Вінницький Національний Технічний Університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра програмного забезпечення

Бакалаврська дипломна робота

на тему: Розробка мобільного застосунку для тестування в ігровій формі знань з правил дорожнього руху

Виконав: студент IV курсу, групи 2ПІ-20б

спеціальності

121 – Інженерія програмного забезпечення
(шифр і назва напряму підготовки, спеціальності)

Кривов'язюк Максим Юрійович
(прізвище та ініціали)

Керівник: к.т.н., доц. каф. ПЗ Ракитянська Г.Б.
(прізвище та ініціали)

Рецензент: к.т.н., доц. каф. ОТ.Городецька О.С.
(прізвище та ініціали)

Допущено до захисту

Зав. кафедри _____

«_10__» _червня_____ 2024 р.

ВНТУ – 2024

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Рівень вищої освіти перший (бакалаврський)
Галузь знань 12 – «Інформаційні технології»
Спеціальність 121 – «Інженерія програмного забезпечення»
Освітньо-професійна програма – «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
д.т.н., професор
О. Н. Романюк _____
«12» березня 2024 р.

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Кривов`язюку Максиму Юрійовичу

1. Тема роботи – розробка мобільного застосунку для тестування в ігровій формі знань з правил дорожнього руху.

Керівник роботи: Ракитянська Ганна Борисівна, к.т.н., доц. каф. ПЗ, затверджені наказом вищого навчального закладу від 11 березня 2024 р. № 80.

2. Строк подання студентом роботи 3 червня 2024.

3. Вихідні дані до роботи: відповіді з бази даних застосунку та збереження результатів проходження вікторин; вихідні дані для обчислення результату проходження вікторин.

4. Зміст текстової частини: вступ; аналіз стану освітніх застосунків; аналіз аналогів; аналіз методів розв'язання задачі; постановка задачі; розробка методів створення вікторин; розробка метод визначення результату проходження вікторин; розробка гри як винагороди за правильно пройдені вікторини; розробка алгоритмів роботи програми; розробка програмного забезпечення; розробка інтерфейсу програми; тестування розробленого програмного забезпечення, створення інструкції користувача.

5. Перелік ілюстративного матеріалу: аналоги програмного застосунку, основні етапи створення вікторин; алгоритми роботи застосунку; нові методи створення вікторин; інтерфейс програмного забезпечення.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата			
		Завдання видав		Завдання прийняв	
1-4	Ракитянська Г.Б. к.т.н., доц. каф. ПЗ	13.03.24		03.06.24	

7. Дата видачі завдання 13.03.24

КАЛЕНДАРНИЙ ПЛАН

№ п\п	Назва етапів бакалаврської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз стану розвитку навчальних вікторин для перевірки знань з правил дорожнього руху та постановка задач дослідження.	14.03.24 – 22.03.24	Виконано
2	Розробка модуль для створення вікторин	13.03.24 – 22.03.24	Виконано
3	Розробка алгоритмів роботи програмного забезпечення	25.03.24 – 19.04.24	Виконано
4	Розробка програмного забезпечення	22.04.24 – 10.05.24	Виконано
5	Тестування програмного забезпечення	13.05.24 – 24.05.24	Виконано
6	Оформлення матеріалів до захисту БДР	27.05.24 – 30.05.24	Виконано

Студент _____ **М. Ю. Кривов`язюк**
(підпис) (ініціали та прізвище)

Керівник бакалаврської дипломної роботи _____ **Г. Б. Ракитянська**
(підпис) (ініціали та прізвище)

АНОТАЦІЯ

УДК 004.912.032.26

Кривов`язюк М. Ю. Розробка мобільного застосунку для тестування в ігровій формі знань з правил дорожнього руху.: бакалаврська дипломна робота зі спеціальності 121 – інженерія програмного забезпечення, освітня програма – інженерія програмного забезпечення. Вінниця: ВНТУ, 2024. 59 с.

Укр. мовою. Бібліогр. : 30 назв ; рис. : 46; табл. 3.

У бакалаврській дипломній роботі проведено детальний аналіз методів тестування знань, визначено ефективність гейміфікації та проаналізовано аналоги ігор-вікторин. На основі цього аналізу було встановлено доцільність розробки власної вікторини.

Була розроблена архітектура гри-вікторини та виконаний аналіз різних класів. Запропоновано алгоритми нарахування балів гравцю, підготовки питання, збереження результатів тестування та створення алгоритму відкриття гри користувачу, після набору потрібної кількості балів, що сприяло підвищенню ефективності та мотивації студентів під час тестування знань.

Для створення застосунку було вибрано середовище розробки Android Studio та мову програмування Java.

Для перевірки правильності роботи застосунку були написані та виконані різні тести, які дозволили підтвердити його відповідність заданим вимогам та функціональності. Також, були визначені системні вимоги до операційної системи Android.

Отримані в бакалаврській дипломній роботі результати можна використати для ефективної перевірки знань за допомогою застосунку.

Ключові слова: мобільна гра, вікторина, гейміфікація, тестування знань, правила дорожнього руху.

ABSTRACT

UDC 004.912.032.26

Kryvov`yazyuk M.Yu. Development of a mobile application for testing knowledge of traffic rules in a game form.: bachelor's thesis on specialty 121 - software engineering, educational program - software engineering. Vinnytsia: VNTU, 2024. 59 p. (to applications)

Ukrainian language Bibliogr. : 30 titles; Fig. : 46; table 3.

In the bachelor's thesis, a detailed analysis of knowledge testing methods was carried out, the effectiveness of gamification was determined, and analogues of quiz games were analyzed. Based on this analysis, the expediency of developing one's own quiz was established.

The architecture of the quiz game was developed and the analysis of different classes was performed. Algorithms for assigning points to the player, preparing questions, saving test results, and creating an algorithm for opening the game to the user after collecting the required number of points were proposed, which contributed to increasing the efficiency and motivation of students during knowledge testing.

The Android Studio development environment and the Java programming language were chosen to create the application.

To verify the correctness of the application, various tests were written and performed, which made it possible to confirm its compliance with the specified requirements and functionality. Also, the system requirements for the Android operating system were determined.

The results obtained in the bachelor's thesis can be used to effectively test knowledge using the application.

Keywords: mobile game, quiz, gamification, knowledge testing, traffic rules.

ЗМІСТ

ВСТУП

Обґрунтування вибору теми дослідження. У сучасному освітньому процесі навчання та оцінювання знань є надзвичайно важливими аспектами, традиційні методи тестування, такі як письмові іспити або усні виступи, часто сприймаються студентами як рутинні та нудні, що може призводити до низької мотивації та зацікавленості. Останнім часом гейміфікація, тобто використання ігрових елементів у неігрових контекстах, набирає все більшої популярності як ефективний метод залучення та мотивації учнів до навчання.

Гейміфікація застосовує принципи гри, щоб створити привабливе середовище, в якому користувачі можуть отримувати нагороди та досягати певних цілей. Це сприяє активнішому залученню студентів, підсилює їхній дух суперництва та прагнення досягати кращих результатів.

Основними причинами вибору теми дослідження є:

1. Необхідність підвищення ефективності навчання. Традиційні методи перевірки знань часто сприймаються студентами як монотонні та нудні, що призводить до зниження мотивації та інтересу. Використання гейміфікації в освітньому процесі допомагає зробити навчання більш захоплюючим і ефективним.

2. Зростаюча популярність гейміфікації. За останні роки гейміфікація набула значного поширення у різних сферах, включаючи освіту. Вона застосовує принципи гри для створення привабливого навчального середовища, що сприяє активному залученню студентів і стимулює їх досягати кращих результатів.

3. Потреба в інноваційних підходах до навчання правил дорожнього руху. Знання правил дорожнього руху є критично важливими для безпеки на дорогах. Розробка мобільного застосунку для тестування знань з правил дорожнього руху в ігровій формі дозволить ефективніше навчати і перевіряти знання користувачів, роблячи цей процес більш цікавим та мотивуючим.

Таким чином, розробка мобільного застосунку для тестування знань з правил дорожнього руху в ігровій формі є актуальною і перспективною темою дослідження, що відповідає сучасним викликам у сфері освіти та безпеки на дорогах.

Зв'язок роботи з науковими програмами, планами, темами. Робота виконувалася згідно плану виконання наукових досліджень на кафедрі програмного забезпечення.

Мета та завдання дослідження. Метою дослідження є підвищення ефективності освітнього процесу під час тестування знань шляхом створення гри-вікторини.

Відповідно до поставленої мети в бакалаврській дипломній роботі потрібно вирішити такі **завдання**:

– Аналіз стану розвитку навчальних вікторин для перевірки знань з правил дорожнього руху та постановка задач дослідження;

– розробити програмний модуль для створення вікторин;

– розробити алгоритм нарахування балів гравцю;

– розробити алгоритм відкриття ігор для гравця;

– розробити графічний інтерфейс користувача для створюваного продукту;

– розробити програмний застосунок для гри-вікторини для перевірки знань з правил дорожнього руху на основі створеної архітектури та алгоритмів; провести ручне тестування програмного застосунку з використанням комплексних сценаріїв виконання.

Об'єкт дослідження – процес розробки програмного застосунку для перевірки правил дорожнього руху у вигляді гри вікторини.

Предмет дослідження – алгоритми і засоби реалізації програмного застосунку для перевірки правил дорожнього руху у вигляді гри вікторини.

Методи дослідження. У процесі дослідження використовувались: комплексний аналіз з метою визначення недоліків існуючих технічних рішень задачі та подальший синтез отриманих даних для формування нових

функціональних характеристик і вимог; теорія алгоритмів для розробки та вдосконалення алгоритмів програмного забезпечення вікторини.

Новизна отриманих результатів.

1. Удосконалено алгоритм нарахування балів гравцям, особливість якого полягає в наданні бонусних балів, що забезпечує проходження порогового рівня знань: якщо правильних балів було більше за неправильні, то різниця зберігається, що дозволяє не тільки стимулювати активну участь гравців, але й при наборі потрібної кількості балів буде доступною гра-вікторина.

2 Удосконалено програмну модель системи, що забезпечує вивчення користувачем теоретичного матеріалу в двох варіантах: онлайн, тобто через відео у YouTube та офлайн через матеріали, що надаються у PDF форматі

Практична цінність отриманих результатів. Практичне значення отриманих результатів полягає у тому, що на основі теоретичних положень, представлених у бакалаврській дипломній роботі, було розроблено алгоритми та програмні засоби, які спрямовані на підвищення ефективності освітнього процесу та мотивації студентів під час тестування знань з правил дорожнього руху.

Особистий внесок здобувача. Усі наукові результати, що викладені у бакалаврській дипломній роботі, отримано автором самостійно. У наукових працях, опублікованих у співавторстві, автору належать такі результати: розробка застосунку для перевірки знань з правил дорожнього руху у вигляді гри-вікторини [3].

Апробація матеріалів бакалаврської дипломної роботи. Результати роботи доповідалися на міжнародній науково-практичній інтернет-конференції " Молодь в науці: дослідження, проблеми, перспективи " (Вінниця, 2024 р.);

Публікації. За тематикою дослідження опубліковано 1 наукову працю у збірниках матеріалів конференцій.

1 АНАЛІЗ СУЧАСНОГО СТАНУ РОЗВИТКУ СИСТЕМ ДЛЯ ПЕРЕВІРКИ ЗНАНЬ З ПРАВИЛ ДОРОЖНЬОГО РУХУ.

1.1 Аналіз предметної області

Зі стрімким розвитком технологій мобільні системи стають все більш популярними в різних галузях. Мобільні пристрої стали невід'ємною частиною нашого щоденного життя, тому розробка навчальних ігор для Android-додатків набуває все більшої актуальності. У цьому контексті створення мобільного застосунку для вивчення правил дорожнього руху в формі вікторини може мати значний педагогічний та практичний ефект. Переваги використання такого застосунку включають інтерактивність та цікавість. Користувачі можуть вивчити правила дорожнього руху через інтерактивні вікторини, що робить процес більш захоплюючим та ефективним. Доступність і зручність такого застосунку дає можливість навчатися в будь-який час і в будь-якому місці, забезпечуючи гнучкість для користувачів.

Персоналізація є ще однією важливою перевагою. Застосунок може підлаштовуватися під рівень знань та потреб кожного користувача, надаючи індивідуальний досвід навчання. Мотивація та змагальність також можуть бути посилені через елементи гейміфікації. За зібрані бали будуть відкриватись додаткові ігри. Цей метод включає в себе використання елементів гри, таких як бали, рівні, нагороди та конкуренція, для залучення та мотивації користувачів до досягнення певних цілей чи виконання завдань. З часом з'являлися все більш складні та захоплюючі навчальні ігри, які використовували різноманітні методи навчання та взаємодію.

Додаткові можливості застосування можуть включати відеоуроки та навчальні матеріали для більш детального вивчення дорожнього руху, а також можливість підготовки до випробувань через вікторини, як на ті, що використовуються на офіційних випробуваннях. Такий застосунок може сприяти розширенню знань про правила дорожнього руху серед широкого загалу та підвищити безпеку дорожнього руху шляхом підготовки компетентних водіїв.

Перші навчальні ігри для мобільних пристроїв виникли на початковому етапі розвитку смартфонів і планшетів. Незважаючи на їхню простоту та обмежену функціональність, вони вже тоді відкривали можливості для вивчення нового та розвитку різних навичок. З часом навчальні ігри стали більш різноманітними, пропонуючи різні завдання та виклики, що сприяють розвитку критичного мислення, логіки, творчості та інших навичок. Поступово навчальні ігри стали візуально привабливішими, використовуючи різноманітні графічні ефекти для залучення користувача.

Отже, мобільний засіб для вивчення правил дорожнього руху у формі вікторини може мати значний вплив на процес навчання та підвищення рівня безпеки на дорогах. Його інтерактивність, доступність, персоналізація та можливості гейміфікації роблять його ефективним та цікавим інструментом для навчання та перевірки знань у цій важливій сфері. Розробка такого програмного застосунку не лише відповідає сучасним технологічним тенденціям, але й сприяє підготовці компетентних водіїв, що підвищує безпеку дорожнього руху.

1.2 Порівняльний аналіз аналогів

Використання індивідуальних платформ для вивчення правил дорожнього руху стає все більш популярним в останні роки, застосунки інтегрується в різні системи освіти, школи та автошколи. Розробка програмних модулів для реалізації мобільної системи для вивчення правил дорожнього руху, підвищити залученість та утримання користувачів, а також запропонувати нові можливості для бізнесу для взаємодії зі своїми клієнтами. До найбільш відомих рішень відносять:

- "ПДР: правила дорожнього руху 2023"
- "ПДР: перевірка знань правил дорожнього руху"
- "ПДР України"
- "ПДР: правила дорожнього руху 2023"

"ПДР: правила дорожнього руху 2023"

Функціональність: Тести та вікторини для перевірки знань, інтерактивні питання з поясненнями.

Актуальність інформації: Оновлюється на основі офіційних джерел.

Інтерфейс користувача: Простий та зручний.

Додаткові функції: Статистика прогресу та підтримка різних мов. Робота застосунку ПДР: правила дорожнього руху 2023 показана на рисунку 1.1.

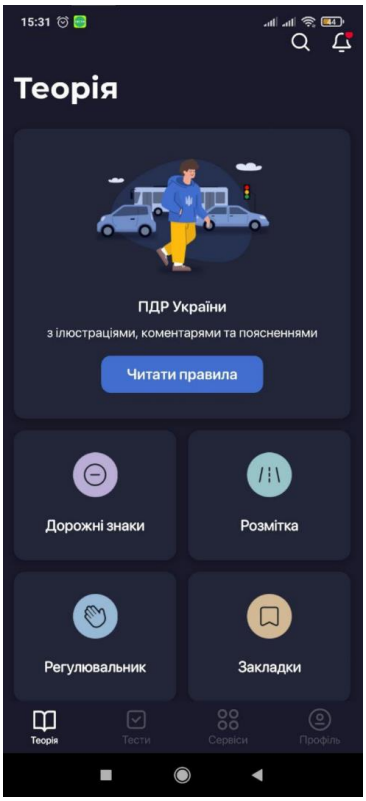

Рисунок 1.1 – "ПДР: правила дорожнього руху 2023"

Візуальні елементи та кольорова схема додатку вибрані таким чином, щоб забезпечити максимальний комфорт для очей та зменшити втому під час тривалого використання. Крім того, додаток включає функціонал, який дозволяє користувачам налаштовувати його відповідно до своїх особистих вподобань.

Наприклад, користувачі можуть обирати різні теми, налаштовувати звукові ефекти, а також використовувати різні типи нагадувань про необхідність проходження чергового тесту. Така персоналізація сприяє створенню індивідуального підходу до навчання, що є важливим фактором у підвищенні

ефективності навчання. Загалом, цей мобільний додаток для вивчення правил дорожнього руху в ігровій формі є потужним інструментом, який поєднує в собі функціональність, зручність та ефективність. Він надає користувачам можливість не лише проходити тести, але й глибше розуміти та запам'ятовувати правила дорожнього руху, що є ключовим аспектом у підвищенні безпеки на дорогах. Завдяки інтерактивності, підтримці різних мов, збереженню результатів та естетичному інтерфейсу, цей додаток є ідеальним вибором для тих, хто прагне стати більш обізнаним та компетентним водієм. Робота тестів показана на рисунку 1.2.

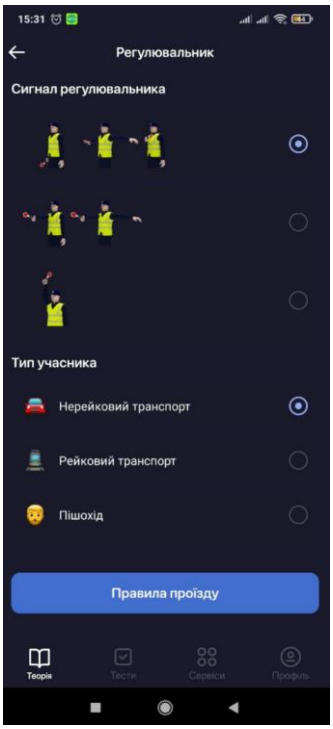

Рисунок 1.2 – Зображення тестів у додатку

"ПДР: перевірка знань правил дорожнього руху"

Функціональність: Тести з різних тем правил дорожнього руху та можливість створення власних тестів.

Цей додаток пропонує користувачам не лише тести, але й можливість перегляду текстів правил дорожнього руху. Він забезпечує доступ до повного списку правил, що дозволяє користувачам легко знайти та вивчити необхідну

інформацію. Крім того, додаток підтримує різні мови, що робить його доступним для користувачів з різних країн та регіонів.

Це особливо важливо в контексті глобалізації, коли люди часто подорожують і можуть мати потребу знати правила дорожнього руху в різних країнах. Додаток також дозволяє зберігати результати тестів для подальшого перегляду та аналізу. Це дає можливість користувачам відстежувати свій прогрес, ідентифікувати слабкі місця та працювати над ними. Збереження результатів може також мотивувати користувачів до подальшого навчання, оскільки вони бачать свої досягнення і можуть порівнювати свої результати з попередніми. Зручний та естетичний інтерфейс додатку робить його одним із популярних варіантів серед користувачів. Інтерфейс розроблений таким чином, щоб бути інтуїтивно зрозумілим, що полегшує навігацію та використання додатку навіть для тих, хто не має великого досвіду у використанні мобільних технологій. Робота застосунку ПДР: перевірка знань правил дорожнього руху показана на рисунку 1.3.

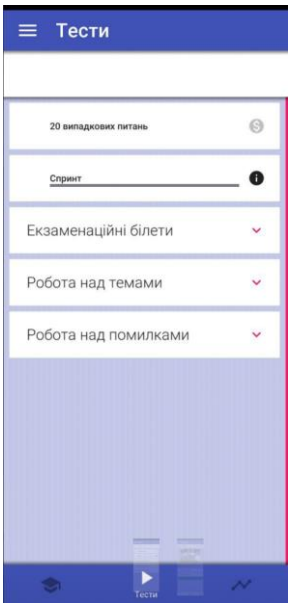

Рисунок 1.3 - "ПДР: перевірка знань правил дорожнього руху"

Цей додаток також надає можливість перевірити знання правил дорожнього руху через тести, але відрізняється від попереднього тим, що

дозволяє користувачам створювати власні тести та ділитися ними з іншими користувачами. Крім того, додаток має зручний інтерфейс, що спрощує навігацію і використання. Робота тестів показана на рисунку 1.4.

Рисунок 1.4 – Зображення тестів у додатку

Цей мобільний додаток для вивчення ПДР України пропонує користувачам не лише тести та вікторини, але й можливість перегляду текстів правил дорожнього руху. Завдяки цьому користувачі можуть легко знайти та вивчити необхідну інформацію без потреби у додаткових джерелах. Додаток включає різні типи тестів та вікторин, які допомагають користувачам перевірити свої знання ПДР України. Інтерактивні завдання сприяють глибшому засвоєнню матеріалу, а можливість перегляду текстів правил дорожнього руху забезпечує додаткову підтримку під час підготовки. Актуальність інформації в додатку гарантована за рахунок регулярних оновлень на основі офіційних джерел. Це означає, що користувачі завжди мають доступ до найсвіжіших даних, що є критично важливим для точного знання правил дорожнього руху та підготовки до іспитів.

