

**Міністерство освіти і науки України
Луцький національний технічний університет
Факультет комп'ютерних та інформаційних технологій
Кафедра інженерії програмного забезпечення**

**КВАЛІФІКАЦІЙНА РОБОТА
ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «БАКАЛАВР»**

**РОЗРОБКА ANDROID-ЗАСТОСУНКУ ДЛЯ УПРАВЛІННЯ ЗАПИСАМИ
ПРО РЕМОНТ ТРАНСПОРТНИХ ЗАСОБІВ НА СТО**

**DEVELOPMENT OF AN ANDROID APPLICATION FOR MANAGING
VEHICLE REPAIR RECORDS AT SERVICE STATIONS**

спеціальність 121 «Інженерія програмного забезпечення»
освітня програма «Інженерія програмного забезпечення»

Виконав: здобувач вищої освіти
групи ІПЗ-42
Пасевич Віктор Васильович

Керівник:
Корень Віталій Володимирович

Кваліфікаційну роботу
допущено до захисту
«___» _____ 20__ р.
Гарант освітньої програми:
к.т.н., доцент
Ліщина Наталія Миколаївна

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет комп'ютерних та інформаційних технологій

Кафедра інженерії програмного забезпечення

Ступінь вищої освіти бакалавр

Галузь знань: 12 «Інформаційні технології»

Спеціальність: 121 «Інженерія програмного забезпечення»

Освітня програма: «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ

Завідувач кафедри

«__» _____ 20__ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

Пасевичу Віктору Васильовичу

(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи «Розробка Android-застосунку для управління записами про ремонт транспортних засобів на СТО»

Керівник роботи: асистент Корень В. В.

затверджені наказом закладу вищої освіти від «31» грудня 2025 р. № 541/01-02

2. Строк подання здобувачем вищої освіти кваліфікаційної роботи бакалавра «29» травня 2026 р.

3. Вихідні дані до роботи: предметна область управління записами про ремонт транспортних засобів на СТО, технології розробки мобільного програмного забезпечення для Android (Kotlin, Android SDK), хмарні сервіси Firebase Authentication та Firebase Firestore, засоби проєктування мобільних інформаційних систем, вимоги до інформаційних систем та забезпечення інформаційної безпеки..

4. Зміст розрахунково-пояснювальної записки (перелік питань, що потрібно розробити): аналіз сучасного стану проблем автоматизації процесу запису транспортних засобів на ремонт та технічне обслуговування, постановка завдання, формування вимог до розроблюваного Android-застосунку, обґрунтування вибору технологій та інструментальних засобів розробки, практична реалізація Android-застосунку, розробка структури бази даних Firebase Firestore, обґрунтування принципів захисту інформаційно-комп'ютерної системи, реалізація рольового контролю доступу, тестування та налагодження програмного забезпечення.

5. Перелік графічного матеріалу: 19 рисунків, 6 лістингів, 4 таблиці.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
Аналіз предметної області	<i>Корень В. В.</i>		
Специфікація вимог до розробленої системи	<i>Корень В. В.</i>		
Розробка об'єкта проектування	<i>Корень В. В.</i>		
Нормоконтроль	<i>Суринович О. М.</i>		
Гарант ОП	<i>Ліщина Н. М.</i>		
Показник запозичень тексту	<u> </u> %		
Академічна доброчесність	<i>Корень В. В.</i>		

7. Дата видачі завдання «3» лютого 2026 р.

№	Назва етапів кваліфікаційної роботи бакалавра	Строк виконання етапів роботи	Примітка
1	Огляд літературних джерел по темі кваліфікаційної роботи бакалавра	до 14.02.2026 р.	
2	Аналіз проблеми розробки та впровадження об'єкту проектування	до 03.03.2026 р.	
3	Обґрунтування вибору шляхів, технологій і засобів вирішення поставленого завдання	до 14.03.2026 р.	
4	Розробка функціонально-структурної схеми роботи об'єкта проектування та проектування бази даних	до 14.04.2026 р.	
5	Практична реалізація об'єкта проектування та розробка бази даних	до 20.04.2026 р.	
6	Тестування та налагодження об'єкта проектування	до 04.05.2026 р.	
7	Здача чистового варіанту кваліфікаційної роботи бакалавра на кафедрі	до 29.05.2026 р.	

Здобувач вищої освіти _____

Віктор ПАСЕВИЧ

Керівник кваліфікаційної роботи _____

Віталій КОРЕНЬ

АНОТАЦІЯ

Пасевич В. В. Розробка Android-застосунку для управління записами про ремонт транспортних засобів на СТО. Рукопис. Кваліфікаційна робота бакалавра ОП «Інженерія програмного забезпечення» спеціальності «Інженерія програмного забезпечення». Луцький національний технічний університет. Луцьк, 2026.

Кваліфікаційна робота бакалавра складається зі вступу, трьох розділів, висновків, списку використаних джерел.

У першому розділі проведено дослідження предметної області, пов'язаної з автоматизацією процесу запису транспортних засобів на ремонт та технічне обслуговування, проаналізовано існуючі програмні рішення та визначено їхні переваги й недоліки, а також обґрунтовано актуальність розробки мобільного Android-застосунку.

У другому розділі сформовано функціональні та нефункціональні вимоги до системи, виконано проектування структури програмного забезпечення та обґрунтовано вибір інструментальних засобів і технологій розробки. Для створення застосунку використано мову програмування Kotlin, середовище Android Studio та хмарні сервіси Firebase.

У третьому розділі описано процес реалізації Android-застосунку, розроблено механізм авторизації та реєстрації користувачів, реалізовано role-based access між користувачем та адміністратором системи, структуру бази даних Firebase Firestore, механізми пошуку, створення, редагування та видалення записів на ремонт транспортних засобів. Також розглянуто питання забезпечення захисту інформації, обробки помилок та наведено результати тестування програмного забезпечення.

У висновках підсумовано результати виконаної роботи, підтверджено досягнення поставленої мети та повне виконання поставлених завдань.

Ключові слова: Android-застосунок, Kotlin, Firebase, Firebase Firestore, Firebase Authentication, role-based access, мобільний застосунок, Android Studio.

ABSTRACT

Pasevych Viktor. Development of an Android Application for Managing Vehicle Repair Records at Service Stations. Manuscript. Bachelor's qualification work in the Educational Program "Software Engineering" in the specialty "Software Engineerin". Lutsk National Technical University. Lutsk, 2026.

The bachelor's qualification work consists of an introduction, three chapters, conclusions, a list of references.

The first chapter investigates the subject area related to the automation of vehicle repair and maintenance appointment processes, analyzes existing software solutions and identifies their advantages and disadvantages, as well as substantiates the relevance of developing a mobile Android application.

The second chapter defines the functional and non-functional requirements of the system, presents the software design structure, and justifies the choice of development tools and technologies. The application was developed using the Kotlin programming language, Android Studio environment, and Firebase cloud services.

The third chapter describes the implementation process of the Android application, including user authentication and registration mechanisms, role-based access between users and administrators, the Firebase Firestore database structure, and mechanisms for searching, creating, editing, and deleting vehicle repair appointments. The chapter also considers information security issues, error handling mechanisms, and presents the results of software testing.

The conclusions summarize the results of the performed work, confirm the achievement of the stated goal, and the complete fulfillment of the assigned tasks.

Keywords: Android application, Kotlin, Firebase, Firebase Firestore, Firebase Authentication, role-based access, mobile application, Android Studio.

ЗМІСТ

ВСТУП.....	7
РОЗДІЛ 1 АНАЛІЗ МЕТОДІВ РОЗРОБКИ ІНФОРМАЦІЙНИХ СИСТЕМ ТА ПОСТАНОВКА ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ	9
1.1 Аналіз сучасного стану проблеми	9
1.2 Постановка завдання на кваліфікаційну роботу бакалавра	11
РОЗДІЛ 2 СПЕЦИФІКАЦІЯ ВИМОГ ДО РОЗРОБЛЮВАНОЇ СИСТЕМИ...	13
2.1 Аналіз, визначення вимог до розроблюваного програмного забезпечення та проєктування програмного забезпечення	13
2.2 Вибір засобів, методів і алгоритмів вирішення поставленого завдання ...	18
РОЗДІЛ 3 РОЗРОБКА ANDROID-ЗАСТОСУНКУ ДЛЯ УПРАВЛІННЯ ЗАПИСАМИ ПРО РЕМОНТ ТРАНСПОРТНИХ ЗАСОБІВ НА СТО.....	25
3.1 Практична реалізація об'єкта проєктування.....	25
3.2 Розробка бази даних	31
3.3 Захист інформаційно-комп'ютерної системи	39
3.4 Тестування та налагодження інформаційно-комп'ютерної системи	43
ВИСНОВКИ	48
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	50

ВСТУП

Актуальність теми. У сучасних умовах розвитку інформаційних технологій мобільні застосунки активно використовуються для автоматизації різноманітних бізнес-процесів, зокрема у сфері обслуговування транспортних засобів [1]. Одним із важливих напрямів цифровізації є автоматизація процесу запису клієнтів на ремонт та технічне обслуговування автомобілів. Використання мобільних застосунків дозволяє значно спростити взаємодію між клієнтами та автосервісами, забезпечити централізоване зберігання інформації та підвищити ефективність управління записами.

На практиці у багатьох невеликих станціях технічного обслуговування процес запису клієнтів здійснюється вручну – за допомогою телефонних дзвінків, паперових журналів або електронних таблиць. Такий підхід створює низку проблем, серед яких: складність контролю актуального стану записів, висока ймовірність втрати інформації, відсутність централізованого доступу до даних, а також труднощі в організації роботи персоналу.

Існуючі програмні рішення для управління автосервісами часто мають надлишкову функціональність, є складними у використанні або потребують значних фінансових витрат на впровадження та підтримку. Крім того, багато з них не адаптовані під потреби невеликих СТО та не забезпечують простого механізму керування записами через мобільні пристрої.

Особливо важливим аспектом є забезпечення багатокористувацької роботи системи та захисту даних. Для цього необхідно реалізувати механізми автентифікації користувачів, розмежування прав доступу та безпечного зберігання інформації у хмарному середовищі.

У зв'язку з цим виникає необхідність у розробці сучасного Android-застосунку для управління записами про ремонт транспортних засобів із використанням хмарних технологій, що забезпечить зручне керування записами, багатокористувацький доступ та синхронізацію даних у режимі реального часу.

Метою кваліфікаційної роботи бакалавра є розробка Android-застосунку для управління записами про ремонт транспортних засобів із використанням хмарних технологій Firebase Authentication та Firebase Firestore.

