Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра програмної інженерії
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

магістр
(назва освітнього ступеня)

на тему: «Проєктування і розробка web-платформи для волонтерської діяльності з використанням сучасних IT технологій та Agile-методології»

Виконала: студентка VI курсу, групи СП-61
спеціальності
121 – Інженерія програмного забезпечення
(шифр і назва спеціальності)

(підпис) Лагола Р.О. (прізвище та ініціали)

Керівник (підпис) Тимків П.О. (прізвище та ініціали)

Нормоконтроль (підпис) Стоянов Ю.М. (прізвище та ініціали)

Завідувач кафедри (підпис) Петрик М.Р. (прізвище та ініціали)

Рецензент (підпис) Тиш Є. В. (прізвище та ініціали)

Тернопіль 2025

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра програмної інженерії
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Петрик М.Р.
(підпис) (прізвище та ініціали)
«_____» _____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня магістр
(назва освітнього ступеня)

за спеціальністю 121 «Інженерія програмного забезпечення»
(шифр і назва спеціальності)

студенту Лаголі Ростиславу Орестовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Проєктування і розробка web-платформи для волонтерської діяльності з використанням сучасних IT технологій та Agile -методології

Керівник роботи Тимків Павло Олександрович, к.т.н., ст.викл.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «27 » листопада 2025 року № 4/7-1045

2. Термін подання студентом завершеної роботи 22 грудня 2025 р.

3. Вихідні дані до роботи Завдання на розробку web-платформи

4. Зміст роботи (перелік питань, які потрібно розробити)
Сформулювати, погодити та затвердити тему кваліфікаційної роботи; розробити та затвердити завдання; проаналізувати завдання, зробити огляд та проаналізувати бібліографічні матеріали щодо стану справ та тенденій в обраному напрямку дослідження; сформулювати завдання, обґрунтувати цілі дослідження; розробити та спроєктувати систему; описати технології, етапи розробки та саму розроблену програмну систему (у разі потреби) розробити план тестування програмного продукту; оформити допоміжну документацію; проаналізувати роботу щодо питань з дотримання положень про охорону праці та безпеку в надзвичайних ситуаціях; оформити пояснювальну записку; зробити відповідні висновки за результатами виконаної роботи.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
Ілюстративна частина кваліфікаційної роботи оформляється у вигляді слайдів, висвітлює основні розділи та етапи роботи та містить: тему роботи, мету, предмет та об'єкт дослідження, сформульовані завдання, перелічено етапи виконання кваліфікаційної роботи, короткий аналіз актуальності теми, варіант архітектури програмної системи, представлено опис програмного забезпечення; опис програмного забезпечення та тестування.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці	Осухівська Г.М.		
Безпека в надзв. ситуаціях			
Нормоконтроль	Стоянов Ю.М.		

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Вибір та погодження теми з керівником. Створення та затвердження технічного завдання	16.06.2025-22.06.2025	виконано
2.	Аналіз технічного завдання, підбір та аналіз	23.06-	
	бібліографічних матеріалів необхідних для виконання	01.07.2025	виконано
	кваліфікаційної роботи		
3.	Проєктування та розробка програмного забезпечення	22.09.2025	виконано
4.	Тестування та дослідна експлуатація програмного		
	забезпечення	17.11.2025	виконано
5.	Створення допоміжної документації. Написання та		
	перевірка на відповідність вимогам пояснювальної		
	записки	01.12.2025	виконано
6.	Апробація результатів роботи	17-18.12.2025	виконано
7.	Перевірка програмою антиплагіат	03.12.2025	виконано
8.	Охорона праці	06.12.2025	виконано
9.	Безпека в надзвичайних ситуаціях	06.12.2025	виконано
10.	Формування записки кваліфікаційної роботи	12.12.2025	виконано
11.	Нормоконтроль	14.12.2025	виконано
12.	Попередній захист	15.12.2025	виконано
13.	Рецензування кваліфікаційної роботи	22.12.2025	виконано
14.	Захист кваліфікаційної роботи	23.12.2025	

Студент _____ Лагола Р.О.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Тимків П.О.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Лагола Ростислав Орестович. Проєктування і розробка web-платформи для волонтерської діяльності з використанням сучасних IT технологій та Agile-методології. Кваліфікаційна робота освітнього ступеня «магістр». Тернопільський національний технічний університет імені Івана Пулюя, кафедра програмної інженерії, група СПм-61 спеціальність 121 «Інженерія програмного забезпечення». Тернопіль, 2025. Сторінок 83, таблиць 21, рисунків 6, додатків 2, бібліографічних посилань 63.

Метою роботи є створення web-платформи для волонтерської діяльності з врахуванням її особливостей ведення та специфіки.

Об’єктом дослідження є процес створення web-платформи орієнтованої на волонтерську діяльність з використанням оптимальних обґрунтованих засобів та ресурсів розробки при забезпечення якості.

Предметом дослідження є методи, моделі та технології проєктування та розробки web-платформи із застосуванням сучасних Frontend і Backend технологій, баз даних та хмарних сервісів. Методи дослідження включають: аналіз конкурентних платформ та існуючих рішень, моделювання архітектури системи, проєктування, розробку та тестування функціональних компонент.

В даній роботі продемонстровано процес проєктування, розробки, та тестування веб-платформи для волонтерських ініціатив, яка реалізована як кластер із двох основних компонентів: веб-сервісу для взаємодії користувачів та API для керування подіями та заявками. Проведено аналіз існуючих сервісів (VolunteerMatch, Добро.ua, LetsDoItUkraine), виділено їх переваги та недоліки для формування плану розробки..

Ключові слова: інженерія програмного забезпечення, волонтерська платформа, веб-технології, управління користувачами, події та заявки, хмарні сервіси, автоматизація процесів, Agile-методологія.

ABSTRACT

Lahola Rostyslav Orestovych. Design and development of a web platform for volunteer activities using modern IT technologies and Agile methodology. Qualification work of the educational level "Master". Ternopil Ivan Puluj National Technical University, Department of Software Engineering, Group SPm-61, Specialty 121 "Software Engineering". Ternopil, 2025. Pages 83, tables 21, figures 6, annexes 2, bibliographic references 63.

The purpose of the paper is to create a web platform for volunteer activities, taking into account its features of management and specificity.

The object of the study is the process of creating a web platform focused on volunteer activities using optimal substantiated development tools and resources while ensuring quality.

The subject of the study is the methods, models and technologies of designing and developing a web platform using modern front-end and back-end technologies, databases and cloud services. The research methods include: analysis of competitive platforms and existing solutions, modeling of the system architecture, design, development and testing of functional components.

This paper demonstrates the process of designing, developing, designing and testing a web platform for volunteer initiatives, which is implemented as a cluster of two main components: a web service for user interaction and an API for managing events and requests. An analysis of existing services (VolunteerMatch, Dobro.ua, LetsDoItUkraine) was conducted, their advantages and disadvantages were highlighted for the formation of a development plan.

Keywords: software engineering, volunteer platform, web technologies, user management, events and applications, cloud services, process automation, Agile methodology.

ЗМІСТ

ВСТУП

Перед тим як обрати напрямок, сформулювати тему та приступити до виконання кваліфікаційної роботи для мене важливо було окреслити ряд актуальних тематик і вже визначатись, яка мені буде найбільш близька. Власне проаналізувавши, я зробив висновок, враховуючи реалії сьогодення, що мені потрібно зробити щось корисне для наближення нашої Перемоги. Завдяки набутим знанням та вмінням в ІТ я можу зробити, щось реально необхідне і вирішив розробляти web-платформу для волонтерів. Спочатку мені потрібно було зрозуміти специфіку та суть волонтерської діяльності і чому вона життєво необхідна сьогоднішньому суспільству і також врахувати відповідністю спеціальності Інженерія програмного забезпечення. Такий аналіз допоміг мені побачити, які завдання можуть і вирішують волонтери, з якими проблемами стикаються учасники та організації, і що саме має підтримувати платформа. Розуміння цих аспектів дозволить мені створити такий продукт і з таким функціоналом, які справді будуть корисні, а не просто красиві на вигляд.

Метою роботи є створення web-платформи, яка б враховувала специфіку волонтерської діяльності з врахуванням частої зміни вимог за використання сучасних, зокрема, веб-технологій.

На основі аналізу я сформулював основні завдання web-платформи для волонтерів до яких відніс:

1 - Підвищення ефективності організації волонтерських подій, що дозволяє зменшити навантаження на організаторів та забезпечує кращу координацію поміж учасниками.

2 - Забезпечення прозорого процесу реєстрації та участі.

3 - Надання адміністратору інструментів контролю та управління користувачами, що включає моніторинг активностей та керування ролями.

4 - Безпека даних.

5 - Можливості для подальшого розширення функціоналу.

Об’єктом дослідження є процес розробки web-платформи орієнтованої на координацію волонтерської діяльності з використанням оптимальних обґрунтованих засобів та ресурсів розробки при забезпечення якості з врахуванням можливості зміни вимог.

Предметом дослідження є методи, моделі та технології проєктування та розробки web-платформи для волонтерських ініціатив із застосуванням сучасних фронтенд- і бекенд-технологій, баз даних та хмарних сервісів.

Робота є практико орієнтованою. Оскільки волонтерська діяльність – це той вид, де передбачити щось вкрай важко, то розширення функціоналу, зміна вимог – це те, що дозволить врахувати реалії військового стану та допомогти врятувати не одне життя і наближатиме Перемогу. І сьогодні – це точно один із найбільш актуальних напрямків.

Методи дослідження включають: аналіз конкурентних платформ та існуючих рішень, моделювання архітектури системи, проєктування, розробку та тестування функціональних компонентів.

У даній роботі продемонстровано етап аналізу вимог до розроблюваного продукту, процеси проєктування, розробки, тестування, впровадження та підтримки. Платформа реалізовуватиметься як кластер із двох основних компонентів: веб-сервісу для взаємодії користувачів та API для керування подіями та заявками. Проведено аналіз існуючих сервісів (VolunteerMatch, Добро.ua, LetsDoItUkraine), виділено їх переваги та недоліки для формування плану розробки..

Обов’язковим елементом кваліфікаційної роботи магістра є її апробація. Результати своєї роботи у вигляді тез я подав до публікації в матеріалах роботи XIII науково-технічної конференції «Інформаційні моделі, системи та технології», яка проводитиметься в період 17-18 грудня 2025 року.

1 АНАЛІЗ ВИМОГ ДО WEB-ПЛАТФОРМИ

1.1. Аналіз предметної області

Перед тим як розробляти web-платформу для волонтерів мені було важливо зрозуміти, що таке волонтерська діяльність і чому вона така важлива і вкрай потрібна суспільству. Такий аналіз допоміг мені побачити, які завдання можуть і вирішують волонтери, з якими проблемами найчастіше вони (учасники та організації) стикаються, і що саме має підтримувати платформа. Розуміння цих аспектів дозволить мені створити такий продукт і з таким функціоналом, які справді будуть корисні користувачам, а не просто красиві на вигляд [1-3].

1.1.1. Суть та значення волонтерської діяльності

Волонтерством, зазвичай, називають добровільну, безплатну діяльність, яка спрямована на вирішення соціальних та/чи інших важливих завдань, підтримку як людей так і громадських ініціатив. В загальному такий вид діяльності охоплює широкий спектр активностей, зокрема:

- соціальна допомога (підтримка малозабезпечених, дітей, людей з обмеженими можливостями);
- освітні та культурні проєкти (організація та проведення широкотематичних тренінгів, майстер-класів, різних культурних подій);
- екологічні ініціативи (охорона навколишнього середовища та з цим пов’язані напрямки);
- гуманітарна допомога (під час надзвичайних ситуацій, воєнних конфліктів).

Реалії сьогодення такі, що роль волонтерства, як ніколи, у сучасному суспільстві є особливо актуальною, оскільки така діяльність сприяє і формуванню громадянської свідомості та соціальної активності; і розвитку не формальної освіти та професійних навичок учасників; зміцненню соціальної взаємодії між організаціями та людьми.

1.1.2. Проблематика та виклики при організації волонтерської діяльності

Не дивлячись на свідомість та значну кількість волонтерів та активних ініціатив, при їх ефективній координації виникають певні, специфічні труднощі, деякі з них я виокремив і наведу далі.

1. Розрізненість інформації (оголошення про волонтерські події досить часто публікують у різних джерелах, таких як: соціальні мережі, на веб-сайти організацій, в групах месенджерів, це все одно ускладнює пошук актуальних заходів та часто зовсім не сприяє чи навіть і унеможливлює ефективне планування.

2. Неефективна комунікація (як організатори так і самі волонтери часто не мають єдиного централізованого інструмента для швидкої взаємодії, а відсутність зручного механізма повідомлень чи сповіщень може призвести до затримок, плутанини або пропуска важливих подій чи активностей).

3. Відсутність прозорої статистики та звітності (без централізованої платформи складно відстежувати, для прикладу, кількість залучених волонтерів; виконані завдання; ефективність конкретних ініціатив. Це ускладнює оцінку впливу та планування ресурсів.

4. Обмежена інтеграція з сучасними технологіями (існуючими рішеннями зазвичай не підтримується інтеграція з картографічними сервісами, месенджерами та соціальними мережами, що обмежує можливості швидкого пошуку подій, сповіщень та взаємодії учасників).

5. Обмежена гнучкість при управлінні подіями (організатори/волонтери не завжди мають змогу швидко оновлювати інформацію про події, змінювати кількість учасників або терміни проведення, і це, в свою чергу може створювати певні незручності для зацікавлених).

1.1.3. Огляд існуючих рішень

Оскільки проблема вкрай є актуальна і вже не нова [1-3], сьогодні існують спеціалізовані платформи, які частково можуть вирішувати викладені в п.1.1.2 проблеми. Я знайшов набільш затребувані, до яких варта віднести:

VolunteerMatch – міжнародний ресурс для пошуку волонтерських можливостей, через який оприлюднюються інформація про події, але який, на мою думку, досить обмежений у персоналізації та інтеграції з локальними сервісами [4].

Добро.ua – діюча українська платформа для реєстрації волонтерів та їхніх ініціатив, за допомогою якої забезпечується базовий функціонал, проте нею не завжди підтримується аналітика та зручна комунікація поміж учасниками [5].

LetsDoItUkraine – ресурс з орієнтацією на організацію акцій екологічного напрямку, який досить гарно зарекомендував себе під конкретний тип подій, але я вважаю, що він не зовсім є підходящим для загальної координації усіх видів волонтерської діяльності [6].

Таким чином, можу констатувати, що жодна з існуючих платформ не в змозі забезпечити комплексний підход, а саме: інтеграцію з сучасними веб-технологіями, централізовану комунікацію, аналітику та гнучке керування подіями.

1.1.4. Технологічні аспекти та тенденції

Таким чином повстає потреба у централізованій веб-платформі для координації волонтерської діяльності [7]. Зокрема, платформа повинна поєднувати: для організаторів можливість створювати та управляти подіями; пошук і подачу заявок волонтерами; систему сповіщень та комунікації; аналітичні інструменти для моніторинга ефективності.

Використання сучасних web-технологій забезпечує масштабованість, зручність та інтеґрацію з іншими сервісами [8-10].

Структура системи повинна бути гнучкою, щоб у майбутньому можна було додати додаткові функції або нових акторів (наприклад, гість, зовнішні сервіси), не змінюючи базову логіку.

На підставі цього всього проведеного в розділі 1.1. аналізу випливає постановка завдання на розробку веб-платформи, яка поєднуватиме зручність для користувачів, ефективну організацію волонтерських ініціатив та можливості подальшого розширення функціоналу.

Сучасні ж веб-технології надають можливості для створення гнучких та масштабованих платформ, зокрема:

- Frontend – динамічні, адаптивні інтерфейси на основі React.js, Vue.js або Angular[11-16];
- Backend – це масштабовані серверні рішення на Node.js, Python (Django, FastAPI) або PHP (Laravel) [17];
- бази даних: SQL (PostgreSQL, MySQL) або NoSQL (MongoDB) для збереження інформації про користувачів та події[18];
- інтеграції: Google Maps API, соціальні мережі, месенджери для швидкого пошуку, сповіщень та залучення користувачів;
- Agile-методологія вважається ітеративним підхідом до розробки, що дозволяє враховувати зворотний зв’язок користувачів та швидко вдосконалювати платформу [19-22].

