Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Інформаційна система обліку гуманітарної допомоги»

08-54.БКР.018.00.000 ПЗ

Виконав: студент 4 курсу, групи 1КІ-21б
спеціальності 123 – «Комп'ютерна інженерія»
освітня програма – «Комп'ютерна інженерія»
Шуляк М.В.

Керівник: к.т.н., доц. каф. ОТ
Городецька О. С.

Рецензент: д.т.н., проф. каф. ПЗ
Ліщинська Л.Б.

Допущено до захисту
Завідувач кафедри ОТ
д.т.н., проф. Азаров О. Д.

«13» 06 2025 р.

Вінниця ВНТУ — 2025 рік

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки
Освітньо—кваліфікаційний рівень бакалвр
Галузь знань — 12 Інформаційні технології
Спеціальність — 123 Комп'ютерна інженерія
Освітня програма — Комп'ютерна інженерія

ЗАТВЕРДЖУЮ
Завідувач кафедри ОТ
д.т.н.проф._____Азаров О. Д.
«21» березня 2025 року

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Шуляку Максиму Вадимовичу

1 Тема роботи: Інформаційна система обліку гуманітарної допомоги, керівник роботи Городецька Оксана Степанівна к.т.н., доцент кафедри ОТ, затверджено наказом вищого навчального закладу від «20» березня 2025 року №97.

2 Строк подання студентом роботи 13.05.2025

3 Вихідні дані до роботи: контент для наповнення бази даних інформаційної системи обліку гуманітарної допомоги, технології реалізації клієнтської, серверної та графічної частин, де графічна частина включає розробку користувацького вебінтерфейсу з адаптивним дизайном.

4 Зміст розрахунково-пояснювальної записки: вступ, аналіз предметної області, аналіз та обґрунтування вибору технологій, розробка інформаційної системи обліку гуманітарної допомоги, тестування інформаційної системи, висновки.

5 Перелік графічного матеріалу: функціональна схема системи у вигляді фізичної ER-моделі бази даних, діаграма розгортання серверної та клієнтської частин системи, структурна схема інформаційної системи. Перелік ілюстративного матеріалу: інтерфейс головної сторінки фонду, інтерфейс сторінки заявок

користувача, інтерфейс сторінки підтримки фонду, інтерфейс головної сторінки адмінпанелі.

6 Консультанти розділів роботи приведені в таблиці 1.

Таблиця 1 — Консультанти роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-3	к.т.н., доц. каф. ОТ Городецька О. С.	21.03.25	13.05.25
Нормоконтроль	ас.. каф. ОТ Швець С. І.	14.05.25	30.05.25

7 Дата видачі завдання 21.03.2025 р.

8 Календарний план приведений в таблиці 2.

Таблиця 2 — Календарний план

№ з/п	Назва етапів дипломної роботи	Строк виконання	Підпис
1	Постановка задачі роботи	21.03.25	
2	Пошук матеріалів та інформаційних джерел	21.03.25— 30.03.25	
3	Огляд та аналіз існуючих інформаційних систем	21.03.25— 30.03.25	
4	Обґрунтування вибору архітектури та мов програмування	31.03.25— 13.04.25	
5	Розробка й тестування інформаційної ситеми	14.04.25— 24.04.25	
6	Оформлення пояснювальної записки та ілюстративного матеріалу	25.04.25— 11.05.25	
7	Перевірка якості виконання бакалаврської кваліфікаційної роботи та усунення недоліків	13.05.25	

Студент Максим ШУЛЯК

Керівник роботи к.т.н., доц. Оксана ГОРОДЕЦЬКА

АНОТАЦІЯ

УДК 004.4

Шуляк М. В. Інформаційна система обліку гуманітарної допомоги. Бакалаврська дипломна робота зі спеціальності 123 — Комп’ютерна інженерія, освітня програма — Комп’ютерна інженерія. Вінниця: ВНТУ, 2025. 75 с. На укр. мові. Бібліогр.: 18 Рис.: 31; табл.: 2.

У БДР розроблено інформаційну систему для автоматизації процесів обліку гуманітарної допомоги. Реалізовано веб-інтерфейс з реєстрацією користувачів, поданням заявок, контролем залишків та формуванням звітів. Система базується на клієнт-серверній архітектурі та використовує документоорієнтовану базу даних.

Описано вибір технологій, структуру системи та результати тестування. Система готова до впровадження у волонтерських та благодійних організаціях для покращення обліку й координації допомоги.

Ключові слова: інформаційна система, гуманітарна допомога, база даних, клієнт-сервер, автоматизація обліку

ABSTRACT

Shuliak M. V. Information System for Humanitarian Aid Management. Bachelor's thesis in specialty 123 — Computer Engineering, educational program — Computer Engineering. Vinnytsia: VNTU, 2025. 75 p.

Ukrainian language. Bibliogr.: 18 titles, figs.: 31; tables: 2.

The bachelor's thesis presents the development of an information system for automating the processes of humanitarian aid accounting. A web interface has been implemented, providing user registration, application submission, stock control, and report generation. The system is based on a client-server architecture and document-oriented database.

The work describes the choice of technologies, the system structure, and the results of testing. The system is ready for implementation in volunteer and charitable organizations to improve aid tracking and coordination.

Keywords: information system, humanitarian aid, database, client-server, accounting automation

ЗМІСТ

ВСТУП 4

1 **АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ** 6

1.1 Характеристика предметної галузі 6

1.1.1 Поняття та класифікація інформаційних систем 6

1.1.2 Компонентний склад інформаційної системи 7

1.2 Основні вимоги до інформаційної системи обліку гуманітарної допомоги .. 10

1.3 Огляд та аналіз існуючих рішень 13

1.4 Постановка задачі 19

2 **АНАЛІЗ ТА ОБҐРУНТУВАННЯ ВИБОРУ ТЕХНОЛОГІЙ** 21

2.1 Обґрунтування вибору архітектури інформаційної системи 21

2.2 Вибір мови програмування 23

2.3 Вибір інструментів для розробки веб-інтерфейсу 25

2.4 Вибір засобів для реалізації бази даних 26

3 РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ ОБЛІКУ ГУМАНІТАРНОЇ ДОПОМОГИ 29

3.1 Проєктування архітектури та розробка загальної структури інформаційної системи 29

3.2 Розробка клієнтської та серверної частини додатку 34

3.3 Розробка бази даних і системи зберігання та захисту даних 48

4 **ТЕСТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ** 53

4.1 Тестування серверної частини 53

4.2 Тестування клієнтської частини 56

ВИСНОВКИ 59

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 60

ДОДАТОК А Технічне завдання 62

ДОДАТОК Б Протокол перевірки кваліфікаційної роботи 67

ДОДАТОК В Графічна частина 68

ДОДАТОК Г Ілюстративна частина 71

ДОДАТОК Д Лістинги програмного забезпечення 74

ВСТУП

У світі з кожним роком зростає кількість надзвичайних ситуацій, пов’язаних із воєнними конфліктами, природними катастрофами та економічними кризами. У таких умовах гуманітарна допомога відіграє ключову роль у забезпеченні базових потреб постраждалих. Проте ефективне управління запитами на допомогу, їх обліком і розподілом залишається складним завданням, особливо за великого обсягу даних і обмежених ресурсів.

Багато організацій досі використовують ручні методи ведення обліку — таблиці, блокноти або неструктуровані цифрові документи. Це призводить до втрати даних, дублювання записів та помилок у розподілі ресурсів. Відсутність централізованої інформаційної системи також ускладнює прозоре звітування й контроль.

Система повинна бути доступною, простою в користуванні й достатньо гнучкою для адаптації під різні організаційні потреби. Її реалізація передбачає використання сучасних веб-технологій та документоорієнтованої бази даних для надійного збереження інформації.

Об’єктом дослідження даної бакалаврської роботи є процес проектування інформаційної системи обліку гуманітарної допомоги.

Предметом дослідження є програмне забезпечення, що реалізує інформаційну систему.

Метою роботи є розробка інформаційної системи обліку гуманітарної допомоги з розширеними функціональними можливостями, а саме автоматизацією процесу реєстрації користувачів, подання заявок та ведення обліку наданих ресурсів.

Для реалізації поставленої мети в роботі передбачено виконання таких основних завдань:

– провести аналіз предметної області, зокрема вивчити специфіку обліку гуманітарної допомоги, типові труднощі, які виникають у процесі обробки запитів

і розподілу ресурсів, а також дослідити існуючі інформаційні системи подібного призначення з метою виявлення їх переваг і недоліків.

– обґрунтувати вибір сучасних інформаційних технологій для розробки інформаційної системи, включаючи мову програмування, систему управління базами даних, засоби створення користувацького інтерфейсу та допоміжні інструменти розробки, з урахуванням вимог до доступності, безпеки, масштабованості та простоти використання.

– сформувати логічну та фізичну структуру бази даних, а також розробити загальну архітектуру системи на основі клієнт-серверної моделі.

– реалізувати інформаційну системи у вигляді веб-додатку з базовим функціоналом: реєстрація та авторизація користувачів, подання заявок на отримання допомоги, перегляд їх статусу та адміністрування даних.

– провести тестування створеної системи, перевіривши працездатність ключових модулів для забезпечення її надійності та стабільності в реальних умовах використання.

Апробація результатів бакалаврської кваліфікаційної роботи здійснена у формі доповіді на LIV Всеукраїнській науково-технічній конференції факультету інформаційних технологій та комп’ютерної інженерії**,** секція «Обчислювальна техніка» (ВНТУ, 2025 р.) [1]:

Інформаційна система обліку гуманітарної допомоги / М. В. Шуляк, О. С. Городецька // Матеріали LIV Всеукраїнської науково-технічної конференції факультету інформаційних технологій та комп’ютерної інженерії. – Вінниця, 2025. – 3 с. [Електронний ресурс]. https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2025/author/downloadFile/24037/84641/1

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Характеристика предметної галузі

1.1.1 Поняття та класифікація інформаційних систем

Інформаційна система (ІС) — це цілісна сукупність взаємопов’язаних компонентів, яка призначена для збирання, зберігання, обробки, аналізу, передавання та представлення інформації. Основним завданням таких систем є підтримка прийняття рішень, організація ефективного управління процесами або автоматизація діяльності в певній галузі. Сучасні інформаційні системи охоплюють як апаратні засоби (сервери, клієнтські пристрої, мережеве обладнання), так і програмні інструменти (бази даних, веб-інтерфейси, серверна логіка), а також користувачів, які безпосередньо взаємодіють із системою.

У широкому розумінні ІС виконує роль «інформаційного посередника» між джерелами даних і кінцевими користувачами. Вона дозволяє перетворювати розрізнену, неструктуровану або надто об’ємну інформацію в зручну для аналізу та використання форму. Це досягається завдяки поєднанню певних моделей обробки даних, алгоритмів логіки роботи системи, а також інтерфейсних компонентів, через які користувачі можуть отримувати доступ до функціоналу.

Інформаційні системи бувають різних типів — від простих локальних рішень для ведення електронних записів до складних розподілених систем, які обслуговують тисячі користувачів одночасно. Їхнє застосування охоплює фінансову аналітику, управління підприємствами (ERP), електронну комерцію, медичну документацію, освітні платформи, логістику, державні реєстри, інтелектуальні транспортні системи тощо. У всіх цих випадках спільною ознакою є потреба у швидкому, надійному та структурованому опрацюванні великих обсягів інформації.

Сучасна тенденція розвитку ІС спрямована на інтеграцію з хмарними технологіями, мобільними пристроями, аналітикою великих даних і штучним інтелектом. Це дозволяє підвищити адаптивність систем до змін у зовнішньому середовищі, зменшити витрати на інфраструктуру, забезпечити масштабованість і

розширити доступ до даних у реальному часі. Крім того, з огляду на посилення вимог до конфіденційності та безпеки, дедалі більше уваги приділяється побудові ІС із високим рівнем захисту персональних і чутливих даних [2].

У реальному житті інформаційні системи застосовуються для вирішення широкого спектра прикладних завдань, які неможливо або вкрай складно виконувати вручну. Наприклад, у сфері фінансів ІС допомагають обробляти тисячі транзакцій за лічені секунди, автоматизувати бухгалтерський облік і забезпечувати аудит. У логістиці вони використовуються для управління складськими запасами, планування маршрутів та моніторингу транспорту в реальному часі. В охороні здоров’я — для збереження медичних карток пацієнтів, ведення графіків приймання, призначення ліків та взаємодії між лікарями та пацієнтами.

Ключовою перевагою інформаційних систем у таких випадках є те, що вони дозволяють працювати з великими обсягами даних швидко, точно та без втрати інформації. Крім того, ІС значно знижують людський фактор: автоматизовані перевірки, алгоритми валідації даних і системи контролю помилок дозволяють уникати багатьох критичних помилок, які можуть призвести до небажаних наслідків.

Завдяки гнучкості та адаптивності, сучасні інформаційні системи можуть бути адаптовані під конкретні задачі, навіть у дуже вузьких галузях. У тих випадках, коли йдеться про повторювані процеси, які залежать від точності даних, їх структури та своєчасної обробки, застосування інформаційних систем є не просто бажаним, а обов’язковим кроком. Саме тому ІС стали базовим інструментом для цифрової трансформації.

1.1.2 Компонентний склад інформаційної системи

Інформаційна система являє собою сукупність взаємопов’язаних елементів, які функціонують спільно з метою підтримки всіх етапів обробки даних: від збирання інформації до її подальшого зберігання, аналізу та передавання. Ефективність функціонування будь-якої інформаційної системи безпосередньо залежить від її складових, які можна класифікувати на дві основні категорії: апаратне

забезпечення та програмне забезпечення [3]. Складові інформаційної системи показані на рисунку 1.1.

Рисунок 1.1 — Складові інформаційної системи

Апаратне забезпечення є технічною основою інформаційної системи, що включає всі фізичні пристрої, необхідні для забезпечення функціонування програмних рішень. Сюди належать комп’ютери, ноутбуки, сервери, периферійні пристрої, мережеве обладнання, термінали введення даних та пристрої для сканування чи ідентифікації. В інформаційній системі обліку гуманітарної допомоги апаратне забезпечення забезпечує роботу клієнтських інтерфейсів, передачу запитів мережею, а також роботу серверної частини, що відповідає за обробку та зберігання інформації. У разі використання мобільних або віддалених пристроїв особливу увагу приділено стабільності мережевого з’єднання та безперервному доступу до сервісів системи.

Програмне забезпечення забезпечує реалізацію логіки функціонування системи, керування пристроями та доступ користувача до необхідного функціоналу. Його доцільно поділяти на системне (операційні системи, драйвери, засоби віртуалізації) та прикладне (вебінтерфейси, адміністративні модулі, бази даних, аналітичні панелі). У межах цього проєкту розроблено клієнтську частину вебзастосунку, серверну логіку з REST API та базу даних MongoDB. Усі

компоненти працюють у єдиному середовищі та забезпечують обробку заявок, автентифікацію користувачів, формування історії взаємодій та збереження звітності. Програмна частина системи є масштабованою і може бути доповнена новими модулями у разі зміни вимог або розширення функціоналу.

Інформаційні ресурси формують інформаційне наповнення системи та включають дані, що надходять у процесі експлуатації: облікові записи користувачів, текстові та графічні запити, категорії допомоги, статуси, а також службові журнали. Структура бази даних побудована з урахуванням забезпечення зв’язків між сутностями, контролю цілісності та швидкого доступу до потрібної інформації. Зібрані дані використовуються для моніторингу поточних запитів, аналізу ефективності обробки заявок, а також для подальшого планування та звітності перед донорами чи партнерами.

Ключову роль у функціонуванні системи відіграє персонал. Це користувачі, які взаємодіють із платформою, модератори, які перевіряють і публікують запити, а також адміністратори, які відповідають за загальну структуру, наповнення і стабільність роботи системи. Для кожної ролі визначено набір функцій та рівень доступу, реалізований на рівні програмного забезпечення. Такий підхід дозволяє гнучко управляти правами доступу, забезпечити захист від несанкціонованих дій та підвищити рівень безпеки платформи загалом.

Організаційно-методичне забезпечення охоплює правила, інструкції, регламенти та інші документи, які регламентують порядок роботи з інформаційною системою. Воно забезпечує єдиний підхід до введення, зберігання та опрацювання даних, а також визначає послідовність дій при поданні заявок, реагуванні на запити користувачів, перевірці автентичності та формуванні звітної інформації. У разі масштабування або впровадження нових функцій саме методичне забезпечення відіграє провідну роль у стандартизації процесів.

Отже, кожна із вказаних складових інформаційної системи виконує специфічну функцію та є критично важливою для досягнення цілей системи. Їхня інтеграція у єдину логічну структуру дозволяє реалізувати комплексне, ефективне

та масштабоване рішення, здатне забезпечити потреби благодійної організації у сфері обліку гуманітарної допомоги.

Рисунок 1.2 — Компонентна структура інформаційної системи

1.2 Основні вимоги до інформаційної системи обліку гуманітарної допомоги

Однією з галузей, де потреба в ефективній обробці та контролі даних є особливо гострою, є облік гуманітарної допомоги. У цій сфері щоденно відбувається велика кількість операцій, пов’язаних із реєстрацією запитів, обліком отриманих і виданих ресурсів, комунікацією між заявниками та надавачами допомоги, формуванням звітності та аналізом потреб. Часто ці процеси ведуться вручну або за допомогою примітивних засобів — електронних таблиць,

месенджерів, блокнотів, що не тільки збільшує навантаження на персонал, а й створює ризики втрати або спотворення інформації [4].

За відсутності централізованої інформаційної системи можуть виникати дублювання записів, плутанина у черговості надання допомоги, незручність у фільтрації заявок за пріоритетом, а також складності у формуванні звітності та прозорому аналізі діяльності організації. Це особливо критично, коли мова йде про великі обсяги запитів або коли допомога повинна надаватися в максимально стислі терміни. У таких умовах навіть незначні затримки чи помилки можуть мати серйозні наслідки для людей, які перебувають у скрутному становищі.

Варто також врахувати, що в умовах кризових ситуацій — таких як війна, стихійне лихо чи масова евакуація населення — кількість заявок та інтенсивність обробки інформації зростає в рази. Ручне опрацювання заявок у таких випадках практично втрачає ефективність. Саме тому впровадження інформаційної системи, яка дозволить централізовано й автоматизовано обробляти ці процеси, є надзвичайно актуальним і необхідним кроком.

Щоб така система могла ефективно працювати та відповідати потребам користувачів, вона повинна відповідати ряду функціональних вимог. У першу чергу, система має забезпечувати реєстрацію та автентифікацію користувачів з різними рівнями доступу: звичайні користувачі (отримувачі допомоги), оператори (волонтери), адміністратори (менеджери фонду або організації). Такий розподіл ролей дозволяє не лише регламентувати доступ до окремих функцій, а й підвищити безпеку системи в цілому.

Наступною ключовою вимогою є можливість подачі та перегляду заявок. Користувач повинен мати змогу легко подати запит на допомогу, вказавши тип потреби (їжа, одяг, ліки тощо), кількість, регіон або категорію. Оператори, своєю чергою, мають отримати зручний інтерфейс для опрацювання цих заявок — схвалення, редагування, позначення як виконаних. Це дозволяє контролювати етапи обробки в режимі реального часу.

Також система повинна підтримувати облік наданої допомоги, тобто зберігати інформацію про те, що було видано, коли, кому і в якій кількості. Це

важливо як для внутрішнього аналізу ефективності, так і для можливості подальшого звітування перед донорами або аудиторами. Такий облік має вестися у формі бази даних із можливістю сортування, пошуку та фільтрації записів за параметрами.

Ще однією важливою вимогою є можливість формування звітів. Це можуть бути як внутрішні аналітичні звіти для прийняття рішень, так і формалізовані документи для звітності перед державними органами чи міжнародними донорами. Система має передбачати автоматичне або напівавтоматичне створення таких звітів із можливістю експорту в стандартні формати — PDF, Excel тощо.

Окрім основного функціоналу, важливу роль відіграють і нефункціональні вимоги, які не завжди помітні на перший погляд, але значною мірою впливають на зручність користування, ефективність і стійкість системи. Однією з таких вимог є доступність. Інформаційна система повинна бути доступною для користувачів у будь-який час, з різних пристроїв і незалежно від їхнього місцезнаходження. Це означає, що веб-застосунок має бути оптимізований як для комп’ютерів, так і для смартфонів, а інтерфейс — працювати без затримок навіть при низькій швидкості інтернету.

Ще одним важливим аспектом є масштабованість. Навіть якщо спочатку система використовується невеликою організацією або локальним волонтерським штабом, з часом кількість користувачів може зрости у декілька разів. Тому система повинна бути розроблена так, щоб її можна було розширити — додати нові модулі, збільшити обсяг бази даних, забезпечити більшу кількість одночасних підключень без втрати продуктивності.

Особливу увагу слід приділити питанню інформаційної безпеки. У системі зберігаються персональні дані користувачів, а також інформація, яка стосується соціально вразливих категорій населення. Тому потрібно забезпечити захист облікових записів через шифрування паролів, використання захищених протоколів передавання даних (наприклад, HTTPS), а також запобігання несанкціонованому доступу до адміністративної частини сайту. Варто також реалізувати систему резервного копіювання, щоб уникнути втрати даних у разі збою.

Не менш важливою є зручність користувацького інтерфейсу. Система має бути інтуїтивно зрозумілою навіть для людей без спеціальних знань у галузі ІТ. Всі форми, кнопки, повідомлення та повідомлення про помилки повинні бути чіткими, зрозумілими та адаптованими під цільову аудиторію. Зайва складність або перевантаженість інтерфейсу лише знижує ефективність роботи користувачів і може призвести до помилок або відмови від використання системи.

1.3 Огляд та аналіз існуючих рішень

Для того щоб розробити ефективну інформаційну систему обліку гуманітарної допомоги, важливо проаналізувати вже існуючі рішення, які частково або повністю реалізують подібний функціонал. Це дозволяє не лише вивчити наявний досвід, але й уникнути повторення помилок, скористатися перевіреними підходами та виявити недоліки, які потребують покращення.

У сучасних умовах в Україні паралельно функціонують як волонтерські ініціативи з цифровими інструментами, так і повноцінні державні інформаційні системи. Їх функціональність, рівень автоматизації та технічна реалізація суттєво відрізняються. Одні орієнтовані на зручність і швидку комунікацію з аудиторією, інші — на масштабованість і регламентовану звітність.

У цьому підрозділі буде проведено аналіз трьох прикладів:

– волонтерського фонду Сергія Стерненка,

– фонду «Повернись живим»

– та державної автоматизованої системи реєстрації гуманітарної допомоги (АС ГД).

Таке порівняння дозволяє охопити повний спектр підходів — від ручного або напівавтоматизованого до повністю централізованого цифрового управління.

Одним із прикладів сучасного волонтерського проєкту, який частково реалізує цифровий облік гуманітарної допомоги, є Благодійний Фонд Спільноти Стерненка. Цей фонд активно займається закупівлею та передачею допомоги для українських військових, зокрема дронів, спорядження, техніки й іншого обладнання. Його діяльність зосереджена переважно на організації швидкої

логістики, прозорості у використанні коштів та залученні підтримки від громадськості.

На сайті реалізовано базову функціональність для співпраці з виробниками — зокрема через окрему сторінку “Виробникам”, де доступна онлайн-форма для подання пропозицій. Сторінку «Виробникам» показано на рисунку 1.3.

