

Міністерство освіти і науки України
Державний заклад
«Луганський національний університет імені Тараса Шевченка»

Факультет технологій та інформаційних систем
Кафедра інформаційних технологій та систем

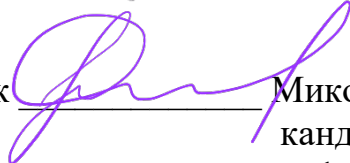
Шинкаренко Ярослав Михайлович

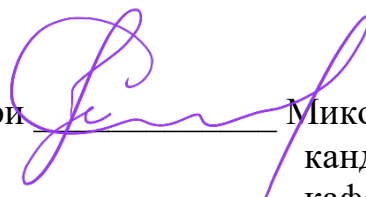
**СТВОРЕННЯ МОБІЛЬНОГО ЗАСТОСУНКУ ДЛЯ ВІДСТЕЖЕННЯ ТА
НАГАДУВАННЯ ПРО ПРИЙОМ ЛІКІВ**

кваліфікаційна робота

**здобувача вищої освіти першого (бакалаврського) рівня
освітньої програми «Інженерія програмного забезпечення»
за спеціальністю 121 Інженерія програмного забезпечення**

Особистий підпис  Ярослав ШИНКАРЕНКО

Науковий керівник  Микола СЕМЕНОВ,
кандидат педагогічних наук, доцент
кафедри інформаційних технологій
та систем

Завідувач кафедри  Микола СЕМЕНОВ,
кандидат педагогічних наук, доцент
кафедри інформаційних технологій
та систем

Міністерство освіти і науки України
Державний заклад „Луганський національний університет
імені Тараса Шевченка”

Інститут

Факультет технологій та інформаційних
систем

Кафедра, циклова комісія

Інформаційних технологій та систем

Рівень освіти

перший (бакалаврський)

Напрямок підготовки (спеціальність)

121 «Інженерія програмного забезпечення»

(код, назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри ІТС

М.А. Семенов

(підпис)

(ініціали, прізвище)

“ ”

2025 р.

**ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ**

Шинкаренку Ярославу Михайловичу

(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) Створення мобільного застосунку для відстеження та
нагадування про прийом ліків

Керівник кваліфікаційної роботи

Семенов М.А.

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджена наказом по університету

Від“ ” 2025 року №

2. Строк подання студентом проекту (роботи)

3. Вихідні дані до роботи (проекту) У результаті виконання роботи повинно

бути розроблено мобільний застосунок для відстеження та нагадування про
прийом ліків

(визначаються кількісні або (та) якісні показники, яким повинен відповідати об'єкт розробки)

**4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно
розробити)** Аналіз вимог до програмного забезпечення, процесів розробки програмного
забезпечення, якості та безпеки програмного забезпечення

(визначаються назви розділів або (та) перелік питань, які повинні увійти до тексту ПЗ)

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
ML-діаграма варіантів прецедентів, UML-діаграма класів шару представлення, домену та
доступу до даних, UML-діаграма послідовності, UML-діаграма кооперації, скріншоти інтерфейсу

6. Консультанти розділів проекту (роботи)

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

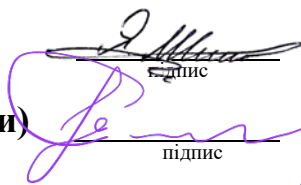
7. Дата видачі завдання „_____” _____ 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1.	Вибір теми роботи, вивчення наукової літератури, затвердження теми та керівника.	До 15 жовтня	
2.	Аналіз предметної області, дослідження ринкових аналогів mHealth. Формування специфікації вимог до ПЗ.	Другий тиждень листопада (10 листопада)	
3.	Проектування архітектури системи. Розробка UML-діаграм (прецедентів, класів, послідовності). Створення структури БД Room.	До 15 грудня	
4.	Практичне конструювання: реалізація шару доступу до даних (DAO, Repository), налаштування шифрування SQLCipher.	До 28 січня	
5.	Практичне конструювання: створення користувацького інтерфейсу (Jetpack Compose), реалізація логіки MVVM. Інтеграція AlarmManager.	Перший тиждень березня	
6.	Впровадження біометричної автентифікації. Проведення тестування ПЗ (модульне, інструментальне) та аналізу безпеки.	До 31 березня	
7.	Оформлення пояснювальної записки та розробка експлуатаційної документації (ТЗ, ПМ, КК, КА).	квітень	
8.	Попередній захист роботи на кафедрі. Усунення зауважень.	За 10 днів до державної атестації	
9.	Подання остаточного варіанта роботи на захист.	За 5 днів до державної атестації	

Студент

Керівник проекту (роботи)


підпис

Я.М.Шинкаренко
(ініціали, прізвище)

М.А.Семенов

АНОТАЦІЯ

Шинкаренко Я.М.

Тема: Розробка мобільного застосунку для відстеження та нагадування про прийом ліків.

Спеціальність: 121 «Інженерія програмного забезпечення»

Установа: ДЗ ЛНУ імені Тараса Шевченка, 2026 р.

Кваліфікаційна робота містить: 99 с., 14 рис., 22 додат., 53 джерел.

Об'єкт дослідження – процес інформатизації системи охорони здоров'я, зокрема впровадження мобільних технологій для управління індивідуальною фармакотерапією пацієнтів.

Предмет дослідження – архітектура, методи, алгоритми та програмні засоби розробки мобільного застосунку для платформи Android з використанням мови Kotlin та інструментарію Jetpack Compose.

Мета роботи – розробка надійного, безпечного та зручного мобільного застосунку для ОС Android, який допомагатиме користувачам ефективно керувати складними графіками прийому медикаментів, підвищуючи загальний рівень медичної адгерентності.

Результати роботи. У дипломній роботі досліджено предметну область та проаналізовано недоліки існуючих комерційних рішень. Спроектовано програмну архітектуру з використанням патерну Model-View-ViewModel та принципів Clean Architecture. Створено інтуїтивно зрозумілий, інклюзивний користувацький інтерфейс за допомогою декларативного фреймворку Jetpack Compose. Реалізовано локальну базу даних на основі бібліотеки Room із застосуванням криптографічного шифрування AES-256. Розроблено

автономну підсистему планування фонових завдань на базі AlarmManager, що адаптована до енергозберігаючих обмежень сучасних версій Android.

Висновки. В результаті розробки було створено повністю автономний кросфункціональний Android-застосунок медичного призначення. Продукт відповідає вимогам концепції «offline-first», гарантує безперебійну доставку сповіщень та забезпечує найвищий рівень захисту персональних медичних даних завдяки локальному шифруванню та обов'язковій біометричній автентифікації користувача.

Ключові слова: МОБІЛЬНИЙ ЗАСТОСУНОК, ANDROID, KOTLIN, JETPACK COMPOSE, ROOM, МЕДИЧНА АДГЕРЕНТНІСТЬ, ШИФРУВАННЯ ДАНИХ, ALARMMANAGER.

ABSTRACT

Shynkarenko Y.M.

Theme: Development of a mobile application for tracking and reminding about medication intake.

Specialty: 121 "Software Engineering"

Institution: Luhansk Taras Shevchenko National University (LTSNU), 2026.

Diploma work contains: 99 pages, 14 Fig., 22 adj., 53 sources.

Object of research is the process of informatization of the healthcare system, particularly the implementation of mobile technologies for managing individual patient pharmacotherapy.

Subject of research is the architecture, methods, algorithms, and software tools for developing an Android mobile application using Kotlin and Jetpack Compose.

Aim of work is the development of a reliable, secure, and user-friendly mobile application for Android to effectively help users manage complex medication schedules, thereby increasing the overall level of medical adherence.

Job performances. The thesis researched the subject area and analyzed the shortcomings of existing commercial solutions. The software architecture was designed using the Model-View-ViewModel pattern and Clean Architecture principles. An intuitive, inclusive user interface was created using the Jetpack Compose declarative framework. A local database was implemented based on the Room library, utilizing cryptographic encryption AES-256 algorithm. An autonomous background task scheduling subsystem was developed based on AlarmManager, adapted to the energy-saving restrictions of modern Android versions.

Conclusions. As a result of the development, a fully autonomous cross-functional medical Android application was created. The product complies with the "offline-first" concept, guarantees uninterrupted delivery of notifications, and ensures the highest level of protection for personal medical data through local encryption and mandatory user biometric authentication.

Keywords: MOBILE APPLICATION, ANDROID, KOTLIN, JETPACK COMPOSE, ROOM, MEDICAL ADHERENCE, DATA ENCRYPTION, ALARMMANAGER.

Міністерство освіти і науки України
Державний заклад «Луганський національний університет
імені Тараса Шевченка»

Факультет (інститут)

Факультет технологій та
інформаційних систем

(повна назва)

Кафедра

Інформаційних технологій та
систем

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання програмної розробки:

**«СТВОРЕННЯ МОБІЛЬНОГО ЗАСТОСУНКУ ДЛЯ
ВІДСТЕЖЕННЯ ТА НАГАДУВАННЯ ПРО ПРИЙОМ ЛІКІВ»**

ІТС.ІПЗ4.22seg61121-02-ТЗ

ПОГОДЖЕНО

Керівник кваліфікаційної
роботи

Семенов М.А.

«__» _____ 2026 р.

ВИКОНАВЕЦЬ

Студент групи 4 ІПЗ

Шинкаренко Я.М.

«__» _____ 2026 р.

Лубни - 2026

ЗМІСТ

1. ВСТУП.....	3
1. ХАРАКТЕРИСТИКА ОБ'ЄКТА	3
2. ПРИЗНАЧЕННЯ ТОВАРІВ	3
3. ОСНОВНІ ВИМОГИ ДО ПРОГРАМНОГО КОМПЛЕКСУ	4
4. ТЕХНІКО-ЕКОНОМІЧНІ ВИМОГИ ДО КІНЦЕВОГО ПРОДУКТУ.....	5
5. ВИМОГИ ДО МАТЕРІАЛІВ І КОМПЛЕКТУЮЧИХ	6
6. ЕТАПИ ВИКОНАННЯ ПР	6
7. ПРИЙОМ	7
8. ПОРЯДОК ВНЕСЕННЯ ЗМІН ДО ТЕХНІЧНОГО ЗАВДАННЯ, ЩО ЗАТВЕРДЖЕНО	8

1. ВСТУП.

1.1. Найменування розробки: Мобільний застосунок для відстеження та нагадування про прийом ліків (Робоча назва: "BeWell").

1.2. Шифр ПР: ІТС.ІП34.22seg61121

1.3. Підстава для розробки: необхідність створення безкоштовного, безпечного та автономного програмного продукту для вирішення проблеми низької медичної адгерентності (недотримання графіка прийому ліків пацієнтами).

1.4. Терміни розробки:

1.4.1. Початок: 15 жовтня 2025 р.

1.4.2. Закінчення: 10 травня 2026 р.

1.5. Фінансування: Розробка виконується в рамках навчального процесу без залучення комерційного фінансування.

1. ХАРАКТЕРИСТИКА ОБ'ЄКТА

1.1. Розроблювана програма повинна мати організаційно-медичний характер. До складу об'єкту, що створюється, повинно входити:

1.1.1. Графічний клієнтський додаток, що створюється.

1.1.2. Локальна зашифрована база даних.

1.1.3. Модуль фонових планувань нагадувань.

1.2. До вхідної інформації належать вимоги користувачів щодо розкладів прийому медикаментів, дозувань та частоти терапії.

2. ПРИЗНАЧЕННЯ ТОВАРІВ

2.1. Призначення: автоматизація контролю за медикаментозним лікуванням, збереження історії хвороби та генерація своєчасних сповіщень.

2.2. Основні критерії ефективності:

2.2.1. Зручний, інклюзивний та інтуїтивно зрозумілий інтерфейс, адаптований, зокрема, для літніх людей;

2.2.2. Надійна система «наполегливих» сповіщень, яка працює у фоновому режимі;

2.2.3. Гарантування конфіденційності медичних даних на пристрої користувача.

3. ОСНОВНІ ВИМОГИ ДО ПРОГРАМНОГО КОМПЛЕКСУ

3.1. Загальні вимоги:

3.1.1. Графічний додаток працює на смартфонах на базі ОС Android версії 7.0 (API Level 24) або вище;

3.1.2. Вимоги до апаратного забезпечення смартфона не є жорсткими, рекомендовано від 2 ГБ оперативної пам'яті, встановлюються розробником для забезпечення плавної роботи;

3.1.3. Мобільний додаток повинен мати сучасний, реактивний графічний інтерфейс.

3.2. Додаткові вимоги:

3.2.1. Мова програмування – Kotlin. Фреймворк для побудови інтерфейсу – Jetpack Compose. База даних – Room.

3.2.2. Вимоги до ліцензійного ПЗ: використовуються виключно безкоштовні технології та Open Source бібліотеки.

3.3. Вимоги до складу і архітектури:

3.3.1. Розробник самостійно обирає склад і виконує розробку архітектури ПР з обов'язковим дотриманням патерну MVVM та принципів Clean Architecture.

3.3.2. Система повинна мати offline-first архітектуру, що не потребує обов'язкового підключення до мережі Інтернет для виконання базових функцій.

3.4. Вимоги до якості і надійності:

3.4.1. Додаток повинен надійно працювати, обробляючи процеси життєвого циклу ОС Android без втрати запланованих нагадувань.

3.4.2. Розробник гарантує безпеку даних шляхом впровадження криптографічного шифрування локальної бази даних алгоритмом AES-256 через SQLCipher та біометричної автентифікації.

3.4.3. Розробник гарантує роботу графічного додатку без збоїв та втрати збереженого графіка прийомів ліків.

3.5. Вимоги до експлуатації:

3.5.1. Розробник використовує смартфон, на якому графічний додаток повинен надійно працювати та тестуватися.

4. ТЕХНІКО-ЕКОНОМІЧНІ ВИМОГИ ДО КІНЦЕВОГО ПРОДУКТУ

Розробка здійснюється без залучення комерційних API. Вартість експлуатації для кінцевого користувача зведена до нуля. Продукт розповсюджується за повністю безкоштовною моделлю, не містить вбудованої реклами та прихованих абонентських платежів, що робить його економічно доцільним та доступним для широких верств населення.

5. ВИМОГИ ДО МАТЕРІАЛІВ І КОМПЛЕКТУЮЧИХ

5.1. Вимоги до екологічної безпеки при експлуатації: Не пред'являються.

5.2. Спеціальні вимоги до кінцевого продукту: Не пред'являються.

5.3. Вимоги до безпеки для населення при експлуатації продукції: Не пред'являються в частині фізичної шкоди; щодо інформаційної безпеки – ризики мінімізовано за рахунок локального зберігання без передачі медичних даних на сторонні сервери.

6. ЕТАПИ ВИКОНАННЯ ПР

Етапи виконання ПР можуть уточнюватися згідно з календарним планом робіт за погодженням між керівником і виконавцем.

№	Етапи виконання роботи	Термін виконання і обсяг робіт	Звітні матеріали
1	Аналіз предметної області та розробка першої версії. Аналіз вимог. Розробка архітектурної структури БД Room. Попереднє тестування.	До кінця грудня 2025 р.	Фрагмент програмного комплексу, UML-моделі, структура БД, звітна документація.
2	Коригування структури. Розробка логіки MVVM та UI Jetpack Compose. Розробка остаточної версії програмного комплексу, інтеграція AlarmManager та SQLCipher. Тестування ПЗ.	До кінця березня 2026 р.	Готовий програмний комплекс - .apk файл на пристрої розробника і звітна документація.
3	Оформлення та підготовка експлуатаційної документації згідно з п.7 цього ТЗ.	До початку травня 2026 р.	Звітні матеріали згідно з пунктом 7 (повний комплект).

7. ПРИЙОМ

7.1. Необхідні вимоги для впровадження ПР і завершення робіт.

Оцінка результатів розробки і доцільність її прийняття здійснюється керівником за поданням наступних матеріалів:

- встановлено програмний комплекс на смартфоні або ЕОМ розробника;
- перелік вихідних файлів (репозиторій) на електронному носії;
- короткий опис роботи ПР і опис всіх модулів, які необхідні для роботи ПР;
- перелік документів:
 - Технічне завдання (ТЗ);
 - Пояснювальна записка (ПЗ);
 - Програма і методика тестування (ПМТ);
 - Керівництво користувача (КК);
 - Керівництво адміністратора/програміста (КА).

7.2. Перелік звітних документів, необхідних для прийняття етапів роботи:

- короткий опис результатів етапу у вигляді поточних звітів;
- частковий програмний комплекс на пристрої розробника згідно з календарним планом робіт;
- завершений та скопійований продукт. Звітні матеріали подаються у вигляді звітів на папері відповідно до встановлених стандартів.

7.3. Загальний перелік до прийому звітних документів, макетів, експериментальних зразків.

До прийому пред'являються: кваліфікаційна робота, супровідна документація та готовий пакет додатку .apk.

7.4. Тестування ПР.

Тестування виконується відповідно до "Програми та методики тестування", яка розробляється виконавцем і затверджується керівником.

8. ПОРЯДОК ВНЕСЕННЯ ЗМІН ДО ТЕХНІЧНОГО ЗАВДАННЯ, ЩО ЗАТВЕРДЖЕНО

Дане технічне завдання може уточнюватися в процесі розробки ПР при узгодженні сторін з оформленням відповідних доповнень або коригувань до ТЗ.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЗ «ЛУГАНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА»

Факультет технологій та інформаційних систем

(назва факультету, інституту)

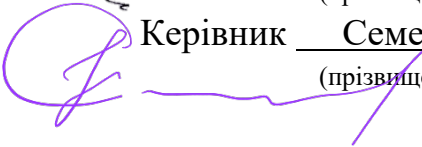
Інформаційних технологій та систем

(назва кафедри)

Пояснювальна записка
до дипломного проекту (роботи)
БАКАЛАВРА
(освітньо-кваліфікаційний рівень)
на тему: **СТВОРЕННЯ МОБІЛЬНОГО ЗАСТОСУНКУ ДЛЯ
ВІДСТЕЖЕННЯ ТА НАГАДУВАННЯ ПРО ПРИЙОМ ЛІКІВ**

Виконав: студент 4 курсу
напряму підготовки (спеціальності)
121 «Інженерія програмного забезпечення»
(шифр і назва напряму підготовки, спеціальності)

 Шинкаренко Я.М.
(прізвище та ініціали)

 Керівник Семенов М.А.
(прізвище та ініціали)

Лубни – 2026 року

ЗМІСТ

ЗМІСТ	2
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ	4
ВСТУП.....	5
РОЗДІЛ 1	7
АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	7
1.1. Огляд літератури за темою та актуальність проблеми.....	7
1.2. Аналіз відомих рішень та програмних продуктів.....	11
1.3. Аналіз вимог до програмного забезпечення та специфікація вимог	13
Висновки до розділу	16
РОЗДІЛ 2	17
АНАЛІЗ ПРОЦЕСІВ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ.....	17
2.1. Моделювання та аналіз програмного забезпечення.	17
2.2. Архітектура програмного забезпечення	25
2.3. Конструювання програмного забезпечення	30
2.4. Аналіз безпеки.....	34
Висновки до розділу	37
РОЗДІЛ 3	38
АНАЛІЗ ЯКОСТІ ТА БЕЗПЕКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	38
3.1. Аналіз якості ПЗ та опис процесів тестування.	38
3.2. Аналіз безпеки програмного забезпечення	41
3.4. Системні вимоги до ПЗ	48
Висновки до розділу	49
ЗАГАЛЬНІ ВИСНОВКИ.....	51
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	53
ДОДАТКИ.....	60
Додаток А Діаграма послідовності взаємодії компонентів системи	60
Додаток Б Лістинг HomeDashboardScreen.....	61
Додаток Б.1 Лістинг DateDivider	62
Додаток Б.2 Лістинг FilterSection	63
Додаток Б.3 Лістинг SwipeableMedicationItem	64
Додаток Б.4 Лістинг HomeViewModel.....	65

Додаток В Лістинг AddMedicationScreen	66
Додаток В.1 Лістинг InputTextField	67
Додаток В.2 Лістинг MedicationType	68
Додаток В.3 Лістинг CustomTimePicker	69
Додаток В.4 Лістинг AddMedicationViewModel	70
Додаток Г Лістинг HistoryScreen	71
Додаток Г.1 Лістинг HorizontalCalendar	72
Додаток Г.2 Лістинг TimelineItem	73
Додаток Г.3 Лістинг HistoryScreenViewModel	74
Додаток Г.4 Лістинг HistoryMedicationCard	75
Додаток Д Лістинг ReportsScreen	76
Додаток Д.2 Лістинг WeeklyTrendCard	78
Додаток Д.3 Лістинг ExportCard	79
Додаток Е Лістинг ProfileScreen	80
Додаток Е.1 Лістинг ProfileMenuItem	81
Додаток Е.2 Лістинг ProfileMenuSection	82
Додаток Ж Лістинг AppNavigation	83
Додаток З Лістинг BeWellBottomBar	84
Додаток К Лістинг AuthWrapper	85
Додаток Л Лістинг MedicationRepository	86
Додаток М Лістинг PillEntity	87
Додаток Н Лістинг IntakeRecordEntity	88
Додаток П Лістинг IntakeWithPill	89
Додаток Р Лістинг AlarmScheduler	90
Додаток С Лістинг AlarmSchedulerImpl	91
Додаток Т Лістинг PillDao	92
Додаток Ф Лістинг BeWellRoomDB	94
Додаток Х Лістинг EncryptionManager	95
Додаток Ц Лістинг MedicationAlarmReceiver	96
Додаток Ш Лістинг NotificationHelper	97
Додаток Щ Лістинг LockScreen	98

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

БД – база даних

ОС – операційна система

ПЗ – програмне забезпечення

СУБД – система управління базами даних

ADB – Android Debug Bridge

AES – Advanced Encryption Standard

API – Application Programming Interface

CIA – Confidentiality, Integrity, Availability

CRUD – Create, Read, Update, Delete

CSV – Comma-Separated Values

DAO – Data Access Object

DLP – Data Leakage Prevention

GCM – Galois/Counter Mode

GDPR – General Data Protection Regulation

JVM – Java Virtual Machine

MVVM – Model-View-ViewModel

ORM – Object-Relational Mapping

PDF – Portable Document Format

PIN – Personal Identification Number

SQL – Structured Query Language

TEE – Trusted Execution Environment

UDF – Unidirectional Data Flow

UI – User Interface

UML – Unified Modeling Language

URI – Uniform Resource Identifier

XML – eXtensible Markup Language

ВСТУП

Актуальність теми. У сучасному світі проблема недотримання режиму медикаментозного лікування є одним із найсерйозніших викликів для глобальної системи охорони здоров'я. За даними Всесвітньої організації охорони здоров'я, близько 50% пацієнтів із хронічними захворюваннями не дотримуються призначених схем лікування. Це призводить до зниження ефективності терапії, загострення хвороб, збільшення кількості госпіталізацій та значних економічних втрат. Особливо гостро ця проблема стоїть серед людей похилого віку, які часто мають поліпрагмазію та стикаються зі складними графіками прийому ліків.

