МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ ЕЛЕКТРОНІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК
СЕКЦІЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ПРОЕКТУВАННЯ

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: «Мобільний додаток для тестування знань з мов програмування»

за спеціальністю 122 «Комп'ютерні науки»,
освітньо-професійна програма «Інформаційні технології проектування»

Виконавець роботи: студент групи ІТ.м-82 Падалиця Дмитро Анатолійович

Кваліфікаційну роботу
захищено на засіданні ЕК
з оцінкою _____ «___» грудня 2019 р.

Науковий керівник _____ к.т.н., доц., Марченко А. В.
(підпис)

Голова комісії _____ Шифрін Д.М.
(підпис)

Засвідчую, що у цій дипломній роботі немає
запозичень з праць інших авторів
без відповідних посилань.

Студент _____
(підпис)

Суми-2019

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук
Секція інформаційних технологій проектування
Спеціальність 122 «Комп'ютерні науки»
Освітньо-професійна програма «Інформаційні технології проектування»

ЗАТВЕРДЖУЮ
Зав. секцією ІТП

_____ В. В. Шендрик

«___» _____ 2019 р.

ЗАВДАННЯ

на кваліфікаційну роботу магістра студентові

Падалиця Дмитро Анатолійович
(прізвище, ім'я, по батькові)

1 Тема проекту *Мобільний додаток для тестування знань з мов програмування*

затверджена наказом по університету від «19» листопада 2019 р. №2305-III

2 Термін здачі студентом закінченого проекту « 10 » грудня 2019 р.

3 Вхідні дані до проекту Інформація, щодо створення аутентифікації користувача
Інформація вибору складності тесту користувача
Екран з основними можливостями програмного продукту для проходження тестування

4 Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) 1.Аналіз предметної області
2.Постановки задачі та методи дослідження
3.Моделювання мобільного додатку
4.Розробка мобільного додатку
5.Тестування

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень 1. Мобільний додаток
2.Діаграма взаємодії
3. Презентація роботи

6. Консультанти випускної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
1. Дослідження предметної області	Марченко А.В.		

Дата видачі завдання _____.

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів випускної проекту	Термін виконання етапів проекту	Примітка
1	Дослідження предметної області	23.09.2019 – 01.10.2019	
2	Аналіз функціональних вимог	02.10.2019 – 09.10.2019	
3	Аналіз нефункціональних вимог	10.10.2019 – 18.10.2019	
4	Розробка дизайну мобільного додатку	18.10.2019 – 24.10.2019	
5	Верстка екранів додатку	24.10.2019 – 30.10.2019	
6	Розробка функціоналу додатку	31.10.2019 – 20.11.2019	
7	α-тестування	21.11.2019 – 23.11.2019	
8	β-тестування	24.11.2019 – 25.11.2019	
9	Виправлення знайдених помилок	26.11.2019 – 30.11.2019	
10	Оформлення ПЗ	01.12.2019 – 04.12.2019	
11	Захист кваліфікаційної роботи	18.12.2019 – 20.12.2019	

Магістрант _____ Падалиця Д.А.

Керівник роботи _____ к.т.н., доц. Марченко А.В.

РЕФЕРАТ

Тема кваліфікаційної роботи магістра «Мобільний додаток для тестування знань з мов програмування».

Пояснювальна записка складається зі вступу, 5 розділів, висновків, списку використаних джерел із 14 найменувань, додатків. Загальний обсяг 82 сторінок, у тому числі 52 сторінок основного тексту, 2 сторінки списку використаних джерел, 27 сторінок додатків.

Кваліфікаційна робота магістра присвячена розробці мобільного додатку для тестування знань з мов програмування. В роботі проведено аналіз предметної області, аналіз функціональних вимог та не функціональних вимог. Було схематично розроблено прототип додатку та користувацький інтерфейс. З даних макету екранів додатку було розроблено візуальну частину додатку. Підключено систему аутентифікації користувача через Firebase Auth та встановлено функціональну частину додатку. Проведений етап тестування мобільного додатку для знаходження помилок згідно отриманих даних тестування додатку було виправлено всі знайдені помилки.

Результатом проведеної роботи є готовий, функціональний - мобільний додаток для тестування знань з мов програмування.

Ключові слова: мобільний додаток, тестування, проходження тестування, тестування знань, Java, Firebase Auth

ЗМІСТ

ВСТУП

Одним з напрямків узагальнення теоретичного матеріалу є система тестування, яка впроваджена в навчальних закладах. Під час виконання проекту встановлено що, програмний продукт необхідний для обслуговування користувача під час проходження тестування.

Мета проекту - створення мобільного додатку для проходження тестування з декількох мов програмування. Мотивуючи чинником для ідеї створення продукту стає її актуальність в самій системі тестування та тематикою проходження тестів з мов програмування.

Під час проходження роботи було встановлено та записано: мета проекту, задачі, які буде виконувати додаток, вибір методів технологій на чому буде проводитися розробка та планування робіт – обрав способи планування календарного графіку методом Ганта. В частині моделювання мобільного додатку було проаналізована цільову аудиторію, сценарій взаємодії клієнта та додатку. Побудовано схематично прототип та структура проходження головних екранів. Описано архітектуру мобільного додатку. Для формалізації процесів була розроблена діаграма IDEF0. Наступним кроком було етап реалізація на програмному рівні де було розглянуто основний функціонал додатку. Після створення додатку був етап тестування, перевірено всі можливі взаємодії користувача з програмним продуктом.

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1Актуальність проблеми

Розглядаючи питання системи або процесу навчання зустрічаємо не приємні проблеми, які не всі правильно вирішують або зовсім не починають роботи кроки для подолання тяжких моментів навчання. Основними проблема це не ефективне використання часу на навчання та не якісне засвоєння теоретичного або практичного матеріалу. “Досвід сам по собі нічому не вчить... Без теорії досвід не має сенсу. Без теорії немає питань, що можна задати. Отже, без теорії немає навчання.”[1] – цитата американського вченого, статиста Едварда Демінга, з якою погодився кожен досвідчений професіонал своєї справи, тому зроблено певний підсумок засвоєння теоретичного матеріалу є важливим та необхідним етапом навчання, а після проходження через деякий час відповідного тесту знання почнуть залишатися у свідомості, а для остаточного завершення теми потрібно опрацювати тематику практично та декілька разів. Спосіб проходження тесту є одним з головних, допоміжних рушіїв людини для усвідомлення отриманої інформації.

Обрання теми тестування мов програмування, а саме JavaScript та Python вирішено за допомогою проведеного аналізу популярності відповідних мов програмування на порталі dou.ua (рис 1.1) Графік складений з відгуків програмістів, які живуть в Україні та пишуть відповідними мовами у робочий час.

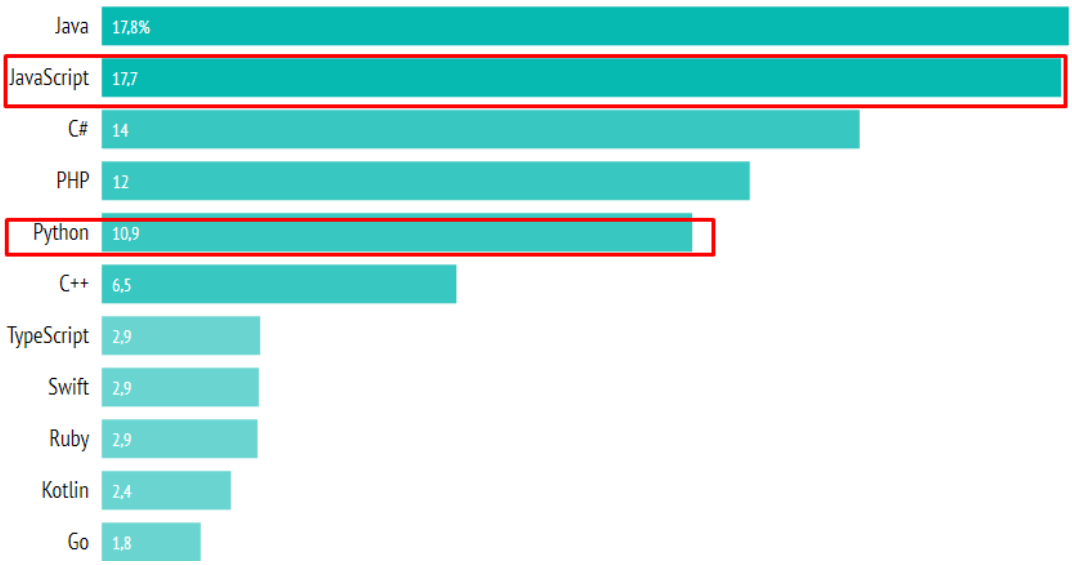


Рисунок 1.1 – Статистика популярності мов програмування на листопад 2019

Наступний графік від порталу stackoverflow.com – відома система для it – фахівців, являє собою питання та відповідь. Графік збудований на основі заданих користувачами питань на порталі до відповідних технологій (рис.1.2)

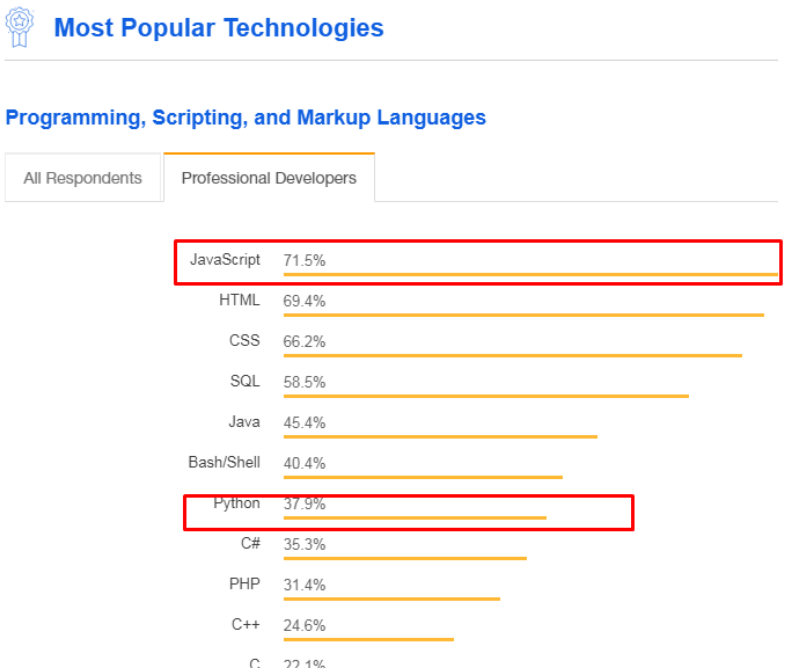


Рисунок 1.2 – Статистика запитів користувачами з відповідних технологій

Рейтинг від tiobe.com - світовий рейтинг мов програмування за листопад 2019 (рис. 1.3) Сервіс збирає інформацію про популярність мов програмування в різних країнах і формує щомісячні рейтинги. Дані засновані на кількості курсів, думках досвідчених програмістів і запитах в Google, Bing, Yahoo!, Wikipedia, Amazon, YouTube і Baidu. [2]

Dec 2019	Dec 2018	Change	Programming Language	Ratings	Change
1	1		Java	17.253%	+1.32%
2	2		C	16.086%	+1.80%
3	3		Python	10.308%	+1.93%
4	4		C++	6.196%	-1.37%
5	6	^	C#	4.801%	+1.35%
6	5	v	Visual Basic .NET	4.743%	-2.38%
7	7		JavaScript	2.090%	-0.97%
8	8		PHP	2.048%	-0.39%
9	9		SQL	1.843%	-0.34%
10	14	^^	Swift	1.490%	+0.27%

Рисунок 1.3 – Світовий рейтинг мов програмування за листопад 2019

Зібравши відповідну статистику бачимо тенденції популярності мови програмування JavaScript - на українському ринку. Відомі фреймворки JavaScript: React, View.js, Angular, Node.js є актуальними та необхідними технологіями для західних замовників. Також серед часто заданих питань на порталі stackoverflow, JS є лідером для професійних розробників та студентів. Python також вже почав викликати інтерес серед розробників на світовому рівні. Тому було обрано дані мови програмування, які в наш час посідають провідні місця серед програмістів та новачків. Для успішного вивчення теоретичного або практичного матеріалу потрібно завжди закріплювати свої знання спочатку в тестовій формі, а згодом і самостійно в середовищі розробки програмного забезпечення. Кожен хто знайомий з будь-якими мовами програмування на базовому рівні або має певний досвід в розробці ПП буде зацікавлений перевірити свої знання та отримати результат свого

тесту. Користувачу потрібно знайти для себе відповідний блок з мовою програмування та рівнем складності та перевірити свої здібності. У даній роботі представлено проект мобільного додатку – тестування мов програмування, де користувач знаходить для себе практичні або теоретичні питання, а далі вже перевіряє свої навички та отримує результат проходження.

1.2Аналіз мобільних додатків для проходження тестування

Під час планування проекту було розглянуто декілька популярних та подібних додатків з метою аналізу функціоналу продукту, користувацького інтерфейсу, виокремлення сильних факторів та знаходження недоліків.

Схожі мобільного додатку бувають з вмістом теоретичним матеріалом та проходження тестування, турнірні дуелі між користувачами додатка та тільки проходження тесту. Перший приклад один з популярних додатків Solo learn, зображений на рисунку 1.4, на тему тестування та освіти загалом, яка має велику базу даних теоретичного матеріалу та тестових завдань. Користувачі також можуть змагатися між собою хто більше відповість правильних питань той і переміг таким чином присутній дух змагання, який подобається користувачам. Недоліки додатку полягають у відсутності ,частково, актуального контенту, що не відповідає сучасному стану розвитку мов програмування, що може заплутати початківців, присутність реклами, яку можна відключити тільки купляючи підписку.

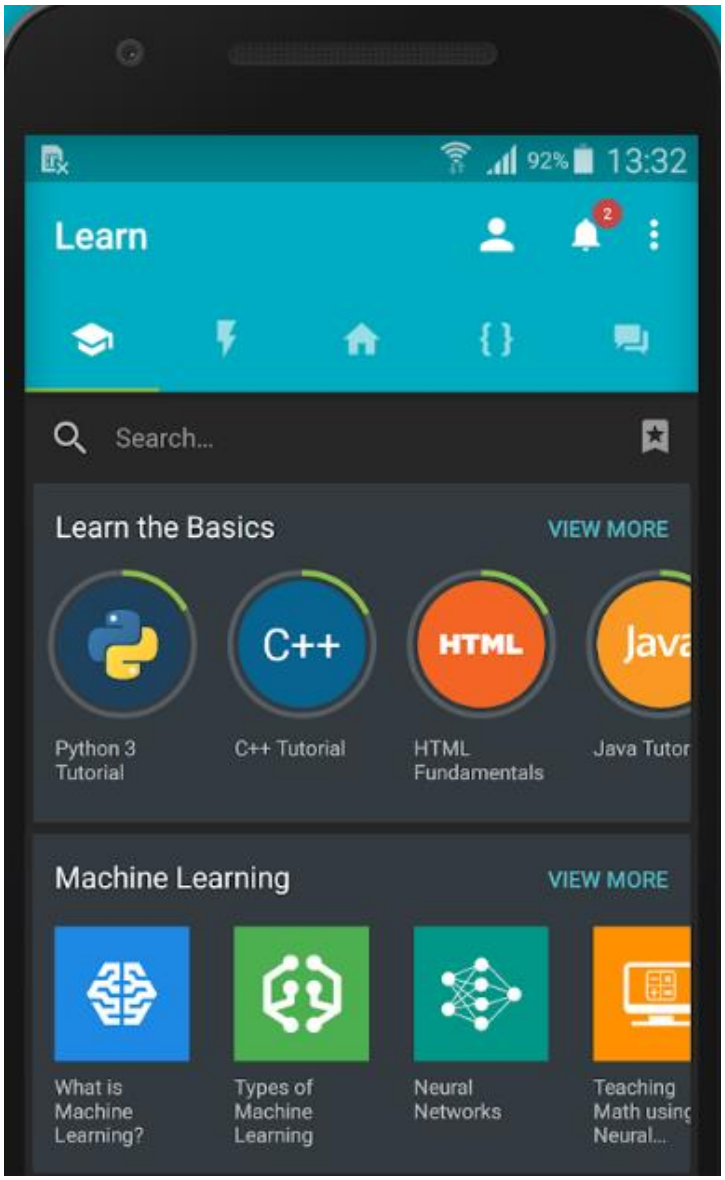


Рисунок 1.4 – Головна сторінка мобільного додатку “Solo learn”

Наступний додаток Coding Quiz зображений на рисунку 1.5, влаштований тільки для тестування, представляє велику кількість питань з рівнем складності в тестовому форматі. Сильні сторони приємний та інтуїтивно зрозумілий інтерфейс та великий вибір різних мов програмування. Також додаток має присутність реклами, яка відключається за гроші. Присутня тільки англомовна версія додатку, тому хто не володіє буде не комфортно користуватися.

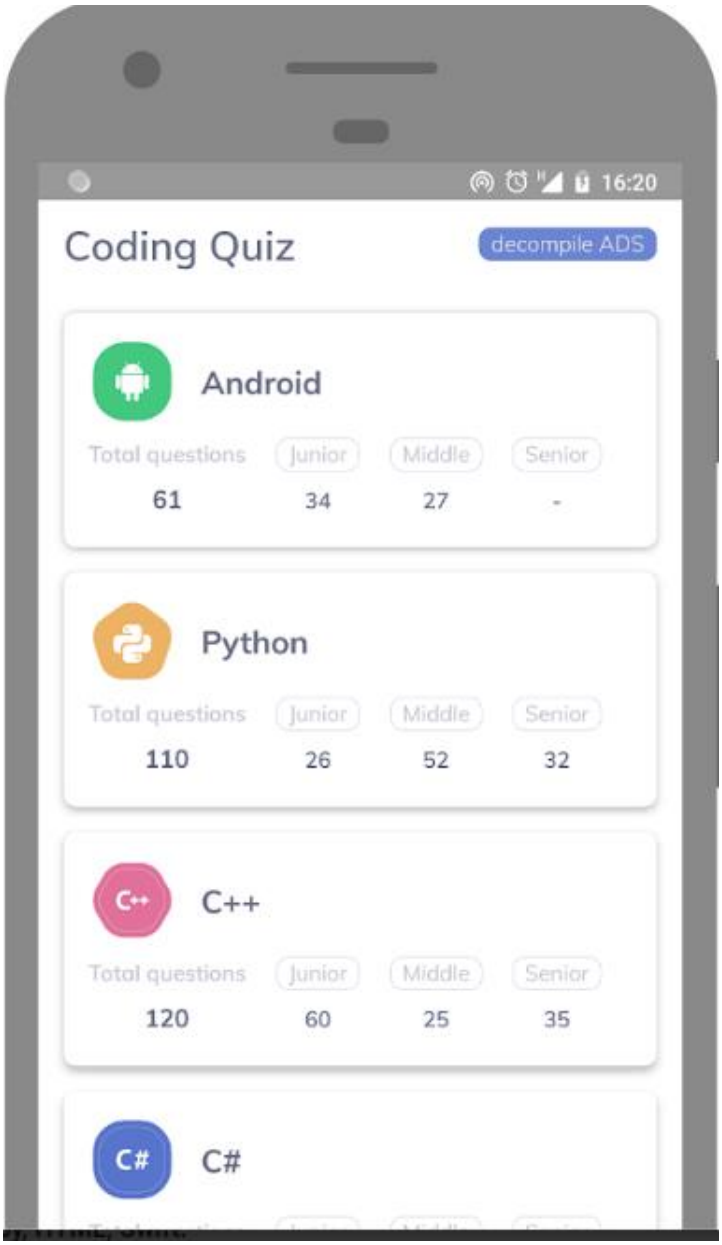


Рисунок 1.5 – Головна сторінка мобільного додатку “ Coding Quiz ”

Додаток Programming Quiz зображений на рисунку 1.6 має широкий функціонал для вибору проходження тесту на більшість тем з комп'ютерних наук. Також продукт не має реклами, що є позитивним чинником для користувачів. Після проходження тесту користувач отримує результат проходження та відповідний коментар який залежить від оцінки складання тесту. Згідно отриманих даних було складено порівняльну таблицю 1.1

