МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Західноукраїнський національний університет
Факультет комп'ютерних інформаційних технологій
Кафедра інформаційно- обчислювальних систем і управління

ЩЕГЛОВА Марія Ігорівна

Веб-базована система управління інфраструктурою територіальної громади/Web-based system for managing the infrastructure of a territorial community

спеціальність: 122 - Комп'ютерні науки
освітньо-професійна програма - Комп’ютерні науки

Кваліфікаційна робота

Виконала студентка групи КН-41
М. І. Щеглова

_____

Науковий керівник:
к.т.н., доцент, Х. В.
Ліп'яніна-Гончаренко

_____

Кваліфікаційну роботу
допущено до захисту:

"___" _____ 20___ р.

Завідувач кафедри
_____ **М. П. Комар**

ТЕРНОПІЛЬ - 2024

Факультет комп'ютерних інформаційних технологій
Кафедра інформаційно-обчислювальних систем і управління
Освітній ступінь «бакалавр»
спеціальність 122 – Комп'ютерні науки
освітньо-професійна програма – Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ М.П. Комар
«____» _____2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
ЩЕГЛОВІЙ Марії Ігорівні
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи: Веб-базована система управління інфраструктурою територіальної громади/Web-based system for managing the infrastructure of a territorial community
керівник роботи Ліп'яніна-Гончаренко Христина Володимирівна, к.т.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від 12 грудня 2023 р. № 753.

2. Строк подання студентом закінченої кваліфікаційної роботи 15 травня 2024 р.

3. Вихідні дані до кваліфікаційної роботи: завдання на кваліфікаційну роботу студента, наукові статті, технічна література.

4. Основні питання, які потрібно розробити:
- розглянути ключові аспекти функціонування ОТГ;
- визначення важливості систем управління та обліку в контексті територіальних громад, їх вплив на якість життя громадян;
- введення процесу автоматизації для покращення всіх аспектів функціонування ОТГ;
- порівняння автоматизованої громади із наявною системою управління;
- здійснити опис бази даних, яка використовувалася у роботі;
- здійснити опис методу класифікації факторів, які впливають на доцільність автоматизованої системи;
- розробити візуалізацію (веб-застосунок) для автоматизованої громади.

5. Перелік графічного матеріалу в роботі:
- Архітектура системи
- Діаграма прецедентів
- Таблиці глобальної моделі
- графіки з результатами експериментальних досліджень.

6. Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання 12 грудня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів кваліфікаційної роботи	Примітка
1	Затвердження теми кваліфікаційної роботи, ознайомлення з літературними джерелами та складання плану роботи.	до 01.01. 2024 р.	
2	Написання 1 розділу кваліфікаційної роботи	до 01.03. 2024 р.	
3	Написання 2 розділу кваліфікаційної роботи	до 01.04.2024 р.	
4	Написання 3 розділу кваліфікаційної роботи	до 01.05. 2024 р.	
5	Представлення попереднього варіанту кваліфікаційної роботи, перевірка та внесення змін керівником	до 15.05.2024 р.	
6	Опрацювання зауважень та представлення завершеного варіанту кваліфікаційної роботи. Підготовка супроводжуючих документів.	до 20.05.2024 р.	
7	Перевірка кваліфікаційної роботи на оригінальність тексту у системі «Unicheck».	до 10.06.2024 р.	
8	Оформлення кваліфікаційної роботи та отримання допуску до захисту	до 14.06.2024 р.	
9	Подання кваліфікаційної роботи до захисту на засіданні атестаційної комісії.	до 14.06. 2024 р.	

Студент_____ М.І. Щеглова
(підпис) (прізвище та ініціали)

Керівник кваліфікаційної роботи_____ Х.В. Ліп’яніна-Гончаренко
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Кваліфікаційна робота на тему «Веб-базована система управління інфраструктурою територіальної громади» на здобуття освітнього ступеня «бакалавр» зі спеціальності 122 «Комп’ютерні науки» освітньої програми «Комп’ютерні науки» написана обсягом в 43 сторінки і містить 22 ілюстрації, 1 таблицю та 31 використане джерело.

Метою роботи є розробка веб-базованої системи управління інфраструктурою територіальної громади, яка дозволяє ефективно керувати інфраструктурними об'єктами, забезпечуючи прозорість та оперативність процесів управління, а також підвищуючи рівень надання послуг населенню.

Методами розроблення обрано метод аналізу (для дослідження існуючих підходів до управління інфраструктурою), метод синтезу (для поєднання переваг існуючих методів), методи моделювання (для представлення та дослідження процесів управління інфраструктурою), метод порівняльного аналізу (для оцінювання адекватності моделі управління інфраструктурою).

Внаслідок виконання роботи обґрунтовано раціональний підхід до розроблення моделей управління інфраструктурою територіальної громади та розроблено програмний засіб, який дозволяє створювати і досліджувати моделі управління інфраструктурою.

Результати дослідження можуть бути використані в науково-дослідних установах і підрозділах підприємств, що займаються розробленням моделей управління інфраструктурою.

Ключові слова: ВЕБ-БАЗОВАНА СИСТЕМА, УПРАВЛІННЯ ІНФРАСТРУКТУРОЮ, ТЕРИТОРІАЛЬНА ГРОМАДА, АВТОМАТИЗАЦІЯ, ІНФОРМАЦІЙНІ СИСТЕМИ.

ANNOTATION

Qualification work on the topic «Web-based system for managing the infrastructure of a territorial community» for Bachelor's degree on speciality 122 «Computer Science» educational and professional program «Computer Science» is written on 43 pages and it contains 22 figures, 1 table, and 31 sources.

The purpose of the work is to develop a web-based system for managing the infrastructure of a territorial community, which allows effective management of infrastructure objects, ensuring transparency and operational efficiency of management processes, and improving the level of service delivery to the population.

Research methods include analysis (to study existing approaches to infrastructure management), synthesis (to combine the advantages of existing methods), modeling (to represent and study infrastructure management processes), and comparative analysis (to evaluate the adequacy of the infrastructure management model).

As a result of the work, a rational approach to the development of infrastructure management models for a territorial community was substantiated, and a software tool was developed that allows creating and researching infrastructure management models.

The research results can be used in research institutions and enterprise departments involved in the development of infrastructure management models.

Keywords: WEB-BASED SYSTEM, INFRASTRUCTURE MANAGEMENT, TERRITORIAL COMMUNITY, AUTOMATION, INFORMATION SYSTEMS.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ШІ – штучний інтелект

API – набір визначених правил, який дозволяє одному програмному продукту взаємодіяти з іншим для створення програмного забезпечення

NLP – обробки природної мови

БД – база даних

ОТГ – об’єднана територіальна громада

ТГ – територіальна громада

ІС - інформаційна система

ВСТУП

У контексті сучасних викликів, перед якими стоїть Україна, особливо у зв'язку з триваючими воєнними діями та необхідністю подальшої відбудови країни, важливість розробки та впровадження інноваційних автоматизованих систем управління в територіальних громадах набуває особливої актуальності. Це стосується не тільки покращення ефективності адміністративних процесів, але й забезпечення швидкого реагування на потреби громадян, оптимізацію розподілу ресурсів, підвищення якості життя населення та сприяння сталому розвитку регіонів.

В умовах, коли традиційні підходи до управління територіальними громадами виявляються недостатньо гнучкими та ефективними для вирішення виникаючих проблем, застосування сучасних технологічних рішень, таких як штучний інтелект, машинне навчання, великі дані та інтернет речей, відкриває нові можливості для реформування системи управління на користь громадян. Ці технології дозволяють не лише автоматизувати рутинні процеси, але й аналізувати великі обсяги даних для прийняття обґрунтованих рішень, прогнозування майбутніх тенденцій та адаптації до мінливих умов.

Актуальність цього дослідження полягає в тому, що воно спрямоване на визначення шляхів розробки та впровадження інтелектуальних систем управління в територіальних громадах України, які б допомогли відповісти на виклики, поставлені воєнним станом, та підготувати міцний фундамент для майбутньої відбудови країни. Дослідження зосереджується на аналізі існуючих рішень в цій сфері, виявленні їхніх сильних та слабких сторін, та розробці рекомендацій щодо створення ефективних, гнучких та масштабованих систем, здатних адаптуватися до змінних умов та вимог громад.

Важливо підкреслити, що успішна реалізація цих рішень вимагає не лише технологічних інновацій, але й глибокого розуміння соціально-економічних процесів, що відбуваються в громадах, а також активної участі громадян у процесах прийняття рішень. Таким чином, розробка та впровадження інтелектуальних

систем управління виступає не лише як технологічне завдання, але й як важливий крок до створення відкритих, прозорих та ефективних механізмів управління, що сприяють залученню громадян до активної участі в житті своїх громад і в цілому сприяють демократизації суспільства.

Метою даної дипломної роботи є розробка веб-базованої системи управління інфраструктурою територіальної громади, яка дозволяє ефективно керувати інфраструктурними об'єктами, забезпечуючи прозорість та оперативність процесів управління, а також підвищуючи рівень надання послуг населенню.

Для досягнення поставленої мети були визначені такі завдання:

– розглянути ключові аспекти функціонування ОТГ;

– визначення важливості систем управління та обліку в контексті територіальних громад, їх вплив на якість життя громадян;

– введення процесу автоматизації для покращення всіх аспектів функціонування ОТГ;

– порівняння автоматизованої громади із наявною системою управління;

– здійснити опис бази даних, яка використовувалася у роботі;

– здійснити опис методу класифікації факторів, які впливають на доцільність автоматизованої системи;

– розробити візуалізацію (веб-застосунок) для автоматизованої громади.

Об'єктом дослідження є процеси управління інфраструктурою територіальної громади, включаючи технічне обслуговування, планування розвитку, фінансовий облік та взаємодію з населенням.

Предметом дослідження є веб-базована система управління інфраструктурою, її архітектура, функціональні можливості, алгоритми обробки даних, а також методи інтеграції різних компонентів системи для забезпечення її ефективного функціонування..

Методи дослідження. Для досягнення мети дипломної роботи були використані такі методи дослідження: аналіз та синтез для визначення вимог до системи та розробки її архітектури; моделювання для проектування процесів управління інфраструктурою та розробки алгоритмів обробки даних;

програмування для реалізації веб-базованої системи з використанням сучасних технологій та фреймворків; тестування для перевірки функціональності системи та її відповідності вимогам користувачів; емпіричні методи для збору даних про потреби користувачів та оцінки ефективності розробленої системи в реальних умовах експлуатації.

Практичне значення даного дослідження полягає в розробці веб-базованої системи управління інфраструктурою територіальної громади, яка підвищує ефективність і прозорість процесів управління, забезпечує оперативний доступ до інформації про стан інфраструктурних об'єктів, покращує взаємодію з населенням та оптимізує використання ресурсів громади. Реалізована система сприятиме покращенню якості надання послуг громадянам, ефективному плануванню та обслуговуванню інфраструктурних об'єктів, що в свою чергу підтримує сталий розвиток та підвищує рівень життя в громаді.

Робота складається зі вступу, трьох глав, загальних висновків, списку використаної літератури, що містить 31 найменування, та чотирьох додатків. Зміст роботи висвітлено на 48 сторінках основного тексту і містить 22 рисунки.

Апробація результатів дослідження. Основні теоретичні положення роботи й практичні результати дослідження представлено у фаховому журналі «Вимірювальна та обчислювальна техніка в технологічних процесах» 2024, №2.

1 АНАЛІЗ АВТОМАТИЗОВАНОГО ОТГ

1.1 Коротка характеристика функціонування ОТГ

Територіальна громада [1] - жителі, об'єднані постійним проживанням у межах села, селища, міста, що є самостійними адміністративно-територіальними одиницями, або добровільне об'єднання жителів кількох сіл, селищ, міст, що мають єдиний адміністративний центр [1];

Основною діяльністю є забезпечення соціальних, інфраструктурних та економічних потреб мешканців та розвиток громади в цілому [1].

Система місцевого самоврядування включає:

– територіальну громаду;
– сільську, селищну, міську раду;
– сільського, селищного, міського голову;
– виконавчі органи сільської, селищної, міської ради [1].

Основні характеристики територіальної громади включають такі аспекти:

Географічні межі. Територіальна громада охоплює певну географічну область, яка може включати в себе міста, села, селища та інші населені пункти.

Населення. ТГ має власне населення, яке проживає на її території. Населення складається з громадян, які мешкають у населених пунктах, які є частиною ТГ.

Органи влади. Територіальна громада має свої органи влади, такі як обрана рада та голова. Ці органи влади призначені для управління та прийняття рішень щодо розвитку ТГ та надання послуг мешканцям.

Бюджет і фінанси. ТГ має свій бюджет, який формується з різних джерел, таких як податки, субвенції від вищих рівнів влади та інші джерела фінансування. Цей бюджет використовується для фінансування інфраструктури, освіти, охорони здоров'я та інших громадських послуг.

Розвиток і інфраструктура. ТГ бере на себе відповідальність за розвиток своєї території, включаючи будівництво та обслуговування доріг, водопостачання, каналізації, шкіл, лікарень і іншої інфраструктури.

Громадська участь. Територіальна громада залучає мешканців до участі у прийнятті рішень і плануванні розвитку. Громадська участь є важливою складовою прозорого та демократичного управління. Територіальні громади можуть мати різну структуру та компетенції в залежності від законів і правил, прийнятих в країні. Україна, наприклад, реформувала систему територіальних громад, створивши об'єднані територіальні громади (ОТГ), які об'єднують сусідні сільські ради та міські об'єднані територіальні громади. Ця реформа спрямована на покращення управління, забезпечення послуг та підвищення якості життя мешканців [2].

Основні характеристики територіальної громади визначаються географічними межами, власним населенням, органами влади, бюджетом і фінансами, розвитком і інфраструктурою, а також громадською участю. Ці елементи є ключовими для ефективного функціонування громади та забезпечення її потреб.

Україна, ініціюючи створення об'єднаних територіальних громад, виходить за метою покращення управління, забезпечення послуг та підвищення якості життя мешканців. Ця реформа відображає тенденцію до більш ефективного та демократичного локального управління, де важливу роль відіграє активна участь громадян у прийнятті рішень та розвитку їхнього регіону.

1.2 Структурно-організаційне управління громадою

Проектування автоматизованої системи управління громадою вимагає глибокого аналізу, дизайну та оптимізації для відповідності специфічним потребам користувачів та забезпечення високої продуктивності. Важливими кроками є визначення функціональності, структурування баз даних, розробка алгоритмів обробки даних та інтеграція компонентів для злагодженого функціонування системи. Цей процес також передбачає розробку моделі, що враховує оптимальність та гнучкість, а також встановлення ефективних зв'язків між різними модулями системи для покращення взаємодії та передачі даних.

Розглянемо організаційну структуру управління громадою, яка подана у вигляді ієрархічної схеми (рисунок 1.1).

1. Голова ОТГ

“Головна посадова особа територіальної громади села чи кількох сіл обирається на основі загального, рівного, прямого виборчого права шляхом таємного голосування терміном на 5 років. Голова очолює виконавчий комітет ради ОТГ, головує на скликаних ним сесіях ради, формує їхній порядок денний, затверджує рішення ради, ухвалені на засіданнях сесій” [2].

2. Депутати ради ОТГ

Це представники інтересів територіальної громади села/селища обираються також на 5 років. За цьогорічними змінами, громади з виборцями понад 10 тисяч обиратимуть депутатів не за мажоритарним принципом, як це було раніше, а лише за партійними списками.

3. Старости

Представляють інтереси жителів сіл свого округу у раді територіальної громади та її виконкомі, бере участь у підготовці бюджету громади в частині фінансового забезпечення округу; вказує, чому і на які саме потреби необхідні кошти на розвиток села/селища;

4. Відділи, управління та інші виконавчі органи сільської, селищної, міської, районної в місті ради [1].

Відділи, управління та інші виконавчі органи ради є підзвітними і підконтрольними раді, яка їх утворила, підпорядкованими її виконавчому комітету, сільському, селищному, міському голові, голові районної у місті ради [1].

Рисунок 1.1 - Ієрархічна схема управління ОТГ

Розглянемо потоки даних (рисунок 1.2) між головою ОТГ, старостами, радою ОТГ, виконавчим комітетом і відділами.

Існують такі інформаційні потоки в ОТГ:

1. Від старост до голови ОТГ

Староста може подавати звіти голові ОТГ щодо роботи у своїй території. Це може включати звіти про роботу комунальних служб, стан інфраструктури, питання безпеки тощо. Староста може передавати інформацію про нагальні потреби та проблеми на своїй території, які потребують уваги та ресурсів.

2. Від ради ОТГ до голови ОТГ

Рада ОТГ приймає рішення та резолюції, які можуть передаватися голові ОТГ для виконання.

Голова може отримувати інформацію від ради про бюджетні рішення та виділені кошти на конкретні проекти і програми.

3. Від відділів до голови ОТГ

Різні відділи ОТГ можуть подавати інформацію голові про свою діяльність, досягнення та поточні завдання.

Відділи можуть висувати запити до голови ОТГ щодо ресурсів, персоналу чи змін у політиці та програмах.

4. Від голови ОТГ до старост, ради ОТГ і відділів

Голова може приймати виконавчі рішення на основі рішень ради та потреб, які виборюється між головою та старостами.

Голова може впроваджувати плани та програми для розвитку ОТГ та виконання завдань, визначених радою.

5. Від виконавчого комітету до голови ОТГ

Виконавчий комітет може подавати голові звіти про поточну роботу та розглядати щоденні питання.

Виконавчий комітет може надавати голові пропозиції та рекомендації щодо політичних та адміністративних питань.

Якщо виконавчий комітет приймає важливі рішення, такі як фінансові зобов'язання чи участь у проектах, ця інформація повинна надходити до голови.

6. Від голови ОТГ до виконавчого комітету

Голова може направляти запити до виконавчого комітету для виконання певних завдань чи розгляду питань, які потребують уваги комітету.

Голова може повідомляти виконавчому комітету про скарги та питання, які надійшли від громадян, і просити комітет розглянути їх.

7. Від виконавчого комітету до відділів

Виконавчий комітет може надавати старостам, раді ОТГ і відділам звіти, документи та інформацію, яка стосується їхніх функцій і завдань.

Інформація про проекти, які виконавчий комітет планує реалізувати або вже реалізує, може бути передана старостам та іншим членам ОТГ для спільної роботи.

8. Від старост, ради ОТГ і відділів до виконавчого комітету

Старости, рада ОТГ і відділи можуть надсилати виконавчому комітету пропозиції та запити щодо різних питань.

Рисунок 1.2 – Потоки даних

Апарат ради Чорноострівської об'єднаної територіальної громади складається з керівних кадрів та фахівців, які забезпечують ефективне управління та функціонування громади. Він включає такі структурні підрозділи (рисунок 1.3)

1. Селищний голова

2. Секретар ради

3. Заступник селищного голови з питань діяльності виконавчих органів ради [3]

4. Керуючий справами виконавчого комітету

5. Старости сіл

– Староста сіл Рідкодуби, Лапківці

– Староста сіл Антонівка, Катеринівка

– Староста сіл Захарівці, Ляпинці, Крачки

– Староста сіл Везденьки, Малі Орлинці

– Староста сіл Миколаїв, Манилівка

– Староста села Грузевиця

– Староста села Осташки

– Староста сіл Педоси, Мартинівка, Бережанка

– Староста сіл Ставчинці, Польові Гринівці

6. Відділ бухгалтерського обліку та звітності

7. Начальник відділу бухгалтерського обліку та звітності, головний бухгалтер [3]

8. Заступник начальника відділу бухгалтерського обліку та звітності

9. Спеціалісти відділу бухгалтерського обліку та звітності

10. Фінансовий відділ

– Начальник фінансового відділу

– Спеціалісти фінансового відділу

11. Юридичний відділ

– Начальник юридичного відділу

– Спеціалісти юридичного відділу

12. Відділ освіти, культури, молоді та спорту

– Начальник відділу освіти, культури, молоді та спорту

– Спеціалісти з питань освіти, молоді та спорту

– Головний спеціаліст з питань культури, національностей, релігій та туризму

13. Відділ організаційної та інформаційної роботи

– Начальник відділу організаційної та інформаційної роботи

– Оператор електронно-обчислювальних та обчислювальних машин

– Діловоди відділу організаційної та інформаційної роботи

14. Інші відділи та служби

– Відділ земельних відносин

– Відділ праці та соціального захисту населення

– Сектор цивільного захисту населення та з питань житлово-комунального господарства

– Служба у справах дітей

– Центр надання адміністративних послуг [3]

Селищний голова відповідає за загальне керівництво та представництво селища, вирішує стратегічні питання розвитку та співпраці з іншими органами влади.

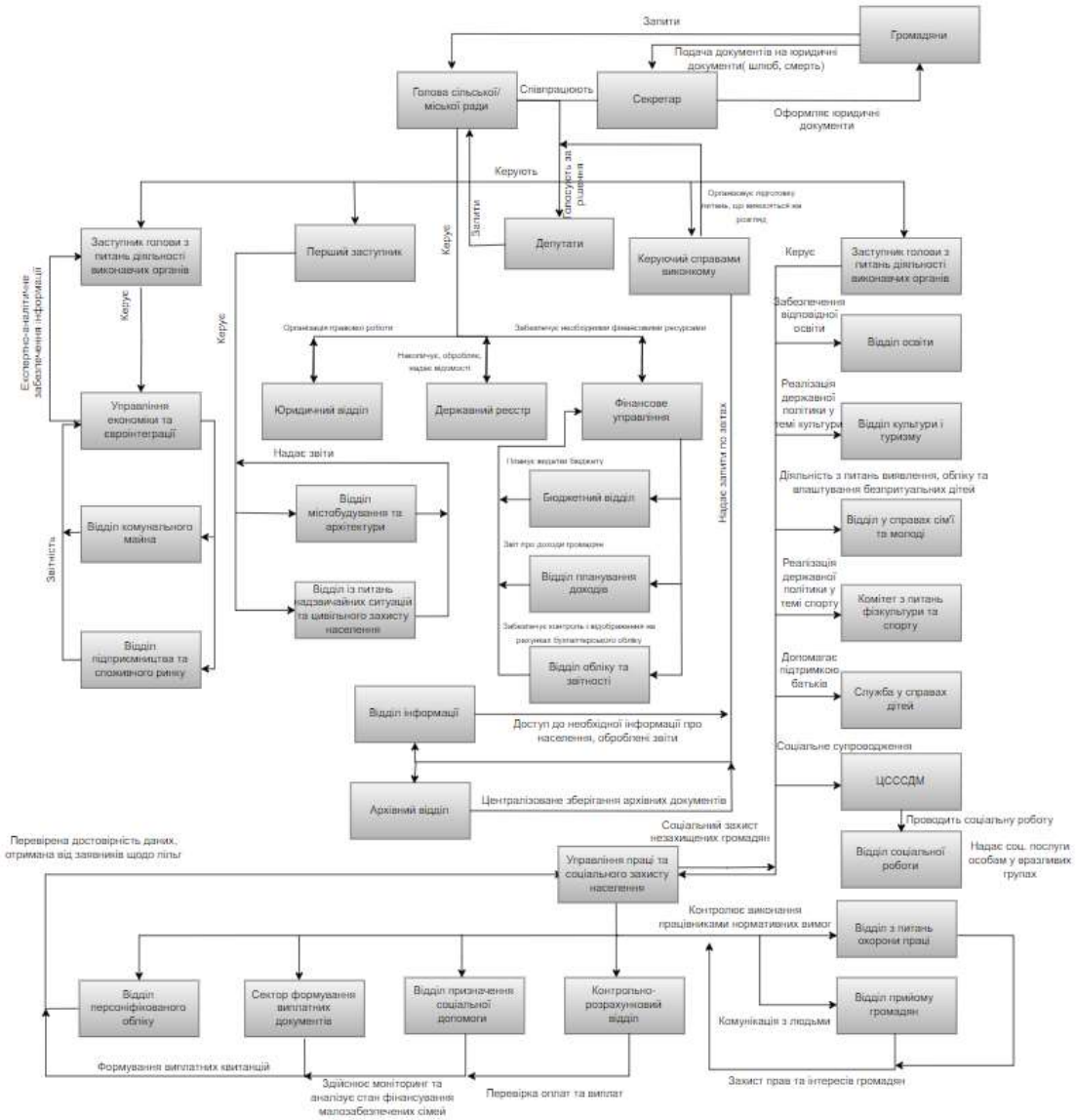


Рисунок 1.3 - Схема функціонування організаційних елементів ОТГ

Секретар ради забезпечує організаційно-процедурні аспекти роботи ради, веде протоколи засідань та забезпечує доступ до інформації.

Заступник селищного голови з питань діяльності виконавчих органів ради сприяє координації діяльності виконавчих органів ради та забезпечує виконання рішень.

Керуючий справами виконавчого комітету відповідає за організаційні аспекти роботи виконавчого комітету, управління документообігом та адміністративні питання.