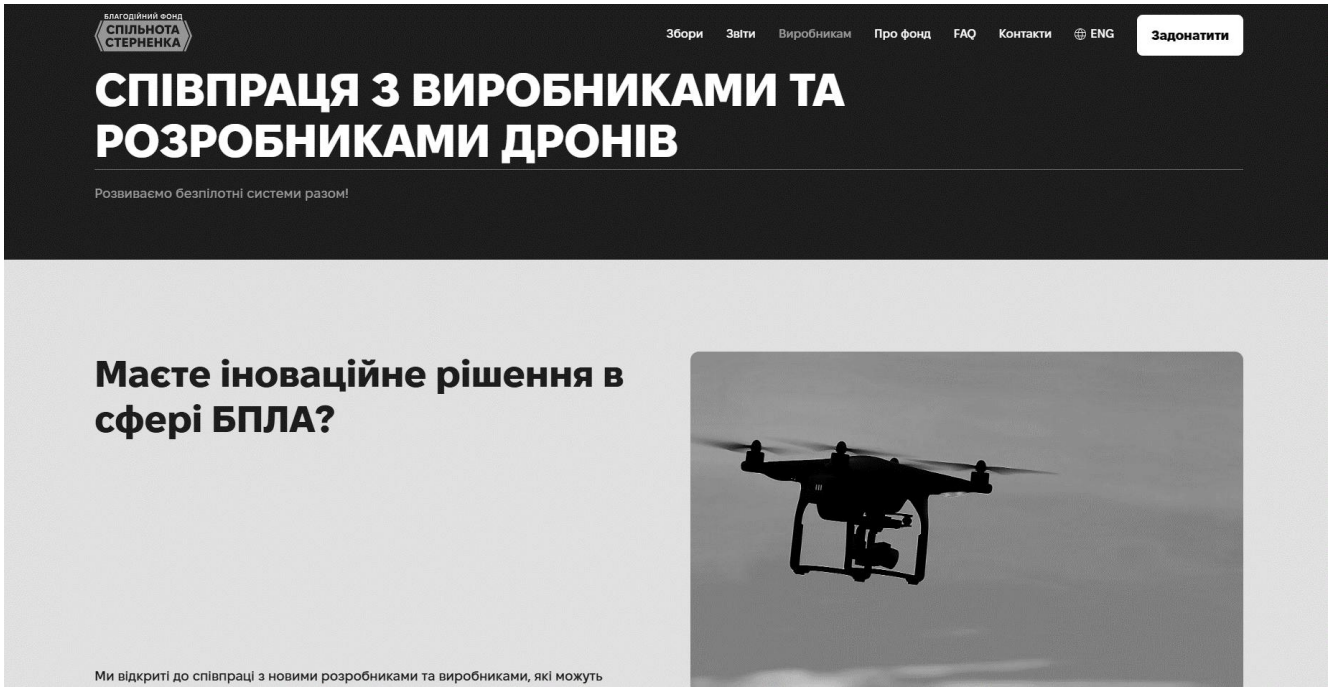


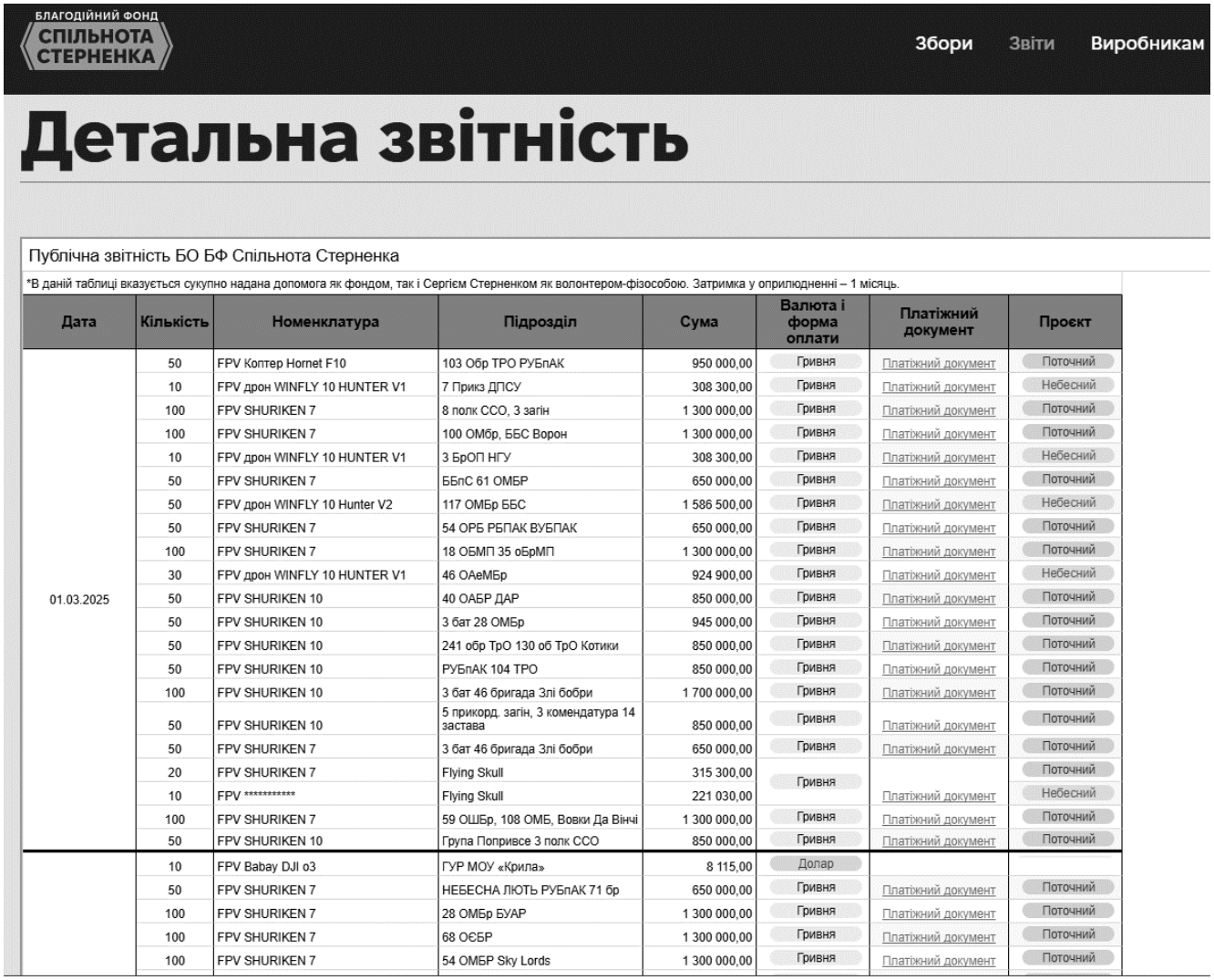
Рисунок 1.3 — Сторінка «Виробникам», реалізована БФ Спільнота Стерненка

Це дозволяє налагодити комунікацію між фондом і потенційними постачальниками дронів або комплектуючих. Усі звіти про закупівлі, передані засоби та витрачені кошти регулярно публікуються у відкритому доступі (див. рис. 1.4), що значно підвищує довіру до організації.

Водночас із технічної точки зору така система не є повноцінною інформаційною системою. На сайті відсутні особисті кабінети, централізована база даних заявок або інтеграція з обліком наданої допомоги. Уся обробка інформації, імовірно, здійснюється вручну або через сторонні сервіси (наприклад, Google-форми, таблиці, месенджери). Це обмежує можливості для масштабування та підвищує ризик втрати або дублювання інформації.

Проте фонд Стерненка можна вважати прикладом ефективного використання цифрових засобів зв’язку й прозорої звітності без впровадження складної ІС. Його

діяльність підкреслює, що навіть мінімальний рівень цифровізації вже дає змогу досягати вагомих результатів у волонтерській сфері — особливо за рахунок чіткої організації, відкритості та довіри до бренду.


БЛАГОДІЙНИЙ ФОНД СПІЛЬНОТА СТЕРНЕНКА

Збори Звіти Виробникам

Детальна звітність

Публічна звітність БО БФ Спільнота Стерненка

*В даній таблиці вказується сукупно надана допомога як фондом, так і Сергієм Стерненком як волонтером-фізособою. Затримка у оприлюдненні – 1 місяць.

Дата	Кількість	Номенклатура	Підрозділ	Сума	Валюта і форма оплати	Платіжний документ	Проєкт
01.03.2025	50	FPV Коптер Hornet F10	103 Обр ТРО РУБпАК	950 000,00	Гривня	Платіжний документ	Поточний
	10	FPV дрон WINFLY 10 HUNTER V1	7 Прикз ДПСУ	308 300,00	Гривня	Платіжний документ	Небесний
	100	FPV SHURIKEN 7	8 полк ССО, 3 загін	1 300 000,00	Гривня	Платіжний документ	Поточний
	100	FPV SHURIKEN 7	100 ОМБр, ББС Ворон	1 300 000,00	Гривня	Платіжний документ	Поточний
	10	FPV дрон WINFLY 10 HUNTER V1	3 БрОП НГУ	308 300,00	Гривня	Платіжний документ	Небесний
	50	FPV SHURIKEN 7	ББпС 61 ОМБР	650 000,00	Гривня	Платіжний документ	Поточний
	50	FPV дрон WINFLY 10 Hunter V2	117 ОМБр ББС	1 586 500,00	Гривня	Платіжний документ	Небесний
	50	FPV SHURIKEN 7	54 ОРБ РБПАК ВУБПАК	650 000,00	Гривня	Платіжний документ	Поточний
	100	FPV SHURIKEN 7	18 ОБМП 35 оБрМП	1 300 000,00	Гривня	Платіжний документ	Поточний
	30	FPV дрон WINFLY 10 HUNTER V1	46 ОАеМБр	924 900,00	Гривня	Платіжний документ	Небесний
	50	FPV SHURIKEN 10	40 ОАБР ДАР	850 000,00	Гривня	Платіжний документ	Поточний
	50	FPV SHURIKEN 10	3 бат 28 ОМБр	945 000,00	Гривня	Платіжний документ	Поточний
	50	FPV SHURIKEN 10	241 обр ТрО 130 об ТрО Котики	850 000,00	Гривня	Платіжний документ	Поточний
	50	FPV SHURIKEN 10	РУБпАК 104 ТРО	850 000,00	Гривня	Платіжний документ	Поточний
	100	FPV SHURIKEN 10	3 бат 46 бригада Злі бобри	1 700 000,00	Гривня	Платіжний документ	Поточний
	50	FPV SHURIKEN 10	5 прикорд. загін, 3 комендатура 14 застава	850 000,00	Гривня	Платіжний документ	Поточний
	50	FPV SHURIKEN 7	3 бат 46 бригада Злі бобри	650 000,00	Гривня	Платіжний документ	Поточний
	20	FPV SHURIKEN 7	Flying Skull	315 300,00	Гривня		Поточний
	10	FPV **********	Flying Skull	221 030,00		Платіжний документ	Небесний
	100	FPV SHURIKEN 7	59 ОШБр, 108 ОМБ, Вовки Да Вінчі	1 300 000,00	Гривня	Платіжний документ	Поточний
	50	FPV SHURIKEN 10	Група Поприсве 3 полк ССО	850 000,00	Гривня	Платіжний документ	Поточний
	10	FPV Babay DJI o3	ГУР МОУ «Крила»	8 115,00	Долар		
	50	FPV SHURIKEN 7	НЕБЕСНА ЛЮТЬ РУБпАК 71 бр	650 000,00	Гривня	Платіжний документ	Поточний
	100	FPV SHURIKEN 7	28 ОМБр БУАР	1 300 000,00	Гривня	Платіжний документ	Поточний
	100	FPV SHURIKEN 7	68 ОЄБР	1 300 000,00	Гривня	Платіжний документ	Поточний
	100	FPV SHURIKEN 7	54 ОМБР Sky Lords	1 300 000,00	Гривня	Платіжний документ	Поточний

Рисунок 1.4 — Сторінка «Звіти», БФ Спільнота Стерненка

Ще одним прикладом платформи, що активно використовує цифрові інструменти в гуманітарній сфері, є благодійний фонд «Повернись живим». Це одна з наймасштабніших організацій в Україні, що з 2014 року здійснює підтримку військових підрозділів. Її діяльність охоплює не лише закупівлю спорядження, техніки та навчання, але й впровадження технічних засобів контролю, аналітики та звітності.

Сайт фонду виконує функцію потужної інформаційно-презентаційної платформи. Через нього реалізовано повноцінну систему прийому пожертв:

користувачі можуть одноразово або регулярно підтримувати конкретні напрями діяльності. Інтеграція з банківськими інструментами це дозволяє автоматизувати облік надходжень та формувати фінансову звітність. Сторінка з формою для пожертв показана на рисунку 1.5 Усі надходження та витрати супроводжуються публічними звітами, які оновлюються на сайті (див. рис. 1.6) у відкритому доступі.

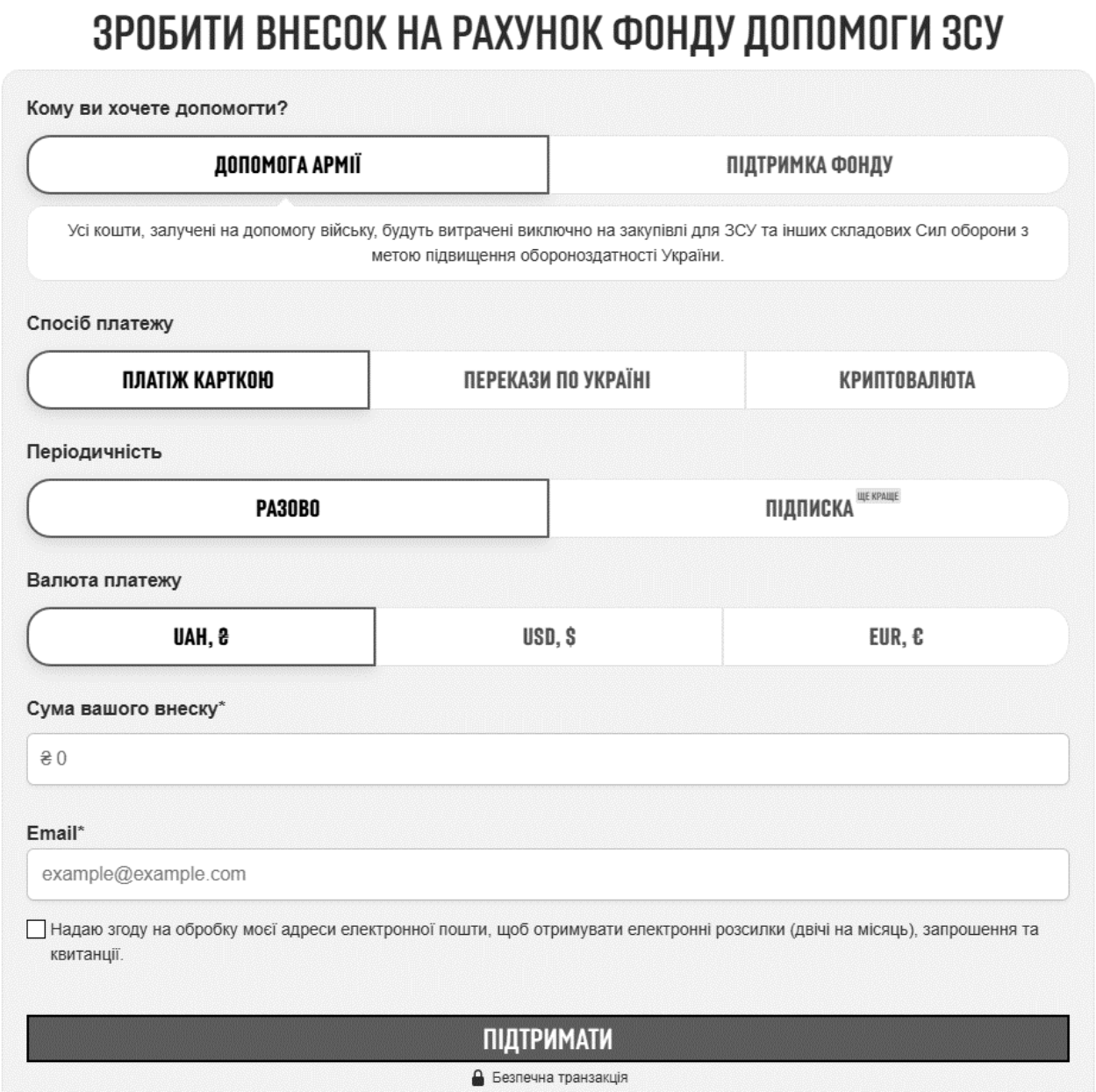


Рисунок 1.5 — Форма для пожертв реалізована сайтом Повернись Живим

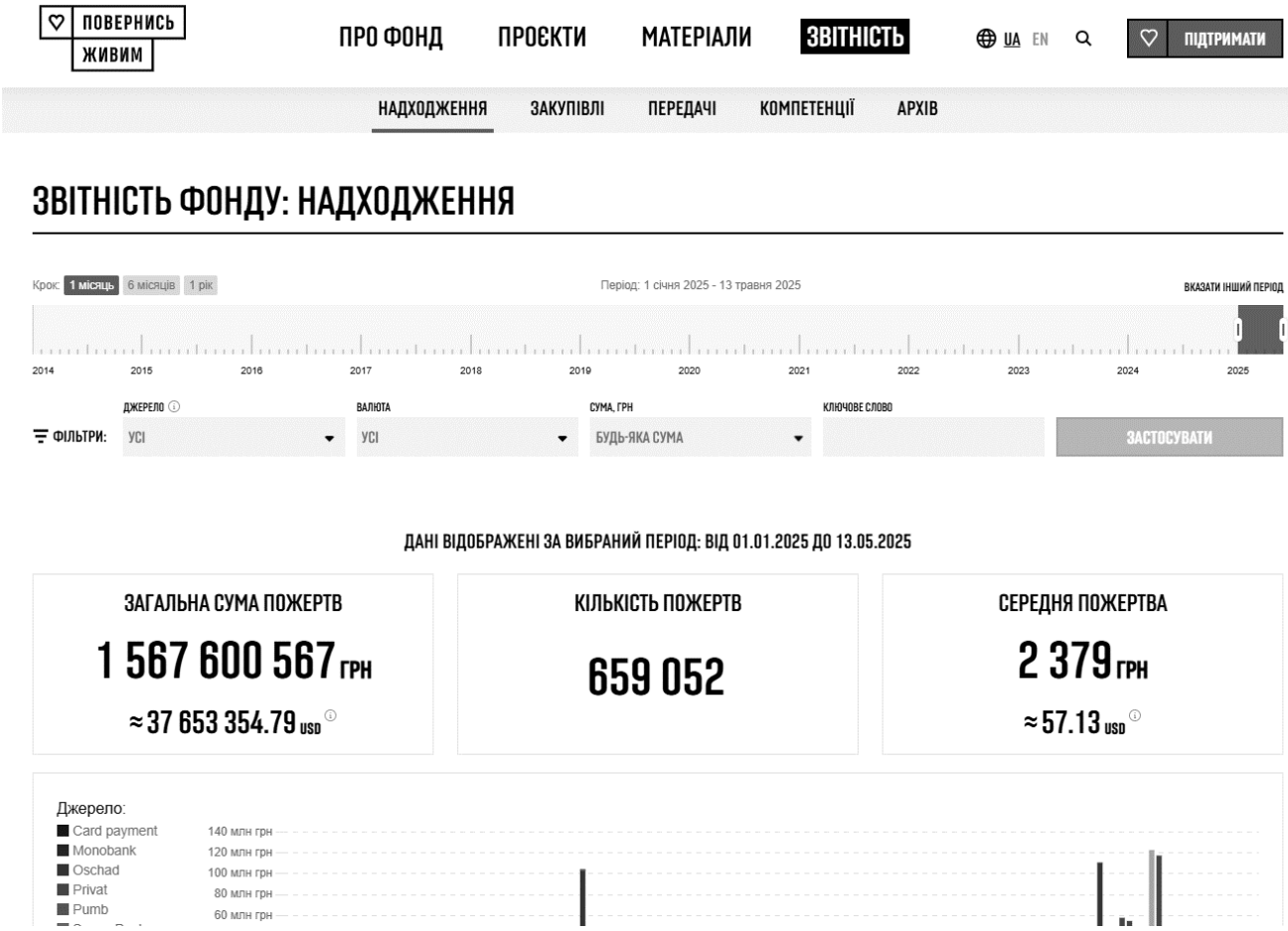


Рисунок 1.6 — Сторінка звітності витрачених коштів Повернись Живим

Проте, попри високий рівень прозорості, фонд «Повернись живим» не надає користувачам можливості подавати заявки на допомогу або зареєструватися в системі для подальшої взаємодії. Вся логіка побудована навколо централізованого управління — тобто ініціатива з боку фонду. Таким чином, фонд ефективно виконує роль прозорої платформи збору та використання ресурсів, але не виступає як інформаційна система обліку потреб і наданої допомоги у класичному розумінні.

Попри це, приклад «Повернись живим» заслуговує уваги як еталон масштабної донорської системи, у якій особливу роль відіграють надійність, звітність і технологічна інтеграція з фінансовими інструментами. Це демонструє, як навіть без реєстрації користувачів та обробки заявок можна досягти високого рівня довіри й ефективності за рахунок автоматизації ключових процесів — зокрема фінансового обліку та звітності.

Прикладом повноцінної державної інформаційної системи, орієнтованої саме

на централізований облік гуманітарної допомоги, є Автоматизована система реєстрації гуманітарної допомоги (АС ГД). Ця платформа була розроблена за ініціативи Міністерства соціальної політики України та набула офіційного статусу з обов'язковим використанням з 1 квітня 2024 року для всіх організацій, що здійснюють ввезення чи розподіл гуманітарної допомоги.

АС ГД реалізована як веб-платформа з доступом для благодійних фондів, державних установ, волонтерських організацій і митних служб. Її основний функціонал включає реєстрацію отримувачів гуманітарної допомоги, подання декларацій на ввезення вантажів, облік надходжень та відправлень, а також формування звітності за заданими параметрами. Система інтегрована з митними службами, що дозволяє автоматизувати процес контролю переміщення гуманітарних вантажів через кордон.

АС ГД має чітку логіку користування: організація створює обліковий запис, після чого може вносити інформацію про надану або отриману допомогу, прикріплювати документи, вказувати місце призначення, обсяг і тип ресурсу. Це дозволяє уникати дублювання, відстежувати ланцюг постачання й забезпечити прозорість на всіх етапах.

Серед недоліків варто відзначити складність у використанні для малих або неформалізованих волонтерських груп. Інтерфейс вимагає певної підготовки, а процес реєстрації організації передбачає надання офіційних документів. Крім того, система орієнтована насамперед на юридичних осіб і передбачає суворе дотримання процедур.

Незважаючи на це, АС ГД — це приклад функціонально повної та юридично інтегрованої ІС, яка може виступати як еталон у плані обліку, звітності та контролю в сфері гуманітарної допомоги на державному рівні. Вона демонструє, як цифрові рішення можуть забезпечити прозору та контрольовану взаємодію між усіма учасниками процесу — від донора до кінцевого отримувача.

