ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробка інформаційної системи для обробки заявок до волонтерських організацій»

на здобуття освітнього ступеня бакалавра
зі спеціальності 121 Інженерія програмного забезпечення
освітньо-професійної програми «Інженерія програмного забезпечення»

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших мають посилання на відповідне джерело

_____ Богдан ДАНИЛЮК
(підпис)

Виконав: здобувач вищої освіти групи ПД-42

Богдан ДАНИЛЮК

Керівник: Іван ШАХМАТОВ
Викладач кафедри ІПЗ

Рецензент: _____

Київ 2025

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти Бакалавр

Спеціальність 121 Інженерія програмного забезпечення

Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення
_____ Ірина ЗАМРІЙ
« ____ » _____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Данилюку Богдану Миколайовичу

1. Тема кваліфікаційної роботи: «Розробка інформаційної системи для обробки заявок до волонтерських організацій»
керівник кваліфікаційної роботи викладач кафедри ІПЗ Іван ШАХМАТОВ, затверджені наказом Державного університету інформаційно-комунікаційних технологій від «07» квітня 2025 р. № 115.

2. Строк подання кваліфікаційної роботи «28» травня 2025 р.

3. Вихідні дані до кваліфікаційної роботи: теоретичні відомості про методи розробки веб-застосунків, опис роботи веб-додатків,.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Проаналізувати специфіку процесів обробки заявок у волонтерських організаціях, а також оцінити існуючі програмні рішення, що застосовуються у цій сфері.
2. Сформулювати функціональні та нефункціональні вимоги до інформаційної системи на основі потреб користувачів і специфіки волонтерської діяльності..
3. Розробити архітектуру клієнт-серверної системи, яка забезпечуватиме її гнучкість, масштабованість, розширюваність та безпеку.

4. Програмна реалізація та тестування системи для обробки заявок до волонтерських організацій

5. Перелік графічного матеріалу: *презентація*
   1. Аналіз аналогів.
   2. Вимоги до програмного забезпечення.
   3. Програмні засоби реалізації.
   4. Модель взаємодії користувачів з інформаційною системою.
   5. Діаграма послідовності
   6. Архітектура веб-застосунку
   7. Діаграма класів
   8. Екранні форми.
   9. Демонстрація роботи застосунку
   10. Апробація результатів дослідження

6. Дата видачі завдання «08» квітня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Підбір та аналіз науково-технічної літератури	08.04-12.04.2025	
2	Аналіз та дослідження існуючих аналогів	13.04-15.04.2025	
3	Огляд та аналіз систем для обробки волонтерських заявок	16.04-17.04.2025	
4	Огляд та аналіз засобів реалізації веб-застосунку для обробки волонтерських заявок	18.04-19.04.2025	
5	Проєктування архітектури веб-застосунку для обробки волонтерських заявок	20.04-24.04.2025	
6	Програмна реалізація та тестування веб-застосунку для обробки волонтерських заявок	24.04-30.04.2025	
7	Оформлення роботи: вступ, висновки, реферат	01.05-05.05.2025	
8	Розробка демонстраційних матеріалів	06.05-12.05.2025	
9	Попередній захист роботи	13.05-31.05.2025	

Здобувач вищої освіти _____ Богдан ДАНИЛЮК
(підпис)

Керівник
кваліфікаційної роботи _____ Іван ШАХМАТОВ
(підпис)

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 65 стор., 1 табл., 21 рис., 20 джерел.

Мета роботи – оптимізація та аналіз результатів волонтерської діяльності.

Об'єкт дослідження – процеси управління заявками у волонтерських організаціях.

Предмет дослідження – системи для обробки та управління заявками у волонтерських організаціях.

Короткий зміст роботи: В роботі проаналізовано предметну галузь там методи реалізації застосунку для автоматизації волонтерських запитів. Проведено глибокий аналіз процесів обробки заявок у волонтерських організаціях, що дозволило виявити типові труднощі, які виникають при ручній або неструктурованій обробці запитів. Визначено функціональні та нефункціональні вимоги до інформаційної системи. Спроєктовано логічну структуру бази даних із дотриманням принципів нормалізації, що забезпечує ефективне управління даними та підтримку типових CRUD-операцій. Проведено функціональне тестування системи, яке підтвердило її працездатність, відповідність вимогам та зручність у реальному використанні.

Сферою використання програмного засобу є цифрова підтримка волонтерської діяльності в межах громадських ініціатив, що передбачає автоматизацію процесів реєстрації заявок на допомогу, управління обліком волонтерів, розподіл завдань та моніторинг виконання заяв, з метою підвищення ефективності та оперативності реагування на потреби населення.

КЛЮЧОВІ СЛОВА: FLASK, SQLITE, BACK-END, FRONT-END, ВЕБДОДАТОК, АУТЕНТИФІКАЦІЯ, АВТОМАТИЗАЦІЯ.

ЗМІСТ

ВСТУП

В умовах сучасних викликів, з якими стикається Україна, волонтерська діяльність набуває особливої значущості та масштабів. Ефективна координація та організація волонтерської допомоги вимагає впровадження сучасних інформаційних технологій, які б забезпечували оперативність, прозорість та результативність надання допомоги. Оптимізація процесів обробки заявок до волонтерських організацій є одним із ключових факторів підвищення ефективності волонтерської діяльності в цілому.

Волонтерські організації щодня стикаються з необхідністю обробки значної кількості різноманітних заявок, їх категоризації, пріоритезації та розподілу між виконавцями. Традиційні методи організації цих процесів, такі як використання електронних таблиць, месенджерів або соціальних мереж, мають суттєві обмеження з точки зору масштабованості, контролю та аналітики. Відсутність спеціалізованих інформаційних систем призводить до затримок у обробці заявок, втрати інформації, дублювання зусиль та зниження ефективності волонтерської діяльності в цілому.

Актуальність даної роботи зумовлена гострою потребою у створенні спеціалізованого програмного забезпечення для автоматизації процесів обробки заявок волонтерських організацій, яке б враховувало специфіку українського контексту та забезпечувало ефективну взаємодію між різними учасниками процесу – заявниками, волонтерами та координаторами. Така система має потенціал значно підвищити ефективність волонтерської допомоги, забезпечити прозорість процесів та сприяти оптимальному використанню обмежених ресурсів.

Мета роботи оптимізація та аналіз результатів волонтерської діяльності. Для досягнення поставленої мети необхідно вирішити ряд взаємопов'язаних завдань, зокрема:

– проаналізувати специфіку процесів обробки заявок у волонтерських організаціях та існуючі рішення в цій сфері;

– визначити функціональні та нефункціональні вимоги до системи;

– розробити архітектуру системи, яка забезпечуватиме її гнучкість, масштабованість та безпеку;

– спроєктувати структуру бази даних для ефективного зберігання та обробки інформації;

– створити інтуїтивно зрозумілий інтерфейс користувача, адаптований для різних ролей та пристроїв;

– реалізувати систему з використанням сучасних технологій та провести її всебічне тестування.

Практичне значення роботи полягає у створенні готового до впровадження програмного продукту, який може бути безпосередньо використаний волонтерськими організаціями для підвищення ефективності їхньої діяльності. Розроблена система має потенціал стати стандартним інструментом для координації волонтерської допомоги в українських реаліях.

Структура роботи відображає логіку дослідження та розробки системи. У першому розділі проводиться аналіз предметної області, існуючих рішень та технологій. Другий розділ присвячено проєктуванню системи, включаючи архітектуру, моделювання вимог, проєктування бази даних та інтерфейсу користувача. Третій розділ описує процес реалізації серверної та клієнтської частин системи. Четвертий розділ присвячено тестуванню та оцінці розробленої системи, включаючи функціональне тестування, оцінку продуктивності та прийнятність для користувачів.

Розробка інформаційної системи для обробки заявок волонтерських організацій має не лише технологічне, але й соціальне значення, оскільки сприяє підвищенню ефективності волонтерської діяльності, яка відіграє критично головну роль у подоланні сучасних викликів, з якими стикається Україна. Впровадження такої системи дозволить оптимізувати процеси надання допомоги та забезпечити більш ефективне використання ресурсів, що

в кінцевому результаті сприятиме посиленню стійкості українського суспільства в умовах кризи.

Практична значущість результатів полягає в можливості використання реалізованого вебзастосунку для підвищення ефективності волонтерської діяльності шляхом автоматизації процесів обробки заявок, розподілу завдань, взаємодії між учасниками та контролю виконання. Система може бути розгорнута на більшості сучасних пристроїв, що забезпечує її доступність та масштабованість у межах громадських або благодійних ініціатив.

РОЗДІЛ 1. АНАЛІЗ ПРОБЛЕМИ ТА ІСНУЮЧИХ РІШЕНЬ

1.1 Опис задачі та предметної області

В умовах сучасних викликів волонтерська діяльність набуває особливої актуальності та значимості в Україні. Волонтерські організації стикаються з необхідністю ефективної координації значних обсягів допомоги, які надходять від громадян та організацій. Ключовим аспектом такої координації є процес обробки та розподілу запитів на допомогу, який має бути максимально оперативним та прозорим. Предметна область даного дослідження охоплює процеси функціонування волонтерських організацій, зокрема механізми прийому, обробки та виконання заявок на різні види допомоги . Типовий цикл обробки заявки включає кілька етапів: створення запиту заявником, перевірка та затвердження координатором, призначення волонтера для виконання, реалізація допомоги та звітування про результати [1].

Сфера волонтерської діяльності в Україні характеризується високою динамічністю та різноманітністю запитів — від невідкладної медичної допомоги до гуманітарної підтримки цивільного населення та технічного забезпечення оборонних потреб. Кожен із цих напрямків має свою специфіку, що вимагає гнучкого підходу до організації роботи волонтерів. В умовах значного попиту на волонтерську допомогу виникає проблема ефективної комунікації між трьома ключовими сторонами процесу: заявниками (отримувачами допомоги), волонтерами (виконавцями) та координаторами (представниками волонтерських організацій) [2]. Традиційні механізми комунікації через соціальні мережі, месенджери чи телефонні дзвінки мають суттєві обмеження: інформація розпорошується, втрачається цілісність процесу, ускладнюється контроль виконання та аналіз результатів.

Особливої гостроти набуває питання прозорості та підзвітності волонтерської діяльності. Заявники хочуть мати впевненість, що їхні запити

отримають належний розгляд, волонтери потребують чітких інструкцій та інформації для виконання завдань, а координатори мають забезпечувати ефективний розподіл ресурсів та контроль за виконанням заявок.

Вагомим фактором є також питання безпеки та конфіденційності даних. Волонтерська діяльність може стосуватися чутливих сфер, таких як медична допомога чи підтримка уразливих груп населення, що вимагає забезпечення належного рівня захисту персональних даних та інформації про допомогу.

У зв'язку з масштабністю волонтерської діяльності виникає необхідність категоризації та пріоритезації запитів. Різні типи допомоги мають різні рівні терміновості та вимагають різних компетенцій від волонтерів, що має відображатися в системі обробки заявок. Географічний аспект також є необхідним чинником у волонтерській діяльності. Розподіл волонтерів та заявників по різних регіонах країни вимагає ефективної системи геолокації та географічного узгодження, щоб забезпечити оптимальну логістику допомоги [3].

Аналіз існуючих процесів у волонтерських організаціях свідчить про значну частку ручної роботи та використання неспеціалізованих інструментів, таких як текстові документи, електронні таблиці чи окремі облікові системи. Це призводить до дублювання роботи, затримок у обробці заявок та відсутності єдиної системи звітності. У контексті цифрової трансформації у всіх сферах діяльності, волонтерський сектор також потребує впровадження сучасних технологічних рішень, які дозволять автоматизувати рутинні процеси та зосередити зусилля на вирішенні ключових завдань.

Кадрове питання є одним із найскладніших аспектів волонтерської діяльності, оскільки вона переважно базується на добровільних засадах. Ефективний менеджмент волонтерів вимагає створення зручних інструментів для їх залучення, координації та мотивації, а також для моніторингу їх активності та результативності. Нарешті, необхідним критерієм ефективності волонтерської діяльності є швидкість реагування. Особливо в кризових

ситуаціях критичним є час від подання заявки до її реалізації, що вимагає мінімізації проміжних етапів та бюрократичних перепон [4].

Враховуючи всі ці аспекти, розробка спеціалізованої інформаційної системи для обробки заявок до волонтерських організацій є актуальним та необхідним завданням. Така система має забезпечити ефективну взаємодію всіх учасників процесу, автоматизацію рутинних операцій, прозорість та підзвітність, а також надати інструменти для аналізу та оптимізації волонтерської діяльності.

1.2 Аналіз існуючих програмних рішень

У сфері організації волонтерської діяльності існує низка програмних продуктів, які пропонують функціональність для обробки заявок та координації волонтерів. Проаналізуємо найбільш поширені рішення, їх переваги та недоліки в контексті поставленої задачі.

VolunteerMatch — одна з найбільших міжнародних платформ, яка з'єднує волонтерів із організаціями. Система дозволяє волонтерам шукати проєкти за інтересами та локацією, а організаціям — публікувати запити на допомогу. Серед переваг VolunteerMatch можна виділити велику базу користувачів та інтуїтивний інтерфейс. Однак, це рішення має обмежену функціональність щодо управління життєвим циклом заявок, зокрема, відсутня повноцінна система модерації та категоризації запитів за терміновістю. Крім того, платформа не адаптована під специфіку українського контексту, особливо в умовах кризових ситуацій.

Salesforce Nonprofit Cloud пропонує комплексне рішення для управління некомерційними організаціями, включаючи модулі для координації волонтерів та управління запитами на допомогу. Перевагами цієї системи є висока масштабованість, потужні аналітичні інструменти та інтеграція з іншими бізнес-процесами. Проте, Salesforce вимагає значних ресурсів для впровадження та підтримки, має складний інтерфейс, який потребує

спеціального навчання користувачів, та високу вартість ліцензування, що може бути непідйомним для більшості українських волонтерських організацій.

Better Impact (раніше відомий як Volunteer Impact) — спеціалізована система управління волонтерами, яка включає функції реєстрації, відстеження годин роботи, комунікації та звітності. Система має зручний мобільний застосунок для волонтерів та інтуїтивний інтерфейс для адміністраторів. Однак, у Better Impact недостатньо розвинені функції управління заявками на допомогу та відсутня гнучка система категоризації запитів, що необхідно для диверсифікованої волонтерської діяльності в Україні.

Volunteer Hub — платформа, що фокусується на автоматизації процесів реєстрації та розподілу волонтерів. Вона пропонує функції відстеження годин, комунікації та звітності. Серед переваг — висока надійність та безпека даних, однак система має обмежену функціональність щодо управління кризовими ситуаціями та не підтримує українську мову, що ускладнює її використання локальними організаціями [5].

Доброчин — українська платформа для координації благодійної та волонтерської допомоги. Система забезпечує базову функціональність для публікації потреб та пошуку волонтерів. Перевагою є адаптація до українського контексту та простота використання. Проте, платформа має обмежені можливості для координації та моніторингу виконання заявок, а також відсутні розвинені аналітичні інструменти.

Galaxy Digital (Get Connected) — система управління волонтерськими програмами з акцентом на залучення громади. Вона пропонує функції реєстрації, пошуку можливостей та звітності. Перевагою є гнучкість налаштувань та інтеграція з соціальними мережами. Проте, система має високу вартість та орієнтована переважно на американський ринок, що ускладнює її адаптацію до українських реалій.

Для більш системного порівняння існуючих рішень, представимо їх основні характеристики у вигляді таблиці (табл. 1.1).

Таблиця 1.1

Порівняння існуючих програмних рішень для управління волонтерською діяльністю

Характеристика	**VolunteerMatch**	**Salesforce Nonprofit**	**Better Impact**	**Volunteer Hub**	**Доброчин**	**Galaxy Digital**	**Angels Volunteer**
Управління заявками	-	+	+	-	+	+	+
Категоризація запитів	+	+	+	+	+	+	+
Географічна локалізація	+	+	-	+	+	+	+
Терміновість запитів	-	+	-	-	+	-	+
Українська мова	-	-	-	-	+	-	+
Кризове управління	-	-	-	-	-	-	+
Аналітика	-	+	+	-	-	+	+
Мобільний доступ	+	+	+	+	-	+	+

Аналіз існуючих рішень виявив, що жодне з них повністю не відповідає специфічним потребам українських волонтерських організацій, особливо в контексті оперативної обробки заявок різних категорій та терміновості. Більшість міжнародних платформ не локалізовані українською мовою та не враховують особливості локального контексту, а українські рішення мають обмежену функціональність для комплексного управління процесами [6].

Таким чином, існує об'єктивна потреба у розробці спеціалізованої інформаційної системи, яка б враховувала всі зазначені аспекти та забезпечувала ефективну координацію волонтерської допомоги в українських реаліях. Така система має поєднувати функціональність управління заявками, координації волонтерів та звітності, бути доступною за вартістю впровадження та підтримки, а також простою у використанні для всіх категорій користувачів.

1.3 Аналіз доступних ІТ-засобів та технологій

Розробка інформаційної системи для волонтерських організацій вимагає ретельного вибору технологічного стеку, який забезпечить оптимальне співвідношення функціональності, простоти впровадження та вартості підтримки. Особливістю таких систем є необхідність швидкого розгортання, інтуїтивно зрозумілого інтерфейсу та можливості модифікації під конкретні потреби організацій [7].

У сфері серверних технологій сьогодні домінує кілька основних підходів. Платформа Node.js дозволяє використовувати JavaScript як на клієнтській, так і на серверній частині, що спрощує розробку та підтримку. Однак для систем з комплексною бізнес-логікою більш відповідними є рішення на базі Python, Java або PHP. Python з його фреймворками Django та Flask надає оптимальний баланс між швидкістю розробки та функціональністю, особливо коли проєкт потребує поступового розширення можливостей. Вагомим компонентом системи є база даних, вибір якої залежить від очікуваного навантаження та складності даних. Для невеликих організацій з обмеженими технічними ресурсами SQLite надає достатню функціональність без необхідності налаштування окремого сервера баз даних. При збільшенні масштабу діяльності доцільним стає перехід на повноцінні системи управління базами даних, такі як PostgreSQL або MySQL, які забезпечують кращу продуктивність та надійність при одночасній роботі багатьох користувачів.

У контексті клієнтської частини використання HTML, CSS та JavaScript з фреймворком Bootstrap дозволяє створювати адаптивні інтерфейси, які однаково добре працюють на різних пристроях — від настільних комп'ютерів до мобільних телефонів. Така мультиплатформність критично важлива для волонтерських систем, де користувачі можуть взаємодіяти з системою в різних умовах та з різних пристроїв.

Системи автентифікації та авторизації становлять особливий інтерес, оскільки в межах волонтерської платформи необхідно чітко розмежовувати права доступу між різними типами користувачів — заявниками, волонтерами та координаторами. Реалізація таких механізмів може базуватися на стандартних рішеннях, що пропонуються веб-фреймворками, або на спеціалізованих протоколах, таких як OAuth 2.0 або JSON Web Tokens [8]. Аналіз доступних інструментів розгортання показує, що для волонтерських проєктів оптимальним є використання хмарних платформ, які мінімізують витрати на інфраструктуру та спрощують процеси масштабування. Такі сервіси, як Heroku або Netlify, дозволяють швидко розгорнути застосунок без глибоких знань у сфері системного адміністрування [9].

Детальне вивчення потреб волонтерських організацій, а також доступних технологічних рішень, дозволяє визначити оптимальний стек для розробки інформаційної системи. У даному проєкті обрано Python з використанням фреймворку Flask для серверної частини, що забезпечує гнучкість та легкість розширення функціональності [10]. Для зберігання даних використовується SQLite — легка вбудована база даних, яка не потребує окремого серверного програмного забезпечення. Інтерфейс користувача реалізовано за допомогою HTML та CSS з використанням фреймворку Bootstrap, що забезпечує адаптивність та сучасний вигляд додатку.

Такий технологічний стек оптимальний для невеликих та середніх волонтерських організацій, оскільки не вимагає значних ресурсів для розробки, впровадження та підтримки, водночас забезпечуючи всю необхідну функціональність для ефективної обробки заявок. Додатковою перевагою є широка спільнота розробників та велика кількість навчальних матеріалів для обраних технологій, що спрощує подальшу підтримку та розвиток системи.