Старости представляють інтереси окремих сіл перед радою, допомагають вирішувати проблеми місцевого значення та здійснюють контакт з мешканцями.

Відділ бухгалтерського обліку веде облік фінансово-господарської діяльності громади, складає фінансові звіти та забезпечує їхнє подання до відповідних органів.

Фінансовий відділ відповідає за збирання, обробку та розподіл фінансових ресурсів, а також контроль за фінансовою діяльністю громади.

Юридичний відділ забезпечує правовий супровід рішень та діяльності ради, надає консультації з юридичних питань.

Відділ освіти, культури, молоді та спорту організовує та контролює діяльність у сферах освіти, культури, молоді та спорту, сприяє розвитку цих галузей в громаді.

Відділ організаційної та інформаційної роботи відповідає за інформаційну підтримку діяльності ради, організацію роботи з громадськістю та внутрішній документообіг.

1.3 Аналіз існуючих рішень

В останні роки значно зросла увага до розробки автоматизованих систем управління в різних галузях, включаючи сільське господарство, розумні спільноти, управління енергетичними мережами, управління водними ресурсами, та оптимізацію виробництва в нафтогазовій промисловості. У сфері агротехнологій, [4] розробили автоматизовану систему управління для садівництва,

використовуючи такі технології як ADO.NET, Dapper, шаблон проектування MVC, Bootstrap, та Android Studio для розробки мобільних додатків, спрямованих на підвищення ефективності аграрних технологій. В контексті розумних спільнот, [5] запропонував модель динамічного аналізу великих даних на основі логістичної регресії для вирішення проблем, таких як високі витрати на будівництво та низький рівень інтелектуальності у розумних громадах. У галузі управління розумними мережами, [6] зосередилися на інтелектуальних системах контролю для управління мережами розумних грідів з метою зниження вартості та забезпечення безпеки ланцюгів постачання енергії. [7] описали систему WRESTORE, яка використовує передові обчислювальні підходи, такі як гідрологічна модель SWAT і алгоритми машинного навчання для проектування за інтересами стейкхолдерів у водозборах. В області нафти та газу, [8] створили розумну програмну систему для управління потоковими запевненнями, що сприяє оптимізації виробництва в реальному часі. У [9] розробив веб-базовану систему управління скаргами студентів для вищих навчальних закладів, використовуючи PHP, JavaScript, HTML, CSS та MySQL. У [10] застосував технології ASP і VB.NET для створення системи управління мережею рухів спільноти. У [11] обговорив розумну систему управління для контролю за потоковими запевненнями в нафтогазовій галузі. У [12] досліджували технології розумних медіа та застосування, включаючи обчислювальний інтелект та аналітику великих даних. Нарешті, [13] представили доступну, масштабовану еко-систему IoT для академічної спільноти, інтегруючи апаратні платформи з веб-серверами на хмарній основі та мобільними додатками.

У сучасних дослідженнях активно розглядаються різноманітні підходи до управління містами, селами та громадами, з акцентом на розвиток інтегрованих систем, що спрямовані на покращення адміністративної ефективності та залучення спільнот до участі в управлінні. У [14] розробили веб-базовану інформаційну систему для міських сіл, яка сприяє підвищенню ефективності та стабільності адміністрування населення, пропонуючи значний крок вперед до цифрового урядування та обслуговування громади. У [15] дослідили моделі управління та динаміку продукції в міських спільнотах, зокрема, зосередившись на Селі взуттєвої

промисловості в Cibaduyut, Bandung, Індонезія, підкреслюючи роль місцевих промислових підприємств у міському економічному розвитку. У [16] дослідив трансформацію "сіл всередині міст" під час урбанізації, розглядаючи реформи у сфері земельної нерухомості, громадської безпеки та структури громади для переходу від сільського до міського управління, відображаючи складності процесів урбанізації. У [17] обговорив будівництво міських і сільських громад в Китаї, прагнучи вирішити проблему нерівномірного розвитку та сприяти гармонійним соціальним структурам через рівність базових публічних послуг та координацію зусиль з будівництва міських і сільських громад. У [18] підкреслив роль приватного сектора та залучення громади у управлінні відходами у малих містах, як-от Амбарава, Центральна Ява, показуючи значення участі громади поряд з зусиллями уряду. У [18] зосередився на підвищенні участі громади у розвитку села через управління BUMDesa в Східній Яві, підкреслюючи вплив характеристик регіону на економіку та життя громади. У [19] розробили веб-базовану інформаційну систему адміністративного управління для адміністрації PKK у селі Mlatiharjo, прагнучи покращити ефективність та ефективність розподілу робочих програм, збору даних та процесів звітування. У [20] дослідили вплив збереження спадщини та туризму на стійкість життя громади в місці спадщини урбаністичного сільськогосподарського виноградарства Xuanhua, підкреслюючи роль туризму як альтернативного джерела доходу, але також висловлюючи занепокоєння щодо стійкості місцевого сільського життя. У [21] запропонували соціальний капітал як важливий інструмент для моніторингу впровадження просторового планування на Балі, зосереджуючись на викликах управління землею в секторі туризму шляхом залучення корінних сіл до контролю та інституційної реформи. У [22] обговорили переваги впровадження розумної громадської системи на основі дизайну Інтернету речей у житлових районах, що призводить до автоматичного контролю обладнання, інтеграції електронного обладнання та покращення житлового середовища та безпеки та комфорту для мешканців села.

Веб-портали та мобільні додатки для спілкування з мешканцями [23] надають можливість подання заявок, вираження думок та голосування за важливі питання

безпосередньо через інтернет, забезпечуючи максимальну зручність та доступність. Ці інструменти дозволяють мешканцям стежити за процесами управління та брати активну участь у вирішенні проблем своєї громади.

Інтегровані платформи управління територіальними громадами [24] об'єднують різні функції, включаючи спілкування з мешканцями, подання заяв, організацію голосувань та моніторинг реалізації інфраструктурних проектів. Ці платформи забезпечують централізований доступ до інформації та дозволяють ефективно координувати дії між владою та громадою.

Запропоноване дослідження вирізняється серед інших (вище проаналізованих) завдяки комплексному підходу до інтеграції автоматизованих систем управління в територіальних громадах, який охоплює не лише технічні аспекти розробки систем, але й глибоке залучення громадськості та адаптацію до локальних особливостей. На відміну від інших досліджень, які зосереджуються на конкретних аспектах, таких як управління відходами, розумні гріди чи агротехнології, наш проект пропонує голістичну модель, що враховує соціальні, економічні та екологічні потреби громади. У даній розробці акцентується на створенні платформ, що дозволяють мешканцям не тільки отримувати доступ до інформації та послуг, але й активно брати участь в процесах прийняття рішень, сприяючи підвищенню прозорості та відповідальності місцевої влади.

Крім того, дана розробка впроваджує передові технологічні рішення, такі як штучний інтелект та машинне навчання, для аналізу великих даних, що дозволяє прогнозувати тенденції розвитку та потреби громади, оптимізувати ресурси та покращувати якість життя мешканців. Відмінною рисою цього проекту є також розробка адаптивних алгоритмів, здатних самонавчання та автоматичної адаптації до змінних умов, що забезпечує високу ефективність та гнучкість системи управління. Такий підхід дозволяє не тільки вирішувати поточні завдання, але й прогнозувати майбутні виклики, випереджаючи потреби громади та забезпечуючи сталий розвиток.

1.4 Постановка задачі дослідження

Актуальність дослідження полягає у тому, що територіальні громади є важливими соціально-економічними одиницями, які відіграють ключову роль у забезпеченні потреб їхніх мешканців. Розуміння структури та мети існування територіальних громад є важливим для розробки та впровадження ефективних стратегій управління, спрямованих на поліпшення якості життя громадян.

Ключові аспекти функціонування визначаються їхньою структурою та метою існування. Територіальна громада, об'єднана єдністю проживання, виконує важливу роль у забезпеченні соціальних, інфраструктурних та економічних потреб її мешканців.

Система місцевого самоврядування включає різні складові, такі як територіальна громада, рада, голова та виконавчі органи, і всі вони спільно працюють для забезпечення життєвих умов та розвитку громади. Головні аспекти включають географічні межі, населення, органи влади, бюджет і фінанси, розвиток і інфраструктура та громадську участь.

Однак існуюча система управління може мати свої недоліки, такі як недостатня швидкість та продуктивність. Оптимізація та впровадження автоматизованої системи може вирішити ці проблеми, забезпечуючи більш ефективне та швидше управління, що покращить якість надання послуг та рівень задоволення мешканців.

Однією з ключових проблем, які виникають це необхідність ефективного управління ресурсами та інфраструктурою для забезпечення соціального, інфраструктурного та економічного розвитку. Це означає, що необхідно досліджувати та аналізувати структуру функціонування територіальних громад, їхні потреби та можливості для вдосконалення управління.

Актуальність дослідження полягає в тому, що у сучасному світі більшість бізнес-процесів в організаціях пов'язані з інформаційними технологіями, а також використанням різноманітних технічних пристроїв та обробкою даних за допомогою новітніх технологій. Впровадження та підтримка таких технічних

засобів та процесів вимагають певного рівня кваліфікації співробітників, а також може потребувати звернення до кваліфікованих спеціалістів.

Отже, метою даної дипломної роботи є розробка веб-базованої системи управління інфраструктурою територіальної громади, яка дозволяє ефективно керувати інфраструктурними об'єктами, забезпечуючи прозорість та оперативність процесів управління, а також підвищуючи рівень надання послуг населенню.

Для досягнення поставленої мети були визначені такі завдання:

1. розглянути ключові аспекти функціонування ОТГ;
2. визначення важливості систем управління та обліку в контексті територіальних громад, їх вплив на якість життя громадян;
3. введення процесу автоматизації для покращення всіх аспекти функціонування ОТГ;
4. порівняння автоматизованої громади із наявною системою управління;
5. здійснити опис бази даних, яка використовувалася у роботі;
6. здійснити опис методу класифікації факторів, які впливають на доцільність автоматизованої системи;
7. розробити візуалізацію (веб-застосунок) для автоматизованої громади.

2 АЛГОРИТМІЧНЕ ТА ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ СИСТЕМИ

2.1 Моделювання процесів системи управління ОТГ

Моделювання автоматизованої системи є ключовим етапом в розробці та впровадженні ефективних технологічних рішень. Цей процес включає в себе аналіз, проектування та оптимізацію системи з метою покращення продуктивності та забезпечення відповідності потребам користувачів. Важливим етапом є визначення функціональності системи, розробка структури бази даних, визначення алгоритмів обробки даних та інтеграція різних компонентів для забезпечення плавного функціонування.

Моделювання також включає в себе визначення взаємозв'язків між різними модулями системи та ефективних методів обробки та передачі даних. Враховуючи потреби користувачів та особливості конкретної області застосування, модель створюється з урахуванням оптимальності та гнучкості.

У результаті моделювання автоматизованої системи досягається високий рівень функціональності, ефективності та надійності, що дозволяє забезпечити успішне впровадження системи в реальне виробниче чи бізнесове середовище.

Діяльність складається з таких трьох процесів :

– Процес обслуговування населення
– Процес фінансового обліку
– Процес управління інфраструктурними об’єктами

1. Процес обслуговування населення

Цей процес включає в себе всі аспекти обслуговування населення, такі як прийом та обробка звернень від громадян, вирішення їхніх питань і проблем. Він може включати такі етапи, як реєстрація звернень, взаємодія з клієнтами, вирішення та виконання запитань чи послуг.

2. Процес фінансового обліку

Цей процес включає в себе всі фінансові операції та облік бюджетних коштів. Це може включати в себе створення та виконання бюджету, контроль фінансових операцій, аудит та звітність.

3. Процес управління інфраструктурними об’єктами

Цей процес включає в себе управління інфраструктурними об'єктами, такими як дороги, водопостачання, освітні та медичні заклади. Він може включати в себе технічне обслуговування, планування розвитку інфраструктури, контроль за її ефективністю.

Кожен з цих процесів взаємодіє між собою, створюючи цілісну систему для ефективного функціонування організації. Результативне управління обслуговуванням населення, фінансами та інфраструктурою сприяє створенню комфортних умов для громадян та підтримує сталість та розвиток місцевої спільноти.

Щоб наочно продемонструвати функціональне моделювання автоматизованої територіальної громади системи була розроблена IDEF0 діаграма, яка демонструє побудову ієрархічної системи діаграм. Представлення даних діаграм наведені на рисунках (див. Рисунок А.1-А.4, Додаток А). Функціональне моделювання автоматизованої громади (в IDEF0)

Одним із важливих висновків є той факт, що правильно спроектовані потоки дозволяють оптимізувати роботу системи та підвищити загальну продуктивність. Чітке визначення потоків даних від введення до обробки та виведення дозволяє уникнути зайвої затримки чи втрати інформації.

Моделювання також дозволило виявити можливі точки оптимізації в потоках, щоб забезпечити більш ефективну роботу системи в цілому. Наприклад, шляхи оптимізації можуть включати в себе покращення алгоритмів обробки даних, зменшення часу відповіді та підвищення надійності обробки інформації.

Проведене моделювання автоматизованої системи виявило ключові аспекти та оптимальні потоки інформації в її структурі. Потоки в системі ретельно прокладені, щоб забезпечити ефективний обмін даними між різними компонентами та модулями.

2.2 Алгоритмічне забезпечення процесів управління об’єднаною територіальною громадою

Щодо характеристики інтелектуальної системи, то можна організувати зовнішнє керування, але для неї характерною є самокерованість [25]. Система має певну мету і прагне так планувати свої дії, щоб досягати цієї мети. Як вхідні стимули системи можна розглядати поточну ситуацію, що сприймається і аналізується системою [25]. Результатом реакції системи стає зміна зовнішньої ситуації, і поведінка системи коригується в залежності від того, бажаною чи небажаною є ця зміна [25].

Побудуємо схеми алгоритмів роботи системи територіальної громади, це дасть змогу зрозуміти основні етапи, які виконують функції: голосування за проект та комунікація з населенням через сформовані звернення.

Загальний алгоритм процесу обслуговування має таку логіку (див. Рисунок А.5, Додаток А):

1. Джерелом інформації є мешканці, які в свою чергу подають скарги, заяви та звернення.
2. Наступним процесом є реєстрація заяви та запис зареєстрованих заяв у базу даних.
3. Оцінка потреб аналізує чи необхідно розглядати звернення та які дії приймати. Якщо заява не потребує розгляду, то подаємо звітність мешканцям, якщо розгляду підпадає, то маємо процес прийняття рішення.
4. Якщо необхідне відшкодування, то відбувається відшкодування коштів.
5. Завершальний етап включає формування звітностей, які передаються керівним органам та мешканцям.

Загальний алгоритм процесу управління інфраструктурними об’єктами має таку логіку (див. Рисунок А.6, Додаток А):

1. Із зовнішніх джерел отримуємо інформацію про інфраструктуру.
2. Процес обслуговування та перевірка наявної інфраструктури на виявлення пошкоджень та наявності змін.

3. Перевірка на потребу в ремонті, якщо так, розподіляємо кошти, які беремо із файлу про фінансові дані, та вносимо зміни в інфраструктуру. Якщо ні, то переходимо до наявності залишку коштів на нові потреби.

4. Розробка плану нових проектів за наявності залишку коштів та наявності наказу про потребу будівництва.

5. Процес будівництва та створення бази даних та файлу з новими інфраструктурними об’єктами.

6. Формуємо електронні, паперові звіти та вносимо зміни в наявну базу даних та файл з коштами.

У сучасному світі інтелектуальні технології стають все більш важливими у різних сферах життя, включаючи веб-розробку. Тому при розробці сайту для об’єднаної Чорноострівської територіальної громади було розроблено інтелектуальний телеграм-бот для комунікації мешканців з владою.

Завдяки штучному інтелекту телеграм бот розуміє запити користувачів та надає їм персоналізовані відповіді та рекомендації. Це допомагає покращити задоволеність користувачів та підвищити їхню лояльність. Також ШІ автоматизує багато рутинних завдань, таких як модерація контенту, обробка звернень та надання технічної підтримки. Це може звільнити час та ресурси персоналу ОТГ для більш важливих завдань. Бот збирає та аналізує дані про поведінку користувачів в телеграмі. Ці дані можуть бути використані для покращення роботи сайту, а також для кращого розуміння потреб жителів ОТГ.

Основне завдання розробленого інтелектуального боту спрямоване на підтримку мешканців, тобто допомагає з питаннями щодо користування сайтом, подачею всіх видів заяв, питань про зв’язок з владою, інформацію про заходи та новини ОТГ.

Для розробки бота з штучним інтелектом були зібрані та підготовлені дані, тобто приклади запитів користувачів, розмови та сценарії, які відповідають формату моделі машинного навчання. Вибором машинного навчання для розробки телеграм-бота було розуміння природної мови NLP.

NLP (Natural Language Processing) [27], або обробка природної мови – це окремий технологічний напрямок в частині штучного інтелекту, який присвячений тому, як машина розуміє, аналізує та відтворює природну мову, якою людина звертається до неї [27]. В науковій сфері існує так званий тест Тюрінга, який говорить про те, що якщо людина, при спілкуванні з комп'ютером, не може визначити, що з нею спілкується машина, машина є штучним інтелектом [27].

Наступним кроком було проведення навчання моделі, яке включало налаштування параметрів моделі, розмір пакета та число епох. При закінченні тестування набору даних, було оцінено модель, таким чином переконались, що модель працює добре.

Отримали готовий чат-бот, який було розгорнуто на платформі Телеграм, та підключено до розробленого веб-сайту.

2.3 Потоки даних у системі

Дані є основою будь-якої програми. Незалежно від того, чи йдеться про програму для мобільних пристроїв, веб-сайт чи програму для комп'ютера, дані створюються та зберігаються в базі даних, щоб отримати доступ до них пізніше. Можливість створювати, читати, оновлювати та видаляти дані є фундаментальною для багатьох програмних програм.

Круд операції (Create, Read, Update, Delete) є одним з основних наборів операцій, які використовуються для керування даними у базах даних та веб-додатках (рисунок 2.7).

Почнемо із створення, це можливість додавати нові дані є основною функцією системи керування базами даних. Завдяки цій можливості можна збирати будь-яку інформацію. Операція Create зазвичай є першою операцією, яка використовується під час створення нової таблиці бази даних або додавання нових даних до існуючої. Складається з двох частин: 1) Додавання нового запису 2) Встановлення значень для кожного поля в цьому записі.

Читання, отримання існуючих даних - операція Read дозволяє отримати наявні дані з таблиці бази даних або подання. Зазвичай це друга операція після створення нової таблиці бази даних або додавання нових записів.

Оновити, змінити існуючі дані – операція, яка може знадобитись після зчитування даних зі сховища перед збереженням у системі постійного зберігання. Функція оновлення дозволяє користувачам змінювати існуючий запис без видалення або створення повністю нового запису для кожної зміни (наприклад, під час редагування тексту в документі).

Видалити - дозволяє користувачам видалити один або кілька записів зі своєї бази даних. Функція використовується, щоб звільнити місце для нових записів або якщо користувач завершив роботу з елементом і йому більше не потрібен. Важливо зауважити, що коли видаляється запис, усі пов’язані дані також буде видалено зі сховища.

Рисунок 2.7 – CRUD операції

Так як CRUD операції є основою створення програм, то їх було застосовано при розробці WEB-застосунку. Візьмемо за приклад два аспект звернень, у класі було застосовано CRUD операції, такі як GET, POST, PUT, DELETE, кожна із яких має свої дані для передачі, а отже працює для додавання, зміни, отримання та видалення даних про звернення громадян (рисунок 2.8).

Пропонована веб-орієнтована архітектура - підхід до розробки програмних додатків, де клієнтська та серверна частини програми взаємодіють через мережу, через протокол HTTP (Hypertext Transfer Protocol), що використовується в Інтернеті. Цей підхід передбачає, що клієнтська частина програми є веб-браузером, який може звертатися до веб-сервера для отримання даних та відображення їх користувачеві.

Рисунок 2.8 – CRUD операції в розробленій системі

Серверною частиною виступаю платформа ASP.Net WEB API, а клієнтською React.

“Для обробки запитів використовується HTTP, який заснований на компонентах Katana і специфікації OWIN. Його модульність дозволяє легко додати свої власні компоненти. Також ASP.NET Core дозволяє використання Web API для побудови програми ASP.NET, що спеціально створений для роботи в стилі REST (Representation State Transfer або "передача стану"). REST-стиль особливо зручний при створенні будь-якого Single Page Application (SPA), які нерідко використовують спеціальні Javascript-фреймворки, наприклад, Angular, React або Vue, і який було використано у даному проекті спільно з React”.

“React – відкрита JavaScript бібліотека для створення інтерфейсів користувача, яка покликана вирішувати проблеми часткового оновлення вмісту вебсторінки, з якими стикаються в розробці односторінкових застосунків. React дозволяє створювати великі вебзастосунки, які використовують дані, котрі змінюються з часом, без перезавантаження сторінки. React обробляє тільки користувацький інтерфейс у застосунках. Це відповідає видові у шаблоні модель-вид-контролер (MVC)”.

HTTPS використовує шифрування для забезпечення безпеки під час передачі даних між клієнтом і сервером в Інтернеті. При відправленні запиту на HTTPS-ресурс браузер клієнта відправляє запит на сервер захищеної веб-сторінки. Запит

містить інформацію про те, що браузер хоче встановити захищене з'єднання. Після отримання запиту веб-сервер, який підтримує HTTPS, відповідає, і процес рукостискання SSL/TLS починається. Це перший етап у встановленні захищеного з'єднання. Під час рукостискання браузер і сервер обмінюються криптографічними ключами і встановлюють спільний ключ для шифрування та розшифрування даних. Після успішного рукостискання SSL/TLS весь трафік між браузером і сервером шифрується. Це означає, що будь-яка інформація, яка передається між клієнтом і сервером, залишається зашифрованою і не доступною для перехоплення або читання третіми особами. Після встановлення захищеного з'єднання клієнт і сервер можуть обмінюватися даними. Завершальним етапом браузер отримує всю необхідну інформацію і завершує відображення веб-сторінки, з'єднання закривається.

Таким чином у нас є передача даних між клієнтом та сервером (рисунок 2.9).

Діаграма прецедентів (Case-diagram) [28] — це графічне зображення можливої взаємодії користувача з системою. Діаграма варіантів використання показує різні варіанти використання та різні типи користувачів, які має система, і часто супроводжуватиметься діаграмами інших типів. Варіанти використання представлені колами або еліпсами [28].

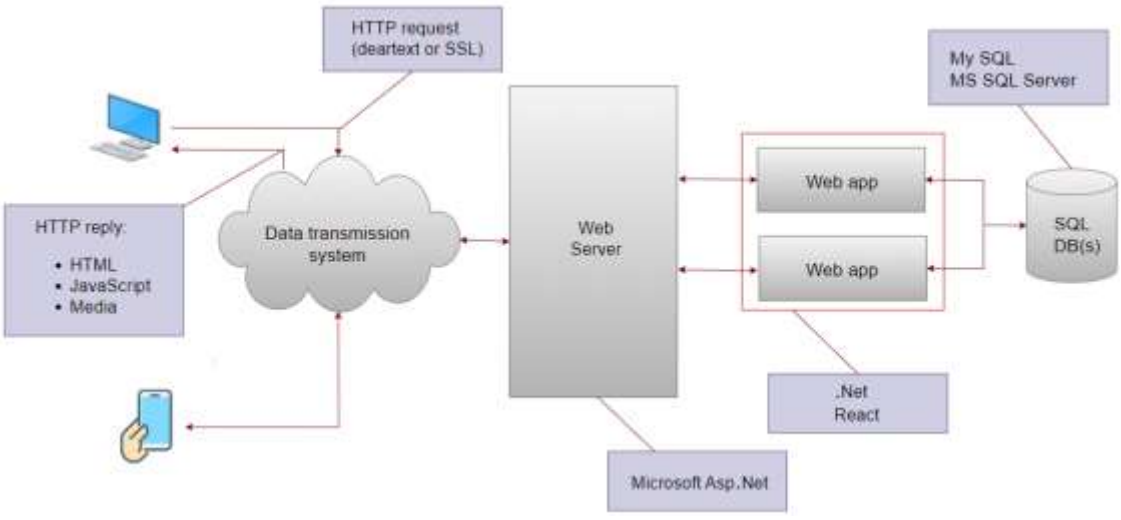


Рисунок 2.9 - Архітектура системи

В уніфікованій мові моделювання (UML) діаграма варіантів використання узагальнює деталі користувачів системи та їх взаємодію з системою. Щоб створити його, було використано набір спеціалізованих символів і з'єднувачів.

Діаграма варіантів використання відображає огляд взаємозв'язків на загальному рівні між акторами та системами (рисунок 2.10).