Швидкий розвиток технологій мобільного здоров'я пропонує ефективні інструменти для вирішення цієї проблеми. Мобільні застосунки здатні автоматизувати процес контролю за лікуванням, надсилаючи своєчасні нагадування та зберігаючи історію хвороби. Проте, аналіз існуючих рішень виявляє низку суттєвих недоліків. У зв'язку з цим, розробка сучасної, безкоштовної, безпечної та повністю автономної мобільної екосистеми для платформи Android є вкрай актуальним завданням. Використання новітнього декларативного фреймворку Jetpack Compose дозволяє створити інтуїтивно зрозумілий та інклюзивний інтерфейс, а застосування мови Kotlin у поєднанні з архітектурою MVVM та Clean Architecture забезпечує високу продуктивність, масштабованість та надійність програмного продукту, враховуючи суворі обмеження операційної системи Android 14-15 щодо фонові роботи додатків.

Об'єкт дослідження – процес інформатизації системи охорони здоров'я, зокрема впровадження мобільних технологій для управління фармакотерапією та підтримки пацієнтів.

Предмет дослідження – архітектура, методи, алгоритми та програмні засоби розробки мобільного застосунку для відстеження та нагадування про прийом ліків на базі операційної системи Android з використанням мови Kotlin та інструментарію Jetpack Compose.

Метою кваліфікаційної роботи є проектування та розробка надійного, безпечного та зручного мобільного застосунку для платформи Android, який допомагатиме користувачам ефективно керувати графіками прийому медикаментів, підвищуючи рівень медичної адгерентності та захищаючи їхні персональні дані.

Для досягнення поставленої мети було сформульовано та вирішено такі завдання:

1. Провести аналіз предметної області, дослідити існуючі аналоги на ринку mHealth та визначити їхні недоліки.
2. Сформулювати функціональні та нефункціональні вимоги до програмного забезпечення, з акцентом на конфіденційність та інклюзивність.
3. Спроектувати архітектуру мобільного додатку, застосовуючи патерн MVVM та принципи Clean Architecture для забезпечення незалежності бізнес-логіки.
4. Розробити локальну базу даних для зберігання медичних записів за допомогою бібліотеки Room та реалізувати механізми надійного планування фонових завдань з використанням WorkManager та AlarmManager.
5. Створити сучасний, реактивний користувацький інтерфейс за допомогою Jetpack Compose.
6. Провести тестування якості та аналіз безпеки розробленого програмного забезпечення, включаючи впровадження біометричної автентифікації.

Розроблений продукт може застосовуватися пацієнтами для повсякденного контролю за прийомом ліків, ведення журналу дозувань та контролю запасів медикаментів. Офлайн-архітектура робить застосунок незалежним від інтернет-з'єднання та гарантує повну конфіденційність медичних даних користувача на його пристрої. Крім того, застосовані архітектурні рішення роблять програмний код відкритим для подальшого масштабування, наприклад, для інтеграції з державними реєстрами ліків або електронною системою охорони здоров'я у майбутньому.

РОЗДІЛ 1

АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1. Огляд літератури за темою та актуальність проблеми

Проблема недотримання режиму лікування, яка в медичній спільноті часто становить одну з найбільш значущих перешкод на шляху до покращення глобальних показників здоров'я. За статистичними даними, близько 50% пацієнтів у всьому світі не приймають ліки згідно з призначеннями своїх лікарів. Це призводить до каскаду негативних наслідків, включаючи загострення хронічних станів, збільшення кількості госпіталізацій, підвищення рівня смертності та значні економічні втрати для систем охорони здоров'я [10].

Згідно з даними Управління з продовольства і медикаментів США (FDA), щороку фіксується понад 300 000 медикаментозних помилок, які безпосередньо впливають на близько 1,5 мільйона пацієнтів [10]. Помилки при адмініструванні ліків становлять від 25 до 40 відсотків усіх медикаментозних помилок. Головними причинами таких помилок є неправильна інтерпретація медичних інструкцій пацієнтами, їхня забудькуватість та нездатність дотримуватися жорстких розкладів [10].

Особливо гостро ця проблема стоїть для літніх людей та пацієнтів із хронічними захворюваннями наприклад, серцевою недостатністю, діабетом, гіпертонією. Для таких пацієнтів управління ліками є ключовим елементом догляду, проте вони часто стикаються з явищем поліпрагмазії – одночасним прийомом великої кількості препаратів. Деякі пацієнти змушені приймати в середньому від 10 до 25 таблеток щодня [11]. За даними Центрів з контролю та профілактики захворювань у США (CDC), понад третина 34,5% дорослих віком 60-70 років приймають п'ять або більше рецептурних препаратів [11]. Поліпрагмазія створює високі ризики небезпечних лікарських взаємодій, передозувань та побічних ефектів, що робить ручне відстеження прийому ліків практично неможливим.

У відповідь на ці виклики сегмент мобільного здоров'я, mHealth, зазнав значної трансформації від простих календарних нагадувань до складних інтелектуальних платформ [10]. Технології mHealth розглядаються дослідниками як ефективний інструмент для подолання проблеми низької прихильності [36].

Аналіз сучасної наукової літератури дозволяє констатувати, що впровадження мобільних рішень у сферу охорони здоров'я відкриває широкі можливості для мінімізації критичних помилок, пов'язаних із часовим фактором прийому препаратів. Завдяки механізмам персоналізованих сповіщень суттєво знижується ризик пропуску терапевтичних доз або, навпаки, їхнього передчасного вживання, що є фундаментом успішного лікування. Разом з тим, функціональний потенціал сучасних систем не обмежується лише нагадуваннями. Спеціалізовані електронні алгоритми здатні здійснювати комплексний аналіз списків медикаментів, запобігаючи терапевтичному дублюванню та виявляючи потенційно небезпечні міжлікарські взаємодії ще на етапі планування курсу.

Окрім суто технічних аспектів контролю, мобільні застосунки відіграють ключову роль у трансформації комунікаційних процесів між пацієнтом та фахівцем. Можливість оперативно передавати верифіковану історію прийомів медичним працівникам або довіреним опікунам значно підвищує точність діагностики та прискорює процес клінічного консультування (рис.1.1.).

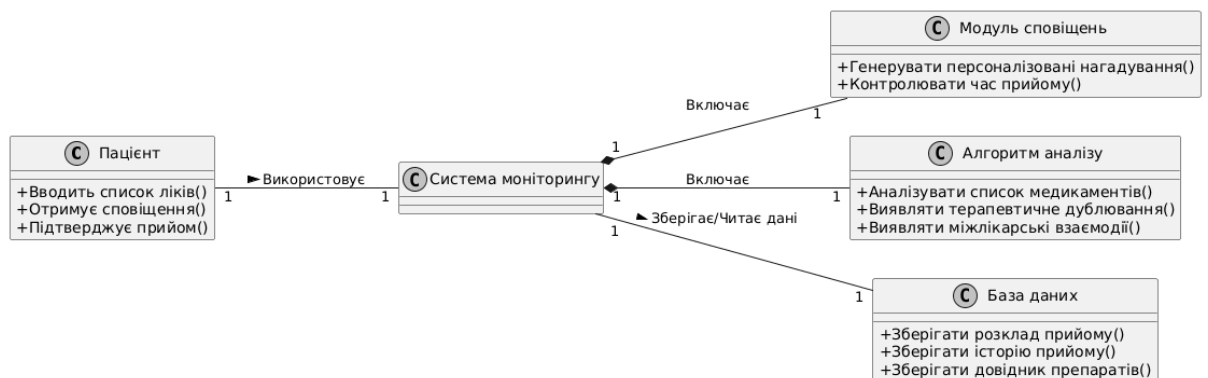


Рис. 1.1. UML діаграма схеми взаємодії компонентів системи моніторингу медикаментів

Таким чином, інтеграція подібних інструментів стає необхідною умовою для формування якісного цифрового середовища підтримки здоров'я. На основі проведеного аналізу переваг мобільної терапії, далі ми детально розглянемо архітектурні особливості проєктування.

У дослідженнях, що застосовують валідовану шкалу оцінки мобільних застосунків, зазначається, що хоча об'єктивна якість багатьох застосунків є прийнятною, їх суб'єктивна якість часто залишається низькою [11]. Значна частина програмного забезпечення не є адаптованою для цільової аудиторії, застосунки часто перевантажені складними інтерфейсами, мають дрібні шрифти, не пояснюють, як вводити дані про медикаменти, та не надають інструкцій щодо правил прийому ліків [10].

Так у наукових працях, присвячених архітектурі планування медичних прийомів, розглядаються різні алгоритмічні моделі для управління препаратами, що взаємодіють між собою.

OMAT (One-Medication-at-a-Time) – алгоритм генерує повний розклад для кожного препарату окремо. Вважається складнішим і зазвичай створюються для розкладів в офлайн-режимі [51]. Рис 1.2.

ODAT (One-Dose-at-a-Time) – планує лише одну наступну дозу, не маючи попередніх знань про майбутні. Він гірше будує ідеальні графіки, але набагато краще адаптується до динамічних змін у поведінці користувача [51]. Рис 1.2.

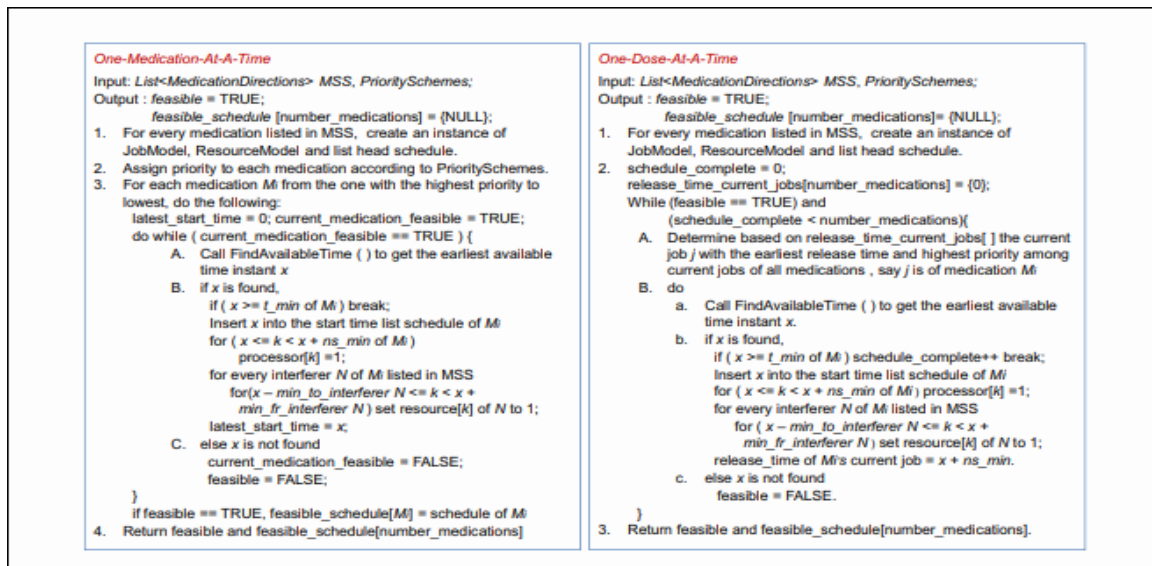


Рис. 1.2. Алгоритми OMAT і ODAT

З точки зору архітектури бази даних для таких розкладів, дослідники акцентують на необхідності впровадження концепції «часової валідності» та стабільних порядкових номерів. Це дозволяє гарантувати історичну точність даних: якщо пацієнт змінює час прийому препарату, система повинна зберегти попередню статистику без руйнування логіки минулих прийомів, що є критично важливим для медичних звітів [29].

Ще однією вагомою проблемою, яка формує актуальність розробки нових застосунків, є агресивна монетизація та низький рівень конфіденційності в існуючих продуктах. Дослідження, опубліковане в *Journal of the American Medical Informatics Association*, виявило, що більшість топових медичних додатків на платформі Android не мають належного захисту приватності, а понад половина з них відкрито заявляють у своїх політиках про передачу медичних даних третім особам. Оскільки більшість мобільних застосунків не підпадають під дію жорстких законів типу HIPAA у США, особисті дані пацієнтів часто залишаються незахищеними [22].

Ринок також страждає від проблеми приховування критичних функцій за оплатою. Наприклад, один з найвідоміших застосунків Medisafe з початку 2026 року обмежив безкоштовне використання лише двома препаратами. Для пацієнтів із поліпрагмазією, які приймають п'ять або більше медикаментів, це робить базову безкоштовну версію абсолютно марною, змушуючи їх купувати платну підписку. Відсутність по-справжньому безкоштовних інструментів, здатних підтримувати складні терапевтичні курси без обмежень на кількість ліків та без нав'язливої реклами, є значною прогалиною на ринку [33].

Узагальнюючи огляд літератури та аналіз сучасного стану проблеми, можна зробити висновок, що попри значну кількість існуючих рішень для відстеження прийому ліків, ринок гостро потребує систем, які б гармонійно поєднували в собі клінічно орієнтований функціонал, інклюзивний та доступний дизайн UI/UX, адаптований для літніх людей з урахуванням їхніх моторних та когнітивних особливостей, безкомпромісну безпеку даних і доступність критичних функцій.

Створення нової мобільної екосистеми на базі сучасних інструментів, таких як мова програмування Kotlin та декларативний фреймворк Jetpack Compose, дозволить реалізувати надійний, реактивний інтерфейс та забезпечити гарантовану доставку сповіщень навіть у фоновому режимі, що робить розробку такого програмного продукту високо актуальним та науково обґрунтованим завданням.

1.2. Аналіз відомих рішень та програмних продуктів

Ринок мобільних медичних додатків пропонує широкий спектр інструментів для контролю прийому ліків. Для розуміння поточних тенденцій, переваг та недоліків існуючих систем, було проведено детальний аналіз трьох популярних програмних продуктів – Medisafe, MyTherapy та Pillo.

Medisafe є одним із найвідоміших та найпопулярніших додатків на ринку, який посідає лідируючі позиції за об'єктивною шкалою якості мобільних додатків та має клінічне підтвердження ефективності у рандомізованих контрольованих дослідженнях [50]. Головною перевагою системи є використання пропрієтарної технології Just-In-Time-Intervention (JITI), яка за допомогою штучного інтелекту адаптує час та тип сповіщень під індивідуальні звички користувача для досягнення найкращих результатів [50]. Додаток також підтримує функцію залучення опікунів або членів сім'ї, попереджає про лікарські взаємодії, відстежує понад 20 показників здоров'я та глибоко інтегрується зі смарт-годинниками, наприклад, Apple Watch [50].

Однак, суттєвим недоліком Medisafe є його агресивна модель монетизації. З 1 січня 2026 року розробники перевели більшість функцій у платну підписку у вигляді \$4.99 на місяць, жорстко обмеживши безкоштовну версію відстеженням лише двох медикаментів. Для літніх пацієнтів та осіб із поліпрагмазією, таке обмеження робить базову безкоштовну версію додатку фактично непридатною для використання [33]. Крім того, розширені налаштування звуків сповіщень та візуальних тем також були приховані.

MyTherapy – популярний у Європі безкоштовний додаток, що поєднує функції нагадування про прийом ліків та комплексного щоденника здоров'я [50].

Додаток дозволяє фіксувати життєві показники - артеріальний тиск, рівень цукру в крові, вагу, відстежувати симптоми та настрій. Завдяки використанню предиктивної аналітики, MyTherapy здатний виявляти патерни недотримання режиму лікування та попереджати про це опікунів [50]. Також він пропонує зручну функцію генерації звітів у форматі PDF для лікарів та має спеціальний веб-дашборд для медичних працівників.

Попри відсутність жорстких обмежень на кількість препаратів, MyTherapy має певні UI/UX та функціональні недоліки. Наприклад, додаток підтримує «наполегливі» будильники, покладаючись на стандартні push-сповіщення, які легко випадково змахнути з екрана або пропустити під час міцного сну. Крім того, розробники інтегрували в додаток Шекламні модулі та здійснюють обмін даними з рекламодавцями; для відключення реклами користувачам необхідно сплатити відповідну комісію [22].

Pillo позиціонується на ринку як потужна безкоштовна альтернатива додаткам із жорсткими фінансовими обмеженнями [3]. Головною відмінністю Pillo від конкурентів є наявність системи «наполегливих» будильників, які продовжують дзвонити доти, доки користувач свідомо не підтвердить або не відкладе прийом ліків. Ця функція є критично важливою для пацієнтів, схильних до забудькуватості, та осіб із глибоким сном.

Pillo пропонує необмежену кількість медикаментів для відстеження без необхідності оформлення підписки [3], підтримує складні гнучкі розклади та інтегрує трекери показників здоров'я безпосередньо у єдиний інтерфейс. Додаток також містить функцію управління ліками для залежних осіб, що дозволяє дистанційно контролювати та налаштовувати графік прийому ліків для літніх батьків зі свого смартфона. Незважаючи на те, що Pillo є повністю безкоштовним для основного функціоналу, він містить рекламу, що для деяких користувачів може бути відволікаючим фактором.

Проведений огляд показує, що хоча на ринку існують потужні рішення з розвиненим клінічним функціоналом, вони все частіше вдаються до обмеження базових функцій або компрометують користувацький досвід рекламою та

передачею даних третім особам. Аналоги без обмежень на кількість ліків, такі як Pillo, вирішують значну частину цих проблем завдяки наполегливим сповіщенням, проте наявність реклами все ще залишається компромісом. Це підтверджує необхідність розробки власної системи, яка б гармонійно поєднувала гарантовані механізми сповіщень, підтримку складних розкладів, повну конфіденційність та відсутність штучних обмежень на кількість препаратів.

1.3. Аналіз вимог до програмного забезпечення та специфікація вимог

На основі проведеного аналізу предметної області та існуючих програмних рішень було сформовано детальну специфікацію вимог до мобільного застосунку для відстеження та нагадування про прийом ліків. Вимоги поділяються на функціональні, вимоги до безпеки та продуктивності, вимоги до інтерфейсу користувача та архітектурні вимоги.

Система повинна підтримувати різні сценарії прийому медикаментів. Сюди входять щоденні нагадування, інтервальні графіки[51]. Крім того, користувач повинен мати змогу відкладати нагадування або додавати позапланові дози.

