МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет
імені Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

ДОПУЩЕНО ДО ЗАХИСТУ
Завідувач кафедри інтелектуальних
інформаційних систем, д-р техн. наук, проф.
_____Ю. П. Кондратенко
«____»_____2024 р.

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

ВЕБЗАСТОСУНОК ДЛЯ ПІДТРИМКИ ВОЛОНТЕРСЬКОЇ ДІЯЛЬНОСТІ

Спеціальність 122 «Комп'ютерні науки»

122 – КРБ – 401.22230101

Виконав студент 4-го курсу, групи 401
_____ Б. О. Бєлік
«17» червня 2024 р.

Керівник: д-р фіз.-мат. наук, проф.
_____Е. А. Лисенков
«17» червня 2024 р.

Миколаїв – 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет ім. Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

Рівень вищої освіти **бакалавр**
Спеціальність **122 «Комп'ютерні науки»**
(шифр і назва)
Галузь знань **12 «Інформаційні технології»**
(шифр і назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри інтелектуальних інформаційних систем, д-р техн. наук, проф.
_____ Ю. П. Кондратенко
«____» _____2024 р.

З А В Д А Н Н Я
на виконання кваліфікаційної роботи

Видано студенту групи 401 факультету комп'ютерних наук Бєліку Богдану Олександровичу

1. Тема кваліфікаційної роботи «Вебзастосунок для підтримки волонтерської діяльності».

Керівник роботи Лисенков Едуард Анатолійович, д-р фіз.-мат. наук, проф.

Затв. наказом Ректора ЧНУ ім. Петра Могили від «28» грудня 2023 р. № 271

2. Строк представлення кваліфікаційної роботи студентом «17» червня 2024 р.

3. Вхідні (початкові) дані до роботи: експертні оцінки існуючих систем для підтримки волонтерської діяльності.

Очікуваний результат: розроблена вебсистема для підтримки волонтерської діяльності.

4. Перелік питань, що підлягають розробці (зміст пояснювальної записки):

- загальна характеристика предметної області;
- аналіз існуючих систем для підтримки волонтерської діяльності;

– створення сценаріїв використання вебзастосунку;

– аналіз та опис використаних технологій;

– проєктування вебзастосунку для підтримки волонтерської діяльності.

5. Перелік графічного матеріалу: презентація.

6. Завдання до спеціальної частини: «Захист від іонізуючих випромінювань»

7. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис
Спеціальна частина з охорони праці	Алексєєва А. О., доцент кафедри екології	

Керівник роботи __д-р фіз.-мат. наук, проф. Лисенков Е. А._____
(наук. ступінь, вчене звання, прізвище та ініціали)

_____
(підпис)

Завдання прийнято до виконання ___Бєлік Б. О._____
(прізвище та ініціали)

_____
(підпис)

Дата видачі завдання «_14_» _____січня____ 2024 р.

КАЛЕНДАРНИЙ ПЛАН
виконання кваліфікаційної роботи

Тема: Вебзастосунок для підтримки волонтерської діяльності

№	Найменування роботи	Початок	Закінчення	Примітки
1	Подання заяви на затвердження теми та керівників КРБ	10.11.2023	15.11.2023	Виконано
2	Отримання завдання на виконання КРБ	10.01.2024	15.01.2024	Виконано
3	Складання календарного плану роботи на весь період виконання КРБ	16.01.2024	30.01.2024	Виконано
4	Отримання завдання на переддипломну практику	15.04.2024	29.04.2024	Виконано
5	Проходження переддипломної практики, збір та аналіз матеріалів до КРБ	29.04.2024	11.05.2024	Виконано
6	Розробка звіту з переддипломної практики	12.05.2024	15.05.2024	Виконано
7	Виконання КРБ: аналіз систем для підтримки волонтерської діяльності, огляд існуючих технологій, розробка програмного забезпечення	13.05.2024	22.06.2024	Виконано
8	Перший попередній захист КРБ на засіданні комісії кафедри	27.05.2024	27.05.2024	Виконано
9	Доробка та остаточне оформлення КРБ	28.05.2024	09.06.2024	Виконано
10	Другий попередній захист КРБ на засіданні комісії кафедри	10.06.2024	10.06.2024	Виконано
11	Подання КРБ рецензенту	13.06.2024	13.06.2024	Виконано
11	Подання КРБ, її електронної копії та інших документів (відгуку, рецензії) до захисту	17.06.2024	21.06.2024	Виконано
12	Захист БКР перед екзаменаційною комісією (ЕК)	24.06.2024	24.06.2024	Виконано

Розробив студент __Бєлік Б. О._____ _____
(прізвище, ім'я, по батькові студента) (*підпис*)

Керівник роботи __ д-р фіз.-мат. наук, проф. Лисенков Е. А. _____
(*посада, прізвище, ім'я, по батькові)* *(підпис)*

«_29_» _____01_____ 2024 р.

АНОТАЦІЯ

кваліфікаційної роботи студента групи 401 ЧНУ ім. Петра Могили
Бєліка Богдана Олександровича
Тема: «Вебзастосунок для підтримки волонтерської діяльності»

Актуальність теми дипломної роботи обумовлена потребою у створенні зручних та функціональних вебзастосунків для підтримки волонтерської діяльності, що дозволяють автоматизувати рутинні процеси, покращити взаємодію між волонтерами та організаторами проектів, залучити більше учасників.

Об'єктом дослідження є процес розробки вебзастосунків для підтримки волонтерської діяльності.

Предметом дослідження є методи та технології створення вебзастосунків, інструменти проектування баз даних та вебінтерфейсів.

Метою роботи є розробка вебзастосунку для підтримки волонтерської діяльності, що дозволить підвищити ефективність роботи волонтерів та організаторів волонтерських проектів.

Робота складається зі вступу, чотирьох розділів, висновків та додатків. У першому розділі проаналізовано предметну сферу та вимоги до застосунку. Другий розділ присвячений проектуванню функціональності, архітектури та бази даних. У третьому розділі описано реалізацію, інтерфейс та тестування застосунку. Четвертий розділ розглядає питання охорони праці.

Розроблений вебзастосунок пропонує зручні інструменти для реєстрації волонтерів, створення проектів, комунікації та відстеження прогресу, що сприятиме розвитку волонтерського руху в Україні.

Кваліфікаційна робота містить 80 сторінок, 15 рисунків, 1 таблицю, використано 36 використаних джерел та 1 додаток.

Ключові слова: вебзастосунок, волонтерство, база даних, вебінтерфейс, охорона праці.

ABSTRACT

for bachelor's qualification work of a student of group 401 of Petro Mohyla Black Sea National University
Bohdan Bielik

Topic: "Web application to support volunteer activities"

The relevance of the thesis is due to the need for creating convenient and functional web applications to support volunteer activities, which allow automating routine processes, improving interaction between volunteers and project organizers, and involving more participants.

The object of research is the process of developing web applications to support volunteer activities.

The subject of research is the methods and technologies for creating web applications, tools for designing databases and web interfaces.

The aim of the work is to develop a web application to support volunteer activities, which will increase the efficiency of volunteers and organizers of volunteer projects.

The work consists of an introduction, four chapters, conclusions, and appendices. The first chapter analyzes the subject area and the requirements for the application. The second chapter is devoted to the design of functionality, architecture, and database. The third chapter describes the implementation, interface, and testing of the application. The fourth chapter addresses occupational safety issues.

The developed web application offers convenient tools for registering volunteers, creating projects, communication, and tracking progress, which will contribute to the development of the volunteer movement in Ukraine.

The qualification work contains 93 pages, 15 figures, 1 table, uses 36 used sources, and 5 appendices.

Keywords: web application, volunteering, database, web interface, occupational safety.

ЗМІСТ

ВСТУП

Сучасний світ характеризується стрімким розвитком інформаційних технологій, які проникають в усі сфери життя людини. Не оминув технологічний прогрес і таку важливу галузь, як волонтерська діяльність. Все більше волонтерських організацій використовують у своїй роботі спеціалізовані вебзастосунки та інформаційні системи для підвищення ефективності комунікації, координації дій волонтерів, управління проектами тощо.

Актуальність теми дипломної роботи обумовлена тим, що на сьогодні існує потреба у створенні зручних та функціональних вебзастосунків для підтримки волонтерської діяльності. Такі застосунки дозволяють автоматизувати багато рутинних процесів, покращити взаємодію між волонтерами та керівниками волонтерських проектів, залучити більше людей до волонтерства через зручні онлайн-інструменти.

Провідні світові IT-компанії, такі як Microsoft, IBM, Google, приділяють значну увагу розробці технологічних рішень для некомерційного сектору, в тому числі для волонтерських організацій. Серед відомих фахівців у сфері розробки IT-систем для волонтерства можна відзначити Дж. Коена, А. Майорова, Н. Сміта та інших.

Метою даної роботи є розробка вебзастосунку для підтримки волонтерської діяльності, який дозволить підвищити ефективність роботи волонтерів та організаторів волонтерських проектів.

Для досягнення поставленої мети необхідно виконати наступні завдання:

1) проаналізувати предметну сферу волонтерської діяльності та вимоги до вебзастосунку;
2) виконати огляд існуючих аналогів та публікацій за темою роботи;
3) спроектувати архітектуру та функціональність вебзастосунку;
4) розробити структуру бази даних для вебзастосунку;
5) реалізувати вебзастосунок з використанням сучасних вебтехнологій;

6) протестувати роботу застосунку та проаналізувати результати.

Об'єктом дослідження є процес розробки вебзастосунків для підтримки волонтерської діяльності.

Предметом дослідження є методи та технології створення вебзастосунків, інструменти проектування баз даних та вебінтерфейсів.

При виконанні роботи використовувались такі методи, як аналіз предметної сфери, моделювання бізнес-процесів, проектування баз даних та архітектури вебзастосунків, об'єктно-орієнтоване програмування, тестування програмного забезпечення.

Практичне значення отриманих результатів полягає у створенні функціонального вебзастосунку, який може використовуватись волонтерськими організаціями для підвищення продуктивності своєї діяльності. Розроблений застосунок пропонує зручні інструменти для реєстрації волонтерів, створення волонтерських проектів, комунікації між учасниками, відстеження прогресу тощо.

У першому розділі роботи проведено аналіз предметної сфери волонтерської діяльності, розглянуто вимоги до вебзастосунку, а також виконано огляд існуючих аналогів та літературних джерел за темою. У другому розділі описано процес проектування функціональності та архітектури застосунку, розробки структури бази даних. Третій розділ присвячений особливостям програмної реалізації застосунку з використанням сучасних вебтехнологій, а також тестуванню його роботи. У спеціальній частині розглянуто питання охорони праці при роботі з комп'ютером.

1 АНАЛІЗ ПРЕДМЕТНОЇ СФЕРИ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Опис предметної сфери

Волонтерська діяльність - це добровільна, безкорислива, соціально спрямована, неприбуткова діяльність, що здійснюється волонтерами та волонтерськими організаціями шляхом надання волонтерської допомоги. Волонтерство є однією з форм благодійної діяльності та широко розповсюджене в усьому світі. Воно відіграє важливу роль у підтримці вразливих верств населення, вирішенні соціальних, екологічних, гуманітарних проблем.

Волонтерський рух має давню історію, його витоки можна простежити ще з часів античності. Проте особливого розвитку волонтерство набуло у XX столітті, зокрема після Першої та Другої світових воєн, коли виникла гостра потреба у залученні добровольців для допомоги постраждалим. У 1970-х роках волонтерство почало набувати більш організованих форм, з'явилися міжнародні волонтерські організації, такі як "Корпус миру", "Лікарі без кордонів" тощо [1].

В Україні волонтерський рух почав активно розвиватися після здобуття незалежності. Особливо помітним волонтерство стало під час Помаранчевої революції 2004 року та Революції Гідності 2013-2014 років. Українські волонтери зробили вагомий внесок у підтримку армії, допомогу постраждалим внаслідок бойових дій на Сході України, а також у боротьбу з пандемією COVID-19.

На сьогодні в Україні діє Закон "Про волонтерську діяльність", який визначає правові засади провадження волонтерської діяльності та статус волонтерів. Згідно з цим законом, волонтерська діяльність ґрунтується на принципах законності, гуманності, рівності, добровільності, безоплатності, неприбутковості [2].

Волонтерська діяльність може здійснюватись у різних сферах суспільного життя: соціальній підтримці та захисті вразливих категорій населення, охороні здоров'я, культурі, освіті, науці, спорті, охороні навколишнього природного

середовища, запобіганні надзвичайним ситуаціям та ліквідації їх наслідків, сприянні проведенню заходів національного та міжнародного значення тощо.

Організація волонтерства передбачає залучення та координацію роботи добровольців, планування та реалізацію волонтерських проектів, взаємодію з партнерськими організаціями та благодійниками. Для ефективного функціонування волонтерського руху необхідно забезпечити належну підготовку та мотивацію волонтерів, створити умови для їх роботи, налагодити комунікацію між учасниками волонтерської діяльності.

В сучасних умовах важливу роль в організації волонтерства відіграють інформаційні технології. Вони дозволяють оптимізувати процеси залучення та управління волонтерами, сприяють поширенню інформації про волонтерські проекти та ініціативи, полегшують комунікацію та координацію дій учасників волонтерського руху.

Використання спеціалізованих вебзастосунків та онлайн-платформ для підтримки волонтерської діяльності набуває все більшого поширення. Такі рішення дозволяють автоматизувати рутинні процеси, пов'язані з реєстрацією волонтерів, пошуком відповідних вакансій, розподілом завдань тощо. Вони надають інструменти для зручної комунікації між волонтерами та організаторами проектів, дозволяють відслідковувати прогрес виконання завдань, генерувати звітність.

Впровадження інформаційних технологій у сферу волонтерства сприяє підвищенню ефективності та масштабованості волонтерських проектів, дозволяє залучити більше учасників та охопити ширшу аудиторію. В той же час, розробка та використання таких технологій вимагає відповідних компетенцій від волонтерських організацій та їх технічних партнерів [3].

Отже, предметна сфера волонтерської діяльності є надзвичайно важливою та актуальною в сучасному суспільстві. Волонтерство робить вагомий внесок у вирішення соціальних, гуманітарних, екологічних проблем, сприяє розвитку

громадянського суспільства. Використання інформаційних технологій, зокрема спеціалізованих вебзастосунків, відкриває нові можливості для організації та підтримки волонтерського руху, дозволяє підвищити його ефективність та залучити більше учасників.

Проте, незважаючи на наявність певних програмних рішень для волонтерства, все ще існує потреба у створенні більш функціональних та адаптованих під потреби сфери вебзастосунків. Такі застосунки мають враховувати специфіку волонтерської діяльності, надавати зручні інструменти для керування проектами, комунікації, мотивації волонтерів тощо.

Розробка вебзастосунку для підтримки волонтерської діяльності має враховувати низку специфічних вимог та особливостей предметної сфери. Зокрема, такий застосунок повинен надавати функціональність для ефективного управління волонтерськими проектами, починаючи від етапу планування і закінчуючи звітністю та аналізом результатів.

Однією з ключових функцій вебзастосунку має бути можливість створення та публікації волонтерських вакансій. Організатори волонтерських проектів повинні мати змогу детально описувати суть роботи, вимоги до волонтерів, часові рамки та інші важливі деталі. Волонтери, в свою чергу, повинні мати можливість переглядати доступні вакансії, фільтрувати їх за різними критеріями (наприклад, за місцем проведення, сферою діяльності, необхідними навичками тощо), а також подавати заявки на участь у проектах.

Для забезпечення ефективної комунікації між організаторами та волонтерами, вебзастосунок має включати функції обміну повідомленнями, створення чатів або форумів. Це дозволить учасникам проекту оперативно вирішувати робочі питання, обмінюватися ідеями та досвідом, а також підтримувати командний дух.

Важливою складовою вебзастосунку має бути можливість управління завданнями та розподіл обов'язків між волонтерами. Організатори повинні мати змогу

створювати та призначати завдання, встановлювати терміни їх виконання, відслідковувати прогрес. Волонтери, в свою чергу, повинні мати доступ до персонального списку завдань, можливість відмічати їх виконання та надавати звіти про проведену роботу [4].

Для мотивації та заохочення волонтерів, вебзастосунок може включати елементи гейміфікації, такі як нарахування балів або відзнак за виконані завдання, створення рейтингів найактивніших учасників тощо. Це сприятиме підвищенню залученості волонтерів та стимулюватиме їх до активної участі у проектах.

Не менш важливою є можливість генерування звітності та аналітики у вебзастосунку. Організатори повинні мати змогу отримувати детальні звіти про хід виконання проектів, кількість залучених волонтерів, витрачений час та інші ключові показники. Ці дані дозволять оцінювати ефективність волонтерських ініціатив, виявляти проблемні місця та приймати обґрунтовані рішення щодо подальшого розвитку проектів.

Окрему увагу слід приділити питанням безпеки та конфіденційності у вебзастосунку. Оскільки він може містити персональні дані волонтерів та іншу чутливу інформацію, необхідно забезпечити надійний захист від несанкціонованого доступу, використовувати шифрування даних та дотримуватись вимог законодавства щодо захисту персональних даних.

Крім того, вебзастосунок має бути зручним та інтуїтивно зрозумілим у використанні, мати привабливий та сучасний дизайн. Він повинен бути адаптивним та коректно працювати на різних пристроях (комп'ютерах, планшетах, смартфонах), щоб волонтери могли взаємодіяти з ним у зручному для них форматі.

Важливо також передбачити можливість інтеграції вебзастосунку з іншими системами та сервісами, які можуть використовуватися у волонтерській діяльності. Наприклад, інтеграція з платформами для онлайн-пожертв дозволить залучати додаткові кошти на реалізацію волонтерських проектів, а інтеграція з соціальними

мережами сприятиме поширенню інформації про ініціативи та залученню нових волонтерів.