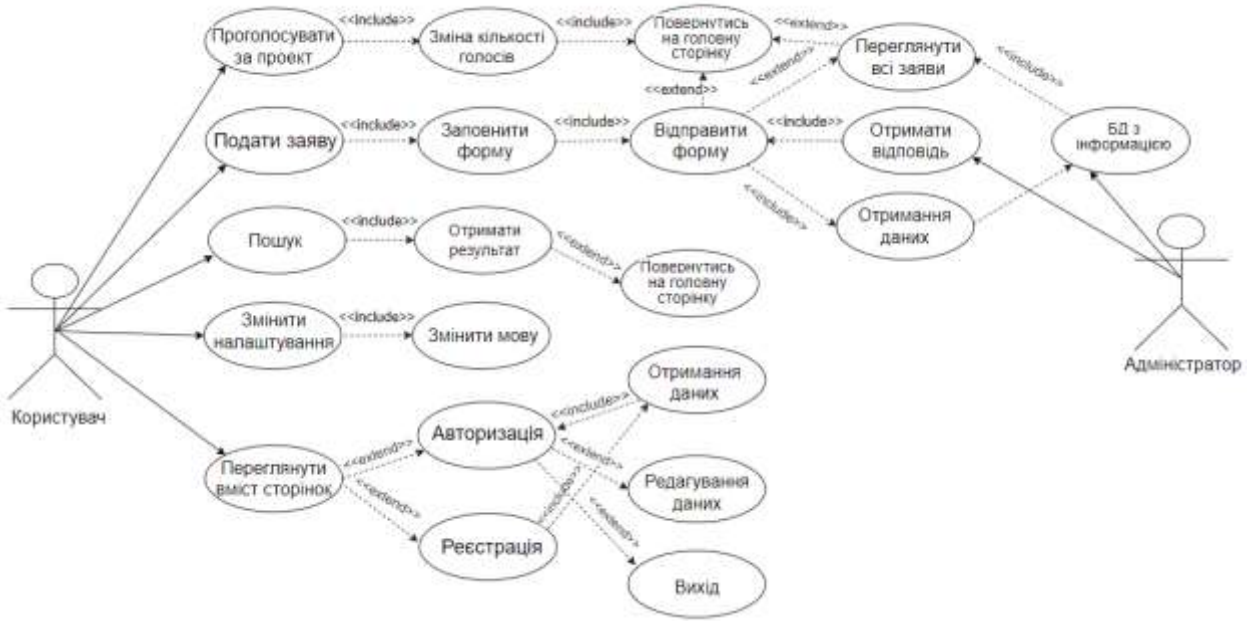


Рисунок 2.10 – Діаграма прецедентів

Діаграма UML використана для:

– Відображення цілей взаємодії системи та користувача

– Визначення та організація функціональних вимог у системі

– Моделювання основного потоку подій у випадку використання

Актори – користувачі (мешканці) та адміністратори (представники влади), які взаємодіють із системою, виробляють або споживають дані.

Система, сценарій – певна послідовність дій і взаємодій між акторами та системою.

Цілі – кінцевий результат більшості випадків використання. Успішна діаграма описує дії та варіанти, використані для досягнення мети.

Щодо зв'язку між ними, то було використано два include та extend.

Розширення зв'язку – варіант використання необов'язковий і стоїть після базового варіанту використання. Він представлений пунктирною стрілкою в напрямку базового варіанту використання з позначенням <<extend>>.

Включити зв’язок – варіант використання є обов’язковим і є частиною базового варіанту використання. Він представлений пунктирною стрілкою в напрямку включеного варіанту використання з позначенням <<include>>.

Потоки даних у системі є критичним компонентом, що забезпечує функціонування програми, незалежно від її типу. CRUD операції (Create, Read, Update, Delete) виступають основними механізмами для маніпулювання даними в базах даних та веб-додатках, дозволяючи додавати, отримувати, оновлювати та видаляти дані. При розробці веб-застосунку застосовуються ці операції через HTTP-запити, що забезпечує ефективну взаємодію між клієнтською (React) та серверною (ASP.NET WEB API) частинами системи. Безпека передачі даних забезпечується через протокол HTTPS, який шифрує трафік, захищаючи інформацію від перехоплення. Використання діаграм варіантів використання (Use Case) у UML дозволяє візуалізувати взаємодії користувачів із системою, сприяючи чіткому розумінню функціональних вимог та організації процесів взаємодії.

2.4 Концептуальне інфологічне проектування, опис локальної та глобальної моделі даних

Концептуальне інфологічне проектування - це процес розроблення структури та організації даних для системи або проекту. Для об'єднаної територіальної громади це означає створення моделей даних на двох рівнях: локальному та глобальному [26].

Локальна модель даних описує структуру даних на рівні конкретного об'єкта або системи всередині об'єднаної територіальної громади. Ця модель визначає, як дані будуть організовані та використовуватися в межах конкретного підрозділу або проекту. Включає структуру бази даних для управління інформацією про мешканців, їхні адреси, послуги, що надаються громадою тощо.

Глобальна модель даних описує загальну структуру та взаємозв'язки між різними локальними моделями даних всередині об'єднаної територіальної громади.

Ця модель визначає, як дані обмінюються між різними системами та підрозділами громади, щоб забезпечити єдність та зручність використання.

Локальна модель даних включає реєстрацію мешканців, зберігання особистої інформації про кожного мешканця, таку як ім'я, адреса, дата народження тощо; управління послугами, описує доступні для мешканців послуги та їхні властивості, такі як графік роботи, вартість, умови отримання; фінансова звітність, облік бюджету, витрати та надходження коштів, фінансові звіти; інфраструктура та обслуговування, описується стан та обслуговування інфраструктури на території громади, такі як дороги, комунальні послуги тощо.

Глобальна модель даних забезпечує взаємодію між різними локальними системами для обміну інформацією та забезпечення цілісності даних; включає створення звітів та аналізу даних для прийняття управлінських рішень на рівні громади; забезпечує ефективне використання ресурсів громади, таких як фінанси, людські ресурси, матеріальні активи.

Отже з локальної сторони було розроблено таблицю для мешканців - User, де кожен запис відповідає одному мешканцю, зберігаються дані для реєстрації та авторизації, подача заяв та відповідь на заяви для конкретної особи (рисунок 2.11).

Рисунок 2.11 – Таблиця User

Глобальна модель реалізована через інтеграцію різних баз даних, що забезпечують обмін інформацією між різними локальними системами. Інформація про фінанси громади інтегрована з поданням звернень, щоб розраховувати скільки необхідно виділити на відшкодування, інші таблиці не є пов’язані, адже співіснують як окремі складові, зберігаючи кожна свою інформацію. Це таблиці підписка, що містить електронні пошти користувачів, які підписались на розсилку

новин, відділи – всі відділи Чорноострівської територіальної громади з відповідними номерами для звернень, інфраструктура – зображення та опис об’єктів територіальної громади, голосування за проєкти – зображення опис та кількість голосів, відданих за конкретний проект, новини – фото, опис всіх новин громади, карта з чіткими координатами та індексами кожної частинки ОТГ (рисунок 2.12).

Рисунок 2.12 – Таблиці глобальної моделі

Концептуальне інфологічне проектування відіграє ключову роль у забезпеченні ефективного управління даними для об'єднаної територіальної громади, створюючи чіткі моделі на локальному та глобальному рівнях. Локальні моделі даних деталізують структуру та організацію інформації на рівні окремих підрозділів або проектів, що дозволяє ефективно керувати даними про мешканців, послуги, фінансову звітність та інфраструктуру. Глобальні моделі, навпаки, забезпечують інтеграцію та обмін даними між різними локальними системами, що сприяє цілісності та узгодженості інформації, необхідної для прийняття управлінських рішень та ефективного використання ресурсів громади. Таким чином, через взаємодію локальних і глобальних моделей досягається оптимальне управління даними, що є фундаментальним для успішного функціонування громади.

3 ПРОГРАМНО-ТЕХНОЛОГІЧНЕ ЗАБЕЗПЕЧЕННЯ СИСТЕМИ

3.1 Інтелектуальний модуль

Go, також відома як Golang, була розроблена компанією Google у 2007 році та офіційно представлена у 2009 році. Мова створена для ефективної роботи з сучасними системами і великими масштабованими програмами. Go має чіткий і лаконічний синтаксис, що робить його легким для вивчення та використання. Go підтримує конкурентність через горутини (goroutines) та канали (channels). Це дозволяє легко створювати паралельні програми, що можуть ефективно використовувати багатоядерні процесори.

Боти часто потребують обробки великої кількості одночасних запитів. Горутини і канали в Go дозволяють легко реалізовувати асинхронну обробку запитів та ефективно використовувати ресурси системи, компілюється у швидкий машинний код, що дозволяє ботам швидко реагувати на запити користувачів. Статична типізація і компіляція допомагають виявляти помилки на етапі розробки, що підвищує надійність роботи ботів. Go має вбудовану підтримку роботи з мережевими протоколами, що спрощує розробку ботів, які взаємодіють з API інших сервісів.

При розробці інтелектуального модуля телеграм-бота на Go були використані наступні технології та підходи:

1. Бібліотека для роботи з Telegram Bot API. Використана бібліотека go-telegram-bot-api, яка забезпечує простий і зручний інтерфейс для взаємодії з Telegram Bot API.

2. Конкурентна обробка запитів. Для обробки повідомлень від користувачів використовуються горутини. Це дозволяє ботам обробляти кілька запитів одночасно, забезпечуючи високу продуктивність.

3. Інтеграція з NLP та ML сервісами. Для розуміння природної мови та генерації відповідей використовуються зовнішні сервіси NLP (Natural Language Processing) та ML (Machine Learning), такі як OpenAI API, Google Cloud Natural Language API тощо.

4. Зберігання та управління даними. Використання баз даних для зберігання інформації про користувачів та їхні запити. Це дозволяє ботам бути більш контекстно обізнаними та надавати персоналізовані відповіді.

5. Тестування та налагодження. Використання вбудованих інструментів для тестування та налагодження коду, що забезпечує високу якість розробки.

Розроблений чат бот працює за схемою (див. Рисунок В.1, Додаток В).

1. Користувач надсилає повідомлення системі, воно може бути текстовим, голосовим або візуальним.

2. Система класифікує намір повідомлення. Це означає, що система намагається визначити, що користувач хоче зробити своїм повідомленням. Наприклад, користувач може хотіти поставити запитання, надати відгук або запросити допомогу.

3. Система визначає сутності в повідомленні. Сутності – це об’єкти, про які йдеться в повідомленні. Наприклад, у повідомленні "Не вдається подати скаргу" сутностями є "подати" та "скаргу".

4. Система вибирає відповідь, яка, на її думку, найкраще відповідає наміру та сутностям повідомлення користувача.

5. Система надсилає відповідь користувачеві (рисунок 3.2).

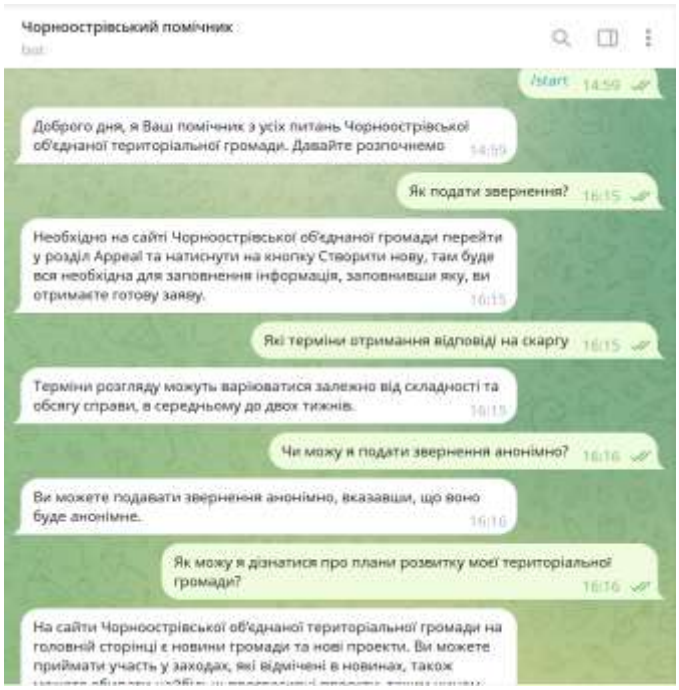


Рисунок 3.2 - Розроблений Телеграм-бот

3.2 Тестування

Веб-тестування є важливою складовою процесу розробки програмного забезпечення. Включає в себе тестування веб-сайту або додатку на відповідність функціональним вимогам. Наприклад, перевірка роботи кнопок, форм, функціональності пошуку, реєстрації, авторизації тощо.

Функціональне тестування, яке є частиною веб-тестування, спрямоване на перевірку того, чи працює система відповідно до очікуваних функцій та вимог.

Позитивний сценарій у функціональному тестуванні відображає ситуацію, коли програмне забезпечення працює як очікувалося і відповідає всім специфікаціям. Наприклад, якщо веб-сайт має форму для реєстрації, позитивний сценарій буде той, коли користувач може успішно заповнити форму, натиснути кнопку "Відправити" і отримати підтвердження про успішну реєстрацію.

Негативний сценарій включає у себе ситуації, коли програмне забезпечення не працює так, як очікувалося або не відповідає вимогам. Наприклад, у разі форми реєстрації негативний сценарій може включати спроби відправити форму з недопустимими даними (наприклад, неправильний формат електронної пошти або пароль занадто короткий) або спроби відправити форму без обов'язкових полів.

Під час розробки сайти для автоматизованої Чорноострівської територіальної громади, було проведене веб-тестування для визначення позитивних та негативних сценаріїв. Було протестовано форму для реєстрації, форму для нових звернень, поле пошуку та функцію для кабінету користувача. При тестуванні були виявлені сценарії зображені на рисунку 3.3.

Під час функціонального тестування розробники та тестувальники виявляють проблеми та дефекти в програмному забезпеченні, які можуть виникати як у позитивних, так і у негативних сценаріях. Ці проблеми потім реєструються в баг-репортах, що включають опис проблеми, кроки для відтворення, очікувані результати та фактичні результати.

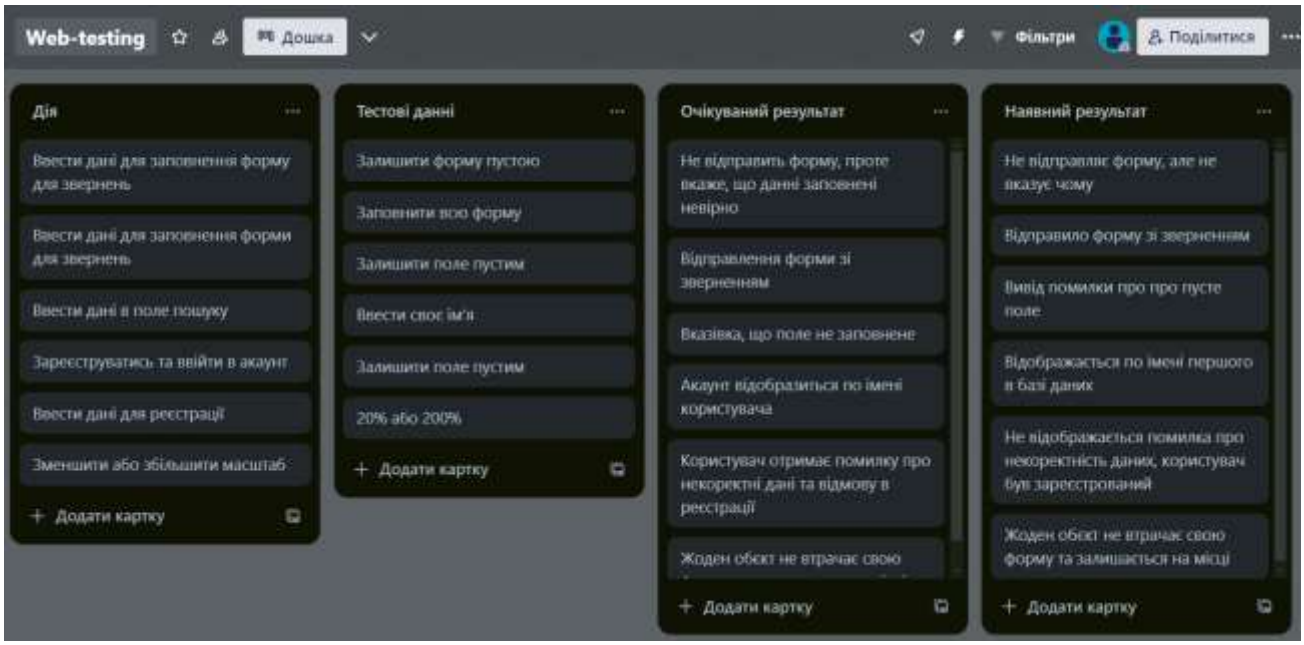


Рисунок 3.3 - Веб-тестування

Так як при розробці були виявлені негативні сценарію, то були опрацьовані баг-репорти, щоб дослідити та усунути проблему (рисунки 3.4- 3.7).

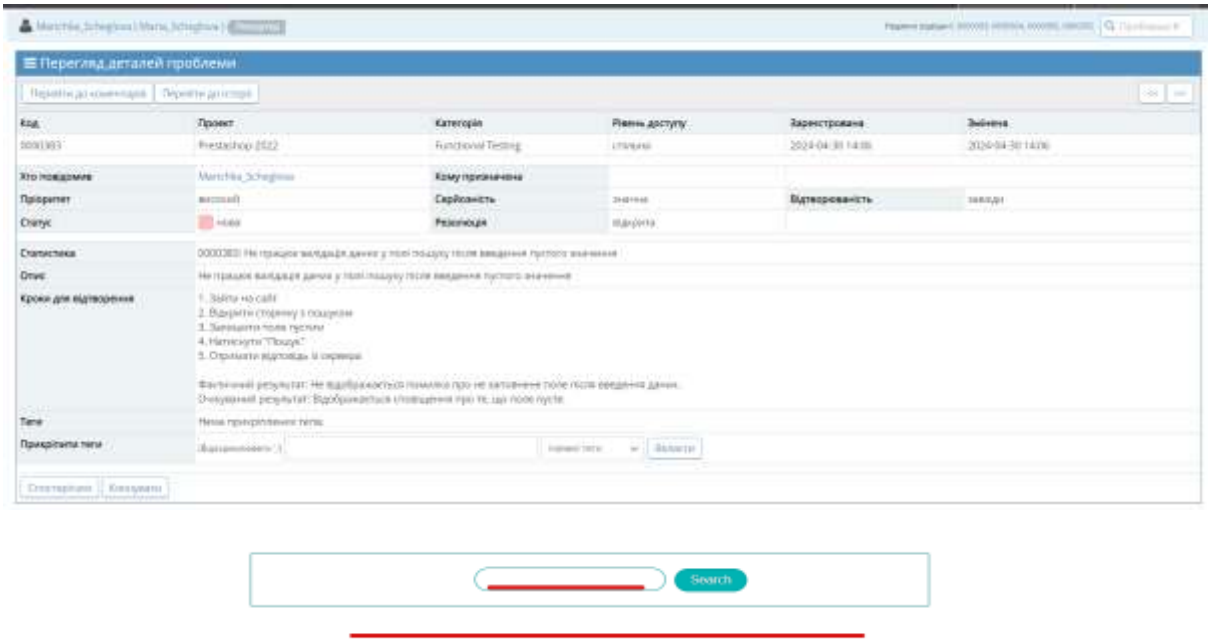


Рисунок 3.4 - Баг-репорт валідації даних у полі пошуку

Було опрацьовано сторінку подання звернень, та виявлено баг у формі заповнення даних, де залишивши поле пустим, не спрацьовує валідація, опис якого зображено на рисунку 3.5.

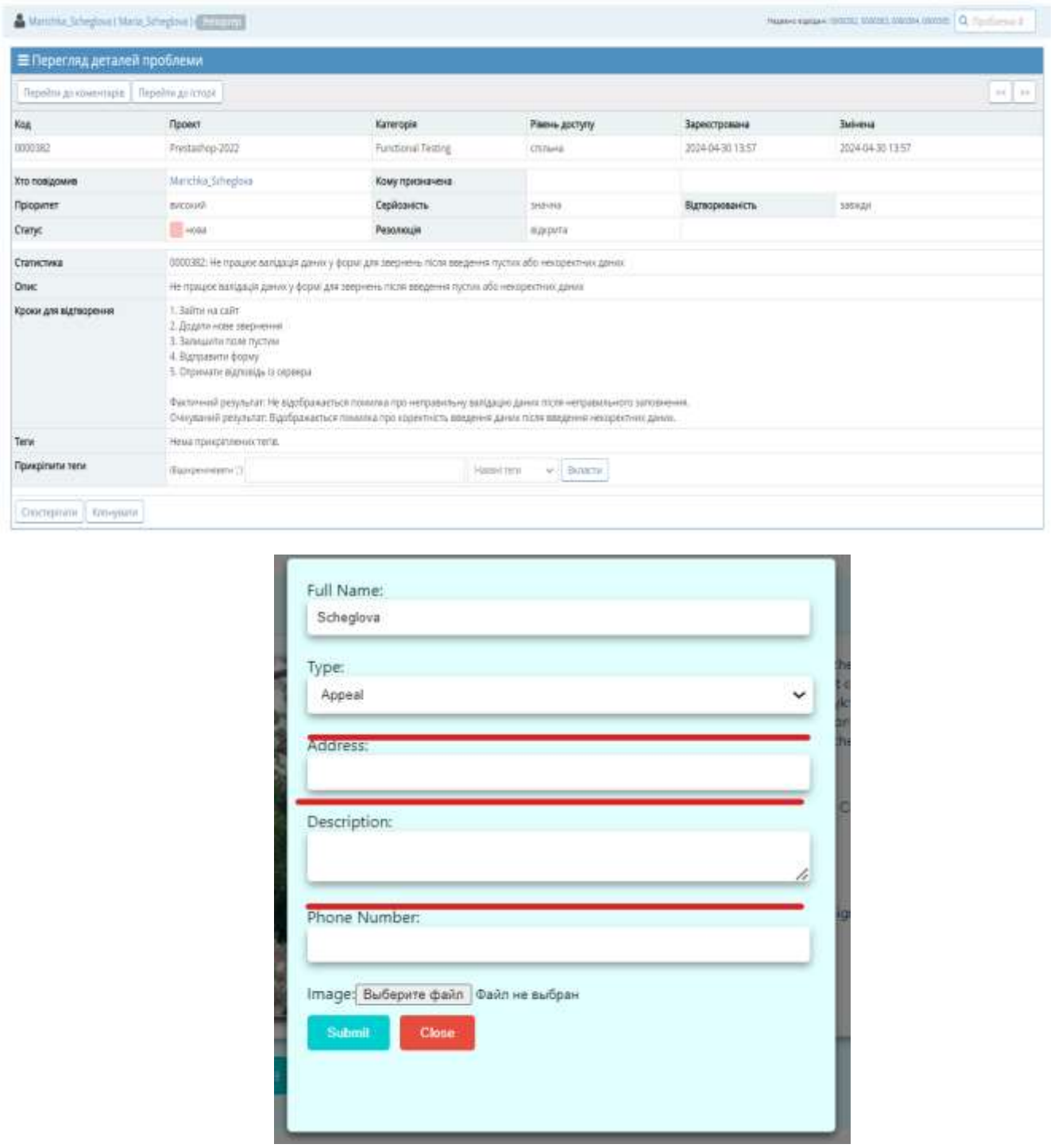
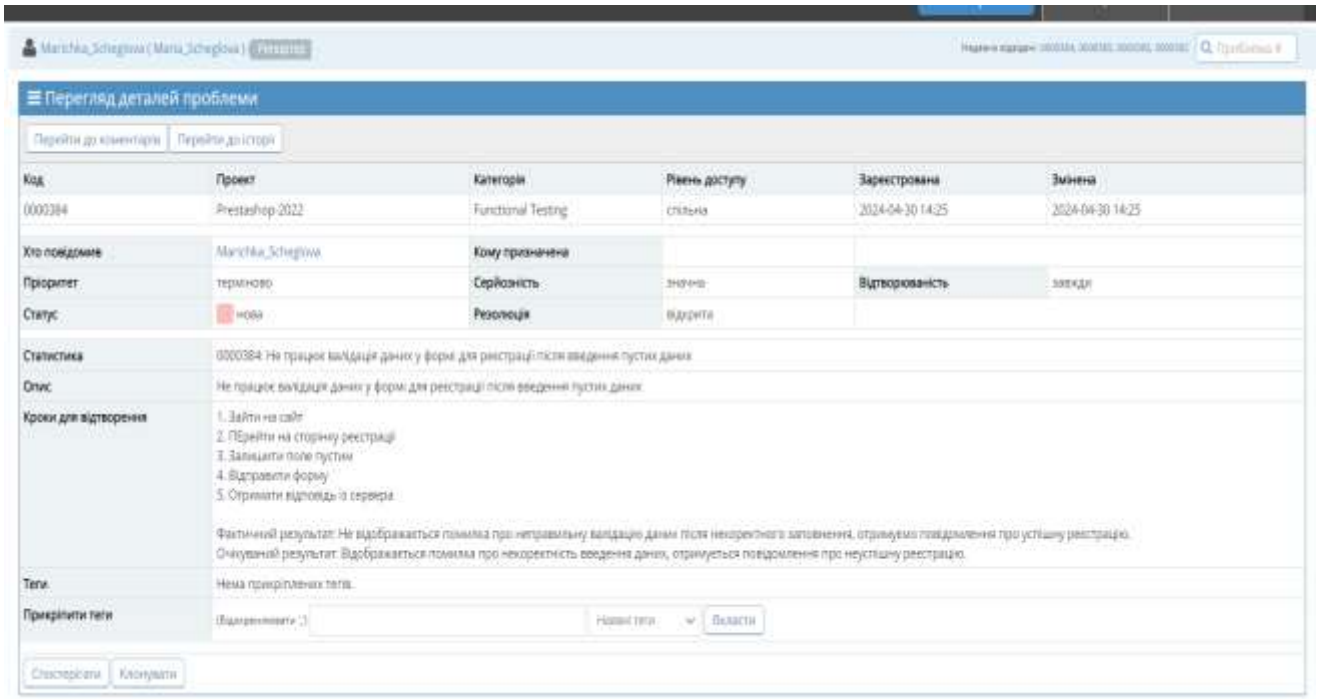


Рисунок 3.5 - Баг-репорт валідації даних у формі звернень

Було опрацьовано сторінку реєстрації користувача, та виявлено баг у формі, де залишивши поле пустим, не спрацьовує валідація та не відображається помилка про некоректність даних, опис якого зображено на рисунку 3.6.