Зручний та естетичний інтерфейс додатку робить його популярним серед користувачів. Інтерфейс розроблений з урахуванням потреб різних вікових груп,

що робить його легким у використанні навіть для тих, хто не має великого досвіду у використанні мобільних додатків. Інтуїтивно зрозуміла навігація та приємний візуальний дизайн забезпечують комфорт під час користування додатком. Робота застосунку ПДР України показана на рисунку 1.5.

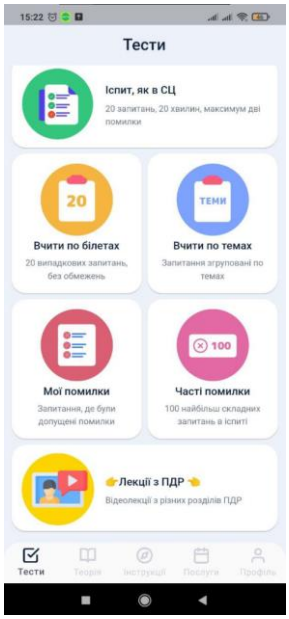

Рисунок 1.5 - "ПДР України"

Цей додаток надає можливість користувачам перевірити свої знання правил дорожнього руху через тести та вікторини. Він включає різні категорії питань, які охоплюють широкий спектр тем. Користувачі можуть проходити тести та перевіряти свій прогрес за допомогою статистики, яка показує результати кожного тесту. Додаток оновлюється регулярно для забезпечення актуальності інформації, а його інтуїтивний інтерфейс робить його зручним у використанні. Крім того, додаток дозволяє зберігати результати тестів для подальшого перегляду та аналізу. Це дає можливість користувачам відстежувати свій прогрес, ідентифікувати слабкі місця та працювати над ними. Збереження результатів може також мотивувати користувачів до подальшого навчання, оскільки вони бачать свої досягнення і можуть порівнювати свої результати з попередніми. Робота тестів показана на рисунку 1.6.

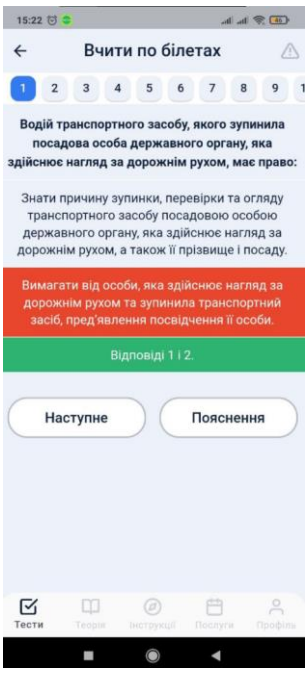

Рисунок 1.6 – Зображення тестів у додатку

Під час аналізу аналогів застосунку було виявлено такі недоліки:

1. "ПДР: правила дорожнього руху 2023"

Може містити обмежений обсяг інформації порівняно з більш продвинутими додатками.

Деякі користувачі можуть виявити тестові завдання надто простими або обмеженими.

2. "ПДР: перевірка знань правил дорожнього руху"

Додаток доступний лише для користувачів iOS, тому власники пристроїв на Android не можуть його використовувати.

Може бути меншою кількістю тестових завдань порівняно з іншими додатками.

3. "ПДР України"

Не всі функції можуть бути доступні для користувачів безкоштовної версії.

Може містити рекламу, що може бути дратуючою для деяких користувачів.

Деякі користувачі можуть виявити інтерфейс менш інтуїтивно зрозумілий порівняно з іншими додатками.

Створення нового застосунку, який буде абсолютно безкоштовним і пропонуватиме покращену інформацію про домедичну допомогу, має ряд переваг: Відсутність вартості для користувачів дозволить привернути більше уваги та збільшити користувацьку базу. Забезпечення більшої та детальнішої інформації про домедичну допомогу допоможе користувачам краще розуміти та вміти реагувати на екстренні ситуації. Розробка зручного інтерфейсу та навігації дозволить користувачам легко знаходити необхідну інформацію та використовувати додаток без зайвих труднощів. Приємний візуальний дизайн та зручність використання сприяють більшому залученню користувачів.

Користувачі, які отримують задоволення від використання додатку, більш схильні до регулярних занять, що підвищує їхню мотивацію. Зручний інтерфейс може включати можливості персоналізації, дозволяючи користувачам налаштовувати додаток відповідно до своїх вподобань та потреб. Гнучкість у використанні різних функцій підвищує задоволеність користувачів.

Можливість роботи у двох режимах - онлайн та офлайн - забезпечить доступність інформації в будь-який час та незалежно від наявності Інтернету. Офлайн режим буде особливо корисним у випадку втрати зв'язку або екстрених ситуацій. Наявність довідника у додатку дозволить користувачам знайти необхідну інформацію швидко та ефективно, що може бути критично в різних ситуаціях.

Оновлення і актуальність: Постійне оновлення та додавання нової інформації забезпечить актуальність та корисність додатку для користувачів на протязі часу.

Загалом, створення такого застосунку буде відповідати потребам користувачів у безкоштовній, доступній та корисній інформації про правила дорожнього руху та домедичну допомогу. Результати порівняння аналогів зведено в табл. 1.1.

Таблиця 1.1 – Порівняльні характеристика онлайн платформ

Критерії	ПДР України 2024	ПДР: правила дорожнього руху 2023	ПДР: перевірка знань ПДР	ПДР України
Функціональність	Тести та вікторина	Тести	Тести	Тести
Актуальність інформації	Постійно оновлюється на основі офіційних джерел	Оновлюється	Оновлюється з затримкою	Оновлюється на основі офіційних джерел
Інтерфейс користувача	Зручний та простий	Зручний	Простий	Зручний

Відповідно до таблиці порівняльних характеристик розробка власної мобільної гри-вікторини для перевірки знань правил дорожнього руху є доцільною. Отриманий продукт зможе покрити недоліки існуючих рішень та забезпечити новий досвід та можливості застосування вікторин для вивчення правил дорожнього руху.

Отже, розробка програмних модулів для впровадження мобільної системи з використанням гри вікторини відкриває перед бізнесом широкі можливості для взаємодії зі своїми клієнтами, створення унікального користувацького досвіду, а також підвищення рівня залученості та утримання користувачів. Розуміючи потенційні можливості застосування гри-вікторини та вимоги до розробки програмного забезпечення, компанії можуть створювати інноваційні та привабливі мобільні системи, які використовують гру вікторину для вивчення правил дорожнього руху для зростання та успіху.

Такий підхід до розробки мобільних систем відкриває шлях до нових можливостей у сфері освіти, розваг та взаємодії з аудиторією, що сприяє подальшому розвитку бізнесу та позитивному впливу на суспільство.

1.3 Аналіз методів розв’язання задачі

Розробка програмних модулів для реалізації мобільної застосунку для перевірки знань правил дорожнього руху підходів до вирішення проблеми. Існують різні методи, кожен з яких має свої переваги та недоліки. У цьому розділі проаналізовано найбільш ефективні методи розробки таких програмних модулів.

Одним з варіантів вирішення проблеми розробки програмних модулів для реалізації мобільного застосунку для перевірки знань правил дорожнього руху є створення окремої бази даних, що містить повний набір правил дорожнього руху разом з відповідями та поясненнями. Цей підхід дозволяє забезпечити швидкий доступ до інформації та ефективне управління контентом, що є ключовим для забезпечення точності та актуальності даних. Ще одним підходом є використання технологій штучного інтелекту для створення інтерактивного навчального середовища. Штучний інтелект може аналізувати відповіді користувачів та надавати індивідуалізовані рекомендації для подальшого вдосконалення знань з правил дорожнього руху. Такий підхід підвищує ефективність навчання та дозволяє адаптувати матеріали до потреб кожного користувача.

Крім того, можливим є використання методів гейміфікації для залучення користувачів та стимулювання їхньої активності. Це може включати в себе створення ігрових елементів, нагород та досягнень, які мотивують користувачів до активного вивчення та покращення своїх знань. Регулярні оновлення на основі офіційних джерел та збереження актуальності інформації є ключовими аспектами успішної розробки та використання мобільного застосунку для перевірки знань правил дорожнього руху.

Вибір методу для розробки мобільного застосунку для перевірки знань правил дорожнього руху є складним завданням, оскільки кожен підхід має свої переваги та недоліки. Однак, після уважного аналізу різних методів, було вирішено скористатися методом розробки власної мобільної платформи з вбудованим модулем для перевірки знань правил дорожнього руху. Хоча існують інші ефективні методи, такі як використання окремої бази даних або

технологій штучного інтелекту, вирішено обрати саме цей шлях через його кілька переваг. По-перше, власна мобільна платформа дозволить зберегти повний контроль над розробкою та підтримкою продукту. Це означає, що можна буде швидко реагувати на зміни та оновлення, а також вдосконалювати застосунок відповідно до потреб користувачів. По-друге, розробка власної мобільної платформи забезпечить унікальний функціонал та інтерфейс, що може відрізнятися від інших аналогічних застосунків.

Розглянувши переваги та недоліки кожного методу, було вирішено використати метод розробки власної мобільної-платформи з вбудованим модулем для реалізації мобільного застосунку для перевірки знань з правил дорожнього руху. Такий підхід дозволить отримати повноцінний програмний продукт без необхідності використання потужного апаратного забезпечення. Більше того, такий метод розробки дозволить створити унікальну платформу з унікальним функціоналом.

1.4 Висновки

У першому розділі розглянуто існуючі технології для перевірки правил дорожнього руху у вигляді гри-вікторини. Проведено порівняння відомих застосунків для перевірки знань, таких як: ПДР: правила дорожнього руху 2023, ПДР: перевірка знань правил дорожнього руху, ПДР України, ПДР: правила дорожнього руху 2023 У результаті порівняння виявлено, що досліджувані платформи набули значної популярності серед учасників освітнього процесу завдяки своїй зручності та доступності. Проаналізувавши переваги та недоліки відомих платформ виявлено, що вони не надають деякі можливості. Таким чином доведено доцільність розробки власного програмного рішення.

2 РОЗРОБКА МЕТОДІВ ТА МОДЕЛІ ПРОГРАМНОЇ СИСТЕМИ

2.1 Аналіз інформаційного забезпечення

У мобільному додатку для перевірки знань правил дорожнього руху, спроектованому з основною метою забезпечення освітнього процесу, кожна складова розроблена з ретельною увагою до деталей для досягнення максимальної зручності та ефективності для користувачів. Основний функціонал системи спрямований на глибоке засвоєння теоретичних аспектів правил дорожнього руху та стимулювання інтерактивного навчання. Система має базу даних, що містить різноманітні вікторини, інформацію про які обмінюється через визначені запитання та відповіді на тестові завдання. Поєднання різноманітних тематичних блоків забезпечує різноманіття та покриття всіх аспектів правил дорожнього руху. Крім того, додаток має модуль теорії, що працює в режимах онлайн та офлайн. Онлайн-режим дозволяє користувачам вивчати теоретичний матеріал за допомогою навчальних відео на платформі YouTube, тоді як офлайн-режим надає можливість засвоювати матеріал через навчальні посібники у форматі PDF. Також додаток включає ігри, доступ до яких користувач отримує після успішного проходження вікторин.

Це стимулює користувачів до активного навчання та розважає їх, дозволяючи відпочити від інтенсивного навчання. Крім того, додаток пропонує систему мотивації через нагороди та досягнення, що додає ігрового елементу до процесу навчання. Користувачам надається можливість вибору свого власного шляху навчання: лінійно пройти тести або відпочити та розважитися граючи. Цей підхід підвищує зацікавленість користувачів та підвищує ефективність засвоєння матеріалу. Крім того, система гарантує логічну послідовність у навчанні, розпочинаючи з правил дорожнього руху та поступово переходячи до більш складних аспектів, таких як дорожньо-транспортні пригоди та домедична допомога. Це дозволяє користувачам оцінити свій рівень знань об'єктивно та обирати подальший шлях у навчанні. Нарешті, додаткові програмні модулі розширюють функціональні можливості навчальної системи, включаючи блок

теоретичного матеріалу, модуль перевірки знань, вільний метод навчання, ведення статистики проходження тестів та модуль мотивації та нагородження.

Це сприяє створенню більш привабливого та ефективного навчального середовища для користувачів.

2.2 Розробка алгоритму нарахування балів гравцю

Для забезпечення ефективного та захоплюючого вивчення матеріалу у мобільній грі, розроблена система нарахування балів, яка базується на продуктивності та швидкості реагування користувачів. Ця система гарантує об'єктивність оцінювання і мотивує гравців до активної участі у процесі навчання. Алгоритм працює так: він ретельно аналізує час, за який гравці відповідають на запитання. Якщо гравець дає правильну відповідь і кількість правильних відповідей перевищує кількість неправильних, він отримує додаткові бали, які обчислюються як різниця між правильними та неправильними відповідями. Це стимулює гравців давати швидкі та точні відповіді. У випадку, якщо гравець набирає менше правильних відповідей, ніж необхідно, бали не додаються. Більше того, якщо гравець не завершив вікторину за 60 секунд, вікторина закривається, і він змушений починати з початку.

Це підкреслює важливість швидкості у відповіді на запитання. Такий алгоритм допомагає розвивати швидкість та точність реагування гравців, що позитивно впливає на їхній прогрес у грі. Система нарахування балів стимулює гравців досягати високих результатів, швидко і правильно відповідати, що підвищує інтенсивність та цікавість гри. Цей алгоритм є ключовим елементом у створенні захоплюючого і ефективного ігрового середовища. Він дає кожному гравцеві можливість проявити свої навички та змагатися з іншими, відчуваючи постійне заохочення до вдосконалення. Блок-схема алгоритму нарахування балів не лише стимулює активність гравців, але й сприяє створенню динамічного і змістовного ігрового процесу. Основна мета цього алгоритму - забезпечити чесну і справедливу гру, винагороджуючи тих, хто швидко і правильно реагує на запитання. Перевагою цього підходу є його адаптивність до різних типів гравців.

Наприклад, для тих, хто добре знає правила гри та швидко реагує, цей алгоритм підтримує високий темп відповіді. Для новачків або тих, хто бажає ретельно перевірити свої знання перед відповіддю, алгоритм пропонує заохочення до точності та дозволяє вчитися на помилках. Таким чином, блок-схема алгоритму не лише стимулює змагання і підвищує мотивацію гравців, але й створює сприятливі умови для розвитку їхніх навичок. Це забезпечує цікавий та захоплюючий ігровий досвід для кожного учасника. Додатково, система нарахування балів передбачає кілька рівнів складності, які адаптуються під рівень знань гравців.

Це дозволяє забезпечити поступове ускладнення завдань, що є важливим для підтримання інтересу і стимулювання постійного вдосконалення. Враховуючи всі ці аспекти, система нарахування балів у мобільній грі є важливим елементом, який сприяє не лише мотивації та змагальності, але й забезпечує глибоке та ефективне вивчення матеріалу. Завдяки цьому, користувачі отримують не тільки задоволення від гри, але й значні знання та навички, які можуть бути корисними у реальному житті. Розробка мобільного додатку для перевірки знань правил дорожнього руху із системою нарахування балів, орієнтованою на продуктивність та швидкість реагування користувачів, є важливим кроком у створенні ефективного навчального інструменту. Завдяки ретельному аналізу часу відповідей та врахуванню точності відповідей, алгоритм стимулює гравців до швидкого і точного виконання завдань.

Цей підхід не тільки підвищує інтенсивність та захопливість ігрового процесу, але й сприяє глибокому освоєнню матеріалу. Адаптивність системи до різних рівнів підготовки користувачів забезпечує індивідуальний підхід до навчання, дозволяючи як новачкам, так і досвідченим гравцям отримати максимальну користь від використання додатку. Інтеграція елементів гейміфікації, таких як ігрові досягнення та нагороди, підтримує високу мотивацію користувачів та стимулює їх до постійного вдосконалення знань.

Таким чином, використання мобільного додатку для вивчення правил дорожнього руху із вдосконаленою системою нарахування балів забезпечує не

тільки цікаве та захопливе навчання, але й сприяє підвищенню загального рівня знань користувачів. Блок-схему алгоритму начислення балів користувачу зображено на рисунку 2.1.

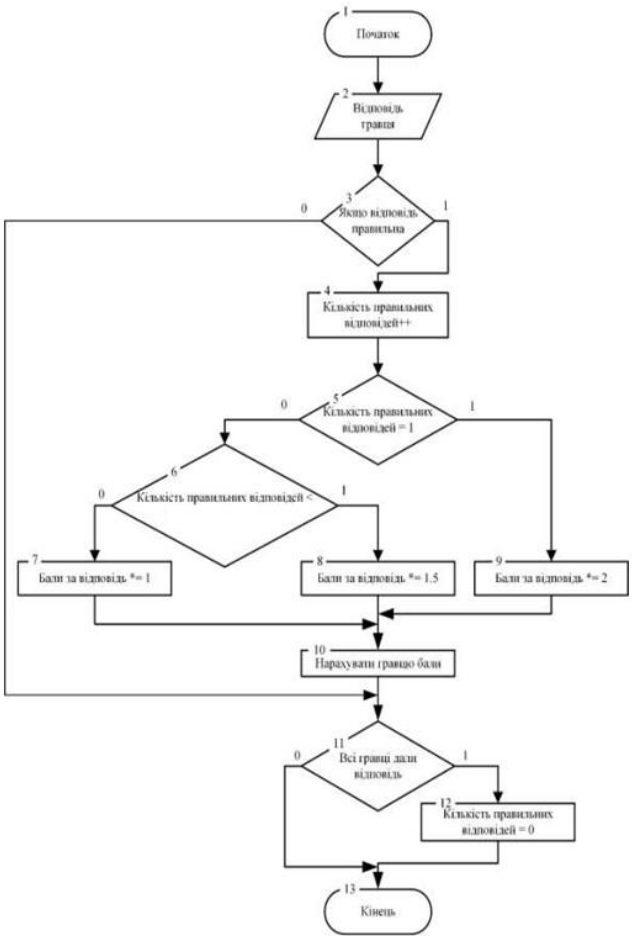

Рисунок 2.1 – Блок-схема алгоритму начислення балів користувачу

Опис блоків алгоритму:

1. Початок алгоритму. Перейти на блок (2).
   Вхідні параметри – Відповідь користувача. Перейти на блок (3).
2. Перевірка на правильну відповідь гравця. Якщо відповідь правильна перейти на блок (4), інакше перейти на блок (11).
3. Збільшити лічильник правильних відповідей на 1. Перейти на блок (5).
4. Перевірка чи кількість правильних відповідей дорівнює 1. Якщо
5. кількість правильних відповідей дорівнює 1 перейти на блок (9), інакше перейти на блок (6).

6. Перевірка чи кількість правильних відповідей менша ніж 50% гравців. Якщо кількість правильних відповідей менша за 50% гравців перейти на блок (8), інакше перейти на блок (7).
7. Нарахувати гравцю бали. Перейти на блок (11).
8. Перевірка чи користувач відповів на питання. Якщо користувач відповів на питання перейти на блок (12), інакше перейти на блок (13).
9. Встановити значення кількості правильних відповідей на 0. Перейти на блок (13).
10. Кінець алгоритму.

2.3 Розробка моделі системи

Для кращого розуміння роботи програмних модулів мобільного застосунку для перевірки правил дорожнього руху було вирішено розробити модель роботи системи на прикладі UML діаграми діяльності (рис. 2.10). Згідно з вказаною діаграмою, першим кроком користувач повинен обрати бажану вікторину. Після цього він може розпочати проходження вікторини. Якщо результат тестування виявився невдалим, користувачу пропонується пройти вікторину знову.

У разі невдалого проходження вікторини користувач має кілька варіантів дій. Він може розпочати повторне проходження вікторини, що дозволяє ще раз спробувати відповісти на запитання і покращити свій результат. Це допомагає закріпити знання та підвищити свої шанси на успішне завершення вікторини в наступній спробі.

Якщо користувач не бажає повторно проходити вікторину, він може перейти у вікно "Теорія". У цьому розділі користувач має можливість покращити свої знання, переглядаючи теоретичні матеріали. Ці матеріали можуть бути представлені у вигляді статей, відео або інтерактивних навчальних модулів, що допомагають користувачу глибше розібратися в темі і підготуватися до наступної спроби проходження вікторини. На рисунку 2.2 зображена діаграма діяльності системи.