Отже, розробка вебзастосунку для підтримки волонтерської діяльності вимагає врахування широкого спектру функціональних та нефункціональних вимог, які випливають зі специфіки предметної сфери. Такий застосунок має стати ефективним інструментом для організації та координації волонтерських проектів, забезпечити зручну взаємодію між організаторами та волонтерами, сприяти залученості та мотивації учасників, а також надавати можливості для аналізу та вдосконалення волонтерських ініціатив.

Саме тому розробка спеціалізованого вебзастосунку для підтримки волонтерської діяльності є перспективним та актуальним завданням. Такий застосунок дозволить удосконалити процеси організації волонтерства, підвищити залученість та ефективність роботи волонтерів, що в свою чергу сприятиме розвитку волонтерського руху та посилить його позитивний вплив на суспільство.

1.2 Огляд наявних аналогів та публікацій

Для більш повного розуміння актуальності та доцільності створення вебзастосунку для підтримки волонтерської діяльності, необхідно розглянути існуючі аналоги та дослідити наявні публікації у цій сфері [5].

Серед наявних програмних рішень для підтримки волонтерства можна виділити кілька категорій: онлайн-платформи для пошуку волонтерських вакансій, системи управління волонтерами, мобільні застосунки для волонтерів тощо.

Однією з найбільш відомих міжнародних платформ для пошуку волонтерських можливостей є VolunteerMatch (https://www.volunteermatch.org/). Цей сервіс дозволяє волонтерам знаходити відповідні проекти та організації за різними критеріями, такими як місце розташування, сфера діяльності, необхідні навички тощо. Організації, в свою чергу, можуть розміщувати вакансії для волонтерів та керувати заявками.

Інтерфейс платформи VolunteerMatch є досить простим та інтуїтивно зрозумілим (рис. 1.1).

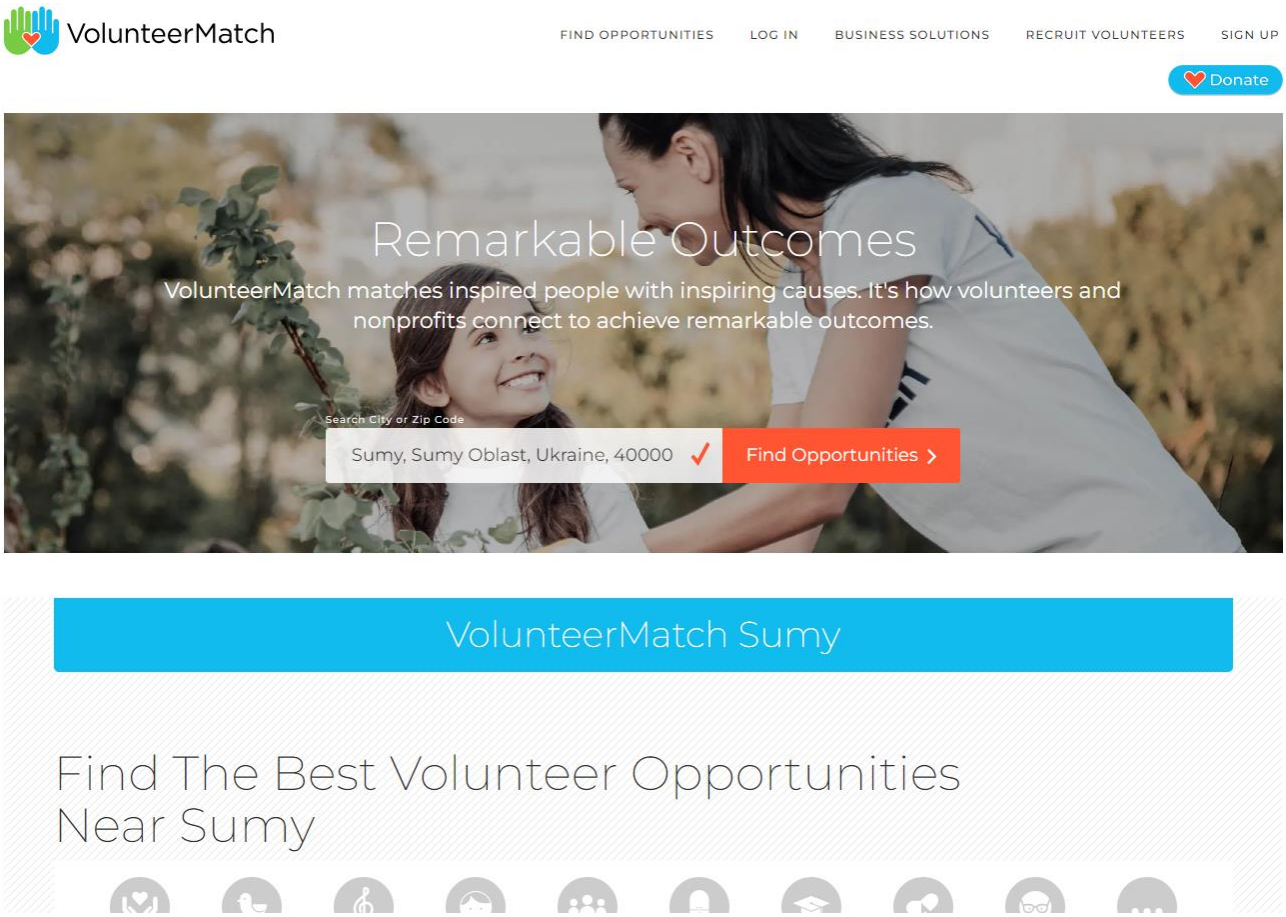

Рисунок 1.1 – Головна сторінка платформи VolunteerMatch

Ще одним популярним сервісом є Idealist (https://www.idealist.org/), який дозволяє шукати не лише волонтерські вакансії, а й оплачувану роботу та стажування у неприбуткових організаціях по всьому світу. Idealist також надає можливості для створення профілів організацій та розміщення вакансій (рис. 1.2).

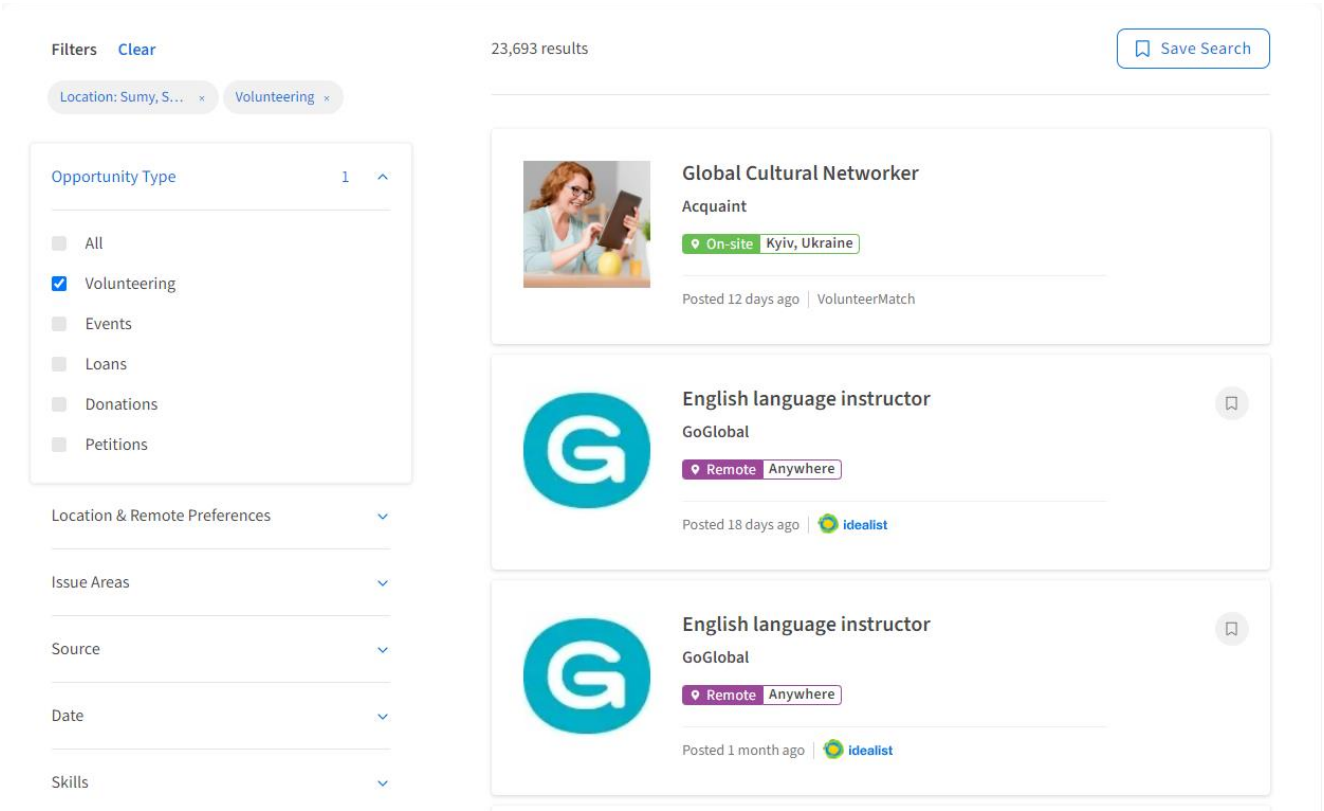

Рисунок 1.2 – Сторінка пошуку волонтерських вакансій на платформі Idealist

Окрім глобальних платформ, існують також регіональні та локальні сервіси для пошуку волонтерських можливостей. Наприклад, у США функціонує платформа AllForGood (https://www.allforgood.org/), яка агрегує волонтерські вакансії з різних джерел та дозволяє шукати їх за штатами та містами (рис. 1.3).

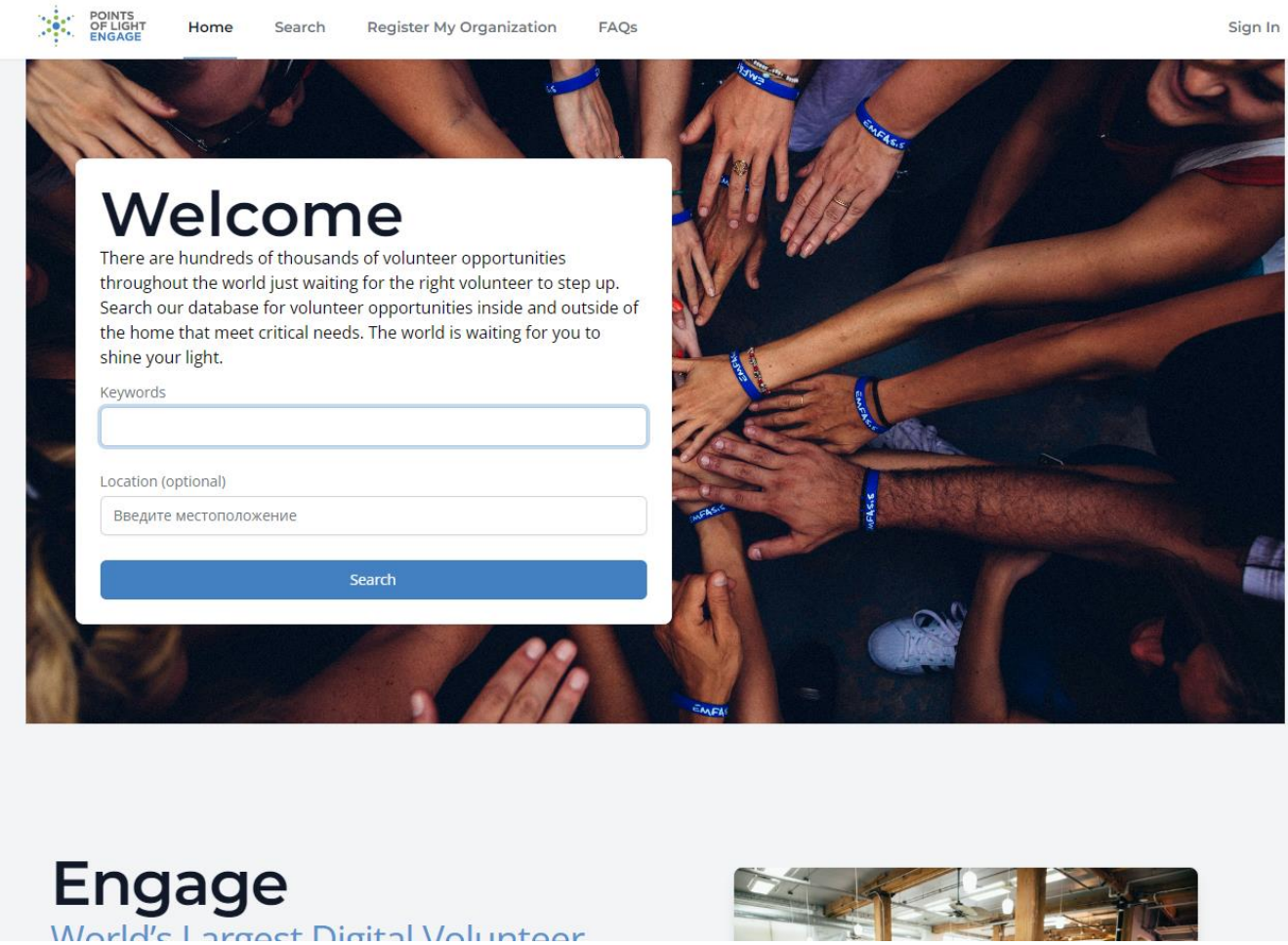

Рисунок 1.3 – Головна сторінка платформи AllForGood

Щодо систем управління волонтерами, то одним з лідерів у цій сфері є програмне забезпечення Volgistics (https://www.volgistics.com/). Це комплексне рішення для волонтерських організацій, яке дозволяє автоматизувати різні аспекти роботи з волонтерами: від набору та онбордингу до розподілу завдань та генерації звітності. Volgistics має зручний веб-інтерфейс та можливості інтеграції з іншими сервісами (рис. 1.4).

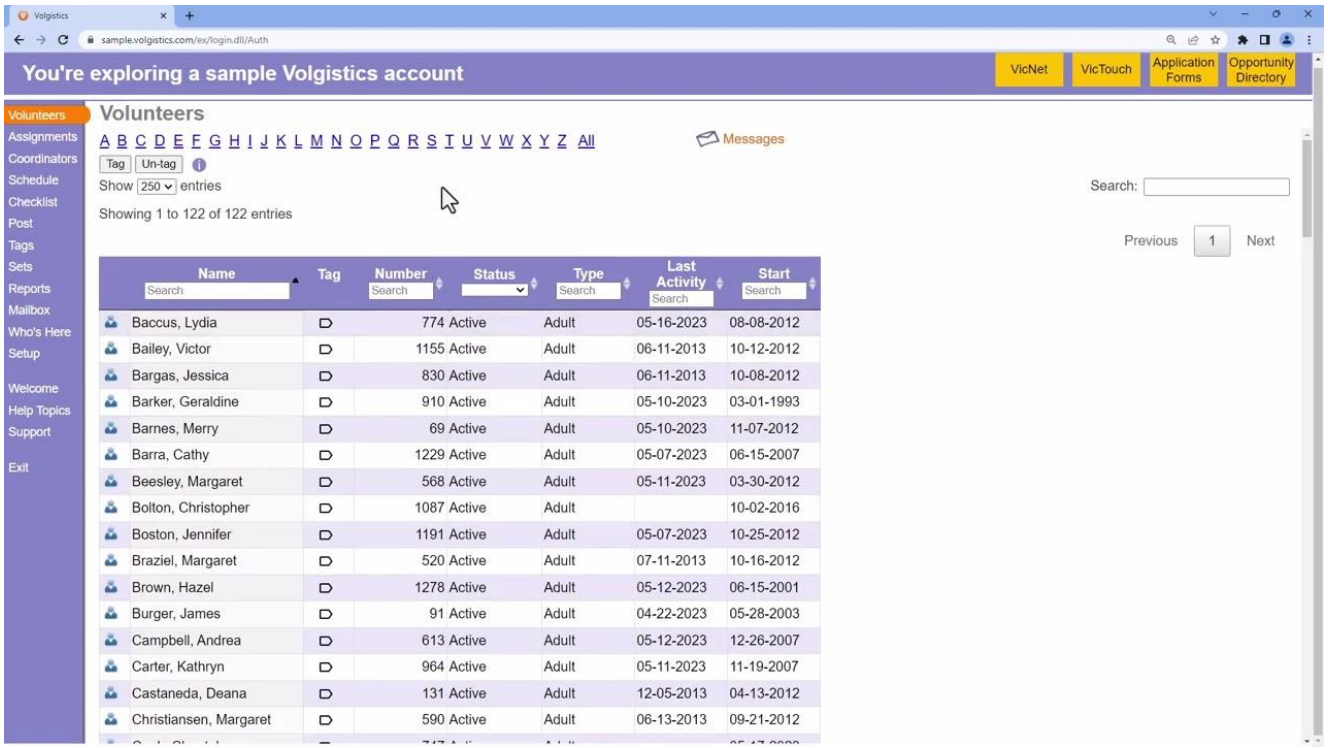

Рисунок 1.4 – Дашборд системи управління волонтерами Volgistics

Серед мобільних застосунків для волонтерів варто відзначити додаток Golden (https://www.goldenvolunteer.com/). Він дозволяє волонтерам знаходити відповідні можливості поруч, відстежувати свої години волонтерства та отримувати винагороди від партнерських організацій. Додаток Golden має привабливий дизайн та використовує гейміфікацію для мотивації волонтерів (рис. 1.5).