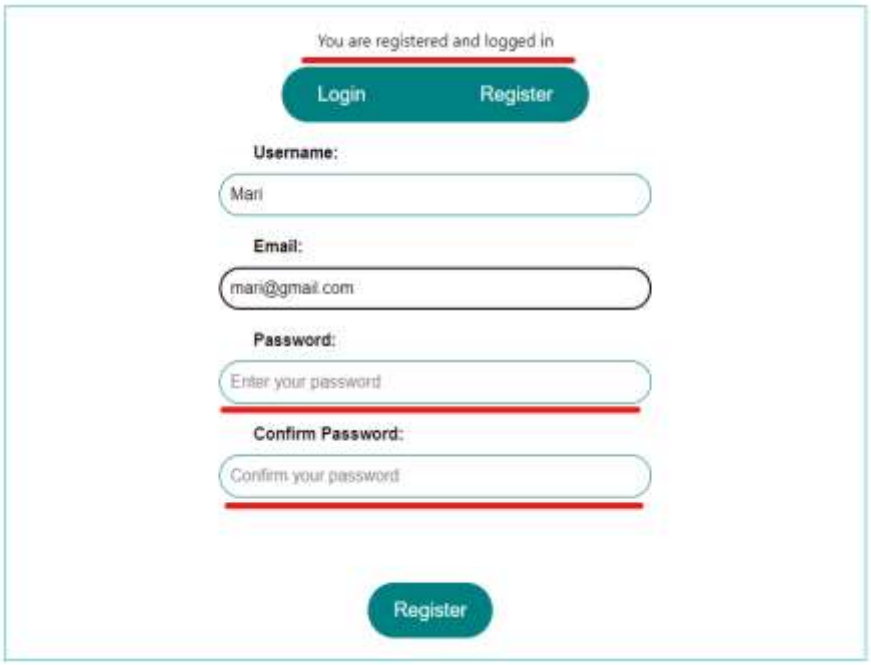


Рисунок 3.6 - Баг-репорт пусті дані у формі реєстрації

При вході в обліковий запис відображається некоректна інформація про користувача, та значок імені у верхньому віконечку (рисунок 3.7).

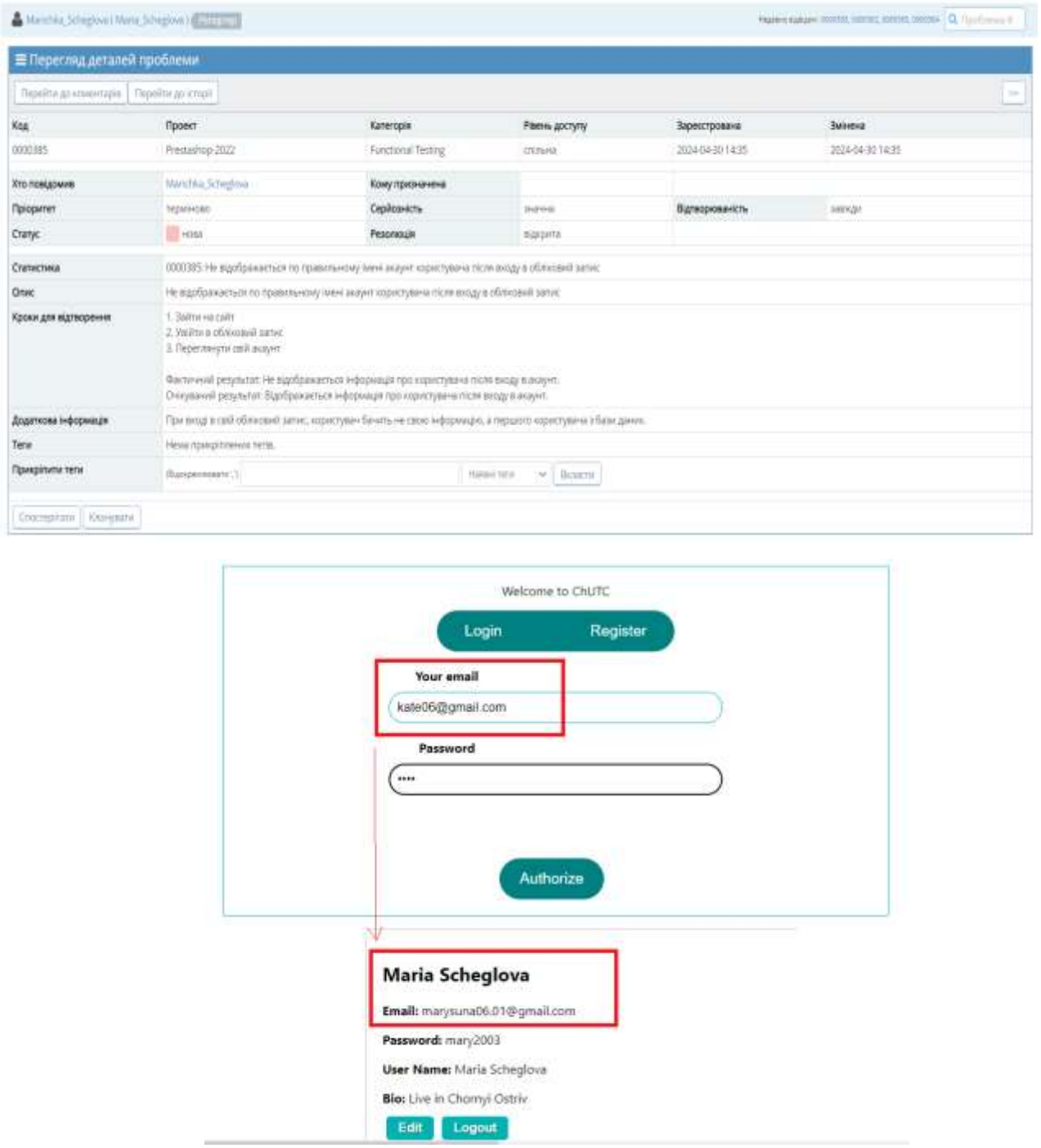


Рисунок 3.7 – Баг-репорт про невідповідність зареєстрованого користувача

Після створення баг-репортів було виправлено знайдені дефекти, та перевірено, чи виправлені проблеми та чи не виникають нові. Цей процес покращив якість програмного забезпечення та забезпечив більш стабільну та надійну роботу розробленої системи.

3.3 Сценарій роботи користувача з системою

Користувач відкриває веб-сайт Чорноострівської об’єднаної територіальної громади і вводить свої облікові дані або реєструється у системі, якщо це потрібно (див. Рисунок В.8 – В.9, Додаток В). Також для користувача буде доступний його акаунт, де він зможе переглянути та змінити інформацію або вийти з облікового запису (див. Рисунок В.10, Додаток В).

Після входу користувач отримує доступ до основних функцій системи. Він може переглядати новини та оголошення (див. Рисунок В.11, В.14, Додаток В), отримувати актуальну інформацію про проекти (див. Рисунок В.12, В.13, Додаток В), рішення органів влади та інші аспекти життя територіальної громади, переглядати відділи ОТГ (див. Рисунок В.19, Додаток В) та карти з точними координатами селищ (див. Рисунок В.20, Додаток В). Також доступні функція пошуку (див. Рисунок В.25, Додаток В), перегляду фото та відео з гугл диску (див. Рисунок В.15, Додаток В) та зміни мови (див. Рисунок В.24, Додаток В).

Користувач може взаємодіяти з системою для отримання різних послуг та сервісів по типу подачі звернень (див. Рисунок В.17, В.18, Додаток В). Сторінка звернень містить у собі інформацію про типи заяв, звернень і скарг, які можуть бути створені, а також містить форму для подачі звернення, та опісля сформований лист. Мешканець може надсилати запити на інформацію, слідкувати за статусом заявок та отримувати відповіді від органів влади.

Користувач може брати участь у громадських обговореннях, голосуваннях або опитуваннях через систему (див. Рисунок В.13, Додаток В). Прочитавши інформацію, користувач може віддати свій голос за новий проект, який буде реалізовано.

Користувач може отримувати сповіщення про важливі події, зміни у роботі органів влади, нові послуги або події, що цікавлять його, через систему повідомлень електронної пошти або мобільний додаток, підписавшись на оновлення (див. Рисунок В.23, Додаток В).

ВИСНОВКИ

Дослідження інтелектуальної системи управління для територіальних громад України демонструє значний потенціал у підвищенні адміністративної ефективності та залученні громадян до управління та відбудови після війни. Реалізація системи призвела до вражаючих результатів, що відображаються в кількісних показниках ефективності. Зокрема, автоматизація процесів управління дозволила скоротити час на обробку звернень громадян на 70%, що значно оптимізувало роботу адміністративних органів та покращило якість надання послуг.

Впровадження алгоритмів машинного навчання та аналітичних інструментів сприяло підвищенню точності прогнозування потреб громади, що дозволило досягти 80% ефективності у плануванні та реалізації інфраструктурних проектів. Така оптимізація ресурсів сприяє ефективнішому використанню бюджетних коштів та спрямуванню їх на найбільш нагальні потреби громад.

Залучення громадян до процесів управління через інтерактивні платформи та мобільні додатки підвищило рівень громадської активності, де участь у голосуваннях та опитуваннях зросла на 50%. Це не тільки сприяло підвищенню прозорості та відповідальності місцевої влади, але й зміцнило довіру громадян до адміністративних інституцій.

На основі аналізу отриманих даних та порівняльного аналізу з іншими системами можна зробити висновок, що впровадження інтелектуальної системи управління ОТГ в Україні є ефективним інструментом для підвищення адміністративної ефективності, залучення громади до управлінських процесів та підтримки сталого розвитку територіальних громад у поствоєнний період. Враховуючи позитивні результати дослідження, рекомендується подальше розширення функціоналу системи, інтеграція нових технологічних рішень та адаптація до змінних умов для забезпечення ефективного відновлення та розвитку України.

Майбутні наукові дослідження в області інтелектуальних систем управління територіальними громадами (ОТГ) в Україні можуть зосередитися на розширенні функціональності та інтеграції передових технологій, таких як штучний інтелект (ШІ), блокчейн та Інтернет речей (IoT), для підвищення ефективності, безпеки та прозорості управління. Особлива увага може бути приділена створенню адаптивних моделей машинного навчання, здатних прогнозувати соціально-економічні тенденції та аналізувати великі обсяги даних для прийняття обґрунтованих рішень у реальному часі.

Також важливим напрямком є розробка захищених комунікаційних платформ для забезпечення надійного обміну інформацією між мешканцями та органами влади, що сприятиме залученню громади до управління та розвитку територій. Ці дослідження не тільки сприятимуть технологічному прогресу в управлінні територіальними громадами, але й забезпечать стійкий розвиток та відновлення України в поствоєнний період.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

[1] Місцеве самоврядування в Україні. Офіційний вебпортал парламенту України - [Електронний ресурс]. URL: http://old.phm.gov.ua/node/1184 (стаття 13).

[2] Місцеве самоврядування в Україні. Офіційний вебпортал парламенту України - [Електронний ресурс]. URL: http://old.phm.gov.ua/node/1184 (стаття 5).

[3] Офіційний веб-сайт Чорноострівської об'єднаної територіальної громади - [Електронний ресурс]. URL: https://chornoostrivska-otg.gov.ua/ (українською мовою).

[4] Khort, D., Kutyrev, A., Smirnov, I., & Voronkov, I. (2021). Розвиток автоматизованої системи управління сільськогосподарськими технологіями в городництві. URL: https://dx.doi.org/10.22314/2073-7599-2021-15-2-61-68.

[5] Jiang, H. (2022). Дизайн та впровадження моделі динамічного аналізу великих даних розумного співтовариства на основі моделі логістичної регресії. URL: https://dx.doi.org/10.1155/2022/4038084.

[6] Kišcan, G., Kolarić, S., Šagovac, M., & Baus, Z. (2016). Використання інтелектуальних систем управління для динамічного управління мережею розумної сітки. URL: https://dx.doi.org/10.1109/SST.2016.7765630.

[7] Mukhopadhyay, S., Singh, V., & Babbar-Sebens, M. (2014). Моделювання користувача з обмеженими даними: застосування до проектування басейнів за участю стейкхолдерів. URL: https://dx.doi.org/10.1109/SMC.2014.6974532.

[8] Jain, A., Patel, N., Hammonds, P., & Pandey, S. (2018). Розумна програмна система для управління забезпеченням потоку. URL: https://dx.doi.org/10.2118/191951-MS.

[9] Dr. Anusiuba Overcomer Ifeanyi Alex. (2023). Автоматизована та надійна веб-система управління скаргами студентів терціарного навчального закладу. URL: https://dx.doi.org/10.47001/irjiet/2023.712016.

[10] Wang, Y. (2015). Дизайн мережевої системи управління рухом громад на основі технологій ASP і VB.NET. URL: https://dx.doi.org/10.2991/ICCSET-14.2015.42.

[11] Carpenter, C. (2019). Система програмного забезпечення забезпечує ефективне управління забезпеченням потоку. URL: https://dx.doi.org/10.2118/1119-0084-jpt.

[12] Ko, H., & Marreiros, G. (2019). Розумні медіа та застосунок. URL: https://dx.doi.org/10.1002/cpe.5491.

[13] Radhanand, A., Kumar, K. N., & Namburu, S. (2020). Доступна, масштабована, відкрита архітектура, IoT екосистема для академічної спільноти. URL: https://dx.doi.org/10.2174/2210327910999201029192934.

[14] Khozaimi, A., Negara, Y. D. P., & Syakur, A. (н.д.). Розробка веб-системи інформації про міське село. URL: https://dx.doi.org/10.1051/e3sconf/202132804031.

[15] Bestin, B., & Sulaeman, D. (2020). Розвиток виробництва в міській області. URL: https://dx.doi.org/10.2991/assehr.k.200321.005.

[16] Xiao-ming, Z. (н.д.). Трансформація "Села у місті" в ході урбанізації.

[17] Feng-chun, L. (н.д.). Відображення та дослідження сучасного будівництва міських та сільських громад в Китаї.

[18] Maryunani. (2023). Збільшення участі громади у розвитку села через управління БУМДЕСА у Східній Яві. URL: https://dx.doi.org/10.32535/jcda.v6i2.2278.

[19] Sudiati, L. E., Susandi, G. P., Haryani, N., & Puryono, D. A. (2023). Система управління громадою Mlatiharjo Urban Village PKK в управлінні системою управління підконтрольною Державній адміністрації села Семаранг. URL: https://dx.doi.org/10.55927/eajmr.v2i12.6770.

[20] Su, M., Sun, Y.-h., Min, Q., & Jiao, W. (2018). Підхід до забезпечення засобів існування спільноти у вирощуванні сільськогосподарської спадщини та розвитку туризму: Xuanhua Grape Garden Urban Agricultural Heritage Site, Hebei Province of China. URL: https://dx.doi.org/10.3390/SU10020361.

[21] Putra, I. W. E. D., & Pratama, R. A. (2014). Перетворення культури: покращення та інтеграція соціального капіталу для підтвердження контролю за земельним використанням. URL: https://dx.doi.org/10.14710/IJPD.1.1.51-56.

[22] Zhang, Y., & Jiang, X.-f. (2019). Інтелектуальна система спільноти на основі Інтернету речей: дослідження і аналіз дизайну. Тези доповідей Міжнародної конференції з соціальних наук, громадського здоров'я та освіти 2019 року (SSPHE 2019). URL: https://dx.doi.org/10.25236/FSST.20190316.

[23] Місцеве самоврядування в Україні. Офіційний вебпортал парламенту України - [Електронний ресурс]. URL: http://old.phm.gov.ua/node/1184 (стаття 5) (українською мовою).

[24] Приклади видобування даних - [Електронний ресурс]. URL: https://uk.myserver (українською мовою).

[25] Електронний репозиторій Ужгородського національного університету - [Електронний ресурс]. URL: dspace.uzhnu.edu.ua.

[26] Концептуальне інфологічне проектування – [Електронний ресурс] URL: https://studfile.net/preview/6329372/page:8/.

[27] Метінвест Діджитал. (2022). Стратегія розумної спільноти. URL: https://metinvest.digital/ua/page/1052

[28] Моделювання проєкту з використанням UML діаграм статичних та динамічних логічних та фізичних. Тернопіль: ЗУНУ, 28 с.

[29] Комар М.П., Саченко А.О., Васильків Н.М., Гладій Г.М., Коваль В.С., Ліп'яніна-Гончаренко Х.В. Методичні рекомендації до виконання кваліфікаційної роботи з освітньо-професійної програми «Комп'ютерні науки» спеціальності 122 «Комп'ютерні науки» за першим (бакалаврським) рівнем вищої освіти. Тернопіль: ЗУНУ, 2024. 52 с.

[30] Загальні методичні рекомендації з підготовки, оформлення, захисту та оцінювання кваліфікаційних робіт здобувачів вищої освіти першого (бакалаврського) і другого (магістерського) рівнів. Тернопіль: ЗУНУ, 2024. 83 с.

[31] Щеглова М., Ліп'яніна-Гончаренко Х.В. Інтелектуальна система управління інфраструктурою територіальної громади. (2024). Вимірювальна та обчислювальна техніка в технологічних процесах, (2), 5-17. https://doi.org/10.31891/2219-9365-2024-78-1

ДОДАТОК А
IDEF-діаграми процесів управління ОТГ

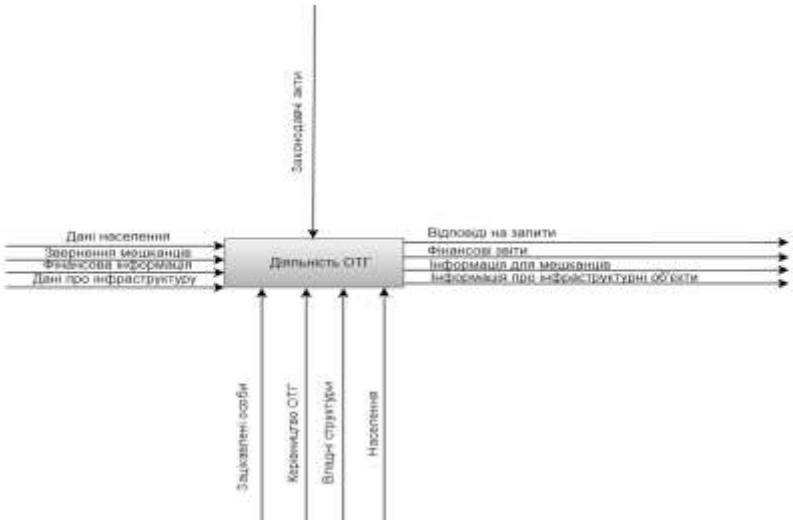


Рисунок А.1 - IDEF3-діаграма діяльності ОТГ

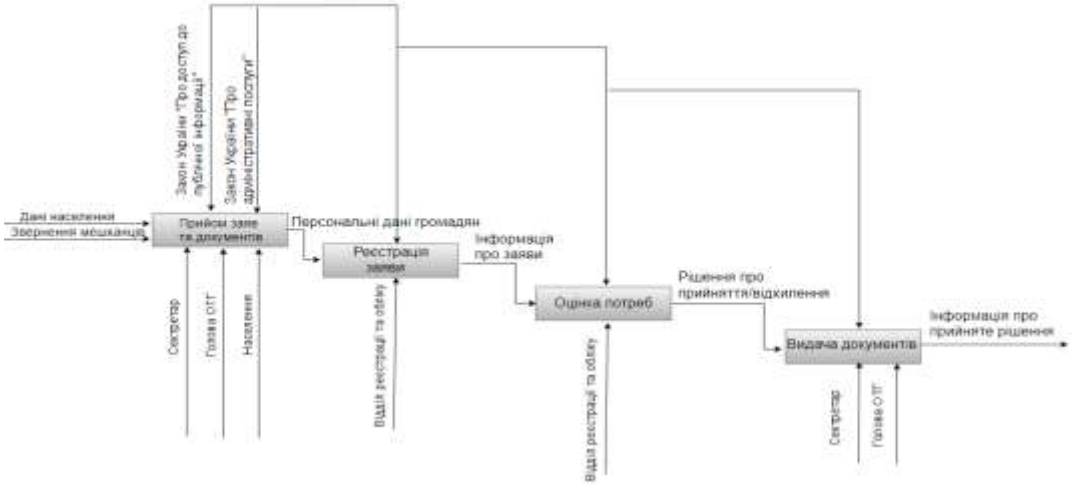


Рисунок А.2 – IDEF3-діаграма декомпозиції процесу обслуговування населення

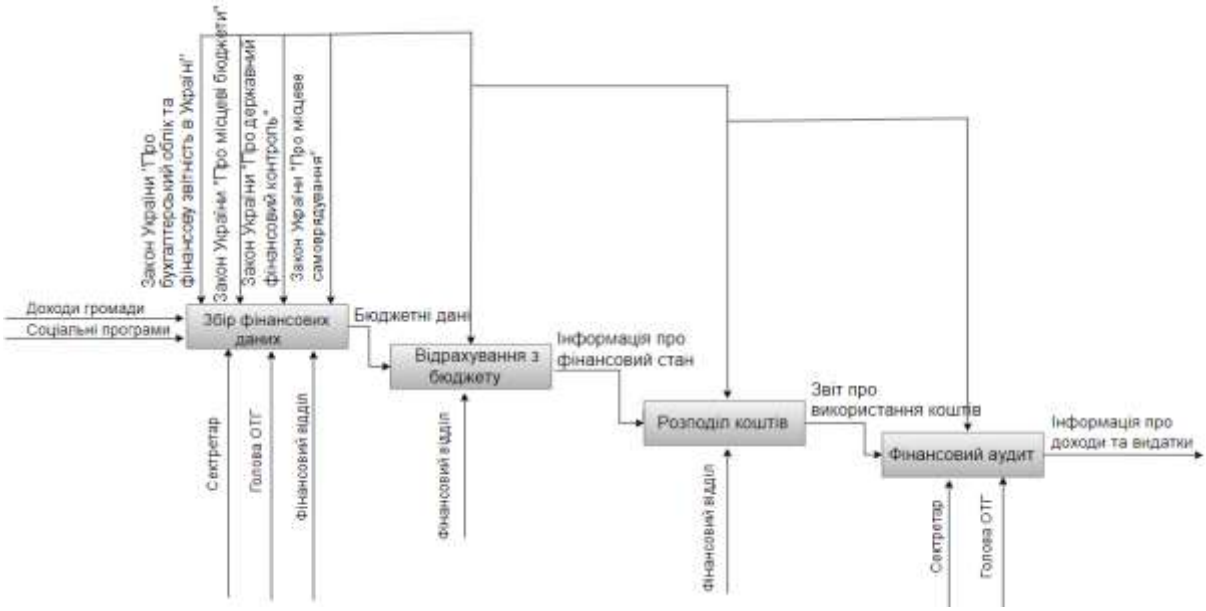


Рисунок А.3 – IDEF3-діаграма декомпозиції процесу фінансового обліку

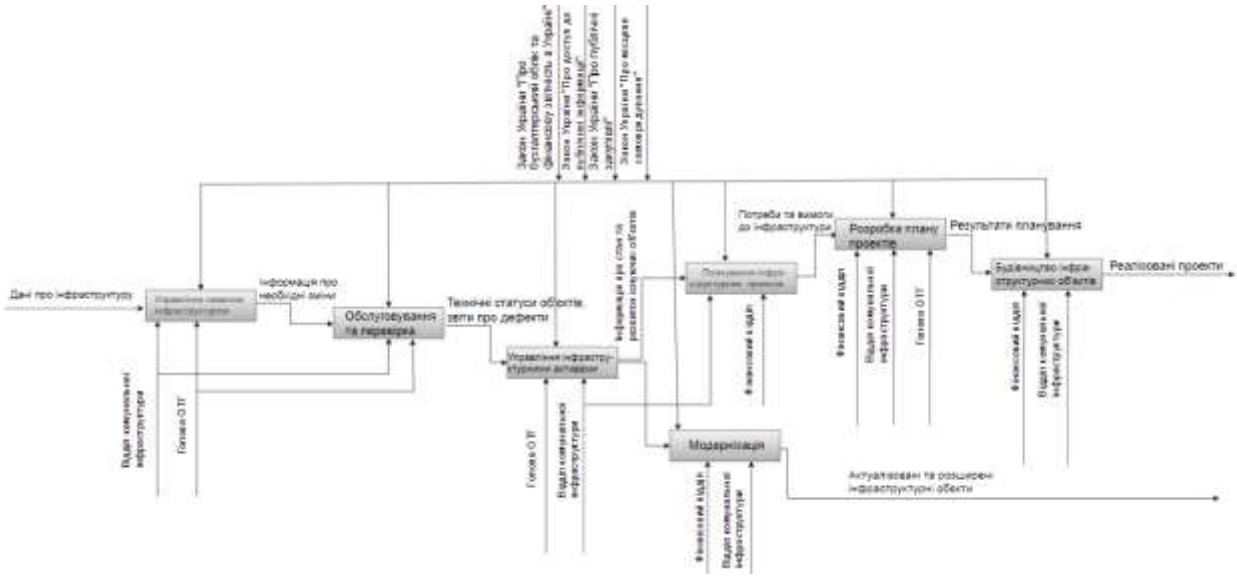


Рисунок А.4 – IDEF3-діаграма декомпозиції процесу управління інфраструктурними об’єктами

