Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

РОЗРОБКА ВЕБ-СИСТЕМИ КРАУДФАНДІНГОВОЇ ПЛАТФОРМИ З ІНТЕГРАЦІЄЮ ШТУЧНОГО ІНТЕЛЕКТУ

Виконав: студент II курсу
групи ___1ПІ-22М_____ спеціальності
121 – Інженерія програмного забезпечення
Ярошевич Максим Сергійович

Керівник: к.т.н. доцент, Коваленко О. О.
«06» 12 2023 р.

Опонент: к.т.н., проф. Кондратенко. Н. Р.
«06» 12 2023 р.

Допущено до захисту

Завідувач кафедри ПЗ
д.т.н., проф. Романюк О. Н.
(прізвище та ініціали)
«11» 12 2023 р.

ВНТУ – 2023

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Рівень вищої освіти II-й (магістерський)
Галузь знань 12 – Інформаційні технології
Спеціальність 121 – Інженерія програмного забезпечення
Освітньо-професійна програма – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
Романюк О. Н.
«_19_» _вересня_ 2023 р.

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Ярошевичу Максиму Сергійовичу

1. Тема роботи – розробка веб-системи краудфандінгової платформи з інтеграцією штучного інтелекту.

Керівник роботи: Коваленко Олена Олексіївна, к.т.н., затверджені наказом вищого навчального закладу від « 18 » вересня 2023 р. № 247.

2. Строк подання студентом роботи
_____5 грудня 2023 р._____

3. Вихідні дані до роботи:

Потужність множини користувачів - не менше 3; мінімальна кількість записів в базі даних - не менше 15, база даних - реляційна, мова програмування - веб-орієнтовна.

4. Зміст текстової частини: Аналіз поняття кравдфаднінгу, існуючих кравдфандінгових систем їх переваг та недоліків. Розробка методу створення кравдфандінгової платформи з інтеграцією штучного інтелекту, розробка алгоритмів, проектування архітектури. Програмна реалізація веб-системи кравдфандінгової платформи. Тестування розробленого програмного забезпечення.

5. Перелік ілюстративного матеріалу: моделі автентифікації у системі; авторизації у системі; роботи попередньої обробки данних та визначення тональності механізмом NLP; роботи та схем архітектури модулів клієнтської та серверної частини; UML-діаграми класів Entity системи; презентація результатів МКР, титульний слайд, аналіз актуальності, мета та завдання, об'єкт, предмет та методи, наукова новизна, існуючі системи, недоліки, цілі

досліження, сособи оцінювання, удосконалений метод, розробка алгорит
проектування архітектури, вибір технологій, тест-кейси, висновки, публікації

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	Коваленко О. О., к.т.н., доцент каф. ПЗ	19.09.23	05.12.23
5	Причепа І. В. к.е.н., доцент каф. ЕПВМ	16.11.23	21.11.23

7. Дата видачі завдання 19 вересня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної Роботи	Строк виконання етапів роботи	Примітк
1	Аналіз поняття кравдфандінгу, та кравдфандінгових систем	20.09.2023 - 27.09.2023	Вик.
2	Розробка методу створення кравдфандінгової платформи, проектування архітектури системи	28.09.2023 - 10.10.2023	Вик.
3	Програмна реалізація системи за допомогою Angular, Nest.js, Natural	11.10.2023 - 27.10.2023	Вик.
4	Розробка тест-кейсів та тестування програмного забезпечення	28.10.2023 - 15.11.2023	Вик.
5	Виконання економічної частини	16.11.2023 - 21.11.2023	Вик.
6	Оформлення пояснювальної записки і графічного матеріалу	22.11.2023 - 27.11.2023	Вик.
7	Підготовка до захисту	28.11.2023 - 05.12.2023	Вик

Студент _____ **Ярошевич М. С.**
(підпис) (прізвище та ініціали)

Керівник магістерської кваліфікаційної роботи _____ **Коваленко О. О.**
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Ярошевич М.С. Розробка веб-системи краудфандінгової платформи з інтеграцією штучного інтелекту: магістерська кваліфікаційна робота зі спеціальності 121 – інженерія програмного забезпечення, освітня програма – інженерія програмного забезпечення. Вінниця: ВНТУ, 2023. 102 с.

На укр. мові. Бібліогр. : 36 назв ; рис. : 35; табл. 17.

У магістерській кваліфікаційній роботі розроблено веб-систему краудфандінгової платформи в якій розширюється функціонал за рахунок впровадження методу створення краудфандінгової платформи з інтеграцією штучного інтелекту. Практична реалізація дозволяє автоматизовано надавати оцінки краудфандінговим проектам, використовуючи системи обробки природньої мови. Реалізація виконана у вигляді клієнт-серверного додатку з використанням фреймворків Angular та Nest.js. Також під час розробки велику увагу було приділено архітектурі системи, аби забезпечити її масштабованість у майбутньому. В результаті розроблене програмне забезпечення було протестоване на відповідність поставленим вимогам та констатовано досягнення мети роботи.

Отримані в магістерській кваліфікаційній роботі результати можна використати для побудови аналогічних систем в яких можливо автоматизувати процеси за допомогою систем обробки природної мови.

Ключові слова: веб-система, кравдфандінг, кравдфандінгова платформа, кравдфандінгова кампанія, обробка природної мови, штучний інтелект.

ANNOTATION

Yarosevich M.S. Development of a web system of a crowdfunding platform with the integration of artificial intelligence: master's qualification work on the specialty 121 - software engineering, educational program - software engineering.

In Ukrainian language. Bibliographer: 36 titles ; fig. : 35; tabl. 17.

In the master's qualification work, a web system of a crowdfunding platform was developed, in which the functionality is expanded due to the implementation of the method of creating a crowdfunding platform with the integration of artificial intelligence. A practical implementation allows for automated evaluation of crowdfunding projects using natural language processing systems. The implementation is made in the form of a client-server application using Angular and Nest.js frameworks. Also, during the development, a lot of attention was paid to the architecture of the system to ensure its scalability in the future. As a result, the developed software was tested for compliance with the set requirements and it was confirmed that the goal of the work had been achieved.

The results obtained in the master's qualification work can be used to build similar systems in which it is possible to automate processes using natural language processing systems.

Keywords: web system, crowdfunding, crowdfunding platform, crowdfunding campaign, natural language processing, artificial intelligence.

ЗМІСТ

ВСТУП

Обґрунтування вибору теми дослідження. У сучасному глобальному суспільстві із загальним поширенням цифрових технологій та зростанням впливу інтернету на всі сфери життя, трансформація традиційних методів фінансування стає необхідністю. Крім того в наш доволі нелегкий час війни, коли потреби армії несумнівно великі, широкого масштабу набула волонтерська діяльність. Майже кожного дня створюються сотні зборів різними фондами або просто окремими волонтерами для збору коштів на ті чи інші засоби які потребують військові. Концепція краудфандінгу вже давно перетворилася з експерименту в ефективний та динамічний інструмент для залучення коштів на проекти з різних галузей, не тільки військових. Від благодійних ініціатив та до стартапів та інноваційних досліджень, краудфандінг здатний перетворити ідеї в реальність шляхом залучення широкої аудиторії спонсорів та інвесторів [1-3].

Проте, разом із зростанням популярності краудфандінгу виникають нові виклики та можливості особливо в контексті військових зборів. Різноманітні потреби користувачів та безупинне зростання кількості створюваних зборів вимагають постійної адаптації та удосконалення краудфандінгових платформ. Розробка ефективних методів та засобів для створення динамічних краудфандінгових платформ стає актуальною задачею для забезпечення успіху проектів та задоволення потреб спонсорів. Саме тому, у даному дослідженні, спрямовано свою увагу на розгляд сучасного стану краудфандінгу, аналізуючи його переваги та обмеження. Аналіз тенденцій в розвитку цього інструментарію, виявлення важливих аспектів, недоліків та можливостей для їх оптимізації; визначення найкращих підходів для створення та управління краудфандінговими платформами дозволяють підвищити рівень ефективності використання краудфандінгових платформ [4-5]. Запровадження нових технологій також допоможе вирішити важливі завдання з покращення ефективності краудфандінгових платформ та забезпечити їх адаптацію до змін за вимогами користувачів.

Для розширення функціоналу, підвищення ефективності використання каудфандінгової платформи необхідно розглянути основні виклики та можливості у сфері краудфандінгу, дослідити відомі підходи та важливі аспекти розробки динамічних краудфандінгових платформ, розробити удосконалені методи та моделі створення та використання платформи та запропонувати практичні підходи для програмної реалізації.

Зв'язок роботи з науковими програмами, планами, темами. Робота виконувалася згідно плану виконання наукових досліджень на кафедрі програмного забезпечення.

Мета та завдання дослідження. Мета дослідження полягає в створенні та в запровадженні методу розробки автоматизованої краудфандінгової платформи, що дозволить запровадити автоматизоване оцінювання рівня привабливості краудфандінгових проектів за рахунок використання алгоритмів обробки природної мови.

Завдання дослідження:

Для того щоб досягнути поставленої мети, необхідно сформувати та вирішити наступні задачі:

1. Проаналізувати поняття краудфандінгу та краудфандінгових систем.
2. Провести аналіз існуючих методів та інструментарію краудфандінгових платформ та їх дизайну.
3. Запропонувати методи автоматизації процедур оцінювання краудфандінгових проектів.
4. Розробити високорівневу структуру модулів веб системи краудфандінгової платформи.
5. Розробити модуль інтеграції з механізмом штучного інтелекту.
6. Провести тестування розробленого модуля який інтегрується з механізмом штучного інтелекту.

Об'єктом дослідження є процеси розширення функціоналу краудфандінгових платформ шляхом інтеграції з механізмом штучного інтелекту.

Предметом дослідження є методи та моделі створення та використання програмних засобів краудфандінгових платформ.

Методи дослідження. Під час розробки системи використовувались наступні методи, а саме аналізу – для аналізу існуючих рішень функціонування краудфандінгових платформ; архітектури веб-продуктів; синтезу – для формування методу моделювання та проектування архітектури; моделювання та візуалізації, теорії множин, теорії алгоритмів – для формування моделей, алгоритмів роботи платформи.

Наукова новизна отриманих результатів.

1. Удосконалено метод створення краудфандінгової платформи за рахунок використання NLP-технологій, моделі прийняття рішення щодо інвестування проєкту, алгоритмів формування клієнтської та серверної частини, які, на відміну, від існуючих, дозволяють здійснювати аналіз текстової інформації краудфандінгових проєктів, на основі результатів якого виконується автоматизоване оцінювання, сортування релевантних компаній, що дозволяє розширити функціонал платформи, більш детально та достовірно підготувати дані для прийняття рішення щодо обґрунтованого вибору проєкту для інвестування.

2. Набула подальшого розвитку модель прийняття рішення на основі результатів аналізу текстових даних щодо фінансування проєктів, яка, на відміну від існуючих, дозволяє здійснювати аналіз текстової інформації краудфандінгових проєктів за рахунок використання NLP-технології та враховувати експертну оцінку користувача (або/і запрошених експертів), що дозволяє розширити функціонал платформи, врахувати результати автоматизованого та експертного оцінювання, більш детально та достовірно підготувати дані для прийняття рішення щодо вибору проєкту користувачем.

Практична цінність отриманих результатів. Практична цінність полягає в тому що програмний продукт може бути використаний власниками краудфандінгової платформи для автоматизованого моніторингу та оцінювання

проєктів, формування рейтингового списку для прийняття рішення користувачем.

Особистий внесок здобувача. Усі наукові результати, викладені у магістерській кваліфікаційній роботі, отримані автором особисто. У друкованих працях, опублікованих у співавторстві, автору належать такі результати: архітектури краудфандінгової системи з використанням nlp для аналізу текстових даних [1]; метод оцінювання та ранжування проєктів за допомогою інтелектуальних алгоритмів [2].

Апробація результатів виконана на міжнародних науково-технічних конференціях:

"Сучасні тенденції розвитку техніки та технологій". Міжнародний центр технологічних інновацій (Харків, 31 жовтня 2023 р); "Електронні інформаційні ресурси: створення, використання, доступ 2023". 20-21 листопада 2023 р. Суми-Вінниця- Люблін.

Публікації. Основні результати досліджень опубліковано в 2 наукових працях.

Ярошевич М.С., Коваленко О. О. Розробка архітектури краудфандінгової системи з використанням nlp для аналізу текстових даних. Матер. Міжнародної науково-технічної конференції “Сучасні тенденції розвитку техніки та технологій”.Міжнародний центр технологічних інновацій (Харків, 31 жовтня 2023 р). Research Europe, 2023. 13 с.[6].

Ярошевич М.С., Коваленко О. О. Удосконалення методів оцінювання та ранжування проєктів за допомогою інтелектуальних алгоритмів Матер. Міжнародної науково-практичної Інтернет конференції “Електронні інформаційні ресурси: створення, використання, доступ” 20-21 листопада 2023 р. – Суми/Вінниця: НІКО/ КЗВО «Вінницька академія безперервної освіти». 2023. – 323 с. [7].

1 ТЕОРЕТИКО-ПРАКТИЧНІ ЗАСАДИ СТВОРЕННЯ ТА ВИКОРИСТАННЯ КРАУДФАНДІНГОВИХ ПЛАТФОРМ

1.1 Поняття краудфандінгу та краудфандінгових систем

В наш час пошук та залучення капіталу є ключем до зростання будь-якої компанії. Найбільшим та авторитетним корпораціям може бути легко отримати гроші від інвесторів або взяти на себе додатковий борг від своїх кредиторів. Однак деякі підприємства можуть зіткнутися з перешкодами, які можуть стримати їхнє зростання. Особливо це актуально та відчутно для невеликих компаній та стартапів. Крім того необов'язково бути підприємцем з власною бізнес ідеєю, для пошуку коштів, існує велика кількість прикладів того коли люди не в змозі зібрати необхідну суму коштів на лікування тощо. У першому та другому сценарію є єдина проблематика, а саме залучення фінансових ресурсів для реалізації проектів або допомоги тим, хто цього потребує.

Як результат концепція краудфандінгу в таких випадках може виявитися доволі корисною. Оскільки краудфандінг дає можливість людям які цього потребують залучити фінанси від будь-кого, хто має гроші для інвестування. Він надає форум для всіх, хто має ідею представити його очікуваним інвесторам.

Краудфандинг – це спосіб залучити кошти для росту і розвитку проекту, ініціативи, підприємства або програми за рахунок внесків від великої кількості сторонніх осіб, які можуть бути не пов’язані ані з самим проєктом, ані з професійним бізнес-інвестуванням. Для засновника це безризикові кошти, оскільки залучаються вони на доброчинній основі від небайдужих до можливого проєкту людей [1].

У сучасному світі кошти збираються переважно на онлайн-платформах, а за внесок доброчинці отримують винагороду – якісь пам’ятні речі, або сертифікат на майбутній продукт, якщо його вдасться виготовити [1]. Такі краудфандінгові платформи дозволяють організаторам проектів створювати профілі проектів, описувати їхні цілі, встановлювати обрані цілі фінансування

та надавати нагороди для спонсорів. В свою чергу для спонсорів такі платформи забезпечують можливість внесків через онлайн-платежі, можливості взаємодії з творцями проектів у вигляді коментарів, питань та обговорень. Крім того, ці системи надають інструменти аналітики для відстеження прогресу та ефективності кампаній краудфандінгу.

В залежності від моделі краудфандінгу платформи можуть базуватись на якомусь конкретному його виді, або ж поєднувати в собі декілька. Поділення краудфандінга приведено нижче на рисунку 1.1.

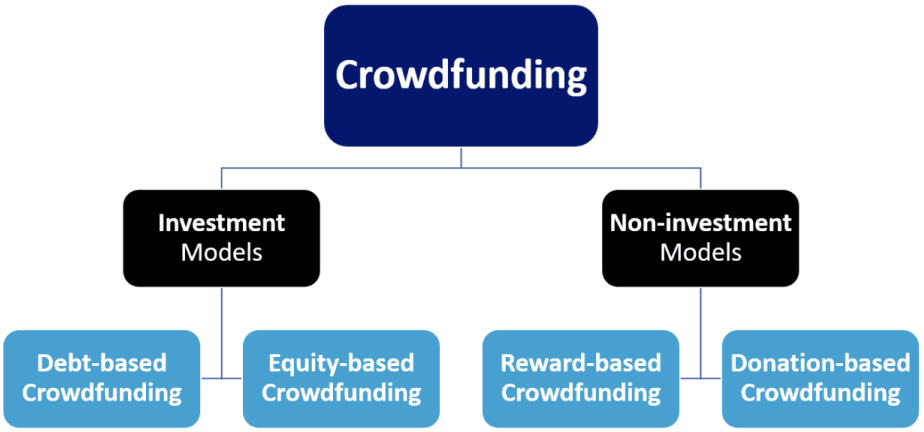


Рисунок 1.1 - Види краудфандінгу

З рисунку можемо побачити що в цілому краудфандінг поділяється на 2 основні моделі, інвестоорієнтовну в якій спонсори очікують повернення вкладених коштів у вигляді капіталу, акцій тощо та неінвестоорієтовну, яка передбачає внесення коштів з отриманням нефінансових нагород, або ж взагалі без якої-небудь віддачі.

1.2 Аналіз функціоналу та дизайну існуючих платформ карудфандінгу

В наш час найбільш популярними краудфандинговими платформами є GoFundMe, Kickstarter, Patreon, Indiegogo и StartEngine. Всі вони залучають мільйони людей, які прагнуть розвивати чи підтримувати нові ідеї, майбутні

ділові чи шляхетні справи. Всі разом вони вже зібрали понад 25 мільярдів доларів на соціальні ініціативи, заходи, художників, інноваційні продукти та підприємницькі ідеї. Розглянемо функціонал кожної з них більш детально та виділимо переваги та недоліки.

Першою платформою яку розглянемо є платформа GoFundMe, яка є однією з найпопулярніших та найбільш успішних платформ краудфандінгу в світі. Вона була заснована у 2010 році та має глибокий соціальний вплив, дозволяючи тисячам людей залучати кошти для вирішення найрізноманітніших завдань та проблем. На рисунку 1.2 зображено головну сторінку платформи.

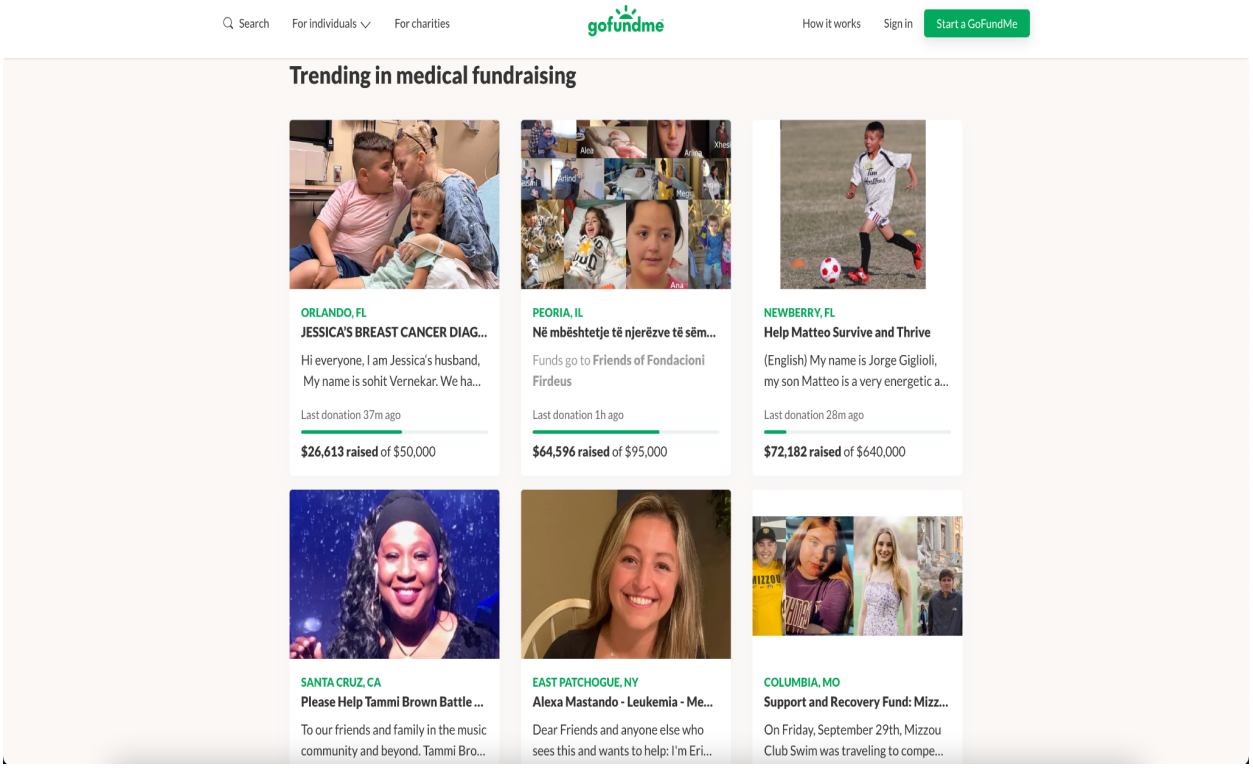


Рисунок 1.2 - Список проектів на платформі GoFundMe

На рис. 1.3 представлено сторінку одного з краудфандінгових проєктів на зазначеній платформі GoFundMe.

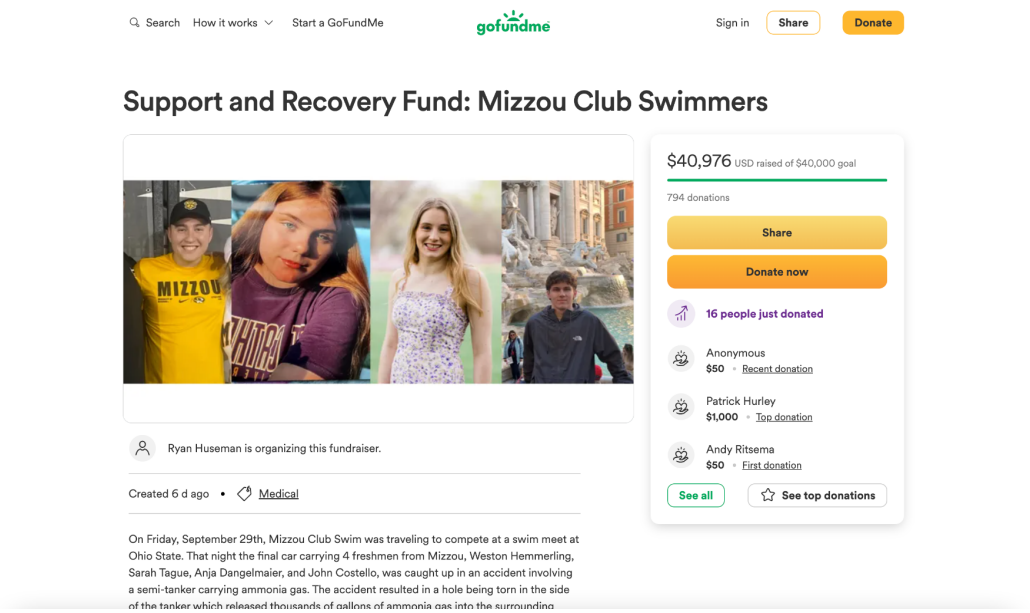


Рисунок 1.3 - Обраний проект на платформі GoFundMe

Розглянемо переваги та недоліки платорми GoFundMe як наведені у таблиці 1.1.

Таблиця 1.1 – Переваги та недоліки платформи GoFundMe

Переваги	Недоліки
Зручний інтерфейс	Проблеми з швидкодією
Різноманітність напрямів	Відсутність механізмів оцінки проектів
Адаптивність	Безпека та зловживання

Наступною платформою є Kickstarter, яка також є однією з найбільш популярних та впливових краудфандінгових платформ у світі, але на відміну від попередньої спеціалізована на вужчому сегменті проектів. Так як, основна мета платформи - дати можливість творчим та креативним особам залучити фінансування для реалізації своїх ідей та проектів, Kickstarter охоплює широкий спектр категорій, включаючи мистецтво, дизайн, технології, гри, моду, кіно, музику, гастрономію, комікси та багато інших. Основний екран зображений на рисунку 1.4.

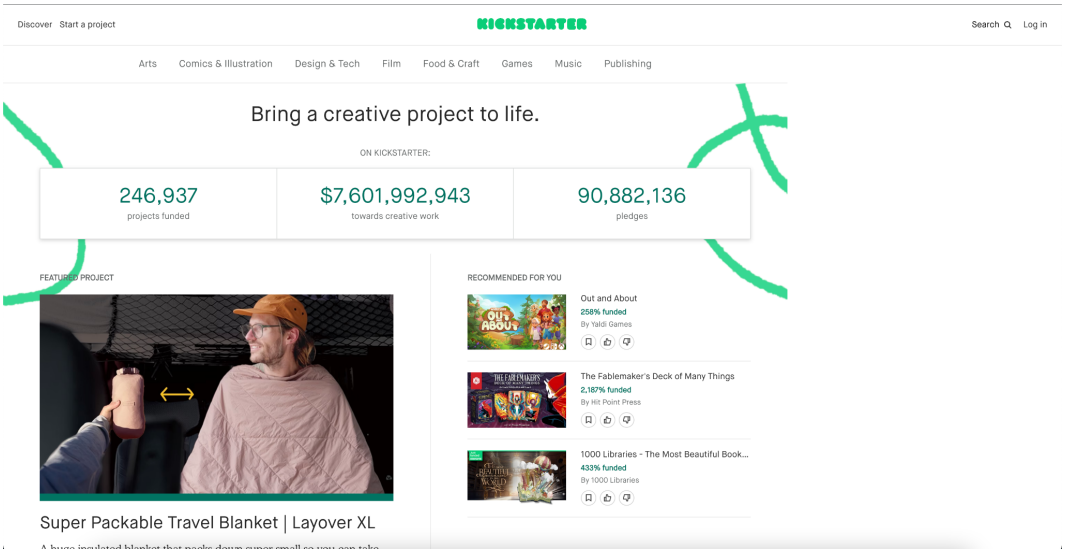


Рисунок 1.4 – Список проектів на платформі Kickstarter

На рис. 1.5 представлено обраний краудфандінговий проєкт на платформі Kickstarter.

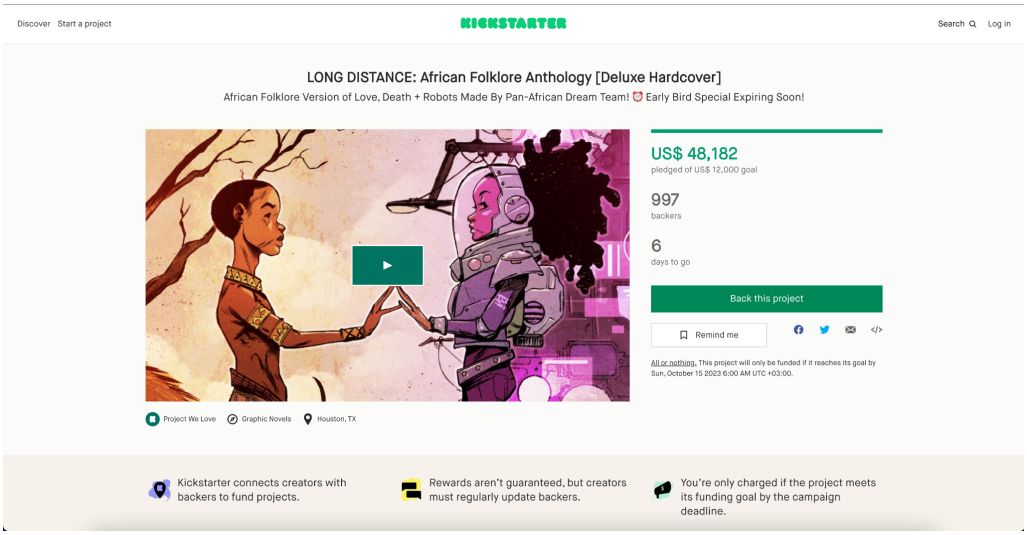


Рисунок 1.5 – Обраний проект на платформі Kickstarter

Розглянемо переваги та недоліки платорми Kickstarter як наведені у таблиці 1.2.

Таблиця 1.2 – Переваги та недоліки платформи Kickstarter

Переваги	Недоліки
Зручний інтерфейс	Складність у проходженні контролю та схвалення проектів перед їхнім запуском
Інтегрованість з системами платежів	Відсутність механізмів оцінки проектів
Адаптивність	-

Останньою з найбільш популярних платформ які ми розглянемо є StartEngine яка є платформою, що сприяє зростанню стартапів та компаній через еквіті краудфандінг. Вона надає можливість інвестувати в перспективні проекти та сприяє розвитку інновацій та технологічного прогресу. Основні екрани зображені на рисунках 1.6.

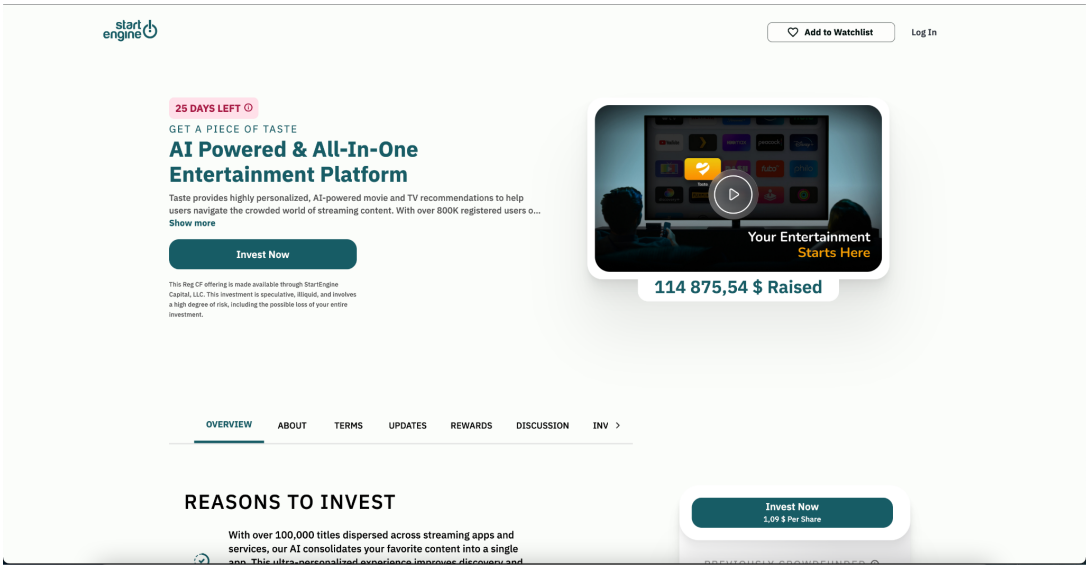


Рисунок 1.6 - Обраний проект на платформі StartEngine

Розглянемо переваги та недоліки платорми StartEngine як наведені у таблиці 1.3.

Таблиця 1.3 – Переваги та недоліки платформи StartEngine

Переваги	Недоліки
Зручний інтерфейс	Складність у проходженні контролю та схвалення проектів перед їхнім запуском
Інтегрованість з системами платежів	Відсутність механізмів оцінки проектів
Електронний обмін документами	Технічні затримки

Як результат розглянувши платформи можемо зробити висновок, кожна з платформ має ряд переваг та надає обширні можливості у пошуку фінансів. Але у кожної з плаформ можна побачити відсутність механізмів оцінки привабливості проектів, що передбачає наступні недоліки:

1. Суб'єктивність експертної оцінки, оскільки в такому випадку оцінка може ґрунтуватися на особистих поглядах експертів, що врезультаті може спотворювати її точність .
2. Недостатня об'єктивність голосування спільноти, яка обумовлена спромежним впливом маркетингових стратегій проектів або маніпуляцій соціальними мережами
3. Висока складність врахування багатофакторності, особливо без використання автоматизованих систем враховування багатьої критеріїв може бути складною задачею

З наведеного вище можна побачити необхідність та актуальність розширення функціоналу існуючих платформ краудфандінгу, з-за допомогою використання автоматизованих систем оцінки проектів. Досягнути цієї мети можна різними способами, наприклад

1. Створення автоматичних опитувань стосовно проекту.
2. Оцінка проектів на базі успішності та інформації про проекти які були створені раніше.
3. Використання штучного інтелекту у частині сентимент-аналізу та обробки природної мови.

