

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавр

(ступінь вищої освіти)

на тему ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ГРОМАДСЬКОГО ФІНАНСУВАННЯ
ТВОРЧИХ ПРОЄКТІВ

(назва теми)

SOFTWARE FOR PUBLIC FUNDING OF CREATIVE PROJECTS

Виконав: студент(ка) 4 курсу, групи КНТ-139

Спеціальності 121 Інженерія програмного

(код і найменування спеціальності)

забезпечення

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

ДИКАРЬОВ А.А.

(ПРИЗВИЩЕ та ініціали)

Керівник ЛЬОВКІН В.М.

(ПРИЗВИЩЕ та ініціали)

Рецензент ТИМЕНКО А.В.

(ПРИЗВИЩЕ та ініціали)

2023

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

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

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

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

ДИКАРЬОВА Артема Андрійовича
(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Програмне забезпечення громадського фінансування творчих проєктів. Software for Public Funding of Creative Projects

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

затверджені наказом закладу вищої освіти від “4” квітня 2023 року № 87

2. Строк подання студентом проєкту (роботи) 1 червня 2023 року

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

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Проєктування програми. 3. Розробка програми. 4. Експлуатація та тестування програми.

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

6. Консультанти розділів проєкту (роботи)

Розділ	ПРІЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1–4 Основна частина	ЛЬОВКІН В.М., доцент		
Нормоконтроль	БСЛОВА А.В., асистент		

7. Дата видачі завдання « 4 » квітня 2023 року.

КАЛЕНДАРНИЙ ПЛАН

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

Студент(ка)

_____ Артем ДИКАРЬОВ
(підпис) (Ім'я ПРІЗВИЩЕ)

Керівник проєкту (роботи)

_____ Валерій ЛЬОВКІН
(підпис) (Ім'я ПРІЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра:
103 с., 51 рис., 4 табл., 3 дод., 17 джерел.

ГРОМАДСЬКЕ ФІНАНСУВАННЯ, КРАУДФАНДИНГ, ТВОРЧІ
ПРОЄКТИ, БЛАГОДІЙНІСТЬ, ПЛАТФОРМИ ГРОМАДСЬКОГО
ФІНАНСУВАННЯ.

Об'єкт дослідження – процес громадського фінансування творчих проєктів.

Предмет дослідження – програмне забезпечення громадського фінансування творчих проєктів.

Мета роботи – розробка програмного забезпечення для підтримки громадського фінансування творчих проєктів.

Матеріали, методи та технічні засоби: мова програмування PHP, вебфреймворк Laravel, система керування базами даних MySQL.

Результати. Основним результатом роботи є створене програмне забезпечення, що надає інструмент для збору коштів на підтримку різних творчих проєктів та ініціатив з врахуванням особливостей таких проєктів і спрощенням відповідних процесів підтримки, у тому числі сама програма і описані рішення щодо її створення.

Висновки. Розроблене програмне забезпечення надає можливість збору коштів на підтримку різних культурних та мистецьких ініціатив. Її використання в подальшому може зменшити витрати на маркетинг та просування проєктів та залучити більше вкладників.

Галузь використання – це сфера культури та мистецтва, а також соціального підприємництва та благодійних організацій.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 103 pages, 51 figures, 4 tables, 3 appendixes, 17 sources.

PUBLIC FINANCING, CROWDFUNDING, CREATIVE FUNDING PROJECTS, CHARITY, CROWD FUNDING PLATFORMS.

The object of research is the process of public financing of creative projects.

The subject of research is software for public financing of creative projects.

The purpose of the work is to develop software to support public financing of creative projects.

Materials, methods and technical means: PHP programming language, Laravel web framework, MySQL database management system.

The results. The main result of the work is the created software, which provides a tool for collecting funds to support various creative projects and initiatives, taking into account the specifics of such projects and simplifying the relevant support processes, including the program itself and the described solutions for its creation.

Conclusions. The developed software provides an opportunity to collect funds to support various cultural and artistic initiatives. Its use in the future can reduce the costs of marketing and promotion of projects and attract more investors.

The field of use is the field of culture and art, as well as social entrepreneurship and charitable organizations.

ЗМІСТ

	С.
Перелік скорочень та умовних позначок.....	8
Вступ.....	9
1 Аналіз предметної області.....	10
1.1 Загальний стан проблеми громадського фінансування творчих проєктів.....	10
1.2 Огляд аналогів програмного забезпечення громадського фінансування творчих проєктів.....	10
1.3 Висновки за розділом 1.....	16
2 Проєктування програми.....	17
2.1 Вибір засобів проєктування і розробки.....	17
2.2 Визначення функціональних вимог до програмного забезпечення.....	19
2.3 Проєктування структури бази даних.....	23
2.4 Структурна схема програмного проєкту.....	28
2.5 Висновки за розділом 2.....	32
3 Розробка програми.....	33
3.1 Алгоритм функціонування програмного забезпечення.....	33
3.2 Опис реалізованих класів.....	35
3.3 Висновки за розділом 3.....	46
4 Експлуатація та тестування програми.....	47
4.1 Призначення програми.....	47
4.2 Умови виконання програми.....	47
4.3 Виконання програми.....	47
4.4 Тестування програми.....	58
4.5 Висновки за розділом 4.....	60
Висновок.....	60
Перелік джерел посилання.....	62
Додаток А Технічне завдання.....	64

	7
Додаток Б Текст програми.....	68
Додаток В Слайди презентації.....	95

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

БД – база даних;

СУБД – система управління базами даних;

API – Application Programming Interface;

MVC – Model View Controller.

ВСТУП

В наш час творчі проєкти вважаються джерелом для підтримки життєдіяльності людства. Однак, деякі творці не отримують повного фінансування для подальшого розвитку своїх творчих ідей.

На сьогодні існують певні платформи для залучення фінансування від громадськості для реалізації власних ідей. Наприклад, за допомогою платформи Kickstarter у 2019 році було зібрано суму 11 млн доларів для створення власного анімаційного вебсеріалу Critical Role [1].

Тому метою даної дипломної роботи є розробка програмного забезпечення для підтримки громадського фінансування творчих проєктів.

В даній роботі необхідно провести дослідження та аналіз всіх існуючих аналогів, проєктування програмного забезпечення, розробку та тестування розробленого програмного забезпечення. Результати роботи мають бути представлені у вигляді пояснювальної записки та створеного програмного продукту.

Таким чином, програмне забезпечення, яке має бути створено в роботі, повинно допомогти авторам (творцям) творчих проєктів отримувати додаткове фінансування від громадськості на подальший розвиток власних ідей.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Загальний стан проблеми громадського фінансування творчих проєктів

На даний момент існують краудфандингові платформи, які вважаються одним із найпопулярніших видів громадського фінансування творчих проєктів.

Краудфандинг – це спосіб залучення певної грошової суми для розвитку власних проєктів від великої кількості учасників, які зацікавлені самою ідеєю проєкту [2].

Для залучення коштів на творчі проєкти краудфандингові платформи працюють за певним алгоритмом: представник проєкту створює його відеопрезентацію, яка привертає увагу людей і викликає їхнє бажання зробити пожертвування [3]. Цей метод залучення фінансування допомагає неприбутковим громадським організаціям ефективно привернути увагу людей до важливих питань і отримати їхню підтримку – як моральну, так і матеріальну [3].

1.2 Огляд аналогів програмного забезпечення громадського фінансування творчих проєктів

На даний момент існує багато краудфандингових платформ для збору коштів на реалізацію творчих ідей. Було зроблено огляд відомих краудфандингових платформ для підтримки фінансування творчих проєктів,

Kickstarter (рис. 1.1) є найбільш популярною краудфандинговою платформою, що дозволяє отримати певне грошове фінансування для реалізації власних творчих проєктів. За допомогою платформи Kickstarter можна зібрати поставлену грошову суму пожертвування від зацікавленої аудиторії для реалізації власних креативних ідей [4]. Також Kickstarter зобов'язує кожний проєкт мати чітку мету: наприклад, створення альбому, книги або мистецького твору. Після того як проєкт отримав певне фінансування від користувачів, він

повинен обов'язково бути реалізований та приносити певний результат для своїх прихильників [4].

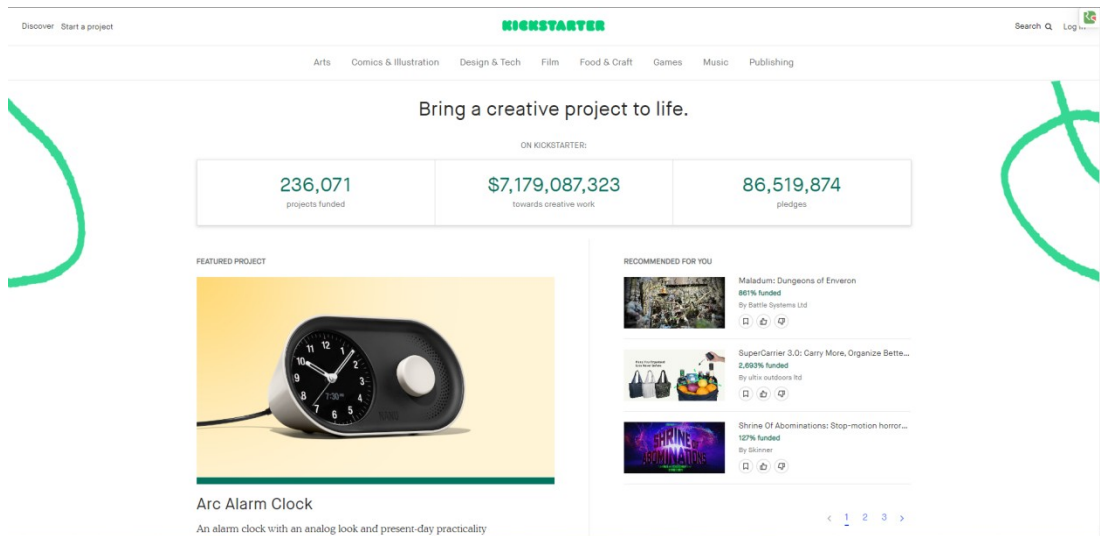


Рисунок 1.1 – Інтерфейс Kickstarter

До переваг Kickstarter відносять:

- опублікований проєкт на Kickstarter може безкоштовно привернути багато уваги серед людей;
- опублікований проєкт може отримати зворотний зв'язок від своїх прихильників;
- Kickstarter надає можливість творцям отримати грошове фінансування для своїх задуманих проєктів, не звертаючись до банків та інвесторів [5].

До недоліків Kickstarter відносять:

- Kickstarter стягує комісію у розмірі 5 % від суми пожертвування, а також додаткові комісії для обробки платежів;
- немає гарантій на повернення грошей, якщо збір на проєкт не буде завершено або не буде виконано відповідно до очікувань;
- відсутнє обмеження на зібрану суму, через це проєкти можуть отримати більше грошей, аніж їм насправді потрібно;

– ризик шахрайства через те, що шахраї можуть створювати проекти, які не мають ніякої реальної мети, але при цьому отримують гроші від звичайних спонсорів [5].

Платформа Patreon (рис. 1.2) забезпечує фінансову підтримку творчих та незалежних проектів через підписку на спеціальний контент або інші можливості, які обирає автор. Patreon дуже популярна платформа серед творців на YouTube, письменників, музикантів, авторів вебкоміксів, подкастерів та блогерів, оскільки дозволяє отримувати фінансування напряму від підписників на постійній основі або за кожну окрему роботу [6].

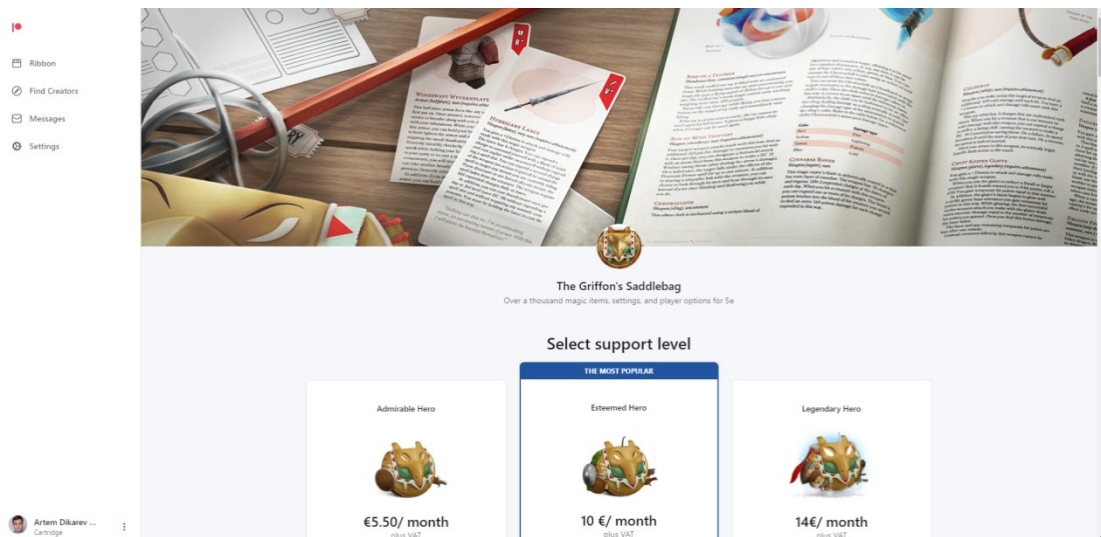


Рисунок 1.2 – Інтерфейс Patreon

До переваг Patreon відносять:

- зручна передплата на регулярні щомісячні платежі для творців творчих проектів;
- єдиний інтерфейс для всіх учасників;
- спілкування з творцями творчих проектів;
- творці проектів можуть надавати свій ексклюзивний контент (відео, записи, фото) для своїх підписників;

– гарантія безпеки та захист даних шанувальників та творчих людей, а також надає підтримку для вирішення будь-яких проблем чи питань, які можуть виникнути [7].

Донедоліків Patreon відносять:

- повільна обробка платежів, яка може тривати кілька днів;
- Patreon не просуває творців проєктів, тому дуже важко знайти аудиторію для творчого проєкту;
- автори повинні постійно викладати контент своїх творчих проєктів для підвищення привабливості проєкту [8].

Краудфандингова платформа IndieGoGo (рис. 1.3) дозволяє збирати кошти для різних цілей, включаючи бізнес, благодійні та некомерційні організації, політичні кампанії та багато іншого. Завдяки своїм потужним інструментам підтримки проєктів та глобальному поширенню, ця платформа стала однією з провідних краудфандингових платформ у світі. З самого початку в 2008 році IndieGoGo залучила десятки мільйонів користувачів, які запустили проєкти в різних сферах від творчості до технологій та дизайну [9].

Користувачі платформи можуть жертвувати кошти на кампанію в обмін на матеріальну винагороду або просто за підтримку проєкту [9].

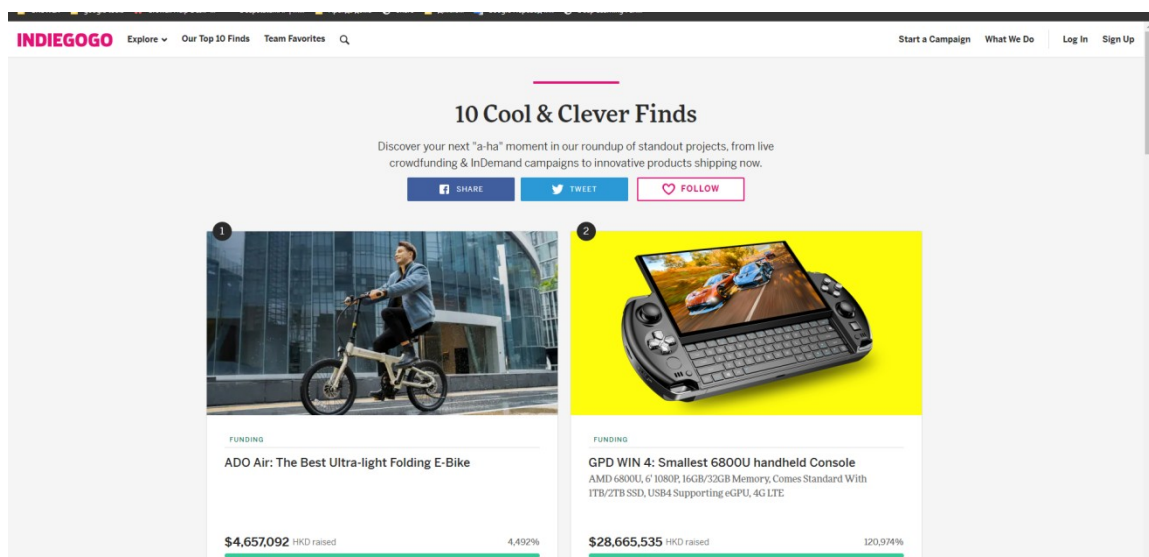


Рисунок 1.3 – Інтерфейс Indiegogo

До переваг Indiegogo відносять:

- Indiegogo дозволяє продовжувати збирати гроші після успішного збору коштів;
- Indiegogo надає різні типи зборів коштів, які можна запускати: бувають у формі «фіксованого» чи «гнучкого» фінансування;
- платформа доступна для будь-якого типу компанії. Indiegogo пропонує різні категорії товарів від некомерційних організацій до технічних гаджетів [10]-[11].

До недоліків Indiegogo відносять:

- дуже висока комісія, яка може досягати 9 % порівняно з 5 % на Kickstarter;
- зобов'язання відправляти винагороду спонсорам: якщо проєкт не досягає своєї поставленої мети з гнучким варіантом оплати, то автор зобов'язаний відправити винагороду користувачам, які вже зробили пожертвування для проєкту [11].

За результатами огляду трьох основних програмних застосунків для громадського фінансування творчих проєктів, було виокремлено ключові критерії, за якими будуть порівнюватися аналоги та застосунок (табл. 1.1).

Таблиця 1.1 – Порівняння аналогів і програмного застосунку

Критерій оцінки	Аналоги			
	Kickstarter	Patreon	Indiegogo	Застосунок
1	2	3	4	5
Спілкування з аудиторією	+	+	–	+
Інтеграція з платіжними системами	+	+	+	+

Кінець таблиці 1.1

1	2	3	4	5
Зручна аналітика по пожертвуванням	—	—	—	+
Швидка модерація проєктів	—/+	—	+	+
Простий інтерфейс для звичайного користувача	+	—	+	+
Ексклюзивний контент для спонсорів проєктів	—	+	+	+

Виходячи з аналізу, програмне забезпечення має забезпечувати такі властивості і можливості:

- зручний інтерфейс програмного забезпечення;
- керування творчими проєктами;
- відправлення пожертвувань для творчих проєктів через міжнародну платіжну систему PayPal;
- додавання фотографій кожного учасника творчого проєкту;
- надання безкоштовного доступу до ексклюзивних матеріалів;
- перегляд діаграми фінансування певного творчого проєкту для творця;
- перегляд діаграми за відправленими пожертвуваннями для спонсора;
- коментування під кожним проєктом;
- подання скарги на будь-який творчий проєкт, який порушує правила сервісу;

- визначення періоду збору коштів, протягом якого користувачі можуть перераховувати кошти на фінансування на творчі проєкти;
- додавання відеороліка для кожного творчого проєкту.

При визначенні концепції розроблюваного програмного забезпечення не ставилася за необхідність заміна аналогічних програмних застосунків, а використовувалася ідея втілення зручних методів для громадського фінансування саме творчих проєктів.

1.3 Висновки за розділом 1

В даному розділі було виконано огляд таких застосунків: Kickstarter, Patreon, Indiegogo. В якості особливостей розробки визначено, що застосунок має бути орієнтованим на авторів творчих проєктів, а також мати можливість залучення інвесторів, які більше цікавляться творчістю, мистецтвом.

Завданнями дипломної роботи є:

- аналіз предметної області;
- визначення вимог до програмного забезпечення громадського фінансування творчих проєктів;
- проєктування програмного забезпечення громадського фінансування творчих проєктів;
- розробка та тестування програмного забезпечення громадського фінансування творчих проєктів.

2 ПРОЄКТУВАННЯ ПРОГРАМИ

2.1 Вибір засобів проєктування і розробки

Для створення програмного забезпечення громадського фінансування творчих проєктів було обрано мову програмування PHP.

PHP є мовою програмування, яка часто використовується для створення вебзастосунків. Оскільки вона є вільною та відкритою мовою програмування, її можна легко змінювати та поширювати. PHP є популярним вибором серед веброзробників, оскільки більшість хостинг–провайдерів підтримують цю мову [12]. Додатковим свідченням її практичної придатності для великих активно використовуваних проєктів є те, що вона є основною мовою програмування для багатьох відомих вебсайтів, таких як Facebook і Wikipedia, які використовують її для розробки та підтримки функціонування своїх вебсайтів [13]. Коли вебсервер обробляє вихідні дані PHP, він інтерпретує їх у HTML-код та передає його до клієнтської сторони [14].

На сьогодні мова програмування PHP є найбільш зручним інструментом для веброзробки, особливо при роботі з великою спільнотою розробників або різними програмними бібліотеками [14]. Тому було вирішено розробити даний програмний застосунок мовою програмування PHP з використанням фреймворку Laravel.

Також було зазначені ключові переваги мови програмування PHP [15]:

- простий синтаксис;
- має велику спільноту розробників;
- кросплатформність самої мови програмування;
- має зрозумілу документацію;
- PHP можна застосовувати на всіх відомих видах серверів;
- написання та редагування коду можливе у будь-якому редакторі тексту;
- дає можливість для з'єднання з найпопулярнішими СУБД (MySQL, SQLITE, PostgreSQL та інші).

Також було зроблено порівняння мови програмування PHP з іншими мовами програмування, що показано в табл. 2.1.

Таблиця 2.1 – Порівняння PHP з іншими мовами програмування

	PHP	Python	C#
Кросплатформеність	+	+	+
Швидкість розробки програми	+	+	–
Швидкість роботи кінцевого продукту	+	–	+
Легкість вивчення	+	+	–
Зрозуміла документація	+	+	+/-

Для розробки застосунку був обраний PHP-фреймворк Laravel. Головною перевагою серед інших фреймворків є те, що архітектура Laravel повністю заснована на патерні MVC. Laravel підходить для розробки високонавантажених програмних застосунків [16].

Порівняння Laravel з іншими PHP-фреймворков наведено в табл. 2.2.

Таблиця 2.2 – Порівняння Laravel з іншими PHP-фреймворков

	Laravel	Symfony	Yii2
Швидкість розробки	++	–	+
Безпека	+	+	+
Підтримка	+	+	–
Безпека	++	+	+
Продуктивність	+	+	+
Масштабованість	+	+	+

Отже, було вирішено використовувати мову програмування PHP та вебфреймворк Laravel для реалізації програмного забезпечення громадського фінансування творчих проєктів.