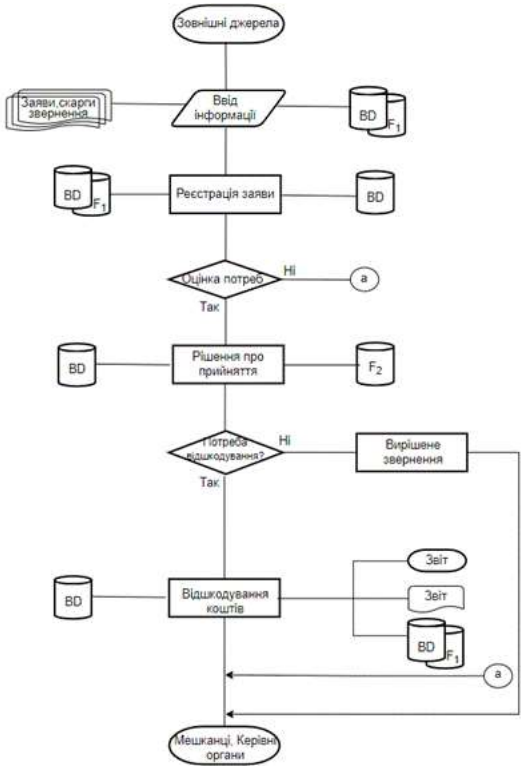


Рисунок А.5 - Схема алгоритму процесу обслуговування населення

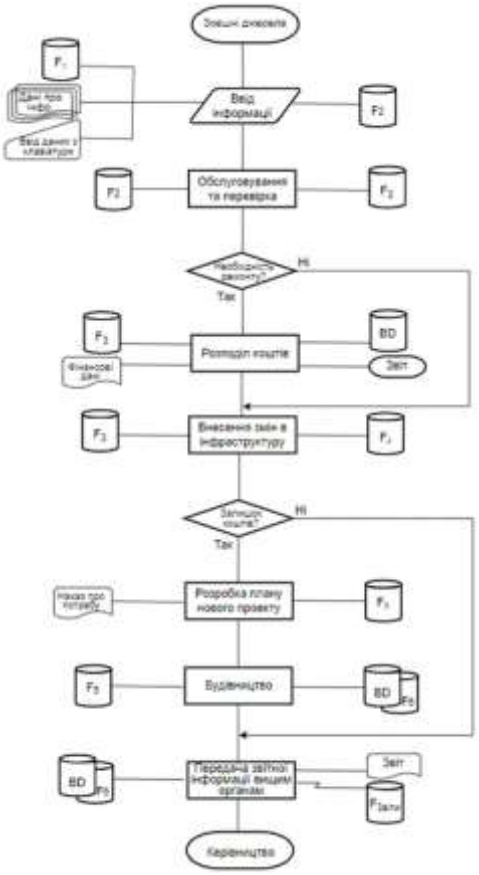


Рисунок А.6 - Схема алгоритму процесу управління інфраструктурними об’єктами

ДОДАТОК Б

Схема процесів та знімки екрану при роботі програми

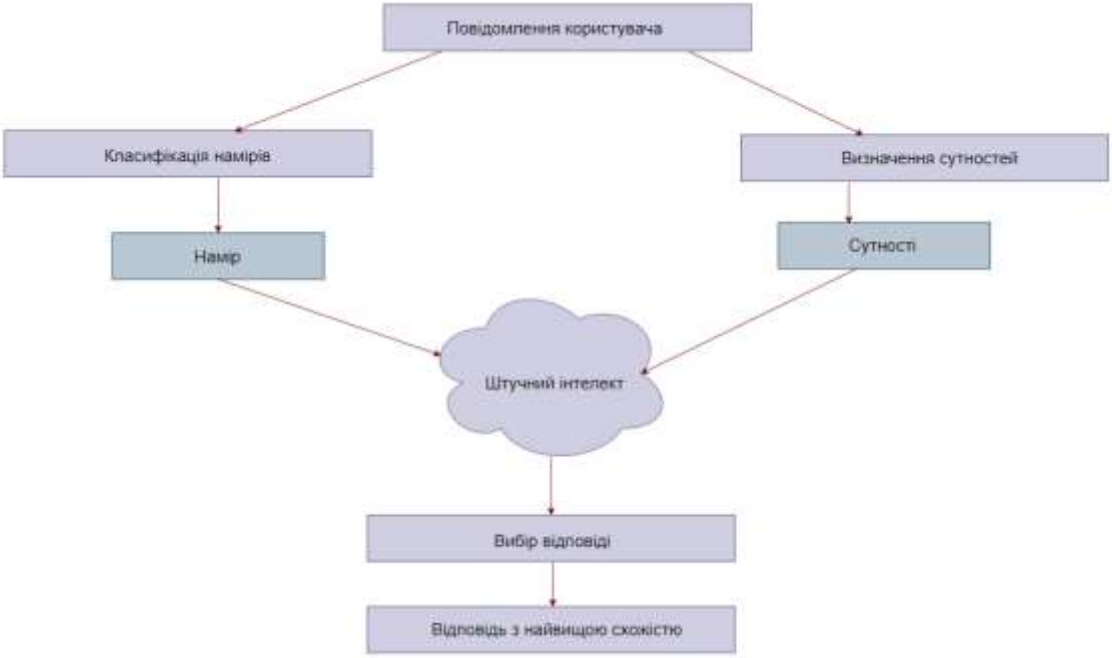


Рисунок В.1 - Схема процесів чат-бота

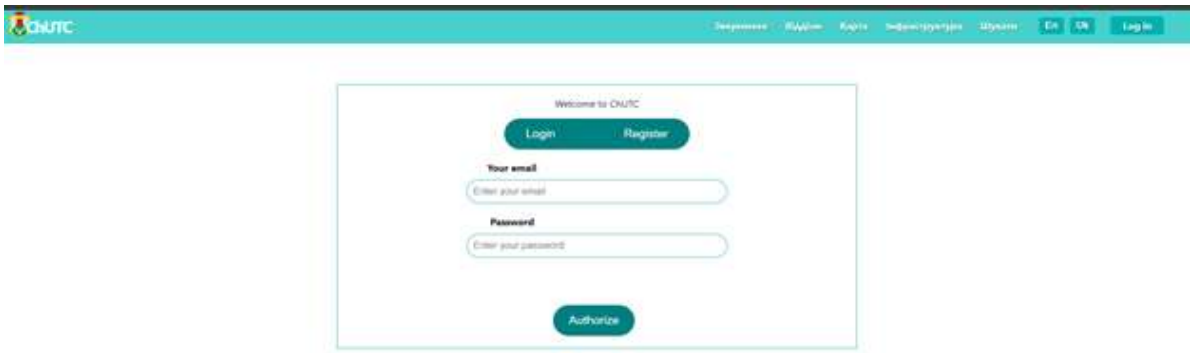


Рисунок В.8 – Вхід в обліковий кабінет

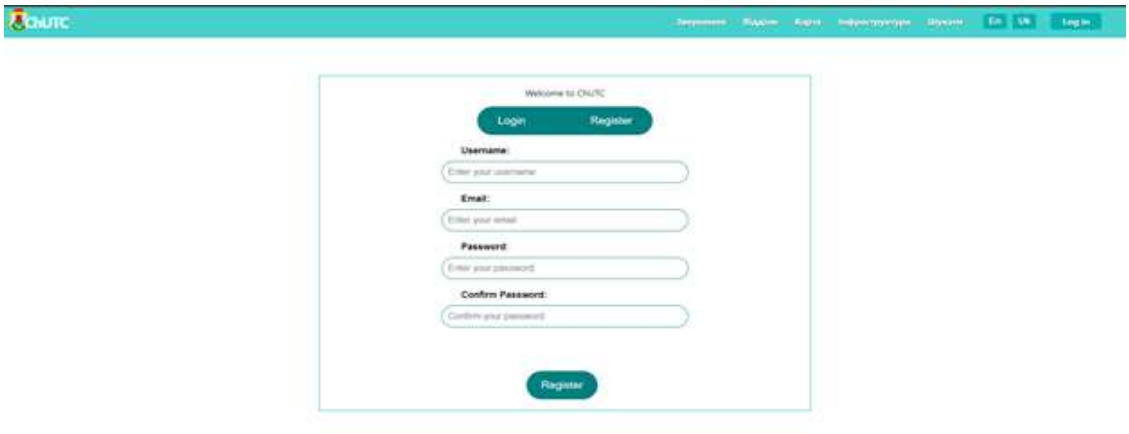


Рисунок В.9 – Реєстрація користувача

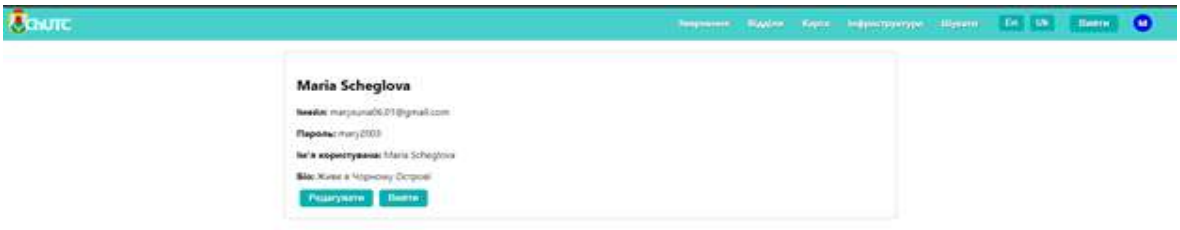


Рисунок В.10 – Кабінет авторизованого користувача

Рисунок В.11 – Головна сторінка

Рисунок В.12 – Банер голосування за проекти

Рисунок В.13 – Сторінка з проектами

Рисунок В.14 – Сторінка з новинами ОТГ

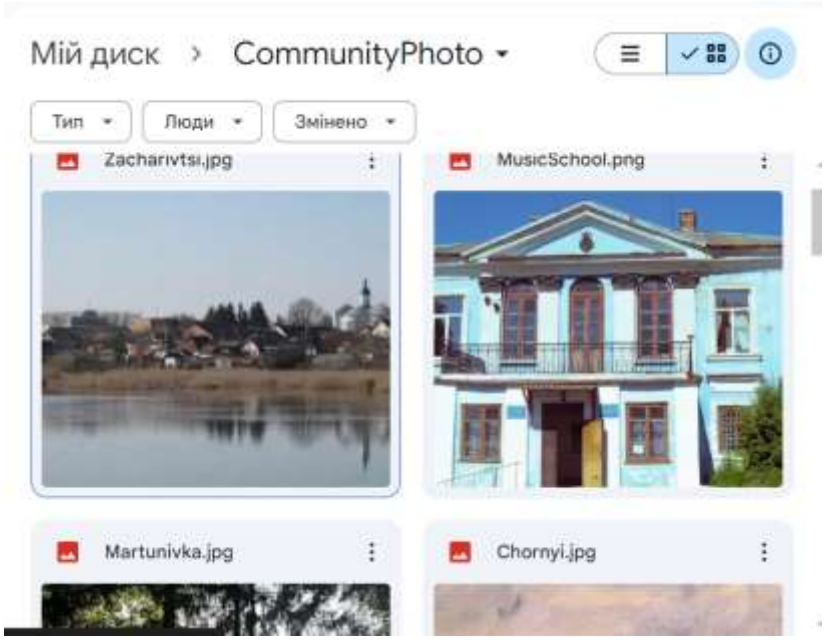


Рисунок В.15 – Гугл диск з фото громади

Рисунок В.16 – Інформація “Про нас” на головній сторінці

Рисунок В.17 – Сторінка звернення

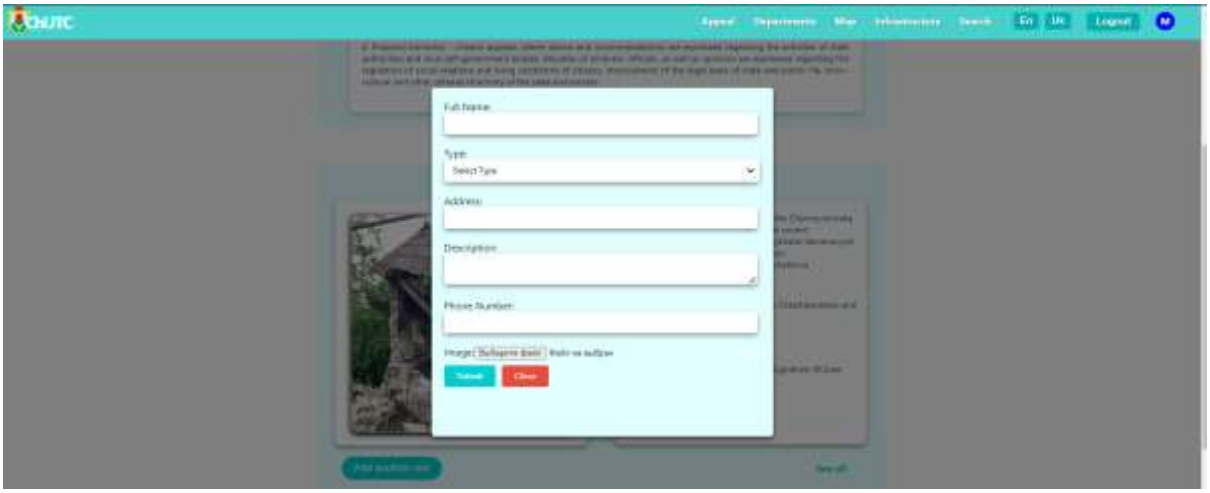


Рисунок В.18 – Форма подачі нового звернення

Рисунок В.19 – Сторінка відділів

Рисунок В.20 – Сторінка карти з індексами територіальних громад

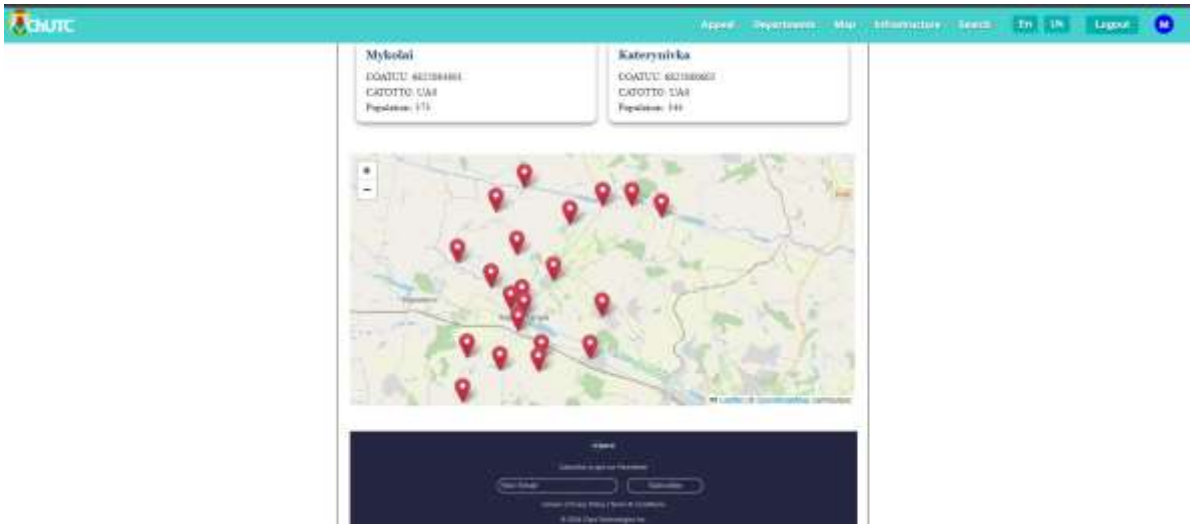
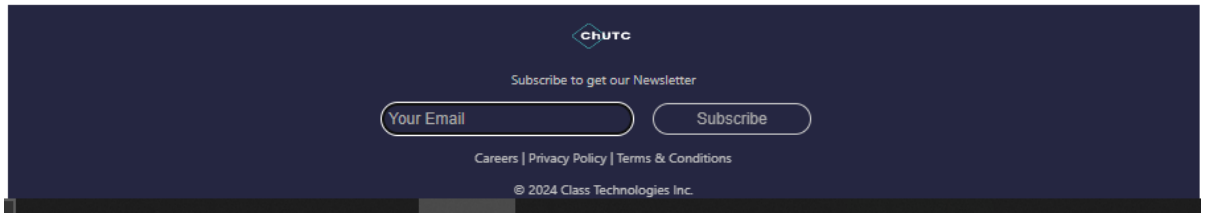


Рисунок В.21 – Координати громад на карті

Рисунок В.22 – Сторінка даних про інфраструктуру



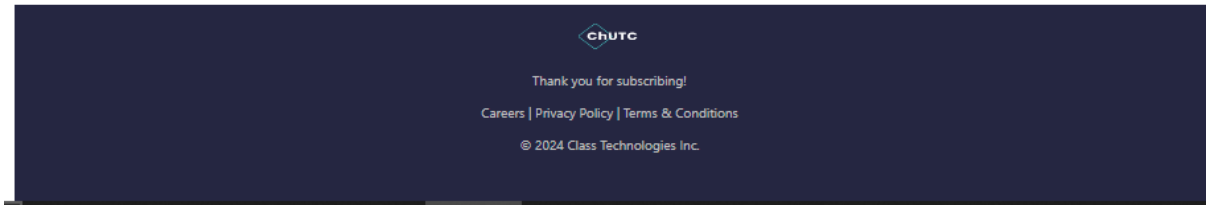


Рисунок В.23 – Подача підписки на розсилку новин

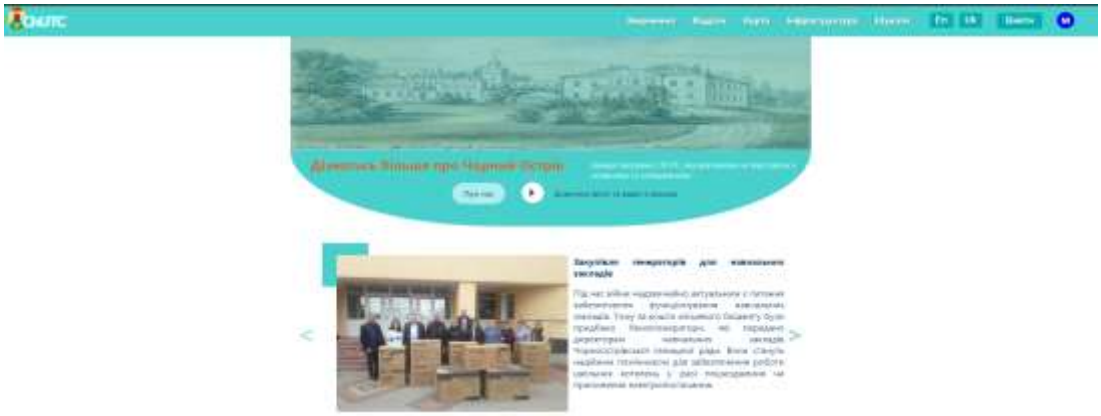


Рисунок В.24 – Застосування функції перекладу мову

Рисунок В.25 – Функція пошуку

ДОДАТОК В
Апробація отриманих результатів

ISSN 2219-9365
DOI: 10.31891/2219-9365

Міжнародний науково-технічний журнал

ВИМІРЮВАЛЬНА ТА ОБЧИСЛЮВАЛЬНА ТЕХНІКА В ТЕХНОЛОГІЧНИХ ПРОЦЕСАХ

2024, № 2

International scientific-technical journal

MEASURING AND COMPUTING DEVICES IN TECHNOLOGICAL PROCESSES

2024, Issue 2

Хмельницький 2024
Khmelnytskyi 2024

МІЖНАРОДНИЙ НАУКОВО-ТЕХНІЧНИЙ ЖУРНАЛ
ВИМІРЮВАЛЬНА ТА ОБЧИСЛЮВАЛЬНА ТЕХНІКА В ТЕХНОЛОГІЧНИХ ПРОЦЕСАХ

Затверджений як фахове видання (перереєстрація), група «Б»
Наказ МОН 28.12.2019 №1643

Засновано в травні 1997 р. ***Виходить 4 рази на рік***

Хмельницький, 2024, № 2 (78)

Засновник і видавець: **Хмельницький національний університет (до 2005 р. — Технологічний університет Поділля, м. Хмельницький)**

Наукова бібліотека України ім. В.І. Вернадського http://nbuv.gov.ua/j-tit/vott

Журнал включено до наукометричних баз:

Index Copernicus http://jml2012.indexcopernicus.com/p24781565.3.html
Google Scholar http://scholar.google.com.ua/citations?user=mN_mxAAAAJ&hl=uk
CrossRef http://doi.org/10.31891/2219-9365

Рекомендовано до друку рішенням Вченої ради Хмельницького національного університету, протокол № 0, від 30.05.2024

Адреса редакції: **Україна, 29016, м. Хмельницький, вул. Інститутська, 11, Хмельницький національний університет, Редакція журналу "Вимірювальна та обчислювальна техніка в технологічних процесах"**

☎ **067-347-74-57**

e-mail: vottp@khmnu.edu.ua

web: http://vottp.khmnu.edu.ua

Зареєстровано Міністерством України у справах преси та інформації.
Свідоцтво про державну реєстрацію друкованого засобу масової інформації
Серія КВ № 24923-14863 ПР від 12 липня 2021 року (перереєстрація)

https://doi.org/10.31891/2219-9365-2024-78-1
УДК 044.11

ЩЕГЛОВА Марія
Західноукраїнський національний університет
e-mail: marysuna06.01@gmail.com
ЛІП'ЯНІНА-ГОНЧАРЕНКО Христина
Західноукраїнський національний університет
https://orcid.org/0009-0009-4376-9585
e-mail: kh.lipianina@wunu.edu.ua

ІНТЕЛЕКТУАЛЬНА СИСТЕМА УПРАВЛІННЯ ІНФРАСТРУКТУРОЮ ТЕРИТОРІАЛЬНОЇ ГРОМАДИ

Це дослідження присвячено розробці та аналізу ефективності інтелектуальної системи управління територіальними громадами (ОТГ) в Україні, з акцентом на значне підвищення адміністративної ефективності та залучення громад до управління в умовах та для потреб поствоєнної відбудови. В результаті впровадження системи було досягнуто 70% зниження часу на обробку звернень громадян та 80% ефективності у плануванні інфраструктурних проектів завдяки застосуванню алгоритмів машинного навчання. Також спостерігалося 50% зростання участі громадян у вирішенні місцевих питань через інтерактивні платформи. Ці результати підкреслюють потенціал інтелектуальної системи як інструменту для оптимізації ресурсів, підвищення прозорості та відповідальності місцевої влади, а також сприяння активній участі громад у процесах прийняття рішень. Дослідження вказує на шляхи подальшого розвитку та інтеграції новітніх технологій у систему управління для забезпечення її адаптивності та гнучкості у відповідь на мінливі соціально-економічні умови, що є ключовим для сталого розвитку та відновлення України.

