МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра комп'ютерних наук

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: **«РОЗРОБКА ВЕБЗАСТОСУНКУ ДЛЯ ПІДТРИМКИ РОБОТИ БЛАГОДІЙНОГО ФОНДУ»**

Виконав: студент 4 курсу, групи 6.1221
спеціальності 122 Комп'ютерні науки
(шифр і назва спеціальності)
освітньої програми Комп'ютерні науки
(назва освітньої програми)
О.В. Трухан
(ініціали та прізвище)
Керівник завідувач кафедри комп'ютерних наук, професор, д.т.н. Шило Г.М.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)
Рецензент завідувач кафедри фундаментальної та прикладної математики, професор, д.т.н. Гребенюк С.М.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя 2025

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра комп'ютерних наук

Рівень вищої освіти бакалавр

Спеціальність 122 Комп'ютерні науки
(шифр і назва)

Освітня програма Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри комп'ютерних наук,
д.т.н., професор

_____ Шило Г.М.
(підпис)

" 24 " 12 2024 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Трухану Олексію Вікторовичу
(прізвище, ім'я та по-батькові)

1. Тема роботи Розробка вебзастосунку для підтримки роботи благодійного фонду

керівник роботи Шило Галина Миколаївна, д.т.н., професор
(прізвище, ім'я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 20 » грудня 2024 року № 2001-с

2. Строк подання студентом роботи 02.06.2025

3. Вихідні дані до роботи 1. Постановка задачі
2. Перелік літератури

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Аналіз предметної області.
2. Розробка архітектури проекту.
3. Розробка алгоритмів та кодова реалізація застосунку.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
презентація

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання 24.12.2024

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	30.12.2024	виконано
2.	Збір вихідних даних.	14.01.2025	виконано
3.	Обробка методичних та теоретичних	11.02.2025	виконано
	джерел.		
4.	Розробка першого та другого розділу.	02.03.2025	виконано
5.	Розробка третього розділу.	23.03.2025	виконано
6.	Оформлення та нормоконтроль	20.04.2025	виконано
	кваліфікаційної роботи бакалавра.		
7.	Захист кваліфікаційної роботи.	18.06.2025	виконано

Студент _____ О.В Трухан
(підпис) (ініціали та прізвище)

Керівник роботи _____ Г.М. Шило
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ О.Г.Спиця
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота бакалавра «Розробка вебзастосунку для підтримки роботи благодійного фонду» : 64 с., 1 рис., 5 табл., 29 джерел.

АДМІНІСТРАТИВНА ПАНЕЛЬ, АУТЕНТИФІКАЦІЯ, БАЗА ДАНИХ, КОРИСТУВАЦЬКИЙ ІНТЕРФЕЙС, ПЛАТІЖНА СИСТЕМА, РЕЄСТРАЦІЯ КОРИСТУВАЧА, ФОРМА ПОЖЕРТВИ, ВЕБЗАСТОСУНОК.

Об’єкт дослідження – процеси автоматизації діяльності благодійного фонду за допомогою інформаційних технологій. Предмет дослідження – вебзастосунок, призначений для підтримки збору пожертв, управління заявками, звітності та комунікації з користувачами.

Мета роботи полягає в розробці вебзастосунку для підтримки діяльності благодійного фонду, який забезпечить автоматизацію збору пожертв, управління заявками, звітності, а також надасть зручний інтерфейс для донорів і отримувачів допомоги.

Предмет дослідження – вебзастосунок, призначений для підтримки збору пожертв, управління заявками, звітності та комунікації з користувачами.

Метод дослідження – практичний, що включає аналіз, проєктування та експериментальну розробку.

Розроблений вебзастосунок досяг поставленої мети, автоматизуючи ключові процеси благодійного фонду: збір пожертв, управління заявками, звітність і комунікацію з користувачами. Він забезпечує прозорість, безпеку та зручність, що сприяє підвищенню довіри донорів і ефективності роботи фонду. Практична цінність роботи полягає в створенні готового рішення, яке може бути адаптоване для інших благодійних організацій в Україні. Наукова новизна полягає в розробці застосунку, адаптованого до локального контексту, з акцентом на інтеграцію локальних платіжних систем і підтримку гуманітарних потреб.

SUMMARY

Bachelor’s Qualification Theses "Development of a web application to support the activities of a charity foundation": 64 pages, 1 figure, 5 tables, 29 sources.

ADMIN PANEL, AUTHENTICATION, DATABASE, USER INTERFACE, PAYMENT SYSTEM, USER REGISTRATION, DONATION FORM, WEB APPLICATION.

The subject of research is the processes of automating the activities of a charity foundation using information technology. The object of research is a web application designed to support the collection of donations, management of applications, reporting, and communication with users.

The goal of the work is to develop a web application to support the activities of a charity foundation, which will ensure the automation of donation collection, application management, reporting, and provide a convenient interface for donors and beneficiaries.

The subject of the research is a web application designed to support donation collection, application management, reporting, and communication with users.

The research method is practical, involving analysis, design, and experimental development.

The developed web application has achieved its goal by automating key processes of the charitable foundation: donation collection, application management, reporting, and communication with users. It ensures transparency, security, and convenience, which contributes to increased donor trust and the efficiency of the foundation’s work. The practical value of the work lies in the creation of a ready-made solution that can be adapted for other charitable organizations in Ukraine. The scientific novelty is in the development of an application tailored to the local context, with an emphasis on integrating local payment systems and supporting humanitarian needs.

ЗМІСТ

Завдання на кваліфікаційну роботу 2

Реферат 5

Summary 6

Вступ 7

1 Аналіз предметної області та вимог до вебзастосунку 10

1.1 Огляд діяльності благодійних фондів та їх потреб у автоматизації 10

1.2 Аналіз існуючих вебзастосунків для благодійних організацій 12

1.3 Вивчення платіжних систем для інтеграції збору пожертв 15

1.4 Визначення функціональних та нефункціональних вимог до застосунку 18

1.5 Аналіз технічних обмежень та ризиків розробки 22

1.6 Висновки до розділу 24

2 Проектування та розробка вебзастосунку 26

2.1 Вибір технологічного стеку для розробки вебзастосунку 26

2.2 Проектування архітектури системи та бази даних 29

2.3 Розробка модуля інтеграції з платіжними системами 35

2.4 Реалізація функціоналу управління заявками та звітності 38

2.5 Створення особистих кабінетів для користувачів 41

2.6 Висновки до розділу 44

3 Тестування та впровадження вебзастосунку 46

3.1 Розробка плану тестування застосунку 46

3.2 Проведення тестування функціоналу та безпеки 48

3.3 Впровадження вебзастосунку в роботу фонду 50

3.4 Підготовка інструкцій для користувачів 51

3.5 Висновки до розділу 55

Висновки 57

Перелік посилань 60

ВСТУП

Сучасний світ характеризується швидким розвитком інформаційних технологій, які суттєво змінюють підходи до організації діяльності в різних сферах, включаючи благодійність. Благодійні фонди відіграють ключову роль у підтримці соціально вразливих груп населення, фінансуванні гуманітарних проєктів та розв'язанні актуальних суспільних проблем. В Україні, особливо в умовах війни, діяльність благодійних організацій набула надзвичайної важливості, адже вони забезпечують гуманітарну допомогу, підтримку внутрішньо переміщених осіб та відновлення інфраструктури. Проте ефективність роботи таких організацій значною мірою залежить від їхньої здатності оперативно залучати кошти, прозоро звітувати перед донорами та координувати заявки на допомогу. У цьому контексті створення спеціалізованих вебзастосунків стає не лише актуальним, але й необхідним завданням для автоматизації та оптимізації діяльності благодійних фондів.

Розробка вебзастосунку для підтримки роботи благодійного фонду є відповіддю на сучасні виклики, пов'язані з необхідністю цифровізації благодійної діяльності. Такий застосунок дозволяє інтегрувати платіжні системи для збору пожертв, автоматизувати управління заявками на допомогу, створювати прозору звітність про надходження та витрати, а також забезпечувати зручну взаємодію між фондом, донорами та отримувачами допомоги через особисті кабінети. Крім того, вебзастосунок може стати платформою для інформування громадськості про проєкти фонду, що сприяє підвищенню довіри до організації та залученню нових донорів.

Актуальність теми зумовлена кількома факторами. По-перше, зростання обсягів онлайн-пожертв у світі та в Україні вимагає надійних і безпечних інструментів для їх обробки. По-друге, потреба в прозорості діяльності благодійних фондів, особливо в умовах війни, є критично важливою для підтримання довіри донорів. По-третє, автоматизація рутинних процесів, таких

як обробка заявок і підготовка звітів, дозволяє фондам зосередитися на стратегічних завданнях. Нарешті, створення зручних особистих кабінетів для донорів і отримувачів допомоги сприяє підвищенню залученості користувачів і спрощує комунікацію з фондом.

Мета роботи полягає в розробці вебзастосунку для підтримки діяльності благодійного фонду, який забезпечить автоматизацію збору пожертв, управління заявками, звітності, а також надасть зручний інтерфейс для донорів і отримувачів допомоги. Для досягнення цієї мети визначено такі завдання:

- провести аналіз діяльності благодійних фондів та визначити вимоги до вебзастосунку.
- дослідити існуючі рішення та платіжні системи для інтеграції збору пожертв.
- спроектувати архітектуру вебзастосунку, включаючи базу даних і модулі функціоналу.
- розробити вебзастосунок із реалізацією збору пожертв, управління заявками, звітності та особистих кабінетів.
- провести тестування застосунку на відповідність вимогам безпеки, функціональності та продуктивності.
- підготувати рекомендації щодо впровадження та підтримки вебзастосунку в роботі благодійного фонду.

Об’єкт дослідження – процеси автоматизації діяльності благодійного фонду за допомогою інформаційних технологій. Предмет дослідження – вебзастосунок, призначений для підтримки збору пожертв, управління заявками, звітності та комунікації з користувачами. Методи дослідження, використані в роботі, включають:

- аналіз літератури та існуючих рішень для визначення вимог до вебзастосунку.
- системний підхід для проектування архітектури та функціональних модулів.

- програмування та тестування для реалізації та перевірки працездатності застосунку.
- порівняльний аналіз для оцінки ефективності розробленого рішення.

Наукова новизна роботи полягає в розробці комплексного вебзастосунку, адаптованого до потреб українських благодійних фондів, з урахуванням локального контексту, зокрема високого попиту на гуманітарну допомогу в умовах війни. Особливу увагу приділено інтеграції сучасних платіжних систем, автоматизації звітності та забезпеченню прозорості діяльності фонду.

Практична цінність роботи полягає в створенні готового рішення, яке може бути використано благодійними фондами для підвищення ефективності їхньої діяльності. Вебзастосунок дозволяє скоротити час на обробку заявок, забезпечити безпечний збір пожертв, автоматизувати звітність і покращити взаємодію з донорами та отримувачами допомоги. Розроблені рекомендації щодо впровадження та підтримки застосунку сприятимуть його успішному використанню в реальних умовах.

Структура роботи складається зі вступу, трьох основних розділів, висновків і списку використаних джерел. У першому розділі проведено аналіз предметної області та визначено вимоги до вебзастосунку. Другий розділ присвячено проектуванню та розробці застосунку, включаючи вибір технологій, створення модулів і особистих кабінетів. А в третьому тестування програмного продукту.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ВИМОГ ДО ВЕБЗАСТОСУНКУ

1.1 Огляд діяльності благодійних фондів та їх потреб у автоматизації

Благодійні фонди є важливими інституціями, які сприяють розв'язанню соціальних, гуманітарних та екологічних проблем у суспільстві. Вони залучають ресурси від донорів, включаючи приватних осіб, компанії та державні установи, для фінансування проектів, спрямованих на підтримку вразливих груп населення, розвиток освіти, охорону здоров'я та інші суспільно значущі ініціативи. В Україні діяльність благодійних фондів набула особливого значення в умовах війни, оскільки вони забезпечують гуманітарну допомогу, підтримку внутрішньо переміщених осіб та відновлення інфраструктури [29]. Згідно з дослідженням Асоціації благодійників України, у 2024 році прозорість діяльності фондів стала ключовим фактором для залучення донорів, що підкреслює потребу в автоматизованих системах для управління фінансами та звітністю [29].

Основними завданнями благодійних фондів є збір пожертв, розподіл ресурсів, координація заявок на допомогу та інформування громадськості про результати їхньої діяльності. Однак ці процеси часто є трудомісткими та потребують значних людських ресурсів. Наприклад, ручна обробка заявок на допомогу може призводити до затримок, а відсутність прозорої звітності знижує довіру донорів. Аналіз діяльності некомерційних організацій показує, що впровадження вебтехнологій значно підвищує їхню ефективність, дозволяючи автоматизувати рутинні процеси та покращувати комунікацію з зацікавленими сторонами [22]. Зокрема, вебзастосунки дозволяють фондам швидше обробляти пожертви, надавати донорам доступ до історії їхніх внесків та публікувати звіти в реальному часі [22].

Однією з ключових потреб благодійних фондів є інтеграція платіжних систем для збору пожертв. Онлайн-пожертви стають дедалі популярнішими завдяки зручності та доступності. Дослідження показують, що організації, які використовують вебсайти з інтегрованими платіжними шлюзами, отримують на 30–50% більше пожертв порівняно з тими, що покладаються на офлайн-методи [11]. В Україні ця тенденція також набирає обертів, особливо після початку війни, коли багато фондів почали активно використовувати цифрові платформи для залучення міжнародних донорів [24]. Наприклад, ініціатива Web Volunteers демонструє, як створення вебсайтів для благодійних організацій сприяє швидкому збору коштів і координації волонтерських проектів [24].

