

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування інституту, назва факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

магістра

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДІВ
ПРОГНОЗУВАННЯ ТРИВАЛОСТІ КАМПАНІЇ ДЛЯ ЗБОРУ КОШТІВ НА
ФІНАНСУВАННЯ ПРОЄКТІВ
RESEARCH AND SOFTWARE IMPLEMENTATION OF METHODS FOR
PREDICTING THE DURATION OF THE FINANCIAL PROJECT
FUNDRAISING CAMPAIGN

Виконав: студент(ка) 2 курсу, групи КНТ-121м
Спеціальності 121 Інженерія програмного
забезпечення

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

Дмитрієв І.О.

(прізвище та ініціали)

Керівник Скрупський С.Ю.

(прізвище та ініціали)

Рецензент Голуб Т.В.

(прізвище та ініціали)

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

Інститут, факультет ФКНТ
 Кафедра програмних засобів
 Ступінь вищої освіти магістр
 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
 “ ” 2022 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Дмитрієва Іллі Олеговича

(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження та програмна реалізація методів прогнозування тривалості кампанії для збору коштів на фінансування проєктів
Research and Software Implementation of Methods for Predicting the Duration of the Financial Project Fundraising Campaign

керівник проєкту (роботи) Скрупський Степан Юрійович, к.т.н., доцент,
 (прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від “4” листопада 2022 року № 363

2. Строк подання студентом проєкту (роботи) грудень 2022 року

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

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз проблеми та постановка завдань дослідження. 2. Матеріали і методи. 3. Проєктування програмного забезпечення. 4. Основні рішення щодо реалізації компонентів системи. 5. 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
1-5 Основна частина	Скрупський С.Ю., доцент		
Нормоконтролер	Дейнега Л.Ю., ст. викладач		

7. Дата видачі завдання “1” вересня 2022 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 та дослідження предметної області	2-5 тижні	Розділи 1, 2
3	Розробка архітектури програми	6 тиждень	Розділ 3
4	Розробка програми	7-9 тижні	Розділ 4
5	Тестування та експериментальне дослідження програми	10-11 тижні	Розділ 5
6	Оформлення пояснювальної записки та документів до неї. Нормоконтроль та рецензування	12 тиждень	Додатки
7	Захист роботи	13 тиждень	

Студент(ка)

(підпис) Дмитрієв І.О.
(прізвище та ініціали)

Керівник проєкту (роботи)

(підпис) Скрупський С.Ю.
(прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи магістра:
97 с., 30 рис., 4 табл., 3 дод., 18 джерел.

ФІНАНСУВАННЯ ПРОЄКТУ, ЗБІР КОШТІВ, ВЕБДОДАТОК,
ДЕРЕВА УХВАЛЕННЯ РІШЕНЬ, ЗАДАЧА ПРОГНОЗУВАННЯ.

Об'єкт роботи – процес збору коштів на фінансування проєктів.
Предмет роботи – методи прогнозування тривалості кампанії для збору коштів на фінансування проєктів.

Мета роботи – розробити програмне забезпечення збору коштів на фінансування проєктів, що підтримує прийняття рішень автором проєкту щодо встановлення основних параметрів кампанії зі збору коштів.

Матеріали, методи та технічні засоби: мова Python, бібліотека scikit-learn, фреймворк веброзробки Django, система керування базами даних MySQL.

Результати. Наукова новизна роботи сформована в методі прогнозування тривалості кампанії для збору коштів на фінансування проєктів, який виконує визначення строку, необхідного на збір певної суми коштів на фінансування конкретного проєкту, до початку кампанії, використовуючи дерева ухвалення рішень на основі алгоритму CART, що дозволяє збалансувати встановлені суму коштів кампанії та тривалість такої кампанії, щоб в підсумку вона з більшою вірогідністю завершилась успішно.

Висновки. Представлене програмне забезпечення дозволяє організовувати кампанії зі збору коштів на фінансування проєктів, керувати цими кампаніями, шукати проєкти, брати участь у фінансуванні проєктів з отриманням різного роду подарунків, матеріалів.

Галузь використання – збір коштів для виконання іноваційних, творчих, неприбуткових проєктів.

ABSTRACT

Explanatory note to the diploma qualifying work of the master: 97 pages, 30 figures, 4 tables, 3 appendixes, 18 sources.

PROJECT FINANCING, FUNDRAISING, WEB APPLICATION, DECISION TREES, FORECASTING PROBLEM.

The object of the work is the process of financial project fundraising. The subject of the work is methods for predicting the duration of the financial project fundraising campaign.

The purpose of the work is to develop fundraising software for project financing, which supports decision making by the project author regarding the establishment of the main parameters of the fundraising campaign.

Materials, methods and technical means: Python language, scikit-learn library, Django web development framework, MySQL database management system.

The results. The scientific novelty of the work is presented in the method methods for predicting the duration of the financial project fundraising campaign, which performs the determination of the period required to collect a certain amount of funds for the financing of a specific project before the start of the campaign, using decision trees based on the CART algorithm. This method allows balancing the set amount of campaign funds and the duration of such a campaign, so that in the end it is more likely to end successfully.

Conclusions. The presented software allows users to organize fundraising campaigns for project financing, manage these campaigns, search for projects, participate in project financing with the receipt of various gifts and materials.

The field of use is the fundraising for the implementation of innovative, creative, non-profit projects.

ЗМІСТ

	С.
Перелік скорочень та умовних познач.....	8
Вступ.....	9
1 Аналіз проблеми та постановка завдань дослідження	10
1.1 Загальний стан проблеми збору коштів на фінансування проєктів.....	10
1.2 Огляд програмного забезпечення збору коштів на фінансування проєктів.....	11
1.3 Висновки за розділом 1	15
2 Матеріали і методи.....	16
2.1 Задача прогнозування тривалості кампанії для збору коштів на фінансування проєктів	16
2.2 Дерева ухвалення рішень для розв’язання задачі прогнозування	17
2.3 Метод прогнозування тривалості кампанії для збору коштів на фінансування проєктів	20
2.4 Висновки за розділом 2	22
3 Проєктування програмного забезпечення	23
3.1 Вибір рішень для розробки програмного забезпечення.....	23
3.2 Визначення функціональних вимог до програми.....	24
3.3 Визначення структури бази даних	27
3.4 Висновки за розділом 3	43
4 Основні рішення щодо реалізації компонентів системи.....	44
4.1 Порядок роботи над кампанією зі збору коштів для фінансування проєкту.....	44
4.2 Основні реалізовані класи	45
4.3 Висновки за розділом 4	51
5 Експлуатація, тестування та експериментальне дослідження програми	52
5.1 Опис роботи з програмою	52
5.2 Експериментальне дослідження	56
5.3 Висновки за розділом 5	59

Висновки	60
Перелік джерел посилання	61
Додаток А Технічне завдання	63
Додаток Б Текст програми	68
Додаток В Слайди презентації	89

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

CART – Classification And Regression Tree;

ID3 – iterative dichotomiser 3;

TF-IDF – Term Frequency – Inverse Document Frequency;

БД – база даних.

ВСТУП

В Україні кампанії зі збору коштів в умовах воєнного стану стали широко розповсюдженими. Кошти часом збираються певним фондом, який потім вирішує, куди їх витратити, а часом на певну задачу, розв'язання якої цілісно можна розглядати як певний проєкт. Звичайно, що проєкти, на які збираються кошти, можуть бути не тільки воєнними, але і такими, що представляють будь-які потенційно цікаві ідеї. Це можуть бути нові пристрої, нові технології, нові ініціативи людей. Найважливішим при цьому для авторів проєкту є те, щоб знайти аудиторію, яка готова сплатити кошти на фінансування такого проєкту. Для цього потрібні різного роду механізми, які дозволять авторам знаходити свою аудиторію, а користувачам знаходити проєкти, які їх можуть зацікавити.

Для підтримки пов'язаних з цим процесів і необхідне програмне забезпечення, що розробляється. За наявності інструментів прогнозування програма в підсумку полегшить роботу власників цих проєктів, адже якщо сума, яку необхідно зібрати на фінансування проєкту, може бути обчислена за витратами на виконання проєкту, то тривалість виконання збору може бути в кращому випадку обмежена певною датою, з якої точно має початися виконання проєкту, але з іншого боку через це може бути підготовлено декілька варіантів виконання кампанії зі збору коштів. За наявності механізму прогнозування тривалості кампанії можна буде обрати найкращий варіант з доступних.

Тому мета роботи полягала в тому, щоб розробити програмне забезпечення збору коштів на фінансування проєктів, що підтримує прийняття рішень автором проєкту щодо встановлення основних параметрів кампанії зі збору коштів.

1 АНАЛІЗ ПРОБЛЕМИ ТА ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ

1.1 Загальний стан проблеми збору коштів на фінансування проєктів

Модель фінансування – системний та інституційний підхід до створення надійної бази доходів для підтримки основних програм і послуг організації [1].

Хоча корпоративний і трастовий збір коштів є порівняно недавнім явищем, некомерційні організації та фандрайзери існують з самого початку писемної історії: століттями завдання збору коштів від окремих осіб в основному виконувалося Церквою, служителі якої використовували багато з тих самих методів збору коштів, що широко використовуються й сьогодні, проте у двадцятому сторіччі було багато нововведень у способах застосування цих методів, використанні засобів масової інформації, а прогрес у інформаційних технологіях остаточно оформив методи збору коштів у тому вигляді, у якому сучасні люди до нього звикли [2].

До основних методів збору коштів належать: пряма поштова розсилка, події, онлайн-пожертви, обходи домівок, телефонні звернення, електронна пошта, краудфандинг, партнерство разом зі спонсорством і грантами, попарний збір коштів [3]-[5].

Одним із найпоширеніших видів збору коштів на фінансування проєктів є спільний збір коштів (краудфандинг), який використовують некомерційні організації: цей тип збору коштів дозволяє їм охопити своїх прихильників і отримати максимальну користь від пожертв, некомерційні організації можуть отримати вигоду від краудфандингу, оскільки це дозволяє їм охопити потенційних донорів, які інакше б про них не знали, однак створити успішну кампанію може бути складно, оскільки для цього потрібен належний підхід, щоб охопити людей, які потенційно готові профінансувати проєкт у певній його частині [1]-[5].

1.2 Огляд програмного забезпечення збору коштів на фінансування проєктів

Проведений огляд вказує на те, що зараз існує одразу декілька дуже відомих і популярних програмних засобів збору коштів для фінансування проєктів. Деякі з них мають певні особливості, концентруючись на окремих областях, деякі мають загальний механізм, який стандартний для всіх сучасних кампаній зі збору коштів на фінансування проєктів. Серед всіх основних відомих рішень було відібрано декілька для подальшого аналізу.

Patreon (рис. 1.1) має найактуальніше використання для людей, які створюють контент (художники, письменники, журналісти), і на даний момент відомо, що було зібрано і виплачено понад 200 000 проєктам понад 2 мільярди доларів [6]-[7].

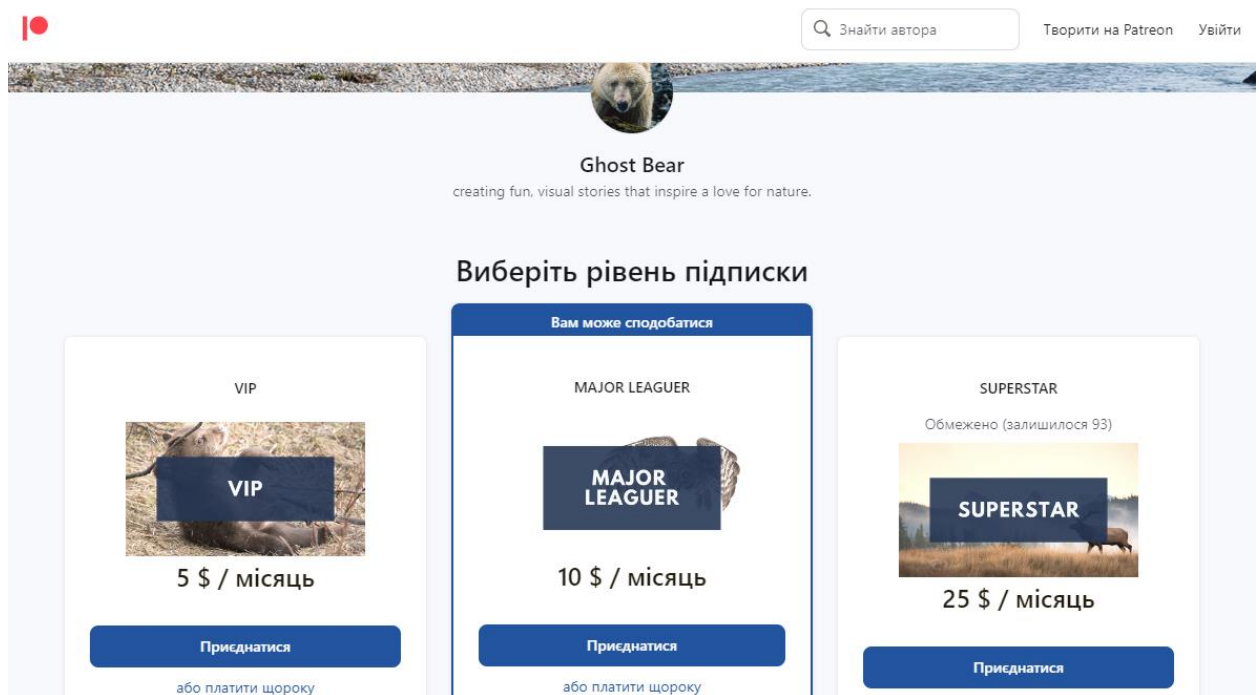


Рисунок 1.1 – Інтерфейс Patreon

Поціновувачі контенту, створеного авторами платформи, перераховують кошти на основі платних підписок. Тобто використовується формат не одноразового перерахування будь-якої суми, а саме оформлення

підписки. Пропонується авторам різний варіант роботи, який включає різні відсотки, які отримує платформа з платежів (від 5 % до 12% плюс комісії за обробку платежів).

Платформа також організує підтримку авторам, надаючи доступ до освітньої платформи, щоб навчитися керувати власними проєктами, мобільний додаток на додачу до вебсайту та інструменти, які допомагають отримати кошти, відстежувати прогрес і спілкуватися безпосередньо з користувачами, які оформили платну підписку [7].

Користувачі, які оформили підписку, отримують доступ фактично до контенту, створеного автором або проєктом, на який було оформлено відписку. Кожна підписка має свої умови, тобто за різного варіанту надається доступ до різної кількості, типу матеріалів.

FundRazr (рис. 1.2) є програмним засобом, який також використовується для збору коштів на фінансування проєктів: на даний момент зібрано 212 мільйонів доларів, надаються інструменти для інтеграції програмного забезпечення, мобільний додаток [8].

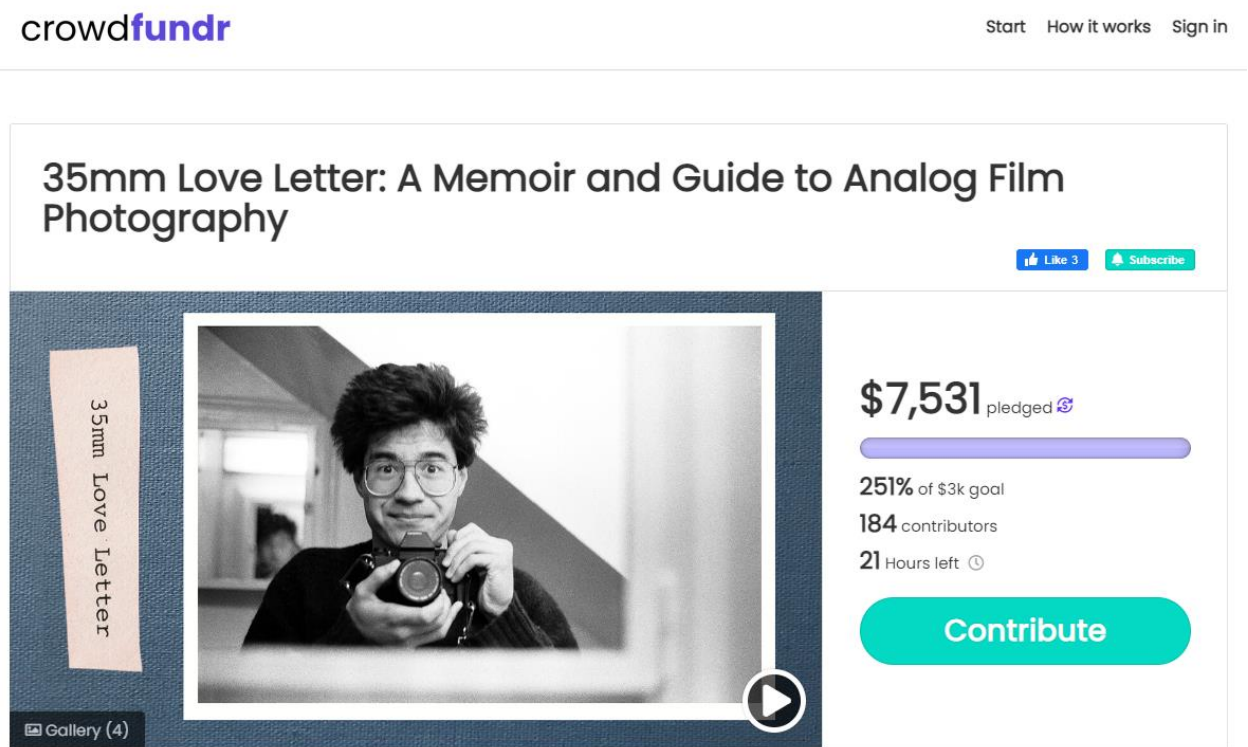


Рисунок 1.2 – Інтерфейс FundRazr

Платформа підтримує роботу в дуже різних сферах, включаючи митців, некомерційні організації, при чому одним з варіантів роботи є базовий безкоштовний варіант, коли автор не сплачує ніяких коштів, але окрім нього є стандартний і професійний варіант, коли кошти сплачуються за користування платформою. У підсумку такий підхід дозволяє знизити витрати і за безкоштовного варіанту невеликі зібрані суми можуть більш доцільно розподілятися на виконання проєкту. При цьому добровільно користувачі, які надають кошти на проєкт, за такого варіанту можуть надати певні кошти і самій платформі, тобто такий варіант роботи також слід вважати взаємовигідним.

Щодо логіки організації збору коштів слід виділити:

- стандартний план стягує комісію за платформу з автора проєкту, при чому це 5 %, які сплачуються безпосередньо автором, а після цього комісія за обробку платежу складає 2,9 % і 0,3 доларів США;
- тільки професійний план дає змогу контролювати, хто сплачує комісію за платформу (сам автор чи інвестори);
- окрім варіанту збору за принципом отримання всієї суми або жодної, доступний також варіант, коли платежі переводяться одразу при зарахуванні, надаючи таким чином доступ до коштів і їх використання одразу [7].

Kickstarter (рис. 1.3) є один із найбільших вебсайтів для збору коштів на проєкти творчих людей, який допоміг зібрати понад 6,2 мільярда доларів США, виконавши фінансування понад 200 000 проєктів з застосуванням фінансових засобів понад 20 мільйонів: платформа має модель збору коштів, яку можна охарактеризувати за принципом «все або нічого», тобто якщо буде успішно зібрано потрібну суму, тільки тоді буде проведено платіж і стягнуто суму з платника, окрім того має комісію в розмірі 5% від загальної суми зібраних коштів разом з комісією за обробку платежу (0,2 доларів США та 3 % від кожної застави) [7], [9].

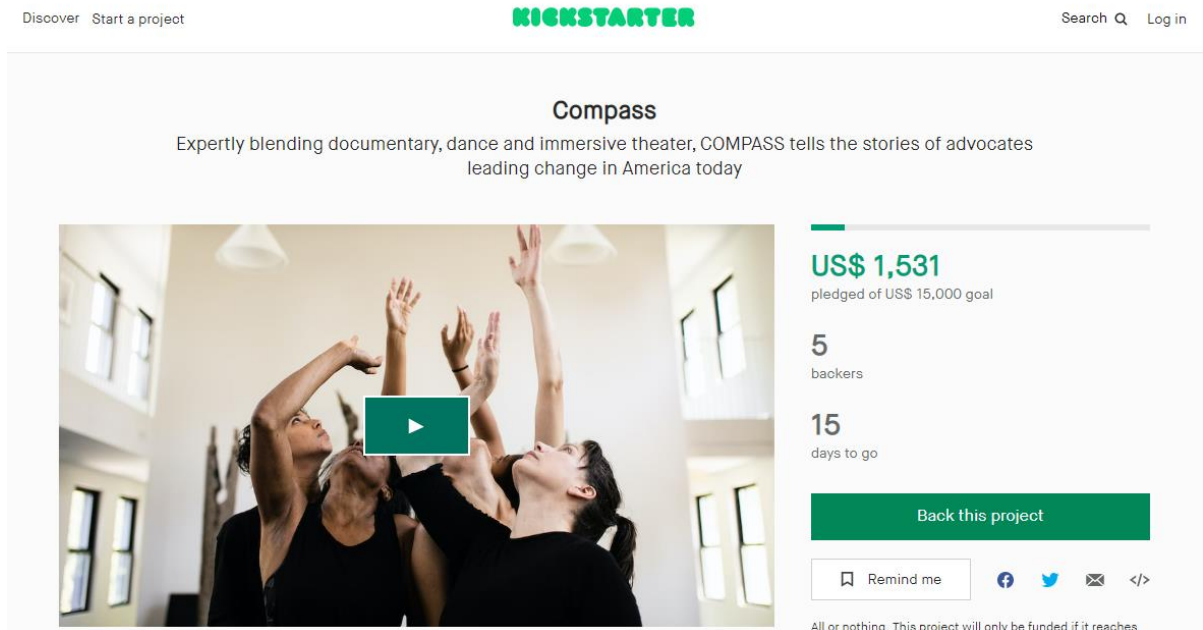


Рисунок 1.3 – Інтерфейс Kickstarter

Kickstarter дозволяє не тільки внести бажану суму платежем на рахунок сервісу, але і контролювати варіанти подарунків за такі платежі. Визначаються декілька варіантів на проєкт і потрібна для цього сума (рис. 1.4). В залежності від сплаченої суми використовується той чи інший подарунок.