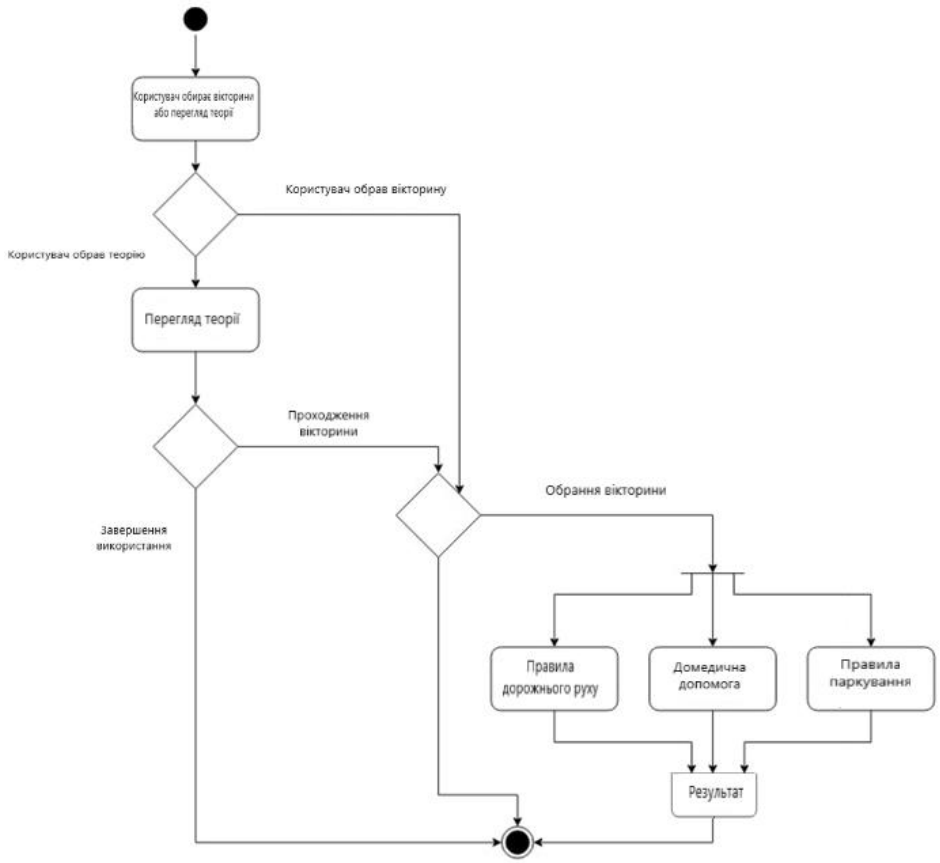

Рисунок 2.2 – Діаграма діяльності мобільної системи

2.4 Розробка алгоритмів роботи системи

Для розробки мобільного застосунку, для перевірки правил дорожнього руху, необхідно розробити загальний алгоритм, згідно якого буде працювати мобільний застосунок. Згідно даного алгоритму, першим ділом, користувач отриму знаходиться на головному меню та має можливість пройти вікторину або повторити теорію. Обравши вікторину, йому необхідно переглянути яку саме вікторину він хоче пройти. Доки користувач не набрав відповідну кількість балів йому недоступні ігри у застосунку. Отже, після входу в систему, користувач обирає вікторину, гру або теорію

Додатковий алгоритм для модуля перегляду пам'яток має велике значення для покращення розуміння роботи цього модуля. Згідно з цим алгоритмом, перший крок - це обрання вікторини. Результати проходження вікторини зберігаються у відокремленому файлі. Користувач має можливість переглянути теорію, де детально описані правила дорожнього руху або першої медичної

допомоги. Також, надається можливість перегляду теорії у режимі онлайн. Додатково, користувач може взяти участь у грі, що стимулює та гейміфікує процес навчання. Алгоритм підлаштовується в залежності від кількості балів, набраних користувачем. Розглянувши ці алгоритми, можна зрозуміти, що розроблений додаток сприяє навчанню та покращенню знань. Автоматизація системи перевірки вікторин зроблює додаток привабливим вибором для користувачів. Такий підхід не лише полегшує процес навчання, а й надає мотивацію до вдосконалення своїх знань та навичок у дорожньому русі та медичній допомозі.

Для кращого розуміння роботи модуля перегляду пам'яток, було розроблено додатковий алгоритм. Згідно з даним алгоритмом, першим етапом є обрання вікторини. Після проходження вікторини, результат зберігається в окремому файлі, що дозволяє користувачу відстежувати свої досягнення і прогрес.

Користувач має можливість переглянути теорію, де детально описано правила дорожнього руху або надання першої домедичної допомоги. Це забезпечує доступ до важливої інформації, що допомагає користувачу підготуватися до вікторини та поглибити свої знання. Теорія може бути представлена у вигляді текстових матеріалів, ілюстрацій та інтерактивних елементів, що полегшує засвоєння інформації.

Також є можливість переглянути теорію у режимі онлайн. Це дозволяє користувачу отримати доступ до оновленої інформації, відеоуроків та інших ресурсів, які можуть бути корисними для підготовки. Онлайн-режим забезпечує актуальність та доступність матеріалів незалежно від місця знаходження користувача. Крім того, користувач може пограти у гру, що додає елемент гейміфікації та підвищує мотивацію. Гра дозволяє закріпити отримані знання у веселій та інтерактивній формі. Це робить процес навчання більш захоплюючим і цікавим. Алгоритм підлаштовується відповідно до кількості балів, що набрав користувач. Якщо користувач набрав достатню кількість балів, йому можуть відкриватися нові рівні, вікторини або додаткові ігрові можливості. Це стимулює

користувача до подальшого навчання та досягнення кращих результатів. Алгоритм діяльності головного екрану мобільної платформи зображено на рисунку 2.3.

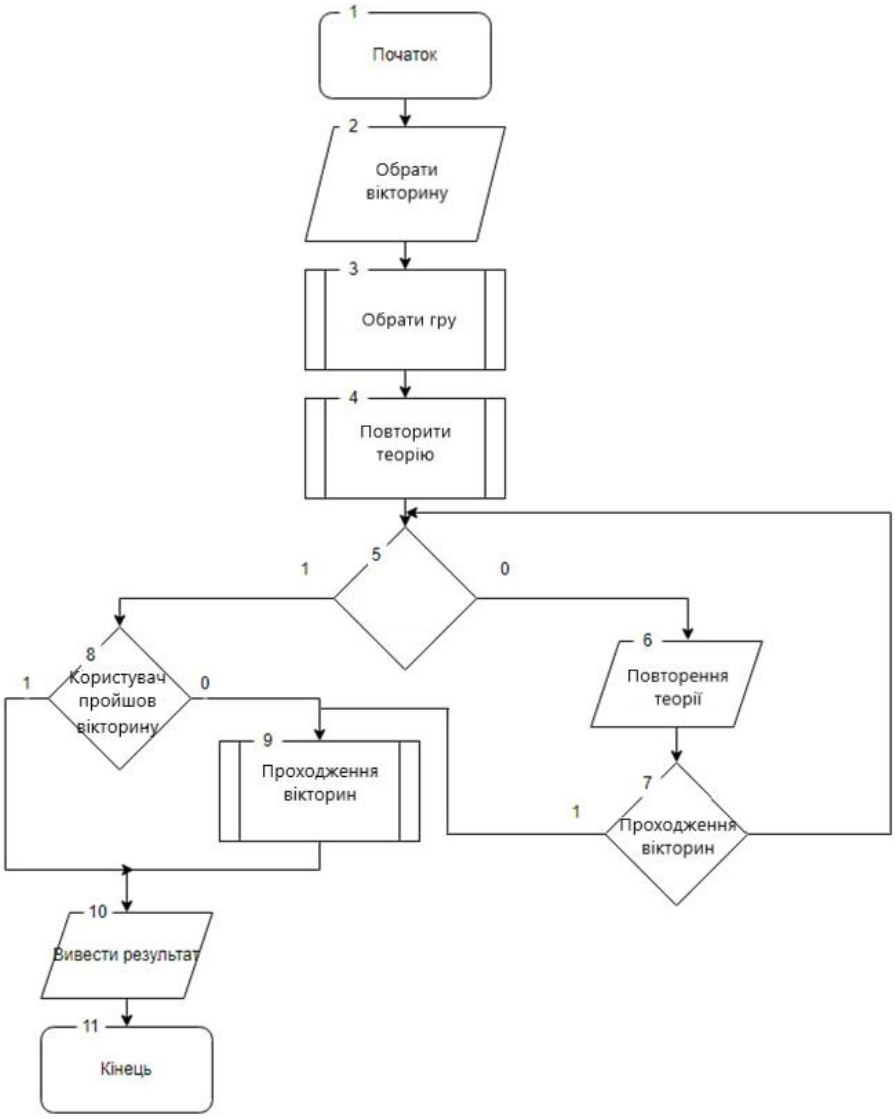

Рисунок 2.3 – Головний екран мобільної платформи

На головному екрані користувач має змогу обрати тему вікторини «Правила дорожнього руху», «Перша домедична допомога», «Теорія». Також є можливість пограти у гру якщо набрана потрібна кількість балів.

2.4 Алгоритм розробки гри для перевірки знань з правил дорожнього руху

Алгоритми розробки гри для мобільного застосунку перевірки знань правил дорожнього руху Додавання гри до мобільного застосунку для перевірки знань правил дорожнього руху має на меті зробити його більш привабливим, цікавим та ефективним для користувачів. Графічні елементи, звукові ефекти та інтерактивний ігровий процес можуть значно покращити користувацький досвід, мотивуючи користувачів активно використовувати додаток та підвищуючи їхню залученість. Створення інтерфейсу гри: На початку розробки потрібно створити інтерфейс гри. У методі onCreate() здійснюється ініціалізація основних компонентів, таких як кнопка запуску гри (startBtn), контейнер для гри (rootLayout), текстове поле для відображення результату (score) та сама гра (mGameView).

Також налаштовуються медіаплеєри для відтворення звукових ефектів, що супроводжують різні події у грі, такі як старт і фініш. Обробка запуску гри: Коли користувач натискає кнопку старту гри, активується обробник подій. Цей обробник запускає звук запуску автомобіля за допомогою медіаплеєра (mediaPlayerStartCar), встановлює фонове зображення дороги для модуля гри (mGameView) і додає саму гру до кореневого макету (rootLayout). Після цього кнопка старту та текстове поле для відображення результату приховуються, що дозволяє користувачу зосередитись на ігровому процесі. Закінчення гри та обробка результатів: Після завершення гри викликається метод closeGame(), який відповідає за відображення результатів гри у текстовому полі (score). Він також відтворює звук завершення гри за допомогою медіаплеєра (mediaPlayerFinishCar) і відновлює видимість кнопки старту з новим написом "Головне меню". При натисканні на цю кнопку користувач перенаправляється на головний екран додатку (MainActivity), де він може вибрати інші можливості або почати гру знову. Переваги та недоліки гри у застосунку Переваги: Залученість користувачів: Гра привертає увагу користувачів, роблячи застосунок більш привабливим та цікавим.

Мотивація: Ігровий процес та можливість отримувати бали чи нагороди стимулюють користувачів частіше використовувати додаток, підвищуючи їхню активність.

Унікальність: Додавання гри робить застосунок більш унікальним та відмінним від конкурентів, що може залучити нових користувачів.

Взаємодія: Гра надає користувачам додаткові можливості для взаємодії із застосунком, що підвищує тривалість сесій та активність користувачів.

Покращення користувацького досвіду: Графіка та звукові ефекти роблять користування додатком приємним та захоплюючим, створюючи позитивні враження. Недоліки: Складність розробки: Розробка та інтеграція гри вимагають додаткових ресурсів, включаючи час та технічні знання, що може ускладнити процес розробки.

Відволікання: Ігровий елемент може відволікати користувачів від основної мети додатку – навчання правилам дорожнього руху, що може негативно позначитись на їхній концентрації.

Інтеграція гри в мобільний додаток для перевірки знань правил дорожнього руху може значно покращити його привабливість та ефективність. Гра підвищує мотивацію користувачів, збільшує тривалість сесій та створює додаткові можливості для взаємодії із застосунком. Загалом, додавання гри може бути вигідним кроком для підвищення конкурентоспроможності та успішності додатку на ринку, забезпечуючи користувачам захоплюючий та ефективний інструмент для вивчення правил дорожнього руху.

2.5 Розробка інтерфейсу програмного додатку

Розробляючи інтерфейс мобільної системи вирішено зробити його максимально простим та дружнім для використання користувачем.

Основним кольором інтерфейсу обрано теплі тони жовтого кольору, оскільки даний колір

Теплі тони жовтого кольору можуть бути відмінним вибором для застосунків з кількома причинами: Жовтий колір асоціюється з енергією,

радістю і оптимізмом. Використання теплих тонів жовтого може підвищити настрій користувачів і зробити їх відчуття радісними та енергійними. Теплі тони жовтого можуть бути привабливими і привертають увагу користувачів. Це може допомогти залучити більше користувачів до застосунку і створити позитивне перше враження. Кольори грають важливу роль у запам'ятовуваності застосунків. Теплі тони жовтого можуть зробити ваш застосунок більш запам'ятовуваним для користувачів, особливо якщо вони використовуються у логотипах, іконках або інших ключових елементах дизайну. Теплі кольори нагадують про сонце і тепло. Вони можуть створювати відчуття комфорту та затишку, що робить застосунок більш привабливим для користувачів. У загальному, використання теплих тонів жовтого може сприяти покращенню враження користувачів від вашого застосунку та зробити його більш привабливим і енергійним.

При відкритті додатку, користувача зустрічає екран меню зображено на рисунку 2.1, з якою користувач буде одразу розуміти логіку дисплею.

Рисунок 2.1 – Головне меню

Відкриття головного меню після завантаження застосунку може бути доброю практикою з кількох причин: Після запуску застосунку користувач очікує бачити головне меню, де він може здійснити вибір між різними функціями або розділами. Це робить використання застосунку більш інтуїтивно зрозумілим і зручним для нього. Відкриття головного меню дозволяє користувачам знайти доступні функції та навігаційні елементи безпосередньо після запуску застосунку, що сприяє збереженню чіткості і простоти інтерфейсу. Головне меню може включати посилання на різні розділи або функції застосунку, дозволяючи користувачам швидко переходити до потрібної інформації або функціоналу. Відкриття головного меню може також слугувати певним сигналом для користувача про те, що застосунок готовий до використання і очікує його вибору. Це може створити позитивне перше враження від застосунку.

Отже, відкриття головного меню після завантаження застосунку може поліпшити загальний досвід користувача та зробити застосунок більш зручним і привабливим для використання Екран інформації для користувача зображено на рисунку 2.2.

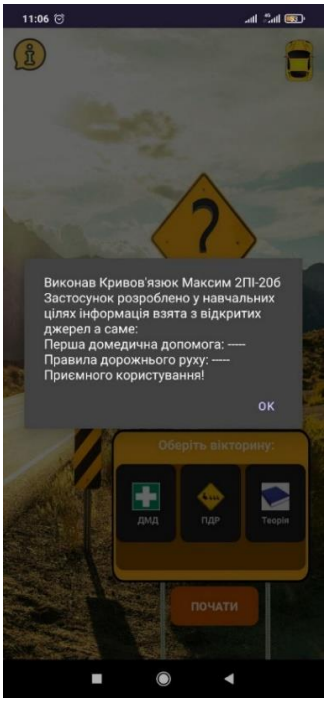

Рисунок 2.2 – Екран інформації для користувача

При натисканні на значок «Інформація» користувач отримає повідомлення з інформацією про розробника та посилання на основні джерела інформацією. Використання іконки гри, яка стає доступною лише при набутті користувачем необхідної кількості балів (у вашому випадку, 30 балів), має декілька переваг: Користувачеві стає цікавою можливість грати у гру, що додає мотивації для участі в вікторині та активного взаємодії з вашим застосунком. Він буде старатися набрати необхідну кількість балів, щоб отримати доступ до гри. Якщо гра стає доступною лише при досягненні певного рівня, користувач буде більш зацікавлений у виконанні завдань та отриманні балів, щоб розблокувати її. Це може зробити його залучення до застосунку більш стійким та продуктивним. Користувач буде мотивований вивчати матеріал і виконувати завдання, щоб отримати більше балів. Отже, використання іконки гри як мотиваційного елементу може позитивно підвищити зацікавленість та активність користувачів у використанні вашого застосунку, а також сприяти їхньому навчанню та залученню. Повідомлення про потрібну кількість балів зображено на рисунку 2.3.

Рисунок 2.3 – Повідомлення про потрібну кількість балів

Переваги перекидання користувача у внутрішнє меню першої домедичної допомоги після натискання на гру можуть включати: Перекидання користувача у внутрішнє меню першої домедичної допомоги під час натискання на гру чітко вказує на те, що гра пов'язана з медичною тематикою. Це може зробити застосунок більш послідовним та логічним для користувача.

Перехід до внутрішнього меню першої домедичної допомоги дозволяє користувачам отримати більше інформації або доступ до конкретних функцій, пов'язаних з медичною тематикою, що може поліпшити їхній досвід використання застосунку. Перехід до внутрішнього меню першої домедичної допомоги дозволяє користувачам легко знайти потрібну інформацію або функціонал, пов'язаний з медичною допомогою, без зайвих кліків або пошуків та зображено на рисунку 2.4.

Рисунок 2.4 – Меню ДМД

Переваги створення окремого меню з правилами дорожнього руху можуть включати: Окреме меню дозволяє структурувати інформацію про правила дорожнього руху у вашому застосунку. Користувачі зможуть легко знаходити

потрібні правила та інформацію про них. Меню з правилами дорожнього руху може допомогти підвищити обізнаність користувачів щодо дорожнього руху, що може бути особливо важливим для безпеки на дорозі. Можливість швидко знайти потрібну інформацію про правила дорожнього руху у вашому застосунку допоможе уникнути відволікання користувачів від пошуку цієї інформації в інших джерелах. Наявність інформації про правила дорожнього руху у вашому застосунку може зробити його більш цінним для користувачів, особливо для тих, хто цікавиться покращенням своєї безпеки на дорозі або підготовкою до іспитів на права.Меню ПДР зображено на рисунку 2.5.

Рисунок 2.5 – Меню ПДР

У вікні з вікториною присутня кнопка назад, назва вікторина та таймер при завершенні якого вікторина закривається. Нижче наведено зображення з питанням, нижче варіанти відповідей та кнопка наступне. Створення інтерфейсу вікторини з кнопкою "Назад", назвою вікторини, таймером та варіантами відповідей має кілька переваг: Кнопка "Назад" дозволяє користувачам легко повертатися до попереднього екрана або виходити з вікторини в будь-який момент, що робить використання застосунку більш зручним та інтуїтивно

зрозумілим. Наявність назви вікторини та таймера дозволяє користувачам завжди мати уявлення про те, яка вікторина є активною, і скільки часу залишилося для відповідей.

Це робить процес участі в вікторині більш контрольованим та зорієнтованим. Наявність зображень з питаннями та варіантами відповідей робить інтерфейс вікторини більш привабливим та цікавим для користувачів, що може позитивно підвищити їхній інтерес до участі в вікторині. Наявність кнопки "Наступне" дозволяє користувачам переходити до наступного питання швидко та без зайвих зусиль, що допомагає зберегти їхній інтерес та захопленість процесом вікторини. Вікно вікторини зображено на рисунку 2.6.

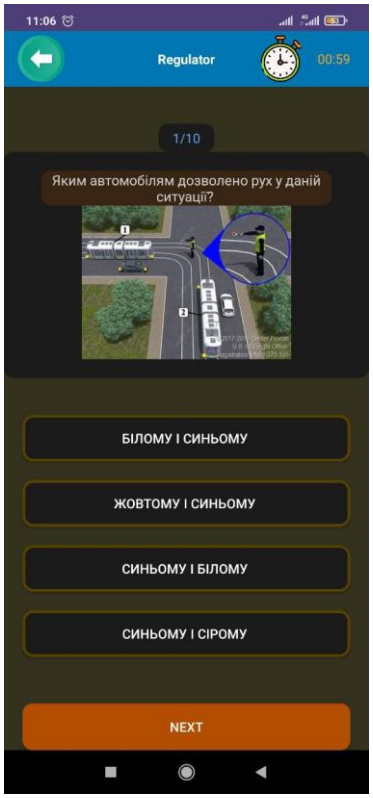

Рисунок 2.6 – Екран «Цікаве»

Після проходження вікторини користувач бачить вікно з результатом проходження вікторини, відповідно до набраних балів відрізняється анімація зображення «крокодила», якщо більше 7/10 правильно, то крокодил радісний їде на машині, якщо менше, то крокодил злий та без машини. Також присутній вивід

загального результату він формується наступним чином: різниця між правильними та неправильними, якщо правильних більше, то вони додаються у відповідний файл, потім відкриваються ігри відповідно до кількості балів

Створення такого інтерфейсу для відображення результатів проходження вікторини має декілька переваг: Анімація зображення «крокодила» залежно від кількості правильних відповідей може стимулювати користувача до досягнення кращих результатів. Вікно з результатом зображено на рисунку 2.7.