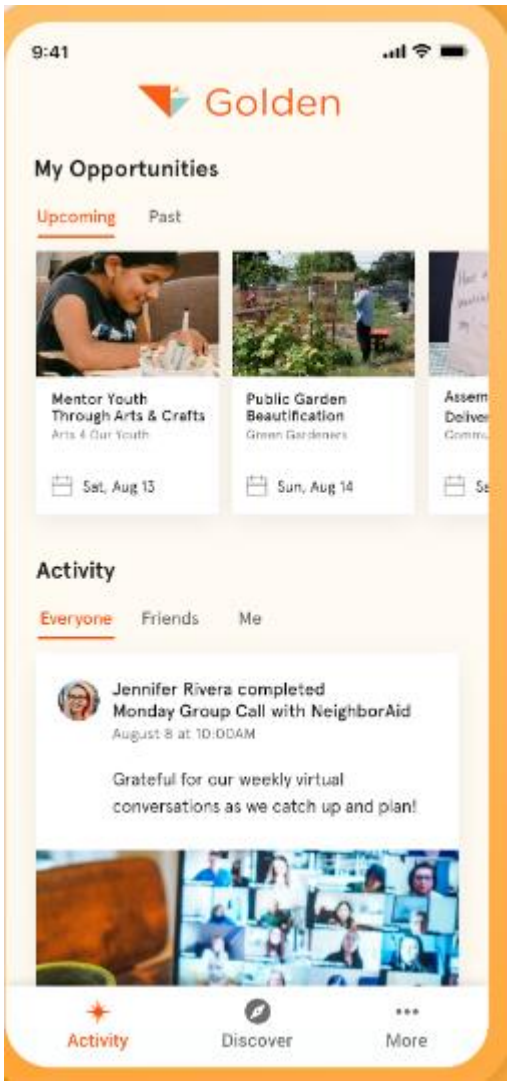

Рисунок 1.5 – Інтерфейс мобільного додатку Golden

Цікавим прикладом вебзастосунку для волонтерського менеджменту є InitLive (https://www.initlive.com/). Ця платформа орієнтована на організацію волонтерів під час проведення масштабних заходів, таких як фестивалі, конференції тощо. InitLive дозволяє здійснювати набір волонтерів, розподіляти їх по змінах та локаціях, комунікувати з ними в режимі реального часу та відстежувати їх активність. Особливістю платформи є наявність мобільного додатку для волонтерів, який дозволяє їм отримувати завдання, відмічати виконання та зв'язуватися з координаторами (рис. 1.6).

Рисунок 1.6 – Інтерфейс вебзастосунку InitLive для організаторів заходів

Таблиця 1.1 - Порівняльна характеристика аналогів вебзастосунку для підтримки волонтерської діяльності

Назва	**Пошук вакансій**	**Реєстрація волонтерів**	**Управління волонтерами**	**Мобільний додаток**	**Особливості**
VolunteerMatch	+	+	-	-	Широка база вакансій, фільтри пошуку
Idealist	+	+	-	-	Можливість пошуку оплачуваної роботи та стажувань

Закінчення таблиці 1.1

AllForGood	+	-	-	-	Агрегація вакансій з різних джерел
Volgistics	-	+	+	-	Комплексне рішення для волонтерського менеджменту
Golden	+	+	+	+	Гейміфікація, винагороди для волонтерів
Do-it	+	+	-	-	Рекомендації вакансій, інтеграція з соцмережами
InitLive	-	+	+	+	Орієнтація на масштабні заходи, управління розкладом волонтерів
Catchafire	+	+	-	-	Спеціалізація на проектах pro bono

Підсумовуючи огляд аналогів, можемо зробити висновок, що існуючі вебзастосунки та платформи покривають різні аспекти та сфери волонтерської

діяльності, пропонуючи функціональність для пошуку вакансій, набору та управління волонтерами. В той же час, більшість з них не є універсальними рішеннями та мають певні обмеження.

Зокрема, можна виділити наступні проблемні моменти:

– орієнтація на глобальний ринок та недостатня адаптація під потреби локальних волонтерських спільнот;

– відсутність комплексних рішень, які б поєднували функції пошуку вакансій, набору, онбордингу, управління та мотивації волонтерів;

– обмежена підтримка мобільних пристроїв та інтеграції з іншими сервісами та системами, які використовуються волонтерськими організаціями;

– недостатня увага до побудови волонтерських спільнот та комунікації між волонтерами.

Таким чином, незважаючи на наявність ряду програмних рішень для підтримки волонтерства, все ще існує потреба у створенні вебзастосунку, який би враховував виявлені обмеження та пропонував комплексний підхід до організації волонтерської діяльності на локальному рівні.

При розробці такого застосунку варто взяти до уваги кращі практики існуючих аналогів, зокрема:

– зручний та інтуїтивно зрозумілий інтерфейс для пошуку вакансій та створення профілів волонтерів;

– функціональність для набору, онбордингу та управління волонтерами, включаючи розподіл завдань, облік часу, комунікацію та звітність;

– гейміфікацію та інші засоби мотивації та утримання волонтерів;

– мобільний додаток для волонтерів з можливістю отримання повідомлень та завдань, відстеження активності тощо;

– інтеграцію з поширеними сервісами та системами, які використовуються волонтерськими організаціями (Google календар, Trello, Slack тощо);

– можливості для побудови локальних волонтерських спільнот, обміну досвідом та підтримки між волонтерами.

Врахування цих аспектів дозволить створити вебзастосунок, який буде корисним та затребуваним інструментом для організації та розвитку волонтерської діяльності на локальному рівні.

Окрім огляду програмних продуктів, важливо також дослідити наукові публікації, присвячені питанням розробки та використання інформаційних систем у волонтерському менеджменті [16].

Однією з ключових робіт у цій сфері є стаття "Volunteer management information systems: Challenges and opportunities for the third sector" (Voida et al., 2011), опублікована в журналі "Nonprofit Management and Leadership". Автори дослідили досвід впровадження систем управління волонтерами у різних неприбуткових організаціях та виділили основні виклики та можливості, пов'язані з їх використанням. Зокрема, вони відзначають важливість адаптації таких систем під потреби конкретних організацій, забезпечення зручності використання для волонтерів та персоналу, а також інтеграції з іншими інформаційними системами організації.

Інше дослідження "Volunteer retention through information systems: A case study" (Senseney et al., 2017), опубліковане в "Journal of Community Informatics", розглядає роль інформаційних систем у утриманні волонтерів на прикладі однієї з некомерційних організацій. Автори приходять до висновку, що використання спеціалізованої системи управління волонтерами дозволило покращити комунікацію з волонтерами, підвищити їх залученість та зменшити відтік.

Питанням гейміфікації у волонтерському менеджменті присвячена стаття "Gamification in volunteer management: A case study of online volunteer communities" (Lee et al., 2018) у журналі "International Journal of Human–Computer Interaction". Дослідники проаналізували досвід використання ігрових елементів на прикладі двох

онлайн-спільнот волонтерів та виявили, що гейміфікація може сприяти підвищенню мотивації та залученості волонтерів, особливо серед молодшої аудиторії [15].

Ще одним важливим аспектом є дослідження факторів, які впливають на готовність волонтерів використовувати інформаційні системи. Цьому питанню присвячена робота "Factors influencing volunteers' intention to use information systems: A case of a non-profit organization" (Choi & Kim, 2018) в "Journal of Computer Information Systems". Автори виявили, що на намір волонтерів користуватися системою впливають такі фактори, як очікувана користь, простота використання, сумісність з цінностями та досвідом волонтера, а також підтримка з боку організації.

Підсумовуючи огляд публікацій, можна зробити висновок, що дослідження у сфері розробки та використання інформаційних систем для волонтерського менеджменту підтверджують їх важливість та потенціал для підвищення ефективності роботи волонтерських організацій. В той же час, вони акцентують увагу на необхідності ретельного проектування таких систем з урахуванням потреб та особливостей волонтерської діяльності, забезпечення зручності та доступності для користувачів.

Таким чином, огляд існуючих програмних рішень та наукових публікацій підтверджує актуальність та доцільність розробки вебзастосунку для підтримки волонтерської діяльності. Такий застосунок має потенціал заповнити нішу комплексного рішення, адаптованого під потреби локальних волонтерських спільнот та організацій, що дозволить підвищити ефективність їх роботи та розширити можливості для залучення та утримання волонтерів.

1.3 Постановка задачі

На основі проведеного аналізу предметної сфери волонтерської діяльності та огляду існуючих програмних рішень, можемо сформулювати мету та основні вимоги до вебзастосунку, який необхідно розробити.

Метою створення вебзастосунку є підвищення ефективності та зручності організації волонтерської діяльності на локальному рівні шляхом забезпечення комплексної підтримки процесів пошуку волонтерських вакансій, набору та управління волонтерами, комунікації та мотивації учасників волонтерського руху [14].

Для досягнення поставленої мети вебзастосунок повинен вирішувати наступні задачі:

1) надавати можливість волонтерським організаціям створювати профілі та розміщувати вакансії для волонтерів з детальним описом вимог, обов'язків та умов співпраці;

2) забезпечувати зручний та ефективний пошук волонтерських вакансій для потенційних волонтерів з можливістю фільтрації за різними критеріями (місце розташування, сфера діяльності, необхідні навички тощо);

3) дозволяти волонтерам створювати персональні профілі з інформацією про свої навички, досвід, інтереси та побажання щодо волонтерської діяльності;

4) забезпечувати процес подачі заявок волонтерами на вакансії та їх обробку волонтерськими організаціями, включаючи можливість проведення онлайн-співбесід та обміну документами;

5) надавати волонтерським організаціям інструменти для управління волонтерами, зокрема, розподіл завдань, облік відпрацьованого часу, генерацію звітності тощо;

6) підтримувати ефективну комунікацію між волонтерами та організаторами, включаючи можливість відправки повідомлень, сповіщень, організації заходів та обміну інформацією;

7) сприяти мотивації та утриманню волонтерів шляхом впровадження елементів гейміфікації (нарахування балів, відзнаки, рейтинги тощо), а також можливостей для навчання та розвитку;

8) підтримувати створення та розвиток локальних волонтерських спільнот шляхом надання можливостей для спілкування, обміну досвідом, взаємної підтримки між волонтерами;

9) забезпечувати інтеграцію з іншими поширеними сервісами та системами, які використовуються волонтерськими організаціями (календарі, системи управління проектами, месенджери тощо);

10) бути доступним та зручним у використанні як через веб-інтерфейс, так і через мобільний додаток для основних платформ (iOS, Android).

Для успішної реалізації поставлених задач, вебзастосунок повинен відповідати наступним функціональним та нефункціональним вимогам:

Функціональні вимоги:

– реєстрація та авторизація користувачів з розмежуванням прав доступу (волонтери, представники організацій, адміністратори);

– створення та редагування профілів волонтерських організацій з можливістю додавання опису, контактної інформації, фото та відео матеріалів;

– розміщення волонтерських вакансій з детальним описом вимог, обов'язків, умов співпраці, термінів тощо;

– багатокритеріальний пошук вакансій з можливістю фільтрації та сортування результатів;

– створення та редагування профілів волонтерів з можливістю додавання персональної інформації, навичок, досвіду, побажань;

– подача заявок на вакансії з можливістю відстеження статусу розгляду;

– обмін повідомленнями між волонтерами та організаціями, в тому числі в рамках конкретних вакансій;

– управління завданнями та розкладом волонтерів з можливістю призначення відповідальних, встановлення термінів, відстеження прогресу виконання;

– облік відпрацьованого волонтерами часу з можливістю підтвердження зі сторони організацій;

– генерація звітності щодо волонтерської активності, кількості залучених волонтерів, відпрацьованих годин тощо;

– гейміфікація волонтерської діяльності: нарахування балів за виконання завдань, отримання відзнак, формування рейтингів волонтерів;

– можливість створення та модерації тематичних обговорень, опитувань, обміну корисною інформацією між учасниками спільноти;

– інтеграція з сервісами Google Calendar, Trello, Slack для синхронізації подій, завдань та комунікації;

– можливість одночасної роботи з веб-версією застосунку та мобільним додатком зі збереженням даних в хмарному сховищі;

– нефункціональні вимоги;

– забезпечення безпеки даних користувачів шляхом використання надійних протоколів передачі та зберігання даних, шифрування конфіденційної інформації;

– дотримання законодавчих вимог щодо захисту персональних даних (GDPR);

– забезпечення стабільної роботи застосунку при високих навантаженнях, швидке відновлення після збоїв;

– оптимізація продуктивності застосунку, забезпечення швидкого відгуку інтерфейсу користувача;

– кросплатформеність та адаптивність дизайну для коректного відображення на різних пристроях та розмірах екрану;

– підтримка основних веб-браузерів (Chrome, Firefox, Safari, Edge) та мобільних платформ (iOS, Android);

– локалізація інтерфейсу застосунку на українську та англійську мови;

– надання вичерпної документації та інструкцій користувача для ефективної роботи з застосунком;

– організація ефективної системи підтримки користувачів, обробки запитів та вирішення проблем;

– забезпечення масштабованості застосунку для можливості розширення функціональності та підтримки зростаючої кількості користувачів.

Окрім цього, при розробці вебзастосунку необхідно врахувати специфіку волонтерської діяльності та потреб користувачів. Зокрема, особливу увагу слід приділити зручності та простоті використання застосунку, оскільки серед волонтерів можуть бути люди різного віку та рівня технічної підготовки[6].

Також важливо забезпечити гнучкість налаштувань застосунку під потреби конкретних волонтерських організацій, враховуючи відмінності в масштабах, сферах діяльності та організаційних процесах.

При проектуванні та реалізації застосунку доцільно використовувати сучасні підходи та технології веброзробки, такі як адаптивний дизайн, PWA (Progressive Web Apps), мікросервісну архітектуру, контейнеризацію, хмарні обчислення тощо. Це дозволить забезпечити високу якість, продуктивність та масштабованість рішення [13].

Для успішного виконання проекту необхідно провести ретельне планування, що включає деталізацію вимог, розробку архітектури застосунку, вибір технологічного стеку, формування графіку робіт та розподіл обов'язків у команді розробників. Також важливо організувати ефективну комунікацію з потенційними користувачами (волонтерськими організаціями) для отримання зворотного зв'язку та валідації прийнятих рішень.

Очікується, що реалізація вебзастосунку з заданою функціональністю дозволить суттєво підвищити ефективність та зручність організації волонтерської діяльності,

спростити процеси пошуку та взаємодії волонтерів з організаціями, покращити комунікацію та мотивацію учасників волонтерського руху.

В результаті, це сприятиме розвитку волонтерства на локальному рівні, залученню більшої кількості людей до суспільно корисної діяльності та посиленню позитивного впливу волонтерських ініціатив на життя громад.

Таким чином, постановка задачі на розробку вебзастосунку для підтримки волонтерської діяльності є актуальною та обґрунтованою, а реалізація проекту матиме значну практичну цінність для розвитку волонтерського руху в Україні.

Висновки до розділу 1

У першому розділі дипломної роботи було проведено ґрунтовний аналіз предметної сфери волонтерської діяльності, досліджено існуючі програмні рішення для підтримки волонтерства та сформульовано постановку задачі на розробку спеціалізованого вебзастосунку.

Волонтерська діяльність є важливою складовою розвитку громадянського суспільства та вирішення соціальних, гуманітарних, екологічних проблем. Вона базується на принципах добровільності, безкорисливості та спрямована на надання допомоги тим, хто її потребує. Волонтерський рух в Україні активно розвивається та залучає все більше людей, особливо молоді.

Проведений аналіз показав, що ефективна організація волонтерської діяльності потребує використання сучасних інформаційних технологій. Вони дозволяють автоматизувати процеси пошуку та залучення волонтерів, управління волонтерськими проектами, комунікації та координації дій учасників.

Огляд існуючих програмних рішень для підтримки волонтерства виявив, що на ринку представлено ряд онлайн-платформ та систем управління волонтерами, які надають певний набір функцій, таких як пошук вакансій, реєстрація волонтерів,

розподіл завдань тощо. Серед розглянутих аналогів можна відзначити VolunteerMatch, Idealist, Volgistics, InitLive та інші.

Водночас, проведений аналіз дозволив виявити певні обмеження та недоліки існуючих рішень. Зокрема, більшість з них орієнтовані на глобальний ринок та не в повній мірі враховують потреби локальних волонтерських спільнот. Крім того, часто бракує комплексних рішень, які б поєднували різні аспекти організації волонтерської діяльності - від залучення волонтерів до управління проектами та мотивації учасників [12].

Дослідження наукових публікацій у сфері розробки та використання інформаційних систем для волонтерського менеджменту підтвердило актуальність та перспективність даного напрямку. Автори відзначають важливість адаптації таких систем під потреби конкретних організацій, забезпечення зручності використання, інтеграції з іншими сервісами. Також наголошується на потенціалі застосування ігрових механік та соціальних функцій для підвищення залученості волонтерів.

На основі проведеного аналізу було сформульовано постановку задачі на розробку вебзастосунку для підтримки волонтерської діяльності на локальному рівні. Метою створення такого застосунку є підвищення ефективності та зручності організації волонтерства шляхом забезпечення комплексної підтримки основних процесів - від пошуку вакансій до управління проектами та мотивації волонтерів.

Визначено основні задачі, які повинен вирішувати вебзастосунок, зокрема: розміщення вакансій, пошук та відбір волонтерів, управління завданнями, облік часу, комунікація учасників, генерація звітності, гейміфікація, інтеграція з іншими сервісами тощо.

Сформульовано функціональні та нефункціональні вимоги до застосунку, які враховують специфіку волонтерської діяльності та потреби користувачів. Серед них можна відзначити вимоги до безпеки даних, продуктивності, зручності використання, крос-платформності, масштабованості тощо.