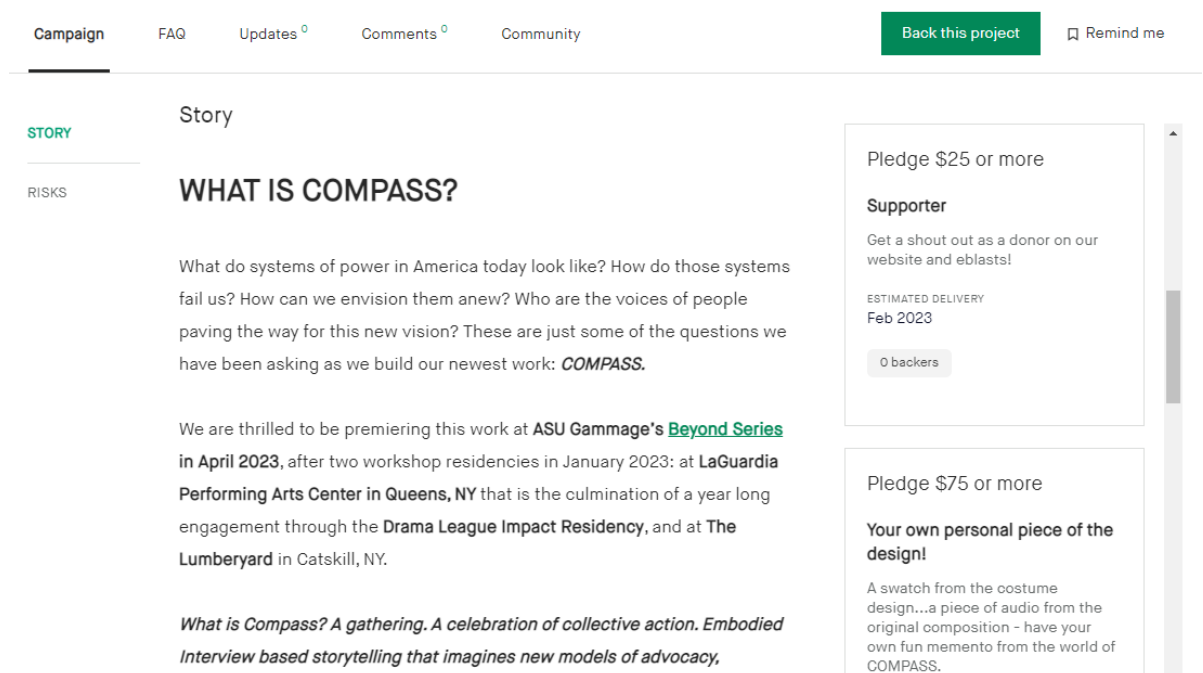


Рисунок 1.4 – Детальне представлення проєкту на Kickstarter

Порівняння цих програмних засобів між собою і з власною програмою, що створюється в роботі, дозволило виявити їх відмінності, переваги і недоліки (табл. 1.1).

Таблиця 1.1 – Порівняння програмних засобів

Критерій	FundRazr	Patreon	Kickstarter	Власна програма
Модель збору коштів	Все або нічого, вся сума	Сума підписок	Все або нічого	Все або нічого, мінімальна сума
Спосіб фінансування проєктів	Окремі платежі	Підписка	Окремі платежі	Окремі платежі
Безкоштовна підписка для власників проєктів	+	—	—	+
Прогнозування тривалості кампанії	—	—	—	+

1.3 Висновки за розділом 1

У першому розділі проаналізовано загальний стан проблеми збору коштів на фінансування проєктів, основні особливості і можливості програмних платформ, які використовуються для підтримки даного процесу, включаючи зокрема FundRazr, Patreon, Kickstarter. Встановленими завданнями дипломної кваліфікаційної роботи магістра є:

- виконати дослідження методів прогнозування тривалості кампанії для збору коштів на фінансування проєктів;
- спроектувати програмне забезпечення збору коштів на фінансування проєктів;
- реалізувати програмне забезпечення.

2 МАТЕРІАЛИ І МЕТОДИ

2.1 Задача прогнозування тривалості кампанії для збору коштів на фінансування проєктів

Кожна кампанія зі збору коштів на фінансування проєкту є унікальною, тому що унікальним зокрема є кожен проєкт, на який ці кошти збираються. Але авторів проєкту найбільше цікавить саме в кампанії зі збору коштів її основні результати. Цими основними результатами є зібрана сума коштів і співвідношена з цією величиною тривалість кампанії. Необхідно розуміти, що в подібних системах процес збору коштів не може тривати необмежено. Він завжди обмежений в часі. Відповідно і проєкт має почати реалізовуватись не в далекій перспективі, інакше сенс проведення кампанії зі збору коштів зникає. Однак окрім необхідності розпочати реалізовувати проєкт в певний момент, слід також розуміти, що зібрана сума залежить і від встановленого строку, а не тільки від проєкту, який пропонується, і загального моменту, коли проєкт заплановано розпочати. Цей строк має суттєвий вплив на успішність проєкту. Адже якщо строк буде встановлено невірно, а кампанія організована за принципом «все або нічого», то може статися ситуація, коли потрібні кошти вдалось би зібрати, але трохи пізніше, при чому це міг би бути період часу, який є прийнятним для проєкту, тобто його можна було б розпочати реалізовувати трохи пізніше, але при цьому зібравши кошти. У протилежному ж випадку всі кошти повертаються платникам і проєкт не реалізовується взагалі. Тому розуміння балансу між встановленою сумою коштів, які необхідно зібрати, і часом, який на цей збір виділений, є критичним. Фактично ж для конкретного проєкту сума коштів, яку хоче зібрати автор, є однією з вхідних ознак, якою характеризується проєкт і кампанія зі збору коштів на фінансування цього проєкту як екземпляр вибірки даних.

Усі екземпляри об'єднані у вибірку даних, що має застосовуватися для навчання. Для цієї вибірки відомі не тільки значення вхідних ознак для

кожного екземпляру, але відомо і значення вихідної ознаки, якою є тривалість кампанії зі збору коштів. При чому в якості суми коштів у вхідній ознаці встановлюється не заплановане значення, а фактично зібране. Тобто за кожним екземпляром вибірки тоді відомо, за який саме час яку саме суму коштів було зібрано.

Задача прогнозування тривалості кампанії для збору коштів на фінансування проєктів є задачею визначення тривалості кампанії для збору коштів на фінансування проєкту, для якого не відоме це значення, але відомий набір значень вхідних ознак. При цьому значення і вхідних ознак, і вихідної ознаки відомі на деякій навчальній вибірці.

Існуючі методи, які використовуються для підтримки кампаній для збору коштів на фінансування проєктів направлені на прогнозування успішності кампаній або прогнозування зібраних коштів.

У роботі [10] на основі глибокої нейронної мережі відбувається прогнозування фінансових результатів кампанії.

У роботі [11] також відбувається прогнозування саме фінансового результату, при цьому найкращі результати продемонстрували штучні нейронні мережі, дерева ухвалення рішень.

Однак важливим параметром є і встановлені строки. При цьому це параметр, який стосується вже не результатів кампанії, а прийняття рішень щодо керування самою кампанією початково. Якщо автор буде мати можливість скоригувати їх і визначити найближчий варіант до успішного проєкту в підсумку, це буде достатньо корисною інформацією для прийняття рішень. Окрім того це дозволить прийняти найбільш ефективне рішення.

2.2 Дерева ухвалення рішень для розв'язання задачі прогнозування

Дерево ухвалення рішень – це спосіб моделювання результатів і рішень, коли кожне рішення в розгалуженій структурі представляє іншу

гілку, використовується для розрахунку потенціалу успіху різних рішень для досягнення конкретної мети [12]. Дерева ухвалення рішень можуть застосовуватися як для розв'язання задачі класифікації, так і для розв'язання задачі прогнозування.

У формі прогнозного моделювання дерева ухвалення рішень допомагають відображати різні рішення або рішення, які виникають для певної ситуації: для створення дерев потрібні різні вузли, початком дерева є кореневий вузол, який є цілим набором даних, який використовується в машинному навчанні [12].

Переваги дерев ухвалення рішень полягають у наступному;

- легко інтерпретуються: дерева ухвалення рішень легко інтерпретувати, оскільки вони забезпечують спосіб візуалізації алгоритмів;
- низькі вимоги до попередньої обробки даних або їх відсутність: на відміну від інших алгоритмів, дерева ухвалення рішень потребують менше часу для моделювання, оскільки вони потребують мало аналізу, кодування та фіктивних змінних, оскільки для кожної точки даних буде цілий набір;
- універсальність даних: можна стандартизувати дані, можна наповнювати категоріальні та числові дані, оскільки дерева ухвалення рішень здатні працювати з обома типами [12]-[16].

Алгоритм Classification And Regression Tree (CART) створює дерева таким чином:

- кожен внутрішній вузол двійкового дерева має рівно два вихідних ребра, а розбиття внутрішніх вузлів дерева здійснюється за допомогою критеріїв ентропії або Джіні;
- згенероване дерево CART можна скоротити, і для скорочення дерева використовується критерій вартості та складності, який може враховувати витрати на неправильну класифікацію при створенні дерева, а також можна обчислити розподіл ймовірностей кожного вузла;
- однією з важливих характеристик CART є його здатність генерувати дерева регресії для прогнозування;

– у деревах регресії листи передбачають дійсне число та розглядають розділення, де квадрат помилки передбачення (квадрат відхилення між прогнозованими та реальними значеннями) мінімізований, відповідно прогноз для кожного листа базуватиметься на середній вазі вузла;

– у випадку застосування для регресії для кожного термінального вузла замість призначення класу призначається числове значення, яке обчислюється шляхом взяття середнього або медіани вибірки на навчальних даних, що відповідають вузлу, під час процесу вирощування дерева, розподіл, вибраний на кожному етапі, є таким, що призводить до найбільшого зменшення суми абсолютних різниць між значеннями відповідей для навчальних вибірок, що відповідають певному вузлу, та їх медіаною вибірки, для обрізки дерев також використовується сума квадратних або абсолютних різниць [12]-[16].

Основна проблема з деревами ухвалення рішень полягає в тому, що воно має таблицю, наповнену атрибутами до такої міри, що додати більше стає важко, може бути важко навчити набір даних, коли є так багато інформації, при цьому якщо для дерева немає встановленого обмеження, можна досягти 100 % точності навчання, у гіршому випадку це створить один листок на спостереження, що вплине на точність прогнозування для нових даних, яких не було в навчальній вибірці, а тому застосовуються техніки видалення надлишку, серед яких є обрізка: процес розбиття забезпечить вирощене дерево, поки воно не досягне кінцевої стадії, однак повністю вирощене дерево характеризується перенавчанням, що призводить до низької точності даних, яких не було в навчальній вибірці (при обрізанні вузли рішення відрізаються від вузла листя, навчальний набір можна відфільтрувати на два набори: набір даних перевірки та набір навчальних даних, дерево ухвалення рішень можна побудувати шляхом фільтрації навчального набору даних, процес обрізки може тривати, доки не буде досягнуто необхідної точності) [12]-[16].

До основних алгоритмів на основі дерев ухвалення рішень належать:

- iterative dichotomiser 3 (ID3);
- C4.5;
- CART.

Їх порівняння виконано в таблиці 2.1 і враховуючи, що тільки CART дозволяє розв'язувати задачу регресії, тобто виконувати прогнозування, то саме його обрано для створення власного методу.

Таблиця 2.1 – Порівняння алгоритмів на основі дерев ухвалення рішень

Критерій	ID3	C4.5	CART
Задача, для розв'язання якої може використовуватися	Класифікація	Класифікація	Класифікація, регресія
Здатність працювати з пропущеними значеннями	Відсутня	Відсутня	Присутня
Швидкість	Низька	Вища	Середня

2.3 Метод прогнозування тривалості кампанії для збору коштів на фінансування проєктів

Використовуючи логіку задачі прогнозування тривалості кампанії для збору коштів на фінансування проєктів і отримані результати аналізу подібних задач і методів, дерев рішень, було сформульовано метод прогнозування тривалості кампанії для збору коштів на фінансування проєктів.

Метод вимагає на початку створення вибірки даних, яка буде використовуватися в подальшому для навчання моделі. Для створення вибірки даних має бути з зовнішнього джерела, з бази даних (БД) витягнуто всю наявну інформацію про будь-які кампанії зі збору коштів на фінансування проєктів. При цьому можуть використовуватися як успішні, так

і неуспішні кампанії. Вибірка даних має бути представлена екземплярами, що складаються з наступних ознак:

- галузь знань або категорія проєкту – цілочисельне значення, що відповідає одній з категорій з множини категорій проєктів;
- сума коштів проєкту: в якості даного значення використовується сума, яку було фактично зібрано за результатами збору коштів;
- запланована тривалість кампанії зі збору коштів;
- назва проєкту: використовуються текстові рядки до 100 символів;
- фактична тривалість збору коштів на фінансування проєкту, за яку було зібрано дану суму коштів, – вихідна ознака вибірки даних.

На другому етапі має бути виконано перетворення текстових даних у векторизовану числову форму, використовуючи один з методів для виконання такого перетворення. Запропоновано використовувати в якості такого методу метод Term Frequency – Inverse Document Frequency (TF-IDF).

На третьому етапі відбувається навчання за цією вибіркою у формі, отриманій за результатами другого етапу, дерева ухвалення рішення на основі алгоритму CART.

На четвертому етапі відбувається збереження моделі, навчання якої було виконано на третьому етапі, в файл або завантаження її, якщо даний етап виконується окремо для прогнозування. Окремо даний етап може виконуватися, наприклад, під час роботи програми, в якій використовується навчена і раніше збережена модель.

На п'ятому етапі виконується витягання з БД або на основі заданих параметрів даних, які характеризують кампанію зі збору коштів, яка ще не була запущена і для якої невідома фактична тривалість. Ці дані відповідають ознакам вибірки, виключаючи вихідну ознаку. Даний етап може виконуватися повторно, якщо значення ознак кампанії змінюються її автором або менеджером для того, щоб дізнатися прогнозовану тривалість кампанії для заданих параметрів, що включають, наприклад, заплановану суму збору коштів, якщо вона може бути різною в різних варіантах виконання проєкту.

На шостому етапі на основі сформованих даних навченою моделлю відбувається прогнозування тривалості виконання збору коштів для заданої на п'ятому етапі кампанії.

2.4 Висновки за розділом 2

Описано задачу прогнозування тривалості кампанії для збору коштів на фінансування проєктів, проаналізовано методи, які використовуються для розв'язання задач в даній проблематиці, проаналізовано алгоритми на основі дерев ухвалення рішень.

Наукова новизна роботи сформована в методі прогнозування тривалості кампанії для збору коштів на фінансування проєктів, який виконує визначення строку, необхідного на збір певної суми коштів на фінансування конкретного проєкту, до початку кампанії, використовуючи дерева ухвалення рішень на основі алгоритму CART, що дозволяє збалансувати встановлені суму коштів кампанії та тривалість такої кампанії, щоб в підсумку вона з більшою вірогідністю завершилась успішно.

3 ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Вибір рішень для розробки програмного забезпечення

Мова Python є однією з сучасних потужних мов програмування, які підтримують і створення вебдодатків, і роботу з даними. Важливою перевагою мови є те, що вона дозволяє писати достатньо короткий читабельний код, який призводить до швидкого створення нових потужних рішень. Оскільки для Python є велика кількість готових пакетів для роботи з даними, машинного навчання, то її було використано як основну мову веброзробки. Для створення вебдодатків не з нуля вона потребує використання відповідного фреймворку. Потужним фреймворком для цієї проблеми є фреймворк Django. Альтернативним для нього було розглянуто фреймворк TurboGears (табл. 3.1), зважаючи на те, що він має також достатньо багато вбудованих можливостей, що значно спрощують створення програмного проєкту рівня даної роботи. Обрано було фреймворк Django, врахувавши, що він має багато вбудованих засобів проти різного роду атак, що відсутнє в TurboGears і потребує значного досвіду у вирішенні цих проблем.

Таблиця 3.1 – Порівняння фреймворків для розробки вебдодатку

Критерій	Django	TurboGears
Підтримка об'єктно-реляційного відображення	Вбудована, Django ORM	Вбудована, SQLAlchemy
Рушій шаблонів	Django	Genshi
Адміністрування	Вбудоване	Вбудоване
Безпека	Потужна проти великої кількості атак	Максимально обмежена

3.2 Визначення функціональних вимог до програми

Користувачі мають взаємодіяти з програмою або як неавторизовані (рис. 3.1), або як авторизовані. Програма не передбачає розділення користувачів на тих, хто можуть тільки сплачувати за інші проєкти, і тих, хто виставляють власні проєкти. Тобто кожен авторизований користувач може виступати будь-якою з цих двох сторін.

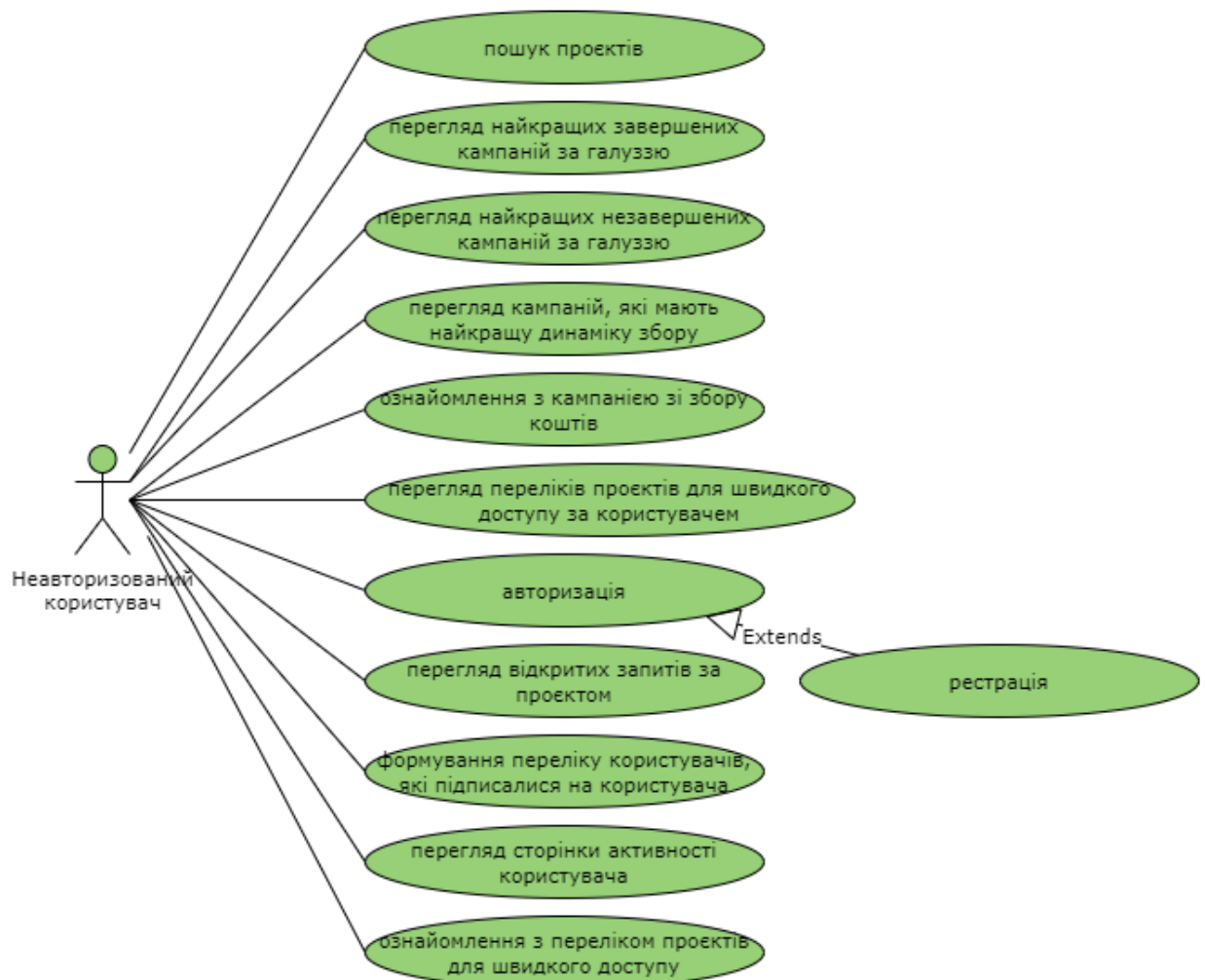


Рисунок 3.1 – Діаграма прецедентів неавторизованого користувача

Сценаріями неавторизованого користувача є:

- пошук проєктів;
- перегляд найкращих завершених кампаній за галуззю;
- перегляд найкращих незавершених кампаній за галуззю;

- перегляд кампаній, які мають найкращу динаміку збору;
- ознайомлення з кампанією зі збору коштів для фінансування проєкту;

- перегляд переліків проєктів для швидкого доступу за користувачем;
- ознайомлення з переліком проєктів для швидкого доступу;
- перегляд відкритих запитів за проєктом;
- формування переліку користувачів, які підписалися на користувача;
- перегляд сторінки активності користувача;
- авторизація, що має як альтернативний сценарій реєстрацію.

Сценаріями авторизованого користувача (рис. 3.2) є:

- запуск кампанії зі збору коштів на фінансування проєкту;
- перегляд платежів за кампанією зі збору коштів на фінансування проєкту;

- перегляд власних платежів на фінансування проєктів;
- перегляд власних переліків проєктів для швидкого доступу;
- перегляд приватних запитів за проєктом;
- перегляд усіх власних запитів за всіма проєктами;
- перегляд власних кампаній зі збору коштів на фінансування проєктів, де є альтернативний сценарій редагування параметрів власної кампанії зі збору коштів на фінансування проєкту;

- перегляд підтримуваних кампаній зі збору коштів на фінансування проєктів;

- визначення користувачів, які підтримують кампанію зі збору коштів на фінансування проєкту;

- створення персонального подарунку для користувача;

- вподобання проєкту, збір коштів на фінансування якого відбувається, де як альтернативний сценарій використовується скасування вподобання проєкту (рис. 3.2).