Оскільки медичні сповіщення є критично важливими, застосунок має використовувати надійні системні механізми планування, наприклад, AlarmManager в ОС Android, які здатні виводити пристрій із режиму глибокого сну для точного спрацьовування будильників [49]. Також необхідна наявність «наполегливих» сигналів, які вимагають свідомої дії користувача для їх вимкнення.

Програма повинна зберігати точну історію минулих прийомів ліків та використання стабільних порядкових номерів доз. Це гарантує, що зміна поточного налаштування розкладу пацієнтом не призведе до руйнування чи втрати минулої статистики [29].

Програмний продукт повинен відповідати міжнародним стандартам захисту медичних даних, що вимагає впровадження політик мінімізації збору даних та згоди користувача на обробку інформації [22].

Локальна база даних повинна бути зашифрована. Для цього слід використовувати інструменти рівня SQLCipher AES-256, щоб унеможливити доступ до даних у разі фізичного компрометування пристрою. Ключі шифрування мають генеруватися випадковим чином і надійно зберігатися в апаратному сховищі [21]. Для забезпечення додаткового рівня захисту доступ до чутливої медичної інформації в застосунку має бути захищений за допомогою біометричної автентифікації пристрою [43].

Враховуючи, що цільовою аудиторією часто є літні люди з можливими когнітивними та моторними обмеженнями, інтерфейс має розроблятися за принципами інклюзивного дизайну [10], [54]. А саме використання великих, читабельних шрифтів, не менше 16-18 пунктів та забезпечення високої контрастності елементів, не менше 4.5:1 відповідно до стандарту WCAG 2.2 AA [10], [54]. Інтерактивні кнопки повинні мати розмір щонайменше 44x44 пікселі з достатнім вільним простором навколо, щоб уникнути випадкових натискань [10], [54]. Використовувати комбінації візуальних, звукових сигналів та вібрації для гарантованого привернення уваги користувача під час нагадувань [10], [54].

Розробка мобільного клієнта під платформу Android має базуватися на сучасній мові програмування Kotlin. Для розробки декларативного та реактивного інтерфейсу користувача повинен використовуватися фреймворк Jetpack Compose [6].

Система має бути побудована за принципами Clean Architecture у поєднанні з патерном Model-View-ViewModel (MVVM) , що забезпечує чіткий розподіл на шари представлення, домену та даних, полегшуючи тестування та подальшу підтримку [5], [31]. Рис. 1.3.1.

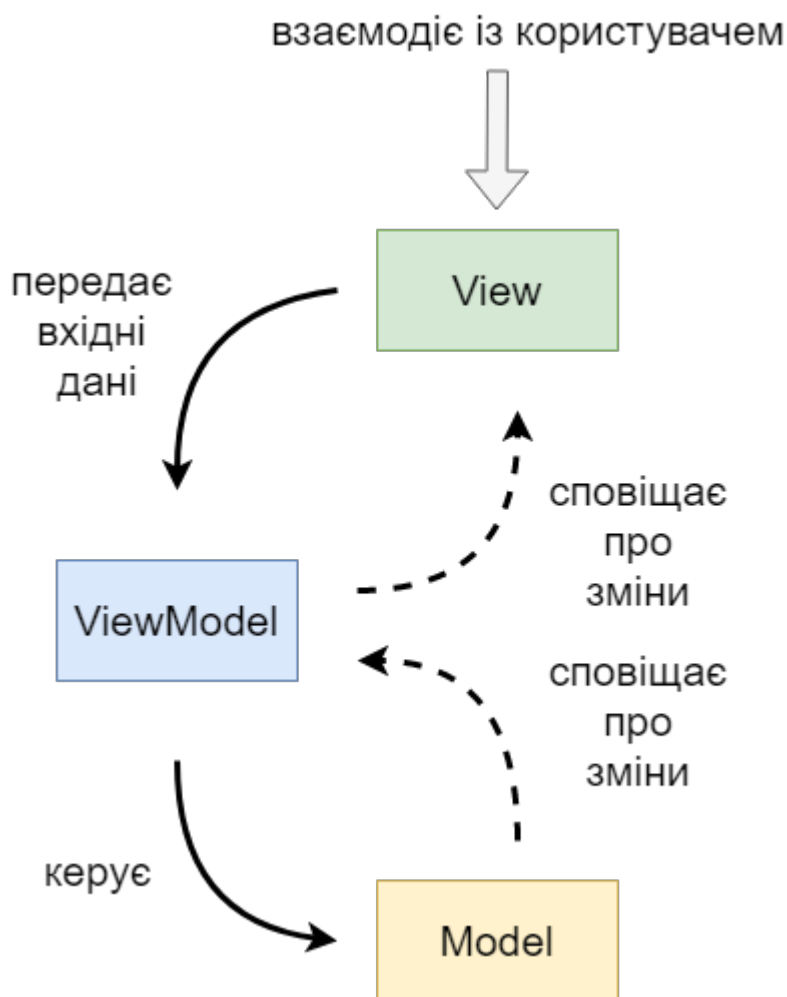


Рис. 1.3.1 Model-View-ViewModel – шаблон проектування

Застосовувати асинхронне програмування на базі Kotlin Coroutines, механізми впровадження залежностей за допомогою Hilt, та локальне зберігання даних з використанням абстракції бази даних Room [41].

Отже, на основі проведеного аналізу сформовано комплексну специфікацію вимог, яка є надійним підґрунтям для подальшої розробки мобільного застосунку. Визначені функціональні вимоги гарантують точність та гнучкість планування прийому ліків у поєднанні з безвідмовною системою сповіщень та контролем взаємодії препаратів. Нефункціональні вимоги забезпечують найвищий рівень безпеки чутливих медичних даних відповідно до міжнародних стандартів HIPAA, GDPR, використовуючи надійні методи шифрування та біометричну автентифікацію.

Окрема увага до UI/UX дизайну гарантує інклюзивність та доступність інтерфейсу для головної цільової аудиторії - пацієнтів літнього віку. Зрештою, вибір сучасного технологічного стеку Kotlin, Jetpack Compose у поєднанні з передовими підходами Clean Architecture та MVVM дозволить створити стабільний, масштабований та зручний у підтримці медичний програмний продукт, здатний ефективно вирішувати проблему низької адгерентності.

Висновки до розділу

У першому розділі було проведено комплексний аналіз предметної області, який підтвердив високу актуальність проблеми низької медичної адгерентності. Дослідження показало, що недотримання пацієнтами, особливо літнього віку, режиму прийому ліків призводить до серйозних медичних ускладнень та підвищує навантаження на систему охорони здоров'я. Огляд сучасного ринку систем мобільного здоров'я та існуючих програмних рішень виявив низку критичних недоліків у популярних застосунках. Більшість із них є надмірно комерціалізованими, обмежують життєво необхідний функціонал у безкоштовних версіях, перевантажені рекламою та зберігають чутливі медичні дані пацієнтів на сторонніх серверах, що створює потенційні ризики порушення конфіденційності.

На основі виявлених ринкових прогалин та потреб користувачів було сформовано детальну специфікацію функціональних і нефункціональних вимог до розроблюваної мобільної екосистеми. Визначено, що новий застосунок має бути безкоштовним, працювати за архітектурним принципом offline-first для гарантування абсолютної безпеки локальних даних, а також підтримувати гнучке налаштування складних терапевтичних графіків.

Окрему увагу було приділено аналізу цільової аудиторії, значну частину якої складають літні люди. Це зумовило необхідність впровадження жорстких принципів інклюзивного UI/UX дизайну. Сформований у цьому розділі комплекс вимог та обмежень утворив надійний концептуальний фундамент для переходу до етапу проектування архітектури, вибору інструментарію та конструювання застосунку.

РОЗДІЛ 2

АНАЛІЗ ПРОЦЕСІВ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ.

2.1. Моделювання та аналіз програмного забезпечення.

Для моделювання усіх можливих сценаріїв взаємодії між кінцевим користувачем та самою комп'ютерною системою на етапі проєктування традиційно використовується діаграма прецедентів. Рис.2.1. Цей інструмент дозволяє абстрагуватися від внутрішньої технічної реалізації та зосередитися на тому, що саме робить система у відповідь на дії зовнішніх суб'єктів, яких прийнято називати акторами. У контексті розроблюваного додатку для контролю прийому медикаментів виділяються два ключові актори - користувач та модуль фонових завдань та сповіщень.

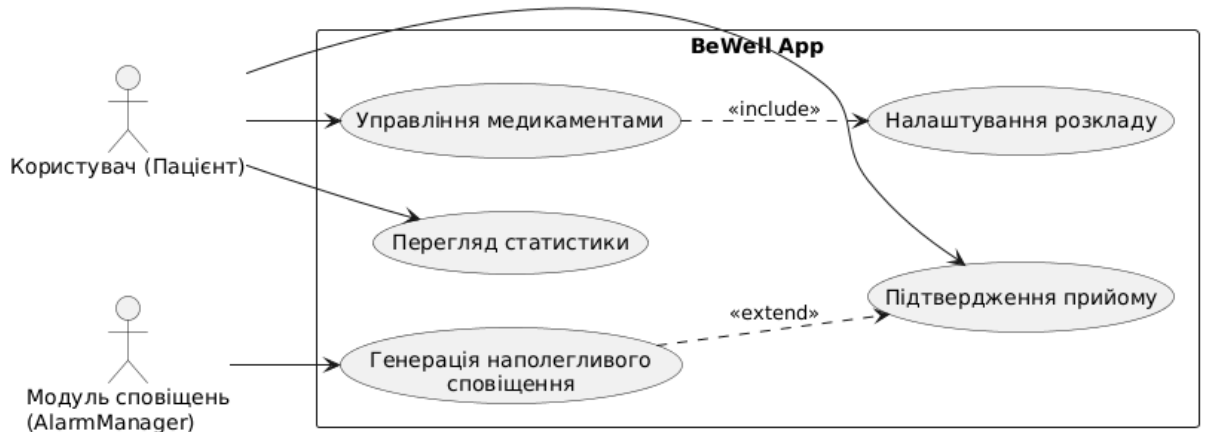


Рис. 2.1. UML діаграма варіантів прецедентів

Головним ініціатором більшості процесів є користувач. Його взаємодія із системою починається з базового, але найважливішого сценарію – управління медикаментами. Це комплексний прецедент, який охоплює не лише просте введення назви ліків, але й глибоке налаштування терапевтичного курсу. Під час додавання нового запису користувач визначає форму випуску препарату, наприклад: таблетки, сироп чи ін'єкції. Також задає суворі параметри розкладу, частоту прийомів та точний час. Цей етап є критичним, оскільки саме на основі введених тут даних формується вся подальша бізнес-логіка додатка.

Наступний, найбільш частотний сценарій взаємодії – це управління щоденним графіком лікування. На головному екрані системи користувач взаємодіє зі сформованим переліком завдань на поточний день. Тут реалізується прецедент підтвердження прийому, людина свідомо фіксує факт вживання ліків, позначає їх як вжиті або ж видаляє за потреби. Усі ці дії система ретельно протоколює та зберігає у захищеній локальній базі даних у відповідності до вимог та завдань визначених у розділі 1.3. .

Збережена історія взаємодій формує фундамент для наступного прецеденту – аналітичного. Користувач отримує змогу переглядати ретроспективну статистику та аналізувати графіки дотримання режиму лікування за різні проміжки часу. Більше того, цей прецедент розширюється можливістю експорту зібраних медичних даних у зручні формати, PDF або CSV, що дозволяє легко поділитися цією інформацією зі своїм лікуючим лікарем для коригування терапії.

Другим, невидимим для ока, але технічно необхідним актором виступає сам програмний комплекс – підсистема планування. На відміну від користувача, цей актор діє проактивно та повністю автономно у фоновому режимі. Спираючись на дані, збережені в локальній базі Room, планувальник безперервно відстежує настання запланованих подій. У визначений розкладом момент він ініціює прецедент генерації системного сповіщення. Важливо зазначити, що це не просто пасивне інформування, система генерує наполегливі нагадування, які вимагають від користувача обов'язкової зворотної реакції – підтвердження прийому або відкладення нагадування. Це замикає цикл взаємодії та безпосередньо виконує головну мету додатка визначену в розділі 1.3., а саме підвищення дисципліни медикаментозної терапії.

Функціональна модель системи базується на тісній та безперервній співпраці двох сутностей, ініціативного користувача, який формує правила та аналізує результати і точної автоматизованої системи, яка гарантує безперебійне нагадування про ці правила у реальному часі.

Для формалізованого опису статичної структури розроблюваного програмного забезпечення та візуалізації внутрішніх ієрархічних зв'язків між його ключовими компонентами застосовується об'єктно-орієнтований підхід із використанням UML-діаграм класів. Оскільки мобільний Додаток Цроєктується для операційної системи Android, його фундаментальна архітектура базується на сучасному патерні MVVM, який тісно переплітається з принципами чистої архітектури. Такий симбіоз архітектурних рішень є критично важливим для медичних додатків, адже він гарантує низьку зв'язність між окремими модулями, забезпечує суворий розподіл відповідальності та створює надійну основу для подальшого масштабування й автоматизованого тестування коду. Зважаючи на значну кількість взаємопов'язаних сутностей у системі, для забезпечення високої читабельності та уникнення когнітивного перевантаження загальної моделі, структуру було декомпозиовано на три логічні складові. Кожна з них детально розкриває специфіку окремого архітектурного шару: шару інтерфейсу користувача, шару бізнес-логіки та роботи з даними, а також автономної підсистеми фонових сервісів.

Першим і найбільш наближеним до користувача є шар представлення, який відповідає за рендеринг графічного інтерфейсу та первинну обробку подій взаємодії, що детально відображено на рисунку 2.1.1. Візуальна частина даної системи повністю реалізована за допомогою сучасного декларативного фреймворку Jetpack Compose. Структурно цей шар об'єднаний класом AppNavigation, який містить контролер навігації NavHostController і виступає єдиним центром управління переходами між незалежними функціями-екранами. Кожен ключовий екран системи має власний виділений контролер стану – відповідний клас ViewModel. Ці класи відіграють роль інтелектуальних посередників, які ізолюють користувацький інтерфейс від складної бізнес-логіки. Вони зберігають свій стан незалежно від життєвого циклу активності, інкапсулюють логіку валідації користувацького вводу та використовують реактивні потоки даних для безпечної та асинхронної передачі оновленої інформації на екран. Усі моделі представлення спроектовані з дотриманням

принципу інверсії залежностей: вони цілком абстраговані від фізичних джерел даних і спілкуються з ними виключно через інтерфейси репозиторію.

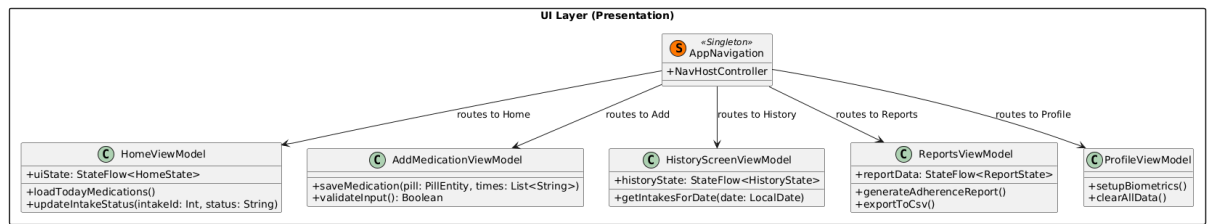


Рис. 2.1.1. Діаграма класів шару представлення

Наступним фундаментальним рівнем системи є шар бізнес-логіки та доступу до даних, статична структура якого представлена на рисунку 2.1.2. Ядром цієї підсистеми виступає клас `MedicationRepository`. Репозиторій агрегує запити від різних моделей представлення і самостійно вирішує алгоритм їх обробки, скеровуючи виклики до локальної реляційної бази даних. Сама база даних побудована на основі ORM-бібліотеки `Room`. Фізична структура збереження даних описується через дата-класи, сутностями таблиць. Сутність `PillEntity` інкапсулює незмінні характеристики медикаменту, `IntakeRecordEntity` фіксує динамічні параметри кожного прийому ліків. Для формування реляційних зв'язків між цими таблицями застосовується допоміжний клас `IntakeWithPill`. Безпосередня маніпуляція цими сутностями здійснюється через спеціалізовані об'єкти доступу до даних `PillDao` та `IntakeRecordDao`, які містять необхідні SQL-запити та повертають реактивні потоки `Flow` для миттєвого оновлення інтерфейсу. Критично важливим компонентом цього шару є `EncryptionManager`, який відповідає за криптографічні перетворення. Цей менеджер інтегровано безпосередньо в процес запису та читання, що гарантує надійне шифрування локальних даних та захист конфіденційності користувача у випадку несанкціонованого фізичного доступу до пристрою.

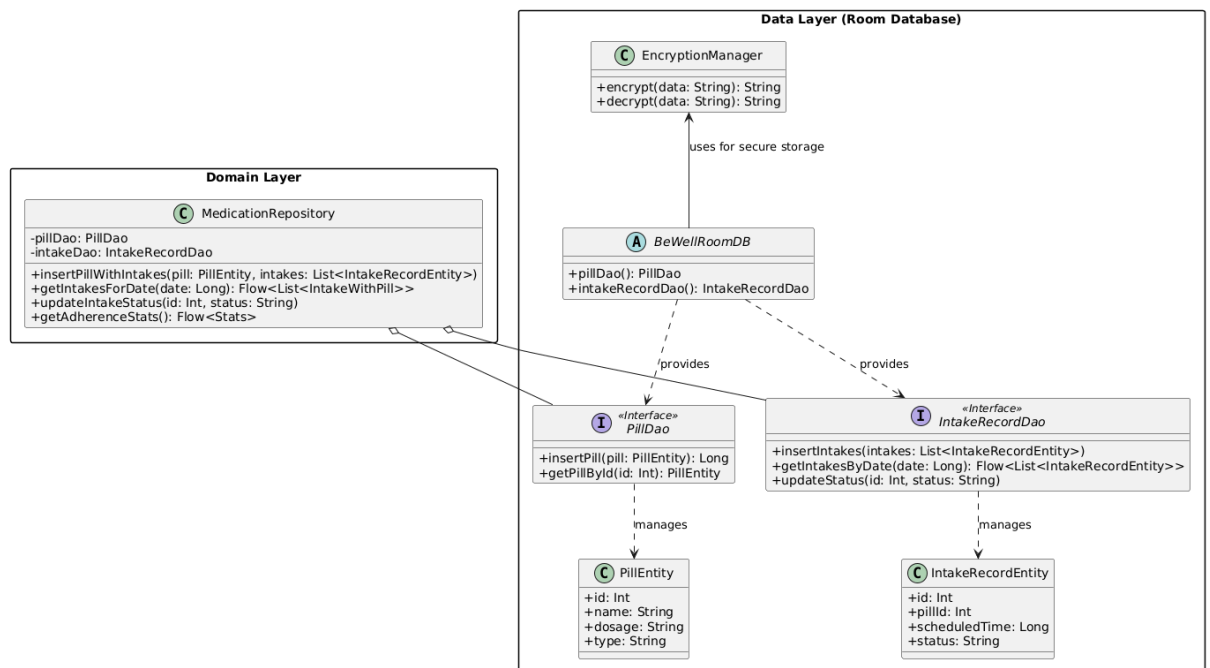


Рис. 2.1.2. Діаграма класів шару доступу до даних

Третя складова структурної моделі, візуалізована на рисунку 2.1.3., описує архітектуру підсистеми фонових завдань та сповіщень. Це модуль, який забезпечує своєчасне нагадування про прийом ліків незалежно від поточного стану активності додатка. Основою цієї підсистеми є інтерфейс `AlarmScheduler` та його практична реалізація `AlarmSchedulerImpl`. Даний клас безпосередньо взаємодіє з системним сервісом Android – `AlarmManager` для реєстрації точних подій у часі. При настанні запланованого часу операційна система ініціює пробудження спеціального компонента `MedicationAlarmReceiver`, успадкованого від базового класу `BroadcastReceiver`. Отримавши сигнал, цей приймач зчитує з наміру ідентифікатор відповідного медикаменту та делегує формування інтерфейсу допоміжному класу `NotificationHelper`. Останній створює захищені канали сповіщень згідно із сучасними вимогами безпеки ОС Android, конструює візуальне інтерактивне push-сповіщення та додає до нього кнопки швидкого реагування, виводячи на екран пристрою вимогу до користувача щодо свідомого підтвердження прийому.