Ключові слова: аналітика даних, інтерактивна платформа, управління даними мешканців, комунікація з мешканцями, ОТГ.

SCHEGLOVA Maria, LIPIANINA-HONCHARENKO Khrystyna
West Ukrainian National University

INTELLIGENT SYSTEM OF TERRITORIAL COMMUNITY INFRASTRUCTURE MANAGEMENT

In the context of modern challenges facing Ukraine, especially in connection with the ongoing hostilities and the need for further reconstruction of the country, the importance of developing and implementing innovative automated management systems in territorial communities becomes especially relevant. This applies not only to improving the efficiency of administrative processes, but also to ensuring a quick response to the needs of citizens, optimizing the distribution of resources, improving the quality of life of the population and promoting the sustainable development of regions.

In conditions where traditional approaches to the management of territorial communities are not flexible and effective enough to solve emerging problems, the application of modern technological solutions, such as artificial intelligence, machine learning, big data and the Internet of Things, opens up new opportunities for reforming the management system for the benefit of citizens. These technologies allow not only to automate routine processes, but also to analyze large volumes of data to make informed decisions, predict future trends and adapt to changing conditions.

This study focuses on the development and analysis of the effectiveness of an intelligent system for managing territorial communities (TCs) in Ukraine, with a focus on significantly increasing administrative efficiency and community engagement in the context of and for the needs of post-war reconstruction. As a result of the system implementation, a 70% reduction in the time required to process citizen requests and 80% efficiency in planning infrastructure projects was achieved through the use of machine learning algorithms. There was also a 50% increase in citizen participation in solving local issues through interactive platforms. These results emphasize the potential of an intelligent system as a tool for optimizing resources, increasing the transparency and accountability of local authorities, and promoting active community participation in decision-making processes. The study points to ways to further develop and integrate the latest technologies into the governance system to ensure its adaptability and flexibility in response to changing socio-economic conditions, which is key to Ukraine's sustainable development and recovery.

Keywords: data analytics, interactive platform, management of residents' data, communication with residents, ATCs.

ПОСТАНОВКА ПРОБЛЕМИ У ЗАГАЛЬНОМУ ВИГЛЯДІ ТА ЇЇ ЗВ'ЯЗОК ІЗ ВАЖЛИВИМИ НАУКОВИМИ ЧИ ПРАКТИЧНИМИ ЗАВДАННЯМИ

У контексті сучасних викликів, перед якими стоїть Україна, особливо у зв'язку з триваючими воєнними діями та необхідністю подальшої відбудови країни, важливість розробки та впровадження інноваційних автоматизованих систем управління в територіальних громадах набуває особливої актуальності. Це стосується не тільки покращення ефективності адміністративних процесів, але й забезпечення швидкого реагування на потреби громадян, оптимізацію розподілу ресурсів, підвищення якості життя населення та сприяння сталому розвитку регіонів.

В умовах, коли традиційні підходи до управління територіальними громадами виявляються недостатньо гнучкими та ефективними для вирішення виникаючих проблем, застосування сучасних технологічних рішень, таких як штучний інтелект, машинне навчання, великі дані та інтернет речей,

відкриває нові можливості для реформування системи управління на користь громадян. Ці технології дозволяють не лише автоматизувати рутинні процеси, але й аналізувати великі обсяги даних для прийняття обґрунтованих рішень, прогнозування майбутніх тенденцій та адаптації до мінливих умов.

Актуальність цього дослідження полягає в тому, що воно спрямоване на визначення шляхів розробки та впровадження інтелектуальних систем управління в територіальних громадах України, які б допомогли відповісти на виклики, поставлені воєнним станом, та підготувати міцний фундамент для майбутньої відбудови країни. Дослідження зосереджується на аналізі існуючих рішень в цій сфері, виявленні їхніх сильних та слабких сторін, та розробці рекомендацій щодо створення ефективних, гнучких та масштабованих систем, здатних адаптуватися до змінних умов та вимог громад.

Важливо підкреслити, що успішна реалізація цих рішень вимагає не лише технологічних інновацій, але й глибокого розуміння соціально-економічних процесів, що відбуваються в громадах, а також активної участі громадян у процесах прийняття рішень. Таким чином, розробка та впровадження інтелектуальних систем управління виступає не лише як технологічне завдання, але й як важливий крок до створення відкритих, прозорих та ефективних механізмів управління, що сприяють залученню громадян до активної участі в житті своїх громад і в цілому сприяють демократизації суспільства.

Ця стаття зосереджена на розробці та впровадженні інтелектуальної системи управління територіальними громадами (ОТГ) в Україні, акцентуючи на підвищенні адміністративної ефективності та залученні громад в умовах та після воєнних руйнувань. Розділ 2 пропонує огляд існуючих рішень, підкреслюючи важливість автоматизованих систем управління в різних секторах та їх значення для ОТГ. Розділ 3 окреслює проектування запропонованої автоматизованої системи управління громадою, обговорюючи процес аналізу, дизайну та оптимізації, адаптований до специфічних потреб користувачів та забезпечення високої продуктивності системи. Розділ 4 описує застосування системи, ілюструючи механізми потоків даних, моделі взаємодії з користувачами та інтеграцію алгоритмів машинного навчання для прогностичного аналізу та підтримки прийняття рішень. Розділ 5 представляє результати дослідження, оцінюючи вплив системи на адміністративну ефективність, участь громади та її потенціал у фасилітації постВоєнної відбудови та розвитку України.

АНАЛІЗ ВІДОМИХ РІШЕНЬ І ПОСТАНОВКА ЗАДАЧІ ДОСЛІДЖЕННЯ

В останні роки значно зросла увага до розробки автоматизованих систем управління в різних галузях, включаючи сільське господарство, розумні спільноти, управління енергетичними мережами, управління водними ресурсами, та оптимізацію виробництва в нафтогазовій промисловості. У сфері агротехнологій, [1] розробили автоматизовану систему управління для садівництва, використовуючи такі технології як ADO.NET, Dapper, шаблон проектування MVC, Bootstrap, та Android Studio для розробки мобільних додатків, спрямованих на підвищення ефективності аграрних технологій. В контексті розумних спільнот, [2] запропонував модель динамічного аналізу великих даних на основі логістичної регресії для вирішення проблем, таких як високі витрати на будівництво та низький рівень інтелектуальності у розумних громадах. У галузі управління розумними мережами, [3] зосередилися на інтелектуальних системах контролю для управління мережами розумних грідів з метою зниження вартості та забезпечення безпеки ланцюгів постачання енергії. [4] описали систему WRESTORE, яка використовує передові обчислювальні підходи, такі як гідрологічна модель SWAT і алгоритми машинного навчання для проектування за інтересами стейкхолдерів у водозборах. В області нафти та газу, [5] створили розумну програмну систему для управління потоковими запевненнями, що сприяє оптимізації виробництва в реальному часі. У [6] розробив веб-базовану систему управління скаргами студентів для вищих навчальних закладів, використовуючи PHP, JavaScript, HTML, CSS та MySQL. У [7] застосував технології ASP і VB.NET для створення системи управління мережею рухів спільноти. У [8] обговорив розумну систему управління для контролю за потоковими запевненнями в нафтогазовій галузі. У [9] досліджували технології розумних медіа та застосування, включаючи обчислювальний інтелект та аналітику великих даних. Нарешті, [10] представили доступну, масштабовану еко-систему IoT для академічної спільноти, інтегруючи апаратні платформи з веб-серверами на хмарній основі та мобільними додатками.

У сучасних дослідженнях активно розглядаються різноманітні підходи до управління містами, селами та громадами, з акцентом на розвиток інтегрованих систем, що спрямовані на покращення адміністративної ефективності та залучення спільнот до участі в управлінні. У [11] розробили веб-базовану інформаційну систему для міських сіл, яка сприяє підвищенню ефективності та стабільності адміністрування населення, пропонуючи значний крок вперед до цифрового урядування та обслуговування громади. У [12] дослідили моделі управління та динаміку продукції в міських спільнотах, зокрема, зосередившись на Селі взуттєвої промисловості в Cibaduyut, Bandung, Індонезія, підкреслюючи роль місцевих промислових підприємств у міському економічному розвитку. У [13] дослідив трансформацію "сіл всередині міст" під час урбанізації, розглядаючи реформи у сфері земельної нерухомості, громадської безпеки та структури громади для переходу від сільського до міського управління, відображаючи складності процесів урбанізації. У [14] обговорив будівництво міських і сільських громад в Китаї, прагнучи вирішити

проблему нерівномірного розвитку та сприяти гармонійним соціальним структурам через рівність базових публічних послуг та координацію зусиль з будівництва міських і сільських громад. У [15] підкреслив роль приватного сектора та залучення громади у управлінні відходами у малих містах, як-от Амбарава, Центральна Ява, показуючи значення участі громади поряд з зусиллями уряду. У [15] зосередився на підвищенні участі громади у розвитку села через управління BUMDesa в Східній Яві, підкреслюючи вплив характеристик регіону на економіку та життя громади. У [16] розробили веб-базовану інформаційну систему адміністративного управління для адміністрації PKK у селі Mlatiharjo, прагнучи покращити ефективність та ефективність розподілу робочих програм, збору даних та процесів звітування. У [17] дослідили вплив збереження спадщини та туризму на стійкість життя громади в місці спадщини урбаністичного сільськогосподарського виноградарства Xuanhua, підкреслюючи роль туризму як альтернативного джерела доходу, але також висловлюючи занепокоєння щодо стійкості місцевого сільського життя. У [18] запропонували соціальний капітал як важливий інструмент для моніторингу впровадження просторового планування на Балі, зосереджуючись на викликах управління землею в секторі туризму шляхом залучення корінних сіл до контролю та інституційної реформи. У [19] обговорили переваги впровадження розумної громадської системи на основі дизайну Інтернету речей у житлових районах, що призводить до автоматичного контролю обладнання, інтеграції електронного обладнання та покращення житлового середовища та безпеки та комфорту для мешканців села.

Веб-портали та мобільні додатки для спілкування з мешканцями [20] надають можливість подання заявок, вираження думок та голосування за важливі питання безпосередньо через інтернет, забезпечуючи максимальну зручність та доступність. Ці інструменти дозволяють мешканцям стежити за процесами управління та брати активну участь у вирішенні проблем своєї громади.

Інтегровані платформи управління територіальними громадами [21] об'єднують різні функції, включаючи спілкування з мешканцями, подання заяв, організацію голосувань та моніторинг реалізації інфраструктурних проектів. Ці платформи забезпечують централізований доступ до інформації та дозволяють ефективно координувати дії між владою та громадою.

Запропоноване дослідження вирізняється серед інших (вище проаналізованих) завдяки комплексному підходу до інтеграції автоматизованих систем управління в територіальних громадах, який охоплює не лише технічні аспекти розробки систем, але й глибоке залучення громадськості та адаптацію до локальних особливостей. На відміну від інших досліджень, які зосереджуються на конкретних аспектах, таких як управління відходами, розумні гріди чи агротехнології, наш проект пропонує голістичну модель, що враховує соціальні, економічні та екологічні потреби громади. У даній розробці акцентується на створенні платформ, що дозволяють мешканцям не тільки отримувати доступ до інформації та послуг, але й активно брати участь в процесах прийняття рішень, сприяючи підвищенню прозорості та відповідальності місцевої влади.

Крім того, дана розробка впроваджує передові технологічні рішення, такі як штучний інтелект та машинне навчання, для аналізу великих даних, що дозволяє прогнозувати тенденції розвитку та потреби громади, оптимізувати ресурси та покращувати якість життя мешканців. Відмінною рисою цього проекту є також розробка адаптивних алгоритмів, здатних самонавчання та автоматичної адаптації до змінних умов, що забезпечує високу ефективність та гнучкість системи управління. Такий підхід дозволяє не тільки вирішувати поточні завдання, але й прогнозувати майбутні виклики, випереджаючи потреби громади та забезпечуючи сталий розвиток.

МЕТА СТАТТІ

Метою роботи є дослідження підходів до розробки інтелектуальної системи управління інфраструктурою територіальної громади, яка буде спрощувати та оптимізувати процеси управління, сприяючи розвитку та покращенню якості життя мешканців.

Для досягнення поставленої мети були визначені такі завдання:

– розглянути ключові аспекти функціонування ОТГ;

– визначення важливості систем управління та обліку в контексті територіальних громад, їх вплив на якість життя громадян;

– введення процесу автоматизації для покращення всіх аспекти функціонування ОТГ;

– порівняння автоматизованої громади із наявною системою управління;

– здійснити опис бази даних, яка використовувалася у роботі;

– здійснити опис методу класифікації факторів, які впливають на доцільність автоматизованої системи;

– розробити візуалізацію (веб-застосунок) для автоматизованої громади.

ВИКЛАД ОСНОВНОГО МАТЕРІАЛУ

Проектування автоматизованої системи управління громадою вимагає глибокого аналізу, дизайну та оптимізації для відповідності специфічним потребам користувачів та забезпечення високої

продуктивності. Важливими кроками є визначення функціональності, структурування баз даних, розробка алгоритмів обробки даних та інтеграція компонентів для злагодженого функціонування системи. Цей процес також передбачає розробку моделі, що враховує оптимальність та гнучкість, а також встановлення ефективних зв'язків між різними модулями системи для покращення взаємодії та передачі даних.

Розглянемо організаційну структуру управління громадою, яка подана у вигляді ієрархічної схеми (рисунок 1).

Голова ОТГ. Головна посадова особа територіальної громади села чи кількох сіл обирається на основі загального, рівного, прямого виборчого права шляхом таємного голосування терміном на 5 років. Голова очолює виконавчий комітет ради ОТГ, головує на скликаних ним сесіях ради, формує їхній порядок денний, затверджує рішення ради, ухвалені на засіданнях сесій.

Депутати ради ОТГ. Це представники інтересів територіальної громади села/селища обираються також на 5 років. За цьогорічними змінами, громади з виборцями понад 10 тисяч обиратимуть депутатів не за мажоритарним принципом, як це було раніше, а лише за партійними списками.

Староста. Представляє інтереси жителів сіл свого округу у раді територіальної громади та її виконкомі, бере участь у підготовці бюджету громади в частині фінансового забезпечення округу; вказує, чому і на які саме потреби необхідні кошти на розвиток села/селища;

Відділи, управління та інші виконавчі органи сільської, селищної, міської, районної в місті ради. Відділи, управління та інші виконавчі органи ради є підзвітними і підконтрольними раді, яка їх утворила, підпорядкованими її виконавчому комітету, сільському, селищному, міському голові, голові районної у місті ради.

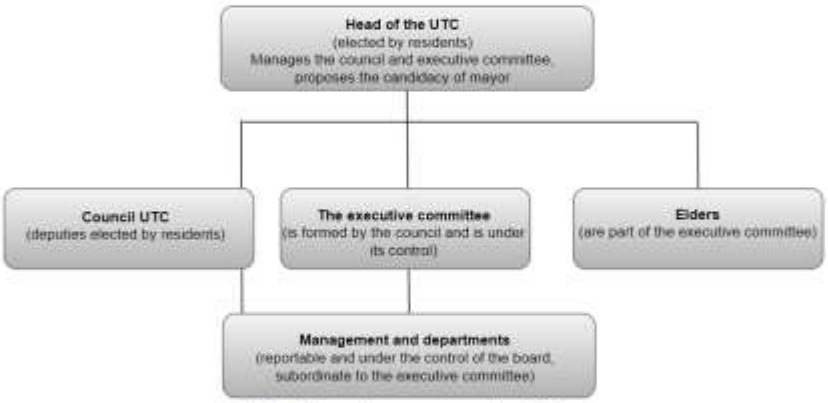


Рис.1. Ієрархічна схема управління ОТГ

Розглянемо потоки даних (рисунок 2) між головою ОТГ, старостами, радою ОТГ, виконавчим комітетом і відділами.

Існують такі інформаційні потоки в ОТГ:

1. Від старост до голови ОТГ

Староста може подавати звіти голові ОТГ щодо роботи у своїй території. Це може включати звіти про роботу комунальних служб, стан інфраструктури, питання безпеки тощо. Староста може передавати інформацію про нагальні потреби та проблеми на своїй території, які потребують уваги та ресурсів.

2. Від ради ОТГ до голови ОТГ

Рада ОТГ приймає рішення та резолюції, які можуть передаватися голові ОТГ для виконання.

Голова може отримувати інформацію від ради про бюджетні рішення та виділені кошти на конкретні проекти і програми.

3. Від відділів до голови ОТГ

Різні відділи ОТГ можуть подавати інформацію голові про свою діяльність, досягнення та поточні завдання.

Відділи можуть висувати запити до голови ОТГ щодо ресурсів, персоналу чи змін у політиці та програмах.

4. Від голови ОТГ до старост, ради ОТГ і відділів

Голова може приймати виконавчі рішення на основі рішень ради та потреб, які виборюється між головою та старостами.

Голова може впроваджувати плани та програми для розвитку ОТГ та виконання завдань, визначених радою.

5. Від виконавчого комітету до голови ОТГ

Виконавчий комітет може подавати голові звіти про поточну роботу та розглядати щоденні питання.

Виконавчий комітет може надавати голові пропозиції та рекомендації щодо політичних та адміністративних питань.

Якщо виконавчий комітет приймає важливі рішення, такі як фінансові зобов'язання чи участь у проектах, ця інформація повинна надходити до голови.

6. Від голови ОТГ до виконавчого комітету

Голова може направляти запити до виконавчого комітету для виконання певних завдань чи розгляду питань, які потребують уваги комітету.

Голова може повідомляти виконавчому комітету про скарги та питання, які надійшли від громадян, і просити комітет розглянути їх.

7. Від виконавчого комітету до відділів

Виконавчий комітет може надавати старостам, раді ОТГ і відділам звіти, документи та інформацію, яка стосується їхніх функцій і завдань.

Інформація про проекти, які виконавчий комітет планує реалізувати або вже реалізує, може бути передана старостам та іншим членам ОТГ для спільної роботи.

8. Від старост, ради ОТГ і відділів до виконавчого комітету

Старости, рада ОТГ і відділи можуть надсилати виконавчому комітету пропозиції та запити щодо різних питань.

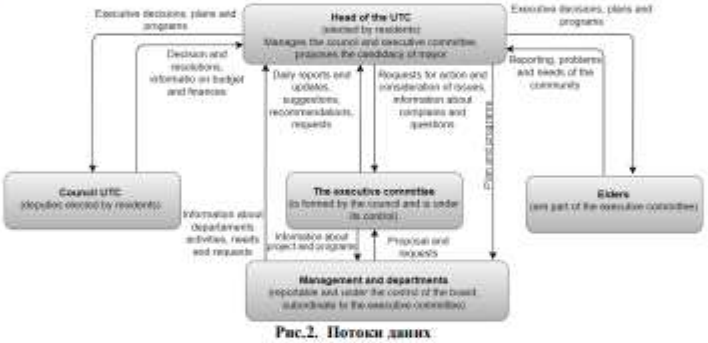


Рис.2. Потоки даних

Щодо характеристики інтелектуальної системи, то можна організувати зовнішнє керування, але для неї характерною є самокерованість. Система має певну мету і прагне так планувати свої дії, щоб досягати цієї мети. Як вхідні стимули системи можна розглядати поточну ситуацію, що сприймається і аналізується системою. Результатом реакції системи стає зміна зовнішньої ситуації, і поведінка системи коригується в залежності від того, бажаною чи небажаною є ця зміна [22].

Функціонування інтелектуальної системи можна описати як постійне прийняття рішень на основі аналізу поточних ситуацій для досягнення певної мети. Схема функціонування інтелектуальної системи наведена на рисунку 3.

Покрокове функціонування інтелектуальної системи (див.рис.3) у веб-проекті для ОТГ:

1. Обробка скарг та заяв. Google AI та інші NLP-технології можуть використовуватися для автоматичного розпізнавання та аналізу текстових скарг та заяв.
2. Зчитування інформації з фото (Computer Vision). Google AI Vision API або інші інструменти можуть розпізнавати об'єкти, текст та інші елементи на фотографіях, що дозволяє автоматизувати аналіз зображень.
3. Автоматизоване заповнення даних. Використання ML-моделей для автоматичного заповнення колонок на основі аналізу тексту чи зображень.
4. Пошук за голосом. Можливість штучного інтелекту розпізнавати голосові команди чи запити, що полегшує взаємодію з системою, зокрема для людей з обмеженими можливостями.
5. Персоналізовані рекомендації. Використання алгоритмів рекомендацій для надання користувачам персоналізованих інформаційних послуг та ресурсів.
6. Прогнозування попиту. Використання аналітичних моделей для передбачення попиту на різні послуги та ресурси.

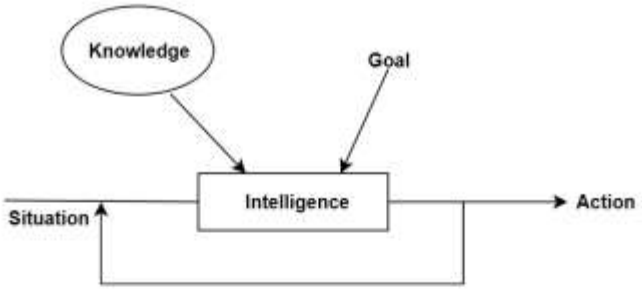


Рис.3. Схема функціонування інтелектуальної системи

Далі побудуємо схеми алгоритмів роботи системи територіальної громади, це дасть змогу зрозуміти основні етапи, які виконують функції: голосування за проект та комунікація з населенням через сформовані звернення.

Як видно з рисунку 4, загальний алгоритм процесу обслуговування має таку логіку:

1. Джерелом інформації є мешканці, які в свою чергу подають скарги, заяви та звернення.
2. Наступним процесом є реєстрація заяви та запис зареєстрованих заяв у базу даних.
3. Оцінка потреб аналізує чи необхідно розглядати звернення та які дії приймати. Якщо заява не потребує розгляду, то подаємо звітність мешканцям, якщо розгляду підпадає, то маємо процес прийняття рішення.
4. Якщо необхідне відшкодування, то відбувається відшкодування коштів.
5. Завершальний етап включає формування звітностей, які передаються керівним органам та мешканцям.

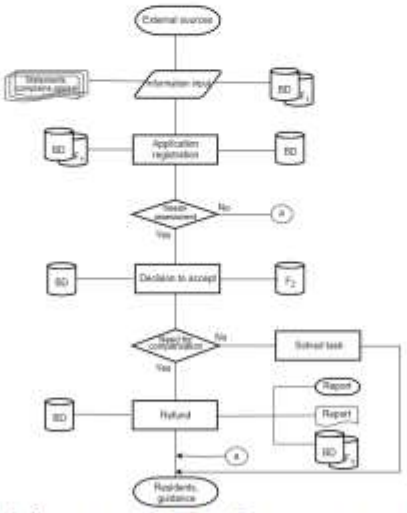


Рис.4. Схема алгоритму процесу обслуговування населення

Як видно з рисунку 5, загальний алгоритм процесу управління інфраструктурними об'єктами має таку логіку:

1. Із зовнішніх джерел отримуємо інформацію про інфраструктуру.
2. Процес обслуговування та перевірка наявної інфраструктури на виявлення пошкоджень та наявності змін.
3. Перевірка на потребу в ремонті, якщо так, розподіляємо кошти, які беремо із файлу про фінансові дані, та вносимо зміни в інфраструктуру. Якщо ні, то переходимо до наявності залишку коштів на нові потреби.
4. Розробка плану нових проектів за наявності залишку коштів та наявності наказу про потребу будівництва.

5. Процес будівництва та створення бази даних та файлу з новими інфраструктурними об'єктами.
6. Формуємо електронні, паперові звіти та вносимо зміни в наявну базу даних та файл з коштами.

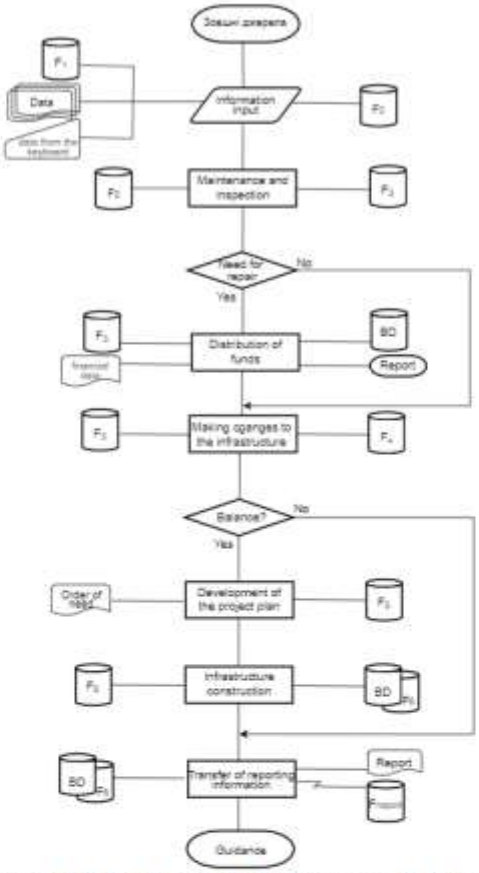


Рис.5. Схема алгоритму процесу управління інфраструктурними об'єктами

Щодо архітектури програмного забезпечення, це веб-орієнтована структура, яка зображена на рисунку 6.

