МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ЕКОНОМІКО-ПРАВНИЧИЙ ФАХОВИЙ КОЛЕДЖ
ЗАПОРІЗЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ»

Циклова комісія математичних дисциплін та інформаційних технологій

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «РОЗРОБКА МОБІЛЬНОГО ДОДАТКУ ДЛЯ ТЕСТУВАННЯ ТА ОЦІНЮВАННЯ ЗНАНЬ»

Виконав:	здобувач освіти 4 курсу, групи К121-21
Спеціальність	121 Інженерія
	програмного забезпечення
	(шифр і спеціальність)
	Олександр ГАГІН
	(ім'я та ПРІЗВИЩЕ)
Керівник	Дмитро ДРАЄВСЬКИЙ
	(ім'я та ПРІЗВИЩЕ)
Рецензент	доцент кафедри програмної інженерії ЗНУ, к.т.н., доцент Віталій МУХІН
	(посада, науковий ступінь, вчене звання, ім'я та ПРІЗВИЩЕ)

Запоріжжя
2025

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ВСП «Економіко-правничий фаховий коледж ЗНУ»

Освітньо-професійний ступінь фаховий молодший бакалавр

Спеціальність 121 – Інженерія програмного забезпечення
(шифр і назва)

ЗАТВЕРДЖУЮ
Голова циклової комісії математичних дисциплін та інформаційних технологій

_____ Тетяна СМОЛЯНКОВА
(підпис)

“ 14 ” червня 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

ГАГІНУ Олександру Сергійовичу
(ПРІЗВИЩЕ ім’я та по–батькові)

1. Тема роботи Розробка мобільного додатку для тестування та оцінювання знань

Керівник роботи ДРАЄВСЬКИЙ Дмитро Сергійович
(посада, науковий ступінь, вчене звання, ПРІЗВИЩЕ ім’я по-батькові)

Затверджено наказом ВСП ЕПФК ЗНУ від « 26» листопада 2024 р № 1852-с

2. Строк подання студентом роботи 13.05.2025

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Навчальні мобільні додатки
2. Розробка проєкту
3. Програмна реалізація

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)
презентація до захисту

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання 01.12.2024

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи:		
	Постановка задачі	Березень 2025	виконано
2.	Збір вихідних даних, обробка методичних		
	та теоретичних джерел	Квітень 2025	виконано
3.	Розробка першого розділу:		
	Огляд інформаційних систем та технологій для їхнього створення	Квітень 2025	виконано
4.	Розробка другого розділу:		
	Проєктування архітектури, інтерфейсу, бази даних	Квітень 2025	виконано
5.	Розробка третього розділу:		
	Програмна реалізація мобільного додатку (Kotlin, Android Studio)	Травень 2025	виконано
6.	Оформлення і нормоконтроль кваліфікаційної роботи та перевірка на плагіат	Червень 2025	виконано
7.	Захист кваліфікаційної роботи	20.06.2025	виконано

Здобувач освіти _____ Олександр ГАГІН
(підпис) (ім’я та ПРІЗВИЩЕ)

Керівник роботи _____ Дмитро ДРАЄВСЬКИЙ
(підпис) (ім’я та ПРІЗВИЩЕ)

Нормоконтроль пройдено

Нормоконтролер _____ Юлія БОРИСОВСЬКА
(підпис) (ім’я та ПРІЗВИЩЕ)

РЕФЕРАТ

Кваліфікаційна робота: 31 сторінка, 5 ілюстрацій, 22 джерела.

Об’єкт дослідження – мобільні додатки для тестування знань.

Предмет дослідження – мобільний додаток для проведення міні НМТ.

Мета дослідження – створення мобільного додатку для тестування та оцінювання знань у форматі міні НМТ на базі Android з використанням мови Kotlin.

У межах кваліфікаційної роботи здійснено всебічну розробку мобільного додатку «Міні НМТ», який призначений для перевірки знань користувачів у форматі Національного мультипредметного тесту. Проєкт охоплює повний цикл створення програмного продукту: від ідеї та аналізу потреб до технічної реалізації та тестування функціональності.

Застосунок створено мовою програмування Kotlin з використанням сучасних технологій Android-розробки в середовищі Android Studio. Для побудови інтерфейсу застосовано Jetpack Compose, що забезпечило швидку, адаптивну та гнучку верстку, яка коректно працює на різних мобільних пристроях. Логіка програми побудована за архітектурною моделлю MVVM, а для роботи з локальною базою даних інтегровано бібліотеку Room.

У ході реалізації було створено функціональні модулі:

- авторизація та реєстрація користувача;
- вибір предмету для тестування;
- проходження тестів з фіксацією відповідей;
- оцінювання результатів із відображенням статистики.

Особлива увага приділена зручності користування – інтерфейс спроєктовано відповідно до принципів Material Design, а функціональність адаптована до реальних потреб учнів і студентів, які готуються до НМТ.