1.2. Постановка завдання і формулювання цілей проєкту

Після проведеного аналізу предметної області[23] я зробив висновки про те, що існує реальна потреба в створенні централізованої web-платформи, яка б забезпечувала координацію волонтерських ініціатив та робила процес взаємодії між організаторами і волонтерами простим і ефективним.

На підставі цього аналізу я визначив, що платформа повинна надавати наступні можливості:

Для організаторів: створення та керування подіями, налаштування параметрів участі, контроль за учасниками.

Для волонтерів: пошук актуальних заходів, подача заявок на участь, отримання сповіщень про зміни та оновлення.

Система комунікацій та сповіщень: забезпечення зручного обміну повідомленнями між учасниками та організаторами.

Аналітичні інструменти: моніторинг ефективності заходів, статистика участі та результатів волонтерської діяльності.

Використання сучасних веб-технологій дозволяє забезпечити гнучкість і масштабованість платформи, враховуючи специфіку волонтерської діяльності та потреби користувачів. Це ґарантує, що платформа буде зручною, інтегрованою з іншими сервісами і зможе легко розширюватися в майбутньому.

Структура проєкту має бути достатньо гнучкою, щоб у майбутньому можна було додавати нові функції або інтегрувати нових акторів (наприклад, гостей або сторонні сервіси) без порушення базової логіки платформи.

На основі цього аналізу я сформулював наступні основні завдання розроблюваної web-платформи для волонтерських ініціатив, це:

1 - Підвищення ефективності організації волонтерських подій, що дозволяє зменшити навантаження на організаторів та забезпечує кращу координацію учасників.

2 - Забезпечення прозорого процесу реєстрації та участі, щоб волонтери могли легко знаходити та подавати заявки на заходи.

3 - Надання адміністратору інструментів контролю та керування користувачами, що включає моніторинг активності та управління ролями.

Таким чином, запропонований проєкт поєднуватиме зручність для користувачів, ефективну організацію волонтерських ініціатив, автоматизацію управління подіями та заявками, підвищує ефективність взаємодії між учасниками і гарантує безпеку даних, одночасно відкриваючи можливості для подальшого розширення функціоналу.

1.2.1. Вимоги до системи

Далі я сформулював вимоги до системи у вигляді функціональних, не функціональних та бізнес-вимог.

До функціональних вимог я відніс:

- можливість реєстрації користувачів (волонтерів та організаторів),
- можливість входу у систему з автентифікацією 9захист данних),
- можливість для організаторів створювати і керувати подіями,
- можливість подавати заявки на участь у подіях,
- можливість затверджувати та відхиляти заявки,
- можливість надсилати користувачам повідомлення про зміни.

До не функціональних вимог варта віднести:

- продуктивність: обробка до 1000 користувачів одночасно,
- безпека: шифрування паролів, контроль доступу,
- юзабіліті: зручний інтерфейс для мобільних та десктопних пристроїв,
- надійність: резервне копіювання даних.

Бізнес-вимоги:

- підвищення можливостей участі у подіях волонтерів,
- забезпечення простого, логічного та оптимального процесу адміністрування подій.

Почавши роботу над темою моєї кваліфікаційної роботи з зрозумів, що потрібно вчасно зупинитись і закінчити почате, хоча хотілось зробити якомога більше. Саме це стало пріорітетним у виборі моделі проєктування. На даному етапі роботи я визначив, враховуючи потребу замовників, пріорітетність вимог:

1 - Обов’язкові (англ. must-have): реєстрація користувачів, створення та керування подіями, подання та затвердження заявок.

2- Бажані (англ. should-have): повідомлення користувачів, фільтрація подій за категоріями.

3- Додаткові (англ. nice-to-have): інтеграція з соціальними мережами, рейтинг волонтерів.

Враховуючи вимоги до виконання кваліфікаційної роботи магістра [23], я вважаю, що переліченого достатньо для виконання роботи, а враховуючи що в готовий проєкт можна буде вносити зміни (функціонал можна буде змінювати та значно розширювати), враховуючи потреби та напрям діяльності.

Також логічним є встановлення хоча б мінімальних обмежень до системи. Зокрема: технічні (вебплатформа, підтримка сучасних браузерів, адаптивність під мобільні пристрої) та організаційні (обмежений бюджет, інтеґрація з існуючими базами даних).

1.2.2. Постановка завдання на розробку

Виходячи з проведеного аналізу предметної області, було сформульовано остаточне завдання розробки веб-платформи для підтримки волонтерських ініціатив. Платформа повинна забезпечувати наступні основні функції:

1 - Реєстрація та аутентифікація користувачів різних ролей: волонтер, організатор та адміністратор. Це дозволить кожному користувачу отримувати доступ до відповідного функціоналу відповідно до його прав.

2 - Створення, перегляд та керування волонтерськими подіями, що дає можливість організаторам планувати заходи та відслідковувати їх стан.

3 - Подання, обробка та затвердження заявок на участь у заходах, забезпечуючи прозорий та зручний процес для волонтерів та організаторів.

4 - Контроль доступу до адміністративних функцій та керування користувачами, що гарантує безпеку та правильне розподілення прав між різними ролями.

Окрім цього, продукт повинен бути інтуїтивно зрозумілим та зручним для користувачів, забезпечувати безпечну роботу з даними і допускати можливість масштабування, щоб ефективно підтримувати велику кількість волонтерів та заходів.

1.2.3. Цілі розробки

Остаточні цілі розробки я сформулював в наступному об’ємі та такій послідовності:

1. Провести аналіз вимог до системи та визначити ключових акторів та сценарії використання.
2. Розробити UML-діаграми для опису функціональної структури системи (Use Case, Class, Activity).
3. Реалізувати прототип веб-платформи із основними функціями: реєстрація, події, заявки.
4. Виконати тестування системи для перевірки її працездатності та відповідності вимогам.
5. Підготувати пояснювальну записку відповідно до вимог до кваліфікаційної роботи магістра [23], яка включатиме документування основних етапів по розробленій системі для подальшого використання та/або масштабування.
6. Підготувати презентацію для представлення роботи на попередньому захисті та на розгляд ЕК на основному захисті[23].
7. Успішно захистити кваліфікаційну роботу на здобуття другого освітнього рівня «магістр».

1.3. Пошук акторів та варіантів використання

З метою успішного проєктування web-платформи необхідно означити акторів, йдеться прокористувачів або зовнішні системи, які взаємодіютимуть із платформою, а також варіанти використання(англ.use case) [24-27], тобто функціональні сценарії, які виконуються системою повинна з метою задоволення потреб означених акторів.

До перших акторів проєкту я відніс:

1 – Волонтера.

2 – Організатора.

3 - Адміністратор.

До основних сценаріїв, на рівні аналіза вимог, я відніс:

1 - Реєстрацію користувачів.

2 - Вхід у систему.

3 - Створення подій

4 - Подання та управління заявками

1.3.1. Актори системи

Для платформи волонтерських ініціатив попередньо я визначив наступних ключових акторів та описав їх в таблиці 1.1 далі.

Таблиця 1.1 – Ключові актори та опис

Актор	Опис
«Волонтер»	Зареєстрований користувач, який може подавати заявки на участь у подіях, переглядати доступні події та отримувати інформацію про організовані ініціативи.
«Організатор»	Користувач, який створює та керує подіями, переглядає подані заявки, затверджує або відхиляє заявки волонтерів.
«Адміністратор»	Користувач із правами управління системою: керування користувачами, контроль доступу, адміністрування подій та бази даних.

Важливо, що передбачено можливість додавання акторів, зокрема зовнішні сервіси (електронна пошта, SMS), які можна включити при розширенні системи.

1.3.2. Визначення та опис варіантів використання

Далі опишу основні варіанти використання, які наведу далі в таблиці.

Таблиця 1.2 – Основні варіанти використання платформи

Use Case	**Актор**
Реєстрація	«Волонтер», «Організатор»
Вхід в систему / Аутентифікація	«Волонтер», «Організатор», «Адміністратор»
Перегляд подій	«Волонтер»
Подання заявки на участь у події	«Волонтер»
Створення події	«Організатор»
Перегляд заявок на події	«Організатор»
Затвердження/Відхилення заявок	«Організатор»
Керування користувачами	«Адміністратор»
Керування подіями	«Адміністратор»
Контроль доступу	«Адміністратор»

Для наочності я представив UML-діаграму варіантів використання[28-30], яка показує взаємодію ключових акторів із системою, як на рисунку 1.1

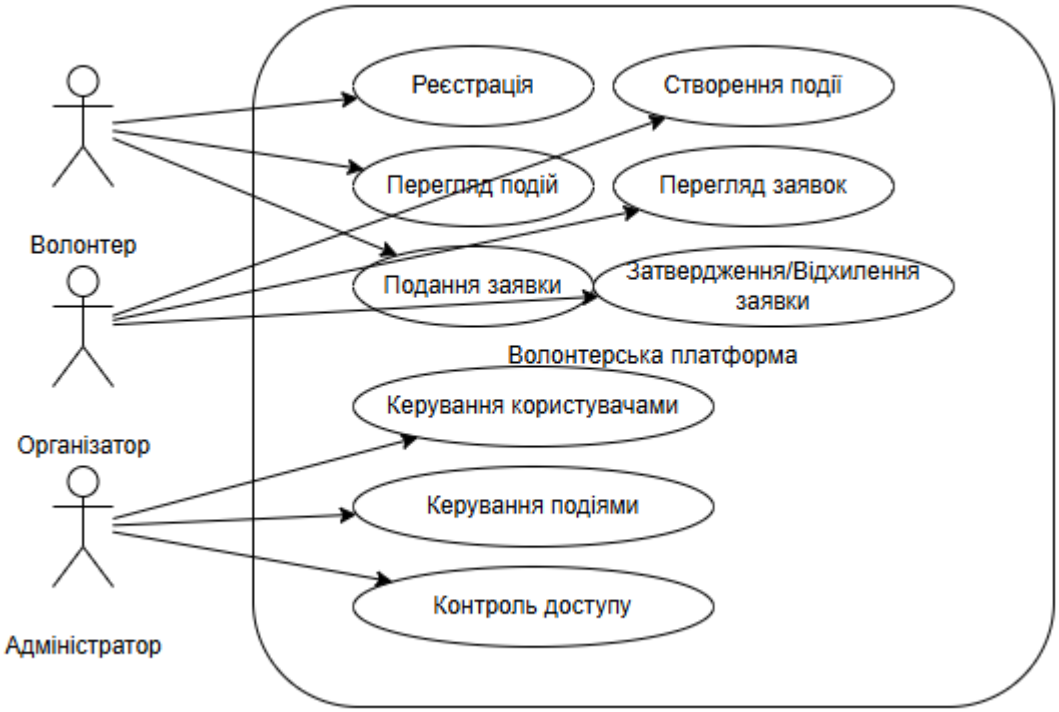


Рис. 1.1 – UML-діаграма варіантів використання

На рисунку 1.1 представлено функції головних акторів, зокрема:

«Волонтер» – реєстрація, вхід, перегляд подій, подання заявки;

«Організатор» – вхід, створення події, перегляд заявок, затвердження/відхилення заявок

«Адміністратор» – Вхід, Керування користувачами, Керування подіями, Контроль доступу

Отже, для моделювання веб-платформи для волонтерів були визначені основні типи користувачів. Цей вибір забезпечує повне охоплення ключових функціональних взаємодій у системі, дозволяючи продемонструвати логіку реєстрації, управління подіями, подання та опрацювання заявок, а також контроль за безпекою та доступом.

Варто відзначити, що обрана модель з трьома основними акторами є мінімально достатньою для реалізації та демонстрації основного функціоналу платформи. При потребі систему можна вдосконалити, додавши додаткових акторів, таких як «Гість» (незареєстрований користувач, який має обмежений доступ до перегляду інформації) або «Зовнішні сервіси» для інтеграції з API карт, месенджерів або соціальних мереж. Таке розширення дозволить мені підвищити гнучкість платформи та адаптувати її до більш широкого спектру сценаріїв використання, зберігаючи при цьому зрозумілу та логічну структуру взаємодії.

1.4. Опис ключових варіантів використання

Для забезпечення чіткого розуміння функціоналу web-платформи вважаю за необхідність детально описати ключові варіанти використання (Use Case Description)[25,26], які представляють основні сценарії взаємодії акторів із системою.

Кожен варіант включатиме:

1- назву;

2- актора;

3- опис;

4- передумови;

5- результати;

6- основний сценарій;

7- альтернативні сценарії.

Таблиця 1.3 – Use Case 1: Реєстрація користувача

Параметр	**Опис**
Назва	Реєстрація користувача
Актор	«Волонтер», «Організатор»
Передумови	Користувач не зареєстрований в системі
Опис	Користувач: заповнює реєстраційну форму, підтверджує електронну адресу, створює обліковий запис
Основний сценарій	1. Користувач відкриває сторінку реєстрації. 2. Вводить персональні дані (ім’я, e-mail, пароль). 3. Підтверджує реєстрацію через e-mail. 4. Отримує доступ у систему.
Альтернативні сценарії	Некоректні дані, а саме повідомлення про помилку. E-mail вже зареєстрований. Пропозиція відновити пароль.

Таблиця 1.4 – Use Case 2: Створення події

Параметр	**Опис**
Назва	Створення події
Актор	«Організатор»
Передумови	Користувач зареєстрований та увійшов у систему як Організатор
Опис	Організатор створює нову волонтерську подію з описом, датою та місцем проведення
Основний сценарій	1. Організатор відкриває сторінку «Створити подію». 2. Вводить інформацію про подію (назва, опис, дата, місце). 3. Публікує подію в системі.
Альтернативні сценарії	Не коректні данні. Повідомлення про помилку.

Таблиця 1.5 – Use Case 3: Подання заявки на участь у події

Параметр	**Опис**
Назва	Подання заявки на участь
Актор	«Волонтер»
Передумови	Користувач зареєстрований та увійшов у систему як Волонтер
Опис	Волонтер обирає подію та подає заявку на участь
Основний сценарій	1. Волонтер переглядає список подій. 2. Вибирає подію, яка цікавить. 3. Натискає «Подати заявку». 4. Система підтверджує отримання заявки.
Альтернативні сценарії	Волонтер вже подав заявку. Повідомлення про дубль.

Таблиця 1.6 – Use Case 4: Затвердження/Відхилення заявки

Параметр	**Опис**
Назва	Затвердження/Відхилення заявки
Актор	Організатор
Передумови	Організатор увійшов у систему та має створені події
Опис	Організатор переглядає подані заявки і вирішує, чи приймати учасника
Основний сценарій	1. Організатор відкриває список заявок на подію. 2. Переглядає інформацію про волонтера. 3. Вибирає «Затвердити» або «Відхилити». 4. Система повідомляє волонтера про рішення.
Альтернативні сценарії	Організатор відміняє рішення. Повернення до списку заявок.

2 ПРОЄКТУВАННЯ ТА РОЗРОБКА WEB-ПЛАТФОРМИ

2.1 Вибір процесу розробки

При розробці web-платформи для волонтерських ініціатив важливо обрати процес, який забезпечує: гнучкість при їх можливій зміні, ефективну комунікацію між розробниками та користувачами, якісне тестування на кожному етапі, контроль термінів та наяних ресурсів.

Враховуючи специфіку волонтерської діяльності та важливість враховувати вимоги сьогодення, тобто вимоги до системи точно будуть змінюватись з часом, оскільки у військовий час прогнозувати та передбачити щось важко, я розглянув наступні, оптимально можливі для моєї роботи, підходи до розробки:

1. Водоспадна модель (англ.Waterfall) , яка передбачає послідовність етапи: від аналізу вимог, проєктування, реалізації, тестування, і аж до впровадження. Головними її перевагами, вважаю чітку структуру та зрозумілі етапи. До недоліків віднесу складність до внесення змін особливо на пізніх етапах та досить повільний цикл отримання результату [31-33].