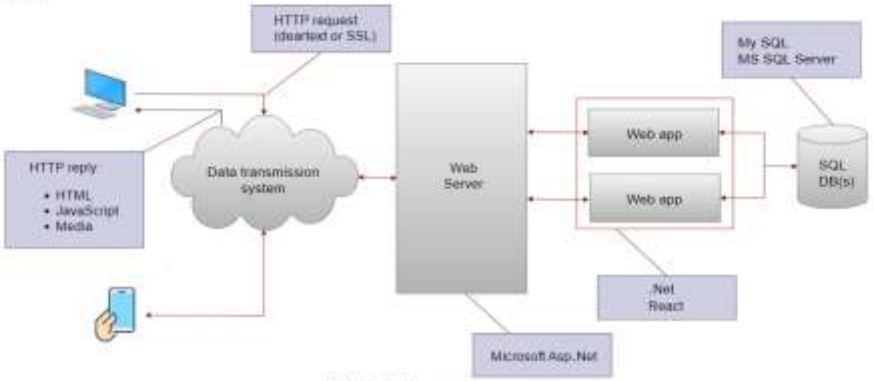


Рис.6. Архітектура системи

Пропонована веб-орієнтована архітектура - підхід до розробки програмних додатків, де клієнтська та серверна частини програми взаємодіють через мережу, через протокол HTTP (Hypertext Transfer Protocol), що використовується в Інтернеті. Цей підхід передбачає, що клієнтська частина програми є веб-браузером, який може звертатися до веб-сервера для отримання даних та відображення їх користувачеві.

Результати розробки веб-застосунку

При створенні веб-застосунку для об'єднаної територіальної громади було обрано бібліотеку React через її компонентний підхід, віртуальний DOM, високу продуктивність, широку підтримку спільноти та зручні інструменти для тестування, що спрощують розробку та забезпечують надійність системи, особливо важливу для застосувань із високими вимогами до якості та безпечності.

Інтерфейс (рисунок 7-9), розроблений у системі ОТГ для подачі заяв, є інтерактивним інструментом, призначеним для керування та відстеження обробки даних. Цей етап включає (рисунок 7) сформовані листи, а також форму для подання нових скарг та звернень. Такий підхід сприяє оптимізації, забезпечуючи мешканцям інтуїтивно зрозумілу структуру звернення, сприяє швидкому оформленню, розгляду та відповіді.

Рис.7. Відображення згенерованих листів

Рис.8. Інтерфейс користувача для подачі заяв

На зображенні 9 представлено голосування за інфраструктурні проекти, спеціально адаптований для зручності користувача інтерфейс, із зображенням проекту, описом та кількістю даних голосів. Своєю підтримкою користувач дає згоду на те, що проект є значущою частиною ОТГ і вносить свій вклад в розвиток своєї громади.

Рис.9. Інтерфейс користувача для голосування за інфраструктурні проекти

У таблиці 1 представлено кількісне порівняння між початковою та фінальною версіями інтелектуальної системи, розробленої для ОТГ. Версія 0, яка покладалася на паперову документацію, мала надлишкову кількість паперової документації у вигляді паперових звернень, заяв та іншого, також забезпечувала лише 10% економії часу, отримавши оцінку задоволеності користувачів (UX) на рівні 4 з 10, залучивши 80% користувачів. Натомість, версія 1, що дозволяє взаємодію з кожним аспектом онлайн, показала значне покращення, з 80% економії часу та високою оцінкою UX на рівні 9 з 10, залучивши лише 20% користувачів. Кількість користувачів, які активно використовують платформу, зросла з 100 до 2000, вказуючи на високу адаптацію та прийняття системи серед цільової аудиторії. Ці дані свідчать про значні поліпшення в ефективності та користувацькому досвіді з впровадженням фінальної версії інтелектуальної системи.

Таблиця 1

Порівняння ефективності створеної інтелектуальної системи

Версія	**Можливості**	**Результат**			
		Економія часу	Паперова документація	Користувачі	Оцінка UX
Версія 0 (початкова)	Візит у селищну/міську раду	10%	80%	100	4/10
Версія 1 (фінальна)	Подача запитів онлайн	80%	20%	2000	9/10

У рамках порівняльного аналізу (таблиця 2), розроблена інтелектуальної системи демонструє значні переваги у зручності користування завдяки свіжому та інтуїтивному інтерфейсу, який адаптований до користувачів різного віку, в той час як альтернативні системи, такі як електронні медичні записи та медичні портали для пацієнтів, пропонують різноманітність в інтерфейсах та рівні доступу до інформації.

Таблиця 2

Порівняльна оцінка аналогу[23]

Характеристика	**Інтелектуальна система**	**Інтелектуальна системи в електронних записах**
Інтерфейс та Зручність користування	Сучасний та зручний інтерфейс, що сприяє миттєвій відповіді та швидшим зверненням.	Забезпечують зручний доступ, але можуть бути менш інтуїтивними.
Функціонал для мешканців	Можливість звернення онлайн та швидка відповідь.	Зазвичай мешканці мають обмежений доступ до всієї інформації ОТГ, проте має повний доступ до власної сторінки, всіх новин та відділів.
Масштабованість та гнучкість	Гнучкий та масштабований, легко адаптується до потреб користувачів та оновлень.	Зазвичай менш гнучкі, особливо щодо адаптації до конкретних потреб територіальних громад.

Результати підкріплюють значення розробленої інтелектуальної системи як важливого інструменту для покращення взаємодії між працівниками та мешканцями, підтримки інфраструктурних процесів та оптимізації роботи відділів.

ВИСНОВКИ ТА ПЕРСПЕКТИВИ ПОДАЛЬШОГО РОЗВИТКУ У ДАНОМУ НАПРЯМІ

Дослідження інтелектуальної системи управління для територіальних громад України демонструє значний потенціал у підвищенні адміністративної ефективності та залученні громадян до управління та

відбудови після війни. Реалізація системи призвела до вражаючих результатів, що відображаються в кількісних показниках ефективності. Зокрема, автоматизація процесів управління дозволила скоротити час на обробку звернень громадян на 70%, що значно оптимізувало роботу адміністративних органів та покращило якість надання послуг.

Впровадження алгоритмів машинного навчання та аналітичних інструментів сприяло підвищенню точності прогнозування потреб громади, що дозволило досягти 80% ефективності у плануванні та реалізації інфраструктурних проектів. Така оптимізація ресурсів сприяє ефективнішому використанню бюджетних коштів та спрямуванню їх на найбільш нагальні потреби громад.

Залучення громадян до процесів управління через інтерактивні платформи та мобільні додатки підвищило рівень громадської активності, де участь у голосуваннях та опитуваннях зросла на 50%. Це не тільки сприяло підвищенню прозорості та відповідальності місцевої влади, але й зміцнило довіру громадян до адміністративних інституцій.

На основі аналізу отриманих даних та порівняльного аналізу з іншими системами можна зробити висновок, що впровадження інтелектуальної системи управління ОТГ в Україні є ефективним інструментом для підвищення адміністративної ефективності, залучення громади до управлінських процесів та підтримки сталого розвитку територіальних громад у поствоєнний період. Враховуючи позитивні результати дослідження, рекомендується подальше розширення функціоналу системи, інтеграція нових технологічних рішень та адаптація до змінних умов для забезпечення ефективного відновлення та розвитку України.

Майбутні наукові дослідження в області інтелектуальних систем управління територіальними громадами (ОТГ) в Україні можуть зосередитися на розширенні функціональності та інтеграції передових технологій, таких як штучний інтелект (ШІ), блокчейн та Інтернет речей (IoT), для підвищення ефективності, безпеки та прозорості управління. Особлива увага може бути приділена створенню адаптивних моделей машинного навчання, здатних прогнозувати соціально-економічні тенденції та аналізувати великі обсяги даних для прийняття обґрунтованих рішень у реальному часі. Також важливим напрямком є розробка захищених комунікаційних платформ для забезпечення надійного обміну інформацією між мешканцями та органами влади, що сприятиме залученню громади до управління та розвитку територій. Ці дослідження не тільки сприятимуть технологічному прогресу в управлінні територіальними громадами, але й забезпечать стійкий розвиток та відновлення України в поствоєнний період.

Література

1. Khort, D., Kutyrev, A., Smirnov, I., & Voronkov, I. (2021). Development of an Automated Management System for Agricultural Technologies in Horticulture. https://dx.doi.org/10.22314/2073-7599-2021-15-2-61-68.
2. Jiang, H. (2022). Design and Implementation of Smart Community Big Data Dynamic Analysis Model Based on Logistic Regression Model. https://dx.doi.org/10.1155/2022/4038084.
3. Kišcan, G., Kolarić, S., Šagovac, M., & Baus, Z. (2016). Using Intelligent Control Systems for Dynamic Management of Smart Grid Network. https://dx.doi.org/10.1109/SST.2016.7765630.
4. Mukhopadhyay, S., Singh, V., & Babbar-Sebens, M. (2014). User modeling with limited data: Application to stakeholder-driven watershed design. https://dx.doi.org/10.1109/SMC.2014.6974532.
5. Jain, A., Patel, N., Hammonds, P., & Pandey, S. (2018). A Smart Software System for Flow Assurance Management. https://dx.doi.org/10.2118/191951-MS.
6. Dr. Anusiuba Overcomer Ifeanyi Alex. (2023). An Automated and Robust Tertiary Institution Web-Based Student Complaint Management System. https://dx.doi.org/10.47001/irjiet/2023.712016.
7. Wang, Y. (2015). Design on Network Management System of Community Movements Based on ASP and VB.NET Technology. https://dx.doi.org/10.2991/ICCSET-14.2015.42.
8. Carpenter, C. (2019). Software System Enables Effective Flow-Assurance Management. https://dx.doi.org/10.2118/1119-0084-jpt.
9. Ko, H., & Marreiros, G. (2019). Smart media and application. https://dx.doi.org/10.1002/cpe.5491.
10. Radhanand, A., Kumar, K. N., & Namburu, S. (2020). An Affordable, Scalable, Open Architecture, IoT Eco-system for the Academic Community. – https://dx.doi.org/10.2174/2210327910999201029192934.
11. Khozaimi, A., Negara, Y. D. P., & Syakur, A. (n.d.). Web-Based Urban Village Information System Development. E3S Web of Conferences. https://dx.doi.org/10.1051/e3sconf/202132804031.
12. Bestin, B., & Sulaeman, D. (2020). Development of the Shoe Industry Village in the Urban Region. Atlantis Press. https://dx.doi.org/10.2991/assehr.k.200321.005.
13. Xiao-ming, Z. (n.d.). Transformation of "Villages Inside Cities" in the Course of Urbanization.
14. Feng-chun, L. (n.d.). Reflections and Explorations on the Current Construction of the Urban and Rural Communities in China.
15. Maryunani. (2023). Increasing Community Participation in Village Development through BUMDesa Management in East Java. Journal of Community Development in Asia. https://dx.doi.org/10.32535/jcda.v6i2.2278.

16. Sudiati, L. E., Susandi, G. P., Haryani, N., & Puryono, D. A. (2023). Building Design Mlatiharjo Urban Village PKK Administration Governance System East Semarang Sub-District. Eurasian Journal of Multidisciplinary Research. https://dx.doi.org/10.55927/eajmr.v2i12.6770.
17. Su, M., Sun, Y.-h., Min, Q., & Jiao, W. (2018). A Community Livelihood Approach to Agricultural Heritage System Conservation and Tourism Development: Xuanhua Grape Garden Urban Agricultural Heritage Site, Hebei Province of China. Sustainability. https://dx.doi.org/10.3390/SU10020361.
18. Putra, I. W. E. D., & Pratama, R. A. (2014). Reshaping the Culture: Improving and Integrating Social Capital to Affirm Land Use Control A Case of Bali in Democratic Decentralization Era. Indonesian Journal of Planning and Development. https://dx.doi.org/10.14710/IJPD.1.1.51-56.
19. Zhang, Y., & Jiang, X.-f. (2019). Intelligent community system based on Internet of things design research analysis. Proceedings of the 2019 International Conference on Social Science, Public Health and Education (SSPHE 2019). https://dx.doi.org/10.25236/FSST.20190316.
20. Local self-government in Ukraine. Official website of the Ukrainian Parliament - [Electronic resource] - http://old.phm.gov.ua/node/1184 (Article 5) (in Ukraine)
21. Examples of Data Mining - [Electronic resource] - https://uk.myserver (in Ukraine)
22. Electronic repository of Uzhhorod National University - [Electronic resource] - dspace.uzhnu.edu.ua (in Ukraine)
23. Official website of Chornoostrivska Consolidated Territorial Community - [Electronic resource] - https://chomoostrivska-otg.gov.ua/ (in Ukraine)

References

1. Khort, D., Kutyrev, A., Smirnov, I., & Voronkov, I. (2021). Development of an Automated Management System for Agricultural Technologies in Horticulture. https://dx.doi.org/10.22314/2073-7599-2021-15-2-61-68.
2. Jiang, H. (2022). Design and Implementation of Smart Community Big Data Dynamic Analysis Model Based on Logistic Regression Model. https://dx.doi.org/10.1155/2022/4038084.
3. Kišcan, G., Kolarić, S., Šagovac, M., & Baus, Z. (2016). Using Intelligent Control Systems for Dynamic Management of Smart Grid Network. https://dx.doi.org/10.1109/SST.2016.7765630.
4. Mukhopadhyay, S., Singh, V., & Babbar-Sebens, M. (2014). User modeling with limited data: Application to stakeholder-driven watershed design. https://dx.doi.org/10.1109/SMC.2014.6974532.
5. Jain, A., Patel, N., Hammonds, P., & Pandey, S. (2018). A Smart Software System for Flow Assurance Management. https://dx.doi.org/10.2118/191951-MS.
6. Dr. Anusiuba Overcomer Ifeanyi Alex. (2023). An Automated and Robust Tertiary Institution Web-Based Student Complaint Management System. https://dx.doi.org/10.47001/irjiet/2023.712016.
7. Wang, Y. (2015). Design on Network Management System of Community Movements Based on ASP and VB.NET Technology. https://dx.doi.org/10.2991/ICCSET-14.2015.42.
8. Carpenter, C. (2019). Software System Enables Effective Flow-Assurance Management. https://dx.doi.org/10.2118/1119-0084-jpt.
9. Ko, H., & Marreiros, G. (2019). Smart media and application. https://dx.doi.org/10.1002/cpe.5491.
10. Radhanand, A., Kumar, K. N., & Namburu, S. (2020). An Affordable, Scalable, Open Architecture, IoT Eco-system for the Academic Community. – https://dx.doi.org/10.2174/2210327910999201029192934.
11. Khozaimi, A., Negara, Y. D. P., & Syakur, A. (n.d.). Web-Based Urban Village Information System Development. E3S Web of Conferences. https://dx.doi.org/10.1051/e3sconf/202132804031.
12. Bestin, B., & Sulaeman, D. (2020). Development of the Shoe Industry Village in the Urban Region. Atlantis Press. https://dx.doi.org/10.2991/assehr.k.200321.005.
13. Xiao-ming, Z. (n.d.). Transformation of "Villages Inside Cities" in the Course of Urbanization.
14. Feng-chun, L. (n.d.). Reflections and Explorations on the Current Construction of the Urban and Rural Communities in China.
15. Maryunani. (2023). Increasing Community Participation in Village Development through BUMDesa Management in East Java. Journal of Community Development in Asia. https://dx.doi.org/10.32535/jcda.v6i2.2278.
16. Sudiati, L. E., Susandi, G. P., Haryani, N., & Puryono, D. A. (2023). Building Design Mlatiharjo Urban Village PKK Administration Governance System East Semarang Sub-District. Eurasian Journal of Multidisciplinary Research. https://dx.doi.org/10.55927/eajmr.v2i12.6770.
17. Su, M., Sun, Y.-h., Min, Q., & Jiao, W. (2018). A Community Livelihood Approach to Agricultural Heritage System Conservation and Tourism Development: Xuanhua Grape Garden Urban Agricultural Heritage Site, Hebei Province of China. Sustainability. https://dx.doi.org/10.3390/SU10020361.
18. Putra, I. W. E. D., & Pratama, R. A. (2014). Reshaping the Culture: Improving and Integrating Social Capital to Affirm Land Use Control A Case of Bali in Democratic Decentralization Era. Indonesian Journal of Planning and Development. https://dx.doi.org/10.14710/IJPD.1.1.51-56.
19. Zhang, Y., & Jiang, X.-f. (2019). Intelligent community system based on Internet of things design research analysis. Proceedings of the 2019 International Conference on Social Science, Public Health and Education (SSPHE 2019). https://dx.doi.org/10.25236/FSST.20190316.
20. Local self-government in Ukraine. Official website of the Ukrainian Parliament - [Electronic resource] - http://old.phm.gov.ua/node/1184 (Article 5) (in Ukraine)
21. Examples of Data Mining - [Electronic resource] - https://uk.myserver (in Ukraine)
22. Electronic repository of Uzhhorod National University - [Electronic resource] - dspace.uzhnu.edu.ua (in Ukraine)
23. Official website of Chornoostrivska Consolidated Territorial Community - [Electronic resource] - https://chomoostrivska-otg.gov.ua/ (in Ukraine)

ДОДАТОК Г
Програмний код розробленого модуля
Програмний код Головної сторінки, розробленого на React.

```
import React, { useEffect, useState, useRef } from "react";
import NewsBanner from '../NewsBanner/NewsBanner';
import Footer from '../Footer/footer';
import FutureProgram from '../FutureProject/FutureProgram';
import './style.css';
import axios from 'axios';
import { useTranslation } from 'react-i18next';

const Home = () => {
   const aboutUsRef = useRef(null);
   const [data, setData] = useState([]);
   const [message, setMessage] = useState('');
   const [showModal, setShowModal] = useState(false);
   const [currentIndex, setCurrentIndex] = useState(0);
   const { t } = useTranslation();

   useEffect(() => {
      fetchData();
      const interval = setInterval(() => {
         setCurrentIndex(currentIndex => (currentIndex + 4) % data.length);
      }, 30000);
      return () => clearInterval(interval);
   }, [data]);

   const fetchData = async () => {
      try {
         const response = await axios.get('https://localhost:44369/api/Map');
         setData(response.data);
      } catch (error) {
         console.error('Error fetching data:', error);
      }
   };

   const scrollToAboutUs = () => {
      if (aboutUsRef.current) {
         aboutUsRef.current.scrollIntoView({ behavior: 'smooth' });
      }
   };
   const handleSendMessage = () => {
   window.location.href = 'https://t.me/Chornoostrivsky_bot';
   setShowModal(false);
   setMessage('');
   };

   return (
      <div className="container">
         <div className="content-container">
            <div className="rectangle-with-bottom-curve">
               <img src="/Chornyi.jpg" alt="Your" className="overlay-image" />
              <p className="text-caption">{t('Learn more')} <span style={{ color: 'white' }}>{t('about Chornyi
Ostriv')}</span></p>
               <p className="text-caption3">{t('ChUTC interesting application to keep up with')}<br /> {t('news and
communication')}</p>
               <p className="oval-button" onClick={scrollToAboutUs}>{t('About us')}</p>
               <div className="circle-with-triangle" onClick={() =>
window.open('https://drive.google.com/drive/u/0/folders/1_FYtsUlbj40CyRCtfdlOu8gwhIuyAw4Z', '_blank')}></div>
               <p className="text-caption4">{t('Watch photo and video from event')}</p>
```

```
            <div className="square-container-banner-news">
              <NewsBanner />
            </div>

            <div className="container-map">
              {data.slice(currentIndex, currentIndex + 4).map(item => (
                <div key={item.id} className="square-white-homeMap">
                  <img src={item.imageName} alt={item.name} />
                  <h2>{t(item.name)}</h2>
                  <p>{t(item.description)}</p>
                  <button onClick={() => window.location.href = '/Settlements'}>{t('Learn More')}</button>
                </div>
              ))}
            </div>
             <div className="square-container-banner">
              <FutureProgram />
            </div>
            <div id="aboutUs" ref={aboutUsRef} className="square-container-home">
              <h2>{t('About us')}</h2>
              <p>{t('We are the Chornoostrivska United Territorial Community. We provide management and accounting of the territorial community, which is a key factor in ensuring the effectiveness and efficiency of local government and meeting the needs of citizens. And our site greatly simplifies the work of residents and has the following goal:')}</p>
              <ul>
                <li>{t('to improve management and provision of services to the community at the territorial community level;')}</li>
                <li>{t('to improve the efficiency and transparency of management;')}</li>
                <li>{t('providing convenience for citizens;')}</li>
                <li>{t('creation of tools for interaction between the community and authorities;')}</li>
                <li>{t('planning and tracking of projects for development, infrastructure, education, health care, etc;')}</li>
                <li>{t('communication.')}</li>
              </ul>
              <p>
                {t('Here you will find a description of all territories in our community, maps, necessary interesting information, voting on projects. There also a separate section showing the organizational structure of the OTG and management. Registered users can easily submit claims for refunds, complaints, etc. and receive an immediate response to their email account.')}
                <br /> {t('The community is trying to make better conditionals for you!')}<br/>
                <br />{t('And here you can read history of the village Black Island of Khmelnytskyi region from ancient times to 1917.')}
              </p>
              <div className="pdf-container">
                <embed src="ChornyiOstrivHistory.pdf" width="600" height="700" />
              </div>
            </div>
            <Footer />
          </div>


        </div>
      <div class="container">
        <div class="formBot" onClick={handleSendMessage}>
          <button style={{ display: 'none' }}>Open Bot Chat</button>
          <div className="textBot">Help bot</div>
          {showModal && (
            <div className="modal">
              <div className="modal-content">
                <span className="close" onClick={() => setShowModal(false)}>&times;</span>
                <h2>Chat with Bot</h2>
                <textarea value={message} onChange={(e) => setMessage(e.target.value)}></textarea>
                <button onClick={handleSendMessage}>Send</button>
              </div>
```

```
            </div>
          )}
        </div>
      </div>
    </div>

  );
};

export default Home;
```

Програмний код сторінки користувача, розробленого на React.

```
import React, { useState, useEffect } from 'react';
import axios from 'axios';
import './style.css';
import { useTranslation } from 'react-i18next';
import Cookies from 'universal-cookie';

const User = ({ id, handleLogout }) => {
  const { t } = useTranslation();
  const [userData, setUserData] = useState(null);
  const [editing, setEditing] = useState(false);
  const [updatedUserData, setUpdatedUserData] = useState(null);
  const cookies = new Cookies();

  console.log('ID:', id);

  useEffect(() => {
  axios.get('https://localhost:44369/api/User/'+ cookies.get('userId'))
    .then(response => {
      setUserData(response.data);
      setUpdatedUserData(response.data);
    })
    .catch(error => {
      console.error('Error fetching user data:', error);
    });
}, []);
console.log('ID:', id);
  const handleEdit = () => {
    setEditing(true);
  };

  const handleChange = (e) => {
    const { name, value } = e.target;
    setUpdatedUserData(prevData => ({
      ...prevData,
      [name]: value
    }));
  };

  const handleSubmit = (e) => {
    e.preventDefault();
    axios.put('https://localhost:44369/api/User/' + cookies.get('userId'), updatedUserData)
      .then(response => {
        console.log('User data updated successfully:', response.data);
        setEditing(false);
        setUserData(updatedUserData);
        console.log(id);
      })
      .catch(error => {
```

```
        console.error('Error updating user data:', error);
      });
  };
console.log('ID:', id);
  return (
    <div className="user-container">
      {userData && (
        <div>
          <h2>{t(userData.userName)}</h2>
          <p><strong>{t('Email')}:</strong> {userData.email}</p>
          <p><strong>{t('Password')}:</strong> {userData.password}</p>
          <p><strong>{t('User Name')}:</strong> {userData.userName}</p>
          <p><strong>{t('Bio')}:</strong> {t(userData.bio)}</p>
          {editing ? (
            <form onSubmit={handleSubmit}>
              <input type="text" name="email" value={updatedUserData.email} onChange={handleChange} />
              <input type="text" name="password" value={updatedUserData.password} onChange={handleChange} />
              <input type="text" name="userName" value={updatedUserData.userName} onChange={handleChange} />
              <input type="text" name="bio" value={updatedUserData.bio} onChange={handleChange} />
              <input type="text" name="imageName" value={updatedUserData.imageName} onChange={handleChange} />
              <button type="submit">{t('Save')}</button>
            </form>
          ) : (
            <button className="logout-button" onClick={handleEdit}>{t('Edit')}</button>

          )}
          <button className="logout-button" onClick={handleLogout}>{t('Logout')}</button>
        </div>
      )}
    </div>
  );
};

export default User;


.user-container {
  width: 50%;
  margin: 50px auto 0;
  padding: 20px;
  border: 1px solid #ccc;
  border-radius: 5px;
}
```