Реалізація вебзастосунку з визначеною функціональністю дозволить суттєво спростити та покращити процеси організації волонтерської діяльності, залучити більше людей до волонтерства, підвищити координацію та мотивацію учасників. Це матиме позитивний вплив на розвиток волонтерського руху та посилення його ролі у вирішенні суспільно значущих проблем.

2 ПРОЕКТУВАННЯ ТА МОДЕЛЮВАННЯ ВЕБЗАСТОСУНКУ

2.1 Функціональна модель вебзастосунку

Для розробки вебзастосунку для підтримки волонтерської діяльності необхідно спочатку визначити основні функціональні вимоги та побудувати функціональну модель системи. Функціональна модель описує поведінку системи, її основні можливості та взаємодію з користувачами [7].

Основними акторами вебзастосунку є:

– волонтер - користувач, який зареєстрований у системі та може брати участь у волонтерських заходах, переглядати інформацію про доступні активності та свою історію волонтерства;

– організатор - користувач, який має права створювати нові волонтерські заходи, керувати ними та призначати волонтерів;

– адміністратор - користувач з розширеними правами, який може керувати користувачами, налаштовувати систему та здійснювати модерацію контенту.

Основні функціональні можливості вебзастосунку включають:

- реєстрацію та авторизацію користувачів;
- перегляд списку доступних волонтерських заходів;
- можливість для волонтерів записатися на участь у заході;
- створення та редагування волонтерських заходів організаторами;
- призначення волонтерів на заходи організаторами;
- перегляд історії участі у волонтерських заходах для волонтерів;
- керування користувачами та модерація контенту для адміністраторів;
- сповіщення користувачів про нові заходи та зміни у статусі їх участі.

Для моделювання функціональності вебзастосунку використано UML-діаграму варіантів використання (Use Case Diagram). Ця діаграма відображає взаємодію між

акторами (користувачами) системи та основними функціями (варіантами використання).

Діаграма варіантів використання для вебзастосунку підтримки волонтерської діяльності представлена на рис. 2.1.

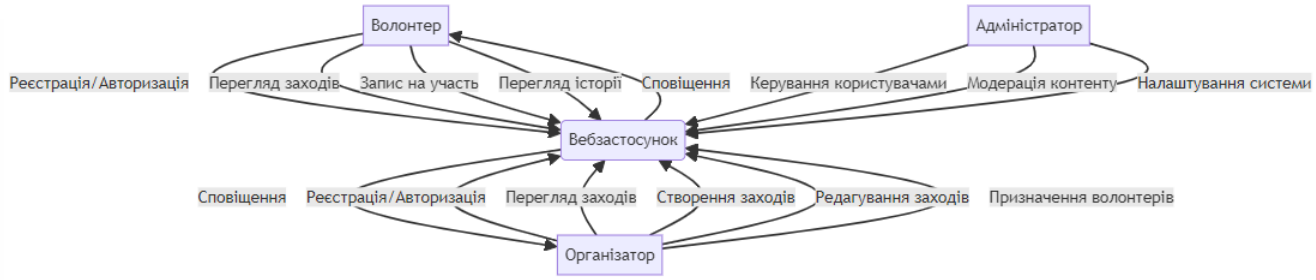

Рисунок 2.1 – Діаграма варіантів використання вебзастосунку

На діаграмі зображено три основні актори: Волонтер, Організатор та Адміністратор. Кожен з них має доступ до певного набору функцій системи.

Волонтер може реєструватися та авторизуватися в системі, переглядати список доступних волонтерських заходів, записуватися на участь у них та переглядати історію своєї волонтерської діяльності.

Організатор, окрім функцій доступних волонтеру, має можливість створювати нові волонтерські заходи, редагувати інформацію про них та призначати волонтерів для участі.

Адміністратор має повний доступ до керування користувачами (створення, редагування, видалення акаунтів), модерації контенту (видалення неприйнятних заходів або коментарів) та налаштування параметрів системи.

Окрім цього, система передбачає функцію сповіщення користувачів про нові волонтерські заходи та зміни у статусі їх участі через електронну пошту або push-повідомлення [8].

Така функціональна модель дає чітке розуміння можливостей вебзастосунку, ролей користувачів та їх взаємодії з системою. Це є основою для подальшого проектування архітектури, бази даних та користувацького інтерфейсу застосунку.

Наступним кроком буде проектування бази даних для збереження інформації про користувачів, волонтерські заходи та історію участі волонтерів, а також вибір технологій та архітектури для ефективної реалізації визначеної функціональності.

2.2 Проектування бази даних

Для забезпечення ефективного функціонування вебзастосунку для підтримки волонтерської діяльності необхідно спроектувати базу даних, яка буде зберігати всю необхідну інформацію. У цьому проекті використовується реляційна база даних PostgreSQL.

Структура бази даних складається з трьох основних таблиць: "users", "volunteering_activity" та "volunteering_application".

Таблиця "users" зберігає інформацію про користувачів системи. Вона має наступні поля:

- id (integer) - унікальний ідентифікатор користувача, первинний ключ таблиці;
- created_at (timestamp) - дата та час створення запису про користувача;
- email (varchar) - електронна адреса користувача, унікальне поле;
- enabled (boolean) - прапорець, що вказує, чи активований акаунт користувача;
- first_name (varchar) - ім'я користувача;
- last_name (varchar) - прізвище користувача;
- locked (boolean) - прапорець, що вказує, чи заблокований акаунт користувача;
- mobile (varchar) - номер мобільного телефону користувача, унікальне поле;

- password (varchar) - пароль користувача у захешованому вигляді;
- role (varchar) - роль користувача в системі (волонтер, організатор, адміністратор);
- updated_at (timestamp) - дата та час останнього оновлення запису про користувача;

Таблиця "volunteering_activity" містить інформацію про волонтерські заходи. Вона має такі поля:

- act_id (bigint) - унікальний ідентифікатор заходу, первинний ключ таблиці, генерується автоматично за допомогою послідовності "volunteering_activity_act_id_seq";
- activity_address (varchar) - адреса проведення заходу;
- activity_description (varchar) - опис заходу;
- activity_name (varchar) - назва заходу;
- activity_photo_url (varchar) - URL-адреса фотографії заходу;
- activity_time (timestamp) - дата та час проведення заходу;
- activity_type (varchar) - тип заходу (наприклад, прибирання, допомога літнім людям тощо).

Таблиця "volunteering_application" зберігає інформацію про заявки волонтерів на участь у заходах. Вона має наступні поля:

- id (bigint) - унікальний ідентифікатор заявки, первинний ключ таблиці, генерується автоматично за допомогою послідовності "volunteering_application_id_seq";
- content_type (varchar) - тип вмісту документа, доданого до заявки;
- doc_id (varchar) - ідентифікатор документа, доданого до заявки;
- file_name (varchar) - назва файлу документа, доданого до заявки;
- file_type (varchar) - тип файлу документа, доданого до заявки;
- volunteer_email (varchar) - електронна адреса волонтера;

- volunteer_address (varchar) - адреса волонтера;
- volunteer_age (integer) - вік волонтера;
- volunteer_document (oid) - документ, доданий до заявки;
- volunteer_names (varchar) - ім'я та прізвище волонтера;
- volunteer_phone_number (varchar) - номер телефону волонтера;
- volunteer_status (varchar) - статус заявки (подана, прийнята, відхилена);
- activity_id (bigint) - зовнішній ключ, що посилається на ідентифікатор заходу з таблиці "volunteering_activity";
- activity_application_fk (bigint) - зовнішній ключ, що також посилається на ідентифікатор заходу з таблиці "volunteering_activity".

Зв'язки між таблицями забезпечуються за допомогою зовнішніх ключів. Таблиця "volunteering_application" має два зовнішніх ключі (activity_id та activity_application_fk), які посилаються на первинний ключ таблиці "volunteering_activity" (act_id). Це дозволяє встановити зв'язок між заявками волонтерів та відповідними заходами.

Для автоматичної генерації первинних ключів у таблицях "volunteering_activity" та "volunteering_application" використовуються послідовності (sequences) - "volunteering_activity_act_id_seq" та "volunteering_application_id_seq" відповідно. Вони забезпечують унікальність ідентифікаторів при додаванні нових записів.

Окрім первинних та зовнішніх ключів, у таблиці "users" також визначені унікальні обмеження (UNIQUE) для полів "email" та "mobile", щоб гарантувати унікальність електронних адрес та номерів телефонів користувачів.

Така структура бази даних дозволяє ефективно зберігати та обробляти інформацію про користувачів, волонтерські заходи та заявки на участь. Вона забезпечує цілісність даних за рахунок встановлення зв'язків між таблицями та обмежень на рівні полів.

При реалізації вебзастосунку буде використано об'єктно-реляційне відображення (ORM) для взаємодії з базою даних. Це дозволить абстрагуватися від деталей конкретної СУБД і працювати з даними у вигляді об'єктів мови програмування.

2.3 Вибір технологій та архітектури вебзастосунку

Для розробки вебзастосунку для підтримки волонтерської діяльності необхідно обрати відповідні технології та архітектуру, які забезпечать ефективну реалізацію функціональності, масштабованість, безпеку та зручність у використанні.

Після аналізу вимог до системи та огляду сучасних технологій було прийнято рішення використовувати наступний стек:

1) мова програмування - Java. Java є однією з найпопулярніших мов програмування для розробки веб-застосунків. Вона має потужну екосистему, широкий вибір фреймворків та бібліотек, а також забезпечує кросплатформеність і масштабованість;

2) фреймворк - Spring Boot. Spring Boot - це фреймворк, який значно спрощує розробку веб-застосунків на Java. Він надає готові рішення для багатьох типових завдань, таких як конфігурація, створення REST API, робота з базами даних, безпека тощо. Spring Boot дозволяє швидко створювати застосунки, використовуючи принципи конвенції над конфігурацією та автоконфігурації;

3) база даних - PostgreSQL. PostgreSQL - це потужна реляційна база даних з відкритим кодом. Вона забезпечує високу продуктивність, надійність та підтримує складні запити. PostgreSQL має широкий набір функцій, включаючи транзакції, індекси, збережені процедури та тригери. Крім того, вона добре інтегрується з Java та Spring Boot;

4) ORM (Object-Relational Mapping) - Hibernate. Hibernate - це бібліотека ORM для Java, яка значно спрощує роботу з базами даних. Вона дозволяє відображати

об'єкти Java на таблиці бази даних і виконувати операції з даними за допомогою об'єктно-орієнтованого підходу. Hibernate підтримує різні бази даних, включаючи PostgreSQL, і має потужний набір функцій, таких як кешування, ледаче завантаження та оптимізація запитів;

5) веб-шар - Spring MVC. Spring MVC - це модуль Spring Framework, який надає можливості для створення веб-застосунків. Він реалізує шаблон Model-View-Controller (MVC) і забезпечує зручний спосіб обробки HTTP-запитів, маршрутизації та рендерингу HTML-сторінок. Spring MVC інтегрується з іншими компонентами Spring, такими як Spring Security для забезпечення безпеки та Spring Data для роботи з базами даних;

6) шаблонізатор - Thymeleaf. Thymeleaf - це сучасний шаблонізатор для Java, який дозволяє створювати динамічні HTML-сторінки. Він має простий синтаксис і тісну інтеграцію зі Spring MVC. Thymeleaf підтримує природний спосіб написання HTML-коду з додаванням спеціальних атрибутів для динамічного контенту;

7) клієнтська частина - HTML, CSS, JavaScript, Bootstrap. Для розробки клієнтської частини вебзастосунку буде використано стандартні веб-технології - HTML для розмітки, CSS для стилізації та JavaScript для динамічної поведінки. Крім того, для створення адаптивного та привабливого дизайну буде використано фреймворк Bootstrap, який надає готові компоненти та стилі;

8) система керування версіями - Git. Git - це розподілена система керування версіями, яка дозволяє ефективно працювати над проектом у команді. Вона забезпечує відстеження змін у коді, можливість створення гілок для паралельної розробки та злиття змін. Для розміщення віддаленого репозиторію буде використано сервіс GitHub;

9) система збірки - Maven. Maven - це інструмент для збірки та управління залежностями Java-проектів. Він дозволяє визначати залежності від зовнішніх бібліотек, виконувати компіляцію коду, запускати тести та створювати готові

артефакти (JAR, WAR файли). Maven має потужну систему плагінів та репозиторіїв, що спрощує управління проектом.

Архітектура вебзастосунку буде базуватися на принципах мікросервісів. Це означає, що система буде розділена на незалежні модулі (сервіси), кожен з яких відповідає за певну функціональність. Наприклад, окремі сервіси можуть відповідати за управління користувачами, волонтерськими заходами та заявками на участь.

Мікросервісна архітектура має наступні переваги:

- незалежність розробки та розгортання сервісів;
- можливість масштабування окремих сервісів залежно від навантаження;
- гнучкість у виборі технологій для кожного сервісу;
- спрощення тестування та підтримки завдяки модульності;
- підвищення відмовостійкості системи в цілому.

Для забезпечення комунікації між мікросервісами буде використано протокол HTTP і формат обміну даними JSON. Для реалізації API мікросервісів буде застосовано архітектурний стиль REST (Representational State Transfer).

Розгортання вебзастосунку буде виконано за допомогою контейнеризації з використанням Docker. Docker дозволяє створювати ізольовані контейнери з усіма необхідними залежностями для кожного мікросервісу. Це спрощує процес розгортання та масштабування застосунку в різних середовищах.

Для забезпечення безпеки вебзастосунку будуть реалізовані наступні заходи:

- автентифікація користувачів за допомогою Spring Security та JWT (JSON Web Tokens);
- авторизація на основі ролей користувачів (волонтер, організатор, адміністратор);
- шифрування паролів користувачів за допомогою надійних алгоритмів (наприклад, bcrypt);

– захист від поширених веб-атак, таких як SQL-ін'єкції та міжсайтовий скриптинг (XSS);

– використання HTTPS для шифрування даних під час передачі між клієнтом і сервером.

Для забезпечення якості коду та зменшення кількості помилок буде налаштовано автоматизовані тести (модульні, інтеграційні, системні) з використанням бібліотек JUnit та Mockito. Також буде налаштовано систему неперервної інтеграції та доставки (CI/CD) для автоматизації процесу збірки, тестування та розгортання застосунку.

Така комбінація технологій та архітектурних рішень дозволить створити надійний, масштабований та безпечний вебзастосунок для підтримки волонтерської діяльності. Використання сучасних фреймворків та інструментів значно спростить процес розробки та забезпечить високу якість кінцевого продукту.

Наступним етапом буде безпосередня реалізація вебзастосунку згідно з обраними технологіями та архітектурою, а також його тестування та аналіз результатів.

Висновки до розділу 2

У другому розділі дипломної роботи було проведено проектування та моделювання вебзастосунку для підтримки волонтерської діяльності. Цей етап є надзвичайно важливим, оскільки від якості проектування залежить успішність реалізації та ефективність роботи майбутньої системи.

Першим кроком було створення функціональної моделі вебзастосунку. Для цього було визначено основних акторів системи (волонтер, організатор та адміністратор) та їх взаємодію з системою. За допомогою діаграми варіантів використання (Use Case Diagram) було візуалізовано функціональні вимоги до

системи та показано, які дії може виконувати кожен з акторів. Це дало чітке розуміння меж системи та послужило основою для подальшого проектування.

Наступним важливим етапом було проектування бази даних. Для вебзастосунку було обрано реляційну базу даних PostgreSQL, яка забезпечує надійне зберігання та ефективну обробку даних. Було розроблено структуру бази даних, що складається з трьох основних таблиць: "users" для зберігання інформації про користувачів, "volunteering_activity" для зберігання даних про волонтерські заходи та "volunteering_application" для зберігання заявок волонтерів на участь у заходах. Для забезпечення цілісності даних було встановлено зв'язки між таблицями за допомогою зовнішніх ключів. Також було визначено обмеження на рівні полів, такі як унікальність електронних адрес та номерів телефонів користувачів.

Після проектування бази даних було проведено ретельний аналіз та вибір технологій і архітектури для розробки вебзастосунку. Основою стеку технологій стала мова програмування Java та фреймворк Spring Boot, які забезпечують потужні можливості для розробки веб-застосунків. Для роботи з базою даних було обрано бібліотеку Hibernate, що реалізує об'єктно-реляційне відображення (ORM) та спрощує взаємодію з базою даних. У якості шаблонізатора для створення динамічних HTML-сторінок було обрано Thymeleaf, який має зручний синтаксис та інтегрується зі Spring MVC. Для розробки клієнтської частини було вирішено використовувати стандартні веб-технології (HTML, CSS, JavaScript) та фреймворк Bootstrap для створення адаптивного дизайну.

Архітектура вебзастосунку базується на принципах мікросервісів, що дозволяє розділити систему на незалежні модулі, кожен з яких відповідає за певну функціональність. Це забезпечує гнучкість, масштабованість та можливість незалежної розробки і розгортання окремих сервісів. Для комунікації між мікросервісами було обрано протокол HTTP та формат обміну даними JSON, а для реалізації API - архітектурний стиль REST.

Особливу увагу було приділено питанням безпеки вебзастосунку. Було передбачено використання автентифікації користувачів за допомогою Spring Security та JWT, авторизацію на основі ролей, шифрування паролів, захист від поширених веб-атак та використання HTTPS для шифрування даних під час передачі.

Для забезпечення якості коду та мінімізації помилок було заплановано розробку автоматизованих тестів (модульних, інтеграційних, системних) з використанням бібліотек JUnit та Mockito. Також було вирішено налаштувати систему неперервної інтеграції та доставки (CI/CD) для автоматизації процесу збірки, тестування та розгортання вебзастосунку.