2.2 Визначення функціональних вимог до програмного забезпечення

Програмний застосунок повинен бути розроблений для чотирьох типів користувачів:

- творець;
- спонсор;
- неавторизований користувач;
- менеджер.

Тому можна класифікувати наступні функціональні вимоги відповідно до вищевказаного списку типів користувачів.

Для візуального сприйняття функціональних вимог для неавторизованого користувача було створено діаграму прецедентів на рис. 2.1.

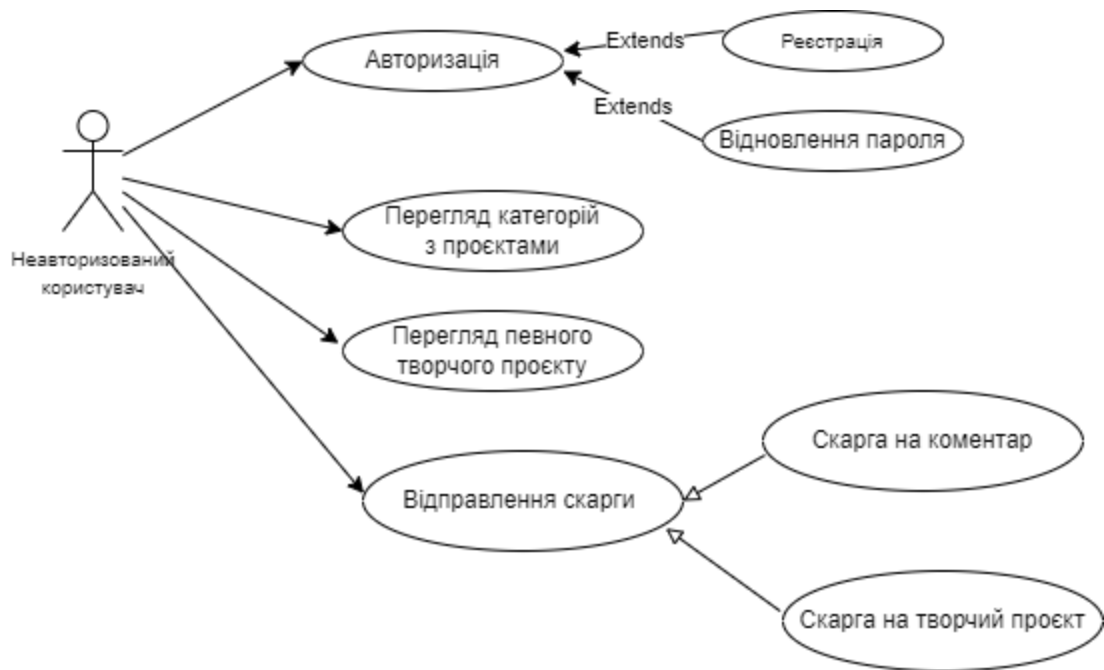


Рисунок 2.1 – Діаграма прецедентів неавторизованого користувача

Програма повинна задовольняти наступні функціональні вимоги для творця творчих проєктів:

- групування проєктів за категоріями;
- створення проєктів;
- редагування та видалення проєктів;
- перегляд платежів за кампанією зі збору коштів на фінансування проєкту;
- створення ексклюзивних матеріалів для кожного творчого проєкту;
- визначення періоду збору коштів, протягом якого користувачі можуть перераховувати кошти на фінансування;
- перегляд статистики отримання пожертвувань по кожному творчому проєкту;
- перегляд графіку отримання пожертвувань по кожному творчому проєкту;
- завантаження фотографій кожного учасника творчого проєкту;
- завантаження відеопрезентації проєкту;
- перегляд коментарів по кожному проєкту;

- перегляд платежів від спонсорів, які підтримали проєкт;
- визначення періоду збору коштів, протягом якого користувачі можуть перераховувати кошти на фінансування.

Для візуального сприйняття функціональних вимог для творця було створено діаграму прецедентів на рис. 2.2.

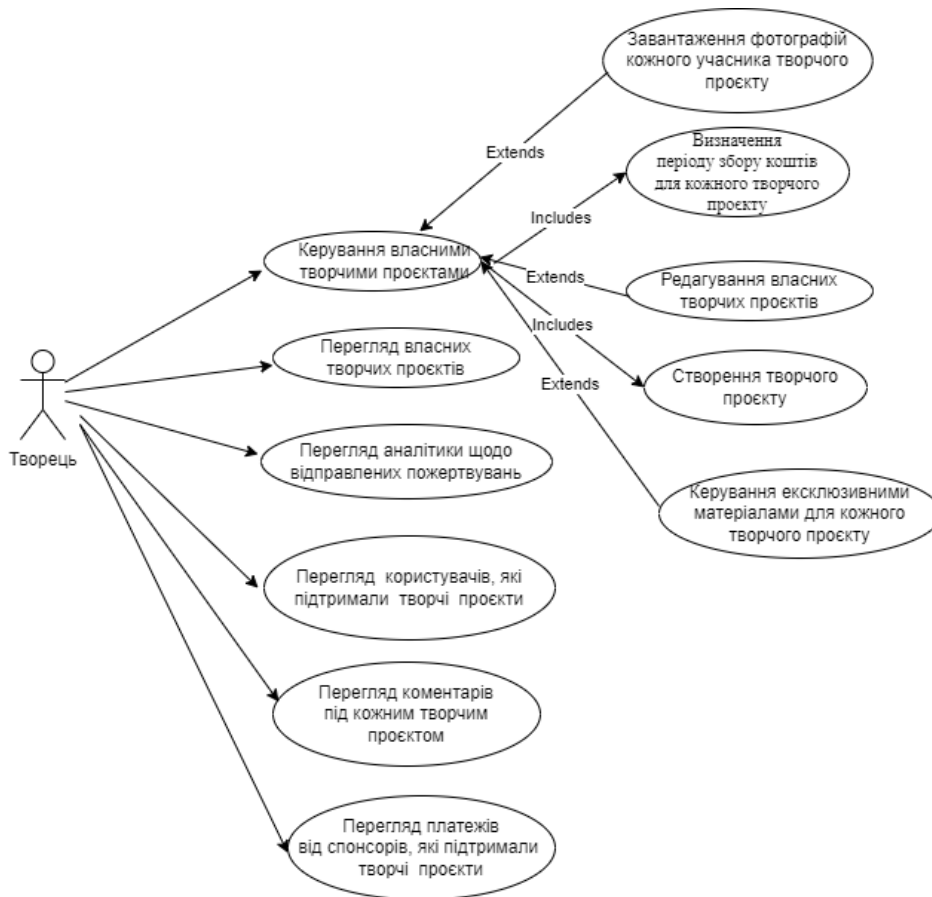


Рисунок 2.2 – Діаграма прецедентів творців

Програма повинна задовольняти наступні функціональні вимоги для спонсорів:

- додавання коментаря під кожним проєктом;
- перегляд кожного проєкту;
- перегляд категорій з проєктами;
- подання скарг;
- перегляд фотографій кожного учасника творчого проєкту;
- перегляд відеопрезентації кожного творчого проєкту;

- перегляд аналітики щодо відправлених пожертвувань;
- відправлення грошового переказу для певного творчого проєкту за допомогою платіжної системи PayPal;
- перегляд ексклюзивних матеріалів;
- перегляд історії фінансування певного творчого проєкту;
- перегляд графіка отримання пожертвувань за кожний місяць.

Для візуального сприйняття функціональних вимог для спонсорів було створено діаграму прецедентів на рис. 2.3.

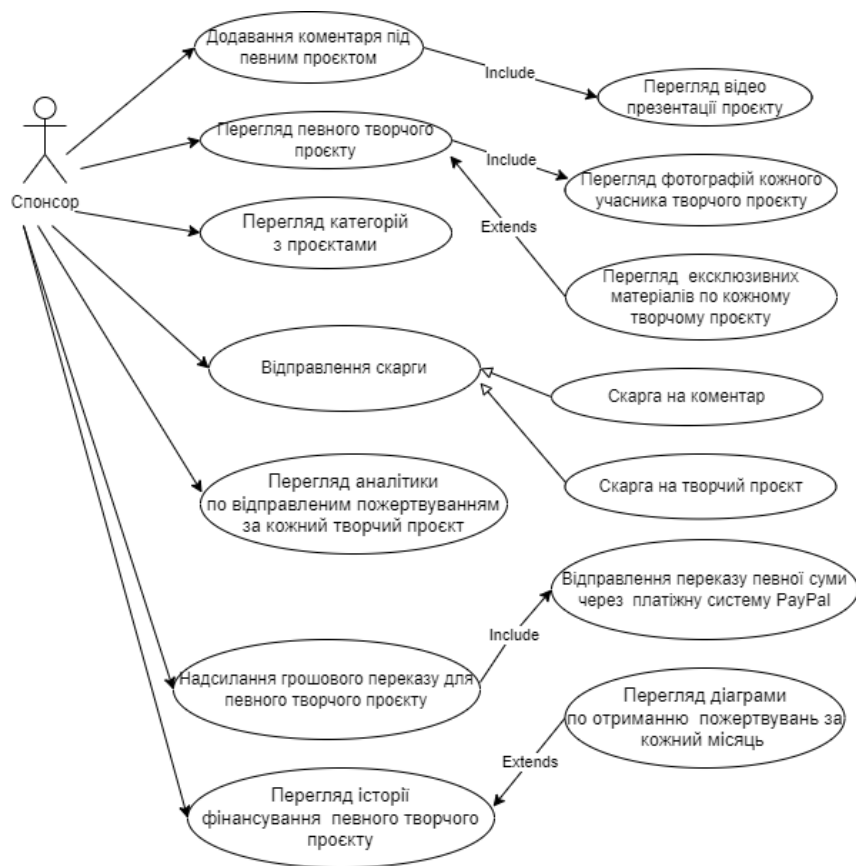


Рисунок 2.3 – Діаграма прецедентів спонсорів

Програма повинна задовольняти наступні функціональні вимоги для менеджера:

- модерація проєктів;
- видалення проєктів;
- редагування проєктів;

- видалення коментарів;
- редагування коментарів;
- опрацювання (менеджером) коментарів;
- керування скаргами;

Для візуального сприйняття функціональних вимог для менеджера було створено діаграму прецедентів на рис. 2.4.



Рисунок 2.4 – Діаграма прецедентів менеджера

2.3 Проектування структури бази даних

Для роботи з базою даних (БД) було використано систему управління базами даних MySQL [17].

Переліки творчих проєктів, які беруть участь у зборі коштів, зберігаються в таблиці БД projects з наступними полями:

- id – номер проєкту, який бере участь у зборі коштів;
- categories_id – унікальний номер категорії проєкту;
- title – назва проєкту;
- img – зображення проєкту;

- video – відеопрезентація проєкту;
- slug – URL-посилання, за яким знаходиться творчий проєкт;
- annotation – короткий опис творчого проєкту;
- started_at – дата початку компанії по фінансуванню проєкту;
- started_end – дата кінця компанії по фінансуванню проєкту;
- created_at – дата створення творчого проєкту;
- updated_at – дата оновлення творчого проєкту;
- user_id – унікальний номер користувача, який створив проєкт;
- amount – запланована сума для збору;
- website – посилання на вебсайт творчого проєкту;
- instagram – посилання на instagram-сторінку творчого проєкту;
- twitter – посилання на twitter-сторінку творчого проєкту;
- facebook – посилання на facebook-сторінку творчого проєкту;
- telegram – посилання на telegram-сторінку творчого проєкту.

Категорії для проєктів зберігаються в таблиці БД `category` з наступними полями:

- id – номер категорії;
- name_category – назва категорії;
- created_at – дата створення категорії;
- updated_at – дата оновлення категорії.

Переліки користувачів зберігаються в таблиці БД `users` з наступними полями:

- id – номер користувача;
- name – ім'я користувача;
- email – електронна адреса користувача;
- email_verified_at – верифікація електронної адреси користувача;
- password – пароль користувача у форматі `crypt`;
- created_at – дата створення запису користувача;
- updated_at – дата оновлення даних користувача ;
- role – роль користувача.

Коментарі до кожного проєкту зберігаються в таблиці БД `commentproject` з такими полями:

- `id` – номер коментаря;
- `user_id` – унікальний номер користувача, який створив коментар;
- `project_id` – унікальний номер проєкту, до якого відноситься коментар;
- `body` – текст коментаря;
- `created_at` – дата створення коментаря;
- `updated_at` – дата оновлення коментаря.

Ексклюзивні матеріали до кожного проєкту зберігаються в таблиці БД `exclusive_material` з наступними полями:

- `id` – номер ексклюзивного матеріалу;
- `project_id` – унікальний номер проєкту, якому належить ексклюзивний матеріал;
- `body` – текстовий контент ексклюзивного матеріалу;
- `slug` – URL-посилання, за яким знаходиться ексклюзивний матеріал;
- `img` – зображення ексклюзивного матеріалу;
- `user_id` – унікальний номер творця, який створив ексклюзивний матеріал.

Грошові перекази до кожного проєкту зберігаються в таблиці БД `donation` з наступними полями:

- `id` – номер грошового переказу;
- `user_id` – унікальний номер користувача, який створив грошовий переказ;
- `project_id` – унікальний номер проєкту, за яким було надіслано грошовий переказ;
- `amount` – сума грошового переказу;
- `created_at` – дата створення грошового переказу;
- `updated_at` – дата оновлення грошового переказу;
- `paupal_payment_id` – унікальний номер, код грошового переказу, який було створено через платіжну систему Paupal.

Фото учасників творчого проєкту зберігаються в таблиці БД `photo_member` з такими полями:

- `id` – номер фото учасника творчого проєкту;
- `project_id` – унікальний номер проєкту;
- `name` – назва учасника творчого проєкту;
- `img` – зображення творчого проєкту;
- `created_at` – дата створення;
- `updated_at` – дата оновлення;
- `user_id` – унікальний номер користувача, який додав фото учасника

творчого проєкту.

Перелік скарг на творчі проєкти зберігається в таблиці БД `complaint` з наступними полями:

- `id` – номер скарги;
- `name` – назва скарги;
- `type` – причина скарги на проєкт;
- `body` – текст скарги на проєкт;
- `done` – схвалені скарги на проєкти адміністрацією;
- `project_id` – унікальний номер проєкту;
- `created_at` – дата створення скарги;
- `updated_at` – дата оновлення скарги.

Перелік скарг на певні коментарі зберігається в таблиці БД `complaintcomment` з наступними полями:

- `id` – номер скарги;
- `comment` – унікальний номер коментаря;
- `created_at` – дата створення скарги;
- `done` – схвалені скарги на коментар адміністрацією;
- `updated_at` – дата оновлення скарги.

Окремо необхідно розглянути зв'язки між сутностями БД.

Зв'язок між сутністю `category` та `project` один-до-багатьох: одна категорія може мати декілька творчих проєктів.

Зв'язок між сутністю user та project один-до-багатьох: один користувач може створювати декілька творчих проєктів.

Зв'язок між сутністю project та exclusive_material один-до-багатьох: один творчий проєкт може мати декілька ексклюзивних матеріалів.

Зв'язок між сутністю user та exclusive_material один-до-багатьох: один користувач може створювати декілька ексклюзивних матеріалів.

Зв'язок між сутністю project та photo_members один-до-багатьох: один творчий проєкт може мати декілька фото учасників проєкту.

Зв'язок між сутністю user та photo_members один-до-багатьох: один користувач може створювати декілька фото учасників проєкту.

Зв'язок між сутністю project та donation один-до-багатьох: один творчий проєкт може мати декілька грошових пожертвувань.

Зв'язок між сутністю user та donation один-до-багатьох: один користувач може створювати декілька грошових пожертвувань.

Зв'язок між сутністю project та complaint один-до-багатьох: один творчий проєкт може мати декілька скарг.

Зв'язок між сутністю user та complaint один-до-багатьох: один користувач може створювати декілька скарг на творчі проєкти. Зв'язок між сутністю project та complaintcomment один-до-багатьох: один коментар може мати декілька скарг.

Зв'язок між сутністю user та complaintcomment один-до-багатьох: один користувач може створювати декілька скарг на коментарі.

Створена схема БД показана на рис. 2.5.

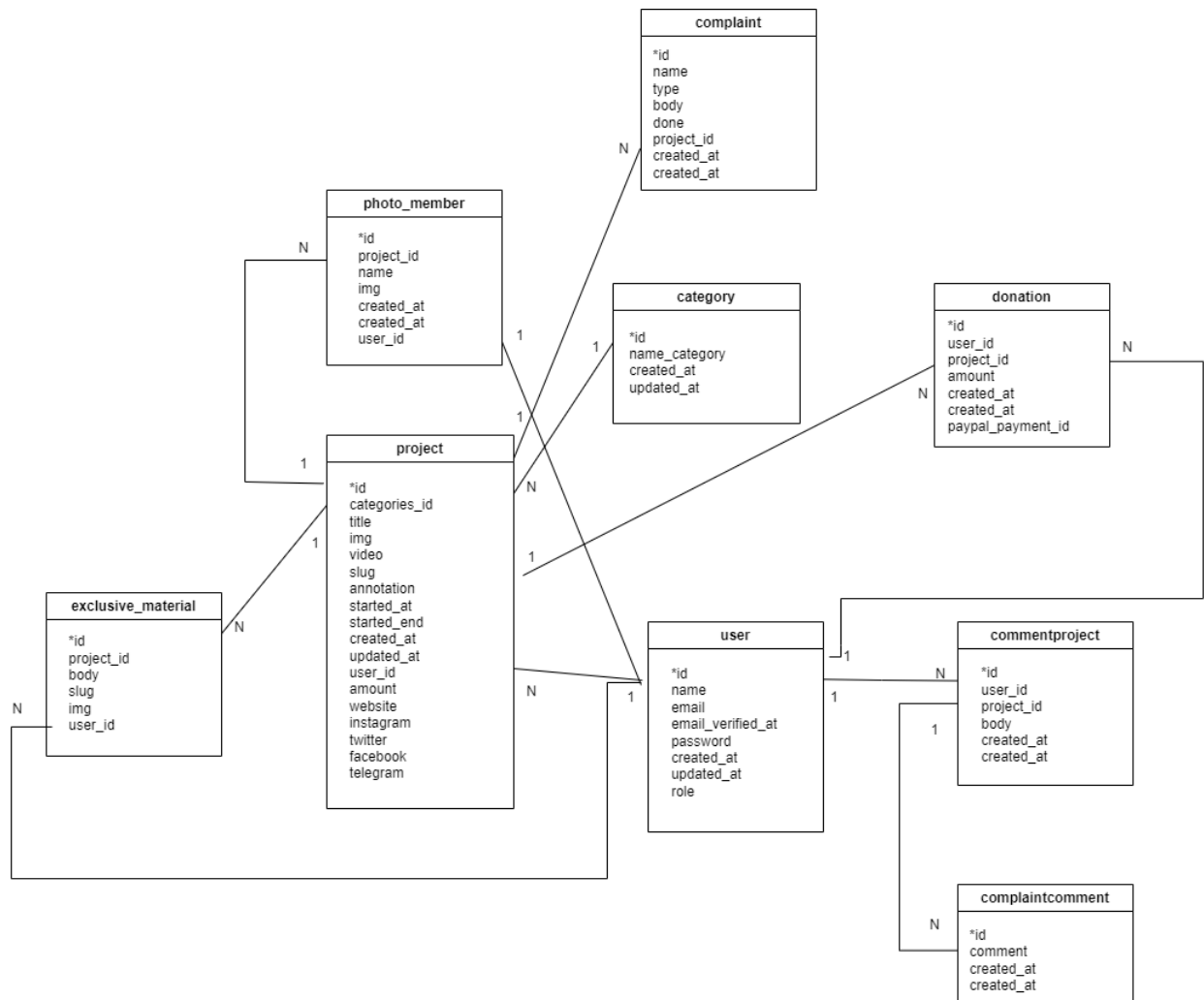


Рисунок 2.5 – Схема БД

2.4 Структурна схема програмного проєкту

Для розробки програмного забезпечення було використано архітектурний патерн Model–View–Controller (рис. 2.6).

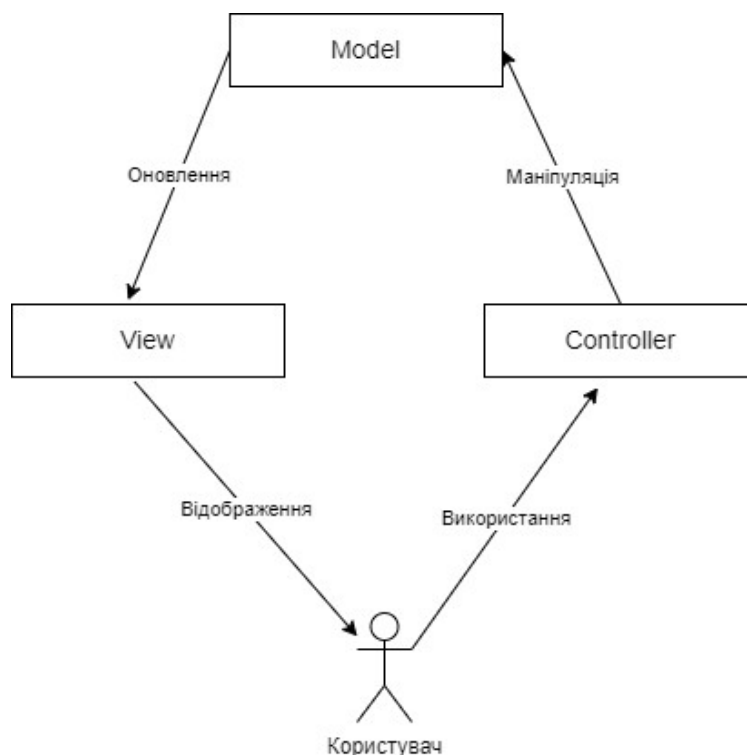


Рисунок 2.6 – Архітектурний патерн Model-View-Controller

Патерн Model-View-Controller складається з таких компонентів:

- model — відповідає за основні дані та бізнес-логіку програми;
- view – відповідає за візуалізацію даних, які зберігаються в моделі;
- controller – відповідає за керування взаємодією між моделлю та представленням.

Архітектура фреймворку Laravel заснована на шаблоні проєктування Model-View-Controller та складається з декількох директорій, які призначені для внутрішньої роботи програми.

Директорія views (рис. 2.7) містить файли, які відповідають за візуалізацію даних (View), які показують, як дані повинні відображатися в вебінтерфейсі. Також в назвах файлів можна помітити атрибут blade, який позначає файли як шаблони для створення макетів вебінтерфейсу. У самих файлах знаходиться HTML-код, який відповідає за побудову структури сторінок, при відображенні у самому браузері.

