Міністерство освіти і науки України
Луцький національний технічний університет
Факультет комп'ютерних та інформаційних технологій
Кафедра інженерії програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА

ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «БАКАЛАВР»

РОЗРОБКА ПЛАТФОРМИ ДЛЯ КООРДИНАЦІЇ ВОЛОНТЕРСЬКОЇ ДОПОМОГИ З ВИКОРИСТАННЯМ REACT, NODE.JS ТА MYSQL

DEVELOPMENT OF A PLATFORM FOR COORDINATING VOLUNTEER ASSISTANCE USING REACT, NODE.JS AND MYSQL

спеціальність 121 «Інженерія програмного забезпечення»
освітня програма «Інженерія програмного забезпечення»

Виконала: здобувачка вищої освіти
групи ІПЗ-41
Надашкевич Анастасія Геннадіївна

Керівник:
Вознюк Анастасія Вадимівна

Кваліфікаційну роботу
допущено до захисту
«10» 06 2025 р.
Гарант освітньої програми:
к.т.н., доцент
Ліщина Наталія Миколаївна

Луцьк – 2025 року

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет *комп'ютерних та інформаційних технологій*
Кафедра *інженерії програмного забезпечення*
Ступінь вищої освіти *бакалавр*
Галузь знань: *12 «Інформаційні технології»*
Спеціальність: *121 «Інженерія програмного забезпечення»*
Освітня програма: «*Інженерія програмного забезпечення»*

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____
«*20*» *12* 20*24* р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

Надашкевич Анастасії Геннадіївні
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи *«Розробка платформи для координації волонтерської допомоги з використанням React, node.js та MySQL»*
Керівник роботи: *асистент Вознюк А. В.*
затверджені наказом закладу вищої освіти від *«19» грудня 2024 р.* № *474/01-02*
2. Строк подання здобувачем вищої освіти кваліфікаційної роботи бакалавра *«10» червня 2025 р.*
3. Вихідні дані до роботи: *React, node.js, npm, Visual Studio code, Chakra UI, MySQL, phpMyAdmin, методичні вказівки до виконання кваліфікаційної роботи бакалавра.*
4. Зміст розрахунково-пояснювальної записки (перелік питань, що потрібно розробити): *аналіз існуючих систем координації волонтерської допомоги, постановка задачі, визначення функціональних і нефункціональних вимог, розробка UML-діаграм, вибір технологічного стеку, реалізація програмного забезпечення, створення бази даних, тестування, налагодження, механізми захисту, розробка документації.*
5. Перелік графічного матеріалу: *12 рисунків, 4 таблиці, 1 додаток.*

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
Аналіз предметної області	*Вознюк А. В.*		
Специфікація вимог до розробленої системи	*Вознюк А. В.*		
Розробка об`єкта проектування	*Вознюк А. В.*		
Нормоконтроль	*Вознюк А. В.*		
Гарант ОП	*Ліщина Н. М.*		
Показник запозичень тексту	2.29 %		
Академічна доброчесність	*Вознюк А. В.*		

7. Дата видачі завдання *«20» грудня 2024 р.*

№	Назва етапів кваліфікаційної роботи бакалавра	Строк виконання етапів роботи	Примітка
1	Огляд літературних джерел по темі кваліфікаційної роботи бакалавра	до 04.02.2025 р.	Виконано
2	Аналіз проблеми розробки та впровадження об`єкту проектування	до 01.03.2025 р.	Виконано
3	Обгрунтування вибору шляхів, технологій і засобів вирішення поставленого завдання	до 15.03.2025 р.	Виконано
4	Розробка функціонально-структурної схеми роботи об'єкта проектування та проектування бази даних	до 29.03.2025 р.	Виконано
5	Практична реалізація об`єкта проектування та розробка бази даних	до 26.04.2025 р.	Виконано
6	Тестування та налагодження об'єкта проектування	до 03.05.2025 р.	Виконано
7	Здача чистового варіанту кваліфікаційної роботи бакалавра на кафедру	до 10.06.2025 р.	Виконано

Здобувач вищої освіти Анастасія НАДАШКЕВИЧ

Керівник кваліфікаційної роботи Анастасія ВОЗНЮК

АНОТАЦІЯ

Надашкевич А. Г. Розробка платформи для координації волонтерської допомоги з використанням React, node.js та MySQL. Рукопис. Кваліфікаційна робота бакалавра ОП «Інженерія програмного забезпечення» спеціальності «Інженерія програмного забезпечення». Луцький національний технічний університет. Луцьк, 2025.

Кваліфікаційна робота бакалавра складається зі вступу, трьох розділів, висновків, списку використаних джерел та додатків.

У першому розділі здійснено аналіз сучасного стану проблем координації волонтерської допомоги, розглянуто аналоги та сформульовано завдання розробки. У другому розділі визначено функціональні та нефункціональні вимоги до системи, здійснено проєктування програмного забезпечення, побудовано UML-діаграми, обґрунтовано вибір технологій та інструментів розробки. У третьому розділі описано реалізацію платформи на основі React, Node.js, MySQL, проведено тестування, налагодження, реалізовано захист, а також підготовлено документацію для впровадження. У висновках підсумовано результати роботи, визначено її практичну значущість і напрями подальшого вдосконалення.

Ключові слова: координація, волонтерська допомога, вебплатформа, React, Node.js, MySQL.

ABSTRACT

Nadashkevych A. Development of a Platform for Coordinating Volunteer Assistance Using React, node.js and MySQL. Manuscript. Qualification work of the bachelor's program "Software engineering" in the specialty "Software engineering". Lutsk National Technical University. Lutsk, 2025.

The bachelor's qualification thesis consists of an introduction, three chapters, conclusions, a list of references, and appendices.

The first chapter analyzes the current state of volunteer coordination systems, explores existing solutions, and defines the objectives of the project. The second chapter specifies the functional and non-functional requirements of the system, presents the software design, UML diagrams, and justifies the choice of tools and technologies. The third chapter describes the implementation of the platform using React, Node.js, and MySQL, as well as testing, debugging, security features, and system deployment documentation. The conclusions summarize the results, outline the practical significance, and suggest directions for further development.

Keywords: coordination, volunteer assistance, web platform, React, Node.js, MySQL.

ЗМІСТ

ВСТУП

Актуальність теми. Сучасні соціальні виклики, зокрема повномасштабна війна в Україні, спричинили стрімке зростання потреби у системній та ефективній координації волонтерської допомоги. У цих умовах цифрові рішення відіграють важливу роль в оптимізації комунікації між волонтерами та отримувачами допомоги, забезпеченні прозорості, відстежуваності та доступності гуманітарних ініціатив. Однак, попри численні локальні ініціативи, більшість наявних платформ або фокусуються на обмежених напрямах, або не надають повного функціоналу для інтерактивної взаємодії між сторонами – зокрема, відсутні можливості відстеження статусу заявок, спілкування між учасниками, сортування запитів за категоріями та персоналізоване відображення інформації.

У світі активно розвиваються інструменти civic-tech і волонтерських платформ, що базуються на сучасних вебтехнологіях, таких як React, Node.js, MySQL, а також використовують можливості API-інтеграцій. Це дає змогу створювати зручні, масштабовані, безпечні та інтуїтивно зрозумілі інтерфейси для організацій волонтерського руху. Україна також має значний потенціал у впровадженні таких рішень на місцевому рівні.

Відповідно, актуальність цієї роботи зумовлена відсутністю комплексних цифрових платформ для підтримки повного циклу координації волонтерської допомоги з урахуванням ролей, статусів та двосторонньої комунікації. Саме тому основною метою кваліфікаційної роботи є розробка вебплатформи для координації волонтерської допомоги з функціоналом реєстрації за ролями, створенням, виконанням і відстеженням замовлень, фільтрацією за категоріями, можливістю коментування та персоналізованими кабінетами для користувачів і волонтерів.

Мета роботи – розробити повнофункціональну вебплатформу для координації волонтерської допомоги, яка дозволяє створювати та виконувати запити на допомогу, керувати ролями користувачів, забезпечувати безпечну

автентифікацію, інтерактивну взаємодію між волонтерами та отримувачами, а також адміністративний контроль і супровід проєкту.

Об’єктом кваліфікаційної роботи є процес організації та цифрової підтримки координації волонтерської діяльності.

Предметом кваліфікаційної роботи виступає процес розробки інформаційної системи управління волонтерськими замовленнями. Робота має міждисциплінарний характер і пов’язана з проєктами в галузях інформаційних технологій, управління проєктами, соціального менеджменту та кібербезпеки.

Завдання роботи:

1) проаналізувати існуючі цифрові рішення для координації волонтерської допомоги та виділити їхні недоліки;

2) сформулювати технічне завдання на розробку платформи з урахуванням функціональних і нефункціональних вимог, розробити UML-діаграми, спроєктувати логічну структуру системи;

3) обрати засоби розробки, бібліотеки, мови програмування та архітектурні рішення для реалізації системи;

4) реалізувати клієнтську та серверну частини системи відповідно до вимог,

5) протестувати розроблену систему та усунути виявлені помилки;

6) розробити базу даних, структуру таблиць, ключі зв’язків, реалізувати SQL-запити для взаємодії з даними;

7) забезпечити безпеку системи через механізми автентифікації, хешування паролів, захист API;

8) розробити документацію, включаючи інструкції для запуску системи, опис інтеграцій, вимоги до хостингу та середовища.

Розробка ведеться з використанням технологічного стеку React для клієнтської частини, Node.js – для серверної логіки, MySQL – як системи керування базами даних. Передбачено інтеграцію зовнішніх API (у разі потреби) для підвищення функціональності платформи, зокрема в аспектах геолокації чи повідомлень.

Кваліфікаційна робота є логічним продовженням попередніх навчальних проєктів з веброзробки, а також тісно пов’язана з практичним досвідом проходження виробничої практики в організаціях, залучених до гуманітарної діяльності.

Апробація результатів дослідження. Основні результати, отримані в процесі розробки вебплатформи для координації волонтерської допомоги, були апробовані на X Міжнародній науково-практичній конференції з проблем вищої освіти і науки «Інформаційні технології в освіті, науці і виробництві (ІТОНВ-2025)», що відбулася 23-24 травня 2025 року в місті Луцьк. За результатами апробації було оприлюднено тези на тему «Роль вебтехнологій у розвитку громадських ініціатив» (додаток А), у яких представлено як теоретичне обґрунтування проєкту, так і практичні результати. Конференційне обговорення підтвердило актуальність тематики дослідження та його практичну значущість для соціально активних громад і організацій.

РОЗДІЛ 1
АНАЛІЗ СИСТЕМ КООРДИНАЦІЇ ВОЛОНТЕРСЬКОЇ ДОПОМОГИ І ПОСТАНОВКА ЗАВДАНЬ НА КВАЛІФІКАЦІЙНУ РОБОТУ

1.1 Аналіз сучасного стану проблеми

В умовах тривалої гуманітарної кризи, спричиненої війною в Україні, координація волонтерської допомоги набула надзвичайної ваги. У зв’язку з цим виникла потреба у цифрових інструментах, які б забезпечували ефективну комунікацію, облік запитів на допомогу, контроль за їх виконанням і аналітичну звітність.

Аналіз сучасних наукових і прикладних досліджень у сфері волонтерської діяльності дозволяє зробити висновок про надзвичайну актуальність створення ефективних цифрових рішень для підтримки та координації волонтерських ініціатив в Україні, особливо в умовах воєнного часу. У монографії П. Горінова та Р. Драпушка [1] детально розглядається волонтерська діяльність як соціальний феномен і складова національної безпеки. Автори підкреслюють недостатність науково-методичного забезпечення у цій сфері та звертають увагу на необхідність удосконалення нормативно-правової бази, а також використання міжнародного досвіду для гармонізації українського законодавства з європейськими стандартами.

Наукова стаття Г. М. Закалик та співавторів [2] розкриває волонтерство як форму громадянської активності, що стала відповіддю суспільства на соціально-економічні виклики та державну неспроможність забезпечити належний рівень соціального захисту. Автори акцентують увагу на тому, що волонтерська діяльність в умовах агресії росії виконує не лише гуманітарну функцію, а й формує національну єдність і мобілізаційну спроможність суспільства. Також висвітлюються шляхи залучення молоді до волонтерської спільноти та роль правової та психологічної підтримки волонтерів.

Робота С. С. Яремчук та співавторів [3] аналізує нормативне підґрунтя інституту волонтерства та доводить, що сучасне волонтерство в Україні є

переважно ініціативою громадян і базується на принципах самоорганізації, моральної відповідальності та патріотизму. Автори звертають увагу на нормативні прогалини, що стримують розвиток волонтерського руху, та пропонують механізми їх законодавчого врегулювання. Особливо цінним є порівняльний аналіз європейського досвіду, який може бути адаптований в Україні.

Окрему увагу заслуговує дослідження С. О. Доценко [4], де розглянуто зв'язок між цифровізацією та волонтерською активністю молоді. Авторка доводить, що в умовах цифрової епохи необхідно використовувати знайомі для молодого покоління інструменти – блоги, форуми, онлайн-опитування – для стимулювання соціальної активності та залучення до волонтерської роботи. Цифрові інструменти сприяють розвитку комунікативних навичок, вмінню працювати в команді, а також формуванню соціальної компетентності.

Найбільш прикладний характер має робота І. Л. Суботіна та О. І. Трунова [5], де аналізується досвід розробки інформаційно-аналітичної системи для координації волонтерської діяльності в прикордонному Чернігівському регіоні. Автори підкреслюють важливість цифровізації процесів координації, логістики, комунікації та інформування у волонтерському середовищі. У роботі детально описано реалізовані функції платформи «Volunteer», зокрема систему авторизації, чат-ботів, логістичних ланцюжків та систему управління подіями. Зазначено, що успіх таких систем значною мірою залежить від урахування реальних потреб волонтерів та зворотного зв'язку з ними.

Літературні джерела засвідчують як теоретичну, так і практичну потребу в сучасних цифрових рішеннях, які забезпечують ефективну, безпечну та гнучку платформу для організації волонтерської допомоги. Це підтверджує доцільність і обґрунтованість обраного напряму кваліфікаційної роботи.