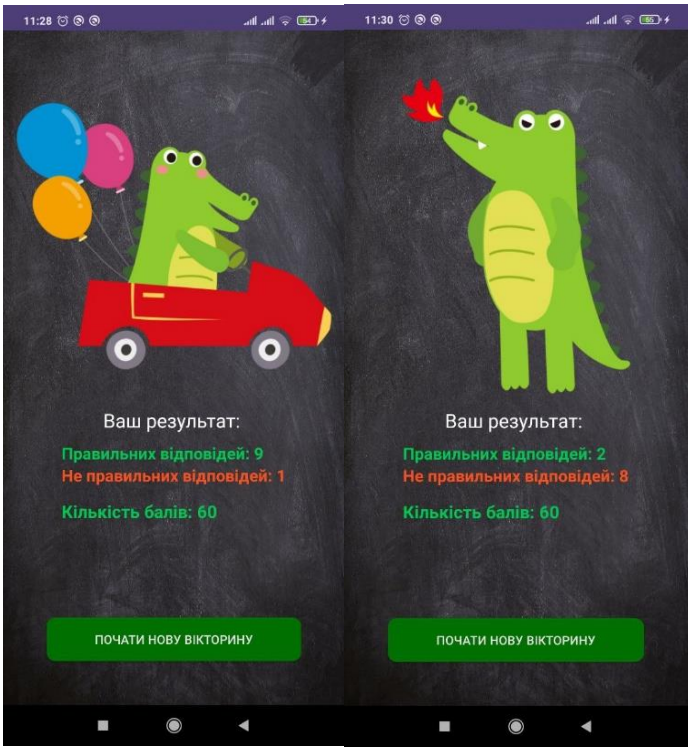

Рисунок 2.7 – Екран з результатом «Позитивний» та «Негативний»

Вивід загального результату у вигляді різниці між правильними та неправильними відповідями дозволяє користувачеві швидко оцінити свій успіх у вікторині та зрозуміти, де саме він зробив помилки або де вже є знання. Збереження результатів у відповідний файл та відкриття ігор відповідно до кількості набраних балів дозволяє персоналізувати досвід користувача. Він може бачити свій прогрес та відчувати, що застосунок враховує його досягнення. Отже, такий інтерфейс дозволяє не лише ефективно відображати результати

вікторини, але й підвищує мотивацію та зацікавленість користувача в подальшому використанні застосунку гейплей зображено на рисунку 2.8.

Рисунок 2.8 – Екран «Гра»

Переваги заохочення користувачів проходити тести з правил дорожнього руху та домедичної допомоги та набирати необхідну кількість балів для відкриття інших функцій у вашому застосунку Заохочення користувачів проходити тести та набирати бали може стимулювати їх активно навчатися правилам дорожнього руху та основам домедичної допомоги, оскільки вони будуть мати конкретну мету - отримати доступ до інших функцій або можливостей в застосунку.

Поступове відкриття нових можливостей у застосунку заохочує користувачів до постійного використання та просування в навчанні. Користувачі можуть мати різний темп навчання, тому деякі з них можуть відчувати тиск або відставання у відношенні до інших, які швидше прогресують. Зосередженість на отриманні балів може призвести до поверхневого засвоєння матеріалу без глибокого розуміння та застосування його у реальних ситуаціях. Отже, хоча заохочення користувачів проходити тести та набирати бали може мати свої

переваги, важливо також ретельно розробляти цю систему, щоб уникнути негативних наслідків та забезпечити ефективну та збалансовану навчальну діяльність.Зображено на рисунку 2.9.

Рисунок 2.9 – Екран «Теорія»

Екран «Теорія» що має дві кнопки Онлайн та Офлайн, у вікні онлайн знаходяться навчальні відео а у вікні офлайн довідники у форматі PDF(рис. 2.9).

Розробка графічного інтерфейсу онлайн платформи була проведена у середовищі розробки Android Studio з використанням технології Java.

2.6 Висновки

У другому розділі було проведено аналіз вхідних та вихідних даних програмного продукту. Також було розглянуто можливість покращення продуктивності мобільної платформи шляхом кешування вихідних даних. Було розроблено основний алгоритм роботи мобільної платформи та алгоритм валідації особистих даних. Розроблено інтнрфейс застосунку. Також було розроблену модель роботи програмного продукту за допомогою UML діаграм.

3 РОЗРОБКА ПРОГРАМНИХ МОДУЛІВ ДЛЯ ЗАСТОСУНКУ ДЛЯ ПЕРЕВІРКИ ЗНАНЬ З ПРАВИЛ ДОРОЖНЬОГО РУХУ У ВИГЛЯДІ ГРИ-ВІКТОРИНИ.

3.1 Варіантний аналіз і обґрунтування вибору засобів для реалізації програмного засобу

Розробляючи програмні модулі для мобільного застосунку для перевірки знань з правил дорожнього руху, важливо ретельно обирати технології, що використовуються. Java та Kotlin - це дві з популярних мов програмування, які знаходять широке застосування у розробці додатків для різних платформ, зокрема для Android. Обидві мови мають свої власні особливості та переваги, але вони доповнюють одна одну у світі мобільної розробки.

Java - це мова програмування, яка займає провідне місце в історії розробки програмного забезпечення. Вона вже довгий час використовується для розробки Android-додатків і є основною мовою програмування для цієї платформи. Однією з головних переваг Java є її надійність та стабільність. Це мова з великою кількістю бібліотек та фреймворків, які сприяють швидкій і ефективній розробці додатків для Android.

Kotlin - це мова програмування, яка швидко набирає популярність в середовищі розробки Android-додатків. Вона була розроблена компанією JetBrains із метою покращення розробки на базі JVM. Kotlin пропонує багато сучасних можливостей, таких як коротка синтаксична конструкція, безпечні типи даних та нульова безпека. Вона є доповненням до Java, оскільки вони можуть існувати разом у проектах, що дозволяє розробникам поступово переходити на нову мову без великого внеску в переписування коду.

Android Studio - це інтегроване середовище розробки (IDE), яке спеціально призначене для створення додатків для платформи Android. Воно забезпечує розробникам всі необхідні інструменти та середовище для ефективної розробки додатків, включаючи редактор коду, інструменти для збірки та налагодження додатків, емулятори пристроїв та інше. Android Studio підтримує як Java, так і

Kotlin, надаючи розробникам вибір між цими двома мовами програмуваннями для реалізації своїх ідей та проектів.

Створення застосунка для перевірки знань з правил дорожнього руху за допомогою Java та Kotlin має свої переваги: Використання Java та Kotlin дає можливість вибрати мову програмування в залежності від потреби. Java вже має велику базу розробників і бібліотек для розробки Android-додатків, тоді як Kotlin надає сучасний підхід до програмування з багатьма сучасними можливостями та безпекою типів даних. Застосунок доступний на різних платформах, Kotlin Multiplatform, є можливість поділити логіку програми між Android і iOS.

Якщо застосунок потребує обробки багатьох запитів одночасно, є можливість використовувати потоки (threads) для оптимізації роботи програми. Java та Kotlin надають потужні інструменти для роботи з потоками. Застосунок має важливу функціональність, важливо, щоб він був стабільним і безпечним. Java, так і Kotlin мають вбудовані механізми безпеки та перевірки типів даних, що допомагає уникнути багатьох типових помилок програмування. Kotlin відомий своєю зручністю та простотою використання, що може значно збільшити швидкість розробки застосунка. Його короткий синтаксис та розширена підтримка інструментів розвитку допомагають розробникам швидше досягти своїх цілей.

Отже, Вибір технологій для розробки мобільних додатків дійсно вирішальний крок, оскільки від нього залежить якість, продуктивність та зручність додатку для користувачів. Використання інструментів вікторини в Java, Kotlin та Multiplatform може створити потужний фундамент для розробки. Java є однією з найпопулярніших мов програмування, вона має велику спільноту розробників та багатий екосистему. Kotlin, з іншого боку, є сучасною мовою, що надає безліч переваг, таких як безпека типів, компактний синтаксис та інтероперабельність з Java. Використання Kotlin дозволяє розробникам швидше та зручніше створювати якісні додатки для Android.

3.2 Розробка модуля реалізації меню та вікторини.

Головною задачею мобільного застосунку для превірки правил дорожнього руху є перевірка та мотивація вивчання правил дорожнього руху. Мета створення застосунку для перевірки знань правил дорожнього руху з ігровим елементом та відео-або текстовою теорією полягає в створенні комплексного інструменту, який сприятиме підвищенню рівня свідомості та навичок користувачів у сфері безпеки на дорозі. Застосунок надає можливість користувачам вивчати правила дорожнього руху через вікторини, що дозволяє їм відповідати на питання і перевіряти свої знання.

Після вікторини користувачі можуть звернутися до розділу з теоретичним матеріалом, який може бути представлений у форматі відео або тексту. Це дозволяє навчити користувачів не лише знати правила, але і розуміти їх суть та важливість. Наявність ігрового елементу в застосунку, такого як набір балів за відповіді у вікторині, стимулює користувачів до активного вивчення матеріалу та досягнення кращих результатів.

Додатково, можливість отримати доступ до цікавого та інформативного контенту (відео або текст) після досягнення певного рівня балів ще більше заохочує користувачів до використання застосунку. Застосунок може бути доступним безкоштовно для всіх користувачів, що дозволяє широкому колу людей отримати доступ до якісного освітнього контенту з правил дорожнього руху. Більше того, застосунок може бути розроблений для різних платформ, таких як Android і iOS, що робить його доступним для більшого кола аудиторії.

Освіченість учасників дорожнього руху щодо правил дорожнього руху сприяє підвищенню безпеки на дорозі та зменшенню кількості дорожньо-транспортних пригод. Застосунок створює можливість для кожного користувача здобути знання та навички, які допоможуть їм стати більш відповідальними учасниками дорожнього руху. Цей клас MainActivity є основною активністю застосунку для перевірки знань з правил дорожнього руху. Він містить код для налаштування головного екрану, де користувач може вибрати різні теми для вікторин, переглянути теоретичні матеріали, а також запустити гру.

Основні функціональні можливості включають вибір теми для вікторини, перегляд інформаційного діалогового вікна, відстеження першого запуску програми та управління звуковими ефектами. На початку оголошуються змінні: PREFS_NAME та FIRST_RUN_KEY, які використовуються для збереження інформації про перший запуск програми, і об'єкти MediaPlayer для відтворення різних звукових ефектів. У методі onCreate встановлюється основне представлення (activity_main), ініціалізуються графічні елементи (макети та кнопки), а також звукові ефекти. Встановлюється слухач натискань для елементу з інформацією, який відкриває діалогове вікно. Перевіряється, чи є це першим запуском програми, та оновлюється відповідне налаштування.

Користувачі натискають макети DMD, PDR та theoryMenu змінюють вибрану тему і відповідно змінюють фонові ресурси для візуальної індикації вибору. При натисканні на кнопку startQuiz перевіряється, чи вибрано тему, і відповідно запускається відповідна активність або відображається повідомлення про необхідність вибору теми. Встановлюється слухач натискань для макету game1Layout, який перевіряє, чи достатньо балів для відкриття гри, і якщо так, запускає активність гри, в іншому випадку відображається повідомлення про недостатність балів. Метод showAlertDialog відображає інформаційне діалогове вікно з повідомленням про автора та джерела інформації. Метод getTotalResultFromFile читає результати з файлу "saving_result.txt" і повертає загальну суму балів, враховуючи обмеження на максимальні та мінімальні значення.

Клас DmdActivityMenu в мобільному додатку для перевірки знань з першої допомоги надає користувачам інтерфейс для вибору теми вікторини. Після запуску активності встановлюється макет activity_dmd_menu. Ініціалізуються елементи інтерфейсу, включаючи кнопки, лінійні макети та звукові ефекти за допомогою об'єктів MediaPlayer. Змінна selectedTopicDMD зберігає вибрану тему для вікторини. Звукові ефекти mediaPlayerNext, mediaPlayerFail, mediaPlayerOption, mediaPlayerTrue та mediaPlayerTrue_2 використовуються для покращення взаємодії користувача з додатком.

Кнопка backBtn має слухача натискань, який відтворює звук і повертає користувача на головний екран, відкриваючи активність MainActivity. Для кожного лінійного макету (Stop_bleeding, Bone_fracture, Poisoning, Reanimation) встановлюються слухачі натискань, які змінюють вибрану тему, змінюють фоновий ресурс для візуального позначення вибору та відтворюють звук. Наприклад, при натисканні на Stop_bleeding, змінна selectedTopicDMD встановлюється на "Stop_bleeding", змінюється фоновий ресурс макету, щоб показати його вибір, і відтворюється звук.

Кнопка qizStartDMD запускає вікторину. При натисканні перевіряється, чи вибрано тему вікторини. Якщо тему не вибрано, відтворюється звук і відображається повідомлення "Оберіть вікторину!". Якщо тему вибрано, відтворюється звук, створюється намір для запуску активності QuizActivity, передається вибрана тема за допомогою методу putExtra, і запускається нова активність.

Метод onBackPressed перевизначається для обробки події натискання кнопки назад, що забезпечує повернення користувача на головний екран, відкриваючи активність MainActivity і завершуючи поточну активність.

Клас QuizActivity у мобільному додатку відповідає за проведення вікторини з обраної теми першої допомоги. Активність містить графічний інтерфейс, який включає текстові поля для запитань та варіантів відповідей, кнопки для вибору варіантів, таймер, а також звукові ефекти для покращення користувацького досвіду. При створенні активності в методі onCreate встановлюється макет activity_quiz. Ініціалізуються елементи інтерфейсу, такі як кнопки, текстові поля, таймер, звукові ефекти, а також список запитань questionsList, отриманий із класу QuestionsBank на основі обраної теми. Таймер запускається методом startTimer, який відраховує час, поки не закінчиться вікторина. При цьому, якщо таймер доходить до нуля, вікторина завершується, і користувач повертається на головний екран. Метод stopTimer зупиняє таймер.

Для кожного з варіантів відповідей (option1, option2, option3, option4) встановлюються слухачі натискань. Після вибору варіанту викликається метод revealAnswer, який підсвічує правильний варіант зеленим кольором, та метод soudForAnswerOption, який відтворює відповідний звук залежно від правильності вибору користувача. Кнопка nextBtn запускає метод changeNextQuestion, який перемикає на наступне запитання, або завершує вікторину, якщо всі запитання були переглянуті. При завершенні вікторини відкривається активність QuizResults, яка показує результати користувача.

Метод revealAnswer підсвічує правильний варіант відповіді зеленим кольором, змінюючи фоновий ресурс відповідної кнопки. Метод soudForAnswerOption відтворює відповідний звук на основі правильності обраного варіанту. Метод getCorrectAnswers підраховує кількість правильних відповідей користувача, а метод getInCorrectAnswers – кількість неправильних. Метод onBackPressed перевизначається для зупинки таймера та повернення на головний екран при натисканні кнопки назад.

Клас QuestionsList служить для зберігання даних про питання вікторини, включаючи самі питання, варіанти відповідей, правильну відповідь, обрану користувачем відповідь і зображення, пов'язане з питанням. Він містить такі приватні поля: option1, option2, option3, option4, question, answer, userSelectedAnswer і questionImage. Конструктор класу приймає ці параметри та перевіряє, щоб жоден з них не був null, кидаючи IllegalArgumentException у разі порушення цієї умови. Геттери забезпечують доступ до всіх полів, а сеттер setUserSelectedAnswer дозволяє встановлювати відповідь, обрану користувачем. Поле questionImage зберігає ідентифікатор ресурсу зображення, пов'язаного з питанням.

Клас QuestionBank визначає дві статичні методи regulatorQuestions() і road_SignsQuestions(), які повертають список об'єктів типу QuestionsList. Кожен об'єкт QuestionsList представляє собою питання із варіантами відповідей та зображенням, пов'язаним із цим питанням. Метод regulatorQuestions() створює список питань, пов'язаних із сигналізацією регулятора дорожнього руху, в той

час як метод road_SignsQuestions() генерує список питань, пов'язаних із дорожніми знаками. Кожен об'єкт QuestionsList ініціалізується з конкретними питаннями, варіантами відповідей та зображенням, яке використовується для графічного представлення питання. Після створення об'єктів питань вони додаються до відповідного списку питань (questionsList), який потім повертається як результат відповідного методу.

Метод getQuestions(String selectedTopicName) використовується для отримання списку питань залежно від переданої теми selectedTopicName. Він викликає відповідний метод (наприклад, regulatorQuestions() або road_SignsQuestions()) на основі переданої теми і повертає список питань з вибраної теми. Якщо передана тема не відповідає жодній з визначених умов (case), метод повертає список питань про дорожні знаки за замовчуванням.

Узагальнюючи, цей код створює і повертає списки питань з двох різних тем: сигналізація регулятора руху і дорожні знаки, і використовується для підготовки даних для тестування знань з правил дорожнього руху.

Цей код визначає активіті TheoryMenyActivity_DMD, яке є підкласом AppCompatActivity і відповідає за відображення меню теоретичних матеріалів з різних тем. У методі onCreate() відбувається ініціалізація елементів інтерфейсу, таких як кнопки backBtn і різні LinearLayout, які відповідають за різні теми (наприклад, "Stop_bleeding", "Bone_fracture", тощо).

Кожен LinearLayout налаштовується з власним обробником подій OnClickListener, який відкриває відео на YouTube за допомогою методу openYouTubeVideo(), передаючи URL відповідного відео для кожної теми. Наприклад, при кліку на Stop_bleeding, відкривається відео за URL "https://www.youtube.com/watch?v=7yz8CKJ6_34". Метод openYouTubeVideo() створює новий Intent, щоб відкрити відео на YouTube за допомогою офіційного додатку YouTube. URL передається через Uri.parse(url), і Intent налаштовується для відкриття додатку YouTube. Крім того, в активності присутня кнопка backBtn, яка при кліку закриває поточну активність за допомогою методу finish().

3.3 Розробка модуля гри

Гра може привертати увагу користувачів та зробити застосунок більш привабливим. Графіка, звукові ефекти та ігровий процес можуть зробити користування застосунком більш приємним і захоплюючим. Гра може стати додатковим мотиватором для користувачів використовувати застосунок. Це може підвищити активність користувачів та збільшити тривалість сесій в застосунку. Гра може надати користувачам додаткові можливості для взаємодії з застосунком. Наприклад, користувачі можуть отримувати бали або нагороди за досягнення в грі, що може стимулювати їх використання застосунку. Додавання гри може зробити застосунок більш унікальним та відмінним від конкурентів. Це може залучити нових користувачів та збільшити конкурентоспроможність застосунку на ринку.

Отже, додавання гри до застосунку може бути вигідним кроком для покращення його успішності та привабливості для користувачів.

У методі onCreate() розміщено код для ініціалізації основних компонентів інтерфейсу гри. Це включає в себе налаштування кнопки запуску гри (startBtn), контейнера для гри (rootLayout), текстового поля для відображення результату (score) та самої гри (mGameView). Також створюються екземпляри медіаплеєрів для відтворення звукових ефектів.

Обробник кнопки старту гри: Коли користувач натискає кнопку старту гри (startBtn), викликається обробник подій, де спочатку відтворюється звук запуску автомобіля (mediaPlayerStartCar). Потім фонове зображення дороги встановлюється для mGameView, і сама гра додається до rootLayout, щоб користувач міг почати гру.

Після закінчення гри метод closeGame() викликається із класу GameTask, який реалізовується в MainGameActivity. У цьому методі відображається результат гри у текстовому полі score, відтворюється звук фінішу автомобіля (mediaPlayerFinishCar). Потім кнопка startBtn знову стає видимою з написом "Головне меню", і при натисканні на неї викликається головне меню гри (MainActivity), щоб гравець міг вибрати інші можливості або почати гру знову.

Цей підхід робить гру користувачезависимою та додає взаємодію з користувачем, забезпечуючи гарне візуальне та звукове враження.

Клас MainGameActivity є центральною частиною ігрової програми на платформі Android. Він відповідає за управління відображенням гри та взаємодією з користувачем через інтерфейс користувацького інтерфейсу (UI).

Activity Lifecycle: Клас розширює AppCompatActivity, що дає змогу використовувати всі можливості життєвого циклу активності Android, такі як onCreate(), onStart(), onResume(), і так далі. UI Елементи: В активності є різні елементи UI, такі як кнопка startBtn, текстове поле score, і контейнер rootLayout, який є LinearLayout. Ці елементи дозволяють відтворювати інтерактивність ігрового інтерфейсу для користувача. GameView: Використовується екземпляр GameView, який вставляється в rootLayout.

GameView є власним класом, що відповідає за відображення гри на екрані, може містити логіку гри і взаємодію з користувачем через введення дотику екрана або інші події. Звукові ефекти: Використовуються звукові ефекти з використанням MediaPlayer для створення аудіо з файлів ресурсів raw. Звуки відтворюються під час різних подій, таких як початок гри (mediaPlayerStartCar) і закінчення гри (mediaPlayerFinishCar). Після початку гри за допомогою кнопки startBtn, інтерфейс користувача змінюється, щоб приховати кнопку startBtn і текстове поле score.

Після завершення гри метод closeGame() відновлює видимість цих елементів, відображаючи результати гри і пропонуючи можливість повернутися до головного меню. Для переходу до інших активностей використовується клас Intent, що ініціює перехід до MainActivity, який є основним екраном додатка. Цей клас є основною частиною ігрового додатка, яка керує відображенням гри, взаємодіє з користувачем і контролює аудіо ефекти під час гри.

Клас GameView відповідає за відображення ігрового простору і взаємодію з користувачем у грі на платформі Android.

Ініціалізація і налаштування: Конструктор класу отримує контекст додатку (Context) і інтерфейс GameTask, який використовується для взаємодії з основною

активністю гри. У init блоку ініціалізуються різні змінні, такі як пензлик для малювання (myPaint), швидкість гри (speed), лічильник часу (time), інші автомобілі (otherCars), монети і так далі. Також налаштовується MediaPlayer для відтворення звуку при зборі монет (mediaPlayerPointPlus).

