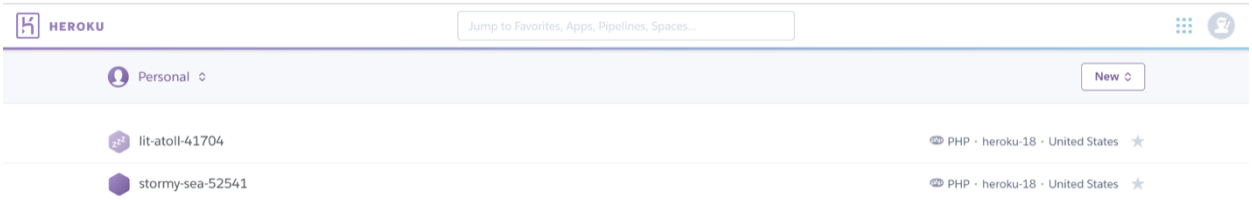


Рисунок 3.26 – Панель моніторингу Heroku

Далі встановимо деяку конфігурацію всередині Heroku, в меню "Налаштування" (рис. 3.27).

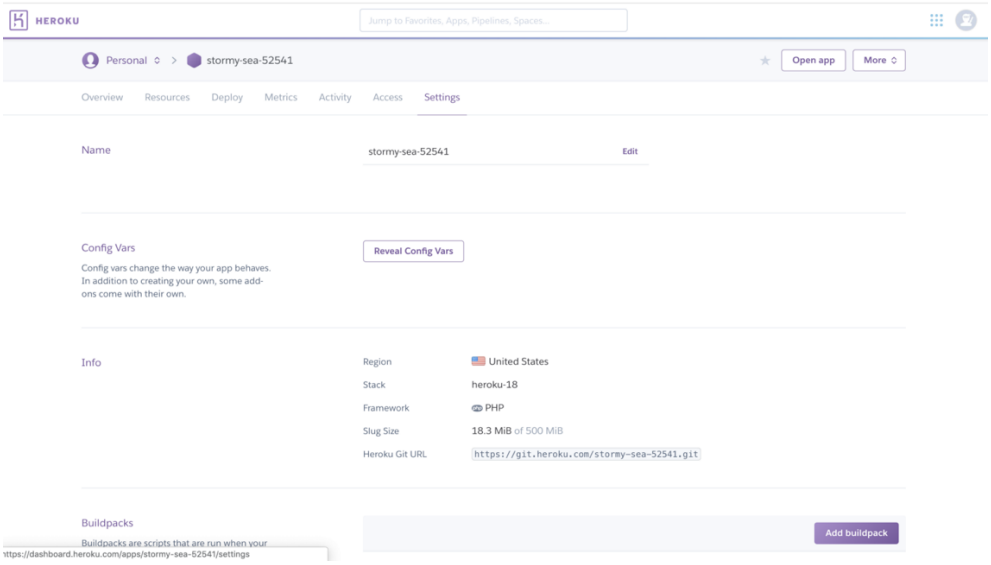


Рисунок 3.27 – Меню нашалтувань Heroku

Потім натисніть кнопку "Reveal Config Vars" (рис. 3.28).

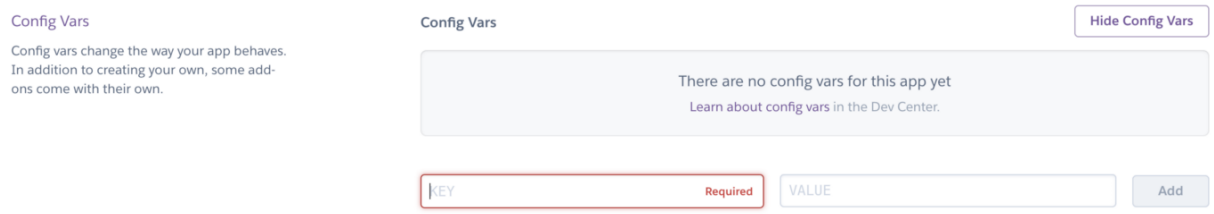


Рисунок 3.28 – Reveal Config Vars в Heroku

Нам потрібно додати деяку конфігурацію від нашого .env файл в це поле (рис. 3.29).

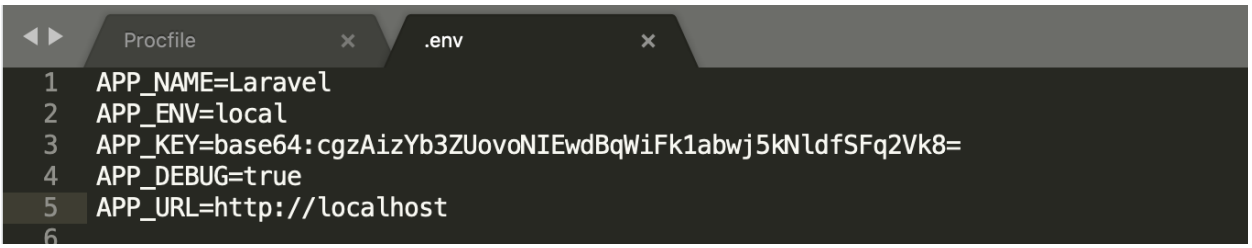


Рисунок 3.29 – .env для Heroku

Після того як буде добавлено конфігураційні налаштування (рис. 3.30).

Рисунок 3.30 – .env для Heroku

Останній крок стосується конфігурації бази даних, щоб запустити міграції в Heroku. Спочатку потрібно перейти в меню "ресурси", а потім в розділі "доповнення" натиснути кнопку "знайти додаткові доповнення". Користувач буде перенаправлений на сторінку доповнень Heroku.

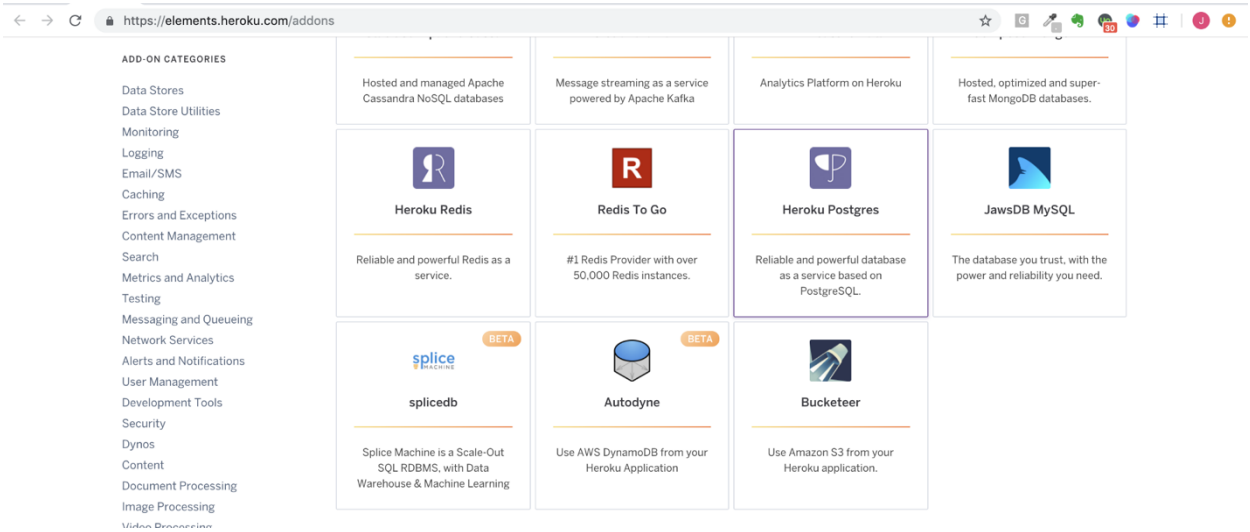


Рисунок 3.31 – Доповнення для Heroku

Знайшовши Heroku Postgres встановлюємо його та підключаємо до проекту в HEROKU, натиснувши на кнопку “Provision add-on”.

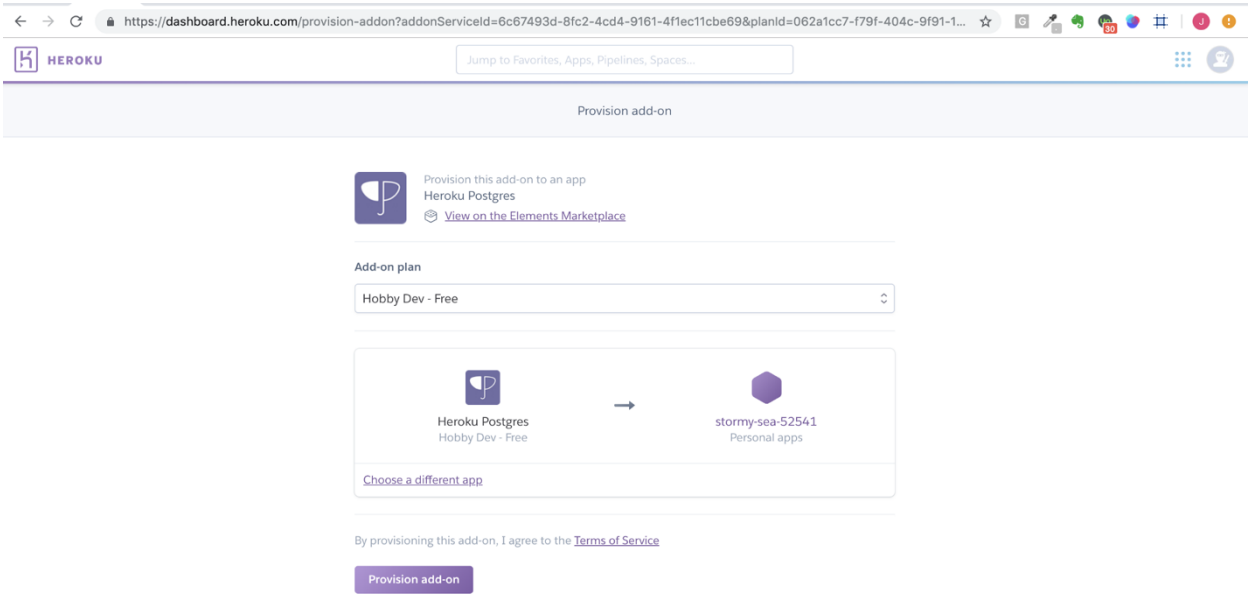


Рисунок 3.32 – Підключення БД в Heroku

Після успішного встановлення переходимо в термінал та прописуєм:

heroku config

Серед отриманого списку конфігурацій знаходимо потрібну “DATABASE_URL”.

В файлі config/database.php потрібно замінити “DATABASE_URL” на значення отримане в терміналі. Далі встановити по замовчуванню pgsql базою даних.

Останім кроком буде запушити всі зроблені зміни на HEROKU репозиторій та запустити всі міграційні файли laravel:

git add .

git commit -m “Update database connection”

git push heroku master

3.6 Тестування веб-платформи

Тестування веб-сайтів поділяються на такі види тестів:

– модульне тестування – це метод тестування, при якому перевіряється на працездатність необхідних модулів сайту. Модуль – це найменша частина програмного забезпечення, яка може бути протестована. Для даного програмного забезпечення необхідно перевірити усі методи дій та дані які вони повертають.

– інтеграційні тести – їх призначення тестувати усі модулі разом взяті або їх певне поєднання. Для веб-сторінки варто провести перевірку коректного відображення усіх компонентів системи та їх роботу.

– UI тести – призначені для тестування користувацького інтерфейсу, імітуючи дії користувача програма по різному взаємодіє з інтерфейсом перевіряючи його на працездатність і правильність.

– Stress тести – дані тести призначені для того, щоб показати навантаження, що може обробляти сервер, кількість запитів та швидкість обробки та віддачі інформації за певний проміжок часу.

Тестування програмного забезпечення – процес перевірки відповідності заявлених до продукту вимог і реально реалізованої функціональності,

здійснюваний шляхом спостереження за його роботою в штучно створених ситуаціях і на обмеженому наборі тестів, обраних певним чином.

Висновки до розділу

У даному розділі було розкрито процес створення бази даних та її підключення до веб-застосунку. Крім того, була проведена декомпозиція системи на модулі та детально розглянуті кожен з них. Технічні характеристики веб-застосунку були визначені. Також було описано алгоритм інсталяції та експлуатації розробленого програмного продукту. Після проведення тестування можна зробити висновок, що програмний засіб успішно пройшов всі необхідні тести, задовольняє вимоги, викладені у технічному завданні, і готовий до розгортання та використання.

На підставі вимог та поставлених задач, був розроблений веб за стосунок, який містить модулі, такі як:

– система наповнення веб-платформи контентом, заповнення бази даних інформацією;

– система управління зворотнього зв`язку з адміністратором;

– система управління обміну інформації між користувачами.

ВИСНОВКИ

У процесі розробки кваліфікаційної роботи було проведено аналіз предметної області, сформульовано завдання роботи, визначено вимоги до програмного забезпечення та розроблено специфікацію продукту, яка супроводжується UML діаграмами варіантів використання та послідовності.

Наступним кроком було перейти до проектування веб-платформи. На цьому етапі було проведено аналіз переваг і недоліків популярних архітектурних патернів, які використовуються при розробці веб-застосунків. В результаті було визначено, що найбільш підходящою для цієї програмної системи є клієнт-серверна архітектура з використанням патерну MVC.