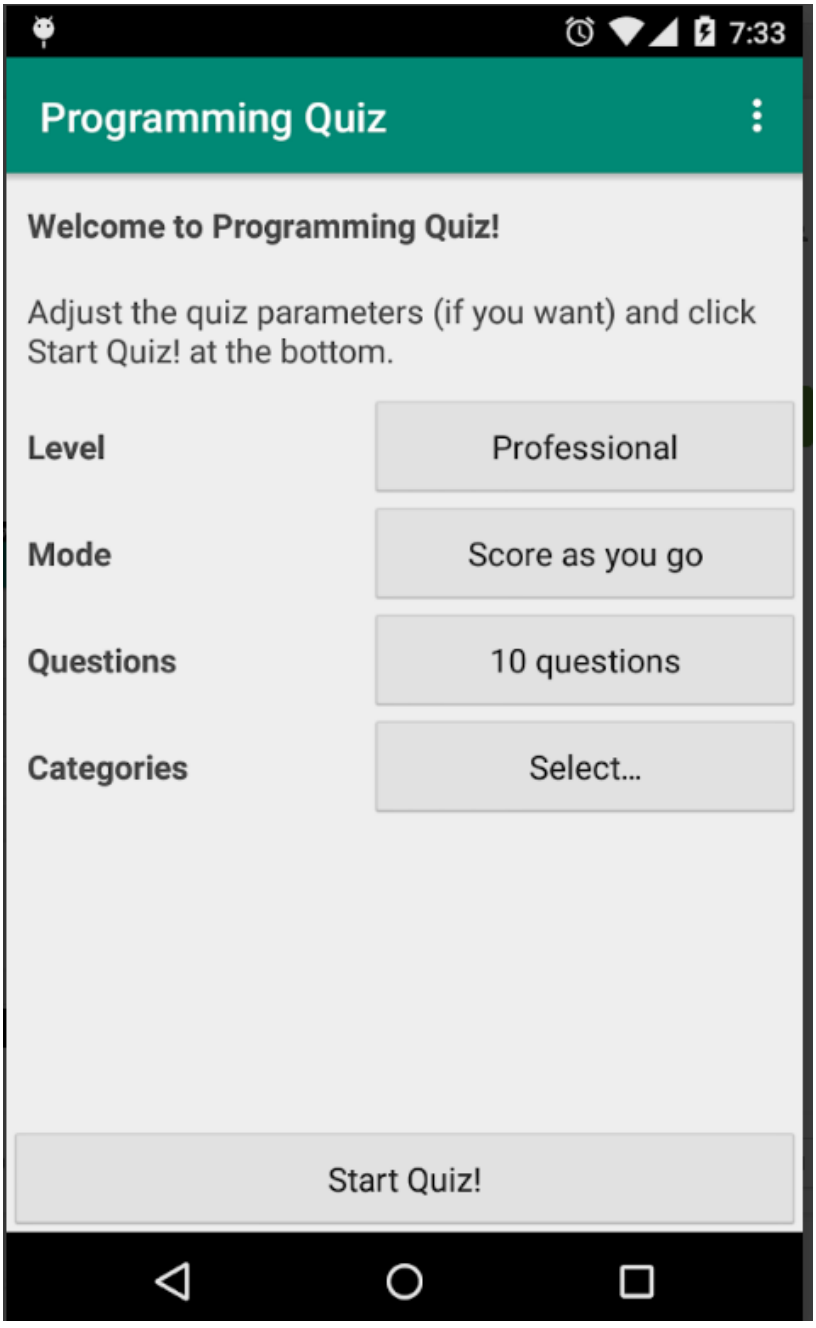


Рисунок 1.6 – Головна сторінка мобільного додатку “ Programming Quiz”

Таблиця 1.1 – Порівняння мобільних додатків зі схожим функціоналом

Характеристика	Solo learn	Coding Quiz	Programming Quiz	Challenge Quiz
Зручний інтерфейс	+	+	-	+
Мови користування Ukranian	-	-	-	+
Сучасний дизайн	+	+	-	+
Відсутня нав’язлива реклама	-	-	+	+
Можливість обрання рівня складності	-	+	-	+
Кількість завантажень в Google Play	+5 000 000	+10 000	+10 000	

Зробивши відповідні підсумки на таблиці 1.1 продемонстровано основні переваги мобільного додатка. Схожі додатки мають свою мету, задачі та цільову аудиторію тому вони можуть відрізнятися, але мета порівняння знаходження позитивних властивостей та характеристик, які потрібно уникнути при створенні ПП, що зробить продукт ціннішим серед потенційної аудиторії.

2. ПОСТАНОВКА ЗАДАЧІ ТА МЕТОДИ ДОСЛІДЖЕННЯ

2.1.Задачі програмного продукту

На сьогоднішній день велика кількість людей вважає за потрібне знаходити час для покращення освітніх навичок та здобувати нові знання для власних потреб з метою самовиховання, яка є результатом досягнення успіхів у професійній діяльності. Кожен обирає власний спосіб, як користуватися отриманою інформацією з підручника, мережі або від товариша, але для закріплення та удосконаленню нових знань потрібно пройти тест на відповідну, профільну тематику. Тому створити мобільний додаток з зручним інтерфейсом для проходження тестів з мов програмування є рішучим фактором для самоосвіти користувача.

Повний список функціональних вимог показано в табл. 2.2

- Аутентифікація користувача
- Вибір декількох розділів з мовами програмування(JavaScript, Python)
- Наявність “життів”, після втрати яких завершується тест з відповідним коментарем та оцінкою.
- Після складання тесту можливість перегляду правильних і не правильних відповідей
- Наявність панелі інструментів з посиланнями на документацію з теорією на інші джерела

Корисний додаток для проходження тестування, буде містити профільні питання та можливі відповіді з яких тільки одна правильна. Основний функціонал є вибір предмета тесту далі вибір складності та безумовно сам процес проходження тесту і вже в результаті отримання оцінки проходження. Під час проходження тестування користувач має можливість пропустити питання обрати правильний варіант, який він вважає за потрібне та вийти з процесу проходження тесту.

Додаток допоможе перевірити знання користувачів на різних рівнях складності з декількох мов програмування таким чином з мотивує підкреслити знання або перевірити правильність відповіді, що в будь-якому випадку корисно вплине на користувача.

2.2Вибір методів

Після проведення аналізу актуальності роботи можна зазначити про популярність мови програмування Java, яка займає першу сходинку за рейтингом tiobe.com та є одним з кращим інструментом для написання Android додатків. Більшість смартфонів працює на базі операційної системи Android це підтверджує статистика від statcounter.com (рис. 2.1), тому було вирішено писати додаток для Android користувачів. Досвідчені розробники від Google пропонують користуватися середовищем Android Studio, яка бездоганно підтримує мову Java, має емулятор для тестування додатка та систему контроля версій Git. Дані питань тестів буде складати у відповідному JSON файлі. Аутентифікація користувачів відбувається за допомогою Firebase Auth – це служба, яка буде виконувати серверну частину аутентифікації додатка.

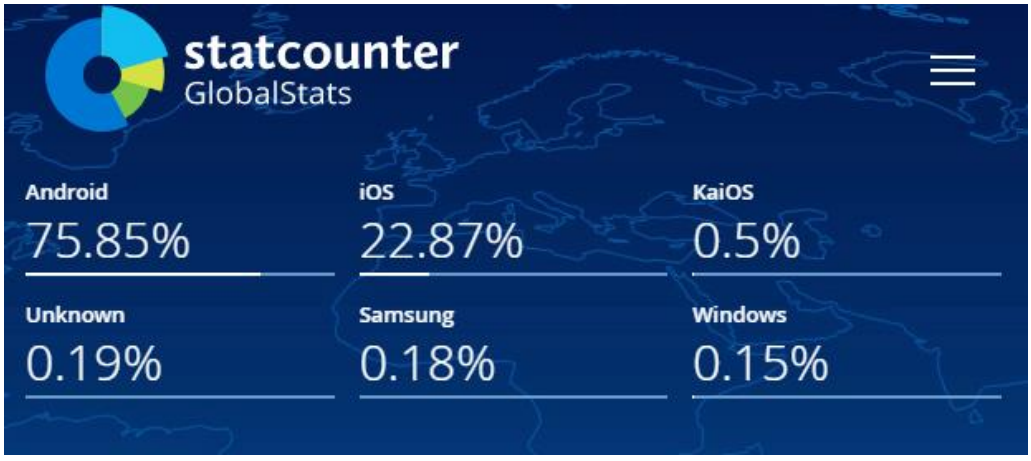


Рисунок 2.2 Статистика користувачів мобільних ОС

Firebase платформ обирають тому що вона швидко та легко інтегрується до додатків, сайті, не потрібно писати налаштування серверної частини, має оновлення

в реальному часі, має аналітику користувачів. Платформа містить безліч корисних функцій для застосування, в тому числі серверний код для мобільних сервісів, статистику, а також інструменти для монетизації і розширення аудиторії. Потреба використання Firebase в тому що, пакет розробника Firebase об'єднує інтуїтивно зрозумілі API, позбавляючи від необхідності керувати окремими пакетами[3].

2.3Планування робіт

Виокремлення мети проекту методом SMART

SMART-цілі - метод постановки конкретних цілей, реалістичних завдань, які чітко обмежені часі та мають свій кінцевий термін та аналіз роботи виконавця який знає скільки роботи вже виконано.

Методологія SMART являє собою концепцію правил:

- Конкретність - прозора постановка мети, де ціль повинна бути зрозумілою.
- Вимірюваність – виконавець має уявлення на якому етапі йде розробка проекту
- Досяжність – мета конкретної задачі закріплена за одним виконавцем який повинен розуміти всі критерії цілі та нести відповідальність.
- Реалістичність – мета повинна мати свої певні реальні потреби, які залежать від виконавця.
- Обмеженість в часі – проект має бути виконаний в певний, заданий час.
- Кожен етап проекту потрібен мати свій ліміт часу.

Результати розміщені у таблиці 2.1

Таблиця 2.1 – Висновок мети методом SMART

Specific (Конкретність)	Мобільний додаток для проходження тестування з мов програмування
Measurable (Вимирюваність)	Результатом роботи є відгуки користувачів, які надали свою оцінку
Achievable (Досяжність)	Реалізація виконана на мові програмування Java, для користувачів ОС Android
Realistic (Актуальність)	В наявності є всі необхідні технічні та програмні засоби.
Timed (Обмеженість в часі)	Ціль має свій кінцевий термін. Кожен етап повинен бути виконаний вчасно. Терміни повинні бути обговорені виконавцем та керівним проекту або замовником.

Планування змісту структури робіт проекту (WBS)

“Work Breakdown Structure - є ієрархічним розкладом роботи, яка повинна бути виконана для досягнення цілей проекту та створення необхідних результатів.”[4] Робота дерева проекту WBS мусить бути поділена на головні задачі та під задачі.

Переваги використання WBS:

- Чітка структуризація проекту;
- Надає допомогу в описі змісту проекту для зацікавлених сторін;
- Розподіл обов'язків серед команди проекту;
- Наглядно описує конкретні етапи проекту

Створена WBS-діаграма представлена на схемі Б.1 у Додатку Б

Планування структури організації(OBS)

“Organizational Breakdown Structure - це ієрархічна модель, що описує сформовані організаційні рамки виконавців роботи, які поділені на свої задачі.” [5]

Згідно проектування OBS діаграми, представлена на схемі у Додатку Б

Маємо команду проекту:

- розробник проекту – Падалиця Д.А.
- тестувальник – Тарасенко О.Н.
- керівник – Марченко А.В.

Побудова матриці відповідальності

Після побудови структур OBS та WBS складають матрицю відповідальності проекту, за кожним етапом проекту повинен бути відповідальний учасник команди проекту. Етап роботи необов’язково повинен виконувати один член команди їх може бути декілька, головне має бути відповідальний, який контролює та відповідає за процес роботи та результат. Якщо на певному етапі декілька виконавців, відповідальний розподіляє на виконавця та помічників або консультуючих.

Матриця відповідальності представлена у таблиці Б2 у Додатку Б.

Побудова календарного графіку виконання

Побудова календарного графіку виконання проекту неодмінно важлива частина тому розробник має наглядний план дій про тривалість виконання робіт. За допомогою діаграми Ганта розробник бачить терміни кожної задачі через те, що багато завдань проекту часто залежать один від одного, буває важко вирахувати, скільки часу необхідно відвести на завдання, коли саме її необхідно почати і до якого часу закінчити. Стовпці в діаграмах Ганта використовуються як індикатор часу, яку необхідно відвести на завдання, і таким чином, виконавець отримуєте більш широку і детальну картину всього проекту і його кінцеві терміни задачі.[6]

Діаграма Ганта було зроблено за допомогою онлайн сервісу GantPro в некомерційних цілях. Проект розробки додатку було розбито на шість частин, кожна частина включає свої задачі, які мають свою тривалість в днях. Загалом кожна задача достатньо унікальна тому де – які задачі виконувались за своєю тривалістю по різному. Кожна задача включає прогрес та статус виконання, тому між тотожніми задачами є логічні зв’язки. Всі шість частин проекту мають бути послідовними і виконуватись вчасно.

На рисунках 2.1 – 2.3 представлена діаграма Ганта

1	**Дослідження предметної області**	23.09.2019	01.10.2019	1н 2д
1.1	Вивчення документації	23.09.2019	25.09.2019	3д
1.2	Пошук варіантів реалізації поставленого проекту	26.09.2019	28.09.2019	3д
1.3	Аналіз додатків схожих за тематикою	29.09.2019	01.10.2019	3д
	Добавить задачу | Добавить веху			
2	**Анаіз функціональних вимог**	02.10.2019	09.10.2019	1н 1д
2.1	Авторизація користувача	02.10.2019	03.10.2019	2д
2.2	При відновленні пароля, користувачу потрібно вказати свою пошту, яка вже була зареєстрована	03.10.2019	03.10.2019	1д
2.3	Мобільний додаток повинен на запит користувача відправити на пошту інструкцію відновленн...	04.10.2019	04.10.2019	1д
2.4	Користувач потрібно перейти за посиланням та вказати новий пароль для свого профілю щоб...	05.10.2019	05.10.2019	1ч
2.5	ПП повинен відображати головний екран після проходження аутентифікації	06.10.2019	06.10.2019	1д
2.6	ПП повинен відображати панель навігації	07.10.2019	07.10.2019	1д
2.7	У ПП повинен бути передбачена вихід з тестування	07.10.2019	07.10.2019	1д
2.8	У ПП повинна бути можливість переходу на сайт через додаток на теоретичний матеріал	07.10.2019	07.10.2019	1д
2.9	Користувацький інтерфейс ПП повинен бути інтуїтивно зрозумілим для користувача	08.10.2019	08.10.2019	1д
2.10	У ПП повинно знімати " життя " якщо користувач не правильно відповідає на питання тесту:	08.10.2019	08.10.2019	1д
2.11	У ПП повинно припиняти тестування якщо користувач відповів не правильно 3 рази	09.10.2019	09.10.2019	1д
2.12	ПП повинний показувати результат проходження тесту	09.10.2019	09.10.2019	1д
	Добавить задачу | Добавить веху			
3	**Аналіз не функціональних вимог**	10.10.2019	18.10.2019	1н 1д 1ч
3.1	ПП маж підтримувати ОС Android 8.1	10.10.2019	14.10.2019	5д
3.2	ПП повинен бути виконаний українською мовою	15.10.2019	15.10.2019	1д
3.3	Реалізація ПП відбувається Java у середовищі розробки Android Studio	16.10.2019	18.10.2019	2д 1ч
	Добавить задачу | Добавить веху			
4	**Розробка концепції дизайну**	18.10.2019	24.10.2019	6д
4.1	Підбір кольорів зовнішнього виду додатку	19.10.2019	19.10.2019	7ч
4.2	Підбір відповідних шрифтів та іконок, логотипу	18.10.2019	19.10.2019	1д
4.3	Поетапна розробка макетів	18.10.2019	24.10.2019	6д

Рисунок 2.2 Календарний графік роботи проекту(початок)

5	**Разработка мобільного додатку**	24.10.2019	20.11.2019	3н 6д	
5.1	Створення відповідних файлів з питаннями тесту та відповідями	24.10.2019	24.10.2019	7ч	
5.2	Верстка головних екранів користувача	24.10.2019	30.10.2019	6д	
5.3	Розробка аутинтифікації користувача	31.10.2019	01.11.2019	1д	
5.4	Підключення панелі навігації користувача	01.11.2019	02.11.2019	1д	
5.5	Розробка сторінки про додаток	02.11.2019	02.11.2019	1д	
5.6	Розробка головного меню	03.11.2019	06.11.2019	3д	
5.7	Розробка вибору складності	06.11.2019	07.11.2019	1д	
5.8	Розробка фінального результату тесту	07.11.2019	10.11.2019	3д	
5.9	Встановлення взаємодії всього функціоналу додатка	11.11.2019	20.11.2019	1н 2д 1ч	
	Добавить задачу | Добавить веху				
6	**Тестування**	20.11.2019	30.11.2019	1н 3д	
6.1	1α-тестування	20.11.2019	21.11.2019	1д	
6.2	2β-тестування	21.11.2019	22.11.2019	1д	
6.3	3Тестування зручності	22.11.2019	25.11.2019	3д	
6.4	4функціональне тестування	27.11.2019	28.11.2019	1д	
6.5	5Знаходження дрібних помилок	28.11.2019	29.11.2019	1д	
6.6	7Виправлення помилок	29.11.2019	30.11.2019	1д	
	Добавить задачу | Добавить веху				

Рисунок 2.3 Календарний графік роботи проекту (завершення)

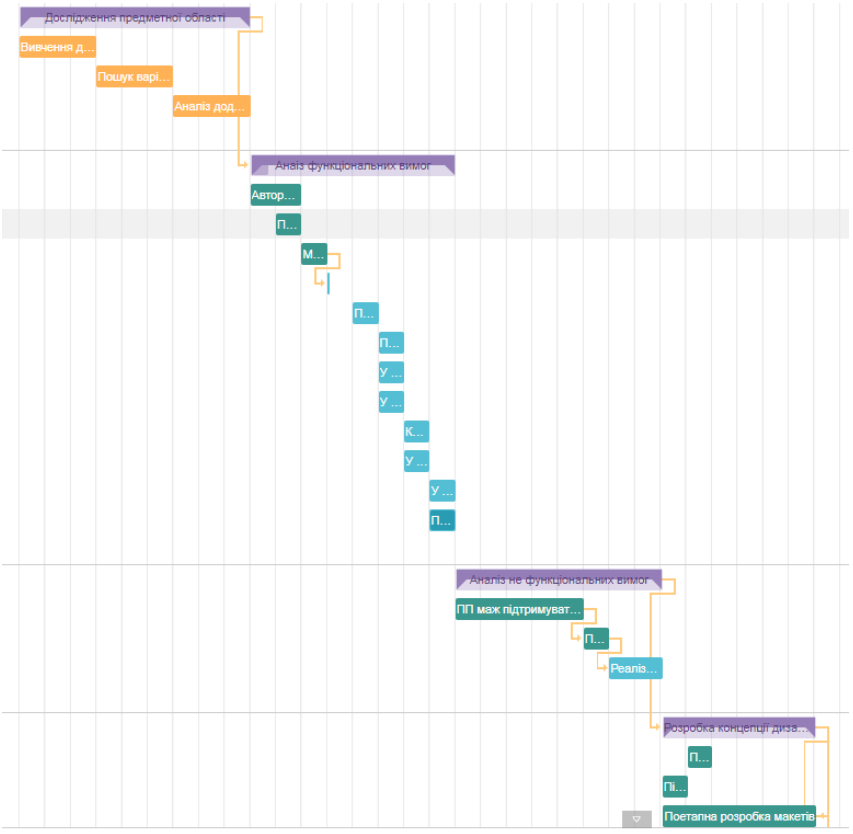


Рисунок 2.4 Діаграма Ганта в візуальній формі

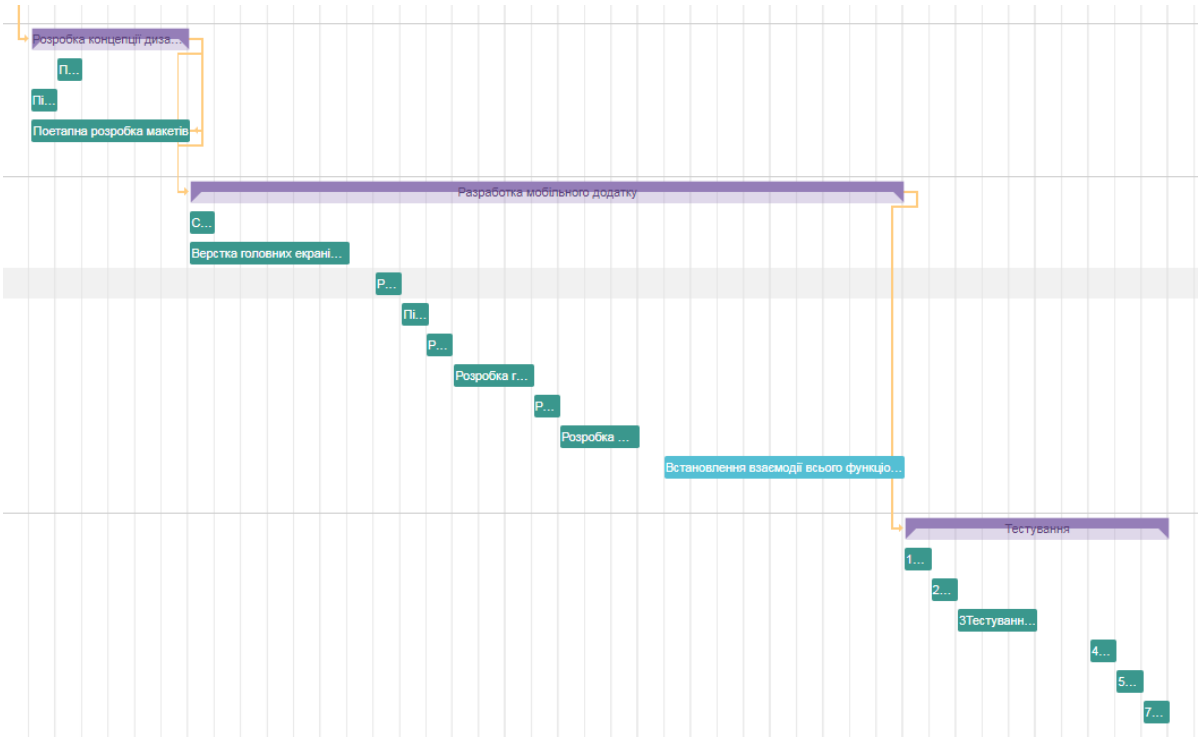


Рисунок 2.5 Діаграма Ганта в візуальній формі

Управління вимогами

“Функціональні вимоги - це опис дій програмного продукту. Він описує програмну систему або її компонент. На приклад може бути взаємодія з користувачем, запит даних користувача. Функціональні вимоги також називаються функціональними специфікаціями.”[7]

