Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра програмної інженерії
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Розробка мобільного застосунку для координації волонтерської діяльності з використанням мови програмування Kotlin

Виконав: студент IV курсу, групи СП-42 спеціальності
121 Інженерія програмного забезпечення
(шифр і назва спеціальності)

Остап'юк О. С.
(підпис) (прізвище та ініціали)

Керівник Михалик Д. М.
(підпис) (прізвище та ініціали)

Нормоконтроль Стоянов Ю. М.
(підпис) (прізвище та ініціали)

Завідувач кафедри Петрик М. Р.
(підпис) (прізвище та ініціали)

Рецензент
(підпис) (прізвище та ініціали)

Тернопіль
2026

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет Комп'ютерно інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра Програмної інженерії
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Петрик М. Р.
(підпис) (прізвище та ініціали)
« » 2026 р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня бакалавр
(назва освітнього ступеня)

за спеціальністю 121 Інженерія програмного забезпечення
(шифр і назва спеціальності)

студенту Остап'юку Олександру Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка мобільного застосунку для координації волонтерської діяльності з використанням мови програмування Kotlin

Керівник роботи к.т.н., доц. Михалик Д.М.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «___» _____ 2026 року № _____

2. Термін подання студентом завершеної роботи _____

3. Вихідні дані до роботи Предметна область, технічне завдання, вимоги та специфікація, програмне рішення

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. Розділ 1. Аналіз предметної області мобільного застосунку для координації волонтерської діяльності. Розділ 2. Проєктування мобільного застосунку Aidly. Розділ 3. Реалізація та тестування мобільного застосунку Aidly. Розділ 4. Безпека життєдіяльності, основи охорони праці. Висновок. Список використаних джерел. Додатки.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
Схеми та діаграми, знімки екрану з проробленою роботою, ілюстративні зображення, інформативні зображення для доповнення тексту.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи охорони праці			

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Ознайомлення з темою кваліфікаційної роботи		
2	Аналіз предметної області та підбір джерел інформації		
3	Формування вимог до програмної системи Aidly		
4	Вибір технологій, інструментів і сервісів для реалізації проєкту		
5	Проєктування архітектури застосунку, структури даних і UML-моделей		
6	Розробка основного функціоналу Android-застосунку		
7	Тестування застосунку та перевірка основних сценаріїв роботи		
8	Розробка та оформлення бази даних		
9	Підготовка матеріалів до розділу з охорони праці		
10	Оформлення пояснювальної записки кваліфікаційної роботи		
11	Нормоконтроль		
12	Перевірка на плагіат		
13	Попередній захист кваліфікаційної роботи		
14	Захист кваліфікаційної роботи		

Студент _____ Остап’юк О. С.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Михалик Д.М.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Розробка мобільного застосунку для координації волонтерської діяльності з використанням мови програмування Kotlin // Кваліфікаційна робота освітнього рівня "Бакалавр" // Остап’юк Олександр Сергійович // ТНТУ ім. Івана Пулюя, ФКІСПІ, кафедра програмної інженерії, група СП-42 // Тернопіль, 2026 // Ст. – 58, рис. – 27, табл. – 7 , додат. – 1, бібліогр. – 37.

Ключові слова: Android, Kotlin, Jetpack Compose, Firebase Authentication, Cloud Firestore, Cloudinary, мобільний застосунок, волонтерська діяльність, благодійник, волонтер, звітність, тестування.

Метою кваліфікаційної роботи є розробка мобільного застосунку Aidly для координації волонтерської діяльності з використанням мови програмування Kotlin та сучасних технологій Android-розробки.

У роботі проведено аналіз предметної області, сформовано вимоги, спроєктовано архітектуру, ролі, сценарії використання, моделі даних і структуру Cloud Firestore.

Реалізація виконана мовою Kotlin із використанням Jetpack Compose, Firebase Authentication, Cloud Firestore і Cloudinary. Застосунок забезпечує реєстрацію, авторизацію, створення зборів, перегляд стрічки, подання допомоги, підтвердження заявок, профілі користувачів, рейтинг і звітність.

ABSTRACT

Development of a mobile application for coordinating volunteer activities using Kotlin // Qualification Thesis for the Bachelor's Degree // Ostapiuk Oleksandr Serhiiovych // Ternopil Ivan Puluj National Technical University, Faculty of Computer and Information Systems and Software Engineering, Department of Software Engineering, Group SP-42 // Ternopil, 2026 // Pages – 58, Figures – 27, Tables – 7, Appendices – 1, References – 37.

Keywords: Android, Kotlin, Jetpack Compose, Firebase Authentication, Cloud Firestore, Cloudinary, mobile application, volunteer activity, donor, volunteer, reporting, testing.

The purpose of the qualification work is to develop a mobile application Aidly for coordinating volunteer activities using the Kotlin programming language and modern Android development technologies.

The work includes an analysis of the subject area, requirements, architecture, roles, usage scenarios, data models, and Cloud Firestore structure.

The implementation is done in Kotlin using Jetpack Compose, Firebase Authentication, Cloud Firestore, and Cloudinary. The application provides registration, authorization, meeting creation, feed viewing, assistance submission, application confirmation, user profiles, rating, and reporting.

ПЕРЕЛІК СКОРОЧЕНЬ І ТЕРМІНІВ

Aidly – мобільний застосунок для координації волонтерської діяльності, розроблений для платформи Android.

API (Application Programming Interface) – прикладний програмний інтерфейс, що забезпечує взаємодію між різними програмними компонентами та сервісами.

CRUD (Create, Read, Update, Delete) – базові операції роботи з даними: створення, читання, оновлення та видалення.

Cloud Firestore – хмарна NoSQL-база даних Firebase, яка використовується для зберігання профілів користувачів, зборів, заявок допомоги, звітів і підписок.

Cloudinary – хмарний сервіс для завантаження, зберігання та отримання зображень, який використовується у застосунку для роботи з медіафайлами.

Flow – механізм асинхронної передачі потоку даних у Kotlin, що використовується для реактивного оновлення інформації у застосунку.

Jetpack Compose – сучасний декларативний інструментарій Android для побудови користувацького інтерфейсу.

JSON (JavaScript Object Notation) – текстовий формат обміну даними, що використовується під час взаємодії із зовнішніми сервісами.

MVVM (Model – View – ViewModel) – архітектурний шаблон, відповідно до якого застосунок поділяється на модель даних, інтерфейс користувача та компонент керування станом і логікою представлення.

StateFlow – компонент Kotlin Coroutines для зберігання та спостереження за станом даних у реальному часі.

UI (User Interface) – користувацький інтерфейс програмного застосунку.

UML (Unified Modeling Language) – уніфікована мова моделювання, яка використовується для опису структури та поведінки програмної системи.

UX (User Experience) – сукупність вражень користувача від взаємодії із застосунком, зокрема зручність, зрозумілість і логічність використання.

ViewModel – компонент архітектури Android, який відповідає за збереження стану екрана та обробку логіки взаємодії між інтерфейсом і даними.

ЗМІСТ

ВСТУП

Цифрова трансформація суттєво змінює способи організації волонтерської та благодійної діяльності. Сучасні волонтерські ініціативи потребують швидкого поширення інформації, зручної взаємодії між учасниками, прозорого підтвердження допомоги та фіксації результатів виконаної роботи.

Актуальність теми зумовлена тим, що інформація про волонтерські збори часто розміщується у різних соціальних мережах, месенджерах або окремих публікаціях. Через це користувачам складно швидко знайти потрібний збір, перевірити його актуальність, оцінити репутацію волонтера та переглянути результати попередньої діяльності. Така фрагментарність комунікації знижує зручність взаємодії між волонтерами та благодійниками.

Одним із шляхів вирішення цієї проблеми є створення мобільного застосунку, який об’єднує основні процеси координації допомоги в одній системі. Мобільний формат є доцільним, оскільки смартфон є основним засобом оперативної комунікації для більшості користувачів. Такий застосунок дозволяє переглядати актуальні збори, шукати потрібні ініціативи, фільтрувати їх за категорією чи регіоном, подавати заявку про надану допомогу та переглядати звіти після завершення зборів.

У межах кваліфікаційної роботи розроблено мобільний застосунок Aidly для координації волонтерської діяльності у форматі соціальної мережі. Система має дві основні ролі користувачів: волонтера та благодійника. Волонтер може проходити демонстративну верифікацію, створювати фінансові або матеріальні збори, переглядати заявки допомоги, підтверджувати або відхиляти їх, закривати збори та додавати звіти. Благодійник може переглядати стрічку, шукати збори й профілі, надсилати заявку “Я допоміг”, додавати суму або опис допомоги та отримувати рейтинг після підтвердження заявки.

Особливістю Aidly є поєднання функцій соціальної платформи та системи обліку волонтерської допомоги. У застосунку реалізовано профілі користувачів, підписки, соціальну стрічку, фільтрацію зборів, пошук, заявки допомоги, рейтинг

благодійників і звіти. Це дозволяє зробити процес взаємодії більш структурованим, зрозумілим і прозорим.

Технічною основою проєкту є платформа Android і мова програмування Kotlin. Для побудови інтерфейсу використано Jetpack Compose та Material Design 3. Архітектура застосунку побудована за шаблоном MVVM. Для автентифікації та зберігання даних використано Firebase Authentication і Cloud Firestore, а для роботи із зображеннями застосовано Cloudinary та Coil.

Метою кваліфікаційної роботи є розробка мобільного застосунку Aidly для координації волонтерської діяльності з використанням мови програмування Kotlin.

Об’єктом дослідження є процеси цифрової координації волонтерської та благодійної діяльності.

Предметом дослідження є програмні засоби та технології розробки Android-застосунку для роботи зі зборами, заявками допомоги, профілями користувачів, рейтингом і звітами.

Для досягнення мети виконано такі завдання: проаналізовано предметну область; визначено основні вимоги до системи; обґрунтовано вибір технологій; спроєктовано архітектуру та структуру даних; реалізовано основні функціональні модулі; виконано тестування ключових сценаріїв; розглянуто питання впровадження, підтримки та безпеки життєдіяльності.

Практичне значення роботи полягає у створенні мобільного застосунку, який може бути використаний як основа для цифрової платформи координації волонтерської діяльності. Розроблена система спрощує взаємодію між волонтерами та благодійниками, структурує інформацію про збори, підвищує прозорість підтвердження допомоги та забезпечує зручний перегляд звітів.

Кваліфікаційна робота складається зі вступу, чотирьох розділів, висновків, списку використаних джерел і додатків. У першому розділі розглянуто предметну область та вимоги до системи. У другому розділі описано архітектуру та моделі даних застосунку. У третьому розділі наведено реалізацію та тестування застосунку. У четвертому розділі розглянуто питання безпеки життєдіяльності та охорони праці.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ МОБІЛЬНОГО ЗАСТОСУНКУ ДЛЯ КООРДИНАЦІЇ ВОЛОНТЕРСЬКОЇ ДІЯЛЬНОСТІ

Перший розділ присвячено аналізу предметної області, визначенню проблем волонтерської координації, огляду існуючих цифрових інструментів, обґрунтуванню необхідності застосунку Aidly та формуванню вимог до програмної системи.

1.1 Аналіз предметної області

Волонтерська діяльність передбачає добровільну участь людей у наданні допомоги іншим особам, громадам, організаціям або суспільно важливим ініціативам [26]. У цифровому середовищі така діяльність потребує не тільки швидкого обміну повідомленнями, а й структурованого збереження інформації про потреби, учасників, статуси зборів, підтвердження допомоги та публічну звітність.

Для програмного продукту Aidly предметна область розглядається як сукупність взаємопов’язаних дій двох основних груп користувачів: волонтерів і благодійників. Волонтер створює збір, описує потребу, додає реквізити або матеріальну ціль, приймає заявки, підтверджує або відхиляє допомогу, закриває збір і додає звіт. Благодійник переглядає стрічку зборів, шукає актуальні потреби, фільтрує записи, переглядає профілі, надає допомогу поза межами застосунку та надсилає заявку "Я допоміг".

Особливістю предметної області є те, що довіра формується не лише через текстовий опис збору, а й через історію дій користувача [17]. Для волонтера важливими є верифікація, профіль, кількість завершених зборів, наявність звітів і підписників. Для благодійника важливими є підтверджені заявки, рейтинг, сума допомоги та історія участі. Тому застосунок повинен не просто показувати окремі оголошення, а формувати соціальний контекст навколо користувачів.

Проблемою традиційної організації волонтерських зборів є розпорошеність даних. Опис потреби може бути опублікований у соціальній мережі, реквізити

передані в окремому повідомленні, підтвердження допомоги надіслане через месенджер, а звіт розміщений в іншому каналі. У такій ситуації користувачам складно перевірити актуальний статус збору, знайти попередні звіти, зрозуміти, чи було підтверджено їхню допомогу, і відстежити прогрес.

Додатковою проблемою є відсутність формалізованих статусів. Якщо збір активний, завершений або вже прозвітований, це має бути видно в інтерфейсі. Якщо заявка очікує підтвердження, підтверджена або відхилена, це також має бути збережено як стан. Саме тому Aidly використовує окремі статуси кампаній і заявок, а також обмежує доступ до дій залежно від ролі користувача.

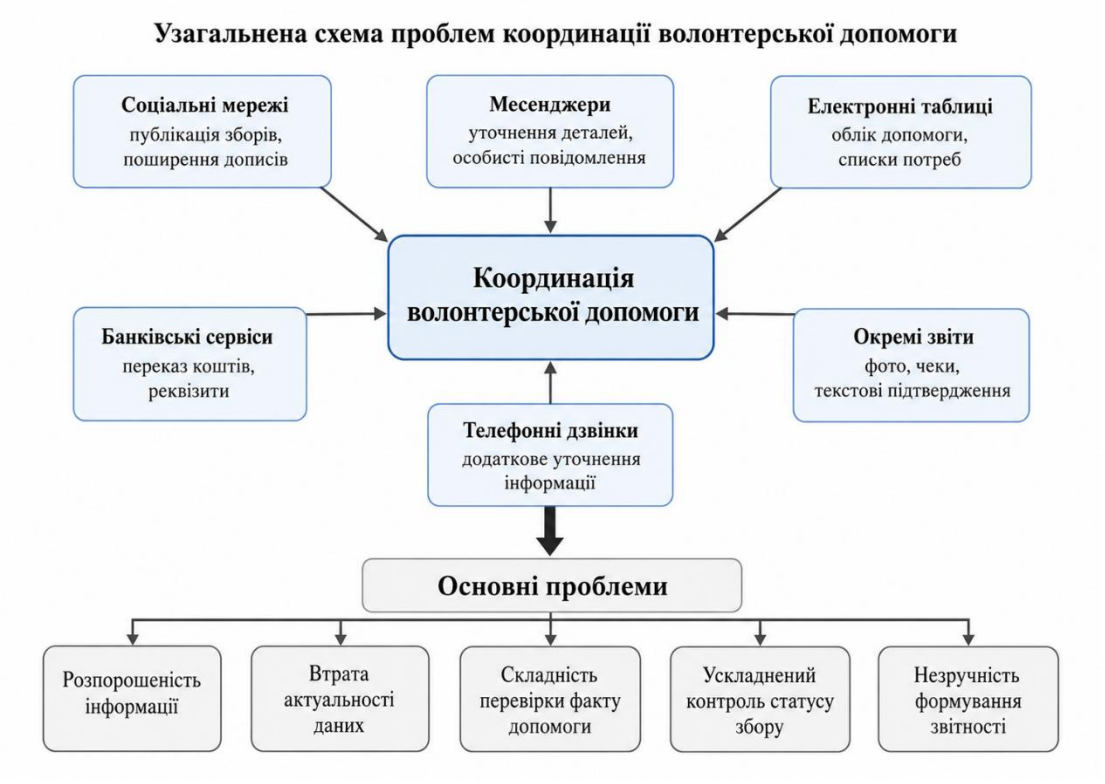

Рисунок 1.1 – Узагальнена схема проблем координації волонтерської допомоги

Мобільний формат є важливим для цієї предметної області, оскільки значна частина взаємодії відбувається швидко і не завжди з комп’ютера. Користувач повинен мати змогу переглянути збір, скопіювати реквізити, подати заявку, завантажити скріншот або переглянути профіль з телефона. Тому Aidly реалізовано як native Android-застосунок, а не як набір окремих веб-сторінок або таблиць [2].