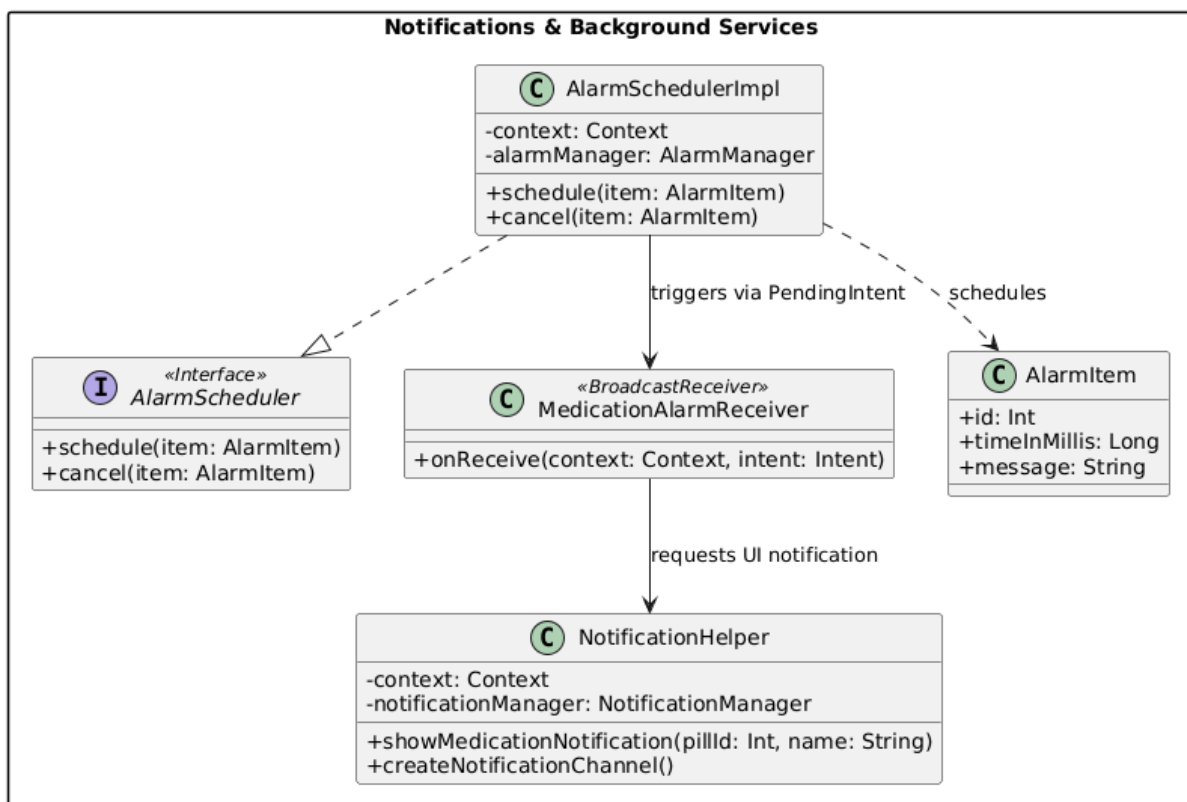


Рис 2.1.3. Діаграма класів підсистеми фонових сповіщень

Усі вище описані підсистеми утворюють єдиний, безперервний цикл роботи продукту. Життєвий цикл типового сценарію розпочинається із введення нового медикаменту. Ці дані захоплюються класом `AddMedicationViewModel`, проходять первинну валідацію, трансформуються в об'єкти та передаються до `MedicationRepository`. Репозиторій, використовуючи об'єкти DAO, асинхронно зберігає зашифровану інформацію в базу даних `Room`. Одразу після успіху, система звертається до `AlarmScheduler` для встановлення системного таймера на основі розрахованого часу наступного прийому. Згодом генерується системне сповіщення. Коли користувач взаємодіє з ним, оновлює статус відповідного запису `IntakeRecordEntity` на «Прийнято» через репозиторій. Завдяки тому, що база даних `Room` працює з реактивними потоками, це оновлення миттєво отримується всіма активними моделями, змушуючи відповідні екрани перемалювати свій стан.

Для розуміння поведінки розроблюваної системи недостатньо лише визначити її статичну структуру, важливим є дослідження обміну

повідомленнями між об'єктами. З цією метою використовується діаграма послідовності. У розроблюваного додатку для контролю прийому медикаментів найбільш репрезентативним є сценарій, що охоплює повний цикл роботи програми: від ініціалізації розкладу користувачем до автономного спрацьовування фоновому нагадування та реактивного оновлення графічного інтерфейсу. Візуалізація цього процесу подана в додатку А.

З додатку А можна зрозуміти що, процес взаємодії розпочинається на рівні шару представлення, коли користувач вводить назву препарату, обирає його форму, дозування та точний час прийомів. Цей масив даних передається до контролера екрану – класу `ViewModel`. Останній виконує валідацію введених параметрів, формує з них об'єкти та ініціює асинхронний виклик, звертаючись до центрального репозиторію. Важливо зазначити, що на цьому етапі управління потоком виконання передається у фоновий потік, щоб уникнути блокування головного потоку інтерфейсу користувача.

Отримавши управління, репозиторій починає процес збереження (додаток А). Спочатку він звертається до об'єктів доступу до даних, передаючи їм сутності ліків та пов'язані з ними записи про заплановані прийоми. Запити проходять через менеджер шифрування та синхронно зберігаються у локальній базі даних `Room`. Одразу після успішного підтвердження транзакції від бази даних, репозиторій викликає планувальник, передаючи йому ідентифікатори записів та розрахований час. Планувальник, своєю чергою, здійснює системний виклик до API операційної системи `Android AlarmManager`, реєструючи подію пробудження з точністю до хвилини. На цьому етап ініціалізації завершується, `ViewModel` отримує повідомлення про успіх і оновлює стан інтерфейсу, повертаючи користувача на головний екран.

Друга фаза взаємодії, що також відображена в додатку А, є повністю автономною і розгортається у часі незалежно від того, чи відкрито програму на екрані пристрою. Коли внутрішній годинник операційної системи досягає запланованого часу, `Android` ініціює повідомлення, яке перехоплюється

системним приймачем додатку. Таким чином, система проактивно нагадує про необхідність прийому ліків.

Завершальна фаза, відповідно до додатка А, демонструє реактивність. Отримавши сповіщення, користувач натискає кнопку підтвердження прийому. Ця дія формує новий запит, який передається до репозиторію для оновлення статусу запису в базі даних Room. Ключовою архітектурною особливістю є те, що після оновлення статусу репозиторію не потрібно вручну сповіщати інтерфейс про зміни. Головний екран та екран історії, будучи підписаними на зміни, миттєво і синхронно отримують оновлений стан, та перемальовують інтерфейс.

До моделювання часової послідовності подій, комплексний аналіз динаміки програмного забезпечення вимагає моделювання взаємозв'язків між об'єктами. Для вирішення цього завдання застосовується діаграма кооперації. Ця діаграма демонструє цілісну картину взаємодії між модулями користувацького інтерфейсу, шаром бізнес-логіки, локальним сховищем та системними фоновими сервісами, як це проілюстровано на рисунку 2.1.4.

Відповідно до рисунка 2.1.4, центром, що ініціює більшість потоків управління, є користувач, який взаємодіє з графічним інтерфейсом. Будь-яка активність, формується як первинне повідомлення, маркер 1 на діаграмі. Зважаючи на правила архітектури MVVM, графічний інтерфейс не має прямих зв'язків із базою даних чи системними сервісами. Тому повідомлення передається до відповідного контролера стану – об'єкта ViewModel, маркер 1.1. він перетворює їх на команди та передає їх далі до MedicationRepository.

Як демонструє рисунок 2.1.4, MedicationRepository виступає головним диспетчером системи, маючи зв'язки відразу з кількома критичними підсистемами. Отримавши запит від ViewModel, маркер 1.2, репозиторій розділяє потік виконання. З одного боку, він ініціює взаємодію з об'єктами доступу до даних, передаючи зашифровані пакети інформації для фізичного збереження в локальній базі даних Room, маркер 1.2.1. З іншого боку, через паралельний канал зв'язку, репозиторій надсилає повідомлення до модуля

AlarmSchedulerImpl, маркер 1.2.2. Ця комунікація є необхідною для синхронізації збережених даних із розкладом операційної системи. Планувальник, контактує з Android для реєстрації таймера, маркер 1.2.3. Таким чином, структурно забезпечується ізоляція бази даних від механізмів планування.

Другий глобальний потік повідомлень, відображений на діаграмі кооперації, характеризує асинхронну реакцію системи на настання запланованих подій. Точкою входу стає операційна система Android, яка генерує сигнал, маркер 2. Цей сигнал структурно направлений об'єкта, здатного його обробити – MedicationAlarmReceiver. Отримавши, приймач не звертається до бази даних напряду, а використовує зв'язок з допоміжним класом NotificationHelper, відправляючи йому інструкцію, маркер 2.1. Після того як сповіщення виводиться на екран, користувач отримує змогу відреагувати на нього, що замикає цикл комунікації, повертаючи управління до графічного інтерфейсу та ViewModel. Подібні взаємозв'язки підтверджує дотримання принципів чистої архітектури.

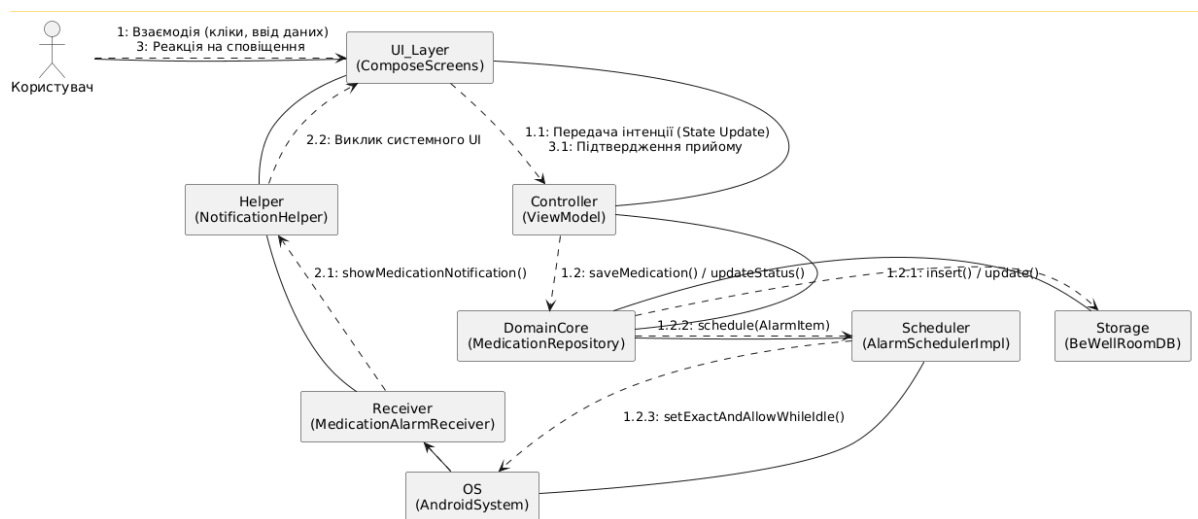


Рис. 2.1.4 Діаграма кооперації компонентів системи

2.2. Архітектура програмного забезпечення

На основі концептуальних та поведінкових моделей, розроблених та описаних у підрозділі 2.1, була сформована багаторівнева архітектура програмного забезпечення (див. розділ 2.2). Якщо UML-діаграми визначали теоретичну топологію взаємозв'язків між об'єктами, статичну структуру,

послідовність дій та кооперацію, то фізична реалізація системи вимагає чіткого групування цих об'єктів у логічні архітектурні шари. Рівень представлення є найвищим з таких шарів розроблюваної програмної системи, який безпосередньо відповідає за рендеринг графічного інтерфейсу та організацію інтерактивної взаємодії з кінцевим користувачем.

Ключовою особливістю реалізованого рівня представлення є повна відмова від класичного підходу побудови інтерфейсів через XML-розмітки на користь UI-фреймворку Jetpack Compose. Замість покрокового керування станом окремих візуальних компонентів, система декларативно описує, як саме повинен виглядати інтерфейс для будь-якого заданого стану даних. Графічний інтерфейс додатка побудований з набору незалежних, слабо зв'язаних функцій-компонентів Composables.

Як було закладено під час проєктування, зокрема, на діаграмах класів та кооперації, архітектурно цей рівень жорстко підпорядкований патерну Model-View-ViewModel. Роль представлення відіграють функції Jetpack Compose, які формують логічні екрани додатка: головну панель розкладу HomeDashboardScreen(додаток Б), форму налаштування ліків AddMedicationScreen(додаток В), календар історії HistoryScreen(додаток Г), візуальні аналітичні звіти ReportsScreen(Додаток Д) та профіль користувача ProfileScreen(Додаток Е). Кожен із цих екранів має тільки логіку відображення. За управління логікою взаємодії відповідають класи ViewModel. Вони виступають контролерами, які утримують стан екрану .

Організація переходів між модулями реалізована за концепцією Single-Activity Architecture – архітектура єдиної активності. Замість створення окремих компонентів Activity, додаток використовує бібліотеку Navigation Compose. Клас навігації ініціалізує єдиний контейнер (див. Додаток Ж), який керує стеком екранів на основі заданих маршрутів. Нижня навігаційна панель (Додаток З) інтегрована в загальний каркас екрану і динамічно синхронізується з поточним станом навігаційного контролера.

Оскільки програма оперує чутливою інформацією щодо стану здоров'я користувача, доступ до графа навігації захищено екраном блокування. Цей компонент є обгорткою над усім інтерфейсом, за це відповідає AuthWrapper (Додаток К). Доступ до основних функцій застосунку отримується тільки після успішної автентифікації користувача.

Наступним рівнем у розробленій архітектурі, є рівень домену. Рівень домену визначає те, що ці дані означають і як вони повинні оброблятися. Головною метою виділення цього шару є повне відокремлення бізнес-логіки додатка. Такий підхід є ядром чистої архітектури, оскільки дозволяє тестувати бізнес-правила ізольовано та забезпечує високу гнучкість системи при можливих майбутніх модифікаціях.

Центральним компонентом в розробленому додатку виступає клас MedicationRepository (Додаток Л). Він реалізує фундаментальний патерн проєктування «Єдине джерело істини». Це означає, що жоден контролер стану з рівня представлення не має права звертатися до бази даних безпосередньо. Усі запити на отримання розкладу ліків, додавання нових препаратів, оновлення статусів прийому чи генерацію статистичних звітів проходять виключно через репозиторій. Такий бар'єр дозволяє репозиторію не просто ретранслювати дані, а оперувати складними процесами.

Важливою функцією рівня домену є агрегація та трансформація даних. Дані, що зберігаються у реляційній базі, часто не є готовими для прямого відображення на екрані. Репозиторій бере на себе завдання з об'єднання нормалізованих табличних сутностей наприклад, PillEntity (Додаток М) та IntakeRecordEntity (Додаток Н) у зручній моделі IntakeWithPill (додаток Ї). Саме на цьому рівні зосереджена логіка розрахунку показників, таких як індекс дотримання режиму лікування Adherence score(див. Додаток Л). Репозиторій аналізує історичні дані про прийняті, пропущені дози, застосовує математичні алгоритми оцінки та повертає статистику.

Для забезпечення асинхронного обміну інформацією між компонентами системи, репозиторій використовує Kotlin Coroutines та потоки даних Flow. Це

дозволяє домену підтримувати актуальність даних у реальному часі без необхідності ручного втручання.

Окрім роботи з даними, рівень домену виконує роль координатора системних сервісів. Збереження нового графіка лікування є бізнес-подією, яка вимагає не лише запису на диск, але й реакції операційної системи. Тому репозиторій взаємодіє з інтерфейсами підсистеми фонових завдань AlarmScheduler (Додаток Р). Після успішного формування графіку прийомів, домен передає планувальнику завдання з реєстрації точних системних будильників (Додаток С) для запису. Рівень домену утворює надійний та інтелектуальний центр додатка, який гарантує цілісність.

Якщо репозиторій визначає правила обробки інформації, то рівень даних відповідає за довготривале збереження, цілісність, консистентність та ефективне вилучення з енергонезалежної пам'яті мобільного пристрою. Для реалізації даного рівня було обрано бібліотеку збереження даних Room, яка входить до складу рекомендованих архітектурних компонентів Android Jetpack.

Фундаментом рівня даних є сутності (Entities), які описують схему реляційної бази даних. У розробленому програмному забезпеченні структура збереження інформації нормалізована та логічно розділена на кілька взаємопов'язаних таблиць. Базовою сутністю є клас PillEntity (див. Додаток М). Для відстеження щоденної динаміки курсу лікування використовується сутність IntakeRecordEntity (див. Додаток Н). Оскільки один медичний препарат передбачає множинні прийоми протягом курсу лікування, між цими двома таблицями встановлено класичний реляційний зв'язок типу «один-до-багатьох». Для забезпечення ефективного отримання об'єднаних даних застосовується спеціальний допоміжний клас IntakeWithPill (див. Додаток П).

Взаємодія з нормалізованими таблицями бази даних здійснюється виключно через об'єкти доступу до даних Data Access Objects (DAO). Інтерфейси PillDao (Додаток Т) та IntakeRecordDao (Додаток У) містять оптимізовані методи для запису даних, оновлення статусів прийому та вилучення розкладу на конкретну дату.

Об'єднання всіх описаних сутностей та об'єктів DAO в єдину працездатну екосистему здійснюється через головний абстрактний клас бази даних BeWellRoomDB, який безпосередньо успадковується від системного класу RoomDatabase (Додаток Ф). Цей клас виконує роль конфігуратора параметрів підключення, використовуючи патерн проєктування Singleton. Враховуючи специфіку медичного спрямування додатка та суворі вимоги до захисту персональної інформації, рівень даних інтегрує механізми криптографічного перетворення. Перед збереженням полів у таблиці SQLite, вони проходять процедуру шифрування за допомогою класу EncryptionManager, який використовує сучасні стандарти криптографії (Додаток Х).

Враховуючи специфіку розроблюваного програмного забезпечення, інформація про призначені медичні препарати, дозування та деталізована історія їх прийому належить до категорії критично чутливих персональних даних. Компрометація або несанкціонований витік такої інформації може призвести до розголошення лікарської таємниці та серйозного втручання в приватне життя користувача. Саме тому в архітектуру системи на етапі проєктування було закладено механізм захисту даних, що поєднує криптографічне перетворення інформації та безпосередньо на рівні користувацького інтерфейсу.

Першим рубежем захисту є шифрування локальної реляційної бази даних. Оскільки SQLite зберігає інформацію у системі у відкритому текстовому вигляді, для забезпечення конфіденційності було розроблено модуль управління шифруванням. Система функціонує автономно, вона генерує 32-байтову випадкову послідовність. Для максимально безпечного зберігання згенерованого ключа застосовується компонент EncryptedSharedPreferences у інтеграції з системним сховищем Android Keystore. Операційна система створює головний майстер-ключ на базі захищеного апаратного модуля пристрою, використовуючи стандарт симетричного шифрування AES-256 (Додаток Х).

Другим, компонентом безпеки є контроль доступу на рівні самого додатка, який реалізується через біометричну автентифікації. Під час кожного запуску програми або її повернення з фоновому режиму роботи, система ініціює

звернення до системного API BiometricPrompt. Цей сервіс взаємодіє з апаратними сенсорами пристрою, перевіряючи наявність валідної біометричної ознаки користувача, відбитка пальця або пароля. Перехід до основного графа навігації Jetpack Compose та розшифрування інформації блокується до моменту успішного проходження верифікації (Додаток К).

Синтез цих двох підходів створює надійну екосистему безпеки розробленого рішення. Шифрування бази даних ефективно унеможливорює офлайн-атаки та несанкціоноване вилучення даних через файлову систему, у той час як біометричний контроль доступу захищає інформацію від компрометації під час використання активного пристрою.

2.3. Конструювання програмного забезпечення

Процес розробки розпочинається з реалізації фізичного рівня зберігання даних. Відповідно до визначеної архітектури, для реалізації локальної реляційної бази даних у розроблюваному Android-додатку було використано бібліотеку Room.

Процес реалізації бази даних розпочався зі створення дата-класів, які виконують роль табличних сутностей. Для збереження незмінної інформації про медичний препарат розроблено клас PillEntity. За допомогою анотації @Entity цей клас прив'язано до таблиці з назвою «pills». У структурі класу визначено первинний ключ з автоматичною генерацією унікальних значень, а також набір полів для збереження назви ліків, форми випуску, дозування та додаткових інструкцій щодо прийому (Додаток М). Для ведення динамічного журналу призначень створено сутність IntakeRecordEntity, що відповідає таблиці «intake_records». Цей клас містить зовнішній ключ, який вказує на ідентифікатор конкретного препарату з таблиці «pills», а також зберігає запланований час прийому у форматі мілісекунд та поточний текстовий статус виконання події (Додатку І).