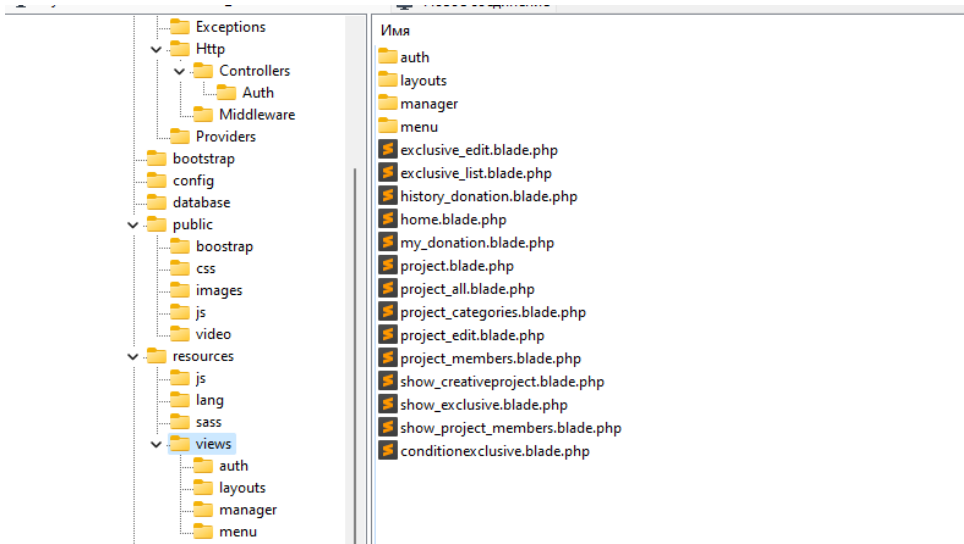


Рисунок 2.7 – Директорія views

Директорія routes (рис. 2.8) містить файли, що відповідають за визначення маршрутів (routes) у застосунку. Можна одразу побачити основні файли маршрутів web.php та api.php. Файл web.php відповідає за визначення маршрутів, які приймає HTTP-запити від користувачів, а файл api.php відповідає за визначення маршрутів, які приймає API-запити від користувачів.

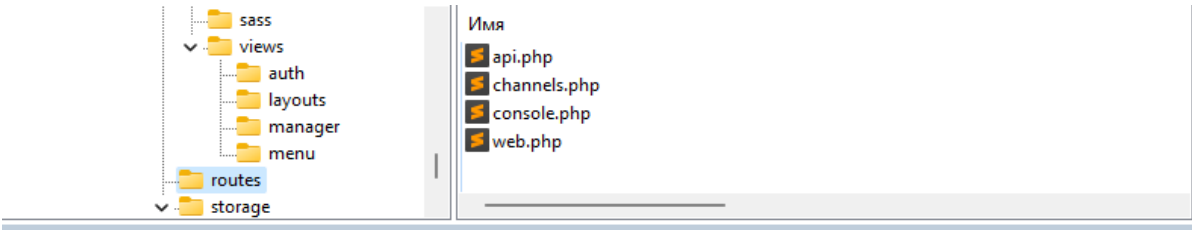


Рисунок 2.8 – Директорія routes

Директорія App (рис. 2.9) містить файли, які відповідають за бізнес-логіку програми, та керування функціями вебфреймворку Laravel.

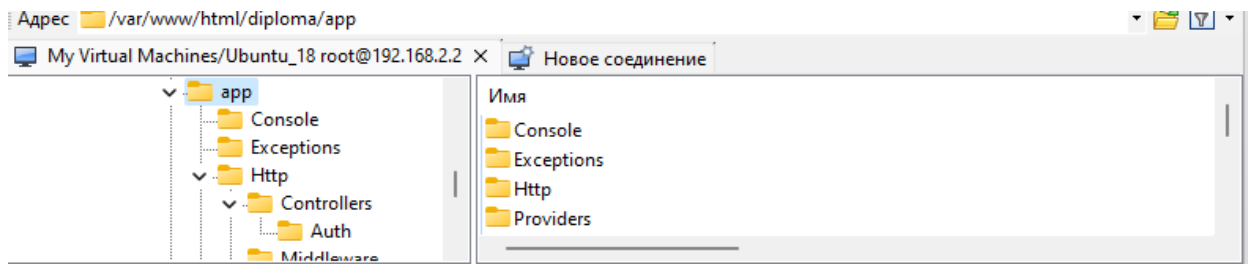


Рисунок 2.9 – Директорія App

Також директорія App містить у собі файли, які відповідають за основні дані (Models), та представляють таблиці БД (рис. 2.10). Класи моделей можуть керувати даними в кожній таблиці БД та встановлювати між таблицями відношення EloquentORM.

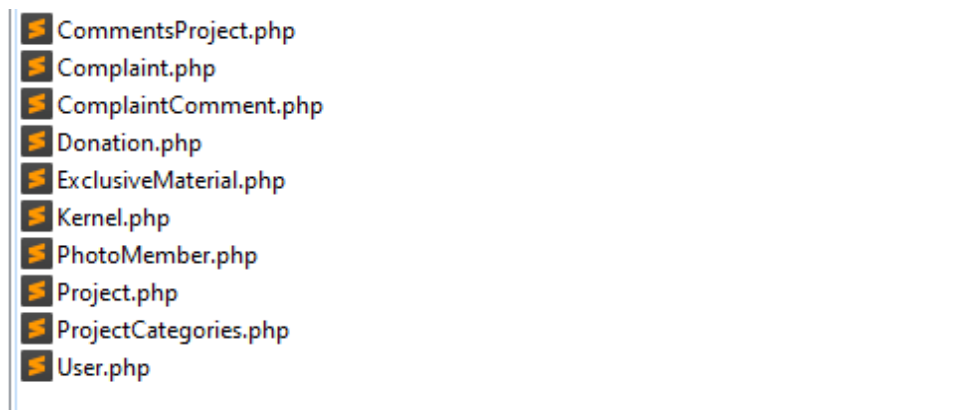


Рисунок 2.10 – Директорія Models

Директорія Controllers (рис. 2.11) містить файли, які відповідають за керування взаємодією між моделлю та представленням (controller), та обробляють HTTP-запит користувача.

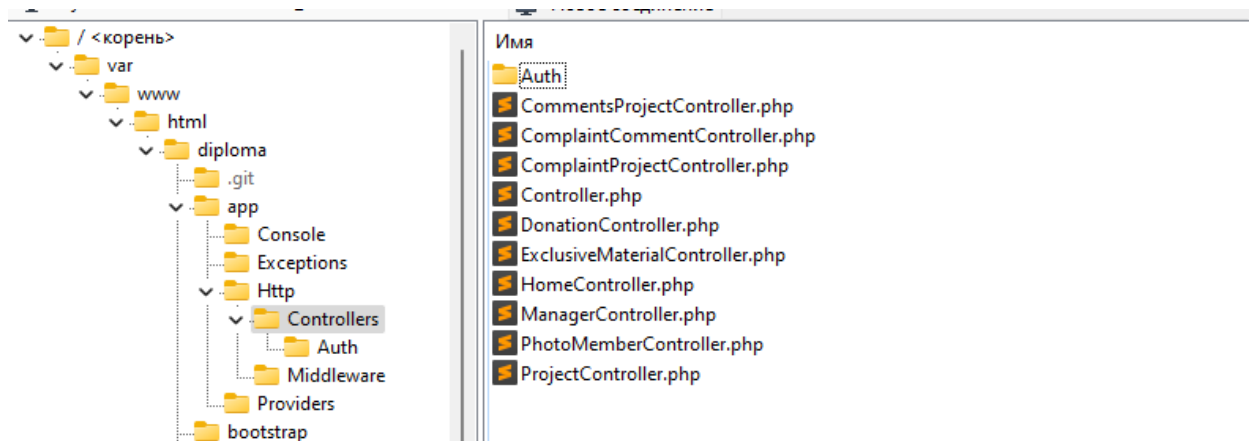


Рисунок 2.11 – Директорія Controllers

2.5 Висновки за розділом 2

В цьому розділі було обрано такі засоби проєктування та подальшої розробки: мову програмування PHP, вебфреймворк Laravel, шаблон проєктування Model-View-Controller, обрана БД MySQL. Представлено діаграми прецедентів творця, спонсора, неавторизованого користувача та менеджера. Реалізовано проєктування БД.

3 РОЗРОБКА ПРОГРАМИ

3.1 Алгоритм функціонування програмного забезпечення

Алгоритм функціонування програмного забезпечення розрахований на чотири типи користувачів, які зазначені у другому розділі.

Після початку роботи програмного забезпечення користувачеві пропонується переглянути творчі проекти в кожній категорії та перейти на сторінку певного творчого проекту.

Для того щоб користувач увійшов до системи, необхідно пройти авторизацію. Якщо обліковий запис не існує, потрібно зареєструватися.

Після авторизації визначається роль користувача.

Якщо користувач має роль творця, то він може керувати власними проектами та переглядати фінансові пожертвування на кожний проект у вигляді аналітики та графіків.

Якщо користувач має роль спонсора, для нього є доступними фінансові пожертвування для кожного творчого проекту, надається можливість переглядати історію фінансових внесків для творчих проектів та залишати відгук під кожним проектом.

Користувач, що має роль менеджера, керує всіма творчими проектами, категоріями для проектів, та проводить модерацію кожної скарги на будь-який творчий проект та коментар.

Порядок роботи з функціями програми всіх типів користувачів показаний на рис. 3.1

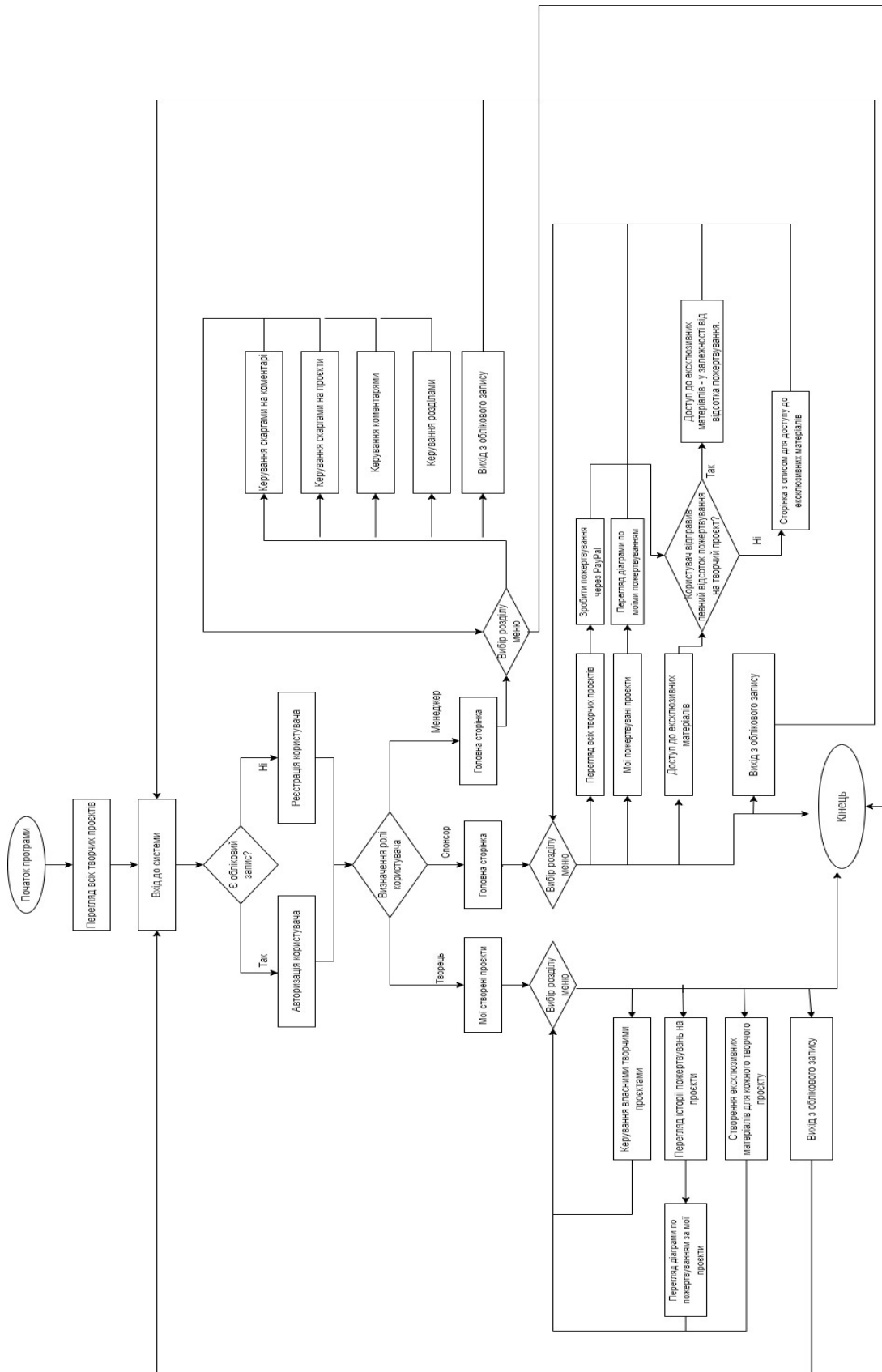


Рисунок 3.1 – Схема функціонування

3.2 Опис реалізованих класів

Під час розробки класів програмного забезпечення було використано PHP вебфреймворк Laravel, тому основна класова архітектура програмного забезпечення побудована на MVC паттерні.

Клас моделі User визначає доступ до даних користувача в таблиці user і містить такі поля:

- `const ROLE_USER` – константа, яка позначає роль звичайного користувача (спонсора) в системі. Дана константа передає ціле число 0, яке буде позначати дану роль користувача;

- `const ROLE_CREATOR` – константа, яка позначає роль творця в системі. Дана константа передає ціле число 1, яке буде позначати дану роль користувача;

- `const ROLE_MANAGER` – константа, яка позначає роль менеджера в системі. Дана константа передає ціле число 2, яке буде позначати дану роль користувача;

- `protected fillable` – звичайна масив, який визначає певні поля з таблиці user у БД. Масив fillable дозволяє уникнути масового заповнення зловмисниками небажаних полів у таблиці для маніпуляції певних даних;

- `public static function getRoles()` – визначає роль кожного користувача на основі заданого типу користувача та виводить певну роль авторизованого користувача у програмі.

Клас моделі ExclusiveMaterial визначає доступ до ексклюзивних матеріалів в таблиці exclusive_material, і містить такі поля:

- `use Sluggable` – імпорт класу Sluggable, який знаходиться в пакеті Cviebrock\EloquentSluggable для створення URL-адреси для кожного запису;

- `protected fillable` – звичайний масив, який визначає певні поля з таблиці exclusive_material у БД. Масив fillable дозволяє уникнути масового заповнення зловмисниками небажаних полів у таблиці для маніпуляції певними даними;

– `public function user()` – метод, який визначає відношення один-до-багатьох між моделями `ExclusiveMaterial` і `User` на основі зовнішнього ключа `user_id`. Даний метод дозволяє визначити користувача, який створив ексклюзивний матеріал;

– `public function project()` – метод, який визначає відношення один-до-багатьох між моделями `ExclusiveMaterial` і `Project` на основі зовнішнього ключа `project_id`. Даний метод дозволяє визначити творчий проєкт, котрий відноситься до ексклюзивних матеріалів;

– `public function sluggable():array` – метод який повертає масив конфігурації з пакету `Cviebrock\EloquentSluggable`. Даний метод визначає поля, які будуть використовуватися для створення URL-адреси для кожного запису.

Клас моделі `Project` визначає доступ до творчих проєктів в таблиці `project` і містить такі поля:

– `use Sluggable` – імпорт класу `Sluggable`, який знаходиться в пакеті `Cviebrock\EloquentSluggable` для створення URL адреси для кожного запису;

– `protected fillable` – звичайний масив, котрий визначає певні поля з таблиці `project` у БД. Масив `fillable` дозволяє уникнути масового заповнення зловмисниками небажаних полів у таблиці для маніпуляції певними даними;

– `public function category()` – метод, який визначає відношення один-до-багатьох між моделлю `Project` та моделлю `ProjectCategories` на основі зовнішнього ключа `categories_id`. Даний метод дозволяє визначити категорію, в котрій знаходяться творчі проєкти;

– `public function comments()` – метод, який визначає відношення один-до-багатьох між моделлю `Project` та моделлю `CommentsProject`. Даний метод дозволяє визначити коментарі для певного творчого проєкту;

– `public function sluggable():array` – метод який повертає масив конфігурації з пакету `Cviebrock\EloquentSluggable`. Даний метод визначає поля, які будуть використовуватись для створення URL-адреси для кожного запису;

– `public function donations()` – метод, який визначає відношення один-до-багатьох між моделлю `Project`, і моделлю `Donation`. Даний метод дозволяє визначити пожертвування для певного творчого проєкту.

Клас моделі `Donation` визначає доступ до фінансових пожертвувань в таблиці `donation` і містить такі поля:

– `protected fillable` – звичайний масив, котрий визначає певні поля з таблиці `donation` у БД. Масив `fillable` дозволяє уникнути масового заповнення зловмисниками небажаних полів у таблиці для маніпуляції певними даними;

– `public function project()` – метод, який визначає відношення один-до-багатьох між моделями `Donation`, і `Project` на основі зовнішнього ключа `project_id`;

– `public function user()` – метод, який визначає відношення один-до-багатьох між моделями `Donation` і `User` на основі зовнішнього ключа `user_id`.

Клас моделі `ProjectCategories` визначає доступ до категорій творчих проєктів в таблиці `category` і містить такі поля:

– `protected fillable` – звичайний масив, який визначає певні поля з таблиці `category` у БД. Масив `fillable` дозволяє уникнути масового заповнення зловмисниками небажаних полів у таблиці для маніпуляції певними даними;

– `public function project()` – метод, який визначає відношення один-до-багатьох між моделями `ProjectCategories` і `Project` на основі зовнішнього ключа `project_id`.

Клас моделі `CommentsProject` визначає доступ до коментарів проєктів в таблиці `comment` і містить такі поля:

– `protected fillable` – звичайний масив, який визначає певні поля з таблиці `comment` у БД. Масив `fillable` дозволяє уникнути масового заповнення зловмисниками небажаних полів у таблиці для маніпуляції певними даними;

– `public function project()` – метод, який визначає відношення один-до-багатьох між моделями `CommentsProject` і `Project` на основі зовнішнього ключа `project_id`;

Клас моделі PhotoMember визначає доступ до фотографій учасників кожного творчого проєкту в таблиці photo_member і містить такі поля:

- protected fillable – звичайний масив, який визначає певні поля з таблиці photo_member у БД. Масив fillable дозволяє уникнути масового заповнення зловмисниками небажаних полів у таблиці для маніпуляції певними даними;

- public function project() – метод, що визначає відношення один-до-багатьох між моделями PhotoMember, і Project на основі зовнішнього ключа project_id.

Клас моделі Complaint визначає доступ до скарг на кожний творчий проєкт в таблиці comment і містить такі поля:

- protected fillable – звичайний масив, котрий визначає певні поля з таблиці complaint у БД. Масив fillable дозволяє уникнути масового заповнення зловмисниками небажаних полів у таблиці для маніпуляції певними даними;

- public function project() – метод, який визначає відношення один-до-багатьох між моделями Complaint і Project на основі зовнішнього ключа project_id.

Клас моделі ComplaintComment визначає доступ до скарг на кожний коментар в таблиці complaintcomment і містить такі поля:

- protected fillable – звичайний масив, котрий визначає певні поля з таблиці complaintcomment у БД. Масив fillable дозволяє уникнути масового заповнення зловмисниками небажаних полів у таблиці для маніпуляції певними даними;

- public function project() – метод, який визначає відношення один-до-багатьох між моделями ComplaintComment і Project на основі зовнішнього ключа project_id;

- public function comments() – метод, що визначає відношення один-до-багатьох між моделями ComplaintComment і Comment на основі зовнішнього ключа comment.

Клас ProjectController служить для обробки всіх вхідних HTTP-запитів від користувачів, пов'язаних з творчими проєктами, включаючи відображення,

створення, оновлення та видалення творчих проєктів. Клас `ProjectController` містить такі методи:

– `public function index()` – забезпечує відображення списку всіх творчих проєктів певного автора (творця) на головній сторінці. Метод містить модель `Project`, яка відображає творчі проєкти користувача по його унікальному ідентифікатору, та модель `ProjectCategories`, котра показує категорії, в яких знаходяться творчі проєкти. Також в методі міститься бібліотека `Carbon::now()`, яка показує терміни збору пожертвувань для кожного творчого проєкту. Також дані про кожний проєкт контролер передає до шаблону `project.blade.php`;

– `public function ProjectForm(Request $request)` – відповідає за додання творчих проєктів до БД. В даному методі міститься валідація для перевірки на заповнення форми для створення проєкту, після проходження валідації цей проєкт додається до БД. Також в цьому методі відбувається завантаження зображення та відеопрезентації створеного творчого проєкту до файлової системи, для подальшого відображення у програмі;

– `public function update(Request $request,$id)` – метод відповідає за оновлення існуючих творчих проєктів у БД. В даному методі міститься валідація для перевірки на заповнення форми для оновлення певного творчого проєкту, після проходження валідації цей проєкт оновлюється в БД. Також в цьому методі відбувається оновлення вже існуючого зображення та відеопрезентації творчого проєкту в файлової системі;

– `public function categories_show ($id)` – метод відповідає за відображення всіх творчих проєктів, які пов'язані із певною творчою категорією. В цьому методі визначаються творчі проєкти по переданому унікальному ідентифікатору певної категорії, а потім передається масив творчих проєктів до шаблону `project_categories.blade.php`;

– `public function show_creativeproject($slug)` – відповідає за відображення однієї сторінки з певним творчим проєктом. В цьому методі спочатку виконується пошук у БД творчого проєкту по його заданій URL-адресі, після цього отримується певний творчий проєкт, термін збору пожертвування,

фактична сума зібраного пожертвування. Потім передається масив з певним творчим проектом до шаблону `show_creativeproject.blade.php`;

- `public function allproject()` – метод відповідає за відображення всіх існуючих творчих проектів у шаблоні `project_all.blade.php`;

- `public function showProject($id)` – відображає сторінку для оновлення творчого проекту, яка доступна для користувача, який має роль творця. В цьому методі передається певний творчий проект з БД до шаблону `project_all.blade.php`;

- `public function destroy($id)` – метод видаляє певний творчий проект у БД за заданим унікальним ідентифікатором. В даному методі спочатку видаляється певний творчий проект з БД, а потім видаляються всі пожертвування, котрі пов'язані з цим проектом.