1.4 Визначення вимог до ПЗ

Проаналізувавши сутність застосунку, його специфіку, цільову аудиторію та аналоги, було визначено функціональні та нефункціональні вимоги до застосунку

1.4.1 Функціональні вимоги

- Застосунок має підтримувати створення облікових записів та вхід для трьох основних ролей: заявник, волонтер, координатор.
- Заявник повинен мати змогу створювати нові заявки, переглядати та відстежувати їх статус.
- Координатор повинен мати можливість переглядати нові заявки, змінювати їх статуси та призначати їх відповідним волонтерам. Вибір та виконання заявок волонтерами
- Волонтери можуть переглядати список доступних заявок, фільтрувати їх за критеріями та брати у роботу.
- Доступ до функціоналу має бути обмежений відповідно до ролі користувача.
- Координатор повинен мати змогу переглядати статистику щодо виконаних заявок, активності волонтерів тощо.
- Усі ролі повинні мати інструменти фільтрації та пошуку заявок за статусом, категорією, терміновістю, локацією тощо.

1.4.2 Нефункціональні вимоги

- Інтерфейс має бути інтуїтивно зрозумілим, адаптованим під різні ролі та легко доступним з браузера. Застосунок має коректно обробляти помилки введення, збої при підключенні до БД тощо.
- Зберігання паролів у хешованому вигляді, валідація форм.

- Час відгуку інтерфейсу та основних дій (пошук, авторизація, зміна статусу заявки) не має перевищувати 1 секунди.

1.5 Обґрунтування вибору програмних засобів реалізації

1. **Back-End: Flask (Python)**
   - Забезпечує гнучкість та легкість розширення.
   - Добре підходить для невеликих систем, які згодом можуть масштабуватись.
   - Має велику спільноту та багато документації.
   - Підтримує ORM через SQLAlchemy, що підвищує безпеку та спрощує роботу з базою даних.
2. **База даних: SQLite**
   - Легка, вбудована СУБД, яка не потребує окремого сервера.
   - Оптимальна для невеликих і середніх проєктів.
   - Дає змогу зберігати дані у одному файлі, що спрощує резервне копіювання.
3. **Front-End: HTML/CSS з Bootstrap**
   - Дозволяє створити адаптивний і сучасний інтерфейс.
   - Працює на різних пристроях — ПК, планшетах, смартфонах.
   - Bootstrap пришвидшує верстку завдяки готовим компонентам.
4. **Аутентифікація та безпека**
   - Використано Flask-Login, PBKDF2, CSRF-захист, ORM-запити (SQLAlchemy), що відповідає стандартам безпеки вебзастосунків.

РОЗДІЛ 2. ПРОЄКТУВАННЯ СИСТЕМИ

2.1 Проєктування архітектури системи

Проєктування архітектури інформаційної системи для обробки заявок волонтерських організацій вимагає особливого підходу, який враховує специфіку предметної області, необхідність забезпечення гнучкості та подальшого розвитку системи. Для даного проєкту було обрано багаторівневу архітектуру на основі патерну Model-View-Controller (MVC), який зарекомендував себе як ефективне рішення для веб-додатків.

Основою архітектури є чіткий поділ відповідальності між різними компонентами системи. Компонент моделі даних відповідає за структуру та взаємодію з базою даних, забезпечуючи абстракцію даних від решти системи. Компонент представлення формує інтерфейс користувача, адаптований для різних ролей та пристроїв. Компонент контролера реалізує бізнес-логіку, обробляє запити користувачів та координує взаємодію між моделлю та представленням.

Особливістю архітектури є інтеграція системи автентифікації та авторизації, яка забезпечує розмежування доступу залежно від ролі користувача. Заявники, волонтери та координатори мають різні можливості в системі, що відображено як на рівні інтерфейсу, так і на рівні доступу до функціональності. Такий підхід дозволяє забезпечити необхідний рівень безпеки при збереженні зручності використання. На рисунку 2.1 представлено діаграму компонентів архітектури системи.

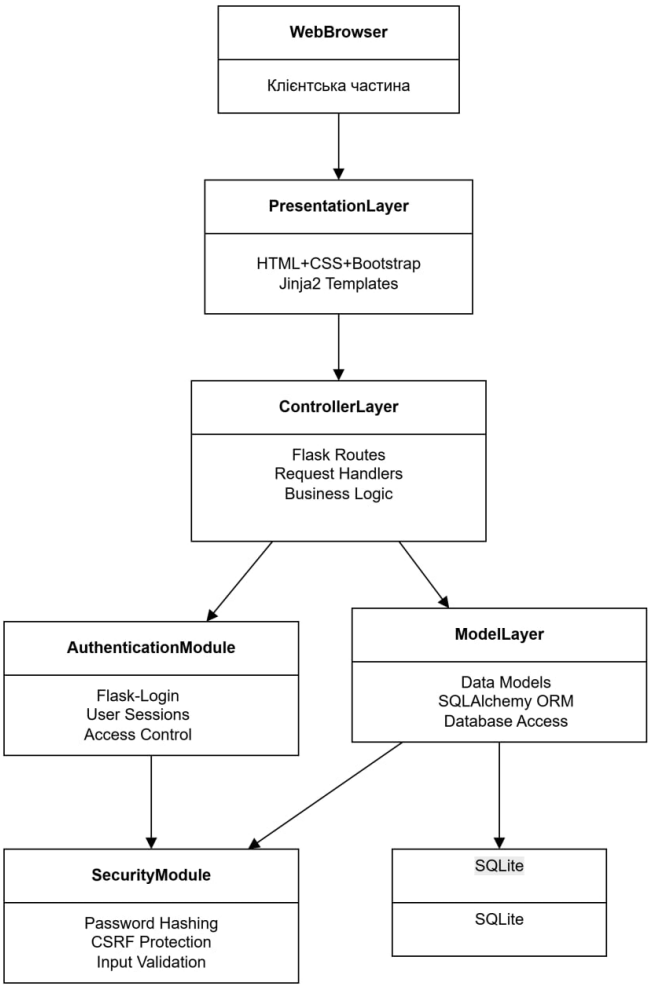


Рис. 2.1 Діаграма компонентів архітектури системи

На серверній стороні використовується фреймворк Flask, який надає необхідну гнучкість для реалізації бізнес-логіки. Для взаємодії з базою даних застосовується ORM-підхід через SQLAlchemy, що забезпечує абстракцію від конкретної СУБД та спрощує роботу з даними. Даний підхід також сприяє безпеці додатку, мінімізуючи ризики SQL-ін'єкцій та інших поширених вразливостей. Клієнтська частина реалізована з використанням HTML-шаблонів з адаптивним дизайном, що забезпечує коректне відображення інтерфейсу на різних пристроях. Такий підхід особливо необхідним для волонтерської системи, оскільки користувачі можуть працювати з нею в різноманітних умовах — від офісного комп'ютера до мобільного телефону в польових умовах [11]. На рисунку 2.2 представлено діаграму класів основних сутностей системи.

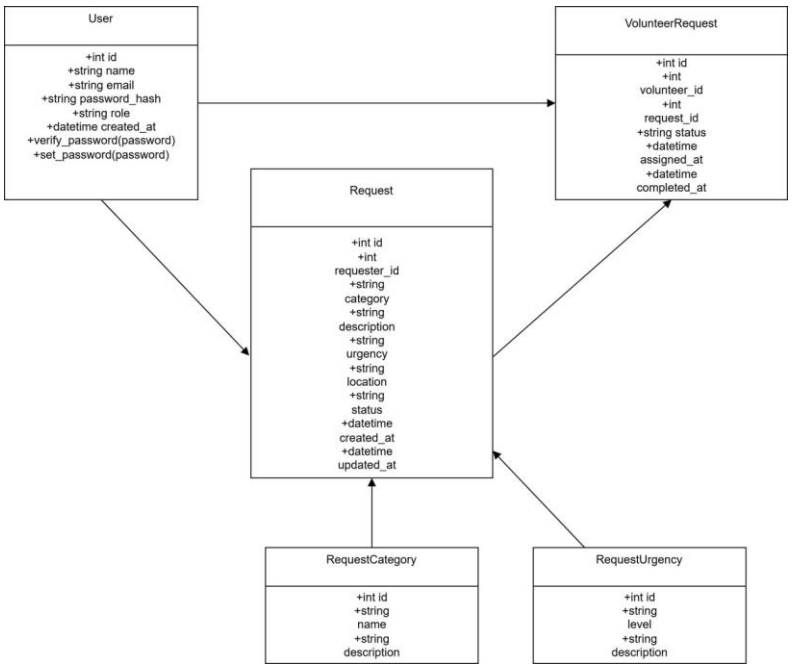


Рис. 2.2 Діаграма класів основних сутностей системи

Вагомим архітектурним рішенням є організація системи як моноліту, що спрощує процес розгортання та підтримки. Враховуючи специфіку волонтерських організацій, які часто мають обмежені технічні ресурси, таке рішення є оптимальним на початковому етапі впровадження. При зростанні навантаження архітектура дозволяє еволюціонувати в напрямку мікросервісів без суттєвої реорганізації коду.

Архітектура системи також передбачає можливості для подальшого розширення функціональності. Модульний підхід до організації компонентів дозволяє додавати нові функції без впливу на існуючі, що є критично необхідним для систем, які активно розвиваються відповідно до змінних потреб користувачів.

Значна увага при проєктуванні приділялася питанням безпеки. Крім захисту від стандартних вразливостей, система передбачає шифрування критичних даних, зокрема паролів користувачів, та реалізує механізми захисту сесій для запобігання несанкціонованому доступу. Локалізація інтерфейсу системи українською мовою є необхідним архітектурним аспектом, орієнтованим на цільову аудиторію. Структура проєкту при цьому дозволяє в

майбутньому реалізувати підтримку додаткових мов без суттєвої переробки коду.

Таким чином, спроєктована архітектура інформаційної системи для обробки заявок волонтерських організацій забезпечує оптимальний баланс між функціональністю, безпекою та простотою використання, що є ключовим фактором успішного впровадження в умовах обмежених ресурсів волонтерських організацій.

2.2 Моделювання вимог

Ефективна розробка інформаційної системи для обробки заявок волонтерських організацій вимагає чіткого визначення та моделювання вимог. Цей процес забезпечує відповідність кінцевого продукту очікуванням користувачів та специфіці предметної області. У даному розділі проведено аналіз та моделювання функціональних і нефункціональних вимог до системи.

Функціональні вимоги до системи визначають конкретні можливості, які вона повинна надавати користувачам. Зважаючи на специфіку волонтерської діяльності, система має забезпечувати диференційований доступ до функціоналу для трьох основних категорій користувачів: заявників, волонтерів та координаторів. Кожна категорія має унікальний набір потреб та задач.

Для заявників (осіб, які потребують допомоги) система повинна надавати можливість реєстрації та автентифікації, створення нових заявок з детальним описом потреби, визначенням категорії допомоги, рівня терміновості та географічної локації. Вагомим аспектом є можливість відстеження статусу поданих заявок у режимі реального часу, що забезпечує прозорість процесу та підвищує довіру до системи.

Волонтери потребують інструментів для перегляду доступних заявок з можливістю фільтрації за різними параметрами (категорія, локація, терміновість), що дозволяє ефективно обирати завдання відповідно до їхніх

можливостей та компетенцій. Після вибору заявки система має забезпечувати механізм її закріплення за волонтером та відстеження процесу виконання. Також необхідна функціональність для позначення заявки як виконаної та формування звіту про результати.

Для координаторів (представників волонтерських організацій) критично головним є інструменти модерації та управління. Система повинна забезпечувати можливість перевірки нових заявок, їх затвердження або відхилення, а також моніторинг активності волонтерів. Координаторам також необхідні аналітичні інструменти для оцінки ефективності роботи організації, визначення проблемних зон та прийняття управлінських рішень [12].

Для моделювання взаємодії користувачів з системою використано діаграму варіантів використання (Use Case Diagram), яка зображена на рисунку 2.3 і візуалізує основні функціональні можливості системи для кожної категорії користувачів.

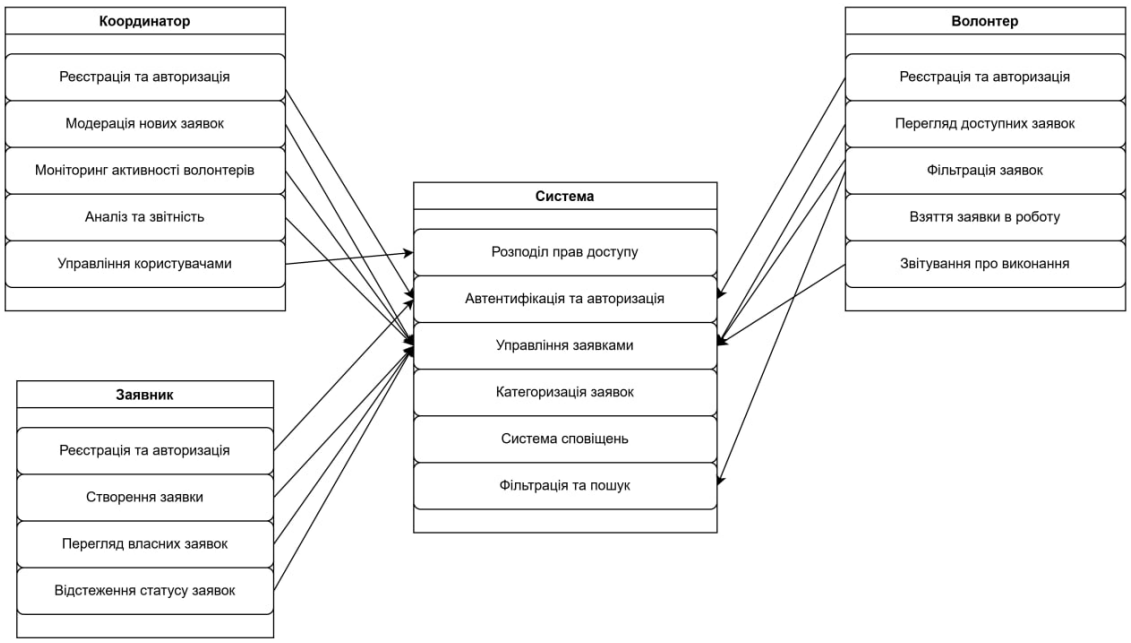


Рис. 2.3 Діаграма варіантів використання системи

Нефункціональні вимоги визначають якісні характеристики системи, які впливають на задоволеність користувачів та ефективність роботи.

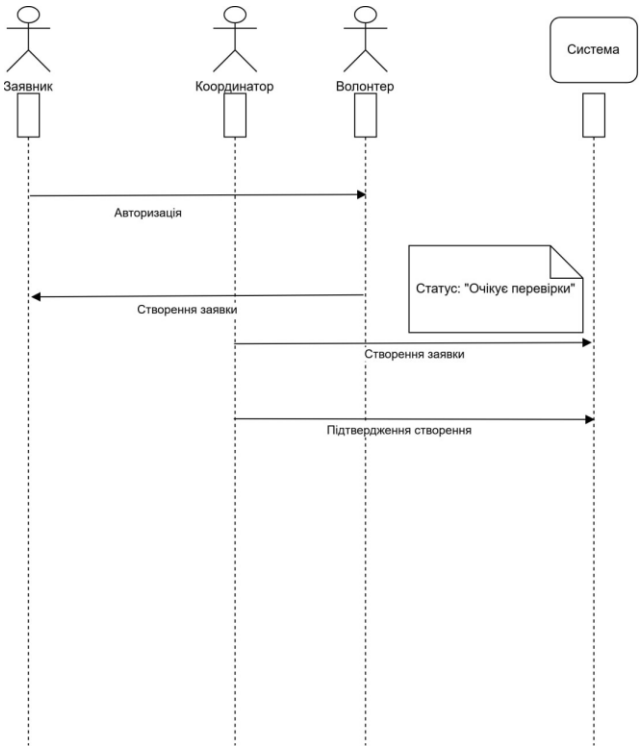


Рис. 2.4 Діаграма послідовності процесу обробки заявки

Для більш детального визначення вимог до кожного компонента системи були розроблені користувацькі історії (user stories), які описують функціональність з точки зору конкретного користувача. Такий підхід дозволяє краще зрозуміти потреби та очікування різних категорій користувачів [13].

У результаті моделювання вимог було визначено, що система повинна забезпечувати функціональність для трьох основних ролей користувачів, підтримувати повний життєвий цикл заявки від створення до завершення, забезпечувати механізми фільтрації та категоризації заявок, а також надавати аналітичні інструменти для координаторів. Особлива увага приділяється безпеці даних та зручності використання, що критично важливо для успішного впровадження системи в реальних умовах волонтерської діяльності.

Визначені та змодельовані вимоги становлять основу для подальшого проєктування структури бази даних та інтерфейсу користувача, забезпечуючи їх відповідність реальним потребам користувачів та бізнес-процесам волонтерських організацій.

2.3 Проєктування структури бази даних

Ефективна система обробки заявок волонтерських організацій потребує ретельно спроєктованої структури бази даних, яка забезпечуватиме цілісність інформації, швидкий доступ до даних та можливість масштабування. В рамках розробки інформаційної системи було спроєктовано реляційну модель бази даних, яка відображає основні сутності, їх атрибути та зв'язки між ними. Концептуальна модель бази даних формувалася на основі аналізу предметної області та виявлених вимог до системи. Було визначено три основні сутності, які є ключовими для функціонування волонтерської платформи: користувачі, заявки та призначення заявок волонтерам. Кожна з цих сутностей має свій набір атрибутів та специфіку використання в системі.

Сутність "Користувач" (User) є центральною в системі, оскільки представляє всіх учасників процесу волонтерської діяльності. Атрибути цієї сутності включають унікальний ідентифікатор, ім'я, електронну адресу, пароль (зберігається у захищеному вигляді) та роль користувача в системі [14]. Роль є критично необхідним атрибутом, який визначає доступні функції та інтерфейс для користувача. В системі підтримуються три основні ролі: заявник, волонтер та координатор.

Сутність "Заявка" (Request) представляє запит на волонтерську допомогу. Кожна заявка має унікальний ідентифікатор, зв'язок з користувачем-заявником, категорію допомоги, детальний опис, рівень терміновості, географічну локацію, статус та часову мітку створення. Категорія допомоги дозволяє класифікувати заявки за типом потреби (медикаменти, спорядження, дрони, автотранспорт), що спрощує пошук та розподіл заявок між волонтерами. Статус заявки відображає її поточний стан у життєвому циклі: від початкового "Очікує перевірки" до кінцевого "Завершено".

Сутність "Призначення заявки" (VolunteerRequest) реалізує зв'язок між волонтером та заявкою, яку він обрав для виконання. Атрибути цієї сутності

включають унікальний ідентифікатор, посилання на користувача-волонтера, посилання на заявку, часову мітку призначення та статус виконання. Ця сутність дозволяє відстежувати, хто саме з волонтерів працює над конкретною заявкою та на якому етапі знаходиться процес її виконання.

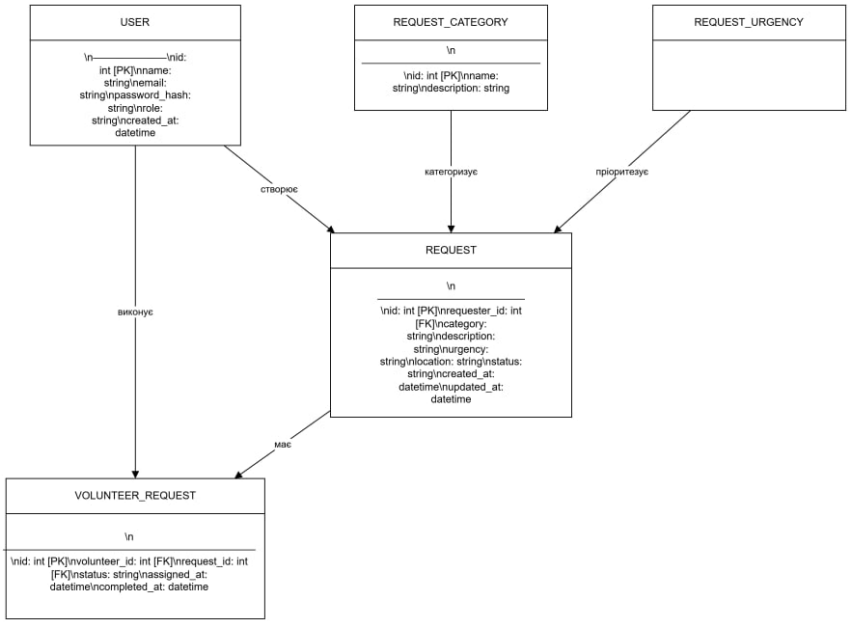


Рис. 2.5 ER-діаграма бази даних

На основі ER-діаграми була розроблена фізична схема бази даних,яка зображена на рисунку 2.6 і детально відображає структуру таблиць, типи даних, первинні та зовнішні ключі, а також індекси для оптимізації запитів.

В процесі фізичного проєктування бази даних було обрано систему управління базами даних SQLite. Цей вибір обумовлений простотою налаштування та обслуговування, відсутністю необхідності в окремому сервері баз даних та достатньою продуктивністю для невеликих та середніх волонтерських організацій. SQLite зберігає всю базу даних у єдиному файлі, що спрощує резервне копіювання та перенесення системи між серверами.