Результати розробки підтверджують, що створений додаток може ефективно використовуватися як додатковий засіб навчання та самоконтролю.

Він також має потенціал до подальшого вдосконалення – шляхом інтеграції з хмарними сервісами, розширення бази запитань та підключення аналітичного модуля.

Таким чином, у результаті дослідження було досягнуто поставлену мету – створено мобільний додаток, що поєднує сучасні технології Android-розробки з практичними освітніми завданнями.

KOTLIN, ANDROID STUDIO, JETPACK COMPOSE, ROOM, МІНІ НМТ, МОБІЛЬНИЙ ДОДАТОК, ТЕСТУВАННЯ ЗНАНЬ, АДАПТИВНИЙ ІНТЕРФЕЙС.

SUMMARY

Qualification work: 31 pages, 5 illustrations, 22 sources.

Object of study – mobile applications for knowledge testing.

Subject of study – a mobile application for conducting the mini-NMT (National Multi-Subject Test).

Research aim – to develop a mobile application for testing and evaluating knowledge in the mini-NMT format on the Android platform using the Kotlin programming language.

As part of the qualification project, a comprehensive development of the mobile application "Mini NMT" was carried out. This app is designed to assess users' knowledge in the format of the National Multi-Subject Test (NST). The project covers the full software development cycle – from concept and user needs analysis to technical implementation and functionality testing.

The application was developed using the Kotlin programming language and modern Android development technologies within the Android Studio environment. The user interface was built with Jetpack Compose, which enabled fast, adaptive, and flexible layouts that perform well across various mobile devices. The app's logic follows the MVVM architectural pattern, and local data storage is handled via the Room library.

During the implementation, the following key modules were developed:

- user registration and login;
- subject selection for testing;
- test-taking with answer tracking;
- result evaluation with statistical feedback.

Special attention was paid to usability – the interface was designed in accordance with Material Design principles, and functionality was tailored to the real needs of students preparing for the NMT.

The results of the development confirm that the application can be effectively

used as a supplementary learning and self-assessment tool. It also holds potential for future improvements, such as integration with cloud services, expansion of the question database, and the addition of an analytics module.

Thus, the main objective of the study has been achieved – a mobile application was created that combines modern Android development technologies with practical educational goals.

KOTLIN, ANDROID STUDIO, JETPACK COMPOSE, ROOM, MINI NMT, MOBILE APPLICATION, KNOWLEDGE TESTING, ADAPTIVE INTERFACE.

ЗМІСТ

Завдання на кваліфікаційну роботу студенту 2
Реферат 4
Summary 6
Вступ 9
1 Навчальні мобільні додатки 11
1.1 Що таке мобільний додаток 11
1.2 Технології для створення додатку 11
1.3 Поняття та призначення тестових систем 12
1.4 Основні функції сучасних систем тестування 12
1.5 Порівняння популярних платформ 13
1.6 Проблеми та обмеження існуючих рішень 13
1.7 Актуальність мобільного підходу 14
1.8 Висновок до розділу 1 14
2 Розробка мобільного додатку 15
2.1 Постановка задачі 15
2.2 Діаграма варіантів використання 15
2.3 Архітектура додатку 16
2.4 Інтерфейс користувача 17
2.5 UX/UI рішення 18
2.6 Room DAO 19
2.7 Висновок до розділу 2 21
3 Програмна реалізація 22
3.1 Вибір технологій та середовища розробки 22
3.2 Структура проєкту 22
3.3 Основні класи додатку 23
3.4 Взаємодія з Room DAO 26
3.5 Розмітка інтерфейсу в Jetpack Compose 27
3.6 Тестування додатку 28
3.7 Висновок до розділу 3 28
Висновки 29
Перелік використаних джерел 30

ВСТУП

У сучасному світі стрімкого розвитку цифрових технологій дедалі більшої популярності набувають мобільні додатки для тестування знань. Завдяки зручності, мобільності та широким можливостям персоналізації такі додатки стають невід’ємною частиною освітнього процесу як у середніх, так і у вищих навчальних закладах. Особливо актуальною є потреба у створенні ефективних засобів перевірки знань учнів та студентів у форматі національного мультипредметного тесту (НМТ), який дозволяє швидко і об'єктивно оцінити рівень знань із кількох предметів одночасно.

Метою цієї дипломної роботи є створення адаптивного Android-додатку для тестування знань з кількох навчальних предметів з використанням сучасної мови програмування Kotlin. Вибір Kotlin зумовлений її простотою, безпечністю, а також офіційною підтримкою компанії Google, що гарантує високу продуктивність і сумісність із сучасними пристроями на платформі Android.

У процесі розробки було реалізовано такі основні функції, як:

1. Реєстрація та авторизація користувачів.
2. Вибір предмета тестування.
3. Проходження тестів у форматі одного питання з чотирма варіантами відповідей та питання з текстовою відповід’ю.
4. Підрахунок і відображення результатів тестування з можливістю перегляду правильних та неправильних відповідей.