В рамках цих способів, найбільш детально варто розглянути способи інтеграції штучного інтелекту до краудфандінгових платформ. Оскільки протягом останнього року модулі штучного інтелекту показують вражаючу ефективність у різних галузях. В контексті проблеми автоматизації оцінки краудфандінгових проектів використання штучного інтелекту може бути у наступних напрямках:

1. Автоматична аналітика проектів, наприклад ШІ може аналізувати додаткові матеріали проекту, такі як соціальні мережі, коментарі, зображення та відео.
2. Сентимент-аналіз та обробка природної мови, для аналізу відгуків та коментарів щодо проекту, його мети, опису з метою визначення загального настрою спільноти. Позитивні відгуки можуть свідчити про великий інтерес та підтримку проекту, в той час як негативні відгуки можуть вказувати на можливі ризики.
3. Прогнозування успіху проекту на базі аналіз попередніх успішних та невдалих проектів для прогнозування успішності нового проекту. Це може бути корисним для спонсорів та платформи для прийняття обґрунтованих рішень.
4. Автоматизоване Ранжування Проектів використовуючи різні критерії, такі як популярність, потенційна дохідність тощо.

Найбільш ефективним способом автоматизації оцінки проектів в рамках краудфандінгової платформи є використання сентимент аналізу та обробки природної мови(NLP).

В результаті чого можливо досягнути наступних цілей:

1. Об'єктивна оцінка, так як алгоритми машинного навчання можуть враховувати безліч різноманітних факторів, наприклад опис проекту, його фінансові показники, коментарі та відгуки спільноти, що дозволяє надати більш об'єктивну оцінку кожному проекту.

2. Аналіз емоцій та настроїв, так як NLP дозволяє, аналізувати тональність тексту, це може виявляти емоційне сприйняття та настрої стосовно проектів на базі його опису та відгуків..
3. Швидкість та масштабованість, оскільки алгоритми машинного навчання та NLP дозволяють обробляти великі обсяги даних в режимі реального часу, що важливо для швидкого та ефективного відбору проектів.

В результаті, застосування сентимент аналізу та аналізу природної мови в сфері краудфандінгу відкриває можливості для більш точної, об'єктивної та ефективної оцінки проектів, що сприяє зростанню успішних та інноваційних ініціатив для осіб які шукають фінансування та полегшення прийняття рішень стосовно надання коштів для інвесторів.

1.3 Висновки розділу 1

В результаті цього розділу було проаналізовано поняття краудфандінгу його основних видів та можливостей. Також було розглянуто поняття краудфандінгових платформ та проаналізовано переваги та недоліки найбільш популярних краудфандінгових систем. В результаті було виявлено їх недоліки та розроблено шляхи їх покращення.

2 РОЗРОБКА МЕТОДУ СТВОРЕННЯ КРАУДФАНДІНГОВОЇ ПЛАТФОРМИ З ІНТЕГРАЦІЄЮ ШТУЧНОГО ІНТЕЛЕКТУ

2.1 Метод створення краудфандінгової платформи з інтеграцією NLP

2.1.1 Основні підходи та складові методу створення краудфандінгової платформи

Метод створення краудфандінгової платформи включає в себе – правила, принципи, моделі та алгоритми створення платформи. Досліджуються моделі та алгоритми розробки як і з серверної так і з клієнтської частини, алгоритми обробки природної мови; методи інтеграції з системою штучного інтелекту.

Для досягнення задач які були поставлені в попередньому розділі розробимо веб-додаток, який реалізує основний функціонал краудфандігових платформ та інтегрується з системами NLP для обробки природної мови. Розглянемо основні підходи для розробки веб-додатків в наш час. Серед основних можна виокремити два, а саме single page application(SPA) та multi page application(MPA), розглянемо кожну з них більш детально.

Сайти SPA складаються лише з однієї сторінки, динамічно замінюючи дані на цій сторінці замість того, щоб обслуговувати абсолютно нові сторінки. Як ви можете собі уявити, це забезпечує надзвичайно швидкий перегляд [9]. В той час як MPA сайти це багатосторінкові програми які можна розглядати як більш «традиційні» веб-сайти, тобто сайти з кількома сторінками та відповідно URL-адресами, які мають завантажуватися щоразу, коли користувач натискає нове посилання або кнопку. Принципи роботи SPA детально зображені на рисунку 2.1.

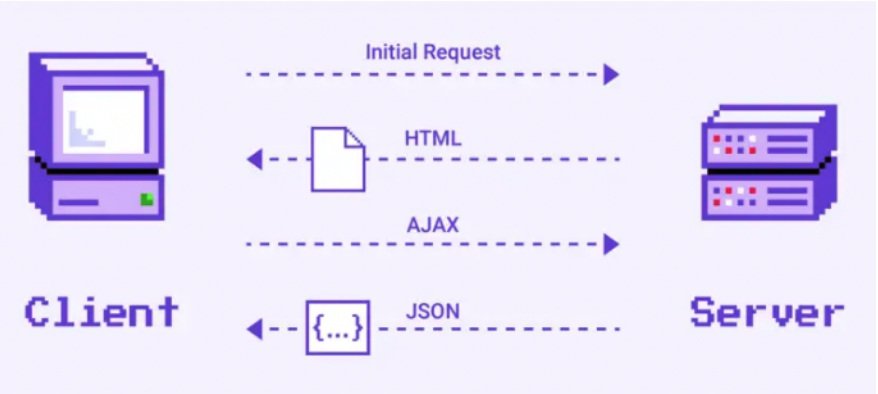


Рисунок 2.1 – Принцип роботи SPA

На рисунку 2.2 представлені принципи МПА.

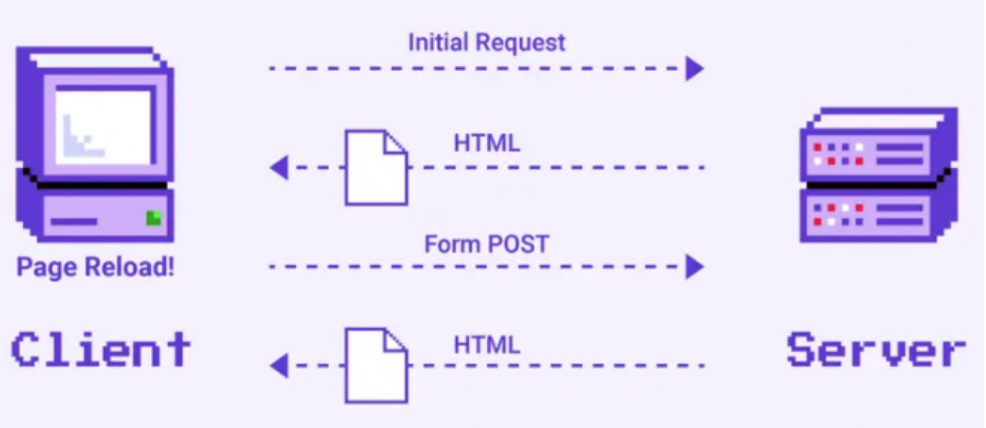


Рисунок 2.2 – Принцип роботи MPA

Протягом довгого періоду вже декілька років поспіль дуже широкого розповсюдження набувають single page application, оскільки мають ряд переваг порівняно з традиційними MPA. Тому розглянемо переваги та недоліки кожного з підходів.

Розглянемо переваги та недоліки single page application, почнемо з переваг:

1. Якісний UX/UI у рідному середовищі браузера - веб-сторінки можна завантажувати та відтворювати швидше в односторінковій програмі, оскільки немає непотрібних відповідей від сервера. Це позбавляє користувача від необхідності перезавантажувати сторінку та часто значно покращує час завантаження сторінки у веб-додатках.

2. Збережена пропускна здатність. Спосіб роботи односторінкових програм в Інтернеті та архітектура, на яку вони покладаються, дозволяють істотно заощадити використання пропускної здатності. SPA підключаються до сервера невеликими партіями, а операції залишаються прямими та простими. Односторінкові програми замість того, щоб надсилати інший запит на сервер, запитують дані JSON для оновлення однієї необхідної властивості на екрані користувача. Це призводить до зменшення кількості даних, що передаються по дроту, звільнення пропускної здатності та загального зменшення мережевого трафіку.
3. Простіший процес виправляння помилок. Використовуючи односторінковий підхід, команда розробників отримує набагато швидший спосіб налаштування та налагодження своєї програми у веб-переглядачі. SPA зазвичай складається з окремих пакетів CSS, HTML і JavaScript, і після завантаження цих ресурсів програма готова до використання.
4. Зосередженість на API. Створюючи сучасну односторінкову програму, команда розробників повинна знати, що програма має широко використовувати API. Це робиться замість генерації логіки інтерфейсу на стороні сервера. Це щаслива нагода скористатися перевагами сучасного підходу до розробки додатків, зосередженого на API . Зокрема, така зосередженість на API, дозволяє розділити серверну та клієнтську частини що в свою чергу полегшує всім сторонам використання широкого спектру різних інструментів.
5. Ефективність процесу кешування . Можливості кешування односторінкових програм роблять їх ідеальними для ситуацій, коли програмі потрібно часто оновлювати один елемент. Тим не менш, клієнт не заперечує один раз кешувати мегабайти JavaScript, необхідні для вашої веб-програми. SPA ефективні для кешування будь-яких локальних даних. Вони надсилають єдиний запит на веб-сервер і зберігають усі виявлені дані програми в сховищі. Локальне кешування надає односторінковим програмам постійний доступ до цих даних, тому користувачі, які

відчувають високу затримку під час навігації інтерфейсом користувача, не зазнають негативного впливу. Вони можуть продовжувати працювати в звичайному режимі, навіть якщо з’єднання погане.

Головним недоліком SPA сайтів була і залишається невеликі можливості для пошукової оптимізації, але в сучасному світі необхідності такої оптимізації відходять на другий план оскільки зазвичай для таких цілей використовуються інші засоби.

Наступним етапом розглянемо переваги та недоліки multi page application. Серед основних переваг можемо виокремити наступні:

1. Легка оптимізація для пошукових систем . Багатосторінкові програми можуть використовувати можливості сканування пошуковою системою без додаткових зусиль з боку розробника. Також MPA можна оптимізувати, щоб вони могли відображатися в результатах пошуку, як вони повинні. Усі відомі практики SEO з легікстю можуть бути застосовні до багатосторінкових додатків.
2. Масштабованість. Підхід MPA дозволяє створювати більш ієрархічні системи з такою кількістю сторінок, скільки потрібно вашому сайту. Завдяки цій архітектурі DOM має майже необмежену ємність, і немає жодних обмежень щодо кількості розширень, з якими може інтегруватися сторінка. Більшість MPA побудовано як динамічно масштабована система, яка має працювати з безліччю послуг і продуктів. Масштабованість проекту навряд чи є проблемою для багатосторінкової веб-програми.

Порівняно з single page application, multi page application мають значно більше недоліків, на які варто звернути увагу, а саме:

1. Низька продуктивність. Продуктивність MPA, завжди потребує багато зусиль для оптимізації. Причина, чому більшість великих компаній обирають багатосторінкові програми, полягає в тому, що цим компаніям доводиться мати справу з великою кількістю даних і вмісту, пропонуючи широкий спектр послуг і продуктів. Однак, якщо бізнес вузький і

веб-додаток не обробляє багато вмісту або має багато ключових слів, які потрібно оптимізувати для веб-пошуку, тоді краще буде вибрати SPA.

2. Менша гнучкість. Багатосторінкові програми покладаються на занадто багато складних механізмів для адаптації інтерфейсу відповідно до вмісту кожної сторінки та пристрою кожного користувача. Таким чином, з обмеженою кількістю ресурсів для роботи, інтерфейс багатосторінкової програми не буде таким гнучким, як інтерфейс односторінкової програми. Крім того, традиційні MPA зазвичай містять більше рівнів інтерфейсу користувача, тому це додатково обмежує можливість адаптації інтерфейсу користувача.
3. Менша можливість повторного використання. Для веб-програми з кількома сторінками немає можливості повторно використовувати той самий код. MPA — це блок тісно пов’язаних передньої та задньої частини. Це означає, що вам доведеться створити окрему кодову базу для мобільного додатка, якщо ваш веб-додаток є багатосторінковим.

Як результат можемо сказати що SPA буде кращим вибором під час розробки краудфандінгової системи, оскільки з його використанням з легкістю можна досягти високої продуктивності, гнучкості та можливості повторного використання.

Метод створення краудфандінгової платформи з інтеграцією NLP включає в себе визначення функціональних блоків застосунку.

Перш за все у системі потрібно забезпечити функціонал реєстрації та авторизації який дозволить користувачам мати власний обліковий запис та зможе обмежити доступ людям які не є користувачами системи. Після чого варто забезпечити можливості перегляду компаній по збору коштів, та способів перегляду, створення або видалення власних кампаній. Розглядаючи питання кампаній варто забезпечити функціонал автоматичного надання оцінки проекту системами NLP. Також в системі варто забезпечити можливість перегляду інформації про організаторів збору, редагування власного профілю. Тому як результат об'єднаємо визначений функціонал в окремі групи, до яких увійде:

1. Функціонал роботи з компаніями, який включає в себе створення, редагування, видалення, перегляд кампаній
2. Функціонал інформації про користувача, який включає редагування профілю, перегляд інформації про організаторів кампанії.
3. Функціонал авторизації в системі, який включає реєстрацію та авторизацію
4. Функціонал автоматичного надання оцінки проекту, відповідно до аналізу засобами NLP.

2.2 Метод створення краудфандінгової платформи з інтеграцією NLP

Метод створення краудфандінгової платформи може бути удосконалений за рахунок використання NLP-технологій, моделі прийняття рішень щодо інвестування проєкту, алгоритмів формування клієнтської та серверної частини, які, на відміну, від існуючих, дозволяють здійснювати аналіз текстової інформації краудфандінгових проєктів, на основі результатів якого виконується автоматизоване оцінювання, сортування релевантних компаній, що дозволяє розширити функціонал платформи, більш детально підготувати дані для прийняття рішення щодо обґрунтованого вибору проєкту для інвестування. Логічною основою такого методу є удосконалена модель прийняття рішення на основі результатів аналізу текстових даних щодо фінансування проєктів, яка, на відміну від існуючих, включає в себе алгоритм аналізу текстових даних та використання елементів штучного інтелекту, що дозволяє отримати структуровану інформацію для прийняття рішення щодо інвестування проєкту.

2.2.1 Удосконалена модель прийняття рішення щодо інвестування проєкту

З огляду на те що основною метою є полегшення прийняття рішення потенційним інвесторам щодо інвестицій у відповідний проект, розглянемо модель людинно-машинної взаємодії, яка буде удосконаленою моделлю прийняття рішення щодо інвестиції у проєкт. Важливими складовими даної моделі, є:

1. Аналіз тональності інформації, які за рахунок технологій NLP дозволяють визначати характер інвестиційного проекту.
2. Агрегація оцінок, оскільки аналіз тональності інформації про проект є лише одним з факторів, варто забезпечити алгоритм агрегації який би обєднував ці оцінки в єдиний показник сентименту.
3. Створення сентимент-індексу, який може служити числовим показником загальної емоційної насиченості інвестиційного проекту
4. Експертна оцінка, оскільки потенційний інвестор може враховувати вагу результатів алгоритмів агрегації, а також вносити власні корективи на основі свого досвіду та експертного бачення
5. Процес прийняття рішення, потенційним інвестором на базі отриманих результатів.

Розглянемо дану модель з математичної точки зору, використовуючи загальні поняття, відтак нехай D - певна множина оцінок інвестиційного проекту з різних джерел. Для кожної оцінки d_i яка входить до D ми отримуємо значення тональності $V(d_i)$ у проміжку [-1,1].

В рамках агрегації використаємо вагові коефіцієнти w_i для відповідних оцінок d_i. Як результат сумарна оцінка може бути визначена як,

$$S = \sum_i * w_i * V(d_i)$$

Сентимент індекс може бути функцією сумарної оцінки $SI = f(S)$.

Оскільки інвестор може внести свої власні корективи додамо H, який буде оцінювати вплив людського фактору, як результат сентимент-індекс з урахуванням людського фактору буде мати наступний вигляд $SI = SI + H$.

В результаті інвестор приймає рішення використовуючи отриманий результат сентимент-аналізу, а також інших факторів:

$$Decision = g(SI),$$

де g - є функцією прийняття рішення.

Отже, математична модель сумарного оцінювання проєктів на основі сентимент-індексу використовує вагові коефіцієнти та показники тональності, що дозволяє сформувати рейтингову таблицю для прийняття рішення.

Для формування структурованої інформації пропонуємо удосконалити модель прийняття рішення на основі результатів аналізу текстових даних щодо фінансування проєктів, а також використання алгоритмів обробки природної мови та базових алгоритмів для представлення відсортованої за параметрами структурованої інформації. Логіка, закладена в модель структурування текстових даних дозволяє визначити такі удосконалення для більш ефективного структурування інформації:

1. Визначення користувачем параметрів пріоритизації проєктів.
2. Фільтрація отриманих результатів за параметрами.
3. Формування візуальної інформації щодо рекомендацій вибору (наприклад, інший колір, визначені пріоритетні проєкти на початку списку).
4. Рекомендації, щодо внесення коректив до даних обробки.

Опис основних етапів виконання процесів структурування та візуалізації інформації є розширенням описової моделі для прийняття рішення щодо фінансування проєкту.

2.2.2 Узагальнений алгоритм роботи клієнтської частини

Відповідно до вищеописаного функціоналу створимо узагальнений алгоритм роботи з точки зору клієнтської частини, який описує можливі розгалуження функціоналу системи. Узагальнений алгоритм роботи клієнтської частини зображений на рисунку 2.3.

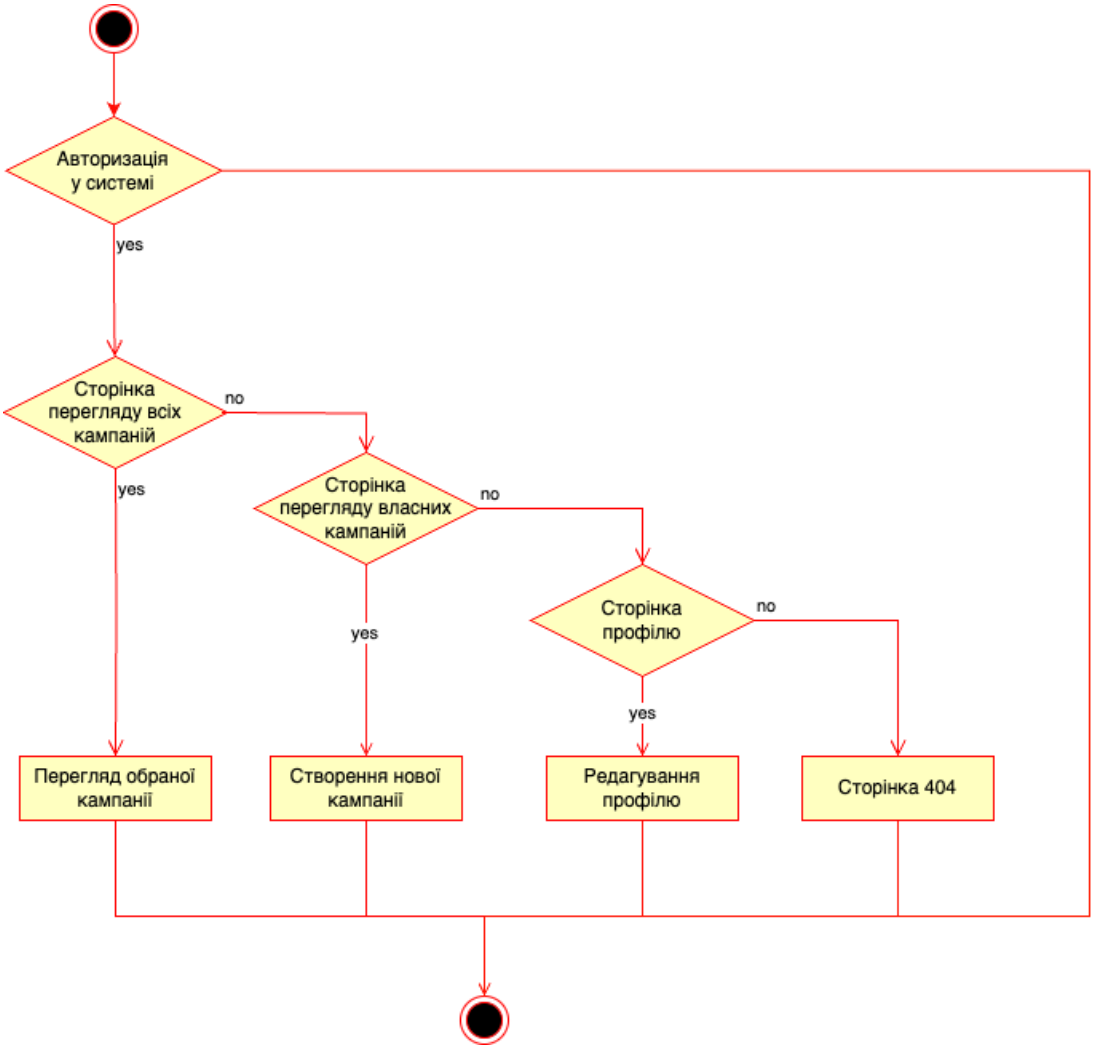


Рисунок 2.3 - Узагальнений алгоритм клієнтської частини краудфандінгової платформи

З рисунку можемо побачити що початок роботи з клієнтською частиною починається з авторизації, яка забезпечує реєстрацію в разі відсутності облікового запису, або вхід у систему. Після входу в систему у користувача є можливість виконувати навігацію по різним сторінкам, які в свою чергу забезпечують відповідний функціонал системи, такий як перегляд створених кампаній для фінансування, можливість перегляду власних створених кампаній, та створення нової. В разі зацікавленості одним з проектів є можливість переглянути його детальну інформацію з оцінкою яку надала система. Відповідно до кожної групи функціоналу, виконується набір API запитів до серверної частини, які необхідні для того аби обробити або надати інформацію клієнту.

Після чого побудуємо узагальнений алгоритм роботи серверної частини, який буде відображати основні етапи роботи серверу та буде опрацьовувати запити від клієнтської частини, в результаті чого виконувати основну бізнес логіку розроблювальної краудфандінгової системи.

2.2.3 Узагальнений алгоритм роботи серверної частини

Узагальнений алгоритм роботи серверної частини розроблювальної краудфандінгової платформи зображений на рисунку 2.4.

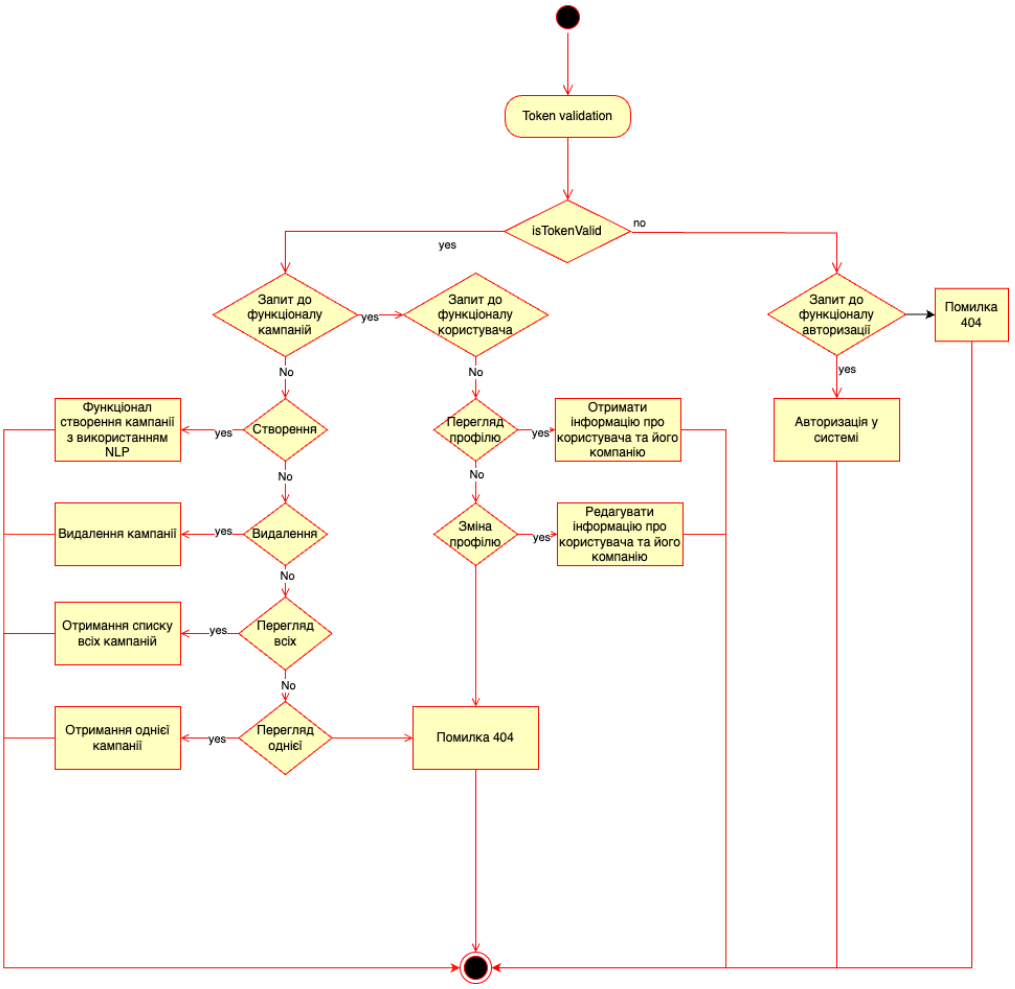


Рисунок 2.4 - Узагальнений алгоритм серверної частини краудфандінгової платформи

З рисунку можемо бачити що початковим етапом обробки запитів на стороні серверу є валідація токену, яка визначає чи є поточний запит від користувача чи в інакшому випадку від людини яка не зареєстрована в системі. В разі якщо запит прийшов від користувача, виконується один з запропонованих сервером методів API, відповідно до модуля який був викликаний. Так в нашому випадку можуть бути опрацьовані операції отримання, створення, видалення краудфандінгових кампаній, інформації про користувача та її редагування, а також функціонал авторизації або повернення помилки в разі, якщо клієнтська частина надсилає запит, яку не обробляє сервер.

2.2.4 Алгоритм забезпечення безпеки системи

Ще одним важливим аспектом, який потрібно передбачити під час реалізації краудфандінгової платформи є безпека системи. Оскільки в наш час завжди існує небезпека для інформації, особливо у мережі інтернет, критично важливо забезпечити безпеку користувачів та їх даних, які використовують платформу. Існує велика кількість різних способів забезпечити безпеку у веб-додатках, ключовим тут стає питання авторизації та автентифікації, оскільки під час цих процесів найбільш правильно виконувати операції які пов'язані з безпекою системи. Існує велика кількість різних способів авторизації та автентифікації у системі, варто розглянути кожен з них та вибрати найбільш оптимальний відповідно до наших вимог. Перш за все розглянемо способи аутентифікації у системі, серед основних можна виокремити наступні:

1. Password Authentication, яка відбувається з використанням імені користувача або його email та паролем.
2. Multi-Factor Authentication, яка потребує двофакторну аутентифікаційну інформацію, таку email, пароль, мобільний телефон, тощо.
3. Certificate Authentication, у якій використовується публічний та приватний ключ для підтвердження ідентичності користувача.
4. Biometric Authentication, включає в себе використання біометричних даних, таких як відбитки пальців, розпізнавання обличчя, голосу тощо.
5. OAuth та OpenID Connect, де відбувається аутентифікація користувача через сторонні сервіси, наприклад Google, Facebook, GitHub.

Серед способів авторизації у системі можна виокремити ряд наступних популярних рішень які використовуються під час вирішення питання надання певного ряду прав доступу користувачу на виконання тих чи інших функцій або ж фільтрування реальних користувачів від людей які не є користувачами. Розглянемо найбільш популярні методи авторизації в системі:

1. Role-Based Access Control, який надає користувачам права доступу відповідно до їхніх ролей.
2. Attribute-Based Access Control, в якому визначаються права доступу на основі різних атрибутів користувача наприклад, час, місцезнаходження, властивості ресурсу до якого необхідно доступ.
3. Discretionary Access Control Користувачі можуть надавати права доступу іншим користувачам до своїх ресурсів.
4. Mandatory Access Control, контроль прав доступу на рівні системи, незалежно від волі користувача.
5. Rule-Based Access Control, авторизація на основі різних правил та умов, визначених для конкретного контексту.
6. JWT, який використовується для передачі інформації про права доступу в зашифрованій формі, так званому токені.

Як результат важливо обрати максимально простий і в той самий час максимально захищений спосіб авторизації та автентифікації. Задля забезпечення простоти використання з точки зору користувача найкращим вибором буде реалізація password authentication, за допомогою email та паролю. В той самий час найбільш ефективним з точки зору авторизації буде використання Json Web Token (JWT), який використовується для безпечної аутентифікації користувачів та надання їм прав доступу у веб-додатках за допомогою jwt-токенів.

Json Web Token — це широко використовуваний стандарт автентифікації та авторизації користувачів, який використовується для безпечного обміну даними [10]. Сам токен складається з декількох частин а саме заголовка, корисного навантаження та підпису.

Захист який надається під час використання JWT може дуже різнитись, тому для ефективної безпеки варто забезпечити правильність налаштування JWT, розглянемо ключові аспекти на яких базується безпека в json web token:

1. Підпис: результуючий токен може бути підписаним для перевірки цілісності даних. Важливо використовувати цей механізм для запобігання зміні токена під час його передачі через мережу.
2. Шифрування, є одним з важливих етапів, який дозволяє шифрувати данні для забезпечення їх конфіденційності.
3. Термін дії який надає обмежений термін дії токену, після якого він стає недійсним.

Якщо дотримуватись цих трьох пунктів можна забезпечити ефективну безпеку у системі. Розглянемо яким чином можна побудувати авторизацію та автентифікацію у системі з використанням JWT.

Авторизація JWT — це механізм автентифікації та авторизації без збереження стану, який усуває потребу в сеансах і файлах cookie. Він забезпечує безпечні засоби передачі інформації, оскільки має цифровий підпис за допомогою секретного ключа, відомого лише серверу. Саме це гарантує, що інформація, яка міститься в JWT, не буде підроблена або змінена під час передачі. Як результат побудуємо алгоритм авторизації в системі. Для цього спочатку потрібно розробити алгоритм авторизації, який отримує в якості вхідних даних email та пароль користувача, шукає у базі даних інформацію про цього користувача за даними полями, в разі якщо не знаходить повертає помилку, або в разі успішної операції пошуку генерує JWT-токен та повертає його клієнту. Алгоритм автентифікації у системі за допомогою email і password зображений на рисунку 2.5.