Для досягнення поставленої мети необхідно вирішити такі завдання:

- проаналізувати сучасні підходи до автоматизації процесу запису транспортних засобів на ремонт;
- визначити функціональні та нефункціональні вимоги до програмного забезпечення;
- обґрунтувати вибір технологій та інструментальних засобів розробки;
- спроектувати структуру Android-застосунку та систему зберігання даних;
- реалізувати систему автентифікації та авторизації користувачів;
- реалізувати механізми створення, редагування, видалення та перегляду записів;
- забезпечити рольовий контроль доступу між користувачем та адміністратором;
- провести тестування та налагодження програмного забезпечення.

Об'єктом дослідження є процес управління записами про ремонт та технічне обслуговування транспортних засобів.

Предметом дослідження є методи та засоби розробки мобільних Android-застосунків із використанням хмарних технологій Firebase.

Практичне значення роботи полягає у створенні Android-застосунку, який може бути використаний станціями технічного обслуговування для автоматизації процесу запису клієнтів на ремонт транспортних засобів, зберігання інформації у хмарному середовищі та забезпечення багатокористувацького доступу до системи.

РОЗДІЛ 1

АНАЛІЗ МЕТОДІВ РОЗРОБКИ ІНФОРМАЦІЙНИХ СИСТЕМ ТА ПОСТАНОВКА ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

1.1 Аналіз сучасного стану проблеми

У сучасних умовах розвитку цифрових технологій автоматизація процесів обслуговування клієнтів стала важливою складовою ефективної роботи підприємств різних сфер діяльності, зокрема станцій технічного обслуговування транспортних засобів. Використання інформаційних систем дозволяє оптимізувати процес керування записами клієнтів, покращити організацію роботи персоналу та забезпечити швидкий доступ до необхідної інформації.

На сьогодні у багатьох невеликих автосервісах процес запису клієнтів на ремонт транспортних засобів все ще виконується вручну. Для цього використовуються телефонні дзвінки, паперові журнали, блокноти або електронні таблиці. Такий підхід має ряд недоліків, серед яких:

- висока ймовірність втрати інформації;
- складність швидкого пошуку записів;
- відсутність централізованого зберігання даних;
- труднощі контролю актуального статусу ремонту;
- неможливість одночасної роботи декількох користувачів із системою;
- відсутність захисту даних та розмежування прав доступу.

Використання паперових носіїв та неавтоматизованих методів обліку також ускладнює взаємодію між працівниками СТО та клієнтами. Адміністраторам необхідно витратити додатковий час на обробку записів, уточнення інформації та контроль виконання робіт. Крім того, у разі великої кількості клієнтів виникає ризик помилок при формуванні графіку ремонту транспортних засобів.

Для вирішення подібних проблем сьогодні існує велика кількість програмних продуктів та онлайн-сервісів, призначених для автоматизації процесу запису клієнтів. До таких систем належать CRM-системи, сервіси

онлайн-бронювання та спеціалізовані платформи для автосервісів. Однак більшість із них мають ряд недоліків: високу вартість використання, надлишкову функціональність, складність налаштування або відсутність адаптації під потреби невеликих СТО. Порівняння існуючих систем керування записами наведено у таблиці 1.1.

Таблиця 1.1 – Порівняння існуючих систем керування записами

Система	Переваги	Недоліки
EasyWeek	Онлайн запис	Платна система
Booksy	Мобільний доступ	Складний інтерфейс
Google Sheets	Простота використання	Відсутність ролей
Розроблений застосунок	Firebase, полі, Android	Потребує інтернету

Сучасні мобільні технології дозволяють створювати Android-застосунки, які забезпечують швидкий доступ до системи через смартфон та підтримують багатокористувацьку роботу [2]. Особливо перспективним є використання хмарних технологій, зокрема сервісів Firebase Authentication та Firebase Firestore. Firebase Authentication забезпечує безпечну автентифікацію користувачів та підтримку механізмів авторизації, а Firebase Firestore дозволяє організувати централізоване хмарне зберігання даних із можливістю синхронізації між пристроями у режимі реального часу [3, 4].

Для забезпечення ефективної організації роботи СТО доцільно передбачити розмежування прав доступу між звичайними користувачами та адміністраторами. Звичайні користувачі повинні мати можливість працювати лише з власними записами, тоді як адміністраторам доцільно надати розширені права для перегляду всіх записів, зміни статусів ремонту та керування інформацією в системі.

Таким чином, розробка Android-застосунку для управління записами про ремонт транспортних засобів є актуальним завданням, оскільки дозволяє автоматизувати процеси обліку, забезпечити централізоване зберігання

інформації, реалізувати багатокористувацький доступ та підвищити ефективність роботи автосервісу.

Використання мобільного Android-застосунку дозволяє забезпечити швидкий доступ до системи із використанням смартфона або планшета, що є особливо важливим для працівників станцій технічного обслуговування транспортних засобів. Користувачі отримують можливість оперативно переглядати інформацію про записи та взаємодіяти із системою незалежно від місця перебування.

1.2 Постановка завдання на кваліфікаційну роботу бакалавра

Метою кваліфікаційної роботи бакалавра є розробка Android-застосунку для управління записами про ремонт транспортних засобів на СТО із використанням хмарних технологій Firebase.

Для досягнення поставленої мети необхідно вирішити такі завдання:

- проаналізувати сучасні підходи до автоматизації процесу запису транспортних засобів на ремонт;
- визначити функціональні та нефункціональні вимоги до програмного забезпечення;
- обґрунтувати вибір технологій та інструментальних засобів розробки;
- спроектувати структуру Android-застосунку та систему зберігання даних;
- реалізувати систему автентифікації та авторизації користувачів;
- реалізувати механізми створення, редагування, видалення та перегляду записів;
- забезпечити рольовий контроль доступу між користувачем та адміністратором;
- провести тестування та налагодження програмного забезпечення.

У результаті виконання кваліфікаційної роботи бакалавра має бути створений Android-застосунок, який забезпечуватиме ефективне керування

записами на ремонт транспортних засобів, підтримуватиме багатокористувацький режим роботи та дозволить централізовано зберігати інформацію у хмарному середовищі.

РОЗДІЛ 2

СПЕЦИФІКАЦІЯ ВИМОГ ДО РОЗРОБЛЮВАНОЇ СИСТЕМИ

2.1 Аналіз, визначення вимог до розроблюваного програмного забезпечення та проєктування програмного забезпечення

Перед початком розробки Android-застосунку для управління записами про ремонт транспортних засобів проведено аналіз вимог до програмного забезпечення та визначено основні функціональні можливості системи. Формування вимог є важливим етапом проєктування програмного продукту, оскільки дозволяє визначити структуру системи, логіку взаємодії користувачів із застосунком та необхідний набір функцій.

Основною метою розроблюваного застосунку є забезпечення зручного механізму створення, редагування та контролю записів на ремонт транспортних засобів із використанням мобільних Android-пристроїв та хмарних технологій Firebase.

У системі передбачено два основних типи користувачів – користувач та адміністратор.

Користувач має можливість:

- проходити реєстрацію та авторизацію у системі;
- переглядати власні записи;
- створювати нові записи на ремонт;
- редагувати власні записи та видаляти;
- переглядати статус ремонту транспортного засобу.

Адміністратор, окрім функцій звичайного користувача, має додаткові можливості: переглядати всі записи користувачів, змінювати статус ремонту та видаляти записи із системи.

Архітектуру системи доцільно будувати на основі клієнт-серверного підходу, що дозволяє організувати взаємодію між мобільним застосунком та хмарними сервісами. У запропонованій архітектурі Android-застосунок розглядається як клієнтська частина системи, тоді як Firebase Authentication

доцільно використовувати для реалізації механізму автентифікації користувачів, а Firebase Firestore – для централізованого зберігання та синхронізації даних. Загальна архітектура розробленої системи наведена на рисунку 2.1.

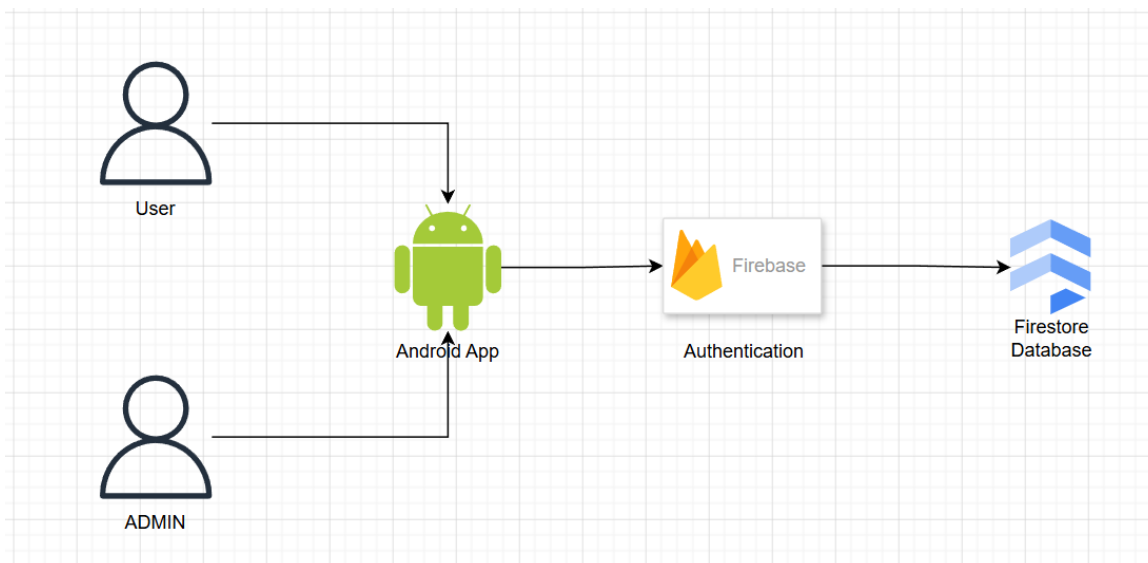


Рисунок 2.1 – Архітектура Android-застосунку для управління записами про ремонт транспортних засобів

Окрім загальної архітектури системи, визначено архітектуру Android-застосунку. Під час проєктування особлива увага приділяється забезпеченню зрозумілої структури програмного коду, розподілу функціональних обов’язків між окремими екранами та організації взаємодії із хмарними сервісами Firebase. Запропонований підхід сприяє спрощенню супроводу програмного забезпечення, підвищенню зручності його подальшого розвитку та забезпеченню стабільної роботи основних функцій системи.

