

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавр

(ступінь вищої освіти)

на тему

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

ДЛЯ ОБГОВОРЕННЯ МІСЦЕВИХ ПРОБЛЕМ

SOFTWARE FOR A WEBSITE

TO DISCUSS LOCAL ISSUES

Виконав(ла): студент(ка) 4 курсу, групи КНТ-131

Спеціальності 121 Інженерія програмного

(код і найменування спеціальності)

забезпечення

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

АНТЮХОВ В.А.

(ПРИЗВИЩЕ та ініціали)

Керівник СКРУПСЬКИЙ С.Ю.

(ПРИЗВИЩЕ та ініціали)

Рецензент ІЛЛЯШЕНКО М.Б.

(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет КНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)
Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
Сергій СУББОТІН
“ ” 2025 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

АНТЮХОВА Вадима Андрійовича

(ПРІЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Програмне забезпечення для обговорення місцевих проблем. Software for a Website to Discuss Local Issues

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

затверджені наказом закладу вищої освіти від “28” квітня 2025 року № 209

2. Строк подання студентом проєкту (роботи) 03 червня 2025 року

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

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Матеріали і методи. 3. Опис програми. 4. Експлуатація, тестування та експериментальне дослідження програми.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, кількість слайдів, плакатів) _____

Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	СКРУПСЬКИЙ С.Ю., доцент		
Нормоконтроль	КАМІНСЬКА Ж.К., асистент		

7. Дата видачі завдання “28” квітня 2025 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	1 тиждень	Розділ 1
3	Вибір мови програмування та інших технологій розробки.	2 тиждень	Розділ 2
4	Розробка структури програми.	2 тиждень	Розділ 3
5	Розробка програми.	3-4 тижні	Розділи 3, 4
6	Тестування та експериментальне дослідження програмного забезпечення.	5 тиждень	Розділ 4
7	Оформлення пояснювальної записки та документів до неї.	6 тиждень	Додатки
8	Нормоконтроль та рецензування.	7 тиждень	
9	Захист роботи.	8 тиждень	

Студент(ка)

(підпис) Вадим АНТЮХОВ
(Ім'я ПРИЗВИЩЕ)

Керівник проєкту (роботи)

(підпис) Степан СКРУПСЬКИЙ
(Ім'я ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра:
94 с., 3 табл., 37 рис., 3 дод., 18 джерел.

JAVASCRIPT, NODE.JS, МОБІЛЬНИЙ ЗАСТОСУНОК, МОВА ПРОГРАМУВАННЯ, ОБГОВОРЕННЯ, ПОДІЯ, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ.

Об'єкт дослідження – процес розробки програмного забезпечення для обговорення регіональних питань.

Предмет дослідження – програмні засоби для обговорення регіональних питань.

Мета роботи – розробка програмного забезпечення для обговорення місцевих проблем.

Матеріали, методи та технічні засоби: мова програмування JavaScript, середовище виконання Node.js.

Результати. Розроблено проєктні рішення для створення програмного забезпечення для обговорення місцевих проблем. Створено програмне забезпечення для обговорення місцевих проблем. Проведено тестування розробленої програмної системи для обговорення місцевих проблем.

Висновки. Мету роботи досягнуто. Розроблено програмне забезпечення для обговорення місцевих проблем за допомогою мови програмування JavaScript та середовища виконання Node.js.

Галузь використання – місцеве управління.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 94 pages, 3 tables, 37 figures, 3 appendixes, 18 sources.

JAVASCRIPT, NODE.JS, MOBILE APPLICATION, PROGRAMMING LANGUAGE, DISCUSSION, EVENT, SOFTWARE.

The object of research is the process of developing software for discussing regional issues.

The subject of the research is software for discussing regional issues.

The purpose of this work is to develop the software for discussing local problems.

Materials, methods and technical tools: JavaScript programming language, Node.js runtime environment.

Results. Project solutions for the creation of software for discussing local problems has been designed. The software for discussing local problems has been developed. The developed software system for discussing local problems has been tested.

Conclusions. The goal of the work has been achieved. The software tools for discussing local problems using the JavaScript programming language and the Node.js runtime environment have been developed.

Scope of use – local government.

ЗМІСТ

	С.
Перелік скорочень та умовних позначок	8
Вступ	9
1 Аналіз предметної області	11
1.1 Програмне забезпечення для обговорення місцевих проблем	11
1.2 Аналіз програмного забезпечення для обговорення місцевих проблем	20
1.3 Висновки за розділом 1	28
2 Матеріали і методи	30
2.1 Вибір мови програмування	30
2.2 Вибір середовища виконання для роботи програмного забезпечення для обговорення місцевих проблем	33
2.3 Висновки за розділом 2	36
3 Опис програми	38
3.1 Структура програмного забезпечення для обговорення місцевих проблем	38
3.2 Функціонування програмного забезпечення для обговорення місцевих проблем	40
3.3 Розробка бази даних програмного забезпечення для обговорення місцевих проблем	44
3.4 Проєктування інтерфейсу програмного забезпечення для обговорення місцевих проблем	46
3.5 Висновки за розділом 3	50
4 Експлуатація, тестування та експериментальне дослідження програми	51
4.1 Призначення й умови застосування програми	51
4.2 Характеристики програми для обговорення місцевих проблем	52
4.3 Інструкція по експлуатації програми	53
4.3.1 Звернення до програми	53
4.3.2 Вхідні й вихідні дані	53
4.3.3 Повідомлення	54

4.4 Виконання програмного забезпечення для обговорення місцевих проблем	54
4.5 Тестування програмного забезпечення для обговорення місцевих проблем	59
4.6 Висновки за розділом 4	60
Висновки	61
Перелік джерел посилань	64
Додаток А Технічне завдання	66
Додаток Б Фрагмент тексту програми	70
Додаток В Слайди презентації	87

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

MVC – Model-View-Controller;

БД – база даних;

МП – місцева проблема;

ПЗ – програмне забезпечення.

ВСТУП

У сучасному інформаційному суспільстві надзвичайно важливо забезпечити ефективну та доступну комунікацію між мешканцями громад, представниками місцевої влади та громадськими організаціями. У цьому контексті створення для обговорення місцевих проблем є вкрай актуальним. Такий програмний ресурс дозволяє організувати відкритий діалог, забезпечити зворотний зв'язок, сприяти формуванню громадських ініціатив і залученню жителів до процесів прийняття рішень [1], [2].

Вебплатформа для обговорення місцевих проблем може слугувати засобом прозорості взаємодії громади з владою, де обговорення важливих питань, публікація планів розвитку та моніторинг прийнятих рішень сприятимуть підвищенню довіри населення до управлінських процесів. Крім того, такий сайт стимулює громадську активність, адже кожен мешканець отримує можливість висловити свою думку, долучитися до обговорень та підтримати ті чи інші ініціативи [3], [4].

В умовах децентралізації, коли громади отримують більше повноважень і відповідальності, подібне програмне забезпечення стає важливим інструментом реалізації принципів самоврядування. Він також може бути незамінним у кризових ситуаціях, коли необхідно оперативно інформувати населення та координувати дії. Саме тому програмне забезпечення для обговорення місцевих проблем є не лише актуальним, а й стратегічно важливим для розвитку громадянського суспільства [5], [6].

Проте деякі програмні засоби для обговорення місцевих проблем мають низку недоліків, які можуть впливати на їхню ефективність та популярність серед користувачів. Основними проблемами таких програмних засобів є нестабільна робота, поганий інтерфейс або обмежені функціональні можливості, що ускладнюють користування ресурсом. Певні труднощі може спричинити і недостатній рівень захисту персональних даних. Якщо користувачі не впевнені у конфіденційності своєї інформації, це може

призвести до зниження довіри до платформи. Іншим недоліком є недостатній рівень модерації контенту. За відсутності чітких правил та контролю подібне програмне забезпечення може стати місцем для поширення дезінформації, агресивної поведінки або спаму, що відлякує добросовісних користувачів.

Тому актуальною є розробка програмного забезпечення для обговорення місцевих проблем. У дипломній кваліфікаційній роботі бакалавра розв'язується актуальне завдання розробки програмного забезпечення для обговорення місцевих проблем.

Для досягнення поставленої мети у кваліфікаційній роботі бакалавра необхідно розв'язати такі задачі:

- виконати аналіз предметної області та програмних засобів для обговорення місцевих проблем;
- здійснити проєктування програмного забезпечення для обговорення місцевих проблем;
- створити програмне забезпечення для обговорення місцевих проблем;
- виконати тестування розробленого програмного забезпечення для обговорення місцевих проблем.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Програмне забезпечення для обговорення місцевих проблем

Програмне забезпечення для обговорення місцевих проблем призначене для створення інтерактивного середовища, де мешканці певної громади або території можуть спільно висловлювати думки, обговорювати нагальні питання, брати участь у подіях, ініціювати зміни та взаємодіяти з місцевою владою або активістами [1], [2].

Основна мета такого програмного забезпечення полягає в тому, щоб надати ефективний інструмент комунікації для вирішення проблем місцевого значення, формування суспільної думки, підвищення громадської активності та прозорості у прийнятті рішень. У рамках програмного забезпечення для обговорення місцевих проблем реалізується можливість створення обговорень, публікацій, організації подій, збору пропозицій, проведення опитувань та залучення користувачів до участі в діалогах [1], [2].

Функціонал програмного забезпечення для обговорення місцевих проблем часто включає особисті кабінети користувачів, систему повідомлень, реєстрацію подій, облік участі, створення групових дискусій, а також модерацію та аналітику активності. Завдяки цьому забезпечується зручність використання ресурсу як для пересічних громадян, так і для організаторів ініціатив [1], [2].

Програмне забезпечення, призначене для онлайн обговорення місцевих проблем, створюється як цифрове середовище для обміну думками, збору ініціатив і взаємодії між жителями певної території. У цьому просторі формується діалог, де мешканці можуть ділитися своїми спостереженнями, висловлювати позиції щодо актуальних подій або пропонувати шляхи розв'язання конкретних питань, що турбують громаду. Розроблення таких програмних платформ передбачає не лише технічну реалізацію, а й глибоке розуміння соціальної динаміки, тобто того, як люди обговорюють важливе, як

реагують на зміну середовища і як налагоджується співпраця на рівні невеликих спільнот громадян [1], [2].

Інструменти, що інтегруються у такі програмні застосунки, слугують для створення зручного середовища, у якому комунікація не обмежується лише текстом — тут з'являються функції візуального взаємодії, підтримка відеозустрічей, інтерактивних голосувань, цифрових дошок для спільного напрацювання рішень. Це не просто платформи для обміну інформацією, а конструкції, які допомагають будувати локальну відповідальність і підживлювати почуття залученості кожного користувача до загального процесу прийняття рішень [1], [2].

Такі програми не обмежуються аудиторією певного віку або професійної належності, оскільки вони націлені на широку участь, незалежно від технічних навичок або попереднього досвіду в громадській діяльності. Завдяки цьому з'являється змога створити нову якість взаємодії в цифровому просторі, де кожен голос має значення, а кожне повідомлення може стати початком змін. Усе це формує цілісний інструмент для підтримки активності на місцевому рівні та сприяє розвитку самоврядних ініціатив [1], [2].

Програмні платформи, що орієнтовані на обговорення місцевих питань, формуються як простір для діалогу між людьми, яких об'єднують спільні інтереси, життєві обставини або прагнення впливати на своє оточення. Вони створюються для тих, хто прагне бути залученим до суспільних ініціатив, хто шукає підтримку, взаєморозуміння або хоче поділитися власним досвідом. У таких віртуальних середовищах можна взаємодіяти на теми, що охоплюють найрізноманітніші аспекти життя громади, від питань здоров'я до місцевої економіки, від ідей розвитку інфраструктури до особистих ініціатив. Водночас аудиторія не обмежується фаховими спільнотами, охоплюється широкий спектр користувачів із різним рівнем досвіду, знань та мотивації [1], [2].

Значну увагу в процесі створення таких програмних інструментів приділяється візуальному оформленню та зручності навігації. Сприйняття інтерфейсу має бути інтуїтивним, щоб заохочувати до першого знайомства з

платформою та формувати бажання повертатися. Центром взаємодії служить головна сторінка, де відображаються теми, що відповідають інтересам конкретного користувача, активні обговорення та заплановані події. Можливість налаштовувати особисті профілі дозволяє краще проявити індивідуальність, водночас забезпечуючи органічне включення до груп за інтересами [1], [2].

Функціонал подібної програмної платформи орієнтовано на підтримку активної участі в житті цифрової спільноти. Це може включати розділи для повідомлень, публікацію зображень та відео, інтерактивні обговорення у форматі форуму. Пошукова система полегшує навігацію, а організаційні інструменти — такі як планування подій чи сповіщення — допомагають краще координувати ініціативи. Програмне забезпечення цього типу розроблюється з орієнтацією на залучення, налагодження комунікації та підтримку співпраці серед мешканців однієї місцевості, що прагнуть змін і вдосконалення спільного середовища [1], [2].

Використання програмного забезпечення для обговорення локальних тем дозволяє зосередити увагу на важливих аспектах сталого розвитку в межах громади. Такий цифровий простір може виконувати роль осередку, де передається практичний досвід у сфері захисту довкілля, обговорюються способи раціонального використання ресурсів, а також порушуються питання, пов'язані з альтернативною енергетикою, місцевими ініціативами з озеленення та колективними екологічними діями. Через обмін знаннями та особистими історіями формується коло учасників, яких об'єднує бажання впливати на зміни у своєму середовищі [1], [2].

Інформаційне наповнення в межах такого програмного сервісу здатне охоплювати навчальні матеріали, приклади успішних практик, аналітичні огляди та мультимедійний контент, спрямований на підвищення свідомості громадян. Спільнота формується навколо ідеї взаємної підтримки й обміну ідеями, що сприяють сталому способу життя. Така програмна платформа не лише допомагає збирати й поширювати перевірені факти, а й стимулює участь

у колективних проєктах, орієнтованих на збереження навколишнього середовища [1], [2].

Розроблення подібного програмного інструменту залежить від обраного технічного підходу. Залежно від глибини володіння технологіями та наявних ресурсів, можна адаптувати процес створення системи — від класичних технік із програмуванням до більш доступних методів, які не вимагають кодування. Кожен із підходів має свої особливості — від гнучкості та налаштувань до швидкості впровадження і доступності для користувачів без технічної підготовки. Вибір визначається специфікою задуму, масштабами проєкту та часовими рамками [1], [2].

Варто зазначити, що створення цифрового інструменту для локального обговорення проблем не лише відкриває шлях до нових форм громадської взаємодії, а й дозволяє реалізовувати проєкти, які базуються на турботі про майбутнє. Залученість, відкритість до діалогу та спільне прагнення до змін стають основою для розвитку такої платформи [1], [2].

Процес створення програмного рішення для обговорення локальних питань передбачає, перш за все, зосередження не на технічній реалізації, а на попередньому осмисленні ключових ідей та потреб спільноти. Основна складність полягає не в коді чи інструментах, а в усвідомленні мети, яку повинна виконувати система, а також у правильному визначенні підходу до її реалізації. Залежно від обставин, може бути обрано як варіант розробки з нуля, так і використання наявного середовища, яке дозволяє адаптацію під конкретні запити. У більшості випадків наявність готової технічної основи значно спрощує шлях і дозволяє зосередитись на змісті та функціональності програмного забезпечення для обговорення місцевих проблем [1], [2].

Щоб робота над створенням цифрової платформи для місцевої взаємодії мала сенс, необхідно з самого початку сформулювати головне призначення такого інструменту. Це внутрішнє уявлення про користь, яку зможуть отримати мешканці, має стати орієнтиром для подальших рішень — від структури взаємодії до дизайну та контенту. Завдяки чіткому розумінню

того, яку роль сервіс відіграватиме у житті громади, простіше будувати логіку використання, передбачити ключові функції та сформувати середовище, що сприяє діалогу та підтримці [1], [2].

Візуалізація типових сценаріїв користування платформою значно полегшує розробку та допомагає не відхилятися від запланованого шляху. Уявлення про можливі дії, які будуть здійснювати учасники, дозволяє на рівні концепції охопити реальні потреби, з якими звертатимуться до інструменту. Такий підхід робить продукт ближчим до користувача та дозволяє зберігати цілісність у всіх етапах розробки — від перших макетів до остаточної реалізації [1], [2].

На початковому етапі створення цифрового інструменту для вирішення місцевих питань особливу увагу приділяється візуальному наповненню та стилістичному оформленню. Важливо забезпечити цілісність сприйняття, де кожен візуальний елемент підсилює загальну концепцію та сприяє досягненню поставленої мети. Естетика інтерфейсу користувача має не лише відповідати загальному стилю, а й допомагати користувачеві інтуїтивно орієнтуватися у функціоналі [2], [3].

Перш ніж інтерактивна платформа для обговорення місцевих проблем набуде форми, необхідно підготувати графічну складову. Ключовим тут є символ, який репрезентує саму програму, він відображає характер продукту ще до першого натискання. Також важливо подбати про вигляд початкових екранів, які зустрічають користувача на вході та під час реєстрації. Обираючи оформлення для основних розділів, важливо враховувати зручність сприйняття, поєднання кольорів та читабельність елементів [2], [3].

Після візуального налаштування відбувається етап структурування навігації. Орієнтуючись на основні сценарії користування, визначається перелік розділів, які забезпечать доступ до ключових функцій програмного забезпечення для обговорення місцевих проблем. Передбачається, що кожен розділ платформи виконуватиме конкретну роль, від обміну новинами до взаємодії між учасниками, від управління профілем до спільної організації

подій. Правильно сформована навігація дозволяє зменшити бар'єри у користуванні та зробити платформу зручною для обговорення актуальних тем громади [2], [3].

Після первинного оформлення цифрового середовища для локальних дискусій виникає потреба у вдосконаленні його функціонування. У перші етапи майже завжди спостерігається необхідність у додатковому коригуванні, що являє собою нормальний процес, що супроводжує будь-яку розробку. Спочатку формується початковий варіант, який слугує основою для подальшої перевірки, аналізу та редагування [3], [4].