Серед наявних рішень, які частково покривають потреби волонтерських ініціатив, можна виокремити такі системи, як HelpUkraine.center [6], Volunteer Country [7], UkraineNow [8], а також ініціативи на базі Google Forms або

Telegram-ботів. Ці рішення демонструють високу швидкість розгортання та доступність, однак мають істотні обмеження: відсутність захищеної багаторівневої реєстрації користувачів, відсутність структурованого контролю за виконанням запитів, обмежені можливості з фільтрації даних та комунікації між сторонами. Порівняльний аналіз функціональних можливостей наявних рішень подано в таблиці 1.1.

Таблиця 1.1 – Порівняльна характеристика існуючих рішень для координації волонтерської допомоги

Система / Рішення	Реєстрація за ролями	Відстеження статусу замовлення	Комунікація між сторонами	Пошук і фільтрація	Простота розгортання
HelpUkraine.center	Ні	Частково	Обмежена	Ні	Висока
Volunteer Country	Частково	Ні	Ні	Ні	Висока
UkraineNow	Ні	Ні	Ні	Ні	Висока
Google Forms + таблиці	Ні	Ні	Ні	Ні	Дуже висока
Telegram-боти (ручні)	Частково	Ні	Обмежена	Ні	Висока

Пошук за відкритими патентними базами (наприклад, Google Patents) не виявив повноцінних запатентованих аналогів, що б забезпечували інтерактивну координацію гуманітарної допомоги з фокусом на українські реалії та задіюванням сучасного вебстеку React, Node.js та MySQL. Іноземні системи, як-от VolunteerMatch [9], GivePulse [10] чи Be My Eyes [11], зосереджені переважно на волонтерських ініціативах у сфері навчання, інклюзії та дозвілля, і не мають достатньої гнучкості для оперативного реагування в умовах криз.

В цілому, доцільність створення нової платформи полягає в інтеграції кращих практик UI/UX-дизайну, безпечного управління базами даних, персоналізованого доступу до функціоналу та ефективної моделі взаємодії між замовником і виконавцем. Нова система дозволить вирішити існуючі проблеми фрагментації, забезпечити прозорість виконання запитів і зменшити навантаження на волонтерські організації за рахунок автоматизації основних процесів.

1.2 Постановка завдання на кваліфікаційну роботу бакалавра

Метою цієї кваліфікаційної роботи є створення ефективної вебплатформи для координації волонтерської допомоги із забезпеченням зручної взаємодії між користувачами та волонтерами, а також реалізацією повного циклу функціонування системи: від реєстрації та створення запитів до їх виконання, комунікації між учасниками та підтримки захисту й масштабованості системи.

Для досягнення цієї мети необхідно вирішити низку взаємопов’язаних завдань:

1) провести аналіз сучасного стану цифрових інструментів у сфері координації волонтерської допомоги, зокрема міжнародного та українського досвіду використання вебплатформ у громадських ініціативах;

2) сформулювати функціональні та нефункціональні вимоги до програмного забезпечення відповідно до очікувань користувачів платформи – волонтерів та отримувачів допомоги. На основі аналізу побудувати загальну модель системи та визначити основні компоненти її архітектури;

3) обґрунтувати вибір засобів реалізації, включно з клієнтською технологією (React), серверною частиною (Node.js), СУБД (MySQL), а також допоміжними бібліотеками та модулями (Chakra UI, Leaflet, JWT, OAuth, тощо). Визначити оптимальні методи проєктування та алгоритмічні рішення;

4) реалізувати практичну частину платформи, яка включає: створення зручного інтерфейсу для обох ролей користувачів, авторизацію, реєстрацію, створення та обробку запитів, коментарі, карту з маркерами, особисті кабінети та модальні вікна для підтвердження дій;

5) здійснити тестування розробленої системи, включаючи юніт-тести, ручне тестування інтерфейсів, перевірку логіки взаємодії з API та базою даних. За потреби – усунути виявлені помилки та провести налагодження функціональності;

6) створити базу даних, що охоплює основні сутності проєкту: користувачів, запити, коментарі, категорії, ролі. Описати реалізацію схеми БД у

середовищі MySQL з використанням SQL-запитів і поясненням структури таблиць;

7) реалізувати базові механізми безпеки: автентифікацію користувачів із використанням JWT-токенів і Google OAuth, захист маршрутизаторів API, шифрування паролів за допомогою bcrypt, а також обмеження дій відповідно до ролей користувачів;

8) підготувати документацію до вебзастосунку, яка включає системні вимоги, інструкції з розгортання на сервері, підключення до бази даних і налаштування зовнішніх сервісів. Описати довідкову інформацію для користувача.

Виконання зазначених завдань дозволить розробити повнофункціональну, зручну й безпечну платформу для координації волонтерської допомоги, що відповідатиме сучасним вимогам до інформаційних систем.

Висновки до розділу 1

Проведений аналіз літературних джерел, практичних прикладів і функціонуючих рішень у сфері волонтерської діяльності підтверджує актуальність створення спеціалізованої цифрової платформи для координації гуманітарної допомоги. Сучасні виклики, спричинені воєнними діями та гуманітарною кризою в Україні, зумовлюють потребу в надійному та гнучкому інструменті, що забезпечуватиме прозору взаємодію між отримувачами допомоги та волонтерами, ефективний розподіл ресурсів, а також контроль за виконанням запитів.

Наявні рішення, як-от HelpUkraine.center, Volunteer Country, UkraineNow та Telegram-боти, хоча й виконують частину функцій, не мають достатньої функціональної глибини, безпеки, можливостей для фільтрації інформації та інтерактивної взаємодії між учасниками. Водночас огляд патентних баз не виявив комплексного технологічного рішення, орієнтованого саме на

український контекст, що підтверджує інноваційність та доцільність розробки нової системи.

Поставлені завдання кваліфікаційної роботи враховують основні функціональні й технічні вимоги до платформи, зокрема авторизацію за ролями, створення та керування замовленнями, механізми комунікації, захист даних, а також гнучкий інтерфейс користувача з можливістю персоналізації. Реалізація запропонованої архітектури з використанням React, Node.js та MySQL забезпечить високу швидкість роботи, масштабованість і адаптивність системи до потреб користувачів.

Отже, результати аналізу обґрунтовують вибір напряму роботи, визначають необхідність її реалізації та слугують базою для розробки технічної специфікації в наступному розділі.

РОЗДІЛ 2
СПЕЦИФІКАЦІЯ ВИМОГ ДО РОЗРОБЛЮВАНОЇ СИСТЕМИ

2.1 Аналіз, визначення вимог до розроблюваного програмного забезпечення та проектування програмного забезпечення

Розробка вебплатформи для координації волонтерської допомоги потребує чіткого формулювання вимог, вибору відповідної моделі програмної системи, а також формалізованого опису її архітектури та поведінки. Для досягнення цього було використано підхід, заснований на моделі «клієнт-сервер», який найкраще відповідає природі платформи з розподіленим доступом через інтернет. Модель реалізована за допомогою архітектури REST API, де клієнтська частина на React.js звертається до серверної логіки на Node.js через HTTP-запити, з обробкою даних у базі MySQL.

Для виявлення та аналізу вимог було застосовано комплексний підхід, що включав аналіз предметної області та існуючих аналогічних рішень, інтерв'ювання потенційних користувачів, серед яких були волонтери, координатори та отримувачі допомоги. Також використовувались сценарії використання, що дозволило змоделювати типові ситуації взаємодії з системою, а побудова таблиці «подія – відгук» допомогла формалізувати очікувану поведінку програмного забезпечення у відповідь на дії користувачів.

Цілі системи:

1) забезпечити просту реєстрацію користувачів за ролями: «Замовник» (отримувач допомоги) і «Волонтер»;

2) надати користувачам можливість створювати, переглядати та оновлювати статуси замовлень;

3) забезпечити виконання та відстеження замовлень із прив'язкою до виконавців;

4) реалізувати фільтрацію, пошук і сортування замовлень за категоріями;

5) забезпечити безпечну двосторонню комунікацію між виконавцем і замовником у вигляді коментарів;

6) надати особисті кабінети з персоналізованим відображенням інформації.

Функціональні вимоги:

1) реєстрація та авторизація користувачів із поділом на ролі (волонтер, отримувач);

2) створення, перегляд і редагування запитів на допомогу;

3) призначення волонтера до запиту та зміна статусу («новий», «в роботі», «виконано»);

4) виведення запитів на мапі з категоріями та геолокацією;

5) коментування запитів, додавання відгуків;

6) перегляд списку особистих запитів, сортування й фільтрація за категоріями.

Нефункціональні вимоги:

1) забезпечення безпечного зберігання та передачі даних (JWT, HTTPS, bcrypt);

2) адаптивність інтерфейсу під мобільні пристрої;

3) швидкість відгуку до 500 мс при запитах до сервера;

4) можливість масштабування системи;

5) доступність 24/7 із мінімальним простоєм.

Специфікація вимог по типу «Подія – Відгук» наведена в таблиці 2.1.

Таблиця 2.1 – Специфікація вимог

Подія	Відгук системи
Користувач натискає «Зареєструватися»	Відкривається форма реєстрації з вибором ролі
Запит створено	Запит зберігається в БД і з’являється в загальному списку
Волонтер натискає «Взяти в роботу»	Запит змінює статус, призначається поточному користувачу
Користувач відкриває карту	Система завантажує та виводить маркери запитів на Leaflet-карті

Інтерфейс спроєктовано відповідно до принципів доступності (accessibility), зрозумілості (usability) та логічної структурованості. Для побудови візуальних компонентів використано Chakra UI. Основні сторінки:

1) головна (список усіх запитів із фільтрами);

2) реєстрація / вхід;

3) особистий кабінет (для волонтера / отримувача);

4) сторінка створення нового запиту;

5) інтерактивна карта;

6) замовлення користувача.

Для проєктування програмного забезпечення платформи координації волонтерської допомоги розроблено UML-діаграми, які дозволяють візуалізувати логіку функціонування системи, її структуру та взаємодію користувачів з основними компонентами [12]. Діаграма варіантів використання (рис. 2.1) ідентифікує дії «Замовника» та «Волонтера».

Рисунок 2.1 – Діаграма варіантів використання

Ця діаграма показує взаємодію двох типів користувачів із системою: «Замовника» та «Волонтера». Обидва можуть реєструватися і входити в систему. «Замовник» створює замовлення, стежить за його статусом і коментує. «Волонтер» переглядає загальні замовлення, бере в роботу, змінює статус і також коментує.

Діаграма класів (рис. 2.2) – описує основні сутності: «Користувач», «Замовлення», «Коментар», «Категорія».

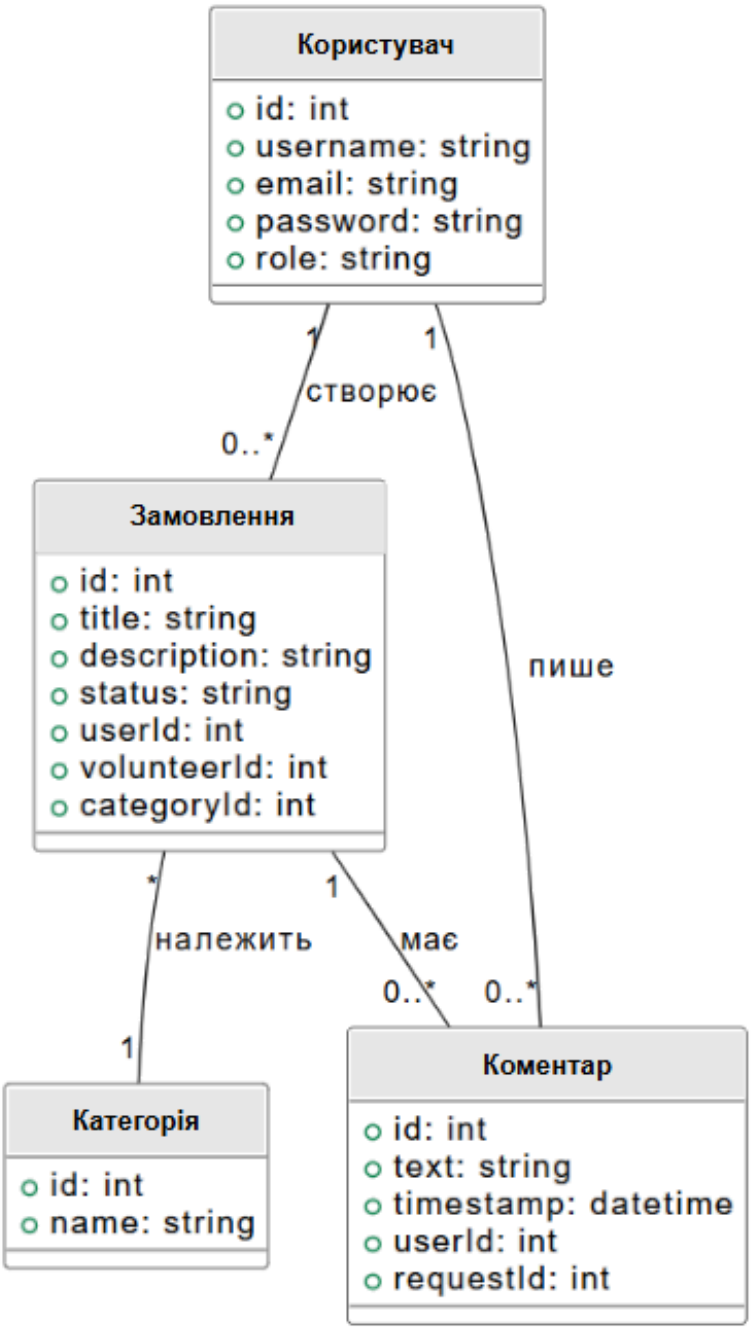

Рисунок 2.2 – Діаграма класів

Зв’язки показують, що один користувач може створювати багато замовлень і залишати багато коментарів, а кожне замовлення належить до певної категорії.