2. Ітеративно-інкрементальний підхід – це коли система розробляється по частинах (тобто інкрементах), кожен з яких проходить повний цикл розробки. До переваг віднесу можливість раннього тестування та можливість отримання робочого продукту; також досить зручну адаптація до змін[34,35].

3. Agile (Scrum, Kanban), який орієнтований на швидкі ітерації, гнучке управління вимогами, з можливістю активної участі користувачів. Короткі ітерації (так звані спринти), регулярні зустрічі команди (англ.Daily Scrum), демонстрації результатів (англ.Review). До переваг, особливо враховуючи специфіку обраного мною волонтерського напрямку, я віднесу максимальну гнучкість, швидкий зворотний зв’язок, високу якість продукту[20-22].

Вже навіть на попередньому етапі розробки, який я описав детально в попередньому розділі, враховуючи особливості волонтерської діяльності, мені

було ясно, що для моєї веб-платформи волонтерських ініціатив оптимально буде зупинитись на виборі Agile/Scrum моделі, оскільки і найголовніще, що :

1- Вимоги користувачів можуть змінюватися під час розробки (наприклад, додавання нових акторів, в моєму випадку «Гість», «Зовнішні сервіси» та інші на вимогу замовника, нових функцій для подій тощо).

2- Команда розробників має можливість демонструвати проміжні результати, це дуже важливо, оскільки вплинути можна раніше, а для волонтерської дільяності під час особливо воєнного стану це ще й життєво необхідно може бути, ну і отримувати швидкий фідбек.

3- Покроковий (ітерактивний) підхід до розробки дозволить зменшити ризик помилок і покращить якість коду, що в свою чергу дасть змогу швидше передати готовий проєкт в роботу волонтерам та причетним.

В свою чергу до етапів процесу розробки за методологією Scrum я віднесу[20]:

1. Планування спринту (Sprint Planning) сюди: формування завдань на поточну ітерацію.
2. Розробка: безпосереднє написання коду функціональних модулів.
3. Тестування: перевірка на відповідність вимогам та стабільність роботи.
4. Демонстрація (Sprint Review): презентація готового функціоналу користувачам.
5. Ретроспектива: аналіз виконаної роботи, пошук шляхів оптимізації процесу.

Таким чином, підсумовуючи наведений аналіз, я обґрунтував та обрав модель Agile/Scrum, яка дасть змогу забезпечити оптимальну комбінацію таких характеристик як гнучкість, контролю за якістю та ефективністю розробки. Саме такий підхід дозволить мені вдосконалити розробку фактично в будь, який момент та швидко зреагувати на зміни, мінімізувати ризики і створювати продукт, максимально відповідний до очікувань користувачів.

Отже, для розробки web-платформи волонтерських ініціатив було обрано ітеративно-інкрементальний процес розробки (Agile-підхід), зокрема Scrum, через:

1. Гнучкість – можливість змінювати вимоги під час розробки у відповідь на потреби користувачів.

2. Ітеративність розробки – продукт розвивається по етапно, і це дасть змогу отримувати робочі версії на ранніх стадіях.

3. Зворотний зв’язок – активна участь користувачів та організаторів у тестуванні та уточненні вимог.

4. Прозорість процесу – всі учасники команди та зацікавлені сторони мають доступ до прогресу проєкту.

До ключових етапів процесу розробки я сформулюю:

1. Планування спринтів: визначення функціоналу для кожної ітерації.
2. Розробка: програмування функцій та модулів системи.
3. Тестування: перевірка працездатності функцій.
4. Ретроспектива: оцінка результатів спринту та коригування плану.

2.2 Проєктування архітектури web-платформи

Архітектура програмної системи визначає її структуру, взаємодію компонентів та організацію даних [36-39]. Вона є основою для подальшої розробки, тестування та підтримки системи. Для web-платформи, яка полегшить волонтерську діяльність обрана клієнт-серверна трирівнева архітектура, яка забезпечує масштабованість, безпеку та зручність розвитку продукту.

2.2.1 Обґрунтування вибору архітектурного стилю розробки веб-платформи

Для розробки я обрав три-рівневу архітектура, суть якої розділення системи на три логічні рівні [40-42]:

1. Рівень представлення (Presentation Layer). Відповідає за взаємодію користувача із системою та забезпечує зручний та інтуїтивно зрозумілий інтерфейс для основних акторів: волонтерів, організаторів і адміністраторів. Основні

технології: HTML, CSS, JavaScript, фреймворки React або Vue. Особливості: адаптивність під мобільні та десктопні пристрої, швидкий відгук на дії користувача.

2. Рівень бізнес-логіки (Application / Business Logic Layer). Виконує обробку запитів від користувачів та реалізацію бізнес-процесів системи, зокрема: створення і керування подіями; подання та опрацювання заявок на участь; відправка сповіщень та повідомлень; використання серверних технологій: Python (Django, Flask), Node.js чи інших. Забезпечує інтеґрацію з базою данних та зовнішніми сервісами (наприклад, соціальні мережі).

3. Рівень збереження даних (Data Layer). Відповідає за збереження, пошук і опрацювання інформації: дані користувачів, подій, заявок, повідомлень. Використовуються реляційні бази даних (PostgreSQL, MySQL) або NoSQL (MongoDB) для специфічних випадків та вимог. Забезпечує безпеку даних, резервне копіювання та масштабованість.

2.2.2 Компоненти системи

Архітектура розроблюваного продукту включає наступні основні модулі, до яких я відніс:

1. Модуль керування користувачами. Сюди:

- реєстрація та аутентифікація волонтерів, організаторів та адміністраторів;
- можливість редагування профілю, зміни пароля, налаштувань сповіщень.

2. Модуль подій. До нього:

- створення, редагування і публікація подій;
- керування доступом до подій (відкриті /закриті);
- категоризація подій для зручного пошуку.

3. Модуль заявок включає:

- подання та управління заявками на участь у подіях;
- автоматичне повідомлення організаторів про нові заявки;
- можливість затвердження або відхилення заявки.

4. Модуль повідомлень та сповіщень включає:

- інформування користувачів про статус заявки або зміни в події;
- відправка електронних листів та push-повідомлень.

5. Модуль адміністрування включає:

- контроль за користувачами та контентом платформи;
- можливість блокування користувачів, модерація подій, звітність.

Рис. 2.3 – UML-діаграма компонент web-платформи волонтерських активностей

На діаграмі представлено основні компоненти системи та взаємозв’язки між ними.

Зокрема клієнтська частина (Front-end) взаємодіє з сервером застосунку через REST API. Сервером реалізуються модулі керування користувачами, подіями, заявками, повідомленнями та адмініструванням в цілому.
Усі дані зберігаються у базі данних, а сповіщення надсилаються електронною поштою та push-сервісами.

Оскільки діаграма компонентів відображає високорівневу структуру системи, наступним етапом буде детальне проєктування логіки через UML-діаграми, в результаті це дасть змогу уточнити внутрішню взаємодію класів та об’єктів.

Для кращого розуміння того, яким чином користувачі взаємодіють з платформою, далі на слідуючому рисунку 2.4 я наведу UML-діаграму варіантів використання (Use Case Diagram), за допомогою якої я проілюструю ролі користувачів та основні дії, які вони можуть виконувати в системі.

Рис. 2.4 – UML-діаграма варіантів використання
web-платформи волонтерських активностей

Отже, визначивши основні компоненти веб-платформи та їхню взаємодію, я можу перейти до аналізу переваг обраної архітектури, адже саме архітектурний підхід забезпечує узгоджену роботу всіх структурних елементів системи та визначає подальші принципи її розробки.

2.2.3 Переваги обраної архітектури

Підсумовуючи все вище сказане вибір архітектури є очевидним, оскільки повністю відповідає завданню та специфіці реалізовного проєкту, зокрема до переваг віднесу:

1 - масштабованість: зручність та простота при додаванні нових функцій та модулів. При цьому не потрібно перебудовувати всю систему.

2 - розділення обов’язків: чітка сегментація між інтерфейсом, логікою та данними.

3 - безпека: централізоване керування доступом як до даних так і до функцій.

4 - продуктивність: можливість оптимізації окремих модулей не залежно від інших.

5 - легкість та зручність підтримки: модульна структура спрощує тестування та внесення змін.

В свою чергу запропонована трьох-рівнева архітектура забезпечить мені:

1 - надійність роботи web-платформи в цілому.

2 - ефективну взаємодію між користувачами і системою.

3 - простоту розширення функціоналу та інтеграції з іншими сервісами.

4 - високий рівень безпеки даних та їх доступності.

Оскільки архітектура системи визначає основні модулі та їхню взаємодію, наступним етапом в мене в роботі буде проєктування бази даних, яка забезпечуватиме зберігання та цілісність інформації у розроблюваній мною web-платформі.

2.3 Побудова схеми бази даних для розроблюваної web-платформи

На цьому етапі роботи я хочу зазначити, що база даних є центральним компонентом веб-платформи для волонтерської діяльності, оскільки нею забезпечується безпосереднє зберігання та обробка усієї інформації про користувачів, події, заявки, а також повідомлення. Для розроблюваного програмного продукту я обрав базу данних реляційного типу, яка дозволить мені без проблем структуровано організувати дані у вигляді таблиць зі зв’язками між ними.

2.3.1 Обґрунтування вибору типу бази даних

Для платформи я використовуватиму базу даних реляційного типу[43-45], наприклад PostgreSQL чи MySQL, через переваги та оптимальність для виконання саме моєї, специфічної, задачі. Зокрема використання реляційної бази данних забезпечить:

1 - підтримку структурованих даних та жорстких зв’язків поміж таблицями.

2 - для ефективного пошуку та обробки даних, можливість використання SQL-запитів.

3 - високе надійність та масштабованість.

4 - забезпечення цілісності даних при допомозі зовнішніх ключів та обмежень.

2.3.2 Основні таблиці бази даних, міжтабличні зв’язки, додаткові аспекти

Схема бази даних включає такі ключові таблиці:

1. Таблиця «Users» («Користувачі»):

user_id – унікальний ідентифікатор користувача;

name – ім’я користувача;

email – електронна пошта;

password – хешований пароль;

role – роль користувача (волонтер, організатор, адміністратор);

created_at – дата створення профілю.

2. Таблиця «Events» («Події»):

event_id – унікальний ідентифікатор події;

title – назва події;

description – опис;

start_date – дата та час початку;

end_date – дата та час завершення;

organizer_id – зовнішній ключ до таблиці Users, організатор події.

3. Таблиця «Applications» («Заявки»):

application_id – унікальний ідентифікатор заявки;

user_id – зовнішній ключ до Users;

event_id – зовнішній ключ до Events;

status – статус заявки (подано, затверджено, відхилено);

submitted_at – дата подання заявки.

4. Таблиця «Messages» («Повідомлення»):

message_id – унікальний ідентифікатор повідомлення;

sender_id – зовнішній ключ до Users, відправник;

receiver_id – зовнішній ключ до Users, отримувач;

content – текст повідомлення;

sent_at – дата та час відправки.

5. Таблиця «EventCategories» («Категорії подій»):

category_id – унікальний ідентифікатор категорії;

name – назва категорії;

description – опис категорії.

6. Таблиця «EventCategoryMapping» («Зв’язок між подіями та категоріями»):

event_id – зовнішній ключ до Events;

category_id – зовнішній ключ до EventCategories.

Забезпечує багато-до-багатьох зв’язок між подіями та категоріями.

Зв’язки між таблицями матимуть наступний вигляд як в таблиці 2.1

Таблиця 2.1 – Міжтабличні зв’язки

Users	↔	«Events»: одним організатором може створюватись багато подій (1:N)
Users	↔	«Applications»: один користувач може подавати багато заявок (1:N)
Events	↔	«Applications»: одна подія може мати багато заявок (1:N)
Events	↔	«EventCategories»: багато-до-багатьох через таблицю EventCategoryMapping
Users	↔	«Messages»: один користувач може відправляти і отримувати багато повідомлень (1:N)

До додаткових аспектів я віднесу:

1. Встановлення обмеження цілісності даних (PRIMARY KEY, FOREIGN KEY).

2. Використання, з метою прискорення пошуку, індексів на полях e-mail, event_id, user_id.

3. Забезпечено резервне копіювання та відновлення даних на рівні бази.

Таким чином, обрана до впровадження реляційна схема бази даних дозволить мені:

- організовано зберігати всю інформацію про користувачів та події;

- забезпечити надійність та цілісність даних;
- легко масштабувати систему при додаванні нових функцій;
- ефективно виконувати пошук, сортування та фільтрацію даних.

Далі наведу логічну модель бази даних, яка матиме вигляд як на рис. 2.5

Рис. 2.5 – ER-діаграма бази даних системи управління подіями

Таким чином, на рис. 2.5 представлено логічну частину схеми, структуру, проте для роботи бази цього все ще не достатньо.

Далі наведу лістинги та описи полів, для пояснення призначення кожного із стовпців для всіх таблиць, тобто повну реалізацію фізичної моделі бази

Лістинг 2.1 – Таблиця «Користувачі» (Users) (демонструє створення таблиці «Користувачі», в якій зберігається інформація про користувачів системи, їх ролі та контактні дані. Поле id_користувача є первинним ключем таблиці)

```
CREATE TABLE Користувачі (
  id_користувача INT PRIMARY KEY AUTO_INCREMENT,
  ім_я VARCHAR(100) NOT NULL,
  email VARCHAR(100) UNIQUE NOT NULL,
  пароль VARCHAR(100) NOT NULL,
  роль VARCHAR(50) NOT NULL,
  дата_створення DATETIME DEFAULT CURRENT_TIMESTAMP
);
```

Таблиця 2.2 – Опис полів таблиці «Користувачі»

№	Назва поля	Тип даних	Ключ	Опис
1	id_користувача	INT	PK	Унікальний ідентифікатор користувача
2	ім_я	VARCHAR(100)	-	Ім’я користувача
3	e-mail	VARCHAR(100)	-	Електронна пошта для авторизації
4	пароль	VARCHAR(100)	-	Пароль користувача (зберігається у хеші)
5	роль	VARCHAR(50)	-	Роль у системі (організатор/учасник)
6	дата_створення	DATETIME	-	Дата створення акаунта

Лістинг 2.2 – Таблиця «Події» (Events) (містить таблицю «Події», яка описує всі події системи, їх назву, опис та дати проведення. Поле id_організатора є зовнішнім ключом на таблицю «Користувачі» і забезпечує зв’язок один-до-багатьох між організатором і подіями)

```
CREATE TABLE Події (
  id_події INT PRIMARY KEY AUTO_INCREMENT,
  назва VARCHAR(200) NOT NULL,
  опис TEXT,
  дата_початку DATE NOT NULL,
  дата_закінчення DATE,
  id_організатора INT,
  FOREIGN KEY (id_організатора) REFERENCES Користувачі(id_користувача)
);
```

Таблиця 2.3 – Опис полів таблиці «Події»

№	Назва поля	Тип даних	Ключ	Опис
1	id_події	INT	PK	Унікальний ідентифікатор події
2	назва	VARCHAR(200)	-	Назва події
3	опис	TEXT	-	Опис події
4	дата_початку	DATE	-	Дата початку події
5	дата_закінчення	DATE	-	Дата завершення події
6	id_організатора	INT	FK	Зовнішній ключ на «Користувачі» («Організатор»)

Лістинг 2.3 – Таблиця «Заявки» (Applications) (описує таблицю «Заявки», що зберігає інформацію про заявки користувачів на участь у подіях. Для реалізації відповідних зв'язків 1:N, в таблиці містяться зовнішні ключі на користувачів та події)

```
CREATE TABLE Заявки (
  id_заявки INT PRIMARY KEY AUTO_INCREMENT,
  id_користувача INT,
  id_події INT,
  статус VARCHAR(50),
  дата_створення DATETIME DEFAULT CURRENT_TIMESTAMP,
  FOREIGN KEY (id_користувача) REFERENCES Користувачі(id_користувача),
  FOREIGN KEY (id_події) REFERENCES Події(id_події)
);
```

