

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут інформатики та радіоелектроніки,
Факультет комп'ютерних наук і технологій

(повне найменування інституту, назва факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавр

(ступінь вищої освіти)

на тему РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ
ДЛЯ ЗБОРУ КОШТІВ НА РЕАЛІЗАЦІЮ ВЛАСНИХ ПРОЄКТІВ
FUNDRAISING SOFTWARE DEVELOPMENT
FOR INDIVIDUAL PROJECTS

Виконав: студент(ка) 4 курсу, групи КНТз-118_2
Спеціальності 121 Інженерія програмного
забезпечення

(код і найменування спеціальності)

Освітня програма (спеціалізація)
Інженерія програмного забезпечення

Клименко Є.В.

(прізвище та ініціали)

Керівник Зайко Т.А.

(прізвище та ініціали)

Рецензент Гофман Є.О.

(прізвище та ініціали)

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

Інститут, факультет ІРЕ, 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.
С.О. Субботін
“ 20 ” травня 2021 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Клименка Євгенія Вікторовича
(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Розробка програмного забезпечення для збору коштів на реалізацію власних проєктів.

Fundraising Software Development for Individual Projects

керівник проєкту (роботи) Зайко Тетяна Анатоліївна, к.т.н., доцент,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від “30” березня 2021 року № 103

2. Строк подання студентом проєкту (роботи) 20 травня 2021 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 програмного забезпечення. 3. Основні рішення щодо реалізації компонентів системи. 4. Експлуатація, тестування та експериментальне дослідження 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	Зайко Т.А., доцент		
Нормоконтроль	Липовець М.В., асистент		

7. Дата видачі завдання “10” березня 2021 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.	2-3 тижні	Розділ 1
3	Розробка архітектури програми.	4 тиждень	Розділ 2
4	Розробка програми.	5-6 тижні	Розділ 3
5	Тестування та експериментальне дослідження програми.	7 тиждень	Розділ 4
6	Оформлення пояснювальної записки та документів до неї. Нормоконтроль та Рецензування.	8 тиждень	Додатки
7	Захист роботи.	9 тиждень	

Студент(ка)


(підпис)

Клименко Є.В.
(прізвище та ініціали)

Керівник проєкту (роботи)

(підпис)

Зайко Т.А.
(прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 76 с., 34 рис., 3 табл., 3 дод., 15 джерел.

ПРОЄКТИ, ФІНАНСУВАННЯ, РОЗРОБКА, ПРОЄКТУВАННЯ, ВЕБЗАСТОСУНОК.

Об'єкт роботи – процеси збору колективного фінансування для реалізації авторських проєктів.

Предмет роботи – програмне забезпечення збору коштів на реалізацію проєктів.

Мета роботи – розробити програмне забезпечення для збору коштів на реалізацію власних проєктів.

Матеріали, методи та технічні засоби: об'єктно-орієнтоване програмування, мова програмування PHP, фреймворк Yii, система керування базою даних MySQL.

Результати. Розроблено програмне забезпечення для збору коштів на реалізацію власних проєктів, що реалізує інструментарій для створення власного рахунку в системі для підтримки проєктів.

Практична цінність роботи полягає в розробці кроссплатформного вебзастосунку для створення власного профілю із балансом коштів, які можна поповнити та направити на фінансування проєктів інших користувачів.

Висновки. Розроблено програмне забезпечення, що реалізує платформу для фінансової підтримки проєктів користувачів.

Галузь використання – активні користувачі Інтернету, стартапери та впливові інвестори.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 76 pages, 34 figures, 3 tables, 3 appendixes, 15 sources.

PROJECTS, FINANCING, DEVELOPMENT, DESIGN, WEB APPLICATION.

The object of the work is the processes of collecting collective funding for the implementation of author's projects.

The subject of the work is the software for raising funds for the implementation of projects.

The purpose of the work is to develop software for raising funds for the implementation of their own projects.

Materials, methods and technical tools: object-oriented programming, PHP programming language, Yii framework, MySQL database management system.

Results. Software has been developed to raise funds for the implementation of their own projects, which implements tools for creating their own account in the system to support projects.

The practical value of the work lies in the development of a cross-platform web application for creating your own profile with a balance of funds that can be replenished and used to finance projects of other users.

Conclusions. Software has been developed that implements a platform for financial support of user projects.

The field of use is active Internet users, startups, and influential investors.

ЗМІСТ

	С.
Перелік скорочень та умовних познач.....	8
Вступ.....	9
1 Аналіз предметної області.....	11
1.1 Основні принципи колективного фінансування проєктів	11
1.2 Способи фінансування в краудфандінгу	12
1.3 Огляд програмних аналогів.....	13
1.3.1 Краудфандинговая платформа «Спільнокошт».....	13
1.3.2 Краудфандинговая платформа «Kickstarter».....	14
1.3.3 Платформа «Patreon»	15
1.3.4 Порівняння існуючих аналогів	17
1.4 Постановка завдання.....	19
1.5 Висновки за розділом 1	19
2 Проєктування програмного забезпечення	20
2.1 Вибір середовища проєктування	20
2.1.1 Вибір мови програмування	20
2.1.2 Вибір фреймворку для розробки	23
2.2 Проєктування системи.....	25
2.3 Проєктування логіки взаємодії	26
2.4 Висновки за розділом 2	27
3 Основні рішення щодо реалізації компонентів системи.....	28
3.1 Проєктування системи на основі предметної області.....	28
3.2 Висновки за розділом 3	34
4 Експлуатація, тестування та експериментальне дослідження програми	35
4.1 Опис застосування	35
4.2 Умови виконання програми	35
4.3 Інструкція по експлуатації програми.....	36
4.3.1 Звернення до програми.....	36
4.3.2 Вхідні й вихідні дані	36

4.3 Тестування сценаріїв роботи з програмою.....	36
4.4 Тестування за допомогою програми Xenu's Link Sleuth	40
Висновки	43
Перелік джерел посилання	45
Додаток А Технічне завдання	47
Додаток Б Текст програми	52
Додаток В Слайди презентації.....	70

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

- MVC – Model-view-controller;
- UML – Unified Modeling Language;
- ОС – операційна система;
- ПЗ – програмне забезпечення;
- ПК – персональний комп'ютер;
- ТЗ – технічне завдання.

ВСТУП

Через швидке зростання кількості активних Інтернет-користувачів, ідеї щодо надання послуг онлайн – стають дедалі актуальними серед власників бізнесу. З'являється все більше таких сервісів, як: інтернет-магазини, онлайн книгарні, сайти афіші, та навіть сервіси з оплатою комунальних послуг.

Краудфандинг – колективне фінансування ідей. Краудфандинг – заняття добровільне. Ідея повинна захопити спонсора, тоді він дасть грошей на її втілення. Він може фінансувати не повністю весь проєкт, ставши одноосібним інвестором, а внести лише невелику частину суми.

Проєкти можуть бути культурні, соціальні, комерційні, політичні, креативні та особисті. Збори проводяться через Інтернет за допомогою краудфандінгових платформ. Спонсори (бекери) не отримують нічого натомість пожертвування або отримують винагороду нефінансове (сувенір, передзамовлення продукту) або фінансове (роялті, акції). Запустити кампанію можна своїми силами і безкоштовно: зареєструватися, вибрати схему фінансування, просувати проєкт, отримати кошти. Для того щоб краудфандінгова кампанія стала успішною і про проєкт дізналися і підтримали якомога більше людей, необхідні, крім оригінальної ідеї, відмінний маркетинг і піар. Для проходження авторизації на Kickstarter і здійснення прийому платежів потрібен статус місцевого резидента (наявність юрособи в одній з країн роботи Платформи або партнер). Для IndieGoGo досить мати рахунок у банку.

Автор реєструє свій проєкт на сайті крауд-платформи, описує ідею, складає кошторис, готує матеріали для просування. Експерт перевіряє проєкт на відповідність законодавству і правилам майданчика, розміщує інформацію і запускає кампанію.

Автор веде збір коштів 1-3 місяці. Автор і крауд-платформа рекламують проєкт, спонсори приходять на сторінку і дають гроші.

Якщо за відведений час проєкт набирає потрібну суму, він вважається успішним, гроші йдуть автору, і він дякує спонсорів за їх вклади.

Коли проєкт не залучає запитуваних коштів, він вважається проваленим, і гроші повертаються спонсорам.

Метою роботи в результаті було встановлено розробити програмне забезпечення (ПЗ) для збору коштів на реалізацію власних проєктів.

Об'єкт роботи – збору колективного фінансування для реалізації авторських проєктів. Предмет роботи – ПЗ збору коштів на реалізацію проєктів.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Основні принципи колективного фінансування проєктів

Однією з найбільш серйозних проблем, що перешкоджають початку бізнесу, є відсутність стартового капіталу. Десять років тому в мережі з'явився новий інструмент залучення капіталу – краудфандинг [1].

Краудфандинг – це спосіб збору коштів для майбутнього фінансування певного проєкту від будь-якого бажаючого. Ті, хто збирають гроші, називаються реципієнтами, а тих, хто жертвує – донори, спонсори або інвестори [2].

Реципієнт пропонує (виносить на розгляд) бізнес-ідею, задати необхідну грошову суму і термін її збору, провести калькуляцію всіх витрат, оголосити умови для донорів (безоплатне фінансування, фінансування в обмін на нематеріальну подяку, фінансування в обмін на частку в майбутньому прибутку) і згенерувати пакет своєї пропозиції на краудфандинговій платформі. Бізнес-ідея має бути в правовому полі та не порушувати чинне державне законодавство. Інформація про процес збору коштів повинна бути відкрита та прозора для всіх бажаючих (або хоча б для більшості авторизованих користувачів платформи), а зібрані гроші мають бути використані виключно для реалізації поставлених цілей (надається певна звітність про витрати) [1], [3].

Краудфандингова платформа для стартапера можж бути не просто способом залучення фінансування, а й можливістю привернути увагу до бізнес-ідеї. Це в свою чергу підвищує вірогідність стартапера потрапити на розгляд в сфері інтересів бізнес-ангелів, великих інвесторів, бізнес-інкубаторів і акселераторів. Основні складові успішної краудфандингової кампанії-маркетинг та зв'язок із спільнотою [4].

Приватні проєкти, на які наколюються фінансування з використанням краудфандингу, для інвесторів в більшості своїй мають вигляд певних венчурних інвестицій (для некомерційних проєктів повернення донорам

вкладень не мається на увазі в принципі). Іншими словами, ніхто не надає гарантій того, що бізнес-проект закінчиться успішно та прибутково, і донор зможе взяти участь у розподілі цього прибутку. Іноді стартапери можуть запропонувати донорам проектів альтернативну винагороду (найчастіше публічне висвітлення імені донора), спрямоване на збільшення їх репутаційного бекграунду, який стає все більш значущим фактором бізнес-успіхів [2]-[4].

1.2 Способи фінансування в краудфандінгу

Існує кілька способів фінансування в краудфандінгу.

Все або нічого. Бізнес-проект публікується на краудфандінг-платформі. Встановлюється певний термін, протягом якого може відбуватися фінансування. Якщо за цей термін не вийшло зібрати необхідну суму, то всі грошові ресурси, що були залучені, повертаються інвесторам [5].

Залишити все. Навіть якщо надходжень виявилось недостатньо, вся зібрана сума все рівно перераховується реципієнту. Виключенням може слугувати певний процент – комісія платформи [2], [5].

Вічне фінансування. Реципієнт не обмежений у часі діючої можливості направити фінансування на проект. Фінансування не припиняється доти, поки проект не вважається здійсненим або реалізованим.

Переломний момент. Фінансування має накопичувальний характер і не надходить стартаперу поки не дійде до певної контрольної точки. Наприклад, тільки після зборів половини від необхідної суми, гроші передаються реципієнту – це є свідченням актуальності проекту [1], [4].

Угода з вільною ціною. Форма певного аукціону – продукт виставляється на продаж реципієнтом, а інвестори пропонують за нього вартість. Таким чином можна зібрати кошти на реалізацію нової ідеї [3], [5].

1.3 Огляд програмних аналогів

На сьогоднішній день існує достатньо багато платформ, що в тій або іншій мірі надають можливості колективного збору грошей на реалізацію певних проєктів.

Такі рішення можуть бути всесвітньо визнаними та розповсюдженими, а можуть мати локальний характер. Більшість з них все ж таки мають певні спільні риси.

1.3.1 Краудфандинговая платформа «Спільнокошт»

Платформа націлена на розвиток сильного і відкритого суспільства, на реалізацію бажаних системних змін [6]. На платформі «Спільнокошт» збирають кошти на проєкти у сфері освіти, охорони здоров'я, літератури, музики, журналістики та наукових досліджень (рис. 1.1.) [6].

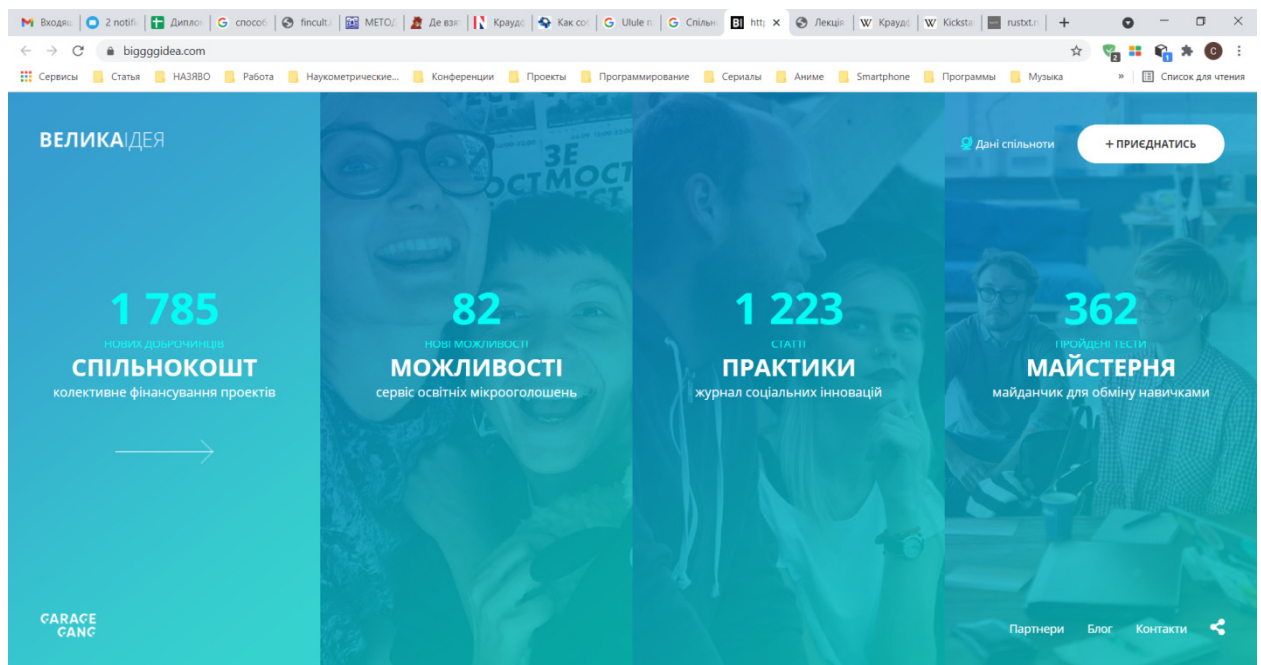


Рисунок 1.1 – Головна сторінка проєкту «Спільнокошт»

Спільнокошт – це новий інструмент для України [6]. Цей інструмент розрахований на більшість громад та об'єднань [6]. Проте цей проєкт вже має перевірений авторитет серед різних сфер – від культури, освіти, і міських інновацій до незалежних медіа-ресурсів. Кошти, залучені від реальних спонсорів, дають початковий капітал – фінансовий трамплін. Більш того, цей бекграунд є свідченням людської довіри та зацікавленості в бізнес-проєктах. Це формує абсолютно нову переговорну позицію у відносинах між владою і бізнесом [6].