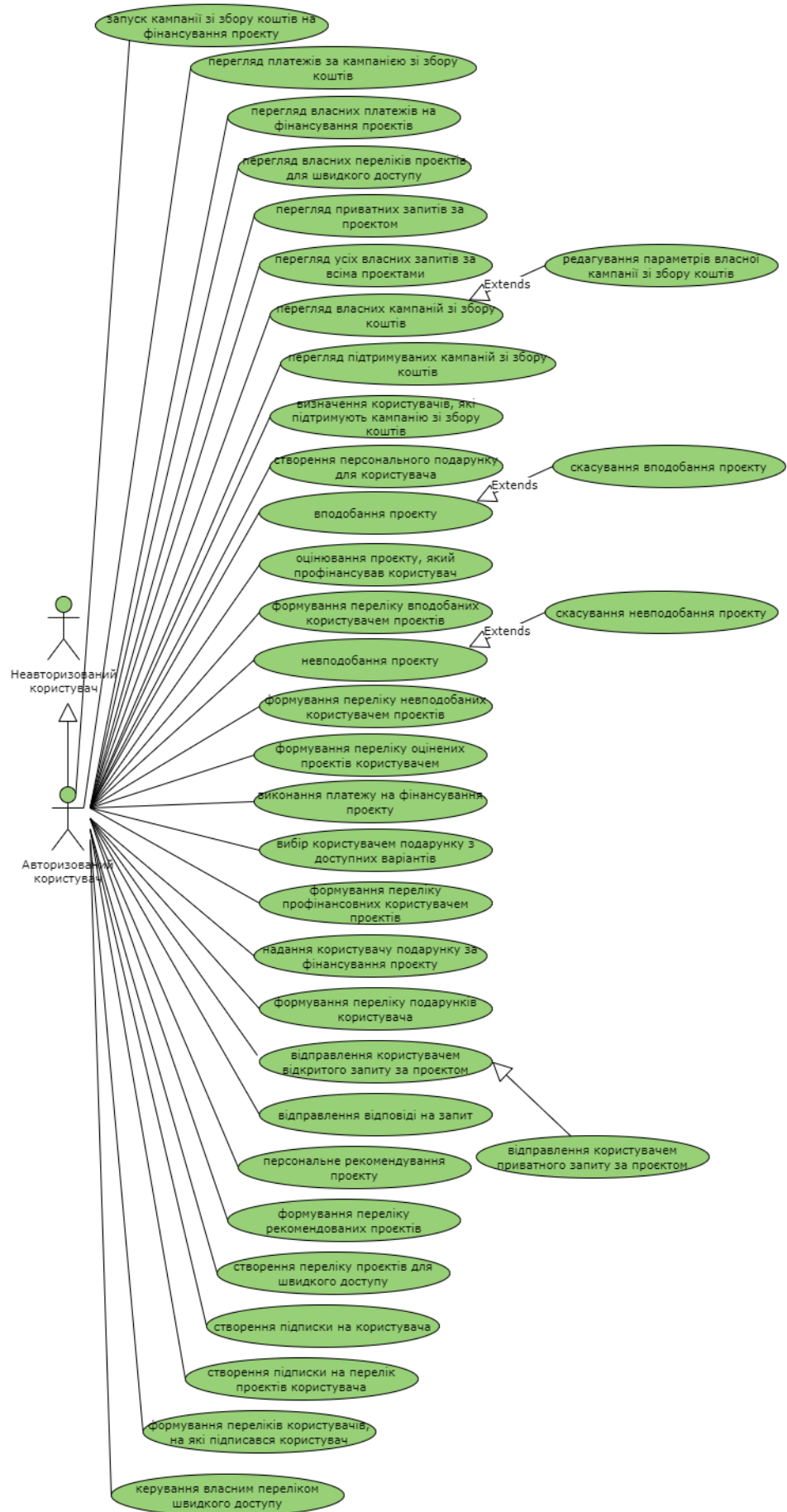


Рисунок 3.2 – Діаграма прецедентів авторизованого користувача

Також до сценаріїв авторизованого користувача належать:

- несподобання проєкту, збір коштів на фінансування якого відбувається, де як альтернативний сценарій використовується скасування несподобання проєкту;
- оцінювання проєкту, який профінансував користувач;
- формування переліку вподобаних користувачем проєктів;
- формування переліку несподобаних користувачем проєктів;
- формування переліку оцінених проєктів користувачем;
- виконання платежу на фінансування проєкту;
- вибір користувачем подарунку з доступних варіантів;
- формування переліку профінансованих користувачем проєктів;
- надання користувачу подарунку за фінансування проєкту;
- формування переліку подарунків користувача;
- відправлення користувачем відкритого запиту за проєктом;
- відправлення користувачем приватного запиту за проєктом;
- відправлення відповіді на запит;
- персональне рекомендування проєкту;
- формування переліку рекомендованих проєктів;
- створення переліку проєктів для швидкого доступу;
- керування переліком швидкого доступу;
- створення підписки на користувача;
- формування переліків користувачів, на які підписався авторизований користувач;
- створення підписки на перелік проєктів користувача.

3.3 Визначення структури бази даних

БД побудована на основі використання системи MySQL, що є однією з широко розповсюджених безкоштовних, потужних і надійних подібних систем.

Проекти, фінансування яких має накопичуватися через програму, фактично асоціюються з кампаніями по збору коштів на них, а відповідні дані про проекти і кампанії зі збору коштів зберігаються в таблиці БД `faundcampsoft_funding` з наступними полями:

- `id` – номер кампанії зі збору коштів на фінансування проекту;
- `caption` – назва проекту, збір коштів на який реалізується;
- `caption_en` – назва проекту англійською мовою;
- `area` – галузь проектів, до якої відноситься даний;
- `idea` – опис основної ідеї проекту;
- `textdescription` – повний текстовий опис проекту, який повністю репрезентує проект;
- `minsum` – мінімальна сума, яку має бути зібрано, щоб кошти були не повернені людям, які їх направили, тобто це мінімальний рівень фінансування, за якого автор все ж почне виконувати проект, хоча в підсумку створить розширений прототип або обмежене рішення порівняно з першочерговою ідеєю: якщо значення не встановлено, то має бути зібрано тільки всю суму повністю;
- `plannedsum` – запланована для збору сума фінансування: якщо не встановлено `minsum`, то у випадку, якщо на момент завершення кампанії цю суму не зібрано, то повертаються всі кошти всім людям, які долучились до збору;
- `gatheredsum` – фактично зібрана на поточний момент (якщо кампанія триває) сума коштів на фінансування проекту;
- `payments` – кількість платежів, які було отримано на фінансування проекту;
- `forecast` – прогнозована тривалість кампанії зі збору коштів на фінансування даного проекту;
- `startdate` – дата, час початку збору коштів на фінансування проекту;
- `duedate` – дата, час завершення кампанії зі збору коштів на фінансування проекту;

– stopdate – дата, час, до якого буде тривати збір коштів, якщо до моменту duedate вже вдалося зібрати потрібну суму: якщо дату не задано, то кампанія завершується в момент duedate, якщо fullsum не визначає завершення одразу за отримання потрібної суми фінансування (раніше кінцевої дати);

– fullsum – варіант збору коштів: за отримання потрібної суми кампанія одразу завершується чи вона продовжується до встановленої дати завершення;

– creationdate – дата, час внесення інформації про проєкт і кампанію зі збору коштів на нього в програму;

– updatedate – дата, час останнього оновлення будь-якої інформації стосовно проєкту;

– lastrequestdate – дата, час останнього запиту за проєктом;

– lastasnwerdate – дата, час останньої відповіді на запит за проєктом;

– author – автор проєкту або власник кампанії зі збору коштів на нього;

– state – стан кампанії зі збору коштів на фінансування проєкту;

– requests – кількість запитів, яка була поставлена за проєктом;

– likes – кількість уподобань проєкту всіма користувачами: входять і ті, хто фінансували проєкт, і ті, хто не віддавав власні кошти;

– dislikes – кількість незадоволень проєктом всіма користувачами: входять і ті, хто фінансували проєкт, і ті, хто не віддавав власні кошти;

– rating – середній рівень рейтингу проєкту за оцінками користувачів, які перерахували власні кошти на фінансування проєкту;

– lists – кількість переліків проєктів, у які було внесено даний проєкт усіма користувачами;

– meta – мета проєкту;

– team – опис команди, яка буде працювати з проєктом;

– sustainability – підтримка стабільності проєкту після його виконання.

Поле `author` таблиці представлено зовнішнім ключем, який представляє таблицю `faundcampsoft_participant`. Будь-який користувач може бути власником різних кампаній зі збору коштів, тобто обмеження на кількість створених проєктів фактично немає. Кожен проєкт може мати тільки одного власника. Значить тип зв'язку між таблицями один-до-багатьох.

Поле `area` таблиці представлено зовнішнім ключем, який представляє таблицю `faundcampsoft_area`. Будь-яка галузь проєктів має бути представлена різними проєктами, інакше немає сенсу виділяти таку галузь. Кожен проєкт може належати тільки одній галузі. Значить тип зв'язку між таблицями один-до-багатьох.

Галузі проєктів знаходяться в таблиці `faundcampsoft_area` з наступними полями:

- `id` – номер галузі проєктів;
- `name` – назва галузі проєктів.

Користувачі, які одночасно є учасниками або потенційними учасниками кампаній зі збору коштів на фінансування проєктів, зберігаються в таблиці БД `faundcampsoft_participant` з наступними полями:

- `id` – номер користувача, який розглядає можливості участі в кампаніях зі збору коштів на фінансування проєктів;
- `nickname` – логін користувача, за яким він авторизується в програмі;
- `password` – пароль користувача, за яким він авторизується в програмі;
- `paiedtotal` – сума, яку сплатив користувач на фінансування всіх проєктів у програмі загалом;
- `likes` – кількість уподобань різних проєктів, які були виконані загалом даним користувачем;
- `dislikes` – кількість незадоволень різними проєктами, які були виконані загалом даним користувачем;

- ratings – кількість оцінювань різних проєктів, які були виконані загалом даним користувачем;
- payments – кількість платежів, які перенаправив на фінансування всіх проєктів загалом користувач;
- requests – кількість запитів, яку стосовно різних проєктів загалом відправив користувач;
- photo – фотографія користувача;
- bage – позначка, яку було встановлено користувачу за сплаченими сумами на фінансування проєктів та активністю в програмі;
- email – електронна пошта користувача;
- phone – номер телефону користувача;
- subscribers – кількість користувачів, які підписалися на даного користувача.

Одразу декілька користувачів можуть працювати над кампанією зі збору коштів на фінансування проєкту. Сам власник може встановлювати основні параметри проєкту. Інші користувачі можуть здійснювати підтримку проєкту, наприклад, завантажуючи персональні подарунки для інвесторів у проєкт. Кожен користувач може працювати з різними проєктами, кожен проєкт може підтримуватися різними користувачами. Значить тип зв'язку між таблицями багато-до-багатьох. Зв'язок реалізовано через створення додаткової таблиці `faundcampsoft_fundingparticipant` з наступними полями:

- id – номер запису;
- funding – кампанія зі збору коштів на проєкт, з яким працює даний користувач;
- participant – користувач, який працює над підтримкою даного проєкту.

Уподобання або невподобання користувачами проєктів, збір коштів на фінансування яких виконується, знаходяться в таблиці `faundcampsoft_like` з наступними полями:

- id – номер уподобання (невподобання) користувачем проєкту, збір коштів на фінансування якого виконується;
- funding – проєкт, кампанія зі збору коштів на фінансування якого відбувається в програмі і який уподобав користувач;
- participant – користувач, який уподобав даний проєкт;
- likeordis – позначка про те, чи є це вподобанням чи навпаки невподобанням.

Поле funding таблиці представлено зовнішнім ключем, який представляє таблицю faundcampsoft_funding. Будь-який проєкт може бути вподобаним або невподобаним різними користувачами. Кожне вподобання стосується тільки одного проєкту зі всіх. Значить тип зв'язку між таблицями один-до-багатьох.

Поле participant таблиці представлено зовнішнім ключем, який представляє таблицю faundcampsoft_participant. Будь-який користувач може мати різні уподобання або невподобання. Кожне вподобання або невподобання виконується тільки одним користувачем. Значить тип зв'язку між таблицями один-до-багатьох.

Оцінювання, які формують середній рівень рейтингу проєктів, виконуються користувачами, які профінансували проєкт, і знаходяться в таблиці faundcampsoft_rating з наступними полями:

- id – номер оцінювання проєкту, збір коштів на фінансування якого відбувається, користувачем;
- funding – проєкт, збір коштів на фінансування якого відбувається і який був оцінений користувачем, що профінансував його;
- participant – користувач, який профінансував проєкт;
- points – оцінка, яку виставив користувач проєкту;
- message – текстова оцінка користувачем проєкту;
- creationdate – дата, час оцінювання.

Поле funding таблиці представлено зовнішнім ключем, який представляє таблицю faundcampsoft_funding. Будь-який проєкт може бути

оцінений різними користувачами. Кожне оцінювання стосується тільки одного проєкту зі всіх. Значить тип зв'язку між таблицями один-до-багатьох.

Поле `participant` таблиці представлено зовнішнім ключем, який представляє таблицю `faundcampsoft_participant`. Будь-який користувач може оцінювати різні проєкти. Кожне оцінювання виконується тільки одним користувачем. Значить тип зв'язку між таблицями один-до-багатьох.

Платежі на фінансування проєкту в межах кампанії зі збору коштів знаходяться в таблиці `faundcampsoft_fundpayment` з наступними полями:

- `id` – номер платежу на фінансування проєкту в межах кампанії зі збору коштів;
- `participant` – користувач, який виконав платіж;
- `funding` – кампанія зі збору коштів на фінансування проєкту, за якою виконано платіж;
- `paiedsum` – сплачена сума в цьому платежу;
- `ppaccount` – обліковий запис PayPal, через який було реалізовано платіж;
- `paieddate` – дата, час виконання платежу;
- `caption` – повідомлення від платника.

Поле `funding` таблиці представлено зовнішнім ключем, який представляє таблицю `faundcampsoft_funding`. Будь-який проєкт може мати різну кількість платежів на його фінансування. Кожен платіж стосується тільки одного проєкту зі всіх. Значить тип зв'язку між таблицями один-до-багатьох.

Поле `participant` таблиці представлено зовнішнім ключем, який представляє таблицю `faundcampsoft_participant`. Будь-який користувач може виконувати різні платежі для фінансування проєктів. Кожен платіж виконується тільки одним користувачем. Значить тип зв'язку між таблицями один-до-багатьох.

Подарунки для користувачів, які профінансували проєкт, знаходяться в таблиці `faundcampsoft_present` з наступними полями:

- id – номер подарунка;
- filedata – дані файлу, який представляє подарунок (якщо це саме файл);
- filename – ім'я файлу, який представляє подарунок (якщо дані не завантажені в полі filedata);
- iffile – подарунок представлений файлом або це інформаційне повідомлення про подарунок, який буде надісланий користувачу;
- name – назва подарунку;
- caption – текстовий опис подарунку;
- quantity – сумарна кількість доступних екземплярів даного подарунку;
- leftquantity – кількість доступних екземплярів даного подарунку, яка залишилась не подарованою;
- check – необхідність вибору даного подарунку серед всіх подарунків даного рівня або обов'язковість надання даного подарунку за досягнення відповідного рівня.

Якщо подарунок має вибиратися з аналогічних самим користувачем, який фінансує проєкт, то вибір користувача знаходиться в таблиці faundcampsoft_participantpresent з наступними полями:

- id – номер вибору користувачем подарунку;
- participant – користувач, який виконав цей вибір подарунку;
- present – подарунок, який обрав користувач серед доступних йому;
- creationdate – дата, час виконання вибору подарунку.

Поле participant таблиці представлено зовнішнім ключем, який представляє таблицю faundcampsoft_participant. Будь-який користувач може виконувати в різні моменти вибір подарунків після внесення сплат за різні проєкти. Кожен вибір виконується тільки одним користувачем. Значить тип зв'язку між таблицями один-до-багатьох.

Поле present таблиці представлено зовнішнім ключем, який представляє таблицю faundcampsoft_present. Будь-який подарунок може

вибиратися різними користувачами в різні моменти, якщо подарунок доступний в декількох екземплярах. Кожен вибір стосується тільки одного вибраного подарунку. Значить тип зв'язку між таблицями один-до-багатьох.

Окрім стандартних подарунків є персональні подарунки, тобто подарунки, які призначені для конкретних користувачів. Вони знаходяться в таблиці `faundcampsoft_personalpresent` з наступними полями:

- `id` – номер персонального подарунку;
- `fundinglevelpersonalpresent` – доступність персональних подарунків для відповідного рівня участі користувачів у фінансуванні проєкту, до якої відноситься цей подарунок;
- `participant` – користувач, якому адресовано цей персональний подарунок;
- `beforedue` – персональний подарунок має бути віддано до завершення збору коштів на проєкт чи одразу після завершення;
- `filedata` – дані файлу, який представляє персональний подарунок (якщо це саме файл);
- `filename` – ім'я файлу, який представляє персональний подарунок (якщо дані не завантажені в полі `filedata`);
- `iffile` – подарунок представлений файлом або це інформаційне повідомлення про подарунок, який буде надісланий користувачу;
- `name` – назва подарунку;
- `caption` – текстовий опис подарунку.

Поле `participant` таблиці представлене зовнішнім ключем, який представляє таблицю `faundcampsoft_participant`. Будь-який користувач може мати декілька адресованих йому персональних подарунків. Кожен подарунок може бути адресований тільки одному користувачу. Значить тип зв'язку між таблицями один-до-багатьох.

Поле `fundinglevelpersonalpresent` таблиці представлене зовнішнім ключем, який представляє таблицю `faundcampsoft_fundinglevelpersonalpresent`. Будь-який персональний

подарунок визначається однією групою персональних подарунків для відповідного рівня участі користувачів у фінансуванні проєкту, до якої відноситься цей подарунок. Одна група може мати багато різних персональних подарунків. Значить тип зв'язку між таблицями один-до-багатьох.

Матеріали використовуються для візуального відображення ідей проєкту, складаються з фотографій або відео, знаходяться в таблиці `faundcampsoft_material` з наступними полями:

- `id` – номер матеріалу, який представляє проєкт (фотографія або відео);
- `funding` – проєкт, кампанія зі збору коштів для фінансування якого виконується;
- `filedata` – дані файлу, який представляє матеріал, якщо він міститься на сервері;
- `filename` – ім'я файлу, який представляє матеріал (якщо дані не завантажені в полі `filedata`);
- `caption` – підпис до матеріалу.

Запит користувача, який може бути відкритим або приватним, зберігається в таблиці `faundcampsoft_participantrequest` з наступними полями:

- `id` – номер запиту користувача;
- `participant` – користувач, який створив запит;
- `funding` – проєкт, кампанія зі збору коштів для фінансування якого виконується;
- `messagerequest` – текст запиту;
- `messageanswer` – відповідь на запит;
- `creationdate` – дата, час публікації запиту;
- `wread` – прочитаний запит або ні;
- `opened` – тип запиту: відкритий для публікації всім користувачам або приватний для відтворення тільки користувачам, що є автором запиту і що пов'язаний з підтримкою проєкту.

Поле `funding` таблиці представлено зовнішнім ключем, який представляє таблицю `faundcampsoft_funding`. Будь-який проєкт може мати різну кількість запитів від користувачів. Кожен запит стосується тільки одного проєкту зі всіх. Значить тип зв'язку між таблицями один-до-багатьох.

Поле `participant` таблиці представлено зовнішнім ключем, який представляє таблицю `faundcampsoft_participant`. Будь-який користувач може виконувати запити за проєктами. Кожен запит виконується тільки одним користувачем. Значить тип зв'язку між таблицями один-до-багатьох.

Поточний рівень участі користувача в фінансуванні проєкту знаходиться в таблиці `faundcampsoft_participantlevel` з наступними полями:

- `id` – номер поточного рівня участі користувача в фінансуванні проєкту;
- `participant` – користувач, який бере участь у фінансуванні проєкту;
- `fundinglevel` – рівень участі користувача в фінансуванні проєкту;
- `updatedate` – дата, час встановлення даного рівня;
- `paied` – сума, яку сплатив користувач на фінансування даного проєкту на даний момент;
- `leftforpresent` – сума, яку ще не було гарантовано подарунком зі сплаченої на поточний момент.

Поле `funding` таблиці представлено зовнішнім ключем, який представляє таблицю `faundcampsoft_funding`. Будь-який проєкт може мати різний рівень фінансування різними користувачами. Кожен рівень стосується тільки одного проєкту зі всіх. Значить тип зв'язку між таблицями один-до-багатьох.

Поле `fundinglevel` таблиці представлено зовнішнім ключем, який представляє таблицю `faundcampsoft_fundinglevel`. Будь-який рівень участі користувачів у фінансуванні проєкту може мати різних користувачів, які знаходяться на цьому рівні. Кожен рівень участі конкретного користувача в конкретному проєкті характеризується тільки одним загальним (визначеним

для цілого проєкту) рівнем участі користувачів. Значить тип зв'язку між таблицями один-до-багатьох.

Ці загальні рівні участі користувачів, які визначені для всього проєкту загалом, знаходяться в таблиці `faundcampsoft_fundinglevel` з наступними полями:

- `id` – номер визначеного для всього проєкту рівня участі користувачів у його фінансуванні;
- `funding` – проєкт, кампанія зі збору коштів для фінансування якого виконується;
- `level` – числове значення рівня фінансування проєкту;
- `caption` – текстове значення рівня фінансування проєкту;
- `paidsum` – сплачена користувачем сума, за якої активується цей рівень участі.

Поле `funding` таблиці представлено зовнішнім ключем, який представляє таблицю `faundcampsoft_funding`. Будь-який проєкт може мати декілька рівнів участі користувачів у його фінансуванні. Кожен рівень стосується тільки одного проєкту зі всіх. Значить тип зв'язку між таблицями один-до-багатьох.