Таблиця 2.2 – Список функціональних вимог

№	Вимога
1	Користувачу потрібно зайти в свій профіль або зареєструватися
2	Користувачу потрібно вказати у екрані відновлення паролю свою пошту, яка вже буда зареєстрована в системі
3	Мобільний додаток повинен на запит користувача відправити на пошту інструкцію відновлення паролю
4	Користувач потрібно перейти за посиланням та вказати новий пароль для свого профілю щоб відновити доступ до додатку

Продовження таблиці 2.2

5	ПП повинен відображати головний екран після проходження аутентифікації
6	ПП повинен відображати список всіх рівнів складностей
7	ПП повинен відображати панель навігації
8	У ПП повинен бути передбачена вихід з тестування
9	У ПП повинна бути передбачена можливість переходу на сайт через додаток на теоретичний матеріал
10	Користувацький інтерфейс ПП повинен бути інтуїтивно зрозумілим для користувача
11	У ПП повинно знімати “ життя ” якщо користувач не правильно відповідає на питання тесту:
12	ПП повинно припиняти тестування якщо користувач відповів не правильно 3 рази
13	ПП повинний показувати результат проходження тесту(кількість правильних балів, не правильних та пропущених) Та список пройдених тестів з позначкою правильна відповідь та ваша відповідь

“Нефункціональна вимога (NFR) визначає атрибут якості програмної системи. Вони оцінюють програмну систему на основі чутливості, зручності користування, безпеки, портативності та інших нефункціональних стандартів, що мають вирішальне значення для успіху програмної системи. Невиконання нефункціональних вимог може призвести до систем, які не задовольняють потреби користувача. (табл. 2.3)” [8].

Таблиця 2.3 – Список нефункціональних вимог

№	Вимога
1	ПП має бути підтримувати ОС Android 8.1
2	ПП повинен бути виконаний українською мовою.
3	Реалізація ПП відбувається мовою Java у середовищи розробки Android Studio, підключена система Firebase

Управління ризиками

Ідентифікація ризиків - завчасно оцінити та проаналізувати всі можливі ризики проекту та структуровано описати їх та їхні наслідки. В складанні такого списку ризиків беруть участь менеджер проекту старший розробник, архітектор проекту, замовник та команда аналітиків. Якісна оцінка ризиків - процес подання якісного аналізу ідентифікації ризиків і визначення ризиків, що вимагають швидкого реагування. Така оцінка ризиків визначає ступінь важливості ризику і вибирає спосіб реагування.

Якісна оцінка ризиків може здійснюватися поширеним методом аналогій, сутність якого полягає в аналізі сукупності даних за аналогічними проектами. Тут проводять дослідження наслідки впливів на них несприятливих чинників для того, щоб точно визначити потенційний ризик реалізації нових проектів.

“Основна складність використання методу аналогій полягає в точному і правильному виборі аналогів, оскільки не існує формальних критеріїв для установки ступеня подібних ситуацій” [9].

Таблиця 2.3 – Матриця ризиків

Оцінка	Ймовірність виникнення:	Величина втрат:
1	Вірогідність мінімальна	Мінімальна
2	Практично малоймовірно	Низька
3	Можливе виникнення на достатньому рівні	Середня
4	Виникнення загрози скоріше ймовірне	Висока
5	Вірогідність досить близька до катастрофи	Максимальна

Ймовірність втрат показана в таблиці Б.3 Додатку Б.

На рисунку 2.5 Представлена матриця ймовірності втрат

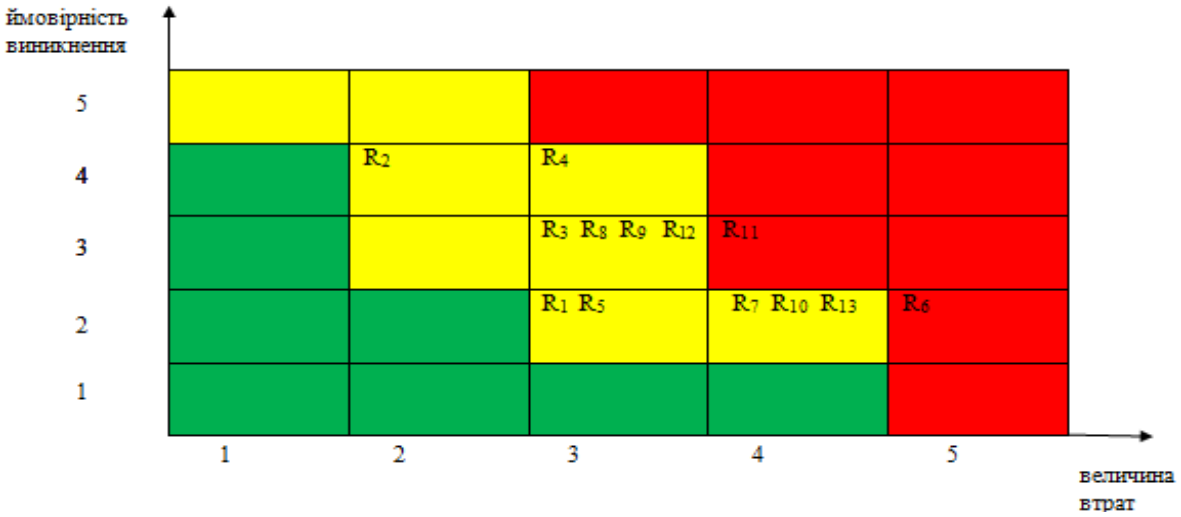


Рисунок 2.5 – Матриця ймовірності втрат

Класифікація за ступенем впливу та за рівнем ризику

Класифікація за ступенем впливу:

- ігноровані (1≤R≤3);
- незначні (7≤R≤9);
- помірні (10≤R≤12);
- вище середнього (15≤R≤20);

- катастрофічні (23≤R≤29).

Класифікація за рівнем ризику:

- прийнятні ризики;
- виправданні ризики;
- недопустимі ризики;

Класифікація за ступенем впливу та за рівнем ризику показана в таблиці Б.4Додатку Б.

План по усуненню ризиків

- Відкритість та адекватна робота з замовником проекту
- Взаємодія з замовником заздалегідь вказаними датами
- Прототипування і узгодження інтерфейсу
- Поставки замовникам результати виконаних задач
- Надійний вибір інструментів виконання проекту

3. МОДЕЛЮВАННЯ МОБІЛЬНОГО ДОДАТКУ

3.1.Збір вимог

Цільова аудиторія

Результатом даного проекту є мобільний додаток, що слугує помічником для користувачів, які бажають перевірити свої здібності у проходженні тестування з мов програмування JavaScript, Python. Користувач може обрати складність розділу та приступити до проходження тесту вкінці проходження користувач отримує результат з коментарями та відповідним балом. Цільова аудиторія –зацікавлені особи, які полюбляють програмувати тому проходження тестів на відповідну тематику буде цікаве та корисне.

Сценарії взаємодії

Після відкриття додатку користувач бачить стартову сторінку з авторизацією, далі є відповідно зрозумілі поля вводу персональних даних електронну пошту та пароль також є кнопка для реєстрації користувача якщо це перше користування додатком та присутня можливість відновлення пароля, в даному екрані користування потрібно вказати вже зареєстровану електронну адресу та відправити дані. В найближчий час на пошту прийде повідомлення про відновлення пароля де потрібно буде перейти за посиланням та придумати новий пароль. Наступним етапом користувач потрапляє в головний екран де вже він має змогу обрати відповідний розділ (мову програмування) далі відповідний рівень складності, а потім вже почне тестування. Під час проходження користувач відразу бачить кількість “життів” втративши їх всі, тестування завершується. В момент натиску на відповідний варіант відповіді користувач одразу бачить правильно він відповів чи не правильну. Правильність відповіді буде зеленим кольором, а не правильна відповідь - червоним. Таким чином після завершення проходження користувач потрапляє на екран результату, одразу видно кількість правильних та не правильних відповідей, коментар результату тестування та перелік всіх пройдених тестів з

відповідними підписом відповідь користувача та правильна відповідь. Переглянувши всі відповіді також має змогу повторно пройти тест натиснувши на відповідну кнопку.

3.2 Проектування прототипу та структура проходження додатка

На даному етапі моделювання програмного продукту вказано прототип екранів взаємодії з користувачем - це потрібно для встановлення основних елементів інтерфейсу для вже більш легкого проектування дизайну UI. Дана модель використання прототипу додатка використовується під час отримання інформації від замовника, де він вказує основні потреби інтерфейсу та всіх варіантів взаємодії додатка з користувачем, які йому потрібні, а після погодження прототипу проект переходить на стадію створення UI. Зовнішня структура додатку представлена у вигляді об’єкта де вказано основні елементи інтерфейсу поля вводу даних (пошта, пароль), кнопки, панель інструментів . Після запуску додатку користувач потрапляє до вікна авторизації.(рисунок 3.1). За даними технічного завдання, вказаний Додаток А, користуватися програмним продуктом можуть тільки авторизовані користувачі.

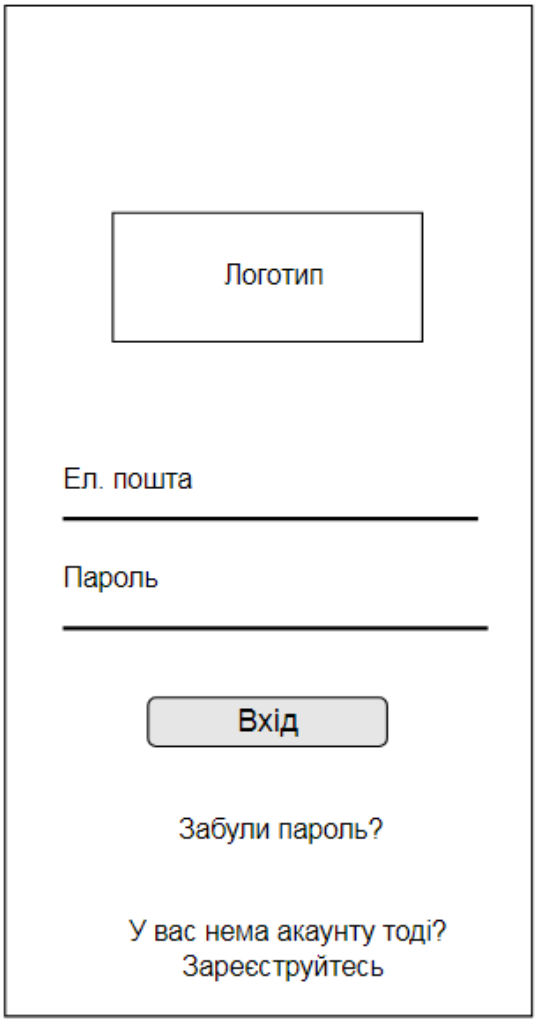


Рисунок 3.1– Схематичне зображення екрану додатку авторизації

Після успішної авторизації чи реєстрації користувач потрапляє до головного екрану обрання розділу тесту (рисунок 3.2). Структура основної частини головного екрану складає вибір між мовами програмування (JavaScript, Python), панель інструментів в якому присутнє вікно з описом основних функцій додатка та посиланнями на теоретичний матеріал. Після обрання відповідного розділу з’являється екран рівня складності тесту продемонстрований на (рисунок 3.3).

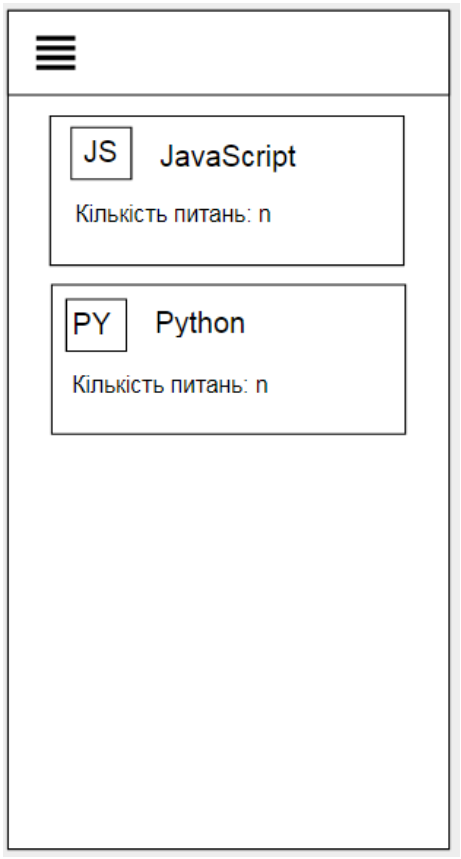


Рисунок 3.2– Схематичне зображення головного екрану додатку

Рисунок 3.3– Схематичне зображення екрану вибору складності тесту

Далі було спроектовано екран проходження тесту.(рис. 3.4). Потрібно звернути увагу на можливість виходу з тестування, кількість “життів” користувача, питання тесту, варіанти відповідей та дію – пропустити питання. Та кінцевий етап тестування – результат.(рис. 3.5)

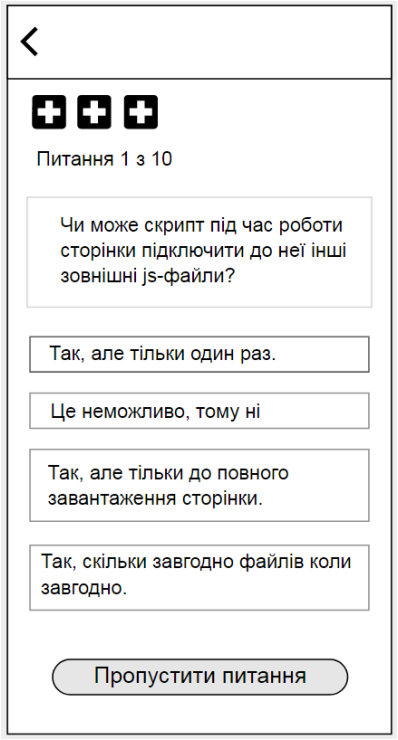


Рисунок 3.4 – Проходження тестування

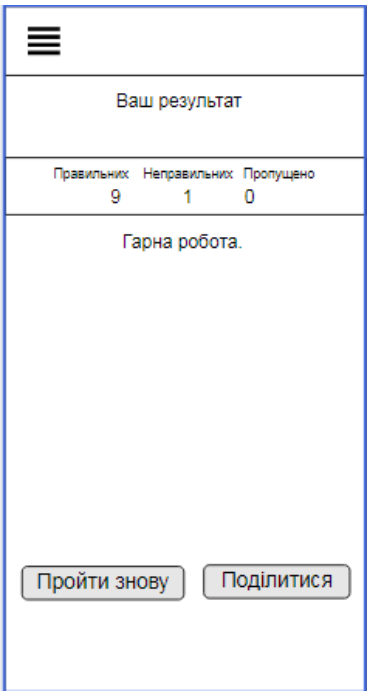


Рисунок 3.5 – Результат тестування

3.3.Архітектура програмного продукту

При створенні додатку було вирішено застосувати “клієнт серверну архітектуру”. “Яка повинна включати ядро додатку, яке містить в собі компоненти системи, не доступні для взаємодії з користувачем. Графічний інтерфейс користувача.”[10] Передача даними об’єктів інтерфейсом “Parsable”. Графічні файли, звуки, необхідні бінарні файли.

Якщо більш детально описувати архітектуру серверна частина працює лише за допомогою Firebase Google системи, що значно зменшує написання кількісті коду та швидкості додатку. Клієнтська частина працює на основі даних json файлу з питаннями, рівнем складностями, правильними відповідями та іншими даними. Обробка даних працює у відповідних класах додатків методами завантаженнями даних з json та їх детальним опрацюванням для того щоб дані були коректно та правильно опрацьовані було створено відповідний адаптер даних, який зв’язує інтерфейс View частину та модель, яка в свою чергу за допомогою інтерфейсу “Parcelable” буде передавати дані між різними класами програмного продукту.

Переваги використання Firebase в цілому:

- Швидка інтеграція до програмного продукту серверна частина Firebase.
- Не потрібна конфігурація на стороні сервера.
- Оновлення в даних реальному часі при підключення додатку до мережі.
- Можливість користуватися сервісом безкоштовно.
- Надійна API для Javascript та платформ iOS та Android.
- Можливість аутентифікації за допомогою вже створених аккаунтів користувача Facebook, Google і Twitter, GitHub.
- Надійність та безпечність даних користувачів бази даних.
- Аналітика власного додатку, залучення нових користувачів
- Запис, читання, та конфігурація даних користувача при використанні бази даних.

- Підключення push повідомлень.
- Монетизація додатку. [11]

Рисунок 3.6 – Архітектура програмного продукту

3.4.Діаграма взаємодії

“Діаграма прецедентів це перелік дій, сценарій по якому користувач взаємодіє з додатком, програмою для виконання будь-якої дії для досягнення конкретної мети.”[12] Варіанти використання було вказано в текстовій та представлені діаграмою прецедентів. Складова сутності моделювання діаграми взаємодії допомагає розробити систему з погляду майбутнього користувача. “За допомогою Use Case може бути описано і призначене для користувача вимога, і вимога до взаємодії систем, і опис взаємодії людей і компаній в реальному житті” [13]

Була створена діаграма прецедентів, яка відображає відносини між акторами і прецедентами і є складовою частиною моделі прецедентів. Діаграма була побудована за допомогою UML та сайту moqups.com(рис.3.7).

Варіанти використання додатком авторизованим користувачам:

- Вибір розділу для проходження тестування
- Вибір рівень складності
- Отримання результат та коментар проходження
- Перехід до теоретичного матеріалу в панелі навігації додатку
- Перехід до вікна про додаток

Варіанти використання додатком не авторизованим користувачам:

- Реєстрація користувача.
- Авторизація користувача.
- Відновлення пароля.