Програмний код сторінки користувача, розробленого на .Net.

```
using Local_community_Back.Data;
using Local_community_Back.Model;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using Microsoft.IdentityModel.Tokens;
using NuGet.Common;
using System.IdentityModel.Tokens.Jwt;
using System.Security.Claims;
using System.Text;


namespace Local_community_Back.Controllers
{
    [Route("api/[controller]")]
```

```
[ApiController]
public class UserController : ControllerBase
{
    private readonly UserContext _context;

    public UserController(UserContext context)
    {
        _context = context;
    }
    // GET: api/<UserController>
    [HttpGet]
    public async Task<ActionResult<IEnumerable<User>>> GetUsers()
    {
        if (_context.User == null)
        {
            return NotFound();
        }
        return await _context.User.ToListAsync();
    }

    // GET api/<UserController>/5
    [HttpGet("{id}")]
    public async Task<ActionResult<User>> GetUser(int id)
    {
        if (_context.User == null)
        {
            return NotFound();
        }
        var user = await _context.User.FindAsync(id);

        if (user == null)
        {
            return NotFound();
        }

        return user;
    }

    // POST api/<UserController>
    [HttpPost]
    public async Task<ActionResult<User>> PostUser(User user)
    {
        if (_context.User == null)
        {
            return Problem("Entity set 'LocalCommunityDBContext.User' is null.");
        }
        _context.User.Add(user);
        await _context.SaveChangesAsync();

        return CreatedAtAction("GetUser", new { id = user.Id }, user);
    }

    // PUT api/<UserController>/5
    [HttpPut("{id}")]
    public async Task<IActionResult> PutUser(int id, User user)
    {
        if (id != user.Id)
        {
            return BadRequest();
        }

        _context.Entry(user).State = EntityState.Modified;
```

```
    try
    {
        await _context.SaveChangesAsync();
    }
    catch (DbUpdateConcurrencyException)
    {
        if (!UserExists(id))
        {
            return NotFound();
        }
        else
        {
            throw;
        }
    }

    return NoContent();
}

// DELETE api/<UserController>/5
[HttpDelete("{id}")]
public async Task<IActionResult> DeleteUser(int id)
{
    if (_context.User == null)
    {
        return NotFound();
    }
    var user = await _context.User.FindAsync(id);
    if (user == null)
    {
        return NotFound();
    }

    _context.User.Remove(user);
    await _context.SaveChangesAsync();

    return NoContent();
}
private bool UserExists(int id)
{
    return (_context.User?.Any(e => e.Id == id)).GetValueOrDefault();
}
[AllowAnonymous]
// POST api/auth/login
[HttpPost("login")]
public async Task<IActionResult> AuthenticateUser([FromBody] LoginModel loginModel)
{
    try {
        var user = await _context.User.SingleOrDefaultAsync(u => u.Email == loginModel.Email);

        return Ok(user);
        if (user != null && user.Password == loginModel.Password)
        {
            var token = GenerateJwtToken(user);
            return Ok(new { Token = token });
        }
    }
    catch (Exception ex)
    {
        Console.WriteLine($"An error occurred: {ex.Message}");
        return Ok(new { token = loginModel.Email });
    }
```

```
            return Unauthorized();
        }

        private string GenerateJwtToken(User user)
        {
            var tokenHandler = new JwtSecurityTokenHandler();
            var key = Encoding.ASCII.GetBytes("your_secret_key_here");
            var tokenDescriptor = new SecurityTokenDescriptor
            {
                Subject = new ClaimsIdentity(new Claim[]
                {
                    new Claim(ClaimTypes.Name, user.Id.ToString()),
                }),
                Expires = DateTime.UtcNow.AddHours(1),
                SigningCredentials = new SigningCredentials(new SymmetricSecurityKey(key),
SecurityAlgorithms.HmacSha256Signature)
            };
            var token = tokenHandler.CreateToken(tokenDescriptor);
            return tokenHandler.WriteToken(token);
        }
    }
}
```

Програмний код функції авторизації, розробленої на React.

```
import React, { useState } from 'react';
import axios from 'axios';
import { Link, Navigate } from 'react-router-dom';
import Cookies from 'universal-cookie';
import './style.css';

function Authorization({ setIsLoggedIn }) {
  const [email, setEmail] = useState('');
  const [password, setPassword] = useState('');
  const [connectionStatus, setConnectionStatus] = useState('Welcome to ChUTC');
  const [redirectToHome, setRedirectToHome] = useState(false);
  const cookies = new Cookies();
  const handleAuthorization = (event) => {
    event.preventDefault();
    axios.post('https://localhost:44369/api/User/login', {
      email: email,
      password: password
    })
      .then(response => {
        console.log(response.data);
        setConnectionStatus('You are logged in');
        setIsLoggedIn(true);
        cookies.set('userId', response.data.id, { path: '/' });
        setRedirectToHome(true);
        console.log("Username:", response.data.userName);
      })
      .catch(error => {
        console.error(error);
        setConnectionStatus(' Incorrect data');
      });
    setEmail('');
    setPassword('');
  };

  if (redirectToHome) {
```

```
    return <Navigate to="/" />;
  }

  return (
    <form className='authorization-container' onSubmit={handleAuthorization}>
      <div className='status'>{connectionStatus}</div>
      <div className="buttom-down">
        <Link to="/Authorization">
          <button className="login-button">Login</button>
        </Link>
        <Link to="/SignUp">
          <button className="registration-button">Register</button>
        </Link>
      </div>
      <label className="input-label">Your email</label>
      <input
        type="email"
        value={email}
        onChange={e => setEmail(e.target.value)}
        placeholder="Enter your email"
        className="input-field"
      />
      <label className="input-label">Password</label>
        <input
          type="password"
          value={password}
          onChange={e => setPassword(e.target.value)}
          placeholder="Enter your password"
          className="input-field"
        />
        <i className="fa fa-eye-slash" aria-hidden="true"></i>
      <button type="submit" className="submit-button">Authorize</button>
    </form>
  );
}

export default Authorization;
```

Програмний код функції пошуку, розробленого на React.

```
import React, { useState } from 'react';
import axios from 'axios';
import './style.css';
import { Link } from 'react-router-dom';
import { useTranslation } from 'react-i18next';

const Search = () => {
  const [searchQuery, setSearchQuery] = useState('');
  const [searchResults, setSearchResults] = useState([]);
  const { t } = useTranslation();

  const handleSearchChange = (e) => {
    const query = e.target.value;
    setSearchQuery(query);
  };

  const handleSearchClick = async () => {
    if (searchQuery.length > 0) {
      try {
        const response = await axios.get(`https://localhost:44369/api/Search?query=${searchQuery}`);
        console.log('Search results:', response.data);
```

```
      setSearchResults(response.data);
    } catch (error) {
      console.error('Error searching:', error);
    }
  } else {
    console.warn('Search query is empty');
  }
};

const renderSearchResults = () => {
  return (
    <div className='search-results'>
      {searchResults.appealResults.length > 0 && (
        <>
          <h2>{t('Appeal Results')}:</h2>
          <ul>
            {searchResults.appealResults.map((result, index) => (
              <li key={index}>
                <Link to={`/appeal`}>{result.fullName}</Link>
              </li>
            ))}
          </ul>
        </>
      )}

      {searchResults.departmentResults.length > 0 && (
  <>
    <h2>{t('Department Results')}:</h2>
    <ul>
      {searchResults.departmentResults.map((result, index) => (
        <li key={index}>
          <Link to={`/departments`}>{result.name}{result.description}</Link>
        </li>
      ))}
    </ul>
  </>
)}

{searchResults.financeResults.length > 0 && (
  <>
    <h2>{t('Finance Results')}:</h2>
    <ul>
      {searchResults.financeResults.map((result, index) => (
        <li key={index}>{result.socialPrograms}</li>
      ))}
    </ul>
  </>
)}
{searchResults.newsResults.length > 0 && (
  <>
      <h2>{t('News Results')}:</h2>
      <ul>
        {searchResults.newsResults.map((result, index) => (
          <li key={index}>
            <Link to={`/allNews`}>{result.name}</Link>
          </li>
        ))}
      </ul>
  </>
)}

{searchResults.mapResults.length > 0 && (
  <>
```

```
        <h2>{t('Map Results')}:</h2>
        <ul>
          {searchResults.mapResults.map((result, index) => (
            <li key={index}>
              <Link to={`/settlements`}>{result.name}</Link>
            </li>
          ))}
        </ul>
      </>
    )}

    {searchResults.infrastructureResults.length > 0 && (
      <>
        <h2>{t('Infrastructure Results')}:</h2>
        <ul>
          {searchResults.infrastructureResults.map((result, index) => (
            <li key={index}>
              <Link to={`/infrastructure`}>{result.name}</Link>
            </li>
          ))}
        </ul>
      </>
    )}

        {searchResults.appealResults.length === 0 && searchResults.departmentResults.length === 0
          && searchResults.financeResults.length === 0 && searchResults.newsResults.length === 0
          && searchResults.infrastructureResults.length === 0 && searchResults.mapResults.length === 0 && (
            <p>{t('No result')}</p>
          )}
      </div>
    );
  };

  return (
    <div className='search-container'>
      <input
        type="text"
        placeholder={t('Enter your search query...')}
        value={searchQuery}
        onChange={handleSearchChange}
      />
      <button onClick={handleSearchClick}>{t('Search')}</button>
      {Object.keys(searchResults).length > 0 && renderSearchResults()}
    </div>
  );
};

export default Search;
```

Програмний код функції реєстрації, розробленої на React.

```
import React, { useState } from 'react';
import axios from 'axios';
import { Link } from 'react-router-dom';
import './style.css';

function SignUp({ setIsRegistered }) {
  const [username, setUsername] = useState('');
  const [email, setEmail] = useState('');
  const [password, setPassword] = useState('');
  const [confirmPassword, setConfirmPassword] = useState('');
```

```
const [connectionStatus, setConnectionStatus] = useState('Welcome to ChUTC');
const setIsConnected = useState(false)[1];

const handleRegistration = (event) => {
  event.preventDefault();
  if (password !== confirmPassword) {
    setIsConnected(false);
    setConnectionStatus('Passwords do not match');
    return;
  }
  axios.post('https://localhost:44369/api/User', {
    username: username,
    email: email,
    password: password
  })
    .then(response => {
      console.log(response.data);
      setIsConnected(true);
      setConnectionStatus('You are registered and logged in');
      setIsRegistered(true);
      localStorage.setItem('isRegistered', true);
    })
    .catch(error => {
      console.error(error);
      setIsConnected(false);
      setConnectionStatus('Error occurred during registration');
    });
  setUsername('');
  setEmail('');
  setPassword('');
  setConfirmPassword('');
};

return (
  <form className='signup-container' onSubmit={handleRegistration}>
    <div className='status_signup'>{connectionStatus}</div>
    <div className="buttom-down-s">
      <Link to="/Authorization">
        <button className="login-button-s">Login</button>
      </Link>
      <Link to="/SignUp">
        <button className="registration-button-s">Register</button>
      </Link>
    </div>
    <label htmlFor="username">Username:</label>
    <input
      type="name"
      id="username"
      value={username}
      onChange={e => setUsername(e.target.value)}
      placeholder="Enter your username"
    />
    <label htmlFor="email">Email:</label>
    <input
      type="email"
      id="email"
      value={email}
      onChange={e => setEmail(e.target.value)}
      placeholder="Enter your email"
    />
    <label htmlFor="password">Password:</label>
    <input
      type="password"
```

```
        id="password"
        value={password}
        onChange={e => setPassword(e.target.value)}
        placeholder="Enter your password"
      />
      <label htmlFor="confirmPassword">Confirm Password:</label>
      <input
        type="password"
        id="confirmPassword"
        value={confirmPassword}
        onChange={e => setConfirmPassword(e.target.value)}
        placeholder="Confirm your password"
      />
      <button type="submit" className="submit-button">Register</button>
    </form>
  );
}

export default SignUp;
```

Програмний код функції пошуку, розробленої на .Net.

```
using Local_community_Back.Data;
using Local_community_Back.Model;
using Microsoft.AspNetCore.Mvc;


namespace Local_community_Back.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class SearchController : ControllerBase
    {
        private readonly CommunitySearch _communitySearch;

        public SearchController(CommunitySearch communitySearch)
        {
            _communitySearch = communitySearch;
        }

        [HttpGet]
        public async Task<IActionResult> Search([FromQuery] string query)
        {
            var appealResults = await _communitySearch.SearchAppeals(query);
            var departmentResults = await _communitySearch.SearchDepartments(query);
            var financeResults = await _communitySearch.SearchFinances(query);
            var newsResults = await _communitySearch.SearchNews(query);
            var mapResults = await _communitySearch.SearchMaps(query);
            var infrastructureResults = await _communitySearch.SearchInfrastructures(query);

            var combinedResults = new
            {
                AppealResults = appealResults,
                DepartmentResults = departmentResults,
                FinanceResults = financeResults,
                NewsResults = newsResults,
                MapResults = mapResults,
                InfrastructureResults = infrastructureResults
            };

            return Ok(combinedResults);
```

```
      }
    }
}
```

Програмний код сторінки внесення заяв, розробленого на React.

```
import React, { useEffect, useState } from "react";
import { Link } from 'react-router-dom';
import Footer from '../Footer/footer';
import './style.css';
import axios from 'axios';
import { useTranslation } from 'react-i18next';

const Appeal = () => {
  const [data, setData] = useState([]);
const appealTypes = ['Appeal', 'Complaints', 'Statements', 'Proposal'];
  const [formData, setFormData] = useState({
    fullName: '',
    type: '',
    address: '',
    description: '',
    phoneNumber: '',
    image: null,
  });
  const [showForm, setShowForm] = useState(false);

  useEffect(() => {
    fetchData();
  }, []);

  const fetchData = async () => {
    try {
      const response = await axios.get('https://localhost:44369/api/Appeal');
      setData(response.data);
    } catch (error) {
      console.error('Error fetching data:', error);
    }
  };

  const handleInputChange = (e) => {
    setFormData({ ...formData, [e.target.name]: e.target.value });
  };

  const handleImageChange = (e) => {
    setFormData({ ...formData, image: e.target.files[0] });
  };

  const submitForm = async () => {
    try {
      const form = new FormData();
      form.append('fullName', formData.fullName);
      form.append('type', formData.type);
      form.append('address', formData.address);
      form.append('description', formData.description);
      form.append('phoneNumber', parseInt(formData.phoneNumber, 10));
      form.append('ImageName', formData.image.name);

      const response = await axios.post('https://localhost:44369/api/Appeal', form);
      console.log('Data sent successfully:', response.data);
      fetchData();
      setShowForm(false);
```

```
  } catch (error) {
    if (error.response) {
      console.error('Error sending data. Server responded with an error:', error.response.data);
    } else if (error.request) {
      console.error('Error sending data. No response received from the server.');
    } else {
      console.error('Error sending data:', error.message);
    }
  }
};

const handleSubmit = async (e) => {
  e.preventDefault();
  submitForm();
};

const toggleForm = () => {
  setShowForm(!showForm);
};
const { t } = useTranslation();
console.log('Rendering Appeal component');

return (
  <div className="container">
    <div className="content-container">
      <div className="square-container">
        <div className="square-container-white">
          <h2>
            {t('Here you can leave a statement, suggestions, complaints, and appeals. To leave a complaint, log in to your account, upload a photo if necessary, and describe the problem in detail')}<br/>
          </h2>
          <p>
            <br/>1. {t('A citizens complaint  is one of the types of citizens appeals with a demand for restoration of rights and protection of the legitimate interests of citizens violated by actions (inaction), decisions of state bodies, local self-government bodies, enterprises, institutions, organizations, associations of citizens, officials.')}<br/>
            <br/>2. {t('An application is a document in which a private or official person makes a request with a specific proposal to the institution or official.')}<br/>
            <br/>3. {t('Appeal provides citizens with the opportunity to defend their rights and legitimate interests and to restore them in case of violation, to participate in the management of state and public affairs and to influence the improvement of the work of state and local self-government bodies, institutions, enterprises, and organizations.')}<br/>
            <br/>4. {t('Proposal (remarks) - citizens appeals, where advice and recommendations are expressed regarding the activities of state authorities and local self-government bodies, deputies of all levels, officials, as well as opinions are expressed regarding the regulation of social relations and living conditions of citizens, improvement of the legal basis of state and public life, socio-cultural and other spheres of activity of the state and society.')}
          </p>
        </div>
      </div>
      <div className="square-space"></div>
      <div className="square-container-appeal">
        <div className="square-white">
          {data.map(item => (
            <div key={item.id}>
              <img src={item.imageName} alt="" className="image" />
            </div>
          ))}
        </div>
        <div className="square-text-appeal">
          {data.length > 0 && (
            <div key={data[data.length - 1].id}>
              <p className="text-header-appeal">{t('Heads of the Chornoostrivska')} <br /> {t('settlement council')} <br />
                {t('Dzysya Mykhailo Semenovych')} <br />
                {t('legal person')}<br />
                {t(data[data.length - 1].fullName)}
```

```
            </p>
            <p className="type-centre">{t(appealTypes[data[data.length - 1].type])}</p>
            <p className="text-appeal"> {t('I')} {t(data[data.length - 1].fullName)} {t('living in')} {data[data.length -
1].adress} {t('and will ask about')} {data[data.length - 1].description}
            </p>
            <p className="appeal-footer">{t('Phone Number')}: 0{data[data.length - 1].phoneNumber} <br /></p>
            <p className="appeal-footer-left">{t('Signature')}: M.Svee</p>
           </div>
          )}
         <div>
          <button onClick={toggleForm} className="open-modal-btn">{t('Add another one')}</button>
          <Link to="/AppealAll" className="see-all-btn">{t('See all')}</Link>
          {showForm && (
           <div className="modal-overlay">
            <div className="modal-content">
             <form onSubmit={handleSubmit}>
              <label>
               Full Name:
               <input type="text" name="fullName" value={formData.fullName} onChange={handleInputChange} />
              </label>
              <label>
              Type:
              <select
               name="type"
               value={formData.type}
               onChange={handleInputChange}
               style={{ width: '105%', boxSizing: 'border-box' }}
              >
               <option value="">Select Type</option>
               <option value="Appeal">Appeal</option>
               <option value="Complaints">Complaints</option>
               <option value="Statements">Statements</option>
               <option value="Proposal">Proposal</option>
              </select>
             </label>

              <label>
               Address:
               <input type="text" name="address" value={formData.address} onChange={handleInputChange} />
              </label>
              <label>
               Description:
               <textarea name="description" value={formData.description} onChange={handleInputChange} />
              </label>
              <label>
               Phone Number:
               <input type="text" name="phoneNumber" value={formData.phoneNumber}
onChange={handleInputChange} />
              </label>
              <label>
               Image:
               <input type="file" name="image" accept="image/*" onChange={handleImageChange} />
              </label>
              <button type="submit">Submit</button>
              <button type="button" onClick={toggleForm}>Close</button>
             </form>
            </div>
           </div>
          )}
         </div>
        </div>
       </div>
       <div className="square-space"></div>
```

```
        <div className="footer-app">
          <Footer />
        </div>
      </div>
    </div>
  );
};

export default Appeal;
```

Програмний код сторінки внесення заяв, розробленого на .Net.

```
using Local_community_Back.Data;
using Local_community_Back.Model;
using Local_community_Back.ModelDto;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;


namespace Local_community_Back.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class AppealController : ControllerBase
    {
        private readonly CommunityContext _context;
        public AppealController(CommunityContext context)
        {
            _context = context;
        }

        // GET: api/<AppealController>
        [HttpGet]
        public async Task<ActionResult<IEnumerable<Appeal>>> Get()
        {
            return await _context.Appeal.ToListAsync();
        }

        // GET api/<AppealController>/5
        [HttpGet("{id}")]
        public async Task<ActionResult<Appeal>> Get(int id)
        {
            var appeal = await _context.Appeal.FindAsync(id);

            if (appeal == null)
            {
                return NotFound();
            }

            return appeal;
        }

        // POST api/<AppealController>
        [HttpPost]
        public async Task<ActionResult<Appeal>> Post([FromForm] AppealDto appealDto)
        {
            var appeal = new Appeal
            {
                FullName = appealDto.FullName,
                Adress = appealDto.Address,
                Description = appealDto.Description,
```

```
            PhoneNumber = appealDto.PhoneNumber,
            ImageName = appealDto.ImageName
        };

        if (appealDto.Type != "Appeal" && appealDto.Type != "Complaints" && appealDto.Type != "Statements" &&
appealDto.Type != "Proposal")
        {
            return BadRequest("Invalid appeal type.");
        }
        _context.Appeal.Add(appeal);
        await _context.SaveChangesAsync();

        return CreatedAtAction("Get", new { id = appeal.Id }, appeal);
    }

    // PUT api/<AppealController>/5
    [HttpPut("{id}")]
    public void Put(int id, [FromBody] string value)
    {
    }

    // DELETE api/<AppealController>/5
    [HttpDelete("{id}")]
    public void Delete(int id)
    {
    }
  }
}
```

Програмний код Telegram-Bot, розробленого на Go.

```
package bot
import (
  "log/slog"

  tgbotapi "github.com/crocone/telegram-bot-api"
  "main.go/internal/data"
  "main.go/internal/logger"
)
const (
  CommandStart    = "start"
  CommandCategory = "category"
)

var YesNo = map[string]bool{
  "YES": true,
  "NO":  false,
}
type SLogger interface {
  GetSLog() *slog.Logger
}
type SLog struct{}
func (L SLog) GetSLog() *slog.Logger { return logger.GetLogger() }

type button struct {
  name string
```

```
    data string
}

var sL = &SLog{}
var sLogger = new(SLog).GetSLog()

func HandleUpdates(updates tgbotapi.UpdatesChannel) {
    for update := range updates {
        if update.CallbackQuery != nil {
            handleCallbacks(update, updates)
        } else if update.Message.IsCommand() {
            handleCommands(update, updates)
        } else {
            handleMessage(update)
        }
    }
}

func handleCallbacks(update tgbotapi.Update, updates tgbotapi.UpdatesChannel) {
    data := update.CallbackQuery.Data
    chatID := update.CallbackQuery.Message.Chat.ID

    switch data {
    case CommandCategory:
        executeCategoryCommand(chatID)
    }
}

func handleCommands(update tgbotapi.Update, updates tgbotapi.UpdatesChannel) {
    command := update.Message.Command()
    chatID := update.Message.Chat.ID
    switch command {
    case CommandStart:
        executeStartCommand(chatID)
    default:
        UnknownCommand(chatID)
    }
}

func executeCategoryCommand(chatID int64) {
    msg := tgbotapi.NewMessage(chatID, "Задайте Ваше питання")
    sendMessage(msg, chatID)
}

func executeStartCommand(chatID int64) {
    msg := tgbotapi.NewMessage(chatID, "Доброго дня, я Ваш помічник з усіх питань Чорноострівської об’єднаної територіальної громади. Давайте розпочнемо")

    sendMessage(msg, chatID)
}
```

```
func UnknownCommand(chatId int64) {
   msg := tgbotapi.NewMessage(chatId, "Невідома команда")
   sendMessage(msg, chatId)
}

func sendMessage(msg tgbotapi.Chattable, chatID int64) {
   if _, err := Bot.Send(msg); err != nil {
      sLogger.Error("Send message error", "err", err)

      msg := tgbotapi.NewMessage(chatID, "Вибачте, я Вас не розумію, перефразуйте своє питання")
      sendMessage(msg, chatID)
   }
}
func YesNoKeyMarkup() tgbotapi.ReplyKeyboardMarkup {
   buttons := tgbotapi.NewReplyKeyboard()
   row := make([]tgbotapi.KeyboardButton, len(YesNo))

   counter := 0
   for button, _ := range YesNo {
      row[counter] = tgbotapi.NewKeyboardButton(button)
      counter++
   }

   buttons.Keyboard = append(buttons.Keyboard, row)

   return buttons
}

func handleMessage(update tgbotapi.Update) {
   chatID := update.Message.Chat.ID
   msg := tgbotapi.NewMessage(chatID, data.GetAnswer(update.Message.Text))

   sendMessage(msg, chatID)
}
```