Таблиця 2.4 – Опис полів таблиці «Заявки» (Applications)

№	Назва поля	Тип даних	Ключ	Опис
1	id_заявки	INT	PK	Унікальний ідентифікатор заявки
2	id_користувача	INT	FK	Зовнішній ключ на «Користувачі»
3	id_події	INT	FK	Зовнішній ключ на «Події»
4	статус	VARCHAR(50)	-	Статус заявки (на розгляді, прийнята, відхилено)
5	дата_створення	DATETIME	-	Дата подачі заявки

Лістинг 2.4 – Таблиця «Повідомлення» (Messages) (демонструє таблицю «Повідомлення», в котрій зберігаються повідомлення користувачів. Поле id_користувача є зовнішнім ключом, який забезпечує взаємозв'язок між повідомленням та автором)

```
CREATE TABLE Повідомлення (
  id_повідомлення INT PRIMARY KEY AUTO_INCREMENT,
  id_користувача INT,
  текст TEXT NOT NULL,
  дата_створення DATETIME DEFAULT CURRENT_TIMESTAMP,
  FOREIGN KEY (id_користувача) REFERENCES Користувачі(id_користувача)
);
```

Таблиця 2.5 – Опис полів таблиці «Повідомлення» (Messages)

№	Назва поля	Тип даних	Ключ	Опис
1	id_повідомлення	INT	PK	Унікальний ідентифікатор повідомлення

2	id_користувача	INT	FK	Автор повідомлення (зовнішній ключ)
3	текст	TEXT	-	Текст повідомлення
4	дата_створення	DATETIME	-	Дата і час створення повідомлення

Лістинг 2.5 – Таблиця «КатегоріяПодій» (EventCategories) (містить таблицю «КатегоріяПодій», яка описує всі можливі категорії подій. Поле id_категорії є первинним ключем таблиці)

```
CREATE TABLE КатегоріяПодій (
  id_категорії INT PRIMARY KEY AUTO_INCREMENT,
  назва VARCHAR(100) NOT NULL
);
```

Таблиця 2.6 – Опис полів таблиці «КатегоріяПодій» (EventCategories)

№	Назва поля	Тип даних	Ключ	Опис
1	id_категорії	INT	PK	Унікальний ідентифікатор категорії
2	назва	VARCHAR(100)	-	Назва категорії події

Лістинг 2.6 – Таблиця «ПодіяКатегорія» (EventCategoryMapping) (показує проміжну таблицю «ПодіяКатегорія», що реалізує зв’язок багато-до-багатьох між подіями та категоріями. Складений первинний ключ (id_події, id_категорії) забезпечує унікальність кожного поєднання події та категорії)

```
CREATE TABLE ПодіяКатегорія (
  id_події INT,
  id_категорії INT,
  PRIMARY KEY (id_події, id_категорії),
  FOREIGN KEY (id_події) REFERENCES Події(id_події),
  FOREIGN KEY (id_категорії) REFERENCES КатегоріяПодій(id_категорії)
);
```

Таблиця 2.7 – Опис полів таблиці «ПодіяКатегорія» (EventCategoryMapping)

№	Назва поля	Тип даних	Ключ	Опис
1	id_події	INT	PK/FK	Зовнішній ключ на Події (частина первинного ключа)
2	id_категорії	INT	PK/FK	Зовнішній ключ на КатегоріяПодій (частина первинного ключа)

Таким чином, в лістингах 2.1–2.6 я навів SQL-код створення таблиць бази даних, за допомогою яких реалізується логічна модель, представлена на ER-діаграмі. Тут кожна таблиця відповідає окремій сутності та містить необхідні поля, первинні (PK) та зовнішні ключі (FK) та з метою обмеження для забезпечення цілісності даних.

Для підтвердження технічної реалізації бази даних наведу DBML-код, за допомогою якого була зґенерована ER-діаграма.

Лістинг 2.8 – DBML-код бази даних, відповідний ER-діаграмі

```
Table Users {
  user_id int [pk]
  name varchar
  email varchar [unique]
  password varchar
  role varchar
  created_at date
}

Table Events {
  event_id int [pk]
  title varchar
  description text
  start_date date
  end_date date
  organizer_id int [ref: > Users.user_id]
}

Table Applications {
  application_id int [pk]
  user_id int [ref: > Users.user_id]
  event_id int [ref: > Events.event_id]
  status varchar
  created_at date
}
```

Продовження лістингу 2.8

```
Table Messages {
  message_id int [pk]
  user_id int [ref: > Users.user_id]
  text text
  created_at date
}

Table EventCategories {
```

```
  category_id int [pk]
  name varchar
}

Table EventCategoryMapping {
  event_id int [ref: > Events.event_id, pk]
  category_id int [ref: > EventCategories.category_id, pk]
}
```

Таким чином,структура бази даних задає основу для формування об’єктної моделі розробки, тому, після визначення таблиць і зв’язків, я виконав побудову UML-діаграми класів, яка відображатиме внутрішню логіку програмних сутностей.

2.4 Побудова UML-діаграм класів

Для відображення структури та логіки взаємодії об’єктів системи я побудував UML-діаграму класів, яка демонструє основні класи, їх атрибути, при потребі методи, а також зв’язки між класами.

Далі перелічу основні кроки побудови діаграми, отже:

1. Визначив класи, відповідно до таблиць бази даних: Користувач, Подія, Заявка, Повідомлення, КатегоріяПодії, ПодіяКатегорія.

2. Для кожного класу навів атрибути, які відповідають полям таблиць бази даних. Первинні та зовнішні ключі відзначив відповідними позначками {PK} та {FK}.

3. Встановив зв’язки між класами:

Один користувач може організовувати багато подій (1:N).

Один користувач може подавати багато заявок (1:N).

Одна подія може мати багато заявок (1:N).

Один користувач може надсилати багато повідомлень (1:N).

Зв’язок багато-до-багатьох між подіями та категоріями подій через проміжну таблицю ПодіяКатегорія.

4. Для побудови діаграми використав PlantUML, що дозволяє генерувати графічне представлення класів за текстовим описом.

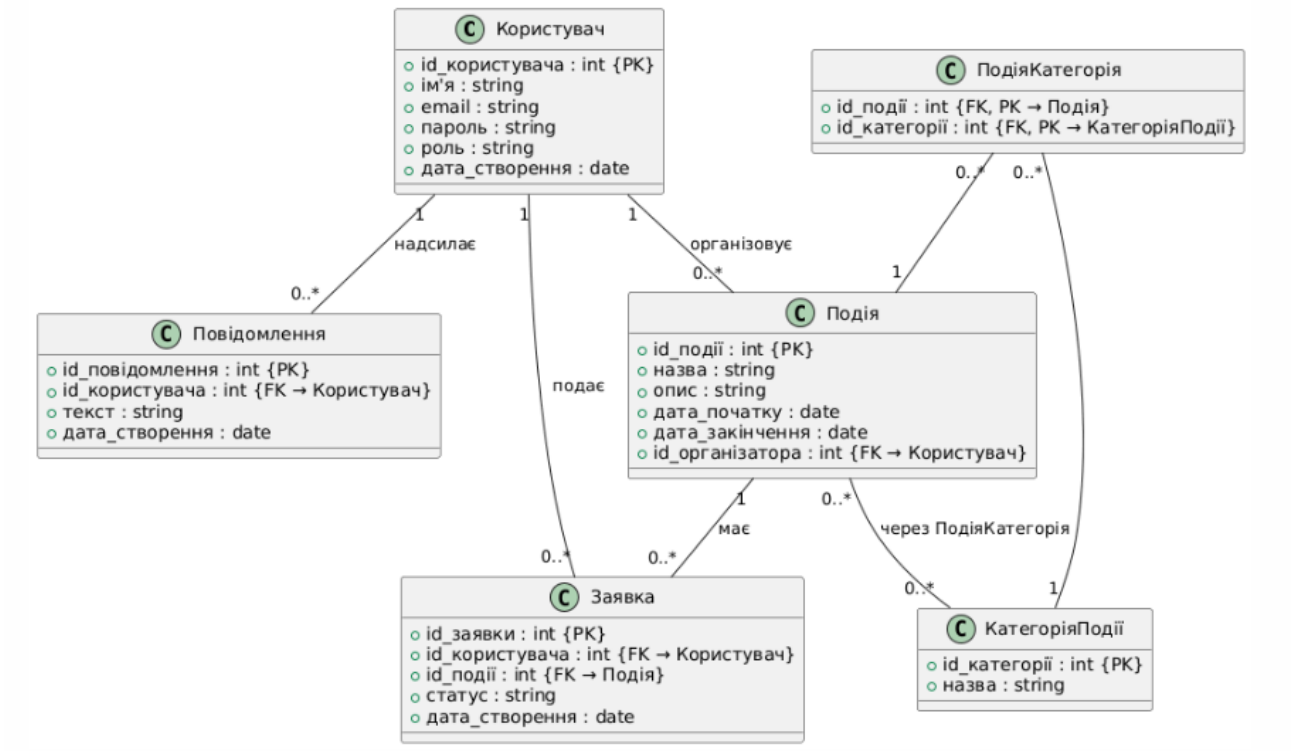


Рисунок 2.6 – UML-діаграма класів бази даних веб-платформи

Далі вважаю за потрібне навести короткий опис класів UML:

Клас «Користувач»: містить інформацію про користувачів платформи, включно з організаторами та волонтерами.

Клас «Подія»: зберігає інформацію про волонтерські події, включаючи організатора, дати та опис.

Клас «Заявка»: представляє заявки користувачів на участь у подіях, зберігає статус та дату створення.

Клас «Повідомлення»: відповідає за обмін повідомленнями між користувачами.

Клас «КатегоріяПодії»: містить категорії подій для зручної класифікації.

Клас «ПодіяКатегорія»: проміжний клас для реалізації зв'язку багато-до-багатьох між подіями та категоріями.

Для побудови діаграми використав PlantUML-код (Лістинг 2.7).

```
@startuml
class Користувач {
```

```
  +id_користувача : int {PK}
  +ім'я : string
  +email : string
  +пароль : string
  +роль : string
  +дата_створення : date
}
class Подія {
  +id_події : int {PK}
  +назва : string
  +опис : string
  +дата_початку : date
  +дата_закінчення : date
  +id_організатора : int {FK → Користувач}
}
class Заявка {
  +id_заявки : int {PK}
  +id_користувача : int {FK → Користувач}
  +id_події : int {FK → Подія}
  +статус : string
  +дата_створення : date
}
class Повідомлення {
  +id_повідомлення : int {PK}
  +id_користувача : int {FK → Користувач}
  +текст : string
  +дата_створення : date
}
class КатегоріяПодії {
  +id_категорії : int {PK}
  +назва : string
}
class ПодіяКатегорія {
  +id_події : int {FK, PK → Подія}
  +id_категорії : int {FK, PK → КатегоріяПодії}
}
Користувач "1" -- "0..*" Подія : організовує
Користувач "1" -- "0..*" Заявка : подає
Подія "1" -- "0..*" Заявка : має
Користувач "1" -- "0..*" Повідомлення : надсилає
Подія "0..*" -- "0..*" КатегоріяПодії : через ПодіяКатегорія
ПодіяКатегорія "0..*" -- "1" Подія
ПодіяКатегорія "0..*" -- "1" КатегоріяПодії
@enduml
```

2.5 Вибір мови програмування і обґрунтування технологій розробки

Для розробки веб-платформи, основним призначенням якої є волонтерська діяльність, попередньо я провів аналіз доступних технологій та методів організації

процесу розробки. Основною моєю метою було забезпечити швидку реалізацію, надійність, простоту підтримки та масштабованість системи.

Для клієнтської та серверної частин платформи я зупинився на JavaScript як основній мові програмування [46-48].

Серверна частина:

Node.js – забезпечить асинхронне виконання запитів, високу продуктивність та дозволяє створювати масштабовані web-сервіси. До переваг я віднесу, що достатньо однієї мови як для фронтенду так і для бекенду, передбачено велику система модулів, підтримку RESTful API, легку інтеграція з базами даних.

Стосовно клієнтської частини:

React.js – бібліотека для побудови інтерфейсів користувача. Суттєвими перевагами для розроблюваного продукту є компонентна архітектура, можливість повторного використання компонентів, інтерактивний UI, велика спільнота та готові бібліотеки для форм, таблиць та навігації.

Для бази даних:

PostgreSQL / MySQL – реляційна база даних для зберігання користувачів, подій, заявок, повідомлень та категорій. Перевагами я назву можливість структурованого зберігання данних, підтримку складних запитів, високу надійність, можливість транзакцій та забезпечення цілісності даних.

2.6 Вибір середовища та інструментів для розробки

Для ефективності здійснення розробки я зупинив вибір на наступних, оптимальних на мою думку, інструментах:

VS Code – як основне середовище розробки із досить великою кількістю плаґінів для JavaScript, Node.js та React[49].

Postman – для тестування, налагодження API, перевірки запитів та відповідей сервера[50].

dbdiagram.io – з призначенням для побудови та візуалізації структури бази даних, що дозволяє наочно демонструвати зв’язки між таблицями.

PlantUML / PlantText – з метою ґенерації UML-діаграм класів та зв’язків, при наявності яких значно спрощується розуміння архітектури системи.

Git / GitHub – для контролю версій та спільної роботи над проєктом, також зручно, що забезпечує відновлення історії змін та передбачено командну розробку.

2.7 Методологія розробки

В першому розділі я вже писав, що оптимально для обраного предмету та з врахуванням специфіки волонтерської діяльності і писав, що для організації процесу розробки, я свій вибір зупиню на гнучкому Agile-підході, який дасть мені змогу:

1 - планувати та поетапно реалізовувати функціонал,

2 – швидко та вчасно реагувати на зміни вимог користувачів,

3 – тестувати, а також отримувати зворотний зв’язок одразу після кожного спринту.

До основних переваг Agile саме для реалізації мого проєкту, підсумовуючи я відніс:

1 - можливість швидкої адаптації функціоналу під потреби волонтерських ініціатив;

2 – можливість розприділювати по пріоритетності ключові функції, такі як: створення подій, подача заявок, обмін повідомленнями;

3 – зважаючи на те, що я розробляю базовий проєкт, який можна розширювати, важливою є можливість ефективної командної роботи, розподілу завдань поміж розробниками та прозорість виконання поставлених задач;

4 – постійне вдосконалення системи на основі підтримки зворотного зв’язку від користувачів.

Таким чином, поєднання JavaScript + React+ Node.js, а також реляційної бази даних та Agile-підходу для моєї розробки є оптимальним та дасть змогу забезпечити:

1 - швидку та ефективну розробку,

2 - можливість масштабування та підтримку системи в цілому,

3 - гнучкість при внесенні змін,

4 - прозорість і зручність при необхідності залучення до роботи додатково членів команди.

Таким чином, обраний стек технологій і методологія розробки максимально відповідають потребам розроблюваної мною web-платформи для оптимізації волонтерської роботи і дозволять реалізувати продукт у найкоротші терміни при врахуванні специфіки та високій якості кінцевого продукту.

Після детального проєктування бази даних, побудови ER-діаграм та UML-діаграм класів, а також визначення стеку технологій і методології розробки, я планую перейти до наступного розділу в якому наведу основні етапи щодо технічної реалізації веб-платформи на основі попередньо спроєктованої архітектури та обраного стеку технологій.

2.8 Реалізація основних класів та методів

У цьому підрозділі я продемонструю, як реалізував основну логіку системи на основі побудованих UML-діаграм класів та ER-діаграм бази даних. Далі я представлю основні класи, їх атрибути та методи, а також надам приклади взаємодії між ними в контексті волонтерської діяльності.

