Вінницький національний технічний університет
(повне найменування вищого навчального закладу)
Факультет інформаційних технологій та комп'ютерної інженерії
(повне найменування інституту, назва факультету (відділення))
Кафедра програмного забезпечення
(повна назва кафедри (предметної, циклової комісії))

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Розробка програмної системи моніторингу та контролю за благоустроєм»

Виконав: студент 2-го курсу, групи ЗПІ-22м
спеціальності 121 – Інженерія програмного забезпечення
(шифр і назва напряму підготовки, спеціальності)
Кондрук Р. В .
(прізвище та ініціали)
Керівник: к.т.н., доц. каф. ПЗ:
Ракитянська Г. Б.
(прізвище та ініціали)
«15» грудня 2023 р.
Опонент к.т.н., доц. каф. ЗІ:
Баришев Ю. В.
(прізвище та ініціали)
«15» грудня 2023 р.

Допущено до захисту
Завідувач кафедри ПЗ
д.т.н, проф. Романюк О.Н.
(прізвище та ініціали)
«15» грудня 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 КВАЛІФІКАЦІЙУ РОБОТУ СТУДЕНТУ

Кондруку Роману Володимировичу

1. Тема роботи – розробка програмної системи моніторингу та контролю за благоустроєм.
Керівник роботи: Ракитянська Ганна Борисівна, к.т.н., доц. кафедри ПЗ, затверджені наказом вищого навчального закладу від « 18 » вересня 2023 р. № 247.
2. Строк подання студентом роботи
5 грудня 2023 року
3. Вихідні дані до роботи: алгоритм кластеризації маркерів на мапі, метод внесення змін до проектів благоустрою; вихідні дані – програмна система моніторингу та контролю за благоустроєм.
4. Зміст розрахунково-пояснювальної записки: вступ; аналіз стану систем моніторингу та контролю за благоустроєм, порівняльний аналіз аналогів, аналіз методів розв'язання поставленої задачі, постановка задачі; аналіз інформаційного забезпечення, розробка загальної моделі програмної системи, аналіз принципів кластеризації маркерів на мапі, розробка алгоритму кластеризації для програмної системи моніторингу та контролю за благоустроєм, розробка моделі програмної системи "БлагоМіст", розробка моделі інтерфейсу користувача та вибір дизайну додатку, розробка алгоритмів роботи додатку; аналіз і обґрунтування вибору засобів для реалізації програмного засобу, програмна реалізація додатку; тестування програмного додатку; економічна частина; висновки; перелік посилань; додатки

5. Перелік графічного матеріалу: моделі системи; блок-схеми алгоритмів роботи додатку; тестування додатку.
6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	Ракитянська Г. Б., к.т.н., доцент кафедри ПЗ	19.09.2023	01.12.2023
5	Причепа І. В., к.е.н., доц. каф. ЕПВМ	22.11.2023	01.12.2023

7. Дата видачі завдання 19 вересня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Обґрунтування вибору методу розробки та постановка задачі дослідження	20.09.2023 – 30.09.2023	вик
2	Розробка методу та моделей програмної системи моніторингу та контролю за благоустроєм	01.10.2023 – 17.10.2023	вик
3	Розробка програмних компонент додатку	18.10.2023 – 04.11.2023	вик
4	Тестування програмного додатку	05.11.2023 – 21.11.2023	вик
5	Економічна частина	22.11.2023 – 01.12.2023	вик

Студент (підпис) Кондрук Р. В. (прізвище та ініціали)

Керівник магістерської кваліфікаційної роботи (підпис) Ракитянська Г. Б. (прізвище та ініціали)

Анотація

УДК 519.7:004.89

Кондрук Р.В. Розробка програмної системи моніторингу та контролю за благоустроєм. Магістерська кваліфікаційна робота зі спеціальності 121 – інженерія програмного забезпечення, освітня програма – інженерія програмного забезпечення. Вінниця: ВНТУ, 2023. 114 с.
На укр. мові. Бібліогр.: 30 назв; рис.: 31; табл. 15.

У магістерській кваліфікаційній роботі проведено детальний аналіз підходів до створення програмних систем з благоустрою.

Запропоновано використовувати метод кластеризації маркерів на мапі.

Розроблено алгоритм кластеризації маркерів на мапі, який використовує принцип визначення відстаней між точками та об’єднання їх за критеріями розміщення, що сприяє покращеній роботі об’єднання маркерів у групи-кластери. Розроблено модуль для відображення запитів на проведення робіт, базуючись на інформації переданій від користувачів, та повний функціонал управління даними запитами. Розроблено програмну систему з використанням технологій Express js, Angular, Node js та мови програмування JavaScript. Програмний застосунок використовує NoSql базу даних MongoDB, що має вбудовану підтримку Mongoose.

Отримані в магістерській кваліфікаційній роботі результати можна використати для проведення тендерів на проекти з благоустрою як приватним особам так і муніципалітетам.

Ключові слова: програмна система, проекти з благоустрою, благоустрій, кластеризація маркерів, муніципалітети, запити громадян.

Abstract

Kondruk R.V. Development of a software system for monitoring and control of public works. Master's thesis on the specialty 121 - software engineering, educational program - software engineering. Vinnytsia: VNTU, 2023. 114 p.
In Ukrainian speech Bibliography: 30 titles; Fig.: 31; table 15.

In the master's qualification work, a detailed analysis of approaches to the creation of software systems with landscaping was carried out.

It is proposed to use the method of clustering markers on the map.

An algorithm for clustering markers on the map has been developed, which uses the principle of determining the distances between points and combining them according to placement criteria, which contributes to the improved work of combining markers into cluster groups. A module has been developed for displaying requests for work, based on information provided by users, and full functionality for managing these requests. A software system was developed using Express js, Angular, Node js and JavaScript programming languages. The software application uses the NoSql database MongoDB, which has built-in support for Mongoose.

The results obtained in the master's qualification work can be used for conducting tenders for improvement projects for both private individuals and municipalities.

Keywords: software system, improvement projects, improvement, marker clustering, municipalities, citizen requests.

ЗМІСТ

ВСТУП

Обґрунтування вибору теми дослідження. Сучасне суспільство переживає значний ріст уваги до питань благоустрою та розвитку громадських просторів. Забезпечення комфортних та безпечних умов для проживання є важливою складовою високої якості життя громадян [1]. У цьому контексті розробка та впровадження програмної системи моніторингу і контролю за благоустроєм стає надзвичайно актуальною та перспективною ініціативою.

У сучасних містах зростає кількість населення та розвивається інфраструктура, що призводить до виникнення проблем у забезпеченні якісного благоустрою. Перспективи ефективного управління міським середовищем та підвищення якості комунальних послуг роблять цю тему надзвичайно актуальною.

Якість благоустрою напряму впливає на якість життя мешканців [2]. Це стосується якісної інфраструктури (дороги, парки, сквери), так і надання комунальних послуг (водопостачання, вивіз сміття). Розробка системи моніторингу та контролю може покращити ці аспекти.

Сучасні інформаційні технології надають широкі можливості для вирішення подібних завдань. Використання програмного забезпечення для моніторингу та контролю є ефективним та інноваційним підходом.

Ефективне управління ресурсами [3] та оптимізація витрат у сфері благоустрою може призвести до значних економічних вигод для місцевої влади та громади в цілому.

Розробка системи, яка включає в себе механізми залучення громадськості до моніторингу та контролю за благоустроєм, може сприяти активнішій участі громадян у власному міському середовищі та стимулювати взаємодію з місцевою владою.

Ця програмна система призначена не лише для ефективного відстеження різноманітних благоустрійних проектів, а й для залучення громадськості до активної участі у розвитку своєї місцевості. Вона надає можливість громадянам слідкувати за виконанням проектів, висловлювати свої думки через голосування

та коментування, а також надавати власні ідеї та зміни для розгляду органами місцевого самоврядування.

Таким чином, обґрунтовуючи вибір цієї теми, можна зазначити, що розробка програмної системи моніторингу і контролю за благоустроєм відповідає сучасним викликам міського середовища, сприяє покращенню якості життя мешканців, забезпечує ефективне використання ресурсів та розвиває партнерські відносини між владою та громадою.

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

Мета та завдання дослідження. Метою роботи є підвищення ефективності роботи програмної системи з моніторингу та контролю за благоустроєм, що дозволить покращати комунікацію громадськості з місцевим самоврядування у питаннях урбаністики, з використанням методу кластеризації, що дозволить покращити роботу інтерактивної мапи програмної системи.

Задачі дослідження:

- проаналізувати сучасний стан веб-додатків з питань благоустрою;
- проаналізувати існуючі алгоритми та їх реалізацію для вирішення питання розміщення проектів по благоустрою на інтерактивній мапі;
- розробити алгоритм для реалізації власної програми для інформаційної системи по контролю за благоустроєм;
- розробити метод кластеризації маркерів на мапі;
- розробити алгоритм опрацювання проектів з благоустрою, від подачі ініціатором до введення в експлуатацію;
- розробити веб-додаток для громадян та міської адміністрації для комунікації по заданим проектам;
- провести тестування розробленого програмного веб-додатку.

Об'єктом дослідження є процес створення програмної системи "БлагоМіст" з моніторингу та керування благоустроєм із використанням інтерактивної мапи проектів.

Предмет дослідження – засоби і методи управління благоустроєм шляхом комунікації громадськості та місцевого самоврядування.

Методи дослідження. У магістерській кваліфікаційній роботі використовувалися: алгоритм кластеризації маркерів на мапі, для об’єднання маркерів у групи, алгоритм визначення відстані між точками на мапі, для більш оптимізованої кластеризації, алгоритм верифікації користувачів, що забезпечує автентифікацію на платформі, методи тестування для перевірки роботи створеного програмного додатку.

Наукова новизна отриманих результатів.

- Подальшого розвитку дістав метод управління проектами з благоустрою, який на відміну від існуючих, реалізує принципи комунікації з громадянами, принципи проведення робіт з благоустрою та пришвидшує комунікацію з муніципалітетами шляхом надання зручного інструментарію комунікації наживо;
- Подальшого розвитку дістав метод кластеризації маркерів на мапі, який на відміну від існуючих з більшою точністю та швидкістю розбиває маркери за координатами та допомагає користувачам більш точно орієнтуватися в проектах його міста.

Практична цінність одержаних результатів. Практична цінність одержаних результатів полягає в тому, що на основі отриманих в магістерській кваліфікаційній роботі теоретичних положень запропоновано алгоритми та розроблено програмні засоби для підвищення ефективності моніторингу та контролю за благоустроєм.

Особистий внесок здобувача. Усі наукові результати, викладені у магістерській кваліфікаційній роботі, отримані автором особисто.

Структура та обсяг роботи. Магістерська кваліфікаційна робота складається зі вступу, п’яти розділів, висновків, списку літератури, що містить 30 найменувань, 4 додатки. Робота містить 31 ілюстрацію, 15 таблиць. Ілюстративний матеріал до роботи подано у додатку Г.

1 ОБҐРУНТУВАННЯ ВИБОРУ МЕТОДУ РОЗРОБКИ ТА ПОСТАНОВКА ЗАДАЧІ ДОСЛІДЖЕННЯ

1.1 Аналіз стану систем моніторингу та контролю за благоустроєм

У минулому системи моніторингу та контролю за благоустроєм в Україні були менш розвинені та ефективні. Комунальні служби використовували традиційні методи та засоби для відстеження стану міської інфраструктури, такі як паперові звіти, ексель-таблиці та ручні огляди місцевості. Це призводило до неефективного використання ресурсів, нестабільності якості благоустрою в різних районах міста та надмірних витрат бюджетних коштів через неефективність розподілу фінансів.

З іншого боку, інформаційні технології та програмні рішення для моніторингу благоустрою почали активно розвиватися у останні десятиліття. Завдяки використанню сучасних технологій, таких як сенсори, геолокація та хмарні рішення, стало можливим відстеження стану інфраструктури у реальному часі. Розробники стали активно впроваджувати системи моніторингу та контролю, які дозволяють автоматизовано відстежувати робочі процеси, контролювати якість виконання робіт та здійснювати ефективний моніторинг витрат та використання ресурсів.

Ця перетворення відбулися завдяки швидкому розвитку технологій і зростанню інтересу до оптимізації процесів управління міською інфраструктурою. Інформаційні системи стали більш доступними та ефективними, допомагаючи владі та комунальним службам забезпечити стабільну якість благоустрою для всіх мешканців міста.

В сучасному суспільстві галузь благоустрою міст та населених пунктів є надзвичайно важливою [4]. Зростання населення, інтенсивний розвиток міських територій та збільшення обсягів будівництва призводять до ряду проблем, пов'язаних із ефективністю управління та наданням комунальних послуг. Однією з ключових проблем є нестабільність якості благоустрою в різних частинах міста.

Це може включати у себе нерівномірне розміщення парків, скверів, асфальтованих доріжок, системи освітлення та інших комунальних сервісів.

Ще однією важливою проблемою є неефективне використання ресурсів та фінансів у процесі реалізації проектів з благоустрою [5]. Це може призвести до надмірних витрат бюджетних коштів або, навпаки, до непридатності здійснених робіт через недостатні обсяги фінансування. Нерідко відбувається відсутність ефективної взаємодії між міською владою, комунальними службами та мешканцями, через що існуючі проблеми не можуть бути вирішені вчасно та належним чином.

Розробка програмної системи моніторингу та контролю за благоустроєм спрямована на вирішення цих проблем. Система буде забезпечувати можливість відстеження та аналізу реалізації проектів з благоустрою в режимі реального часу. Це дозволить ефективно спрямовувати ресурси, контролювати якість виконаних робіт та уникнути переплат за непотрібні послуги.

Система також забезпечить відкритий інтерфейс [6] для взаємодії між міською владою, комунальними службами та мешканцями міста. Це створить можливість для мешканців міста активно брати участь у формуванні та моніторингу проектів, висловлювати свої побажання та спостереження. Такий взаємозв'язок сприятиме підвищенню рівня задоволеності мешканців комунальними послугами та створить умови для розвитку відкритого та відповідального міського управління.

У зазначеному контексті виокремлюються декілька ключових проблемних питань, які виникають у галузі благоустрою міст та населених пунктів:

Нестабільність якості благоустрою: різниця в якості благоустрою між різними районами міста створює нерівномірні умови для мешканців. Це може включати недоліки в інфраструктурі, непродуману розстановку об'єктів громадського користування та недостатню якість комунальних послуг.

Неефективне використання ресурсів: недостатня координація між міською владою, комунальними службами та підрядними організаціями може призвести

до неефективного використання фінансових ресурсів, що у свою чергу може вплинути на якість виконаних робіт.

Відсутність взаємодії та залучення громади: відсутність засобів для взаємодії між місцевою владою, комунальними службами та мешканцями ускладнює залучення громади до процесів прийняття рішень та моніторингу робіт з благоустрою.

Недостатня прозорість та інформованість: мешканці міста можуть недостатньо інформовані про плани щодо благоустрою своєї місцевості, що ускладнює їхню участь у формуванні вимог та контролі за виконанням робіт.

Відсутність реального часу в моніторингу: Зазвичай інформація про роботи з благоустрою подається з певним запізненням, що ускладнює реагування на негативні явища та потребує більше зусиль для виправлення ситуації.

Непостійний контроль якості робіт: брак систематичного контролю за якістю виконаних робіт може призвести до недоліків у інфраструктурі та комунальних послугах, що може стати на шляху до задоволеності мешканців.

Ці проблеми стають основою для розробки програмної системи моніторингу та контролю за благоустроєм. Реалізація такої системи повинна адресувати ці питання, забезпечуючи ефективний контроль, взаємодію між всіма учасниками процесу, а також забезпечення відкритості та прозорості у діяльності міської влади.

Отже, розробка інформаційної системи моніторингу та контролю за благоустроєм є важливим кроком у напрямку покращення якості життя мешканців міста. Ця система надасть можливість ефективно відстежувати та контролювати реалізацію проєктів з благоустрою, забезпечуючи відкритий та прозорий процес для усіх учасників. Шляхом впровадження цієї інформаційної системи можливо зробити крок у напрямку створення міського середовища, яке відповідає сучасним стандартам життя та сприяє задоволенню потреб мешканців у комфортних умовах для життя та розвитку. Цей захід має ключове значення для формування позитивного враження про місто, створення відчуття приналежності до нього та підвищення загального рівня життя у громаді.

1.2 Порівняльний аналіз аналогів

Серед існуючих аналогів можна виділити певні платформи, які включають можливість опитування громади щодо різних аспектів благоустрою. Однак, їхні можливості часто обмежені або не охоплюють всі потреби громади в цьому питанні.

Більшість інформаційних ресурсів забезпечують можливість перегляду якогось одного аспекту громадської діяльності, але не надають повного інструментарію для комунікації з аніціаторами проектів та кінечним замовником – огранами місевого самоврядування.

Громадський бюджет [7] – це можливість для кожного мешканця ініціювати власні та голосувати за інші важливі проекти, кошти на які виділяють з місцевого бюджету. Для громадян платформа є ефективним способом подати ідею та своїм голосом особисто вирішити, на що саме спрямувати кошти місцевого бюджету.

Рисунок 1.1 – Платформа електронної демократії “Громадський бюджет”

Даний застосунок є найбільш розвинутим в сфері управління благоустроєм у громадах, що інтегрований у систему “Єдина платформа місцевої електроної демократії” та фінансується Швейцарією.

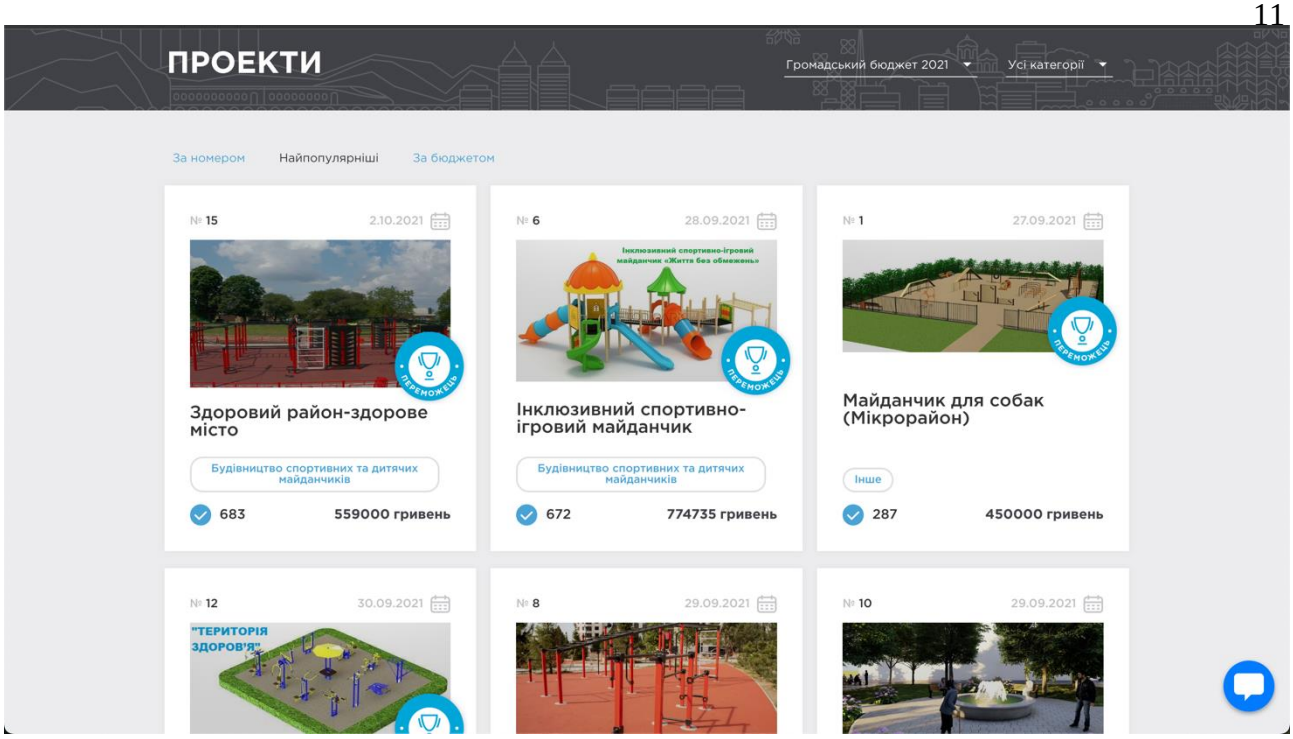

Рисунок 1.2 – Загальний список проектів

Перевагами даного інформаційного ресурсу є:

- До конкурсу залучається ширша, ніж зазвичай, аудиторія
- Платформу можна легко інтегрувати з веб-сайтами громад за допомогою віджетів
- Платформа упрозорює сам конкурс, мінімізуючи ризики некоректного голосування
- Функціонал платформи дозволяє адаптувати систему голосування до специфіки громади
- Громадяни можуть голосувати легко і комфортно, не заносячи паперових носіїв до владних установ, а робити це в зручний для себе час зі смартфона чи комп’ютера.

Недоліками даного рішення є:

- Недостатній функціонал комунікації з громадськістю, що забезпечує лише можливість голосування;

- Відсутність інтерактивної мапи для перегляду проектів на ній з можливістю огляду детальної інформації та розуміння розташування об’єктів описаних у проекті;
- Інтуїтивно складні описи проектів, без зображення графіків та інших візуально простіших форм зображення інформації;
- Не реалізовані можливості підпису проектів уповноваженими особами, що б значно пришвидшило усі етапи від подання ідеї до її реалізації.

У загальному даний додаток є найбільш підходящим аналогом розроблюваної програмної системи проте з рядом проблем, що не дають громадськості та міським органам краще комунікувати один з одним. В першу чергу це зумовлено тим, що платформа електронної демократії вміщає в себе безліч ініціатив, що не дають повною мірою розкрити кожну з них детальніше та реалізувати більш вузько направлені ідеї та функціональності.

Електронна петиція [8] – це урядовий портал, що забезпечує можливість комунікації з Офісом Президента для розгляду потрібних питань. Це особлива форма колективного звернення громадян до Президента України, Верховної Ради України, Кабінету Міністрів України, органу місцевого самоврядування, яке обов'язкове до розгляду за умови набрання необхідної кількості голосів.

Значною перевагою електронної петиції є зручність, оскільки зникає потреба фізично збирати підписи на підтримку петиції. Достатньо зареєструватись, викласти своє звернення на сайті та спостерігати за кількістю голосів, поданих на підтримку петиції. З моменту оприлюднення електронної петиції, її може прочитати та підтримати кожен громадянин. Також мають можливість ознайомитись із петицією представники ЗМІ, депутати та чиновники ще до набрання такою необхідної кількості голосів.

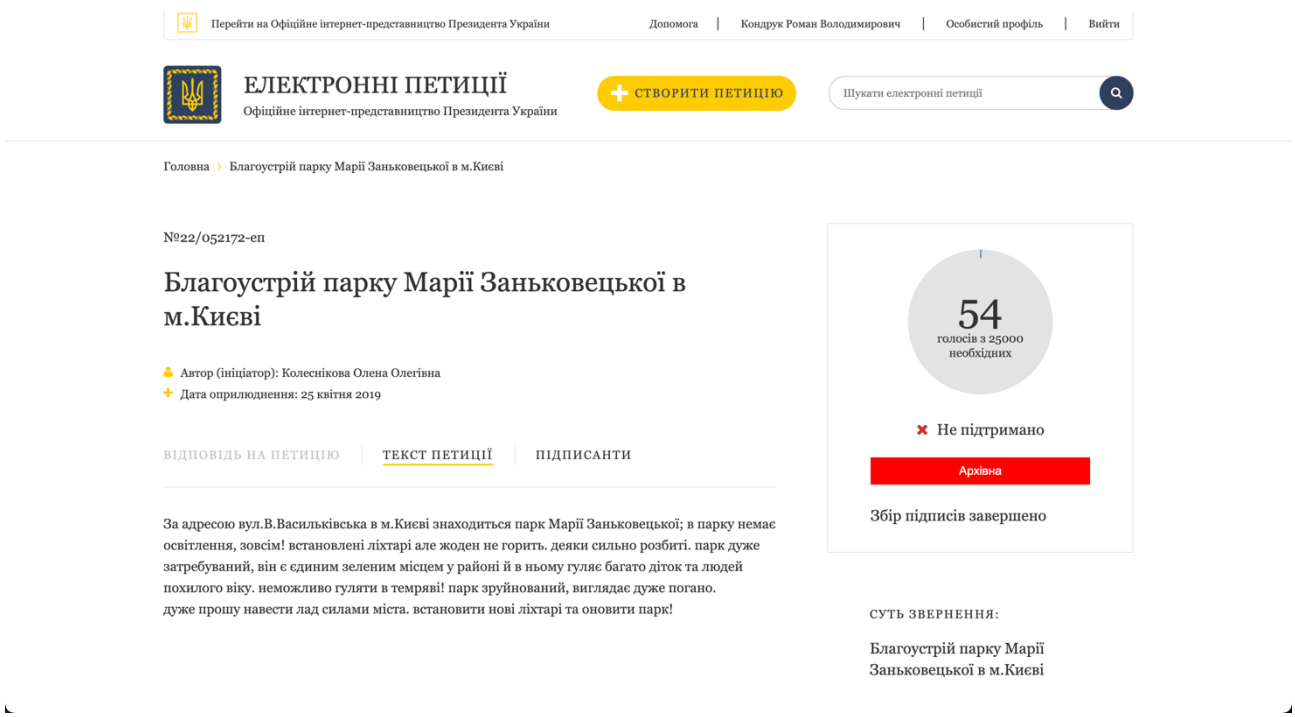

Рисунок 1.3 – Приклад петиції веб-порталу “Електронні петиції”

Даний додаток створений в першу чергу для звернення до голови держави та офісу президента, а тому не дає потрібної функціональності для вирішення питань саме міських громад. Проте він все одно є варіантом комунікації, що може дозволити у потрібний момент відмінити або ж змінити процес благоустрою певних об’єктів коли всі інші методи комунікації виявилися без успішними.

Переваги веб-додатку “Електронні петиції”:

- Охоплення величезної аудиторії громадян;
- Впевненість у вирішенні питання, адже петиція розглядається Президентом України;
- Зручність підпису петицій через додаток “ДІЯ”.

До недоліків можна віднести:

- Перенасиченість різного роду петицій, що ускладнює громадянам знайти потрібну;
- Звернення до Офісу Президента, потрібно надсилати тільки при нагальній потребі, в іншому ж випадку потрібно комунікувати з міськими органами, тому що це в межах їх компетенції;

- Недостатня інформативність петиції, що недає змоги громадянину повноцінно оглядати проекти, включаючи зображення, креселення, бюджети та інше.

Для зручності порівняння, було складено Таблицю 1.1 з результатами порівнянь.

Таблиця 1.1 – Порівняння аналогів

Критерії	Громадський бюджет	Електронні петиції	Власна реалізація
Спеціалізація на проектах благоустрою	+	-	+
Навність інтерактивкної мапи	-	-	+
Змога комунікувати з користувачами у чаті	-	-	+
Можливість пропонування змін до проекту	-	+	+
Зручність інтерфейсу	+	-	+
Швидкодія	+	+	+
Сума	3	2	6

Виходячи з існуючих аналогів можна зрозуміти доцільність реалізації програмного продукту з моніторингу та керування благоустроєм. Програмний продукт повинен вирішувати проблематику уже існуючих рішень та впроваджувати нові функціональні можливості з керування проектами по благоустрою у громадах.

1.3 Аналіз методів розв'язання поставленої задачі

Обрання веб-додатку має глибокі обґрунтування в контексті сучасних технологічних можливостей та вимог користувачів. По-перше, веб-додатки забезпечують універсальний доступ, оскільки вони можуть бути відкриті на будь-якому пристрої з доступом до Інтернету. Це робить їх доступними для широкого кола користувачів, включаючи мешканців міста, адміністрацію та комунальні служби.

Веб-додатки [9] надають можливість гнучкості та зручності в користуванні. Користувачі можуть отримувати доступ до системи моніторингу та контролю без необхідності встановлення додаткових програм на своїх пристроях. Вони можуть легко взаємодіяти з додатком через веб-браузери, що робить процес використання інтуїтивно зрозумілим та приємним.

Ще однією перевагою веб-додатків є можливість централізованого управління безпекою та оновленнями. Забезпечення безпеки важливе для захисту конфіденційності та цілісності даних користувачів. Оновлення можна легко впроваджувати безпосередньо на сервері, що гарантує, що всі користувачі мають доступ до найновіших функцій та поліпшень.

Веб-додатки відкривають широкі можливості для інтеграції з іншими сервісами та додатками. Наприклад, ми можемо легко інтегрувати систему моніторингу з соціальними мережами, що дозволить мешканцям міста взаємодіяти та ділитися інформацією щодо благоустрою.

Вибір веб-додатку для проекту є стратегічно важливим кроком [10], оскільки він гарантує максимальний охоплення аудиторії, зручність користування, безпеку даних та широкі можливості для подальшої розвитку та

інтеграції. Цей вибір покладає фундамент для створення високоефективної та інноваційної системи моніторингу та контролю за благоустроєм міста.

У веб-додатку який розробляється використано геолокацію для визначення місця положення користувача, для того аби підтвердити місто де він проживає, також використано технології Google Maps API, що надають можливість опрацювання інформації з мап.

Google Maps API [11] – це потужний інструмент, що надає безліч можливостей для розробки інтерактивних картографічних додатків та сервісів. Використовуючи цей інтерфейс програмування застосунків, розробники можуть відобразити карти безпосередньо на веб-сайтах та мобільних додатках. Однією з ключових можливостей є відображення мап, що дозволяє користувачам взаємодіяти з географічною інформацією.

Одна з важливих функцій – це геолокація, яка дозволяє отримувати точні координати місцезнаходження користувачів. Це може бути корисно для сервісів доставки, додатків навігації або будь-яких додатків, які потребують інформації про місцезнаходження.

Ще однією функцією є маршрутизація, яка дозволяє знаходити оптимальний маршрут між двома або більше точками. Це може бути використано для створення функцій навігації в додатках про подорожі або транспортних додатках.

Також Google Maps API надає можливість знаходити різні місця навколо конкретної локації, такі як ресторани, магазини або готелі. Це може бути корисно для додатків, які допомагають користувачам знаходити бажані місця в їхній околиці.

Крім того, Google Maps API дозволяє відобразити географічні дані у вигляді різних графічних елементів на карті. Це може включати в себе маркери, лінії або полігони, що дозволяє відображати різноманітні дані ефективно та інтерактивно.

Загалом, цей інструмент надає безмежні можливості для розробки додатків, які потребують географічної інформації та взаємодії з нею. Його висока

функціональність і зручність у використанні роблять його популярним вибором серед розробників.