Клас `DonationController` служить для обробки всіх вхідних HTTP-запитів від користувачів, пов'язаних з фінансовими пожертвуваннями на кожен творчий проект, включно з обробкою вхідних платежів, здійснених через платіжну систему PayPal. Клас `DonationController` містить такі методи:

- `public function __construct()` – конструктор класу, в якому зберігаються налаштування та ініціалізація API-контексту для роботи з функціями PayPal. В даному методі відбувається аутентифікація з сервером PayPal за допомогою спеціального згенерованого токена для роботи з API;

- `public function add_donation(Request $request)` – відповідає за відправлення виставленого рахунку користувачу для переведення грошової суми для певного творчого проекту через платіжну систему PayPal. В даному методі відбувається формування рахунку для переведення суми, в якому міститься певна грошова сума, та назва певного творчого проекту. Після формування рахунку, користувача перенаправляють до платіжної системи, де він може відправити свою грошову суму через PayPal;

- `public function getPaymentStatus(Request $request)` – даний метод відповідає за отримання статусу зробленого користувачем грошового переказу за допомогою платіжної системи PayPal. Спершу метод отримує статус

зробленого грошового переказу від користувача, потім додає отриманий статус до БД. Під час обробки грошового переказу відбувається перевірка отриманого статусу: якщо переказ буде успішним, платіж буде додано до загальної суми пожертвувань творчого проєкту;

– `public function show_donated()` – відповідає за отримання списку зроблених грошових пожертвувань певним користувачем та загальну суму пожертвувань за кожний місяць. Метод групує отриманий список з БД грошових переказів по унікальному ідентифікатору користувача, а потім за допомогою циклу обчислює загальну суму зроблених пожертвувань за кожний місяць. Після проведеного обчислення передається список пожертвувань та загальні суми пожертвувань за кожний місяць до шаблону `my_donation.blade.php`;

– `public function history_donated()` – метод відповідає за отримання списку історій зроблених грошових пожертвувань певним користувачем за певний період. Метод групує отриманий список з БД успішних грошових переказів по унікальному ідентифікатору авторизованого користувача. Врешті, даний метод відправляє список пожертвувань до шаблону `history_donation.blade.php`.

Клас `CommentsProjectController` призначений для обробки всіх вхідних HTTP-запитів від користувачів, пов'язаних з зробленими коментарями на кожний творчий проєкт, включно зі створенням та видаленням коментарів. Клас `CommentsProjectController` містить такі методи:

– `public function store(Request $request)` – відповідає за додавання коментарів до БД, після чого повертає повідомлення про успішне додавання коментаря до БД;

– `public function delete($id)` – метод відповідає за видалення певного коментаря по його унікальному ідентифікатору з БД та повертає повідомлення про успішне видалення коментаря з БД;

Клас `ManagerController` необхідний для обробки всіх вхідних HTTP-запитів від користувачів, пов'язаних з керуванням облікового запису

менеджера, в тому числі опрацювання скарг, керування категорій та творчими проєктами.

Клас `ManagerController` містить такі методи:

- `public function project_manager()` – метод, що повертає менеджеру сторінку зі списками усіх творчих проєктів, які відображаються менеджеру в шаблоні `project_manager.blade.php`;

- `public function update_project($id, Request $request)` – дає право менеджеру редагувати існуючі творчі проєкти у БД. В даному методі використовується валідація для перевірки на заповнення форми для оновлення певного творчого проєкту, після проходження валідації цей проєкт оновлюється в БД. Також в цьому методі відбувається оновлення вже існуючого зображення та відеопрезентації творчого проєкту в файловій системі;

- `public function destroy_project($id, Request $request)` – метод, що надає право менеджеру видаляти певний творчий проєкт у БД за заданим унікальним ідентифікатором. В даному методі спочатку видаляється певний творчий проєкт з БД, після цього видаляються всі пожертвування, котрі пов'язані з цим проєктом;

- `public function categoriesmanager_show()` – метод, що повертає сторінку зі списками усіх категорій, які відображаються менеджеру в шаблоні `categoriesmanager_show.blade.php`;

- `public function categoriesmanager_edit($id)` – даний метод надає можливість менеджеру редагувати існуючі категорії у БД. В даному методі міститься валідація для перевірки на заповнення форми для оновлення певної категорії після проходження валідації змінена категорія оновлюється до БД;

- `public function destroy_categories($id)` – метод, завдяки якому менеджер видаляє певну категорію у БД за заданим унікальним ідентифікатором. В даному методі видаляється певна категорія з БД за даними унікальним ідентифікатором;

- `public function categoriesmanager_create(Request $request)` – метод, що дозволяє менеджеру додавати нову категорію до БД. В даному методі міститься

валідація для перевірки на заповнення форми для створення категорії, і після проходження валідації нова категорія додається до БД;

– `public function complaintmanager_show()` – даний метод реалізує можливість менеджера переглядати сторінку зі всіма скаргами на творчі проекти у шаблоні `complaintmanager_show.blade.php`;

– `public function commentsmanager_show()` – метод, котрий дозволяє менеджеру переглядати сторінку зі всіма скаргами на коментарі на кожний творчий проект у шаблоні `commentsmanager_show.blade.php`;

Клас `PhotoMemberController` використовується для обробки всіх вхідних HTTP-запитів від користувачів, пов'язаних з керуванням фотографій учасників кожного творчого проекту, в тому числі додавання, редагування, та видалення фотографій учасників певного творчого проекту. Клас `PhotoMemberController` містить такі методи:

– `public function index()` – метод, який відображає сторінку з зображеннями учасників певного творчого проекту. Спершу метод шукає всі записи фотографій учасників творчого проекту за певним ідентифікатором творчого проекту. Після цього пошуку відбувається передання колекції фотографій учасників творчих проектів до шаблону `project_members.blade.php`;

– `public function AddMemberForm(Request $request)` – метод відповідає за додавання фотографій учасників творчих проектів до БД. В даному методі міститься валідація для перевірки на заповнення форми для додавання фотографій учасників, після проходження валідації фотографія учасників, його ім'я додається до БД. Також в цьому методі відбувається завантаження зображення учасника творчого проекту до файлової системи, для подальшого відображення у програмі;

– `public function delete($id)` – метод, що дозволяє видаляти певну фотографію учасника творчого проекту у БД за заданим унікальним ідентифікатором учасника творчого проекту;

– `public function show($id)` – метод, завдяки якому можливо переглянути сторінку з фотографії та ім'я учасника певного творчого проекту. На початку

виконується пошук певної фотографії у БД за заданим унікальним ідентифікатором учасника творчого проєкту, а потім метод повертає певну фотографію та ім'я учасника певного творчого проєкту у шаблон `show_project_members.blade.php`.

Клас `ComplaintProjectController` призначений для обробки всіх вхідних HTTP-запитів від користувачів, пов'язаних зі скаргами на творчі проєкти, включаючи відправлення, та опрацювання скарги на певний творчий проєкт. Клас `ComplaintProjectController` містить такі методи:

- `public function store(Request $request)` – відповідає за додавання скарги на творчий проєкт до БД. В даному методі міститься валідація для перевірки вибору користувачем причини скарги, у разі проходження валідації – скарга додається до БД;

- `public function complaint_done(Request $request)` – метод відповідає за позначення скарги на творчий проєкт як «опрацьована менеджером». Спершу відбувається пошук скарги на творчий проєкт по його унікальному ідентифікатору у БД. Відбувається оновлення значення `done` у БД, якщо скарга була вже опрацьована менеджером. Врешті функція `back()` повертає менеджера назад на попередню сторінку.

Клас `ComplaintCommentController` потрібний для обробки всіх вхідних HTTP-запитів від користувачів, пов'язаних зі скаргами на коментарі, в тому числі відправлення та опрацювання скарг на певні коментарі. Клас `ComplaintCommentController` містить такі методи:

- `public function store(Request $request)` – метод відповідає за додавання скарги на коментар до БД. В даному методі міститься валідація для перевірки вибору користувачем причини скарги на коментар, після проходження валідації скарга додається до БД;

- `public function complaint_done(Request $request)` – метод відповідає за позначення скарги на коментар як «опрацьована менеджером». Спочатку відбувається пошук скарги на коментар по його унікальному ідентифікатору у БД. Потім оновлюється значення `done` у БД, якщо скарга була вже опрацьована

менеджером. Врешті функція `back()` повертає менеджера назад на попередню сторінку.

Клас `ExclusiveMaterialController` необхідний для обробки всіх вхідних HTTP-запитів від користувачів, пов'язаних із ексклюзивними матеріалами на творчі проєкти, зокрема з керуванням ексклюзивними матеріалами для певного творчого проєкту.

Клас `ExclusiveMaterialController` містить такі методи:

- `public function index()` – метод, що забезпечує відображення списку всіх ексклюзивних матеріалів певного творчого проєкту. Метод містить модель `ExclusiveMaterial` для знаходження всіх ексклюзивних матеріалів певного творчого проєкту. По завершенню, метод передає знайдені ексклюзивні матеріали та проєкти до шаблону представлення `exclusive_list.blade.php`;

- `public function add(Request $request)` – метод відповідає за додавання ексклюзивних матеріалів до БД. Даний метод містить валідацію для перевірки на заповнення форми для створення ексклюзивних матеріалів до БД, після проходження валідації даний ексклюзивний матеріал додається до БД. Також в цьому методі відбувається завантаження зображення створеного ексклюзивного матеріалу до файлової системи для подальшого відображення у програмі;

- `public function destroy($id)` – метод видаляє певний ексклюзивний матеріал у БД за заданим унікальним ідентифікатором;

- `public function exclusive_edit($id)` – метод, що необхідний для оновлення існуючих ексклюзивних матеріалів у БД. В даному методі використовується валідація для перевірки на заповнення форми для оновлення певних ексклюзивних матеріалів, після проходження валідації даний ексклюзивний матеріал оновлюється у БД;

- `public function show($slug)` – даний метод дозволяє переглядати ексклюзивні матеріали на певний період, залежно від відсотка пожертвування. Наприклад, якщо користувач пожертвує більш як 1 % від поставленої суми, йому надається доступ на 10 днів. Якщо користувач пожертвує більш як 10 % від поставленої суми, то йому надається доступ на 100 днів. Якщо користувач

внесе більш як 50 % поставленої суми, в такому разі йому надається доступ назавжди. У методі відбувається пошук творчого проєкту за унікальної URL-адреси і всіх ексклюзивних матеріалів, пов'язаних з ним. Потім у методі обраховується сума пожертвування поточного користувача, після чого визначається можливість надати доступ до ексклюзивних матеріалів. Те, на який термін буде надано доступ, залежить від відсотка пожертвування.

3.3 Висновки за розділом 3

Було представлено результати розробки програми, в яких описано структуру розробленого рішення згідно з відповідною структурною схемою програми та розкрито структуру основних класів, що були реалізовані.

4 ЕКСПЛУАТАЦІЯ ТА ТЕСТУВАННЯ ПРОГРАМИ

4.1 Призначення програми

Програма призначена, насамперед, для допомоги авторам, творцям і творчим особливостям збирати кошти на реалізацію своїх творчих проєктів, залучаючи пожертвування від звичайних людей.

4.2 Умови виконання програми

До вебсервера, на якому виконується програмне забезпечення, висуваються наступні апаратні вимоги:

- 32-розрядний (x86) або 64-розрядний (x64) процесор 1 ГГц;
- 1 Гб (для 32-розрядної системи) або 2 Гб (для 64-розрядної системи) оперативної пам'яті;
- операційна система CentOS або Linux Ubuntu;
- 16 Гб (для 32-розрядної системи) або 20 Гб (для 64 розрядної системи) на жорсткому диску.

4.3 Виконання програми

Для виконання програми необхідно запустити локальний вебсервер. Після цього програма стає доступною для певного користувача (рис. 4.1). Користувач може переглядати категорії з творчими проєктами, а також переходити на певну сторінку творчого проєкту.

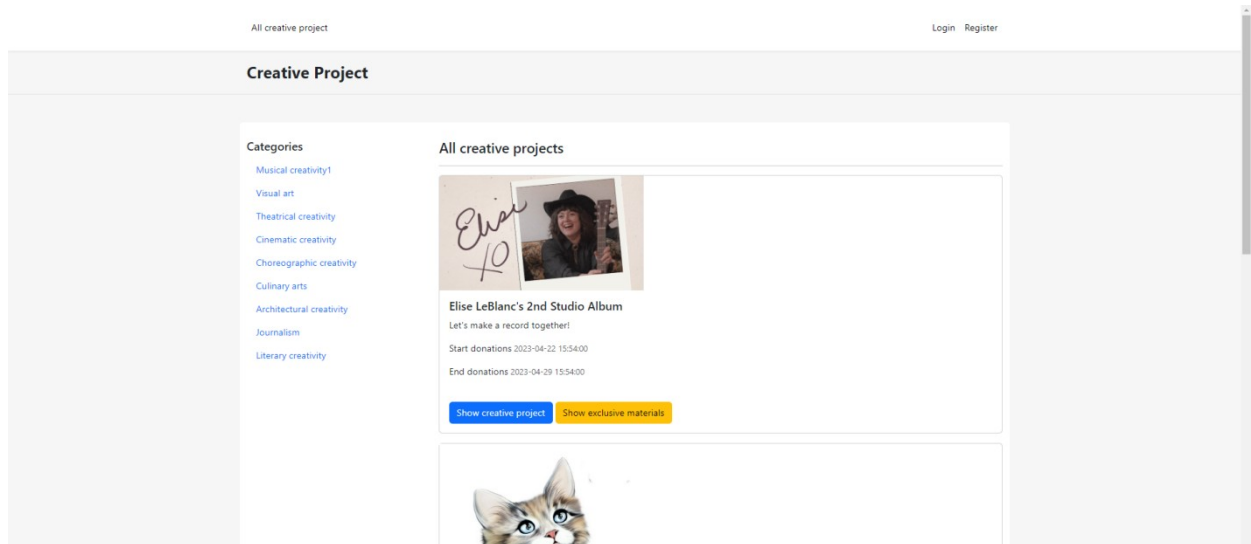
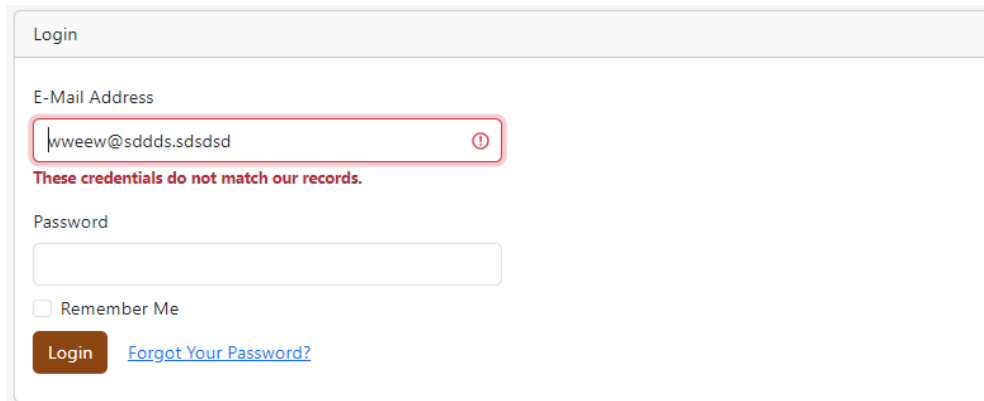


Рисунок 4.1 – Головна сторінка, яка відкривається при запуску застосунку

Далі користувач повинен авторизуватися або зареєструватися в системі. У верхньому лівому куті вікна доступні дві опції: вхід в обліковий запис для зареєстрованих користувачів та реєстрація для нових користувачів.

При переході на сторінку авторизації (рис. 4.2), відображається форма для введення логіну та паролю. Якщо дані введені некоректно, програма повідомляє про це користувача, показуючи відповідне повідомлення, та підкреслює некоректно введене текстове поле (рис. 4.3).

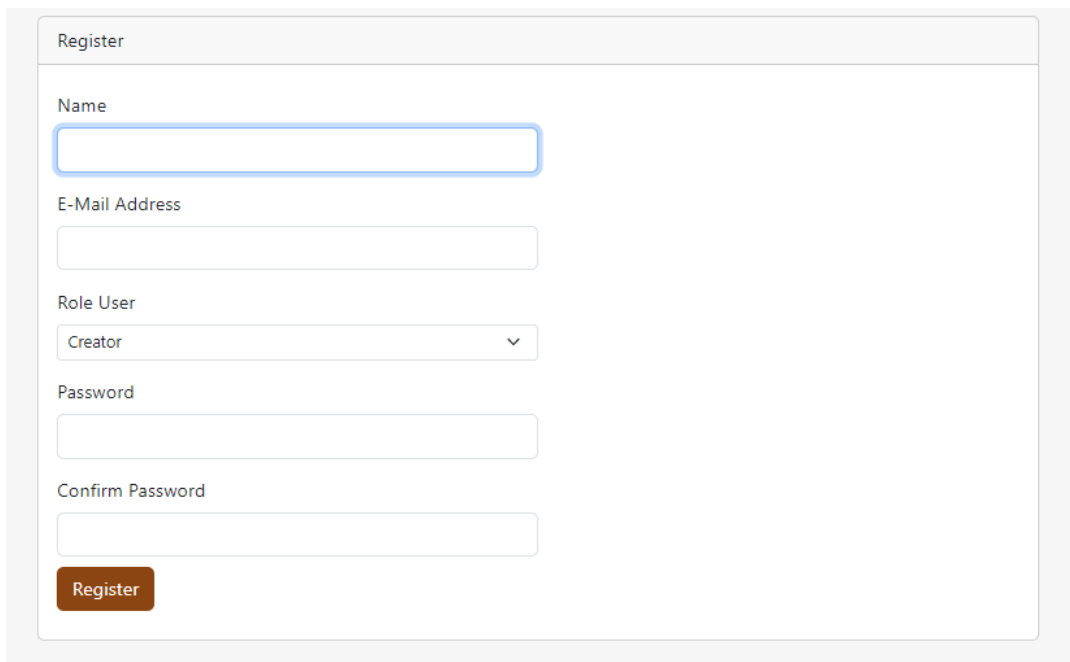
Рисунок 4.2 – Сторінка авторизації



The image shows a login form titled "Login". It contains two input fields: "E-Mail Address" and "Password". The "E-Mail Address" field contains the text "wwew@sddd.sdsdsd" and is highlighted with a red border. Below this field, a red error message reads: "These credentials do not match our records." Below the "E-Mail Address" field is the "Password" field, which is empty. Below the "Password" field is a checkbox labeled "Remember Me". At the bottom of the form, there is a brown "Login" button and a blue link labeled "Forgot Your Password?".

Рисунок 4.3 – Повідомлення про помилку при авторизації

На сторінці реєстрації (рис. 4.4) користувач відразу бачить форму створення облікового запису. Під час створення облікового запису користувач заповнює такі поля: логін, пароль, адресу електронної пошти та може вибрати свою роль. Після того, як користувач зареєструвався, він відразу авторизується в системі під своєю роллю.



The image shows a registration form titled "Register". It contains five input fields: "Name", "E-Mail Address", "Role User", "Password", and "Confirm Password". The "Name" field is highlighted with a blue border. The "Role User" field is a dropdown menu with "Creator" selected. At the bottom of the form, there is a brown "Register" button.

Рисунок 4.4 – Сторінка реєстрації

Розглянемо роль творця. Після введення коректних авторизаційних даних, творець потрапляє на сторінку керування творчими проєктами (рис.4.5). Тут творець створює (рис. 4.6), редагує та видаляє творчі проєкти.

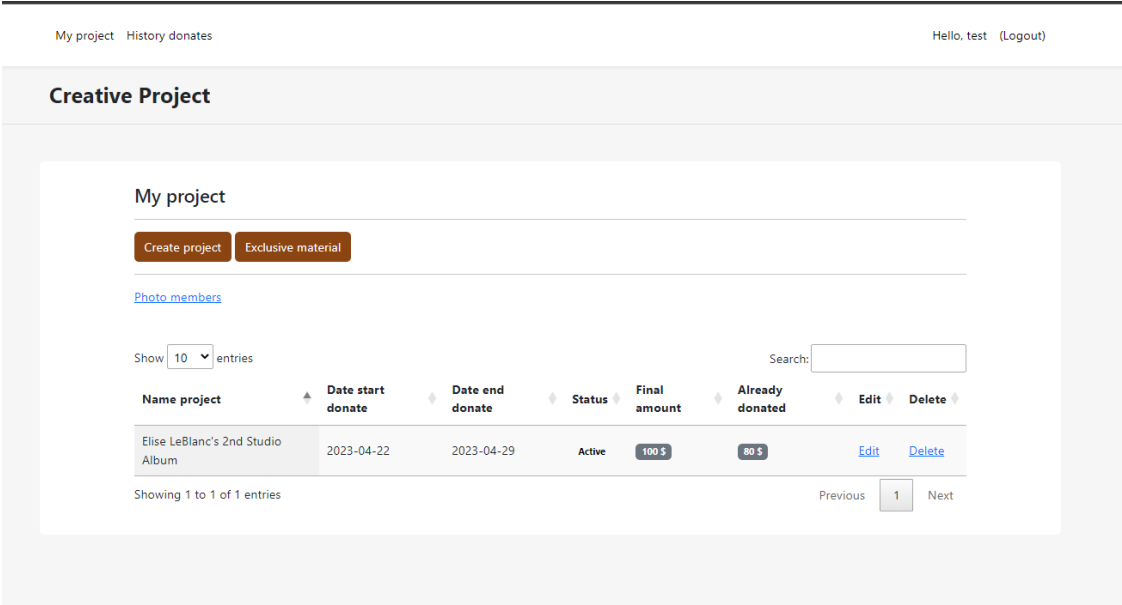


Рисунок 4.5 – Сторінка реєстрації

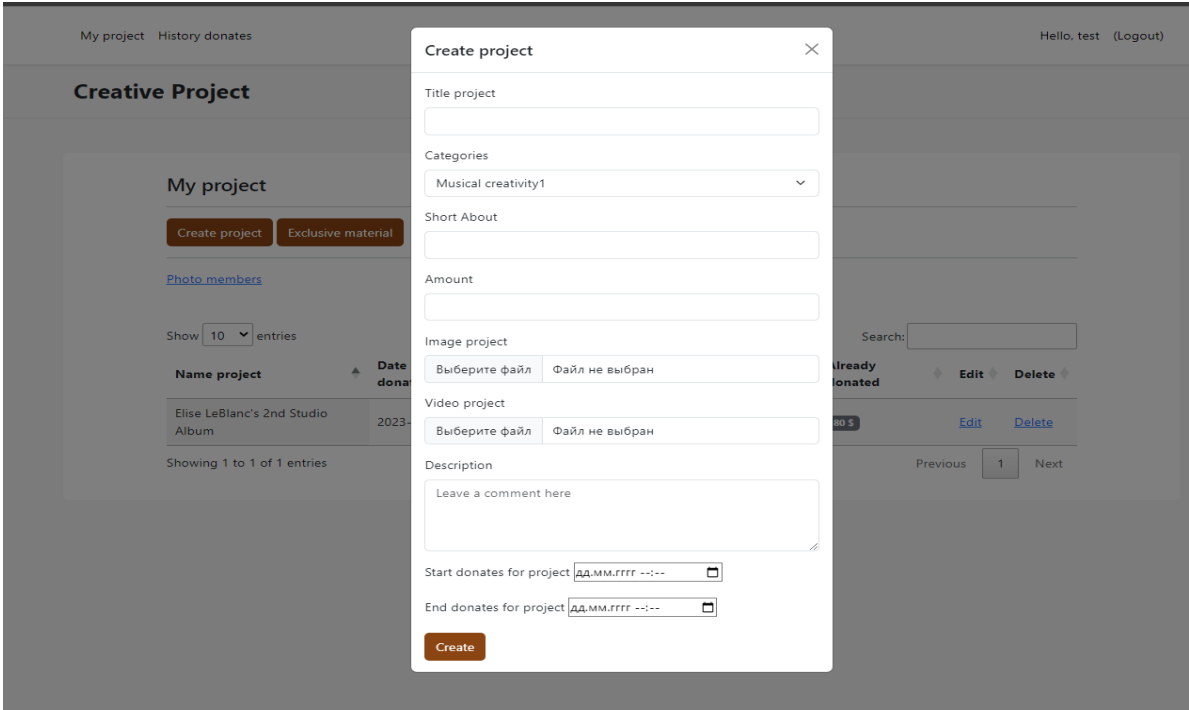


Рисунок 4.6 – Створення творчого проєкту

На сторінці з історіями пожертвування (рис. 4.7), творець може побачити діаграму, яка показує загальну суму пожертвувань за кожний місяць, а також таблицю, де відображено грошові пожертвування від спонсорів для кожного творчого проєкту.

