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-ДОДАТКУ ДЛЯ ЕЛЕКТРОННИХ ПЕРЕПУСТОК В ГУРТОЖИТОК ЛНТУ З ВИКОРИСТАННЯМ KOTLIN JETPACK COMPOSE ТА FIRESTORE DATABASE

DEVELOPMENT OF AN ANDROID APPLICATION FOR ELECTRONIC DORMITORY PASSES AT LNTU USING KOTLIN JETPACK COMPOSE AND FIRESTORE DATABASE

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

Виконав: здобувач вищої освіти
групи ІПЗ-42
Рудін Олег Сергійович

Керівник:
Корень Віталій Володимирович

Кваліфікаційну роботу
допущено до захисту
«10» 06 2025 р.
Гарант освітньої програми:
к.т.н., доцент
Ліщина Наталія Миколаївна

_____

Луцьк – 2025 року

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» 12 2024 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

Рудіну Олегу Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи *«Розробка Android-додатку для електронних перепусток в гуртожиток ЛНТУ з використанням Kotlin Jetpack Compose та Firestore Database»*
Керівник роботи: *Корень В. В.*
затверджені наказом закладу вищої освіти від *«19» грудня 2024 р.* № *474/01-02*
2. Строк подання здобувачем вищої освіти кваліфікаційної роботи бакалавра *«10» червня 2025 р.*
3. Вихідні дані до роботи: *Kotlin, Jetpack Compose, Firestore, методичні вказівки до виконання кваліфікаційної роботи бакалавра.*

4. Зміст розрахунково-пояснювальної записки (перелік питань, що потрібно розробити): *аналіз предметної області, постановка завдань, формування вимог до ПЗ, проектування архітектури додатку, вибір технологій, розробка Android-додатку, створення бази даних, реалізація авторизації, тестування системи, підготовка технічної документації.*

5. Перелік графічного матеріалу: *11 рисунків, 3 таблиці.*

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
Аналіз предметної області	*Корень В. В.*		
Специфікація вимог до розробленої системи	*Корень В. В.*		
Розробка об'єкта проектування	*Корень В. В.*		
Нормоконтроль	*Вознюк А. В.*		
Гарант ОП	*Ліщина Н. М.*		
Показник запозичень тексту	1,74 %		
Академічна доброчесність	*Корень В. В.*		

7. Дата видачі завдання *«20» грудня 2025 р.*

№	Назва етапів кваліфікаційної роботи бакалавра	Строк виконання етапів роботи	Примітка
1	Огляд літературних джерел по темі кваліфікаційної роботи бакалавра	до 04.02.2025 р.	Виконано
2	Аналіз проблеми розробки та впровадження об'єкту проектування	до 01.03.2025 р.	Виконано
3	Обгрунтування вибору шляхів, технологій і засобів вирішення поставленого завдання	до 15.03.2025 р.	Виконано
4	Розробка функціонально-структурної схеми роботи об'єкта проектування та проектування бази даних	до 29.03.2025 р.	Виконано
5	Практична реалізація об'єкта проектування та розробка бази даних	до 26.04.2025 р.	Виконано
6	Тестування та налагодження об'єкта проектування	до 03.05.2025 р.	Виконано
7	Здача чистового варіанту кваліфікаційної роботи бакалавра на кафедру	до 10.06.2025 р.	Виконано

Здобувач вищої освіти Рудін Олег РУДІН

Керівник кваліфікаційної роботи Віталій КОРЕНЬ

АНОТАЦІЯ

Рудін О. С. Розробка Android-додатку для електронних перепусток в гуртожиток ЛНТУ з використанням Kotlin Jetpack Compose та Firestore Database. Рукопис. Кваліфікаційна робота бакалавра ОП «Інженерія програмного забезпечення» спеціальності «Інженерія програмного забезпечення». Луцький національний технічний університет. Луцьк, 2025.

Кваліфікаційна робота бакалавра складається зі вступу, трьох розділів, висновків, списку використаних джерел та додатків. У першому розділі здійснено аналіз предметної області, обґрунтовано вибір інструментів та сформульовано завдання. У другому розділі подано проєктування, визначення вимог, UML-діаграми, вибір архітектури та реалізацію структури додатку. У третьому розділі описано процес реалізації програмного забезпечення, проведено тестування, налагодження, організацію захисту інформації та створення документації. У висновках сформульовано результати виконання роботи, перспективи розвитку і можливості практичного застосування.

Ключові слова: QR-код, Android, Firebase, шифрування, мобільний додаток, Jetpack Compose.

ABSTRACT

Rudin O. Development of an Android Application for Electronic Dormitory Passes at LNTU Using Kotlin Jetpack Compose and Firestore Database. Manuscript. Bachelor qualification work of the educational program "Software Engineering" in the specialty "Software Engineering". Lutsk National Technical University. Lutsk, 2025.

The bachelor's qualification work consists of an introduction, three chapters, conclusions, a list of references, and appendices. The first chapter provides an analysis of the subject area, justification for tool selection, and task formulation. The second chapter includes software requirement specification, system design, UML diagrams, architecture selection, and application structure implementation. The third chapter presents the practical implementation of the software, debugging and testing procedures, information security mechanisms, and documentation development. The conclusions summarize the results of the work, development prospects, and potential areas of practical application.

Keywords: QR code, Android, Firebase, encryption, mobile application, Jetpack Compose.

ЗМІСТ

ВСТУП

Актуальність теми. У сучасному світі цифрових технологій усе більше процесів, що раніше виконувалися вручну, автоматизуються за допомогою інформаційних систем. Однією з важливих сфер застосування таких рішень є організація та контроль доступу до об’єктів з обмеженим або регламентованим входом – зокрема, у студентські гуртожитки. Забезпечення безпечного, прозорого та зручного доступу для мешканців гуртожитків є актуальною задачею для багатьох закладів вищої освіти, зокрема й Луцького національного технічного університету.

На практиці система пропускного режиму в гуртожитках часто реалізується у вигляді паперових перепусток або ручних реєстрів, що є застарілими та неефективними методами контролю. Вони не забезпечують належного рівня безпеки, не дають можливості оперативно перевіряти актуальність перепустки та не мають інтеграції з електронними базами даних. Такі системи також не враховують можливостей мобільного доступу, що є звичним для студентів як активних користувачів смартфонів.

Світовий досвід демонструє широке впровадження мобільних додатків і хмарних баз даних у сферу контролю доступу, зокрема на базі Android-платформи з використанням таких технологій, як Firebase/Firestore, QR-коди, NFC тощо. Ці рішення забезпечують зручність, швидкість та безпеку обміну інформацією між користувачем і системою контролю доступу.

Актуальність цієї кваліфікаційної роботи зумовлена потребою підвищення ефективності та автоматизації процесу видачі та перевірки перепусток у гуртожитках ЛНТУ. Запропонований підхід дозволяє модернізувати систему доступу без використання паперових носіїв, а також забезпечити збереження даних у хмарному середовищі, що підвищує її масштабованість і надійність.

Метою цієї роботи є розробка Android-додатку для електронних перепусток із використанням технології Kotlin Jetpack Compose для побудови

інтерфейсу та Firestore Database як хмарної бази даних. Додаток повинен реалізовувати функціональність генерації та перевірки електронної перепустки, авторизації користувачів, зберігання даних у базі та обмеження доступу до програми відповідно до ролей (мешканець, охоронець, адміністратор).

Об’єктом кваліфікаційної роботи є процес організації доступу до студентського гуртожитку.

Предметом роботи є програмний продукт у вигляді мобільного додатку, що автоматизує видачу й перевірку електронних перепусток.

Для досягнення поставленої мети були поставлені наступні завдання:

1) проаналізувати сучасний стан вирішення задачі автоматизації контролю доступу до студентських гуртожитків, дослідити наявні аналоги, їхні функціональні можливості, переваги та недоліки;

2) виконати моделювання системи з використанням UML-нотацій, побудувати діаграми для відображення логіки роботи додатку;

3) обґрунтувати вибір засобів розробки та архітектури програмного забезпечення, а саме: мови Kotlin, технології Jetpack Compose для побудови інтерфейсу та Firestore Database для зберігання даних;

4) розробити прототип Android-додатку, що реалізує створення, візуалізацію та перевірку електронних перепусток, а також модулі авторизації та доступу;

5) розробити схему зберігання даних у Firestore Database, описати структуру колекцій та документів, реалізувати логіку взаємодії клієнтського додатку з хмарною базою;

6) реалізувати базові механізми захисту додатку, включаючи аутентифікацію користувачів, розмежування прав доступу та базову перевірку даних;

7) провести тестування та налагодження програмного продукту, перевірити коректність функціонування основних модулів та інтерфейсу на реальному пристрої Android;

8) підготувати технічну документацію до розробленого додатку, включаючи інструкції користувача, опис інтерфейсів, вимоги до встановлення та оновлення.

Запропоноване рішення має міждисциплінарний характер і взаємозв’язане з іншими розробками в галузі мобільного програмування, хмарних баз даних, безпеки інформації та проєктування UI/UX. Реалізація такого додатку може бути використана не лише в ЛНТУ, а й адаптована для інших навчальних закладів, гуртожитків, готелів чи офісів, де потрібен контрольований доступ.

Отже, робота спрямована на створення сучасного, зручного та безпечного цифрового рішення, що дозволяє оптимізувати процес ідентифікації мешканців та перевірки перепусток з урахуванням сучасних технологій розробки Android-додатків.

РОЗДІЛ 1
АНАЛІЗ ІНФОРМАЦІЙНИХ СИСТЕМ КОНТРОЛЮ ДОСТУПУ В ГУРТОЖИТКАХ ТА ПОСТАНОВКА ЗАВДАНЬ НА КВАЛІФІКАЦІЙНУ РОБОТУ

1.1 Аналіз сучасного стану проблеми

У сучасних умовах цифрової трансформації багато процесів, пов’язаних із безпекою та контролем доступу, поступово автоматизуються. Проте в більшості гуртожитків українських закладів вищої освіти досі переважають традиційні, паперові або усно-контрольовані системи перепусток. Вони характеризуються високою трудомісткістю, відсутністю централізованого обліку та низьким рівнем захисту від підробок або втрат. Це створює потребу у впровадженні сучасних інформаційних рішень, що поєднують зручність використання та надійність зберігання даних.

Проблематика автоматизації доступу та розселення студентів у гуртожитках розглядається у низці сучасних досліджень, які демонструють зростаючу увагу до цифрових рішень у сфері управління студентським середовищем.