Таблиця 1.1 — Порівняльна характеристика існуючих платформ

Критерій	Спільнота Стерненка	Повернись живим	АС ГД (державна система)
Тип платформи	Волонтерський проєкт	Благодійна організація	Державна інформаційна система
Наявність обліку заявок	Відсутній	Відсутній	Реалізований повною мірою
Формат звітності	Публічні звіти у відкритому доступі	Автоматизовані фінансові звіти	Офіційна звітність у межах системи
Інтеграція з базами даних	Відсутня	Часткова інтеграція (API)	Повна інтеграція з державними реєстрами
Зручність використання	Висока	Висока	Складна для невеликих організацій
Рівень автоматизації	Мінімальний	Частковий (платежі та облік)	Повний (облік, заявки, звіти)

1.4 Постановка задачі

У результаті проведеного аналізу предметної області було з’ясовано, що процес обліку гуманітарної допомоги в багатьох організаціях, зокрема волонтерських та благодійних ініціативах, часто здійснюється вручну або із застосуванням малоефективних інструментів, таких як електронні таблиці чи месенджери. Це призводить до дублювання даних, втрати інформації, низької прозорості розподілу ресурсів та обмежених можливостей аналітики.

Огляд існуючих інформаційних систем, які використовуються у сфері гуманітарної допомоги дозволив зробити висновок, що жодне з них не забезпечує водночас простоту впровадження, гнучкість, масштабованість і відкритість архітектури. Деякі системи є надто складними для невеликих ініціатив, інші нереалізовують повноцінний облік або не мають системного підходу до керування заявками.

Таким чином, було сформульовано необхідність створення власної інформаційної системи, яка дозволить автоматизувати основні процеси: подання та реєстрацію заявок на допомогу, облік ресурсів, формування звітів і взаємодію між користувачами з різними ролями. Особливу увагу при цьому передбачено приділити зручності інтерфейсу, мінімальним вимогам до розгортання системи, а також можливості подальшого масштабування.

Розробка такої системи передбачає застосування сучасних веб-технологій, побудову клієнт-серверної архітектури та реалізацію документоорієнтованої бази даних, що дозволить забезпечити ефективне зберігання даних із можливістю масштабування.

2 АНАЛІЗ ТА ОБҐРУНТУВАННЯ ВИБОРУ ТЕХНОЛОГІЙ

2.1 Обґрунтування вибору архітектури інформаційної системи

Розробка сучасної інформаційної системи потребує не лише підбору відповідних інструментів реалізації, а насамперед — раціонального підходу до архітектурної побудови. Вибір архітектури визначає основні принципи функціонування майбутнього програмного продукту, впливає на швидкодію, масштабованість, безпеку та зручність обслуговування. У випадку створення системи обліку гуманітарної допомоги, яка має забезпечувати доступ багатьох користувачів, обробку заявок і збереження структурованої інформації, критично важливим є обрати надійну, гнучку та розширювану архітектуру.

На початковому етапі були розглянуті базові моделі розробки програмного забезпечення — каскадна, інкрементна, спіральна та гнучкі підходи (Agile). Каскадна модель, хоча й відзначається чіткою структурою, не враховує змін у вимогах під час реалізації, що є поширеним явищем у проектах, орієнтованих на волонтерські або громадські ініціативи. Натомість ітеративна модель розробки була обрана як основна, оскільки вона дозволяє створювати функціонал поступово, тестуючи й покращуючи систему на кожному кроці. Це забезпечує адаптацію до змінних умов, гнучке оновлення функціоналу та кращу взаємодію з користувачем.

З огляду на очікувану багатокористувацькість та потребу у відокремленні логіки обробки даних від інтерфейсу, було прийнято рішення використовувати клієнт-серверну архітектуру [3]. Діаграма архітектури зображена на рисунку 2.1.

Такий підхід передбачає чітке розділення функцій між двома частинами: клієнтом, який відповідає за відображення даних і взаємодію з користувачем, та сервером, що обробляє запити, керує базою даних та реалізує бізнес-логіку. Цей тип архітектури має низку переваг: він дозволяє масштабувати систему при зростанні кількості користувачів, легко адаптується до змін, а також забезпечує централізоване зберігання даних, що важливо для безпеки та консистентності.

Такий підхід передбачає чітке розділення функцій між двома частинами: клієнтом, який відповідає за відображення даних і взаємодію з користувачем, та

сервером, що обробляє запити, керує базою даних та реалізує бізнес-логіку. Цей тип архітектури має низку переваг: він дозволяє масштабувати систему при зростанні кількості користувачів, легко адаптується до змін, а також забезпечує централізоване зберігання даних, що важливо для безпеки та консистентності.

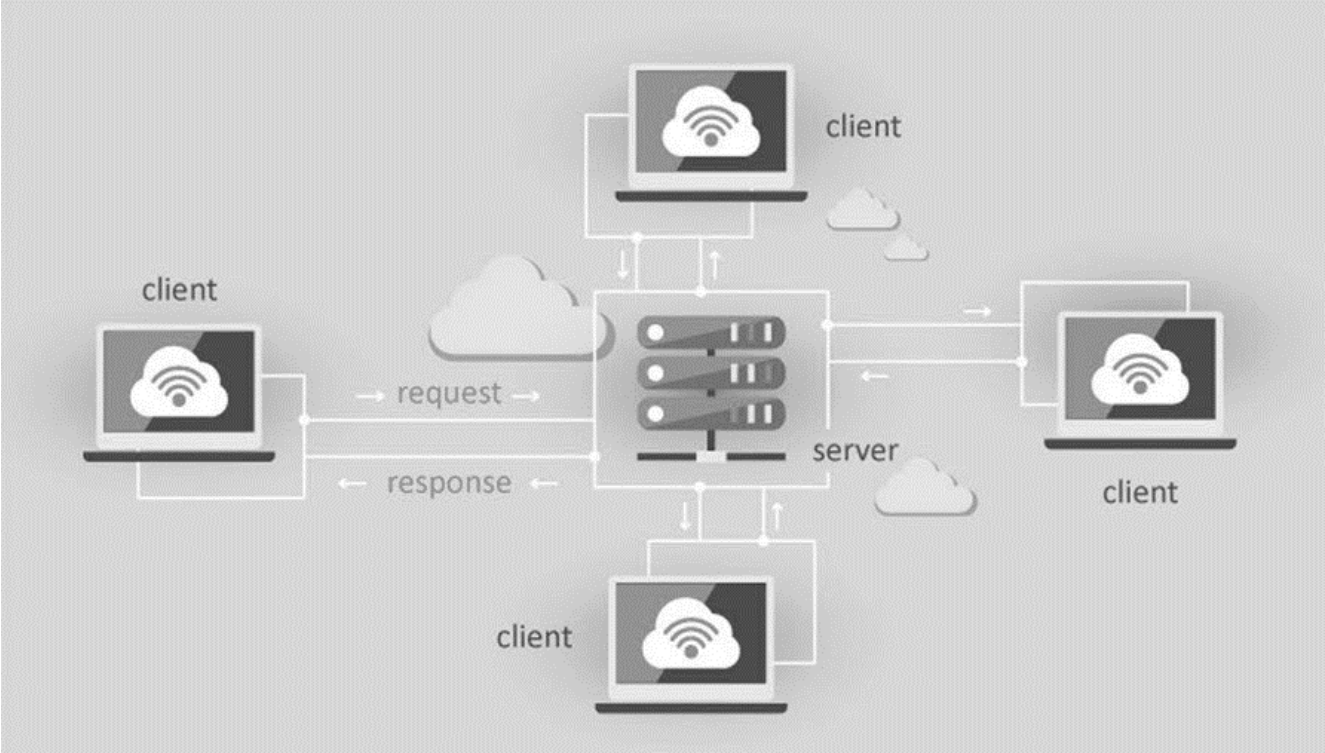


Рисунок 2.1 — Схема клієнт-серверної архітектури

Такий підхід передбачає чітке розділення функцій між двома частинами: клієнтом, який відповідає за відображення даних і взаємодію з користувачем, та сервером, що обробляє запити, керує базою даних та реалізує бізнес-логіку. Цей тип архітектури має низку переваг: він дозволяє масштабувати систему при зростанні кількості користувачів, легко адаптується до змін, а також забезпечує централізоване зберігання даних, що важливо для безпеки та консистентності.

Крім того, клієнт-серверна модель дозволяє створити основу для подальшого переходу до більш складної багаторівневої архітектури або навіть мікросервісної, якщо в майбутньому виникне потреба у модульності чи високій навантажувальності системи. У нашому випадку це відкриває можливість

додавання нових модулів, наприклад, аналітики, інтеграції з платіжними системами або зовнішніми API без повного переписування системи.

Оскільки система створюється як веб-додаток, обраний архітектурний підхід також дозволяє забезпечити платформну незалежність: користувачі зможуть отримувати доступ до сервісу з будь-якого пристрою, що має браузер. Це критично важливо для благодійних організацій і волонтерів, які можуть працювати з різних регіонів та пристроїв.

Таким чином, застосування ітеративного підходу до розробки у поєднанні з клієнт-серверною архітектурою забезпечує гнучкість, розширюваність, зручність обслуговування та високий рівень адаптивності — всі ці характеристики є необхідними для створення ефективної інформаційної системи обліку гуманітарної допомоги.

2.2 Вибір мови програмування

Одним із найважливіших рішень у процесі створення інформаційної системи є вибір мови програмування, адже саме вона визначає рівень складності розробки, швидкість реалізації, можливості масштабування та гнучкість підтримки проєкту. У випадку інформаційної системи обліку гуманітарної допомоги важливо було обрати такі засоби, які дозволяють створити легкий, зрозумілий і доступний інтерфейс, придатний для взаємодії з користувачами різного рівня технічної підготовки.

На початковому етапі розглядалось декілька мов програмування, які потенційно могли бути використані для реалізації цієї системи:

Python, зокрема в поєднанні з фреймворком Django або Flask. Ця мова вирізняється лаконічним синтаксисом, широкими можливостями для обробки даних, хорошою документацією. Однак для повноцінної реалізації веб-інтерфейсу з боку клієнта все одно потрібно було б додатково впроваджувати HTML, CSS і JavaScript. Крім того, розгортання Python-серверу вимагає додаткових налаштувань, що ускладнює швидке впровадження у волонтерських або непрофесійних середовищах.

PHP це доволі поширене рішення для серверної частини веб-додатків. Його перевагою є простота інтеграції з HTML, наявність великої кількості готових CMS та шаблонів. Проте мова вважається менш сучасною порівняно з іншими альтернативами, а її застосування доцільне переважно в класичних CRUD-застосунках або за умов обмежених ресурсів. У рамках цієї роботи було вирішено уникати залежності від сторонніх CMS, і натомість зробити акцент на базових відкритих інструментах [6].

Java також була розглянута як одна з найпотужніших мов для побудови масштабованих корпоративних систем. Вона забезпечує високу продуктивність, безпеку та багатопоточність. Однак складність середовища, необхідність встановлення JVM, громіздкість при створенні простих веб-додатків та потреба в додаткових інструментах (Spring, JSP) зробили цей варіант надмірним для цілей дипломної роботи.

Після аналізу переваг і недоліків альтернатив було прийнято рішення зосередитися на класичному стеку веб-технологій для клієнтської частини, а саме:

HTML (HyperText Markup Language) — мова розмітки, яка дозволяє створити базову структуру сторінки, визначаючи логіку відображення інформації, полів для вводу, кнопок та таблиць. Її застосування є обов’язковим у будь-якому веб-додатку, а її підтримка є універсальною на всіх платформах[8].

CSS (Cascading Style Sheets) – засіб стилізації, який забезпечує зовнішній вигляд елементів сторінки: кольори, шрифти, відступи, позиціонування, адаптивний дизайн. Завдяки CSS система набуває зручного інтерфейсу, зрозумілого як для звичайного користувача, так і для адміністратора[9],[10].

JavaScript це мова клієнтської логіки, яка дає змогу додавати інтерактивність: обробку подій, валідацію форм, а також (у перспективі) — обмін даними з сервером через API або AJAX-запити. JavaScript підтримується всіма сучасними браузерами й дозволяє працювати без потреби в сторонньому ПЗ [11].

Таке рішення є оптимальним для проєкту, який має на меті створення простої, адаптивної та доступної системи, що працює без встановлення програм на ПК користувача, використовуючи лише браузер. Обраний стек технологій дозволяє

швидко створити прототип, поетапно розширювати функціонал та інтегруватися в майбутньому з серверною частиною, яка може бути реалізована на будь-якій сумісній мові (наприклад, PHP або Python).

Крім того, HTML, CSS та JavaScript є відкритими стандартами, що не прив’язують розробку до конкретного виробника або програмного середовища, а отже, повністю відповідають принципам прозорості, масштабованості та мінімальних вимог до апаратних ресурсів — що є критично важливим у контексті гуманітарної діяльності.

2.3 Вибір інструментів для розробки веб-інтерфейсу

Розробка користувацького інтерфейсу є одним із найбільш відповідальних етапів при створенні інформаційної системи, адже саме він визначає зручність взаємодії з додатком, логіку користування, доступність та загальне враження від системи. У випадку системи обліку гуманітарної допомоги особливо важливо зробити інтерфейс інтуїтивно зрозумілим навіть для людей без технічної підготовки, зокрема волонтерів, координаторів і представників благодійних фондів.

Для розробки веб-інтерфейсу було вирішено використовувати комбінацію HTML, CSS та JavaScript у поєднанні з текстовим редактором Visual Studio Code як основним інструментом програмування. Це рішення базується на поєднанні зручності, поширеності та гнучкості у використанні.

Visual Studio Code — безкоштовне, кросплатформне середовище розробки від Microsoft, яке має розширену підтримку HTML, CSS і JavaScript, дозволяє підключати різноманітні розширення, а також підтримує контроль версій через Git. Завдяки високій продуктивності, легкому інтерфейсу та потужному автодоповненню VS Code дає змогу ефективно реалізовувати проєкти будь-якої складності без потреби у важких IDE.

Для візуального структурування та створення інтерфейсу було використано шаблонний підхід: спочатку створювались базові сторінки HTML, до яких додавалась стилізація за допомогою CSS. Стилі розділяються на глобальні та

локальні — це дозволяє гнучко контролювати зовнішній вигляд елементів без дублювання коду. Додатково використовуються адаптивні медіа-запити, що дозволяє інтерфейсу коректно відображатися як на десктопах, так і на мобільних пристроях.

У процесі роботи також розглядалася можливість використання CSS-фреймворків, таких як Bootstrap або Bulma. Ці інструменти дозволяють значно пришвидшити розробку, однак мають типову стилістику, яка потребує подальшого кастомізування. З метою досягнення унікального дизайну було вирішено обійтися власними CSS-стилями, при цьому зберігаючи хорошу читаємість і адаптивність макету.

З боку JavaScript було прийнято рішення реалізувати взаємодію з формами, валідацію введених даних, а також динамічне оновлення елементів сторінки. Завдяки використанню нативного JavaScript, вдалося уникнути перевантаження додатку зовнішніми бібліотеками, що зберегло високу швидкодію й легкість коду. В якості допоміжних інструментів також застосовуються:

— Live Server це розширення до VS Code, що дозволяє переглядати зміни в реальному часі без перезапуску браузера;

— Emmet використовується для прискореного написання HTML-розмітки;

— Git допомагає зберігати версій проєкту та можливості відкату при потребі.

Загалом, вибрані інструменти дозволяють організувати швидку, зрозумілу та ефективну розробку інтерфейсу з мінімальними вимогами до ресурсів. Вони не потребують ліцензій, легко налаштовуються, і тому є ідеальними для невеликих проєктів, що реалізуються в освітньому або волонтерському середовищі.

2.4 Вибір засобів для реалізації бази даних

База даних є одним із ключових елементів інформаційної системи, оскільки забезпечує надійне зберігання, обробку та доступ до структурованих даних. У процесі розробки системи обліку гуманітарної допомоги було проаналізовано кілька варіантів сучасних систем управління базами даних (СУБД), що дозволяють

ефективно реалізувати поставлені завдання: збереження інформації про користувачів, обробка запитів, журналювання дій тощо.

Серед основних критеріїв вибору були: підтримка реляційної моделі, відкритість платформи, наявність активної спільноти, безпека, масштабованість, а також сумісність із вибраним технологічним стеком. Було розглянуто такі СУБД, як MySQL, PostgreSQL та MongoDB [7, 12].

Система MySQL є однією з найпоширеніших реляційних СУБД у світі. Вона має зручний синтаксис, високу швидкодію при виконанні простих запитів та великий обсяг документації. Однак її обмежені можливості з роботи зі складними типами даних та транзакціями дещо знижують гнучкість у складних проєктах.

MongoDB, як документно-орієнтована СУБД, надає гнучкість у структурі даних і добре підходить для зберігання неструктурованої інформації. Проте для реалізації чітких зв’язків між сутностями та забезпечення цілісності даних, що є критичним у випадку обліку заявок та користувачів.

Після попереднього аналізу було ухвалено рішення використовувати MongoDB — документно-орієнтовану NoSQL базу даних, яка дедалі частіше застосовується в сучасних веб-проєктах завдяки своїй простоті, гнучкості та масштабованості [7].

MongoDB зберігає дані у вигляді документів у форматі BSON (бінарний аналог JSON), що дозволяє легко працювати з динамічними структурами даних. Це особливо корисно у випадку, коли структура записів може змінюватися або бути різною залежно від типу запиту. Наприклад, у системі подання заявок на допомогу частина заявок може містити додаткові поля, прикріплені файли або специфічну метаінформацію. У традиційних реляційних СУБД для цього необхідно створювати додаткові таблиці, зв’язки та зовнішні ключі, що ускладнює структуру та уповільнює обробку запитів.

MongoDB забезпечує горизонтальне масштабування за допомогою шардінгу, що дозволяє обробляти великі обсяги даних при зростанні кількості користувачів і запитів. Це важлива перевага в умовах використання системи у середовищах з високим навантаженням, як-от великі волонтерські платформи або державні

структури. Крім того, MongoDB має вбудовані механізми реплікації, резервного копіювання та індексації, що забезпечує надійність і безперервність доступу до даних.

Важливим фактором при виборі MongoDB стало також її зручне інтегрування з JavaScript-технологіями, які використовуються на клієнтській частині системи. Завдяки JSON-подібному формату даних, передача, обробка та візуалізація запитів значно спрощується, а логіка програми стає прозорішою.

Таким чином, вибір MongoDB є виправданим у контексті даного проєкту, який не потребує складних транзакційних зв’язків, але вимагає гнучкості, швидкої розробки та можливості масштабування в майбутньому. Це рішення дозволяє ефективно обробляти дані користувачів, заявки, коментарі, документи та інші компоненти системи, забезпечуючи при цьому зручну підтримку і розвиток платформи.

Таблиця 2.1 — Порівняння СУБД, що розглядалися

СУБД	Тип	Переваги	Недоліки
MySQL	Реляційна	Простота, швидкість, підтримка CMS	Обмежена робота з транзакціями
PostgreSQL	Реляційна	Потужність, типізація, розширення	Складніше в конфігурації
MongoDB	Документна	Гнучкість структури, швидкість	Відсутність зв’язків, не-SQL

3 РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ ОБЛІКУ ГУМАНІТАРНОЇ ДОПОМОГИ

3.1 Проєктування архітектури та розробка загальної структури інформаційної системи

Проєктування архітектури та побудова загальної структури інформаційної системи — це складний та багатоетапний процес, що охоплює аналіз вимог, вибір моделі архітектури, проєктування структури бази даних, підбір технологій розробки, створення користувацького інтерфейсу та визначення логіки взаємодії між компонентами. Усе це спрямоване на те, щоб забезпечити ефективну, стабільну й масштабовану роботу інформаційної системи в умовах реального навантаження. Наприклад, у деяких випадках доцільним може бути застосування розподіленої архітектури, за якої різні підсистеми функціонують на окремих серверах або фізично розділених вузлах, що дозволяє підвищити масштабованість і відмовостійкість програмного комплексу. Однак у нашому випадку, враховуючи потребу у швидкому розгортанні та зручності обслуговування, було обрано класичну клієнт-серверну архітектуру [13, 14, 15].

Для зберігання й обробки даних можуть застосовуватись різні системи керування базами даних, серед яких найбільш поширеними є MySQL, PostgreSQL і MongoDB. З огляду на потребу у структурованому зберіганні заявок, користувачів і зв’язаних даних, для реалізації інформаційної системи обліку гуманітарної допомоги було обрано MongoDB. Вона забезпечує гнучку схему даних, підтримку транзакцій, ефективну роботу з великим обсягом запитів.

У процесі розробки особлива увага приділяється потребам користувачів, простоті використання та доступності інтерфейсу. Саме тому для побудови клієнтської частини застосовуються сучасні вебтехнології, які дозволяють створити зручний вебінтерфейс, доступний із будь-якого пристрою, що має підключення до Інтернету. Це забезпечує гнучкий доступ до системи як для співробітників фонду, так і для користувачів, які подають заявки на допомогу.

У межах реалізації інформаційної системи гуманітарної допомоги була обрана клієнт-серверна структура, що дозволяє розподілити відповідальність між двома основними логічними рівнями: клієнтським — у вигляді вебінтерфейсу, з яким взаємодіє користувач, і серверним — де обробляються запити, керуються дані та реалізовано перевірку прав доступу. Такий підхід є найдоцільнішим з погляду розширюваності, безпеки та підтримки. Крім того, він дозволяє ефективно організувати керування ролями — користувач, модератор, адміністратор — кожна з яких має визначений набір повноважень, що реалізуються на рівні серверної логіки [16].

Структурну модель розгортання програмних компонентів системи в інфраструктурі представлено у вигляді UML-діаграми розгортання, що дозволяє наочно відобразити взаємозв'язки між елементами системи, серверами, клієнтами та базою даних. Схему розгортання наведено на рисунку Б.1.

Використання вебзастосунку як базової платформи дає низку переваг. По-перше, система є незалежною від платформи, тобто працює однаково на різних пристроях. Це важливо для користувачів, які мають обмежений доступ до техніки, але потребують взаємодії з фондом.

По-друге, система є централізованою, що дозволяє швидко вносити зміни, оновлювати функціонал та контролювати безпеку даних. Оновлення здійснюється на стороні сервера і не потребує додаткових дій від користувача [17].

Клієнтська частина розроблена з використанням HTML, CSS і JavaScript. HTML формує структуру сторінки, CSS відповідає за зовнішній вигляд, а JavaScript забезпечує динамічну поведінку, обробку подій та взаємодію з сервером [18]. У результаті формується привабливий, швидкий і зручний вебінтерфейс. Завдяки такому підходу користувачі можуть легко подати заявку, прикріпити фото, отримати повідомлення про розгляд запиту та залишатися в курсі поточного стану.

Серверна частина системи відповідає за бізнес-логіку. Вона обробляє усі запити, які надходять від клієнтів, виконує перевірку автентичності, керує базою даних і формує відповіді. Дані зберігаються у структурованому вигляді, що дозволяє здійснювати пошук, фільтрацію, сортування і контроль над усіма етапами

роботи. У майбутньому така архітектура дає змогу легко реалізувати нові модулі — аналітику, інтерфейси для зовнішніх партнерів, автоматичну звітність, сповіщення, інтеграцію з CRM або мобільними додатками.