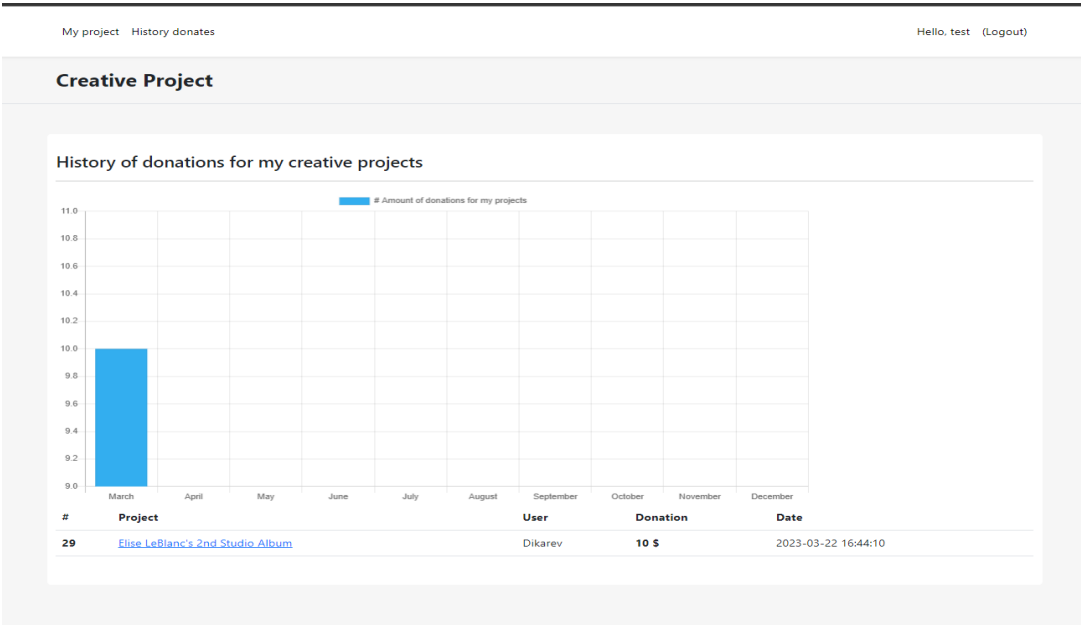


Рисунок 4.7 – Сторінка з історіями пожертвування

Крім того користувач може створювати власні ексклюзивні матеріали (рис. 4.8) для кожного творчого проєкту. У цих матеріалах користувач описує поетапний розвиток свого творчого проєкту для спонсорів, які вже зробили певне грошове пожертвування.

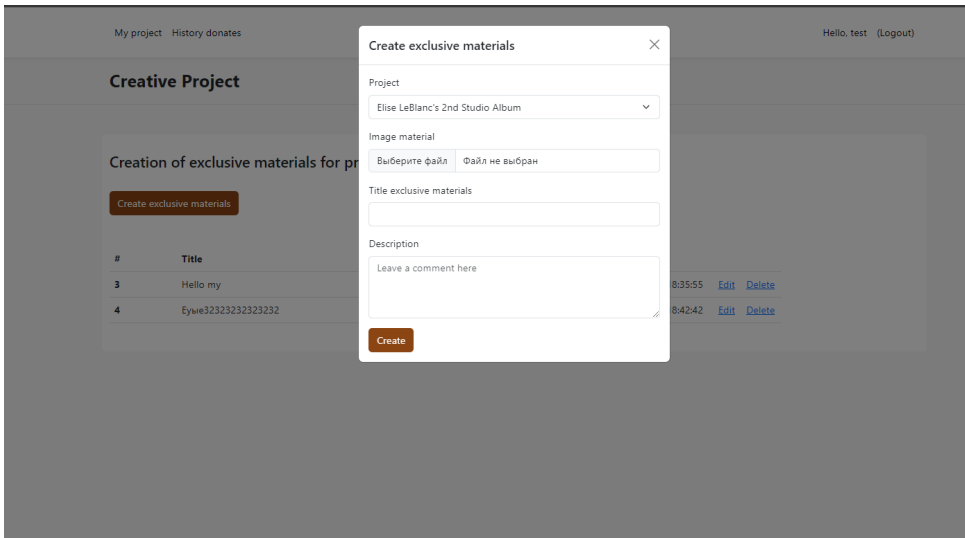


Рисунок 4.8 – Створення ексклюзивних матеріалів для творчого проєкту

Наступною є роль спонсора. Після введення коректних авторизаційних даних, спонсор потрапляє на сторінку зі всіма проєктами. Спонсор переглядає кожну категорію творчих проєктів та переходить на сторінку з певним творчим проєктом (рис. 4.9). На сторінці певного творчого проєкту спонсор додає свої коментарі під кожним творчим проєктом (рис. 4.10). Також спонсор може подати скаргу на творчий проєкт, якщо в цьому проєкті є порушення правил.

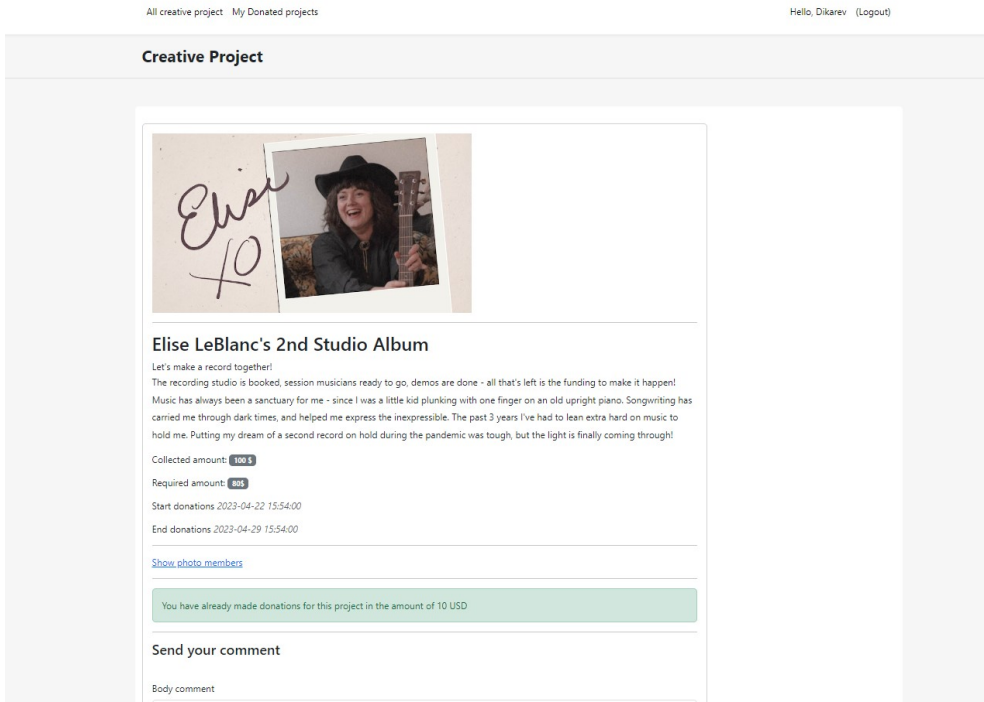


Рисунок 4.9 – Сторінку з певним творчим проєктом

The screenshot shows a web interface for sending a comment. At the top, there is a section titled "Send your comment". Below this title is a text input field labeled "Body comment". Underneath the input field is a blue button labeled "Send comments". Below the button is a red button labeled "Complain about the project". Below the red button is a section titled "Show comment about project (1)". This section contains a comment by "Author: Dikarev" dated "2023-04-24 21:45:33". The comment text is "Hello!". Below the comment text is a blue link labeled "Complain comment".

Рисунок 4.10 – Коментарі під кожним творчим проєктом

Також користувач може зробити грошове пожертвування для певного творчого проєкту. Для цього потрібно натиснути кнопку «Donation to the project», щоб викликати модальне вікно (рис. 4.11) з текстовим полем для вводу грошової суми. Після того, як користувач ввів грошову суму у модальному вікні та натиснув на кнопку «Send donation», користувач потрапляє до платіжної системи PayPal (рис. 4.12), де він бачить виставлений рахунок з його пожертвуванням на певний творчий проєкт. Якщо користувач підтвердив оплату рахунку в PayPal, користувача перенаправляють на сторінку з його пожертвуваннями.

The screenshot shows a modal window titled "Make a donation for this creative project". Inside the modal, there is a text input field labeled "Amount donation" with the number "12" entered. To the right of the input field is a blue button labeled "Send donation". The modal has a close button (X) in the top right corner.

Рисунок 4.11 – Введення певної суми пожертвування для творчого проєкту

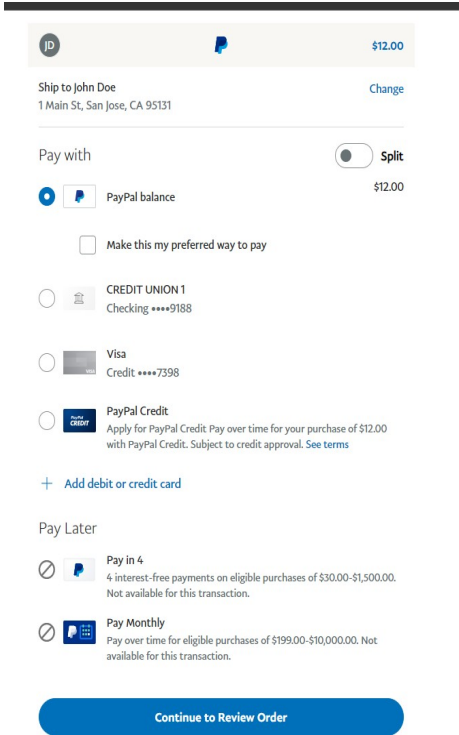


Рисунок 4.12 – Платіжна система Раурал

На сторінці з історіями пожертвувань (рис. 4.13), відображаються діаграма, яка показує загальну суму всіх пожертвувань за кожний місяць, а також таблиця зі всіма пожертвувань на творчі проєкти зроблених певним користувачем.

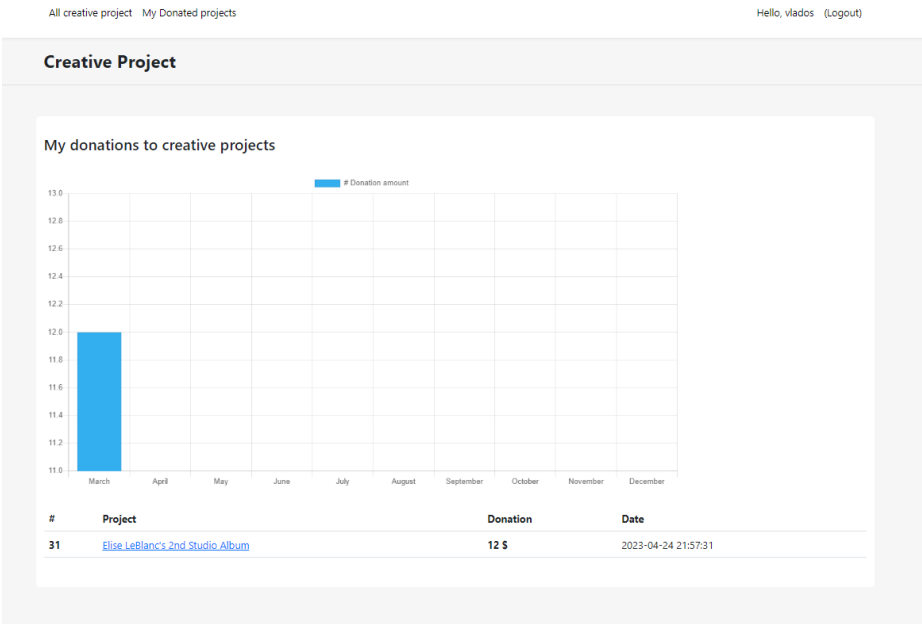


Рисунок 4.13 – Сторінка із історіями пожертвувань

Також після успішного пожертвування на певний творчий проєкт, спонсор отримує безкоштовний доступ до ексклюзивних матеріалів (рис. 4.14) на 10 днів, якщо він пожертвував більше ніж 1 % від суми, або на 100 днів, якщо пожертвував більше ніж 10 %. Якщо спонсор зробив пожертвування понад 50 % від всієї суми, він отримає безкоштовний доступ до контенту назавжди.

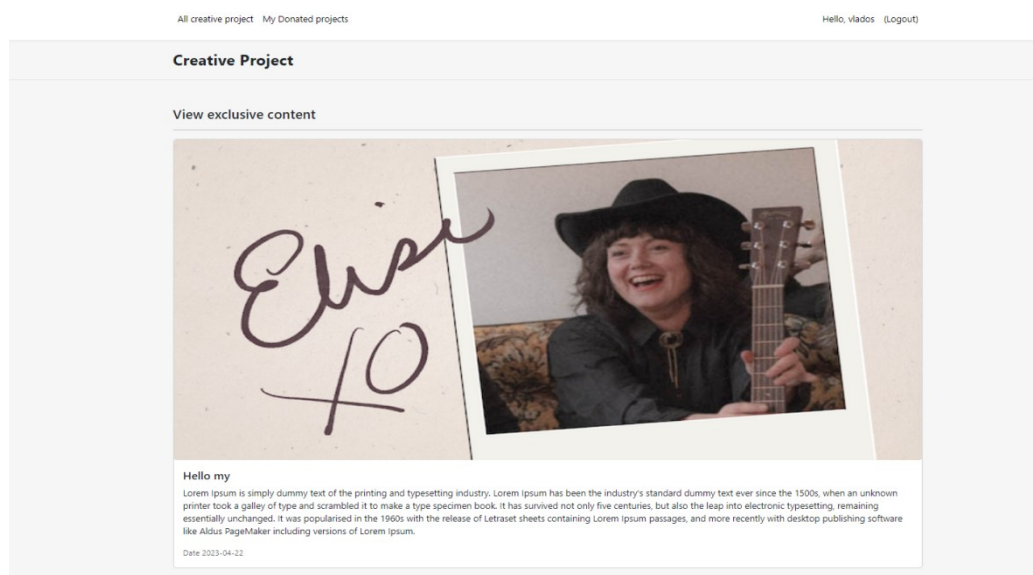


Рисунок 4.14 – Сторінка з ексклюзивним матеріалом

Далі описано роль менеджера. Після введення коректних авторизаційних даних менеджер потрапляє на сторінку зі всіма творчими проєктами (рис 4.15). Менеджеру доступне редагування (рис. 4.16) та видалення творчих проєктів.

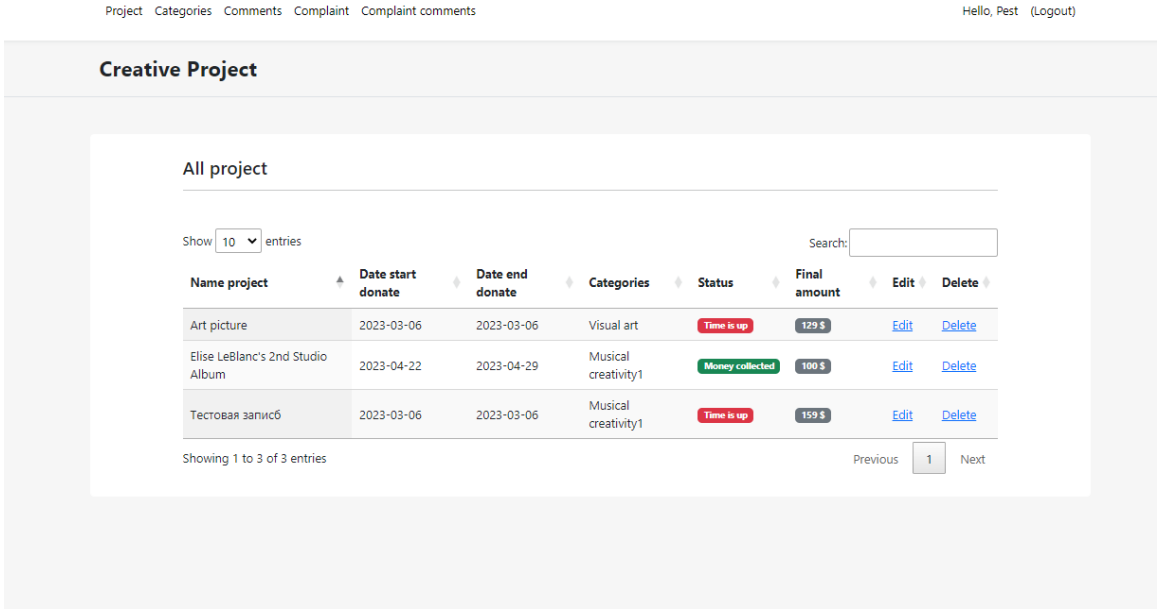


Рисунок 4.15 – Сторінка зі всіма творчими проєктами

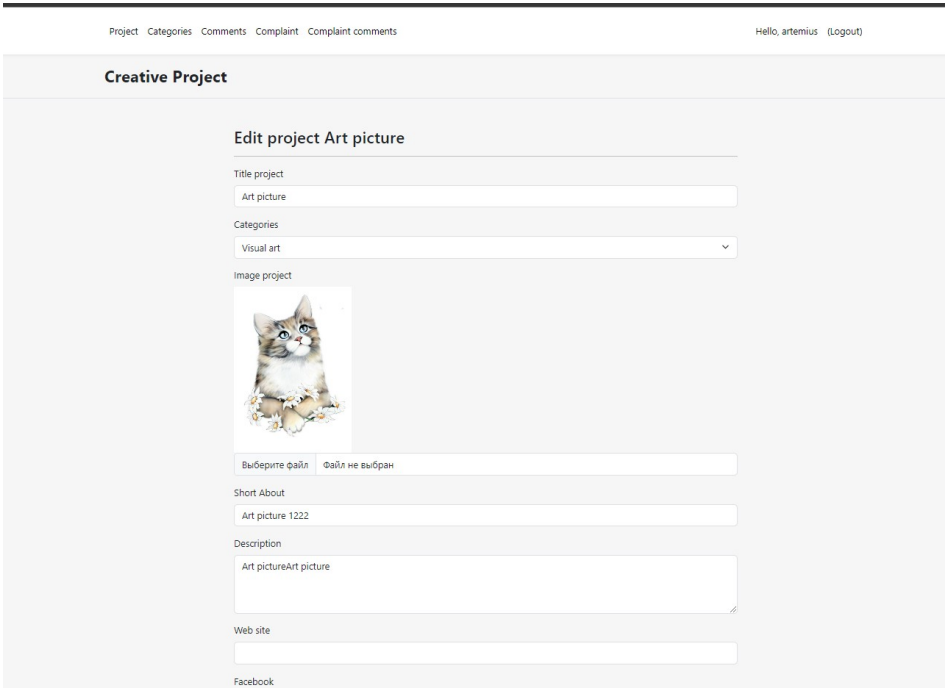


Рисунок 4.16 – Редагування творчого проєкту

При переході на сторінку керування категоріями (рис. 4.17), менеджер створює (рис. 4.18), редагує та видаляє категорії для творчих проєктів.

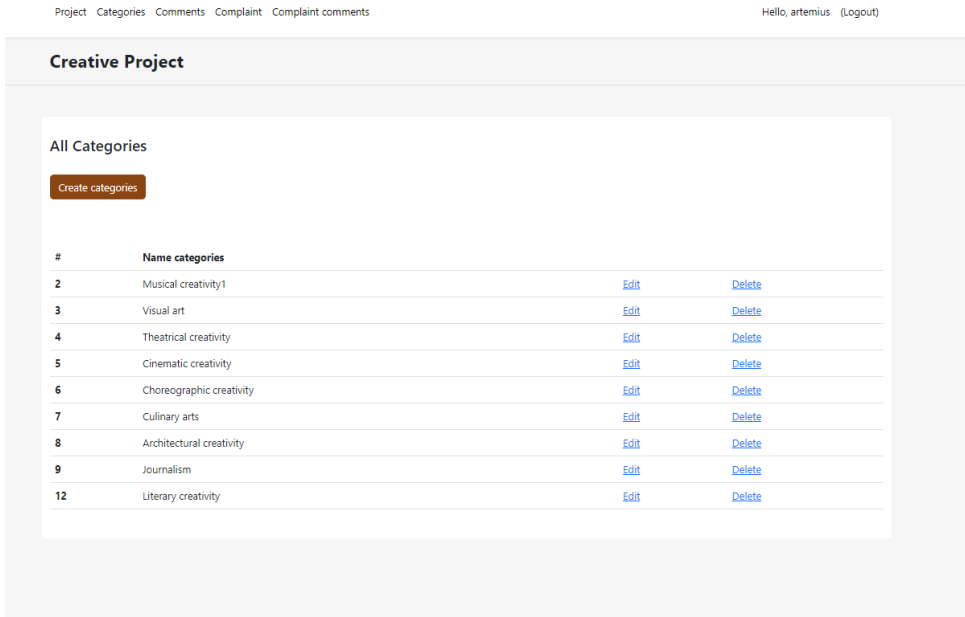


Рисунок 4.17 – Керування категоріями

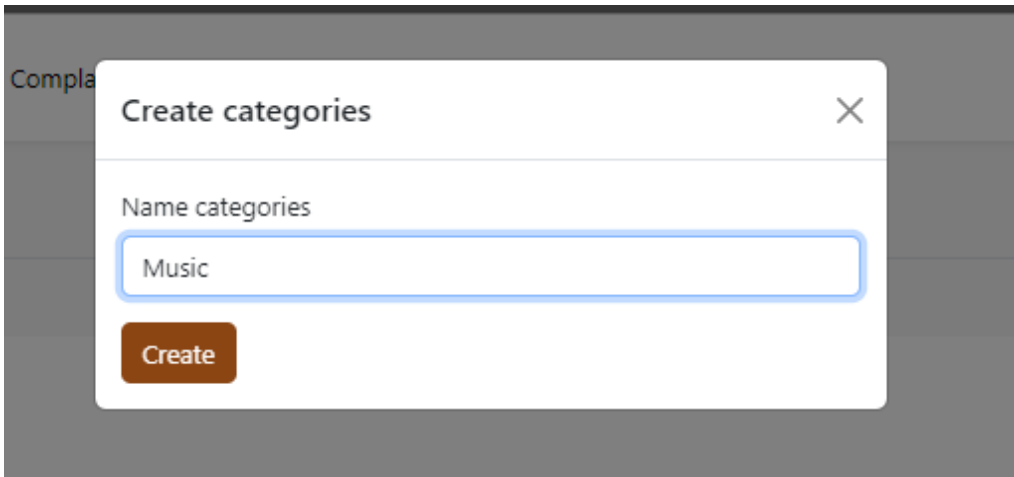


Рисунок 4.18 – Створення категорії для творчих проєктів

При переході на сторінку керування скаргами на творчі проєкти (рис. 4.19) менеджер переглядає кожну скаргу на творчий проєкт та відмічає її як опрацьовану. Також, менеджеру доступний перехід на сторінку керування скаргами на коментарі (рис. 4.20), де він також переглядає кожну скаргу на коментар під проєктом та відзначає її як опрацьовану.