У роботі Полякова М. О. [1] акцентовано на важливості модернізації традиційної системи пропускного режиму в студентських гуртожитках як засобу забезпечення безпеки, що включає як програмно-апаратні, так і організаційні складові. Автор пропонує систему на основі унікального ідентифікатора, який дозволяє відстежувати переміщення студентів та автоматизувати фіксацію входів і виходів. Загалом, такий підхід є близьким до концепції мобільних електронних перепусток, реалізованих у рамках цього проєкту.

Щебетін Б. Ю. у своєму дослідженні [2] висвітлює процес автоматизації основних функцій деканату, включаючи поселення, виселення та сплату за контракт. Важливим є підхід до створення універсального інформаційного середовища з елементами workflow-логіки. Скороход А. А. [3] розробив

багатосторінкову веб-систему автоматизації розселення студентів із функціоналом звітності, перевірки статусу проживання та особистих кабінетів. Реалізація здійснена на основі технологій, що відрізняється від мобільного підходу, використаного в цьому проєкті. Водночас концептуальні ідеї взаємодії з базою даних і персоналізованого обліку мешканців є дотичними.

Зарубіжні дослідження зосереджуються на розширеному функціоналі та інтеграції із фізичними системами управління. Bhandari P. [4] у своєму кейс-дослідженні проаналізував труднощі студентів у гуртожитках під час вселення, проживання та комунікації. На основі зібраних даних було розроблено персоналізований мобільний застосунок, що дозволив покращити зворотний зв’язок і самостійність студентів. Цей підхід близький до концепції, реалізованої у кваліфікаційній роботі, з акцентом на UX-дизайн і зручність користувача. Окремо варто відзначити дослідження Wen K. і Fang Y. [5], в якому розроблено систему щоденного обліку відвідування гуртожитку з використанням мобільної веб-платформи, біометрії та алгоритмів моделювання завантаження системи.

Усі розглянуті джерела свідчать про високу актуальність проблеми автоматизації систем доступу та управління гуртожитками. Водночас реалізований у цій роботі підхід – мобільний додаток з генерацією QR-перепусток на базі Kotlin і Firestore – є інноваційним у межах українських освітніх практик і спирається на найкращі рішення, описані в наукових і прикладних дослідженнях.

Серед сучасних аналогів мобільних рішень для контролю доступу особливу увагу привертають системи, реалізовані на базі зв’язки Firebase Authentication та Cloud Firestore. Такі рішення активно впроваджуються у країнах з високим рівнем цифровізації, зокрема у США, Німеччині, Нідерландах, Канаді. Вони забезпечують централізований доступ до хмарної бази даних користувачів, мають високий рівень масштабованості, дозволяють легко адаптувати функціонал під потреби різних установ – від офісів і житлових комплексів до університетських кампусів і гуртожитків. До переваг

подібних систем належать: безперервний доступ до даних з будь-якого пристрою, підтримка багаторівневої автентифікації, зручне адміністрування, підтримка інтеграції з іншими сервісами Google, можливість впровадження елементів аналітики та віддаленого моніторингу.

Однак, незважаючи на технічну досконаліть, більшість таких систем створюються з урахуванням особливостей інфраструктури західних закладів освіти, де присутня розвинена система пропускного контролю, стандартизовані протоколи взаємодії з охороною та централізоване IT-адміністрування. В українських реаліях ці рішення часто виявляються або занадто дорогими, або складними для розгортання через відсутність уніфікованих IT-середовищ у державних навчальних закладах, недостатній рівень автоматизації процесів поселення і контролю, а також низьку цифрову адаптацію персоналу.

На території України подібні проєкти впроваджуються переважно в межах пілотних ініціатив – наприклад, у приватних освітніх закладах, інноваційних кампусах або технопарках. У деяких випадках мобільні перепустки реалізуються як частина загальної системи «розумного кампусу», але такі приклади залишаються поодинокими. У державних університетах систематичне впровадження мобільних додатків для доступу в гуртожитки практично відсутнє. Це свідчить про значну ринкову нішу, в межах якої існує реальна потреба у створенні локалізованого, адаптивного та бюджетного мобільного рішення, здатного працювати в умовах українських закладів вищої освіти та підтримувати базовий рівень інформаційної безпеки. Відповідно, запропонований у межах кваліфікаційної роботи проєкт має не лише освітнє, а й практичне суспільне значення.

Патентний пошук за ключовими словами «електронна перепустка», «мобільний додаток контролю доступу», «гуртожиток», «QR-код пропуск» виявив низку авторських рішень, орієнтованих на загальні системи ідентифікації або офісний простір. Це підтверджує відсутність унікального програмного рішення, адаптованого до специфіки студентських гуртожитків

України, що в свою чергу обґрунтовує доцільність розробки нової інформаційної системи.

Ураховуючи наведений аналіз, можна дійти висновку, що розробка мобільного додатку для електронної перепустки в гуртожиток ЛНТУ є не лише технічно обґрунтованою, а й соціально значущою ініціативою. Запропоноване рішення поєднує передові технології мобільної розробки, хмарні сервіси та принципи інформаційної безпеки, водночас залишаючись доступним для впровадження в умовах українських університетів. Це визначає предмет, мету та завдання дослідження, а також актуалізує потребу у створенні прототипу, який може стати основою для масштабування подібних систем у закладах вищої освіти України.

Загалом, актуальність дослідження обумовлена потребою у створенні сучасного мобільного додатку, який дозволяє автоматизувати облік перепусток, підвищити рівень безпеки гуртожитку ЛНТУ, зменшити навантаження на охоронний персонал та забезпечити зручність для користувачів.

1.2 Постановка завдання на кваліфікаційну роботу бакалавра

Метою кваліфікаційної роботи є розробка Android-додатку для електронних перепусток у гуртожиток ЛНТУ з використанням сучасних мобільних та хмарних технологій – Kotlin Jetpack Compose та Firestore Database. Розроблений застосунок має забезпечити зручну і безпечну ідентифікацію студентів за допомогою QR-кодів, інтеграцію з Firebase-сервісами та можливість гнучкого адміністрування доступу до гуртожитку.

Для досягнення цієї мети було поставлено такі завдання:

1) проаналізувати сучасний стан вирішення задачі автоматизації контролю доступу до студентських гуртожитків, дослідити наявні аналоги, їхні функціональні можливості, переваги та недоліки;

2) виконати моделювання системи з використанням UML-нотацій, побудувати діаграми для відображення логіки роботи додатку;

3) обґрунтувати вибір засобів розробки та архітектури програмного забезпечення, а саме: мови Kotlin, технології Jetpack Compose для побудови інтерфейсу та Firestore Database для зберігання даних;

4) розробити прототип Android-додатку, що реалізує створення, візуалізацію та перевірку електронних перепусток, а також модулі авторизації та доступу;

5) розробити схему зберігання даних у Firestore Database, описати структуру колекцій та документів, реалізувати логіку взаємодії клієнтського додатку з хмарною базою;

6) реалізувати базові механізми захисту додатку, включаючи аутентифікацію користувачів, розмежування прав доступу та базову перевірку даних;

7) провести тестування та налагодження програмного продукту, перевірити коректність функціонування основних модулів та інтерфейсу на реальному пристрої Android;

8) підготувати технічну документацію до розробленого додатку, включаючи інструкції користувача, опис інтерфейсів, вимоги до встановлення та оновлення.

Загалом, виконання зазначених завдань дозволяє комплексно вирішити проблему цифровізації процесу доступу до гуртожитку, підвищити ефективність контролю та безпеки, а також продемонструвати практичне застосування сучасних технологій у мобільній розробці та хмарних обчисленнях.

Висновки до розділу 1

У першому розділі було проведено ґрунтовний аналіз предметної області, що охоплює сучасні підходи до організації контролю доступу в гуртожитках закладів вищої освіти. Було встановлено, що більшість українських студентських гуртожитків досі використовують застарілі методи пропускного

режиму, зокрема паперові перепустки, ручні журнали реєстрації або усну ідентифікацію. Ці підходи мають суттєві недоліки: низький рівень безпеки, відсутність централізованого обліку, складність в управлінні та неможливість інтеграції з іншими інформаційними системами.

Світові тенденції демонструють активне впровадження мобільних додатків і хмарних технологій для автоматизації подібних процесів у навчальних закладах, бізнес-центрах, готелях тощо. Зокрема, успішно використовуються Android-додатки з підтримкою електронної аутентифікації через QR-коди, NFC або онлайн-бази даних на базі Firebase/Firestore. Аналіз аналогів показав наявність технологічних рішень для контролю доступу, однак жодне з них не є адаптованим до потреб гуртожитків ЛНТУ або не враховує специфіку українських умов, студентської мобільності, обмежень ресурсів та вимог до безпеки.

У результаті патентного пошуку виявлено, що більшість запатентованих рішень орієнтовані на корпоративне або комерційне застосування і не враховують потреб студентського середовища. Відповідно, виникає потреба у створенні нового мобільного рішення, яке дозволить спростити процес отримання та перевірки перепусток, підвищити безпеку проживання, автоматизувати реєстрацію користувачів та забезпечити зручну взаємодію між студентами, адміністрацією та охороною гуртожитку.

На основі проведеного аналізу сформульовано основні завдання кваліфікаційної роботи, які включають розробку архітектури системи, вибір технологій, реалізацію додатку, побудову бази даних, організацію захисту інформації, проведення тестування та підготовку супровідної документації. Ці завдання стали основою для наступних розділів роботи та безпосередньої реалізації Android-додатку з електронними перепустками.

РОЗДІЛ 2
СПЕЦИФІКАЦІЯ ВИМОГ ДО РОЗРОБЛЮВАНОЇ СИСТЕМИ

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

Для ефективної реалізації мобільного додатку електронних перепусток LNTU Pass було обрано орієнтовану на користувача модель розробки, що передбачає ітеративне уточнення вимог, побудову прототипів інтерфейсу та подальше впровадження функціональних компонентів. Моделювання системи виконувалося з використанням UML-нотації, яка дозволяє наочно подати структуру, поведінку та взаємодію елементів додатку.

Для визначення функціональних та нефункціональних вимог були застосовані методи інтерв'ю з потенційними користувачами (студентами, охоронцями, адміністрацією), а також аналіз аналогів та спостереження за процесами проходу до гуртожитку. Це дозволило сформулювати чіткі цілі системи.

У таблиці 2.1 наведено відповідність між основними подіями у додатку та відповідною реакцією системи на них.

