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

КВАЛІФІКАЦІЙНА РОБОТА

ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «МАГІСТР»

РОЗРОБКА ТА ДОСЛІДЖЕННЯ МОБІЛЬНОГО ДОДАТКА ДЛЯ НАГАДУВАННЯ ПРО ПРИЙМАННЯ ЛІКІВ

DEVELOPMENT AND RESEARCH OF A MOBILE APPLICATION FOR REMINDING ABOUT TAKING MEDICATION

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

Виконав: здобувач вищої освіти
групи ІПЗм-21
Шворак Д. М.
Керівник:
к.ф.-м.н., доцент
Хвищун М. В.

Кваліфікаційну роботу
допущено до захисту
«___» _____ 20__ р.
Гарант освітньої програми:
к.т.н., доцент Суринович О. М.
_____

Луцьк – 2025 року

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет *комп'ютерних та інформаційних технологій*
Кафедра *інженерії програмного забезпечення*
Ступінь вищої освіти *магістр*
Галузь знань: *12 «Інформаційні технології»*
Спеціальність: *121 «Інженерія програмного забезпечення»*
Освітня програма: «*Інженерія програмного забезпечення»*

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____
«___» _____ 202__ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧА

ДРУГОГО (МАГІСТЕРСЬКОГО) РІВНЯ ВИЩОЇ ОСВІТИ

Швораку Дмитру Михайловичу

1. Тема кваліфікаційної роботи: Розробка та дослідження мобільного додатка для нагадування про приймання ліків
Керівник роботи: Хвищун Микола Вячеславович, доцент, к.ф.-м.н.
_____
затверджені наказом закладу вищої освіти від «29» березня 2025 року № 190/01-02
2. Строк подання здобувачем вищої освіти кваліфікаційної роботи: 4 грудня 2025 р.
3. Вихідні дані до роботи технічне та програмне забезпечення ЕОМ
4. Зміст розрахунково-пояснювальної записки: аналіз стану проблеми управління нагадуваннями в інформаційних системах, огляд методів та засобів розробки програмного забезпечення, проєктування і реалізацію мобільного додатка, а також аналіз та оцінку ефективності отриманих результатів
5. Перелік графічного матеріалу 13 рисунків, 6 таблиць, 4 лістинги коду. 1 додаток.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
Аналіз проблеми за темою роботи та постановка завдань дослідження	*Хвищун М. В.*		
Теоретичне дослідження та практична реалізація	*Хвищун М. В.*		
Експериментальне дослідження системи	*Хвищун М. В.*		
Нормоконтроль	*Повстяна Ю. С.*		
Гарант ОП	*Андрущак І. Є.*		
Показник запозичень тексту	***___%***		
Академічна доброчесність	*Хвищун М. В.*		

7. Дата видачі завдання *«02» квітня 2025 р.*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи магістра	Строк виконання етапів роботи	Примітка
1	Провести огляд літературних джерел по темі кваліфікаційної роботи	02.05.2025	
2	Провести аналіз загальної проблеми і вибір напрямків дослідження	24.09.2025	
3	Розробити функціональну модель та архітектуру системи	01.11.2025	
4	Описати засоби розробки об'єкта проектування	19.11.2025	
5	Практична реалізація об'єкта проектування	26.11.2025	
6	Розробити методику для проведення експерименту	05.11.2025	
7	Провести аналіз результатів експерименту	15.11.2025	
8	Здача чистового варіанту кваліфікаційної роботи на кафедру	04.12.2025	

Здобувач вищої освіти _____ Шворак Д. М.

Керівник кваліфікаційної роботи _____ Хвищун М. В.

АНОТАЦІЯ

Шворак Д. М. Розробка та дослідження мобільного додатка для нагадування про приймання ліків. Рукопис.

Кваліфікаційна робота магістра ОП «Інженерія програмного забезпечення» спеціальності 121 «Інженерія програмного забезпечення». Луцький національний технічний університет. Луцьк, 2025.

Кваліфікаційна робота магістра складається з вступу, 3 розділів, висновків, списку використаних джерел та додатку.

У роботі проведено аналіз предметної області, розглянуто існуючі методи та засоби розробки подібних програмних продуктів. Обґрунтовано вибір технологій для реалізації системи. Розроблений мобільний додаток забезпечує планування графіка прийому ліків, сповіщення користувача у визначений час, збереження історії лікування та перегляд статистики виконання. Проведено тестування функціональності, оцінено якість роботи системи та підтверджено її відповідність поставленим вимогам.

Ключові слова: мобільний додаток, нагадування, прийом ліків, Android, Kotlin, Jetpack Compose, Room, AlarmManager, інформаційна система.

ABSTRACT

Shvorak D. M. Development and Research of a Mobile Application for Reminding About Taking Medication. Manuscript.

Qualification master's thesis OP «Software Engineering» specialty 121 «Software Engineering». Lutsk National Technical University. Lutsk, 2025.

Qualification master's thesis consists of an introduction, 3 chapters, conclusions, a list of sources used, and an appendix.

The study analyzes the subject area, reviews existing methods and tools for developing similar software products, and justifies the choice of technologies used for the system implementation. The developed mobile application enables scheduling medication intake, sending timely notifications to the user, saving treatment history, and viewing performance statistics. Functionality testing was carried out, the system's performance quality was evaluated, and its compliance with the specified requirements was confirmed.

Keywords: mobile application, reminders, medication intake, Android, Kotlin, Jetpack Compose, Room, AlarmManager, information system.

ЗМІСТ

ВСТУП

Актуальність теми дослідження. Сучасні тенденції розвитку охорони здоров'я дедалі більше орієнтуються на цифровізацію та індивідуалізацію процесів лікування. Однією з найважливіших проблем залишається забезпечення дотримання пацієнтами призначеного режиму терапії. За даними Всесвітньої організації охорони здоров'я, майже половина хворих не дотримується встановленого графіка прийому ліків, що призводить до зниження ефективності лікування, виникнення ускладнень і зростання витрат на медичну допомогу.

У контексті поширення мобільних технологій значного розвитку набули програмні засоби, що підтримують концепцію mHealth – мобільного здоров'я. Мобільні додатки для моніторингу, нагадування й аналізу прийому ліків виступають інструментом підвищення самодисципліни користувачів, сприяють профілактиці хронічних захворювань і оптимізують взаємодію пацієнта з лікарем. Водночас більшість наявних рішень мають низку обмежень – недостатню інтуїтивність інтерфейсу, відсутність локального збереження даних, обмежені можливості персоналізації або нестабільну роботу в автономному режимі.

Розвиток інженерії програмного забезпечення та сучасних технологій Android, зокрема Kotlin, Jetpack Compose, Room ORM і AlarmManager, створює передумови для проєктування високоякісних мобільних систем, що поєднують функціональність, надійність і простоту використання. Створення такого застосунку має не лише технічне, а й соціальне значення – воно сприяє покращенню якості життя користувачів, підтримці регулярності лікування й розвитку цифрової культури турботи про власне здоров'я.

Таким чином, актуальність теми дослідження зумовлена необхідністю розроблення сучасного мобільного програмного засобу, який би забезпечував ефективний контроль за прийомом лікарських препаратів, відповідав принципам зручності, безпеки та автономності, а також інтегрував сучасні підходи програмної інженерії у сфері охорони здоров'я.

Метою дослідження є розробка та дослідження програмного забезпечення у вигляді мобільного додатка для Android, що забезпечує нагадування користувачеві про приймання лікарських засобів, а також зберігання, перегляд і аналіз історії лікування.

Об'єктом дослідження кваліфікаційної роботи є процес розроблення та функціонування мобільних інформаційних систем, призначених для організації персональних нагадувань і моніторингу дотримання графіка приймання ліків.

Предметом дослідження кваліфікаційної роботи є методи, алгоритми, архітектурні рішення та програмні засоби, що забезпечують реалізацію функцій планування, нагадування, зберігання та аналізу даних у мобільному додатку медичного призначення.

Завданням кваліфікаційної роботи є:

– провести аналіз предметної області та визначити основні проблеми у процесі цифрової підтримки приймання лікарських засобів;

– виконати огляд і порівняння існуючих технологій Android-розробки та вибрати оптимальні інструменти для реалізації застосунку;

– розробити архітектуру системи та структуру бази даних із урахуванням вимог до надійності, стабільності та масштабованості;

– створити мобільний додаток мовою Kotlin із використанням Jetpack Compose, Room, AlarmManager та OkHTTP;

– провести тестування працездатності програмного продукту та виконати оцінку його функціональності, надійності й зручності використання;

– здійснити експериментальний аналіз отриманих результатів та підтвердити ефективність запропонованого технічного рішення.

Наукова новизна даної кваліфікаційної роботи полягає у розробленні інтегрованої архітектури мобільного додатка, що поєднує технології Jetpack Compose, Room ORM, AlarmManager та OkHTTP у межах архітектурної моделі MVVM (Model-View-ViewModel). Запропоновано підхід до автоматизованого формування нагадувань з урахуванням часових інтервалів і повторюваності подій, який забезпечує стабільність роботи навіть у фоновому режимі.

Наукова новизна також полягає у комплексному поєднанні подієво-орієнтованого програмування та реактивного інтерфейсу, що підвищує ефективність і надійність системи.

Практичним значенням кваліфікаційної роботи є створення реально функціонуючого мобільного додатка, який може бути використаний: як персональний інструмент для користувачів, що проходять курс медикаментозного лікування.

