

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова

Навчально-науковий інститут комп'ютерних наук та управління проєктами
Кафедра інформаційних управляючих систем та технологій

"Допущений до захисту"

Завідувач кафедри ІУСТ



к.т.н, доц. Михелєв І.Л.

" ____ " _____ 2025 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти "магістр"

на тему: Дослідження вебплатформ та розробка
інформаційної системи волонтерської діяльності

Виконав: студент групи 6145 м


(підпис)

Белік Б.О.

Керівник роботи:

доцент кафедри ІУСТ, к.т.н.,

(посада, науковий ступінь вчене звання)


(підпис)

Кроптя А.В.

м. Миколаїв – 2025 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова

Навчально-науковий інститут комп'ютерних наук та управління проєктами

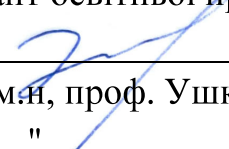
Кафедра інформаційних управляючих систем та технологій

Спеціальність 124 "Системний аналіз"

Освітня програма "Системний аналіз"

"ЗАТВЕРДЖУЮ"

Гарант освітньої програми


д.ф-м.н, проф. Ушкац М.В.

" " 2025 р.

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ на здобуття ступеня вищої освіти "магістр"

Студенту Беліку Богдану Олександровичу

(прізвище, ім'я, по батькові)

1. Тема роботи: Дослідження вебплатформ та розробка інформаційної системи волонтерської діяльності

Керівник роботи к.т.н.доцент Кроптя А.В.

Затверджені наказом ректора № 1355-уч. від "14" листопада 2025 року.

2. Термін подання роботи: 15 грудня 2025 р.

3. Вихідні дані до проєкту (роботи) ДСТУ щодо обробки інформації, літературні джерела, технічна документація на існуючі аналоги систем, матеріали наукового стажування

4. Перелік питань, що належать до розробки (найменування розділів): Аналіз предметної області та постановка задачі, Проектування та моделювання ІС, Реалізація та тестування вебзастосунку, Розрахунок економічного ефекту, Охорона праці, Охорона навколишнього середовища

5. Перелік презентаційних матеріалів Актуальність роботи, Мета та завдання роботи, Структура стартапу волонтерської діяльності, Модель системи волонтерської діяльності, Концепція системи волонтерської діяльності, Вибір архітектури ІС, ER-діаграма сутностей бази даних, Діаграма класів-сутностей, Використані технології, Інтерфейс, Результати впровадження, Шляхи вдосконалення розробки

6. Консультанти розділів роботи

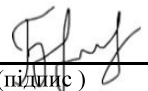
Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1	Кроптя А.В.	05.09.2024	22.12.2024
2	Кроптя А.В.	23.12.2024	15.06.2025
3	Кроптя А.В.	15.06.2025	25.09.2025
4		25.09.2025	14.11.2025
5		25.09.2025	14.11.2025
6		25.09.2025	14.11.2025

7. Дата видачі завдання 05 вересня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів роботи	Примітка
1	Дослідження та аналіз предметної галузі	22.12.2024	
2	Розробка концепції	15.03.2025	
3	Розділ з розробки проєктних рішень по системі (та її частинах)	15.06.2025	
4	Реалізація проєкту та розробка робочої документації	25.09.2025	
5	Обчислення економічної ефективності	14.11.2025	
6	Вирішення питань охорони праці та навколишнього середовища	14.11.2025	
7	Оформлення пояснювальної записки	14.11.2025	
8	Подання магістерської роботи на перевірку на плагіат	16.11.2025	
9	Подання магістерської роботи рецензенту	20.11.2025	
10	Подання магістерської роботи на державний захист	15.12.2025	

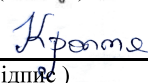
Студент


(підпис)

Бєлік Б.О.

(ПІБ)

Керівник роботи


(підпис)

Кроптя А.В.

(ПІБ)

АНОТАЦІЯ

У роботі досліджуються вебплатформи та розробка інформаційної системи для автоматизації взаємодії волонтерських організацій з волонтерами й уніфікації доступу до даних спільноти. Проведено аналіз існуючих рішень, визначено вимоги, розроблено концепцію, технічне завдання та проектні рішення, а також оформлено робочу документацію. Окремі розділи присвячені економіці, охороні праці й довкілля. Кваліфікаційна робота налічує 115 сторінок, 15 рисунків, 1 таблицю, 42 джерела і 3 додатки.

Ключові слова: вебзастосунок, волонтерство, база даних, вебінтерфейс, Spring, PostgreSQL.

ABSTRACT

This thesis is dedicated to the examination of web platforms and the development of an information system designed to support volunteer activities. The primary objective is to automate processes that facilitate interaction between volunteer organizations and volunteers, as well as to standardize access to information available on the volunteer community's web portal.

The study conducts a comprehensive analysis of existing solutions, explores methodologies for constructing such systems, establishes system requirements, develops multiple design concepts, selects the optimal solution, and prepares detailed technical specifications. It proceeds with the development of key design decisions and the creation of technical documentation for the software. Additionally, the thesis includes sections addressing economic considerations, occupational safety, and environmental protection.

The qualification work comprises 115 pages, 15 figures, 1 table, 42 references, and 3 appendices.

Keywords: web application, volunteering, database, web interface, Spring, PostgreSQL

ЗМІСТ

ВСТУП	
РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ	
1.1 Опис предметної області	
1.2 Огляд наявних аналогів та публікацій	
1.3 Постановка задачі	
Висновки до розділу 1	
РОЗДІЛ 2. РОЗРОБКА ПРОЄКТНИХ РІШЕНЬ ПО ІНФОРМАЦІЙНІЙ СИСТЕМІ	
2.1 Функціональна модель ІС	
2.2 Проєктування бази даних	
2.3. Вибір технологій та архітектури ІС	
Висновки до розділу 2	
РОЗДІЛ 3 РЕАЛІЗАЦІЯ ІС	
3.1 Особливості програмної реалізації	
3.2 Інтерфейс користувача	
3.3 Тестування та аналіз результатів	
Висновки до розділу 3	
РОЗДІЛ 4. РОЗРАХУНОК ЕКОНОМІЧНОГО ЕФЕКТУ ВІД ВПРОВАДЖЕННЯ ІС	
РОЗДІЛ 5. ОХОРОНА ПРАЦІ	
РОЗДІЛ 6. ОХОРОНА НАВКОЛИШНЬОГО СЕРЕДОВИЩА	
ВИСНОВКИ	
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	
ДОДАТКИ	

ВСТУП

Сучасне суспільство відзначається динамічним розвитком інформаційних технологій, які дедалі глибше інтегруються у всі сфери життєдіяльності людини. Не залишилася осторонь цих процесів і така вагома сфера, як волонтерська діяльність. Зростає кількість волонтерських організацій, що впроваджують у свою діяльність спеціалізовані програмні комплекси та інформаційні системи, спрямовані на підвищення ефективності комунікації, координації дій волонтерів, управління проектами тощо.

Актуальність теми дипломної роботи зумовлена існуванням нагальної потреби у створенні зручних та функціональних інформаційних систем, які б підтримували волонтерську діяльність. Такі системи забезпечують автоматизацію значної частини рутинних процесів, сприяють покращенню взаємодії між волонтерами та керівниками проектів, а також дозволяють залучати ширше коло учасників завдяки доступним онлайн-інструментам.

Провідні світові ІТ-компанії, серед яких можна відзначити Microsoft, Google, Facebook значну увагу приділяють розробці технологічних рішень для некомерційного сектору, включаючи волонтерські організації. Серед авторитетних фахівців у галузі розробки інформаційних систем для підтримки волонтерської діяльності варто виокремити Н. Сміта, Д. Коена та інших.

Метою цієї роботи є розроблення інформаційної системи для підтримки волонтерської діяльності, що дозволить підвищити ефективність функціонування волонтерів і організаторів волонтерських проектів.

Для досягнення поставленої мети необхідно виконати наступні завдання:

1. Провести аналіз предметної області волонтерської діяльності та визначити вимоги до інформаційної системи;
2. Здійснити огляд існуючих аналогічних рішень та публікацій за обраною тематикою;

3. Розробити архітектуру та визначити функціональні можливості майбутньої інформаційної системи;
4. Сформулювати структуру бази даних, необхідної для функціонування системи;
5. Реалізувати інформаційну систему із використанням сучасних вебтехнологій;
6. Провести тестування роботи створеної системи та здійснити аналіз отриманих результатів.

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

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

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

Практичне значення отриманих результатів полягає у створенні функціональної інформаційної системи, яка може використовуватись волонтерськими організаціями для підвищення продуктивності своєї діяльності. Розроблена система пропонує зручні інструменти для реєстрації волонтерів, створення волонтерських проектів, комунікації між учасниками, відстеження прогресу тощо.

Перший розділ роботи присвячений проведенню аналізу предметної галузі волонтерської діяльності, розглянуто вимоги до інформаційної системи, а також виконано огляд сучасних аналогів та літературних джерел та інтернет-ресурсів за темою. Другий розділ містить описання процесу проектування функціональних можливостей та архітектури застосунку, аналіз та створення структури бази даних. У третьому розділі розглянуто особливості програмної реалізації ІС з використанням сучасних Java технологій, а також тестуванню її роботи.

РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Опис предметної області

Волонтерська діяльність являє собою добровільну, безоплатну, соціально орієнтовану, неприбуткову активність, яку здійснюють волонтери та волонтерські організації шляхом надання допомоги. Волонтерство розглядається як одна з форм благодійної діяльності, поширеної в багатьох країнах світу. Даний вид діяльності відіграє істотну роль у підтримці соціально незахищених груп населення, а також у вирішенні різних соціальних, екологічних та гуманітарних завдань.

Історія волонтерського руху має давнє коріння і може бути прослідкована ще з періоду античності. Значний розвиток волонтерства відбувся у XX столітті, особливо після закінчення Першої та Другої світових воєн, коли виникла нагальна потреба залучення добровольців для надання допомоги постраждалим. У 70-х роках минулого століття така діяльність стала більш організованою, що сприяло створенню міжнародних волонтерських організацій, таких як "Корпус миру", "Лікарі без кордонів" тощо [1].

Волонтерський рух в Україні почав набирати значного розвитку після проголошення незалежності держави. Особливої актуальності волонтерська діяльність набула під час подій Помаранчевої революції 2004 року та Революції Гідності 2013–2014 років. Українські волонтери відіграли суттєву роль у підтримці Збройних Сил України, наданні допомоги постраждалим від бойових дій на сході країни, а також у заходах із протидії пандемії COVID-19.

Станом на зараз в Україні прийнято та діє Закон «Про волонтерську діяльність», яким регламентуються правові основи організації волонтерської діяльності та визначається статус волонтерів. Відповідно до положень цього закону, волонтерська діяльність базується на принципах законності, гуманності, рівності, добровільності, безоплатності та неприбутковості [2].

Волонтерська діяльність охоплює багато сфер суспільного життя: допомога вразливим категоріям населення, охорона здоров'я, культура, освіта, наука, спорт, захист навколишнього середовища, реагування на надзвичайні ситуації, а також підтримка різноманітних заходів національного та міжнародного рівня.

Організація волонтерства включає набір і координацію добровольців, планування й реалізацію проектів, а також співпрацю з партнерами та благодійниками. Для ефективної роботи важливо забезпечити підготовку і мотивацію волонтерів, створити їм сприятливі умови, а також налагодити якісну комунікацію між усіма учасниками процесу.

Сьогодні інформаційні технології відіграють ключову роль у розвитку волонтерства. Вони дозволяють швидше залучати нових учасників, оптимізують організаційні процеси, сприяють поширенню інформації про ініціативи, полегшують комунікацію та координацію.

Все більше популярності набувають спеціалізовані вебзастосунки та онлайн-платформи, які автоматизують рутинні завдання: реєстрацію волонтерів, пошук вакансій, розподіл обов'язків. Завдяки їм можна легко спілкуватися всередині проекту, відстежувати виконання завдань і формувати звітність.

Використання сучасних технологій підвищує ефективність та масштабованість волонтерських ініціатив, допомагає залучати ширшу аудиторію. Проте для цього організаціям необхідно мати відповідні технічні знання та партнерів [3].

Отже, предметна область волонтерської діяльності є надзвичайно важливою та актуальною в сучасному суспільстві. Волонтери роблять вагомий внесок у вирішення проблем соціальної сфери, гуманітарних та екологічних проблем, сприяють сталому розвитку громадянського суспільства. Застосування сучасних інформаційних технологій, наприклад спеціалізованих програмних комплексів та інформаційних систем, надає нові потужні можливості для організації та підтримки активного волонтерського руху, що в свою чергу дозволяє збільшити ефективність такого руху із залученням більшої кількості учасників.

Незважаючи на існуючі програмні рішення для волонтерської діяльності, все ще спостерігається потреба у створенні більш функціональних та адаптованих під галузеві потреби інформаційних систем. Такі системи мають враховувати специфіку волонтерської сфери, забезпечувати зручні інструменти для управління проектами, організації комунікації, мотивації волонтерів та інших ключових аспектів.

Процес розробки інформаційної системи для підтримки волонтерської діяльності має ґрунтуватися на аналізі низки спеціалізованих вимог і особливостей предметної області. Зокрема, система повинна забезпечувати ефективне управління волонтерськими проектами на всіх етапах: від планування до звітності й аналізу результатів виконання.

Однією з основних функцій інформаційної системи є можливість створення та публікації вакансій для волонтерів. Організатори проектів повинні мати змогу детально описувати суть роботи, вимоги до кандидатів, часові рамки й інші релевантні умови. Волонтери, своєю чергою, повинні мати доступ до актуальних вакансій, можливість фільтрування за різними параметрами (зокрема, локацією, сферою діяльності, необхідними навичками тощо) та подавати заявки на участь у проектах.