Рекомендації звернути увагу на проєкт від одного користувача іншому знаходяться в таблиці `faundcampsoft_recommendation` з наступними полями:

- `id` – номер рекомендації;
- `author` – користувач, який є автором рекомендації;
- `receiver` – користувач, якому адресована рекомендація;
- `funding` – проєкт, кампанія зі збору коштів для фінансування якого виконується;
- `creationdate` – дата, час створення рекомендації.

Поле `funding` таблиці представлено зовнішнім ключем, який представляє таблицю `faundcampsoft_funding`. Будь-який проєкт може мати різну кількість рекомендацій. Кожна рекомендація стосується тільки одного проєкту зі всіх. Значить тип зв'язку між таблицями один-до-багатьох.

Поле `author` таблиці представлено зовнішнім ключем, який представляє таблицю `faundcampsoft_participant`. Будь-який користувач може створити декілька рекомендацій. Кожна рекомендація має тільки одного автора. Значить тип зв'язку між таблицями один-до-багатьох.

Поле `receiver` таблиці представлено зовнішнім ключем, який представляє таблицю `faundcampsoft_participant`. Будь-який користувач може отримати декілька рекомендацій. Кожна рекомендація має тільки одного адресата. Значить тип зв'язку між таблицями один-до-багатьох.

Переліки проєктів для зручного доступу зберігаються в таблиці `faundcampsoft_partlist` з наступними полями:

- `id` – номер переліку проєктів, кампанії зі збору коштів для фінансування яких виконуються;
- `participant` – користувач, який створив даний перелік проєктів, кампанії зі збору коштів для фінансування яких виконуються;
- `caption` – назва переліку проєктів, кампанії зі збору коштів для фінансування яких виконуються;
- `creationdate` – дата, час створення переліку проєктів, кампанії зі збору коштів для фінансування яких виконуються;
- `updatedate` – дата, час останнього оновлення переліку проєктів, кампанії зі збору коштів для фінансування яких виконуються;
- `subs` – кількість підписок на перелік проєктів.

Поле `participant` таблиці представлено зовнішнім ключем, який представляє таблицю `faundcampsoft_participant`. Будь-який користувач може створити декілька переліків проєктів. Кожний перелік має тільки одного автора. Значить тип зв'язку між таблицями один-до-багатьох.

Самі ж проєкти, які входять до переліку, містяться в окремій таблиці, що реалізує зв'язок багато-до-багатьох. Кожен проєкт може входити до різних переліків. Кожен перелік може містити різні проєкти.

Таблиця `faundcampsoft_fundinglist` є таблицею з наступними полями:

- `id` – номер запису;

– funding – проєкт, кампанія зі збору коштів для фінансування якого виконується;

– partlist – номер переліку проєктів, кампанії зі збору коштів для фінансування яких виконуються.

Підписки користувачів на інших користувачів зберігаються в таблиці faundcampsoft_partsubscribe з наступними полями:

- id – номер підписки;
- participant – користувач, підписка на якого оформлена;
- subscriber – підписник.

Таблиця реалізує зв'язок багато-до-багатьох. Кожен підписник може мати багато підписок. На кожного користувача може бути оформлено багато підписок.

Поле participant таблиці представлене зовнішнім ключем, який представляє таблицю faundcampsoft_participant. Поле subscriber таблиці представлене зовнішнім ключем, який представляє таблицю faundcampsoft_participant.

Підписки можуть створюватися і на переліки проєктів, не тільки на інших користувачів. Підписки на переліки проєктів знаходяться в таблиці faundcampsoft_listsubscribe з наступними полями:

- id – номер підписки на перелік проєктів;
- partlist – перелік проєктів, на який було створено підписку;
- participant – користувач, який створив підписку.

Поле participant таблиці представлене зовнішнім ключем, який представляє таблицю faundcampsoft_participant. Будь-який користувач може підписатися на декілька переліків проєктів. Кожна підписка є підпискою конкретного користувача. Значить тип зв'язку між таблицями один-до-багатьох.

Поле partlist таблиці представлене зовнішнім ключем, який представляє таблицю faundcampsoft_partlist. Будь-який перелік може мати

декілька підписок. Кожна підписка є підпискою на конкретний перелік проєктів. Значить тип зв'язку між таблицями один-до-багатьох.

Подарунки, які можна обрати за відповідного рівня участі в фінансуванні проєкту, містяться в окремій таблиці, що реалізує зв'язок багато-до-багатьох. Кожен рівень участі у фінансуванні проєкту може мати декілька пропонованих подарунків. Кожен подарунок потенційно може використовуватися в різних проєктах (наприклад, для одного автора).

Відповідна таблиця `faundcampsoft_fundinglevelpresent` є таблицею з наступними полями:

- `id` – номер запису;
- `fundinglevel` – рівень участі користувача в фінансуванні проєкту, якому відповідають ці подарунки;
- `present` – подарунок, який може бути отриманий за досягнення цього рівня участі в фінансуванні проєкту.

Доступність персональних подарунків для відповідного рівня участі користувачів у фінансуванні проєкту знаходиться в таблиці `faundcampsoft_fundinglevelpersonalpresent` з наступними полями:

- `id` – номер запису про доступність персональних подарунків;
- `fundinglevel` – рівень участі користувачів у фінансуванні проєктів;
- `caption` – назва подарунку;
- `quantity` – кількість доступних заявок на такий персональний подарунок;
- `leftquantity` – кількість персональних подарунків, яка доступна все ще для даного рівня;
- `check` – необхідність вибору даного подарунку або автоматичне його відправлення за досягнення цього рівня участі користувачів у фінансуванні проєктів.

Описані таблиці та зв'язки між ними подані у вигляді схеми структури БД (рис. 3.3).

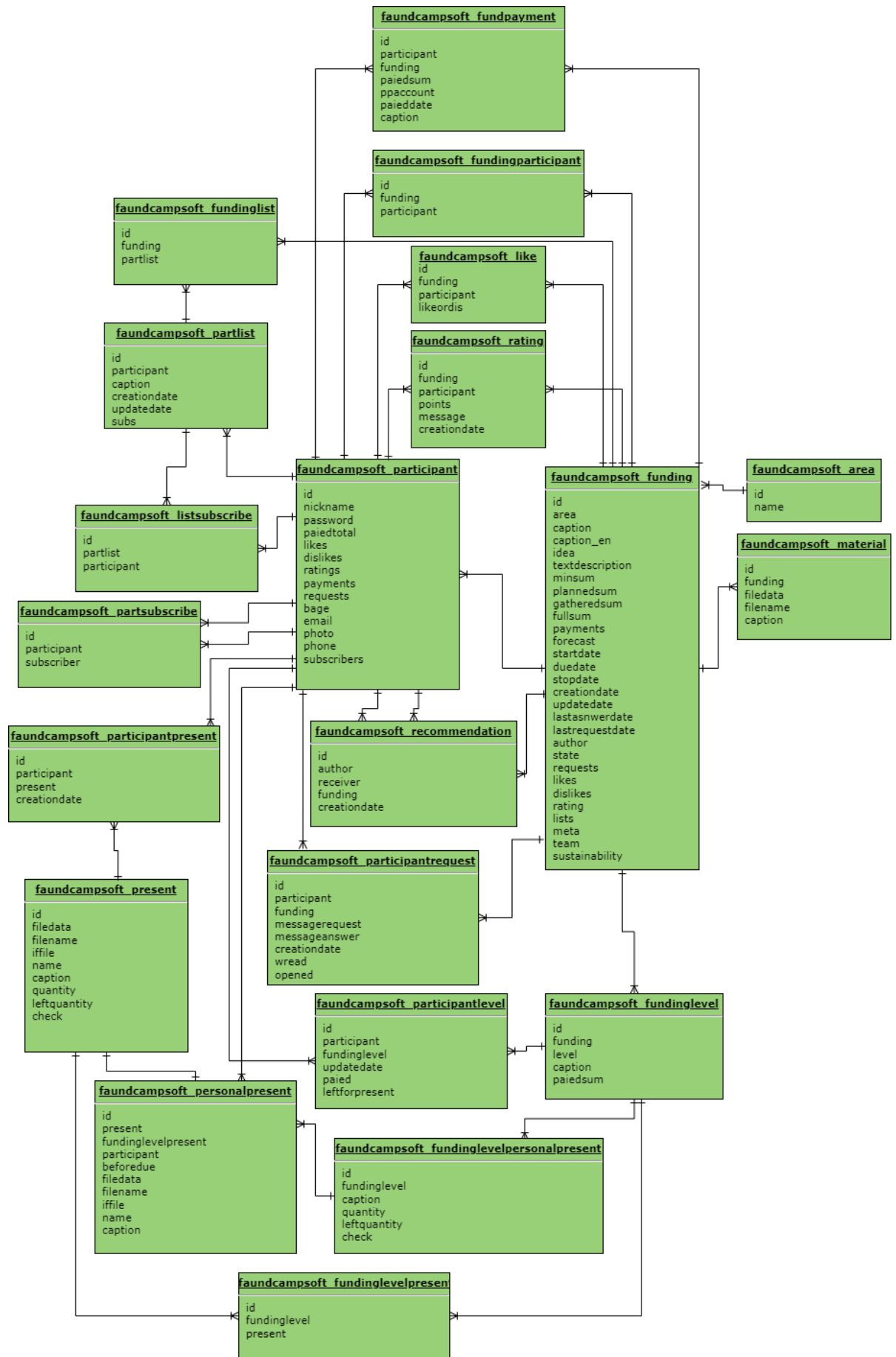


Рисунок 3.3 – Схема структуры БД

3.4 Висновки за розділом 3

У третьому розділі за результатами проєктування програми обрано сукупність рішень для проєктування і реалізації, що включають мову Python, фреймворк веброзробки Django, систему для роботи з БД MySQL, визначено і змодельовано роботу зі сценаріями користувачів програми, визначено структуру даних в БД.

4 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ

4.1 Порядок роботи над кампанією зі збору коштів для фінансування проєкту

Кампанія зі збору коштів розпочинається з моменту створення відповідного запису в системі, тоді вона знаходиться в стані «збережена» (рис. 4.1). Після того, як настає дата і час, коли має розпочатися збір коштів, кампанія переходить до стану «розпочата». З цього моменту інформація про проєкт доступна всім користувачам у програмі і вони можуть переводити кошти на фінансування проєкту.

Після того як кампанія розпочата, вона може перейти в один з наступних станів:

- якщо під час кампанії зібрана мінімальна потрібна сума, за якої власник готовий реалізувати хоча би частково проєкт, то кампанія переходить в стан «мінімальна»;
- якщо під час кампанії зібрана потрібна сума, але визначено, що вона має продовжувати тривати до завершення строку, то кампанія переходить до стану «триває успішно»;
- якщо строк кампанії вийшов, а потрібна сума зібрана, то кампанія переходить до стану «завершена успішно»;
- якщо строк кампанії завершився, але потрібну суму не вдалось зібрати, при чому мінімальна сума не була встановлена в кампанії або вона не була зібрана;
- якщо після завершення строку кампанії вдалось зібрати тільки мінімальну потрібну суму, то кампанія переходить до стану «завершена з мінімальним результатом»;
- кампанія була завершена вручну до завершення строку виконання, то кампанія переводиться до стану «скасована».

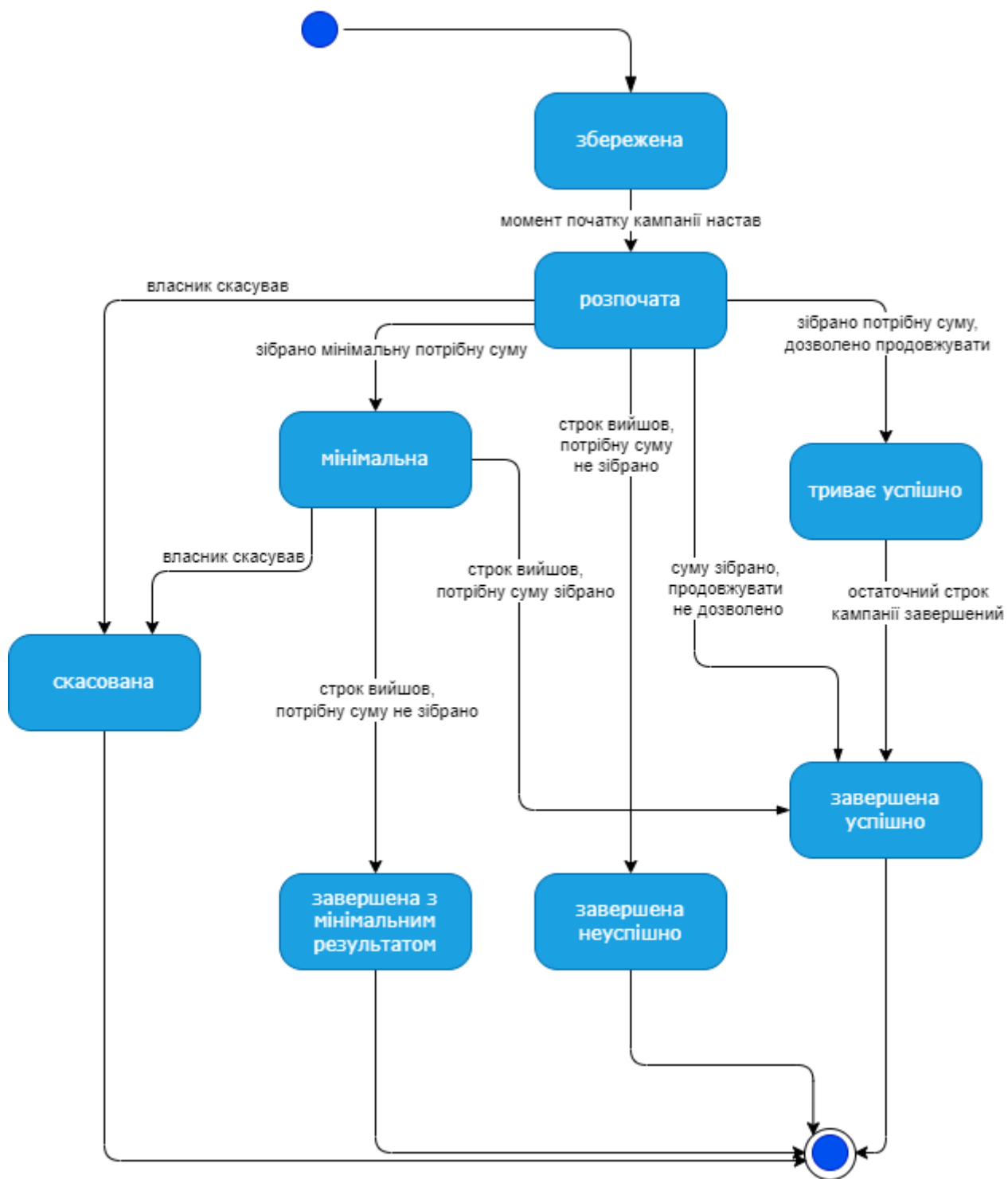


Рисунок 4.1 – Діаграма станів кампанії зі збору коштів для фінансування проекту

4.2 Основні реалізовані класи

Клас представлення `FundingView` формує дані для заповнення шаблону вебсторінки `funding.html`. Шаблони створені за допомогою

безкоштовного шаблону [17]. Використано клас `DetailView` для побудови даного класу на його основі. Під час роботи основного методу відбувається витягання даних про кампанію зі збору коштів за заданим номером як параметр `pk` методу `get`. Далі ці дані розширюються додатковими даними, до яких входить:

- позначка про те, чи є авторизований користувач автором цього проєкту і кампанії зі збору коштів, у вигляді значення за ключем `author`;
- перелік всіх уподобань проєкту у вигляді значення за ключем `likes`;
- перелік всіх невподобань проєкту у вигляді значення за ключем `dislikes`;
- позначка про те, чи вже вподобав цей проєкт авторизований зараз користувач, у вигляді значення за ключем `heart`;
- позначка про те, чи вже негативно поставився авторизований зараз користувач до цього проєкту, у вигляді значення за ключем `unheart`;
- перелік всіх оцінювань проєкту у вигляді значення за ключем `ratings`;
- позначка про те, чи вже оцінив авторизований зараз цей проєкт, у вигляді значення за ключем `rate`;
- позначка про те, чи підтримує цей користувач цей проєкт, у вигляді значення за ключем `part`;
- перелік з 5 останніми платежами за проєктом у вигляді значення за ключем `payments`;
- позначка про те, чи фінансував цей проєкт авторизований зараз користувач, у вигляді значення за ключем `paied`;
- перелік матеріалів проєкту у вигляді значення за ключем `materials`;
- перелік 5 останніх відкритих запитів за проєктом у вигляді значення за ключем `requests`;
- перелік рівнів участі користувачів у фінансуванні проєкту у вигляді значення за ключем `levels`;

– перелік доступних варіант персональних подарунків у вигляді значення за ключем `perspresents`.

Перелік відкритих запитів за проєктом формується класом представлення `PublicParticipantrequestsView` через наповнення сторінки шаблону `funding_pubrequests.html`. У даному представленні відбираються всі публічні запити за кампанією зі збору коштів на фінансування проєкту з номером `fund`, витягнутим з запиту до вебсторінки. Результати впорядковуються за датою створення запиту.

Перелік приватних запитів за проєктом формується класом представлення `PrivateParticipantrequestsView` через наповнення сторінки шаблону `funding_privrequests.html`. У даному представленні відбираються всі приватні запити за кампанією зі збору коштів на фінансування проєкту з номером `fund`, витягнутим з запиту до вебсторінки. Перевіряється можливість доступу користувача до цих даних. Для цього перевіряється, чи співпадає авторизований користувач з користувачем, який є власником кампанії. Усі результати впорядковуються за датою створення.

З іншого боку підтримка користувачів реалізується через клас представлення `OwnParticipantrequestsView` і шаблон вебсторінки `requests.html`. Він наповнює шаблон даними зі всіма запитами, які були створені користувачем, який авторизувався в системі. Результати впорядковуються за датою створення.

Клас представлення `FundingPaymentsView` заповнює весь перелік платежів за кампанією зі збору коштів на фінансування проєкту. Якщо авторизований користувач не є власником проєкту, то платежі не повертаються. Клас реалізований наслідуванням класу `ListView` як переліку даних та `LoginRequiredMixin`, що вимагає авторизації користувач для роботи з цим переліком. Через параметр запиту `fund` визначається номер кампанії зі збору коштів. Через параметр `direction` визначається варіант сортування всіх платежів, які були проведені за цим проєктом. Значення 0 визначає

сортування за датою сплати, а значення 1 визначає сортування за сплаченою сумою.

Клас представлення `ParticipantPaymentsView` відповідно формує перелік користувачів, які виконали платежі для фінансування даного проєкту, і передає його в шаблон вебсторінки `funding_payments_participant.html`. Клас побудований на основі моделі `Participant`. Спочатку витягаються всі платежі за кампанією з номером `fund`, що визначається з запиту до вебсторінки, а далі зі всіх платежів формується перелік користувачів, які їх виконали.

Клас представлення `OwnPartlistsViews` формує перелік власних переліків швидкого доступу до проєктів для заповнення шаблону вебсторінки `own_partlists.html`. Відповідні переліки для користувача з номером `user` з адреси вебсторінки формуються в класі представлення `PartlistsViews`.

Перелік всіх платежів на фінансування проєктів, які виконав авторизований користувач, дозволяє сформувати клас представлення `ParticipantFundingsView`.

Перелік підтримуваних авторизованим користувачем кампаній для збору коштів на фінансування проєктів формується на основі класу представлення `ParticipantManagedFundingsView`. Результируючий перелік вноситься в шаблон вебсторінки `participant_managed_fundings.html`.

Перелік кампаній зі збору коштів для фінансування проєктів, які належать авторизованому користувачу, формується в класі представлення `AuthorManagedFundingsView`.

Запуск кампанії зі збору коштів на фінансування проєктів реалізується через клас представлення `FundingCreateView`. Внесення даних про саму кампанію, про проєкт виконується через форму `FundingForm`, яка містить поля, необхідні для даних в таблиці кампанії. У методі `post` під час збереження отриманого об'єкту відбувається звернення до підпаketу проєкту `funforecaster`, в якому реалізовано запропонований метод. У цей момент створюється об'єкт класу `Forecaster` і викликається метод прогнозування

predict з передачею в якості параметра кампанії зі збору коштів funding. Після внесення відповідних даних відбувається збереження даних кампанії.

Редагування даних кампанії зі збору коштів виконується через клас представлення FundingUpdateView. Даний клас підтримує також і зміну стану в частині переведення кампанії до стану «розпочата» кампанія та «скасована» кампанія. Додатково відбувається перевірка авторства. Якщо користувач не є власником кампанії, то редагування йому забороняється.

Пошук проєктів підтримується класом представлення FundingSearchView. Параметрами, які визначають параметри пошуку, є значення, які витягаються з адреси вебсторінки, включаючи:

- текст для пошуку з параметру text: текст шукається або в назві кампанії, або в ідеї, в текстовому описі, в меті, в команді проєкту, в описі підтримки стабільності проєкту, відповідно ці дві групи результатів містяться окремо в fundings_caption і fundings_in відповідно;

- галузь проєкту з параметру area;

- задана мінімальна частка зібраних коштів на проєкт з параметру part;

- задана мінімальна частка часу виконання проєкту зі всього запланованого часу з параметру date;

- активність в останніх відповідях на запити користувачів з параметру ans: значення True вимагає перевірки і встановлення того факту, що остання відповідь на запит за проєктом тривала не більше 1 доби, значення False відповідно не вимагає такої перевірки;

- заданий стан кампанії з параметру state;

- мінімальний середній рейтинг кампанії з параметру rating;

- максимальний середній рейтинг кампанії з параметру maxrating;

- тип пошуку з параметру captioned: якщо встановлено значення 1, то результати пошуку в назві і в інших елементах залишаються окремо і не перемішуються, окремо сортуючись за рейтингом всередині кожної цієї

групи окремо, якщо встановлено значення 0, то всі проєкти об'єднуються разом і перевішуються, разом сортуючи їх за рейтингом.

