Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)
Кафедра Інформатики
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

РОЗРОБКА ІНТЕРАКТИВНИХ ТЕСТІВ З ІНФОРМАТИКИ ДЛЯ МОБІЛЬНИХ ПРИСТРОЇВ
(тема)

Виконав:
студент 4 курсу, групи ІТІНФ-19-2
Жилін М.Ю.
(прізвище, ініціали)

Спеціальності 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

Керівник доц. Сакало Є.С.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Кобилін О.А.
(підпис) (прізвище, ініціали)

2023 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)

Кафедра Інформатики
(повна назва)

Рівень вищої освіти перший (бакалаврський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Жиліну Михайлу Юрійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка інтерактивних тестів з інформатики для мобільних пристроїв

затверджена наказом університету від 15 травня 2023 року № 474 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 29 травня 2023 р.

3. Вихідні дані до роботи науково-методична та науково-технічна література, матеріали конференцій, дані інтернет-мережі, мова програмування Java, середовище розробки Android Studio, дані про комп'ютерний зір та аналіз Qr кодів.

4. Перелік питань, що потрібно опрацювати в роботі
1. Аналіз процесу розробки Android застосунків.
2. Математичні моделі зберігання даних та комп'ютерного зору.
3. Програмна реалізація застосунку на Android для створення і проходження інтерактивних тестів.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) Актуальність проблеми розробки мобільних застосунків, постанова задачі, математичні методи під час розробки мобільних застосунків, програмна реалізація застосунку, ілюстрації з роботи застосунку.

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Консультант з дотримання діючих стандартів та норм	Доцент Творошенко І.С.		

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	10.04.2023	
2	Аналіз завдання, підбір літератури	11.04.23-17.04.23	
3	Аналіз літератури з досліджуваної проблеми	18.04.23-20.04.23	
4	Аналіз засобів розробки мобільних застосунків	21.04.23-30.04.23	
5	Проєктування застосунку	01.05.23-14.05.23	
6	Програмна реалізація	15.05.23-23.05.23	
7	Оформлення пояснювальної записки	24.05.23-26.05.23	
8	Перевірка на плагіат	27.05.23	
9	Рецензування	28.05.23	
10	Підготовка презентації та доповіді	29.05.23-31.05.23	
11	Занесення роботи в електронний архів	01.06.23	
12	Попередній захист кваліфікаційної роботи	06.06.23	

Дата видачі завдання 10 квітня 2023 р.

Студент _____
(підпис)

Керівник роботи _____ _____ доц. Сакало Є.С._____
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 63 с., 2 табл., 17 рис., 30 джерел.

МОБІЛЬНІ ПРИСТРОЇ, ІНФОРМАТИКА, ІТНЕРАКЦІЯ, ANDROID, IOS, QR КОД.

Об’єктом роботи є застосунок до мобільних пристроїв.

Метою роботи є створення застосунка у якому можливо реалізувати створення та проходження різноманітних тестів з інформатики використовуючи мобільні пристрої при зберіганні автономії від мережі інтернет.

Використано методи класичної розробки мобільних застосунків та аналітичного обґрунтування. Проведено дослідження інтеракції користувачів с різними інтерфейсами для пророоблення найкращого UX. Вивчені найновітніші способи інтеракції користувача з мобільним пристроєм. Також використаний реляційний підхід до бази даних.

MOBILE DEVICES, IT, INTERACTION, ANDROID, IOS, QR CODE.

The object of the work is an application for mobile devices.

The method of operation is the creation of an application in which it is possible to implement the creation and distribution of various IT tests using mobile devices while maintaining autonomy from the Internet.

The methods of classical development of mobile applications and analytical reasoning are used. Conducted user interaction research with various interfaces to develop the best UX. The latest methods of user interaction with a mobile device are studied. A relational approach to the database is also used.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ОС – операційна система

IDE – Integrated Development Environment (інтегроване середовище розробки)

APK – Android Package Kit (набір пакетів Android, це формат файлу, який використовується для розповсюдження та встановлення застосунків на операційну систему Android)

Activity – вікно з яким взаємодіє користувач

UI – User Interface (інтерфейс користувача)

UX – User Experience (досвід користувача, який він отримує під час використання застосунку)

ASCII – American Standard Code for Information Interchange (американський стандартний код для обміну інформацією)

Патерн – загальний підхід до вирішення задач

ВСТУП

Щодня кількість користувачів мобільних пристроїв збільшується, і мобільні програми стають все більш популярними. В освіті також з’являється все більше технологій, які дозволяють студентам та викладачам використовувати мобільні пристрої для навчальних цілей. Розробка інтерактивних тестів з інформатики для мобільних застосунків – це одна з таких технологій, яка може суттєво покращити освітній процес.

Інтерактивні тести пропонують студентам можливість перевірити свої знання та практичні навички з інформатики. Такі тести можуть бути створені з використанням різних технологій та інструментів, які дозволяють розробникам створювати цікаві та креативні питання та завдання. Крім того, інтерактивні тести можуть допомогти викладачам покращити якість своїх занять, дозволяючи їм легко оцінювати знання студентів та підлаштовувати навчальний процес під їхні потреби.

У цьому контексті розробка інтерактивних тестів з інформатики для мобільних застосунків стає все більш важливою для сучасної освітньої системи. У цій галузі є багато перспектив для розвитку та покращення освіти.

1 ОГЛЯД ОСНОВНИХ МЕТОДІВ РОЗРОБКИ МОБІЛЬНИХ ЗАСТОСУНКІВ НА ANDROID

1.1 Актуальність сфери розробки мобільних застосунків та її роль у сфері навчання

З кожним роком кількість користувачів смартфонів зростає. Так за статистикою в Україні на 2013 рік лише 9% населення мали досвід користування смартфоном, а вже у 2019 ця частка сягнула межі у 66% громадян. У Євросоюзі цей відсоток є ще вищим. Так наприклад у 2019 році відсоток громадян з смартфоном дорівнював 85%.

Згідно статистики отриманої Київським міжнародним інститутом соціології за липень 2022 року частка користувачів смартфонів серед рецензентів віком від 18 до 29 років була у районі 96% [1]. Це свідчить про зацікавленість молоді у використанні сучасних технологій, і її інтегрованості у цю сферу.

З початку карантину 9 грудня 2020 року, інтеграція сучасних технологій у навчальний процес суттєво збільшила свої обсяги. Заняття в багатьох школах та університетах перейшли на дистанційну форму навчання. Великий відсоток учнів та студентів почали використовувати свою смартфони та комп’ютери для навчального процесу, але для досконалої його реалізації бракувало необхідних інструментів. Тому перший час інтеграція проводилася не дуже вдало та зручно. Постала потреба у розробці відповідних застосунків, для налагодження зручності та якості дистанційної освіти.

Також збільшилась необхідність учнів у знаннях у сфері інформаційних технологій для вирішення локальних проблем з обладнання та адаптації підтвердження результатів своєї роботи викладачу. З метою облегшення інтеграції учнів у таку форму навчального процесу. Тому

з’явилася потреба у створенні застосунків для полегшення та зручності навчального процесу.

1.2 Процес розробки мобільного застосунку

Розробка мобільного застосунку – це процес створення програмного забезпечення, яке націлене на використання на мобільних пристроях, планшетах та інших девайсах, які здатні запустити мобільні ОС.

Найрозповсюдженішими ОС є IOS та Android. Мобільні застосунки здатні виконувати різні функції такі як соціальні мережі, месенджери, ігри, утиліти, онлайн банкінг, навчання та інші.

Переважна більшість користувачів використовує пристрої на базі системи Android. Приблизно 70% ринку мобільних пристроїв належать до ОС Android. Також перевагою даної ОС є більша доступність пристроїв за рахунок нижчої ціни на них [2].

Серед переваг IOS є більша мінімальна продуктивність девайса. Також процес розробки застосунку під IOS є більш простим з точки зору того, що кількість моделей пристроїв на цій ОС значно менша.

Існують декілька підходів до розробки мобільних застосунків. Одним з них є крос-платформений підхід. Його суть полягає у тому, що код, написаний для застосунку може запускатися на усіх розповсюджених ОС. Очевидною перевагою цього підходу є можливість виходу застосунку на усі доступні ринки застосунків. Також перевагою можна назвати менший час для написання коду для всіх ОС [3]. З мінусів можна виділити те, що важче буде адаптувати застосунок під усі можливі пристрої.

Наступним підходом буде розробка окремих застосунків для різних ОС. Основним плюсом цього підходу є краща оптимізація застосунку під кожен окремий девайс. Із мінусів можна виділи те що процес розробки та

підтримки застосунку буде важчим та довшим, оскільки необхідно розроблювати 2 повноцінних застосунки.

Третім варіантом буде розробка застосунку лише для однієї платформи. Цей варіант має наступні переваги: полегшений процес розробки, можливість оптимізувати застосунок для всіх пристроїв обраної ОС. Із мінусів можна виділити те що, застосунок не буде доступним частині користувачів, що є негативним фактором, якщо стоїть потреба максимальної монетизації застосунку.

1.2.1 Основні етапи розробки мобільного застосунку

Розбробка будь-якого мобільного застосунку складається з наступних етапів:

– ідея проєкту. Як і усі проєкти розробка мобільного застосунку починається з створення ідеї застосунку. На цьому етапі необхідно визначати основний функціонал проєкту;

– вибір платформи розповсюдження. Чи буде він доступним на всіх ОС, чи буде крос-платформеним, чи буде націлений лише на одну конкретну платформу;

– модель розповсюдження застосунку. Чи буде він безкоштовним чи користувач повинен буде заплатити для доступу до застосунку;

– визначення архітектури проєкту. З яких компонентів буде скаладатися застосунок;

– безпосередньо розробка. На цьому етапі пройде кодування функціоналу проєкту;

– тестування. На цьому етапі перевіряється правильність функціонування застосунку;

– створення документації. Необхідно створити документацію для полегшення використання застосунку кінцевим користувачем;

– випуск застосунку у цифровий ринок. На цьому етапі застосунок публікується у відповідних магазинах застосунків.

1.3 Специфіка розробки застосунку для Android

Android – це операційна система для мобільних пристроїв, яку створив Google. Оригінальною метою Android було створення платформи для мобільних телефонів, але з часом вона стала використовуватися на різноманітних пристроях, таких як планшети, ноутбуки, телевізори та інші пристрої Інтернету речей.

Android базується на ядрі Linux та забезпечує широкий спектр можливостей для розробників та користувачів. Він містить багато різноманітних функцій, таких як сповіщення, налаштування та керування застосунками, багатофункціональний екран блокування, підтримку різних типів мультимедіа, можливості для розробки графічних ігор та застосунків штучного інтелекту, технології розпізнавання голосу, вбудовані сервіси Google та інші [4].

Важливо зазначити також важливість вибору IDE для найзручнішого процесу розробки застосунку.

1.3.1 Вибір IDE

Найрозповсюдженішою IDE для розробки можна вважати Android Studio від Google (рис. 1.1). Основними перевагами цього варіанту можна виділити підтримку розробника ОС даної IDE. Завдяки цьому можна вільно користуватися головними складовими Android платформи, а саме: Activity,

Service, Broadcast Receiver, Content Provider. Наступною перевагою можна виділити інтуїтивно зрозумілий інтерфейс який пришвидшує розробку застосунку. Завдяки інтерфейсу створеному спеціально для розробки Android застосунків, ця IDE дозволяє чітко корегувати кінцевий інтерфейс за допомогою вбудованих елементів корегування відображення (рис. 1.2). Завдяки цим інструментам можливо використовувати конструктор інтерфейсу можливо розробити інтерфейс не пишучи код напряму.