Для ефективної взаємодії між організаторами та волонтерами інформаційна система повинна включати засоби обміну повідомленнями, створення чатів або форумів. Це сприятиме оперативному вирішенню робочих питань, передачі досвіду, взаємообміну ідеями та підтримці командної взаємодії.

Важливою складовою інформаційної системи є модуль керування завданнями та розподілу ролей між учасниками. Організатори повинні мати інструменти для створення й призначення завдань, визначення термінів виконання, відстеження прогресу. Волонтери – отримувати персональні списки завдань, позначати їх виконання та формувати відповідні звіти щодо виконаної роботи [4].

Для мотивації та заохочення волонтерів, інформаційна система повинна містити у своєму складі спеціалізований вебзастосунок, що має елементи гейміфікації, наприклад, нарахування балів або відзнак (стікерів) за виконання певних завдань,

створення та відображення рейтингів найактивніших учасників тощо. Такий підхід сприятиме підвищенню зацікавленості волонтерів та буде стимулювати їх до більш активної участі у різноманітних заходах.

Функціонал генерації звітності та аналітики у інформаційній системі є важливою складовою. Організатори повинні мати доступ до деталізованих звітів щодо виконання проєктів, кількості залучених волонтерів, витраченого часу та інших ключових показників. Така інформація дає змогу ефективно оцінювати результати волонтерської діяльності, виявляти проблемні аспекти та приймати обґрунтовані управлінські рішення стосовно подальшого розвитку проєктів.

Особливу увагу необхідно приділити питанням безпеки й конфіденційності системи. Оскільки вона містить персональні дані волонтерів та іншу чутливу інформацію, потрібно впровадити надійні механізми захисту від несанкціонованого доступу, використовувати шифрування даних та дотримуватися чинного законодавства щодо захисту персональної інформації.

Крім цього, вебінтерфейс клієнтської частини інформаційної системи повинен бути зручним, інтуїтивно зрозумілим, сучасним і привабливим. Система має коректно функціонувати на комп'ютерах, планшетах та смартфонах, забезпечуючи комфортне користування для всіх категорій користувачів.

Необхідно також передбачити можливість інтеграції системи з іншими системами і сервісами, котрі можуть бути задіяні у волонтерській сфері. Зокрема, інтеграція з платформами для онлайн-пожертв сприятиме залученню додаткових коштів на реалізацію проєктів, а інтеграція із соціальними мережами – поширенню інформації про ініціативи та розширенню аудиторії волонтерів.

Таким чином, створення інформаційної системи для підтримки волонтерської діяльності потребує урахування комплексу функціональних і нефункціональних вимог, продиктованих особливостями сфери застосування. Створена інформаційна система має стати результативним інструментом для організації, координації та аналізу волонтерських проєктів, оптимізувати взаємодію між організаторами та

учасниками, підвищувати рівень мотивації, залученості та забезпечувати постійне вдосконалення процесів.

Розробка спеціалізованої інформаційної системи для волонтерської сфери є актуальним і перспективним завданням. Вона дозволить удосконалити організаційні процеси, підвищити ефективність роботи волонтерів та сприятиме сталому розвитку волонтерського руху, посилюючи його позитивний вплив на суспільство.

1.2 Огляд наявних аналогів та публікацій

Для комплексного аналізу актуальності й доцільності впровадження інформаційної системи підтримки волонтерської діяльності необхідно розглянути існуючі аналоги, а також проаналізувати науково-практичні публікації у відповідній галузі [5].

Аналіз існуючих рішень дозволяє виявити основні функціональні можливості, на які слід орієнтуватися під час розробки власної інформаційної системи. Водночас варто звернути увагу на недоліки та обмеження сучасних аналогів, зокрема, недостатню локалізацію під український ринок, обмеження у кастомізації процесів під потреби конкретних організацій, а також питання інтеграції з державними чи локальними інформаційними ресурсами. Знання сильних і слабких сторін аналогічних систем дає змогу адаптувати найкращі практики та уникнути повторення типових помилок у проєктуванні нової системи.

Серед сучасних програмних засобів, що сприяють розвитку волонтерства, можна виділити такі категорії: онлайн-платформи для пошуку волонтерських позицій, системи управління волонтерами, мобільні додатки для волонтерів тощо.

Однією з провідних міжнародних платформ для пошуку волонтерських можливостей є VolunteerMatch (<https://www.volunteermatch.org/>). Дана платформа забезпечує підбір проєктів і організацій для волонтерів за низкою параметрів, включаючи географічне розташування, сферу діяльності та необхідні навички. Організації можуть публікувати вакансії для волонтерів та здійснювати управління

заявками через інтерфейс, який відзначається простотою та інтуїтивною зрозумілістю (рис. 1.1).

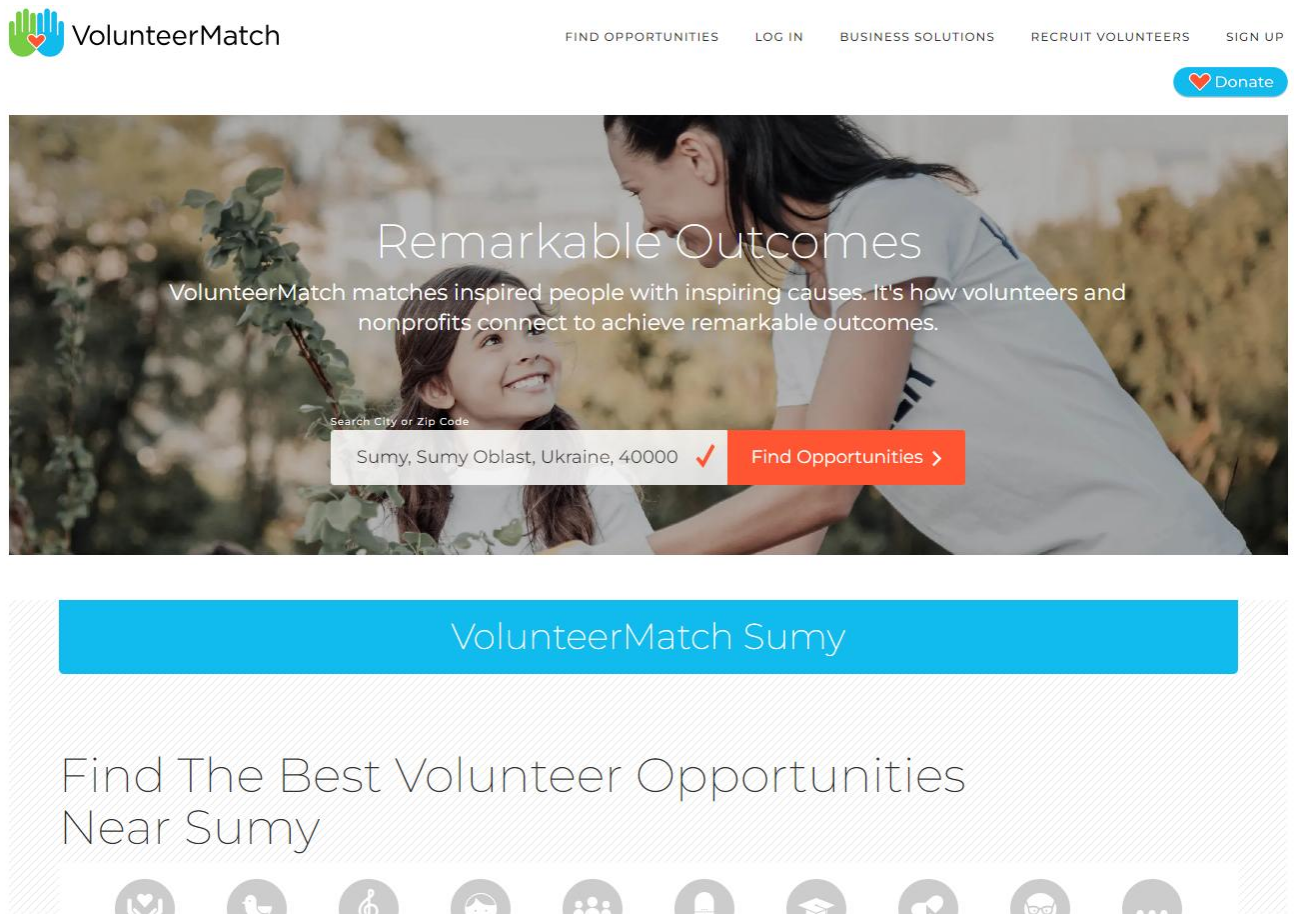


Рисунок 1.1 – Головна сторінка платформи VolunteerMatch

Ще одним відомим ресурсом є Idealist (<https://www.idealist.org/>), який надає можливість пошуку не лише волонтерських позицій, а також оплачуваних вакансій і стажувань у неприбуткових організаціях по всьому світу. Платформа також пропонує інструменти для створення організаційних профілів і розміщення вакансій (рис. 1.2).

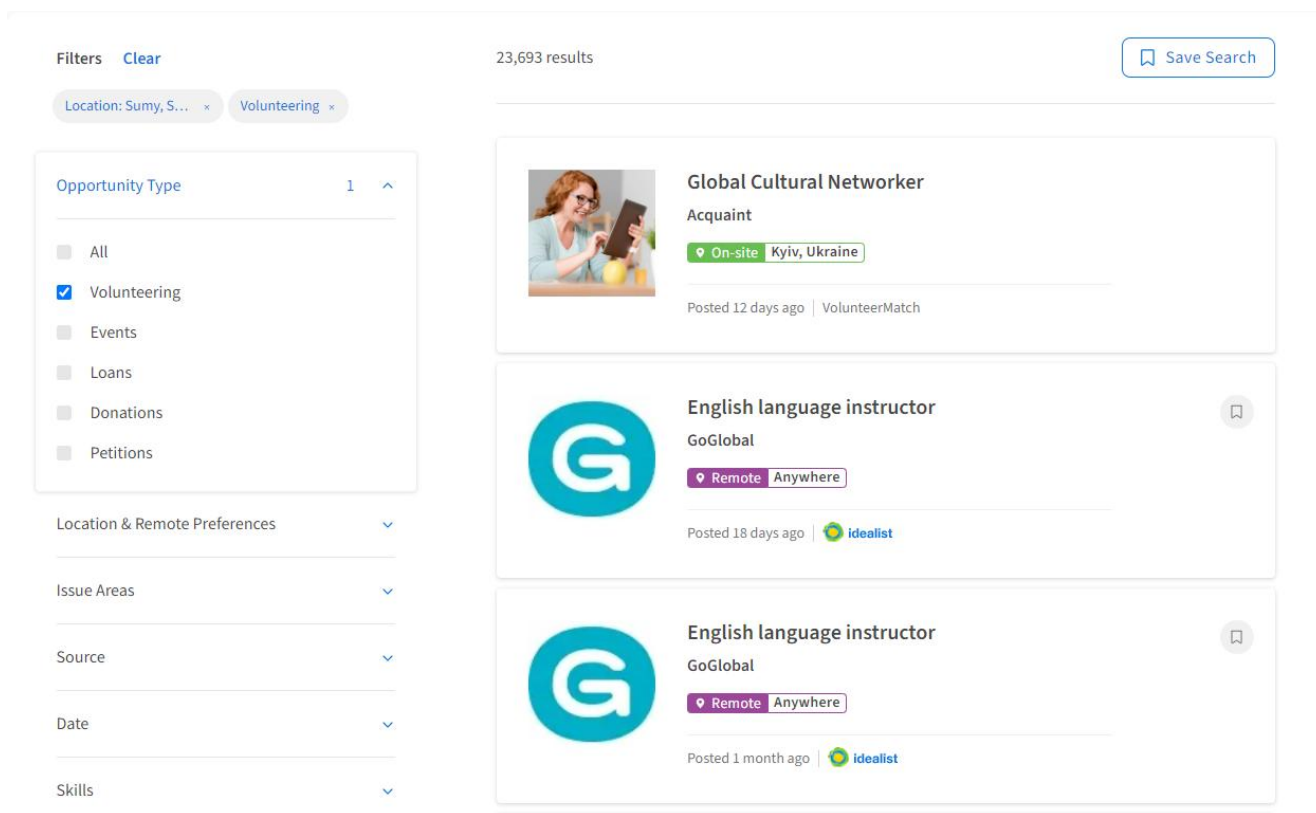


Рисунок 1.2 – Сторінка пошуку волонтерських вакансій на платформі Idealist

Окрім глобальних платформ, існують також регіональні та локальні сервіси для пошуку волонтерських можливостей. Наприклад, у Сполучених Штатах діє платформа AllForGood (<https://www.allforgood.org/>), яка централізовано збирає волонтерські вакансії з різних джерел і надає можливість здійснювати пошук за конкретними штатами та містами (див. рис. 1.3).