Таким чином, у другому розділі було проведено ґрунтовну роботу з проектування та моделювання вебзастосунку для підтримки волонтерської діяльності. Було створено функціональну модель, спроектовано базу даних та обрано відповідні технології та архітектуру для ефективної реалізації системи. Це закладає міцний фундамент для подальшої розробки та забезпечує чітке розуміння структури та принципів роботи майбутнього вебзастосунку.

Наступним кроком буде безпосередня реалізація спроектованої системи з використанням обраних технологій та архітектури. Це включатиме створення серверної та клієнтської частин вебзастосунку, інтеграцію з базою даних, реалізацію функціональності для різних акторів системи, забезпечення безпеки та розробку автоматизованих тестів. Після завершення реалізації буде проведено ретельне тестування та аналіз результатів роботи вебзастосунку.

Проведене проектування та моделювання є надійною основою для успішної реалізації вебзастосунку для підтримки волонтерської діяльності, що відповідатиме всім визначеним вимогам та забезпечить ефективну взаємодію волонтерів, організаторів та адміністраторів у рамках єдиної системи.

3 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ВЕБЗАСТОСУНКУ

3.1 Особливості програмної реалізації

Вебзастосунок для підтримки волонтерської діяльності було розроблено з використанням фреймворку Spring Boot та мови програмування Java. Spring Boot надає потужний набір інструментів та функціональних можливостей для швидкого та ефективного створення вебзастосунків.

Архітектура застосунку побудована на основі шаблону проектування Model-View-Controller (MVC). Цей шаблон дозволяє розділити логіку застосунку на три основні компоненти:

- model (модель): представляє дані та бізнес-логіку застосунку. У цьому проекті моделі включають класи User, VolunteeringActivity та VolunteeringApplication, які відповідають таблицям бази даних;
- view (представлення): відповідає за відображення даних користувачеві. У проекті використовується шаблонізатор Thymeleaf для створення динамічних HTML-сторінок;
- controller (контролер): обробляє запити користувача, взаємодіє з моделями та оновлює представлення. Контролери в проекті включають UserController, VolunteeringActivityController, VolunteeringApplicationController, AuthController та інші.

Для роботи з базою даних використовується PostgreSQL - потужна реляційна база даних з відкритим кодом. Spring Boot надає вбудовану підтримку для роботи з базами даних за допомогою Spring Data JPA. JPA (Java Persistence API) - це специфікація для об'єктно-реляційного відображення (ORM), яка дозволяє зручно працювати з базою даних, використовуючи об'єктно-орієнтований підхід.

У проекті визначено репозиторії (repositories), які відповідають за взаємодію з базою даних. Наприклад, UserRepo, VolunteeringApplicationRepository та інші. Ці

репозиторії розширюють інтерфейс JpaRepository, який надає набір готових методів для виконання типових операцій з базою даних, таких як збереження, оновлення, видалення та пошук даних.

Для реалізації бізнес-логіки застосунку використовуються сервіси (services). Сервіси інкапсулюють логіку обробки даних та взаємодіють з репозиторіями для виконання операцій з базою даних. Наприклад, UserService, VolunteeringActivityService, VolunteeringApplicationService та інші. Сервіси також можуть містити додаткову логіку, таку як валідація даних, обробка помилок та інші специфічні операції.

Для забезпечення безпеки застосунку використовується Spring Security. Spring Security надає потужний набір інструментів для автентифікації та авторизації користувачів. У проекті налаштовано автентифікацію користувачів за допомогою електронної пошти та пароля. Паролі користувачів зберігаються у захешованому вигляді з використанням алгоритму BCrypt.

Авторизація користувачів здійснюється на основі ролей. У проекті визначено три ролі: USER, ADMIN та ORGANIZER. Кожна роль має відповідні права доступу до певних функцій застосунку. Наприклад, користувачі з роллю USER можуть переглядати список волонтерських заходів та подавати заявки на участь, тоді як користувачі з роллю ADMIN мають доступ до адміністративних функцій, таких як керування користувачами та модерація контенту.

Для зручності користувачів реалізовано функціонал завантаження та скачування файлів. Користувачі можуть завантажувати документи у форматі PDF або зображення при подачі заявки на участь у волонтерському заході. Ці файли зберігаються в базі даних у вигляді двійкових даних (BLOB). При скачуванні файлу, він передається користувачеві з відповідним типом контенту (Content-Type) та заголовком Content-Disposition для вказівки імені файлу.

Для зручності навігації та відображення даних у вебзастосунку використовується пагінація. Це дозволяє розділити великі списки даних (наприклад, список волонтерських заходів або заявок) на менші сторінки. За допомогою параметрів запиту користувач може переходити між сторінками та сортувати дані за певними полями.

Взаємодія між клієнтською та серверною частинами застосунку здійснюється за допомогою HTTP-запитів. Контролери обробляють GET та POST запити, отримують дані від користувача, викликають відповідні методи сервісів та повертають результат у вигляді HTML-сторінок або перенаправлень.

Для забезпечення зручності розробки та тестування використовується система збірки Maven. Maven дозволяє управляти залежностями проекту, компілювати код, запускати тести та створювати готовий артефакт (WAR або JAR файл) для розгортання на сервері.

Тестування застосунку здійснюється за допомогою фреймворку JUnit. JUnit дозволяє писати модульні тести для перевірки коректності роботи окремих компонентів застосунку, таких як контролери, сервіси та репозиторії. Тести допомагають виявити потенційні помилки та забезпечити стабільність роботи застосунку.

Для забезпечення безпеки та конфіденційності даних використовується протокол HTTPS. HTTPS забезпечує шифрування даних при передачі між клієнтом та сервером, що захищає від перехоплення та несанкціонованого доступу до чутливої інформації, такої як паролі та особисті дані користувачів.

Налаштування застосунку здійснюється за допомогою файлу application.properties. У цьому файлі вказуються параметри підключення до бази даних, порт, на якому запускається застосунок, та інші налаштування, специфічні для певного середовища розгортання.

Розгортання застосунку може здійснюватися на різних платформах, таких як локальний сервер, хмарні платформи (наприклад, AWS або Heroku) або контейнерні середовища (наприклад, Docker). Spring Boot надає зручні механізми для створення готового артефакту (WAR або JAR файлу), який можна розгорнути на відповідному сервері.

Для створення інтерфейсу користувача використовується шаблонізатор Thymeleaf. Thymeleaf дозволяє легко інтегрувати дані, отримані з серверної частини, в HTML-шаблони. В проекті створено ряд HTML-сторінок, які відображають різні функціональні можливості застосунку.

Сторінка "userApplication.html" призначена для перегляду списку заявок користувачів. На цій сторінці адміністратор може бачити детальну інформацію про кожну заявку, таку як імена волонтерів, їх вік, email, номер телефону, адресу, назву активності, статус заявки та ім'я файлу, який було завантажено при поданні заявки. Також реалізовано можливість сортування та пагінації списку заявок для зручної навігації.

Сторінка "userApplicationForm.html" містить форму для подання заявки волонтером. Користувач може вибрати активність зі списку доступних активностей, ввести свої персональні дані (імена, вік, email, номер телефону, адресу) та завантажити файл. Після заповнення форми та натискання кнопки "Подати заявку" дані відправляються на сервер для подальшої обробки.

На сторінці "volunteering-activities.html" відображається список доступних волонтерських активностей. Для кожної активності показується її тип, опис, час проведення, адреса та URL фотографії. Користувач може подати заявку на участь у певній активності, натиснувши кнопку "Подати заявку".

Сторінка "updateActivitys.html" дозволяє адміністратору оновлювати інформацію про існуючі волонтерські активності. Адміністратор може змінити назву

активності, тип, опис, адресу та URL фотографії. Після внесення змін та натискання кнопки "Оновити" дані зберігаються в базі даних.

Сторінка "updateApplication.html" призначена для оновлення інформації про подану заявку. Адміністратор може змінити назву активності, імена волонтерів, вік, email, номер телефону, адресу та статус заявки (прийнято або відхилено). Також є можливість завантажити новий файл. Після оновлення даних заявка зберігається зі внесеними змінами.

Клієнтська частина застосунку взаємодіє з серверною частиною за допомогою HTTP-запитів. При відправленні форм або виконанні певних дій на сторінках, дані передаються на відповідні контролери, які обробляють запити та повертають результат. Наприклад, при поданні заявки дані форми передаються на контролер, який зберігає інформацію про заявку в базі даних та відображає повідомлення про успішне подання заявки.

Для забезпечення зручності та швидкості роботи користувачів з великими обсягами даних реалізовано пагінацію та сортування. На сторінках зі списками заявок чи активностей дані розбиваються на сторінки, що дозволяє швидко завантажувати та відображати лише необхідну кількість елементів. Користувач може перемикатися між сторінками та сортувати дані за різними полями (наприклад, за назвою активності або статусом заявки).

В цілому, програмна реалізація інтерфейсу користувача та взаємодії з ним забезпечує зручність та ефективність роботи з вебзастосунком. Використання шаблонізатора Thymeleaf дозволяє створювати динамічні та інтерактивні сторінки, які відображають актуальні дані з серверної частини. Реалізація функціональності подання заявок, перегляду та оновлення інформації про активності та заявки, а також пагінація та сортування даних роблять застосунок зручним та інтуїтивно зрозумілим для користувачів.

Навігація по сторінках застосунку здійснюється за допомогою меню, яке розташоване у шапці сторінки. Меню містить посилання на основні розділи застосунку, такі як "Головна", "Про нас", "Контакти", "Переглянути заявки" та "Переглянути активності". Це дозволяє користувачам легко переміщуватися між різними сторінками та отримувати доступ до потрібної інформації.

Для розмежування доступу та відображення відповідного контенту залежно від ролі користувача використовується бібліотека Spring Security. У шаблонах Thymeleaf застосовуються спеціальні атрибути та вирази, такі як "sec:authorize" та "sec:authentication", які дозволяють динамічно відображати або приховувати певні елементи інтерфейсу залежно від наявності відповідних прав доступу.

Наприклад, на головній сторінці ("homepage.html") відображається різний текст для анонімних користувачів, авторизованих користувачів та адміністраторів. Анонімним користувачам доступні кнопки "Увійти" та "Зареєструватися", тоді як авторизовані користувачі бачать своє ім'я та роль, а також мають можливість перейти до панелі користувача або адміністратора залежно від їхньої ролі.

У шаблоні "main-page.html", який є основним шаблоном для сторінок адміністратора, в меню відображаються додаткові пункти, такі як "Переглянути заявки" та "Переглянути активності". Ці пункти доступні лише для користувачів з роллю адміністратора. Крім того, в шапці сторінки відображається ім'я поточного користувача та кнопка "Вийти" для можливості завершення сеансу.

Для відображення списку активностей на сторінці "list_activitie.html" використовується таблиця, яка містить інформацію про кожну активність, таку як назва, тип, опис та адреса. Адміністратор має можливість редагувати або видаляти активності за допомогою відповідних кнопок "Оновити" та "Видалити". Також реалізовано можливість сортування списку активностей за назвою.

Аналогічно, на сторінці "list_application.html" відображається список заявок від волонтерів. Кожна заявка містить інформацію про волонтера, таку як імена, вік, email,

номер телефону, адресу, назву активності та статус заявки. Адміністратор може завантажити файл, прикріплений до заявки, а також оновити або видалити заявку за допомогою відповідних кнопок. Реалізовано можливість сортування списку заявок за різними полями.

Для зручності навігації по великій кількості записів у списках активностей та заявок використовується пагінація. Це дозволяє розбивати дані на сторінки та відображати лише обмежену кількість записів на кожній сторінці. Користувач може переміщуватися між сторінками за допомогою кнопок "Наступна" та "Остання".

Таким чином, програмна реалізація навігації та розмежування доступу забезпечує зручність використання застосунку та захищає дані від неавторизованого доступу. Використання шаблонізатора Thymeleaf та бібліотеки Spring Security дозволяє динамічно відображати відповідний контент залежно від ролі користувача та наявності необхідних прав доступу.

Крім того, в застосунку реалізовано функціональність для взаємодії з користувачем через форми. Наприклад, на сторінці "userApplicationForm.html" користувач може заповнити форму заявки на участь у волонтерській активності. Форма містить поля для введення імен волонтерів, віку, email, номера телефону та адреси. Також користувач може вибрати активність зі списку доступних активностей та завантажити файл. Після заповнення форми та натискання кнопки "Подати заявку" дані відправляються на сервер для подальшої обробки.

Для оновлення інформації про активність використовується форма на сторінці "updateActivitys.html". Адміністратор може змінити назву активності, тип, опис та адресу. Після внесення змін та натискання кнопки "Оновити" дані зберігаються в базі даних.

Для забезпечення безпеки та обмеження доступу до певних частин застосунку реалізовано систему входу та реєстрації користувачів. На сторінці входу ("login.html") користувач може ввести свій email та пароль для автентифікації. Якщо введені дані

коректні, користувач отримує доступ до відповідних розділів застосунку залежно від своєї ролі (користувач або адміністратор). У разі невірних даних відображається повідомлення про помилку.

Для нових користувачів передбачено можливість реєстрації на сторінці "register.html". Користувач може ввести своє ім'я, прізвище, номер мобільного телефону, email та пароль. Після успішної реєстрації користувач може увійти до застосунку, використовуючи свої облікові дані.

Після успішної автентифікації користувач потрапляє на відповідну панель залежно від своєї ролі. Для користувачів з роллю "USER" доступна панель користувача ("dashboard.html"), де відображається ім'я користувача та його роль. З панелі користувач може перейти на головну сторінку або вийти з облікового запису.

Для адміністраторів з роллю "ADMIN" доступна панель адміністратора ("dashboard.html"), яка містить додаткові функції та можливості керування застосунком. Адміністратор також бачить своє ім'я та роль, а також має можливість перейти на головну сторінку або вийти з облікового запису.

Однією з важливих функцій адміністратора є можливість створення нових волонтерських активностей. На сторінці "createActivitys.html" адміністратор може ввести інформацію про активність, таку як назва, тип, опис, адреса та URL-адресу фотографії. Після заповнення форми та натискання кнопки "Зареєструвати" активність зберігається в базі даних і стає доступною для перегляду та участі волонтерів.

Таким чином, програмна реалізація авторизації та аутентифікації користувачів забезпечує безпеку та конфіденційність даних у вебзастосунку. Розмежування доступу на основі ролей дозволяє надавати відповідні права та функціональність різним типам користувачів. Адміністратори мають розширені можливості керування застосунком, такі як створення нових активностей, тоді як звичайні користувачі можуть переглядати доступні активності та подавати заявки на участь.

Крім того, варто відзначити, що в застосунку використовуються різні способи передачі даних між клієнтською та серверною частинами. Наприклад, при реєстрації користувача дані з форми передаються на сервер за допомогою POST-запиту. При цьому застосовується шаблон Thymeleaf для динамічної генерації HTML-сторінок на основі даних, отриманих від сервера.

Для забезпечення безпеки при передачі даних використовується протокол HTTPS, який шифрує дані під час передачі між клієнтом та сервером. Це захищає конфіденційну інформацію, таку як паролі та персональні дані користувачів, від перехоплення та несанкціонованого доступу.

Таким чином, особливості програмної реалізації вебзастосунку для підтримки волонтерської діяльності включають використання фреймворку Spring Boot, архітектурного шаблону MVC, роботу з базою даних PostgreSQL за допомогою Spring Data JPA, реалізацію автентифікації та авторизації користувачів за допомогою Spring Security, функціонал завантаження та скачування файлів, пагінацію даних, взаємодію через HTTP-запити, тестування за допомогою JUnit, забезпечення безпеки за допомогою HTTPS та гнучкість розгортання на різних платформах.

3.2 Інтерфейс користувача

Вебзастосунок для підтримки волонтерської діяльності має зручний та інтуїтивно зрозумілий інтерфейс користувача, який забезпечує легку навігацію та доступ до всіх необхідних функцій. Інтерфейс розроблено з використанням сучасних веб-технологій, таких як HTML, CSS та JavaScript, а також фреймворку Bootstrap для створення адаптивного та привабливого дизайну.

Головна сторінка застосунку (рис. 3.1) надає користувачеві загальну інформацію про волонтерську діяльність та можливості, які надає платформа. На головній сторінці розміщено заклик до дії, який спонукає користувачів приєднатися до волонтерського руху та зареєструватися на платформі.