Після цього було розроблено архітектуру бази даних, визначено поля, таблиці та зв'язки між ними. З урахуванням спроектованої архітектури веб-застосунку та бази даних було прийнято рішення щодо вибору технологій та інструментів, які найкраще відповідають цьому типу архітектури. Зокрема, були обрані фреймворки Laravel, Bootstrap та реляційна база даних MySQL. Швидкість та надійність роботи, що забезпечують ці технології, були ключовими факторами при виборі.

Наступним етапом було розбиття програмної системи на модулі та проектування взаємодії між ними. Крім того, було проведено проектування інтерфейсу для веб-застосунку.

На основі розробленої архітектури та інтерфейсу було створено базу даних та реалізовано серверну та клієнтську частини системи, а також механізм взаємодії між ними.

Останнім етапом було проведення тестування, під час якого виявлено та виправлено всі недоліки щодо функціональності та зовнішнього вигляду. Були проведені модульні та функціональні тести, і на основі результатів тестування був складений звіт. Проведені тести підтвердили, що система виконує всі

функції, описані у технічному завданні, та має зручний інтерфейс для користувача без зайвих елементів.

В результаті роботи було розроблено веб-застосунок, який виконує поставлені завдання:

– розроблено простий та привабливий інтерфейс користувача для зручного використання веб-платформи.

– розроблено систему наповнення веб-платформи контентом та заповнення бази даних інформацією.

– забезпечено систему управління зворотного зв'язку з адміністратором.

– розроблено систему управління обміном інформації між користувачами.

Таким чином, поставлене завдання кваліфікаційної роботи було повністю виконано. Розроблений веб-застосунок для обміну інформацією працює бездоганно та ефективно. Ця робота дозволила отримати практичні та теоретичні навички на всіх етапах розробки програмного забезпечення та закріпити отримані знання.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Дипломний проект: методичні вказівки щодо його виконання для студентів спеціальності 121 «Інженерія програмного забезпечення» / Бедратюк Л. П., Радельчук Г. І., Форкун Ю. В., Яшина О. М. Хмельницький : ХНУ, 2020. 77с .
2. Сервер. Створення сервера – URL: https://metanit.com/web/nodejs/3.1.php/ . (дата звернення – 17.04.2023).
3. Laracasts – відео уроки по Laravel – URL: https://laracasts.com/ . (дата звернення – 18.04.2023).
4. NOSQL – переваги та недоліки нереляційних баз даних / QAinfo. – URL: https://www.quality-assurance-group.com/nosql-perevagy-ta-nedoliky-nerelyatsijnyh-baz-danyh/ . (дата звернення – 04.05.2023).
5. SQL – переваги баз даних . – URL: https://www.quality-assurance-group.com/sql-perevagy-ta-nedoliky-nerelyatsijnyh-baz-danyh/ . (дата звернення – 04.05.2023).
6. Laravel – php фреймворк – URL: https:/laravel.com/ . (дата звернення – 04.05.2023).
7. Laravel – офіційна документація Laravel – URL: https://laravel.com/docs/ . (дата звернення – 05.05.2023)
8. MySQL – серверна реляційна база даних – URL: https:/mysql.com/ . (дата звернення – 06.05.2023).
9. NOSQL – переваги та недоліки нереляційних баз даних / QAinfo. – URL: https://www.quality-assurance-group.com/nosql-perevagy-ta-nedoliky-nerelyatsijnyh-baz-danyh/ . (дата звернення – 04.05.2023).
10. GitHub – хмарне кодове середовище – URL: https://github.com/ . (дата звернення – 07.05.2023).
11. GitHub – deploy документація – URL: https://docs.github.com/en/actions/deployment/ . (дата звернення – 10.05.2023).

12. PHP – php документація – URL: https://www.php.net/ . (дата звернення – 07.05.2023).

13. Модульне тестування / QaLight. – URL: https://qalight.ua/baza-znaniy/modulne-testuvannya/ . (дата звернення – 08.05.2023).

14. Уніфікована мова програмування UML / Портал знань. – URL: http://www.znannya.org/?view=uml . (дата звернення – 11.05.2023).

15. UML – специфікація UML URL: https://www.omg.org/spec/UML/ . (дата звернення – 11.05.2023).

16. HTML – документація HTML – URL: https://developer.mozilla.org/en-US/docs/Web/HTML . (дата звернення – 25.04.2023).

17. Bootstrap – документація Bootstrap – URL: https://getbootstrap.com/docs/ . (дата звернення – 30.04.2023).

18. Bootstrap – шаблони Bootstrap – URL: https://bootstrapbay.com/ . (дата звернення – 30.04.2023).

19. Heroku – документація Heroku – URL: https://devcenter.heroku.com/ . (дата звернення – 12.05.2023).

20. PHP –документація PHP– URL: https://www.php.net/ . (дата звернення – 07.05.2023).

21. W3SHOOLS – HTML документація – URL: https://w3shools.com/ . (дата звернення – 25.04.2023).

22. Laravel – API документація – URL: https://laravel.com/api/8.x/ . (дата звернення – 05.05.2023).

23. Docker – документація Docker – URL: https://docs.docker.com/ . (дата звернення – 17.04.2023).

24. Laravel Eloquent ORM – документація Eloquent ORM – URL: https://laravel.com/docs/eloquent . (дата звернення – 05.05.2023).

25. W3SHOOLS – MySQL документація – URL: https://www.w3schools.com/sql/ . (дата звернення – 06.05.2023).

ДОДАТОК А
(обов’язковий)

ТЕХНІЧНЕ ЗАВДАННЯ

Введення

Робота виконується в рамках проекту розробки веб-платформи “LabHub” для обміну інформацією на базі фреймворка Laravel.». Технічне завдання розроблено у відповідності до стандарту ГОСТ 19.201–78.

1 Підстава для розробки

Підставою для розробки є «Завдання на дипломний проект», затверджене завідувачем кафедри інженерії програмного забезпечення. Найменування розробки: Веб-платформа “LabHub” для обміну інформацією на базі фреймворка Laravel.

2 Призначення розробки

2.1 Функціональне призначення

Функціональним призначенням додатку пошук та замовлення подорожей для звичайного користувача, а також облік турів та замовлень для адміністратора веб-додатку.

2.2 Експлуатаційне призначення

Програма повинна експлуатуватися на будь-яких пристроях, на яких є браузер. Кінцевим користувачем додатку може виступати будь-яка особа.

3 Вимоги до програми

3.1 Вимоги до функціональних характеристик

Для адміністратора:

– перегляд та пошук за певними критеріями опублікованих файлів користувачів з нищим рівнем доступом ніж у адміністатора;

– видалення опублікованих файлів користувачів;

– перегляд та пошук за датою проведення загального журналу транзакцій;

– перегляд та пошук авторизованих користувачів, які пройшли повний етап авторизації, можливість надання даним користувачам найнищих прав доступу, а саме «Ban», але залишивши їм лише можливість авторизуватись та зв’язатися з адміністрацією веб-платформи;

– створення та видалення нових навчальних предметів;

– створення та видалення нових категорій, які будуть використовуватись під час опублікування, авторизованим, користувачем його інформації;

– перегляд отриманих повідомлень від авторизованих користувачів, з можливістю надання цим повідомлення відмітки про виконання, а також видалення повідомлень, які вже були відміченні як «Виконаний».

Для авторизованого користувача:

– перегляд наявних в нього умовних одиниць коштів;

– перегляд та фільтрація опублікованих файлів іншими користувачами;

– створення та відправлення повідомлення адміністраторам;

– створення та видалення власних публікацій з певною інформацією, фільтрація та пошук серед списку опублікованих матеріалів даного авторизованого користувача, перегляд замовлень до кожного опублікованого файлу, з подальшою можливістю підтвредженням або відхиленням їх;

– створення та видалення власних замовлень, прикріплених до публікацій інших авторизованих користувачів;

– перегляд та пошук за датою проведення журналу транзакцій, даного авторизованого користувача.

Для заблокованого користувача:

– зв’язок з адміністрацією.

Для не авторизованого користувача:

– можливість авторизації або регістрації.

3.2 Вимоги до надійності

Веб-платформа повинна забезпечувати такі вимоги до надійності:

– обробляти невірні дії користувача і попереджати його про можливі наслідки;

– можливість самостійно відновлюватись у разі збою.

3.3 Умови експлуатації та вимоги до технічних засобів

Веб-додаток повинен працювати на всіх пристроях, які мають браузер та стабільний доступ до мережі «Інтернет»: смартфонах, планшетах та комп’ютерах. Браузер може бути будь-яким: Safari, GoogleChrome, Opera, MozillaFirefox, тощо.

Для роботи платформи без збоїв, потрібно, щоб пристрій, на якому вона запускається задовольняв наступні мінімальні вимоги:

– 400 Мб внутрішньої пам’яті;

– 4 Гб оперативної пам’яті;

– 2-ядерний процесор;

– доступ до мережі «Інтернет» зі швидкістю мінімум 20 Мбіт/с.

3.4 Вимоги до інформаційної та програмної сумісності

Для розробки веб-платформи був використаний бекенд PHP фреймворк Laravel, та база даних SQL

3.5 Спеціальні вимоги

Програма повинна мати зручний, гарний та зрозумілий зовнішній інтерфейс користувача.

4 Вимоги до програмної документації

У момент здачі проекту замовнику надається наступний набір документів:

– текст програми;

– опис програми;

– технічне завдання;

– керівництво користувача.

5 Стадії та етапи розробки

Стадії та етапи розробки інтернет-платформи «Агентство подорожей» подані у таблиці А.1.

Таблиця А.1 – Стадії та етапи розробки проекту

Стадія розробки	Етапи робіт	Зміст робіт
1	2	3
Технічне завдання 02.01.23 – 31.01.23	Обґрунтування необхідності розробки програми	Коротка характеристика програмного забезпечення; підстава і призначення розробки; вимоги до програмної системи і документація; стадії і етапи розробки програми; порядок контролю і приймання
Ескізний проект 01.02.23 – 26.02.23	Розробка ескізного проекту	Попередня розробка структури вхідних і вихідних даних; уточнення середовища програмування; розробка і опис загальної алгоритмічної структури
Технічний проект 29.02.23 – 19.03.23	Розробка технічного проекту	Уточнення структури вхідних і вихідних даних; розробка докладного алгоритму; розробка структури програми
Робочий проект 20.03.23 – 15.04.23	Розробка програмного забезпечення	Реалізація програмного забезпечення; відлагодження; проведення попереднього тестування
Розробка програмної документації 16.04.23 – 22.04.23	Розробка документації до програмного забезпечення	Розробка необхідної документації, передбаченої технічним завданням
Тестування системи 23.04.23 – 30.04.23	Проведення тестування програмного забезпечення	Розробка методики тестування; проведення основних тестів; коректування програмного забезпечення
Впровадження	Підготовка і передача програми	Підготовка і розгортання програмного забезпечення
Розробка програмної документації 16.04.23 – 22.04.23	Розробка документації до програмного	Розробка необхідної документації, передбаченої технічним завданням

	забезпечення	

6 Порядок контролю та приймання

Контроль здійснюється кінцевими користувачами системи, підключеними на етапі тестування додатку.

ДОДАТОК Б
(обов’язковий)

ДІАГРАМИ

Рисунок Б.1 - Схема бази даних

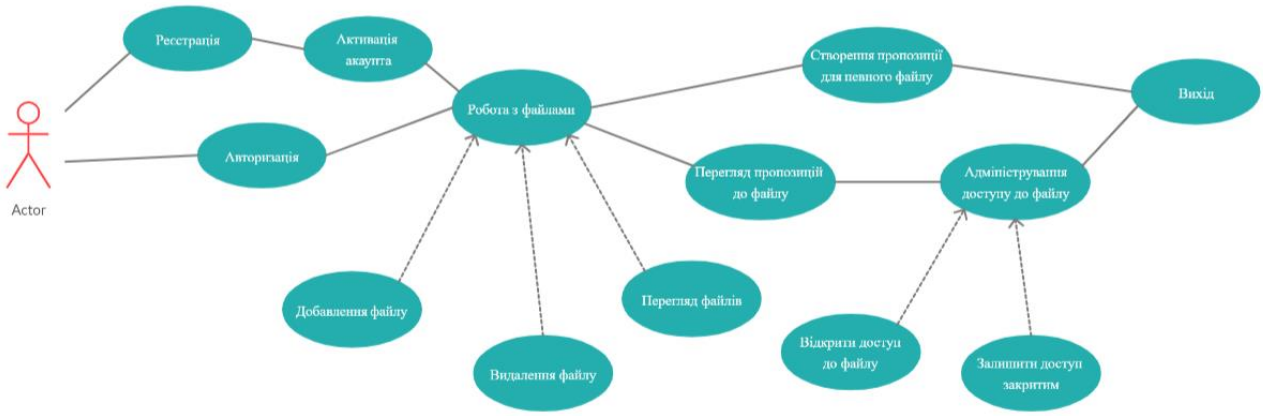


Рисунок Б.2 - Діаграма варіантів використання

ДОДАТОКВ
(обов’язковий)