Для побудови інтерфейсу користувача обрано використання Activity та XML-розмітки. Такий підхід дозволяє розділити функціональні можливості системи між окремими екранами та забезпечити зрозумілу структуру програмного забезпечення. Передбачається виділення окремих екранів для авторизації користувачів, перегляду записів, створення та редагування записів, а також роботи з профілем користувача. Розподіл функціональних можливостей між окремими екранами сприяє зручній навігації та спрощує взаємодію

користувача із системою. Архітектуру Android-застосунку наведено на рисунку 2.2.

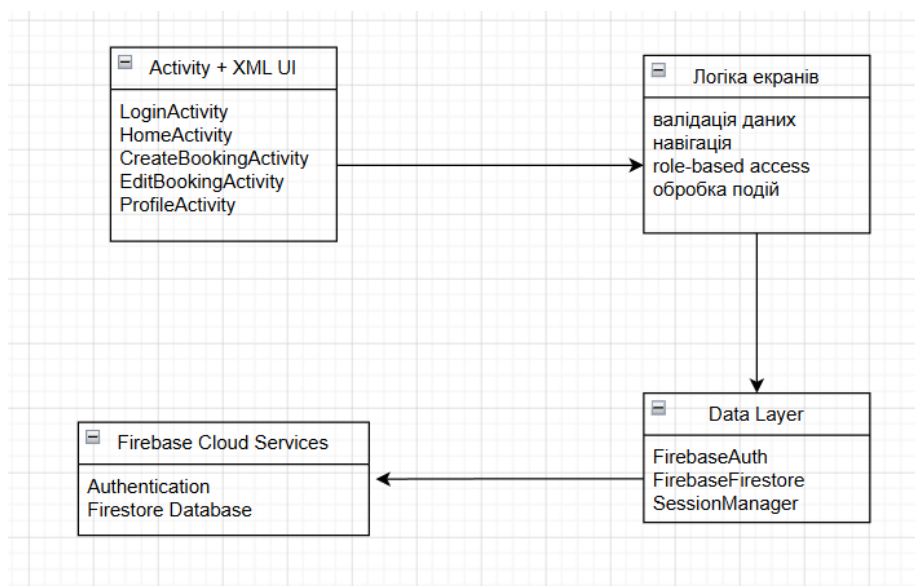


Рисунок 2.2 – Архітектура Android-застосунку

Для проєктування Android-застосунку обрано спрощену багаторівневу архітектуру, яка передбачає розподіл системи на рівень інтерфейсу користувача, рівень логіки та рівень даних. На рівні інтерфейсу доцільно використовувати Activity та XML-розмітку, рівень логіки призначений для обробки подій, перевірки введених даних, навігації та реалізації рольового контролю доступу, а рівень даних забезпечує взаємодію із сервісами Firebase Authentication та Firebase Firestore. Такий підхід є доцільним для невеликого Android-застосунку, оскільки дозволяє забезпечити зрозумілу структуру програмного коду, спростити супровід програмного забезпечення та прискорити процес розробки. Для подальшого масштабування системи доцільним є використання окремого Repository Layer та архітектурного підходу MVVM.

Для побудови інтерфейсу користувача доцільно використовувати компоненти Material Design, які забезпечують сучасний зовнішній вигляд застосунку та покращують зручність взаємодії користувача із системою. Для організації переходів між основними екранами передбачається використання навігаційної структури, що охоплює сторінки авторизації, профілю користувача,

перегляду записів та створення нових записів. Такий підхід сприяє підвищенню зручності використання програмного забезпечення та забезпечує логічну організацію функціональних можливостей системи. Навігаційну структуру застосунку наведено на рисунку 2.3.

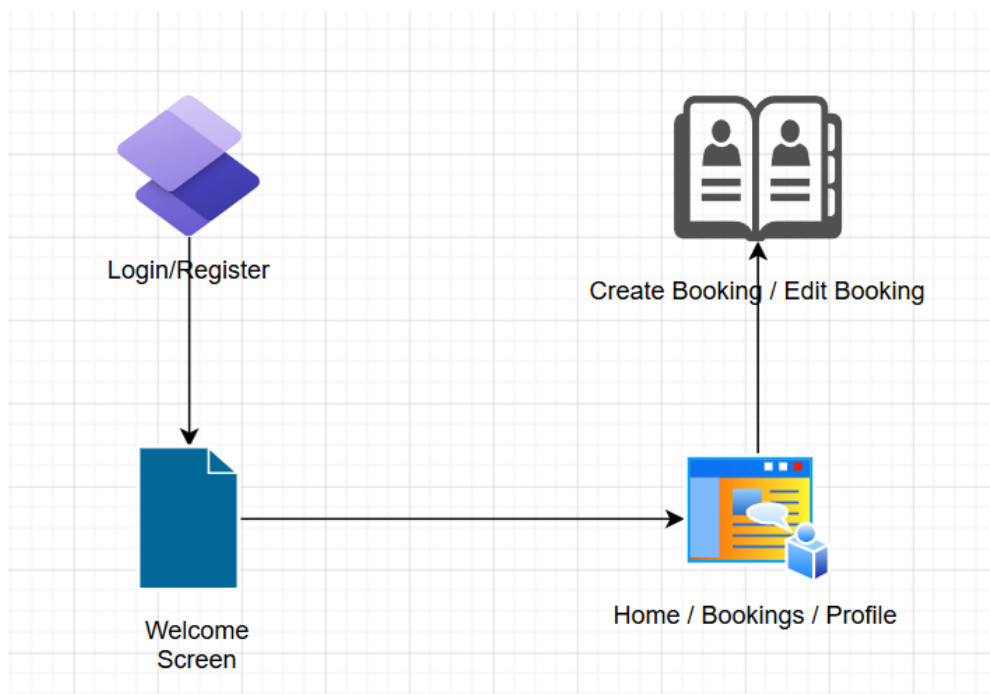


Рисунок 2.3 – Навігаційна структура Android-застосунку

Одним із важливих етапів проєктування системи є визначення функціональних та нефункціональних вимог до програмного забезпечення. До функціональних вимог належать:

- реєстрація та авторизація користувачів;
- створення записів на ремонт транспортних засобів;
- редагування та видалення записів;
- зміна статусу ремонту;
- пошук записів;
- підтримка багатокористувацького режиму роботи.

До нефункціональних вимог належать:

- зручність використання мобільного застосунку;
- стабільність роботи системи;

- безпечне зберігання даних;
- підтримка синхронізації даних у хмарному середовищі;
- швидкодія інтерфейсу користувача;
- підтримка сучасних версій Android.

Основні функціональні можливості системи залежно від ролі користувача наведено у таблиці 2.1.

Таблиця 2.1 – Функціональні можливості системи

Функція	USER	ADMIN
Реєстрація	Так	Так
Створення запису	Так	Так
Редагування запису	Так	Так
Видалення запису	Тільки свої	Всі
Зміна статусу	Ні	Так
Перегляд усіх записів	Ні	Так

Зручність використання мобільного застосунку є важливою нефункціональною вимогою, оскільки система орієнтована на щоденне використання працівниками СТО та клієнтами. Інтерфейс застосунку повинен забезпечувати швидкий доступ до основних функцій системи, просту навігацію між екранами та зрозуміле відображення інформації.

Стабільність роботи системи є необхідною умовою коректного функціонування Android-застосунку. Застосунок повинен коректно обробляти запити користувачів, підтримувати роботу із Firebase Firestore та забезпечувати стабільну синхронізацію даних між пристроями.

Швидкодія інтерфейсу користувача також є важливою нефункціональною вимогою, оскільки застосунок повинен швидко завантажувати списки записів, обробляти пошук інформації та забезпечувати плавну роботу інтерфейсу навіть при великій кількості записів у системі.

2.2 Вибір засобів, методів і алгоритмів вирішення поставленого завдання

Для розробки Android-застосунку для управління записами про ремонт транспортних засобів обрано сучасні інструментальні засоби та технології, які забезпечують стабільність роботи системи, підтримку мобільних платформ і можливість інтеграції із хмарними сервісами. Вибір відповідних технологій здійснюється з урахуванням функціональних вимог до програмного забезпечення, особливостей мобільної платформи Android та необхідності забезпечення багатокористувацького режиму роботи системи.

Для розробки Android-застосунку обрано мову програмування Kotlin. Вибір даної мови обумовлений тим, що Kotlin є офіційною мовою розробки Android-застосунків та рекомендована компанією Google для створення сучасного мобільного програмного забезпечення [5]. У порівнянні з мовою Java, Kotlin забезпечує більш лаконічний синтаксис, меншу кількість шаблонного коду, підвищену безпечність роботи з даними та зручні засоби реалізації асинхронних операцій. Завдяки зазначеним перевагам Kotlin є доцільним вибором для розробки Android-застосунку та сприяє підвищенню якості програмного забезпечення.

Для реалізації асинхронної обробки даних обрано механізм Kotlin Coroutines. Вибір даного підходу обумовлений можливістю виконання мережових запитів та операцій обробки даних без блокування головного потоку застосунку. У порівнянні з традиційними підходами до багатопотоковості Kotlin Coroutines характеризуються простішим синтаксисом, меншим споживанням системних ресурсів та спрощують підтримку програмного коду. Використання даного механізму сприяє підвищенню продуктивності та стабільності роботи Android-застосунку.

Як середовище розробки обрано Android Studio – офіційне інтегроване середовище для створення Android-застосунків [6]. Вибір Android Studio обумовлений її повною підтримкою мови Kotlin, наявністю вбудованих засобів

проектування інтерфейсу, емуляції мобільних пристроїв, тестування та налагодження програмного забезпечення. Крім того, Android Studio забезпечує підтримку інтеграції із сервісами Google та Firebase, що спрощує розробку мобільних застосунків із використанням хмарних технологій.

Для реалізації механізму автентифікації та авторизації користувачів розглянуто два підходи: створення власної системи керування обліковими записами та використання готового хмарного сервісу. Для даного проекту обрано Firebase Authentication, оскільки цей сервіс забезпечує високий рівень безпеки, підтримує сучасні механізми автентифікації та дозволяє уникнути розробки окремої серверної частини. Запропоновану схему взаємодії Android-застосунку із сервісом Firebase Authentication наведено на рисунку 2.4.