Отже, предметна область Aidly поєднує соціальну довіру, структурований облік зборів, підтвердження допомоги, профілі користувачів, підписки, рейтинг і звітність. Ці характеристики визначають вимоги до архітектури, моделей даних, інтерфейсу та бізнес-логіки застосунку.

1.2 Огляд існуючих цифрових рішень

Для координації волонтерської діяльності на практиці використовуються різні цифрові інструменти: соціальні мережі, месенджери, банківські сервіси, веб-сайти благодійних фондів, електронні таблиці та окремі сервіси для збору пожертв [26]. Кожен із цих інструментів вирішує частину задачі, але не завжди забезпечує повний цикл взаємодії між волонтером і благодійником.

Соціальні мережі зручні для швидкого поширення інформації. Вони дозволяють публікувати дописи, додавати фото, поширювати збори та привертати увагу аудиторії. Проте соціальні мережі не мають спеціалізованої структури для статусів зборів, підтвердження допомоги, обліку заявок і формування рейтингу благодійника. Дописи швидко зникають у стрічці, а важлива інформація може губитися серед коментарів.

Месенджери забезпечують швидку комунікацію між людьми. Вони корисні для уточнення деталей, надсилання фото, скріншотів і швидких повідомлень. Проте месенджери не є повноцінною системою обліку зборів. У них складно зберігати історію в структурованому вигляді, фільтрувати потреби за категоріями, відображати профілі учасників, автоматично оновлювати статуси та формувати прозору звітність.

Банківські сервіси зручні для створення фінансових банок і збору коштів. Вони добре вирішують задачу прийому платежів, але не охоплюють матеріальну допомогу, заявки про виконану допомогу, соціальні профілі, підписки, верифікацію волонтера та звіти у форматі соціальної стрічки. Такі сервіси також не завжди дозволяють пов’язати внесок із профілем благодійника в окремій волонтерській системі.

Веб-сайти благодійних фондів можуть бути надійними джерелами інформації про великі ініціативи. Вони зазвичай містять сторінки про фонди, реквізити, новини та звіти. Однак такі сайти часто орієнтовані на конкретну організацію, а не на відкриту соціальну взаємодію між багатьма волонтерами і благодійниками. Крім того, для мобільного користувача окремий Android-застосунок може бути зручнішим за веб-інтерфейс.

Таблиця 1.1 – Порівняння цифрових інструментів для волонтерської координації

Інструмент	Переваги	Обмеження
Соціальні мережі	Швидке поширення, велика аудиторія, фото і коментарі	Немає структурованих статусів, заявок, рейтингу і контролю ролей
Месенджери	Оперативна комунікація, просте надсилання фото	Дані губляться в чатах, складно вести облік і звітність
Банківські сервіси	Зручність фінансових зборів, реквізити, платежі	Обмежена підтримка матеріальної допомоги, профілів і соціальної взаємодії
Веб-сайти фондів	Офіційність, звіти, інформація про організацію	Часто прив’язані до однієї організації, не завжди зручні для мобільної взаємодії
Aidly	Ролі, збори, заявки, підтвердження, профілі, підписки, рейтинг, звіти	Не виконує платежі всередині застосунку

Проведений огляд показує, що існуючі рішення закривають окремі частини процесу, але не забезпечують єдиного мобільного середовища для повного циклу волонтерської взаємодії. Саме тому доцільним є розроблення застосунку Aidly, який об’єднує основні функції координації, соціальної довіри та звітності [18].

1.3 Обґрунтування необхідності розроблення Aidly

Необхідність розроблення Aidly зумовлена потребою у спеціалізованому мобільному застосунку, який поєднує функції волонтерської платформи і соціальної мережі [18, 26]. На відміну від універсальних каналів комунікації, Aidly орієнтований саме на логіку зборів, заявок, підтвердження допомоги, рейтингу та звітів.

Основна ідея Aidly полягає в тому, що волонтер створює збір, а благодійник знаходить його у стрічці, надає допомогу поза межами застосунку та подає заявку "Я допоміг". Після цього волонтер перевіряє заявку і підтверджує або відхиляє її. У разі підтвердження прогрес збору оновлюється, благодійник отримує рейтинг, а історія допомоги зберігається в системі.

Важливою перевагою такого підходу є контроль ролей [17]. У Aidly передбачено дві ролі: VOLUNTEER і DONOR. Волонтер відповідає за створення зборів і обробку заявок, а благодійник за перегляд ініціатив і подання допомоги. Застосунок не додає сторонніх ролей, що спрощує логіку доступу і робить систему зрозумілішою для користувача.

Окремою функцією є демонстративна верифікація волонтера [17, 27]. Вона реалізована як вибір фото документа або посвідчення, після чого профіль отримує статус верифікованого. Така функція не замінює юридичної перевірки особи, але демонструє архітектурну можливість обмежити створення зборів лише для перевірених волонтерів. Це важливо для дипломного проєкту, оскільки показує роботу з ролями, станами, профілем і бізнес-правилами.

Aidly також враховує потребу в матеріальній допомозі [26]. Не всі волонтерські потреби виражаються у грошовій сумі. Часто йдеться про речі, товари, спорядження, медикаменти або інші матеріальні об’єкти. Тому в застосунку передбачено два типи зборів: FINANCIAL і MATERIAL. Для фінансового збору зберігається цільова сума, поточний прогрес і реквізити. Для матеріального збору зберігається опис матеріальної потреби, а прогрес може відображатися через кількість підтверджених допомог.

Соціальна складова Aidly реалізована через профілі, підписки, публічні сторінки користувачів і стрічку. Це дає змогу благодійнику відстежувати волонтерів, яким він довіряє, переглядати їхні кампанії та звіти, а також оцінювати активність через статистику профілю. Для волонтера це створює простір для формування репутації.

Отож, Aidly є доцільним програмним рішенням для теми кваліфікаційної роботи, оскільки об’єднує актуальну соціальну проблему, практичні сценарії користувачів, сучасний Android-стек, хмарну базу даних, завантаження зображень, архітектуру MVVM і повноцінну систему бізнес-правил [2, 3, 18].

1.4 Формування функціональних вимог

Функціональні вимоги визначають, які можливості повинна надавати система користувачам [18]. Для Aidly вимоги сформовано на основі предметної області, ролей користувачів і реальних сценаріїв взаємодії між волонтером та благодійником:

1) Система повинна забезпечувати реєстрацію, авторизацію та вихід користувача з облікового запису.
2) Система повинна підтримувати два типи користувачів: волонтера та благодійника.
3) Система повинна забезпечувати створення та редагування профілю користувача.
4) Система повинна забезпечувати перевірку унікальності username та пошук користувачів за публічними профілями.
5) Система повинна надавати волонтеру можливість пройти демонстративну верифікацію.
6) Система повинна дозволяти верифікованому волонтеру створювати, переглядати, закривати та супроводжувати збори.
7) Система повинна забезпечувати перегляд стрічки зборів і звітів із можливістю пошуку та фільтрації.

8) Система повинна дозволяти благодійнику подавати заявку про надану допомогу.
9) Система повинна дозволяти волонтеру підтверджувати або відхиляти заявки про допомогу.
10) Система повинна забезпечувати публікацію звітів, підписки на користувачів і відображення рейтингу активності.

Ключовою вимогою є коректне обмеження дій відповідно до ролі [17]. Благодійник може переглядати стрічку, надсилати заявки допомоги та підписуватися на користувачів, але не може створювати збори. Волонтер може створювати збори лише після верифікації, обробляти заявки лише до власних зборів, закривати власні кампанії та додавати звіти після завершення збору або досягнення цілі.

Фінансові та матеріальні збори мають різну логіку. Для фінансового збору обов’язковою є цільова сума, а після підтвердження заявки поточна сума збільшується на суму допомоги. Для матеріального збору важливим є опис переданої допомоги, а підтвердження заявки збільшує кількість підтверджених внесків. Такий поділ дає змогу підтримати різні типи волонтерських потреб.

Рейтинг благодійника формується на основі підтверджених заявок. Для фінансової допомоги рейтинг залежить від суми: за кожні 100 грн підтвердженої допомоги нараховується 1 бал. Для матеріальної допомоги може використовуватися фіксована кількість балів за підтверджену заявку. Це дозволяє показати активність благодійника у профілі та мотивує користувачів подавати коректні заявки.

1.5 Формування нефункціональних вимог

Нефункціональні вимоги визначають якісні характеристики програмної системи: зручність, надійність, підтримуваність, масштабованість, безпеку доступу та стабільність роботи [18]. Для Aidly ці вимоги особливо важливі, оскільки

застосунок працює з користувацькими профілями, зображеннями, заявками, ролями та хмарними даними.

Список нефункціональних вимог:

1) Застосунок повинен працювати на широкому колі Android-пристроїв [2].
2) Застосунок повинен бути реалізований як native Android-рішення.
3) Інтерфейс користувача повинен бути зручним, сучасним і зрозумілим.
4) Архітектура застосунку повинна бути розділена на незалежні логічні шари.
5) Застосунок повинен коректно працювати з асинхронними операціями та оновленням даних.
6) Дані в системі повинні залишатися узгодженими під час складних операцій.
7) Застосунок повинен обмежувати доступ до функцій відповідно до ролі користувача.
8) Система повинна забезпечувати зрозумілу обробку помилок, порожніх станів і процесів завантаження.
9) Застосунок повинен бути зручним для подальшої підтримки та розширення.
10) Інтерфейс і повідомлення застосунку повинні відповідати цільовій аудиторії.

Застосунок повинен бути підтримуваним, тому важливим є розділення відповідальності між шарами. UI-екрани не повинні напряму працювати з Firestore або Cloudinary [3, 19, 20]. Вони повинні передавати події у ViewModel, а ViewModel повинні делегувати операції репозиторіям. Такий підхід спрощує тестування, зменшує дублювання та робить код зрозумілішим. Надійність роботи з даними забезпечується транзакціями Firestore [11].

Зручність користувача забезпечується зрозумілими формами, станами завантаження, повідомленнями про помилки, фільтрами, пошуком і короткими сценаріями. Для мобільного застосунку важливо, щоб основні дії виконувалися за кілька кроків і не потребували складної навігації.

1.6 Вибір технологій розроблення

Для реалізації Aidly обрано стек технологій, орієнтований на native Android-розробку, хмарне збереження даних, реактивне оновлення інтерфейсу та зручну підтримку зображень.

Основною мовою програмування є Kotlin [1, 2]. Вона добре інтегрується з Android SDK, має лаконічний синтаксис, підтримує null-safety, корутини та сучасні підходи до асинхронного програмування. Для Aidly це важливо, оскільки застосунок виконує багато операцій з Firebase, Firestore, Cloudinary і локальним станом інтерфейсу.

Для побудови інтерфейсу використано Jetpack Compose [4]. Він дає змогу описувати UI декларативно, тобто інтерфейс автоматично відображає поточний стан ViewModel. Такий підхід добре підходить для екранів авторизації, стрічки, профілю, деталей збору, форм заявок і звітів.

Firebase Authentication використано для реєстрації, входу та виходу користувача [9]. Після автентифікації застосунок додатково перевіряє наявність профілю у Cloud Firestore, оскільки саме там зберігаються роль, username, статистика, верифікація та інші дані користувача.

Cloud Firestore обрано як основне сховище даних [10]. Документна модель Firestore добре відповідає структурі Aidly: користувачі, збори, заявки, звіти, підписки та username можуть зберігатися в окремих колекціях. Підтримка snapshot listener дозволяє отримувати оновлення даних без ручного перезапуску застосунку.

Cloudinary використано для збереження зображень [12, 13]. Firestore не призначений для зберігання великих медіафайлів, тому доцільно зберігати у Firestore лише URL зображення, а сам файл завантажувати в хмарне медіасховище. Через Cloudinary завантажуються аватари та шапки профілю, фото зборів, скріншоти допомоги та фото звітів.

Coil використано для відображення зображень у Jetpack Compose [14]. Бібліотека дозволяє зручно завантажувати зображення за URL і показувати їх у компонентах AsyncImage.

Архітектурною основою є MVVM у поєднанні з Repository Pattern [3, 19, 20]. ViewModel зберігають стан екранів і обробляють події користувача, а репозиторії відповідають за зовнішні джерела даних. Це робить код більш структурованим і придатним до розширення.

2 ПРОЄКТУВАННЯ МОБІЛЬНОГО ЗАСТОСУНКУ AIDLY

Другий розділ присвячено проєктуванню архітектури Aidly, ролей користувачів, сценаріїв використання, моделей даних, структури Cloud Firestore, навігації, інтерфейсу та бізнес-правил.

2.1 Загальна архітектура системи

Aidly побудовано за шаровою архітектурою з використанням патерну MVVM [3, 19, 20]. Такий підхід дозволяє розділити відповідальність між інтерфейсом користувача, станом екранів, бізнес-логікою, доступом до даних і зовнішніми сервісами.

Архітектура застосунку містить такі основні шари:

1. UI layer - екрани та багаторазові компоненти Jetpack Compose.
2. ViewModel layer - класи, що зберігають стан екранів і обробляють дії користувача.
3. Repository layer - класи для роботи з Firebase Authentication, Cloud Firestore і Cloudinary.
4. Model layer - доменні моделі, enum-класи та sealed-моделі станів.
5. Utility layer - константи, валідатори, форматування дат, списки регіонів і ліміти форм.

Такий поділ відповідає принципу розділення відповідальності між компонентами програмної системи [18; 19].