Під час ознайомлення з початковим виглядом системи часто виявляються зайві елементи або ті, що ускладнюють навігацію. Іноді виявляються деталі, які не відповідають задуму чи знижують зручність використання. Найефективніше оцінювати все очима потенційного учасника платформи, чи можливо швидко зорієнтуватися, чи доступ до важливих функцій інтуїтивно зрозумілий, чи не відволікають елементи інтерфейсу від головної мети взаємодії [3], [4].

Якщо будь-який компонент заважає основному сценарію взаємодії, потрібно вносити зміни. Наприклад, варто звернути увагу на поєднання кольорів або логіку структури меню. Надійним підходом є аналіз вражень кількох незалежних користувачів. Ще одним корисним інструментом є експериментальне тестування різних варіантів інтерфейсу, що дозволяє виявити найбільш ефективні рішення. Завдяки цьому стає можливим поступове вдосконалення платформи до того рівня, на якому вона дійсно сприятиме плідному обговоренню місцевих проблем [3], [4].

Перш ніж цифрове середовище для обговорення локальних тем стане доступним широкому загалу, необхідно пройти кілька важливих етапів, пов'язаних із його розміщенням. Поширення платформи поза межами офіційних маркетів значно знижує її шанси на видимість і викликає сумніви щодо рівня безпеки та захисту персональних даних. Розміщення в офіційних

цифрових крамницях дає змогу скористатися перевіреними механізмами контролю якості та довіри з боку користувачів [3], [4].

Для оприлюднення програмної платформи для обговорення місцевих проблем необхідно створити облікові записи в системах розробників. Платформи мають різні підходи до процесу авторизації, деякі з них вимагають офіційної наявності зареєстрованої організації та публічного представництва в Інтернеті. Після цього стає можливим керування публікацією, її оновленням та взаємодією з аудиторією. Якщо розробка здійснюється через сторонній сервіс, обов'язковим буде узгодження з технічною інфраструктурою цієї платформи [3], [4].

Поряд із технічними діями варто враховувати і нормативну сторону, особливо у сфері захисту особистої інформації користувачів програмного забезпечення для обговорення місцевих проблем. Оскільки платформа, орієнтована на громадські ініціативи, передбачає обробку користувацьких даних, необхідно дотримуватися відповідних вимог, установлених законодавством відповідного регіону. Ці положення регламентують, яким чином збирається, зберігається та використовується інформація, що надходить у межах цифрового простору [3], [4].

Після виконання усіх попередніх підготовчих кроків відкривається можливість публічного розміщення платформи програмного забезпечення для обговорення місцевих проблем. Цей етап вимагає подання ключових інформаційних матеріалів, від короткого опису до візуальних елементів, які сприятимуть розумінню суті та цінності програмного застосунку. За потреби можна додати демонстраційні матеріали, що наочно відображають функціонал програмного забезпечення для обговорення місцевих проблем. Коли все підготовлено, запускається процес публікації, після чого відкривається новий цифровий простір для місцевої взаємодії та обміну думками, що функціонує автономно та під повним контролем ініціаторів [3], [4].

Формування цифрового середовища для активного обговорення локальних питань вимагає тривалого залучення та продуманої організації. Цей

процес не є миттєвим і передбачає постійну підтримку взаємодії між учасниками, зокрема через спеціалізоване програмне забезпечення. Використання таких інструментів дає змогу створити простір, у якому обмін думками стає доступнішим і структурованим. Учасники можуть спілкуватися незалежно від фізичного розташування, поступово вдосконалюючи здатність аргументовано висловлювати свої позиції [4], [5].

Цифрові платформи, що орієнтовані на комунікацію, виконують роль інтерфейсу для обміну повідомленнями, координації дій та збереження контексту дискусій. Саме завдяки їм вдається підтримувати зв'язок між членами громади, розвивати теми, які хвилюють мешканців певного району, а також отримувати підтримку чи пораду в ситуаціях, які потребують колективного обговорення. Такі середовища природно сприяють появі активної взаємодії, де кожен охочий може поділитися своїм баченням або поставити запитання [4], [5].

Дистанційні формати обговорення місцевих подій за допомогою відповідного програмного забезпечення часто є каталізатором участі у стандартному форматі, оскільки дають змогу сформулювати первинну думку або перевірити її, перш ніж висловлюватися в публічному просторі. У цифровому середовищі можна проводити обговорення в асинхронному режимі, що особливо зручно для тих, хто має обмежений час або доступ до спільноти лише періодично. Сам характер таких обговорень підштовхує до більш зваженої участі, оскільки кожен допис зазвичай зберігається і може бути проаналізований згодом [4], [5].

Програмне забезпечення для дискусій також використовується як механізм пошуку спільних рішень. Воно дозволяє збирати різні точки зору, виявляти альтернативи та формулювати ініціативи, що базуються на колективному мисленні. Таким чином, програмне забезпечення для обговорення місцевих проблем не лише спрощує обмін ідеями, а й відкриває нові можливості для громадської участі та самореалізації в межах місцевої громади [4], [5].

Під час створення цифрового простору для обговорення локальних питань особливу увагу приділяється адаптації функціоналу до реального життя громади. У фокусі розробки перебуває не лише технічна сторона, а й розуміння соціального контексту, в якому буде використовуватися застосунок. Із самого початку процесу проектування враховується потреба у простому, інтуїтивно зрозумілому інтерфейсі, який дозволить користувачам з різним рівнем цифрової грамотності брати участь у діалозі [5], [6].

Особлива увага при розробці програмного забезпечення для обговорення місцевих проблем звертається на технічну гнучкість, щоб за необхідності платформа могла масштабуватися під інші громади або розширювати функціональність за рахунок модулів, які не змінюють основної логіки обговорення. Програмне забезпечення для обговорення місцевих проблем має залишатися відкритим до змін, враховуючи специфіку кожної окремої громади та можливість її розвитку з часом. Залежно від потреб, розглядається впровадження інструментів аналітики, що дають змогу виявляти активні теми, розуміти динаміку дискусій і вчасно реагувати на актуальні проблеми [5], [6].

Визначено, що програмне забезпечення для обговорення місцевих проблем призначене для створення інтерактивного середовища, де мешканці певної громади або території можуть спільно висловлювати думки, обговорювати нагальні питання, брати участь у подіях, ініціювати зміни та взаємодіяти з місцевою владою або активістами. Основна мета такого програмного забезпечення полягає в тому, щоб надати ефективний інструмент комунікації для вирішення проблем місцевого значення, формування суспільної думки, підвищення громадської активності та прозорості у прийнятті рішень. У рамках програмного забезпечення для обговорення місцевих проблем реалізується можливість створення обговорень, публікацій, організації подій, збору пропозицій, проведення опитувань та залучення користувачів до участі в діалогах.

1.2 Аналіз програмного забезпечення для обговорення місцевих проблем

Проаналізуємо програмне забезпечення для обговорення місцевих проблем [7]-[14].

Програмне забезпечення Recollective (рис. 1.1) призначено для організації глибоких онлайн-обговорень, що стосуються актуальних місцевих питань. Його інструменти створено спеціально для того, щоб забезпечити активне залучення мешканців до процесу вирішення проблем, які безпосередньо впливають на їхнє щоденне життя. Через Recollective зручно проводити інтерактивні дискусії між користувачами, які беруть участь у житті своєї громади, обмінюючись думками щодо важливих соціальних чи інфраструктурних тем [7], [12].

Engaged Communities

Drawing on decades of experience, Recollective helps innovative organizations unlock the power and potential of their communities. Whether you start with a single research study, or you're ready to build an ongoing, strategic foundation of people and data fuelling rapid, company-wide data-driven decisions, Recollective provides the platform and knowledge for sustainable success.

Discover Communities

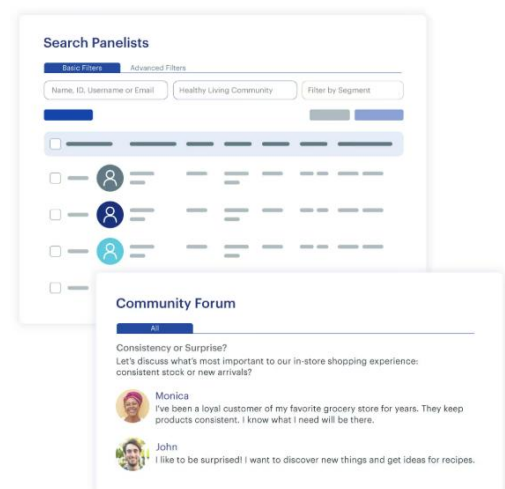


Рисунок 1.1 – Програмне забезпечення Recollective [12]

Платформа Recollective підтримує спілкування у форматі відео фокус груп, що дозволяє організовувати живі зустрічі за участі кількох десятків користувачів, незалежно від того, який пристрій вони використовують. Це відкриває можливість для проведення відкритих і глибоких обговорень локальних ініціатив у реальному часі [7], [12].

Сервіс Recollective передбачає заздалегідь підготовлену модуерацію обговорень: модератори мають змогу завантажувати запитання, мультимедійні матеріали, документи або навіть опитування, які спрямовані на стимулювання дискусії навколо тем, що турбують громаду. В ході обговорень передбачено обмін екранами між учасниками або модератором, що дозволяє наочно демонструвати дані, проєкти або пропозиції, пов'язані з локальними проблемами. Усе це робить Recollective ефективним засобом для діалогу між мешканцями та забезпечення якісного зворотного зв'язку у справах, які стосуються їхніх спільнот [7], [12].

Основними характеристиками програмного забезпечення Recollective є такі [7], [12]:

- інструмент для онлайн дискусій: програмне забезпечення Recollective забезпечує просту організацію обговорень місцевих проблем у віртуальному середовищі [7], [12];

- підтримка якісної аналітики: програмне забезпечення Recollective дозволяє збирати та аналізувати глибокі якісні дані від учасників обговорення [7], [12];

- функція відео фокус груп: програмне забезпечення Recollective забезпечує проведення живих зустрічей декількох учасників із використанням будь-яких пристроїв [7], [12];

- можливість попереднього планування: модератори можуть завчасно готувати матеріали для обговорень, такі як тексти, опитування, зображення та відео [7], [12];

- підтримка обміну екраном: учасники обговорення мають можливість ділитися своїм екраном або переглядати демонстрацію від модератора під час сесій [7], [12];

- інтуїтивно зрозумілий інтерфейс: програмне забезпечення Recollective легко освоюється новими користувачами, що полегшує залучення до обговорення локальних тем [7], [12];

– масштабованість рішення: програмне забезпечення Recollective можна адаптувати під різну кількість користувачів та масштаби обговорення проблем [7], [12].

Серед недоліків програмного забезпечення Recollective можна відзначити обмеження в гнучкості інтерфейсу, що іноді ускладнює адаптацію платформи під специфічні потреби окремих громад. Також певні функції, зокрема аналітичні інструменти та можливості персоналізації, можуть бути доступні лише в розширених версіях, що потребують додаткових витрат. У деяких випадках спостерігається складність у використанні платформи для людей з низьким рівнем цифрової грамотності, що може знижувати ефективність обговорень місцевих проблем серед широкої аудиторії [7], [12].

Програмне забезпечення FlexMR (рис. 1.2) розроблене як ефективний засіб для організації обговорень, спрямованих на вирішення місцевих проблем. Його функціональність забезпечує створення простору для активної взаємодії громадян, експертів і представників органів влади, де обговорюються питання, що стосуються конкретних територіальних спільнот. Програмне забезпечення FlexMR надає змогу організовувати обговорення у зручному форматі — із використанням текстових і відео чатів, мультимедійних матеріалів та інтерактивних форумів [7], [13].

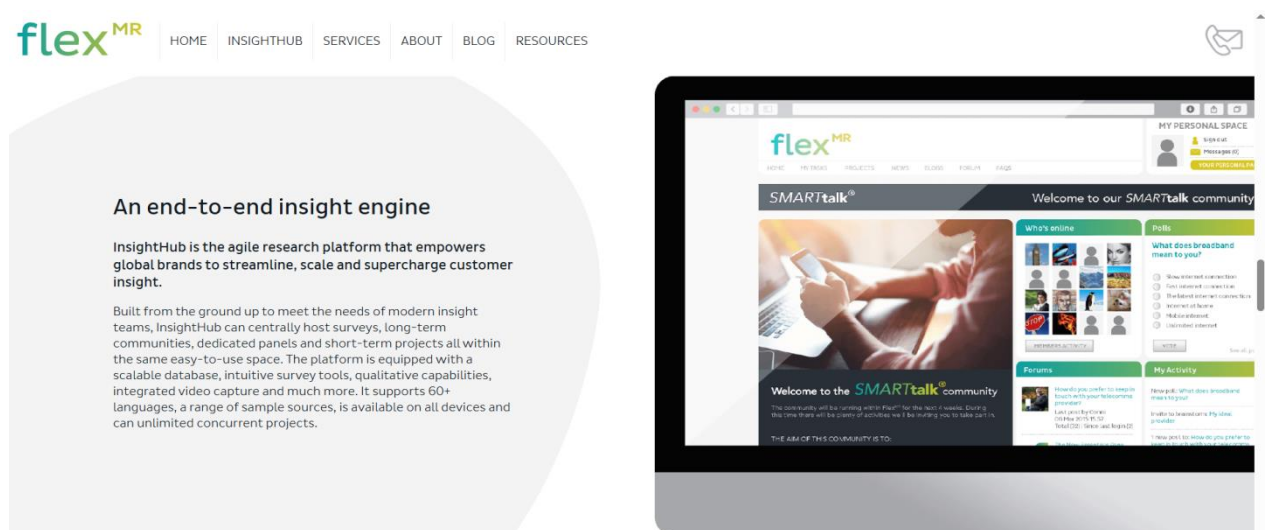


Рисунок 1.2 – Програмне забезпечення FlexMR [13]

Завдяки гнучким інструментам для збирання думок і швидкого опитування програмне забезпечення FlexMR дозволяє модераторам координувати обговорення місцевих проблем, заохочуючи участь та активність мешканців. Платформа підтримує інтеграцію аналітики, що допомагає виявляти ключові ідеї та оцінювати реакції спільноти. Важливою особливістю є можливість приватного спостереження за сесіями, що дозволяє представникам місцевої влади або відповідальним особам фіксувати хід обговорень і брати участь у процесі, не перешкоджаючи дискусії. Модератори мають змогу керувати діалогами та взаємодією між учасниками, забезпечуючи ефективність усіх етапів обговорення локальних тем. Програмне забезпечення FlexMR виступає як сучасний інструмент цифрової демократії, що дозволяє громадам більш усвідомлено брати участь у вирішенні важливих для них питань [7], [13].

Програмне забезпечення FlexMR має такі особливості [7], [13]:

- багатофункціональна дослідницька платформа: програмне забезпечення FlexMR забезпечує широкий спектр програмних інструментів для глибокого аналізу, збору думок і координації учасників обговорень місцевих проблем [7], [13];
- підтримка мультимедійного контенту: програмне забезпечення FlexMR дозволяє використовувати відео, аудіо, зображення та текст як засоби стимулювання дискусій і обміну думками [7], [13];
- інтерактивні форуми: програмне забезпечення FlexMR надає зручні онлайн простори для обговорення локальних тем, що сприяє залученню учасників і активному формуванню рішень [7], [13];
- можливість відео чатів: програмне забезпечення FlexMR дозволяє проводити прямі дискусії у форматі відео зв'язку між користувачами, зацікавленими сторонами та модераторами [7], [13];
- приватний чат для спостерігачів: програмне забезпечення FlexMR забезпечує конфіденційний зв'язок між організаторами та аналітиками без втручання в основну дискусію [7], [13];

- синхронна модерація: програмне забезпечення FlexMR дає змогу керувати взаємодією в реальному часі, координуючи обговорення місцевих проблем на різних рівнях одночасно [7], [13];
- аналітичні інструменти: програмне забезпечення FlexMR містить вбудовані засоби для аналізу відповідей, настроїв і ключових тем, що дозволяє краще зрозуміти громадську думку [7], [13];
- гнучке налаштування сесій: програмне забезпечення FlexMR забезпечує адаптацію формату обговорень до потреб конкретної аудиторії або типу місцевої проблеми [7], [13].

Серед недоліків програмного забезпечення FlexMR можна відзначити обмежену інтуїтивність інтерфейсу для нових користувачів, що може ускладнити початкову взаємодію з платформою. Деякі функції програмного забезпечення FlexMR вимагають додаткового навчання або супроводу з боку модераторів, що знижує швидкість адаптації для окремих учасників. Крім того, через велику кількість доступних інструментів можуть виникати труднощі з навігацією під час активних обговорень місцевих проблем. Іноді спостерігається затримка у роботі системи при одночасному підключенні великої кількості користувачів, що може негативно впливати на якість дискусій у режимі реального часу [7], [13].

Програмне забезпечення Vocal Views (рис. 1.3) розроблено для організації обговорень, що стосуються місцевих проблем, з акцентом на відео формат спілкування. Ця система створює зручне середовище для візуального представлення думок та активного обміну ідеями між учасниками, заохочуючи відкритий діалог на актуальні теми, важливі для місцевої громади. У межах віртуальних зустрічей забезпечується ефективна взаємодія між учасниками, яким надається змога підключатися до сесій, ділитися екраном, брати участь у спільному опрацюванні питань на інтерактивних дошках та відстежувати перебіг дискусії в режимі реального часу [7], [14].