Таблиця 2.1 – Відповідність «Подія – Відгук»

Подія (дія користувача/системи)	Відгук системи (реакція, відповідь)
Користувач вводить логін і пароль	Перевірка даних у Firebase, перехід до профілю або виведення повідомлення про помилку
Користувач натискає кнопку «Відновити пароль»	Відправка email з інструкцією скидання пароля через Firebase
Користувач входить у профіль	Відображення персональних даних (ПІ, факультет, кімната, фото)
Натискання кнопки «Згенерувати QR»	Генерація зашифрованого QR-коду (AES), відображення на екрані з таймером (10 секунд)
Натискання кнопки «Вийти» у профілі	Вихід з акаунта, повернення на екран логіну
Контролер натискає «Відсканувати QR»	Відкриття інтерфейсу сканера з дозволом на використання камери
Сканування QR-коду	Розшифровка коду, відображення профілю студента з даними
QR-код недійсний або завершився час дії	Повідомлення про помилку, повернення до режиму сканування

Основні цілі системи:

1) реалізація зручної системи входу в додаток з автентифікацією через Firebase;

2) створення профілю користувача з виведенням персональних даних;

3) генерація QR-коду, що містить зашифровану інформацію;

4) створення окремого додатку для контролера з функцією сканування QR-кодів;

5) безпечне зберігання даних у хмарній базі даних Firestore;

6) обмеження доступу до даних відповідно до ролі користувача.

Функціональні вимоги:

1) реєстрація/вхід користувача з Firebase Authentication – забезпечення авторизації користувачів за допомогою email і пароля через сервіс Firebase, з підтримкою перевірки автентичності та збереження облікових записів у хмарі;

2) перегляд власного профілю з ПІ, факультетом і кімнатою – користувач має змогу переглядати персональну інформацію, яка зберігається у Firestore, включаючи зображення профілю, повне ім’я, факультет та номер кімнати у гуртожитку;

3) генерація QR-коду на 10 секунд з використанням AES-шифрування – створення динамічного QR-коду, що містить зашифровані персональні дані користувача, який автоматично видаляється після короткочасного показу (10 секунд) для підвищення безпеки;

4) окремий додаток для контролера для сканування QR-коду – другий застосунок дозволяє співробітнику гуртожитку сканувати QR-код і розшифровувати дані з використанням того ж ключа шифрування;

5) виведення даних після успішного сканування – після розпізнавання QR-коду контролер бачить ім’я студента, факультет і номер кімнати, що дозволяє швидко перевірити особу;

6) захист від стороннього доступу та видалення коду після сесії – система автоматично очищає QR-код після завершення сеансу перегляду або при спробі доступу сторонніх користувачів;

7) вивід повідомлень у випадку помилки або відсутності даних – реалізовано обробку винятків з повідомленням користувача про проблеми (наприклад, відсутність інтернету, помилки аутентифікації, відсутність профілю в базі).

Нефункціональні вимоги:

1) платформа Android 8.0+ – додаток підтримується на пристроях з операційною системою Android версії 8.0 (API 26) і новіших;

2) мова програмування Kotlin – уся логіка додатку реалізована мовою Kotlin, що забезпечує надійність, безпечну типізацію та сучасний підхід до розробки;

3) архітектура MVVM – використовується архітектурний шаблон Model–View–ViewModel для чіткого розділення логіки, даних і користувацького інтерфейсу, що спрощує супровід і тестування;

4) база даних Firestore Database (cloud-based) – зберігання та синхронізація профілів користувачів відбувається у хмарній NoSQL-базі даних Firestore від Google з підтримкою реального часу.

5) безпека, AES-шифрування даних у QR, Firebase Authentication – забезпечено захист переданих даних за рахунок шифрування за алгоритмом AES та безпечної автентифікації користувачів через Firebase;

6) простота інтерфейсу, інтуїтивна навігація, мінімалістичний дизайн – інтерфейс користувача розроблено відповідно до принципів Material Design з урахуванням зручності, простоти та доступності для студентів і персоналу.

У процесі проектування були створені відповідні UML-діаграми. Діаграма варіантів використання (рис. 2.1) ілюструє основні дії користувачів системи: студента та контролера. Студент має змогу здійснювати вхід у застосунок, переглядати власний профіль, генерувати QR-код, виходити з облікового запису та оновлювати свої дані. Контролер, у свою чергу, може сканувати QR-коди та переглядати зашифровані дані студентів. Ця діаграма чітко формалізує функціональність системи з урахуванням ролей.

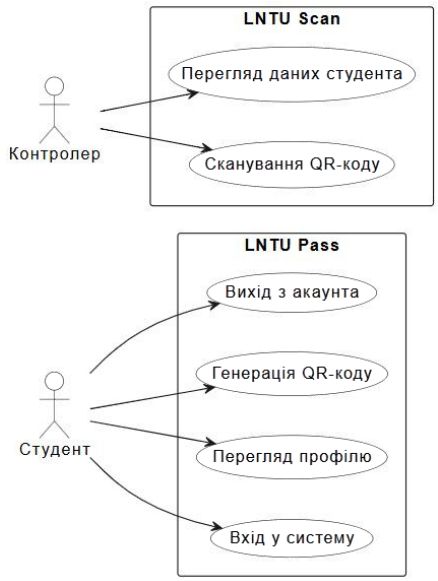

Рисунок 2.1 – Діаграма варіантів використання

Діаграма класів (рис. 2.2) показує логічну структуру основних сутностей системи. Клас User зберігає дані про студента: ідентифікатор, ім’я, прізвище, факультет, кімнату та посилання на фото. Він має зв’язок з класом QRCode, який містить зашифровані дані, часову мітку створення і метод перевірки валідності. Клас Session відповідає за авторизаційні сесії користувача, зберігаючи ID користувача, токен і час створення. Така структура сприяє підтримці безпеки та інтеграції з Firebase Firestore.

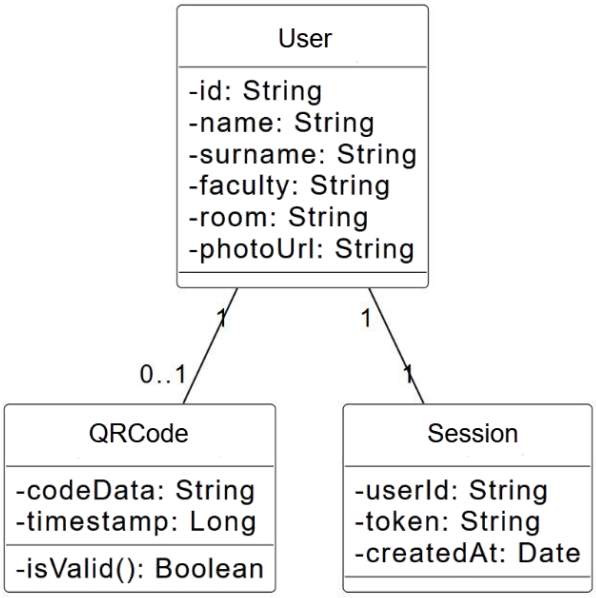

Рисунок 2.2 – Діаграма класів

Діаграма активностей (рис. 2.3) моделює послідовність дій студента: від запуску додатку та перевірки автентифікації до генерації та відображення QR-коду. Вона включає гілкування при визначенні, чи автентифікований користувач, та демонструє дві альтернативні гілки: успішний вхід і помилку. Основна гілка завершується виведенням QR-коду на 10 секунд. Ця діаграма дозволяє наочно побачити логіку переходів між екранами додатку.

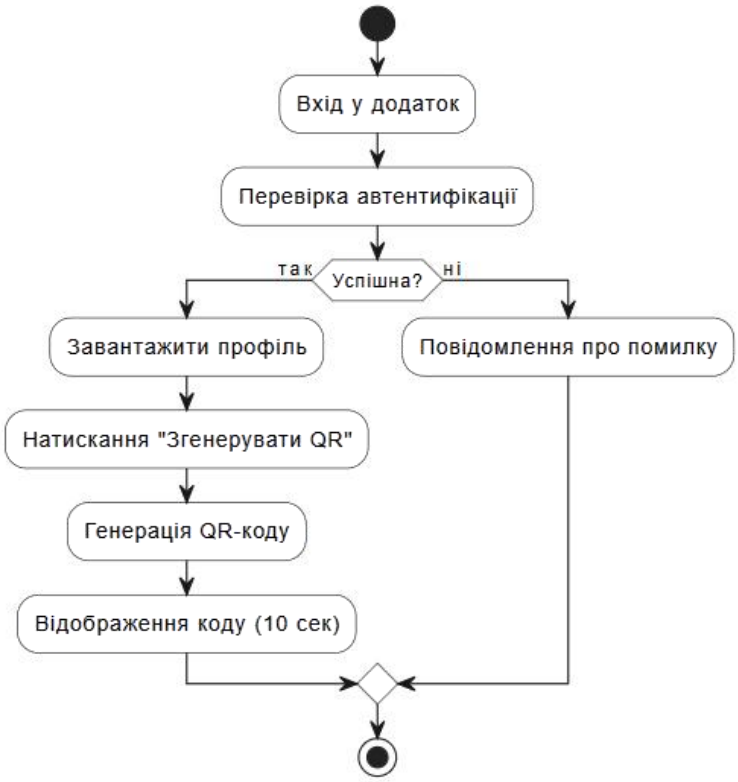

Рисунок 2.3 – Діаграма активностей

Діаграма компонентів (рис. 2.4) відображає архітектуру системи, розділену на дві частини: клієнтську (LNTU Pass – студент) і контролерську (LNTU Scan – охоронець). У студентському модулі є окремі компоненти для входу, перегляду профілю та генерації QR-коду. У контролерському модулі реалізовано компоненти для сканування та перегляду даних. Взаємодія обох застосунків відбувається через спільний компонент Firestore Client, що забезпечує обмін даними через хмарну базу Firestore.