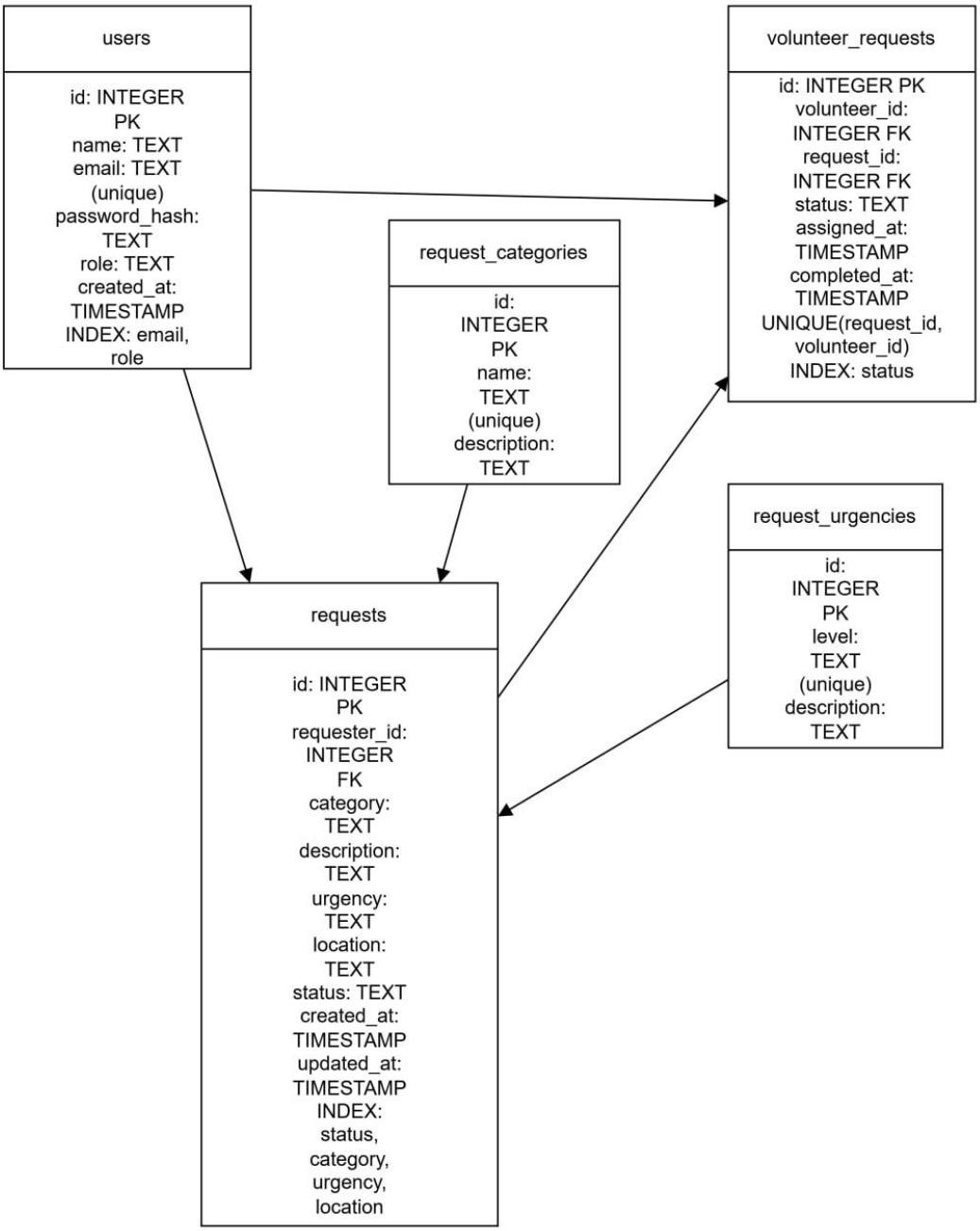


Рис. 2.6 Фізична схема бази даних

Для забезпечення цілісності даних та оптимізації пошуку в базі даних були визначені ключові обмеження та індекси. Первинні ключі (Primary Keys) створені для унікальної ідентифікації записів у кожній таблиці. Зовнішні ключі (Foreign Keys) встановлені для підтримки зв'язків між таблицями та забезпечення референційної цілісності даних. Індекси створені для полів, які часто використовуються в операціях пошуку та сортування, зокрема для полів статусу заявки, категорії та локації. Особлива увага при проєктуванні бази

даних приділялася питанням безпеки. Паролі користувачів зберігаються не у відкритому вигляді, а у формі хешів з використанням алгоритму PBKDF2 та додаванням солі, що захищає систему від витоку чутливих даних навіть у випадку несанкціонованого доступу до бази даних [15].

Для зручності взаємодії з базою даних у коді додатку використовується ORM (Object-Relational Mapping) підхід через SQLAlchemy. Це дозволяє працювати з даними як з об'єктами Python, абстрагуючись від конкретних SQL-запитів. Такий підхід не лише спрощує розробку, але й підвищує рівень безпеки, зменшуючи ризик SQL-ін'єкцій.

База даних спроєктована з урахуванням можливого масштабування системи в майбутньому. Хоча початкова реалізація використовує SQLite, структура даних дозволяє легко мігрувати на більш потужні СУБД, такі як PostgreSQL або MySQL, без суттєвих змін у логіці додатку.

Схема бази даних також передбачає можливість розширення для підтримки додаткових функцій. Наприклад, у майбутньому можливе додавання таблиць для зберігання файлових вкладень до заявок, детальної статистики діяльності волонтерів або налаштувань системних повідомлень.

Таким чином, спроєктована структура бази даних забезпечує надійне зберігання та ефективну обробку інформації в системі волонтерської підтримки, підтримуючи всі необхідні бізнес-процеси та забезпечуючи можливості для подальшого розвитку системи.

2.4 Проєктування інтерфейсу користувача

При проєктуванні інтерфейсу користувача для системи обробки заявок волонтерських організацій було враховано декілька ключових принципів юзабіліті, які забезпечують зручність використання системи для всіх категорій користувачів. Основна концепція дизайну базується на простоті, інтуїтивності та доступності, що особливо важливо у контексті волонтерської діяльності, де швидкість доступу до інформації часто відіграє критичну роль.

Інтерфейс користувача розроблено з урахуванням трьох основних ролей користувачів: заявників (отримувачів допомоги), волонтерів та координаторів. Для кожної категорії користувачів передбачені специфічні функції та відповідні елементи інтерфейсу, які відображають їх потреби та завдання. Така диференціація дозволяє оптимізувати доступ користувачів до найбільш необхідних для них функцій. Для забезпечення швидкого орієнтування в системі було розроблено уніфіковану навігаційну панель, доступну на кожній сторінці сайту. Навігаційний блок містить пункти меню, які змінюються в залежності від ролі користувача, що дозволяє швидко переходити між різними розділами системи. Таке рішення допомагає користувачам швидко знаходити потрібні функції без необхідності вивчати складну структуру сайту [16].

Для забезпечення естетичної привабливості та комфортного сприйняття було обрано нейтральну кольорову гаму з акцентами, що не відволікає від основного контенту. Основними кольорами інтерфейсу є білий фон для основного контенту, темно-синій (#2d3e50) для навігаційної панелі та акцентних елементів, що забезпечує гарний контраст і сприяє легшому зчитуванню інформації. Така кольорова схема також відповідає принципам доступності для користувачів з особливими потребами.

Форми введення даних, які є одним з основних елементів взаємодії з системою, були спроєктовані з дотриманням принципів послідовності та передбачуваності. Всі форми мають чітке маркування полів, зрозумілі підказки та повідомлення про помилки. Обов'язкові поля виділені відповідними позначеннями, а система валідації даних допомагає користувачам уникнути помилок при заповненні форм. Для різних типів даних були обрані відповідні елементи управління: текстові поля для вільного вводу, випадаючі списки для вибору з фіксованого набору опцій, текстові області для введення розширених описів. Така диференціація допомагає користувачам розуміти, який тип інформації очікується в кожному конкретному полі, та робить процес введення даних більш інтуїтивним.

Сторінка реєстрації нових користувачів містить мінімальний набір полів, необхідних для створення облікового запису: ім'я, електронна пошта, пароль та вибір ролі. Це спрощує процес реєстрації та зменшує поріг входу для нових користувачів. Після реєстрації користувачам пропонується увійти в систему, використовуючи створені облікові дані.

Для заявників (отримувачів допомоги) було розроблено зручний інтерфейс створення нових заявок з чіткою структурою та логічним групуванням полів. Форма створення заявки включає вибір категорії допомоги (медикаменти, спорядження, дрони, авто), поле для детального опису потреби, вибір рівня терміновості та вказання місцезнаходження. Такий підхід полегшує процес створення заявок навіть для користувачів з мінімальним досвідом використання веб-систем.

Для волонтерів розроблено інтерфейс перегляду доступних заявок з можливістю фільтрації за різними параметрами: категорією, терміновістю та локацією. Це дозволяє волонтерам швидко знаходити заявки, які відповідають їх можливостям та географічному розташуванню. Для кожної заявки відображається коротка інформація з можливістю взяти її в роботу, що спрощує процес вибору та прийняття рішень.

Для координаторів створено розділ управління заявками з можливістю перегляду, підтвердження або відхилення заявок, які очікують на перевірку. Інтерфейс для кожної заявки містить основну інформацію та кнопки для швидких дій, що дозволяє координаторам ефективно опрацьовувати великий обсяг заявок за мінімальний час.

Для відстеження стану заявок реалізовано статус-бари або текстові індикатори, які чітко відображають поточний статус кожної заявки: "Очікує перевірки", "Прийнято", "Виконується", "Завершено". Це дозволяє всім учасникам процесу отримувати актуальну інформацію про стан заявок без необхідності зайвих комунікацій.

Для забезпечення можливості використання системи на різних пристроях, реалізовано адаптивний дизайн, який коректно відображається як

на десктопних комп'ютерах, так і на мобільних пристроях. Це особливо важливо для волонтерів, які часто працюють у польових умовах і потребують доступу до системи з мобільних телефонів.

Для забезпечення інформативності інтерфейсу без його перевантаження реалізовано систему повідомлень (флеш-повідомлень), які інформують користувачів про результати їхніх дій: успішне створення заявки, помилки при заповненні форм, підтвердження або відхилення заявки тощо [17]. Ці повідомлення допомагають користувачам розуміти, що відбувається в системі, і отримувати зворотний зв'язок на свої дії. Кожна сторінка системи має чітку структуру з заголовками, що допомагає користувачам розуміти, де вони зараз знаходяться і яку функцію виконують. Такий підхід покращує загальну орієнтацію в системі та зменшує когнітивне навантаження на користувачів при вирішенні їхніх завдань у системі обробки заявок волонтерських організацій.

Запропонований дизайн інтерфейсу користувача був протестований на відповідність критеріям юзабіліті та отримав позитивні відгуки від представників різних категорій користувачів. Це дозволяє стверджувати, що розроблений інтерфейс забезпечує ефективне використання системи всіма учасниками процесу волонтерської допомоги.

2.5 Проєктування системи безпеки

Проєктування системи безпеки є критично необхідним компонентом розробки інформаційної системи для обробки заявок волонтерських організацій, оскільки система працює з конфіденційними даними користувачів та чутливою інформацією про потреби та локації. У цьому розділі детально розглядаються заходи безпеки, що були впроваджені для захисту системи від несанкціонованого доступу та забезпечення конфіденційності даних.

Першим рівнем забезпечення безпеки є система автентифікації користувачів. Для доступу до функціоналу системи кожен користувач повинен пройти процедуру реєстрації та подальшої автентифікації. Реєстрація вимагає

введення основних даних: ім'я, електронна пошта, пароль та вибір ролі. Для кожного користувача створюється унікальний обліковий запис, пов'язаний з його електронною поштою, що дозволяє однозначно ідентифікувати користувача в системі.

Для зберігання паролів використовується сучасний алгоритм хешування PBKDF2 з SHA-256, який забезпечує надійний захист від можливого витоку паролів у випадку компрометації бази даних. Реалізація хешування здійснюється через функцію generate_password_hash з бібліотеки werkzeug.security, що є стандартною практикою в Flask-додатках.

Для управління сесіями користувачів та забезпечення безпеки автентифікації використовується розширення Flask-Login, яке дозволяє створювати, підтримувати та завершувати сесії користувачів. Це забезпечує належний контроль доступу до системи та дозволяє реалізувати механізм виходу з системи (logout), що важливо для користувачів, які працюють на спільних комп'ютерах. Вагомим аспектом безпеки є система розмежування прав доступу, що базується на ролях користувачів. Кожен користувач має одну з трьох ролей - заявник, волонтер або координатор - та отримує відповідні права доступу. Перевірка ролей здійснюється на рівні серверної логіки через декоратори @login_required та перевірки current_user.role, що унеможливлює несанкціонований доступ до функцій, не призначених для конкретної ролі.

Для захисту від CSRF-атак (Cross-Site Request Forgery) використовуються вбудовані механізми Flask, які генерують унікальні токени для кожної форми. Ці токени перевіряються сервером при обробці POST-запитів, що запобігає виконанню небажаних дій від імені авторизованого користувача [18].

Захист від SQL-ін'єкцій забезпечується за допомогою ORM SQLAlchemy, яка правильно екранує параметри запитів перед їх виконанням до бази даних. Це дозволяє уникнути найпоширеніших векторів атаки на базу даних та захистити конфіденційність інформації.

Для зберігання конфіденційних даних, таких як секретні ключі, паролі до бази даних тощо, використовується підхід з використанням змінних середовища або конфігураційних файлів, що не входять до системи контролю версій. У конфігурації Flask застосовується секретний ключ (SECRET_KEY), який використовується для підписання сесійних файлів cookie.

Для захисту від XSS-атак (Cross-Site Scripting) у системі використовується автоматичне екранування виводу у шаблонах Jinja2, що є стандартним механізмом у Flask. Це захищає від можливого впровадження шкідливого скрипту через вміст, що вводиться користувачами.

Система забезпечує належну валідацію всіх вхідних даних як на стороні клієнта (через HTML5-атрибути required, type="email" тощо), так і на стороні сервера, що запобігає введенню некоректних або потенційно небезпечних даних. Валідація на сервері є основним захисним механізмом, оскільки клієнтську валідацію можна обійти. Для забезпечення конфіденційності даних при передачі між клієнтом і сервером рекомендується використовувати протокол HTTPS, що забезпечує шифрування трафіку. У виробничому середовищі необхідно налаштувати SSL/TLS-сертифікат для домену, на якому розміщується система, та переконатися, що всі запити перенаправляються на захищене з'єднання.

Проєктування системи безпеки передбачає також регулярне оновлення всіх компонентів системи, включаючи фреймворк Flask, бібліотеки та підсистеми, для захисту від відомих вразливостей. Рекомендується налаштувати автоматичне сповіщення про нові вразливості у використовуваних компонентах.

Система реєстрації подій (логування) дозволяє відстежувати всі критичні операції в системі, такі як входи користувачів, створення та зміна статусів заявок, що дозволяє виявити потенційні проблеми безпеки або несанкціонований доступ. Ці логи повинні зберігатися у захищеному місці з обмеженим доступом. Для захисту від можливої втрати даних рекомендується налаштувати регулярне резервне копіювання бази даних та іншої важливої

інформації. Копії повинні зберігатися в безпечному місці, окремо від основної системи [19].

Розроблена система безпеки забезпечує комплексний захист даних користувачів та функціональності системи, що є необхідною умовою для роботи з чутливою інформацією у сфері волонтерської допомоги. Впроваджені механізми відповідають сучасним стандартам безпеки веб-додатків та сприяють зміцненню довіри користувачів до системи.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ СИСТЕМИ

3.1 Архітектура реалізації

Процес реалізації інформаційної системи для обробки заявок волонтерських організацій базується на чіткій архітектурній концепції, яка забезпечує взаємодію всіх компонентів системи та ефективне виконання функціональних вимог. В основу реалізації покладено класичний патерн Model-View-Controller (MVC), який дозволяє логічно розділити код на окремі компоненти, що полегшує розробку, тестування та подальше підтримування системи. Структурно архітектура реалізації включає серверну частину (Backend), базу даних SQLite та клієнтську частину (Frontend), що схематично зображено на рисунку 3.1.

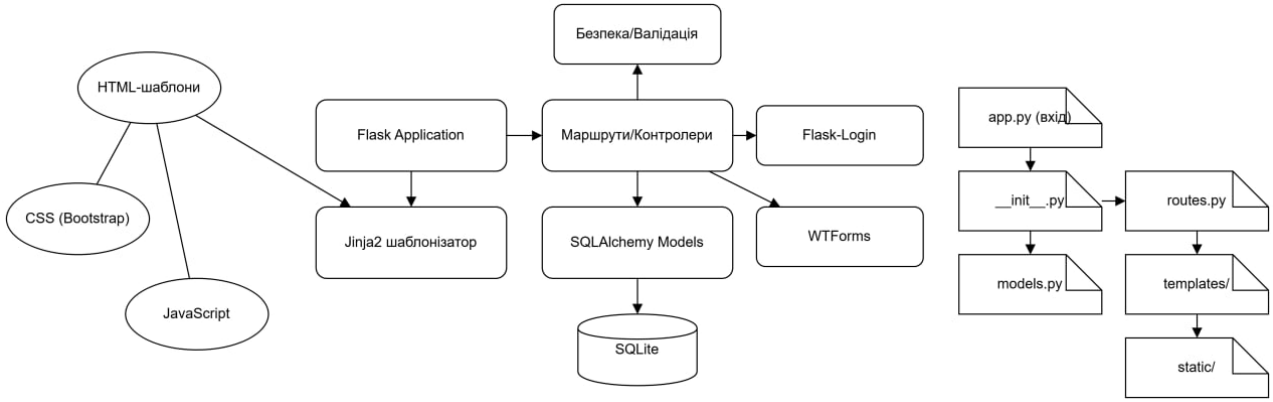


Рис. 3.1 Архітектура реалізації системи

Структурно архітектура реалізації включає серверну частину (Backend), розроблену на базі Python з використанням мікрофреймворку Flask, базу даних SQLite для зберігання та управління даними, та клієнтську частину (Frontend), що базується на HTML з використанням фреймворку Bootstrap для створення адаптивного інтерфейсу користувача.

Проєкт організовано у модульну структуру, яка сприяє кращій організації коду та полегшує підтримку системи. Основний пакет додатка

містить файл ініціалізації, моделі даних та маршрутизацію. Окремо організовано директорії з HTML-шаблонами для відображення сторінок та з статичними ресурсами, такими як CSS-стилі.

Взаємодія між компонентами системи реалізована за схемою MVC, де модель представлено класами, які використовують SQLAlchemy для об'єктно-реляційного мапінгу, представлення реалізовано через HTML-шаблони, що використовують шаблонізатор Jinja2, а контролер реалізовано маршрутами, які обробляють запити користувачів. Взаємодія між компонентами системи реалізована за схемою MVC, що наочно представлено на рисунку 3.2.

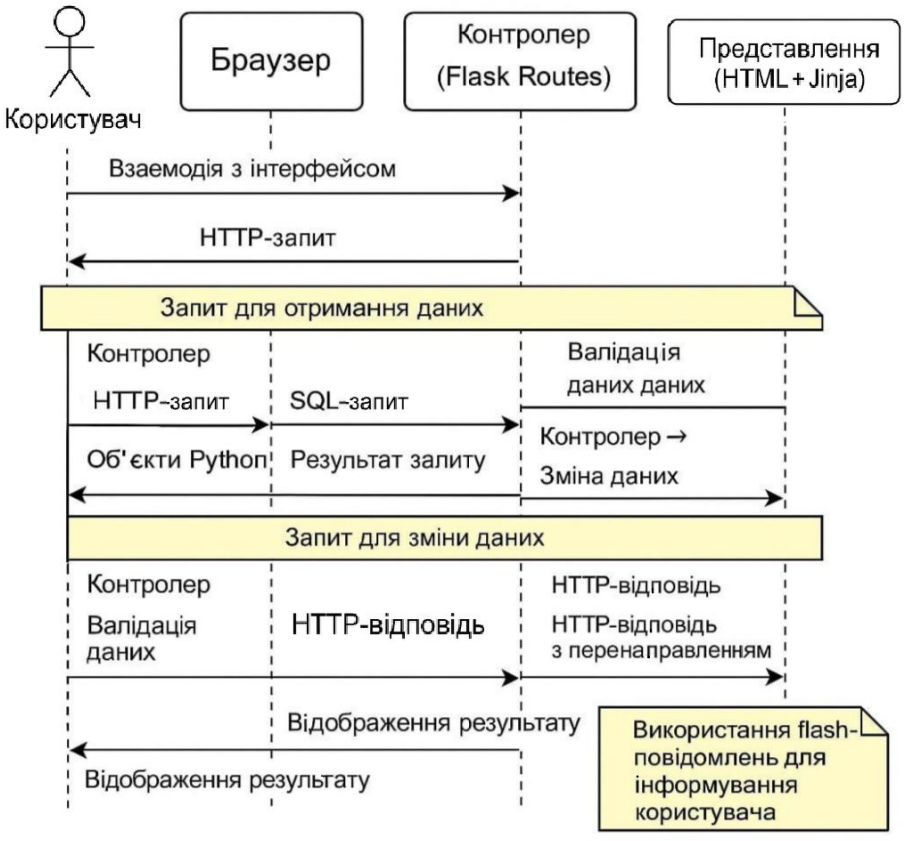


Рис. 3.2 Діаграма потоку запитів в MVC-архітектурі

Типовий потік обробки запиту в системі починається з взаємодії користувача з веб-інтерфейсом, що генерує HTTP-запит. Flask обробляє запит і передає його відповідному маршруту, який взаємодіє з моделями для отримання або зміни даних у базі даних. Потім дані передаються у відповідний

HTML-шаблон для відображення, і готова HTML-сторінка надсилається користувачу як відповідь.