КОД (ЛІСТИНГ) ПРОГРАМИ

MODELS:

```
<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Category extends Model
{
    private $idYear;
    private $idSubject;
}

<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class File extends Model
{
    /**
     * @var array|string
     */
    private $tittle;
    /**
     * @var array|string
     */
    private $url;
    /**
     * @var array|integer
     */
    private $user_id;
    /**
     * @var array|integer
     */
    private $category_id;
    /**
     * @var array|double
     */
    private $price;
    private $description;
}

<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Message extends Model
{
    private $title;
    private $body;
    private $from;
    private $to;
    private $toRead;

}

<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Offer extends Model
{
    /**
     * @var array|integer
     */
    private $user_id;

    /**
     * @var array|integer
     */
    private $file_id;

    /**
```

```
     * @var array|double
     */
    private $price;

    /**
     * @var array|integer
     */
    private $status;

    /**
     * @var array|string
     */
    private $message;
}


<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class OfferStatus extends Model
{
    private $message_status;
}

<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Role extends Model
{
    //
}


<?php

namespace App;

use Illuminate\Database\Eloquent\Model;


class Status extends Model
{

    private $user_id;
    private $role;
    private $money;

}

<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Subject extends Model
{
    private $nameSubject;
}


<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Transaction extends Model
{
    private $user_id;
    private $message;
    private $money;
    private $balance;
```

```
    private $offer_id;
}
```

```
<?php

namespace App;

use Illuminate\Notifications\Notifiable;
use Illuminate\Foundation\Auth\User as Authenticatable;

class User extends Authenticatable
{
    use Notifiable;

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

    /**
     * The attributes that should be hidden for arrays.
     *
     * @var array
     */
    protected $hidden = [
        'password', 'remember_token',
    ];
}
```

```
<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Year extends Model
{
    private $year;
}
```

REQUEST

```
<?php

namespace App\Http\Requests;

use Illuminate\Foundation\Http\FormRequest;

class CategoryRequest extends FormRequest
{
    /**
     * Determine if the user is authorized to make this request.
     *
     * @return bool
     */
    public function authorize()
    {
        return true;
    }

    /**
     * Get the validation rules that apply to the request.
     *
     * @return array
     */
    public function rules()
    {
        return [
            'year' => 'required',
            'nameSubject'=>'required'
        ];
    }
}
```

```php
<?php

namespace App\Http\Requests;

use Illuminate\Foundation\Http\FormRequest;

class DataSearchRequest extends FormRequest
{
    /**
     * Determine if the user is authorized to make this request.
     *
     * @return bool
     */
    public function authorize()
    {
        return true;
    }

    /**
     * Get the validation rules that apply to the request.
     *
     * @return array
     */
    public function rules()
    {
        return [
            'DataSearch' => 'required'
        ];
    }
}
```

```php
<?php

namespace App\Http\Requests;

use Illuminate\Foundation\Http\FormRequest;

class FileRequest extends FormRequest
{
    /**
     * Determine if the user is authorized to make this request.
     *
     * @return bool
     */
    public function authorize()
    {
        return true;
    }

    /**
     * Get the validation rules that apply to the request.
     *
     * @return array
     */
    public function rules()
    {
        return [
            'tittle'=>'required|min:3|max:50',
            'url'=>'required|min:6|max:100',
            'description' => 'max:100',
            'price' =>'required|min:1|max:6',
            'subject'=>'required'
        ];
    }
}
```

```php
<?php

namespace App\Http\Requests;

use Illuminate\Foundation\Http\FormRequest;

class MessageRequest extends FormRequest
{
    /**
     * Determine if the user is authorized to make this request.
     *
```

```
     * @return bool
     */
    public function authorize()
    {
        return true;
    }

    /**
     * Get the validation rules that apply to the request.
     *
     * @return array
     */
    public function rules()
    {
        return [
//            'getter' => 'required',
            'title' => 'required|max:50',
            'body' => 'required|max:100'
        ];
    }
}
```

```
<?php

namespace App\Http\Requests;

use Illuminate\Foundation\Http\FormRequest;

class MoneyUserRequest extends FormRequest
{
    /**
     * Determine if the user is authorized to make this request.
     *
     * @return bool
     */
    public function authorize()
    {
        return true;
    }

    /**
     * Get the validation rules that apply to the request.
     *
     * @return array
     */
    public function rules()
    {
        return [
            'money' => 'required|min:1'
        ];
    }
}
```

```
<?php

namespace App\Http\Requests;

use Illuminate\Foundation\Http\FormRequest;

class OfferRequest extends FormRequest
{
    /**
     * Determine if the user is authorized to make this request.
     *
     * @return bool
     */
    public function authorize()
    {
        return true;
    }

    /**
     * Get the validation rules that apply to the request.
     *
     * @return array
     */
    public function rules()
```

```
    {
        return [
            'price'=>'required',
            'message'=>'required|min:0|max:50'
        ];
    }
}
```

```
<?php

namespace App\Http\Requests;

use Illuminate\Foundation\Http\FormRequest;

class SubjectRequest extends FormRequest
{
    /**
     * Determine if the user is authorized to make this request.
     *
     * @return bool
     */
    public function authorize()
    {
        return true;
    }

    /**
     * Get the validation rules that apply to the request.
     *
     * @return array
     */
    public function rules()
    {
        return [
            'nameSubject'=>'required|string|max:50|unique:subjects'
        ];
    }
}
```

CONTROLLERS:

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class AboutUsController extends Controller
{
    public function index(){
        return view('about');
    }
}
```

```
<?php

namespace App\Http\Controllers;

use App\File;
use App\Http\Requests\CategoryRequest;
use App\Http\Requests\SubjectRequest;
use App\Subject;
use App\Year;
use App\Category;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\DB;
use PhpParser\Node\Expr\Array_;

class CategoryController extends Controller
{
    public function index()
    {
        $cat = Category::all();
        $year = Year::all();
        $subject = Subject::all();
        $arr = array($cat, $year, $subject);
        return view('admin/category', ['category' => $arr]);
```

```
}

static function getYearById($id)
{
    $years = DB::table('years')
        ->where('id', '=', $id)
        ->get();
    return $years[0]->year;
}

static function getSubjectNameById($id)
{
    $subject = DB::table('subjects')
        ->where('id', '=', $id)
        ->get();
    return $subject[0]->nameSubject;
}

public function indexSubject()
{
    $subjects = Subject::all();
    return view('admin/subject', ['data' => $subjects]);
}

public function addSubject(SubjectRequest $req)
{
    $subjects = new Subject();
    $subjects->nameSubject = $req->input('nameSubject');
    $subjects->save();
    return redirect()->route('showSubjects');
}

public function deleteSubject($id)
{
    if ($id != 1) {
        $category = DB::table('categories')
            ->where('idSubject','=',$id)
            ->count();

        if($category != 0 ){
            $categories = DB::table('categories')
                ->where('idSubject','=',$id)
                ->get();
            foreach ($categories as $cat){
                self::deleteForSubAndCategory($cat->id);
            }
        }
        $subject = Subject::find($id)->delete();
        return redirect()->route('showSubjects')->with('success', 'Subject was deleted');
    }
    else{
        return redirect()->route('showSubjects')->with('danger', 'Other cannot be deleted');
    }
}

/**
 * @param CategoryRequest $req
 * @return \Illuminate\Http\RedirectResponse
 */
public function addCategory(CategoryRequest $req)
{
    $inputSubject = $req->input('nameSubject');
    $inputYear = $req->input('year');

    if (
        $inputSubject == "Other" && $inputYear == '-'
        ||
        $inputSubject == "Other" && $inputYear != '-'
        ||
        $inputSubject != "Other" && $inputYear == '-'
        ) {
        return redirect()->route('showCategory')->with('danger', 'Sorry, `Other` are connected with `-`');
    } else {
        $subjects = DB::table('subjects')
            ->where('nameSubject', '=', $inputSubject)
            ->get();
        $years = DB::table('years')
            ->where('year', '=', $inputYear)
            ->get();
```

```
            $cat = new Category();
            $cat->idYear = $years[0]->id;
            $cat->idSubject = $subjects[0]->id;
            $cat->save();
            return redirect()->route('showCategory');
        }
    }

    /**
     * @param $id_category
     */
    static function deleteForSubAndCategory($id_category)
    {
        $files = File::all();
        foreach ($files as $file) {
            if ($file->category_id == $id_category) {
                DB::table('files')
                    ->where('id', '=', $file->id)
                    ->where('category_id', '=', $id_category)
                    ->update(['category_id' => 1]);
            }
        }
        $category = Category::find($id_category)->delete();
    }

    public function deleteCategory($id)
    {
        $category = Category::find($id);
        if ($category->id == 1) {
            return redirect()->route('showCategory')->with('danger', 'Sorry, `Other` and `-
` cannot be deleted');
        } else {
            self::deleteForSubAndCategory($category->id);
            return redirect()->route('showCategory')->with('success', 'Category was deleted');
        }
    }
}
```

```
<?php

namespace App\Http\Controllers;

use App\Http\Requests\MessageRequest;
use App\Message;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\DB;
use Illuminate\Support\Facades\Mail;

class ContactController extends Controller
{
    public function index(){
        if(StatusController::checkUserStatus(auth()->user()->id)){
            return redirect()->route('home');
        }
        else{
            return view('contact');
        }
    }

    public function addMessage(MessageRequest $request){
        $admin = DB::table('statuses')
            ->where('role', '=', 2)
            ->get();

        $mail = new Message();
        $mail->title = $request->input('title');
        $mail->body = $request->input('body');
        $mail->from = auth()->user()->id;
        $mail->to = $admin[0]->user_id;
        $mail->toRead = 0;
        $mail->save();

        return redirect()->route('contact')->with('success', 'Email was sent!');
    }
}
```

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Foundation\Bus\DispatchesJobs;
use Illuminate\Routing\Controller as BaseController;
use Illuminate\Foundation\Validation\ValidatesRequests;
use Illuminate\Foundation\Auth\Access\AuthorizesRequests;

class Controller extends BaseController
{
    use AuthorizesRequests, DispatchesJobs, ValidatesRequests;
}
```

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\File;
use Illuminate\Support\Facades\DB;



class FileController extends Controller
{
    public function __construct()
    {
        $this->middleware('auth');
    }

    static function getNameById($id){
        $file = File::find($id);
        return $file->tittle;
    }

    static function getUserById($id){
        $file = File::find($id);
        return $file->user_id;
    }

    static function getPriceById($id){
        $file = File::find($id);
        return $file->price;
    }

    static function hasOffer($file){

        $offers = DB::table('offers')
            ->get();
        foreach ($offers as $el){
            if(
                $el->toDelete != 1 &&
                $el->file_id == $file->id &&
                $el->status == 1
            ){
                //If the file has any offers
                return true;
            }
        }

        //If the file can be deleted
        return false;
    }
}
```

```php
<?php

namespace App\Http\Controllers;

use App\Category;
use App\Subject;
use App\Year;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\DB;

class HomeController extends Controller
```

```
{
    /**
     * Create a new controller instance.
     *
     * @return void
     */
    public function __construct()
    {
        $this->middleware('auth');
    }

    /**
     * Show the application dashboard.
     *
     * @return \Illuminate\Http\Response
     */
    public function index()
    {
        $id = auth()->user()->id;
        if(!StatusController::checkUserStatus($id) && StatusController::getUserStatus($id) == 4){
            return redirect()->route('contact')->with('danger', 'Ban za parushennya!');
        }
        else{
            $files = DB::table('files')
                ->where('user_id', '<>', auth()->user()->id)
                ->where('toDelete', '<>', 1)
                ->get();
            $year = DB::table('years')
                ->get();
            $subject = DB::table('subjects')
                ->get();
            $arr = array($year,$subject,$files);
            return view('home', ['data' => $arr]);
        }
    }
    public function filter(Request $req){
        $subject = DB::table('subjects')
            ->where('nameSubject','=',$req->subject)
            ->get();
        $year = DB::table('years')
            ->where('year','=',$req->year)
            ->get();
        $category = DB::table('categories')
            ->where('idYear','=',$year[0]->id)
            ->where('idSubject','=',$subject[0]->id)
            ->get();
        if (sizeof($category) != 0) {
            $category = Category::find($category[0]->id);
            $files = DB::table('files')
                ->where('category_id', '=', $category->id)
                ->where('user_id', '<>', auth()->user()->id)
                ->where('toDelete', '<>', 1)
                ->get();
            $year = Year::all();
            $subject = Subject::all();
            $arr = array($year, $subject, $files);
            return view('home', ['data' => $arr]);
        } else {
            $year = Year::all();
            $subject = Subject::all();
            $files = null;
            $arr = array($year, $subject, $files);
            return view('home',['data'=>$arr]);
        }
    }
}


<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Support\Facades\DB;

class MessageController extends Controller
{
    static function newMessages(){
        $mess = DB::table('messages')
```

```
            ->where('toRead', '=', 0)
            ->count();
        return $mess;
    }
}
```

```
<?php

namespace App\Http\Controllers;

use App\File;
use App\Http\Requests\OfferRequest;
use App\Offer;
use App\OfferStatus;
use App\Status;
use App\Transaction;
use App\Message;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\DB;

class OfferController extends Controller
{

    public function __construct()
    {
        $this->middleware('auth');
    }

    /**
     * @param $id
     * @return \Illuminate\Contracts\View\Factory|\Illuminate\View\View
     */
    public function index($id){
        $file = File::find($id);
        $offers = DB::table('offers')
            ->where('file_id', '=', $id)
            ->get();

        $data = array($file, $offers);
        return view('order', ['data' => $data]);
    }