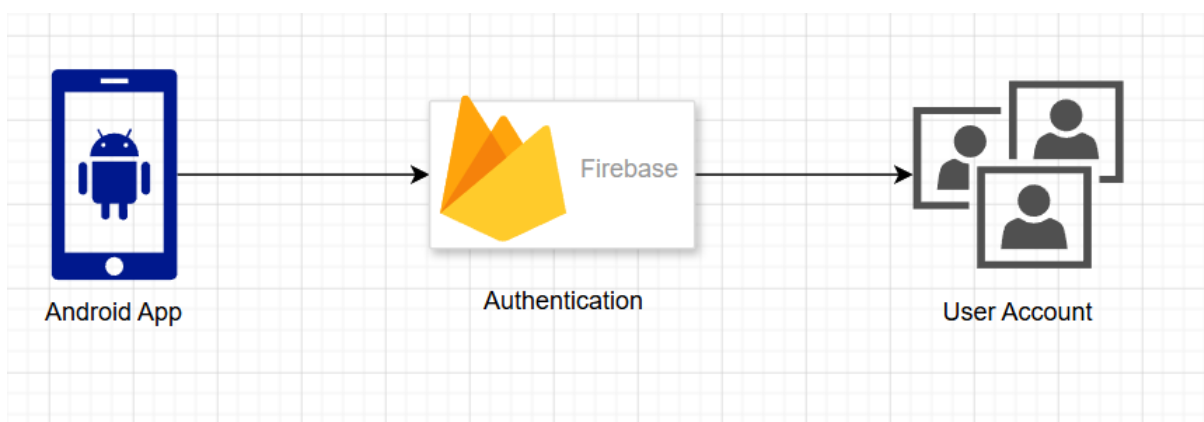


Рисунок 2.4 – Схема автентифікації користувачів у системі

Firebase Authentication підтримує автоматичне керування сесіями користувачів та забезпечує відновлення автентифікованого стану після повторного запуску Android-застосунку. Завдяки цьому відпадає необхідність у створенні окремого механізму збереження авторизації користувача. Для локального зберігання допоміжних даних, зокрема ролі користувача та параметрів інтерфейсу, може використовуватися SessionManager на основі механізму SharedPreferences.

Для організації зберігання даних розглянуто використання локальних баз даних, реляційних серверних СУБД та хмарних NoSQL-рішень. Для

розроблюваної системи обрано Firebase Firestore, оскільки він забезпечує централізоване зберігання інформації, підтримує синхронізацію даних у режимі реального часу та не потребує розробки окремого серверного програмного забезпечення. Вибір Firebase Firestore дозволяє зменшити складність програмної реалізації системи та забезпечити доступ до актуальних даних із різних мобільних пристроїв.

Додатковою перевагою Firebase Firestore є його інтеграція з іншими сервісами Firebase, зокрема Firebase Authentication. Це дозволяє спростити реалізацію багатокористувацького режиму роботи та організувати доступ до даних відповідно до ролі користувача. Крім того, Firestore підтримує автоматичну синхронізацію інформації між пристроями, що є важливим для мобільних застосунків із хмарною архітектурою.

Однією з переваг Firebase Firestore є відсутність необхідності розробки та супроводу власного серверного програмного забезпечення. Використання даного сервісу дозволяє зменшити складність програмної системи, спростити процес розгортання застосунку та зосередити увагу на реалізації основних функціональних можливостей. Крім того, застосування хмарного сервісу сприяє скороченню витрат на адміністрування та підтримку серверної інфраструктури.

Firebase Firestore також підтримує механізми контролю доступу до даних, що дозволяють організувати рольове розмежування прав доступу між користувачами системи. Такий підхід забезпечує обмеження доступу до інформації відповідно до ролі користувача та сприяє підвищенню рівня безпеки програмного забезпечення. Звичайний користувач має доступ лише до власних записів, тоді як адміністратор отримує розширені можливості роботи з даними системи.

Для зберігання інформації про записи на ремонт транспортних засобів обрано документно-орієнтовану модель даних, що використовується у Firebase Firestore. Такий підхід забезпечує гнучке зберігання інформації та дозволяє легко адаптувати структуру даних до можливих змін функціональних вимог системи. Структуру зберігання даних у Firebase Firestore наведено на рисунку 2.5.

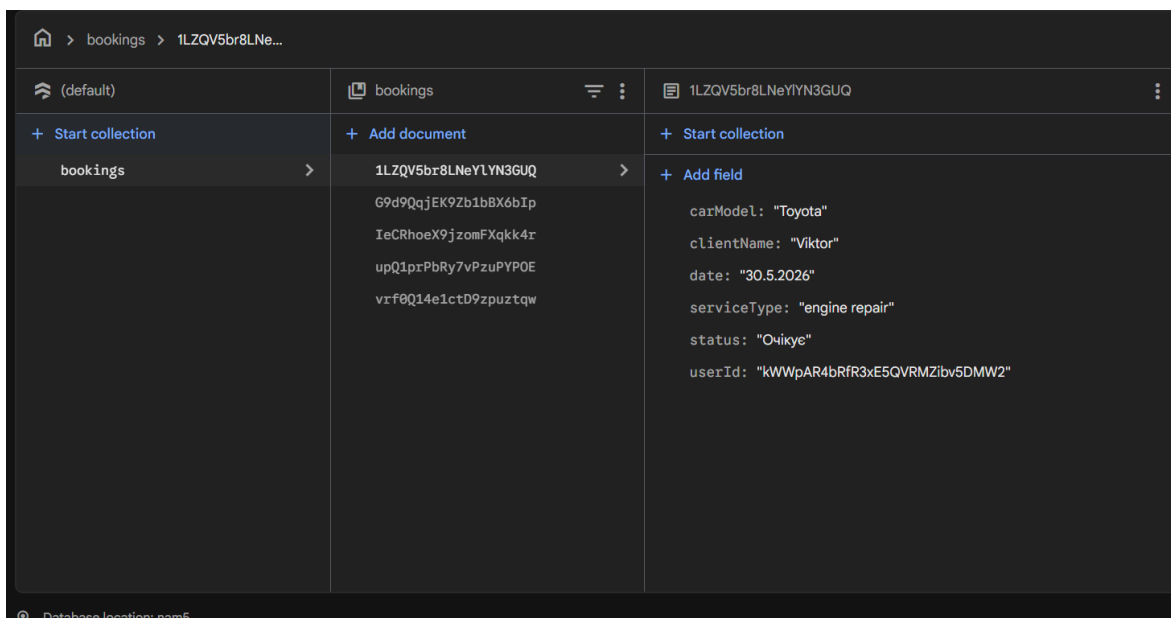


Рисунок 2.5 – Структура колекції bookings у Firebase Firestore

Для побудови інтерфейсу користувача обрано концепцію Material Design [7], яка є офіційною системою проєктування інтерфейсів від Google. Вибір Material Design обумовлений можливістю забезпечення єдиного стилю оформлення Android-застосунку, покращення зручності взаємодії користувача із системою та створення сучасного й зрозумілого інтерфейсу. Додатковою перевагою є наявність великої кількості готових компонентів, що спрощує процес розробки мобільного застосунку.

Для роботи зі списками даних обрано компонент RecyclerView, який є стандартним засобом Android SDK для відображення великої кількості однотипних елементів. Вибір даного компонента обумовлений його високою продуктивністю та можливістю ефективного використання системних ресурсів під час роботи зі значними обсягами даних. Перевагою RecyclerView є повторне використання елементів інтерфейсу, що дозволяє зменшити споживання пам'яті пристрою та забезпечити плавне відображення списків. Даний компонент є доцільним для реалізації сторінки перегляду записів на ремонт транспортних засобів.

Для реалізації механізму пошуку записів доцільно використовувати компонент SearchView, який інтегрується зі стандартними засобами Android

SDK та забезпечує швидку фільтрацію інформації. Вибір даного компонента обумовлений простотою реалізації, зручністю використання та можливістю ефективного пошуку даних без створення додаткових елементів інтерфейсу.

Для організації навігації між основними екранами Android-застосунку обрано компонент Bottom Navigation. Вибір даного компонента обумовлений його поширеним використанням у сучасних мобільних застосунках та можливістю забезпечення швидкого доступу до ключових функцій системи. Застосування нижньої панелі навігації сприяє покращенню користувацького досвіду та спрощує взаємодію із програмним забезпеченням.

Для реалізації інтерфейсу Android-застосунку обрано класичний підхід із використанням Activity та XML-розмітки. Незважаючи на поширення сучасного підходу Jetpack Compose, використання Activity та XML залишається актуальним для невеликих і середніх Android-проектів завдяки простоті реалізації, великій кількості навчальних матеріалів та зрозумілій структурі програмного коду. Даний підхід повністю задовольняє функціональні вимоги розроблюваної системи та не потребує додаткового ускладнення архітектури. Використання окремих Activity дозволяє логічно розподілити функціональні можливості між екранами авторизації, перегляду записів, створення записів та профілю користувача, що спрощує супровід і подальший розвиток програмного забезпечення.

Для наочного представлення функціональних можливостей системи та розмежування прав доступу між користувачами використовується UML-діаграму варіантів використання (рис. 2.6). Діаграма відображає основні функції Android-застосунку та взаємодію звичайного користувача й адміністратора із системою. Використання UML-діаграми дозволяє на етапі проектування визначити набір функціональних можливостей програмного забезпечення та встановити взаємозв'язки між користувачами і системою.



Рисунок 2.6 – UML-діаграма варіантів використання системи

Для організації контролю версій проєкту обрано систему Git та хмарний сервіс GitHub [8, 9]. Вибір Git обумовлений його широким використанням у сфері розробки програмного забезпечення, можливістю відстеження змін у вихідному коді та підтримкою колективної розробки. GitHub обрано як платформу для централізованого зберігання репозиторію проєкту, резервного копіювання вихідного коду та забезпечення доступу до актуальної версії програмного забезпечення. Використання даних засобів сприяє підвищенню надійності процесу розробки та спрощує подальший супровід програмного продукту. Використані технології розробки наведені в таблиці 2.2.

Таблиця 2.2 – Використані технології розробки

Технологія	Призначення
Kotlin	Розробка Android-застосунк
Android Studio	Середовище розробки
Firebase Authentication	Авторизація користувачі
Firebase Firestore	Хмарне зберігання даних
Material Design	Інтерфейс користувача
GitHub	Контроль версій

Використання даних засобів дозволяє підвищити надійність процесу розробки та спрощує супровід програмного продукту.

РОЗДІЛ 3

РОЗРОБКА ANDROID-ЗАСТОСУНКУ ДЛЯ УПРАВЛІННЯ ЗАПИСАМИ ПРО РЕМОНТ ТРАНСПОРТНИХ ЗАСОБІВ НА СТО

3.1 Практична реалізація об'єкта проєктування

Практична реалізація Android-застосунку для управління записами про ремонт транспортних засобів виконувалася із використанням мови програмування Kotlin та середовища розробки Android Studio. Основною метою реалізації створення багатокористувацької мобільної системи, яка забезпечує керування записами на ремонт транспортних засобів, підтримує автентифікацію користувачів та використовує хмарне зберігання даних.