Для реалізації механізмів авторизації та управління сесіями використано розширення Flask-Login, яке забезпечує функціональність управління сесіями користувачів, перевірку прав доступу та перенаправлення неавторизованих користувачів на сторінку входу.

Для роботи з базою даних у системі використовується ORM SQLAlchemy, що дозволяє працювати з даними у вигляді об'єктів Python, абстрагуючись від безпосередніх SQL-запитів. Моделі даних представляють основні сутності системи та їх взаємозв'язки, а також містять методи для роботи з цими даними. Вагомим аспектом архітектури реалізації є система обробки помилок та сповіщень користувачів. Для цього використовується механізм flash-повідомлень Flask, який дозволяє відображати інформаційні повідомлення користувачам після виконання певних дій.

Архітектура реалізації спроєктована з урахуванням можливості подальшого розширення функціональності. Модульна структура проєкту дозволяє легко додавати нові компоненти без суттєвої переробки існуючого коду. Наприклад, для додавання нових типів заявок чи розширення функціоналу ролей достатньо внести зміни в моделі даних та відповідні контролери, не змінюючи загальну структуру системи. Необхідною особливістю реалізованої архітектури є її адаптивність до змін вимог, що є критично головним для волонтерських систем, які функціонують у динамічному середовищі та часто потребують швидкого реагування на нові потреби.

Таким чином, архітектура реалізації інформаційної системи для обробки заявок волонтерських організацій забезпечує ефективне виконання всіх функціональних вимог, підтримує взаємодію між різними компонентами системи та надає можливості для подальшого розширення та вдосконалення.

3.2 Реалізація серверної частини

Серверна частина інформаційної системи є основою, що забезпечує функціонування всіх бізнес-процесів волонтерської організації, управління даними та логіку взаємодії між різними типами користувачів. Реалізація серверної частини ґрунтується на принципах модульності, безпеки та масштабованості, що дозволяє створити надійну та гнучку систему для обробки заявок.

В якості основи для серверної частини було обрано фреймворк Flask, який є легким мікрофреймворком для Python, що надає необхідні інструменти для створення веб-додатків без накладання жорстких обмежень на структуру проєкту. Цей вибір обумовлений простотою використання, гнучкістю та високою продуктивністю Flask, а також обширною екосистемою розширень, що дозволяє легко інтегрувати додаткову функціональність. Структура проєкту організована відповідно до модульного підходу, що сприяє кращій організації коду та полегшує подальшу підтримку системи. Основу серверної частини становлять такі ключові файли: точка входу, що запускає сервер розробки Flask; ініціалізація додатку, підключення розширень та створення контексту додатку; визначення моделей даних та їх взаємозв'язків; визначення маршрутів та обробників запитів.

У процесі ініціалізації додатку відбувається кілька головних кроків: створення екземпляра Flask із вказівкою шляхів до директорій із шаблонами та статичними файлами; налаштування секретного ключа для підписання сесій; налаштування з'єднання з базою даних SQLite; ініціалізація розширень для роботи з базою даних (SQLAlchemy) та управління авторизацією (Flask-Login); реєстрація Blueprint, що містить маршрути додатку; створення таблиць бази даних на основі визначених моделей.

Моделі даних відображають основні сутності системи та їх взаємозв'язки. Для реалізації моделей використовується SQLAlchemy — потужний ORM (Object-Relational Mapper), що дозволяє працювати з базою

даних на рівні об'єктів Python, абстрагуючись від безпосередньої роботи з SQL-запитами. Основними моделями системи є: User — представляє користувача системи, що може виступати в одній із трьох ролей: заявник, волонтер або координатор; Request — представляє заявку на допомогу, створену заявником; VolunteerRequest — зв'язує волонтера із заявкою, яку він взяв на виконання.

Для автентифікації користувачів використовується Flask-Login — розширення, що надає зручний інтерфейс для управління сесіями користувачів. Для завантаження користувача за його ідентифікатором реалізовано відповідну функцію.

Основна бізнес-логіка та обробка запитів реалізована через визначення маршрутів для різних функцій системи. Для організації маршрутів використовується Blueprint, що дозволяє логічно групувати маршрути та спрощує структуру коду. В системі реалізовано маршрути для реєстрації та автентифікації користувачів, роботи із заявками для заявників, роботи із заявками для волонтерів, роботи із заявками для координаторів.

Особлива увага при реалізації серверної частини приділена безпеці. Для захисту паролів використовується хешування з сіллю. Для захисту маршрутів від несанкціонованого доступу використовується відповідний декоратор, що перенаправляє неавторизованих користувачів на сторінку входу. Крім того, для кожного маршруту реалізовано перевірку ролі користувача, щоб забезпечити, що користувач має доступ лише до функцій, які відповідають його ролі. Для передачі повідомлень користувачам про результати їхніх дій використовується механізм flash-повідомлень Flask. Необхідною частиною серверної реалізації є функціональність фільтрації заявок для волонтерів, яка дозволяє фільтрувати доступні заявки за категорією, терміновістю та локацією. Для координаторів реалізовано функціональність аналітики активності волонтерів, що дозволяє відстежувати кількість взятих та завершених заявок для кожного волонтера.

Таким чином, реалізована серверна частина забезпечує основну функціональність системи обробки заявок волонтерських організацій: реєстрацію та автентифікацію користувачів; створення та перегляд заявок для заявників; пошук, фільтрацію та взяття в роботу заявок для волонтерів; модерацію заявок та моніторинг активності волонтерів для координаторів. Розроблена серверна частина відповідає принципам безпеки, гнучкості та масштабованості, що дозволяє ефективно використовувати систему для координації волонтерської допомоги в українських реаліях.

3.3 Реалізація клієнтської частини

Клієнтська частина інформаційної системи відіграє ключову роль у забезпеченні ефективної взаємодії користувачів з функціоналом волонтерської платформи.

Успішна реалізація інтерфейсу користувача визначає зручність використання системи та її привабливість для всіх категорій користувачів. На основі розробленого в розділі 2.4 проєкту інтерфейсу було реалізовано клієнтську частину, яка забезпечує інтуїтивно зрозумілий та функціональний доступ до всіх можливостей системи.

Технологічно клієнтська частина базується на використанні традиційних веб-технологій: HTML для структурування контенту, CSS для стилізації та візуального оформлення, а також мінімального JavaScript для інтерактивності. Для забезпечення адаптивності та сучасного вигляду інтерфейсу було використано фреймворк Bootstrap, що дозволяє створювати інтерфейси, які коректно відображаються на пристроях з різними розмірами екранів.

Центральним елементом клієнтської частини є система HTML-шаблонів, реалізованих за допомогою шаблонізатора Jinja2, що інтегрований у Flask. Така структура забезпечує гнучкість розробки та можливість повторного використання компонентів інтерфейсу. Базовий шаблон (base.html) містить загальну структуру сторінки, включаючи навігаційну

панель та область для відображення повідомлень користувачу. Інші шаблони розширюють базовий, додаючи специфічний контент для різних функціональних сторінок. Особлива увага при реалізації клієнтської частини приділена навігації, яка динамічно змінюється в залежності від ролі авторизованого користувача. Для заявників доступні пункти меню для створення нових заявок та перегляду власних заявок. Волонтери мають доступ до перегляду доступних заявок та управління своїми поточними задачами. Координатори отримують можливість переходити до розділів модерації заявок та моніторингу активності волонтерів. Така диференціація забезпечує інтуїтивний доступ до найбільш релевантних функцій для кожної категорії користувачів.

Для неавторизованих користувачів реалізовано просту головну сторінку з інформацією про систему та можливістю реєстрації або входу. Форми реєстрації та входу зображено на рисунках 3.3 – 3.5 і спроєктовані мінімалістично, з фокусом на збір лише необхідної інформації, що знижує поріг входу для нових користувачів .

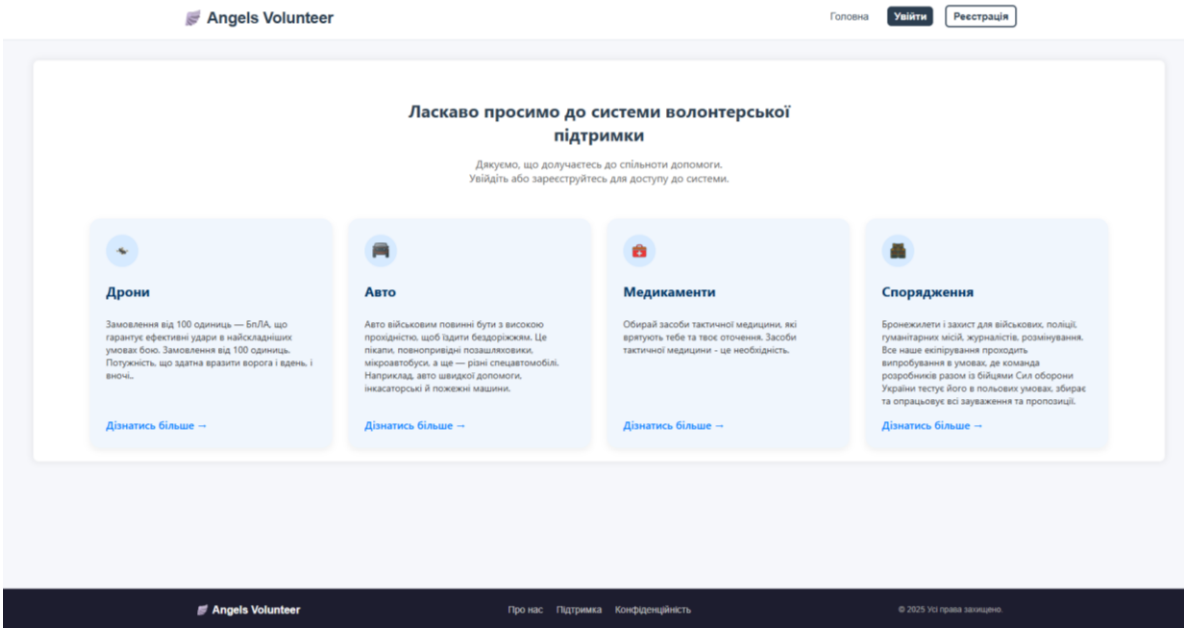


Рис. 3.3 Головна сторінка системи для неавторизованих користувачів

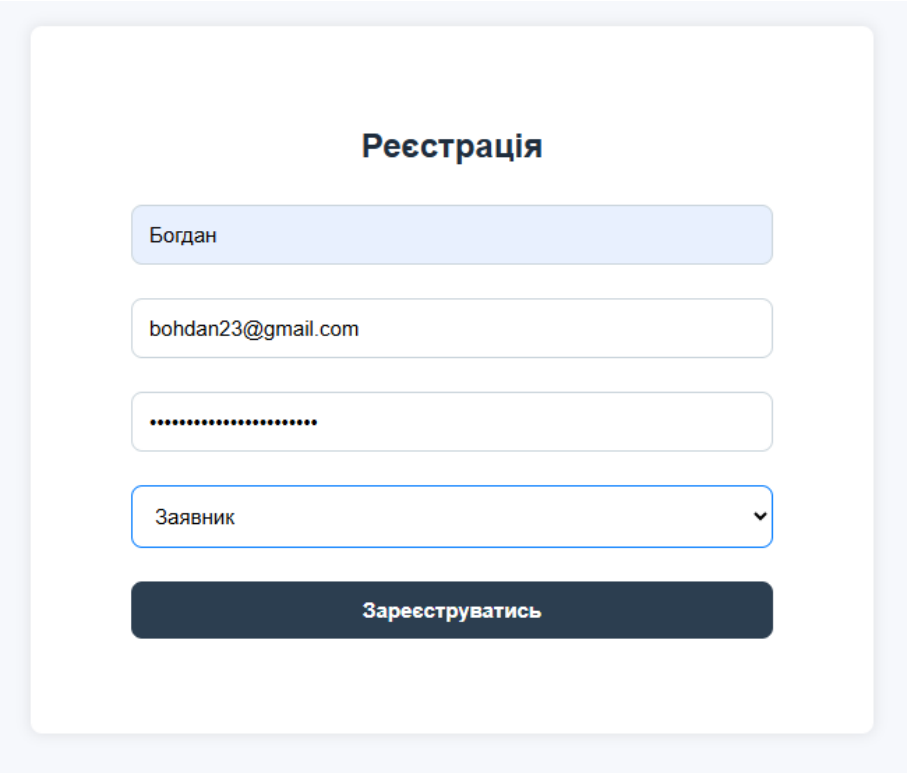


Рис. 3.4 Форма реєстрації нового користувача

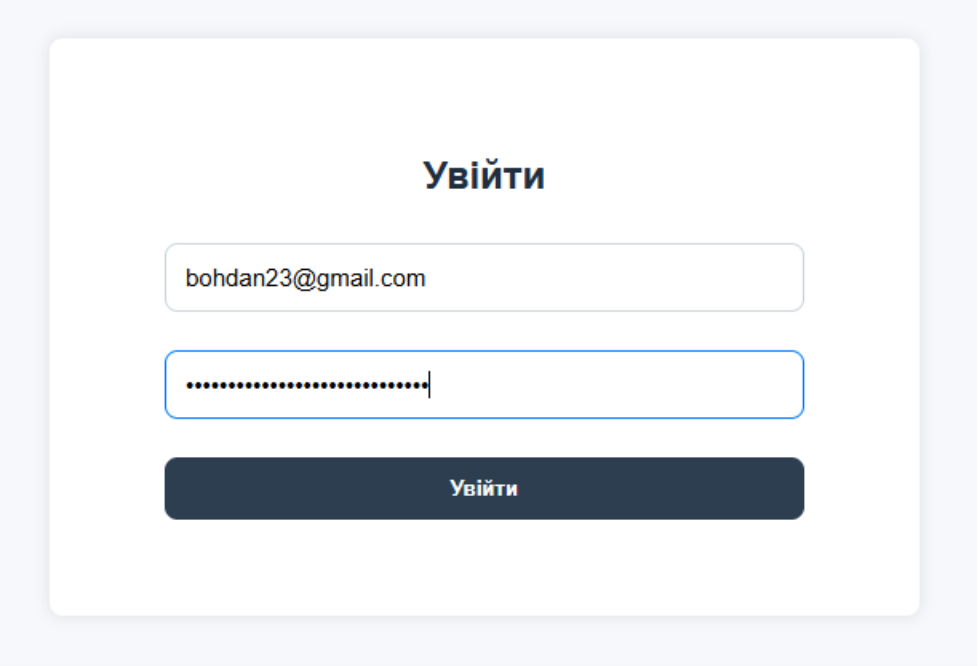


Рис. 3.5 Форма входу в систему

Сторінка особистого кабінету користувача реалізована з урахуванням ролі та надає швидкий доступ до основних функцій. Заявники бачать кнопки

для створення нових заявок та перегляду існуючих, волонтери – для пошуку доступних заявок та управління взятими у роботу, а координатори – для управління заявками та моніторингу волонтерів зображено на рисунку 3.6.

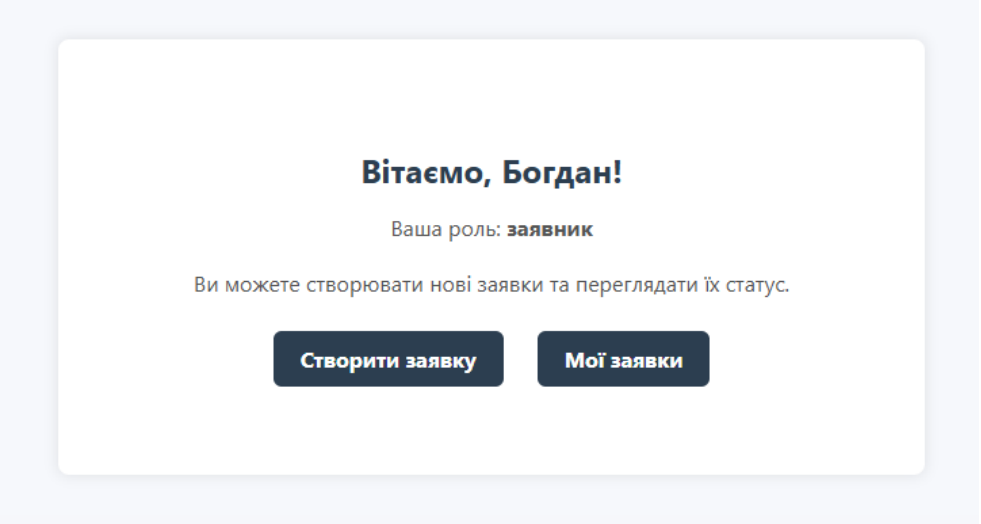


Рис. 3.6 Особистий кабінет користувача з роллю "заявник"

Форма створення заявки реалізована з використанням різних типів елементів введення: випадаючих списків для вибору категорії та терміновості, текстового поля для опису та вказання локації зображено на рисунку 3.7.

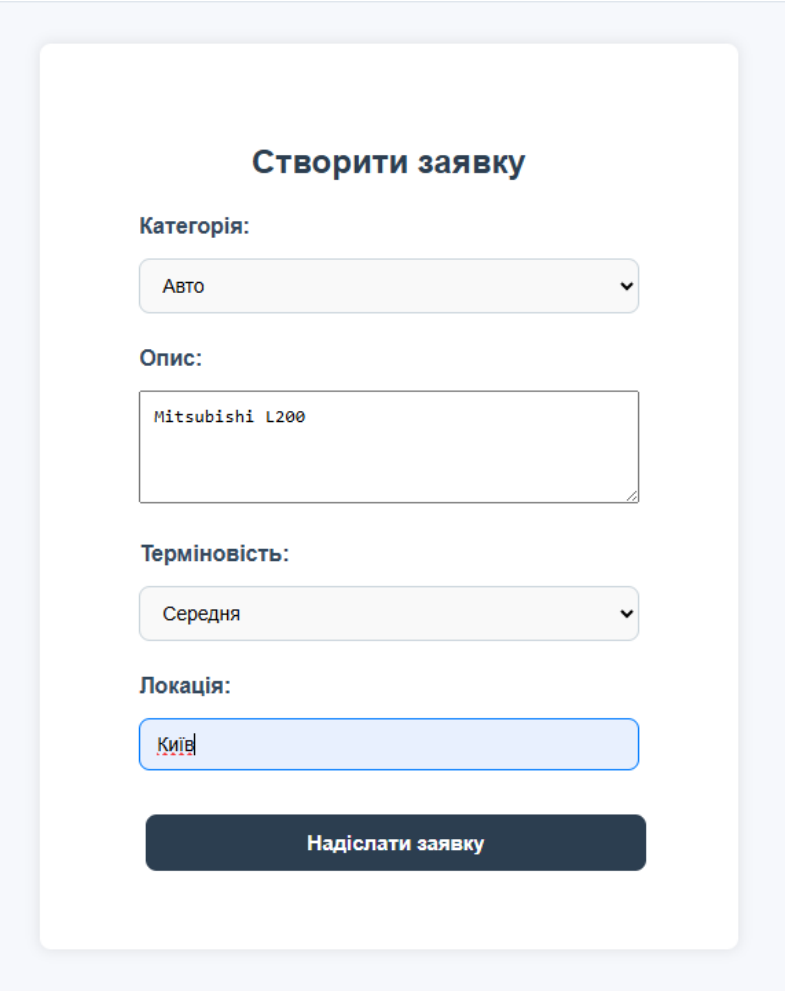


Рис. 3.7 Форма створення нової заявки

Інтерфейс розроблений таким чином, щоб мінімізувати можливі помилки користувача під час заповнення форми. Обов'язкові поля позначені атрибутом required, що забезпечує валідацію на стороні клієнта.