1. Клас «User» – відповідає за інформацію про користувачів платформи та їхні дії.

```
class User {
    constructor(id, name, email, password, role, createdAt) {
        this.id = id;
        this.name = name;
        this.email = email;
        this.password = password;
        this.role = role; // volunteer або organizer
        this.createdAt = createdAt;
    }

    register() { /* Логіка реєстрації користувача */ }
```

```
    login(email, password) { /* Логіка авторизації */ }

    updateProfile(newData) { Object.assign(this, newData); }
```

2. Клас «Event» – відповідає за створення та керування волонтерськими подіями.

```
class Event {
    constructor(id, title, description, startDate, endDate,
organizerId) {
        this.id = id;
        this.title = title;
        this.description = description;
        this.startDate = startDate;
        this.endDate = endDate;
        this.organizerId = organizerId;
    }

    createEvent() { /* Створення події */ }

    updateEvent(newData) { Object.assign(this, newData); }

    deleteEvent() { /* Видалення події */ }
}
```

3. Клас «EventCategory» – відповідає за категорії подій та їх зв’язки з подіями.

```
class EventCategory {
    constructor(id, name) {
        this.id = id;
        this.name = name;
    }

    assignToEvent(eventId) { /* Призначення категорії події */ }
}
```

4. Клас «Application» – відповідає за заявки волонтерів на участь у подіях.

```
class Application {
    constructor(id, userId, eventId, status, createdAt) {
        this.id = id;
        this.userId = userId;
        this.eventId = eventId;
        this.status = status; // pending, accepted, rejected
        this.createdAt = createdAt;
    }
```

```
    submitApplication() { /* Подання заявки */ }

    updateStatus(newStatus) { this.status = newStatus; }
}
```

5. Клас «Message» – відповідає за обмін повідомленнями між користувачами.

```
class Message {
    constructor(id, userId, text, createdAt) {
        this.id = id;
        this.userId = userId;
        this.text = text;
        this.createdAt = createdAt;
    }

    sendMessage() { /* Відправка повідомлення */ }
}
```

Далі я покажу варіант взаємодії класів для волонтерської платформи

```
// Створення користувача-волонтера
let volunteer = new User(1, "Ростислав", "rostyk@gmail.com",
"vol_pass123", "volunteer", new Date());

// Створення користувача-організатора
let organizer = new User(2, "Лагола", "orest@gmail.com",
"org_pass456", "organizer", new Date());

// Організатор створює подію
let event = new Event(1, "Допомога постраждалим Тернопіль", "Збір
одягу та фінансова допомога", new Date('2025-11-19'), new
Date('2025-11-20'), organizer.id);
event.createEvent();

// Волонтер подає заявку на участь у події
let application = new Application(1, volunteer.id, event.id,
"pending", new Date());
application.submitApplication();

// Волонтер та організатор обмінюються повідомленнями
let message1 = new Message(1, volunteer.id, "Я хотів би допомогти
постраждалим", new Date());
message1.sendMessage();

let message2 = new Message(2, organizer.id, "Дякую за не байдужість,
ми розраховуємо на вас та очікуємо 19 листопада", new Date());
message2.sendMessage();
```

Наданий приклад варіанту взаємодії продемонстровано в таблиці 2.8

Таблиця 2.8 – Варіант взаємодії

User ↔ Event	– організатор створює волонтерську подію
User ↔ Application ↔ Event	– волонтер подає заявку на участь
User ↔ Message	– обмін повідомленнями між волонтером і організатором
Event ↔ EventCategory	– подія може бути віднесена до однієї або кількох категорій, наприклад «Допомога постраждалим», «Соціальні ініціативи», «Фінансова допомога»

У підрозділі 2.4 я продемонстрував реалізацію основних класів та методів web-платформи для волонтерських ініціатив. Коди класів та приклал взаємодії демонструють:

- структуру класів: User, Event, EventCategory, Application та Message;

- основні атрибути та методи, якими забезпечується функціональність платформи;

- приклад взаємодії класів, який ілюструє реальні сценарії підтверджуючи концепцію UML-діаграми (див. підроздів 2.4): створення події організатором, подання заявки волонтером та обмін повідомленнями;

- готовність системи для подальшої розробки та інтеграції з базою даних.

Таким чином, реалізація класів та методів обгрунтовує технічне підґрунтя платформи та її готовність вже до подальшої розробки, далі до інтеграції із базою даних та впровадження функціоналу вже безпосередньо для користувачів.

2.9 Розробка інтерфейсу користувача

Метою цього підрозділу є продемонструвати підхід щодо створення інтерфейса зі сторони користувача волонтерської web-платформи, основною функцією якого є забезпечення інтуїтивної, зручної та ефективної взаємодії користувачів (волонтерів, організаторів подій) безпосередньо із системою.

В зв’язку із цим при розробці я враховував сучасні принципи дизайну інтерфейса зі сторони користувача, а саме:

1. Простоту та зрозумілість. Інтерфейс повинен бути легким для сприйняття користувачами різного рівня підготовки, тобто розрахований на широку цільову аудиторію. З цією ж метою доцільно також мінімізувати кількість зайвих чи додаткових елементів і дотримуватись чіткості та однозначності структури меню та кнопок.

2. Адаптивність та доступність. Інтерфейс повинен підлаштовуватись під різні ґаджети: мобільні телефони, планшети, десктопи. Використання семантичних тегів HTML для можливості використання продукту користувачами з обмеженими можливостями (screen readers).

3. Консистентність. Одинаковість та однотипність стилів, шрифтів та компонентів на всіх сторінках платформи. Створення бібліотеки повторно використовуваних компонентів в React.js.

4. Інтерактивність та зворотний зв’язок. Можливість миттєвого оновлення даних без необхідності перезавантажувати сторінки (AJAX, React). Підтвердження дій користувача через спливаючі повідомлення та візуальні індикатори.

Якщо говорити про структура інтерфейсу в цілому, то далі в вигляді таблиці 2.9 я перелічу основні розділи платформи та їх функції.

Таблиця 2.9 – Опис полів таблиці «ПодіяКатегорія» (EventCategoryMapping)

1.	Головна сторінка:	Вітальний блок із коротким описом платформи. Панель пошуку та фільтрації подій за категоріями, датою та місцем проведення. Відображення популярних або актуальних подій.
2.	Профіль користувача	Персональні дані волонтера чи організатора. Список подій, у яких користувач бере участь або які організовує. Можливість редагувати дані профілю та змінювати налаштування облікового запису.

3.	Сторінка подій	Детальна інформація про подію: назва, опис, дата, місце, категорія. Кнопка подання заявки для волонтерів із підтвердженням статусу. Відображення кількості учасників та контактної інформації організатора.
4.	Система повідомлень	Обмін повідомленнями між волонтерами та організаторами. Історія листування для зручного відстеження комунікації. Можливість відправляти повідомлення безпосередньо зі сторінки події.
5.	Адміністративна панель (для організаторів)	Керування подіями: створення, редагування, видалення. Модерація заявок: прийняття або відхилення волонтерів. Можливість надсилати масові повідомлення учасникам події.

Далі я обрав сучасні веб-технології для реалізації інтерфейса, зокрема:

HTML5 та CSS3 – забезпечують семантичну структуру та стилізацію сторінок.

JavaScript – додає інтерактивність та обробку дій користувача.

React.js – дозволяє створювати компонентний, повторно використовуваний інтерфейс.

Bootstrap – забезпечує адаптивний дизайн та готові UI-компоненти.

Axios /Fetch API – для обміну даними з сервером без перезавантаження сторінки.

Для наочної демонстрації взаємодії користувачів із платформою я пропоную розглянути приклад послідовності етапів типового сценарію:

1. «Волонтер» реєструється та входить у систему.
2. «Волонтер» переглядає події на головній сторінці та застосовує фільтри.
3. «Волонтер» обирає подію та подає заявку через сторінку події.
4. «Волонтер» отримує підтвердження про статус заявки та має можливість зв’язатися із організатором/ми через систему повідомлень/сповіщень.
5. «Організатор» отримує повідомлення про нову заявку та може її підтвердити, таким чином оновлюючи статус для користувача-волонтера.

Представлений варіант сценарію демонструє логіку взаємодії класів User, Event, Application та Message, яку вже реалізовано у коді підрозділу 2.6.

Таким чином, розробка інтерфейса користувача забезпечує:

1 - інтуїтивну та зрозумілу роботу для волонтерів і організаторів.

2 - швидкий доступ до ключових функцій платформи: подій, заявок та повідомлень.

3 - адаптивність і сучасний зовнішній вигляд завдяки використанню React.js та Bootstrap.

4 - можливість подальшого масштабування та інтеграції нових функцій без кардинальної зміни структури UI.

2.10 Реалізація логіки взаємодії користувачів

Далі я описав, як було реалізовано бізнес-логіку платформи для полегшення, якщо так можна сказати, волонтерської діяльності, а саме як відбувається взаємодія користувачі із системою, опрацьовуються дані та забезпечується основний функціонал.

Основні сценарії взаємодії

1. Реєстрація та аутентифікація користувачів.

Користувач створює обліковий запис (ім’я, email, пароль, роль: волонтер або організатор).

Система перевіряє унікальність e-mail і надійність паролю.

Після успішної аутентифікації користувач «потрапляє» у свій профіль.

2. Створення та керування подіями (для організаторів).

Організатор створює подію з назвою, описом, датою та категорією.

Події зберігаються у таблиці Event та пов’язуються з користувачем через id_організатора.

Організатор може редагувати або видаляти події та переглядати список заявок від волонтерів.

3. Подання та обробка заявок (для волонтерів).

Волонтер обирає подію та подає заявку.

Заявка зберігається у таблиці Application з посиланням на користувача та подію.

Організатор підтверджує чи ж відхиляє заявку, статус змінюється після чого відображається волонтеру.

4. Обмін повідомленнями.

Користувачі можуть обмінюватись повідомлення один з одним через систему (Message).

Повідомлення прив’язані до користувача через id_користувача і відображаються у хронологічному порядку.

5. Пошук та фільтрація подій.

Користувач може шукати події за категоріями, датою чи місцем проведення.

Система обробляє запити та повертає відповідні події.

Для підтвердження реалізації бізнес-логіки далі на рис. 2.7 я наведу PlantUML-код, яким описав основні дії користувачів та організаторів платформи волонтерських ініціатив.

Лістинг 2.7 - PlantUML-код для реалізації логіки взаємодії користувачів

```
@startuml
start
:Реєстрація / Вхід;
if (Email унікальний?) then (так)
  :Створення профілю;
else (ні)
  :Повідомлення про помилку;
endif
:Перегляд подій;
if (Волонтер подає заявку?) then (так)
  :Створення заявки;
  :Повідомлення організатора;
else (ні)
  :Продовжує перегляд;
endif
if (Організатор обробляє заявку?) then (так)
  :Підтвердження / Відхилення;
  :Оновлення статусу заявки;
  :Повідомлення волонтера;
endif
```

```
stop
@enduml
```

В результаі я отримав блок-схему логіки користувача (PlantUML) яку представлю на рисунку 2.7

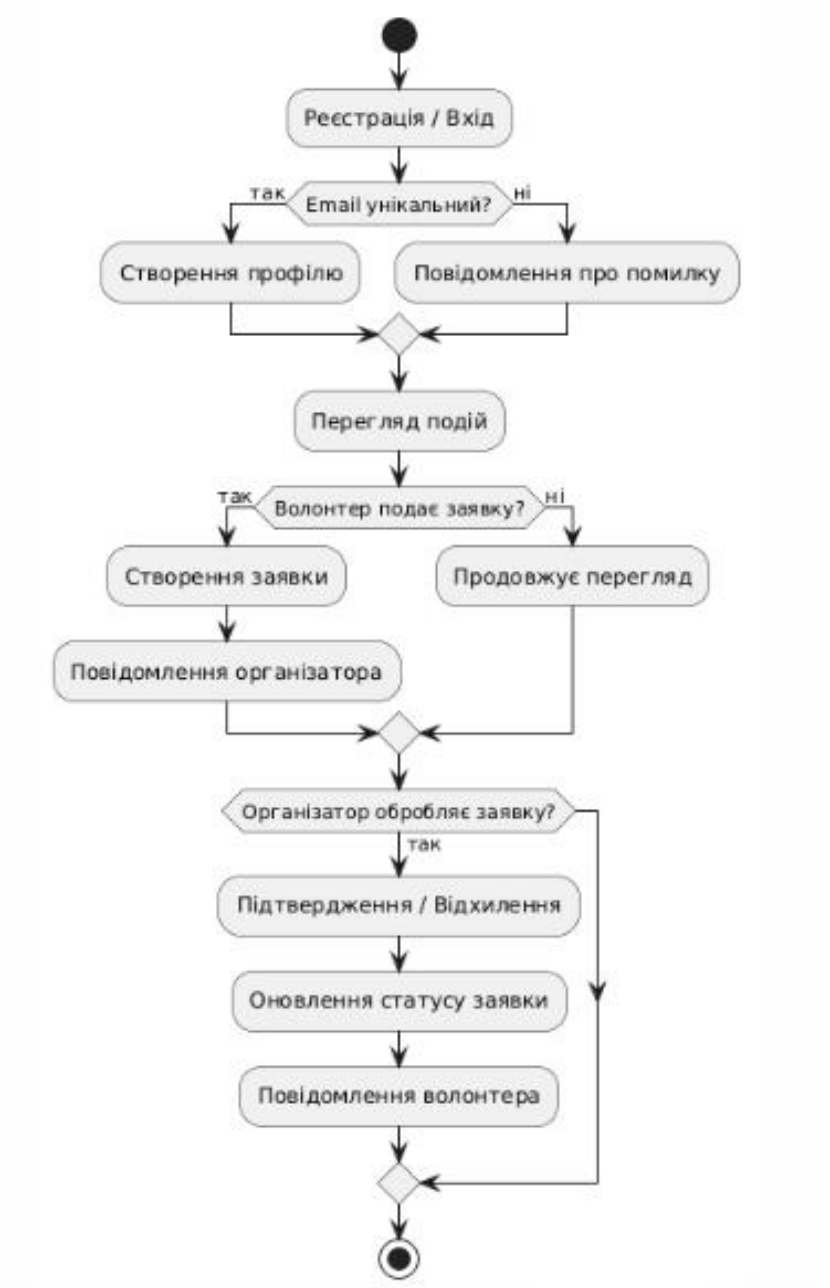


Рис. 2.7 Блок-схема логіки користувача

3 ТЕСТУВАННЯ, ВПРОВАДЖЕННЯ ТА ПІДТРИМКА

Тепер, коли основний функціонал платформи я вже створив, потрібно переконатися, що всі її частини працюватимуть правильно і будуть взаємодіяти між собою без будь-яких проблем. Отже, на етапі тестування я перевірятиму на наявність помилок в цілому, чи зручно користуватися інтерфейсом, і перевірю, як працюють різні модулі разом. Якщо щось поводиться не так як планувалось, я це зможу виправити, а процес повторюватиметься до тих пір, поки система не стане стабільною.

Після основних перевірок настає етап впровадження під час якого треба налаштувати сервер, підготувати робоче середовище та перевірити, чи платформа без проблем запускатиметься при реальних умовах. Проте, навіть після запуску робота не закінчується оскільки важливо слідкувати за стабільністю системи, оновлювати її та допомагати користувачам, якщо виникатимуть запитання.

Саме так, на мою думку, забезпечується надійна робота сервісу і зручність для всіх, для кого його призначено і хто ним користується.

3.1. Тестування програмної системи

Тестування веб-платформи є ключовим етапом забезпечення її функціональності, безпеки та відповідності вимогам користувачів [51-53]. На основі вимог до системи (пункт 1.2.1), описаних ключових сценаріїв (Use Case, підрозділ 1.4) та компонентів платформи (2.2.2) було розроблено план тестування, що охоплює модульне, інтеграційне, системне та приймальне тестування.