Малювання елементів на екрані: Метод onDraw(canvas: Canvas) викликається автоматично системою Android для малювання кожного кадру гри. В цьому методі реалізовано малювання гравця, інших автомобілів, монет, а також виведення інформації про результати гри (результат, рекорд, швидкість і кількість монет). Взаємодія з користувачем: Метод onTouchEvent(event: MotionEvent?) обробляє події торкання екрана. В залежності від напрямку та місця дотику користувача, переміщується машина гравця вліво або вправо. Це впливає на підрахунок балів.

Метод createCoinIfNeeded() створює нові монети на полі гри з певним інтервалом часу. Це залежить від часу, який пройшов з моменту створення останньої монети. Коли гравець збирає монету, відтворюється звук за допомогою MediaPlayer. Методи increaseSpeedIfNeeded() і startSpeedIncreaseTimer() відповідають за збільшення швидкості гри з плином часу. Це реалізовано за допомогою Handler і Runnable, які викликаються з певним інтервалом.

При досягненні нового рекорду в грі, викликається збереження у внутрішньому сховищі даних додатку за допомогою SharedPreferences. Цей клас забезпечує повністю функціональне ігрове середовище на Android, з можливістю управління гравцем, обробки колізій, генерації інших об'єктів на полі гри, відтворення звукових ефектів і відображення результатів гри.

Інтерфейс використовується для взаємодії між головним класом гри (MainGameActivity) і класом, який відповідає за логіку гри. Взаємодія між цими компонентами забезпечує коректне управління ігровим процесом та обробку результатів гри. Ініціювання методу closeGame(mScore: Int, record: Int) є ключовим моментом у цьому процесі.

Метод closeGame приймає два параметри: mScore, який представляє поточний результат гравця, і record, який представляє рекорд гравця. Основною

функцією цього методу є передача інформації про результати гри від класу, що управляє логікою гри, до головного класу. Це дозволяє головному класу оновити інтерфейс користувача, зберегти результати та виконати інші необхідні дії, такі як показ підсумкових повідомлень чи запуск анімацій.

Клас, який реалізує інтерфейс GameTask, повинен надати реалізацію цього методу для визначення дій, які повинні виконатися після закінчення гри. Реалізація може включати оновлення інтерфейсу для відображення підсумкового екрану з результатами гри, збереження результатів у базу даних або інші сховища, аналіз прогресу гравця для визначення його досягнень, а також підготовку до нової гри шляхом очищення тимчасових даних і налаштування початкових значень.

Таким чином, впровадження інтерфейсу GameTask та методу closeGame дозволяє ефективно управляти ігровим процесом, забезпечуючи інтерактивну взаємодію між логікою гри та головним інтерфейсом користувача. Це сприяє створенню цілісного і плавного ігрового досвіду, де всі дії та події чітко синхронізовані і відповідним чином обробляються.

Інтерфейс GameTask, який використовується для взаємодії між класами у вашому Android додатку. Назва і пакет: Інтерфейс називається GameTask і знаходиться у пакеті com.example.pdr_ukraine_2024.

closeGame(mScore: Int, record: Int): Це функція без тіла, яка призначена для закриття гри або іншої дії, пов'язаної з закінченням гри. Вона очікує два параметри типу Int: mScore (поточний результат гравця) і record (рекорд гравця). Реалізація цієї функції буде виконана в класах, що імплементують цей інтерфейс, для визначення конкретної поведінки при закритті гри.

Використання: Цей інтерфейс дозволяє створювати взаємодію між компонентами додатку, наприклад, між GameView (який представляє ігрове поле) і MainGameActivity (основною активністю гри). Коли гравець закінчує гру в GameView, він може викликати метод closeGame() з необхідними параметрами, щоб MainGameActivity могла відповідно обробити цю подію (наприклад, відобразити результати гри або перейти в головне меню).

3.4 Висновки

У третьому розділі було проведено ретельний аналіз та обґрунтування вибору мови програмування та технологій для розробки мобільного застосунку для перевірки знань з правил дорожнього руху. Основні аспекти, які було розглянуто та виконано, включають: Java: Було обрано через її надійність, стабільність та багатий набір бібліотек і фреймворків, які сприяють ефективній розробці Android-додатків. Java також забезпечує високу продуктивність та безпеку, що є важливим для створення стабільного та безпечного застосунку. Kotlin: Доповнює Java сучасними можливостями, такими як короткий синтаксис, безпека типів даних та нульова безпека.

Використання Android Studio: Було обрано Android Studio як основне інтегроване середовище розробки (IDE) для створення додатків на платформі Android. Android Studio підтримує як Java, так і Kotlin, надаючи всі необхідні інструменти для розробки, налагодження та тестування додатків, що значно полегшує процес створення застосунку.

Модуль вікторини дозволяє користувачам проходити вікторини з правил дорожнього руху. Використано графічний інтерфейс для відображення головного меню, вибору тем вікторини та відтворення звукових ефектів під час взаємодії з інтерфейсом. Головне меню має інтуїтивно зрозумілий дизайн, що дозволяє користувачам легко навігувати між різними розділами вікторини. Користувачі можуть обирати з різних тем, що охоплюють різні аспекти правил дорожнього руху. Відповіді користувача зчитуються та обробляються для визначення правильності обраних варіантів. Результати вікторин зберігаються у файлі для аналізу прогресу користувача.

Модуль гри створено для залучення користувачів та підвищення їхньої мотивації до вивчення правил дорожнього руху. Гра включає яскравий та привабливий графічний інтерфейс, який робить гру цікавою та доступною для користувачів різного віку. Інтеграція звукових ефектів створює реалістичне ігрове середовище та підвищує залучення користувачів. Гра також надає якісний освітній контент, що робить процес навчання більш цікавим та ефективним.

4 ТЕСТУВАННЯ ПРОГРАМИ

4.1 Аналіз методів тестування програмного забезпечення

Тестування програмного забезпечення (ТПЗ) – це критичний етап, що включає в себе перевірку та оцінку якості програми для виявлення помилок, дефектів і проблем. Його основною метою є забезпечення коректної роботи програмного забезпечення, відповідності вимогам та очікуванням користувачів, а також гарантування надійності, безпеки й ефективності його функціонування. Тестування є невід'ємною частиною процесу розробки програмного забезпечення і сприяє виявленню та усуненню проблем перед випуском програми. Процес тестування програмного забезпечення проходить через кілька ключових етапів. Починається все з юніт-тестування, що перевіряє окремі компоненти програми. Потім виконується інтеграційне тестування, яке перевіряє взаємодію цих компонентів. Наступним етапом є системне тестування, де перевіряється робота всієї системи з використанням різних сценаріїв. На завершальному етапі проводиться приймальне тестування для перевірки відповідності програми вимогам та її готовності до впровадження. Ці етапи забезпечують якість та надійність програмного забезпечення перед його випуском.

Для досягнення цих цілей використовуються різні методи тестування:

1. Тестування продуктивності (Performance Testing) – це важливий аспект нефункціонального тестування, спрямований на оцінку реакції системи під навантаженням у різних сценаріях використання. Мета цього тестування полягає у перевірці швидкодії, працездатності та стабільності системи. Параметри оцінки включають час відгуку, надійність, використання ресурсів та масштабованість. Тестування продуктивності допомагає виявити потенційні вразливості та недоліки у програмному забезпеченні, щоб запобігти їхньому впливу на систему під час реального використання.

2. Функціональне тестування (Functional Testing) аналізує функціональні характеристики програмного забезпечення та перевіряє відповідність реальної роботи реалізованих функцій очікуваній поведінці згідно зі специфікацією та бізнес-вимогами. Це тестування імітує фактичне використання системи і проводиться на різних рівнях тестування, включаючи компонентний, інтеграційний, системний та приймальний. Функціональні тести базуються на вимогах, функціональних специфікаціях або сценаріях використання системи (Use case).

3. Тестування безпеки (Security Testing) – процес, спрямований на оцінку і перевірку системи з точки зору її стійкості до зовнішніх атак, забезпечення цілісності, доступності, конфіденційності та надійності програмного забезпечення. В рамках цього тестування перевіряються автентифікація, авторизація, доступність, конфіденційність та безвідмовність програми під впливом різних вторгнень і атак.

4. Тестування білої скриньки (White Box Testing) – метод, де тестувальники аналізують внутрішню структуру програмного забезпечення, розуміють його внутрішні механізми, перевіряють вірність виконання функцій, логіку програми та її відповідність вимогам.

5. Тестування чорної скриньки (Black Box Testing) – метод, при якому тестувальники не мають доступу до внутрішнього коду програми і розглядають її як "чорну скриньку". Вони аналізують програму, фокусуючись лише на вхідних даних, обробці та вихідних результатах, щоб переконатися, що вона працює відповідно до специфікацій та вимог.

Після детального аналізу різних методів тестування програмного забезпечення було вирішено використовувати метод тестування чорної скриньки. Цей вибір обумовлений тим, що тестування чорної скриньки надає більш об'єктивні результати і дозволяє виявити потенційні проблеми, які можуть виникнути при фактичному використанні програмного забезпечення в реальних умовах.

4.2 Тестування системи

Тестування мобільної системи на Android – це процес перевірки функціональності, безпеки та коректності роботи програм на мобільній платформі від Android. Одним з найважливіших критеріїв успішного тестування є перевірка сумісності програми з різними версіями операційної системи.

Для початку тестування необхідно встановити тестову версію програми на мобільний пристрій з операційною системою Android, яка буде використовуватись під час тестування. Далі проводяться функціональні тести для перевірки правильності роботи програми. Ці тести включають перевірку введення та виведення даних, реакцію програми на негативні вхідні дані, а також відповідність бізнес логіки. Такі тести дозволяють переконатися, що програма функціонує як очікується в різних сценаріях використання.

Тестування програмного забезпечення для мобільної платформи Android - це складний та важливий етап розробки, що включає різноманітні аспекти. Одним із ключових аспектів є тести на безпеку. Тут використовуються спеціальні інструменти для моделювання можливих атак, які можуть загрожувати програмі. Такі тести включають перевірку на проникнення, тобто спроби незаконного доступу до системи, а також тести на збереження конфіденційності, які перевіряють, як програма впорається з обробкою та захистом конфіденційної інформації користувача. Додатково проводяться тести на стійкість до зламу, щоб переконатися, що програма може витримати можливі атаки з боку зловмисників.

Окрім безпеки, важливо також оцінити продуктивність програми. Тести на швидкість роботи дозволяють визначити, наскільки ефективно програма працює, реагуючи на введення користувача та виконуючи різноманітні операції. Крім того, тести на використання ресурсів мобільного пристрою допомагають виявити, як програма впливає на батарею, пам'ять та інші ресурси пристрою. Це є важливим аспектом для користувачів, оскільки вплив на ресурси може значно вплинути на зручність використання додатка та загальний досвід користувача.

Додатково, у тестуванні враховуються аспекти інтерфейсу користувача та зручності використання. Це включає тести на зручність навігації, зрозумілість інтерфейсу, а також відповідність дизайну вимогам платформи Android. Такий підхід забезпечує, що програма не лише функціонує правильно, але й є приємною та інтуїтивно зрозумілою для користувачів.

Загалом, тестування мобільної системи на Android є багатогранним процесом, який включає функціональні, безпекові, продуктивні та користувацькі аспекти. Це забезпечує створення надійного, безпечного та зручного для користувачів програмного забезпечення, яке відповідає високим стандартам якості та задовольняє потреби сучасних користувачів. Тестування продуктивності системи зображено на рисунку 4.1.

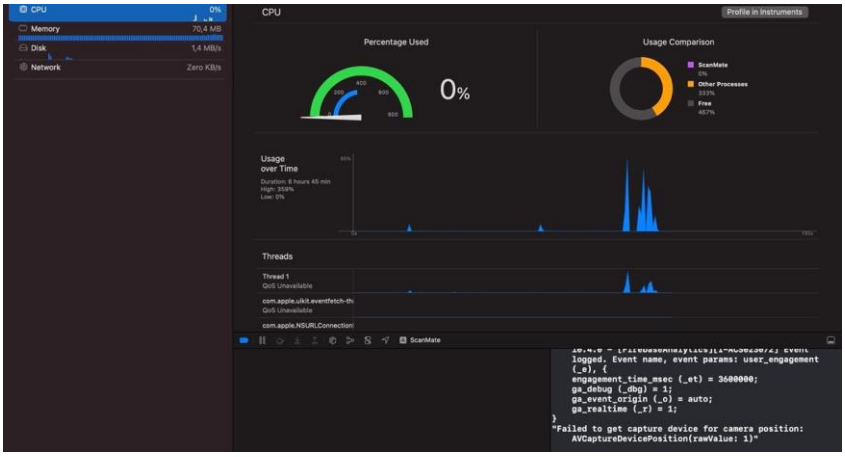

Рисунок 4.1 – Тестування продуктивності системи

Наступним кроком буде перевірка головного екрану, зокрема функціональності кнопки «Почати». Перш за все, необхідно переконатися, що кнопка «Почати» реагує на натискання користувачем. Якщо користувач натискає кнопку «Почати», але не вибрав тему вікторини, система повинна відреагувати належним чином. У такому випадку повинно з’явитися спливаюче повідомлення з текстом «Оберіть вікторину!». Це повідомлення повинно бути чітко видимим і відображати відповідний інструктивний контекст для користувача. Далі перевіряємо, що повідомлення відображається у правильному місці на екрані і має відповідний дизайн згідно з вимогами до користувацького інтерфейсу.

Наприклад, повідомлення може бути розташоване по центру екрану, на світлому фоні, з великим шрифтом і з помітною кнопкою для закриття повідомлення або повторного вибору вікторини. На рисунку показано приклад, як саме має виглядати повідомлення «Оберіть вікторину!».

Воно повинно з’являтися негайно після натискання кнопки «Почати», якщо тема вікторини не вибрана. Цей крок гарантує, що користувачі отримають чітке сповіщення про необхідність вибору вікторини перед початком, що значно покращує зручність використання та запобігає можливим непорозумінням. Крім того, варто перевірити, як система поводиться при багаторазовому натисканні кнопки «Почати» без вибору вікторини. Повідомлення повинно з'являтися кожного разу, щоб забезпечити користувачеві достатньо інформації про необхідні дії зображено на рисунку 4.2.

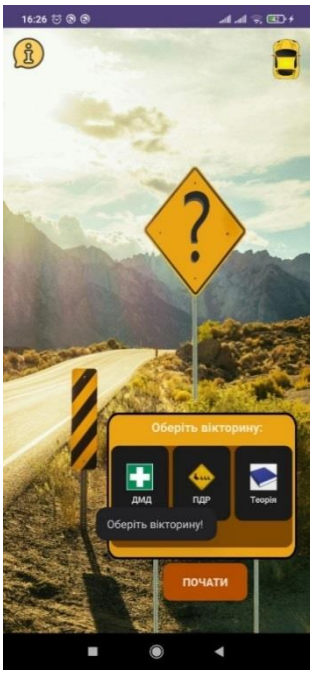

Рисунок 4.2 – Початок вікторини без обрання її теми

Після натискання кнопки «Почати» без вибору теми вікторини, на екрані має з’явитися сповіщення про необхідність вибору теми. Це сповіщення повинно бути чітко видимим. У разі відсутності вибору теми, подальші дії користувача повинні бути заблоковані до моменту виконання цієї умови. Після завершення

тестування аналізуємо результати. Якщо всі перевірки пройдено успішно, то функціональність кнопки «Почати» і спливаючого повідомлення відповідає вимогам. Це означає, що користувачі отримують належні інструкції і можуть легко орієнтуватися у процесі використання програми.

Після натискання кнопки «Почати» без достатньої кількості балів, на екрані має з’явитися сповіщення про необхідність мати більше 30 балів. Це сповіщення повинно бути чітко видимим. У разі недостатньої кількості балів, подальші дії користувача повинні бути заблоковані до моменту виконання цієї умови. Зображено на рисунку 4.3

Рисунок 4.3 – Початок гри без потрібної кількості балів.

Мета тестування: переконатися, що після обрання теми вікторини та натискання кнопки «Почати», користувач успішно переходить на сторінку меню вікторин. Після вибору теми вікторини та натискання кнопки «Почати», користувач повинен бути успішно перенаправлений на сторінку меню вікторин. Це означає, що всі налаштування і вибір теми були виконані правильно, а система коректно обробила дію користувача.

Проведення тестування

– Вибрати одну з доступних тем вікторини зі списку.

– Після обрання теми натиснути кнопку «Почати».

– Система має перенаправити користувача на сторінку меню вікторин.

Результат перенаправлення відображений на рисунку, що підтверджує успішне виконання цього кроку. Таким чином, ретельне тестування процесу обрання теми вікторини та перенаправлення на сторінку меню вікторин є критично важливим етапом у забезпеченні якості програмного забезпечення.

Результат успішного переходу на сторінку меню вікторин зображено на рисунку 4.4

Рисунок 4.4 – Результат успішного переходу на сторінку меню вікторин

Перевірка повідомлення:

– Переконатися, що після натискання кнопки «Next» без обрання відповіді система відображає повідомлення «Оберіть варіант відповіді».

– Повідомлення повинно бути чітко видимим і зрозумілим для користувача.

Переконатися, що кнопка «Next» не дозволяє перейти до наступного питання без обрання відповіді. При натисканні кнопки «Next» без вибору

варіанту відповіді користувач отримує повідомлення «Оберіть варіант відповіді», що відображено на рисунку 4.5. Це підтверджує, що система правильно реагує на дії користувача і запобігає можливим помилкам. Повторити тест для різних питань вікторини, щоб переконатися у стабільності функціоналу. Перевірити, що система кожного разу правильно обробляє натискання кнопки «Next» без обрання відповіді і відображає відповідне повідомлення.

Переконатися, що повідомлення «Оберіть варіант відповіді» відображається у відповідному стилі, який не порушує дизайн і загальну естетику програми. Таким чином, ретельне тестування процесу переходу між питаннями вікторини без обрання відповіді є критично важливим етапом у забезпеченні якості програмного забезпечення.Зображено на рисунку 4.5

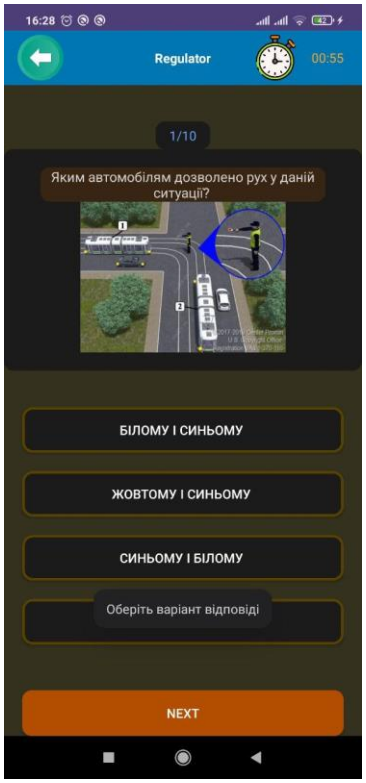

Рисунок 4.5 – Вікно вікторини

4.3 Розробка інструкції користувача

Для успішного використання додатка на пристроях з операційною системою Android рекомендується мати пристрій, який відповідає встановленим технічним вимогам. Мінімальна конфігурація пристрою, яка дозволить

користуватися основними функціями додатка, наведена у таблиці 4.1. Проте, для оптимальної продуктивності та забезпечення швидкодії та стабільності додатка, рекомендується використовувати пристрій, що відповідає рекомендованій конфігурації, представленій у таблиці 4.2.

Таблиця 4.1 – Мінімальна конфігурація:

Тип процесора	Qualcomm Snapdragon 665
Об’єм оперативної пам’яті	2 ГБ для 64-разрядної системи
Місце на жорсткому диску	1 ГБ
Операційна система	Android

Таблиця 4.2 – Рекомендована конфігурація:

Тип процесора	Qualcomm Snapdragon 665
Об’єм оперативної пам’яті	4 ГБ для 64-разрядної системи
Місце на жорсткому диску	4 ГБ
Операційна система	Android

Після запуску платформи користувач повинен обрати тему вікторини або перейти у вікно "Теорія", де можна покращити теоретичні знання. Далі, йому надається можливість обрати певну вікторину. Обравши вікторину, йому необхідно відповідати на запитання. Після проходження вікторини користувач переходить у вікно результатів, де відображається результат проходження вікторини та загальний бал.

Загальний бал вираховується наступним чином: від кількості правильних відповідей віднімається кількість неправильних, і якщо результат більше нуля, то він додається до загального результату. Якщо користувач отримав більше

тридцяти балів, то йому відкриваються ігри. Доки користувач не набере потрібну кількість балів, гра буде зачиненою.

Отже, після входу в систему запускається модуль меню, який відображає на екрані вікторини та іконку "Теорія" у вигляді статті. Користувач також може переглянути теорію у вигляді відео, якщо є під’єднання до інтернету, де детально описана теорія. Також є можливість переглянути гру, якщо користувач набрав потрібну кількість балів. Після завершення гри або вікторини, користувач може зберегти свій результат або поділитися ним у соціальних мережах Меню гри зображено на рисунку 4.6.