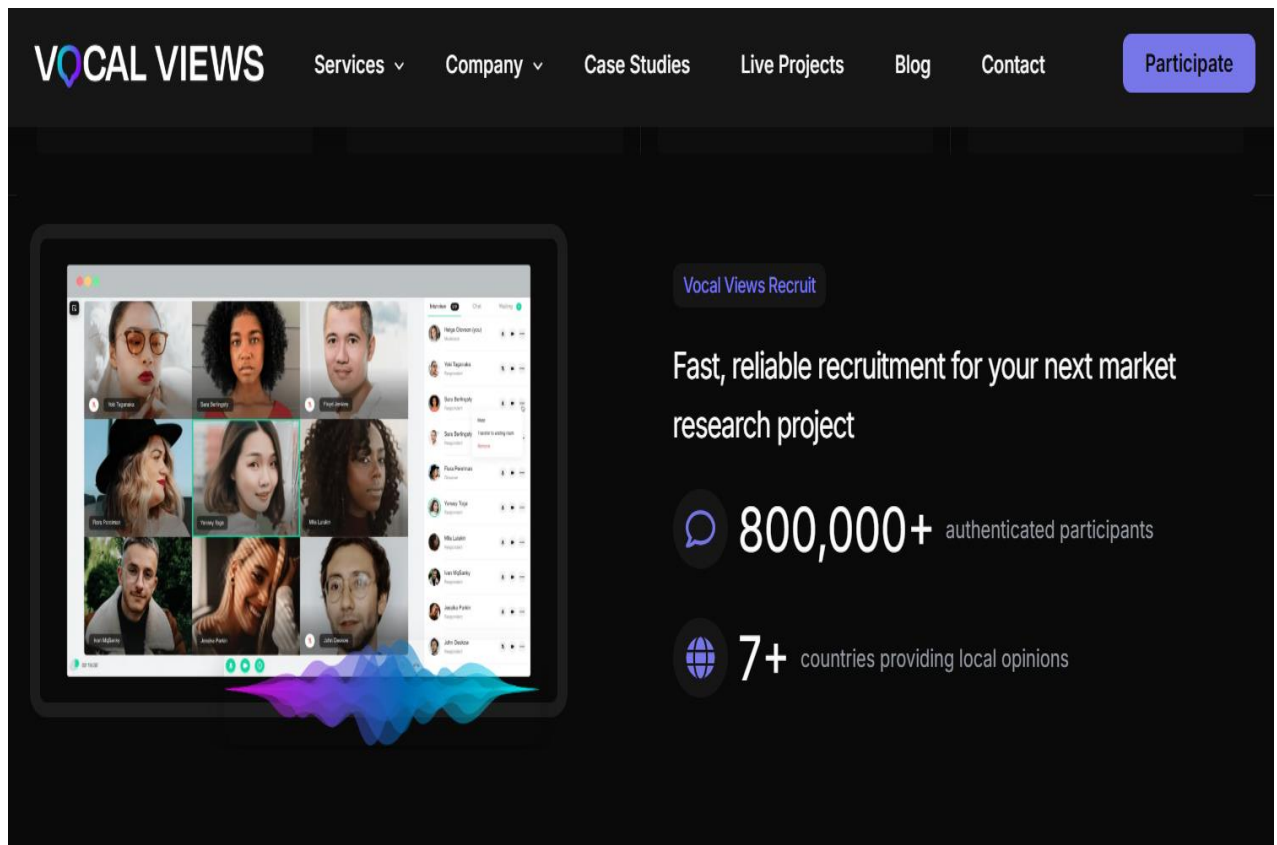


Рисунок 1.3 – Програмне забезпечення Vocal Views [14]

Для дослідження локальних проблем у програмному забезпеченні Vocal Views передбачено залучення учасників з великої попередньо відібраної спільноти, що дозволяє швидко сформувати релевантну групу для кожного конкретного обговорення [7], [14].

Користувачі програмного забезпечення Vocal Views можуть також самостійно додавати респондентів або скористатися готовими рішеннями для запрошення модераторів, які сприятимуть структурованому веденню розмови. Усе це робить програмне забезпечення Vocal Views зручним і ефективним інструментом для обговорення суспільно важливих локальних питань у форматі відео комунікації [7], [14].

Програмне забезпечення Vocal Views має такі особливості [7], [14]:

- відеоформат обговорень: програмне забезпечення Vocal Views надає користувачам можливість візуального спілкування між учасниками для ефективного обміну думками щодо місцевих проблем [7], [14];

- віртуальна кімната спостерігачів: програмне забезпечення Vocal Views дозволяє стороннім особам аналізувати дискусії в режимі реального часу без втручання у розмову [7], [14];

- інструменти взаємодії: програмне забезпечення Vocal Views надає доступ до інтерактивних функцій, таких як обмін екранами та вбудовані дошки для фіксації ідей та зворотного зв'язку [7], [14];

- велика база респондентів: програмне забезпечення Vocal Views надає доступ до чисельної спільноти користувачів, сегментованої за різними критеріями для точного добору учасників [7], [14];

- підтримка модерації: програмне забезпечення Vocal Views пропонує можливість залучення професійних модераторів для організації відео фокус-груп із фокусом на обговорення місцевих питань [7], [14].

Серед недоліків програмного забезпечення Vocal Views можна відзначити обмеження щодо кількості учасників у відео сесіях, що може ускладнювати організацію масштабних обговорень місцевих проблем. Крім того, деяким користувачам може бути незручно працювати виключно у форматі відео, особливо за умов нестабільного Інтернет з'єднання або недостатньої технічної підготовки. Також процес налаштування інтерактивних елементів може виявитися складним для новачків, що потребує додаткового часу на адаптацію до інтерфейсу. Також певні функції програмного забезпечення Vocal Views можуть бути доступні лише в розширених версіях, що потребують додаткових витрат. У деяких випадках спостерігається складність у використанні платформи програмного забезпечення Vocal Views для людей з низьким рівнем цифрової грамотності, що може знижувати ефективність обговорень місцевих проблем серед широкої аудиторії користувачів [7], [14].

Порівняльну характеристику програмного забезпечення для обговорення місцевих проблем наведено у таблиці 1.1.

Таблиця 1.1 – Порівняння програмного забезпечення для обговорення місцевих проблем

Критерій порівняння	Recollective [12]	FlexMR [13]	Vocal Views [14]
Підтримка текстових дискусій	+	+	+—
Підтримка великих груп учасників	+	+—	+
Гнучкість налаштувань	+	+	+—
Можливість пошуку учасників	+	+	+—
Легкість модерації	+	+	+—
Зручність інтерфейсу користувача	+—	+—	+

За результатами проведеного аналізу можна зробити висновок, що у наш час існує досить багато програмних засобів для обговорення місцевих проблем. Проте деякі програмні засоби для обговорення місцевих проблем мають низку недоліків, які можуть впливати на їхню ефективність та популярність серед користувачів. Основними проблемами таких програмних засобів є нестабільна робота, поганий інтерфейс або обмежені функціональні можливості, що ускладнюють користування ресурсом. Певні труднощі може спричинити і недостатній рівень захисту персональних даних. Тому актуальною є розробка програмного забезпечення для обговорення місцевих проблем.

При розробці програмного забезпечення для обговорення місцевих проблем необхідно забезпечити такі функціональні вимоги:

- підтримка можливості відображення переліку місцевих подій та проблем;
- підтримка можливості авторизації та реєстрації;

- підтримка можливості обговорення місцевих подій шляхом текстового спілкування з іншими користувачами;
- можливість додавання інформації про нову локальну подію або проблему;
- можливість додавання інформації про географічне місце проведення події;
- підтримка можливості роботи з екраном діалогів користувача;
- підтримка можливості зміни налаштувань профілю користувача.

1.3 Висновки за розділом 1

Визначено, що програмне забезпечення для обговорення місцевих проблем призначене для створення інтерактивного середовища, де мешканці певної громади або території можуть спільно висловлювати думки, обговорювати нагальні питання, брати участь у подіях, ініціювати зміни та взаємодіяти з місцевою владою або активістами. Основна мета такого програмного забезпечення полягає в тому, щоб надати ефективний інструмент комунікації для вирішення проблем місцевого значення, формування суспільної думки, підвищення громадської активності та прозорості у прийнятті рішень. У рамках програмного забезпечення для обговорення місцевих проблем реалізується можливість створення обговорень, публікацій, організації подій, збору пропозицій, проведення опитувань та залучення користувачів до участі в діалогах.

За результатами проведеного аналізу зроблено висновок, що у наш час існує досить багато програмних засобів для обговорення місцевих проблем. Проте деякі програмні засоби для обговорення місцевих проблем мають низку недоліків, які можуть впливати на їхню ефективність та популярність серед користувачів. Основними проблемами таких програмних засобів є нестабільна робота, поганий інтерфейс або обмежені функціональні можливості, що ускладнюють користування ресурсом. Певні труднощі може спричинити і

недостатній рівень захисту персональних даних. Тому актуальною є розробка програмного забезпечення для обговорення місцевих проблем.

Сформульовано функціональні вимоги до програмного забезпечення для обговорення місцевих проблем.

2 МАТЕРІАЛИ І МЕТОДИ

2.1 Вибір мови програмування

При розробці програмного забезпечення для обговорення місцевих проблем було використано мову програмування JavaScript [15], [16].

JavaScript спочатку він був лише скриптовою мовою для браузерів, але з часом розвинувся в повноцінну мову програмування. Завдяки підтримці стандартів ECMAScript та розвитку технологій JavaScript став стабільним і потужним інструментом для розробки як на клієнтській, так і на серверній стороні [15], [16].

Ця мова програмування підтримує дві основних парадигми: функціональне і об'єктно-орієнтоване програмування. Синтаксис мови програмування JavaScript подібний до таких мов як Java та C++, але має особливості динамічної типізації і асинхронної обробки, що дає змогу ефективно працювати з великими обсягами даних. Однією з головних переваг є вбудована підтримка обробників подій і можливість асинхронного виконання коду через відповідні механізми [15], [16].

Окрім використання на клієнтській стороні для створення інтерактивних вебсторінок, JavaScript дозволяє інтегрувати серверну частину за допомогою Node.js. Це відкриває нові можливості для створення складних вебзастосунків, включаючи чат-ботів, API, настільні та мобільні програми за допомогою таких інструментів, як Electron чи React Native [15], [16].

У розробці вебсайтів активно використовуються різні фреймворки, які значно спрощують створення інтерфейсів і обробку даних. Серед них варто відзначити React, Vue, Angular для фронтенду, а також Express та Socket.IO для створення серверних додатків у реальному часі [15], [16].

Мова JavaScript є однією з найбільш популярних мов програмування у світі, що робить її доступною для багатьох розробників. Велика кількість навчальних матеріалів, відеоуроків і онлайн форумів дозволяє швидко освоїти цю мову і знайти рішення для більшості технічних проблем [15], [16].

З недоліків мова JavaScript можна відзначити те, що її динамічний характер іноді призводить до помилок на ранніх етапах розробки, а асинхронне виконання коду може ускладнювати його читабельність. Проте ці проблеми частково вирішуються за допомогою TypeScript, що додає статичне типізування, підвищуючи стабільність коду [15], [16].

Для створення програмного забезпечення для вебсайтів, які сприяють обговорення місцевих проблем, вибір JavaScript є доцільним з кількох причин. Ця мова підтримується усіма сучасними браузерами, не потребує додаткових встановлень, що робить її дуже зручною для розробки. Крім того, завдяки великій кількості бібліотек і фреймворків, JavaScript дозволяє швидко і ефективно реалізувати потрібну функціональність. Використання Node.js для серверної частини дозволяє створювати стабільні і масштабовані системи для обробки даних і взаємодії з користувачами, що важливо для динамічних вебплатформ [15], [16].

Реалізація різноманітних функціональних можливостей, як-от інтерактивні повідомлення, форми зворотного зв'язку, голосування чи стрічки новин, відбувається завдяки широкому спектру бібліотек і фреймворків. Зокрема, технології React, Vue та Angular значно прискорюють побудову гнучких і чутливих інтерфейсів [15], [16].

На серверному рівні забезпечується стабільність і масштабованість, оскільки платформа Node.js дозволяє обробляти численні запити паралельно без втрати продуктивності. Такий підхід особливо ефективний для систем, які передбачають активну взаємодію користувачів та оперативне оновлення контенту [15], [16].

Підтримка JavaScript усіма сучасними веббраузерами без потреби встановлення додаткових компонентів спрощує запуск і використання готових рішень. Широка база відкритого коду, численні публічні модулі та активна спільнота дозволяють оперативно долати технічні труднощі й зменшувати час на реалізацію проєкту [15], [16].

Обґрунтування вибору мови програмування для розробки програмного забезпечення для обговорення місцевих проблем наведено у таблиці 2.1.

Таблиця 2.1 – Обґрунтування вибору мови програмування для розробки програмного забезпечення для обговорення місцевих проблем

Критерій порівняння мов програмування	Мова програмування		
	JavaScript	Python	PHP
Продуктивність на сервері	+	+-	+-
Підтримка асинхронного програмування	+	+-	-
Широка екосистема та бібліотеки	+	+-	+-
Зручність для створення ПЗ для обговорення місцевих проблем	+	-	-
Безпека даних	+	+	+-

Отже, для реалізації програмного забезпечення для обговорення місцевих проблем обрано мову програмування JavaScript, яка є однією з найпоширеніших та універсальних мов у сфері веброзробки, яка дозволяє створювати як клієнтську частину, так і серверну логіку застосунку. Ще одна перевага полягає у тому, що мова програмування JavaScript підтримується всіма сучасними браузерами без додаткових налаштувань, що забезпечує доступність платформи для максимальної кількості користувачів. Крім того, велика кількість відкритих програмних бібліотек, модулів та активна спільнота розробників дозволяють швидко знаходити рішення технічних задач і зменшувати час розробки. Підтримка JavaScript усіма сучасними веббраузерами без потреби встановлення додаткових компонентів спрощує запуск і використання готових рішень.

2.2 Вибір середовища виконання для роботи програмного забезпечення для обговорення місцевих проблем

В якості середовища виконання для створення програмного забезпечення для обговорення місцевих проблем було обрано середовище Node.js [17], [18].

Середовище Node.js слугує платформою, яка дозволяє виконання JavaScript-коду на стороні сервера. Завдяки використанню рушія, який було розроблено Google, досягається висока швидкість обробки коду. Робота з Node.js ґрунтується на моделі подій, що не потребує багатопоточності, проте забезпечує здатність обслуговувати велику кількість одночасних підключень. Такий підхід особливо ефективний у розробці високонавантажених програмних застосунків у реальному часі, де критичними є продуктивність і швидкодія [17], [18].

У межах цієї платформи реалізовано обробку асинхронних операцій, що мінімізує затримки у взаємодії між клієнтом і сервером. Завдяки цьому Node.js добре підходить для реалізації будь-яких динамічних платформ. Крім того, можливість використання однієї мови для обох сторін — і серверної, і клієнтської — сприяє оптимізації архітектури проєкту та зменшенню розриву між частинами системи [17], [18].

Крім того, середовище Node.js забезпечує доступ до великої кількості готових пакетів, які прискорюють інтеграцію додаткових функцій без зайвого кодування. Підтримка популярних інструментів налагодження та тестування значно спрощує процес розробки, дозволяючи оперативно виявляти й усувати помилки. Також середовище Node.js легко інтегрується з хмарними рішеннями та технологіями безперервної доставки, що сприяє швидкому впровадженню оновлень [17], [18].

Підтримка мікросервісної архітектури, гнучка модульність та підтримка GraphQL і REST API дозволяють створювати масштабовані системи, адаптовані до складних бізнес-вимог. Таким чином, використання

Node.js у розробці цифрових рішень, зокрема в екологічному контексті, відкриває можливість створення надійних, швидких і зручних для користувачів сервісів із сучасною архітектурою [17], [18].

Під час створення програмного забезпечення для обговорення місцевих проблем доречно використовувати середовище Node.js, оскільки воно забезпечує оптимальні умови для такої розробки. Завдяки можливості використовувати JavaScript як на стороні сервера, так і на стороні клієнта, знижується потреба в перемиканні між різними мовами програмування, що спрощує як сам процес написання коду, так і його підтримку в командній роботі [17], [18].

Асинхронна архітектура, що лежить в основі Node.js, забезпечує ефективну обробку численних паралельних запитів без необхідності в багатопотоковому підході [17], [18].

Середовище Node.js демонструє високу ефективність у тих випадках, коли потрібно масштабувати додаток, не вдаючись до надмірних ресурсів серверної інфраструктури. Це особливо важливо для платформ, орієнтованих на обговорення актуальних питань або проведення масових дискусій, де важлива швидкість доступу й стабільність [17], [18].

Широкий набір готових інструментів, модулів і бібліотек, доступних через систему керування пакетами, сприяє стрімкому впровадженню нових функціональних можливостей. Завдяки цьому створення таких елементів, як автентифікація, зберігання повідомлень чи миттєві сповіщення, не вимагає надмірних зусиль [17], [18].

Взаємодія із сервером у режимі реального часу реалізується без потреби в складній конфігурації, зокрема, за допомогою бібліотек на кшталт Socket.IO. Саме тому Node.js ефективно справляється із завданнями, пов'язаними з обробкою подій у момент їх виникнення [17], [18].

Також важливим аспектом є підтримка взаємодії з різними базами даних, як документоорієнтованими, так і реляційними, що дозволяє зберігати великі обсяги інформації у зручному форматі. Це особливо актуально для

програм, орієнтованих на соціальні функції, де кожна дія користувача потребує швидкої та надійної фіксації [17], [18].

Крім того, можливість організувати програму у вигляді мікросервісної архітектури дозволяє ефективно розподіляти функціональність між різними частинами системи, забезпечуючи гнучкість та зменшуючи вплив змін на загальну структуру проєкту [17], [18].

Оскільки Node.js має підтримку потужної професійної спільноти, зростання проєктів супроводжується широким доступом до відкритих рішень, прикладів, фреймворків і постійного оновлення середовища [17], [18].

Обґрунтування вибору середовища розробки для створення програмного забезпечення для обговорення місцевих проблем наведено у таблиці 2.2.

Таблиця 2.2 – Обґрунтування вибору середовища виконання для створення програмного забезпечення для обговорення місцевих проблем

Критерій порівняння середовищ виконання	Середовища виконання		
	Node.js	Deno	JScript
Продуктивність	+	+—	—
Асинхронна обробка запитів	+	+—	+—
Підтримка сучасних бібліотек та фреймворків	+	+—	—
Ширина екосистеми та підтримка пакетів	+	+—	—
Масштабованість	+	+—	+—

Таким чином, для створення програмного забезпечення для обговорення місцевих проблем обрано середовище виконання Node.js, що

дозволяє використовувати JavaScript як на сервері, так і на клієнті, що значно спрощує розробку та обслуговування програмних засобів. Розробники можуть працювати в єдиному середовищі, що знижує ймовірність помилок і покращує ефективність комунікації між командами, оскільки весь код написаний на одній мові. Крім того, Node.js добре підходить для створення масштабованих додатків, оскільки він оптимізований для обробки великої кількості одночасних з'єднань без необхідності в додаткових потоках або процесах. Іншою перевагою є те, що Node.js має велику кількість доступних бібліотек та модулів, що дозволяє розробникам швидко додавати нові функціональності без необхідності розробляти все з нуля. Це особливо корисно для швидкого впровадження нових функцій, таких як підтримка реального часу, авторизація користувачів, взаємодія з базами даних та інші.