Варіанти використання додатком адміністратором:

- Редагування вмісту самих питань тестів, варіантів відповідей
- Перегляд та редагування даних користувачів додатка

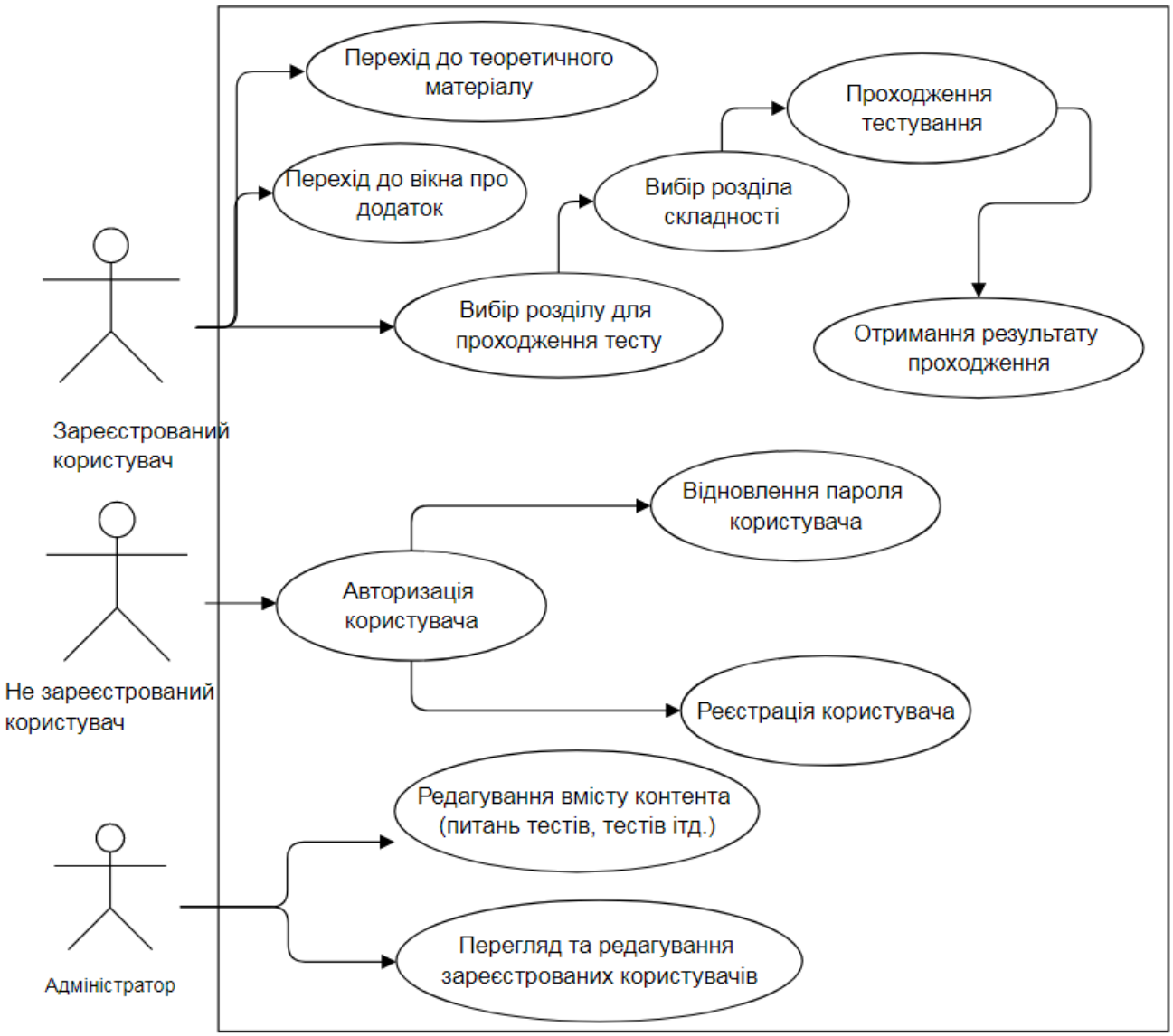


Рисунок 3.7 – Діаграма прецедентів

3.5.Діаграма IDEF0

Для конкретизація процесів була спроектована діаграма IDEF0. Позитивним явищем “методології IDEF0” є ієрархічне представлення об'єктів, що значно полегшує розуміння предметної області. “Така модель є однією з найпрогресивніших моделей і використовується в організації бізнес проектів, що базуються на моделюванні всіх процесів як адміністративних, так і організаційних”[14].

Діаграма IDEF0 представлена на рисунку 3.8. На вхід системи потрапляє запит користувача, який на виході отримує отримання результату проходження відповідного тесту.

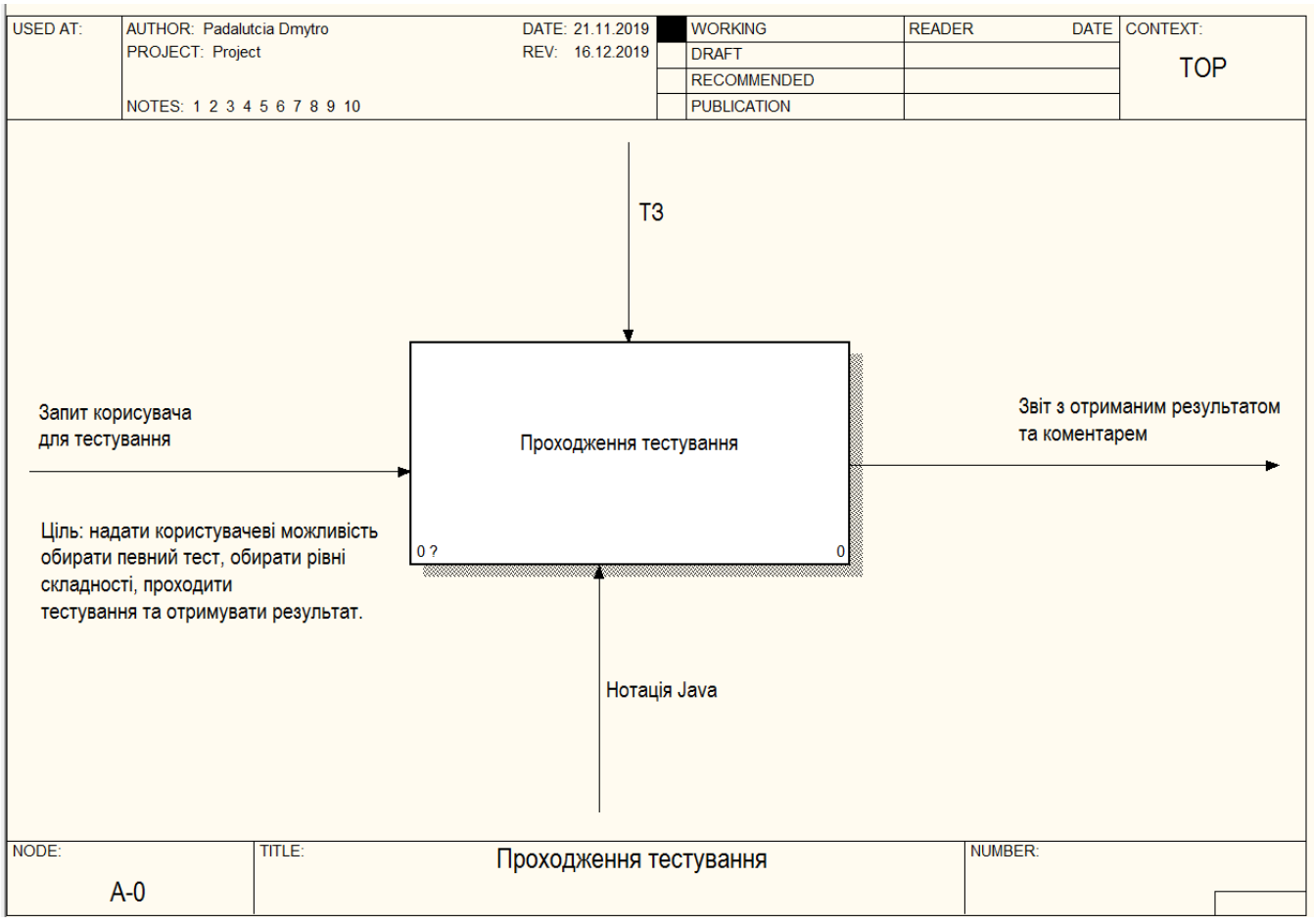


Рисунок 3.8 – Діаграма IDEF0

На діаграмі рисунку 3.9 представлена декомпозиція бізнес-процесів.

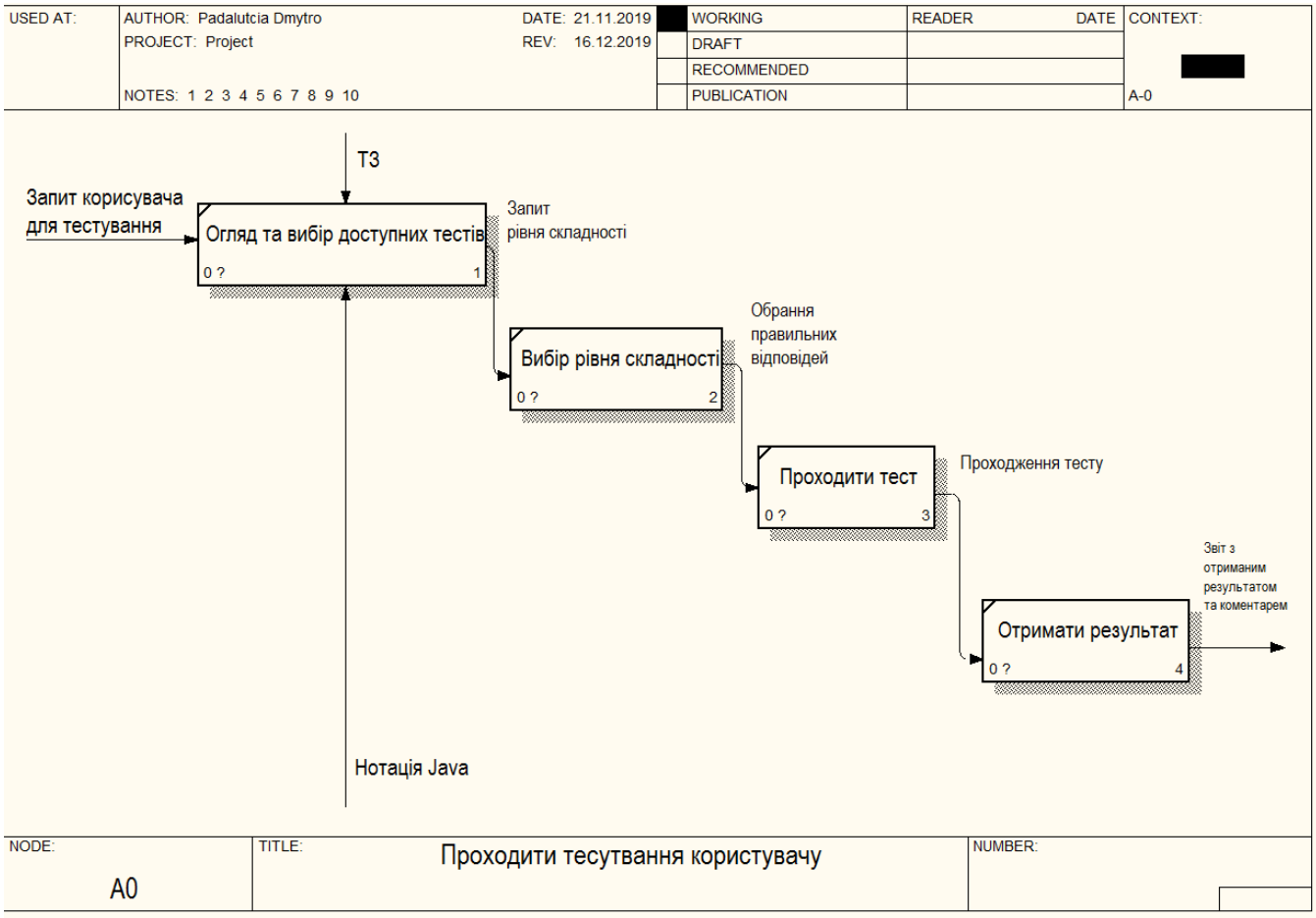


Рисунок 3.9 – Декомпозиція роботи «Проходження тестування»

4. РОЗРОБКА МОБІЛЬНОГО ДОДАТКУ

4.1.Огляд основних можливостей програмного продукту

Розроблюваний мобільний додаток повинен забезпечувати наступний функціонал:

1. **Інтуїтивно зрозумілий інтерфейс**. Було створено заздалегідь макети екранів взаємодії з користувачем та підібрані відповідні стилі для приємного перегляду інформації додатку.
2. **Своєчасна швидкість продукту та відображення актуальної інформації**. Було застосовано сучасні засоби для обслуговування користувача.
3. **Автентифікація користувача в системі.** Всі дані користувача знаходяться у службі Firebase Auth.
4. **Вибір користувачем рівня складності та мови програмування теста.** Всі режими переходів між екранами додатка повинні працювати доцільно.
5. **Проходження тестування.** Користувачеві надається можливість обрати одну правильну відповідь та перейти на інше питання.
6. **Отримання результату проходження тесту.** Фінальний результат виглядає – кількість набраних балів, не правильних балів та пропущених

Програмний продукт створений у вигляді мобільного додатку. Основна перевага мобільних додатків що вони гнучкі до використання і єдине що не обхідно це доступ до Інтернету.

4.2.Розробка дизайну інтерфейсу

В даному розділі було спроектовано UI користувача де було підібрано основні кольори додатка проходження тесту на (рисунок 4.1) всі інші макети розробки було представлено в Додатку А

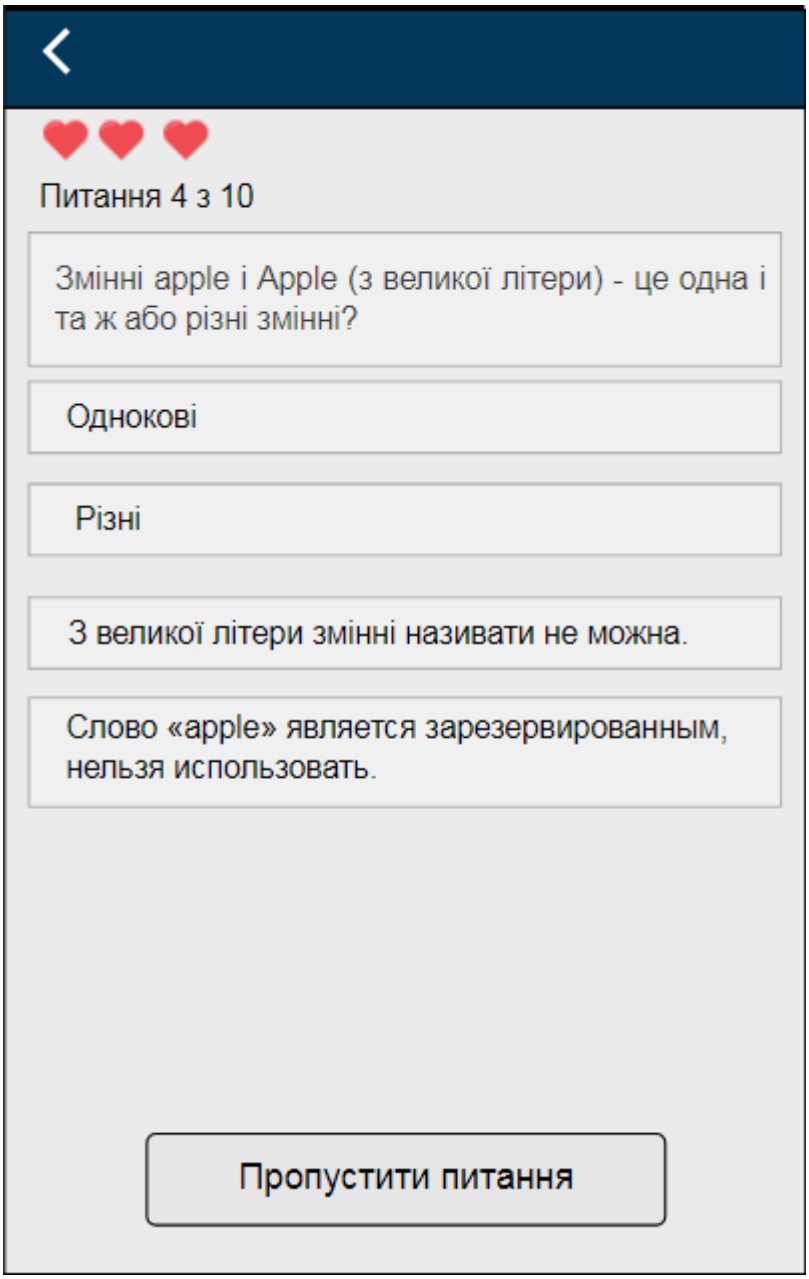


Рисунок 4.1 – Макет екрану проходження тестування

4.3.Опис взаємодії додатка з Firebase

Першим кроком під час створення проекту було вказано версію API 28.1 android 8.1 Далі було створено екран авторизації, реєстрації та відновлення пароля. Як було вказано в архітектурі додатку було використано Firebase систему. С початку потрібно зареєструвати свій проект в firebase.google.com а далі за інструкцією було названо проект та присвоєно відповідне унікальне ім’я далі ми отримаємо відповідний файл від системи Firebase який потрібно додати кореня проекту а там вже підключити firebase до свого проекту за допомогою посилання треба вставити в файл build.grandle(Module app)

implementation 'com.google.firebase:firebase-auth:16.0.5'

В налаштуваннях Firebase було встановлено, що авторизація проходить за допомогою пошти та пароля також в налаштуваннях проекту було змінено мову з англійської на українську для того щоб коли користувач буде відновлювати свій аккаунт на пошту прийде лист з відповідним посиланням для зміни пароля (рис.4.8) Після підключення до проекту Firebase було створено макет xml з авторизацією, реєстрацією та відновлення пароля тоді було створено класи activity а згодом додано до AndroidManifest.xml. В самих класах було створено приватні змінні полів вводу даних пошта та пароль відповідні кнопки btnSignup btnLogin btnReset. Змінна auth відповідає за ключ користувача. В методі onCreate було підключено вже створений макет авторизації та інші елементи за унікальним ідентифікатором. За допомогою auth ми викликаємо функцію signInWithEmailAndPassword з Firebase та реалізуємо умови для входу в систему за допомогою пошти та пароля (рис.4.2). На приклад, якщо пароль занадто короткий користувач отримує відповідне повідомлення.(рис.4.3)

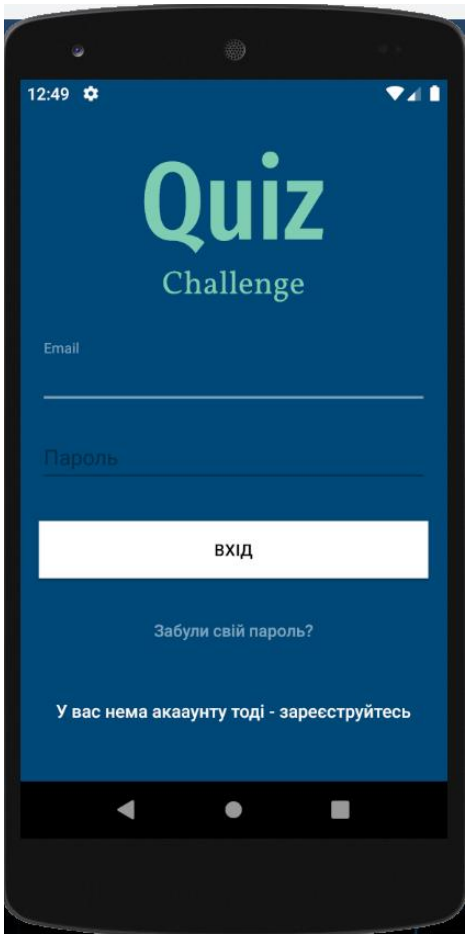


Рисунок 4.2 – Екран зображення авторизації додатку

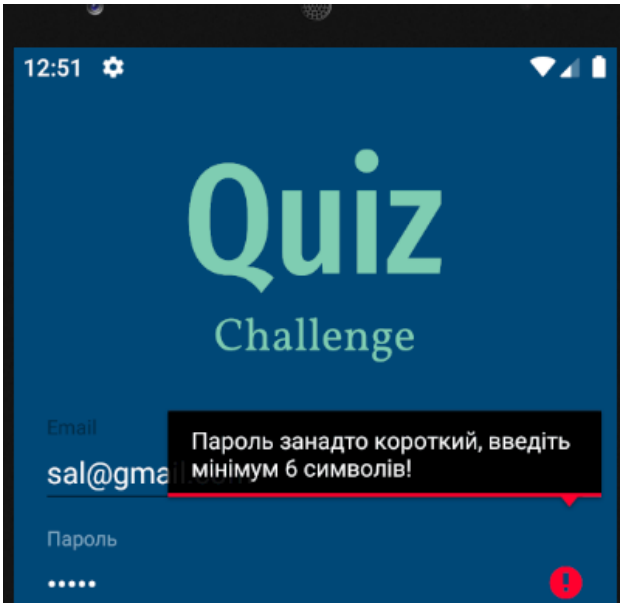


Рисунок 4.3 Екран отримання помилки

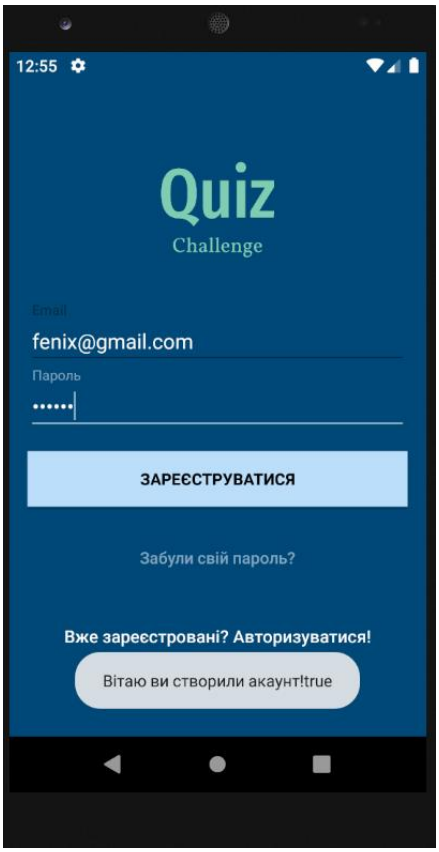


Рисунок 4.4 Екран отримання помилки

Якщо користувач забув власний пароль потрібно вказати лише пошту для відновлення доступа до програми(рис. 4.5)