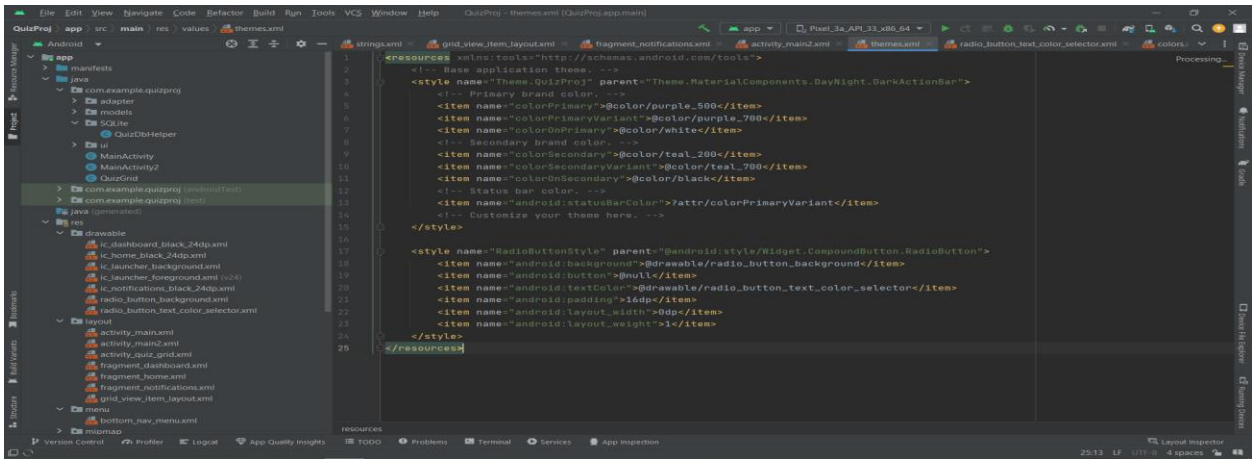

Рисунок 1.1 – Інтерфейс Android Studio

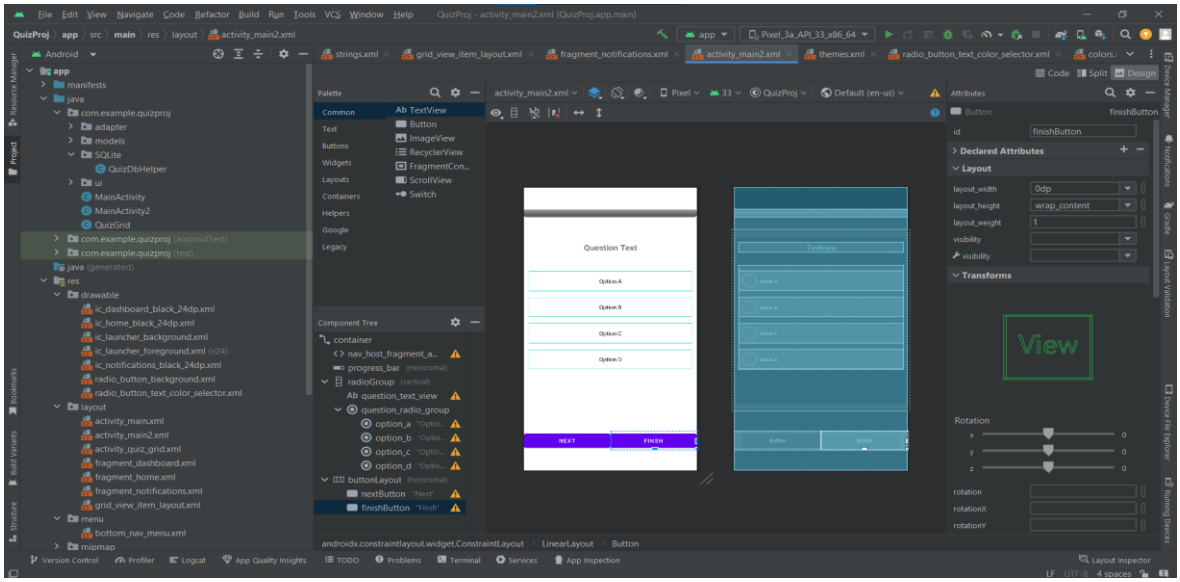

Рисунок 1.2 – Інтерфейс інструменту для створення інтерфейсу

В Android Studio також вбудовані засоби збирання застосунків, що дозволяє максимально швидко збирати та встановлювати APK файли створених Android застосунків. Ще одним перевагою можна вважати інтеграцію проєкту з сервісом Git, що дозволяє більш вільно підходити до процесу розробки та менеджменту проєкту. Також Android Studio надає своєму користувачу зручні та зрозумілі інструменти тестування та відлагоджування створеного застосунку, що пришвидшує процес написання коду. Також найбільшою перевагою Android Studio можна вважати наявність вбудованого емулятору, яких дозволяє запустити застосунок прямо на комп'ютері розробника, що дозволяє кожен раз не збирати APK файл та не завантажувати його на телефон, а просто запустити емуляцію необхідної моделі пристрою.

Наступним варіантом IDE можливо назвати IntellijJ IDEA (рис. 1.3). Загалом ця середа розробка є більш загальною версією Android Studio від Google. Тому слід зазначити інструменти які відсутні у цій середі розробки. Основним недоліком є відсутність вбудованого емулятору, що дуже сильно присповільнює процес тестування та розробки застосунку. Наступним недоліком буде відсутність вбудованих інструментів для роботи за інтерфейсом Android застосунку, що відіграє свою ролі при тестування та проєктуванні інтерфейсу для кінцевого користувача. Інші переваги Android Studio дійсні й для цієї середи розробки.

Іншою популярною IDE можна вважати Eclipse (рис. 1.4).

Загалом Eclipse має ті ж переваги що й IntelliJ IDEA, за винятком інструментів налагодження, підтримки Kotlin за замовчування, більш слабкий аналізатор коду та системи завершення коду.

Отже обираючи IDE для розробки Android застосунку загальною рекомендацією можна вважати використання Android Studio, оскільки вона надає найбільшу кількість можливостей та інструментів порівняно з іншими середами розробки, що позитивно позначиться на швидкості розробки та якості кінцевого продукту. Але інші IDE також можуть бути використаними

розробниками для створення застосунку при їхньому бажанні, оскільки у всіх них можливо розробити застосунок від початку до кінця.

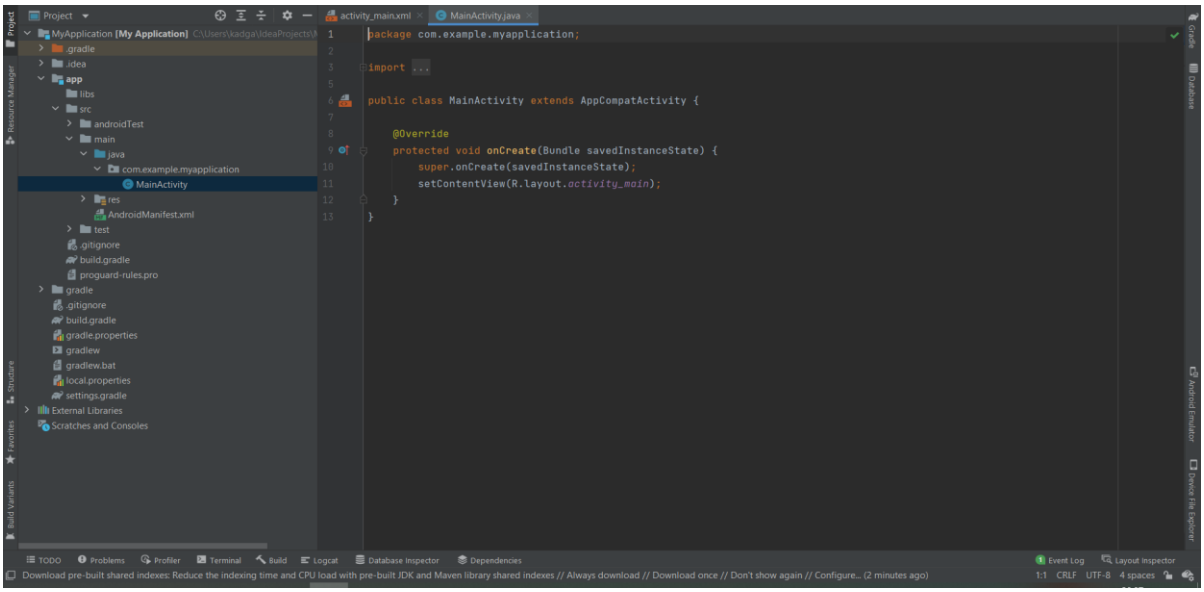

Рисунок 1.3 – Інтерфейс IntellijJ IDEA

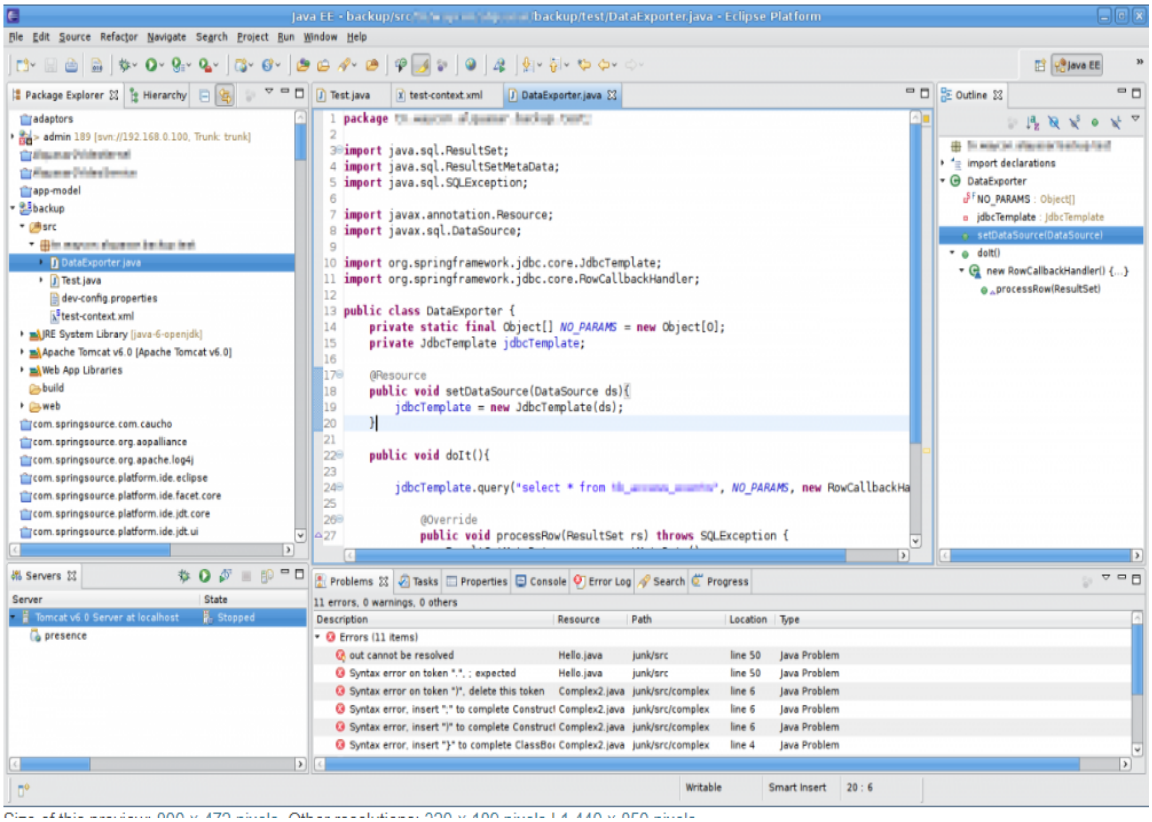

Рисунок 1.4 – Інтерфейс Eclipse

1.3.2 Архітектура Android застосунку

Базова архітектура Android застосунку використовує підхід «компонентної моделі», де кожен компонент виконує певну роль у застосунку і взаємодіє з іншими компонентами для досягнення певних цілей. Розглянемо детальніше кожен компонент:

Activity – відповідає за відображення користувачевого інтерфейсу (UI) на екрані пристрою. Кожна Activity має свій власний життєвий цикл і може бути запущена з іншої Activity або з системи, наприклад, коли користувач вибирає застосунок зі списку застосунків на своєму пристрої. Крім того, Activity може бути запущена з параметрами, що дозволяє передавати дані між Activity.

Service – відповідає за виконання довгострокових операцій, які не потребують взаємодії з користувачем. Service може бути запущений з Activity або з іншого Service. Service може працювати у фоновому режимі, навіть якщо користувач не взаємодіє з застосунком, і може виконувати такі операції, як відтворення музики, синхронізація даних з сервером або підрахунок даних.

Broadcast Receiver може виконувати певні дії відповідно до отриманих повідомлень, наприклад, відтворення звуку, відображення повідомлення або запуск іншого застосунку. Оскільки Broadcast Receiver може бути запущений в фоновому режимі, він може використовуватись для створення різноманітних фонових задач, таких як синхронізація даних, відслідковування подій, відтворення звуку та інше.

Content Provider надає доступ до даних застосунку іншим застосункам. Це дає можливість іншим застосунками отримувати, зберігати та оновлювати дані, які зберігаються у базі даних застосунку. Content Provider може бути використаний, наприклад, для забезпечення доступу до контактів, календарів, збережених файлів або зображень. Content Provider використовує систему URI для ідентифікації даних, які необхідно отримати або змінити.

Крім того, Content Provider може забезпечувати захист даних, використовуючи різні рівні доступу до них.

Кожен з компонентів Android застосунку має власну роль і взаємодіє з іншими компонентами. Наприклад, Activity може запускати інші Activity або Service, а Service може запускати Broadcast Receiver або Content Provider. Таким чином, використовуючи ці компоненти разом, можна створювати складні та багатофункціональні застосунки для платформи Android.

1.3.3 Вибір мови програмування

Основними мовами програмування для розробки Android застосунку є Java та Kotlin.

Java була основною мовою програмування для розробки Android застосунків до випуску Kotlin у 2017 році. Вона є однією з найпоширеніших мов програмування в світі та має багато ресурсів та інструментів для розробки. Java забезпечує велику кількість функцій та можливостей для розробки Android застосунків та досить проста у вивченні.

Kotlin, який був розроблений компанією JetBrains, є новішою мовою програмування, яка стає все більш популярною серед розробників Android застосунків. Kotlin є більш безпечною та зручною мовою, оскільки має більш строгу систему типів та уникне деяких типових помилок, які можуть виникнути під час розробки на Java. Kotlin також має дещо більш простий та зрозумілий синтаксис, що робить код більш зрозумілим та легшим для підтримки.

Від вибору мови програмування можуть залежати наступні складові:

– продуктивність: Вибір мови може вплинути на продуктивність розробки. Kotlin має менше написаного коду, що забезпечує більшу продуктивність розробки. Java може бути трохи повільнішою, оскільки потрібно більше коду для реалізації тих самих функцій;

– складність розробки: Kotlin має більш строгу систему типів та забезпечує більш безпечну розробку, що може сприяти зменшенню кількості помилок під час розробки. Java має менш строгу систему типів, тому вона може бути менш безпечною, але простішою в освоєнні;

– простота синтаксису: Kotlin має більш сучасний та зрозумілий синтаксис, що робить код більш зрозумілим та легшим для підтримки. Java має менш сучасний синтаксис, що може зробити код складнішим та менш зрозумілим;

– доступність інструментів та бібліотек: Обидві мови програмування мають доступ до багатьох інструментів та бібліотек, що дозволяє розробникам швидше розробляти та підтримувати свої застосунки. Однак, Kotlin, як новіша мова, може мати менше бібліотек та ресурсів, ніж Java.

1.3.4 Вибір бази даних

Не менш важливим етапом розробки будь-якого застосунку є вибір бази даних. Перед розробником є вибір між реляційними та нереляційними і локальними та серверними базами даних.