Апробація результатів дослідження. Основні результати дослідження та програмні рішення були обговорені на засіданнях кафедри інженерії програмного забезпечення Луцького національного технічного університету, а також представлені під час студентських наукових конференцій і практичних семінарів факультету комп'ютерних та інформаційних технологій. Зокрема, опубліковані тези «Сучасні технології та архітектурні підходи до розробки мобільних застосунків на платформі Android» в збірнику наукових праць за матеріалами 3-ї Міжнародної науково-практичної конференції «Innovative Approaches in Modern Science and Technology» (додаток А).

РОЗДІЛ 1
АНАЛІЗ ПРОБЛЕМАТИКИ ТА ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ

1.1 Огляд і аналіз предметної області проблеми, результатів існуючих теоретичних та експериментальних досліджень

У сучасному суспільстві, що характеризується високим темпом життя та інформаційним перевантаженням, підтримання здоров'я людини стає все складнішим завданням. Одним із важливих аспектів є дотримання приписів лікаря щодо регулярного приймання лікарських засобів. Проте численні дослідження вказують, що значна частина пацієнтів не дотримується графіка лікування через забудькуватість, недостатню мотивацію чи труднощі в організації часу. За даними Всесвітньої організації охорони здоров'я, рівень дотримання пацієнтами медичних призначень становить у середньому лише 50 %, що призводить до зниження ефективності терапії та збільшення ризику ускладнень.

Розвиток мобільних технологій відкрив нові можливості для вирішення цієї проблеми. Мобільні застосунки медичного спрямування, що належать до категорії mHealth (mobile health), стали ефективним інструментом цифрової підтримки здоров'я. Вони забезпечують користувачам можливість контролювати власний графік лікування, отримувати нагадування, вести облік приймання препаратів, а також аналізувати індивідуальні показники.

Наукові праці Marques G., Ferreira C. [1], Zhou L. [2] підкреслюють, що впровадження мобільних інформаційних систем у медичну практику сприяє підвищенню дисциплінованості пацієнтів і покращенню результатів лікування.

У літературі сформувалися кілька напрямів розвитку таких систем:

– інформаційно-довідкові додатки, які містять бази даних препаратів та інструкції до них;

– системи контролю прийому ліків (Medication Reminder Systems), які реалізують функцію нагадування;

– комплексні платформи моніторингу здоров'я, що поєднують аналіз фізичної активності, сну, харчування і приймання ліків.

До найпоширеніших комерційних реалізацій належать:

– Medisafe, який забезпечує формування детального графіка приймання препаратів і можливість обміну даними з родичами чи лікарями;

– MyTherapy, що поєднує функціонал нагадувань з фіксацією симптомів, активності та аналізом показників здоров'я;

– Pill Reminder, який орієнтований на простоту використання і мінімалістичний інтерфейс.

Їхній аналіз показує, що, незважаючи на функціональність, ці продукти часто мають перевантажений інтерфейс, обмежені можливості персоналізації, а також низький рівень інтеграції з іншими сервісами (календарями, хмарними сховищами, обліковими записами лікаря). Тому одним із ключових завдань сучасних досліджень є створення інтуїтивно зрозумілих і персоналізованих додатків, які враховують потреби конкретного користувача та забезпечують зручну взаємодію.

Наукова публікація Jenifer Tidwell [3] акцентує увагу на важливості UX/UI-дизайну в медичних застосунках, адже саме простота інтерфейсу та його доступність визначають готовність користувача систематично взаємодіяти з програмою. Водночас окремі інші праці підкреслюють значення архітектурної чіткості та якісної програмної інженерії у розробленні надійних рішень для критичних сфер, таких як медицина [4-5].

Технічні дослідження останніх років демонструють ефективність використання мови програмування Kotlin, фреймворку Jetpack Compose для побудови інтерфейсу користувача, бібліотеки ORM Room для збереження даних та OkHTTP для організації взаємодії з мережею. Застосування архітектурного підходу MVVM (Model-View-ViewModel) дає змогу забезпечити модульність системи, полегшити тестування та підвищити стабільність програмного продукту. Такі технології і концепції активно розглядаються в сучасних

методичних джерелах і підтверджують свою ефективність у мобільних рішеннях для охорони здоров'я.

Окрему увагу дослідники приділяють питанням інформаційної безпеки та захисту персональних даних у медичних застосунках. Рекомендовано впроваджувати шифрування локальних баз даних, автентифікацію користувачів, а також мінімізацію збору конфіденційної інформації.

У ряді експериментальних робіт, опублікованих у фахових виданнях останніх років, розглядається перспектива інтеграції елементів штучного інтелекту (AI) для персоналізованих рекомендацій – наприклад, прогнозування часу приймання ліків на основі історії користувача чи поведінкових моделей.

Підсумовуючи результати аналізу теоретичних і практичних джерел, можна зробити висновок, що предметна область мобільних додатків для нагадування про приймання ліків активно розвивається, однак має низку нерозв'язаних завдань, серед яких:

– підвищення рівня персоналізації та адаптивності додатків;
– забезпечення безпеки зберігання медичних даних;
– спрощення взаємодії користувача з інтерфейсом;
– інтеграція інтелектуальних аналітичних компонентів.

Отже, є актуальною задача розробки інтелектуального мобільного застосунку для управління нагадуваннями про приймання лікарських засобів, що поєднає сучасні технології Android-розробки, принципи матеріального дизайну, безпечну роботу з даними та високу зручність користування.

1.2 Огляд і аналіз методів та засобів розробки для вирішення проблеми дослідження

Для вирішення проблеми забезпечення користувачів зручним інструментом контролю приймання лікарських засобів можуть бути застосовані різні методи та засоби комп'ютерних технологій. Головна ідея полягає у створенні інформаційної системи у вигляді мобільного програмного засобу, який

дозволяє користувачам планувати графік приймання ліків, отримувати своєчасні нагадування, зберігати історію лікування та аналізувати виконання розкладу.

Серед основних напрямів, у межах яких може бути реалізовано завдання створення системи нагадувань про приймання лікарських засобів, виділяють чотири ключові підходи: веб-орієнтований, десктопний, хмарний та мобільний. Кожен із них має свої переваги, недоліки та специфічні сфери застосування, що визначають доцільність використання у конкретному проєкті.

Веб-орієнтований підхід передбачає створення інформаційної системи у вигляді веб-додатку, який функціонує через браузер і використовує централізовану базу даних. Його головною перевагою є кросплатформність, що забезпечує доступ із будь-якого пристрою, незалежно від операційної системи. Крім того, такий підхід спрощує адміністрування, оскільки всі дані зберігаються на сервері, а оновлення системи відбуваються централізовано. Завдяки цьому лікарі або медичні заклади можуть отримувати доступ до інформації про пацієнтів у будь-який момент часу. Проте основним недоліком веб-рішень є необхідність постійного підключення до мережі Інтернет, що обмежує зручність використання у повсякденному житті, особливо для пацієнтів, які не мають стабільного доступу до мережі. Крім того, веб-додатки не можуть ефективно працювати у фоновому режимі, а отже, не здатні забезпечувати своєчасні нагадування без участі користувача. Саме тому цей підхід частіше застосовується для професійних медичних систем, а не для персональних рішень.

Десктопні програми реалізують подібні функції, але встановлюються безпосередньо на комп'ютер користувача. Вони забезпечують високий рівень продуктивності, дозволяють працювати з великими обсягами даних, реалізовують складні алгоритми аналізу та можуть взаємодіяти з медичними приладами, наприклад, для автоматичного зчитування показників здоров'я. Однак такі програми не є мобільними, адже користувач не може отримати сповіщення поза робочим місцем або під час відсутності біля комп'ютера. Це значно знижує ефективність у контексті особистих нагадувань, коли важливою є саме своєчасність реакції системи. Таким чином, десктопні рішення

залишаються доцільними переважно для фахівців медичної галузі, аналітиків або систем підтримки прийняття рішень у лікарнях.

Хмарно-орієнтований підхід базується на використанні віддалених серверів для зберігання даних і синхронізації між різними пристроями користувача. У такій системі дані зберігаються не локально, а у хмарному сховищі, що забезпечує доступ до них із будь-якої точки світу, резервне копіювання та високий рівень надійності. Додатковою перевагою є можливість аналітичної обробки даних, зокрема відстеження статистики прийомів, формування звітів або рекомендацій. Однак цей підхід потребує постійного підключення до Інтернету та автентифікації користувачів, що створює певні ризики з погляду безпеки персональної інформації. До того ж передача медичних даних через мережу потребує шифрування та дотримання стандартів конфіденційності, таких як GDPR або HIPAA. Хмарні технології особливо ефективні в інтегрованих рішеннях, де мобільний додаток виступає клієнтом, а сервер – платформою для зберігання історії прийомів, взаємодії з лікарем або телемедичного моніторингу.

Мобільний підхід сьогодні є найбільш ефективним та зручним способом вирішення задачі контролю приймання ліків. Смартфон – це постійний супутник користувача, оснащений сенсорами, годинником, системою сповіщень і можливістю фонової роботи. Завдяки цим властивостям мобільний додаток може оперативно надсилати нагадування навіть без підключення до мережі, забезпечуючи автономність і стабільність роботи. Основними перевагами цього підходу є мобільність, персоналізація сповіщень, інтеграція з календарем і можливість адаптації до звичок користувача. Проте існують і певні обмеження, пов'язані з апаратними ресурсами пристроїв, обмеженням енергоспоживання та необхідністю оптимізації процесів для забезпечення швидкої реакції системи. Незважаючи на це, саме мобільні застосунки найкраще відповідають потребам звичайного користувача, оскільки забезпечують швидкість, автономність і комфорт у користуванні.