MainActivity запускає кореневий Compose-компонент застосунку. AidlyApp ініціалізує залежності, зокрема репозиторії. AppNavigation керує переходами між екранами і реагує на SessionState. Якщо користувач не авторизований, відкривається LoginScreen. Якщо користувач авторизований, але профіль відсутній, відкривається RegisterScreen або сценарій завершення профілю. Якщо профіль існує, відкривається MainScreen.

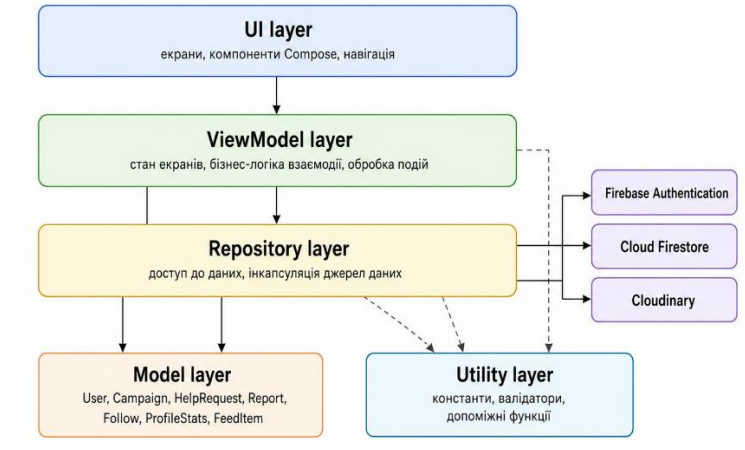

Рисунок 2.1 – Архітектура мобільного застосунку Aidly

Перевагою такої архітектури є те, що UI не містить прямого доступу до бази даних [3, 19]. Наприклад, екран створення збору не звертається безпосередньо до Firestore. Він передає дані у CampaignViewModel, яка викликає FirestoreRepository. Якщо потрібно завантажити зображення, ViewModel спочатку викликає CloudinaryRepository, отримує URL і лише після цього створює документ кампанії у Firestore.

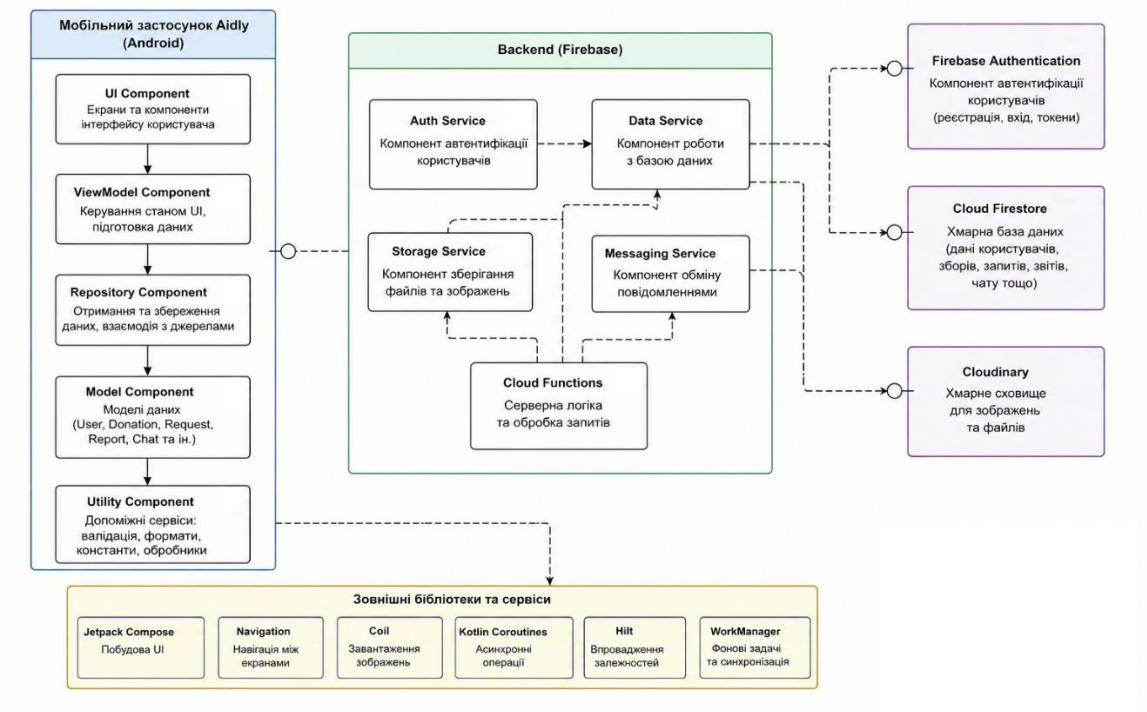

Рисунок 2.2 – Діаграма компонентів застосунку

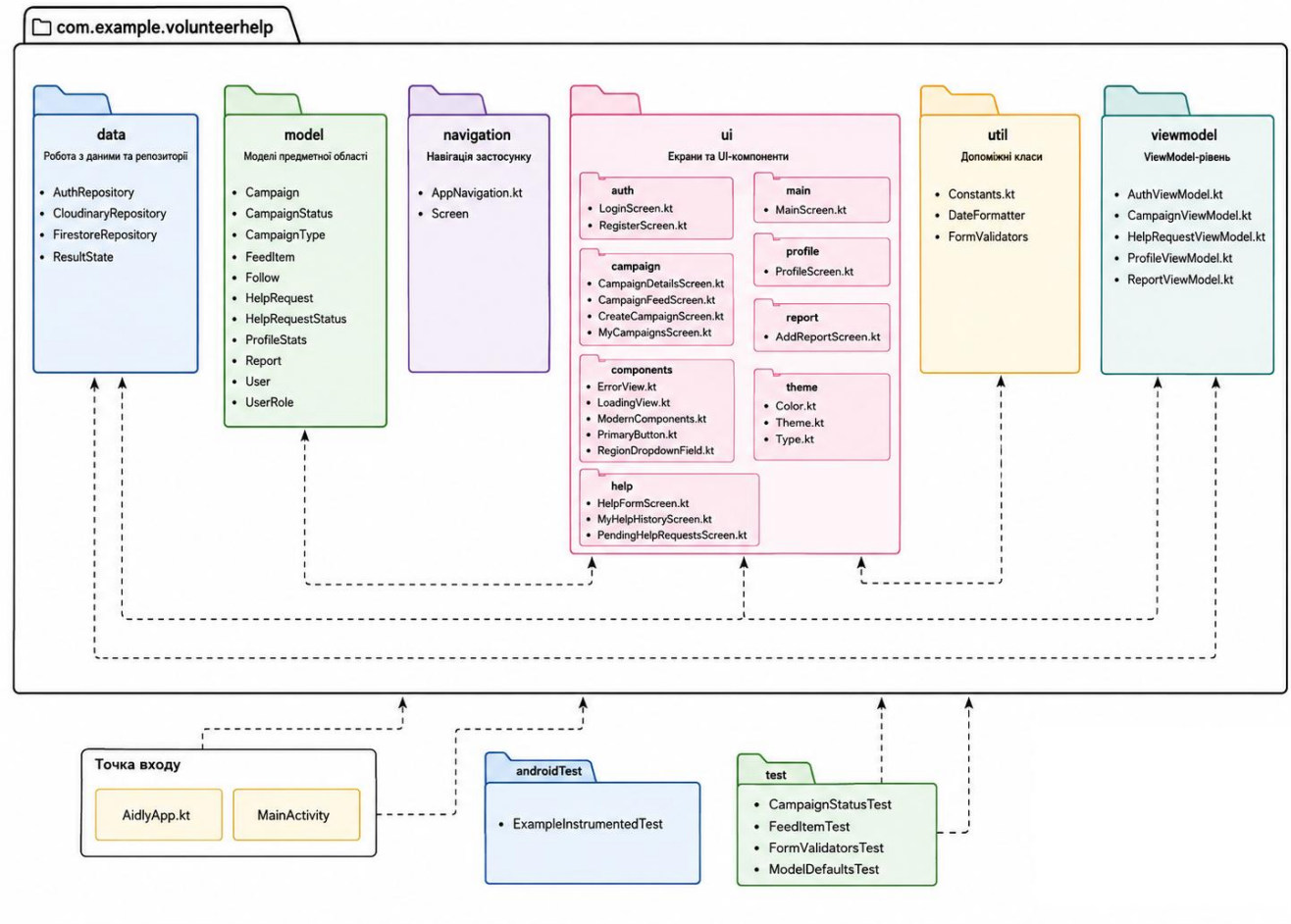

Рисунок 2.3 – Діаграма пакетів Android-проєкту

Таблиця 2.1 – Технологічний стек Aidly

Компонент	Технологія	Призначення
Мова програмування	Kotlin 1.9.24	Основна мова реалізації
UI	Jetpack Compose, Material3	Побудова екранів і компонентів
Навігація	Navigation Compose	Переходи між екранами
Архітектура	MVVM, Repository Pattern	Розділення відповідальності
Стан	StateFlow, Kotlin Flow	Реактивне оновлення UI
Асинхронність	Kotlin Coroutines	Фонові операції
Авторизація	Firebase Authentication	Реєстрація, вхід і вихід

Продовження таблиці 2.1

База даних	Cloud Firestore	Збереження профілів, зборів, заявок, звітів
Зображення	Cloudinary, Coil	Завантаження і показ медіафайлів
Збірка	Gradle Kotlin DSL, AGP 8.5.1	Конфігурація Android-проєкту

2.2 Проєктування ролей користувачів і прав доступу

У Aidly передбачено дві основні ролі: VOLUNTEER і DONOR. Такий розподіл відповідає логіці предметної області та дозволяє чітко обмежити доступ до дій [17, 18].

Роль VOLUNTEER призначена для користувача, який організовує волонтерські збори [26]. Волонтер може пройти верифікацію, створювати фінансові або матеріальні збори, переглядати заявки до зборів, підтверджувати або відхиляти допомогу, закривати кампанії та додавати звіти. Важливим правилом є те, що створення зборів доступне лише верифікованому волонтеру.

Роль DONOR призначена для благодійника. Благодійник може переглядати стрічку, шукати збори та профілі, підписуватися на користувачів, копіювати реквізити збору, подавати заявку "Я допоміг", завантажувати фото або скріншот підтвердження, переглядати власний рейтинг і статистику допомоги.

Діаграма варіантів використання застосовується для формалізації взаємодії акторів із функціями системи [18].

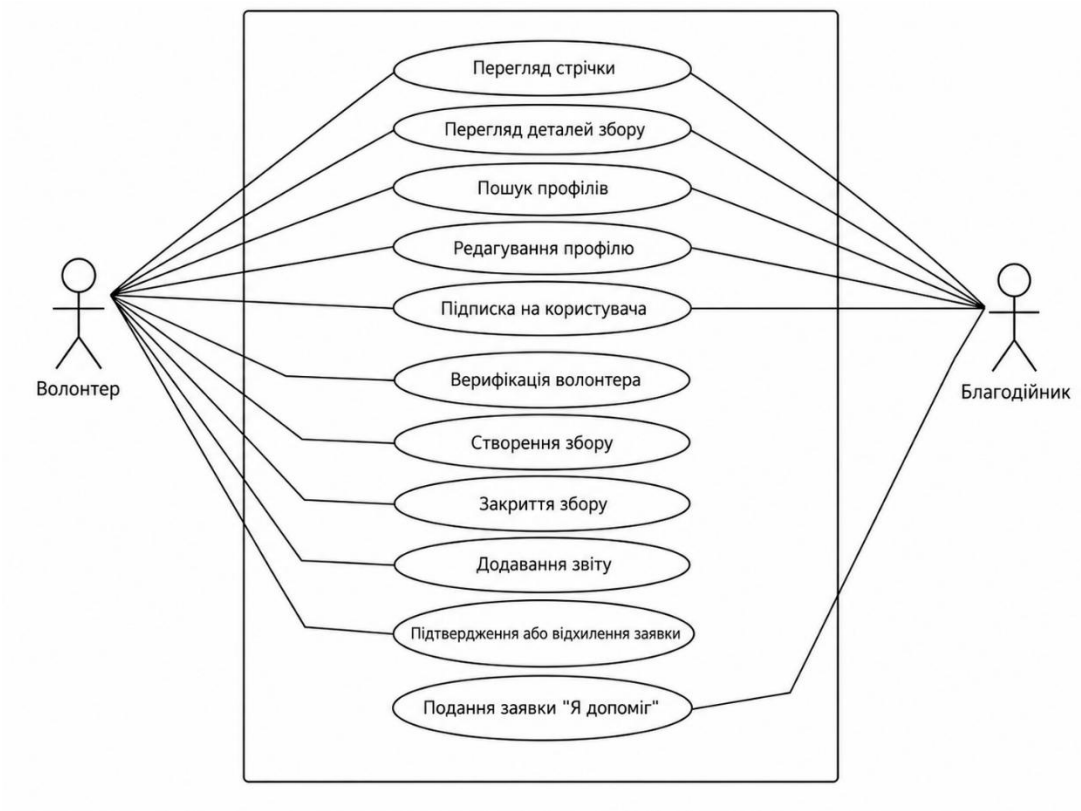

Рисунок 2.4 – Діаграма варіантів використання

2.3 Проєктування сценаріїв використання

Основні сценарії використання Aidly охоплюють повний життєвий цикл взаємодії користувачів із системою [18]. Сценарії починаються з реєстрації або входу, а далі залежать від ролі користувача.

Сценарій реєстрації передбачає введення імені, username, email, пароля та вибір ролі. Після створення Firebase-користувача у Firestore створюється профіль [9, 10]. Username перевіряється на унікальність через окрему колекцію usernames. Це дозволяє уникнути дублювання публічних імен користувачів [10, 11].

Сценарій входу передбачає введення email і пароля. Після успішної автентифікації AuthViewModel перевіряє, чи існує профіль у Firestore [9, 10]. Якщо профіль знайдено, користувач переходить на MainScreen. Якщо профіль відсутній, система переводить користувача до завершення реєстрації.

Сценарій створення збору доступний лише для верифікованого волонтера. Користувач заповнює назву, опис, тип збору, категорію, місто, область, реквізити

або матеріальну ціль, а також може додати зображення. Якщо зображення обране, воно завантажується у Cloudinary, після чого URL зберігається у документі кампанії [12, 13].

Сценарій подання заявки допомоги доступний благодійнику. Користувач відкриває активний збір, натискає "Я допоміг", вводить суму або опис матеріальної допомоги, додає коментар і за потреби завантажує фото або скріншот. Після надсилання створюється HelpRequest зі статусом PENDING.

Сценарій підтвердження заявки виконує волонтер, який є власником відповідного збору. Після підтвердження статус заявки змінюється на APPROVED, а прогрес збору оновлюється [10, 11]. Для фінансового збору currentAmount збільшується на суму заявки. Якщо сума досягає цілі, статус кампанії змінюється на GOAL_REACHED. Для матеріального збору currentAmount збільшується на 1.