Спільнокошт – це спільний внесок людей в розвиток сильного і відкритого суспільства, де більшість може зібрати фінансування та забезпечити свою діяльність. Формування міцної основи та прозора, відкрита робота об'єднання над побудовою спільноти навколо своїх активностей [6].

1.3.2 Краудфандинговая платформа «Kickstarter»

Kickstarter – сайт для залучення грошових коштів на реалізацію творчих, наукових і виробничих проєктів за схемою краудфандингу (рис. 1.2.) [7]. Kickstarter фінансує різноманітні проєкти, такі, як створення фільмів незалежного кінематографа США, коміксів, відеоігор, музики і т.п. [7].

Kickstarter полегшує збір грошових коштів, створивши модель, яка може бути краще традиційних способів інвестування. Той, хто хоче отримати фінансування, повинен зареєструватися і розмістити опис проєкту на Kickstarter. Kickstarter містить рекомендації, які проєкти будуть прийняті [7].

Власник проєкту повинен вказати термін і мінімальну кількість коштів, яке необхідно зібрати. Якщо проєкт не зібрав потрібну кількість коштів до певного терміну, то гроші повертаються спонсорам. Гроші збираються за допомогою Stripe [7].

Kickstarter бере 5% від залучених коштів; Stripe стягує додаткові 3-5%. На відміну від багатьох форумів зі збору коштів або інвестицій,

Kickstarter не претендує на право власності на проекти і роботи, які вони приймають до публікації на своєму сайті, але мають право використовувати отриману від Вас інтелектуальну власність в будь-яких цілях, не ексклюзивно. Проте, проекти, що здійснюються на сайті, зберігаються і доступні для громадськості. Після того, як фінансування проєктів завершується, завантажена інформація та матеріали не можуть бути відредаговані або видалені з сайту [7].

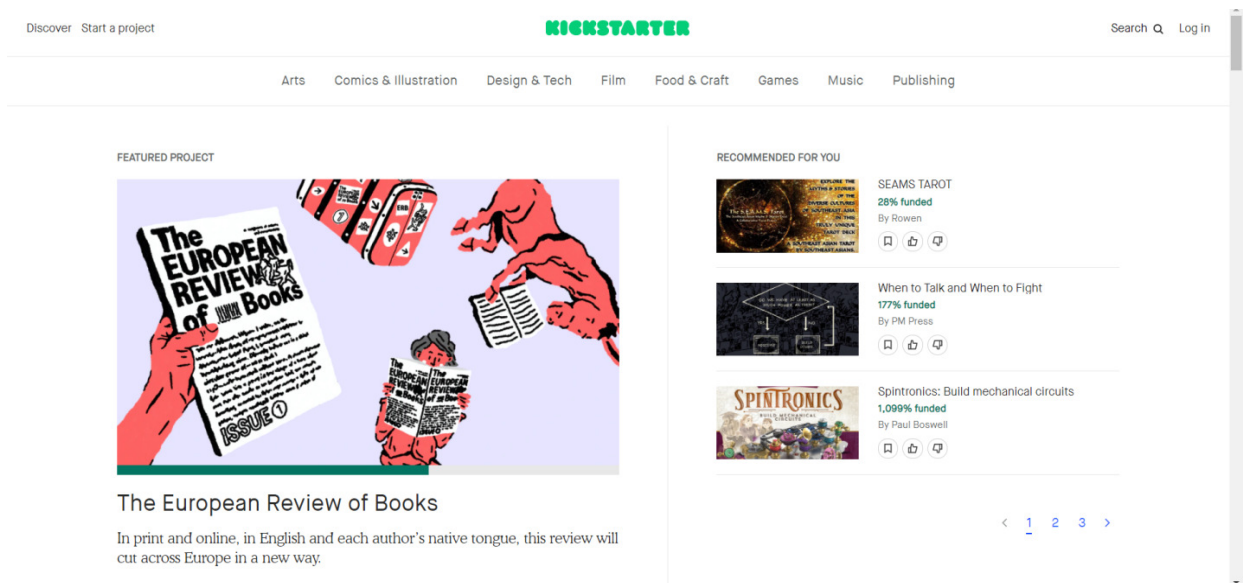


Рисунок 1.2 – Головна сторінка проєкту «Kickstarter»

1.3.3 Платформа «Patreon»

Patreon – вебплатформа, на якій автори творчих виробів (відео, фото, зображень, музики) можуть поширювати свої роботи по платній підписці або надавати додатковий контент для своїх передплатників, званих «покровителями» (англ. patrons) [8].

Сайт популярний серед YouTube-відеоблогерів, художників вебкоміксів, письменників, подкастерів, музикантів і діячів інших областей творчості, що публікується в Інтернеті [8]. Patreon дозволяє їм отримувати фінансування безпосередньо від фанатів на регулярній основі або за кожен

публікований твір. Компанія була заснована музикантом Джеком Конте і розробником Семом Ямом в 2013 році, базується в Сан-Франциско [8].

Користувачі Patreon згруповані за видами контенту, що включає відео/фільми, подкасти, комедійний контент, комікси, ігри, освітній контент, і так далі. Ці автори контенту заводять особисті сторінки на сайті Patreon (рис. 1.3), на яких шанувальники можуть вирішити почати пересилати автору фіксовані суми на щомісячній основі [8]. В якості альтернативи автори контенту можуть налаштувати свої сторінки так, щоб шанувальники платили за кожне публікується творіння. Зазвичай автори задають мету-суму, яку вони бажають отримувати – і можуть встановити ліміт щомісячних платежів. Шанувальники в будь-який момент можуть припинити фінансування. Як правило, автори дають членські привілеї (зазвичай у вигляді доступу до ексклюзивного контенту і подробиць про роботу за лаштунками) для своїх шанувальників в залежності від щомісячно перерахованої суми [8].

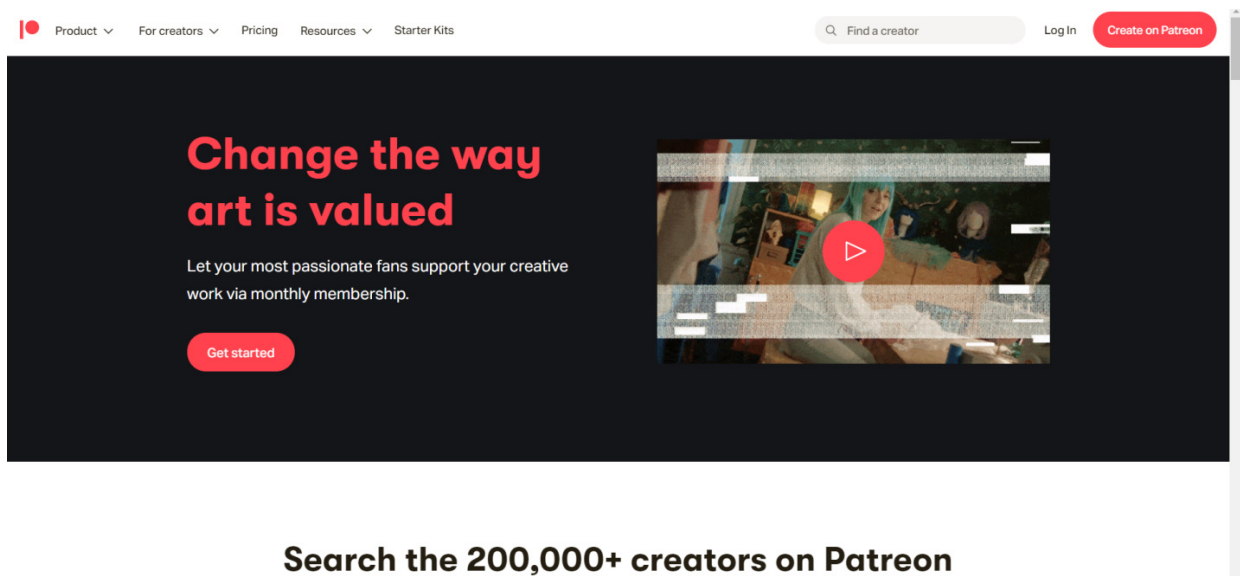


Рисунок 1.3 – Інтерфейс головної сторінки платформи Patreon

Значна частина авторів на Patreon: YouTube-відеоблогери. У них є можливість створювати різний контент на декількох платформах: в той час як YouTube-відео можуть залишатися доступними для широкої публіки, шанувальники будуть отримувати ексклюзивний особистий контент в обмін

за фінансову допомогу [8]. Patreon залишає собі 5% в якості комісії, ще 5% може бути взято за проведення транзакції — таким чином, автору дістається 90% переведених грошей [8].

За даними лютого 2014 року, майже половина користувачів сайту займається створенням відео для YouTube, а більша частина решти є письменниками, художниками вебкоміксів, звичайними художниками, музикантами або подкастерами [8].

1.3.4 Порівняння існуючих аналогів

Слід зазначити, що більшість проаналізованих платформ мають спільні риси та, врешті решт, використовуються для спільних задач, проте між ними є ряд відмінностей. Спробуємо порівняти існуючі аналоги з системою, що планується розробляти. Серед критеріїв порівняння виділимо наступні:

- типи проєктів для фінансування – види та типи проєктів, що можуть розміщуватися на платформі для подальшого збору фінансування;
- відкритість проєктів – доступність інформації про проєкт та його поточний стан фінансування для спільноти;
- захист даних платників – загальний критерій для визначення платіжної системи та власне захисту даних користувачів під час збору коштів;
- можливість безкоштовної підтримки – чи надає платформу можливість безкоштовної підтримки проєктів (через репости, поширення інформації, тощо).

Результати порівняння зведені в табл. 1.1.

Таблиця 1.1 – Порівняння програмних аналогів

	Спільнокошт	Kickstarter	Patreon	Підтримка
Типи проєктів для фінансування	Громадські проєкти	Розважальні та технологічні проєкти	Творчі проєкти	Громадські, розважальні та технологічні проєкти
Відкритість проєктів	Середній рівень	Середній рівень	Високий рівень	Високий рівень
Захист даних платників	Стороння система	Стороння система	Стороння система	Власна система
Можливість безкоштовної підтримки	Відсутня	Відсутня	Присутня	Присутня

З порівняння можливо зробити висновок, що більшість платформ мають свої нюанси. Найчастіше це пов'язано із особливостями платформи з самого початку та відображається на бізнес-проєктах. Так платформа «Спільнокошт» з самого початку планувалася, як платформа для суспільних та громадських проєктів. Саме тому там не публікують свої проєкти представники творчих груп. «Kickstarter» же навпаки з самого початку використовувався як платформа для розміщення проєктів творчими митцями, а потім популярність зробила можливим публікацію там і проєктів технічного спрямування. «Patreon» же взагалі являє собою, так би мовити, творчу збірку, адже платформа надає можливість підтримати улюбленого контент-мейкера. Проте, саме спільної платформи одразу із більшістю пропозицій на сьогоднішній день не існує. Більш того, такі платформи досить часто використовують сторонні платіжні системи (наприклад, «PayPal»), що не може в повній мірі гарантувати безпечність та захищеність особистих даних спонсора.

Саме тому, актуальною є задача розробки нової краудфандингової системи із розширеними можливостями для збору коштів на реалізацію проєктів із більш широким колом тематик бізнес-проєктів та захистом даних платників.

1.4 Постановка завдання

За результатами аналізу проблемної області та дослідження існуючих аналогів для вирішення проблемних питань, сформулюємо постановку задачі.

Метою є ПЗ для збору коштів на реалізацію власних проєктів. Серед функціональних вимог визначимо:

- реєстрація користувачів;
- авторизація користувачів;
- публікація об'яви про збір коштів;
- слідування за історією грошових операцій;
- можливість проводити фінансові операції;
- можливість перегляду історії фінансових операції клієнтів.

Система буде розроблена у вигляді вебзастосунку, що гарантує простоту розгортання на робочих станціях користувачів.

1.5 Висновки за розділом 1

Виконаний аналіз проблеми дозволив виділити основні принципи організації ПЗ для збору коштів на реалізацію власних проєктів, проаналізувати і порівняти програмні аналоги, на основі чого виділити особливості програмного забезпечення, що розробляється.

Для того щоб досягти поставлену мету, в роботі було визначено наступні завдання:

- виконати аналіз предметної області, що було досягнуто в даному розділі;
- виділити вимоги до програмного забезпечення;
- виконати проєктування програмного забезпечення;
- виконати реалізацію програмного забезпечення.

2 ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

У попередньому розділі було проведено аналіз та дослідження існуючих аналогів – платформ для спільного збору коштів на реалізацію проєктів. Виходячи з проведеного аналізу, було сформульовано основні можливості, якими повинна оперувати система, що проєктується.

Що стосується технічних та програмних можливостей, то система повинна забезпечувати можливість зручної публікації проєктів для збору коштів, надавати можливість підтримати проєкт або розповсюдити інформацію про нього. Система повинна бути кроссплатформною. Також система повинна бути простою й зрозумілою користувачу.

Даний розділ містить опис питань, які стосуються безпосередньо структури системи, що розробляється, вибору середовища проєктування, а також визначаються основні та специфічні архітектурні рішення, які має система.

2.1 Вибір середовища проєктування

2.1.1 Вибір мови програмування

PHP – мова програмування, яка найбільш поширена в сфері веброзробки [9].

В даний час PHP є одним з лідерів серед серверних мов програмування, що застосовуються для створення динамічних вебсайтів і вебдодатків. Велика частина коробкових систем управління сайтами написана саме на PHP, мова підтримується переважною більшістю хостинг-провайдерів, мова отримала широке поширення завдяки своїй простоті, швидкості, мультипарадигмальності, багатій функціональності і кроссплатформності [9]-[10].

В якості основних переваг PHP найчастіше називають широку поширеність, досить високу затребуваність PHP-розробників в поєднанні з низьким порогом входу при навчанні [9]-[10].

PHP часто критикують за неузгоджений синтаксис функцій, неортогональність дизайну, зворотну несумісність версій, а також за досить низький середній рівень культури розробки [9]-[10].