Таким чином, аналіз підходів показує, що кожен із них може бути ефективним у своїй сфері застосування, однак для задачі нагадування про прийом лікарських засобів найдоцільнішим є саме мобільний варіант реалізації. Він поєднує переваги автономної роботи, високої доступності, зручності користування та можливості інтеграції з іншими цифровими сервісами. Саме на цій основі доцільно розробляти інтелектуальний мобільний додаток, який забезпечуватиме користувачеві простий, надійний і ефективний контроль виконання рекомендацій лікаря.

Для створення інформаційних систем, подібних до мобільного застосунку для нагадування про приймання лікарських засобів, можуть бути використані різні підходи до програмування та методи інженерії програмного забезпечення, кожен із яких має свої переваги та сферу застосування.

Одним із ключових напрямів є об'єктно-орієнтоване програмування (ООП), яке забезпечує структурування системи у вигляді взаємопов'язаних класів і об'єктів. Такий підхід дозволяє логічно моделювати реальні сутності, наприклад: лікарський засіб (Medicine), нагадування (Reminder), користувача (User). Завдяки інкапсуляції, наслідуванню та поліморфізму досягається висока модульність коду, можливість повторного використання компонентів і гнучкість під час подальшого розширення системи. Об'єктно-орієнтований підхід забезпечує зручне масштабування програми та полегшує її тестування й супровід.

Іншим важливим напрямом є подієво-орієнтоване програмування, яке ґрунтується на реакції системи на певні події. У випадку мобільного додатка подіями можуть бути натискання кнопки, настання визначеного часу для нагадування чи зміна стану бази даних. Такий підхід є надзвичайно ефективним для створення інтерактивних застосунків, де важливою є миттєва реакція на дії користувача або системні сигнали. Він дозволяє будувати асинхронну логіку обробки подій і забезпечує природну інтеграцію з системними сервісами Android, наприклад AlarmManager чи NotificationManager.

Важливу роль у створенні масштабованих і зручних у супроводі програм відіграє модульне програмування, що передбачає поділ програмного продукту на незалежні компоненти. У контексті розроблюваної системи це можуть бути окремі модулі: керування нагадуваннями, робота з базою даних, аналітика статистики або обробка сповіщень. Такий підхід значно спрощує налагодження, дозволяє працювати над різними частинами системи незалежно й полегшує інтеграцію нових функцій без впливу на основну логіку.

Під час розроблення сучасних програмних рішень важливе місце займають архітектурні патерни, які визначають логічну структуру застосунку та взаємодію його частин. Серед найпоширеніших моделей – MVC (Model-View-Controller), MVP (Model-View-Presenter) та MVVM (Model-View-ViewModel). Для мобільних застосунків під платформу Android найбільш ефективною є архітектура MVVM, оскільки вона дозволяє відокремити логіку роботи з даними від інтерфейсу користувача. Це підвищує стабільність програми, полегшує її тестування та робить код більш зрозумілим. Завдяки підтримці бібліотек Android Jetpack і механізмів ViewModel, MVVM забезпечує реактивну взаємодію між даними й інтерфейсом, що особливо важливо у застосунках, де інформація часто оновлюється.

На етапі проєктування системи доцільно застосовувати методи візуального моделювання, зокрема мову UML (Unified Modeling Language). Вона використовується для створення різних типів діаграм – варіантів використання (use case), класів (class), діяльності (activity), послідовності (sequence) – які допомагають формалізувати логіку роботи системи, відобразити взаємозв'язки між компонентами та описати сценарії взаємодії користувача із застосунком. Використання UML-діаграм сприяє узгодженню роботи між аналітиками, розробниками й тестувальниками, а також підвищує зрозумілість структури проєкту на етапі супроводу.

Ще одним важливим аспектом є методи управління процесом розробки програмного забезпечення. Залежно від цілей і ресурсів проєкту можуть застосовуватись різні моделі життєвого циклу – каскадна (Waterfall), спіральна

(Spiral) або гнучкі методології (Agile, Scrum). Для невеликих команд і коротких циклів оновлень найчастіше використовується Agile-підхід, який дозволяє поступово удосконалювати систему, отримуючи зворотний зв'язок від користувачів і вносячи корективи без необхідності повної перебудови структури. Водночас для науково-дослідних розробок чи систем із високими вимогами до формальної документації може застосовуватись каскадна модель, що забезпечує чітке проходження усіх етапів – від аналізу до впровадження.

Таким чином, різноманіття програмних парадигм і методів інженерії програмного забезпечення дає змогу обрати оптимальну комбінацію під конкретне завдання. У випадку створення мобільного застосунку для контролю прийому ліків найбільш ефективним є поєднання об'єктно-орієнтованого, подієво-орієнтованого та модульного підходів у межах архітектури MVVM із використанням UML-моделювання й гнучкої методології управління розробкою. Така стратегія забезпечує структурованість, надійність, масштабованість і високу якість програмного продукту [6].

Сучасні мобільні рішення для Android розробляються у середовищі Android Studio із використанням мови програмування Kotlin. Ця мова поєднує лаконічність, високу продуктивність і безпечність типів даних, що робить її оптимальним інструментом для створення надійних застосунків [7].

Для побудови інтерфейсу користувача застосовується Jetpack Compose – сучасний фреймворк декларативного UI-програмування, який дає змогу створювати динамічні та адаптивні екрани без використання XML [8].

Для зберігання даних у локальному сховищі використовуються технології SQLite або її надбудова ORM Room, яка надає об'єктно-реляційне відображення даних та мінімізує помилки під час роботи з базою.

Для керування сповіщеннями застосовується системний сервіс AlarmManager, який дозволяє програмно планувати виконання подій у певний час і забезпечує роботу нагадувань навіть у фоновому режимі.

Для обміну даними з віддаленими сервісами або синхронізації користувацької інформації доцільно використовувати OkHTTP або інші

бібліотеки для HTTP-запитів. Це дозволяє реалізувати можливість збереження профілю користувача у хмарі чи передачу статистики.

Крім того, при розробленні подібних систем застосовуються методи візуального моделювання – побудова UML-діаграм (use case, activity, class), які описують логіку роботи, взаємозв'язки компонентів і сценарії користування. Це дозволяє оцінити адекватність майбутньої системи до вимог користувачів і перевірити її цілісність на етапі проєктування.

Таким чином, аналіз існуючих методів і засобів показує, що найефективнішим шляхом вирішення поставленої задачі є розроблення мобільного додатка під платформу Android з використанням мови Kotlin, архітектури MVVM, інструментарію Jetpack Compose, бази даних Room і системи сповіщень AlarmManager.

Обрана сукупність засобів дозволяє створити програмний продукт, який працює автономно, без постійного підключення до мережі; має інтуїтивний інтерфейс; забезпечує надійне збереження користувацьких даних; гарантує точність і стабільність нагадувань.

Використання цих технологій дає змогу ефективно реалізувати поставлену проблему дослідження – створення інтелектуального мобільного засобу для управління графіком приймання ліків і покращення дисципліни користувачів щодо лікування.

1.3 Постановка завдання на кваліфікаційну роботу магістра

На основі проведеного аналізу предметної області, огляду наукових праць та сучасних методів розробки програмного забезпечення було визначено, що одним із актуальних напрямів є створення інтелектуального мобільного засобу для нагадування про приймання лікарських препаратів. Такий програмний продукт покликаний сприяти підвищенню дисциплінованості користувачів під час лікування, забезпечити контроль за виконанням медичних призначень і зменшити кількість пропусків прийому ліків.

Для досягнення поставленої мети у роботі визначено такі завдання дослідження:

– здійснити аналіз предметної області мобільних застосунків для нагадування про приймання лікарських засобів та узагальнити сучасні підходи до їх реалізації;

– виконати огляд існуючих технологій Android-розробки й визначити інструменти, що найкраще відповідають вимогам до створення системи нагадувань;

– обґрунтувати вибір архітектури програмного засобу, структури бази даних та засобів побудови інтерфейсу користувача;

– розробити компоненти системи та реалізувати мобільний додаток у середовищі Android Studio з використанням Kotlin, Jetpack Compose, Room, AlarmManager та OkHTTP;

– провести функціональне тестування та оцінити якість роботи програмного продукту за критеріями точності, зручності й надійності;

– здійснити експериментальний аналіз отриманих результатів та підтвердити ефективність розробленої системи нагадувань.

Відповідно до поставленої мети й завдань, у межах кваліфікаційної роботи розробляється мобільний застосунок для нагадувань про прийом лікарських засобів, який реалізує функції реєстрації медикаментів, формування графіка прийомів, надсилання сповіщень, збереження історії та відображення статистики виконання.

РОЗДІЛ 2
ТЕОРЕТИЧНЕ ДОСЛІДЖЕННЯ ТА ПРАКТИЧНА РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Обґрунтування вибору шляхів, технологій, алгоритмів і засобів вирішення поставленого завдання

Розроблення програмного засобу для управління нагадуваннями про приймання лікарських засобів потребує комплексного підходу, який поєднує аналіз предметної області, вибір відповідних методів, технологій і архітектурних рішень.

Обґрунтування цих рішень базується на принципах інженерії програмного забезпечення, теорії моделювання інформаційних систем та методах забезпечення якості програмного продукту.

Основною метою дослідження є створення інтелектуального мобільного додатка, який забезпечує:

- реєстрацію та облік медикаментів;
- налаштування часу і періодичності приймання;
- генерацію нагадувань у заданий час;
- збереження історії виконання завдань.

Виходячи з цього, розробка передбачає реалізацію подієво-орієнтованої системи, у якій робота побудована на обробці подій (сповіщення, введення даних користувача, оновлення стану). Такий підхід дозволяє мінімізувати ресурси пристрою, забезпечити автономність і високу реактивність системи.