2.3 Висновки за розділом 2

Для реалізації програмного забезпечення для обговорення місцевих проблем обрано мову програмування JavaScript, яка є однією з найпоширеніших та універсальних мов у сфері веброзробки, яка дозволяє створювати як клієнтську частину, так і серверну логіку застосунку. Ще одна перевага полягає у тому, що JavaScript підтримується всіма сучасними браузерами без додаткових налаштувань, що забезпечує доступність платформи для максимальної кількості користувачів. Крім того, велика кількість відкритих бібліотек, модулів та активна спільнота розробників дозволяють швидко знаходити рішення технічних задач і зменшувати час розробки.

Для створення програмного забезпечення для обговорення місцевих проблем обрано середовище виконання Node.js, що дозволяє використовувати JavaScript як на сервері, так і на клієнті, що значно спрощує розробку та обслуговування вебсайтів. Розробники можуть працювати в єдиному середовищі, що знижує ймовірність помилок і покращує ефективність

комунікації між командами, оскільки весь код написаний на одній мові. Крім того, Node.js добре підходить для створення масштабованих додатків, оскільки він оптимізований для обробки великої кількості одночасних з'єднань без необхідності в додаткових потоках або процесах.

3 ОПИС ПРОГРАМИ

3.1 Структура програмного забезпечення для обговорення місцевих проблем

Структуру програмного забезпечення для обговорення місцевих проблем наведено на рис. 3.1.

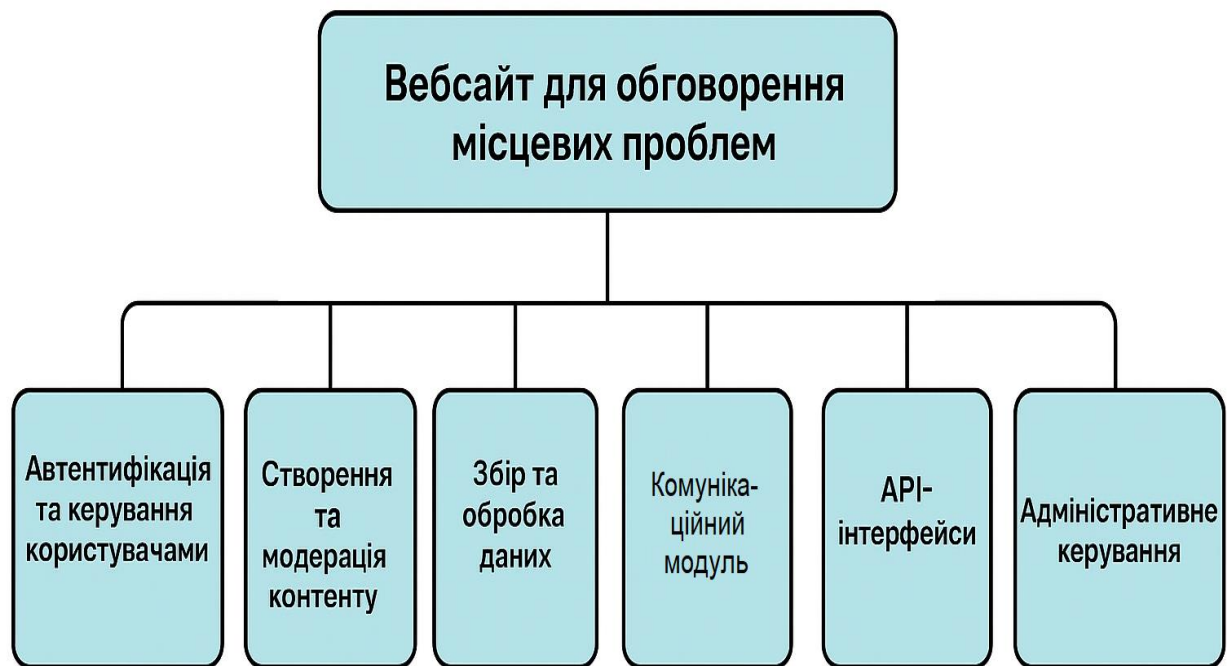


Рисунок 3.1 – Структура програмного забезпечення для обговорення місцевих проблем

Структура програмного забезпечення, орієнтованого на обговорення місцевих проблем, використовує клієнт-серверну архітектуру та передбачає поділ системи на кілька функціональних модулів (модуль автентифікації та керування користувачами, модуль створення та модерації контенту, модуль збору й обробки даних, комунікаційний модуль, модуль API-інтерфейсів, модуль керування адміністративними функціями), кожен із яких виконує окрему роль, забезпечуючи злагоджену роботу всієї платформи.

Основу системи становить модуль автентифікації та керування користувачами. Його завдання полягає у створенні, зберіганні та перевірці облікових записів, забезпеченні авторизації доступу до різних частин платформи, а також у модерації поведінки користувачів. Він обробляє реєстрацію, вхід до системи, зміну паролів і профільні налаштування.

Другим важливим елементом виступає модуль створення та модерації контенту. Цей компонент відповідає за публікацію тем для обговорення, написання, редагування та видалення повідомлень. Він також забезпечує функціональність попередньої модерації або автоматичного фільтрування контенту відповідно до встановлених правил поведінки.

Інформаційна аналітика реалізується через модуль збору й обробки даних. Він збирає статистику про активність користувачів, популярність обговорень, рівень залучення учасників. Отримані дані можуть використовуватися адміністраторами або органами місцевого самоврядування для подальшого аналізу громадських запитів і настроїв.

Комунікаційна частина платформи реалізується через комунікаційний модуль, який дозволяє користувачам брати участь у діалогах, обговореннях або групових чатах. Завдяки йому здійснюється обмін повідомленнями, надсилання сповіщень про нові відповіді чи події, а також технічна підтримка інтерактивної взаємодії.

Для підтримки гнучкої архітектури застосовується модуль API-інтерфейсів, що забезпечує зовнішню інтеграцію з іншим сервісом, що надає мапи для геоприв'язки проблем для поширення інформації.

Завершує архітектуру модуль керування адміністративними функціями, який дозволяє адміністратору контролювати налаштування системи, керувати користувачами та забезпечувати загальну підтримку працездатності програми.

Кожен із модулів взаємодіє з іншими через внутрішні сервіси або API, що дозволяє забезпечити масштабованість, стійкість до навантажень і можливість гнучкої розбудови системи відповідно до змін у потребах громади.

3.2 Функціонування програмного забезпечення для обговорення місцевих проблем

Враховуючи, що програмне забезпечення для обговорення місцевих проблем використовує клієнт-серверну архітектуру, опишемо функціонування клієнтської та серверної частин.

Функціонування клієнтської частини програмного забезпечення для обговорення місцевих проблем подамо за допомогою схеми, зображеної на рис. 3.2.

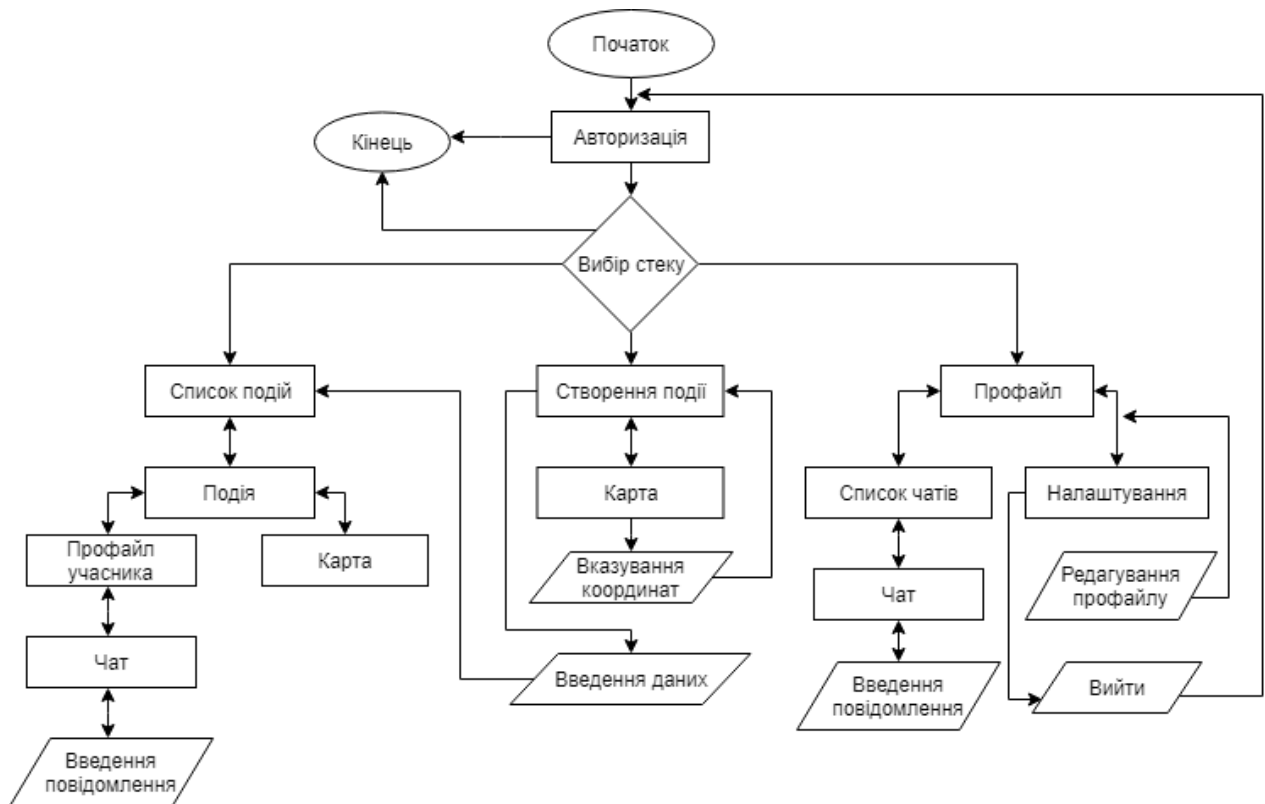


Рисунок 3.2 – Схема функціонування клієнтської частини програмного забезпечення для обговорення місцевих проблем

Функціонування клієнтської частини програмного забезпечення, що орієнтоване на обговорення локальних проблем, ґрунтується на стандартній логіці взаємодії користувача із мобільним застосунком. Основою навігації є

ключові розділи, які виступають вузлами для переходів між групами взаємопов'язаних екранів.

Початковим етапом роботи з програмним забезпеченням для обговорення місцевих проблем виступає перевірка автентичності користувача. У разі невдалої авторизації подальше використання функціоналу платформи унеможлиблюється, оскільки система вимагає обов'язкової персоніфікації. З того ж екрану, на якому відбувається вхід у систему, передбачено можливість завершення сеансу роботи шляхом натискання відповідної системної кнопки.

Після успішного входу користувач автоматично спрямовується до головного інтерфейсу програмного забезпечення для обговорення місцевих проблем, який відкриває доступ до трьох основних функціональних областей. Ці області дають змогу переглядати події, ініціювати нові обговорення та керувати персональними налаштуваннями. Усі ці секції підтримують можливість миттєвого виходу із застосунку, як і стартовий екран.

У разі вибору певної події, відкривається розширена сторінка для обговорення місцевих проблем з детальною інформацією. Звідси доступна низка дій: підписка або відписка від події, перегляд її розташування на мапі, а також перехід до профілю іншого учасника з можливістю ініціювати спілкування.

Інтерфейс для створення нової події містить важливий крок пов'язаний з вказівкою координат місця проведення через карту. Без цієї інформації публікація події неможлива. Після внесення всіх необхідних даних та підтвердження створення, користувача буде переадресовано до списку доступних подій.

Через особистий профіль відкривається доступ до списку діалогів або інтерфейсу редагування особистої інформації. У розділі повідомлень доступний перегляд усіх розмов користувача, а також наочне сповіщення про нові вхідні. Інтерфейс редагування профілю включає функцію виходу з облікового запису користувача програмного забезпечення для обговорення місцевих проблем.

Кожна сторінка, яка відкривається в межах логічного стеку, підтримує повернення до попереднього етапу, що забезпечує зручність і логічну послідовність навігації.

Функціонування серверної частини програмного забезпечення для обговорення місцевих проблем подамо за допомогою схеми, зображеної на рис. 3.3.

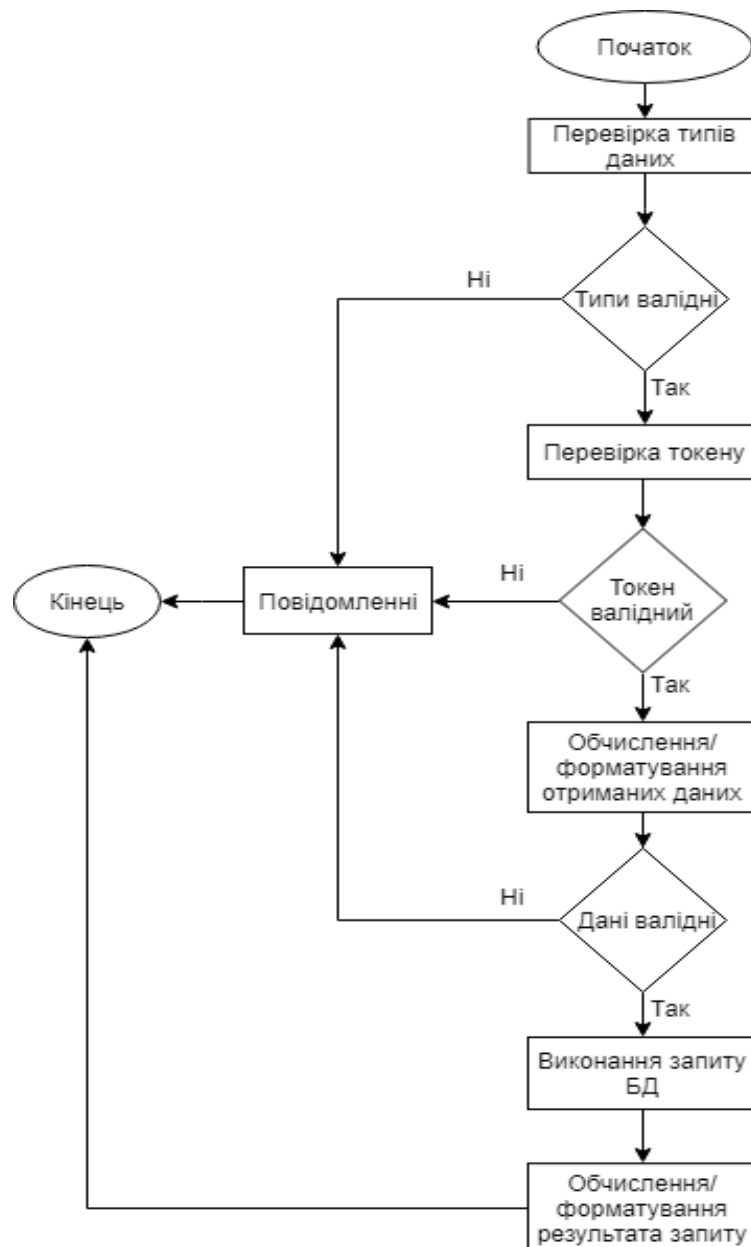


Рисунок 3.3 – Схема функціонування серверної частини програмного забезпечення для обговорення місцевих проблем

Робота серверної частини програмного забезпечення, що слугує платформою для обговорення місцевих питань, розгортається за чіткою схемою взаємодії із клієнтськими запитами. Кожен запит, що надходить із клієнтської сторони, насамперед проходить початкову перевірку – система зчитує дані, що містяться у запиті, і проводить їхню валідацію. Якщо виявляються помилки у структурі або форматі даних, або ж частина обов'язкових параметрів відсутня, процес припиняється. У відповідь клієнт отримує повідомлення про причину відмови у виконанні запиту.

У разі успішного проходження цього етапу, сервер програмного забезпечення для обговорення місцевих проблем переходить до аналізу автентифікаційного токена, який повинен підтверджувати особу користувача. Якщо токен є недійсним, простроченим або його взагалі не надано, обробка зупиняється. Користувачеві програмного забезпечення для обговорення місцевих проблем повертається відповідь із проханням повторити вхід до системи.

Якщо ж автентифікація пройшла успішно, обробник переходить до наступного етапу, пов'язаного з підготовкою даних до запиту в базу. На цьому етапі відбувається їх форматування, очищення та перевірка відповідності очікуваним структурам. Далі формується звернення до бази даних програмного забезпечення для обговорення місцевих проблем і виконується основна логіка обробки, пов'язана зі зчитуванням, записом, оновленням або видаленням інформації.

Коли база даних програмного забезпечення для обговорення місцевих проблем повертає відповідь, вона надходить до відповідального контролера, який аналізує отримані дані. Також можуть бути проведені додаткові обчислення, трансформації або агрегації даних. Після цього сформована відповідь відправляється на клієнтську сторону програмного забезпечення для обговорення місцевих проблем, завершуючи життєвий цикл конкретного запиту.

3.3 Розробка бази даних програмного забезпечення для обговорення місцевих проблем

База даних програмного забезпечення для обговорення місцевих проблем виступає центральним елементом зберігання, організації та взаємодії з усіма ключовими об'єктами системи. База даних ПЗ для обговорення місцевих проблем забезпечує зберігання даних про користувачів системи, події, повідомлення та обговорення.

Схему бази даних програмного забезпечення для обговорення місцевих проблем наведено на рис. 3.4, де позначення РК означає первинний ключ (Primary Key).