Під час пошуку відкидаються всі кампанії, які є тільки створеними і скасовані. Усі інші беруть участь в початковому пошуку.

Створення рейтингу кампаній зі збору коштів виконується серед успішних кампаній на основі класу представлення `OverallFundingRatingView`. Впорядкування відбувається за рейтинговою оцінкою середньою або за зібраною сумою (абсолютним значенням). 100 найкращих за відповідним критерієм кампаній повертаються в результаті.

Створення рейтингу діючих кампаній зі збору коштів виконується на основі класу представлення `OverallUnFundingRatingView`. Впорядкування відбувається за рейтинговою оцінкою середньою або за зібраною сумою (абсолютним значенням або відносним значенням). 100 найкращих за відповідним критерієм кампаній повертаються в результаті. Варіант сортування розглядається при цьому ще один додатковий порівняно з попереднім класом — це впорядкування за відносною часткою зібраних коштів на поточний момент.

Клас представлення `RatedFundingsView` формує перелік оцінених авторизованим користувачем проєктів. Відповідно перелік уподобаних проєктів забезпечується через клас представлення `LikedFundingsView`. Перелік невподобаних проєктів забезпечується через клас представлення `DislikedFundingsView`.

Підтримка перегляду переліку кампаній для швидкого доступу виконується через клас представлення `PartlistView`. Дані самого переліку містяться на сторінці через ключ `partlist`, а перелік самих кампаній міститься за ключем `fundings`.

Клас представлення `ParticipantlevelView` формує деталізацію даних про кампанію зі збору коштів, надаючи можливі рівні участі користувачів у фінансуванні заданого проєкту, а для кожного рівня повертаючи всіх таких користувачів, які досягли цього рівня.

Клас представлення `RatingCreateView` підтримує збереження оцінювання проєкту. Передбачається спочатку перевірка авторизованого користувача. Тільки якщо цей користувач виконав якийсь платіж на фінансування проєкту і якщо він ще не оцінював цей проєкт, йому надається можливість створити і зберегти це оцінювання.

Клас представлення `PartlistCreateView` підтримує створення переліку швидкого доступу, а клас `PartlistUpdateView` – його редагування.

Уподобання кампанії зі збору коштів на фінансування проєкту виконується через клас представлення `FundingLikeView`. Автоматично відбувається перевірка тих параметрів уподобання, які вже були встановлені за кампанією раніше. Якщо проєкт було уподобано, то уподобання знімається. Якщо проєкт було неуподобано, то кількість неуподобань зменшується, а кількість уподобань в результаті збільшується. Якщо ніякої інформації такої ще не було внесено, то створюється нове уподобання, збільшується кількість уподобань. Аналогічні дії виконуються в представленні `FundingDislikeView` для створення неуподобання.

Створення запиту за проєктом виконується через клас представлення `RequestCreateView`, а створення відповідні на запит відбувається через клас представлення `AnswerCreateView`.

Так само влаштовані інші класи представлень.

4.3 Висновки за розділом 4

Порядок роботи над кампанією зі збору коштів для фінансування проєкту розглянуто як в текстовому описі, так і у вигляді діаграми станів. Реалізовано програмне забезпечення за допомогою фреймворку Django і описано класи представлень, які підтримують функціональні можливості програми.

5 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

5.1 Опис роботи з програмою

Взаємодія з програмою відбувається в одному з двох режимах. У першому режимі, який відкривається при зверненні до вебсервера, користувач не виконав авторизацію і працює тільки з загальними можливостями. Для цього в меню йому доступні розділи «Пошук», де можна визначити параметри проєкту, який цікавить користувача, отримати результати, звідки вже перейти до конкретного проєкту (рис. 5.1), який зацікавив користувача, «Завершені», де виводиться рейтинг зі 100 найкращих завершених проєктів, «Незавершені», де обирається один з варіантів рейтингу незавершених проєктів (за часткою зібраних коштів або за абсолютним значенням).

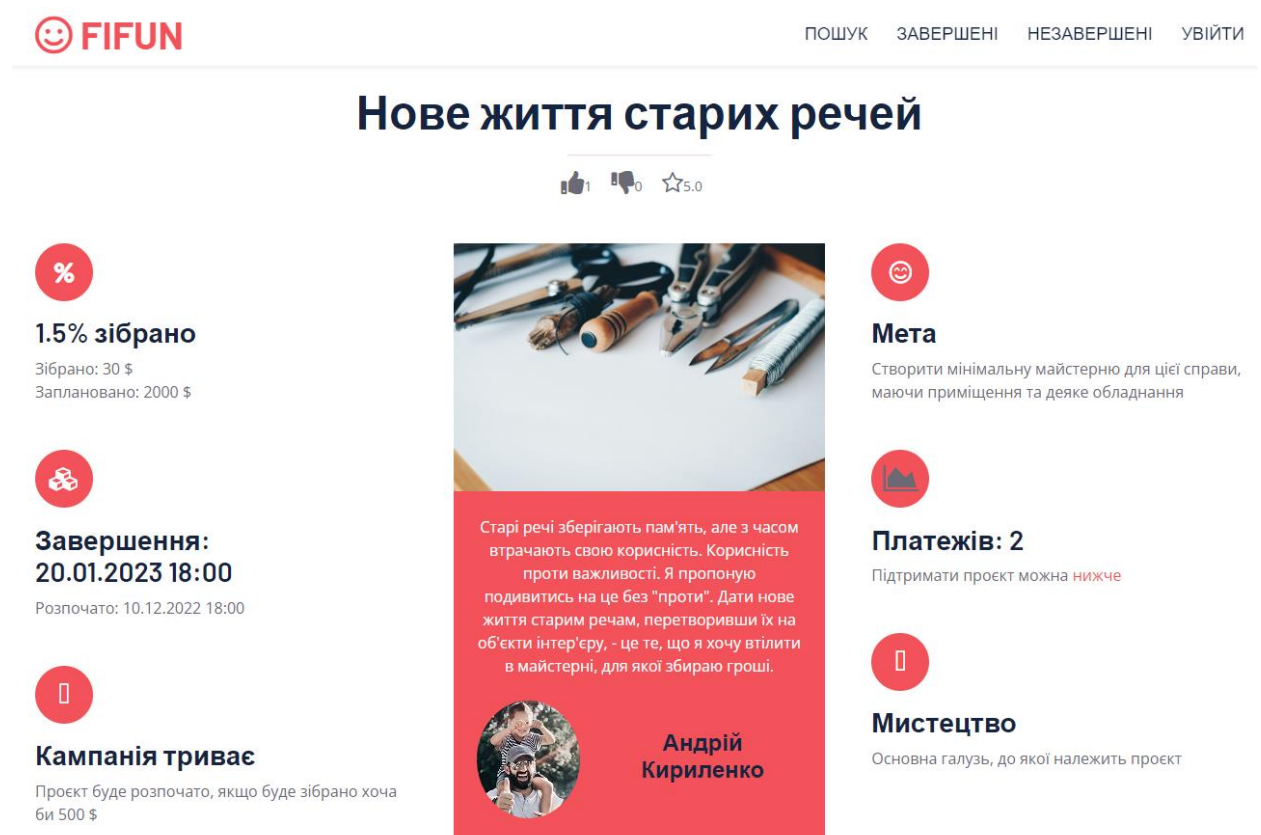


Рисунок 5.1 – Сторінка ознайомлення з кампанією зі збору коштів на фінансування проєкту

На сторінці проєкту, де представлена кампанія зі збору коштів для його фінансування, приведено назву проєкту, відсоток, а також абсолютні числа зібраних коштів для проєкту і запланованих значень, дату і час завершення кампанії та дату і час початку кампанії зі збору коштів, стан кампанії, опис ідеї, інформацію про автора (натискаючи на ім'я, можна перейти на сторінку активності даного користувача, звідки, наприклад, переглянути переліки швидкого доступу до проєктів, створені цим користувачем), мету проєкту, його галузь, кількість отриманих платежів на фінансування, опис проєкту, команду проєкту. Також розташовано перелік можливих рівнів участі в фінансуванні проєкту та форму з введенням суми платежу. Після введення потрібної суми необхідно натиснути на кнопку «Сплатити», після чого відбудеться переведення на сторінку виконання платежу, де необхідно задати дані свого рахунку PayPal.

Меню авторизованого користувача набагато ширше, а всі приведені розділи для неавторизованого користувача зведені до розділу «Пошук», який має підрозділи, до яких і входять всі описані вище.

Розділ меню «Мої» дозволяє переглянути перелік усіх власних проєктів, тобто проєктів і кампаній зі збору коштів на них, для яких автором є авторизований користувач. Якщо користувач є не автором, а тільки доданий автором для підтримки кампанії, то такі проєкти знаходяться в розділі «Підтримка».

На сторінці власних кампаній (рис. 5.2) приводяться всі кампанії зі збору коштів на фінансування проєкту на основі загальної інформації і основних керуючих кнопок. Інші керуючі кнопки приведені на сторінці кампанії в режимі для власника. Для кожної кампанії доступна її назва, основна фотографія (всі інші фотографії та матеріали за проєктом приводяться на сторінці ознайомлення з кампанією згори), кількість платежів і галузь (над назвою), зібрана сума, запланована для збору сума (нижче назви), кількість уподобань, кількість невподобань, середня виставлена оцінка.

Ваші кампанії зі збору коштів

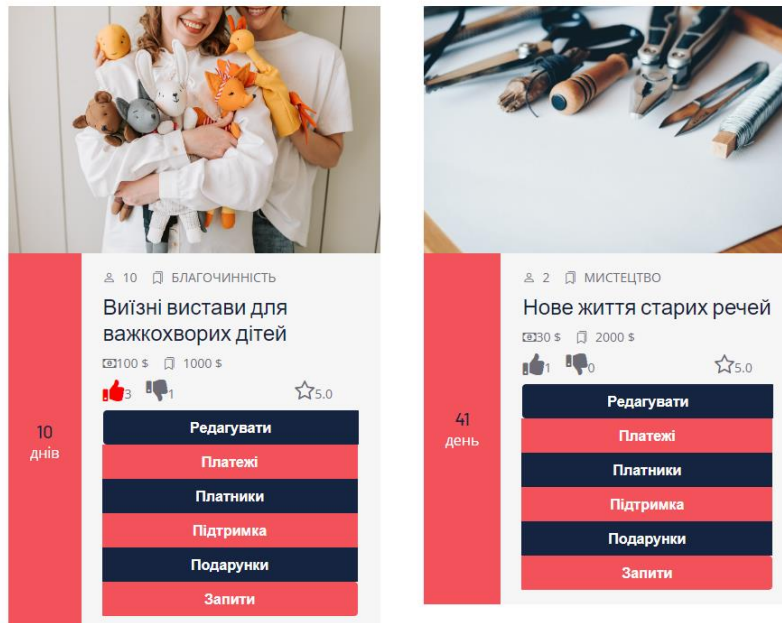


Рисунок 5.2 – Сторінка керування власними кампаніями зі збору коштів

Власник для кожного свого проєкту може на основі наступних кнопок виконати такі дії:

- кнопка «Редагувати» дозволяє перейти до редагування параметрів кампанії і пов’язаних елементів;
- кнопка «Платежі» дозволяє переглянути всі платежі, отримані на фінансування цього проєкту;
- кнопка «Платники» дозволяє переглягнути вже не платежі, а користувачів, які профінансували проєкт;
- кнопка «Підтримка» дозволяє перейти до переліку користувачів, які підтримують кампанію, тобто мають право відповідати на запити інших користувачів, додавати загальні і персональні подарунки;
- кнопка «Подарунки» дозволяє перейти до визначення загальних і персональних подарунків, які надаються користувачам за фінансування проєкту, з визначенням відповідних рівнів участі в фінансуванні проєкту;

– кнопка «Запити» дозволяє перейти до запитів, які були поставлені користувачами стосовно цього проєкту.

Розділ меню «Платежі» містить перелік усіх платежів за всіма кампаніями зі збору коштів, в яких взяв участь авторизований користувач.

Розділ меню «Запити» містить перелік усіх запитів, які створив авторизований користувач, разом з відповідями, якщо вони були надані.

Розділ меню «Проекти» містить декілька розділів, що дозволяють обрати варіант переліку проєктів. До розділів належать «Улюблені» для відображення переліку проєктів, які вподобав авторизований користувач, «Неулюблені» для відображення переліку проєктів, які користувач позначив як невподобані, «Оцінені» для відображення переліку проєктів, які користувач оцінив, «Рекомендовані» для відображення переліків проєктів, які користувачу рекомендував інший користувач, «Переліки» для відображення власних переліків швидкого доступу, «Підписані» для відображення переліків швидкого доступу, на які підписався користувач.

Розділ меню «Подарунки» містить перелік усіх загальних і персональних подарунків за проєктами, які профінансував користувач. Вибір конкретного варіанту виконується кнопкою. Подарунками можуть бути і фактично матеріали, доступні користувачам, які допомогли з фінансуванням проєкту.

Розділ меню «Профіль» дозволяє керувати власними даними.

Для власника кампанії зі збору коштів на сторінці кампанії згори відображаються результати прогнозування тривалості кампанії для збору коштів на фінансування проєкту (рис. 5.3). Окрім того можна ввести інше значення суми коштів, запланованої тривалості кампанії і отримати новий прогноз. Якщо кампанія ще не стартувала, а власник бажає зберегти ці значення даних показників, то він може зберегти ці зміни в БД.

Портативна пральна машина

Задайте значення параметрів для виконання прогнозування.

Запланована сума
27000

Запланована тривалість
60

Прогнозувати

Продуктивність
Категорія задається одразу під час створення кампанії і не може бути змінена в подальшому.

47 днів
Прогнозована тривалість кампанії на основі заданих характеристик кампанії

27000 \$ за 60 днів
Збережені на даний момент заплановані значення характеристик кампанії

Рисунок 5.3 – Сторінка прогнозування для кампанії зі збору коштів на фінансування проєкту, для якого авторизований користувач є власником

5.2 Експериментальне дослідження

Виконання експериментального дослідження проводилось на основі застосування вибірки даних [18]. У ній представлені дані про результати збору коштів на фінансування проєктів, витягнуті з одного з відповідних сервісів. Вибірка містить наступні ознаки:

- валюту, в якій виконувався збір коштів;
- галузь знань, до якої відноситься проєкт;
- дата і час закінчення кампанії;
- зібрана сума;
- відсоток зібраних коштів відносно запланованих;
- дата і час початку кампанії;
- номер проєкту;
- назва проєкту;
- адреса вебсторінки з проєктом;
- зібрана сума в доларах США;

- запланована для збору сума коштів;
- країна проєкту.

Аналізуючи наявні дані, було виявлено, що відсутня пряма залежність між тривалістю кампанії та зібраною сумою, проте очевидно, що певний вплив існує, при цьому цей вплив відрізняється в залежності від галузей знань. Для галузі мистецтва (рис. 5.4), культури (рис. 5.5) цей зв'язок є більш вираженим, а для галузі продуктивності (рис. 5.6) більше присутньо значень, які не підпадають під загальну тенденцію. Так для частини проєктів потрібну суму було зібрано за строк не більше 1 доби. Вивчаючи таким чином залежності тривалості кампанії від різних ознак, було визначено набір ознак, який сформував вибірку, що використовувалась в описі запропонованого методу.

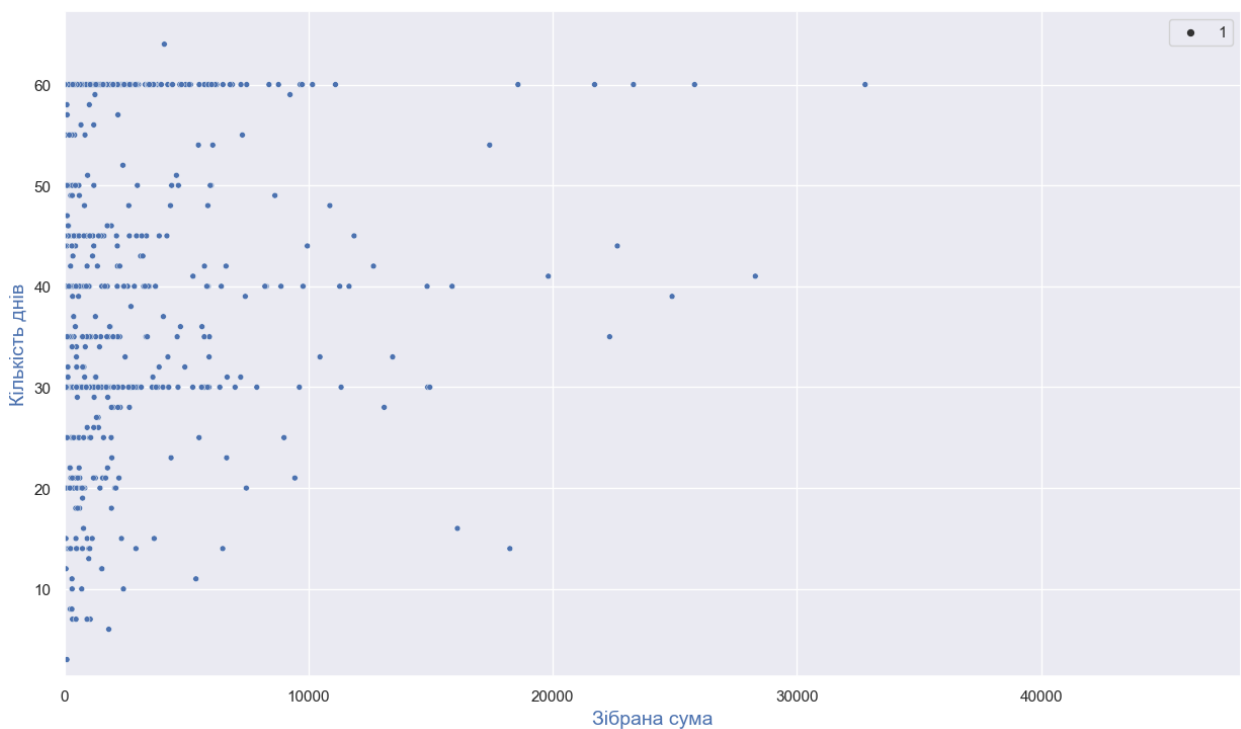


Рисунок 5.4 – Графік розподілу кількості днів збору коштів в залежності від зібраної суми на фінансування проєкту для галузі мистецтва

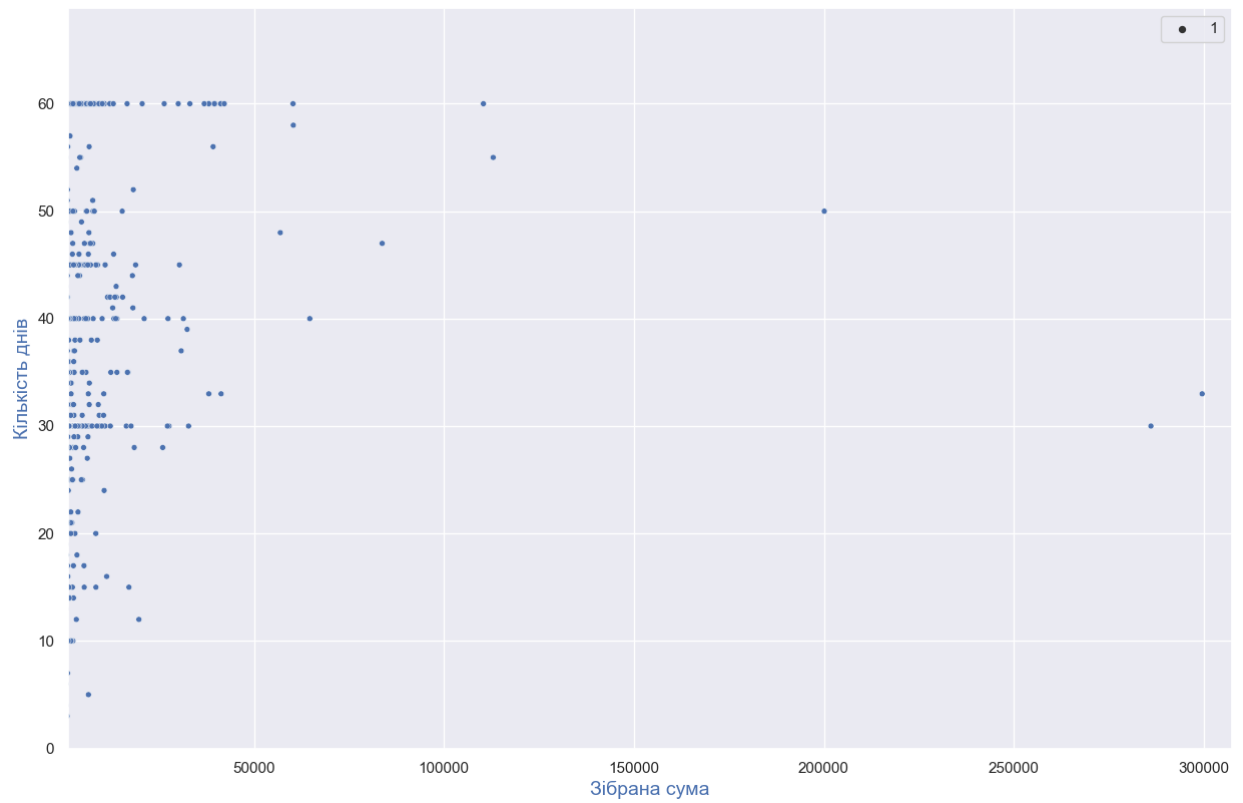


Рисунок 5.5 – Графік розподілу кількості днів збору коштів в залежності від зібраної суми на фінансування проєкту для галузі культура

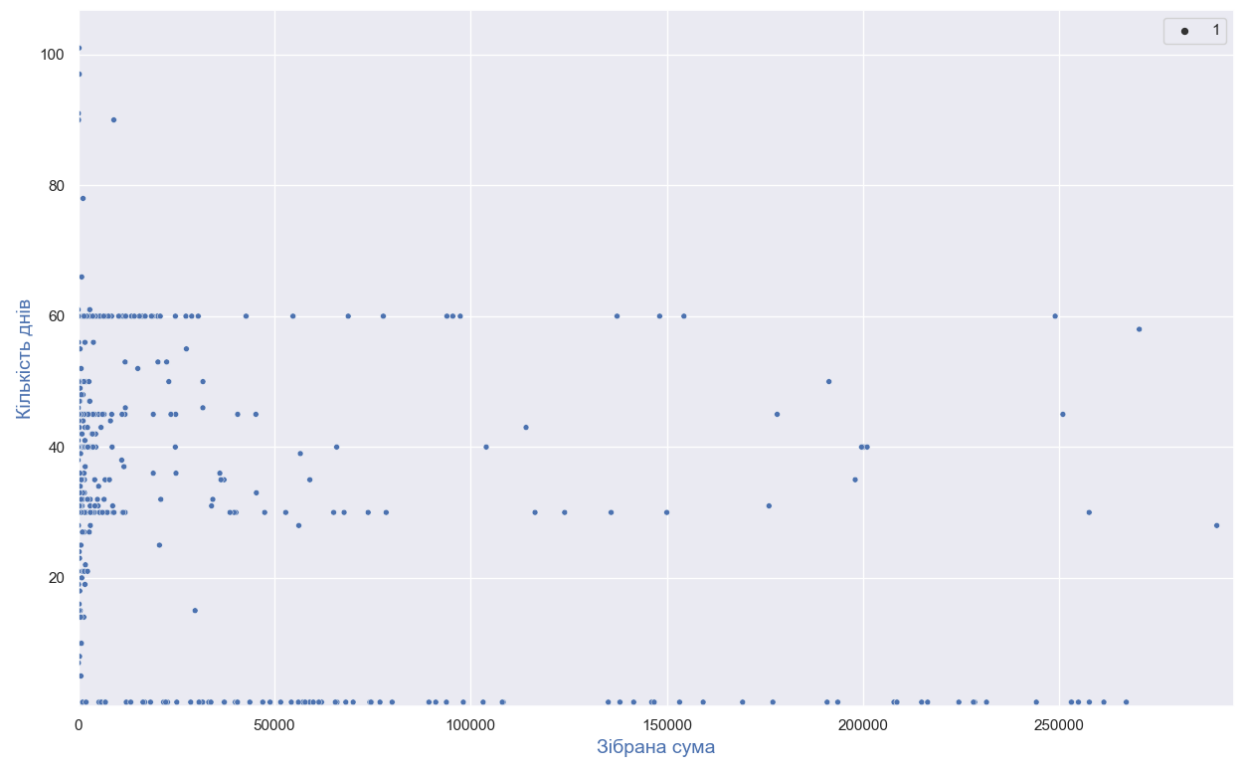


Рисунок 5.6 – Графік розподілу кількості днів збору коштів в залежності від зібраної суми на фінансування проєкту для галузі продуктивність

Використовуючи метод прогнозування тривалості кампанії для збору коштів на фінансування проєктів та представлену вище вибірку даних, а також лінійну регресійну модель як альтернативу, було виконано порівняння точності прогнозування. Точність прогнозування оцінювалась на основі коефіцієнту детермінації. Запропонований метод за даним показником був оцінений у 0,801. Лінійна регресійна модель дозволила отримати значення 0,553 (табл. 5.1).