Для досягнення цілей застосовано інкрементно-ітераційну модель розроблення, у якій кожна функціональна частина (модуль додавання ліків, модуль нагадувань, статистика) реалізується й тестується окремо. Це забезпечує гнучкість, легкість внесення змін і високу якість кінцевого продукту.

Згідно з принципами системного підходу, програмний засіб розглядається як інформаційна система, що складається з таких основних підсистем:

– підсистема введення даних – створення записів про ліки, дозування, час приймання, тривалість курсу;

– підсистема керування нагадуваннями – планування подій за часом і частотою, генерація повідомлень;

– підсистема збереження даних – формування та підтримка бази даних користувача;

– підсистема візуалізації – інтерфейс користувача для взаємодії з додатком;

– аналітична підсистема (у перспективі) – аналіз виконання графіка прийому, прогнозування та рекомендації.

Для опису структури й взаємозв'язків використано UML-діаграми.

Побудуємо діаграму Use Case Diagram – для визначення сценаріїв взаємодії користувача з системою (рис. 2.1).

Рисунок 2.1 – Діаграма сценаріїв використання

Діаграма Activity Diagram служить для моделювання процесу приймання ліків (рис. 2.2).

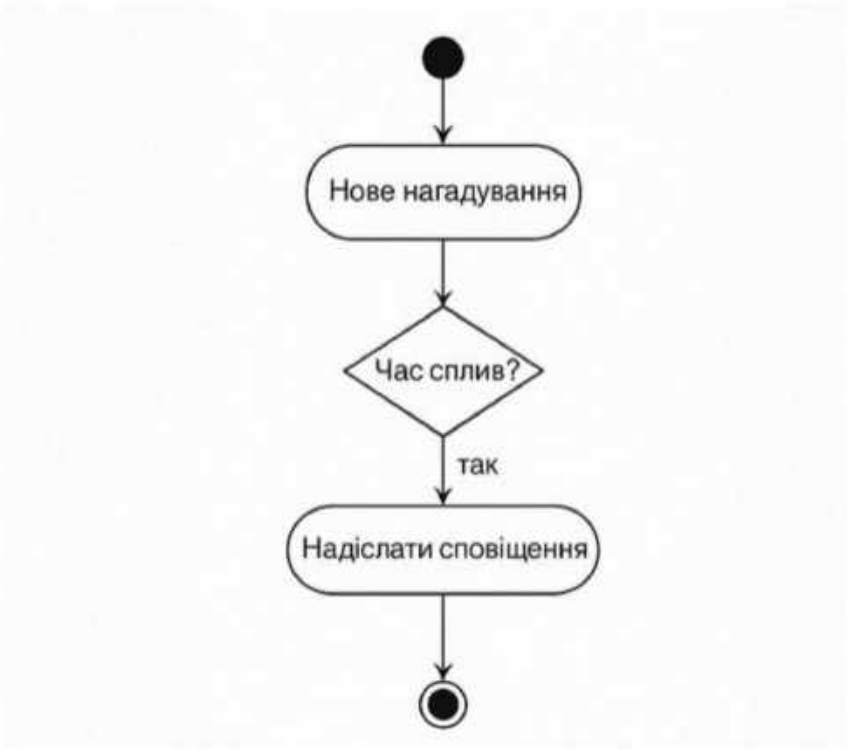

Рисунок 2.2 – Діаграма діяльності

Class Diagram призначена для представлення основних сутностей і зв'язків між ними (ліки, нагадування, користувач, розклад) (рис. 2.3).

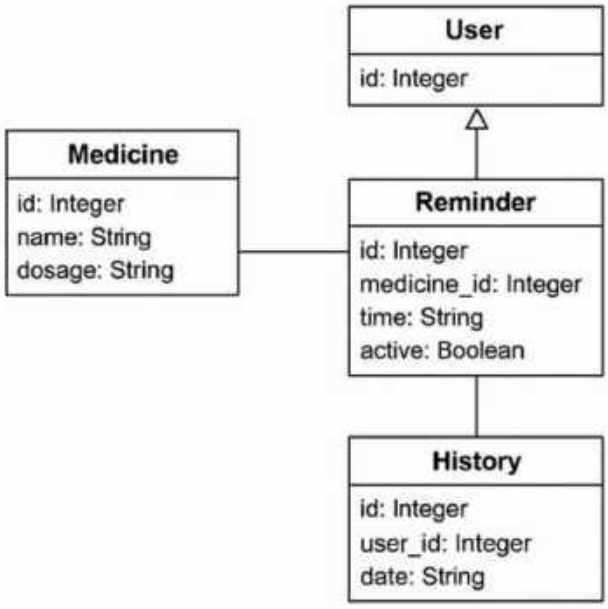

Рисунок 2.3 – Діаграма класів

Побудуємо діаграму Sequence Diagram – для демонстрації послідовності дій під час виконання нагадування (рис. 2.4).

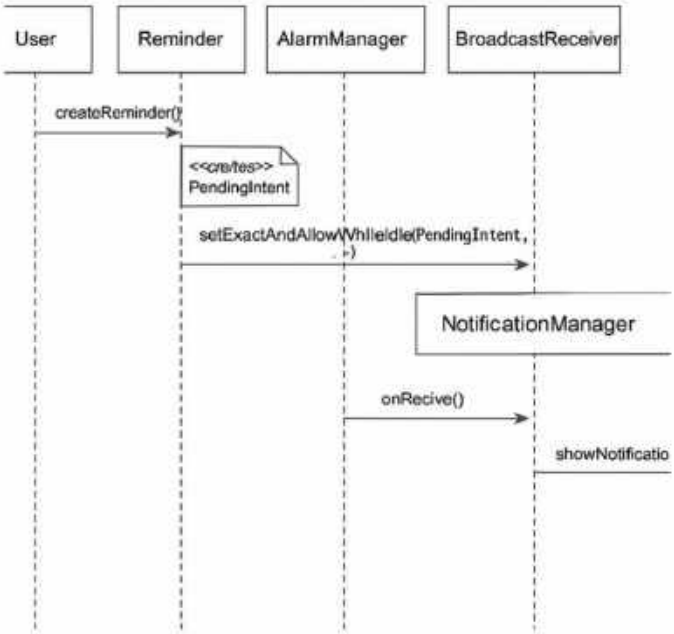

Рисунок 2.4 – Діаграма послідовності

Таке моделювання дає змогу побудувати адекватну логічну модель системи, що відповідає вимогам предметної області.

Для проєктування мобільного додатка обґрунтовано обрано архітектуру MVVM (Model-View-ViewModel), яка є рекомендованою для сучасної Android-розробки (рис. 2.5).

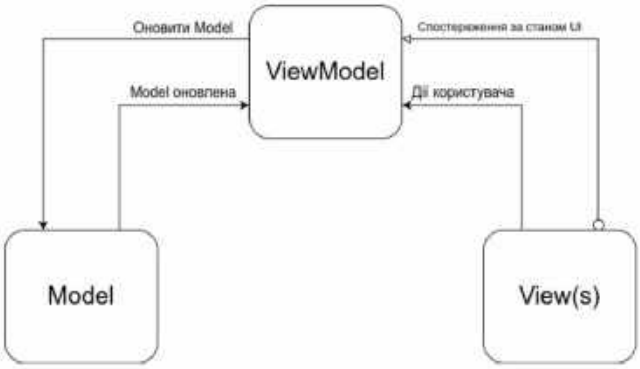

Рисунок 2.5 – Діаграма моделі MVVM

Вона забезпечує чітке розділення бізнес-логіки, інтерфейсу та даних; легке тестування та масштабування; зменшення кількості взаємозалежностей між компонентами; стабільність роботи при зміні стану інтерфейсу.

В архітектурі MVVM:

– Model містить класи даних і методи взаємодії з базою Room (додавання, видалення, оновлення записів);

– ViewModel забезпечує логіку керування даними та їхнє спостереження через LiveData;

– View (UI) реалізована з використанням Jetpack Compose, що динамічно відображає дані в інтерфейсі [9-10].

Вибір мови Kotlin обґрунтований її офіційною підтримкою з боку Google, безпечністю, сумісністю з Java, компактністю синтаксису та підтримкою корутин, які дозволяють ефективно виконувати асинхронні операції (зокрема таймери й сповіщення).

Мова програмування Kotlin є сучасним інструментом розробки програмного забезпечення, орієнтованим на підвищення продуктивності, надійності та зручності створення коду. Вона офіційно підтримується компанією Google як основна мова для розроблення Android-додатків і завдяки тісній інтеграції з екосистемою Java отримала широке поширення серед розробників [11].

Однією з ключових переваг Kotlin є лаконічність та виразність синтаксису. Код, написаний на цій мові, є коротшим і зрозумілішим у порівнянні з аналогічними фрагментами на Java. Завдяки усуненню надлишкового шаблонного коду (boilerplate) і застосуванню сучасних конструкцій, процес розробки стає швидшим, а ймовірність помилок – меншою [11].

Важливою характеристикою Kotlin є повна сумісність із Java, що дозволяє безперешкодно використовувати існуючі бібліотеки, фреймворки та модулі. Це забезпечує гнучкість при міграції проєктів, створених на Java, і сприяє поступовому переходу на більш сучасні технологічні рішення.

Особливу увагу розробники приділили забезпеченню безпеки коду. Мова має вбудовані механізми запобігання поширеним помилкам, зокрема NullPointerException. Система типів Kotlin дозволяє надійно контролювати роботу з null-значеннями, що підвищує стабільність програм і зменшує ризик збоїв під час виконання.

До функціональних можливостей мови належать підтримка лямбда-виразів, корутин для асинхронного програмування, розширення класів без їхньої модифікації, а також інші сучасні засоби, що розширюють потенціал розробників. Завдяки цьому Kotlin поєднує в собі переваги об'єктно-орієнтованого та функціонального підходів до програмування.