Враховуючи специфіку нашої інформаційної системи, що орієнтована на взаємодію з широким колом користувачів, важливою вимогою є забезпечення доступності з будь-якого пристрою з виходом в інтернет. Саме тому особливий акцент зроблено на адаптивності інтерфейсу, швидкодії та зручності навігації. Вебзастосунок розроблено таким чином, щоб його функціонал був інтуїтивно зрозумілим, навіть для користувачів, які не мають глибоких технічних знань. Це дозволяє залучати більше людей до використання платформи та підвищує ефективність взаємодії між усіма сторонами, які беруть участь у наданні або отриманні допомоги.

Використання вебдодатку як основної платформи для роботи з інформаційною системою благодійної організації сприяє покращенню комунікації, спрощенню внутрішніх процесів обліку заявок, скороченню часу реагування та підвищенню довіри до процесу надання допомоги. Прозорість процесів і зручний доступ до інформації дозволяє не лише залучити нових користувачів, а й стимулювати зворотний зв’язок і побудову спільноти навколо платформи.

Для розробки вебінтерфейсу було обрано сучасний технологічний стек. HTML забезпечує логічне структурування контенту: заголовки, абзаци, списки, таблиці — усе це дозволяє формувати сторінки, що є зручними для читання та навігації. CSS використовується для стилізації елементів: від кольорів і шрифтів до розміщення блоків, створення темної/світлої теми, тіней та анімацій. Це дозволяє надати вебінтерфейсу професійного вигляду, зробити його естетично привабливим. JavaScript, своєю чергою, дозволяє зробити сайт живим: форма авторизації реагує миттєво на некоректні дані, списки заявок автоматично оновлюються без перезавантаження, зображення завантажуються асинхронно, а статуси змінюються одразу після дії адміністратора чи модератора. Такий підхід значно покращує користувацький досвід та формує відчуття надійності й швидкодії системи.

Серверна частина, окрім обробки запитів і збереження інформації в базі

даних, виконує й ряд інших критично важливих функцій. Зокрема, вона перевіряє наявність облікового запису, проводить автентифікацію, призначає ролі, записує зміни до журналу дій. Усі дії модераторів і адміністраторів логуються, що дозволяє відслідковувати історію обробки запитів і, у разі потреби, повертатися до попередніх станів. Реалізований механізм сортування і фільтрації дозволяє адміністрації швидко знаходити потрібні запити, формувати статистику, виявляти дублікати чи відхилені заявки. Це не лише спрощує роботу модераторів, а й підвищує ефективність загальної взаємодії.

Захист даних — ще один важливий аспект. Усі чутливі дані, включаючи паролі, зберігаються в зашифрованому вигляді. Сервер не передає клієнту зайвих даних, а лише те, що необхідно в конкретний момент. Такий підхід мінімізує ризики витоку інформації. Крім того, у планах реалізація двофакторної автентифікації, що значно підвищить безпечність системи.

Щодо логіки взаємодії в системі, ключовим є процес авторизації, який починається з перевірки введених користувачем даних. Якщо адреса електронної пошти вже зареєстрована, система пропонує виконати вхід. У разі відсутності такої пошти — користувачу запропоновано зареєструватися, заповнивши відповідну форму. У випадку некоректного логіна або пароля, система одразу повідомляє про помилку. Це дозволяє уникати непорозумінь та забезпечує зручний вхід у систему. Якщо користувач має правильні дані, система виконує перевірку його ролі. У разі наявності прав адміністратора він отримує доступ до панелі керування, де може переглядати всі запити, змінювати статуси, додавати або редагувати записи, а також самостійно створювати публікації. Якщо користувач є звичайним зареєстрованим учасником, він отримує доступ до особистого кабінету, де може створити запит на допомогу, переглядати стан заявки та отримувати зворотний зв’язок.

У реалізованій структурі системи кожна роль має чітко визначене коло повноважень. Користувач може взаємодіяти тільки з власними заявками, модератор має доступ до загального переліку запитів, проте не може змінювати ролі або налаштування системи, а адміністратор — має найвищий рівень доступу

та може виконувати дії з усіма об'єктами бази даних. Це дозволяє підтримувати порядок, розмежувати відповідальність, уникати конфліктів доступу та підвищувати загальну безпеку функціонування платформи.

Однією з цілей, яку ми ставили перед собою при проєктуванні системи, було створення максимально відкритої й прозорої платформи, яка не вимагатиме реєстрації для ознайомлення з основною інформацією про діяльність фонду. Тому навіть неавторизовані користувачі можуть переглядати новини, ознайомлюватися з описом запитів, переглядати реквізити для переказів та вільно копіювати їх. Ця функція реалізована спеціально для того, щоб кожен охочий міг допомогти без зайвих технічних бар’єрів. У разі, якщо користувач захоче подати власний запит — йому пропонується створити обліковий запис.

Після авторизації користувачеві надається доступ до функціоналу, що відповідає його ролі. В особистому кабінеті можна створити нову заявку, додати до неї короткий опис ситуації, прикріпити фото або скановані документи, відправити на розгляд.

З боку адміністратора доступна розширена панель, яка дозволяє переглядати всі подані заявки, змінювати їх статус, публікувати або приховувати, видаляти помилкові або дубліковані записи, а також створювати власні заявки від імені організації

На завершення слід зазначити, що замість використання локального встановлення програмного забезпечення на кожному пристрої, обрана веборієнтована клієнт-серверна архітектура дозволяє централізовано зберігати всі обчислювальні потужності та логіку на сервері. Це означає, що навіть користувачі з обмеженими ресурсами, застарілими пристроями або слабким підключенням до Інтернету можуть ефективно працювати з системою. Вся складна логіка обробки відбувається на сервері, що значно знижує вимоги до кінцевих пристроїв, а отже — розширює доступність сервісу.

Централізація також позитивно впливає на безпеку системи. Вся чутлива інформація, така як облікові дані користувачів, історія заявок, тощо — зберігається на захищеному сервері, доступ до якого обмежено лише відповідним правам. Це

дозволяє уникнути дублювання даних, втрати інформації або несанкціонованого доступу.

Таким чином, вибір клієнт-серверної архітектури з вебінтерфейсом та централізованою обробкою даних є виправданим технічним і стратегічним рішенням. Він забезпечує стабільність, безпеку, доступність, простоту розгортання та підтримку, а також дозволяє розвивати систему відповідно до зростаючих потреб користувачів і організацій, що працюють у сфері гуманітарної допомоги.

3.2 Розробка клієнтської та серверної частини додатку

Основною метою створення даної інформаційної системи є організація публічного доступу до інформації про діяльність благодійного фонду, що дозволить підвищити рівень обізнаності громадськості та сприятиме залученню нових учасників до реалізації ініціатив та проєктів. З цією метою передбачено побудову структури інтерфейсу, яка включатиме окремі сторінки та функціональні елементи, орієнтовані на зручне подання й сприйняття контенту.

З цією метою в межах вебдодатку реалізовано декілька окремих сторінок, кожна з яких має індивідуальну структуру та визначений функціональний набір. Першою сторінкою, яку побачить користувачу при зверненні до системи, є головна сторінка. Зображення головної сторінки наведено на рисунку 3.1 та 3.2.

Головна сторінка є базовою точкою входу користувача до системи. У верхній частині розміщено навігаційну панель, яка містить кнопки для переходу до основних розділів: «Потреби», «Історії», «Увійти» та кнопка перемикання теми. Елемент реалізовано у вигляді фіксованого верхнього меню, що забезпечує постійний доступ до навігаційних елементів під час прокручування сторінки.

Основним візуальним компонентом головної сторінки є банер, реалізований у вигляді слайдера з перемиканням вмісту. Банер містить графічні блоки, які відображають поточні активні збори або інформаційні кампанії, що проводяться через платформу.

Нижче банера розміщено інформаційний блок, у якому описано призначення платформи, основні цілі її використання, історії людей, якими була отримана

допомога, а також коротка загальна статистика за весь час роботи фонду.

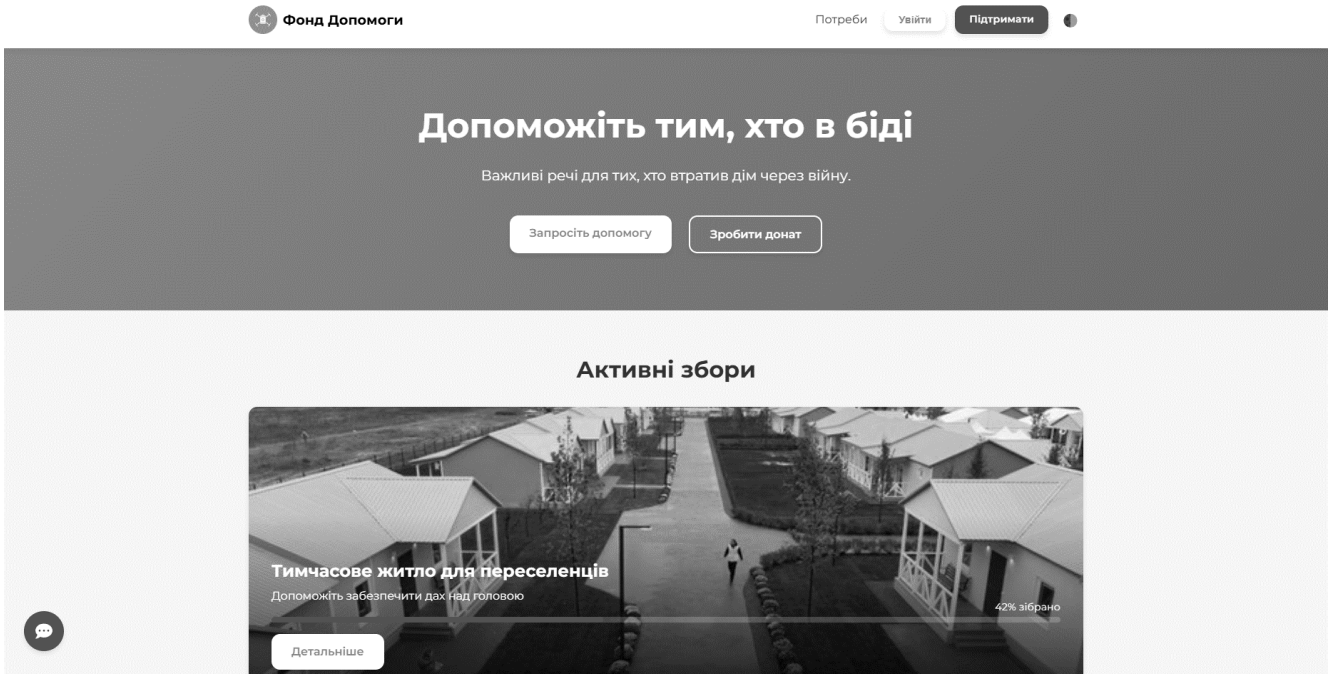


Рисунок 3.1 — Головна сторінка фонду

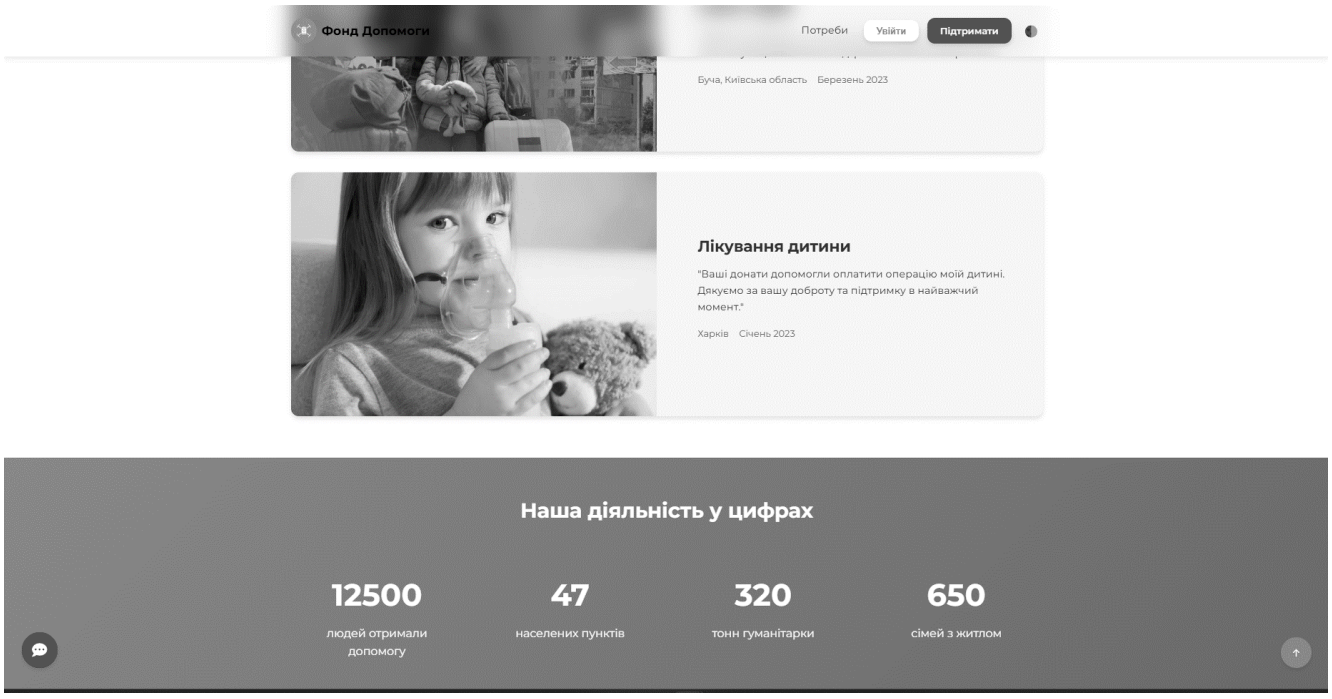


Рисунок 3.2 — Частина головної сторінки з історіями людей та статистикою

Сторінка авторизації та реєстрації користувача призначена для забезпечення контролю доступу до функціоналу інформаційної системи обліку гуманітарної

допомоги. Вона є обов’язковим елементом будь-якої платформи з персоніфікованим обліком користувачів. Вигляд сторінки реєстрації показано на рисунку 3.3.

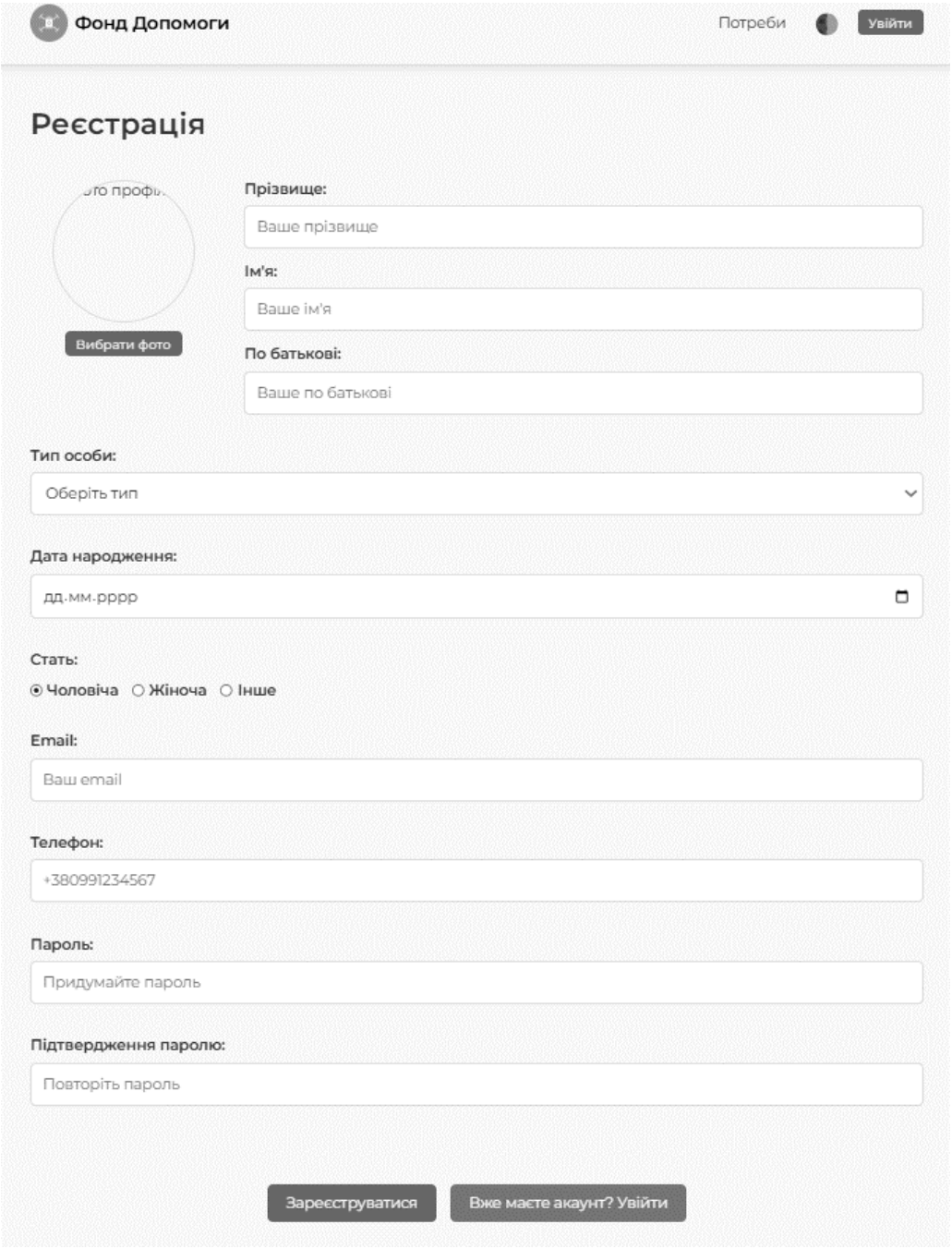


Рисунок 3.3 — Сторінка реєстрації особистого кабінету

На вкладці входу користувач вводить адресу електронної пошти та пароль, які були зазначені під час реєстрації. Передбачено перевірку коректності введених даних та обробку повідомлень про помилки (наприклад, неправильний логін або

пароль). Після успішної автентифікації користувач отримує доступ до персонального кабінету відповідно до своєї ролі. Вікно входу показано на рисунку 3.4.

Рисунок 3.4 — Вікно входу до особистого кабінету

Вкладка реєстрації призначена для створення нового облікового запису. У формі реєстрації користувач заповнює такі поля: повне ім’я, адреса електронної пошти, пароль та підтвердження пароля. Передбачена базова валідація введених даних, у тому числі перевірка унікальності електронної адреси, відповідності пароля вимогам безпеки та співпадіння пароля з його підтвердженням. У разі успішної реєстрації користувач автоматично перенаправляється до сторінки авторизації.

Ця сторінка є універсальною для всіх категорій користувачів, а доступ до функціоналу системи формується на основі ролі, що присвоюється користувачу після входу до системи.

Сторінка особистого кабінету користувача є основним елементом взаємодії зареєстрованого користувача з системою. Вона відображається після успішного входу до системи та надає доступ до персонального функціоналу. Структуру сторінки подано на рисунку 3.5.

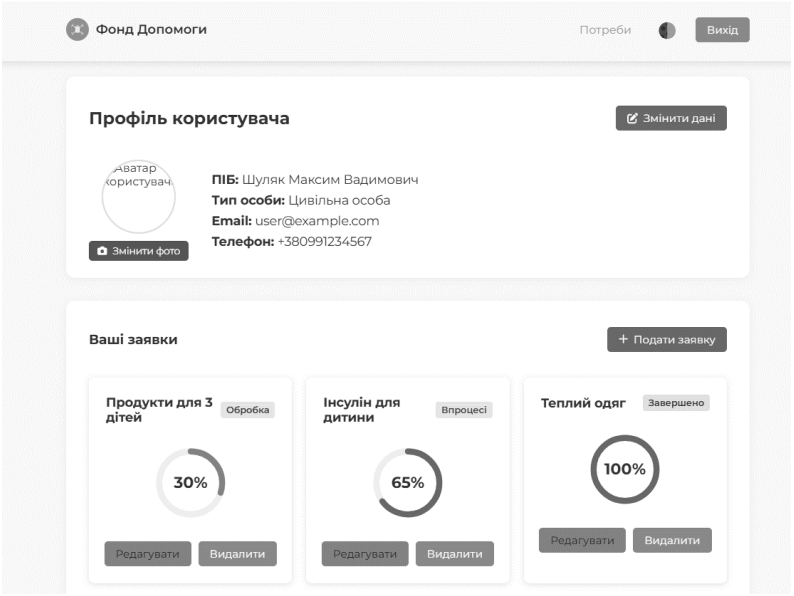


Рисунок 3.5 — Сторінка особистого кабінету

На сторінці реалізовано декілька логічних блоків. Першим є блок профілю, де відображається ім’я користувача, електронна пошта та фото профілю. Передбачено можливість змінити аватар, редагувати особисту інформацію (наприклад, змінити ім’я або опис), а також оновити пароль.

Нижче розташовано блок з історією раніше поданих заявок. У цьому розділі користувач може переглянути перелік усіх своїх запитів, ознайомитися з їхнім статусом (наприклад, «Обробка», «В процесі», «Завершено»), а також, за потреби, редагувати або видалити подану заявку. Кожен запис в історії є інтерактивним і містить короткий опис звернення. Блок з історією запитів зображений на рисунку 3.5.

Другим основним елементом є функціональний блок для створення нової заявки на допомогу. Він містить кнопку для переходу до відповідної форми, розміщеної на окремій сторінці. Це дозволяє користувачу оперативно подати звернення до фонду із зазначенням типу допомоги та описом ситуації.

Сторінка створення заявки на отримання гуманітарної допомоги реалізована у вигляді модального вікна, яке відкривається поверх основного інтерфейсу після авторизації користувача. Дана форма дозволяє швидко подати запит на отримання допомоги, вказавши всю необхідну інформацію для подальшого розгляду з боку

адміністрації чи модераторів платформи. Загальний вигляд форми наведено на рисунку 3.6.