Для перегляду заявок розроблено уніфікований інтерфейс, який адаптується під контекст використання: заявники бачать власні заявки з їх статусами, волонтери – доступні для взяття у роботу заявки або взяті ними заявки з можливістю позначення як завершених, координатори – заявки, що очікують на підтвердження. Кожна заявка візуально представлена в форматі картки, що зображено на рисунку 3.8 та містить ключову інформацію та відповідні кнопки для дій .

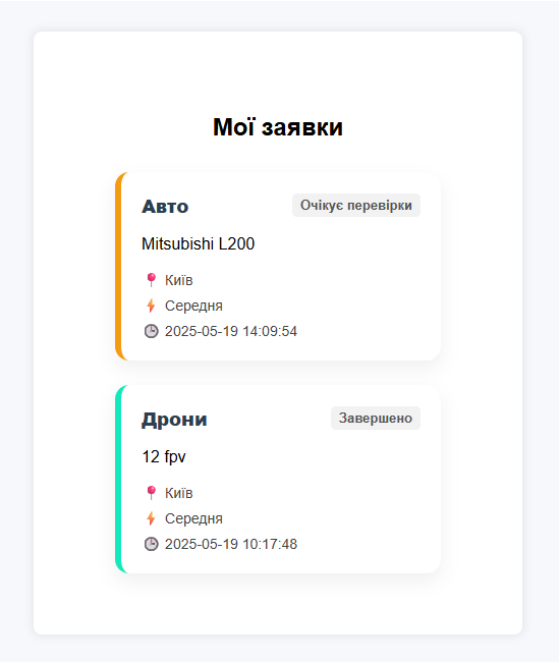


Рис. 3.8 Сторінка перегляду заявок користувачем з роллю "заявник"

Для волонтерів реалізовано зручну систему фільтрації доступних заявок за категорією, терміновістю та локацією. Форма фільтрації розміщена над списком заявок і дозволяє легко обирати параметри пошуку. Підтримка методу GET для фільтрів дозволяє зберігати та ділитися посиланнями на відфільтровані списки заявок.

Для координаторів реалізовано табличне представлення статистики волонтерів, що дозволяє швидко оцінити активність кожного волонтера за кількістю взятих та завершених заявок. Такий підхід сприяє ефективному управлінню волонтерським ресурсом та мотивації активних учасників. Візуальне оформлення інтерфейсу відповідає сучасним трендам веб-дизайну, з акцентом на мінімалізм та функціональність. Основна кольорова гама включає білий фон для контентних блоків, темно-синій (#2d3e50) для навігаційної панелі та акцентних елементів. Такий вибір кольорів забезпечує добрий контраст та читабельність контенту, а також створює професійний вигляд системи.

Інтерфейс реалізований з урахуванням принципів адаптивного дизайну: елементи форм, таблиці та списки автоматично підлаштовуються під розмір екрану пристрою, що забезпечує зручне використання системи як на

настільних комп'ютерах, так і на мобільних пристроях. Це особливо важливо для волонтерів, які часто працюють у польових умовах і потребують доступу до системи з різних пристроїв.

Для забезпечення інформаційної підтримки користувачів реалізовано систему повідомлень (flash-повідомлення), яка відображає результати виконаних дій та інформує про можливі помилки. Візуально повідомлення виділені кольором, що зображено на рисунку 3.9 та розміщене у верхній частині контентної області, що гарантує його помітність для користувачів.

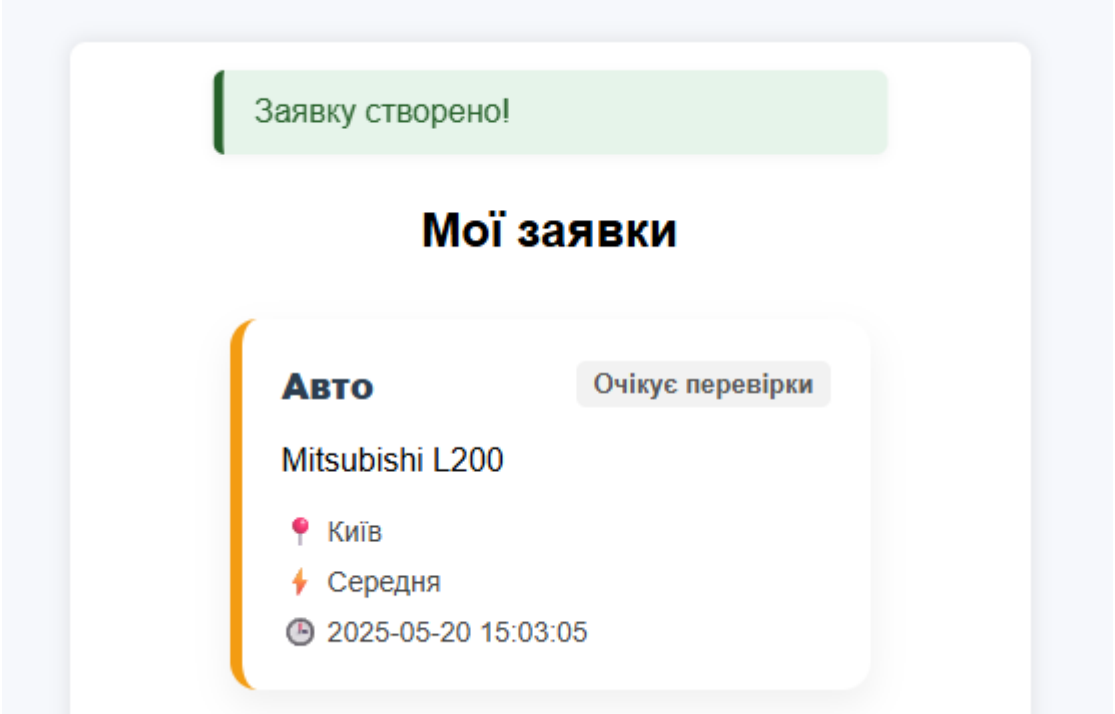


Рис. 3.9 Сторінка зі списком заявок та повідомленням про успішне створення нової заявки

Вагомим аспектом клієнтської частини є забезпечення безпеки. Всі форми з методом POST включають захист від CSRF-атак через приховані токени, що генеруються на стороні сервера. Вхідні дані проходять валідацію як на стороні клієнта (через HTML5-атрибути), так і на стороні сервера, що забезпечує цілісність та безпеку даних.

У процесі розробки клієнтського інтерфейсу використовувався ітеративний підхід з постійним тестуванням на різних пристроях та в різних

браузерах для забезпечення максимальної сумісності. Отримані результати демонструють, що реалізований інтерфейс користувача повністю відповідає вимогам, сформульованим на етапі проєктування, та забезпечує зручну та ефективну взаємодію з системою для всіх категорій користувачів.

Реалізована клієнтська частина демонструє оптимальний баланс між функціональністю, естетикою та зручністю використання, що є критично необхідним для успішного впровадження системи у реальну волонтерську діяльність. Використання сучасних веб-технологій та принципів дизайну інтерфейсів забезпечує привабливість системи для користувачів та її ефективність у вирішенні практичних завдань волонтерських організацій.

3.4 Демонстрація функціональності системи

Розроблена інформаційна система для обробки заявок волонтерських організацій реалізує повний цикл управління заявками від їх створення заявниками до виконання волонтерами. У цьому розділі буде продемонстрована функціональність системи в контексті основних процесів та різних ролей користувачів. Основними користувачами системи є три типи учасників: заявники (особи, що потребують допомоги), волонтери (виконавці) та координатори (представники волонтерських організацій). Кожна роль має свій набір функцій та відповідний інтерфейс, що зображено на рисунку 3.10 та забезпечує ефективну роботу в межах відповідних повноважень.

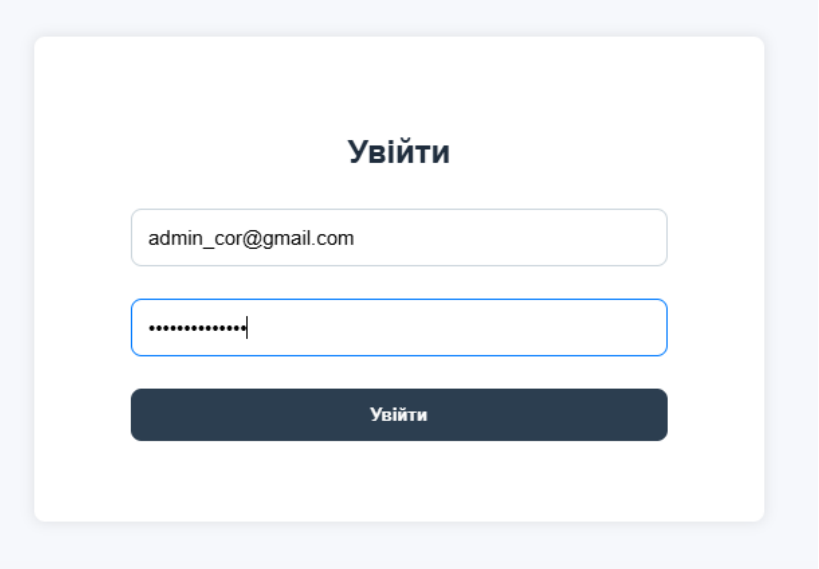


Рис. 3.10 Форма входу для користувача з роллю "координатор"

Початок роботи з системою починається з реєстрації нового користувача. Процес реєстрації передбачає введення базової інформації (ім'я, електронна пошта, пароль) та вибір ролі в системі. Після успішної реєстрації користувач отримує доступ до функціоналу відповідно до обраної ролі.

Для заявників ключовою функцією є створення нових заявок на допомогу. Інтерфейс створення заявки забезпечує можливість вибору категорії допомоги (медикаменти, спорядження, дрони, автотранспорт), введення детального опису потреби, вибору рівня терміновості та визначення географічної локації. Після відправлення заявка отримує статус "Очікує перевірки" та стає доступною для модерації координаторами.

Заявники мають доступ до перегляду власних заявок, де вони можуть відстежувати їх поточний статус (очікує перевірки, прийнято, виконується, завершено, відхилено). Такий зворотний зв'язок забезпечує прозорість процесу та дозволяє заявникам розуміти, на якому етапі знаходиться виконання їхніх запитів.

Координатори виконують ключову роль модерації заявок. Через спеціальний інтерфейс вони мають доступ до всіх заявок, що очікують перевірки. Для кожної заявки координатор може ознайомитися з детальною інформацією та прийняти рішення про підтвердження або відхилення.

Підтверджені заявки отримують статус "Прийнято" та стають доступними для волонтерів, що зображено на рисунках 3.11-3.13.

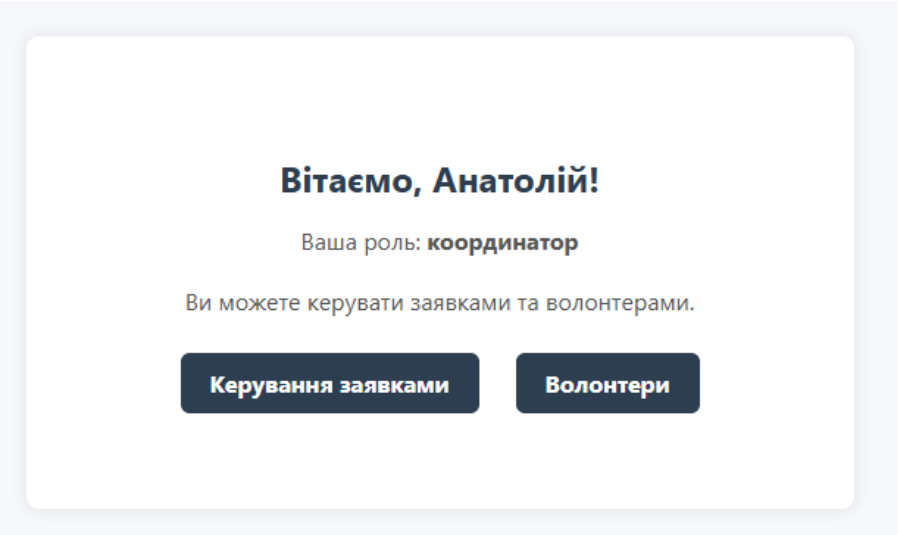


Рис. 3.11 Особистий кабінет користувача з роллю "координатор"

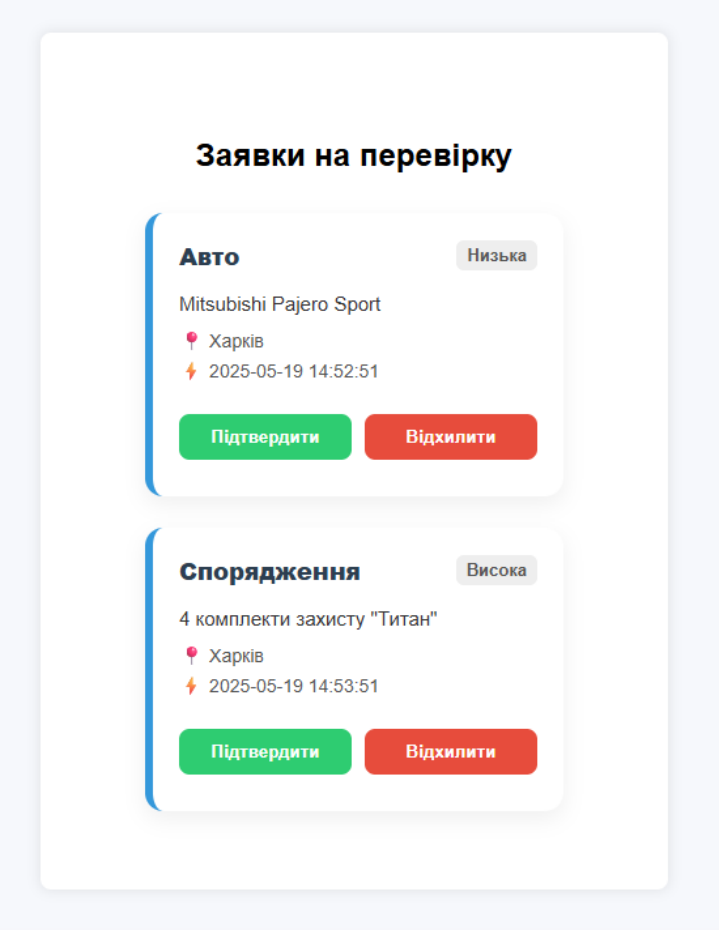


Рис. 3.12 Інтерфейс координатора для модерації заявок, що очікують перевірки

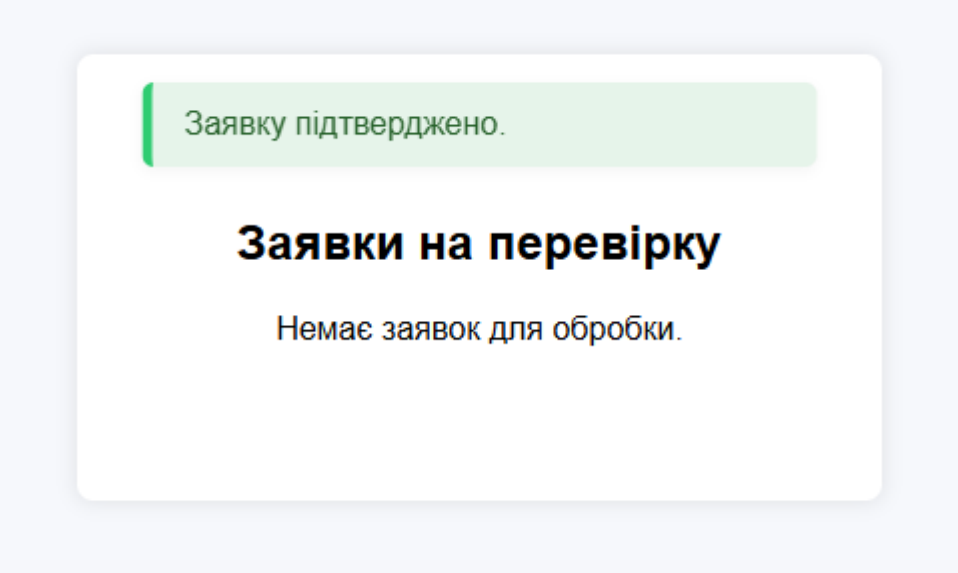


Рис. 3.13 Сповіщення про підтвердження заявки координатором

Крім модерації заявок, координатори мають доступ до аналітичної інформації про активність волонтерів. Спеціальний розділ відображає статистику для кожного волонтера: кількість взятих у роботу заявок та кількість успішно завершених. Ця функціональність важлива і зображена на рисунку 3.14, що дозволяє оцінювати ефективність роботи волонтерів та приймати управлінські рішення на основі об'єктивних даних.

Рис. 3.14 Аналітичний інтерфейс координатора для моніторингу активності волонтерів

Волонтери отримують доступ до списку підтверджених заявок, які потребують виконання. Для зручності пошуку реалізовано систему фільтрації заявок за категорією, терміновістю та географічною локацією. Це дозволяє волонтерам швидко знаходити заявки, які відповідають їхнім можливостям та компетенціям, що зображено на рисунку 3.15.

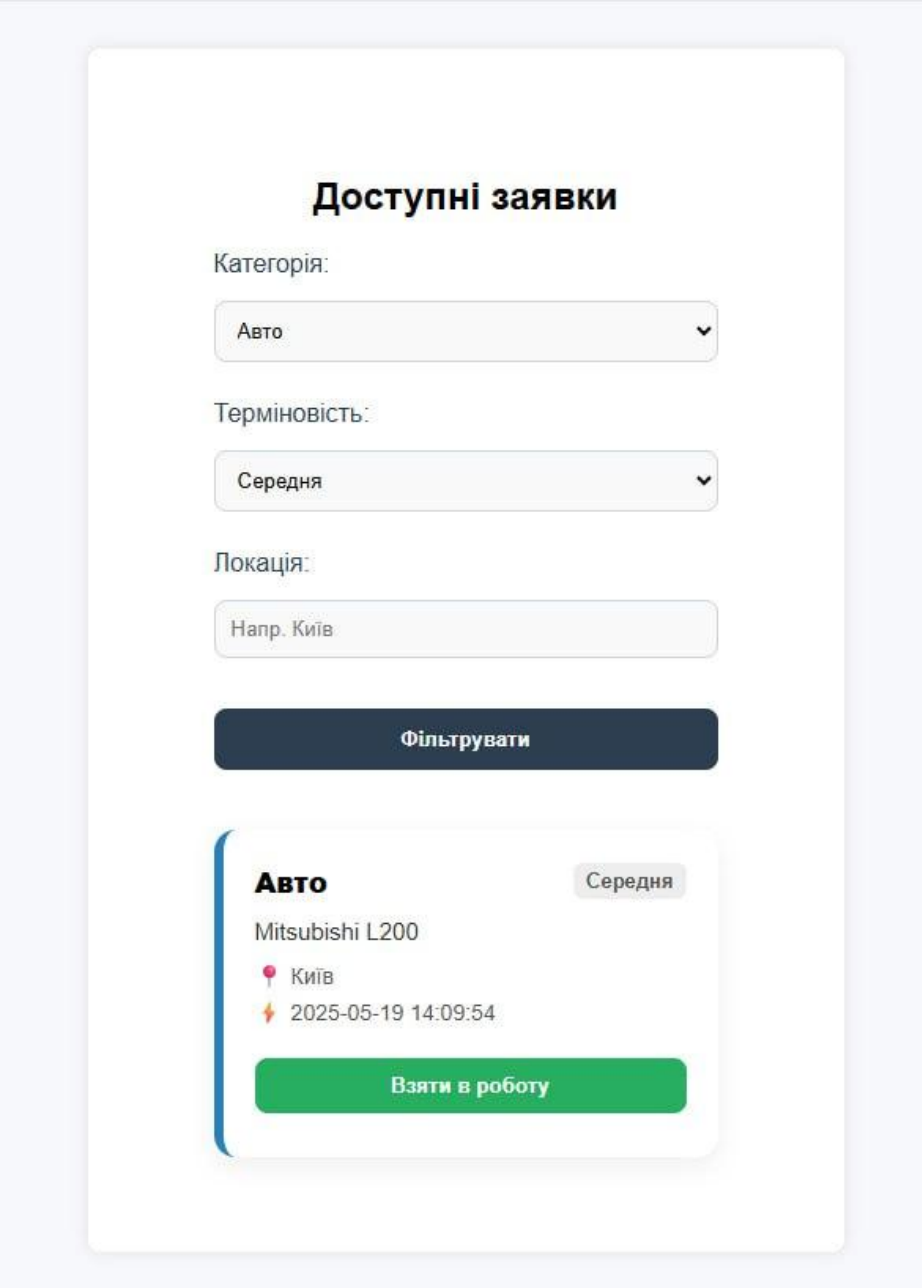


Рис. 3.15 Інтерфейс волонтера для пошуку та фільтрації доступних заявок

Обравши підходящу заявку, волонтер може взяти її в роботу, після чого вона отримує статус "Виконується" та закріплюється за конкретним волонтером. Система забезпечує, що одна заявка не може бути взята в роботу кількома волонтерами одночасно, що запобігає дублюванню зусиль.