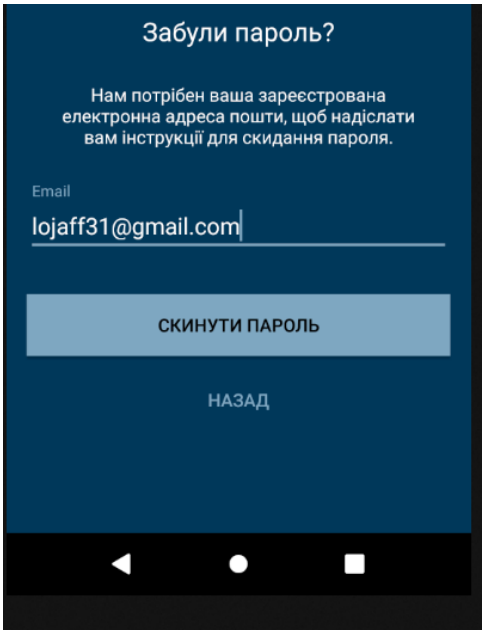


Рисунок 4.5 Екран відновлення доступу до додатку

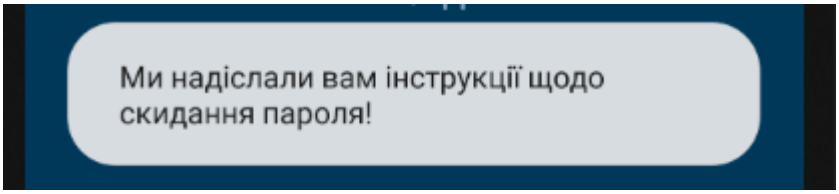


Рисунок 4.6 Екран отримання повідомлення

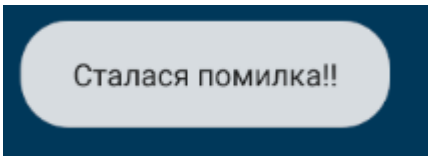


Рисунок 4.7 Екран отримання повідомлення

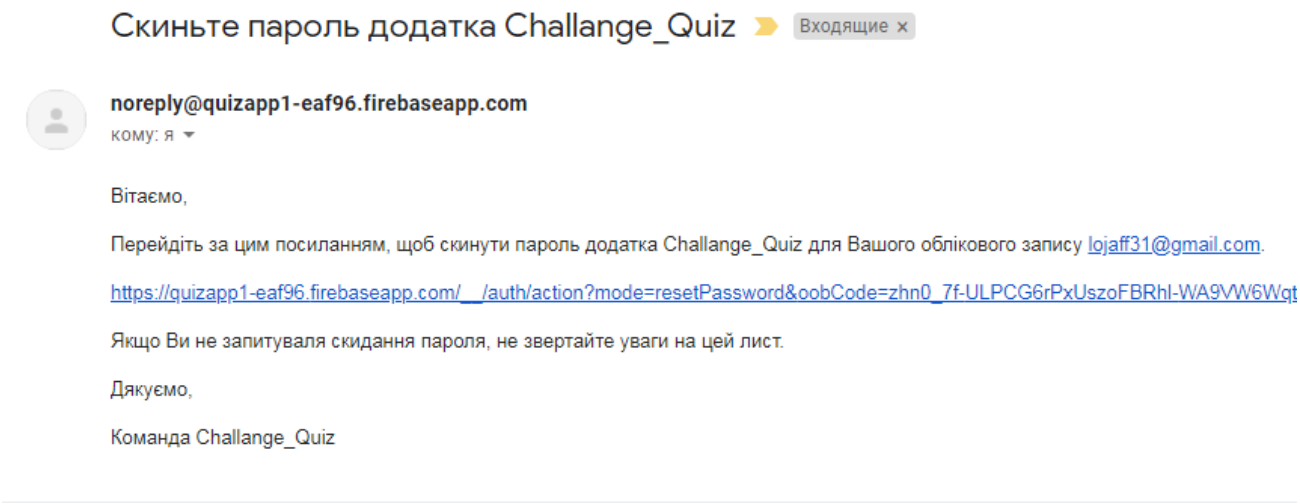


Рисунок 4.8 Екран відновлення доступу

Після отримання листа потрібно перейти за посиланням та відновити профіль створивши новий пароль (рис. 4.9)

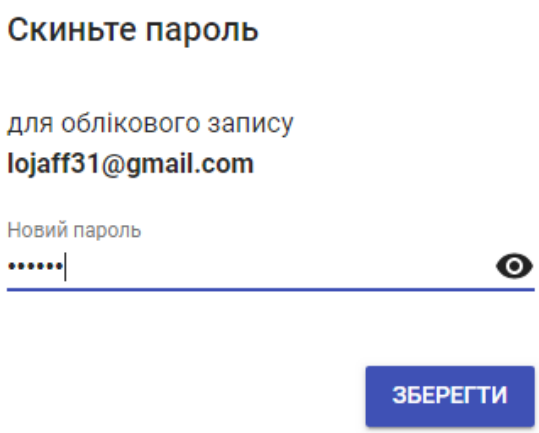


Рисунок 4.9 Створення нового паролю

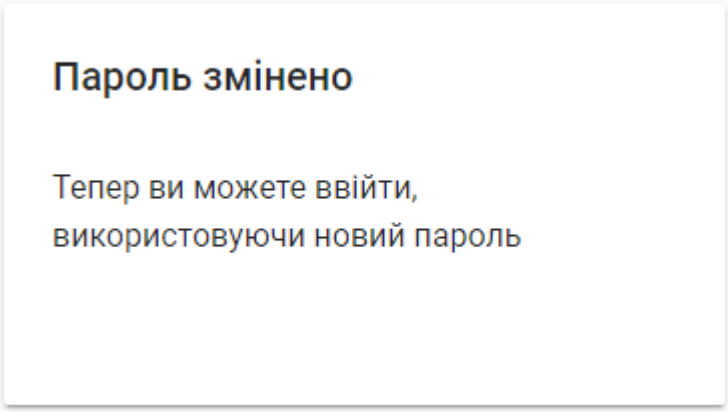


Рисунок 4.10 Відновлення доступу

Етап закінчений вдало на (рис 4.10) показано повідомлення про успішне виконання операції. Тепер користувач може відновити свою профіль та користуватися основним функціоналом додатка. Після проходження етапу аутентифікації користувач потрапляє на головну сторінку (рис.4.11) У вигляді карток виглядають розділи з мов програмування обравши відповідний він потрапляє до екрану обрання рівня складності (рис.4.12)

Рисунок 4.11 Розділи з мов програмування

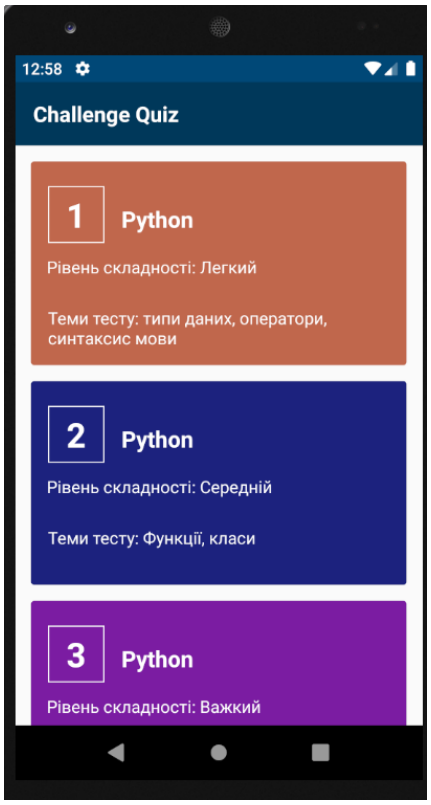


Рисунок 4.12 Вибір рівня складності

Після обрання рівнів складності користувач потрапляє до етапу проходження тестування рис 4.13 В даному випадку бачимо, що вже друге питання та залишилось двоє “життів”

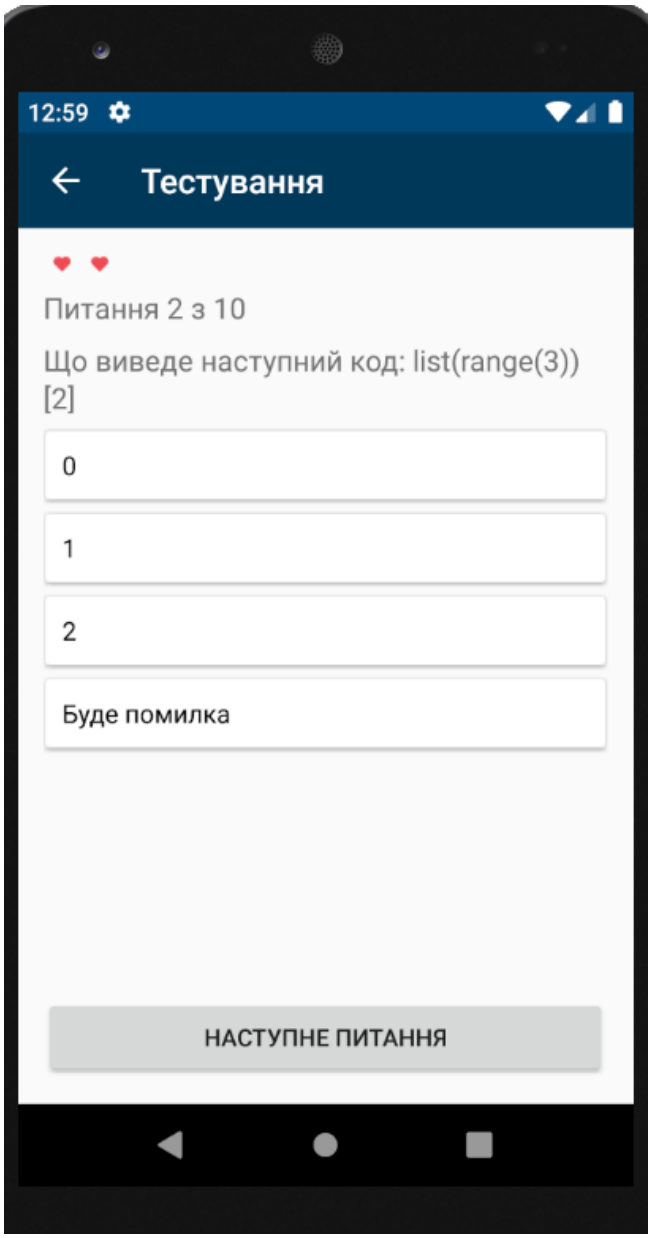


Рисунок 4.13 Вибір рівня складності

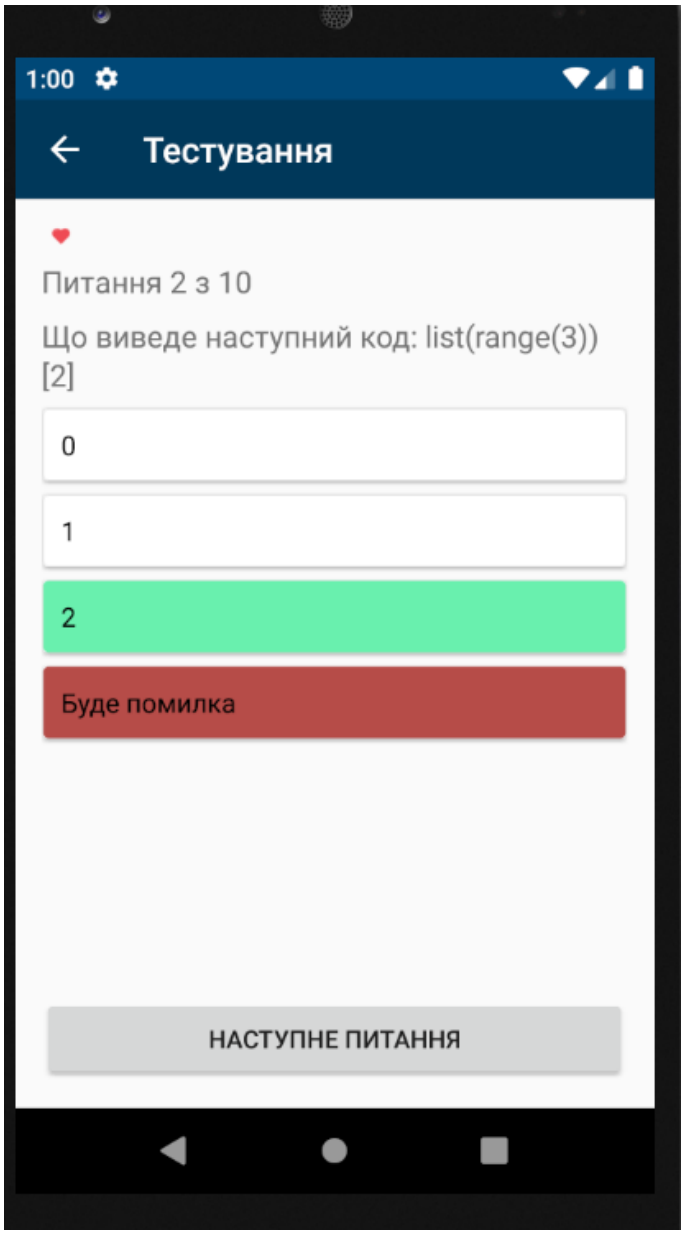


Рисунок 4.14 Обрання не вірного варіанту відповіді

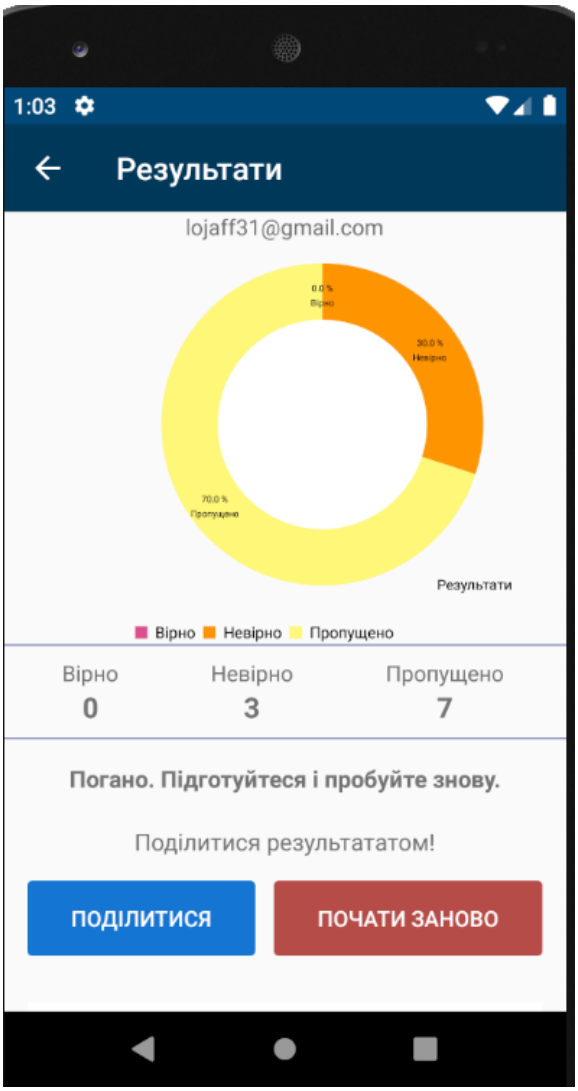


Рисунок 4.15 Отримання результату проходження тестування

На даному вікні спостерігаємо пошту користувача кількість правильних та неправильних відповідей та коментар додатку. Також присутня можливість почати тест заново та на (рис 4.16) показано правильні та неправильні варіанти відповідей які були в тесті

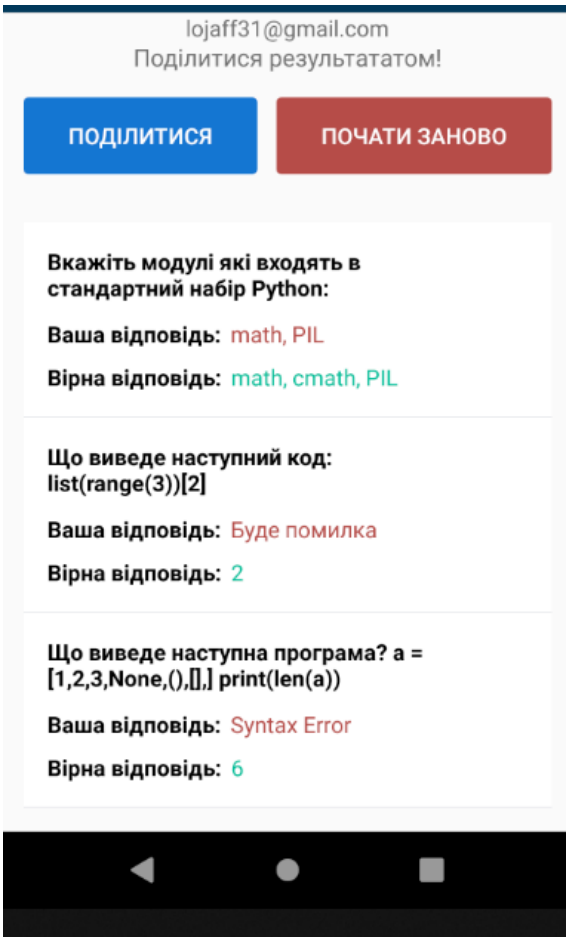


Рисунок 4.16 Отримання результату проходження тестування

5. ТЕСТУВАННЯ

Після закінчення етапу розроблення програмного продукту потрібно протестувати всі можливі варіанти взаємодії користувача з додатком. В таблиці 5.1 продемонстровано детальний опис всіх тестів.