Основна різниця між реляційними та нереляційними базами даних полягає в тому, як дані зберігаються та як до них звертається.

Реляційні бази даних зберігають дані у вигляді таблиць, що містять рядки та стовпці. Кожен рядок відповідає окремому запису даних, а кожен стовпець представляє окреме поле даних. Реляційні бази даних забезпечують стандартні засоби для роботи з даними, такі як SQL, який дозволяє виконувати складні операції з даними, такі як запити, сортування та фільтрація.

Нереляційні бази даних зберігають дані у вигляді колекцій документів, таких як JSON, XML або BSON. Кожен документ представляє окремий запис

даних, а внутрішня структура документа може відрізнятися від документів в іншій колекції. Нереляційні бази даних забезпечують зручний та гнучкий спосіб зберігання даних, який може бути корисним для застосунків зі змінними схемами даних та великим обсягом неструктурованих даних.

Локальні бази даних – це бази даних, що зберігаються на пристрої користувача та можуть бути використані для зберігання та редагування даних без підключення до Інтернету. Такі бази даних можуть бути інтегровані безпосередньо в застосунок та забезпечувати швидкий доступ до даних. Зазвичай, ці бази даних використовуються для зберігання локальних налаштувань, кешування даних, або невеликих обсягів даних.

Серверні бази даних – це бази даних, що зберігаються на сервері та доступні через мережу Інтернет. Такі бази даних забезпечують доступ до даних з різних пристроїв та дозволяють забезпечувати доступ до даних між різними користувачами. Серверні бази даних можуть бути використані для зберігання великих обсягів даних, а також для забезпечення реального часу змін даних.

1.4 Постанова задачі

Таким чиним розробка мобільного застосунку є дуже актуальною та необхідною задачею. Тому ставиться завдання для реалізації Android застосунку для реалізації інтерактивних тестів з інформатики для налагодження та прискорення процесу дистанційного навчання.

Об’єктом роботи є застосунок до мобільних пристроїв.

Метою роботи є створення застосунка у якому можливо реалізувати створення та проходження різноманітних тестів з інформатики використовуючи мобільні пристрої при зберіганні автономії від мережі інтернет.

Для досягнення мети необхідно вирішити такі завдання:

- провести аналіз розробки мобільних застосунків;
- вивчити предметну область;
- виконати програмну реалізацію алгоритмів та провести тесування.

2 МАТЕМАТИЧНІ ОБГРУНТУВАННЯ МЕТОДІВ РЕАЛІЗАЦІЇ МОБІЛЬНОГО ЗАСТОСУНКУ

Розробка мобільного застосунку для інтерактивних тестів включає у себе велику кількість складових. Досить актуальною є задача розробки застосунку який може функціонувати без доступу до мережі інтернет. Сучасний досвід використання застосунків демонструє необхідність створення максимально автономного системи для найбільш зручного користування. Для реалізації автономності мобільного застосунку було вирішено використовувати локальну базу даних. Для поширення тестів було вирішено використовувати метод поширення тестів за допомогою технології QR кодів які використовують комп'ютерний зір сучасних пристроїв.

Також важливою складовою є те як саме тести будуть проходитися кожним користувачем. Оскільки у одному тесті сталий набір запитань, з часом це може призвести до викривлення статистики та погіршення об'єктивності оцінки. З цією метою до тестів додаються можливості випадкового відображення питань та варіантів відповіді, вибірка певної кількості питань з загального набору питань у тесті.

2.1 Вирішення питання однаковості тестів для користувачів

При однаковій послідовності запитань та варіантів відповідей збільшується похибка при оцінці відповідей. Це відбувається через те що де-факто кожний користувач проходить одну й ту саму версію тесту. Для збільшення кількості варіацій одного тесту прийнято використовувати методи комбінаторики. Нехай у тесті є 25 запитань і на кожне з них є лише 1 з 4 правильних варіантів відповіді. При відсутності комбінаторики є лише 1 можлива форма цього тесту [5].

Якщо використати перетасування лише запитань то можна отримати 1,551121004×10^25 варіантів послідовностей запитань. Це можна вирахувати за звичайною формулою факторіалу:

$$P(n) = n!, \tag{2.1}$$

де *P(n)* – це кількість можливих комбінацій;

n – це число питань у тесті.

Для реалізації такої можливості можна використати алгоритм Фішера-Йетса для перетасування елементів списку. Його перевага полягає у тому, що вірогідність генерації кожної комбінації є однаковою. Складність сучасної версії цього алгоритму становить *O(n)*. Сутність цього алгоритму можна представити наступним чином.

Нехай ми маємо масив *A* з n елементами, де *A[0], A[1], ..., A[n-1]* – елементи масиву.

Ініціалізація.

Встановимо $i = n - 1$ (поточний індекс, що починається з останнього елемента).

Поки $i > 0$, продовжуємо наступні кроки.

Генеруємо ціле випадкове число j в діапазоні від *0* до *i* (включно).

Змінюємо місцями елементи *A[i]* та *A[j]*.

Зменшуємо *i* на *1*.

Якщо додати до цієї перетасовки також перетасовку варіантів відповідей то отримаємо наступну формулу:

$$P(n, m) = n! * (m!)^n, \tag{2.2}$$

де *m* – це кількість варіантів відповідей у тесті.

Перестановку варіантів відповідей також можна реалізувати за допомогою алгоритму Фішера-Йетса [6]. Його складність збільшиться до

*O(n*m)*. Збільшення кількості комбінацій можна продемонструвати на прикладі тесту з кількістю відповідей на питання 4 (табл. 2.1, 2.2).

Таблиці 2.1 – Збільшення кількості варіантів тесту при перетасовці лише запитань

Кількість питань	Кількість можливих варіантів тесту
1	1
2	2
3	6
4	24
5	120
6	720
7	5040
8	40320
9	362880
10	3628800

Таблиця 2.2 – Збільшення кількості варіантів тесту при перетасовці запитань та варіантів відповідей

Кількість питань	Кількість можливих варіантів тесту
1	24
2	576
3	13824
4	331776
5	7962624
6	191102976
7	4586471424
8	1,10075E+11
9	2,64181E+12

Отже, застосувавши ці два види ускладнення тесту, можна гарантувати, що кожен тест буде унікальним з точки зору комбінацій порядку питань та відповідей на них.

2.2 Поширення даних за допомогою QR коду

Оскільки данні тесту містять інформацію у вигляді тексту та цифр постає задача з правильного генерування відповідного QR коду. Ця задача включає в себе кілька кроків, а саме кодування даних, додавання форматування та версії, генерацію образу QR коду та застосування маскування [7]. Розглянемо ці кроки докладніше:

– кодування даних: Перетворення вхідного тексту або цифр на послідовність бітів, яка буде представляти QR код. Для цього використовується відповідний метод кодування, такий як ASCII, який визначає способи представлення даних [8-10];

– додавання форматування та версії: Додавання інформації про форматування, версії QR-коду та інших метаданих до послідовності бітів даних. Ці дані визначають параметри QR-коду, такі як рівень корекції помилок та розмір модулів [11, 12];

– генерація образу QR коду: Створення матриці пікселів, що представляє зображення QR коду. У цьому кроці використовуються алгоритми розміщення блоків даних, кодових слів, опорних патернів та інших елементів QR коду всередині матриці [10, 13]. Кожен піксель у матриці матиме значення 0 або 1 залежно від наявності або відсутності модуля QR коду цієї позиції;

– застосування маскування: Застосування маскування до матриці QR коду для покращення стійкості до помилок та забезпечення рівномірного розподілу пікселів. У цьому кроці використовуються різні алгоритми

маскування, такі як алгоритми Бавека або Кавамура [14, 15], які вибирають певну маску та застосовують її до матриці QR коду;

– відображення QR коду: Перетворення матриці пікселів на зображення PNG, JPEG або іншого формату для відображення готового QR коду.

2.2.1 Кодування даних

Кодування початкових даних є дуже важливим і вимогливим процесом оскільки на основі цього кодування буде згенеровано кінцевий результат. Тому надважливо правильно перетворити дані на послідовність бітів і прослідкувати за тим, щоб усі дані потрапили до цієї послідовності.

Існує декілька найрозповсюдженіших варіантів кодування: ASCII та Unicode.

ASCII є одним із найпоширеніших методів кодування тексту. В ASCII кожному символу зіставляється числове значення від 0 до 127. Наприклад, символ A відповідає числу 65, символ B – 66 і так далі. Перетворення тексту в число в цьому випадку зводиться до перетворення кожного символу на його числове значення за таблицею ASCII і об’єднання цих числових значень в одне число або послідовність чисел.

Unicode є ширшим стандартом представлення символів різних писемностей. У Unicode кожному символу призначається унікальний кодовий пункт. Перетворення тексту на число у разі використання Unicode може бути виконане аналогічно перетворення ASCII, де кожному символу зіставляється його кодовий пункт в Unicode.

Варто зазначити, що з перетворення тексту на число необхідно враховувати використовуваний метод кодування і зворотне перетворення для відновлення і інтерпретації даних при декодуванні. Різні методи кодування

можуть використовуватись у різних ситуаціях залежно від вимог та контексту програми.

У випадку числових даних вони можуть бути представлені безпосередньо в числовій формі.

Далі відбувається бітове представлення кодованої у числа інформації. Це відбувається наступним чином:

– визначення довжини бітової послідовності: Залежно від типу даних та необхідної точності визначається кількість бітів, необхідних для представлення кожного символу або числа. Для ASCII символів може знадобитися 8 бітів (1 байт), а цілих чисел може знадобитися певна кількість бітів залежно від діапазону значень;

– перетворення символів або чисел на біти: Кожен символ або число перетворюється на послідовність бітів, використовуючи певне кодування. У випадку ASCII символів, кожен символ може бути перетворений на його числове значення, а потім це числове значення може бути представлене у вигляді послідовності бітів;

– об’єднання бітів: Для представлення послідовності символів або чисел біти об’єднуються в одну велику послідовність бітів. Це може бути виконано шляхом конкатенації бітових послідовностей разом.

2.2.2 Додавання форматування та версії

Цей етап полягає у розширенні згенерованої послідовності бітів додаванням метадати про QR код. До цієї метадати входять інформація про данні про корекцію помилок, масштабування та інші параметри необхідні для декодування інформації [13, 16, 17]. Рівень корекції помилок – це ступінь корекції помилок яку можливо присвоїти QR коду. Існує 4 ступеня цього рівня:

– L – низький рівень; забезпечує найменший рівень корекції помилок, але має найменшу ємність для даних. Він призначений для випадків, коли очікується невелика кількість помилок. Рівень L включає близько 7% кодових слів корекції помилок;

– M – середній рівень; забезпечує помірну ємність даних. Він підходить для більшості звичайних випадків використання QR-кодів. Рівень M включає близько 15% кодових слів корекції помилок;

– Q – помірний рівень; забезпечує високий ступінь корекції помилок і має більшу ємність для даних. Він рекомендований для випадків, коли очікується підвищена кількість помилок. Рівень Q включає близько 25% кодових слів виправлення помилок;

– H – високий рівень; забезпечує самий високий ступінь корекції помилок і має найбільшу ємність для даних. Він рекомендований для ситуацій, де важлива максимальна надійність і корекція помилок. Рівень H включає близько 30% кодових слів виправлення помилок.

Вибір рівня корекції помилок залежить від конкретних вимог та сценаріїв використання QR коду. Якщо дані в QR коді схильні до шуму, спотворень або поганої якості сканування, рекомендується вибрати більш високий рівень корекції помилок для забезпечення надійності та можливості відновлення даних.

Проте варто зазначити, що вищий рівень корекції помилок вимагає більшої кількості кодових слів корекції помилок, що збільшує розмір QR коду та знижує ємність для даних [11, 18] . Тому слід врахувати баланс між ступенем корекції помилок та ємністю даних залежно від конкретних потреб програми або використання QR коду.

Режим кодування визначає тип даних, які були закодовані: текст, числа та інше.

Масштабування визначає кінцевий розмір згенерованого коду та щільність елементів на ньому.

Версія – інформація про версію QR коду, визначає загальний розмір та ємність, включаючи кількість модулів та максимальну кількість даних, які можна закодувати. Версія QR коду позначається числом від 1 до 40, де більш висока версія означає більший розмір і більшу ємність. Вища версія може містити більше модулів і дозволяє закодувати більше даних.

2.2.3 Генерація образу QR коду

На цьому етапі генерується відповідне матричне відображення QR коду. Воно проходить за наступними етапами:

– визначення розміру QR коду: Спочатку визначається розмір QR коду у вигляді квадратної матриці, яка міститиме модулі QR коду (чорні та білі квадрати). Розмір матриці залежить від вибраної версії та рівня корекції помилок QR коду;

– розміщення опорних патернів: У верхньому лівому та правому кутах матриці розміщуються опорні патерни, які допомагають пристроям сканування QR коду визначити розмір та орієнтацію коду. Опорні патерни є квадратними ділянками з певними візерунками [19];

– розміщення даних і кодових слів: У частині матриці, що залишилася, розміщуються дані і кодові слова, що представляють закодовану інформацію QR коду. Кодові слова забезпечують корекцію помилок та допомагають відновити дані за наявності помилок сканування;

– відображення модулів QR коду: Залежно від даних та кодових слів, кожен модуль QR коду (квадратний елемент) у матриці буде встановлений як чорний або білий. Чорні модулі представляють біти даних або кодових слів, а білі модулі – простір між ними;

2.2.4 Застосування маскування

Застосування маскування в процесі генерації QR коду є зміною значень модулів відповідно до певних правил маскування. Мета маскування – покращити стійкість до помилок та забезпечити рівномірний розподіл пікселів у QR коді [20]. Основні кроки застосування маскування:

– вибір маски: Перший крок – вибір певної маски із заданого набору масок. Кожна маска є шаблоном, який визначає, які модулі QR коду будуть змінені, а які залишаться без змін;

– застосування маски до модулів QR коду: Після вибору маски вона застосовується до модулів QR коду. Відповідно до шаблону маски деякі модулі можуть змінити своє значення (чорний стане білим, а білий - чорним), а інші модулі можуть залишитися без змін;

– оцінка якості маскування: Після застосування маски оцінюється якість маскування. Для цього перевіряється, наскільки добре маска відповідає певним критеріям, таким як рівномірне розподілення чорних і білих модулів, мінімальна кількість патернів або послідовностей, які можуть викликати проблеми при скануванні та інші метрики;

– вибір оптимальної маски: З усіх застосованих масок вибирається та, яка найбільше відповідає заданим критеріям якості маскування. Оптимальна маска буде використана для фінального QR коду.