    /**
     * @param $file_id
     * @param OfferRequest $req
     * @return \Illuminate\Http\RedirectResponse
     */
    public function addOrderFile($file_id, OfferRequest $req){
        if(UserController::getMoneyById(auth()->user()->id) < $req->input('price')){
            return redirect()->route('orderFile', $file_id)->with('danger', 'Not enough money!');
        }
        else{
            $offer = new Offer();
            $offer->user_id = auth()->user()->id;
            $offer->file_id = $file_id;
            $offer->price = $req->input('price');
            $offer->message = $req->input('message');

            $status = Status::find($offer->user_id);
            $status->money -= $offer->price;
            $status->save();

            $offer->save();

            $trans = new Transaction();
            $trans->user_id = auth()->user()->id;
            $trans->message = 'Make offer';
            $trans->money = $req->input('price');
            $trans->balance = UserController::getMoneyById($offer->user_id);
            $trans->offer_id = $offer->id;
            $trans->save();



            return redirect()->route('orderFile', $file_id)->with('success', 'Your offer was added');
        }
    }
```

```
/**
 * @param $id
 * @return mixed
 */
public function getOffersById($id){
    $offer = DB::table('offers')
        ->where('user_id', '=', $id)
        ->get();
    return $offer;
}

/**
 * @param $id
 * @return \Illuminate\Http\RedirectResponse
 */
public function confirmOffer($id){
    $offer = Offer::find($id);
    $statusOffer = DB::table('offer_statuses')
        ->where('id','=','2')
        ->get();
    $offer->status = $statusOffer[0]->id;
    $offer->save();

    $status = Status::find(auth()->user()->id);
    $status->money += $offer->price;
    $status->save();

    $trans1 = new Transaction();
    $trans1->user_id = $offer->user_id;
    $trans1->message = 'Confirm offer';
    $trans1->money = $offer->price;
    $trans1->balance = UserController::getMoneyById($offer->user_id);
    $trans1->offer_id = $offer->id;
    $trans1->save();

    $trans2 = new Transaction();
    $trans2->user_id = FileController::getUserById($offer->file_id);
    $trans2->message = 'Buy file';
    $trans2->money = $offer->price;
    $trans2->balance = UserController::getMoneyById(FileController::getUserById($offer->file_id));
    $trans2->offer_id = $offer->id;
    $trans2->save();


    return redirect()->route('userProfileFiles')->with('success', 'Transaction saved');
}

public function repelOffer($id){
    self::returnMoney($id);

    return redirect()->route('userProfileFiles')->with('success', 'Transaction saved');
}

/**
 * @param $id
 * @return \Illuminate\Http\RedirectResponse
 */
public function deleteOfferFile($id){
    $offer = Offer::find($id);
    if($offer->status != 1){
        $offer->toDelete = 1;
        $offer->save();

        return redirect()->route('userProfileOffer')->with('success', 'Offer was deleted!');
    }
    else{
        $status = Status::find($offer->user_id);
        $status->money += $offer->price;
        $status->save();
        $offer->toDelete = 1;
        $offer->save();

        $trans = new Transaction();
        $trans->user_id = $offer->user_id;
        $trans->message = 'Delete offer';
```

```
        $trans->money = $offer->price;
        $trans->balance = UserController::getMoneyById($offer->user_id);
        $trans->offer_id = $offer->id;
        $trans->save();

        return redirect()->route('userProfileOffer')->with('success', 'Offer was deleted!');
    }
}



/**
 * @param Request $req
 * @return \Illuminate\Contracts\View\Factory|\Illuminate\View\View
 */
public function filterOffer(Request $req){
    $status_offer = DB::table('offer_statuses')
        ->where('message_status','=',$req->message_status)
        ->get();
    $offers = DB::table('offers')
        ->where('user_id', '=', auth()->user()->id)
        ->where('toDelete', '<>', 1)
        ->where('status','=',$status_offer[0]->id)
        ->get();
    $status_offer = OfferStatus::all();
    $arr = array($status_offer,$offers);
    return view('user/profileOffers', ['data' => $arr]);
}

/**FOR FILES
 * @param $id
 * @return mixed
 */
static function parseDate($id){
    $file = File::find($id);
    return $file->created_at->format('d.m.Y H:i:s');
}

/**
 * @param $id
 * @return mixed
 */
static function parseDateForTransactions($id){
    $trans = Transaction::find($id);
    return $trans->created_at->format('d.m.Y H:i:s');
}

/**
 * @param $id
 * @return mixed
 */
static function parseDateForOffers($id){
    $offer = Offer::find($id);
    return $offer->created_at->format('d.m.Y H:i:s');
}

/**
 * @param $id
 * @return mixed
 */
static function parseDateForMessages($id){
    $message = Message::find($id);
    return $message->created_at->format('d.m.Y H:i:s');
}

/**
 * @param $status
 * @return mixed
 */
static function getStatusById($status){
    $statusOffer = DB::table('offer_statuses')
        ->where('id','=',$status)
        ->get();
    return $statusOffer[0]->message_status;
}

/**
 * @param $id
 * @return string
```

```
     */
    static function getFileUrl($id)
    {
        $offer = Offer::find($id);
        if ($offer->status == 2) {
            $files = File::find($offer->file_id);
            return $files->url;
        } else {
            return '**********';
        }
    }
    static function newOffers(){
        $newOffers = 0;
        $files = DB::table('files')
            ->where('user_id', '=', auth()->user()->id)
            ->where('toDelete', '=', 0)
            ->get();
        foreach ($files as $file) {
            $newOffers += DB::table('offers')
                ->where('file_id', '=', $file->id)
                ->where('status', '=', 1)
                ->where('toDelete', '=', 0)
                ->count();
        }
        return $newOffers;
    }

    /**
     * @param $id
     */
    static function returnMoney($id){
        $offer = Offer::find($id);

        $statusOffer = DB::table('offer_statuses')
            ->where('id','=','3')
            ->get();
        $offer->status = $statusOffer[0]->id;
        $offer->save();

        $status = Status::find($offer->user_id);
        $status->money += $offer->price;
        $status->save();

        $trans = new Transaction();
        $trans->user_id = $offer->user_id;
        $trans->message = 'Repel offer';
        $trans->money = $offer->price;
        $trans->balance = UserController::getMoneyById($offer->user_id);
        $trans->offer_id = $offer->id;
        $trans->save();
    }
}


<?php

namespace App\Http\Controllers;

use App\Category;
use App\File;
use App\Http\Requests\FileRequest;
use App\Http\Requests\MoneyUserRequest;
use App\Message;
use App\Status;
use App\Subject;
use App\Transaction;
use App\Year;
use Illuminate\Http\Request;
use App\User;
use Illuminate\Support\Facades\DB;

class ProfileController extends Controller
{

    public function __construct()
    {
        $this->middleware('auth');
    }
```

```
/**
 * @return \Illuminate\Contracts\View\Factory|\Illuminate\View\View
 */


public function index(){
    $user_id = auth()->user()->id;
    $user = new User;

    if(StatusController::getUserStatus($user_id) == 4){
        return redirect()->route('contact')->with('danger', 'Ban za parushennya!');
    }
    else{
        $status = DB::table('statuses')
            ->where('user_id', '=', $user_id )
            ->get();

        if($status[0]->role == 2){
            $files = DB::table('files')
                ->get();
            return view('admin/profileFiles', ['data' => $files]);
        }
        else{

            $files = DB::table('files')
                ->where('user_id', '=', $user_id )
                ->where('toDelete', '<>', 1)
                ->get();
            $offer = DB::table('offers')->get();

            $data = array($files, $offer);
            return view('user/profileFiles', ['data' => $data]);
        }
    }

}



/**
 * @return \Illuminate\Contracts\View\Factory|\Illuminate\View\View
 */
public function addFileIndex(){
    $category = DB::table('categories')
        ->get();
    return view('user/profileAddFile',['data'=>$category]);
}


/**
 * @return \Illuminate\Contracts\View\Factory|\Illuminate\View\View
 */
public function offerIndex(){
    $offers = DB::table('offers')
        ->where('user_id', '=', auth()->user()->id)
        ->where('toDelete', '<>', 1)
        ->get();
    $status_offer = DB::table('offer_statuses')
        ->get();
    $arr = array($status_offer,$offers);
    return view('user/profileOffers', ['data' => $arr]);
}


/**
 * @return \Illuminate\Contracts\View\Factory|\Illuminate\View\View
 */
public function transactionIndex(){
    $trans = DB::table('transactions')
        ->where('user_id', '=', auth()->user()->id)
        ->orderBy('created_at', 'DESC')
        ->get();
    return view('user/profileTransactions', ['data' => $trans]);
}

public function listOfTransactions(){
    $trans = DB::table('transactions')
        ->orderBy('created_at', 'DESC')
```

```
        ->get();
    return view('admin/profileTransactions', ['data' => $trans]);
}

/**
 * @param $id
 * @return \Illuminate\Contracts\View\Factory|\Illuminate\View\View
 */
public function userMoreIndex($id){
    $listUsers = DB::table('statuses')
        ->where('user_id', '<>', auth()->user()->id)
        ->get();
    $user = DB::table('statuses')
        ->where('user_id', '=', $id)
        ->get();
    $trans = DB::table('transactions')
            ->where('user_id', '=', $id)
            ->get();

    $data = array($listUsers, $user, $trans);

    return view('admin/profileUser_Bank', ['data' => $data]);
}


/**
 * @return \Illuminate\Contracts\View\Factory|\Illuminate\View\View
 */
public function listOfUsers(){
    $user = new User;
    $listUsers = DB::table('statuses')
        ->where('user_id', '<>', auth()->user()->id)
        ->where('role', '<>', 2)
        ->get();
    return view('admin/profileUsers', ['data' => $listUsers]);
}
/**
 * @param $id
 * @return \Illuminate\Http\RedirectResponse
 */
public function banUser($id){
    $status = Status::find($id);
    if($status->role == 4){
        $status->role = 3;
    }elseif($status->role = 3){
        $status->role = 4;
    }
    $status->save();
    return redirect()->route('adminProfileUsers');
}

/**
 * @param $id
 * @param MoneyUserRequest $request
 * @return \Illuminate\Http\RedirectResponse
 */
public function addMoney($id, MoneyUserRequest $request){
    $status = Status::find($id);
    $money = $request->input('money');
    if(($status->money + $money) < 0){
        $status->money = 0;
    }
    else{
        $status->money += $money;
    }
    $status->save();

    $trans = new Transaction();
    $trans->user_id = $id;
    $trans->message = 'Present from admin';
    $trans->money = $money;
    $trans->balance = UserController::getMoneyById($id);
    $trans->offer_id = null;
    $trans->save();

    return redirect()->route('adminProfileUsersMore', $id)->with('success', 'Transaction was success!');
}
```

```
static function showNameSubjectByCategoryId($id){
    $files = DB::table('files')
        ->where('category_id','=',$id)
        ->get();
    $category = Category::find($files[0]->category_id);
    $subject = Subject::find($category->idSubject);
    return $subject->nameSubject;
}

static function showYearByCategoryId($id){
    $category = DB::table('categories')
        ->where('id','=',$id)
        ->get();
    $idYear = CategoryController::getYearById($category[0]->idYear);
    return $idYear;
}

public function addFile(Request $request){
    $file = new File();
    $subject = DB::table('subjects')
        ->where('nameSubject','=',$request->subject)
        ->get();
    $category= DB::table('categories')
        ->where('idSubject','=',$subject[0]->id)
        ->get();
    $file->tittle = $request->input('tittle');
    $file->user_id = auth()->user()->id;
    $file->url = $request->input('url');
    $file->price = $request->input('price');
    $file->category_id = $category[0]->id;
    $file->description = $request->input('description');
    $file->save();
    return redirect()->route('userProfileFiles')->with('success', 'File was added!');
}
public function deleteFile($id){
    $file = File::find($id);
    if(FileController::hasOffer($file)){

        $statusOffer = DB::table('offer_statuses')
            ->where('id','=', 1)
            ->get();


        $offers = DB::table('offers')
            ->where('file_id','=',$id)
            ->where('status', '=', $statusOffer[0]->id)
            ->get();
        foreach ($offers as $offer){
            OfferController::returnMoney($offer->id);
        }
        $file->toDelete = 1;
        $file->save();

        return redirect()->route('userProfileFiles')->with('danger', 'File was deleted!');
    }
    else{
        $file->toDelete = 1;
        $file->save();
        return redirect()->route('userProfileFiles')->with('success', 'File was deleted!');
    }
}
public function listOfMessages(){
    $mess_0 = DB::table('messages')
        ->where('toRead', '=', 0)
        ->orderBy('created_at', 'DESC')
        ->get();
    $mess_1 = DB::table('messages')
        ->where('toRead', '=', 1)
        ->orderBy('created_at', 'DESC')
        ->get();

    $data = array($mess_0, $mess_1);

    return view('admin/profileMessages', ['data' => $data]);
}

public function deleteMessage($id){
    $mess = Message::find($id)->delete();
    return redirect()->route('adminProfileMessages')->with('success', 'Message was deleted');
```