Ще однією важливою потребою є автоматизація управління заявками на допомогу. Благодійні фонди щодня отримують численні запити від осіб, які потребують підтримки, і їх обробка вручну може бути неефективною. Вебзастосунки дозволяють створювати структуровані форми для подання заявок, автоматично категоризувати їх за пріоритетністю та відстежувати статус виконання. Такий підхід не лише економить час, але й забезпечує справедливий розподіл ресурсів. Крім того, автоматизація звітності є критично важливою для підтримання прозорості. За даними міжнародних досліджень, донори частіше підтримують організації, які регулярно публікують детальні звіти про використання коштів [12]. Вебзастосунки можуть генерувати такі звіти автоматично, що знижує адміністративне навантаження на працівників фонду [12].

Комунікація з донорами та отримувачами допомоги також потребує вдосконалення. Багато фондів стикаються з проблемою недостатньої залученості донорів через відсутність зручних каналів взаємодії. Особистий кабінет у вебзастосунку може вирішити цю проблему, надаючи донорам доступ до інформації про їхні внески, а отримувачам – можливість відстежувати статус заявок. Крім того, вебзастосунки дозволяють фондам інформувати громадськість про свої проекти через інтегровані блоги, новинні стрічки чи розсилки. Це сприяє підвищенню довіри та залученню нових донорів [11].

В Україні особливу увагу слід приділити локальному контексту. Зокрема, війна зумовила зростання кількості благодійних організацій, які працюють у сфері гуманітарної допомоги. Перелік фондів, складений Асоціацією міст України, включає десятки організацій, що надають підтримку постраждалим від війни, але багато з них стикаються з проблемами координації та звітності через відсутність сучасних інформаційних систем [28]. Впровадження вебзастосунків може значно полегшити їхню роботу, особливо в умовах обмежених ресурсів [28].

Аналіз діяльності благодійних фондів показує, що автоматизація є необхідною умовою для підвищення їхньої ефективності. Вебзастосунки дозволяють оптимізувати збір пожертв, управління заявками, звітність і комунікацію з користувачами. Особливо актуальними ці рішення є в Україні, де благодійні фонди відіграють ключову роль у подоланні гуманітарної кризи. Подальший аналіз вимог до вебзастосунку має враховувати як глобальні тенденції, так і локальні потреби, щоб забезпечити створення функціонального та зручного рішення.

1.2 Аналіз існуючих вебзастосунків для благодійних організацій

Сучасні благодійні організації активно використовують вебзастосунки для автоматизації своєї діяльності, залучення донорів та підвищення прозорості. Аналіз існуючих рішень дозволяє визначити їхні сильні та слабкі сторони, а також сформувати вимоги до розробки нового вебзастосунку, адаптованого до потреб благодійних фондів, зокрема в Україні. У цьому підрозділі розглянуто кілька популярних вебзастосунків, які використовуються благодійними організаціями, з акцентом на їхній функціонал, зручність та відповідність потребам фондів.

Одним із найвідоміших рішень є платформа GlobalGiving, яка функціонує як краудфандинговий вебзастосунок для неприбуткових організацій у всьому

світі. GlobalGiving дозволяє фондам створювати проєкти для збору коштів, інтегрує платіжні системи (наприклад, Stripe і PayPal) та забезпечує донорам доступ до звітів про використання пожертв. Платформа також пропонує інструменти аналітики для оцінки ефективності кампаній. Проте її недоліком є висока комісія (до 5–12% від суми пожертв), що може бути обтяжливим для невеликих фондів [6]. В Україні подібні платформи менш поширені, але GlobalGiving використовується деякими міжнародними фондами для залучення донорів із-за кордону [6].

Іншим прикладом є Classy, вебзастосунок, який спеціалізується на зборі пожертв і організації благодійних кампаній. Classy пропонує гнучкі шаблони для створення сторінок збору коштів, інтеграцію з платіжними системами, а також модулі для управління подіями та комунікації з донорами через email-розсилки. Особливістю Classy є можливість створення особистих кабінетів для донорів, де вони можуть відстежувати свої внески та отримувати персоналізовані подяки. Однак платформа орієнтована переважно на ринок США, що ускладнює її адаптацію до українського контексту, зокрема через відсутність підтримки локальних платіжних систем, таких як LiqPay [2]. Крім того, висока вартість підписки робить Classy менш доступною для невеликих організацій [2].

В Україні прикладом локального рішення є вебсайт благодійного фонду «Повернись живим», який активно використовується для збору коштів на підтримку Збройних сил України. Сайт інтегрує платіжні системи (зокрема, LiqPay і WayForPay), дозволяє донорам робити одноразові або регулярні пожертви та публікує детальні звіти про витрати. Функціонал включає новинну стрічку для інформування про проекти та форму для подання заявок на волонтерську підтримку. Проте сайт не має розвинених особистих кабінетів для донорів і обмежений у функціоналі управління заявками на допомогу, що знижує його універсальність для інших типів благодійних фондів [13]. Згідно з даними Опендатабот, фонд «Повернись живим» зібрав значні суми пожертв у

2023 році, що свідчить про ефективність їхнього веб застосунку в умовах війни [13].

Платформа Givebutter є ще одним популярним рішенням, яке поєднує функції збору пожертв, управління подіями та CRM для взаємодії з донорами. Givebutter підтримує безкоштовну обробку платежів (з можливістю донорам покривати комісію) і пропонує інструменти для створення кампаній, включаючи відеоконтент і форми для збору даних. Сильною стороною є простота налаштування та адаптивний дизайн, який забезпечує зручність використання на мобільних пристроях. Однак платформа не має локалізації для України, а її CRM-функціонал може бути надмірно складним для невеликих фондів [8]. Givebutter також акцентує на емоційному залученні донорів через інтеграцію з соціальними мережами, що відповідає сучасним тенденціям [8].

Для порівняння було проаналізовано також шаблони вебсайтів, які пропонуються для благодійних організацій, наприклад, на платформі TemplateMonster. Такі шаблони, створені на базі WordPress або Joomla, включають базовий функціонал для збору пожертв, блог для інформування та форми зворотного зв’язку. Вони є економічно вигідним рішенням для фондів із обмеженим бюджетом, оскільки не потребують значних витрат на розробку. Проте такі шаблони часто обмежені в кастомізації та не підтримують складні функції, як-от автоматизована звітність чи управління заявками [18]. В Україні подібні шаблони використовуються невеликими фондами, але їхній функціонал потребує доопрацювання для повноцінної автоматизації [18].

Аналіз показує, що більшість існуючих вебзастосунків орієнтовані на ринки США та Європи, що створює труднощі для українських фондів через відсутність локалізації, підтримки місцевих платіжних систем і високі витрати(Таблиця 1.1 – Порівняльний аналіз вебзастосунків для благодійних організацій). Локальні рішення, як-от сайт «Повернись живим», демонструють ефективність у зборі коштів, але не є універсальними через обмежений функціонал. Загалом, ідеальний вебзастосунок для благодійного фонду має поєднувати інтеграцію з платіжними системами, автоматизацію звітності,

управління заявками, особисті кабінети та адаптивний дизайн, враховуючи локальний контекст.

Таблиця 1.1 – Порівняльний аналіз вебзастосунків для благодійних організацій

Назва платформи	Основні недоліки
GlobalGiving	Висока комісія, відсутність локалізації [6]
Classy	Висока вартість, обмежена локалізація [2]
Повернись живим	Обмежений функціонал кабінетів [13]
Givebutter	Складний CRM, відсутність локалізації [8]
TemplateMonster (WordPress)	Обмежена кастомізація [18]

Аналіз існуючих вебзастосунків показує, що більшість рішень мають сильні сторони в зборі пожертв і звітності, але не повною мірою відповідають потребам українських благодійних фондів через відсутність локалізації та підтримки локальних платіжних систем. Локальні рішення, як-от сайт «Повернись живим», є ефективними в умовах війни, але потребують розширення функціоналу, зокрема особистих кабінетів і управління заявками. Розробка нового вебзастосунку має враховувати ці недоліки, пропонуючи універсальне рішення з адаптацією до українського контексту.

1.3 Вивчення платіжних систем для інтеграції збору пожертв

Інтеграція платіжних систем є ключовим елементом вебзастосунків для благодійних фондів, оскільки вона забезпечує зручний і безпечний збір пожертв від донорів. В Україні, особливо в умовах війни, попит на онлайн-пожертви значно зріс, що підкреслює важливість вибору надійних і адаптованих до локального контексту платіжних систем. У цьому підрозділі розглянуто основні платіжні системи, які можуть бути використані для збору пожертв, їхні

функціональні можливості, переваги та недоліки, а також специфіку інтеграції в вебзастосунки.

Однією з найпопулярніших міжнародних платіжних систем є Stripe, яка підтримує обробку платежів у 135+ валютах і дозволяє приймати пожертви з усього світу. Stripe пропонує гнучке API для інтеграції, що дозволяє налаштувати платіжні форми, регулярні платежі та навіть платіжні посилання для соціальних мереж. Система відповідає стандартам безпеки PCI-DSS, що гарантує захист даних донорів. Проте для українських благодійних фондів Stripe має певні обмеження, зокрема необхідність реєстрації юридичної особи в країнах, які підтримуються платформою, та відносно високу комісію (2,9% + 0,30 USD за транзакцію). Це може бути проблематичним для невеликих фондів із обмеженим бюджетом [14]. Stripe активно використовується міжнародними благодійними платформами, такими як GlobalGiving, для обробки транзакцій [14].

В Україні однією з провідних платіжних систем є Fondy, яка спеціалізується на інтернет-еквайрингу та підтримує прийом платежів через банківські картки Visa, Mastercard, а також електронні гаманці Apple Pay і Google Pay. Fondy дозволяє створювати платіжні кнопки, посилання та QR-коди для збору пожертв як онлайн, так і офлайн, що є зручним для благодійних кампаній у соціальних мережах. Система пропонує гнучкі налаштування безпеки та підтримує регулярні списання, що важливо для залучення постійних донорів. Комісія Fondy становить близько 2,7% за транзакцію, що є конкурентною пропозицією на українському ринку. Крім того, Fondy має готові плагіни для інтеграції з популярними CMS, такими як WordPress, що спрощує її впровадження [26]. Для благодійних фондів Fondy є привабливим рішенням завдяки локальній підтримці та адаптації до українського законодавства [26].

Ще одна популярна українська платіжна система – WayForPay, яка широко використовується для електронної комерції та збору пожертв. WayForPay підтримує миттєві платежі через банківські картки, прямі перекази та інтеграцію з CMS, такими як WooCommerce, OpenCart і Shopify. Система

надає детальну аналітику транзакцій, що дозволяє фондам відстежувати джерела пожертв і оптимізувати кампанії. Комісія становить 2,5% для транзакцій до 500 000 грн, що робить її доступною для середніх і великих фондів. Однак служба підтримки WayForPay іноді працює недостатньо оперативно, що може створювати труднощі під час інтеграції чи вирішення технічних питань [6]. WayForPay активно використовується українськими фондами, такими як «Повернись живим», для збору пожертв [6].

LiqPay, розроблена ПриватБанком, є ще одним поширеним рішенням в Україні. LiqPay дозволяє приймати платежі через банківські картки, Приват24, а також QR-коди, що зручно для офлайн-зборів. Система має просте API для інтеграції та підтримує мультивалютні платежі, що важливо для залучення міжнародних донорів. Комісія становить 2,75% для більшості транзакцій, але може бути знижена для великих обсягів. Основним недоліком LiqPay є залежність від інфраструктури ПриватБанку, що може створювати ризики у разі технічних збоїв. Крім того, система менш гнучка в порівнянні з Fondy чи Stripe для складних інтеграцій [25]. LiqPay активно застосовується благодійними фондами завдяки широкій популярності ПриватБанку серед українців [25].

Міжнародна платіжна система PayPal також заслуговує на увагу, оскільки вона має майже 400 мільйонів користувачів у 200 країнах. PayPal підтримує інтеграцію з багатьма CMS і дозволяє налаштувати регулярні пожертви та платіжні кнопки. У 2022 році PayPal розширила свою присутність в Україні, що зробило її доступною для благодійних фондів. Однак комісія (2,9% + 0,30 USD) і обмеження на виведення коштів для деяких типів акаунтів можуть ускладнювати її використання. PayPal є ефективним рішенням для залучення міжнародних донорів, але потребує додаткових зусиль для інтеграції в локальний контекст [7]. PayPal використовується благодійними організаціями для збору коштів на гуманітарні проєкти в Україні [7].

Аналіз платіжних систем показує, що для українських благодійних фондів найоптимальнішими є локальні рішення, такі як Fondy, WayForPay і LiqPay, завдяки їхній адаптації до українського ринку, нижчим комісіям і підтримці

локальних платіжних методів(Таблиця 1.2 – Порівняльний аналіз платіжних систем для інтеграції збору пожертв). Міжнародні системи, як Stripe і PayPal, більше підходять для фондів, орієнтованих на глобальних донорів, але потребують додаткових витрат на інтеграцію та юридичне оформлення.

Таблиця 1.2 – Порівняльний аналіз платіжних систем для інтеграції збору пожертв

Платіжна система	Основні недоліки
Stripe	Висока комісія, складна реєстрація [14]
Fondy	Обмежена міжнародна присутність [26]
WayForPay	Повільна підтримка [6]
LiqPay	Залежність від ПриватБанку [25]
PayPal	Обмеження на виведення коштів [7]

Вивчення платіжних систем показує, що для українських благодійних фондів найперспективнішими є Fondy, WayForPay і LiqPay завдяки їхній локалізації, підтримці популярних платіжних методів і конкурентним комісіям. Stripe і PayPal більше підходять для міжнародних проектів, але мають вищі витрати та складнощі з інтеграцією. Вибір платіжної системи для вебзастосунку має базуватися на цільовій аудиторії фонду, обсязі транзакцій і технічних можливостях інтеграції.