Додатковою перевагою є активна підтримка спільноти та компанії JetBrains, яка є розробником мови. Регулярні оновлення, покращення інструментів, доступна документація та навчальні ресурси сприяють її швидкому розвитку й зручності використання у практичних проєктах [12].

Отже, Kotlin можна вважати оптимальним вибором для створення сучасних Android-додатків, що поєднують ефективність, безпеку та гнучкість. Його можливості дають змогу розробляти надійні, продуктивні та зручні програмні рішення, які відповідають вимогам сучасної програмної інженерії.

Android Studio забезпечує інтеграцію з Gradle, автоматичне збирання проєкту, налагодження, вбудовані емулятори, інструменти профілювання та перевірки продуктивності.

Jetpack Compose використовується для побудови інтерфейсу користувача декларативним методом. Завдяки реактивності даних (State Management), зміни у ViewModel миттєво відображаються в інтерфейсі. Це підвищує стабільність і швидкість реакції додатка.

Jetpack Compose – це сучасний фреймворк для розроблення інтерфейсів користувача на платформі Android, орієнтований на спрощення процесу створення, налагодження та підтримки графічного середовища додатків. Інструмент побудований на декларативному підході, який дозволяє описувати

структуру інтерфейсу безпосередньо в коді програми за допомогою функцій, замість використання традиційних XML-файлів розмітки [13-15].

Однією з ключових переваг Jetpack Compose є скорочення складності та часу розроблення. Завдяки декларативному синтаксису, інтерфейс формується динамічно на основі поточного стану даних, що спрощує оновлення елементів і реакцію на зміни в програмі. Це дає змогу швидко створювати прототипи, тестувати окремі компоненти й оперативно вносити зміни під час розробки.

Фреймворк має повну інтеграцію з екосистемою Jetpack, зокрема з бібліотеками Material Design, Navigation, ViewModel та іншими компонентами Android Architecture. Підтримка Material Design 3 забезпечує можливість побудови сучасних, адаптивних і естетично привабливих інтерфейсів, які відповідають останнім вимогам до дизайну мобільних застосунків.

Висока гнучкість Jetpack Compose дає змогу розробникам створювати власні елементи інтерфейсу, налаштовувати їхню поведінку відповідно до контексту, динамічно змінювати зовнішній вигляд і реакцію на дії користувача. Завдяки компонентному підходу, кожен елемент інтерфейсу може бути повторно використаний у різних частинах програми.

Jetpack Compose також сприяє підвищенню продуктивності. Механізм «recomposition» оновлює лише ті елементи інтерфейсу, які зазнали змін, що зменшує навантаження на систему та забезпечує плавність роботи навіть на пристроях із обмеженими ресурсами [13-15].

Важливою перевагою є і зручність тестування. Компоненти Compose реалізовані у вигляді функцій, тому їх легко ізолювати й перевіряти незалежно від решти програми. Це спрощує процес налагодження, підвищує надійність системи та сприяє дотриманню принципів модульного тестування.

Отже, Jetpack Compose можна розглядати як ефективний і прогресивний інструмент для створення інтерфейсів користувача, який поєднує швидкість розроблення, високу продуктивність, зручність супроводу та відповідність сучасним стандартам дизайну. Його використання забезпечує створення

інтуїтивно зрозумілих, естетичних і стабільних Android-додатків, орієнтованих на потреби користувача.

Для локального зберігання даних використовується бібліотека Room, що забезпечує автоматичну генерацію SQL-запитів; контроль цілісності даних через анотації; асинхронну обробку запитів; мінімізацію ручного коду SQL.

Room (Object-Relational Mapping, ORM) – це сучасна бібліотека для роботи з базами даних у мобільних застосунках, що реалізуються на платформі Android. Вона є складовою екосистеми Android Jetpack і призначена для спрощення доступу до локальної бази даних SQLite. Основна ідея Room полягає у забезпеченні зручної взаємодії з даними через об'єктно-реляційне відображення, що дозволяє розробникам працювати з базою не на рівні SQL-запитів, а через об'єкти мови програмування Kotlin [16].

Однією з ключових переваг Room є автоматизація процесів взаємодії з базою даних. Бібліотека самостійно генерує код запитів, перевіряє їх коректність під час компіляції та зменшує ризик логічних помилок. Такий підхід значно підвищує надійність, безпечність і стабільність програмного забезпечення, особливо при роботі з великими обсягами даних.

Room має тісну інтеграцію з Kotlin та іншими компонентами Jetpack, що дозволяє ефективно поєднувати роботу з базою даних і реактивне оновлення інтерфейсу користувача. Визначення структур даних здійснюється за допомогою анотацій, що робить код більш читабельним і мінімізує кількість шаблонних конструкцій.

Архітектура Room передбачає використання трьох основних компонентів:

– Entity – клас, який описує структуру таблиці бази даних. Кожен об'єкт цього класу відповідає окремому рядку таблиці;

– Dao (Data Access Object) – інтерфейс, що містить методи для виконання запитів. Реалізація цих методів автоматично генерується бібліотекою;

– Database – клас, який визначає базу даних і містить посилання на всі сутності та DAO, що використовуються у застосунку.

Завдяки такій структурі Room дозволяє уникнути надлишкової ручної роботи із SQL-запитами, підтримує асинхронні операції, транзакції, індекси та зовнішні ключі, що забезпечує високу гнучкість і ефективність управління даними.

Бібліотека також підтримує різні типи зв'язків між сутностями – один-до-одного, один-до-багатьох та багато-до-багатьох. Це дає змогу моделювати складні інформаційні структури, зберігати колекції об'єктів і забезпечувати узгодженість даних у межах додатка.

Висока продуктивність Room досягається завдяки використанню оптимізованих механізмів роботи з SQLite, індексації даних і компіляційних перевірок запитів. Це робить бібліотеку придатною навіть для мобільних систем із обмеженими ресурсами.

Таким чином, Room ORM є надійним та ефективним засобом збереження й обробки даних у застосунках Android. Завдяки поєднанню простоти використання, високої продуктивності, безпеки та інтеграції з Kotlin і Jetpack Compose, ця бібліотека забезпечує створення стабільних і продуктивних програмних рішень, які відповідають сучасним вимогам до мобільної розробки.

Для реалізації нагадувань про прийом ліків обрано системний сервіс AlarmManager. Він дозволяє планувати виконання подій у заданий час навіть у неактивному стані додатка, забезпечуючи надійність і точність сповіщень.

OkHTTP – це високопродуктивна бібліотека для здійснення мережевих запитів за протоколом HTTP/HTTPS, яка широко застосовується в Android-розробці для обміну даними між клієнтською та серверною частинами застосунку. Вона забезпечує розробникам зручний, гнучкий і надійний інтерфейс для взаємодії з мережею, абстрагуючи складні технічні деталі передачі даних [17].

Однією з ключових характеристик бібліотеки є підтримка основних типів HTTP-запитів (GET, POST, PUT, DELETE та ін.), а також можливість їх глибокої кастомізації. OkHTTP дозволяє легко визначати заголовки запитів, параметри, тіла запитів і обробляти відповіді у різних форматах – JSON, XML, текстових або

мультимедійних. Завдяки роботі з потоками даних, бібліотека ефективно оперує великими файлами, мінімізуючи використання оперативної пам'яті.

Важливою перевагою OkHTTP є асинхронна обробка мережевих операцій. Такий підхід дозволяє виконувати запити у фоновому режимі, не блокуючи головний потік інтерфейсу користувача. Це особливо актуально для мобільних додатків, де стабільність і швидкість реакції мають критичне значення. Крім того, бібліотека має вбудовані механізми обробки помилок та повторних спроб запиту (retry-policy), що підвищує надійність і стійкість програмного продукту до перебоїв у мережі.

Серед додаткових можливостей OkHTTP варто відзначити систему кешування результатів запитів, яка дає змогу зменшити навантаження на сервер і прискорити отримання відповідей при повторних зверненнях. Це особливо корисно в умовах обмеженого доступу до мережі або низької пропускної здатності. Бібліотека також інтегрується з іншими популярними інструментами Android-екосистеми, зокрема з Retrofit, що спрощує створення RESTful API-клієнтів і підвищує рівень абстракції при роботі з серверними даними.

Важливою складовою є й забезпечення безпеки передавання інформації. OkHTTP підтримує сучасні стандарти захищених з'єднань, зокрема SSL/TLS, а також дає змогу розробникам налаштовувати власні параметри перевірки сертифікатів та керування довіреними джерелами. Це гарантує конфіденційність і цілісність даних під час обміну між клієнтом і сервером.

Бібліотека відзначається високою продуктивністю та стабільністю, оскільки оптимізує роботу з потоками, зменшує затримки при з'єднаннях та повторно використовує HTTP-з'єднання завдяки внутрішньому пулу. Її архітектура побудована з урахуванням масштабованості, що дозволяє ефективно працювати як із невеликими застосунками, так і з великими мережевими системами.

Отже, OkHTTP є потужним і надійним інструментом для реалізації мережевої взаємодії в Android-застосунках. Її простота використання, асинхронність, підтримка кешування, захищених з'єднань і широка інтеграція з

іншими бібліотеками роблять OkHTTP оптимальним вибором для створення стабільних, масштабованих і продуктивних клієнт-серверних програм.

Основні характеристики, призначення, переваги та роль цих технологій у проєкті наведено в таблиці 2.1.