Оскільки архітектура додатку вимагає частого отримання комплексної інформації для вирішення цього завдання було спроектовано проміжний data-клас IntakeWithPill. Використовуючи анотації @Embedded для вбудовування

полів сутності запису та @Relation для визначення зв'язку за ідентифікатором препарату, цей клас дозволяє бібліотеці Room автоматично об'єднувати пов'язані дані у вигляді єдиного об'єкта під час виконання запиту (Додаток П).

Наступним кроком стала реалізація DAO. Інтерфейси PillDao та IntakeRecordDao містять набір методів для виконання базових CRUD-операцій. Методи для вставки, оновлення та видалення оголошені з модифікатором suspend, що змушує компілятор виконувати ці важкі транзакції виключно у фонових потоках, не блокуючи головний потік інтерфейсу користувача. Натомість методи читання даних, такі як отримання розкладу на конкретну дату, спроектовані для повернення об'єкта Flow. (Див. Додаток Т, Додаток У) .

Завершальним етапом проєктування локального сховища стало створення головного класу бази даних BeWellRoomDB. У цьому класі задекларовано всі використані сутності та вказано поточну версію схеми бази даних. Для забезпечення єдиної точки доступу до підключення реалізовано патерн Singleton. Важливим аспектом стала інтеграція фабрики SupportSQLiteOpenHelper, що надається класом EncryptionManager. Це дозволило підключити механізм 256-бітного шифрування AES, забезпечивши надійний захист збережених медичних записів (Додаток Ф). Реалізована локальна база даних утворює швидкий, реактивний та криптографічно захищений рівень збереження інформації, що повністю відповідає вимогам розроблюваної багаторівневої системи.

Після забезпечення надійного та безпечного механізму локального збереження медичних даних та розкладу прийомів , наступним завданням було реалізація підсистеми фонових сповіщень. Сучасні версії операційної системи Android мають агресивну політику енергозбереження. Однак для програмного забезпечення медичного призначення затримка нагадування навіть на кілька хвилин є неприпустимою. З огляду на це, від використання стандартного планувальника відкладених завдань WorkManager було відмовлено на користь сервісу AlarmManager, який здатний "пробуджувати" пристрій у точно визначений час.

Практична реалізація розпочалася зі створення абстрактного інтерфейсу `AlarmScheduler`. Безпосередня взаємодія з операційною системою реалізована у класі `AlarmSchedulerImpl`. Ключовим технічним рішенням у цьому класі є використання методу `setExactAndAllowWhileIdle()`. На відміну від звичайних будильників, цей метод гарантує спрацьовування таймера точно в задану секунду, ігноруючи навіть глибокий режим сну операційної системи. Для безпечної передачі даних генерується об'єкт `PendingIntent`, який зберігає контекст виклику та корисне навантаження до моменту настання події. Крім того, на рівні цього класу імплементовано перевірку системних дозволів, запроваджених у нових версіях `Android` для контролю за точними будильниками (Додаток С).

Наступною ланкою є перехоплення сигналу від операційної системи після спрацьовування таймера. Для цього в архітектуру додатка було інтегровано клас `MedicationAlarmReceiver`. Метод `onReceive()` цього класу виступає точкою входу: він вилучає з отриманого наміру раніше переданий ідентифікатор ліків та ініціює фінальний етап – генерацію візуального інтерфейсу нагадування (Додаток Ц).

За формування та виведення сигналу відповідає клас `NotificationHelper..` Саме сповіщення конструюється за допомогою патерну `Builder`, де налаштовуються візуальні атрибути: іконка додатка, заголовок, зміст нагадування та параметри автоматичного скасування. Для забезпечення інтерактивності до сповіщення додаються дії-наміри, які реалізуються через додаткові об'єкти `PendingIntent` (Додаток Ш).

Після успішного налаштування локального сховища даних та реалізації автономної підсистеми фонових сповіщень, найбільш об'ємним завданням етапу конструювання стала реалізація графічного інтерфейсу користувача. Відповідно до обраної архітектурної стратегії, візуальна частина розроблюваної системи будується на базі сучасного фреймворку `Jetpack Compose`. Процес конструювання інтерфейсу зводиться до написання функцій `Composables`, які приймають стан даних як аргумент і генерують відповідне візуальне представлення, автоматично оновлюючись при будь-яких змінах цього стану.

Основою візуальної навігації додатка є AppNavigation (Додаток Ж). Замість створення окремих активностей для кожного екрану, система використовує підхід єдиної активності, де контролер навігації о підмінює активні компоненти-екрани. Цей контролер інтегровано з каркасом інтерфейсу BottomBar (див. Додаток Ж), що забезпечує користувачу швидкий і безшовний доступ до ключових розділів програми: головної панелі, історії, звітів та профілю. Першим візуальним компонентом, який ініціалізується при запуску системи, обгортка безпеки AuthWrapper (Додаток К) та компонент LockScreen (Додаток Щ). Вони відображають інтерфейс вимоги біометричної автентифікації, блокуючи рендеринг медичних даних до моменту успішної перевірки особи користувача.

Після проходження процедури автентифікації користувач потрапляє на головний екран HomeDashboardScreen (Додаток Б). Цей екран спроектовано як комплексну панель управління поточним днем. У верхній частині розташовано динамічний компонент дати HomeDashboardDate (Додаток Б.1) та систему фільтрів за статусом прийому HomeDashboardFilters (Додаток Б.2), що дозволяє сортувати відображені завдання. Основну площу екрану займає список призначених препаратів. Кожен елемент списку реалізовано як компонент SwipeableMedicationItem (Додаток Б.3). Взаємодія з ним миттєво генерує подію для HomeViewModel (Додаток Б.4), яка ініціює запис у базу даних та подальше оновлення списку.

Для введення даних про нові призначення розроблено екран додавання медикаменту AddMedicationScreen (Додаток В). Екран розділено на логічні блоки, реалізовані як окремі функції. Компонент InputTextField (Додаток В.1) відповідає за текстове введення назви та дозування з вбудованою валідацією, компонент MedicationType (Додаток В.2) представляє собою сітку іконок для вибору форми випуску, а CustomTimePicker (Додаток В.3) забезпечує зручний вибір часу прийому. Усі введені дані збираються у стані AddMedicationViewModel (Додаток В.4), яка гарантує, що кнопка збереження стане активною лише після коректного заповнення всіх обов'язкових полів.

Аналіз курсу лікування здійснюється на екрані історії HistoryScreen (Додаток Г). Його архітектура складається з двох основних синхронізованих компонентів: горизонтального прокручуваного календаря HorizontalCalendar (Додаток Г.1) та вертикальної часової шкали TimeLineItem (Додаток Г.2). При виборі конкретного дня у верхньому календарі, компонент надсилає подію до HistoryScreenViewModel (Додаток Г.3), яка отримує з бази даних Room історичний зріз за вказану добу. Частина екрану реактивно перемальовується, відображаючи хронологію прийомів у вигляді карток HistoryMedicationCard (Додаток Г.4) з відповідними кольоровими маркерами статусу, що дозволяє користувачу візуально оцінити свою дисципліну в минулому.

Для глибшого аналізу довгострокової статистики спроектовано екран звітів ReportsScreen (Додаток Д). Цей модуль використовує бібліотеки для малювання у Compose Canvas, формуючи наочні аналітичні віджети. До них належать картка місячного дотримання режиму MonthlyAdherenceCard (Додаток Д.1), яка відображає загальний відсоток успішних прийомів, та графік тижневого тренду WeeklyTrend (Додаток Д.2). Крім того, на цьому екрані розміщено компонент ExportCard (Додаток Д.3), який надає інтерфейс для генерації звітів у зовнішні формати PDF або CSV для подальшої передачі лікуючому лікарю.

Завершальним елементом інтерфейсу є екран профілю користувача ProfileScreen (Додаток Е), який виконує функцію системних налаштувань. Екран побудовано з використанням перевикористовуваних компонентів списку ProfileMenuItem (Додаток Е.1), об'єднаних у логічні секції ProfileMenuSection (Додаток Е.2). Тут користувач може керувати загальними параметрами додатка. Загалом, використання Jetpack Compose дозволило створити цілісний, сучасний та реактивний користувацький інтерфейс, який безшовно реагує на події базових архітектурних шарів та забезпечує оптимальний досвід взаємодії з програмним продуктом.

2.4. Аналіз безпеки

Надійна багаторівнева архітектура програмного забезпечення, розроблена на попередніх етапах конструювання, створює міцний функціональний

фундамент системи, обробка медичної інформації вимагає імплементації механізмів захисту. Оскільки стандартні засоби збереження даних у мобільних операційних системах, зберігають інформацію на фізичному носії у відкритому текстовому вигляді, вони є вразливими до компрометації. У разі втрати пристрою, зломисник може отримати прямий доступ до файлової системи та вилучити всю історію хвороби користувача. Для нівелювання цього в архітектуру розробленого додатка було нативно інтегровано механізм, який забезпечує криптографічне перетворення всіх локальних даних.

Процес ініціалізації захисту розпочинається з генерації унікальної фрази-пароля. Для цього застосовується криптографічно стійкий генератор псевдовипадкових чисел SecureRandom, який формує масив із 32 байтів високої ентропії, що згодом кодується у формат Base64 для зручності подальшої обробки.

Наступним викликом стало безпечне збереження цієї згенерованої фрази-пароля, адже збереження її у відкритому вигляді у звичайних налаштуваннях звело б нанівець усю стратегію захисту. Для вирішення цієї проблеми використано апаратно-підтримувану підсистему Android Keystore. Модуль шифрування звертається до системи для створення об'єкта MasterKey, що базується на симетричному алгоритмі шифрування AES-256 у режимі лічильника з автентифікацією Галуа (GCM). Використовуючи цей апаратний майстер-ключ, система ініціалізує захищене сховище EncryptedSharedPreferences, куди і записується раніше згенерована 32-байтова фраза-пароль бази даних.

Безпосереднє шифрування самої реляційної бази даних Room реалізується за допомогою розширення SQLCipher. Модуль EncryptionManager виступає фабрикою SupportSQLiteOpenHelper.Factory, яка надає розшифровану фразу-пароль під час ініціалізації підключення. Завдяки цьому інтеграція відбувається абсолютно прозоро для вищих архітектурних шарів: об'єкти доступу до даних (DAO) та репозиторій продовжують працювати зі звичайними SQL-запитами та реактивними потоками, тоді як на найнижчому рівні SQLCipher автоматично

шифрує кожен сторінку бази даних перед записом на носій та дешифрує її при завантаженні в оперативну пам'ять.

Цей підхід гарантує повну конфіденційність медичної історії користувача на фізичному рівні, при цьому залишаючись абсолютно непомітним для кінцевого споживача.

Надійне криптографічне шифрування локальної бази даних, описане вище, ефективно вирішує проблему захисту інформації в стані спокою. Однак для забезпечення комплексного захисту медичних даних цього недостатньо, оскільки існує можливість отримати дані вже розблокованого пристрою. Для протидії таким загрозам, загальну архітектуру додатка було доповнено механізмом обов'язкової біометричної автентифікації користувача під час кожного сеансу роботи.

Практична реалізація цього механізму базується на використанні офіційної бібліотеки `AndroidX Biometric`. Процес перевірки особи інтегровано безпосередньо у граф навігації рівня представлення `Jetpack Compose` через спеціальний компонент-обгортку `AuthWrapper` (Додаток К).

Під час активної фази перевірки життєвий цикл додатка тимчасово призупиняється, а управління екраном передається захищеному системному процесу ОС `Android`. Це унеможливорює перехоплення даних шкідливим програмним забезпеченням або зчитування даних екрану. У разі успішного підтвердження особи системний `callback` оновлює реактивну змінну стану всередині `AuthWrapper`, що автоматично дозволяє рендеринг основного навігаційного графа та завантаження медичної інформації з репозиторію. Якщо ж автентифікація не пройдена або свідомо скасована користувачем, система автоматично перенаправляє потік виконання до ізольованого компонента `LockScreen`. Цей екран блокування структурно перекриває весь користувацький інтерфейс, запобігаючи витоку даних через систему багатозадачності або знімки останніх відкритих додатків, і надає лише базову кнопку для повторної ініціалізації процесу перевірки особи (Див. Додаток К).

Цей підхід гарантує, що доступ до розкладу лікування та історії хвороби отримає виключно авторизований власник пристрою, зберігаючи при цьому високий рівень повсякденної зручності використання додатка.

Висновки до розділу

Отже на основі проведеного аналізу було обґрунтовано вибір сучасного технологічного стеку від Google, де базовою мовою програмування виступає Kotlin, а для побудови користувацького інтерфейсу використовується декларативний фреймворк Jetpack Compose. Такий вибір дозволив забезпечити високу швидкість розробки, реактивність інтерфейсу та легкість впровадження принципів інклюзивного дизайну, що є критично важливим для цільової аудиторії літнього віку.

Для забезпечення високої масштабованості, незалежності бізнес-логіки та спрощення подальшого тестування системи було впроваджено архітектурний патерн MVVM у суворій комбінації з принципами Clean Architecture. Завдяки розподілу застосунку на незалежні шари та використанню принципів інверсії залежностей, вдалося досягти високої модульності коду. Локальне зберігання медичних даних пацієнта реалізовано за допомогою надійної ORM-бібліотеки Room, що повністю задовольняє архітектурну вимогу «offline-first» і гарантує безперебійну роботу додатку без підключення до мережі Інтернет.

Особливу увагу в процесі конструювання було приділено вирішенню найбільш складної технічної задачі даного класу застосунків – забезпеченню абсолютної точності та надійності спрацьовування нагадувань про прийом медикаментів. Зважаючи на агресивні політики енергозбереження в сучасних операційних системах та нові суворі обмеження фонових процесів в ОС Android 14 та Android 15, було розроблено комбінований механізм планування. Для критичних за часом сповіщень імплементовано використання AlarmManager із дозволом SCHEDULE_EXACT_ALARM, тоді як для менш пріоритетних фонових задач задіяно можливості WorkManager. Цей симбіоз гарантує, що пацієнт не пропустить життєво важливий прийом ліків за жодних обставин.

РОЗДІЛ 3

АНАЛІЗ ЯКОСТІ ТА БЕЗПЕКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1. Аналіз якості ПЗ та опис процесів тестування.

Після успішного завершення етапу архітектурного проектування та практичного конструювання програмного забезпечення, які були висвітлені у другому розділі, наступним кроком життєвого циклу розробки є аналіз якості та стабільності створеного продукту. Оскільки розроблена система належить до категорії програмного забезпечення медичного призначення, оперує конфіденційними даними та відповідає за своєчасне інформування користувача про графік терапії, будь-яка логічна помилка у коді може мати серйозні наслідки для здоров'я пацієнта. Для мінімізації цих ризиків та забезпечення максимальної надійності, процес автоматизованої верифікації розпочинається з базового та найнижчого рівня забезпечення якості – модульного тестування.

Модульне тестування передбачає сувору, ізольовану перевірку окремих, найменших функціональних компонентів вихідного коду на предмет їхньої повної відповідності закладеній бізнес-логіці.

Головним об'єктом модульного тестування у створеній системі виступає рівень домену, зокрема центральний клас `MedicationRepository`. Оскільки цей репозиторій курує складні процеси збереження даних та паралельного встановлення системних таймерів, для його ізольованої перевірки застосовується методика створення об'єктів-імітацій за допомогою спеціалізованих інструментів. Під час виконання тестових сценаріїв реальні об'єкти доступу до локальної бази даних, `PillDao`, `IntakeRecordDao` та інтерфейс системного планувальника `AlarmScheduler` підміняються контрольованими тестовими двійниками. Це дозволяє перевірити алгоритми розрахунку розкладу прийомів ліків, коректність даних та правильність обробки станів без необхідності реальної ініціалізації зашифрованої бази даних `Room` чи системних сервісів мобільного пристрою.

Наступним етапом є тестування контролерів стану – класів `ViewModel`. Оскільки ці класи використовують асинхронні корутини та реактивні потоки даних, їх тестування вимагає підміни стандартних диспетчерів потоків на спеціалізовані тестові диспетчери. Тестові набори для класів `ViewModel` сфокусовані на перевірці того, чи правильно контролер реагує, чи коректно оновлюється структура стану екрану після отримання відповіді від імітованого репозиторію, та чи генеруються відповідні інформаційні повідомлення або винятки у разі введення невалідних параметрів під час налаштування графіка лікування.

Комплексне покриття модульними тестами базових архітектурних шарів формує надійний фундамент якості всього програмного продукту.

Продовженням процесу автоматизованої верифікації, після успішного підтвердження коректності бізнес-логіки на етапі модульного тестування, є тестування користувацького інтерфейсу. Якщо модульні тести гарантують правильність обробки даних у ядрі системи, то інструментальні тести інтерфейсу покликані верифікувати коректність візуального відображення цих даних та адекватність реакції програми на дії кінцевого користувача. Для перевірки екранів використовується спеціалізоване API тестування, яке дозволяє розробнику ізолювати окремі функції-компоненти, маніпулювати їхнім станом та верифікувати наявність або відсутність певних семантичних вузлів у дереві користувацького інтерфейсу.

У рамках аналізу якості UI тестуванню підлягають як базові атомарні віджети, так і комплексні функціональні екрани. Особлива увага приділяється тестуванню складних інтерактивних елементів, таких як `SwipeableMedicationItem` на головній панелі. За допомогою автоматизованих інструментів імітуються жести змахування вліво та вправо по екрану. Після виконання жесту система перевіряє, чи коректно відображаються фонові іконки підтвердження або скасування, і головне – чи правильно генерується відповідна подія-інтенція для оновлення статусу, яка маршрутизується до контролера стану `ViewModel`.

Іншим напрямком інструментального тестування є перевірка наскрізних навігаційних потоків та механізмів захисту конфіденційності. Зважаючи на наявність обов'язкового екрана блокування та обгортки безпеки перевіряється ключова архітектурна вимога, граф навігації не повинен ініціалізувати відображення чутливих медичних даних до моменту отримання позитивного результату від імітованого системного сервісу біометричної автентифікації. Завдяки таким тестам підтверджується відсутність вразливостей, які могли б призвести до витоку інформації під час завантаження системи.

Також аналізу підлягає форма додавання нового запису. Сценарії тестування автоматизують процес заповнення полів введення як валідними, так і некоректними даними, імітують взаємодію з компонентами вибору форми випуску препарату та системними діалогами вибору часу. Метою цих перевірок є підтвердження того, що система своєчасно блокує кнопку збереження при нестачі даних, адекватно обробляє та забезпечує безпомилкову трансляцію введеної користувачем інформації у внутрішні структури даних. Цей підхід гарантує, що розроблена програма не лише виконує точні математичні розрахунки графіків прийому, але й стабільно, інтуїтивно зрозуміло та без візуальних артефактів взаємодіє з кінцевим користувачем на екрані мобільного пристрою.

Наступним етапом верифікації програмного продукту, є перевірка надійності фізичного рівня збереження інформації. Оскільки розроблена система покладається на локальну реляційну базу даних Room для зберігання критично важливих медичних записів, тестування цього компонента вимагає специфічного підходу. На відміну від чистих функцій рівня домену, DAO тісно взаємодіють із внутрішнім API SQLite операційної системи Android. Тому їхня верифікація зазвичай виконується у вигляді інструментальних тестів, які запускаються в середовищі реального пристрою або повноцінного емулятора, що забезпечує максимальну наближеність до реальних умов експлуатації додатка.

Для досягнення цієї мети під час ініціалізації тестів використовується спеціальний вбудований метод створення бази даних в оперативній пам'яті.

Безпосередній процес тестування фокусується на суворій верифікації методів, задекларованих в інтерфейсах об'єктів доступу до даних, зокрема `PillDao` та `IntakeRecordDao`. Тестові сценарії моделюють повний життєвий цикл інформаційних сутностей, охоплюючи всі базові CRUD-операції. Окрема увага приділяється перевірці складних реляційних зв'язків.

Враховуючи, що архітектура додатка концептуально базується на асинхронних технологіях, тестування бази даних також глибоко інтегрує інструментарій `Kotlin Coroutines`. Оскільки операції запису та оновлення статусів у `Room` маркуються як призупиняючі функції, їхній виклик у тестах обгортається у спеціальні блоки для тестування. Це дозволяє блокувати виконання тестового потоку до повного завершення асинхронної транзакції бази даних, забезпечуючи детерміноване і послідовне виконання перевірок. Особливій верифікації підлягає реактивна природа методів читання. Тестують здатність бази даних не лише повертати статичний результат, але й генерувати безперервний потік даних. Такий глибокий та ізольований аналіз рівня даних доводить, що SQL-запити сформовані синтаксично правильно, а додаток здатен стабільно і швидко маніпулювати великими масивами історії лікування.