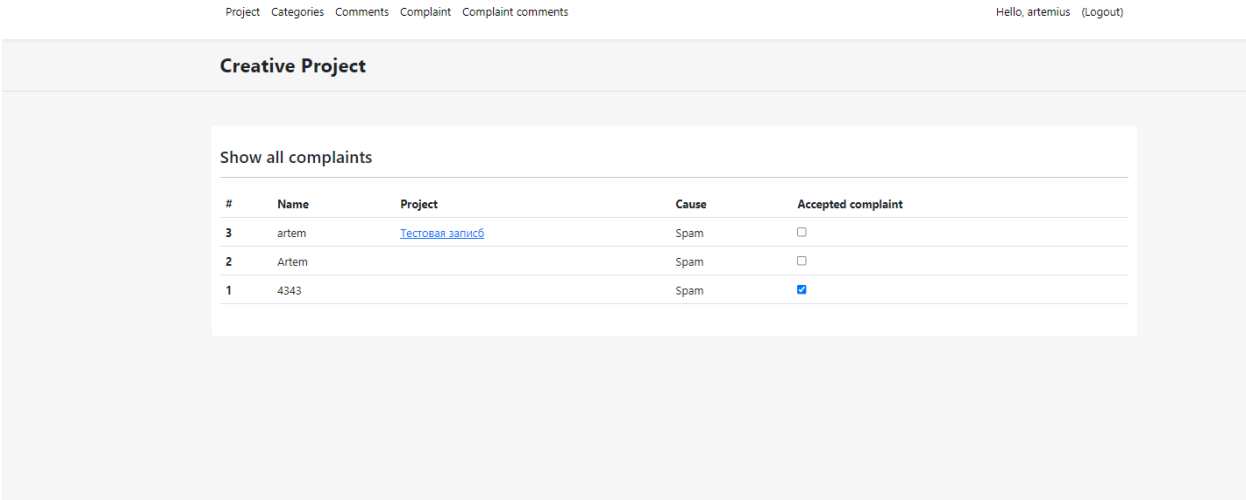


Рисунок 4.19 – Сторінка керування скаргами на творчі проєкти

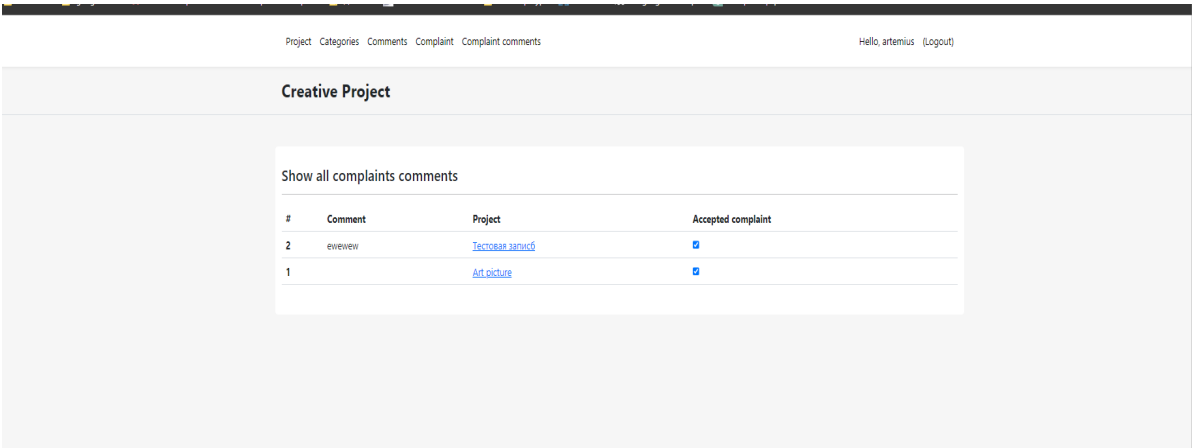


Рисунок 4.20 – Сторінка керування скаргами на коментарі

4.4 Тестування програми

Результати виконаного тестування за перевіркою функціональних вимог до програмного забезпечення представлено в табл. 4.1.

Таблиця 4.1 – Чеклист тестування програмного забезпечення

Перевірка	Результат
1	2
Авторизація користувача	Пройдено
Реєстрація користувача	Пройдено
Створення творчого проєкту	Пройдено
Редагування творчого проєкту	Пройдено
Видалення творчого проєкту	Пройдено
Створення ексклюзивних матеріалів	Пройдено
Редагування ексклюзивних матеріалів	Пройдено
Перегляд діаграми пожертвувань на власні творчі проєкти за кожний місяць	Пройдено
Перегляд творчих проєктів у певній категорії	Пройдено
Видалення ексклюзивних матеріалів	Пройдено
Додавання коментаря під творчим проєктом	Пройдено
Подання скарги на творчий проєкт	Пройдено
Подання скарги на коментар	Пройдено
Пожертвування для певного творчого проєкту через платіжну систему PayPal	Пройдено
Доступ до ексклюзивних матеріалів на певний	Пройдено
Перегляд діаграми власних пожертвувань на певні творчі проєкти за кожний місяць	Пройдено
Модерація творчих проєктів	Пройдено
Створення категорії для творчих проєктів	Пройдено

Кінець таблиці 4.1

1	2
Редагування категорій для творчих проєктів	Пройдено
Видалення категорії для творчих проєктів	Пройдено
Групування проєктів за категоріями	Пройдено
Визначення періоду збору коштів, протягом якого користувачі можуть перераховувати кошти на фінансування	Пройдено
Опрацювання скарг на творчі проєкти	Пройдено
Опрацювання скарг на коментарі	Пройдено

4.5 Висновки за розділом 4

В даному розділі було перевірено розроблене програмне забезпечення, а також було продемонстровано основні функції програми з підтвердженням реалізації під час тестування програми.

ВИСНОВОК

Під час виконання дипломної кваліфікаційної роботи було розроблено програмне забезпечення громадського фінансування творчих проєктів. Було виконано огляд існуючих методів вирішення завдання та аналогів, проаналізовано переваги та недоліки і розроблено функціональність власного програмного продукту.

Для реалізації програмного забезпечення було використано мову програмування PHP, вебфреймворк Laravel, шаблон проєктування MVC та систему керування базами даних MySQL. Для розробки інтерфейсної частини було використано Bootstrap, а також мову розмітки HTML та каскадну таблицю стилів (CSS). У програмі було реалізовано інтеграцію з API платіжної системи PayPal для зручної відправки грошових пожертвувань для будь-яких творчих проєктів.

Створений застосунок дозволив реалізувати такі можливості:

- створення творчих проєктів;
- відправлення пожертвування для творчих проєктів через платіжну систему PayPal;
- додавання фотографій кожного учасника творчого проєкту;
- надання безкоштовного доступу до ексклюзивних матеріалів для користувача на певний термін при умові, що він зробив певний відсоток пожертвування;
- перегляд статистики щодо отримання пожертвувань по кожному творчому проєкту;
- додавання власних коментарів на кожний творчий проєкт;
- модерація творчих проєктів та коментарів.

Після реалізації програми було проведено її тестування, що підтвердило працездатність розробленої програмної системи.

ПЕРЕЛІКДЖЕРЕЛ ПОСИЛАННЯ

1. About Us [Electronic resource]. – Access mode : <https://www.kickstarter.com/about?ref=global-footer>.
2. Краудфандинг: що це таке і як він працює в Україні [Електрон. ресурс]. – Режим доступу : <https://buduysvove.com/publications/kraudfandyng-shcho-ce-take-i-yak-vin-pracyuye-v-ukrayini>.
3. 7 платформ для збору коштів на розвиток вашої організації [Електрон. ресурс]. – Режим доступу : <https://www.gurt.org.ua/news/recent/31173/>.
4. What is Kickstarter? [Electronic resource]. – Access mode : <https://help.kickstarter.com/hc/en-us/articles/115004996453-What-is-Kickstarter->.
5. 14 Pros and Cons of Kickstarter [Electronic resource]. – Access mode : <https://brandongaille.com/14-pros-and-cons-of-kickstarter/>.
6. Що таке Patreon, або Як монетизувати свій блог чи сайт [Електрон. ресурс]. – Режим доступу : <https://osvitanova.com.ua/posts/4312-shcho-take-patreon-abo-yak-monetyzuvaty-svii-bloh-chy-sait>.
7. Pros And Cons Of Patreon [Electronic resource]. – Access mode : <https://www.ablison.com/pros-and-cons-of-patreon/>.
8. 6 Patreon Alternatives for Your Startup [Electronic resource]. – Access mode : <https://www.sumup.com/en-us/business-guide/6-patreon-alternatives-for-your-business/>.
9. What's IndieGoGo? [Electronic resource]. – Access mode : <https://www.doublejump.media/what-is-indiegogo-and-how-does-it-work/>.
10. 7 Key Indiegogo Pros and Cons for Beginners [Electronic resource]. – Access mode : <https://www.crowdcruх.com/indiegogo-pros-and-cons-for-beginners/>.
11. Kickstarter Alternatives: Funding and Validation of Products[Electronic resource]. – Access mode : <https://prelaunch.com/blog/kickstarter-alternatives>.
12. PHP: що це за мова програмування (простими словами),

особливості, застосування [Електрон. ресурс]. – Режим доступу : <https://kh-news.net/lang/uk/php-what-is-this-programming-language-in-simple-words-features-applications.html>.

13. Як стати PHP-розробником. План дій для початківців [Електрон. ресурс]. – Режим доступу : <https://dou.ua/lenta/articles/how-to-learn-php/>.

14. ЩО ТАКЕ PHP? [Електрон. ресурс]. – Режим доступу : <https://freehost.com.ua/ukr/faq/wiki/что-такое-php/>.

15. Vaswani, V. PHP A Beginner's Guide [Text] / V. Vaswani. – New York: The McGraw – Hill Companies, 2009. – 550 p.

16. Stauffer, M. Laravel [Text] : Up & Running / M. Stauffer. – Sebastopol : O'Reilly Media Inc, 2019. – 552 p.

17. The Mysql Manual [Electronic resource]. – Access mode : <https://www.mysql.com/>.

ДОДАТОК А
Технічне завдання

Вступ

В наш час творчі проєкти вважаються головною умовою підтримки життєдіяльності людства. Однак, деякі творці не отримують повного фінансування для подальшого розвитку своїх творчих ідей.

На сьогодні існують певні платформи для залучення фінансування від громадськості для реалізації власних ідей.

A.1 Підстави для розробки

Дипломна кваліфікаційна робота виконується за темою «Програмне забезпечення громадського фінансування творчих проєктів» у відповідності з наказом № 87 від 4 квітня 2023 р. за Національним університетом «Запорізька політехніка».

A.2 Призначення розробки

Програма розроблена для збору коштів на підтримку різних культурних та мистецьких проєктів. Це може значно зменшити витрати на маркетинг та просування проєктів та залучити більше вкладників.

A.3 Основні вимоги до програми

A.3.1 Вимоги до функціональних характеристик

Створений застосунок повинен у підсумку дозволити реалізувати такі можливості:

- створення творчих проєктів;
- відправлення пожертвування для творчих проєктів через платіжну систему PayPal;
- додавання фотографій кожного учасника творчого проєкту;

- надання безкоштовного доступу до ексклюзивних матеріалів для користувача на певний термін при умові, якщо він зробив певний відсоток пожертвування;
- перегляд статистики отримання пожертвувань по кожному творчому проєкту;
- додавання власних коментарів на кожний творчий проєкт;
- модерація творчих проєктів та коментарів.

A.3.2 Умови експлуатації

Для даного програмного забезпечення потрібно придбати вебсервер, а також сервер бази даних для постійного доступу користувачів до застосунку у будь-який час.

A.3.3 Вимоги до складу та параметрів технічних засобів

До вебсервера, на якому виконується програмне забезпечення, висуваються наступні апаратні вимоги:

- 32-розрядний (x86) або 64-розрядний (x64) процесор 1 ГГц;
- 1 ГБ (для 32-розрядної системи) або 2 ГБ (для 64-розрядної системи) оперативної пам'яті;
- операційна система CentOS або Linux Ubuntu;
- 16 ГБ (для 32-розрядної системи) або 20 ГБ (для 64 розрядної системи) на жорсткому диску.

A.4 Порядок контролю та приймання

Для виконання дипломної кваліфікаційної роботи критичним є виконання визначеного завдання, що включає безпосередньо завдання на роботу і технічне завдання, за темою, яка зафіксована наказом, у строки, які

визначені завданням на роботу.

У підсумку виконується захист роботи за оприлюдненим наперед графіком перед екзаменаційною комісією.

ДОДАТОК Б
Текст програми

Б.1 Текст файла CommentsProject.php

```
<?php

namespace App;

use Illuminate\Database\Eloquent\Model;
use App\Project;
class CommentsProject extends Model
{
    protected $table = 'commentsproject';
    protected $fillable = [
        'user_id',
        'project_id',
        'body',
        'created_at',
    ];

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

    public function project()
    {
        return $this->belongsTo(Project::class,'project_id');
    }
}
```

Б.2 Текст файла Complaint.php

```
<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Complaint extends Model
{
    protected $table = 'complaint';
    protected $fillable = [
        'name',
```

```

'body',
'type',
'done',
'project_id',
'created_at',
];
public function project()
{
return $this->belongsTo(Project::class,'project_id');
}
}

```

Б.3 Текст файла ComplaintComment.php

```

<?php

namespace App;

use Illuminate\Database\Eloquent\Model;
use App\Project;
use App\CommentsProject;
class ComplaintComment extends Model
{
protected $table = 'complaintcomments';
protected $fillable = [
'project_id',
'comment',
'done',
];

public function project()
{
return $this->belongsTo(Project::class,'project_id');
}
public function comments()
{
return $this->belongsTo(CommentsProject::class,'comment');
}
}

```

Б.4 Текст файла Donation.php

```
<?php

namespace App;

use Illuminate\Database\Eloquent\Model;
use App\Project;
use App\User;
class Donation extends Model
{
    protected $table = 'donations';
    protected $fillable = [
        'amount',
        'user_id',
        'paypal_payment_id',
        'check',
        'project_id',
        'created_at',
    ];

    public function project()
    {
        return $this->belongsTo(Project::class,'project_id');
    }

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

}
```

Б.5 Текст файла ExclusiveMaterial.php

```
<?php

namespace App;

use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Cviebrock\EloquentSluggable\Sluggable;
```

```

use Illuminate\Database\Eloquent\SoftDeletes;

class ExclusiveMaterial extends Model
{
    use Sluggable;
    protected $table = 'exclusive_material';

    protected $fillable = [
        'title',
        'body',
        'slug',
        'img',
        'user_id',
        'created_at',
        'project_id',
    ];

    public function user()
    {
        return $this->belongsTo(User::class);
    }
    public function project()
    {
        return $this->belongsTo(Project::class,'project_id');
    }
    public function sluggable():array
    {
        return [
            'slug' => [
                'source' => 'title',
                'onUpdate' => true
            ]
        ];
    }
}

```

Б.6 Текст файла PhotoMember.php

```

<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

```

```

class PhotoMember extends Model
{
protected $table = 'photo_members';
protected $fillable = [
'project_id',
'name',
'user_id',
'img',
];

public function project()
{
return $this->belongsTo(Project::class,'project_id');
}
}

```

Б.7 Текст файла Project.php

```

<?php

namespace App;

use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Cviebrock\EloquentSluggable\Sluggable;
use Illuminate\Database\Eloquent\SoftDeletes;
use App\Donation;
class Project extends Model
{

use Sluggable;
protected $fillable = [
'categories_id',
'title',
'img',
'slug',
'annotation',
'amount',
'user_id',
'body',
'video',
'started_at',
'started_end',

```

```

'website',
'instagram',
'twitter',
'facebook',
'telegram',
];

public function category()
{
return $this->belongsTo (ProjectCategories::class,'categories_id');
}
public function comments()
{
return $this->belongsTo(CommentsProject::class);
}

public function sluggable():array
{
return [
'slug' => [
'source' => 'title',
'onUpdate' => true
]
];
}
public function donations()
{
return $this->hasMany(Donation::class);
}
}

```

Б.8 Текст файла ProjectCategories.php

```

<?php

namespace App;
use Illuminate\Database\Eloquent\Model;

class ProjectCategories extends Model
{
//name_categories
protected $fillable = [
'name_categories', ];
}

```

```

public function project()
{
    return $this->hasMany(Project::class);
}
}

```

Б.9 Текст файла ProjectCategories.php

```

<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class ProjectCategories extends Model
{
    //name_categories
    protected $fillable = [
        'name_categories',
    ];

    public function project()
    {
        return $this->hasMany(Project::class);
    }
}

```

Б.10 Текст файла User.php

```

<?php

namespace App;

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

class User extends Authenticatable
{
    use Notifiable;

    const ROLE_USER = 0;

```

```

const ROLE_CREATOR =1;
const ROLE_MANAGER=2;
/**
 * The attributes that are mass assignable.
 *
 * @var array
 */
protected $fillable = [
    'name', 'email', 'password','role',
];

public static function getRoles() {
    return [
        self::ROLE_CREATOR =>"Creator",
        self::ROLE_USER =>"User",
        self::ROLE_MANAGER =>"Manager",
    ];
}

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

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

```

Б.11 Текст файла CommentsProjectController.php

```

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Project;
use App\CommentsProject;
class CommentsProjectController extends Controller
{
public function store(Request $request)
{
    $comment_t = new CommentsProject;
    $comment_t->user_id = $request->input('user_id');
    $comment_t->project_id = $request->input('project_id');
    $comment_t->body = $request->input('body');
    $comment_t->save();
    return back()->with('success', 'Your form has been submitted.');
```

```

    }

    public function delete($id) {
        $delete = CommentsProject::find($id);
        $delete->delete();
        return back()->with('success', 'You have successfully deleted the categories.');
```

```

    }

}

```

Б.12 Текст файла ComplaintCommentController.php

```

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Comments;
use App\ComplaintComment;
class ComplaintCommentController extends Controller
{
public function store(Request $request)
{

```

```

$complaint = new ComplaintComment;
$complaint->project_id = $request->input('project_id');
$complaint->comment = $request->input('comment');
$complaint->done = 0;
$complaint->save();
return back()->with('success_complaint', 'You have successfully submitted a
complaint about the project, and the administrator will contact you within a day');
}

public function complaint_done(Request $request)
{
    $id = $request->input('id');
    $complaint_d = ComplaintComment::find($id);
    $complaint_d->done = $request->input('checked');
    $complaint_d->save();
    return back();
}
}

```

Б.13 Текст файла ComplaintProjectController.php

```

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Complaint;
class ComplaintProjectController extends Controller
{
    public function store(Request $request)
    {
        $complaint = new Complaint;
        $complaint->project_id = $request->input('project_id');
        $complaint->name = $request->input('name');
        $complaint->type = $request->input('type');
        $complaint->body = $request->input('body');
        $complaint->done = 0;
        $complaint->save();
        return back()->with('success_complaint', 'You have successfully submitted a
complaint about the project, and the administrator will contact you within a day');
    }
    public function complaint_done(Request $request)
    {

```

```

$Id = $request->input('id');
$complaint_d = Complaint::find($Id);
$complaint_d->done = $request->input('checked');
$complaint_d->save();
return back();
}
}

```

Б.14 Текст файла DonationController.php

```

<?php

namespace App\Http\Controllers;
use Illuminate\Support\Facades\Redirect;

use Illuminate\Http\Request;
use App\Donation;
use App\Project;
use PayPal\Rest\ApiContext;
use PayPal\Auth\OAuthTokenCredential;
use PayPal\Api\Agreement;
use PayPal\Api\Payer;
use PayPal\Api\Plan;
use PayPal\Api\PaymentDefinition;
use PayPal\Api\PayerInfo;
use PayPal\Api\Item;
use PayPal\Api\ItemList;
use PayPal\Api\Amount;
use PayPal\Api\Transaction;
use PayPal\Api\RedirectUrls;
use PayPal\Api\Payment;
use PayPal\Api\PaymentExecution;
use Carbon\Carbon;

use URL;
use Session;

class DonationController extends Controller
{
    public function __construct()
    {

        $paypal_configuration = \Config::get('paypal');
    }
}

```

```

$this->_api_context = new ApiContext(new
OAuthTokenCredential($paypal_configuration['client_id'],
$paypal_configuration['secret']));
$this->_api_context->setConfig($paypal_configuration['settings']);
}
public function add_donation(Request $request) {

    $project_name = Project::find($request->input('project_id'));
    $amountToBePaid = $request->input('amount');
    $payer = new Payer();
    $payer->setPaymentMethod('paypal');

    $item_1 = new Item();
    $item_1->setName('Donation project')
->setCurrency('USD')
->setQuantity(1)
->setPrice($amountToBePaid);

    $item_list = new ItemList();
    $item_list->setItems(array($item_1));

    $amount = new Amount();
    $amount->setCurrency('USD')
->setTotal($amountToBePaid);

    $redirect_urls = new RedirectUrls();
    $redirect_urls->setReturnUrl(URL::route('status'))
->setCancelUrl(URL::route('status'));

    $transaction = new Transaction();
    $transaction->setAmount($amount)
->setItemList($item_list)
->setDescription('Donation project ');

    $payment = new Payment();
    $payment->setIntent('Sale')
->setPayer($payer)
->setRedirectUrls($redirect_urls)
->setTransactions(array($transaction));
    try {
        $payment->create($this->_api_context);
    } catch (\PayPal\Exception\PPConnectionException $ex) {
        if (\Config::get('app.debug')) {
            \Session::put('error', 'Connection timeout');
        }
        return response()->json(['check' => 1]);
    }
}