Взяті в роботу заявки відображаються в особистому кабінеті волонтера в розділі "Мої задачі". Після виконання заявки волонтер може позначити її як завершену, чим змінює її статус на "Завершено". Така система статусів забезпечує чіткий контроль за усіма етапами життєвого циклу заявки.

Для всіх користувачів реалізовано зручний особистий кабінет, який адаптується під роль користувача та надає швидкий доступ до найбільш релевантних функцій. Інтуїтивна навігація полегшує орієнтування в системі та забезпечує швидкий доступ до потрібних розділів.

Інтерфейс системи реалізовано з дотриманням принципів адаптивного дизайну, що забезпечує коректне відображення на різних пристроях – від настільних комп'ютерів до мобільних телефонів. Це особливо важливо для польових умов, де волонтери можуть потребувати доступу до системи через мобільні пристрої.

Система зворотного зв'язку реалізована через механізм flash-повідомлень, які інформують користувачів про результати їхніх дій: успішне створення заявки, зміну статусу, помилки заповнення форм тощо. Такі повідомлення підвищують зрозумілість інтерфейсу та покращують досвід користування системою.

Демонстрація функціональності системи підтверджує, що реалізована інформаційна система забезпечує комплексну підтримку процесів обробки заявок у волонтерських організаціях. Вона надає зручні інструменти для всіх категорій користувачів, забезпечуючи прозорість, ефективність та контрольованість процесів надання волонтерської допомоги.

Реалізована система має інтуїтивно зрозумілий інтерфейс, що не потребує спеціального навчання користувачів, та може бути швидко впроваджена в діяльність волонтерських організацій різного масштабу. Модульна архітектура системи забезпечує можливість подальшого розширення функціональності відповідно до зростаючих потреб волонтерського сектору.

РОЗДІЛ 4. ТЕСТУВАННЯ ТА ОЦІНКА

4.1 Методологія тестування

Тестування інформаційної системи для обробки заявок волонтерських організацій є критично необхідним етапом розробки, який забезпечує належну якість, надійність та відповідність рішення заявленим вимогам. Процес тестування для даної системи був спроєктований таким чином, щоб охопити всі аспекти функціональності та різні сценарії використання, враховуючи специфічні потреби різних категорій користувачів — заявників, волонтерів та координаторів.

Для забезпечення комплексного підходу до перевірки працездатності системи було застосовано багаторівневу методологію тестування, яка включає різні типи перевірок. Основними рівнями тестування виступали: модульне тестування (unit testing), інтеграційне тестування, функціональне тестування та користувацьке тестування прийнятності. Такий підхід дозволяє послідовно виявляти та усувати недоліки на кожному рівні абстракції, починаючи від окремих компонентів і закінчуючи системою в цілому.

На етапі модульного тестування у фокусі уваги знаходилась перевірка коректності роботи окремих функціональних модулів та класів системи. Для серверної частини, розробленої на Python з використанням Flask, було створено набір тестів, які перевіряли базову функціональність моделей даних, механізми автентифікації, валідацію форм та бізнес-логіку обробки заявок. Модульні тести запускалися автоматично при кожній суттєвій зміні коду, що дозволяло швидко виявляти та виправляти регресії.

При розробці модульних тестів використовувався фреймворк unittest, який є стандартним для Python-проєктів. Кожен тест був структурований за принципом "потрійного А" (Arrange-Act-Assert): підготовка початкових даних, виконання дії, що тестується, та перевірка результату. Для тестування

компонентів, що взаємодіють з базою даних, використовувалася тестова SQLite база даних в пам'яті, що забезпечувало ізольованість тестів та швидкість їх виконання.

Інтеграційне тестування було спрямоване на перевірку коректної взаємодії між різними компонентами системи. В контексті волонтерської платформи особливу увагу приділяли інтеграції між моделями даних та контролерами, а також між серверною та клієнтською частинами додатку. Інтеграційні тести симулювали повний цикл обробки запитів користувачів — від створення нової заявки заявником до її виконання волонтером та контролю координатором.

Для тестування API-ендпоінтів використовувався клієнт для тестування Flask, який дозволяє симулювати HTTP-запити до додатку та перевіряти отримані відповіді. Це дозволило забезпечити правильну роботу маршрутизації, валідації вхідних даних та генерації відповідей сервера. При тестуванні більш складних сценаріїв, які включали кілька взаємопов'язаних запитів, використовувалися фікстури для створення попереднього стану системи.

Функціональне тестування мало на меті перевірку відповідності системи функціональним вимогам, визначеним на етапі аналізу та проєктування. Для кожної основної функції системи (реєстрація, створення заявок, їх модерація, призначення волонтерам тощо) були розроблені тест-кейси, які описували вхідні дані, очікувані результати та критерії успішності. Функціональні тести виконувалися як мануально, так і з використанням автоматизованих інструментів для перевірки web-додатків.

Для автоматизації функціонального тестування використовувався фреймворк Selenium, який дозволяє програмно взаємодіяти з веб-браузером та імітувати дії користувача. Це допомогло перевірити не тільки базову функціональність, але й складні сценарії використання системи різними категоріями користувачів. Автоматизовані тести запускалися на різних браузерах для забезпечення крос-браузерної сумісності системи.

Тестування безпеки було виділено в окремий напрямок через чутливість даних, що обробляються в системі волонтерської підтримки. Було проведено перевірку на наявність поширених вразливостей веб-додатків, таких як SQL-ін'єкції, XSS (Cross-Site Scripting), CSRF (Cross-Site Request Forgery) та недоліки автентифікації. Для цього використовувалися як спеціалізовані інструменти аналізу безпеки, так і ручне тестування потенційно вразливих частин системи.

Окрема увага приділялася тестуванню продуктивності системи, яке включало перевірку швидкодії та стабільності роботи при різних рівнях навантаження. Для симуляції одночасного доступу багатьох користувачів використовувалися інструменти навантажувального тестування, які дозволяли оцінити час відгуку системи та її поведінку при піковому навантаженні. Результати цих тестів допомогли оптимізувати конфігурацію сервера та структуру БД для забезпечення прийнятної продуктивності.

Головною складовою методології тестування було тестування локалізації, адже система розроблена специфічно для українських волонтерських організацій. Перевірялася коректність відображення українських символів, правильність перекладу інтерфейсу та узгодженість термінології. Це дозволило забезпечити комфортну роботу користувачів з системою на рідній мові та врахувати культурні особливості цільової аудиторії.

Тестування доступності (accessibility testing) було спрямоване на перевірку використання системи людьми з різними можливостями. Перевірялася контрастність кольорів, наявність альтернативних текстів для зображень, можливість навігації за допомогою клавіатури та коректна робота з програмами читання з екрану. Хоча це не було основним фокусом проєкту, базові принципи доступності були враховані для забезпечення інклюзивності системи.

Завершальним етапом методології тестування стало приймальне тестування, яке проводилося за участі реальних представників цільових груп

користувачів — волонтерів, координаторів волонтерських організацій та потенційних заявників. Учасники тестування виконували типові задачі в системі та давали зворотний зв'язок щодо зручності використання, зрозумілості інтерфейсу та відповідності функціональності їхнім потребам. Такий підхід дозволив виявити неочевидні проблеми та зібрати цінні пропозиції щодо вдосконалення системи.

Весь процес тестування супроводжувався документуванням виявлених дефектів та відслідковуванням їх виправлення. Для кожного знайденого недоліку фіксувалися: опис проблеми, кроки для відтворення, очікуваний та фактичний результат, ступінь критичності та пропозиції щодо виправлення. Такий структурований підхід дозволив ефективно керувати процесом усунення дефектів та підвищувати якість системи.

Реалізована методологія тестування інформаційної системи для обробки заявок волонтерських організацій забезпечила високу якість кінцевого продукту. Багаторівневий підхід до тестування, який охоплює технічні аспекти, функціональність, безпеку та зручність використання, дозволив створити стабільну та надійну систему, що відповідає потребам користувачів та специфіці волонтерської діяльності в Україні.

4.2 Функціональне тестування

Функціональне тестування інформаційної системи для обробки заявок волонтерських організацій виступає ключовим етапом перевірки якості розробленого рішення, адже саме воно дає можливість підтвердити відповідність системи базовим вимогам користувачів. Для забезпечення якісного функціонального тестування було розроблено комплексний набір тест-кейсів, які охоплюють основні функціональні блоки системи та різні сценарії використання для кожної ролі користувачів – заявників, волонтерів та координаторів.

Для тестування функцій реєстрації та автентифікації користувачів було проведено серію тестів, які включали перевірку створення облікових записів з різними ролями, валідацію вхідних даних, обробку помилкових ситуацій та захист від несанкціонованого доступу. Окрема увага приділялася перевірці механізмів захисту паролів, зокрема їх хешуванню в базі даних та відновленню доступу. В результаті тестування було виявлено та усунуто проблеми з валідацією електронної пошти та механізмом перевірки унікальності облікових даних.

Тестування функціональності заявників включало перевірку процесів створення нових заявок, їх перегляду та відстеження статусів. Тест-кейси охоплювали різні комбінації вхідних даних: різні категорії допомоги, різні рівні терміновості та різні географічні локації. Особлива увага приділялася валідації даних на стороні сервера та коректності відображення статусів заявок. В процесі тестування було виявлено і виправлено неправильне сортування заявок за часом створення та покращено відображення статусів для забезпечення максимальної інформативності для заявників.

Для тестування функціональності волонтерів було розроблено сценарії, які охоплювали пошук та фільтрацію доступних заявок, їх взяття в роботу та позначення як виконаних. Тестувалися всі можливі комбінації фільтрів за категорією, терміновістю та локацією, а також поведінка системи при одночасному доступі кількох волонтерів до однієї заявки. В результаті було виявлено та виправлено проблему з конкурентним доступом, коли одна заявка могла бути взята в роботу кількома волонтерами через помилку в логіці контролера.

Тестування функціональності координаторів фокусувалося на процесах модерації заявок та моніторингу активності волонтерів. Перевірялася можливість перегляду заявок, що очікують на перевірку, їх підтвердження або відхилення, а також доступ до аналітичної інформації про роботу волонтерів. Під час тестування було виявлено проблему з підрахунком завершених заявок для волонтерів, яку було усунено корекцією SQL-запиту для отримання

статистики. Окрему увагу в процесі функціонального тестування було приділено взаємодії між різними ролями та переходам заявок між різними статусами. Для цього було розроблено комплексні сценарії, які моделювали повний життєвий цикл заявки від створення до завершення, включаючи всі проміжні етапи. Такий підхід дозволив виявити проблеми, пов'язані з некоректною зміною статусів та неправильним відображенням заявок для різних користувачів.

Тестування функцій фільтрації та пошуку заявок виявило важливість коректної роботи цих механізмів для зручності користувачів. Було проведено тести з різними комбінаціями фільтрів, включаючи граничні випадки, такі як порожні результати пошуку або дуже великі набори даних. В результаті було оптимізовано SQL-запити для підвищення продуктивності фільтрації та покращено користувацький інтерфейс для випадків, коли результати пошуку відсутні.

Вагомим аспектом функціонального тестування було перевірка механізмів валідації вхідних даних на всіх рівнях системи. Тестувалися різні сценарії введення некоректних даних, включаючи порожні значення, надто довгі тексти, спеціальні символи та потенційно небезпечні дані для запобігання ін'єкціям. Це дозволило виявити та виправити вразливості в обробці даних та підвищити загальну безпеку системи.

Тестування локалізації інтерфейсу підтвердило коректність відображення українських символів у всіх частинах системи, включаючи форми введення даних, повідомлення користувачам та відображення вмісту заявок. Особливу увагу було приділено перевірці коректності сортування та пошуку з урахуванням особливостей української мови. В результаті покращено коректність відображення та обробки українських текстів у системі. Для перевірки логіки управління статусами заявок було розроблено спеціальні тест-кейси, які моделювали різні сценарії зміни статусів та перевіряли коректність доступу до заявок різними користувачами залежно від поточного статусу. Це допомогло виявити та виправити логічні помилки в

контролерах, що відповідають за зміну статусів заявок та їх відображення для різних ролей.

Тестування механізмів сповіщень користувачів (flash-повідомлень) підтвердило їх ефективність для інформування про результати дій в системі. Перевірялася коректність відображення повідомлень при різних сценаріях: успішне виконання операцій, виникнення помилок, відмова в доступі тощо. В результаті було стандартизовано формат повідомлень та покращено їх інформативність для користувачів.

Функціональне тестування дозволило також виявити неочевидні проблеми з інтерфейсом користувача, такі як відсутність підтверджень для критичних дій, неінтуїтивне розміщення елементів керування та недостатню інформацію для прийняття рішень. Ці спостереження були враховані при доопрацюванні інтерфейсу для підвищення загальної зручності використання системи.

Результати функціонального тестування підтвердили загальну відповідність розробленої інформаційної системи для обробки заявок волонтерських організацій функціональним вимогам, визначеним на етапі аналізу та проєктування. Виявлені в процесі тестування проблеми були успішно усунуті, що дозволило підвищити якість та надійність системи. Подальше вдосконалення функціональності може відбуватися на основі зворотного зв'язку від реальних користувачів системи в процесі її експлуатації.

Загалом, результати функціонального тестування продемонстрували, що розроблена система ефективно реалізує всі основні бізнес-процеси обробки заявок у волонтерських організаціях, забезпечуючи зручну взаємодію між заявниками, волонтерами та координаторами. Це створює надійний фундамент для впровадження системи в реальну волонтерську діяльність та її подальшого розвитку відповідно до зростаючих потреб користувачів.

4.3 Тестування продуктивності

Тестування продуктивності інформаційної системи для обробки заявок волонтерських організацій є критично вагомим аспектом оцінки якості розробленого рішення. Метою цього виду тестування було визначення здатності системи функціонувати з прийнятною швидкістю при різних рівнях навантаження та в різних умовах експлуатації. Особливу актуальність тестування продуктивності набуває в контексті волонтерської діяльності, де в кризові періоди може спостерігатися різке зростання кількості заявок та користувачів.

Для забезпечення комплексної оцінки продуктивності системи були використані різні методи та інструменти тестування. Першим кроком стало проведення навантажувального тестування (load testing), яке моделювало реалістичні сценарії використання системи з різною кількістю одночасних користувачів. Для цього був використаний інструмент Apache JMeter, який дозволяє створювати віртуальних користувачів та симулювати їхню взаємодію з системою. Система тестувалася з поступовим збільшенням навантаження від 10 до 100 одночасних користувачів, що перевищує очікувану максимальну кількість користувачів для малих та середніх волонтерських організацій.

Результати навантажувального тестування показали, що система демонструє стабільну продуктивність при одночасній роботі до 50 користувачів, з середнім часом відгуку менше 1 секунди для основних операцій. При збільшенні кількості одночасних користувачів до 100 спостерігалося поступове зростання часу відгуку, однак система зберігала працездатність, із середнім часом відгуку до 3 секунд, що є прийнятним для більшості сценаріїв використання.

Вагомим аспектом тестування продуктивності стало дослідження ефективності роботи з базою даних, особливо для операцій, які потенційно можуть стати вузьким місцем системи. Було виявлено, що операції фільтрації заявок за кількома критеріями можуть викликати сповільнення при значному

обсязі даних. Для оптимізації цих операцій були створені додаткові індекси в базі даних SQLite та переглянуто логіку формування SQL-запитів, що дозволило зменшити час виконання складних пошукових запитів на 40-60%.

Окремим напрямком тестування стало стрес-тестування (stress testing), яке ставило за мету визначення граничних можливостей системи та її поведінки в екстремальних умовах. Для цього використовувалося симулювання аномально високого навантаження, що суттєво перевищувало розрахункові показники. Результати стрес-тестування показали, що при навантаженні понад 150 одночасних користувачів система значно сповільнюється, а при перевищенні 200 користувачів можливі відмови в обслуговуванні через вичерпання ресурсів сервера. Ці дані необхідні для планування можливого масштабування системи у майбутньому.

Особлива увага при тестуванні продуктивності була приділена аналізу споживання ресурсів сервера. Моніторинг використання процесора, пам'яті та дискового простору під час тестів дозволив виявити, що система є досить економною щодо ресурсів. При нормальному навантаженні використання процесора рідко перевищувало 30%, а споживання оперативної пам'яті становило близько 200-300 МБ, що робить можливим розгортання системи навіть на серверах з обмеженими ресурсами або на віртуальних хостингах.

Тестування швидкодії інтерфейсу користувача проводилося з використанням інструментів розробника в різних браузерах (Chrome DevTools, Firefox Developer Tools). Аналіз часу завантаження сторінок та швидкості виконання JavaScript-коду дозволив виявити можливості для оптимізації. Зокрема, було зменшено розмір CSS-файлів шляхом видалення невикористаних стилів та оптимізовано роботу з DOM для покращення відгуку інтерфейсу на дії користувача.

Окрему увагу було приділено оцінці продуктивності системи на мобільних пристроях, оскільки значна частина волонтерів використовує саме такі пристрої для доступу до інформаційних систем, особливо в польових умовах. Тестування на різних мобільних пристроях (смартфонах та

планшетах) з різною продуктивністю та різними версіями операційних систем показало, що система демонструє прийнятну швидкодію навіть на пристроях середнього та бюджетного сегментів.

Вагомим аспектом тестування продуктивності стала оцінка роботи системи при обмеженій пропускній здатності мережі, що особливо актуально для волонтерів, які можуть працювати в регіонах з неякісним інтернет-з'єднанням. Для моделювання таких умов використовувалися інструменти для обмеження швидкості з'єднання (Network Throttling). Тести показали, що система залишається функціональною навіть при низькій швидкості з'єднання (256 Кбіт/с), хоча час завантаження сторінок значно збільшується. Для покращення роботи в таких умовах було оптимізовано розмір сторінок та мінімізовано кількість запитів до сервера.

За результатами тестування продуктивності були розроблені рекомендації щодо оптимальних умов розгортання системи: мінімальні вимоги до сервера, рекомендована конфігурація бази даних, налаштування кешування та інші аспекти, які впливають на продуктивність. Ці рекомендації включені в документацію системи для адміністраторів, що значно полегшить процес впровадження та експлуатації системи в реальних умовах.

Тестування продуктивності інформаційної системи для обробки заявок волонтерських організацій підтвердило її здатність ефективно функціонувати в умовах, характерних для волонтерської діяльності в Україні. Система демонструє стабільну роботу при очікуваних рівнях навантаження, ефективно використовує ресурси сервера та забезпечує прийнятний час відгуку навіть при обмежених можливостях клієнтських пристроїв та мережного з'єднання. Виявлені під час тестування можливості для оптимізації були успішно реалізовані, що дозволило підвищити загальну продуктивність системи та забезпечити комфортну роботу користувачів з різними технічними можливостями.

4.4 Тестування прийнятності користувачами

Тестування прийнятності користувачами є завершальним та надзвичайно необхідним етапом перевірки якості інформаційної системи для обробки заявок волонтерських організацій. На відміну від технічно орієнтованих видів тестування, цей підхід фокусується на оцінці системи з точки зору її кінцевих користувачів, їхнього сприйняття та задоволеності. Особлива цінність такого тестування полягає в отриманні реального зворотного зв'язку від представників цільової аудиторії, що дозволяє виявити аспекти, які потребують доопрацювання для максимальної відповідності потребам волонтерської спільноти.

Для проведення тестування прийнятності було залучено 15 учасників, які представляли різні категорії користувачів системи: п'ятеро потенційних заявників (осіб, які потребують волонтерської допомоги), шестеро волонтерів з досвідом роботи у різних волонтерських організаціях та четверо координаторів волонтерських проєктів. Такий склад учасників забезпечив комплексну оцінку системи з різних перспектив та врахування специфічних потреб кожної групи користувачів.

Методологія тестування включала кілька етапів. Спочатку учасникам надавалася коротка інформація про систему та її призначення без детальних інструкцій щодо використання. Це дозволило оцінити інтуїтивність інтерфейсу та легкість освоєння системи новими користувачами. На другому етапі учасникам пропонувалося виконати набір типових задач, відповідних до їхньої ролі в системі. Для заявників це було створення нових заявок та перегляд їхнього статусу, для волонтерів – пошук, фільтрація та взяття заявок у роботу, для координаторів – модерація заявок та моніторинг активності волонтерів. На завершальному етапі проводилося обговорення досвіду використання системи та збір структурованого зворотного зв'язку через спеціально розроблені опитувальники.