Таблиця 2.1 – Основні характеристики використаних технологій

Технологія	Призначення	Переваги	Роль у проєкті
Jetpack Compose	Створення інтерфейсу користувача	Декларативний підхід, швидка розробка, інтеграція з Material Design	UI Layer
Room	Робота з локальною базою даних	Автоматичне створення SQL-запитів, типобезпечність, простота інтеграції	Data Layer
AlarmManager	Планування нагадувань	Робота у фоновому режимі, точне виконання задач	Бізнес-логіка нагадувань
OkHTTP	Мережеві запити до веб-сервісів	Підтримка кешування, SSL/TLS, асинхронність	Мережева взаємодія

Якщо система потребує інтеграції з віддаленим сервером або синхронізації даних, застосовується OkHTTP – бібліотека для HTTP-запитів із підтримкою кешування, обробки помилок і безпечних з'єднань (HTTPS, TLS).

Ключовим елементом роботи додатка є алгоритм формування нагадувань. Він передбачає: зчитування з бази даних часу та періодичності приймання ліків; обчислення інтервалів між подіями відповідно до встановленого графіка; передавання подій до AlarmManager, який активує системні сповіщення; при настанні часу сповіщення – відображення повідомлення у вигляді Notification; зміна статусу прийому (виконано / пропущено) у базі даних.

Розрахунок часу базується на алгоритмі додавання часових інтервалів (time offsets) від базової точки старту курсу лікування. Для забезпечення коректності роботи враховується системна зона часу пристрою (TimeZone) і параметри повторюваності (щодня, через день, щотижня тощо).

Для перевірки адекватності алгоритму проводиться тестування на різних часових інтервалах, включаючи перезапуск системи, зміну дати та переведення часу, що гарантує надійну роботу програми в реальних умовах.

Основна гіпотеза полягає в тому, що застосування поєднання архітектури MVVM, технологій Jetpack Compose, Room і AlarmManager забезпечує створення програмного засобу, який:

– стабільно працює на різних версіях Android;

– забезпечує точність нагадувань і збереження даних без збоїв;

– є зручним і зрозумілим для користувача;

– не потребує постійного підключення до Інтернету.

Очікуваний результат – розроблення функціонального мобільного додатка, який підвищує дисципліну користувачів щодо приймання лікарських засобів і демонструє ефективність запропонованого технічного рішення.

Обґрунтований вибір технологій та архітектурних підходів дозволяє побудувати ефективну, стабільну та масштабовану систему. Застосування Kotlin, Jetpack Compose, Room і AlarmManager забезпечує узгоджену роботу компонентів, надійне збереження даних, зручний інтерфейс і точність нагадувань.

Таким чином, сукупність обраних методів, алгоритмів і засобів є адекватною до поставленої проблеми та забезпечує досягнення цілей дослідження.

2.2 Практична реалізація об’єкта проектування

Практична реалізація розробленого програмного засобу базується на результатах теоретичного дослідження та передбачає створення функціонального мобільного додатка для управління нагадуваннями про приймання лікарських засобів.

Основною метою створення додатка є підвищення організованості користувачів у дотриманні графіка лікування, мінімізація пропусків прийомів ліків і покращення контролю за виконанням рекомендацій лікаря.

Розроблений програмний продукт має на меті забезпечити:

– автоматичне формування графіка прийому лікарських засобів;

– нагадування користувачеві у визначений час;

– ведення історії прийомів і фіксацію виконаних подій;

– можливість перегляду статистики приймання;

– простоту та інтуїтивність інтерфейсу користувача.

Для досягнення поставленої мети реалізовано такі основні завдання:

– створення моделі даних для зберігання інформації про медикаменти, дозування, час і тривалість прийому;

– розроблення інтерфейсу введення та редагування даних користувачем;

– впровадження системи нагадувань через сповіщення Android;

– збереження історії дій користувача у локальній базі даних;

– реалізація механізму тестування і перевірки коректності функціонування додатка.

Система має модульну структуру та складається з кількох основних компонентів (рис. 2.6).

Модуль даних (Data Layer) відповідає за роботу з базою даних Room, створення сутностей (Entity), DAO-інтерфейсів і запитів.

Модуль логіки (Domain Layer) реалізує бізнес-логіку, обробку подій, керування нагадуваннями та взаємодію між даними та інтерфейсом.

Модуль представлення (UI Layer) побудований за допомогою Jetpack Compose і забезпечує візуальну взаємодію користувача із системою.

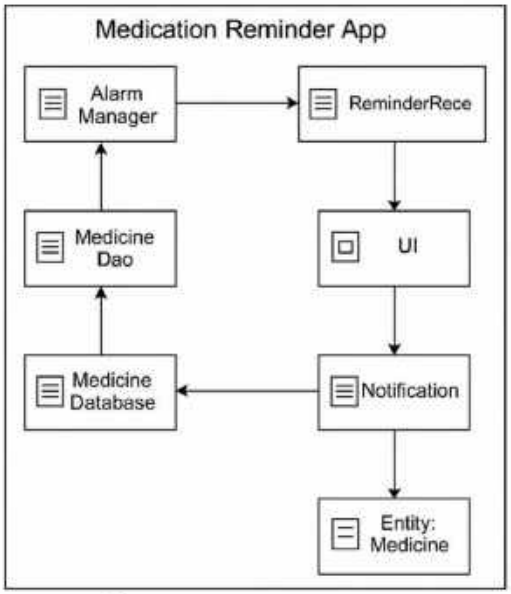

Рисунок 2.6 – Структура мобільного додатка для нагадувань про приймання ліків

Функціональність додатка охоплює такі основні можливості:

– додавання записів про лікарські засоби (назва, дозування, частота, період лікування);

– редагування та видалення існуючих записів;

– автоматичне створення нагадувань, які активуються у визначений користувачем час;

– сповіщення користувача через системні повідомлення (Notification Manager);

– відображення списку активних нагадувань і виконаних прийомів;

– збереження історії прийомів у локальній базі даних із можливістю її перегляду;

– облік статистики: кількість виконаних нагадувань, відсоток дотримання графіка;

– реалізація нагадувань здійснена через системний сервіс AlarmManager, який планує сповіщення навіть при неактивному стані програми (табл. 2.2).

Таблиця 2.2 – Ключові функції системи

Функція	Опис	Модуль
Додавання ліків	Користувач вводить дані про препарат (назва, дозування, час прийому)	UI, Data
Редагування записів	Можливість змінювати параметри або видаляти записи	UI, Domain
Створення нагадування	Формування події в AlarmManager для заданого часу	Domain
Перегляд історії	Відображення прийнятих доз і статистики	UI, Data
Сповіщення користувача	Показ повідомлення у системній панелі	UI, Domain

У лістингу 2.1 наведено фрагмент коду, що реалізує метод створення нагадування у визначений час за допомогою системного сервісу AlarmManager, який забезпечує виконання події навіть у фоновому режимі.

Лістинг 2.1 – Метод створення нагадування у визначений час за допомогою AlarmManager

```
fun setReminder(context: Context, medicine: Medicine) {
    val alarmManager = context.getSystemService(Context.ALARM_SERVICE) as
AlarmManager
    val intent = Intent(context, ReminderReceiver::class.java).apply {
        putExtra("medicine_name", medicine.name)
    }

    val pendingIntent = PendingIntent.getBroadcast(
        context, medicine.id, intent, PendingIntent.FLAG_UPDATE_CURRENT
or PendingIntent.FLAG_IMMUTABLE
    )
    val timeInMillis = LocalTime.parse(medicine.time)
        .atDate(LocalDate.now())
        .atZone(ZoneId.systemDefault())
        .toInstant().toEpochMilli()
    alarmManager.setExactAndAllowWhileIdle(
        AlarmManager.RTC_WAKEUP,
        timeInMillis,
```

```
        pendingIntent
    )
}
```

кінець лістингу 2.1

Повідомлення користувачеві реалізовані через NotificationManager, що дозволяє отримувати нагадування у вигляді push-сповіщень. У лістингу 2.2 подано реалізацію класу ReminderReceiver, який обробляє подію сповіщення та формує push-повідомлення для користувача через NotificationManager.

Лістинг 2.2 – Обробник події нагадування ReminderReceiver

```
class ReminderReceiver : BroadcastReceiver() {
    override fun onReceive(context: Context, intent: Intent) {
        val medicineName = intent.getStringExtra("medicine_name")
        val notification = NotificationCompat.Builder(context,
"reminder_channel")
            .setContentTitle("Нагадування")
            .setContentText("Час прийняти ліки: $medicineName")
            .setSmallIcon(R.drawable.ic_pill)
            .setPriority(NotificationCompat.PRIORITY_HIGH)
            .build()
        val manager = NotificationManagerCompat.from(context)
        manager.notify(medicineName.hashCode(), notification)
    }
}
```

кінець лістингу 2.2

База даних Room використовується для локального збереження інформації про ліки та події. Для визначення сутностей застосовано анотації @Entity, @Dao та @Database.

У лістингу 2.3 наведено визначення сутності Medicine для локальної бази даних Room, що описує структуру таблиці та основні атрибути лікарського засобу.

Лістинг 2.3 – Клас сутності Medicine у базі даних Room

```
@Entity(tableName = "medicines")
data class Medicine(
    @PrimaryKey(autoGenerate = true) val id: Int = 0,
    val name: String,
    val dosage: String,
    val time: String,
    val duration: Int
)
```

кінець лістингу 2.3

Для забезпечення можливості майбутнього розширення системи було реалізовано модуль синхронізації історії прийомів із віддаленим веб-сервісом (ліст. 2.4). Синхронізація виконується через HTTP-запит, сформований за допомогою бібліотеки OkHTTP.