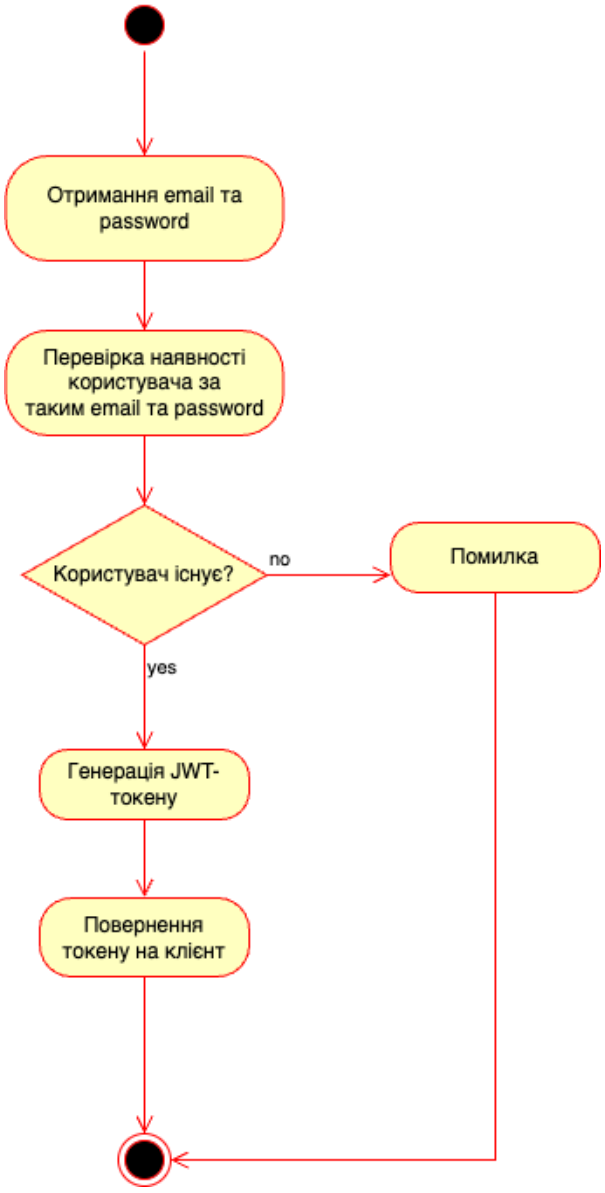


Рисунок 2.5 - Автентифікація у системі

Наступний алгоритм який необхідно розробити, це алгоритм авторизації за допомогою jwt токена який був отриманий під час автентифікації та був відправлений з клієнтської частини до серверу разом з іншим запитом. Як результат при обробці такого запиту перш за все необхідно провалідувати токен секретним ключем який зберігається на сервері, для того щоб зрозуміти чи є отриманий токен, токеном користувача системи, а не спробою шахраїв отримати доступ до платформи. Процес авторизації зображено на рисунку 2.6.

Рисунок 2.6 - Алгоритм авторизації у системі

Як результат реалізація такого механізму авторизації та автентифікації у системі забезпечить високий рівень безпеки системи, та захистить дані користувачів від несанкціонованого доступу, або зміни інформації в їхніх кабінетах.

2.3 Запровадження методу обробки природної мови

2.3.1 Обґрунтований вибір методу обробки природної мови

Обробка природної мови (NLP) – це галузь інформатики, лінгвістики та штучного інтелекту, яка займається взаємодією між людською мовою та комп’ютером. Це допомагає програмувати машини так, щоб вони могли аналізувати та обробляти великі обсяги даних, пов’язаних із природними мовами [11]. Ключовою метою цього напряму є забезпечення комп'ютерам здатності розуміти та використовувати людську мову у всій її складності.

В NLP використовуються методи машинного навчання та статистичні алгоритми для аналізу та обробки текстової інформації. Таким чином НЛП поєднує комп’ютерну лінгвістику зі статистичними моделями, моделями машинного та глибокого навчання. Разом ці технології дозволяють комп’ютерам обробляти людську мову у формі тексту і «розуміти» її повне значення разом із наміром і почуттям того, хто говорить або пише. Алгоритми NLP можуть аналізувати величезний обсяг даних та відокремлювати ключові відомості з тексту, розпізнавати патерни та виявляти відмінності в значущих концепціях.

Обробка природної мови в даний час є рушійною силою машинного інтелекту в багатьох додатках, розглянемо ряд прикладів того як може використовуватись NLP:

1. Виявлення спаму. В даному випадку використовується класифікація тексту NLP для сканування електронних листів на наявність мови, яка часто вказує на спам і фішинг. Ці показники можуть включати надмірне використання фінансових термінів, характерну погану граматику, образливі висловлювання, неналежну терміновість, неправильно написані назви компаній тощо.
2. Машинний переклад. Google перекладач є прикладом широкодоступної технології NLP. По-справжньому корисний машинний переклад передбачає більше, ніж заміну слів однієї мови словами іншої.

Ефективний переклад має точно вловлювати зміст і тон мови введення та перекладати їх у текст із тим самим значенням і бажаним впливом на вихідній мові. Інструменти машинного перекладу досягають значного прогресу з точки зору точності.

3. Віртуальні агенти та чат-боти. Віртуальні агенти, такі як Siri від Apple або Alexa від Amazon, використовують розпізнавання мовлення для розпізнавання шаблонів у голосових командах і створення природної мови, щоб відповідати відповідними діями чи корисними коментарями. Чат-боти виконують таку саму магію у відповідь але у текстовому форматі. Найкращі з них також навчаються розпізнавати контекстуальні підказки про людські запити та використовувати їх, щоб з часом надавати ще кращі відповіді чи варіанти.
4. Аналіз настроїв у соціальних мережах. NLP на даний момент є важливим бізнес-інструментом для виявлення прихованих даних із каналів соціальних мереж. Аналіз настроїв може аналізувати людську мову, яка використовується в публікаціях у соціальних мережах, відповідях, оглядах тощо, щоб виділити ставлення та емоції у відповідь на продукти, рекламні акції та події.
5. Резюмування тексту. Резюмування тексту використовує методи NLP для перероблення величезних обсягів цифрового тексту та створення резюме та конспектів для покажчиків. Найкращі програми для резюмування тексту використовують семантичне міркування та генерацію природної мови (NLG), щоб додати корисний контекст і висновки до резюме.

Як результат можна сказати, що NLP відіграє значущу роль у розробці та вдосконаленні систем автоматизованого аналізу тексту, машинного перекладу, особистих асистентів, систем розпізнавання мови, аналізу відгуків у соціальних мережах, класифікації текстів та багатьох інших додатків тощо. Тому можна ще раз підкреслити актуальність і необхідність використання таких систем в рамках краудфандінгових платформ.

Розглянемо процес та ключові методи і підходи до обробки природної мови. перед безпосереднім опрацюванням тексту виконується так звана попередня обробка даних, яка передбачає підготовку та «очищення» текстових даних для машин, щоб вони могли їх аналізувати. Попередня обробка перетворює дані в придатну для роботи форму та виділяє функції в тексті, з якими може працювати алгоритм. Це можна зробити кількома способами, зокрема:

1. Токенізація: Суть якої полягає в розбитті тексту на окремі частини, які називаються токенами. Токени можуть бути словами, реченнями, символами або навіть фразами. Цей процес полегшує подальший аналіз тексту.
2. Стемінг та лематизація: які спрямовані на зведення слова до їхнього основного вигляду. Це допомагає зменшити розмір словникового запасу та полегшити аналіз тексту.
3. Частиномовний аналіз: передбачає визначення частину мови кожного слова в тексті, наприклад, іменник, дієслово, прийменник, що є важливим для розуміння синтаксичної структури речення.
4. Синтаксичний аналіз: для визначення синтаксичних зв'язків між словами у реченні, що допомагає розібрати синтаксичну структуру тексту.
5. Семантичний аналіз: для визначення слів та їх контексту та виявлення семантичних відношень між словами.
6. Розпізнавання іменованих сутностей: що визначає та класифікує іменовані сутності у тексті, такі як імена осіб, організації, місця, дати тощо.
7. Аналіз емоцій: дозволяє визначати настрій тексту (позитивний, негативний, нейтральний) для оцінки відгуків, коментарів, описів тощо.

Також, синтаксис і семантичний аналіз є двома основними методами обробки природної мови.

Синтаксис — це розташування слів у реченні для надання граматичного сенсу. НЛП використовує синтаксис для оцінки значення мови на основі граматичних правил [12]. В свою чергу синтаксичні методи включають:

1. Розбір. Який є граматичний розбір речення. Приклад: Алгоритм обробки природної мови подає речення «Собака гавкав». Синтаксичний розбір передбачає розбиття цього речення на частини мови, тобто собака = іменник, гавкав = дієслово. Це корисно для більш складних завдань обробки нижньої течії.
2. Сегментація слова. Що є актом взяття рядка тексту та отримання з нього словоформ. Приклад: людина сканує рукописний документ у комп'ютер. Алгоритм зможе проаналізувати сторінку та розпізнати, що слова розділені пробілами.
3. Порушення речень. Означає розміщення межі речень у великих текстах. Приклад: Алгоритм обробки природної мови подає текст «Собака гавкав. Я прокинувся». Алгоритм може розпізнати крапку, яка розділяє речення за допомогою розриву речень.
4. Морфологічна сегментація. Поділ слова на менші частини, які називаються морфемами. Приклад: слово untestably буде розбито на [[un[[test]able]]ly], де алгоритм розпізнає «un», «test», «able» і «ly» як морфеми. Це особливо корисно для машинного перекладу та розпізнавання мовлення.
5. Витікання. Де в основі лежить поділя слова з флексією на кореневі форми. Приклад: у реченні «Собака гавкав» алгоритм зможе розпізнати корінь слова «гавкав» — «гавкати». Це було б корисно, якщо користувач аналізував текст на всі випадки слова bark, а також на всі його сполучення. Алгоритм бачить, що це, по суті, одне й те саме слово, навіть якщо літери різні.

В свою чергу семантика передбачає використання та значення слів. NLP застосовує алгоритми для розуміння значення та структури речень. Методи семантики включають:

1. Розпізнавання сенсу слова. Де виводиться значення слова на основі контексту. Приклад: розглянемо речення «Свиня в кошарі». Слово перо має різні значення. Алгоритм, який використовує цей метод, може зрозуміти, що використання слова ручка тут стосується огородженої території, а не інструменту для письма.
2. Розпізнавання іменованих сутностей. Це визначення слів, які можна розділити на групи. Як приклад алгоритм, який використовує цей метод, може проаналізувати статтю новин і визначити всі згадки про певну компанію чи продукт. Використовуючи семантику тексту, він міг би розрізняти сутності, які візуально однакові. Наприклад, у реченні «Син Деніела МакДональдса пішов у МакДональдс і замовив Хеппі Міл» алгоритм міг розпізнати два екземпляри «МакДональдз» як дві окремі сутності — одну — ресторан, а іншу — людину.
3. Генерація природної мови. Де відбувається використання бази даних для визначення семантики слів і створення нового тексту. Наприклад, Алгоритм може автоматично написати зведення результатів платформи бізнес-аналітики, зіставляючи певні слова та фрази з характеристиками даних у платформі BI. Іншим прикладом може бути автоматичне створення статей новин або твітів на основі певного тексту, який використовується для навчання.

Ці методи разом з іншими дозволяють розуміти, аналізувати та обробляти природну мову, що важливо в багатьох сферах, включаючи машинний переклад, пошукові системи, аналіз соціальних мереж та інше.

Як результат для надання автоматизованої оцінки проектам, необхідно використовувати методи, які можуть опрацьовувати текст, та розуміти його суть та контекст. Відповідно до цього необхідно використовувати всі методи попередньої обробки тексту для того щоб підготувати текст для подальшої обробки, та використовувати семантичний метод розпізнавання сенсу слів, який забезпечить розпізнавання тональності тексту, в результаті якого тексту можна

буде отримувати значення тональності відповідно до якого будемо формувати оцінку.

2.3.2 Інтеграція NLP в архітектуру платформи

Інтеграція механізму обробки природної мови у архітектуру краудфандінгової платформи є ключовим аспектом для створення автоматизованого механізму оцінки краудфандінгових проектів. Оскільки NLP дозволяє розпізнавати, аналізувати людську мову, визначати тональність тексту в результаті чого можна використовувати ці данні для надання чітко визначеного статусу для того чи іншого проекту.

Для того щоб інтегрувати механізми NLP у веб-систему краудфандінгової платформи, варто забезпечити окрему функціональну частину яка б мала змогу отримувати текстові дані які необхідно потрібно опрацювати та повертала розрахований статус.

Базовий алгоритм роботи такої частини має включати в себе, отримання текстових даних на вхід, їх попередню обробка після чого виконання аналізу тональності на базі наявного або створеного словника у системах NLP, після чого виконання перетворення отриманого показнику тональності до відповідного статусу у нашій системі, який у майбутньому буде присвоюватись тому чи іншому проекту який створюється на платформі.

Розробимо алгоритм попередньої обробки тексту який буде виконуватись під час отримання текстових даних на аналіз та визначення тональності. Алгоритм роботи такого функціоналу зображений нижче на рисунку 2.7.

Рисунок 2.7 - Алгоритм роботи попередньої обробки данних та визначення тональності механізмом NLP

В результаті основними етапами роботи алгоритму попередньої обробки даних та визначення тональності тексту який оцінюється буде виконання наступних операцій, розпізнавання скорочень, приведення тексту до єдиного регістру, вилучення спецсимволів, токенізація слів, стеммінг та як результат визначення тональності.

2.4 Архітектура та структура краудфандінгової платформи

Відповідно до розглянутих у попередньому підрозділі алгоритмів роботи функціональних блоків системи, створимо розглянемо архітектури системи на різних рівнях. Перш за все розглянемо високорівневу архітектуру взаємодії клієнту та серверу, як результат це буде клієнт серверну архітектуру, яка застосовується у більшості випадків коли розробляють веб-застосунки. Деталі архітектури наведено на рисунку 2.8.

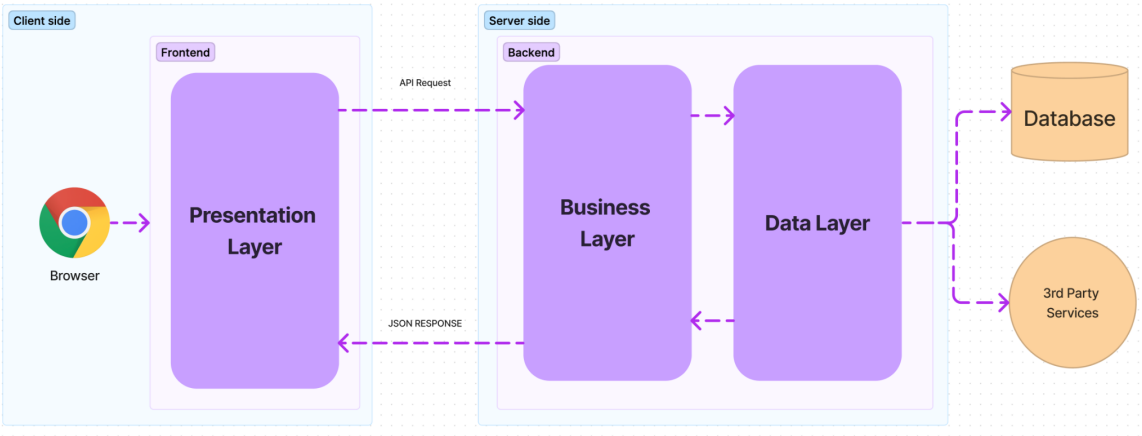


Рисунок 2.8 - Базова верхорівнева архітектура веб-системи краудфандінгової платформи

Розглянемо кожну складову більш детально, почнемо з клієнтської частини, яка є інтерфейсом для користувачів, та який надає можливість користувачам взаємодіяти з платформою та виконувати різноманітні дії в рамках наданої платформою функціональністю. Ця системи має за мету забезпечити зручний та привабливий інтерфейс для користувачів, де вони можуть створювати свої кампанії та підтримувати інші.

Серверна частина є центральною складовою системи, оскільки обробляє та керує запитами користувачів, забезпечує безпечні та ефективні транзакції, відповідає за зв'язок з зовнішніми сервісами та базами даних. Крім того в рамках серверної частини буде інтегровано модуль штучного інтелекту, який

відповідає за визначення тональності текстів кожного з краудфандінгових проектів.

Як результат клієнтська та серверні частини взаємодіють з-за допомогою HTTP-запитів які клієнтська частина відправляє до серверу з метою отримати відповідь.

Після чого деталізуємось на один рівень та розглянемо архітектуру модулів з яких складається клієнтська та серверна частини відповідно. Кожен модуль відокремлює та інкапсулює конкретну частину функціоналу додатку. У більшій частині функціоналу модулі клієнтської частини відповідають аналогічним модулям у серверній, але не всі. Так наприклад на стороні сервера буде реалізовано додатковий модуль який буде взаємодіяти з механізмами NLP. Архітектура модулів клієнтської частини зображена на рисунку 2.9.

Рисунок 2.9 – Архітектура модулів серверної частини

Архітектура модулів серверної частини клієнтської зображена на рисунку 2.10.

Рисунок 2.10 – Архітектура модулів клієнтської частини

Розробивши архітектуру модулів системи клієнтської та серверної частин, розглянемо головні концепції архітектури на рівні бізнес правил. Для написання масштабованої та легко підтримуваної кодової бази використаємо концепцію “чистої” архітектури як була визначена Робертом Мартіном у його книзі “Чиста архітектура” [13]. Серед основних елементів цієї архітектури можна виділити тільки два основних елементи, а саме політики та процедури, політиками є бізнес-правила, а процедурами - елементи які необхідні для виконання цих бізнес правил. Схема концепції чистої архітектури зображена на рисунку 2.11.

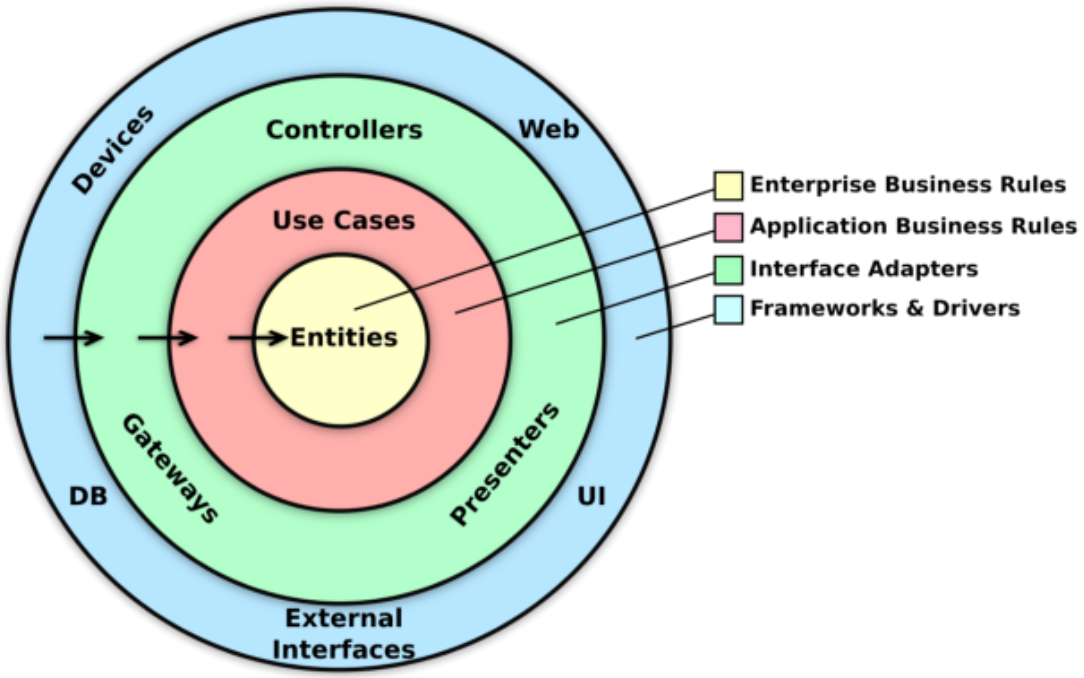


Рисунок 2.11 - Схема чистої архітектури

З рисунку можемо бачити що архітектура складається з ряду концентричних шарів які взаємодіють один з одним до ядра. Кожен з відповідних шарів представляє різні області програмного забезпечення, внутрішня частина є політиками а зовнішня механізмами. Основним правилом для данного виду архітектури є так зване правило залежностей, яке стверджує те, що бізнес-логіка повинна бути ізольованою від деталей інфраструктури. Це означає, що подробиці інфраструктури не повинні проникати всередину бізнес-логіки. На противагу цьому, бізнес-логіка може і повинна бути повністю відокремлена і не повинна залежати від деталей інфраструктури. Таким чином, можна сказати, що елементи внутрішнього шару не можуть мати жодної інформації про елементи зовнішнього шару. Класи, функції, змінні, формат даних або будь-яка сутність, оголошена на зовнішньому рівні, не повинні згадуватися в коді внутрішнього рівня.

Внутрішній рівень — це рівень Entities, який містить бізнес-сутності системи, що містить найзагальніші правила найвищого рівня. Сутність може бути набором структур даних і функцій або об’єктом з методами, якщо ця сутність може використовуватися кількома програмами. Цей рівень не повинен бути змінений змінами на зовнішніх рівнях, тобто жодні операційні зміни в будь-якій програмі не повинні впливати на цей рівень. В рамках нашої краудфандінгової системи серед Entites можна виокремити наступні:

1. Ticket
2. Company
3. User
4. PaymentMethod

Нижче наведена діаграма класів Entity розроблювальної краудфандінгової платформи на рисунку 2.12 [14].

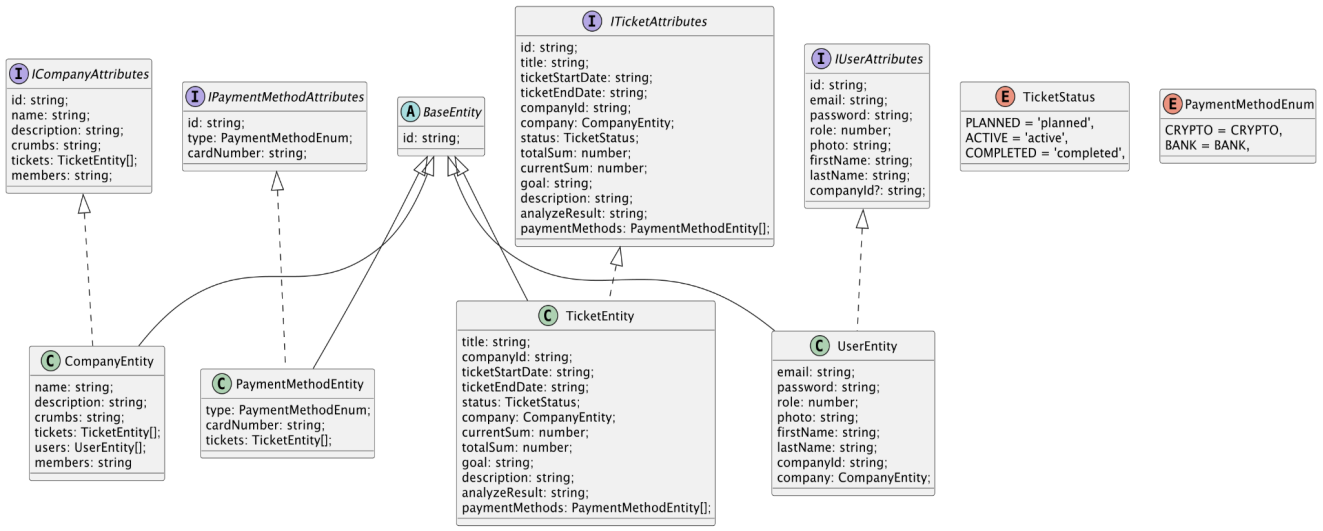


Рисунок 2.12 - Діаграма класів Entity системи

Наступним рівнем в чистій архітектурі йде рівень use case, який містить специфічні для програми бізнес-правила, та в якому реалізуються всі варіанти бізнес логіки системи. На цьому рівні організовується потік даних до об’єктів і від них і направляють об’єкти у застосуванні важливих бізнес-правил для досягнення цілей варіантів використання. На цей шар також не повинні впливати зовнішні шари, і ваші зміни не повинні впливати на рівень сутностей. Однак якщо зміниться деталі варіанту використання, це вплине на частину коду на цьому рівні.

Розглядаючи цей рівень архітектури в рамках нашого додатку, до нього увійдуть різні сервіси та репозиторії, які відповідають конкретній бізнес сутності у системі або процесу, в нашому випадку це TicketService, CompanyService, UserService, AuthService. Діаграма класів цих сервісів наведена на рисунку 2.13.

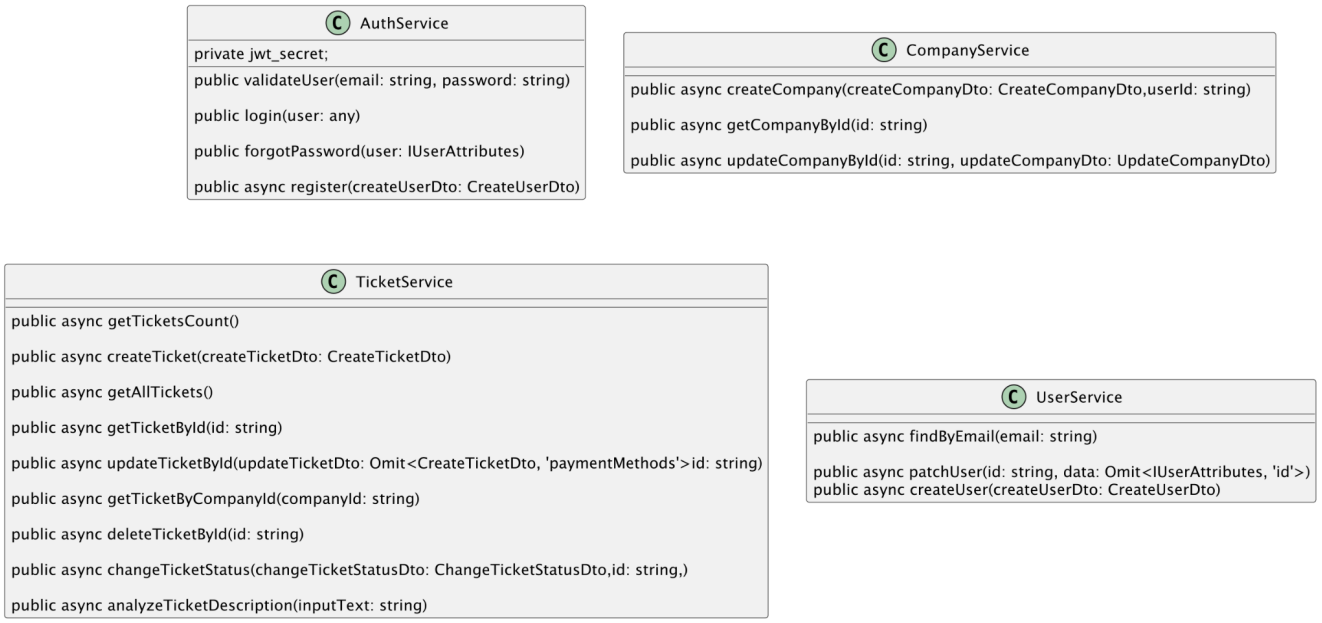


Рисунок 2.13 - Діаграма класів сервісів системи

Наступним йде рівень інтерфейсних адаптерів, який має набір адаптерів, які в свою чергу перетворюють дані в найбільш зручний формат для шарів навколо нього. Наприклад він бере дані з бази даних, наприклад, і перетворює їх у найбільш зручний формат для рівнів об’єктів і випадків використання. Зворотний шлях перетворення також може бути здійснений, від даних із внутрішніх шарів до самих зовнішніх. Презентатори, представлення та контролери належать до цього рівня, в нашому випадку до них будуть відноситись контроллери, такі як AuthController, TicketController, CompanyController. Діаграма цих класів наведена на рисунку 2.14.

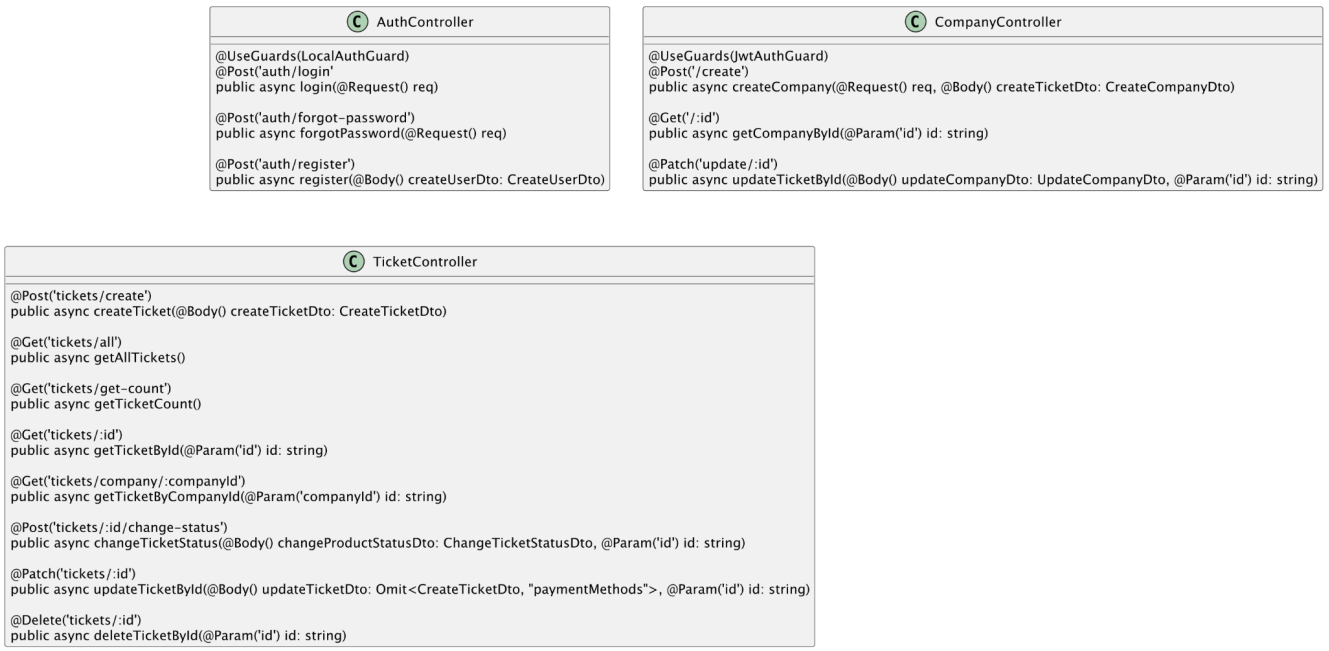


Рисунок 2.14 - Діаграма класів контролерів системи

Останнім є зовнішній рівень діаграми, який складається з фреймворків, баз даних, тощо. Цей рівень містить код, який встановлює зв’язок із рівнем адаптерів інтерфейсу.

2.5 Висновки розділу 2

У цьому розділі було розглянуто метод створення краудфандінгової платформи який включає в себе, алгоритми роботи системи в цілому та окремо її серверної та клієнтської частин. Розглянуто ключові поняття та методи обробки природної мови, та інтегровано механізми NLP у функціонал поточного додатку. В результаті було розроблено архітектуру розроблювальної системи на всіх рівнях.

3 РОЗРОБКА КРАУДФАНДІНГОВОЇ СИСТЕМИ З ІНТЕГРАЦІЄЮ ШТУЧНОГО ІНТЕЛЕКТУ

3.1. Вибір технологій для реалізації клієнтської частини

Для розробки клієнтської частини веб-додатку краудфандінгової платформи, необхідно обрати правильні технології які будуть використовуватись для її розробки. Так, обрана технологія має забезпечувати швидкодію та масштабованість, що є ключовими показниками при виборі технології. Оскільки розроблювальна система є веб-додатком, основною мовою програмування яка буде використовуватись під час написання є JavaScript. Але в наш час використання просто JavaScript є не самим оптимальним рішенням, для автоматизації та полегшення ряду процесів під час розробки слід використовувати фреймворки. Серед найбільш популярних JavaScript фреймворків в даний момент можна виокремити React, Vue та Angular. Розглянемо кожен з фреймворків детально та оберемо ефективніший відповідно до наших задач.