Насправді, простота освоєння і високий рівень затребуваності в дуже простих проєктах якраз і є причинами низького середнього рівня технічної експертизи і культури розробки на цій мові [9]-[10]. Просто дуже багато хто з тих, хто називає себе PHP-розробниками, насправді не мають навіть базових знань, достатніх, щоб називатися програмістами. А велика кількість проєктів з низькими вимогами до якості породжує майже повсюдне застосування методології [11].

Зрозуміло, що в PHP-розробці є і дуже хороший програмісти, що працюють з використанням кращих практик в сфері розробки ПЗ. Але їх не так вже й просто знайти, тому середній рівень за технологією визначають не вони, а величезна кількість вебмайстрів, що створюють тривіальні сайти по максимально простому технологічному процесу [9]-[11].

Роботодавці зіткнулися з кадровою проблемою при ускладненні реалізованих проєктів. На PHP досить просто знайти розробників, які можуть зібрати сайт на коробкової CMS-системі, особливо, якщо немає ніяких вимог по швидкості його роботи, надійності і безпеки. Як тільки посилюються вимоги до технічних характеристик проєкту або коли потрібна розробка на фреймворку з якісним опрацюванням архітектури, то кадровий ринок в екосистемі PHP залишає бажати кращого [9]-[11].

Дійсно хороших програмістів в принципі мало на будь-якій мові програмування, але наша практика показала, що їх пошук в рамках PHP-стека більш складне завдання в порівнянні з іншими мовами: Ruby, Python, GoLang, Java [9]-[11]. Це була одна з причин, по якій ми в 2013 році зважилися на зміну технологічного стека, перейшовши від повсюдного

застосування PHP до застосування Ruby для веброзробки і до використання Python для аналізу даних. Також свою роль зіграла і різниця екосистем цих мов програмування - в Ruby і Python виявилось більше якісних інструментів, потрібних нам для реалізації складних проєктів. Ще в Ruby і Python співтовариство розробників сконцентровано навколо невеликого числа потужних фреймворків і бібліотек, тоді як в PHP кілька сотень в різному ступені популярних фреймворків і CMS. Іншими словами, випадково обраний веброзробник на Ruby в 99% випадків вже знає і працював з Ruby on Rails, веброзробник на Python — з імовірністю 95% знає Django, а ось в PHP такого явного лідера серед платформ просто немає [9]-[11].

Проведемо порівняння мов програмування для реалізації вебзастосунку, результати порівняння представимо в табличній формі (табл. 2.1). У якості критеріїв будемо використовувати наступні:

- розповсюдженість матеріалів – критерій для визначення кількості матеріалів (відео уроків, туторіалів, тощо) та активних представників спільноти для знайомства із мовою програмування;
- платформи для спрощення розробки – наявність відкритих фреймворків та систем керування вмістом для спрощення розробки;
- масштабованість застосунків – простота розгортання та підтримки і розвитку застосунків;
- стресостійкість застосунків – здатність застосунку справлятися зі збільшенням робочого навантаження (збільшувати свою продуктивність) при додаванні ресурсів (зазвичай апаратних).

Таблиця 2.1 – Вибір мови програмування

	PHP	Python	Java (JSP)
Стресостійкість застосунків	++	+	++
Масштабованість застосунків	++	+	+
Платформи для спрощення розробки	++	+	+
Розповсюдженість матеріалів	++	++	+

За результатами порівняння можна зробити висновок, що не дивлячись на тривалий час PHP залишається кращою альтернативою, адже має і розширену спільноту, і велику кількість фреймворків, що спрощують розробку. Для написання нашого застосунку за обраною темою розробки цілком достатньо вибрати мову PHP.

2.1.2 Вибір фреймворку для розробки

Yii-це популярний фреймворк для PHP-розробки, заснований на парадигмі MVC. Основна перевага-дуже висока швидкість роботи і, як наслідок, продуктивність [11]-[12].

Фреймворк активно розвивається спільнотою. Yii не виглядає титаном в порівнянні з фреймворками Symfony і Zend Framework, кодова база яких досить об'ємна. Фреймворк Yii досить простий в освоєнні і у використанні, що сприяє швидкій розробці на ньому проєктів [11]-[12]. Однак при виборі в якості платформи для створення вебпроєкту цього фреймворку варто все одно враховувати той факт, що швидкість розробки на ньому типових рішень все одно буде нижче, ніж розробка на CMS. Як і всі фреймворки, Yii «заточений» під розробку технічно складних вебпроєктів: бізнес-додатків, вебсервісів, а також сайтів зі складною бізнес-логікою і вимогливих до швидкодії [11]-[12].

Основні переваги та можливості фреймворку Yii [11]-[12]:

- забезпечує високу продуктивність щодо інших PHP-фреймворків;
- заснований на парадигмі MVC (Модель-представлення-контролер);
- є інтерфейси DAO і ActiveRecord для роботи з базами даних (використовується PDO);
- підтримує інтернаціоналізацію;
- дозволяє кешувати як сторінки цілком, так і окремі фрагменти;

- здійснює перехоплення і обробка помилок;
- має функціонал роботи з формами, забезпечує їх побудову і валідацію;
- реалізовані аутентифікація та авторизація;
- зручний для реалізації Ajax-інтерфейсів, інтегрується з jQuery;
- у фреймворк вбудовані генератори базового PHP-коду для CRUD-операцій (скаффолдинг);
- підтримує теми оформлення;
- має можливість підключення сторонніх бібліотек;
- працює з міграціями баз даних (генерація, застосування і відкат);
- дозволяє здійснювати автоматичне тестування і вести розробку в стилі TDD;
- підтримує стиль REST.

Серед недоліків виділяють [11]-[12]:

- слабка екосистема навколо фреймворку серед англomовного сегмента розробників. Спільнота занадто сильно розмазана по різних місцях: кілька форумів з невеликою активністю, stackoverflow і т. п.;
- хоч фреймворк і дозволяє робити код простим, але далеко не елегантним. Якщо його синтаксис порівнювати з фреймворком Laravel, то він поступається;
- Yii відстає від мови, стандартів та інших фреймворків. Нові оновлення з дійсно корисними функціями виходять не так часто;
- занадто велика зв'язаність backend і frontend частин Yii2. Фреймворк пропонує використовувати бібліотеку jQuery і Bootstrap, які вбудовані за замовчуванням в ядро фреймворку. Цей мінус планується виправити в новій версії фреймворку Yii 3, зробивши його компоненти менш пов'язаними.

У таблиці 2.2 наведено порівняння фреймворків за критеріями. Серед характеристик для порівняння виділімо:

- простота розробки представлень (view – front-end) – використання шаблонізаторів прискорює розробку і спрощує створення front-end програми за рахунок, наприклад, вирішення завдання автоматичного екранування HTML;
- плагіни та розширення – підтримка сторонніх плагінів (модулів або розширень), що спрощують розробку через повторне використання коду;
- спільнота – загальний критерій, що характеризує наявність актуальних матеріалів для вирішення проблемних питань під час розробки;
- швидкість роботи розробленого застосунку – швидкість відпрацювання застосунку, розробленого з використанням того чи іншого фреймворку.

Таблиця 2.2 – Порівняння середовищ розробки

	Yii	Laravel
Простота розробки представлень	++	+
Плагіни та розширення	+	++
Спільнота	++	++
Швидкість роботи розробленого застосунку	++	±

2.2 Проєктування системи

Для побудови архітектурного каркасу ПЗ буде використано шаблон проєктування Model-view-controller (MVC) [13]. Ця схема використання декількох шаблонів проєктування, за допомогою яких модель даних програми, користувацький інтерфейс і взаємодія з користувачем розділені на три окремих компоненти.

View – відповідає за збір даних від користувачів в різних формах та обробку їх у відповідності зі специфікацією.

Controller – обробляє дані, отримані від користувача для передачі на наступний рівень.

Model – сукупність правил, принципів, залежностей поведінки об'єктів предметної області. Це реалізація предметної області в інформаційній системі.

На рисунку 2.1 показано, який вигляд має архітектура ПЗ на основі предметної області з використанням шаблону проектування MVC.

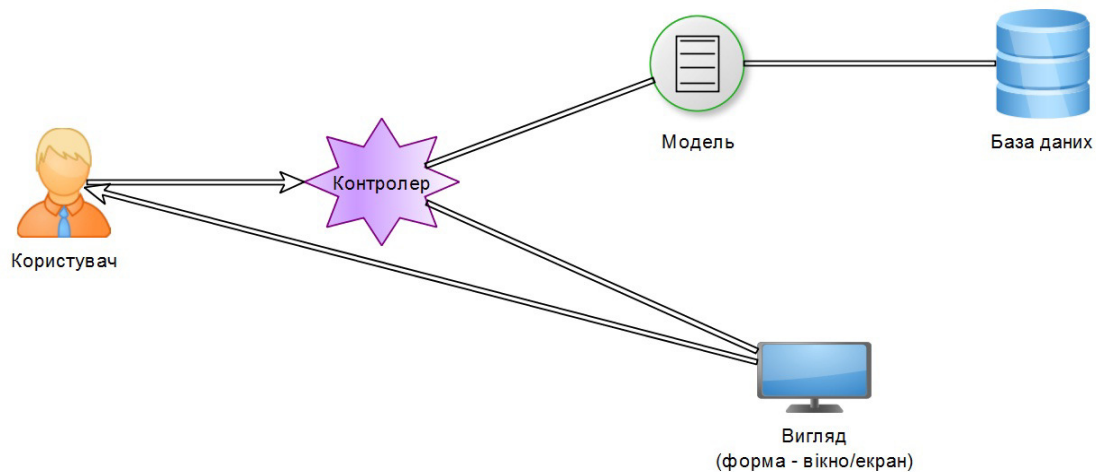


Рисунок 2.1 – Архітектуру вебзастосунку

База даних – набір відомостей, що зберігаються деяким впорядкованим способом.

Модель – надає дані і методи роботи з цими даними.

Вигляд (Представлення) – відповідає за відображення інформації. Найчастіше в якості подання виступає форма (вікно, екран) з графічними елементами.

Контролер – забезпечує зв'язок між користувачем і системою: контролює введення даних користувачем і використовує модель та подання для реалізації необхідної реакції.

2.3 Проектування логіки взаємодії

Діаграми прецедентів застосовуються для моделювання вигляду системи з точки зору прецедентів (або варіантів використання). Даний вид

діаграм використовується для моделювання динамічних аспектів системи, так само, як і діаграми станів та діяльності.

На рисунку 2.2 зображено діаграму прецедентів для системи «Підтримка», що відображає логіку взаємодії Користувача та Адміністратора із системою.

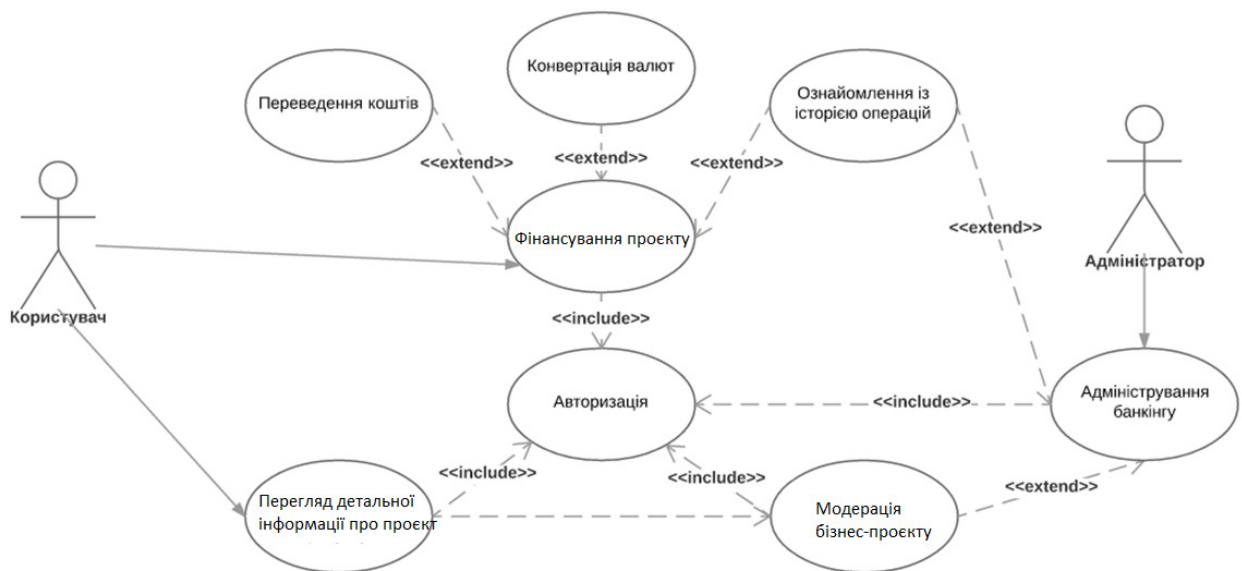


Рисунок 2.2 – Діаграма прецедентів

2.4 Висновки за розділом 2

Таким чином, архітектура програмного засобу дозволяє вирішувати задачі, що були поставлені перед системою, пов'язані із створенням новин про нові проєкти для фінансування та переведення коштів для фінансування. Засоби, що використовуються, забезпечують реалізацію функцій програми.

Для програмної реалізації можливостей ПЗ, що надає можливість колективного фінансування проєктів, виділених під час аналізу у попередньому розділі, проведено вибір засобів та обґрунтовано структуру системи, що проєктується.

Аналізуючи обрану структуру ПЗ, можна зробити висновок, що вона є лаконічною, але, разом з тим, досить працездатною, використовує ресурси раціонально.

3 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ

3.1 Проєктування системи на основі предметної області

Unified Modeling Language (UML) – уніфікована мова моделювання, що використовується у парадигмі об’єктно-орієнтованого програмування та є невід’ємною частиною уніфікованого процесу розроблення ПЗ [13]-[14]. UML заснована на діаграмах, що дають можливість представити систему у такому вигляді, щоб її можна було легко перевести у програмний код.

Діаграми класів – статичні структурні діаграми, що описують структуру системи, демонструють класи систем, їх атрибути, методи і залежності між класами [13]-[14].

Загальна діаграма класів системи зображена на рисунку 3.1.