Особливу увагу було приділено адаптивному дизайну, що забезпечує комфортну взаємодію користувачів з додатком на будь-яких пристроях, незалежно від розміру екрана.

Об’єктом дослідження у дипломній роботі виступають сучасні програмні засоби, які використовуються для автоматизованого тестування та оцінювання знань. Під час роботи було детально проаналізовано як популярні

комерційні рішення, так і безкоштовні відкриті платформи, що дозволило визначити основні переваги та недоліки існуючих систем.

Предметом дослідження є безпосередньо розроблений мобільний додаток для проведення міні НМТ. Він покликаний розширити доступні засоби тестування, зробити процес перевірки знань більш інтерактивним, зручним і технологічно сучасним.

У процесі дослідження застосовувались такі методи, як порівняльний аналіз існуючих аналогів, проєктування з використанням сучасних підходів та архітектурних шаблонів, безпосередня програмна реалізація додатку, а також тестування функціоналу для підтвердження працездатності та відповідності поставленим вимогам.

1 НАВЧАЛЬНІ МОБІЛЬНІ ДОДАТКИ

1.1 Що таке мобільний додаток

Мобільний додаток – це програмне забезпечення, розроблене спеціально для використання на мобільних пристроях, таких як смартфони та планшети. Основні характеристики мобільних додатків – це зручність, швидкість доступу, інтерактивність та можливість персоналізації користувацького досвіду. Мобільні додатки встановлюються безпосередньо на пристрої користувачів та можуть працювати як онлайн, так і офлайн, залежно від функціональних особливостей [1].

1.2 Технології для створення додатку

Щоб зробити зручний мобільний додаток, ми обрали такі технології:

- Kotlin – мова програмування, яка офіційно підтримується Google для розробки Android-додатків. Вона проста у використанні, сучасна і добре працює з Android SDK [2].
- Android Studio – це офіційне інтегроване середовище розробки (IDE) від Google для створення Android-додатків. Тут ми пишемо код, тестуємо програму, перевіряємо її на віртуальних пристроях [3].
- Figma – онлайн-інструмент, який ми використали для створення макетів інтерфейсу. З його допомогою було зручно спланувати зовнішній вигляд додатку [4].

Ці інструменти сучасні, зручні та безкоштовні – саме те, що потрібно для студентського проєкту.

1.3 Поняття та призначення тестових систем

Інформаційні системи тестування – це спеціалізовані програмні рішення, які дозволяють проводити автоматизовану та стандартизовану перевірку знань користувача. Вони знайшли широке застосування в різних сферах, таких як освіта, бізнес, управління персоналом (HR), медицина, сертифікація професійних компетенцій та інші. Основне завдання таких систем полягає в тому, щоб швидко, ефективно і об’єктивно оцінити знання, уміння та навички користувачів, а також забезпечити наочне представлення результатів для подальшого аналізу [5].

Важливим аспектом тестових систем є здатність до масштабування та налаштування під конкретні потреби користувачів. Це дозволяє використовувати їх як у великих освітніх закладах з численними користувачами, так і в невеликих корпоративних середовищах для регулярного контролю професійних компетенцій.

1.4 Основні функції сучасних систем тестування

Сучасні системи тестування надають широкий спектр функціональних можливостей, серед яких особливо виділяються [6].

- проведення тестування у вигляді запитань із декількома варіантами відповідей;
- автоматизоване оцінювання результатів та миттєвий зворотний зв’язок;
- детальне відображення статистичних даних та аналітичних звітів;
- реєстрація та авторизація користувачів для персоналізації досвіду;
- можливість вибору предмету або окремої теми для проведення тестування;

- адаптивність інтерфейсу до різноманітних пристроїв (від пк до мобільних гаджетів);
- інтеграція з іншими інформаційними системами для розширення функціоналу.

1.5 Порівняння популярних платформ

У світі на сьогодні існує велика кількість платформ для проведення тестування знань, серед яких популярністю користуються Google Forms, Moodle, Kahoot, Quizlet та інші. Кожна платформа має свої особливості, переваги і недоліки:

- Google Forms вирізняється простотою використання, доступністю та інтеграцією з іншими продуктами Google, але має обмежені можливості кастомізації дизайну та аналітики [7].
- Moodle пропонує широкий спектр функцій, включаючи підтримку стандарту SCORM, глибоку аналітику та гнучке налаштування, однак потребує серйозних ресурсів для адміністрування та підтримки серверної частини [8].
- Kahoot має яскравий інтерактивний дизайн, що робить його особливо популярним серед школярів, але має обмежені можливості для складних аналітичних завдань [9].
- Quizlet пропонує зручні засоби для створення навчальних карток та швидкого тестування, однак також має обмеження в деталізованій аналітиці та офлайн-доступі [10].