Діаграма послідовності (рис. 2.3) демонструє часову логіку процесів – від створення запиту до подальшого його прийняття відповідним волонтером. Вона показує, як саме користувач взаємодіє з інтерфейсом, а той – із сервером і базою даних. Всі етапи – запит, збереження, отримання, оновлення – відображені покроково та дозволяють точно зрозуміти послідовність подій у системі.

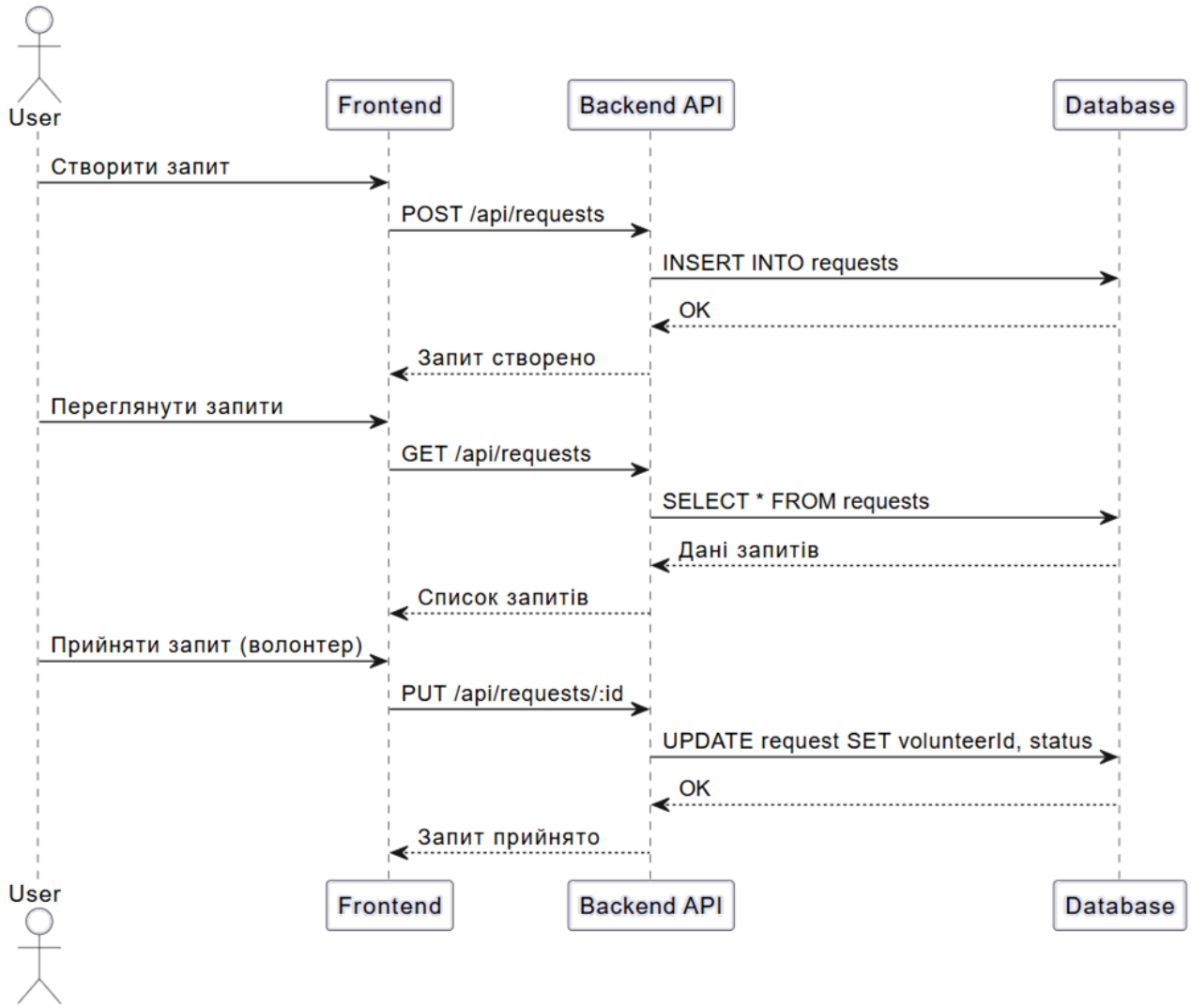

Рисунок 2.3 – Діаграма послідовності

Створені діаграми полегшують процес розробки, тестування та підтримки платформи, забезпечуючи цілісне бачення архітектури майбутнього програмного продукту.

Система організована за принципом однонаправлених потоків даних. Користувач взаємодіє через інтерфейс, що генерує події (onClick, onSubmit), які через React useEffect/useState передаються API. API маршрути обробляються в Node.js, виконуються SQL-запити до бази, і результат повертається у форматі JSON. Усі запити захищені JWT-токенами.

У результаті проведеного аналізу та проєктування програмного забезпечення для вебплатформи координації волонтерської допомоги було сформульовано повну специфікацію функціональних і нефункціональних вимог, розроблено UML-діаграми, визначено логіку взаємодії користувачів із системою, обґрунтовано вибір архітектури клієнт-сервер і стеку технологій, спроєктовано зручний UX/UI-інтерфейс і описано інформаційні потоки, що забезпечують ефективну, безпечну та масштабовану роботу платформи.

2.2 Вибір засобів, методів і алгоритмів вирішення поставленого завдання

У процесі розробки інформаційної системи координації волонтерської допомоги було проаналізовано низку інструментів, які забезпечують ефективне створення сучасного, масштабованого і безпечного вебзастосунку. З огляду на вимоги до системи, її функціональні особливості, надійність та адаптивність, було обрано стек технологій: React.js для клієнтської частини, Node.js (з Express) для серверної логіки, MySQL як СКБД, а також бібліотеки Chakra UI, JWT, Axios та Passport.js для додаткової функціональності.

React.js забезпечує гнучку та ефективну розробку сучасного користувацького інтерфейсу завдяки використанню компонентного підходу, що дозволяє будувати інтерфейс з незалежних, повторно використовуваних елементів [13]. Однією з основних переваг React є застосування віртуального

DOM – внутрішньої репрезентації реального DOM-дерева, яка дає змогу швидко відстежувати зміни в інтерфейсі та оновлювати лише ті його частини, які були змінені. Це значно зменшує навантаження на браузер, пришвидшує рендеринг і підвищує продуктивність вебзастосунку. Завдяки хукам, таким як useState, useEffect та useContext, розробник отримує зручний та декларативний спосіб керування станом, виконання побічних ефектів та передавання глобального контексту в додатку. React має активну спільноту, широке екосередовище та інтегрується з численними бібліотеками й фреймворками, що робить його ідеальним вибором для створення динамічного, адаптивного інтерфейсу користувача [14]. На графіках на рисунку 2.4 зображено динаміку популярності бібліотеки React і фреймворку Angular серед вебсайтів у різних категоріях (топ 10k, 100k, 1м і весь Інтернет). React демонструє стабільне зростання використання, тоді як Angular після піку в 2021 році має тенденцію до зниження.

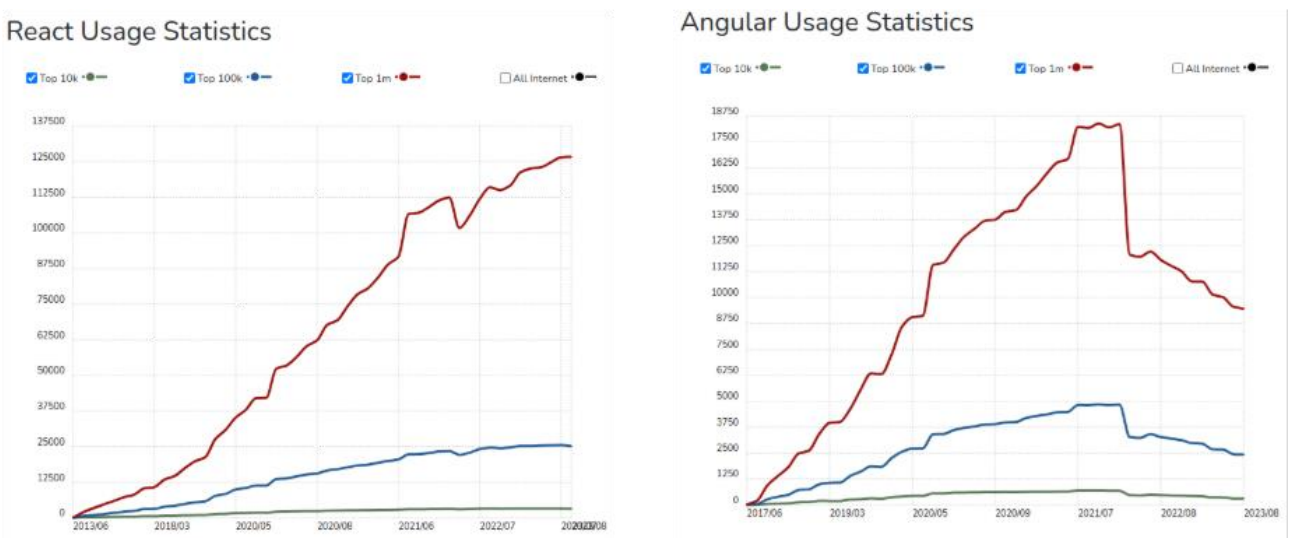

Рисунок 2.4 – Порівняння статистики використання React та Angular у веброзробці [15]

Node.js разом із Express.js слугує надійним середовищем для побудови серверної частини платформи, оскільки поєднує високу продуктивність з масштабованістю [16]. Node.js побудований на рушії V8 від Google і підтримує

неблокувальну (асинхронну) модель обробки запитів, що забезпечує ефективну роботу навіть при великій кількості одночасних з’єднань. Express.js, як мінімалістичний фреймворк поверх Node.js, значно спрощує створення RESTful API, забезпечуючи зрозумілу систему маршрутизації, зручну обробку запитів і відповідей, а також легке підключення middleware для авторизації, валідації чи обробки помилок [17]. За допомогою Express реалізується повний набір CRUD-операцій для основних сутностей системи, таких як запити про допомогу, користувачі, коментарі та категорії [18]. Сервіс також підтримує інтеграцію з базою даних MySQL, а для підвищення безпеки використовується JWT-токенізація, шифрування паролів та контроль доступу до ресурсів згідно з ролями користувачів. Такий вибір дозволяє побудувати надійний, логічно структурований і розширюваний бекенд, що відповідає вимогам сучасних вебсистем.

У проєкті реалізації платформи координації волонтерської допомоги базу даних побудовано з використанням системи управління базами даних MySQL, яка зарекомендувала себе як надійне та масштабоване рішення для вебзастосунків [19]. MySQL забезпечує підтримку транзакцій, зовнішніх ключів і складних SQL-запитів, що дозволяє реалізовувати складні логічні зв’язки між сутностями системи [20]. У структурі бази даних були передбачені основні таблиці – такі як users, requests, comments, categories, messages, кожна з яких виконує конкретну роль у збереженні та обробці даних. Зв’язки між таблицями реалізовані через зовнішні ключі, що забезпечує цілісність даних і дозволяє коректно здійснювати каскадні операції. Для підвищення продуктивності були створені індекси, які оптимізують виконання запитів на вибірку, особливо для фільтрації за категоріями, статусами та ідентифікаторами користувачів.

З метою забезпечення безпеки системи реалізовано механізм автентифікації та авторизації на основі JSON Web Tokens (JWT). Цей підхід дозволяє реалізувати токенізований доступ, який зберігає автентифікаційні дані у вигляді зашифрованого токена, що передається між клієнтом і сервером [21]. JWT використовується як основний спосіб контролю доступу до захищених

маршрутів, з урахуванням ролі користувача – чи то звичайний користувач, волонтер, чи адміністратор. Для зручності та швидкого входу користувачів, а також з метою мінімізації бар'єрів у реєстрації, додано можливість авторизації через Google, яка реалізується з використанням бібліотеки Passport.js з інтеграцією OAuth2. Цей механізм забезпечує безпечну взаємодію з Google API та спрощує процес ідентифікації, гарантуючи високий рівень захисту персональних даних [22].

На стороні клієнта активно використовується бібліотека Axios, яка дозволяє реалізувати асинхронний обмін даними між клієнтським інтерфейсом і серверним API. Axios спрощує обробку HTTP-запитів, обробку помилок, авторизаційних заголовків і взаємодію з JWT. Для побудови інтерфейсу користувача застосовується Chakra UI – сучасна бібліотека компонентів, що поєднує простоту стилізації з високою адаптивністю. Вона дозволяє створювати інтерфейси, які автоматично підлаштовуються під різні розміри екранів, мають вбудовану підтримку темної та світлої тем, а також забезпечують послідовність стилів [23]. Chakra UI полегшує розробку форм, діалогових вікон, елементів управління, модальних вікон і спливаючих повідомлень, що сприяє покращенню юзабіліті та доступності розробленої системи.

Алгоритмічна складова системи включає:

1) фільтрацію запитів за категоріями, статусами та ролями, що реалізовано через SQL-запити з параметрами;

2) сортування запитів та коментарів за датою створення;

3) захист API-ендпоінтів через проміжні функції перевірки ролі та токена (middleware);

4) генерацію унікальних токенів автентифікації;

5) реалізацію геолокаційної інтеграції через зовнішній API Nominatim для відображення запитів на карті (якщо активовано).

На рівні моделювання застосовані UML-діаграми. Діаграма на рисунку 2.5 ілюструє архітектуру фронтенду, побудованого на React. App

підключає окремі модулі (авторизація, запити, профіль), які містять відповідні компоненти сторінок і підкомпоненти для взаємодії з користувачем.