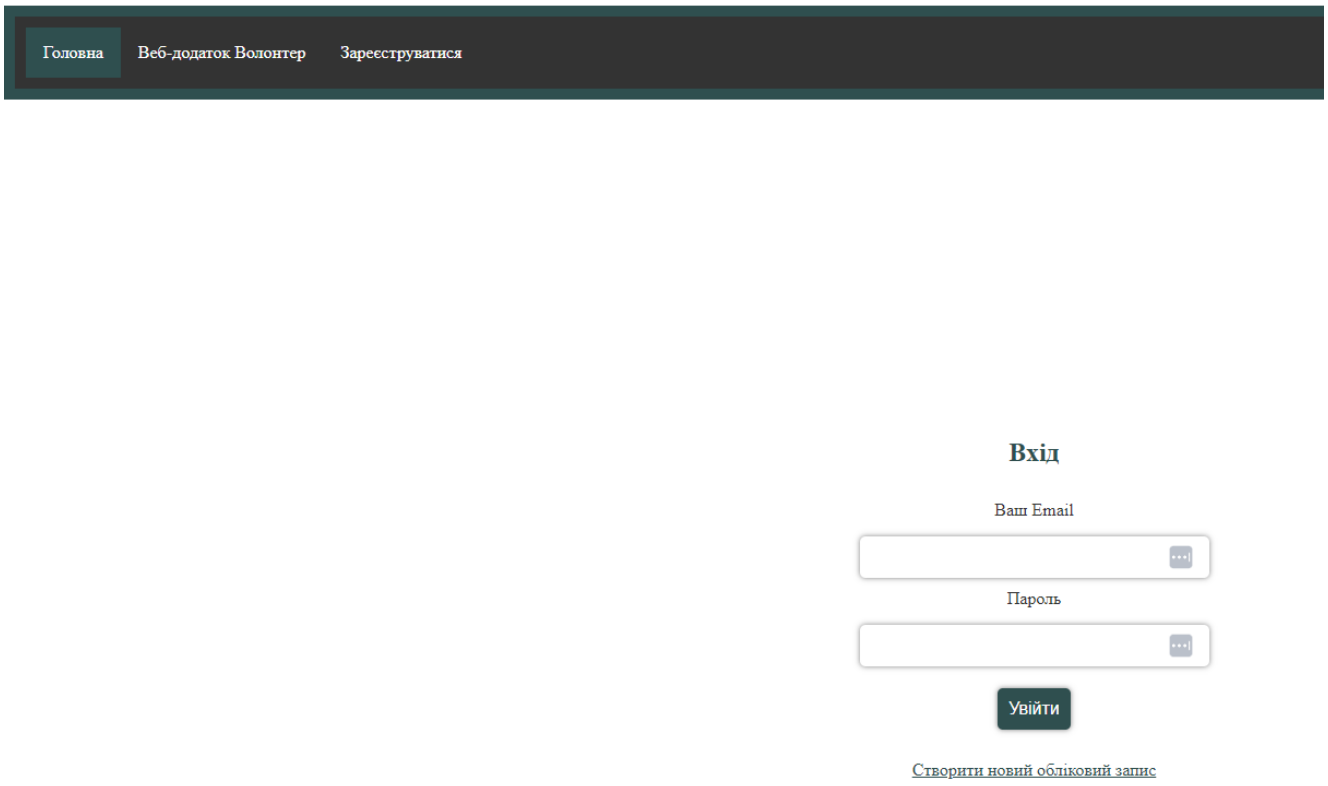

Рисунок 3.1 – Головна сторінка вебзастосунку

Для того, щоб користувач міг скористатися можливостями платформи, він повинен авторизуватися або зареєструватися. Сторінка авторизації (рис. 3.2) містить форму входу, де користувач може ввести свою електронну пошту та пароль. Якщо користувач ще не має акаунту, він може перейти на сторінку реєстрації (рис. 3.3), де потрібно заповнити відповідну форму з особистими даними, такими як ім'я, прізвище, електронна пошта, пароль та номер телефону.

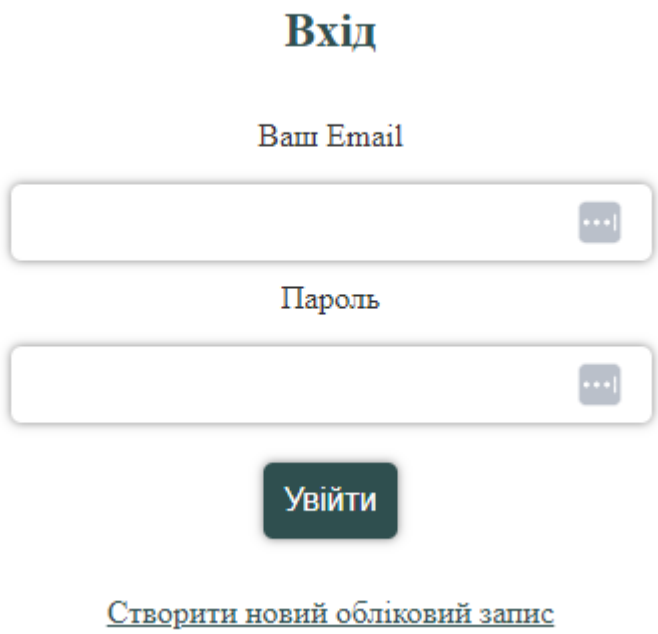

Рисунок 3.2 – Сторінка авторизації користувача

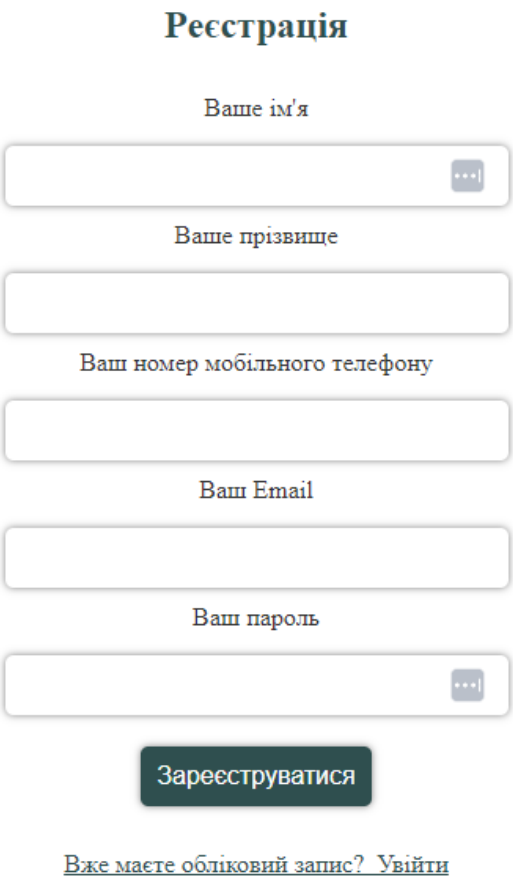

Рисунок 3.3 – Сторінка реєстрації нового користувача

Після успішної авторизації користувач потрапляє на свою особисту сторінку (дашборд) (рис. 3.4). На цій сторінці відображається персональна інформація користувача, а також доступні йому функції відповідно до його ролі (волонтер, організатор або адміністратор).

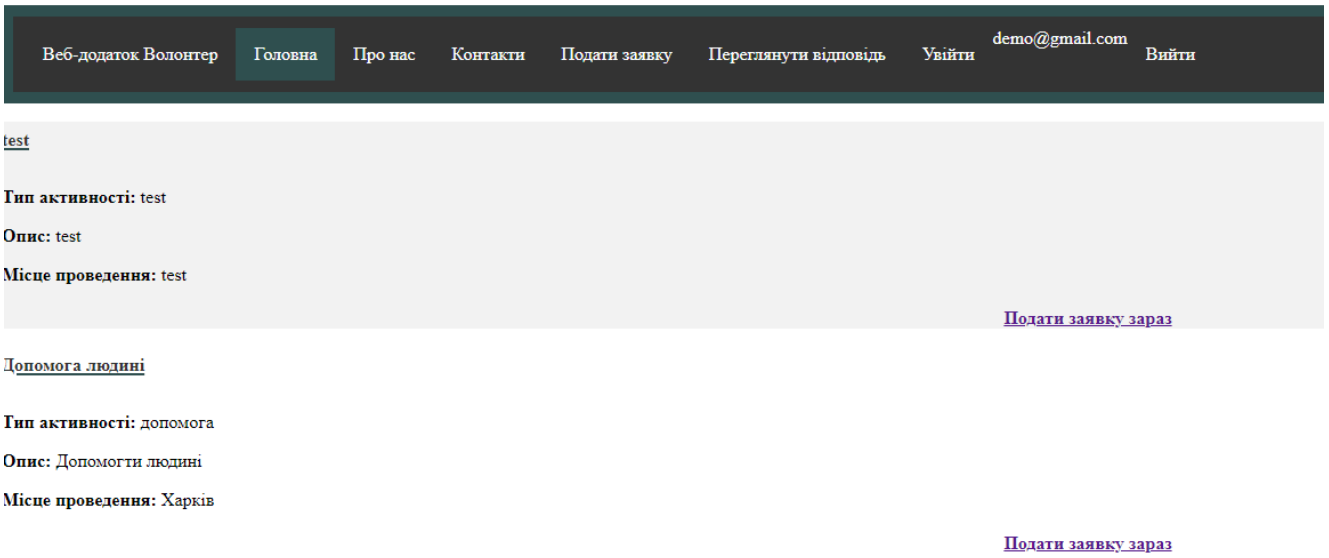

Рисунок 3.4 – Особиста сторінка користувача (дашборд)

Для волонтерів на дашборді відображається список доступних волонтерських заходів (рис. 3.5), на які вони можуть подати заявку на участь. Кожен захід містить детальну інформацію, таку як назва, опис, дата та час проведення, місце проведення та кількість потрібних волонтерів. Волонтери можуть переглядати деталі заходу та подавати заявки на участь, заповнюючи відповідну форму (рис. 3.6).

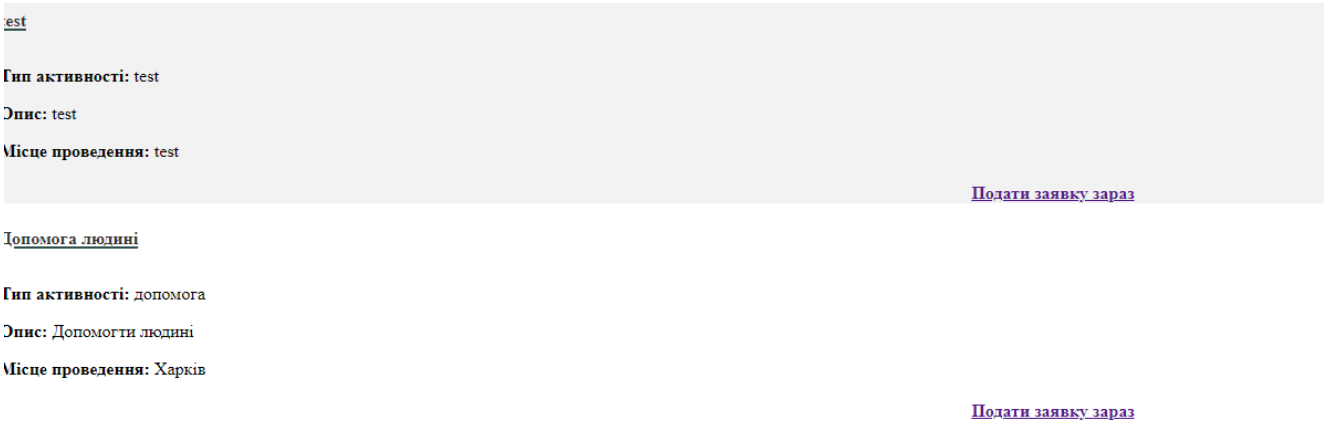

Рисунок 3.5 – Список доступних волонтерських заходів

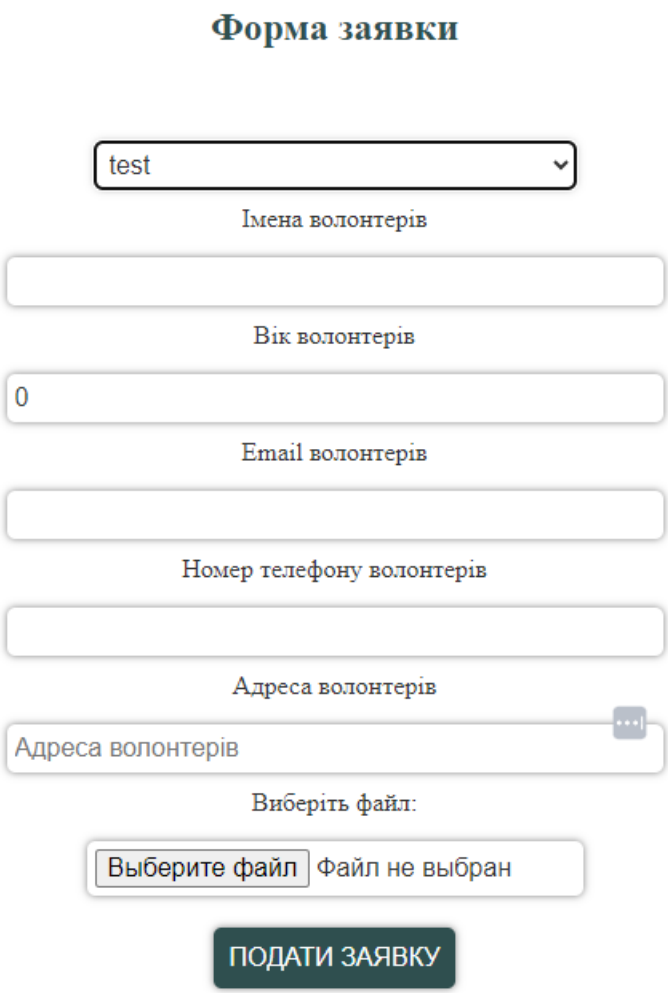

Рисунок 3.6 – Форма подачі заявки на участь у волонтерському заході

Організатори мають можливість створювати нові волонтерські заходи через спеціальну форму (рис. 3.7). Вони можуть вказати назву заходу, опис, дату та час проведення, місце проведення, потрібну кількість волонтерів та інші деталі. Також організатори можуть редагувати інформацію про існуючі заходи та переглядати список заявок від волонтерів на участь у їхніх заходах.

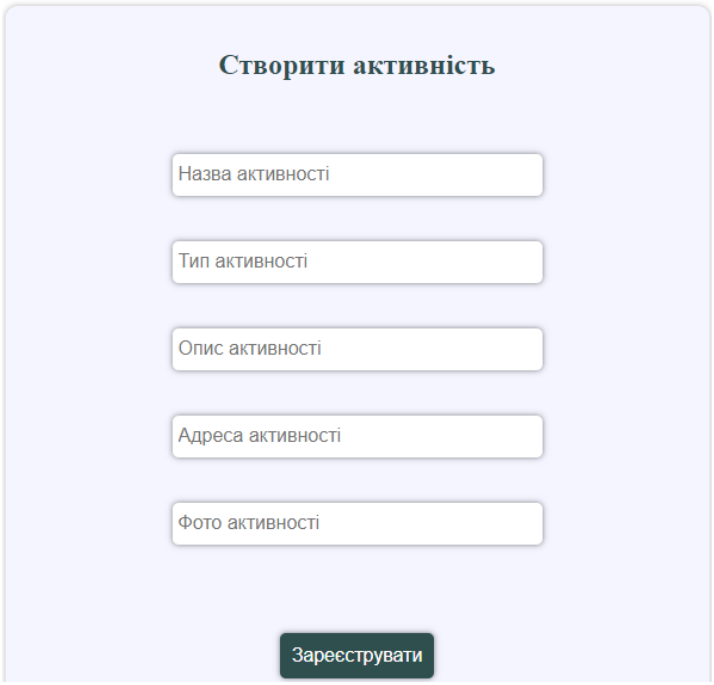

Рисунок 3.7 – Форма створення нового волонтерського заходу

Адміністратори мають доступ до панелі адміністрування (рис. 3.8), де вони можуть керувати користувачами, модерувати контент та виконувати інші адміністративні функції. Вони можуть переглядати список усіх користувачів, редагувати їхні дані, надавати або забирати права доступу, а також видаляти користувачів за потреби.

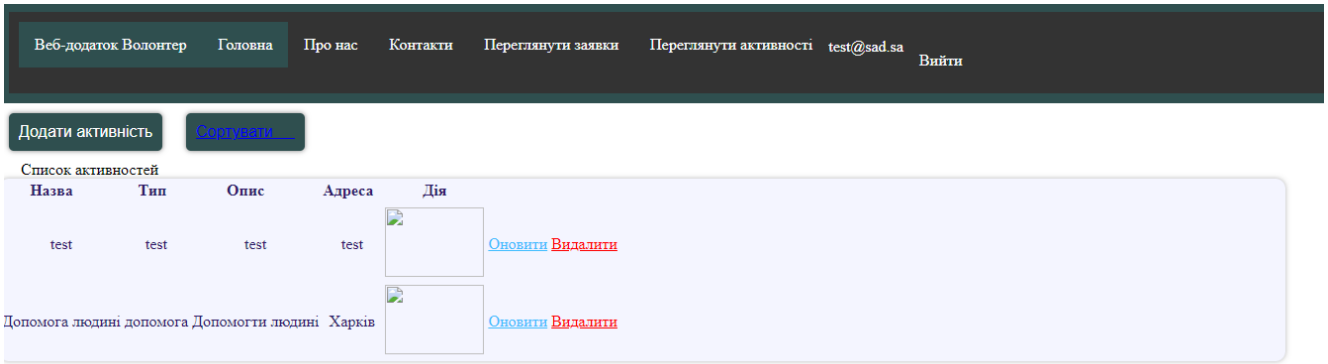

Рисунок 3.8 – Панель адміністрування

Окрім основних функцій, вебзастосунок містить додаткові сторінки, такі як сторінка "Про нас", де розміщена інформація про команду розробників та мету створення платформи, а також сторінка з контактною інформацією для зворотного зв'язку та підтримки користувачів.

Загалом, інтерфейс користувача вебзастосунку для підтримки волонтерської діяльності розроблено з урахуванням принципів зручності використання (usability) та естетичної привабливості. Використання фреймворку Bootstrap забезпечує адаптивність дизайну, що дозволяє комфортно користуватися застосунком на різних пристроях, таких як настільні комп'ютери, планшети та мобільні телефони.

Навігація по застосунку здійснюється за допомогою інтуїтивно зрозумілого меню, яке містить посилання на основні розділи та функції. Кнопки та посилання мають чіткі та зрозумілі назви, що полегшує орієнтування користувача на сторінках.