Веб-додатки доступні з будь-якого пристрою з доступом до Інтернету. Це означає, що користувачі можуть отримати доступ до додатку з комп'ютера, смартфона або планшета незалежно від їхнього місця перебування. Веб-додатки є універсальними і можуть бути використані на будь-якій платформі, що робить їх дуже зручними для користувачів.

Веб-додатки оновлюються автоматично для всіх користувачів без необхідності їхньої установки. Розробники можуть вносити зміни, виправляти помилки і додавати нові функції без того, щоб користувачам потрібно було завантажувати та встановлювати оновлення.

Розробка таких додатків зазвичай вимагає менше часу і зусиль порівняно з розробкою мобільних додатків або інших видів програмного забезпечення. Це пов'язано з тим, що веб-додатки можуть використовувати загальні технології та бібліотеки, які спрощують процес розробки.

Загалом, розробка веб-додатків дозволяє швидко створювати, оновлювати та поширювати програми з мінімальними витратами часу та зусиль, що робить їх надзвичайно вигідними для бізнесу та користувачів.

1.4 Постановка задачі дослідження для програмної систем моніторингу та контролю за благоустроєм

Проаналізувавши питання розробки веб-додатку для моніторингу та контролю за благоустроєм, було визначено наступні завдання, які необхідно виконати для розробки програмного продукту:

– проаналізувати сучасний стан веб-додатків моніторингу та контролю за благоустроєм;

– проаналізувати існуючі алгоритми та їх реалізацію для вирішення питання знаходження місця розташування користувачів;

– розробити метод кластеризації маркерів на мапі;

– розробити алгоритми для реалізації власної програми для системи пошуку проектів з питань благоустрою;

– розробити веб-додаток для зберігання та відстеження змін в проектів з питань благоустрою на мапі;

– провести тестування розробленого програмного веб-додатку.

1.5 Висновки

У даному розділі було визначено ключові аспекти розробки програмного продукту, спрямованого на моніторинг та контроль за благоустроєм міста. Розглянуті галузі застосування продукту виявили важливість його впливу на якість життя мешканців міста та доцільного використання бюджетних коштів.

Призначення програмного продукту полягає в удосконаленні механізмів моніторингу та контролю за виконанням робіт з благоустрою, що дозволить ефективно використовувати ресурси та забезпечити якість наданих послуг.

Аналіз існуючих аналогів допоміг ідентифікувати найкращі практики та уникнути можливих помилок у розробці. Переваги та недоліки аналогів надали змогу визначити ключові вимоги до програмного продукту, зокрема, гнучкість у роботі з даними, можливість відстеження змін у режимі реального часу.

Було встановлено основні завдання, які необхідно виконати для розробки веб-додатку моніторингу та контролю за благоустроєм.

2 РОЗРОБКА МЕТОДУ ТА МОДЕЛЕЙ ПРОГРАМНОЇ СИСТЕМИ “БЛАГОМІСТ” З МОНІТОРИНГУ ТА КОНТРОЛЮ ЗА БЛАГОУСТРОЄМ

2.1 Аналіз інформаційного забезпечення

Головний функціонал програмної системи полягає в забезпечені функціональної можливості та спрощення комунікації громадян з муніципалітетами на рахунок проектів з благоустрою. Також додаток повністю проводить проект від подачі користувачем до відміни або ж виконання проекту в повному обсязі. Інтерфейс забезпечує можливість обробки інформації про проект в зручній формі, також дозволяє вносити свої корективи, голосувати за проекти і також спілкуватися на рахунок проектів.

Важливим аспектом є також перевірка поданих рекомендації до змін від користувачів, адже не можна допустити затвердження правок не умісного характеру. Для цього кожен такий запит повинен пройти верифікацію адміністратором.

Чат громадян повинен буде реалізований таким чином, щоб вони могли спілкуватися наживо, та зберігати чат для того щоб кожен міг побачити предмет дискусії.

Найважливішим аспектом також є інтерактивна мапа користувача, яка повина інформувати його про місце знаходження проектів, бути добре оптимізованою за допомогою кластеризації, щоб не було ніяких проблем з знаходженням потрібної інформації.

Для реалізації всіх необхідних функцій програмна система складається з наступних модулів:

- авторизація та реєстрація користувача;
- інтерактивна мапа проектів [12];
- створення, перегляд та редагування інформації по проектам;
- відображення запитів на зміну інформації.

2.2 Розробка загальної моделі програмної системи

Розглянемо алгоритм функціонування веб-додатку для моніторингу та контролю за благоустроєм, фокусуючись на взаємодії різних типів користувачів. Головна мета полягає в забезпеченні громадянам зручного інструменту для висловлення їхніх потреб та нагляду за проектами благоустрою.

Авторизація користувача [13] є першочерговим алгоритмом з котрим зтикається користувач при відкритті додатку. Тому варто чітко сформулювати блок-схему роботи цього алгоритму, для коректної роботи даної частини застосунку та для мінімізації зайвих рішень та помилок у проектуванні.

Користувачі, які авторизуються на платформі, отримують можливість переглядати та додавати проекти благоустрою. Після авторизації громадяни потрапляють на головну сторінку, де відображаються останні проекти громади разом із їхнім статусом. Громадяни можуть переходити на сторінку створення нового проекту, де заповнюють необхідну інформацію та прикріплюють фотографії. Також доступні сторінки для перегляду конкретного проекту та списку всіх проектів, відсортованих за статусом.

Далі важливо забезпечити можливість користувачеві змінювати інформацію саме проектів з благоустрою. Потрібно забезпечити підтримку обробки проектів від першого етапу до останього.

При виборі розробки веб-додатку, потрібно розуміти важливість правильної архітектурної розробки додатку, а саме доцільного підходу для розробки клієнтської та серверної частини.

Програмна система "БлагоМіст" використовує дворівневу архітектуру, розподіляючи функції між серверною та клієнтською частинами. На сервері використовується Node.js [14] для обробки запитів, Express.js для побудови веб-додатків, і MongoDB як база даних для зберігання та управління інформацією про благоустрій. З серверного боку відбувається опрацювання запитів, взаємодія з базою даних та надсилання відповідей клієнтам.

На клієнтській частині використовується HTML, CSS та JavaScript для візуального представлення та інтеракції з користувачем. JavaScript-бібліотеки та фреймворки (такі як React або Angular) забезпечують динамічність та ефективну організацію коду. Використовуються картографічні API (наприклад, Google Maps або Leaflet) для відображення та взаємодії з мапою. Ajax або Fetch API використовуються для асинхронного обміну даними між клієнтом та сервером без перезавантаження сторінки.

Такий підхід забезпечує чітке розділення обов'язків, полегшує масштабованість та забезпечує більшу гнучкість у розвитку та оптимізації системи. Всі частини співпрацюють для забезпечення ефективного моніторингу та керування благоустроєм.

Node.js є серверною технологією, яка дозволяє виконувати JavaScript на стороні сервера. В контексті "БлагоМіст", Node.js виступає як операційна система, яка обробляє запити від клієнтської частини, взаємодіє з базою даних та повертає результати клієнту. Він забезпечує асинхронний та подієвий підхід до обробки запитів, що є важливим для забезпечення ефективності в реальному часі.

MongoDB є документно-орієнтованою базою даних, яка забезпечує гнучкість у схемі даних. У випадку "БлагоМіст", MongoDB використовується для зберігання інформації про благоустрій, таку як дані про об'єкти благоустрою, їхні властивості та стан.

Для полегшеної роботи з базою даних також використовується Compass, що дозволяє легко маніпулювати даними та контролювати стан колекцій та документів у базі.

MongoDB Compass [15] – це графічний інтерфейс для роботи з MongoDB. Він надає зручний спосіб взаємодії з базою даних, дозволяючи адміністраторам та розробникам візуально досліджувати дані, створювати та виконувати запити, візуалізувати схеми, та взагалі полегшує управління та роботу з MongoDB. З його допомогою можна виготовляти складні запити, стежити за виконанням операцій та взагалі спрощувати роботу з базою даних MongoDB.

2.3 Аналіз принципів кластеризації маркерів на мапі

Кластеризація [16] – це статистична процедура, задача якої полягає в розбитті вибірки об'єктів на підмножини, що не перетинаються і називаються кластерами. Кожен кластер має складається зі схожих об'єктів, а об'єкти різних кластерів мають істотно відрізнятися один від одного. Задача кластеризації відноситься до статистичної обробки, а також до широкого класу задач навчання без вчителя. Головна ідея кластеризації полягає в об’єднані об’єктів за певними критеріями та за допомогою найбільш швидкого та підходящого алгоритму об’єднання. Так критерієм для об’єднання маркерів виступають координати маркерів та відстані між ними. Далі потрібно сформувати групи маркерів об’єднані за даним критерієм та зобразати їх на мапі.

Для найшвидшої та найбільш точної кластеризації варто використати метод k-середніх, що є найпопулярнішим та найшвидшим.

Головною особливістю алгоритму є процес визначення відстаней між точками та метод визначення центрів кластерів.

Алгоритм k-середніх [17] використовує Евклідову відстань для знаходження дистанцій. Проте, потрібно провести аналіз інших методів для розуміння проблематики даного підходу.

Серед основних методів варто виділити: Евклідову відстань, Манхетенську відстань, та відстань Хаверсайн.

Евклідова відстань [18] – це відстань між двома точками в евклідовому просторі. Вона базується на геометричних принципах евклідової геометрії, і вимірює пряму лінію між двома точками на площині або в просторі. Цей підхід особливо корисний, коли важлива "пряма" відстань між двома точками, і він широко використовується в різних галузях, таких як комп'ютерна графіка, статистика, машинне навчання та інші області.

Манхеттенська відстань [19] (також відома як L1-відстань, метрика Манхеттена або таксі-відстань) – це метрика в евклідовому просторі, яка вимірює абсолютну різницю між координатами двох точок на площині.

Назва "Манхеттенська відстань" походить від схожості з відстанню, яку потрібно подолати вулицями Манхеттена в Нью-Йорку, де можна рухатися тільки вздовж паралельних або перпендикулярних вулиць, а не по прямих лініях.

Ця метрика широко використовується в задачах розпізнавання образів, кластеризації даних та інших областях, де важливо враховувати "поодинокі" рухи вздовж координатних вісей, а не діагональні рухи.

Відстань Хаверсайн [20], також відома як гаванська відстань, є методом обчислення відстані між двома точками на поверхні сфери, такої як Земля. Цей метод особливо корисний для обчислення географічних відстаней, коли маємо координати точок задані в широті та довготі.

Відстань Хаверсайн дозволяє досить точно визначити географічні відстані між точками на сфері і використовується в геолокації, картографії та інших областях, де важливо враховувати форму сфери при обчисленнях.

Для проведення аналізу складено Таблицю 2.1.

Таблиця 2.1 – Порівняльний аналіз алгоритмів визначення відстаней

Критерій порівняння	Евклідова відстань	Відстань Хаверсайн	Манхеттенська відстань
Ефективність в міському плануванні	-	+	+
Швидкість роботи алгоритму	+	-	+
Простота обчислень	-	+	+
Наявність прямих ліній в плануванні маршруту	+	-	+

Виходячи з порівняльного аналізу стає зрозуміло, що доцільно замінити метод Евклідової відстані на Манхетенську.

Також варто покращити алгоритм визначення центрів кластерів і виконувати їх відбір за допомогою визначення найбільш віддалених точок на мапі, що дасть початковий список маркерів, що з більшою вірогідністю будуть підпадати під окремий кластер, це дасть змогу зменшити кількість операцій для знаходження потрібних маркерів для кластеризації.

З урахуванням вище приведених особливостей розроблюваної системи алгоритм кластеризації для системи моніторингу та контролю за благоустроєм розробляється з дотриманням таких принципів:

- зформувати кластер за коорденатами маркера;
- використовувати Манхетенську відстань для визначення відстаней між точками;
- оптимізувати процес вибору центрів кластерів;
- діставати координати через GPS модуль;
- мінімізувати лишні обрахунки;
- не зберігати лишньої інформації через нагромадженість;
- врахувати розмір девайсу для коректного відображення інформації;

2.4 Розробка алгоритму кластеризації для програмної системи моніторингу та контролю за благоустроєм

Для коректної роботи та розробки алгоритму користувач повинен надати координати точки де саме він хоче ініціювати проведення робіт з благоустрою. Дуже важливо коректно дістати координати користувача використавши GPS модуль телефону з збільшеною чіткістю геоданих. Це надасть максимальну точність та дозволить коректно опрацювати дані за допомогою алгоритму.

Для того аби додати маркер на мапу користувач в першу чергу має пройти реєстрацію, та отримати підтвердження місцезнаходження. Це виконується задля перевірки того чи користувач є жителем конкретної громади та може приймати участь в опитуваннях.

Впровадження кластеризації маркерів на картах [21] веб-додатків виявляється критично важливим для покращення їхньої візуальної доступності та ефективності. Особливо це стає актуальним у випадках, коли на карті присутня значна кількість маркерів, особливо у регіонах із високою щільністю об'єктів. Некластеризовані дані можуть призвести до перегруження інтерфейсу та ускладнення сприйняття користувачем інформації.

Ефективна кластеризація дозволяє покращити загальну продуктивність додатка, надаючи оперативний відгук на дії користувача та підвищуючи швидкість взаємодії. Швидка обробка та відображення кластерів сприяє зменшенню обсягу передаваних та оброблюваних даних, що важливо для оптимізації трафіку та ресурсів. Швидкий відгук на взаємодію користувача, такий як переміщення по карті чи зміна масштабу, покращує загальний користувацький досвід та надає більш зручну навігацію. Швидка обробка кластерів сприяє ефективному використанню ресурсів браузера, що є критичним для забезпечення стабільної роботи на різних пристроях.

Прискорення алгоритмів кластеризації сприяє оптимізації відображення маркерів на карті, забезпечуючи кращий інтерфейс та покращений користувацький досвід.

Для роботи алгоритму кластеризації потрібно використати Манхетенську відстань для визначення відстаней між точками.

Манхеттенська відстань, також відома як L1-відстань, манхеттенська метрика або таксі-відстань, вимірює відстань між двома точками на площині, як суму абсолютних різниць їхніх координат. Цей метод отримав назву "манхеттенська" через те, що він відповідає відстані, яку вам потрібно подолати, пересуваючись по вулицях Манхеттена, якщо можна рухатися тільки вздовж паралельних або перпендикулярних вулиць.

Для двох точок з координатами (x1, y1) та (x2, y2) манхеттенська відстань обчислюється за формулою:

$$d_1(\mathbf{p}, \mathbf{q}) = \|\mathbf{p} - \mathbf{q}\|_1 = \sum_{i=1}^{n} |p_i - q_i|, \tag{2.1}$$

Це визначає суму абсолютних різниць між відповідними координатами точок по осях x та y. Геометрично, це відстань, яку ви пройдете, пересуваючись вздовж сітки вздовж горизонталей та вертикалей.

Манхеттенська відстань може бути особливо корисною в задачах, де важливо враховувати тільки "поодинокі" рухи по координатних вісях, а не діагональні рухи, які враховуються, наприклад, евклідовою відстанню.

Тому розроблюваний алгоритм кластеризації використовує Манхетенську відстань, що дає можливість швидше маніпулювати даними маркерів.

Основна ідея алгоритму полягає у визначенні n центрів кластерів та розподілі точок даних до найближчого кластеру на основі евклідової відстані. Процес виконується ітеративно з оновленням центрів та перерозподілом точок до кластерів до тих пір, поки центри кластерів не стабілізуються або досягнуто максимальну кількість ітерацій.

Кластеризація даним методом [22] – це впорядкування множини об'єктів у порівняно однорідні групи.

Цей алгоритм ефективний для групування даних та виділення паттернів, а його робота базується на пошуку оптимальних центрів кластерів, що найкращим чином відображають структуру вхідних даних.

Мета методу – розділити n спостережень на k кластерів, так щоб кожне спостереження належало до кластера з найближчим до нього середнім значенням. Метод базується на мінімізації суми квадратів відстаней між кожним спостереженням та центром його кластера, тобто функції.

У початковий момент роботи алгоритму довільним чином обираються центри кластерів, далі для кожного елемента множини ітеративно обраховується відстань від центрів з приєднанням кожного елемента до кластера з найближчим центром. Для кожного з отриманих кластерів обчислюються нові значення центрів, намагаючись при цьому мінімізувати функцію, після чого повторюється процедура перерозподілу елементів між кластерами.

- вибрати n інформаційних точок як центри кластерів поки не завершиться процес зміни центрів кластерів;
- зіставити кожну інформаційну точку з кластером, відстань до центра якого мінімальна;
- переконатися, що в кожному кластері міститься хоча б одна точка. Для цього кожний порожній кластер потрібно доповнити довільною точкою, що розташована «далеко» від центра кластера;
- центр кожного кластера замінити середнім від елементів кластера;
- кінець.

Відповідно заданій формулі (2.2) можна провести кластеризацію точок на мапі найшвидшим шляхом.

$$V = \sum_{i=1}^{k} \sum_{x_j \in S_i} (x_j - \mu_i)^2 \tag{2.2}$$

Головним недоліком алгоритму є сильна залежність від точності початкових даних, що може призвести до недоцільних об’єднань в такому випадку, але головною перевагою є саме швидкість обробки, що є пріоритетним параметром даного алгоритму.

Також для пришвидшеної роботи алгоритму було оптимізовано метод визначення центрів кластерів для алгоритму k-середніх, так для швидшого вибору маркерів використано процес визначення найбільш віддалених точок, що з більшою вірогідністю є центрами власних кластерів через віддаленість (Формула 2.3).

$$\sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \ldots + (p_n - q_n)^2} = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2}. \tag{2.3}$$

2.5 Розробка моделі програмної системи “БлагоМіст”

Задля повного зображення функціональних можливостей додатку було створено UML діаграму прецедентів, що висвілтлює функціональні можливості користувача.

UML (Unified Modeling Language) діаграма [23] – це графічний інструмент для візуалізації, проектування та документування структури та поведінки системи. UML є стандартом, який використовується для моделювання програмних систем та інших видів систем і процесів.

UML діаграми є потужним інструментом для моделювання програмних систем та володіють декількома ключовими перевагами.

Вони дозволяють візуалізувати структуру системи, представляючи класи, об'єкти, компоненти та їхні зв'язки. Це полегшує розуміння архітектури програми.

Діаграми UML також використовуються для моделювання поведінки системи, що допомагає визначити взаємодію об'єктів та процесів.

Цей інструментарій важливий для аналізу та проектування систем. Він дозволяє розробникам визначити вимоги до системи та створити моделі, які можна адаптувати до конкретних потреб.

Діаграми UML служать також засобом документування, надаючи структуровані та зрозумілі зображення для різних аспектів програмного продукту.

Цей інструмент полегшує комунікацію між різними учасниками проекту, створюючи спільну мову для обговорення та узгодження планів.

Також, UML діаграми використовуються для тестування та супроводу систем, допомагаючи створювати тестові сценарії та визначати вимоги до тестування. Це також сприяє підтримці та розумінню системи на етапі супроводу. Загалом, вони є корисним інструментом для розробників та всіх учасників проекту.

Так як веб-застосунком може користуватися як звичайний користувач так і адміністратор. Це додає певну проблемність розуміння архітектурних можливостей користувачів. Тому потрібно чітко зобразити та описати ролі та наявні інструменти для роботи з певними аспектами програми.

Модель програмної сестеми “БлагоМіст” для моніторингу та контролю за благоустроєм за допомогою діаграми прецедентів зображено а рисунку 2.1.

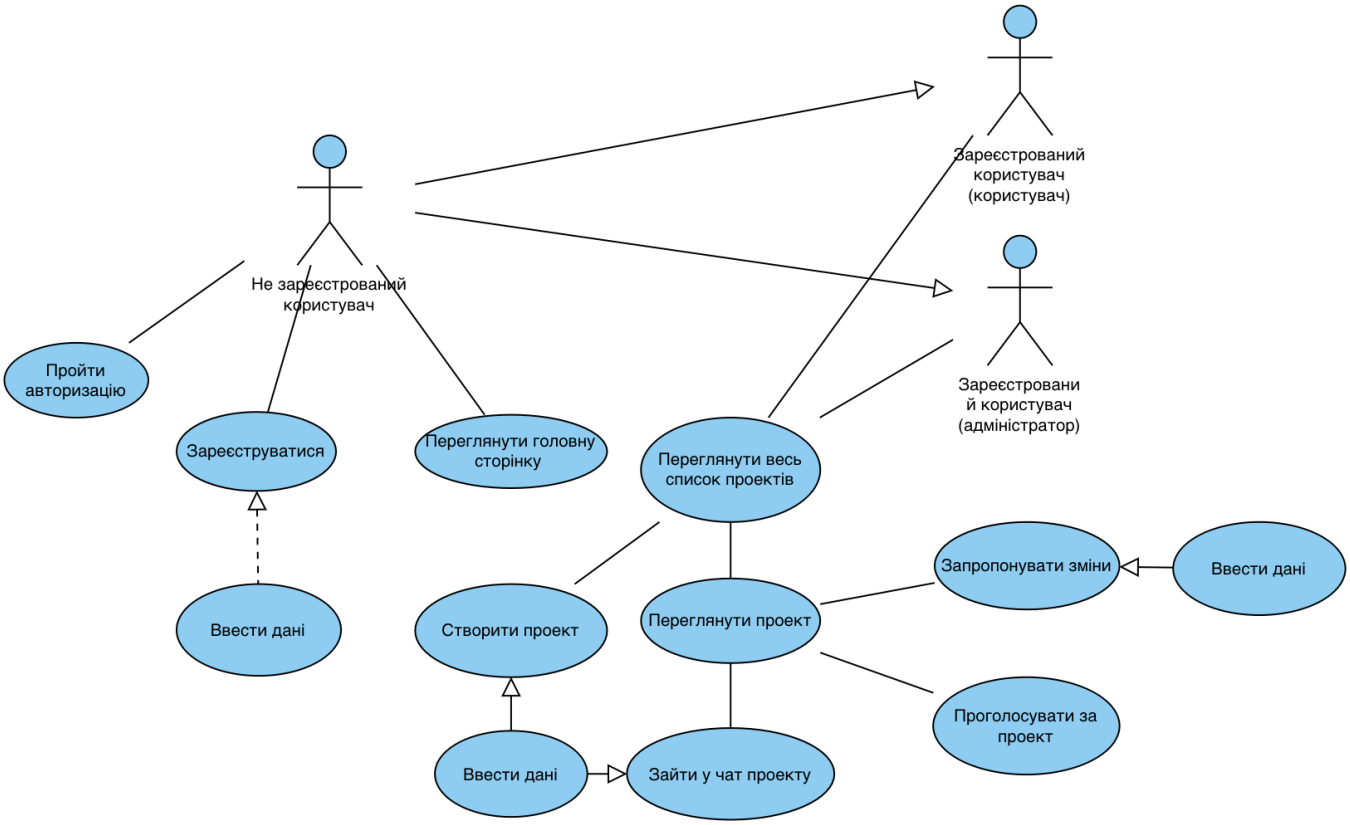

Рисунок 2.1 – Модель програмної системи у вигляді діаграми прецедентів

Перелік варіантів використання та їх короткий опис показано в таблиці 2.2.

Таблиця 2.2 – Реєстр варіантів використання

Актор	Прецедент	Опис
Неавторизований користувач	Пройти авторизацію	Введення логіну та паролю для здійснення входу в навчальну систему.
	Зареєструватися	Введення та збереження особистих даних нового користувача для подальшої можливості авторизації.
	Переглянути головну сторінку	Можливість перегляду головної сторінки з основною інформацією

Продовження таблиці 2.2.

Авторизований користувач (користувач або ж адміністратор)	Переглянути весь список проектів	Зображення списку всіх проектів громади.
	Переглянути проект	Перегляд інформації по конкретному проекту
	Створити проект	Можливість заповнити інформацію та створити проект з благоустрою
	Зайти у чат проекту	Можливість спілкуватися з користувачами на рахунок проекту
	Запропонувати зміни	Можливість надіслати запит на зміну певної інформації програми
	Проголосувати за проект	Можливість віддати голос за реалізацію певного проекту

Головний функціонал програмної системи “БлагоМіст” для моніторингу та керування благоустроєм полягає в покращені комунікації громадян з муніципалітетами та падтримки проектів з благоустроб на всіх етапах.

2.6 Розробка моделі інтерфейсу користувача та вибір дизайну додатку

Інтерфейс додатку розробляється з метою забезпечення зручності та ефективності користувачам. Він повинен бути інтуїтивно зрозумілим, естетично приємним та легким у використанні. Додаток матиме сучасний та привабливий дизайн, що залучатиме користувачів і сприятиме їхній активності. Ось опис основних елементів інтерфейсу:

1. Головна сторінка:

При вході на головну сторінку користувач бачитиме карусель із заголовними проектами, що актуальні для його регіону. Кожен проект супроводжуватиметься зображенням, назвою та коротким описом. На головній сторінці також буде кнопка "Створити Проект", яка веде на сторінку створення нового проекту.

2. Профіль користувача:

Особистий кабінет користувача буде містити інформацію про його проекти, взаємодії та активності. Тут користувач може змінювати свої дані, переглядати список своїх проектів, отримувати сповіщення та взаємодіяти з іншими користувачами.

3. Сторінка проекту:

Кожен проект має свою окрему сторінку, де розміщена детальна інформація про нього. Ця сторінка містить фотографії, опис, місцезнаходження на мапі, коментарі та рейтинг. Тут користувач може приєднатися до проекту, залишити коментар, поділитися ним у соціальних мережах тощо.

4. Мапа проектів:

Сторінка з мапою дозволяє користувачам переглядати усі проекти на карті. Проекти позначаються маркерами з короткою інформацією. Користувач може клікнути на маркер, щоб перейти на сторінку проекту.

5. Форма створення проекту:

Форма створення проекту містить поля для введення назви, опису, вибору категорії, завантаження фотографій, вибору місцезнаходження на мапі. Користувач може також додати теги, які описують проект.

6. Система Сповіщень:

Сповіщення виводяться у вигляді спливаючих вікон або повідомлень в особистому кабінеті. Вони повідомляють про нові коментарі, приєднання до проекту, зміни в статусі проекту тощо.

7. Фільтри та Сортування:

На сторінці з проектами користувач може застосовувати фільтри за категоріями, статусом та іншими параметрами. Також є можливість сортувати проекти за рейтингом.

Першою сторінкою, що зустрічає користувача є “Головна” саме тому, проектування варто починати саме з неї. (Рисунок 2.2).

1. Стрічка навігації;
2. Робоча область додатку;
3. Мапа населеного пункту;
4. Список останніх проектів громади;

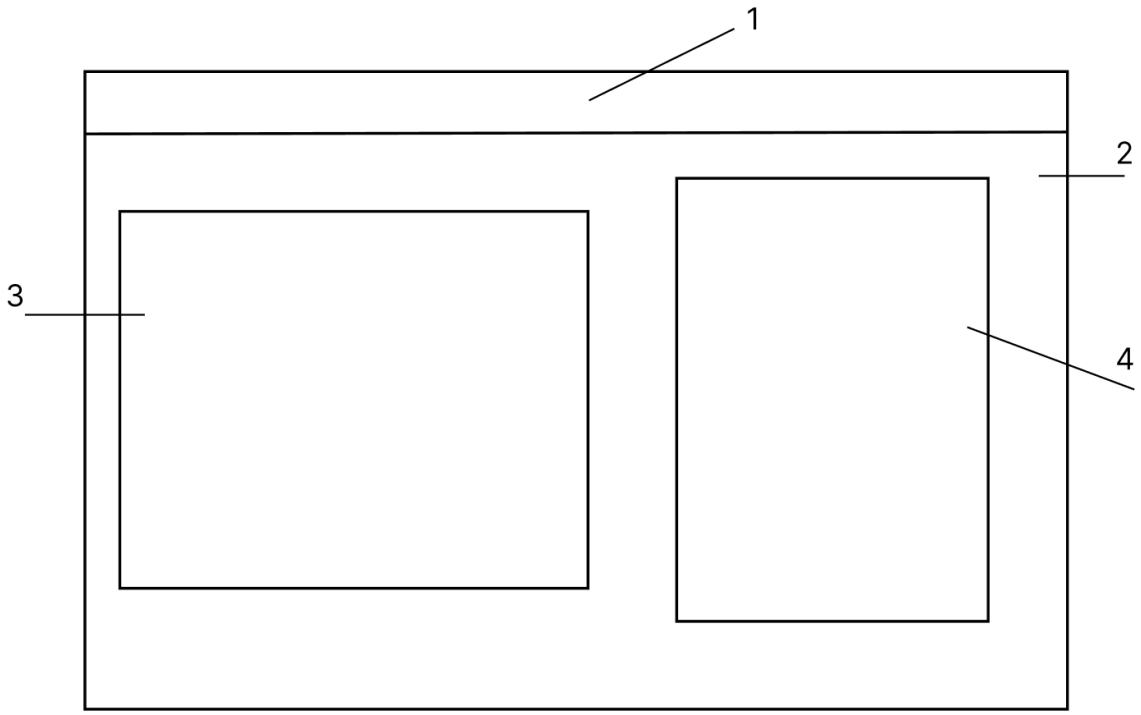

Рисунок 2.2 – Інтерфейс користувача сторінки “Головна”

Далі варто відзначити сторінку профілю користувача яка має забезпечити зручний інтерфейс для маніпуляції даними профілю. Повинно бути реалізовано можливість зміни паролю за допомогою листа на електронну пошту, мають бути інструменти для зміни особистої інформації користувача (Рисунок 2.3).