3.1.1. Види та план тестування

Якщо говорити про тестування цілому по видах, то оптимальними будуть наступні:

1-модульне тестування (Unit). Тут відбувається перевірка окремих модулів системи, зокрема:

- модуль керування користувачами: реєстрація, логін, редагування профілю;
- модуль подій: створення, редагування та публікація подій;
- модуль заявок: подання та управління заявками;
- модуль повідомлень: відправка e-mail та push-сповіщень;
- модуль адміністрування: модерація користувачів та подій.

2-інтеграційне тестування під час якого виконується перевірка взаємодії модулів між собою та із базою данних через REST API. Метою такого тестування є забезпечити правильну передачу даних між фронтендом і бекендом та коректну роботу логіки бізнес-процесів.

3-системне тестування – оцінка роботи платформи як єдиного цілого, сюди: обробка подій, подання заявок, сповіщення користувачів.

4-приймальне тестування (Acceptance). До цього виду відноситься перевірка відповідності платформи на вимоги користувачів і бізнес-вимоги: реєстрація користувачів, створення подій, подання і затвердження заявок, робота повідомлень.

Далі на представлю план проведення тестування:

1-Unit-тести виконуються після реалізації кожного модуля;

2-інтеграційні тести проводяться після об’єднання модулів;

3-системні та приймальні тести виконуються перед впровадженням і верифікацією платформи.

3.1.2. Розробка тестових сценаріїв

На основі ключових сценаріїв Use Case я сформував тест-кейси для перевірки функціональних та нефункціональних вимог розроблюваної web-платформи [54-55].

Таблиця 3.1 – Тест-кейси для Use Case “Реєстрація користувача”

№	Назва тесту	Передумови	Кроки	Очікуваний результат
1	Успішна реєстрація	Користувач не зареєстрований	Відкрити сторінку реєстрації, ввести ім’я, e-mail, пароль, підтвердити e-mail	Обліковий запис створено, доступ надано
2	Некоректні дані	—	Ввести некоректні дані	Повідомлення про помилку, підсвічено поле з помилкою
3	E-mail уже зареєстрований	Користувач у базі даних	Ввести існуючий e-mail	Повідомлення “E-mail вже використовується”

Таблиця 3.2 – Тест-кейси для Use Case “Створення події”

№	Назва тесту	Передумови	Кроки	Очікуваний результат
1	Успішне створення події	Користувач – Організатор	Відкрити сторінку “Створити подію”, ввести дані події, натиснути “Публікувати”	Подія створена, відображається у списку
2	Некоректні дані	—	Ввести некоректні дані	Повідомлення про помилку, подія не створена

Таблиця 3.3 – Тест-кейси для Use Case “Подання заявки на участь”

№	Назва тесту	Передумови	Кроки	Очікуваний результат
1	Успішне подання заявки	Користувач – Волонтер, зареєстрований	Вибрати подію, натиснути “Подати заявку”	Система підтверджує отримання заявки
2	Дубль заявки	Волонтер вже подав заявку	Подати заявку ще раз	Повідомлення про дубль заявки

Таблиця 3.4 – Тест-кейси для Use Case “Затвердження/Відхилення заявки”

№	Назва тесту	Передумови	Кроки	Очікуваний результат
1	Затвердження заявки	Організатор має створену подію та список заявок	Відкрити список заявок, натиснути “Затвердити”	Заявка підтверджена, користувач отримує повідомлення
2	Відхилення заявки	—	Натиснути “Відхилити”	Заявка відхилена, користувач отримує повідомлення

Крім функціональних тестів, передбачено не функціональні, до яких віднесу: продуктивність (до 1000 одночасних користувачів), безпека (шифрування паролів,

контроль доступу), надійність (резервне копіювання) та юзабіліті (інтерфейс на мобільних і десктопних пристроях).

3.2. Розгортання програмної системи та системні вимоги

Важливо також перелічити системні вимоги:

до клієнтської частини: сучасні браузери (Chrome, Firefox, Edge), адаптивний дизайн;

до серверної частина: веб-сервер (Node.js/PHP/Python), REST API, база даних SQL/NoSQL;

до обладнання: мінімум 2 ядра CPU, 4 ГБ RAM, стабільне підключення до мережі.

Стосовно розгортання:

1. Локальне середовище для розробки та тестування.
2. Розгортання на хмарному сервері для доступу користувачів.
3. Налаштування бази даних і підключення всіх модулів системи.
4. Конфігурація системи повідомлень (e-mail, push).

3.3. Верифікація програмної системи

Верифікація полягає у перевірці відповідності реалізованої платформи до заявлених вимог, зокрема щодо: функціональних вимог **(**реєстрація, логін, створення подій, подання та затвердження заявок, повідомлення); не функціональних вимог **(**продуктивність, безпека, надійність, зручність користування); бізнес-вимог **(**забезпечення простоти адміністрування та участі волонтерів у подіях).

3.4. Підтримка та супровід системи

Після впровадження web-платформи вважаю за правльне передбачити регулярну підтримку та супровід, що дасть можливість забезпечити і стабільну роботу і відповідність новим вимогам користувачів. До основних заходів підтримки я би включив:

1 -моніторинг працездатності, тобто контроль за роботою серверів, API, бази даних та модулів системи.

2 -виправлення помилок (Bug Fixing) – вчасне усунення виявлених проблем у функціонуванні платформи.

3 -оновлення функціонала, зокрема додавання нових можливостей та покращення існуючих модулей на основі зворотного зв’язку від користувачів.

4 -резервне копіювання та відновлення даних, а саме регулярне, періодичне створення резервних копій бази даних та налаштувань платформи для того, щоби запобігти втраті данних.

5 -технічна підтримка користувачів. Сюди віднесу консультації для волонтерів та організаторів щодо питань пов’язаних із використанням платформи, усунення можливих проблем при реєстрації, поданні заявок та/або при інших процесах.

Таким чином, в цьому підрозділі я продемонстрував, що платформу я не тільки розробляв і протестував, а й забезпечив подальшу підтримку, оскільки це вкрай важливо, оскільки я пропоную розробку для довгострокового успішного використання.

Під час тестування я самостійно контролював роботу модулів на локальному сервері та аналізував повідомлення про помилки, що дозволило оптимізувати процес обробки заявок і покращити стабільність платформи

Для наочності та контролю я додатково підготував матрицю відповідності тестів вимогам, і це дозволить оцінити покриття тестування та готовність платформи до її впровадження. Тобто представлена в таблиці 3.5 матриця

відображає, якими тестами перевіряються конкретні функціональні, не функціональні, а також бізнес-вимоги.

Таблиця 3.5 – Матриця відповідності тестів вимогам

Вимога	Тип вимоги	Тест-кейси / Сценарії	Примітки
Реєстрація користувачів	Функціональна	ТК 3.1.1, 3.1.2, 3.1.3	Перевірка облікового записа, дублів, некоректних даних
Логін / Аутентифікація	Функціональна	ТК 3.1.1, 3.1.2	Перевірка доступу для волонтерів та організаторів
Створення подій	Функціональна	ТК 3.2.1, 3.2.2	Перевірка публікації, категоризації, редагування
Подання заявок на участь	Функціональна	ТК 3.3.1, 3.3.2	Перевірка успішного подання та дублювання
Затвердження / Відхилення заявок	Функціональна	ТК 3.4.1, 3.4.2	Перевірка сповіщень організаторів та волонтерів
Повідомлення користувачів	Функціональна	ТК 3.1–3.4	E-mail та push-повідомлення при зміні статусу
Продуктивність (до 1000 користувачів одночасно)	Нефункціональна	Навантажувальне тестування	Перевірка швидкодії та стійкості сервера
Безпека (шифрування паролів, контроль доступу)	Нефункціональна	Тести безпеки	Перевірка облікових записів, захист даних
Надійність / Резервне копіювання	Нефункціональна	Тест відновлення даних	Перевірка збереження даних при збоях
Юзабіліті (зручність для мобільних та десктопних пристроїв)	Нефункціональна	UX-тести	Перевірка інтерфейсу, адаптивності
Підвищення участі волонтерів	Бізнес-вимога	Acceptance-тестування	Перевірка функціоналу подачі та затвердження заявок
Простота адміністрування	Бізнес-вимога	Тести модуля адміністрування	Перевірка модерації користувачів та подій

Таким чином, в таблиці 3.5 представлено логічний зв’язок між вимогами і тестами, а також продемонстровано, що всі важливі функції та не функціональні аспекти платформи перевірено.

3.5. План подальшого розвитку та масштабування платформи

Важливим, на мою думку, аспектом підтримки цуи-платформи є врахування та планування її подальшого розвитку та масштабування. Це підхід на перспективу та включає наступні можливості:

1. Розширення функціоналу (додавання нових типів подій, інтеграція з платіжними або соціальними сервісами, створення рейтингів волонтерів та системи винагород, інше).

2. Оптимізація продуктивності (впровадження кешування, балансування/розприділення навантаження та оптимізація запитів до бази даних для забезпечення швидкої обробки понад 1000 користувачів одночасно).

3. Інтеграція з мобільними платформами (розробка мобільних додатків або адаптивного веб-інтерфейсу під Android та iOS, що дозволить волонтерам швидко подавати заявки та отримувати сповіщення).

4. Підтримка багатомовності (розширення платформи для користувачів із різних регіонів, що дасть змогу підвищити охоплення та збільшити доступність).

3.6. Документування процесів тестування та супроводу

З метою для забезпечення послідовності та повторюваності процесів тестування та підтримки розроблюваною web платформи я рекомендував би створювати повну документацію, яка повинна включати:

- детальний опис модулей та їхніх функцій;

- детальні тест-кейси з наведенням результатів їх виконання;

- обов’язкові процедури резервного копіювання та передбачену можливість відновлення даних;

- наявність журналу помилок та способи їх усунення;

- інструкції як для користувачів так і для адміністраторів щодо основних дій у системі.

Таке документування, на мою думку, дозволить новим членам команди швидко адапттовуватися та підтримувати web-платформу на достатньо високому рівні якості.

3.7. Висновки стосовно тестування, впровадження та підтримки

Таким чином, враховуючи все вище сказане, можна зазначити, що комплексне тестування, системне впровадження та організована підтримка дадуть змогу забезпечити:

Стабільність роботи системи. Користувачі отримують безперервний доступ до всіх функцій.

Високу якість продукту. Завдяки тестуванню на різних рівнях (модульному, інтеграційному, системному, приймальному) можна реально ґарантувати відповідність вимогам до розроблюваного продукту.

Безпеку та надійність. Йдеться про захист персональних даних, можливість резервного копіювання, важливі також можливості контролю за доступом та шифрування інформації.

Гнучкість для розвитку. Один із найважливіших пунктів, оскільки йдеться про можливість зміни вимог під реальні запити, оскільки у волонтерській діяльності, особливо під час військового стану, без змін вимог, система на мою думку не мала б перспектив – можливість швидкого впровадження нових необхідних функцій та адаптації до змін потреб користувачів.

Таким чином, інтеграція всіх етапів розробки, тестування та супроводу дозволить мені забезпечити довгострокове ефективне функціонування платформи, дасть змогу підвищувати довіру користувачів і створить фундамент для подальшого розвитку розробленої web-платформи призначенням якої є волонтерська діяльність, яка життєво необхідна в наш, такий складний, військовий час.

Важливо зазначити, що ефективне тестування та регулярний супровід всіх програмних продуктів, в тому числі і моєї web-платформи, дозволять не лише

забезпечити надійну, безперебійну роботу, а й своєчасно реагувати на потреби користувачів та зміни у волонтерській діяльності. Постійний моніторинг та аналіз логів системи дозволяють виявляти «вузькі» місця у роботі модулів і підвищувати продуктивність системи.

Крім того, впровадження механізмів резервного копіювання та відновлення даних ґарантують збереження інформації навіть у разі можливих технічних якихось збоїв, що особливо критично для платформ, де передбачається опрацювання персональних даних користувачів.

Розробка та використання тестових сценаріїв на основі ключових варіантів використання також дозволять забезпечити високу якість кінцевого програмного продукту та підвищити довіру користувачів до системи, що в свою чергу сприятиме і моєму іміджу на ринку ІТ. Це створює також і основу для подальшого масштабування платформи, надасть можливість додавання нових модулів та інтеґрацій із зовнішніми сервісами.

Таким чином, поєднання комплексного тестування, системного впровадження та регулярного супроводу забезпечує стабільну та безпечну роботу веб-платформи, а також дозволяє оперативно адаптовувати її до нових вимог користувачів та волонтерської спільноти.

4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

Охорона та безпека праці та захист у надзвичайних ситуаціях є невід'ємною складовою сучасної інженерної діяльності, зокрема в галузі розробки програмного забезпечення. Незважаючи на те, що основна робота програміста або команди розробників відбувається у цифровому середовищі, відсутність належних заходів охорони праці може призвести до професійних захворювань, травм або психологічного вигорання.

У сучасних ІТ-компаніях та науково-дослідних лабораторіях особлива увага приділяється організації робочого місця, ергономіці, використанню безпечного обладнання та дотриманню правил електробезпеки. Крім того, розробка веб-платформ та програмного забезпечення передбачає взаємодію з електронними пристроями, серверним обладнанням та локальними мережами, що накладає додаткові вимоги до охорони праці.

Не менш важливим аспектом є готовність до надзвичайних ситуацій, таких як пожежі, короткі замикання, витік електроенергії, а також природні катастрофи та надзвичайні події техногенного характеру. Своєчасна організація планів евакуації, наявність засобів пожежогасіння, систем сигналізації та навчання персоналу забезпечує мінімізацію ризиків і гарантує безпеку користувачів і розробників.

Мета цього розділу — узагальнити основні принципи охорони праці та безпеки у надзвичайних ситуаціях, адаптовані до специфіки розробки програмного забезпечення, веб-платформ та сучасних інформаційних технологій. В розділі також розглядаються методи оцінки ризиків, правила організації робочого місця, нормативно-правова база України та рекомендації щодо підвищення безпеки робочих процесів.

4.1. Основи охорони праці

Для забезпечення належного рівня охорони праці та безпеки у надзвичайних ситуаціях в ІТ-сфері, зокрема при розробці веб-платформ, важливе дотримання чинного законодавства та державних стандартів. Правова база визначає обов'язки роботодавців і працівників, встановлює вимоги до організації безпечного робочого середовища та передбачає заходи запобігання надзвичайним ситуаціям. Нижче наведено основні нормативні документи, на які орієнтуються при створенні систем безпеки на робочому місці:

Закон України «Про охорону праці» № 2694-XII [57] Закон встановлює правові, економічні та організаційні основи охорони праці в Україні. Він визначає обов'язки роботодавців та працівників щодо створення безпечних і нешкідливих умов праці, правила розслідування нещасних випадків, вимоги до навчання та інструктажу працівників. Цей закон є основним нормативним документом, на який спираються підприємства та організації при впровадженні систем охорони праці, включно з ІТ-компаніями та командами розробників програмного забезпечення.

ДСТУ 2293:99 «Охорона праці. Терміни та визначення» [58] Державний стандарт України визначає основні терміни та поняття, що використовуються у сфері охорони праці. Він забезпечує єдине тлумачення ключових понять, таких як ризик, небезпека, засоби індивідуального захисту, професійні захворювання тощо. ДСТУ 2293:99 є важливим для правильного застосування норм законодавства та розробки внутрішніх положень безпеки на підприємстві, у тому числі в ІТ-сфері.

ДСТУ 9327:2025 «Пожежна безпека будівель і приміщень» [59] Стандарт встановлює вимоги до протипожежного захисту будівель і приміщень, що застосовуються для офісних, виробничих та навчальних приміщень. Він описує заходи запобігання займанням, організацію евакуаційних шляхів, облаштування систем пожежної сигналізації та вогнегасників. Для команд розробників веб-платформ дотримання цього стандарту гарантує безпеку робочого простору та

мінімізує ризик надзвичайних ситуацій у приміщеннях із комп’ютерним та серверним обладнанням.