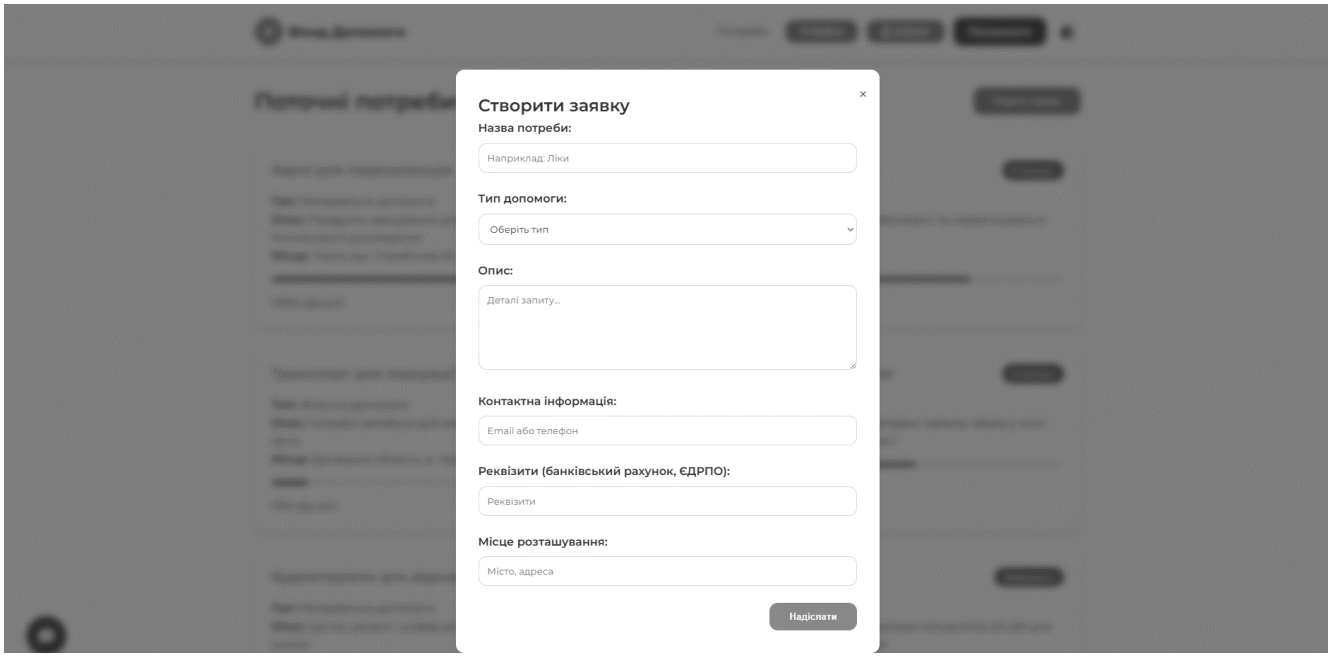


Рисунок 3.6 — Форма створення запиту

Сторінка створення заявки на отримання гуманітарної допомоги реалізована у вигляді модального вікна, яке відкривається поверх основного інтерфейсу після авторизації користувача. Дана форма дозволяє швидко подати запит на отримання допомоги, вказавши всю необхідну інформацію для подальшого розгляду з боку адміністрації чи модераторів платформи. Загальний вигляд форми наведено на рисунку 3.6.

Форма складається з кількох логічно структурованих полів, які користувач заповнює вручну. Першим елементом є поле «Назва потреби», де заявник коротко зазначає суть запиту, наприклад: "Ліки", "Продукти", "Засоби гігієни". Далі користувач має змогу вибрати тип допомоги зі списку, який формується динамічно відповідно до категорій, закладених у системі. Це забезпечує уніфіковану класифікацію запитів і полегшує подальшу обробку.

У полі «Опис» надається можливість деталізувати запит: зазначити обставини, мету допомоги, вказати для кого саме вона призначена (наприклад, для внутрішньо переміщених осіб, медичного закладу, багатодітної сім’ї тощо). Це

текстове поле є рекомендованим до заповнення, оскільки надає додатковий контекст для ухвалення рішення модератором.

Окрема група полів відведена для контактної інформації. Користувач вказує електронну пошту або номер телефону, що дає можливість адміністратору оперативно зв'язатися для уточнення деталей. Також передбачено введення реквізитів — банківського рахунку або коду ЄДРПОУ, що особливо актуально у випадку збору коштів для офіційних організацій або юридичних осіб.

Форма завершується полем «Місце розташування», де заявник вказує населений пункт або конкретну адресу, що дозволяє модераторам врахувати географічну прив’язку запиту при формуванні пріоритетів.

Після заповнення всіх полів користувач натискає кнопку «Надіслати», після чого дані автоматично передаються на сервер та зберігаються в базі даних. Створена заявка отримує статус «очікує підтвердження» і відображається у внутрішній системі для подальшої модерації. Такий підхід дозволяє забезпечити ефективну обробку запитів, структурувати їх та мінімізувати кількість неповних або не валідних звернень.

Також створено сторінку для здійснення донату. Вона є однією з ключових частин інформаційної системи, оскільки вона забезпечує користувачам можливість фінансово підтримати діяльність благодійного фонду. Вона реалізована у вигляді зручної веб-форми з інтуїтивним інтерфейсом, що дозволяє обрати суму внеску, спосіб оплати, переглянути мету збору коштів і реквізити для переказу.

Ліва частина сторінки призначена для вибору способу та суми платежу. Користувач може обрати одну з фіксованих сум (100, 200, 500, 1000, 2000, 5000 грн) або ввести власне значення в окремому полі. Це дає змогу як швидко зробити стандартний внесок, так і налаштувати його індивідуально під фінансові можливості користувача.

Далі користувач обирає зручний спосіб оплати: банківська картка, Google Pay, Apple Pay, Приват24 або Monobank. У разі вибору оплати карткою необхідно заповнити стандартні поля: номер картки, термін дії, CVV-код та ім’я власника. Додатково можна активувати опцію щомісячного автоматичного платежу, що

дозволяє підтримувати фонд на постійній основі. Структуру сторінки представлено на рисунку 3.7.

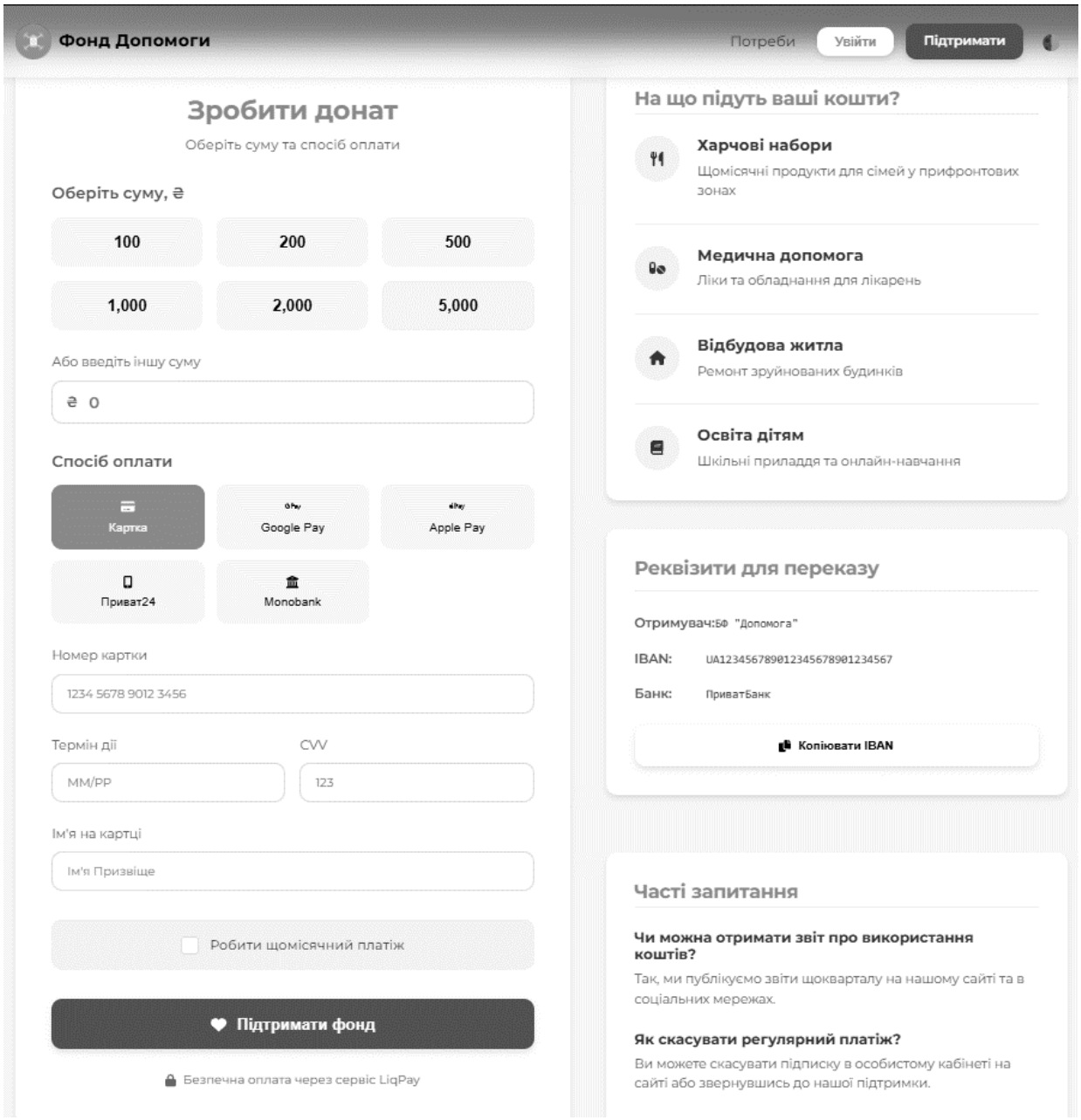


Рисунок 3.7 — Сторінка підтримки фонду

У правій частині інтерфейсу розташовані інформаційні блоки, що доповнюють основну форму донату.

Блок «На що підуть ваші кошти?» містить перелік основних напрямів, за якими розподіляються зібрані внески. До них належать надання харчових наборів, медична допомога, ремонт житла та забезпечення освітніх потреб дітей. Така

деталізація сприяє підвищенню прозорості платформи та зміцнює довіру користувачів.

Блок «Реквізити для переказу» включає інформацію про отримувача, IBAN-номер та назву банку. Для зручності користувачів реалізована функція копіювання реквізитів одним натисканням, що полегшує здійснення переказу через сторонні фінансові сервіси.

Інформаційний розділ «Часті запитання» (FAQ) містить відповіді на найпоширеніші запити користувачів. Зокрема, надається роз’яснення щодо можливості отримання фінансових звітів про використання пожертв та порядку скасування регулярних платежів.

У нижній частині сторінки розміщено кнопку «Підтримати фонд», яка запускає процес підтвердження транзакції. Весь процес оплати захищено, що підкреслюється підписом: «Безпечна оплата через сервіс LiqPay».

Таким чином, сторінка донату виконує не лише фінансову функцію, а й інформує, мотивує та забезпечує прозору взаємодію між фондом і відвідувачами сайту. Вона орієнтована як на одноразові, так і на регулярні внески, і повністю адаптована під вимоги зручності та безпеки користувача.

Також була створена сторінка «Поточні потреби» призначена для відображення усіх активних та архівних заявок на допомогу, поданих користувачами платформи. Її інтерфейс реалізований у вигляді сітки карток, кожна з яких відображає окремий запит на підтримку із зазначенням основної інформації, статусу та рівня виконання. Такий підхід забезпечує зручність візуального сприйняття та швидку навігацію між потребами. Загальний вигляд сторінки представлено на рисунку 3.8.

Кожна картка містить такі структурні елементи: назву запиту, його тип, опис, місце реалізації потреби, індикатор заповнення (прогрес-бар) та статус. Наприклад, запити можуть стосуватися медичної, матеріальної, фізичної допомоги або волонтерської підтримки. Тип допомоги виводиться як текстове поле, що дає можливість швидко ідентифікувати категорію заявки.

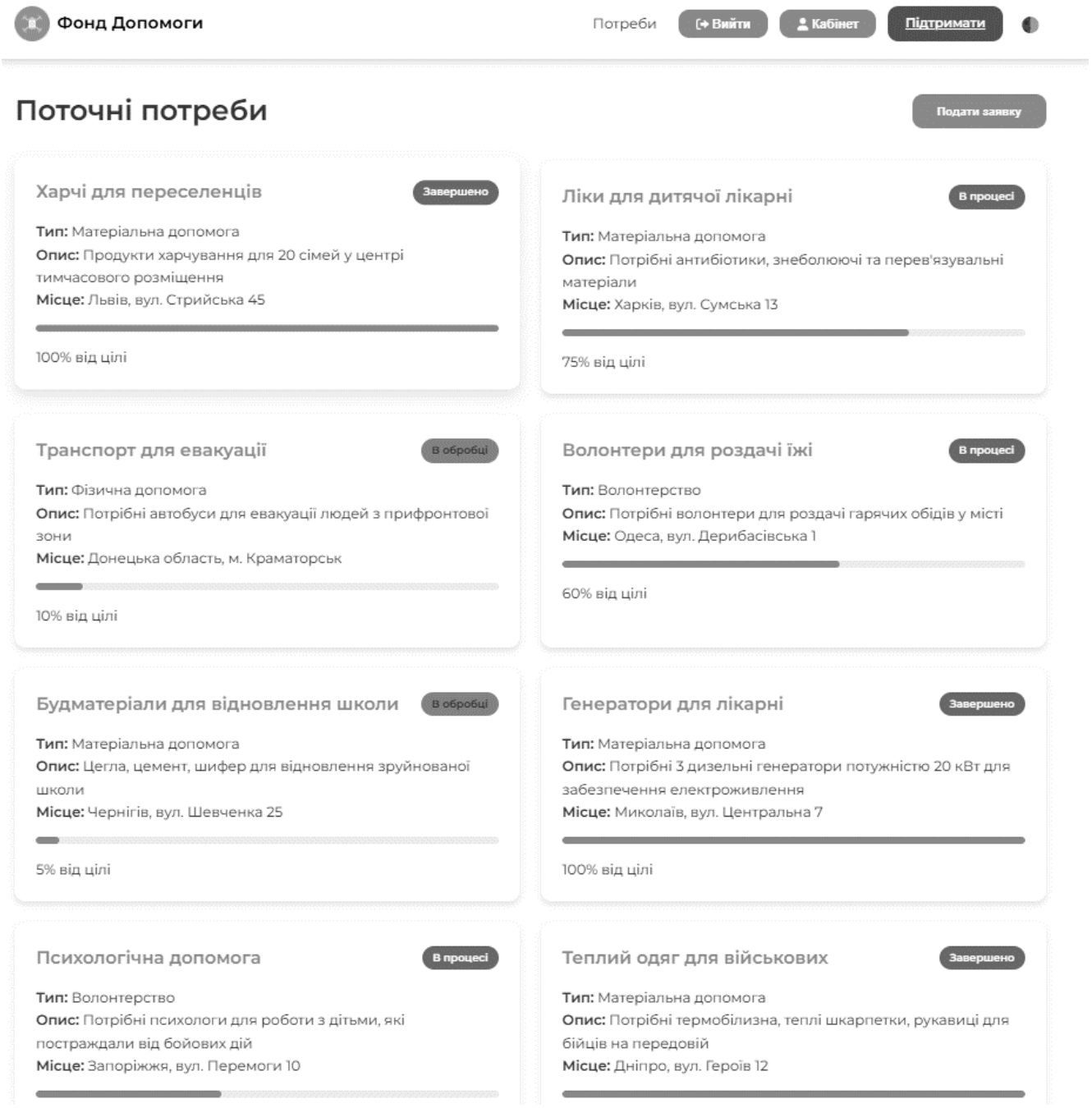


Рисунок 3.8 — Сторінка потреб

Кожна картка містить такі структурні елементи: назву запиту, його тип, опис, місце реалізації потреби, індикатор заповнення (прогрес-бар) та статус. Наприклад, запити можуть стосуватися медичної, матеріальної, фізичної допомоги або волонтерської підтримки. Тип допомоги виводиться як текстове поле, що дає можливість швидко ідентифікувати категорію заявки.

Поле опису містить стисле формулювання суті звернення. Це дозволяє

користувачеві зрозуміти конкретну потребу без переходу на окрему сторінку. Нижче зазначається географічне розташування: місто, область, конкретна адреса — що важливо для координації логістики та розподілу ресурсів.

Кожна заявка має відповідний статус, який відображається кольоровим маркером: "В процесі", "В обробці", "Завершено". Це дозволяє відвідувачам одразу бачити, які запити ще потребують підтримки, а які вже були реалізовані. Додатково, візуальний прогрес виконання запиту представлено у вигляді горизонтального індикатора, який демонструє відсоткове співвідношення зібраних або виконаних ресурсів до встановленої мети.

У правому верхньому куті сторінки розташована кнопка «Подати заявку», що забезпечує перехід до форми створення нового звернення. Цей функціональний елемент дозволяє швидко ініціювати нову потребу без необхідності переходу в окремі розділи.

Таким чином, сторінка «Поточні потреби» виконує функцію публічного відображення актуальних звернень, сприяє прозорості діяльності платформи та полегшує процес збору допомоги завдяки зрозумілому і функціонально адаптивному інтерфейсу.

Також у межах реалізації інформаційної системи була впроваджена повнофункціональна адміністративна панель, що забезпечує централізоване управління контентом, користувачами, модераторами та всією структурою платформи. Її впровадження дозволяє адміністраторам оперативно реагувати на запити, здійснювати контроль за публікаціями, відстежувати активність користувачів та вести внутрішній аудит дій у системі.

Адміністративна панель представлена у вигляді окремого модуля з власним навігаційним меню, яке включає ключові розділи: «Панель», «Оголошення», «Сторінка потреб», «Користувачі», «Модератори», «Логи». Така структура сприяє логічному розділенню функцій та забезпечує зручність користування навіть за значного обсягу даних, що продемонстровано на рисунку 3.9.

На головному екрані відображається зведена панель статистики, що включає кількість нових оголошень, нових користувачів, виконаних потреб та заблокованих

облікових записів. Окрім абсолютних значень, наведена також динаміка змін порівняно з попереднім місяцем. Додатково реалізовані інтерактивні графіки: активність оголошень за останні 30 днів та розподіл потреб за категоріями.

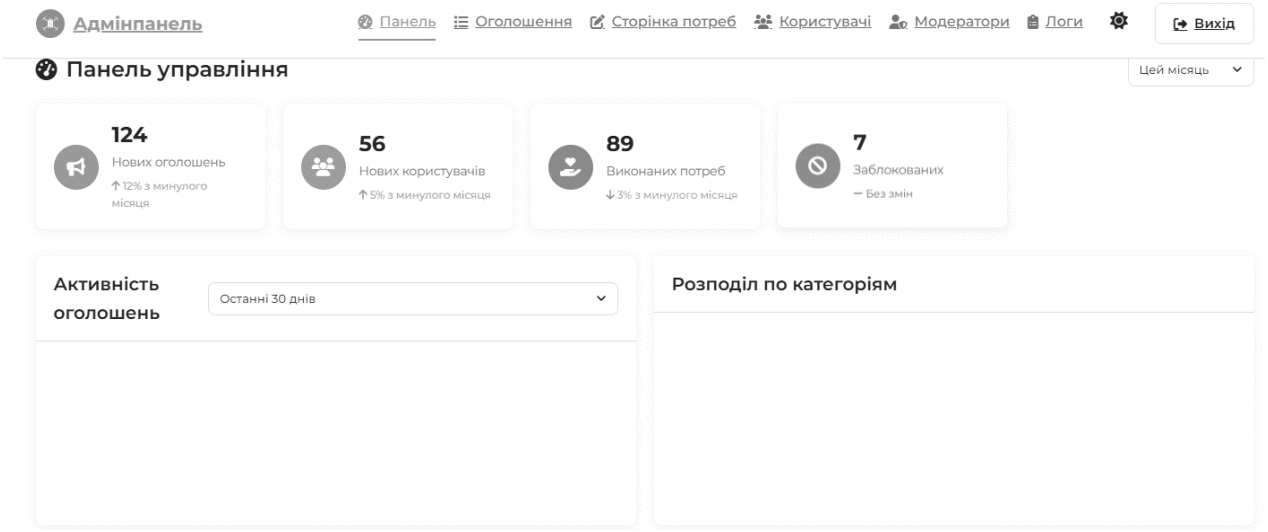


Рисунок 3.9 — Головна сторінка адміністративної панелі інформаційної системи

На головному екрані відображається зведена панель статистики, що включає кількість нових оголошень, нових користувачів, виконаних потреб та заблокованих облікових записів. Окрім абсолютних значень, наведена також динаміка змін порівняно з попереднім місяцем. Додатково реалізовані інтерактивні графіки: активність оголошень за останні 30 днів та розподіл потреб за категоріями.

У розділі «Оголошення» адміністратор має змогу здійснювати повноцінне управління запитами на допомогу. Кожне оголошення відображається у вигляді інформативної картки, яка містить ім’я автора, дату, тип потреби, опис і поточний статус модерації. Для взаємодії з оголошенням передбачено кнопки: «Опублікувати», «Відхилити», «Редагувати», «Архівувати», «Переглянути» (рис. 3.10).

Редагування сторінки потреб дозволяє адміністрації змінювати назви, описи, пріоритети та статуси категорій, які відображаються на публічному інтерфейсі сайту. Наприклад, доцільно тимчасово приховати неактуальну категорію або

підвищити пріоритет для нагальних потреб. Такий інтерфейс показано на рисунку 3.11.

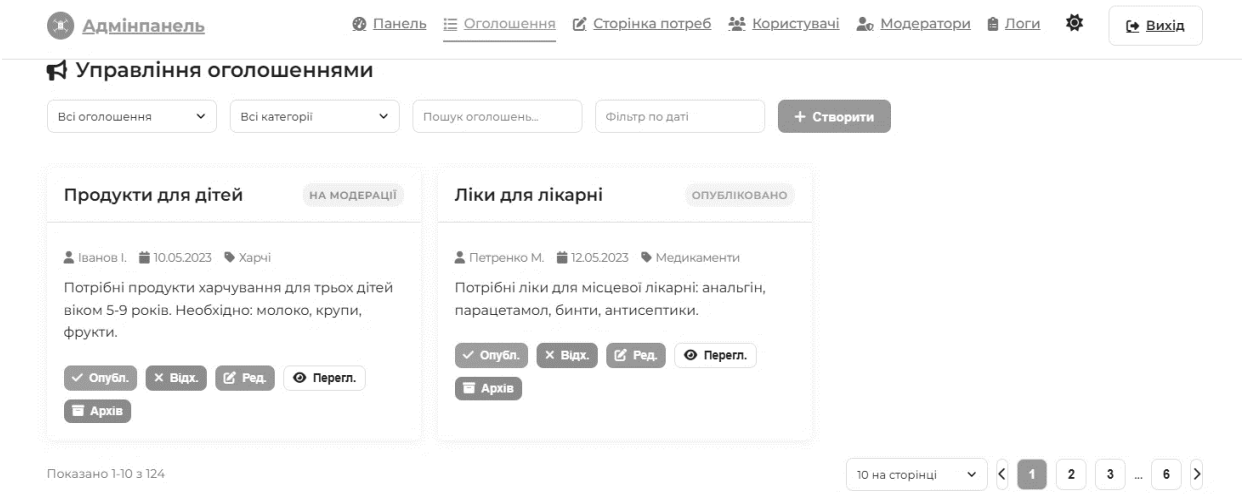


Рисунок 3.10 — Інтерфейс управління оголошеннями користувачів

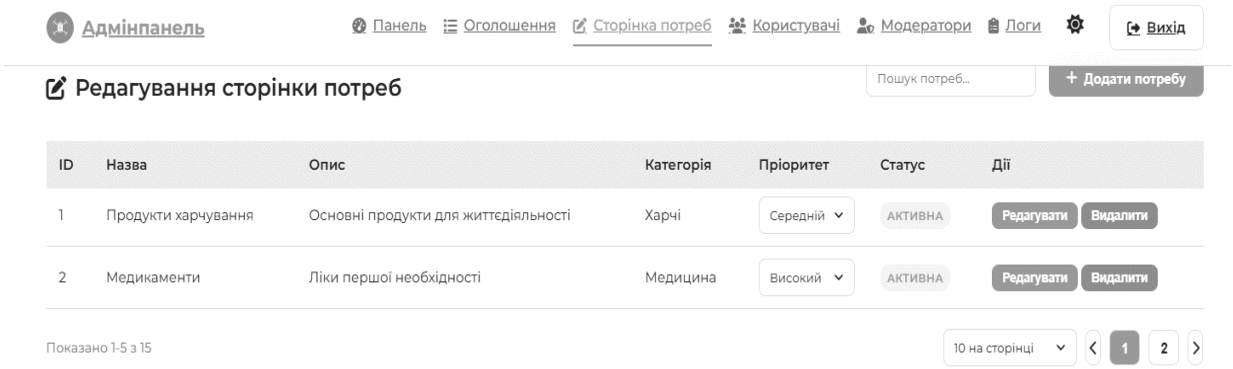


Рисунок 3.11 — Сторінка редагування категорій потреб

У розділі користувачів представлена таблиця всіх зареєстрованих користувачів системи, в якій відображаються такі ключові дані, як унікальний ідентифікатор, email-адреса, номер мобільного телефону, дата реєстрації в системі та поточний статус користувацького акаунта. Адміністратор має можливість у ручному режимі змінювати статус обраного користувача — зокрема, тимчасово заблокувати або розблокувати його доступ, а також призначити користувача модератором із відповідними правами доступу. Крім того, передбачена функція створення резервної копії таблиці, яка містить усі наявні дані користувачів на

збереження. Візуальне представлення цього розділу проілюстровано на рисунку 3.12.