```
    }

    public function deleteAllMessage(){
        if(DB::table('messages')->where('toRead', '=', 1)->count() == 0){
            return redirect()->route('adminProfileMessages')->with('danger', 'No more read message');
        }
        else{
            $mess = DB::table('messages')
                ->where('toRead', '=', 1)
                ->delete();
            return redirect()->route('adminProfileMessages')->with('success', 'All messages was deleted');
        }
    }

    public function doneMessage($id){
        $mess = Message::find($id);
        $mess->toRead = 1;
        $mess->save();

        return redirect()->route('adminProfileMessages');
    }
}

<?php

namespace App\Http\Controllers;

use App\Status;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\DB;

class StatusController extends Controller
{

    public function __construct()
    {
        $this->middleware('auth');
    }

    /**
     * @param $id
     * @return bool
     */
    static function checkUserStatus($id){
        $status = DB::table('statuses')
            ->where('user_id', '=', $id );
        if($status->count() == 0){
            return true;
        }
        else{
            return false;
        }
    }

    /**
     * @param $id
     * @return integer
     */
    static function getStatusById($id){
        $status = DB::table('statuses')
            ->where('user_id', '=', $id )->get();
        $role = DB::table('roles')
            ->where('id', '=', $status[0]->role)->get();
        return $role[0]->role;
    }

    /**
     * @param $id
     * @return mixed
     */
    static function getUserStatus($id){
        $status = Status::find($id);
        return $status->role;
    }

    /**
     * @return \Illuminate\Http\RedirectResponse
     */
    static function addStatusForUser(){
```

```
        $status = new Status();
        $status->user_id = auth()->user()->id;

        $status->save();
        return redirect()->route('home')->with('success', 'Account activated');
    }
}
```

```
<?php

namespace App\Http\Controllers;

use App\File;
use App\Http\Requests\DataSearchRequest;
use App\Offer;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\DB;
use Symfony\Component\VarDumper\Cloner\Data;

class TransactionController extends Controller
{

    public function __construct()
    {
        $this->middleware('auth');
    }

    static function getFileName($offer_id){
        $offer = Offer::find($offer_id);
        $file = File::find($offer->file_id);
        return $file->tittle;
    }

    static function getUserName($offer_id){
        $offer = Offer::find($offer_id);
        $file = File::find($offer->file_id);
        return UserController::getNameById($file->user_id);
    }

    static function getBuyerName($offer_id){
        $offer = Offer::find($offer_id);
        return UserController::getNameById($offer->user_id);
    }

    static function getDataFormat($el){
            $time = strtotime($el);
            $newFormat = date('Y-m-d',$time);
            return $newFormat;
    }
}
```

```
<?php

namespace App\Http\Controllers;

use App\User;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\DB;

class UserController extends Controller
{

    public function __construct()
    {
        $this->middleware('auth');
    }

    static function getNameById($id){
        $user = User::find($id);
        return $user->name;
    }

    static function getEmailById($id){
        $user = User::find($id);
        return $user->email;
    }
```

```
    static function getMoneyById($id){
        $status = DB::table('statuses')
            ->where('user_id', '=', $id)
            ->get();
        return $status[0]->money;
    }


}
```

ROUTS:

```
<?php


Route::get('/', function () {
    return view('welcome');
})->name('welcome');

Auth::routes();

//Main
Route::get('/home', 'HomeController@index')->name('home');
Route::get('/contact', 'ContactController@index')->name('contact');
Route::post('/contact', 'ContactController@addMessage')->name('addMessage');
Route::get('/about', 'AboutUsController@index')->name('about-us');
Route::get('/home/filter','HomeController@filter')->name('filter');


//Profile page USER
Route::get('/profile/Files', 'ProfileController@index')->name('userProfileFiles');
Route::get('/profile/Offers', 'ProfileController@offerIndex')->name('userProfileOffer');
Route::get('/profile/Transactions', 'ProfileController@transactionIndex')->name('userProfileTransaction');

//File action
Route::post('/profile/Files', 'ProfileController@addFile')->name('addFile');
Route::post('/profile/Files/{id}', 'ProfileController@deleteFile')->name('deleteFile');
Route::get('/profile/Files/Add', 'ProfileController@addFileIndex')->name('userProfileFilesAdd');

//Offer action
Route::post('/profile/Offers/{id}', 'OfferController@deleteOfferFile')->name('userProfileOfferDelete');
Route::get('/profile/Offers/Info', 'ProfileController@offerIndex')->name('userProfileOfferInfo');
Route::get('/profile/Offers/filter','OfferController@filterOffer')->name('filterOffers');
Route::post('/profile/Offers/Confirm/{id}', 'OfferController@confirmOffer')->name('userProfileOfferConfirm');
Route::post('/profile/Offers/Repel/{id}', 'OfferController@repelOffer')->name('userProfileOfferRepel');



//Profile page ADMIN
Route::get('/profileUsers', 'ProfileController@listOfUsers')->name('adminProfileUsers');
Route::get('/profileFiles', 'ProfileController@index')->name('adminProfileFiles');
Route::post('/profileUsers/Ban/{id}', 'ProfileController@banUser')->name('adminProfileUsersBan');
Route::get('/profileMessages', 'ProfileController@listOfMessages')->name('adminProfileMessages');
Route::get('/profileUsers/More/{id}', 'ProfileController@userMoreIndex')->name('adminProfileUsersMore');
Route::post('/profileUsers/More/{id}/AddMoney', 'ProfileController@addMoney')-
>name('adminProfileUsersAddMoney');
Route::get('/profileTransactions', 'ProfileController@listOfTransactions')->name('adminProfileTransactions');

//Category
Route::get('/Category', 'CategoryController@index')->name('showCategory');
Route::get('/Subject', 'CategoryController@indexSubject')->name('showSubjects');
Route::post('/Subject', 'CategoryController@addSubject')->name('addSubjects');
Route::post('/Subject/Delete/{id}', 'CategoryController@deleteSubject')->name('deleteSubjects');
Route::post('/Category', 'CategoryController@addCategory')->name('addCategories');
Route::post('/Category/Delete/{id}', 'CategoryController@deleteCategory')->name('deleteCategory');

//Status route
Route::post('/addStatusForUser', 'StatusController@addStatusForUser')->name('addStatusForUser');

//Order route
Route::get('/order/{idfile}', 'OfferController@index')->name('orderFile');
Route::post('/order/{idfile}', 'OfferController@addOrderFile')->name('addOrderFile');