1.6 Проблеми та обмеження існуючих рішень

Попри численні переваги, сучасні тестові системи мають ряд суттєвих обмежень, що можуть впливати на їх ефективність. Основні проблеми включають:

- недостатню адаптацію для мобільних пристроїв у деяких системах;
- обмежену інтерактивність та недостатню залученість користувачів;
- відсутність або обмежене використання локального збереження результатів;
- обмежені можливості для візуальної та функціональної персоналізації;
- відсутність офлайн-режиму у більшості популярних платформ, що обмежує їх використання у місцях з нестабільним інтернет-з’єднанням.

1.7 Актуальність мобільного підходу

Сучасний світ характеризується масовим поширенням смартфонів та планшетів, що робить мобільні додатки ефективними та затребуваними засобами тестування знань. Використання мобільних технологій дозволяє проходити тестування в будь-який зручний для користувача час, не залежати від доступності персонального комп'ютера або місця. Завдяки адаптивному дизайну, такі мобільні додатки зручні у використанні на пристроях з різними розмірами екранів, що значно покращує зручність і доступність освітніх ресурсів для широкого кола користувачів.

1.8 Висновок до розділу 1

У першому розділі було проаналізовано сучасні програмні засоби для автоматизованого тестування знань. Визначено, що мобільні додатки відіграють важливу роль у сучасному освітньому процесі, особливо завдяки своїй зручності, адаптивності та доступності. Розглянуто основні функції, переваги, недоліки та порівняння популярних платформ, що дозволило обґрунтувати доцільність створення власного мобільного додатку для міні НМТ з урахуванням актуальних потреб користувачів.

2 РОЗРОБКА МОБІЛЬНОГО ДОДАТКУ

2.1 Постановка задачі

Метою даної дипломної роботи є створення мобільного додатку для тестування та оцінювання знань користувачів у форматі міні НМТ. Додаток повинен бути зручним, адаптивним, швидким та функціональним. Для досягнення цієї мети було поставлено такі задачі:

1. Розробити простий та інтуїтивний інтерфейс користувача.
2. Реалізувати реєстрацію та вхід користувачів.
3. Забезпечити вибір предмета для проходження тестування.
4. Організувати відображення тестових запитань та варіантів відповідей.
5. Здійснювати підрахунок правильних/неправильних відповідей.
6. Показувати оцінку за результатами тестування.
7. Реалізувати адаптивний дизайн для різних типів пристроїв.

2.2 Діаграма варіантів використання

Для візуалізації взаємодії користувача з системою було побудовано діаграму варіантів використання (Use Case Diagram) (див. рис. 2.1).

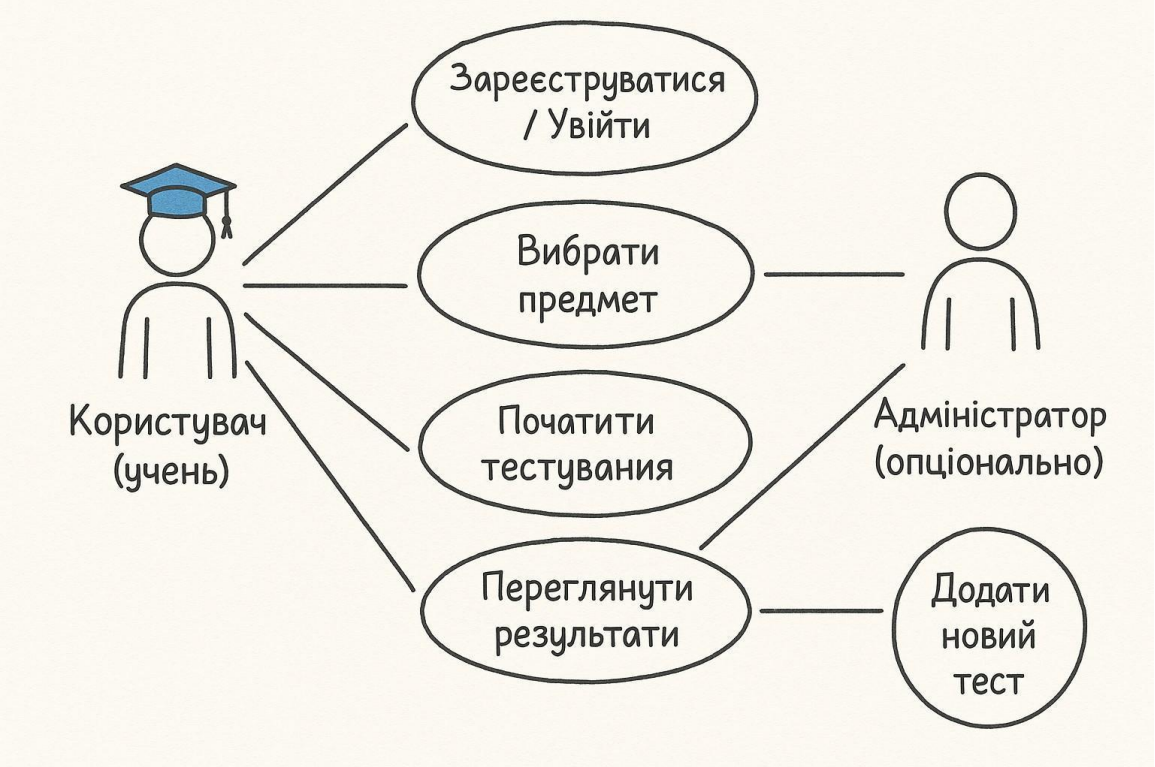


Рисунок 2.1 – Діаграма варіантів використання

2.3 Архітектура додатку

Було обрано архітектурний шаблон MVVM (Model-View-ViewModel) (див. рис. 2.2), який широко використовується в Android-розробці. Це дозволяє розділити логіку бізнес-процесів від візуального інтерфейсу. Основні компоненти [11].:

- Model – відповідає за зберігання даних та взаємодію з джерелами даних;
- View – відображає інтерфейс користувача;
- ViewModel – містить логіку управління даними та станами інтерфейсу.

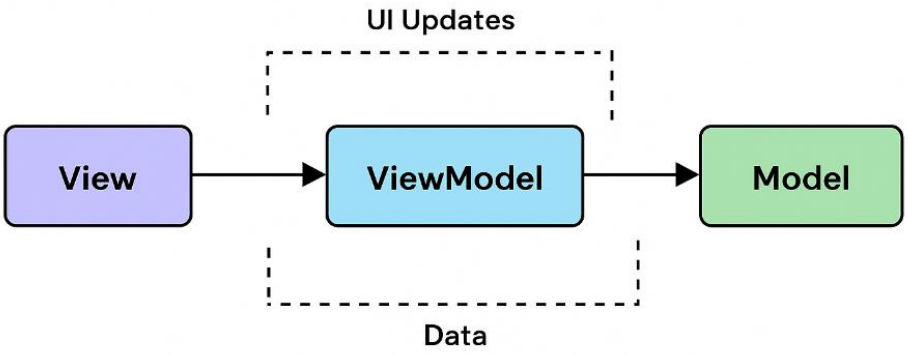


Рисунок 2.2 – Архітектура додатку

2.4 Інтерфейс користувача

У додатку реалізовано чотири основні екрани (див. рис. 2.3):

1. Екран вітання та входу/реєстрації – вхід для існуючих користувачів або створення нового акаунта.

2. Меню вибору предмета – користувач обирає предмет для тестування.

3. Екран тестування – відображається одне запитання та чотири варіанти відповідей.

4. Екран результатів – показ оцінки, правильних і неправильних відповідей.



Рисунок 2.3 – Основні екрани додатку

2.5 UX/UI рішення

Під час проєктування дизайну було враховано принципи адаптивності, контрастності, мінімалізму та простоти навігації. Усі елементи мають оптимальні розміри для зручного натискання на сенсорних екранах. Кольори

підібрані з урахуванням WCAG (Web Content Accessibility Guidelines).

Основні UX/UI рішення:

- мінімальна кількість натискань для запуску тесту (максимум 2);
- візуальний прогрес бар під час проходження тесту;
- миттєвий фідбек після вибору відповіді;
- збереження результатів локально (у базі даних або sharedpreferences).

2.6 Room DAO

Бібліотека Room – це офіційна ORM (Object Relational Mapping) для Android, представлена Google у 2017 році як частина Android Architecture Components. Вона забезпечує високорівневу абстракцію над SQLite, спрощуючи роботу з локальними базами даних у додатках [12].

2.6.1 Основні компоненти Room

Room складається з трьох основних частин [12]:

1. @Entity – позначає клас, що відповідає таблиці в базі даних.
2. @Dao – інтерфейс для доступу до даних, що містить методи для виконання SQL-запитів.
3. @Database – абстрактний клас, що розширює RoomDatabase і об'єднує всі DAO та Entity.

2.6.2 Переваги та недоліки Room

Переваги [13]:

- простота використання: завдяки аннотаціям та генерації коду, room значно спрощує роботу з базою даних;

- перевірка sql-запитів під час компіляції: це дозволяє виявити помилки на ранньому етапі розробки;
- інтеграція з livedata та rxjava: підтримка реактивного програмування для спостереження за змінами в базі даних;
- автоматичне управління потоками: room виконує запити в окремих потоках, що забезпечує безпечну роботу в багатопотоковому середовищі;
- підтримка міграцій: полегшує оновлення схеми бази даних без втрати даних.

Недоліки [13]:

- обмежена підтримка зв'язків: room не обробляє автоматично складні зв'язки між сутностями, такі як "один до багатьох" або "багато до багатьох"; розробнику потрібно вручну реалізовувати ці зв'язки за допомогою додаткових класів або конвертерів типів;
- крива навчання: для початківців може бути складно освоїти всі можливості та нюанси room;
- проблеми з продуктивністю при неправильному використанні: неправильне використання room може призвести до проблем з продуктивністю, наприклад, виконання запитів у головному потоці.