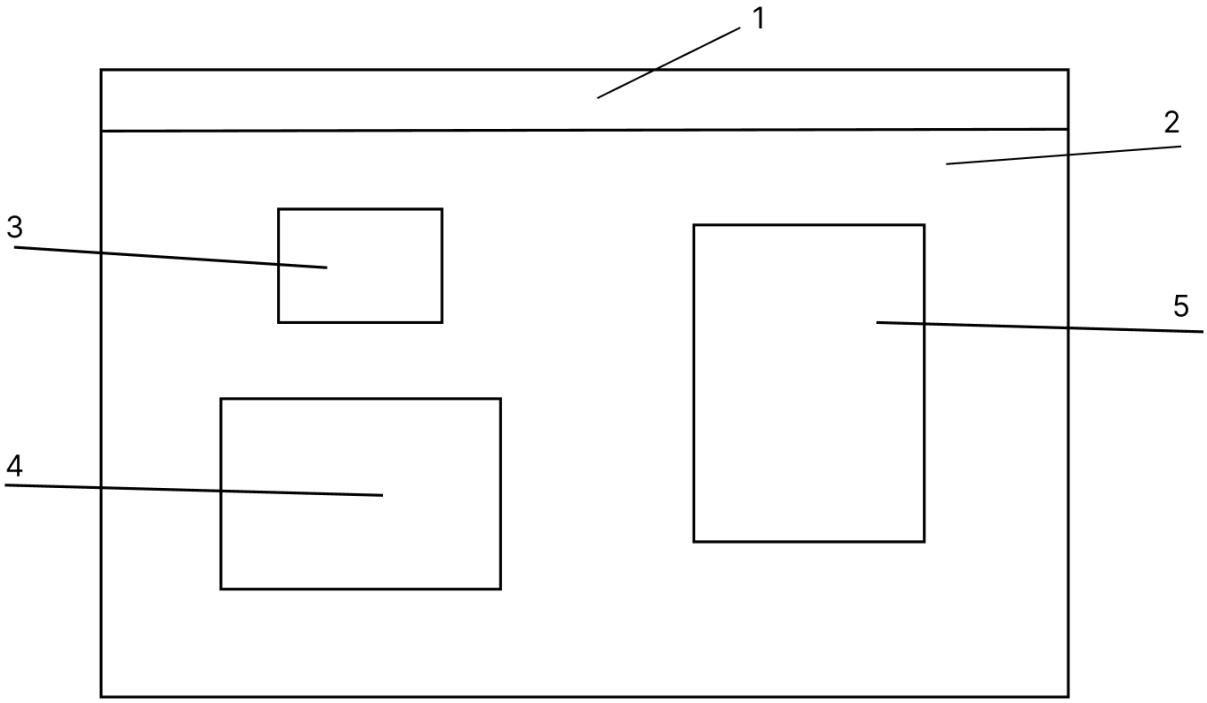

Рисунок 2.3 – Інтерфейс користувача сторінки “Профіль”

1. Панель навігації;
2. Робоча область додатку;
3. Зображення користувача з можливістю його зміни;
4. Опис основної інформації користувача з можливістю його зміни;
5. Список проектів в голосуванні за які користувач приймав участь.

Наступною важливою сторінкою застосунку є сторінка проекту. Це сторінка, що надає велику кількість інформації для користувача а також дає можливість користувачу змінювати певну інформацію, пропонувати зміни у проекті та просто спілкуватися у чаті проекту (Рисунок 2.4).

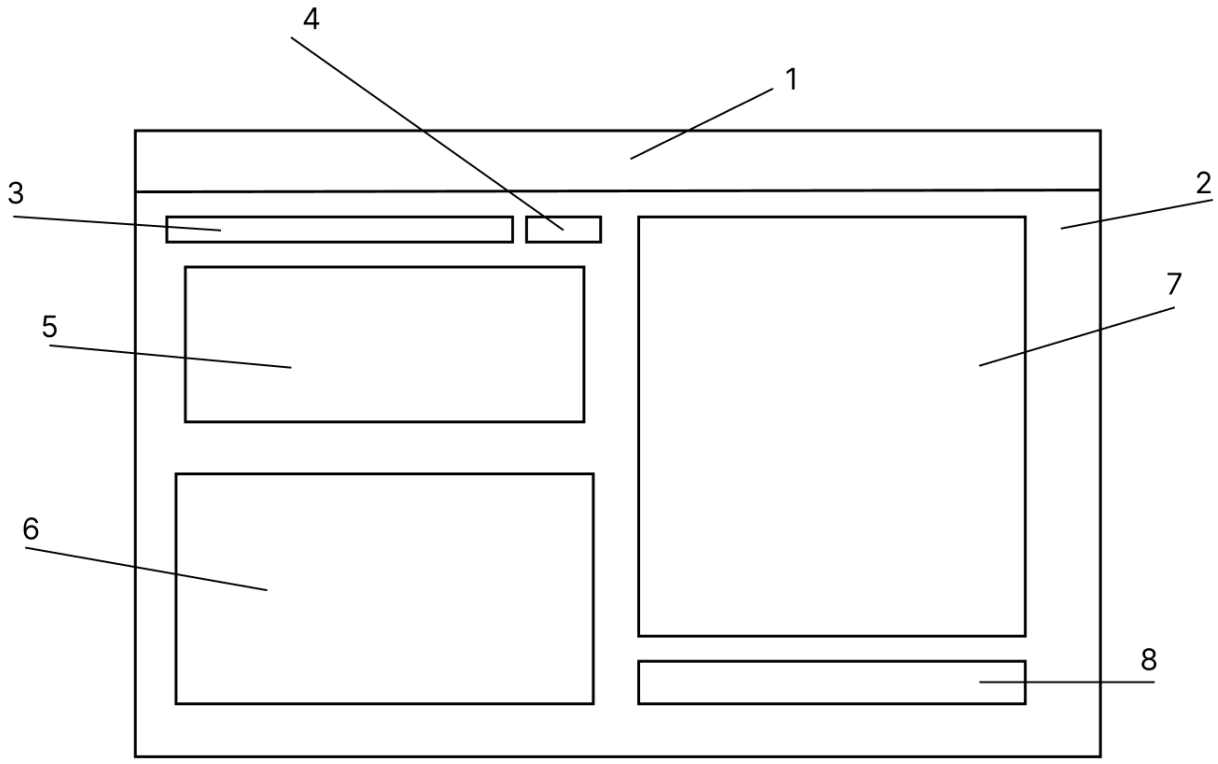

Рисунок 2.4 – Інтерфейс користувача сторінки “Профіль проекту”

1. Панель навігації;
2. Робоча область додатку;
3. Назва проекту;
4. Стан проекту;
5. Зображення у вигляді каруселі;
6. Чат для обговорення проекту;
7. Опис проекту та список пропозицій на доопрацювання;
8. Блок голосування за проект чи пропозицій по змінам;

Дані основні сторінки застосунку надають функціональність, що забезпечує повноцінну роботу додатку та реалізує головні особливості сайту. Також було змодельовано і інші діаграми інтерфейсів користувача, що наведені у Додатку Б.

Створення користувацького інтерфейсу - це ключовий аспект, що визначає зв'язок між додатком та користувачем. Правильно розроблений дизайн може

стати важливим інструментом для залучення та утримання клієнтів, адже він впливає на сприйняття та задоволення користувачів.

Існують конкретні рекомендації та принципи, які допомагають досягти успішного дизайну інтерфейсу. Усі елементи повинні бути інтуїтивно зрозумілими: їх зовнішній вигляд повинен бути простим і передавати їхню функціональність. Розміщення елементів також повинно бути в очікуваних місцях, щоб користувач не виникало сумнівів при взаємодії з системою. Елементи повинні логічно пов'язуватися між собою, а розташування їх не повинно утворювати довгих ланцюжків. Забезпечення читабельності сторінки - ще один ключовий аспект дизайну, при якому важливо дотримуватися контрасту між фоном та текстом.

Важливо також враховувати естетичність використання дизайну, оскільки приємний зовнішній вигляд може поліпшити враження від використання системи. Забезпечення гармонії між виглядом та функціональністю елементів дизайну визначає успіх дизайну інтерфейсу.

При виборі кольорової гами важливо раціонально підходити до цього питання, враховуючи психологічний вплив кольору на людину. Кольори можуть викликати певний спектр емоцій і впливати на сприйняття додатку. У даному випадку, оскільки система призначена для навчання основам кулінарного мистецтва, вибір нейтральної теми відтінків синього кольору може створити спокійну та робочу атмосферу. Однак, слід підібрати відтінки так, щоб не перебільшувати з їхнім використанням та уникати втоми користувача.

Таким чином, можна розглянути інші варіанти кольорових схем, враховуючи основні принципи контрастності, естетичності та психологічного впливу кольорів (Рисунок 2.5).

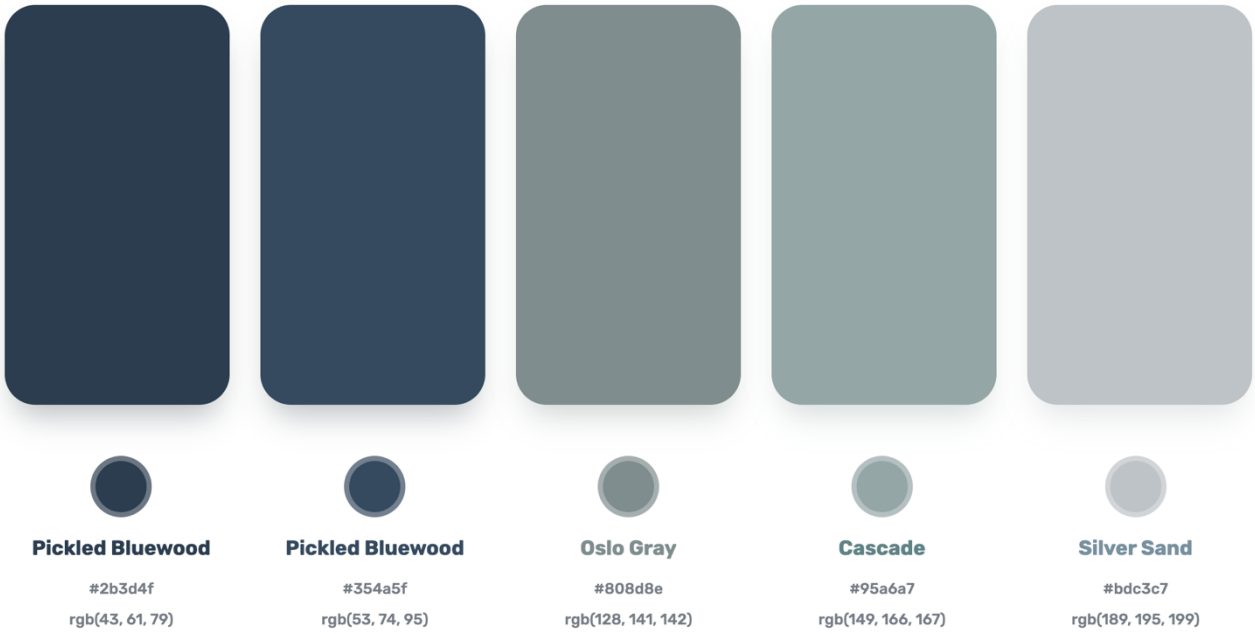

Рисунок 2.5 – Кольори додатку

Таким чином, були вибрані основні кольори для блоків та кнопок у відтінках синього кольору та сірого кольорів. Фоновий колір визначено як темно-синій. Щодо кольору тексту, обрано сірий, що гарантує високу читабельність через використання контрасту.

2.7 Розробка алгоритмів роботи додатку

Загальний алгоритм додатку є ключовим елементом, що визначає його функціональність, продуктивність та користувацький досвід. Він розглядає всі можливі сценарії використання, реакції на користувацькі взаємодії та інші аспекти роботи програми.

1. Загальний вигляд та інтерфейс:
- Додаток відкривається на головній сторінці, де користувач бачить основні функції та можливості. Інтерфейс додатку повинен бути інтуїтивно зрозумілим і зручним для користувачів.
2. Авторизація та реєстрація:

- Користувач може авторизуватися або зареєструватися, якщо він новий. Процес реєстрації повинен бути простим та швидким, з мінімальною кількістю обов'язкових полів.

3. Особистий кабінет:
- Після авторизації користувач має доступ до особистого кабінету, де він може переглядати і змінювати свої дані, додавати нові проекти, переглядати статус існуючих проектів та отримувати сповіщення.

4. Створення та управління проектами:
- Користувач може створювати нові проекти з благоустрою, вказуючи їхні деталі та область реалізації. Також він може переглядати і долучатися до існуючих проектів у своєму місті або районі.

5. Взаємодія та комунікація:
- Додаток повинен забезпечити можливість взаємодії між користувачами, коментування та підтримку спільної роботи над проектами. Це може включати систему коментарів, приватні повідомлення та можливість ділитися проектами у соціальних мережах.

6. Використання GPS та Google Maps API:
- Додаток використовує GPS технологію для визначення місцезнаходження користувача та Google Maps API для відображення проектів на карті. Це надає можливість точно визначати місцезнаходження проектів та знаходити їх легко на мапі.

7. Система рейтингів та відгуків:
- Користувачі можуть залишати відгуки та ставити рейтинги проектам. Це сприяє залученню уваги до найбільш важливих проектів та формує довіру до їхньої якості.

8. Система сповіщень:

- Додаток надсилає сповіщення користувачам про зміни в їхніх проектах, нові коментарі

Далі подано загальний алгоритм додатку для кращого розуміння та реалізації поставленої задачі (Рисунок 2.6).

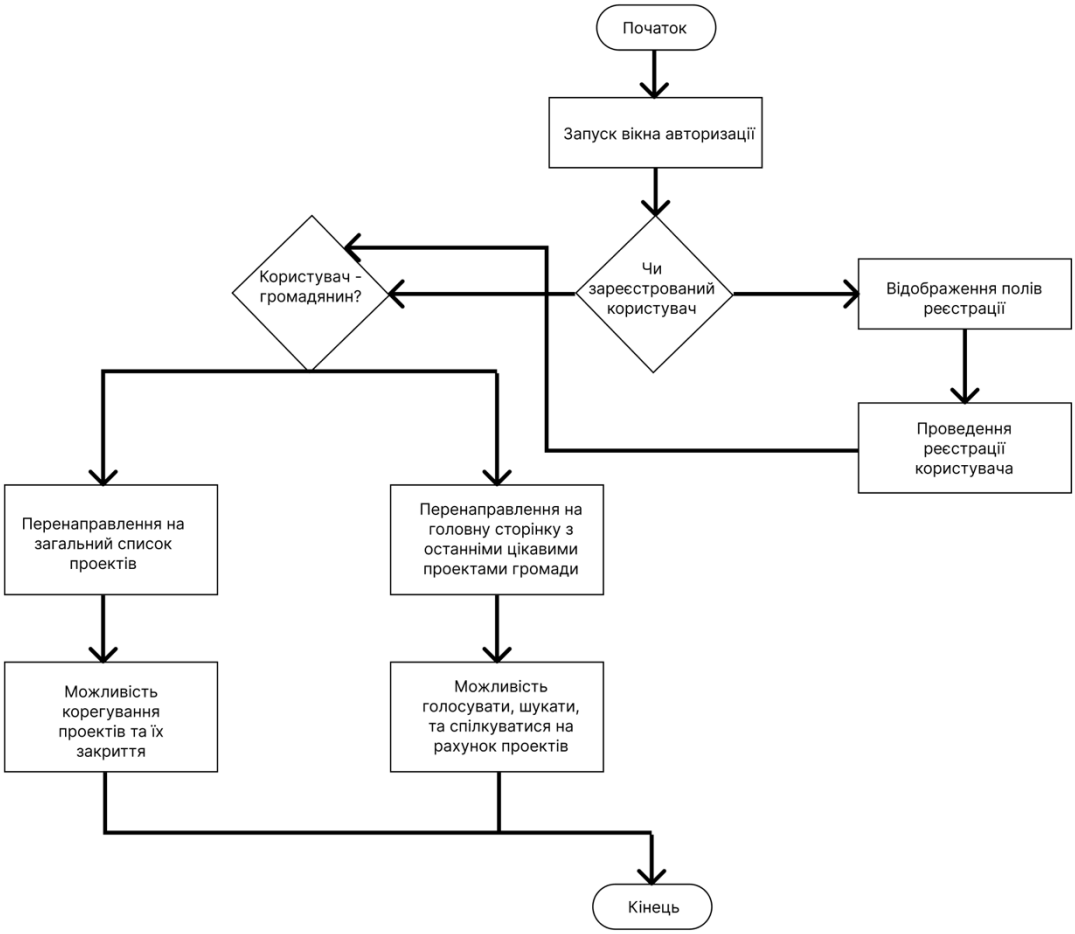

Рисунок 2.6 – Загальний алгоритм додатку

Через сильну нагромадженість функціоналу, також варто провести аналіз та змоделювати діаграми станів окремого модулю додатку, а саме модулю

‘Опрацювання проекту’. Цей модуль повинен забезпечувати максимальну функціональність та має бути інтуїтивно зрозумілий та без додаткової пояснювальної інформації давати зрозуміти користувачу про стан модулю в конкретний проміжок часу (Рисунок 2.7).

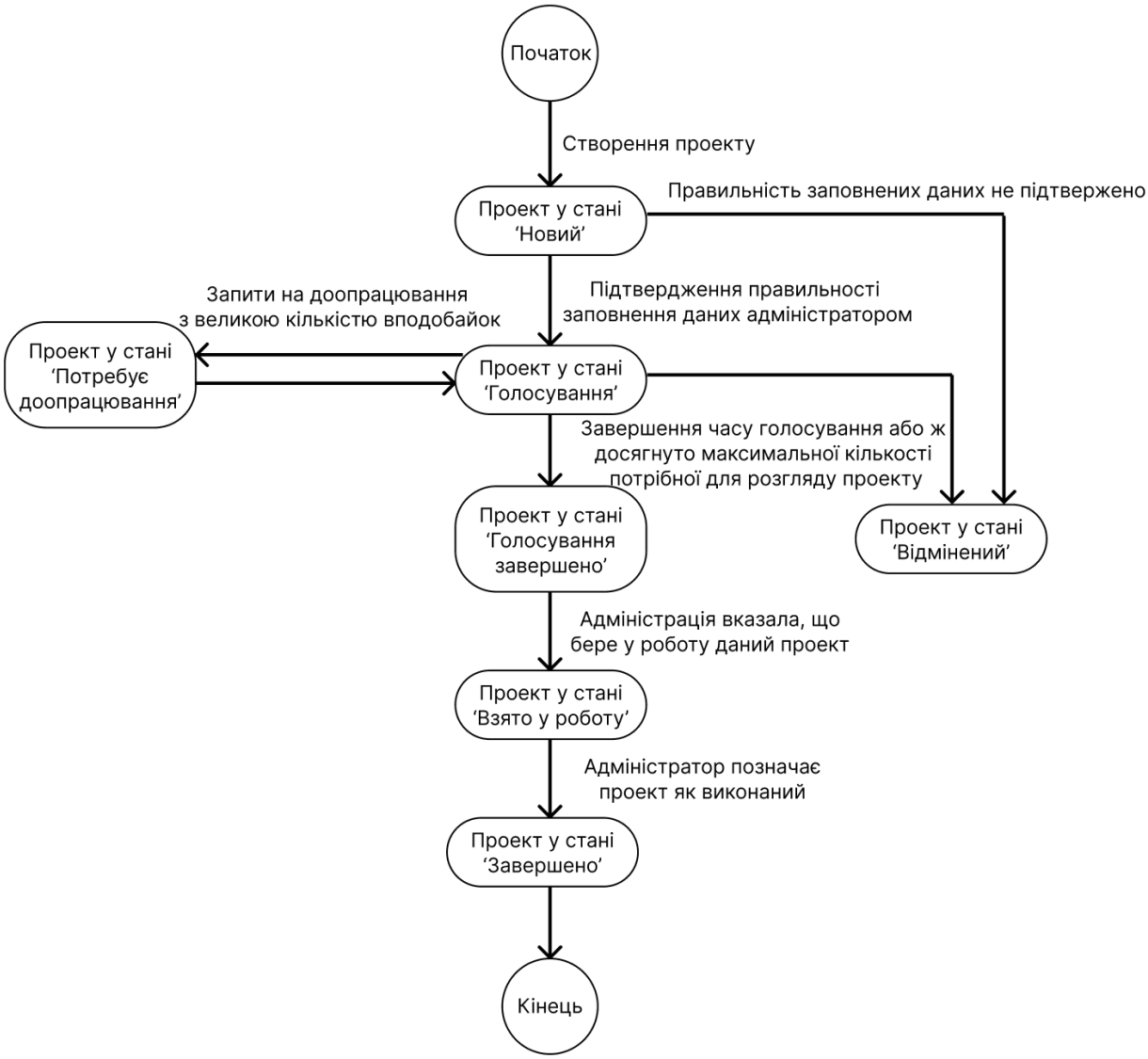

Рисунок 2.7 – Діаграма стану проекту

В результаті було отримано діаграму станів моделі Проекту, яка доступно та чітко описує всі можливі стани даної сутності. Це у подальшому допоможе чітко сформувати потрібний інтерфейс користувача для вирішення поставлених задач.

2.8 Висновки

У даному розділі було ретельно розроблено алгоритм функціонування додатку та спроектовано інтерфейс користувача з урахуванням найсучасніших технологій та потреб користувачів. Алгоритм додатку був створений з урахуванням всіх можливих сценаріїв використання, що дозволяє користувачам легко та ефективно взаємодіяти з платформою.

Основні моменти алгоритму включають в себе зручний процес авторизації, можливість створення, перегляду та редагування проектів, взаємодію з іншими користувачами та реагування на сповіщення. Застосування GPS та Google Maps API дозволяє точно визначати місцезнаходження проектів та відображати їх на мапі, забезпечуючи користувачам зручний інструмент для навігації та пошуку проектів у їхній області.

Щодо інтерфейсу користувача, було приділено особливу увагу його інтуїтивності та зручності. Користувачі зможуть легко зорієнтуватися в додатку завдяки зрозумілій навігації, яка включає в себе головну сторінку з актуальними проектами, особистий кабінет для управління власними проектами та взаємодії з іншими користувачами, сторінку проекту для детального перегляду інформації, мапу проектів для швидкого знаходження потрібних проектів, а також форму створення нового проекту для швидкого додавання власних ініціатив.

У результаті роботи над алгоритмом додатку та його інтерфейсом було досягнуто гармонійного злагодження функціональності та зручності використання, що є ключовим фактором для успіху платформи серед користувачів.

3 РОЗРОБКА ПРОГРАМНИХ КОМПОНЕНТ ДОДАТКУ

3.1 Аналіз і обґрунтування вибору засобів для реалізації програмного засобу

У цьому розділі магістерської роботи буде проведено обгрунтування вибору програмних засобів та методів, які будуть використані при розробці програмної системи моніторингу та контролю за благоустроєм. Аналіз вибору мов програмування, методів розробки, архітектурних підходів, а також вибір бази даних є важливою частиною роботи, оскільки від цих рішень залежить якість та ефективність розробленої програмної системи. В розділі буде детально розглянуто причини вибору конкретних технологій, а також їх переваги у контексті вирішення завдань, поставлених перед системою моніторингу та контролю за благоустроєм.

В сучасному світі веб-розробки існує багато мов програмування, які можна використовувати для створення веб-сайтів та додатків. Однак серед усіх цих варіантів JavaScript виокремлюється як одна з найпоширеніших, потужних і універсальних мов програмування в галузі веб-розробки.

JavaScript [24] – це високорівнева, інтерпретована мова програмування, яка використовується для створення веб-сайтів і веб-додатків. Вона є однією з найпоширеніших мов програмування у веб-розробці і дозволяє розробникам створювати динамічні, інтерактивні веб-сторінки.

Однією з ключових переваг JavaScript є його вбудовані можливості у всіх сучасних веб-браузерах, що означає, що користувачі можуть запускати JavaScript-код безпосередньо на своїх пристроях, не потрібно встановлювати додаткові плагіни або розширення.

JavaScript є мовою програмування, яка дозволяє створювати динамічні зміни на веб-сторінках без перезавантаження сторінки. Це дозволяє реалізувати різноманітні функції, такі як валідація форм, завантаження даних без перезавантаження сторінки (асинхронний запит), анімація та багато інших інтерактивних можливостей, які забезпечують кращий користувацький досвід.

JavaScript також використовується для розробки серверних додатків за допомогою платформи Node.js [25], що розширює його застосування на область розробки серверних застосунків.

Іншою важливою перевагою JavaScript є його велика спільнота розробників та велика кількість готових бібліотек і фреймворків, які спрощують розробку та розвиток веб-додатків. Це робить JavaScript потужним інструментом для розробки різноманітних веб-проектів.

Python [26] – це високорівнева, інтерпретована мова програмування загального призначення. Вона була розроблена Гвідо ван Россумом і вперше випущена у 1991 році. Однією з головних особливостей Python є його простий і лаконічний синтаксис, який дозволяє програмістам виразно виражати свої ідеї з меншою кількістю рядків коду порівняно з іншими мовами програмування.

Python є мовою програмування, яка легко вивчається і використовується, особливо для новачків. Він рекомендується як початкова мова для тих, хто тільки вчиться програмувати через свою простоту і доступність. Python відомий своєю легкістю в читанні, що полегшує співпрацю між розробниками, адже розуміння іншого коду є менш проблематичним.

Ще однією важливою особливістю Python є його велика стандартна бібліотека, яка містить різноманітні корисні модулі та інструменти для розв'язання різних задач. Це дозволяє розробникам ефективно використовувати готові рішення без необхідності переписування коду з нуля.

Python також славиться своєю універсальністю і гнучкістю. Він використовується у різних галузях, включаючи веб-розробку, наукові обчислення, штучний інтелект, аналіз даних та багато інших. Мова має велику спільноту розробників, що означає наявність безлічі ресурсів, документації та сторонніх бібліотек для використання.

Однією з важливих переваг Python є його висока продуктивність та швидкість розробки. Розробники можуть швидко створювати функціональні прототипи та додатки з меншими зусиллями порівняно з іншими мовами програмування. Python також підтримує функціональне, процедурне та об'єктно-

орієнтоване програмування, що робить його дуже гнучкою для використання в різних сценаріях розробки.

Java [27] – це об'єктно-орієнтована, клас-базована мова програмування, яка була розроблена в середині 1990-х років компанією Sun Microsystems. Вона стала вельми популярною завдяки своїй платформонезалежності, що означає, що програми Java можуть запускатися на будь-якому пристрої, який має відповідний віртуальний машинний інтерпретатор (Java Virtual Machine - JVM).

Однією з ключових особливостей Java є її простота в використанні та легкість вивчення. Мова має чіткий і зрозумілий синтаксис, який спрощує написання коду та його читання іншими програмістами. Java також підтримує об'єктно-орієнтоване програмування, що сприяє використанню ефективних та організованих методів розробки.

Ще однією важливою особливістю Java є її велика стандартна бібліотека, яка містить різноманітні готові рішення для різних завдань. Це полегшує розробку програм та зменшує час, необхідний для написання коду, оскільки багато вбудованих функцій можна використовувати безпосередньо зі стандартних бібліотек.

Ще однією важливою перевагою Java є її висока переносимість. Програми Java можуть бути запущені на будь-якій платформі, яка підтримує JVM, без необхідності змінювати код. Це робить її ідеальним вибором для розробки крос-платформених додатків.

Крім того, Java дозволяє розробникам створювати великі, розширювані програми за допомогою механізмів класів і об'єктів, а також підтримує різні парадигми програмування, такі як мультипотоковість та мережеві додатки.

Загалом, Java - це потужна та гнучка мова програмування, яка знаходить застосування у багатьох областях розробки програмного забезпечення, від веб-додатків до мобільних застосунків і великих корпоративних систем.

Нижче наведено порівняльний аналіз даних мов програмування (Таблиця 3.1).

Таблиця 3.1 – Порівняльний аналіз мов програмування

Критерій порівняння	JavaScript	Java	Python
Застосування в веб-розробці	+	-	-
Популярність та спільнота розробників	+	+	+
Синтаксис та зрозумілість	+	-	+
Ефективність та швидкодія	-	+	+
Розширюваність та бібліотеки	+	+	+
Сумісність та підтримка веб-браузерами	+	-	-
Можливості асинхронного програмування	+	-	+
Сумарний бал	+4	+1	+3

В даній таблиці JavaScript виявляється найбільш доцільним вибором для веб-розробки. Він має широкий спектр застосування в веб-розробці, є дуже популярним та має велику спільноту розробників. Синтаксис JavaScript зазвичай вважається досить зрозумілим для багатьох розробників, і він має багато розширюваних можливостей та бібліотек для розвитку веб-додатків. Він також добре сумісний з веб-браузерами і має потужні можливості асинхронного програмування, що робить його перевагою у веб-розробці порівняно з Java і Python.

Visual Studio Code [28] є відмінним інструментом для розробки веб-додатків. Його безкоштовність та відкритий вихідний код роблять його доступним для широкого кола розробників. Легкий у використанні інтерфейс спрощує процес роботи, а розширюваність та підтримка різних мов програмування роблять його універсальним інструментом для веб-розробки. Інтеграція з Git полегшує керування версіями коду, а вбудовані засоби для відлагодження та аналізу коду полегшують процес розробки. Активна спільнота та регулярні оновлення від Microsoft гарантують підтримку та вдосконалення інструменту. Крім того, крос-платформенність дозволяє використовувати VS Code на різних операційних системах. З усім цим урахуванням, Visual Studio Code стає вибором для розробки веб-додатків з урахуванням його зручностей, функціональності та підтримки.

Далі варто звернути увагу на вибір бази даних проекту [29].

Обравши MongoDB для веб-додатку, враховано кілька ключових аспектів. MongoDB привернула мене своєю гнучкістю схеми даних, дозволяючи працювати з різною структурою даних. Можливість горизонтального масштабування робить її відмінним вибором для додатків зі змінюючимся обсягом даних.

JSON-подібний формат зберігання даних у MongoDB робить взаємодію з даними зручною та читабельною. Швидкодія операцій, підтримка геоданих та індексація забезпечують високу ефективність обробки даних для веб-додатка в реальному часі.

Активна спільнота розробників та популярність MongoDB в сфері веб-розробки вказують на доступність ресурсів та підтримку, що є важливим для успішної розробки та підтримки веб-додатку.

Використання Node.js для веб-додатку є логічним вибором з кількох причин. Node.js базується на JavaScript, що спрощує розробку веб-додатків, оскільки можна використовувати одну мову програмування для фронтенду та бекенду. Його асинхронна модель програмування сприяє високій

продуктивності, особливо при обробці багатьох одночасних запитів, що є важливим для веб-додатків.

Node.js також дозволяє використовувати пакетний менеджер npm, що спрощує управління залежностями та інтеграцію різноманітних модулів для розширення функціональності додатку. Висока швидкодія та здатність масштабування Node.js дозволяють ефективно обробляти запити в реальному часі та легко масштабувати додаток при зростанні його обсягу.

Завдяки великій та активній спільноті розробників, Node.js надає широкий спектр ресурсів, документації та плагінів, що сприяє швидкому вирішенню завдань розробки та усуненню можливих проблем. Його крос-платформенність дозволяє використовувати його на різних операційних системах, що робить його універсальним інструментом для створення веб-додатків.

На останок було обрано фреймворк для роботи з клієнтською частиною, серед основних в цій ніші є Angular, React, Vue.

Angular [30] обраний для розробки веб-додатку з кількох обґрунтованих причин. Цей фреймворк надає повноцінний набір інструментів для створення динамічних та інтерактивних веб-інтерфейсів. Використання TypeScript, яке є сучасним синтаксисом JavaScript, дозволяє створювати масштабовані та легко підтримувані додатки.