2.2.5 Відображення QR коду

На цьому етапі користувач отримує інтерпретацію даних у вигляді згенерованого QR коду.

2.3 Отримання даних за допомогою QR коду

Аналіз QR коду є дуже складною операцією яка використовує багато інструментів з сфери комп’ютерного зору та декодування інформації.

Процес починається з того, що камера користувача повинна визначити те, що перед нею знаходиться QR код. Цей процес складається з декількох етапів.

2.3.1 Фокусування на коді

За допомогою комп’ютерного зору, відбувається аналіз поточного зображення на камері застосунку. Відбуває пошук опорних пунктів, які є маркером того, що на зображенні присутній QR код. Для цього проводиться прибирання непотрібної інформації з зображення, яка точно не може містити необхідні елементи. Далі відбувається аналіз не фільтрованих ділянок зображення [21]. Для цього на кожному пристрої використовуються різні алгоритми, наприклад:

– алгоритм контрастного фокусування: цей алгоритм ґрунтується на пошуку області з найбільшим контрастом на зображенні. Він обчислює градієнти яскравості у різних областях зображення та шукає область з найбільшою різницею у яскравості [22]. Така область є центром фокусування;

– алгоритм фазового фокусування цей алгоритм використовує фазову інформацію вхідного світла для визначення фокусної відстані. Він аналізує зміни фази світлових хвиль, створюваних предметом, і знаходить точку, де різницю фаз найбільш виражена. Ця точка відповідає фокусній відстані;

– алгоритм зіставлення шаблонів: у цьому алгоритмі порівнюються шаблонні зразки з різними відстанями фокусними з поточним зображенням.

Обчислюється міра подібності між шаблонами та зображенням, і вибирається фокусна відстань, що дає найкращий збіг;

– алгоритми на основі фазової кореляції. Ці алгоритми використовують кореляцію між фазами сигналів для визначення фокусної відстані. Вони порівнюють фази сигналів при різних фокусних відстанях та знаходять пікові значення кореляції, які відповідають оптимальній фокусній відстані.

Приклад розміщення компонентів на готовому коді зображено на рисунку 2.1.

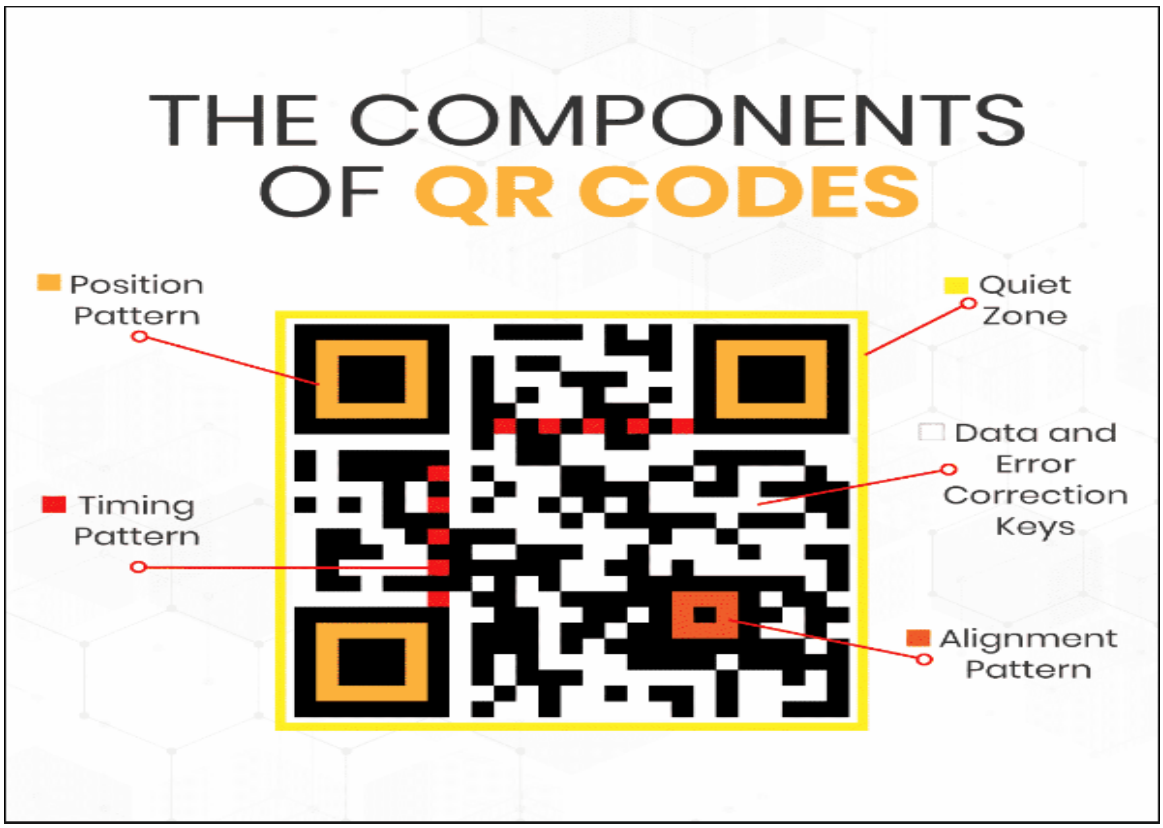

Рисунок 2.1 – Складові QR коду

2.3.2 Обробка зображення

Отримане зображення проходить процес декодування з метою отримання даних зашифрованих у коді.

Цей процес проходить через наступні етапи:

– підготовка даних: Закодовані дані QR коді представлені у вигляді бітового представлення. Спочатку ці біти групуються відповідно до певних правил і структури QR-коду, щоб отримати блоки даних, кодові слова корекції помилок та іншу необхідну інформацію;

– корегування помилок: Для обробки можливих помилок у скануванні QR коду застосовується процес корекції помилок. Кодові слова корекції помилок, включені до QR коду, використовуються для відновлення втрачених або пошкоджених бітів даних [23];

– інтерпретація структури: Інформація про структуру QR коду, таку як розмір, версія, рівень корекції помилок та інші параметри, інтерпретується визначення того, як дані були розділені і організовані в QR-коді;

– декодування даних: Декодування основних даних QR коду відбувається з використанням спеціальних алгоритмів та схем кодування. Ці алгоритми інтерпретують групи бітів, перетворюючи їх на відповідні символи, числа, текст або інші формати даних, залежно від вмісту QR-коду;

– перевірка контрольної суми: QR код містить контрольні суми, які використовуються для перевірки правильності декодування даних. Ці контрольні суми перевіряються, щоб переконатися, що дані правильно розшифровані;

– інтерпретація даних: Результатом декодування QR коду є інтерпретовані дані, такі як текст, URL адреса, номер телефону та інша інформація, закодована в QR коді. Інтерпретовані дані можуть бути представлені у зручній для читання формі або використані для подальших дій, залежно від мети використання QR коду.

3 КОМП’ЮТЕРНА МОДЕЛЬ МОБІЛЬНОГО ЗАСТОСУНКУ

3.1 Обґрунтування вибору середовища програмної реалізації

У рамках кваліфікаційної роботи був розроблений застосунок для створення та проходження інтерактивних тестів. Для реалізації було обране середовище Android Studio. Це обумовлено тим, що Android Studio створена та підтримується розробниками та власниками ОС Android, компанією Google. Для написання коду було обрано мову програмування Java. Базою даних слугує SQLite.

3.1.1 Обґрунтування вибору операційної системи

Операційна система Android є найрозповсюдженішою та найдоступнішою системою для мобільних пристроїв. Більшість користувачів користуються саме гаджетами на базі цієї операційної системи, а також саме девайси на Android є самими доступними на ринку. Для розробників ця ОС також має переваги такі як доступність середи розробки та її безкоштовність, а також величезну цільову аудиторію для кінцевого продукту, що є дуже привабливим для розробників початківців.

3.1.2 Обґрунтування вибору мови програмування

Мовою програмування для розробки застосунку була обрана Java. Основною мовою для написання Android застосунків є Kotlin, але Java також має свої переваги:

– об’єктно-орієнтованість: Java – повністю об’єктно-орієнтована мова програмування, що дозволяє розробляти Android-застосунки з

використанням концепцій спадкування, поліморфізму, інкапсуляції та абстракції;

– платформна незалежність: Java використовує віртуальну машину Java (JVM), що забезпечує платформну незалежність. Це означає, що один і той же код Java може виконуватися на різних платформах, включаючи Android;

– велика екосистема: Java має велику екосистему розробників, інструментів, бібліотек та фреймворків, які полегшують розробку Android-застосунків. Ви можете використовувати бібліотеки, такі як Retrofit, Gson, Dagger, та багато інших, щоб прискорити розробку та додати функціональність до вашого застосунку;

– підтримка багатопоточності: Java надає потужні засоби для роботи з багатопоточністю, що дозволяє ефективно використовувати ресурси пристрою та створювати чуйні Android-програми. Ви можете використовувати потоки (Threads) та синхронізацію для виконання паралельних завдань;

– безпека: Java має вбудовані механізми безпеки, які допомагають захистити програми від уразливостей. Наприклад, за допомогою модифікаторів доступу, винятків та системи контролю доступу (Java Security Manager) можна забезпечити безпеку програми;

– широкий вибір інструментів розробки: Java має безліч інструментів розробки, таких як Android Studio, Eclipse та IntelliJ IDEA, які полегшують створення, налагодження та тестування застосунків Android. Інтегровані середовища розробки (IDE) мають потужні функціональності та інструменти, такі як автодоповнення коду, відладчик і профільник;

– велика кількість ресурсів та документації: Java має величезну кількість ресурсів та документації, включаючи офіційні документи Oracle, підручники, відеоуроки та спільноти розробників. Це полегшує вивчення та отримання підтримки при розробці Android-застосунків Java;

– зворотна сумісність: Java забезпечує зворотну сумісність із попередніми версіями операційної системи Android. Це означає, що програми, розроблені на Java, працюватимуть на старіших версіях Android без необхідності внесення значних змін до коду;

– надійність та стабільність: Java є зрілою мовою програмування, яка тривалий час використовувалась для розробки Android-застосунків. Він відомий своєю надійністю та стабільністю, що сприяє створенню надійних та стабільних Android-застосунків;

– великий вибір розробників: Java є однією з найпопулярніших мов програмування, і багато розробників мають досвід роботи з ним. Це означає, що ви зможете знайти більше розробників, які готові приєднатися до вашої команди розробки, і отримати підтримку від досвідчених фахівців;

– налагодження та профільування: Java має розвинену інфраструктуру для налагодження та профілювання застосунків. Інструменти, такі як Android Studio та Java Profiler, дозволяють виявляти помилки, проводити аналіз продуктивності та оптимізувати роботу програм;

– хороша підтримка множинного успадкування: Java дозволяє реалізовувати множинне успадкування через інтерфейси. Це дозволяє розробникам створювати гнучкі та розширювані структури застосунків, реалізуючи різні інтерфейси із загальними та специфічними функціональностями;

– великий вибір сторонніх бібліотек: Java має величезну кількість сторонніх бібліотек, які розширюють функціональність та можливості Android-застосунків. Ви можете використовувати бібліотеки для роботи з графікою, мережевими запитами, базами даних та багатьма іншими, що дозволяє заощадити час та спростити розробку програм;

– широка підтримка архітектурних шаблонів: Java має широкий спектр підтримки різних архітектурних шаблонів, таких як MVC (Model-View-Controller), MVP (Model-View-Presenter) та MVVM (Model-View-

ViewModel). Це дозволяє розробникам вибирати підхід, який найкраще відповідає вимогам та структурі їх застосування;

– велика кількість прикладів та рішень: завдяки популярності Java у розробці Android-застосунків, існує безліч прикладів коду, рішень та посібників, які допоможуть розробникам у вирішенні типових завдань та знаходженні найефективніших способів реалізації певного функціоналу;

– інтеграція з іншими мовами програмування: Java дозволяє інтегрувати код, написаний іншими мовами програмування, до застосунків Android з використанням механізму Java Native Interface (JNI). Це дозволяє використовувати існуючий код або бібліотеки, написані на C/C++, для розширення функціональності програми;

– підтримка різних платформ: Java дозволяє розробляти не тільки Android-програми, але й програми для інших платформ, таких як настільні програми, серверні програми та веб-програми. Це означає, що розробники з досвідом роботи Java можуть використовувати свої навички для розробки застосунків для різних платформ;

– простота та зрозумілість коду: Java має простий та зрозумілий синтаксис, який полегшує читання та розуміння коду. Це робить розробку, підтримку та супровід Android-застосунків на Java більш зручними для розробників;

– великий досвід та легаси-код: завдяки довгій історії використання Java для розробки Android-застосунків, є великий досвід та величезна кількість існуючого легасу-коду, написаного на Java. Це дозволяє розробникам використовувати та скористатися існуючими рішеннями та кодом, що може прискорити процес розробки та знизити складність проєкту.