Лістинг 2.4 – Відправлення історії прийому на сервер

```
object SyncService {

    private val client = OkHttpClient()
    fun syncHistory(historyJson: String) {
        val requestBody =
historyJson.toRequestBody("application/json".toMediaType())
        val request = Request.Builder()
            .url("https://example.com/api/sync")
            .post(requestBody)
            .build()
        client.newCall(request).enqueue(object : Callback {
            override fun onFailure(call: Call, e: IOException) {
                Log.e("SyncService", "Sync failed: ${e.message}")
            }
            override fun onResponse(call: Call, response: Response) {
                Log.d("SyncService", "Sync success: ${response.code}")
            }
}
```

кінець лістингу 2.4

Інтерфейс реалізований із використанням Jetpack Compose – сучасного декларативного підходу до UI-програмування.

Основні екрани системи:

– головний екран – відображає список активних нагадувань;

– екран додавання ліків – форма введення назви, дозування та часу приймання;

– екран історії прийомів – містить статистику виконання;

– екран налаштувань – дозволяє коригувати параметри сповіщень і тему оформлення.

Використання Material Design 3 забезпечує візуальну узгодженість, доступність та адаптивність під різні розміри екранів.

Після реалізації було проведено функціональне тестування з метою перевірки відповідності програми вимогам. Тестування охоплювало такі аспекти:

– перевірку додавання, редагування й видалення даних;

– правильність відображення інтерфейсу;

– точність спрацювання нагадувань;

– збереження даних у базі Room після перезапуску програми;

– роботу сповіщень у фоновому режимі.

Для автоматизованої перевірки використано JUnit (модульне тестування логіки) та Espresso (UI-тестування).

Результати тестування засвідчили стабільну роботу всіх модулів програми та відповідність її функціональності вимогам проекту.

Оцінювання якості здійснювалося за основними критеріями:

– функціональність – повна реалізація вимог користувача;

– зручність використання (usability) – простий і зрозумілий інтерфейс;

– надійність – відсутність збоїв під час роботи програми;

– продуктивність – швидке виконання запитів і мінімальне навантаження на систему;

– сумісність – коректна робота на пристроях із різними версіями Android.

Під час експериментального тестування помилки, що виникали, було усунуто. Програма пройшла етап внутрішнього налагодження та демонструє стабільну роботу.

У результаті практичної реалізації створено повнофункціональний мобільний додаток для управління нагадуваннями про приймання лікарських засобів.

Застосування сучасних технологій Kotlin, Jetpack Compose, Room, AlarmManager та OkHTTP забезпечило високу якість програмного продукту, його зручність, стабільність і надійність.

Програма може бути використана як персональний інструмент для користувачів, що прагнуть ефективно організувати прийом лікарських засобів, а також як основа для подальшого розширення – інтеграції з хмарними сервісами, аналітикою чи системами медичного моніторингу.

Почнемо з того що користувачу необхідно завантажити та встановити додаток. Відразу після запуску додатка відкривається початковий екеран з якої можна почати використовувати «Додаток» та всі його функції. Це зображено на рисунку 2.7.

Рисунок 2.7 – Початковий екран

Натиснувши на кнопку додати – користувач буде запропоноване заповнити форму та додати новий лікарський засіб (рис. 2.8).

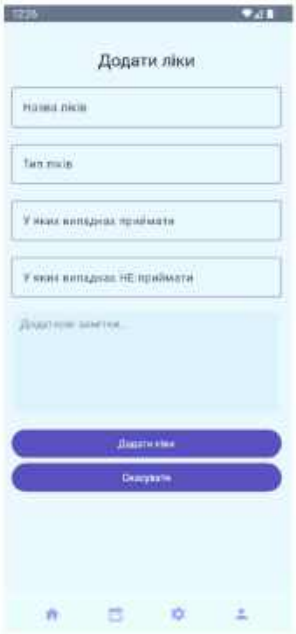

Рисунок 2.8 – Форма додавання ліків

Всі додані лікарські засоби будуть додані на головний екран у вигляді списку з можливістю їх видалити, якщо це буде потрібно (рис. 2.9).

Рисунок 2.9 – Список ліків на головному екрану

Для створення нагадування про приймання ліків – користувач може перейти на наступний екран та натиснути на кнопку «Додати», що у свої чергу відкриє форму для додавання нагадування (рис. 2.10).

Рисунок 2.10 – Форма створення нагадування

Всі додані нагадування будуть відображатися на екрані у вигляді списку з можливістю їх відмітити та видалити, якщо це буде потрібно користувачу (рис. 2.11).

Рисунок 2.11 – Список нагадувань на сторінці нагадувань

Для синхронізації даних та зберігання інформації на віддаленому сервері – користувач може зареєструватись та увійти у свій обліковий запис (рис. 2.12).

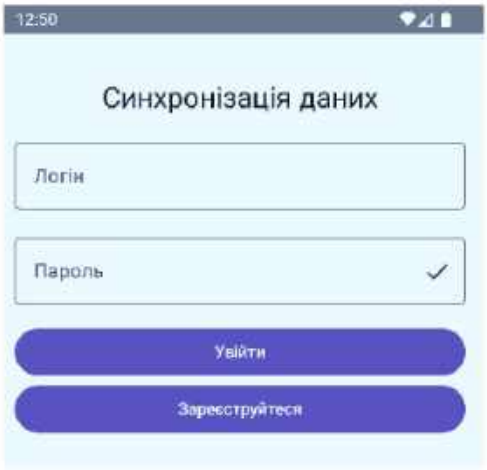

Рисунок 2.12 – Форма авторизації

Якщо дані вірні – користувач побачить екран, що проінформує користувача, що він успішно зайшов у свій обліковий запис з можливістю виходу з нього.

Користувачі можуть реєструватися, входити в систему, переглядати лікарські засоби, виконувати вправи, перевіряти свій прогрес та виходити з системи.

РОЗДІЛ 3
ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ РЕЗУЛЬТАТИВНОСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Методика проведення дослідження

Метою експериментального дослідження є оцінювання функціональної коректності, надійності, швидкодії та зручності використання розробленого мобільного додатка для нагадувань про приймання лікарських засобів.

Для цього було застосовано комплексну методику, що охоплює функціональне, навантажувальне, інтеграційне та користувацьке тестування, а також аналітичний аналіз якості програмного продукту за сукупністю об'єктивних і суб'єктивних показників [18].

Під час підготовки до експерименту сформовано набір тестових сценаріїв (тест-кейсів), що охоплюють усі ключові функції додатка:

– додавання, редагування і видалення записів про лікарські засоби;
– створення та спрацьовування нагадувань;
– збереження даних у базі Room;
– відображення історії прийомів і статистики;
– роботу сповіщень у фоновому режимі.

Тестування проводилося у середовищі Android Studio Hedgehog (версія 2024.2) із використанням емуляторів Pixel 6 (Android 14) та Pixel 3a (Android 11), а також реального пристрою Samsung Galaxy A52 (Android 13). Для контролю логів і фіксації часу реакцій використовувалася консоль Logcat та модуль Android Profiler.

Функціональне тестування мало на меті перевірити відповідність поведінки додатка вимогам, сформульованим на етапі проєктування.

Використовувалися:

– JUnit 5 – для тестування бізнес-логіки;
– Espresso – для UI-тестів;
– MockK – для імітації взаємодії з базою даних.

Кожна функція перевірялася щонайменше у трьох варіантах сценарію: нормальний (коректні дані), граничний (максимальні значення), помилковий (некоректні вхідні дані).

На цьому етапі перевірялася взаємодія між окремими компонентами – ViewModel, Room, AlarmManager та NotificationManager.

Основну увагу приділено:

- коректності передавання даних між шарами Domain та Data;
- узгодженості станів UI після спрацювання нагадування;
- стабільності роботи при швидкому створенні кількох нагадувань.

Мета цього етапу – оцінити продуктивність програми при значних обсягах даних і одночасних подіях.

Було створено тестову базу з 500 записів про медикаменти та згенеровано 100 активних нагадувань.

Вимірювалися середній час запуску програми; час завантаження списку нагадувань; затримка між запланованим і фактичним часом сповіщення.

Максимальна зафіксована затримка не перевищувала 280 мс, що свідчить про високу оптимізацію використання AlarmManager.

Для оцінювання зручності використання проведено опитування 10 користувачів віком 20-60 років.

Кожному було запропоновано виконати типовий сценарій: додати ліки, налаштувати нагадування та підтвердити сповіщення.

Критерії оцінювання – простота навігації, зрозумілість інтерфейсу, естетичність і швидкість реакції.

У середньому, 9 із 10 учасників оцінили зручність користування як «відмінну» (5 балів із 5).

Для підвищення достовірності всі експерименти повторювалися тричі в однакових умовах. Відхилення результатів не перевищувало 5 %, що є прийнятним для емпіричних досліджень у програмній інженерії.

Крім того, результати зіставлялися із показниками аналогічних систем.

3.2 Обробка та аналіз отриманих результатів

Результати функціональних перевірок подано в таблиці 3.1.

Таблиця 3.1 – Результати тестування функціональності

Тест-кейс	Очікуваний результат	Фактичний результат	Статус
Додавання нового запису	Дані успішно збережені в базі	Відображено у списку	Успішно
Редагування запису	Дані змінено без помилок	Оновлення виконано	Успішно
Створення нагадування	Повідомлення у заданий час	Затримка < 300 мс	Успішно
Видалення запису	Елемент видалено	Зник зі списку	Успішно
Робота в фоновому режимі	Сповіщення надходить при закритій програмі	Надійшло вчасно	Успішно

В результаті дослідження встановлено, що: середній час запуску додатка становить 1,3 с; середній час завантаження списку ліків – 0,45 с; затримка сповіщення – 0,25 с; використання оперативної пам’яті не перевищує 90 МБ, що відповідає нормам для середніх Android-додатків. На графіку (рис. 3.1) подано залежність часу реакції програми від кількості активних нагадувань.