```

```

    } else {
    return response()->json(['check' => 2]);
    }
    }

    foreach ($payment->getLinks() as $link) {
    if ($link->getRel() == 'approval_url') {
    $redirect_url = $link->getHref();
    break;
    }
    }
    session()->put('paypal_payment_id', $payment->getId());
    if (isset($redirect_url)) {
    //return Redirect::away($redirect_url);
    $newDonation = Donation::create($request->all());
    $newDonationId = $newDonation->id;
    session()->put('newDonationId', $newDonationId);
    return response()->json(['check' => 0, 'redirect_url'=>$redirect_url]);
    }

    return response()->json(['check' => 3]);
    //return response()->json(['check' => 1]);
    }

    public function getPaymentStatus(Request $request)
    {
    $payment_id = Session::get('paypal_payment_id');
    $newDonation = Session::get('newDonationId');

    Session::forget('paypal_payment_id');
    Session::forget('newDonation');
    if (empty($request->PayerID) || empty($request->token)) {
    //dd($payment_id);
    }
    $payment = Payment::get($payment_id, $this->_api_context);
    $execution = new PaymentExecution();
    $execution->setPayerId($request->PayerID);

    $result = $payment->execute($execution, $this->_api_context);

    if ($result->getState() == 'approved') {

    $donations = Donation::findOrFail($newDonation);
    $donations->paypal_payment_id = $payment_id;
    $donations->check = 1;
    $donations->save();

```

```

    }
    return redirect('/my_donates');
  }
  public function show_donated() {
    $donation = Donation::where('user_id', auth()->user()->id)->where('check',
1)->get();
    $donationByMonth = $donation->groupBy(function($donations) {
    return Carbon::parse($donations->created_at)->format('m');
    });
    $totalsByMonth = [];
    foreach ($donationByMonth as $month => $donation_s) {
    $total = $donation_s->sum('amount');
    $monthName = Carbon::createFromDate(null, $month, null)->format('F');

    $totalsByMonth[$monthName] = $total;
    }
    return view('my_donation',
['donations'=>$donation,'totalsByMonth'=>$totalsByMonth]);
  }
  public function history_donated() {
    $projects_all = Project::where('user_id', auth()->user()->id)->get();
    $myproject_id = [];
    foreach ($projects_all as $project_all) {
    $myproject_id[]=$project_all->id;
    }
    $donation_my = Donation::WhereIn('project_id', $myproject_id)-
>where('check', 1)->get();
    $donationByMonth = $donation_my->groupBy(function($donations) {
    return Carbon::parse($donations->created_at)->format('m');
    });

    $totalsByMonth = [];
    foreach ($donationByMonth as $month => $donation_s) {
    $total = $donation_s->sum('amount');
    $monthName = Carbon::createFromDate(null, $month, null)->format('F');
    $totalsByMonth[$monthName] = $total;
    }
    return view('history_donation',
['donation_my'=>$donation_my,'totalsByMonth'=>$totalsByMonth]);
  }
}

```

Б.15 Текст файла ExclusiveMaterialController.php

```

<?php

namespace App\Http\Controllers;
use Illuminate\Http\Request;
use App\ExclusiveMaterial;
use App\Donation;
use Carbon\Carbon;
use App\Project;
class ExclusiveMaterialController extends Controller
{
    public function index() {
        $exclusivematerial = ExclusiveMaterial::where('user_id', auth()->user()->id)-
>get();
        $projects = Project::where('user_id', auth()->user()->id)->get();
        return view("exclusive_list",
['exclusivematerial'=>$exclusivematerial,'projects'=>$projects]);
    }

    public function add(Request $request) {
        $this->validate($request, [
            'title' =>'required|min:5',
            'project_id' =>'required',
            'title'=>'required',
            'body'=>'required|min:10',
            'image'=>'required',
        ]);

        $exclusive = new ExclusiveMaterial;
        $exclusive->title = $request->input('title');
        $exclusive->project_id = $request->input('project_id');
        $exclusive->user_id = auth()->user()->id;
        $exclusive->body = $request->input('body');

        if ($request->hasFile('image')) {
            $img = $request->file('image');
            $filename = time() . '_' . $img->getClientOriginalName();
            $img->move(public_path('images/exclusivematerial'), $filename);
            $exclusive->img = $filename;
        }

        $exclusive->save();
        return back()->with('success', 'Your form has been submitted.');
```

```

    }
    public function destroy($id)
    {
        $exclusiveMaterial = ExclusiveMaterial::find($id);
        $exclusiveMaterial->delete();
        return back()->with('success', 'You have successfully deleted the project.');
```



```

    }
    public function exclusive_edit($id) {
        $exclusivematerial = ExclusiveMaterial::where('user_id', auth()->user()->id)-
>where('id',$id)->get();
        // $projects = Project::where('user_id', auth()->user()->id)->get();
        return view("exclusive_edit",['exclusivematerial'=>$exclusivematerial]);
    }

    public function update(Request $request,$id) {
        $this->validate($request, [
            'title' =>'required|min:5',
            'body'=>'required|min:10',
        ]);

        $exclusivematerial = ExclusiveMaterial::find($id);
        if ($request->hasFile('img')) {
            $img = $request->file('img');
            $filename = time() . '_' . $img->getClientOriginalName();
            $img->move(public_path('images/exclusivematerial'), $filename);
            $exclusivematerial->img = $filename;
        }
        $exclusivematerial->title = $request->input('title');
        $exclusivematerial->body = $request->input('body');
        $exclusivematerial->save();
        return back()->with('success', 'Your form has been submitted.');
```



```

    }
    public function show($slug) {
        $project= Project::where('slug', $slug)->firstOrFail();
        $exclusivematerial = ExclusiveMaterial::where('project_id', $project->id)-
>orderBy('id', 'desc')->get();
        $donation_total = Donation::where('project_id',$project->id)-
>where('user_id',auth()->user()->id)->where('check',1)->sum('amount');
        $donation = Donation::where('project_id',$project->id)-
>where('user_id',auth()->user()->id)->where('check',1)->first();

        if(!$donation) {
            return view('conditionexclusive');
        }
    }

```

```

$percent = ($donation_total / $project->amount) * 100;
$startdonation = Carbon::parse($donation->created_at);
$todayDate = Carbon::now()->toDateString();
$daysSinceDonation = $startdonation->diffInDays(Carbon::now());
if(intval($percent)>=1) {
if($daysSinceDonation<=10) {
return view('show_exclusive',['exclusivematerial'=>$exclusivematerial]);
} else {
return view('conditionexclusive');
}
} else if(intval($percent)>=10) {
if($daysSinceDonation<=100) {
return view('show_exclusive',['exclusivematerial'=>$exclusivematerial]);
} else {
return view('conditionexclusive');
}
} else if(intval($percent)<=50) {
return view('show_exclusive',['exclusivematerial'=>$exclusivematerial]);
} else {
return view('conditionexclusive');
}
}
}
}

```

Б.16 Текст файла ManagerController.php

```
<?php
```

```

namespace App\Http\Controllers;
use Illuminate\Http\Request;
use App\Project;
use App\ProjectCategories;
use Carbon\Carbon;
use App\Complaint;
use App\CommentsProject;
use App\ComplaintComment;
class ManagerController extends Controller
{
public function project_manager()
{
$project = Project::all();
$mytime = Carbon::now();

```

```

        $mytime->toDateTimeString();
        return view('manager.project_manager',
['projects'=>$project,'mytime'=>$mytime]);
    }
    public function showProject($id) {
        $project = Project::where('id',$id)->get();
        $category = ProjectCategories::all();
    return
    view('manager.projectmanager_edit', ['project'=>$project,'categories'=>$category]);
    }
    public function update_project($id, Request $request) {
        $this->validate($request, [
            'title' =>'required|min:5',
            'categories_id' =>'required',
            'annotation'=>'required|min:10',
            'amount'=>'required',

            'body' =>'required|min:10',
            'started_at' =>'required',
            'started_end' =>'required',
        ]);
    }

```

```

        $project = Project::find($id);
        if ($request->hasFile('img')) {
            $img = $request->file('img');
            $filename = time() . '_' . $img->getClientOriginalName();
            $img->move(public_path('images/project'), $filename);
            $project->img = $filename;
        }
        $project->website = $request->input('website');
        $project->facebook = $request->input('facebook');
        $project->telegram = $request->input('telegram');
        $project->instagram = $request->input('instagram');
        $project->twitter = $request->input('twitter');

        $project->title = $request->input('title');
        $project->categories_id = $request->input('categories_id');
        $project->annotation = $request->input('annotation');
        $project->body = $request->input('body');
        $project->amount = $request->input('amount');
        $project->started_at = $request->input('started_at');
        $project->started_end = $request->input('started_end');
        $project->save();
        return back()->with('success', 'Your form has been submitted.');
```

```

    }
    public function destroy_project($id)
    {
        $delete = Project::find($id);
        $delete->delete();
        $delete_donation = Donation::where('project_id',$id)->get();
        $delete_donation = Donation::where('project_id',$id)->get();
        if(count($delete_donation)>0) {
            $delete_donation->delete();
        }
        return back()->with('success', 'You have successfully deleted the project.');
```

```

    }
    public function categoriesmanager_show()
    {
        $category = ProjectCategories::all();
        return view('manager.categoriesmanager_show',['categories'=>$category]);
    }

    public function categoriesmanager_edit($id) {
        $project_edit = ProjectCategories::find($id);
        return view('manager.categoriesmanager_edit',
['project_edit'=>$project_edit]);
    }

    public function complaintmanager_show() {
        $complaint_show = Complaint::orderBy("id","DESC")->get();
        return view('manager.complaintmanager_show',
['complaints'=>$complaint_show]);
    }

    public function commentsmanager_show() {
        $commentsadmin_show = CommentsProject::orderBy("id","DESC")->get();
        return view('manager.commentsmanager_show',
['comments'=>$commentsadmin_show]);
    }

    public function categoriesmanager_create(Request $request) {
        $this->validate($request, [
            'name_categories' =>'required',
        ]);
        $category_project = new ProjectCategories();
        $category_project->name_categories = $request->input('name_categories');
        $category_project->save();
        return back()->with('success', 'Your form has been submitted.');
```

```

    }
    public function categoriesmanager_update($id,Request $request) {

```

```

$this->validate($request, [
    'name_categories' => 'required',
]);
$project_categories = ProjectCategories::find($id);
$project_categories->name_categories = $request->input('name_categories');
$project_categories->save();
return back()->with('success', 'You have successfully deleted the categories.');
```



```

}
public function destroy_categories($id) {
    $delete = ProjectCategories::find($id);
    $delete->delete();
    return back()->with('success', 'You have successfully deleted the categories.');
```



```

}
public function complaintcommentsmanager_show() {
    $complaintcommentsmanager_show =
ComplaintComment::orderBy("id","DESC")->get();
    return view('manager.complaintcomments_show',
['complaint_comments'=>$complaintcommentsmanager_show]);
}
}

```

Б.17 Текст файла PhotoMemberController.php

```

<?php

namespace App\Http\Controllers;
use Illuminate\Http\Request;
use App\PhotoMember;
use App\Project;
class PhotoMemberController extends Controller
{
    public function index()
    {
        $photo_member = PhotoMember::where('user_id', auth()->user()->id)->get();
        $project = Project::where('user_id', auth()->user()->id)->get();
return view('project_members', ['photo_members'=>$photo_member,'projects'=>$project]);
    }
    public function AddMemberForm(Request $request) {
        $this->validate($request, [
            'name' => 'required',
        ]);
    }
}

```

```

$project_t = new PhotoMember;
$project_t->name = $request->input('name');
$project_t->project_id = $request->input('project_id');
$project_t->user_id = $request->input('user_id');

if ($request->hasFile('img')) {
    $img = $request->file('img');
    $filename = time() . '_' . $img->getClientOriginalName();
    $img->move(public_path('images/members'), $filename);
    $project_t->img = $filename;
}
$project_t->save();
return back()->with('success', 'Your form has been submitted.');
```

```

}
public function delete($id) {
    $delete = PhotoMember::find($id);
    $delete->delete();
    return back()->with('success', 'You have successfully deleted the categories.');
```

```

}
public function show($id) {
    $photo_members = PhotoMember::where('project_id', $id)->get();
    return
view('show_project_members', ['photo_members'=>$photo_members]);
}
}

```

Б.18 Текст файла ProjectController.php

```

<?php
namespace App\Http\Controllers;
use Carbon\Carbon;
use Illuminate\Http\Request;
use App\Project;
use App\ProjectCategories;
use App\Donation;
use App\CommentsProject;
class ProjectController extends Controller
{
    public function index()
    {
        $project = Project::where('user_id', auth()->user()->id)->get();
        $category = ProjectCategories::all();
        $mytime = Carbon::now();
    }
}

```

```

$mytime->toDateTimeString();
return view('project',['projects'=>$project,'categories'=>$category,
'mytime'=>$mytime]);
}

```

```

public function ProjectForm(Request $request) {
    $this->validate($request, [
        'title' =>'required',
        'categories_id' =>'required',
        'annotation'=>'required',
        'amount'=>'required',
        'image'=>'required',
        'body' =>'required',
        'started_at' =>'required',
        'started_end' =>'required',
    ]);
    $project_t = new Project;
    $project_t->title = $request->input('title');
    $project_t->user_id = $request->input('user_id');
    $project_t->categories_id = $request->input('categories_id');
    $project_t->annotation = $request->input('annotation');
    $project_t->body = $request->input('body');
    $project_t->amount = $request->input('amount');
    $project_t->started_at = $request->input('started_at');
    $project_t->started_end = $request->input('started_end');

    if ($request->hasFile('image')) {
        $img = $request->file('image');
        $filename = time() . '_' . $img->getClientOriginalName();
        $img->move(public_path('images/project'), $filename);
        $project_t->img = $filename;
    }
    if($request->file('video')) {
        $video = $request->file('video');
        $filename = time() . '_' . $video->getClientOriginalName();
        $video->move(public_path('video/project'), $filename);
        $project_t->video = $filename;
    }
    $project_t->save();
    return back()->with('success', 'Your form has been submitted.');
```

```

}

public function update(Request $request,$id) {
    $this->validate($request, [
        'title' =>'required|min:5',
        'categories_id' =>'required',

```

```

'annotation'=>'required|min:10',
'amount'=>'required',
'body'=>'required|min:10',
'started_at'=>'required',
'started_end'=>'required',
]);
$project = Project::find($id);
if ($request->hasFile('img')) {
$img = $request->file('img');
$filename = time() . '_' . $img->getClientOriginalName();
$img->move(public_path('images/project'), $filename);
$project->img = $filename;
}
if ($request->file('video')) {
$video = $request->file('video');
$filename = time() . '_' . $video->getClientOriginalName();
$video->move(public_path('video/project'), $filename);
$project->video = $filename;}
$project->title = $request->input('title');
$project->categories_id = $request->input('categories_id');
$project->annotation = $request->input('annotation');
$project->body = $request->input('body');
$project->amount = $request->input('amount');
$project->started_at = $request->input('started_at');
$project->started_end = $request->input('started_end');
$project->website = $request->input('website');
$project->facebook = $request->input('facebook');
$project->telegram = $request->input('telegram');
$project->instagram = $request->input('instagram');
$project->twitter = $request->input('twitter');
$project->save();
return back()->with('success', 'Your form has been submitted.');
```

```

}
public function categories_show($id) {
$projects = Project::where('categories_id', $id)->get();
$name_category = ProjectCategories::where('id', $id)->get();
$category = ProjectCategories::all();
return view('project_categories',
['projects_incategories'=>$projects,'categories'=>$category,'name_category'=>$name_category]);}
public function show_creativeproject($slug) {
$mytime = Carbon::now();
$mytime->toDateTimeString();
$project = Project::where('slug', $slug)->get();
$project_id = Project::where('slug', $slug)->firstOrFail();

```

```

$project_id = $project_id->id;
$donation_total = Donation::where('project_id',$project_id)->sum('amount');
$comment = CommentsProject::where('project_id',$project_id)->get();
if (auth()->user()) {
$donation = Donation::where('user_id', auth()->user()->id)
->where('project_id',$project_id)
->where('check',1)
->get();
return view('show_creativeproject',
['project'=>$project,'mytime'=>$mytime,'donation'=>$donation,'donation_total'=>$d
onation_total,'comments'=>$comment]);
} else {
return view('show_creativeproject'
['project'=>$project,'donation_total'=>$donation_total,'donation'=>"",'comments'=>$
comment]);
}}
public function allproject() {
$project = Project::orderBy('id', 'DESC')->get();
$category = ProjectCategories::all();
return view('project_all',['projects'=>$project,'categories'=>$category]);
}
public function showProject($id) {
$project = Project::where('id',$id)->where('user_id', auth()->user()->id)-
>get();
$category = ProjectCategories::all();
return view('project_edit',['project'=>$project,'categories'=>$category]);
}
public function destroy($id)
{
$delete_donation = Donation::where('project_id',$id)->first();
$delete_donation->delete();
$delete_project = Project::where('id',$id)->first();
$delete_project->delete();
return back()->with('success', 'You have successfully deleted the project.')}}

```

Б.19 Текст файла routers.php

```

<?php
use Illuminate\Support\Facades\Route;
use App\Http\Controllers\Project;
Auth::routes();
Route::get('/allproject', 'ProjectController@allproject');
Route::get('/categories/{id}', 'ProjectController@categories_show');

```

```

Route::get('/show_creativeproject/{slug}',
'ProjectController@show_creativeproject');
Route::post('/add_commentsproject', 'CommentsProjectController@store');
Route::post('/complaint_project', 'ComplaintProjectController@store');
Route::post('/complaint_comment', 'ComplaintCommentController@store');
Route::get('/show_project_members/{id}', 'PhotoMemberController@show');
Route::group(['middleware' => ['role_creator']], function () {
Route::get('/project', 'ProjectController@index');
Route::post('/project', 'ProjectController@ProjectForm')-
>name('validate.form');
Route::get('/project_edit/{id}', 'ProjectController@showProject');
Route::post('/project_edit/{id}', 'ProjectController@update');
Route::get('/project_delete/{id}', 'ProjectController@destroy')-
>name('project_delete.destroy');
Route::get('/history_donated', 'DonationController@history_donated');
Route::get('/project_members', 'PhotoMemberController@index');
Route::post('/project_members',
'PhotoMemberController@AddMemberForm');
Route::get('/delete_members/{id}', 'PhotoMemberController@delete');
Route::get('/exclusive_list', 'ExclusiveMaterialController@index');
Route::post('/exclusive_list', 'ExclusiveMaterialController@add');
Route::get('/exclusive_delete/{id}', 'ExclusiveMaterialController@destroy');
Route::get('/exclusive_edit/{id}',
'ExclusiveMaterialController@exclusive_edit');
Route::post('/exclusive_edit/{id}', 'ExclusiveMaterialController@update');
});
Route::group(['middleware' => ['role_user']], function () {
Route::get('/my_donates', 'DonationController@show_donated');
Route::post('/add_donation', 'DonationController@add_donation');
Route::get('/add_donation',['as' => 'status', 'uses'
=> 'DonationController@getPaymentStatus']);
Route::get('/show_exclusivematerials/{slug}',
'ExclusiveMaterialController@show');
});
Route::group(['middleware' => ['role_manager']], function () {
Route::get('/project_manager', 'ManagerController@project_manager');
Route::get('/projectmanager_edit/{id}', 'ManagerController@showProject');
Route::post('/projectmanager_edit/{id}',
'ManagerController@update_project');
Route::get('/projectmanager_delete/{id}',
'ManagerController@destroy_project');
Route::get('/categoriesmanager_show',
'ManagerController@categoriesmanager_show');
Route::post('/categoriesmanager_create',
'ManagerController@categoriesmanager_create');

```

```

        Route::get('/categoriesmanager_edit/{id}',
'ManagerController@categoriesmanager_edit');
        Route::post('/categoriesmanager_edit/{id}',
'ManagerController@categoriesmanager_update');
        Route::get('/categoriesmanager_delete/{id}',
'ManagerController@destroy_categories');
        Route::get('/complaintmanager_show',
'ManagerController@complaintmanager_show');
        Route::post('/complaint_done',
'ComplaintProjectController@complaint_done');
        Route::get('/commentsmanager_show',
'ManagerController@commentsmanager_show');
        Route::get('/delete_comment/{id}', 'CommentsProjectController@delete');
        Route::get('/complaint_comments',
'ManagerController@complaintcommentsmanager_show');
        Route::post('/complaintcomments_done',
'ComplaintCommentController@complaint_done');});
Route::get('/home', 'HomeController@index')->name('home');
Route::get('/', 'HomeController@index')->name('home');
Route::get('logout', function (){
auth()->logout();
Session()->flush();
return Redirect::to('/');
})->name('logout');
Auth::routes();