Таблиця 5.1 – Результати експериментального дослідження

Метод / модель	Точність
Лінійна регресійна модель	0,553
Метод прогнозування тривалості кампанії для збору коштів на фінансування проєктів	0,801

5.3 Висновки за розділом 5

Створена програма дозволила реалізувати всі заплановані сценарії роботи користувачів. Експериментальне дослідження виконано на основі вибірки даних, представлених у відкритому доступі. Запропонований метод проявив себе достатньо ефективно в даному дослідженні.

ВИСНОВКИ

За результатами виконання роботи розроблено програмне забезпечення збору коштів на фінансування проєктів, що підтримує прийняття рішень автором проєкту щодо встановлення основних параметрів кампанії зі збору коштів, надаючи при цьому широкий набір інструментів зі взаємодії з цими проєктами. Основні функціональні вимоги до програми виділено на основі аналізу програмних засобів, які на даний момент є відомими в цій сфері, після чого вони були застосовані для моделювання сценаріїв роботи за допомогою діаграм прецедентів.

Для реалізації програми використано мову Python і фреймворк веброзробки Django, що забезпечив визначення структури розробленої програми, з якої детально описано основні класи, систему керування базами даних MySQL.

Представлене програмне забезпечення дозволяє організовувати кампанії зі збору коштів на фінансування проєктів, керувати цими кампаніями, обмежувати доступ до матеріалів проєкту користувачам в залежності від того, якою сумою вони профінансували проєкт, а користувачам у свою чергу шукати проєкти, брати участь у фінансуванні проєктів з отриманням різного роду подарунків, матеріалів, створювати переліки швидкого доступу, рекомендувати проєкти іншим користувачам.

Наукова новизна роботи сформована в методі прогнозування тривалості кампанії для збору коштів на фінансування проєктів, який виконує визначення строку, необхідного на збір певної суми коштів на фінансування конкретного проєкту, до початку кампанії, використовуючи дерева ухвалення рішень на основі алгоритму CART, що дозволяє збалансувати встановлені суму коштів кампанії та тривалість такої кампанії, щоб в підсумку вона з більшою вірогідністю завершилась успішно.

Виконання роботи було побудовано на основі сформульованого переліку завдань, кожен пункт якого було виконано.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Understanding the Funding Model for raising funds from High Net individuals (HNI) [Electronic resource]. – Access mode : <https://www2.fundsforngos.org/featured/understanding-the-funding-model-for-raising-funds-from-high-net-individuals-hni/>.
2. Sargeant, A. Fundraising Management [Text] : Analysis, Planning and Practice / A. Sargeant. – London : Routledge, 2014. – 456 p.
3. A Guide to Different Types of Fundraising [Electronic resource] : Pros and Cons. – Access mode : <https://donorbox.org/nonprofit-blog/different-types-of-fundraising>.
4. Everything About Nonprofit Funding Models [Electronic resource]. – Access mode : <https://whydonate.nl/en/blog/nonprofit-funding-models/>.
5. Foster, W. L. Ten Nonprofit Funding Models [Electronic resource] / W. L. Foster, P. Kim, B. Christiansen. – Access mode : https://ssir.org/articles/entry/ten_nonprofit_funding_models.
6. Next Generation Online Fundraising Platform FundRazr [Electronic resource]. – Access mode : <https://fundrazr.com/>.
7. Kearl, M. Best Alternatives to Kickstarter [Electronic resource] / M. Kearl. – Access mode : <https://www.investopedia.com/best-alternatives-to-kickstarter-5081260>.
8. Найкращий спосіб для митців та авторів отримувати обґрунтований дохід і підтримувати зв'язок із шанувальниками. Patreon [Electronic resource]. – Access mode : <https://www.patreon.com/>.
9. Kickstarter [Electronic resource]. – Access mode : <https://www.kickstarter.com/?ref=nav>.
10. Wang, W. Prediction of fundraising outcomes for crowdfunding projects based on deep learning [Text] : a multimodel comparative study / W. Wang, H. Zheng, Y. J. Wu // Soft Computing. – 2020. – Vol. 24 (11). – P. 8323-8341.

11. Predicting Fundraising Performance in Medical Crowdfunding Campaigns Using Machine Learning [Electronic resource] / Nianjiao Peng, Xinlei Zhou, Ben Niu, and Yuanyue Feng // Electronics. – 2021. – Vol. 10 (2). – Access mode : https://www.researchgate.net/publication/348401106_Predicting_Fundraising_Performance_in_Medical_Crowdfunding_Campaigns_Using_Machine_Learning.

12. Why Do We Use Decision Trees in Machine Learning? [Electronic resource]. – Access mode : <https://www.turing.com/kb/importance-of-decision-trees-in-machine-learning>.

13. Waraich, K. Classification And Regression Tree (CART) Algorithm [Electronic resource] / K. Waraich. – Access mode : <https://www.codingninjas.com/codestudio/library/classification-and-regression-tree-algorithm>.

14. CART (Classification And Regression Tree) in Machine Learning [Electronic resource]. – Access mode : <https://www.geeksforgeeks.org/cart-classification-and-regression-tree-in-machine-learning/>.

15. Jithin, J. ID3, C4.5, CART and Pruning [Electronic resource] / J. Jithin. – Access mode : <https://bitmask93.github.io/ml-blog/ID3-C4-5-CART-and-Pruning/>.

16. Chouinard, J.-C. Decision Trees in Machine Learning, with Examples (Python) [Electronic resource] / J.-C. Chouinard. – Access mode : <https://www.jcchouinard.com/decision-trees-in-machine-learning/>.

17. CONSULT Free Website Template. Free CSS Templates. Free CSS [Electronic resource]. – Access mode : <https://www.free-css.com/free-css-templates/page286/consult>.

18. Mcteer, Q. Indiegogo Crowdfunding Campaigns [Electronic resource] / Q. Mcteer. – Access mode : <https://www.kaggle.com/datasets/quentinmcteer/indiegogo-crowdfunding-data>.

ДОДАТОК А
Технічне завдання

Вступ

В Україні кампанії зі збору коштів в умовах воєнного стану стали широко розповсюдженими. Кошти часом збираються певним фондом, який потім вирішує, куди їх витратити, а часом на певну задачу, яку цілісно можна розглядати як певний проєкт. Звичайно, що проєкти, на які збираються кошти, можуть бути не тільки воєнними, але і такими, що представляють будь-які потенційно цікаві ідеї. Це можуть бути нові пристрої, нові технології, нові ініціативи людей. Для підтримки пов'язаних з цим процесі і необхідне програмне забезпечення, що розробляється. За наявності інструментів прогнозування програма в підсумку полегшить роботу власників цих проєктів.

А.1 Підстави для розробки

Розробка дипломної кваліфікаційної роботи магістра виконується на основі завдання за темою «Дослідження та програмна реалізація методів прогнозування тривалості кампанії для збору коштів на фінансування проєктів», затвердженою наказом № 363 від 4 листопада 2022 р. за Національним університетом «Запорізька політехніка».

А.2 Призначення розробки

Програма призначена для організації і проведення кампаній для збору коштів на фінансування проєктів з можливостями надання подарунків користувачам, які профінансували проєкти, та прогнозуванням тривалості такої кампанії.

A.3 Основні вимоги до програми

A.3.1 Вимоги до функціональних характеристик

Для забезпечення власного призначення програма повинна дозволяти виконувати функції:

- пошук проєктів;
- демонстрація найкращих завершених або незавершених кампаній за галуззю;
- демонстрація кампаній, які мають найкращу динаміку збору;
- запуск кампанії зі збору коштів на фінансування проєкту;
- ознайомлення з переліком проєктів для швидкого доступу;
- редагування параметрів кампанії зі збору коштів на фінансування проєкту;
- ознайомлення з кампанією зі збору коштів для фінансування проєкту;
- перегляд платежів за кампанією зі збору коштів на фінансування проєкту;
- перегляд власних платежів на фінансування проєктів;
- перегляд власних переліків проєктів для швидкого доступу;
- перегляд переліків проєктів для швидкого доступу за користувачем;
- перегляд відкритих та приватних запитів за проєктом;
- перегляд усіх власних запитів за всіма проєктами;
- перегляд власних або підтримуваних кампаній зі збору коштів на фінансування проєкту;
- визначення користувачів, які підтримують кампанію зі збору коштів на фінансування проєкту;
- створення персонального подарунку для користувача;
- вподобання або пряме невподобання проєкту, збір коштів на фінансування якого відбувається;
- оцінювання проєкту, який профінансував користувач;

- формування переліку вподобаних або несподобаних користувачем проєктів;
- формування переліку оцінених проєктів користувачем;
- виконання платежу на фінансування проєкту;
- формування переліку профінансованих користувачем проєктів;
- надання користувачу подарунку за фінансування проєкту;
- формування переліку подарунків користувача;
- вибір користувачем подарунку з доступних варіантів;
- відправлення користувачем відкритого або приватного запиту за проєктом;
- відправлення відповіді на запит;
- персональне рекомендування проєкту;
- формування переліку рекомендованих проєктів;
- створення переліку проєктів для швидкого доступу;
- додавання або вилучення проєктів з переліку швидкого доступу;
- створення підписки на користувача;
- скасування вподобання проєкту;
- скасування несподобання проєкту;
- формування переліків користувачів, на які підписався авторизований користувач або які підписалися на авторизованого користувача;
- перегляд сторінки активності користувача;
- створення підписки на перелік проєктів користувача.

А.3.2 Вимоги до складу та параметрів технічних засобів

Доступ з клієнтського пристрою користувача, який виконує будь-яку роль у вебдодатку, виконується за наявності підключення до мереж інтернет та за наявності актуальної версії будь-якого сучасного браузера. Для запуску вебсервера, який буде підтримувати роботу вебдодатку безпосередньо,

зокрема запускаючи і виконуючи метод, який має бути запропонований в роботі, необхідно скористатися послугами хостингу з можливостями підтримки роботи з системою керування БД MySQL, або підтримувати власний вебсервер, при цьому він повинен бути з оперативною пам'яттю з мінімальним обсягом 4 Гб, жорстким диском з мінімальним обсягом 1 Гб та щонайменше чотирьохядерним процесором (частота не менше 2 ГГц).

A.3.3 Умови експлуатації

Для розгортання вебдодатку на сервері з подальшим доступом користувачів знадобиться вебсервер. В якості стандартного рішення даної проблеми для додатків, розроблених за допомогою Django, використовується uWSGI. Вебсервер використовується як шлюз і фактично перенаправляє запити, які надходять до нього з клієнтських пристроїв, у Django-додаток.

A.3.4 Вимоги до програмної документації

Технічне завдання і текст програми, що повинен вміщувати фрагменти програми, оформлюються як окремі програмні документи, а опис роботи з програмою представляється в розділі 5 пояснювальної записки.

A.4 Порядок контролю та приймання

Контроль і приймання роботи мають відбуватися за календарним планом, який входить до складу завдання на дипломну кваліфікаційну роботу магістра.

ДОДАТОК Б
Текст програми

```

from django.contrib.auth.mixins import LoginRequiredMixin
from django.shortcuts import render
from django.views.generic import DetailView
from django.views.generic import ListView
from fifun.forms import FundingForm
from fifun.forms import RateForm
from fifun.forms import SearchFundingForm
from fifun.forms import RequestForm
from fifun.forms import PartlistForm
from django.views import View
from django.shortcuts import get_object_or_404
from django.shortcuts import redirect
from django.urls import reverse
from fifun.models import Funding
from fifun.models import Like
from fifun.models import Participant
from fifun.models import Rating
from fifun.models import Fundingparticipant
from fifun.models import Fundpayment
from fifun.models import Material
from fifun.models import Participantrequest
from fifun.models import Fundinglevel
from fifun.models import Present
from fifun.models import Fundinglevelpresent
from fifun.models import Fundinglevelpersonalpresent
from fifun.models import Recommendation
from fifun.models import Partlist
from fifun.models import Fundinglist
from django.http import HttpResponse
from django.db.models import F
from django.db.models import Q
from fifun.funforecaster import Forecaster
import datetime

class FundingView(DetailView):
    model = Funding
    template_name = 'fifun/funding.html'

    def get(self, request, pk):
        fund = Funding.objects.get(id = pk)
        this_participant = self.request.user
        this_participant = Participant.objects.get(user =
this_participant)
        if fund.author == this_participant:
            author = True
        else:

```

```

        author = False
        all_likes = Like.objects.filter(funding = fund,
likeordis = True)
        all_dislikes = Like.objects.filter(funding = fund,
likeordis = False)
        if all_likes.get(participant = this_participant):
            heart = True
        else:
            heart = False
        if all_dislikes.get(participant =
this_participant):
            unheart = True
        else:
            unheart = False
        ratings = Rating.objects.filter(funding = fund)
        if ratings.get(participant = this_participant):
            rate = True
        else:
            rate = False
        parts = Fundingparticipant.objects.filter(funding =
fund, participant = this_participant)
        if parts:
            part = True
        else:
            part = False
        c = Fundpayment.objects.filter(funding =
fund).order_by('-paieddate')[0:5]
        pays = Fundpayment.objects.filter(funding = fund,
participant = this_participant)
        if pays:
            paied = True
        else:
            paied = False
        materials = Material.objects.filter(funding = fund)
        requests =
Participantrequest.objects.filter(funding = fund, opened =
True).order_by('-creationdate')[0:5]
        fundlevels = Fundinglevel.objects.filter(funding =
fund)
        flpresents =
Fundinglevelpresent.objects.filter(fundinglevel__in =
fundlevels)
        perspresents =
Fundinglevelpersonalpresent.objects.filter(fundinglevel__in =
fundlevels)
        funding_context = {}

```

```

        funding_context ['likes'] = all_likes
        funding_context ['dislikes'] = all_dislikes
        funding_context ['heart'] = heart
        funding_context ['unheart'] = unheart
        funding_context ['ratings'] = ratings
        funding_context ['rate'] = rate
        funding_context ['part'] = part
        funding_context ['payments'] = payments
        funding_context ['paied'] = paied
        funding_context ['materials'] = materials
        funding_context ['requests'] = requests
        funding_context ['presents'] = flpresents
        funding_context ['levels'] = fundlevels
        funding_context ['perspresents'] = perspresents
        funding_context ['author'] = author
        funding_context ['fund'] = fund
        return render(request, self.template_name,
funding_context)

    class PublicParticipantrequestsView(LoginRequiredMixin,
ListView):
        model = Participantrequest
        template_name = 'fifun/funding_pubrequests.html'

        def get_queryset(self, *args, **kwargs):
            requests = super().get_queryset(*args, **kwargs)
            fund = int(self.request.GET.get('fund'))
            fund = Funding.objects.get(id = fund)
            this_participant = self.request.user
            this_participant = Participant.objects.get(user =
this_participant)
            requests = requests.filter(funding = fund, opened =
True)

            requests = requests.order_by('-creationdate')
            return requests

    class PrivateParticipantrequestsView(LoginRequiredMixin,
ListView):
        model = Participantrequest
        template_name = 'fifun/funding_privrequests.html'

        def get_queryset(self, *args, **kwargs):
            requests = super().get_queryset(*args, **kwargs)
            fund = int(self.request.GET.get('fund'))
            fund = Funding.objects.get(id = fund)
            this_participant = self.request.user

```

```

        this_participant = Participant.objects.get(user =
this_participant)
        if fund.author == this_participant:
            requests = requests.filter(funding = fund,
opened = False)
            requests = requests.order_by('-creationdate')
        else:
            requests = []
        return requests

```

```

class OwnParticipantrequestsView(LoginRequiredMixin,
ListView):
    model = Participantrequest
    template_name = 'fifun/requests.html'

    def get_queryset(self, *args, **kwargs):
        requests = super().get_queryset(*args, **kwargs)
        fund = int(self.request.GET.get('fund'))
        fund = Funding.objects.get(id = fund)
        this_participant = self.request.user
        this_participant = Participant.objects.get(user =
this_participant)
        requests = requests.filter(participant =
this_participant)
        requests = requests.order_by('-creationdate')
        return requests

```

```

class FundingPaymentsView(LoginRequiredMixin, ListView):
    model = Fundpayment
    template_name = 'fifun/funding_payments.html'

    def get_queryset(self, *args, **kwargs):
        payments = super().get_queryset(*args, **kwargs)
        fund = int(self.request.GET.get('fund'))
        fund = Funding.objects.get(id = fund)
        this_participant = self.request.user
        this_participant = Participant.objects.get(user =
this_participant)
        if fund.author != this_participant:
            return []
        pays = Fundpayment.objects.filter(funding = fund)
        direction = int(self.request.GET.get('direction'))
        if direction == 0:
            pays = pays.order_by('-paieddate')
        elif direction == 1:
            pays = pays.order_by('-paiedsum')

```

```

        return pays

    class ParticipantPaymentsView(LoginRequiredMixin,
    ListView):
        model = Participant
        template_name =
        'fifun/funding_payments_participant.html'

        def get_queryset(self, *args, **kwargs):
            participants = super().get_queryset(*args,
            **kwargs)

            fund = int(self.request.GET.get('fund'))
            fund = Funding.objects.get(id = fund)
            this_participant = self.request.user
            this_participant = Participant.objects.get(user =
            this_participant)
            if fund.author != this_participant:
                return []
            pays = Fundpayment.objects.filter(funding =
            fund).order_by('-paieddate')
            parts = []
            for p in pays:
                if p.participant not in parts:
                    parts.append(p)
            return parts

    class OwnPartlistsViews(LoginRequiredMixin, ListView):
        model = Partlist
        template_name = 'fifun/own_partlists.html'

        def get_queryset(self, *args, **kwargs):
            parts = super().get_queryset(*args, **kwargs)
            this_participant = self.request.user
            this_participant = Participant.objects.get(user =
            this_participant)
            parts = parts.filter(participant =
            this_participant)
            parts = parts.order_by('-creationdate')
            return parts

    class PartlistsViews(LoginRequiredMixin, ListView):
        model = Partlist
        template_name = 'fifun/partlists.html'

        def get_queryset(self, *args, **kwargs):
            parts = super().get_queryset(*args, **kwargs)

```

```

        part = float(self.request.GET.get('user'))
        this_participant = Participant.objects.get(user =
part)

        parts = parts.filter(participant =
this_participant)
        parts = parts.order_by('-creationdate')
        return parts

class ParticipantFundingsView(LoginRequiredMixin,
ListView):
    model = Funding
    template_name = 'fifun/participant_paied_fundings.html'

    def get_queryset(self, *args, **kwargs):
        fundings = super().get_queryset(*args, **kwargs)
        this_participant = self.request.user
        this_participant = Participant.objects.get(user =
this_participant)
        pays = Fundpayment.objects.filter(participant=
this_participant).order_by('-paieddate')
        funds = []
        for p in pays:
            if p.funding not in funds:
                funds.append(p)
        return funds

class ParticipantManagedFundingsView(LoginRequiredMixin,
ListView):
    model = Funding
    template_name =
'fifun/participant_managed_fundings.html'

    def get_queryset(self, *args, **kwargs):
        fundings = super().get_queryset(*args, **kwargs)
        this_participant = self.request.user
        this_participant = Participant.objects.get(user =
this_participant)
        participants =
Fundingparticipant.objects.filter(participant= this_participant)
        funds = []
        for p in participants:
            if p.funding not in funds:
                funds.append(p)
        return funds