Першим фреймворком, який ми розглянемо є React. React — це декларативна, ефективна та гнучка бібліотека JavaScript для створення інтерфейсів користувача, яка дозволяє створювати складні інтерфейси користувача з невеликих ізольованих фрагментів коду, які називаються «компонентами» []. React використовується для розробки веб-додатків, мобільних додатків (за допомогою React Native). Він дозволяє розробникам створювати сучасні та інтерактивні інтерфейси, і має велике співтовариство, що надає багато ресурсів та підтримку для розробників. Серед основних особливостей даного фреймворку можна виокремити:

1. Компоненту архітектуру, яка має за собою концепцію невеликих компонентів які об'єднують в собі HTML, JS та CSS стилізацію. Поєднання таких компонентів, створює користувацькі інтерфейси та повноцінні додатки.

2. Використання віртуального DOM, який оптимізує рендеринг сторінок.
3. Одночасна підтримка як функціональних так і класових компонентів.

Наступним фреймворком який варто розглянути є - Vue.js. Vue — це фреймворк JavaScript для створення інтерфейсів користувача. Він створений на основі стандартних HTML, CSS і JavaScript і забезпечує декларативну та компонентну модель програмування, яка допомагає ефективно розробляти інтерфейси користувача, прості чи складні [16]. Vue.js використовується для розробки різноманітних веб-додатків, від невеликих прототипів до великих enterprise систем. Він має доволі велику та активну спільноту користувачів і широкий вибір розширень, що дозволяє розробникам підлаштовувати його відповідно до своїх потреб. Серед особливостей Vue можна виокремити наступні:

1. Декларативний синтаксис, оскільки Vue використовує декларативний підхід для опису інтерфейсу. Під час написання коду описується, який вигляд має мати ваша сторінка у вигляді шаблонів, і Vue відповідає за відображення цього шаблону на сторінці.
2. Компонентна архітектура, яка по аналогії з React дозволяє створювати і використовувати компоненти, які можна вкладати один в одного. Це сприяє повторному використанню коду та структуризації інтерфейсу.
3. Реактивність. Vue надає можливість встановлювати зв'язки між даними та їх відображенням. Зміни в даних автоматично оновлюють відображення і навпаки, завдяки чому інтерфейс завжди відповідає поточному стану даних.

Останнім з фреймворків, які варто розглянути є Angular, який є фреймворком для розробки додатків і платформа для створення ефективних і складних односторінкових додатків[17]. Angular є одним з найпопулярніших і найпотужніших фреймворків для розробки веб-додатків та інтерфейсів. Він був створений і розроблений компанією Google та має велику спільноту користувачів та розробників. Розглянемо велику кількість основних особливостей даного фреймворку, до них можна віднести:

1. Типізація: Angular використовує TypeScript, як основну мову програмування. TypeScript - це сильно типізована мова, яка дозволяє виявляти помилки під час компіляції та підвищує надійність коду.
2. Компонентна архітектура. Angular сприяє створенню користувацьких інтерфейсів за допомогою компонентів. Кожен компонент - це окремий блок, який містить HTML-шаблон, логіку та стилі.
3. Модульність: Angular дозволяє розділяти додаток на невеликі модулі, що полегшує організацію та розширення коду.
4. Реактивність: Angular має підтримку реактивного програмування за допомогою RxJS. Це дозволяє працювати з асинхронними подіями та потоками даних.
5. Маршрутизація. Angular має вбудовану систему маршрутизації, яка дозволяє створювати односторінкові додатки з різними сторінками та URL.
6. HTTP запити. Angular надає можливості для виконання HTTP-запитів до серверів, що дозволяє взаємодіяти з серверними API.
7. Підтримка PWA.

Як результат, маючи таку велику кількість особливостей Angular використовується для розробки різноманітних додатків, включаючи веб-додатки, мобільні додатки (за допомогою NativeScript або Ionic). Він підходить для створення великих корпоративних додатків та додатків зі складною логікою. Angular важчий у вивченні порівняно з іншими фреймворками, але він надає широкий набір інструментів та можливостей для розробки сучасних, надійних та масштабованих додатків.

Виходячи з розглянутої інформації про кожний з фреймворків порівняємо їх за певним набором параметрів, які наведені нижче:

1. Розмір та продуктивність:

   React: Зазвичай володіє найменшим розміром завдяки можливості вибору окремих бібліотек.

   Vue.js: Також має досить компактний розмір та хорошу продуктивність.

Angular: Може бути більшим за React і Vue.js, що може вплинути на завантаження сторінки та продуктивність. Проте Angular має оптимізації для покращення продуктивності.

2. Компонентна архітектура:

   React: Вимагає вибору додаткових бібліотек, таких як Redux, для керування станом і зв'язками між компонентами.

   Vue.js: Має вбудований механізм управління станом та компонентами.

   Angular: Має вбудований механізм управління станом, модулі та ін'єкцію залежностей.

3. Інструменти та спільнота:

   React: Має велику та активну спільноту розробників і багато сторонніх бібліотек і розширень.

   Vue.js: Має швидко зростаючу спільноту та багато корисних розширень.

   Angular: Має потужний набір інструментів та велику спільноту, але може бути менш гнучким у порівнянні з React і Vue.js.

4. Використання великих проектів:

   React: Добре підходить для великих проектів, зокрема тому, що ви можете вибирати необхідні бібліотеки та плагіни.

   Vue.js: Хороший вибір для великих проектів завдяки простому та лаконічному синтаксису.

   Angular: Рекомендується для великих проектів, особливо великих корпоративних додатків.

5. Типізація:

   React: Не накладає обов'язкового використання типізації.

   Vue: Підтримує типізацію за допомогою TypeScript та Flow, а також може бути використаний з чистим JavaScript.

   Angular: Фреймворк розроблений з використанням TypeScript, що надає вищий рівень типізації та безпеки в порівнянні з React і Vue.js. В Angular типізація вбудована у фреймворк і вимагає строгого дотримання TypeScript.

6. Масштабованість

React: Може бути використаний для проектів різного розміру, але великі проекти вимагають додаткових зусиль для організації коду та управління станом.

Vue: Також потребує додаткових зусиль для великих проектів.

Angular: Спеціально розроблений для масштабних проектів та корпоративних додатків. Він має вбудовані інструменти для масштабування, такі як модулі та ін'єкція залежностей. У великих проектах Angular може забезпечити більш високий рівень структурованості та контролю.

7. Стейт-менеджмент

React: Не надає конкретного рішення для стейт-менеджменту у вбудованому вигляді. Розробники можуть вибирати з різних бібліотек, таких як Redux, MobX, або використовувати стандартний локальний стан компонентів.

Vue: Не надає конкретного рішення для стейт-менеджменту у вбудованому вигляді, але може бути використана бібліотека Vuex надає централізоване зберігання стану, мутації для зміни стану та можливість слідкувати за змінами стану.

Angular: Angular використовує вбудований механізм стейт-менеджменту через сервіси та ін'єкцію залежностей та надає можливості для створення сервісів, які управляють станом та можуть бути ін'єктовані у компоненти. Велика частина стейту зберігається в об'єктах, що називаються сервісами. У великих корпоративних додатках Angular надає більший рівень контролю над станом.

На основі наявної інформації створимо порівняльну таблицю для цих фреймворків.

Таблиця 3.1 Порівняння фреймворків для клієнтської частини

	Vue	Angular	React
Продуктивність	+	+	+
Компонентність	+	+	+
Спільнота	+	+	+
Використання для великих проектів	-	+	-
Типізація	-	+	-
Масштабованість	-	+	-
Стейт-менеджмент	-	+	+

З порівняльної таблиці можна зробити висновок що для реалізації клієнтської частини краще за все підходить фрейворк Angular оскільки він забезпечує продуктивність, вбудовану типізацію та стейтменеджмет, а також є фреймворком яких забезпечує легку масштабованість кодової бази.

3.2 Вибір технологій для реалізації серверної частини

Як і при розробці клієнтської частини, так і при розробці серверної частини, технології які мають використовуватись для розробки, повинні забезпечувати швидкодію та масштабованість. Серед основних фреймворків які використовуються для розробки серверних додатків на базі Node.js можна виокремити два це express.js та nest.js. Розглянемо кожен більш детально та почнемо з express.js.

Express — це мінімалістичний фреймворк Node.js. Незважаючи на те, що він охоплює кілька основних аспектів створення програми на стороні сервера, він є популярним через свою простоту, гнучкість і продуктивність [17]. Express.js використовується для розробки веб-серверів, веб-додатків та API. Він

дозволяє розробникам створювати серверну логіку за допомогою JavaScript у середовищі Node.js.

В свою чергу Nest.js — це платформа для створення ефективних, масштабованих серверних програм Node.js [18]. Nest побудовано на основі поширених фреймворків Node.js таких як Express та Fastify. Він використовує TypeScript і поєднує елементи об’єктно-орієнтованого, функціонального і функціонально реактивного програмування. Також Nest реалізує принципи SOLID, що забезпечує масштабованість та гнучкість архітектури додатків, які розробляються [19].

Порівняємо ці два фреймворки за рядом параметрів.

1. Підтримка TypeScript:

Express: Express - це найпростіший та мінімалістичний фреймворк, який надає базовий набір інструментів для обробки HTTP-запитів. Express не накладає обов'язкового використання TypeScript або будь-якого конкретного структурного патерну. Все робиться за бажанням розробника.

NestJS: NestJS визначає багатомодульну структуру для додатків та надає вбудовану підтримку TypeScript. Він рекомендує використовувати TypeScript і пропонує об'єктно-орієнтований підхід до розробки.

2. Структура коду та модульність:

Express: Express надає велику свободу у виборі структури проекту, і розробнику слід самостійно організувати код та додавати бібліотеки для розширення функціональності.

NestJS: NestJS надає модульну структуру та визначені патерни для організації коду. Він рекомендує використовувати модулі, контролери та провайдери для структуризації додатку.

3. Типізація та статична перевірка типів:

Express: Express не вимагає використання TypeScript, і статична перевірка типів не є обов'язковою. Ви можете використовувати JavaScript або вибрати TypeScript за бажанням.

NestJS: NestJS рекомендує використовувати TypeScript і надає вбудовану підтримку для статичної перевірки типів, що забезпечує більш високий рівень безпеки коду.

4. Інтеграція з ORM та базами даних:

Express: Express не надає вбудованих інструментів для роботи з базами даних або ORM. Розробники повинні вибирати бібліотеки самостійно.

NestJS: NestJS підтримує інтеграцію з різними ORM, такі як TypeORM, та надає зручність для створення моделей та міграцій.

На базі цієї інформації створимо порівняльну характеристику серверних фреймворків, яка наведена у таблиці 5.

З таблиці та порівняльної характеристики можемо побачити що задля забезпечення масштабованості у нашому серверному додатку, під час розробки варто використовувати Nest.js.

Ще одним важливим пунктом при виборі технологій для розробки серверної частини є вибір бази даних. В рамках нашої платформи важливим критерієм є легка масштабованість та структурованість даних.

Таблиця 3.2 – Порівняльна характеристика серверних фреймворків

	Express.js	Nest.js
Підтримка TS	+	+
Модульність	-	+
Типізація	-	+
Інтеграція з ORM	-	+

Розглянемо найпоширеніші варіанти баз даних та оберемо найкращий для нас варіант. Перш за все варто визначитись який тип баз даних нам потрібен. Їх існує велика кількість, але зупинитись варто на реляційному та документо-орієнтовному типі, оскільки в наш час вони використовуються частіше за все і підходять для 95% задач.

Реляційна база даних – це база даних, в якій усі дані, доступні користувачу, організовані у вигляді таблиць, а всі операції над даними зводяться до операцій над цими таблицями. Для представлення реляційних баз даних розроблена формальна теорія баз даних, теоретичну основу якої складає алгебра та математична логіка. Реляційна модель орієнтована на організацію даних у вигляді двовимірних таблиць. Кожна реляційна таблиця являє собою двовимірний масив [20]. Реляційні база даних використовуються в різних сценаріях і додатках, де важлива структурованість даних та можливість виконання складних операцій зв'язку та аналізу, оскільки забезпечують надійність, структурованість та можливість виконання складних операцій з даними.

В свою чергу документо-орієнтовані баз даних характеризуються тим, що в них нема схеми організації даних[21]. Документо-орієнтована модель передбачає, що вся інформація зберігатиметься у вигляді документів формату BSON, який

походить від JSON і розшифровується як javascript object notation. BSON – це бінарний JSON, цей формат дозволяє легко описувати об’єкти та інші структури даних та може бути з легкістю прочитаний людиною [22]. Виходячи з цього можна зробити висновок, що задля забезпечення структурованості даних варто обирати серед реляційних БД. В такому разі розглянемо найбільш популярні СУБД для управління реляційними базами даних, до них можна віднести MySQL, MSSQL, PostgreSQL.

MySQL є найпопулярнішою у світі базою даних з відкритим кодом. За даними DB-Engines , MySQL є другою за популярністю базою даних після Oracle Database. Серед основних особливостей mysql можна виокремити її швидкодію та безпеку [23].

В свою чергу, MSSQL є комерційною системою управління базами даних, розробленою корпорацією Microsoft. Серед її головних особливостей є широкий спектр можливостей реплікації, підтрика T-SQL та інтеграція з іншими інструментами Miscrosoft [24].

Останньою СУБД є PostgreSQL яка надає багато продуктивних можливостей та розширюваності, і є однією з найпопулярніших СУБД з відкритим вихідним кодом [25].

Розглянемо порівняльну характеристику даних СУБД яка наведена у таблиці 3.3.

Таблиця 3.3 – Порівняльна характеристика реляційних СУБД

	MySQL	MSSQL	PostgreSQL
Безкоштовність	+	-	+
Наявність великої кількості функцій	-	+	+
Масштабованість та реплікація	+	+	+

Виходячи з порівняльної таблиці можемо побачити що наявні СУБД є відносно співставними, але за рахунок ширшого спектру фукнцій postgres буде кращим вибором порівняно з іншими опонентами.

3.3 Вибір бібліотеки яка надає функціональність NLP

В рамках інтеграції механізмів обробки природньої мови в систему карудфандінгової платформи важливим аспектом є правильно вибрати ефективну бібліотеку для реалізації функціональності NLP. У цьому підрозділі розглянемо існуючі бібліотеки які надають рішення обробки природньої мови в середовищі nodejs.

На сьогоднішній день існує значна кількість бібліотек для обробки природної мови в середовищі Node.js. Деякі з них є загального призначення, тоді як інші спеціалізовані на конкретні задачі. Найбільш попялрними є наступні:

1. Natural

2. Franc
3. NLP.JS

Кожна з цих бібліотек надає функціональність обробки природньої мови, та має свої переваги та недоліки. Розглянемо кожну з цих бібліотек більш детально.

Natural є біблотекою для Node.js, яка надає різноманітні інструменти та утиліти для завдань обробки природної мови, таких як токенізація, створення основи та класифікація. Він підтримує кілька мов і надає попередньо підготовлені моделі для таких завдань, як аналіз настроїв, додавання тегів до частин мови та розпізнавання іменованих об’єктів. Бібліотека також дозволяє легко налаштовувати та навчати моделі за допомогою даних, наданих користувачем [26].

Franc це бібліотека обробки природної мови для Node.js, яка визначає мову певного тексту. Він підтримує понад 300 мов і використовує статистичні методи для визначення мови на основі частоти слів і розподілу букв. Franc простий у використанні та може бути інтегрований у проекти Node.js для автоматичного визначення мови контенту, створеного користувачами, наприклад коментарів чи повідомлень чату. Завдяки своєму простому API Franc робить визначення мови доступним для розробників усіх рівнів[27].

NLP.JS є потужною та універсальною бібліотекою для обробки природної мови в середовищі Node.js. Вона спеціально створена для того, щоб забезпечити програмістам доступ до таких та функцій як токенізація, лематизація, стемінг, частиномовне маркування, аналіз емоцій, аналіз ключових слів.

Порівняємо бібліотеки між собою за наступними критеріями:

1. Наявність повного спектру функціональних можливостей NLP
2. Простота використання, наявність документації
3. Активність спільноти та підтримка

Порівняння наведено в таблиці 3.4.

Таблиця 3.4 – Порівняння бібліотек NLP

Бібліотека	Функціональність	Документація	Активність спільноти
Natural	+	+	+
NLP.js	+	+	-
Franc	-	-	-

З порівняння можемо зробити висновок, що Natural є бібліотекою з багатшим функціоналом для обробки природної мови, оскільки вона надає широкий спектр інструментів та можливостей, і має розвинуту спільноту що робить її вигідним вибором для багатьох NLP-задач.

3.4 Програмна реалізація серверної частини

Розпочнемо програмну реалізацію платформи з реалізації серверної частини. В підрозділах вище було з'ясовано що для написання серверної частини необхідно використовувати фреймворк Nest.js та posgtresql в якості СУБД. Перш за все необхідно створити nest проект за допомогою наступної команди Nest CLI:

$ nest new crowdfinding-be

Дана команда створить початкову архітектуру та файлову структуру, базового серверного додатку з використанням фреймворку nest.js [28]. Яка зображена на рисунку 3.10.

Рисунок 3.1 - Базова файлова структура серверу в Nest.js

З рисунку можна побачити що було створено 5 файлів, які необхідні для запуску самого простого серверу. Розглянемо кожен з файлів і його призначення більш детально. Почнемо з файлу main.ts, який містить в собі створення серверу на базі головного модуля AppModule та його запуск на вказаному порті, за замовчуванням він дорівнює 3000.

Наступні файли будемо розглядати через призму шарів “чистої архітектури” аби зрозуміти який з типів файлів за що відповідає. Почнемо з модулів та розглянемо створений app.module.ts.

Модулі – це контейнери для різних частин програми, таких як контролери, сервіси та інші пов’язані компоненти з яких складається та чи інша частина додатку [29]. Вони служать способом організації та інкапсуляції функціональності розроблювальної програми. Таким чином, модуль можна розглядати як певний розділ програми який відповідає за її певну частину. Як було сказано раніше, модуль вміщує в собі контроллери та сервіси, розглянемо ці частини також.

Контролери в NestJS відповідають за обробку вхідних HTTP-запитів і надання відповідей клієнту після їх обробки [29]. Таким чином можна сказаи що вони є точками входу в логіку нашого серверу. Кожен контролер може бути пов’язаний із певним маршрутом або кінцевою точкою та відповідає за визначення методів обробки запитів. Контролери використовують декоратори

для визначення шляху маршруту, HTTP методів (GET, POST, PUT, DELETE тощо) і параметрів запиту. Після надходження запиту до контролера, він починає ваємодію з сервісами для виконання поточної бізнес-логіки.

Сервіси — це частини програми які виконують основну логіку додатку. Вони інкапсулюють бізнес-логіку та маніпуляції над даними [30]. Сервіси зазвичай використовуються контролерами для виконання різних операцій, таких як запит до бази даних, обробка даних або взаємодія із зовнішніми API. Вони відповідають за функціональність програми, зберігаючи контролери простими та зосередженими на обробці вхідних запитів від клієнтської частини.

Наступним етапом почнемо безпосередню реалізацію краудфандінгової платформи, перш за все створимо ряд необхідних модулів, аби розділити нашу кодову базу на різні функціональні частини. У розділі 2 було описано модулі з яких має складатись серверна частина, відповідно до створеної у тому ж розділі схеми, створимо наступні модулі за допомогою Nest.js CLI:

1. TicketModule - який буде вміщувати в собі все що стосується кампаній
2. CompanyModule - модуль який буде вміщувати інформацію про організаторів зборів.
3. PaymentModule - відповідатиме за кодову базу платіжних засобів
4. UserModule - буде вміщувати все що стосується користувачів системи
5. AuthModule - відповідає за частину авторизації та реєстрації у системі

Відповідно до визначення що таке модуль та його призначення, можемо зробити висновок що кожен модуль в нашій програмі повинен як мінімум містити в собі контролер, та сервіс. Ще однією частиною яку буде містити наш модуль є entities, яка відповідно до чистої архітектури є ядром навколо якого будуються інші шари. Таким чином розглянемо один з модулів нашої системи та його складові, наприклад TicketModule. Перш за все розглянемо файлову структуру нашого модуля та зрозуміємо з яких компонентів він складається, яка зображена на рисунку 3.2.

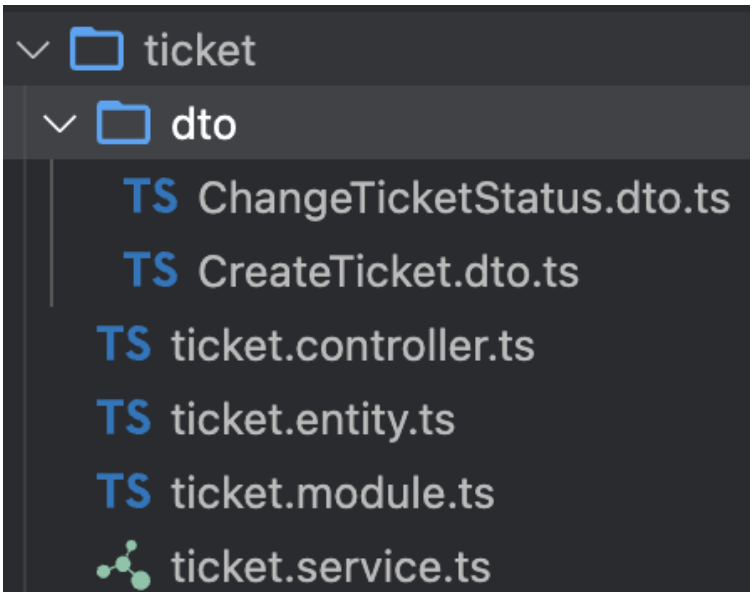


Рисунок 3.2 - Файлова структура TicketModule

З рисунку можемо побачити що модуль складається з беспосереднього файла модуля, контролера, сервіса, ентіті, та допоміжних dto файлів. Розглянемо кожен з файлів більш детально, почнемо з файла модуля який вміщає в собі декларацію окремих частин, а саме контролерів, сервісів та бібліотек, які використовуються в модулі.

З лістингу у додатку можемо побачити що у модулі використовуються TicketController, TicketService, та бібліотека TypeOrm для взаємодії з базою даних, яка приймає в себе масив entity нашої системи. Розглянемо entity який описується в даному модулі, а саме TicketEntity, який представляє собою сутність кампанії яка створюється для збору коштів, та вміщує в собі ряд полів які описують інформацію про компанію таких як до них такі як *@Column*, які описують параметри. Як результат можемо побачити клас з різним набором полів в таблиці у реляційній базі даних. Такоє можемо побачити додаткові декоратори *@ManyToMany* та *@ManyToOne,* які описують зв'язки з іншими таблицями, а саме таблицями PaymentMethod та Companies.

Наступним етапом розглянемо TicketService який вміщує в собі всю основну бізнес логіку яка пов'язана з кампаніями, а саме їх створення, видалення, оновлення, отримання всіх або якоїсь конкретної. Метод класу

TicketService відповідає за створення нової компанії для збору коштів та реалізує основну бізнес логіку цього процесу отримуючи набір вхідних даних з контролера та взаємодіючи з іншими частинами модуля для виконання заданого процесу.

Останнім файл який необхідно розглянути є ticket.controller.ts, який необхідний для створення REST інтерфейсу, аби дати можливість клієнтській частині викликати необхідні функції за допомогою http запитів. Всередині даного файлу зберігається клас, який описує вхідні маршрути та виклики функцій відповідно до маршрута. Розглянемо список маршрутів які надаються даним контролером. Кожен маршрут та його призначення наведені у таблиці 3.4.

Таблиця 3.5 – Маршрути які надає TicketController

Назва маршрута	Тип	Опис
/tickets/create	POST	Маршрут для створення нової компанії
/tickets/all	GET	Маршрут для отримання списку всіх кампаній
/tickets/get-count	GET	Маршрут для отримання кількості всіх кампаній
/tickets/:id	GET	Маршрут для отримання однієї кампанії
/tickets/company/:companyId	GET	Маршрут для отримання списку всіх кампаній за конкретним організатором
/tickets/:id/change-status	PATCH	Маршрут для зміни статусу кампанії
/tickets/:id	PATCH	Маршрут для редагування кампанії
/tickets/:id	DELETE	Маршрут для видалення кампанії

В результаті можемо побачити що модуль складається з контроллера який надає набір REST запитів, сервісу який виконує основну бізнес логіку модуля,

сутність яка є ядром модуля навколо якого будуються всі інші шари архітектури та створюється таблиця з налаштуваннями в базі даних.

Всі інші модулі нашої системи реалізуються аналогічним чином, аналогічним набором файлів, єдине що в них різниться це ключова функціональність яка закладена в рамках того чи іншого модуля, так AuthModule буде містити всю необхідну кодову базу яка відповідає за авторизацію та реєстрацію у системі, CompanyModule містить все що стосується організаторів збору, UserModule в якому описується бізнес логіка роботи користувачів системи, їх права доступа, конфіденційна інформація тощо.

3.5 Реалізація модуля автоматичного оцінювання проєкту

Оскільки однією з головних задач є реалізація механізму автоматичного надання оцінки проекту, необхідно реалізувати частину функціоналу яка б відповідала за взаємодію з механізмами обробки природної мови, а саме обраною нами бібліотекою Natural, з метою визначення тональності тексту. Оскільки даний функціонал дуже тісно пов'язаний з створенням кампанії по збору коштів, цілком логічним є реалізація функцій надання оцінки в рамках TicketModule.

Отож, реалізуємо базову функціональність механізмів які працюють з NLP та визначають тональність тексту, відповідно до алгоритму описаного в розділі 2.4. Для цього створимо функцію *analyze()*, яка примає аргументом певний текстовий набір даних для якого необхідно визначити його тональність. Для цьго виконаємо кроки які описані в алгоритмі, а саме застосуємо базові словники для розпізнавання скорочень, приведемо текст до єдиного регістру, вилучимо з тексту спецсимволи, проведемо токенізацію за допомогою WordTokenizer() та стеммінг і як результат визначимо тональність тексту.

Код функції *analyze()* яка визначає тональність тексту для заданого фрагменту наведено в додатку.

Розроблену функцію будемо використовувати при створені нової кампанії, інтегрувавши її виклик перед записом до бази даних.

3.6 Програмна реалізація клієнтської частини

Відповідно до обраного фреймворку для реалізації клієнтської частини, так само як і прireалізації серверної частини перш за все необхідно створити проект з використанням обраного фреймворку Angular. В такому випадку для того щоб це зробити необхідно виконати наступну команду *$ ng new crowdfinding-fe*. По аналогії з Nest.js ця команда створить базову файлову структуру та архітектуру додатку для клієнтської частини [31]. Яка зображена на рисунку 3.16.

Рисунок 3.3 - Файлова структура початкового Angular додатку

Наступним етапом є виділення основних частин з яких будується клієнтський додаток. Так як Nest.js побудований надихаючись прикладами з Angular ці два фреймворки не дивлячись на їх використання у різних предметних областях є дуже схожими у частині архітектури та структурних елементів, отож розглянемо які структурні елементи надає Angular.

Angular так само як і Nest.js будує власну архітектуру навколо модулів, які відіграють в Angular ключову роль під час написання додатку та складаються з інших частин системи таких як компоненти та сервіси.

Наступною ключовою частиною і структурною одиницею в Angular є компоненти які визначаються класом, який містить дані та логіку окремої частини додатку, і він пов’язаний із шаблоном HTML, який визначає представлення, що відображатиметься в цій частині [32]. Як результат весь додаток складається з різної кількості компонентів які поєднуються між собою.

В Angular також використовуються сервіси які виконують схожу роль порівнюючи їх з сервісами у Nest.js, а саме вміщують у собі основну бізнес логіку системи, а також дозволяє спільно використовувати дані між різними компонентами.

З розуміння ключових структурних елементів Angular реалізуємо клієнтську частину краудфандінгової платформи. Так само як і в серверній частині, в клієнтській можна виокремити модулі які будуть відповідати за різні частини функкціоналу, і безпосереднє поділення на модулі буде виглядати майже так само. Саме тому доведеться створити TicketModule, CompanyModule, AuthModule. Додатковим модулем який необхідно буде створити є SharedModule, який необхідний для того щоб організувати та зберігати спільні ресурси, компоненти, директиви, та інших функціональних блоків, які можуть бути використані в кількох інших модулях розроблювального додатку.

Розглянемо детально реалізацію TicketModule з його компонентами та сервісами. В рамках розробки даного модуля необхідно забезпечити сторінку всіх компаній, сторінку власно-створених кампаній, сторінку детальної інформації про одну кампанію та сторінку створення нової компанії. Як приклад

розглянемо як будується один з компонентів, а саме TicketComponent, який відповідає за перегляд детальної інформації про одну із кампаній. Перш за все переглянемо .ts файл компоненту у якому зберігається основна логіка його виконання. Клас описується декоратором @Component який містить в собі додаткову мета-дату, а саме інформацію про розташування темплейту та розмітки даного компоненту його стилів а також селектору який необхідний для того щоб використовувати даний компонент в інших структурних одиницях розроблювального додатку. Також варто зазначити те що компонент використовує один з 8 етапів життєвого циклу компоненту, а саме ngOnInit, який викликає тоді коли проходить ініціалізація компоненту. В рамках цієї ініціалізації ми можемо виконувати бізнес логіку компоненту. У випадку даного компоненту в обробнику етапу життєвого циклу викликається функція сервісу який відповідає за основну бізнес логіку роботи з компаніями в рамках даного модуля, а саме TicketService. Як ми можемо побачити викликається метод *getTicketById()*, який ініціює запит до нашого серверу за шляхом /tickets/:id, та отримує повну інформацію про кампанію за її id, після чого ця інформація використовується для відображення у темплейті даного компоненту. Темплейт описує html структуру компонента, та відображає необхідні дані які зберігаються в .ts файлі компоненту, так в нашому випадку можемо бачити як виконується відображення інформації про компанію яка зберігається у змінній *ticket*. Налаштування стилів зберігається у .scss файлі, та вміщує в собі налаштування стилів для відповідних тегів або класів розроблювального компоненту.

Список та опис всіх реалізованих сторінок наведено у таблиці 3.5.

Таблиця 3.6 – Список розроблених сторінок та їх опис

Назва сторінки	Url	Опис
Вхід	/auth/login	Сторінка для авторизації та входу в систему
Реєстрація	/auth/signup	Сторінка для створення облікового запису в системі
Список всіх кампанії	/main/orders/orders	Сторінка з списком всіх активних компаній
Список власних кампаній	/main/orders/my	Сторінка з списком власних кампаній
Профіль	/main/profile/:id	Сторінка профілю компанії
Редагування профілю	/main/profile/edit	Сторінка редагування профілю компанії
Створити кампанію	/main/orders/create	Сторінка яка дозволяє створювати нову кампанію для збору коштів
Отримати кампанію по id	/main/orders/:id	Сторінка як відображає детальну інформацію про компанію

Відповідно до того як було реалізовано компонент отримання інформації про кампанію для збору коштів, аналогічним чином відбувається реалізація інших компонентів, єдине що змінюється це основна бізнес логіка, відображення, та обєми данних які відправляються та отримуються з серверу.

3.7 Висновки розділу 3

Отже у даному розділі було розглянуто існуючі фреймворки для реалізації серверної та клієнтської частини, в результаті проведено аналіз їх переваг та недоліків, після чого обрані найбільш оптимальні. Також було проаналізовано існуючі бібліотеки які надають функціональність NLP, та обрано серед них ту, що найбільш відповідає нашим вимогам. Відповідно до обраної бібліотеки був реалізований алгоритм роботи модуля інтеграції з NLP який було розглянуто у розділі 2.4. Та розроблено клієнтські та серверні додатки які формують систему.