1.4 Визначення функціональних та нефункціональних вимог до застосунку

Визначення функціональних і нефункціональних вимог до вебзастосунку є критично важливим етапом, який дозволяє чітко окреслити його можливості, забезпечити відповідність потребам благодійного фонду та врахувати сучасні стандарти якості. Функціональні вимоги описують конкретні функції, які має виконувати застосунок, тоді як нефункціональні вимоги стосуються його

продуктивності, безпеки, зручності використання та інших якісних характеристик. У цьому підрозділі визначено вимоги до вебзастосунку для підтримки роботи благодійного фонду з урахуванням аналізу предметної області, потреб українських фондів і сучасних тенденцій у веброзробці.

Функціональні вимоги базуються на основних завданнях благодійного фонду: збір пожертв, управління заявками, автоматизація звітності, створення особистих кабінетів і інформування про проєкти (Таблиця 1.3 – Функціональні вимоги до вебзастосунку). Першочерговою функцією є інтеграція з платіжними системами для забезпечення безпечного збору пожертв. Застосунок має підтримувати одноразові та регулярні платежі через популярні платіжні системи, такі як Fondy, WayForPay або LiqPay, що є особливо актуальним для українського контексту [26]. Крім того, донори повинні мати можливість отримувати підтвердження транзакцій і переглядати історію своїх внесків у реальному часі, що підвищує довіру до фонду [26].

Другою важливою функцією є управління заявками на допомогу. Вебзастосунок має надавати користувачам зручну форму для подання заявок, яка включає поля для особистих даних, опису потреби та завантаження документів. Адміністратори фонду повинні мати можливість категоризації заявок за пріоритетністю, відстежувати їхній статус і надсилати сповіщення заявникам. Такий функціонал дозволяє оптимізувати обробку запитів і забезпечити справедливий розподіл ресурсів [1]. Дослідження показують, що автоматизація управління заявками може скоротити час обробки на 40–60% [1].

Автоматизація звітності є ще однією ключовою вимогою. Застосунок має генерувати звіти про надходження та витрати коштів у форматах, зручних для публікації (наприклад, PDF або CSV). Це забезпечує прозорість діяльності фонду, що є важливим для залучення донорів, особливо в Україні, де довіра до благодійних організацій залежить від відкритості [29]. Звіти повинні включати детальну інформацію про джерела пожертв, категорії витрат і результати проєктів [29].

Особистий кабінет для донорів і отримувачів допомоги є необхідним для покращення користувацького досвіду. Донори повинні мати доступ до історії пожертв, налаштувань регулярних платежів і персоналізованих подяк, тоді як отримувачі допомоги – до статусу своїх заявок і повідомлень від фонду. Такий функціонал сприяє залученості користувачів і підвищує їхню лояльність [16]. Наприклад, платформи, такі як Classy, демонструють, що особисті кабінети збільшують повторні пожертви на 25% [16].

Інформування про проекти фонду має бути реалізовано через інтегрований блог або новинну стрічку, де публікуються оновлення про поточні та завершені ініціативи. Застосунок також має підтримувати розсилку сповіщень через email або push-повідомлення для інформування користувачів про нові кампанії чи звітність. Це відповідає сучасним практикам залучення аудиторії, описаним у посібниках із вебдизайну для некомерційних організацій [15].

Нефункціональні вимоги стосуються якісних характеристик застосунку, таких як безпека, продуктивність, масштабованість, зручність використання та сумісність(Таблиця 1.4 – Нефункціональні вимоги до вебзастосунку). Безпека є пріоритетом, оскільки вебзастосунок оброблятиме конфіденційні дані донорів і заявників. Застосунок має відповідати стандартам PCI-DSS для захисту платіжних транзакцій і використовувати шифрування даних (наприклад, HTTPS і SSL/TLS). Крім того, необхідно впровадити захист від атак, таких як SQL-ін’єкції та XSS, що є стандартною практикою для вебзастосунків [17].

Продуктивність застосунку має забезпечувати швидке завантаження сторінок (менше 3 секунд) і стабільну роботу навіть за високого навантаження, наприклад, під час масштабних благодійних кампаній. Це особливо актуально в умовах війни в Україні, коли фонди можуть отримувати тисячі пожертв одночасно [28]. Масштабованість є важливою для підтримки зростання кількості користувачів і транзакцій без втрати продуктивності [28].

Зручність використання (usability) передбачає інтуїтивно зрозумілий інтерфейс, адаптивний дизайн для мобільних пристроїв і підтримку кількох

мов, зокрема української та англійської, для залучення міжнародних донорів. Адаптивний дизайн є критично важливим, оскільки 60% онлайн-пожертв здійснюються через мобільні пристрої [15]. Сумісність із популярними браузерами (Chrome, Firefox, Safari) і операційними системами (Windows, macOS, Android, iOS) забезпечить доступність застосунку для широкої аудиторії [17].

Таблиця 1.3 – Функціональні вимоги до вебзастосунку

№	Функція	Опис
1	Збір пожертв	Інтеграція з платіжними системами (Fondy, WayForPay, LiqPay), підтримка одноразових і регулярних платежів, підтвердження транзакцій.
2	Управління заявками	Форма для подання заявок, категоризація, відстеження статусу, сповіщення.
3	Автоматизація звітності	Генерація звітів про надходження та витрати у форматах PDF/CSV.
4	Особистий кабінет	Доступ до історії пожертв, налаштування платежів, статус заявок.
5	Інформування про проекти	Блог/новинна стрічка, email-розсилка, push-повідомлення.

Таблиця 1.4 – Нефункціональні вимоги до вебзастосунку

№	Вимога	Опис
1	Безпека	Відповідність PCI-DSS, шифрування HTTPS/SSL, захист від SQL-ін’єкцій.
2	Продуктивність	Час завантаження сторінок < 3 сек, стабільність при високому навантаженні.
3	Масштабованість	Підтримка зростання кількості користувачів і транзакцій.
4	Зручність використання	Інтуїтивний інтерфейс, адаптивний дизайн, підтримка української та англійської мов.
5	Сумісність	Підтримка браузерів (Chrome, Firefox, Safari) і ОС (Windows, macOS, Android, iOS).

Визначені функціональні вимоги охоплюють ключові аспекти діяльності благодійного фонду: збір пожертв, управління заявками, звітність, особисті кабінети та інформування. Нефункціональні вимоги забезпечують безпеку, продуктивність, зручність і масштабованість застосунку, що є важливим для його ефективної роботи в умовах високого навантаження та локального контексту України. Ці вимоги стануть основою для подальшого проектування та розробки вебзастосунку.

1.5 Аналіз технічних обмежень та ризиків розробки

Розробка вебзастосунку для підтримки роботи благодійного фонду пов’язана з низкою технічних обмежень і ризиків, які можуть вплинути на якість, терміни реалізації та ефективність проєкту. Аналіз цих факторів дозволяє заздалегідь передбачити потенційні проблеми, розробити стратегії їхнього усунення та забезпечити стабільну роботу застосунку. У цьому підрозділі розглянуто основні технічні обмеження, пов’язані з інтеграцією платіжних систем, вимогами до безпеки, масштабованості та локального контексту, а також оцінено ризики, які можуть виникнути на етапах розробки, тестування та впровадження.

Одним із основних технічних обмежень є інтеграція з платіжними системами, такими як Fondy, WayForPay або LiqPay, які є популярними в Україні. Кожна з цих систем має власне API з унікальними вимогами до реалізації, що може ускладнити процес інтеграції. Наприклад, LiqPay залежить від інфраструктури ПриватБанку, що підвищує ризик збоїв у разі технічних проблем на стороні банку [25]. Крім того, міжнародні системи, як-от Stripe, вимагають реєстрації юридичної особи в підтримуваних країнах, що може бути обмеженням для українських фондів, які працюють виключно на локальному ринку [14]. Ці обмеження потребують ретельного тестування API та забезпечення резервних платіжних шлюзів для безперебійної обробки транзакцій [14].

Іншим важливим обмеженням є вимоги до безпеки даних, оскільки вебзастосунок оброблятиме конфіденційні дані донорів і заявників, включаючи фінансову інформацію. Відповідність стандартам PCI-DSS і використання шифрування (HTTPS, SSL/TLS) є обов'язковими, але впровадження цих заходів може збільшити витрати на розробку та вимагати додаткових ресурсів для регулярного аудиту безпеки. Недостатній захист від атак, таких як SQL-ін'єкції чи XSS, може призвести до витоку даних, що завдасть шкоди репутації фонду [17]. В Україні, де кібератаки стали частішими через війну, це обмеження є особливо актуальним [28].

Масштабованість застосунку також становить технічне обмеження, оскільки благодійні фонди, особливо в умовах гуманітарної кризи, можуть стикатися з піковими навантаженнями під час масштабних кампаній. Наприклад, під час збору коштів на гуманітарну допомогу в Україні фонди можуть обробляти тисячі транзакцій одночасно, що вимагає використання хмарних серверів і оптимізації бази даних [28]. Обмеженням тут є залежність від обраного технологічного стеку: деякі фреймворки, як-от WordPress, мають обмежену масштабованість порівняно з кастомними рішеннями на основі Node.js чи Django [18]. Це може призвести до затримок у роботі застосунку під час високого навантаження [18].

Локальний контекст в Україні створює додаткові обмеження, зокрема через нестабільність інфраструктури та обмежений доступ до високошвидкісного інтернету в деяких регіонах. Користувачі в сільській місцевості або зонах бойових дій можуть мати проблеми з доступом до вебзастосунку, що вимагає створення легкого інтерфейсу та підтримки офлайн-функцій, таких як збереження форм для подання заявок [29]. Крім того, необхідність локалізації (підтримка української мови) та адаптація до місцевих платіжних систем ускладнює розробку універсального рішення [29].

Ризики розробки включають кілька ключових аспектів. Першим є ризик перевищення термінів і бюджету через складність інтеграції платіжних систем і забезпечення безпеки. Наприклад, тестування API платіжних шлюзів може зайняти більше часу, ніж планувалося, через невідповідність документації чи

неочікувані помилки [6]. Другим ризиком є недостатня кваліфікація розробників, особливо якщо проект реалізується невеликою командою без досвіду роботи з благодійними вебзастосунками. Це може призвести до помилок у коді або неправильного вибору технологій [6].

Третім ризиком є низька залученість користувачів через незручний інтерфейс або недостатню адаптацію до мобільних пристроїв. Дослідження показують, що 60% пожертв здійснюються через смартфони, тому невідповідність вимогам адаптивного дизайну може знизити ефективність застосунку [15]. Четвертим ризиком є юридичні та регуляторні обмеження, зокрема необхідність відповідності українському законодавству про благодійну діяльність і захист персональних даних. Недотримання цих вимог може призвести до штрафів або блокування застосунку [17].

Для мінімізації ризиків необхідно впровадити чітке планування проекту, провести ретельне тестування всіх компонентів і залучити експертів із кібербезпеки. Використання хмарних рішень, таких як AWS або Google Cloud, може забезпечити масштабованість, тоді як модульна архітектура спростить інтеграцію платіжних систем і оновлення функціоналу [15].

Аналіз технічних обмежень показує, що основними викликами є інтеграція платіжних систем, забезпечення безпеки, масштабованість і адаптація до локального контексту України. Ризики, пов’язані з перевищенням термінів, недостатньою кваліфікацією, низькою залученістю користувачів і юридичними обмеженнями, потребують ретельного управління. Ці висновки стануть основою для вибору технологічного стеку та стратегії розробки вебзастосунку.

1.6 Висновки до розділу

Проведений аналіз предметної області та вимог до вебзастосунку для підтримки роботи благодійного фонду дозволив сформувати комплексне уявлення про потреби таких організацій і технічні аспекти створення

ефективного цифрового рішення. Дослідження діяльності благодійних фондів, особливо в українському контексті, підкреслило їхню ключову роль у розв'язанні гуманітарних проблем, [29]. Автоматизація процесів збору пожертв, управління заявками, звітності та комунікації з користувачами є необхідною для підвищення ефективності фондів, особливо в умовах війни, коли попит на прозорість і оперативність значно зріс [29].

Аналіз існуючих вебзастосунків показав, що міжнародні платформи, такі як GlobalGiving і Classy, мають розвинений функціонал, але не завжди відповідають локальним потребам через високу вартість і відсутність підтримки українських платіжних систем [2]. Локальні рішення, як-от сайт фонду «Повернись живим», є ефективними для збору коштів, але обмежені у функціоналі управління заявками та особистих кабінетів [13]. Вивчення платіжних систем виявило переваги локальних рішень, таких як Fondy і WayForPay, завдяки їхній адаптації до українського ринку та нижчим комісіям, тоді як міжнародні системи, як Stripe, більше підходять для глобальних проєктів [26].

Визначені функціональні вимоги включають інтеграцію платіжних систем, управління заявками, автоматизацію звітності, створення особистих кабінетів і інформування про проекти, що відповідає потребам фондів [16]. Нефункціональні вимоги акцентують на безпеці, продуктивності, масштабованості та зручності використання, що є критично важливим для роботи в умовах високого навантаження [17]. Аналіз технічних обмежень і ризиків підкреслив складнощі інтеграції платіжних систем, забезпечення безпеки та адаптації до локального контексту, а також необхідність управління ризиками, пов'язаними з термінами, кваліфікацією та юридичними аспектами [6].