3.2. Аналіз безпеки програмного забезпечення

Після завершення верифікації функціональних компонентів системи, , як було описано в розділі вище, критично важливим етапом стає проведення глибокого аналізу безпеки програмного забезпечення. Розділ присвячений дослідженню стійкості реалізованого шифрування локальної бази даних `Room` до потенційних атак. Основною моделлю загрози в цьому контексті виступає отримання зловмисником несанкціонованого фізичного доступу до файлової системи смартфона – наприклад, через втрату пристрою, використання експлойтів для отримання прав суперкористувача або створення примусових резервних копій через інтерфейс відлагодження `ADB`. Аналіз безпеки фокусується на підтвердженні неможливості вилучення відкритих текстових даних із файлів сховища без проходження легітимної процедури ініціалізації додатка.

Першим кроком безпекового аналізу є аудит процесу управління криптографічними ключами, який інкапсульовано в модулі EncryptionManager. Верифікація підтверджує, що архітектура генерації та зберігання ключового матеріалу цілком покладається на апаратно-захищене системне сховище Android Keystore. Аналіз властивостей згенерованого головного ключа засвідчує, що він створюється безпосередньо всередині ізольованого апаратного середовища виконання сучасних мобільних процесорів. Крім того, верифікується процес створення 32-байтової фрази-пароля, яка генерується з максимальним рівнем ентропії за допомогою класу SecureRandom і зберігається виключно в зашифрованому вигляді через компонент EncryptedSharedPreferences (Див. Додаток X).

Наступним етапом є перевірка стійкості збереженого файлу бази даних у стані спокою. Під час тестування безпеки моделюється ситуація примусового вилучення файлів бази даних SQLite. Подальші спроби відкрити ці файли за допомогою стандартних систем управління базами даних або проаналізувати підтверджують повну відсутність читабельних структур даних, текстових записів або навіть метаданих таблиць.

Таким чином, проведений аналіз безпеки доводить, що застосований механізм шифрування локальної бази даних працює коректно і безвідмовно. Він гарантує абсолютну конфіденційність медичної історії та розкладу лікування користувача, повністю нейтралізуючи загрози, пов'язані з фізичним вилученням або несанкціонованим копіюванням інформаційних носіїв мобільного пристрою.

Аналізом криптографічної стійкості локальної бази даних, яка захищає інформацію в стані спокою, є верифікація механізмів захисту даних під час їх активного використання. Для цього у системі було реалізовано обов'язкову біометричну автентифікацію. Головна мета тестування полягала у підтвердженні неможливості обходу екрану блокування та недопущенні витoku конфіденційної інформації через механізми багатозадачності операційної системи.

Першим етапом аудиту стала перевірка архітектурної надійності компонента-обгортки AuthWrapper, який виконує роль єдиної точки входу в

користувачський інтерфейс. Аналіз дерева декларативних компонентів Jetpack Compose підтверджує, що ініціалізація головного навігаційного графа та завантаження відповідних моделей предметної області жорстко заблоковані до моменту отримання позитивного системного токена від сервісу BiometricPrompt. Під час моделювання атак, спрямованих на перехоплення життєвого циклу активності, система продемонструвала стабільну поведінку: стан автентифікації миттєво скидається, і користувач перенаправляється на захищений екран .

Окремому дослідженню підлягав сам процес взаємодії з біометричними сенсорами через API операційної системи Android. Верифікація підтвердила, що додаток працює відповідно до принципу мінімальних привілеїв і не отримує прямого доступу до біометричних даних. Процес обробки та порівняння біометричних шаблонів здійснюється виключно в апаратно-ізолюваному середовищі. Запит на автентифікацію викликає захищений системний діалог, який функціонує в окремому процесі. Крім того, аналіз підтвердив коректність реалізації механізмів резервного доступу, у випадку апаратного збою сенсора або тимчасової неможливості зчитування біометрики, система безпечно делегує перевірку системному PIN-коду або графічному ключу пристрою, не знижуючи загального рівня захищеності (Див. Додаток К).

Проведений комплексний аналіз доводить, що інтегрована підсистема біометричної автентифікації є надійною та стійкою до спроб компрометації..

Завершальним етапом комплексного аналізу безпеки, який органічно доповнює механізми криптографічного захисту даних у стані спокою та біометричної автентифікації в процесі використання, є верифікація політик запобігання витоку даних. Навіть за умови надійного локального шифрування та суворого контролю фізичного доступу, існує ризик неконтрольованого копіювання або експорту конфіденційної медичної інформації через вбудовані механізми операційної системи чи сторонні канали комунікації. З метою нейтралізації цих загроз у базову архітектуру розробленого додатка інтегровано низку превентивних заходів, що жорстко обмежують несанкціонований рух даних за межі ізолюваного середовища виконання програми.

Першою є стандартна підсистема автоматичного резервного копіювання операційної системи Android. За замовчуванням система може періодично архівувати локальні файли додатка та вивантажувати їх у хмарне сховище Google Drive. У випадку компрометації хмарного облікового запису користувача, ці резервні копії можуть стати безперешкодним джерелом витоку медичної історії. Для блокування цього вектора в конфігурації додатка було задекларовано спеціалізовані правила екстракції даних та резервного копіювання. Аналіз цих XML-правил підтверджує, що директорії, які містять зашифровану базу даних Room та ключовий матеріал, суворо виключені з будь-яких процесів хмарної синхронізації або локального перенесення даних через інструменти налагодження. Це гарантує, що база даних існує виключно на поточному фізичному пристрої і не може бути приховано скопійована зовнішніми системними сервісами.

Другим є контроль за легітимним експортом інформації. Оскільки додаток надає користувачу право ділитися історією свого лікування з профільним лікарем, система передбачає спеціалізований модуль генерації звітів у форматах PDF та CSV. Безпековий аудит цього компонента підтвердив, що процес експорту ініціюється виключно за умови активної сесії всередині біометрично захищеного графа навігації. Крім того, згенеровані файли звітів не зберігаються у загальнодоступних теках файлової системи пристрою. Таким чином, додаток не залишає стійких цифрових слідів або залишкових файлів після завершення операції експорту.

Окремій перевірці підлягала відповідність системи сучасним юридичним стандартам конфіденційності. Аналіз контролера профілю користувача засвідчив наявність функції повного та безповоротного знищення всієї медичної інформації. Під час ініціалізації процесу очищення система не просто видаляє візуальні зв'язки в інтерфейсі, а виконує глибоке каскадне очищення всіх таблиць бази даних та знищує пов'язані конфігураційні файли. Цей процес унеможлиблює відновлення розкладу терапії після того, як користувач вирішив припинити використання програмного продукту. Комплексне застосування

описаних підходів підтверджує, що додаток стійкий як до зовнішніх, так і до внутрішніх загроз витоку конфіденційних медичних даних.

Фіналом тестування та перевірки політик безпеки, є верифікація системи шляхом реалізації наскрізного контрольного прикладу. Цей підхід передбачає наочну демонстрацію повного життєвого циклу взаємодії кінцевого користувача з додатком у реальних умовах експлуатації. Метою контрольного прикладу є підтвердження того, що всі раніше описані архітектурні абстракції, алгоритми шифрування бази даних Room та реактивні компоненти Jetpack Compose успішно інтегровані в єдиний, безперебійно функціонуючий програмний продукт.

Відповідно до закладених політик захисту конфіденційності, які було описано в розділі 3.2., взаємодія із системою розпочинається з запуску додатка, процес якого негайно перехоплюється компонентом екрану блокування. На екрані мобільного пристрою з'являється системний діалог біометричної автентифікації, який вимагає від користувача підтвердити свою особу за допомогою сканера відбитка пальця. Будь-які спроби обійти цей екран без успішної верифікації блокуються обгорткою безпеки, а медичні дані залишаються в зашифрованому стані.

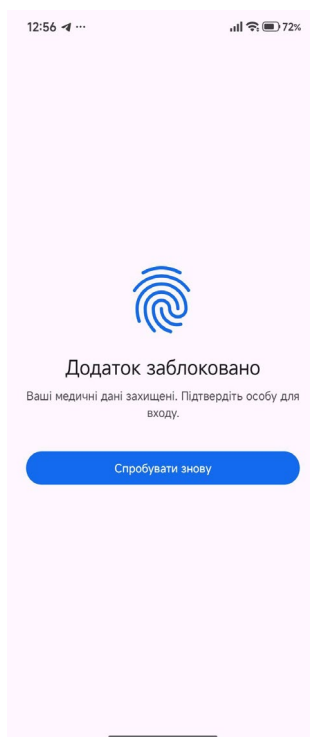


Рис. 3.1. Процес біометричної автентифікації користувача

Після успішного проходження процедури автентифікації система ініціює дешифрування бази даних та перенаправляє потік управління на головну панель розкладу. Оскільки під час першого запуску локальна база даних є порожньою, користувач спостерігає інтерфейс із пропозицією створити нове призначення. Для цього він натискає на відповідну кнопку і переходить до екрана додавання медикаменту. Тут користувач послідовно заповнює форму.

Рис. 3.2. Налаштування розкладу прийому препарату

Натискання кнопки збереження ініціює каскад внутрішніх процесів: валідацію введених даних на рівні ViewModel, їх асинхронне шифрування та збереження в таблиці SQLite через MedicationRepository, а також реєстрацію точного будильника за допомогою AlarmManager. Одразу після цього система автоматично повертає користувача на головну панель, де щойно доданий препарат миттєво з'являється у вигляді інтерактивної картки у списку завдань на поточний день зі статусом очікування.

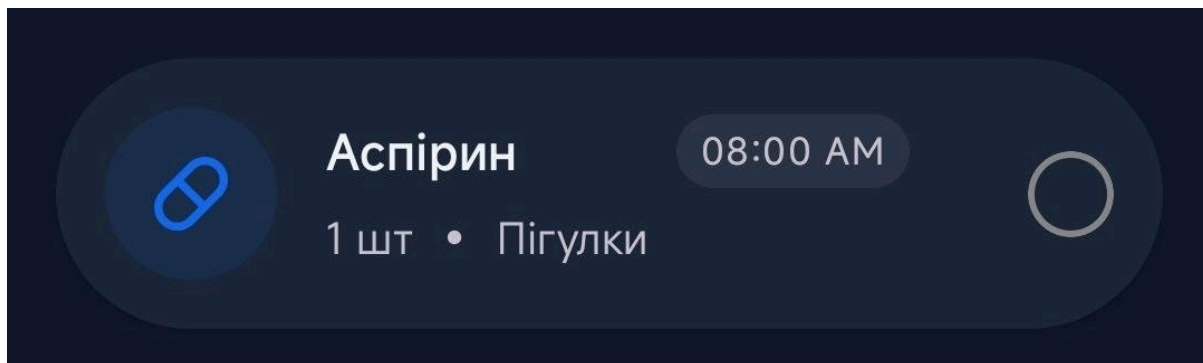


Рис. 3.3. Головна панель додатка з відображенням запланованого прийому ліків

Кульмінацією контрольного прикладу є перевірка механізму фонових нагадувань та реакції користувача на них. З настанням раніше запрограмованого часу, операційна система пробуджує автономний приймач додатка, який генерує високопріоритетне системне Push-сповіщення. Сповіщення містить назву препарату та дозування, вимагаючи свідомої реакції. Користувач відкриває додаток і, використовуючи інтуїтивний жест тапання по картці медикаменту на головному екрані, підтверджує факт вживання ліків. Візуальний стан картки миттєво змінюється, фіксуючи успішне виконання завдання.

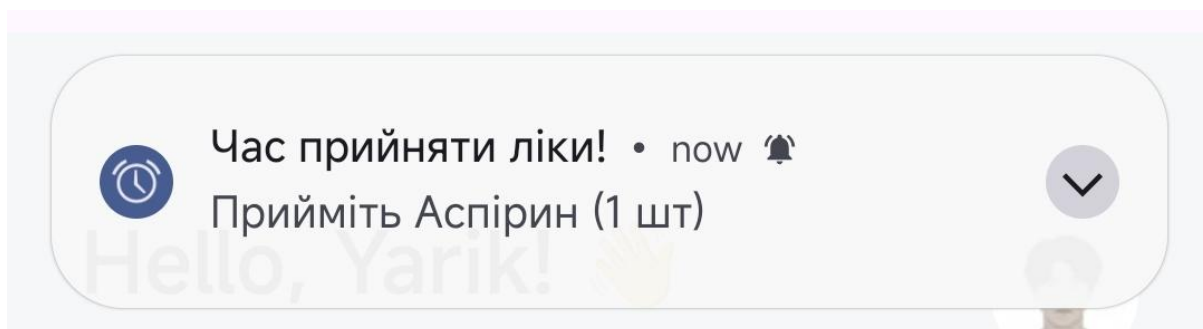


Рис. 3.4. Спрацювання системного нагадування

Для перевірки коректності агрегації історичних даних користувач здійснює перехід до модуля історії History Screen за допомогою нижньої панелі навігації. Вибравши поточну дату в горизонтальному календарі, він спостерігає деталізований запис про прийнятий препарат із точною фіксацією часу. Завершується контрольний приклад переходом до екрана звітів Reports Screen, де система, проаналізувавши записи в базі даних, автоматично розраховує та візуалізує загальний відсоток дотримання режиму лікування за допомогою

графічних віджетів, а також надає можливість експортувати ці дані у формат PDF.

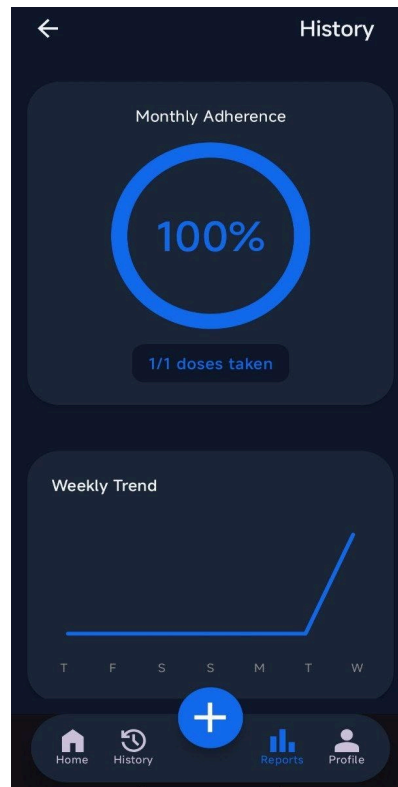


Рис. 3.5. Успішне підтвердження прийому ліків

Успішне виконання описаного наскрізного сценарію підтверджує абсолютну працездатність програмного комплексу. Контрольний приклад емпірично доводить, що модулі інтерфейсу, бізнес-логіки, бази даних та безпеки функціонують злагоджено, забезпечуючи виконання головної мети додатка – зручного та безпечного контролю за процесом медикаментозної терапії.

3.4. Системні вимоги до ПЗ

Для стабільного функціонування мобільного застосунку "BeWell" цільовий пристрій повинен відповідати наступним програмним вимогам:

- Операційна система: Android.
- Мінімальна версія SDK: API Level 24 (Android 7.0 Nougat).
- Цільова версія SDK: API Level 35 (Android 15)
- Необхідні системні дозволи: * POST_NOTIFICATIONS

3.5. Апаратні вимоги до ПЗ

Апаратне забезпечення мобільного пристрою має відповідати сучасним стандартам для забезпечення плавної взаємодії з анімаціями фреймворку Jetpack Compose та швидкого криптографічного перетворення даних:

- Процесор (CPU): пристрій на базі архітектури ARM64 (aarch64) із тактовою частотою від 1.5 ГГц.
- Оперативна пам'ять (RAM): мінімум 2 ГБ. Рекомендований обсяг – 4 ГБ або більше.
- Фізична пам'ять: близько 50 МБ вільного простору для встановлення базового APK файлу застосунку, а також щонайменше 20 МБ додаткового простору для локальної бази даних SQLite.

Висновки до розділу

З метою забезпечення стабільності та безвідмовності мобільного застосунку було застосовано багаторівневу стратегію тестування. За допомогою модульного тестування було верифіковано коректність роботи критично важливої бізнес-логіки системи – зокрема, математичних алгоритмів розрахунку часу наступного прийому препарату для складних терапевтичних графіків. UI-тестування підтвердило правильність відтворення реактивного інтерфейсу Jetpack Compose.

Значний обсяг досліджень у цьому розділі було присвячено питанням кібербезпеки та захисту персональних медичних даних пацієнта. Оскільки застосунок оперує чутливою інформацією про стан здоров'я, було реалізовано архітектуру, яка виключає несанкціонований доступ третіх осіб або передачу не зашифрованих даних на зовнішні сервери. Додатковим і вкрай важливим контуром захисту стала імплементація біометричної автентифікації за допомогою Android Biometric API. Це рішення дозволило захистити медичну інформацію на екрані блокування застосунку навіть у випадку фізичної втрати пристрою або передачі його стороннім особам.

Наприкінці розділу було чітко специфіковано системні та апаратні вимоги до цільових пристроїв користувачів. Проведена оптимізація споживання

оперативної пам'яті та ресурсів процесора гарантує плавну роботу додатку не лише на флагманських смартфонах, але й на бюджетних апаратах початкового рівня, якими найчастіше користуються люди похилого віку. За результатами проведених тестувань та перевірок безпеки можна стверджувати, що розроблена мобільна екосистема відповідає сучасним індустріальним стандартам.

ЗАГАЛЬНІ ВИСНОВКИ

У результаті виконання кваліфікаційної роботи бакалавра було успішно вирішено актуальне науково-практичне завдання – спроектовано та розроблено сучасну мобільну екосистему для відстеження та нагадування про прийом ліків на базі платформи Android. Розроблений програмний продукт спрямований на подолання кризи медичної адгерентності шляхом надання пацієнтам надійного, безпечного та зручного інструменту для контролю власної фармакотерапії.

У ході досягнення поставленої мети було виконано всі заплановані завдання та отримано такі результати:

1. Досліджено предметну область. Аналіз існуючих рішень виявив критичні недоліки популярних додатків. Це дозволило сформулювати чіткі вимоги до розроблюваного продукту, головними пріоритетами якого стали: офлайн-робота, відсутність збору метрик, інклюзивний дизайн та підтримка складних терапевтичних графіків.

2. Спроектовано надійну архітектуру. Застосування принципів Clean Architecture у поєднанні з шаблоном проектування MVVM дозволило розділити систему на незалежні шари. Такий підхід забезпечив високу гнучкість коду, полегшив процес тестування та створив міцний фундамент для майбутнього масштабування додатку.

3. Реалізовано інноваційний користувацький інтерфейс. Завдяки використанню новітнього декларативного UI-фреймворку Jetpack Compose, було створено плавний, реактивний та адаптивний інтерфейс. Було враховано особливості UX-дизайну для літніх людей.

4. Розроблено надійний механізм нагадувань та зберігання даних. Використання бібліотеки Room забезпечило швидку та надійну локальну роботу з базою даних. Для гарантованого спрацьовування нагадувань про прийом медикаментів було імплементовано алгоритми планування за допомогою AlarmManager та WorkManager, які були спеціально адаптовані під нові обмеження операційних систем Android 14 та Android 15.

5. Забезпечено високий рівень безпеки. Оскільки додаток оперує чутливою медичною інформацією, було впроваджено захист доступу до застосунку за допомогою біометричної автентифікації. Вся інформація обробляється та зберігається виключно на пристрої користувача.

Практичний результат: готовий до використання, кросфункціональний Android-застосунок, який повністю відповідає сучасним стандартам розробки програмного забезпечення від Google. Додаток не лише виконує функцію будильника для таблеток, але й веде історію прийому, відстежує запаси препаратів та надає користувачеві вичерпну статистику.