Таблиця 5.1 – Тестування мобільного додатку

TC_ID	Test Scenario/Test Step Name	Actions	ER	Статус
Маємо доступ до інтернету та додаток відкритий на телофоні Nexus 5X				
TC1.1	Реєстрація	1. Заповнити всі поля данними користувача	Перехід до головного екрану	+
TC1.2	Авторизація	1. Заповнити пошту та пароль	Відкривається головноий екран	+
TC1.3	Перевірка всіх розділів тестів	На головному екрані 1. Натиснути на розділ JS т 2. Натиснути на розділ PY	Відкривається Екран рівней складносетей JS Рівні складності Junior Medium Hard Відкривається Екран рівней складносетей PY Рівні складності Junior Medium Hard	+
TC1.4	Перевірити всі екрани складностей	На екрані з JS 1. Перевірити всі екрани складностей 2. Перевірити коректність данних	Унікальні питання для кожного тесту	+

Продовження таблиці 5.1 – Тестування мобільного додатку

ТС1.5	Перевірка Правильності проходження тесту	1. Під час проходження потрібно щоб було встановлено кнопка виходу з тесту 2. Кількість життів повинна бути 3 та при не правильності вибору в тесті зменшуватись на одиницю 3. Питання повинно бути добре видно 4. Варіантів відповідей повинно бути 4 5. Правильна відповідь повинна бути одна 6. Перевірити щоб працювати кнопка пропустити питання	Після натискання кнопки виходу показує вікно для уточнення виходу з тесту. Кількість життів працює коректно. Питання видно відмінно. Після вибору варіанта відповіді показує відповідних колір зелений – правильна відповідь червона не правильна. Після натискнення кнопки пропуску питання з'являється попереджувальне вікно про пропуск тесту	+
ТС1.6	Перевірка результату тесту	Після проходження тесту з'являється вікно результату 1. Натиснути почати тест знову "+" 2. Перевірити правильну кількість балів(набраних очок та не правильних балів, та пропущених)	Після натискання на пропуск тесту з'являється вікно з привітанням та в якому вказано минулу кількість балів у проходженні тестування Всі дані відображені правильно	+

ВИСНОВОК

Результатом виконання кваліфікаційної роботи є працюючий мобільний додаток для тестування знань мов програмування.

Результати аналізу предметної області дозволили підтвердити актуальність роботи. На етапі аналізу порівняння існуючих додатків зі схожим функціональним призначенням були встановленні позитивні риси відомих додатків та риси, яких потрібно уникнути при проектуванні. Дотримання визначених вимог до програмного продукту дозволило розробити конкурентно спроможний мобільний додаток, готовий до впровадження. Вибір методів реалізації був обраний за рекомендаціями від фахівців з провідної компанії Google.

Планування IT – проекту виконано за класичною методикою. Результати планування представлені у формі діаграми Ганта, діаграми WBS, таблицями результатів оцінювання ризиків тощо.

Під час моделювання проекту визначені кінцеві споживачі продукту та сценарій взаємодії користувача з додатком, розглянуті всі можливі сценарії взаємодії користувача з додатком в корисній та розважальній формі.

Для формалізації процесів була розроблена діаграма IDEF0 та декомпозиція процесів, які ілюструють проходження тестування з використанням розробленого додатку. Макет додатку, розроблений засобами сайту moqus.com. Реалізація внутрішнього функціоналу проходження тестування реалізовано методом завантаженням відповідних даних з json далі відповідних класах activity триває обробку отриманих даних правильно опрацьовані було створено відповідний адаптер даних який зв’язує інтерфейс View частину та модель, яка в свою чергу за допомогою інтерфейсу “Parceble” буде передавати дані між різними класами програмного продукту. Аутентифікація користувача виконана за допомогою Firebase.

СПИСОК ЛІТЕРАТУРИ

1. ЦИТАТИ: Вільям Едвардс Демінг Веб-сайт URL http://www.management.com.ua/cit.php?author=%C2%B3%EB%FC%FF%EC%20%C5%E4%E2%E0%F0%E4%F1%20%C4%E5%EC%B3%ED%E3 (дата звернення 04.10.2019)

2. TIOBE Index for December 2019 Веб-сайт URL https://www.tiobe.com/tiobe (дата звернення 04.10.2019)

3. Використовуйте Firebase, єдиний багатоплатформовий SDK від Google, щоб поліпшити додатки веб-сайт URL https://developer.android.com/distribute/best-practices/develop/build-with-firebase?hl=ru (дата звернення 06.10.2019)

4. What is Work Breakdown Structure? Веб-сайт URL https://www.visual-paradigm.com/guide/project-management/what-is-work-breakdown-structure/ (дата звернення 06.10.2019)

5. Organization Breakdown Structure (OBS) Веб-сайт URL https://uplandsoftware.com/psa/resources/glossary/organization-breakdown-structure-obs/ (дата звернення 09.10.2019)

6. Діаграма Ганта для Управління Проектами: Плюси і Мінуси. Веб-сайт URL https://blog.ganttpro.com/ru/gantt-chart-online-project-planning-pros-cons-ru/ (дата звернення 12.10.2019

7. What is a Functional Requirement? Specification, Types, EXAMPLES Веб-сайтURLhttps://www.guru99.com/functional-requirement-specification-example.html (дата звернення 15.10.2019)

8. What is Non-Functional Requirement? Веб-сайт URL https://www.guru99.com/non-functional-requirement-type-example.html (дата звернення 16.10.2019)

9. Якісна оцінка ризиків. Веб-сайт URL http://ru.solverbook.com/spravochnik/menedzhment/kachestvennaya-ocenka-riskov/ (дата звернення 18.10.2019)

10. Архітектура мобільного клієнт-серверного додатка Веб-сайт URL https://habr.com/ru/post/246877/.(дата звернення 20.10.2019)

11. Features of Firebase https://www.androidhive.info/2016/06/android-getting-started-firebase-simple-login-registration-auth/ /.(дата звернення 21.10.2019)

12. Що таке Юзкейс (Use Case) або "Сценарій Використання" в тестуванні ПО? Веб-сайт URL https://software-testing.org/testing/chto-takoe-yuzkeys-use-case-ili-scenariy-ispolzovaniya-v-testirovanii-po.html /.(дата звернення 23.10.2019)

13. USE CASES Що це таке і навіщо вони потрібні? Веб-сайт URL https://systems.education/use-case/ дата звернення (23.10.2019)

14. Методологія IDF0 Веб-сайт URL https://uk.wikipedia.org/wiki/IDEF0 (дата звернення 27.10.2019)

ДОДАТОК А

ТЕХНІЧНЕ ЗАВДАННЯ

на розробку “Мобільний додаток для тестування знань з мов програмування”

Суми 2019

Продовження додатку А

1 Призначення й мета створення мобільного додатку.

1.1. Призначення мобільного додатку

Мобільний додаток призначений для перевірки знань користувачів з декількох мов програмування з різним рівнем складності.

1.2. Мета створення додатку

Допомогти користувачам перевірити свої знання та подивитися свою кількість балів.

1.3.Цільова аудиторія

Аудиторія, яка знайома з програмуванням та мовами JavaScript та Python

2 Вимоги програмного продукту

2.1.Вимоги до програмного продукту загалом

2.1.1. Вимоги до структури й функціонування

Мобільний додаток повинен бути в мінімалістичному стилі та інтуїтивно зрозумілим для користувачів, продукт здатен перевіряти користувачів у вигляді тестування та показувати набранні бали. Кожен тест має 3 “життів” втративши їх користувач закінчується тест з негативною оцінкою.

2.1.2. Вимоги до користувачів та персоналу

Для користуванням додатком потрібен відповідний apk файл, який можна встановити на смартфон на базі Android версією не менше 8.1

Продовження додатку А

2.1.3. Вимоги до подання інформації

Вся основна інформація тестування є онлайн та користувач має змогу проходити тестування тільки з доступом до інтернету

2.1.4. Основні вимоги

2.1.4.1. Структура додатку

Додаток повинен складатися з наступних екранів користувача:

- Екран аутентифікація користувача (авторизація, реєстрація, відновлення пароля).
- Головний екран – користувач обирає складність тесту та МП.
- Екран проходження тестування – користувач читає питання та обирає правильний варіант відповіді після чого переходить на інше питання.
- Екран результату тестування – користувач після закінчення тесту дивиться на кількість правильних, не правильних та пропущених питань. Також має змогу побачити на правильні відповіді на питання, які він відповів не правильно. Та можливість повторно пройти тестування.
- Екран про додаток – основні правила користування додатком та його мета створення.
- Екран тулбар – навігаційне вікно де користувач може обрати посилання на документацію теоретичного матеріалу та інше.

Продовження додатку А

2.1.4.2. Навігація

Інтерфейс додатку повинен бути інтуїтивно зрозумілий і мати досить простий функціонал, розділи додатку повинні бути доступні для користувача. Система повинна забезпечувати навігацію по всіх доступних користувачеві розділам і відображати відповідну інформацію.

2.1.4.3. Система навігації (дизайн головного екрану додатку)

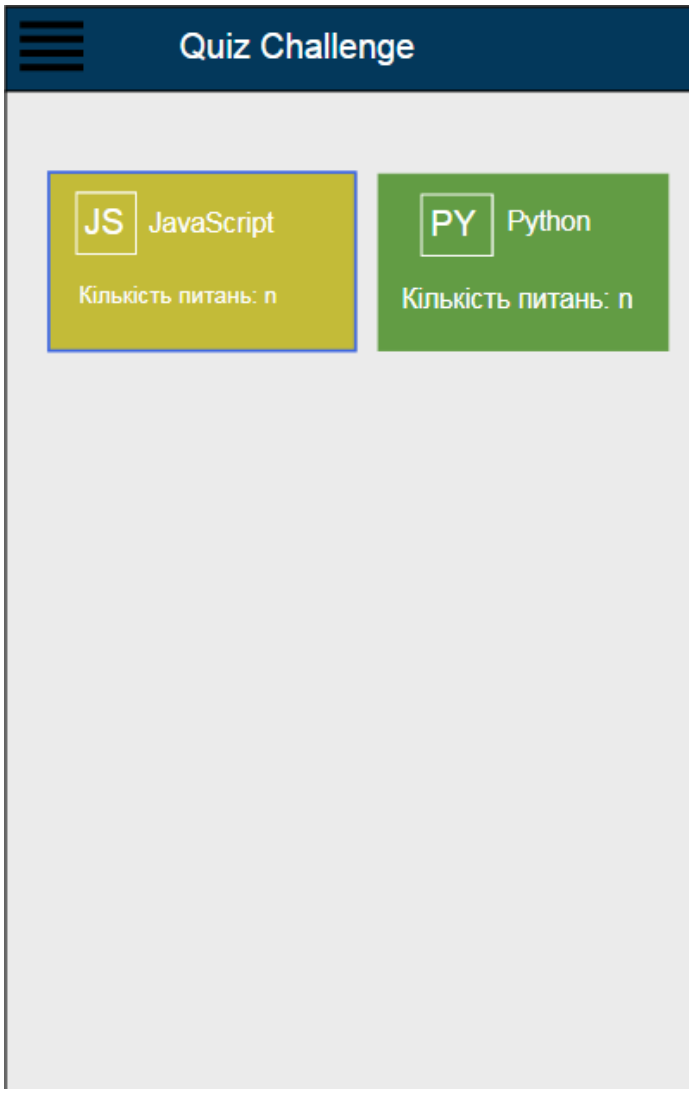


Рисунок А.1 – дизайн головного екрану додатку

Продовження додатку А

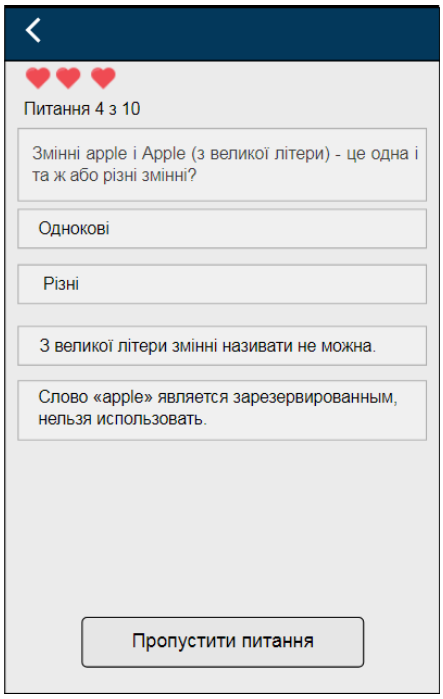


Рисунок А.2 – дизайн проходження тестування екрану додатку

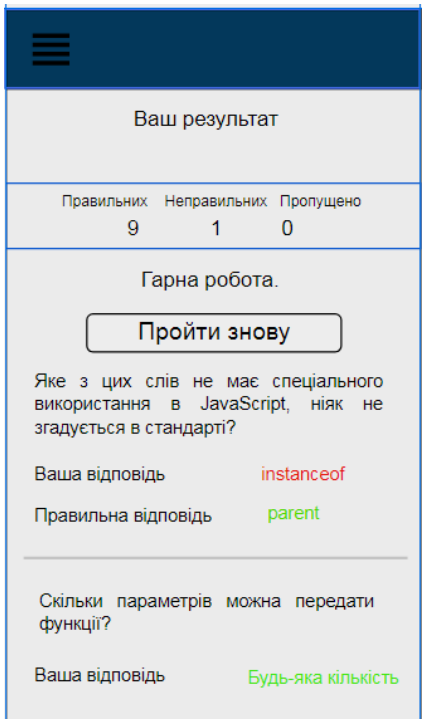


Рисунок А.3 – дизайн результату проходження

Продовження додатку А

2.1.4.4. Типові навігаційні й інформаційні елементи

- Верхня частина додатку(тулбар)
- Основне поле для вибору розділа тестування

2.2.Вимоги до видів забезпечення

2.2.1. Вимоги до інформаційного забезпечення

Реалізація програмного продукту відбувається за допомогою мови програмування Java. В середовищі Android Studio

2.2.2. Вимоги до лінгвістичного забезпечення

Додаток повинен бути виконаний українською мовою.

2.2.3. Вимоги до програмного забезпечення

Програмне забезпечення клієнтської частини повинне задовольняти наступним вимогам:

- Наявність смартфону на базі Android версія не менше 8.1

Продовження додатку А

3 Склад і зміст робіт

Докладний опис етапів роботи зі створення додатку наведено в табл. А1.

Таблиця А1 – Етапи створення сайту

№	Склад і зміст робіт	Строк розробки (у робочих днях)
1	Дослідження предметної області	10 днів
2	Створення дизайн макетів екранів додатка	10 днів
3	Верстка екранів	5 днів
4	Створення функціоналу додатку	20 день
5	Тестування та знаходження несправностей. Виправлення недоліків	10 днів

Для створення умов функціонування, при яких гарантується відповідність створюваного додатку вимогам сьогодення ТЗ і можливість його ефективної роботи, повинен бути проведений певний комплекс заходів.

ДОДАТОК Б

Схема Б.1 – Планування змісту структури мобільного додатку(WBS) для тестування знань з мов програмування

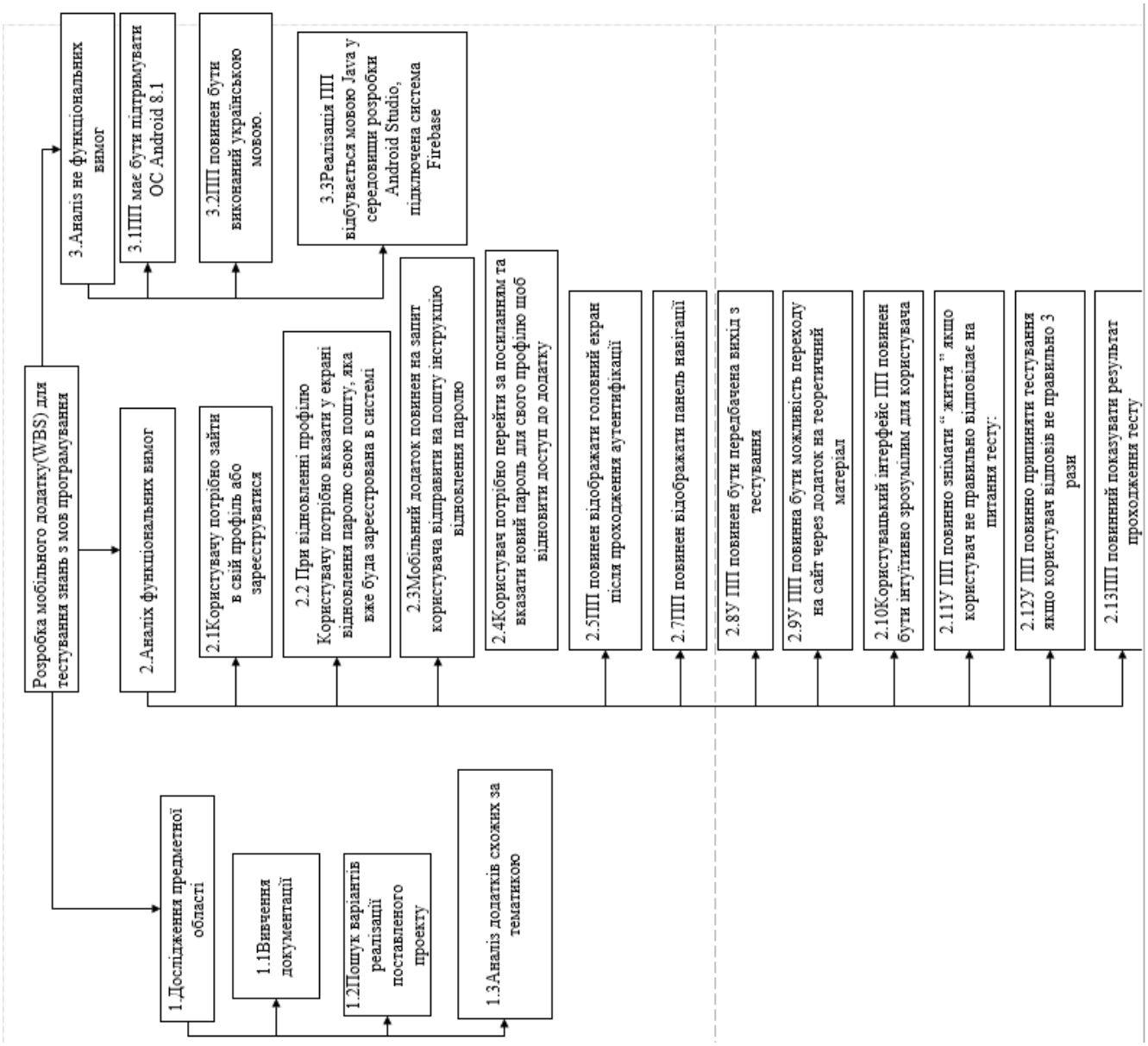
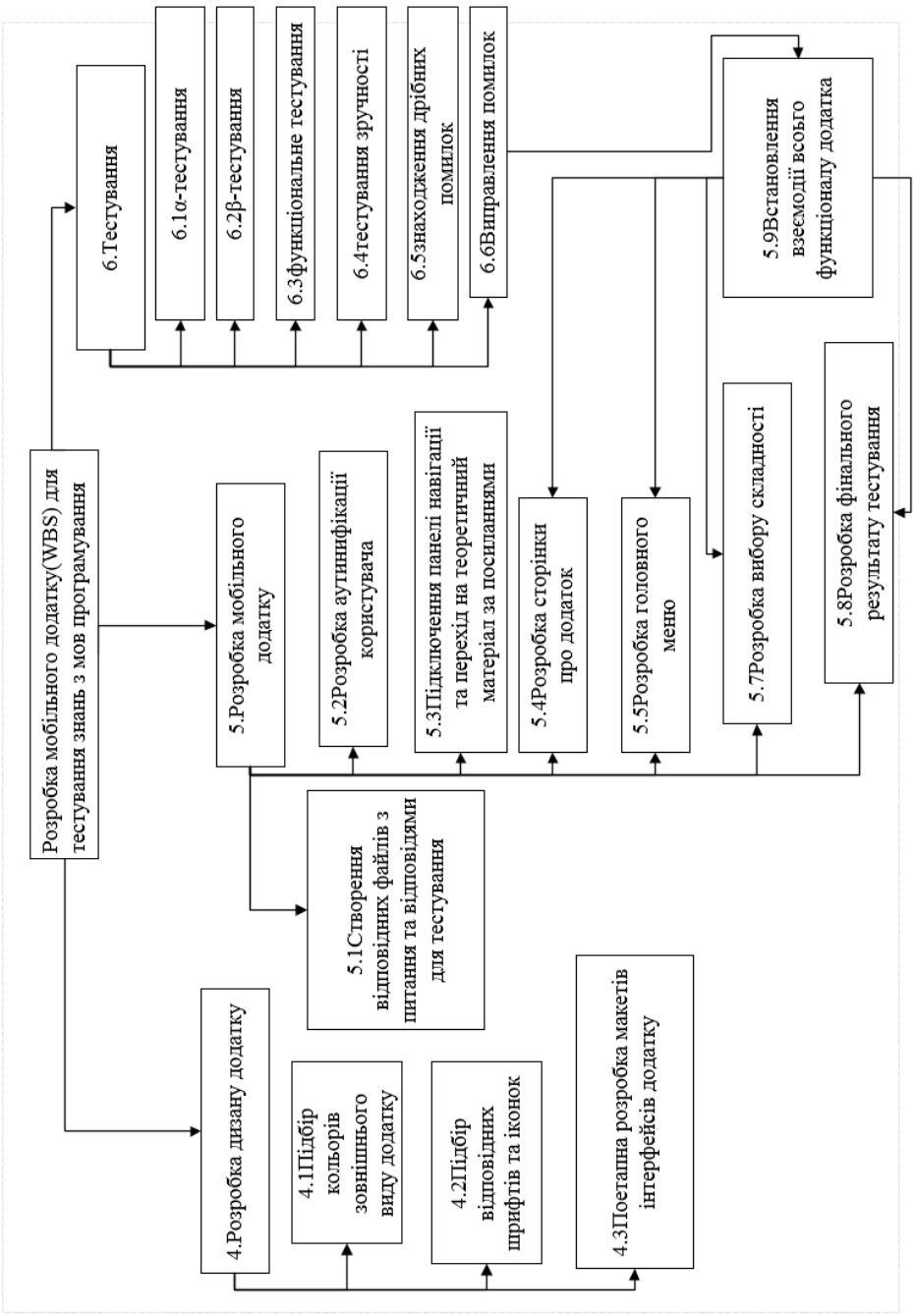