Сценарій додавання звіту доступний після закриття збору або досягнення цілі. Волонтер додає опис звіту та за потреби фото. Після створення звіту статус кампанії змінюється на REPORTED, а звіт відображається у стрічці та профілі волонтера.

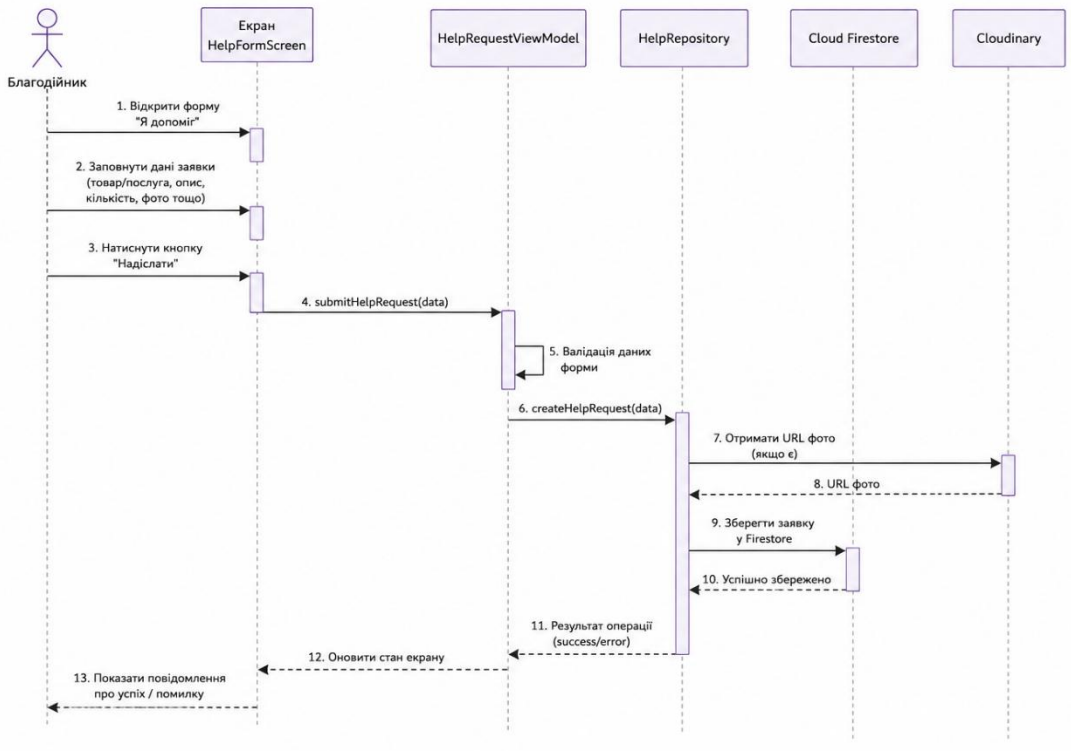

Рисунок 2.5 – Sequence diagram для подання заявки «Я допоміг»

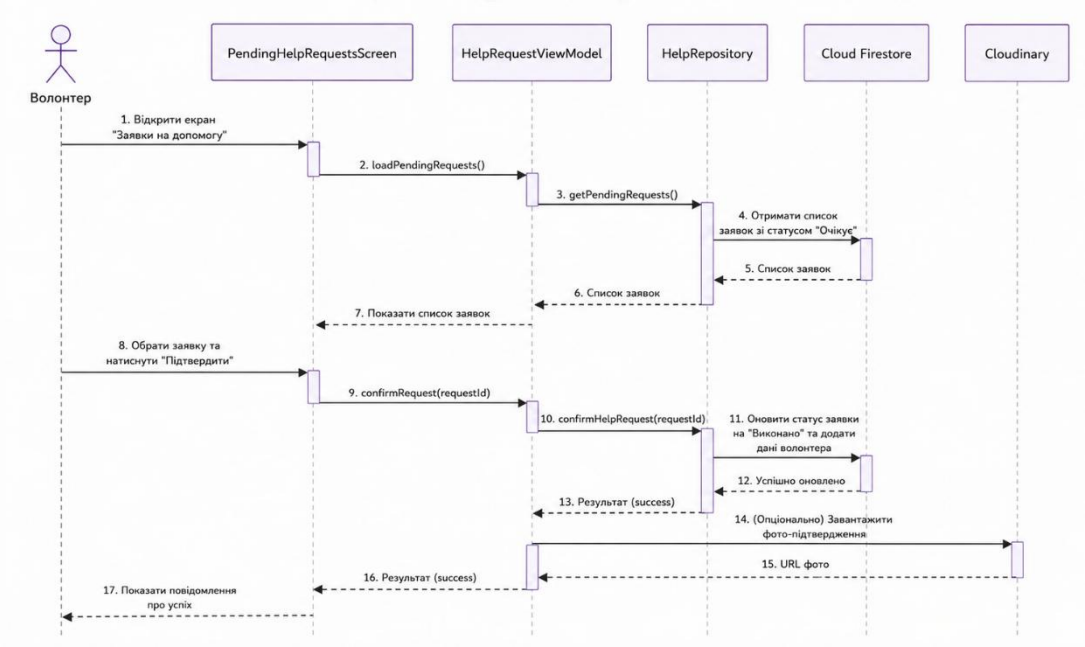

Рисунок 2.6 – Sequence diagram для підтведжння заявки волонтером

2.4 Проєктування моделей даних

Моделі даних Aidly відображають основні сутності предметної області [18, 22]. До них належать User, Campaign, HelpRequest, Report, Follow, ProfileStats і FeedItem.

Модель User описує профіль користувача. Вона містить ідентифікатор Firebase UID, ім’я, email, username, роль, аватар, шапку профілю, bio, місто, область, рейтинг, ознаку верифікації, дату створення та лічильники підписників і підписок.

Модель Campaign описує волонтерський збір. Вона містить назву, опис, тип збору, категорію, цільову суму або матеріальну ціль, поточний прогрес, місто, область, реквізити, зображення, дані волонтера та статус.

Модель HelpRequest описує заявку благодійника про надану допомогу. Вона містить ідентифікатор збору, назву кампанії, дані благодійника, дані волонтера, тип допомоги, суму або опис речей, коментар, скріншот і статус.

Модель Report описує звіт після завершення збору. Вона містить посилання на кампанію, назву збору, дані волонтера, опис, зображення і дату створення.

Модель Follow описує підписку між користувачами. Вона містить ідентифікатор підписника, ідентифікатор користувача, на якого підписалися, і дату створення.

Модель ProfileStats використовується для відображення статистики профілю. Вона агрегує кількість зборів, звітів, заявок, підтверджених допомог, суму допомоги, рейтинг, рівень і титул користувача.

Таблиця 2.2 – Основні моделі даних Aidly

Модель	Призначення
User	Профіль користувача, роль, username, рейтинг, верифікація
Campaign	Фінансовий або матеріальний збір волонтера
HelpRequest	Заявка благодійника про надану допомогу
Report	Звіт волонтера після завершення збору
Follow	Зв’язок підписки між користувачами
ProfileStats	Агрегована статистика профілю
FeedItem	Узагальнений елемент стрічки, що може бути збором або звітом

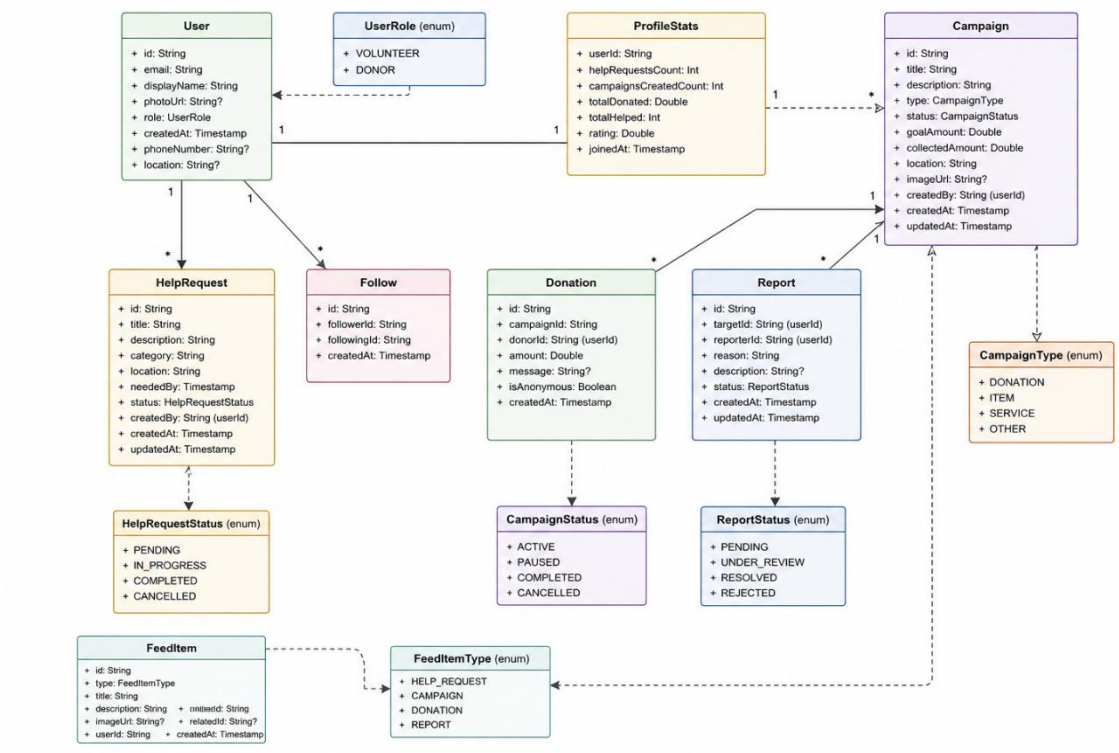

Рисунок 2.7 – Діаграма класів основних моделей

2.5 Проєктування структури бази даних Cloud Firestore

Для збереження даних Aidly використано Cloud Firestore [10]. Структура бази даних побудована на окремих колекціях, кожна з яких відповідає певному типу сутностей.

Основними колекціями є:

- users;
- usernames;
- follows;
- campaigns;
- helpRequests;
- reports.

Колекція users зберігає профілі користувачів. Колекція usernames використовується як індекс унікальних username. Колекція follows зберігає підписки. Колекція campaigns містить збори. Колекція helpRequests містить заявки допомоги. Колекція reports містить звіти.

Таблиця 2.3 – Колекції Cloud Firestore

Колекція	Призначення	Основні дані
users	Профілі користувачів	name, username, role, avatarUrl, rating, isVerified
usernames	Унікальні username	usernameLowercase, userId
follows	Підписки	followerId, followingId, createdAt
campaigns	Збори	title, type, targetAmount, currentAmount, status
helpRequests	Заявки допомоги	campaignId, donorId, amount, status
reports	Звіти	campaignId, volunteerId, description, imageUrl

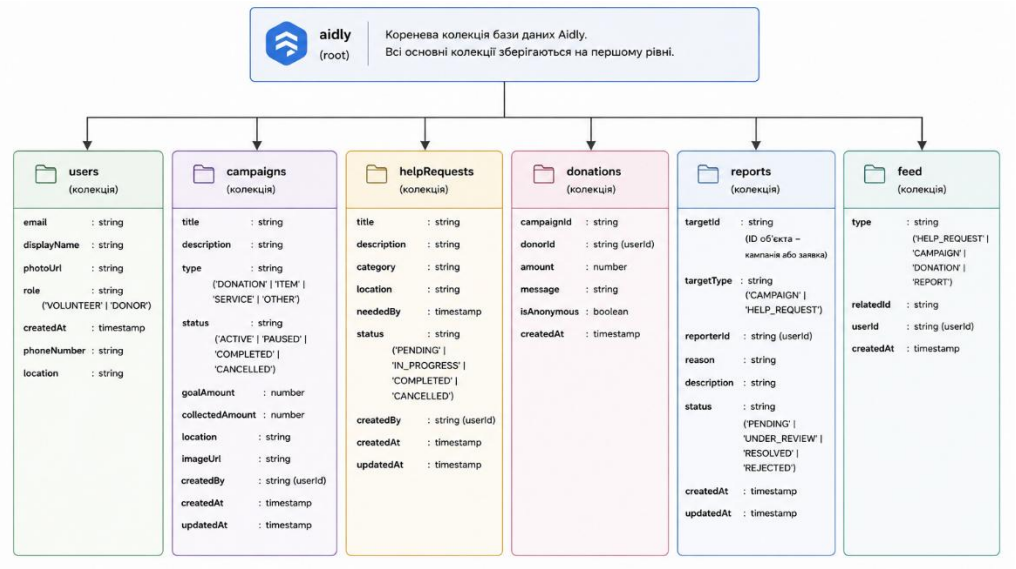

Рисунок 2.8 – Структура колекцій Cloud Firestore

Документна модель Firestore має особливості, які вплинули на проєктування. Оскільки Firestore не використовує SQL-з'єднання між таблицями, частина даних денормалізується. Наприклад, у Campaign зберігаються ім'я, username, аватар і статус верифікації волонтера. У HelpRequest зберігаються дані благодійника і назва кампанії. Це дозволяє швидше будувати списки без великої кількості додаткових запитів.

Для критичних операцій використовуються транзакції. Транзакція потрібна для створення username, підтвердження заявки, оновлення прогресу збору, підписки на користувача, закриття кампанії та створення звіту [11]. Такий підхід забезпечує узгодженість даних.

Окремої уваги потребує оновлення денормалізованих даних. Якщо користувач змінює ім'я, username або аватар, ці дані потрібно оновити не тільки в документі users, а й у пов'язаних кампаніях, заявках і звітах.

2.6 Проєктування навігації та інтерфейсу користувача

Навігація Aidly побудована на Navigation Compose [5]. Основними маршрутами є Login, Register, Main, CampaignDetails, CreateCampaign, HelpForm, AddReport, EditProfile, VolunteerVerification і PublicProfile.

Початковий маршрут залежить від стану сесії. Якщо користувач не авторизований, система відкриває LoginScreen. Якщо користувач авторизований, але його профіль не знайдено, система відкриває RegisterScreen або екран завершення профілю. Якщо користувач авторизований і профіль існує, відкривається MainScreen.