Подальший розвиток проекту може включати розробку версії для iOS, інтеграцію зі смарт-годинниками для більш зручних сповіщень, а також підключення до центральної бази даних здоров'я для автоматичного завантаження електронних рецептів від лікарів

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Як підключити МІС до центральної бази даних здоров'я eHealth [Електронний ресурс] // Medcenter CRM. – Режим доступу: <https://medcentercrm.com/blog/business/kak-podkluchit-mis-k-centralnoy-baze-dannih-zdorovya-ehealth/> (дата звернення: 01.03.2026).
2. 10 Best APIs for Digital Health in 2025 [Електронний ресурс] // Medtech Founder. – 2025. – Режим доступу: <https://medtechfounder.com> (дата звернення: 05.03.2026).
3. 5 Medication Reminder Apps That Are Actually Free (2026) [Електронний ресурс] // Pillo. – Режим доступу: <https://pillo.care/blog/best-free-medication-reminder-app> (дата звернення: 10.03.2026).
4. A scheduling algorithm - java [Електронний ресурс] // Stack Overflow. – 2012. – Режим доступу: <https://stackoverflow.com/q/10025132> (дата звернення: 15.03.2026).
5. Android Clean Architecture With MVVM Using Jetpack Compose [Електронний ресурс] // GitHub. – Режим доступу: <https://github.com/samadtalukder/Android-Clean-Architecture-MVVM-With-Compose> (дата звернення: 20.03.2026).
6. Behavior changes: all apps [Електронний ресурс] // Android Developers. – Режим доступу: <https://developer.android.com> (дата звернення: 25.03.2026).
7. Calendar heatmaps [Електронний ресурс] // Omni Docs. – Режим доступу: <https://docs.omni.co/showcase/visualizations/calendar-heatmaps> (дата звернення: 01.03.2026).
8. Compose UI Architecture [Електронний ресурс] // Jetpack Compose | Android Developers. – Режим доступу: <https://developer.android.com/develop/ui/compose/architecture> (дата звернення: 05.03.2026).
9. Creating a calendar heatmap chart (Github Contributions Like) in ReactJS

[Электронный ресурс] // Our Code World. – Режим доступа: <https://ourcodeworld.com/articles/read/563/creating-a-calendar-heatmap-chart-github-contributions-like-in-reactjs> (дата звернения: 10.03.2026).

10. Designing for Elderly Patients: UI/UX Tips [Электронный ресурс] // eSEOSpace. – Режим доступа: <https://eseospace.com/blog/designing-for-elderly-patients/> (дата звернения: 15.03.2026).

11. Diaz-Skeete Y. M., McQuaid D., Akinosun A. S. Analysis of Apps With a Medication List Functionality for Older Adults With Heart Failure Using the Mobile App Rating Scale // JMIR Mhealth Uhealth. – 2021. – Vol. 9(11). – P. e30674.

12. Dose App download [Электронный ресурс] // SourceForge.net. – Режим доступа: <https://sourceforge.net/projects/dose-app.mirror/> (дата звернения: 20.03.2026).

13. Flutter vs React Native: Which Is Better for Healthcare & Fitness Apps? [Электронный ресурс] // DEV Community. – Режим доступа: <https://dev.to/ciphernutz/flutter-vs-react-native-which-is-better-for-healthcare-fitness-apps-178> (дата звернения: 25.03.2026).

14. Full-screen intent limits [Электронный ресурс] // Android Open Source Project. – Режим доступа: <https://source.android.com/docs/core/permissions/fsi-limits> (дата звернения: 01.03.2026).

15. Futsch1/medTimer: MedTimer Android app [Электронный ресурс] // GitHub. – Режим доступа: <https://github.com/Futsch1/medTimer> (дата звернения: 05.03.2026).

16. GitHub - aritra-tech/MedSync: MedSync is an Android application [Электронный ресурс] // GitHub. – Режим доступа: <https://github.com/aritra-tech/MedSync> (дата звернения: 10.03.2026).

17. GitHub - shugo/medicine_shield: an Android medication [Электронный ресурс] // GitHub. – Режим доступа: https://github.com/shugo/medicine_shield (дата звернения: 15.03.2026).

18. Health Connect UI guidelines [Электронный ресурс] // Android health & fitness - Android Developers. – Режим доступа:

<https://developer.android.com/health-and-fitness/health-connect/ui/guidelines> (дата звернення: 20.03.2026).

19. Health Content and Services [Електронний ресурс] // Play Console Help. – Режим доступу: <https://support.google.com/googleplay/android-developer/answer/16679511?hl=en> (дата звернення: 25.03.2026).

20. Health app categories and additional information [Електронний ресурс] // Play Console Help - Google Help. – Режим доступу: <https://support.google.com/googleplay/android-developer/answer/13996367?hl=en> (дата звернення: 01.03.2026).

21. Heydari P. How to Encrypt Your Room Database in Android Using SQLCipher [Електронний ресурс] // Medium / ProAndroidDev. – Режим доступу: <https://proandroiddev.com/how-to-encrypt-your-room-database-in-android-using-sqlcipher-0bce78328bd6> (дата звернення: 05.03.2026).

22. HIPAA and GDPR Compliance for Health App Developers [Електронний ресурс] // LLIF.org. – Режим доступу: <https://llif.org/2025/01/31/hipaa-gdpr-compliance-health-apps/> (дата звернення: 10.03.2026).

23. HIPAA vs. GDPR Compliance: What's the Difference? [Електронний ресурс] // OneTrust Blog. – Режим доступу: <https://www.onetrust.com/blog/hipaa-vs-gdpr-compliance/> (дата звернення: 15.03.2026).

24. HIPAA-Compliant App Development: Rules, Security Measures, and Best Practices [Електронний ресурс] // Knack. – Режим доступу: <https://www.knack.com/blog/hipaa-compliant-app-development/> (дата звернення: 20.03.2026).

25. How the Digital Register of Medicinal Products is Transforming Ukraine's Healthcare System? [Електронний ресурс] // Digital State UA. – Режим доступу: <https://digitalstate.gov.ua/> (дата звернення: 25.03.2026).

26. How to Create a Medicine Tracker Android App with Firebase? [Електронний ресурс] // GeeksforGeeks. – Режим доступу: <https://www.geeksforgeeks.org/android/how-to-create-a-medicine-tracker-android-app-with-firebase/>

(дата звернення: 01.03.2026).

27. How to customize calendar view in jetpack compose? [Електронний ресурс] // Stack Overflow. – Режим доступу: <https://stackoverflow.com/questions/71854768/how-to-customize-calendar-view-in-jetpack-compose> (дата звернення: 05.03.2026).

28. How to Use Room Database with Kotlin [Електронний ресурс] // OneUptime. – Режим доступу: <https://oneuptime.com/blog/post/2026-02-02-kotlin-room-database/view> (дата звернення: 10.03.2026).

29. How would one represent scheduled events in an RDBMS? [Електронний ресурс] // Stack Overflow. – Режим доступу: <https://stackoverflow.com/questions/1016170/how-would-one-represent-scheduled-events-in-an-rdbms> (дата звернення: 15.03.2026).

30. I built an AI-powered medication reminder app - just got approved on the App Store [Електронний ресурс] // r/iOSProgramming - Reddit. – Режим доступу: <https://www.reddit.com/r/iOSProgramming/> (дата звернення: 20.03.2026).

31. Ideal Architecture for Jetpack Compose: MVVM + Clean Architecture Explained [Електронний ресурс] // Reddit. – Режим доступу: <https://www.reddit.com/r/Kotlin/> (дата звернення: 25.03.2026).

32. Key Insights for Developing a Medication Reminder App [Електронний ресурс] // Exoft. – Режим доступу: <https://www.exoft.net/blog/medication-reminder-app-development/> (дата звернення: 01.03.2026).

33. Lemos N., Machado C. S., Cardoso C. Free apps and paid apps: monetization strategies for health apps in the Portuguese market // International Journal of Pharmaceutical and Healthcare Marketing. – 2024. – Vol. 18, No. 2. – Режим доступу: <https://doi.org/10.1108/IJPHM-01-2023-0001> (дата звернення: 05.03.2026).

34. MedAlert - Medication Reminder & Health Assistant-Flutter template [Електронний ресурс] // InitioTechMedia.com. – Режим доступу: <https://www.initiotechmedia.com> (дата звернення: 10.03.2026).

35. Medication Management App Development: Comprehensive Guide

[Электронный ресурс] // Orangesoft. – Режим доступа: <https://orangesoft.co/blog/guide-to-medication-management-app-development> (дата звернения: 15.03.2026).

36. Medication Management Apps: A Digital Solution for Medication Adherence and Safety [Электронный ресурс] // DelveInsight. – Режим доступа: <https://www.delveinsight.com/blog/medication-management-apps> (дата звернения: 20.03.2026).

37. Medication tracker app suggestions? [Электронный ресурс] // r/ADHDDUK - Reddit. – Режим доступа: <https://www.reddit.com/r/ADHDDUK/> (дата звернения: 25.03.2026).

38. Ministry of Health updates state register of medicinal products [Электронный ресурс] // Cabinet of Ministers of Ukraine. – Режим доступа: <https://www.kmu.gov.ua> (дата звернения: 01.03.2026).

39. Monetization For Healthcare Apps [Электронный ресурс] // Meegle. – Режим доступа: https://www.meegle.com/en_us/topics/monetization-models/monetization-for-healthcare-apps (дата звернения: 05.03.2026).

40. My medication tracking app just got approved after 3 App Store rejections [Электронный ресурс] // Reddit. – Режим доступа: https://www.reddit.com/r/SideProject/comments/1rjmjua/my_medication_tracking_app_just_got_approved/ (дата звернения: 10.03.2026).

41. Persist data with Room [Электронный ресурс] // Android Developers. – Режим доступа: <https://developer.android.com/codelabs/basic-android-kotlin-compose-persisting-data-room> (дата звернения: 15.03.2026).

42. Pharmacy App Development: Complete Guide for 2025 [Электронный ресурс] // JetBase. – Режим доступа: <https://jetbase.io/blog/pharmacy-app-development-complete-guide-for-2025> (дата звернения: 20.03.2026).

43. PIN Screen with Biometric Auth in Jetpack Compose [Электронный ресурс] // Francesc Vilariño. – Режим доступа: <https://www.francescvilarino.com/pin-screen-with-biometric-auth> (дата звернения: 25.03.2026).

44. React Native vs Flutter for Healthcare Apps: I've Shipped Both, Here's What I'd Pick [Електронний ресурс]. – Режим доступу: <https://aarondsilva.me/blog/react-native-vs-flutter-healthcare-apps/> (дата звернення: 01.03.2026).

45. Reform of the Regulatory System in the Pharmaceutical Sector [Електронний ресурс] // UMA. – Режим доступу: <https://uma.gov.ua/en> (дата звернення: 05.03.2026).

46. Schedule that runs every day at a specific time with hh:mm:ss [Електронний ресурс] // Stack Overflow. – Режим доступу: <https://stackoverflow.com/questions/52317863> (дата звернення: 10.03.2026).

47. Scheduling Background Tasks in Android - WorkManager vs AlarmManager vs JobScheduler [Електронний ресурс] // r/JetpackComposeDev - Reddit. – Режим доступу: https://www.reddit.com/r/JetpackComposeDev/comments/1nwa26t/scheduling_background_tasks_in_android/ (дата звернення: 15.03.2026).

48. Singh A. Best Medication Reminder App Development for Seniors [Електронний ресурс] // Taction Software. – 2025. – Режим доступу: <https://www.tactionsoft.com/> (дата звернення: 20.03.2026).

49. Task scheduling. Background work [Електронний ресурс] // Android Developers. – Режим доступу: <https://developer.android.com/develop/background-work/background-tasks/persistent> (дата звернення: 25.03.2026).

50. Top 10 Medication Adherence Apps: Features, Pros, Cons & Comparison [Електронний ресурс] // DevOpsSchool. – 2025-2026. – Режим доступу: <https://www.devopsschool.com> (дата звернення: 01.03.2026).

51. Ukrainian Health Ministry updates medicinal products register [Електронний ресурс] // United Nations Development Programme (UNDP). – 2025. – Режим доступу: <https://www.undp.org/ukraine/press-releases/ukrainian-health-ministry-updates-medicinal-products-register> (дата звернення: 05.03.2026).

52. Ukrainian medical professionals pilot Estonian digital drug interaction platform [Електронний ресурс] // ERR News. – 2024. – Режим доступу:

<https://news.err.ee/1609552852/ukrainian-medical-professionals-pilot-estonian-digital-drug-interaction-platform> (дата звернення: 10.03.2026).

53. UX Design for Seniors: Guide to Healthcare App Design for Elderly...
[Електронний ресурс] // Orangesoft. – Режим доступу:
<https://orangesoft.co/blog/guide-to-designing-healthcare-apps-for-seniors> (дата
звернення: 15.03.2026).

ДОДАТКИ

Додаток А Діаграма послідовності взаємодії компонентів системи

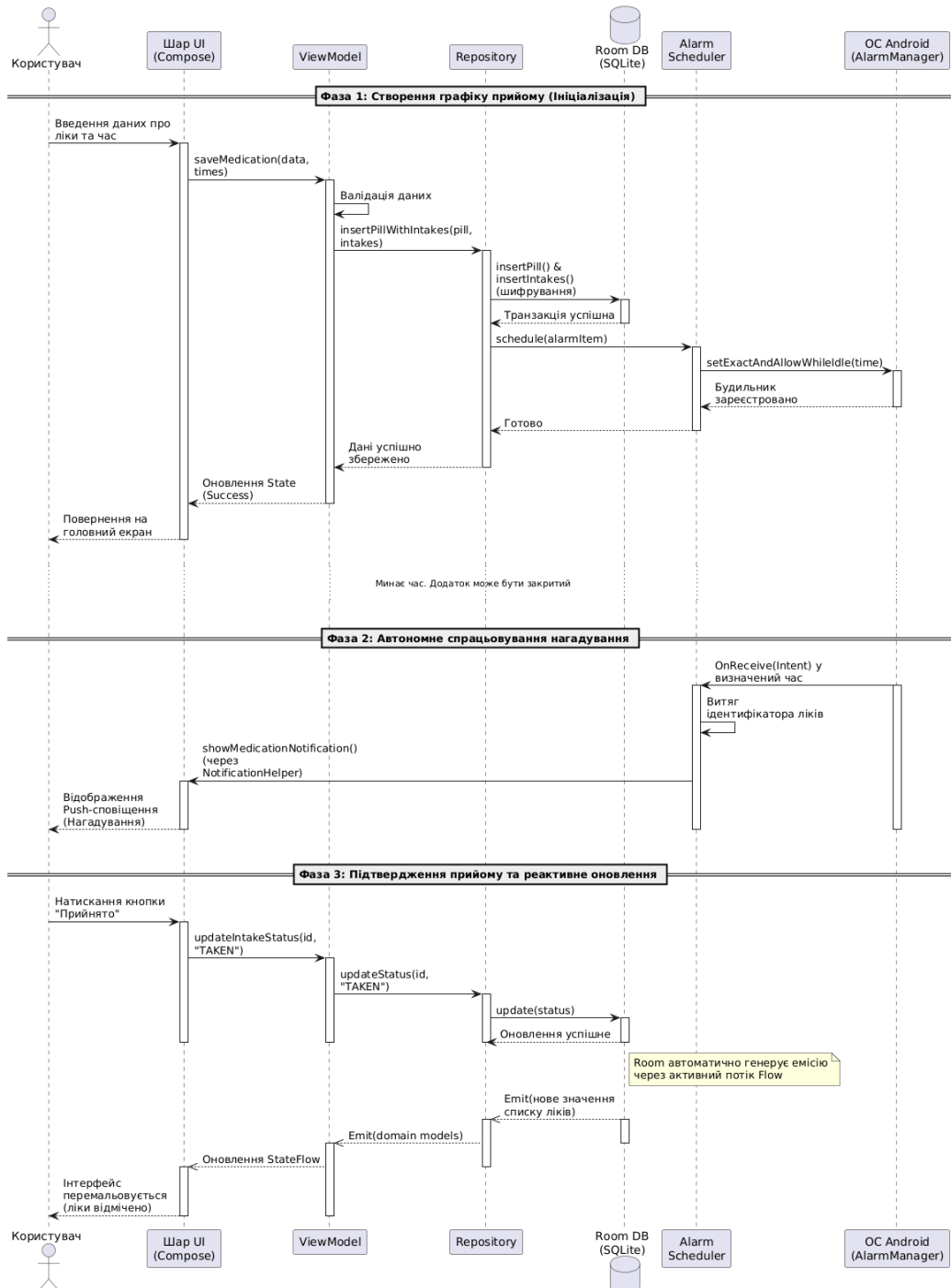


Рис. А.1 Діаграма послідовності взаємодії компонентів системи

Додаток Б Лістинг HomeDashboardScreen

<https://github.com/Avdwd/BeWell/blob/main/app/src/main/java/com/avdwd/be/well/ui/home/HomeDashboardScreen.kt>

Додаток Б.1 Лістинг DateDivider

<https://github.com/Avdwd/BeWell/blob/main/app/src/main/java/com/avdwd/be%20well/ui/home/components/HomeDashboardDate.kt>

Додаток Б.2 Лістинг FilterSection

<https://github.com/Avdwd/BeWell/blob/main/app/src/main/java/com/avdwd/be/well/ui/home/components/HomeDashboardFilters.kt>

Додаток Б.3 Лістинг SwipeableMedicationItem

<https://github.com/Avdwd/BeWell/blob/main/app/src/main/java/com/avdwd/be%20well/ui/home/components/SwipeableMedicationItem.kt>

Додаток Б.4 Лістинг HomeViewModel

<https://github.com/Avdwd/BeWell/blob/main/app/src/main/java/com/avdwd/be%20well/ui/home/HomeViewModel.kt>

Додаток В Лістинг AddMedicationScreen

<https://github.com/Avdwd/BeWell/blob/main/app/src/main/java/com/avdwd/be%0Awell/ui/addmedication/AddMedicationScreen.kt>

Додаток В.1 Лістинг InputTextField

<https://github.com/Avdwd/BeWell/blob/main/app/src/main/java/com/avdwd/be%0Awell/ui/addmedication/components/InputTextField.kt>

Додаток В.2 Лістинг MedicationType

<https://github.com/Avdwd/BeWell/blob/main/app/src/main/java/com/avdwd/be%20well/ui/addmedication/components/MedicationType.kt>

Додаток В.3 Лістинг CustomTimePicker

[https://github.com/Avdwd/BeWell/blob/main/app/src/main/java/com/avdwd/be
well/ui/addmedication/components/CustomTimePicker.kt](https://github.com/Avdwd/BeWell/blob/main/app/src/main/java/com/avdwd/be%20well/ui/addmedication/components/CustomTimePicker.kt)

Додаток В.4 Лістинг AddMedicationViewModel

https://github.com/Avdwd/BeWell/blob/main/app/src/main/java/com/avdwd/be_well/ui/addmedication/AddMedicationViewModel.kt

Додаток Г Лістинг HistoryScreen

<https://github.com/Avdwd/BeWell/blob/main/app/src/main/java/com/avdwd/be/well/ui/history/HistoryScreen.kt>

Додаток Г.1 Лістинг HorizontalCalendar

https://github.com/Avdwd/BeWell/blob/main/app/src/main/java/com/avdwd/be_well/ui/history/components/HorizontalCalendar.kt

Додаток Г.2 Лістинг TimelineItem

[https://github.com/Avdwd/BeWell/blob/main/app/src/main/java/com/avdwd/be
well/ui/history/components/TimeLineItem.kt](https://github.com/Avdwd/BeWell/blob/main/app/src/main/java/com/avdwd/be
well/ui/history/components/TimeLineItem.kt)

Додаток Г.3 Лістинг HistoryScreenViewModel

<https://github.com/Avdwd/BeWell/blob/main/app/src/main/java/com/avdwd/be%0Awell/ui/history/HistoryScreenViewModel.kt>

Додаток Г.4 Лістинг HistoryMedicationCard

https://github.com/Avdwd/BeWell/blob/main/app/src/main/java/com/avdwd/be_well/ui/history/components/HistoryMedicadionCard.kt

Додаток Д Лістинг ReportsScreen

<https://github.com/Avdwd/BeWell/blob/main/app/src/main/java/com/avdwd/be/well/ui/reports/ReportsScreen.kt>

Додаток Д.1 Лістинг MonthlyAdherenceCard

https://github.com/Avdwd/BeWell/blob/main/app/src/main/java/com/avdwd/be_well/ui/reports/components/MonthlyAdherenceCard.kt

Додаток Д.2 Лістинг WeeklyTrendCard

<https://github.com/Avdwd/BeWell/blob/main/app/src/main/java/com/avdwd/be%0Awell/ui/reports/components/WeeklyTrend.kt>

Додаток Д.3 Лістинг ExportCard

<https://github.com/Avdwd/BeWell/blob/main/app/src/main/java/com/avdwd/be%0Awell/ui/reports/components/ExportCard.kt>

Додаток Е Лістинг ProfileScreen

<https://github.com/Avdwd/BeWell/blob/main/app/src/main/java/com/avdwd/be/well/ui/profile/ProfileScreen.kt>

Додаток Е.1 Лістинг ProfileMenuItem

https://github.com/Avdwd/BeWell/blob/main/app/src/main/java/com/avdwd/be_well/ui/profile/components/ProfileMenuItem.kt