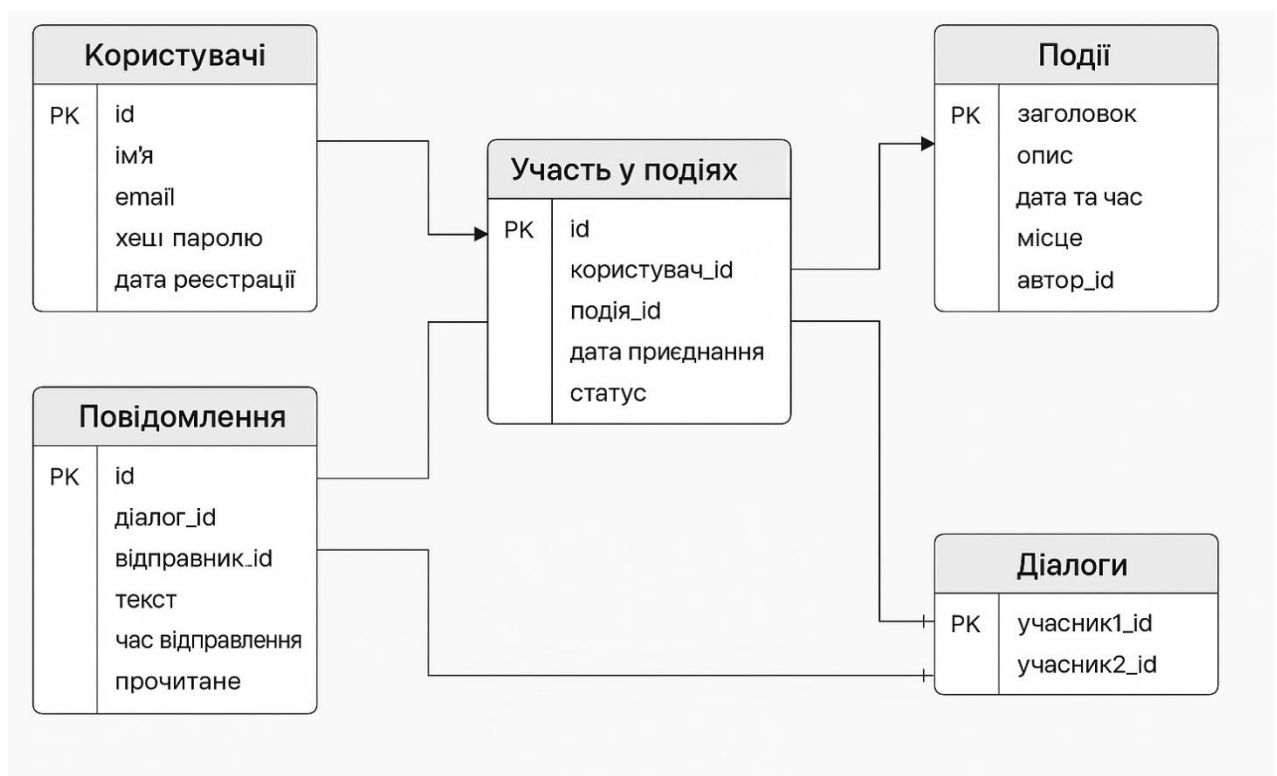


Рисунок 3.4 – Схема бази даних програмного забезпечення для обговорення місцевих проблем

База даних програмного забезпечення для обговорення місцевих проблем побудована навколо кількох ключових сутностей, які відображають основні процеси взаємодії користувачів із системою. Центральне місце займає

сутність користувачів, що містить всю необхідну інформацію про осіб, зареєстрованих у системі: від особистих даних до параметрів авторизації та налаштувань профілю. Кожен користувач може створювати або долучатися до подій, які представлені у вигляді окремої сутності. Події містять відомості про місце, час, опис, а також пов'язані з організатором — тобто автором, що створив подію.

Інша важлива таблиця бази даних програмного забезпечення для обговорення місцевих проблем – таблиця подій. Тут зберігається інформація про локальні події, які створюють користувачі: назва події, опис, час проведення, географічні координати, автор події, список учасників тощо. Ця таблиця зв'язана із таблицею участі, яка фіксує, хто з користувачів підписаний на яку подію.

Взаємозв'язок між користувачами та подіями реалізується через проміжну сутність, яка фіксує участь у подіях. Ця таблиця дозволяє зберігати інформацію про те, хто саме приєднався до певної події, коли саме це сталося, який статус участі на даний момент і чи змінювався він з часом. Таким чином забезпечується багатосторонній облік взаємодії користувачів з активностями на платформі.

Комунікація між користувачами реалізована через систему діалогів, яка підтримує приватне листування. Кожен діалог поєднує двох учасників, а повідомлення, які належать до конкретного діалогу, записуються у відповідній сутності. Повідомлення містять текст, час надсилання, відправника та стан прочитаності. Через ці дві сутності реалізується логіка обміну інформацією, інформування про нові повідомлення та формування історії спілкування.

У результаті база даних програмного забезпечення для обговорення місцевих проблем є гнучким інструментом, що забезпечує швидкий обмін інформацією між усіма компонентами програмного забезпечення і гарантує стабільність у зберіганні та доступі до критично важливої інформації програми.

3.4 Проєктування інтерфейсу програмного забезпечення для обговорення місцевих проблем

Проєктування інтерфейсу взаємодії користувача з програмним забезпеченням для обговорення місцевих проблем виконано з урахуванням вимог технічного завдання. При розробленні програмного забезпечення для обговорення місцевих проблем враховані функціональні вимоги до програми:

- підтримка можливості відображення переліку місцевих подій та проблем;
- підтримка можливості авторизації та реєстрації;
- підтримка можливості обговорення місцевих подій шляхом текстового спілкування з іншими користувачами;
- можливість додавання інформації про нову локальну подію або проблему;
- можливість додавання інформації про географічне місце проведення події;
- підтримка можливості роботи з екраном діалогів користувача;
- підтримка можливості зміни налаштувань профілю користувача.

Інтерфейс головної форми програмного забезпечення для обговорення місцевих проблем наведено на рис. 3.5.

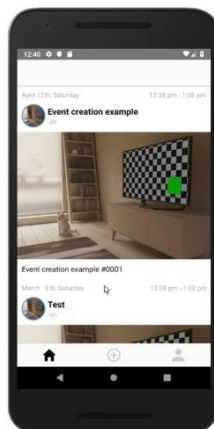


Рисунок 3.5 – Основний екран ПЗ для обговорення місцевих проблем

Екрани авторизації та реєстрації програмного забезпечення для обговорення місцевих проблем наведено на рис. 3.6 та рис. 3.7.

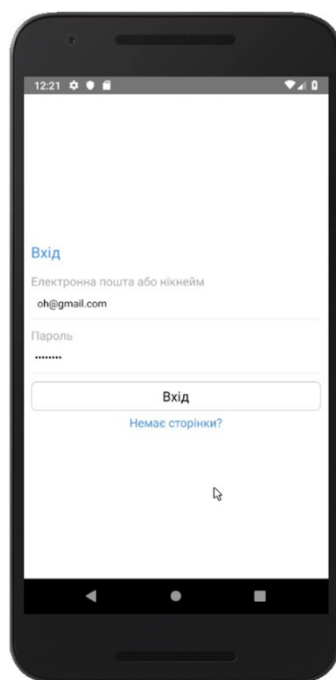


Рисунок 3.6 – Екран авторизації програмного забезпечення для обговорення місцевих проблем

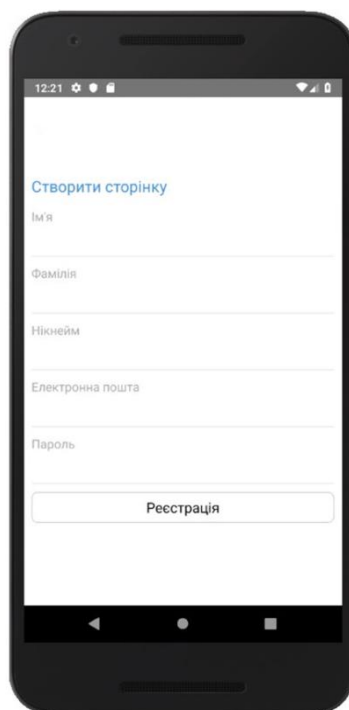


Рисунок 3.7 – Екран реєстрації програмного забезпечення для обговорення місцевих проблем

Екран додавання інформації про нову локальну подію або проблему наведено на рис. 3.8. Екран сторінки користувача наведено на рис. 3.9

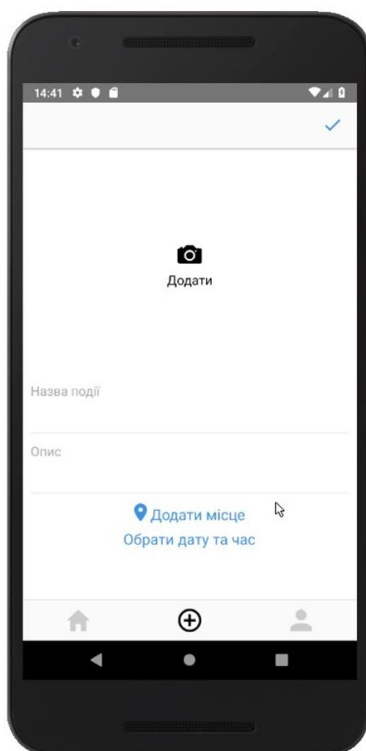


Рисунок 3.8 – Додавання інформації про нову локальну подію або проблему

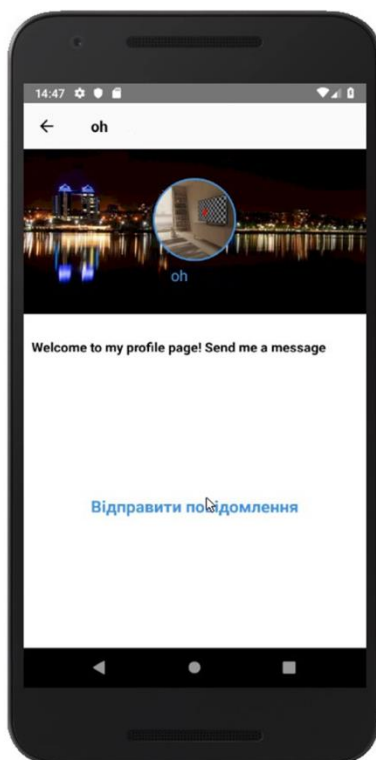


Рисунок 3.9 – Сторінка користувача

Екран спілкування з іншим користувачем наведено на рис. 3.10.

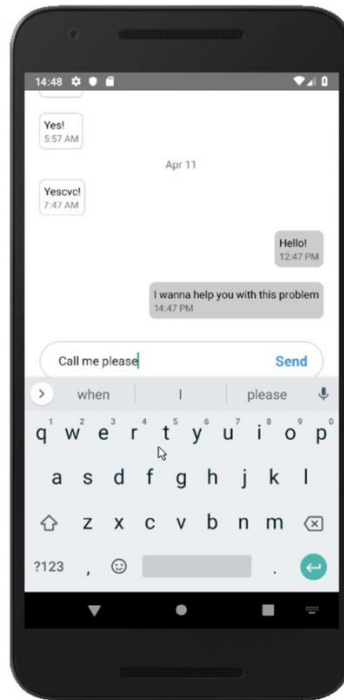


Рисунок 3.10 – Екран спілкування з іншим користувачем

Екран сторінки налаштувань профілю користувача наведено на рис. 3.11.

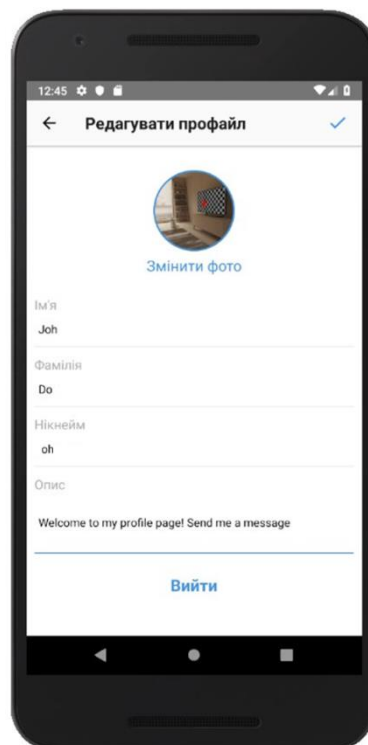


Рисунок 3.11 – Екран сторінки налаштувань профілю користувача

3.5 Висновки за розділом 3

Розроблено програмне забезпечення для обговорення місцевих проблем.

Запропоновано структуру програмного забезпечення для обговорення місцевих проблем. Визначено, що структура програмного забезпечення, орієнтованого на обговорення місцевих проблем, використовує клієнт-серверну архітектуру та передбачає поділ системи на кілька функціональних модулів (модуль автентифікації та керування користувачами, модуль створення та модерації контенту, модуль збору й обробки даних, комунікаційний модуль, модуль API-інтерфейсів, модуль керування адміністративними функціями), кожен із яких виконує окрему роль, забезпечуючи злагоджену роботу всієї платформи.

Розроблено базу даних програмного забезпечення для обговорення місцевих проблем, яка виступає центральним елементом зберігання, організації та взаємодії з усіма ключовими об'єктами системи. База даних ПЗ для обговорення місцевих проблем забезпечує зберігання даних про користувачів системи, події, повідомлення та обговорення.

Описано функціонування програмного забезпечення для обговорення місцевих проблем.

Описано особливості реалізації програмного забезпечення для обговорення місцевих проблем. Виконано проєктування інтерфейсу взаємодії користувача з програмним забезпеченням для обговорення місцевих проблем.

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

4.1 Призначення й умови застосування програми

Програма призначена для обговорення місцевих проблем. Програма написана на мові JavaScript, яка є однією з найпоширеніших та універсальних мов у сфері веброзробки, яка дозволяє створювати як клієнтську частину, так і серверну логіку програми. В якості середовища виконання для створення програмного забезпечення для обговорення місцевих проблем обрано Node.js, що дозволяє використовувати JavaScript як на сервері, так і на клієнті, що значно спрощує розробку та обслуговування програмних засобів.

Програмне забезпечення для обговорення місцевих проблем забезпечує виконання таких функцій:

- підтримка можливості відображення переліку місцевих подій та проблем;
- підтримка можливості авторизації та реєстрації;
- підтримка можливості обговорення місцевих подій шляхом текстового спілкування з іншими користувачами;
- можливість додавання інформації про нову локальну подію або проблему;
- можливість додавання інформації про географічне місце проведення події;
- підтримка можливості роботи з екраном діалогів користувача;
- підтримка можливості зміни налаштувань профілю користувача.

Для роботи програмного забезпечення для обговорення місцевих проблем потрібні такі технічні та програмні ресурси. На серверній стороні необхідна машина з процесором не нижче ніж 4-ядерний з тактовою частотою 2.4 ГГц, що забезпечить стабільну обробку багатопотокових запитів користувачів. Об'єм оперативної пам'яті повинен бути не менше 8 ГБ, що дозволить ефективно працювати з базою даних, кешем та обробкою логіки

програми. На жорсткому диску має бути передбачено щонайменше 100 ГБ вільного місця. З програмного забезпечення необхідна сучасна операційна система, наприклад, Ubuntu Server 22.04 LTS або Windows Server 2019. Для запуску серверної частини застосунку потрібне встановлене середовище виконання Node.js, вебсервер Nginx або Apache, а також система керування базами даних. На стороні клієнта достатньо мобільного телефона з актуальною версією веббраузера (Google Chrome, Mozilla Firefox, Safari, Microsoft Edge), доступом до мережі Інтернет, операційною системою Android 8.0+ або iOS 13+, процесором на рівні 4-ядерного 1.8 ГГц, щонайменше 2 ГБ оперативної пам'яті, а також 50 МБ вільного простору для встановлення застосунку.

Функціональні характеристики розробленого програмного забезпечення для обговорення місцевих проблем наведено у технічному завданні (додаток А). Фрагмент тексту програмного забезпечення для обговорення місцевих проблем наведено у додатку Б.

4.2 Характеристики програми для обговорення місцевих проблем

Програмне забезпечення для обговорення місцевих проблем використовує клієнт-серверну архітектуру та передбачає поділ системи на кілька функціональних модулів, зокрема, модуль автентифікації та керування користувачами, модуль створення та модерації контенту, модуль збору й обробки даних, комунікаційний модуль, модуль API-інтерфейсів, модуль керування адміністративними функціями, кожен з цих модулів виконує окрему роль, забезпечуючи злагоджену роботу всієї платформи.

Програмне забезпечення для обговорення місцевих проблем має зручний вебінтерфейс, адаптований до різних типів пристроїв, що гарантує комфортну взаємодію незалежно від платформи. Архітектура системи орієнтована на модульність, що дозволяє ефективно масштабувати функціональність та здійснювати оновлення без зупинки всієї системи.

Програмне забезпечення для обговорення місцевих проблем використовує базу даних, яка виступає центральним елементом зберігання, організації та взаємодії з усіма ключовими об'єктами системи. База даних для обговорення місцевих проблем забезпечує зберігання даних про користувачів системи, події, повідомлення та обговорення.

Система підтримує обробку запитів у реальному часі, що дає змогу оперативно оновлювати інформацію та надсилати сповіщення. Завдяки використанню серверної логіки із захистом авторизаційних токенів забезпечується надійність і стабільність роботи програми.

Інтерфейс програмного забезпечення для обговорення місцевих проблем передбачає наявність профілів користувачів, списку подій, можливості створення нових подій, ведення діалогів та участі в обговореннях. Додатково реалізовано геолокаційну інтеграцію, яка дозволяє вказувати точне місце проведення події. Уся інформація зберігається в структурованій базі даних із підтримкою цілісності та зв'язків між сутностями.

4.3 Інструкція по експлуатації програми

4.3.1 Звернення до програми

Для звертання до клієнтської сторони необхідно перейти на відповідну вебсторінку.