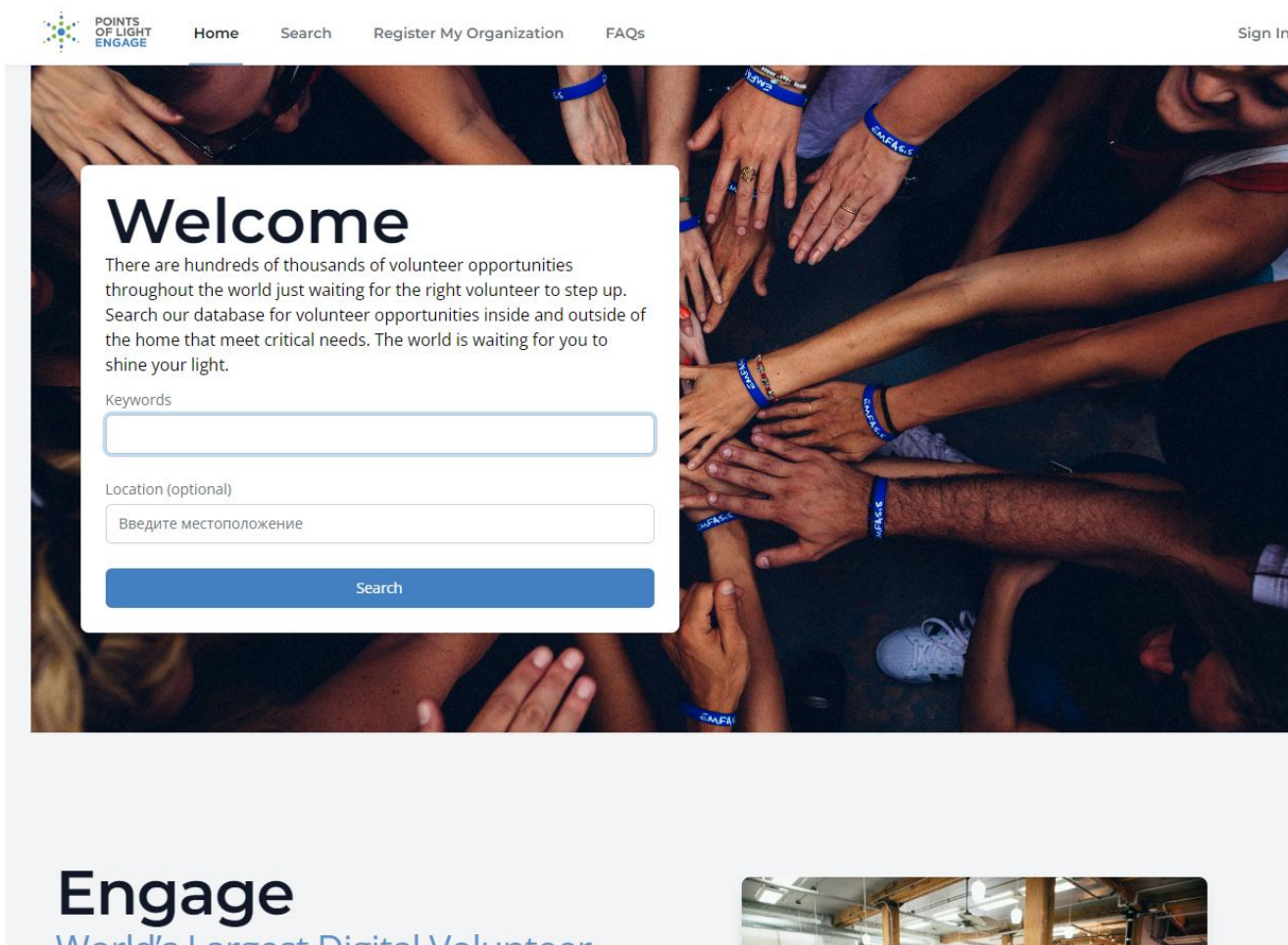


Рисунок 1.3 – Головна сторінка платформи AllForGood

Серед систем для управління волонтерами виділяється програмне забезпечення Volgistics (<https://www.volgistics.com/>), яке вважається одним із лідерів галузі. Це рішення дає змогу волонтерським організаціям ефективно автоматизувати процеси – від підбору та ознайомлення добровольців до розподілу завдань і створення звітів. Volgistics пропонує зручний веб-інтерфейс і може легко інтегруватися з іншими сервісами (див. рис. 1.4).

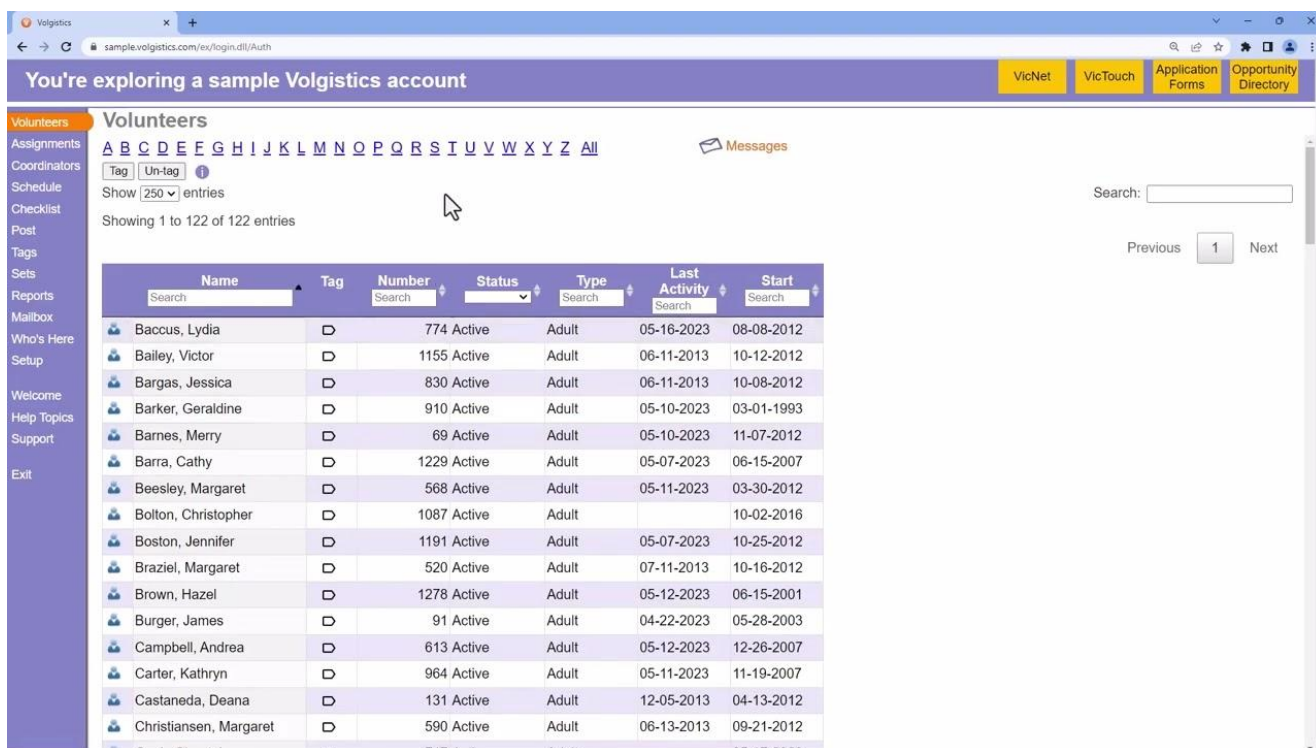


Рисунок 1.4 – Дашборд системи управління волонтерами Volgistics

Серед мобільних застосунків для волонтерів слід виділити платформу Golden (<https://www.goldenvolunteer.com/>). Цей застосунок надає можливість волонтерам знаходити відповідні ініціативи поблизу, вести облік відпрацьованих годин волонтерської діяльності та отримувати винагороди від партнерських організацій. Додаток Golden вирізняється сучасним дизайном і впроваджує елементи гейміфікації для підвищення мотивації користувачів (рис. 1.5).

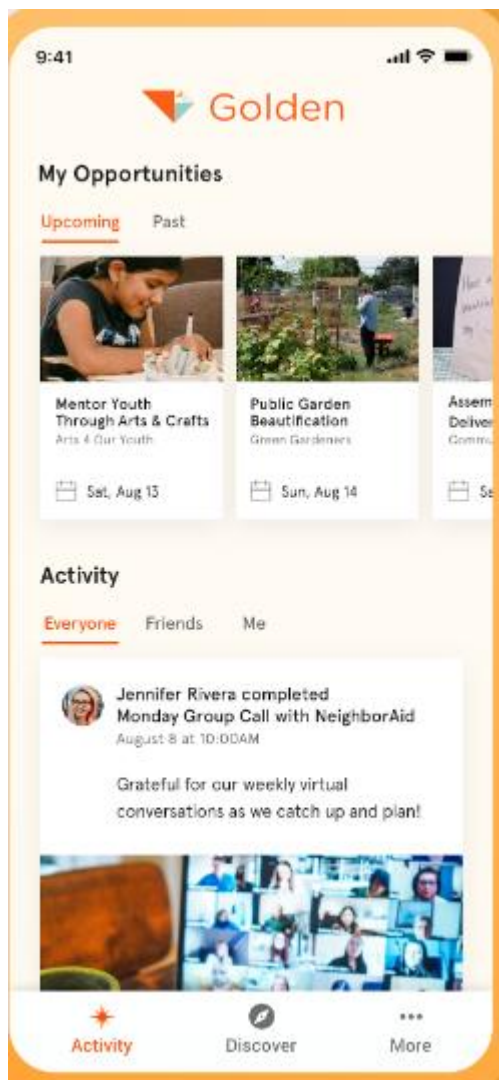


Рисунок 1.5 – Інтерфейс мобільного додатку Golden

InitLive (<https://www.initlive.com/>) є прикладом сучасного вебзастосунку для управління волонтерською діяльністю. Дана платформа призначена для організації роботи волонтерів під час проведення великих заходів, таких як фестивалі та конференції. InitLive забезпечує можливість набору волонтерів, їхнього розподілу за змінами та локаціями, організації комунікації в режимі реального часу та моніторингу активності учасників. Важливою перевагою системи є інтегрований мобільний додаток для волонтерів, який дає змогу отримувати завдання, відзначати їх виконання та здійснювати зв'язок з координаторами (рис. 1.6).



Рисунок 1.6 – Інтерфейс вебзастосунку InitLive для організаторів заходів

Таблиця 1.1 - Порівняльна характеристика існуючих рішень 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

Назва	Пошук вакансій	Реєстрація волонтерів	Управління волонтерами	Мобільний додаток	Особливості
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 проєктах pro bono

Аналіз аналогічних рішень демонструє, що наявні програмні рішення та платформи охоплюють різноманітні аспекти волонтерської діяльності, забезпечуючи інструменти для пошуку вакансій, рекрутингу та управління волонтерами. Водночас більшість із них не є універсальними та мають низку обмежень.

Серед основних проблем можна визначити наступне:

- орієнтація на глобальний ринок зі слабкою адаптацією до потреб локальних волонтерських спільнот;
- відсутність комплексних платформ, які б інтегрували функції пошуку вакансій, рекрутингу, онбордингу, управління та мотивації волонтерів;
- обмежена підтримка мобільних пристроїв та інтеграцій з популярними сервісами й системами, що використовуються організаціями;
- недоліки у підтримці розвитку волонтерських спільнот та ефективної комунікації між учасниками.

Таким чином, попри розмаїття цифрових рішень для підтримки волонтерської діяльності, існує необхідність створення інформаційної системи, яка б враховувала зазначені обмеження та пропонувала комплексний підхід для організації волонтерської роботи на локальному рівні.

За основу такого рішення доцільно взяти найкращі практики вже існуючих продуктів, зокрема:

- зручний та інтуїтивно зрозумілий інтерфейс для пошуку волонтерських вакансій і створення профілів користувачів;
- функціонал для рекрутингу, онбордингу та управління волонтерами (розподіл завдань, облік часу, комунікація, звітність);
- впровадження механік гейміфікації та інших інструментів мотивації для утримання волонтерів;
- розробка мобільного застосунка для отримання сповіщень, завдань та відстеження активності;

- інтеграція з популярними сервісами та системами (Google Календар, Trello, Slack тощо);
- створення можливостей для розвитку локальних спільнот, обміну досвідом і взаємної підтримки.

Реалізація зазначених підходів сприятиме розробці затребуваного й ефективного вебзастосування для підтримки та розвитку волонтерської діяльності на місцевому рівні.

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

Однією з основоположних робіт у вказаній галузі є стаття "Volunteer management information systems: Challenges and opportunities for the third sector" (Volda et al., 2011), опублікована в журналі "Nonprofit Management and Leadership". Автори здійснили аналіз впровадження систем управління волонтерами у різних неприбуткових організаціях, висвітливши головні проблеми та можливості, пов'язані з їх застосуванням. Зокрема, наголошено на необхідності адаптації таких систем до специфічних потреб організацій, забезпечення їх зручності для волонтерів і персоналу, а також інтеграції із внутрішньоорганізаційними інформаційними системами.

Дослідження "Volunteer retention through information systems: A case study" (Senseney et al., 2017), опубліковане у "Journal of Community Informatics", присвячене ролі інформаційних систем у процесах утримання волонтерів на прикладі конкретної неприбуткової організації. У роботі обґрунтовано, що використання спеціалізованої системи управління волонтерами сприяє покращенню комунікації, підвищує рівень залученості волонтерів та зменшує плинність кадрів.

Аспекти гейміфікації у волонтерському менеджменті розглядаються у статті "Gamification in volunteer management: A case study of online volunteer communities" (Lee et al., 2018), надрукованій в "International Journal of Human–Computer Interaction". Дослідники вивчили досвід інтеграції ігрових елементів в онлайн-волонтерських

спільнотах і встановили, що гейміфікація позитивно впливає на мотивацію й залученість волонтерів, особливо серед молоді [15].

Додатковим напрямом досліджень є визначення чинників, які впливають на готовність волонтерів використовувати інформаційні системи. В роботі "Factors influencing volunteers' intention to use information systems: A case of a non-profit organization" (Choi & Kim, 2018), опублікованій у "Journal of Computer Information Systems", зазначено, що на намір волонтерів користуватися системою впливають очікувана корисність, легкість використання, відповідність цінностям та досвіду волонтера, а також підтримка організації.

Підсумовуючи аналіз наукових публікацій, можна констатувати, що проведені дослідження у сфері розробки та впровадження інформаційних систем для волонтерського менеджменту підтверджують їх значущість та потенціал щодо підвищення ефективності діяльності волонтерських організацій. Одночасно акцентується на важливості ретельного проектування таких систем з урахуванням специфіки волонтерської роботи та необхідності забезпечення доступності й зручності для кінцевих користувачів.