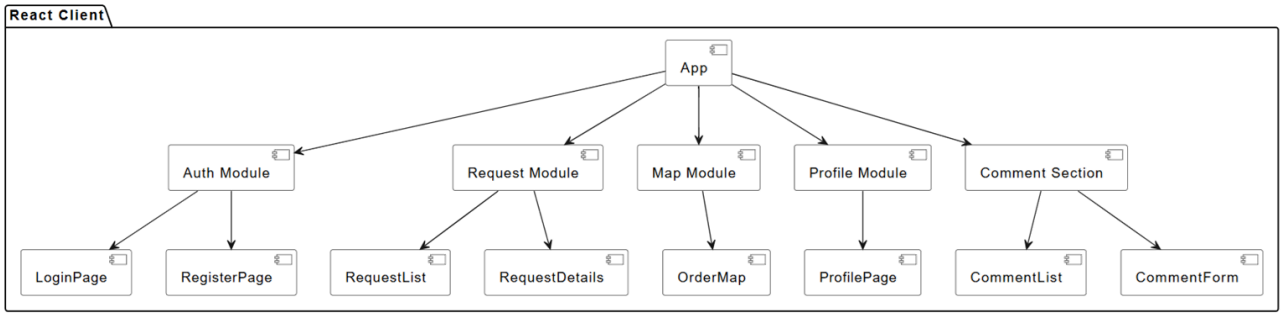

Рисунок 2.5 – Діаграма компонентів фронтенду

Блок-схема, що ілюструє процес створення запиту наведена на рисунку 2.6.

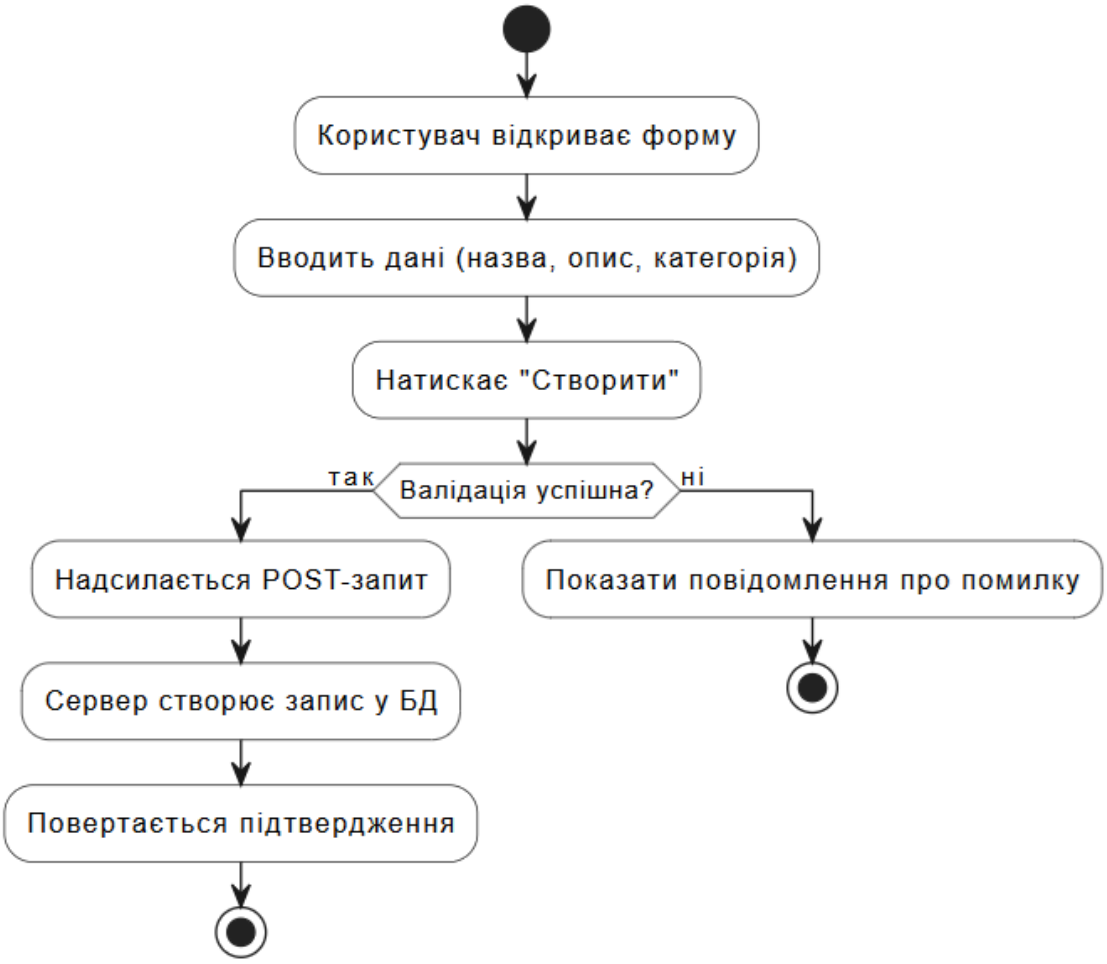

Рисунок 2.6 – Блок-схема створення запиту

Блок-схема процесу прийняття запиту волонтером наведена на рисунку 2.7.

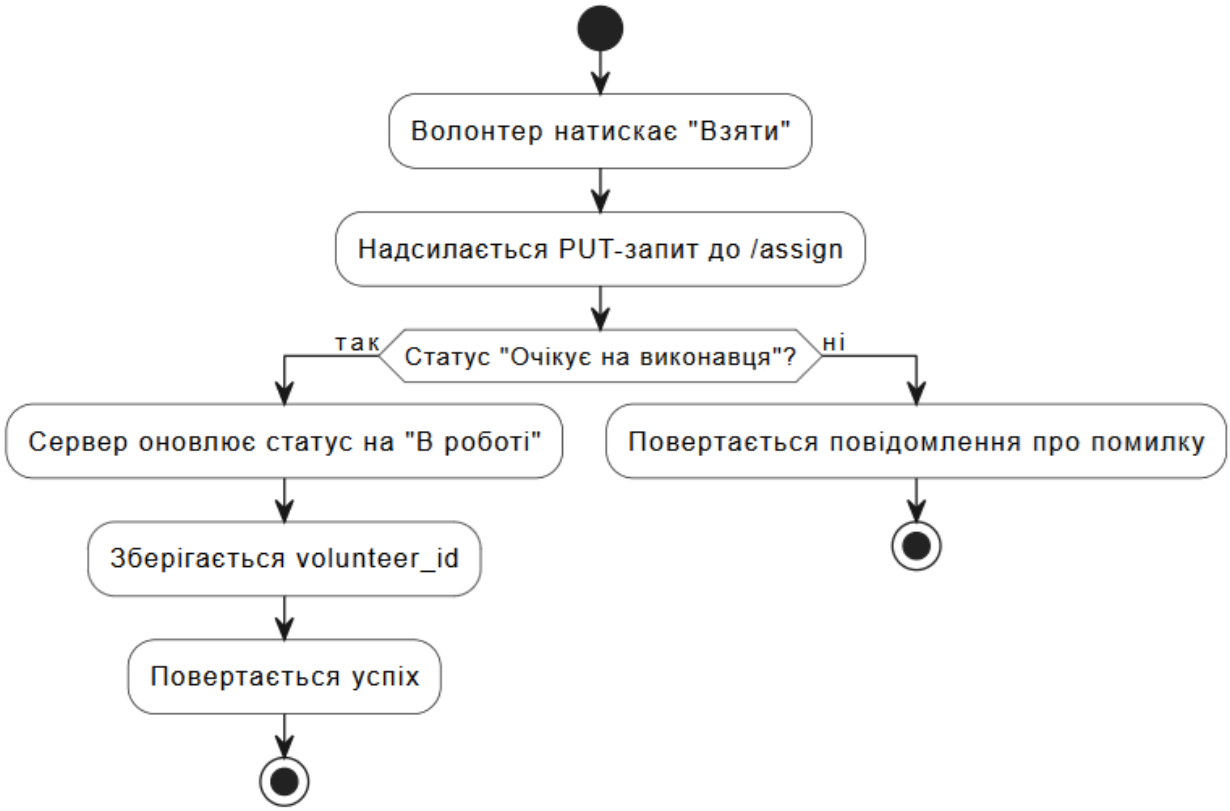

Рисунок 2.7 –Блок-схема прийняття запиту волонтером

Діаграма на рисунку 2.8 демонструє фізичне розміщення елементів платформи на різних вузлах:

1) клієнтський пристрій (браузер) завантажує React-застосунок і здійснює HTTP-запити до бекенду;

2) вебсервер на базі Node.js + Express обробляє логіку авторизації, замовлень, коментарів;

3) база даних MySQL зберігає всі сутності: користувачів, запити, категорії, коментарі;

4) Google OAuth Server забезпечує вхід через облікові записи Google.

Це дозволяє візуалізувати архітектуру з погляду взаємодії клієнта, сервера і бази даних. Якщо потрібно – можу згенерувати зображення цієї діаграми.

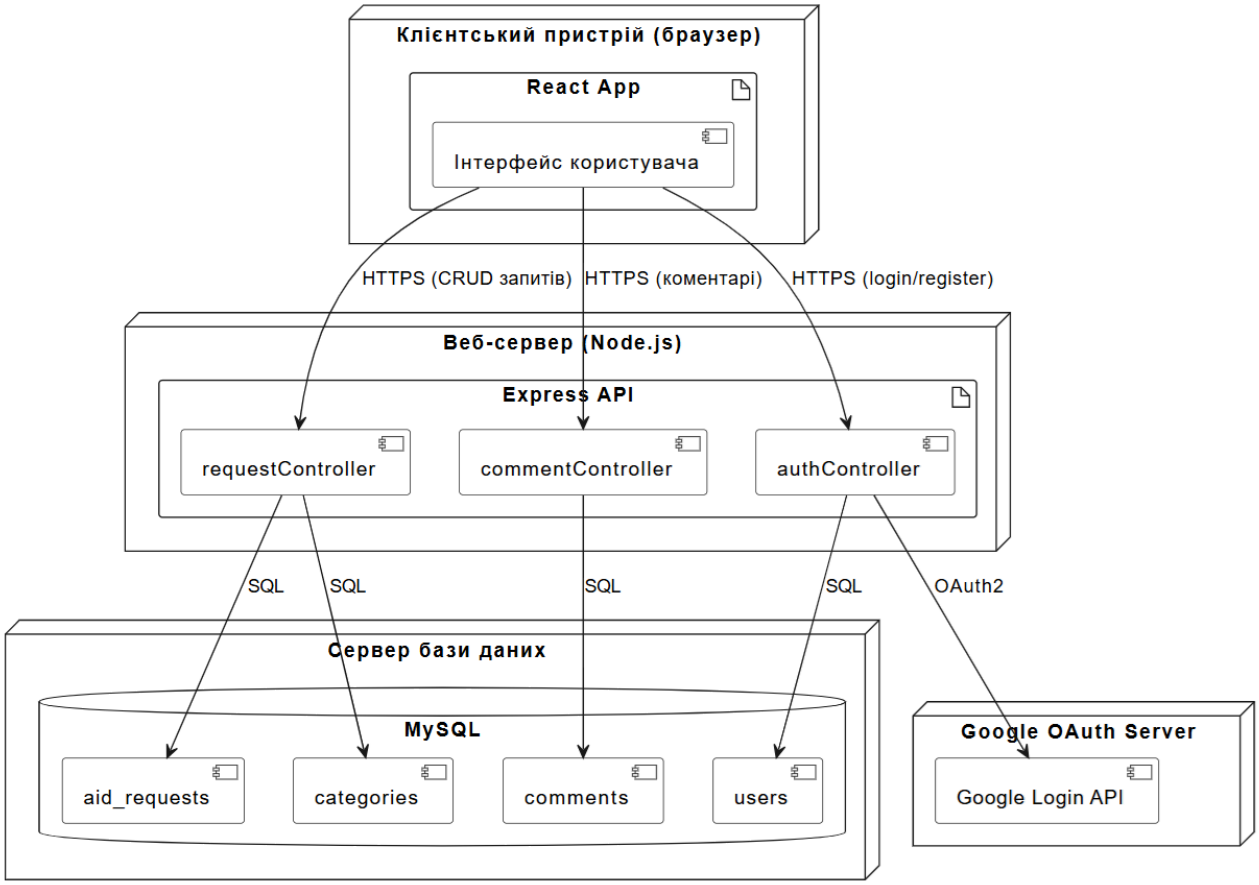

Рисунок 2.8 – Діаграма розгортання

Загалом, використання вищеописаних інструментів і технологій дозволяє створити повністю надійну, розширювану та зручну для користувачів інформаційну систему, адаптовану до задач координації волонтерської допомоги.

Висновки до розділу 2

У результаті виконання другого розділу було детально проаналізовано вимоги до розроблюваної вебплатформи для координації волонтерської допомоги, сформовано їхню формальну специфікацію, визначено архітектурну модель «клієнт-сервер» і методологію реалізації функціональних і нефункціональних характеристик системи.

Проведено проєктування інформаційної системи з використанням нотації UML, що дозволило візуалізувати логіку взаємодії користувачів із платформою,

структуру сутностей і поведінку при виконанні основних бізнес-процесів. Обґрунтовано вибір стеку технологій – React.js, Node.js, Express, MySQL, JWT, Chakra UI, Axios, Passport.js – з огляду на їхню гнучкість, масштабованість і відповідність сучасним вимогам веброзробки. Продемонстровано застосування асинхронної обробки запитів, реалізацію авторизації з розмежуванням доступу за ролями, фільтрацію та геолокацію запитів, збереження структурованих даних у базі MySQL.

Описані алгоритми, методи захисту, моделі даних і структури інтерфейсу забезпечують ефективну реалізацію функціоналу системи, її безпеку та зручність для кінцевих користувачів. Отримані результати підтверджують доцільність обраного підходу та закладають основу для подальшої розробки і тестування програмного продукту.

РОЗДІЛ 3
РОЗРОБКА ПЛАТФОРМИ ДЛЯ КООРДИНАЦІЇ ВОЛОНТЕРСЬКОЇ ДОПОМОГИ

3.1 Практична реалізація об’єкта проєктування

Процес розробки програмного забезпечення для платформи координації волонтерської допомоги було реалізовано з використанням стеку React.js для клієнтської частини, Node.js + Express.js для серверної логіки та MySQL як системи керування базами даних. Розробка велась у середовищах Visual Studio Code для фронтенду та бекенду, а також MySQL Workbench для адміністрування бази даних.