У процесі розробки реалізовано систему авторизації та реєстрації користувачів із використанням Firebase Authentication. Користувач має можливість створити власний обліковий запис, після чого виконати вхід до системи за допомогою електронної пошти та пароля. Інтерфейси авторизації та реєстрації користувачів побудовані із використанням компонентів Material Design та забезпечують зручну взаємодію користувача із системою.

У розробленому застосунку сервіс Firebase Authentication використовується для підтримки авторизації та автоматичного відновлення сесії користувача після повторного запуску системи. Додатково SessionManager та SharedPreferences застосовуються для локального збереження допоміжної інформації про роль користувача та параметри роботи Android-застосунку.

Для покращення взаємодії користувача із системою у застосунку реалізовано головний інформаційний екран, який містить короткий опис автосервісу, перелік доступних послуг та контактну інформацію. Даний екран забезпечує швидкий доступ користувача до основних можливостей системи та дозволяє підвищити інформативність інтерфейсу. Інтерфейс головного інформаційного екрана наведений на рисунку 3.1.

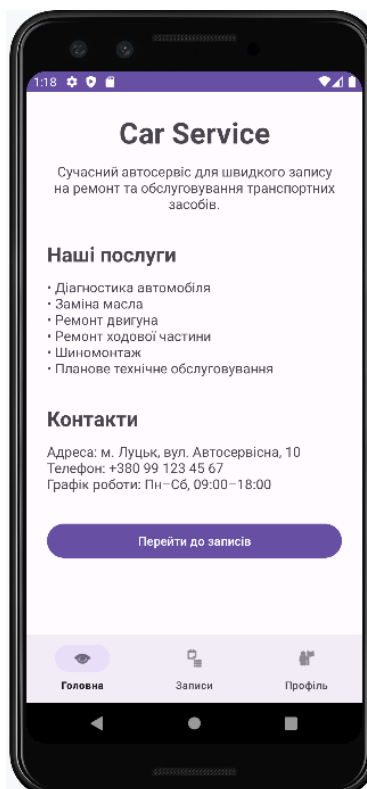


Рисунок 3.1 – Головний інформаційний екран Android-застосунку

Під час розробки інтерфейсу особлива увага приділялась простоті навігації та зручності використання застосунку. Для цього використано компоненти Material Design та Bottom Navigation, які забезпечують швидкий перехід між основними екранами системи [10].

Після переходу до сторінки записів користувач отримує доступ до списку створених записів на ремонт транспортних засобів. Для відображення списку записів використано компонент RecyclerView [11]. На даному екрані реалізовано відображення інформації про клієнта, транспортний засіб, тип послуги, дату ремонту та поточний статус виконання робіт.

Для забезпечення швидкого доступу до необхідної інформації записи відображаються у вигляді структурованого списку, що дозволяє користувачу оперативно переглядати актуальні дані про заплановані ремонтні роботи. Використання компонента RecyclerView забезпечує ефективне відображення великої кількості записів та оптимізує використання ресурсів мобільного

пристрою. Крім того, користувач має можливість швидко знаходити необхідні записи та контролювати поточний стан виконання робіт.

Для підвищення зручності використання застосунку реалізовано кольорове відображення статусів ремонту, індикатор завантаження даних та пошук записів. Інтерфейс сторінки перегляду записів наведений на рисунку 3.2.

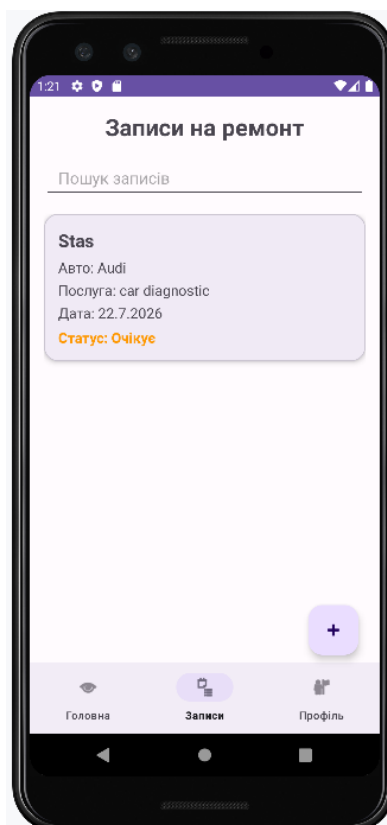


Рисунок 3.2 – Екран перегляду записів на ремонт

Для підвищення зручності роботи із системою у застосунку реалізовано механізм пошуку записів за ключовими словами. Це дозволяє швидко знаходити необхідні записи серед великої кількості даних та спрощує взаємодію користувача із системою.

Механізм пошуку реалізовано із використанням компонента `SearchView`, який забезпечує фільтрацію інформації у режимі реального часу [12]. Під час введення користувачем ключових слів список записів автоматично оновлюється та відображає лише ті елементи, які відповідають параметрам пошуку. Реалізація пошуку дозволяє значно підвищити швидкість роботи із системою, особливо при

великій кількості записів у Firebase Firestore. Інтерфейс пошуку записів наведений на рисунку 3.3.

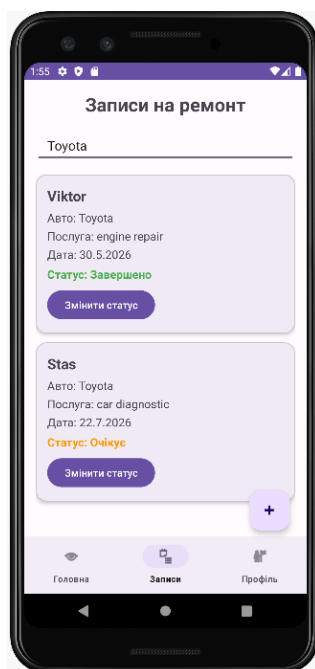


Рисунок 3.3 – Пошук записів у системі

Використання `SearchView` також покращує зручність користувацького інтерфейсу та робить взаємодію із Android-застосунком більш комфортною для користувачів системи.

Для створення нових записів на ремонт транспортних засобів у застосунку реалізовано окремий екран введення даних. Введена користувачем інформація зберігається у Firebase Firestore та автоматично синхронізується між пристроями.

Під час розробки форми створення запису особлива увага приділялась зручності введення інформації та простоті користувацького інтерфейсу. Для введення дати ремонту у застосунку реалізовано `DatePickerDialog`, що дозволяє користувачу швидко обрати необхідну дату без ручного введення інформації [13].

Після успішного створення запису користувач отримує інформаційне повідомлення про результат виконання операції, а новий запис автоматично

відображається у списку записів системи. Результат збереження та синхронізації записів у Firebase Firestore наведений на рисунку 3.4.

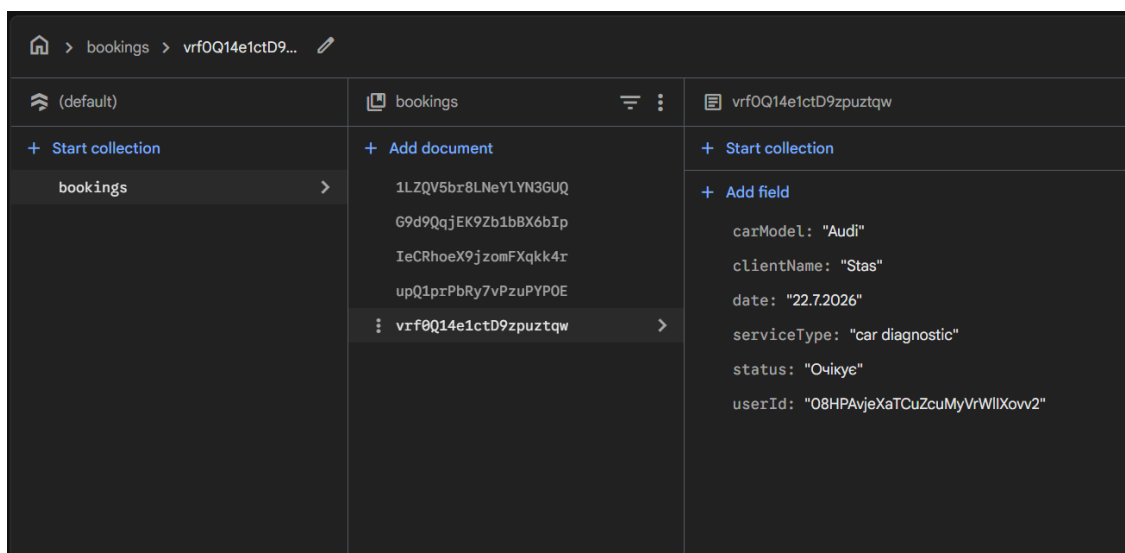


Рисунок 3.4 – Результат збереження записів у Firebase Firestore

Для забезпечення безпечної взаємодії із хмарною базою даних Firebase Firestore у застосунку реалізовано механізми контролю доступу до даних. Доступ до інформації здійснюється лише після успішної автентифікації користувача. Це дозволяє уникнути несанкціонованого перегляду та модифікації записів.

У системі реалізовано механізм рольового контролю доступу, який дозволяє обмежувати функціональні можливості Android-застосунку відповідно до ролі користувача. Для реалізації role-based access у системі використовуються ролі USER та ADMIN.

Звичайний користувач має доступ лише до власних записів на ремонт транспортних засобів. Користувач може створювати нові записи, редагувати інформацію про ремонт та переглядати поточний статус виконання робіт. При цьому користувач не має можливості переглядати або змінювати записи інших користувачів системи.

Адміністратор системи, окрім функцій звичайного користувача, отримує додаткові можливості керування інформацією у системі. Адміністратор може

переглядати всі записи користувачів, змінювати статус ремонту транспортних засобів та видаляти будь-які записи із Firebase Firestore (рис. 3.5-3.6).