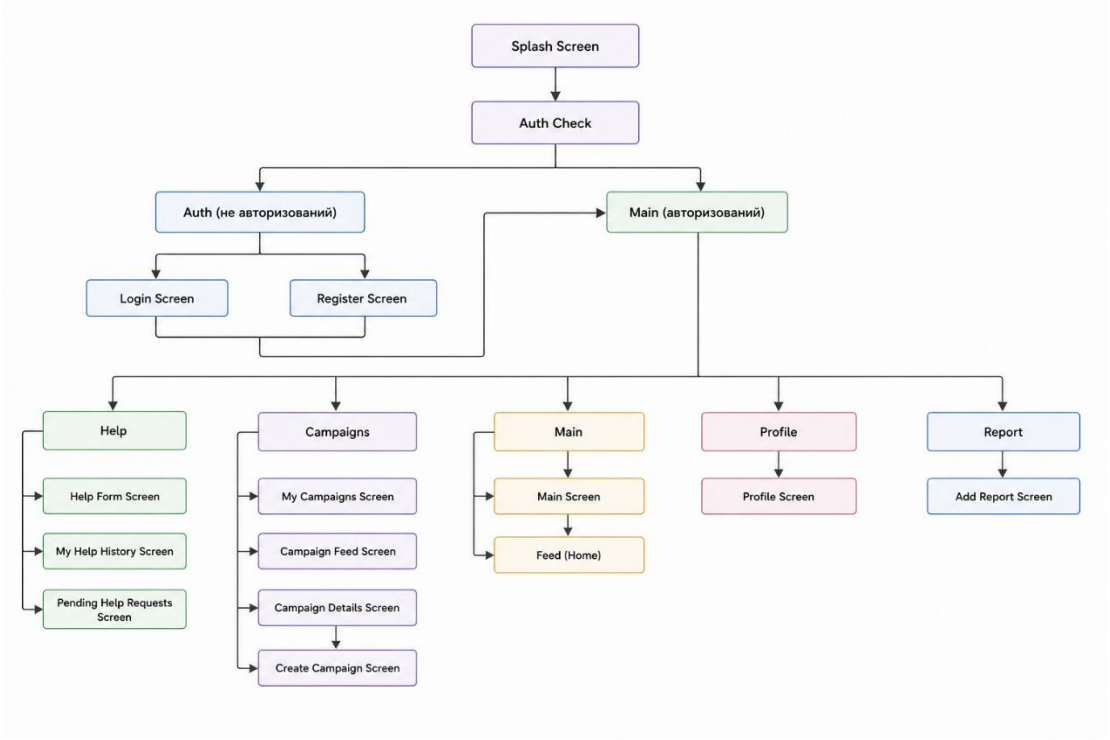

Рисунок 2.9 – Навігаційна схема застосунку

Інтерфейс користувача побудовано навколо коротких мобільних сценаріїв. Екран авторизації містить поля email і пароль, кнопку входу та перехід до реєстрації. Екран реєстрації містить ім’я, username, email, пароль, підтвердження пароля і вибір ролі.

MainScreen використовує нижню навігацію. Для різних ролей набір доступних дій може відрізнятися. Це дозволяє не перевантажувати інтерфейс зайвими кнопками. Наприклад, благодійник орієнтується на стрічку, пошук, заявки допомоги та профіль, а волонтер також працює зі створенням зборів і заявками до власних кампаній.

Стрічка є центральним екраном застосунку. Вона містить кампанії та звіти, відсортовані за датою створення. Користувач може шукати записи за назвою, описом, містом, областю або автором. Також доступні фільтри за типом, регіоном, категорією, підписками, завершеними зборами та зборами, які майже досягли цілі.

Профіль користувача виконує функцію сторінки довіри. Він містить аватар, шапку профілю, ім’я, username, роль, верифікацію, bio, місто, область, рейтинг, статистику, підписників, підписки, кампанії та звіти.

2.7 Проєктування станів і бізнес-правил

Aidly використовує кілька важливих станів, які визначають поведінку системи [18]. До них належать стан сесії, статус кампанії, статус заявки та стан верифікації волонтера.

SessionState описує стан користувача у застосунку. Можливими станами є Loading, LoggedOut, ProfileIncomplete і Authenticated. Цей стан використовується для визначення стартового екрана. CampaignStatus описує стан збору. Основними станами є ACTIVE, GOAL_REACHED, CLOSED і REPORTED. Активний збір може приймати заявки. Якщо фінансова ціль досягнута, збір переходить у GOAL_REACHED. Волонтер може закрити збір, після чого він переходить у CLOSED. Після додавання звіту статус змінюється на REPORTED.

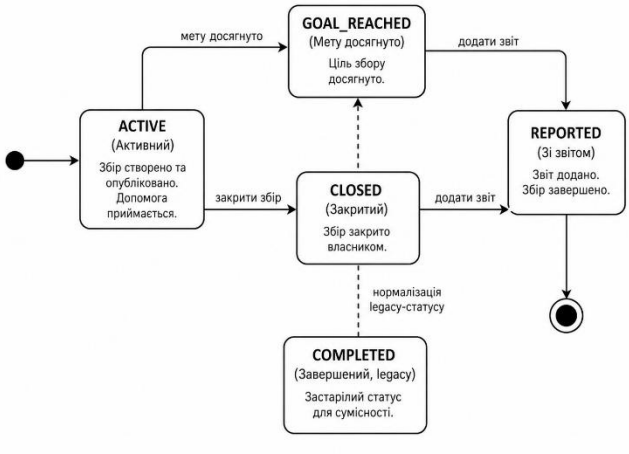

Рисунок 2.10 – Діаграма станів збору

HelpRequestStatus описує стан заявки допомоги. Заявка створюється у статусі PENDING. Волонтер може підтвердити її, тоді статус змінюється на APPROVED. Також волонтер може відхилити заявку, тоді статус змінюється на REJECTED.

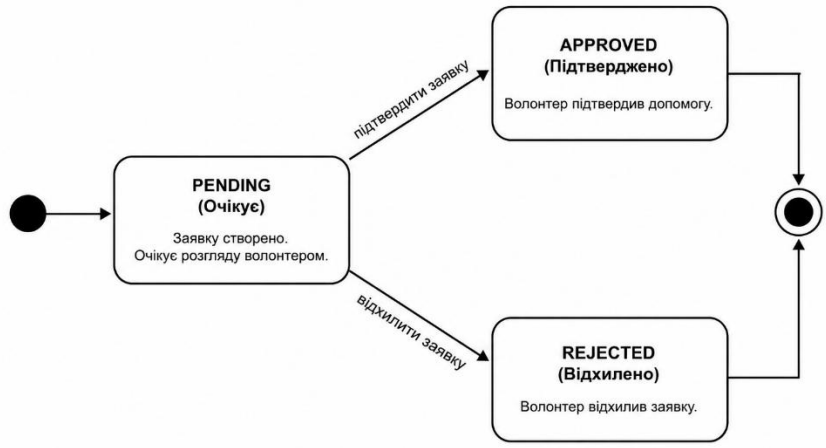

Рисунок 2.11 – Діаграма станів заявки про допомогу

Основні бізнес-правила Aidly:

1. Лише авторизований користувач може працювати у застосунку.
2. У системі використовуються лише ролі VOLUNTEER і DONOR.
3. Лише верифікований VOLUNTEER може створювати збір.
4. Лише DONOR може подати заявку "Я допоміг". Заявку можна подати лише до активного збору.
5. Лише волонтер-власник збору може підтвердити або відхилити заявку.
6. Підтвердивши фінансову заявку сума збору збільшується на суму заявки.
7. Після досягнення фінансової цілі статус збору переходить у GOAL_REACHED.
8. Звіт можна додати лише після CLOSED або GOAL_REACHED.
9. Після додавання звіту кампанія переходить у статус REPORTED.
10. Username повинен бути унікальним.

Отож, запропонована структура відповідає вимогам предметної області та забезпечує основу для реалізації мобільного застосунку.

3 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ МОБІЛЬНОГО ЗАСТОСУНКУ AIDLY

Третій розділ присвячено практичній реалізації Android-застосунку Aidly, структурі програмного проєкту, реалізації основних функціональних модулів, інтеграції хмарних сервісів і тестуванню.

3.1 Структура Android-проєкту

Android-проєкт Aidly реалізовано мовою Kotlin [1, 2]. Основний package name залишено com.example.volunteerhelp, що відповідає наявній структурі проєкту. Проєкт використовує Gradle Kotlin DSL, version catalog і сучасний Android Gradle Plugin [15].

Структура проєкту поділена на логічні пакети [19, 20]. Пакет model містить доменні моделі User, Campaign, HelpRequest, Report, Follow, ProfileStats, FeedItem та enum-класи. Пакет data містить репозиторії для роботи з Firebase Authentication, Cloud Firestore і Cloudinary. Пакет viewmodel містить AuthViewModel, CampaignViewModel, HelpRequestViewModel, ProfileViewModel і ReportViewModel. Пакет ui містить Compose-екрани та reusable-компоненти. Пакет navigation містить маршрути і логіку переходів. Пакет util містить константи, валідацію, форматування дат, список областей України та ліміти форм.

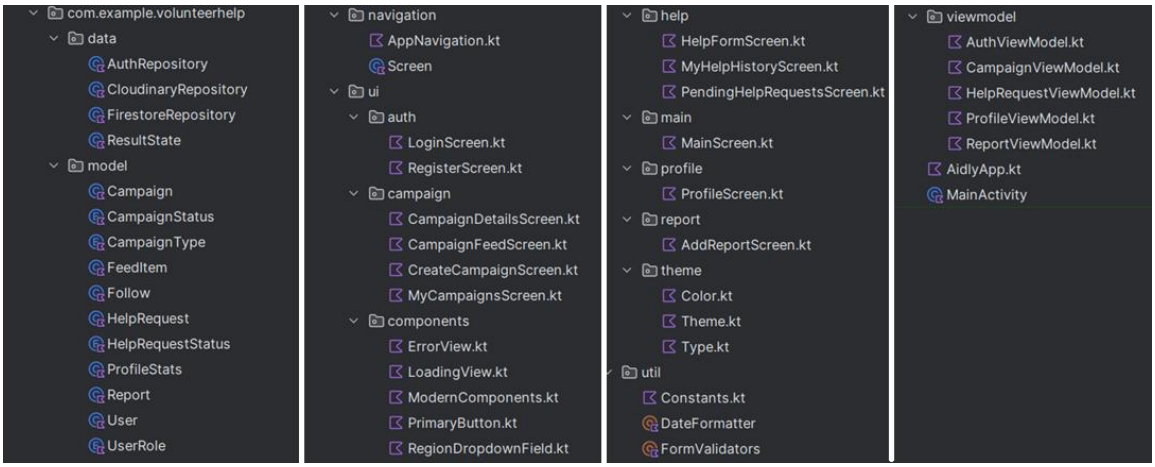

Рисунок 3.1 - Структура Android-проєкту в Android Studio

Така структура полегшує підтримку застосунку, оскільки кожна група класів має чітке призначення. Наприклад, зміни у вигляді кнопки або картки не впливають на репозиторії, а зміни в запитах до Firestore не потребують прямого редагування UI-екранів.

3.2 Реалізація авторизації та реєстрації

Авторизація реалізована з використанням Firebase Authentication [9]. Користувач вводить email і пароль, після чого AuthRepository інкапсулює роботу з FirebaseAuth, що відповідає підходу Repository Pattern [3, 19]. Результат операції передається у AuthViewModel, яка оновлює стан екрана.

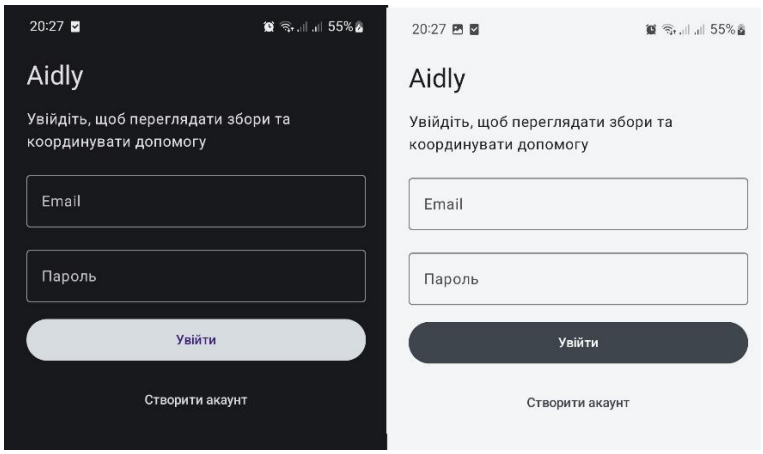

Рисунок 3.2 – Екран авторизації користувача

Після успішного входу система перевіряє профіль користувача у Firestore [9, 10]. Якщо профіль існує, формується стан Authenticated. Якщо Firebase-користувач існує, але профілю немає, формується стан ProfileIncomplete. Це дозволяє відокремити технічну автентифікацію від повного профілю застосунку.

Реєстрація передбачає введення імені, username, email, пароля, підтвердження пароля та вибір ролі. Після створення Firebase-користувача у Firestore створюється документ User. Username зберігається також в окремій колекції usernames, що забезпечує його унікальність.

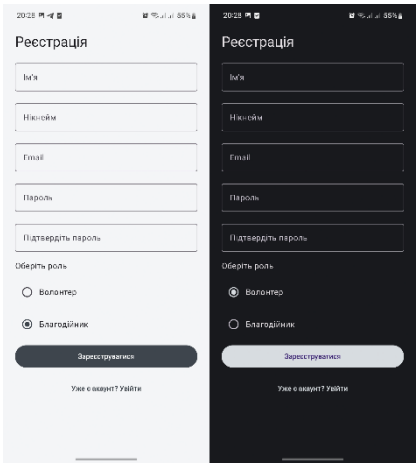

Рисунок 3.3 – Екран реєстрації та вибору ролі

Для форм авторизації і реєстрації використано валідацію. Перевіряється заповнення обов'язкових полів, коректність email, довжина пароля, збіг пароля і підтвердження, а також коректність username. Помилки відображаються українською мовою, що робить інтерфейс зрозумілим для цільових користувачів.

3.3 Реалізація профілю користувача і верифікації волонтера

Профіль користувача є одним із центральних модулів Aidly. Він містить основну інформацію про користувача, роль, username, аватар, обкладинку, опис, місто, область, рейтинг, підписників, підписки та статистику.

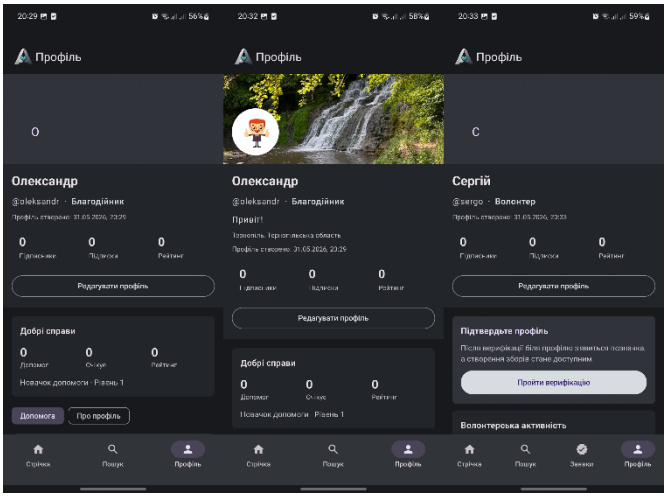

Рисунок 3.4 – Екран профілю користувача

Редагування профілю реалізовано через EditProfileScreen і ProfileViewModel. Користувач може змінити ім’я, username, bio, місто, область, аватар і шапку профілю. Якщо користувач змінює зображення, файл завантажується в Cloudinary, після чого URL зберігається у Firestore [10, 12, 13].