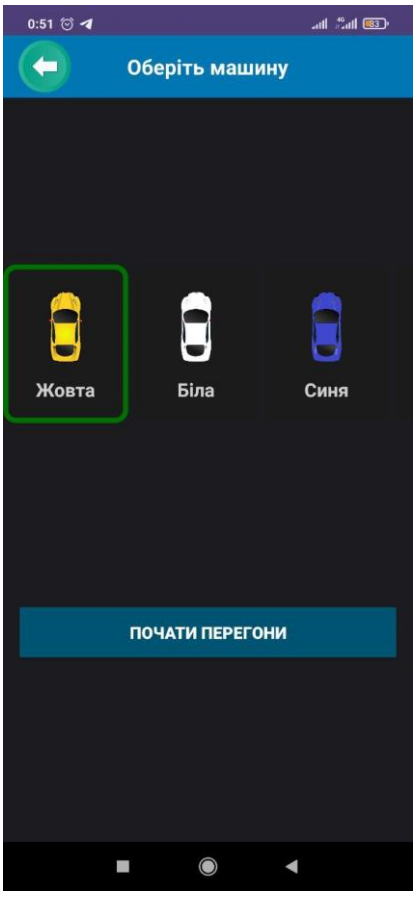

Рисунок 4.6 – Екран «Меню гри»

У процесі тестування гри було перевірено такі функції. Взаємодія: гра надає користувачам додаткові можливості для взаємодії із застосунком, що підвищує тривалість сесій та активність користувачів. Було протестовано всі інтерактивні елементи гри, включаючи кнопки, меню та можливості взаємодії з

ігровим середовищем. Покращення користувацького досвіду: графіка та звукові ефекти роблять користування додатком приємним та захоплюючим, створюючи позитивні враження. Тестування включало оцінку якості графіки, звукових ефектів та загальної естетики гри, щоб забезпечити приємний користувацький досвід. Гру перегони зображено на рисунку 4.7.

Рисунок 4.7 – Екран «Гра перегони»

Було також враховано та протестовано можливі недоліки, зокрема: складність розробки: розробка та інтеграція гри вимагають додаткових ресурсів, включаючи час та технічні знання, що може ускладнити процес розробки. Було оцінено ресурси, необхідні для створення та інтеграції гри, та перевірено їхню достатність. Витрати: створення якісної гри може потребувати додаткових фінансових витрат на графіку, звук та тестування, що збільшує загальну вартість проекту.

Було проаналізовано витрати, пов'язані з розробкою гри, та розроблено бюджет для їхнього покриття. Технічні проблеми: інтеграція ігрових елементів

може призвести до появи технічних проблем, які можуть вплинути на стабільність та продуктивність застосунку. Було проведено ретельне тестування на виявлення можливих технічних проблем та їхнє усунення. Відволікання: ігровий елемент може відволікати користувачів від основної мети додатку – навчання правилам дорожнього руху, що може негативно позначитись на їхній концентрації. Було оцінено вплив ігрових елементів на концентрацію користувачів та внесено корективи для мінімізації відволікаючих факторів.

Екран «Результат гри» зображено на рисунку 4.8.

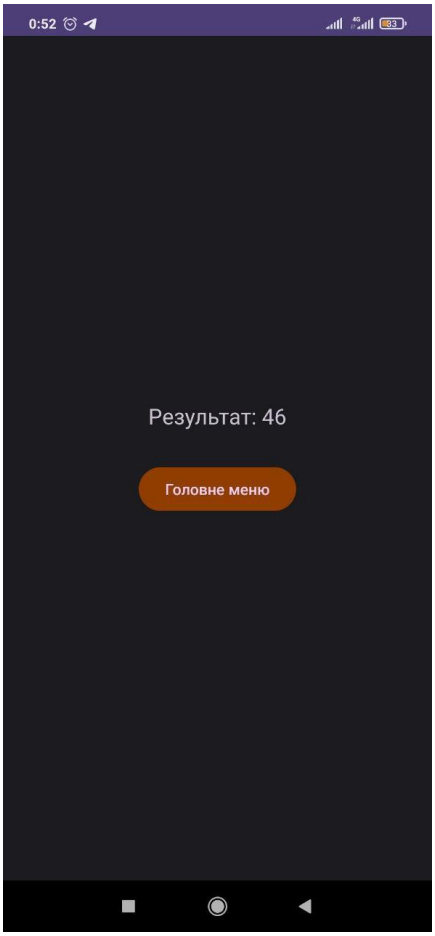

Рисунок 4.8 – Екран «Результат гри»

Інтеграція гри в мобільний додаток для перевірки знань правил дорожнього руху може значно покращити його привабливість та ефективність. Гра підвищує мотивацію користувачів, збільшує тривалість сесій та створює додаткові можливості для взаємодії із застосунком. Загалом, додавання гри може

бути вигідним кроком для підвищення конкурентоспроможності та успішності додатку на ринку, забезпечуючи користувачам захоплюючий та ефективний інструмент для вивчення правил дорожнього руху.

Впровадження гейміфікації в освітні мобільні додатки дозволяє створити більш інтерактивне та залучаюче середовище для користувачів. Ігрові елементи сприяють підвищенню зацікавленості, оскільки користувачі мають можливість отримувати винагороди за успіхи та досягнення. Це не тільки робить навчання цікавішим, але й стимулює повторення та закріплення матеріалу.

Крім того, гра допомагає користувачам ефективніше запам'ятовувати інформацію та застосовувати отримані знання на практиці. Інтерактивні сценарії та завдання, які включаються в ігровий процес, роблять навчання більш реалістичним та практичним, що особливо важливо для таких предметів, як правила дорожнього руху. Таким чином, гра забезпечує більш глибоке розуміння матеріалу та підвищує готовність користувачів до реальних ситуацій на дорозі.

У підсумку, інтеграція гри в мобільний додаток не лише покращує користувацький досвід, але й робить навчання більш ефективним та цікавим, що є ключовим фактором для успіху додатку на ринку.

4.4 Висновки

У четвертому розділі було проведено тестування наступних функціональностей мобільного застосунку для перевірки знань з правил дорожнього руху: Запуск платформи та вибір теми вікторини або перехід до вікна теорії. Вибір певної вікторини. Відповіді на запитання вікторини. Перехід до вікна результату після проходження вікторини, де відображається результат та загальний бал. Розрахунок загального балу на основі правильних та неправильних відповідей. Відкриття доступу до ігор, якщо користувач набрав більше тридцяти балів. Перегляд теорії у вигляді статті або відео з під'єднанням до Інтернету.

ВИСНОВКИ

У бакалаврській дипломній роботі розроблено застосунок для перевірки знань з правил дорожнього руху. Було здійснено детальний аналіз предметної області онлайн-освіти, а також проведено порівняння існуючих програмних продуктів, що використовуються для тестування знань з правил дорожнього руху та їхнього практичного застосування. На основі цього порівняння було аргументовано доцільність створення власного програмного продукту для автоматизації процесу тестування. Також було визначено завдання для розробки цього програмного забезпечення.

У роботі описано методи та модель функціонування програми, створено загальний алгоритм її роботи, а також розроблено алгоритми для виявлення тестових випадків у файлах і папках та для генерації Java-звіту. Було також створено блок-схеми для цих алгоритмів.

Проведено аналіз можливих варіантів та обґрунтовано вибір інструментів для реалізації програмного продукту. На основі цього аналізу було обрано мову програмування Java та інтегроване середовище розробки Android Studio.

В процесі розробки обґрунтовано та реалізовано інтерфейс Java-звіту, описано програмну архітектуру та методи розробки програми. Також створено модулі для виявлення тестових випадків у файлах і папках та для генерації звіту.

Проаналізовано основні методи тестування програмного забезпечення та на основі цього аналізу обрано методику "black-box testing". Розроблено інструкцію для користувачів, а також проведено експериментальне порівняння ручного та автоматизованого методів тестування.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Гаджети для освіти [Електронний ресурс]. – Режим доступу до ресурсу: https://learning.ua/blog/201612/hadzhety-dlia-osvity/

2. Гейміфікація та ігрове навчання, у чому різниця? [Електронний ресурс]. – Режим доступу до ресурсу: https://www.futureschool.online

3. КЗВО«Вінницька академія безперервної освіти», 2023. – С. 68-71.

4. Розробка мобільних додатків: тенденції, які варто знати у 2023 [Електронний ресурс]. – Режим доступу до ресурсу: https://careers.easternpeak.com/blog/mobile-app-development-trends/

5. Android Applications and Their Categories [Електронний ресурс]. – Режим доступу до ресурсу: https://www.geeksforgeeks.org/android-applications-and- their-categories/

6. Коваленко О. О. Моделі гейміфікації в системах управління навчанням: монографія / О. О. Коваленко, Є. А. Паламарчук. – Вінниця : ВНТУ, 2023. – 85 с.

7. The effect of educational game design process on students' creativity Derman Bulut, Yavuz Samur & Zeynep Cömert Categories [Електронний ресурс]. – Режим доступу до ресурсу: https://slejournal.springeropen.com/articles/10.1186/s40561-022-00188-9

8. Розробка мобільних додатків і їх види [Електронний ресурс]. – Режим доступу до ресурсу: http://apeps.kpi.ua/rozrobka-mobilnykh-dodatkiv-i-yii-vidy

9. Model-View-ViewModel [Електронний ресурс]. – Режим доступу до ресурсу: https://www.wikidata.uk-ua.nina.az/Model-View-ViewModel.html

10.LiveData overview [Електронний ресурс]. – Режим доступу до ресурсу: https://developer.android.com/topic/libraries/architecture/livedata

11.View Binding in Android Jetpack [Електронний ресурс]. – Режим доступу до ресурсу: https://www.geeksforgeeks.org/view-binding-in-android-jetpack/

12.Reto Meier Professional Android 4 Application Development 3rd Edition / Reto Meier – Wrox, 2012. – 816 p.

13.MySQL Crash Course: A Hands-on Introduction to Database Development / Rick Silva // No Starch Press – May 23, 2023 – p. 352

14.Бази даних та СУБД. [Електронний ресурс]. Режим доступу до ресурсу: https://timeweb.com/ru/community/articles/bazy-dannyh-i-subd-1

15.Що таке FIREBASE? [Електронний ресурс]. – Режим доступу до ресурсу: https://avada-media.ua/ua/services/firebase/

16.Принцип індивідуалізації і колективності навчання [Електронний ресурс]. – Режим доступу до ресурсу: https://studfile.net/preview/5722650/page:11/

17.Westwood P. Inclusive and Adaptive Teaching: Meeting the Challenge of Diversity in the Classroom. Taylor & Francis Group, 2018. 128 p.

18.Why Every Organization Needs an Augmented Reality Strategy [Електронний ресурс] – Режим доступу до ресурсу: https://hbr.org/2017/11/why-every-organization-needs-an-augmented-reality-strategydas

19.The Past, Present, and Future of Virtual and Augmented Reality Research: A Network and Cluster Analysis of the Literature [Електронний ресурс] – Режим доступу до ресурсу: https:/ /www.frontiersin.org/ articles/10.3389/ fpsyg.2018.02086/full

20.Augmented Reality – The Past, The Present and The Future [Електронний ресурс] – Режим доступу до ресурсу: https://www.interaction-design.org/literature/article/augmented-reality-the-past-the-present-and-the-future

21.Why Augmented Reality Is One Of The Most Promising Experimental Technologies Of This Decade https://www.forbes.com/sites/ forbestechcouncil/2023/02/06/why-augmented-reality-is-one-of-the-most-pro mising-experimental-technologies-of-this-decade/?sh=4366e4fd3c85

22.Луценко Р. С. Використання ARKIT в освіті, охороні здоров'я та сфері розваг / Р. С. Луценко, Н. П. Бабюк // Матеріали LII Науково-технічної конференції підрозділів Вінницького національного технічного університету, Вінниця, 2023. [Електронний ресурс] – Режим доступу до ресурсу: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2023/paper/view/17464/14582

23.Augmented Reality Development Platforms [Електронний ресурс] – Режим доступу до ресурсу: https://www.trustradius.com/augmented-reality-development.

24.Best Industrial AR Platforms [Електронний ресурс] – Режим доступу до ресурсу: https://www.g2.com/categories/industrial-ar-platforms

25.Drag and Drop AR Platforms [Електронний ресурс] – Режим доступу до ресурсу: https://www.plugxr.com/augmented-reality/the-ultimate-ar-platform/OLX [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/OLX.

26.Довідник ПДР [Електронний ресурс] – Режим доступу до ресурсу: https://green-way.com.ua/uk

27.Автошкола «Світлофор» [Електронний ресурс] – Режим доступу до https://www.svetofor.zp.ua/uk/pravila-dorozhnogo-ruxu/yak-samostijno-vivchiti-pravila-dorozhnogo-ruxu/

28.Oracle Java Documentation [Електронний ресурс] – Режим доступу до ресурсу: https:// docs.oracle.com/en/java/

29.Домедична допомога [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki

30.Домедична допомога [Електронний ресурс] – Режим доступу до ресурсу: https://medicover.ua/blog/persha-domedychna-dopomoga-pry-nadzvychajnij-sytuaciji.html

ДОДАТКИ

67

Додаток А – Технічне завдання

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
д.т.н., професор
_____ О. Н. Романюк
«12» березня 2024 р.

Технічне завдання
на бакалаврську дипломну роботу «Розробка мобільного застосунку для тестування в ігровій формі знань з правил дорожнього руху» за спеціальністю 121 – Інженерія програмного забезпечення

Керівник бакалаврської дипломної роботи:
_____ к.т.н., доц. каф. ПЗ Ракитянська Г.Б.
«12» березня 2024 р.

Виконав:
_____ студент гр. 2ПІ-20б М. Ю Кривов'язюк
«12» березня 2024 р.

Вінниця – 2024 року

1. Найменування та галузь застосування

Бакалаврська дипломна робота: «Розробка мобільного застосунку для тестування в ігровій формі знань з правил дорожнього руху».

Галузь застосування – освіта.

2. Підстава для розробки

Підставою для виконання бакалаврської дипломної роботи (БДР) є індивідуальне завдання на БДР та наказ №80 від «11» березня 2024 р. ректора ВНТУ про закріплення тем БДР.

3. Мета та призначення розробки

Метою дослідження є підвищення якості освіти під час перевірки правил дорожнього руху.

4. Вихідні дані для проведення НДР

Перелік основних літературних джерел, на основі яких буде виконуватись БДР.

1. Романюк О. Н., Кривов`язюк М. Ю., Снігур А. В., Коваль Л.В., Михайлов П. І., Чехмейструк Р. Ю. Використання тривимірного моделювання для визначення масо-вагових характеристик людини по її антропометричним параметрам/ Прикладні питання математичного моделювання т. 4, № 2.1, 2021.- С.188-199.

2. Information Technology in Medical Diagnostics / ed. by W. Wójcik, A. Smolarz. CRC Press, 2017.

5. Технічні вимоги

Вихідні дані до роботи: кольоровий режим, Truecolor, розмір кординат, розмір екрану 2560-1440px, результати вимірювань антропометричних параметрів, з можливістю їх візуалізації на тривимірній моделі тіла людини.

6. Конструктивні вимоги

Графічна та текстова документація повинна відповідати діючим стандартам України.

7. Перелік технічної документації, що пред'являється по закінченню робіт:

1. Пояснювальна записка до БДР.
2. Технічне завдання.
3. Лістинги програми

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ

9. Стадії та етапи розробки:

№ п\п	Назва етапів бакалаврської дипломної роботи	Строк виконання етапів роботи
1	Аналіз стану розвитку навчальних вікторин для перевірки знань з правил дорожнього руху та постановка задач дослідження.	14.03.24 –22.03.24
2	Розробка модуль для створення вікторин	13.03.24 –22.03.24
3	Розробка алгоритмів роботи програмного забезпечення	25.03.24 –19.04.24
4	Розробка програмного забезпечення	22.04.24 –10.05.24
5	Тестування програмного забезпечення	13.05.24 – 24.05.24
6	Оформлення матеріалів до захисту БДР	27.05.24 – 30.05.24

10. Порядок контролю та прийняття

Виконання етапів бакалаврської дипломної роботи контролюється керівником згідно з графіком виконання роботи. Прийняття бакалаврської дипломної роботи здійснюється ДЕК, затвердженою зав. кафедрою згідно з графіком.

70

Додаток Б – Протокол перевірки на плагіат

ПРОТОКОЛ
ПЕРЕВІРКИ БАКАЛАВРСЬКОЇ ДИПЛОМНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Розробка мобільного застосунку для тестування в ігровій формі знань з правил дорожнього руху

Тип роботи: БДР

Підрозділ: кафедра програмного забезпечення, ФІТКІ

Науковий керівник: Ракитянська Г.Б.

Оригінальність	94.1%
Схожість	5.9%

Аналіз звіту подібності

▪ **Запозичення, виявлені у роботі, оформленні коректно і не містять ознак плагіату.**

☐ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цілісності роботи і відсутності самостійності її автора. Роботу направити на доопрацювання.

☐ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховання недобросовісних запозичень.

Особа, відповідальна за перевірку Черноволик Г.О.

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck

Автор роботи Кривов`язюк М.Ю

Керівник роботи Ракитянська Г.Б.