4 ТЕСТУВАННЯ РОЗРОБЛЕНОЇ СИСТЕМИ НА ВІДПОВІДНІСТЬ ПОСТАВЛЕНИМ ВИМОГАМ

4.1 Методи які використовуються для тестування

Тестування програмного забезпечення - це процес під час якого перевіряється відповідність заявлених до продукту вимог до реально реалізованої функціональності, що відбувається шляхом спостереження роботою системи в штучно створених ситуаціях[33]. Існує велика кількість різних підходів до тестування розробленого ПЗ, розглянемо кожен з них та їх особливості. Деталі підходів до тестування програмного забезпечення зображено в таблиці 4.1.

Таблиця 4.1 - Підходи до тестування програмного забезпечення

Підхід до тестування	Особливості
Метод білого ящика	Включає в себе тестування усіх можливих шляхів в коді, проводить оцінка внутрішньої структури програми.
Метод чорного ящика	Визначанє тестування ПЗ з точки зору користувача без уваги до внутрішньої реалізації.
Метод сірого ящика	Комбінує в собі методи білого на чорного ящика, що означає тестування з обмеженим розумінням внутрішньої структури ПЗ.
Методи тестування на рівнях	Полягає у тестуванні починаючи з функцій у коді до тестування на системному рівні.
Методи функціонального тестування	Тестування на відповідність програми специфікаціям та вимогам, які були визначені на початку розробки.

Як результат для тестування веб-системи краудфандінгової платформи оберемо метод чорного ящика та будемо тестувати систему з точки зору кінцевого користувача системи. Перш за все виокремимо функціональні блоки

системи які необхідно протестувати, для цього опишемо тест кейси, які необхідно виконати [34]. Таблиця тест-кейсів наведена у таблиці 9, та включає в себе назву модуля системи, функціонал, який тестується, необхідні кроки та очікуваний результат.

Таблиця 4.2 - Опис тест-кейсів які необхідно відтворити для тестування системи

Функціонал	Назва модуля	Кроки для виконання	Очікуваний результат
Реєстрація у системі	Модуль авторизації	1. Відкрити сторінку з реєстрацією за маршрутом /auth/signup 2. Заповнити реєстраційну форму 3. Відправити форму на сервер	Успішна реєстрація та перехід на сторінку входу у систему
Вхід у систему	Модуль авторизації	1. Відкрити сторінку з реєстрацією за маршрутом /auth/login 2. Заповнити форму для входу 3. Відправити форму на сервер	Успішний вхід перехід на головну сторінку у системі
Перегляд списку активних кампаній	Модуль кампаній	1. Відкрити головну сторінку за маршрутом /main/orders/orders 2.Дочекатись отримання списку кампаній або повідомлення про те що їх не існує	Відображення списку активних кампаній, або повідомлення про те що на данний момент кампаній не знайдено
Перегляд власних кампаній	Модуль кампаній	1. Відкрити головну сторінку за маршрутом /main/orders/my 2.Дочекатись отримання списку кампаній або повідомлення про те що їх не існує	Відображення списку створених кампаній, або повідомлення про те що на данний момент кампаній не знайдено

Продовження таблиці 4.2

Створити власну кампанію	Модуль кампаній	1. Відкрити головну сторінку за маршрутом /main/orders/my 2. Натиснути на кнопку “Створити” 3. Заповнити форму для створення кампанії 4. Натиснути кнопку створити	Відображення новоствореної кампанії у списку власних кампаній
Переглянути детальну інформацію про кампанію	Модуль кампаній	1. Відкрити головну сторінку або сторінку власних кампаній та обрати одну з них	Відкриття детальної інформації про кампанію
Переглянути профіль	Модуль інформації про користувача	1. Натиснути на назву компанії на сторінці детальної інформації про збір коштів	Відкриття сторінки з інформацією про компанію
Відредагувати профіль	Модуль інформації про користувача	1. Натиснути на вкладку “Профіль” у боковому меню 2. Відредагувати інформацію про профіль 3. Натиснути кнопку “Оновити”	Отримання повідомлення про успішне оновлення профілю

4.2 Перевірка тест кейсів модуля авторизації

Перший тест-кейс який необхідно виконати для того щоб протестувати модуль авторизації це реєстрація у системі для цього відкриємо сторінку реєстрації та заповнюємо форму, як зображено на рисунку 4.1.

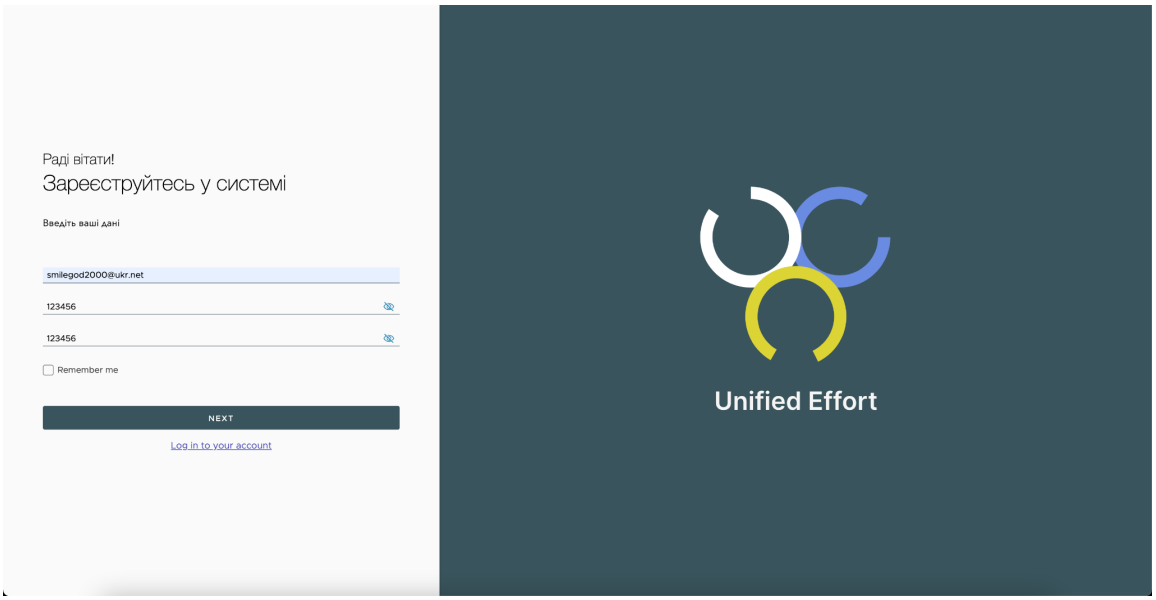


Рисунок 4.1 - Заповенна форма реєстрації у системі

Результат виконання на рисунку 4.2 з якого можемо зробити висновок що даний функціонал працює коректно.

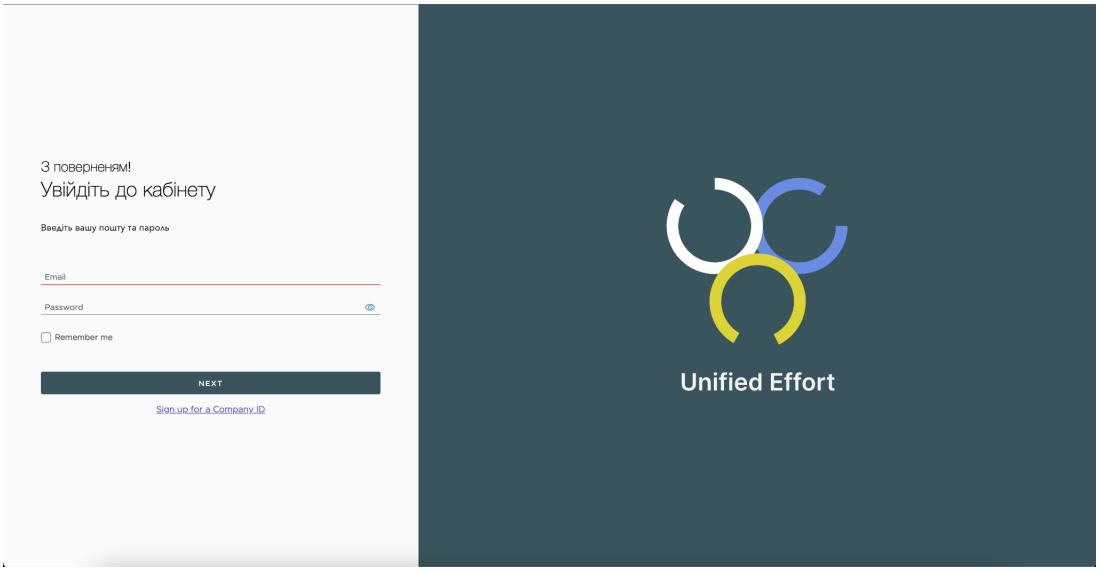


Рисунок 4.2 - Результат реєстрації у системі

Наступним етапом є автентифікація у системі новоствореним користувачем з тест-кейсу номер один, для цього використаємо email та пароль,

які використовувались під час реєстрації та натиснемо на кнопку “Вхід”, заповнена форма входу зображена на рисунку 4.3.

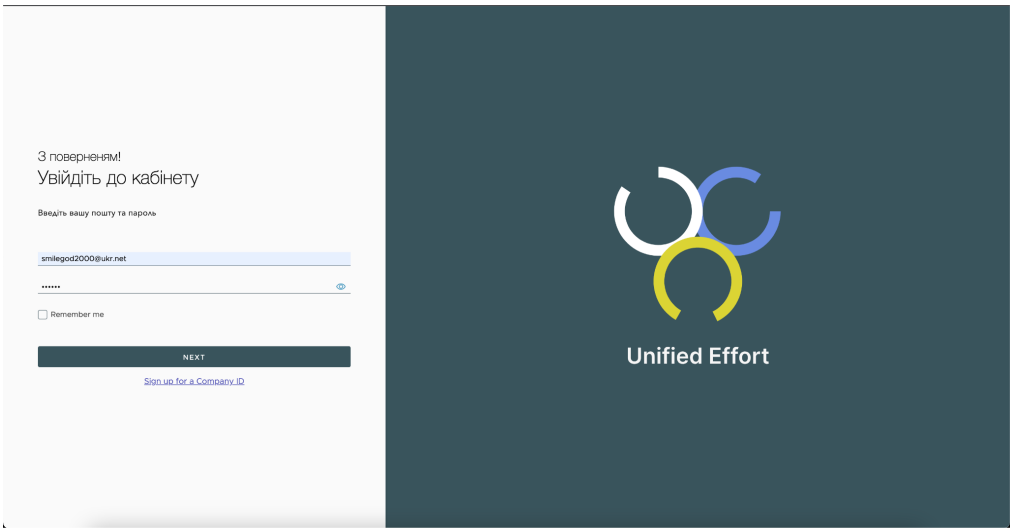


Рисунок 4.3 - Заповнена форма входу у системі

Результат виконання входу представлено на рис. 4.4.

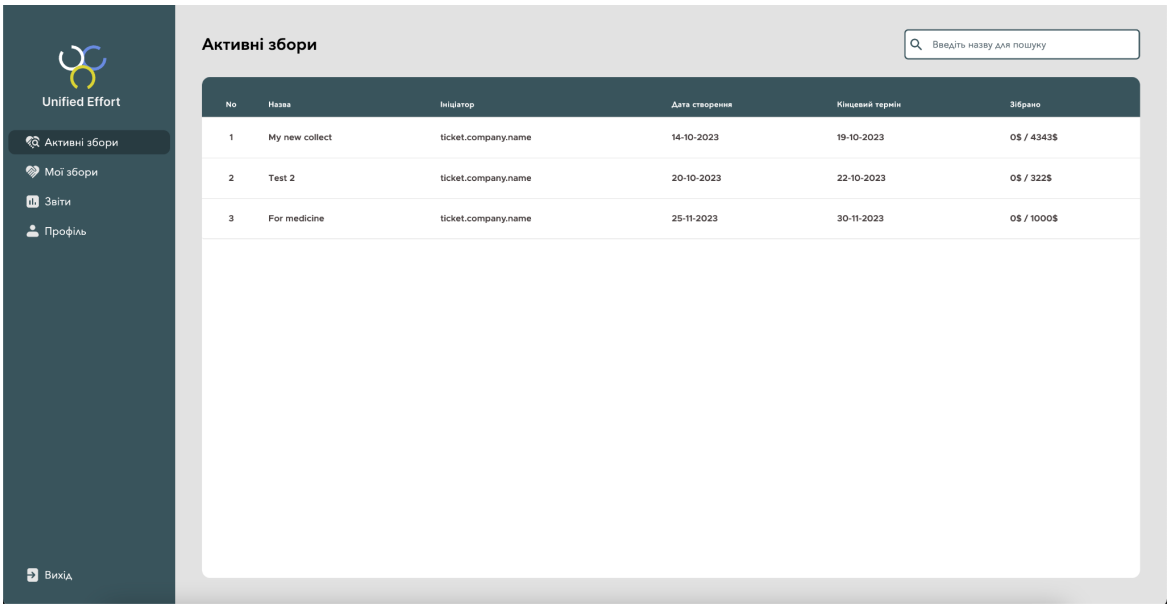


Рисунок 4.4 - Результат входу у систему

4.3 Перевірка тест кейсів модуля кампаній

Після входу у систему новим юзером, протестуємо правильність отримання списку активних кампаній, з рисунку 4.5 можемо бачити що даний функціонал працює коректно і ми отримуємо список активних компаній який відображається на сторінці, після чого можемо перейти на вкладку “Мої збори”, де відображаються власно створені збори, які описуються у тест-кейсі 4.

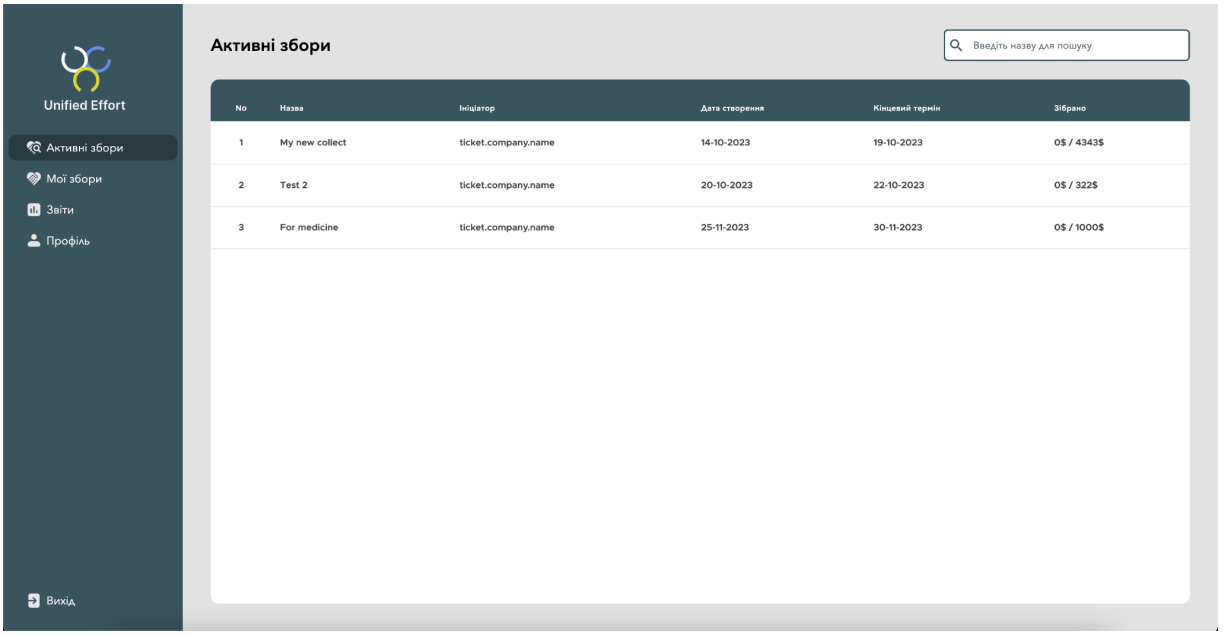


Рисунок 4.5 - Список активних зборів

З рисунку 4.6 можемо побачити що в нового користувача не існує створених кампаній, саме тому відображається повідомлення про те що кампаній не знайдено.

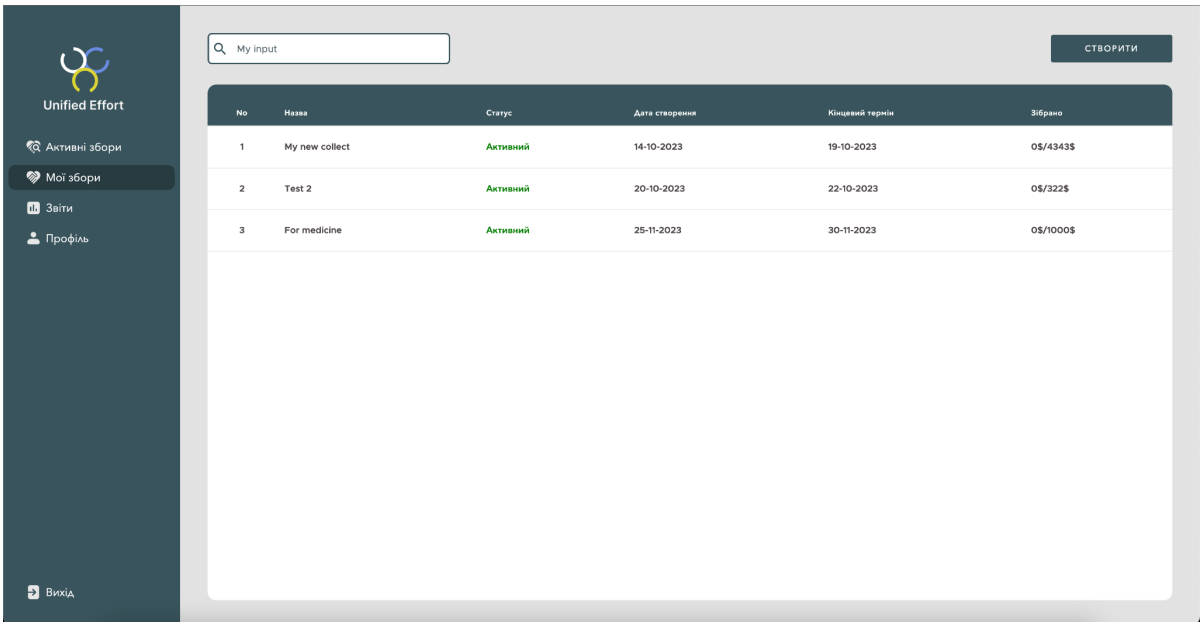


Рисунок 4.6 - Список власних зборів

Наступним етапом протестуємо тест-кейс 5, який відповідає за функціонал створення нової компанії, для цього натиснемо на кнопку “Створити” після чого на новій сторінці заповнемо форму яка необхідна для того щоб додати новий збір, заповнена форма зображена на рисунку 4.7.

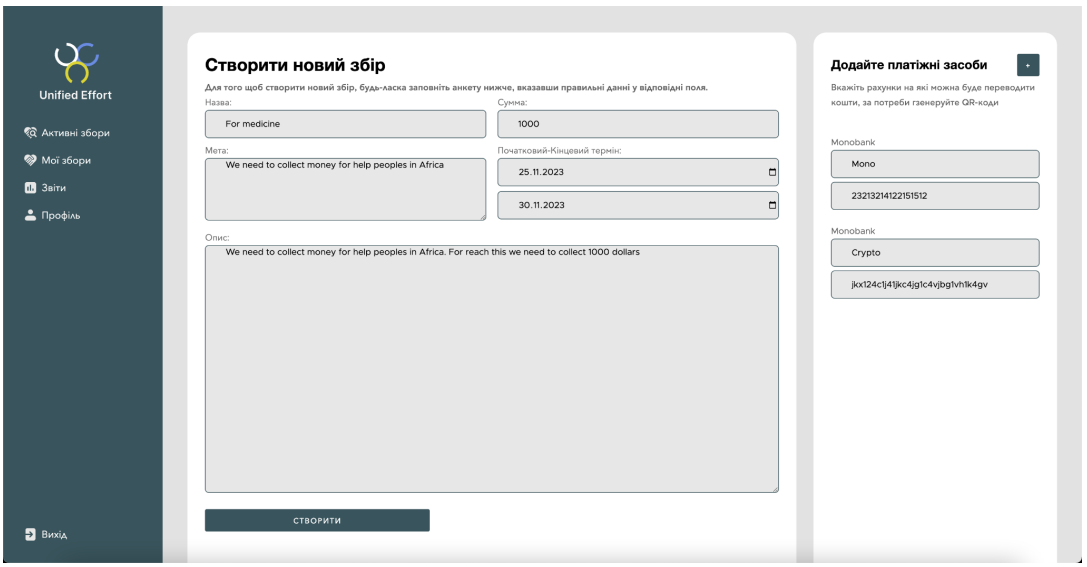


Рисунок 4.7 - Заповнення форми створення

Для того щоб перевірити чи створилась нова кампанія перейдемо на сторінку з власними зборами і перевіримо чи утворився там збір, як можемо побачити на рисунку 4.8 операція створення прошла успішно.

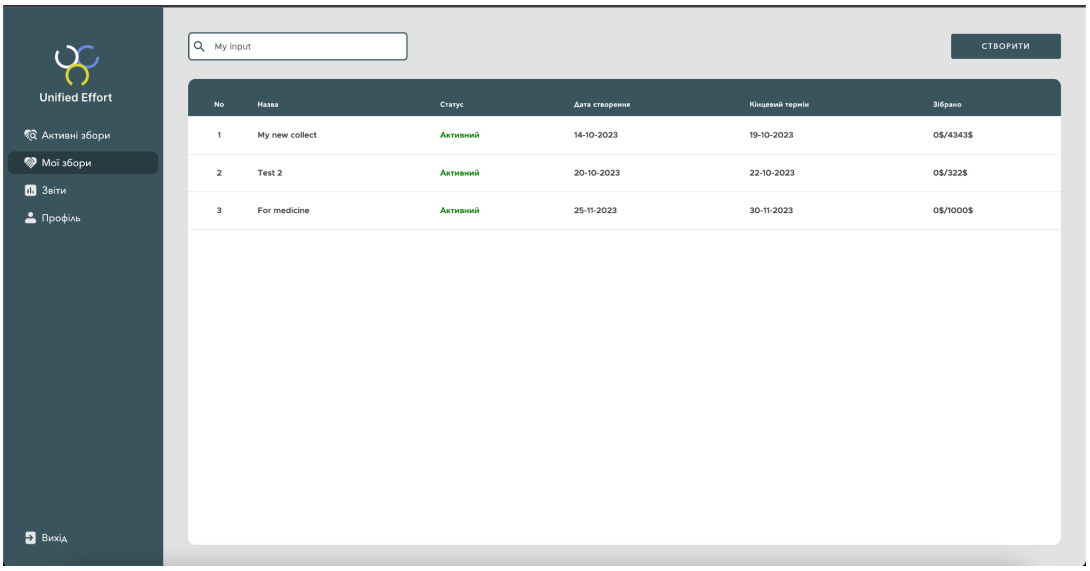


Рисунок 4.8 - Список власних зборів після створення

Перевіримо тест-кейс який відповідає за перегляд детальної інформації про кампанію, для цього натиснемо на нашу новостворену компанію, результат виконання даної операції зображений на рисунку 4.9.

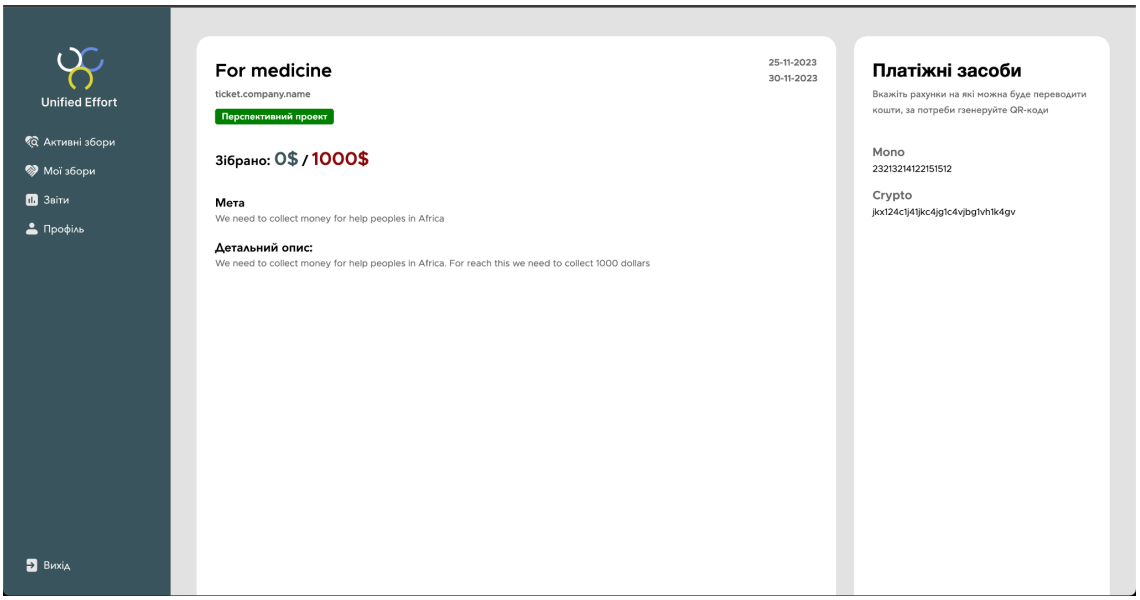


Рисунок 4.9 - Детальна інформація про кампанію

Як можемо побачити з рисунку, інформація яка вказувалась при створені збереглась та тепер доступна для перегляду іншим користувачам. Важливим моментом є перевірити наявність автоматизованої оцінки яку надала система даному проекту. Для цего перевіримо наявність інформації у полі оцінка системи, з рисунку можемо побачити наявність інформації у даному полі що може свідчити про правильність виконання даної функції. Останніми тест-кейсами які потрібно перевірити є перегляд інформацію про компанію яка організовує збір. Для того щоб переглянути таку інформацію необхідно натиснути на її назву в детальному описі збору. Після чого має відкритись нова сторінка на якій присутня інформація про компанію яка зображена на рисунку 4.12.

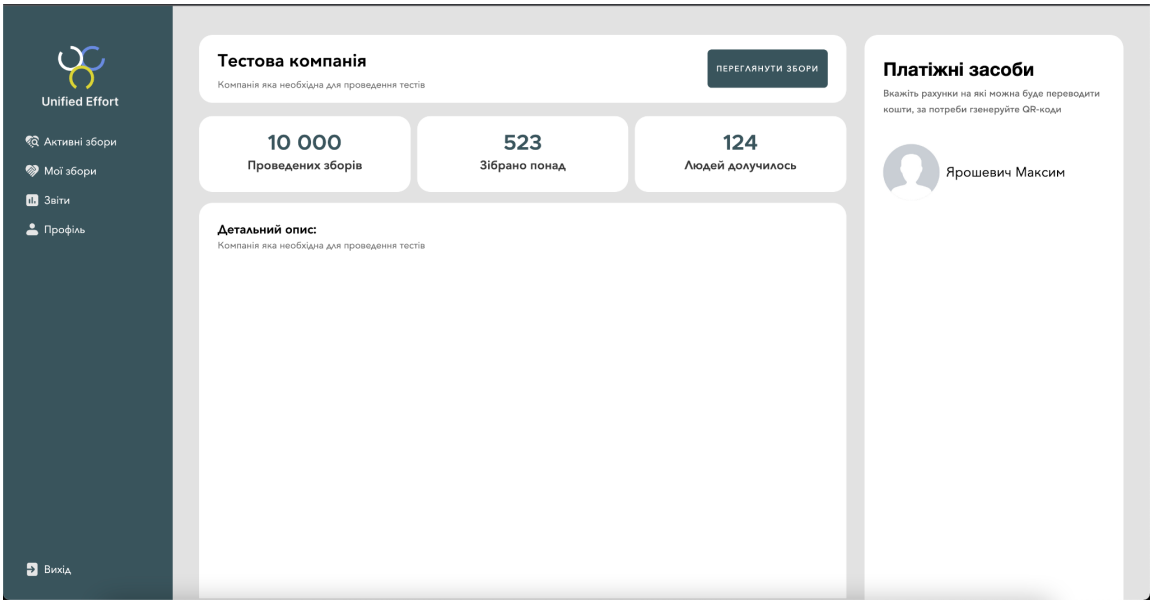


Рисунок 4.12 - Перегляд інформації про компанію організатора

В разі якщо необхідно відредагувати інформацію про власну компанію, необхідно перейти у “Профайл” в боковому меню та змінити значення у відповідних полях для вводу, наприклад змінимо опис компанії та збережемо ці зміни. Для того щоб перевірити чи збереглись зміни коректно виконаємо

перезапуск сторінки. Як можемо бачити на рисунку 4.14 інформація, яка вводиться під час редагування, збереглась.

Як результат можемо зробити висновок що всі тест-кейси які було описано на початку розділу було виконано, та перевірено на відповідність до очікуваного результат. Можна зробити висновок що розроблений функціонал відповідає поставленим вимогам, та реалізує функціонал краудфандінгової платформи.

4.4 Тестування модуля взаємодії з механізмами NLP

Після тестування системи методом чорної скриньки, варто приділити особливу увагу тестуванню механізму який взаємодіє з механізмами NLP, для тестування візьмемо два фрагменту тексту, один з позитивними намірами інший з негативними і надішлемо його у функцію яка визначає тональність тексту.

Результати перевірки фрагменту тексту з негативними намірами наведені на рисунку 4.13.

Фрагмент тексту: "Unfortunately, the service provided was terrible. The staff was rude and unprofessional, and the quality of the product was far below expectations. I will not be returning to this establishment."

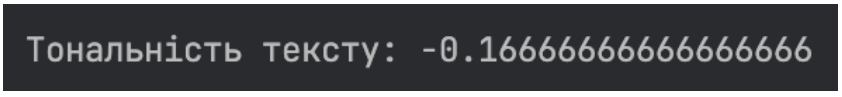


Рисунок 4.13 - Результати оцінки тексту з негативними намірами

Результати перевірки фрагменту тексту з позитивними намірами наведені на рисунку 4.14.

Рисунок 4.14 - Результати оцінки тексту з позитивними намірами

Фрагмент тексту: "The customer service at this establishment is outstanding! The staff is very friendly, attentive, and always willing to assist. Moreover, the quality of the products is exceptional. I highly recommend this place."

Як результат можемо побачити що при оцінюванні негативно забарвленого тексту отримали результат який меньше за 0, що підтверджує його негативну тональність, в той самий час позитивно забарвлений текст отримав результат вище 0, що свідчить про позитивну тональність. Як результат можемо зробити висновок що механізм працює правильно та на основі цих значень можна давати оцінку описам проектів.

4.5 Висновки розділу 4

В результаті у даному розділі було розглянуто методи які використовуються для тестування системи, обрано метод яким буде відбуватись перевірка розробленого додатку. Після чого був сформований список тест-кейсів, які необхідно перевірити задля того щоб зрозуміти чи працює розроблений додаток відповідно до поставлених вимог. Після перевірки можна зробити висновок всі тест-кейси, які було описано на початку розділу було виконано, та перевірено на відповідність до очікуваного результат. Як результат можна констатувати що розроблений функціонал відповідає поставленим вимогам, та реалізує функціонал краудфандінгової платформи.