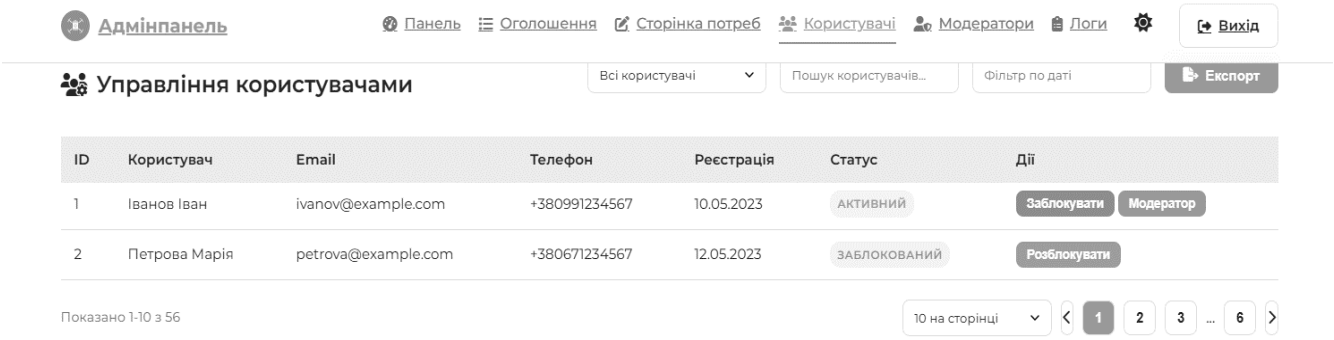


Рисунок 3.12 — Список зареєстрованих користувачів з можливістю керування

Управління модераторами дозволяє адміністрації призначати або відкликати відповідні повноваження. Можна призначити модератора лише для конкретних функцій (наприклад, лише перевірка користувачів або контенту), або надати повний доступ. Процес призначення здійснюється за допомогою спеціальної форми зі списком користувачів та випадаючим меню рівнів доступу, як показано на рисунку 3.13.

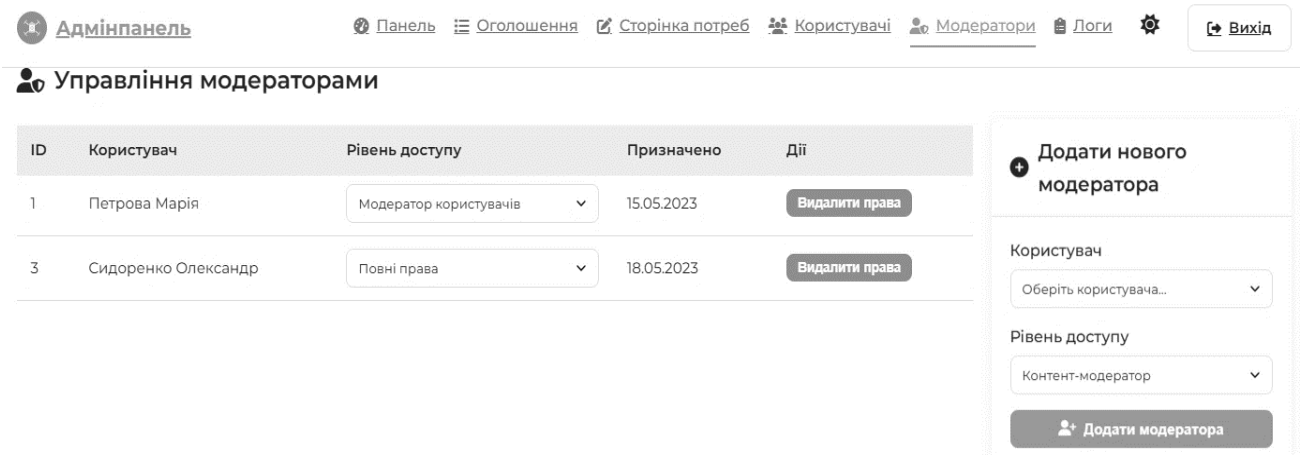


Рисунок 3.13 — Призначення ролі модератора через адміністративну панель

Останній розділ — «Журнал подій» — є важливим компонентом системи моніторингу. У ньому фіксуються всі ключові дії, виконані адміністраторами або системними модулями: публікація оголошень, блокування користувачів, резервне

копіювання тощо. Кожен запис містить дату, виконавця, об’єкт дії та короткий опис події. Вигляд цього журналу подано на рисунку 3.14.

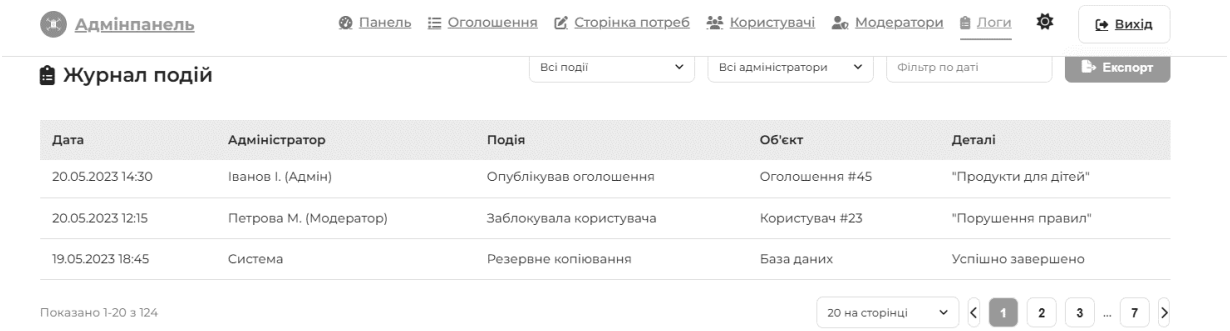


Рисунок 3.14 — Журнал подій системи з відстеженням дій адміністратора

Отже, впровадження адміністративної панелі значно розширює функціональні можливості системи, забезпечує гнучке управління вмістом, оперативну модерацію та аналітичне спостереження за всіма аспектами функціонування платформи.

3.3 Розробка бази даних і системи зберігання та захисту даних

Проєктування системи зберігання даних є критично важливим етапом створення інформаційної системи обліку гуманітарної допомоги, оскільки від її структури та рівня захисту залежить ефективність функціонування платформи в цілому. Для розробки даної системи було обрано документоорієнтовану базу даних MongoDB, яка дозволяє зберігати інформацію у вигляді гнучких JSON-подібних документів. Такий підхід забезпечує масштабованість, високу швидкість обробки запитів, а також зручність у моделюванні складних структур, що характерні для гуманітарної сфери. Структура колекцій у базі даних демонструється на рисунку 3.15.

Побудова колекцій users та aidrequests в базі даних MongoDB реалізована з дотриманням структурних зв’язків, які наочно представлені у фізичній ER-моделі (див. додаток В). У цій моделі відображено типи даних, обов’язковість полів та відношення між сутностями.

Побудова колекцій users та aidrequests в базі даних MongoDB реалізована з дотриманням структурних зв’язків, які наочно представлені у фізичній ER-моделі (див. додаток В). У цій моделі відображено типи даних, обов’язковість полів та відношення між сутностями.

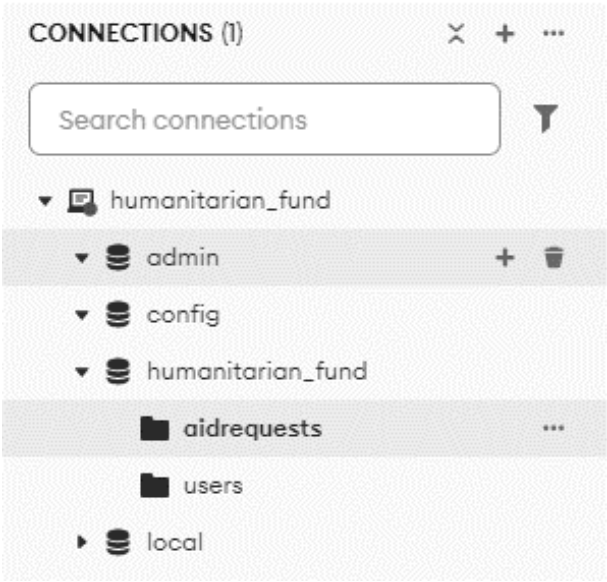


Рисунок 3.15 — Структура колекцій бази даних MongoDB у MongoDB Compass

Побудова колекцій users та aidrequests в базі даних MongoDB реалізована з дотриманням структурних зв’язків, які наочно представлені у фізичній ER-моделі (див. додаток В). У цій моделі відображено типи даних, обов’язковість полів та відношення між сутностями.

У системі реалізовано дві основні колекції: users, яка відповідає за зберігання даних користувачів, та aidrequests, що містить інформацію про подані заявки. Колекція користувачів включає такі поля, як ім’я, прізвище, адреса електронної пошти, номер телефону, роль користувача (адміністратор, модератор або звичайний користувач), статус блокування облікового запису та дата створення. Як приклад, у записі також присутній хешований пароль, збережений із використанням bcrypt, а також масив токенів авторизації, що зображено на рисунку 3.16.

Структура колекції aidrequests містить поля, які описують суть кожної гуманітарної потреби: назва, тип допомоги (матеріальна, фізична, волонтерська

тощо), короткий опис, контактна інформація, реквізити, місце розташування, статус обробки заявки, відсоток виконання та час створення.

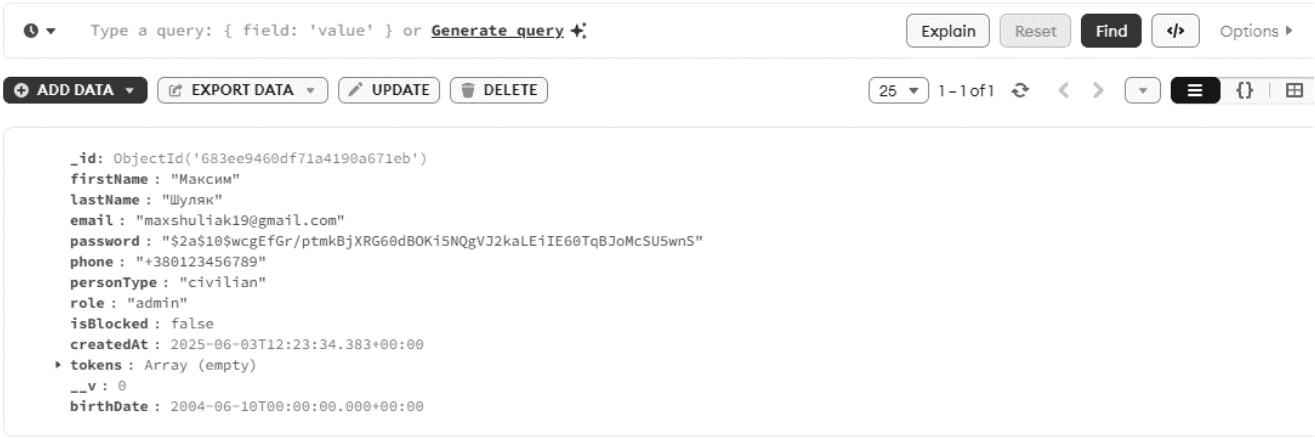


Рисунок 3.16 — Приклад документа користувача в колекції users

Структура колекції aidrequests містить поля, які описують суть кожної гуманітарної потреби: назва, тип допомоги (матеріальна, фізична, волонтерська тощо), короткий опис, контактна інформація, реквізити, місце розташування, статус обробки заявки, відсоток виконання та час створення. Кожна заявка також прив’язана до облікового запису автора через унікальний ідентифікатор. Повну структуру документа заявки наведено на рисунку 3.17.

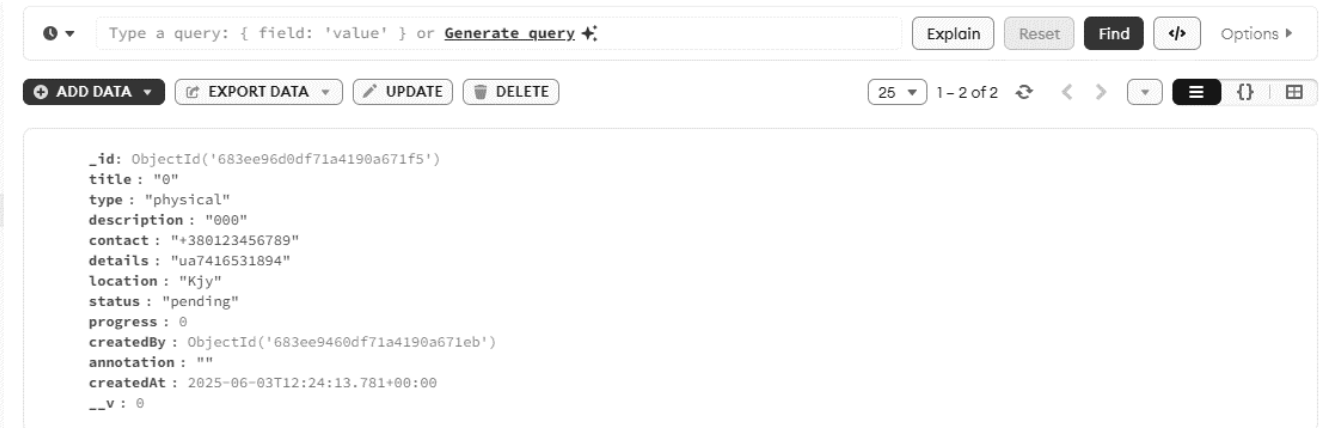


Рисунок 3.17 — Приклад документа користувача в колекції aidrequests

Для зручності конфігурації та запуску серверної частини використовується файл server.js, у якому відбувається ініціалізація необхідних модулів: CORS, body-parser, маршрутизація, підключення до бази даних через Mongoose, а також

глобальна обробка помилок. Змінні середовища, включно з URI доступу до бази, ключем підпису JWT-токенів та іншими параметрами, винесено у файл .env. Його використання забезпечує розмежування конфіденційної інформації та гнучкість при зміні середовища розгортання. Реалізації описаного алгоритму наведена у лістингу 3.1.

Лістинг 3.1 – Ініціалізація серверного застосунку Node.js

```
require('dotenv').config();
const express = require('express');
const cors = require('cors');
const connectDB = require('./config/db');
const authRoutes = require('./routes/authRoutes');
const aidRoutes = require('./routes/aidRoutes');
const requestRoutes = require('./routes/requestRoutes');
const adminRoutes = require('./routes/adminRoutes');
const app = express();
// Middleware
app.use(express.json());
app.use(cors({
   origin: ['http://localhost:5500', 'http://127.0.0.1:5500'],
   methods: ['GET', 'POST', 'PUT', 'DELETE', 'PATCH'],
   credentials: true,
   allowedHeaders: ['Content-Type', 'Authorization']
}));
app.use('/uploads', express.static('public/uploads'));
// Підключення до MongoDB
connectDB();
// Маршрути
app.use('/api/auth', authRoutes);
app.use('/api/requests', aidRoutes);
app.use('/api/requests', requestRoutes);
app.use('/api/admin', adminRoutes);
const PORT = process.env.PORT || 5000;
app.listen(PORT, () => console.log(`Сервер працює на порті ${PORT}`));
```

Для захисту користувацьких даних реалізовано кілька механізмів. Зокрема, усі паролі зберігаються у хешованому вигляді за допомогою бібліотеки bcrypt. Вхід до системи супроводжується створенням JWT-токена, який додається до заголовків подальших HTTP-запитів і використовується для перевірки автентичності. Перевірка автентичності реалізована у вигляді middleware-функції auth.js, яка

забезпечує допуск до захищених маршрутів лише для авторизованих користувачів. Додатково, за допомогою middleware checkRole.js, у системі реалізовано контроль доступу до окремих ділянок функціоналу залежно від ролі користувача. Частину коду, що реалізує перевірку автентичності, наведено у лістингу 3.2.

Лістинг 3.2 – Middleware-функція перевірки користувача за JWT-токеном

```
const jwt = require('jsonwebtoken');
module.exports = async (req, res, next) => {
  try {
    const token = req.headers.authorization?.split(' ')[1];
    if (!token) {
      return res.status(401).json({ message: 'Неавторизований доступ' });
    }
    const decoded = jwt.verify(token, process.env.JWT_SECRET);
    req.userId = decoded.userId;
    req.userRole = decoded.role;
    next();
  } catch (err) {
    res.status(401).json({ message: 'Неавторизований доступ' });}}
```

Також у поточній версії системи всі дані про подані заявки зберігаються виключно у вигляді текстових записів у базі даних. Для кожної заявки формується документ з такими полями, як назва, тип допомоги, опис, контактна інформація, статус, дата створення та ідентифікатор автора. Модель AidRequest у Mongoose визначає обов’язкові та опціональні поля, встановлює типи даних і забезпечує валідацію.

Таким чином, реалізована система зберігання інформації базується на сучасній технології MongoDB, доповненій інструментами для керування файлами, резервного копіювання та гнучкої структури даних. Завдяки використанню middleware, ORM-рівня Mongoose і відокремленого файлового сховища забезпечується висока ефективність, безпека та надійність функціонування всієї інформаційної системи.

4 ТЕСТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

4.1 Тестування серверної частини

У процесі розробки інформаційної системи обліку гуманітарної допомоги важливу увагу було приділено тестуванню серверної частини, оскільки саме вона відповідає за збереження, обробку та захист критичних даних користувачів і заявок. З огляду на специфіку дипломного проєкту, тестування проводилося вручну через взаємодію з клієнтською частиною в середовищі браузера. Це дозволило безпосередньо перевірити працездатність основних REST-маршрутів та логіку автентифікації та авторизації.

На першому етапі перевірялася функціональність реєстрації користувачів. Після заповнення реєстраційної форми на клієнтській стороні, сервер обробляв запит, хешував пароль за допомогою бібліотеки bcrypt, створював новий запис у базі даних MongoDB та повертав статус успішної реєстрації. У випадку повторного використання email-адреси сервер коректно повертав повідомлення про помилку (рис. 4.1).

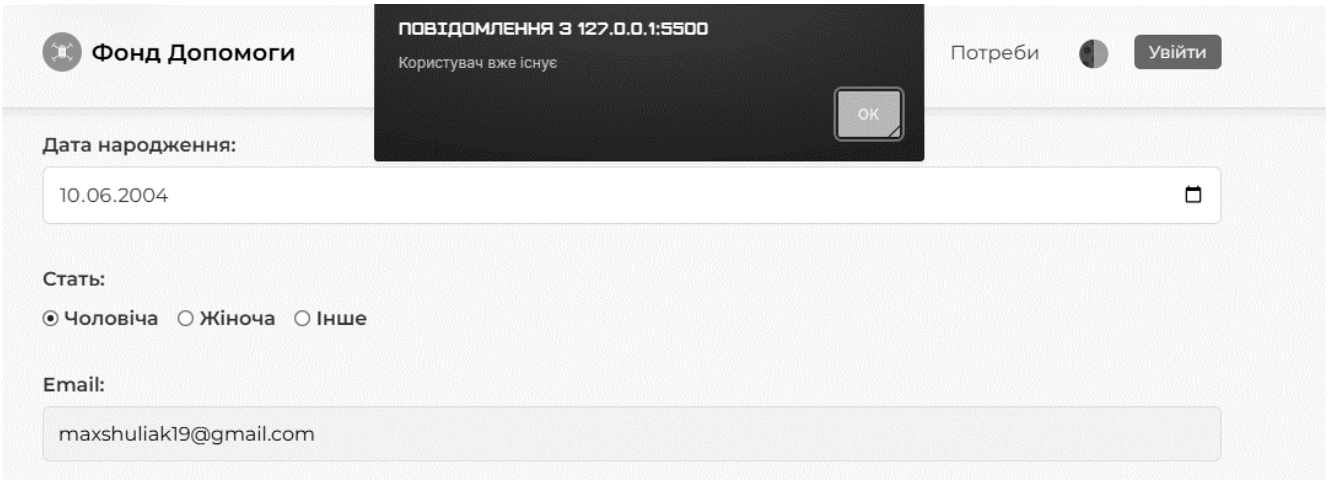


Рисунок 4.1 — Повідомлення про успішну або неуспішну реєстрацію користувача

Далі тестувалася процедура авторизації. Після введення логіна і пароля, сервер формував JWT-токен, який зберігався у локальному сховищі клієнта та додавався до заголовків наступних запитів. У разі введення некоректних даних користувач отримував відповідне повідомлення про помилку

автентифікації (рис. 4.2). Успішна авторизація відкривала доступ до персоналізованих функцій, включаючи створення заявок.

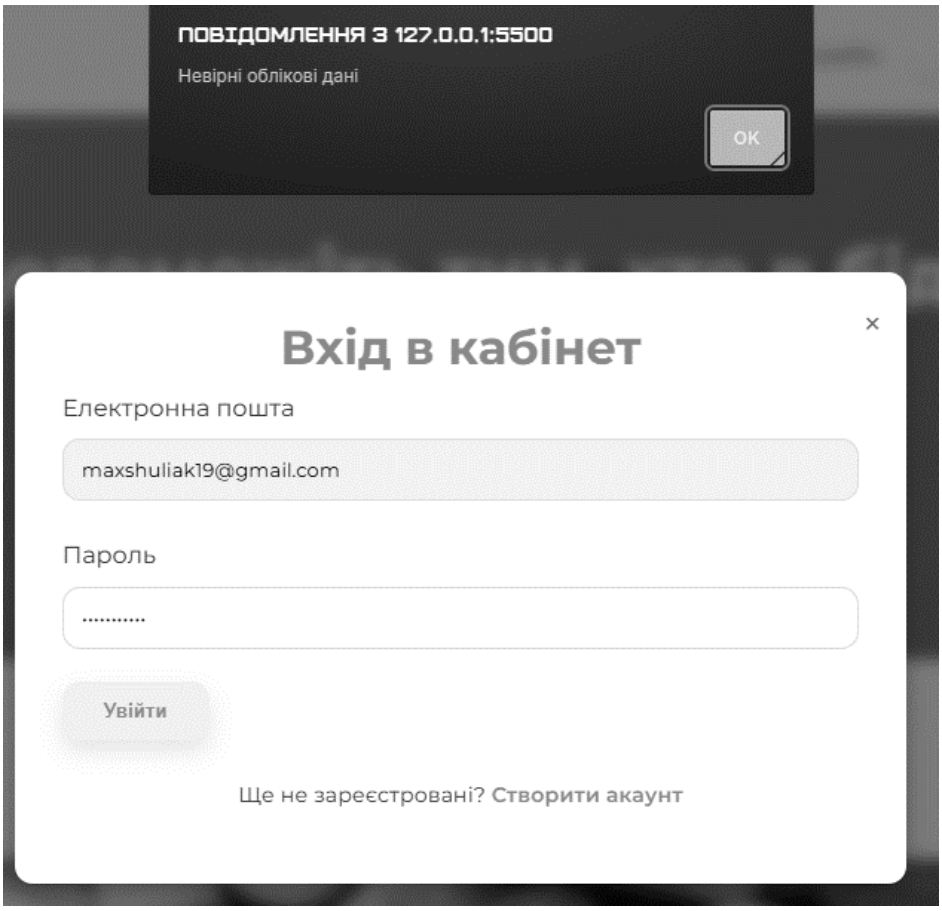


Рисунок 4.2 — Відповідь сервера під час авторизації користувача

Особливу увагу приділено перевірці реалізації рольового доступу. У системі передбачено три рівні прав: користувач, модератор і адміністратор. Кожна роль має доступ до власного набору функцій, що реалізовано на сервері через middleware-функцію checkRole. Під час ручного тестування проводилися спроби виконати дії, заборонені для певної ролі, саме була спроба викона вхід до адмінпанелі користувачем у якого немає на те прав і система успішно блокувала такі запити з відповідним статусом відповіді (рис. 4.3).