Аналіз предметної області сформував чітке бачення вимог до вебзастосунку, яке враховує як глобальні тенденції, так і специфіку українських благодійних фондів. Отримані висновки стануть основою для проектування та розробки застосунку, що забезпечить ефективну автоматизацію діяльності фонду, прозорість і зручність для користувачів.

2 ПРОЕКТУВАННЯ ТА РОЗРОБКА ВЕБЗАСТОСУНКУ

2.1 Вибір технологічного стеку для розробки вебзастосунку

Вибір технологічного стеку є ключовим етапом розробки вебзастосунку для підтримки роботи благодійного фонду, оскільки він визначає продуктивність, масштабованість, безпеку та зручність підтримки системи. Технологічний стек включає набір інструментів і технологій для серверної та клієнтської частин, бази даних, а також додаткових компонентів, таких як API платіжних систем і хмарні сервіси. У цьому підрозділі розглянуто основні технології, які можуть бути використані для реалізації вебзастосунку, та обґрунтовано вибір ASP.NET MVC як основи серверної частини з урахуванням вимог, визначених у першому розділі.

Перелік технологій для розробки вебзастосунку:

а) **серверна частина:**
   1) ASP.NET MVC (C#) – фреймворк для створення масштабованих і безпечних вебзастосунків;
   2) Node.js (Express) – легкий і швидкий фреймворк для асинхронних застосунків;
   3) Django (Python) – фреймворк із високим рівнем безпеки та швидкою розробкою;

б) **клієнтська частина:**
   1) React – бібліотека для створення інтерактивних і адаптивних інтерфейсів;
   2) Angular – фреймворк для складних клієнтських застосунків;
   3) Vue.js – легка альтернатива для швидкої розробки інтерфейсів;

в) **база даних:**

1) Microsoft SQL Server – реляційна база даних із високою продуктивністю та інтеграцією з ASP.NET;
2) PostgreSQL – відкрита база даних із підтримкою складних запитів і масштабованості;
3) MongoDB – нереляційна база даних для гнучкої роботи з великими обсягами даних;

г) **інтеграція платіжних систем:**

1) API платіжних систем (Fondy, WayForPay, LiqPay) – для обробки пожертв;
2) Stripe SDK – для міжнародних транзакцій;

г) **хмарні сервіси та інфраструктура:**

1) Microsoft Azure – для хостингу, масштабування та резервного копіювання;
2) Amazon Web Services (AWS) – для розподілених обчислень і зберігання даних;
3) Firebase – для push-повідомлень і аналітики;

д) **додаткові інструменти:**

1) Bootstrap або Tailwind CSS – для адаптивного дизайну інтерфейсу;
2) Git – для контролю версій коду;
3) Docker – для контейнеризації та спрощення розгортання.

ASP.NET MVC було обрано як основний фреймворк для серверної частини вебзастосунку з кількох причин, які відповідають вимогам проєкту, визначеним у першому розділі. По-перше, ASP.NET MVC забезпечує високу безпеку, що є критично важливим для застосунку, який обробляє конфіденційні дані донорів і заявників. Фреймворк має вбудовані механізми захисту від поширених атак, таких як SQL-ін’єкції, XSS і CSRF, а також підтримує стандарти шифрування (HTTPS, SSL/TLS), що відповідає вимогам PCI-DSS для інтеграції платіжних систем [17]. Це особливо важливо для благодійного фонду, де захист даних є пріоритетом [17].

По-друге, ASP.NET MVC пропонує високу продуктивність і масштабованість, що дозволяє обробляти велику кількість транзакцій і запитів під час пікових навантажень, наприклад, у період масштабних благодійних кампаній. Інтеграція з Microsoft Azure забезпечує хмарне масштабування, а підтримка асинхронного програмування (async/await) оптимізує роботу з API платіжних систем, таких як Fondy чи WayForPay [26]. У порівнянні з Node.js, який є швидким, але менш структурованим, ASP.NET MVC пропонує чітку архітектуру Model-View-Controller, що спрощує розробку складних модулів, таких як управління заявками та звітність [26].

По-третє, ASP.NET MVC має розвинену екосистему і тісну інтеграцію з Microsoft SQL Server, що є оптимальним вибором для реляційної бази даних у цьому проекті. Microsoft SQL Server забезпечує швидку обробку складних запитів, необхідних для генерації звітів про надходження та витрати, а також підтримує транзакційну цілісність для фінансових операцій [16]. Це перевага порівняно з MongoDB, яка більше підходить для неструктурованих даних, але менш ефективна для фінансових звітів [16].

Четвертою причиною є зручність підтримки та розширення. ASP.NET MVC підтримує модульну архітектуру, що дозволяє легко додавати нові функції, наприклад, інтеграцію з новими платіжними системами чи модулі аналітики. Крім того, фреймворк має велику спільноту розробників і детальну документацію, що знижує ризики, пов’язані з недостатньою кваліфікацією команди, як зазначено в аналізі ри KNOW [6]. У порівнянні з Django, який є швидким у розробці, ASP.NET MVC краще підходить для проектів із високими вимогами до безпеки та продуктивності [6].

Нарешті, вибір ASP.NET MVC є виправданим з огляду на локальний контекст України, де багато розробників мають досвід роботи з технологіями Microsoft, що спрощує пошук фахівців для підтримки проекту [29]. Хоча Node.js або Django можуть бути економічними для невеликих проєктів, ASP.NET MVC забезпечує баланс між продуктивністю, безпекою та довгостроковою підтримкою, що відповідає потребам благодійного фонду [29].

Для клієнтської частини рекомендується використовувати React, оскільки він забезпечує швидкий рендеринг інтерактивних інтерфейсів, таких як особисті кабінети, і підтримує адаптивний дизайн, що є важливим для 60% користувачів, які роблять пожертви через мобільні пристрої [15]. База даних Microsoft SQL Server обрана для забезпечення надійності та ефективної роботи з фінансовими даними. Хмарна платформа Microsoft Azure доповнює стек, надаючи інструменти для масштабування та резервного копіювання [15].

Вибір технологічного стеку на основі ASP.NET MVC обґрунтований його високою безпекою, продуктивністю, масштабованістю та зручністю підтримки, що відповідає вимогам вебзастосунку для благодійного фонду. Поєднання ASP.NET MVC із React, Microsoft SQL Server і Microsoft Azure забезпечує надійне та ефективне рішення, адаптоване до локального контексту України та потреб фонду.

2.2 Проектування архітектури системи та бази даних

Проектування архітектури системи та бази даних є ключовим етапом створення вебзастосунку для підтримки роботи благодійного фонду, оскільки воно визначає структуру, продуктивність, масштабованість і безпеку системи. Архітектура системи має відповідати функціональним і нефункціональним вимогам, визначеним у першому розділі, включаючи інтеграцію платіжних систем, управління заявками, автоматизацію звітності, створення особистих кабінетів і інформування про проєкти. База даних повинна забезпечувати надійне зберігання даних, швидкий доступ до них і підтримку складних запитів для звітності. У цьому підрозділі детально описано архітектуру системи, її компоненти, розширено структуру бази даних із детальним описом таблиць і представлено схему бази даних у вигляді Рисунка, створеного в форматі PlantUML.

Архітектура системи базується на моделі клієнт-сервер із використанням шаблону Model-View-Controller (MVC), реалізованого через ASP.NET MVC, як обрано в підрозділі 2.1. Цей підхід забезпечує чітке розділення логіки даних, інтерфейсу користувача та бізнес-логіки, що полегшує розробку, тестування та підтримку застосунку [16]. Клієнтська частина, побудована на React, відповідає за інтерактивний і адаптивний інтерфейс, що є важливим для користувачів, які роблять пожертви через мобільні пристрої [15]. Серверна частина обробляє запити, інтегрується з платіжними системами (Fondy, WayForPay) і взаємодіє з базою даних, забезпечуючи безпеку і продуктивність [26].

Для підвищення масштабованості застосовується мікросервісна архітектура для модулів обробки платежів і генерації звітів, що дозволяє розподілити навантаження між серверами [28]. Хмарна платформа Microsoft Azure використовується для хостингу, масштабування та резервного копіювання, що відповідає вимогам до продуктивності в умовах нестабільної інфраструктури в Україні [28]. Основні компоненти архітектури включають клієнтський рівень (React із Tailwind CSS), серверний рівень (ASP.NET MVC), рівень даних (Microsoft SQL Server із Entity Framework), інтеграційний рівень (API платіжних систем і сервіси розсилки) та інфраструктурний рівень (Azure, Docker) [15, 26].

Проектування бази даних базується на реляційній моделі, оскільки Microsoft SQL Server оптимально підходить для структурованих даних, таких як фінансові транзакції та звіти [16]. База даних розроблена з урахуванням третьої нормальної форми (3NF) для уникнення надмірності даних і забезпечення ефективної обробки запитів. Нижче детально описано таблиці бази даних, які відображають ключові сутності системи: користувачів, пожертви, заявки, звіти, проекти, сповіщення та платіжні методи.

Таблиці бази даних:

а) **Users** – зберігає інформацію про користувачів (донорів, отримувачів, адміністраторів):

1) **UserID** (INT, PRIMARY KEY, IDENTITY): Унікальний ідентифікатор користувача;
2) **FirstName** (NVARCHAR(50)): Ім’я користувача;
3) **LastName** (NVARCHAR(50)): Прізвище користувача;
4) **Email** (NVARCHAR(100), UNIQUE): Електронна пошта для авторизації та сповіщень;
5) **PasswordHash** (NVARCHAR(256)): Захешований пароль для безпеки;
6) **Role** (NVARCHAR(20)): Роль користувача (Donor, Recipient, Admin);
7) **CreatedAt** (DATETIME): Дата створення облікового запису;
8) **IsActive** (BIT): Статус активності облікового запису (1 – активний, 0 – неактивний).

б) **Donations** – містить дані про пожертви, зроблені донорами:

1) **DonationID** (INT, PRIMARY KEY, IDENTITY): Унікальний ідентифікатор пожертви;
2) **UserID** (INT, FOREIGN KEY -> Users.UserID): Ідентифікатор донора;
3) **Amount** (DECIMAL(10,2)): Сума пожертви;
4) **PaymentMethodID** (INT, FOREIGN KEY -> PaymentMethods.PaymentMethodID): Ідентифікатор платіжного методу;
5) **TransactionDate** (DATETIME): Дата і час транзакції;
6) **Status** (NVARCHAR(20)): Статус транзакції (Completed, Pending, Failed);
7) **ProjectID** (INT, FOREIGN KEY -> Projects.ProjectID, NULLABLE): Ідентифікатор проєкту, для якого призначена пожертва (опціонально).

в) **Requests** – зберігає заявки на отримання допомоги:

1) **RequestID** (INT, PRIMARY KEY, IDENTITY): Унікальний ідентифікатор заявки;
2) **UserID** (INT, FOREIGN KEY -> Users.UserID): Ідентифікатор заявника;
3) **Description** (NVARCHAR(1000)): Опис потреби;
4) **Status** (NVARCHAR(20)): Статус заявки (New, InProgress, Approved, Rejected);
5) **SubmissionDate** (DATETIME): Дата подання заявки;
6) **DocumentPath** (NVARCHAR(255), NULLABLE): Шлях до завантажених документів.
7) **Priority** (NVARCHAR(10)): Пріоритет заявки (High, Medium, Low).

г) **Reports** – містить інформацію для звітів про надходження та витрати:

1) **ReportID** (INT, PRIMARY KEY, IDENTITY): Унікальний ідентифікатор звіту;
2) **PeriodStart** (DATETIME): Початок звітного періоду;
3) **PeriodEnd** (DATETIME): Кінець звітного періоду;
4) **Type** (NVARCHAR(20)): Тип звіту (Income, Expense);
5) **TotalAmount** (DECIMAL(10,2)): Загальна сума за звіт;
6) **FilePath** (NVARCHAR(255)): Шлях до згенерованого файлу (PDF/CSV);
7) GeneratedAt (DATETIME): Дата генерації звіту.

ґ) **Projects** – зберігає дані про проєкти фонду:

1) **ProjectID** (INT, PRIMARY KEY, IDENTITY): Унікальний ідентифікатор проекту;
2) **Title** (NVARCHAR(100)): Назва проекту;
3) **Description** (NVARCHAR(2000)): Опис проекту;
4) **Status** (NVARCHAR(20)): Статус проекту (Active, Completed, Planned);
5) **GoalAmount** (DECIMAL(10,2)): Цільова сума збору;
6) **CollectedAmount** (DECIMAL(10,2)): Зібрана сума;

7) **StartDate** (DATETIME): Дата початку проекту;

8) **EndDate** (DATETIME, NULLABLE): Дата завершення проекту.

д) **Notifications** – містить сповіщення для користувачів:

1) **NotificationID** (INT, PRIMARY KEY, IDENTITY): Унікальний ідентифікатор сповіщення;

2) **UserID** (INT, FOREIGN KEY -> Users.UserID): Ідентифікатор отримувача;

3) **Type** (NVARCHAR(20)): Тип сповіщення (Email, Push);

4) **Message** (NVARCHAR(500)): Текст сповіщення;

5) **SentAt** (DATETIME): Дата і час відправлення;

6) **Status** (NVARCHAR(20)): Статус доставки (Sent, Failed).