5 ЕКОНОМІЧНА ЧАСТИНА

5.1 Оцінювання комерційного потенціалу розробки

Розроблювальний веб-додаток відноситься до тих науково-технічних робіт, які призначаються для виведення на ринок, тобто коли відбувається так звана комерціалізація науковотехнічної розробки. Для того аби отримати економічний ефект необхідно знайти потенційного інвестора, який би взявся за реалізацію цього проекту і переконати його в економічній доцільності такого кроку. Для цього випадку мають бути виконані наступні етапи робіт:

а) проведення комерційного аудиту науково-технічної розробки, тобто встановлення її науково-технічного рівня та комерційного потенціалу;

б) розрахунок витрат на здійснення науково-технічної розробки;

в) розрахунок економічної ефективності науково-технічної розробки у випадку її впровадження та комерціалізації потенційним інвестором і обґрунтування економічної доцільності комерціалізації потенційним інвестором розробленої у магістерській кваліфікаційній роботі науково-технічної розробки [35].

Метою проведення комерційного та технологічного аудиту є оцінювання комерційного потенціалу розробки метод і засобів веб-системи користувачів та їх комунікації для відео сервісу. Для проведення технологічного аудиту було залучено 3-х незалежних експертів Вінницького національного технічного університету: Мельник Денис Олександрович, Грабарчук Антон Володимирович, Капченко Карина Григорівна. Для проведення технологічного аудиту було використано таблицю 5.1 в якій за п’ятибальною шкалою використовуючи 12 критеріїв здійснено оцінку комерційного потенціалу.

Таблиця 5.1 – Рекомендовані критерії оцінювання комерційного потенціалу розробки та їх можлива бальна оцінка

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Критерії	0	1	2	3	4
Технічна здійсненність концепції:					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздатність продукту в реальних умовах
Ринкові переваги (недоліки):					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї

Продовження таблиці 5.1

9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військовопромисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Таблиця 5.2 – Рівні комерційного потенціалу розробки

Середньоарифметична сума балів СБ, розрахована на основі висновків експертів	Рівень комерційного потенціалу розробки
0-10	Низький

Продовження таблиці 5.2

11-20	Нижче середнього
21-30	Середній
31-40	Вище середнього
41-48	Високий

В таблиці 5.3 наведено результати оцінювання експертами комерційного потенціалу розробки.

Таблиця 5.3 – Результати оцінювання комерційного потенціалу розробки

Критерії	Прізвище, ініціали експерта		
	Мельник Д. О.	Грабарчук А. В.	Капченко К. Г.
	Бали, виставлені експертами:		
1	4	4	4
2	2	2	2
3	4	4	4
4	3	3	3
5	4	4	4
6	2	3	2
7	3	3	3
8	4	4	4
9	3	3	3
10	4	4	4
11	3	3	3
12	2	2	2
Сума балів	38	39	38
Середньоарифметична сума балів $\overline{СБ}$	$\overline{СБ} = \frac{\sum_{1}^{3} СБ_i}{3} = \frac{38+39+38}{3} = 38$		

Врезульаті отримуємо середньоарифметичну сума балів, яка була розрахована на основі висновків експертів і складає 38 балів, що відповідно до таблиці 5.2 вважається, вище середнім рівнем комерційного потенціалу.

Методи і програмі засоби веб-системи краудфандінгової платформи, що розробляються в рамках магістерської кваліфікаційної роботи будуть цікаві інвесторам як потенційно прибутковий стартап.

Порівняємо розробку, що розробляється в магістерській кваліфікаційній роботі з аналогом, який існує на ринку.

Аналогом з яким будемо порівнювати є краудфандінгова система StartEngine. Серед основних недоліків аналогу можна виокремити Складність у проходженні контролю та схвалення проектів перед їхнім запуском, відсутність механізмів оцінки проектів та технічні затримки.

У розробленій системі відповідні проблеми вирішується за допомогою функціоналу, який не обмежує користувачів в створенні власниї кампаній по збору коштів, реалізацією механізму який надає автоматизовано оцінку проекту під час його створення на основі аналізу текстової інформації, та використання ефективних інструменів для реалізації попередньо спроектованої архітектури яка є масштабованою.

Проведемо оцінку якості і конкурентоспроможності нової розробки порівняно з аналогом. В таблиці 5.4 наведено порівняння технічних параметрів аналога та науково-технічної розробки.

Параметр «Продуктивність» означає оптимізованість роботи системи. Коефіцієнт вагомості параметра становить 0.3.

Параметр «Автоматизація оцінки проекту» означає можливість системи надавати оцінку новостворених кампаніям автоматично. Коефіцієнт вагомості параметра становить 0.4.

Параметр «Простота запуску кампанії» означає відсутність обмежень для створення кампанії. Коефіцієнт вагомості параметра – 0.3.

Таблиця 5.4 – Основні параметри нової розробки та товару-конкурента

Показник	Варіанти		Відносний показник якості	Коефіцієнт вагомості параметра
	Базовий (товар-конкурент)	Новий (інноваційне рішення)		
Продуктивність	1	2	2	30%
Автоматизація оцінки кампанії	1	2	2	40%
Простота запуску кампанії	1	2	2	30%

Проведемо оцінку якості продукції, яка є найефективнішим засобом забезпечення вимог споживачів та порівняємо її з аналогом.

Визначимо відносні одиничні показники якості по кожному параметру за формулами (5.1) та (5.2) і занесемо їх у відповідну колонку табл. 5.5.

$$q_i = \frac{P_{\text{Н}i}}{P_{\text{Б}i}} \tag{5.1}$$

або

$$q_i = \frac{P_{\text{Б}i}}{P_{\text{Н}i}} \tag{5.2}$$

де $P_{\text{Н}i}$, $P_{\text{Б}i}$ – числові значення i-го параметру відповідно нового і базового виробів

$$q_1 = \frac{2}{1} = 2; q_2 = \frac{2}{1} = 2; q_3 = \frac{2}{1} = 2;$$

Відносний рівень якості нової розробки визначаємо за формулою:

$$K_{\text{Я.В.}} = \sum_{i=1}^{n} q_i \bullet \alpha_i, \tag{5.3}$$

$$K_{\text{Я.В.}} = 2{\cdot}0,3 + 2{\cdot}0,4 + 2{\cdot}0,3 = 2$$

Загальний показник конкурентоспроможності інноваційного рішення (К) з урахуванням вищезазначених груп показників можна визначити за формулою:

$$K = \frac{I_{m.n.}}{I_{e.n.}}, \tag{5.4}$$

де $I_{m.n.}$ – індекс технічних параметрів; $I_{e.n.}$ – індекс економічних параметрів.

Індекс технічних параметрів є відносним рівнем якості інноваційного рішення. Індекс економічних параметрів визначається за формулою (5.5)

$$I_{e.n.} = \frac{\sum_{i=1}^{n} P_{Hei}}{\sum_{i=1}^{n} P_{Бei}} \quad (5.5)$$

де P_{Hei}, $P_{Бei}$ – економічні параметри (ціна придбання та споживання товару) відповідно нового та базового товарів. В нашому випадку це ціна використання додатку.

$$I_{e.n.} = \frac{600}{500} = 1,2\ K = \frac{2}{1.2} = 1,6$$

Зважаючи на розрахунки, можна зробити висновок, що нова розробка буде більш конкурентоспроможною, ніж конкурентний товар.

5.2 Прогнозування витрат на виконання науково-дослідної роботи

Витрати, пов’язані з проведенням науково-дослідної роботи групуються за такими статтями: витрати на оплату праці, витрати на соціальні заходи, матеріали, паливо та енергія для науково-виробничих цілей, витрати на службові відрядження, програмне забезпечення для наукових робіт, витрати на роботи які виконують сторонні підприємства, установи і організації, інші витрати, накладні витрати.

5.2.1 Витрати на оплату праці

В даному випадку розуміються витрати на виплату основної та додаткової заробітної плати керівникам відділів, лабораторій, секторів і груп, науковим,

інженерно-технічним працівникам, конструкторам, технологам, креслярам, копіювальникам, лаборантам, робітникам, 15 студентам, аспірантам та іншим працівникам, безпосередньо зайнятим виконанням конкретної теми.

Основна заробітна плата кожного із дослідників $З_0$, якщо вони працюють в наукових установах бюджетної сфери визначається за формулою:

$$З_0 = \frac{М}{Т_р} \bullet t(грн), \quad (5.6)$$

де М – місячний посадовий оклад конкретного розробника (інженера, дослідника, науковця тощо), грн.;

$Т_Р$ – число робочих днів в місяці; приблизно $Т_р \approx 21...23$ дні;

t – число робочих днів роботи дослідника.

На базі данної формули та її показників побудуємо таблицю, та обрахуємо заробітну плату кожного з працівників. Обрахунки наведені в таблиці 5.5.

Таблиця 5.5 – Заробітна плата дослідника в науковій установі бюджетної сфери

Найменування посади	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Число днів роботи	Витрати на заробітну плату, грн
Керівник	22000	1000	40	40000
Програміст	20000	909	40	36360
Тестувальник	12000	545	40	21818
Всього				98178

Додаткова заробітна плата $З_Д$ всіх розробників та робітників, які приймали устають в розробці нового технічного рішення розраховується як 10 - 12 % від основної заробітної плати робітників.

В нашому випадку додаткова заробітна плата начисляється в розмірі 10% від основної заробітної плати.

$$З_Д = З_о \cdot \frac{Н_{дод}}{100\%}, \quad (5.7)$$

$$З_Д = 0,1 \cdot 98178 = 9817,8 \text{ (грн)},$$

5.2.2 Відрахування на соціальні заходи

Нарахування на заробітну плату НЗП дослідників та робітників, які брали участь у виконанні даного етапу роботи, розраховуються за формулою (5.8):

$$Н_{ЗП} = \left(З_о + З_Д\right) \bullet \frac{\beta}{100} \text{ (грн)}, \quad (5.8)$$

де $З_о$ – основна заробітна плата розробників, грн.;

$З_д$ – додаткова заробітна плата всіх розробників та робітників, грн.;

β – ставка єдиного внеску на загальнообов'язкове державне соціальне страхування, % .

Дана діяльність відноситься до бюджетної сфери, тому ставка єдиного внеску на загальнообов'язкове державне соціальне страхування буде складати 22%, тоді:

$$Н_{ЗП} = (98178 + 9817,8) \cdot \frac{22}{100} = 23759 \text{ (грн)}$$

5.3.3 Сировина та матеріали

Для розробки необхідно надати канцелярське приладдя та офісний папір для кожного з учасників команди, а також флешки, для зберігання інформації відповідно до проекту.

Таблиця 5.6 – Матеріали, що використані на розробку

Найменування матеріалу	Ціна за одиницю, грн	Витрачено	Вартість витраченого матеріалу, грн.
Папір	250	2	500
Канцелярське приладдя	1000	3	3000
Флешка	150	3	450
Всього			3950
З врахуванням коефіцієнта транспортування			4000

5.3.4 Розрахунок витрат на комплектуючі

В нашому випадку дана стаття витрат не використовувалась.

5.3.5 Спецустаткування для наукових (експериментальних) робіт

До цього сегменту належать витрати на виготовлення та придбання спецустаткування, верстатів, пристроїв, інструментів, приладів, стендів, апаратів, механізмів, іншого спецобладнання, необхідного для проведення досліджень, також витрати на їх проектування, виготовлення, транспортування, монтаж та встановлення.

Вартість спецустаткування визначається за прейскурантом гуртових цін або за даними базових підприємств за відпускними і договірними цінами. До балансової вартості устаткування окрім прейскурантної вартості входять витрати на його транспортування і монтаж, тому ці витрати беруться додатково в розмірі 10…12% від вартості устаткування.

Балансову вартість спецустаткування розраховують за формулою:

$$В_{спец} = \sum_{i}^{k} С_i \bullet Ц_i \bullet К_i, \quad (5.9)$$

де $С_i$ – кількість одиниць устаткування даного виду;

$Ц_i$ – вартість матеріалу і-го найменування, грн./кг.;

$К_i$ – коефіцієнт транспортних витрат, Кі = (1,1…1,15);

Таблиця 5.7 – Устаткування, що використане на розробку

Найменування матеріалу	Ціна за одиницю, грн	Витрачено	Вартість витраченого матеріалу, грн.
Ноутбук	40000	1	40000
Зарядний пристрій	1000	1	1000
Всього			41000
З врахуванням коефіцієнта транспортування			45100

5.3.5 Програмне забезпечення для проведення (експериментальних) робіт

Програмне забезпечення для наукової роботи включає в себе використання безкоштовних інструментів такі як Visual Studio Code, PgAdmin, Postman, Google Docs.

5.3.6 Амортизація обладнання, програмних засобів та приміщень

Сюди відносять амортизаційні відрахування по кожному виду обладнання, устаткування та інших приладів і пристроїв, а також програмного забезпечення для проведення науково-дослідної роботи, за його наявності в дослідній організації або на підприємстві. В спрощеному вигляді амортизаційні відрахування по кожному виду обладнання, приміщень та програмному

забезпеченню тощо можуть бути розраховані з використанням прямолінійного методу амортизації за формулою:

$$А = \frac{Ц \cdot Т}{Т_{кор} \cdot 12} (грн), \quad (5.10)$$

де Ц – балансова вартість даного виду обладнання (приміщень), грн.;

$Т_{кор}$ – час користування;

Т – термін використання обладнання (приміщень), цілі місяці.

Згідно пункту 137.3.3 Податкового кодексу амортизація нараховується на основні засоби вартістю понад 20000 грн. В нашому випадку для написання магістерської роботи використовувався ноутбук вартістю 40000 грн., а саме ноутбук msi gf63.

$$А = \frac{40000 \cdot 2}{2 \cdot 12} = 3333,3 (грн)$$

5.3.7 Паливо та енергія для науково-виробничих цілей

До статті «Паливо та енергія для науково-виробничих цілей» відносяться витрати на всі види палива й енергії, що безпосередньо використовуються з технологічною метою на проведення досліджень.

$$В_е = \sum_{i=1}^{n} \frac{W_{yt} \bullet t_i \bullet Ц_е \bullet К_{впі}}{\eta_i} (грн), \quad (5.11)$$

де W_{yt} – встановлена потужність обладнання на певному етапі розробки, кВт;

t_i – тривалість роботи обладнання на етапі дослідження, год;

$Ц_е$ – вартість 1 кВт-години електроенергії, грн;

$К_{впі}$ – коефіцієнт, що враховує використання потужності, $К_{впі} < 1$;

η_i – коефіцієнт корисної дії обладнання, $\eta_i < 1$.

Для написання магістерської роботи використовується ноутбук для якого розрахуємо витрати на електроенергію.

$$В_е = \frac{0,05 \cdot 528 \cdot 7.5 \cdot 0,05}{0,8} = 12,38(грн)$$

5.3.8 Службові відрядження

Витрати на службові відрядження, витрати на роботи, які виконують сторонні підприємства, установи, організації та інші витрати в нашому досліджені не враховуються оскільки їх не було.

5.3.9 Накладні (загальновиробничі) витрати

Накладні (загальновиробничі) витрати Внзв охоплюють: витрати на управління організацією, оплата службових відряджень, витрати на утримання, ремонт та експлуатацію основних засобів, витрати на опалення, освітлення, водопостачання, охорону праці тощо. Накладні (загальновиробничі) витрати $В_{нзв}$ можна прийняти як (100…150)% від суми основної заробітної плати розробників та робітників, які виконували дану МКНР, тобто:

$$В_{нзв} = (З_О + З_Р) \cdot \frac{Н_{нзв}}{100\%} (грн), \qquad (5.12)$$

де $Н_{нзв}$ – норма нарахування за статтею «Інші витрати»

$$В_{нзв} = 98178 \cdot \frac{100}{100\%} = 98178 \ (грн)$$

Сума всіх попередніх статей витрат дає витрати, які безпосередньо стосуються даного розділу МКНР:

$$В = 98178 + 9817 + 23759 + 45100 + 4000 + 3333,3 + 12.38 + 98178 = 282377.$$ (грн)

Прогнозування загальних втрат ЗВ на виконання та впровадження результатів виконаної МКНР здійснюється за формулою:

$$ЗВ = \frac{В}{\eta} (грн), \quad (5.13)$$

де η – коефіцієнт, який характеризує стадію виконання даної НДР.

Оскільки, робота знаходиться на стадії науково-дослідних робіт, то коефіцієнт β = 0,9.

Звідси:

$$ЗВ = \frac{282377.68}{0,9} = 313752.9 (грн)$$

5.4 Розрахунок економічної ефективності науково-технічної розробки за її можливої комерціалізації потенційним інвестором

У даному підрозділі кількісно спрогнозуємо, яку вигоду, зиск можна отримати у майбутньому від впровадження результатів виконаної наукової роботи. Розрахуємо збільшення чистого прибутку підприємства ΔПі, для кожного із років, протягом яких очікується отримання позитивних результатів від впровадження розробки, за формулою

$$\Delta П_i = \sum_{i}^{n} (\Delta Ц_0 \bullet N + Ц_o \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{v}{100}) (грн), \quad (5.14)$$

де $\Delta Ц_0$ – покращення основного оціночного показника від впровадження результатів розробки у даному році.

N – основний кількісний показник, який визначає діяльність підприємства у даному році до впровадження результатів наукової розробки;

ΔN – покращення основного кількісного показника діяльності підприємства від впровадження результатів розробки:

$Ц_о$ – основний оціночний показник, який визначає діяльність підприємства у даному році після впровадження результатів наукової розробки;

n – кількість років, протягом яких очікується отримання позитивних результатів від впровадження розробки:

λ – коефіцієнт, який враховує сплату податку на додану вартість. Ставка податку на додану вартість дорівнює 20%, а коефіцієнт λ = 0,8333.

р – коефіцієнт, який враховує рентабельність продукту. р = 0,25;

х – ставка податку на прибуток. У 2023 році – 18%.

Припустимо, що дохід від одного користувача зросте на 100 грн, порівняно з початковою ціною від одного користувача у аналога в розмірі 500 грн за використання додатку. Кількість одиниць реалізованої продукції (в даному випадку кількість користувачів) також збільшиться, порівняно з аналогом яка до реалізації розробки дорівнювала 5000: протягом першого року на 15 000, протягом другого року – на 5 000, протягом третього року на 5 000. Розрахуємо прибуток, яке отримає підприємство протягом трьох років.

$$\Delta П_1 = [100 \cdot 5000 + (500 + 100) \cdot 15000] \cdot 0,833 \cdot 0,25 \cdot (1 + \frac{18}{100}) = 2334482.5 (грн)$$

$$\Delta П_2 = [100 \cdot 5000 + (500 + 100) \cdot 20000] \cdot 0,833 \cdot 0,25 \cdot \left(1 + \frac{18}{100}\right) = 3071687.5 (грн)$$

$$\Delta П_3 = [100 \cdot 5000 + (500 + 100) \cdot 25000] \cdot 0,833 \cdot 0,25 \cdot (1 + \frac{18}{100}) = 3808892.5 (грн)$$

Таким чином, розрахунки показують, що відповідно прогнозуванню комерційний ефект від впровадження розробки виражається у значному збільшенні чистого прибутку підприємства.

5.4 Розрахунок ефективності вкладених інвестицій та періоду їх окупності

Розрахуємо основні показники, які визначають доцільність фінансування наукової розробки певним інвестором, є абсолютна і відносна ефективність вкладених інвестицій та термін їх окупності.

Розрахуємо величину початкових інвестицій PV, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки.

$$PV = k_{\text{інв}} \cdot \text{ЗВ}\,, \tag{5.15}$$

$k_{\text{інв}}$ – коефіцієнт, що враховує витрати інвестора на впровадження науковотехнічної розробки та її комерціалізацію. Це можуть бути витрати на підготовку приміщень, розробку технологій, навчання персоналу, маркетингові заходи тощо ($k_{\text{інв}}$= 2…5).

$$PV = 2 \cdot 313752.9 = 627505.8\ (\text{грн})$$

Розрахуємо абсолютну ефективність вкладених інвестицій Еабс згідно наступної формули:

$$\text{Е}_{\text{абс}} = (\text{ПП} - PV)\,, \tag{5.16}$$

де ПП – приведена вартість всіх чистих прибутків, що їх отримає підприємство від реалізації результатів наукової розробки, грн.;

$$\text{ПП} = \sum_{\text{i}}^{\text{T}} \frac{\Delta\text{П}_{\text{i}}}{(1+\tau)^{t}}\,, \tag{5.17}$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої НДЦКР, грн.;

Т – період часу, протягом якою виявляються результати впровадженої НДДКР, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні; для України цей показник знаходиться на рівні 0,2;

t – період часу (в роках).

$$ПП = \frac{2334482.5}{(1+0,2)^1} + \frac{3071687.5}{(1+0,2)^2} + \frac{3808892.5}{(1+0,2)^3} = 6282738,5 \text{ (грн)}$$

$$Е_{абс} = (6282738,5 - 627505.8) = 5655232.7 \text{ (грн)}$$

Оскільки $Е_{абс} > 0$ то вкладання коштів на виконання та впровадження результатів НДДКР може бути доцільним.

Розрахуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій . Для цього користуються формулою:

$$Е_в = \sqrt[Т_ж]{1 + \frac{Е_{абс}}{PV}} - 1, \quad (5.18)$$

де $Т_ж$ – життєвий цикл наукової розробки, роки.

$$Е_в = \sqrt[3]{1 + \frac{6282738,5}{627505.8}} - 1 = 1.22 \text{ або } 122\%$$

Визначимо мінімальну ставку дисконтування, яка у загальному вигляді визначається за формулою:

$$\tau = d + f, \quad (5.19)$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2023 році в Україні d= (0,14…0,2);

f – показник, що характеризує ризикованість вкладень; зазвичай, величина f = (0,05...0,1).

$$\tau_{min} = 0,18 + 0,05 = 0,23$$

Так як $E_в > \tau_{min}$ то інвестор може бути зацікавлений у фінансуванні даної наукової розробки.

Розрахуємо термін окупності вкладених у реалізацію наукового проекту інвестицій за формулою:

$$T_{OK} = \frac{1}{E_в}, \quad (5.20)$$

$$T_{ок} = \frac{1}{1,22} = 0,81 \text{ роки}$$

Так як $T_{ок} \leq 3...5$ -ти років, то фінансування даної наукової розробки є доцільним.

5.5 Висновки розділу 5

Як результат, в даному розділі проаналізовано економічну частину науково-дослідної роботи. Проведено комерційний та технологічний аудит НДР. В результаті якого трьома незалежними експертами сформовано таблицю оцінювання (таблиця 5.1) комерційного потенціалу. За результатами дослідження з’ясувалося, що розробка має рівень комерційного потенціалу - вище середнього і науково-технічного рівня. Такий рівень показника досягнуто в результаті впровадження методів обробки природньої мови для надання

автоматизованої оцінки новим проектам, що беруть за основу розпізнавання тексту та визначення його тональності. Було виконано прогнозування витрат на виконання НДР. На основі показників статей витрат розраховано загальні витрати на завершення НДР, які становлять 313752.9 грн. Далі розраховано показник ефективності вкладених інвестицій. Величина цього показника складає 122 %, що більше за 24 %, тобто мінімальний поріг, що визначає потенційну можливість інвестування. Також розраховано період окупності, який дорівнює 0,81 року. Такий термін свідчить про комерційну привабливість проєкту.

ВИСНОВКИ

В результаті написання магістерської роботи було досліджено методи та засоби удосконалення краудфандінгових платформ, розроблено веб-систему краудфандінгової платформи з інтегрованими механізмами NLP, які забезпечать автоматизоване надання оцінки краудфандінговим проєктам, підвищать рівень ефективності прийняття обгрунтованого рішення щодо інвестування проєкту.

В роботі проаналізовано та доведено актуальність розробки програмного забезпечення, визначено мету, предмет, об'єкт та задачі дослідження, описано наукову новизну.

Було проведено аналіз поняття краудфандінгу його основних видів та можливостей. Також було розглянуто поняття краудфандінгових платформ та проаналізовано переваги та недоліки найбільш популярних краудфандінгових систем та визначено яким чином можна розширити їх функціональність у розроблюваній системі для забезпечення потреб цільової аудиторії.

Розроблено метод створення краудфандінгової платформи з інтеграцією штучного інтелекту, складовими якого є правила, принципи, моделі прийняття рішень, алгоритми роботи клієнтської, серверної частини додатку.

В магістерській кваліфікаційній роботі використано теорію множин для формування математичної моделі прийняття рішення щодо інвестування проектів. Також розглянуто ключові методи обробки природної мови які були використані для розробки алгоритму визначення тональності опису проекту відповідно до якої формується оцінка проекту, розроблено архітектуру системи на різних рівнях.

Проаналізовано існуючі фреймворки та обґрунтовано їх вибір для розробки клієнтської, серверної частин та технології які надають функціональність обробки природної мови. Представлено реалізацію основних модулів клієнтської та серверної частин.

Реалізовуючи систему було використано мову програмування JavaScript та фреймворк Angular для реалізації клієнтської частини, а фреймворк Nest.js для

реалізації серверної, в якості бази даних було використано Postgresql, а в якості бібліотеки яка виконує функції обробки природної мови - бібліотека Natural.

Для перевірки роботи платформи було сформовано набір тест-кейсів для тестування розробленого програмного забезпечення на відповідність поставленим вимогам. Виконані процедури тестування підтвердили роботу функціоналу системи за моделлю прийняття рішень щодо інвестицій в проєкт.

Результати досліджень можуть бути використані для розробки аналогічних систем оцінювання та підготовки даних для прийняття управлінських рішень.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Що таке кравдфандинг? URL: https://buduysvoe.com/publications/kraudfandyng-shcho-ce-take-i-yak-vin-pracyuye-v-ukrayini (дата звернення: 11.10.2023).
2. Ведерніков М. Д., Базалійська Н. П., Волянська-Савчук Л. В., Зелена М. І., Кошонько О. В., Чернушкіна О. О. Фінансування стартапів із використанням краудфандингових платформ в Україні та світі. *Вісник Хмельницького національного університету.* 2021. № 6. С. 143-155
3. Версаль Н., Дудник Я. Краудфандинг як альтернативна Fintech-екосистема на фінансовому ринкую *Економіка. Київ.* 2021. 4(217). С. 26-37. URL https://doi.org/10.17721/1728-2667.2021/217-4/2
4. Воробей В., Кобринович М., Кривецька Л. Характеристики різних краудфандингових моделей. *Український культурний фонд.* Львів. 2020. С. 1-24.
5. Гвоздь М.Я., Бондаренко Ю.Г., Кулиняк І.Я. Краудфандинг як інструмент залучення коштів для фінансування стартап-проєктів: аналіз зарубіжного досвіду та вітчизняного досвіду. Економіка та управління підприємства.. Львів. 2020. Вип. 43. С. 131-136.
6. Ярошевич М.С., Коваленко О. О. Розробка архітектури краудфандінгової системи з використанням nlp для аналізу текстових даних. Матер. Міжнародної науково-технічної конференції “Сучасні тенденції розвитку техніки та технологій”.Міжнародний центр технологічних інновацій (Харків, 31 жовтня 2023 р). Research Europe, 2023. С. 13
7. Ярошевич М.С., Коваленко О. О. Удосконалення методів оцінювання та ранжування проєктів за допомогою інтелектуальних алгоритмів Матер. Міжнародної науково-практичної Інтернет конференції “Електронні інформаційні ресурси: створення, використання, доступ” 20-21 листопада

2023 р. – Суми/Вінниця: НІКО/ КЗВО «Вінницька академія безперервної освіти». 2023. – С. 323

8. SPA vs MPA Applications: What Are the Differences? URL: https://medium.com/@theadkgroup/spa-vs-mpa-applications-what-are-the-differences-7dc004e62397 (дата звернення: 13.10.2023).
9. JWT Authorization: How It Works and Implementing in Your Application. URL: https://frontegg.com/guides/jwt-authorization (дата звернення: 14.10.2023).
10. Пояснення алгоритмів обробки природної мови (NLP). URL - https://techukraine.net/%D0%BF%D0%BE%D1%8F%D1%81%D0%BD%D0%B5%D0%BD%D0%BD%D1%8F-%D0%B0%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC%D1%96%D0%B2-%D0%BE%D0%B1%D1%80%D0%BE%D0%B1%D0%BA%D0%B8-%D0%BF%D1%80%D0%B8%D1%80%D0%BE%D0%B4/ (дата звернення: 15.10.2023).
11. What is natural language processing? URL: https://www.techtarget.com/searchenterpriseai/definition/natural-language-processing-NLP#:~:text=How%20does%20natural%20language%20processing,way%20a%20computer%20can%20understand. (дата звернення: 15.10.2023).
12. Clean Code: A Handbook of Agile Software Craftsmanship. URL: https://www.oreilly.com/library/view/clean-code-a/9780136083238/ (дата звернення: 17.10.2023).
13. UML для бізнес-моделювання: для чого потрібні діаграми процесів. URL: https://evergreens.com.ua/ua/articles/uml-diagrams.html (дата звернення: 20.10.2023).
14. Tutorial: Intro to React. URL: https://legacy.reactjs.org/tutorial/tutorial.html (дата звернення: 25.10.2023).
15. What is Vue? URL: https://vuejs.org/guide/introduction.html#what-is-vue (дата звернення: 25.10.2023).
16. What is Angular? URL: https://angular.io/guide/what-is-angular (дата звернення: 25.10.2023).

17. Express.js vs Nest.js (All you need to know). URL: https://medium.com/@karahanozen/express-js-vs-nest-js-2e39fc0ce22c (дата звернення: 27.10.2023).
18. What is Nest.js? URL: https://nestjs.com/ (дата звернення: 27.10.2023).
19. The SOLID Principles of Object-Oriented Programming Explained. URL: https://www.freecodecamp.org/news/solid-principles-explained-in-plain-english/ (дата звернення: 29.10.2023).
20. Реляційна модель бази даних. URL: https://core.ac.uk/download/60829388.pdf (дата звернення: 03.11.2023).
21. Banker K. MongoDB in action / K. Banker. – Manning, NY, 2012. – Pp. 288.
22. Banker K. MongoDB in action / K. Banker. – NY : Manning, 2012. – 288 p.
23. What is mysql? URL: https://www.mysql.com/ (дата звернення: 04.11.2023).
24. What is MSSQL? URL: https://medium.com/@toprak.mhmt/what-is-mssql-9a152d7d4ed0 (дата звернення: 04.11.2023).
25. What is PostgreSQL? URL: https://www.postgresqltutorial.com/postgresql-getting-started/what-is-postgresql/ (дата звернення: 04.11.2023).
26. Natural Language Processing in Node.js: Top Libraries and Tools. URL: https://ai.plainenglish.io/natural-language-processing-in-node-js-top-libraries-and-tools-4702a46b3575 (дата звернення: 08.11.2023).
27. Best Way to Structure Your Directory/Code (NestJS). URL: https://medium.com/the-crowdlinker-chronicle/best-way-to-structure-your-directory-code-nestjs-a06c7a641401 (дата звернення: 08.11.2023).
28. Modules in NestJs. URL: https://docs.nestjs.com/modules (дата звернення: 08.11.2023).
29. Controllers in NestJs. URL: https://docs.nestjs.com/controllers (дата звернення: 08.11.2023).
30. Services in NestJs. URL: https://docs.nestjs.com/providers (дата звернення: 08.11.2023).

31. Angular File Structure and Best Practices (that help to scale). URL: https://medium.com/@shijin_nath/angular-right-file-structure-and-best-practices-that-help-to-scale-2020-52ce8d967df5 (дата звернення: 11.11.2023).
32. Angular components overview. URL: https://angular.io/guide/component-overview (дата звернення: 11.11.2023).
33. Що таке тестування програмного забезпечення? URL: https://qalight.ua/baza-znaniy/shho-take-testuvannya-programnogo-zabezpechennya/ (дата звернення: 15.11.2023).
34. Що таке test-case? URL: https://qalight.ua/baza-znaniy/test-case-2/ (дата звернення: 15.11.2023).
35. В. О. Козловський, О. Й. Лесько, В. В. Кавецький. *Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт*. Вінниця, Україна : ВНТУ, 2021. – 42 с.
36. Методичні вказівки до виконання магістерської кваліфікаційної роботи для студентів спеціальності 121 «Інженерія програмного забезпечення» / уклад. : О. Н. Романюк, Г. О. Черноволик. Вінниця : ВНТУ, 2022. – 50с.