2.6.3 Аналоги Room

Існують інші ORM-бібліотеки для Android, які можуть служити альтернативою Room:

- ObjectBox: швидка та легка база даних з підтримкою реактивного програмування [14].
- ORMLite: легка ORM-бібліотека, що підтримує як Android, так і Java SE [15].
- DBFlow: гнучка ORM-бібліотека з підтримкою складних запитів та міграцій [16].

2.7 Висновок до розділу 2

Другий розділ присвячено проектуванню структури мобільного додатку. Було сформульовано технічне завдання, розроблено діаграму варіантів використання та архітектуру системи за шаблоном MVVM. Визначено основні екрани інтерфейсу та принципи їх побудови відповідно до UX/UI вимог. Також розглянуто використання бібліотеки Room для роботи з базою даних. Отримані результати створили ґрунтовну основу для подальшої програмної реалізації проєкту.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Вибір технологій та середовища розробки

Для реалізації мобільного додатку було обрано мову програмування Kotlin та середовище Android Studio (див. рис. 3.1). Kotlin є офіційно рекомендованою мовою для Android-розробки, яка має сучасний синтаксис, підтримку функціонального стилю та чудову інтеграцію з Android SDK.

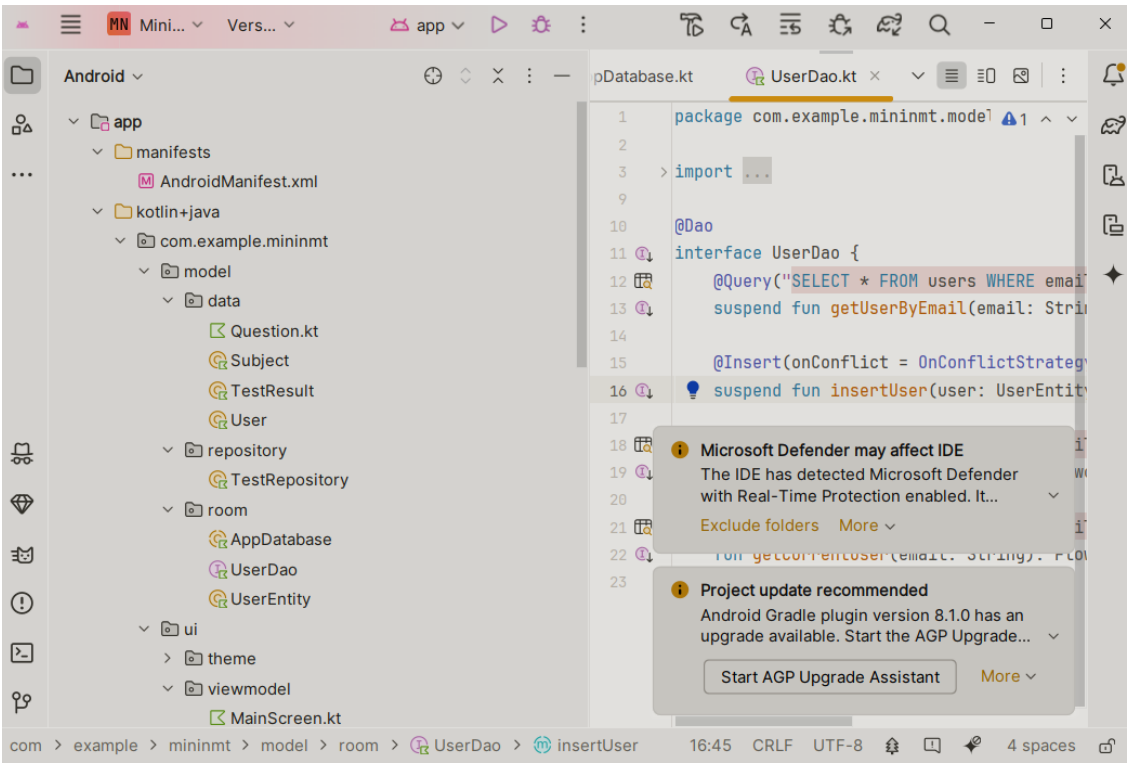


Рисунок 3.1 – Середовище розробки Android Studio

3.2 Структура проєкту

Мобільний додаток реалізовано у середовищі Android Studio мовою програмування Kotlin. Проєкт організовано за архітектурною моделлю MVVM (Model-View-ViewModel), що забезпечує розділення відповідальностей,

зручність у підтримці коду, легке тестування.

Структура каталогу проєкту:

- /model – класи даних (User, Question, Subject, TestResult) та Room база даних (DAO, Database).
- /repository – логіка доступу до даних.
- /viewmodel – ViewModel-и для кожного екрану.
- /ui – активності, фрагменти та layout-файли для інтерфейсу користувача.