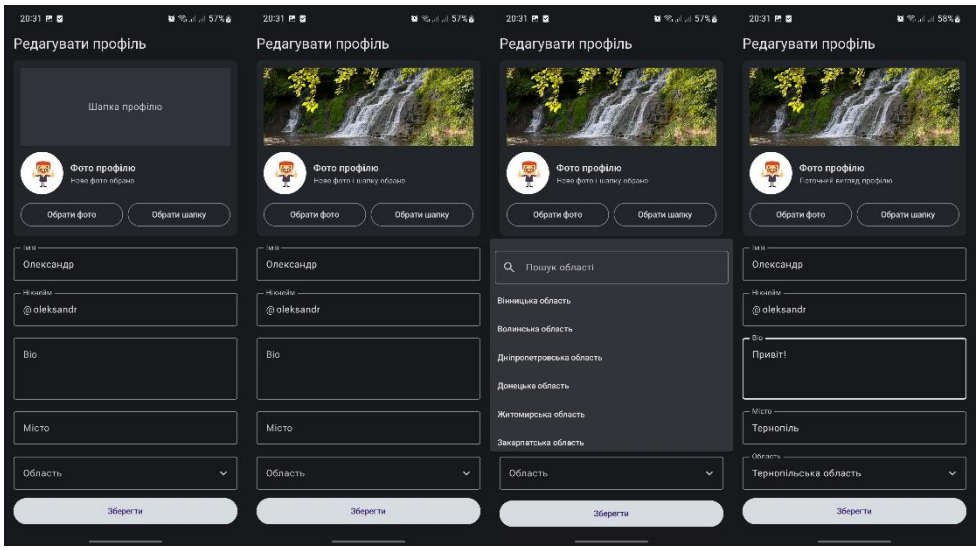

Рисунок 3.5 – Редагування профілю

Оскільки ім’я, username і аватар користувача можуть бути денормалізовані у кампаніях, заявках і звітах, після зміни профілю система оновлює пов’язані документи. Це забезпечує узгоджене відображення користувача в різних частинах застосунку.

Верифікація волонтера реалізована демонстративно. Волонтер відкриває екран верифікації, обирає фото документа або посвідчення та підтверджує дію. Після цього у профілі встановлюється isVerified = true і зберігається час верифікації.

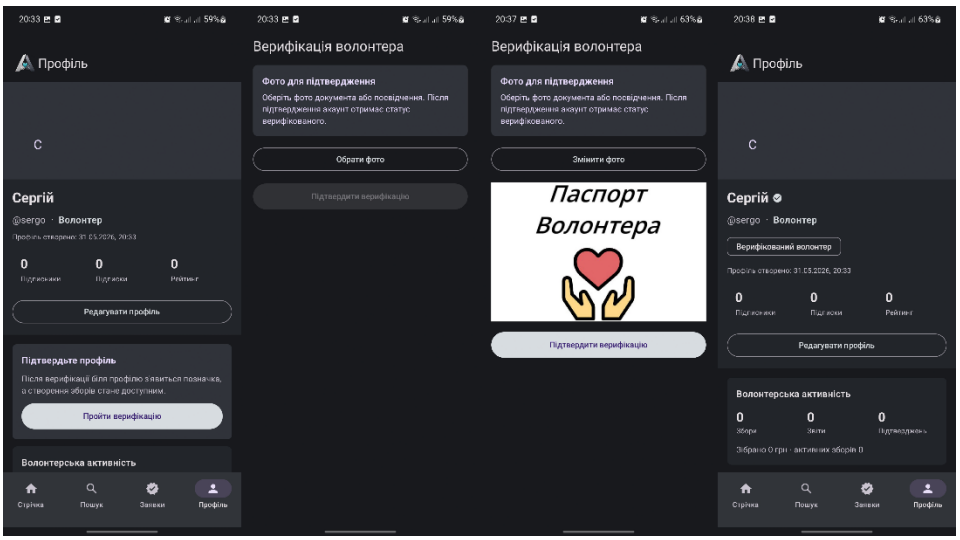

Рисунок 3.6 – Екран верифікації волонтера

Верифікація використовується як умова для створення зборів. Якщо користувач має роль VOLUNTEER, але не пройшов верифікацію, система не дозволяє створити кампанію. Це правило підвищує контроль над діями, які впливають на довіру в системі.

3.4 Реалізація зборів і соціальної стрічки

Збори реалізовано через модель Campaign, CampaignViewModel, FirestoreRepository і Compose-екрани стрічки, створення збору та деталей кампанії.

Стрічка відображає кампанії та звіти у вигляді соціального списку. Для цього використовується модель FeedItem, яка дозволяє об’єднувати різні типи записів. Користувач бачить актуальні збори, звіти волонтерів, статуси, зображення, дані авторів і коротку інформацію про прогрес.

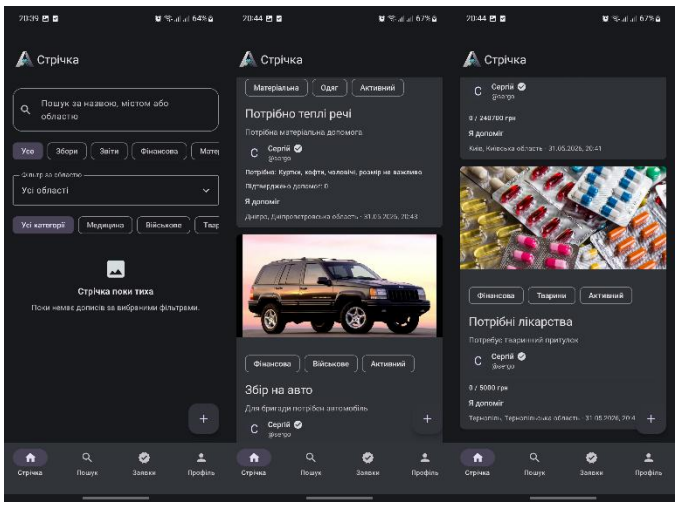

Рисунок 3.7 – Головна стрічка зборів і звітів

У стрічці реалізовано пошук і фільтрацію. Користувач може знайти записи за назвою, описом, містом, областю або автором. Також доступні фільтри за типом запису, фінансовою або матеріальною допомогою, регіоном, категорією, підписками, майже завершеними та завершеними зборами [7].

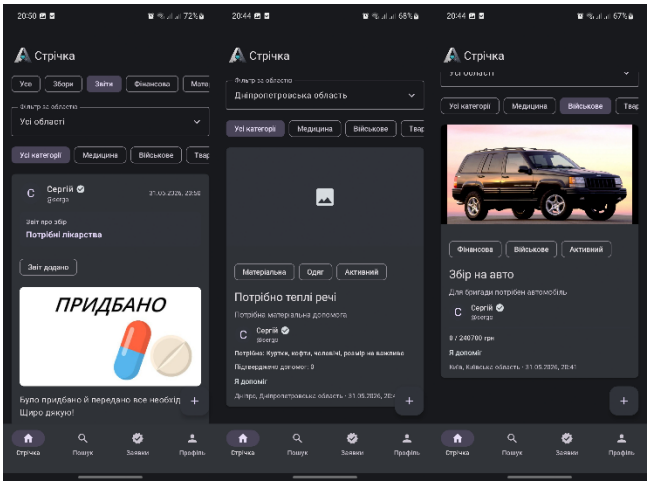

Рисунок 3.8 – Фільтрація стрічки за типом, регіоном і категорією

Створення збору доступне лише верифікованому волонтеру. На екрані створення користувач вводить назву, опис, тип, категорію, місто, область, реквізити або матеріальну ціль. Для фінансового збору вказується цільова сума. Для матеріального збору вказується опис потрібних речей.

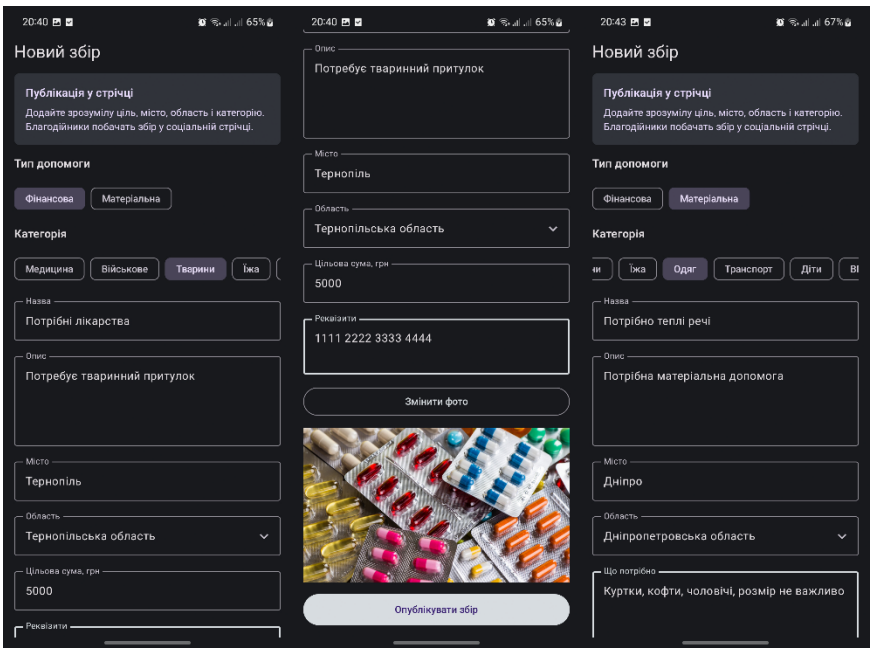

Рисунок 3.9 – Екран створення збору

Екран деталей збору показує повний опис потреби, тип збору, прогрес, статус, дані волонтера, реквізити, місто, область, категорію та доступні дії. Благодійник може скопіювати реквізити і перейти до форми "Я допоміг". Волонтер-власник може закрити збір або додати звіт після завершення.

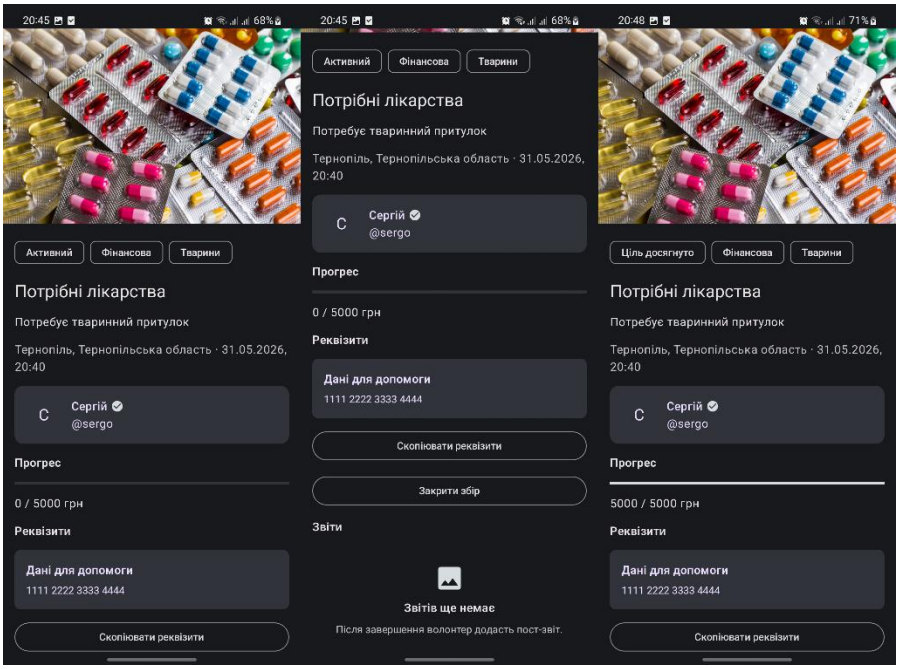

Рисунок 3.10 – Екран деталей збору

3.5 Реалізація заявок про допомогу

Заявки про допомогу реалізовано через модель HelpRequest, HelpRequestViewModel, HelpFormScreen і методи FirestoreRepository.

Благодійник може подати заявку тільки до активного збору. Для фінансового збору він вводить суму допомоги, а для матеріального збору описує передані речі. Також користувач може додати коментар і фото або скріншот підтвердження.

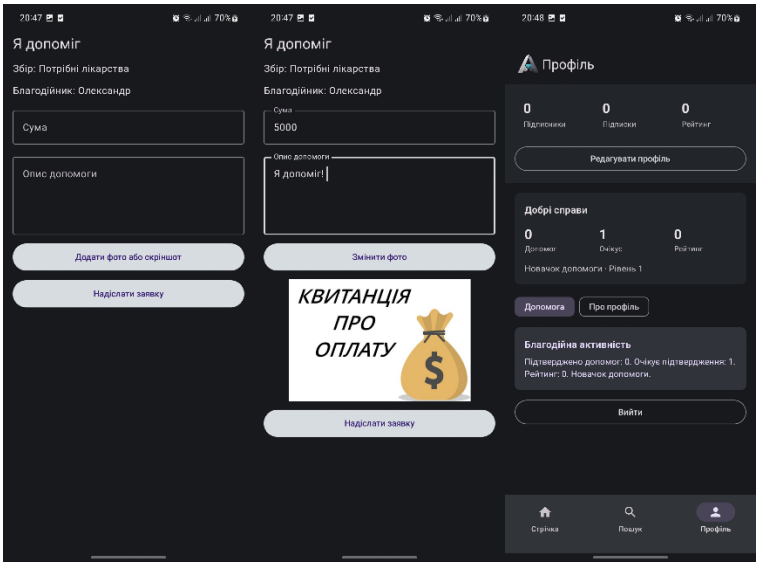

Рисунок 3.11 – Форма подання заявки "Я допоміг"

Після надсилання заявки створюється документ у колекції helpRequests зі статусом PENDING. У документі зберігаються дані кампанії, благодійника, волонтера, тип допомоги, сума або опис речей, коментар, URL скріншота і дата створення.

Волонтер переглядає заявки, що стосуються його власних зборів. У списку заявок відображаються дані благодійника, назва кампанії, сума або опис допомоги, коментар, фото підтвердження і статус.