Клієнтська частина (frontend) реалізована з використанням бібліотеки React, яка дозволяє створювати динамічні інтерфейси користувача. Для візуального оформлення компонентів використовувалась бібліотека Chakra UI, що забезпечує швидку розробку адаптивних інтерфейсів.

Одним із важливих функціональних модулів є система автентифікації, яка підтримує як реєстрацію за email/паролем, так і вхід через Google OAuth (ліст. 3.1).

Лістинг 3.1 – Авторизація користувача через API

```
const response = await
axios.post("http://localhost:8000/api/v1/auth/login", {
  email,
  password,
});
localStorage.setItem("authToken", response.data.token);
setAuthState({ token: response.data.token, user: response.data.user });
```

кінець лістингу 3.1

Інтерфейс сторінки входу та реєстрації з відповідними формами зображено на рисунку 3.1.

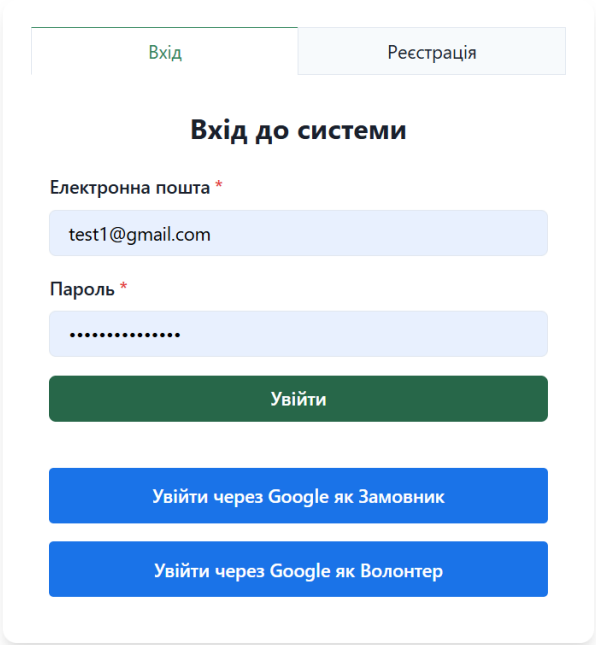

Рисунок 3.1 – Інтерфейс сторінки входу та реєстрації

Для обробки запитів на стороні сервера використовується фреймворк Express.js. Наприклад, обробник створення нового запиту на допомогу реалізовано наступним чином (ліст. 3.2).

Лістинг 3.2 – Створення нового запиту на допомогу (бекенд)

```
router.post("/requests", authenticateToken, async (req, res) => {
  const {
    title,
    category_id,
    description,
    budget,
    priority,
    city,
    region,
  } = req.body;

  try {
    const newRequest = await db.query(
      "INSERT INTO aid_requests (title, category_id, description, budget,
priority, city, region, requester_id, status) VALUES
(?, ?, ?, ?, ?, ?, ?, ?, 'Очікує на виконавця')",
      [
        title,
        category_id,
```

```
        description,
        budget,
        priority,
        city,
        region,
        req.user.id,
      ]
    );
    res.status(201).json({ success: true, request_id:
newRequest.insertId });
  } catch (err) {
    console.error("DB Error:", err);
    res.status(500).json({ message: "Помилка створення запиту" });
  }
});
```

кінець лістингу 3.2

Для збереження та взаємодії з базою даних застосовувалась бібліотека mysql2. Реалізація асинхронної обробки запитів дозволяє масштабувати систему без блокування процесів. Особливу увагу було приділено структурі бази даних, забезпеченню зв’язків між таблицями (користувачі, запити, коментарі, категорії), використанню зовнішніх ключів.

На стороні клієнта користувач має можливість залишати коментарі до запитів (ліст. 3.3).

Лістинг 3.3 – Кнопка додавання коментаря на клієнті

```
<Button
  colorScheme="green"
  onClick={handleAddComment}
  isLoading={submitting}
  loadingText="Надсилання"
>
  Додати
</Button>
```

кінець лістингу 3.3

На рисунку 3.2 зображено інтерфейс функціоналу коментування замовлення.

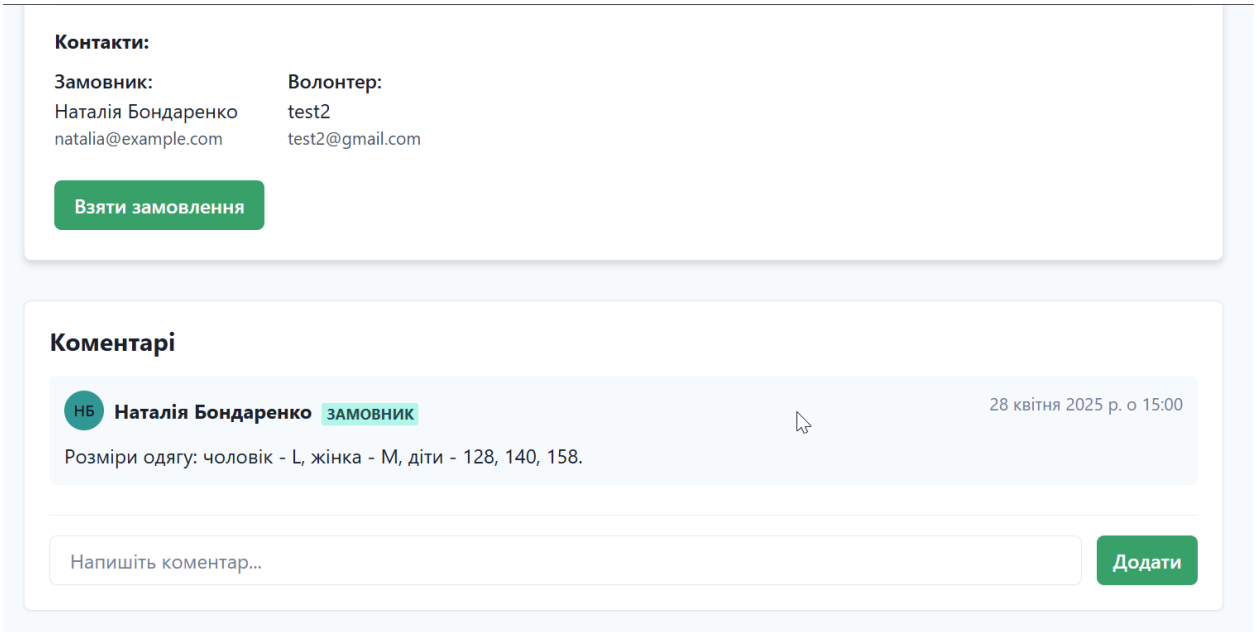

Рисунок 3.2 – Інтерфейс функціоналу коментування замовлення

Усі дані перед відправкою проходять базову валідацію. Для сповіщення користувача про результати дій використовується система toast-повідомлень (ліст. 3.4).

Лістинг 3.4 – Сповіщення про успішну дію

```
toast({
  title: "Коментар додано",
  status: "success",
  duration: 2000,
  isClosable: true,
});
```

кінець лістингу 3.4

Карта замовлень реалізована за допомогою бібліотеки react-leaflet, що дозволяє візуалізувати географічне розташування запитів на допомогу. Маркери формуються на основі координат, отриманих через геокодування API (ліст. 3.5).

Лістинг 3.5 – Візуалізація запиту на карті за допомогою Leaflet

```
<Marker
  key={loc.id}
```

```
  position={loc.coords}
  icon={getCustomIcon(getStatusColor(loc.status), markerSize)}
>
  <Popup>
    <Text>{loc.title}</Text>
  </Popup>
</Marker>
```

кінець лістингу 3.5

На рисунку 3.3 зображено інтерфейс інтерактивної мапи з відображенням замовлень на ній.

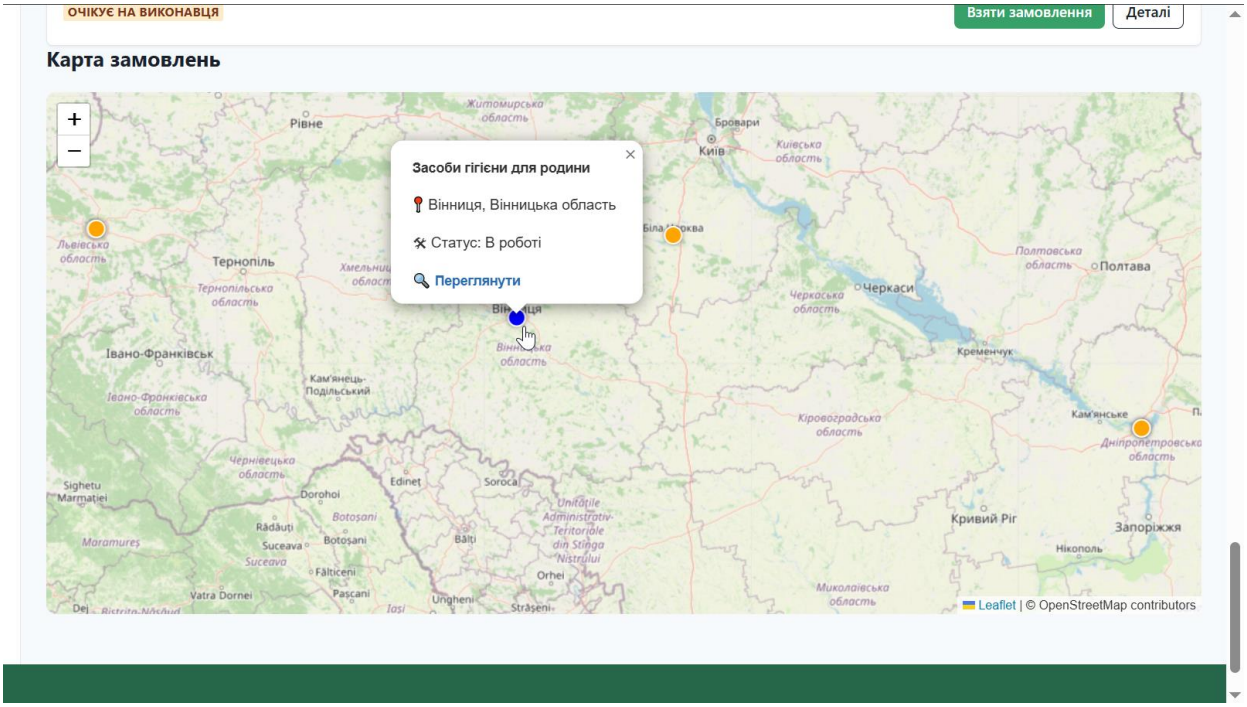

Рисунок 3.3 – Інтерактивна мапа

Окрім базової автентифікації, система підтримує авторизацію з поділом на ролі (волонтер та отримувач допомоги). Це дозволяє обмежувати доступ до певних маршрутів залежно від типу користувача. Наприклад, тільки волонтери можуть брати замовлення, а отримувачі – створювати нові (ліст. 3.6).

Лістинг 3.6 – Обмеження доступу за роллю користувача

```
{authState.user?.role === "volunteer" && request.status === "Очікує на
виконавця" && (
  <Button colorScheme="green" onClick={handleAcceptRequest}>
```

```
    Взяти замовлення
  </Button>
)}
```

кінець лістингу 3.6

На сервері відповідний обробник цього запиту виглядає наступним чином (ліст. 3.7).

Лістинг 3.7 – Призначення волонтера на запит (бекенд)

```
router.put("/requests/:id/assign", authenticateToken, async (req, res) =>
{
  const requestId = req.params.id;
  const volunteerId = req.user.id;

  try {
    await db.query(
      "UPDATE aid_requests SET status = ’В роботі’, volunteer_id = ?
WHERE id = ? AND status = ’Очікує на виконавця’",
      [volunteerId, requestId]
    );

    res.json({ success: true });
  } catch (err) {
    res.status(500).json({ message: "Не вдалося призначити волонтера" });
  }
});
```

кінець лістингу 3.7

Ще одним важливим компонентом є модуль коментарів, що реалізує асинхронне завантаження, перевірку на помилки та рендеринг списку повідомлень. Наприклад, фрагмент завантаження коментарів наведено в лістингу 3.8.

Лістинг 3.8 – Завантаження коментарів асинхронно

```
const fetchComments = async () => {
  try {
    const response = await axios.get(
      `http://localhost:8000/api/v1/requests/${requestId}/comments`
    );

     setComments(response.data.comments);
```

```
  } catch (err) {
    setError("Не вдалося завантажити коментарі");
  }
};
```

кінець лістингу 3.8

Сторінка з картою замовлень дозволяє волонтерам знаходити потрібні запити за географічним принципом. Щоб уникнути накладання маркерів у межах одного міста, застосовується алгоритм зміщення координат (ліст. 3.9).

Лістинг 3.9 – Алгоритм уникнення накладання маркерів

```
const generateOffset = (index) => {
  const angle = index * 45 * (Math.PI / 180); // кут у радіанах
  const radius = 0.004; // радіус зміщення
  return [radius * Math.cos(angle), radius * Math.sin(angle)];
};
```

кінець лістингу 3.9

Цей підхід дозволяє візуально розділити запити, що мають однакові координати (наприклад, у межах одного населеного пункту).

Загалом, застосунок побудований за сучасними архітектурними підходами, демонструє повноцінну реалізацію API, інтеграцію бази даних, управління станом користувача та інтерфейсом, а також візуалізацію геопросторових даних. Усі ці рішення реалізовані засобами мови JavaScript та технологій на її основі – що підтверджує практичну компетентність здобувача в розробці інформаційних систем.

3.2 Тестування та налагодження інформаційно-комп’ютерної системи

У процесі розробки платформи SupportFlow було реалізовано поетапне тестування та налагодження кожного з модулів як серверної, так і клієнтської частини системи. Основна мета полягала у виявленні помилок на ранніх стадіях, забезпеченні стабільності роботи та перевірці логіки взаємодії з базою даних.