Angular пропонує компонентний підхід до розробки, що полегшує структурування та управління кодом. Система залежностей та вбудована підтримка для тестування допомагають забезпечити стабільність та ефективність веб-додатку.

Завдяки своєму механізму двостороннього зв'язку даних, Angular дозволяє швидко взаємодіяти з користувачем та динамічно оновлювати вміст сторінок. Його широка спільнота та офіційна документація роблять його потужним інструментом для розробників, сприяючи швидкому вирішенню завдань та вирішенню можливих проблем.

Використання Angular дозволяє ефективно створювати односторінкові додатки (SPA), що робить його ідеальним вибором для сучасних веб-додатків з високим рівнем взаємодії та динамічним вмістом.

3.2 Програмна реалізація додатку

В першу чергу варто розглянути основні інтерфейси створені для сутностей додатку, що спростять та зобразять технічні залежності одних колекцій від інших.

Головною колекцією додатку є сутність “Проект”, що вміщає в себе інші колекції та зберігає всю інформцію про проект (Рисунок 3.1).

```
export interface Project {
  _id?: string;
  name: string;
  description: string;
  likes: string[];
  photos: string[];
  messages: Message[];
  requests: Request[];
  latitude: number;
  longitude: number;
}
```

Рисунок 3.1 – Інтерфейс сутності “Проект”

Також не менш важливою сутністю є “Користувач” котрий повинен містити особисту інформацію та можливість змінювати її лише ним самим (Рисунок 3.2).

```
export interface User {
  _id?: string;
  name: string;
  login: string;
  age: number;
  photo: string;
  description: string;
  admin: boolean
}
```

Рисунок 3.2 – Інтерфейс сутності “Користувач”

Також важливою складовою є сутність “Запит”, що допомагає адміністратору збирати правки до проекту та змінювати проект в залежності від поданої інформації. Важливо щоб редагування проекту не здійснювалося на пряму користувачем, а лише відправкою і підтвердженням запиту адміністратором (Рисунок 3.3).

```
export interface Request {
  _id?: string;
  text: string;
  author: string;
  approved: boolean;
  createdAt: string[];
  updatedAt: string[];
}
```

Рисунок 3.3 – Інтерфейс сутності “Запит”

В ході розробки додатку було створено покращений алгоритм кластеризації маркерів на мапі за допомогою JavaScript інструментів. Який складається з певних складових. Для початку потрібно реалізувати алгоритм визначення відстаней між точками (Рисунок 3.4).

```
function manhattanDistance(point1, point2) {
    return Math.abs(point2.x - point1.x) + Math.abs(point2.y - point1.y);
}
```

Рисунок 3.4 – Алгоритм визначення відстаней між точками

Після того як визначено відстані за допомогою Маннхетенської відстані можна приступити до функції, що обирає центри кластерів. Важливо детально опрацювати процес вибору перших центрів кластерів щоб кластеризація не виконувалася на максимально приближених точках, що в свою чергу значно пришвидшить роботу алгоритму та не допустить помилкових об’єднань маркерів. (Рисунок 3.5).

```
function calculateCentroids(clusters) {
  return clusters.map(cluster => {
    const clusterSize = cluster.length;
    const clusterCenter = cluster[0].map(coord => 0);

    cluster.forEach(point => {
      point.forEach((coord, i) => {
        clusterCenter[i] += coord / clusterSize;
      });
    });

    return clusterCenter;
  });
}
```

Рисунок 3.5 – Алгоритм визначення відстаней між точками

Далі було розроблено “Головну” сторінку яка надає загальну інформацію по проектам та зображає інтеркативну мапу з маркерами проектів. (Рисунок 3.6).

```
<section id="map">

</section>

 <section id="projects">
     <h2>Проекти з благоустрою</h2>
     <ul>
         <li **ngFor="let new of news"><a href="#">{{new.name}}: {{new.description}}</a></li>
     </ul>
 </section>

 <section id="news">
     <h2>Головні новини</h2>
     <article **ngFor="let paper of papers">
         <h3><a href="#">{{paper.title}}</a></h3>
         <p>{{paper.text}}</p>
     </article>
 </section>
```

Рисунок 3.6 – Html частина сторінки “Головна”

Також для правильної роботи сторінки було додано функціональну частину у вигляді TypeScript компонента (Рисунок 3.7).

```
@Component({
  selector: 'app-home',
  templateUrl: './home.component.html',
  styleUrls: ['./home.component.css']
})
export class HomeComponent implements OnInit {
  projects: any[];
  constructor(private mapService: MapService, private projectService: ProjectService) { }

  ngOnInit() {
    this.loadProjects();
  }

  loadProjects() {
    this.projectService.getProjects().subscribe(projects => {
      this.projects = projects;
      this.mapService.initMap('map', projects);
    });
  }
}
```

Рисунок 3.7 – TypeScript частина сторінки “Головна”

3.3 Висновки

Аналіз існуючих аналогів допоміг ідентифікувати найкращі практики та уникнути можливих помилок у розробці. Переваги та недоліки аналогів надали змогу визначити ключові вимоги до програмного продукту, зокрема, гнучкість у роботі з даними, можливість відстеження змін у режимі реального часу, а також відкритий інтерфейс для взаємодії з користувачами.

У цьому розділі ми розробили та інтегрували компонент головної сторінки нашого веб-додатку. Ми успішно використали Angular для створення цього компоненту, який включає в себе інтерактивну мапу та відображення списку проектів користувача.

Важливим аспектом цього розділу є інтеграція мапи за допомогою, який дозволяє відобразити маркери проектів на мапі та реалізує їхню кластеризацію для зручності відображення багатьох маркерів на невеликій області.

В цілому, розроблений компонент головної сторінки не лише забезпечує візуалізацію даних, але й створює зручний та інтуїтивний інтерфейс для користувачів, що є ключовим аспектом розробки веб-додатків. У наступних

розділах ми будемо продовжувати розробку та реалізацію інших функцій та можливостей додатку.

Також було розроблено інтерфейси основних документів платформи таких як “Користувач”, “Проект”, “Запит”. Ці інтерфейси забезпечують основну функціональність додатку, а тому над важливо правильно та доцільно оперувати даними, що їм належать та не робити хибних залежностей між ними.

На останок було описано головний розроблюваний алгоритм кластеризації котрий вміщує в себе декілька етапів, що значно покращують його в порівнянні з іншими алгоритмами та дозволяють швидко і точно об’єднувати маркери на мапі.

4 ТЕСТУВАННЯ ПРОГРАМНОГО ДОДАТКУ

4.1 Аналіз методів та засобів тестування

У контексті MEAN стек додатку розділ включає в себе огляд та розгляд різних аспектів тестування, щоб забезпечити високу якість програмного продукту. MEAN стек включає MongoDB, Express.js, AngularJS (або Angular), та Node.js, і кожен з цих компонентів потребує особливого підходу до тестування.

Unit-тестування є ключовим елементом для забезпечення якості коду в MEAN стеку. Для MongoDB можна використовувати тестові фреймворки, такі як Mocha або Jest. Для Express.js - Supertest. Angular та Node.js також мають свої вбудовані функціональності для unit-тестування.

Задача unit-тестування в MEAN стеці (MongoDB, Express.js, Angular, Node.js) полягає у перевірці окремих компонентів коду (функцій, методів, класів тощо) для визначення, чи вони працюють вірно і відповідають очікуванням.

Для тестування взаємодії з базою даних MongoDB, можна використовувати фреймворки Mocha або Jest:

Mocha – це гнучкий та розширюваний фреймворк для тестування JavaScript. Ви можете використовувати Mocha для написання unit-тестів, які перевіряють різні операції з базою даних MongoDB, такі як вставка, оновлення, вилучення тощо.

Приклад коду для тестування MongoDB з використанням Mocha (Рисунок 4.1):

```
const assert = require('assert');
const { MongoClient } = require('mongodb');

describe('MongoDB Tests', function () {
  it('should insert a document into the collection', async function () {
    const client = new MongoClient('mongodb://localhost:27017', { useNewUrlParser: true });
    await client.connect();

    const db = client.db('testDB');
    const collection = db.collection('testCollection');

    const document = { key: 'value' };
    await collection.insertOne(document);

    const result = await collection.findOne({ key: 'value' });
    assert.deepStrictEqual(result, document);

    await client.close();
  });
});
```

Рисунок 4.1 – Код для тестування MongoDB з використанням Mocha

Jest – це фреймворк для тестування, який забезпечує вбудовану підтримку для асинхронних тестів та зручні засоби для створення mock-об'єктів. Він також може використовуватися для тестування MongoDB операцій.

Приклад коду для тестування MongoDB з використанням Jest:

```
const { MongoClient } = require('mongodb');

describe('MongoDB Tests', () => {
  it('should insert a document into the collection', async () => {
    const client = new MongoClient('mongodb://localhost:27017', { useNewUrlParser: true });
    await client.connect();

    const db = client.db('testDB');
    const collection = db.collection('testCollection');

    const document = { key: 'value' };
    await collection.insertOne(document);

    const result = await collection.findOne({ key: 'value' });
    expect(result).toEqual(document);

    await client.close();
  });
});
```

Рисунок 4.2 – Код для тестування MongoDB з використанням Jest

Для тестування додатку було обрано саме Mocha через її гнучкість, спрощену архітектуру та асинхроність.

Для тестування Express.js додатків використовують фреймворк Supertest, який спрощує відправку HTTP-запитів та перевірку їх відповідей.

Supertest може бути використаний для тестування різних API-маршрутів та перевірки відповідей сервера.

Supertest – це бібліотека для тестування API веб-додатків, написаних на Node.js, за допомогою JavaScript або TypeScript. Вона надає зручний інтерфейс для відправки HTTP-запитів до вашого веб-сервера та перевірки його відповідей.

Angular та Node.js мають вбудовані функціональності для unit-тестування.

Jasmine – це фреймворк для тестування JavaScript-коду, спеціалізований на тестуванні коду, написаного на мові програмування JavaScript. Jasmine найчастіше використовується для unit-тестування коду веб-додатків, включаючи ті, які написані на популярних фреймворках, таких як Angular.

Jasmine використовує підхід Behavior-Driven Development, який ставить акцент на те, як поведінка системи повинна взаємодіяти з її складовими. Такий підхід робить тести більш читабельними та зрозумілими для всієї команди.

Тестові сценарії в Jasmine виглядають як набір описових вимог. Синтаксис побудований так, щоб він був зрозумілим та природним для людей, що розробляють.

Jasmine постачається з багатьма функціями спостереження (matchers), які дозволяють розробникам перевіряти різні аспекти поведінки коду. Наприклад, expect(result).toBe(5) перевіряє, чи отриманий результат дорівнює 5.

Jasmine надає можливості для створення "шпигунів" (spies), які дозволяють вам стежити за викликами функцій і перевіряти їхнє викликання, передачу параметрів тощо.

Jasmine підтримує асинхронні тести, що дозволяє ефективно тестувати код, який використовує асинхронні функції, такі як обіцянки (Promises) або колбеки.

Jasmine часто використовується в контексті тестування коду Angular, але може бути використаний і для інших JavaScript-проектів. Загальна його мета - надати зручний і природний спосіб визначення та виконання тестів, що зрозумілий як для розробників, так і для тестувальників чи менеджерів проекту (Рисунок 4.3).

```
describe('AppComponent', () => {
  beforeEach(async () => {
    await TestBed.configureTestingModule({
      imports: [
        RouterTestingModule
      ],
      declarations: [
        AppComponent
      ],
    }).compileComponents();
  });

  it('should create the app', () => {
    const fixture = TestBed.createComponent(AppComponent);
    const app = fixture.componentInstance;
    expect(app).toBeTruthy();
  });

  it(`should have as title 'opesjet-op-angular'`, () => {
    const fixture = TestBed.createComponent(AppComponent);
    const app = fixture.componentInstance;
    // expect(app.title).toEqual('opesjet-op-angular');
  });

  it('should render title', () => {
    const fixture = TestBed.createComponent(AppComponent);
    fixture.detectChanges();
    const compiled = fixture.nativeElement;
    expect(compiled.querySelector('.content span').textContent).toContain('opesjet-op-angular app is running!');
  });
```

Рисунок 4.3 – Коду для тестування Angular додатку з використанням Jasmine

Protractor – це фреймворк для автоматизованого тестування для веб-додатків, написаних на мові програмування JavaScript або TypeScript, особливо для додатків, розроблених на платформі Angular.

Protractor використовує Selenium WebDriver для автоматизації дій та інтеракції з елементами веб-сторінок.

Protractor розроблений спеціально для тестування Angular-додатків і має вбудовану підтримку для виявлення та чекання на асинхронні події, такі як завантаження сторінки, обробка запитів до сервера, тощо.

Protractor підтримує два популярних фреймворки для тестування - Jasmine і Mocha. Ви можете використовувати їх для організації та написання тестів.

Node.js зазвичай використовується для створення серверної частини додатку. Для тестування можна використовувати фреймворки, такі як Mocha або Jest, аналогічно тому, як було показано раніше для MongoDB.

Тестування [31] – це процес перевірки та оцінки програми чи системи для визначення того, чи вона відповідає вимогам та очікуванням користувачів. Основна мета тестування – виявлення помилок, аномалій чи несправностей у програмному продукті перед його випуском або впровадженням в експлуатацію.

Тестування допомагає впевнитися в тому, що програмний продукт працює правильно, відповідає вимогам, ефективний та надійний. Воно також допомагає виявити та виправити проблеми, які можуть виникнути під час роботи системи.

Метод чорної скриньки (Black Box Testing) – це метод тестування програми або системи, в якому тестувальник не знає внутрішніх деталей реалізації програми. Тобто тестувальник розглядає систему як чорний ящик, який взаємодіє з вхідними даними та генерує відповіді, не знаючи, як саме це відбувається всередині.

Метод чорної скриньки є ефективним для тестування функціональності, коректності та взаємодії програми з користувачем. Однак цей метод може не виявити деякі проблеми внутрішньої реалізації, такі як помилки в коді чи проблеми з продуктивністю. Тому часто використовують комбінацію методів чорної та білої скриньки для більш комплексного тестування системи.

На відміну від методу чорної скриньки та білої скриньки, метод сірої скриньки (Grey Box Testing) лежить між цими двома підходами. Він комбінує елементи знань про внутрішню структуру системи (схожі на білу скриньку) та зберігає зовнішню перспективу (схожу на чорну скриньку).

Цей метод використовується тоді, коли тестувальник має деяке розуміння внутрішніх процесів системи, але не повністю освідомлює всю її структуру. Метод сірої скриньки дозволяє забезпечити більш широкий огляд програмного

продукту, порівняно з методом чорної скриньки, і виявити проблеми на рівнях внутрішньої реалізації.

У даному підрозділі проведено детальний аналіз методів та засобів, які використовувались для верифікації розробленого програмного продукту. Розглянуто різні підходи до тестування, такі як модульне тестування, інтеграційне тестування, системне тестування та приймальне тестування.

Проведено порівняльний аналіз існуючих методів тестування, вибір оптимального підходу та обґрунтування вибору конкретних засобів для використання в експериментальній перевірці розробленого застосунку на платформі MEAN stack. У результаті аналізу визначено оптимальну стратегію тестування для забезпечення найвищої ефективності та надійності розробленого продукту.

Для тестування програмної системи «БлагоМіст» було обрано тестування методом сірої скриньки. Це дозволить перевірити функціонал системи з точки зору користувача та впевнитися, що всі операції успішно виконуються, а дані в базі даних обробляються коректно.

4.2 Тестування програмної системи “БлагоМіст” з міноторингу та контролю за благоустроєм

Тестування розробленого продукту є ключовим етапом в життєвому циклі розробки, спрямованим на забезпечення якості та надійності програмного забезпечення. Цей етап дозволяє виявити та усунути потенційні дефекти, забезпечуючи користувачів продукту найвищим рівнем задоволення та ефективності в його використанні.

Так як було обрано метод сірої скриньки то він включає в себе підходи як білої так і чорної скриньки.

Першим модулем з яким стикається користувач при відкритті веб-додатку стає гловна сторінка. Для перевірки даного модулю потрібно перевірити чи мапа коректно відображає маркери проектів, та додатково дублює їх у списку поряд вже з більш інформативним наповненням.

Запустивши веб-додаток перед користувачем виникне головна сторінка сайту (Рисунок 4.4).

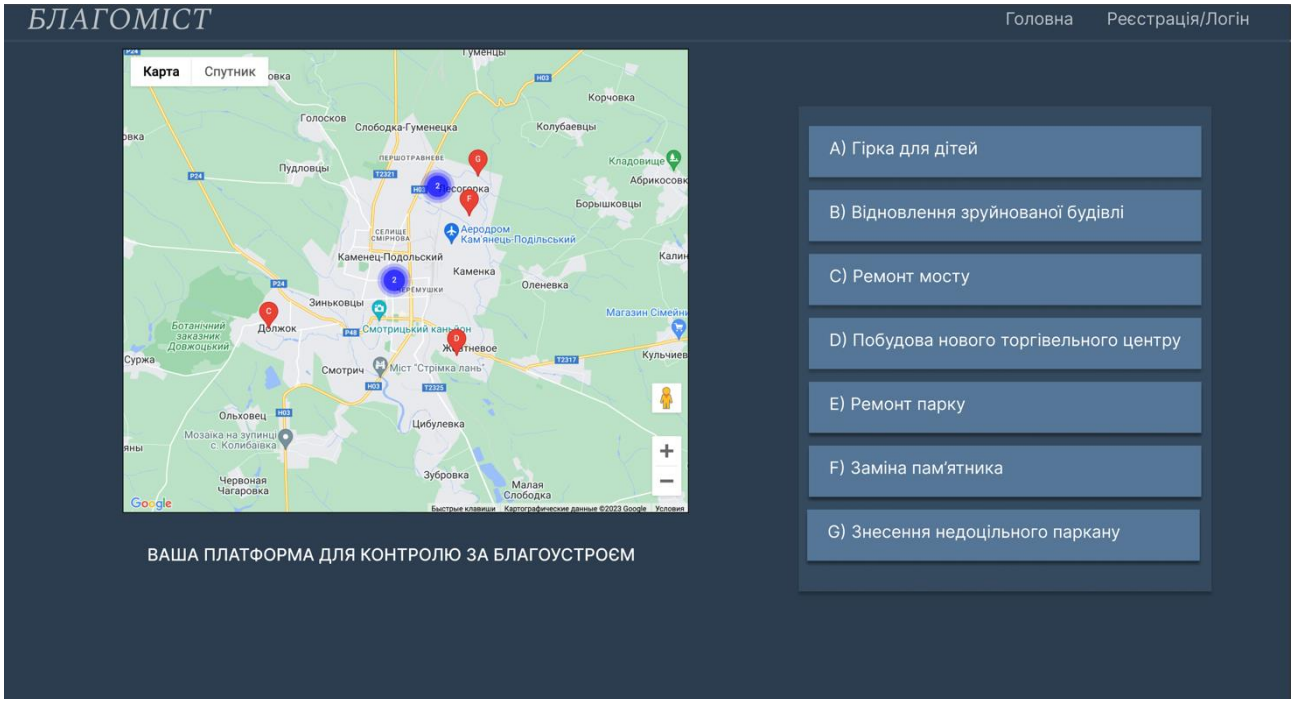

Рисунок 4.4 – Вікно головної сторінки

Якщо користувач не надав доступ до геоданих він не буде мати доступ до більшості функціоналу, та зможе лише переглядати наявні проекти без можливості якось впливати не їх перебіг.

Після цього користувач має змогу пройти реєстрацію або ж при наявності аккаунту увійти в нього.

Для входу у акаунт користувач повинен надати логін та пароль від створеного акаунту. При невірних введених даних користувач отримає повідомлення про некоректність введених даних та йому буде запропонувати спробувати ще раз.

Важливою складовою автентифікації є безпека користувача, тому навіть при успішній авторизації користувачу не можна повертати чуттєву інформацію, що може призвести до втрати даних.

При умові що користувач не має створеного акаунту йому потрібно буде перейти до реєстрації. Для цього йому потрібно відкрити вікно реєстрації за посиланням під кнопкою “Ввійти” (Рисунок 4.5).

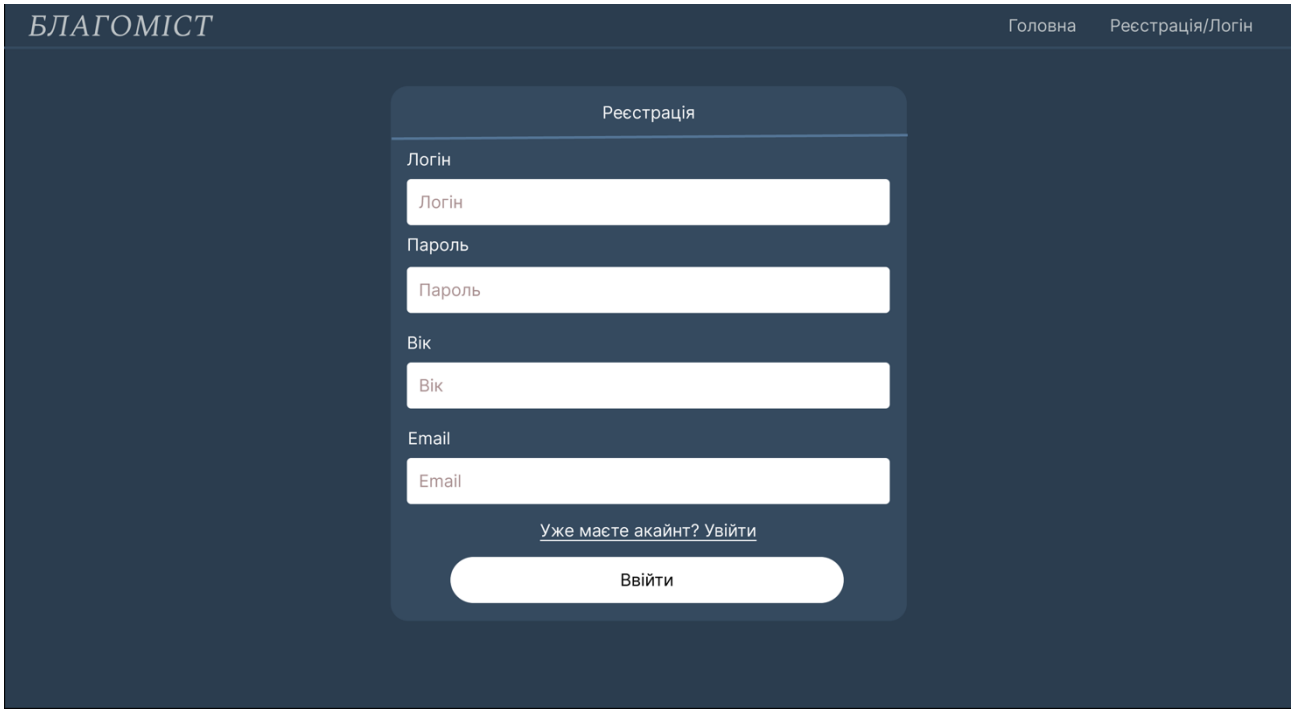

Рисунок 4.5 – Вікно реєстрації користувача

Після переходу до вікна реєстрації потрібно заповнити всю потрібну інформацію. У випадку некоректно введених даних користувач отримає повідомлення. Якщо всі дані коректні та користувач не присутній на платформі, система авторизує його та відправить на головну сторінку але вже авторизованим, з можливістю використання всього функціоналу програми.

Для перевірки авторизації було зроблено запит до бази даних до (Рисунок 4.6) та після створення користувача.

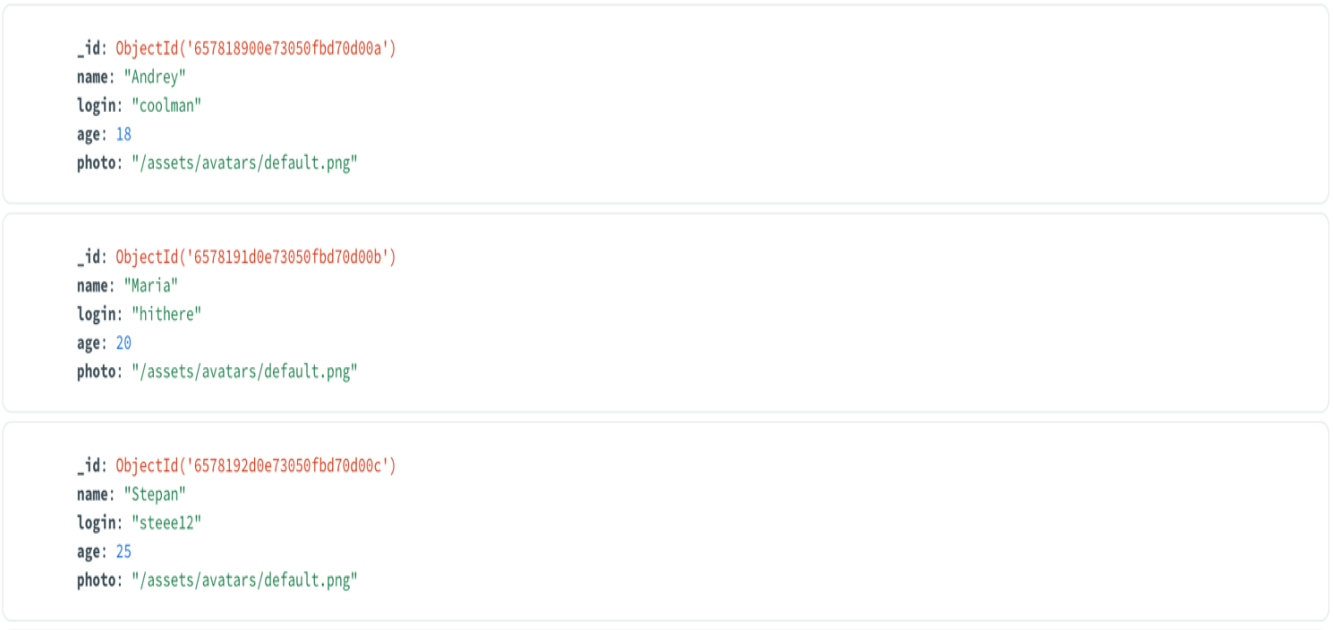

Рисунок 4.6 – Стан бази даних до створення користувача

Виходячи з рисунку 4.7 в базу даних було успішно додано нового користувача.

Рисунок 4.7 – Стан бази даних після створення користувача

Після відповіді системи про успішну авторизацію, користувач має змогу відкривати загальний список проектів з можливістю додавання інших, або ж відкрити сторінку налаштувань свого профілю.

Якщо користувач перейде на сторінку всіх проектів, то він побачить таблицю з можливістю пагінації документів. Також у нього буде можливість шукати проекти за їх назвою або адресою та сортувати їх за певними критеріями (Рисунок 4.8).

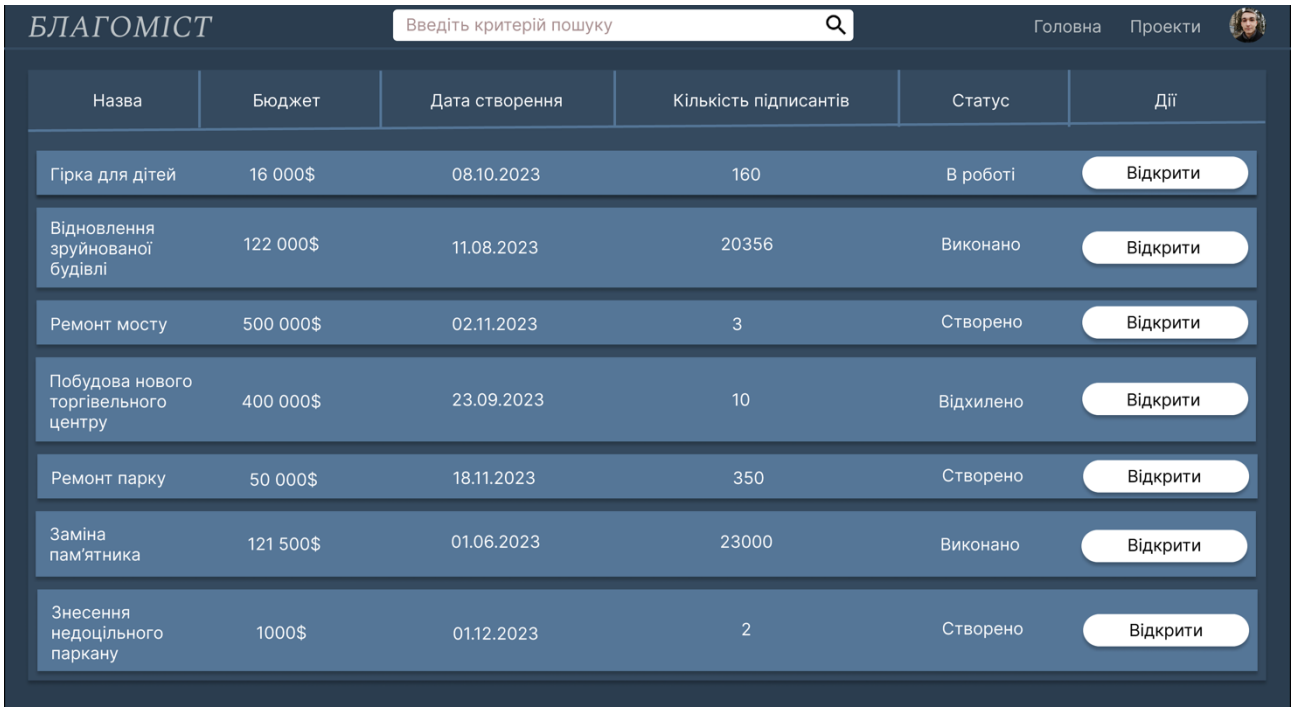

Рисунок 4.8 – Вікно сторінки проектів

Якщо користувач натисне на конкретний проект він відкриє його профіль з усією доступною інформацією про нього. У нього буде можливість переглянути фотографії та опис проекту детально. Також у нього буде змога натиснути на кнопку “Запропонувати зміну”, що надасть йому змогу надіслати повідомлення про бажання певних змін у проекті (Рисунок 4.9).