Охорона праці в умовах використання веб-платформ та інформаційних технологій базується на загальних принципах забезпечення безпечних і здорових умов праці, визначених нормативно-правовими актами та державними стандартами України [57, 58]. Особливістю такої діяльності є переважна робота з персональними комп’ютерами, що зумовлює необхідність урахування специфічних ризиків та факторів впливу на здоров’я користувачів.

Одним із ключових принципів є раціональна організація робочих місць, яка передбачає дотримання вимог ергономіки, достатнього рівня освітлення та належної вентиляції приміщень [58]. Правильна організація робочого середовища сприяє зменшенню фізичного навантаження, запобігає перевтомі та негативному впливу на зір і опорно-руховий апарат користувачів.

Важливе місце займає принцип технічної безпеки, що полягає у дотриманні правил експлуатації персональних комп’ютерів, електричного та периферійного обладнання [61]. У межах веб-платформи реалізація цього принципу можлива шляхом розміщення інформаційних матеріалів і рекомендацій щодо безпечної роботи з технічними засобами.

З метою збереження здоров’я користувачів особлива увага приділяється профілактиці професійних захворювань, пов’язаних із тривалою роботою за комп’ютером. До таких заходів належать регламентовані перерви в роботі, виконання вправ для очей та дотримання режиму праці й відпочинку, що відповідає вимогам чинних нормативних документів [58].

Не менш важливим є принцип навчання та інструктажу з охорони праці, який включає проведення вступних, первинних і повторних інструктажів, а також навчання з надання першої домедичної допомоги [57, 60]. Веб-платформа може використовуватися як зручний інструмент для дистанційного проходження інструктажів, ознайомлення з нормативними вимогами та перевірки рівня знань користувачів.

Ефективна реалізація принципів охорони праці забезпечується через контроль та відповідальність, що передбачає ведення журналів інструктажів, фіксацію фактів навчання та контроль за дотриманням установлених вимог [57, 58]. Адміністратори веб-платформи можуть здійснювати такий контроль у цифровому форматі, що підвищує прозорість і зручність управління процесами охорони праці.

Все вище наведене регламентується міжнародними стандартами ISO 45001:2018 — управління охороною праці, аналіз ризиків, навчання персоналу [60] та ISO/IEC 27001:2022 — інформаційна безпека, захист даних користувачів [61].

Таким чином в підрозділі 4.1 «Охорона праці» я продемонстрував комплексний підхід до безпеки волонтерів і адміністраторів, інтегруючи фізичні, психологічні та інформаційні ризики, нормативну базу України та міжнародні стандарти ISO. Таблиці і схема демонструють практичну реалізацію безпеки через веб-платформу та контроль виконання інструктажів.

4.2. Безпека в надзвичайних ситуаціях

Безпека при надзвичайних ситуаціях є невід’ємною складовою будь-якої організованої діяльності, а волонтерська діяльність по своїй суті вже передбачає якісь не сприятливі, не типові події, які потребують і певних навиків і координування, тому моя тема напряму відповідає розділу, адже волонтери і їх діяльність напряму пов’язані із нетиповими ситуаціями. Вона охоплює комплекс заходів, спрямованих на захист та збереження життя, здоров’я, психологічного та психічного стану учасників. Для успішної роботи веб-платформи з волонтерських ініціатив необхідно враховувати всі можливі ризики: фізичні, психологічні та інформаційні.

У цьому підрозділі розглянуто основні положення щодо організації безпеки в надзвичайних ситуаціях, методи запобігання надзвичайним ситуаціям, а також інтеграцію цих заходів у роботу цифрової платформи [57-59, 62,63].

Безпека життєдіяльності і цілому – це комплекс організаційних, технічних та психологічних заходів, спрямованих на захист життя і здоров'я учасників волонтерської діяльності. Для веб-платформи безпека включає: фізичну, психологічну, інформаційну безпеку та готовність до надзвичайних ситуацій [57,58,60].

Далі в таблиці 4.2 я наведу основні ризики та заходи для їх мінімізації чи уникнення.

Таблиця 4.2. – Основні ризики, приклади ситуацій та заходи щодо їх мінімізації чи уникнення

Категорія ризику	**Приклади ситуацій**	**Заходи на платформі**
Фізичні	Травми під час подій, акцій, падіння	Онлайн-інструкції, алгоритми дій у НС, обов'язковий інструктаж перед подією [57,58]
Психологічні	Стрес, вигорання	Тренінги з управління стресом, рекомендації психологів [60]
Інформаційні	Витік даних користувачів	Двофакторна аутентифікація, шифрування, контроль доступу [61]
Надзвичайні ситуації	Пожежі, медичні інциденти	План евакуації, автоматичні сповіщення, форма для повідомлення про НС [59]

Також варта зазначити про можливості інтеграції безпеки в тому числі і при настанні надзвичайних ситуацій у веб-платформу. Сюди я відніс:

- модуль довідкових матеріалів (тексти, відео, інфографіка) щодо дій у надзвичайних ситуаціях [60];

- онлайн-тести та сертифікація волонтерів перед подіями [57,60];

- система сповіщень: нагадування про правила безпеки [57];

- форма для повідомлення про небезпеку [57,58].

В свою чергу навчання та інструктажі охоплюють правила безпеки, алгоритми дій у надзвичайних ситуаціях, використання засобів захисту, правила обробки персональних даних [57,57,60-63]. Результати інструктажів зберігаються на платформі для контролю організаторами [60,62,63].

ВИСНОВКИ

В результаті виконання кваліфікаційної роботи магістра метою якої було створення web-платформи, яка б враховувала специфіку волонтерської діяльності з врахуванням частої зміни вимог за використання сучасних, зокрема, веб-технологій та методології Agile.

В роботі було виконано поставлені завдання, а саме:

1 – Підвищити ефективність організації волонтерських подій, що дозволить зменшити навантаження на організаторів та забезпечить кращу координацію поміж учасниками.

2 – Забезпечити прозорий процес реєстрації та участі.

3 – Надати адміністратору інструменти контролю та управління користувачами, що включає моніторинг активностей та керування ролями.

4 – Забезпечити безпеку даних.

5 – Можливість для подальшого вдосконалення та розширення функціоналу.

В роботі я продемонстрував основні етапи життєвого циклу розробки програмного забезпечення, зокрема: процес проєктування, розробки, та тестування веб-платформи, яке реалізована як кластер із двох основних компонентів: web-сервісу для взаємодії користувачів та API для керування подіями та заявками з врахуванням особливостей роботи, переваг та недоліків існуючих сервісів типу VolunteerMatch, Добро.ua, LetsDoItUkraine.

Стосовно обґрунтовано обраних технологій, було обрано ітеративно-інкрементальний процес розробки (Agile-підхід), зокрема Scrum. Також я використав базу даних реляційного типу, наприклад PostgreSQL чи MySQL, через переваги та оптимальність для виконання саме моєї, специфічної, задачі. Для клієнтської та серверної частин платформи я зупинився на JavaScript як основній мові програмування. Для ефективності здійснення розробки я зупинив свій вибір на наступних інструментах: VS Code, Postman, dbdiagram.io, PlantUML / PlantText, Git / GitHub.

Таким чином, поєднання JavaScript + React+ Node.js, а також реляційної бази даних та Agile-підходу для моєї розробки є оптимальним та дасть змогу забезпечити: швидку та ефективну розробку, можливість масштабування та підтримку системи в цілому, гнучкість при внесенні змін, прозорість і зручність при необхідності залучення до роботи додатково членів команди.

Для реалізації інтерфейса я обрав сучасні веб-технології, зокрема: HTML5 та CSS3, JavaScript, React.js, Bootstrap, Axios /Fetch.

Використання можливостей сучасних web-технологій дозволило мені забезпечити гнучкість і масштабованість платформи, враховуючи специфіку волонтерської діяльності та потреби користувачів. Це ґарантує, що платформа буде зручною, інтегрованою з іншими сервісами і зможе легко розширюватися зокрема можна додавати нові функції або інтегрувати нових акторів (наприклад, гостей або сторонні сервіси) без порушення базової логіки платформи.

Таким чином, запропонований проєкт поєднуватиме зручність для користувачів, ефективну організацію волонтерських ініціатив, автоматизацію управління подіями та заявками, підвищить ефективність взаємодії між учасниками і ґарантує безпеку даних, одночасно відкриваючи можливості для подальшого розширення функціоналу.

Окрім цього, продукт буде інтуїтивно зрозумілим та зручним для користувачів, забезпечить безперебійну та безпечну роботу з великими даними з передбаченням можливості масштабування, щоб ефективно підтримувати велику кількість користувачів та подій.

Обов’язковим елементом кваліфікаційної роботи магістра є її апробація. Результати своєї роботи у вигляді тез опубліковано в матеріалах роботи XIII науково-технічної конференції «Інформаційні моделі, системи та технології», яка проводитиметься в період 17-18 грудня 2025 року.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Верховна Рада України. Закон України «Про волонтерську діяльність» від 19.04.2011 № 3236-VI / Офіційний сайт Верховної Ради України. URL: https://zakon.rada.gov.ua/go/3236-17

2. Руденко В. С. Волонтерство, як провідне явище в процесі забезпечення соціальної безпеки України: еволюція, значення, сучасний стан та проблематика / В. С. Руденко. — Київ : Крінов, 2023. — 56 с.

3. Лобуренко А. Волонтерство в Україні: виклики та перспективи розвитку / А. Лобуренко. — 10 жовтня 2024. URL:: https://volunteer.country/library/volonterstvo-v-ukrayini/

4. VolunteerMatch : міжнародний ресурс для пошуку волонтерських можливостей / [Електронний ресурс]. URL: https://www.volunteermatch.org

5. Добро.ua : українська платформа для реєстрації волонтерів та ініціатив / [Електронний ресурс]. URL: https://dobro.ua

6. LetsDoItUkraine : платформа для організації екологічних акцій / [Електронний ресурс]. URL: https://letsdoitukraine.org

7. Saura J. R. What drives volunteers to accept a digital platform that supports volunteering? / J. R. Saura. — Frontiers in Psychology, Vol. 11, Article 429 (2020). URL: https://www.frontiersin.org/articles/10.3389/fpsyg.2020.00429/full

8. Xiaofei W. Design and Development of Volunteer Management System Based on Low-Code Platform / W. Xiaofei. — International Journal of Trend in Scientific Research and Development (IJTSRD), Vol. 7, Issue 5, Sep-Oct 2023, pp. 887-892. — URL: https://www.ijtsrd.com/papers/ijtsrd60064.pdf

9. Kotelevets A. E-Volunteering as a Possibility A. Kotelevets. — 2023. URL: https://pnap.ap.edu.pl/index.php/pnap/article/view/1077

10. Mishra S., et al. Volunteer and NGO Matching Platform / S. Mishra et al., 2024. URL: https://pdfs.semanticscholar.org/58a3/6f37b8fb79f8e096d7ffad1588446d3c4aec.pdf

11. Larsen, John (2021). React Hooks in Action With Suspense and Concurrent Mode. Manning. ISBN 978-1-72004-399-7.

12. Hámori, Fenerec (2022-05-31). "The History of React.js on a Timeline". RisingStack. URL: https://blog.risingstack.com/the-history-of-react-js-on-a-timeline/

13. Filipova, Olha (2016). Learning Vue.js 2: learn how to build amazing and complex reactive web applications easily with Vue.js. ISBN 978-1-78646-113-1

14. Uzayr, Sufyan bin; Cloud, Nicholas; Ambler, Tim (November 2019). JavaScript Frameworks for Modern Web Development: The Essential Frameworks, Libraries, and Tools to Learn Right Now. Apress. ISBN 978-1484249949.

15. Rojas, Carlos (13 November 2020). Building Native Web Components: Front-End Development with Polymer and Vue.js. Apress. ISBN 978-1484259047.

16. Hands-On JavaScript High Performance: Build faster web apps using Node.js, Svelte.js, and WebAssembly. ISBN 978-1838821098.

17. PHP 7 vs Node.js? They Can Be Partners, Not Competitors For a Developer URL: https://web.archive.org/web/20170223211427/https://belitsoft.com/php-development-services/php7-vs-nodejs

18. MySQL :: MySQL 5.7 Reference Manual :: 1.3 What Is New in MySQL5.7. URL: dev.mysql.com.

19. Beck, Kent. Extreme Programming Explained: Embrace Change / Kent Beck. — Boston: Addison-Wesley, 2005. — 176 p.

20. Sutherland, Jeff, Schwaber, Ken. The Scrum Guide: The Definitive Guide to Scrum: The Rules of the Game / J. Sutherland, K. Schwaber. — 2020. — 37 p.

21. Cohn, Mike. User Stories Applied: For Agile Software Development / Mike Cohn. — Upper Saddle River, NJ: Addison-Wesley, 2004. — 228 p.

22. Poppendieck, Mary, Poppendieck, Tom. Lean Software Development: An Agile Toolkit / Mary Poppendieck, Tom Poppendieck. — Boston: Addison-Wesley, 2003. — 338 p.

23. Методичні вказівки до виконання кваліфікаційної роботи магістра для здобувачів спеціальності 121 – Інженерія програмного забезпечення, всіх форм

навчання / укладачі: Михалик Д.М., Цуприк Г.Б., Бревус В.М., Мудрик І.Я. – Тернопіль: Тернопільський національний технічний університет імені Івана Пулюя, 2024. – 44 с. (https://elartu.tntu.edu.ua/handle/lib/50316)

24. Cockburn A. Writing Effective Use Cases. — Boston: Addison-Wesley, 2000. — 270 p.

25. Jacobson I., Christerson M., Jonsson P., Övergaard G. Object-Oriented Software Engineering: A Use Case Driven Approach. — Reading, MA: Addison-Wesley, 1992. — 512 p.

26. Larman C. Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development. — 3rd ed. — Upper Saddle River: Prentice Hall, 2004. — 738 p.

27. Booch G., Rumbaugh J., Jacobson I. The Unified Modeling Language User Guide. — 2nd ed. — Boston: Addison-Wesley, 2005. — 496 p.

28. OMG (Object Management Group). *Unified Modeling Language (UML) Specification, Version 2.5.1*. — 2017. URL: https://www.omg.org/spec/UML

29. ISO/IEC 19505-1:2012. Information technology — Object Management Group Unified Modeling Language (OMG UML) — Part 1: Infrastructure.

30. ISO/IEC 19505-2:2012. Information technology — Object Management Group Unified Modeling Language (OMG UML) — Part 2: Superstructure.

31. Sommerville I. Software Engineering. — 10th ed. — Boston: Pearson, 2015.

a. 816 p.

32. Pressman R. S., Maxim B. R. Software Engineering: A Practitioner’s Approach. — 8th ed. — New York: McGraw-Hill, 2014. — 978 p.

33. Fairley R. Managing and Leading Software Projects. — Hoboken: Wiley, 2009. — 432 p.

34. ISO/IEC/IEEE 12207:2017. Systems and software engineering — Software life cycle processes.

35. Boehm B., Turner R. Management challenges for sequential and agile lifecycle models // *Computer*. — 2005. — Vol. 38, No. 8. — P. 32–40.

36. Fielding R. T. *Architectural Styles and the Design of Network-based Software Architectures* / Roy Thomas Fielding. — Doctoral dissertation, University of California, Irvine, 2000. — 180 p. (PDF). URL: https://roy.gbiv.com/pubs/dissertation/fielding_dissertation.pdf

37. Bass L., Clements P., Kazman R. *Software Architecture in Practice*. — 3rd ed. — Boston: Addison-Wesley, 2012. — 624 p. URL: https://books.google.com/books/about/Software_Architecture_in_Practice.html?id=-II73rBDXCYC&utm_source=chatgpt.com

38. Newman S. *Building Microservices*. — 2nd ed. — O'Reilly Media, 2021. — 310 p.

39. OWASP. *OWASP Top Ten: 2021 — The most critical web application security risks*. — 2021. URL: https://owasp.org/Top10/2021/

40. Microsoft Docs. *Three-Tier Architecture Overview*. — Microsoft, 2021. URL: https://learn.microsoft.com/en-us/architecture/

41. Modern web application stack: presentation, application and storage layers / Nepyivoda M. et al. *Modern Engineering and Innovative Technologies*. — 2022. — T. 1(22-01), C. 104–112.