Огляд наявних програмних рішень та профільних наукових джерел засвідчує актуальність і доцільність розробки вебзастосунку для підтримки волонтерської діяльності. Такий продукт має потенціал стати комплексним рішенням, орієнтованим на потреби локальних волонтерських спільнот і організацій, що сприятиме підвищенню ефективності їх роботи, розширенню можливостей залучення та утримання волонтерів.

1.3 Постановка задачі

На основі здійсненого аналізу предметної області волонтерської діяльності та огляду сучасних програмних рішень сформовано мету та ключові вимоги до розробки інформаційної системи.

Метою створення даної інформаційної системи є підвищення ефективності та покращення організації волонтерської діяльності на локальному рівні шляхом забезпечення комплексної підтримки процесів пошуку волонтерських вакансій, набору і управління волонтерами, а також комунікації та мотивації учасників волонтерського руху [14].

Для досягнення зазначеної мети система має виконувати такі основні функції:

- надання волонтерським організаціям можливості створювати профілі та публікувати вакансії з детальним описом вимог, обов'язків і умов співпраці;
- забезпечення потенційним волонтерам зручного та результативного пошуку вакансій із використанням різноманітних фільтрів (за місцем розташування, сферою діяльності, необхідними навичками тощо);
- створення волонтерами персональних профілів, що містять інформацію про навички, досвід, інтереси й побажання щодо участі у волонтерській діяльності;
- підтримка процесу подачі заявок на вакансії та їх обробки організаціями, включаючи можливість проведення онлайн-співбесід та обміну документами;
- забезпечення організаціям інструментів для управління волонтерами, включаючи розподіл завдань, облік відпрацьованого часу, формування звітності тощо;
- впровадження ефективних засобів комунікації між волонтерами й організаторами: обмін повідомленнями та сповіщеннями, організація заходів, поширення інформації;
- стимулювання мотивації та залучення волонтерів за допомогою гейміфікації (система балів, відзнаки, рейтинги), а також реалізація інструментів навчання та розвитку;
- сприяння формуванню та розвитку локальних волонтерських спільнот, які отримують можливості для спілкування, обміну досвідом і взаємної підтримки;

- інтеграція з іншими популярними сервісами й системами, які використовуються волонтерськими організаціями (календарі, системи управління проектами, месенджери тощо);
- забезпечення доступності та зручності користування як через веб-інтерфейс, так і через мобільний застосунок для основних платформ (iOS, Android).

Для успішної реалізації поставлених задач, інформаційна система повинна 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.

Крім того, під час розроблення інформаційної системи необхідно враховувати особливості волонтерської діяльності й потреби користувачів. Особливу увагу слід приділити зручності й простоті використання додатку, оскільки серед волонтерів можуть бути особи різного віку й рівня технічної підготовки [6].

Важливо також забезпечити гнучкість налаштувань застосунку відповідно до вимог конкретних волонтерських організацій, беручи до уваги відмінності в масштабах, сферах діяльності та організаційних процесах.

Проектування й реалізацію програмного продукту доцільно здійснювати із застосуванням сучасних підходів і технологій веброзробки, таких як адаптивний дизайн, PWA (Progressive Web Apps), мікросервісна архітектура, контейнеризація, хмарні обчислення тощо. Це дасть змогу забезпечити високу якість, продуктивність і масштабованість рішення [13].

Для успішної реалізації проекту необхідне ретельне планування, яке включає деталізований аналіз вимог, розроблення архітектури, вибір технологічного стеку, складання робочого графіка й розподіл обов'язків у команді розробників. Не менш важливо організувати ефективну комунікацію з потенційними користувачами — волонтерськими організаціями — для отримання зворотного зв'язку та перевірки прийнятих рішень.

Очікується, що впровадження інформаційної системи з визначеними функціональними можливостями суттєво підвищить ефективність і зручність організації волонтерської діяльності, оптимізує процеси пошуку й взаємодії волонтерів з організаціями, а також покращить комунікацію й мотивацію учасників волонтерського руху.

У підсумку, це сприятиме розвитку волонтерства на місцевому рівні, залученню ширшого кола осіб до суспільно корисної діяльності й посиленню позитивного впливу волонтерських ініціатив на життя громад.

Таким чином, постановка завдання щодо розроблення інформаційної системи для підтримки волонтерської діяльності є актуальною й обґрунтованою, а реалізація відповідного проєкту матиме значну практичну цінність для подальшого розвитку волонтерського руху в Україні.

Висновки до розділу 1

У першому розділі дипломної роботи здійснено детальний аналіз предметної області волонтерської діяльності, а також розглянуто наявні програмні засоби підтримки волонтерських організацій та волонтерів. На основі отриманих результатів було сформовано завдання щодо розробки спеціалізованої інформаційної системи.

Волонтерська діяльність відіграє ключову роль у розвитку громадянського суспільства, допомагаючи вирішувати соціальні, гуманітарні й екологічні питання. Ця діяльність ґрунтується на добровільності, безкорисливості й орієнтована на допомогу тим, хто її потребує. В Україні волонтерський рух активно розширюється, залучаючи дедалі більше людей, особливо молодь.

Аналіз показав, що для ефективної організації волонтерської діяльності необхідне використання сучасних ІТ-рішень. Завдяки їм можна автоматизувати процеси пошуку і залучення волонтерів, координувати проєкти та спілкування між учасниками.

Огляд існуючого ринку програмних продуктів виявив низку онлайн-платформ і систем управління волонтерською діяльністю, які пропонують такі функції, як пошук вакансій, реєстрація, розподіл завдань тощо. Серед них – VolunteerMatch, Idealist, Volgistics, InitLive й інші.

Водночас існуючі рішення мають обмеження: більшість орієнтовані на глобальний ринок і не враховують специфіку локальних волонтерських спільнот. Чимало продуктів не поєднують усі аспекти організації роботи – від залучення до управління проєктами та мотивації [12].

Наукові дослідження у сфері інформаційних систем для волонтерського менеджменту підтверджують актуальність теми. Дослідники наголошують на важливості адаптації таких систем до конкретних організацій, інтеграції з іншими сервісами та зручності користування, а також потенціалі гейміфікації й соціальних функцій для підвищення залученості.

В результаті аналізу сформульовано завдання щодо створення інформаційної системи підтримки волонтерства на локальному рівні. Очікувана мета – зробити організацію волонтерської діяльності максимально ефективною та комфортною, забезпечивши комплексну підтримку всіх основних процесів – від пошуку вакансій до управління проектами й мотивації.

Виділено головні задачі майбутньої системи: розміщення вакансій, пошук і відбір волонтерів, управління завданнями, ведення обліку часу, організація комунікації, створення звітів, застосування гейміфікації, інтеграція з іншими ресурсами тощо.

Формалізовано як функціональні, так і нефункціональні вимоги до інформаційної системи – вони враховують особливості волонтерської діяльності та потреби користувачів. Серед основних вимог: безпека даних, висока продуктивність, простота використання, кросплатформність, масштабованість тощо.

Реалізація системи із зазначеним набором функцій дозволить значно спростити й покращити організацію волонтерської роботи, залучити більше охочих до волонтерства, підвищити координацію та мотивацію учасників. Це сприятиме зміцненню ролі волонтерського руху у вирішенні соціальних проблем.

РОЗДІЛ 2. РОЗРОБКА ПРОЄКТНИХ РІШЕНЬ ПО ІНФОРМАЦІЙНІЙ СИСТЕМІ

2.1 Функціональна модель ІС

Для розробки інформаційної системи 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 інформаційної системи 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 інформаційної системи 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 інформаційної системи 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.

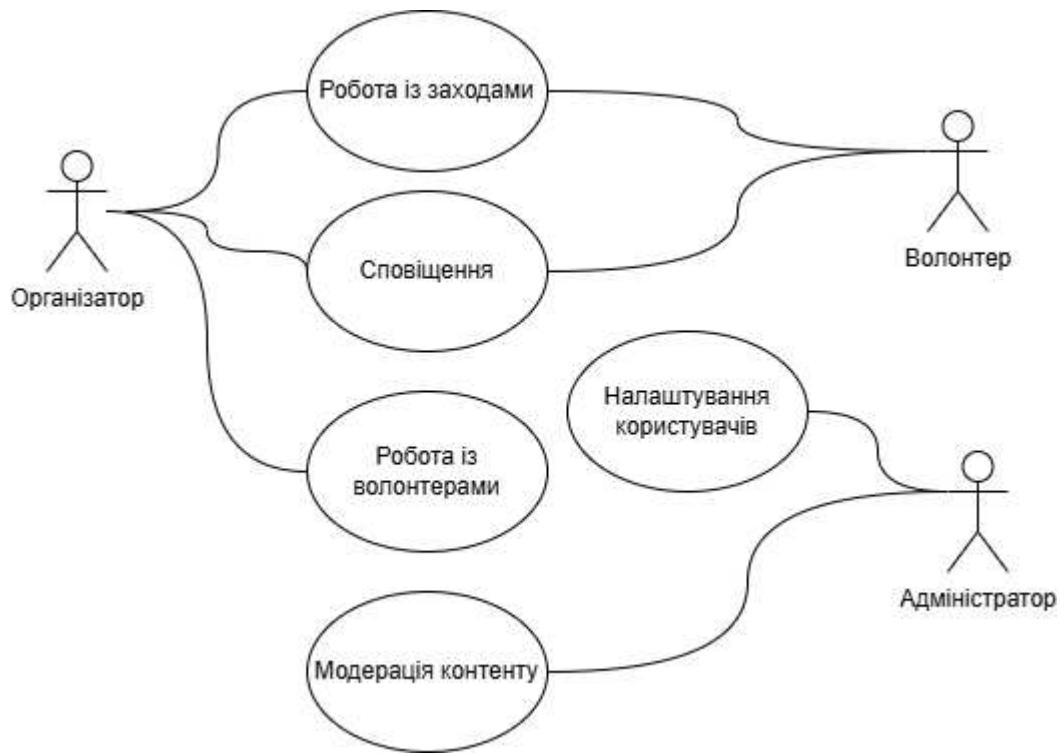


Рисунок 2.1 – Діаграма варіантів використання інформаційної системи

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 системи, 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 інформаційної системи 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 великої кількості таблиць, серед яких можна виділити три основних: "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 посиляється на ідентифікатор заходу з таблиці "volunteering_activity";

– activity_application_fk (bigint) - зовнішній ключ, що також посиляється 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 посиляються 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 інформаційної системи 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 Java.

2.3 Вибір технологій та архітектури ІС

Для розробки інформаційної системи 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 корпоративних веб-застосунків та інформаційних систем. 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 принцип «налаштування за замовчуванням» важливіші за конфігурацію 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 ORM фреймворк для Java, який значно спрощує роботу з базами даних. Він 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, ліниве 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 інформаційної системи 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 збирання – Maven. Maven – це інструмент для збирання 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 інформаційної системи 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 інформаційної системи 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 інформаційної системи 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 надійну, масштабовану та безпечну інформаційну систему із вебзастосунком для підтримки волонтерської діяльності. Використання сучасних фреймворків та інструментів значно спрощує процес розробки та забезпечує високу якість кінцевого продукту.

Наступним етапом буде безпосередня реалізація інформаційної системи згідно з обраними технологіями та архітектурою, а також її тестування та аналіз результатів.

Висновки до розділу 2

У другому розділі дипломної роботи було проведено проектування та моделювання інформаційної системи 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 інформаційної системи. 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 інформаційної системи 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 інформаційної системи. 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 фреймворк 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 інформаційної системи 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 інформаційної системи. 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 інформаційної системи.

Таким чином, у другому розділі було проведено ґрунтовну роботу з проектування та моделювання інформаційної системи 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 майбутньої інформаційної системи.

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 частин інформаційної системи та вебзастосунку.

Проведене проектування та моделювання є надійною основою для успішної реалізації інформаційної системи для підтримки волонтерської діяльності, що

відповідатиме всім визначеним вимогам та забезпечить ефективну взаємодію волонтерів, організаторів та адміністраторів у рамках єдиної системи.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ ІС

3.1 Особливості програмної реалізації

Інформаційну систему 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 корпоративних систем.

Архітектура корпоративного 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 проєкті моделі включають класи User, VolunteeringActivity та VolunteeringApplication, які відповідають таблицям бази даних;
- **view (представлення):** відповідає за відображення даних користувачеві. У проєкті використовується шаблонізатор Thymeleaf для створення динамічних HTML-сторінок;
- **controller (контролер):** обробляє запити користувача, взаємодіє з моделями та оновлює представлення. Контролери в проєкті 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 проєкті 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 інформаційній системі 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.

Вхід

Ваш Email

Пароль

Увійти

[Створити новий обліковий запис](#)

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.

Вхід

Ваш Email

Пароль

Увійти

[Створити новий обліковий запис](#)

Рисунок 3.2 – Сторінка авторизації користувача

Реєстрація

Ваше ім'я

Ваше прізвище

Ваш номер мобільного телефону

Ваш Email

Ваш пароль

Зареєструватися

[Вже маєте обліковий запис? Увійти](#)

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).