ДОДАТОК А

(Обов'язковий)

Міністерство освіти і науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ

д.т.н., проф. О. Н. Романюк

"18" вересня 2023 р.

Технічне завдання

на магістерську кваліфікаційну роботу «Розробка веб-системи краудфандінгової платформи з інтеграцією штучного інтелекту» за спеціальністю

121 – Інженерія програмного забезпечення

Керівник магістерської кваліфікаційної роботи:

_____ к.т.н., Коваленко О. О.

"18" 09 2023 р.

Виконав:

_____студент гр.1ПІ-22м Ярошевич М.С.

"18" 09 2023 р.

Вінниця – 2023 року

1. Найменування та галузь застосування

Магістерська кваліфікаційна робота: «Розробка веб-системи краудфандінгової платформи з інтеграцією штучного інтелекту».

Галузь застосування - краудфандінг.

2. Підстава для розробки.

Підставою для виконання магістерської кваліфікаційної роботи (МКР) є індивідуальне завдання на МКР та наказ вищого навчального закладу від « 18 » вересня 2023 р. № 247 ректора по ВНТУ про закріплення тем МКР.

3. Мета та призначення розробки.

Метою роботи є створення та запровадження методу розробки автоматизованої краудфандінгової платформи, що дозволить запровадити автоматизоване оцінювання рівня привабливості краудфандінгових проектів за рахунок використання алгоритмів обробки природної мови.

Призначення роботи – розробка системи у вигляді модернізованої краудфандінгової системи з базовим функціоналом, яку можна використовувати для створення краудфандінгових кампаній. Створений прототип може бути основою для подальшого розвитку та розширення функціональності системи. В роботі пропонується система з інтегрованими механізмами обробки природної мови, правильним проектуванням системи, яке передбачає простоту використання та масштабованість за рахунок правильно побудованої архітектури.

4 Вихідні дані для проведення НДР

Перелік основних літературних джерел, на основі яких буде виконуватись МКР.

1. Що таке кравдфандинг? URL: https://buduysvoe.com/publications/kraudfandyng-shcho-ce-take-i-yak-vin-pracyuye-v-ukrayini (дата звернення: 11.10.2023).
2. Ведерніков М. Д., Базалійська Н. П., Волянська-Савчук Л. В., Зелена М. І., Кошонько О. В., Чернушкіна О. О. Фінансування стартапів із використанням краудфандингових платформ в Україні та світі. Вісник Хмельницького національного університету. 2021. № 6. С. 143-155
3. Версаль Н., Дудник Я. Краудфандинг як альтернативна Fintech-екосистема на фінансовому ринкую Економіка. Київ. 2021. 4(217). С. 26-37. URL https://doi.org/10.17721/1728-2667.2021/217-4/2
4. Воробей В., Кобринович М., Кривецька Л. Характеристики різних краудфандингових моделей. Український культурний фонд. Львів. 2020. С. 1-24.
5. Clean Code: A Handbook of Agile Software Craftsmanship.

5. Технічні вимоги

Angular framework, Postgresql, Nest.js, HTML, CSS, JavaScript, TypeScript, SCSS; Потужність множини користувачів - не менше 3; мінімальна кількість записів в базі данних - не менше 15, база даних - реляціна, мова програмування - веб-орієнтовна.

6. Конструктивні вимоги.

Розроблена структура системи повинна забезпечувати якість, надійність роботи та легку масштабованість, з огляду на те що в майбутньому система може розширюватись.

7. Перелік технічної документації, що пред'являється по закінченню робіт:

- пояснювальна записка до МКР;
- технічне завдання;
- лістинги програми.

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

9. Стадії та етапи розробки:

№ з/п	Назва етапів магістерської кваліфікаційної Роботи	Строк виконання етапів роботи
1	Аналіз поняття кравдфандінгу, та кравдфандінгових систем	20.09.2023 - 27.09.2023
2	Розробка методу створення кравдфандінгової платформи, проектування архітектури системи	28.09.2023 - 10.10.2023
3	Програмна реалізація системи за допомогою Angular, Nest.js, Natural	11.10.2023 - 27.10.2023
4	Розробка тест-кейсів та тестування програмного забезпечення	28.10.2023 - 15.11.2023
5	Виконання економічної частини	16.11.2023 - 21.11.2023
6	Оформлення пояснювальної записки і графічного матеріалу	22.11.2023 - 27.11.2023
7	Підготовка до захисту	28.11.2023 - 05.12.2023

10. Порядок контролю та прийняття.

Виконання етапів магістерської кваліфікаційної роботи контролюється керівником згідно з графіком виконання роботи. Прийняття магістерської кваліфікаційної роботи здійснюється ДЕК, затвердженою зав. кафедрою згідно з графіком

ДОДАТОК Б

(обов'язковий)

ПРОТОКОЛ ПЕРЕВІРКИ МАГІСТЕРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: Розробка веб-системи краудфандінгової платформи з інтеграцією штучного інтелекту

Тип роботи: магістерська кваліфікаційна робота

Підрозділ : кафедра програмного забезпечення, ФІТКІ, 1ПІ – 22м

Науковий керівник: Коваленко О. О.

Unicheck	
Оригінальність	**94.5%**
Схожість	5.5%

Аналіз звіту подібності

Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.

Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її автора. Роботу направити на доопрацювання.

Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку Черноволик Г. О.

Опис прийнятого рішення: **допустити до захисту**

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck

Автор роботи Ярошевич. М. С.

Керівник роботи Коваленко. О. О.

ДОДАТОК В

Лістинг вихідного коду

```
import { Body, Controller, Delete, Get, Param, Patch, Post } from '@nestjs/common';
import { CreateTicketDto } from './dto/CreateTicket.dto';
import { TicketService } from './ticket.service';
import { ChangeTicketStatusDto } from './dto/ChangeTicketStatus.dto';
import { create } from 'domain';

@Controller()
export class TicketController {

  constructor(private ticketService: TicketService) {}

  @Post('tickets/create')
  public async createTicket(@Body() createTicketDto: CreateTicketDto) {
    return await this.ticketService.createTicket(createTicketDto);
  }

  @Get('tickets/all')
  public async getAllTickets() {
    return await this.ticketService.getAllTickets();
  }

  @Get('tickets/get-count')
  public async getTicketCount() {
    return await this.ticketService.getTicketsCount();
  }

  @Get('tickets/:id')
  public async getTicketById(@Param('id') id: string) {
    return await this.ticketService.getTicketById(id);
  }

  @Get('tickets/company/:companyId')
  public async getTicketByCompanyId(@Param('companyId') id: string) {
    return await this.ticketService.getTicketByCompanyId(id);
  }
```

```
  @Post('tickets/:id/change-status')
   public async changeTicketStatus(@Body() changeProductStatusDto: ChangeTicketStatusDto, @Param('id') id:
string) {
    return await this.ticketService.changeTicketStatus(changeProductStatusDto, id);
  }

  @Patch('tickets/:id')
     public async updateTicketById(@Body() updateTicketDto: Omit<CreateTicketDto, "paymentMethods">,
@Param('id') id: string) {
    return await this.ticketService.updateTicketById(updateTicketDto, id);
  }

  @Delete('tickets/:id')
  public async deleteTicketById(@Param('id') id: string) {
    return await this.ticketService.deleteTicketById(id);
  }
}
import { BaseEntity } from 'src/common/base.entity';
import {
  Column,
  Entity,
  JoinTable,
  ManyToMany,
  ManyToOne,
  OneToMany,
  PrimaryGeneratedColumn,
} from 'typeorm';
import { CompanyEntity } from '../company/company.entity';
import { PaymentMethodEntity } from '../payment/payment-method.entity';

export enum TicketStatus {
  PLANNED = 'planned',
  ACTIVE = 'active',
  COMPLETED = 'completed',
}

export interface ITicketAttributes {
  id: string;
  title: string;
  ticketStartDate: string;
```

```
  ticketEndDate: string;
  companyId: string;
  company: CompanyEntity;
  status: TicketStatus;
  totalSum: number;
  currentSum: number;
  goal: string;
  description: string;
  analyzeResult: string;
  paymentMethods: PaymentMethodEntity[];
}

@Entity({ name: 'Ticket' })
export class TicketEntity extends BaseEntity implements ITicketAttributes {
  @Column({ type: 'varchar', length: 150 })
  title: string;
  @Column({ type: 'varchar', nullable: false })
  companyId: string;
  @Column({ type: 'timestamp without time zone', nullable: true })
  ticketStartDate: string;
  @Column({ type: 'timestamp without time zone', nullable: true })
  ticketEndDate: string;
  @Column({
    type: 'enum',
    enum: TicketStatus,
    default: TicketStatus.PLANNED,
    nullable: true,
  })
  status: TicketStatus;
  @ManyToOne(() => CompanyEntity, (company) => company.tickets)
  company: CompanyEntity;
  @Column({ type: 'double precision', nullable: true, default: 0 })
  currentSum: number;
  @Column({ type: 'double precision', nullable: false, default: 0 })
  totalSum: number;
  @Column({ type: 'varchar', length: 1500, nullable: true })
  goal: string;
  @Column({ type: 'varchar', length: 1500, nullable: true })
  description: string;
  @Column({ type: 'varchar', length: 1500, nullable: true })
```

```
  analyzeResult: string;

  @ManyToMany(() => PaymentMethodEntity, (paymentMethod) => paymentMethod.tickets)
  @JoinTable()
  paymentMethods: PaymentMethodEntity[];
}


import { Injectable } from '@nestjs/common';
import { InjectRepository } from '@nestjs/typeorm';
import {
  TicketEntity,
  TicketStatus,
} from './ticket.entity';
import { Repository } from 'typeorm';
import { CreateTicketDto } from './dto/CreateTicket.dto';
import { ChangeTicketStatusDto } from './dto/ChangeTicketStatus.dto';
import { PaymentMethodEntity } from '../payment/payment-method.entity';
import { PorterStemmer, SentimentAnalyzer, WordTokenizer } from 'natural';
import * as aposToLexForm from 'apos-to-lex-form'

@Injectable()
export class TicketService {
  constructor(
    @InjectRepository(TicketEntity)
    private ticketRepository: Repository<TicketEntity>,
    @InjectRepository(PaymentMethodEntity)
    private paymentMethodRepository: Repository<PaymentMethodEntity>,
  ) {}

  public async getTicketsCount() {
    return await this.ticketRepository.count();
  }

  public async createTicket(createTicketDto: CreateTicketDto) {
    console.log('DTO', createTicketDto);
    const {
      title,
      ticketStartDate,
      ticketEndDate,
```

```
    companyId,
    goal,
    description,
    totalSum,
  } = createTicketDto;

  const analyzeResult = await this.analyzeTicketDescription(description)

  // const ticketScoreAnalyze = this.analyzeTicketDescription(createTicketDto.description);
  let ticket = await this.ticketRepository.create({
    title,
    ticketStartDate,
    ticketEndDate,
    companyId,
    goal,
    description,
    totalSum,
    analyzeResult,
    status: TicketStatus.PLANNED,
  });
  const paymentMethods = []
  for (const paymentMethod of createTicketDto?.paymentMethods) {
    const createdPaymentMethod = await this.paymentMethodRepository.save({
      type: paymentMethod.type,
      cardNumber: paymentMethod.cardNumber,
    });
    paymentMethods.push(createdPaymentMethod)
  }

  ticket.paymentMethods = paymentMethods;

  this.ticketRepository.save(ticket)

  return {ticket}
}

public async getAllTickets() {
  return await this.ticketRepository.find();
}
```

```
public async getTicketById(id: string) {
  return await this.ticketRepository.findOne({where: {id}, relations: ['paymentMethods']})
}

public async updateTicketById(
  updateTicketDto: Omit<CreateTicketDto, 'paymentMethods'>,
  id: string,
) {
  return await this.ticketRepository.update(id, updateTicketDto);
}

public async getTicketByCompanyId(companyId: string) {
  return await this.ticketRepository.findBy({ companyId });
}

public async deleteTicketById(id: string) {
  return await this.ticketRepository.delete(id);
}

public async changeTicketStatus(
  changeTicketStatusDto: ChangeTicketStatusDto,
  id: string,
) {
  return await this.ticketRepository.update(id, changeTicketStatusDto);
}

public async analyzeTicketDescription(inputText: string) {
    const text = inputText

    const lexedReview = aposToLexForm(text);
    const casedReview = lexedReview.toLowerCase();
    const alphaOnlyReview = casedReview.replace(/[^a-zA-Z\s]+/g, '');

    const tokenizer = new WordTokenizer();
    const tokenizedReview = tokenizer.tokenize(alphaOnlyReview);

    const analyzer = new SentimentAnalyzer('English', PorterStemmer, 'afinn');
    const analysis = analyzer.getSentiment(tokenizedReview);

    console.log(`Тональність тексту: ${analysis}`);
```

```
    return analysis.toString();

    // if(analysis > 0) {
    //   return "Проект вартий уваги"
    // } else {
    //   return "Проект не вартий уваги"
    // }
  }
}
import { Controller, Post, UseGuards, Request, Body } from '@nestjs/common';
import { LocalAuthGuard } from './guards/localauth.guard';
import { JwtAuthGuard } from './guards/jwtauth.guard';
import { AuthService } from './auth.service';
import { CreateUserDto } from '../user/dto/CreateUser.dto';

@Controller()
export class AuthController {
  constructor(private authService: AuthService) {}

  @UseGuards(LocalAuthGuard)
  @Post('auth/login')
  public async login(@Request() req) {
    return this.authService.login(req.user)
  }

  @Post('auth/forgot-password')
  public async forgotPassword(@Request() req) {
    return this.authService.forgotPassword(req.user)
  }

  @Post('auth/register')
  public async register(@Body() createUserDto: CreateUserDto) {
    return this.authService.register(createUserDto)
  }
}
import { Injectable } from '@nestjs/common';
import { UserService } from '../user/user.service';
import { JwtService } from '@nestjs/jwt';
import { InjectRepository } from '@nestjs/typeorm';
```

```
import { IUserAttributes, UserEntity } from '../user/user.entity';
import { Repository } from 'typeorm';
import { CreateUserDto } from '../user/dto/CreateUser.dto';
import { ConfigService } from '@nestjs/config';

@Injectable()
export class AuthService {
 private jwt_secret;
 constructor(
  private userService: UserService,
  private jwtService: JwtService,
  private configService: ConfigService
 ) {
  this.jwt_secret = this.configService.get('jwt_secret')
 }

 public async validateUser(email: string, password: string) {
  const user = await this.userService.findByEmail(email);

  if (user && user.password === password) {
   const { password, ...result } = user;
   return result;
  }

  return null;
 }

 public login(user: any) {
  const payload = { email: user.email, id: user.id, companyId: user.companyId };
  return {
   access_token: `Bearer ${this.jwtService.sign(payload, { secret: this.jwt_secret })}`,
  };
 }

 public forgotPassword(user: IUserAttributes) {
  return user.password
 }

 public async register(createUserDto: CreateUserDto) {
  return await this.userService.createUser(createUserDto);
```

```
  }
}
import { Body, Controller, Get, Param, Patch, Post, Request, UseGuards } from '@nestjs/common';
import { CompanyService } from './company.service';
import { CreateCompanyDto, UpdateCompanyDto } from './dto/CreateCompanyDto';
import { JwtAuthGuard } from '../auth/guards/jwtauth.guard';

@Controller('company')
export class CompanyController {

  constructor(private companyService: CompanyService) {}

  @UseGuards(JwtAuthGuard)
  @Post('/create')
  public async createCompany(@Request() req, @Body() createTicketDto: CreateCompanyDto) {
    console.log("REQ", req)
    return await this.companyService.createCompany(createTicketDto, req?.user?.id);
  }

  @Get('/:id')
  public async getTicketById(@Param('id') id: string) {
    return await this.companyService.getCompanyById(id);
  }

  @Patch('update/:id')
  public async updateTicketById(@Body() updateCompanyDto: UpdateCompanyDto, @Param('id') id: string) {
    return await this.companyService.updateCompanyById(id, updateCompanyDto);
  }
}
import { BaseEntity } from 'src/common/base.entity';
import { Column, Entity, JoinTable, OneToMany } from 'typeorm';
import { TicketEntity } from '../ticket/ticket.entity';
import { UserEntity } from '../user/user.entity';

export interface ICompanyAttributes {
  id: string;
  name: string;
  description: string;
  crumbs: string;
  tickets: TicketEntity[];
```

```
  members: string;
}

@Entity({ name: 'Company' })
export class CompanyEntity extends BaseEntity implements ICompanyAttributes {
  @Column({ type: 'varchar', length: 150 })
  name: string;
  @Column({ type: 'varchar', length: 3000, nullable: true })
  description: string;
  @Column({ type: 'varchar', nullable: true })
  crumbs: string;
  @OneToMany(() => TicketEntity, (ticket) => ticket.company)
  tickets: TicketEntity[];
  @OneToMany(() => UserEntity, user => user.company)
  users: UserEntity[];
  @Column({ type: 'varchar', length: "5000", nullable: true })
  members: string
}
import { Injectable } from '@nestjs/common';
import { CreateCompanyDto, UpdateCompanyDto } from './dto/CreateCompanyDto';
import { CompanyEntity } from './company.entity';
import { InjectRepository } from '@nestjs/typeorm';
import { Repository } from 'typeorm';
import { UserEntity } from '../user/user.entity';

@Injectable()
export class CompanyService {
  constructor(
    @InjectRepository(CompanyEntity)
    public companyRepository: Repository<CompanyEntity>,
    @InjectRepository(UserEntity)
    private userEntityRepository: Repository<UserEntity>,
  ) {}

  public async createCompany(
    createCompanyDto: CreateCompanyDto,
    userId: string,
  ): Promise<CompanyEntity> {
    const company = await this.companyRepository.save(createCompanyDto);
    const user = await this.userEntityRepository.findOneBy({ id: userId });
```

```
    if(!user) {
      throw new Error("User does not exist")
    }
    user.company = company
    user.companyId = company.id
    await this.userEntityRepository.save(user)
    return company
  }

  public async getCompanyById(id: string) {
    return await this.companyRepository.findOneBy({ id });
  }

  public async updateCompanyById(id: string, updateCompanyDto: UpdateCompanyDto) {
    return await this.companyRepository.update(id, updateCompanyDto)
  }
}

import { Component, OnInit } from '@angular/core';
import { TicketService } from '../../services/ticket.service';
import { ITicket } from '../../types/types';
import { Router } from '@angular/router';

@Component({
  selector: 'app-main-tickets',
  templateUrl: './main-tickets.component.html',
  styleUrls: ['./main-tickets.component.scss'],
})
export class MainTicketsComponent implements OnInit {
  public tickets!: ITicket[];

  constructor(private ticketService: TicketService, private router: Router) {}

  ngOnInit(): void {
    this.ticketService.getMainTickets().subscribe({
      next: (tickets: any) => {
        console.log("TICKETS", tickets)
        this.tickets = tickets;
      },
    });
```

```
  }

  public selectTicket(ticketId: string) {
    this.router.navigate([`/main/orders/${ticketId}`])
  }
}

import { Component } from '@angular/core';
import { TicketService } from '../../services/ticket.service';
import { ActivatedRoute, Router } from '@angular/router';
import { ITicket } from '../../types/types';

@Component({
  selector: 'app-ticket',
  templateUrl: './ticket.component.html',
  styleUrls: ['./ticket.component.scss'],
})
export class TicketComponent {
  private ticketId!: string;
  public ticket!: ITicket;
  constructor(
    private ticketService: TicketService,
    private route: ActivatedRoute
  ) {
    this.route.params.subscribe({
      next: (params:any) => (this.ticketId = params.id),
    });
  }

  ngOnInit(): void {
    this.ticketService.getTicketById(this.ticketId).subscribe({
      next: (ticket: any) => {
        console.log('TICKET', ticket);
        this.ticket = ticket;
      },
    });
  }
}

import { Component } from '@angular/core';
```

```
import { ITicket } from '../../types/types';
import { TicketService } from '../../services/ticket.service';
import { Router } from '@angular/router';

@Component({
  selector: 'app-created-tickets',
  templateUrl: './created-tickets.component.html',
  styleUrls: ['./created-tickets.component.scss']
})
export class CreatedTicketsComponent {
  public tickets!: ITicket[];

  constructor(private ticketService: TicketService, private router: Router) {}

  ngOnInit(): void {
    this.ticketService.getMainTickets().subscribe({
      next: (tickets: any) => {
        console.log("TICKETS", tickets)
        this.tickets = tickets;
      },
    });
  }

  public selectTicket(ticketId: string) {
    this.router.navigate([`/main/orders/${ticketId}`])
  }

  public navigateToCreateTicket() {
    this.router.navigate([`/main/orders/create`])
  }
}
import { Component } from '@angular/core';
import { FormArray, FormControl, FormGroup, Validators } from '@angular/forms';
import { Router } from '@angular/router';
import { TicketService } from '../../services/ticket.service';
import { JwtService } from 'src/app/modules/shared/services/jwt.service';

@Component({
  selector: 'app-create-ticket',
  templateUrl: './create-ticket.component.html',
```

```
  styleUrls: ['./create-ticket.component.scss'],
})
export class CreateTicketComponent {
  private tokenData: any;
  public createTicketForm!: FormGroup<{
    title: FormControl<string | null>;
    ticketStartDate: FormControl<string | null>;
    ticketEndDate: FormControl<string | null>;
    goal: FormControl<string | null>;
    description: FormControl<string | null>;
    totalSum: FormControl<string | null>;
    paymentMethods: FormArray;
  }>;

  get paymentMethods(): FormArray {
    return this.createTicketForm.get('paymentMethods') as FormArray;
  }

  public baseFormGroup = () => {
    return new FormGroup({
      type: new FormControl(''),
      cardNumber: new FormControl(''),
    })
  }

  public paymentsArray = new FormArray([
    this.baseFormGroup()
  ])

  constructor(
    private router: Router,
    private ticketService: TicketService,
    private jwtService: JwtService
  ) {
    const token = localStorage.getItem('access_token')?.split(' ')[1];
    this.tokenData = this.jwtService.getTokenPayload(token as string);
  }

  ngOnInit(): void {
    this.createTicketForm = new FormGroup({
```

```
    title: new FormControl('', [Validators.required]),
    ticketStartDate: new FormControl('', [Validators.required]),
    ticketEndDate: new FormControl('', [Validators.required]),
    goal: new FormControl('', [Validators.required]),
    description: new FormControl('', [Validators.required]),
    totalSum: new FormControl('0', [Validators.required, Validators.min(0)]),
    paymentMethods: this.paymentsArray
  });
 }

 public addPayment() {
  this.paymentsArray.push(this.baseFormGroup())
 }

 public submitCreation() {
  console.log('this', this.createTicketForm.controls)
  const isFormValid = this.createTicketForm.valid;
  if (!isFormValid) {
   alert('Form invalid');
   return;
  }

  const formValue = this.createTicketForm.value;

  console.log(formValue);

  const createTicketPayload = {
   ...formValue,
   totalSum: +(formValue.totalSum as string),
   companyId: this.tokenData.companyId,
  };

  this.ticketService.createTicket(createTicketPayload as any).subscribe({
   next: (response) => {
    console.log('RES', response);
   },
  });
 }
}
```

```
function transformDateToMS(date: string) {
  return new Date(date).getTime();
}

import { Component, OnInit } from '@angular/core';
import { CompanyService } from '../../services/company.service';
import { JwtService } from 'src/app/modules/shared/services/jwt.service';
import { FormArray, FormControl, FormGroup, Validators } from '@angular/forms';

@Component({
  selector: 'app-profile',
  templateUrl: './profile.component.html',
  styleUrls: ['./profile.component.scss'],
})
export class ProfileComponent implements OnInit {
  public companyInfo: any;
  public companyId!: string;

  public companyForm!: FormGroup<{
    name: FormControl<string | null>;
    crumbs: FormControl<string | null>;
    description: FormControl<string | null>;
    members: FormArray;
  }>;

  get members(): FormArray {
    return this.companyForm.get('members') as FormArray;
  }

  public baseFormGroup = () => {
    return new FormGroup({
      image: new FormControl(''),
      name: new FormControl(''),
      surname: new FormControl(''),
    })
  }

  public membersArray = new FormArray([
    this.baseFormGroup()
  ])
```

```
constructor(
  private companyService: CompanyService,
  private jwtService: JwtService
) {
  const token = localStorage.getItem('access_token')?.split(' ')[1];
  const tokenData = this.jwtService.getTokenPayload(token as string) as any;
  this.companyId = tokenData.companyId;
}

ngOnInit(): void {

  this.companyForm = new FormGroup({
    name: new FormControl('', [Validators.required]),
    crumbs: new FormControl('', [Validators.required]),
    description: new FormControl('', [Validators.required]),
    members: this.membersArray
  });

  this.companyService.getById(this.companyId).subscribe({
    next: (response: any) => {
      console.log('response', response);
      this.companyInfo = {...response, members: JSON.parse(response?.members) };
      console.log(this.companyInfo)
      for (let index = 1; index < this.companyInfo.members.length; index++) {
        this.addGroup();
      }
      this.companyForm.patchValue({
        name: this.companyInfo.name,
        crumbs: this.companyInfo.crumbs,
        description: this.companyInfo.description,
        members: this.companyInfo.members
      })
      this.companyForm.updateValueAndValidity();
    },
  });
}

addGroup() {
  this.members.push(this.baseFormGroup())
```

```
  }

  deleteMember(i: any) {
    this.members.removeAt(i)
  }

  public submitUpdate() {
    const isFormValid = this.companyForm.valid
    if(!isFormValid) {
      alert("Form invalid")
      return
    }

    const formValue = this.companyForm.value;
    const payload = {...formValue, members: JSON.stringify(formValue.members)}

    this.companyService.updateById(this.companyId, payload).subscribe({next: (response: any) => {
      console.log(response)
    }})
  }
}

public submitDelete() {
    const isFormValid = this.companyForm.valid
    if(!isFormValid) {
      alert("Form invalid")
      return
    }

    const formValue = this.companyForm.value;
    const payload = {...formValue, members: JSON.stringify(formValue.members)}

    this.companyService.updateById(this.companyId, payload).subscribe({next: (response: any) => {
      console.log(response)
    }})
  }
}

import { Component, OnInit } from '@angular/core';
import { ActivatedRoute } from '@angular/router';
```

```
import { CompanyService } from '../../services/company.service';


@Component({
  selector: 'app-profile-info',
  templateUrl: './profile-info.component.html',
  styleUrls: ['./profile-info.component.scss'],
})
export class ProfileInfoComponent implements OnInit {
  public companyInfo: any;
  public companyId!: string;

  constructor(
    private route: ActivatedRoute,
    private companyService: CompanyService
  ) {
    this.route.params.subscribe({
      next: (params: any) => (this.companyId = params.id),
    });
  }

  getRole(role: number) {
    return role === 1 ? 'Директор' : "Співробітник"
  }

  ngOnInit(): void {
    this.companyService.getById(this.companyId).subscribe({
      next: (response: any) => {
        console.log('response', response);
        this.companyInfo = {...response, members: JSON.parse(response?.members)};
      },
    });
  }
}
```

ДОДАТОК Г

ІЛЮСТРАТИВНА ЧАСТИНА

РОЗРОБКА ВЕБ-СИСТЕМИ КРАУДФАНДІНГОВОЇ ПЛАТФОРМИ З ІНТЕГРАЦІЄЮ ШТУЧНОГО ІНТЕЛЕКТУ

Рисунок Г.1 - Слайд презентації №1

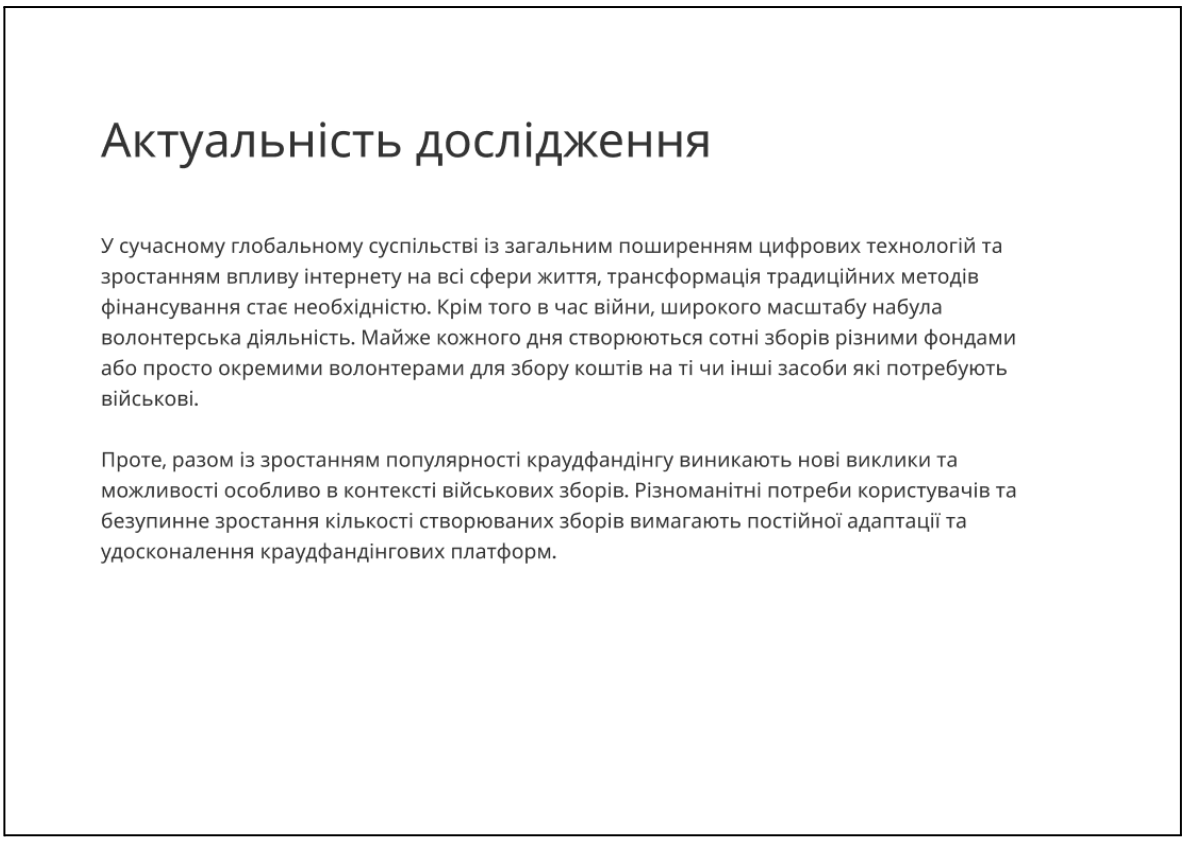


Рисунок Г.2 - Слайд презентації №2

Рисунок Г.3 - Слайд презентації №3

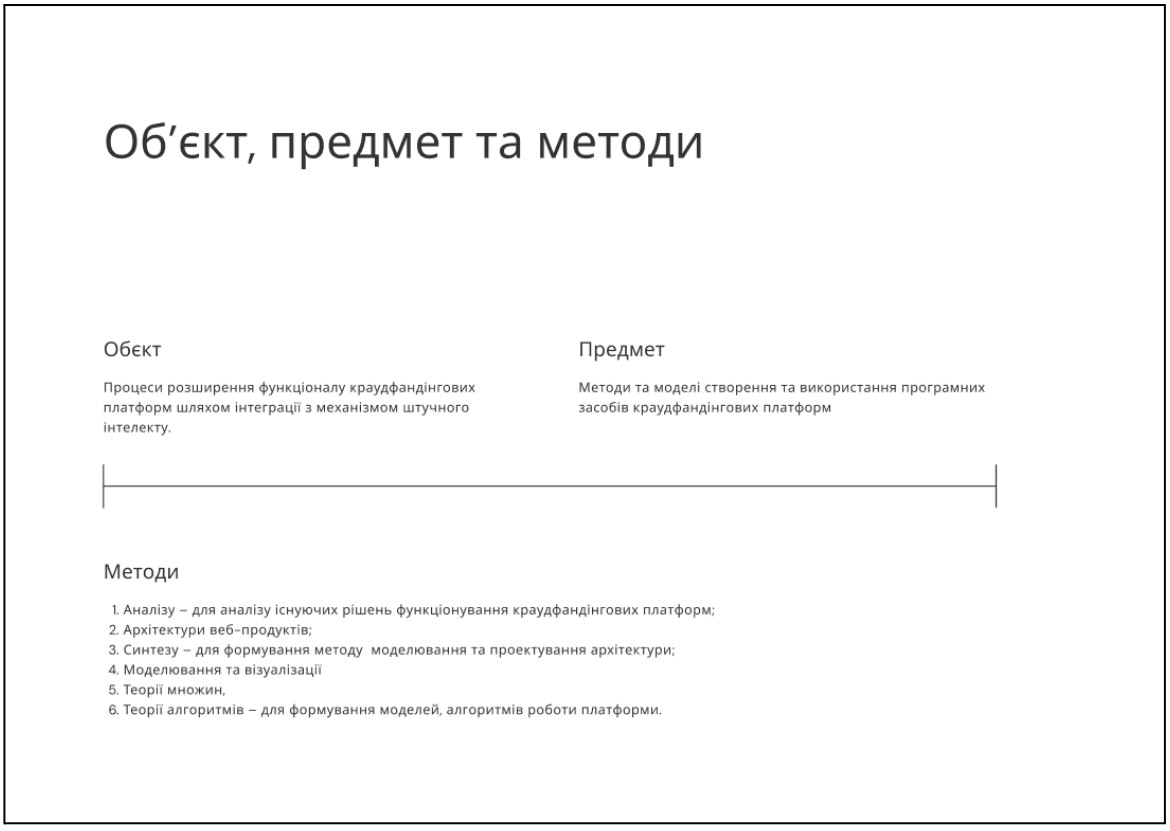


Рисунок Г.4 - Слайд презентації №4

Наукова новизна

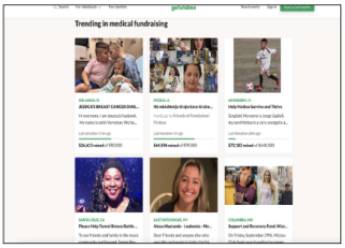
1. Удосконалено метод створення краудфандінгової платформи за рахунок використання NLP-технологій, моделі прийняття рішення щодо інвестування проєкту, алгоритмів формування клієнтської та серверної частини, які, на відміну, від існуючих, дозволяють здійснювати аналіз текстової інформації краудфандінгових проєктів, на основі результатів якого виконується автоматизоване оцінювання, сортування релевантних компаній, що дозволяє розширити функціонал платформи, більш детально та достовірно підготувати дані для прийняття рішення щодо обґрунтованого вибору проєкту для інвестування
2. Набула подальшого розвитку модель прийняття рішення на основі результатів аналізу текстових даних щодо фінансування проєктів, яка, на відміну від існуючих, дозволяє здійснювати аналіз текстової інформації краудфандінгових проєктів за рахунок використання NLP-технології та враховувати експертну оцінку користувача (або/і запрошених експертів), що дозволяє розширити функціонал платформи, врахувати результати автоматизованого та експертного оцінювання, більш детально та достовірно підготувати дані для прийняття рішення щодо вибору проєкту користувачем