3.1.3 Обґрунтування вибору середовища розробки

Середовищем розробки було обрано Android Studio через наступні властивості:

– інтегроване середовище розробки: Android Studio – це спеціально розроблене інтегроване середовище розробки, створене для розробки програм під платформу Android. Вона пропонує безліч інструментів та функцій, які полегшують створення, налагодження та тестування застосунків;

– потужний редактор коду: Android Studio має потужний редактор коду, який надає підсвічування синтаксису, автодоповнення, швидке виправлення помилок, рефакторинг коду та інші корисні функції. Він забезпечує зручне та ефективне написання коду;

– розширена підтримка для Android: Android Studio надає безліч інструментів і функцій спеціально розроблених для розробки Android-застосунків. Він забезпечує інтеграцію з Android SDK, дозволяє створювати різні типи проєктів, і пропонує безліч візуальних редакторів для створення інтерфейсу користувача і макетів;

– емулятори та пристрої для тестування: Android Studio пропонує вбудовані емулятори та можливість підключення реальних пристроїв для тестування програм. Розробник може запускати та налагоджувати свої програми на різних версіях Android та різних розмірах екранів, щоб переконатися в їхній правильній роботі;

– профілювання та оптимізація: Android Studio надає інструменти для профілювання програм, що дозволяє аналізувати продуктивність та оптимізувати використання ресурсів. Користувач може відстежувати використання пам’яті, процесора, мережі та інших аспектів програми для підвищення його ефективності;

– зручна система складання та управління залежностями: Android Studio використовує систему складання Gradle, яка забезпечує гнучкість та

зручність при керуванні залежностями, компіляції та складання проєкту. Програміст може легко додавати сторонні бібліотеки та контролювати залежність проєкту;

– інтеграція із системою контролю версій: Android Studio вбудовано підтримує найпопулярніші системи контролю версій, такі як Git. Користувач може легко керувати своїм проєктом, комітити зміни, розгалужувати та зливати код, а також працювати з віддаленими репозиторіями, використовуючи графічний інтерфейс або командний рядок;

– автоматичне створення коду та генерація ресурсів: Android Studio надає безліч функцій автогенерації коду, які спрощують процес розробки. Це включає автозаповнення коду, створення геттерів та сеттерів, генерацію методів зворотного виклику, створення тестових класів та багато іншого. Також є можливість генерації ресурсів, таких як макети екранів, іконки та рядки перекладу;

– багатий набір інструментів для розробки інтерфейсу користувача: Android Studio пропонує безліч інструментів для розробки інтерфейсу користувача. Візуальний редактор макетів дозволяє створювати інтерактивні та чуйні макети екранів, а також легко працювати з різними віджетами та компонентами. Також можливо переглядати та редагувати ресурси програми, такі як зображення та кольори, за допомогою графічного інтерфейсу;

– зручний налагоджувач: Android Studio надає потужний налагоджувач, який дозволяє виконувати покрокове виконання коду, встановлювати точки зупинки, спостерігати значення змінних та аналізувати стек викликів. Він також надає можливість відстежувати події та помилки, що виникають під час виконання програми;

– оновлення та підтримка від Google: Android Studio постійно оновлюється та покращується Google, щоб бути в курсі останніх змін та вимог платформи Android. Це включає нові функції, виправлення помилок,

поліпшену інтеграцію з Android-платформою і підтримку останніх версій Android API.

3.1.4 Обґрунтування вибору системи контролю версій

В якості системи контролю версії було обрано Git. Git має наступні переваги:

– децентралізованість: Git є розподіленою системою контролю версій, що означає, кожен учасник проєкту має повну копію репозиторію. Це забезпечує незалежність та можливість працювати офлайн;

– історія змін: Git зберігає повну історію змін проєкту. Кожна зміна, коміт або гілка відстежується, що дозволяє повернутися до попередніх версій коду або відновити видалені файли;

– розгалуження та злиття: Git забезпечує потужні можливості розгалуження та злиття коду. Користувач може створювати окремі гілки для розробки нових функцій або виправлення помилок, а потім безпечно об’єднувати їх у основну гілку;

– прискорена швидкість роботи: Git має високу продуктивність і швидкість роботи навіть з великими репозиторіями та історіями змін. Це робить процес комміту, злиття та перемикання гілок швидким та ефективним;

– легкість: Git використовує ефективні алгоритми стиснення, що робить його репозиторії відносно невеликими за розміром. Це дозволяє заощадити місце на диску та прискорює передачу даних під час роботи з віддаленими репозиторіями;

– можливість відкату змін: Git дозволяє відкотитися до попередніх версій коду або скасувати небажані зміни. Це дозволяє легко виправляти помилки або відновлювати попередній стан проєкту;

– розгалуження на вимогу: Git дозволяє створювати гілки не тільки для розробки нових функцій, але й для тимчасової роботи над певними завданнями чи виправленнями. Це допомагає організувати та структурувати процес розробки;

– зручне злиття та вирішення конфліктів: Git забезпечує потужні інструменти для злиття змін із різних гілок та вирішення можливих конфліктів при злитті. Це дозволяє об’єднувати код розробників та вирішувати можливі протиріччя;

– гілки-покажчики (Branches): Гілки в Git – це легковажні покажчики на певний коміт в історії проєкту. Вони дозволяють розробляти різні фічі та виправлення паралельно, не торкаючись основної галузі розробки. Кожна гілка може мати власний стан проєкту, що полегшує роботу в команді та керування різними завданнями;

– можливість віддаленої роботи: Git підтримує віддалену взаємодію, дозволяючи розробникам працювати з віддаленими репозиторіями. Це дозволяє команді розробників спільно працювати над проєктом, ділитися змінами та синхронізувати свою роботу;

– гілка «master» та гілки стабільної версії: Git за замовчуванням створює гілку «master», яка представляє стабільну версію проєкту. Ви можете створювати та керувати іншими гілками для розробки нових функцій, виправлення помилок або проведення експериментів, і зрештою об’єднувати їх з основною гілкою «master»;

– зберігання метаданих: Git зберігає метадані про кожен коментар, включаючи автора, дату та час, коментарі та зміни файлів. Це полегшує відстеження та аналіз історії розробки проєкту;

– підтримка розгалуження та злиття за бажанням: Git дозволяє розгалужувати та зливати код у міру необхідності. Ви можете створювати гілки для різних функціональностей або завдань, а потім об’єднувати їх з

основною гілкою, коли вони готові. Це сприяє організації роботи та забезпечує гнучкість при розробці;

– порівняння змін: Git надає можливість порівняти зміни між різними версіями файлу або коммітами. Це корисно для аналізу та перевірки внесених змін та допомагає виявити помилки чи конфлікти;

– відновлення видалених файлів: Git дозволяє відновити видалені файли або навіть цілі директорії з попередніх комітів. Це допомагає уникнути втрат даних і відновити видалені або випадково видалені файли.

3.1.5 Обґрунтування вибору бази даних

Базою даних було обрано SQLite. Ця база даних має наступні переваги:

– легкість впровадження: SQLite є базою даних, що вбудовується, що означає, що вона може бути легко інтегрована в застосунок без необхідності встановлення окремого сервера або налаштування складної конфігурації. Це полегшує розгортання програми на різних пристроях;

– крос-платформність: SQLite підтримується на багатьох платформах, включаючи Android, iOS, Windows, macOS та інші. Це дозволяє розробникам створювати програми, які можуть працювати на різних операційних системах, використовуючи загальну базу даних;

– ефективність та швидкодія: SQLite має високу продуктивність та ефективність роботи. Вона пропонує компактний розмір бази даних, швидкі операції читання та запису, а також підтримку індексів для оптимізації пошуку та сортування даних;

– надійність та цілісність даних: SQLite забезпечує надійність та цілісність даних. Вона підтримує транзакції, що дозволяє виконувати групу операцій як єдине ціле, забезпечуючи узгодженість даних та запобігаючи їх ушкодженню у разі збою чи помилки;

– широкий набір функцій: SQLite надає безліч функцій для роботи з даними, як от SQL запити, фільтрація, сортування, агрегація та інші операції. Вона підтримує різні типи даних, включаючи числа, рядки, дати та бінарні дані, а також пропонує можливості для створення складних запитів та зв’язків між таблицями;

– підтримка на рівні операційної системи: SQLite інтегрована з операційною системою та може використовувати її механізми кешування та керування пам’яттю. Це дозволяє оптимізувати використання ресурсів пристрою та забезпечити більш ефективну роботу з даними;

– гнучкість та масштабованість: SQLite підтримує роботу з великими обсягами даних та дозволяє масштабувати базу даних залежно від потреб програми. Вона також підтримує резервне копіювання та відновлення даних, що забезпечує захист та збереження інформації;

– підтримка Android-платформи: SQLite є стандартною базою даних для розробки програм під платформу Android. Вона вбудована в Android-фреймворк і надає зручні API для роботи з базою даних програми. Android SDK надає безліч класів та методів для створення, оновлення, запиту та керування даними в SQLite;

– економія ресурсів: SQLite працює на пристроях з обмеженими ресурсами, такими як мобільні пристрої. Вона споживає мінімум оперативної пам’яті та процесорних ресурсів, що дозволяє оптимізувати продуктивність програми та продовжити термін роботи пристрою від батареї;

– легкість роботи з офлайн-даними: SQLite ідеально підходить для роботи з даними в режимі офлайн. Програми можуть зберігати та обробляти дані безпосередньо на пристрої без постійного підключення до мережі. Це особливо важливо для програм, які вимагають доступу до даних в автономному режимі або мають обмежений зв’язок з Інтернетом;

– розширюваність: SQLite пропонує можливість розширення функціональності за допомогою розширень користувача. Ви можете

створювати та підключати власні розширення, написані мовою програмування C/C++, для реалізації специфічної логіки або додавання додаткових функцій до бази даних;

– широке співтовариство та підтримка: SQLite має широке співтовариство розробників та активну підтримку з боку команди розробників. Існує безліч документації, посібників, форумів та онлайн-ресурсів, які допоможуть вам розібратися в роботі з SQLite і вирішити питання та проблеми, що виникають;

Інтерфейс інтеграції Git зображено на рисунку 3.1.

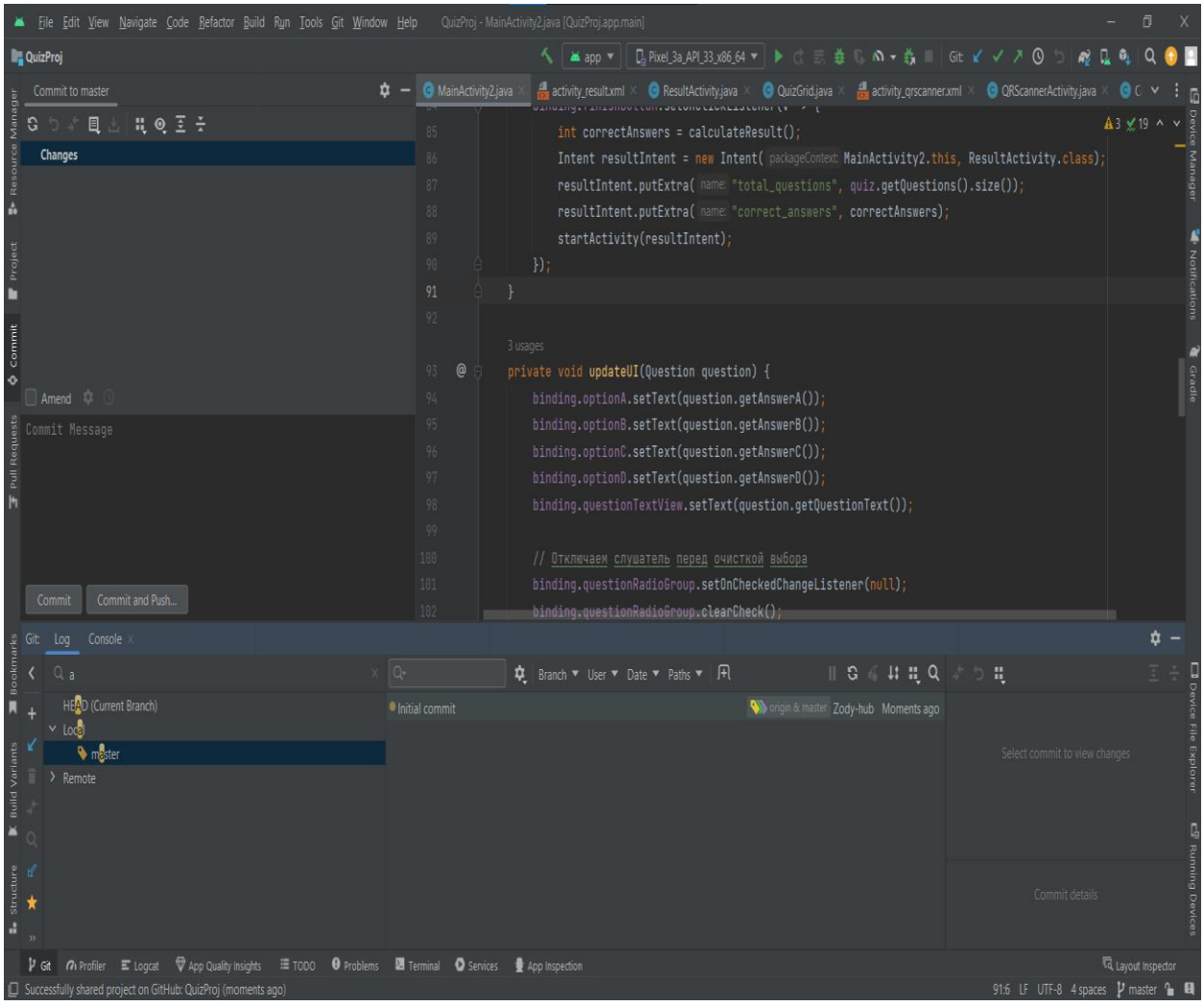

Рисунок 3.1 – Інтерфейс Git у Android Studio

3.2 Програмна реалізація

Для реалізації мобільного застосунку було створено початковий інтерфейс який дозволяє користувачу обрати базові функції застосунку для інтерактивних тестів, а саме: «Take a quiz» та «Create test». Цей інтерфейс зображений на рисунку 3.2. Цій кнопці містять у собі listener на натискання, у якому створюється об’єкт класу Intent який відкриває користувачу нову активність.