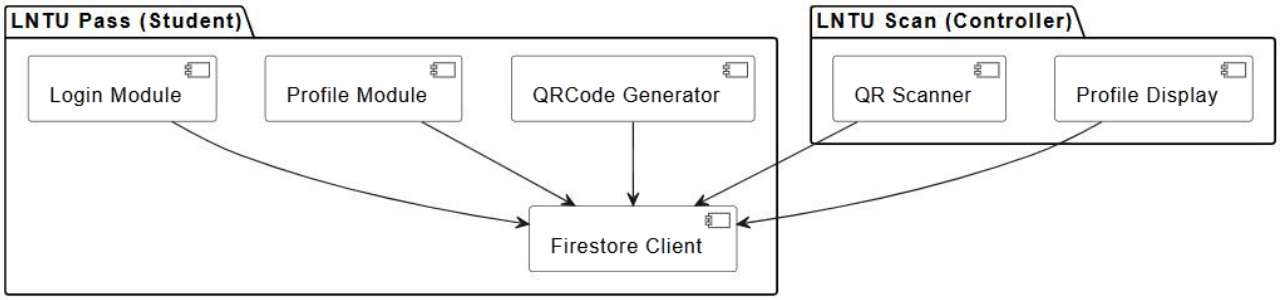

Рисунок 2.4 – Діаграма компонентів

Діаграма послідовності (рис. 2.5) детально демонструє етапи обробки QR-коду контролером. Після запуску сканування надсилається запит на дозвіл використання камери. Далі сканер зчитує QR-код, розшифровує дані та надсилає запит до Firestore для отримання даних користувача. У разі успішного запиту студентський профіль повертається до контролера й відображається на екрані. Діаграма дозволяє простежити повну послідовність між компонентами системи та забезпечує уявлення про реалізацію логіки перевірки QR-коду.

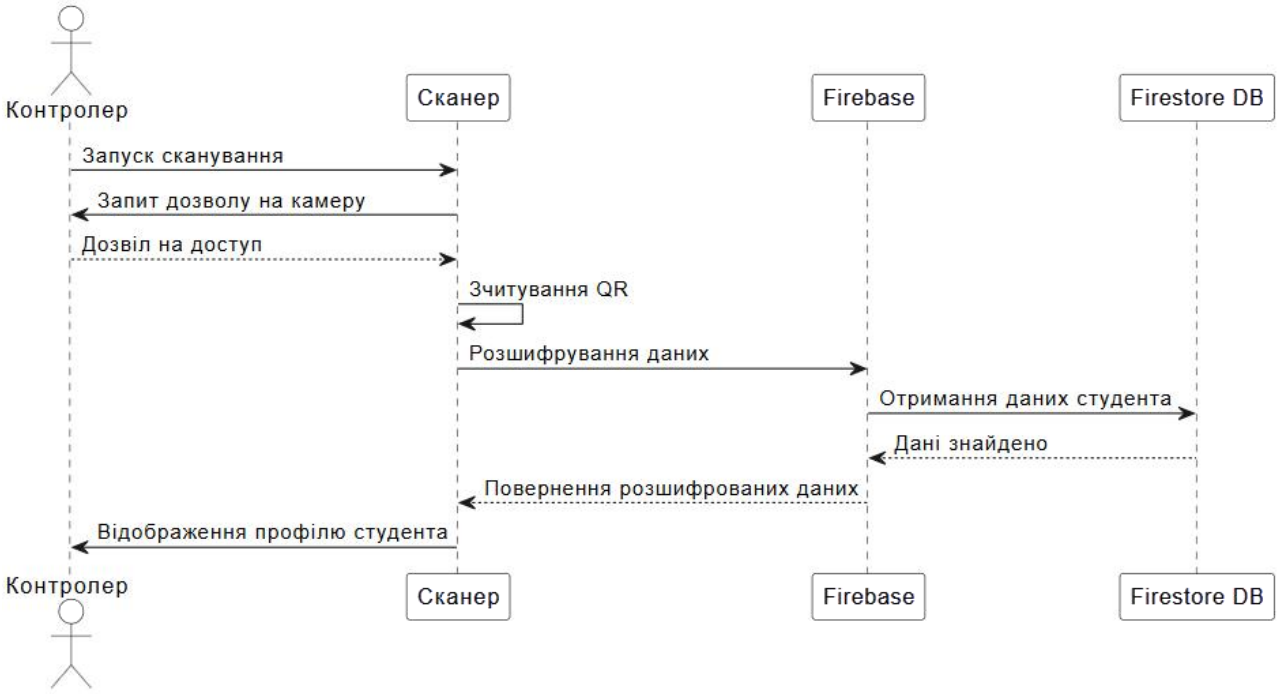

Рисунок 2.5 – Діаграма послідовності

Інтерфейс додатку реалізовано в Jetpack Compose, що дозволяє швидко будувати адаптивні компоненти. Кожне вікно (логін, профіль, генерація QR,

сканування) створено відповідно до стандартів Google Material Design. Потоки даних між екранами забезпечуються через ViewModel, що розділяє логіку і представлення.

Інформаційні потоки в системі охоплюють кілька послідовних етапів. Спочатку здійснюється передача облікових даних користувача (логін і пароль) до сервісу Firebase для автентифікації. Після успішного входу відбувається отримання персоналізованих даних з хмарної бази Firestore. На основі цих даних система виконує шифрування та генерує QR-код, який тимчасово відображається користувачу. На стороні контролера реалізується сканування цього коду з подальшим дешифруванням і перевіркою його дійсності. Завершальним етапом є зворотний зв’язок із користувачем у вигляді повідомлення про успішне зчитування або сповіщення про помилку чи недійсність перепустки.

Сформульовані функціональні та нефункціональні вимоги стали основою для побудови архітектури додатку, визначення ролей користувачів і сценаріїв їх взаємодії. В результаті було створено узгоджену модель системи, що враховує як потреби кінцевих користувачів, так і вимоги до безпеки, масштабованості та зручності інтерфейсу.

2.2 Вибір засобів, методів і алгоритмів вирішення поставленого завдання

У процесі розробки Android-додатку LNTU Pass для електронних перепусток було проаналізовано низку сучасних інструментів і технологій, які дозволяють реалізувати безпечну, масштабовану та зручну інформаційну систему. Оцінювалися як платформи для розробки інтерфейсу, так і бази даних, сервіси аутентифікації, методи шифрування та моделі архітектури застосунку.

Kotlin – це сучасна мова програмування, яка з 2017 року офіційно підтримується Google як основна мова для розробки Android-додатків. Kotlin створена для підвищення продуктивності розробника: вона поєднує у собі

лаконічність вираження, безпечну роботу з null-посиланнями (через систему Nullable/Non-nullable типів), підтримку функціонального програмування та повну сумісність із кодом на Java, що дозволяє поступово переносити існуючі Java-проєкти на Kotlin [6]. Крім того, Kotlin активно розвивається JetBrains і має широку екосистему плагінів, розширень та бібліотек, що робить її дуже привабливою для мобільної, серверної та навіть web-розробки (через Kotlin/JS).

Jetpack Compose – це сучасний декларативний фреймворк для створення UI в Android-додатках, що є частиною Android Jetpack [7]. Він дозволяє розробникам описувати інтерфейс у вигляді функцій, які автоматично оновлюються при зміні стану (state-driven UI). Це зменшує кількість коду, який необхідно писати та підтримувати, оскільки відсутня потреба в XML-розмітках або ж у ручному оновленні елементів UI через findViewById. Compose інтегрується з Android Studio, має вбудовану підтримку Material Design, а також підтримує анімації, теми, навігацію та створення власних UI-компонентів.

Firebase Firestore – це хмарна документоорієнтована NoSQL-база даних, розроблена Google як частина екосистеми Firebase [8]. Вона дозволяє зберігати структуровані дані у вигляді колекцій і документів та забезпечує синхронізацію даних у реальному часі між клієнтом і сервером. Firestore є ідеальним рішенням для мобільних додатків, оскільки підтримує офлайн-режим (дані кешуються на пристрої), має вбудовані правила доступу на рівні документів і колекцій (Firebase Security Rules), автоматично масштабується та інтегрується з іншими сервісами Firebase, такими як Authentication, Functions та Analytics. Проте її безкоштовний план має обмеження на кількість зчитувань, записів і з’єднань, що може стати викликом при великому навантаженні.

Firebase Authentication – це сервіс для управління обліковими записами користувачів, який дозволяє легко реалізувати автентифікацію у мобільних або веб-застосунках [9]. Серед підтримуваних методів входу – email/пароль, телефон, Google, Facebook, GitHub та інші OAuth-провайдери. Firebase Auth надає готовий бекенд для обробки входу, виходу, скидання пароля та захисту облікових даних. Він підтримує зберігання паролів із використанням безпечних

хеш-алгоритмів (наприклад, bcrypt), має інтеграцію з Firestore для керування ролями користувачів, і дозволяє гнучко налаштовувати правила доступу в залежності від ролі чи ідентичності користувача. Це рішення значно скорочує час розробки без втрати в безпеці.

Для захисту інформації, що передається через QR-коди, застосовано алгоритм симетричного шифрування AES (Advanced Encryption Standard). Цей алгоритм є одним із найнадійніших і найпоширеніших методів шифрування у світі [10]. Його використовують у банківських системах, державних структурах і популярних хмарних сервісах. У контексті розробленої системи AES використовується для шифрування персональних даних (імені, прізвища, факультету, кімнати, email тощо) перед вбудовуванням їх у QR-код. Під час сканування контролерський додаток локально розшифровує дані за допомогою того ж симетричного ключа, що унеможливлює прочитання QR-коду сторонніми особами без відповідного ключа. Такий підхід забезпечує конфіденційність і захист навіть у випадку перехоплення коду.

Архітектура мобільного додатку реалізована згідно з принципами шаблону MVVM (Model–View–ViewModel), який є сучасним підходом до розділення відповідальностей у програмному забезпеченні. Model відповідає за роботу з даними (отримання, збереження, обробку), View – за інтерфейс користувача (екрани), а ViewModel виступає як посередник, що керує логікою взаємодії та змін стану [11]. Це дозволяє розділити бізнес-логіку та UI, спрощує налагодження, підвищує повторне використання компонентів і дозволяє легше впроваджувати оновлення в майбутньому. Крім того, MVVM добре інтегрується з Jetpack Compose, що сприяє зручному створенню реактивних і адаптивних інтерфейсів.

Діаграма розгортання на рисунку 2.6 ілюструє архітектуру системи LNTU Pass, яка складається з двох мобільних застосунків – для студента і для контролера – та хмарної платформи Firebase.