```

```

class AuthorManagedFundingsView(LoginRequiredMixin,
ListView):
    model = Funding
    template_name = 'fifun/author_managed_fundings.html'

    def get_queryset(self, *args, **kwargs):
        fundings = super().get_queryset(*args, **kwargs)
        this_participant = self.request.user
        this_participant = Participant.objects.get(user =
this_participant)
        fundings = fundings.filter(author =
this_participant)
        return fundings

class FundingCreateView(LoginRequiredMixin, View):
    template_name = 'fifun/pub_funding.html'

    def get(self, request, pk = None):
        fundform = FundingForm()
        fund_form_context = {'form': fundform }
        return render(request, self.template_name,
fund_form_context)

    def post(self, request, pk):
        fundform = FundingForm (request.POST)
        if not fundform.is_valid():
            fund_form_context = {'form': fundform }
            return render(request, self.template_name,
fund_form_context)
        funding = fundform.save(commit = False)
        f = Forecaster()
        funding.forecast = f.predict(funding)
        funding.save()
        funding.save_m2m()
        return redirect(reverse('fifun:funding', args =
[funding.id]))

class FundingUpdateView (LoginRequiredMixin, View):
    template_name = 'fifun/pub_funding.html'

    def get(self, request, pk):
        fund = get_object_or_404(Funding, id = pk)
        this_participant = self.request.user
        this_participant = Participant.objects.get(user =
this_participant)
        if fund.author != this_participant:

```

```

        return render(request, self.template_name, {})
    fundform = FundingForm(instance = fund)
    fund_form_context = {'form': fundform }
    return render(request, self.template_name,
fund_form_context)

    def post(self, request, pk):
        fund = get_object_or_404(Funding, id = pk)
        fundform = FundingForm (request.POST, instance =
fund)

        this_participant = self.request.user
        this_participant = Participant.objects.get(user =
this_participant)
        if fund.author != this_participant:
            return render(request, self.template_name, {})
        if not fundform.is_valid():
            fund_form_context = {'form': fundform }
            return render(request, self.template_name,
fund_form_context)
        funding = fundform.save(commit = False)
        funding.forecast = Forecaster(funding)
        funding.save()
        funding.save_m2m()
        return redirect(reverse('fifun:funding', args =
[funding.id]))

class FundingSearchView(ListView):
    model = Funding
    template_name = 'fifun/funding_search.html'

    def get_queryset(self, *args, **kwargs):
        funds = super().get_queryset(*args, **kwargs)
        search_text = str(self.request.GET.get('text'))
        area = int(self.request.GET.get('area'))
        part_sum = float(self.request.GET.get('part'))
        part_date = float(self.request.GET.get('date'))
        answered = int(self.request.GET.get('ans'))
        state = int(self.request.GET.get('state'))
        rating_min = int(self.request.GET.get('rating'))
        rating_max = int(self.request.GET.get('maxrating'))
        search_type = int(self.request.GET.get('type'))
        if search_text:
            fundings_caption =
funds.filter(caption__icontains = search_text, state__in = [1,
2, 3, 4, 5])
            cond_1 = Q(idea__icontains = search_text)

```

```

cond_2 = Q(textdescription__icontains =
search_text)
cond_3 = Q(meta__icontains = search_text)
cond_4 = Q(team__icontains = search_text)
cond_5 = Q(sustainability__icontains =
search_text)
fundings_in = Funding.objects.filter(cond_1 |
cond_2 | cond_3 | cond_4 | cond_5)
fundings_caption = fundings_caption.filter(area =
area)
fundings_caption = fundings_caption.filter(state =
state)
fundings_caption =
fundings_caption.filter(rating__gte = rating_min)
fundings_caption =
fundings_caption.filter(rating__lte = rating_max)
fundings_in = fundings_in.filter(area = area)
fundings_in = fundings_in.filter(state = state)
fundings_in = fundings_in.filter(rating__gte =
rating_min)
fundings_in = fundings_in.filter(rating__lte =
rating_max)
fundings = []
for f in fundings_caption:
    now = datetime.datetime.now()
    done = now - f.startdate
    done = done.days
    whole = f.duedate - f.startdate
    whole = whole.days
    res = done / whole
    sum = f.plannedsum
    gs = f.gatheredsum
    res2 = gs / sum
    la = f.lastanswerdate
    lq = f.lastrequestdate
    if answered and lq > la:
        l = la - lq
        if l.days >= 1:
            yes = False
        else:
            yes = True
    elif answered and lq <= la:
        l = la - now
        if l.days >= 1:
            yes = False
        else:

```

```

        yes = True
    else:
        yes = True
        if res < part_date and res2 < part_sum and yes:
            fundings.append(f)
fundings2 = []
captioned = int(self.request.GET.get('captioned'))
for f in fundings_in:
    now = datetime.datetime.now()
    done = now - f.startdate
    done = done.days
    whole = f.duedate - f.startdate
    whole = whole.days
    res = done / whole
    sum = f.plannedsum
    gs = f.gatheredsum
    res2 = gs / sum
    la = f.lastasnwerdate
    lq = f.lastrequestdate
    if answered and lq > la:
        l = la - lq
        if l.days >= 1:
            yes = False
        else:
            yes = True
    elif answered and lq <= la:
        l = la - now
        if l.days >= 1:
            yes = False
        else:
            yes = True
    else:
        yes = True
        if res < part_date and res2 < part_sum and yes
and (captioned == 1):
            fundings2.append(f)
        elif res < part_date and res2 < part_sum and
yes and (captioned == 0):
            fundings.append(f)
if captioned == 1:
    fundings2 = fundings2.order_by('-rating')
    fundings = fundings.order_by('-rating')
    fundings.exted(fundings2)
else:
    fundings = fundings.order_by('-rating')
return fundings

```

```

    def get_context_data(self, **kwargs):
        sp = super(FundingSearchView,
self).get_context_data(**kwargs)
        sp['searched'] = SearchFundingForm (initial={
            'text': self.request.GET.get('text', ''),
            'area': self.request.GET.get('area', '0'),
            'state': self.request.GET.get('state', '0'),
            'part': self.request.GET.get('part', '0'),
            'date': self.request.GET.get('date', '0'),
            'ans': self.request.GET.get('ans', '0'),
            'rating': self.request.GET.get('rating', '0'),
            'maxrating': self.request.GET.get('maxrating',
'5'),
            'captioned': self.request.GET.get('captioned',
'1')
        })
        return sp

```

```

class OverallFundingRatingView(ListView):
    model = Funding
    template_name = 'fifun/funding_rating.html'

    def get_queryset(self, *args, **kwargs):
        fundings = super().get_queryset(*args, **kwargs)
        done = int(self.request.GET.get('done'))
        direction = int(self.request.GET.get('direction'))
        fundings = fundings.objects.filter(state__in = [3,
4])

        direction = int(self.request.GET.get('direction'))
        if direction == 0:
            fundings = fundings.order_by('-rating')
        elif direction == 1:
            fundings = fundings.order_by('-gatheredsum')
        return fundings[0:100]

```

```

class OverallUnFundingRatingView(ListView):
    model = Funding
    template_name = 'fifun/funding_rating.html'

    def get_queryset(self, *args, **kwargs):
        fundings = super().get_queryset(*args, **kwargs)
        done = int(self.request.GET.get('done'))
        direction = int(self.request.GET.get('direction'))
        fundings = fundings.objects.filter(state__in = [1,
2])

        direction = int(self.request.GET.get('direction'))

```

```

        if direction == 0:
            fundings = fundings.order_by('-rating')
        elif direction == 1:
            fundings = fundings.order_by('-gatheredsum')
        elif direction == 2:
            fundings = fundings.order_by
(F('gatheredsum')/F('plannedsum')).asc()
        return fundings[0:100]

class RatedFundingsView(LoginRequiredMixin, ListView):
    model = Funding
    template_name = 'fifun/rated_fundings.html'

    def get_queryset(self, *args, **kwargs):
        this_participant = self.request.user
        this_participant = Participant.objects.get(user =
this_participant)
        ratings = Rating.objects.get(participant =
this_participant).order_by('-creationdate')
        funds = []
        for r in ratings:
            if r.funding not in funds:
                funds.append(r)
        return funds

class LikedFundingsView(LoginRequiredMixin, ListView):
    model = Funding
    template_name = 'fifun/liked_fundings.html'

    def get_queryset(self, *args, **kwargs):
        this_participant = self.request.user
        this_participant = Participant.objects.get(user =
this_participant)
        likes = Like.objects.get(participant =
this_participant, likeordis = True)
        funds = []
        for l in likes:
            if l.funding not in funds:
                funds.append(l)
        return funds

class DislikedFundingsView(LoginRequiredMixin, ListView):
    model = Funding
    template_name = 'fifun/disliked_fundings.html'

```

```

    def get_queryset(self, *args, **kwargs):
        this_participant = self.request.user
        this_participant = Participant.objects.get(user =
this_participant)
        likes = Like.objects.get(participant =
this_participant, likeordis = False)
        funds = []
        for l in likes:
            if l.funding not in funds:
                funds.append(l)
        return funds

class PartlistView(DetailView):
    model = Partlist
    template_name = 'fifun/fundinglist.html'

    def get(self, request, pk):
        part = Partlist.objects.get(id = pk)
        fund = Fundinglist.objects.filter(partlist = part)
        fundings = []
        for f in fund:
            fundings.append(f)
        fundings_context = {}
        fundings_context ['partlist'] = part
        fundings_context ['fundings'] = fundings
        return render(request, self.template_name,
fundings_context)

class ParticipantlevelView(DetailView):
    model = Funding
    template_name = 'fifun/funding_participant_levels.html'

    def get(self, request, pk):
        fund = Funding.objects.get(id = pk)
        fundlevels = Fundinglevel.objects.filter(funding =
fund)

        this_participant = self.request.user
        this_participant = Participant.objects.get(user =
this_participant)
        for f in fundlevels:
            pls =
Participantlevel.objects.filter(participant = this_participant,
fundinglevel = f)
            f.participants = pls
        fundings_context = {}
        fundings_context ['fundlevels'] = fundlevels

```

```

        fundings_context ['fund'] = fund
        return render(request, self.template_name,
fundings_context)

```

```

class RatingCreateView(LoginRequiredMixin, View):
    template_name = 'fifun/funding_rate.html'

    def get(self, request, pk):
        fund = Funding.objects.get(id = pk)
        this_participant = self.request.user
        this_participant = Participant.objects.get(user =
this_participant)
        ratings = Rating.objects.filter(funding = fund,
participant = this_participant)
        payments = Fundpayment.objects.filter(funding =
fund, participant = this_participant)
        if ratings or not payments:
            return render(request, self.template_name, {})
        rateform = RateForm()
        rate_form_context = {'form': rateform }
        return render(request, self.template_name,
rate_form_context)

    def post(self, request, pk):
        this_participant = self.request.user
        this_participant = Participant.objects.get(user =
this_participant)
        rateform = RateForm (request.POST)
        ratings = Rating.objects.filter(funding =
rateform.funding, participant = this_participant)
        payments = Fundpayment.objects.filter(funding =
fund, participant = this_participant)
        if ratings and not payments:
            return HttpResponseRedirect()
        if not rateform.is_valid():
            rate_form_context = {'form': rateform }
            return render(request, self.template_name,
rate_form_context)
        rate = rateform.save(commit = False)
        rate.participant = this_participant
        rate.save()
        return redirect(reverse('fifun:funding', args =
[rate.funding.id]))

```

```

class PartlistCreateView(LoginRequiredMixin, View):
    template_name = 'fifun/partlist_add.html'

```

```

    def get(self, request, pk):
        partform = PartlistForm()
        part_form_context = {'form': reqform }
        return render(request, self.template_name,
part_form_context)

    def post(self, request, pk):
        this_participant = self.request.user
        this_participant = Participant.objects.get(user =
this_participant)
        partform = PartlistForm (request.POST)
        if not partform.is_valid():
            part_form_context = {'form': partform }
            return render(request, self.template_name,
part_form_context)
        part = partform.save(commit = False)
        part.participant = this_participant
        part.save()
        part.save_m2m()
        return redirect(reverse('fifun:partlist', args =
[part.id]))

class PartlistUpdateView (LoginRequiredMixin, View):
    template_name = 'fifun/partlist_add.html'

    def get(self, request, pk):
        part = get_object_or_404(Partlist, id = pk)
        this_participant = self.request.user
        this_participant = Participant.objects.get(user =
this_participant)
        if part.participant!= this_participant:
            return render(request, self.template_name, {})
        partform = PartlistForm (instance = part)
        part_form_context = {'form': partform }
        return render(request, self.template_name,
part_form_context)

    def post(self, request, pk):
        part = get_object_or_404(Partlist, id = pk)
        partform = PartlistForm (request.POST, instance =
fund)

        this_participant = self.request.user
        this_participant = Participant.objects.get(user =
this_participant)
        if part.participant!= this_participant:
            return render(request, self.template_name, {})

```

```

        if not partform.is_valid():
            part_form_context = {'form': partform }
            return render(request, self.template_name,
part_form_context)
        part = partform.save(commit = False)
        part.save()
        part.save_m2m()
        return redirect(reverse('fifun:partlist', args =
[part.id]))

class FundingLikeView(LoginRequiredMixin, View):

    def post(self, request, pk):
        fund = Funding.objects.get(id = pk)
        this_participant = self.request.user
        this_participant = Participant.objects.get(user =
this_participant)
        like = Like.objects.get(funding = fund, participant
= this_participant)
        l = like.likeordis
        if like and l:
            like.delete()
            funding.likes = funding.likes - 1
            funding.save()
        elif like and not l:
            like = Like(funding = fund, participant =
this_participant)
            like.save()
            funding.dislikes = funding.dislikes - 1
            funding.likes = funding.likes + 1
            funding.save()
        else:
            like = Like(funding = fund, participant =
this_participant)
            like.save()
            funding.likes = funding.likes + 1
            funding.save()
        return HttpResponse()

class ListsubscribeView(LoginRequiredMixin, View):

    def post(self, request, pk):
        part = Partlist.objects.get(id = pk)
        this_participant = self.request.user
        this_participant = Participant.objects.get(user =
this_participant)

```

```

        subscribe = Listsubscribe.objects.get(participant =
this_participant, partlist = part)
        if subscribe:
            subscribe.delete()
            part.subs = part.subs - 1
            part.save()
        else:
            subscribe = Listsubscribe(participant =
this_participant, partlist = part)
            subscribe.save()
            part.subs = part.subs + 1
            part.save()
        return HttpResponse()

class FundingDislikeView(LoginRequiredMixin, View):

    def post(self, request, pk):
        fund = Funding.objects.get(id = pk)
        this_participant = self.request.user
        this_participant = Participant.objects.get(user =
this_participant)
        like = Like.objects.get(funding = fund, participant
= this_participant, likeordis = True)
        l = like.likeordis
        if like and not l:
            like.delete()
            funding.dislikes = funding.dislikes - 1
            funding.save()
            this_participant.dislikes =
this_participant.dislikes - 1
            this_participant.save()
        elif like and l:
            like = Like(funding = fund, participant =
this_participant, likeordis = False)
            like.save()
            funding.dislikes = funding.dislikes + 1
            funding.likes = funding.likes - 1
            funding.save()
            this_participant.dislikes =
this_participant.dislikes + 1
            this_participant.likes = this_participant.likes
- 1
            this_participant.save()
        else:
            like = Like(funding = fund, participant =
this_participant, likeordis = False)

```

```

        like.save()
        funding.dislikes = funding.dislikes + 1
        funding.save()
        this_participant.dislikes =
this_participant.dislikes + 1
        this_participant.save()
        return HttpResponse()

class RequestCreateView(LoginRequiredMixin, View):
    template_name = 'fifun/request_add.html'

    def get(self, request, pk):
        reqform = RequestForm()
        req_form_context = {'form': reqform }
        return render(request, self.template_name,
req_form_context)

    def post(self, request, pk):
        this_participant = self.request.user
        this_participant = Participant.objects.get(user =
this_participant)
        reqform = RequestForm (request.POST)
        if not reqform.is_valid():
            req_form_context = {'form': reqform }
            return render(request, self.template_name,
req_form_context)
        req = reqform.save(commit = False)
        req.participant = this_participant
        req.save()
        return redirect(reverse('fifun:funding', args =
[req.funding.id]))

class AnswerCreateView (LoginRequiredMixin, View):
    template_name = 'fifun/request_add.html'

    def get(self, request, pk):
        req = get_object_or_404(Participantrequest, id =
pk)

        this_participant = self.request.user
        this_participant = Participant.objects.get(user =
this_participant)
        if fund.author != this_participant:
            return render(request, self.template_name, {})
        reqform = RequestForm (instance = req)
        req_form_context = {'form': reqform }

```

```

        return render(request, self.template_name,
req_form_context)

    def post(self, request, pk):
        req = get_object_or_404(Participantrequest, id =
pk)

        reqform = RequestForm (request.POST, instance =
req)

        this_participant = self.request.user
        this_participant = Participant.objects.get(user =
this_participant)
        parts = Fundingparticipant.objects.filter(funding =
req.funding)

        supporter = False
        for p in parts:
            if p.participant == this_participant:
                supporter = True
        if (not supporter) and req.funding.author !=
this_participant:
            return render(request, self.template_name, {})
        if not reqform.is_valid():
            req_form_context = {'form': reqform }
            return render(request, self.template_name,
req_form_context)
        req = reqform.save(commit = False)
        req.save()
        return redirect(reverse('fifun:funding', args =
[funding.id]))

class RecommendationView(LoginRequiredMixin, View):

    def post(self, request, pk):
        fund = Funding.objects.get(id = pk)
        nickname = request.data.get("user")
        address_participant =
Participant.objects.get(nickname = nickname)
        this_participant = self.request.user
        this_participant = Participant.objects.get(user =
this_participant)
        recom = Recommendation.objects.get(funding = fund,
receiver = address_participant, author = this_participant)
        recom.save()
        return HttpResponse()

class RecommendationsView(LoginRequiredMixin, ListView):
    model = Recommendation

```

```

template_name = 'fifun/recom_fundings.html'

def get_queryset(self, *args, **kwargs):
    recoms = super().get_queryset(*args, **kwargs)
    this_participant = self.request.user
    this_participant = Participant.objects.get(user =
this_participant)
    recoms = recoms.filter(receiver =
this_participant).order_by('-creationdate')
    return recoms

class CreatedRecommendationsView(LoginRequiredMixin,
ListView):
    model = Recommendation
    template_name = 'fifun/created_recom_fundings.html'

    def get_queryset(self, *args, **kwargs):
        recoms = super().get_queryset(*args, **kwargs)
        this_participant = self.request.user
        this_participant = Participant.objects.get(user =
this_participant)
        recoms = recoms.filter(author =
this_participant).order_by('-creationdate')
        return recoms