Додаток Е.2 Лістинг ProfileMenuSection

<https://github.com/Avdwd/BeWell/blob/main/app/src/main/java/com/avdwd/be%0022well/ui/profile/components/ProfileMenuSection.kt>

Додаток Ж Лістинг AppNavigation

<https://github.com/Avdwd/BeWell/blob/main/app/src/main/java/com/avdwd/be%0Awell/ui/navigation/AppNavigation.kt>

Додаток 3 Лістинг BeWellBottomBar

<https://github.com/Avdwd/BeWell/blob/main/app/src/main/java/com/avdwd/be%20well/ui/bottombar/BottomBar.kt>

Додаток К Лістинг AuthWrapper

[https://github.com/Avdwd/BeWell/blob/main/app/src/main/java/com/avdwd/be
well/ui/auth/AuthWrapper.kt](https://github.com/Avdwd/BeWell/blob/main/app/src/main/java/com/avdwd/be
well/ui/auth/AuthWrapper.kt)

Додаток Л Лістинг MedicationRepository

[https://github.com/Avdwd/BeWell/blob/main/app/src/main/java/com/avdwd/be
well/data/repository/MedicationRepository.kt](https://github.com/Avdwd/BeWell/blob/main/app/src/main/java/com/avdwd/be
well/data/repository/MedicationRepository.kt)

Додаток М Лістинг PillEntity

<https://github.com/Avdwd/BeWell/blob/main/app/src/main/java/com/avdwd/be%0Awell/data/local/entity/PillEntity.kt>

Додаток Н Лістинг IntakeRecordEntity

<https://github.com/Avdwd/BeWell/blob/main/app/src/main/java/com/avdwd/be%0Awell/data/local/entity/IntakeRecordEntity.kt>

Додаток П Лістинг IntakeWithPill

[https://github.com/Avdwd/BeWell/blob/main/app/src/main/java/com/avdwd/be
well/data/local/entity/IntakeWithPill.kt](https://github.com/Avdwd/BeWell/blob/main/app/src/main/java/com/avdwd/be
well/data/local/entity/IntakeWithPill.kt)

Додаток Р Лістинг AlarmScheduler

<https://github.com/Avdwd/BeWell/blob/main/app/src/main/java/com/avdwd/be%0Awell/notifications/AlarmScheduler.kt>

Додаток С Лістинг AlarmSchedulerImpl

<https://github.com/Avdwd/BeWell/blob/main/app/src/main/java/com/avdwd/be%0Awell/notifications/AlarmSchedulerImpl.kt>

Додаток Т Лістинг PillDao

<https://github.com/Avdwd/BeWell/blob/main/app/src/main/java/com/avdwd/be/well/data/local/dao/PillDao.kt>

Додаток У Лістинг IntakeRecordDao

<https://github.com/Avdwd/BeWell/blob/main/app/src/main/java/com/avdwd/be%20well/data/local/dao/IntakeRecordDao.kt>

Додаток Ф Лістинг BeWellRoomDB

<https://github.com/Avdwd/BeWell/blob/main/app/src/main/java/com/avdwd/be%0Awell/data/local/BeWellRoomDB.kt>

Додаток X Лістинг EncryptionManager

[https://github.com/Avdwd/BeWell/blob/main/app/src/main/java/com/avdwd/be
well/data/local/encryption/EncryptionManager.kt](https://github.com/Avdwd/BeWell/blob/main/app/src/main/java/com/avdwd/be
well/data/local/encryption/EncryptionManager.kt)

Додаток Ц Лістинг MedicationAlarmReceiver

<https://github.com/Avdwd/BeWell/blob/main/app/src/main/java/com/avdwd/be%0Awell/notifications/MedicationAlarmReceiver.kt>

Додаток III Лістинг NotificationHelper

https://github.com/Avdwd/BeWell/blob/main/app/src/main/java/com/avdwd/be_well/services/NotificationHelper.kt

Додаток Щ Лістинг LockScreen

https://github.com/Avdwd/BeWell/blob/main/app/src/main/java/com/avdwd/be_well/ui/auth/LockScreen.kt

Міністерство освіти і науки України
Державний заклад «Луганський національний університет імені
Тараса Шевченка»

Факультет
(інститут)

Факультет технологій та інформаційних
систем

Кафедра:

Інформаційних технологій та систем

ПРОГРАМА ТА МЕТОДИКА ТЕСТУВАННЯ

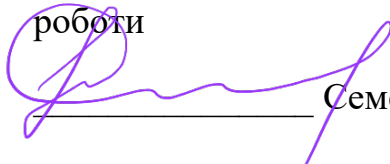
на виконання програмної розробки (ПР):

**«СТВОРЕННЯ МОБІЛЬНОГО ЗАСТОСУНКУ ДЛЯ ВІДСТЕЖЕННЯ
ТА НАГАДУВАННЯ ПРО ПРИЙОМ ЛІКІВ»**

ІТС.ІПЗ4.22seg61121-04-ПМТ

ПОГОДЖЕНО

Керівник кваліфікаційної
роботи

 Семенов М.А.

«__» _____ 2026 р.

ВИКОНАВЕЦЬ

Студент групи 4 ІПЗ

 Шинкаренко Я.М.

«__» _____ 2026 р.

Лубни - 2026

ЗМІСТ

ВСТУП	3
1. ОБ'ЄКТ ТЕСТУВАННЯ	4
2. МЕТА ТЕСТУВАННЯ	4
3. ВИМОГИ ДО СЕРЕДОВИЩА ТЕСТУВАННЯ.....	4
4. МЕТОДИКА ТЕСТУВАННЯ	4
5. ВИСНОВКИ ЗА РЕЗУЛЬТАТАМИ ТЕСТУВАННЯ.....	6

ВСТУП

Цей документ описує програму та методику тестування мобільного застосунку для відстеження та нагадування про прийом ліків, робоча назва «BeWell». Документ встановлює вимоги до середовища, об'єкта тестування та визначає конкретні сценарії, за якими буде перевірятися працездатність, надійність та безпека системи.

1. ОБ'ЄКТ ТЕСТУВАННЯ

Об'єктом тестування є програмний код та скомпільований пакет APK/AAB мобільного застосунку «BeWell» для операційної системи Android.

2. МЕТА ТЕСТУВАННЯ

Метою тестування є перевірка:

- Функціональної повноти системи згідно з Технічним завданням.
- Надійності алгоритмів планування фонових нагадувань.
- Коректності роботи локального реляційного сховища даних.
- Верифікація механізмів криптографічного захисту та біометричної автентифікації.

3. ВИМОГИ ДО СЕРЕДОВИЩА ТЕСТУВАННЯ

3.1. Апаратне забезпечення:

Фізичний смартфон на базі архітектури ARM64 з наявним біометричним сканером.

Емулятор Android Studio AVD для тестування на різних розмірах екранів.

3.2. Програмне забезпечення:

Операційна система: Android версій 7.0 API 24, 13.0 API 33 та 15.0 API 35.

Фреймворки автоматизованого тестування: JUnit 4, Espresso / Compose Testing API.

4. МЕТОДИКА ТЕСТУВАННЯ

4.1. Модульне тестування рівня Домену

Сценарій 4.1.1: Валідація даних при додаванні ліків.

Дія: передача порожньої назви препарату до AddMedicationViewModel.

Очікуваний результат: функція валідації повертає false, кнопка збереження залишається неактивною, виклик до репозиторію блокується.

Сценарій 4.1.2: Коректність розрахунку часу.

Дія: запуск алгоритму планування для щоденного прийому препарату о 08:00.

Очікуваний результат: формується об'єкт AlarmItem із коректним Unix-timestamp наступного дня. Системний лог фіксує виклик методу setExactAndAllowWhileIdle.

4.2. Інтеграційне тестування Баз Даних

Сценарій 4.2.1: Збереження та вилучення реляційних даних.

Дія: ініціалізація In-Memory бази даних Room. Збереження PillEntity та запиту пов'язаного списку IntakeRecordEntity через DAO.

Очікуваний результат: запит повертає коректно сформований об'єкт IntakeWithPill із правильними зв'язками Foreign Keys. Дані не втрачаються під час асинхронних корутин-транзакцій.

4.3. Тестування Інтерфейсу

Сценарій 4.3.1: Змахування картки препарату.

Дія: імітація жесту змахування вправо на компоненті SwipeableMedicationItem на головному екрані.

Очікуваний результат: UI плавно відображає іконку «Видалити». Після завершення жесту стан компонента оновлюється,

відправляється Intent до ViewModel, препарат видаляється.

4.4. Тестування Безпеки

Сценарій 4.4.1: Стійкість локального шифрування.

Дія: Вилучення файлу бази даних .db через Device File Explorer у Android Studio на рутованому емуляторі. Спроба відкрити файл у DB Browser for SQLite.

Очікуваний результат: Файл відмовляється відкриватися, видаючи помилку «File is encrypted or is not a database». Відкритий текст відсутній.

Сценарій 4.4.2: Робота біометричної обгортки.

Дія: Запуск додатка «з нуля» або повернення з фоновому режиму onResumed.

Очікуваний результат: головний граф навігації блокується. На екрані з'являється BiometricPrompt. Медичні дані не рендеряться до отримання callback onAuthenticationSucceeded. При відмові – відображається статичний екран LockScreen.

5. ВИСНОВКИ ЗА РЕЗУЛЬТАТАМИ ТЕСТУВАННЯ

Тестування проведено в повному обсязі згідно з затвердженою методикою. Програмний продукт успішно пройшов усі сценарії, продемонстрував стабільну роботу фонових сервісів та надійний захист даних. Критичних збоїв чи витоків пам'яті не виявлено. Програмне забезпечення вважається придатним до експлуатації.

Міністерство освіти і науки України
Державний заклад «Луганський національний університет імені
Тараса Шевченка»

Факультет
(інститут)

Факультет технологій та інформаційних
систем

Кафедра:

Інформаційних технологій та систем

КЕРІВНИЦТВО АДМІНІСТРАТОРА / ПРОГРАМІСТА

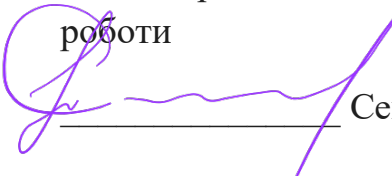
на виконання програмної розробки (ПР):

**«СТВОРЕННЯ МОБІЛЬНОГО ЗАСТОСУНКУ ДЛЯ ВІДСТЕЖЕННЯ
ТА НАГАДУВАННЯ ПРО ПРИЙОМ ЛІКІВ»**

ІТС.ІПЗ4.22seg61121-06-КА

ПОГОДЖЕНО

Керівник кваліфікаційної
роботи

 Семенов М.А.

«__» _____ 2026 р.

ВИКОНАВЕЦЬ

Студент групи 4 ІПЗ

 Шинкаренко Я.М.

«__» _____ 2026 р.

ЗМІСТ

1. АРХІТЕКТУРА СИСТЕМИ.....	4
2. НАЛАШТУВАННЯ СЕРЕДОВИЩА РОЗРОБКИ.....	4
3. РОБОТА З БАЗОЮ ДАНИХ ТА БЕЗПЕКОЮ.....	4
4. ФОНОВІ СЕРВІСИ ТА СПОВІЩЕННЯ	5
5. ЗБІРКА ПРОЄКТУ.....	6

ВСТУП

Цей документ призначений для розробників, системних адміністраторів та інженерів з підтримки програмного забезпечення, які будуть супроводжувати, модифікувати або розгортати вихідний код мобільного застосунку «BeWell». Керівництво містить інформацію про архітектурні особливості, налаштування середовища та криптографічні механізми.

1. АРХІТЕКТУРА СИСТЕМИ

Додаток розроблений виключно мовою **Kotlin** для платформи Android.

- **UI Шар:** повністю реалізований на декларативному фреймворку Jetpack Compose. Навігація побудована через NavHost.
- **Паттерн:** Використовується MVVM. Стан екрану передається з ViewModel до Compose через StateFlow.
- **Доменний шар:** Централізований у MedicationRepository, який виступає єдиним джерелом істини.
- **Ін'єкція залежностей:** Здійснюється за допомогою бібліотеки Hilt / Dagger.

2. НАЛАШТУВАННЯ СЕРЕДОВИЩА РОЗРОБКИ

Для компіляції та редагування вихідного коду необхідно:

- **IDE:** Android Studio версії Koala (2024.1.1) або новішої.
- **JDK:** Java Development Kit (JDK) 17.
- **Android SDK:** * compileSdk = 35
minSdk = 24 (Android 7.0)
targetSdk = 35
- Імпортуйте проєкт через build.gradle.kts та дочекайтеся завершення Gradle Sync.

3. РОБОТА З БАЗОЮ ДАНИХ ТА БЕЗПЕКОЮ

3.1. Локальне сховище:

Система використовує ORM Room. База складається з двох основних сутностей: PillEntity та IntakeRecordEntity.

3.2. Шифрування:

База даних шифрується під час виконання процесу за допомогою SQLCipher AES-256.

- Ключ шифрування генерується системою при першому запуску SecureRandom, 32 байти.
- Ключ зберігається у EncryptedSharedPreferences, який обгорнуто ключем з апаратного Android Keystore.
- Важливо! Неможливо переглянути вміст .db файлу через ADB без виконання дампа Keystore пристрою. У маніфесті встановлено android:allowBackup="false", що запобігає витоку БД через хмарні бекапи Google.

3.3. Біометрія:

Для тестування біометрії на емуляторі Android Studio, перейдіть до налаштувань емулятора, три крапки -> Security та налаштуйте віртуальний відбиток пальця.

4. ФОНОВІ СЕРВІСИ ТА СПОВІЩЕННЯ

Додаток не використовує сторонні Push-сервіси. Усі сповіщення плануються локально.

- Використовується системний AlarmManager з методом setExactAndAllowWhileIdle() для обходу обмежень режиму Doze.
- У класі AndroidManifest.xml обов'язково задекларовано дозвіл USE_EXACT_ALARM та SCHEDULE_EXACT_ALARM. При розгортанні на Android 14+, користувач може бути зобов'язаний надати цей дозвіл вручну в налаштуваннях ОС.

5. ЗБІРКА ПРОЄКТУ

Перед формуванням релізного APK/AAB файлу:

1. Переконайтеся, що налаштовані правила ProGuard / R8 у файлі `proguard-rules.pro`.
2. Критично важливо зберегти класи сутностей Room та SQLCipher:

```
-keep class net.sqlcipher. { *; }  
-keepclassmembers class * extends  
    androidx.room.RoomDatabase { *; }
```
3. Збірка здійснюється через меню Android Studio: Build -> Generate Signed Bundle / APK... з використанням вашого власного keystore-файлу розробника.

Міністерство освіти і науки України
Державний заклад «Луганський національний університет імені
Тараса Шевченка»

Факультет
(інститут)

Факультет технологій та інформаційних
систем

Кафедра:

Інформаційних технологій та систем

КЕРІВНИЦТВО КОРИСТУВАЧА

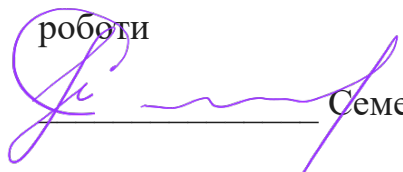
на виконання програмної розробки (ПР):

**«СТВОРЕННЯ МОБІЛЬНОГО ЗАСТОСУНКУ ДЛЯ ВІДСТЕЖЕННЯ
ТА НАГАДУВАННЯ ПРО ПРИЙОМ ЛІКІВ»**

ІТС.ІПЗ4.22seg61121-05-КК

ПОГОДЖЕНО

Керівник кваліфікаційної
роботи

 Семенов М.А.

«__» _____ 2026 р.

ВИКОНАВЕЦЬ

Студент групи 4 ІПЗ

 Шинкаренко Я.М.

«__» _____ 2026 р.

ЗМІСТ

ВСТУП	3
1. ПІДГОТОВКА ДО РОБОТИ З ДОДАТКОМ.....	4
2. РОБОТА З ДОДАТКОМ	5

ВСТУП

Мобільний застосунок «BeWell» призначений для автоматизації контролю за прийомом медикаментів та підвищення медичної дотримання рекомендацій лікаря.

Цей застосунок є самостійною програмою для платформи Android, яка має інтуїтивно зрозумілий та інклюзивний інтерфейс, простий у використанні навіть для людей похилого віку.

У цій програмі ви можете створити особистий розклад прийому ліків, отримувати гарантовані нагадування у визначений час та вести захищену локальну історію свого лікування без передачі даних у мережу Інтернет.

1. ПІДГОТОВКА ДО РОБОТИ З ДОДАТКОМ

Вимоги до пристрою:

Для коректної роботи додатку необхідний смартфон під управлінням операційної системи Android версії 7.0 або вище. Для забезпечення заявленого рівня безпеки медичних даних на пристрої має бути налаштовано екранне блокування, PIN-код, графічний ключ або біометричний захист сканер відбитка пальця.

Встановлення:

Встановлення додатку здійснюється з інсталяційного пакета (APK).

1. Збережіть файл встановлення BeWell-release.apk у пам'ять вашого смартфона.
2. Відкрийте системні налаштування смартфона, перейдіть до розділу «Безпека» та дозвольте встановлення додатків з невідомих джерел.
3. За допомогою файлового менеджера знайдіть завантажений файл та натисніть на нього для початку інсталяції.
4. Після завершення встановлення іконка додатку «BeWell» з'явиться на робочому столі або в меню програм вашого пристрою.

Надання дозволів:

При першому запуску програма запитає дозвіл на:

- Надсилання сповіщень, обов'язково для Android 13 та вище, щоб ви не пропустили час прийому.
- Встановлення точних будильників для Android 12 та вище, щоб

нагадування спрацьовували секунда в секунду.

Для забезпечення надійної роботи застосунку необхідно надати ці дозволи.

2. РОБОТА З ДОДАТКОМ

2.1. Запуск та автентифікація

З метою збереження медичної таємниці, вхід до додатку захищено. При кожному запуску або поверненні до згорнутої програми на екрані з'явиться системне вікно автентифікації.

Прикладіть палець до сканера відбитків або введіть PIN-код вашого пристрою. Тільки після успішної ідентифікації ваші дані розшифровуються та відображаються на екрані.

2.2. Інтерфейс головного екрана

Взаємодія з додатком здійснюється за допомогою дотиків до екрана смартфона.

Після авторизації ви потрапляєте на Головний екран. Тут відображається перелік препаратів, які вам потрібно прийняти сьогодні.

У нижній частині екрана розташована панель навігації для перемикання між основними розділами:

- **Сьогодні:** ваш розклад на поточну добу.
- **Історія:** календар та перелік прийнятих/пропущених препаратів за минулі дні.
- **Налаштування:** керування профілем та загальними параметрами.

2.3. Додавання нових ліків

Для того, щоб програма почала нагадувати про ліки, необхідно їх додати до розкладу:

1. На Головному екрані натисніть плаваючу кнопку з символом «+» у нижньому правому куті.
 2. Відкриється вікно «Додати препарат». У текстове поле введіть назву ліків наприклад, «Но-шпа».
 3. Виберіть форму випуску, натиснувши на відповідну іконку таблетки, капсули, сироп.
 4. Вкажіть дозування наприклад, «1 таб.» або «5 мл».
 5. Натисніть на поле «Час прийому». З'явиться системний годинник, на якому потрібно обрати час. Якщо ліки треба пити кілька разів на день, натисніть кнопку «Додати час» і вкажіть ще один.
 6. Перевірте введені дані та натисніть кнопку «Зберегти».
- Препарат з'явиться у вашому розкладі на сьогодні.

2.4. Взаємодія зі сповіщеннями

Коли настає час прийому ліків, ваш смартфон відтворить звуковий сигнал та вібрацію, а на екрані з'явиться системне сповіщення від «BeWell».

Натисніть на сповіщення, щоб швидко відкрити додаток.

На Головному екрані знайдіть картку потрібного препарату. Щоб підтвердити, що ви випили ліки:

- Натисніть пальцем по картці препарату зліва направо.
- З'явиться іконка підтвердження, колір картки зміниться, і статус препарату буде збережено як «Прийнято».

2.5. Перегляд історії прийомів

Щоб перевірити, чи не забули ви прийняти ліки вчора, або показати статистику лікарю:

1. Натисніть на іконку «Історія» у нижній навігаційній панелі.
2. У верхній частині екрана ви побачите календар (стрічку днів).
Натисніть на потрібну дату.
3. Нижче з'явиться список ліків, запланованих на той день, із зазначенням їхнього статусу (зелений — прийнято, сірий/червоний — пропущено).