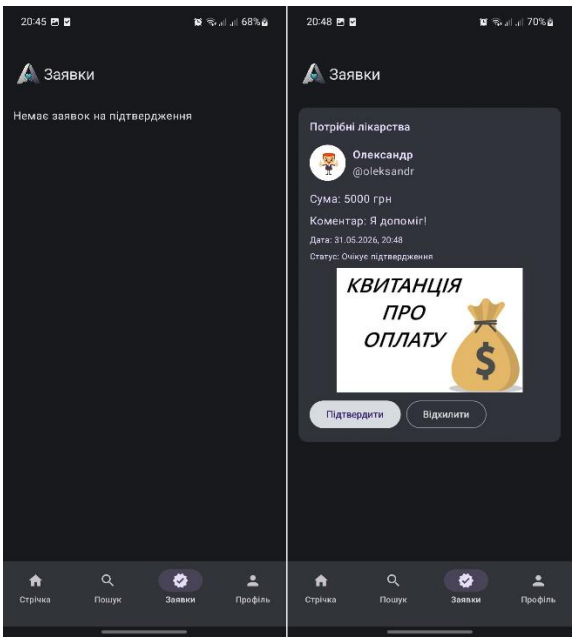

Рисунок 3.12 – Список заявок для волонтера

Підтвердження заявки виконується транзакційно. Якщо заявка фінансова, система змінює її статус на APPROVED і збільшує currentAmount кампанії на суму заявки. Якщо після цього currentAmount дорівнює або перевищує targetAmount, статус кампанії змінюється на GOAL_REACHED. Якщо заявка матеріальна, currentAmount збільшується на 1.

Відхилення заявки змінює її статус на REJECTED. У такому разі прогрес збору і рейтинг благодійника не змінюються.

3.6 Реалізація звітності

Звітність є важливою частиною прозорості Aidly. Після завершення збору або досягнення фінансової цілі волонтер може додати звіт. Звіт містить текстовий опис, прив’язку до кампанії, дані волонтера, дату створення і за потреби зображення.

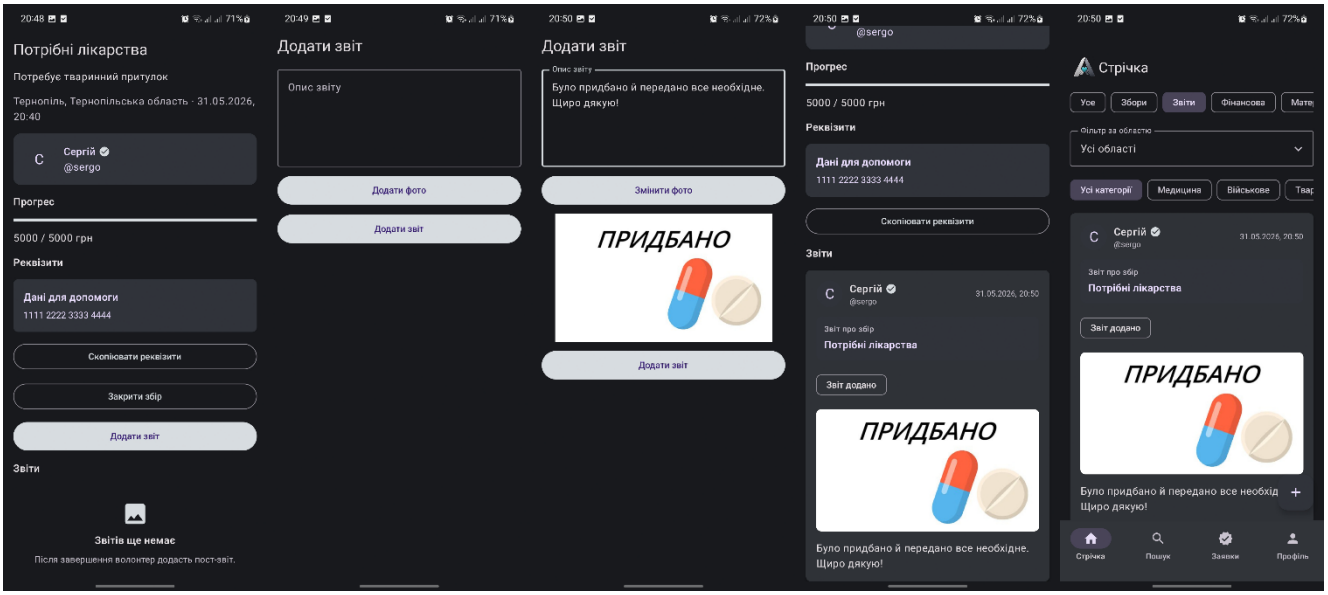

Рисунок 3.13 – Додавання та відображання звіту

Додавання звіту виконується через ReportViewModel і FirestoreRepository. Якщо користувач додає фото, воно завантажується у Cloudinary, а URL зберігається в документі Report. Після створення звіту статус кампанії змінюється на REPORTED.

Звіти відображаються у соціальній стрічці та в профілі волонтера. Це дозволяє користувачам не тільки бачити активні збори, а й перевіряти завершені ініціативи. Для волонтера звіти формують репутацію, а для благодійника підвищують довіру до майбутніх зборів.

3.7 Реалізація рейтингу та статистики

Рейтинг і статистика використовуються для відображення активності користувача. Для благодійника рейтинг формується на основі підтверджених заявок. За кожні 100 грн підтвердженої фінансової допомоги нараховується 1 бал. Наприклад, якщо благодійник надав 1000 грн і волонтер підтвердив заявку, благодійник отримує 10 балів рейтингу.

Матеріальна допомога може оцінюватися фіксованою кількістю балів за підтверджену заявку. Такий підхід дозволяє враховувати не лише фінансову участь, а й передачу речей або товарів.

У профілі відображається статистика: кількість зборів, активних кампаній, закритих кампаній, звітів, підтверджених допомог, заявок в очікуванні, загальна сума допомоги, рейтинг, рівень і титул. Це робить профіль не просто інформаційною сторінкою, а елементом довіри.

3.8 Реалізація завантаження зображень

Зображення використовуються в Aidly для аватарів, шапок профілю, фото зборів, скріншотів допомоги та фото звітів. Для вибору зображення використовується Android Activity Result API [6]. Після вибору файл читається через ContentResolver і передається в CloudinaryRepository.

CloudinaryRepository формує multipart-запит до Cloudinary, передає файл і отримує secure URL [12, 13]. Цей URL зберігається у відповідному документі Firestore. Для відображення зображень у Compose використовується Coil і компонент AsyncImage [14].

Такий підхід дозволяє не перевантажувати Firestore великими медіафайлами. У базі даних зберігаються лише текстові дані та URL, а самі зображення знаходяться у хмарному сховищі [10, 12].

3.9 Тестування мобільного застосунку

Тестування Aidly виконано для перевірки основних сценаріїв і бізнес-правил. Було перевірено запуск застосунку, реєстрацію, вхід, створення профілю, вибір ролі, верифікацію волонтера, створення збору, перегляд стрічки, фільтрацію, подання заявки допомоги, підтвердження заявки, оновлення прогресу, додавання звіту та відображення рейтингу. Ручне функціональне тестування проводилося для перевірки повних користувацьких сценаріїв: реєстрації, авторизації, створення профілю, верифікації волонтера, створення збору, перегляду стрічки, подання заявки про допомогу, підтвердження заявки, додавання звіту та відображення рейтингу.

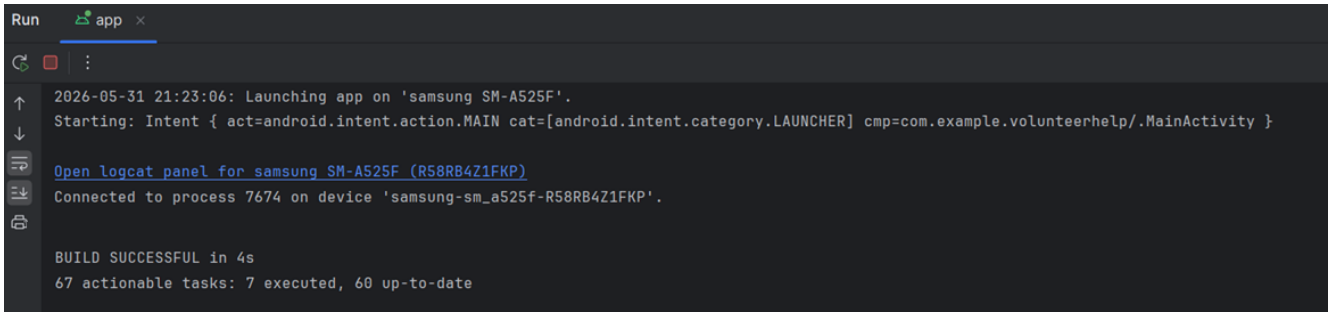

Рисунок 3.14 – Успішний білд та запуск застосунку

Модульне тестування виконувалося за допомогою local unit-тестів. Такі тести запускаються без Android-емулятора і дозволяють швидко перевірити логіку, яка не залежить від інтерфейсу користувача. У межах роботи було створено чотири набори unit-тестів: FormValidatorsTest.kt, CampaignStatusTest.kt, FeedItemTest.kt і ModelDefaultsTest.kt. Основні набори unit-тестів наведено в таблиці 3.1.

Таблиця 3.1 – Набори unit-тестів застосунку Aidly

Набір тестів	Призначення
FormValidatorsTest	Перевіряє валідацію імені, username, міста та максимальних довжин полів.
CampaignStatusTest	Перевіряє бізнес-логіку статусів збору, українські назви статусів і можливість виконання дій залежно від статусу.
FeedItemTest	Перевіряє визначення часу сортування елементів стрічки та правильне сортування змішаної стрічки.
ModelDefaultsTest	Перевіряє початкові значення моделей користувача, кампанії, заявки, звіту та статистики.

У тестах FormValidatorsTest.kt перевірено правильність введення основних полів профілю. Було перевірено як коректні значення, так і помилкові випадки: порожні поля, перевищення максимальної довжини, некоректні символи у

username та неправильне заповнення міста. Це важливо, оскільки профіль користувача є основою для подальшої роботи із застосунком.

У тестах CampaignStatusTest.kt перевірено логіку статусів збору. Зокрема, протестовано нормалізацію застарілого статусу COMPLETED, fallback на статус ACTIVE, українські підписи статусів, можливість закриття збору та можливість додавання звіту. Такі перевірки потрібні для стабільної роботи кампаній, оскільки статус збору визначає доступні дії користувача.

У тестах FeedItemTest.kt перевірено логіку формування соціальної стрічки. Оскільки у стрічці можуть відображатися як збори, так і звіти, важливо, щоб кожен елемент мав коректний час сортування. Тести підтвердили, що час береться з поля createdAt відповідної кампанії або звіту, а змішана стрічка може бути відсортована від найновіших записів до старіших.

У тестах ModelDefaultsTest.kt перевірено початкові значення основних моделей. Було підтверджено, що користувач за замовчуванням має роль DONOR, кампанія має тип FINANCIAL і статус ACTIVE, заявка створюється зі статусом PENDING, звіт є порожнім, а статистика має нульові значення. Такі перевірки зменшують ризик некоректної поведінки системи під час створення нових об’єктів.

Результати модульного тестування наведено в таблиці 3.2.

Таблиця 3.2 – Результати модульного тестування

№	Перевірка	Очікуваний результат	Фактичний результат
1	Валідація коректного імені користувача	Ім’я приймається системою	Виконано
2	Валідація порожнього імені	Система повертає помилку	Виконано
3	Валідація username з дозволеними символами	Username приймається системою	Виконано
4	Валідація username із забороненими символами	Система повертає помилку	Виконано

Продовження таблиці 3.2

5	Перевірка максимальної довжини полів	Надто довгі значення не приймаються	Виконано
6	Нормалізація legacy-статусу COMPLETED	Статус коректно обробляється системою	Виконано
7	Fallback для невизначеного статусу збору	Використовується статус ACTIVE	Виконано
8	Відображення українських назв статусів	Статуси мають зрозумілі українські назви	Виконано
9	Перевірка можливості закриття збору	Закриття доступне лише у допустимих станах	Виконано
10	Перевірка можливості додавання звіту	Звіт доступний після завершення або досягнення цілі	Виконано
11	Отримання часу сортування кампанії	Використовується createdAt кампанії	Виконано
12	Отримання часу сортування звіту	Використовується createdAt звіту	Виконано
13	Сортування змішаної стрічки	Новіші записи відображаються вище	Виконано
14	Дефолтна роль користувача	Встановлено DONOR	Виконано
15	Дефолтний тип і статус кампанії	Встановлено FINANCIAL і ACTIVE	Виконано
16	Дефолтний статус заявки	Встановлено PENDING	Виконано
17	Дефолтні значення звіту	Поля звіту мають початкові порожні значення	Виконано
18	Дефолтна статистика профілю	Числові показники дорівнюють нулю	Виконано

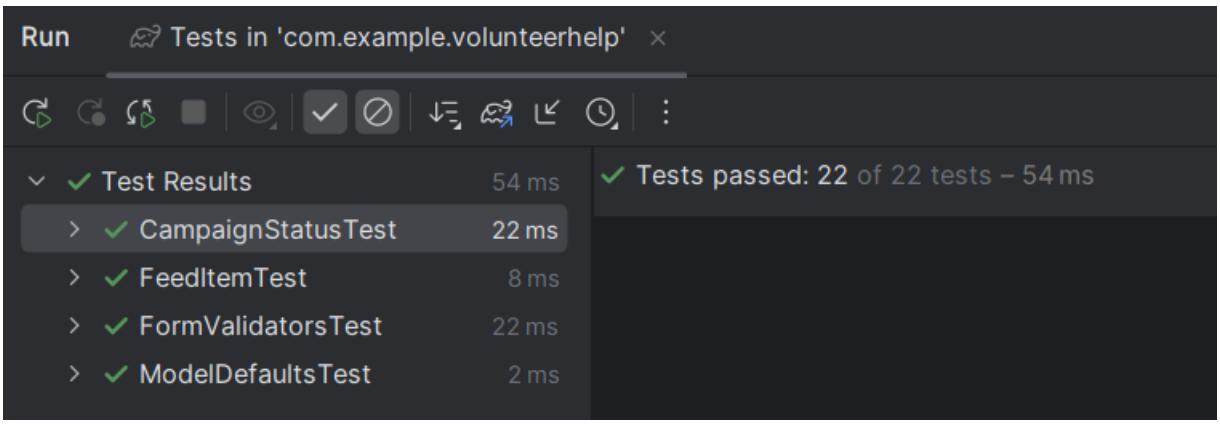

Рисунок 3.15 – Успішне виконання unit-тестів

Проведені unit-тести підтвердили коректність базових правил, які використовуються в різних частинах застосунку. Особливо важливими є тести статусів і моделей, оскільки вони впливають на створення зборів, формування стрічки, відображення звітів і роботу профілю користувача.

Результати тестування показали, що основні сценарії роботи Aidly виконуються коректно. Unit-тести підтвердили правильність базових правил і моделей, а ручне тестування підтвердило працездатність основних користувацьких сценаріїв у реальному інтерфейсі застосунку. Реєстрація, авторизація, створення зборів, подання заявок, підтвердження допомоги, оновлення прогресу, публікація звітів, пошук, фільтрація, підписки та відображення рейтингу працюють відповідно до визначених вимог.