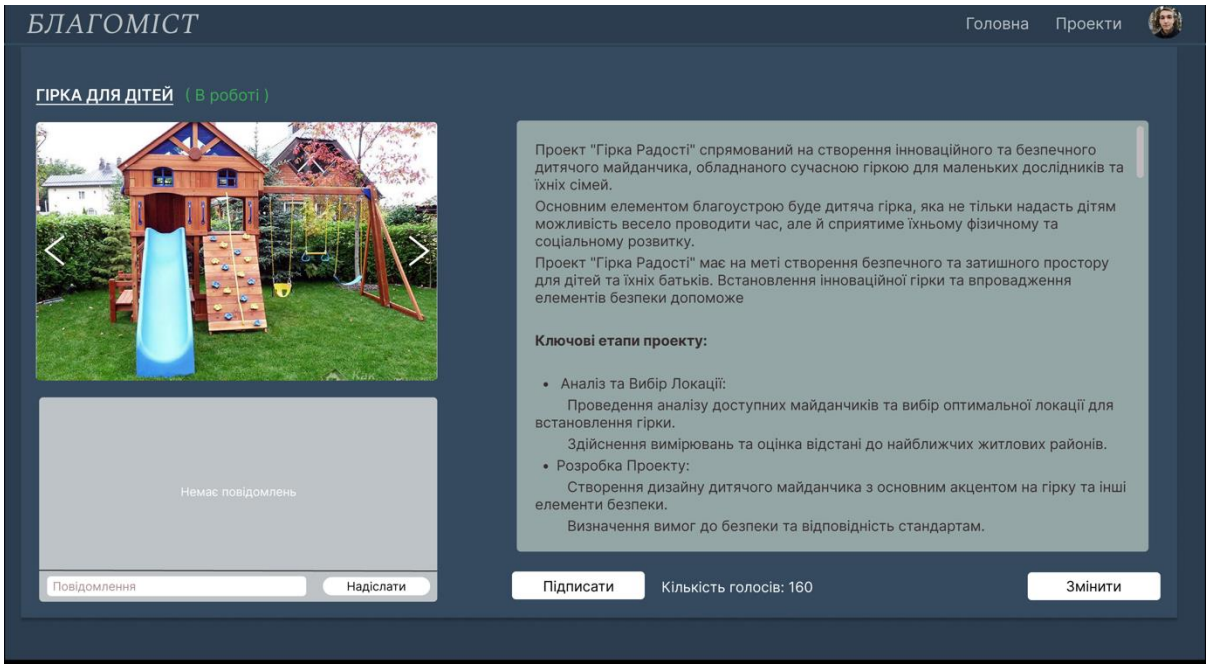

Рисунок 4.9 – Вікно сторінки проекту

Після відправки користувача буле повернуто на профіль проекту та повідомлено, про те що заявка знаходиться у розгляді.

Також на сторінці проекту можна проголосувати за нього натиснувши на кнопку “Підписати”, після цього кількість підписантів буде змінено та записано у базу даних інформацію про проголосувавшу особу (Рисунок 4.10).

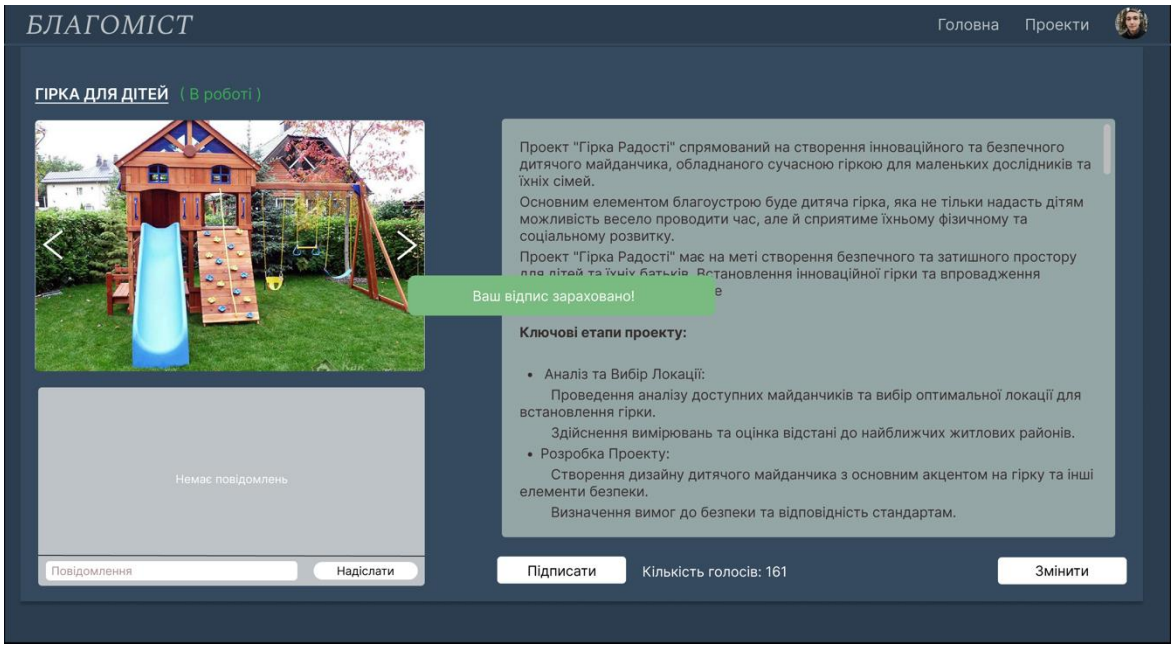

Рисунок 4.10 – Вікно проекту після голосування

При потребі користувач може скористатися вікном чату, в якому він може переписуватися з громадянами в режимі онлайн та дискутувати на різні теми. Для відправки повідомлення потрібно ввести у поле текст бажаний до відправки та натиснути на кнопку “Надіслати”. При успішному створені повідомлення воно буде додано до чату з інформацією про час та автора повідомлення (Рисунок 4.11).

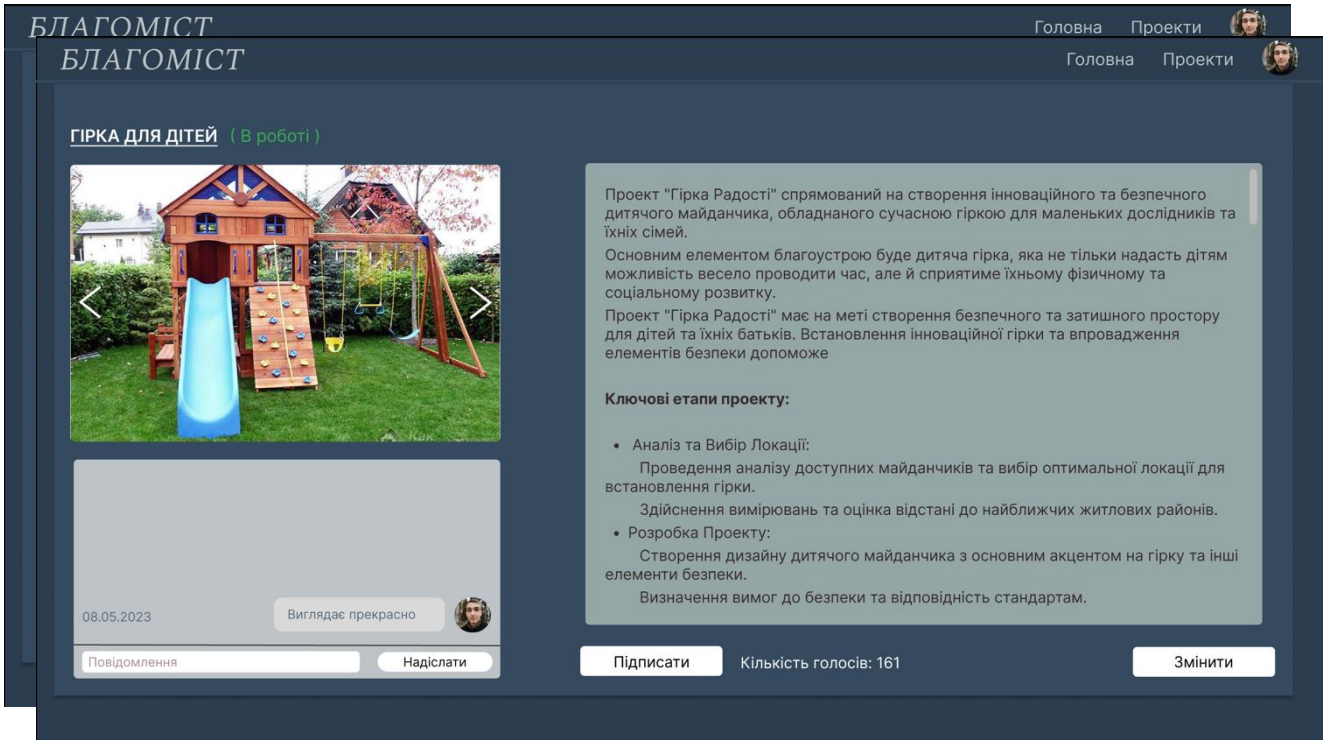

Рисунок 4.11 – Вікно проекту після відправки повідомлення

Для того щоб змінити дані користувача та додати потрібну інформацію, можна перейти на сторінку налаштувань натиснувши на іконку користувача в панелі навігації. Після цього його буде перенаправлено на сторінку де він зможе змінювати свою власну інформацію, таку як фотографія профілю, вік, посада. На сторінці доступний список проектів за котрі голосував користувач з вказанням їх статусу, для зручності розуміння стану проектів, також зображено список запитів на зміни до проектів (Рисунок 4.12).

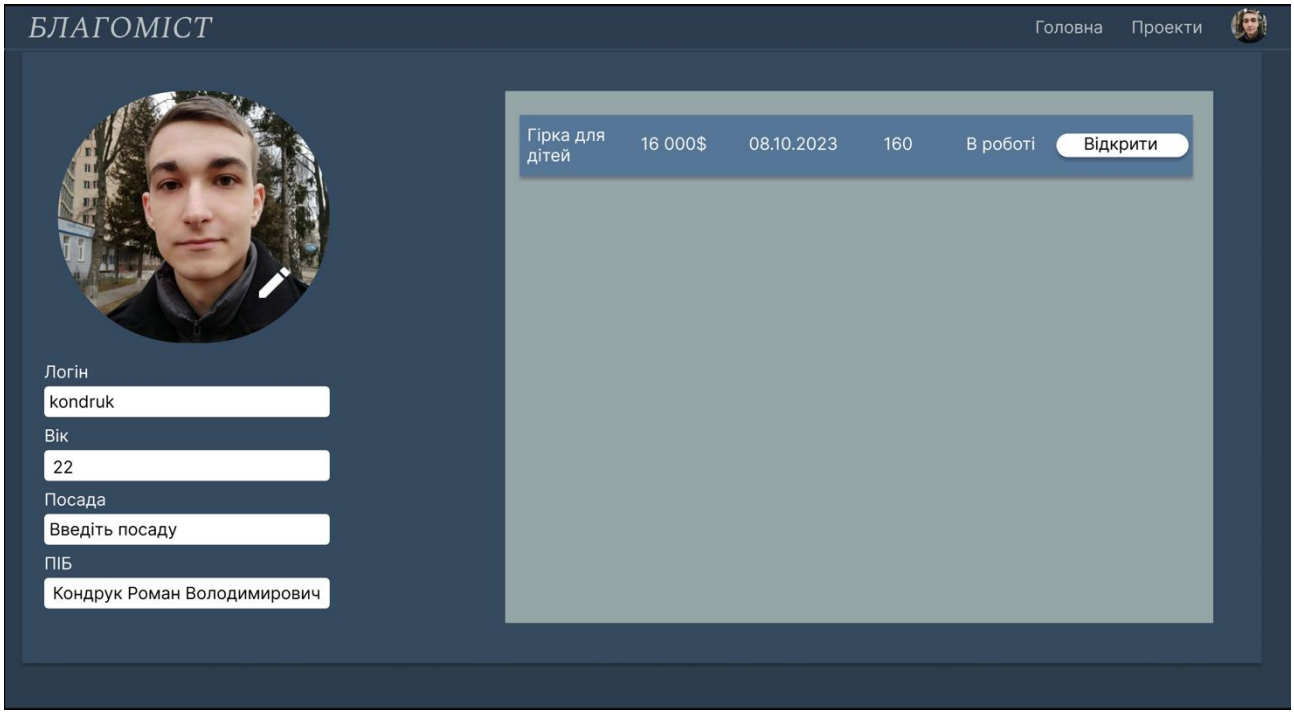

Рисунок 4.12 – Вікно профілю користувача

Тестування CRUD частини контролеру проектів вказало, вірність виконання запитів та сповістило про повну відповідність вимогами.

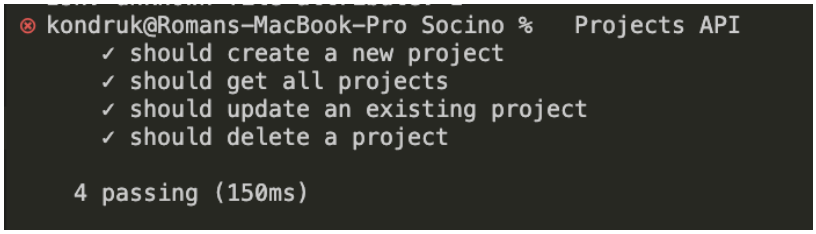

Рисунок 4.13 – Результат тестування бази даних модулю проектів за допомогою Mocha

Далі було проведено тестування Express js частини додатку, що має попередити проблеми з написанням коду запитів. Головна задача такого тестування це недопущення помилок в коді при отримані запиту з клієнтської частини додатку (Рисунок 4.14).

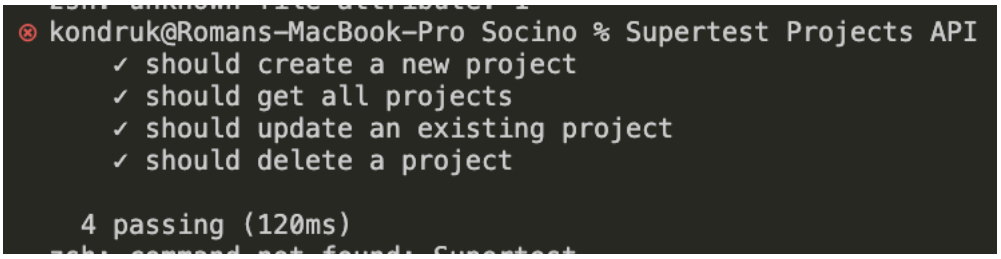

Рисунок 4.14 – Результат тестування серверної частини модулю проектів за допомогою Supertest

Наступним етапом розробки Angular додатку було проведено тестування функціональної частини, зокрема модуля проектів. Цей етап був важливим кроком у забезпеченні якості та надійності додатку. Давайте розглянемо основні аспекти цього тестування.

Для початку тестування було обрано модуль проектів, який перед цим успішно пройшов всі unit-тести та виявив свою працездатність. Це стратегічне рішення дозволяє переконатися, що базові функціональності додатку працюють стабільно перед тим, як переходити до більш складних тестів.

Unit-тести для модуля проектів були успішно виконані, вказуючи на те, що окремі компоненти та сервіси, пов'язані з цим модулем, працюють коректно та відповідають специфікаціям. Це забезпечує впевненість у тому, що базовий функціонал проектів відповідає очікуванням.

Тестування також включало перевірку коректного запуску додатку та відсутності помилок при його ініціалізації. Це важливо, оскільки правильне функціонування додатку в цілому є ключовим аспектом для користувача (Рисунок 4.15).

Результати тестування функціональної частини Angular додатку підтвердили, що він готовий до подальшого розширення та вдосконалення. Впровадження етапу функціонального тестування сприяє виявленню та усуненню можливих проблем на ранніх етапах розробки, забезпечуючи високу якість та надійність програмного забезпечення. Такий підхід також полегшує

подальшу роботу над проектом та допомагає забезпечити зручний та безперебійний досвід користувачів.

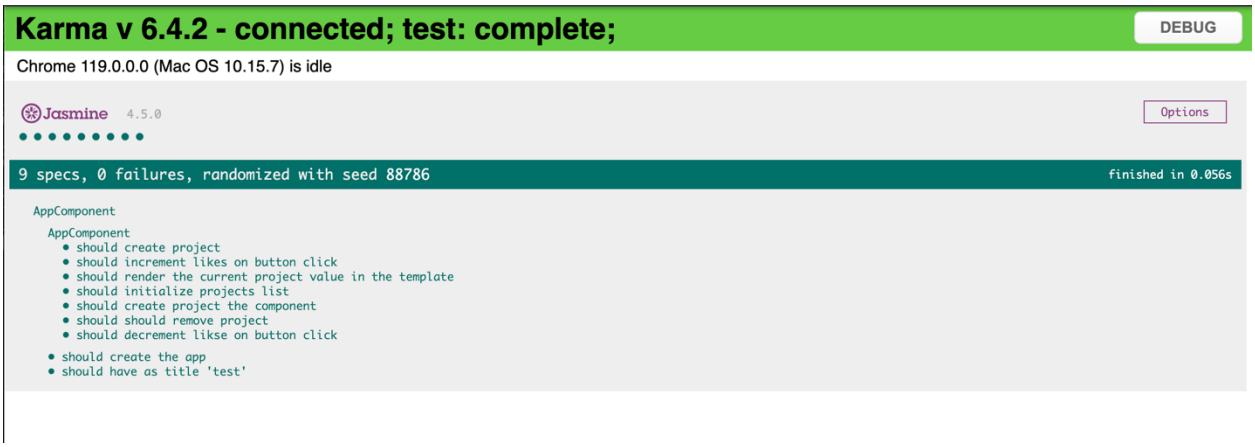

Рисунок 4.15 – Результат тестування клієнтської частини модулю проектів за допомогою Jasmine

В ході тестування ключових запитів до бази даних MongoDB в контексті проектів вдалося зробити кілька важливих висновків.

По-перше, було успішно підтверджено працездатність бази даних, включаючи всі операції додавання, зчитування, оновлення та видалення даних. Це свідчить про те, що механізм взаємодії з базою даних в рамках CRUD-операцій працює стабільно.

Другим важливим аспектом було підтвердження вірності реалізації контролера проектів. Тестування дало змогу переконатися, що контролер правильно взаємодіє з базою даних, забезпечуючи коректну обробку запитів та повернення очікуваних результатів.

Досягнення відповідності реалізації функціоналу вимогам та специфікаціям проекту є іншим ключовим висновком. Всі тести пройшли успішно, вказуючи на те, що розроблений функціонал проектів відповідає зазначеним вимогам.

Тестування також виявило можливі проблеми на ранніх етапах розробки, що дозволяє забезпечити високу якість та надійність коду. Інтеграція тестів до

бази даних робить сприйняття змін стабільним та допомагає уникнути несподіваних помилок під час модифікацій коду.

Успішні результати тестування створюють позитивний фундамент для подальшого розвитку проекту. За основу можна взяти, що основна частина функціоналу, пов’язаного з проектами, працює стабільно та вірно, що дає певну впевненість у готовності системи до подальших викликів та розширення.

4.3 Розробка інструкції користувача

Інструкція користувача – це документ, який надає детальні вказівки та інформацію щодо користування певним продуктом, сервісом або програмним забезпеченням. Головною метою інструкції користувача є допомогти користувачеві ефективно та безпечно використовувати продукт, а також забезпечити розуміння основних функцій та можливостей.

Потрібно визначити мінімальні технічні вимоги додатку для розуміння потенційних проблем в розповсюдженні.

Для взаємодії з програмною системою «БлагоМіст» для моніторингу та контролю за благоустроєм необхідно мати мінімальну конфігурацію персонального комп’ютера (ПК), що наведена у таблиці 4.1.

Таблиця 4.1 – Мінімальна конфігурація ПК

Тип процесора	32-розрядний (x86) або 64-розрядний (x64) процесор з тактовою частотою 1.1 ГГц
Об’єм оперативної пам’яті	200 Мб
Операційна система	Windows 7 і вище, MacOS 10.0 Cheetah і вище. IOS 3.1.3 і вище, Android 1.0 і вище

Веб-додаток "БлагоМіст" рекомендується використовувати в сучасних веб-браузерах, таких як Google Chrome, Safari, Mozilla Firefox або Microsoft Edge.

Для забезпечення оптимальної продуктивності та безпеки рекомендується використовувати оновлені версії браузерів.

Для нормальної роботи додатку вимагає стабільне підключення до Інтернету, адже воно забезпечує завантаження мап, проектів та отримання оновлень.

Зручний та функціональний кабінет користувача надає можливість взаємодії з усіма функціями додатку. Реєстрація та вхід у систему дозволяє користувачам отримати повний доступ до всіх можливостей.

JavaScript повинен бути увімкнений у налаштуваннях браузера для коректної роботи всіх функціональностей.

Додаток також адаптований для відображення на різних пристроях із різними розмірами екранів, забезпечуючи комфортне користування навіть на пристроях із різною діагоналлю екрану.

Періодичне оновлення браузера до останньої версії рекомендується для отримання всіх можливих покращень щодо швидкості, безпеки та сумісності з додатком.

У випадках, коли використання додатку вимагає API-ключів, важливо мати дійсні ключі та правильно їх використовувати для надання необхідного функціоналу.

Авторизація користувача - це процес перевірки ідентичності користувача для надання йому доступу до конкретних ресурсів, послуг чи функцій системи. Основна мета авторизації - забезпечення конфіденційності та безпеки інформації, а також контроль доступу користувачів до різних рівнів функціональності.

Процес авторизації може включати в себе наступні етапи:

Введення ідентифікаційних даних: Користувач повинен представити системі певні ідентифікаційні дані, такі як ім'я користувача та пароль, або використовувати інші методи аутентифікації, такі як біометричні дані чи одноразові коди.

Перевірка ідентичності: Система перевіряє, чи введені ідентифікаційні дані відповідають даним, збереженим в системі для конкретного користувача.

Надання доступу: Якщо ідентифікація пройшла успішно, система надає користувачеві доступ до визначених ресурсів чи функцій.

Щоб забезпечити захист авторизації від постороннього втручання, важливо вживати додаткові заходи безпеки:

Сильні паролі: Застосування паролів, які складаються з різних символів, регістрів і цифр, робить їх важкими для вгадування.

Двофакторна аутентифікація (2FA): Використання двох і більше методів аутентифікації (наприклад, пароль і одноразовий код, який надсилається на мобільний телефон) збільшує безпеку доступу.

Шифрування: Захист інформації шляхом шифрування перед її передачею по мережі може унеможливити перехоплення та читання даних.

Обмеження прав доступу: Користувачеві слід видавати лише ті права доступу, які необхідні для виконання його завдань. Це зменшує ризик недозволеного використання привілеїв.

Моніторинг та журналювання: Система повинна вести журнал подій для виявлення можливих атак або невдачних спроб авторизації, що дозволяє швидко реагувати на потенційні загрози.

Регулярна зміна паролів: Вимагати від користувачів регулярно змінювати свої паролі сприяє утрудненню потенційних атак.

Ці заходи разом сприяють підвищенню безпеки авторизації та захисту від несанкціонованого доступу до системи.

MEAN стек (MongoDB, Express.js, Angular, Node.js) - це набір технологій для розробки веб-додатків, які включає в себе базу даних MongoDB, серверний фреймворк Express.js, клієнтський фреймворк Angular та середовище виконання JavaScript на сервері Node.js. Давайте розглянемо, як можна в MEAN стеку забезпечити авторизацію користувача та захист її від постороннього втручання:

Аутентифікація та авторизація на серверному рівні (Node.js та Express.js):

Використовуйте популярні бібліотеки для аутентифікації та авторизації, такі як Passport.js.

Встановлюйте стратегії аутентифікації (локальну, через соціальні мережі, JWT тощо) відповідно до ваших потреб.

Зберігайте паролі в базі даних у безпечному вигляді, наприклад, здійснюючи хешування з використанням bcrypt.

Двофакторна аутентифікація (2FA):

Додайте можливість включення двофакторної аутентифікації для користувачів, щоб забезпечити додатковий рівень безпеки.

Маршрутизація та контроль доступу:

Визначте різні ролі користувачів (наприклад, адміністратор, звичайний користувач) і обмежте їх доступ до різних ресурсів або функцій.

Використовуйте middleware для перевірки прав доступу перед викликом конкретного маршруту.

JWT (JSON Web Tokens):

Використовуйте JWT для безпечної передачі інформації про аутентифікацію між клієнтом і сервером.

Підписуйте та шифруйте JWT для захисту від недозволеного доступу та модифікацій.

Захист від Cross-Site Request Forgery (CSRF):

Встановлюйте токени CSRF та використовуйте їх для захисту від атак типу CSRF.

Використання HTTPS:

Забезпечте використання протоколу HTTPS для шифрування даних, які передаються між клієнтом і сервером.

Журналювання та моніторинг:

Ведіть журнал подій для моніторингу спроб авторизації та невдалих спроб втручання.

Використовуйте моніторингові інструменти для виявлення аномальної активності.

Оновлення та патчі:

Регулярно оновлюйте всі компоненти стеку та використовуйте патчі для усунення відомих уразливостей.

Впровадження цих практик допоможе забезпечити ефективний рівень безпеки в MEAN стеку та зменшити ризик небажаного втручання в систему.

Додатково до зазначених раніше заходів, існує ряд інструментів і сервісів, які можна використовувати для підвищення рівня захисту вашого додатку. Нижче перераховані деякі інструменти та сервіси для різних аспектів безпеки:

Web Application Firewall (WAF):

Налаштуйте WAF для виявлення та блокування потенційно шкідливого трафіку, включаючи атаки типу SQL Injection, Cross-Site Scripting (XSS), ін'єкції та інші.

Security Headers:

Використовуйте HTTP заголовки безпеки, такі як Content Security Policy (CSP), Strict Transport Security (HSTS), і X-Content-Type-Options, для захисту від певних видів атак, таких як XSS та секретування інформації.

Dependency Scanning:

Використовуйте інструменти для сканування залежностей (наприклад, OWASP Dependency-Check) для виявлення вразливостей у використовуваних бібліотеках та сторонніх компонентах.

Static Application Security Testing (SAST):

Використовуйте інструменти SAST, такі як SonarQube чи Checkmarx, для виявлення потенційних вразливостей у вихідному коді додатку під час розробки.

Dynamic Application Security Testing (DAST):

Використовуйте інструменти DAST, такі як OWASP ZAP або Burp Suite, для виявлення вразливостей та атак на робочому додатку.

Monitoring and Logging Tools:

Використовуйте інструменти моніторингу та журналювання для виявлення аномальної активності та ведення журналів подій для аналізу безпеки.

Incident Response Tools:

Розробіть план відповіді на інциденти та використовуйте інструменти для швидкого виявлення, відновлення та аналізу інцидентів, наприклад, Security Information and Event Management (SIEM) системи.

Penetration Testing:

Використовуйте послуги та інструменти для проведення пенетраційного тестування, щоб виявити слабкі місця в системі та виправити їх перед тим, як їх скористається атакуюча сторона.

Container Security Tools:

Якщо ви використовуєте контейнеризацію (наприклад, Docker), розгляньте інструменти для аналізу та захисту контейнерів, такі як Clair або Anchore.

Continuous Integration/Continuous Deployment (CI/CD) Security:

Забезпечте безпеку CI/CD конвеєра та використовуйте інструменти, які дозволяють автоматично перевіряти безпеку коду перед впровадженням.

Забезпечення безпеки - це постійний процес, і комбінація різних заходів та інструментів допомагає створити ефективний захист для вашого додатку.

4.4 Висновки

У четвертому розділі був використаний метод тестування сірої скриньки для перевірки працездатності готового програмного продукту та його відповідності поставленому технічному завданню. Було проведено unit-тестування всіх компонент застосунку. Була визначена мінімальна конфігурація персонального комп'ютера, необхідна для коректної роботи програми, і розроблена інструкція користувача з користування веб-застосунком.

5 ЕКОНОМІЧНА ЧАСТИНА

Науково-технічна розробка має право на існування та впровадження, якщо вона відповідає вимогам часу, як в напрямку науково-технічного прогресу та і в плані економіки. Тому для науково-дослідної роботи необхідно оцінювати економічну ефективність результатів виконаної роботи.

Магістерська кваліфікаційна робота за темою «Розробка програмної системи моніторингу та контролю за благоустроєм» відноситься до науково-технічних робіт, які орієнтовані на виведення на ринок (або рішення про виведення науково-технічної розробки на ринок може бути прийнято у процесі проведення самої роботи), тобто коли відбувається так звана комерціалізація науково-технічної розробки. Цей напрямок є пріоритетним, оскільки результатами розробки можуть користуватися інші споживачі, отримуючи при цьому певний економічний ефект. Але для цього потрібно знайти потенційного інвестора, який би взявся за реалізацію цього проекту і переконати його в економічній доцільності такого кроку.

Для наведеного випадку нами мають бути виконані такі етапи робіт:

1) проведено комерційний аудит науково-технічної розробки, тобто встановлення її науково-технічного рівня та комерційного потенціалу;

2) розраховано витрати на здійснення науково-технічної розробки;

3) розрахована економічна ефективність науково-технічної розробки у випадку її впровадження і комерціалізації потенційним інвестором і проведено обґрунтування економічної доцільності комерціалізації потенційним інвестором.

5.1 Проведення комерційного та технологічного аудиту науково-технічної розробки

Метою проведення комерційного і технологічного аудиту дослідження за темою «Розробка програмної системи моніторингу і контролю за благоустроєм» є оцінювання науково-технічного рівня та рівня комерційного потенціалу розробки, створеної в результаті науково-технічної діяльності.

Оцінювання науково-технічного рівня розробки та її комерційного потенціалу рекомендується здійснювати із застосуванням 5-ти бальної системи оцінювання за 12-ма критеріями, наведеними в табл. 5.1 [32].

Таблиця 5.1 – Рекомендовані критерії оцінювання науково-технічного рівня і комерційного потенціалу розробки та бальна оцінка

Бали (за 5-ти бальною шкалою)					
	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 працездатність 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	Технічні та споживчі влас-тивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в	Технічні та споживчі властивості продукту значно кращі, ніж в
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	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					

Продовження таблиці 5.1

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
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 військово промисловому 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

Результати оцінювання науково-технічного рівня та комерційного потенціалу науково-технічної розробки потрібно звести до таблиці.

Таблиця 5.2 – Результати оцінювання науково-технічного рівня і комерційного потенціалу розробки експертами

Критерії	Експерт (ПІБ, посада)		
	1	2	3
	Бали:		
1. Технічна здійсненність концепції	4	4	3
2. Ринкові переваги (наявність аналогів)	5	4	4
3. Ринкові переваги (ціна продукту)	4	3	4
4. Ринкові переваги (технічні властивості)	3	4	3
5. Ринкові переваги (експлуатаційні витрати)	4	3	3
6. Ринкові перспективи (розмір ринку)	3	4	3
7. Ринкові перспективи (конкуренція)	3	3	4
8. Практична здійсненність (наявність фахівців)	3	3	3
9. Практична здійсненність (наявність фінансів)	3	3	3
10. Практична здійсненність (необхідність нових матеріалів)	3	3	4
11. Практична здійсненність (термін реалізації)	3	3	4
12. Практична здійсненність (розробка документів)	3	3	3
Сума балів	41	40	41
Середньоарифметична сума балів $СБ_с$	40,7		

За результатами розрахунків, наведених в таблиці 5.2, зробимо висновок щодо науково-технічного рівня і рівня комерційного потенціалу розробки. При цьому використаємо рекомендації, наведені в табл. 5.3 [32].