Intent – це об’єкт, який представляє намір виконати певну дію в Android-застосунку. Він служить для зв’язку різних компонентів програми, таких як активності (Activity), сервіси (Service) та приймачі (BroadcastReceiver), а також для взаємодії з компонентами інших програм. Intent може бути явним (explicit) чи неявним (implicit). Явний Intent вказує на конкретний компонент програми, з якою необхідно взаємодіяти. Наприклад, він може вказувати на певну активність у програмі, яку потрібно відкрити. Неявний Intent використовується для виконання дій, які можуть бути оброблені кількома компонентами. Наприклад, відкриття вебсторінки в браузері, надсилання електронної пошти або вибір контакту зі списку.

Рисунок 3.2 – Початковий екран користувача

Intent може містити додаткові дані, що передаються між компонентами програми. Вони можуть бути використані для передачі інформації, параметрів чи результатів операцій.

Activity – це компонент програми, який є одним екраном з інтерфейсом користувача і взаємодіє з користувачем. Кожне Activity виконує певну функцію в програмі, наприклад, відображення списку, редагування даних або виконання певної дії.

Ключові особливості Activity включають:

– життєвий цикл: Activity має свій життєвий цикл, який складається з різних станів, таких як створення, запуск, призупинення, відновлення та знищення. Програма може реагувати на ці стани та виконувати відповідні операції, наприклад, збереження стану або звільнення ресурсів;

– керування інтерфейсом користувача: Activity відповідає за відображення інтерфейсу користувача та обробку користувацьких подій, таких як натискання кнопок або введення тексту. Вона може містити різні віджети та елементи керування для взаємодії з користувачем;

– навігація між активностями: у програмі можна переходити з однієї Activity в іншу за допомогою Intent. Наприклад, після натискання кнопки в одній активності можна запустити іншу Activity за допомогою явного або неявного Intent, що дозволяє створювати плавний інтерфейс користувача і перемикатися між різними екранами програми;

– контекст та ресурси: Activity надає контекст виконання, який дозволяє отримувати доступ до різних ресурсів програми, таких як рядки перекладу, зображення та розміри екрана. Контекст також використовується для запуску інших компонентів програми та виконання різних операцій;

– взаємодія з іншими компонентами: Activity може взаємодіяти з іншими компонентами програми, такими як послуги, приймачі та фрагменти. Вона може надсилати та приймати дані, отримувати сповіщення та реагувати на події, що відбуваються в інших компонентах;

– жести та мультитач: Activity підтримує обробку різних жестів та мультитача, що дозволяє створювати інтерактивні та багатопотокові інтерфейси користувача. Наприклад, активність може реагувати на свайпи, торкання та переміщення пальців по екрану;

– результати та повернення даних: Activity може повертати результати іншим компонентам програми, наприклад, після вибору елемента зі списку або завершення певної дії. Це дозволяє оновлювати дані та стан програми на основі отриманих результатів;

– зміни конфігурації: Activity може автоматично обробляти зміни конфігурації, такі як поворот екрана або зміна мови. Вона може зберігати стан даних і оновлювати інтерфейс користувача, щоб адаптуватися до нових налаштувань пристрою;

– взаємодія з системою: Activity може виконувати різні дії взаємодії з системою, такі як дзвінки, надсилання повідомлень, відкриття вебсторінок та доступ до різних системних служб. Це дозволяє застосуванню взаємодіяти з іншими програмами та використовувати функціональні можливості пристрою.

Відкривається нова Activity на якій відображені усі тести які є у базі даних користувача та надає можливість з ними користуватися. Приклад інтерфейсу зображено на рисунку 3.3. На цій Activity створено Grid з бази даних, на кожен елемент якого встановлено listner та кнопку для можливості видалення, проходження та поширення даного елементу.

Grid – це елемент інтерфейсу який дозволяє користувачу бачити елементи з бази даних та взаємодіяти з ними.

Можливість видалення була реалізована за допомогою діалогу. DialogFragment – це спеціальний фрагмент Android, який використовується для відображення діалогових вікон (dialog) в застосунку. Він є частиною фреймворку Android і надає зручний спосіб створення та управління

діалоговими вікнами, включаючи стандартні діалоги та користувальницькі діалоги.

DialogFragment – надає наступні можливості розробнику:

– управління життєвим циклом: DialogFragment успадкований від класу Fragment і, отже, має власний життєвий цикл, подібно до інших фрагментів у застосунку. Він дозволяє створювати, відображати, зберігати стан та знищувати діалогові вікна з урахуванням стану активності, до якої він прив’язаний;

– підтримка різних типів діалогів: DialogFragment підтримує різні типи діалогових вікон, включаючи AlertDialog, DatePickerDialog, TimePickerDialog та інші стандартні діалоги Android. Крім того, розробник може створювати власні власні діалоги, визначені всередині DialogFragment;

– гнучкість налаштувань та параметрів: DialogFragment пропонує безліч методів та можливостей для налаштування зовнішнього вигляду та поведінки діалогового вікна. Розробник може встановлювати заголовок, текст, кнопки, вибирати стилі, обробляти події та застосовувати інші параметри, щоб діалог відповідав вашим вимогам та дизайну програми;

– реагування на дії користувача DialogFragment дозволяє реагувати на дії користувача всередині діалогу. Розробник може визначити обробники подій, наприклад, обробку натискань кнопок, вибір дати або часу в діалозі DatePickerDialog або TimePickerDialog;

– гнучкість розташування: DialogFragment може відображатися в різних режимах розташування, включаючи повноекранний режим, режим спливаючого вікна, а також використання як вбудований фрагмент в інших макетах або активності;

– збереження стану: DialogFragment автоматично зберігає свій стан при зміні конфігурації пристрою, наприклад, при повороті екрана. Це дозволяє зберегти поточний стан діалогу та відновити його після зміни конфігурації, щоб користувач не втратив введені дані або прогрес;

– гнучкість використання у фрагменті: DialogFragment може бути використаний всередині інших фрагментів у застосунку, що дозволяє створювати складні макети з кількома діалоговими вікнами або взаємодіяти з іншими компонентами інтерфейсу;

– керування фокусом та введенням: DialogFragment надає зручні методи для керування фокусом та введенням у діалоговому вікні. Розробник може встановити фокус на певний елемент інтерфейсу, визначити правила введення або обмеження, а також обробляти події клавіатури та сенсорного введення;

– сумісність із зворотним викликом: DialogFragment підтримує використання інтерфейсу зворотного виклику, який дозволяє взаємодіяти між DialogFragment та активністю чи іншими фрагментами. Це полегшує передачу даних та виконання дій між компонентами програми;

– підтримка адаптивного дизайну: DialogFragment дозволяє створювати діалогові вікна, які адаптуються до різних розмірів екрана та орієнтацій пристрою. Розробник може налаштувати різні варіанти макета для портретної та альбомної орієнтації, щоб діалоги виглядали та функціонували оптимально на різних пристроях.

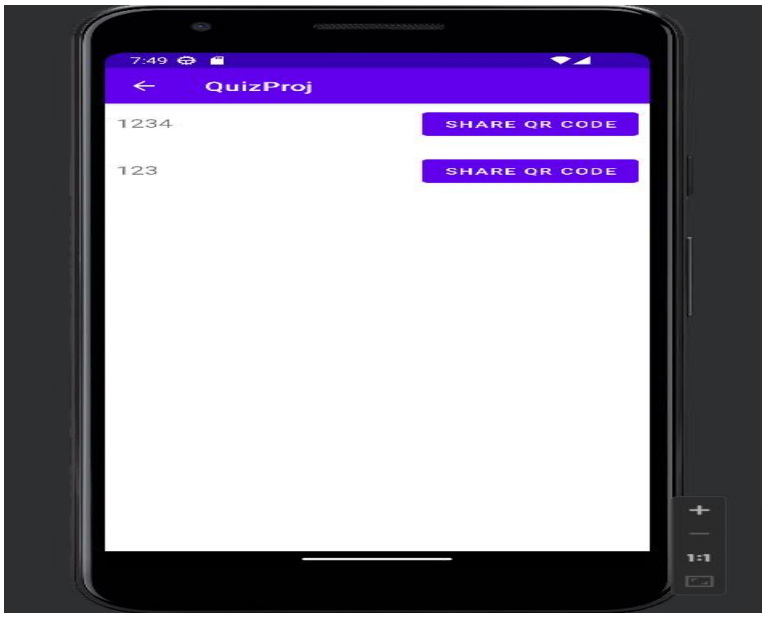

Рисунок 3.3 – Інтрефейс Activity з Grid бази данних

Генерація QR коду реалізована за допомогою перетворення тесту у JSON об’єкт за допомогою бібліотеки Gson від Google. Далі створюється візуальне зображення для QR коду. Цей процесс реалізовано за допомогою бібліотеки Zxing від Google. Ця бібліотека надає розробнику наступний набір можливостей:

– сканування штрих-кодів: ZXing забезпечує можливість сканування різних типів штрих-кодів і QR кодів за допомогою камери пристрою. Вона надає готові компоненти інтерфейсу користувача для запуску сканера та обробки результатів сканування;

– генерація штрих-кодів: ZXing також дозволяє генерувати штрих-коди і QR коди на основі переданих даних. Ви можете створювати штрих-коди різних форматів та налаштовувати їх параметри, такі як вміст, розмір, колір та додаткові опції;

– підтримка різних типів штрих-кодів: ZXing підтримує велику кількість форматів штрих-кодів, включаючи UPC, EAN, Code 39, Code 128, Data Matrix, PDF417 та багато інших [23, 24]. Це дозволяє працювати з різними типами даних і зчитувати/генерувати відповідні штрих-коди;

– гнучкість налаштувань та параметрів: бібліотека ZXing надає широкий набір налаштувань та параметрів, які дозволяють визначити зовнішній вигляд, поведінку та додаткові функції сканера або генератора [25, 26]. Розробник може налаштувати фактори, такі як режим сканування, фокус, підсвічування, звукові сигнали та інші опції відповідно до вимог вашої програми;

– легкість інтеграції: ZXing надає готові компоненти інтерфейсу та API для інтеграції у ваші програми [27 - 29]. Розробник може легко вбудувати сканер або генератор штрих-кодів у свою програму і налаштувати їх відповідно до дизайну та потребами вашої програми;

– крос-платформність: ZXing доступна не тільки для платформ Android, але і для інших платформ, включаючи iOS, Windows, Linux та інші. Це дозволяє використовувати її в різних проєктах і на різних пристроях;

– широке використання: Бібліотека ZXing широко використовується в різних застосунках та галузях, включаючи електронну комерцію, транспорт та логістику, складське управління, маркетинг, лікарські програми та інші. Це доводить надійність та ефективність бібліотеки в роботі зі штрих-кодами та QR кодами;

– підтримка додаткових функцій: разом з основною функціональністю сканування та генерації штрих-кодів, ZXing також пропонує додаткові функції та можливості [30]. Наприклад, вона може розпізнавати текст зображень, декодувати вебпосилання та контактну інформацію з QR кодів, а також підтримувати багатопотокову обробку для оптимізації продуктивності.

Приклад згенерованого QR коду зображено на рисунку 3.4.

При натисканні на назву тесту відкривається вікно проходження тесту. Процес проходження реалізовано наступним чином:

– на нову активність за допомогою Intent передаються дані про обраний користувачем тест;

– список питань перемішується;

– на активність передається 1 з перемішаного списку питання;

– варіанти відповіді також перемішуються;

– на активність виводиться Text field за питанням, Radio group з варіантами відповіді та кнопки для переходу на минуле або наступе питання і завершення тесту;

– на активність виводиться Text field за питанням, Radio group з варіантами відповіді та кнопки для переходу на минуле або наступе питання і завершення тесту;

– при натисканні кнопки завершення створюється новий Intent з даними про обрані варіанти користувачем і виводиться його результат.

Рисунок 3.4 – Згенерований QR код для поширення тесту

Функціонал створення тесту було реалізовано наступним чином:

– при натисканні на кнопку створення створюється нова активність;

– вона складається з тестового поля та 2 кнопок Create Question та Save;

– при натисканні кнопки Create Question створюється нова форма на поточній активності яка складається з 5 текстових полів та списку у якому підвантажуються варіанти відповідей введені користувачем.

3.3 Інструкція користувача

При запуску застосунку користувачу буде відображений простий інтерфейс який зображений на рисунку 3.2.

Далі у користувача є дві опції вибору: створення тесту або його проходження. User experience користувача на цьому етапі зображено на рисунку 3.5.