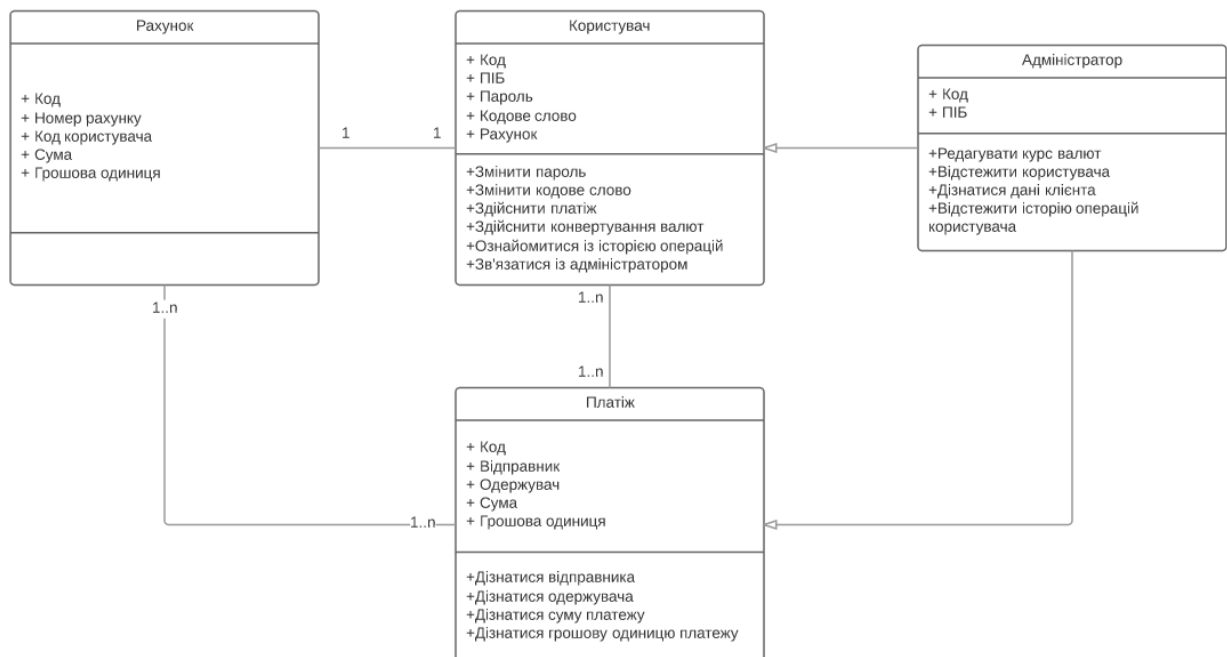


Рисунок 3.1 – Діаграма класів

Діаграми пакетів – структурні діаграми, основним вмістом яких є пакети і відношення між ними [13]-[14]. Ці діаграми слугують, в першу

чергу, для організації елементів в групі по яким-небудь ознакам з метою спрощення структури і організації роботи з моделлю системи.

Загальна діаграма пакетів системи зображена на рисунку 3.2.

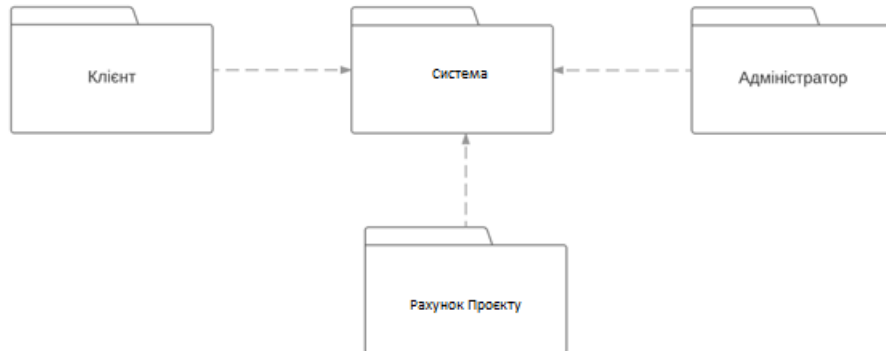


Рисунок 3.2 – Діаграма пакетів

Діаграма компонентів – статична структурна діаграма, що показує розбиття програмної системи на структурні компоненти і зв'язки (залежності) між компонентами. В якості фізичних компонент можуть виступати файли, бібліотеки, модулі, виконавчі файли, пакети і т.п. [13]-[14].

Загальна діаграма компонентів системи зображена на рисунку 3.3.

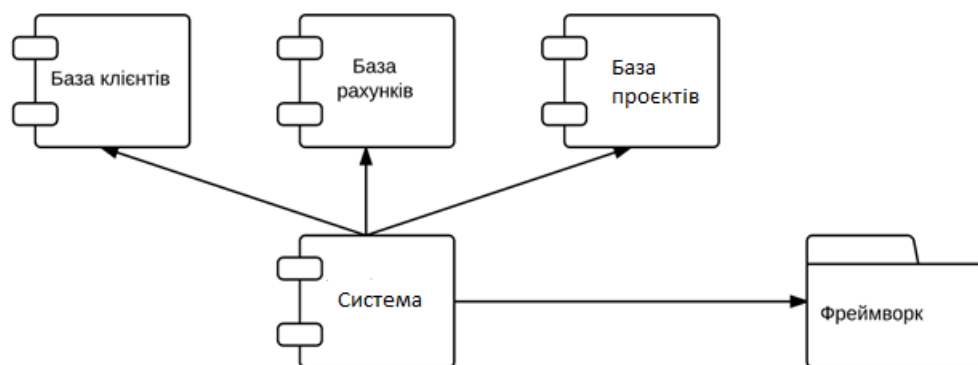


Рисунок 3.3 – Діаграма компонентів

Діаграми станів зображають всі можливі стани, в яких може знаходитись конкретній об'єкт, а також зміни стану об'єкту, які відбуваються в результаті впливу деяких подій на цей об'єкт [13]-[14].

На рисунку 3.4 зображено діаграму станів для вебзастосунку. Діаграма досліджує зміну станів платежу для фінансування проєктів розміщених на сайті.

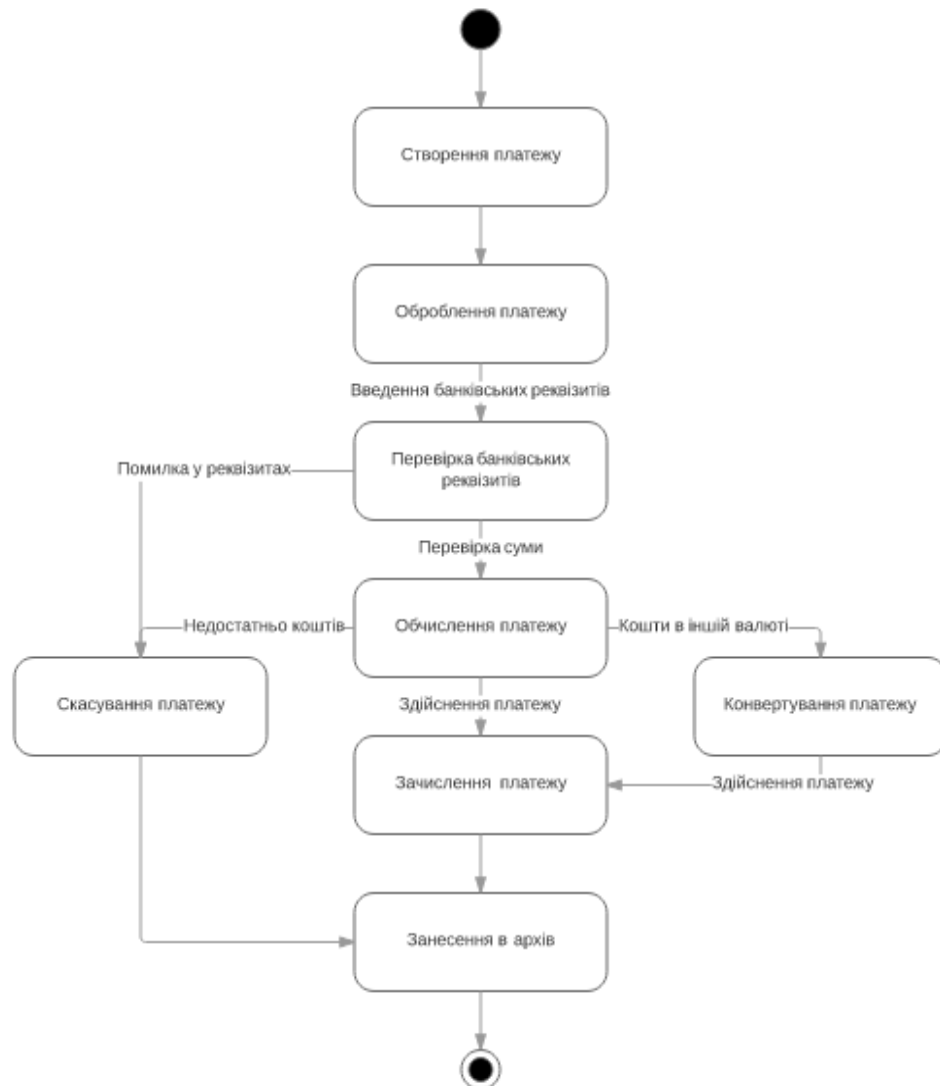


Рисунок 3.4 – Діаграма станів

Діаграма діяльності показує переходи між видами діяльності. Модель видів діяльності може подавати в графічній формі потік подій для прецеденту [13]-[14].

Діаграма діяльності для створення платежу користувачем на рис. 3.5.

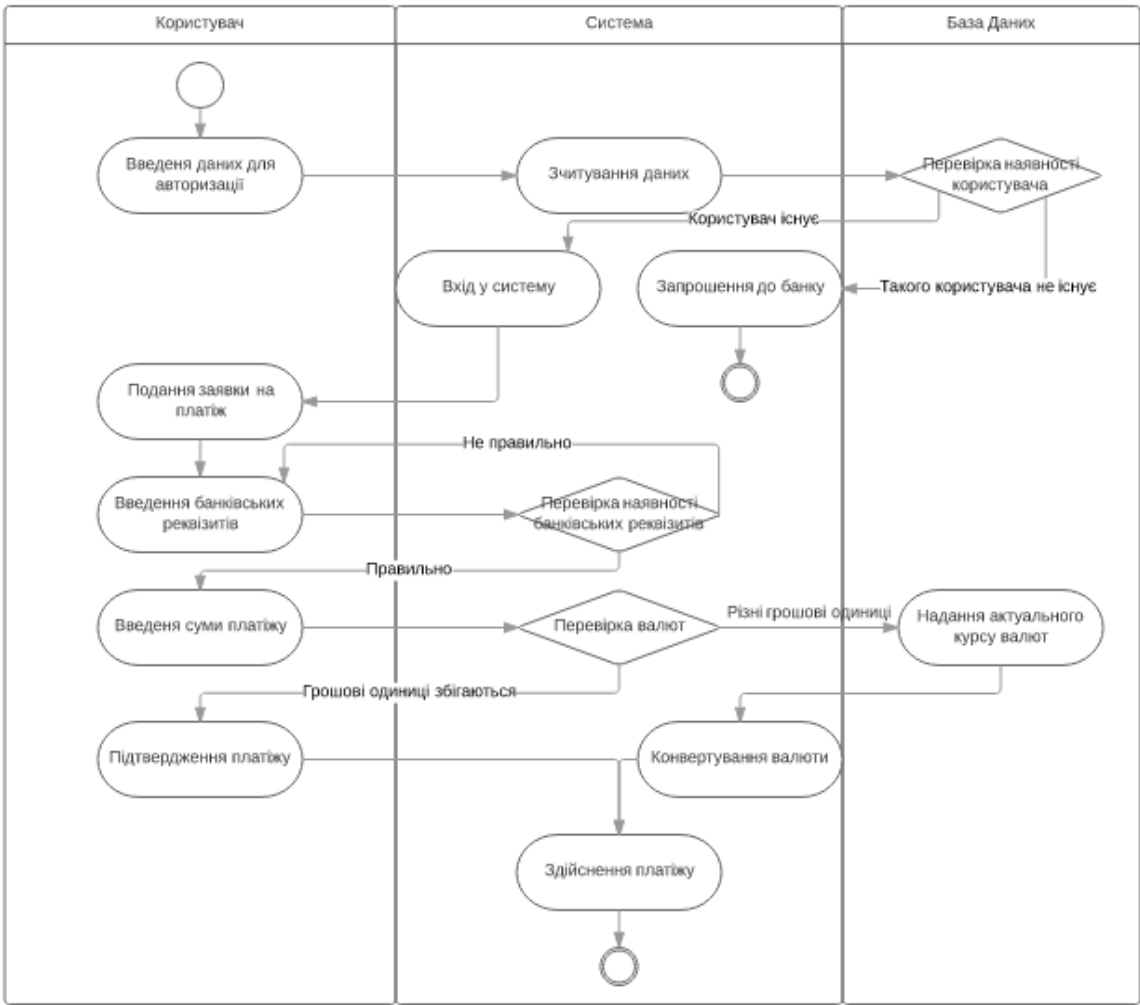


Рисунок 3.5 – Діаграма діяльності

Діаграма кооперацій призначена для специфікації структурних аспектів взаємодії [13]-[14]. Головна особливість діаграми кооперації полягає в можливості графічно уявити не тільки послідовність взаємодії, але і всі структурні відносини між об'єктами, що беруть участь в цій взаємодії.

На рисунку 3.6 наведено діаграму кооперацій для системи під час створення платежу від умовного користувача «Серафима» на фінансування проєкту умовного користувача «Амалії».

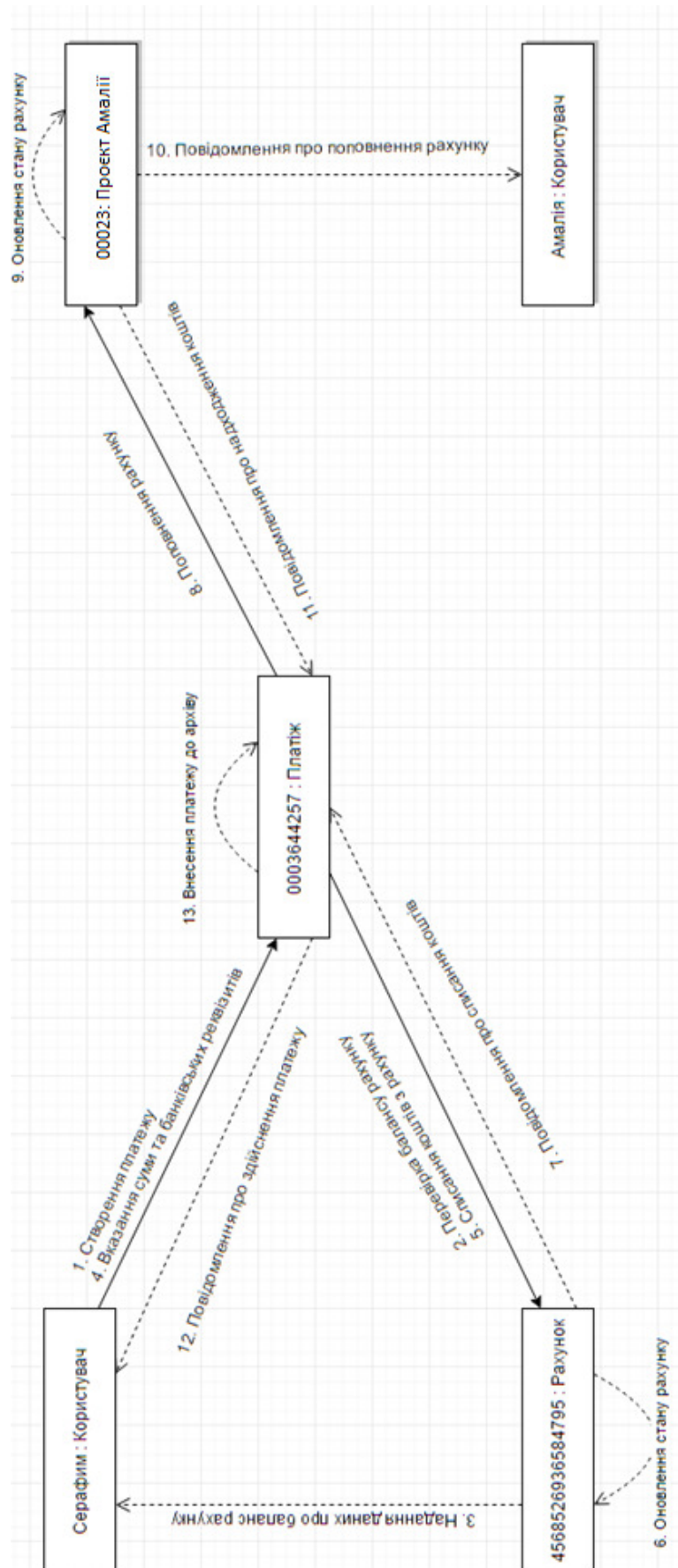


Рисунок 3.6 – Діаграма кооперацій

На рисунку 3.7 наведено діаграму послідовностей під час створення платежу від умовного користувача «Серафима» на фінансування проєкту умовного користувача «Амалії».