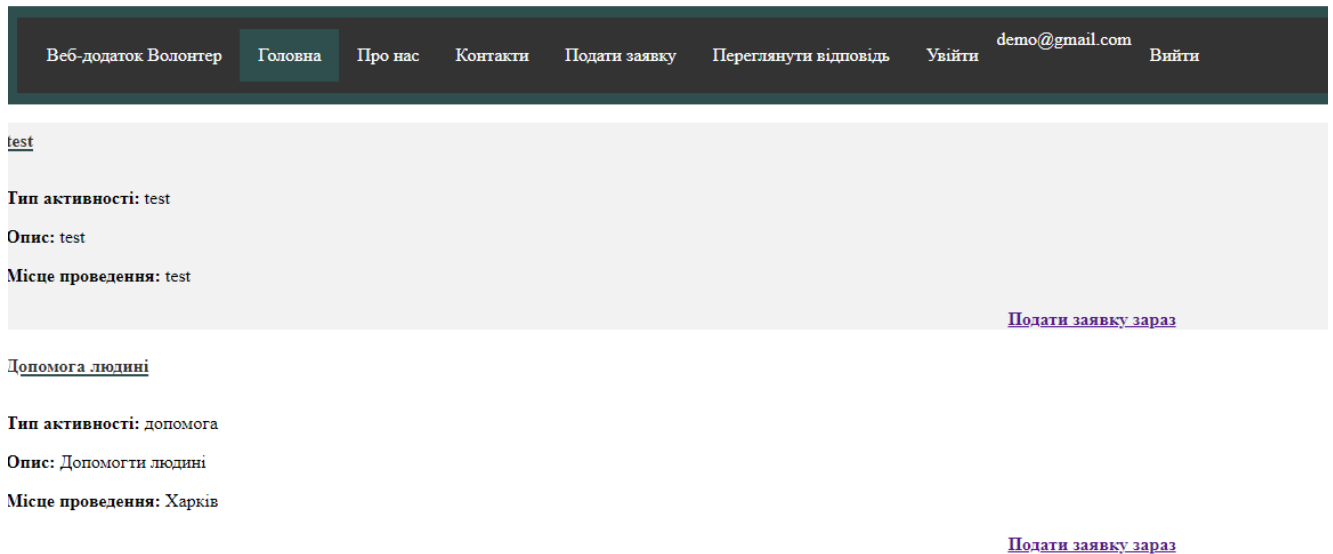


Рисунок 3.4 – Особиста сторінка користувача (дашборд)

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).

est	
Тип активності: test	
Опис: test	
Місце проведення: test	Подати заявку зараз
Допомога людині	
Тип активності: допомога	
Опис: Допомогти людині	
Місце проведення: Харків	Подати заявку зараз

Рисунок 3.5 – Список доступних волонтерських заходів

Форма заявки

test

Імена волонтерів

Вік волонтерів

0

Email волонтерів

Номер телефону волонтерів

Адреса волонтерів

Адреса волонтерів

Виберіть файл:

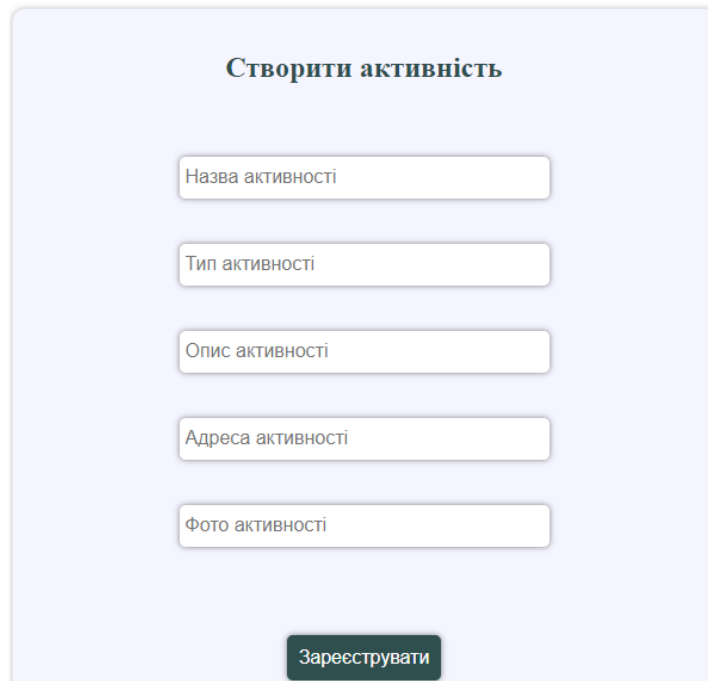
Выберите файл

Файл не выбран

ПОДАТИ ЗАЯВКУ

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.



The image shows a web form titled "Створити активність" (Create activity) on a light blue background. The form contains five input fields stacked vertically: "Назва активності" (Activity name), "Тип активності" (Activity type), "Опис активності" (Activity description), "Адреса активності" (Activity address), and "Фото активності" (Activity photo). At the bottom of the form is a dark green button with the text "Зареєструвати" (Register).

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.

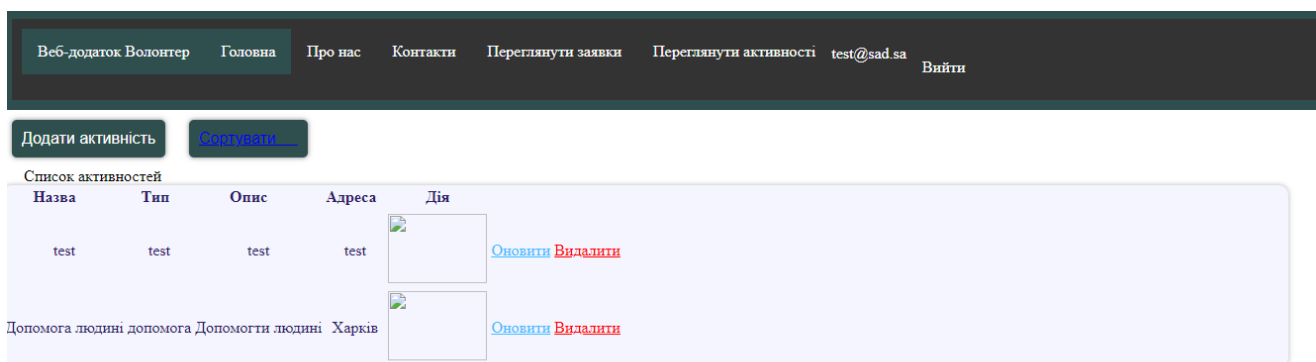


Рисунок 3.8 – Панель адміністрування

Окрім основних функцій, вебзастосунок інформаційної системи 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 вебзастосунк інформаційної системи 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 поводить себе 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 інформаційної системи підтримки волонтерської діяльності. Було описано особливості програмної реалізації, представлено інтерфейс користувача та проаналізовано результати тестування.

Особливості програмної реалізації інформаційної системи 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 шаблонні рішення та автоконфігурацію.

Робота з базою даних PostgreSQL із використанням 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 інформаційної системи 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 інформаційної системи 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 інформаційної системи 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.

РОЗДІЛ 4. РОЗРАХУНОК ЕКОНОМІЧНОГО ЕФЕКТУ ВІД ВПРОВАДЖЕННЯ ІС

4.1 Вступ до проблеми

Інформаційні системи широко використовуються багатьма організаціями України різної форми власності. Ці системи розповсюджуються все більше і більше, змінюючись відповідно до вимог замовників та модернізуючись з плином часу.

Системи управління контентом в цілому не є принципово новою технологією, однак як парадигма створення інформаційних систем та надання послуг у сфері ІТ вони знаходяться лише на початку свого шляху. Безперечно, з розвитком цієї технології все більше і більше компаній використовуватимуть системи управління контентом для створення власної комп'ютерної інфраструктури, вирішення обчислювальних задач та ін.

Створення інформаційної системи вимагає одноразових витрат на її розробку та придбання необхідних технічних засобів, а також поточних витрат, пов'язаних з функціонуванням підсистеми. Економічний ефект від експлуатації системи визначається з урахуванням витрат на її експлуатацію. Відношення економічного ефекту до витрат на створення системи характеризує економічну ефективність капітальних вкладень. Економічні показники визначаються за діючими на момент розрахунку цінами, тарифами та ставками заробітної плати.

4.2 Розрахунок вартості створення інформаційної системи волонтерської діяльності

Загальні витрати (на проектування, реалізацію та впровадження системи)

$$З = C_{п-р} + C_{тз} + C_{пз} + C_{впр},$$

де $C_{п-р}$ – витрати на проектування і реалізацію;

$C_{тз}$ – витрати на придбання або модернізацію обчислювальної техніки, периферійних пристроїв, засобів зв'язку, допоміжного устаткування та оргтехніки (з

урахуванням витрат на транспортування, монтаж, налагодження і запуск), виробничо-господарського інвентарю;

$C_{пз}$ – витрати на придбання та інсталяцію програмного забезпечення (ПЗ), необхідного для розробки;

$C_{впр}$ – витрати на впровадження системи (навчання персоналу тощо).

Витрати на проектування та реалізацію розраховуються за формулою

$$C_{п-р} = C_з + C_{від} + C_м + C_н,$$

де $C_з$ – заробітна плата розробників;

$C_{від}$ – відрахування на соціальні заходи (відповідно до чинного законодавства (становить 47,5% від фонду оплати праці);

$C_м$ - експлуатація та амортизація, $C_м = Z_{експл} + Z_{амор}$;

$C_н$ – накладні витрати у розмірі 50-150% від витрат на оплату праці (охоплюють оплату праці управлінського персоналу з нарахуваннями; оплату службових відряджень; оплату консультаційно-інформаційних послуг; оплату ремонту і техобслуговування основних фондів, крім персональних комп'ютерів (ПК)).

1) Розробка підсистеми вимагає 3 місяці (експертна оцінка, яка базується на досвіді розробки систем такого класу). Заробітна платня розробників за місяць дорівнює 12000 грн (відповідно до сучасних розцінок та попиту на спеціалістів такого профілю). Кількість розробників – одна людина.

$$C_з = 1 * 12000 * 3 = 36000 \text{ грн.}$$

$$C_{від} = 36000 * 0,475 = 17100 \text{ грн.}$$

2) Витрати на амортизацію ЕОМ, на яких виконується розробка:

$$Z_{амор} = (Ц_{еом} / T_{сл}) * (T_{разр} / 12),$$

де $Ц_{еом}$ = 20000 грн. – балансова вартість ЕОМ;

$T_{сл}$ = 5 років – термін служби ЕОМ;

$T_{разр}$ = 0,5 року – час, необхідний для розробки системи, виражений в місяцях, і тому ділиться на 12, щоб одержати число років.

$$З_{\text{амор}} = (20000/5) * 0,5 = 2000 \text{ (грн.)};$$

3) витрати на експлуатацію ЕОМ, на якій виконується розробка, полягають в оплаті споживаної нею електричної енергії:

$$З_{\text{експл}} = P_{\text{ЕОМ}} * T_{\text{разр}} * N_{\text{рд}} * N_{\text{рч}} * E_{\text{ст}},$$

де $P_{\text{ЕОМ}} = 0,4$ кВт/год – потужність ЕОМ;

$T_{\text{разр}} = 3$ місяці – тривалість розробки;

$N_{\text{рд}} = 21$ день – число робочих днів у місяці;

$N_{\text{рч}} = 8$ годин – число годин у робочому дні;

$E_{\text{ст}} = 0.9$ грн – вартість 1 кВт електроенергії

$$З_{\text{експл}} = 0,4 * 3 * 21 * 8 * 0,9 = 181,44 \text{ (грн.)}$$

$$C_{\text{м}} = 2000 + 181,44 = 2181,44 \text{ (грн.)}$$

$$C_{\text{н}} = 36000 * 0,5 = 18000 \text{ (грн.)}$$

$$C_{\text{п-р}} = 36000 + 17100 + 2181,44 + 18000 = 73281,44 \text{ (грн.)}$$

Витрати на програмне забезпечення не включаються до даного кошторису, оскільки у функціонуванні системи передбачається застосування безкоштовного програмного забезпечення з відкритим вихідним кодом. Витрати на апаратне забезпечення становлять 18594 грн, що охоплює придбання серверного обладнання, комутаційних пристроїв та необхідних матеріалів. До складу витрат, пов'язаних із впровадженням системи, належать також витрати на навчання персоналу, монтаж і налаштування мережі, придбання устаткування, а також витрати на програмне забезпечення.

Витрати на впровадження системи приведені в таблиці 4.1.

Таблиця 4.1 – Витрати на впровадження системи.

Найменування інвестицій	Сума, грн
Вартість устаткування	18594,00
Вартість програмного забезпечення	-
Вартість підготовки кадрів	500,00

Монтаж і налаштування мереж	300,00
Разом на впровадження системи	19394,00

Таким чином, загальні витрати на проектування, реалізацію та впровадження системи складають:

$$З = C_{п-р} + C_{тз} + C_{пз} + C_{впр} = 73281,44 + 19394 = 92675,44 \text{ (грн.)}$$

Витрати на створення та експлуатацію системи в перший рік експлуатації

$$З_{се} = З + З_{ер},$$

де $З_{ер}$ – експлуатаційні витрати на технічне забезпечення за рік.

$$З_{ер} = A_{об} + З_{м} + З_{е},$$

де $A_{об}$ – амортизаційні відрахування на устаткування;

$З_{м}$ – витрати на основні і допоміжні матеріали;

$З_{е}$ – витрати на електроенергію.

Амортизаційні відрахування на устаткування складають 60% балансової вартості в рік (за чинним законодавством)

$$A_{об} = 19394 * 0,6 = 11636,4 \text{ (грн.)}$$

В масштабі підприємства річні витрати на основні та допоміжні матеріали (флеш-накопичувачі, папір тощо) визначаються в розмірі 3% вартості основного устаткування

$$З_{м} = 19394 * 0,03 = 581,82 \text{ (грн.)}$$

Річний обсяг робіт ПК у нормо-годинах

$$\Phi_{м} = 253,3 * T_{с},$$

де $T_{с}$ – середнє місячне завантаження устаткування (близько 5 годин); 253,3 – середня кількість робочих днів у році.

$$\Phi_{м} = 253,3 * 5 = 1266,5 \text{ (годин.)}$$

Витрати на електроенергію під час 1266,5 годин роботи устаткування в рік при потужності ЕОМ 0,4 кВт/год та ціні 9 грн за 1 кВт електроенергії складуть

$$З_{е} = 1266,5 * 9 * 0,4 = 4559,4 \text{ (грн.)}$$

Експлуатаційні витрати на технічне забезпечення за рік

$$З_{\text{ер}} = 11636,4 + 581,82 + 4559,4 = 12674,16 \text{ (грн.)}$$

Отже, у перший рік витрати на створення та експлуатацію системи складуть

$$З_{\text{се}} = 92675,44 + 12674,16 = 105349,6 \text{ (грн.)}$$

4.3 Розрахунок економічної ефективності

Економічна ефективність використання інформаційної системи полягає у наступному:

- зменшення часу на обслуговування системи за рахунок її відмовостійкості та автоматичного масштабування;
- можливість надання користувачам інформації в оренду.