//Messages action
Route::post('/profileMessages/Delete/{id}', 'ProfileController@deleteMessage')-
>name('adminProfileDeleteMessage');
Route::post('/profileMessages/DeleteAll', 'ProfileController@deleteAllMessage')-
>name('adminProfileDeleteAllMessages');
Route::post('/profileMessages/Ignore/{id}', 'ProfileController@doneMessage')->name('adminProfileDoneMessage');
```

ДОДАТОК Г
(обов’язковий)

ПРЕЗЕНТАЦІЙНІ МАТЕРІАЛИ

Хмельницький Національний університет
Факультет інформаційних технологій
Кафедра інженерії програмного забезпечення

Кваліфікаційна робота на тему:
“Веб-платформа ‘LabHub’ для обміну інформацією на базі фреймворка Laravel”

Виконав студент Варук Валентин Костянтинович
Керівник: Бедратюк Л.П. д-р фіз.-мат. наук, проф.

Актуальність теми

З кожним роком серед студентів спостерігається зростання кількості інтелектуальної праці, яка реалізується в різних напрямках та сферах навчання. Вони займаються розробкою різноманітних цікавих проектів, вкладаючи в них різну кількість часу і зусиль. Ці проекти можуть бути збережені у вигляді коду на платформі GitHub, але існує ймовірність, що вони можуть бути випадково видалені або втрачені через технічні проблеми, такі як форматування переносного носія або видалення з корзини на комп'ютері.

Ці творчі зусилля студентів часто залишаються лише в рамках позитивних оцінок від викладачів та важливого професійного досвіду. Однак, інформація має велике значення у розвитку інтелектуальних здібностей людини. З появою Інтернету зникла необхідність шукати документацію у книгах або газетах, переглядати новини та здійснювати інші пошуки. Зараз достатньо мати підключення до Wi-Fi або мобільного Інтернету, і усю необхідну інформацію можна знайти онлайн.

В результаті роботи було розроблено веб-застосунок для обміну інформації. Така платформа можу включає в себе функціональні можливості для ефективного пошуку опублікованої інформації, управління цією інформацією та зворотній зв`язок з адміністратором.

Мета та завдання проекту

Мета проекту - розробка веб-застосунку, який забезпечує покращення взаємодії між компанією-забудовником та її клієнтами, що в свою чергу призведе до збільшення ефективності роботи та підвищення рівня задоволення клієнтів.
Для досягнення поставленої мети необхідно виконати наступні завдання:

- дослідження предметної області веб-платформ для обміну інформацією з метою визначення потреб потенційних користувачів;
- аналіз існуючих рішень на ринку;
- розробити технічне завдання;
- розробити архітектуру веб-застосунку та бази даних, враховуючи потреби користувачів;
- обрати технології для розробки;
- розробити зручний та привітний інтерфейс для користувачів;
- розробити веб-застосунок;
- протестувати готовий веб-застосунок.

Порівняння наявного ПЗ

Основною перевагою наявних сайтів можна виділити швидкість передачі файлів та папок, отримання доступу з різних пристроїв, збереження анонімності, та деякі сервіси мають можливість створювати бекапи даних, щоб забрати можливість втрати даних.

На жаль сервіси деякі сервіси мають безкоштовний період в декілька днів або завантажених файлах, і після закінчення цього періода потрібно оформити підписку щоб продовжувати працювати або всі файли які були завантажені в систему будуть видалені. Ще можна відилити те, що сервіси мають застарілий дизайн.

Основними недоліками є те, що більшість платформ містить багато інформації яка була додана довгий період тому, і теперішній момент може не працювати і містити застарілі дані.

Функціональні вимоги

Для звичайного користувача:

- перегляд наявних в нього умовних одиниць коштів
- перегляд та фільтрація опублікованих файлів іншими користувачами
- створення та відправлення повідомлення адміністраторам
- створення та видалення власних публікацій
- створення та видалення власних замовлень
- перегляд та пошук за датою проведених транзакцій

Для адміністратора:

- перегляд та пошук за певними критеріями опублікованих файлів
- видалення опублікованих файлів користувачів
- перегляд та пошук за датою проведення загального журналу транзакцій
- перегляд та пошук авторизованих користувачів
- можливість блокування користувачів
- створення та видалення нових навчальних предметів
- створення та видалення нових категорій

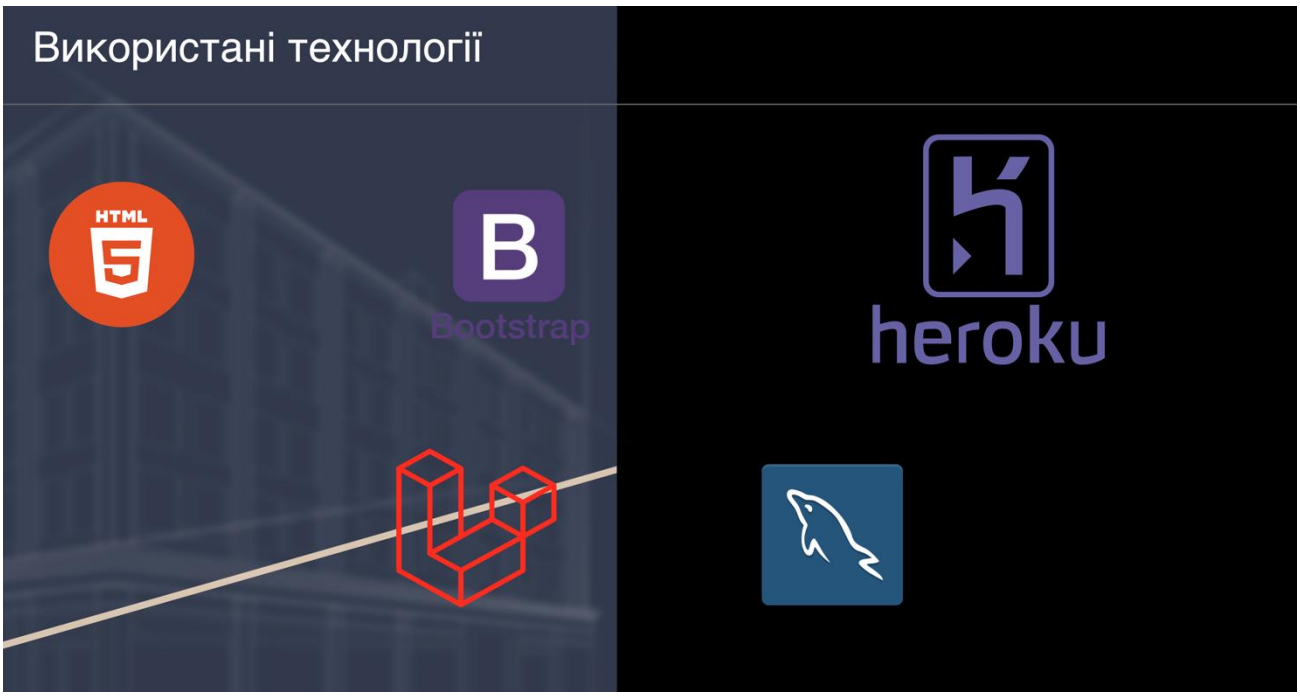
Для заблокованого користувача:

- зв`язок з адміністрацією

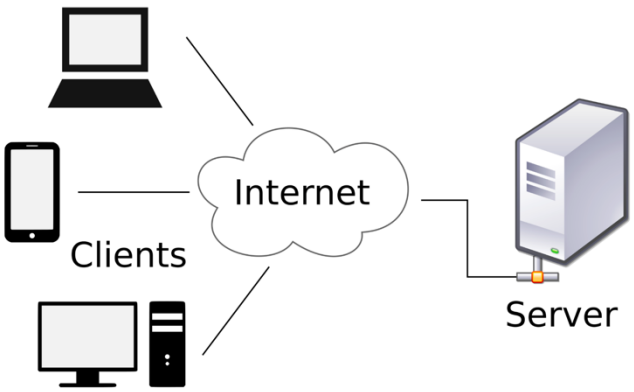
Для не авторизованого користувача:

- можливість авторизації або регістрації

Діаграма варіантів використання



Клієнт серверна архітектура



Архітектура рівня представлення

MVC - Model View Controller

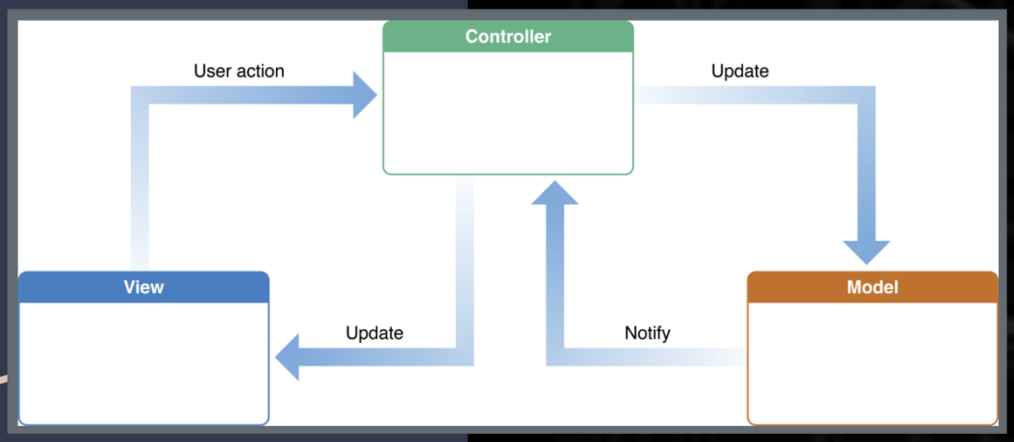
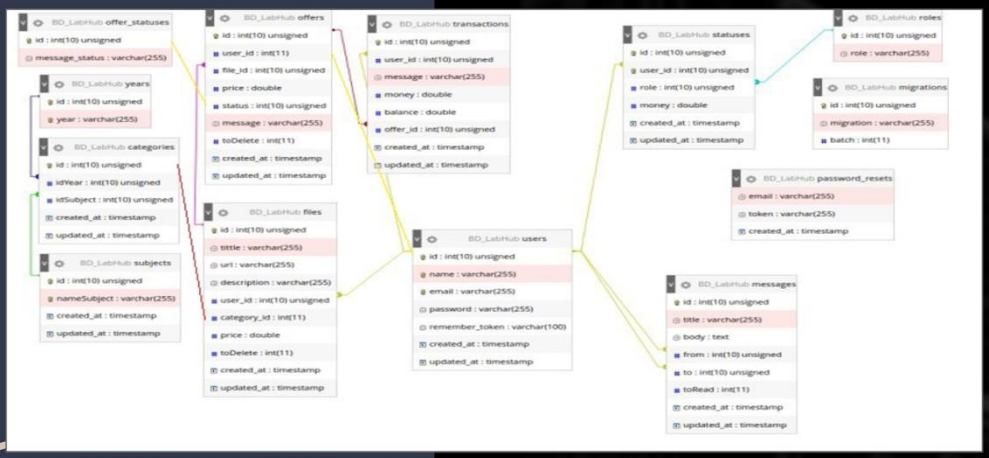
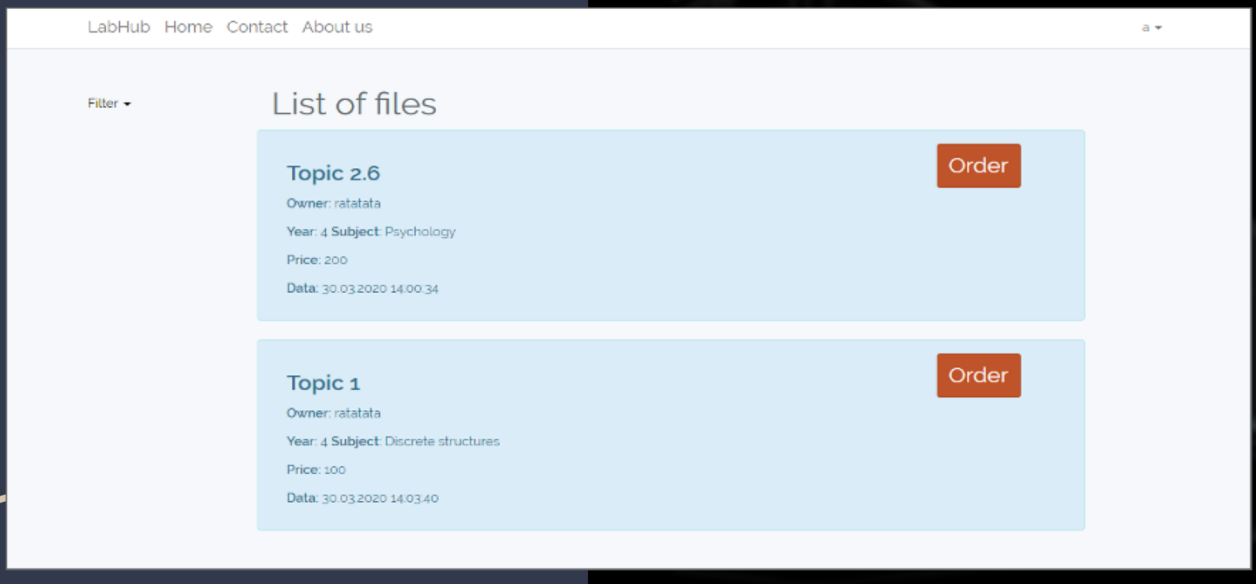


Схема бази даних

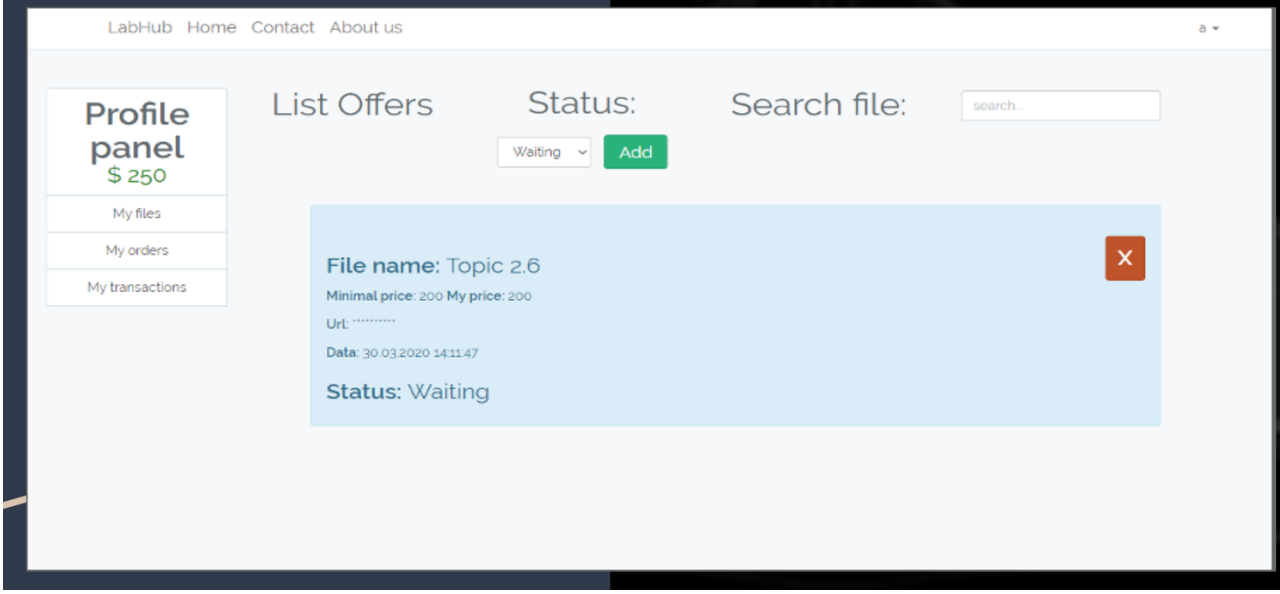


Реалізація проекту. Головна сторінка

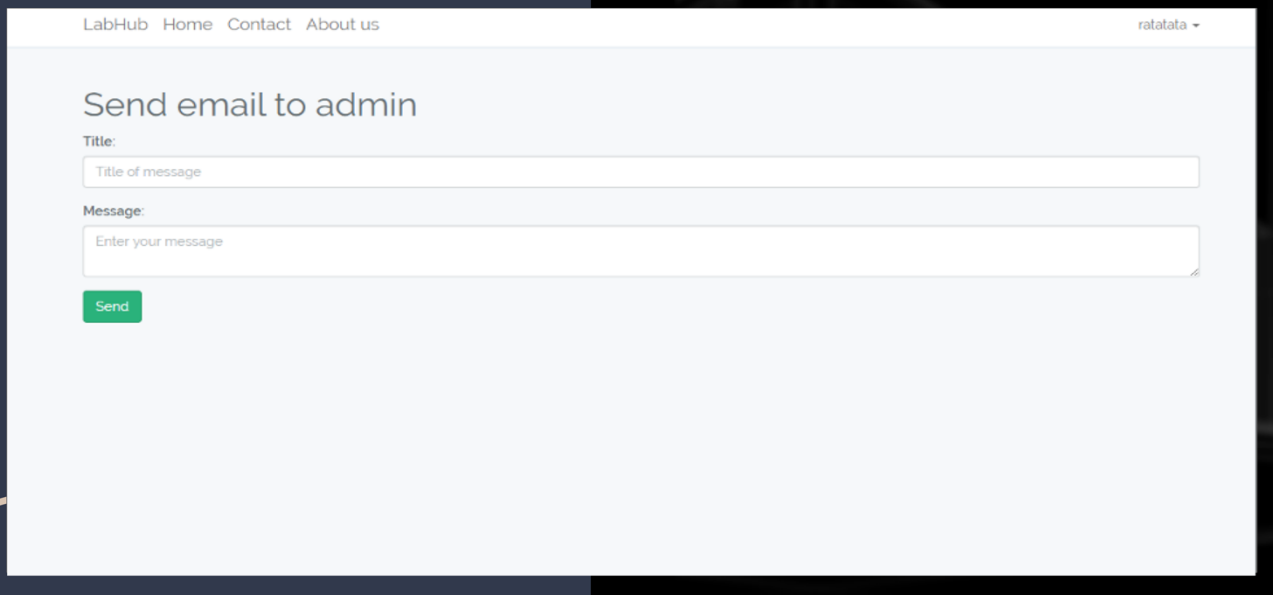
Реалізація проекту. Домашня сторінка



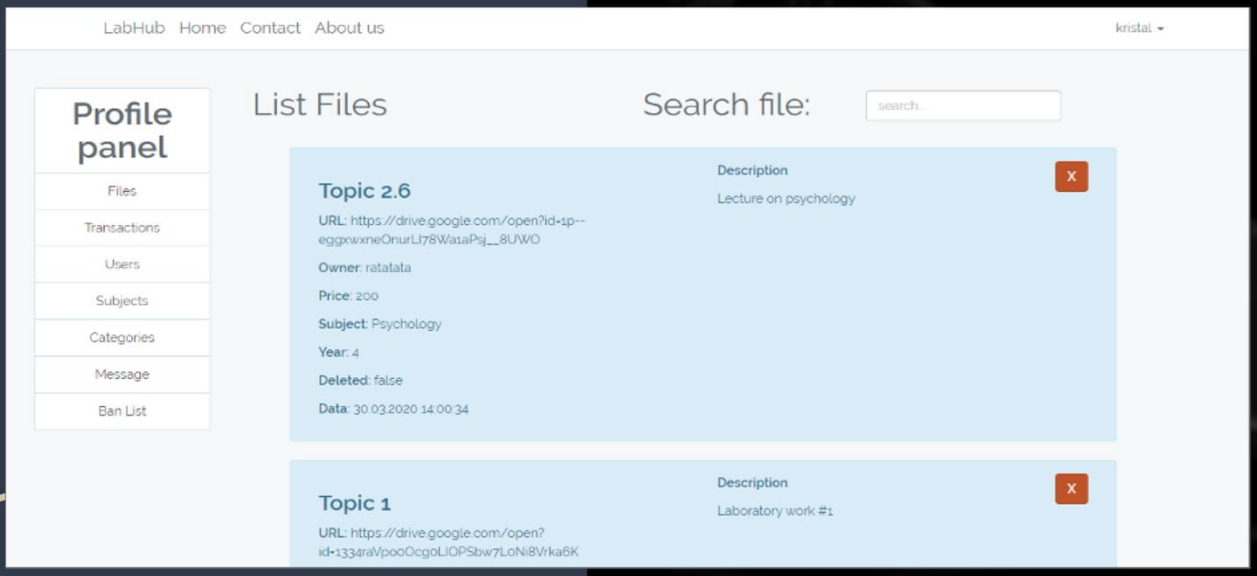
Реалізація проекту. Сторінка профіля користувача



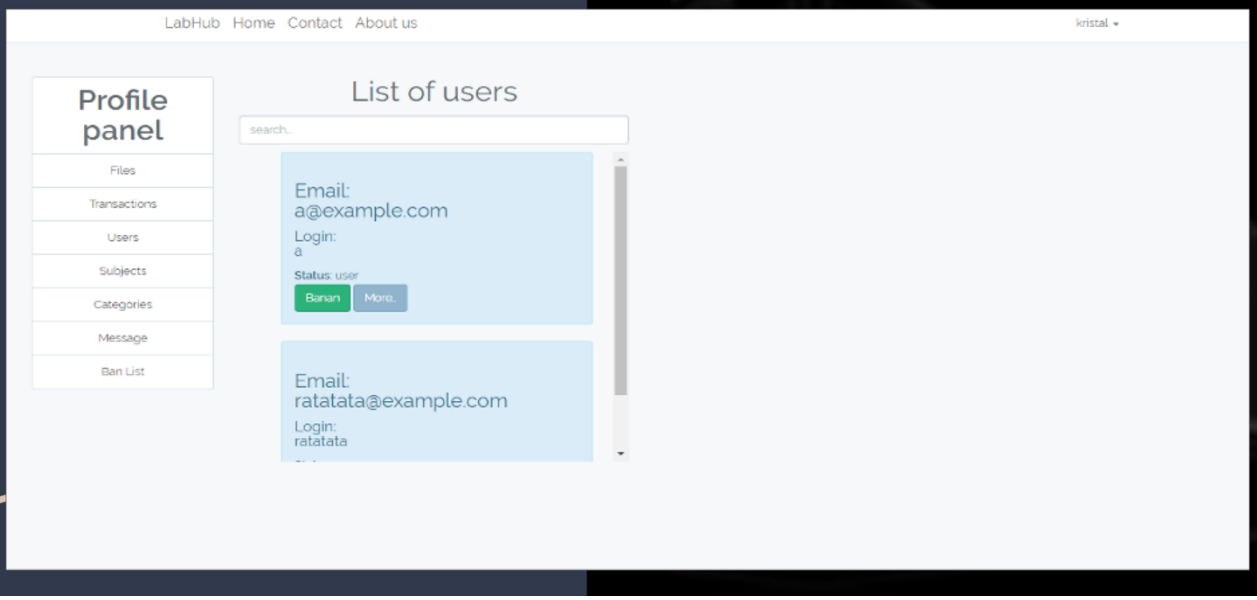
Реалізація проекту. Сторінка зв`язку з адміністратором



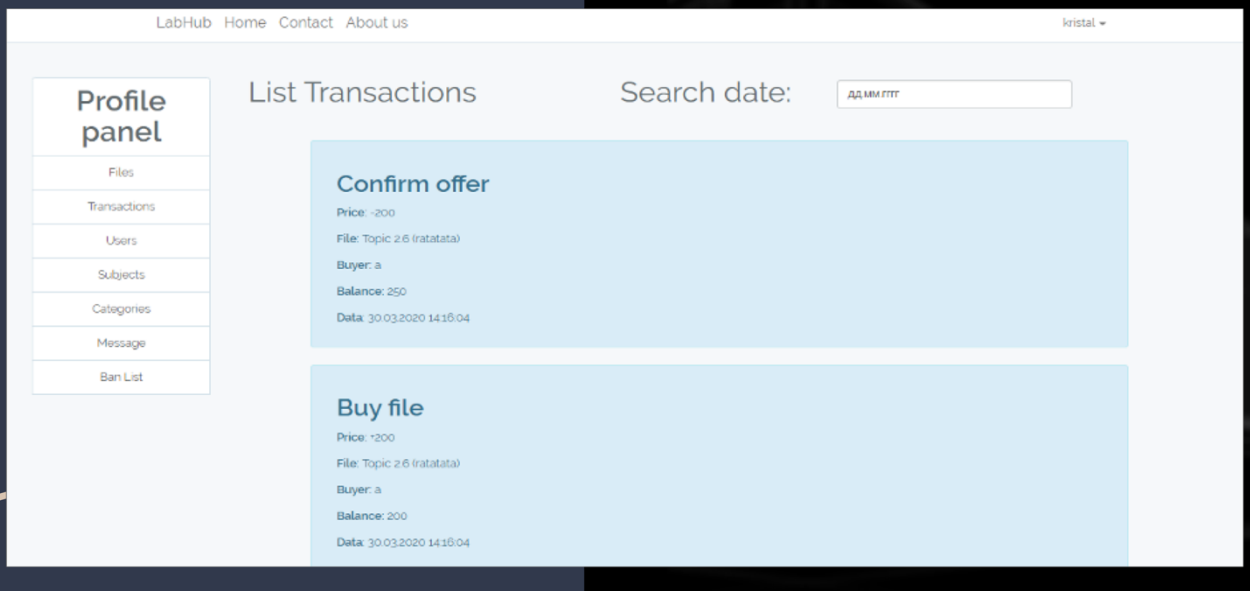