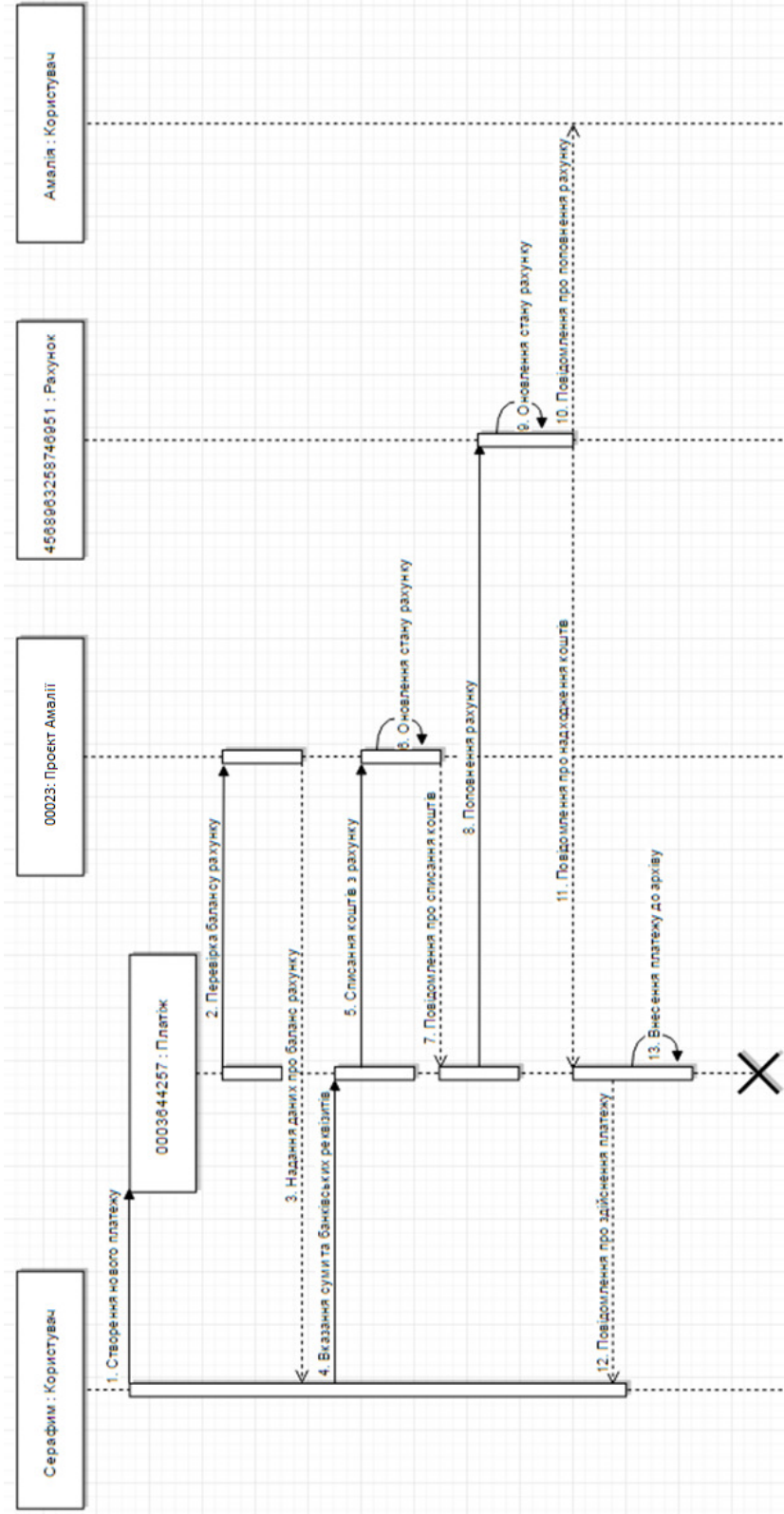


Рисунок 3.7 – Діаграма послідовностей

3.2 Висновки за розділом 3

Для програмної реалізації ПЗ для збору коштів на реалізацію власних проєктів було проаналізовано сучасні засоби для веброзробки на основі виявлених особливостей та архітектури застосунку, що розробляється в рамках даної дипломної кваліфікаційної роботи бакалавра. Було виявлені необхідні засоби. Отже, для розробки практичної частини роботи обрано:

- мова програмування PHP;
- фреймворк Yii.

Проаналізувавши обрану клієнт-серверну архітектуру можна зробити висновок: застосування, що розроблюється є надійним, зручним у використанні, має відкриту структуру, що відкриває широкі можливості по адаптації та масштабуванню.

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

4.1 Опис застосування

Завданням ПЗ для збору коштів на реалізацію власних проєктів є залучення нових інвесторів та збільшення потоку фінансування для власних проєктів стартаперів.

Системою матимуть змогу користуватися усі користувачі мережі Інтернет. Проте повний функціонал доступний виключно для зареєстрованих відвідувачем, що пройшли авторизацію. Користувачі системи зможуть обговорювати проєкти, коментувати їх, а також фінансово підтримати ті, що їх зацікавили.

4.2 Умови виконання програми

Мінімальні системні вимоги та програмне забезпечення, що необхідні для функціонування серверної частини мобільного застосунку:

- серверна операційна система (ОС) – будь-яка з різновидів операційної системи Unix (включаючи Mac OS X) або Windows;
- вебсервер – HTTP-сервер Apache для Unix і Windows, сервер на кшталт XAMPP;
- інтерпретатор PHP5 та PHP7;
- MySQL-сервер із правами створення та вилучення користувачів БД (або база SQLite з аналогічними правами).

Для експлуатації клієнтської частини необхідні такі програмно-технічні засоби:

- персональний комп'ютер (ПК);
- процесор із тактовою частотою від 1,5 GHz;
- обсяг оперативної пам'яті – не менше 3GB;
- підключення до мережі Internet;

– встановлений вебглядач (браузер) останньої версії на рушії Chromium (Google Chrome), Gecko (Mozilla Firefox) або інших (Opera, Microsoft Edge).

4.3 Інструкція по експлуатації програми

4.3.1 Звернення до програми

Для звернення до системи необхідно запустити сервер за допомогою іконки Start XAMPP, відкрити браузер, ввести адресу сайту та за необхідністю зареєструватися або авторизуватися на головній сторінці. Після відкриття доступу до системи, можливо переходити за посиланнями на головній сторінці вебзастосунку для виконання відповідних функцій.

4.3.2 Вхідні й вихідні дані

Вхідними даними є інформація про користувача: логін та пароль людини, що запрошує доступ до системи. Вхідна інформація зберігається у текстовій формі в таблицях бази даних на сервері. Також до вхідної інформації можна віднести всі дані, що відносяться до певного користувача.

Вихідною інформацією є дані, що хоче отримати користувач від системи (наприклад, власний профіль, список друзів, результати пошуку та ін..), систематизована в таблицях бази даних з можливістю фільтрування за допомогою SQL-запитів для представлення користувачеві в табличній формі на сторінках вебзастосунку.

4.4 Тестування сценаріїв роботи з програмою

Тестування програми відбувалось шляхом розміщення вебзастосунку на вебсервері в локальній мережі.

На рисунку 4.1, продемонстровано коректне відображення додатку через браузер GoogleChrome.

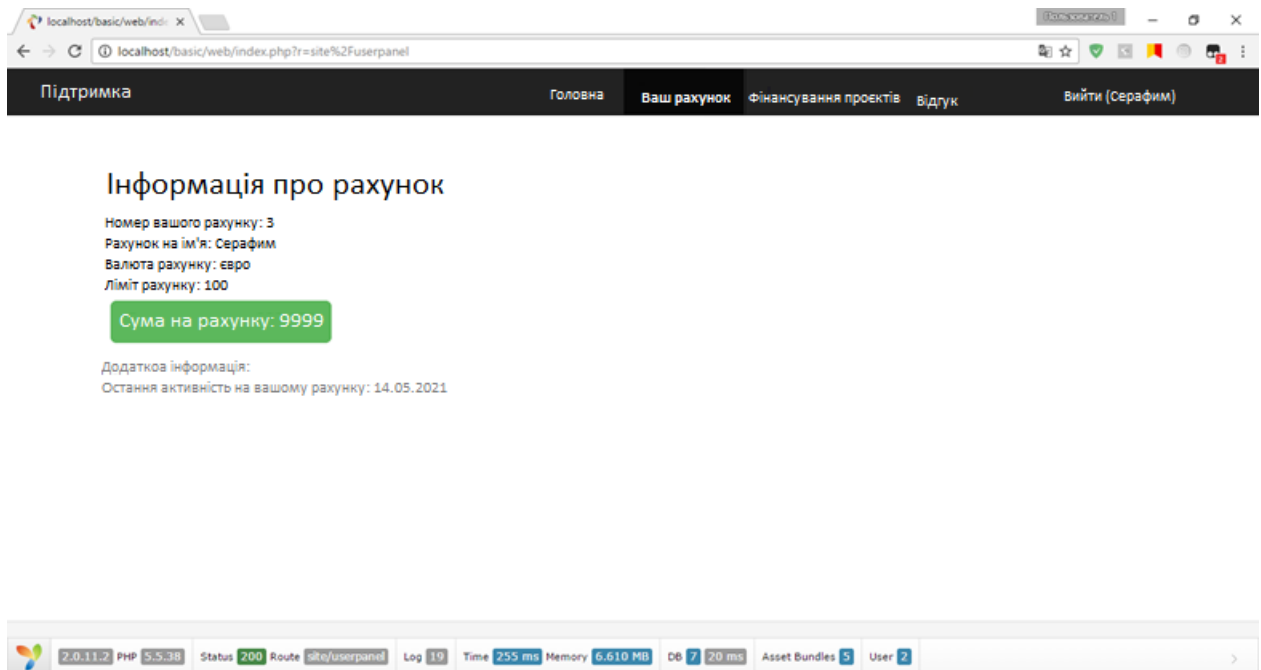


Рисунок 4.1 – Відображення додатку через браузер GoogleChrome

На рисунку 4.2, продемонстровано коректне відображення додатку через браузер MozillaFirefox.

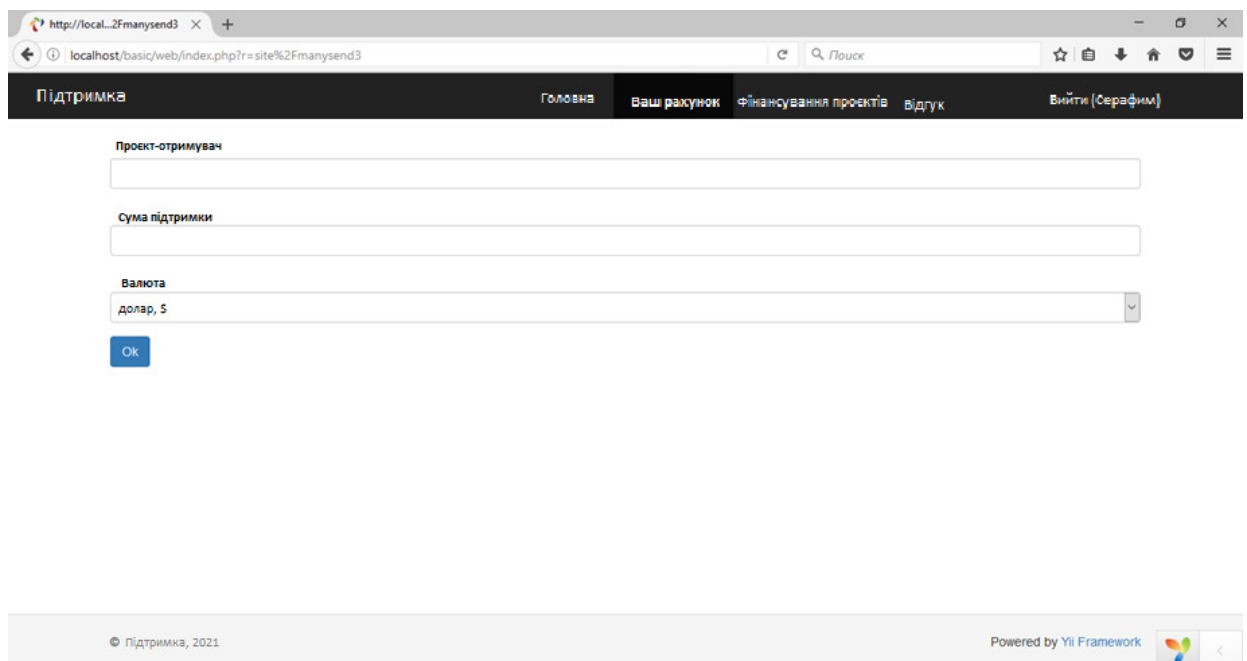


Рисунок 4.2 – Відображення додатку через браузер MozillaFirefox

На рисунку 4.3, продемонстровано коректне відображення додатку через браузер Опера.