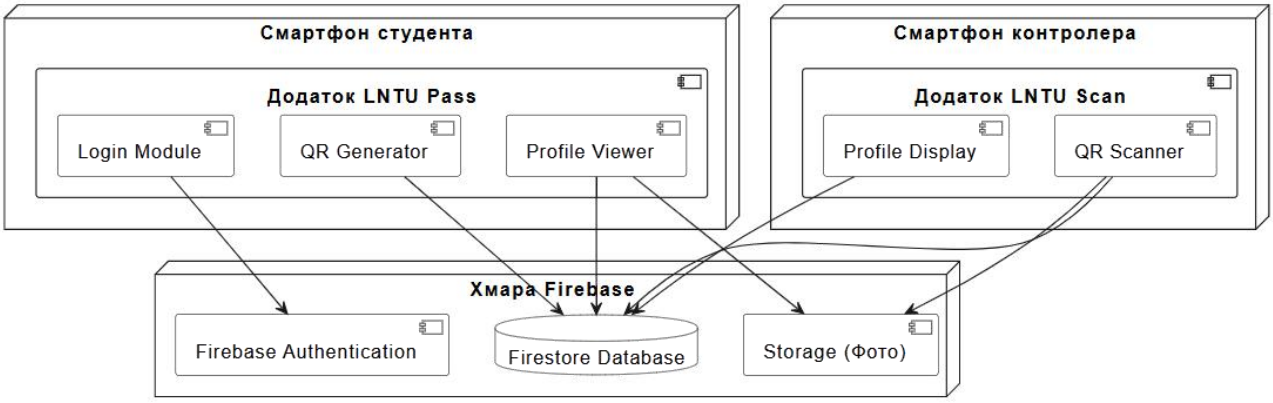

Рисунок 2.6 – Діаграма розгортання інформаційної системи перепусток

Додаток студента містить модулі для входу, перегляду профілю та генерації QR-коду, які взаємодіють із Firebase Authentication та Firestore. Додаток контролера включає сканер і перегляд профілю, що також звертаються до Firebase для розшифрування даних та відображення інформації. Уся взаємодія відбувається через захищене з’єднання, що забезпечує надійність і безпеку обміну даними. Для обробки QR-кодів використано бібліотеку ZXing (Zebra Crossing), яка є стандартом де-факто у сфері сканування штрихкодів в Android-додатках. Вона дозволяє ефективно зчитувати QR-коди з камери пристрою без потреби у складних налаштуваннях.

У підсумку, вибір зазначених інструментів і методів обумовлений їхньою відповідністю таким критеріям, як ефективність, зручність інтеграції, підтримка сучасних стандартів безпеки, а також доступність для мобільних пристроїв студентів і персоналу. Обрані технології дозволяють повною мірою реалізувати цілі кваліфікаційної роботи та забезпечити його подальшу підтримку й розширення.

Висновки до розділу 2

У другому розділі було здійснено формалізацію вимог до розроблюваної інформаційної системи, обґрунтовано вибір архітектурного підходу, інструментів реалізації та методів захисту даних. Сформовано функціональні та

нефункціональні вимоги до мобільного додатку LNTU Pass і системи сканування QR-кодів контролерами.

Використання архітектури MVVM, мови Kotlin та декларативного інтерфейсу Jetpack Compose забезпечує високу гнучкість і зручність розробки. Обрана база даних Firestore дозволяє реалізувати швидку синхронізацію даних у хмарному середовищі з урахуванням потреб мобільного застосування. Для безпечного зберігання та передачі персональних даних впроваджено автентифікацію через Firebase Authentication і шифрування QR-кодів за допомогою алгоритму AES.

За допомогою моделювання у нотації UML розроблено діаграми варіантів використання, класів, послідовності, активностей, компонентів і розгортання, що описують структуру системи, сценарії взаємодії та її фізичну реалізацію. Створені специфікації та таблиці подій дозволили точно визначити логіку взаємодії між компонентами системи та користувачами.

Загалом розділ заклав методологічну основу для подальшої реалізації проєкту та довів доцільність застосування сучасних інструментів мобільної розробки у поєднанні з хмарними технологіями для забезпечення зручності, безпеки та ефективності електронної перепусткової системи в умовах університетської інфраструктури.

РОЗДІЛ 3
РОЗРОБКА ANDROID-ДОДАТКУ ДЛЯ ЕЛЕКТРОННИХ ПЕРЕПУСТОК В ГУРТОЖИТОК ЛНТУ

3.1 Практична реалізація об’єкта проєктування

Практична реалізація об’єкта проєктування охоплює повний цикл створення двох мобільних застосунків – LNTU Pass (для студентів) та LNTU Scann (для контролю доступу) – з використанням мови програмування Kotlin, інтерфейсного фреймворку Jetpack Compose, бібліотеки ZXing та хмарної платформи Firebase. Ці застосунки утворюють єдину систему для генерації, шифрування, сканування та розшифрування QR-кодів, які містять персоніфіковану інформацію про студентів.

MyApplication реалізовано за допомогою декларативного підходу Jetpack Compose, що дозволило суттєво спростити побудову користувацького інтерфейсу, підвищити читаємість коду та полегшити підтримку застосунку. Архітектура включає наступні компоненти:

1) екрани (LoginScreen, QrScreen, UserInfoScreen);
2) компоненти UI (UnderlineTextField, UnderlinedLabel, LoadingProfileScreen);
3) утиліти (QrCodeUtils – модуль генерації та шифрування QR-кодів);
4) тема (LntuBlue, кольорова палітра та типографіка);
5) навігація між екранами за допомогою NavHost і NavController.

LNTU Scann реалізує функціональність розпізнавання QR-коду за допомогою бібліотеки ZXing (IntentIntegrator) та подальше розшифрування зашифрованих даних через алгоритм AES/ECB/PKCS5Padding. У застосунку передбачено відображення профілю студента з валідацією терміну дії коду.

Застосунок LNTU Pass під’єднано до Firebase Authentication для авторизації користувачів та Cloud Firestore для зберігання інформації про користувачів і QR-коди. Усі дії з базою реалізовано асинхронно за допомогою стандартних засобів Firebase SDK.

Розробка здійснювалась у середовищі Android Studio з Gradle-проєктами. Структура проєкту поділена на пакети data, screens, components, utils тощо. Нижче наведено структуру головного класу даних (ліст. 3.1).

Лістинг 3.1 – Структура головного класу даних

```
// User.kt (MyApplication, ScanApp)
data class UserProfile(
    val firstName: String = "",
    val lastName: String = "",
    val faculty: String = "",
    val room: String = "",
    val photoUrl: String = ""
)
```

кінець лістингу 3.1

Для введення логіна та пароля використано власні компоненти із стилізованими підписами та підкресленням (рис. 3.1).

Рисунок 3.1 – Вікно авторизації

Лістинг 3.2 – Форма входу та UI-компоненти

```
@Composable
fun UnderlineTextField(
    value: String,
    onValueChange: (String) -> Unit,
    placeholderText: String,
    isPassword: Boolean = false,
    showPassword: Boolean = false,
    onShowPasswordChange: (Boolean) -> Unit = {}
) {
    val visualTransformation =
        if (isPassword && !showPassword) PasswordVisualTransformation()
        else VisualTransformation.None


    OutlinedTextField(
        value = value,
        onValueChange = onValueChange,
        placeholder = { Text(placeholderText, fontSize = 18.sp) },
        textStyle = TextStyle(fontSize = 20.sp),
        visualTransformation = visualTransformation,
        trailingIcon = if (isPassword) {
            {
                IconButton(onClick =
{ onShowPasswordChange(!showPassword) }) {
                    Icon(
                        if (showPassword) Icons.Filled.VisibilityOff else
Icons.Filled.Visibility,
                        contentDescription = null
                    )
                }
            }
        } else null,
        modifier = Modifier.fillMaxWidth()
    )
    Divider(color = LntuBlue, thickness = 4.dp)
}
```

кінець лістингу 3.2

Основною частиною системи є алгоритм формування QR-коду, який шифрує дані студента (ліст. 3.3).

Лістинг 3.3 – Генерація QR-коду (MyApplication)

```
fun generateQRCodeBitmap(profile: UserProfile, email: String): Bitmap {
    val rawData = """
        Ім’я: ${profile.firstName}
        Прізвище: ${profile.lastName}
```

```
        Факультет: ${profile.faculty}
        Кімната: ${profile.room}
        Email: $email
        GeneratedAt: ${Timestamp.now().seconds}
        PhotoUrl: ${profile.photoUrl}
    """.trimIndent()

    val encryptedData = encryptData(rawData)
    val deepLink = "app://scan-service/data=$encryptedData"

    val bitMatrix = MultiFormatWriter().encode(deepLink,
BarcodeFormat.QR_CODE, 512, 512)
    val bmp = Bitmap.createBitmap(512, 512, Bitmap.Config.RGB_565)
    for (x in 0 until 512) for (y in 0 until 512) {
        bmp.setPixel(x, y, if (bitMatrix[x, y]) Color.BLACK else
Color.WHITE)
    }
    return bmp
}
```

кінець лістингу 3.3

Сканований QR-код розпізнається і дешифрується через AES-алгоритм (ліст. 3.4).

Лістинг 3.4 – Розшифрування даних (ScanApp)

```
fun decryptData(encryptedData: String): String {
    val cipher = Cipher.getInstance("AES/ECB/PKCS5Padding")
    cipher.init(Cipher.DECRYPT_MODE, generateKey())
    val decodedBytes = Base64.decode(encryptedData, Base64.DEFAULT)
    return String(cipher.doFinal(decodedBytes), StandardCharsets.UTF_8)
}
```

кінець лістингу 3.4

Додаток для сканування перевіряє, чи не сплив час дії QR-коду (10 секунд) (ліст. 3.5).

Лістинг 3.5 – Валідація терміну дії QR-коду

```
val currentTime = Timestamp.now().seconds
if (currentTime - generatedAtSeconds > 10) {
    Toast.makeText(this, "QR-код прострочений", Toast.LENGTH_SHORT).show()
    return
}
```

кінець лістингу 3.5

У застосунку використано Jetpack Navigation для навігації між екранами (ліст. 3.6).

Лістинг 3.6 – Навігація між екранами

```
NavHost(navController = navController, startDestination =
Screen.LOGIN.route) {
    composable(Screen.LOGIN.route) { LoginScreen(...) }
    composable(Screen.USER_INFO.route) { UserInfoScreen(...) }
    composable(Screen.QR.route) { QrScreen(...) }
}
```

кінець лістингу 3.6

На рисунку 3.2 зображено вікно генерації QR-коду з згенерованим кодом.