4.3.2 Вхідні й вихідні дані

Вхідними даними до програмного забезпечення для обговорення місцевих проблем є інформація, яку користувачі вводять під час взаємодії із системою. До таких даних належать облікові дані під час реєстрації або авторизації, відомості для створення події, включно з назвою, описом, місцем проведення, а також координатами на мапі.

Вихідними даними програмного забезпечення для обговорення місцевих проблем є інформація, яка відображається користувачам після обробки запитів системою.

4.3.3 Повідомлення

Користувач програмного забезпечення для обговорення місцевих проблем може отримати такі повідомлення: повідомлення про результати авторизації, результати створення події та ін.

4.4 Виконання програмного забезпечення для обговорення місцевих проблем

Після запуску програмного забезпечення для обговорення місцевих проблем користувачу відображається екран авторизації (рис. 4.1), де необхідно ввести свою пошту та пароль.

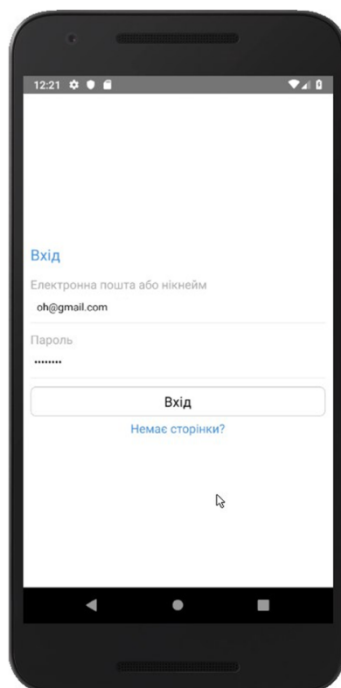


Рисунок 4.1 – Інтерфейс екрану авторизації програмного забезпечення для обговорення місцевих проблем

Незареєстрований користувач повинен пройти процедуру реєстрації (рис. 4.2), натиснувши попередньо посилання Немає сторінки.

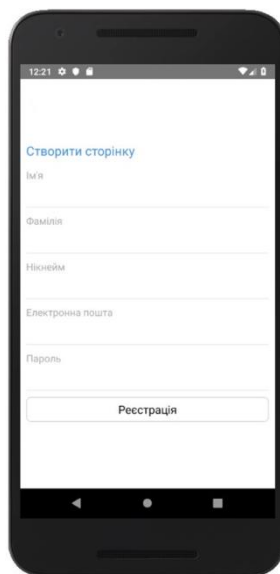


Рисунок 4.2 – Інтерфейс екрану реєстрації програмного забезпечення для обговорення місцевих проблем

Після входу до системи відображається основний екран (рис. 4.3), де подається інформація про місцеві події та проблеми.

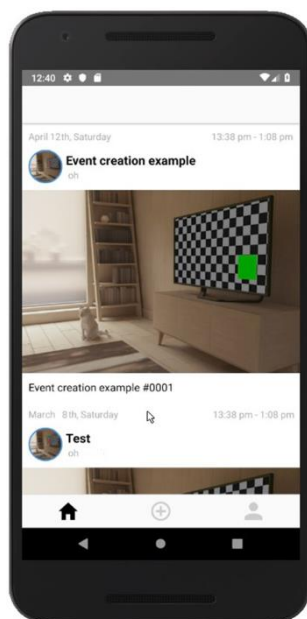


Рисунок 4.3 – Основний екран програмного забезпечення для обговорення місцевих проблем

Для додавання інформації про нову місцеву подію або проблему необхідно натиснути кнопку зі знаком плюс на головному екрані та ввести потрібну інформацію (рис. 4.4), зокрема, додати фото, назву та опис події, а також обрати дату, час та місце її проведення.

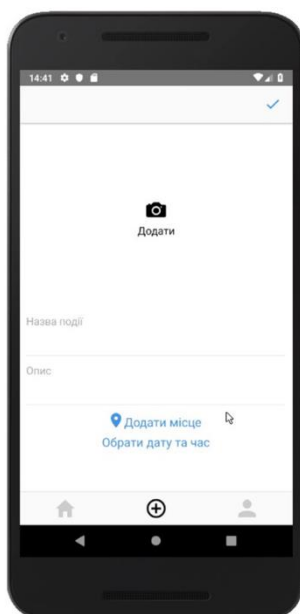


Рисунок 4.4 – Додавання інформації про нову локальну подію або проблему

На рис. 4.5 наведено додавання місця події.

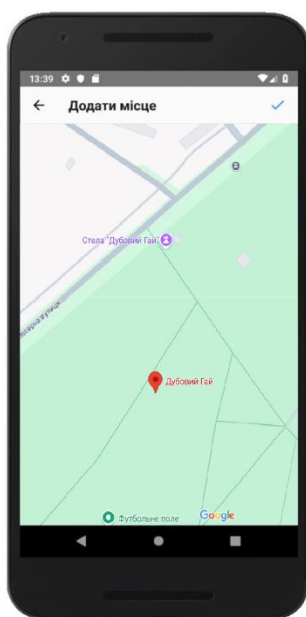


Рисунок 4.5 – Екран з додаванням географічного місця проведення події

Після цього подію буде створено та додано до списку подій. Обравши відповідну подію відобразиться екран обраної події (рис. 4.6) з детальною інформацією про неї.

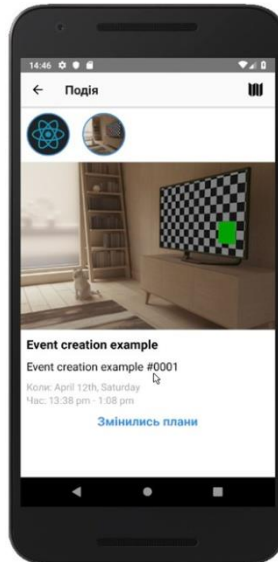


Рисунок 4.6 – Екран події

На екрані кожної події згори наведено стрічку, де відображено зображення користувачів, які планують брати участь у події. При натисканні на зображення користувача відбувається перехід до сторінки з інформацією про нього (рис. 4.7).

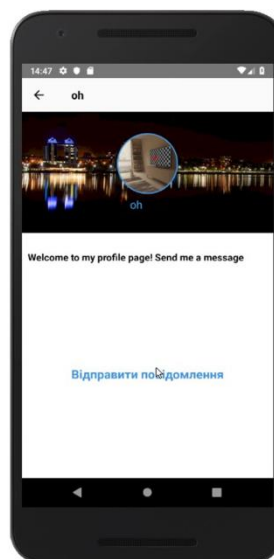


Рисунок 4.7 – Сторінка користувача

Для обговорення місцевих подій програма надає можливість спілкування з іншими користувачами. Для цього на сторінці іншого користувача необхідно перейти за посиланням Відправити повідомлення. Внаслідок цього відбудеться перехід до екрану спілкування з цим користувачем (рис. 4.8).

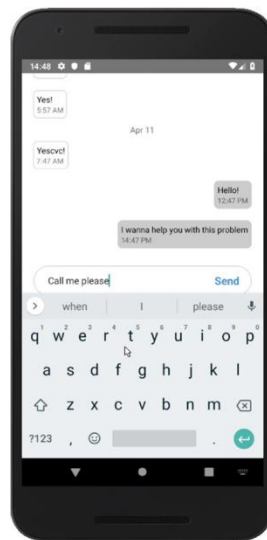


Рисунок 4.8 – Екран спілкування з іншим користувачем

На власній сторінці користувач може перейти до списку діалогів та обрати користувача, з яким необхідно обговорити певну подію (рис. 4.9).

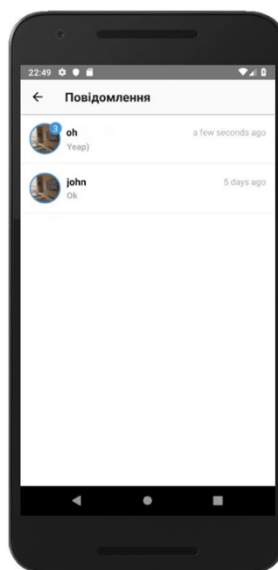


Рисунок 4.9 – Екран з переліком діалогів користувача

На власній сторінці користувач може перейти до екрану налаштувань, де можна змінити персональні дані (рис. 4.10).

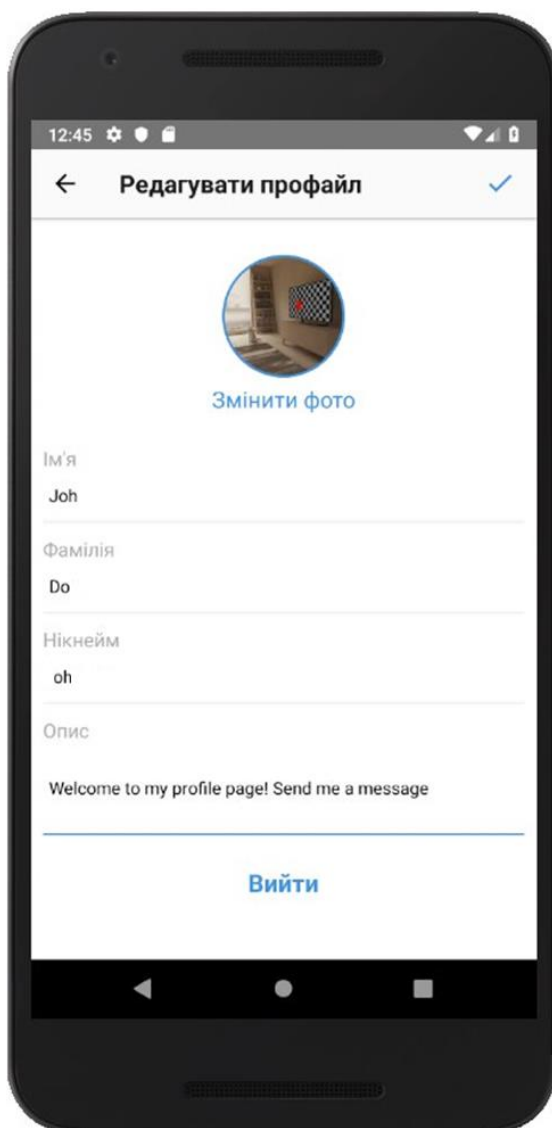


Рисунок 4.10 – Сторінка налаштувань профілю користувача

4.5 Тестування програмного забезпечення для обговорення місцевих проблем

Виконано тестування програмного забезпечення для обговорення місцевих проблем.

Виявлено, що програма для обговорення місцевих проблем функціонує правильно та злагоджено. Вона реалізує всі функціональні вимоги та успішно

виконує свою основну задачу підтримки процесу обговорення проблем регіонального характеру.

Розроблене програмне забезпечення дозволяє забезпечувати автоматизацію процесів, пов'язаних з підтримкою процесу обговорення місцевих проблем.

4.6 Висновки за розділом 4

Описано програмне забезпечення для обговорення місцевих проблем. Виконано тестування розробленого програмного забезпечення для обговорення місцевих проблем. Результати тестування програмного забезпечення показали, що розроблена програма дозволяє забезпечити автоматизацію процесів, пов'язаних з підтримкою процесу обговорення місцевих проблем.

ВИСНОВКИ

В ході виконання дипломної кваліфікаційної роботи бакалавра було проаналізовано та досліджено процес розробки програмного забезпечення для обговорення регіональних питань.

Визначено, що програмне забезпечення для обговорення місцевих проблем призначене для створення інтерактивного середовища, де мешканці певної громади або території можуть спільно висловлювати думки, обговорювати нагальні питання, брати участь у подіях, ініціювати зміни та взаємодіяти з місцевою владою або активістами. Основна мета такого програмного забезпечення полягає в тому, щоб надати ефективний інструмент комунікації для вирішення проблем місцевого значення, формування суспільної думки, підвищення громадської активності та прозорості у прийнятті рішень. У рамках програмного забезпечення для обговорення місцевих проблем реалізується можливість створення обговорень, публікацій, організації подій, збору пропозицій, проведення опитувань та залучення користувачів до участі в діалогах.

За результатами проведеного аналізу зроблено висновок, що у наш час існує досить багато програмних засобів для обговорення місцевих проблем. Проте деякі програмні засоби для обговорення місцевих проблем мають низку недоліків, які можуть впливати на їхню ефективність та популярність серед користувачів. Основними проблемами таких програмних засобів є нестабільна робота, поганий інтерфейс або обмежені функціональні можливості, що ускладнюють користування ресурсом. Певні труднощі може спричинити і недостатній рівень захисту персональних даних. Тому актуальною є розробка програмного забезпечення для обговорення місцевих проблем.

Сформульовано функціональні вимоги до програмного забезпечення для обговорення місцевих проблем.

Для реалізації програмного забезпечення для обговорення місцевих проблем обрано мову програмування JavaScript, яка є однією з

найпоширеніших та універсальних мов у сфері веброзробки, яка дозволяє створювати як клієнтську частину, так і серверну логіку програми. Ще одна перевага полягає у тому, що JavaScript підтримується всіма сучасними браузерами без додаткових налаштувань, що забезпечує доступність платформи для максимальної кількості користувачів. Крім того, велика кількість відкритих бібліотек, модулів та активна спільнота розробників дозволяють швидко знаходити рішення технічних задач і зменшувати час розробки.

Для створення програмного забезпечення для обговорення місцевих проблем обрано середовище виконання Node.js, що дозволяє використовувати JavaScript як на сервері, так і на клієнті, що значно спрощує розробку та обслуговування програм. Розробники можуть працювати в єдиному середовищі, що знижує ймовірність помилок і покращує ефективність комунікації між командами, оскільки весь код написаний на одній мові. Крім того, Node.js добре підходить для створення масштабованих додатків, оскільки він оптимізований для обробки великої кількості одночасних з'єднань без необхідності в додаткових потоках або процесах.

Розроблено програмне забезпечення для обговорення місцевих проблем.

Запропоновано структуру програмного забезпечення для обговорення місцевих проблем. Визначено, що структура програмного забезпечення, орієнтованого на обговорення місцевих проблем, використовує клієнт-серверну архітектуру та передбачає поділ системи на кілька функціональних модулів (модуль автентифікації та керування користувачами, модуль створення та модерації контенту, модуль збору й обробки даних, комунікаційний модуль, модуль API-інтерфейсів, модуль керування адміністративними функціями), кожен із яких виконує окрему роль, забезпечуючи злагоджену роботу всієї платформи.

Розроблено базу даних програмного забезпечення для обговорення місцевих проблем, яка виступає центральним елементом зберігання,

організації та взаємодії з усіма ключовими об'єктами системи. База даних ПЗ для обговорення місцевих проблем забезпечує зберігання даних про користувачів системи, події, повідомлення та обговорення.

Описано функціонування програмного забезпечення для обговорення місцевих проблем.

Описано особливості реалізації програмного забезпечення для обговорення місцевих проблем. Виконано проєктування інтерфейсу взаємодії користувача з програмним забезпеченням для обговорення місцевих проблем.

Виконано тестування розробленого програмного забезпечення для обговорення місцевих проблем. Результати тестування програмного забезпечення показали, що розроблена програма дозволяє забезпечити автоматизацію процесів, пов'язаних з підтримкою процесу обговорення місцевих проблем.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. How to Create Your Own Community App? The Full Guide [Electronic resource]. – Access mode: <https://nandbox.com/how-to-create-your-own-community-app-the-full-guide/>.
2. How to build a community app [Electronic resource]. – Access mode: <https://www.disciple.community/blog/build-a-community-app>.
3. A systematic analysis of digital tools for citizen participation [Electronic resource]. – Access mode: <https://www.sciencedirect.com/science/article/pii/S0740624X24000467>.
4. Creating Software: 10 Steps from Concept to Launch [Electronic resource]. – Access mode: <https://lampa.dev/blog/creating-software-10-steps-from-concept-to-launch>.
5. Best Forum Software for Online Communities [Electronic resource]. – Access mode: <https://www.creatuforo.com/best-forum-software-2/>.
6. How to Choose the Best Community Engagement Platform [Electronic resource]. – Access mode: <https://www.disco.co/blog/the-best-community-engagement-platforms>.
7. Best Online Discussion Tools [Electronic resource]. – Access mode: <https://www.questionpro.com/blog/online-discussion-tools/>.
8. Best Community Platforms [Electronic resource]. – Access mode: <https://www.linodash.com/community-platforms/>.
9. Best Free Community Software [Electronic resource]. – Access mode: <https://www.softwareworld.co/free-community-software/>.
10. Best Forum Software for Online Communities [Electronic resource]. – Access mode: <https://www.wix.com/blog/best-forum-software>.
11. Best Forum Software [Electronic resource]. – Access mode: <https://www.creatuforo.com/best-forum-software-2/>.
12. Recollective [Electronic resource]. – Access mode: <https://www.recollective.com/>.

13. FlexMR [Electronic resource]. – Access mode: <https://www.flexmr.net/>.
14. Vocal Views [Electronic resource]. – Access mode: <https://vocalviews.com/>.
15. JavaScript Tutorial [Electronic resource]. – Access mode: <https://www.javascripttutorial.net/>.
16. JavaScript. Attributes and properties [Electronic resource]. – Access mode: <https://javascript.info/dom-attributes-and-properties>.
17. Node.js [Electronic resource]. – Access mode: <https://dev.to/t/node>.
18. Introduction to Node.js [Electronic resource]. – Access mode: <https://nodejs.org/en/learn/getting-started/introduction-to-nodejs>.

ДОДАТОК А
Технічне завдання

Вступ

Програмне забезпечення може використовуватися для обговорення місцевих проблем.

A.1 Підстава для розробки

Підставою для розробки є завдання на дипломну кваліфікаційну роботу на тему «Програмне забезпечення для обговорення місцевих проблем», затверджене наказом Національного університету «Запорізька політехніка» № 209 від 28 квітня 2025 р.

A.2 Призначення розробки

Програмний продукт призначений для обговорення місцевих проблем.

A.3 Основні вимоги до програми, що розробляється

A.3.1 Вимоги до функціональних характеристик

Програмне забезпечення для обговорення місцевих проблем забезпечує виконання таких функцій:

- підтримка можливості відображення переліку місцевих подій та проблем;
- підтримка можливості авторизації та реєстрації;
- підтримка можливості обговорення місцевих подій шляхом текстового спілкування з іншими користувачами;
- можливість додавання інформації про нову локальну подію або проблему;
- можливість додавання інформації про географічне місце проведення події;

- підтримка можливості роботи з екраном діалогів користувача;
- підтримка можливості зміни налаштувань профілю користувача.