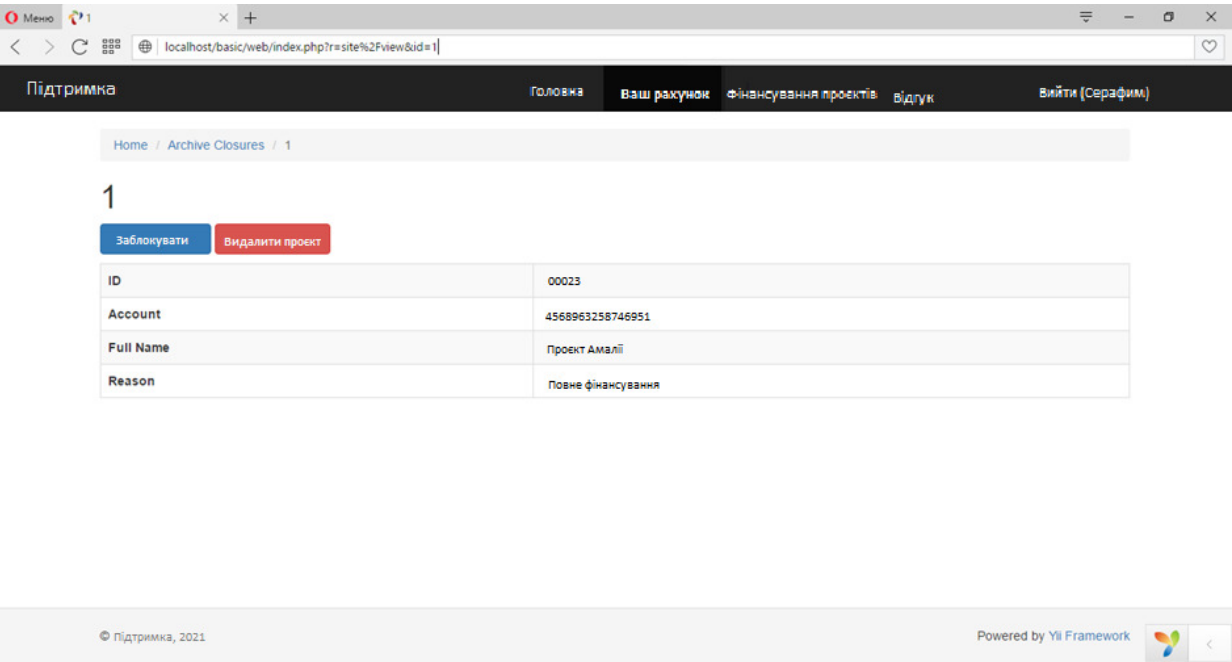


Рисунок 4.3 – Відображення додатку через браузер Опера

На рисунку 4.4, продемонстровано коректне відображення додатку через браузер MicrosoftEdge.

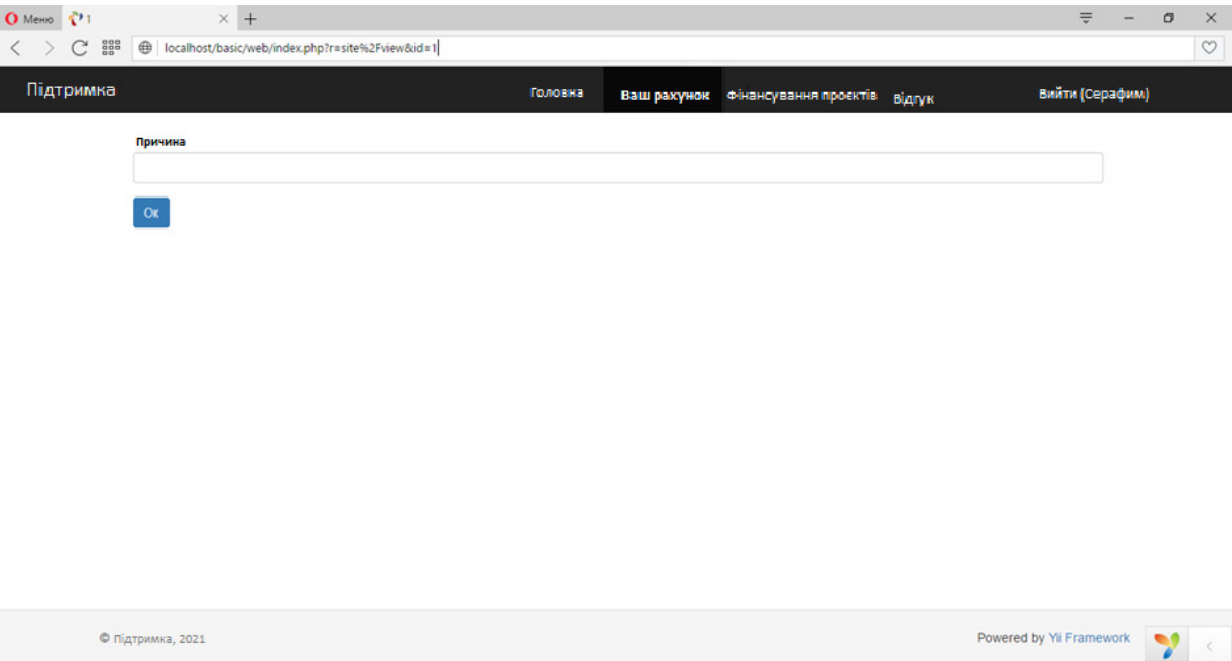


Рисунок 4.4 – Відображення додатку через браузер MicrosoftEdge

На рисунку 4.5 наведено сторінку авторизації. А на рисунку 4.6 – наведено рисунок для зворотнього зв'язку.

The screenshot shows a web browser window with the address bar displaying `localhost/basic/web/index.php?r=site%2Flogin`. The page has a dark header with the text "Підтримка" (Support) on the left and navigation links "Головна" (Home), "Ваш рахунок" (Your account), "Фінансування проєктів" (Project financing), "Відгук" (Review), and "Увійти" (Login) on the right. Below the header, a breadcrumb trail shows "Home / Login". The main heading is "Login". Below it, a message says "Please fill out the following fields to login:". There are two input fields: "Username" with the value "amaliya123" and "Password" with masked characters "****". A checkbox labeled "Remember Me" is checked. A blue "Login" button is below the fields. A note states: "You may login with admin/admin or demo/demo. To modify the username/password, please check out the code `app\models\User::$users`." The footer contains "Підтримка, 2021" on the left and "Powered by Yii Framework" with its logo on the right.

Рисунок 4.5 – Сторінка авторизації користувача

The screenshot shows a web browser window with the address bar displaying `localhost/basic/web/index.php?r=site%2Fcontact`. The page has a dark header with the text "Підтримка" (Support) on the left and navigation links "Головна" (Home), "Ваш рахунок" (Your account), "Фінансування проєктів" (Project financing), "Відгук" (Review), and "Вийти (Амалія)" (Logout (Amaliya)) on the right. Below the header, a breadcrumb trail shows "Home / Contact". The main heading is "Contact". Below it, a message says: "If you have business inquiries or other questions, please fill out the following form to contact us. Thank you." There are four input fields: "Name", "Email", "Subject", and "Body" (a larger text area). Below these fields is a "Verification Code" section with the "lejourn" logo and an input field. The footer contains the "lejourn" logo on the left and a small logo on the right.

Рисунок 4.6 – Сторінка зворотнього зв'язку

4.5 Тестування за допомогою програми Xenu's Link Sleuth

Xenu, або Xenu's Link Sleuth – це комп'ютерна програма, яка перевіряє вебсайти на наявність битих гіперпосилань [15]. Програма є пропрієтарним програмним забезпеченням і доступна на безоплатній основі. Програма названа по імені Ксену, галактичного правителя з Саєнтологічного Писання [15].

Програма створює звіт, за даними якого можна сформулювати посилання для скачування певних файлів [15].

Link sleuth працює на Microsoft Windows. Перевірка посилань проводиться на посиланнях, які з'являються в тегах <a>, а також в зображеннях, рамках, плагінах, фонах, місцевих картах зображень, таблицях стилів, скриптах і Java аплетах [15]. Програма проходить по посиланнях на інші сторінки, і перевіряє також і посилання на цих сторінках, завдяки чому можна перевірити весь сайт на биті посилання в одній сесії. Ксену відображає безперервно оновлюваний список URL-адрес, які можуть бути відсортовані за різними критеріями. Програма використовує «простий, без надмірностей, інтерфейс користувача», і це може допомогти користувачам зрозуміти, як структуровані деякі вебсайти. У програмі є підтримка SSL вебсайтів [15].

На рисунку 4.7 продемонстровано введення адреси сайту для перевірки.

На рисунку 4.8 наведено хід тестування за допомогою Xenu's Link Sleuth.

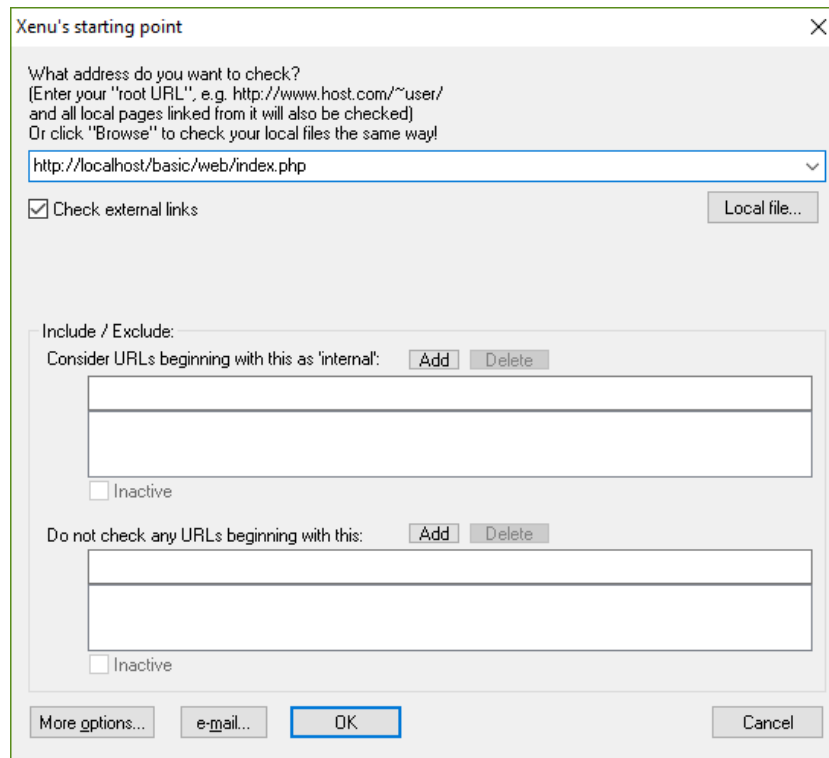


Рисунок 4.7 – Початок тестування

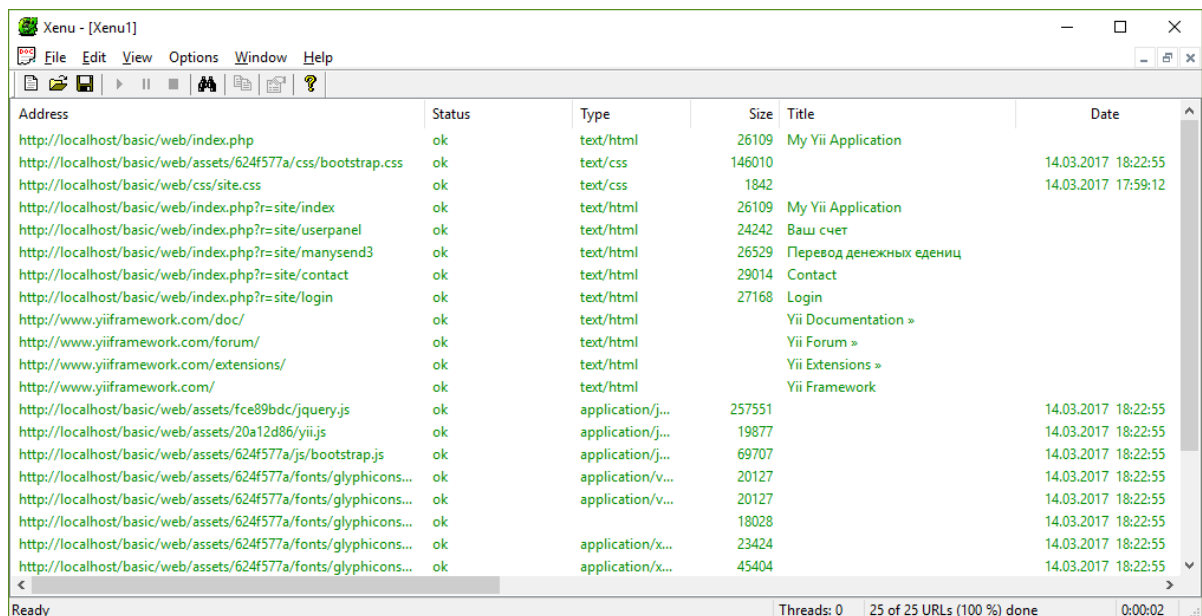


Рисунок 4.8 – Хід тестування

На рисунках 4.9 та 4.10 наведено результати тестування.

Broken links, ordered by link:

0 broken link(s) reported

[Return to Top](#)

Broken links, ordered by page:

0 broken link(s) reported

[Return to Top](#)

List of redirected URLs

[Return to Top](#)

List of valid URLs you can submit to a search engine:

<http://localhost/basic/web/index.php>
<http://localhost/basic/web/index.php?r=site%2Findex>
<http://localhost/basic/web/index.php?r=site%2Fuserpanel>
<http://localhost/basic/web/index.php?r=site%2Fmansend3>
<http://localhost/basic/web/index.php?r=site%2Fcontact>
<http://localhost/basic/web/index.php?r=site%2Flogin>
<http://www.yiiframework.com/doc/>
<http://www.yiiframework.com/forum/>
<http://www.yiiframework.com/extensions/>
<http://www.yiiframework.com/>

Рисунок 4.9 – Статистика непрацюючих посилань

Statistics for managers

Correct internal URLs, by MIME type:

MIME type	count	% count	Σ size	Σ size (KB)	% size	min size	max size	Ø size	Ø size (KB)	Ø time
text/html	6 URLs	28.57%	159171 Bytes	(155 KB)	16.88%	24242 Bytes	29014 Bytes	26528 Bytes	(25 KB)	0.333
text/css	2 URLs	9.52%	147852 Bytes	(144 KB)	15.68%	1842 Bytes	146010 Bytes	73926 Bytes	(72 KB)	
application/javascript	6 URLs	28.57%	396940 Bytes	(387 KB)	42.09%	2055 Bytes	257551 Bytes	66156 Bytes	(64 KB)	
application/vnd.ms-fontobject	2 URLs	9.52%	40254 Bytes	(39 KB)	4.27%	20127 Bytes	20127 Bytes	20127 Bytes	(19 KB)	
	1 URLs	4.76%	18028 Bytes	(17 KB)	1.91%	18028 Bytes	18028 Bytes	18028 Bytes	(17 KB)	
application/x-font-woff	1 URLs	4.76%	23424 Bytes	(22 KB)	2.48%	23424 Bytes	23424 Bytes	23424 Bytes	(22 KB)	
application/x-font-ttf	1 URLs	4.76%	45404 Bytes	(44 KB)	4.81%	45404 Bytes	45404 Bytes	45404 Bytes	(44 KB)	
image/svg+xml	1 URLs	4.76%	108738 Bytes	(106 KB)	11.53%	108738 Bytes	108738 Bytes	108738 Bytes	(106 KB)	
image/png	1 URLs	4.76%	3174 Bytes	(3 KB)	0.34%	3174 Bytes	3174 Bytes	3174 Bytes	(3 KB)	
Total	21 URLs	100.00%	942985 Bytes	(920 KB)	100.00%					

All pages, by result type:

ok	25 URLs	100.00%
Total	25 URLs	100.00%

Рисунок 4.10 – Загальна статистика результатів тестування

ВИСНОВКИ

В ході виконання даної дипломної кваліфікаційної роботи бакалавра було проведено огляд та аналіз методів розробки та підтримки ПЗ для збору коштів на реалізацію власних проєктів. Відібрані матеріали були систематизовані, проаналізовані, узагальнені та надані у першому розділі пояснювальної записки.