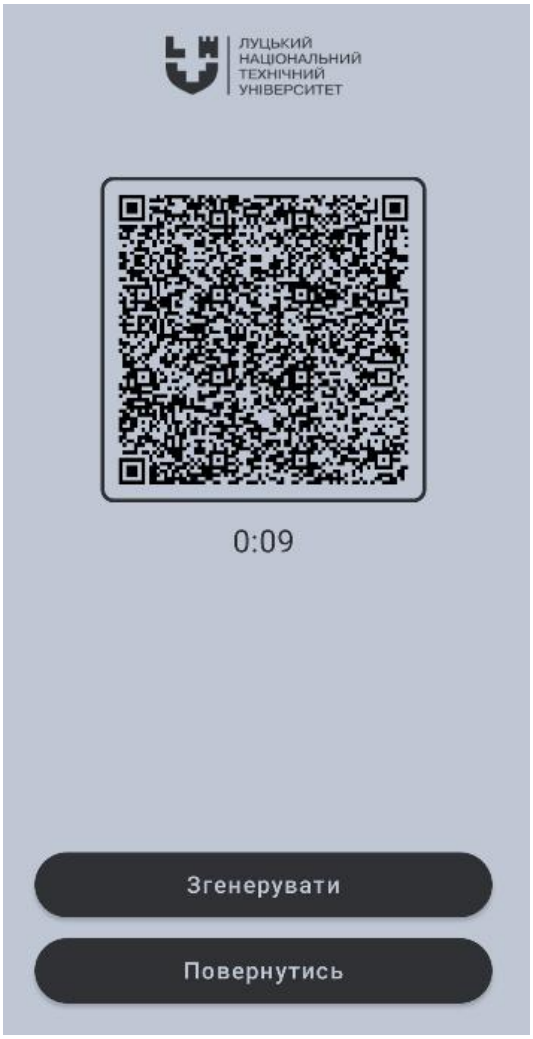

Рисунок 3.2 – Вікно генерації QR-коду

3.2 Тестування та налагодження інформаційно-комп’ютерної системи

Тестування програмного продукту проводиться для виявлення та усунення помилок, перевірки відповідності функціоналу встановленим вимогам, забезпечення стабільної роботи системи в різних умовах і на різних пристроях, а також для підвищення якості, надійності й безпеки додатку перед його впровадженням у реальне користування.

Процес тестування і налагодження розробленого Android-додатку здійснювався на всіх етапах розробки з метою виявлення помилок, перевірки стабільності функціоналу та відповідності системи заданим вимогам. Основну увагу було зосереджено на модульному, інтеграційному та системному тестуванні. Для налагодження використовувалося середовище розробки Android Studio з емуляторами різних моделей пристроїв та фізичними смартфонами на Android 11 і вище.

Для перевірки окремих частин логіки наприклад, розрахунків сум витрат, використовувалося модульне тестування (unit testing) з використанням kotlin.test Нижче наведено приклад простого тесту для перевірки функції обчислення залишку (ліст. 3.7).

Лістинг 3.7 – Приклад тесту

```
class BalanceCalculatorTest{
  @Test
  fun testBalanceCalculation() {
    val income = 10000.0
    val expenses = 7500.0
    val balance = income – expenses

    assertEquals(2500.0, balance, 0.0)
  }
}
```

кінець лістингу 3.7

Також було проведено інтеграційне тестування взаємодії користувача з основними екранами застосунку (навігації, аналізу, налаштувань), що дало змогу перевірити злагодженість роботи компонентів.

Для системного тестування було використано реальні пристрої (Pixel 5, Samsung A32), де перевірялась повна функціональність додатку: авторизація, генерація QR-коду, його сканування та зчитування на іншому пристрої. Зокрема, проводилась перевірка таких сценаріїв: успішна авторизація, некоректне введення пароля, передача зашифрованого QR-коду іншому пристрою та обробка помилки при відсутності інтернету або відключеному Firestore.

Мінімальні системні вимоги:

1) ОС: Android 8.0 (API 26);
2) CPU: 1.4 ГГц, 4 ядра;
3) RAM: 2 ГБ;
4) вільне місце: 50 МБ;
5) дозвіл екрану: 720×1280.

Рекомендовані системні параметри:

1) ОС: Android 11+;
2) CPU: 2.0 ГГц, 8 ядер;
3) RAM: 4 ГБ і більше;
4) вільне місце: 100 МБ;
5) підтримка сканера QR-кодів та камери.

У процесі налагодження було виявлено та усунуто низку помилок, що впливали на коректну роботу додатку. Зокрема, була виявлена помилка, пов’язана з некоректною валідацією порожніх полів під час додавання запису – її усунено шляхом впровадження TextFormField із вбудованою перевіркою обов’язкових значень.

Крім того, під час завантаження даних із Firestore спостерігалися збої при відсутності інтернет-з’єднання, що було вирішено шляхом додавання обробки

винятків для забезпечення стабільності роботи системи в офлайн-режимі або при нестабільному з’єднанні.

Результати проведеного тестування системи узагальнено у таблиці 3.1.

Таблиця 3.1 – Результати тестування мобільного додатку

Тип тестування	Опис	Результат
Модульне тестування	Перевірка функції обчислення залишку доходів та витрат	Успішно
Інтеграційне тестування	Перевірка навігації між екранами, збереження та читання даних	Успішно
Системне тестування	Повна перевірка, авторизації, генерації та сканування QR-кодів	Успішно
Тест на відсутність інтернету	Повідомлення про помилку, стабільність інтерфейсу	Успішно
Тест валідації форм	Виявлення та виправлення помилки при порожніх полях	Успішно
Оновлення інтерфейсу	Рішення проблеми з оновленням UI після додавання запису	Успішно

Загалом, після завершення налагодження система стабільно функціонує, забезпечуючи повний цикл ведення обліку фінансів користувачем: від реєстрації до аналізу статистики та обміну даними через QR-коди.

3.3 Розробка бази даних

Попри те, що обрана платформа реалізує більшість функцій на базі Firebase, розробка логічної моделі бази даних залишалася важливою складовою проєкту. У рамках цього підрозділу було здійснено моделювання та реалізацію структури зберігання даних у хмарному сховищі Firestore, що виконує функцію NoSQL бази даних у середовищі Firebase.

Основними сутностями в системі є:

1) Users (користувачі) – зберігає унікальні ідентифікатори користувачів, їх email-адреси, імена та ролі (наприклад, «user» або «controller»);

2) AccessRecords (записи доступу) – фіксує дату, час, результат сканування QR-коду, інформацію про вхід/вихід, а також UID користувача, що проходив контроль;

3) QRData – необов’язкова допоміжна структура, яка використовується для зберігання попередньо згенерованих QR-кодів.

Діаграма на рисунку 3.3 описує структуру бази даних мобільного додатку LNTU Pass, що використовується для електронної перевірки перепусток студентів на територію гуртожитку. У діаграмі представлені три основні сутності – User, QRCode та Session, між якими встановлені логічні зв’язки.

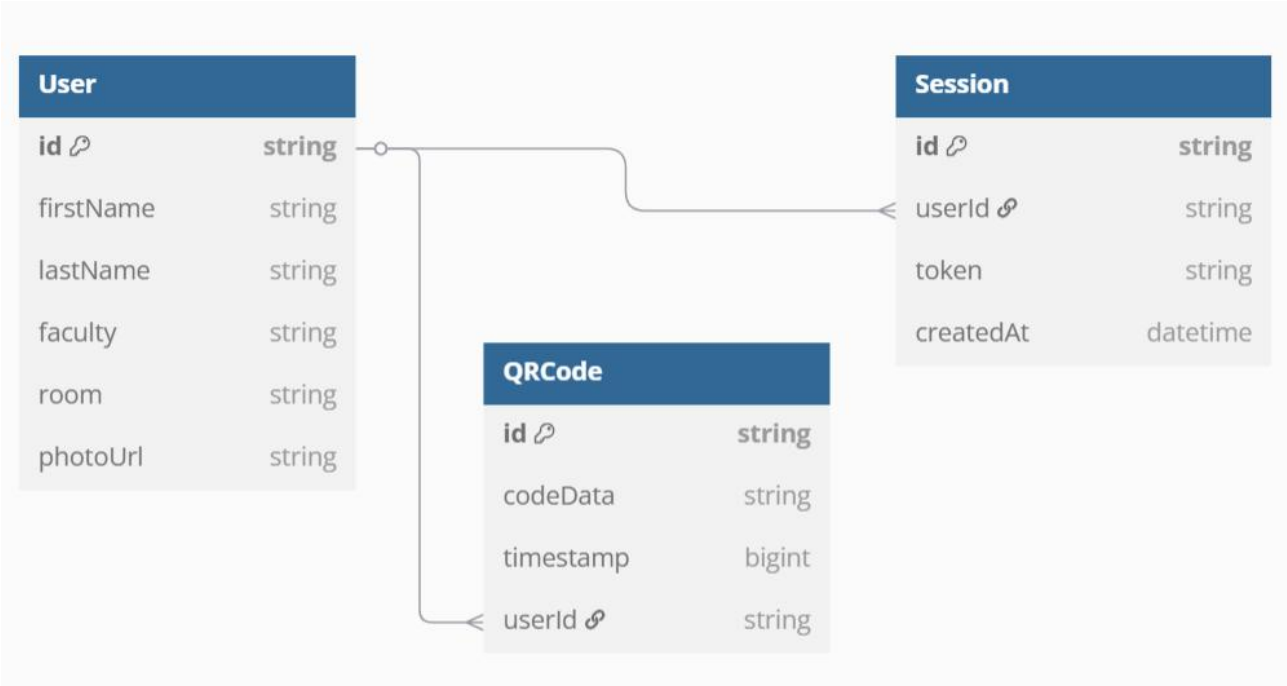

Рисунок 3.3 – Діаграма бази даних

Сутність User є центральною в системі. Вона містить основні поля для ідентифікації студента:

– унікальний ідентифікатор id;

– ім’я (firstName);

– прізвище (lastName);

– факультет (faculty);

– номер кімнати (room);

– посилання на фотографію (photoUrl).

Ці дані відображаються в особистому профілі користувача після входу в додаток (рис. 3.4).