Схема Б.1 - Планування змісту структури мобільного додатку для тестування знань з мов програмування (WBS

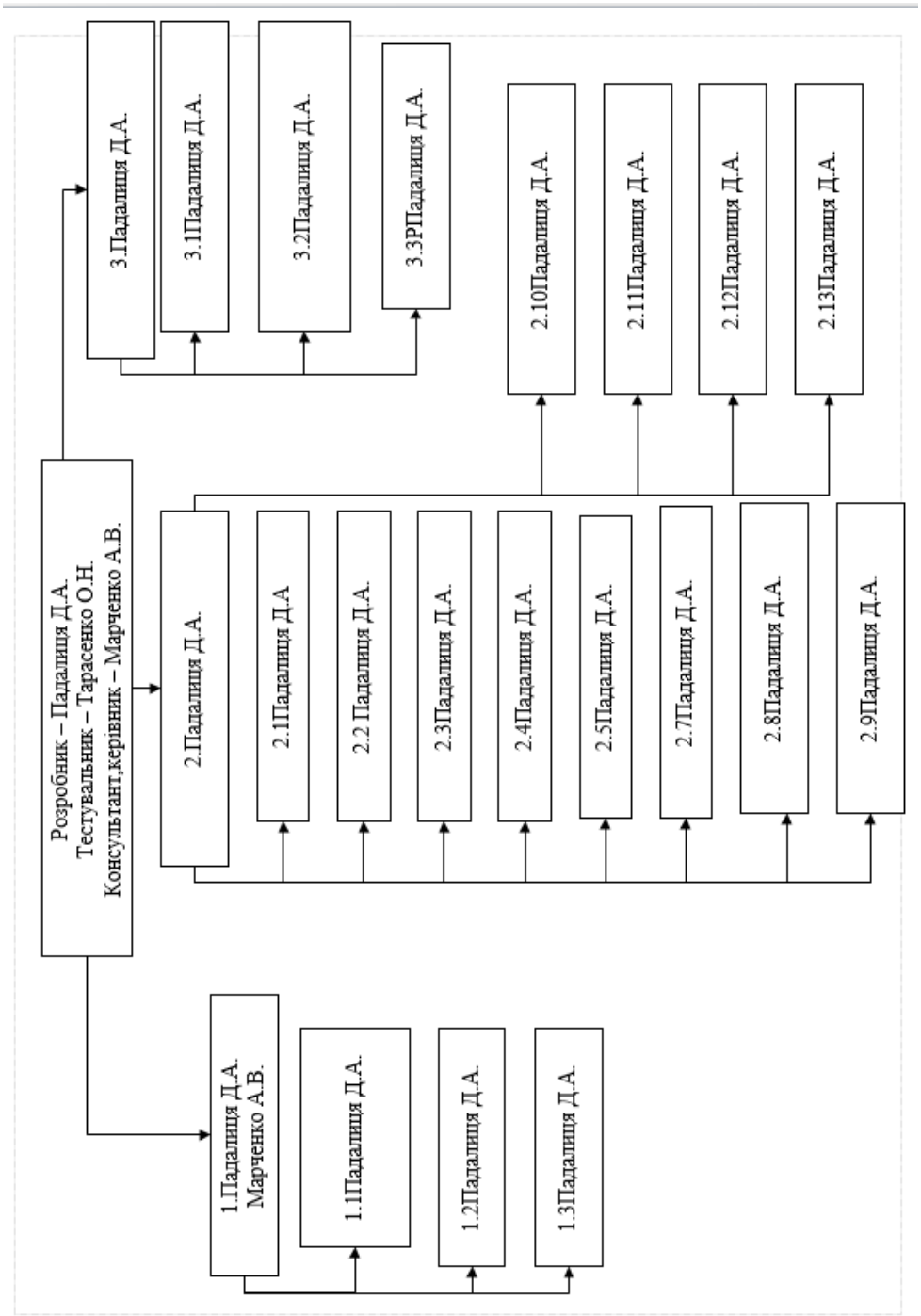
Продовження додатку Б



Продовження схеми Б.1 - Планування змісту структури мобільного додатку для тестування знань з мов програмування (WBS)

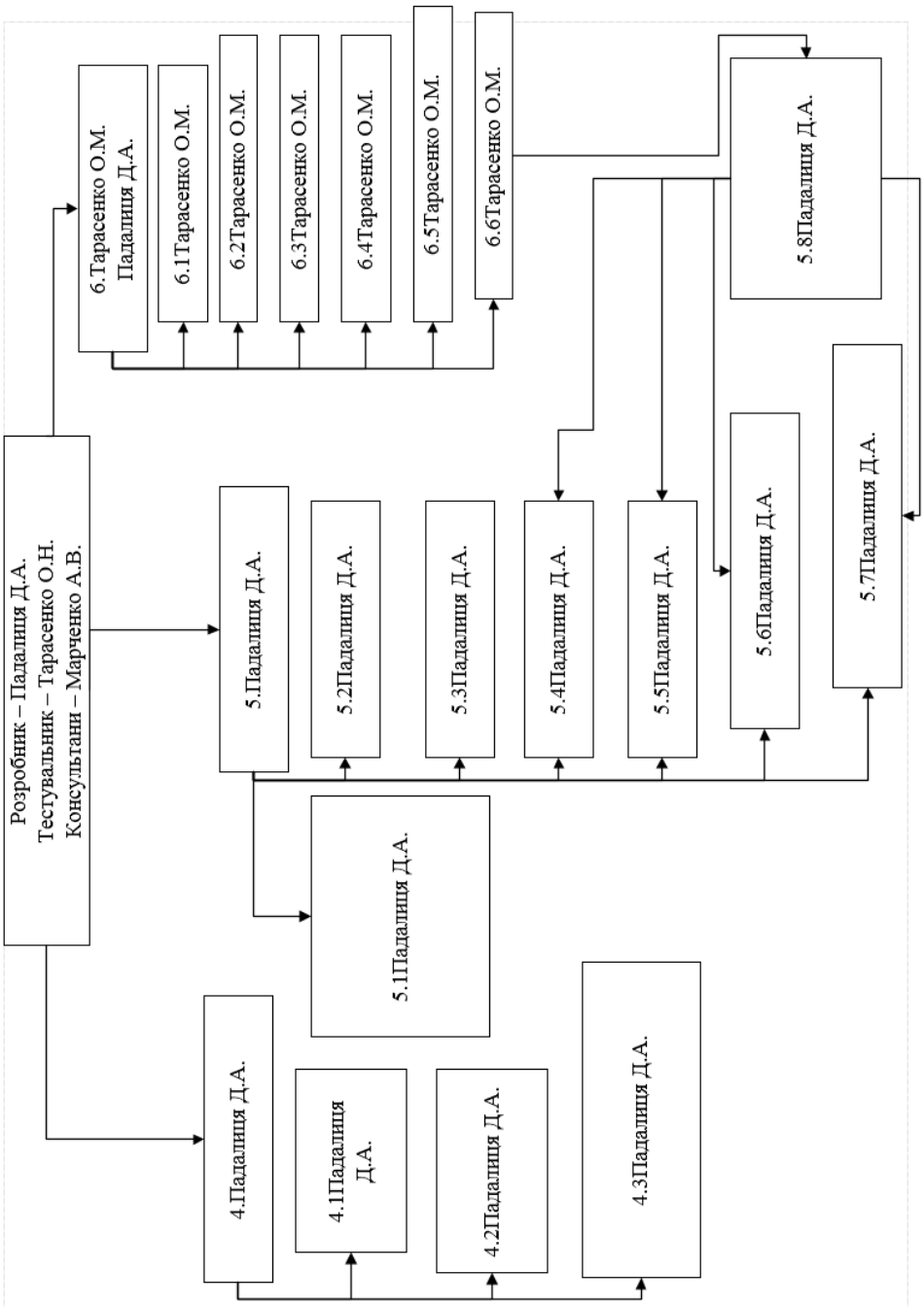
Продовження додатку Б

Схема Б.2 Організаційна структура проектування мобільного додатку для тестування знань з мов програмування (OBS)



Продовження схеми Б.2 Організаційна структура проектування мобільного додатку для тестування знань з мов програмування (OBS)

Продовження додатку Б



Продовження схеми Б.2 Організаційна структура проектування мобільного додатку для тестування знань з мов програмування (OBS)

Продовження додатку Б

Таблиця Б.3 – Матриця відповідальності

	Падалиця Д.А.	Марченко А.В	Тарасенко О.М.
1. Вивчення документації ТЗ	+	+	
1.1Пошук варіантів реалізації поставленого проекту	+		
1.2 Вивчення документації ТЗ	+		
1.3 Ознайомлення з контентом, який буде надано користувачу з додатку	+		
2. Аналіз функціональних вимог	+		
2.1 Користувачу потрібно зайти в свій профіль або зареєструватися	+		
2.2 При відновленні профілю Користувачу потрібно вказати у екрані відновлення паролю свою пошту, яка вже буда зареєстрована в системі	+		
2.3 Мобільний додаток повинен на запит користувача відправити на пошту інструкцію відновлення паролю	+		
2.4 Користувач потрібно перейти за посиланням та вказати новий пароль для свого профілю щоб відновити доступ до додатку	+		
2.5 ПП повинен відображати головний екран після проходження аутентифікації	+		
2.6ПП повинен відображати панель навігації	+		
2.7У ПП повинен бути передбачена вихід з тестування	+		
2.8У ПП повинна бути можливість переходу на сайт через додаток на теоретичний матеріал	+		
2.9Користувацький інтерфейс ПП повинен бути інтуїтивно зрозумілим для користувача	+		

Продовження додатку Б

Продовження таблиці Б.3 – Матриця відповідальності

2.10 У ПП повинно знімати “ життя ” якщо користувач не правильно відповідає на питання тесту:	+		
2.11 У ПП повинно припиняти тестування якщо користувач відповів не правильно 3 рази	+		
2.12ПП повинний показувати результат проходження тесту	+		
3. Аналіз не функціональних вимог	+		
3.1 ПП маж підтримувати ОС Android 8.1	+		
3.2 ПП повинен бути виконаний українською мовою	+		
3.3 Реалізація ПП відбувається Java у середовищі розробки Android Studio	+		
4. Розробка концепції дизайну	+		
4.1 Підбір кольорів зовнішнього виду додатку	+		
4.2 Розробка макетів інтерфейсів додатку	+		
5. Розробка мобільного додатку	+		
5.1 Створення відповідних файлів з питання та відповідями для тестування	+		
5.2 Розробка системи аутентифікації	+		
5.3 Верстка головних екранів	+		
5.5 Підключення панелі навігації та вибір відповідного тесту	+		
5.6. Розробка системи тестування	+		
5.7. Розробка фінального результату тестування	+		
5.8. Встановлення взаємодії всього функціоналу додатка	+		
6. Тестування	+		+
6. 1 α-тестування	+		+
6.2β-тестування	+		+
6.3Тестування зручності	+		+
6.4функціональне тестування	+		+
6.5Знаходження дрібних помилок	+		+
6.6знаходження дрібних помилок	+		+
6.7Виправлення помилок	+		

Продовження додатку Б

Таблиця Б.4 – Ймовірність втрат

Ризик	Ймовірність виникнення	Величина втрат
Не ефективне використання робочим часом	2	3
Втрачання мотивації команди	2	4
Непорозуміння з замовником	3	4
Низька продуктивність роботи	3	3
Недоречне використання ресурсів	3	3
Не доцільно складене ТЗ	2	4
Неефективний розподіл робіт	3	4
Недостатній професіоналізм	3	4
Внесення змін у ТЗ	3	4
Виникнення надзвичайний ситуацій	2	4

Таблиця Б.5 – Класифікація за ступенем впливу та за рівнем ризику

Ризик	Ступінь впливу	Рівень ризику
Не ефективне використання робочим часом	13	виправданні ризики
Втрачання мотивації команди	12	виправданні ризики
Непорозуміння з замовником	18	виправданні ризики
Низька продуктивність роботи	15	виправданні ризики
Недоречне використання ресурсів	12	виправданні ризики

Продовження додатку Б

Продовження таблиці Б.5 – Класифікація за ступенем впливу та за рівнем ризику

Не доцільно складене ТЗ	15	недопустимі ризики
Неефективний розподіл робіт	12	недопустимі ризики
Недостатній професіоналізм	17	недопустимі ризики
Внесення змін у ТЗ	20	виправданні ризики

ДОДАТОК В

MainActivity.java

```
public class MainActivity extends BaseActivity {

    private Activity activity;
    private Context context;
    private Toolbar toolbar;
    private AccountHeader header = null;
    private Drawer drawer = null;
    private ArrayList<CategoryModel> categoryList;
    private CategoryAdapter adapter = null;
    private RecyclerView recyclerView;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        activity = MainActivity.this;
        context = getApplicationContext();
        toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);
        recyclerView = (RecyclerView) findViewById(R.id.rvContent);
        recyclerView.setLayoutManager(new GridLayoutManager(activity, 2, GridLayout
Manager.VERTICAL, false));
        categoryList = new ArrayList<>();
        adapter = new CategoryAdapter(context, activity, categoryList);
        recyclerView.setAdapter(adapter);
        initLoader();
        loadData();
        initListener();
        final IProfile profile = new ProfileDrawerItem().withIcon(R.drawable.ic_lau
ncher_round);
        header = new AccountHeaderBuilder()
                .withActivity(this)
                .withTranslucentStatusBar(true)
                .withHeaderBackground(R.drawable.code)
                .addProfiles(profile)
                .build();

        drawer = new DrawerBuilder()
                .withActivity(this)
                .withToolbar(toolbar)
                .withHasStableIds(true)
                .withAccountHeader(header)
                .addDrawerItems(
                        new PrimaryDrawerItem().withName("Про додаток").withIcon(R.
drawable.ic_launcher_round).withIdentifier(10).withSelectable(false),
                        new SecondaryDrawerItem().withName("YouTube").withIcon(R.dr
awable.ic_youtube).withIdentifier(20).withSelectable(false),
                        new SecondaryDrawerItem().withName("Facebook").withIcon(R.d
rawable.ic_facebook).withIdentifier(21).withSelectable(false),
                        new SecondaryDrawerItem().withName("Теорія JS").withIcon(R.
drawable.ic_idea).withIdentifier(22).withSelectable(false),
                        new SecondaryDrawerItem().withName("Теорія Python").withIco
n(R.drawable.python).withIdentifier(23).withSelectable(false),
                        new DividerDrawerItem()
                                        )
                .withOnDrawerItemClickListener(new Drawer.OnDrawerItemClickListener
() {
                    @Override
                    public boolean onItemClick(View view, int position, IDrawerItem
 drawerItem) {
                        if (drawerItem != null) {
```

Продовження додатку В

```
Intent intent = null;
                            if (drawerItem.getIdentifier() == 10) {
                                ActivityUtilities.getInstance().invokeNewActivity(a
ctivity, AboutDevActivity.class, false);
                            } else if (drawerItem.getIdentifier() == 20) {
                                AppUtilities.youtubeLink(activity);
                            } else if (drawerItem.getIdentifier() == 21) {
                                AppUtilities.facebookLink(activity);
                            }else if (drawerItem.getIdentifier() == 22) {
                                ActivityUtilities.getInstance().invokeCustomUrlActi
vity(activity, CustomUrlActivity.class,
                                        getResources().getString(R.string.site), ge
tResources().getString(R.string.site_url), false);
                            }
                            else if (drawerItem.getIdentifier() == 23) {
                                ActivityUtilities.getInstance().invokeCustomUrlActi
vity(activity, CustomUrlActivity.class,


                                        getResources().getString(R.string.site_pyth
on), getResources().getString(R.string.site_python_url), false);
                            }
                            /* else if (drawerItem.getIdentifier() == 32) {
                                AppUtilities.shareApp(activity);
                            }  */
                            else if (drawerItem.getIdentifier() == 40) {
                            }
                        }
                        return false;
                    }
                })
                .withSavedInstance(savedInstanceState)
                .withShowDrawerOnFirstLaunch(true)
                .withShowDrawerUntilDraggedOpened(true)
                .build();
    }
    @Override
    public void onBackPressed() {
        if (drawer != null && drawer.isDrawerOpen()) {
            drawer.closeDrawer();
        } else {
            AppUtilities.tapPromtToExit(this);
        }
    }
    private void loadData() {
        showLoader();
        loadJson();
    }

    private void loadJson() {
        StringBuffer sb = new StringBuffer();
        BufferedReader br = null;

        try{
            br = new BufferedReader(new InputStreamReader(getAssets().open(AppConst
ants.CONTENT_FILE_MAIN )));
            String temp;
            while ((temp = br.readLine()) != null)
                sb.append(temp);
        } catch (IOException e) {
```

Продовження додатку В

```
e.printStackTrace();
        } finally {
            try {
                br.close();
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
        parseJson(sb.toString());
    }

    private void parseJson(String jsonData) {
        try {
            JSONObject jsonObject = new JSONObject(jsonData);
            JSONArray jsonArray = jsonObject.getJSONArray(AppConstants.JSON_KEY_ITE
MS);

 for (int i = 0; i < jsonArray.length(); i++) {

                JSONObject object = jsonArray.getJSONObject(i);
                String categoryId = object.getString(AppConstants.JSON_KEY_CATEGORY
 ID);
                String categoryName = object.getString(AppConstants.JSON_KEY_CATEGO
RY_NAME);
                String lengthIems = object.getString(AppConstants.JSON_KEY_CATEGORY
_LENGTH);
                String level = object.getString(AppConstants.JSON_KEY_CATEGORY_LEN_
LEVEL_JUN);
                String level1 = object.getString(AppConstants.JSON_KEY_CATEGORY_LEN
_LEVEL_MID);
                String level2 = object.getString(AppConstants.JSON_KEY_CATEGORY_LEN
_LEVEL_HAD);
                categoryList.add(new CategoryModel(categoryId, categoryName, length
Iems, level, level1, level2));
            }
        } catch (JSONException e) {
            e.printStackTrace();
        }
        hideLoader();
        adapter.notifyDataSetChanged();
    }
    private void initListener() {
        adapter.setItemClickListener(new ListItemClickListener() {
            @Override
            public void onItemClick(int position, View view) {
                CategoryModel model = categoryList.get(position);
                ActivityUtilities.getInstance().invokeCommonQuizActivity(activity,
LevelActivity.class, model.getCategoryName(),  true);
            }
        });
    }
}
```

ActivityUtilities.java

```
public class ActivityUtilities {
    private static ActivityUtilities activityUtilities = null;

    public static ActivityUtilities getInstance() {
        if (activityUtilities == null) {
            activityUtilities = new ActivityUtilities();
        }
```