Застосовано такі методи тестування:

1) ручне тестування для перевірки основних сценаріїв користувача: реєстрація, вхід, створення запиту, призначення волонтера, коментування, перегляд на мапі;

2) юніт-тестування окремих функцій серверної частини з використанням бібліотеки mocha та chai;

3) інтеграційне тестування, яке зосереджене на перевірці коректності взаємодії між модулями Node.js і базою MySQL;

4) функціональне тестування UI на основі інструментів браузера та React Testing Library (в обмеженому обсязі).

Для запуску платформи на локальному сервері або хостингу рекомендуються наступні параметри:

1) операційна система: Windows 10 або Ubuntu 20.04+;

2) оперативна пам’ять: мінімум 4 ГБ (рекомендовано – від 8 ГБ);

3) процесор: двоядерний з частотою не менше 2 ГГц;

4) браузер: Chrome, Firefox або Edge (останні версії);

5) node.js: версія 18.x і вище;

6) MySQL Server: версія 8.0 і вище;

7) пакетний менеджер: npm або yarn.

У процесі тестування було виявлено та виправлено деякі помилки (табл. 3.1).

Таблиця 3.1 – Виявлені недоліки та їх усунення

№	Виявлений недолік	Спосіб усунення
1	Неправильна обробка помилки при вході з неправильним паролем	Додано обробку статусу 401 на клієнті та реалізовано відображення toast-повідомлення
2	Можливість повторного прийняття одного й того ж запиту волонтером	Додано перевірку статусу на сервері: якщо запит уже «в роботі», сервер повертає помилку
3	Накладання маркерів на мапі при однакових координатах запитів	Реалізовано функцію generateOffset(index) для зміщення маркерів у межах одного населеного пункту

Приклад юніт-тесту для бекенду наведено в лістингу 3.10. Цей тест перевіряє створення нового запиту на допомогу через API бекенду. За допомогою бібліотек chai та chai-http імітується POST-запит із відповідними

параметрами та заголовком авторизації. Тест очікує відповідь зі статусом 201 і підтвердженням успішного створення запиту у вигляді success: true.

Лістинг 3.10 – Юніт-тест для бекенду

```
const chai = require("chai");
const chaiHttp = require("chai-http");
const server = require("../app");
chai.use(chaiHttp);

describe("Aid Request API", () => {
  it("should create a new aid request", (done) => {
    chai
      .request(server)
      .post("/api/v1/requests")
      .set("Authorization", "Bearer testtoken")
      .send({
        title: "Потрібна допомога з їжею",
        category_id: 1,
        description: "Для літньої людини у Харкові",
        budget: 0,
        priority: "Високий",
        city: "Харків",
        region: "Харківська",
      })
      .end((err, res) => {
        chai.expect(res).to.have.status(201);
        chai.expect(res.body).to.have.property("success", true);
        done();
      });
  });
});
```

кінець лістингу 3.10

Приклад юніт-тесту валідації на клієнті наведено в лістингу 3.11. Цей тест, реалізований засобами @testing-library/react, перевіряє валідацію вводу на клієнті в компоненті CommentSection. Він рендерить компонент з конкретними параметрами і перевіряє, чи кнопка «Додати» неактивна (disabled), якщо поле коментаря порожнє. Це дозволяє запобігти надсиланню порожніх повідомлень і забезпечує базову валідацію інтерфейсу без потреби в серверному зверненні.

Лістинг 3.11 – Юніт-тест для фронтенду

```
import { render, screen, fireEvent } from "@testing-library/react";
import CommentSection from "./CommentSection";

test("disables send button on empty comment", () => {
  render(<CommentSection requestId={1} status="Очікує на виконавця" />);
  const button = screen.getByRole("button", { name: /додати/i });
  expect(button).toBeDisabled();
});
```

кінець лістингу 3.11

Усі функції проходили тестування після кожного етапу імплементації. Це дозволило уникнути серйозних помилок при запуску MVP та забезпечити стабільну роботу в продуктивному середовищі. Платформа успішно проходить базові smoke-тести при кожному розгортанні.

3.3 Розробка бази даних

Для зберігання інформації про запити на допомогу, користувачів, категорії та коментарі було спроєктовано реляційну базу даних humanitarian_aid_db. Проєктування здійснювалось у середовищі MySQL Workbench, що дозволяє створювати візуальні схеми, встановлювати зовнішні ключі та генерувати SQL-код автоматично.

Варто зазначити, що для адміністративного управління й роботи з даними використовувався також phpMyAdmin. На рисунку 3.4 наведено фізичну модель бази даних.

Базу даних складають чотири основні таблиці:

1) users – зберігає дані користувачів (отримувачів і волонтерів);
2) aid_requests – запити на допомогу, створені користувачами;
3) categories – перелік категорій запитів;
4) comments – коментарі до запитів, які залишають користувачі.

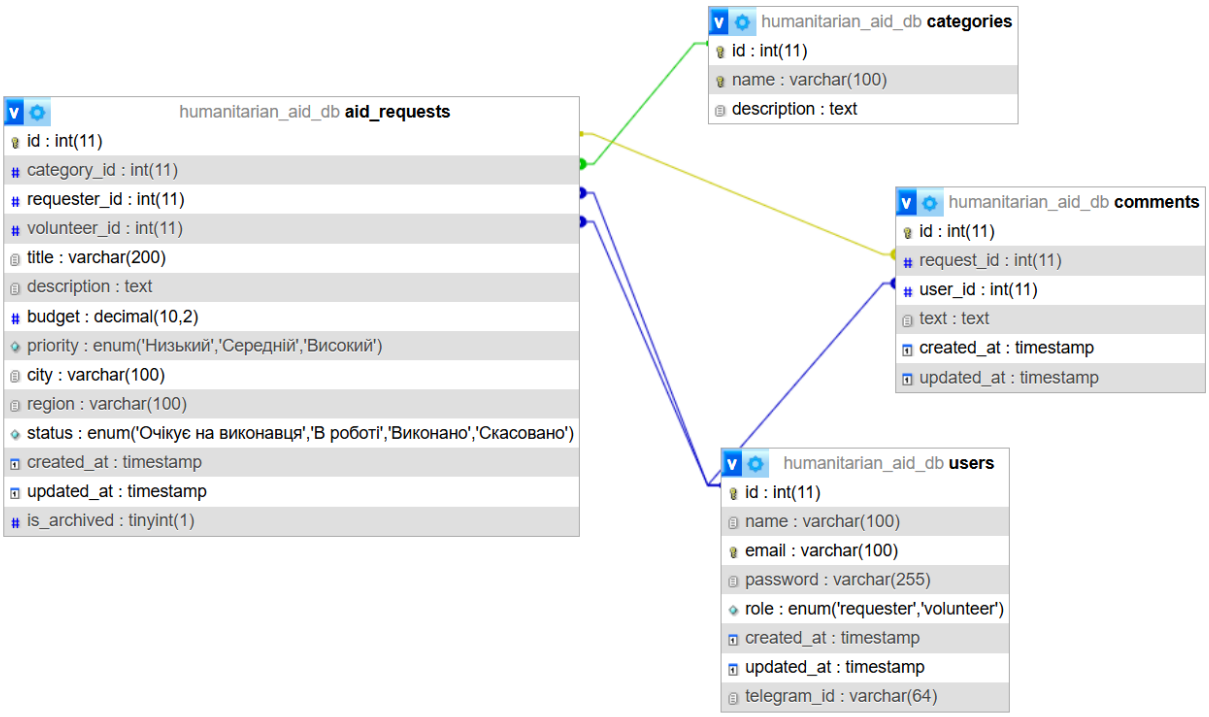

Рисунок 3.4 – Діаграма бази даних, створена за допомогою phpMyAdmin

Між таблицями встановлено зв’язки «один до багатьох», наприклад:

1) один користувач може створити багато запитів (users.id → aid_requests.requester_id);

2) один запит може мати багато коментарів (aid_requests.id → comments.request_id);

3) один користувач може залишати багато коментарів (users.id → comments.user_id).

Наведемо приклади SQL-запитів. Створення таблиці користувачів наведено в лістингу 3.12.

Лістинг 3.12 – Створення таблиці користувачів

```
CREATE TABLE users (
  id INT AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(100) NOT NULL,
  email VARCHAR(100) NOT NULL UNIQUE,
  password VARCHAR(255),
  role ENUM(’requester’, ’volunteer’) NOT NULL,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
```

```
  updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
  telegram_id VARCHAR(64)
);
```

кінець лістингу 3.12

Створення таблиці запитів на допомогу наведено в лістингу 3.13.

Лістинг 3.13 – Створення таблиці запитів

```
CREATE TABLE aid_requests (
  id INT AUTO_INCREMENT PRIMARY KEY,
  category_id INT NOT NULL,
  requester_id INT NOT NULL,
  volunteer_id INT,
  title VARCHAR(200) NOT NULL,
  description TEXT,
  budget DECIMAL(10,2),
  priority ENUM('Низький','Середній','Високий'),
  city VARCHAR(100),
  region VARCHAR(100),
  status ENUM('Очікує на виконавця','В роботі','Виконано','Скасовано')
DEFAULT 'Очікує на виконавця',
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
  is_archived TINYINT(1) DEFAULT 0,
  FOREIGN KEY (category_id) REFERENCES categories(id),
  FOREIGN KEY (requester_id) REFERENCES users(id),
  FOREIGN KEY (volunteer_id) REFERENCES users(id)
);
```

кінець лістингу 3.13

Приклад запиту на виведення всіх активних запитів наведено в лістингу 3.14.

Лістинг 3.14 – Виведення активних запитів

```
SELECT * FROM aid_requests
WHERE status != 'Скасовано';
```

кінець лістингу 3.14

Запит для перегляду коментарів до конкретного запиту наведено в лістингу 3.15.

Лістинг 3.15 – Перегляд коментарів конкретного запиту

```
SELECT c.text, u.name, c.created_at
FROM comments c
JOIN users u ON c.user_id = u.id
WHERE c.request_id = 1
ORDER BY c.created_at DESC;
```

кінець лістингу 3.15

Запит на додавання нового коментаря наведено в лістингу 3.16.

Лістинг 3.16 – Запит додавання коментаря

```
INSERT INTO comments (request_id, user_id, text)
VALUES (1, 2, 'Можу допомогти з транспортом');
```

кінець лістингу 3.16

Усі зовнішні ключі були реалізовані через відповідні SQL-конструкції FOREIGN KEY, що дозволяє забезпечити цілісність даних між таблицями. Приклад наведено в лістингу 3.17.

Лістинг 3.17 – Зовнішні ключі

```
FOREIGN KEY (request_id) REFERENCES aid_requests(id),
FOREIGN KEY (user_id) REFERENCES users(id)
```

кінець лістингу 3.17

Це гарантує, що жоден коментар не буде прикріплено до неіснуючого запиту або користувача. В цілому, база даних була розроблена відповідно до нормалізованої структури, що дозволяє ефективно зберігати, шукати та обробляти інформацію про волонтерську допомогу. Структура підтримує масштабованість системи та є зручною для інтеграції з вебінтерфейсом і API.

3.4 Захист інформаційно-комп’ютерної системи

У розробленій платформі координації волонтерської допомоги реалізовано низку базових, але ефективних заходів інформаційної безпеки, спрямованих на забезпечення конфіденційності, цілісності та доступності даних. Основна увага приділяється автентифікації, авторизації, шифруванню паролів та захисту API-ендпоінтів.

Система підтримує автентифікацію користувачів двома способами:

1) через email і пароль;

2) через зовнішній сервіс Google OAuth 2.0.

Для захисту закритих маршрутів використовується механізм JWT (JSON Web Token). Після входу користувача сервер генерує токен, який містить інформацію про ID, роль користувача та термін дії. Цей токен передається у заголовках Authorization: Bearer ... при наступних запитах.

На стороні сервера запити до захищених маршрутів перевіряються за допомогою middleware authenticateToken, який декодує токен і перевіряє його дійсність. У випадку відсутності або некоректності токена сервер повертає статус 401 Unauthorized (ліст. 3.18).

Лістинг 3.18 – Перевірка токена

```
function authenticateToken(req, res, next) {
  const authHeader = req.headers["authorization"];
  const token = authHeader && authHeader.split(" ")[1];
  if (!token) return res.sendStatus(401);

  jwt.verify(token, process.env.JWT_SECRET, (err, user) => {
    if (err) return res.sendStatus(403);
    req.user = user;
    next();
  });
}
```

кінець лістингу 3.18

Паролі користувачів ніколи не зберігаються у відкритому вигляді. Для хешування використовується бібліотека bcrypt. Перед збереженням у базу

даних пароль хешується за допомогою алгоритму bcrypt із сіллю (salt), що забезпечує захист від атак типу «brute force» та «rainbow table». Приклад використання bcrypt при реєстрації користувача наведено в лістингу 3.19.

Лістинг 3.19 – Використання bcrypt при реєстрації користувача

```
const hashedPassword = await bcrypt.hash(password, 10);
await db.query("INSERT INTO users (name, email, password, role) VALUES
(?, ?, ?, ?)", [
  name,
  email,
  hashedPassword,
  role
]);
```

кінець лістингу 3.19

А при вході відбувається перевірка пароля (ліст. 3.20).

Лістинг 3.20 – Перевірка пароля при вході

```
const isMatch = await bcrypt.compare(inputPassword, storedHashedPassword);
if (!isMatch) return res.status(401).json({ message: "Невірний пароль" });
```

кінець лістингу 3.20

Система реалізує рольову модель доступу. Кожен користувач має роль: requester або volunteer. Відповідно до ролі обмежується доступ до певного функціоналу:

1) лише requester може створювати нові запити;
2) лише volunteer може брати запити в роботу;
3) спроби порушення обмежень блокуються як на клієнті, так і на сервері.

Також усі важливі маршрути захищені авторизацією, а валідація вхідних даних запобігає SQL-ін’єкціям (запити виконуються з параметрами). Отже, у проєкті реалізовано повноцінний рівень захисту, достатній для MVP-версії інформаційної системи, з урахуванням автентифікації, авторизації, хешування паролів та контролю доступу до функціоналу відповідно до ролей користувачів.