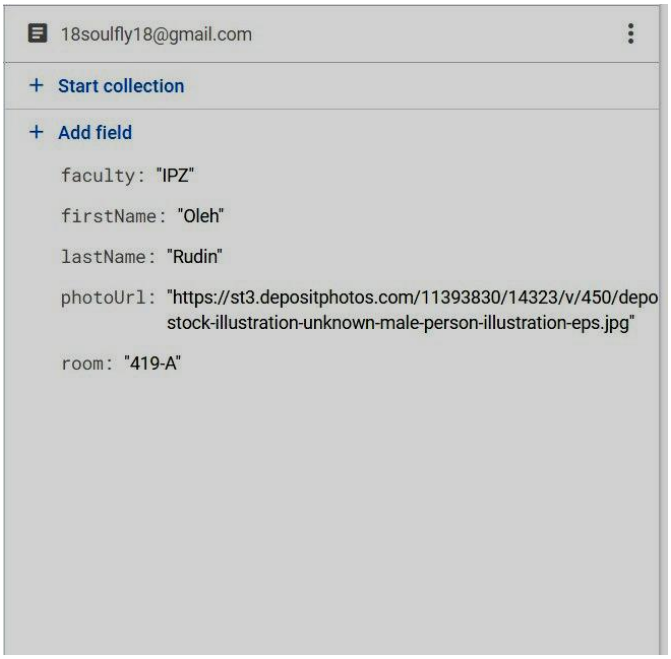

Рисунок 3.4 – Сутність User

На рисунку 3.5 зображено профіль студента, в який вигружаються дані з сутності User.

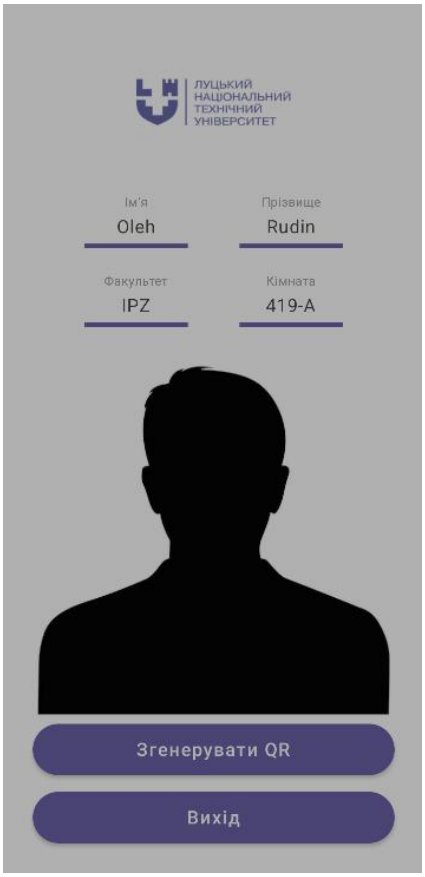

Рисунок 3.5 – Профіль студента

Сутність QRCode зберігає інформацію про згенеровані QR-коди. Вона включає унікальний ідентифікатор id, поле codeData, що містить зашифровану інформацію, а також часову мітку timestamp, яка використовується для перевірки актуальності QR-коду. Кожен QR-код пов’язаний із конкретним студентом через зовнішній ключ userId, який посилається на поле id у таблиці User.

Сутність Session відповідає за збереження даних про активні сесії авторизації. Вона включає id сесії, userId для прив’язки до користувача, token для підтвердження доступу та createdAt – дату і час створення сесії. Ця таблиця необхідна для контролю безперервності входу та може бути використана для забезпечення додаткових рівнів безпеки.

Між таблицями User і QRCode, а також User і Session встановлені зв’язки типу «один до багатьох»: один користувач може мати кілька QR-кодів і кілька сесій входу. Це забезпечує масштабованість системи та можливість гнучкого керування доступом, зберігаючи при цьому безпеку й простоту реалізації.

У процесі реалізації використовувались методи CRUD (Create, Read, Update, Delete) для взаємодії з Firestore. Зокрема, для додавання нового запису використовувався наступний фрагмент коду (ліст. 3.8).

Лістинг 3.8 – Додавання нового запису

```
FirebaseFirestore.instance.collection('AccessRecords').add({
  'uid': user.uid,
  'timestamp': Timestamp.now(),
  'status': 'entered',
});
```

кінець лістингу 3.8

Для отримання історії доступу певного користувача використовується наступний код (ліст. 3.9).

Лістинг 3.9 – Отримання історії доступу певного користувача

```
FirebaseFirestore.instance
```

```
    .collection('AccessRecords')
    .where('uid', isEqualTo: user.uid)
    .orderBy('timestamp', descending: true)
    .get();
```

кінець лістингу 3.9

Для реалізації перевірки статусу користувача перед наданням доступу використовується наступний код (ліст. 3.10).

Лістинг 3.10 – Реалізація перевірки статусу користувача

```
FirebaseFirestore.instance
    .collection('Users')
    .doc(user.uid)
    .get()
    .then((doc) {
      if (doc.exists && doc['role'] == 'user') {
        // дозвіл на вхід
      }
    });
```

кінець лістингу 3.10

Firestore не потребує попереднього визначення структури таблиць або явного створення SQL-запитів типу CREATE TABLE, оскільки працює як документно-орієнтована база даних. Проте підхід до структурування колекцій і документів, їх зв'язки та правила доступу вимагають ретельного планування та реалізації, аналогічного традиційному реляційному підходу з MySQL чи phpMyAdmin.

Загалом, база даних була реалізована як гнучка, масштабована система, що забезпечує швидкий доступ до інформації про вхід/вихід користувачів, підтримує реальний час оновлення і зберігає надійність даних через вбудовані механізми Firestore.

3.4 Захист інформаційно-комп’ютерної системи

Механізми захисту впроваджуються для забезпечення конфіденційності, цілісності та доступності даних, запобігання несанкціонованому доступу, захисту від зловмисних дій і витоків інформації, а також для підвищення довіри користувачів до системи та відповідності стандартам інформаційної безпеки.

У процесі розробки інформаційно-комп’ютерної системи було впроваджено комплекс базових, але ефективних механізмів безпеки, які певною мірою забезпечують захист персональних даних користувачів, цілісність інформації та обмеження доступу до конфіденційних функцій відповідно до ролей. Основним засобом автентифікації є Firebase Authentication, який дозволяє користувачам здійснювати вхід і реєстрацію за допомогою електронної пошти та пароля. Firebase автоматично забезпечує хешування паролів за допомогою сучасного алгоритму bcrypt, що ускладнює злам даних у разі витоку. Firebase також блокує спроби підбору паролів, що є ефективним захистом від brute-force атак.

Взаємодія між клієнтським застосунком і серверною базою даних Firestore відбувається через протокол HTTPS, який базується на TLS і забезпечує шифрування переданих даних у процесі мережевої комунікації. Завдяки цьому всі запити, зокрема автентифікаційні, збереження та отримання інформації, відбуваються у зашифрованому вигляді, що унеможливлює їхнє перехоплення третіми особами.

Важливим захисним елементом є контроль виняткових ситуацій: наприклад, при відсутності інтернету користувач отримує повідомлення про збій замість краху додатку, що знижує ризик втрати інформації та підвищує стабільність роботи програми.

Основним елементом захисту є шифрування QR-кодів. Уся інформація, яка потрапляє у QR, проходить через алгоритм AES (Advanced Encryption Standard) у режимі ECB з паддінгом PKCS5. Ключ для шифрування генерується за допомогою SHA-256 з хешуванням секретної фрази. Це унеможливлює

прочитання даних у QR-коді сторонніми особами, навіть якщо їм вдалося зберегти зображення чи скопіювати вміст. На боці контролера реалізовано функцію дешифрування, яка дозволяє переглядати дані лише за наявності правильного ключа. З метою запобігання багаторазовому використанню коду реалізовано механізм перевірки дійсності за часовою міткою: код діє лише протягом 10 секунд.

Для ускладнення несанкціонованого доступу до внутрішньої логіки застосунку використано обфускацію коду під час релізної збірки. Завдяки використанню Android-інструментів типу R8 здійснюється стискання, оптимізація та перейменування імен класів, змінних і методів, що значно ускладнює спроби декомпіляції та реверс-інжинірингу. Це не є абсолютним захистом, але формує додатковий бар'єр для потенційних зловмисників. Основні механізми безпеки, реалізовані у мобільному додатку, узагальнено в таблиці 3.2.

Таблиця 3.2 – Основні механізми захисту інформаційно-комп'ютерної системи LNTU Pass

Компонент системи	Засіб захисту	Опис реалізації та ефекту
Аутентифікація користувача	Firebase Authentication + bcrypt	Захищене зберігання паролів, захист від brute-force
Передача даних	HTTPS (TLS)	Шифрування трафіку між клієнтом і сервером
Розділення доступу	Розмежування доступу за типом користувача	Відокремлені додатки для студента і контролера
QR-коди	AES-шифрування + SHA-256 ключ	Захист вмісту QR-коду від несанкціонованого доступу
Валідація QR-кодів	Часова мітка + перевірка актуальності	Захист від повторного використання
Логіка застосунку	Обфускація коду через R8	Ускладнення декомпіляції, захист внутрішньої логіки
Обробка винятків	Повідомлення про помилки замість аварійного завершення	Підвищення стабільності при втраті з'єднання

Усі впроваджені заходи захисту відповідають сучасним стандартам безпеки для мобільних застосунків середнього рівня складності. Вони забезпечують достатню надійність та стабільність роботи системи в умовах університетського середовища. У майбутньому можливе розширення

функціоналу безпеки – зокрема впровадження логування подій, двофакторної автентифікації, геолокаційної перевірки або часових обмежень доступу, що дозволить адаптувати систему для ширшого спектра потреб і користувачів.

3.5 Розробка документації для програмного продукту

Для забезпечення повноцінного функціонування розробленої інформаційної системи користувач повинен мати пристрій з операційною системою Android 8.0 (Oreo) або новішою версією. Рекомендовані апаратні характеристики: щонайменше 2 ГБ оперативної пам’яті, 100 МБ вільного простору на диску, активне підключення до Інтернету, підтримка Google Play Services.

Довідкові матеріали інтегровані у додаток у вигляді підказок (tooltip) та структурованих повідомлень інтерфейсу. Наприклад, при першому використанні додатку запускається онбординг (вступний екран із покроковим поясненням основних функцій), а кожна кнопка має пояснювальну піктограму або текстову підказку.