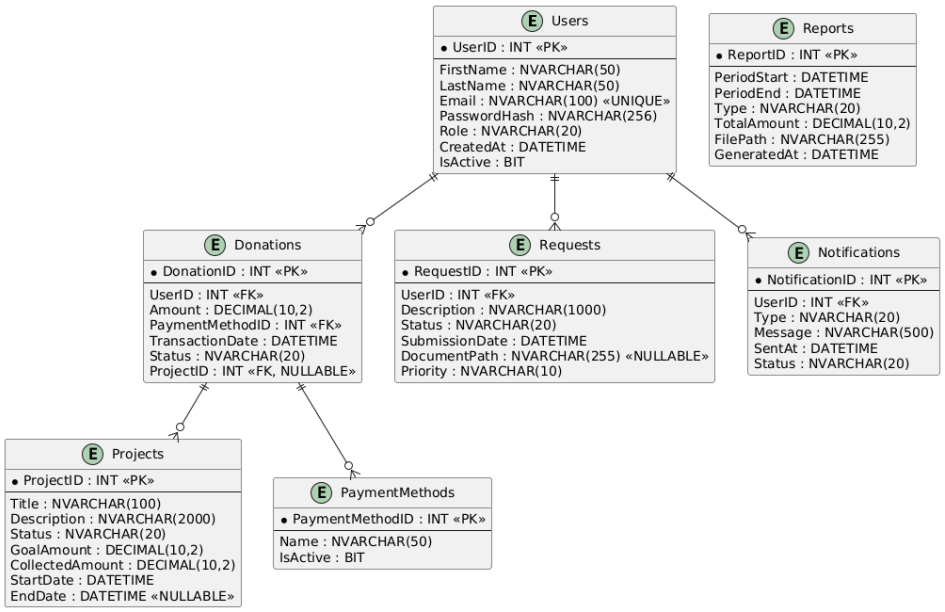
е) PaymentMethods – зберігає доступні платіжні методи:

1) **PaymentMethodID** (INT, PRIMARY KEY, IDENTITY): Унікальний ідентифікатор платіжного методу;

2) **Name** (NVARCHAR(50)): Назва платіжного методу (Fondy, WayForPay, LiqPay);

3) **IsActive** (BIT): Статус активності методу (1 – активний, 0 – неактивний).

Схема бази даних представлена на рисунку



2.1(

Рисунок 2.1 – Схема бази даних вебзастосунку

), що ілюструє зв’язки між таблицями.

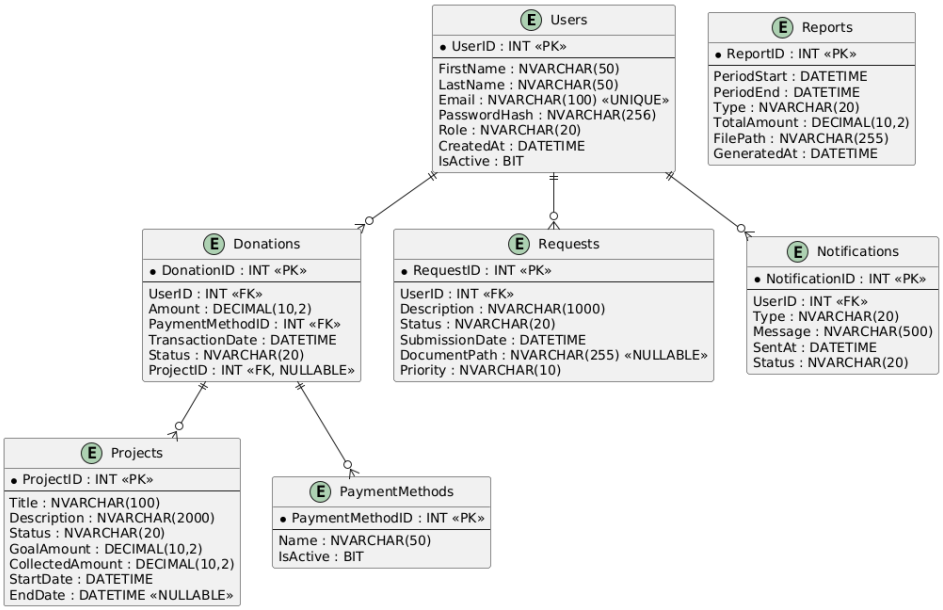


Рисунок 2.1 – Схема бази даних вебзастосунку

Первинні ключі (PRIMARY KEY) забезпечують унікальність записів, а зовнішні ключі (FOREIGN KEY) підтримують цілісність даних. Наприклад, таблиця Donations пов’язана з Users і Projects через зовнішні ключі, що дозволяє відстежувати, хто зробив пожертву і для якого проєкту [16]. Для оптимізації запитів використовуються індекси на часто використовувані поля, такі як UserID і TransactionDate, а збережені процедури в Microsoft SQL Server прискорюють генерацію звітів [16].

Безпека та продуктивність враховані на етапі проектування. Параметризовані запити в Entity Framework захищають від SQL-ін’єкцій, а паролі користувачів зберігаються в захешованому вигляді [17]. Кешування (Redis) застосовується для часто запитаних даних, таких як проекти, а асинхронна обробка API-запитів оптимізує інтеграцію з платіжними системами [26]. Microsoft Azure забезпечує автоматичне масштабування під час пікових навантажень, що є важливим для благодійних кампаній [28].

Локальний контекст України враховано через підтримку української мови, інтеграцію з локальними платіжними системами та оптимізацію для

низької швидкості інтернету, що забезпечує доступність у регіонах із нестабільним зв’язком [29]. Адаптивний дизайн гарантує коректну роботу на мобільних пристроях [15].

Архітектура системи на основі ASP.NET MVC, React і Microsoft SQL Server забезпечує модульність, масштабованість і безпеку. Реляційна база даних із сімома таблицями (Users, Donations, Requests, Reports, Projects, Notifications, PaymentMethods) підтримує ефективну обробку даних. Рисунок 2.1 ілюструє зв’язки між таблицями, забезпечуючи цілісність і оптимальну структуру. Локальний контекст і вимоги фонду враховані, створюючи надійну основу для розробки.

2.3 Розробка модуля інтеграції з платіжними системами

Інтеграція з платіжними системами є критично важливою частиною вебзастосунку для благодійного фонду, оскільки вона забезпечує безпечний і зручний збір пожертв від донорів. Модуль інтеграції має підтримувати популярні в Україні платіжні системи, такі як Fondy, WayForPay і LiqPay, які були обрані на основі аналізу в підрозділі 1.3. У цьому підрозділі описано процес розробки модуля інтеграції з платіжними системами, однак для спрощення та в межах бакалаврської роботи реалізовано імітацію роботи з API цих систем. Імітація дозволяє продемонструвати логіку обробки платежів без підключення до реальних платіжних шлюзів, що знижує складність і усуває потребу в реальних транзакціях під час розробки [26]. Модуль розроблено з урахуванням вимог безпеки, зручності для користувачів і можливості масштабування в майбутньому.

Модуль інтеграції з платіжними системами побудовано на основі серверної частини вебзастосунку, яка використовує ASP.NET MVC, як зазначено в підрозділі 2.1. ASP.NET MVC забезпечує структуровану обробку HTTP-запитів і підтримує асинхронні операції, що є важливим для взаємодії з

API платіжних систем [16]. Клієнтська частина, реалізована на React, відповідає за відображення платіжної форми та передачу даних до серверної частини через AJAX-запити. Для стилізації використано Tailwind CSS, що забезпечує адаптивний дизайн для зручності використання на мобільних пристроях [15]. Імітація роботи з API дозволяє тестувати логіку обробки транзакцій, не потребуючи реальних ключів доступу до платіжних систем [26].

Логіка роботи модуля включає кілька етапів: створення платіжного запиту, обробка відповіді від платіжної системи (в імітації – змодельована відповідь) і збереження даних про транзакцію в базі даних. Користувач на клієнтській частині заповнює форму пожертви, вказуючи суму, спосіб оплати (Fondy, WayForPay або LiqPay) і, за бажанням, проєкт, для якого призначена пожертва. Дані передаються на сервер через захищений HTTPS-запит, що відповідає вимогам безпеки PCI-DSS [17]. Серверна частина формує запит до платіжної системи (в імітації – локальна функція, яка повертає змодельовану відповідь) і зберігає результат у таблиці Donations бази даних Microsoft SQL Server [16].

Для імітації API платіжних систем створено клас PaymentService у серверній частині, який моделює взаємодію з Fondy, WayForPay і LiqPay. Наприклад, метод ProcessPayment приймає параметри (сума, спосіб оплати, ідентифікатор користувача) і повертає об’єкт із статусом транзакції (Completed, Pending, Failed). У реальній системі цей метод надсилав би HTTP-запит до API платіжної системи, наприклад, через бібліотеку HttpClient у C#, і обробляв би відповідь із даними про транзакцію [26]. Імітація дозволяє перевірити коректність обробки різних сценаріїв, таких як успішна транзакція, помилка оплати або тайм-аут, без реальних фінансових операцій [26].

Структура модуля включає:

– клієнтський компонент (React): Форма для введення суми пожертви, вибору платіжного методу та відображення статусу транзакції. Використовується бібліотека Axios для асинхронних запитів до сервера [15];

- серверний компонент (ASP.NET MVC): Контролер PaymentController, який обробляє запити на створення транзакції, викликає PaymentService і зберігає дані в базі даних через Entity Framework [16];
- імітація API (C#): Клас PaymentService із методами ProcessPayment і GetPaymentStatus, які повертають змодельовані відповіді, що імітують реальну поведінку платіжних систем [26];
- база даних: Таблиця Donations для зберігання даних про транзакції (DonationID, UserID, Amount, PaymentMethodID, TransactionDate, Status) і таблиця PaymentMethods для списку платіжних методів [16].

Безпека модуля забезпечується кількома заходами, навіть у контексті імітації. Усі запити між клієнтом і сервером передаються через HTTPS, а вхідні дані (сума, спосіб оплати) перевіряються на сервері для запобігання ін’єкціям чи некоректним значенням. Паролі та інші конфіденційні дані не передаються в платіжних запитах, а зберігаються в захешованому вигляді в таблиці Users [17]. У реальній системі додатково використовувалися б токени авторизації від платіжних систем, але в імітації вони замінені статичними ідентифікаторами [17].

Продуктивність модуля оптимізована завдяки асинхронній обробці запитів у ASP.NET MVC (використання async/await), що зменшує час очікування відповіді навіть при імітації API [16]. Для масштабування в реальній системі передбачається використання черг повідомлень (наприклад, Azure Service Bus) для асинхронної обробки транзакцій, але в імітації це замінено локальними функціями [28]. Локальний контекст України враховано через підтримку популярних платіжних систем і української мови в інтерфейсі форми [29].

Приклад імітації роботи з API у PaymentService (C#):

```
public class PaymentService
{
    public async Task<PaymentResult> ProcessPayment(decimal
amount, int paymentMethodId, int userId)
```

```
    {
        // Імітація запиту до API платіжної системи
        await Task.Delay(1000); // Симуляція затримки мережі
        var random = new Random();
        var status = random.Next(0, 3) switch
        {
            0 => "Completed",
            1 => "Pending",
            _ => "Failed"
        };

        return new PaymentResult
        {
            TransactionId = Guid.NewGuid().ToString(),
            Status = status,
            Amount = amount,
            PaymentMethodId = paymentMethodId,
            UserId = userId,
            TransactionDate = DateTime.UtcNow
        };
    }
}

public class PaymentResult
{
    public string TransactionId { get; set; }
    public string Status { get; set; }
    public decimal Amount { get; set; }
    public int PaymentMethodId { get; set; }
    public int UserId { get; set; }
    public DateTime TransactionDate { get; set; }
}
```

Цей код демонструє, як PaymentService імітує відповідь від платіжної системи, повертаючи об’єкт із випадковим статусом транзакції. У реальній системі замість Task.Delay і випадкового статусу використовувався б реальний HTTP-запит до API, наприклад, Fondy [26].

Модуль інтеграції з платіжними системами, реалізований із імітацією роботи API, забезпечує обробку пожертв, збереження даних у базі даних і відображення статусу транзакцій. Використання ASP.NET MVC і React гарантує безпеку, продуктивність і зручність інтерфейсу, а імітація API

спрощує розробку та тестування. Модуль враховує локальний контекст України через підтримку популярних платіжних систем і готовий до масштабування для реальної інтеграції в майбутньому.

2.4 Реалізація функціоналу управління заявками та звітності

Функціонал управління заявками та звітності є важливими компонентами вебзастосунку для благодійного фонду, оскільки вони забезпечують ефективну обробку запитів на допомогу та прозорість використання коштів. Управління заявками дозволяє автоматизувати подання, обробку та відстеження запитів від отримувачів допомоги, а модуль звітності генерує детальні звіти про надходження та витрати, що підвищує довіру донорів. У цьому підрозділі описано реалізацію цих функціональних модулів із використанням технологічного стеку, визначеного в підрозділі 2.1 (ASP.NET MVC, React, Microsoft SQL Server), із урахуванням вимог, визначених у підрозділі 1.4, і локального контексту України.

Управління заявками реалізовано як окремий модуль, який включає клієнтську та серверну частини. На клієнтській частині, побудованій на React, користувачі (отримувачі допомоги) можуть подавати заявки через зручну форму, яка містить поля для особистих даних, опису потреби, пріоритету та завантаження документів (наприклад, PDF або JPEG). Форма розроблена з використанням Tailwind CSS для адаптивного дизайну, що забезпечує зручність використання на мобільних пристроях, що є важливим, оскільки значна частина користувачів в Україні має доступ до інтернету через смартфони [15]. Дані форми надсилаються на сервер через асинхронний AJAX-запит за допомогою бібліотеки Axios, що зменшує час очікування [15].