3.3 Основні класи додатку

3.3.1 Реєстрація та вхід користувача

Функціонал реєстрації користувача включає введення логіну та пароля з валідацією введених даних. Паролі зберігаються у зашифрованому вигляді. Для цього використовується бібліотека database або інші методи шифрування.

```
{
    when (currentScreen) {
        Screen.Registration -> {
            RegistrationScreen(
                onRegistrationComplete = { name, email, password ->
                    scope.launch {
                        val user = UserEntity(email, name, password)
                        database.userDao().insertUser(user)
                        getSharedPreferences("user_prefs", MODE_PRIVATE)
                            .edit()
                            .putString("user_email", email)
                            .apply()
```

```
                    currentUser = user
                    userId = email
                    userName = name
                    currentScreen = Screen.Main
                }
            // Вміст картки предмета          }
        }
    }
}
```

3.3.2 Вибір предмета

Головний екран мобільного додатку реалізовано у вигляді Composable-функції MainScreen, яка відповідає за відображення списку предметів. Ця функція приймає два параметри: список об'єктів типу Subject та обробник події onSubjectSelected, що викликається при виборі користувачем певного предмета.

```
@Composable
fun MainScreen(
    subjects: List<Subject>,
    onSubjectSelected: (Subject) -> Unit
) {
    // ...
    LazyVerticalGrid(...) {
        items(subjects) { subject ->
            Card(
                onClick = { onSubjectSelected(subject) },
                modifier = Modifier
                    .fillMaxWidth()
                    .height(160.dp)
            ) {
                // Вміст картки предмета          }
        }
    }
}
```

3.3.3 Проходження тесту

Після вибору предмета користувач переходить до екрану тестування, де кожне запитання відображається на окремому екрані разом із варіантами відповідей. Дані для тесту завантажуються з локальної бази даних, зокрема з таблиці Question, фільтруючись за ідентифікатором обраного предмета.

1. Поточне запитання:

```
val currentQuestion = questions[currentQuestionIndex]
```

2. Вибір відповіді користувачем:

```
(currentAnswer = option
answers[currentQuestion.id] = option
```

3. Перехід до наступного запитання:

```
if (currentQuestionIndex < questions.size - 1) {

    Button(

      onClick = { currentQuestionIndex++ },

      enabled = hasAnswer
) {
    Text("Далі")
}
}
```

3.3.4 Обчислення результатів тестування

Після завершення всіх запитань користувач потрапляє на екран результатів. Обчислюється кількість правильних відповідей, відсоток правильності та оцінка за шкалою.

```
        val percentage = if (totalQuestions > 0) {

            (correctAnswers.toDouble() / totalQuestions.toDouble()) * 100

        } else {
            0.0
        }

        val grade = when {

            percentage >= 90 -> "Відмінно"
            percentage >= 75 -> "Добре"
            percentage >= 60 -> "Задовільно"
            else -> "Незадовільно"
        }
```

3.4 Взаємодія з Room DAO

Кожна таблиця в базі даних має відповідний DAO-інтерфейс, що описує методи взаємодії з даними. DAO містить методи вставки, оновлення, видалення та вибірки.

```
        @Dao
        interface UserDao {
            @Query("SELECT * FROM users WHERE email = :email LIMIT 1")
            suspend fun getUserByEmail(email: String): UserEntity?
            @Insert(onConflict = OnConflictStrategy.Companion.REPLACE)
            suspend fun insertUser(user: UserEntity)
            @Query("SELECT * FROM users WHERE email = :email AND password =
:password LIMIT 1")
            suspend fun login(email: String, password: String): UserEntity?
            @Query("SELECT * FROM users WHERE email = :email LIMIT 1")
            fun getCurrentUser(email: String): Flow<UserEntity?>
                }
```

3.5 Розмітка інтерфейсу в Jetpack Compose

Усі інтерфейси додатку були реалізовані з використанням Jetpack Compose – сучасного підходу до розробки UI в Android, що дозволяє описувати інтерфейс без XML, безпосередньо в коді на Kotlin. Компонування елементів здійснюється за допомогою функцій-компонентів (@Composable) та таких структур як Column, Row, Box, що забезпечують гнучке, адаптивне та декларативне оформлення екранів [17].

Приклад розмітки екрану тестування:

```
Column(    modifier = Modifier
        .fillMaxSize()
        .padding(16.dp)
) {

    Row(
        modifier = Modifier

            .fillMaxWidth()
            .padding(bottom = 16.dp),

        horizontalArrangement = Arrangement.SpaceBetween,
        verticalAlignment = Alignment.CenterVertically

    ) {
        Text(

            text = "Час: ${formatTime(elapsedTime)}",
            style = MaterialTheme.typography.titleMedium
        )
        IconButton(onClick = { showExitDialog = true }) {
            Icon(Icons.Default.ExitToApp, contentDescription = "Вихід")
        }
    }
```

3.6 Тестування додатку

Для перевірки працездатності додатку було проведено ручне тестування на декількох пристроях з різними версіями Android. Тестування включало:

- реєстрацію/вхід користувача;
- вибір предмета та проходження тесту;
- перевірку правильності обчислення результатів;
- тестування адаптивності дизайну;
- перевірку роботи з базою даних (читання, запис, очищення).