На завершальному етапі перевірялась логіка обробки заявок. Після заповнення форми та натискання кнопки «Створити запит», дані надсилались методом POST до відповідного маршруту, де здійснювалась їх валідація, збереження у базі та формування підтвердження.

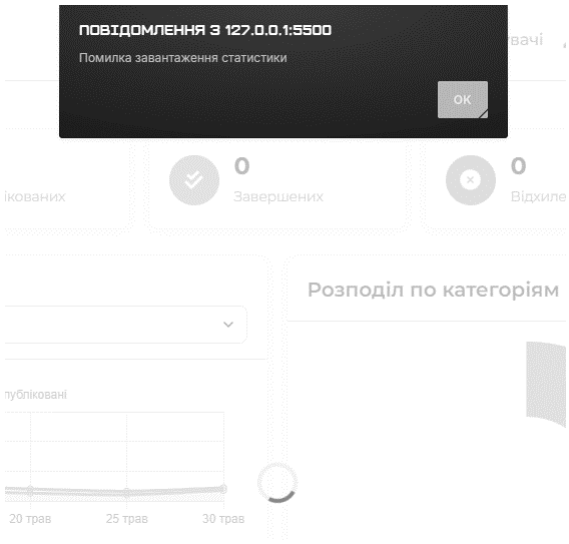


Рисунок 4.3 — Реакція системи на порушення рівня доступу

На завершальному етапі перевірялась логіка обробки заявок. Після заповнення форми та натискання кнопки «Створити запит», дані надсилались методом POST до відповідного маршруту, де здійснювалась їх валідація, збереження у базі та формування підтвердження. Серверна частина системи коректно обробляла нові запити, передаючи їх на розгляд модератору. Після ухвалення або відхилення модератором відповідного рішення, статус заявки автоматично оновлювався та відображався в особистому кабінеті користувача, який її подав, як показано на рисунку 4.4.

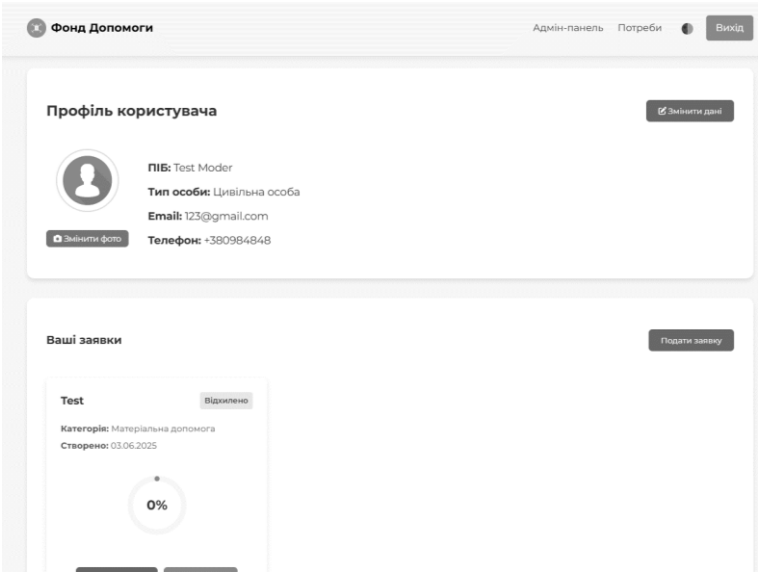


Рисунок 4.4 — Зміна статусу заявки після відхилення модератором

Таким чином, ручне тестування показало стабільну роботу ключових компонентів серверної частини системи, підтвердивши правильність реалізації авторизації, реєстрації, обробки заявок та рольового доступу. Отримані результати засвідчують, що створена система готова до подальшої експлуатації з можливістю масштабування.

4.2 Тестування клієнтської частини

Після завершення розробки клієнтської частини вебзастосунку було проведено всебічне функціональне тестування інтерфейсів з метою верифікації їх працездатності, коректного відображення елементів та належної взаємодії з серверною логікою. Основною метою тестування виступало підтвердження відповідності реалізованого функціоналу раніше сформованим вимогам та забезпечення зручного користувацького досвіду.

На початковому етапі було перевірено відображення головної сторінки, її адаптивність та коректність відображення блоків незалежно від розміру екрана. Особливу увагу приділено правильності роботи банерного слайдера, відповідності стилізації елементів (кнопки, заголовки, навігаційне меню), а також наявності швидкого доступу до основних функцій системи. Сторінка завантажується повністю, без артефактів або збоїв, навігаційне меню працює згідно очікуваної логіки.

Далі тестувалася форма авторизації та реєстрації, яка є критичною для безпечного доступу користувачів до персоналізованого функціоналу. У ході тестування перевірено валідацію полів, коректну обробку помилкових даних, а також перехід до профілю після успішного входу. У випадку неправильно заповненого або порожнього поля система відображає відповідне повідомлення. Це свідчить про реалізацію базових засобів захисту та попередження помилок користувача.

Успішно протестовано сторінку особистого кабінету, яка надає доступ до редагування особистих даних, додавання аватару та перегляду історії запитів. Особливу увагу приділено перевірці коректності завантаження зображення та

збереження змін. Було підтверджено, що після оновлення профільні дані відображаються миттєво (рис. 4.6). Також перевірено, що елементи інтерфейсу не перекриваються та залишаються доступними на різних розмірах екранів.

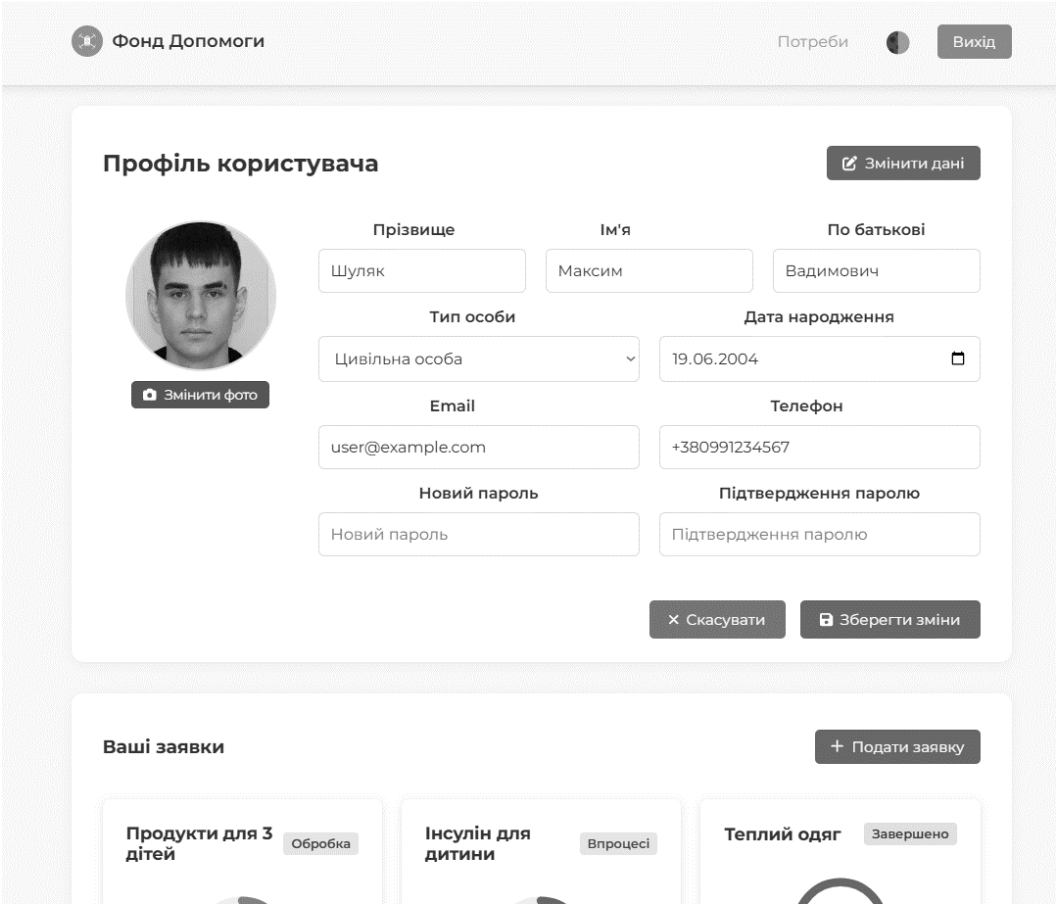


Рисунок 4.6 — Інтерфейс особистого кабінету з полями для редагування профілю

Форма створення заявки на отримання гуманітарної допомоги реалізована у вигляді модального вікна з інтуїтивно зрозумілою структурою. Вона містить послідовні поля для введення ключової інформації: назви потреби, типу допомоги, опису, контактних даних, реквізитів (банківський рахунок або код ЄДРПОУ), а також місця розташування. У процесі тестування перевірено правильність роботи валідації полів, зокрема, заповнення обов’язкових елементів та обробку випадків, коли деякі необов’язкові поля залишаються порожніми. Було підтверджено, що при правильному заповненні форми система успішно обробляє запит і відображає відповідне повідомлення про надсилання. Надіслані заявки коректно фіксуються у

системі та зберігаються в особистому кабінеті користувача у хронологічному порядку.

Окреме тестування було проведено для інтерфейсів адміністратора, що мають додаткові можливості: перегляд усіх звернень, їх редагування та видалення. Було перевірено коректність відображення запитів, реакцію на взаємодію з кнопками «Редагувати» та «Видалити», а також повідомлення, що з’являються після відповідних дій. Як зображено на рисунку 4.7, інтерфейс адміністратора дозволяє миттєво змінювати статуси заявок та керувати вмістом бази даних.

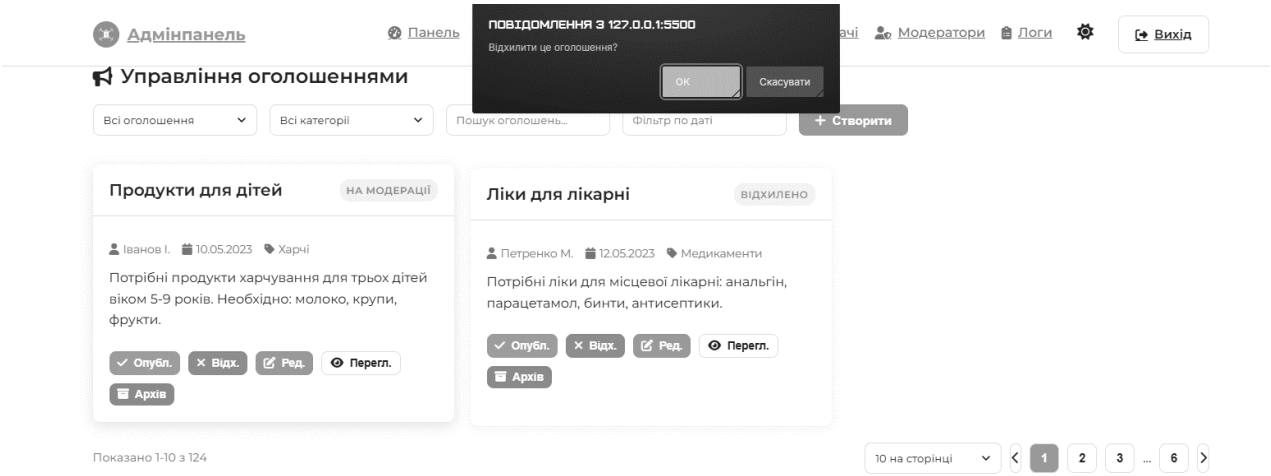


Рисунок 4.7 — Панель адміністратора з функціями редагування запитів

Усі дії клієнта, пов’язані з переходом між сторінками, перевірено на відповідність маршрутизації, що базується на структурі SPA-додатку. Посилання функціонують без перезавантаження сторінки, а зміст динамічно змінюється.

Загалом результати тестування клієнтської частини підтверджують її стабільність, логічну побудову та доступність для користувачів з різним технічним досвідом. Протестовані функції відповідають функціональним вимогам, а взаємодія між клієнтом і сервером є стабільною та надійною.

ВИСНОВКИ

У бакалаврській кваліфікаційній роботі було розроблено та реалізовано інформаційну систему обліку гуманітарної допомоги, яка вирішує актуальні проблеми обліку, обробки та контролю за розподілом ресурсів у благодійних організаціях. Система дозволяє автоматизувати ключові процеси, що до цього часто виконувались вручну або за допомогою неструктурованих цифрових засобів.

На основі аналізу предметної області та вивчення існуючих рішень було сформульовано функціональні вимоги до системи. У роботі обґрунтовано вибір архітектури, мов програмування, засобів для розробки веб-інтерфейсу та інструментів для реалізації бази даних. В основу системи покладено клієнт-серверну модель з використанням HTML, CSS та JavaScript для створення інтерфейсу, а також MongoDB як документоорієнтованої системи управління базами даних. Такий вибір дозволив забезпечити доступність, масштабованість та адаптивність системи до майбутніх змін.

У процесі реалізації створено веб-додаток з можливістю реєстрації користувачів, подання заявок, їх редагування та модерації адміністраторами. Передбачено розмежування ролей із відповідними правами доступу, що підвищує безпеку та керованість системою. Структура бази даних спроєктована з урахуванням цілісності інформації та забезпечення зручного доступу до актуальних записів.

На завершальному етапі було проведено тестування інформаційної системи, яке підтвердило її працездатність і відповідність поставленим вимогам. Система успішно функціонує в середовищі браузера, не потребує встановлення додаткового програмного забезпечення і була розгорнута на доступних хостингових платформах.

Таким чином, у роботі досягнуто поставлену мету: створено інформаційну систему, яка дозволяє автоматизувати облік гуманітарної допомоги, підвищити прозорість процесів і полегшити діяльність благодійних організацій. Результати роботи мають практичну цінність і можуть бути використані в реальних умовах.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Інформаційна система обліку гуманітарної допомоги / М. В. Шуляк, О. С. Городецька // Матеріали LIV Всеукраїнської науково-технічної конференції факультету інформаційних технологій та комп'ютерної інженерії. – Вінниця, 2025. – 3 с. [Електронний ресурс]. Режим доступу: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2025/author/downloadFile/24037/84641/1

2. Швецов А. А. Проектування інформаційних систем: Навч. посібник. – Харків: ХНУРЕ, 2020. – 292 с.

3. Архітектура інформаційних систем [Електронний ресурс]. – Режим доступу: https://elearning.sumdu.edu.ua/free_content/lectured:de1c9452f2a161439391120eef364dd8ce4d8e5e/20160217112601/170352/index.html

4. Таненбаум Е., Стайн А. С. Розподілені системи. Принципи і парадигми. – 2-ге вид. – К.: УМК, 2017. – 704 с.

5. Garcia-Molina H., Ullman J. D., Widom J. Database Systems: The Complete Book. – 2nd ed. – Pearson, 2008. – 1248 p.

6. Coronel C., Morris S., Rob P. Database Systems: Design, Implementation, and Management. – 13th ed. – Cengage Learning, 2019. – 688 p.

7. MongoDB Manual [Електронний ресурс]. – Режим доступу: https://www.mongodb.com/docs/manual/

8. Duckett J. HTML and CSS: Design and Build Websites. – Wiley, 2011. – 490 p.

9. MDN Web Docs. CSS Reference [Електронний ресурс]. – Режим доступу: https://developer.mozilla.org/en-US/docs/Web/CSS

10. Freeman E., Robson E., Bates B., Sierra K. Head First HTML and CSS: A Learner's Guide to Creating Standards-Based Web Pages. – 2nd ed. – O'Reilly Media, 2012. – 768 p.

11. Фленеган Д. JavaScript: Повний довідник. – 6-е вид. – К.: Діалектика, 2020. – 1184 с.

12. Документація PostgreSQL [Електронний ресурс]. – Режим доступу: https://www.postgresql.org/docs/

13. Visual Studio Code Documentation [Електронний ресурс]. – Режим доступу: https://code.visualstudio.com/docs

14. Бойко О. В. Розробка веб-додатків: Практичний посібник. – Львів: Львівська політехніка, 2021. – 216 с.

15. Order Forecasting System for Vehicles Based on Previous Statistics Requests Svitlana Kyrylashchuk, Oksana Horodetska, Olena Voitsekhovska, Serhii Zakharchenko // Lecture Notes in Data Engineering, Computational Intelligence, and Decision-Making, Volume 1 2024 International Scientific Conference "Intelligent Systems of Decision-Making and Problems of Computational Intelligence", Proceedings, Springer 2024, Pages 116-131 Режим доступу: https://link.springer.com/chapter/10.1007/978-3-031-70959-3_6

16. Руденко О. Г., Морозов С. Ю. Інформаційні системи та технології: навч. посібник. – К.: Центр учбової літератури, 2020. – 328 с.

17. Welling L., Thomson L. PHP and MySQL Web Development. – 5th ed. – Addison-Wesley, 2016. – 1008 p.

18. Mozilla Developer Network. JavaScript Reference [Електронний ресурс]. – Режим доступу: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference

ДОДАТОК А

Технічне завдання

Міністерство освіти та науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп’ютерної інженерії

ЗАТВЕРДЖУЮ
Завідувач кафедри ОТ
д.т.н.проф._____Азаров О. Д.
«21» березня 2025 року

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання бакалаврської кваліфікаційної роботи

“Інформаційна система обліку гуманітарної допомоги”

08-54.БКР.0018.00.000 ТЗ

Керівник роботи к.т.н. доц. каф. ОТ

_____Городецька О.С._____

Студент групи 1КІ–21б

_____Шуляк М.В._____

ВНТУ 2025

1 Підстава для виконання комплексного бакалаврської кваліфікаційної роботи (БКР)

1.1 Актуальність роботи обумовлена потребою у цифровізації процесів обліку гуманітарної допомоги. У зв’язку з ростом кількості надзвичайних ситуацій, пов’язаних з війною, природніми катастрофами та внутрішнім переміщенням населення, ефективне управління запитами та розподілом ресурсів стало надзвичайно важливим. Більшість благодійних та волонтерських організацій веде облік вручну або за допомогою електронних таблиць, що створює ризики втрати даних та ускладнює прозоре звітування.

1.2 Наказ про затвердження теми БКР.

2 Мета БКР і призначення розробки

2.1 Мета роботи — розробка інформаційної системи обліку гуманітарної допомоги з розширеними функціональними можливостями, а саме автоматизацією процесу реєстрації користувачів, подання заявок та ведення обліку наданих ресурсів.

2.2 Призначення розробки — інформаційна система призначена для підтримки волонтерської або благодійної діяльності шляхом створення відкритої платформи, де кожен користувач має можливість створити заявку на допомогу. Таким чином, система виконує роль інформаційного посередника між заявником і громадськістю.

3 Вихідні дані для виконання БКР

3.1 Вивчення особливостей інформаційних систем: аналіз архітектури, функціональності та особливостей інформаційних систем.

3.2 Порівняльний аналіз інформаційних систем у благодійних організаціях: дослідження та оцінка різних аспектів інформаційних систем, що використовуються в благодійних організаціях.

3.3 Створення інформаційної системи шляхом розробки серверної та клієнтської частин: проєктування та реалізація обох компонентів інформаційної системи — серверної та клієнтської.

3.4 Розробка заходів для захисту інформаційних систем: розробка та впровадження заходів забезпечення безпеки інформаційних систем, включаючи захист від несанкціонованого доступу.

3. Випробування серверної та клієнтської частин інформаційної системи: проведення тестування обох компонентів інформаційної системи для перевірки їхньої працездатності, стабільності та безпеки.

4 Вимоги до виконання БКР

Головна вимога — розробити легку масштабовану та захищену інформаційну систему для інформаційної системи обліку гуманітарної допомоги, яка може стати платформою для публікації запитів про допомогу від приватних осіб та інших благодійних фондів.

5 Етапи БКП та очікувані результати

Етапи роботи та очікувані результати приведено в Таблиці А .1.

Таблиця А.1 — Етапи БКП

№ Етапу	Назва етапу	Термін виконання		Очікувані результати
		початок	кінець	
1	Аналіз предметної області.	22.03.25	26.03.25	Вступ, розділ 1
2	Аналіз та обгрунтування аналогів	27.03.25	03.04.25	Розділ 1
3	Аналіз та обґрунтування вибору технологій	04.04.25	09.04.25	Розділ 2
4	Розробка інформаційної системи обліку гуманітарної допомоги	10.04.25	25.04.25	Розділ 3
5	Тестування інформаційної системи	26.04.25	01.05.25	Розділ 4

Продовження таблиці А.1

№ Етапу	Назва етапу	Термін виконання		Очікувані результати
		початок	кінець	
6	Апробація та впровадження результатів дослідження	02.05.25	03.05.25	Виступ
7	Опублікування результатів досліджень	04.05.25	06.05.25	Тези
8	Оформлення пояснювальної записки,графічного матеріалу і презентації та тест на плагіат	07.05.25	10.05.25	Пз, графічний матеріал і презентація
9	Підготовка і підпис супроводжуючих документів, нормоконтроль та тест на плагіат	11.05.25	13.05.25	Оформленні документи

6 Матеріали, що подаються до захисту БКР

До захисту подаються: пояснювальна записка БКР, графічні і ілюстративні матеріали, протокол попереднього захисту БКР на кафедрі, відгук наукового керівника, відгук рецензента, анотації до БКР українською та іноземною мовами.

7 Порядок контролю виконання та захисту БКР

Виконання етапів графічної та розрахункової документації БКР контролюється науковим керівником згідно зі встановленими термінами. Захист БКР відбувається на засіданні Екзаменаційної комісії, затвердженої наказом ректора.

8 Вимоги до оформлювання та порядок виконання БКР

8.1 При оформлюванні БКР використовуються:

— ДСТУ 3008: 2015 «Звіти в сфері науки і техніки. Структура та правила оформлювання»;

— ДСТУ 8302: 2015 «Бібліографічні посилання. Загальні положення та правила складання»;

— ГОСТ 2.104–2006 «Єдина система конструкторської документації. Основні написи»;

— методичні вказівки до виконання бакалаврських кваліфікаційних робіт для студентів спеціальності 123 «Комп’ютерна інженерія».