Для цього було виконано наступні завдання:

- розглянуто проблемну область;
- проаналізовано сучасні найбільш популярні краудфандингові платформи;
- виконано проєктування вебзастосунку;
- розроблено вебзастосунок;
- розроблено сценарій та проведено тестування.

Для програмної реалізації ПЗ для збору коштів на реалізацію власних проєктів у вигляді вебзастосунку було проаналізовано сучасні засоби для проєктування та вебзастосунків на основі виявлених особливостей та архітектури застосунку, що розробляється в рамках даної дипломної кваліфікаційної роботи бакалавра. Було виявлені необхідні засоби. Отже, для розробки практичної було використано: мову програмування PHP, фреймворк Yii; СКБД MySQL для проєктування та розробки бази даних з метою централізованого збереження інформації; вебсервер XAMPP.

В процесі виконання роботи було пройдено ряд етапів, що дозволили швидко та якісно розробити необхідну систему:

- збір та аналіз вимог;
- проєктування системи з використанням UML діаграм;
- розгортання вебсерверу;
- аналіз якості та оцінка застосунку.

В результаті виконання завдання було отримано вебзастосунок, що відповідає всім вимогам представленим в технічному завданні, зручний для

використання та виконує всі необхідні функції, для збору фінансування для реалізації проєктів. Застосунок виконує наступні функції:

- реєстрація та авторизація користувачів;
- створення індивідуального профілю, що містить в собі інформацію про користувача;
- розміщення власних проєктів для фінансування;
- створення та поповнення власного рахунку в системі;
- переведення коштів для фінансування проєктів;
- проведення внутрішньої конвертації валюти при здійсненні переводів;
- обмін повідомленнями між користувачами.

В результаті виконання завдання було отримано систему, що відповідає всім вимогам представленим в технічному завданні, зручна для використання та виконує всі необхідні функції.

Усі завдання дипломної кваліфікаційної роботи бакалавра повністю виконано.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Гвоздь М. Я. Краудфандинг як інструмент залучення коштів для фінансування стартап-проектів: аналіз зарубіжного та вітчизняного досвіду [Текст] / М. Я. Гвоздь, Ю. Г. Бондаренко, І. Я. Кулиняк // Інфраструктура ринку. - 2020. – Вип. 43. – С. 131-136.
2. Петренко В. С. Поняття краудфандингу та його роль в національній економіці [Текст] / В. С. Петренко, А. С. Карнаушенко // Таврійський науковий вісник. Серія : Економіка. – 2020. - Вип. 2. – С. 140-150.
3. Диба М. І. Глобальні тенденції та потенціал розвитку ринку краудфандингу в Україні [Текст] / М. І. Диба, Ю. О. Гернего // Економіка України. – 2020. – № 2. – С. 66-79.
4. Оксенюк К. І. Краудфандинг – сучасний інструмент фінансування [Текст] / К. І. Оксенюк // Економічні науки. Серія : Регіональна економіка. – 2017. - Вип. 14. – С. 232-238.
5. Фещур Р. В. Стан і перспективи розвитку краудфандингової діяльності [Текст] / Р. В. Фещур, О. Є. Кузьмін, О. С. Скибінський, О. З. Уголькова // Фінансово-кредитна діяльність: проблеми теорії та практики. - 2017. – Вип. 2. – С. 414-423.
6. Спільнокошт / [Електрон. ресурс]. – Режим доступу: <https://biggggidea.com/>
7. Kickstarter / [Електрон. ресурс]. – Режим доступу: <https://www.kickstarter.com/>
8. Patreon / [Електрон. ресурс] – Режим доступу: <https://www.patreon.com/>
9. Древич А. І. Реалізації програмних засобів WEB на PHP [Текст] / А. І. Древич, Р. В. Моравецький // Збірник наукових праць Інституту проблем моделювання в енергетиці ім. Г. Є. Пухова. – 2014. – Вип. 71. – С. 54-58.

10. Кравчук О. А. Практичний огляд php 7 [Текст] / О. А. Кравчук // Вісник Хмельницького національного університету. Технічні науки. – 2017. – № 3. – С. 231-236.
11. Шолом П. С. Електронний навчальний підручник на базі PHP Framework Yii2 [Текст] / П. С. Шолом, А. Ю. Котвицька, В. Ф. Самарчук // Комп'ютерно-інтегровані технології: освіта, наука, виробництво. – 2016. – № 23. – С. 112-116.
12. Проценко М. М. Аналіз фреймворків як засобів розробки Web-додатків [Текст] / М. М. Проценко // Міжнародний науковий журнал. – 2016. – № 6(1). – С. 86-89.
13. Полотай О. І. Моделювання процедур надання освітніх послуг в процесі управління проєктами інформатизації засобами мови UML (на прикладі Львівської комерційної академії) [Текст] / О. І. Полотай // Управління розвитком складних систем. – 2012. – Вип. 12. – С. 154-158.
14. Яковина В. С. Використання засобів UML для прогнозування надійності програмного забезпечення на етапі його проєктування [Текст] / В. С. Яковина, Ю. І. Парфенюк // Вісник Національного університету "Львівська політехніка". Комп'ютерні системи та мережі. – 2013. – № 773. – С. 151-156.
15. Xenu's Link Sleuth / [Електрон. ресурс]. – Режим доступу: <https://home.snafu.de/tilman/xenulink.html>.

ДОДАТОК А
Технічне завдання

Вступ

Особливостями програми мають стати можливість створення власних проєктів для фінансування різних напрямленостей. Також система буде вести внутрішню систему рахунків, які можливо поповнювати.

A.1 Підстави для розробки

Розробка має бути виконана в межах завдання на дипломну кваліфікаційну роботу, яке було поставлено за темою «Розробка програмного забезпечення для збору коштів на реалізацію власних проєктів» у відповідності з наказом № 103 від 30 березня 2021 р. за Національним університетом «Запорізька політехніка».

A.2 Призначення розробки

Програмне забезпечення, що розробляється, призначене для колективного збору фінансування на реалізацію проєктів різних напрямленостей.

A.3 Основні вимоги до програми

A.3.1 Вимоги до функціональних характеристик

Програма має надавати такі функціональні можливості:

- реєстрація та авторизація користувачів;
- створення індивідуального профілю, що містить в собі інформацію про користувача;
- розміщення власних проєктів для фінансування;
- створення та поповнення власного рахунку в системі;
- переведення коштів для фінансування проєктів;

- проведення внутрішньої конвертації валюти при здійсненні переводів;
- обмін повідомленнями між користувачами.

A.3.2 Умови експлуатації

Для експлуатації програми особливих вмінь від користувачів окрім вміння користуватися комп'ютером не висувається.

Для забезпечення роботи бази даних на сервері необхідні вміння системного адміністратора для підтримки роботи вебсервера, сервера системи керування базами даних Microsoft SQL Server.

A.3.3 Вимоги до складу та параметрів технічних засобів

Мінімальні системні вимоги та програмне забезпечення, що необхідні для функціонування серверної частини мобільного застосунку:

- серверна ОС – будь-яка з різновидів операційної системи Unix (включаючи Mac OS X) або Windows;
- вебсервер – HTTP-сервер Apache для Unix і Windows, сервер на кшталт XAMPP;
- інтерпретатор PHP5 та PHP7;
- MySQL-сервер із правами створення та вилучення користувачів БД (або база SQLite з аналогічними правами).

Для експлуатації клієнтської частини необхідні такі програмно-технічні засоби:

- ПК;
- процесор із тактовою частотою від 1,5 GHz;
- обсяг оперативної пам'яті – не менше 3GB;
- підключення до мережі Internet;

- встановлений вебглядач (браузер) останньої версії на рушії Chromium (Google Chrome), Gecko (Mozilla Firefox) або інших (Opera, Microsoft Edge).

A.4 Порядок контролю та приймання

Виділяють такі етапи в розробці програмного забезпечення:

- збирання та аналіз вимог. Проводиться збір вимог до програмного забезпечення, їх систематизація, документування, аналіз, виявлення протиріч, неповноти, рішення конфліктів в процесі розробки програмного забезпечення;

- розробка ТЗ. На основі раніше зібраних вимог формується вихідний документ (ТЗ) на проєктування програмного забезпечення. Описується всі вимоги до створюваного продукту;

- створення логотипу та дизайн основних сторінок;

- налаштування БД;

- розробка структури. Дозволяє оцінити інформативність та компоновку елементів системи на початковому етапі розробки;

- підключення адміністративного інтерфейсу. Має містити повне управління настройками системи, окремих модулів та контентом сайту;

- робота над програмним кодом. Розробка основного функціоналу системи;

- наповнення та тестування. Професійні тестувальники знаходять «баги» у системі та фіксують їх. Заповнюється базовий «контент», щоб сайт мав привабливіший вигляд для перших користувачів;

- розміщення та відкритий доступ, технічна підтримка. Соціальна мережа розміщується на сайті унтернет-магазину та стає доступною для всіх користувачів;

- оформлення документації.

A.5 Порядок контролю та приймання системи

Приймання системи здійснюється відповідно до складу та змісту робіт, наведених у Технічному завданні, і відповідно до умов контракту.

Звіт про проведену роботу направляється Замовнику на паперовому носії в 2-х екземплярах, підписаних належним чином, а також на електронному носії.

Вимоги до оформлення проміжних звітів і підсумкового звіту узгоджуються з Виконавцем при підписанні контракту.

Одночасно з підсумковим звітом про виконання робіт Виконавець надає Замовнику підписаний зі сторони Виконавця Акт здачі-приймання виконаної роботи.

ДОДАТОК Б
Текст програми

```

<?php

namespace app\models;

use Yii;
use yii\base\NotSupportedException;
use yii\behaviors\TimestampBehavior;
use yii\db\ActiveRecord;
use yii\web\IdentityInterface;

/**
 * User model
 *
 * @property integer $id
 * @property string $username
 * @property string $password_hash
 * @property string $password_reset_token
 * @property string $email
 * @property string $auth_key
 * @property integer $status
 * @property integer $role
 * @property float available_funds
 * @property integer $created_at
 * @property integer $updated_at
 * @property string $password write-only password
 * @property ClientInfo $info
 */
class User extends ActiveRecord implements IdentityInterface
{
    const STATUS_DELETED = 0;
    const STATUS_ACTIVE = 10;

    /**
     * @inheritdoc
     */
    public static function tableName()
    {
        return '{{%user}}';
    }

    /**

```

```

    * @inheritdoc
    */
    public function behaviors()
    {
        return [
            TimestampBehavior::className(),
        ];
    }

    /**
     * @inheritdoc
     */
    public function rules()
    {
        return [
            ['status', 'default', 'value' =>
self::STATUS_ACTIVE],
            ['status', 'in', 'range' => [self::STATUS_ACTIVE,
self::STATUS_DELETED]],
        ];
    }

    /**
     * @inheritdoc
     */
    public static function findIdentity($id)
    {
        return static::findOne(['id' => $id, 'status' =>
self::STATUS_ACTIVE]);
    }

    /**
     * @inheritdoc
     */
    public static function findIdentityByAccessToken($token,
$type = null)
    {
        throw new
NotSupportedException('"findIdentityByAccessToken" is not
implemented.');
```

```

/**
 * Finds user by username
 *
 * @param string $username
 * @return static|null
 */
public static function findByUsername($username)
{
    return static::findOne(['username' => $username,
'status' => self::STATUS_ACTIVE]);
}

/**
 * @inheritdoc
 */
public function getId()
{
    return $this->getPrimaryKey();
}

/**
 * @inheritdoc
 */
public function getAuthKey()
{
    return $this->auth_key;
}

/**
 * @inheritdoc
 */
public function validateAuthKey($authKey)
{
    return $this->getAuthKey() === $authKey;
}
}
<?php

namespace app\models;

use Yii;

```

```

use app\models\ArchivePayment;
use app\models\ArchivePaymentSearch;
use yii\web\Controller;
use yii\web\NotFoundHttpException;
use yii\filters\VerbFilter;

/**
 * ArchivePaymentController implements the CRUD actions for
ArchivePayment model.
 */
class ArchivePaymentController extends Controller
{
    /**
     * @inheritdoc
     */
    public function behaviors()
    {
        return [
            'verbs' => [
                'class' => VerbFilter::className(),
                'actions' => [
                    'delete' => ['POST'],
                ],
            ],
        ];
    }

    /**
     * Lists all ArchivePayment models.
     * @return mixed
     */
    public function actionIndex()
    {
        $searchModel = new ArchivePaymentSearch();
        $dataProvider = $searchModel->search(Yii::$app->request-
>queryParams);

        return $this->render('index', [
            'searchModel' => $searchModel,
            'dataProvider' => $dataProvider,
        ]);
    }

```

```

    }

    /**
     * Displays a single ArchivePayment model.
     * @param integer $id
     * @return mixed
     */
    public function actionView($id)
    {
        return $this->render('view', [
            'model' => $this->findModel($id),
        ]);
    }

    /**
     * Creates a new ArchivePayment model.
     * If creation is successful, the browser will be redirected
to the 'view' page.
     * @return mixed
     */
    public function actionCreate()
    {
        $model = new ArchivePayment();

        if ($model->load(Yii::$app->request->post()) && $model->
>save()) {
            return $this->redirect(['view', 'id' => $model->
>id]);
        } else {
            return $this->render('create', [
                'model' => $model,
            ]);
        }
    }

    /**
     * Updates an existing ArchivePayment model.
     * If update is successful, the browser will be redirected
to the 'view' page.
     * @param integer $id
     * @return mixed

```

```

        */
        public function actionUpdate($id)
        {
            $model = $this->findModel($id);

            if ($model->load(Yii::$app->request->post()) && $model-
>save()) {
                return $this->redirect(['view', 'id' => $model-
>id]);
            } else {
                return $this->render('update', [
                    'model' => $model,
                ]);
            }
        }

        /**
         * Deletes an existing ArchivePayment model.
         * If deletion is successful, the browser will be redirected
to the 'index' page.
         * @param integer $id
         * @return mixed
         */
        public function actionDelete($id)
        {
            $this->findModel($id)->delete();

            return $this->redirect(['index']);
        }

        /**
         * Finds the ArchivePayment model based on its primary key
value.
         * If the model is not found, a 404 HTTP exception will be
thrown.
         * @param integer $id
         * @return ArchivePayment the loaded model
         * @throws NotFoundHttpException if the model cannot be
found
         */
        protected function findModel($id)