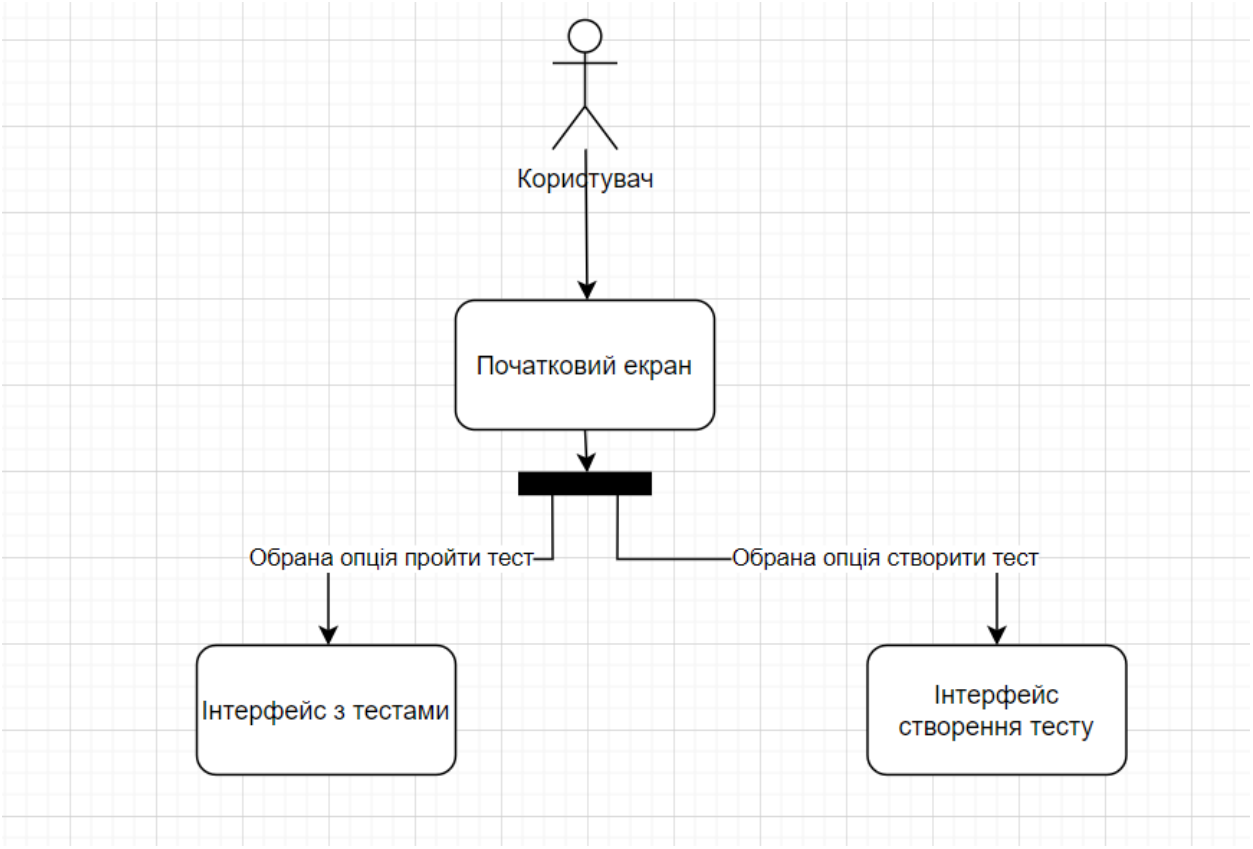

Рисунок 3.5 – UML діаграма початкових можливостей користувача

При виборі варіанта пройти тест на інтерфейсі користувача з’являються тести які знаходятся у базі даних користувача який зображений на рисунку 3.3. При виборі опції пошарити QR код на інтерфесі користувача буде згенерований QR код. Приклад цього зображено на рисунку 3.4. При довгому натисканні на назву тесту відкриеться повідомлення за допомогою якого можна видалити тест за бази даних. Це зображено на рисунку 3.6. При швидкому натисканні на назву тесту відкриеться його проходження. Це зображено на рисунку 3.7. При натисканні на стрілку згори користувача поверне на початковий екран.

При натисканні кнопки створення тесту відкриеться меню зображене на рисунку 3.8. Для того щоб користувач міг додати питання до тесту йому необхідно натиснути на кнопку додати питання, після чого інтерфейс доповниться вікном створення питання, яке можна буде припрбати натиснувши на крестик. По завершенню заповнення необхідно натиснути на кнопку зберегти. Розшерений інтерфейс зображено на рисунку 3.9. Можливості користувача на цих єтапах зображено на рисунку 3.10.

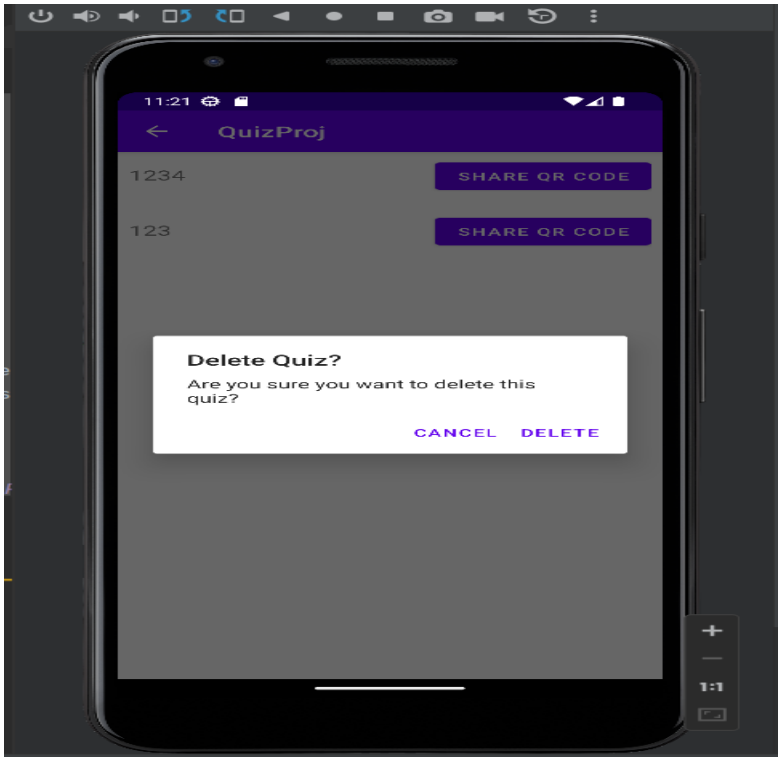

Рисунок 3.6 – Вікно видалення тесту

Рисунок 3.7 – Інтерфейс проходження тесту

Під час проходження тесту користувач може обирати варіанти відповідей або залишати відповідь порожньою. При переході на наступне питання починає відображатися кнопка попереднє питання. При поверненні на минулі питання обрана користувачем відповідь залишається. При натисканні на кнопку завершення на екран користувача виводиться його результат.

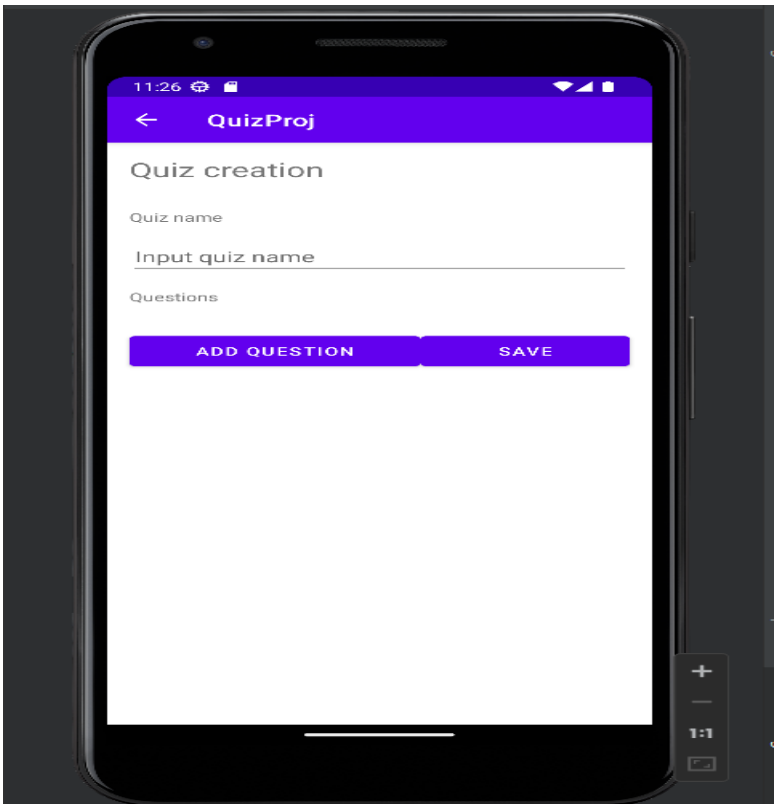

Рисунок 3.8 – Інтерфейс створення тесту

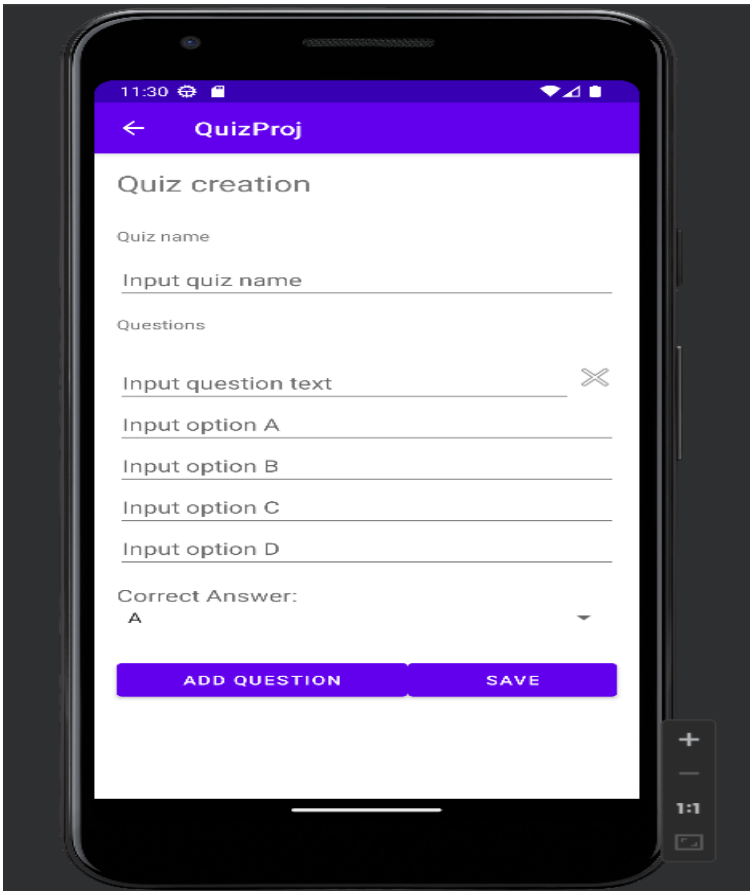

Рисунок 3.9 – Розширений інтерфейс створення тесту

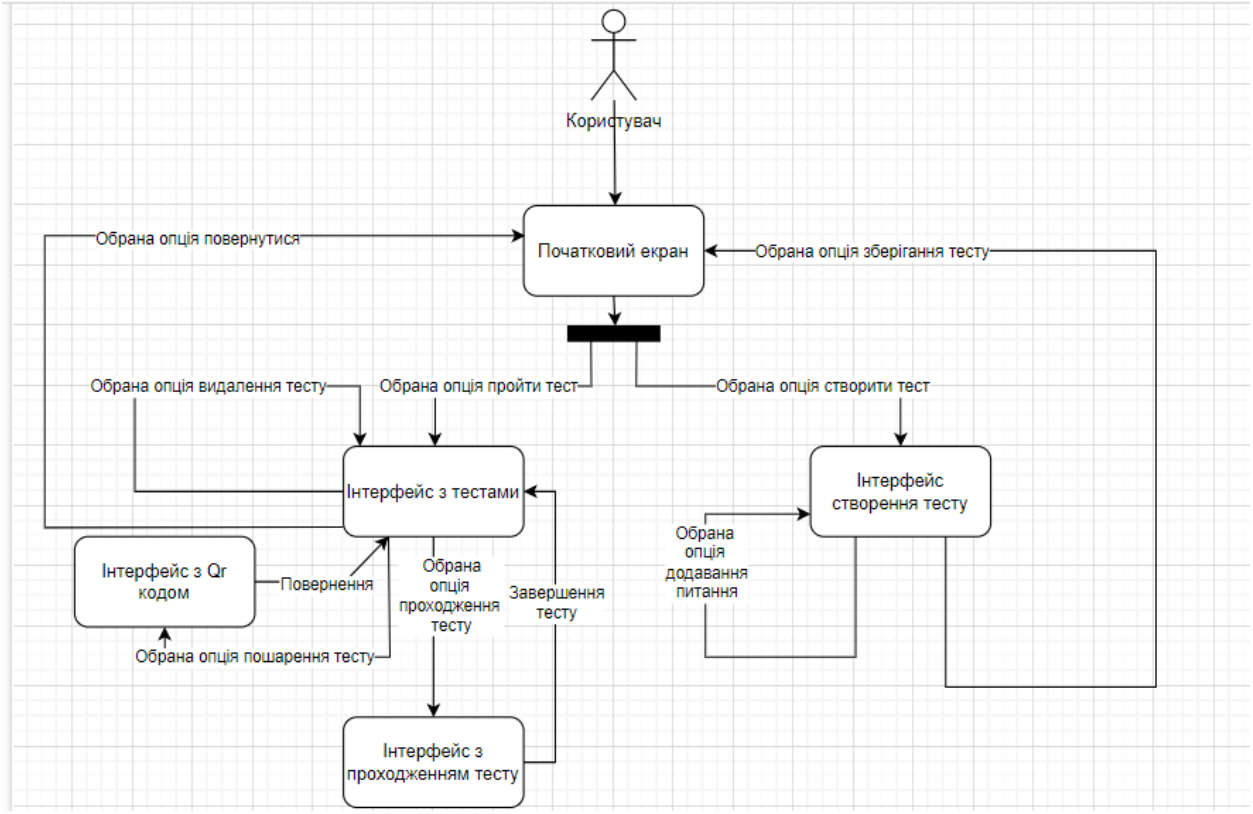

Рисунок 3.10 – UML діаграма головних можливостей користувача

3.4 Тестнування і проблеми під час розробки

В процесі тестування були виявлені наступні можливі проблеми для користування.