```

ДОДАТОК В
Слайди презентації

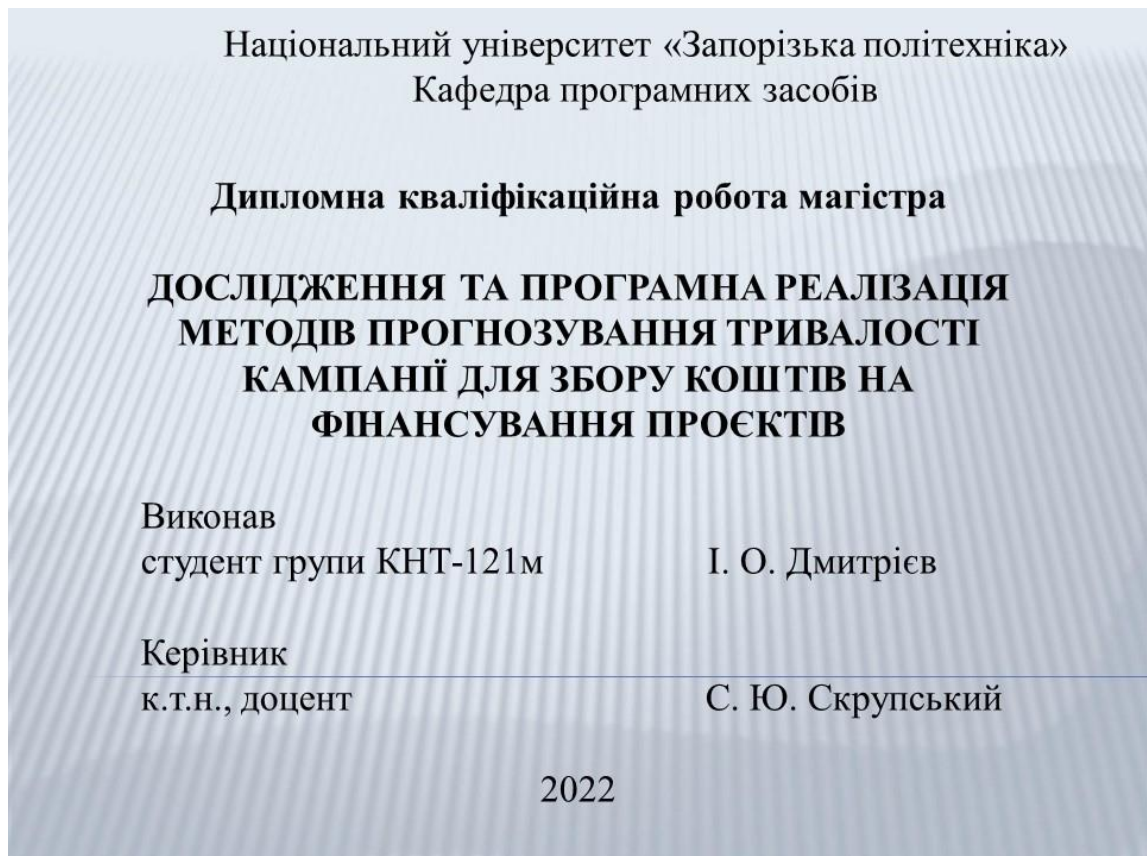


Рисунок В.1 – Титульний слайд

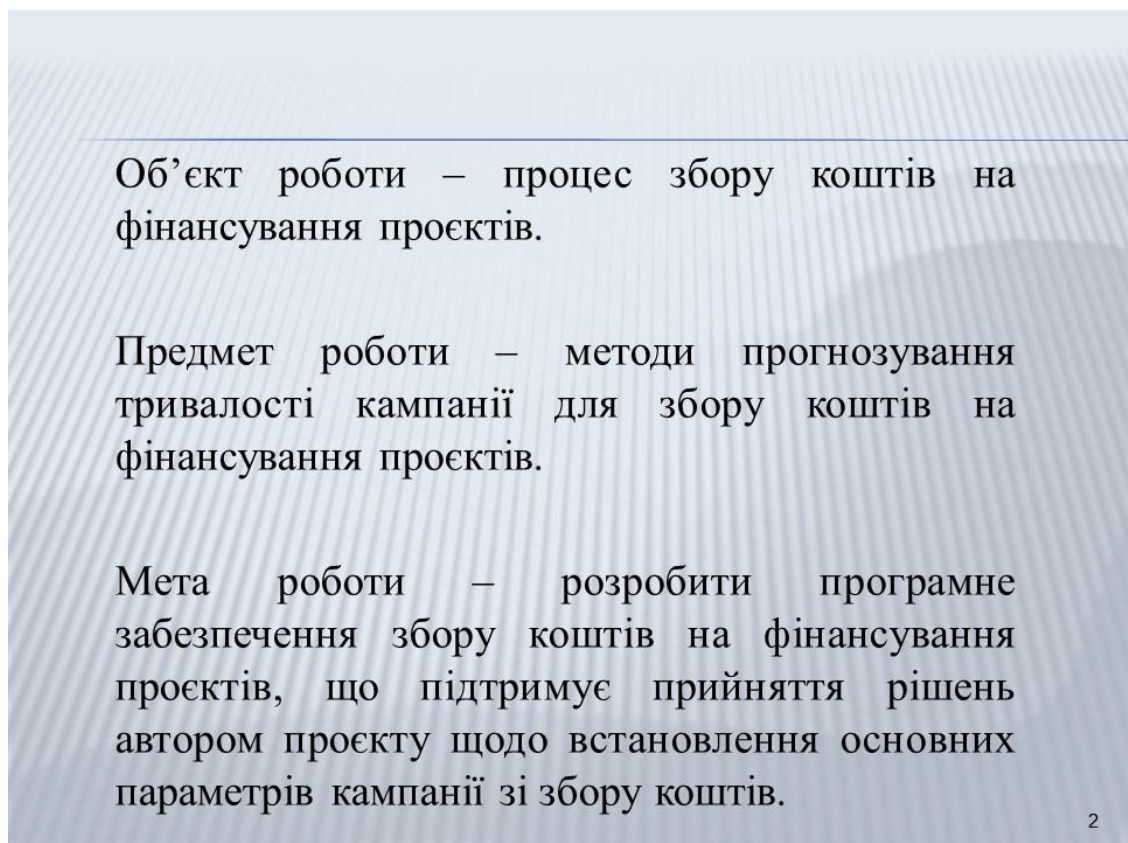


Рисунок В.2 – Об'єкт, предмет, мета роботи

Завдання

- ✓ виконати дослідження методів прогнозування тривалості кампанії для збору коштів на фінансування проєктів;
- ✓ спроектувати програмне забезпечення збору коштів на фінансування проєктів;
- ✓ реалізувати програмне забезпечення.

3

Рисунок В.3 – Завдання роботи

Порівняння програмних засобів

Критерій	FundRazr	Patreon	Kickstarter	Власна програма
Модель збору коштів	Все або нічого-го, вся сума	Сума підписок	Все або нічого	Все або нічого, мінімальна сума
Спосіб фінансуван-ня проєктів	Окремі платежі	Підписка	Окремі платежі	Окремі платежі
Безкоштовна підписка для власників проєктів	+	-	-	+
Прогнозування тривалості кампанії	-	-	-	+

4

Рисунок В.4 – Порівняння програмних засобів

Порівняння алгоритмів на основі дерев ухвалення рішень

Критерій	ID3	C4.5	CART
Задача, для розв'язання якої може використовуватися	Класифікація	Класифікація	Класифікація, регресія
Здатність працювати з пропущеними значеннями	Відсутня	Відсутня	Присутня
Швидкість	Низька	Вища	Середня

5

Рисунок В.5 – Порівняння алгоритмів на основі дерев ухвалення рішень

Метод прогнозування тривалості кампанії для збору коштів на фінансування проєктів

На початку відбувається створення вибірки даних, яка буде використовуватися в подальшому для навчання моделі.

На другому етапі має бути виконано перетворення текстових даних у векторизовану числову форму, використовуючи метод TF-IDF.

На третьому етапі відбувається навчання за цією вибіркою у формі, отриманій за результатами другого етапу, дерева ухвалення рішення на основі алгоритму CART.

На четвертому етапі відбувається збереження моделі, навчання якої було виконано на третьому етапі, в файл або завантаження її, якщо даний етап виконується окремо для прогнозування. Окремо даний етап може виконуватися, наприклад, під час роботи програми, в якій використовується навчена і раніше збережена модель.

На п'ятому етапі виконується витягання з бази даних або на основі заданих параметрів даних, які характеризують кампанію зі збору коштів, яка ще не була запущена і для якої невідома фактична тривалість. Ці дані відповідають ознакам вибірки, виключаючи вихідну ознаку.

На шостому етапі на основі сформованих даних навченою моделлю відбувається прогнозування тривалості виконання збору коштів для заданої на п'ятому етапі кампанії.

6

Рисунок В.6 – Метод прогнозування тривалості кампанії для збору коштів на фінансування проєктів

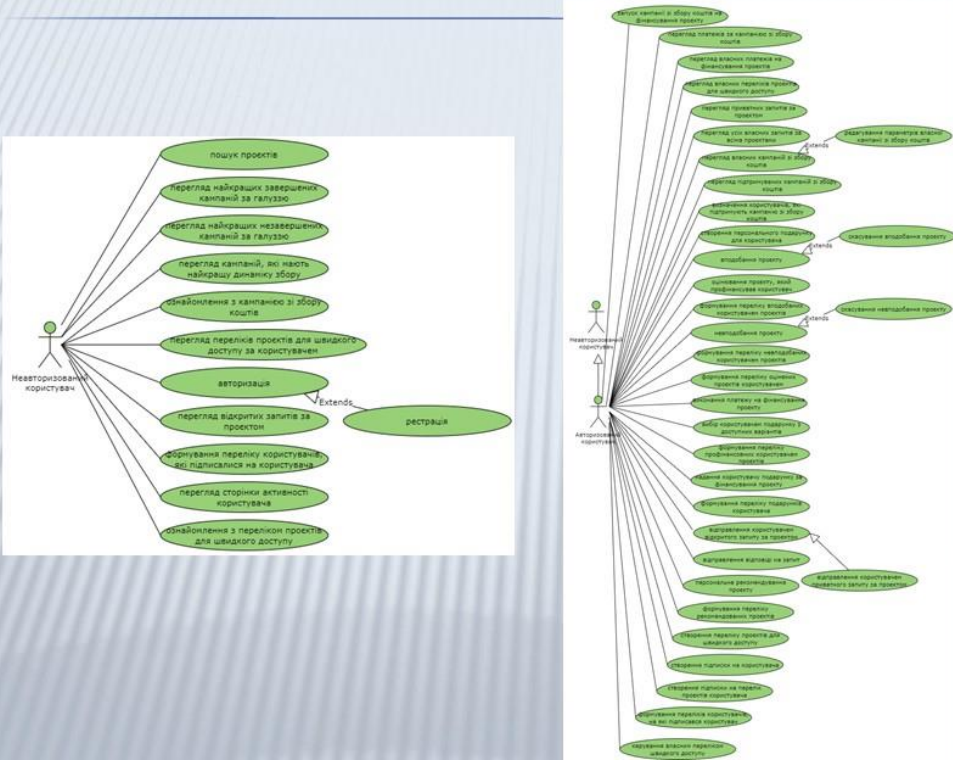
Порівняння фреймворків для розробки вебдодатку

Критерій	Django	TurboGears
Підтримка об'єктно-реляційного відображення	Вбудована, Django ORM	Вбудована, SQLAlchemy
Рушій шаблонів	Django	Genshi
Адміністрування	Вбудоване	Вбудоване
Безпека	Потужна проти великої кількості атак	Максимально обмежена

7

Рисунок В.7 – Порівняння фреймворків для розробки вебдодатку

Діаграми прецедентів



8

Рисунок В.8 – Діаграми прецедентів

Схема структури бази даних

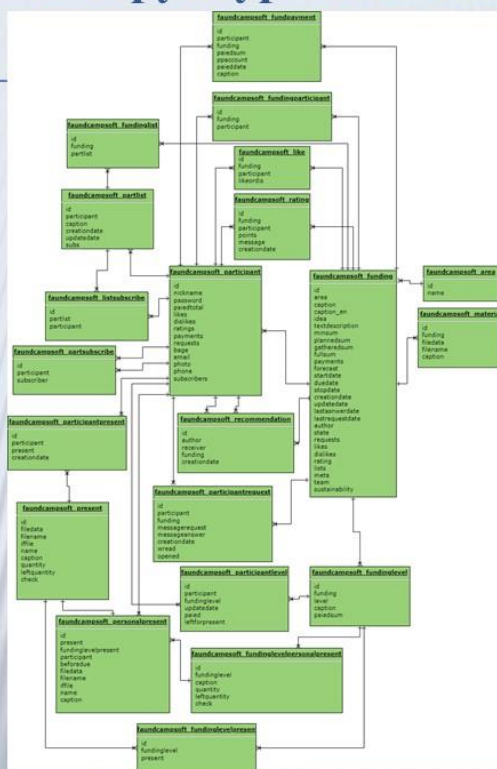


Рисунок В.9 – Схема структури бази даних

Діаграма станів кампанії зі збору коштів для фінансування проєкту

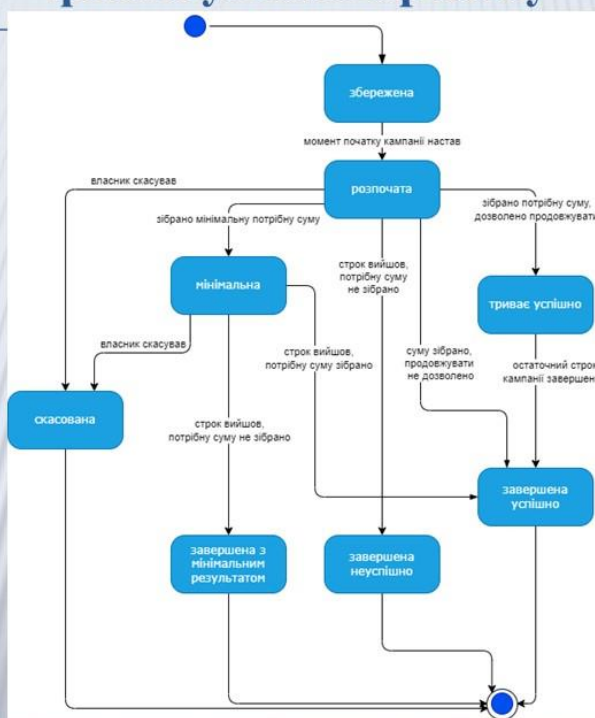


Рисунок В.10 – Діаграма станів кампанії зі збору коштів для фінансування проекту

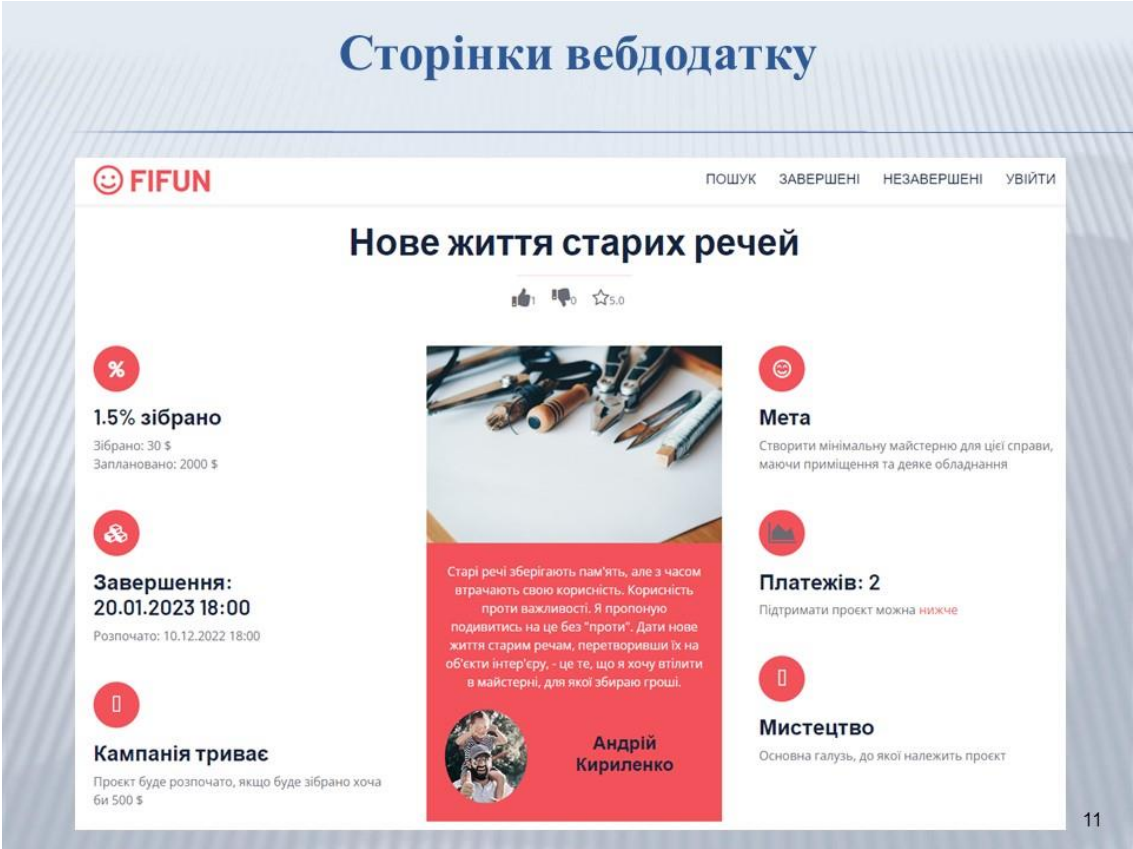


Рисунок В.11 – Сторінки вебдодатку



Рисунок В.12 – Сторінки вебдодатку

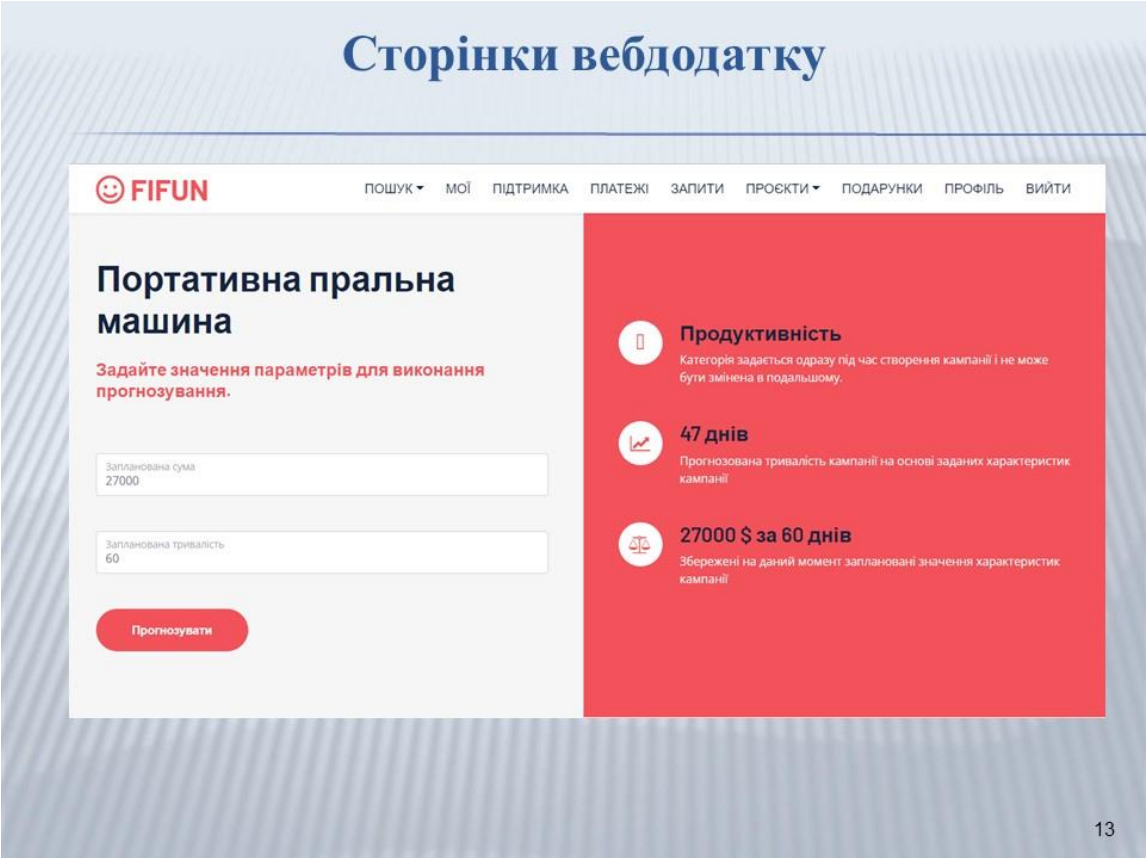


Рисунок В.13 – Сторінки вебдодатку

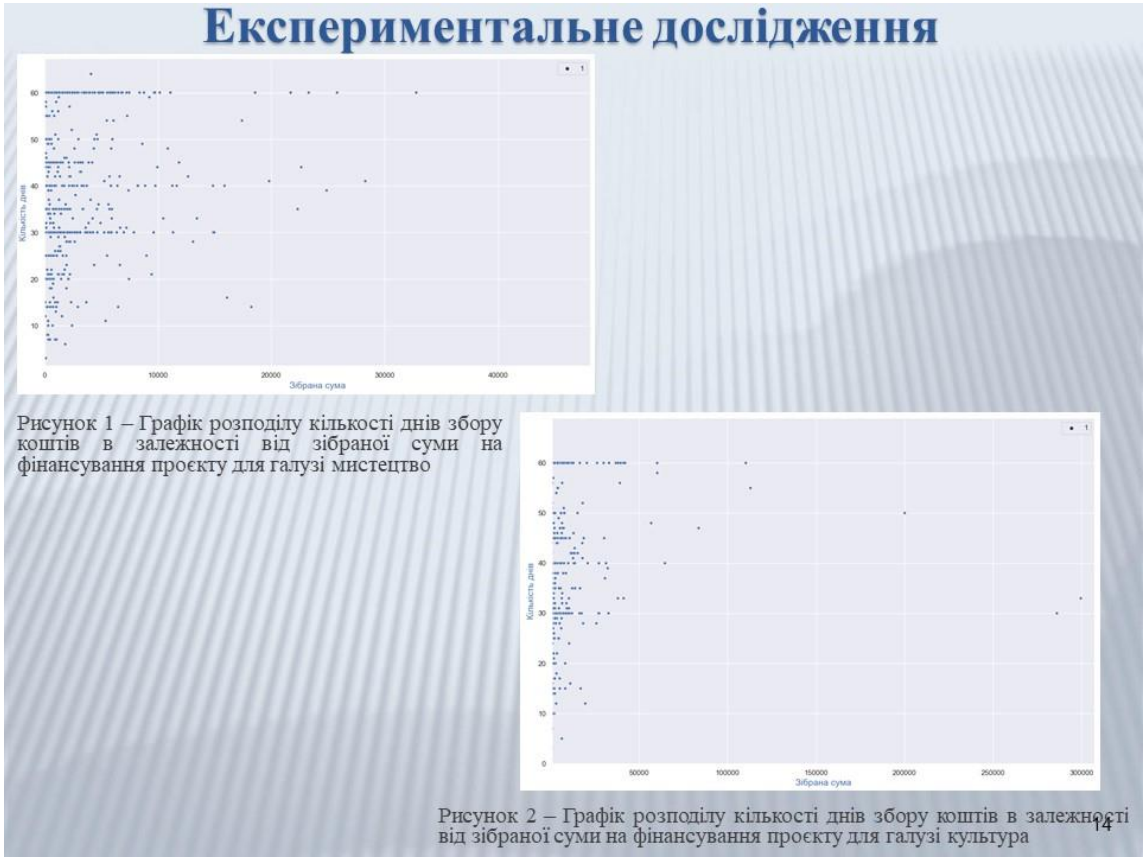


Рисунок В.14 – Експериментальне дослідження

Експериментальне дослідження

Метод/ модель	Точність
Лінійна регресійна модель	0.553
Метод прогнозування тривалості кампанії для збору коштів на фінансування проєктів	0.801

15

Рисунок В.15 – Експериментальне дослідження

Висновки

Представлене програмне забезпечення дозволяє організовувати кампанії зі збору коштів на фінансування проєктів, керувати цими кампаніями, обмежувати доступ до матеріалів проєкту користувачам в залежності від того, якою сумою вони профінансували проєкт, а користувачам у свою чергу шукати проєкти, брати участь у фінансуванні проєктів з отриманням різного роду подарунків, матеріалів, створювати переліки швидкого доступу, рекомендувати проєкти іншим користувачам.

Наукова новизна роботи сформована в методі прогнозування тривалості кампанії для збору коштів на фінансування проєктів, який виконує визначення строку, необхідного на збір певної суми коштів на фінансування конкретного проєкту, до початку кампанії, використовуючи дерева ухвалення рішень на основі алгоритму CART, що дозволяє збалансувати встановлені суму коштів кампанії та тривалість такої кампанії, щоб в підсумку вона з більшою вірогідністю завершилась успішно.

16

Рисунок В.16 – Висновки