3.5 Розробка документації для програмного продукту

Для стабільної роботи розробленої інформаційно-комп'ютерної системи SupportFlow необхідно дотримання наступних системних вимог (табл. 3.2).

Таблиця 3.2 – Мінімальні системні вимоги

Компонент	Мінімальні вимоги
Операційна система	Windows 10 / Ubuntu 20.04+ / macOS 10.15+
ОЗП (RAM)	4 GB
Вільне місце на диску	300 MB
Версія Node.js	16.0 або вище
Версія MySQL	8.0 або вище
Браузер	Google Chrome, Firefox (останні версії)

Користувачам доступна інтегрована довідкова система у вигляді:

1) інтерфейсних підказок – текстові інструкції, які з'являються біля елементів форми (наприклад, при заповненні полів замовлення);

2) toast-повідомлень – сповіщення у правому верхньому куті, що інформують про результат дії (успішна реєстрація, помилка, збереження даних);

3) структуроване меню – бокове меню з логічним поділом: «Всі замовлення», «Мої запити», «Профіль», «Вийти»;

4) онбординг (початкові підказки) –система показує пояснення до основних дій.

Опишемо основні кроки встановлення і розгортання системи:

1) встановити Node.js LTS;

2) встановити MySQL Server;

3) створити базу даних humanitarian_aid_db та імпортувати SQL-структуру;

4) відкрити папку проєкту в середовищі розробки;

5) налаштувати серверну частину за допомогою виконання команди npm install в папці backend;

6) створити .env файл зі змінними (ліст. 3.21);

Лістинг 3.21 – Вміст файлу .env

```
PORT=8000
DB_HOST=localhost
DB_USER=root
DB_PASSWORD=пароль
DB_NAME=humanitarian_aid_db
GOOGLE_CLIENT_ID=...
GOOGLE_CLIENT_SECRET=...
```

кінець лістингу 3.21

7) запустити сервер використовуючи npm run start;

8) налаштування клієнтської частини (frontend) тією є командою – npm install в папці фронтенду;

9) запустити фронтенд командою npm start;

10) frontend працює на http://localhost:5170, Backend – на http://localhost:8000.

З'єднання з MySQL здійснюється через бібліотеку mysql2, конфігурація задається у db.js. Підключення захищене, виконується через пул з обробкою помилок.

Інтеграція плагінів:

1) Chakra UI – для інтерфейсу користувача (npm install @chakra-ui/react);

2) React Leaflet – для інтерактивної мапи (npm install react-leaflet leaflet);

3) Axios – для HTTP-запитів (npm install axios);

4) React Router DOM – для маршрутизації сторінок (npm install react-router-dom);

5) Google OAuth – для входу через Google акаунт (passport-google-oauth20 на сервері).

Для оновлення достатньо замінити файли в директорії frontend та backend, після чого виконати npm install і перезапустити сервер. База даних не потребує зміни при оновленнях без модифікацій структури. Користувачам із правами адміністратора рекомендується періодично експортувати таблиці users, aid_requests, comments за допомогою mysqldump або засобами MySQL Workbench.

Висновки до розділу 3

У межах третього розділу було безпосередньо реалізовано інформаційно-комп’ютерну систему, що виконує функції платформи для координації волонтерської допомоги. Проєкт створено із дотриманням принципів модульності, безпеки, масштабованості та зручності використання. Практична реалізація охоплює повноцінний frontend на базі React.js, backend-логіку на Node.js із використанням Express.js, а також реляційну базу даних MySQL для зберігання інформації про запити, користувачів, ролі, коментарі тощо.

На клієнтській стороні створено зручний і адаптивний інтерфейс користувача за допомогою бібліотеки Chakra UI. Інтерфейс підтримує різні сценарії взаємодії – від створення запитів до їхнього призначення волонтерами. Також реалізовано інтерактивну карту з маркерами на основі координат запитів, що значно підвищує зручність пошуку допомоги за географічним принципом.

Серверна частина включає реалізацію REST API з авторизацією через JWT-токени, підтримкою Google OAuth, валідацією запитів і контролем доступу на основі ролей користувачів. Враховано основні принципи безпеки: реалізовано хешування паролів за допомогою bcrypt, захист API від несанкціонованого доступу, а також перевірку статусів для запобігання дублюванню дій з боку волонтерів.

У процесі розробки проводилось тестування системи: функціональне, юніт-тестування окремих компонентів, а також ручне тестування інтерфейсу. У результаті тестування виявлено низку технічних недоліків, серед яких – некоректна обробка помилок при авторизації, дублювання прийняття запитів волонтерами, накладання маркерів на мапі. Всі ці недоліки були своєчасно усунуті шляхом модифікації клієнтської та серверної логіки.

Базу даних розроблено з урахуванням зв’язків між сутностями, нормалізації, використання зовнішніх ключів для забезпечення цілісності даних. Реалізовано зручну схему фільтрації запитів, підтримку категорій, коментарів, рейтингових систем тощо.

Також створено супровідну документацію до системи, в якій зазначено мінімальні вимоги до серверного середовища, кроки встановлення, налаштування середовища виконання, підключення до бази даних та запуску клієнтської та серверної частин. Додатково в інтерфейс інтегровано довідкові повідомлення, підказки, toast-сповіщення, що підвищує загальний рівень user experience.

У підсумку, реалізована платформа відповідає поставленим вимогам, демонструє сучасний рівень веброзробки та може бути використана в реальному середовищі для забезпечення ефективної взаємодії між волонтерами й отримувачами допомоги. Розробка підтвердила практичну компетентність здобувача в проєктуванні та створенні складних інформаційних систем.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи бакалавра було повністю реалізовано поставлену мету – створення ефективної вебплатформи для координації волонтерської допомоги з інтеграцією геолокації, коментарів, авторизації та ролей користувачів. На основі аналізу існуючих рішень та сучасних викликів у сфері волонтерських ініціатив було визначено актуальність проблеми та доцільність її реалізації саме за допомогою інформаційно-комп'ютерної системи. Створений програмний продукт відповідає сучасним вимогам гнучкості, безпеки та масштабованості, що підтверджується його реалізацією з використанням стеку технологій React, Node.js, Express, MySQL, JWT, Chakra UI, Axios, Google OAuth та інших.

В процесі виконання роботи було розкрито теоретичні аспекти проєктування інформаційно-комунікаційної системи для координації волонтерської допомоги, а також проведено порівняльний аналіз сучасних вебсистем, зокрема «Help Ukraine Center», «Українська Волонтерська Служба» та «UkraineNow». На основі отриманих результатів було обґрунтовано доцільність створення власної платформи з урахуванням специфіки потреб українських користувачів і соціального контексту воєнного стану.

Далі було чітко сформульовано мету та завдання кваліфікаційної роботи, що стали основою для подальшої розробки вебплатформи підтримки громадських ініціатив. Визначено основні функціональні вимоги, очікувані результати та критерії оцінки ефективності проєкту. Здійснено детальний аналіз функціональних і нефункціональних вимог до системи, побудовано UML-діаграми (діаграма варіантів використання, класів, розгортання) та визначено архітектуру майбутнього застосунку. Це дозволило на етапі проєктування закласти гнучку і масштабовану структуру ПЗ.

Також було обґрунтовано вибір інструментів і технологій для реалізації вебзастосунку, серед яких React.js для клієнтської частини, Node.js + Express.js для серверної, а також MySQL як основа для збереження структурованих даних.

Перевага цим інструментам надана через їхню масштабованість, відкритість, активну спільноту й можливість швидкої інтеграції з іншими сервісами (наприклад, Google OAuth, Leaflet, Chakra UI).

Далі було описано реалізацію основного функціоналу, зокрема реєстрації, авторизації (у тому числі через Google), створення, перегляду й призначення заявок, фільтрації за статусом і категоріями. Детально описано реалізовані бібліотеки та модулі, зокрема JWT для захисту маршрутизованих запитів, bcrypt для хешування паролів, Leaflet для інтерактивної карти, а також логіку зв'язку між фронтендом і бекендом. Розглянуто алгоритм роботи рекомендаційної системи на основі контентної фільтрації. Було реалізовано модуль, який порівнює параметри запитів користувача з іншими доступними запитами за категорією, регіоном і ключовими словами, забезпечуючи персоналізовані пропозиції. Такий механізм підвищує ефективність взаємодії користувачів з платформою, сприяє швидшому призначенню волонтерів на актуальні запити.

Також проведено модульне тестування основних компонентів системи з використанням бібліотек chai, mocha, jest, @testing-library/react, що дозволило перевірити коректність API-запитів, валідації форм та інтеграційної логіки. Крім того, було розглянуто навантажувальне тестування й запропоновано шляхи оптимізації, зокрема впровадження кешування координат геолокацій для підвищення швидкодії карти.

Описано процес проєктування та реалізації реляційної бази даних MySQL для зберігання інформації про запити, користувачів, коментарі, ролі та геодані. Розроблено логічну та фізичну моделі, створено ефективну структуру взаємозв’язків між сутностями.

Розглянуто питання інформаційної безпеки: впроваджено механізми автентифікації (JWT та OAuth), обмеження доступу за ролями, захист API-запитів, а також вжито заходів щодо запобігання атакам типу XSS та CSRF. Це підвищує загальний рівень надійності системи.

Також було розроблено супровідну документацію до програмного продукту, включно з технічними характеристиками, інструкціями з інсталяції та використання, а також довідковими матеріалами, що інтегровані в інтерфейс у вигляді підказок та онбордингу для нових користувачів.

Відповідно, сформовано загальні висновки щодо досягнення цілей проєкту, надано оцінку результатам і окреслено перспективи подальшого розвитку системи. Розроблений застосунок має чітко визначену соціальну значущість, може бути масштабований до рівня державної або міжнародної платформи, а також доповнений мобільним клієнтом, механізмами сповіщення та аналітики.

Проведено апробацію проєкту шляхом презентації тез на X Міжнародній науково-практичній конференції «ІТОНВ-2025» (м. Луцьк), що свідчить про науково-практичну значущість розробки.

У процесі апробації та тестування було досягнуто високих якісних показників: стабільна робота при одночасному доступі кількох користувачів, успішна обробка типових сценаріїв, надійне збереження та обробка даних, що засвідчує практичну готовність системи до розгортання у волонтерських організаціях. Реалізація гнучкого інтерфейсу та авторизації надає користувачам інтуїтивно зрозумілий доступ до всіх функцій.

Результати цієї роботи мають значний потенціал для використання в громадському секторі, зокрема в організаціях, що займаються гуманітарною допомогою, соціальною підтримкою, взаємодією з переселенцями або під час кризових ситуацій. Матеріали дослідження можуть бути використані як основа для створення розширених систем з аналітикою, рекомендаційною підтримкою на основі ШІ або мобільними додатками.

Подальші напрями вдосконалення включають реалізацію push-сповіщень, аналітичного кабінету для адміністраторів, мультимовної підтримки, розширеного профілю волонтера, системи рейтингу та гейміфікації взаємодії. Також перспективним є впровадження мікросервісної архітектури та інтеграції

з реальними платформами обліку гуманітарної допомоги (наприклад, через API міжнародних фондів).

Робота узгоджена з тематиками кафедральних досліджень і демонструє практичне застосування знань, отриманих у процесі навчання. У тезах доповіді, презентованих на конференції, було акцентовано увагу на інноваційній ролі вебтехнологій у підтримці громадських ініціатив, що засвідчує суспільну й академічну значущість отриманих результатів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Горінов П., Драпушко Р. Волонтерська діяльність в Україні: соціально-правове дослідження : монографія. Київ : Держ. ін-т сімейн. та молодіж. політики, 2022. 240 с.

2. Закалик Г. М., Коропатов О. М., Шувар Н. М. Волонтерська діяльність в Україні: історія, сьогодення та євроінтеграція. Правова система: теорія і практика. 2022. Т. 1, № 4. С. 104-110. URL: https://doi.org/10.32850/sulj.2022.4.1.17 (дата звернення: 03.03.2025).

3. Яремчук С. С., Семенюк-Прибатень А. В., Адамовський В. І. Правові інструменти розвитку волонтерства в Україні: на виклики воєнного часу. Академічні візії. 2023. № 17. 12 с. URL: http://dx.doi.org/10.5281/zenodo.7730899 (дата звернення: 04.03.2025).

4. Доценко С. О. Виховання в епоху цифровізації. Spiritual-intellectual upbringing and teaching of youth in the 21st century. 2022. № 4. С. 311–314. URL: https://doi.org/10.34142//2708-4809.siuty.2022.75 (дата звернення: 05.03.2025).

5. Суботін І. Л., Трунов О. І. Розробка сучасної інформаційної системи забезпечення волонтерської діяльності. Редакційна колегія, 2024. С. 57-60.

6. Help Ukraine Center. URL: https://helpukraine.center/ (date of access: 03.03.2025).

7. Українська Волонтерська Служба. URL: https://volunteer.country/ (дата звернення: 03.03.2025).

8. UkraineNow – Help Ukraine. URL: https://www.ukrainenow.org/ (date of access: 03.03.2025).

9. VolunteerMatch – Where Volunteering Begins. URL: https://www.volunteermatch.org/ (date of access: 04.03.2025).

10. GivePulse : Giving Platform supporting your community and your causes. URL: https://learn.givepulse.com/ (date of access: 04.03.2025).

11. Головна. Be My Eyes. URL: https://www.bemyeyes.com/uk/ (дата звернення: 04.03.2025).

12. Rumpe B. Modeling with UML. Berlin/Heidelberg, Germany: Springer, 2016. Vol. 98. 281 p.

13. React. URL: https://react.dev/ (date of access: 01.04.2025).

14. Lazuardy M. F. S., Anggraini D. Modern front end web architectures with react. js and next. js. Research Journal of Advanced Engineering and Science. 2022. Vol. 7, No. 1. P. 132-141.

15. Angular vs React: як обрати відповідний фреймворк під проєкт. FoxmindEd. URL: https://foxminded.ua/angular-vs-react/ (дата звернення: 01.04.2025).