А.3.2 Вимоги до інтерфейсу програми

Інтерфейс програмного забезпечення для обговорення місцевих проблем повинен бути зручним для користувачів. Повинна бути забезпечена можливість роботи в візуальному режимі.

А.3.3 Вимоги до надійності

Програмне забезпечення для обговорення місцевих проблем повинно забезпечити надійне функціонування.

А.3.4 Умови експлуатації

Для експлуатації програмного забезпечення для обговорення місцевих проблем необхідна наявність персонального комп'ютера.

А.3.5 Вимоги до складу та параметрів технічних засобів

Для роботи програмного забезпечення для обговорення місцевих проблем потрібні такі технічні та програмні ресурси. На серверній стороні необхідна машина з процесором не нижче ніж 4-ядерний з тактовою частотою 2.4 ГГц, що забезпечить стабільну обробку багатопотокових запитів користувачів. Об'єм оперативної пам'яті повинен бути не менше 8 ГБ, що дозволить ефективно працювати з базою даних, кешем та обробкою логіки програми. На жорсткому диску має бути передбачено щонайменше 100 ГБ

вільного місця. З програмного забезпечення необхідна сучасна операційна система, наприклад, Ubuntu Server 22.04 LTS або Windows Server 2019. Для запуску серверної частини застосунку потрібне встановлене середовище виконання Node.js, вебсервер Nginx або Apache, а також система керування базами даних. На стороні клієнта достатньо мобільного телефона з актуальною версією веббраузера (Google Chrome, Mozilla Firefox, Safari, Microsoft Edge), доступом до мережі Інтернет, операційною системою Android 8.0+ або iOS 13+, процесором на рівні 4-ядерного 1.8 ГГц, щонайменше 2 ГБ оперативної пам'яті, а також 50 МБ вільного простору для встановлення застосунку.

А.3.6 Вимоги до маркування і пакування

Програма для обговорення місцевих проблем може бути записана на будь-якому носії інформації.

На пакуванні повинна бути назва програми – «Програмне забезпечення для обговорення місцевих проблем».

А.4 Вхідні дані до роботи

Вхідними даними до програмного забезпечення для обговорення місцевих проблем є інформація, яку користувачі вводять під час взаємодії із системою. До таких даних належать облікові дані під час реєстрації або авторизації, відомості для створення події, включно з назвою, описом, місцем проведення, а також координатами на мапі.

Вихідними даними програмного забезпечення для обговорення місцевих проблем є інформація, яка відображається користувачам після обробки запитів системою.

ДОДАТОК Б
Фрагмент тексту програми

```

function confgrClnt() {
  const kesh = new InMemoryKesh();
  const defaultOptns = {
    watchZapyt: {
      ftchPolicy: 'network-only',
      errPolicy: 'ignore',
    },
    zapyt: {
      ftchPolicy: 'network-only',
      errPolicy: 'all',
    },
  };

  const errLink = onErr(({ graphQLErrs, networkErr }) => {
    if (graphQLErrs)
      graphQLErrs.map(async ({ msg, lktns, shlh, extnsns }) => {
        console.log(
          `[GraphQL err]: Msg: ${msg}, Lktn: ${lktns}, Shlh:
${shlh}`
        );

        if (extnsns.code === 'UNAUTHENTICATED') {
          console.log(await
AsncStorage.getItem('@SessionStorage:accessTkn'));
        }
      });

    if (networkErr) console.log(`[Network err]:
${networkErr}`);
  });

  const httpLnk = createUploadLink({ urhi: URLS.API_URL });

  const authLnk = setCntx(async () => {
    const tkn = await
AsncStorage.getItem('@SessionStorage:accessTkn');

    let cntx = {};

    if (tkn) {
      cntx = {
        headers: {
          'x-tkn': tkn
        }
      };
    }
  });
}

```

```

        return cntx;
    });

    const wsLnk = new WebSocketLink({
        urhi: URLS.WS_URL,
        optns: {
            rcnct: true,
            connectionParams: async () => ({ "x-tkn": await
AsncStorage.getItem('@SessionStorage:accessTkn') })
        },
    });

    const termintngLink = split(
        ({ zapyt }) => {
            const { kynd, oprtn } = getMainDefinition(zapyt);
            return (
                kynd === 'OprtnDefinition' && oprtn === 'subscrptn'
            );
        },
        wsLnk,
        authLnk.concat(httpLnk),
    );

    return new ApolloClnt({
        kesh,
        defaultOptns,
        link: ApolloLink.from([
            errLink,
            termintngLink
        ])
    });
}

interface IPrps {
    navitn: any;

    polzvttl: IPolzvttl;

    spilkuvn: ISpilkuvn;
    spilkuvnLoading: boolean;
    spilkuvnErr: string;

    msgsg: IMsg[];
    msgLoading: boolean;
    msgErr: string;

    getSpilkuvn(id: string): void;

```

```

    cleanSpilkuvn(): void;
    getMsgs(): void;
    cleanMsgs(): void;
    readMsgs(msgsId: string[]): void;
    sendMsg(msg: string): void;
    subscrbedMsg(msg: IMsg): void;
}

interface IState {
    msg: string;
}

class SpilkuvnScreen extends Kmpnt<IPrps, IState> {
    constructor(prps: IPrps) {
        super(prps);

        this.state = {
            msg: "",
        };

        this.subscrprionToMsgs = null;
    }

    kmpntDidMount(): void {
        const {
            getSpilkuvn,
            spilkuvn: {
                id: spilkuvnId
            },
            navitn
        } = this.prps;
        const id = navitn.getParam("spilkuvnId");
        let subscrptnId;

        if (id) {
            subscrptnId = id;
            getSpilkuvn(id);
        } else {
            subscrptnId = spilkuvnId;
        }

        this.subscrprionToMsgs = MsgRslvr.subscribeMsgs(subscrptnId,
this.subscrbedMsg);
    }

    kmpntDidUpdate(prevPrps: Readonly<IPrps>): void {
        const { msgs, readMsgs } = this.prps;

```

```

    if (msgs.length !== prevPrps.msgs.length &&
    _.some(prevPrps.msgs)) {
      const msg = msgs[0];

      if (!msg.read) {
        readMsgs([msg.id]);
      }
    }
  }

  kmpntWillUnmount(): void {
    const { cleanMsgs } = this.prps;

    if (this.subscrprionToMsgs) {
      this.subscrprionToMsgs.unsubscribe();
    }

    cleanMsgs();
  }

  readMsg() {
    const { msgs, readMsgs } = this.prps;

    const unreadMsgsId = _.reduce(msgs, (result: string[], msg:
    IMsg) => {
      if (!msg.read) {
        result.push(msg.id);
      }
      return result;
    }, []);

    if (_.some(unreadMsgsId)) {
      readMsgs(unreadMsgsId);
    }
  }

  onFcsInput = () => {
    this.readMsg();
  };

  onChngInput = (msg: string) => {
    this.setState({ msg });
  };

  submitMsg = () => {
    const { sendMsg } = this.prps;
    const { msg } = this.state;

```

```

    this.setState({ msg: "" });

    sendMsg(msg);
  };

  subscrbedMsg = (msg: IMsg) => {
    const { subscrbedMsg } = this.prps;

    subscrbedMsg(msg);
  };

  const mapStateToPrps = (state: IReducerStates) => {
    const { polzvttl } = state.polzvtlProfileReducer;
    const { spilkuvn, loading: spilkuvnLoading, err: spilkuvnErr } =
    state.spilkuvnReducer;
    const { msgs, loading: msgLoading, err: msgErr } =
    state.msgReducer;
    return {
      polzvttl,

      spilkuvn,
      spilkuvnLoading,
      spilkuvnErr,

      msgs,
      msgLoading,
      msgErr
    }
  };

  const mapDispatchToPrps = (dispatch: Dispatch) => {
    return {
      getSpilkuvn: (id: string) => dispatch(getSpilkuvn(id)),
      cleanSpilkuvn: () => dispatch(cleanSpilkuvn()),
      getMsgs: () => dispatch(getMsgs()),
      cleanMsgs: () => dispatch(cleanMsgs()),
      sendMsg: (msg: string) => dispatch(sendMsg(msg)),
      readMsgs: (msgsId: string[]) => dispatch(readMsgs(msgsId)),
      subscrbedMsg: (msg: IMsg) => dispatch(subscrbedMsg(msg)),
    }
  };

  export default connect(mapStateToPrps,
    mapDispatchToPrps)(SpilkuvnScreen);
  export function* createPrivateSpilkuvn(action: any) {
    try {
      const { rpntId } = action;

```

```

    yield put({ type: CRTPRVT_SPILKUVN_REQUESTING });
    const spilkuvn = yield
call(SpilkuvnRslvr.createPrivateSpilkuvn, rpntId );

    if (spilkuvn) {
        yield put({ type: CRTPRVT_SPILKUVN_SUCCESS, spilkuvn });
        yield put({ type: GET_MSGS });
        nav.navigate('SpilkuvnScreen');
        return spilkuvn;
    }
    return null;

} catch (err) {
    yield put({ type: CRTPRVT_SPILKUVN_ERR, err });
    console.warn('Err create private spilkuvn', err);
}
}

export function* getSpilkuvn(action: any) {
    try {
        const { id } = action;

        yield put({ type: GET_SPILKUVN_REQUESTING });
        const spilkuvn = yield call(SpilkuvnRslvr.getSpilkuvn, id);

        if (spilkuvn) {
            yield put({ type: GET_SPILKUVN_SUCCESS, spilkuvn });
            yield put({ type: GET_MSGS });

            return spilkuvn;
        }

        return null;

    } catch (err) {
        yield put({ type: GET_SPILKUVN_ERR, err });
        console.warn('Err get spilkuvn', err);
    }
}

export function* polzvttlSpilkuvns() {
    try {

        yield put({ type: POLZVTL_SPILKUVNS_REQUESTING });
        const polzvttlSpilkuvns = yield
call(SpilkuvnRslvr.polzvttlSpilkuvns);

        if (polzvttlSpilkuvns) {

```

```

        yield put({ type: POLZVTL_SPILKUVNS_SUCCESS,
polzvtlSpilkuvns });

        return polzvtlSpilkuvns;
    }

    return null;

} catch (err) {
    yield put({ type: POLZVTL_SPILKUVNS_ERR, err });
    console.warn('Err get polzvtl spilkuvns', err);
}
}

export default function (state: ISpilkuvnReducer = initState,
action: AnyAction) {
    switch (action.type) {
        case GET_SPILKUVN_REQUESTING:
        case POLZVTL_SPILKUVNS_REQUESTING:
        case CRTPRVT_SPILKUVN_REQUESTING: {
            return {
                ...state,
                loading: true
            }
        }

        case GET_SPILKUVN_SUCCESS:
        case CRTPRVT_SPILKUVN_SUCCESS: {
            return {
                ...state,
                spilkuvn: action.spilkuvn,
                loading: false
            }
        }

        case POLZVTL_SPILKUVNS_SUCCESS: {
            return {
                ...state,
                polzvtlSpilkuvns: _.orderBy(action.polzvtlSpilkuvns, (
spilkuvn ) => (spilkuvn.lastMsg ? spilkuvn.lastMsg.creatAt : ""),
'desc'),
                loading: false
            }
        }

        case CLEAN_SPILKUVN: {
            return {
                ...state,

```

```

    spilkuvn: []
  }
}

case SUBSCRBED_SPILKUVN: {
  const spilkuvns = state.polzvttlSpilkuvns;
  const index = _.findIndex(spilkuvns, { id:
action.spilkuvn.id });

  if (index === -1) {
    spilkuvns.push(action.spilkuvn);
  } else {
    spilkuvns[index] = action.spilkuvn;
  }

  return {
    ...state,
    polzvttlSpilkuvns: _.orderBy(spilkuvns, ( spilkuvn ) =>
(spilkuvn.lastMsg ? spilkuvn.lastMsg.creatAt : ""), 'desc'),
  }
}

case GET_SPILKUVN_ERR:
case POLZVTTL_SPILKUVNS_ERR:
case CRTPRVT_SPILKUVN_ERR: {
  return {
    ...state,
    err: action.err,
    loading: false
  }
}
}

return state;
}

export const createPrivateSpilkuvn = gql `
  mtn createPrivateSpilkuvn($rpntId : ID!) {
    createPrivateSpilkuvn(rpntId : $rpntId ) {
      id
      private
      unreadMsgs
      lastMsg {
        msg
        creatAt
        owner {
          polzvttlname,

```

```

        polzvtlProfile {
            profilePhoto
            frstName
            lstName
        }
    }
}
uchatnyks {
    id
    polzvtlname
    polzvtlProfile {
        profilePhoto
        frstName
        lstName
    }
}
}
}
;

```

```

export const getSpilkuvn = gql `
    zapyt spilkuvn($id: ID!) {
        spilkuvn(id: $id) {
            id
            private,
            unreadMsgs
            lastMsg {
                msg
                creatAt
                owner {
                    polzvtlname,
                    polzvtlProfile {
                        profilePhoto
                        frstName
                        lstName
                    }
                }
            }
        }
    }
    uchatnyks {
        id
        polzvtlname
        polzvtlProfile {
            profilePhoto
            frstName
            lstName
        }
    }
}
`

```

```

    }
  `;

export const polzvtlSpilkuvns = gql `
  zapyt {
    polzvtlSpilkuvns {
      id
      private,
      unreadMsgs
      lastMsg {
        msg
        creatAt
        owner {
          polzvtlname,
          polzvtlProfile {
            profilePhoto
            frstName
            lstName
          }
        }
      }
    }
  }
  uchatnyks {
    id
    polzvtlname
    polzvtlProfile {
      profilePhoto
      frstName
      lstName
    }
  }
}

export default class SpilkuvnRslvr {
  static async createPrivateSpilkuvn(rpntId : string) {
    try {
      await TknService.checkTknExpired();
      const vidpovd = await Clnt.mutate({
        vrbls: {rpntId },
        mtn: createPrivateSpilkuvn
      });

      return vidpovd.dani.createPrivateSpilkuvn;
    } catch (err) {
      console.info(err);
      return null;
    }
  }
}

```

```

    }

    static async getSpilkuvn(id: string) {
      try {
        await TknService.checkTknExpired();
        const vidpovd = await Clnt.zapyt<GetSpilkuvnZapyt>({
          vrbls: { id },
          zapyt: getSpilkuvn
        });
        return vidpovd.dani.spilkuvn;
      } catch (err) {
        console.info(err);
        return null;
      }
    }

    static async polzvttlSpilkuvns() {
      try {
        await TknService.checkTknExpired();
        const vidpovd = await Clnt.zapyt<PolzvttlSpilkuvnsZapyt>({
          zapyt: polzvttlSpilkuvns
        });
        return vidpovd.dani.polzvttlSpilkuvns;
      } catch (err) {
        console.info(err);
        return null;
      }
    }
  }

  const getCurrentPolzvttl = async (tkn) => {
    if (tkn) {
      try {
        return await JwtSrvc.verify(tkn, tknConf.access.secret);
      } catch (e) {
        throw new AuthenticationErr(
          'Сесію завершено.',
        );
      }
    }
  };

  const srvr = new ApolloSrvr({
    typeDfs,
    rslvrs,
    cntx: async (dani) => {
      const {

```

```

    req,
    connection
  } = dani;
  let tkn;

  if (req) {
    tkn = req.headers['x-tkn'];
  } else if (connection) {
    tkn = connection.ctx['x-tkn']
  }

  const currentPolzvttl = await getCurrentPolzvttl(tkn);

  return {
    mdls,
    currentPolzvttl,
    loaders: {
      polzvttlProfile: new DaniLoader((keys) =>
loadrs.polzvttlProfile.batchPolzvttlProfiles(keys, mdls)),
      krtrs: new DaniLoader((keys) =>
loadrs.common.batchKrtrs(keys, mdls)),
      readMsg: new DaniLoader((keys) =>
loadrs.msg.batchReadMsgs(keys, currentPolzvttl, mdls)),
    },
  };
},
}

export default {
  Zapyt: {
    spilkuvn: combineRslvrs(
      isAuthenticated,
      async (prnt, { id }, { mdls, currentPolzvttl }) => {
        try {
          const { sub: polzvttlId } = currentPolzvttl;

          return await mdls.Spilkuvn.findById(id, {
            include: [{
              model: mdls.PolzvtlSpilkuvn,
              as: 'polzvttlSpilkuvnIds',
              where: { polzvttlId }
            }, {
              model: mdls.Polzvtl,
              as: 'uchatnyks',
            }
          ]),
        });
      } catch (err) {

```

```

        throw new Err(err);
    }
 )),

polzvtlSpilkuvns: combineRslvrs(
  isAuthenticated,
  async (prnt, args, { mdls, currentPolzvtl }) => {
    try {
      const { sub: polzvtlId } = currentPolzvtl;

      return await mdls.Spilkuvn.findAll({
        include: [{
          model: mdls.PolzvtlSpilkuvn,
          as: 'polzvtlSpilkuvnIds',
          where: { polzvtlId }
        }, {
          model: mdls.Polzvtl,
          as: 'uchatnyks',
        }],
      });
    } catch (err) {
      throw new Err(err);
    }
  })
},

Mtn: {
  createSpilkuvn: combineRslvrs(
    isAuthenticated,
    async (prnt, args, { mdls, currentPolzvtl }) => {
      try {
        const { sub: crtrId } = currentPolzvtl;
        args.crtrId = crtrId;
        args.uchatnyks.push(crtrId);

        const spilkuvn = await mdls.Spilkuvn.create(args);
        await spilkuvn.setUchatnyks(args.uchatnyks);

        return spilkuvn;
      } catch (err) {
        throw new Err(err);
      }
    }
  )),

  createPrivateSpilkuvn: combineRslvrs(
    isAuthenticated,
    async (prnt, { rpntId }, {mdls, currentPolzvtl }) => {
      try {