– при багаторазовому натисканні на кнопку генерації QR коду створювалося декілька Activity и тому користувач повинен був закривати усі створені їм Activity;

– при створенні користувачем тесту та додаванні великою кількості питань неможливо було отримати доступ до всіх елементів;

– при створенні тесту та прибиранні питання неможливо було зберегти тест;

– при проходженні тесту та переходу на наступне питання відповідь була одразу обрана.

Далі були проведені дії по виявленню причин цих помилок.

Помилка пов’язана з генераціє багатьох Activity була викликана тим, що за замовчуванням Activity створюються з можливістю існування кількох екземплярів цієї Activity у застосунку. Для того, щоб виправити це необхідно при створенні Intent додати прапорець FLAG_ACTIVITY_SINGLE_TOP, який вказує Activity, що одночасно може існувати лише один її екземпляр.

Друга помилка була пов’язана з тим що за замовчуванням Activity у Android не мають можливості перемотки вгору та вниз. Для того щоб додати таку можливість необхідно додати Scroll View до макету своєї активності.

Наступна помилка пов’язана з тим, що створене а потім видалене запитання знаходилося у списку запитань у тесті і тому для того зоб виправити цю помилку при видалені користувачем питання воно також видалялось з списку питань тесту.

Помилка пов’язана з тим що при переході на нове запитання у користувача була обрана відповідь пов’язана з тим, що при переході на наступне запитання не генерувалася нова Activity, а лише оновлювався текст, тому обраний варіант відповіді залишався на екрані. Це було вирішено за допомогою того, що при натисканні на кнопку очищувались варіанти відповідей з екрану та зберігалися у даних програми, для того що б при повернені на пройдене запитання користувачу не доводилося знову обирати варіант відповіді.

Також важливо додати Use-Case діаграми для кожного процесу окремо, для демонстрації функціоналу. На рисунку 3.11 зображений процес проходження тесту з точки зору користувача.

Цей сценарій проходить наступним чином: користувач одразу ж може обрати 2 опції поведінки: обрати відповідь на запитання чи завершити тест. При виборі варіанту завершення тестування підраховується кількість обраних коректно відповідей. При виборі варіанта відповіді до кешу програми заносить вибір користувача, який поки не позначається ані як коректний, ані як некоректний.

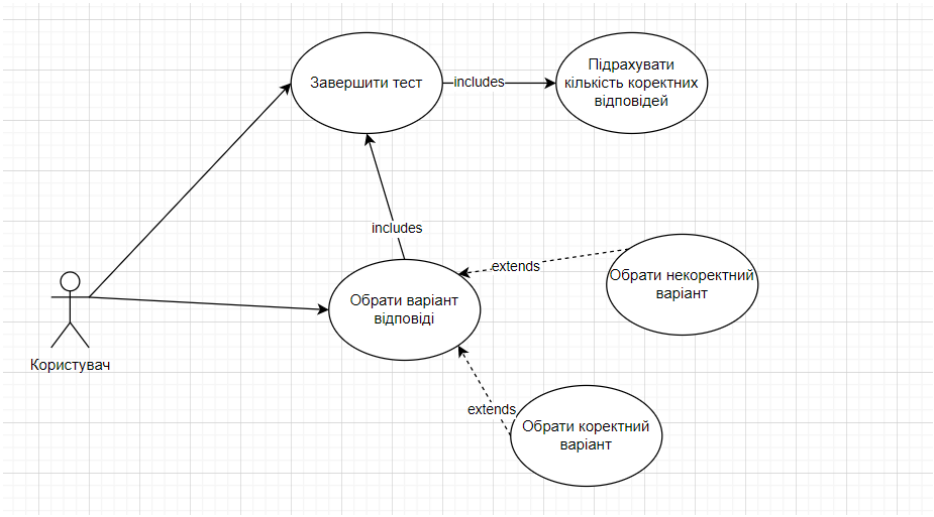

Рисунок 3.11 – Use-case діаграмма проходження тесту

Наступний сценарій це отримання нових тестів. Він відбувається наступним чином: користувач бачить перед собою меню створення тесту та кнопку відсканувати. Якщо користувач власноруч бажає створити тест то він повинен заповнити назву тесту, додати запитання та відповіді на них, обрати до кожного запитання коректну відповідь з списку тих, що були введені до цього питання. При виборі варіанту сканування користувач повинен навести камеру пристрою на згенерований іншим користувач QR код для того щоб отримати на свій пристрій цей тест. Цей процес зображено на рисунку 3.12.

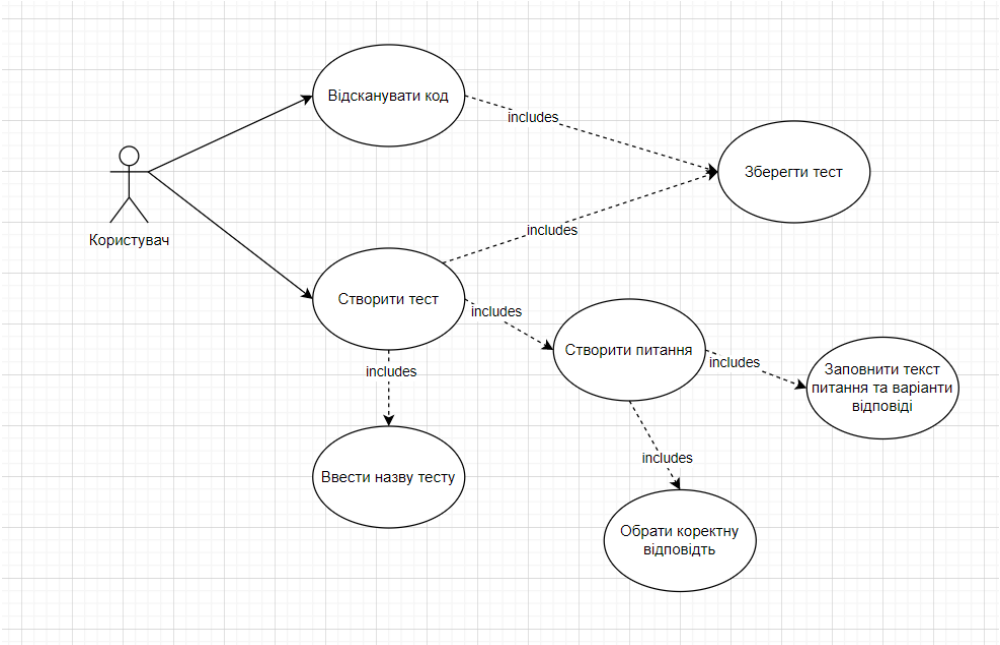

Рисунок 3.12 – Use-case діаграмма створення тесту

ВИСНОВКИ

У рамках кваліфікаційної роботи був створений та реалізований застосунок до мобільних пристроїв у якому можливо реалізувати створення та проходження різноманітних тестів з інформатики використовуючи мобільні пристрої при зберіганні автономії від мережі інтернет.

У цьому застосунку було реалізовано наступні можливості:

– наявність локальної бази даних, що дозволяє користувачу проходити тести без підключення до мережі інтернет;

– реалізовано алгоритм рандомізації кожного тесту, що дозволяє покращити статистику індивідуальності проходження тесту;

– реалізована можливість поширення тестів за допомогою QR кодів;

– при завантаженні застосунку він містить у собі певну базу готових до проходження тестів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. ДЕМОКРАТІЯ, ПРАВА І СВОБОДИ ГРОМАДЯН ТА МЕДІАСПОЖИВАННЯ В УМОВАХ ВІЙНИ. URL: https://www.oporaua.org/report/viyna/antipropaganda/24261-sotsiologichne-doslidzhennia-demokratiia-prava-i-svobodi-gromadian-ta-mediaspozhivannia-v-umovakh-viini. (Дата звернення 10.04.2023 – 21.05.2023).

2. Путятін, Є.П., Гороховатський, В.О., & Матат, О.О. (2006). Методи та алгоритми комп'ютерного зору: навч. посібник.

3. Творошенко, І.С. (2021). Технології прийняття рішень в інформаційних системах: навч. посібник. *Харків: ХНУРЕ*.

4. Гороховатський, В.О., & Творошенко, І.С. (2021). Методи інтелектуального аналізу та оброблення даних: навч. посібник.

5. Кобилін, О.А., & Творошенко, І.С. (2021). Методи цифрової обробки зображень: навч. посібник. *Харків: ХНУРЕ*.

6. Гороховатський В.О., Творошенко І.С. (2022) Аналіз багатовимірних даних за описом у формі множини компонент: монографія. Харків: ХНУРЕ, 124 с.

7. Валерій, Ю.Д., Кобилін, А.М., Кобилін, О.А. (2021) Виконання на мобільних пристроях арифметичних операцій з використанням аксіом класичного та нестандартного інтервального аналізу: С. 128 -132.

8. Kinoshenko, D., Kobylin, O., Mashtalir, S., Stolbovyi, M. (2019) Metric video retrieval speedup by irrelevant data elimination, pp. 176-183.

9. Lyashenko, V., Kobylin, O., Baranchykov, Y. (2018) Ideology of Image Processing in Infocommunication Systems, pp. 47-50

10. Daradkeh, Y.I., Tvoroshenko, I. (2020) Technologies for making reliable decisions on a variety of effective factors using fuzzy logic, pp. 43-50

11. Daradkeh, Y.I., Tvoroshenko, I., Gorokhovatskyi, V., Latiff, L.A., and Ahmad, N. (2021) Development of Effective Methods for Structural Image

Recognition Using the Principles of Data Granulation and Apparatus of Fuzzy Logic, *IEEE Access*, 9, pp. 13417-13428.

12. Daradkeh, Y.I., Gorokhovatskyi, V., Tvoroshenko, I., Gadetska, S., and Al-Dhaifallah, M. (2021) Methods of Classification of Images on the Basis of the Values of Statistical Distributions for the Composition of Structural Description Components, *IEEE Access*, 9, pp. 92964-92973.

13. Gorokhovatskyi, V.O., Tvoroshenko, I.S., and Peredrii O.O. (2020) Image classification method modification based on model of logic processing of bit description weights vector, *Telecommunications and Radio Engineering*, 79(1), pp. 59-69.

14. Daradkeh Y.I., Gorokhovatskyi V., Tvoroshenko I., and Zeghid M. (2022) Cluster representation of the structural description of images for effective classification, *Computers, Materials & Continua,* 73(3), pp. 6069-6084.

15. Гороховатський В.О., Творошенко І.С., Чмутов Ю.В. (2022) Застосування систем ортогональних функцій для формування простору ознак у методах класифікації зображень, *Сучасні інформаційні системи*, 6(3), С. 5-12.

16. Daradkeh Y.I., Gorokhovatskyi V., Tvoroshenko I., and Al-Dhaifallah M. (2022) Classification of Images Based on a System of Hierarchical Features, *Computers, Materials & Continua*, 72(1), pp. 1785-1797.

17. Tvoroshenko I., and Gorokhovatskyi V. (2022) The Application of Hybrid Intelligence Systems for Dynamic Data Analysis, *International Journal of Engineering and Information Systems*, 6(2), pp. 40-48.

18. Гороховатський В., Передрій О., Творошенко І., Марков Т. (2023) Матриця відстаней для множини компонентів структурного опису як інструмент для створення класифікатора зображень, *Сучасні інформаційні системи*, 7(1), С. 5-13.

19. Gorokhovatskyi, V., Gorokhovatskyi, O., Yevgenyi, P., & Olena, P. (2018). Quantization of the Space of Structural Image Features as a Way to.

20. Tvoroshenko, I., & Kukharchuk, V. (2021). Current state of development of applications for recognition of faces in the image and frames of video captures.

21. Gorokhovatskyi, V., & Tvoroshenko, I. (2020). Image Classification Based on the Kohonen Network and the Data Space Modification.

22. Kinoshenko, D., Mashtali, S., Shylyakhov, V., Stolbovyi, M. (2019) Video shots retrieval with use of pivot points, pp. 102-111.

23. V. Gorokhovatskyi, I. Tvoroshenko, Image Classification Based on the Kohonen Network and the Data Space Modification, in: CEUR Workshop Proceedings: Computer Modeling and Intelligent Systems (CMIS-2020), 2020, pp. 1013–1026.

24. Daradkeh, Y. I., Gorokhovatskyi, V., Tvoroshenko, I., & Al-Dhaifallah, M. (2022). Classification of images based on a system of hierarchical features. Computers, Materials & Continua, 72(1), 1785-1797.

25. Daradkeh, Y. I., & Tvoroshenko, I. (2020). Technologies for making reliable decisions on a variety of effective factors using fuzzy logic. International Journal of Advanced Computer Science and Applications, 11(5)

26. Daradkeh, Y. I., Gorokhovatskyi, V., Tvoroshenko, I., & Zeghid, M. (2022). Tools for Fast Metric Data Search in Structural Methods for Image Classification. IEEE Access, 10, 124738-124746. p. 6-10

27. Mashtalir S., Mashtalir V. (2020) Spatio-Temporal Video Segmentation. In: Mashtalir V., Ruban I., Levashenko V. (eds) Advances in Spatio-Temporal Segmentation of Visual Data. Studies in Computational Intelligence, vol 876. Springer, Cham. pp. 161-210.

28. Гороховатський, В. О., Стяглик, Н. І., & Жадан, О. В. (2022). Застосування багатокомпонентної моделі даних для описів класів у задачі класифікації зображень. С. 7.

29. Kinoshenko, D., Mashtalir, S., Shlyakhov, V., & Stolbovyi, M. (2019). Video shots retrieval with use of pivot points. In Advances in Computer Science for Engineering and Education 13 (pp. 102-111).

30. Springer International Publishing Yakovleva, O., & Nikolaieva, K. (2020). Research Of Descriptor Based Image Normalization And Comparative Analysis Of SURF, SIFT, BRISK, ORB, KAZE, AKAZE Descriptors. Advanced Information Systems, 4(4), 89-101.