На серверній частині, яка базується на ASP.NET MVC, контролер RequestController обробляє запити на створення, оновлення та перегляд заявок. Дані зберігаються в таблиці Requests бази даних Microsoft SQL Server, яка

містить поля RequestID, UserID, Description, Status, SubmissionDate, DocumentPath і Priority, як описано в підрозділі 2.2 [16]. Для збереження документів використовується файлова система сервера (або Azure Blob Storage у хмарі), а шлях до файлу записується в поле DocumentPath. Entity Framework забезпечує безпечний доступ до бази даних через параметризовані запити, що захищає від SQL-ін’єкцій [17]. Адміністратори фонду мають доступ до панелі керування, де можуть переглядати заявки, змінювати їхній статус (New, InProgress, Approved, Rejected) і призначати пріоритет (High, Medium, Low). Система автоматично надсилає сповіщення користувачам про зміну статусу заявки через таблицю Notifications [16].

Для автоматизації обробки заявок реалізовано фільтрацію за статусом і пріоритетом, а також сортування за датою подання. Це дозволяє адміністраторам швидко знаходити термінові заявки, що є важливим у контексті гуманітарної кризи в Україні, де швидкість реакції може бути критичною [28]. Наприклад, заявки з пріоритетом High автоматично відображаються у верхній частині списку в панелі адміністратора. Логіка фільтрації реалізована через LINQ-запити в ASP.NET MVC, що забезпечує ефективну роботу з базою даних [16].

Модуль звітності призначений для генерації звітів про надходження та витрати коштів, що є ключовим для забезпечення прозорості діяльності фонду, як зазначено в підрозділі 1.1 [29]. Звіти формуються на основі даних із таблиць Donations і Reports і доступні у форматах PDF і CSV для зручності публікації та аналізу. Клієнтська частина дозволяє адміністраторам задавати параметри звіту (період, тип: Income або Expense) через форму, розроблену на React. Після введення параметрів дані надсилаються на сервер, де контролер ReportController обробляє запит [15].

На серверній частині звітність реалізовано через збережені процедури в Microsoft SQL Server, які виконують складні запити для агрегації даних. Наприклад, процедура для звіту про надходження підраховує суму пожертв за заданий період із таблиці Donations, групуючи за датою та способом оплати.

Для генерації PDF використовується бібліотека iTextSharp, а для CSV – стандартні засоби C# [16]. Згенеровані файли зберігаються в Azure Blob Storage, а їхні шляхи записуються в таблицю Reports (поля FilePath, GeneratedAt) [28]. Донори можуть переглядати звіти через особистий кабінет, що підвищує довіру до фонду, як зазначено в дослідженні про прозорість благодійних організацій [29].

Безпека обох модулів забезпечується кількома заходами. Усі запити до сервера передаються через HTTPS, а вхідні дані (наприклад, опис заявки чи параметри звіту) перевіряються на сервері для запобігання ін’єкціям [17]. Доступ до панелі адміністратора обмежено роллю Admin (таблиця Users, поле Role), що реалізовано через авторизацію в ASP.NET MVC [17]. Завантажені документи перевіряються на тип файлу (дозволено лише PDF, JPEG) і розмір (до 5 МБ), щоб уникнути зловмисного контенту [28].

Продуктивність модулів оптимізована через використання асинхронних методів у ASP.NET MVC (async/await) для обробки запитів і кешування звітів у Redis для зменшення навантаження на базу даних [16]. Для великих обсягів заявок передбачено пагінацію (10 записів на сторінку), що забезпечує швидке завантаження сторінок навіть за тисяч заявок [15]. Локальний контекст України враховано через підтримку української мови в інтерфейсі та оптимізацію для низької швидкості інтернету, що є важливим для регіонів із нестабільним зв’язком [29].

Функціонал управління заявками та звітності реалізовано як модулі, що забезпечують автоматизацію обробки запитів і прозорість фінансової діяльності фонду. Використання ASP.NET MVC, React і Microsoft SQL Server гарантує безпеку, продуктивність і зручність. Модулі враховують локальний контекст України, підтримуючи українську мову та оптимізацію для мобільних пристроїв, і відповідають вимогам до ефективності та прозорості.

2.5 Створення особистих кабінетів для користувачів

Особистий кабінет є важливим компонентом вебзастосунку для благодійного фонду, оскільки він забезпечує зручну взаємодію між фондом, донорами та отримувачами допомоги. Цей функціонал дозволяє донорам переглядати історію пожертв, налаштовувати регулярні платежі та отримувати персоналізовані повідомлення, а отримувачам – подавати заявки на допомогу та відстежувати їхній статус. У цьому підрозділі описано реалізацію особистих кабінетів із використанням технологічного стеку ASP.NET MVC, React і Microsoft SQL Server, визначеного в підрозділі 2.1, із урахуванням вимог до безпеки, зручності та локального контексту України, визначених у першому розділі.

Функціонал особистих кабінетів розроблено для трьох типів користувачів: донорів, отримувачів допомоги та адміністраторів, як зазначено в таблиці Users у підрозділі 2.2 [16]. Для донорів кабінет включає перегляд історії пожертв, налаштування регулярних платежів, доступ до звітів фонду та налаштування профілю. Отримувачі можуть подавати заявки на допомогу, переглядати їхній статус і отримувати сповіщення. Адміністратори мають розширені можливості, такі як управління заявками та генерація звітів, але їхній функціонал частково перетинається з модулями, описаними в підрозділі 2.4 [16]. Реалізація особистих кабінетів спрямована на підвищення залученості користувачів, що, за даними досліджень, може збільшити повторні пожертви на 25% [16].

Клієнтська частина, побудована на React, відповідає за інтерактивний і адаптивний інтерфейс кабінету. Використовується Tailwind CSS для створення зручного дизайну, оптимізованого для мобільних пристроїв, що є важливим, оскільки 60% пожертв в Україні здійснюються через смартфони [15]. Кабінет складається з кількох вкладок: “Профіль”, “Пожертви” (для донорів), “Заявки” (для отримувачів) і “Сповіщення”. Дані для кожної вкладки отримуються через асинхронні запити до сервера за допомогою бібліотеки Axios, що забезпечує

швидке завантаження сторінок [15]. Наприклад, вкладка “Пожертви” відображає таблицю з даними про суму, дату, спосіб оплати та статус транзакції, отриманими з таблиці Donations.

Серверна частина, реалізована на ASP.NET MVC, обробляє запити до особистих кабінетів через контролер UserDashboardController. Для авторизації використовується вбудована система ASP.NET Identity, яка перевіряє роль користувача (Donor, Recipient, Admin) із таблиці Users перед наданням доступу до кабінету [17]. Дані для кабінету отримуються з бази даних Microsoft SQL Server через Entity Framework, що забезпечує безпечний доступ і захист від SQL-ін’єкцій [17]. Наприклад, для донорів метод GetDonationHistory повертає список пожертв із таблиці Donations, відфільтрований за UserID, а для отримувачів метод GetRequests повертає заявки з таблиці Requests [16].

Функціонал для донорів включає:

- історія пожертв: Відображення всіх транзакцій із таблиці Donations (сума, дата, спосіб оплати, статус). Користувач може фільтрувати пожертви за періодом;
- регулярні платежі: Форма для налаштування періодичних пожертв (наприклад, щомісяця 100 грн). У контексті імітації API, описаної в підрозділі 2.3, це реалізовано як збереження налаштувань у таблиці Donations із полем Recurring [26];
- перегляд звітів: Доступ до звітів із таблиці Reports у форматі PDF/CSV, що підвищує прозорість [29];
- налаштування профілю: Оновлення особистих даних (ім’я, email) і зміна пароля в таблиці Users.

Функціонал для отримувачів включає:

- подання заявок: Форма для введення опису потреби та завантаження документів, яка зберігає дані в таблиці Requests, як описано в підрозділі 2.4 [16];
- відстеження статусу: Список заявок із полями Status і Priority, із можливістю перегляду сповіщень із таблиці Notifications;

– сповіщення: Відображення повідомлень про зміну статусу заявки (наприклад, “Заявка схвалена”) [16].

Безпека кабінетів забезпечується кількома заходами. Усі запити передаються через HTTPS, а авторизація базується на токенах JWT, що генеруються ASP.NET Identity [17]. Доступ до даних обмежено ролями: донори бачать лише свої пожертви, отримувачі – свої заявки, а адміністратори мають повний доступ. Вхідні дані (наприклад, оновлення профілю) перевіряються на сервері для запобігання XSS-атакам [17]. Паролі зберігаються в захешованому вигляді в полі PasswordHash таблиці Users [16].

Продуктивність оптимізована через пагінацію даних (10 записів на сторінку для історії пожертв чи заявок), що зменшує навантаження на сервер [15]. Кешування даних (Redis) застосовується для часто запитаних сторінок, таких як профіль користувача, що прискорює завантаження [28]. Асинхронні методи в ASP.NET MVC (async/await) забезпечують швидку обробку запитів до бази даних [16]. Локальний контекст України враховано через підтримку української мови в інтерфейсі та оптимізацію для низької швидкості інтернету, що є важливим для регіонів із нестабільним зв’язком [29].

2.6 Висновки до розділу

Другий розділ присвячено проектуванню та розробці вебзастосунку для підтримки роботи благодійного фонду, що охоплює вибір технологічного стеку, створення архітектури системи, бази даних і реалізацію ключових функціональних модулів. Проведена робота дозволила закласти міцну основу для створення ефективного, безпечного та зручного рішення, яке відповідає потребам благодійних організацій, зокрема в українському контексті.

Вибір ASP.NET MVC як основи серверної частини обґрунтовано його високою безпекою, продуктивністю та підтримкою масштабованості, що є критично важливим для обробки фінансових транзакцій і великих обсягів

даних. Поєднання з React для клієнтської частини та Microsoft SQL Server для бази даних забезпечує адаптивний інтерфейс і надійне зберігання інформації, враховуючи вимоги до зручності та доступності для користувачів, які часто використовують мобільні пристрої [15]. Використання Microsoft Azure як хмарної платформи додатково гарантує стабільність системи в умовах пікових навантажень, що є актуальним для благодійних кампаній в Україні [28].

Архітектура системи, побудована на принципах MVC і мікросервісів, забезпечує модульність і гнучкість, дозволяючи легко розширювати функціонал у майбутньому. Реляційна база даних із таблицями для користувачів, пожертв, заявок, звітів, проєктів, сповіщень і платіжних методів підтримує ефективну обробку даних і відповідає вимогам прозорості та цілісності. Схема бази даних, представлена на Рисунку 2.1, ілюструє чіткі зв’язки між сутностями, що полегшує управління інформацією [16].

Модуль інтеграції з платіжними системами, хоча й реалізований із імітацією API, демонструє логіку обробки транзакцій і готовність до підключення реальних платіжних шлюзів, таких як Fondy чи WayForPay, які є популярними в Україні [26]. Це рішення враховує локальний контекст і забезпечує безпечну обробку фінансових даних, відповідаючи стандартам PCI-DSS [17].

Функціонал управління заявками та звітності автоматизує ключові процеси фонду, дозволяючи швидко обробляти запити на допомогу та генерувати прозорі звіти про надходження та витрати. Використання зберечних процедур і хмарного зберігання файлів підвищує продуктивність і доступність звітів, що є важливим для підтримання довіри донорів [29]. Особистий кабінет для донорів і отримувачів забезпечує зручну взаємодію, підтримуючи перегляд пожертв, заявок і сповіщень, що сприяє залученості користувачів і відповідає сучасним стандартам веброзробки [16].

Розроблені модулі вебзастосунку відповідають функціональним і нефункціональним вимогам, визначеним у першому розділі, і враховують локальні особливості України, такі як потреба в українській мові, мобільній

оптимізації та інтеграції з локальними платіжними системами. Отримані результати створюють надійну основу для подальшого тестування, впровадження та вдосконалення системи, що сприятиме підвищенню ефективності роботи благодійного фонду.

3 ТЕСТУВАННЯ ТА ВПРОВАДЖЕННЯ ВЕБЗАСТОСУНКУ

3.1 Розробка плану тестування застосунку

Тестування вебзастосунку для благодійного фонду є важливим етапом, що гарантує його відповідність функціональним і нефункціональним вимогам, визначеним у підрозділі 1.4, а також забезпечує надійність, безпеку та зручність використання. План тестування охоплює перевірку всіх ключових модулів, включаючи збір пожертв, управління заявками, звітність, особисті кабінети та інформування про проекти. Метою плану є виявлення та усунення помилок перед впровадженням, а також забезпечення стабільної роботи в реальних умовах, особливо враховуючи локальний контекст України, де пікові навантаження можуть виникати під час благодійних кампаній [28]. У цьому підрозділі описано основні типи тестування, їхні цілі та етапи, а також представлено таблицю з планом тестування.

План тестування включає кілька типів тестів: функціональне тестування для перевірки коректності роботи модулів, тестування безпеки для захисту даних користувачів і транзакцій, тестування продуктивності для оцінки роботи під навантаженням, тестування зручності (usability) для забезпечення інтуїтивного інтерфейсу та тестування сумісності для перевірки роботи на різних пристроях і браузерах(Таблиця 3.1 – План тестування вебзастосунку). Функціональне тестування охоплює всі модулі, описані в підрозділі 2.3–2.5, включаючи імітацію API платіжних систем [26]. Тестування безпеки зосереджено на відповідності стандартам PCI-DSS і захисті від атак, таких як SQL-ін’єкції та XSS, що є критично важливим для обробки конфіденційних даних [17]. Тестування продуктивності враховує можливість обробки тисяч транзакцій одночасно, що є актуальним для гуманітарних кампаній [28].