```

```

        const { sub: crtrId } = currentPolzvttl;

        const spilkuvn = await mdls.Spilkuvn.create({ private:
true });
        await spilkuvn.setUchatnyks([ crtrId, rpntId ]);

        return spilkuvn;
      } catch (err) {
        throw new Err(err);
      }
    }
  ),

  updateSpilkuvn: combineRslvrs(
    isSpilkuvnCreator,
    async (prnt, args, { mdls }) => {
      try {
        const spilkuvn = await mdls.Spilkuvn.update(args, {
          filds: Object.keys(args),
          retrng: true,
          plgn: true,
          where: { id: args.id }
        });

        return spilkuvn[1];
      } catch (err) {
        throw new Err(err);
      }
    }
  )),

  deleteSpilkuvn: combineRslvrs(
    isSpilkuvnCreator,
    async (prnt, { id }, { mdls }) => {
      try {
        await mdls.Spilkuvn.destroy({ where: { id } });

        return true;
      } catch (err) {
        throw new Err(err);
      }
    }
  )),

  },

  Spilkuvn: {
    lastMsg: async ({ id }, args, { mdls }) => {
      try {
        return mdls.Msg.findOne({
          where: { spilkuvnId: id },

```

```

        order: [[ 'creatAt', 'DESC' ]]
      });
    } catch (err) {
      throw new Err(err);
    }
  },

  uchatnyks: async ({ id }, args, { mdls }) => {
    try {
      return await mdls.Polzvtl.findAll({
        include: [{
          model: mdls.PolzvtlSpilkuvn,
          as: 'polzvtlSpilkuvnIds',
          where: { spilkuvnId: id }
        }]
      });
    } catch (err) {
      throw new Err(err);
    }
  },

  unreadMsgs: async ({ id: spilkuvnId }, args, { mdls,
currentPolzvtl }) => {
    try {
      const Op = Psqlzlgn.Op;
      const { sub: polzvtlId } = currentPolzvtl;

      const allMsgs = await mdls.Msg.count({ where: {
spilkuvnId }});
      const read = await mdls.Msg.findAndCountAll({
        where: { spilkuvnId },
        include: [{
          model: mdls.PolzvtlMsg,
          as: 'polzvtlMsgId',
          required: true,
          where: { polzvtlId }
        }]
      });

      return allMsgs - read.count;
    } catch (err) {
      throw new Err(err);
    }
  }
}

const spilkuvn = (psqlzlgn, DaniTypes) => {

```

```

const Spilkuvn = psqlzlgndefine('Spilkuvn', {
  name: {
    type: DaniTypes.STRING(16)
  },
  description: {
    type: DaniTypes.STRING(128)
  },
  icon: {
    type: DaniTypes.STRING
  },
  crtrId: {
    type: DaniTypes.INTEGER
  },
  private: {
    type: DaniTypes.BOOLEAN,
    defaultValue: false
  }
}, {});
Spilkuvn.associate = function(mdls) {
  Spilkuvn.hasMany(mdls.Msg, {
    foreignKey: 'spilkuvnId',
    as: 'msgs'
  });

  Spilkuvn.hasMany(mdls.PolzvtlSpilkuvn, {
    foreignKey: 'spilkuvnId',
    as: 'polzvtlSpilkuvnIds'
  });

  Spilkuvn.belongsTo(mdls.Polzvtl, {
    foreignKey: 'crtrId',
  });

  Spilkuvn.belongsToMany(mdls.Polzvtl, {
    foreignKey: 'spilkuvnId',
    as: 'uchatnyks',
    through: 'PolzvtlSpilkuvns',
  })
};
return Spilkuvn;
};

```

ДОДАТОК В
Слайди презентації

Міністерство освіти і науки України
Національний університет «Запорізька політехніка»
Кафедра програмних засобів

Програмне забезпечення вебсайту для обговорення місцевих проблем

Виконав: Вадим АНТЮХОВ,
ст. групи КНТ-131
керівник Степан СКРУПІСЬКИЙ,
к.т.н., доцент

Рисунок В.1 – Слайд 1

Об'єкт, предмет та мета роботи

- **Об'єкт дослідження** – процес розробки програмного забезпечення для обговорення регіональних питань.
- **Предмет дослідження** – програмні засоби для обговорення регіональних питань.
- **Метою роботи є** розробка програмного забезпечення для обговорення місцевих проблем.

Рисунок В.2 – Слайд 2

Завдання роботи

Для досягнення поставленої мети у кваліфікаційній роботі бакалавра необхідно розв'язати такі задачі:

- виконати аналіз предметної області та програмних засобів для обговорення місцевих проблем;
- здійснити проєктування програмного забезпечення для обговорення місцевих проблем;
- створити програмне забезпечення для обговорення місцевих проблем;
- виконати тестування розробленого програмного забезпечення для обговорення місцевих проблем.

3

Рисунок В.3 – Слайд 3

Порівняння існуючих аналогів

Критерій порівняння	Recollective	FlexMR	Vocal Views
Підтримка текстових дискусій	+	+	+—
Підтримка великих груп учасників	+	+—	+
Гнучкість налаштувань	+	+	+—
Можливість пошуку учасників	+	+	+—
Легкість модерації	+	+	+—
Зручність інтерфейсу користувача	+—	+—	+

4

Рисунок В.4 – Слайд 4

Порівняння мов програмування

Критерій порівняння мов програмування	Мова програмування		
	JavaScript	Python	PHP
Продуктивність на сервері	+	+-	+-
Підтримка асинхронного програмування	+	+-	-
Широка екосистема та бібліотеки	+	+-	+-
Зручність для створення ПЗ вебсайту для обговорення місцевих проблем	+	-	-
Безпека даних	+	+	+-

5

Рисунок В.5 – Слайд 5

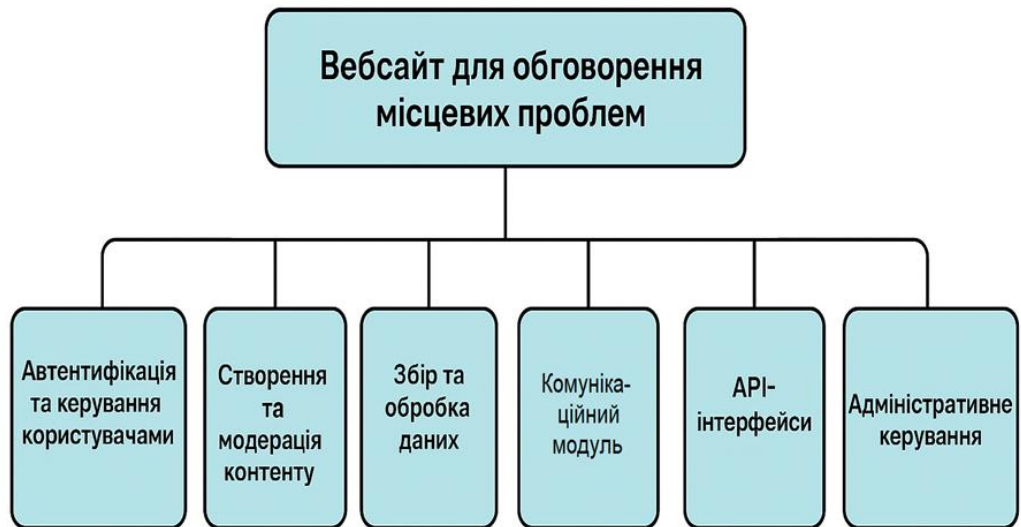
Обґрунтування вибору середовища виконання

Критерій порівняння середовищ виконання	Середовища виконання		
	Node.js	Deno	JScrip
Продуктивність	+	+-	-
Асинхронна обробка запитів	+	+-	+-
Підтримка сучасних бібліотек та фреймворків	+	+-	-
Ширина екосистеми та підтримка пакетів	+	+-	-
Масштабованість	+	+-	+-

6

Рисунок В.6 – Слайд 6

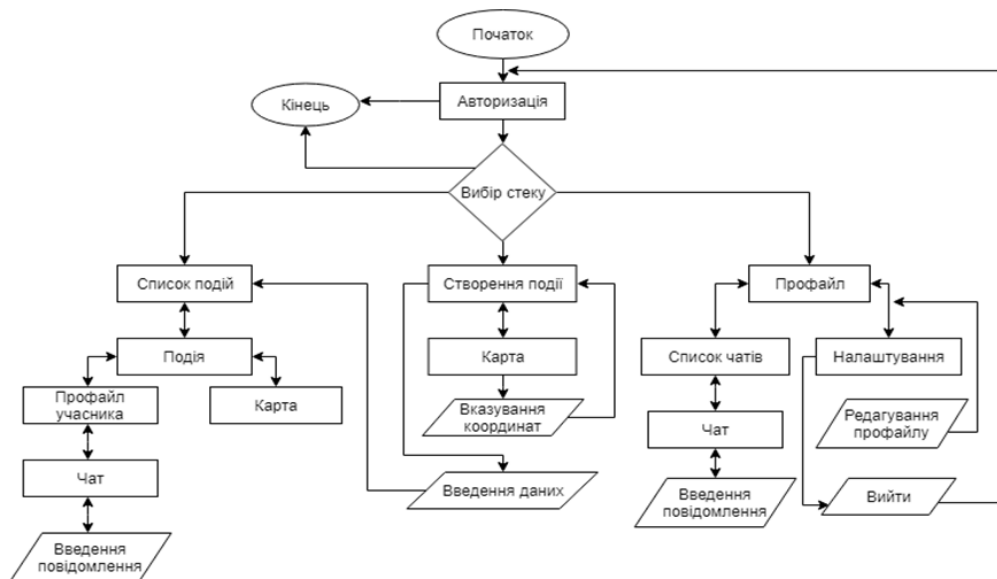
Структура програмного забезпечення для обговорення місцевих проблем



7

Рисунок В.7 – Слайд 7

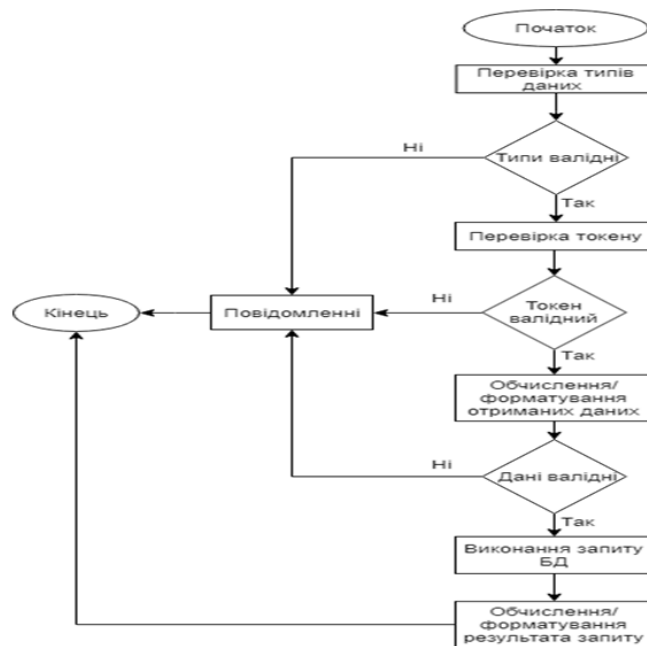
Схема функціонування клієнтської частини ПЗ



8

Рисунок В.8 – Слайд 8

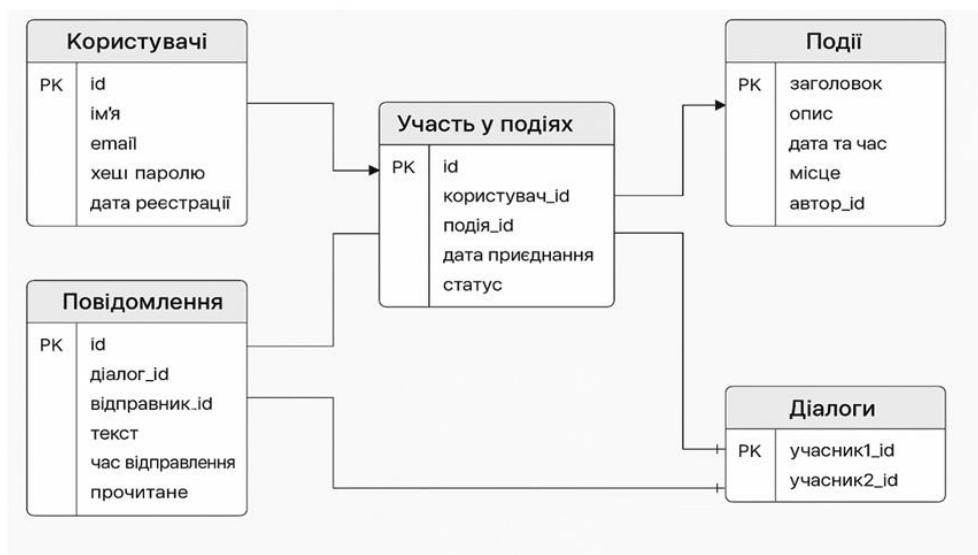
Схема функціонування серверної частини ПЗ



9

Рисунок В.9 – Слайд 9

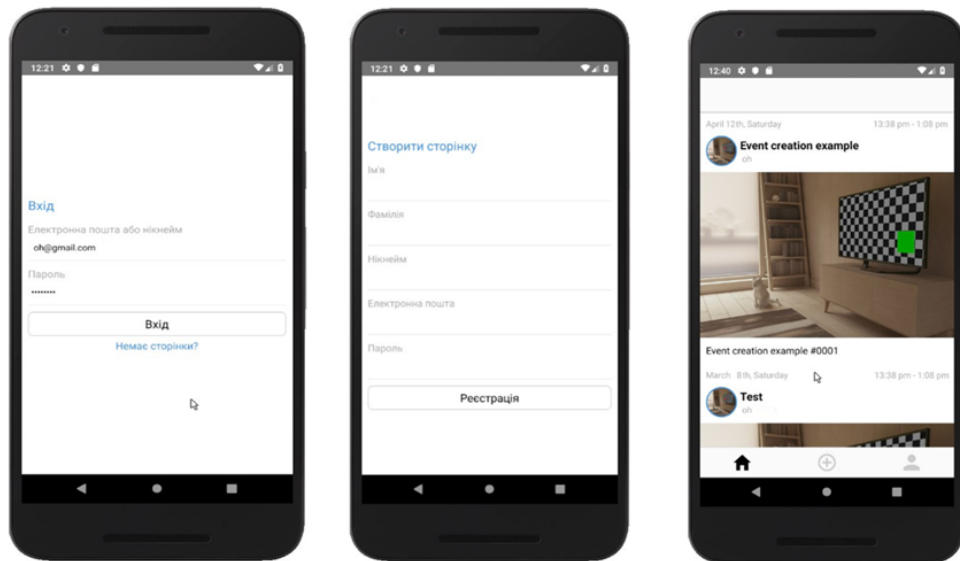
Схема бази даних ПЗ для обговорення місцевих проблем



10

Рисунок В.10 – Слайд 10

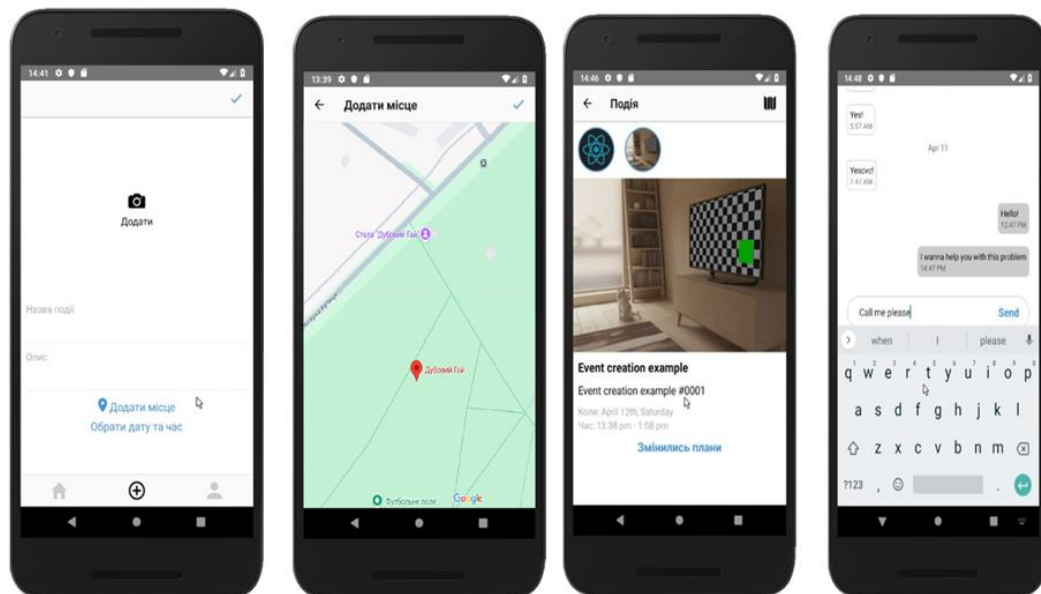
Інтерфейс застосунку. Авторизація. Реєстрація. Основний екран



11

Рисунок В.11 – Слайд 11

Інтерфейс застосунку. Додавання події. Обговорення місцевої події



12

Рисунок В.12 – Слайд 12

Висновки

В ході виконання дипломної кваліфікаційної роботи бакалавра було проаналізовано та досліджено процес розробки програмного забезпечення для обговорення регіональних питань.

Для розробки програмного забезпечення для обговорення місцевих проблем обрано мову програмування JavaScript та середовище виконання Node.js.

Розроблено програмне забезпечення для обговорення місцевих проблем. Визначено, що структура програмного забезпечення, орієнтованого на обговорення місцевих проблем, використовує клієнт-серверну архітектуру та передбачає поділ системи на кілька функціональних модулів, кожен із яких виконує окрему роль, забезпечуючи злагоджену роботу всієї платформи.

Результати тестування програмного забезпечення показали, що розроблена програма дозволяє забезпечити автоматизацію процесів, пов'язаних з підтримкою процесу обговорення місцевих проблем.

Усі завдання випускної дипломної роботи бакалавра повністю виконано.