16. Node.js – Run JavaScript Everywhere. URL: https://nodejs.org/en (date of access: 01.04.2025).

17. Gascón U. Node. js for Beginners. A comprehensive guide to building efficient, full-featured web applications with Node. js. Packt Publishing Ltd, 2024. 382 p.

18. Hoque S. Full-Stack React Projects: Learn MERN stack development by building modern web apps using MongoDB, Express, React, and Node. js. Packt Publishing Ltd, 2020. 716 p.

19. MySQL Workbench. MySQL. URL: https://www.mysql.com/products/workbench/ (date of access: 03.04.2025).

20. Database Management Education in MYSQL / J. Wahyudi et al. Edumaspul: Jurnal Pendidikan. 2022. Vol. 6, no. 2. P. 2413–2417. URL: https://doi.org/10.33487/edumaspul.v6i2.4570 (date of access: 03.04.2025).

21. Enhancing JWT Authentication and Authorization in Web Applications Based on User Behavior History / A. Bucko et al. Computers. 2023. Vol. 12, no. 4. P. 78. URL: https://doi.org/10.3390/computers12040078 (date of access: 05.04.2025).

22. Prakash V., Dua K., Garg L., Shukla V. Web Application Authentication Using Google OAuth, Express, and MongoDB. In International Conference on Cryptology & Network Security with Machine Learning. Singapore: Springer Nature Singapore, 2023. P. 645-657.

23. Chakra UI. URL: https://chakra-ui.com/ (date of access: 15.04.2025).

24. Grippa V. M., Kuzmichev S. Learning MySQL. O’Reilly Media, Inc., 2021. 632 p.

25. Gascón U. Node. js for Beginners: A comprehensive guide to building efficient, full-featured web applications with Node. js. Packt Publishing Ltd, 2024. 382 p.

ДОДАТКИ

Додаток А

Тези X Міжнародної науково-практичній конференції з проблем вищої освіти і науки «Інформаційні технології в освіті, науці і виробництві (ІТОНВ-2025)» м. Луцьк, 23-24 травня 2025 р.

Міністерство освіти і науки України
Волинська обласна рада
Луцька міська рада
Луцький національний технічний університет
Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського»
Національний університет «Львівська політехніка»
Військовий інститут телекомунікацій та інформатизації імені Героїв Крут (м. Київ)
Тернопільський національний педагогічний університет імені В. Гнатюка
Рівненський державний гуманітарний університет
Навчально-науковий інститут «Українська інженерно-педагогічна академія» Харківського національного університету ім. В.Н. Каразіна
Люблінська політехніка (Польща)
Фрайберзька гірнича академія (Німеччина)
Гронінгенський університет (Нідерланди)
Кавказький університет (Грузія)
Політехнічний університет Браганси (Португалія)

ІТОНВ

ТЕЗИ ДОПОВІДЕЙ
X Міжнародної науково-практичної конференції з проблем вищої освіти і науки «Інформаційні технології в освіті, науці і виробництві (ІТОНВ-2025)

23-24 травня 2025 р.

Луцьк 2025

2. Міронов Н. О., Самчук Л. М. Дослідження характеристик та медотології системи розпізнавання жестів для безконтактної взаємодії з комп'ютером з використанням технологій ComputerVision. Комп'ютерно-інтегровані технології: освіта, наука, виробництво. 57. 2024, 111-119. URL:https://doi.org/10.36910/6775-2524-0560-2024-57-13.

3. Mr.E.Sankar, B.NitishBharadwaj, A.V.Vignesh. Virtual Mouse Using Hand Gesture. International Journal of Scientific Researchin Engineering and Management. 7, 5. 2023, 2-4. URL:https://www.researchgate.net/publication/372165002.

4. MediaPipe Hands.https://mediapipe/solutions/hands.html

5. P. J. Ezigbo, O. C. Nosiri, E. S. Mbonu, V. Ofor, J. Obichere. HandGestureRecognitionandControl for Human-RobotInteractionUsingDeepLearning. International Journal of Electrical and Electronic Engineering&Telecommunications. 12, 6. 2023, 433-441. URL: https://www.researchgate.net/publication/375422572

УДК 004.738.5:316.42

РОЛЬ ВЕБТЕХНОЛОГІЙ У РОЗВИТКУ ГРОМАДСЬКИХ ІНІЦІАТИВ

Надашкевич Анастасія Геннадіївна
Луцький національний технічний університет,
студентка 4 курсу, nastia.nadashkevich@gmail.com
Вознюк Анастасія Вадимівна
Луцький національний технічний університет, асистент кафедри інженерії програмного забезпечення, a.vozniuk@lutsk-ntu.com.ua

THE ROLE OF WEB TECHNOLOGIES IN THE DEVELOPMENT OF CIVIC INITIATIVES

Nadashkevych Anastasiia Hennadiivna
Lutsk National Technical University, 4th-year student,
nastia.nadashkevich@gmail.com
Vozniuk Anastasiia Vadymivna
Lutsk National Technical University, Assistant Professor, Department of Software Engineering, a.vozniuk@lutsk-ntu.com.ua

This paper explores the role of modern web technologies in supporting civic initiatives. It highlights the advantages of digital platforms for volunteer coordination, fundraising, and community engagement, as well as the challenges of implementation such as cybersecurity and digital inequality.

Keywords: *web technologies, civic engagement, digital platforms, volunteer coordination, open source, cybersecurity, digital inclusion.*

У сучасному суспільстві громадські ініціативи відіграють важливу роль у вирішенні соціальних, гуманітарних та екологічних проблем, особливо в умовах кризових ситуацій, як-от війна, пандемії чи природні катастрофи. Ефективність таких ініціатив значною мірою залежить від швидкої координації дій, доступу до актуальної інформації та залучення широкого кола учасників. Вебтехнології стали ефективним інструментом для організації громадської активності, дозволяючи створювати масштабовані платформи для взаємодії між волонтерами, організаціями та тими, хто потребує допомоги. Цей напрям є особливо актуальним в Україні, де волонтерський рух став невід'ємною частиною громадянського суспільства.

У XXI столітті вебтехнології стали ефективним ресурсом для мобілізації громадянського суспільства. Вони дозволяють не лише швидко поширювати інформацію, а й об'єднувати людей навколо важливих соціальних, гуманітарних чи екологічних ініціатив [1]. Серед основних переваг таких рішень — доступність з будь-якого пристрою, висока швидкість реагування, можливість інтерактивної взаємодії між користувачами та здатність масштабуватися під потреби великої кількості учасників.

Діаграма на рисунку 1 ілюструє позитивну динаміку зростання цифрових навичок серед дорослого населення України упродовж 2019–2023 років. Це підтверджує актуальність теми впровадження вебтехнологій у громадську діяльність, адже зростання цифрової грамотності розширює коло потенційних учасників волонтерських та соціальних ініціатив, які можуть ефективно користуватись онлайн-платформами.

Рисунок 1 – Динаміка зростання цифрових навчичок дорослого населення [2]

Вебплатформи сприяють формуванню ефективної цифрової інфраструктури для громадської участі. Зокрема, краудсорсингові сервіси дозволяють збирати ідеї або ресурси від широкої спільноти для реалізації соціально важливих проєктів. Волонтерські хаби в онлайн-форматі допомагають організовувати роботу добровольців — від реєстрації до розподілу завдань [3]. Платформи донатів (наприклад, Patreon, DonatePay, Добро.ua) забезпечують фінансову підтримку активістів, а петиційні сайти (change.org, Єдина система місцевих петицій) — інструменти громадського тиску та комунікації з владою.

Особливе значення має інтеграція з соціальними мережами та використання відкритих API. Це дозволяє не лише швидко поширювати інформацію, а й обробляти її в реальному часі, створюючи більш гнучкі та адаптивні системи координації. Наприклад, через Telegram-ботів чи інтеграцію з Google Maps можна спростити логістику, координувати запити або показувати активні точки допомоги на мапі. Загалом, сучасні вебтехнології відкривають нові горизонти для ефективної, швидкої та прозорої громадської активності, створюючи

цифрову екосистему, де кожен охочий може долучитися до позитивних змін.

Запровадження вебтехнологій у громадську діяльність відкриває широкий спектр можливостей для підвищення ефективності та прозорості ініціатив [4]. Серед основних технічних переваг варто відзначити автоматизацію процесів, що дозволяє значно зменшити людський фактор при розподілі ресурсів, комунікації та обробці заявок. Завдяки використанню баз даних та аналітичних інструментів, вебплатформи здатні працювати з великими обсягами інформації — від координат гуманітарних штабів до реєстрів волонтерів або запитів на допомогу.

Втім, із розвитком цифрової інфраструктури постають і серйозні виклики. По-перше, це питання кібербезпеки: вебплатформи, що працюють із персональними даними користувачів, потребують захисту від атак, витоків або підробок інформації. По-друге, цифрова нерівність залишається бар'єром для повноцінної участі частини населення — зокрема, людей старшого віку, мешканців сільських територій або осіб із обмеженим доступом до інтернету. Також важливою проблемою є довіра до систем, особливо у випадках, коли платформи діють без офіційного визнання або належної публічної звітності.

У відповідь на ці виклики спільноти часто використовують відкритий код для забезпечення прозорості та можливості перевірки роботи системи незалежними експертами. Також розробники вдаються до створення адаптивних інтерфейсів, які зручні для користувачів з різним рівнем цифрової грамотності, або до мультиканальної підтримки — наприклад, через SMS, месенджери, чат-боти. Це дозволяє розширити охоплення та зменшити цифровий розрив. В цілому, впровадження вебрішень у сфері громадської активності є складним, але перспективним процесом, що потребує комплексного підходу до технічного, етичного та соціального проєктування.

Також варто зазначити, що цифрові платформи суттєво змінюють механізми громадської взаємодії, створюючи нову культуру комунікації, кооперації та взаємопідтримки. Одним з головних ефектів їх впровадження є підвищення ефективності комунікації між учасниками ініціатив — волонтерами,

організаціями, благодійниками, державними структурами та громадянами. Завдяки онлайн-інструментам обмін інформацією відбувається у реальному часі, що дозволяє швидко реагувати на запити, координувати дії та звітувати про результати [5].

Ще одним важливим аспектом є формування цифрової культури взаємодопомоги. Платформи стимулюють відповідальне громадянство, коли люди не лише споживають інформацію, а й активно долучаються до спільних дій. Це сприяє розвитку соціального капіталу, формуванню горизонтальних зв'язків і поширенню практик самоорганізації. Цифрові інструменти дозволяють прозоро відстежувати внески, результативність кампаній та участь окремих користувачів, що підвищує довіру до ініціатив.

Окрему роль відіграє залучення молоді та нових користувачів, які з дитинства звикли до цифрового середовища. Саме вони найактивніше використовують мобільні додатки, соціальні мережі, месенджери та інші онлайн-інструменти як природне середовище для волонтерства, активізму чи участі в кампаніях. Завдяки зручному інтерфейсу, адаптивному дизайну та інтеграції з популярними сервісами, платформи стають привабливими навіть для тих, хто вперше долучається до громадської діяльності [6]. Загалом, цифрові рішення не лише спрощують технічну реалізацію ініціатив, а й змінюють саму логіку громадської участі, роблячи її більш відкритою, динамічною та масовою.

Отже, можна зробити висновок, що вебтехнології стали невід'ємною складовою сучасного громадянського суспільства, забезпечуючи ефективну комунікацію, організацію та координацію дій учасників громадських ініціатив. Завдяки своїй доступності, інтерактивності та можливості масштабування, цифрові платформи відкривають нові горизонти для розвитку волонтерства, благодійності, активізму та самоорганізації населення. Вони сприяють формуванню нової цифрової культури взаємодопомоги, заснованої на прозорості, швидкості реагування та залученні широких верств населення, зокрема молоді. Разом з тим, широке впровадження вебрішень супроводжується викликами, що потребують комплексного підходу. Реагуючи на ці виклики, суспільство дедалі частіше

звертається до відкритих рішень, мультиканальних інтерфейсів та підвищення цифрової грамотності громадян.

У перспективі подальший розвиток громадських ініціатив спиратиметься на використання новітніх технологій: мобільних застосунків, штучного інтелекту, аналітики великих даних, блокчейну для прозорості транзакцій. Особливо важливо забезпечити державну підтримку цифрових громадських рішень, стандарти безпеки та доступність для всіх категорій населення. В умовах зростаючих соціальних викликів цифрові платформи мають стати не лише інструментом допомоги, а й основою сталого розвитку громадянського суспільства.

Список використаних джерел

1. Зеленін В., Чопик Л., Доскач С. Вплив медіа на масове сприйняття та психологічний стан громадян. *Humanitarian studios: pedagogics, psychology, philosophy*. 2024. Т. 15, №2. С. 213-219.

2. 93% українців володіють цифровими навичками: Мінцифра презентувала результати дослідження цифрової грамотності українців. *Рівненська обласна державна адміністрація*. URL: https://www.rv.gov.ua/news/93-ukraintsiv-volodiiut-tsyfrovymy-navychkamy-mintsyfra-prezentuvala-rezultaty-doslidzhennia-tsyfrovoi-hramotnosti-ukraintsiv (дата звернення: 05.05.2025).

3. Третьякова К. В. Особливості роботи закладів культури громади в умовах військової агресії (на прикладі Тульчинської міської територіальної громади), 2024. 94 с.

4. Василецький М. Цифрові платформи та децентралізація як каталізатори трансформації регіональних медіа в Україні: можливості, виклики та перспективи розвитку. *Редакційна колегія*. 2024. С. 272-275.

5. Панькова О. В., Касперович О. Ю. Українське волонтерство в умовах збройної російської агресії: зміцнення потужностей через залучення ресурсів цифровізації, платформізації, ІКТ та мережевих технологій. *Економічний вісник Донбасу*. 2022. С. 113-123.

6. Журба К. Цифровізація військово-патріотичного виховання української молоді в умовах війни. *Інноватика у вихованні*. 2023. №18. С. 66-75.