42. Three-tier architecture for Internet of Things networks / Globa L. S., Kurdecha V. V., Shoferivskyi A. S. *Information and Telecommunication Sciences*. — 2018. — Vol. 9(2), P. 36–43.

43. lmasri R., Tamer B., Lee J. Trends in Relational Database Systems: New Architectures and Big Data // Journal of Database Management. — 2020. — Vol. 31(4). — P. 1–23.

44. Stonebraker M., Çetintemel U. "One Size Fits All": An Idea Whose Time Has Come and Gone? // Communications of the ACM. — 2015. — Vol. 59(11). — P. 70–77.

45. Gulwani S., Chandra S., Prabhakar P. Modern Approaches to SQL Optimization and Query Processing // IEEE Access. — 2021. — Vol. 9. — P. 101234–101245.

46. Osmani A. *Learning JavaScript Design Patterns: A JavaScript and jQuery Developer's Guide*. — 2nd ed. — O'Reilly Media, 2021. — 320 p.

47. ECMA International. *ECMAScript® 2022 Language Specification*. — ECMA-262, 2022. — URL: https://www.ecma-international.org/publications-and-standards/standards/ecma-262/

48. Flanagan D. *JavaScript: The Definitive Guide*. — 7th ed. — O'Reilly Media, 2020. — 1152 p.

49. Sharp J. *Visual Studio Code: End-to-End Editing and Debugging Tools for Web Developers*. — Packt Publishing, 2019. — 320 p.

50. Pradeep Singh. *Mastering Postman for API Testing: Efficient Strategies for Modern Web Services*. — Independently published, 2021. — 240 p.

51. Postman, Inc. *Postman Documentation: API Testing*. — 2025. URL: https://learning.postman.com/docs/

52. Marick B. *The Craft of Software Testing: Subsystems Testing and Automated Testing*. — 2nd ed. — Prentice Hall, 2017. — 384 p.

53. SeleniumHQ. *Selenium Documentation*. — 2025. URL: https://www.selenium.dev/documentation/

54. Test Automation University. *Web Application Testing Courses*. — 2025. URL: https://testautomationu.applitools.com/

55. JMeter Documentation. *Apache JMeter User Manual*. — 2025. URL: https://jmeter.apache.org/usermanual/index.html

56. Охорона праці : Навчальний посібник [Текст] Геврик Є.О. - Ельга: Ніка-Центр. - 2003. – 279 с.

57. Закон України «Про охорону праці» № 2694-XII. – Відомості Верховної Ради України, 1992. – № 24. – Ст. 194.

58. ДСТУ 2293-99. Охорона праці. Терміни та визначення основних понять. – Київ: Держспоживстандарт України, 1999. – 25 с.

59. ДСТУ 9327:2025. Пожежна безпека. Проєктування огороджувальних конструкцій. – Київ: Держспоживстандарт України, 2025. – 32 с.

60. ISO 45001:2018. Occupational health and safety management systems — Requirements with guidance for use. – Geneva: ISO, 2018. – 40 p.

61. ISO/IEC 27001:2022. Information technology — Security techniques — Information security management systems — Requirements. – Geneva: ISO/IEC, 2022. – 45 p.

62. Методичний посібник для здобувачів освітнього ступеня «магістр» всіх спеціальностей денної та заочної (дистанційної) форм навчання «БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ» / В.С. Стручок –Тернопіль: ФОП Паляниця В. А., –156 с. Отримано з https://elartu.tntu.edu.ua/handle/lib/39196.

63. Інформацію при написанні зазначеного підрозділу можна отримати з навчального посібника: Навчальний посібник «ТЕХНОЕКОЛОГІЯ ТА ЦИВІЛЬНА БЕЗПЕКА. ЧАСТИНА «ЦИВІЛЬНА БЕЗПЕКА»» / автор-укладач В.С. Стручок– Тернопіль: ФОП Паляниця В. А., – 156 с. Отримано з http://elartu.tntu.edu.ua/handle/lib/39424

ДОДАТКИ

ДОДАТОК А

Апробація результатів

УДК 004.41
Р. Лагола; П. Тимків, к. т. н.
(Тернопільський національний технічний університет імені Івана Пулюя, Україна)

ЗАСТОСУВАННЯ СУЧАСНИХ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ ПРАКТИКОРІЄНТОВАНОЇ WEB-ПЛАТФОРМИ «VOLUNTEER»

UDC 004.41
R. Lahola; P. Tymkiv, PhD.

OF A PRACTICALLY ORIENTED WEB PLATFORM «VOLUNTEER»

Волонтерська діяльність – це той вид, де передбачити щось вкрай важко, а отже розширення функціоналу, зміна вимог – це те, що дозволить врахувати реалії військового стану та допомогти врятувати не одне життя і наближатиме Перемогу. І сьогодні – це точно один із найбільш актуальних та пріоритетних напрямків.

Отже, метою роботи є створення web-платформи для волонтерської діяльності з врахуванням її особливостей ведення та специфіки.

Об'єктом дослідження є процес створення web-платформи орієнтованої на волонтерську діяльність з використанням оптимальних обґрунтованих засобів та ресурсів розробки при забезпечення якості.

Предметом дослідження є методи, моделі та технології проєктування та розробки web-платформи із застосуванням сучасних Frontend і Backend технологій, баз даних та хмарних сервісів. Методи дослідження включають: аналіз конкурентних платформ та існуючих рішень, моделювання архітектури системи, проєктування, розробку та тестування функціональних компонент.

В роботі я продемонстрував основні етапи життєвого циклу розробки програмного забезпечення, зокрема: процес проєктування, розробки, та тестування веб-платформи, яке реалізована як кластер із двох основних компонентів: web-сервісу для взаємодії користувачів та API для керування подіями та заявками з врахуванням особливостей роботи, переваг та недоліків існуючих сервісів типу VolunteerMatch, Добро.ua, LetsDoItUkraine.

Стосовно застосованих технологій, було обрано: ітеративно-інкрементальний процесу розробки (Agile/Scrum -підхід) для врахування частої зміни вимог, можливості аналізу та корегування проміжних етапів; базу даних реляційного типу MySQL; для клієнтської та серверної частин платформи мову програмування JavaScript. Для ефективності здійснення розробки я зупинив свій вибір на наступних інструментах: VS Code, Postman, dbdiagram.io, PlantUML / PlantText, Git / GitHub.

Таким чином, поєднання JavaScript, React, Node.js, а також реляційної бази даних та Agile-підходу для моєї розробки є оптимальним та дасть змогу забезпечити: швидку та ефективну розробку, можливість масштабування та підтримку системи в цілому, гнучкість при внесенні змін, прозорість і зручність при необхідності залучення до роботи додатково членів команди.

Література

1. Руденко В. С. Волонтерство, як провідне явище в процесі забезпечення соціальної безпеки України: еволюція, значення, сучасний стан та проблематика / В. С. Руденко. — Київ : Крінов, 2023. — 56 с.
2. Micheal Full (2020). How the Internet Works & the Web Development Process. p. 30. ISBN 979-8641657073.

ДОДАТОК Б

ФРАГМЕНТИ КОДУ ПРОГРАМИ

Лістинг Б.1 – Підключення до бази даних MySQL

Цей фрагмент демонструє підключення серверної частини платформи до реляційної бази даних MySQL для роботи з користувачами, подіями та заявками)

```
// db.js
const mysql = require('mysql2');

const db = mysql.createConnection({
    host: 'localhost',
    user: 'root',
    password: 'password',
    database: 'volunteer_events'
});

db.connect((err) => {
    if (err) {
        console.error('Помилка підключення до бази даних:', err);
        return;
    }
    console.log('Підключено до бази даних MySQL');
});

module.exports = db;
```

Лістинг Б.2 – Модель користувача

Цей клас описує основні властивості користувача системи, включаючи ім’я, електронну пошту та роль (волонтер, організатор, адміністратор

```
// models/User.js
class User {
    constructor(id, name, email, role) {
        this.id = id;
        this.name = name;
        this.email = email;
        this.role = role; // volunteer | organizer | admin
    }
}

module.exports = User;
```

Лістинг Б.3 – Реєстрація нового користувача

Тут демонструється обробка реєстрації користувача на сервері, з хешуванням пароля для безпеки.

```
// routes/auth.js
const express = require('express');
const router = express.Router();
const db = require('../db');
const bcrypt = require('bcrypt');

router.post('/register', async (req, res) => {
```

```
const { name, email, password, role } = req.body;
    const hashedPassword = await bcrypt.hash(password, 10);

    const query = `
        INSERT INTO users (name, email, password, role)
        VALUES (?, ?, ?, ?)
    `;

    db.query(query, [name, email, hashedPassword, role], (err) => {
        if (err) {
            console.error('Помилка реєстрації користувача:', err);
            res.status(500).send('Помилка сервера');
        } else {
            res.send('Користувача зареєстровано успішно');
        }
    });
});

module.exports = router;
```

Лістинг Б.4 – Створення нової події

Тут показується обробка запиту на створення події в системі та запис даних у базу.

```
// routes/events.js
const express = require('express');
const router = express.Router();
const db = require('../db');
```

```
router.post('/create', (req, res) => {
    const { title, description, date, organizer_id } = req.body;

    const query = `
        INSERT INTO events (title, description, date, organizer_id)
        VALUES (?, ?, ?, ?)
    `;

    db.query(query, [title, description, date, organizer_id], (err) => {
        if (err) {
            console.error('Помилка при створенні події:', err);
            res.status(500).send('Помилка сервера');
        } else {
            res.send('Подію успішно створено');
        }
    });
});

module.exports = router;
```

Лістинг Б.5 – Подання заявки на участь у події

Цей фрагмент *д*емонструє обробку заявки волонтера на участь у події та збереження статусу заявки у базі.

```
// routes/applications.js
const express = require('express');
const router = express.Router();
const db = require('../db');
```

```
router.post('/apply', (req, res) => {
    const { event_id, user_id } = req.body;

    const query = `
        INSERT INTO applications (event_id, user_id, status)
        VALUES (?, ?, 'pending')
    `;

    db.query(query, [event_id, user_id], (err) => {
        if (err) {
            console.error('Помилка при поданні заявки:', err);
            res.status(500).send('Помилка сервера');
        } else {
            res.send('Заявка успішно подана');
        }
    });
});

module.exports = router;
```

Лістинг Б.6 – Відправка сповіщення електроною поштою

Модуль надсилає повідомлення користувачам про зміну статусу заявки або подій.

```
// notifications/sendEmail.js
const nodemailer = require('nodemailer');

const transporter = nodemailer.createTransport({
    service: 'gmail',
```

```
auth: { user: 'volunteer.app@gmail.com', pass: 'password' }
});

function sendEmail(to, subject, text) {
    const mailOptions = { from: 'volunteer.app@gmail.com', to, subject,
text };
    transporter.sendMail(mailOptions, (error) => {
        if (error) console.error('Помилка надсилання:', error);
        else console.log('Повідомлення відправлено');
    });
}

module.exports = sendEmail;
```

Лістинг Б.7 – Клієнтський запит на отримання списку подій

Фронтенд отримує події з сервера і відображає їх у веб-інтерфейсі для користувача.

```
// frontend/js/events.js
async function loadEvents() {
    const response = await fetch('/api/events');
    const events = await response.json();

    const container = document.getElementById('eventsList');
    container.innerHTML = '';

    events.forEach(e => {
        const item = document.createElement('div');
        item.classList.add('event-card');
        item.innerHTML =
`<h3>${e.title}</h3><p>${e.description}</p><p>${e.date}</p>`;
        container.appendChild(item);
```

```
  });
}

document.addEventListener('DOMContentLoaded', loadEvents);
```

Лістинг Б.8 – SQL-схема бази даних

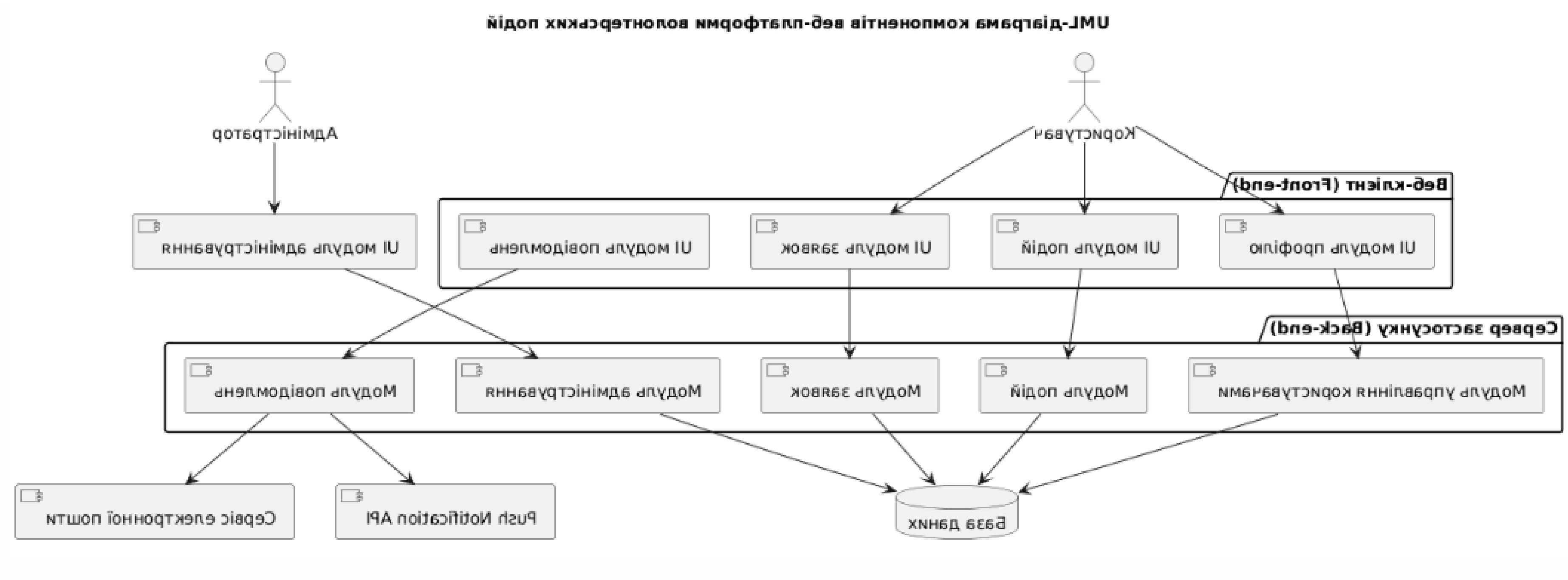
Цей фрагмент показує структуру таблиць users, events та applications для зберігання даних системи.

```
CREATE TABLE users (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100),
    email VARCHAR(100) UNIQUE,
    password VARCHAR(255),
    role ENUM('volunteer','organizer','admin')
);

CREATE TABLE events (
    id INT AUTO_INCREMENT PRIMARY KEY,
    title VARCHAR(150),
    description TEXT,
    date DATE,
    organizer_id INT,
    FOREIGN KEY (organizer_id) REFERENCES users(id)
);

CREATE TABLE applications (
    id INT AUTO_INCREMENT PRIMARY KEY,
    event_id INT,
```

```
 user_id INT,
    status ENUM('pending','approved','rejected'),
    FOREIGN KEY (event_id) REFERENCES events(id),
    FOREIGN KEY (user_id) REFERENCES users(id)
);
```


UML-діаграма компонентів веб-платформи волонтерських подій
Адміністратор
Користувач
Веб-клієнт (Front-end)
UI модуль адміністрування
UI модуль повідомлень
UI модуль заявок
UI модуль подій
UI модуль профілю
Сервер застосунку (Back-end)
Модуль повідомлень
Модуль адміністрування
Модуль заявок
Модуль подій
Модуль управління користувачами
Сервіс електронної пошти
Push Notification API
База даних