Результати тестування показали в цілому позитивне сприйняття системи учасниками. Заявники відзначили простоту та зрозумілість процесу створення нових заявок, а також інформативність відображення їхнього поточного статусу. Особливо високу оцінку отримала можливість обирати категорію допомоги та рівень терміновості, що дозволяє чітко комунікувати свої потреби. Серед побажань заявників було відзначено доцільність додавання функції завантаження фотографій до заявок для кращої візуалізації потреб та можливість отримання сповіщень про зміну статусу заявки через електронну пошту.

Волонтери позитивно оцінили систему фільтрації заявок, яка дозволяє швидко знаходити запити, що відповідають їхнім можливостям та географічному розташуванню. Також було відзначено зручність інтерфейсу сторінки "Мої задачі", де відображаються взяті в роботу заявки. Серед пропозицій для покращення волонтери зазначили потребу в функціональності для прямого спілкування з заявниками через систему та можливість додавати коментарі до заявок для кращої координації між волонтерами.

Координатори найвище оцінили інструменти для моніторингу активності волонтерів та інтерфейс модерації заявок, які значно спрощують процес управління волонтерською діяльністю. Водночас, вони висловили потребу в більш розвинених аналітичних інструментах, зокрема, можливості бачити статистику за різними параметрами (категорії заявок, географічний розподіл, динаміка у часі) та функціональності для експорту даних для подальшого аналізу та звітності.

Загальним зауваженням від усіх категорій користувачів була відсутність мобільного додатку, який би забезпечував зручніший доступ до системи з мобільних пристроїв, хоча адаптивний веб-інтерфейс отримав позитивні відгуки за зручність використання на різних пристроях. Також учасники відзначили потребу в функціональності для обміну повідомленнями між користувачами системи для поліпшення координації та комунікації.

Для кількісної оцінки прийнятності системи використовувалася методика System Usability Scale (SUS), яка дозволяє виміряти суб'єктивне сприйняття зручності використання. Середній показник SUS для системи склав 78,5 за шкалою від 0 до 100, що відповідає оцінці "добре" та вказує на загальне позитивне сприйняття системи користувачами. При цьому найвищі оцінки система отримала від координаторів (83,2), що свідчить про особливу цінність системи для цієї категорії користувачів.

Окремо оцінювалася швидкість виконання типових завдань різними категоріями користувачів. Більшість учасників змогли виконати базові операції без додаткових інструкцій, що підтверджує інтуїтивність інтерфейсу. Середній час створення нової заявки заявниками склав близько 2 хвилин, що є прийнятним показником для подібних систем. Волонтери витрачали в середньому 1-2 хвилини на пошук та взяття в роботу відповідної заявки, а координатори здійснювали модерацію однієї заявки за 30-40 секунд, що свідчить про ефективність інтерфейсу для цих операцій.

Вагомим аспектом тестування прийнятності було виявлення потенційних бар'єрів для користувачів з різним рівнем технічної грамотності. Учасники з обмеженим досвідом використання веб-додатків відзначили, що система є досить зрозумілою, хоча деякі з них потребували додаткових пояснень щодо окремих функцій. На основі цих спостережень було вирішено розробити короткі відеоінструкції для нових користувачів, які будуть інтегровані в систему.

За результатами тестування прийнятності користувачами було сформовано перелік рекомендацій для вдосконалення системи у майбутніх версіях. Пріоритетними напрямками розвитку визначено: реалізацію системи сповіщень для користувачів, додавання функціональності для завантаження файлів до заявок, розширення аналітичних можливостей для координаторів та впровадження механізмів прямої комунікації між користувачами. Ці рекомендації були систематизовані та включені до плану розвитку системи.

Тестування прийнятності користувачами підтвердило, що розроблена інформаційна система для обробки заявок волонтерських організацій відповідає основним потребам та очікуванням цільової аудиторії. Система отримала позитивні відгуки щодо зручності використання, функціональності та відповідності контексту волонтерської діяльності в Україні. Виявлені в процесі тестування зауваження та пропозиції користувачів створюють основу для подальшого вдосконалення системи та розширення її функціональності відповідно до реальних потреб волонтерської спільноти [20].

ВИСНОВКИ

У результаті виконання бакалаврської роботи було розроблено інформаційну систему для обробки заявок волонтерських організацій, яка вирішує актуальну проблему автоматизації та оптимізації процесів надання волонтерської допомоги. Система реалізує повний життєвий цикл обробки заявок – від їх створення заявниками до виконання волонтерами під координацією представників волонтерських організацій.

Проведений аналіз предметної області дозволив чітко визначити специфіку волонтерської діяльності в Україні та виявити ключові вимоги до інформаційної системи. Порівняльний аналіз існуючих програмних рішень показав відсутність спеціалізованих систем, які б повністю відповідали потребам українських волонтерських організацій. Це підтвердило актуальність розробки власного рішення, адаптованого під локальний контекст та специфічні потреби цільової аудиторії.

У процесі проєктування системи було розроблено гнучку архітектуру на базі патерну MVC, яка забезпечує чіткий поділ відповідальності між різними компонентами та сприяє подальшому розвитку системи. Моделювання вимог дозволило формалізувати функціональні та нефункціональні аспекти системи та визначити ключові сценарії використання для різних категорій користувачів. Спроєктована структура бази даних забезпечує ефективне зберігання та обробку інформації про користувачів, заявки та їх статуси. Інтерфейс користувача розроблено з урахуванням принципів юзабіліті та адаптивного дизайну, що забезпечує зручне використання системи на різних пристроях.

Реалізація системи проведена з використанням сучасних технологій: Python з фреймворком Flask для серверної частини, SQLite для бази даних та HTML/CSS з Bootstrap для клієнтської частини. Такий технологічний стек забезпечує оптимальний баланс між функціональністю, простотою

впровадження та підтримки, що особливо головне для волонтерських організацій з обмеженими технічними ресурсами.

Всебічне тестування системи, включаючи функціональне тестування, оцінку продуктивності та тестування прийнятності користувачами, підтвердило її відповідність заявленим вимогам та придатність для використання в реальних умовах. Система демонструє стабільну роботу при очікуваних рівнях навантаження, забезпечує прийнятний час відгуку навіть при обмежених можливостях клієнтських пристроїв та мережного з'єднання. Тестування з залученням представників цільової аудиторії показало високий рівень прийнятності системи користувачами та її відповідність реальним потребам волонтерської спільноти.

Розроблена інформаційна система має ряд необхідних переваг порівняно з існуючими рішеннями:

– повна адаптація під специфіку волонтерської діяльності в Україні, включаючи українську мову інтерфейсу та врахування локального контексту;

– інтуїтивно зрозумілий інтерфейс, який не потребує спеціального навчання користувачів;

– диференційований доступ до функціональності для різних ролей користувачів;

– гнучка система категоризації та пріоритезації заявок;

– механізми моніторингу та аналітики для координаторів волонтерської діяльності;

– оптимізація під обмежені технічні ресурси, що є типовим для волонтерських організацій.

Практичне значення роботи полягає у створенні готового до впровадження програмного продукту, який може бути безпосередньо використаний волонтерськими організаціями для підвищення ефективності їхньої діяльності. Система має потенціал стати стандартним інструментом для координації волонтерської допомоги в українських реаліях, сприяючи більш

ефективному використанню обмежених ресурсів та підвищенню результативності волонтерської діяльності в цілому.

На основі результатів тестування та відгуків потенційних користувачів визначено перспективні напрямки подальшого розвитку системи, зокрема:

– реалізація системи сповіщень для інформування користувачів про зміни статусів заявок;

– додавання функціональності для завантаження файлів та фотографій до заявок;

– розширення аналітичних можливостей для координаторів;

– впровадження механізмів прямої комунікації між користувачами системи;

– розробка мобільного додатку для зручнішого доступу до системи з мобільних пристроїв.

Таким чином, розроблена інформаційна система для обробки заявок волонтерських організацій є практично значущим результатом, який має потенціал суттєво вплинути на ефективність волонтерської діяльності в Україні. Впровадження цієї системи дозволить оптимізувати процеси координації волонтерської допомоги, забезпечити прозорість на всіх етапах виконання заявок та сприятиме більш ефективному вирішенню нагальних проблем, з якими стикається українське суспільство в сучасних умовах.

Робота пройшла апробацію. За її результатами було опубліковано наступні тези доповідей Данилюк Б.М., Шахматов І.О. Розробка інформаційної системи для обробки заявок до волонтерських організацій. // Матеріали Всеукраїнської науково-технічної конференції “Застосування програмного забезпечення в ІКТ”. 24 квітня 2025 року, ДУІКТ, м. Київ. К:ДУІКТ, 2025 С. 79.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Дейнеко, Ж. В., Менделєва, М. В. Методика тестування інтерфейсів сайтів на основі функціонального та юзабіліті тестування // Поліграфічні, мультимедійні та web-технології: тези доповідей IX Міжнародної науково-технічної конференції, 14-18 травня 2024 р., м. Харків. – Т. 1. – Харків: ТОВ "Друкарня Мадрид", 2024. – С. 187-188.

2. Савельчук І. Б. Підготовка соціальних працівників в інноваційному освітньому середовищі університету : монографія. – Житомир : ЖДУ ім. І. Франка, 2019. – 236 с.

3. Perold H., Haas B., Goodrow T. Volunteering and the Digital World: Extending the Power of Volunteering through New Technologies (context paper). IAVE, October 2020. 25 p. URL: https://iave.org/iavewp/wp-content/uploads/2020/09/Volunteering-and-the-Digital-World-Extending-the-Power-of-Volunteering-through-New-Technologies.pdf (дата звернення: 23.04.2025).

4. Haski-Leventhal D., Alony I., Flemons P., Woods A. Online volunteering: Unlocking untapped potential. Research report (Volunteering Australia), November 2022. 15 p. URL: https://volunteeringstrategy.org.au/wp-content/uploads/2023/05/VRP_Online-volunteering-Unlocking-untapped-potential.pdf (дата звернення: 30.04.2025).

5. United Nations Volunteers (UNV). 2022 State of the World's Volunteerism Report: Building Equal and Inclusive Societies. Bonn : UNV, 2022. 124 p. URL: https://swvr2022.unv.org/wp-content/uploads/2021/11/UNV_SWVR_2022.pdf (дата звернення: 30.04.2025).

6. Urrea G., Yoo E. The role of volunteer experience on performance on online volunteering platforms. Production and Operations Management. 2023. Vol. 32, No. 2. P. 416–433. URL: https://ssrn.com/abstract=3784152 (дата звернення: 25.04.2025).

7. Dumanska I., Pavlova O., El Bouhissi H. Creating an Information System for Logistical Support of Volunteer Tasks: Basics and Functionality. // Proceedings of ITTAP'2023 – 3rd International Workshop on Information Technologies: Theoretical and Applied Problems (November 22–24, 2023, Ternopil, Ukraine). CEUR Workshop Proceedings. Vol. 3628. 2023. 11 p. URL: https://ceur-ws.org/Vol-3628/paper28.pdf (дата звернення: 15.04.2025).

8. Sha Y., Wei X., Niu C., Zhang Y., He L. Digital Volunteer Services in Emergency Situations: Typological Characteristics, Advantages, and Challenges. Data Science and Management. 2025. Vol. 8, No. 1. P. 1–10. URL: https://www.sciencedirect.com/science/article/pii/S2666764924000389?via%3Dihub(дата звернення: 27.04.2025).

9. Araque L. Y. Volunteer Management in Non-Profit Organizations: Experience of Huellas Foundation in Medellín, Colombia. Administrative Sciences. 2025. Vol. 15, No. 3. P. 77. URL: https://www.mdpi.com/2076-3387/15/3/77 (дата звернення: 22.04.2025).

10. Xu J., Jamil R., Mai X., Deng L., Zhang J., Yang Y. Volunteer Management in Nonprofit Organizations: A Bibliometric Analysis. SAGE Open. 2024. Vol. 14, Issue 4. 8 p. URL: https://journals.sagepub.com/doi/10.1177/21582440241284244 (дата звернення: 29.04.2025).

11. Naqshbandi K., Taylor S., Pillai A., Ahmadpour N. Designing for Helpers: Identifying new design opportunities for digital volunteerism. // In: Boess S., Cheung M., Cain R. (eds.). Proceedings of DRS 2020 International Conference – Synergy (11–14 Aug 2020, online). 2020. URL: https://dl.designresearchsociety.org/drs-conference-papers/drs2020/researchpapers/88/ (дата звернення: 07.05.2025).

12. Колотило М. О. Онлайн-волонтерство у сучасному світі: особливості та актуальні практики [Електронний ресурс] / Мар'яна Олексіївна Колотило. – 2020. – URL: https://ela.kpi.ua/bitstream/123456789/39756/1/S_r_i_s_X_2020-97-99.pdf (дата звернення: 02.05.2025).

13. Haski-Leventhal D., Alony I., Flemons P., Woods A. Online volunteering: Unlocking untapped potential [Electronic resource] / Debbie Haski-Leventhal, Irit Alony, Paul Flemons, Adam Woods. – 2023. – (Volunteering Research Papers). – URL: https://volunteeringstrategy.org.au/wp-content/uploads/2023/05/VRP_Online-volunteering-Unlocking-untapped-potential.pdf (дата звернення: 30.04.2025).

14. Silva F., Proença T., Ferreira M. R. Volunteers' perspective on online volunteering – a qualitative approach // International Review on Public and Nonprofit Marketing. – 2018. – Vol. 15, No. 4. – P. 531–552. – URL: https://doi.org/10.1007/s12208-018-0212-8 (дата звернення: 01.05.2025).

15. Lin W., Cheng J. Online volunteering and subjective well-being in China // Humanities and Social Sciences Communications. – 2024. – Vol. 11. – Article №1189. – URL: https://www.nature.com/articles/s41599-024-03676-0 (дата звернення: 30.04.2025).

16. Sha Y., Wei X., Niu C., Zhang Y., He L. Digital Volunteer Services in Emergency Situations: Typological Characteristics, Advantages, and Challenges // Data Science and Management. – 2025. – Vol. 8, No. 1. – P. 1–10. – URL: https://www.sciencedirect.com/science/article/pii/S2666764924000389?via%3Dihub (дата звернення: 29.04.2025).

17. Perold H., Haas B., Goodrow T. Volunteering and the Digital World: Extending the Power of Volunteering through New Technologies [Електронний ресурс] / Helene Perold, Benjamin Haas, Tony Goodrow. – 2020. – URL: https://iave.org/iavewp/wp-content/uploads/2020/09/Volunteering-and-the-Digital-World-Extending-the-Power-of-Volunteering-through-New-Technologies.pdf (дата звернення: 20.04.2025).

18. Hager M. A., Brudney J. L. Volunteer Management Capacity in America's Charities: Benchmarking a Pre-Pandemic Field and Assessing Future Directions [Електронний ресурс] / Mark A. Hager, Jeffrey L. Brudney. – 2021. – (Arizona State University). – URL: https://www.volunteeralive.org/docs/Hager_Brudney_VMC2_2021_brief.pdf (дата звернення: 13.04.2025).

19. Araque L. Y. Volunteer Management in Non-Profit Organizations: Experience of Huellas Foundation in Medellín, Colombia // Administrative Sciences. – 2025. – Vol. 15, No. 3. – P. 77. – URL: https://doi.org/10.3390/admsci15030077 (дата звернення: 17.04.2025).

20. Кіріленко О., Кузнецова Ю., Соколова Є., Фролова Г. Методи оцінювання usability інтерфейсу користувача // Вісник Нац. ун-ту «Львівська політехніка». Серія: Комп'ютерні науки та інформаційні технології. – 2013. – № 751. – С. 244–256. – URL: https://science.lpnu.ua/sites/default/files/journal-paper/2024/feb/33844/vis751komp-nauky-244-256.pdf (дата звернення: 15.04.2025).

ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Розробка інформаційної системи для обробки заявок до волонтерських організацій

Виконав студент 4 курсу
групи ПД-42
Данилюк Богдан Миколайович
Керівник роботи
Викладач кафедри ІПЗ Шахматов Іван Олександрович

Київ – 2025

МЕТА, ОБ’ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

Мета роботи: оптимізація та аналіз результатів волонтерської діяльності.

Об’єкт дослідження: процеси управління заявками у волонтерських організаціях.

Предмет дослідження: системи для обробки та управління заявками у волонтерських організаціях.

ЗАДАЧІ ДИПЛОМНОЇ РОБОТИ

1. Проаналізувати специфіку процесів обробки заявок у волонтерських організаціях, а також оцінити існуючі програмні рішення, що застосовуються у цій сфері.

2. Сформулювати функціональні та нефункціональні вимоги до інформаційної системи на основі потреб користувачів і специфіки волонтерської діяльності.

3. Розробити архітектуру клієнт-серверної системи, яка забезпечуватиме її гнучкість, масштабованість, розширюваність та безпеку.

4. Спроєктувати структуру бази даних для забезпечення ефективного зберігання, доступу та обробки інформації, пов'язаної з обробкою заявок.

5. Створити зручний та адаптивний інтерфейс користувача з урахуванням різних ролей (заявник, волонтер, адміністратор) та типів пристроїв.

6. Реалізувати повнофункціональну систему з використанням сучасних веб-технологій (Flask, SQLite, HTML/CSS/JS) та провести її тестування відповідно до заданих вимог.

АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ

Характеристика	VolunteerMatch	Salesforce Nonprofit	Better Impact	Volunteer Hub	Доброчин	Galaxy Digital	Angels Volunteer
Управління заявками	-	+	+	-	+	+	+
Категоризація запитів	+	+	+	+	+	+	+
Географічна локалізація	+	+	-	+	+	+	+
Терміновість запитів	-	+	-	-	+	-	+
Українська мова	-	-	-	-	+	-	+
Кризове управління	-	-	-	-	-	-	+
Аналітика	-	+	+	-	-	+	+
Мобільний доступ	+	+	+	+	-	+	+

ВИМОГИ ДО ЗАСТОСУНКУ

Функціональні вимоги

1. Застосунок має підтримувати створення облікових записів та вхід для трьох основних ролей: заявник, волонтер, координатор.
2. Заявник повинен мати змогу створювати нові заявки, переглядати та відстежувати їх статус.
3. Координатор повинен мати можливість переглядати нові заявки, змінювати їх статуси та призначати їх відповідним волонтерам.Вибір та виконання заявок волонтерами
4. Волонтери можуть переглядати список доступних заявок, фільтрувати їх за критеріями та брати у роботу.
5. Доступ до функціоналу має бути обмежений відповідно до ролі користувача.
6. Координатор повинен мати змогу переглядати статистику щодо виконаних заявок, активності волонтерів тощо.
7. Усі ролі повинні мати інструменти фільтрації та пошуку заявок за статусом, категорією, терміновістю, локацією тощо.

Нефункціональні вимоги

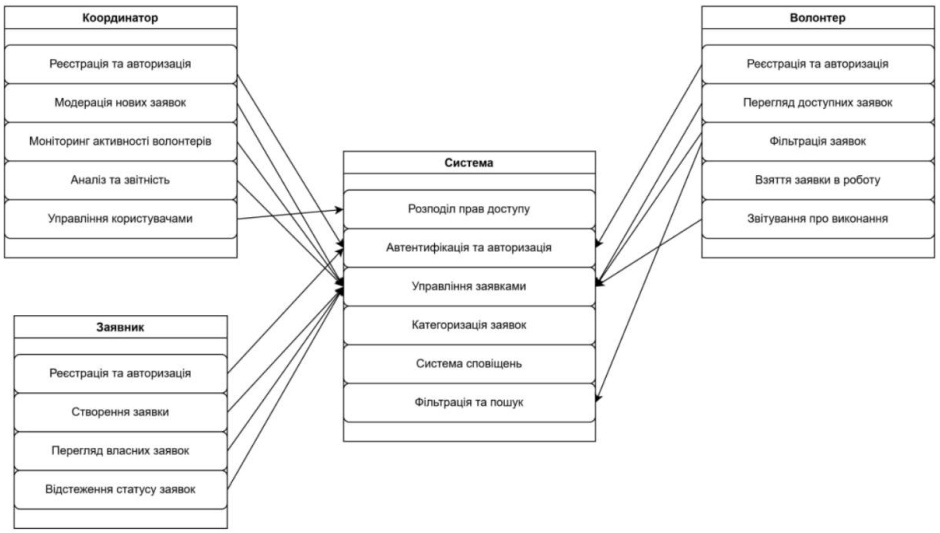
8. Інтерфейс має бути інтуїтивно зрозумілим, адаптованим під різні ролі та легко доступним з браузера.Застосунок має коректно обробляти помилки введення, збої при підключенні до БД тощо.
9. Зберігання паролів у хешованому вигляді, валідація форм.
10. Час відгуку інтерфейсу та основних дій (пошук, авторизація, зміна статусу заявки) не має перевищувати 1 секунди.