```

```

        {
            if (($model = ArchivePayment::findOne($id)) !== null) {
                return $model;
            } else {
                throw new NotFoundException('The requested page
does not exist.');
```

```

            }
        }
    }

    /**
     * Validates password
     *
     * @param string $password password to validate
     * @return bool if password provided is valid for current
user
     */
    public function validatePassword($password)
    {
        return Yii::$app->security->validatePassword($password,
$this->password_hash);
    }

    /**
     * Generates password hash from password and sets it to the
model
     *
     * @param string $password
     */
    public function setPassword($password)
    {
        $this->password_hash = Yii::$app->security-
>generatePasswordHash($password);
    }

    /**
     * Generates "remember me" authentication key
     */
    public function generateAuthKey()
    {
        $this->auth_key = Yii::$app->security-
>generateRandomString();
    }
}

```

```

    }

    public function getInfo()
    {
        return $this->hasOne(ClientInfo::className(), ['account'
=> 'id']));
    }

}
<?php

```

```
namespace app\models;
```

```
use Yii;
```

```

/**
 * This is the model class for table "archivepayment".
 *
 * @property integer $id
 * @property integer $sender
 * @property integer $recipient
 * @property double $amount
 * @property integer $currency
 */

```

```
class Payments extends \yii\db\ActiveRecord
```

```

{
    /**
     * @inheritdoc
     */
    public static function tableName()
    {
        return 'archivepayment';
    }

```

```

    /**
     * @inheritdoc
     */
    public function rules()
    {
        return [
            [['recipient', 'amount', 'currency'], 'required'],

```

```

        [['recipient', 'currency'], 'integer'],
        [['amount'], 'number'],
    ];

}

/**
 * @inheritdoc
 */
public function attributeLabels()
{
    return [
        'id' => 'ID',
        'sender' => 'Sender',
        'recipient' => 'Recipient',
        'amount' => 'Amount',
        'currency' => 'Currency',
    ];
}

public function whoisthis(){

    $this->sender = Yii::$app->user->identity->id;
    return true;
}

public function getManycurrency()
{
    return $this->hasOne(Archivepayment::className(), ['id'
=> 'currency']);
}

}
<?php

namespace app\models;

use Yii;
<?php

```

```

namespace app\models;

use Yii;
use app\models\ClientInfo;
use app\models\ClientInfoSearch;
use yii\web\Controller;
use yii\web\NotFoundHttpException;
use yii\filters\VerbFilter;

/**
 * ClientInfoController implements the CRUD actions for
ClientInfo model.
 */
class ClientInfoController extends Controller
{
    /**
     * @inheritdoc
     */
    public function behaviors()
    {
        return [
            'verbs' => [
                'class' => VerbFilter::className(),
                'actions' => [
                    'delete' => ['POST'],
                ],
            ],
        ];
    }

    /**
     * Lists all ClientInfo models.
     * @return mixed
     */
    public function actionIndex()
    {
        $searchModel = new ClientInfoSearch();
        $dataProvider = $searchModel->search(Yii::$app->request-
>queryParams);

```

```

        return $this->render('index', [
            'searchModel' => $searchModel,
            'dataProvider' => $dataProvider,
        ]);
    }

    /**
     * Displays a single ClientInfo model.
     * @param integer $id
     * @return mixed
     */
    public function actionView($id)
    {
        return $this->render('view', [
            'model' => $this->findModel($id),
        ]);
    }

    /**
     * Creates a new ClientInfo model.
     * If creation is successful, the browser will be redirected
to the 'view' page.
     * @return mixed
     */
    public function actionCreate()
    {
        $model = new ClientInfo();

        if ($model->load(Yii::$app->request->post()) && $model->
>save()) {
            return $this->redirect(['view', 'id' => $model->
>id]);
        } else {
            return $this->render('create', [
                'model' => $model,
            ]);
        }
    }

    /**
     * Updates an existing ClientInfo model.

```

```

        * If update is successful, the browser will be redirected
to the 'view' page.
        * @param integer $id
        * @return mixed
        */
public function actionUpdate($id)
{
    $model = $this->findModel($id);

    if ($model->load(Yii::$app->request->post()) && $model-
>save()) {
        return $this->redirect(['view', 'id' => $model-
>id]);
    } else {
        return $this->render('update', [
            'model' => $model,
        ]);
    }
}

/**
 * Deletes an existing ClientInfo model.
 * If deletion is successful, the browser will be redirected
to the 'index' page.
 * @param integer $id
 * @return mixed
 */
public function actionDelete($id)
{
    $this->findModel($id)->delete();

    return $this->redirect(['index']);
}

/**
 * Finds the ClientInfo model based on its primary key
value.
 * If the model is not found, a 404 HTTP exception will be
thrown.
 * @param integer $id
 * @return ClientInfo the loaded model

```

```

        * @throws NotFoundHttpException if the model cannot be
found
        */
        protected function findModel($id)
        {
            if (($model = ClientInfo::findOne($id)) !== null) {
                return $model;
            } else {
                throw new NotFoundHttpException('The requested page
does not exist.');
```

```

            }
        }
    }
    class Manycurrency extends \yii\db\ActiveRecord
    {
        /**
         * @inheritdoc
         */
        public static function tableName()
        {
            return 'currencyrates';
        }

        /**
         * @inheritdoc
         */
        public function rules()
        {
            return [
                [['currency', 'indexhigh'], 'required'],
                [['currency'], 'string'],
                [['indexhigh'], 'number'],
            ];
        }

        /**
         * @inheritdoc
         */
        public function attributeLabels()
        {
            return [

```

```

        'id' => 'ID',
        'currency' => 'Currency',
        'indexhight' => 'Indexhight',
    ];
}

/**
 * @return \yii\db\ActiveQuery
 */
public function getArchivepayments()
{
    return $this->hasMany(Archivepayment::className(),
['currency' => 'id']);
}

public function getClientInfos()
{
    return $this->hasOne(ClientInfo::className(),
['currency' => 'id']);
}

}
<?php

namespace app\models;

use Yii;

/**
 * This is the model class for table "archive_payment".
 *
 * @property integer $id
 * @property integer $sender
 * @property integer $recipient
 * @property double $amount
 * @property integer $currency
 *
 * @property CurrencyRate $currency0
 */
class Archivepayments extends \yii\db\ActiveRecord
{

```

```

/**
 * @inheritdoc
 */
public static function tableName()
{
    return 'archive_payment';
}

/**
 * @inheritdoc
 */
public function rules()
{
    return [
        // [['sender', 'recipient', 'amount'], 'required'],
        // [['sender', 'recipient', 'currency'], 'integer'],
        // [['amount'], 'number'],
        // [['currency'], 'exist', 'skipOnError' => true,
'targetClass' => Currencyrates::className(), 'targetAttribute' =>
['currency' => 'id']],
    ];
}

/**
 * @inheritdoc
 */
public function attributeLabels()
{
    return [
        'id' => 'ID',
        'sender' => 'Sender',
        'recipient' => 'Recipient',
        'amount' => 'Amount',
        'currency' => 'Currency',
    ];
}

public function whoisthis(){

    $this->sender = Yii::$app->user->identity->id;

```

```

        // $this->currency = 1;
        return true;
    }

    /**
     * @return \yii\db\ActiveQuery
     */
    /* public function getCurrency0()
    {
        return $this->hasOne(Currencyrates::className(), ['id'
=> 'currency']);
    } */
}
<?php

namespace app\models;

use Yii;

/**
 * This is the model class for table "archivepayment".
 *
 * @property integer $id
 * @property integer $sender
 * @property integer $recipient
 * @property double $amount
 * @property integer $currency
 */
class Archivepayment extends \yii\db\ActiveRecord
{
    /**
     * @inheritdoc
     */
    public static function tableName()
    {
        return 'archivepayment';
    }

    /**
     * @inheritdoc

```

```

        */
    public function rules()
    {
        return [
            //  [[ 'recipient', 'amount', 'currency'],
'required'],
            //  [[ 'recipient', 'currency'], 'integer'],
            //  [['amount'], 'number'],
        ];
    }

    /**
     * @inheritdoc
     */
    public function attributeLabels()
    {
        return [
            'id' => 'ID',
            'sender' => 'Sender',
            'recipient' => 'Recipient',
            'amount' => 'Amount',
            'currency' => 'Currency',
        ];
    }

    public function whoisthis(){

        $this->sender = Yii::$app->user->identity->id;
        // $this->currency = 1;
        return true;
    }

}

}

```

ДОДАТОК В
Слайди презентації

Розробка програмного забезпечення для збору коштів на реалізацію власних проєктів

Виконав	Клименко Є.В. ст.гр. КНТз-118_2
Керівник	Зайко Т.А. к.т.н., доцент
Рецензент	Гофман Є.О. к.т.н., доцент

Рисунок В.1 – Слайд № 1

Об'єкт, предмет та мета роботи

Об'єкт роботи – процеси збору колективного фінансування для реалізації авторських проєктів.

Предмет роботи – програмне забезпечення збору коштів на реалізацію проєктів.

Мета роботи – розробити програмне забезпечення для збору коштів на реалізацію власних проєктів.

Рисунок В.2 – Слайд № 2

Постановка завдання

Застосунок виконує наступні функції:

- реєстрація та авторизація користувачів;
- створення індивідуального профілю, що містить в собі інформацію про користувача;
- розміщення власних проєктів для фінансування;
- створення та поповнення власного рахунку в системі;
- переведення коштів для фінансування проєктів;
- проведення внутрішньої конвертації валюти при здійсненні переводів;
- обмін повідомленнями між користувачами.

Рисунок В.3 – Слайд № 3

Порівняння існуючих аналогів

	Спільнокошт	Kickstarter	Patreon	Підтримка
Типи проєктів для фінансування	Громадські проєкти	Розважальні та технологічні проєкти	Творчі проєкти	Громадські, розважальні та технологічні проєкти
Відкритість проєктів	Середній рівень	Середній рівень	Високий рівень	Високий рівень
Захист даних платників	Стороння система	Стороння система	Стороння система	Власна система
Можливість безкоштовної підтримки	Відсутня	Відсутня	Присутня	Присутня

Рисунок В.4 – Слайд № 4

Порівняння мов програмування

	PHP	Python	Java (JSP)
Стресостійкість застосунків	++	+	++
Масштабованість застосунків	++	+	+
Платформи для спрощення розробки	++	+	+
Розповсюдженість матеріалів	++	++	+

Рисунок В.5 – Слайд № 5

Порівняння фреймворків для розробки

	Yii	Laravel
Простота розробки представлень	++	+
Плагіни та розширення	+	++
Спільнота	++	++
Швидкість роботи розробленого застосунку	++	±

Рисунок В.6 – Слайд № 6

Архітектуру веб-застосунку

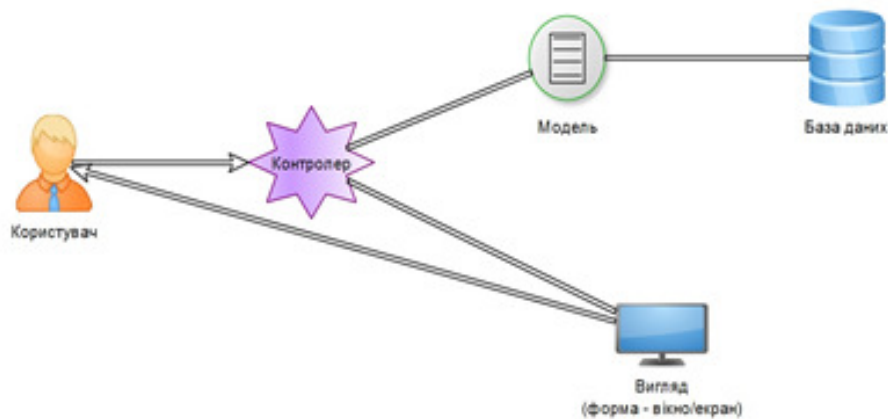


Рисунок В.7 – Слайд № 7

Логічне проектування системи



Рисунок В.8 – Слайд № 8

```

sequenceDiagram
    participant User as Сервис: Користувач
    participant Project as 00025: Проект Амані
    participant Account as 4566963256740951: Рахунок
    participant Amani as Амані: Користувач

    User->>Project: 1. Створення нового платежу
    activate Project
    Project->>Account: 2. Перевірка балансу рахунку
    activate Account
    Account-->>Project: 3. Введення даних про баланс рахунку
    deactivate Account
    Project->>User: 4. Введення дати та балансу платежу
    deactivate Project
    activate User
    User->>Project: 5. Створення платежу в рахунок
    activate Project
    Project->>Account: 6. Оплата платежу
    activate Account
    Account-->>Project: 7. Оплата платежу
    deactivate Account
    Project->>User: 8. Оплата платежу
    deactivate Project
    activate User
    User->>Project: 9. Оплата платежу
    activate Project
    Project->>Account: 10. Оплата платежу
    activate Account
    Account-->>Project: 11. Оплата платежу
    deactivate Account
    Project->>User: 12. Оплата платежу
    deactivate Project
    activate User
    User->>Project: 13. Оплата платежу
    activate Project
    Project->>Account: 14. Оплата платежу
    activate Account
    Account-->>Project: 15. Оплата платежу
    deactivate Account
    Project->>User: 16. Оплата платежу
    deactivate Project
    deactivate User
    
```

10

Рисунок В.10 – Слайд № 10

Висновки

Для програмної реалізації ПЗ для збору коштів на реалізацію власних проєктів у вигляді веб-застосунку було проаналізовано сучасні засоби для проектування та веб-застосунків на основі виявлених особливостей та архітектури застосунку, що розробляється в рамках даної дипломної кваліфікаційної роботи бакалавра. Було виявлені необхідні засоби. Отже, для розробки практичної було використано: мову програмування PHP, фреймворк Yii, СКБД MySQL для проектування та розробки бази даних з метою централізованого збереження інформації; веб-сервер XAMPP.

В результаті виконання завдання було отримано веб-застосунок, що відповідає всім вимогам представленим в технічному завданні, зручний для використання та виконує всі необхідні функції, для збору фінансування для реалізації проєктів. Застосунок виконує наступні функції:

- реєстрація та авторизація користувачів;
- створення індивідуального профілю, що містить в собі інформацію про користувача;
- розміщення власних проєктів для фінансування;
- створення та поповнення власного рахунку в системі;
- переведення коштів для фінансування проєктів;
- проведення внутрішньої конвертації валюти при здійсненні переводів;
- обмін повідомленнями між користувачами.

Усі завдання випускної дипломної роботи бакалавра повністю виконано.

Рисунок В.11 – Слайд № 11

Дякую за увагу

Рисунок В.12 – Слайд № 12