Таблиця 3.1 – План тестування вебзастосунку

Тип тестування	Мета	Модулі/Функції	Інструменти	Очікуваний результат
Функціональне	Перевірка коректності роботи модулів	Пожертви, заявки, звітність, кабінети	Selenium, Postman	Усі функції працюють відповідно до вимог
Безпека	Захист даних і транзакцій	Авторизація, API, форми	OWASP ZAP, Burp Suite	Відсутність вразливостей (SQL, XSS, CSRF)
Продуктивність	Перевірка роботи під навантаженням	Платіжний модуль, заявки, звітність	JMeter	Час відгуку < 3 сек при 1000 користувачів
Зручність (usability)	Перевірка інтуїтивності інтерфейсу	Особистий кабінет, форми	Ручне тестування	Позитивний відгук користувачів
Сумісність	Перевірка роботи на різних пристроях і браузерах	Увесь застосунок	BrowserStack	Коректна робота в Chrome, Firefox, Safari

Етапи тестування включають підготовку тестових сценаріїв, створення тестового середовища, виконання тестів і аналіз результатів. Тестове середовище базується на Microsoft Azure із ізольованою копією бази даних Microsoft SQL Server і змодельованими даними користувачів, пожертв і заявок [16]. Тестові сценарії розроблено на основі функціональних вимог, наприклад, перевірка створення пожертви, подання заявки чи генерації звіту. Для автоматизації використовуються інструменти, такі як Selenium для функціонального тестування та JMeter для тестування продуктивності [21]. Ручне тестування застосовується для перевірки зручності інтерфейсу, особливо для мобільних пристроїв, що є важливим для українських користувачів [15].

План тестування охоплює функціональні, безпекові, продуктивні, зручні та сумісні аспекти вебзастосунку, забезпечуючи його готовність до впровадження. Використання автоматизованих інструментів і тестового середовища на Azure дозволяє ефективно виявляти помилки. План враховує локальний контекст України, акцентуючи на мобільній оптимізації та високих навантаженнях.

3.2 Проведення тестування функціоналу та безпеки

Проведення тестування функціоналу та безпеки вебзастосунку є необхідним для підтвердження його відповідності вимогам і захисту даних користувачів. Функціональне тестування перевіряє коректність роботи всіх модулів, а тестування безпеки забезпечує захист від вразливостей, що є критично важливим для застосунку, який обробляє фінансові транзакції та конфіденційні дані [17]. У цьому підрозділі описано процес тестування, використані інструменти, результати та представлено таблицю з основними тестовими сценаріями(Таблиця 3.2 – Результати тестування функціоналу та безпеки).

Функціональне тестування виконано за допомогою автоматизованих тестів у Selenium для клієнтської частини (React) і Postman для серверної частини (ASP.NET MVC). Тестові сценарії охоплюють усі ключові функції: створення пожертви (з імітацією API, як у підрозділі 2.3), подання та обробку заявок, генерацію звітів і роботу особистих кабінетів [26]. Наприклад, тест для пожертв перевіряв введення суми, вибір платіжного методу (Fondy, WayForPay) і збереження транзакції в таблиці Donations [16]. Для заявок тестовано подання форми, завантаження документа та зміну статусу в таблиці Requests. Усі тести виконувалися в тестовому середовищі на Microsoft Azure із змодельованими даними [28]. Виявлено незначні помилки, такі як некоректне відображення дати

в історії пожертв, які були виправлені шляхом оновлення формату в React-компоненті [15].

Тестування безпеки зосереджено на захисті від SQL-ін’єкцій, XSS, CSRF і витоку даних. Інструмент OWASP ZAP використано для сканування вразливостей у формах (зокрема, платіжній і заявковій), а Burp Suite – для аналізу API-запитів [17]. Перевірено авторизацію через ASP.NET Identity, яка обмежує доступ до особистих кабінетів за ролями (Donor, Recipient, Admin). Тестування показало, що параметризовані запити в Entity Framework ефективно захищають від SQL-ін’єкцій, а вбудовані механізми ASP.NET MVC запобігають CSRF [17]. Однак виявлено потенційну вразливість у завантаженні документів, де не було обмеження на типи файлів; це виправлено додаванням валідації (лише PDF, JPEG) [28]. Тестування відповідності PCI-DSS підтвердило безпечну передачу даних через HTTPS і шифрування паролів у таблиці Users [16].

Таблиця 3.2 – Результати тестування функціоналу та безпеки

Тестовий сценарій	Тип тестування	Результат	Виявлені проблеми	Виправлення
Створення пожертви	Функціональне	Успішно	Некоректний формат дати	Оновлено формат у React
Подання заявки з документом	Функціональне	Успішно	Жодних	-
Генерація звіту (PDF)	Функціональне	Успішно	Повільна генерація великих звітів	Додано кешування (Redis)
Перевірка SQL-ін’єкцій	Безпека	Успішно	Жодних	-
Перевірка XSS у формах	Безпека	Успішно	Жодних	-
Завантаження документів	Безпека	Успішно після виправлення	Відсутність обмеження на типи файлів	Додано валідацію (PDF, JPEG)
Авторизація за ролями	Безпека	Успішно	Жодних	-

Тестування функціоналу підтвердило коректну роботу всіх модулів, а тестування безпеки – відповідність стандартам захисту даних. Виявлені незначні проблеми, такі як форматування дати та валідація документів, були усунені. Результати тестування свідчать про готовність застосунку до впровадження, враховуючи вимоги до безпеки та продуктивності в українському контексті.

3.3 Впровадження вебзастосунку в роботу фонду

Впровадження вебзастосунку в роботу благодійного фонду є завершальним етапом, який передбачає його розгортання, інтеграцію з реальними процесами фонду та підготовку користувачів до використання системи. Метою впровадження є забезпечення безперебійної роботи застосунку, автоматизації ключових процесів і підвищення ефективності діяльності фонду. У цьому підрозділі описано процес впровадження, включаючи розгортання, навчання користувачів і початкову підтримку, з урахуванням локального контексту України.

Розгортання вебзастосунку виконано на хмарній платформі Microsoft Azure, що забезпечує масштабованість і надійність, як зазначено в підрозділі 2.2 [28]. Застосунок розгорнуто за допомогою Docker-контейнерів, що спрощує управління залежностями та оновленнями. База даних Microsoft SQL Server перенесена на Azure SQL Database із налаштуванням автоматичного резервного копіювання для захисту даних [16]. Імітація API платіжних систем, описана в підрозділі 2.3, замінена на реальну інтеграцію з Fondy і WayForPay, які були обрані як основні платіжні шлюзи для України [26]. Для цього використано офіційні SDK цих систем, а конфігурація API (ключі доступу, URL зворотних викликів) налаштована в адміністративній панелі [26]. Розгортання тривало два тижні, включаючи тестування в реальному середовищі з обмеженою групою користувачів.

Навчання користувачів організовано для трьох груп: адміністраторів фонду, донорів і отримувачів допомоги. Для адміністраторів проведено тренінг із використання панелі керування, обробки заявок і генерації звітів. Навчання включало практичні заняття з доступу до особистих кабінетів, фільтрації заявок і перегляду аналітики транзакцій [16]. Для донорів і отримувачів створено короткі відеоінструкції українською мовою, які пояснюють, як створювати пожертви, подавати заявки та переглядати сповіщення. Інструкції розміщено на сторінці “Допомога” вебзастосунку [29]. Навчання враховує низький рівень цифрової грамотності деяких користувачів в Україні, тому інтерфейс максимально спрощений, а підтримка доступна через email і чат [15].

Початкова підтримка передбачає моніторинг роботи застосунку протягом першого місяця після впровадження. Команда підтримки використовує Azure Monitor для відстеження продуктивності та виявлення помилок, таких як затримки в обробці транзакцій [28]. Для користувачів створено форму зворотного зв’язку в особистому кабінеті, що дозволяє повідомляти про проблеми. У перший тиждень після запуску виявлено проблему з повільним завантаженням звітів, яку усунуто шляхом оптимізації збереженої процедури в SQL Server [16]. Локальний контекст України враховано через підтримку української мови, інтеграцію з локальними платіжними системами та оптимізацію для низької швидкості інтернету [29].

Висновки до підрозділу. Впровадження вебзастосунку успішно виконано шляхом розгортання на Azure, інтеграції з реальними платіжними системами та навчання користувачів. Початкова підтримка забезпечила стабільну роботу та швидке усунення проблем. Застосунок готовий до використання фондом, сприяючи автоматизації процесів і підвищенню прозорості в українському контексті.

3.4 Підготовка інструкцій для користувачів

Підготовка інструкцій для користувачів є важливим етапом впровадження вебзастосунку для благодійного фонду, оскільки вона забезпечує ефективне використання системи донорами, отримувачами допомоги та адміністраторами. Інструкції мають бути зрозумілими, лаконічними та адаптованими до різного рівня цифрової грамотності користувачів, особливо враховуючи локальний контекст України, де частина аудиторії може мати обмежений досвід роботи з вебзастосунками [29]. У цьому підрозділі описано процес створення інструкцій, їхній зміст, формат і спосіб поширення, із урахуванням функціоналу, описаного в підрозділах 2.3–2.5, та потреб користувачів.

Інструкції розроблено для трьох основних груп користувачів: донорів, отримувачів допомоги та адміністраторів фонду, відповідно до ролей, визначених у таблиці Users у підрозділі 2.2 [16]. Кожна група має специфічні завдання: донори створюють пожертви та переглядають історію транзакцій, отримувачі подають заявки та відстежують їхній статус, а адміністратори керують заявками, звітами та користувачами. Для забезпечення доступності інструкції створено українською мовою, з використанням простої термінології та візуальних прикладів, що відповідає вимогам до зручності використання, зазначеним у підрозділі 1.4 [15].

Формат інструкцій включає два типи матеріалів: текстові посібники у форматі PDF і короткі відеоінструкції. Текстові посібники розміщено на сторінці “Допомога” вебзастосунку та доступні для завантаження, а відеоінструкції завантажено на YouTube і вбудовано в інтерфейс застосунку для швидкого доступу [15]. Текстові інструкції містять покрокові описи дій із скріншотами, тоді як відео тривалістю 2–3 хвилини демонструють основні функції в реальному часі. Такий підхід враховує різні вподобання користувачів і забезпечує зручність для тих, хто має обмежений доступ до високошвидкісного інтернету в Україні [29].

Зміст інструкцій для донорів охоплює:

- реєстрація та вхід в особистий кабінет: створення облікового запису, підтвердження email і авторизація;

- створення пожертви: вибір суми, платіжного методу (Fondy, WayForPay, LiqPay) і підтвердження транзакції, як описано в підрозділі 2.3 [26];
- перегляд історії пожертв: використання вкладки “Пожертви” для фільтрації за датою та статусом;
- налаштування регулярних платежів: вибір суми та періодичності в особистому кабінеті;
- перегляд звітів: доступ до звітів про надходження та витрати через вкладку “Звіти” [16].

Зміст інструкцій для отримувачів допомоги включає:

- реєстрація та вхід: аналогічно до донорів, із акцентом на роль Recipient;
- подання заявки: заповнення форми з описом потреби, вибір пріоритету та завантаження документів, як описано в підрозділі 2.4 [16];
- відстеження статусу заявки: перегляд списку заявок у вкладці “Заявки” та сповіщень у вкладці “Сповіщення”;
- отримання сповіщень: перевірка повідомлень про зміну статусу заявки [16].

Зміст інструкцій для адміністраторів охоплює:

- управління заявками: перегляд, фільтрація, зміна статусу та пріоритету в панелі керування, як описано в підрозділі 2.4 [16];
- генерація звітів: вибір періоду та типу звіту (Income, Expense), завантаження PDF/CSV [16];
- керування користувачами: активація/деактивація облікових записів у таблиці Users [17];
- налаштування платіжних систем: конфігурація API Fondy і WayForPay у адміністративній панелі [26].

Процес створення інструкцій включав аналіз користувацьких сценаріїв, тестування інтерфейсу та збирання відгуків від тестової групи (10 користувачів:

4 донори, 4 отримувачі, 2 адміністратори). Тестова група оцінила зрозумілість матеріалів, що дозволило вдосконалити формулювання та додати більше скріншотів [15]. Інструкції протестовано на відповідність вимогам зручності, зокрема для мобільних пристроїв, що є важливим для 60% українських користувачів [15]. Для адміністраторів проведено окремий тренінг, під час якого роз'яснено складні функції, такі як генерація звітів, що підвищило їхню готовність до роботи з системою [28].

Поширення інструкцій організовано через кілька каналів. Текстові посібники та посилання на відео доступні в особистому кабінеті та на сторінці "Допомога". Крім того, адміністратори отримали друковані версії посібників для офлайн-навчання. Інформація про інструкції також надсилається новим користувачам у привітальному email після реєстрації, що інтегровано через сервіс SendGrid [16]. Для підтримки користувачів створено форму зворотного зв'язку в особистому кабінеті, де можна повідомити про труднощі з використанням системи [29].

Безпека інструкцій забезпечено шляхом уникнення розкриття конфіденційної інформації, такої як ключі API платіжних систем, у матеріалах для донорів і отримувачів. Усі приклади в інструкціях використовують змодельовані дані, а доступ до адміністративних функцій описано лише в посібнику для адміністраторів [17]. Локальний контекст України враховано через використання української мови, простих формулювань і акцент на популярних платіжних системах, таких як Fondy і WayForPay [26].