5

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

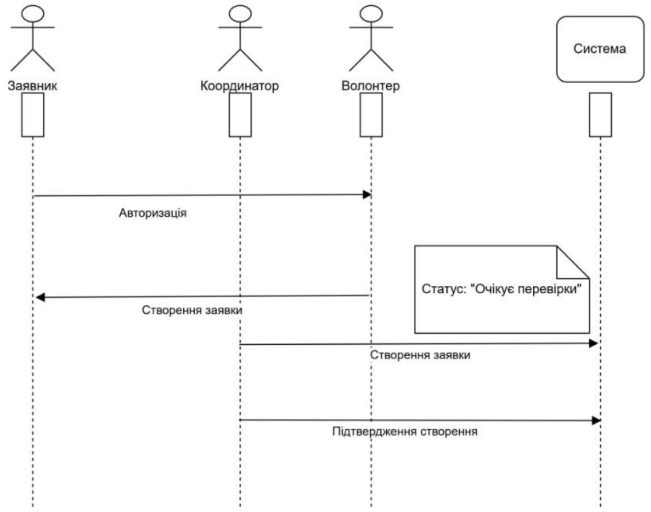
6

МОДЕЛЬ ВЗАЄМОДІЇ КОРИСТУВАЧІВ З ІНФОРМАЦІЙНОЮ СИСТЕМОЮ



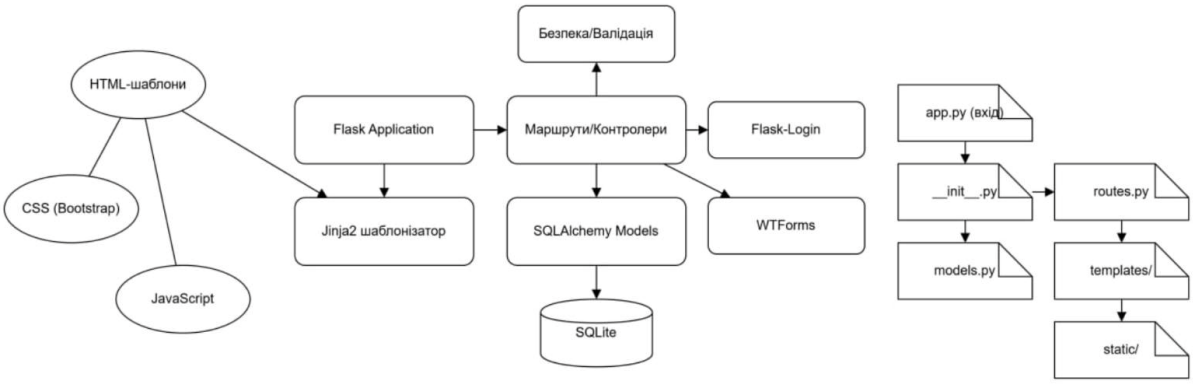
7

ДІАГРАМА ПОСЛІДОВНОСТІ ВЗАЄМОДІЇ КОРИСТУВАЧІВ ПІД ЧАС СТВОРЕННЯ ЗАЯВКИ



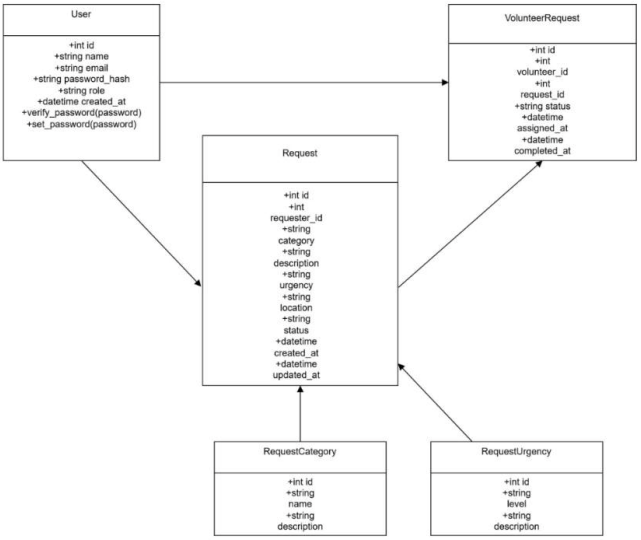
8

АРХІТЕКТУРА ВЕБ-ЗАСТОСУНКУ НА FLASK



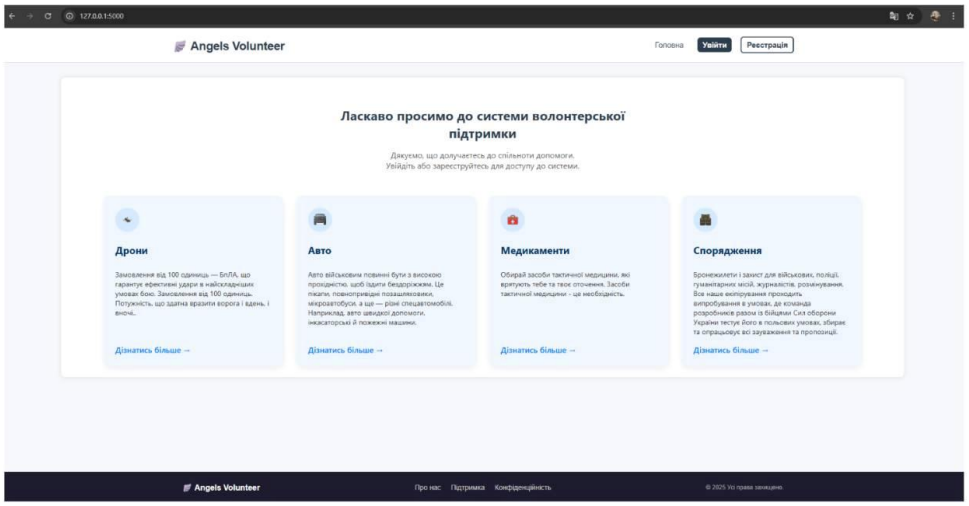
9

ДІАГРАМА КЛАСІВ СИСТЕМИ ОБРОБКИ ВОЛОНТЕРСЬКИХ ЗАЯВОК



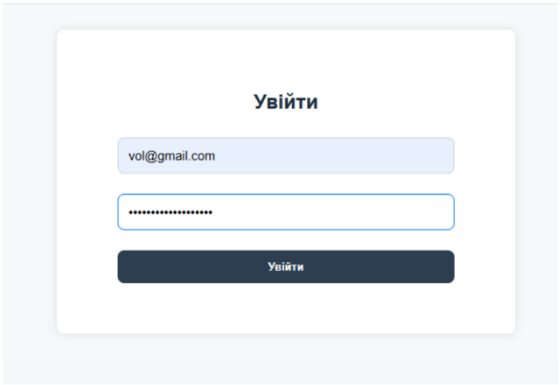
10

ЕКРАННІ ФОРМИ

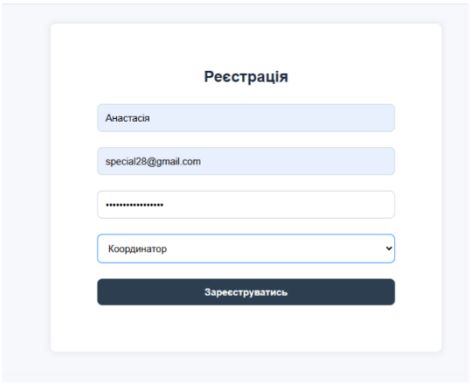


Головна сторінка системи для неавторизованих користувачів

ЕКРАННІ ФОРМИ

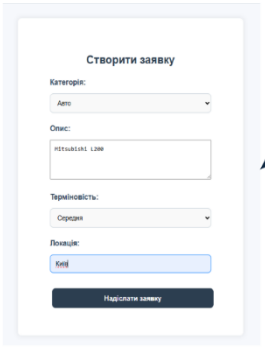


Форма входу в систему

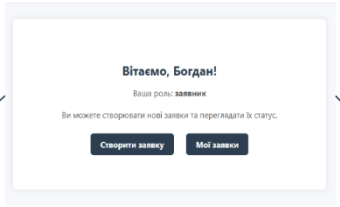


Форма реєстрації нового користувача

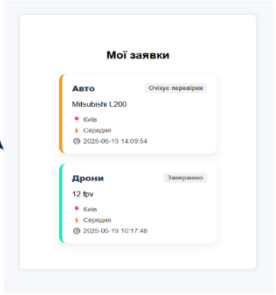
ЕКРАННІ ФОРМИ



Форма створення нової заявки



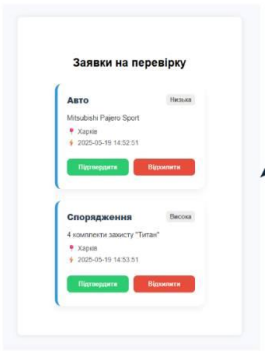
Особистий кабінет користувача з роллю "заявник"



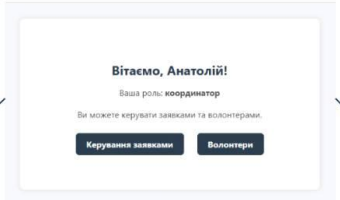
Сторінка перегляду заявок користувачем з роллю "заявник"

13

ЕКРАННІ ФОРМИ



Інтерфейс координатора для модерації заявок, що очікують перевірки

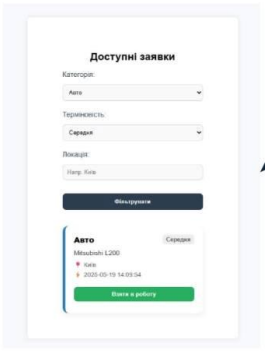


Особистий кабінет користувача з роллю "координатор"

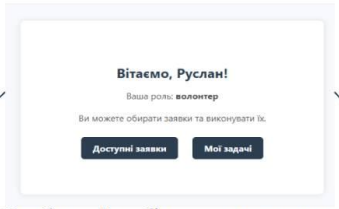
Аналітичний інтерфейс координатора для моніторингу активності волонтерів

14

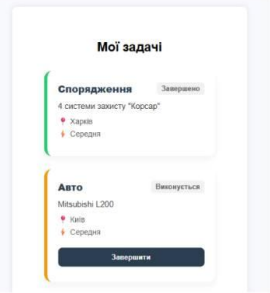
ЕКРАННІ ФОРМИ



Інтерфейс волонтера для пошуку та фільтрації доступних заявок



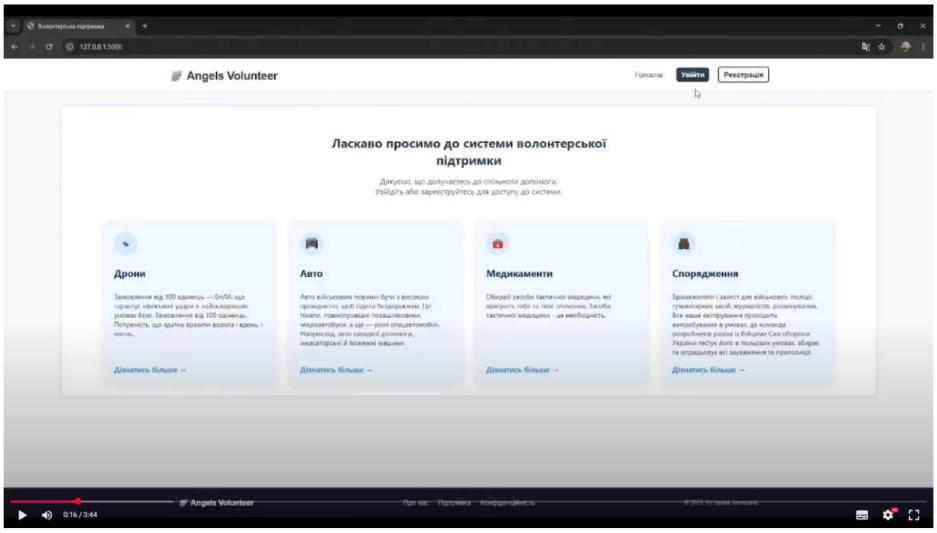
Особистий кабінет користувача з роллю "волонтер"



Сторінка перегляду задач користувачем з роллю "волонтер"

15

ДЕМОНСТРАЦІЯ ЗАСТОСУНКУ



16

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

1. Данилюк Б.М., Шахматов І.О. Розробка інформаційної системи для обробки заявок до волонтерських організацій. // Матеріали Всеукраїнської науково-технічної конференції “Застосування програмного забезпечення в ІКТ”. 24 квітня 2025 року, ДУІКТ, м. Київ .К:ДУІКТ, 2025 С. 79.

17

ВИСНОВКИ

1. Проведено глибокий аналіз процесів обробки заявок у волонтерських організаціях, що дозволило виявити типові труднощі, які виникають при ручній або неструктурованій обробці запитів.
2. Визначено функціональні та нефункціональні вимоги до інформаційної системи, що включають підтримку кількох ролей користувачів, безпеку, зручність у використанні та можливість масштабування.
3. Створено архітектуру системи на основі клієнт-серверного підходу, яка відповідає вимогам надійності, модульності та розширюваності.
4. Спроєктовано логічну структуру бази даних із дотриманням принципів нормалізації, що забезпечує ефективне управління даними та підтримку типових CRUD-операцій.
5. Створено адаптивний інтерфейс користувача з урахуванням потреб різних типів користувачів, що дозволяє легко подавати, переглядати та обробляти заявки.
6. Реалізовано веб-застосунок з використанням Flask та SQLite, інтегровано засоби захисту (в тому числі CSRF), авторизації та автентифікації.
7. Проведено функціональне тестування системи, яке підтвердило її працездатність, відповідність вимогам та зручність у реальному використанні.

18

ДОДАТОК Б. ЛІСТИНГИ ПРОГРАМНИХ МОДУЛІВ

```
from flask import Blueprint, render_template, redirect, url_for, request, flash
from flask_login import login_user, logout_user, login_required,
current_user
from werkzeug.security import generate_password_hash,
check_password_hash

from . import db
from .models import User
from flask import current_app as app

main = Blueprint('main', __name__)

@main.route('/')
def index():
    return render_template('index.html')

@main.route('/register', methods=['GET', 'POST'])
def register():
    if request.method == 'POST':
        name = request.form.get('name')
        email = request.form.get('email')
        password = request.form.get('password')
        role = request.form.get('role')

        existing_user = User.query.filter_by(email=email).first()
        if existing_user:
            flash('Користувач з таким email вже існує.')
```

```
            return redirect(url_for('main.register'))

        new_user = User(
            name=name,
            email=email,
            password=generate_password_hash(password,
method='pbkdf2:sha256'),
            role=role
        )
        db.session.add(new_user)
        db.session.commit()
        flash('Реєстрація успішна. Увійдіть до системи.')
        return redirect(url_for('main.login'))

    return render_template('register.html')

@main.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        email = request.form.get('email')
        password = request.form.get('password')

        user = User.query.filter_by(email=email).first()
        if not user or not check_password_hash(user.password, password):
            flash('Невірний email або пароль.')
            return redirect(url_for('main.login'))

        login_user(user)
        return redirect(url_for('main.dashboard'))
```

```
    return render_template('login.html')

@main.route('/dashboard')
@login_required
def dashboard():
    return render_template('dashboard.html')



from .models import Request

@main.route('/create_request', methods=['GET', 'POST'])
@login_required
def create_request():
    if current_user.role != 'заявник':
        flash('Лише заявники можуть створювати заявки.')
        return redirect(url_for('main.dashboard'))

    if request.method == 'POST':
        category = request.form.get('category')
        description = request.form.get('description')
        urgency = request.form.get('urgency')
        location = request.form.get('location')

        new_request = Request(
            user_id=current_user.id,
            category=category,
            description=description,
            urgency=urgency,
            location=location
        )
```

```
        db.session.add(new_request)
        db.session.commit()
        flash('Заявку створено!')
        return redirect(url_for('main.my_requests'))

    return render_template('create_request.html')



@main.route('/logout')
@login_required
def logout():
    logout_user()
    return redirect(url_for('main.index'))

@main.route('/my_requests')
@login_required
def my_requests():
    if current_user.role != 'заявник':
        flash('Лише заявники можуть переглядати заявки.')
        return redirect(url_for('main.dashboard'))

    requests =
Request.query.filter_by(user_id=current_user.id).order_by(Request.timesta
mp.desc()).all()
    return render_template('my_requests.html', requests=requests)

from .models import VolunteerRequest

@main.route('/available_requests', methods=['GET', 'POST'])
```

```
@login_required
def available_requests():
    if current_user.role != 'волонтер':
        flash('Доступ лише для волонтерів.')
        return redirect(url_for('main.dashboard'))

    taken_ids = [vr.request_id for vr in VolunteerRequest.query.all()]
    query = Request.query.filter(Request.status == 'Прийнято',
~Request.id.in_(taken_ids))

    # Застосування фільтрів
    category = request.args.get('category')
    urgency = request.args.get('urgency')
    location = request.args.get('location')

    if category:
        query = query.filter_by(category=category)
    if urgency:
        query = query.filter_by(urgency=urgency)
    if location:
        query = query.filter(Request.location.ilike(f"%{location}%"))

    requests_filtered = query.all()

    return render_template('available_requests.html',
requests=requests_filtered)



@main.route('/take_request/<int:request_id>', methods=['POST'])
```

```
@login_required
def take_request(request_id):
    if current_user.role != 'волонтер':
        flash('Лише волонтери можуть брати заявки.')
        return redirect(url_for('main.dashboard'))

    req = Request.query.get_or_404(request_id)
    req.status = 'Виконується'

    assignment = VolunteerRequest(
        volunteer_id=current_user.id,
        request_id=request_id
    )
    db.session.add(assignment)
    db.session.commit()

    flash('Заявка взята в роботу!')
    return redirect(url_for('main.available_requests'))



@main.route('/my_tasks')
@login_required
def my_tasks():
    if current_user.role != 'волонтер':
        flash('Доступ лише для волонтерів.')
        return redirect(url_for('main.dashboard'))

    assignments =
VolunteerRequest.query.filter_by(volunteer_id=current_user.id).all()
    requests = [Request.query.get(a.request_id) for a in assignments]
```

```
    return render_template('my_tasks.html', requests=requests)


@main.route('/manage_requests')
@login_required
def manage_requests():
    if current_user.role != 'координатор':
        flash('Доступ лише для координаторів.')
        return redirect(url_for('main.dashboard'))

    requests = Request.query.filter_by(status='Очікує перевірки').all()
    return render_template('manage_requests.html', requests=requests)


@main.route('/approve_request/<int:request_id>', methods=['POST'])
@login_required
def approve_request(request_id):
    if current_user.role != 'координатор':
        flash('Доступ лише для координаторів.')
        return redirect(url_for('main.dashboard'))

    req = Request.query.get_or_404(request_id)
    req.status = 'Прийнято'
    db.session.commit()
    flash('Заявку підтверджено.')
    return redirect(url_for('main.manage_requests'))

@main.route('/reject_request/<int:request_id>', methods=['POST'])
@login_required
```

```
def reject_request(request_id):
    if current_user.role != 'координатор':
        flash('Доступ лише для координаторів.')
        return redirect(url_for('main.dashboard'))

    req = Request.query.get_or_404(request_id)
    req.status = 'Відхилено'
    db.session.commit()
    flash('Заявку відхилено.')
    return redirect(url_for('main.manage_requests'))


@main.route('/complete_request/<int:request_id>', methods=['POST'])
@login_required
def complete_request(request_id):
    if current_user.role != 'волонтер':
        flash('Доступ лише для волонтерів.')
        return redirect(url_for('main.dashboard'))

    req = Request.query.get_or_404(request_id)
    assignment = VolunteerRequest.query.filter_by(
        request_id=request_id,
        volunteer_id=current_user.id
    ).first()

    if not assignment:
        flash('Ця заявка не закріплена за вами.')
        return redirect(url_for('main.my_tasks'))

    req.status = 'Завершено'
```

```
    db.session.commit()
    flash('Заявка позначена як завершена.')
    return redirect(url_for('main.my_tasks'))


@main.route('/volunteers')
@login_required
def volunteers():
    if current_user.role != 'координатор':
        flash('Доступ лише для координаторів.')
        return redirect(url_for('main.dashboard'))

    all_volunteers = User.query.filter_by(role='волонтер').all()
    stats = []

    for v in all_volunteers:
        total = VolunteerRequest.query.filter_by(volunteer_id=v.id).count()
        completed = (
            db.session.query(VolunteerRequest)
            .join(Request, VolunteerRequest.request_id == Request.id)
            .filter(
                VolunteerRequest.volunteer_id == v.id,
                Request.status == 'Завершено'
            )
            .count()
        )
        stats.append({
            'name': v.name,
            'email': v.email,
            'total': total,
```

```
        'completed': completed
    })

return render_template('volunteers.html', stats=stats
```