4 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

У цьому розділі розглянуто інформаційне перевантаження як фактор ризику для життєдіяльності людини, а також електробезпеку і пожежну безпеку під час експлуатації комп'ютерної техніки.

4.1 Інформаційне перевантаження як фактор ризику для життєдіяльності людини

Інформаційне перевантаження виникає тоді, коли людина отримує більше інформації, ніж може якісно опрацювати, проаналізувати та використати для прийняття рішень [33, 36]. У сучасному цифровому середовищі це проявляється через постійний потік повідомлень, електронних листів, новин, технічної документації, навчальних матеріалів і робочих завдань.

Під час розумової праці інформаційне перевантаження є особливо відчутним. Програміст одночасно працює з кодом, середовищем розробки, помилками компіляції, документацією, базою даних, вимогами до проєкту та результатами тестування , що створює підвищене розумове й зорове навантаження [33, 36]. Якщо ці інформаційні потоки не впорядковані, зростає ризик втоми, втрати уваги, неправильних рішень і помилок у роботі.

Інформаційне перевантаження впливає не лише на продуктивність, а й на безпеку життєдіяльності. Людина у стані постійного розумового напруження повільніше реагує на небезпечні ситуації, гірше оцінює ризики, частіше допускає неуважність і може ігнорувати ознаки небезпеки [33, 36]. У робочих або побутових умовах це може призвести до неправильного використання обладнання, порушення правил електробезпеки або травмування.

Основними причинами інформаційного перевантаження є надмірна кількість завдань, багатозадачність, постійні сповіщення, відсутність чіткого плану роботи, тривала робота без перерв, недостатній сон і невпорядкованість джерел інформації.

Основі прояви інформаційного перевантаження наведена у таблиці 4.1.

Таблиця 4.1 – Основні прояви інформаційного перевантаження

Прояв	Можливі наслідки
Зниження концентрації	Помилки, повільніше виконання роботи
Втома	Зниження працездатності
Дратівливість	Погіршення емоційного стану
Погіршення пам’яті	Складність запам’ятовування деталей
Порушення сну	Хронічна втома, зниження уваги

Для зменшення ризиків інформаційного перевантаження потрібно раціонально організовувати робочий процес. Великі завдання доцільно ділити на менші етапи, визначати пріоритети, вести список задач і працювати з одним основним завданням у конкретний проміжок часу. Під час виконання складної роботи бажано вимикати другорядні сповіщення, закривати зайві вкладки браузера та залишати відкритими лише потрібні інструменти.

Важливе значення має режим праці та відпочинку. Після тривалої роботи за комп’ютером необхідно робити короткі перерви, змінювати положення тіла, давати відпочинок очам і виконувати легку розминку [33]. Це допомагає зменшити втому, підтримати концентрацію та знизити ризик помилок [36].

Отже, інформаційне перевантаження є важливим фактором ризику для життєдіяльності людини. Його профілактика полягає у плануванні роботи, обмеженні цифрових подразників, регулярних перервах і контролі обсягу інформації.

4.2 Електробезпека та пожежна безпека під час експлуатації комп’ютерної техніки

Комп’ютерна техніка широко використовується у різних сферах. До складу робочого місця можуть входити ноутбук або персональний комп’ютер, монітор,

клавіатура, миша, зарядний пристрій, мережеве обладнання, подовжувачі та інші електричні пристрої. Усі ці засоби працюють від електромережі, тому потребують дотримання правил електробезпеки та пожежної безпеки [28, 33, 37].

Електробезпека спрямована на захист людини від небезпечного впливу електричного струму та є складовою безпечних умов праці [28, 37]. Основними небезпечними факторами під час роботи з комп'ютерною технікою є пошкоджені кабелі, несправні розетки, перевантаження подовжувачів, неправильне підключення обладнання, використання неякісних зарядних пристроїв і потрапляння вологи на електроприлади [28, 33].

Ураження електричним струмом може призвести до опіків, судом, втрати свідомості, порушення роботи серця або смерті. Тому перед початком роботи необхідно перевіряти стан кабелів, вилок, розеток і подовжувачів. Якщо виявлено пошкодження ізоляції, іскріння, запах горілої пластмаси або перегрівання блока живлення, обладнання потрібно вимкнути і не використовувати до перевірки.

Під час експлуатації комп'ютера або ноутбука потрібно стежити за вентиляцією. Не можна закривати вентиляційні отвори, ставити ноутбук на м'які поверхні або використовувати техніку в умовах сильного запилення. Перегрівання може спричинити нестабільну роботу, пошкодження компонентів або займання.

Пожежа під час роботи з комп'ютерною технікою може виникнути через коротке замикання, перевантаження електромережі, перегрівання обладнання, накопичення пилу або використання несправних зарядних пристроїв [30, 37]. У приміщенні повинні бути вільні проходи, доступ до розеток, засобів пожежогасіння та евакуаційних виходів.

У разі пожежі необхідно повідомити людей поруч, викликати пожежно-рятувальну службу, вимкнути електроживлення, якщо це можна зробити без ризику для життя, і розпочати евакуацію [29, 30, 37]. Електрообладнання під напругою не можна гасити водою [30]. Для цього потрібно використовувати порошковий або вуглекислотний вогнегасник.

Основні правила безпечної роботи з комп'ютерною технікою:

1. Використовувати лише справне обладнання.

2. Не працювати з пошкодженими кабелями або вилками.
3. Не торкатися електрообладнання мокрими руками.
4. Не перевантажувати розетки та подовжувачі.
5. Не закривати вентиляційні отвори ноутбука або комп’ютера.
6. Не розміщувати напої поруч із технікою.
7. Не ремонтувати обладнання без відключення від живлення.
8. Не гасити водою електрообладнання під напругою.

Отже, електробезпека і пожежна безпека є важливими умовами безпечної експлуатації комп’ютерної техніки. Дотримання правил роботи з електрообладнанням дозволяє запобігти ураженню струмом, короткому замиканню, перегріванню пристроїв і пожежам [28, 30, 33, 37].

ВИСНОВКИ

У кваліфікаційній роботі виконано розроблення мобільного застосунку Aidly для координації волонтерської діяльності з використанням мови програмування Kotlin. Розроблений застосунок призначений для взаємодії між волонтерами та благодійниками, створення зборів, подання заявок про допомогу, підтвердження внесків, ведення профілів, підписок, рейтингу та звітності.

У першому розділі проаналізовано предметну область волонтерської діяльності. Визначено, що основними проблемами цифрової координації є розпорошеність інформації, відсутність структурованих статусів, складність підтвердження допомоги та недостатня прозорість звітності. Розглянуто існуючі інструменти, зокрема соціальні мережі, месенджери, банківські сервіси та веб-сайти благодійних фондів. Обґрунтовано потребу у спеціалізованому мобільному застосунку Aidly.

У другому розділі спроєктовано архітектуру застосунку. Визначено ролі VOLUNTEER і DONOR, сценарії використання, моделі даних, структуру Cloud Firestore, навігацію, інтерфейс користувача, стани кампаній і заявок, а також основні бізнес-правила. Для реалізації обрано архітектуру MVVM з Repository Pattern, що забезпечує розділення відповідальності між UI, ViewModel, репозиторіями та моделями.

У третьому розділі описано практичну реалізацію Aidly. Реалізовано авторизацію і реєстрацію через Firebase Authentication, профілі користувачів, унікальні username, демонстративну верифікацію волонтера, створення фінансових і матеріальних зборів, соціальну стрічку, пошук і фільтрацію, заявки "Я допоміг", підтвердження або відхилення допомоги, звіти, рейтинг, статистику і завантаження зображень через Cloudinary. Проведено тестування основних сценаріїв і бізнес-правил.

У четвертому розділі розглянуто питання безпеки життєдіяльності та охорони праці. Описано вимоги до організації робочого місця розробника,

електробезпеки, пожежної безпеки та профілактики психофізіологічного перевантаження під час роботи з комп’ютером.

У перспективі застосунок Aidly може бути розширений у напрямі підготовки до реального ринкового впровадження. Основними напрямами такого розвитку можуть стати інтеграція платіжних сервісів, впровадження push-сповіщень, створення адміністративної панелі, розширення процедури верифікації волонтерів і додавання системи модерації зборів. Такі можливості не змінюють основну логіку розробленого застосунку, а доповнюють його інструментами, необхідними для масштабного використання волонтерськими спільнотами.

Перед виходом на ринок доцільно провести підготовку системи до промислової експлуатації: перевірити стабільність роботи під реальним навантаженням, налаштувати моніторинг помилок, резервне копіювання даних, політику конфіденційності, правила користування та механізми підтримки користувачів. Це дозволить перевести Aidly до рівня практичного сервісу, придатного для використання реальними волонтерами та благодійниками.

Результатом роботи є функціональний Android-застосунок Aidly, який демонструє практичне застосування сучасних технологій Android-розробки, хмарної бази даних, архітектурних патернів і соціально орієнтованої логіки. Поставлену мету досягнуто, а основні завдання кваліфікаційної роботи виконано.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Kotlin Documentation [Електронний ресурс]. Режим доступу: https://kotlinlang.org/docs/home.html
2. Android Developers. Kotlin and Android [Електронний ресурс]. Режим доступу: https://developer.android.com/kotlin
3. Android Developers. Guide to app architecture [Електронний ресурс]. Режим доступу: https://developer.android.com/topic/architecture
4. Android Developers. Jetpack Compose documentation [Електронний ресурс]. Режим доступу: https://developer.android.com/jetpack/compose/documentation
5. Android Developers. Navigation with Compose [Електронний ресурс]. Режим доступу: https://developer.android.com/develop/ui/compose/navigation
6. Android Developers. ActivityResultContracts [Електронний ресурс]. Режим доступу: https://developer.android.com/reference/androidx/activity/result/contract/ActivityResultContracts
7. Material Design 3. Design system documentation [Електронний ресурс]. Режим доступу: https://m3.material.io/
8. Firebase Documentation. Add Firebase to your Android project [Електронний ресурс]. Режим доступу: https://firebase.google.com/docs/android/setup
9. Firebase Documentation. Firebase Authentication on Android [Електронний ресурс]. Режим доступу: https://firebase.google.com/docs/auth/android/start
10. Firebase Documentation. Cloud Firestore [Електронний ресурс]. Режим доступу: https://firebase.google.com/docs/firestore
11. Firebase Documentation. Cloud Firestore transactions and batched writes [Електронний ресурс]. Режим доступу: https://firebase.google.com/docs/firestore/manage-data/transactions

12. Cloudinary Documentation. Android image and video upload [Електронний ресурс]. Режим доступу: https://cloudinary.com/documentation/android_image_and_video_upload

13. Cloudinary Documentation. Android SDK integration [Електронний ресурс]. Режим доступу: https://cloudinary.com/documentation/android_integration

14. Coil Documentation. Compose [Електронний ресурс]. Режим доступу: https://coil-kt.github.io/coil/compose/

15. Gradle User Manual [Електронний ресурс]. Режим доступу: https://docs.gradle.org/current/userguide/userguide.html

16. Android Studio User Guide [Електронний ресурс]. Режим доступу: https://developer.android.com/studio/intro

17. OWASP Mobile Application Security Verification Standard [Електронний ресурс]. Режим доступу: https://mas.owasp.org/MASVS/

18. Guide to the Software Engineering Body of Knowledge (SWEBOK Guide). Version 4.0 / ed. H. Washizaki. IEEE Computer Society, 2024. 411 p.

19. Martin R. C. Clean Architecture: A Craftsman’s Guide to Software Structure and Design. Boston : Prentice Hall, 2017. 432 p.

20. Richards M., Ford N. Fundamentals of Software Architecture: An Engineering Approach. Sebastopol : O’Reilly Media, 2020. 422 p.

21. Freeman E., Robson E. Head First Design Patterns. 2nd ed. Sebastopol : O’Reilly Media, 2020. 672 p.

22. Fowler M. Patterns of Enterprise Application Architecture. Boston : Addison-Wesley, 2002. 560 p.

23. Burns B. Designing Distributed Systems. Sebastopol : O’Reilly Media, 2018. 166 p.

24. Bloch J. Effective Java. 3rd ed. Boston : Addison-Wesley, 2018. 416 p.

25. Gamma E., Helm R., Johnson R., Vlissides J. Design Patterns: Elements of Reusable Object-Oriented Software. Boston : Addison-Wesley, 1994. 395 p.

26. Про волонтерську діяльність : Закон України від 19.04.2011 № 3236-VI [Електронний ресурс]. Режим доступу: https://zakon.rada.gov.ua/laws/show/3236-17

27. Про захист персональних даних : Закон України від 01.06.2010 № 2297-VI [Електронний ресурс]. Режим доступу: https://zakon.rada.gov.ua/laws/show/2297-17

28. Про охорону праці : Закон України від 14.10.1992 № 2694-XII [Електронний ресурс]. Режим доступу: https://zakon.rada.gov.ua/laws/show/2694-12

29. Кодекс цивільного захисту України : Закон України від 02.10.2012 № 5403-VI [Електронний ресурс]. Режим доступу: https://zakon.rada.gov.ua/laws/show/5403-17

30. Правила пожежної безпеки в Україні : наказ Міністерства внутрішніх справ України від 30.12.2014 № 1417 [Електронний ресурс]. Режим доступу: https://zakon.rada.gov.ua/laws/show/z0252-15

31. ДСТУ 3008:2015. Інформація та документація. Звіти у сфері науки і техніки. Структура та правила оформлювання. Київ : ДП «УкрНДНЦ», 2016. 31 с.

32. ДСТУ 8302:2015. Інформація та документація. Бібліографічне посилання. Загальні положення та правила складання. Київ : ДП «УкрНДНЦ», 2016. 20 с.

33. ДСанПіН 3.3.2.007-98. Державні санітарні правила і норми роботи з візуальними дисплейними терміналами ЕОМ. Київ, 1998.

34. Методичні вказівки до виконання кваліфікаційної роботи бакалавра для здобувачів спеціальності 121 Інженерія програмного забезпечення, всіх форм навчання / укладачі: Михалик Д. М., Цуприк Г. Б., Бревус В. М. Тернопіль : Тернопільський національний технічний університет імені Івана Пулюя, 2024. 45 с.

35. Репозиторій програмного коду мобільного застосунку Aidly [Електронний ресурс]. Режим доступу: https://github.com/k0shaq/aidly

36. Левченко О. Г., Землянська О. В., Праховнік Н. А., Зацарний В. В. Безпека життєдіяльності та цивільний захист : підручник. 2-ге вид. Київ : Каравела, 2021. 268 с.

37. Левченко О. Г., Полукаров О. І., Арламов О. Ю., Полукаров Ю. О., Землянська О. В. Охорона праці та цивільний захист : підручник для студентів бакалаврату. Київ : Каравела, 2021. 352 с.

ДОДАТКИ

Додаток А – Лістинг коду із кваліфікаційною роботою бакалавра

Лістинг коду розміщений у віддаленому репозиторію GitHub за посиланням: https://github.com/k0shaq/aidly.