Це зменшення відбудеться за рахунок зменшення складності користування системою, збільшення надійності, стабільності та рівня безпеки. Визначимо, що це зменшення складе 50% від сучасного.

Однак, не піддається прямій грошовій оцінці зменшення кількості помилкових та необережних рішень, підвищення оперативності керування, поліпшення організації роботи та своєчасне отримання необхідної інформації про пріоритетність варіантів рішень тощо.

Використання даної системи дозволяє вивільнити 0,1 робочого часу. Так як з системою працює біля 50 користувачів, то система умовно вивільнить 5 робітників.

Визначимо пряму економічну ефективність, ґрунтуючись на тому, що впровадження системи вивільнить 5 робітників. Річна платня за рахунок вивільнення робочого часу складе

$$\Delta C_n = 5000 * 5 * 12 - 12674,16 = 287325,84 \text{ (грн.)}$$

Річний економічний ефект розраховується за наступною формулою:

$$E_{\text{год}} = \Delta C_n - E_n * k,$$

де E_n – нормативний коефіцієнт економічної ефективності капітальних вкладень у галузь і для обчислювальної техніки приймається 0,5;

k – додаткові капітальні вкладення з урахуванням витрат на проектування, створення і функціонування системи.

$$E_{\text{год}} = 287325,84 - 0,5 * 105349,6 = 234651,04 \text{ (грн.)}$$

Строк окупності підсистеми

$$T = k / \Delta C_n$$

Тобто

$$T = 105349,6 / 287325,84 = 0,37 \text{ (року)}$$

Таким чином, витрати на створення розподіленої інформаційної системи окупляться протягом 0,37 року.

Висновки до розділу 4

В результаті розрахунків витрати на створення системи обліку робочого часу (проектування, реалізація і впровадження) складуть 92675,44 грн; експлуатаційні витрати щорічно складатимуть 12674,16 грн. Загалом в перший рік витрати на модернізацію й експлуатацію складуть 105349,6 грн.

Вивільнюється 5 умовних фахівці.

Вивільнена річна зарплатня працівників складає 287325,84 грн.

Витрати на створення системи окупляться протягом 0,37 року.

За отриманими значеннями річного економічного ефекту та строку окупності можливо зробити висновок про економічну вигідність і обґрунтованість створення інформаційної системи. Слід зазначити, що підвищити економічну ефективність цієї системи можна за рахунок запропонування стороннім користувачам систему як продукт. Таким чином можна буде скоротити строк, за який окупляться витрати на створення системи, а також здобути прибуток у вигляді продажу продукту.

РОЗДІЛ 5. ОХОРОНА ПРАЦІ

Вступ

Охорона праці є невід'ємною частиною організації трудової діяльності в будь-якій сфері. Особливої актуальності питання охорони праці набуває в галузі інформаційних технологій, де фахівці значну частину робочого часу проводять за комп'ютером. Розробка вебзастосунків, як один із напрямків діяльності ІТ-спеціалістів, також передбачає тривалу роботу з персональними комп'ютерами, моніторами та іншими технічними засобами.

Недотримання вимог охорони праці при роботі з комп'ютерним обладнанням може призвести до виникнення різноманітних професійних захворювань, таких як синдром зап'ястного каналу, комп'ютерний зоровий синдром, захворювання опорно-рухового апарату тощо. Крім того, неправильна організація робочого місця, недостатнє освітлення, незадовільні параметри мікроклімату у виробничих приміщеннях можуть негативно впливати на самопочуття та працездатність розробників вебзастосунків.

Саме тому дотримання вимог охорони праці та створення належних умов для здоров'я і безпеки фахівців, задіяних у розробці вебзастосунків, є важливим завданням для керівників підприємств та організацій. Адже турбота про здоров'я працівників не лише сприяє покращенню їхнього самопочуття та зменшенню ризиків виникнення професійних захворювань, але й підвищує продуктивність праці та якість кінцевого продукту.

У цьому розділі дипломної роботи, присвяченої розробці вебзастосунку для підтримки волонтерської діяльності, буде розглянуто основні аспекти охорони праці в галузі інформаційних технологій, зокрема при проектуванні та розробці вебзастосунків. Буде приділено увагу нормативно-правовим актам, що регулюють питання охорони праці, вимогам до організації робочого місця розробника, безпечним

умовам праці та мінімізації впливу шкідливих факторів, режиму праці та відпочинку, а також особливостям заходів з електробезпеки.

5.1. Нормативно-правові акти, що регулюють питання охорони праці при роботі з персональними комп'ютерами та іншими пристроями для розробки вебзастосунків

Питання охорони праці при роботі з персональними комп'ютерами (ПК) та іншими пристроями, що використовуються для розробки вебзастосунків, регулюються низкою нормативно-правових актів України. Основним документом, який встановлює вимоги безпеки та захисту здоров'я працівників під час роботи з екранними пристроями, є Наказ Міністерства соціальної політики України «Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями» від 14.02.2018 № 207 [37]. Цей наказ поширюється на всіх суб'єктів господарювання незалежно від форм власності та організаційно-правової форми, які у своїй діяльності здійснюють роботу, пов'язану з використанням екранних пристроїв.

Згідно з цим документом, роботодавець повинен забезпечити відповідність робочих місць працівників ергономічним, організаційно-технічним та санітарно-гігієнічним вимогам. Зокрема, роботодавець зобов'язаний проінформувати працівників про умови праці та наявність на їх робочих місцях небезпечних і шкідливих виробничих факторів, які ще не усунуто. Також роботодавець повинен забезпечити проведення інструктажів з питань охорони праці та пожежної безпеки при роботі з екранними пристроями.

Окрім вищезазначеного наказу, питання охорони праці при роботі з ПК регулюються і іншими нормативними документами. Так, ДСанПіН 3.3.2-007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» [38] встановлює санітарно-гігієнічні вимоги до

умов праці та організації робочих місць з ВДТ ЕОМ, вимоги до виробничого середовища, освітлення, рівня шуму тощо.

Вимоги електробезпеки при експлуатації ПК та периферійних пристроїв визначаються такими нормативними актами, як Правила улаштування електроустановок (ПУЕ), Правила технічної експлуатації електроустановок споживачів (ПТЕЕС), Правила безпечної експлуатації електроустановок споживачів (ПБЕЕС). Зокрема, згідно з п. 1.7.110 ПУЕ, конструкція та розміщення комп'ютерного обладнання має відповідати вимогам електробезпеки, а також забезпечувати зручність його обслуговування.

Дотримання вимог НПАОП 0.00-7.15-18 «Вимоги щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями», НПАОП 40.1-1.21-98 «Правила безпечної експлуатації електроустановок споживачів», а також інших державних стандартів та санітарних норм є обов'язковим для всіх роботодавців та працівників, які використовують ПК та інші пристрої для розробки вебзастосунків. Лише комплексний підхід та неухильне виконання вимог нормативно-правових актів з охорони праці дозволить створити безпечні та комфортні умови праці для ІТ-фахівців, мінімізувати ризики виникнення професійних захворювань та травматизму.

5.2. Вимоги до організації робочого місця розробника вебзастосунків

Організація робочого місця розробника вебзастосунків відіграє ключову роль у забезпеченні безпечних та комфортних умов праці, а також у підвищенні продуктивності та ефективності роботи фахівця. При облаштуванні робочого простору необхідно враховувати низку ергономічних, санітарно-гігієнічних та організаційно-технічних вимог, які регламентуються відповідними нормативно-правовими актами.

Одним із основних документів, що встановлює вимоги до робочого місця з персональним комп'ютером, є ДСТУ ISO 9241-5:2004 «Ергономічні вимоги до роботи з відеотерміналами в офісі. Частина 5. Вимоги до компонування робочого місця та до

робочої пози» [39]. Згідно з цим стандартом, робоче місце розробника вебзастосунків має бути достатньо просторим для зміни робочого положення та забезпечувати зручний доступ до всіх необхідних елементів обладнання.

Робочий стіл повинен мати достатню площу для розміщення комп'ютера, периферійних пристроїв, документації та інших необхідних матеріалів. Рекомендовані розміри робочої поверхні столу становлять: ширина - не менше 1200 мм, глибина - не менше 800 мм. Висота робочої поверхні має забезпечувати правильне положення рук користувача на клавіатурі, а також відповідати антропометричним даним працівника. Оптимальною вважається висота столу в межах 680-800 мм [40].

Робоче крісло розробника має бути підйомно-поворотним, регульованим за висотою та кутом нахилу спинки, забезпечувати надійну підтримку тіла у зручному положенні. Сидіння та спинка крісла повинні бути м'якими, з заокругленими краями та виготовлені з повітропроникного матеріалу. Для зниження статичного напруження м'язів верхніх кінцівок рекомендується використовувати стаціонарні або змінні підлокітники.

Монітор повинен розташовуватися на оптимальній відстані від очей користувача - 600-700 мм, його верхня точка має знаходитися на рівні очей або трохи нижче. Кут нахилу екрана повинен складати 10-15 градусів, а площа екрана - бути перпендикулярною лінії зору користувача. Клавіатуру слід розміщувати на поверхні столу або на спеціальній полиці на висоті ліктів користувача з кутом нахилу до горизонтальної поверхні 15-20 градусів [40].

Для запобігання перенапруженню зорового аналізатора та розвитку комп'ютерного зорового синдрому необхідно забезпечити належний рівень освітленості робочої зони. Рекомендована освітленість на робочому столі має складати 300-500 лк, а освітленість екрана монітора - не перевищувати 300 лк. Слід уникати прямого попадання світла в очі та відблисків на екрані монітора.

Мікроклімат у приміщенні, де працює розробник вебзастосунків, має відповідати санітарним нормам. Оптимальна температура повітря повинна становити

22-24°C у холодний період року та 23-25°C - у теплий. Відносна вологість повітря - 40-60%, швидкість руху повітря - не більше 0,1 м/с. Для підтримки комфортних параметрів мікроклімату рекомендується використовувати системи кондиціонування або вентиляції.

Дотримання ергономічних та санітарно-гігієнічних вимог до організації робочого місця розробника вебзастосунків дозволить створити оптимальні умови для роботи, знизити ризик розвитку професійних захворювань опорно-рухового апарату та органів зору, підвищити працездатність та ефективність праці фахівця.

5.3. Забезпечення безпечних умов праці та мінімізація впливу шкідливих факторів при роботі з комп'ютерним обладнанням

Забезпечення безпечних умов праці та мінімізація впливу шкідливих факторів при роботі з комп'ютерним обладнанням є одним із ключових завдань охорони праці в галузі інформаційних технологій. Адже, незважаючи на те, що робота розробника вебзастосунків не відноситься до категорії робіт з підвищеною небезпекою, вона все ж таки пов'язана з низкою потенційних ризиків для здоров'я працівника.

До основних шкідливих та небезпечних факторів, які можуть негативно впливати на стан здоров'я розробника при роботі з комп'ютерним обладнанням, належать: підвищений рівень електромагнітного випромінювання, статичне електричне поле, неіонізуюче електромагнітне випромінювання оптичного діапазону, підвищений рівень шуму та вібрації, незадовільні параметри мікроклімату, недостатнє або нераціональне освітлення робочої зони, а також психофізіологічні фактори, такі як розумове перенапруження, перенапруження зорового та опорно-рухового апарату тощо [41].

Для мінімізації впливу вищезазначених факторів та створення безпечних умов праці розробника вебзастосунків необхідно дотримуватися низки вимог та рекомендацій. Зокрема, згідно з ДСанПіН 3.3.2-007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин», рівень електромагнітного випромінювання на відстані 50 см навколо

відеотермінала за електричною складовою має не перевищувати 2,5 В/м - у діапазоні частот 5 Гц - 2 кГц та 0,25 В/м - у діапазоні частот 2 - 400 кГц. Напруженість електростатичного поля на робочому місці з ВДТ не повинна перевищувати гранично допустиме значення - 20 кВ/м [42].

Для зменшення впливу електромагнітного випромінювання рекомендується використовувати монітори, які відповідають сучасним стандартам безпеки, зокрема MPR II або TCO. Також слід дотримуватися регламентованих перерв при роботі з комп'ютером, під час яких необхідно виконувати комплекс вправ для очей, шиї, плечей, спини та кистей рук.

Важливим аспектом забезпечення безпечних умов праці є підтримка оптимальних параметрів мікроклімату у виробничому приміщенні. Згідно з ДСанПіН 3.3.2-007-98, на робочих місцях з ВДТ мають забезпечуватися оптимальні значення температури повітря - 22-24°C, відносної вологості повітря - 40-60% та швидкості руху повітря - не більше 0,1 м/с. У приміщеннях, де встановлені комп'ютери, необхідно щоденно проводити вологе прибирання та регулярно провітрювати.

Для запобігання перенапруженню зорового аналізатора при роботі з комп'ютером необхідно забезпечити раціональне освітлення робочої зони. Освітленість на робочому столі в зоні розташування документів має становити 300-500 лк. Яскравість світильників загального освітлення в зоні кутів випромінювання від 50° до 90° відносно вертикалі в подовжній і поперечній площинах повинна складати не більше 200 кд/м², а яскравість відблисків на екрані ВДТ - не більше 40 кд/м² [42].

Для зниження ризику розвитку синдрому зап'ястного каналу та інших захворювань опорно-рухового апарату рекомендується використовувати ергономічні клавіатури та маніпулятори типу "миша". Також необхідно дотримуватися правильної робочої пози: сидіти прямо, не сутулячись, лікті мають бути зігнуті під кутом 70-90°, стегна - розташовані горизонтально, а стопи - стояти на підлозі або спеціальній підставці.