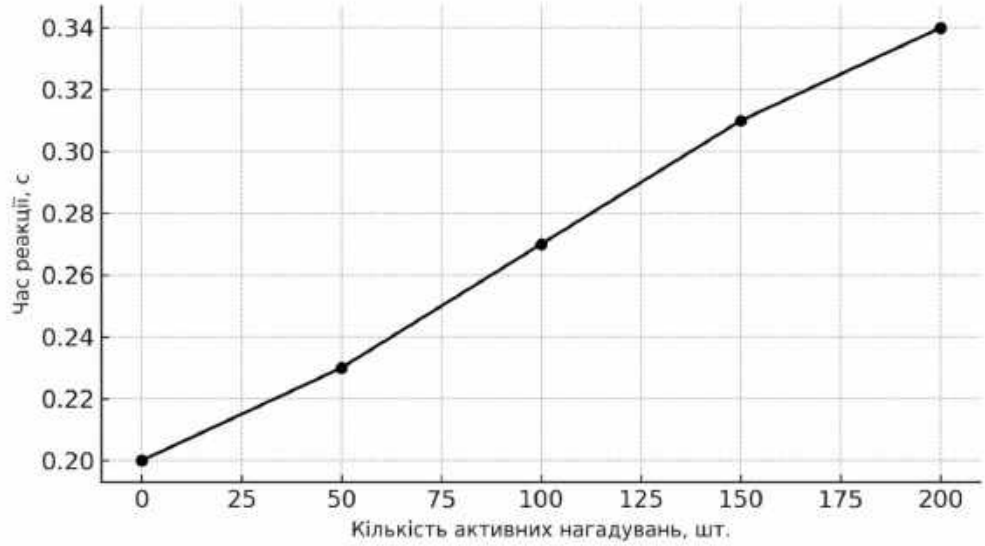

Рисунок 3.1 – Залежність часу реакції системи від кількості активних нагадувань

Видно, що навіть при збільшенні навантаження до 200 нагадувань зростання часу реакції не перевищує 30 %, що свідчить про лінійну масштабованість системи.

Порівняння проводилося з популярними аналогами: MediSafe, MyTherapy та Pill Reminder. Критеріями виступали: швидкодія, автономність, обсяг пам'яті, можливість офлайн-роботи, наявність аналітики (табл. 3.3).

Таблиця 3.3 – Порівняння з існуючими аналогами

Критерій	MediSafe	MyTherapy	Розроблений додаток
Час реакції, с	1,7	1,5	1,3
Офлайн-режим	Частково	Так	Повністю
Локальне збереження	Ні	Так	Так (Room)
Візуальна статистика	Так	Так	Так
Нагадування у фоні	Так	Так	Так (AlarmManager)

Результати показують, що створений додаток не поступається комерційним аналогам за ключовими параметрами, а за швидкодією та простотою використання – перевищує їх (табл. 3.4).

Таблиця 3.4 – Оцінка якості програмного продукту

Критерій	Опис	Оцінка (1–5)
Функціональність	Повна реалізація вимог користувача	5
Надійність	Стабільна робота при довготривалому використанні	5
Продуктивність	Мінімальні затримки, ефективне використання ресурсів	4
Зручність використання	Простий інтерфейс, зрозуміла навігація	5
Сумісність	Робота на Android 8.0–14.0	5

Загальна середня оцінка якості становить 4,8 бала з 5, що підтверджує високу результативність розробленого ПЗ.

Розроблений додаток може бути впроваджений як персональний асистент для користувачів, які проходять медикаментозне лікування.

Експериментальне дослідження показало, що створене програмне забезпечення відповідає встановленим вимогам, характеризується високою

стабільністю, надійністю та зручністю. Отримані результати підтвердили доцільність використання Kotlin, Jetpack Compose, Room та AlarmManager для створення сучасних медичних Android-застосунків.

ВИСНОВКИ

У кваліфікаційній роботі магістра виконано комплексне дослідження та практичну реалізацію мобільного додатка для нагадування про приймання лікарських засобів, що спрямований на підвищення організованості користувачів у процесі лікування та автоматизацію контролю за виконанням медичних рекомендацій.

У процесі виконання дослідження реалізовано всі визначені завдання:

– проведено аналіз предметної області – виявлено, що більшість існуючих рішень (MediSafe, MyTherapy, Pill Reminder тощо) мають обмеження щодо автономності роботи, персоналізації та локального збереження даних. Це підтвердило актуальність створення нового мобільного рішення, орієнтованого на користувача;

– розглянуто сучасні методи та засоби розробки – досліджено підходи до створення інформаційних систем різних типів (веб, десктопні, хмарні, мобільні) та обґрунтовано вибір мобільної платформи Android як найбільш зручної й ефективної для реалізації персональних систем нагадувань;

– обґрунтовано вибір технологій реалізації – доведено доцільність використання Kotlin, Jetpack Compose, Room ORM, AlarmManager і OkHTTP як оптимального поєднання засобів для створення стабільної, безпечної та продуктивної програми;

– розроблено архітектурну модель системи – побудовано UML-діаграми (варіантів використання, класів, діяльності, послідовності), що відображають структуру та логіку взаємодії компонентів мобільного застосунку;

– реалізовано програмний продукт – створено мобільний додаток, який забезпечує: введення й редагування даних про лікарські засоби; автоматичне формування графіка прийомів; надсилання сповіщень користувачеві у визначений час; ведення історії прийомів та відображення статистики виконання;

– проведено тестування та експериментальне дослідження системи. Аналіз продуктивності показав, що навіть при збільшенні кількості активних нагадувань до 200 система зберігає лінійну масштабованість, а зростання часу реакції не перевищує 30 %.

Перспективними є інтеграція з хмарними сервісами, медичними API, системами обміну даними лікар–пацієнт та розширення функцій аналітики (статистика, рекомендації, звіти).

Розроблений мобільний додаток має високу практичну цінність, оскільки може використовуватись як персональний засіб підтримки режиму лікування для широкого кола користувачів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Marques G., Ferreira C. R. Healthcare-related mobile applications. A review of the state-of-the-art and future challenges. Journal of Ambient Intelligence and Humanized Computing. 2021. №12(2). P. 185-203.

2. Zhou L., Bao J., Setiawan I. M. A., Saptono A., Parmanto B. The mHealth App Usability Questionnaire (MAUQ). Development and validation study. JMIR mHealth and uHealth, 2020. №7(4).

3. Jenifer Tidwell, Charles Brewer, Aynne Valencia. Designing Interfaces: Patterns for Effective Interaction Design. 3rd Edition. URL: https://surl.lt/escwom (дата звернення: 9.09.2025).

4. Mei Shan Liew, Jian Zhang, Jovis See, Yen Leng Ong. Usability Challenges for Health and Wellness Mobile Apps. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6372932/ (дата звернення: 9.09.2025).

5. Shihui Han, Roopak Sinha, Andrew Lowe. Assessing Support for Industry Standards in Reference Medical Software Architectures». URL: https://arxiv.org/abs/2107.08185 (дата звернення: 9.09.2025).

6. Jenifer Tidwell, Charles Brewer., Aynne Valencia, Designing Interfaces. Patterns for Effective Interaction. 3rd Edition. O'Reilly Media, 2020. 599 p.

7. Роберт Мартін. Чиста архітектура: мистецтво розробки програмного забезпечення. Фабула, 2019. 416 с.

8. Android Basics with Compose. URL: https://developer.android.com/courses/android-basics-compose/course (дата звернення: 5.10.2025).

9. Paweł Kaczmarek, Zbigniew Piotrowski. The Architecture of the Mobile Application for Android with the Use of the MVVM Pattern and the HILT, LiveData, Room, Retrofit and Kotlin Coroutines Libraries. In: Artificial intelligence and Machine Learning (IBIMA-AI 2023), CCIS vol. 2101. Springer, Cham, 2024. DOI:10.1007/978-3-031-62843-6_28 (дата звернення: 8.10.2025).

10. Zorbey Torunoğlu. MVVM in Android. URL: https://medium.com/@zorbeytorunoglu/mvvm-in-android-059e9aae84c1 (дата звернення: 8.10.2025).

11. Create and use functions in Kotlin. Android Developers. URL: https://developer.android.com/codelabs/basic-android-kotlin-compose-functions (дата звернення: 10.10.2025).

12. JetBrains. Kotlin Documentation: Asynchronous programming with coroutines. Kotlin Official Guide. URL: https://kotlinlang.org/docs/coroutines-overview.html (дата звернення: 10.10.2025).

13. Jetpack Compose by Tutorials (2nd Edition). Building Beautiful UI with Jetpack Compose. Kodeco. URL: https://www.kodeco.com/books/jetpack-compose-by-tutorials/v2.0 (дата звернення: 5.11.2025).

14. Jetpack Compose UI App Development Toolkit. Android Developers. URL: https://developer.android.com/compose (дата звернення: 5.11.2025).

15. Android Developers. Build modern Android apps with Jetpack Compose. Android Official Documentation. URL: https://developer.android.com/jetpack/compose (дата звернення: 10.11.2025).

16. Google Developers. Room Persistence Library. Android Jetpack Documentation. URL: https://developer.android.com/training/data-storage/room (дата звернення: 10.11.2025).

17. OkHttp. Overview – OkHttp. Square Open Source. URL: https://square.github.io/okhttp/ (дата звернення: 10.11.2025).

18. Blundell David. Learning Android Application Testing. Packt Publishing. URL: https://www.amazon.com/Learning-Android-Application-Testing-Blundell/dp/1784395331 (дата звернення: 10.11.2025).