Додаток В – Лістинг

```
package com.example.pdr_ukraine_2024;

import android.app.AlertDialog;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.SharedPreferences;
import android.media.MediaPlayer;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.LinearLayout;
import android.widget.Toast;
import androidx.appcompat.app.AppCompatActivity;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;

public class MainActivity extends AppCompatActivity {

    private static final String PREFS_NAME = "MyPrefsFile";
    private static final String FIRST_RUN_KEY = "firstRun";

    MediaPlayer mediaPlayerNext, mediaPlayerOption, mediaPlayerTheory, mediaPlayerTrue_2;
    private String selectedTopic = "";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        final LinearLayout DMD = findViewById(R.id.DMDLayout);
        final LinearLayout PDR = findViewById(R.id.PDRLayout);
        final LinearLayout theoryMenu = findViewById(R.id.theoryLayoutMenu);
        final Button startQuiz = findViewById(R.id.startQiz);

        mediaPlayerNext = MediaPlayer.create(this, R.raw.sounds_1);
        mediaPlayerOption = MediaPlayer.create(this, R.raw.sounds_2);
        mediaPlayerTheory = MediaPlayer.create(this, R.raw.theory);
        mediaPlayerTrue_2 = MediaPlayer.create(this, R.raw.sounds_4);

        LinearLayout informationLayout = findViewById(R.id.information);
        informationLayout.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                showAlertDialog();
            }
        });

        SharedPreferences settings = getSharedPreferences(PREFS_NAME, 0);
```

```
boolean firstRun = settings.getBoolean(FIRST_RUN_KEY, true);

if (firstRun) {
    SharedPreferences.Editor editor = settings.edit();
    editor.putBoolean(FIRST_RUN_KEY, false);
    editor.apply();
}

DMD.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        mediaPlayerOption.start();
        selectedTopic = "DMD";
        DMD.setBackgroundResource(R.drawable.round_back_stroke10);
        PDR.setBackgroundResource(R.drawable.round_backa_white10);
        theoryMenu.setBackgroundResource(R.drawable.round_backa_white10);
    }
});

PDR.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        mediaPlayerOption.start();
        selectedTopic = "PDR";
        PDR.setBackgroundResource(R.drawable.round_back_stroke10);
        DMD.setBackgroundResource(R.drawable.round_backa_white10);
        theoryMenu.setBackgroundResource(R.drawable.round_backa_white10);
    }
});

theoryMenu.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        mediaPlayerTheory.start();
        selectedTopic = "theory";
        theoryMenu.setBackgroundResource(R.drawable.round_back_stroke10);
        DMD.setBackgroundResource(R.drawable.round_backa_white10);
        PDR.setBackgroundResource(R.drawable.round_backa_white10);
    }
});

startQuiz.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        if (selectedTopic.isEmpty()) {
            mediaPlayerNext.start();
            Toast.makeText(MainActivity.this, "Оберіть вікторину! ", Toast.LENGTH_SHORT).show();
        } else {
            Intent intent;
            if (selectedTopic.equals("DMD")) {
                intent = new Intent(MainActivity.this, DmdActivityMenu.class);
                mediaPlayerTrue_2.start();
            } else if (selectedTopic.equals("PDR")) {
```

```
                    mediaPlayerTrue_2.start();
                    intent = new Intent(MainActivity.this, PdrActivityMenu.class);
                } else if (selectedTopic.equals("theory")) {
                    mediaPlayerTrue_2.start();
                    intent = new Intent(MainActivity.this, TheoryActivity.class);
                } else {
                    // Виконати альтернативну дію, наприклад, відобразити повідомлення користувачеві
                    Toast.makeText(MainActivity.this, "Оберіть тему вікторини!", Toast.LENGTH_SHORT).show();
                    return;
                }
                startActivity(intent);
                finish();
            }
        }
    });

    // Додайте обробник подій для відкриття MainGameActivity при натисканні на елемент з ідентифікатором Game_1
    LinearLayout game1Layout = findViewById(R.id.Game_1);
    game1Layout.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            // Перевірка чи достатньо балів для відкриття гри
            if (getTotalResultFromFile() >= 30) {
                Intent intent = new Intent(MainActivity.this, MenuGameActivity.class);
                startActivity(intent);
            } else {
                // Повідомлення про недостатність балів
                Toast.makeText(MainActivity.this, "Для відкриття гри потрібно набрати більше 30 балів!",
Toast.LENGTH_SHORT).show();
            }
        }
    });
}

    // Метод для відображення діалогового вікна з інформацією
private void showAlertDialog() {
    AlertDialog.Builder builder = new AlertDialog.Builder(MainActivity.this);
    builder.setMessage("Виконав Кривов'язюк Максим 2ПІ-20б\nЗастосунок розроблено у навчальних цілях інформація взята з
відкритих джерел а саме:\nПерша домедична допомога: -----\nПравила дорожнього руху: -----\nПриємного користування!")
            .setPositiveButton("OK", new DialogInterface.OnClickListener() {
                public void onClick(DialogInterface dialog, int id) {
                    dialog.dismiss();
                }
            });
    AlertDialog dialog = builder.create();
    dialog.show();
}

// Метод для отримання загального результату з файлу "saving_result.txt"
private int getTotalResultFromFile() {
    int totalResultFromFile = 0;
    try {
        File file = new File(getFilesDir(), "saving_result.txt");
```

```
            if (file.exists()) {
                BufferedReader br = new BufferedReader(new FileReader(file));
                String line;
                while ((line = br.readLine()) != null) {
                    int result = Integer.parseInt(line);
                    if (result >= 101) {
                        result = 100;
                    } else if (result <= -1) {
                        result = 0;
                    }
                    totalResultFromFile += result;
                }
                br.close();
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
        return totalResultFromFile;
    }
} import android.content.Context
import android.graphics.Canvas
import android.graphics.Color
import android.graphics.Paint
import android.media.MediaPlayer
import android.os.Handler
import android.os.Looper
import android.view.MotionEvent
import android.view.View
import com.example.pdr_ukraine_2024.GameTask
import com.example.pdr_ukraine_2024.R
import kotlin.math.abs

class GameView(var c: Context, var gameTask: GameTask) : View(c) {
    private var myPaint: Paint? = null
    private var speed = 1
    private var time = 0
    private var score = 0
    private var myCarPosition = 0
    private var coinPosition = -1
    private var coinStartTime = 0L
    private var coinInitialY = 0
    private var otherCars = ArrayList<HashMap<String, Any>>()
    private var coins = 0

    lateinit var mediaPlayerPointPlus: MediaPlayer

    private var viewWidth = 0
    private var viewHeight = 0
    private var record = 0

    private var lastCoinCreationTime = 0L
    private val coinCreationInterval = 5000 // Interval for creating a new coin (in milliseconds)
    private val speedIncreaseInterval = 50000 // Interval for increasing speed (in milliseconds)
```

```
private var lastSpeedIncreaseTime = 0L

// Handler and Runnable for speed increase timer
private val speedIncreaseHandler = Handler(Looper.getMainLooper())
private val speedIncreaseRunnable = object : Runnable {
    override fun run() {
        increaseSpeedIfNeeded()
        speedIncreaseHandler.postDelayed(this, speedIncreaseInterval.toLong())
    }
}

init {
    myPaint = Paint()
    val sharedPreferences = c.getSharedPreferences("GAME_DATA", Context.MODE_PRIVATE)
    record = sharedPreferences.getInt("RECORD", 0)
    lastCoinCreationTime = System.currentTimeMillis()
    lastSpeedIncreaseTime = System.currentTimeMillis()
    startSpeedIncreaseTimer() // Start speed increase timer

    // Ініціалізуємо MediaPlayer з аудіофайлом "point_plus"
    mediaPlayerPointPlus = MediaPlayer.create(c, R.raw.point_plus)
}

override fun onDraw(canvas: Canvas) {
    super.onDraw(canvas)
    viewWidth = this.measuredWidth
    viewHeight = this.measuredHeight

    // Generate cars
    if (time % 700 < 10 + speed) {
        val map = HashMap<String, Any>()
        map["lane"] = (0..2).random()
        map["startTime"] = time
        otherCars.add(map)
    }

    time += 10 + speed
    val carWidth = viewWidth / 5
    val carHeight = carWidth + 10
    myPaint!!.style = Paint.Style.FILL

    val selectedCarId = getSelectedCarId()

    // Draw player's car
    val carDrawableId = when (selectedCarId) {
        "CarWhite" -> R.drawable.car_white
        "CarBlue" -> R.drawable.car_blue
        "CarGreen" -> R.drawable.car_green
        else -> R.drawable.car
    }
    val d = resources.getDrawable(carDrawableId, null)
    d.setBounds(
        myCarPosition * viewWidth / 3 + viewWidth / 15 + 25,
```

```
    viewHeight - 2 - carHeight,
    myCarPosition * viewWidth / 3 + viewWidth / 15 + carWidth - 25,
    viewHeight - 2
)
d.draw(canvas)

// Draw other cars and handle collisions
myPaint!!.color = Color.GREEN
var highScore = 0
for (i in otherCars.indices) {
    try {
        val carX = otherCars[i]["lane"] as Int * viewWidth / 3 + viewWidth / 15
        var carY = time - otherCars[i]["startTime"] as Int

        // Draw red cars
        val carDrawableId = R.drawable.car_red
        val carDrawable = resources.getDrawable(carDrawableId, null)
        carDrawable.setBounds(
            carX + 25, carY - carHeight, carX + carWidth - 25, carY
        )
        carDrawable.draw(canvas)

        // Handle collisions with player's car
        if (otherCars[i]["lane"] as Int == myCarPosition) {
            if (carY > viewHeight - 2 - carHeight && carY < viewHeight - 2) {
                gameTask.closeGame(score, record)
            }
        }

        // Remove cars that have passed the view
        if (carY > viewHeight + carHeight) {
            otherCars.removeAt(i)
            score += 1 // Increase score when a car passes the view
            speed = 1 + abs(score / 8)
            if (score > highScore) {
                highScore = score
                if (highScore > record) {
                    record = highScore
                    val sharedPreferences =
                        c.getSharedPreferences("GAME_DATA", Context.MODE_PRIVATE)
                    val editor = sharedPreferences.edit()
                    editor.putInt("RECORD", record)
                    editor.apply()
                }
            }
        }
    } catch (e: Exception) {
        e.printStackTrace()
    }
}

// Draw coins and handle collisions
if (coinPosition != -1) {
```

```
            val coinWidth = viewWidth / 5
            val coinHeight = coinWidth + 10
            val coinDrawableId = R.drawable.coins
            val coinDrawable = resources.getDrawable(coinDrawableId, null)
            val coinY = coinInitialY - calculateCoinJump(time - coinStartTime)
            coinDrawable.setBounds(
                coinPosition * viewWidth / 3 + viewWidth / 15 + 25,
                coinY - coinHeight,
                coinPosition * viewWidth / 3 + viewWidth / 15 + coinWidth - 25,
                coinY
            )
            coinDrawable.draw(canvas)

            // Handle collisions with player's car
            if (coinPosition == myCarPosition && coinY > viewHeight - 2 - coinHeight && coinY < viewHeight - 2) {
                coins++
                coinPosition = -1 // Remove the coin after collection

                // Відтворення звуку "point_plus"
                mediaPlayerPointPlus.start()
            }
        } else {
            createCoinIfNeeded() // Create a new coin if needed
        }

        // Increase speed periodically
        increaseSpeedIfNeeded()

        // Draw UI elements
        myPaint!!.color = Color.WHITE
        myPaint!!.textSize = 40f
        canvas.drawText("Проїхано: $score", 80f, 80f, myPaint!!)
        canvas.drawText("Рекорд: $record", 380f, 80f, myPaint!!)
        canvas.drawText("Швидкість : $speed", 680f, 80f, myPaint!!)
        canvas.drawText("Монеток : $coins", 680f, 160f, myPaint!!)

        invalidate()
    }

    override fun onTouchEvent(event: MotionEvent?): Boolean {
        when (event!!.action) {
            MotionEvent.ACTION_DOWN -> {
                val x1 = event.x
                if (x1 < viewWidth / 2) {
                    if (myCarPosition > 0) {
                        myCarPosition--
                        score += 1 // Оновлюємо score при кожному переміщенні машини гравця
                    }
                }
                if (x1 > viewWidth / 2) {
                    if (myCarPosition < 2) {
                        myCarPosition++
                        score += 1 // Оновлюємо score при кожному переміщенні машини гравця
```

```
                }
            }
            invalidate()
        }
        MotionEvent.ACTION_UP -> {
        }
    }
    return true
}

private fun createCoinIfNeeded() {
    val currentTime = System.currentTimeMillis()
    val elapsedTime = currentTime - lastCoinCreationTime
    if (elapsedTime >= coinCreationInterval) { // Check if 20 seconds have passed
        coinPosition = (0..2).random()
        coinStartTime = time.toLong() // Set the start time for the new coin
        coinInitialY = viewHeight / 2 // Initial coin position
        lastCoinCreationTime = currentTime // Update the last coin creation time
    }
}

private fun increaseSpeedIfNeeded() {
    val currentTime = System.currentTimeMillis()
    val elapsedTime = currentTime - lastSpeedIncreaseTime
    if (elapsedTime >= speedIncreaseInterval) { // Check if the interval for increasing speed has passed
        speed++
        lastSpeedIncreaseTime = currentTime // Update the last speed increase time
    }
}

private fun calculateCoinJump(timeDiff: Long): Int {
    val fallingSpeed = 1 // Скорость падения монеты
    val startY = viewHeight - 1000 // Начальная позиция монеты (нижний край экрана)
    val endY = -20000 // Конечная позиция монеты (верхний край экрана)

    // Рассчитываем новую позицию монеты
    val newY = startY - (fallingSpeed * timeDiff).toInt()

    // Ограничиваем позицию монеты, чтобы она не выходила за пределы экрана
    return if (newY < endY) {
        endY
    } else {
        newY
    }
}

private fun getSelectedCarId(): String {
    val sharedPreferences = c.getSharedPreferences("GAME_DATA", Context.MODE_PRIVATE)
    return sharedPreferences.getString("SELECTED_BUTTON_ID", "") ?: ""
}

// Start speed increase timer
private fun startSpeedIncreaseTimer() {
```

```
        speedIncreaseHandler.postDelayed(speedIncreaseRunnable, speedIncreaseInterval.toLong())
    }
} package com.example.pdr_ukraine_2024;

import android.content.Intent;
import android.media.MediaPlayer;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;

public class PdrActivityMenu extends AppCompatActivity {

    private String selectedTopicPDR = "";
    MediaPlayer mediaPlayerNext, mediaPlayerFail, mediaPlayerOption, mediaPlayerTrue, mediaPlayerTrue_2;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_pdr_menu);

        final ImageView backBtn = findViewById(R.id.backBtn);
        final LinearLayout Road_Signs = findViewById(R.id.Road_Signs);
        final LinearLayout Regulator = findViewById(R.id.Regulator);
        final LinearLayout Rail_crossing = findViewById(R.id.Rail_crossing);
        final LinearLayout Parking = findViewById(R.id.Parking);

        final Button qizStartPDR = findViewById(R.id.QizstartPDR);

        mediaPlayerNext = MediaPlayer.create(this,R.raw.sounds_1);
        mediaPlayerOption = MediaPlayer.create(this,R.raw.sounds_2);;
        mediaPlayerFail = MediaPlayer.create(this,R.raw.sounds_3);
        mediaPlayerTrue = MediaPlayer.create(this,R.raw.sound_true);
        mediaPlayerTrue_2 = MediaPlayer.create(this,R.raw.sounds_4);

        backBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                mediaPlayerNext.start();
                Intent intent = new Intent(PdrActivityMenu.this, MainActivity.class);
                startActivity(intent);
                finish();
            }
        });

        Road_Signs.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                mediaPlayerOption.start();
```

```
        selectedTopicPDR = "Road_Signs";
        Road_Signs.setBackgroundResource(R.drawable.round_back_stroke10);

        Parking.setBackgroundResource(R.drawable.round_backa_white10);
        Rail_crossing.setBackgroundResource(R.drawable.round_backa_white10);
        Regulator.setBackgroundResource(R.drawable.round_backa_white10);
        // Додаткові дії, якщо потрібно
    }
});

Regulator.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        mediaPlayerOption.start();
        selectedTopicPDR = "Regulator";
        Regulator.setBackgroundResource(R.drawable.round_back_stroke10);

        Road_Signs.setBackgroundResource(R.drawable.round_backa_white10);
        Parking.setBackgroundResource(R.drawable.round_backa_white10);
        Rail_crossing.setBackgroundResource(R.drawable.round_backa_white10);
        // Додаткові дії, якщо потрібно
    }
});

Rail_crossing.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        mediaPlayerOption.start();
        selectedTopicPDR = "Rail_crossing";
        Rail_crossing.setBackgroundResource(R.drawable.round_back_stroke10);

        Regulator.setBackgroundResource(R.drawable.round_backa_white10);
        Road_Signs.setBackgroundResource(R.drawable.round_backa_white10);
        Parking.setBackgroundResource(R.drawable.round_backa_white10);
        // Додаткові дії, якщо потрібно
    }
});

Parking.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        mediaPlayerOption.start();
        selectedTopicPDR = "Parking";
        Parking.setBackgroundResource(R.drawable.round_back_stroke10);

        Rail_crossing.setBackgroundResource(R.drawable.round_backa_white10);
        Regulator.setBackgroundResource(R.drawable.round_backa_white10);
        Road_Signs.setBackgroundResource(R.drawable.round_backa_white10);
        // Додаткові дії, якщо потрібно
    }
});

qizStartPDR.setOnClickListener(new View.OnClickListener() {
```

```
        @Override
        public void onClick(View view) {
            if (selectedTopicPDR.isEmpty()) {
                mediaPlayerNext.start();
                Toast.makeText(PdrActivityMenu.this, "Оберіть вікторину! ", Toast.LENGTH_SHORT).show();
            } else {
                Intent intent;
                mediaPlayerTrue_2.start();
                // Відкриття вікна activity_quiz.xml для вікторини
                intent = new Intent(PdrActivityMenu.this, QuizActivity.class);
                intent.putExtra("selectedTopic", selectedTopicPDR);
                startActivity(intent);
                finish();
            }
        }
    });
  }

  @Override
  public void onBackPressed() {
    // Перехід до MainActivity при натисканні кнопки назад
    Intent intent = new Intent(PdrActivityMenu.this, MainActivity.class);
    startActivity(intent);
    finish();
  }
} package com.example.pdr_ukraine_2024;

import android.content.Intent;
import android.media.MediaPlayer;
import android.os.Bundle;
import android.view.View;
import android.widget.ImageView;
import android.widget.TextView;

import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.AppCompatButton;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Random;

public class QuizResults extends AppCompatActivity {

  MediaPlayer mediaPlayerApplause, mediaPlayerNext, mediaPlayerGameOver;
  TextView mainText;
  Random randomText;
  ArrayList<String> arrayLists;

  @Override
```

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_quiz_results);

    mediaPlayerApplause = MediaPlayer.create(this, R.raw.sounds_applause);
    mediaPlayerNext = MediaPlayer.create(this, R.raw.sounds_1);
    mediaPlayerGameOver = MediaPlayer.create(this, R.raw.game_over);

    final TextView totalResultTextView = findViewById(R.id.overallResult);
    final TextView correctAnswersTextView = findViewById(R.id.correctAnswears);
    final TextView incorrectAnswersTextView = findViewById(R.id.incorrectAnswears);
    final AppCompatButton startNewQuiz = findViewById(R.id.startNewQuiz);

    final int getCorrectAnswers = getIntent().getIntExtra("correct", 0);
    final int getIncorrectAnswers = getIntent().getIntExtra("incorrect", 0);

    final int totalResult = getCorrectAnswers - getIncorrectAnswers;

    // Оновлення тексту відповідних TextView
    totalResultTextView.setText("Кількість балів: " + totalResult);
    correctAnswersTextView.setText("Правильних відповідей: " + getCorrectAnswers);
    incorrectAnswersTextView.setText("Не правильних відповідей: " + getIncorrectAnswers);

    // Перевірка кількості помилок та відтворення відповідного звуку
    if (getIncorrectAnswers > 3) {
        // Змінення зображення на crocodile_2.png
        ImageView imageView = findViewById(R.id.congratulayionIcon);
        imageView.setImageResource(R.drawable.crocodile_2);

        // Відтворення звуку mediaPlayerGameOver
        mediaPlayerGameOver.start();
    } else {
        ImageView imageView = findViewById(R.id.congratulayionIcon);
        imageView.setImageResource(R.drawable.crocodilee);

        // Відтворення звуку mediaPlayerApplause
        mediaPlayerApplause.start();
    }

    startNewQuiz.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            mediaPlayerNext.start();
            startActivity(new Intent(QuizResults.this, MainActivity.class));
            finish();
        }
    });

    // Запис totalResult у файл
    writeTotalResultToFile(totalResult);

    // Відображення значення з файлу у TextView
    displayTotalResultFromFile();
```

```
mainText = findViewById(R.id.main_text);
randomText = new Random();
arrayLists = new ArrayList<>();
arrayLists.add(getString(R.string.Text_0000));
arrayLists.add(getString(R.string.Text_0001));
arrayLists.add(getString(R.string.Text_0002));
arrayLists.add(getString(R.string.Text_0003));
arrayLists.add(getString(R.string.Text_0004));
arrayLists.add(getString(R.string.Text_0005));
arrayLists.add(getString(R.string.Text_0006));
arrayLists.add(getString(R.string.Text_0007));
arrayLists.add(getString(R.string.Text_0008));
arrayLists.add(getString(R.string.Text_0009));
arrayLists.add(getString(R.string.Text_0010));
arrayLists.add(getString(R.string.Text_0011));
arrayLists.add(getString(R.string.Text_0012));
arrayLists.add(getString(R.string.Text_0013));
arrayLists.add(getString(R.string.Text_0014));
arrayLists.add(getString(R.string.Text_0015));
arrayLists.add(getString(R.string.Text_0016));
arrayLists.add(getString(R.string.Text_0017));
arrayLists.add(getString(R.string.Text_0018));
arrayLists.add(getString(R.string.Text_0019));
arrayLists.add(getString(R.string.Text_0020));
arrayLists.add(getString(R.string.Text_0021));
arrayLists.add(getString(R.string.Text_0022));
arrayLists.add(getString(R.string.Text_0023));
arrayLists.add(getString(R.string.Text_0024));
arrayLists.add(getString(R.string.Text_0025));
arrayLists.add(getString(R.string.Text_0026));
arrayLists.add(getString(R.string.Text_0027));
arrayLists.add(getString(R.string.Text_0028));
arrayLists.add(getString(R.string.Text_0029));
arrayLists.add(getString(R.string.Text_0030));
arrayLists.add(getString(R.string.Text_0031));
arrayLists.add(getString(R.string.Text_0032));
arrayLists.add(getString(R.string.Text_0033));
arrayLists.add(getString(R.string.Text_0034));
arrayLists.add(getString(R.string.Text_0035));
arrayLists.add(getString(R.string.Text_0036));
arrayLists.add(getString(R.string.Text_0037));
arrayLists.add(getString(R.string.Text_0038));
arrayLists.add(getString(R.string.Text_0039));
arrayLists.add(getString(R.string.Text_0040));
arrayLists.add(getString(R.string.Text_0041));
arrayLists.add(getString(R.string.Text_0042));
arrayLists.add(getString(R.string.Text_0043));
arrayLists.add(getString(R.string.Text_0044));
arrayLists.add(getString(R.string.Text_0045));
arrayLists.add(getString(R.string.Text_0046));
arrayLists.add(getString(R.string.Text_0047));
arrayLists.add(getString(R.string.Text_0048));
```

```
        arrayLists.add(getString(R.string.Text_0049));

        mainText.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                int randomIndex = randomText.nextInt(arrayLists.size());
                mainText.setText(arrayLists.get(randomIndex));
            }
        });
    }

    // Метод для запису totalResult у файл
    private void writeTotalResultToFile(int totalResult) {
        try {
            // Відкрити файл для запису
            FileWriter writer = new FileWriter(new File(getFilesDir(), "saving_result.txt"), true);

            // Записати totalResult у файл та додати символ нового рядка
            writer.write(totalResult + "\n");

            // Закрити потік запису
            writer.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    // Метод для відображення значення з файлу у TextView
    private void displayTotalResultFromFile() {
        try {
            File file = new File(getFilesDir(), "saving_result.txt");
            if (file.exists()) {
                BufferedReader br = new BufferedReader(new FileReader(file));
                String line;
                int totalResultFromFile = 0;
                while ((line = br.readLine()) != null) {
                    int result = Integer.parseInt(line);
                    if (result >= 101) {
                        result = 100;
                    } else if (result <= -1) {
                        result = 0;
                    }
                    totalResultFromFile += result;
                }
                br.close();
                TextView totalResultTextView = findViewById(R.id.overallResult);
                totalResultTextView.setText("Кількість балів: " + totalResultFromFile);
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
} package com.example.pdr_ukraine_2024;
```