Інсталяція додатку відбувається звичайним способом – через файл APK або Google Play (у разі публікації). Для локального запуску з середовища розробки Android Studio слід:

1) відкрити проєкт у Android Studio (версія 2022.3.1 або новіша);
2) перевірити наявність актуального SDK (мінімум API 26);
3) налаштувати з’єднання з Firebase:
    а) створити проєкт у Firebase Console;
    б) додати додаток до Firebase-проєкту;
    в) завантажити google-services.json до папки app/;
    г) увімкнути Firestore, Authentication (email/password), Storage.
4) у файлі build.gradle повинні бути підключені наступні плагіни: classpath "com.google.gms:google-services:4.3.15" та у build.gradle(:app): apply plugin: "com.google.gms.google-services";

5) синхронізувати проєкт і запустити на емуляторі або реальному пристрої.

Довідка для користувача реалізована у вигляді секції «Допомога» в інтерфейсі додатку. Вона містить найпоширеніші запитання та відповіді (FAQ), короткі відео-інструкції та контакт для підтримки.

Усі повідомлення про помилки, валідацію полів та сповіщення мають зрозумілу, адаптовану до користувача форму. Наприклад, при відсутності Інтернету виводиться діалог з поясненням і кнопкою повторити спробу. У разі потреби додаток може бути адаптований під інші мови інтерфейсу за допомогою Android ресурсів strings.xml.

Отже, користувачеві надано повний комплекс засобів для встановлення, налаштування та безпроблемного використання інформаційної системи. Завдяки зручному онбордингу, вбудованим підказкам і системі довідки, навіть користувач без досвіду роботи з подібними застосунками зможе швидко освоїти функціонал. Крім того, реалізована підтримка багатомовності, гнучкі повідомлення про помилки та адаптація під різні пристрої забезпечують інклюзивність і комфорт у користуванні. Це робить додаток придатним для масштабного впровадження в навчальних закладах, де потрібні прості, надійні та доступні рішення контролю доступу.

Висновки до розділу 3

У третьому розділі кваліфікаційної роботи здійснено повну практичну реалізацію проєктованої інформаційно-комп’ютерної системи – Android-додатку для збереження, обробки та шифрування персональних даних з підтримкою сканування та генерації QR-кодів. Було поетапно реалізовано інтерфейс користувача на основі сучасної технології Jetpack Compose, що дозволило забезпечити адаптивний, зручний і сучасний дизайн із підтримкою реактивних компонентів. Архітектура додатку побудована із застосуванням

шаблону MVVM, ViewModel, LiveData та Firebase Firestore як основної хмарної бази даних.

Реалізація бази даних охоплювала створення колекцій і документів у Firebase з підтримкою збереження інформації про профіль користувача, унікальні зашифровані значення, синхронізацію змін у режимі реального часу. Під час налагодження було виявлено та усунуто низку функціональних помилок, пов’язаних із валідацією введення, оновленням інтерфейсу, обробкою винятків при втраті з’єднання з мережею.

У ході реалізації особливу увагу приділено захисту інформації. Впроваджено обов’язкову автентифікацію користувачів із перевіркою електронної пошти. Для захисту QR-кодів використано алгоритм шифрування з використанням унікального симетричного ключа, що дозволяє безпечно кодувати дані перед їх візуалізацією та скануванням. Відповідно, забезпечується приватність переданих даних у межах додатку.

Проведено тестування функціоналу додатку з використанням емуляторів та реальних пристроїв. Перевірено стійкість системи до нестабільного підключення до Інтернету, коректність роботи всіх функціональних модулів, швидкодію інтерфейсу, точність шифрування та декодування QR-кодів. Система продемонструвала стабільну роботу, коректну обробку помилок та відповідність поставленим вимогам.

Також було створено супровідну технічну документацію з описом інструкцій для користувачів, вимог до середовища виконання, а також покроковою інструкцією зі встановлення, налаштування середовища та розгортання застосунку. Особливої уваги надано рекомендаціям щодо інтеграції з Firebase та забезпеченню стабільної синхронізації.

Отже, результати реалізації доводять працездатність, функціональність та безпеку створеної системи. Додаток може бути використаний як основа для створення ширших систем збереження та обміну зашифрованою інформацією з високим рівнем довіри до даних користувачів.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи на тему «Розробка Android-додатку для електронних перепусток в гуртожиток ЛНТУ з використанням Kotlin Jetpack Compose та Firestore Database» було повністю реалізовано поставлену мету – створити мобільну інформаційно-комп’ютерну систему, яка дозволяє надійно зберігати, передавати та обробляти персональні дані користувача з використанням сучасних технологій мобільної розробки, хмарного сховища та засобів шифрування.

Проаналізовано сучасні рішення автоматизації контролю доступу до студентських гуртожитків було виявлено ключові функціональні можливості конкурентних систем, їхні сильні та слабкі сторони, що дало змогу окреслити вимоги до власного продукту й запозичити найкращі практики, усуваючи знайдені недоліки.

Було побудовано UML-діаграм системи, а саме прецедентів, класів, послідовностей і діяльності дозволила чітко візуалізувати логіку роботи додатку й закласти фундамент для подальшої розробки, мінімізувавши ризики архітектурних помилок.

Для розробки було обрано мови Kotlin, технології Jetpack Compose та Firestore Database обґрунтовано їхньою високою ефективністю, гнучкістю для створення сучасного інтерфейсу і потужними можливостями хмарного зберігання даних, що забезпечило відмінну продуктивність і масштабованість рішення.

Розроблений прототип Android-додатку продемонстрував працездатність механізмів створення, візуалізації та перевірки електронних перепусток, а також модулів авторизації та доступу, підтвердивши обрану архітектуру та інтеграцію з Firestore.

Проектування схеми даних у Firestore Database із визначенням колекцій і документів забезпечило логічну структуру збереження інформації та реалізацію

ефективної взаємодії клієнтського додатку з хмарною базою, що гарантує цілісність і доступність даних.

Було реалізавано базові механізми захисту додатку, включно з аутентифікацією, розмежуванням прав доступу та валідацією даних, дозволила забезпечити безпечний рівень роботи системи та запобігти несанкціонованому доступу до ресурсів.

Проведене тестування та налагодження програмного продукту на реальному пристрої Android підтвердили коректність функціонування основних модулів і зручність інтерфейсу, виявили та усунули дрібні помилки, що підвищило стабільність роботи додатку.

Підготовлена технічна документація містить інструкції користувача, опис інтерфейсів, вимоги до встановлення й оновлення, що значно полегшує впровадження й подальшу підтримку розробленого рішення.

Загалом, створений продукт має високу прикладну цінність, може бути адаптований до різних сценаріїв використання та має перспективу подальшого розвитку. Реалізовані рішення підтверджують професійні навички здобувача щодо створення функціонального, безпечного і зручного мобільного додатку з орієнтацією на користувача.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Поляков М. О. Створення інформаційно-облікової системи поселення студентів у гуртожиток НУХТ. Київ: НУХТ. 2020. 87 с.

2. Щебетін Б. Ю. Автоматизована система управління деканат. Київ: КПІ ім. Ігоря Сікорського. 2022. 101 с.

3. Скороход А. А. Системи автоматизації розселення студентів по гуртожиткам. Суми: СумДУ. 2021. 77 с.

4. Bhandari P. Digital Transformation in Student Living: A Design Case Study on Overcoming Dormitory Challenges (Doctoral dissertation, Institute of Business Informatics and New Media, University of Siegen). 2024. 71 p.

5. Wen K., Fang Y. Daily Information Management System for Postgraduates to Check In and Out of the Dormitory Based on Mobile Edge Computing. Mobile Information Systems. 2021. Vol. 2021. P. 1-12. URL: https://doi.org/10.1155/2021/5167395 (date of access: 15.02.2025).

6. Kotlin Programming Language. URL: https://kotlinlang.org/ (date of access: 17.02.2025).

7. Jetpack Compose UI App Development Toolkit – Android Developers. Android Developers. URL: https://developer.android.com/compose (date of access: 28.02.2025).

8. Firestore. URL: https://firebase.google.com/docs/ firestore (date of access: 03.03.2025).

9. Chougale P., Yadav V., Gaikwad A., Vidyapeeth B. Firebase-overview and usage. International Research Journal of Modernization in Engineering Technology and Science. 2021. Vol. 3. P. 1178-1183. URL: https://surl.li/dhhyxu (date of access: 14.03.2025).

10. Muttaqin K., Rahmadoni J. Analysis and design of file security system AES (advanced encryption standard) cryptography based. Journal of Applied Engineering and Technological Science (JAETS). 2020. Vol. 1. P. 113-123. URL: https://core.ac.uk/reader/335293260 (date of access: 21.03.2025).

11. García R. F. MVVM: model–view–viewmodel. In iOS Architecture Patterns: MVC, MVP, MVVM, VIPER, and VIP in Swift. Berkeley, CA: Apress. 2023. P. 145-224. URL: https://link.springer.com/chapter/10.1007/978-1-4842-9069-9_4 (date of access: 25.03.2025).

12. Künneth T. Android UI Development with Jetpack Compose. Bring declarative and native UI to life quickly and easily on Android using Jetpack Compose and Kotlin. Packt Publishing Ltd. 2023. 278 p.

13. Ghita C. Kickstart Modern Android Development with Jetpack and Kotlin. Enhance your applications by integrating Jetpack and applying modern app architectural concepts. Packt Publishing Ltd. 2022. 472 p.

14. Wambua M. S. Modern Android 13 Development Cookbook. Over 70 recipes to solve Android development issues and create better apps with Kotlin and Jetpack Compose. Packt Publishing Ltd. 2023. 322 p.

15. Guśpiel M. Mobile App Development Using Kotlin Multiplatform. 2024. 39 p.

16. Kesavan R., Gay D., Thevessen D., Shah J., Mohan C. Firestore: The nosql serverless database for the application developer. In 2023 IEEE 39th International Conference on Data Engineering (ICDE). 2023. P. 3376-3388.

17. Kot S., Smołka J. performance analysis of a cloud database on mobile devices. Journal of Computer Sciences Institute. 2023. Vol. 29. P. 360–365. URL: https://doi.org/10.35784/jcsi.3798 (date of access: 15.04.2025).

18. Laurence P. O., Hinchman-Dominguez A., Meike G. B., Dunn. M. Programming Android with Kotlin. "O'Reilly Media, Inc.". 2021. 354 p.

19. Sharma A. Android app development using Kotlin programming language. In AIP Conference Proceedings. AIP Publishing. 2023. Vol. 2427, No. 1. URL: https://doi.org/10.1063/5.013072 (date of access: 18.04.2025).

20. Rodríguez G. S. Thriving in Android Development Using Kotlin. Use the newest features of the Android framework to develop production-grade apps. Packt Publishing Ltd. 2024. 410 p.