Продовження додатку В

```
        return activityUtilities;
    }
    public void invokeNewActivity(Activity activity, Class<?> tClass, boolean shoul
gFinish) {
        Intent intent = new Intent(activity, tClass);
        activity.startActivity(intent);
        if (shoulgFinish) {
            activity.finish();
        }
    }
    public void invokeCustomUrlActivity(Activity activity, Class<?> tClass, String
pageTitle, String pageUrl, boolean shouldFinish) {
        Intent intent = new Intent(activity, tClass);
        intent.putExtra(AppConstants.BUNDLE_KEY_TITLE, pageTitle);
        intent.putExtra(AppConstants.BUNDLE_KEY_URL, pageUrl);
        activity.startActivity(intent);
        if (shouldFinish) {
            activity.finish();
        }
    }
    public void invokeCommonQuizActivity(Activity activity, Class<?> tClass, String
 categoryId, boolean shouldFinish) {
        Intent intent = new Intent(activity, tClass);
        intent.putExtra(AppConstants.BUNDLE_KEY_INDEX, categoryId);
        activity.startActivity(intent);
        if (shouldFinish) {
            activity.finish();
        }

}
    public void invokeScoreCardActivity(Activity activity, Class<?> tClass, int que
stionsCount, int score, int wrongAns, int skip, String categoryId, ArrayList<Result
Model> resultList, boolean shouldFinish) {

        Intent intent = new Intent(activity, tClass);
        intent.putExtra(AppConstants.BUNDLE_KEY_SCORE, score);
        intent.putExtra(AppConstants.QUESTIONS_IN_TEST, questionsCount);
        intent.putExtra(AppConstants.BUNDLE_KEY_WRONG_ANS, wrongAns);
        intent.putExtra(AppConstants.BUNDLE_KEY_SKIP, skip);
        intent.putExtra(AppConstants.BUNDLE_KEY_INDEX, categoryId);
        intent.putParcelableArrayListExtra(AppConstants.BUNDLE_KEY_ITEM, resultList
);
        activity.startActivity(intent);
        if (shouldFinish) {
            activity.finish();
        }
    }
```

LoginAcitvity.java

```
public class LoginActivity extends AppCompatActivity {
    private EditText inputEmail, inputPassword;
    private FirebaseAuth auth;
    private ProgressBar progressBar;
    private Button btnSignup, btnLogin, btnReset;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        auth = FirebaseAuth.getInstance();
```

Продовження додатку В

```
        setContentView(R.layout.activity_login);

        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);

        inputEmail = (EditText) findViewById(R.id.email);
        inputPassword = (EditText) findViewById(R.id.password);
        progressBar = (ProgressBar) findViewById(R.id.progressBar);
        btnSignup = (Button) findViewById(R.id.btn_signup);
        btnLogin = (Button) findViewById(R.id.btn_login);
        btnReset = (Button) findViewById(R.id.btn_reset_password);
        auth = FirebaseAuth.getInstance();
        btnSignup.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                startActivity(new Intent(LoginActivity.this, SignupActivity.class))
;
            }
        });
        btnReset.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                startActivity(new Intent(LoginActivity.this, ResetPasswordActivity.
class));
            }
        });
        btnLogin.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String email = inputEmail.getText().toString();
                final String password = inputPassword.getText().toString();

                if (TextUtils.isEmpty(email)) {
                    Toast.makeText(getApplicationContext(), "Введіть електронну адр
есу!", Toast.LENGTH_SHORT).show();
                    return;
                }

                if (TextUtils.isEmpty(password)) {

Toast.makeText(getApplicationContext(), "Введіть пароль", Toast.LENGTH_SHORT).show(
);
                    return;
                }
                progressBar.setVisibility(View.VISIBLE);
                auth.signInWithEmailAndPassword(email, password)
                        .addOnCompleteListener(LoginActivity.this, new OnCompleteLi
stener<AuthResult>() {
                            @Override
                            public void onComplete(@NonNull Task<AuthResult> task)
{
                                progressBar.setVisibility(View.GONE);
                                if (!task.isSuccessful()) {
                                    if (password.length() < 6) {
                                        inputPassword.setError(getString(R.string.m
inimum_password));
                                    } else {
                                        Toast.makeText(LoginActivity.this, getStrin
g(R.string.auth_failed), Toast.LENGTH_LONG).show();
                                    }
                                } else {
```

Продовження додатку В

```
                                Intent intent = new Intent(LoginActivity.this, MainActi
vity.class);
                                        startActivity(intent);
                                        finish();
                                    }
                                }
                            });
                }
            });
    }
}
```

CategoryAdapter.java

```
public class CategoryAdapter extends RecyclerView.Adapter<CategoryAdapter.ViewHolde
r> {
    private Context mContext;
    private Activity mActivity;
    private ArrayList<CategoryModel> categoryList;
    private ListItemClickListener itemClickListener;

    public CategoryAdapter(Context mContext, Activity mActivity, ArrayList<Category
Model> categoryList) {
        this.mContext = mContext;
        this.mActivity = mActivity;
        this.categoryList = categoryList;
    }
    public void setItemClickListener(ListItemClickListener itemClickListener) {
        this.itemClickListener = itemClickListener;
    }
    @NonNull
    @Override
    public ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
        View view = LayoutInflater.from(parent.getContext())
                .inflate(R.layout.item_category_recycler, parent, false);
        return new ViewHolder(view, viewType, itemClickListener);
    }
    @Override
    public void onBindViewHolder(@NonNull CategoryAdapter.ViewHolder holder, int po
sition) {
        final CategoryModel model = categoryList.get(position);
        String categoryName = model.getCategoryName();
        String lenItems = model.getLenItems();
        String nameID = model.getCategoryId();
        String level = model.getLevel();
        String level1 = model.getLevel1();
        String level2 = model.getLevel2();
        holder.tvCategoryTitle.setText(Html.fromHtml(categoryName));
        holder.tvLenCategory.setText(Html.fromHtml(lenItems));
        holder.tvCategoryId.setText(Html.fromHtml(nameID));
        holder.tvCategoryLevel.setText(Html.fromHtml(level));


holder.tvCategoryLevel1.setText(Html.fromHtml(level1));
        holder.tvCategoryLevel2.setText(Html.fromHtml(level2));
        holder.tvCategoryId.setText(Html.fromHtml(nameID));
switch (position) {
            case 0:
                holder.lytContainer.setBackground(ContextCompat.getDrawable(mContex
t, R.drawable.rectangle_yellow));
                break;
```

Продовження додатку В

```
            case 1:
                holder.lytContainer.setBackground(ContextCompat.getDrawable(mContex
t, R.drawable.rectangle_green));
                break;
            case 2:
                holder.lytContainer.setBackground(ContextCompat.getDrawable(mContex
t, R.drawable.rectangle_blue));
                break;
            case 3:
                holder.lytContainer.setBackground(ContextCompat.getDrawable(mContex
t, R.drawable.rectangle_orange));
                break;
            case 4:
                holder.lytContainer.setBackground(ContextCompat.getDrawable(mContex
t, R.drawable.rectangle_red));
                break;
            case 5:
                holder.lytContainer.setBackground(ContextCompat.getDrawable(mContex
t, R.drawable.rectangle_purple));
                break;
        }
    }
    @Override
    public int getItemCount() {
        return (null != categoryList ? categoryList.size() : 0);
    }
    public static class ViewHolder extends RecyclerView.ViewHolder implements View.
OnClickListener {
        private RelativeLayout lytContainer;
        private TextView tvCategoryTitle, tvCategoryId, tvLenCategory, tvCategoryLe
vel, tvCategoryLevel1, tvCategoryLevel2;
        private ListItemClickListener itemClickListener;
        public ViewHolder(View itemView, int viewType, ListItemClickListener itemCl
ickListener) {
            super(itemView);
            this.itemClickListener = itemClickListener;
            lytContainer = (RelativeLayout) itemView.findViewById(R.id.lytContainer
);
            tvCategoryId = (TextView) itemView.findViewById(R.id.categoryId);
            tvCategoryTitle = (TextView) itemView.findViewById(R.id.titleText);
            tvLenCategory = (TextView) itemView.findViewById(R.id.lenItems);
            tvCategoryLevel = (TextView) itemView.findViewById(R.id.level);
            tvCategoryLevel1 = (TextView) itemView.findViewById(R.id.level1);
            tvCategoryLevel2 = (TextView) itemView.findViewById(R.id.level2);
            lytContainer.setOnClickListener(this);
        }
        @Override
        public void onClick(View view) {
            if (itemClickListener != null) {
                itemClickListener.onItemClick(getLayoutPosition(), view);
            }
        }
    }
}
public class CategoryModel implements Parcelable {
    String categoryId;
    String categoryName;
    String lenItems;
    String level;
    String level1;
    String level2;
```

Продовження додатку В

```
    public CategoryModel(String categoryId, String categoryName, String lenItems, S
tring level, String level1, String level2) {
        this.categoryId = categoryId;
        this.categoryName = categoryName;
        this.lenItems = lenItems;


this.level = level;
        this.level1 = level1;
        this.level2 = level2;
    }

    public String getCategoryId() {
        return categoryId;
    }

    public String getCategoryName() { return categoryName; }

    public String getLenItems() {
        return lenItems;
    }
    public String getLevel() {
        return level;
    }

    public String getLevel1() {
        return level1;
    }
    public String getLevel2() {
        return level2;
    }
    @Override
    public int describeContents() {
        return 0;
    }
    @Override
    public void writeToParcel(Parcel dest, int flags) {

        dest.writeString(categoryId);
        dest.writeString(categoryName);
        dest.writeString(lenItems);
        dest.writeString(level);
        dest.writeString(level1);
dest.writeString(level2;
    public CategoryModel(Parcel in) {
        categoryId = in.readString();
        categoryName = in.readString();
        lenItems = in.readString();
        level = in.readString();
        level1 = in.readString();
        level2 = in.readString();
    }
    public static Creator<CategoryModel> getCreator() {
        return CREATOR;
    }
    public static final Creator<CategoryModel> CREATOR = new Creator<CategoryModel>
() {
        @Override
        public CategoryModel createFromParcel(Parcel source) {
            return new CategoryModel(source);
```

Продовження додатку В

```
        }
        @Override
        public CategoryModel[] newArray(int size) {
            return new CategoryModel[size];
        }
    };
}
```

ScoreCardActivity.java

```
public class ScoreCardActivity extends BaseActivity implements OnChartValueSelected
Listener {
    private Activity mActivity;
    private Context mContext;
    private Button mBtnShare, mBtnPlayAgain;
    private TextView mScoreTextView, mWrongAnsTextView, mSkipTextView, mGreetingTex
tView, mUserEmail;
    private int mScore, mWrongAns, mSkip;
    private int mQuestionsCount;
private String mCategoryId;
    private ArrayList<ResultModel> mResultList;
    private ResultAdapter mAdapter = null;
    private RecyclerView mRecyclerResult;
    private PieChart mPieChart;
    private FirebaseDatabase database;
    private DatabaseReference myRef;
    private FirebaseAuth auth;
    public FirebaseUser getCurrentUser;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        initVar();
        initView();
        initFunctionality();
        initListener();
    }
    private void initVar() {
        mActivity = ScoreCardActivity.this;
        mContext = mActivity.getApplicationContext();
        database = FirebaseDatabase.getInstance();
        myRef = database.getReference("items");
        auth = FirebaseAuth.getInstance();
        Intent intent = getIntent();
        if (intent != null) {
            mScore = intent.getIntExtra(AppConstants.BUNDLE_KEY_SCORE, 0);
            mWrongAns = intent.getIntExtra(AppConstants.BUNDLE_KEY_WRONG_ANS, 0);
            mQuestionsCount = intent.getIntExtra(AppConstants.QUESTIONS IN TEST, 0;
            mCategoryId = intent.getStringExtra(AppConstants.BUNDLE_KEY_INDEX);
            mResultList = intent.getParcelableArrayListExtra(AppConstants.BUNDLE_KE
Y_ITEM);
        }
    }
    private void initView() {
        setContentView(R.layout.activity_score_card);
        mRecyclerResult = (RecyclerView) findViewById(R.id.rvContentScore);
        mRecyclerResult.setLayoutManager(new LinearLayoutManager(this, LinearLayout
Manager.VERTICAL, false));
        mBtnShare = (Button) findViewById(R.id.btn_share);
        mBtnPlayAgain = (Button) findViewById(R.id.btn_play_again);
        mScoreTextView = (TextView) findViewById(R.id.txt_score);
        mWrongAnsTextView = (TextView) findViewById(R.id.txt_wrong);
        mSkipTextView = (TextView) findViewById(R.id.txt_skip);
```

Продовження додатку В

```
        mGreetingTextView = (TextView) findViewById(R.id.greeting_text);
        mUserEmail = (TextView) findViewById(R.id.user_email);
        initToolbar(true);
        setToolbarTitle(getResources().getString(R.string.score_card));
        enableUpButton();
    }
    public void initFunctionality() {
        mSkip = mQuestionsCount - (mScore + mWrongAns);
        mScoreTextView.setText(String.valueOf(mScore));
        mWrongAnsTextView.setText(String.valueOf(mWrongAns));
        mSkipTextView.setText(String.valueOf(mSkip));
        mUserEmail.setText(String.valueOf(auth.getCurrentUser().getEmail()));
        float actualScore = ((float) mScore / (float) (mScore + mWrongAns + mSkip))
 * AppConstants.MULTIPLIER_GRADE;
        switch (Math.round(actualScore)) {
            case 10:
                mGreetingTextView.setText(Html.fromHtml(getResources().getString(R.
string.greeting_text3)));
                break;
            case 9:
            case 8:
            case 7:
            case 6:
            case 5:
                mGreetingTextView.setText(Html.fromHtml(getResources().getString(R.
string.greeting_text2)));
                break;
            default:
                mGreetingTextView.setText(Html.fromHtml(getResources().getString(R.
string.greeting_text1)));
                break;
        }
        showPieChart();
        mAdapter = new ResultAdapter(mContext, mActivity, mResultList);
        mRecyclerResult.setAdapter(mAdapter);
    }

    public void initListener() {
        });
        mBtnPlayAgain.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                ActivityUtilities.getInstance().invokeCommonQuizActivity(mActivity,
 QuizPromptActivity.class, mCategoryId, true);

}
        });
    }

    public void showPieChart() {
        mPieChart = (PieChart) findViewById(R.id.piechart);
        mPieChart.setUsePercentValues(true);
        mPieChart.setDrawHoleEnabled(true);
        mPieChart.setTransparentCircleRadius(AppConstants.TRANSPARENT_CIRCLE_RADIUS
);
        mPieChart.setHoleRadius(AppConstants.TRANSPARENT_CIRCLE_RADIUS);
        mPieChart.setDescription(getString(R.string.score_card));
        mPieChart.animateXY(AppConstants.ANIMATION_VALUE, AppConstants.ANIMATION_VA
LUE);
        ArrayList<Entry> yvalues = new ArrayList<Entry>();
        yvalues.add(new Entry(mScore, AppConstants.BUNDLE_KEY_ZERO_INDEX));
```

Продовження додатку В

```
        yvalues.add(new Entry(mWrongAns, AppConstants.BUNDLE_KEY_SECOND_INDEX));
        yvalues.add(new Entry(mSkip, AppConstants.BUNDLE_KEY_FIRST_INDEX));
        PieDataSet dataSet = new PieDataSet(yvalues, AppConstants.EMPTY_STRING);
        dataSet.setColors(ColorTemplate.JOYFUL_COLORS);
        ArrayList<String> xVals = new ArrayList<String>();
        xVals.add(getString(R.string.score));
        xVals.add(getString(R.string.wrong));
        xVals.add(getString(R.string.skipped));
        PieData data = new PieData(xVals, dataSet);
        data.setValueFormatter(new PercentFormatter());
        mPieChart.setData(data);
    }
    @Override
    public void onValueSelected(Entry e, int dataSetIndex, Highlight h) {
        if (e == null)
            return;
    }
    @Override
    public void onNothingSelected() {
    }
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case android.R.id.home:
                ActivityUtilities.getInstance().invokeNewActivity(mActivity, MainAc
tivity.class, true);
                return true;
        }
        return super.onOptionsItemSelected(item);
    }
    @Override
    public void onBackPressed() {
        ActivityUtilities.getInstance().invokeNewActivity(mActivity, MainActivity.c
lass, true);
    }
}
```

RezultModel.java

```
public class ResultModel implements Parcelable {
    private String question;
    private String givenAns;
    private String correctAns;
    private boolean isCorrect;
    private boolean isSkip;
    public ResultModel(String question, String givenAns, String correctAns, boolean
 isCorrect, boolean isSkip) {
        this.question = question;
        this.givenAns = givenAns;
        this.correctAns = correctAns;
        this.isCorrect = isCorrect;
        this.isSkip = isSkip;
    }
    public String getQuestion() {
        return question;
    }
    public String getGivenAns() {
        return givenAns;
    }
    public String getCorrectAns() {
        return correctAns;
    }
```

Продовження додатку В

```
    public boolean isCorrect() {
        return isCorrect;
    }
    public boolean isSkip() {
        return isSkip;
    }
    @Override
    public int describeContents() {
        return 0;
    }
    @Override
    public void writeToParcel(Parcel dest, int flags) {
        dest.writeString(question);
        dest.writeString(givenAns);
        dest.writeString(correctAns);
        dest.writeByte((byte) (isCorrect ? 1 : 0));
        dest.writeByte((byte) (isSkip ? 1 : 0));
    }
    protected ResultModel(Parcel in) {
        question = in.readString();
        givenAns = in.readString();
        correctAns = in.readString();
        isCorrect = in.readByte() != 0;
        isSkip = in.readByte() != 0;
    }
    public static Creator<ResultModel> getCREATOR() {
        return CREATOR;
    }
    public static final Creator<ResultModel> CREATOR = new Creator<ResultModel>() {
        @Override
        public ResultModel createFromParcel(Parcel source) {
            return new ResultModel(source);
        }
        @Override
        public ResultModel[] newArray(int size) {
            return new ResultModel[size];
        }
    };
}
```

RezultAdapter.java

```
public class ResultAdapter extends RecyclerView.Adapter<ResultAdapter.ViewHolder>{
    private Context mContext;
    private Activity mActivity;
    private ArrayList<ResultModel> mItemList;
    private ListItemClickListener mItemClickListener;
    public ResultAdapter(Context mContext, Activity mActivity, ArrayList<ResultMode
l> mItemList) {
        this.mContext = mContext;
        this.mActivity = mActivity;
        this.mItemList = mItemList;
    }
    public void setItemClickListener(ListItemClickListener itemClickListener) {
        this.mItemClickListener = itemClickListener;
    }
    @Override
    public ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        View view = LayoutInflater.from(parent.getContext()).inflate(R.layout.item_
result, parent, false);
        return new ViewHolder(view, viewType, mItemClickListener);
```

Продовження додатку В

```
    }
    public static class ViewHolder extends RecyclerView.ViewHolder implements View.
OnClickListener {
        private ImageView imgAns;
        private TextView tvQuestion, tvGivenAns, tvCorrectAns;
        private RelativeLayout lytAnsContainer;
        private ListItemClickListener itemClickListener;
        public ViewHolder(View itemView, int viewType, ListItemClickListener itemCl
ickListener) {
            super(itemView);
            this.itemClickListener = itemClickListener;
            // Find all views ids
            imgAns = (ImageView) itemView.findViewById(R.id.ans_icon);
            tvQuestion = (TextView) itemView.findViewById(R.id.question_text);
            tvGivenAns = (TextView) itemView.findViewById(R.id.given_ans_text);
            tvCorrectAns = (TextView) itemView.findViewById(R.id.correct_ans_text);
            lytAnsContainer = (RelativeLayout) itemView.findViewById(R.id.your_ans_
container);
            itemView.setOnClickListener(this);
        }
        @Override
        public void onClick(View view) {
            if (itemClickListener != null) {
                itemClickListener.onItemClick(getLayoutPosition(), view);
            }
        }
    }
    @Override
    public int getItemCount() {
        return (null != mItemList ? mItemList.size() : 0);
    }
    @Override
    public void onBindViewHolder(ViewHolder mainHolder, int position) {
        final ResultModel model = mItemList.get(position);
        mainHolder.tvQuestion.setText(Html.fromHtml(model.getQuestion()));
        mainHolder.tvCorrectAns.setText(Html.fromHtml(model.getCorrectAns()));
        if (model.isCorrect()) {
            mainHolder.lytAnsContainer.setVisibility(View.GONE);
        } else {
            mainHolder.tvGivenAns.setText(Html.fromHtml(model.getGivenAns()));
        }
        int imgPosition;
        if (model.isSkip()) {
            imgPosition = AppConstants.BUNDLE_KEY_ZERO_INDEX;
        } else if (model.isCorrect()) {
            imgPosition = AppConstants.BUNDLE_KEY_FIRST_INDEX;
        } else {
            imgPosition = AppConstants.BUNDLE_KEY_SECOND_INDEX;
        }

        Glide.with(mContext)
                .load(mContext.getResources().getIdentifier(AppConstants.DIRECTORY
+ imgPosition, null, mContext.getPackageName()))
                .into(mainHolder.imgAns);
    }
}
```