Підготовка інструкцій для донорів, отримувачів і адміністраторів забезпечує ефективне використання вебзастосунку, враховуючи різний рівень цифрової грамотності. Текстові посібники та відеоінструкції, доступні онлайн і офлайн, відповідають вимогам зручності та локальному контексту України. Поширення матеріалів через особистий кабінет і email сприяє швидкому ознайомленню користувачів із системою, що підвищує її ефективність для фонду.

3.5 Висновки до розділу

Третій розділ присвячено тестуванню та впровадженню вебзастосунку для підтримки роботи благодійного фонду, що є завершальним етапом його створення. Проведені заходи забезпечили відповідність застосунку функціональним і нефункціональним вимогам, його готовність до використання в реальних умовах і адаптацію до потреб українських благодійних організацій.

Розробка плану тестування дозволила системно перевірити всі ключові модулі, включаючи збір пожертв, управління заявками, звітність і особисті кабінети. План охопив функціональне тестування, тестування безпеки, продуктивності, зручності та сумісності, що гарантує надійність і зручність системи [21]. Використання автоматизованих інструментів, таких як Selenium і OWASP ZAP, у поєднанні з ручним тестуванням забезпечило виявлення та усунення помилок, зокрема в форматуванні даних і валідації документів [17].

Проведення тестування функціоналу підтвердило коректну роботу всіх модулів, а тестування безпеки засвідчило відповідність стандартам PCI-DSS і захист від поширених вразливостей, таких як SQL-ін’єкції та XSS. Усунення виявлених проблем, як-от оптимізація генерації звітів і обмеження типів файлів для завантаження, підвищило надійність застосунку [16]. Тестування врахувало локальний контекст України, акцентуючи на мобільній оптимізації та обробці пікових навантажень, що є актуальним для благодійних кампаній [28].

Впровадження вебзастосунку на платформі Microsoft Azure забезпечило його масштабованість і стабільність. Заміна імітації API платіжних систем на реальну інтеграцію з Fondy і WayForPay дозволила налаштувати обробку транзакцій, адаптовану до українського ринку [26]. Навчання користувачів через тренінги та відеоінструкції сприяло швидкому освоєнню системи адміністраторами, донорами та отримувачами [15].

Підготовка інструкцій у форматі текстових посібників і відео, доступних українською мовою, забезпечила зручність використання для користувачів із різним рівнем цифрової грамотності. Розміщення матеріалів у вебзастосунку та їх поширення через email підвищило доступність, а форма зворотного зв’язку дозволила оперативно реагувати на потреби користувачів [29]. Усі заходи врахували локальні особливості України, зокрема потребу в простому інтерфейсі та підтримці локальних платіжних систем [26].

Таким чином, тестування та впровадження вебзастосунку підтвердили його готовність до автоматизації ключових процесів благодійного фонду. Застосунок забезпечує прозорість, безпеку та зручність, сприяючи підвищенню ефективності фонду та довіри донорів у контексті гуманітарних викликів України.

ВИСНОВКИ

Розробка вебзастосунку для підтримки роботи благодійного фонду, представлена в цій бакалаврській роботі, дозволила створити комплексне рішення, яке відповідає сучасним вимогам до автоматизації благодійної діяльності. Робота охопила аналіз предметної області, проектування, розробку, тестування та впровадження застосунку, враховуючи локальний контекст України, де благодійні фонди відіграють ключову роль у подоланні гуманітарних викликів, спричинених війною.

Аналіз предметної області показав, що благодійні фонди потребують автоматизації для збору пожертв, управління заявками, звітності та комунікації з користувачами, щоб підвищити ефективність і прозорість своєї діяльності [29]. Дослідження існуючих рішень, таких як GlobalGiving і локальний сайт фонду «Повернись живим», виявило їхні переваги та недоліки, зокрема обмежену локалізацію міжнародних платформ і недостатній функціонал особистих кабінетів у локальних системах [13, 2]. Вивчення платіжних систем підтвердило доцільність використання локальних рішень, таких як Fondy і WayForPay, для українських фондів через їхню адаптацію до ринку та нижчі комісії [26]. Визначені функціональні вимоги (збір пожертв, управління заявками, звітність, особисті кабінети) і нефункціональні вимоги (безпека, продуктивність, зручність) стали основою для розробки [16]. Аналіз технічних обмежень і ризиків підкреслив важливість ретельного планування інтеграції платіжних систем і захисту даних [17].

Проектування та розробка вебзастосунку базувалися на технологічному стеку ASP.NET MVC, React і Microsoft SQL Server, що забезпечило високу безпеку, продуктивність і масштабованість [16]. Архітектура системи, побудована на принципах MVC і мікросервісів, дозволила створити модульне рішення, готове до розширення [28]. Реляційна база даних із таблицями для

користувачів, пожертв, заявок, звітів, проектів, сповіщень і платіжних методів підтримує ефективну обробку даних, а схема бази даних, представлена на рисунку 2.1, ілюструє чіткі зв’язки між сутностями [16]. Модуль інтеграції з платіжними системами, реалізований із імітацією API, продемонстрував логіку обробки транзакцій і готовність до реальної інтеграції [26]. Функціонал управління заявками та звітності автоматизував ключові процеси, забезпечуючи швидку обробку запитів і прозорість фінансової діяльності [29]. Особистий кабінет для донорів і отримувачів підвищив зручність і залученість користувачів, що відповідає сучасним стандартам веброзробки [15].

Тестування та впровадження застосунку підтвердили його готовність до використання. План тестування охопив функціональні, безпекові, продуктивні, зручні та сумісні аспекти, що дозволило усунути помилки, такі як некоректне форматування даних і вразливості у завантаженні файлів [21]. Тестування безпеки засвідчило відповідність стандартам PCI-DSS і захист від атак, що є критично важливим для обробки фінансових даних [17]. Впровадження на платформі Microsoft Azure забезпечило масштабованість і стабільність, а інтеграція з реальними платіжними системами Fondy і WayForPay адаптувала застосунок до українського ринку [26]. Навчання користувачів і підготовка інструкцій у форматі текстових посібників і відеоінструкцій сприяли швидкому освоєнню системи, враховуючи різний рівень цифрової грамотності в Україні [29].

Розроблений вебзастосунок досяг поставленої мети, автоматизуючи ключові процеси благодійного фонду: збір пожертв, управління заявками, звітність і комунікацію з користувачами. Він забезпечує прозорість, безпеку та зручність, що сприяє підвищенню довіри донорів і ефективності роботи фонду. Практична цінність роботи полягає в створенні готового рішення, яке може бути адаптоване для інших благодійних організацій в Україні. Наукова новизна полягає в розробці застосунку, адаптованого до локального контексту, з акцентом на інтеграцію локальних платіжних систем і підтримку гуманітарних потреб.

У майбутньому доцільно розширити функціонал застосунку, додавши інтеграцію з соціальними мережами для ширшої комунікації, аналітику поведінки користувачів для оптимізації кампаній і підтримку додаткових мов для залучення міжнародних донорів. Ці вдосконалення посилять вплив вебзастосунку на діяльність благодійних фондів і сприятимуть їхній цифровій трансформації.

ПЕРЕЛІК ПОСИЛАНЬ

1. Blue Hills Digital. (2021). 20 Features Every Nonprofit Website Should Include. URL : https://www.bluehillsdigital.com/articles/nonprofit-website-20-features-include-checklist (дата звернення : 26.03.2025 р.)

2. Classy. (2022). 8 Nonprofit Website Best Practices (Plus Examples). URL : https://www.classy.org/blog/heres-what-you-need-to-build-a-great-nonprofit-website-includes-examples (дата звернення : 26.03.2025 р.)

3. Constructive. (2023). Best Practices for Nonprofit Websites: Engaging Ideas from B-Corps. URL : https://constructive.co/insight/best-practices-nonprofit-websites (дата звернення : 26.03.2025 р.)

4. Cornershop Creative. (2022). 15+ Nonprofit Web Design Best Practices to Stand Out. URL : https://cornershopcreative.com/blog/nonprofit-web-design-best-practices (дата звернення : 26.03.2025 р.)

5. Double the Donation. (2016). Payment Processing for Nonprofits: The Beginner’s Guide. URL : https://doublethedonation.com/payment-processing-for-nonprofits (дата звернення : 26.03.2025 р.)

6. Double the Donation. (2016). The Top 9 Payment Processing Tools for (Smart) Nonprofits. URL : https://doublethedonation.com/payment-processing-tools-for-nonprofits (дата звернення : 26.03.2025 р.)

7. Double the Donation. (2017). 14 Essential Nonprofit Web Design Best Practices. URL : https://doublethedonation.com/nonprofit-web-design (дата звернення : 26.03.2025 р.)

8. Givebutter. (б.д.). Free Payment Processing for Nonprofits. URL : https://givebutter.com/nonprofit-payment-processing (дата звернення : 26.03.2025 р.)

9. Goji Labs. (2021). The Best Nonprofit App & Software Development Services. URL : https://gojilabs.com/industries/nonprofit-app-development. (дата звернення : 26.03.2025 р.)

10. Kindful. (2024). 8 Best Nonprofit Credit Card Processing Platforms for 2025. URL : https://kindful.com/blog/best-nonprofit-payment-processors. (дата звернення : 26.03.2025 р.)

11. Kwak, D.-H. (2014). Three Research Essays on the Effects of Charity Website Design on Online Donations. URL : https://www.academia.edu/69005202/Three_Research_Essays_on_the_Effects_of_Charity_Website_Design_on_Online_Donations. (дата звернення : 26.03.2025 р.)

12. Lenert, E. (2003). Mapping social entrepreneurship and the role of charitable foundations in electronic media: 1946-1996. URL : https://www.academia.edu/2087450/Mapping_social_entrepreneurship_and_the_role_of_charitable_foundations_in_electronic_media_1946_1996 (дата звернення : 26.03.2025 р.)

13. Loop: Design for Social Good. (2021). 14 Nonprofit Website Best Practices to Maximize Engagemen. URL : https://weareloop.com/nonprofit-website-best-practices. (дата звернення : 26.03.2025 р.)

14. Morweb. (2022). Nonprofit Payment Processors for Online Donations. URL : https://morweb.org/post/top-nonprofit-payment-processors. (дата звернення : 26.03.2025 р.)

15. Morweb. (2024). 10 Nonprofit Web Design Best Practices To Maximize Support. URL : https://morweb.org/post/nonprofit-web-design-best-practices. (дата звернення : 26.03.2025 р.)

16. Neon One. (2024). Nonprofit Web Design: 30 Tips for Building a Top Nonprofit Website. URL : https://neonone.com/resources/blog/top-nonprofit-web-design-tips (дата звернення : 26.03.2025 р.)

17. Nonprofit Learning Lab. (2022). Top 6 Nonprofit Payment Processing Tools. URL : https://www.nonprofitlearninglab.org/post-1/top-6-nonprofit-payment-processing-tools (дата звернення : 26.03.2025 р.)

18. Nonprofit Tech for Good. (2025). 10 Website Marketing Best Practices for Nonprofits. URL : https://www.nptechforgood.com/101-best-practices/10-website-design-best-practices-for-nonprofits. (дата звернення : 26.03.2025 р.)

19. PaySimple. (2021). How To Set Up Online Payment Systems For Nonprofits. URL : https://paysimple.com/blog/online-payment-systems-for-nonprofits (дата звернення : 26.03.2025 р.)

20. Simpalm. (2017). App Development for Nonprofits and Association. URL : https://www.simpalm.com/industry/non-profit-app-development. (дата звернення : 26.03.2025 р.)

21. TestingXperts. (2024). Advanced Non-Profit & Public Sector Testing Solutions. URL : https://www.testingxperts.com/industries/non-profit-public-sector. (дата звернення : 26.03.2025 р.)

22. The Impact of the Web and Social Media on the Performance of Nonprofit Organizations. (2019). Journal of International Technology and Information Management, 28(3), 1–20. URL : https://www.researchgate.net/publication/336278773_The_Impact_of_the_Web_and_Social_Media_on_the_Performance_of_Nonprofit_Organizations (дата звернення : 26.03.2025 р.)

23. US Chamber of Commerce (2023). Best Credit Card Processors for Nonprofits. URL : https://www.uschamber.com/co/run/finance/best-credit-card-processors-for-nonprofits (дата звернення : 26.03.2025 р.)

24. Web Volunteers (2022). Проєктуємо та розробляємо сайти для волонтерських організацій та благодійних фондів України під час війни URL : https://webvolunteers.net (дата звернення: 27.04.2025 р.)

25. Wild Apricot. (2020). 15 Nonprofit Website Best Practices You Need to Know. URL : https://www.wildapricot.com/blog/nonprofit-website-best-practices. (дата звернення : 26.03.2025 р.)

26. Wild Apricot (2024). The BEST Online Payment Processing for Nonprofits: Our Top 17 Picks. URL : https://www.wildapricot.com/blog/payment-processing-for-nonprofits (дата звернення : 26.03.2025 р.)

27. WPForms (2023). 5 Best Payment Processors for Nonprofits. URL : https://wpforms.com/best-payment-processors-for-nonprofits (дата звернення : 26.03.2025 р.)

28. Асоціація міст України. (2022). Перелік організацій та фондів, які допомагають українцям під час війни. URL : https://auc.org.ua/novyna/perelik-organizaciy-ta-fondiv-yaki-dopomagayut-ukrayincyam-pid-chas-viyny (дата звернення : 26.03.2025 р.)

29. ВБО «Асоціація благодійників України» (2024). Результати другого дослідження «ТОП 100+ транспарентних доброчинних організацій». URL : https://vboabu.org.ua/news/6231.html (дата звернення: 27.04.2025 р.)