Таблиця 5.3 – Науково-технічні рівні та комерційні потенціали розробки

Середньоарифметична сума балів СБ , розрахована на основі висновків експертів	Науково-технічний рівень та комерційний потенціал розробки
41…48	Високий
31…40	Вище середнього
21…30	Середній
11…20	Нижче середнього
0…10	Низький

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Розробка програмної системи моніторингу і контролю за благоустроєм»

становить 40,7 балів, що, відповідно до таблиці 5.3, свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки вище середнього).

5.2 Розрахунок витрат на проведення науково-дослідної роботи

Витрати, пов'язані з проведенням науково-дослідної роботи на тему «Розробка методів і програмних засобів для підвищення ефективності процесу управління гуртожитком університету», під час планування, обліку і калькулювання собівартості науково-дослідної роботи групуємо за відповідними статтями.

5.2.1 Витрати на оплату праці

До статті «Витрати на оплату праці» належать витрати на виплату основної та додаткової заробітної плати керівникам відділів, лабораторій, секторів і груп, науковим, інженерно-технічним працівникам, конструкторам, технологам, креслярам, копіювальникам, лаборантам, робітникам, студентам, аспірантам та іншим працівникам, безпосередньо зайнятим виконанням конкретної теми, обчисленої за посадовими окладами, відрядними розцінками, тарифними ставками згідно з чинними в організаціях системами оплати праці.

Основна заробітна плата дослідників

Витрати на основну заробітну плату дослідників ($З_о$) розраховуємо у відповідності до посадових окладів працівників, за формулою **[Козловський, Лесько, Кавецький]**:

$$З_о = \sum_{i=1}^{k} \frac{M_{ni} \cdot t_i}{T_p}, \quad (5.1)$$

де k – кількість посад дослідників залучених до процесу досліджень;

M_{ni} – місячний посадовий оклад конкретного дослідника, грн;

t_i – число днів роботи конкретного дослідника, дні;

T_p – середнє число робочих днів в місяці, T_p=22 дні.

$$З_о = 50000{,}00 \cdot 63 / 22 = 143181{,}82 \text{ грн.}$$

Проведені розрахунки зведемо до таблиці.

Таблиця 5.4 – Витрати на заробітну плату дослідників

Найменування посади	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Число днів роботи	Витрати на заробітну плату, грн
Керівник проекту	50000	2272,73	63	143181,82
Інженер-розробник програмного забезпечення	38000	1727,27	63	108818,18
Консультант	15000	681,82	10	6818,18
Всього				258818,18

Основна заробітна плата робітників

Витрати на основну заробітну плату робітників ($З_р$) за відповідними найменуваннями робіт НДР розраховуємо за формулою:

$$З_р = \sum_{i=1}^{n} C_i \cdot t_i \,, \tag{5.2}$$

де C_i – погодинна тарифна ставка робітника відповідного розряду, за виконану відповідну роботу, грн/год;

t_i – час роботи робітника при виконанні визначеної роботи, год.

Погодинну тарифну ставку робітника відповідного розряду C_i можна визначити за формулою:

$$C_i = \frac{M_M \cdot K_i \cdot K_c}{T_p \cdot t_{зм}} \,, \tag{5.3}$$

де M_M – розмір прожиткового мінімуму працездатної особи, або мінімальної місячної заробітної плати (в залежності від діючого законодавства), приймемо $Мм$=6700,00 грн;

K_i – коефіцієнт міжкваліфікаційного співвідношення для встановлення тарифної ставки робітнику відповідного розряду (табл. Б.2, додаток Б) [33];

K_c – мінімальний коефіцієнт співвідношень місячних тарифних ставок робітників першого розряду з нормальними умовами праці виробничих об’єднань і підприємств до законодавчо встановленого розміру мінімальної заробітної плати.

$T_р$ – середнє число робочих днів в місяці, приблизно $T_р$ = 22 дні;

$t_{зм}$ – тривалість зміни, год.

$С_1 = 6700,00 \cdot 1,10 \cdot 1,65 / (22 \cdot 8) = 69,09$ грн.
$З_{р1} = 69,09 \cdot 10,00 = 690,94$ грн.

Таблиця 5.5 – Величина витрат на основну заробітну плату робітників

Найменування робіт	Тривалість роботи, год	Розряд роботи	Тарифний коефіцієнт	Погодинна тарифна ставка, грн	Величина оплати на робітника грн
Підговка електронно-обчислювальної техніки	10	2	1,1	69,09	690,94
Підготовка робочого місця дослідника	2	2	1,1	69,09	138,19
Налагодження безперебійного обладнання	10	2	1,1	69,09	690,94
Всього					1520,06

Додаткова заробітна плата дослідників та робітників

Додаткову заробітну плату розраховуємо як 10 … 12% від суми основної заробітної плати дослідників та робітників за формулою:

$$З_{дод} = (З_о + З_р) \cdot \frac{Н_{дод}}{100\%}, \quad (5.4)$$

де $Н_{дод}$ – норма нарахування додаткової заробітної плати. Приймемо 11%.

$З_{дод} = (258818,18 + 1520,06) \cdot 11 / 100\% = 28637,21$грн.

5.2.2 Відрахування на соціальні заходи

Нарахування на заробітну плату дослідників та робітників розраховуємо як 22% від суми основної та додаткової заробітної плати дослідників і робітників за формулою:

$$З_н = (З_о + З_р + З_{дод}) \cdot \frac{Н_{зп}}{100\%} \quad (5.5)$$

де $Н_{зп}$ – норма нарахування на заробітну плату. Приймаємо 22%.

Зн = (258818,18+ 1520,06+ 28637,21) · 22 / 100% = 63574,59926грн.

5.2.3 Сировина та матеріали

До статті «Сировина та матеріали» належать витрати на сировину, основні та допоміжні матеріали, інструменти, пристрої та інші засоби і предмети праці, які придбані у сторонніх підприємств, установ і організацій та витрачені на проведення досліджень.

Витрати на матеріали (*М*), у вартісному вираженні розраховуються окремо по кожному виду матеріалів за формулою:

$$M = \sum_{j=1}^{n} H_j \cdot Ц_j \cdot K_j - \sum_{j=1}^{n} B_j \cdot Ц_{вj}, \quad (5.6)$$

де H_j – норма витрат матеріалу *j*-го найменування, кг;

n – кількість видів матеріалів;

$Ц_j$ – вартість матеріалу *j*-го найменування, грн/кг;

K_j – коефіцієнт транспортних витрат, (K_j = *1,1 ... 1,15*);

B_j – маса відходів *j*-го найменування, кг;

$Ц_{вj}$ – вартість відходів *j*-го найменування, грн/кг.

$М_1 = 2 \cdot 200{,}00 \cdot 1{,}1 - 0{,}000 \cdot 0{,}00 = 440{,}0$ грн.

Проведені розрахунки зведемо до таблиці.

Таблиця 5.6 – Витрати на матеріали

Найменування матеріалу, марка, тип, сорт	Ціна за од	Норма витрат, од	Величина відходів, кг	Ціна відходів, грн/кг	Вартість витраченого матеріалу, грн
Офісний папір А4-500-80	200	2	0	0	440
Канцелярське приладдя (набір)	150	2	0	0	330
Книга "Креативне містотворення. Його сила та можливості"	379	1	0	0	379
					1149

5.2.4 Розрахунок витрат на комплектуючі

Витрати на комплектуючі *($К_в$)*, які використовують при проведенні НДР на тему «Розробка програмної системи моніторингу і контролю за благоустроєм відсутні.

5.2.5 Спецустаткування для наукових (експериментальних) робіт

До статті «Спецустаткування для наукових (експериментальних) робіт» належать витрати на виготовлення та придбання спецустаткування необхідного для проведення досліджень, також витрати на їх проектування, виготовлення, транспортування, монтаж та встановлення.

Балансову вартість спецустаткування розраховуємо за формулою:

$$B_{спец} = \sum_{i=1}^{k} Ц_i \cdot C_{пр.i} \cdot K_i \quad , \tag{5.7}$$

де $Ц_i$ – ціна придбання одиниці спецустаткування даного виду, марки, грн;

$C_{пр.i}$ –кількість одиниць устаткування відповідного найменування, які придбані для проведення досліджень, шт.;

K_i – коефіцієнт, що враховує доставку, монтаж, налагодження устаткування тощо, (K_i = *1,10…1,12);*

k – кількість найменувань устаткування.

$B_{спец}$ *= 40000· 1 · 1,1 = 44000 грн.*

Отримані результати зведемо до таблиці:

Таблиця 5.7 – Витрати на придбання спецустаткування по кожному виду

Найменування устаткування	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
EcoFlow Delta 2	1	40 000	44000
Всього			44000

5.2.6 Програмне забезпечення для наукових (експериментальних) робіт

До статті «Програмне забезпечення для наукових (експериментальних) робіт» належать витрати на розробку та придбання спеціальних програмних засобів і програмного забезпечення, (програм, алгоритмів, баз даних) необхідних для проведення досліджень, також витрати на їх проектування, формування та встановлення.

Балансову вартість програмного забезпечення розраховуємо за формулою:

$$B_{прг} = \sum_{i=1}^{k} Ц_{iпрг} \cdot C_{прг.i} \cdot K_i \ , \qquad (5.8)$$

де $Ц_{iпрг}$ – ціна придбання одиниці програмного засобу даного виду, грн;

$C_{прг.i}$ – кількість одиниць програмного забезпечення відповідного найменування, які придбані для проведення досліджень, шт.;

K_i – коефіцієнт, що враховує інсталяцію, налагодження програмного засобу тощо, (K_i = *1,10…1,12);*

k – кількість найменувань програмних засобів.

$B_{прг} = 6000 \cdot 1 \cdot 1{,}1 = 6720$ *грн.*

Отримані результати зведемо до таблиці:

Таблиця 5.8 – Витрати на придбання програмних засобів по кожному виду

Найменування програмного засобу	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
Google Cloud Services	1	6000	6720
Всього			6720

5.2.7 Амортизація обладнання, програмних засобів та приміщень

В спрощеному вигляді амортизаційні відрахування по кожному виду обладнання, приміщень та програмному забезпеченню тощо, розраховуємо з використанням прямолінійного методу амортизації за формулою:

$$A_{обл} = \frac{Ц_б}{T_в} \cdot \frac{t_{вик}}{12}, \quad (5.9)$$

де $Ц_б$ – балансова вартість обладнання, програмних засобів, приміщень тощо, які використовувались для проведення досліджень, грн;

$t_{вик}$ – термін використання обладнання, програмних засобів, приміщень під час досліджень, місяців;

$T_в$ – строк корисного використання обладнання, програмних засобів, приміщень тощо, років.

$A_{обл} = (70000{,}00 \cdot 3) / (5 \cdot 12) = 3500$ *грн.*

Проведені розрахунки зведемо до таблиці.

Таблиця 5.9 – Амортизаційні відрахування по кожному виду обладнання

Найменування обладнання	Балансова вартість, грн	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн
MacBook Pro 14 2021	70 000	5	3	3500,00
ПК	20000	5	3	1000,00
Google Cloud Services	6 720	3	3	560,00
Комп’ютерний стіл	3000	5	3	150,00
Оргтехніка	6000	4	2	250,00
Приміщення лабораторії	212000	22	3	2409,09
Всього				7869,09

5.2.8 Паливо та енергія для науково-виробничих цілей

Витрати на силову електроенергію ($B_е$) розраховуємо за формулою:

$$B_е = \sum_{i=1}^{n} \frac{W_{уі} \cdot t_i \cdot Ц_е \cdot K_{впі}}{\eta_i}, \quad (5.10)$$

де W_{yi} – встановлена потужність обладнання на визначеному етапі розробки, кВт;

t_i – тривалість роботи обладнання на етапі дослідження, год;

$Ц_е$ – вартість 1 кВт-години електроенергії, грн; (вартість електроенергії визначається за даними енергопостачальної компанії), приймемо $Ц_е = 7,50$ грн;

$K_{вп i}$ – коефіцієнт, що враховує використання потужності, $K_{вп i} < 1$;

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

$B_е = 0,2 \cdot 480,0 \cdot 7,50 \cdot 0,95 / 0,97 = 2820,62$ *грн.*

Проведені розрахунки зведемо до таблиці.

Таблиця 5.10 – Витрати на електроенергію

Найменування обладнання	Встановлена потужність, кВт	Тривалість роботи, год	Сума, грн
MacBook Pro 14 2021	0,6	504	2221,24
ПК	0,2	504	740,41
Робоче місце дослідника	0,15	504	555,31
Оргтехніка	0,45	20	66,11
Всього			3583,07

5.2.9 Службові відрядження

До статті «Службові відрядження» дослідної роботи належать витрати на відрядження штатних працівників, працівників організацій, які працюють за договорами цивільно-правового характеру, аспірантів, зайнятих розробленням досліджень, відрядження, пов’язані з проведенням випробувань машин та приладів, а також витрати на відрядження на наукові з’їзди, конференції, наради, пов’язані з виконанням конкретних досліджень.

Витрати за статтею «Службові відрядження» розраховуємо як 20…25% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{св} = (З_о + З_р) \cdot \frac{Н_{св}}{100\%}, \quad (5.11)$$

де $Н_{св}$ – норма нарахування за статтею «Службові відрядження», приймемо $Н_{св}$ = 20%.

$В_{св}$ = (258818,18+ 1520,06) · 20 / 100% = 52067,65грн.

5.2.10 Витрати на роботи, які виконують сторонні підприємства, установи і організації

Витрати за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації» розраховуємо як 30…45% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{сп} = (З_о + З_р) \cdot \frac{Н_{сп}}{100\%}, \quad (5.12)$$

де $Н_{сп}$ – норма нарахування за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації», приймемо $Н_{сп}$= 40%.

$В_{сп}$ = (258818,18+ 1520,06) · 40 / 100% = 104135,30грн.

5.2.11 Інші витрати

До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками.

Витрати за статтею «Інші витрати» розраховуємо як 50…100% від суми основної заробітної плати дослідників та робітників за формулою:

$$I_в = (З_о + З_р) \cdot \frac{Н_{ів}}{100\%}, \quad (5.13)$$

де $Н_{ів}$ – норма нарахування за статтею «Інші витрати», приймемо $Н_{ів}$ = 60%.

$І_в$ = (258818,18+ 1520,06) · 60 / 100% = 156202,95грн.

5.2.12 Накладні (загальновиробничі) витрати

До статті «Накладні (загальновиробничі) витрати» належать: витрати, пов’язані з управлінням організацією; витрати на винахідництво та раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів; витрати, пов’язані з набором робочої сили; витрати на оплату послуг банків;

витрати, пов'язані з освоєнням виробництва продукції; витрати на науково-технічну інформацію та рекламу та ін.

Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуємо як 100…150% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%}\ , \qquad (5.14)$$

де $Н_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати», приймемо $Н_{нзв}$ = 120%.

$В_{нзв}$ = (258818,18+ 1520,06) · 100 / 120% = 312 405,89грн.

Витрати на проведення науково-дослідної роботи розраховуємо як суму всіх попередніх статей витрат за формулою:

$$B_{заг} = З_о + З_р + З_{дод} + З_н + М + К_в + В_{спец} + В_{прг} + А_{обл} + В_е + В_{св} + В_{сп} + І_в + В_{нзв}. \qquad (5.15)$$

$В_{заг}$= 1040682,99грн.

Загальні витрати *ЗВ* на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховується за формулою:

$$ЗВ = \frac{B_{заг}}{\eta}, \qquad (5.16)$$

де η - коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи, приймемо η=0,9.

ЗВ = 1040682,99/ 0,9 = 1156314,44грн.

5.3 Розрахунок економічної ефективності науково-технічної розробки при її можливій комерціалізації потенційним інвестором

В ринкових умовах узагальнюючим позитивним результатом, що його може отримати потенційний інвестор від можливого впровадження результатів тієї чи іншої науково-технічної розробки, є збільшення у потенційного інвестора величини чистого прибутку.

Результати дослідження передбачають комерціалізацію протягом 3-х років реалізації на ринку.

В цьому випадку майбутній економічний ефект буде формуватися на основі таких даних:

ΔN – збільшення кількості споживачів продукту, у періоди часу, що аналізуються, від покращення його певних характеристик;

1-й рік – 12 користувачів;

2-й рік –9 користувачів;

3-й рік – 7 користувачів.

N – кількість споживачів які використовували аналогічний продукт у році до впровадження результатів нової науково-технічної розробки, приймемо 5 користувачів;

$Ц_б$ – вартість програмного продукту у році до впровадження результатів розробки, приймемо 300000,00 грн;

$\pm\Delta Ц_о$ – зміна вартості програмного продукту від впровадження результатів науково-технічної розробки, приймемо 100000,00 грн.

Можливе збільшення чистого прибутку у потенційного інвестора $\Delta П_i$ для кожного із 3-х років, протягом яких очікується отримання позитивних результатів від можливого впровадження та комерціалізації науково-технічної розробки, розраховуємо за формулою [31]:

$$\Delta П_i = (\pm\Delta Ц_о \cdot N + Ц_о \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}), \quad (5.17)$$

де λ – коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість. У 2023 році ставка податку на додану вартість складає 20%, а коефіцієнт $\lambda = 0{,}8333$;

ρ – коефіцієнт, який враховує рентабельність інноваційного продукту. Приймемо ρ =40%;

ϑ – ставка податку на прибуток, який має сплачувати потенційний інвестор, у 2023 році ϑ =*18%*;

Збільшення чистого прибутку 1-го року:

$\Delta П_1$ =(25·100000,00+400000·12)·0,83·0,4·(1-0,18/100%)= 1994535,2грн.

Збільшення чистого прибутку 2-го року:

$\Delta П_2$ =(25·100000,00+400000·(12+9))·0,83·0,4(1-0,18/100%)= 2978141,6 грн.

Збільшення чистого прибутку 3-го року:

$\Delta П_3$ =(25·100000,00+400000·(12+9+7))·0,83·0,4·(1-0,18/100%)= 3743168,8 грн.

Приведена вартість збільшення всіх чистих прибутків *ПП*, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки:

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \qquad (5.18)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному з років, протягом яких виявляються результати впровадження науково-технічної розробки, грн;

T – період часу, протягом якого очікується отримання позитивних результатів від впровадження та комерціалізації науково-технічної розробки, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, τ =0,15;

t – період часу (в роках) від моменту початку впровадження науково-технічної розробки до моменту отримання потенційним інвестором додаткових чистих прибутків у цьому році.

$ПП$ = 1994535,2/(1+0,15)^1+ 2978141,6/(1+0,15)^2+ 3743168,8/(1+0,15)^3= 6447475,59грн.

Величина початкових інвестицій PV, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки:

$$PV = k_{інв} \cdot ЗВ, \quad (5.19)$$

де $k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію, приймаємо $k_{інв}$=2;

$ЗВ$ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, приймаємо 1156314,44грн.

$PV = k_{інв} \cdot ЗВ$ = 2· 1156314,44= 2312628,88грн.

Абсолютний економічний ефект $E_{абс}$ для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$E_{абс} = ПП - PV \quad (5.20)$$

де $ПП$ – приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації науково-технічної розробки, 6447475,59грн;

PV – теперішня вартість початкових інвестицій, 2312628,88грн.

$E_{абс} = ПП - PV$ = 6447475,59 - 2312628,88= 4134846,72грн.

Внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$E_в = \sqrt[Tж]{1 + \frac{E_{абс}}{PV}} - 1, \quad (5.21)$$

де $E_{абс}$ – абсолютний економічний ефект вкладених інвестицій, 4134846,72грн;

PV – теперішня вартість початкових інвестицій, 2312628,88грн;

$T_{ж}$ – життєвий цикл науково-технічної розробки, тобто час від початку її розробки до закінчення отримування позитивних результатів від її впровадження, 3 роки.

$$E_{в} = \sqrt[Tж]{1 + \frac{E_{абс}}{PV}} - 1 = (1+4134846{,}72/2312628{,}88)^{1/3} = 0{,}41.$$

Мінімальна внутрішня економічна дохідність вкладених інвестицій $\tau_{мін}$:

$$\tau_{мін} = d + f, \qquad (5.22)$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2023 році в Україні d =0,1;

f – показник, що характеризує ризикованість вкладення інвестицій, приймемо 0,18.

$\tau_{мін}$ = 0,1+0,18 = 0,28 < 0,45 свідчить про те, що внутрішня економічна дохідність інвестицій $E_{в}$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки вища мінімальної внутрішньої дохідності. Тобто інвестувати в науково-дослідну роботу за темою «Розробка програмної системи моніторингу і контролю за благоустроєм» доцільно.

Період окупності інвестицій $T_{ок}$ які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$T_{ок} = \frac{1}{E_{в}}, \qquad (5.23)$$

де $E_{в}$ – внутрішня економічна дохідність вкладених інвестицій.

$T_{ок}$ = 1 / 0,41 = 2,45 року.

$T_{ок}$ < 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

5.4 Висновки

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою Розробка програмної системи моніторингу і контролю за благоустроєм» становить 40,7 балів, що, свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки вище середнього).

Також термін окупності становить 2,45 р., що менше 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

Отже можна зробити висновок про доцільність проведення науково-дослідної роботи за темою «Розробка програмної системи моніторингу і контролю за благоустроєм».

ВИСНОВКИ

Під час виконання магістерської кваліфікаційної роботи було розроблено методи і засоби реалізації програмної системи “БлагоМіст” з моніторингу та контролю за благоустроєм. Робота оформлена згідно методичних вказівок [34].

Було проведено огляд поточного стану даної теми з огляду на сучасні реалії. Проведено докладний аналіз основних аналогів, визначено їхні характеристики та слабкі сторони. У результаті цього аналізу було виокремлено мову програмування JavaScript. Для розробки було обрано інтегроване середовище розробки Microsoft Visual Studio Code. Технологіями для розробки додатку було обрано MongoDB для бази даних, Express js та Node js для серверної частини і також для клієнтської частини було обрано фреймворк Angular.

Розроблено програмну систему “БлагоМіст”, яка включає модуль авторизації користувача у системі; клієнтський модуль для подачі заявок; модуль розгляду заявок; адміністративний модуль керування запитом; модуль інтерактивної мапи.

У процесі виконання роботи було розроблено алгоритм кластеризації для програмної системи “БлагоМіст”. Розроблено моделі та алгоритми програмної системи, вибрано кольорову гаму, проведено тестування системи.

Подальшого розвитку дістав алгоритм кластеризації, який на відміну від існуючих, має більш точне визначення відстаней між маркерами та динамічно об’єднує їх у групи без перезапису інформації у кожен існуючий маркер.

Тестування програми довело повну працездатність даного програмного продукту та відповідність поставленому технічному завданню. Розроблено інструкції користувача.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Русланов І. В. Інженерний благоустрій територій / І. В. Русланов, Г. М. Шульга. – Львівська політехніка, 2020. – 260 с.
2. Річард Флорида. Книга урбанізму. Чому міста роблять нас нещасними. – Наш формат, 2019. – 320 с. – ISBN 9786177682973.
3. Іванюта П. В. Управління ресурсами та витратами. / Іванюта П. В. Лугівська О. П. – Центр учбової літератури, 2019. – 320 с. – ISBN 9789663649702.
4. Чарльз Лендрі. Креативне містостворення: його сила і можливості. – Фоліо, 2020. – 252 с. – ISBN 9789660393196.
5. Клименко Є. В. Будівельні конструкції / Є. В. Клименко, В. С. Дорофєєв. – Центр учбової літератури, 2021. – 426 с. – ISBN 9786176730682.
6. Расс Унгер. UX-дизайн / Унгер Р., Чендлер До.. – Центр учбової літератури, 2016. – 336 с. – ISBN 9786176734535.
7. Громадський бюджет [Електронний ресурс] – Режим доступу до ресурсу: https://budget.e-dem.ua/.
8. Електронні петиції [Електронний ресурс] – Режим доступу до ресурсу: https://petition.president.gov.ua/.
9. Бородкіна І. Л. Web-технології та Web-дизайн : застосування мови HTML для створення електронних ресурсів / Бородкіна І. Л., Бородкін Г. О. – Ліра-К, 2020. – 212 с. – ISBN 9786177844142.
10. Макдональд М. Веб-розробка. / Макдональд М. – Фоліо, 2021. – 640 с. – ISBN 9786154846542.
11. Google APIs [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Google_APIs.
12. Створення інтерактивної мапи [Електронний ресурс] – Режим доступу до ресурсу: https://magneticonemt.com/stvorennnya-interativnih-kart/.

13. Аутентифікація та авторизація [Електронний ресурс] – Режим доступу до ресурсу:https://it-rating.ua/autentifikatsiya-ta-avtorizatsiya-zahist-danih-koristuvachiv/.
14. Кантелон М. Node js / Кантелон М., Хартер М., Головайчук Т. – Центр учбової літератури, 2021. – 345 с. – ISBN 9786174520425
15. MongoDB Compass [Електронний ресурс] – Режим доступу до ресурсу: https://www.mongodb.com/products/tools/compass.
16. Кластерний аналіз [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Кластерний_аналіз.
17. Метод k-найближчих сусідів [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/ Метод_k-найближчих_сусідів.
18. Евклідова відстань. [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Евклідова_відстань.
19. Манхеттенська відстань. [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Мангеттенська_метрика.
20. Хаверсайн відстань. [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Хаверсайн_відстань.
21. Ерік Шмідт. Як працює Google API. / Ерік Ш., Джонатан Р., Алан І. – КМ-Букс, 2016. – 304 с. – ISBN 9786135242474.
22. Крістофер Бішоп. Pattern recognition and machine learning. / Крістофер Б. – Кембрідж, 2006. – 234 с. – ISBN 9783435245374.
23. Мартин Фаулер. UML основи / Мартин Ф., Кендалл С. – Символ-Плюс, 2018. – 192 с. – ISBN 9783335245354.
24. Ерік Фрімен. Head First. Програмування на JavaScript / Ерік Фрімен, Елізабет Робсон. – Фабула, 2022. – 672 с. – ISBN 9786175220474.
25. Node js [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Node.js.
26. Python [Електронний ресурс] – Режим доступу до ресурсу: https://www.python.org/.

27. Java [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Java_(programming_language).
28. Visual Studio Code.[Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Visual_Studio_Code.
29. Романюк О. Р. Організація баз даних і знань / О. Н. Романюк, Т.О. Савчук // Навчальний посібник. – Вінниця: «УНІВЕРСУМ-Вінниця», 2003. – 123 с.
30. Адам Фрімен. Angular для професіоналів. / Адам Фрімен. – Апресс, 2021. – 800 с. – ISBN 9724173520474.
31. Лайза Кріспін. Гнучке програмування. / Лайза Кріспін, Джанет Грегори. – Вільямс, 2018. – 464 с. – ISBN 97243453520474.
32. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт / Уклад. : В. О. Козловський, О. Й. Лесько, В. В. Кавецький. Вінниця : ВНТУ, 2021. 42 с.
33. Кавецький В. В. Економічне обґрунтування інноваційних рішень: практикум / В. В. Кавецький, В. О. Козловський, І. В. Причепа. Вінниця : ВНТУ, 2016. 113 с.
34. Романюк О. Н. Методичні вказівки до виконання магістерської кваліфікаційної роботи для студентів спеціальності 121 «Інженерія програмного забезпечення» / уклад. : О. Н. Романюк, Г. О. Черноволик. – Вінниця : ВНТУ, 2022. – 50 с.

ДОДАТКИ

Додаток А. Технічне завдання

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ
д.т.н., проф. О. Н. Романюк
"19" вересня 2023 р.

Технічне завдання
на магістерську кваліфікаційну роботу «Розробка програмної системи моніторингу та контролю за благоустроєм» за спеціальністю
121 – Інженерія програмного забезпечення

Керівник магістерської кваліфікаційної роботи:
к.т.н., доц. Г. Б. Ракитянська
"19" вересня 2023 р.
Виконав:
студент гр.3ПІ-22м Р. В. Кондрук
"19" вересня 2023 р.

Вінниця – 2023 року

1. Найменування та галузь застосування

Магістерська кваліфікаційна робота: «Розробка програмної системи моніторингу та контролю за благоустроєм».

Галузь застосування – комп'ютерні навчальні системи.

2. Підстава для розробки.

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

3. Мета та призначення розробки.

Метою роботи є підвищення ефективності роботи програмної системи моніторингу та контролю за благоустроєм, що дозволить користувачам-громадянам максимально продуктивно комунікувати з муніципалітетами.

Призначення роботи – розробка методів і засобів реалізації управління благоустроєм шляхом комунікації громадськості та місцевого самоврядування.

3 Вихідні дані для проведення НДР

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

1. Річард Флорида. Книга урбанізму. Чому міста роблять нас нещасними. – Наш формат, 2019. – 320 с. – ISBN 9786177682973.
2. Кантелон М. Node js / Кантелон М., Хартер М., Головайчук Т. – Центр учбової літератури, 2021. – 345 с. – ISBN 9786174520425
3. Мартин Фаулер. UML основи / Мартин Ф., Кендалл С. – Символ-Плюс, 2018. – 192 с. – ISBN 9783335245354.
4. Адам Фрімен. Angular для професіоналів. / Адам Фрімен. – Апресс, 2021. – 800 с. – ISBN 9724173520474.

4. Технічні вимоги

Вихідні дані до роботи: алгоритм кластеризації маркерів на мапі, метод управління проектами з благоустрою; база даних – MongoDB; фреймоворк клієнтської частини – Angular; фреймворк серверної частини – Node js; вихідні дані – результат кластеризації маркерів.

5. Конструктивні вимоги.

Конструкція застосунку повинна відповідати естетичним та практичним вимогам, повинна бути зручною в обслуговуванні та керуванні.

Графічна та текстова документація повинна відповідати діючим стандартам України.

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

– пояснювальна записка до МКР;

– технічне завдання;

– лістинги програми.

7. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

8. Стадії та етапи розробки:

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи
1	Обґрунтування вибору методу розробки та постановка задачі дослідження	20.09.2023 – 30.09.2023
2	Розробка методу та моделей програмної системи моніторингу та контролю за благоустроєм	01.10.2023 – 17.10.2023
3	Розробка програмних компонент додатку	18.10.2023 – 04.11.2023
4	Тестування програмного додатку	05.11.2023 – 21.11.2023
5	Економічна частина	22.11.2023 – 01.12.2023

9. 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. Протокол перевірки на плагіат

Додаток Б. Протокол перевірки на плагіат

ПРОТОКОЛ ПЕРЕВІРКИ НАВЧАЛЬНОЇ (КВАЛІФІКАЦІЙНОЇ) РОБОТИ

Назва роботи: **Розробка програмної системи моніторингу та контролю за благоустроєм**

Тип роботи: магістерська кваліфікаційна робота

Підрозділ : кафедра програмного забезпечення, ФІТКІ, ЗПІ – 22м

Науковий керівник:

Unicheck	
Оригінальність	**92,6%**
Схожість	7%

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 Кондрук Р. В.

Керівник роботи Ракитянська Г. Б.

Додаток В. Лістинг програми

Project.scheme.ts

```
const mongoose = require('mongoose');

const projectSchema = new mongoose.Schema({
  title: String,
  description: String,
  location: {
    type: { type: String },
    coordinates: [Number]
  }
});

projectSchema.index({ location: '2dsphere' });

const Project = mongoose.model('Project', projectSchema);

module.exports = Project;
```

User.scheme.ts

```
const mongoose = require('mongoose');

const userSchema = new mongoose.Schema({
  login: { type: String, required: true, unique: true },
  password: { type: String, required: true },
  name: { type: String, required: true },
  age: { type: Number, required: true },
});

const User = mongoose.model('User', userSchema);

module.exports = User;
```

ChangeRequest.scheme.ts

```
const mongoose = require('mongoose');

const changeRequestSchema = new mongoose.Schema({
  description: { type: String, required: true },
  author: { type: String, required: true },
  status: { type: String, enum: ['Pending', 'Approved', 'Rejected'], default: 'Pending' },
```

```
});

const ChangeRequest = mongoose.model('ChangeRequest', changeRequestSchema);

module.exports = ChangeRequest;
```

Message.scheme.ts

```
const mongoose = require('mongoose');

const messageSchema = new mongoose.Schema({
  author: { type: String, required: true },
  sentDate: { type: Date, default: Date.now },
  text: { type: String, required: true },
  projectId: { type: mongoose.Schema.Types.ObjectId, required: true },
});

const Message = mongoose.model('Message', messageSchema);

module.exports = Message;
```

Home.component.ts

```
import { Component, OnInit } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import * as L from 'leaflet';

@Component({
  selector: 'app-home',
  templateUrl: './home.component.html',
  styleUrls: ['./home.component.css']
})
export class HomeComponent implements OnInit {
  projects: any[] = [];
  map: any;

  constructor(private http: HttpClient) { }

  ngOnInit(): void {
    this.http.get<any[]>('http://localhost:3000/api/projects').subscribe(projects => {
      this.projects = projects;
      this.clusterProjects();
    });
  }
```

```
  clusterProjects(): void {
    // Extract coordinates from projects
    const coordinates = this.projects.map(project => [
      project.location.coordinates[1],
      project.location.coordinates[0]
    ]);


    const k = 3;
    const clusters = this.kMeans(coordinates, k);

    // Initialize the map
    this.map = L.map('map').setView([0, 0], 2);

    L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {
      attribution: '© OpenStreetMap contributors'
    }).addTo(this.map);

    this.projects.forEach((project, index) => {
      const clusterIndex = clusters[index];
      const color = this.getColor(clusterIndex);
      const marker = L.marker([project.location.coordinates[1], project.location.coordinates[0]], { icon:
L.divIcon({ className: 'custom-marker', html: `<div style="background-color:${color}"
class="marker"></div>` }) })
        .addTo(this.map)
        .bindPopup(project.title);
    });
  }

  kMeans(data: number[][], k: number): number[] {
    const centroids = this.getRandomCentroids(data, k);

    let prevClusters: number[];
    let currentClusters: number[];

    do {
      currentClusters = data.map(point => this.assignToCluster(point, centroids));

      centroids.forEach((centroid, clusterIndex) => {
        const clusterPoints = data.filter((point, index) => currentClusters[index] === clusterIndex);
        if (clusterPoints.length > 0) {
          centroids[clusterIndex] = this.calculateMean(clusterPoints);
        }
      });

      if (this.areArraysEqual(prevClusters, currentClusters)) {
```

```
      break;
    }

    prevClusters = [...currentClusters];
  } while (true);

  return currentClusters;
}

getRandomCentroids(data: number[][], k: number): number[][] {
  const centroids = [];
  const dataCopy = [...data];

  for (let i = 0; i < k; i++) {
    const randomIndex = Math.floor(Math.random() * dataCopy.length);
    centroids.push(dataCopy.splice(randomIndex, 1)[0]);
  }

  return centroids;
}

assignToCluster(point: number[], centroids: number[][]): number {
  let minDistance = Number.MAX_VALUE;
  let minClusterIndex = -1;

  centroids.forEach((centroid, clusterIndex) => {
    const distance = this.calculateManhattanDistance(point, centroid);
    if (distance < minDistance) {
      minDistance = distance;
      minClusterIndex = clusterIndex;
    }
  });

  return minClusterIndex;
}

calculateManhattanDistance(point1: number[], point2: number[]): number {
  return point1.reduce((acc, val, index) => acc + Math.abs(val - point2[index]), 0);
}

calculateMean(points: number[][]): number[] {
  const numDimensions = points[0].length;
  const mean = Array(numDimensions).fill(0);

  points.forEach(point => {
    point.forEach((coord, index) => {
```

```
        mean[index] += coord;
      });
    });

    return mean.map(coord => coord / points.length);
  }

  areArraysEqual(arr1: any[], arr2: any[]): boolean {
    return arr1.length === arr2.length && arr1.every((value, index) => value === arr2[index]);
  }

  getColor(clusterIndex: number): string {
    const colors = ['#FF0000', '#00FF00', '#0000FF'];
    return colors[clusterIndex % colors.length];
  }
}
```

Home.component.html

```
<div class="container">
    <h2>Список проектів</h2>
    <ul>
      <li *ngFor="let project of projects">{{ project.title }}</li>
    </ul>

    <div id="map" style="height: 400px;"></div>
  </div>
```

Index.ts

```
const express = require('express');
const mongoose = require('mongoose');
const bodyParser = require('body-parser');
const cors = require('cors');

const app = express();
const port = 2444;

app.use(bodyParser.json());
app.use(cors());

mongoose.connect('mongodb://localhost:27017/your-database', { useNewUrlParser: true,
useUnifiedTopology: true });
const db = mongoose.connection;
```

```
db.on('error', console.error.bind(console, 'Connection error:'));
db.once('open', () => {
  console.log('Connected to the database');
});
app.get('/api/projects', projectsController.getProjects);
app.get('/api/projects/:id', projectsController.getProjectById);
app.post('/api/projects', projectsController.createProject);
app.put('/api/projects/:id', projectsController.updateProject);
app.delete('/api/projects/:id', projectsController.deleteProject);

app.listen(port, () => {
  console.log(`Server is running on port ${port}`);
});
```

Projects.controller.ts

```
const Project = require('./project.model');

exports.getProjects = async (req, res) => {
  try {
    const { page = 1, limit = 10 } = req.query;
    const projects = await Project.find()
      .skip((page - 1) * limit)
      .limit(limit);
    res.json(projects);
  } catch (error) {
    res.status(500).json({ message: error.message });
  }
};

exports.getProjectById = async (req, res) => {
  try {
    const project = await Project.findById(req.params.id);
    if (project) {
      res.json(project);
    } else {
      res.status(404).json({ message: 'Project not found' });
    }
  } catch (error) {
    res.status(500).json({ message: error.message });
  }
};

exports.createProject = async (req, res) => {
  const project = new Project(req.body);
```

```
  try {
    const savedProject = await project.save();
    res.status(201).json(savedProject);
  } catch (error) {
    res.status(400).json({ message: error.message });
  }
};

exports.updateProject = async (req, res) => {
  try {
    const updatedProject = await Project.findByIdAndUpdate(req.params.id, req.body, { new: true });
    res.json(updatedProject);
  } catch (error) {
    res.status(400).json({ message: error.message });
  }
};

exports.deleteProject = async (req, res) => {
  try {
    await Project.findByIdAndDelete(req.params.id);
    res.json({ message: 'Project deleted' });
  } catch (error) {
    res.status(500).json({ message: error.message });
  }
};
```

List.component.html

```
<div class="container">
  <h2>Список проектів</h2>

  <table class="table table-striped">
    <thead>
      <tr>
        <th (click)="sortProjects('title')">Назва проекту</th>
        <th (click)="sortProjects('status')">Статус</th>
        <th (click)="sortProjects('votes')">Голосів</th>
        <th (click)="sortProjects('createdAt')">Дата створення</th>
      </tr>
    </thead>
    <tbody>
      <tr *ngFor="let project of sortedProjects">
        <td>{{ project.title }}</td>
        <td>{{ project.status }}</td>
        <td>{{ project.votes }}</td>
        <td>{{ project.createdAt | date:'medium' }}</td>
```

```
    </tr>
   </tbody>
  </table>
</div>
```

List.component.ts

```
import { Component, OnInit } from '@angular/core';
import { HttpClient } from '@angular/common/http';

@Component({
  selector: 'app-project-list',
  templateUrl: './project-list.component.html',
  styleUrls: ['./project-list.component.css']
})
export class ProjectListComponent implements OnInit {
  projects: any[] = [];
  sortedProjects: any[] = [];
  sortDirection: number = 1;

  constructor(private http: HttpClient) { }

  ngOnInit(): void {
    this.http.get<any[]>('http://localhost:3000/api/projects').subscribe(projects => {
      this.projects = projects;
      this.sortProjects('createdAt'); // Default sorting by creation date
    });
  }

  sortProjects(field: string): void {
    this.sortDirection = (field === this.sortField) ? -this.sortDirection : 1;
    this.sortField = field;

    this.sortedProjects = this.projects.slice().sort((a, b) => {
      const valueA = (typeof a[field] === 'string') ? a[field].toLowerCase() : a[field];
      const valueB = (typeof b[field] === 'string') ? b[field].toLowerCase() : b[field];

      if (valueA < valueB) {
        return -1 * this.sortDirection;
      } else if (valueA > valueB) {
        return 1 * this.sortDirection;
      } else {
        return 0;
      }
```

```
    });
  }
}
```

Users.controller.ts

```
const User = require('./user.model');
const jwt = require('jsonwebtoken');

exports.register = async (req, res) => {
  const { username, password } = req.body;
  try {
    const user = new User({ username, password });
    await user.save();
    res.status(201).json({ message: 'Registration successful' });
  } catch (error) {
    res.status(400).json({ message: 'Registration failed', error: error.message });
  }
};

exports.login = async (req, res) => {
  const { username, password } = req.body;
  try {
    const user = await User.findOne({ username });
    if (!user) throw new Error('User not found');

    const isPasswordValid = await user.comparePassword(password);
    if (!isPasswordValid) throw new Error('Invalid password');

    const token = jwt.sign({ userId: user._id, username: user.username }, 'your-secret-key', { expiresIn:
'1h' });
    res.json({ token });
  } catch (error) {
    res.status(401).json({ message: 'Authentication failed', error: error.message });
  }
};
```

Users.routes.js

```
const express = require('express');
const router = express.Router();
const usersController = require('./users.controller');

router.post('/register', usersController.register);
router.post('/login', usersController.login);
```

```
module.exports = router;
```

register.component.html

```
<div class="container">
  <h2>Registration</h2>
  <form (ngSubmit)="onSubmit()">
    <div class="form-group">
      <label for="username">Username:</label>
      <input type="text" id="username" name="username" [(ngModel)]="user.username" required>
    </div>
    <div class="form-group">
      <label for="password">Password:</label>
      <input type="password" id="password" name="password" [(ngModel)]="user.password" required>
    </div>
    <button type="submit">Register</button>
  </form>
</div>
```

Register.component.html

```
import { Component } from '@angular/core';
import { Router } from '@angular/router';
import { AuthService } from '../auth.service';

@Component({
  selector: 'app-register',
  templateUrl: './register.component.html',
  styleUrls: ['./register.component.css']
})
export class RegisterComponent {
  user = { username: '', password: '' };

  constructor(private authService: AuthService, private router: Router) {}

  onSubmit(): void {
    this.authService.register(this.user).subscribe(
      () => this.router.navigate(['/login']),
      error => console.error(error)
    );
  }
}
```

Auth.service.ts

```
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
```

```
import { Observable } from 'rxjs';

@Injectable({
  providedIn: 'root'
})
export class AuthService {
  private baseUrl = 'http://localhost:3000/api/users';

  constructor(private http: HttpClient) {}

  register(user: any): Observable<any> {
    return this.http.post(`${this.baseUrl}/register`, user);
  }

  login(user: any): Observable<any> {
    return this.http.post(`${this.baseUrl}/login`, user);
  }
}
```

Profile.component.ts

```
import { Component, OnInit } from '@angular/core';
import { AuthService } from '../auth.service';
import { ProfileService } from './profile.service';

@Component({
  selector: 'app-profile',
  templateUrl: './profile.component.html',
  styleUrls: ['./profile.component.css']
})
export class ProfileComponent implements OnInit {
  user: any = {};
  projects: any[] = [];
  selectedFile: File | null = null;
  editing: boolean = false;
  updatedUser: any = {};

  constructor(private authService: AuthService, private profileService: ProfileService) {}

  ngOnInit(): void {
    this.authService.getUserDetails().subscribe(user => {
      this.user = user;
      this.getProjects();
    });
  }
```

```
  getProjects(): void {
    this.profileService.getUserProjects().subscribe(projects => {
      this.projects = projects;
    });
  }

  onFileSelected(event: any): void {
    this.selectedFile = event.target.files[0];
  }

  updateProfile(): void {
    if (this.selectedFile) {
      this.profileService.uploadAvatar(this.selectedFile).subscribe(response => {
        this.user.avatar = response.avatarUrl;
        this.updateUserDetails();
      });
    } else {
      this.updateUserDetails();
    }
  }

  private updateUserDetails(): void {
    this.authService.updateUser(this.user._id, this.updatedUser).subscribe(() => {
      this.editing = false;
      this.updatedUser = {};
    });
  }
}
```

Profile.component.html

```
<div class="container">
  <h2>Профіль користувача</h2>

  <div *ngIf="!editing">
    <img [src]="user.avatar" alt="Аватарка" class="avatar">
    <p>Логін: {{ user.username }}</p>
    <p>Ім'я: {{ user.name }}</p>
    <p>Вік: {{ user.age }}</p>
    <button (click)="editing = true">Редагувати профіль</button>
  </div>

  <div *ngIf="editing">
    <input type="file" (change)="onFileSelected($event)">
    <img *ngIf="selectedFile" [src]="getObjectUrl(selectedFile)" alt="Вибрана аватарка" class="avatar-
preview">
```

```
    <br>
    <label for="name">Ім'я:</label>
    <input type="text" id="name" name="name" [(ngModel)]="updatedUser.name">
    <br>
    <label for="age">Вік:</label>
    <input type="number" id="age" name="age" [(ngModel)]="updatedUser.age">
    <br>
    <button (click)="updateProfile()">Зберегти зміни</button>
    <button (click)="editing = false">Відмінити</button>
  </div>

  <h3>Проекти користувача</h3>
  <table class="table table-striped">
    <thead>
      <tr>
        <th>Назва проекту</th>
        <th>Статус</th>
        <th>Голосів</th>
        <th>Дата створення</th>
      </tr>
    </thead>
    <tbody>
      <tr *ngFor="let project of projects">
        <td>{{ project.title }}</td>
        <td>{{ project.status }}</td>
        <td>{{ project.votes }}</td>
        <td>{{ project.createdAt | date:'medium' }}</td>
      </tr>
    </tbody>
  </table>
</div>
```

Додаток Г. Ілюстративна частина

Рисунок Г.1 – Назва роботи

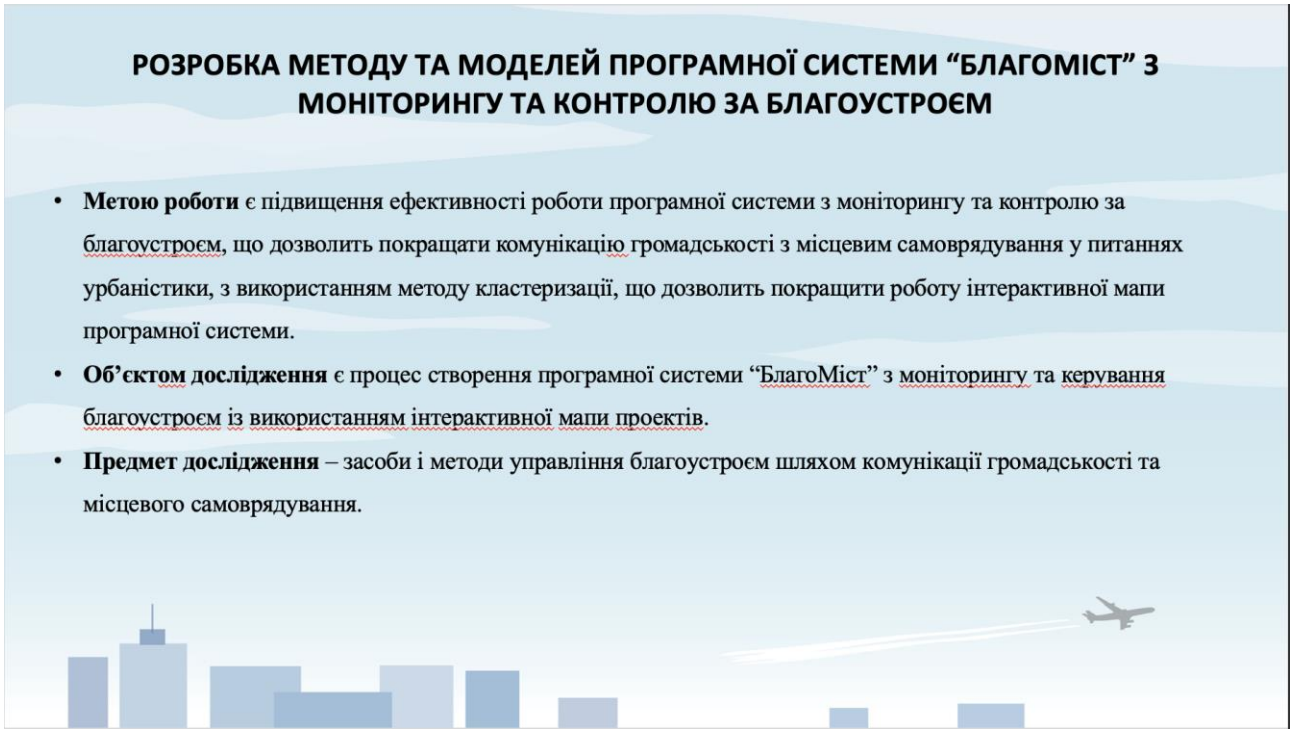

Рисунок Г.2 – Мета, об’єкт і предмет дослідження

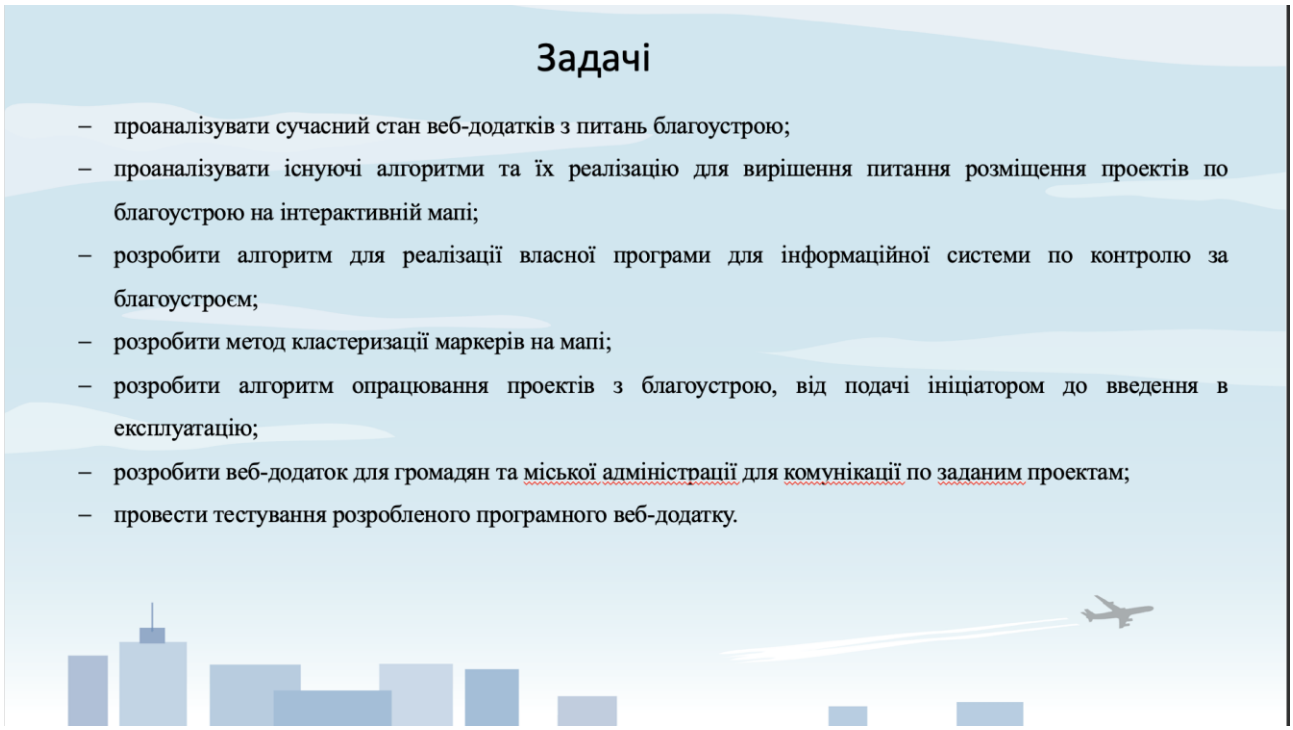

Рисунок Г.3 – Задачі

Порівняння аналогів

Критерії	Громадський бюджет	Електронні петиції	Власна реалізація
Спеціалізація на проектах благоустрою	+	-	+
Навність інтерактивкної мапи	-	-	+
Змога комунікувати з користувачами у чаті	-	-	+
Можливість пропонування змін до проекту	-	+	+
Зручність інтерфейсу	+	-	+
Швидкодія	+	+	+
Сума	3	2	6