Усі виявлені помилки були виправлені. Застосунок працює стабільно, без аварійного завершення роботи.

3.7 Висновок до розділу 3

У третьому розділі описано реалізацію мобільного додатку в середовищі Android Studio з використанням мови Kotlin та технології Jetpack Compose. Детально розглянуто структуру проєкту, основні класи, логіку авторизації, проходження тестів та обчислення результатів. Було проведено ручне тестування додатку, яке підтвердило його стабільну роботу та відповідність поставленим функціональним вимогам. Отже, реалізація успішно демонструє можливість використання додатку в реальному навчальному процесі

ВИСНОВКИ

У кваліфікаційній роботі було розглянуто процес створення мобільного додатку для тестування та оцінювання знань у форматі міні НМТ. Проведено аналіз існуючих систем тестування, визначено основні вимоги до додатку, обрано технології розробки та реалізовано повноцінний прототип.

У результаті виконано наступне:

1. Досліджено актуальні інформаційні системи тестування.
2. Сформовано технічне завдання для мобільного додатку.
3. Розроблено інтерфейс користувача з чотирма основними екранами.
4. Реалізовано додаток мовою Kotlin в Android Studio.
5. Реалізовано проходження тесту та підрахунок оцінки.
6. Застосовано адаптивний дизайн для зручної роботи на різних пристроях.

Створений мобільний додаток є зручним інструментом для самостійного тестування знань учнів і студентів. Проєкт може бути розширений у майбутньому – додаванням нових предметів, складніших форматів запитань, інтеграцією з серверною частиною та статистикою результатів.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Мобільний додаток. URL: https://uk.wikipedia.org/wiki/Мобільний_застосунок (дата звернення: 20.04.2025).

2. Kotlin Documentation. URL: https://kotlinlang.org/docs/home.html (дата звернення: 20.04.2025).

3. Документація Android Developers. URL: https://developer.android.com/ (дата звернення: 20.04.2025).

4. Figma. URL: https://www.figma.com/ui-design-tool/ (дата звернення: 25.04.2025).

5. Програмний модуль інтерактивного тестування знань. URL: https://itce.com.ua/web/uploads/pdf/918-%D0%A2%D0%B5%D0%BA%D1%81%D1%82%20%D1%81%D1%82%D0%B0%D1%82%D1%82%D1%96-890-1-10-20230315.pdf (дата звернення: 12.05.2025).

6. Онлайн тестування як засіб оцінювання якості знань студентів. URL: https://www.aphn-journal.in.ua/archive/59_2023/part_3/52.pdf (дата звернення: 15.05.2025).

7. Google Forms. URL: https://www.google.com/forms/about/ (дата звернення: 06.05.2025).

8. Moodle. URL: https://moodle.org/ (дата звернення: 06.05.2025).

9. Kahoot. URL: https://kahoot.com/ (дата звернення: 06.05.2025).

10. Quizlet URL: https://quizlet.com/ (дата звернення: 06.05.2025).

11. MVVM architecture pattern. URL: https://developer.android.com/topic/libraries/architecture/viewmodel (дата звернення: 05.05.2025).

12. Бібліотека «Room» для початківця Android-розробника. URL: https://kotlinlang.org/docs/home.html (дата звернення: 20.04.2025).

13. Room библиотека, часть Android Jetpack. URL: https://54origins.com/ru/technologies/room/ (дата звернення: 10.05.2025).

14. ObjectBox. URL: https://objectbox.io/ (дата звернення: 10.05.2025).

15. ORMLite. URL: https://ormlite.com/ (дата звернення: 10.05.2025).

16. DBFlow. URL: https://guides.codepath.com/android/DBFlow-Guide (дата звернення: 10.05.2025).

17. Jetpack Compose. URL: https://developer.android.com/compose (дата звернення: 10.05.2025).

Декларація
академічної доброчесності
здобувача освіти ВСП «Економіко-правничого фахового коледжу ЗНУ»

Я, Гагін Олександр Сергійович, здобувач освіти 4 курсу, спеціальності/освітньо-професійної програми Інженерія програмного забезпечення/Розробка програмного забезпечення, групи К121-21, адреса електронної пошти s.gagin74@gmail.com

- підтверджую, що написана мною кваліфікаційна робота на тему «Розробка мобільного додатку для тестування та оцінювання знань» відповідає вимогам академічної доброчесності та не містить порушень, що визначені у ст. 42 Закону України «Про освіту», зі змістом яких ознайомлений;

- заявляю, що надана мною для перевірки електронна версія роботи є ідентичною її друкованій версії;

- згоден на перевірку моєї роботи на відповідність критеріям академічної доброчесності у будь-який спосіб, у тому числі за допомогою інтернет-системи, а також на архівування моєї роботи в базі даних цієї системи.

Дата 13.06.2025 Підпис _____ Олександр ГАГІН

Дата _____ Підпис _____ Дмитро ДРАЄВСЬКИЙ