— документи на які посилаються у вище вказаних.

8.2 Порядок виконання БКР викладено в «Положення про кваліфікаційні роботи на першому (бакалаврському) рівні вищої освіти СУЯ ВНТУ-03.02.02-П.001.01:21.

ДОДАТОК Б
ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: _____Інформаційна система обліку гуманітарної допомоги_____
Тип роботи: _____ бакалаврська кваліфікаційна робота _____
(бакалаврська кваліфікаційна робота / магістерська кваліфікаційна робота)
Підрозділ _____кафедра обчислювальної техніки_____
(кафедра, факультет, навчальна група)

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism ___1___%

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

✓ Запозичення, виявлені у роботі, є законними і не містять ознак плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту

☐ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.

☐ У роботі виявлено ознаки плагіату та/або текстових маніпуляцій як спроб укриття плагіату, фабрикації, фальсифікації, що суперечить вимогам законодавства та нормам академічної доброчесності. Робота до захисту не приймається.

Експертна комісія:

_____Азаров О.Д., зав. каф. ОТ _____ _____
(прізвище, ініціали, посада) (підпис)
____Крупельницький Л.В., доц. каф ОТ_____ _____
(прізвище, ініціали, посада) (підпис)

Особа, відповідальна за перевірку _____ ___Захарченко С.М.____
(підпис) (прізвище, ініціали)

З висновком експертної комісії ознайомлений(-на)

Керівник _____ __ к.т.н., доц. каф. ОТ ____Городецька О.С_____
(підпис) (прізвище, ініціали, посада)

Здобувач _____ _____Шуляк М.В._____
(підпис) (прізвище, ініціали)

ДОДАТОК В

Графічна частина

ІНФОРМАЦІЙНА СИСТЕМА ОБЛІКУ ГУМАНІТАРНОЇ ДОПОМОГИ

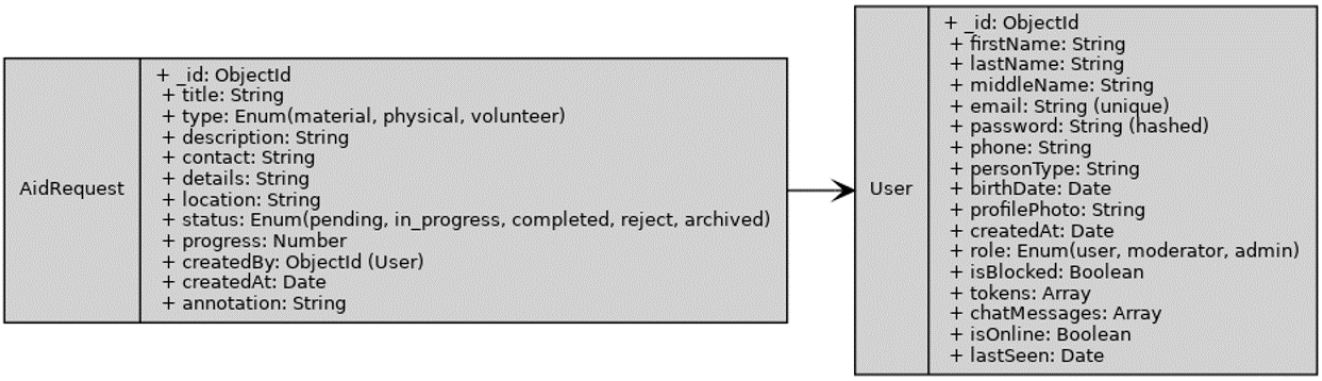


Рисунок В.1 — Фізична ER-модель бази даних

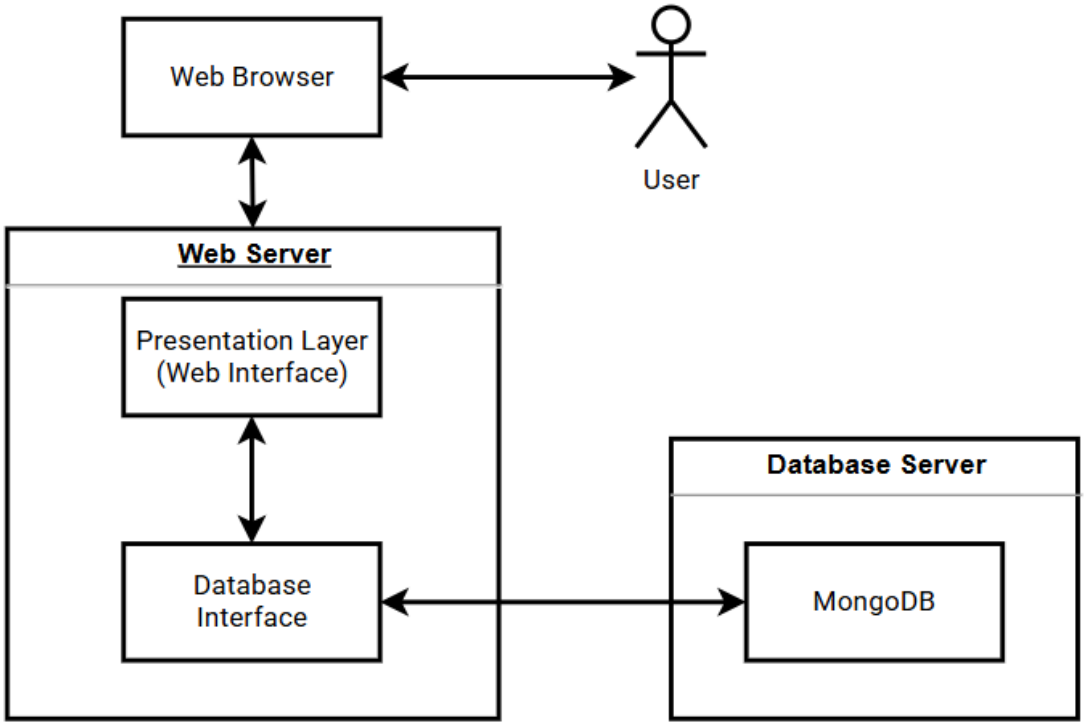


Рисунок В.2 — Діаграма розгортання UML

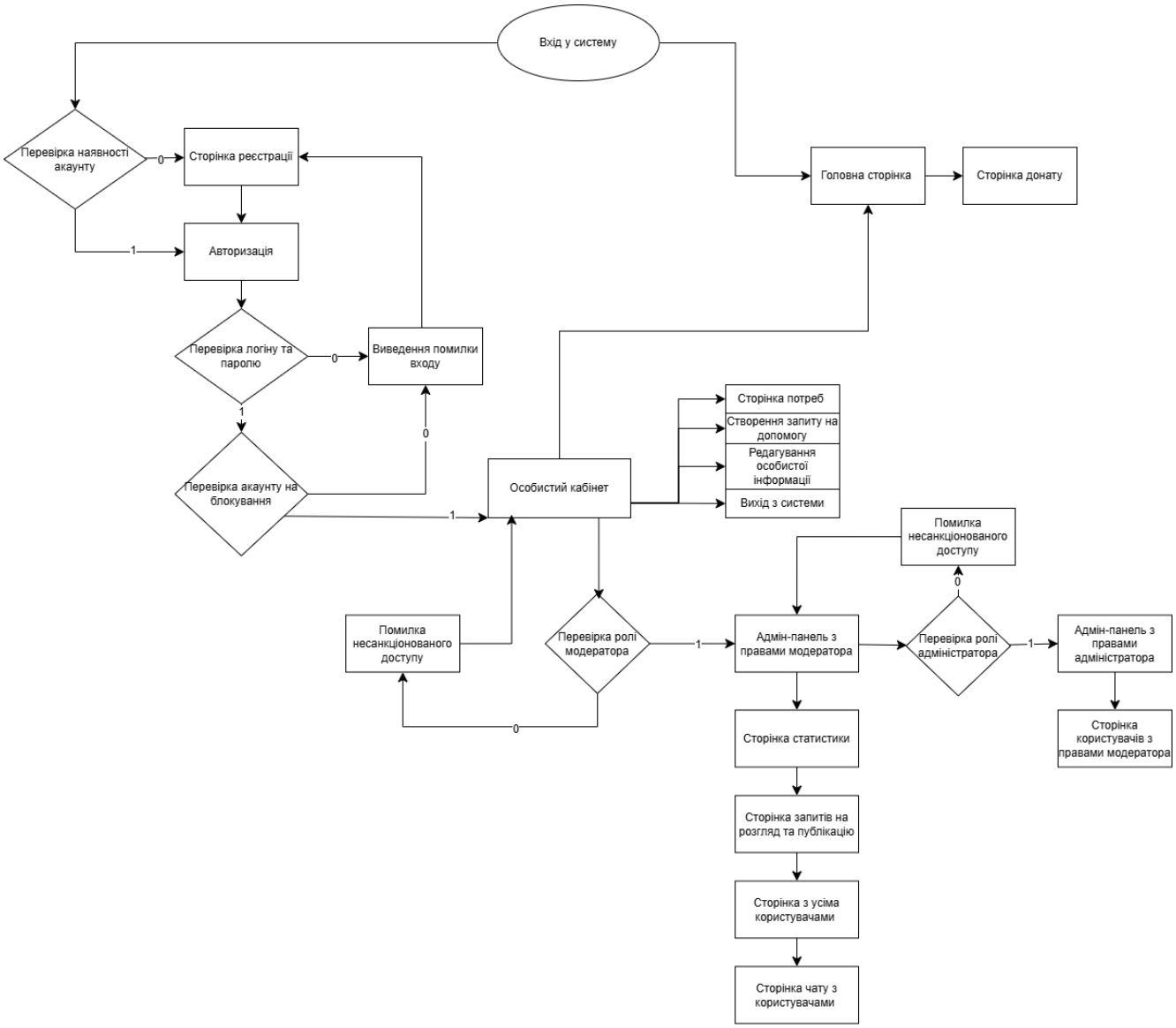


Рисунок В.3 — Загальна структура інформаційної системи

ДОДАТОК Г

Ілюстративна частина

ІНФОРМАЦІЙНА СИСТЕМА ОБЛІКУ ГУМАНІТАРНОЇ ДОПОМОГИ

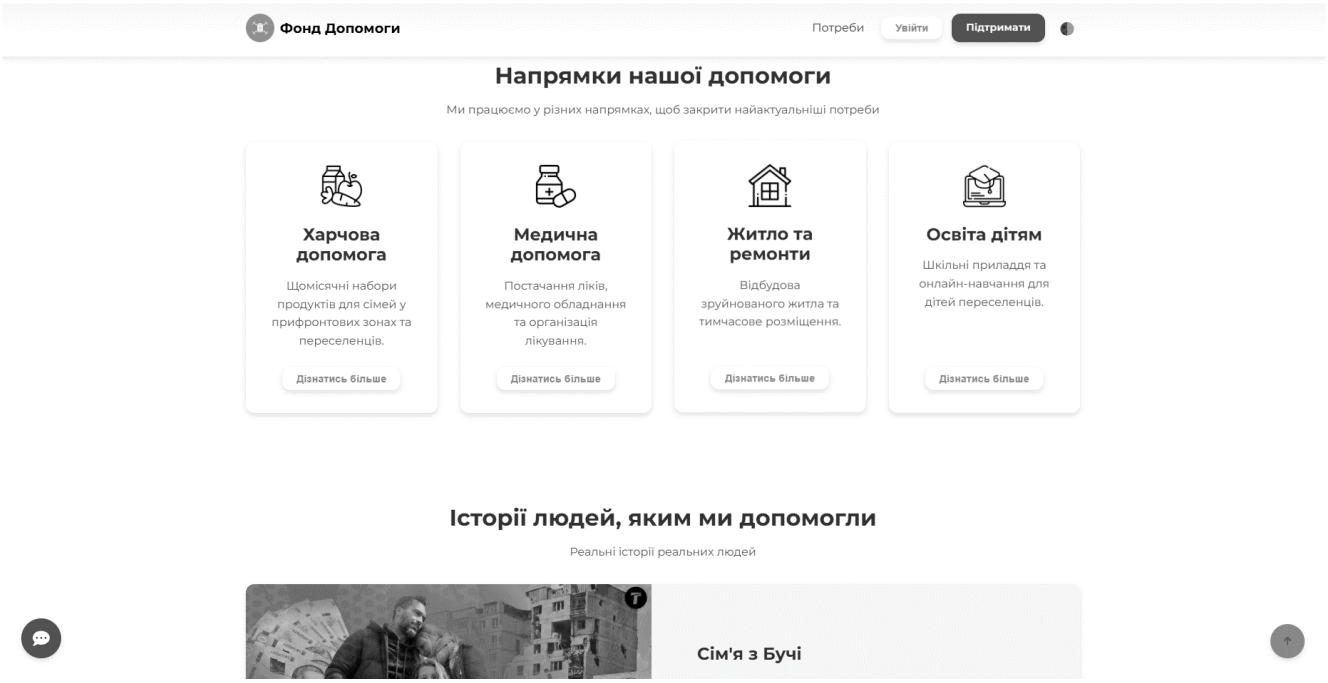


Рисунок Г.1 — Головна сторінка фонду

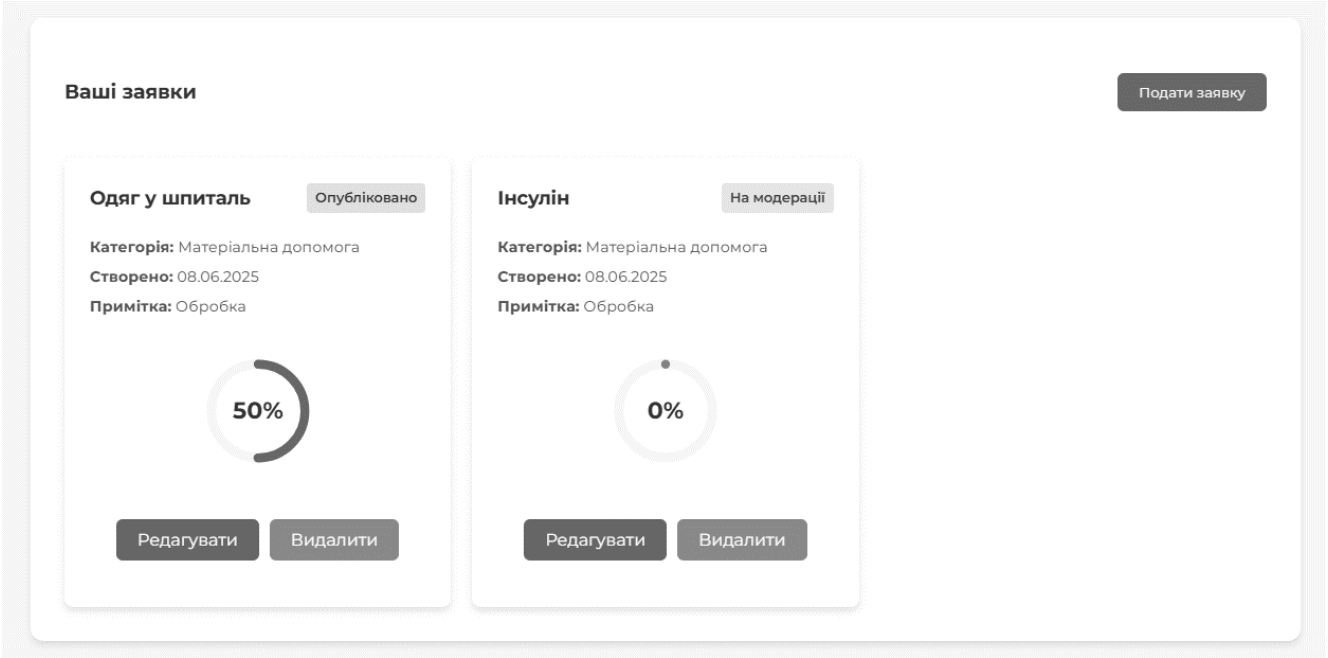


Рисунок Г.2 — Сторінка заявок користувача

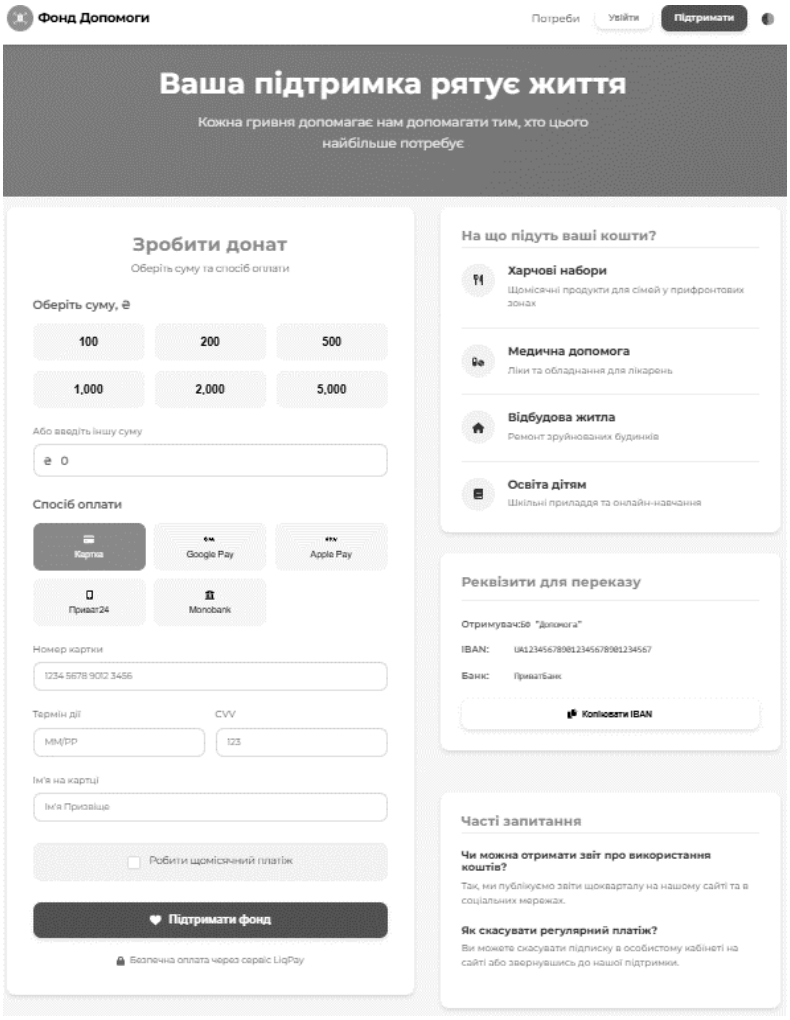


Рисунок Г.3 — Сторінка підтримки фонду

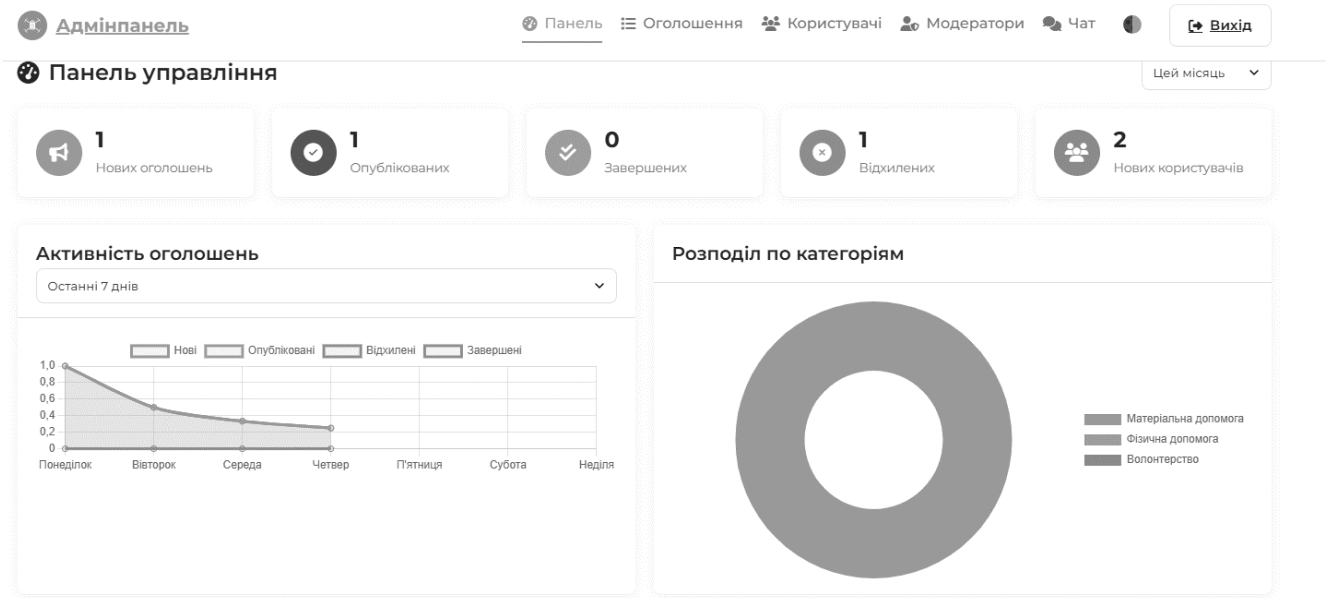


Рисунок Г.4 — Головна сторінка адмінпанелі

ДОДАТОК Д

Лістинги програмного забезпечення

Лістинг Д.1 — Лістинг коду файлу “auth”

```
const jwt = require('jsonwebtoken');
const User = require('../models/User');
module.exports = async (req, res, next) => {
  try {
    const token = req.headers.authorization?.split(' ')[1];
    if (!token) {
      return res.status(401).json({
        success: false,
        message: 'Неавторизований доступ'
      });
    }
    const decoded = jwt.verify(token, process.env.JWT_SECRET);
    const user = await User.findById(decoded.userId);
    if (!user || user.isBlocked) {
      return res.status(401).json({
        success: false,
        message: 'Користувач заблокований або не існує'
      });
    }
    req.userId = decoded.userId;
    req.userRole = decoded.role;
    next();
  } catch (err) {
    res.status(401).json({
      success: false,
      message: 'Неавторизований доступ'});};
```

Лістинг Д.2 — Лістинг коду файлу “server.js’’

```
require('dotenv').config();
const express = require('express');
const cors = require('cors');
const connectDB = require('./config/db');
const authRoutes = require('./routes/authRoutes');
const aidRoutes = require('./routes/aidRoutes');
const requestRoutes = require('./routes/requestRoutes');
const adminRoutes = require('./routes/adminRoutes');
const app = express();
```

```
// Middleware
app.use(express.json());
app.use(cors({
    origin: ['http://localhost:5500', 'http://127.0.0.1:5500'],
    methods: ['GET', 'POST', 'PUT', 'DELETE', 'PATCH'],
    credentials: true,
    allowedHeaders: ['Content-Type', 'Authorization']
}));
app.use('/uploads', express.static('public/uploads'));

// Підключення до MongoDB
connectDB();

// Маршрути
app.use('/api/auth', authRoutes);
app.use('/api/requests', aidRoutes);
app.use('/api/requests', requestRoutes);
app.use('/api/admin', adminRoutes);

const PORT = process.env.PORT || 5000;
app.listen(PORT, () => console.log(`Сервер працює на порті ${PORT}`));
```