Форми для введення даних, такі як форма реєстрації, форма подачі заявки на участь у заході або форма створення нового заходу, розроблені з урахуванням зручності заповнення та валідації даних. Поля форм мають підписи та підказки, що допомагають користувачеві правильно вводити інформацію. У разі некоректного введення даних, застосунок відображає відповідні повідомлення про помилки та підсвічує проблемні поля.

Для забезпечення візуальної привабливості інтерфейсу використовуються приємні кольорові схеми, іконки та зображення, які відповідають тематиці волонтерської діяльності. Текстовий контент оформлений з використанням читабельних шрифтів та розмірів, що забезпечує комфортне сприйняття інформації.

Таким чином, інтерфейс користувача вебзастосунку для підтримки волонтерської діяльності розроблено з метою забезпечення зручності використання, естетичної привабливості та ефективного виконання основних функцій платформи. Користувачі різних ролей (волонтери, організатори, адміністратори) мають доступ до

відповідних розділів та функцій застосунку, що дозволяє їм ефективно взаємодіяти та досягати своїх цілей у рамках волонтерської діяльності.

3.3 Тестування та аналіз результатів

Тестування є невід'ємною частиною процесу розробки програмного забезпечення, і вебзастосунок для підтримки волонтерської діяльності не є винятком. Метою тестування є перевірка коректності роботи застосунку, виявлення потенційних помилок та недоліків, а також забезпечення відповідності вимогам та очікуванням користувачів.

Для тестування вебзастосунку було використано різні методи та підходи, які охоплюють різні аспекти функціонування системи. Розглянемо детальніше кожен з них.

1. Модульне тестування (Unit testing) Модульне тестування передбачає перевірку коректності роботи окремих компонентів (модулів) застосунку незалежно від інших. У цьому проекті для модульного тестування використовувався фреймворк JUnit.

Було написано модульні тести для основних класів та методів застосунку, таких як контролери, сервіси та репозиторії. Наприклад, для класу UserService було перевірено методи реєстрації нового користувача, авторизації, оновлення профілю користувача тощо. Для класу VolunteeringActivityService тестувалися методи створення нового заходу, редагування заходу, отримання списку заходів та інші.

Модульні тести дозволили переконатися, що кожен компонент застосунку працює коректно та повертає очікувані результати при різних вхідних даних. Було перевірено граничні випадки, некоректні дані та можливі помилки, щоб забезпечити стійкість та надійність роботи застосунку.

2. Інтеграційне тестування (Integration testing) Інтеграційне тестування спрямоване на перевірку взаємодії між різними компонентами системи. У цьому

проекті інтеграційне тестування проводилося для перевірки взаємодії між контролерами, сервісами та репозиторіями.

Наприклад, було протестовано процес створення нового волонтерського заходу, який включає взаємодію між VolunteeringActivityController, VolunteeringActivityService та VolunteeringActivityRepository. Тести перевіряли, чи коректно передаються дані між цими компонентами, чи зберігаються дані в базі даних та чи повертаються очікувані результати.

Також було протестовано процес подачі заявки на участь у волонтерському заході, який залучає взаємодію між VolunteeringApplicationController, VolunteeringApplicationService, UserService та відповідними репозиторіями.

Інтеграційне тестування допомогло виявити потенційні проблеми у взаємодії між компонентами та переконатися, що система працює коректно як єдине ціле.

3. Системне тестування (System testing) Системне тестування передбачає перевірку функціонування системи в цілому з точки зору кінцевого користувача. Для вебзастосунку було проведено ручне тестування основних сценаріїв використання.

Було перевірено реєстрацію нового користувача, авторизацію, створення волонтерського заходу, подачу заявки на участь, редагування профілю користувача та інші ключові функції застосунку. Тестувальники виступали в ролі звичайних користувачів і виконували дії, передбачені системою.

Системне тестування дозволило оцінити зручність використання інтерфейсу, коректність відображення даних, правильність роботи навігації та інші аспекти взаємодії користувача з системою. Було виявлено та виправлено деякі недоліки в дизайні інтерфейсу та поведінці застосунку.

4. Тестування безпеки (Security testing) Забезпечення безпеки є критично важливим для будь-якого вебзастосунку. Було проведено тестування безпеки, щоб переконатися у захищеності системи від потенційних загроз та вразливостей.

Тестувалася стійкість автентифікації та авторизації користувачів, перевірялося, чи неавторизовані користувачі не мають доступу до захищених сторінок та функцій. Також було перевірено захист від поширених атак, таких як міжсайтовий скриптинг (XSS), ін'єкції SQL та інші.

Результати тестування безпеки показали, що застосунок має належний рівень захисту та дотримується рекомендованих практик безпеки веброзробки.

5. Тестування продуктивності (Performance testing) Тестування продуктивності спрямоване на оцінку швидкодії та масштабованості застосунку під навантаженням. Було проведено навантажувальне тестування, щоб перевірити, як система поводиться при одночасному доступі великої кількості користувачів.

За допомогою спеціальних інструментів, таких як Apache JMeter, було змодельовано сценарії з різною кількістю одночасних користувачів, які виконували типові дії, такі як перегляд списку заходів, подача заявок на участь тощо. Було зібрано дані про час відгуку системи, використання ресурсів сервера та інші показники продуктивності.

Результати тестування продуктивності показали, що застосунок здатний витримувати навантаження та забезпечувати прийнятний час відгуку для користувачів. Було виявлено деякі потенційні вузькі місця та запропоновано оптимізації для покращення продуктивності системи.

Аналіз результатів тестування Після проведення різних видів тестування було проаналізовано отримані результати та зроблено відповідні висновки.

Модульне тестування показало, що окремі компоненти застосунку працюють коректно та відповідають очікуваній поведінці. Більшість модульних тестів пройшли успішно, а виявлені помилки були оперативно виправлені.

Інтеграційне тестування допомогло переконатися у правильній взаємодії між компонентами системи. Були виявлені деякі проблеми з передачею даних між компонентами, які були успішно вирішені.

Системне тестування дозволило оцінити загальну функціональність та зручність використання застосунку з точки зору користувача. Були виявлені недоліки в інтерфейсі користувача та деякі помилки в роботі окремих функцій, які були виправлені перед випуском застосунку.

Тестування безпеки показало, що застосунок має належний рівень захисту від потенційних загроз. Однак було рекомендовано додатково посилити деякі аспекти безпеки, такі як шифрування даних та використання більш надійних алгоритмів хешування паролів.

Тестування продуктивності виявило, що застосунок здатний обробляти значне навантаження, але все ж потребує деяких оптимізацій для забезпечення кращої масштабованості. Було запропоновано поліпшити алгоритми обробки запитів та використовувати кешування даних для зменшення навантаження на базу даних.

Загалом, результати тестування показали, що вебзастосунок для підтримки волонтерської діяльності функціонує коректно, відповідає визначеним вимогам та забезпечує належний рівень безпеки та продуктивності. Виявлені недоліки та помилки були успішно виправлені, що підвищило якість та надійність системи.

Однак, незважаючи на ретельне тестування, завжди існує ймовірність наявності прихованих помилок або невиявлених проблем. Тому важливо продовжувати моніторинг роботи застосунку після його розгортання та збирати відгуки від реальних користувачів. Це дозволить своєчасно виявляти та виправляти будь-які недоліки, а також вносити необхідні покращення та доповнення до функціональності системи.

Процес тестування та аналізу результатів є невід'ємною частиною розробки якісного програмного забезпечення. Завдяки ретельному тестуванню вдалося забезпечити коректність роботи, безпеку та продуктивність вебзастосунку для підтримки волонтерської діяльності. Це дає впевненість у тому, що застосунок відповідає очікуванням користувачів та здатний ефективно виконувати свої функції у реальних умовах експлуатації.

Висновки до розділу 3

У третьому розділі дипломної роботи було детально розглянуто процес реалізації та тестування вебзастосунку для підтримки волонтерської діяльності. Було описано особливості програмної реалізації, представлено інтерфейс користувача та проаналізовано результати тестування.

Особливості програмної реалізації вебзастосунку полягають у використанні сучасних технологій та фреймворків, таких як Spring Boot, Java, PostgreSQL, Thymeleaf та інших. Застосування архітектурного шаблону MVC дозволило чітко розділити логіку застосунку на компоненти, що відповідають за модель даних, представлення та контролери. Завдяки використанню Spring Boot вдалося швидко та ефективно розробити застосунок, використовуючи готові рішення та автоконфігурацію.

Робота з базою даних PostgreSQL через Spring Data JPA дозволила абстрагуватися від деталей конкретної СУБД та працювати з даними у вигляді об'єктів Java. Для забезпечення безпеки було реалізовано автентифікацію та авторизацію користувачів за допомогою Spring Security, що гарантує захист від несанкціонованого доступу та розмежування прав доступу відповідно до ролей користувачів.

Інтерфейс користувача вебзастосунку було розроблено з використанням HTML, CSS та JavaScript, а також фреймворку Bootstrap для створення адаптивного та привабливого дизайну. Користувачі мають доступ до зручної навігації, зрозумілих форм для введення даних та інформативних сторінок, що відображають списки волонтерських заходів, профілі користувачів та інші дані. Інтерфейс розроблено з урахуванням принципів зручності використання та естетичної привабливості.

Тестування вебзастосунку було проведено з використанням різних методів та підходів, що охоплюють різні аспекти функціонування системи. Модульне тестування дозволило перевірити коректність роботи окремих компонентів застосунку, таких як контролери, сервіси та репозиторії. Інтеграційне тестування було спрямоване на

перевірку взаємодії між різними компонентами системи та виявлення потенційних проблем у їх взаємодії.

Системне тестування дозволило оцінити загальну функціональність та зручність використання застосунку з точки зору кінцевого користувача. Було проведено ручне тестування основних сценаріїв використання, що дозволило виявити недоліки в інтерфейсі та поведінці застосунку. Тестування безпеки показало, що застосунок має належний рівень захисту від потенційних загроз та вразливостей.

Тестування продуктивності дозволило оцінити швидкодію та масштабованість застосунку під навантаженням. Було виявлено, що застосунок здатний витримувати значне навантаження, але все ж потребує деяких оптимізацій для забезпечення кращої продуктивності.

Аналіз результатів тестування показав, що вебзастосунок для підтримки волонтерської діяльності функціонує коректно, відповідає визначеним вимогам та забезпечує належний рівень безпеки та продуктивності. Виявлені недоліки та помилки були успішно виправлені, що підвищило якість та надійність системи.

Однак, незважаючи на ретельне тестування, завжди існує ймовірність наявності прихованих помилок або невиявлених проблем. Тому важливо продовжувати моніторинг роботи застосунку після його розгортання та збирати відгуки від реальних користувачів. Це дозволить своєчасно виявляти та виправляти будь-які недоліки, а також вносити необхідні покращення та доповнення до функціональності системи.

Процес реалізації та тестування вебзастосунку для підтримки волонтерської діяльності показав, що використання сучасних технологій та дотримання кращих практик розробки дозволяє створити якісний та надійний продукт, який відповідає потребам користувачів та ефективно виконує свої функції. Завдяки ретельному тестуванню вдалося забезпечити коректність роботи, безпеку та продуктивність застосунку, що є запорукою успішного функціонування системи у реальних умовах.

Подальші кроки можуть включати розширення функціональності вебзастосунку, додавання нових можливостей та вдосконалення існуючих. Також важливо продовжувати підтримку та оновлення застосунку, щоб забезпечити його відповідність сучасним стандартам та вимогам безпеки.

Загалом, розділ 3 продемонстрував успішну реалізацію та тестування вебзастосунку для підтримки волонтерської діяльності, що є важливим етапом у створенні повноцінного та ефективного програмного продукту. Отримані результати свідчать про те, що застосунок готовий до впровадження та використання у реальних умовах, де він зможе принести користь волонтерам, організаторам та суспільству загалом.

ВИСНОВКИ

У дипломній роботі було розроблено вебзастосунок для підтримки волонтерської діяльності, який забезпечує ефективну взаємодію між волонтерами, організаторами волонтерських заходів та адміністраторами платформи. Застосунок надає зручні інструменти для пошуку та участі у волонтерських активностях, управління заходами, комунікації та координації дій учасників волонтерського руху.

У першому розділі було проведено ґрунтовний аналіз предметної сфери волонтерської діяльності, досліджено її значення та роль у сучасному суспільстві. Було розглянуто історію розвитку волонтерського руху, його правові засади та основні напрямки діяльності. Особливу увагу було приділено ролі інформаційних технологій у підтримці та організації волонтерства, зокрема, використанню спеціалізованих вебзастосунків.

Проведений огляд існуючих програмних рішень для підтримки волонтерської діяльності виявив їх переваги та недоліки. Було відзначено необхідність створення більш функціонального та адаптованого під потреби сфери вебзастосунку, який би враховував специфіку волонтерської діяльності та надавав зручні інструменти для управління проектами, комунікації та мотивації волонтерів.

На основі аналізу предметної сфери та огляду аналогів було сформульовано мету та основні вимоги до вебзастосунку. Застосунок повинен забезпечувати можливості для створення та публікації волонтерських вакансій, пошуку та подачі заявок на участь у заходах, управління завданнями та розкладом волонтерів, комунікації між учасниками, генерації звітності та аналітики, а також підтримувати створення локальних волонтерських спільнот.

У другому розділі було проведено проектування та моделювання вебзастосунку. Було розроблено функціональну модель системи з використанням UML-діаграми варіантів використання, яка відображає взаємодію між акторами (волонтерами, організаторами, адміністраторами) та основними функціями застосунку. Також було

спроектовано базу даних, яка забезпечує ефективне зберігання та обробку інформації про користувачів, волонтерські заходи та заявки на участь.

Після ретельного аналізу було обрано технології та архітектуру для розробки вебзастосунку. Основою стеку технологій стали мова програмування Java, фреймворк Spring Boot, база даних PostgreSQL, бібліотека Hibernate для роботи з базою даних та шаблонізатор Thymeleaf для створення динамічних веб-сторінок. Архітектура застосунку базується на принципах мікросервісів, що забезпечує гнучкість, масштабованість та можливість незалежної розробки окремих компонентів системи.

Особливу увагу було приділено питанням безпеки вебзастосунку. Було передбачено використання надійних механізмів автентифікації та авторизації користувачів, шифрування даних, захист від поширених веб-атак та дотримання вимог законодавства щодо захисту персональних даних.

У третьому розділі було детально описано процес реалізації вебзастосунку з використанням обраних технологій та архітектури. Було представлено особливості програмної реалізації, включаючи роботу з базою даних, імплементацію бізнес-логіки у сервісах, забезпечення безпеки за допомогою Spring Security, реалізацію функціоналу завантаження та скачування файлів, пагінації даних тощо.

Інтерфейс користувача вебзастосунку було розроблено з урахуванням принципів зручності використання та естетичної привабливості. Було створено інтуїтивно зрозумілу навігацію, інформативні сторінки для перегляду волонтерських заходів та профілів користувачів, зручні форми для введення даних та подачі заявок на участь у заходах. Використання адаптивного дизайну забезпечує коректне відображення застосунку на різних пристроях.

Тестування вебзастосунку було проведено з використанням різних методів та підходів, включаючи модульне, інтеграційне, системне тестування, тестування безпеки та продуктивності. Результати тестування показали, що застосунок функціонує коректно, відповідає визначеним вимогам та забезпечує належний рівень

безпеки та продуктивності. Виявлені недоліки та помилки були успішно виправлені, що підвищило якість та надійність системи.

Розроблений вебзастосунок має значну практичну цінність для розвитку волонтерського руху. Він дозволяє спростити та оптимізувати процеси організації волонтерської діяльності, залучити більше людей до участі у суспільно корисних справах, покращити комунікацію та координацію між учасниками волонтерських проектів. Застосунок надає зручні інструменти для пошуку волонтерських можливостей, управління заходами та мотивації волонтерів, що сприяє підвищенню ефективності та масштабованості волонтерських ініціатив.

Впровадження розробленого вебзастосунку матиме позитивний вплив на розвиток волонтерства в Україні та посилення його ролі у вирішенні суспільно значущих проблем. Застосунок дозволить залучити ширше коло людей до волонтерської діяльності, підвищити рівень обізнаності про можливості волонтерства та сприяти формуванню активної громадянської позиції у суспільстві.

Подальші напрямки розвитку вебзастосунку можуть включати розширення функціональності, додавання нових можливостей для взаємодії між учасниками, інтеграцію з іншими платформами та сервісами, які використовуються у волонтерській діяльності. Також важливо забезпечувати постійну підтримку та оновлення застосунку, щоб гарантувати його відповідність сучасним стандартам та вимогам безпеки.

У підсумку, розроблений вебзастосунок для підтримки волонтерської діяльності є важливим інструментом для розвитку та популяризації волонтерського руху. Він демонструє, як сучасні інформаційні технології можуть ефективно застосовуватися для вирішення соціально значущих завдань та сприяння активній участі громадян у житті суспільства. Завдяки ретельному аналізу предметної сфери, проектуванню та реалізації з використанням передових технологій та підходів,

застосунок має потенціал стати надійною платформою для організації та координації волонтерських ініціатив на локальному та національному рівнях.

Дипломна робота продемонструвала комплексний підхід до розробки вебзастосунку для підтримки волонтерської діяльності, який враховує специфіку предметної сфери, потреби користувачів та сучасні тенденції у веброзробці. Результати роботи свідчать про те, що застосунок готовий до впровадження та використання у реальних умовах, де він зможе принести користь волонтерам, організаторам та суспільству загалом.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Закон України "Про волонтерську діяльність" від 19.04.2011 № 3236-VI. URL: https://zakon.rada.gov.ua/laws/show/3236-17 (дата звернення: 20.05.2023).

2. Бєспалова К. О. Мотивація волонтерської діяльності. Вісник Національного університету "Львівська політехніка". Серія: Юридичні науки. 2016. № 850. С. 131-136.