Реалізація проекту. Сторінка профілю адміністратора



Реалізація проекту. Сторінка опублікованої інформації



Реалізація проекту. Сторінка список транзакцій



Висновки

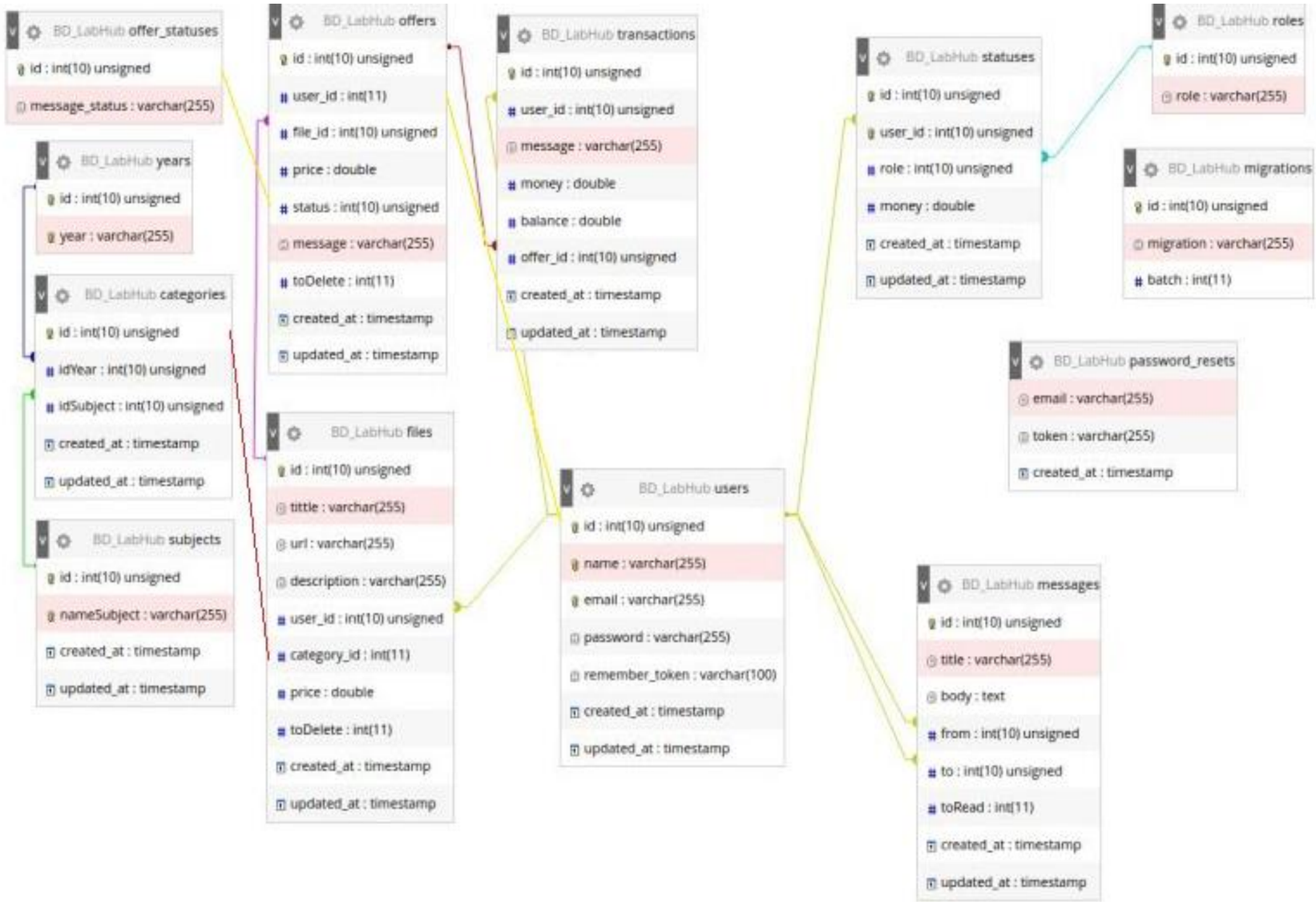
Спроектовано інтерфейс, логіку та архітектуру веб-застосунку, базу даних і метод роботи API.

Створена веб-платформа відповідає поставленому завданню та усім визначеним вимогам.

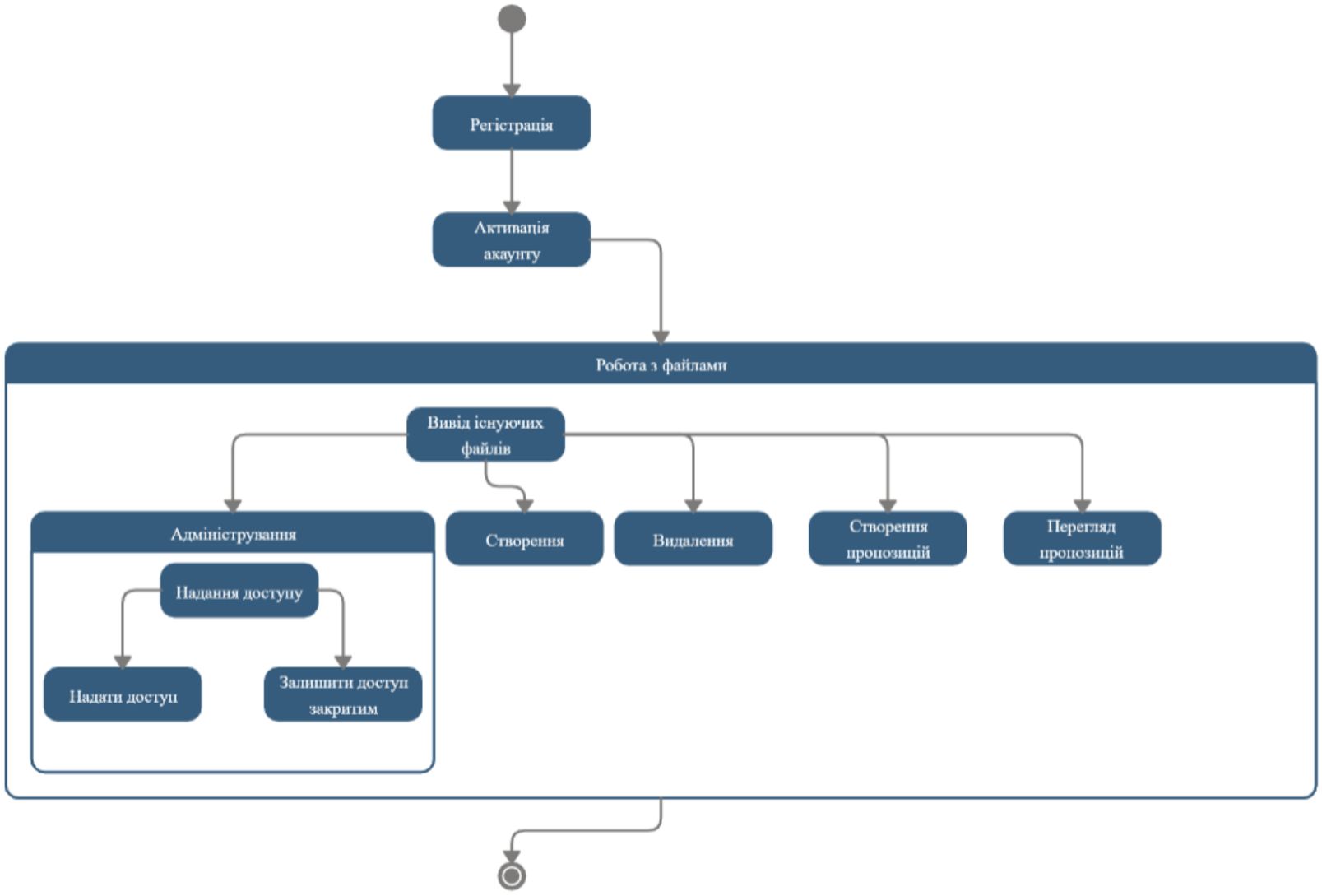
Програмний продукт успішно сконструйований та готовий до використання. Судячи з цього, можна зробити висновок, що даний програмний продукт є цілком працездатним та справним.


Дякую за увагу

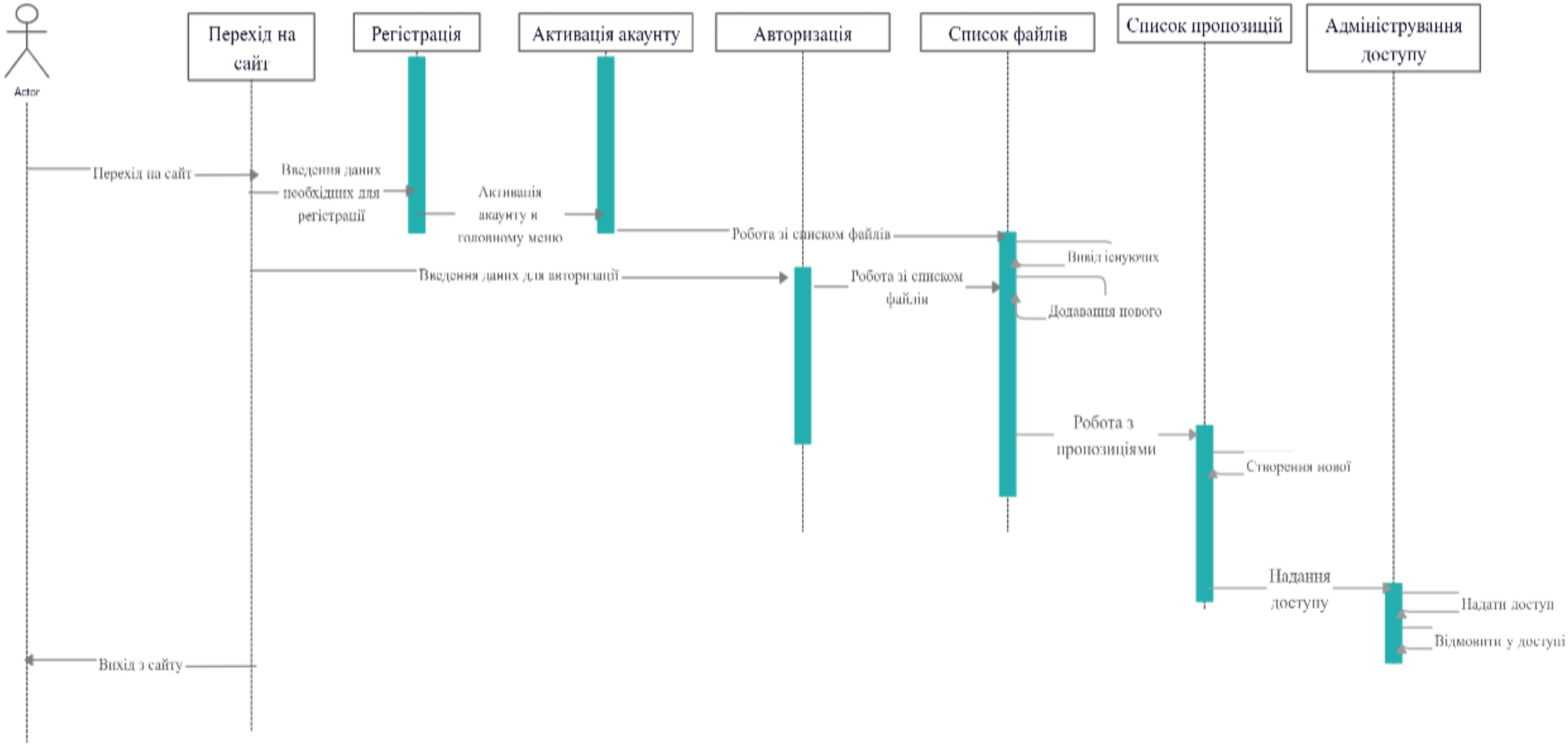
ГРАФІЧНА ЧАСТИНА



					КвРІПЗ.200123.01.04.Е8	
Зм.	Арк.	№ докум.	Підпис	Дата	Веб-платформа "LabHub" для обміну інформацією на базі фреймворку Laravel Структура бази даних	Літера / Маса / Масштаб
Розробив		Варук В.К.		22.05.23		
Керівник		Бедратюк Л. П.		22.05.23		Аркуш 1 / Аркушів 3
Консульт.						
Н. Контр.		Гурман І.В.		22.05.23		ХНУ, ІПЗс-20-1
Зав. каф.		Бедратюк Л.П.		22.05.23		



					КвРІПЗ.200123.01.04.Е8		
					Веб-платформа "LabHub" для обміну інформацією на базі фреймворку Laravel Діаграма станів	Літера	Маса / Масштаб
Зм.	Арк.	№ докум.	Підпис	Дата			
Розробив		Варук В.К.		22.05.23		Аркуш 1	Аркушів 3
Керівник		Бедратюк Л. П.		22.05.23			
Консульт.							
Н. Контр.		Гурман І.В.		22.05.23		ХНУ, ІПЗс-20-1	
Зав. каф.		Бедратюк Л.П.		22.05.23			



					КвРІПЗ.200123.01.04.Е8	
					Веб-платформа "LabHub" для обміну інформацією на базі фреймворку Laravel Діаграма залежностей	Літера / Маса / Масштаб
Зм.	Арк.	№ докум.	Підпис	Дата		Аркуш 1 / Аркушів 3
Розробив		Варук В.К.		22.05.23		
Керівник		Бедратюк Л. П.		22.05.23		
Консульт.						ХНУ, ІПЗс-20-1
Н. Контр.		Гурман І.В.		22.05.23		
Зав. каф.		Бедратюк Л.П.		22.05.23		

Завідувачу кафедри
інженерії програмного забезпечення
проф. Бедратюку Л. П.
студента групи ІПЗс-20-1
Варука Валентина Костянтиновича
Прізвище, ініціали

ЗАЯВА

Прошу закріпити за мною тему кваліфікаційної роботи освітнього ступеня «бакалавр» за спеціальністю 121 «Інженерія програмного забезпечення»: Веб-платформа "LabHub" для обміну інформацією на базі фреймворка Laravel

(керівник роботи – Бедратюк Леонід Петрович)
Прізвище, ім'я, по батькові

05.02.2023
Дата

Підпис студента

Завідувачу кафедри інженерії програмного забезпечення проф. Бедратюку Л. П.

здобувача вищої освіти
Варука Валентин Костянтиновича
Прізвище, ініціали

факультет ІТ, 3 курс, група ІПЗс-20-1

ЗАЯВА

З правилами чинного Положення «Про систему забезпечення академічної доброчесності в Хмельницькому національному університеті» від 01.07.2022, згідно з яким виявлення 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 плагіат оповіщений та надаю свою згоду на обробку й збереження університетом моєї роботи в інституційному репозитарії університету.

Також надаю університету право на передачу моєї роботи для обробки та збереження в базах даних програмно-технічних засобів (Unicheck та/або Anti-Plagiarism) 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 плагіату в текстах робіт.

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

05.06.2023
дата

підпис

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ДЕКЛАРАЦІЯ УЧАСНИКА ОСВІТНЬОГО ПРОЦЕСУ

щодо дотримання академічної доброчесності

Цією декларацією я, Варук Валентин Костянтинович,

студент III курсу спеціальності 121 – Інженерія програмного забезпечення, група ІПЗс-20-1

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.

05 лютого 2023 р.

Підпис

Anti-Plagiarism v-15.257

Максимальне співпадіння з одним документом 4.0%

Словники перевірки: en_US, ru_RU, ua_UA. **Помилок в документах: 12%**

ID: 114798 Назва: БКР Веб-платформа "LabHub" для обміну інформацією на базі фреймворка Laravel Додано в БД: 2023-06-05 Автора: Вирух В.К. Керівники: Бедратюк Г.І. ст. викладач Консультанти: Опоненти:	Документ		Сумарний збіг по Базі Даних	
	Символи	Лексеми	Символи	Лексеми
	58469	525	5926 (10%)	80 (15%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми

UNICHECK

Ім'я користувача:
Кафедра ІПЗ

ID перевірки:
1015437828

Дата перевірки:
05.06.2023 16:31:15 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
05.06.2023 16:34:33 EEST

ID користувача:
100005589

Назва документа: **KvR-Varuk(На плагіат)**

Кількість сторінок: **60** Кількість слів: **8925** Кількість символів: **71779** Розмір файлу: **4.41 MB** ID файлу: **1015098705**

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

14%
Схожість

Найбільша схожість: **5.41%** з джерелом з Бібліотеки (ID файлу: **1014932231**)

8.02% Джерела з Інтернету	433	Сторінка 62
11.2% Джерела з Бібліотеки	133	Сторінка 64

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи	1
Підозріле форматування	16 сторінок

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

Підтверджуємо ознайомлення з результатами звіту/звітів подібності щодо роботи, продукованими програмно-технічним засобом (ами) перевірки текстів на плагіат:

Назва: «Веб-платформа "LabHub" для обміну інформацією на базі фреймворка Laravel»

Автор: Варук Валентин Костянтинович
Спеціальність: 121 – Інженерія програмного забезпечення