Не менш важливим фактором забезпечення безпеки праці розробника вебзастосунків є дотримання вимог електробезпеки. Комп'ютерне обладнання та периферійні пристрої повинні підключатися до електромережі через справні штепсельні з'єднання та розетки. Забороняється використовувати саморобні подовжувачі, застосовувати для опалення приміщень нестандартне (саморобне) електронагрівальне обладнання, а також залишати без нагляду увімкнені в електромережу нагрівальні прилади [41].

Для запобігання ураженню електричним струмом у разі випадкового дотику до струмопровідних частин обладнання необхідно забезпечити наявність заземлення або занулення. Періодично слід проводити перевірку стану ізоляції електропроводів та заземлюючих пристроїв.

Висновок

Дотримання вищезазначених вимог та рекомендацій щодо забезпечення безпечних умов праці та мінімізації впливу шкідливих факторів при роботі з комп'ютерним обладнанням дозволить зберегти здоров'я та працездатність розробника вебзастосунків, а також підвищити ефективність та якість його роботи.

РОЗДІЛ 6. ОХОРОНА НАВКОЛИШНЬОГО СЕРЕДОВИЩА

6.1 Вступ до проблеми

Охорона навколишнього середовища здійснюється різними, у тому числі й правовими, засобами. При цьому в правових формах захищаються переважно всі компоненти, які утворюють природне середовище. Сучасними головними нормативно-правовими актами що регулюють основи організації охорони навколишнього природного середовища, є Закони України «Про охорону навколишнього природного середовища» від 25 червня 1991 р., «Про охорону атмосферного повітря» від 16 жовтня 1992 р., «Про природно-заповідний фонд України» від 16 червня 1992 р., «Про тваринний світ» від 3 березня 1993 р., «Про карантин рослин» від 30 червня 1993 р та інші. До того ж деякі відносини у сфері використання і охорони навколишнього природного середовища врегульовані кодексами (земельним, водним, лісовим, про надра), а також Законами України «Про плату за землю» від 3 липня 1992 р., «Про ветеринарну медицину» від 25 червня 1992 р. Важливе значення у вирішенні цього питання має затверджений Постановою Верховної Ради «Порядок обмеження, тимчасової заборони (зупинення) чи припинення діяльності підприємств, установ, організацій і об'єктів у разі порушення ними законодавства про охорону навколишнього природного середовища».

Різновидами права природокористування є: право землекористування, право водокористування, право лісокористування, право користуватися надрами, право користуватися тваринним світом, право користування природно-заповідним фондом. Право природокористування це процес раціонального використання людиною природних ресурсів для задоволення різних потреб та інтересів. Найважливішими принципами природокористування є його цільовий характер, плановість і тривалість, ліцензування, врахування надзвичайного значення у житті суспільства тощо. При цьому виділяють такі групи природокористування, як право загального і спеціального використання землі, вод, лісів, надр, тваринного світу та інших природних ресурсів. Суб'єктами права загального користування природними ресурсами можуть бути,

згідно з Законом України «Про охорону навколишнього природного середовища», усі громадяни для задоволення найрізноманітніших потреб і інтересів. Воно здійснюється громадянами безкоштовно і безліцензійно, тобто для цього не потрібен відповідний дозвіл уповноважених органів і осіб. Загальним є, наприклад, використання парків, скверів, водойм, лісів, збір дикорослих ягід, грибів, горіхів і т. ін. Право загального природокористування закріплене у ст. 13 Конституції України: «Кожний громадянин має право користуватися природними об'єктами права власності народу відповідно до закону». Похідним від загального природокористування є спеціальне використання природних ресурсів. На відміну від першого, це використання конкретних природних ресурсів, що здійснюється громадянами, підприємствами, установами і організаціями у випадках, коли відповідна, визначена у законодавстві частина природних ресурсів передається їм для використання. Як правило, така передача має вартість і визначена в часі. Надання природних ресурсів відбувається на основі спеціальних дозволів державних актів на право постійного користування. Крім прав суб'єктів, як природокористувачів, сучасною юридичною наукою сформовані й інтенсивно розвиваються екологічні права і обов'язки. Так, у Конституції України записано, що «кожен має право на безпечне для життя і здоров'я довкілля та на відшкодування завданої порушенням цього права шкоди. Кожному гарантується право вільного доступу до інформації про стан довкілля, про якість харчових продуктів і предметів побуту, а також право на її поширення». Аналогічні формулювання є й у Законі України «Про охорону навколишнього природного середовища», бо це право одне з основних прав людини. Цьому праву відповідає обов'язок держави забезпечувати здійснення санітарно-гігієнічних заходів, спрямованих на поліпшення та оздоровлення навколишнього природного середовища. Усі екологічні права громадян захищаються і відновлюються у судовому порядку. Поряд з правами Закон України «Про охорону навколишнього природного середовища» передбачає і певні обов'язки громадян. Так, незалежно від того, є громадяни природокористувачами, чи ні, вони зобов'язані берегти природу, раціонально використовувати її запаси, не завдавати

шкоди. Крім того, Закон України «Про охорону навколишнього природного середовища» покладає на громадян і підприємства, установи й організації, як суб'єктів спеціального використання природних ресурсів, спеціальні обов'язки. Так, плата за спеціальне природокористування встановлюється на основі нормативів плати і лімітів використання природних ресурсів. Ці нормативи визначаються з урахуванням розповсюдження природних ресурсів, їх якості, можливості використання, місцезнаходження, можливості переробки і зберігання відходів. До того ж суб'єкти спеціального природокористування зобов'язані сплачувати певні кошти за забруднення навколишнього природного середовища, що встановлюються за викиди у атмосферу забруднюючих речовин; скидання забруднюючих речовин на поверхню води, у територіальні і морські води, а також під землю. Контроль у сфері природовикористання і охорони навколишнього природного середовища здійснюється шляхом перевірки, нагляду, обстеження, інвентаризації та експертиз. Він може здійснюватись як уповноваженими державними органами, так і громадськими формуваннями. Державний контроль покладається на Ради народних депутатів, державні адміністрації та Міністерство охорони навколишнього природного середовища і його органи на місцях.

До основних пріоритетів охорони довкілля та раціонального використання природних ресурсів належать:

- гарантування екологічної безпеки ядерних об'єктів і радіаційного захисту населення та довкілля, зведення до мінімуму шкідливого впливу наслідків аварії на Чорнобильській АЕС;
- поліпшення екологічного стану басейнів рік України та якості питної води;
- стабілізація та поліпшення екологічного стану в містах і промислових центрах Донецько-Придніпровського регіону;
- будівництво нових та реконструкція діючих потужностей комунальних очисних каналізаційних споруд;

- запобігання забрудненню Чорного та Азовського морів і поліпшення їх екологічного стану;
- формування збалансованої системи природокористування та адекватна структурна перебудова виробничого потенціалу економіки, екологізація технологій у промисловості, енергетиці, будівництві, сільському господарстві, на транспорті;
- збереження біологічного та ландшафтного різноманіття, заповідна справа.

Для досягнення цього передбачається вирішення таких завдань:

- зменшення до мінімуму рівня радіаційного забруднення;
- захист повітряного басейну від забруднення, насамперед у великих містах і промислових центрах;
- захист і збереження земельних ресурсів від забруднення, виснаження і нерационального використання;
- збереження і розширення територій з природним станом ландшафту, посилення природоохоронної діяльності на заповідних і рекреаційних територіях;
- підвищення стійкості та екологічних функцій лісів;
- знешкодження, утилізація та захоронення промислових та побутових відходів;
- запобігання забрудненню морських і внутрішніх вод, зменшення та припинення скиду забруднених стічних вод у водні об'єкти, захист підземних вод від забруднення;
- збереження та відродження малих річок, здійснення управління водними ресурсами на основі басейнового принципу;
- завершення створення державної системи моніторингу навколишнього природного середовища;

- створення системи прогнозування, запобігання та оперативних дій у разі надзвичайних ситуацій природного і природно-техногенного походження;
- забезпечення екологічного супроводу процесу конверсії військово-промислового комплексу;
- здійснення заходів щодо екологічного контролю за діяльністю Збройних Сил України;
- розробка механізмів реалізації схем природокористування;
- впровадження дійових економічних складових впливу на систему природокористування;
- створення системи екологічної освіти, виховання та інформування.

6.2 Забруднення навколишнього середовища підприємством з розробки програмного забезпечення

Основними чинниками забруднення навколишнього середовища фірмою є лише використання автотранспорту. Бо підприємство окрім забруднення повітряного середовища не забруднює ні водне, ні ґрунтове.

В сучасних умовах автомобільний транспорт стає найбільш значним джерелом забруднення атмосферного повітря, особливо великих міст.

Відомо, що атмосферне повітря міст постійно забруднюється і за всіма параметрами докорінно відрізняється від повноцінного природного повітря. Міські поселення характеризуються найвищими рівнями антропогенних навантажень на навколишнє середовище, в результаті чого воно деформується, набуває якісно нових рис, аж до зміни мікрокліматичних факторів і фізико-хімічних властивостей середовища, зокрема повітряного басейну. Метеорологічні спостереження свідчать, що температура повітря в межах міських територій у середньому на декілька градусів вища, ніж у приміських районах. Над містами, особливо великими, частіше випадають атмосферні опади, бувають густі тумани, а також смоги – густі тумани, змішані з

димом, кіптявою та вихлопними газами. Прозорість атмосфери в містах менша, ніж за її межами і в сільській місцевості. Тумани, а також запиленість повітря помітно зменшують проникнення до земної поверхні сонячних променів.

Однак за рахунок автомобільних викидів якість атмосферного повітря в містах погіршилась.

Прогресуючому забрудненню атмосфери в містах сприяє висока питома вага автомобілів, адже зростання кількості автотранспортних засобів супроводжується збільшенням об'ємів забруднюючих речовин із вихлопних труб. Останнім часом у міському повітрі виріс об'єм оксидів вуглецю, вуглеводнів, оксидів азоту, сажі. Але найбільшу небезпеку, окрім оксидів азоту, складають сірчані та свинцеві сполуки. Їх вміст у міському повітрі значною мірою виріс. Місто не пристосоване до такої кількості автотранспорту. Довжина пробігу без зупинок між світлофорами становить лише 400–600 м., внаслідок цього середня швидкість руху вдень у центрі міста і на великих автошляхах знижується до 12–20 км/год, а це збільшує витрати палива в 3–4 рази. Відповідно збільшуються й викиди забруднюючих речовин. Автотранспорт також призводить до специфічних форм забруднення повітря. При русі стираються шини і тисячі тонн гуми у вигляді пилу потрапляє в повітря. Автомобільний транспорт не тільки забруднює повітря продуктами згорання палива, він сприяє зростаючому надходженню свинцю в навколишнє середовище. В Україні поки ще використовують бензин із вмістом свинцю 0,36 г/л, тоді як в Англії, Німеччині та США – 0,013–0,15.

6.3 Розробка заходів щодо зменшення забруднення

Природоохоронною є будь-яка діяльність, спрямована на збереження якості навколишнього середовища на рівні, що забезпечує стійкість біосфери. До неї відноситься як великомасштабна, здійснювана на загальнодержавному рівні діяльність по збереженню еталонних зразків недоторканої природи і збереженню розмаїтості видів на Землі, організації наукових досліджень, підготовці фахівців екологів і вихованню населення, так і діяльність окремих підприємств по очищенню

від шкідливих речовин стічних вод і відхідних газів, зниженню норм використання природних ресурсів і т.д. Така діяльність здійснюється в основному інженерними методами.

Існують два основних напрямки природоохоронної діяльності підприємств. Перший – очищення шкідливих викидів. Цей шлях «у чистому вигляді» малоефективний, тому що з його допомогою далеко не завжди вдається цілком припинити надходження шкідливих речовин у біосферу. До того ж скорочення рівня забруднення одного компонента навколишнього середовища веде до посилення рівня забруднення іншого.

І, наприклад, встановлення вологих фільтрів при газоочищенні дозволяє скоротити забруднення повітря, але веде до ще більшого забруднення води. Уловлені з відхідних газів і зливальних вод речовини часто отруюють значні земельні площі.

Використання очисних споруд, навіть найефективніших, різко скорочує рівень забруднення навколишнього середовища, однак не вирішує цієї проблеми цілком, оскільки в процесі функціонування цих установок теж виробляються відходи, хоча й у меншому обсязі, але, як правило, з підвищеною концентрацією шкідливих речовин. Нарешті, робота більшої частини очисних споруджень вимагає значних енергетичних витрат, що, у свою чергу, теж небезпечно для навколишнього середовища.

Крім того, забруднювачі, на знешкодження яких йдуть величезні засоби, являють собою речовини, на які уже витрачена праця і які за рідкісним винятком можна було б використовувати в народному господарстві.

Для досягнення високих еколого-економічних результатів необхідно процес очищення шкідливих викидів сполучити з процесом утилізації уловлених речовин, що уможливить об'єднання першого напрямку з другим.

Другий напрямок – усунення самих причин забруднення, що вимагає розробки маловідходних, а в перспективі і безвідходних технологій виробництва, які дозволяли б комплексно використовувати вихідну сировину й утилізувати максимум шкідливих для біосфери речовин.

Однак далеко не для усіх виробництв знайдені прийнятні техніко-економічні рішення по різкому скороченню кількості відходів, що утворюються, і їх утилізації, тому на даний час доводиться працювати в обох зазначених напрямках.

Піклуючись про удосконалювання інженерної охорони навколишньої природного середовища, треба пам'ятати, що ніякі очисні спорудження і безвідхідні технології не зможуть відновити стійкість біосфери, якщо будуть перевищені припустимі (граничні) значення скорочення природних, не перетворених людиною природних систем, у чому виявляється чинність закону незамінності біосфери.