Рисунок Г.5 - Слайд презентації №5

Існуючі системи

GoFundMe

Заснована у 2010 році та має глибокий соціальний вплив, дозволяючи тисячам людей залучати кошти для вирішення найрізноманітніших завдань та проблем



StartEngine

Сприяє зростанню стартапів та компаній через еквіті краудфандінг. Вона надає можливість інвестувати в перспективні проекти та сприяє розвитку інновацій та технологічного прогресу

Kickstarter

Основна мета платформи – дати можливість творчим та креативним особам залучити фінансування для реалізації своїх ідей та проектів

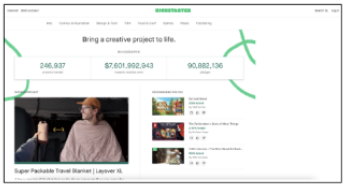


Рисунок Г.6 - Слайд презентації №6

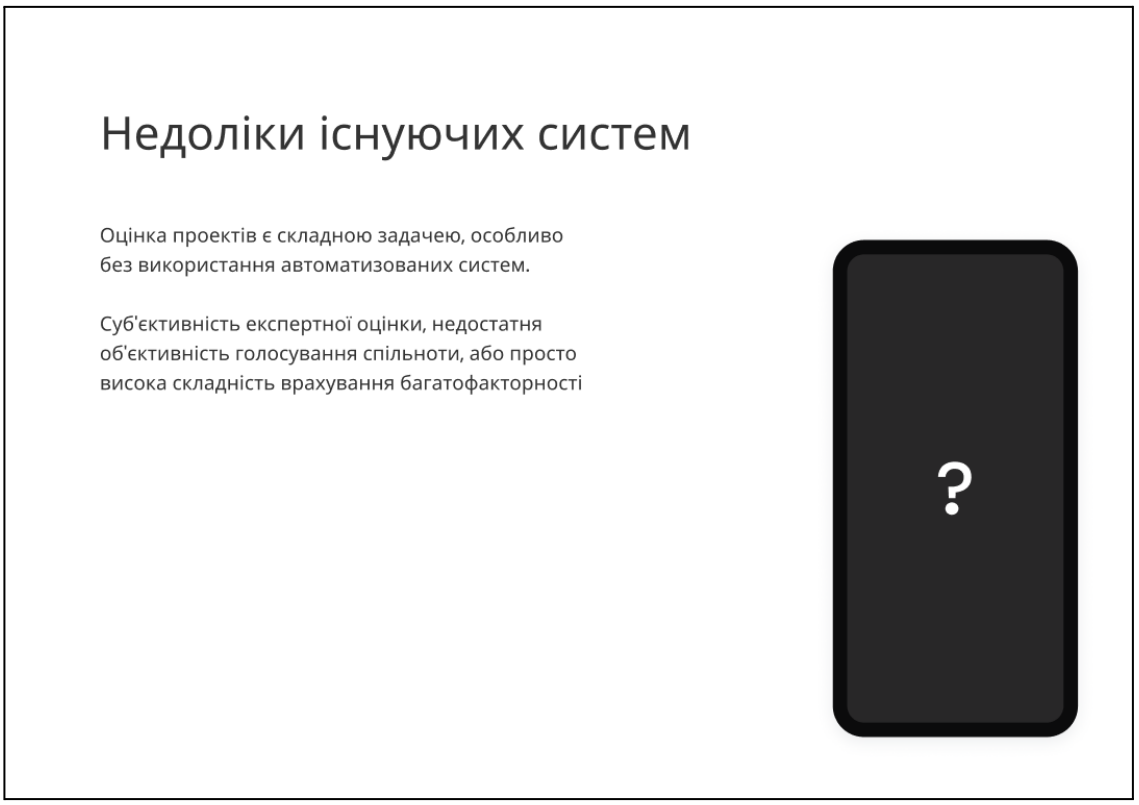


Рисунок Г.7 - Слайд презентації №7

Рисунок Г.8 - Слайд презентації №8

Рисунок Г.9 - Слайд презентації №9

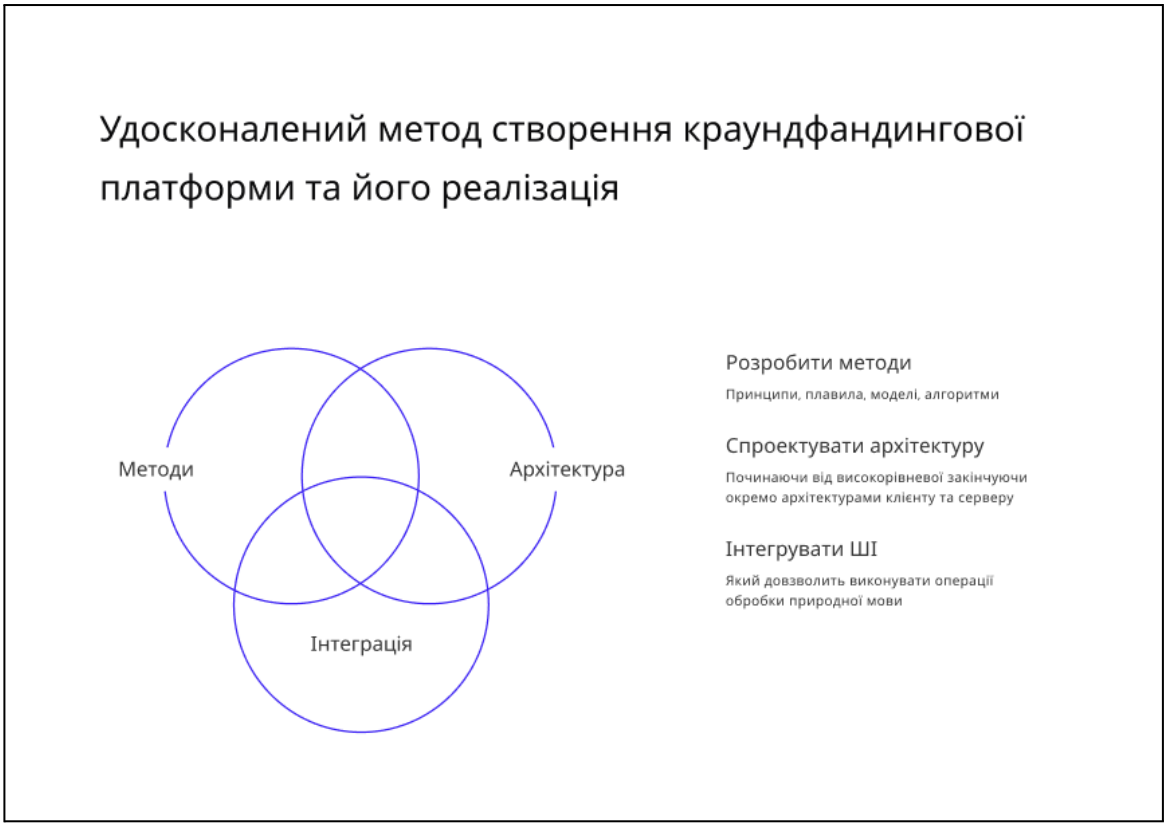


Рисунок Г.10 - Слайд презентації №10

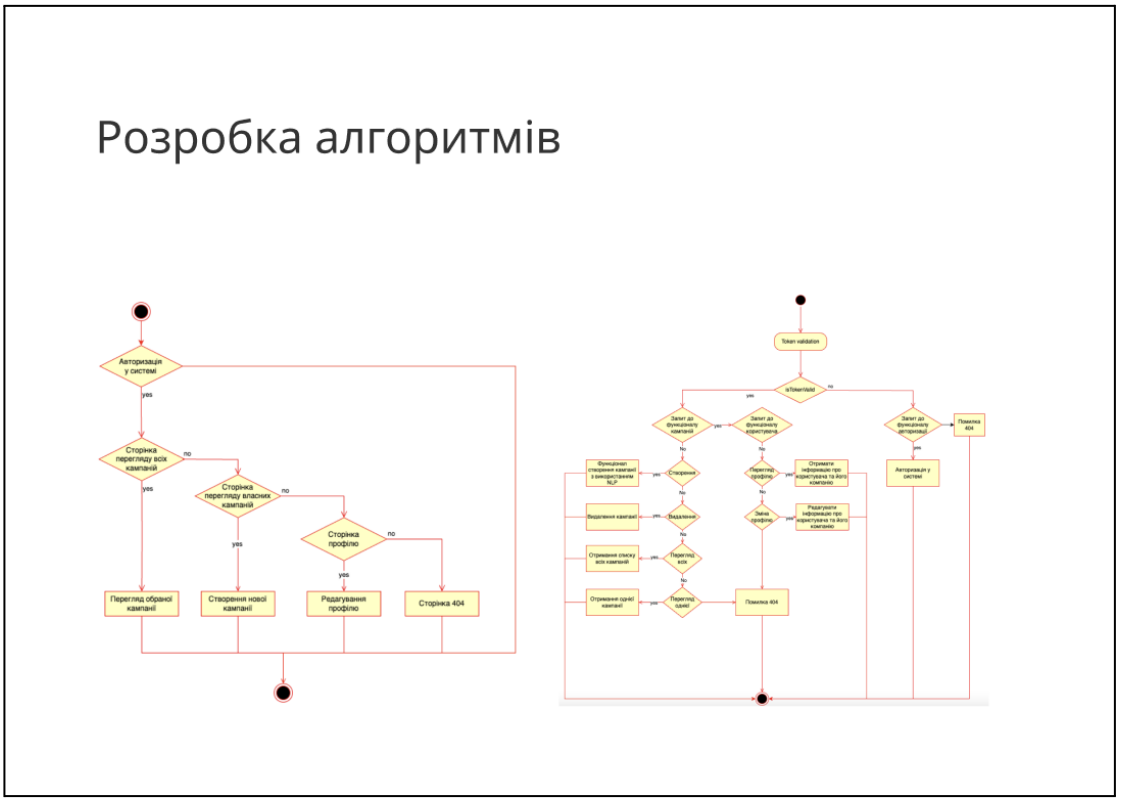


Рисунок Г.11 - Слайд презентації №11

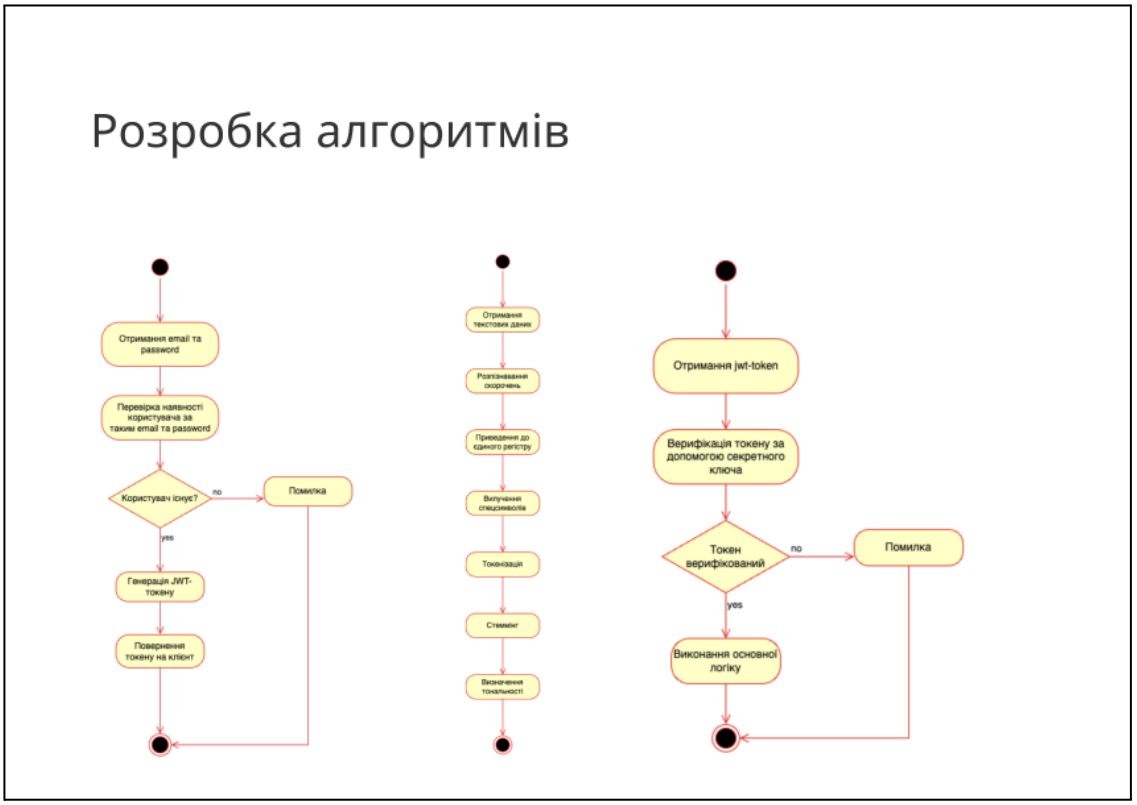


Рисунок Г.12 - Слайд презентації №12

Рисунок Г.13 - Слайд презентації №13

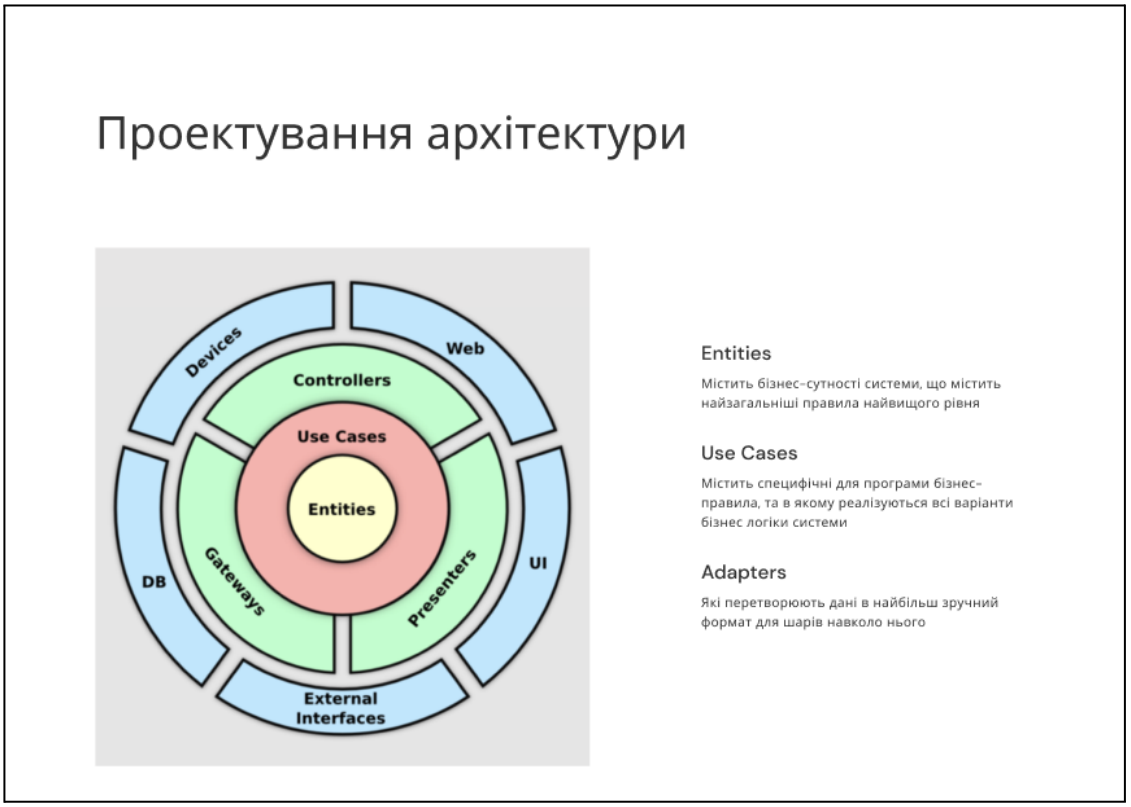


Рисунок Г.14 - Слайд презентації №14

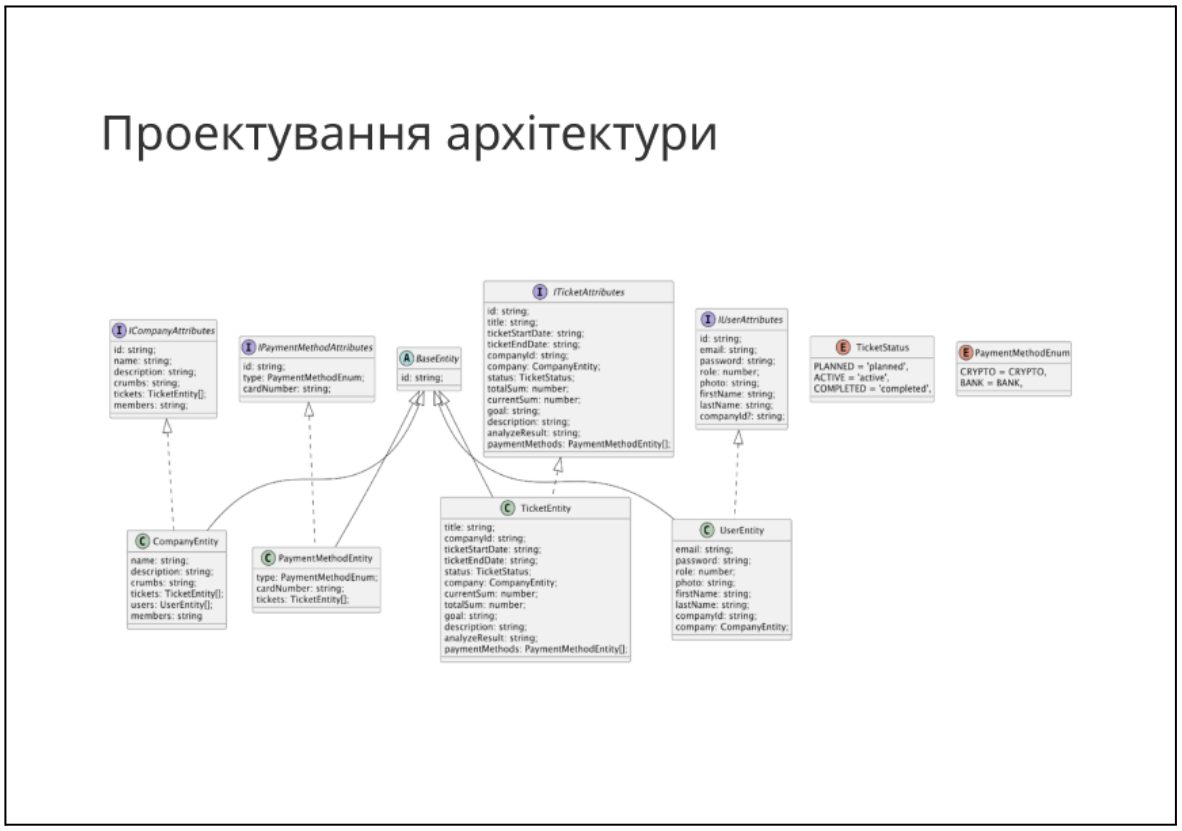


Рисунок Г.15 - Слайд презентації №15

Рисунок Г.16 - Слайд презентації №16

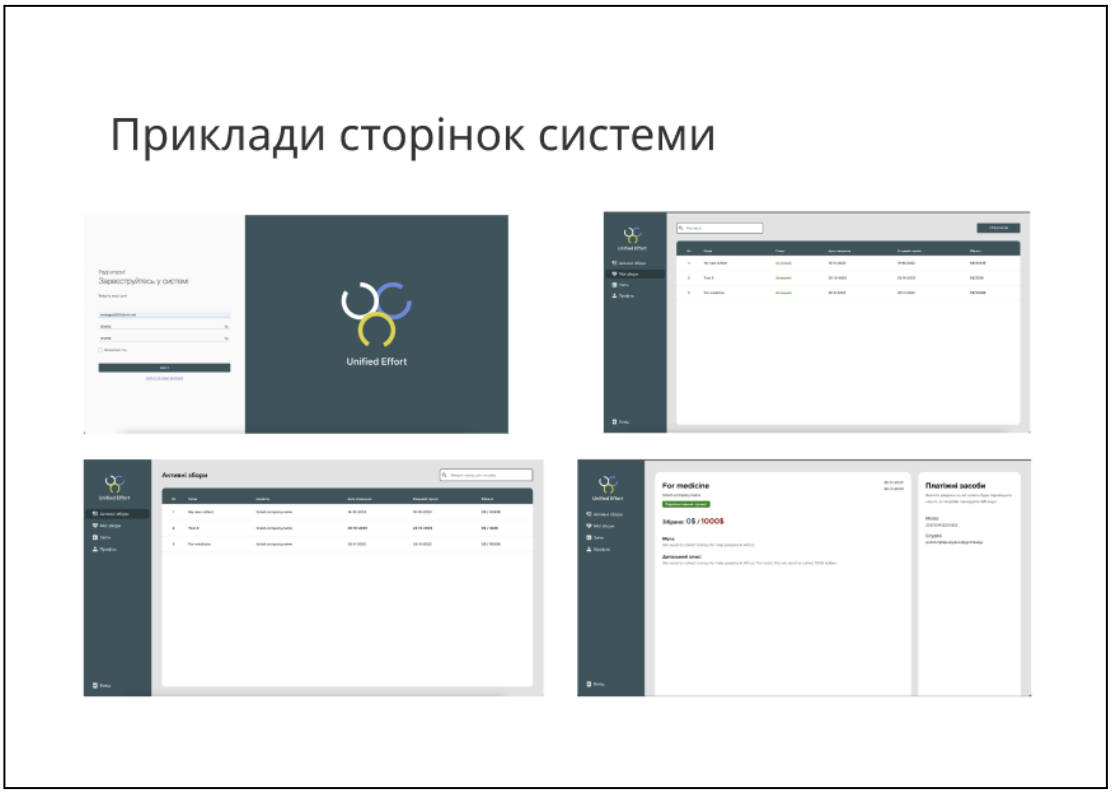


Рисунок Г.17 - Слайд презентації №17

Рисунок Г.18 - Слайд презентації №18

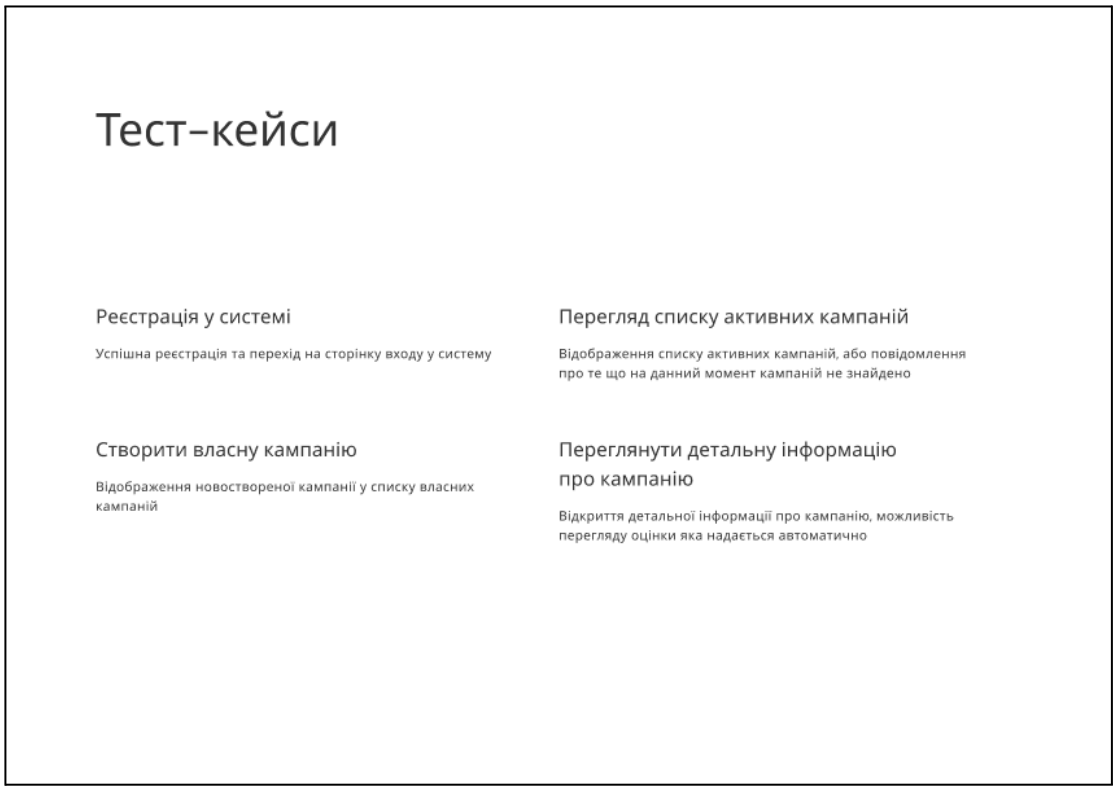


Рисунок Г.19 - Слайд презентації №19

Рисунок Г.20 - Слайд презентації №20

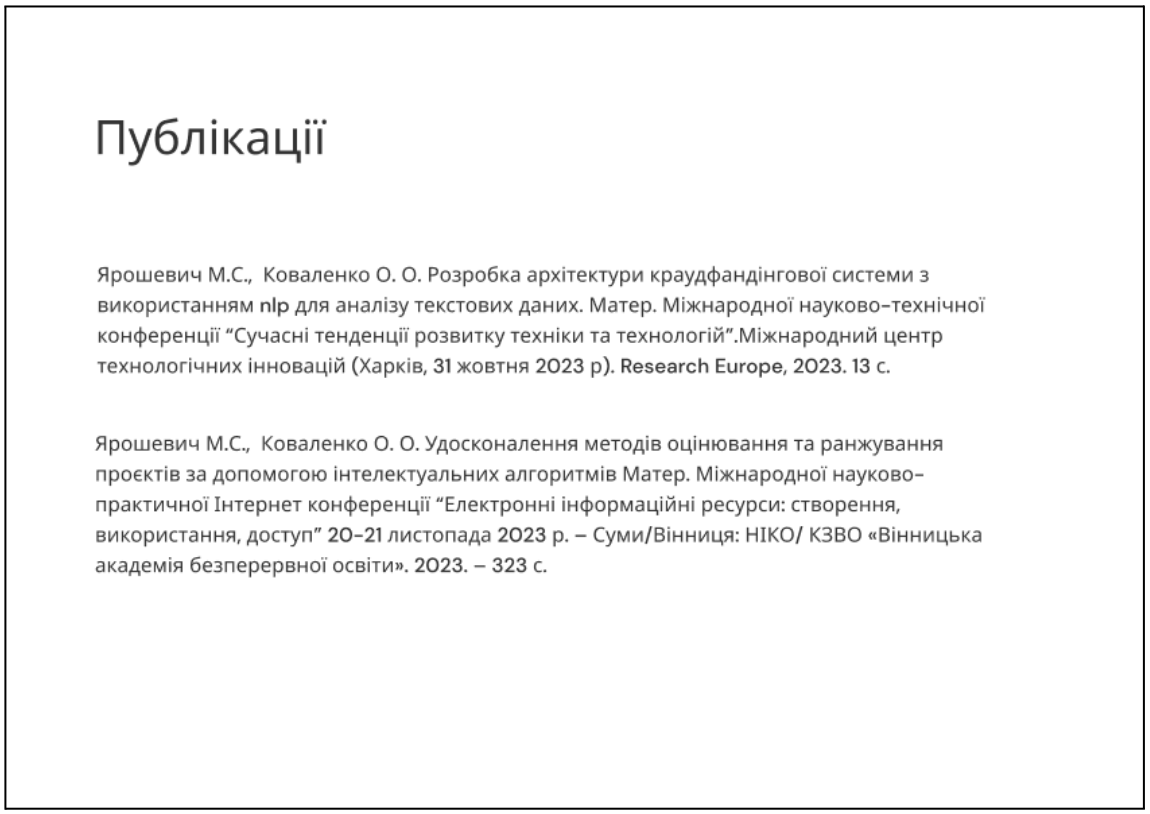


Рисунок Г.21 - Слайд презентації №21