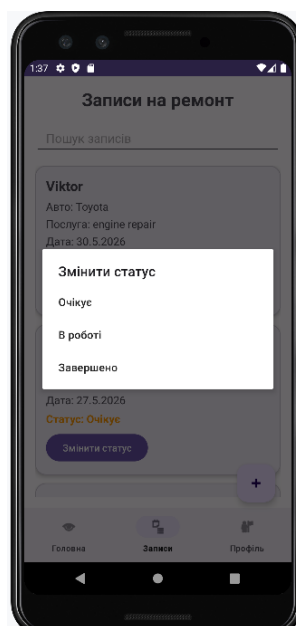


Рисунок 3.5 – Зміна статусу ремонту транспортного засобу адміністратором

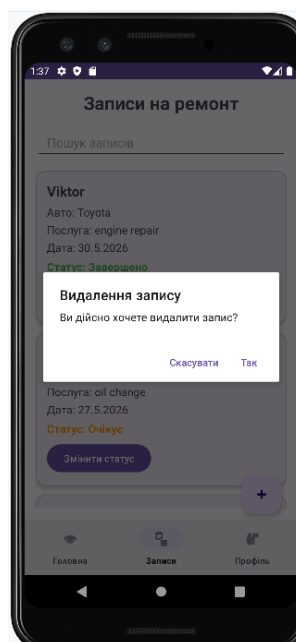


Рисунок 3.6 – Вікно підтвердження видалення запису

Використання механізму role-based access дозволяє підвищити рівень безпеки Android-застосунку та забезпечує більш ефективну організацію роботи системи. Такий підхід також дозволяє уникнути несанкціонованого доступу до

інформації інших користувачів. Розмежування прав доступу забезпечує виконання лише тих операцій, які відповідають ролі конкретного користувача.

У застосунку також реалізовано сторінку профілю користувача, яка містить інформацію про поточного користувача та забезпечує можливість завершення сеансу роботи із системою. На сторінці профілю відображаються основні відомості про користувача, що дозволяє швидко перевірити дані облікового запису. Для адміністраторів додатково відображається інформація про роль користувача. Наявність окремої сторінки профілю підвищує зручність використання системи та забезпечує швидкий доступ до функції виходу з облікового запису. Інтерфейс сторінки профілю наведений на рисунку 3.7.

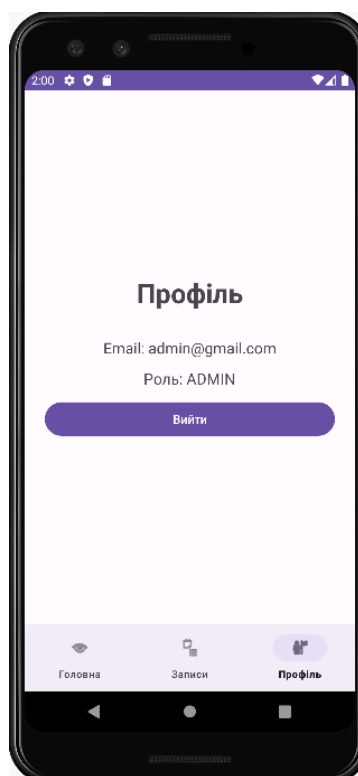


Рисунок 3.7 – Інтерфейс профілю користувача

3.2 Розробка бази даних

Для забезпечення централізованого зберігання інформації у розроблюваному Android-застосунку використано хмарну NoSQL базу даних Firebase Firestore. Використання Firestore дозволяє організувати

багатокористувацький доступ до даних, забезпечити синхронізацію інформації між пристроями та підтримувати роботу системи у режимі реального часу.

У Firebase Firestore дані зберігаються у вигляді колекцій та документів. Для реалізації системи управління записами про ремонт транспортних засобів створено колекцію `bookings`, у якій зберігається інформація про всі записи користувачів.

Кожен документ колекції `bookings` містить набір полів, необхідних для коректної роботи Android-застосунку та забезпечення взаємодії між користувачами системи. Поле `userId` використовується для ідентифікації користувача та реалізації механізму `role-based access`, що дозволяє відображати звичайним користувачам лише власні записи, а адміністратору – усі записи системи.

Поле `clientName` містить інформацію про ім'я клієнта, який створює запис на ремонт транспортного засобу. Для збереження інформації про транспортний засіб використовується поле `carModel`, яке дозволяє відображати модель автомобіля у списку записів та спрощує пошук необхідної інформації.

Поле `serviceType` використовується для збереження типу послуги або виду ремонтних робіт, які необхідно виконати. Для організації графіку ремонту у системі використовується поле `date`, яке містить дату виконання ремонтних робіт.

Для контролю виконання ремонту транспортного засобу використовується поле `status`. У системі реалізовано декілька статусів ремонту, такі як: «Очікує», «В роботі» та «Завершено». Це дозволяє користувачам відстежувати поточний стан виконання ремонтних робіт у режимі реального часу.

Поле `documentId` використовується для ідентифікації документа у Firebase Firestore та забезпечує можливість подальшого редагування, оновлення або видалення запису із системи.

Для взаємодії Android-застосунку із Firebase Firestore реалізовано механізм додавання, оновлення, видалення та отримання записів користувачів. Додавання нового запису до Firebase Firestore здійснюється шляхом створення документа у

колекції bookings. Фрагмент програмного коду додавання запису наведений у лістингу 3.1.

Лістинг 3.1 – Додавання запису до Firebase Firestore

```
val booking = hashMapOf(
    "userId" to currentUser.uid,
    "clientName" to clientName,
    "carModel" to carModel,
    "serviceType" to serviceType,
    "date" to date,
    "status" to "Очікує"
)

FirebaseFirestore.getInstance()
    .collection("bookings")
    .add(booking)
    .addOnSuccessListener {
        Toast.makeText(
            this,
            "Запис створено",
            Toast.LENGTH_SHORT
        ).show()
        finish()
    }
    .addOnFailureListener {
        Toast.makeText(
            this,
            "Помилка створення запису",
            Toast.LENGTH_SHORT
        ).show()
    }
```

Кінець лістингу 3.1

Після введення користувачем необхідної інформації та успішного збереження даних у Firebase Firestore створений запис відображається у системі.

Створення нового запису є однією з основних функцій Android-застосунку, оскільки саме за допомогою даного механізму користувач формує заявку на проведення ремонтних робіт. Для підвищення зручності роботи всі необхідні дані вводяться в межах одного екрану, що дозволяє мінімізувати кількість дій користувача під час взаємодії із системою. Після успішного збереження інформації створений запис стає доступним для подальшого перегляду та редагування відповідно до прав доступу користувача. Інтерфейс створення запису на ремонт транспортного засобу наведений на рисунку 3.8.

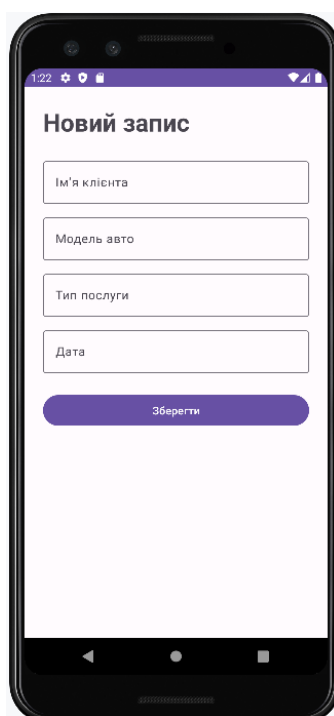


Рисунок 3.8 – Інтерфейс створення запису на ремонт

Для забезпечення багатокористувацької роботи системи кожен запис містить ідентифікатор користувача, що дозволяє відображати звичайним користувачам лише власні записи, а адміністратору – усі записи системи. Фрагмент програмного коду фільтрації записів користувачів наведений у лістингу 3.2.

Лістинг 3.2 – Реалізація role-based access у системі

```
if (currentUserRole == "admin") {

    FirebaseFirestore.getInstance()
        .collection("bookings")
        .get()

        .addOnSuccessListener { result ->
            bookingList.clear()

            for (document in result) {
                val booking = document.toObject(Booking::class.java)
                bookingList.add(booking)
            }
            bookingAdapter.notifyDataSetChanged()
        }
} else {
    FirebaseFirestore.getInstance()
        .collection("bookings")
        .whereEqualTo("userId", currentUser.uid)
        .get()

        .addOnSuccessListener { result ->

            bookingList.clear()

            for (document in result) {
                val booking = document.toObject(Booking::class.java)
                bookingList.add(booking)
            }
            bookingAdapter.notifyDataSetChanged()
        }
}
```

Реалізований механізм рольового контролю доступу забезпечує відображення користувачам лише дозволеної інформації відповідно до їх ролі. Такий підхід дозволяє обмежити доступ до даних інших користувачів та підвищує рівень безпеки Android-застосунку. Реалізація даного механізму сприяє більш ефективній організації роботи застосунку та забезпечує коректне розмежування функціональних можливостей між різними категоріями користувачів. Схему розмежування прав доступу наведено на рисунку 3.9.

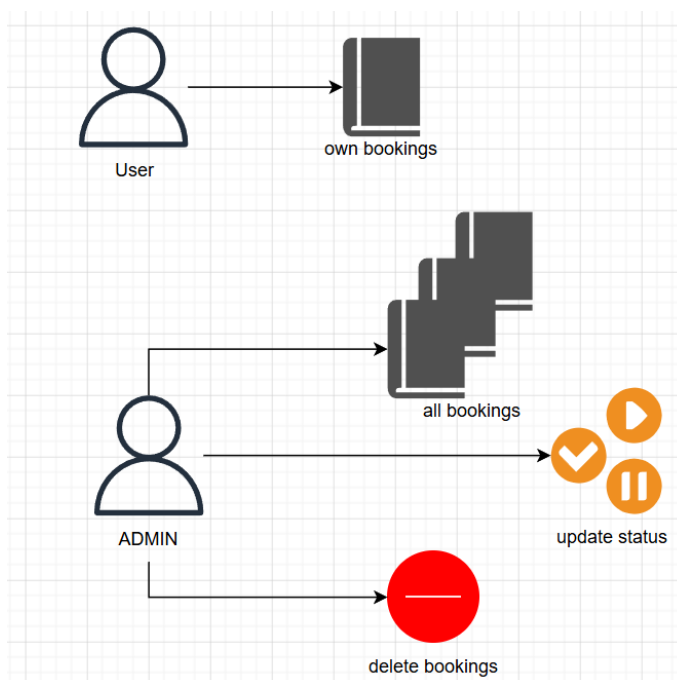


Рисунок 3.9 – Розмежування прав доступу користувачів у системі

Оновлення статусу ремонту транспортного засобу реалізовано за допомогою механізму оновлення документа у Firebase Firestore. Адміністратор має можливість змінювати статус ремонту. Фрагмент програмного коду оновлення статусу ремонту наведений у лістингу 3.3.