```

ДОДАТОК В
Слайди презентації

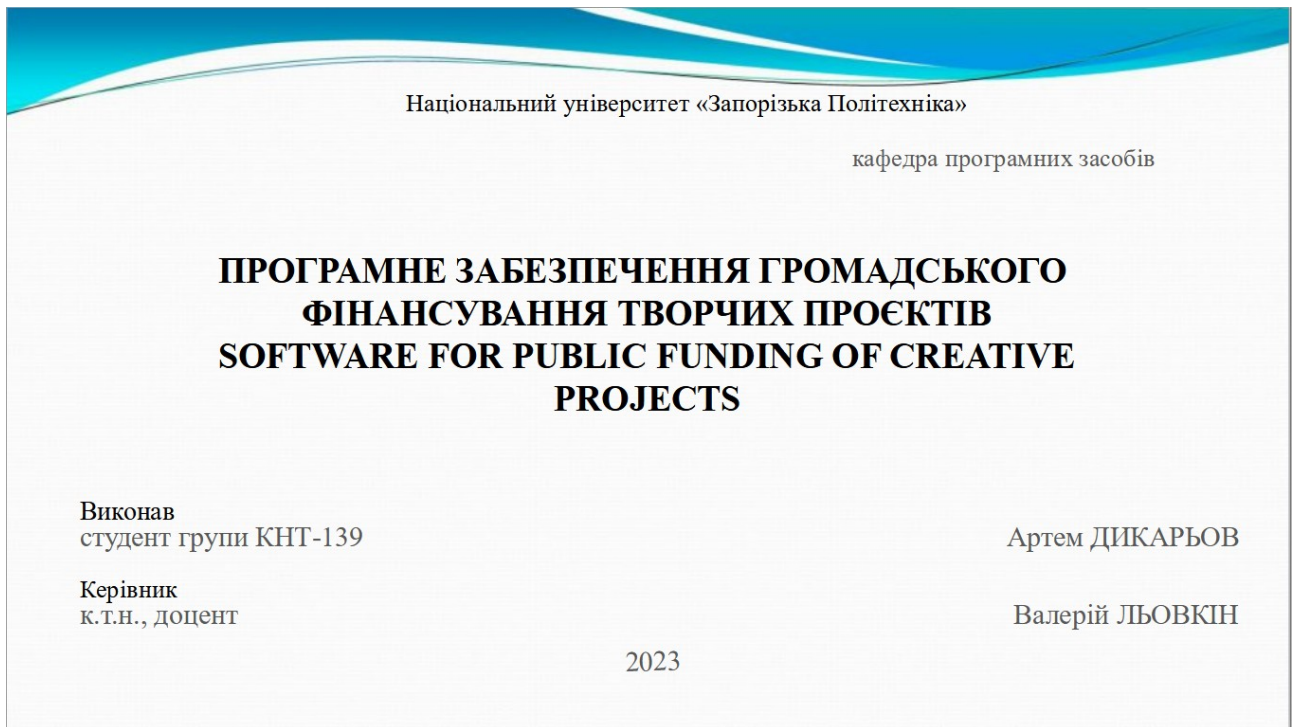


Рисунок В.1 – Слайд №1

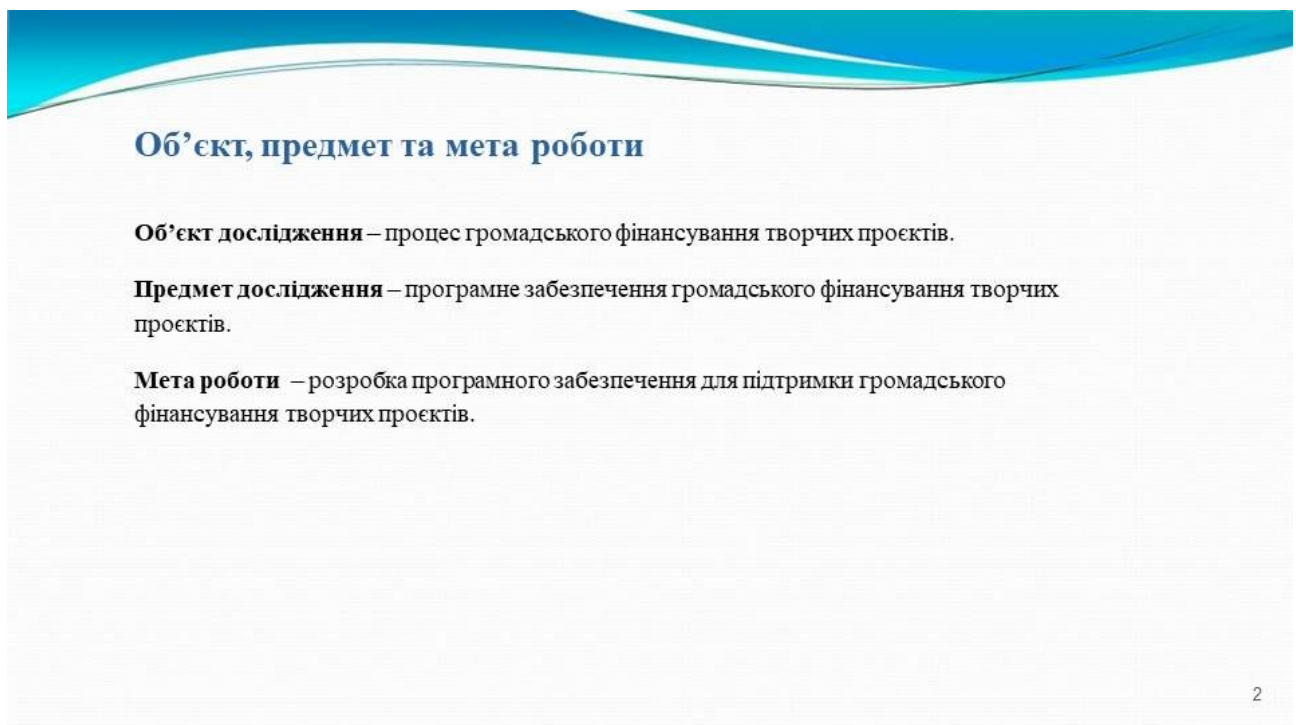


Рисунок В.2 – Слайд №2

Завданнями дипломної роботи

- Аналіз предметної області.
- Визначення вимог до програмного забезпечення громадського фінансування творчих проєктів.
- Проєктування програмного забезпечення громадського фінансування творчих проєктів.
- Розробка та тестування програмного забезпечення громадського фінансування творчих проєктів.

3

Рисунок В.3 – Слайд №3

Порівняння існуючих аналогів

Критерій оцінки	Аналоги			
	Kickstarter	Patreon	Indiegogo	Застосунок
Спілкування з аудиторією	+	+	-/+	+
Інтеграція з платіжними системами	+	+	+	+
Зручна аналітика по пожертвуванням	-	-	-	+
Швидка модерація проєктів	-/+	-	+	+
Простий інтерфейс для звичайного користувача	+	-	+	+
Ексклюзивний контент для спонсорів проєктів	-	+	+	+

4

Рисунок В.4 – Слайд №4

Порівняння мов програмування

	PHP	Python	C#
Кросплатформеність	+	+	+
Швидкість розробки програми	+	+	-
Швидкість роботи кінцевого продукту	+	-	+
Легкість вивчення	+	+	-
Зрозуміла документація	+	+	+/-

5

Рисунок В.5 – Слайд №5

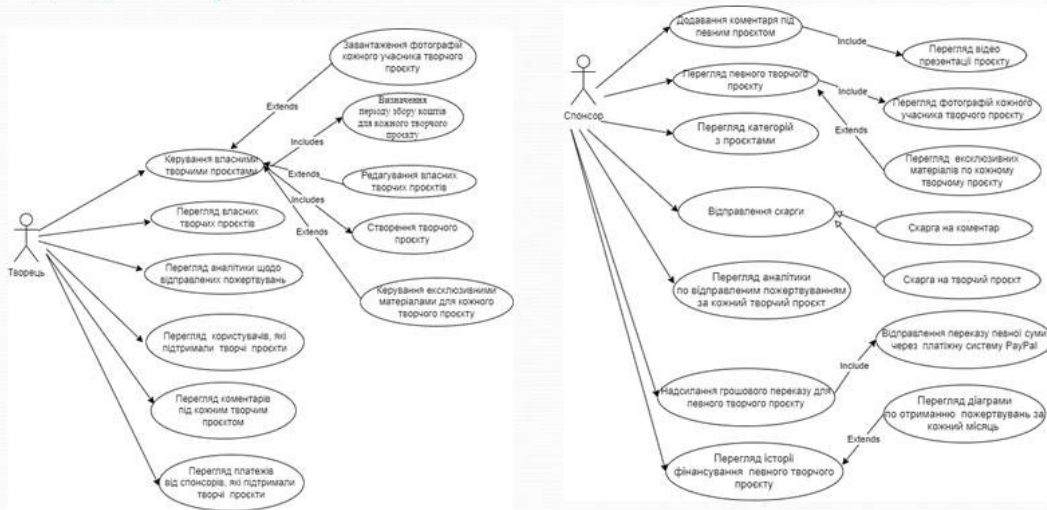
Вибір вебфреймворку розробки мовою PHP

	Laravel	Symfony	Yii2
Швидкість розробки	++	-	+
Безпека	+	+	+
Підтримка	+	+	-
Безпека	++	+	+
Продуктивність	+	+	+
Масштабованість	+	+	+

6

Рисунок В.6 – Слайд №6

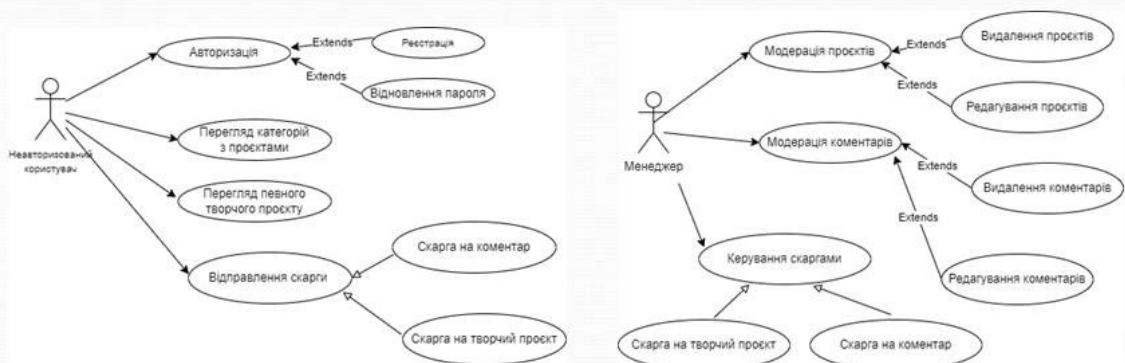
Діаграма прецедентів



7

Рисунок В.7 – Слайд №7

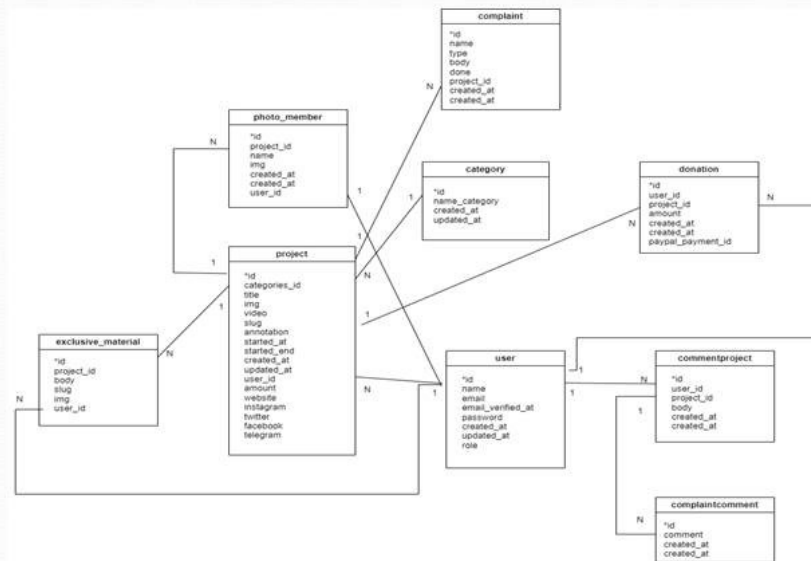
Діаграма прецедентів



8

Рисунок В.8 – Слайд №8

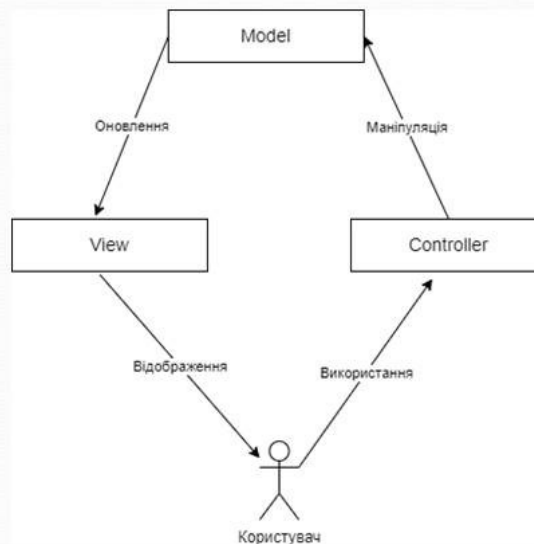
Схема структури бази даних



9

Рисунок В.9 – Слайд №9

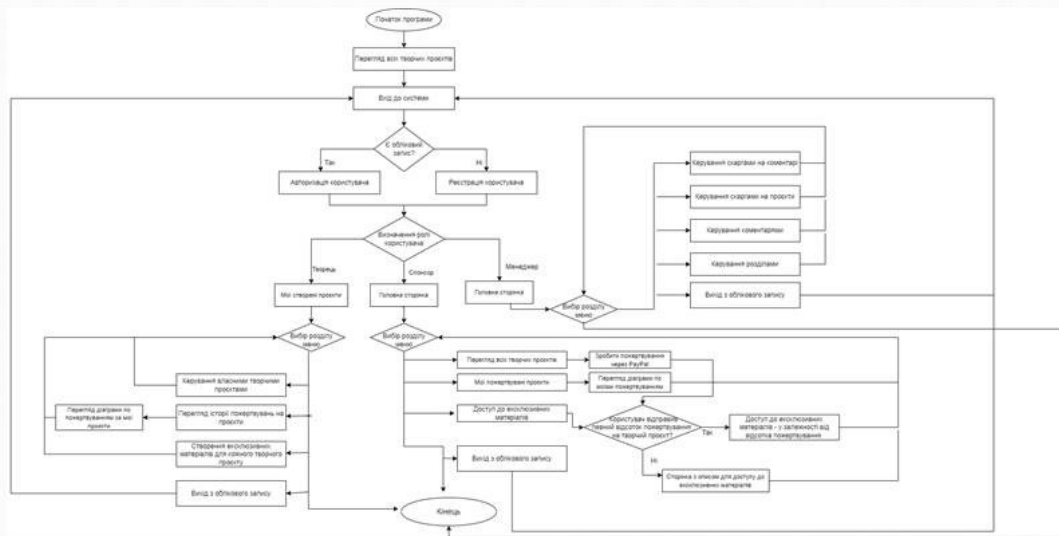
Структурна схема програмного проєкту



10

Рисунок В.10 – Слайд №10

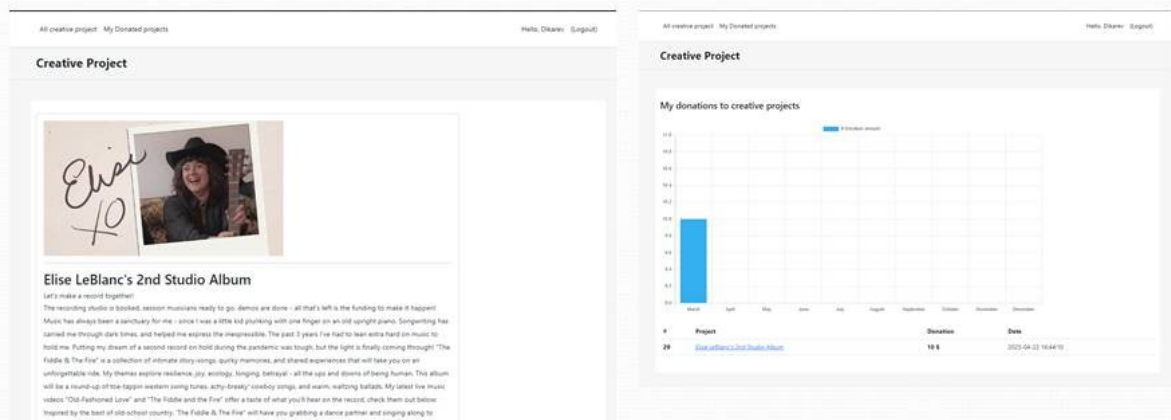
Алгоритм функціонування розробленої програми



11

Рисунок В.11 – Слайд №11

Інтерфейси програмного застосунку



12

Рисунок В.12 – Слайд №12

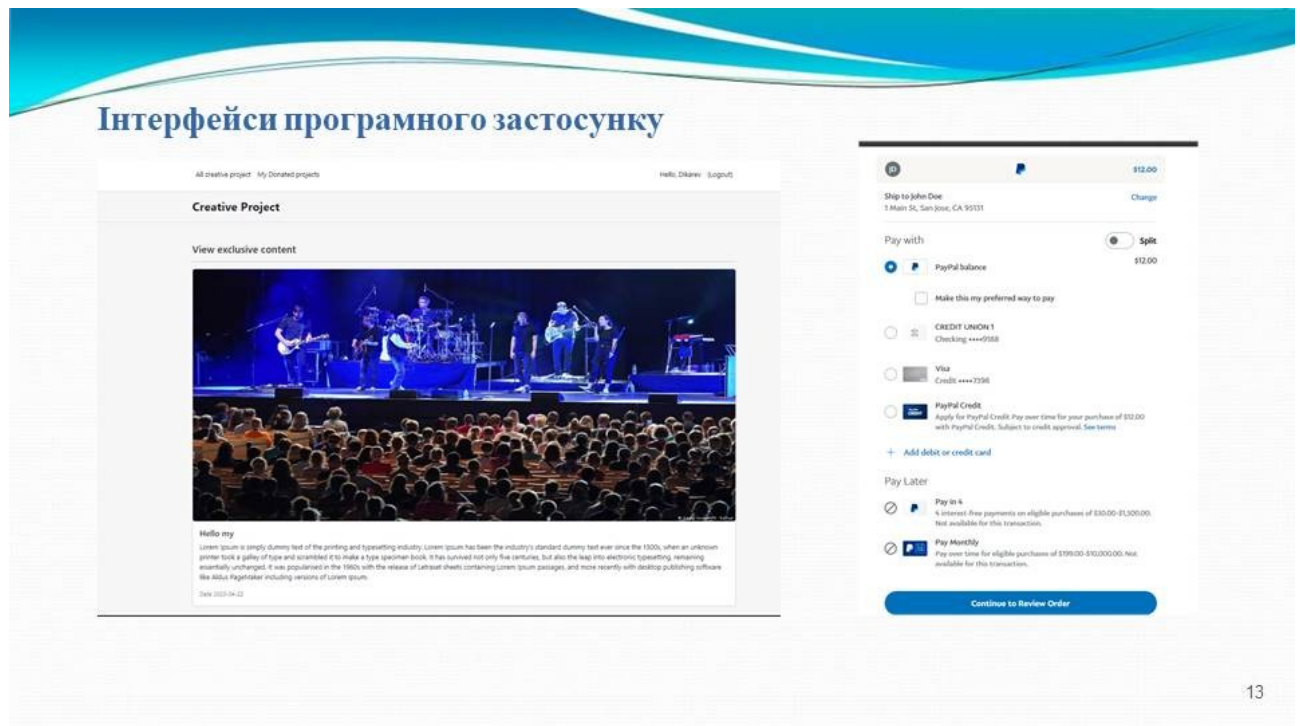


Рисунок В.13 – Слайд №13

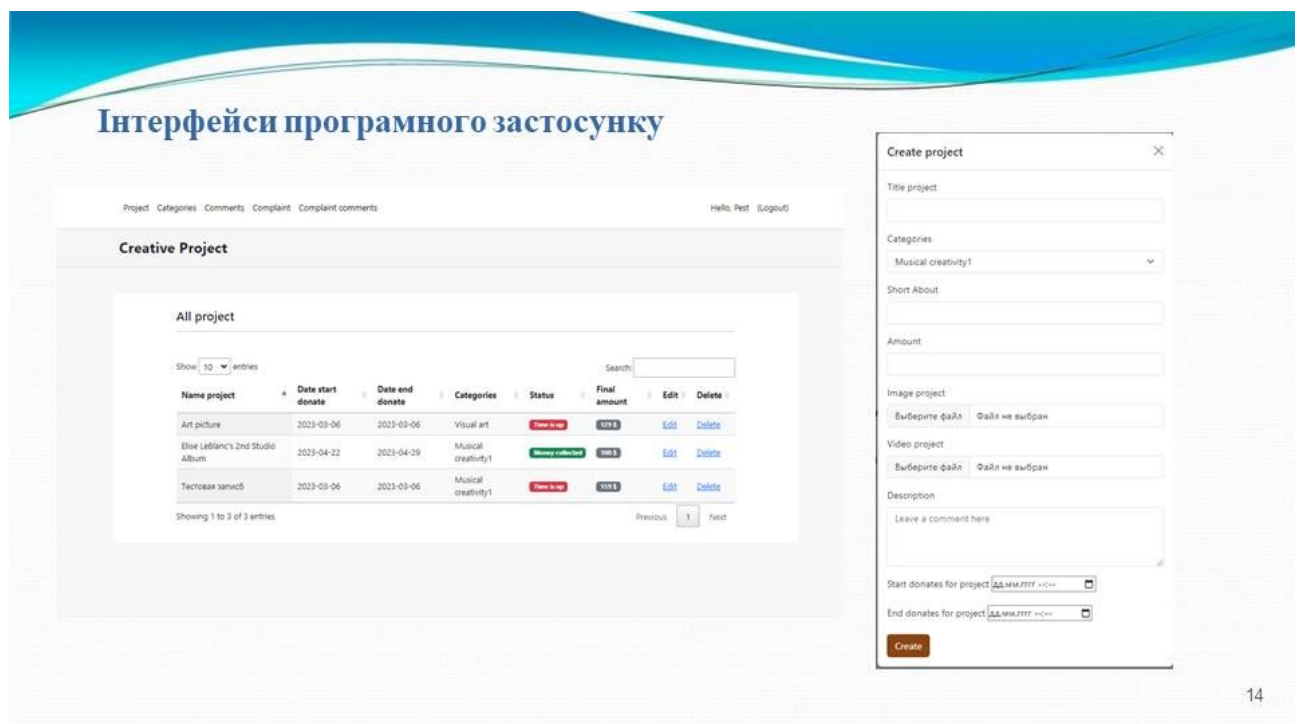


Рисунок В.14 – Слайд №14

Висновок

Під час виконання дипломної кваліфікаційної роботи було розроблено програмне забезпечення громадського фінансування творчих проєктів. Було проаналізовано переваги та недоліки аналогів і розроблено функціональність власного програмного продукту.

Для реалізації програмного забезпечення було використано мову програмування PHP, вебфреймворк Laravel, шаблон проєктування MVC та систему керування базами даних MySQL. Для розробки інтерфейсної частини було використано Bootstrap, а також мову розмітки HTML та каскадну таблицю стилів (CSS). У програмі було реалізовано інтеграцію з API платіжної системи PayPal для зручної відправки грошових пожертвувань для будь-яких творчих проєктів.

Після реалізації програми було проведено її тестування програми, що підтвердило працездатність розробленої програмної системи.

15

Рисунок В.15 – Слайд №15

Публікації

Льовкін, В. М. Програмне забезпечення спільного фінансування творчих проєктів [Електронний ресурс] / В. М. Льовкін, А. А. Дикарьов // Тиждень науки-2023. Факультет комп'ютерних наук і технологій. Тези доповідей науково-практичної конференції, Запоріжжя, 24–28 квітня 2023 р. – Запоріжжя : НУ«Запорізька політехніка», 2023. – (прийнято до друку).

16

Рисунок В.16 – Слайд №16