Таким порогом може виявитися використання більш 1% енергетики біосфери і глибоке перетворення більш 10% природних територій (правила одного і десяти відсотків). Тому технічні досягнення не знімають необхідності рішення проблем зміни пріоритетів суспільного розвитку, стабілізації народонаселення, створення достатнього числа заповідних територій і інших, розглянутих раніше.

Основними напрямками робіт в галузі захисту атмосфери від забруднення викидами автотранспорту є:

- а) створення і розширення виробництва автомобілів з високоекономічними і малотоксичними двигунами, у тому числі подальша дизелізація автомобілів;
- б) розвиток робіт зі створення і впровадження ефективних систем нейтралізації відпрацьованих газів;
- в) зниження токсичності моторних палив;
- г) розвиток робіт з раціональної організації руху автотранспорту в містах, удосконаленню дорожнього будівництва з метою забезпечення невинного руху на автомагістралях.

Висновок

Таким чином, ефективний захист навколишнього середовища потребує комплексного підходу, що поєднує технологічні інновації з усвідомленою зміною суспільних пріоритетів. Важливо не лише впроваджувати нові методи очищення та нейтралізації забруднень, але й зменшувати навантаження на біосферу шляхом

раціонального використання ресурсів і збереження природних територій. Лише гармонійне поєднання технічних рішень і відповідального ставлення людини до природи дозволить забезпечити стабільність екосистем для майбутніх поколінь.

ВИСНОВКИ

У дипломній роботі було розроблено інформаційну систему підтримки волонтерської діяльності, яка 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 інформаційних систем.

Проведений огляд існуючих програмних рішень для підтримки волонтерської діяльності виявив їх переваги та недоліки. Було відзначено необхідність створення більш функціональної та адаптованої під потреби сфери системи, яка би враховувала специфіку волонтерської діяльності та надавала зручні інструменти для управління проєктами, комунікації та мотивації волонтерів.

На основі аналізу предметної області та огляду аналогів було сформульовано мету та основні вимоги до інформаційної системи. Система повинна 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 проєктування та моделювання інформаційної системи. 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 проєктів. 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, проєктуванню 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.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Закон України "Про волонтерську діяльність" від 19.04.2011 № 3236-VI. URL: <https://zakon.rada.gov.ua/laws/show/3236-17> (дата звернення: 20.05.2025).
2. Беспалова 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: 20.05.2025).
6. Idealist. URL: <https://www.idealists.org/> (дата звернення: 20.05.2025).
7. AllForGood. URL: <https://www.allforgood.org/> (дата звернення: 20.05.2025).
8. Volgistics. URL: <https://www.volgistics.com/> (дата звернення: 20.05.2025).
9. Golden. URL: <https://www.goldenvolunteer.com/> (дата звернення: 20.05.2025).
10. InitLive. URL: <https://www.initlive.com/> (дата звернення: 20.05.2025).
11. Volda 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: 20.05.2025).
16. PostgreSQL. URL: <https://www.postgresql.org/> (дата звернення: 20.05.2025).
17. Hibernate. URL: <https://hibernate.org/> (дата звернення: 20.05.2025).
18. Spring MVC. URL: <https://docs.spring.io/spring-framework/docs/current/reference/html/web.html> (дата звернення: 20.05.2025).
19. Thymeleaf. URL: <https://www.thymeleaf.org/> (дата звернення: 20.05.2025).
20. Bootstrap. URL: <https://getbootstrap.com/> (дата звернення: 20.05.2025).
21. Fowler M. Microservices. URL: <https://martinfowler.com/articles/microservices.html> (дата звернення: 20.05.2025).
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: 20.05.2025).
24. JSON Web Tokens. URL: <https://jwt.io/> (дата звернення: 20.05.2025).
25. JUnit. URL: <https://junit.org/> (дата звернення: 20.05.2025).
26. Mockito. URL: <https://site.mockito.org/> (дата звернення: 20.05.2025).
27. Apache JMeter. URL: <https://jmeter.apache.org/> (дата звернення: 20.05.2025).
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 / М. 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.
37. Наказ Міністерства соціальної політики України «Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями» від 14.02.2018 № 207. URL: <https://zakon.rada.gov.ua/laws/show/z0508-18> (дата звернення: 14.05.2025).
38. ДСанПіН 3.3.2-007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин». URL: <https://zakon.rada.gov.ua/rada/show/v0007282-98> (дата звернення: 14.05.2025).
39. ДСТУ ISO 9241-5:2004. Ергономічні вимоги до роботи з відеотерміналами в офісі. Частина 5. Вимоги до компонування робочого місця та до робочої пози. [Чинний від 2006-01-01]. Вид. офіц. Київ, 2006. 24 с.
40. Вимоги до організації робочого місця користувача комп'ютера: веб-сайт. URL: <https://zp.dsp.gov.ua/articles/a-657.html> (дата звернення: 14.05.2025).
41. Охорона праці користувачів комп'ютерів: веб-сайт. URL: https://pidru4niki.com/1821100138196/bzhd/ohorona_pratsi_koristuvachiv_kompyuteriv (дата звернення: 14.05.2025).

42. ДСанПіН 3.3.2-007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин». URL: <https://zakon.rada.gov.ua/rada/show/v0007282-98> (дата звернення: 14.05.2025).

ДОДАТКИ

Додаток А – Технічне завдання на розробку

1 Вступ

1.1 Назва – Система управління волонтерською діяльністю

1.2 Призначення програми: надати можливість зручно та швидко управляти взаємодією волонтерів та організаторів заходів, що проводяться на території України.

1.3 Область застосування: Громадські організації України

2 Призначення розробки

Призначена для спрощення, уніфікації, прискорення роботи волонтерів та організаторів заходів громадських організацій України.

3 Вимоги до системи

3.1 Вимоги до функціональних характеристик

Система буде містити наступні функціональні характеристики:

- Управління інформацією про події, організації, заходи, які мають потребу у волонтерах;
- Своєчасне доведення інформації до волонтерів та інших користувачів web-порталу;
- Управління користувачами;
- Перегляд створеної інформації.

3.2 Вимоги до надійності

3.2.1 Вимоги до забезпечення надійності функціонування програми

Надійне (стійке) функціонування системи має бути забезпечено надійним (стійким) функціонуванням бездротової мережі і мережі електропостачання в університеті, або у іншому місці, де буде розміщений сервер.

3.2.2 Відмова виконання програмного продукту через некоректні дії користувача.

- Перевірка формату вводу даних для коректної роботи.
- Функція автоматичного backup кожен тиждень.

3.3 Умови експлуатації

Данною програмою будуть користуватися працівники, які є впевненими користувачами, та не потребують додаткового навчання по даному ПО, але можуть вимагати додаткову допомогу у роботі з вихідними даними.

3.4 Вимоги до інформаційної та програмної сумісності

3.4.1 Вимоги до інформаційних структур і методів розв'язання

Вимоги до інформаційних структур і методів розв'язання відсутні.

3.4.2 Вимоги до вихідних кодів і мов програмування

Додаткові вимоги не пред'являються.

3.4.3 Вимоги до програмних засобів , які використовуються програмою

Програмні засоби що використовуються мають бути вільно поширювані

3.5 Вимоги до маркування та упаковки

3.5.1 Вимоги до маркування :

- Читабельність тексту, чіткість ілюстрацій;
- Достовірність інформації про систему
- Нанесення на Install CD позначок про дане ПО.

3.5.2 Вимоги до упаковки:

Вимоги до упаковки відсутні.

3.6 Вимоги до транспортування та зберігання

Вимоги до транспортування програми відсутня.

3.7 Спеціальні вимоги

Спеціальні вимоги відсутні.

4 Вимоги до програмної документації

4.1 Попередній склад програмної документації

Попередній склад програмної документації повинен включати в себе:

- технічне завдання;
- устав проекту.

4.2 Склад програмної документації

Програмна документація складається з наступних програмних документів:

- Устав проекту;
- Технічне завдання ;
- Опис системи ;
- Код програми ;
- Керівництво користувача.

Сертифікація містить список всіх програмних документів .

Технічне завдання містить опис завдання роботи, склад функціональних характеристик програми, опис вимог для розробки.

Опис програми містить опис логічної структури програми, форматів вхідних та вихідних даних, використовуваних технічних засобів.

Код програми містить опис блоків програми і запис блоків в програмному коді.

Керівництво користувача містить інформацію , необхідну для ознайомлення користувача з програмним продуктом.

5 Техніко-економічні показники

5.1 Економічні переваги розробки

Орієнтовна економічна ефективність не розраховується.

6 Стадії та етапи розробки програмного забезпечення

6.1 Стадії розробки

Розробка повинна бути проведена в 5 стадій:

1. Технічне завдання
2. Ескізний проект
3. Технічний проект
4. Робочий проект
5. Впровадження

Додаток Б - Програма і методика випробувань

Об'єкт випробувань

Назва об'єкта випробувань: Система «Волонтер», надалі називається «Програма». Програма дозволяє користувачеві пройти автентифікацію та отримати доступ до свого інтерфейсу.

Ціль випробувань

Перевірка придатності програми до використання за призначенням, її відповідність заданим вимогам і програмним документам. Перевірка реально досягнутих характеристик програми.

Вимоги до програми

Програма повинна задовольняти наступним функціональним характеристикам:

- 1) перевірка правильності імені та паролю користувача;
- 2) надання користувачеві доступу до свого інтерфейсу.

Користувач повинен вказати ім'я та пароль.

Вимоги до програмної документації

У комплект програмної документації повинні входити наступні документи:

- Загальний опис системи.
- Посібник користувача – адміністратора.

- Посібник користувача – відповідального модератора.
- Посібник користувача – клієнта.
- Інструкція з формування і ведення бази даних.
- Посібник адміністратора з установки програмного забезпечення на серверах.
- Програма і методика випробувань.
- Текст програми.

Засоби і порядок випробувань

Вимоги до складу і параметрів технічних засобів:

Сервер на базі i586_64 або x86.

Вимоги до складу і параметрів програмних засобів :

- 1) операційна система Windows;
- 2) встановлене з'єднання з сервером баз даних MariaDB 10.1

Група людей, що будуть проводити випробування, повинна складатися з автора програми (розробника) і представника і/або самого замовника. При наявності відповідної можливості, ця група може бути розширена програмістом, спеціалістом-тестувальником і фахівцем-проектувальником, що не брали участь у розробці програми.

Попередньо (не пізніше 3 днів) до початку випробувань уточнюється кінцевий склад групи, що буде проводити випробування. Кожен член групи одержує необхідну

документацію до програми (список вимог до програми, опис програми, керівництво користувача, програму і методику випробувань).

Порядок проведення випробувань:

- послідовно виконуються контрольні тести; результати заносяться у відповідний журнал;
- виробляється аналіз отриманих результатів і встановлюється відповідність програми заданим вимогам і програмним документам.

Методи випробувань

Робота програми залежить тільки від двох параметрів, що вводяться користувачем: від імені та паролю користувача.

Іспити будуть проводитися методом «чорної шухляди».

Тести будуть створюватися на основі класів еквівалентності.

Класи еквівалентності виділяються шляхом вибору кожної вхідної умови і розбивкою його на двох або більш груп. Розрізняють два типи класів еквівалентності: правильні класи еквівалентності, що представляють правильні вхідні дані програми, і неправильні класи еквівалентності, що представляють всі інші можливі стани умов (помилкові вхідні значення). При цьому існує ряд правил:

- один тест може покривати тільки один неправильний клас еквівалентності;
- один тест повинний покривати як можна більше число непокритих правильних класів еквівалентності.

Виділені класи еквівалентності для складання тестів мають унікальні номери і приводяться в таблиці Б.1.

Таблиця Б.1 – Класи еквівалентності

Операція, що тестується	Вхідні умови	Правильні класи еквівалентності	Неправильні класи еквівалентності
Перевірка імені та паролю користувача	Ім'я правильне	Так (1)	Ні (2)
	Пароль правильний	Так (3)	Ні (4)
Надання користувачеві доступу до свого інтерфейсу	Належність користувача до інтерфейсу встановлена	Так (5)	Ні (6)

Тести

Тести складені на основі класів еквівалентності таблиці Б.1. Тести використовують наступні початкові умови:

- 1) користувач – user1, його пароль – pass1 (волонтер);
- 2) користувач – admin, його пароль – adMiNpaSsW0rd (адміністратор системи).

Тест 1

Класи еквівалентності:

1, 3, 5

Вхідні дані:

Ім'я користувача: user1

Пароль користувача: pass1

Результат:

Ім'я та пароль користувача пізнані. Належність до волонтерів встановлена.
Наданий доступ до інтерфейсу волонтера.

Тест 2

Класи еквівалентності:

2, 3, 5

Вхідні дані:

Ім'я користувача: user2

Пароль користувача: pass1

Результати:

Ім'я та пароль користувача не пізнані. Належність до користувачів системи не встановлена. Видане повідомлення про помилку введення імені чи паролю. Доступу до системи не надано.

Тест 3

Класи еквівалентності :

1, 4, 5

Вхідні дані:

Ім'я користувача: user1

Пароль користувача: pass2

Результати:

Ім'я та пароль користувача не пізнані. Належність до користувачів системи не встановлена. Видане повідомлення про помилку введення імені чи паролю. Доступу до системи не надано.

Тест 4

Класи еквівалентності:

1, 3, 5

Вхідні дані:

Ім'я користувача: admin

Пароль користувача: adMinPaSsW0rd

Результати:

Ім'я та пароль користувача впізнані. Наданий доступ до адміністрування системи.

Додаток В. 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";
    }

```

```
}
```

Програмный 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.UserService;
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("/applies")
//    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";
    }
}
;

```

Програмный 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";
    }
}

```

```

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";
}}

```

Програмный 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";
}
}

```