Лістинг 3.3 – Оновлення статусу ремонту у Firebase Firestore

```

val updates = hashMapOf<String, Any>(
    "status" to selectedStatus
)
  
```

```

FirebaseFirestore.getInstance()
    .collection("bookings")
    .document(booking.documentId)
    .update(updates)
    .addOnSuccessListener {
        Toast.makeText(
            this,
            "Статус оновлено",
            Toast.LENGTH_SHORT
        ).show()
    }
    .addOnFailureListener {
        Toast.makeText(
            this,
            "Помилка оновлення статусу",
            Toast.LENGTH_SHORT
        ).show()
    }

```

Кінець лістингу 3.3

Результат роботи механізму оновлення статусу ремонту наведено на рисунку 3.5.

Для забезпечення коректної роботи системи реалізовано механізм видалення записів із Firebase Firestore. Фрагмент програмного коду видалення запису із Firebase Firestore наведений у лістингу 3.4.

Лістинг 3.4 – Реалізація механізму видалення запису

```

AlertDialog.Builder(this)
    .setTitle("Видалення запису")
    .setMessage("Ви дійсно хочете видалити запис?")
    .setPositiveButton("Так") { _, _ ->

        FirebaseFirestore.getInstance()

```

```

        .collection("bookings")
        .document(booking.documentId)
        .delete()
        .addOnSuccessListener {
            Toast.makeText(
                this,
                "Запис видалено",
                Toast.LENGTH_SHORT
            ).show()
        }
        .addOnFailureListener {
            Toast.makeText(
                this,
                "Помилка видалення",
                Toast.LENGTH_SHORT
            ).show()
        }
    }
    .setNegativeButton("Скасувати", null)
    .show()

```

Кінець лістингу 3.4

Результат роботи механізму видалення записів та вікно підтвердження виконання даної операції наведено на рисунку 3.6.

Крім хмарної бази даних Firebase Firestore, у процесі розробки використовувалася локальна база даних Room, яка застосовувалася на ранніх етапах розробки для тестування структури даних та реалізації локального зберігання інформації [14]. Схема взаємодії Android-застосунку із Firebase Firestore наведена на рисунку 3.10.

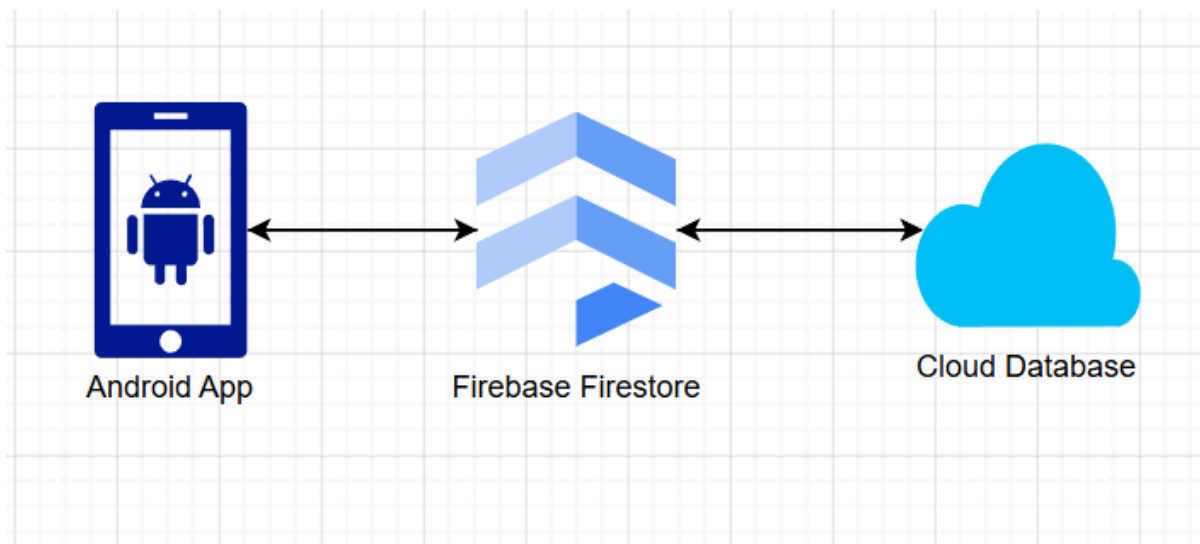


Рисунок 3.10 – Схема взаємодії Android-застосунку із Firebase Firestore

Використання Firebase Firestore дозволило забезпечити централізоване хмарне зберігання інформації, підтримку багатокористувацького режиму роботи та синхронізацію даних між пристроями у режимі реального часу. Це значно підвищило функціональність системи та забезпечило стабільну роботу Android-застосунку.

3.3 Захист інформаційно-комп'ютерної системи

Під час розробки Android-застосунку для управління записами про ремонт транспортних засобів особливу увагу приділено забезпеченню захисту даних користувачів та реалізації механізмів безпечної взаємодії із системою.

Одним із ключових елементів захисту системи є автентифікація користувачів. Для реалізації механізму реєстрації та авторизації у застосунку використано сервіс Firebase Authentication. Даний сервіс забезпечує безпечне зберігання облікових даних користувачів та підтримує механізми входу за допомогою електронної пошти та пароля. Фрагмент програмного коду авторизації користувача за допомогою Firebase Authentication наведений у лістингу 3.5.

Лістинг 3.5 – Авторизація користувача за допомогою Firebase Authentication

```
auth.signInWithEmailAndPassword(email, password)
    .addOnCompleteListener(this) { task ->

        if (task.isSuccessful) {

            Toast.makeText(
                this,
                "Вхід виконано успішно",
                Toast.LENGTH_SHORT
            ).show()

        } else {

            Toast.makeText(
                this,
                "Помилка авторизації",
                Toast.LENGTH_SHORT
            ).show()
        }
    }
}
```

Кінець лістингу 3.5

Після успішної перевірки облікових даних користувач отримує доступ до функціональних можливостей системи. Реалізований механізм авторизації забезпечує перевірку облікових даних користувача перед наданням доступу до функціональних можливостей системи. У разі успішного входу користувач автоматично отримує доступ до відповідних розділів Android-застосунку відповідно до своєї ролі. При введенні неправильних облікових даних система відображає інформаційне повідомлення про помилку та не допускає несанкціонованого доступу до інформації. Такий підхід дозволяє підвищити

рівень захисту даних та забезпечує безпечну роботу користувачів із системою. Інтерфейс авторизації користувача наведено на рисунку 3.11.

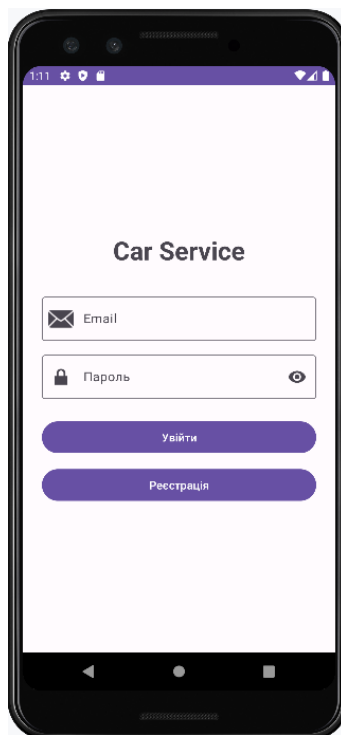


Рисунок 3.11 – Інтерфейс авторизації користувача

Firebase Authentication забезпечує перевірку коректності введених користувачем облікових даних та дозволяє уникнути несанкціонованого доступу до системи. Для авторизації у застосунку використовується вхід за допомогою електронної пошти та пароля. Після успішного входу користувач отримує доступ до функціональних можливостей системи відповідно до своєї ролі.

Усі паролі користувачів зберігаються на стороні Firebase у зашифрованому вигляді, що підвищує рівень безпеки програмного забезпечення та мінімізує ризик витоку конфіденційної інформації. Використання Firebase Authentication дозволяє уникнути необхідності самостійної реалізації серверної частини для обробки авторизації користувачів.

Для підтримки стабільної роботи Android-застосунку у системі реалізовано перевірку коректності введення даних користувачем. У випадку введення некоректної електронної пошти або неправильного пароля система відображає

відповідне повідомлення про помилку. Це дозволяє підвищити зручність використання застосунку та покращує взаємодію користувача із системою.

Додатково у застосунку реалізовано механізм збереження сесії користувача, що дозволяє автоматично виконувати вхід після повторного запуску Android-застосунку без необхідності повторного введення облікових даних.

Для забезпечення багатокористувацького режиму роботи у системі реалізовано механізм рольового контролю доступу. У застосунку передбачено два типи користувачів: звичайний користувач та адміністратор.

Розмежування прав доступу дозволяє забезпечити безпечну роботу із даними та виключає можливість виконання користувачем операцій, які не відповідають його ролі. Такий підхід сприяє підвищенню надійності системи та забезпечує контроль над виконанням основних функцій застосунку.

Для захисту даних у Firebase Firestore кожен запис містить ідентифікатор користувача (`userId`), що дозволяє фільтрувати записи та обмежувати доступ до інформації інших користувачів.

У процесі роботи Android-застосунку реалізовано механізм перевірки заповнення обов'язкових полів форми. Перед створенням запису система перевіряє наявність введених користувачем даних та не допускає збереження порожніх записів. Це дозволяє підтримувати цілісність інформації та зменшує ймовірність виникнення помилок під час роботи із системою.

Для підтвердження критичних дій у застосунку реалізовано діалогові вікна підтвердження. Наприклад, перед видаленням запису користувач повинен підтвердити виконання даної операції. Це дозволяє уникнути випадкового видалення інформації.

Для забезпечення стабільності роботи Android-застосунку реалізовано механізми обробки помилок та повідомлення користувача про результати виконання операцій. У застосунку використовуються повідомлення `Snackbar` та `Toast`, які інформують користувача про успішне виконання операцій або виникнення помилок під час взаємодії із системою [15]. Фрагмент програмного

коду повідомлення користувача про результат виконання операції наведений у лістингу 3.6.

Лістинг 3.6 – Реалізація повідомлень користувача у системі

```
Toast.makeText(  
    this,  
    "Запис успішно створено",  
    Toast.LENGTH_SHORT  
)  
.show()  
Snackbar.make(  
    binding.root,  
    "Помилка збереження даних",  
    Snackbar.LENGTH_LONG  
)  
.show()
```

Кінець лістингу 3.6

Інформаційні повідомлення використовуються під час виконання основних операцій системи, зокрема авторизації користувача та створення записів.

Таким чином, у розробленому Android-застосунку реалізовано основні механізми захисту інформації, автентифікації користувачів, контролю доступу та безпечної взаємодії із хмарною базою даних Firebase Firestore. Це дозволило забезпечити стабільну та безпечну роботу програмного забезпечення.