3. Лях Т. Л. Волонтерство як суспільний феномен. Проблеми громадянського поступу українського суспільства: філософсько-правові та соціально-психологічні аспекти. 2015. С. 119-139.

4. Єнін М. Н. Волонтерський рух в Україні: основні тенденції розвитку. Науковий часопис НПУ імені М. П. Драгоманова. Серія 18: Економіка і право. 2015. Вип. 27. С. 117-123.

5. VolunteerMatch. URL: https://www.volunteermatch.org/ (дата звернення: 20.05.2023).

6. Idealist. URL: https://www.idealist.org/ (дата звернення: 20.05.2023).

7. AllForGood. URL: https://www.allforgood.org/ (дата звернення: 20.05.2023).

8. Volgistics. URL: https://www.volgistics.com/ (дата звернення: 20.05.2023).

9. Golden. URL: https://www.goldenvolunteer.com/ (дата звернення: 20.05.2023).

10. InitLive. URL: https://www.initlive.com/ (дата звернення: 20.05.2023).

11. Voida A. et al. Volunteer management information systems: Challenges and opportunities for the third sector. Nonprofit Management and Leadership. 2011. Vol. 22, No. 2. P. 163-180.

12. Senseney M. et al. Volunteer retention through information systems: A case study. Journal of Community Informatics. 2017. Vol. 13, No. 3. P. 64-80.

13. Lee Y. et al. Gamification in volunteer management: A case study of online volunteer communities. International Journal of Human–Computer Interaction. 2018. Vol. 34, No. 8. P. 744-755.

14. Choi H., Kim M. Factors influencing volunteers' intention to use information systems: A case of a non-profit organization. Journal of Computer Information Systems. 2018. Vol. 58, No. 4. P. 348-358.

15. Spring Boot. URL: https://spring.io/projects/spring-boot (дата звернення: 20.05.2023).

16. PostgreSQL. URL: https://www.postgresql.org/ (дата звернення: 20.05.2023).

17. Hibernate. URL: https://hibernate.org/ (дата звернення: 20.05.2023).

18. Spring MVC. URL: https://docs.spring.io/spring-framework/docs/current/reference/html/web.html (дата звернення: 20.05.2023).

19. Thymeleaf. URL: https://www.thymeleaf.org/ (дата звернення: 20.05.2023).

20. Bootstrap. URL: https://getbootstrap.com/ (дата звернення: 20.05.2023).

21. Fowler M. Microservices. URL: https://martinfowler.com/articles/microservices.html (дата звернення: 20.05.2023).

22. Fielding R. T. Architectural Styles and the Design of Network-based Software Architectures. Doctoral dissertation. University of California, Irvine, 2000.

23. Spring Security. URL: https://spring.io/projects/spring-security (дата звернення: 20.05.2023).

24. JSON Web Tokens. URL: https://jwt.io/ (дата звернення: 20.05.2023).

25. JUnit. URL: https://junit.org/ (дата звернення: 20.05.2023).

26. Mockito. URL: https://site.mockito.org/ (дата звернення: 20.05.2023).

27. Apache JMeter. URL: https://jmeter.apache.org/ (дата звернення: 20.05.2023).

28. Моделі в програмній інженерії: навч. посібник / В. В. Яковина, Д. С. Федасюк, О. О. Нитребич, І. І. Ковалюк. Львів: Львівська політехніка, 2020. 196 с.

29. Patterns of Enterprise Application Architecture / M. Fowler. - Addison-Wesley Professional, 2002. - 560 p.

30. Pro Spring 5: An In-Depth Guide to the Spring Framework and Its Tools / I. Cosmina et al. - Apress, 2017. - 868 p.

31. Clean Code: A Handbook of Agile Software Craftsmanship / R. C. Martin. - Pearson, 2008. - 464 p.

32. Test-Driven Development: By Example / K. Beck. - Addison-Wesley Professional, 2002. - 240 p.

33. Spring Start Here: Learn what you need and learn it well / L. Spilca. - Manning Publications, 2021. - 528 p.

34. Learning Web Design: A Beginner's Guide to HTML, CSS, JavaScript, and Web Graphics / J. Niederst Robbins. - O'Reilly Media, 2018. - 808 p.

35. CSS: The Definitive Guide: Visual Presentation for the Web / E. A. Meyer, E. Weyl. - O'Reilly Media, 2017. - 1090 p.

36. JavaScript: The Definitive Guide: Master the World's Most-Used Programming Language / D. Flanagan. - O'Reilly Media, 2020. - 706 p.

ДОДАТОК А

Лістинг коду

Програмний код application.properties.

```
spring.datasource.url=jdbc:postgresql://127.0.0.1:5432/kanban
spring.datasource.username=user
spring.datasource.password=password
spring.datasource.driver-class-name=org.postgresql.Driver
spring.jpa.hibernate.ddl-auto=update
spring.security.user.name=test
spring.security.user.password=test
spring.jpa.show-sql=true
server.port=25324

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.PostgreSQLDialect
```

Програмний код UserController.java.

```
package com.volunteer.main.controller;

import com.volunteer.main.model.VolunteeringApplication;
import com.volunteer.main.services.VolunteeringApplicationService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;

import java.security.Principal;
import java.util.List;

@Controller
public class UserController {
    @Autowired
```

```
    private VolunteeringApplicationService volunteeringApplicationService;

    @RequestMapping(value = {"/dashboard"}, method = RequestMethod.GET)
    public String homePage(){
        return "/dash";
    }

    @GetMapping("/user")
    public String showUserApplications(Model model, Principal principal) {
        String userEmail = principal.getName();
        List<VolunteeringApplication> applications =
volunteeringApplicationService.getApplicationsForUser(userEmail);
        model.addAttribute("UserAppView", applications);
        return "userApplication";
    }

}
```

VolunteeringActivityController.java

```
package com.volunteer.main.controller;

import com.volunteer.main.model.VolunteeringActivity;
import com.volunteer.main.services.impl.VolunteeringActivityServiceImpl;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.Page;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@Controller

    public class VolunteeringActivityController {

        @Autowired
```

```
        private VolunteeringActivityServiceImpl volunteeringActivityService;



    @GetMapping("/add")
    public String addActivity(Model model){
        model.addAttribute("activity",new VolunteeringActivity());
        //return "saveActivity";
        return "createActivitys";
    }

    @PostMapping("/save")
    public String saveActivity(@ModelAttribute("activity") VolunteeringActivity theActivity)
{
        VolunteeringActivity
activity=volunteeringActivityService.createActivity(theActivity);
        if (activity != null) {
            return "redirect:/view";
        }
        return "sorry";
    }

    @GetMapping("/view")
    public String showActivities(Model model){
        List<VolunteeringActivity> activities=volunteeringActivityService.getAllActivities();
        model.addAttribute("activity",activities);
        return "list_activitie";

    }
    @GetMapping("/Dash_view")
    public String showActivitiesDash(Model model){
        List<VolunteeringActivity> activities=volunteeringActivityService.getAllActivities();
        model.addAttribute("activity",activities);
        return "dis_Activity";

    }
```

```
    @GetMapping("/showUpdateActivity/{actId}")
    public String showUpdateVolunteer(@PathVariable(value = "actId") Long id,
                                      Model model){
        VolunteeringActivity volunteer = volunteeringActivityService.getActivityById(id);
        model.addAttribute("activity",volunteer);
        return "updateActivitys";
    }

    @GetMapping("/deleteActivity/{actId}")
    public String deleteVolunteer(@PathVariable (value = "actId") Long actId){
        this.volunteeringActivityService.deleteActivity(actId);
        return "redirect:/view";
    }

    @GetMapping("/pageVolunteer/{pageNo}")
    public String findPaginatedVolunteer(@PathVariable(value = "pageNo") int pageNo,
                                         @RequestParam("sortFieldVolunteer") String
sortFieldVolunteer,
                                         @RequestParam("sortDirVolunteer") String
sortDirVolunteer,
                                         Model model) {
        int pageSizeVolunteer = 3;

        Page< VolunteeringActivity > page = volunteeringActivityService.findPaginated(pageNo,
pageSizeVolunteer, sortFieldVolunteer, sortDirVolunteer);
        List< VolunteeringActivity > listVolunteers = page.getContent();

        model.addAttribute("currentPageVolunteer", pageNo);
        model.addAttribute("totalPagesVolunteer", page.getTotalPages());
        model.addAttribute("totalItemsVolunteer", page.getTotalElements());

        model.addAttribute("sortFieldVolunteer", sortFieldVolunteer);
        model.addAttribute("sortDirVolunteer", sortDirVolunteer);
        model.addAttribute("listVolunteers", listVolunteers);
        model.addAttribute("reverseSortDirVolunteer", sortDirVolunteer.equals("asc") ? "desc"
: "asc");
```

```
        return "list_activitie";
    }

}
```

Програмний код VolunteeringApplicationController.java.

```
package com.volunteer.main.controller;

import com.volunteer.main.model.VolunteeringActivity;
import com.volunteer.main.model.VolunteeringApplication;
import com.volunteer.main.repository.VolunteeringApplicationRepository;
import com.volunteer.main.services.UserServiceImpl;
import com.volunteer.main.services.VolunteeringActivityService;
import com.volunteer.main.services.VolunteeringApplicationService;

import com.volunteer.main.services.impl.VolunteeringActivityServiceImpl;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.Page;
import org.springframework.http.HttpHeaders;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.multipart.MultipartFile;
import org.springframework.web.servlet.mvc.support.RedirectAttributes;
import org.springframework.web.servlet.support.ServletUriComponentsBuilder;

import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import java.security.Principal;
import java.util.List;


@Controller
    public class VolunteeringApplicationController {
```

```
        @Autowired
        private VolunteeringApplicationService volunteeringApplicationService;
        @Autowired
        private VolunteeringApplicationRepository v_repo;
        @Autowired
        private VolunteeringActivityService v_activity;
    @Autowired
    private VolunteeringActivityServiceImpl volunteeringActivityService;
    @Autowired
    private UserServiceImpl UserService;
        @GetMapping("/main")
        public String loadMainPage() {
        return "main-page";
    }

//    @GetMapping("/applys")
//    public String addActivity(Model model){
//        model.addAttribute("application",new VolunteeringApplication());
//        return "createApplication";
//    }
        @GetMapping("/applicationView")
        public String showApplications(Model model) {
            List<VolunteeringApplication> application =
volunteeringApplicationService.getApplicationsForActivity();
            model.addAttribute("applications", application);
            return "list_application";
        }

    @GetMapping("/about")
    public String uploadDoc() {
       return "about-us";
    }


    @GetMapping("/applications/{id}/download")
```

```
    public String downloadFile(@PathVariable String id, HttpServletResponse response,
RedirectAttributes redirectAttributes) throws Exception {
        VolunteeringApplication volunteeringApplication =
volunteeringApplicationService.downloadFile(id);
        response.setContentType(volunteeringApplication.getContentType());
        response.setHeader(HttpHeaders.CONTENT_DISPOSITION, "attachment; filename=\"" +
volunteeringApplication.getFileName() + "\"");
        response.getOutputStream().write(volunteeringApplication.getVolunteerDocument());
        redirectAttributes.addFlashAttribute("successMessage", "File downloaded
successfully!");
       // return "redirect:/uploads";
        return "redirect:/applications/{id}/download";
    }


    @GetMapping("/deleteApplication/{id}")
    public String deleteVolunteer(@PathVariable (value = "id") Long id){
        this.volunteeringApplicationService.deleteApplication(id);
        return "redirect:/applicationView";
    }

    @GetMapping("/pageApplication/{pageNo}")
    public String findPaginatedVolunteer(@PathVariable(value = "pageNo") int pageNo,
                                         @RequestParam("sortFieldVolunteer") String
sortFieldVolunteer,
                                         @RequestParam("sortDirVolunteer") String
sortDirVolunteer,
                                         Model model) {
        int pageSizeVolunteer = 3;

        Page< VolunteeringApplication > page =
volunteeringApplicationService.findPaginated(pageNo, pageSizeVolunteer, sortFieldVolunteer,
sortDirVolunteer);
        List< VolunteeringApplication > listVolunteers = page.getContent();

        model.addAttribute("currentPage", pageNo);
        model.addAttribute("totalPages", page.getTotalPages());
```

```
        model.addAttribute("totalItems", page.getTotalElements());

        model.addAttribute("sortField", sortFieldVolunteer);
        model.addAttribute("sortDir", sortDirVolunteer);
        model.addAttribute("list", listVolunteers);
        model.addAttribute("reverseSortDir", sortDirVolunteer.equals("asc") ? "desc" :
"asc");


        return "list_application";
    }
}
;
```

Програмний код indexx.java.

```
package com.volunteer.main.controller;

import com.volunteer.main.model.VolunteeringActivity;
import com.volunteer.main.services.impl.VolunteeringActivityServiceImpl;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;

import java.util.List;

@Controller
public class indexx {

    @Autowired
    private VolunteeringActivityServiceImpl volunteeringActivityService;

    @GetMapping("/C")
    public String addActivity() {
        return "dis_Activity";
    }
```

```
    @GetMapping("/views")
    public String showActiviti(Model model) {
        List<VolunteeringActivity> activities =
volunteeringActivityService.getAllActivities();
        model.addAttribute("activitys", activities);
//        return "list_activitie";
        return "dis_Activity";
    }
}
```

Програмний код application.properties.

```
package com.volunteer.main.controller;

import com.volunteer.main.model.User;
import com.volunteer.main.services.UserService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;

import java.util.List;

@Controller
public class AuthController {
    @Autowired
    UserService userService;
//    private User user;

    @RequestMapping(value = {"/login"}, method = RequestMethod.GET)
    public String login(){
            return "auth/login";
        //return "login";
```

```
    }

    @RequestMapping(value = {"/register"}, method = RequestMethod.GET)
    public String register(Model model){
        model.addAttribute("user", new User());
        return "auth/register";
        //return "main-page";
    }

    @RequestMapping(value = {"/register"}, method = RequestMethod.POST)
    public String registerUser(Model model, User user, BindingResult bindingResult){
        System.out.println(
                "Registered well"
        );
        if(bindingResult.hasErrors()){
            model.addAttribute("successMessage", "User registered successfully!");
            model.addAttribute("bindingResult", bindingResult);
            return "auth/register";
        }
        List<Object> userPresentObj = userService.isUserPresent(user);
        if((Boolean) userPresentObj.get(0)){
            model.addAttribute("successMessage", userPresentObj.get(1));
            return "auth/register";
        }

        userService.saveUser(user);
        model.addAttribute("successMessage", "User registered successfully!");

        return "auth/login";
    }}
```

Програмний код userApplicationController.java.

```
package com.volunteer.main.controller;

import com.volunteer.main.model.VolunteeringActivity;
```

```
import com.volunteer.main.model.VolunteeringApplication;
import com.volunteer.main.services.UserServiceImpl;
import com.volunteer.main.services.VolunteeringApplicationService;
import com.volunteer.main.services.impl.VolunteeringActivityServiceImpl;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.multipart.MultipartFile;
import org.springframework.web.servlet.mvc.support.RedirectAttributes;
import org.springframework.web.servlet.support.ServletUriComponentsBuilder;

import javax.servlet.http.HttpSession;
import java.security.Principal;
import java.util.List;

@Controller
public class userApplicationController {
    @Autowired
    private VolunteeringApplicationService volunteeringApplicationService;
    @Autowired
    private VolunteeringActivityServiceImpl volunteeringActivityService;
    @Autowired
    private UserServiceImpl userService;
@GetMapping("/apps")
public String showApplicationForm(Model model) {
    List<VolunteeringActivity> activities = volunteeringActivityService.getAllActivities();
    model.addAttribute("applicationss", new VolunteeringApplication());
    model.addAttribute("activities", activities);
    return "userApplicationForm";
}

    @PostMapping("/appl")
    public String ApplicationForm(@ModelAttribute("applicationss") VolunteeringApplication
volunteeringApplication,
                                  @RequestParam("file") MultipartFile file,
                                  @RequestParam("activity.id") Long volunteeringActivityId,
```

```
                                RedirectAttributes redirectAttributes,
                                HttpSession session, Principal principal) throws Exception
{
        if (session.getAttribute("formSubmitted") != null) {
            // Form has already been submitted, do not process again
            return "redirect:/apps";
        }

        // Set the flag to indicate that the form has been submitted
        session.setAttribute("formSubmitted", true);

        VolunteeringApplication application =
volunteeringApplicationService.applyForActivityWithFile(volunteeringApplication,
volunteeringActivityId, file);
        String downloadUrl = ServletUriComponentsBuilder
                .fromCurrentContextPath()
                .path("/volunteering/applications/")
                .path(application.getId().toString())
                .path("/download")
                .toUriString();
        redirectAttributes.addFlashAttribute("successMessage", "Application and file uploaded
successfully! Download link: " + downloadUrl);
//        return "redirect:/apps";
        if (principal != null) {
            String role = userService.getRole(principal.getName());
            if (role != null && role.equals("ADMIN")) {
               return "redirect:/applicationView";
                //return "createApplication";
            }
        }
       // return "userApplicationForm";
        return "redirect:/apps";
    }
@GetMapping("/showUpdateApplication/{id}")
public String showApplicationForm(@PathVariable("id") Long id, Model model) {
    List<VolunteeringActivity> activities = volunteeringActivityService.getAllActivities();
    model.addAttribute("activities", activities);
```

```
    if (id != null) {
        VolunteeringApplication application =
volunteeringApplicationService.getApplicationByIdUpdate(id).orElse(null);
        model.addAttribute("applicationss", application);
    } else {
        model.addAttribute("applicationss", new VolunteeringApplication());
    }

    return "updateApplication";
}
}
```