Рисунок Г.4 – Порівняння аналогів

Модель програмної системи у вигляді діаграми прецедентів

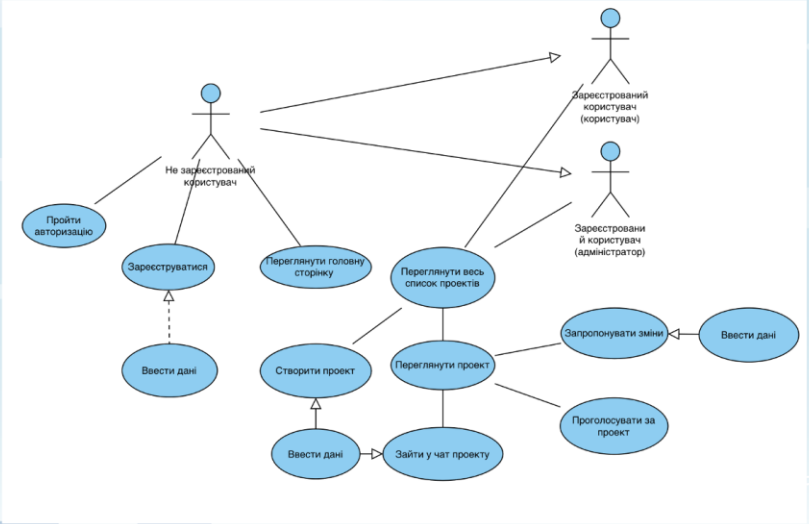

Рисунок Г.5 – Модель програмної системи

Рисунок Г.6 – Загальний алгоритм додатку

Рисунок Г.7 – Діаграма станів проекту

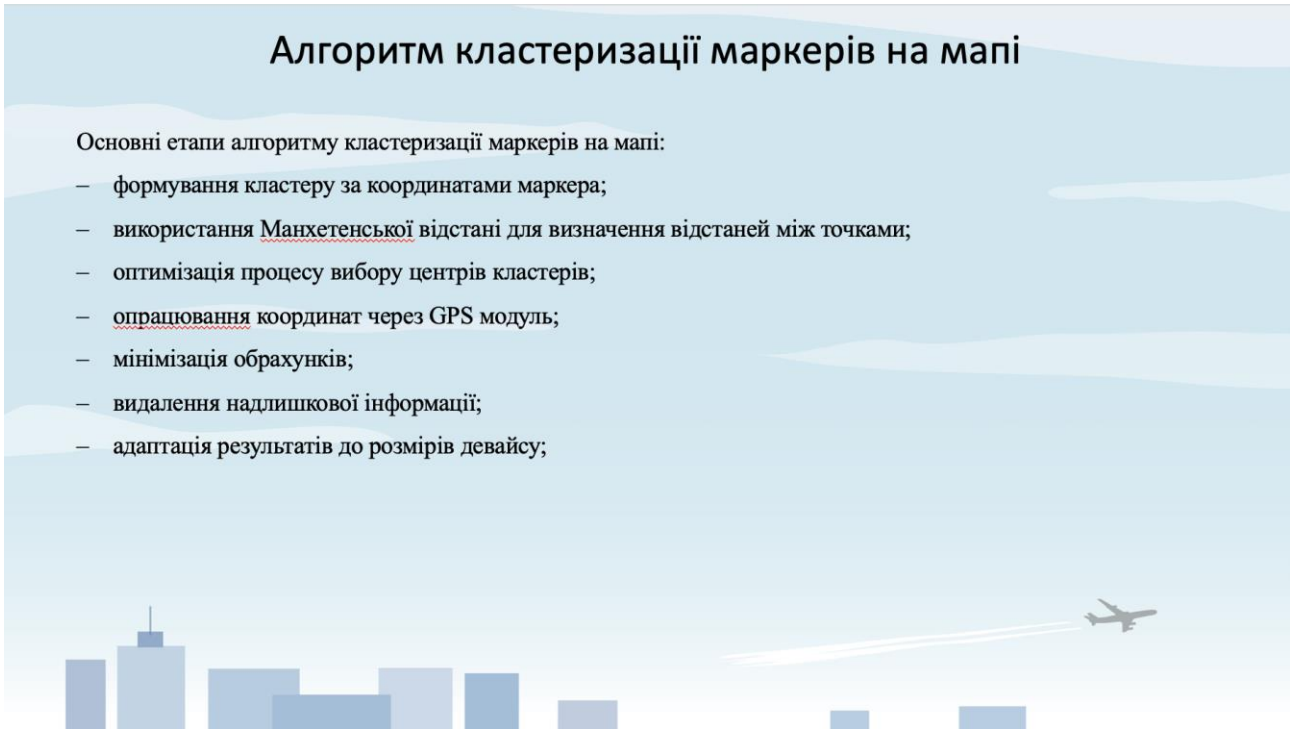

Рисунок Г.8 – Алгоритм кластеризації маркерів

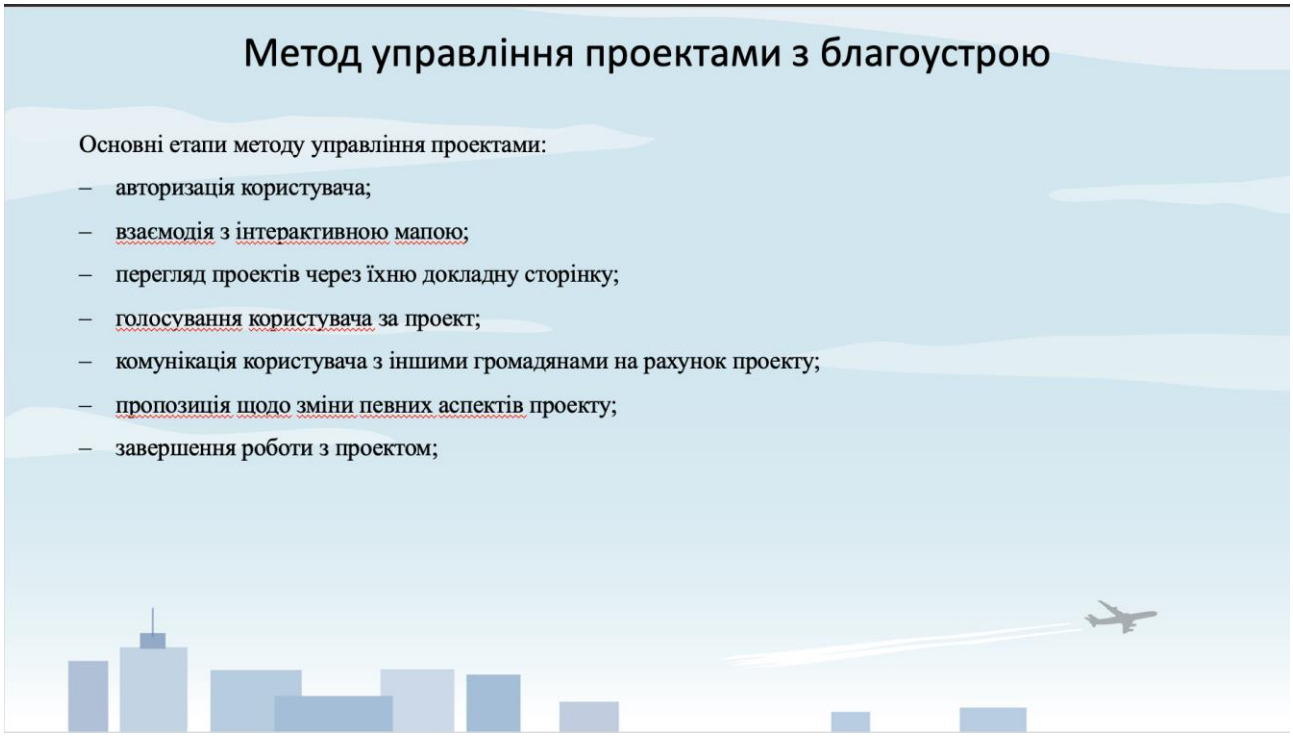

Рисунок Г.9 – Метод управління проектами з благоустрою

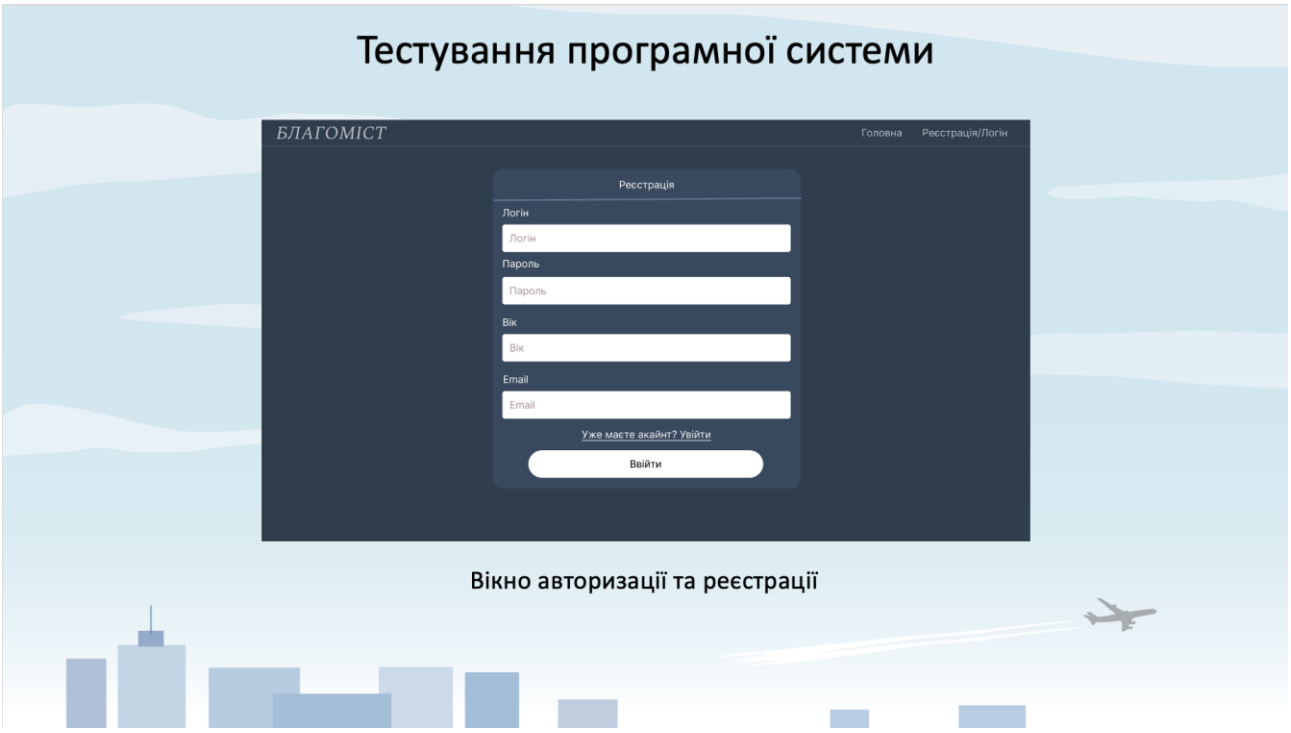

Рисунок Г.10 – Тестування авторизації та реєстрації

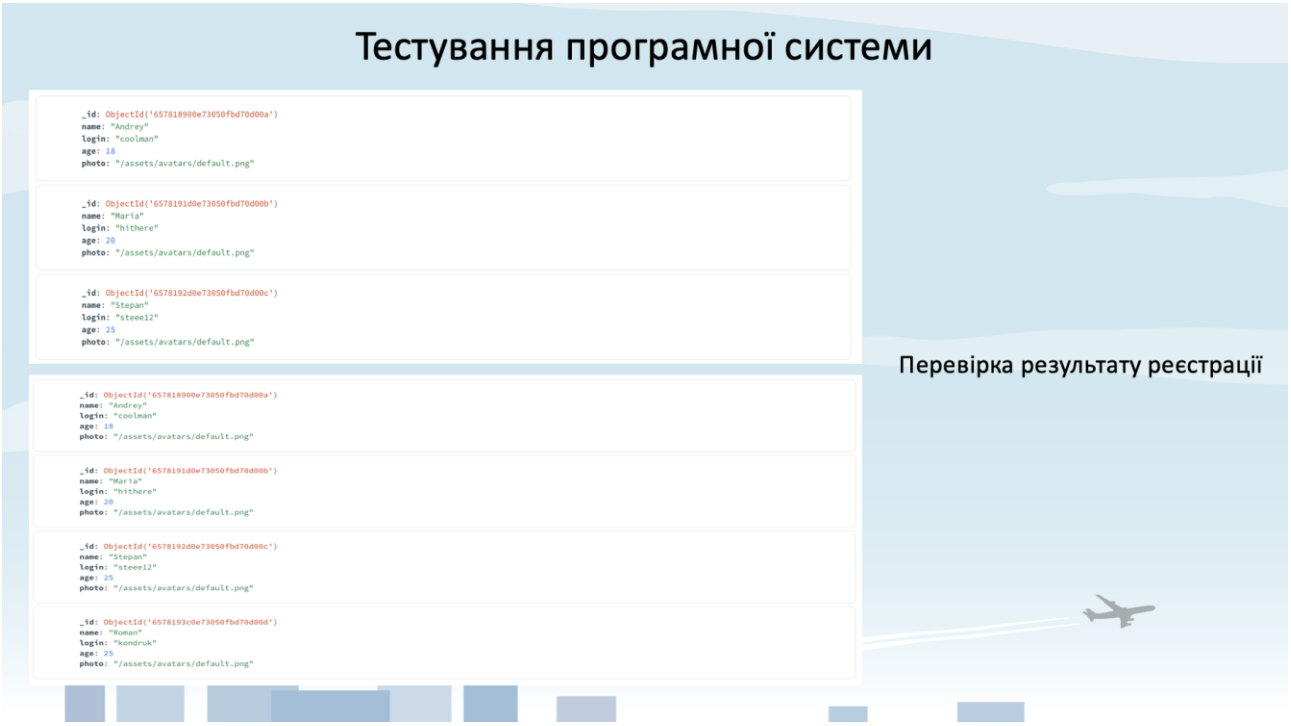

Рисунок Г.11 – Перевірка результату реєстрації

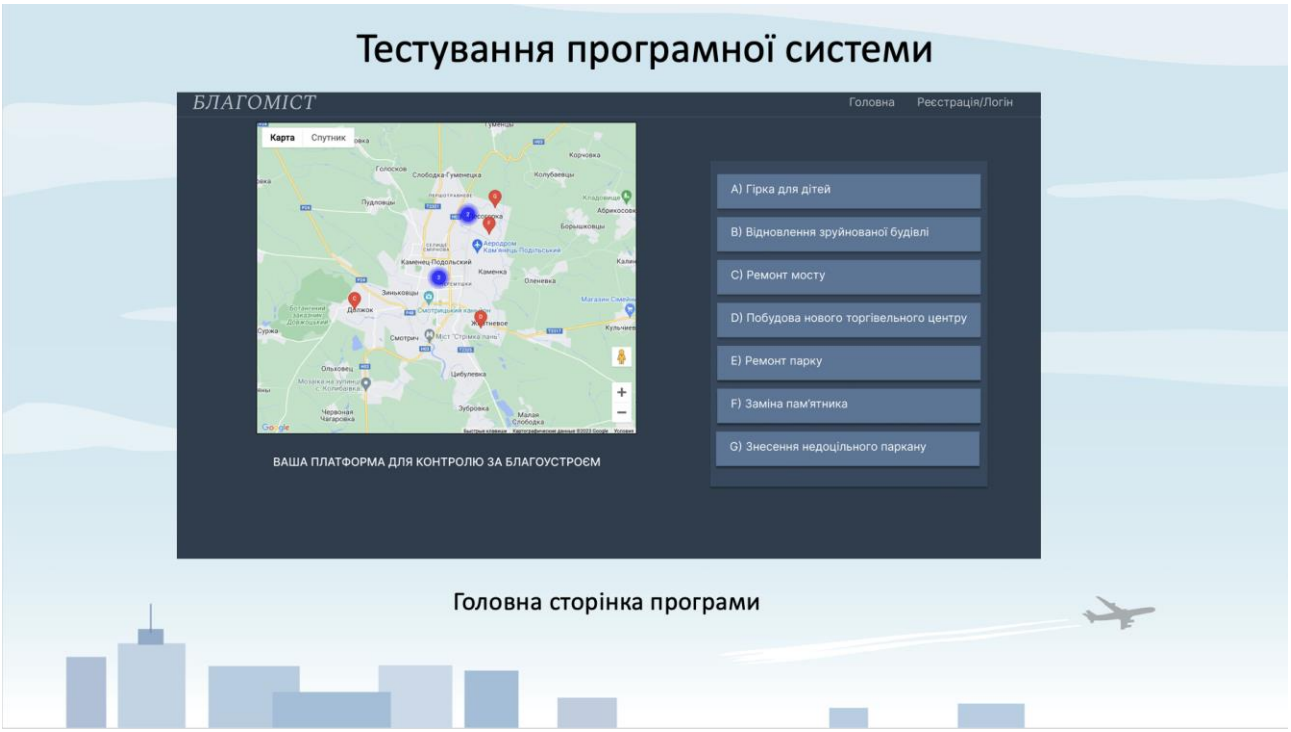

Рисунок Г.12 – Тестування головної сторінки

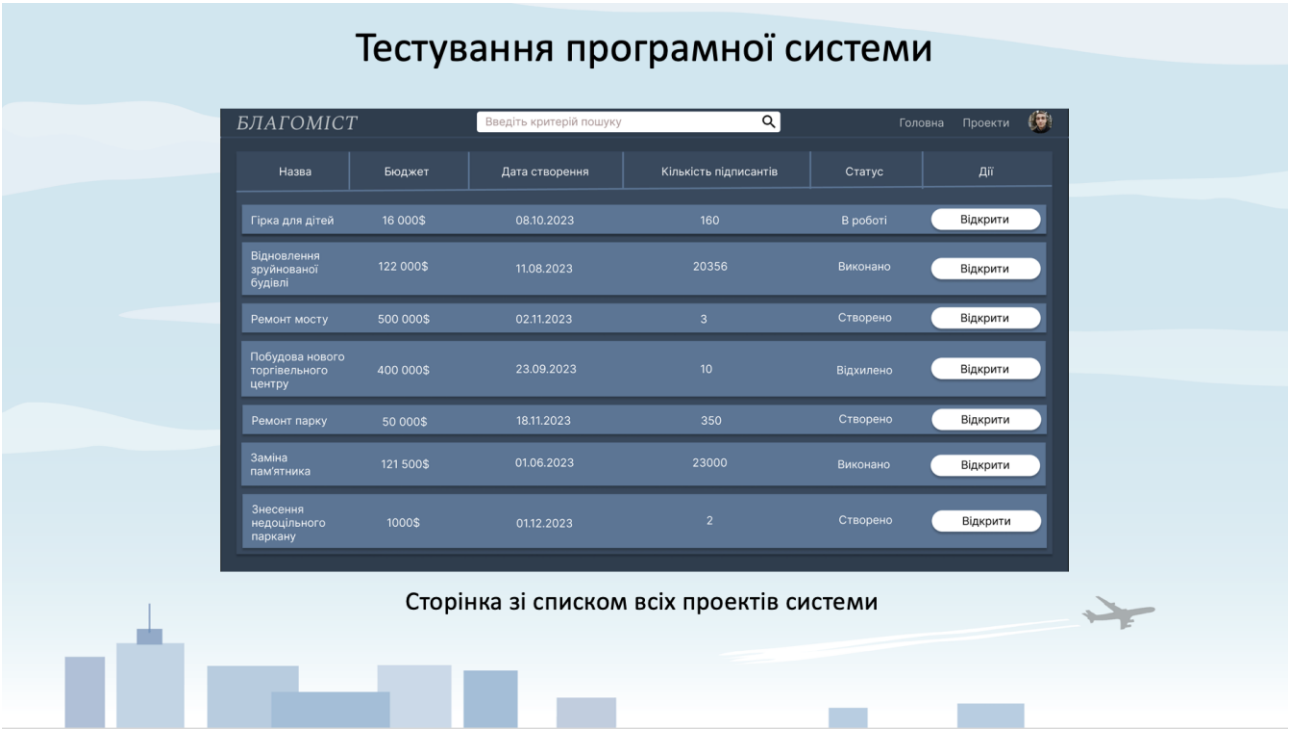

Рисунок Г.13 – Тестування списку проектів

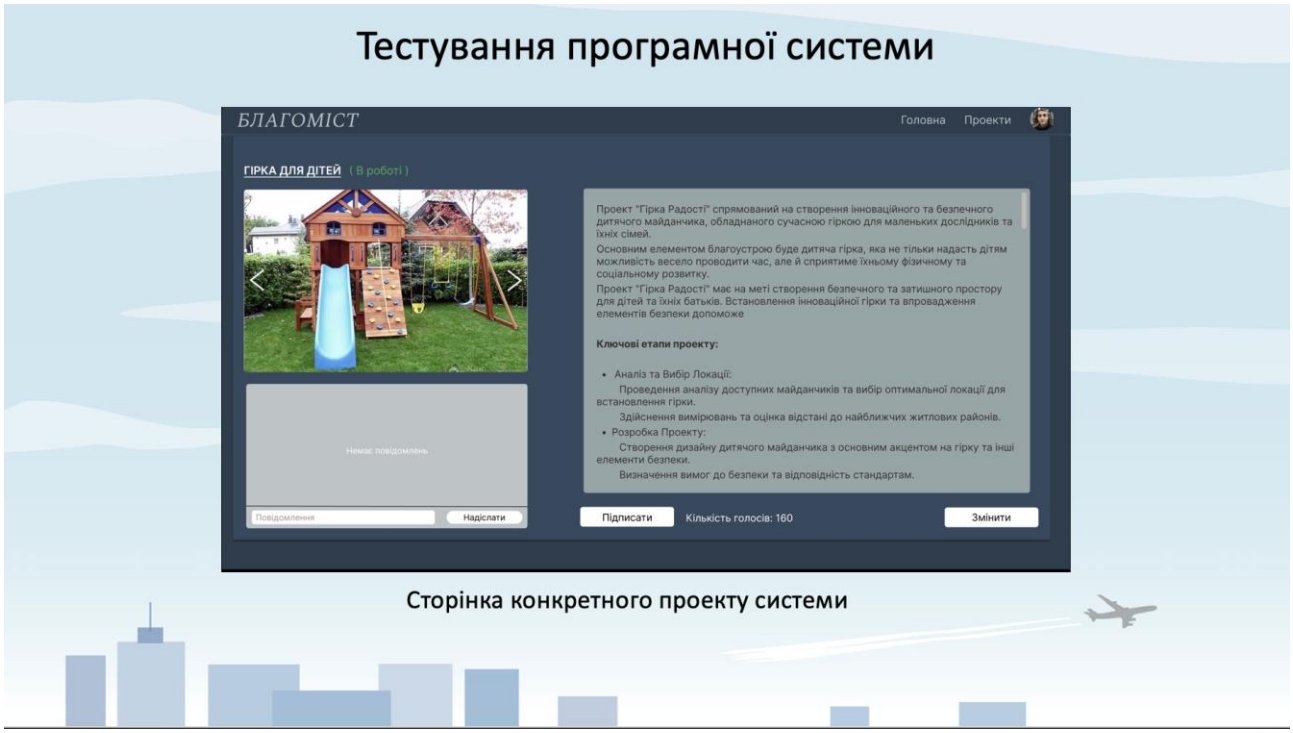

Рисунок Г.14 – Тестування сторінки проекту

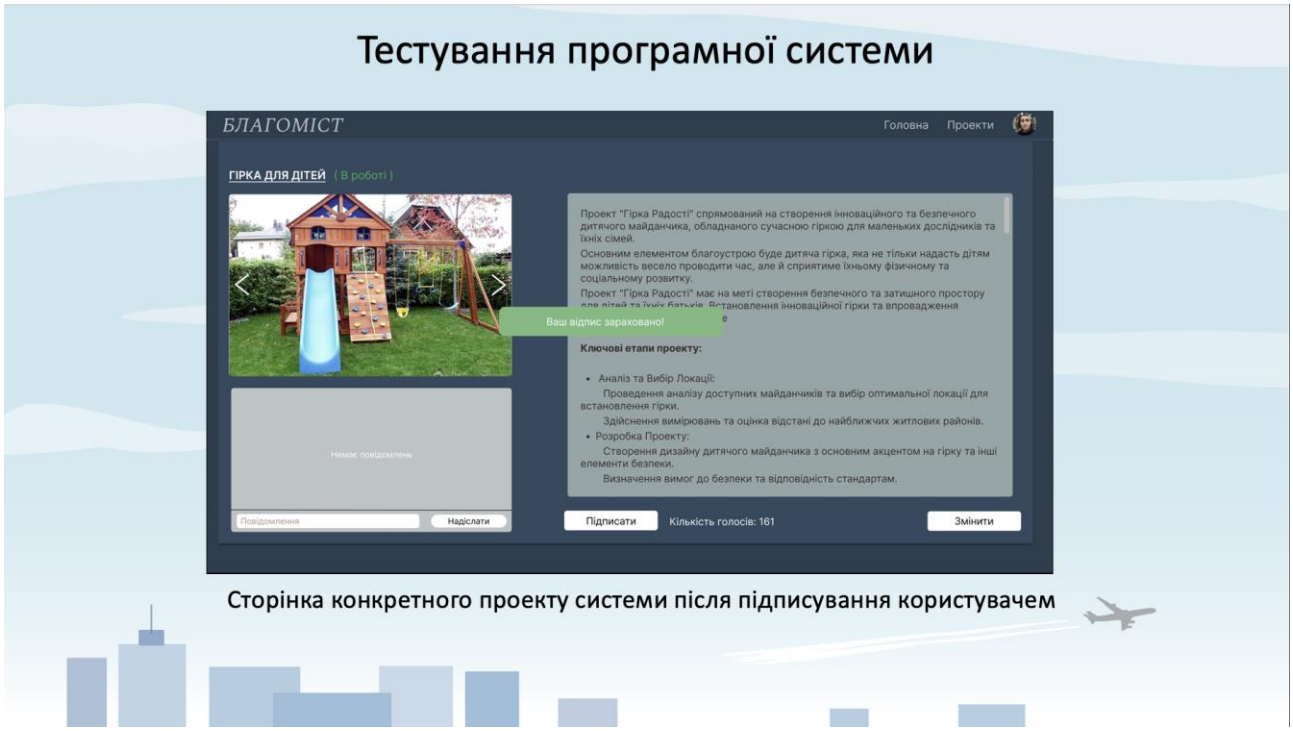

Рисунок Г.15 – Тестування сторінки проекту після підписання

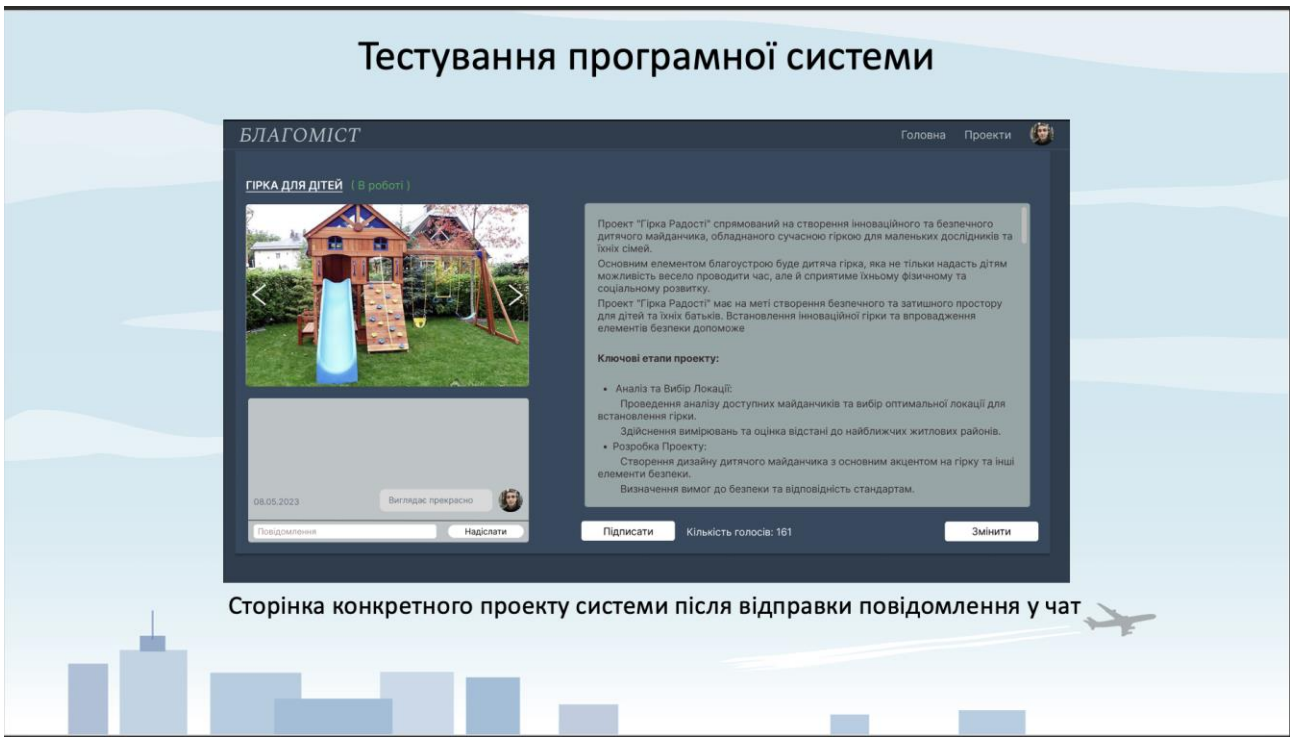

Рисунок Г.16 – Тестування сторінки проекту після відправки повідомлення

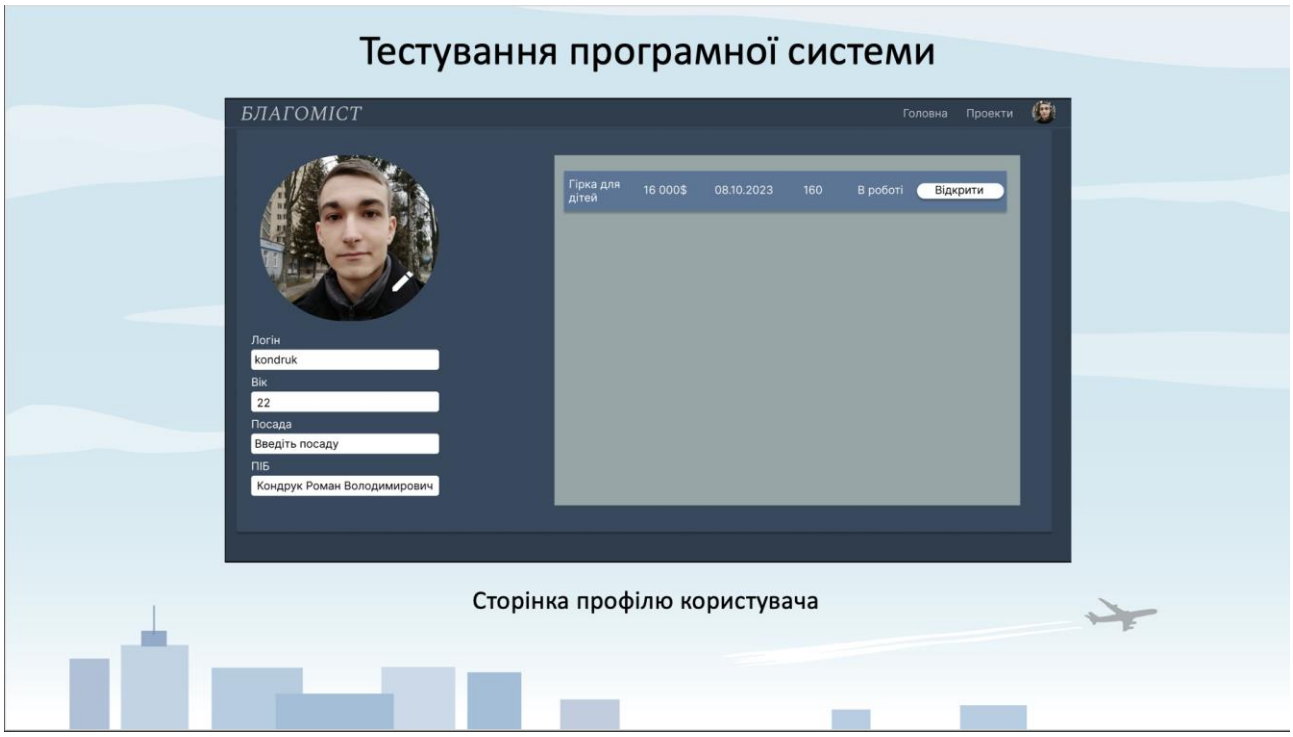

Рисунок Г.17 – Тестування сторінки профілю користувача

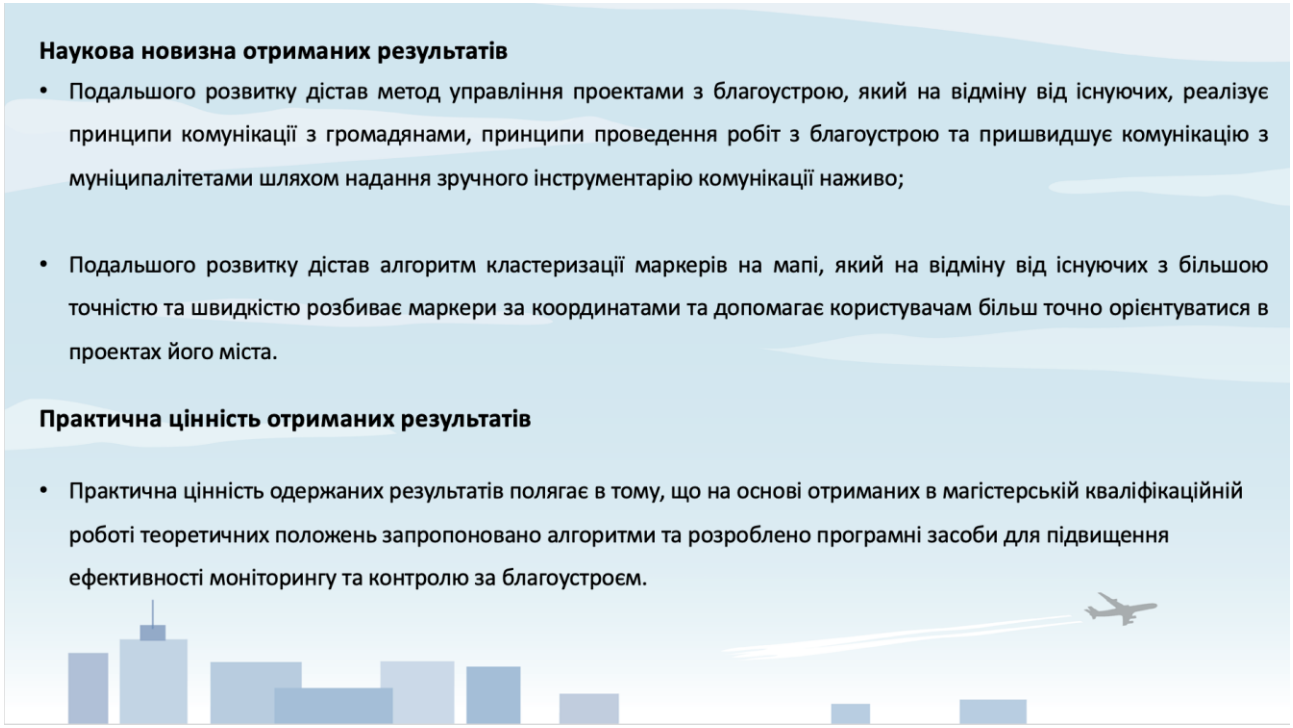

Рисунок Г.18 – Наукова новизна та практична цінність

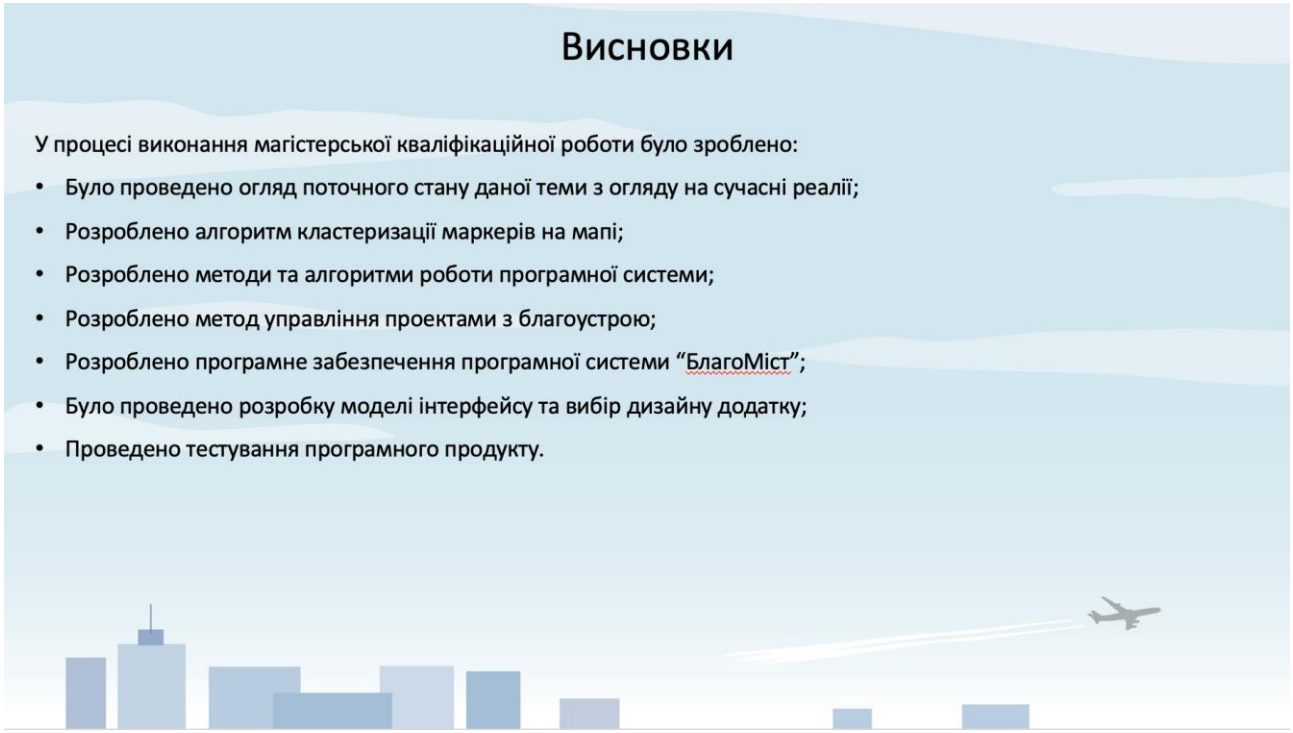

Рисунок Г.19 – Висновки