3.4 Тестування та налагодження інформаційно-комп'ютерної системи

Після завершення розробки Android-застосунку для управління записами про ремонт транспортних засобів проведено тестування програмного забезпечення з метою перевірки коректності роботи системи, стабільності функціонування та відповідності поставленим вимогам.

Основною метою тестування виявлення можливих помилок у роботі застосунку, перевірка функціональних можливостей системи та оцінка коректності взаємодії Android-застосунку із хмарними сервісами Firebase Authentication та Firebase Firestore.

У процесі тестування перевірено:

- реєстрацію користувачів;
- авторизацію користувачів;
- створення записів на ремонт;
- редагування записів;
- видалення записів;
- зміну статусу ремонту;
- роботу пошуку записів;
- синхронізацію даних із Firebase Firestore;
- коректність роботи рольового контролю доступу;
- відображення інформаційних повідомлень Snackbar та Toast;
- коректність переходів між основними екранами застосунку;
- збереження сесії користувача.

Для перевірки функціональності системи проведено функціональне тестування Android-застосунку. Результати тестування наведені у таблиці 3.1.

Таблиця 3.1 – Результати функціонального тестування системи

№	Функція	Результат
1	Реєстрація користувача	Працює коректно
2	Авторизація користувача	Працює коректно
3	Створення запису	Працює коректно
4	Редагування запису	Працює коректно
5	Видалення запису	Працює коректно
6	Зміна статусу ремонту	Працює коректно
7	Пошук записів	Працює коректно
8	Синхронізація із Firestore	Працює коректно
9	Role-based access	Працює коректно
10	Збереження сесії	Працює коректно

Під час тестування системи особлива увага приділялася перевірці механізму багатокористувацького доступу. Встановлено, що система коректно розмежовує права доступу між користувачем та адміністратором відповідно до ролей користувачів. Результат тестування механізму рольового контролю доступу наведений на рисунку 3.12.

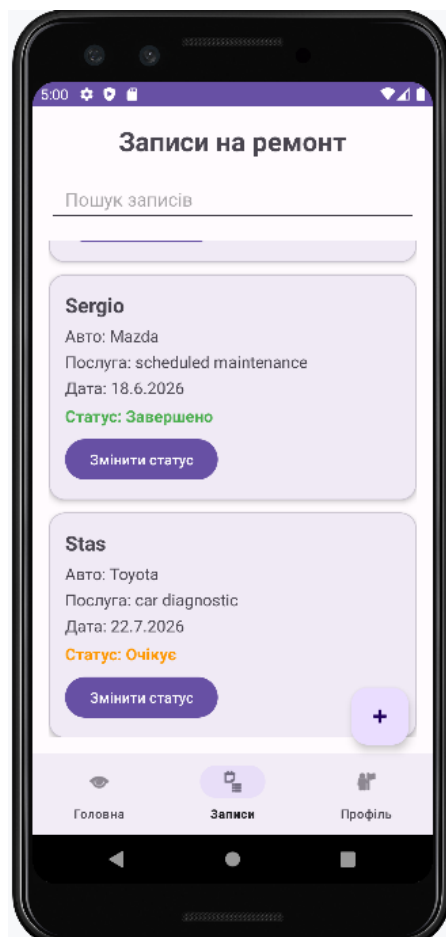


Рисунок 3.12 – Тестування механізму рольового контролю доступу

У процесі тестування також перевірено роботу системи пошуку записів. Реалізований механізм SearchView дозволяє швидко знаходити записи за ключовими словами та коректно фільтрує інформацію у списку записів.

Під час тестування підтверджено коректність відображення результатів пошуку та своєчасне оновлення списку записів відповідно до введеного запиту. Результат тестування механізму пошуку наведений на рисунку 3.13.

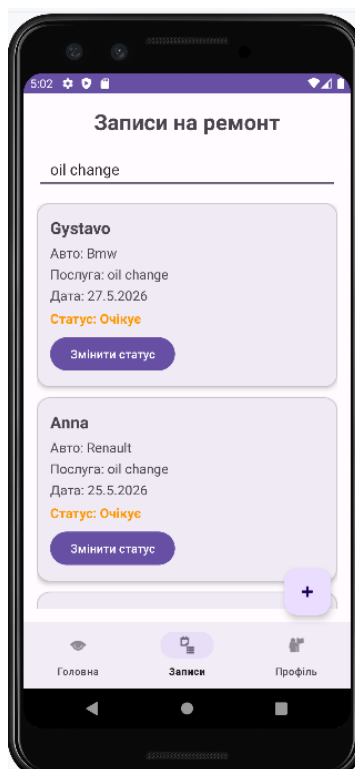


Рисунок 3.13 – Тестування механізму пошуку записів

Для перевірки стабільності роботи Android-застосунку проведено тестування взаємодії із Firebase Firestore. У процесі тестування підтверджено коректну синхронізацію записів між користувачами та стабільне збереження інформації у хмарному середовищі.

Також перевірено механізм автоматичного відновлення сесії користувача після повторного запуску Android-застосунку. Результати тестування підтвердили коректність роботи Firebase Authentication та автоматичної авторизації користувача. Додатково перевірено коректність локального збереження допоміжної інформації про роль користувача за допомогою SessionManager.

У процесі тестування перевірено роботу механізму заповнення обов'язкових полів форми. Встановлено, що система коректно контролює наявність необхідних даних та інформує користувача у випадку спроби створення запису з незаповненими полями.

У результаті проведеного тестування встановлено, що Android-застосунок коректно виконує поставлені функції, забезпечує стабільну роботу із Firebase Firestore та підтримує багатокористувацький режим роботи.

Під час тестування Android-застосунку особлива увага приділялась перевірці коректності роботи механізмів авторизації та реєстрації користувачів. Перевірено можливість створення нового облікового запису, виконання входу до системи за допомогою електронної пошти та пароля, а також коректність обробки помилок при введенні неправильних даних користувачем.

Окремо проводилось тестування механізму створення записів на ремонт транспортних засобів. Перевірено коректність збереження інформації у Firebase Firestore, синхронізацію даних між пристроями та відображення створених записів у списку користувача.

Для перевірки багатокористувацького режиму роботи виконувалось тестування розмежування прав доступу між звичайним користувачем та адміністратором системи. У ході тестування встановлено, що користувач має доступ лише до власних записів, тоді як адміністратор отримує розширені можливості керування інформацією у системі.

Також проводилось тестування механізму видалення записів та перевірка роботи підтвердження критичних дій користувача. Встановлено, що система коректно обробляє запити на видалення записів та відображає інформаційні повідомлення про результати виконання операцій.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи бакалавра розроблено Android-застосунок для управління записами про ремонт транспортних засобів із використанням сучасних мобільних та хмарних технологій.

У процесі виконання роботи проаналізовано сучасні підходи до автоматизації процесу запису транспортних засобів на ремонт та визначено основні недоліки традиційних способів організації роботи станцій технічного обслуговування. На основі проведеного аналізу сформовано функціональні та нефункціональні вимоги до програмного забезпечення.

У ході роботи обґрунтовано вибір технологій та інструментальних засобів розробки. Для реалізації програмного продукту використано мову програмування Kotlin, середовище розробки Android Studio, а також хмарні сервіси Firebase Authentication та Firebase Firestore. Виконано проєктування структури Android-застосунку, архітектури системи та механізмів зберігання даних.

У розробленому застосунку реалізовано систему автентифікації та авторизації користувачів, що забезпечує безпечний доступ до функціональних можливостей системи. Також реалізовано механізми створення, редагування, перегляду та видалення записів на ремонт транспортних засобів із використанням хмарної бази даних Firebase Firestore.

Для забезпечення безпеки та багатокористувацького режиму роботи реалізовано рольовий контроль доступу між звичайними користувачами та адміністраторами. Завдяки цьому користувачі можуть працювати лише з власними записами, тоді як адміністратор має розширені можливості керування даними системи.

У процесі тестування та налагодження програмного забезпечення перевірено працездатність основних функцій застосунку, коректність роботи механізмів авторизації, пошуку, рольового контролю доступу та взаємодії з Firebase Firestore. Результати тестування підтвердили стабільність роботи

системи та відповідність реалізованого програмного забезпечення поставленим вимогам.

Таким чином, усі поставлені завдання виконано в повному обсязі, а мету кваліфікаційної роботи бакалавра досягнуто. Розроблений Android-застосунок може бути використаний для автоматизації процесу управління записами про ремонт транспортних засобів у невеликих станціях технічного обслуговування та може слугувати основою для подальшого розширення функціональних можливостей системи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Android Mobile App Developer Tools – Android Developers. URL: <https://developer.android.com> (дата звернення: 15.02.2026).
2. Design & Plan. Android Developers. URL: <https://developer.android.com/design> (дата звернення: 15.02.2026).
3. Firebase Authentication. URL: <https://firebase.google.com/docs/auth> (дата звернення: 16.02.2026).
4. Firestore. Firebase. URL: <https://firebase.google.com/docs/firestore> (дата звернення: 17.02.2026).
5. Kotlin Programming Language. URL: <https://kotlinlang.org/> (дата звернення: 17.02.2026).
6. Download Android Studio & App Tools - Android Developers. Android Developers. URL: <https://developer.android.com/studio> (дата звернення: 08.03.2026).
7. Material Design. Material Design. URL: <https://m3.material.io/> (дата звернення: 08.03.2026).
8. Git. URL: <https://git-scm.com/> (дата звернення: 08.03.2026).
9. GitHub Change is constant. GitHub keeps you ahead. URL: <https://github.com/> (дата звернення: 10.03.2026).
10. Navigation. App architecture. Android Developers. URL: <https://developer.android.com/guide/navigation> (дата звернення: 09.04.2026).
11. Create dynamic lists with RecyclerView. Views. Android Developers. URL: <https://developer.android.com/develop/ui/views/layout/recyclerview> (дата звернення: 10.04.2026).
12. SearchView. API reference. Android Developers. URL: <https://developer.android.com/reference/android/widget/SearchView> (дата звернення: 10.04.2026).

13. DatePickerDialog. API reference. Android Developers. URL: <https://developer.android.com/reference/android/app/DatePickerDialog> (дата звернення: 12.04.2026).

14. Save data in a local database using Room. App data and files. Android Developers. URL: <https://developer.android.com/training/data-storage/room> (дата звернення: 02.05.2026).

15. Snackbar. API reference. Android Developers. URL: <https://developer.android.com/reference/com/google/android/material/snackbar/Snackbar> (дата звернення: 02.05.2026).