Освітня програма: Освітньо-професійна програма «Інженерія програмного забезпечення»
Науковий керівник: Бедратюк Леонід Петрович, д-р фіз.мат. наук, професор
Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом. Робота приймається до захисту.	**відповідає**
2	Виявлені запозичення не є 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. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої й електронної версії роботи	
3	Виявлені запозичення не є плагіатом, але 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 досягнені. Робота може бути допущена до захисту (наступного року) після того, як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	
5	Інше:	

Підтвердження:

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

1) у тексті кваліфікаційної роботи системами перевірки на плагіат виявлено схожість з деякими документами в частині загальновживаних обов'язкових словосполучень у стандартних бланках (титулка, відомість документів), у структурі змісту, назвах розділів/підрозділів тощо, у назвах публікацій у переліку джерел посилання;

2) в якості запозичень системою було зафіксовано деякі послідовності вихідного коду і посилання на бібліотеки, які є стандартними мовними конструкціями програмування та не можуть розглядатися як об'єкт авторських прав і, відповідно, їх порушення;

3) усі запозичення є фрагментарними або мають належним чином оформлені посилання;

4) виявлені модифікації тексту не впливають на відсоток схожості.

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів ідентичності/ схожості, складає 14% і адресується до 433 джерел, що, з урахуванням наведених обґрунтувань, відповідає характеру теми і свідчить на користь кваліфікаційної роботи.

Дата 7.06.23

Завідувач кафедри _____ Леонід БЕДРАТЮК

Гарант освітньої програми _____ Леонід БЕДРАТЮК

Керівник кваліфікаційної роботи _____ Леонід БЕДРАТЮК

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ
освітнього ступеня «Бакалавр»

Дипломник Варук Валентин Костянтинович

Тема Веб-платформа "LabHub" для обміну інформацією на базі фреймворка Laravel

Спеціальність 121 – Інженерія програмного забезпечення

Обсяг кваліфікаційної роботи:

Кількість листів креслень 3; кількість сторінок записки 55

1.Короткий зміст пояснювальної записки та прийнятих рішень У кваліфікаційній роботі досліджено і проаналізовано предметну область, визначено усі функціональні та нефункціональні вимоги. Був проведений аналіз існуючих програм на ринку, розглянуто їх переваги і недоліки, та доведено актуальність розробки нового програмного забезпечення. Розглянуто інструменти для реалізації спроектованих рішень, в результаті чого було вибране програмне забезпечення для розробки кваліфікаційної роботи. Також було проведено тестування програми, за результатами якого доведено, що розроблене програмне забезпечення працює коректно та готове до експуатації.

2. Висновок про відповідність роботи поставленому завданню Кваліфікаційна робота виконана відповідно до поставленого завдання та з дотриманням всіх вимог.

3. Характеристика виконання кожного розділу роботи, ступінь використання останніх досягнень науки і техніки та передових методів роботи У вступі доведено актуальність теми, визначено мету та завдання дипломного проектування. У першому розділі проведено аналіз предметної області, розглянуто існуючі рішення та визначені функціональні і нефункціональні вимоги до розроблюваного програмного забезпечення. У другому розділі проведено аналіз сучасних архітектур, розглянуто їх переваги і недоліки та визначено, що система буде відповідати монолітній архітектурі та моделі клієнт-сервер. У третьому розділі підготовлено всі залежності для написання коду та виконано практичну розробку програмних модулів і описано їх особливості, в результаті чого створено програмний продукт. Також у цьому розділі виконано модульне тестування системи та проведено його у відповідності до функціональних вимог, в результаті було підтверджено коректну роботу програми.

4. Позитивні сторони роботи Тематика кваліфікаційної роботи є актуальною, оскільки на сьогодні потрібно витрачати багато часу на пошук потрібної інформації, та ця платформа є доступна для будь-якого пристрою

5. Негативні сторони роботи У роботі пошук публікацій був реалізований лише за категоріями та роками – було б доцільно додати розширений пошук. Також потрібно було додати перевірку інформації, яку користувачі завантажують на платформу за допомогою ботів.

6. Оцінка графічного оформлення та пояснювальної записки Графічне оформлення виконано відповідно до теми кваліфікаційної роботи та подано у вигляді діаграм і рисунків. Пояснювальна записка оформлена згідно вимог чинних стандартів.

7. Відгук про кваліфікаційну роботу в цілому Кваліфікаційна робота заслуговує позитивної оцінки. Матеріал пояснювальної записки структурований, послідовний, чіткий та простий, що дозволяє чітко зрозуміти викладений матеріал у рамках тематики проектування. Графічний матеріал дає можливість наочно побачити деталі проектування системи.

8. Інші зауваження

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

РЕЦЕНЗЕНТ Гнатчук Єлізавета Геннадіївна, доцент кафедри КІС

"6" червня 2023 р. _____
(підпис)