```
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.view.View;
import android.widget.ImageView;
import android.widget.LinearLayout;

import androidx.appcompat.app.AppCompatActivity;

public class TheoryMenyActivity_PDR extends AppCompatActivity {

    private ImageView backBtn;
    private LinearLayout Road_signs_theory;
    private LinearLayout Regulator_theory;
    private LinearLayout RR_movement_theory;
    private LinearLayout Parking_movement_theory;
    private LinearLayout Practice_theory;
    private LinearLayout Practice_theory2;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_theory_meny_pdr); // Set layout

        backBtn = findViewById(R.id.backBtn);
        Road_signs_theory = findViewById(R.id.Road_signs_theory);
        Regulator_theory = findViewById(R.id.Regulator_theory);
        RR_movement_theory = findViewById(R.id.RR_movement_theory);
        Parking_movement_theory = findViewById(R.id.Parking_movement_theory);
        Practice_theory = findViewById(R.id.Practice_theory);
        Practice_theory2 = findViewById(R.id.Practice_theory2);

        backBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                finish(); // Закриваємо активність (Close activity)
            }
        });

        // Додаємо обробник натискання на лейаути (Add click listeners)
        Road_signs_theory.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                openYouTubeVideo("https://www.youtube.com/road_signs_theory");
            }
        });

        Regulator_theory.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                openYouTubeVideo("https://www.youtube.com/regulator_theory");
            }
```

```
        });

        RR_movement_theory.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                openYouTubeVideo("https://www.youtube.com/rr_movement_theory");
            }
        });

        Parking_movement_theory.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                openYouTubeVideo("https://www.youtube.com/parking_movement_theory");
            }
        });

        Practice_theory.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                openYouTubeVideo("https://www.youtube.com/practice_theory");
            }
        });

        Practice_theory2.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                openYouTubeVideo("https://www.youtube.com/practice_theory2");
            }
        });
    }

    // Метод для відкриття відео на YouTube (Method to open YouTube video)
    private void openYouTubeVideo(String url) {
        Intent intent = new Intent(Intent.ACTION_VIEW, Uri.parse(url));
        intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
        intent.setPackage("com.google.android.youtube");
        startActivity(intent);
    }
} package com.example.pdr_ukraine_2024;

import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.view.View;
import android.widget.ImageView;
import android.widget.LinearLayout;

import androidx.appcompat.app.AppCompatActivity;

public class TheoryMenyActivity_DMD extends AppCompatActivity {

    private ImageView backBtn;
    private LinearLayout Stop_bleeding;
```

```
private LinearLayout Bone_fracture;
private LinearLayout Poisoning;
private LinearLayout Reanimation;
private LinearLayout Infarct_theory;
private LinearLayout Depression_theory;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_theory_meny_dmd);

    backBtn = findViewById(R.id.backBtn);
    Stop_bleeding = findViewById(R.id.Stop_bleeding_theory);
    Bone_fracture = findViewById(R.id.Bone_fracture_theory);
    Poisoning = findViewById(R.id.Poisoning_theory);
    Reanimation = findViewById(R.id.Reanimation_theory);
    Infarct_theory = findViewById(R.id.Infarct_theory);
    Depression_theory = findViewById(R.id.Depression_theory);

    backBtn.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            finish(); // Закриваємо активність
        }
    });

    Stop_bleeding.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            openYouTubeVideo("https://www.youtube.com/watch?v=7yz8CKJ6_34");
        }
    });

    Bone_fracture.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            openYouTubeVideo("https://www.youtube.com/watch?v=DmzKrFp8ImY");
        }
    });

    Poisoning.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            openYouTubeVideo("https://www.youtube.com/watch?v=-8EZoX2HqiA");
        }
    });

    Reanimation.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            openYouTubeVideo("https://www.youtube.com/watch?v=E1N9YYH5JbU");
        }
    });
```

Додаток Г - Графічна частина

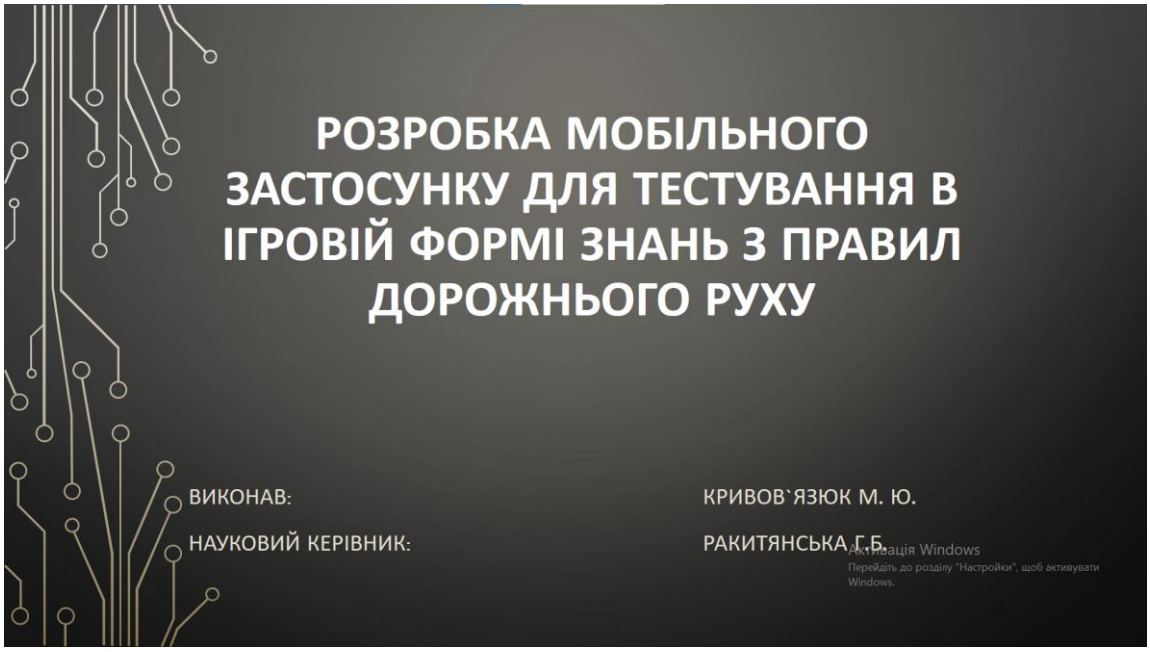

Рисунок Г.1 – Назва роботи

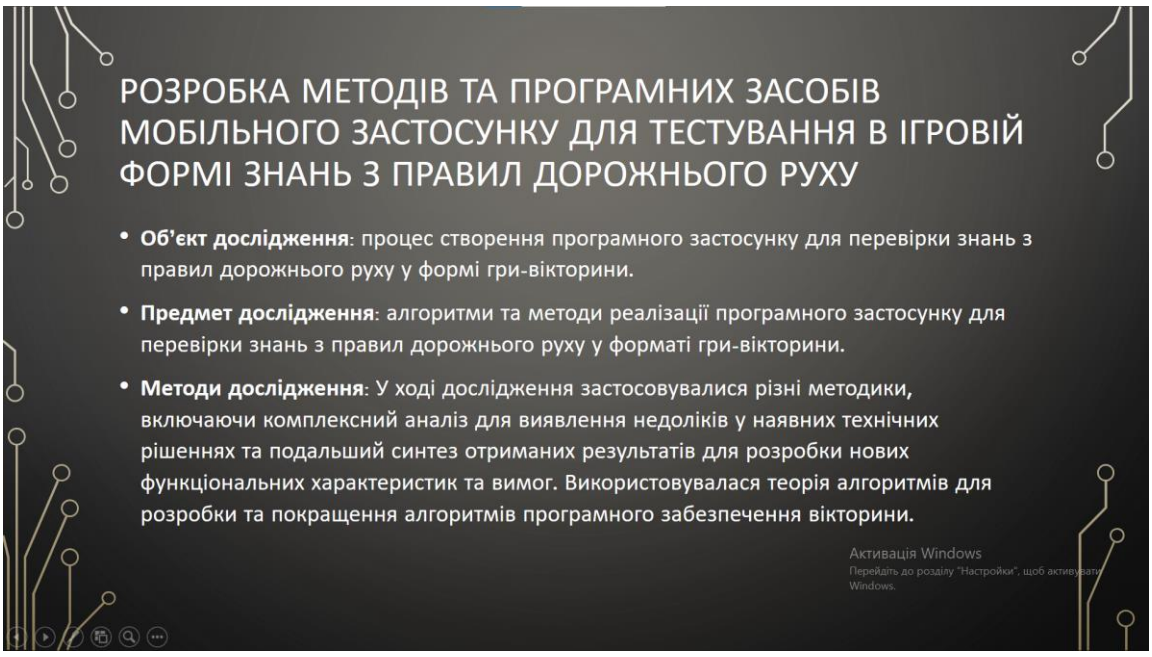

Рисунок Г.2 – Мета роботи

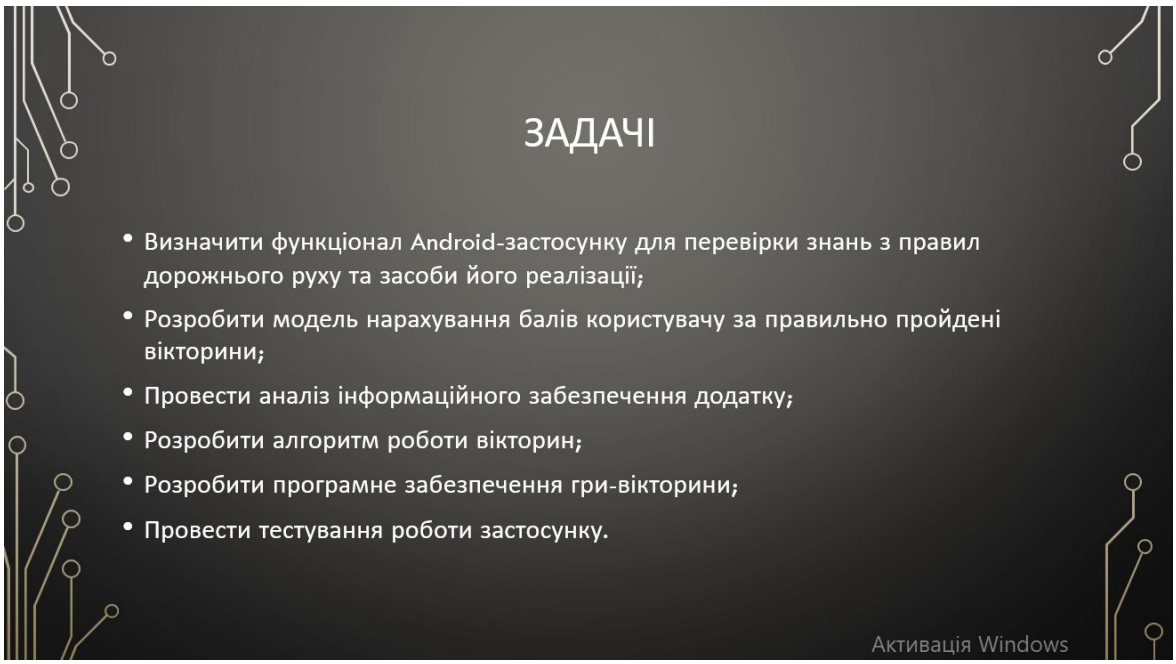

Рисунок Г.3 – Задачі

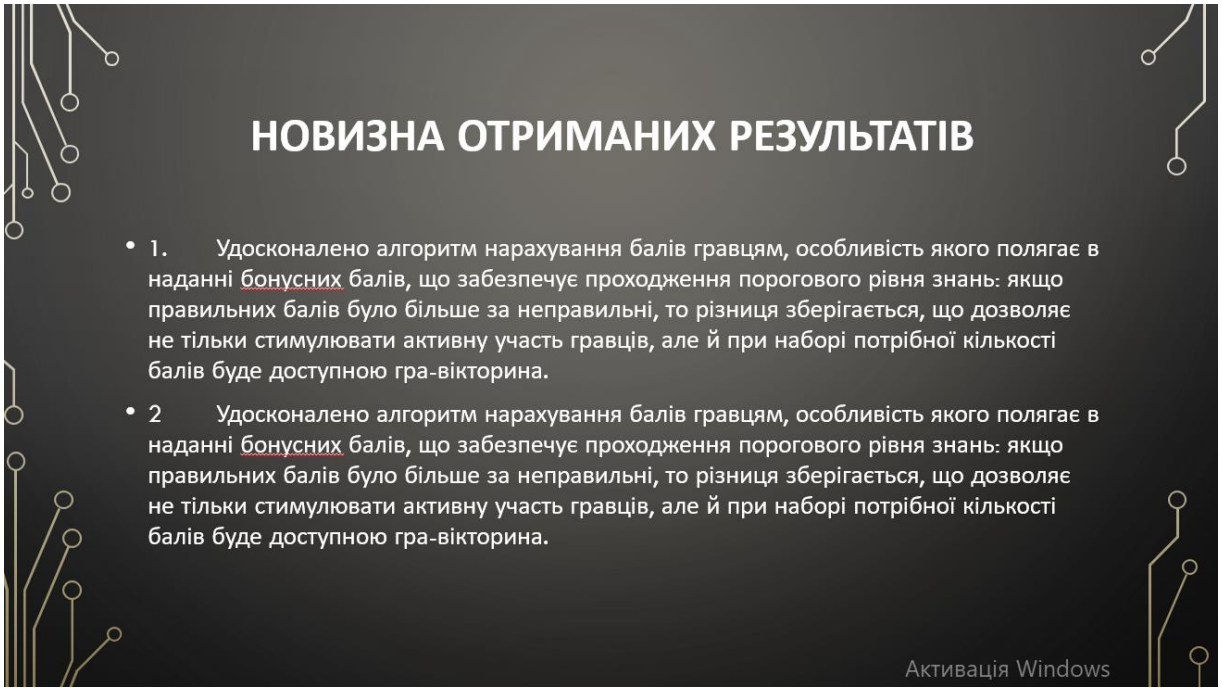

Рисунок Г.4 – Новизна отриманих результатів

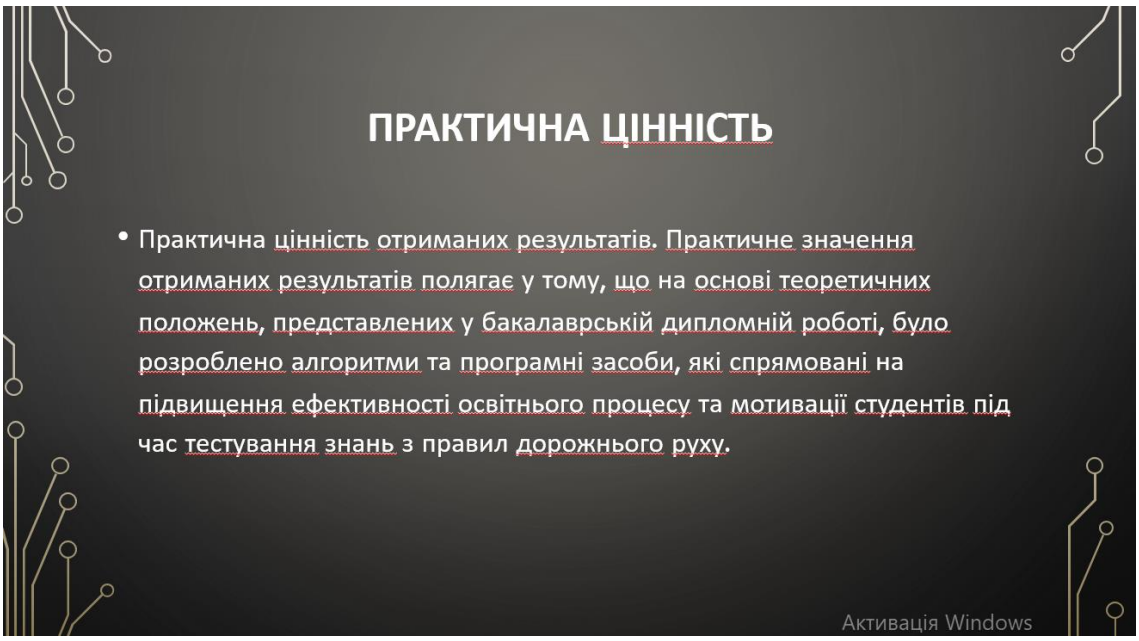

Рисунок Г.5 – Практична цінність

Рисунок Г.6 – Порівняння аналогів

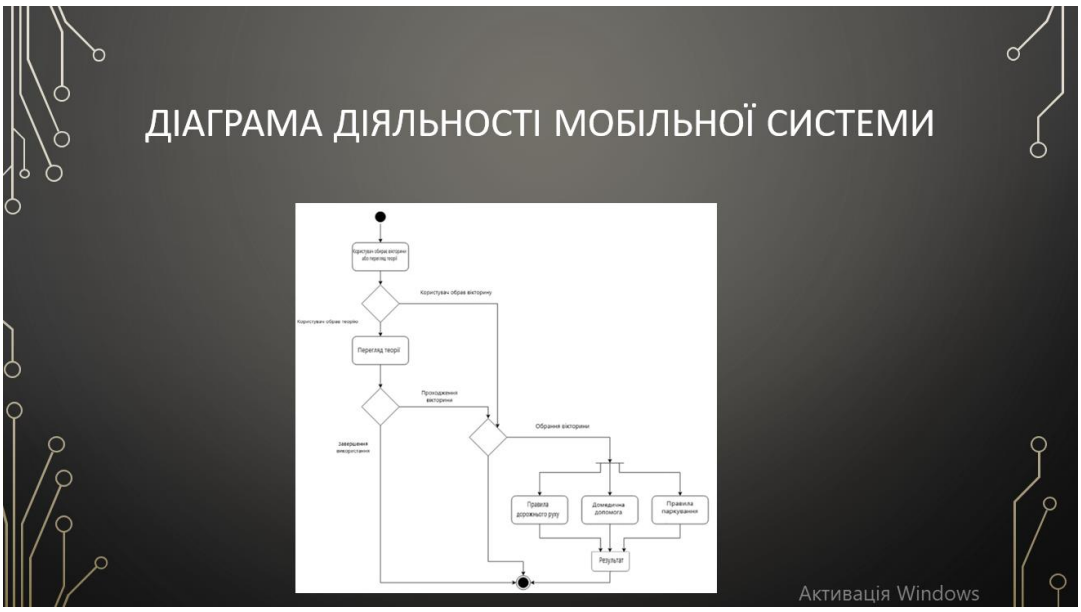

Рисунок Д.Г – Діаграма діяльності

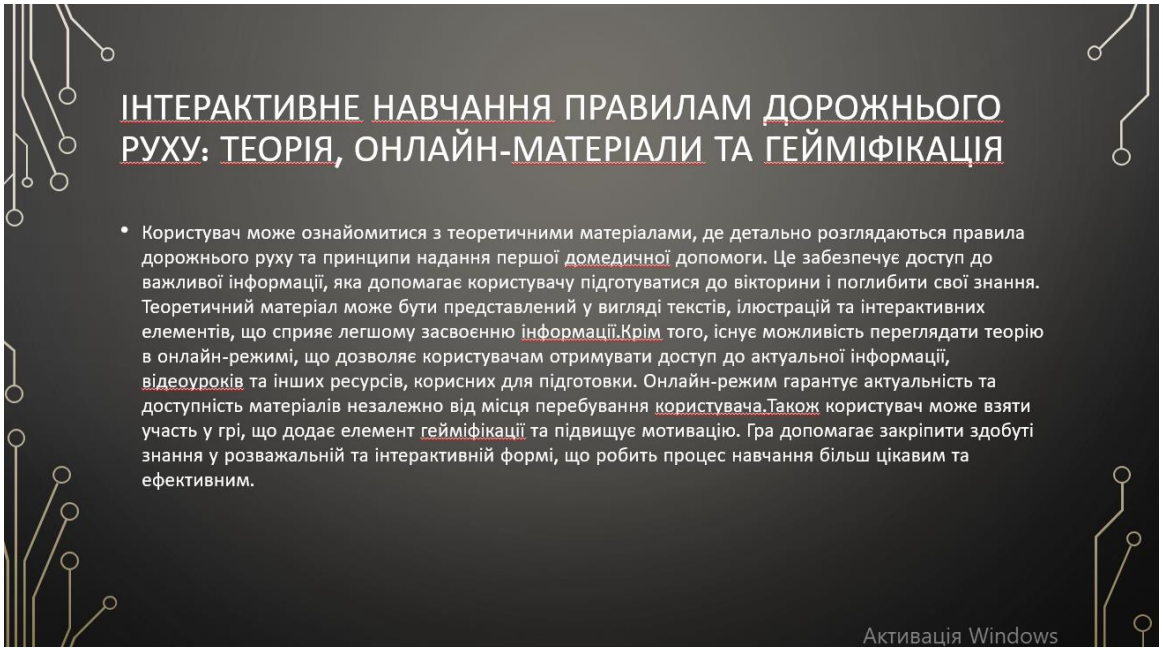

Рисунок Г.8 – Інтерактивне навчання правилам дорожнього руху

Рисунок Г.9 – Блок-схема алгоритму начислення балів користувачу

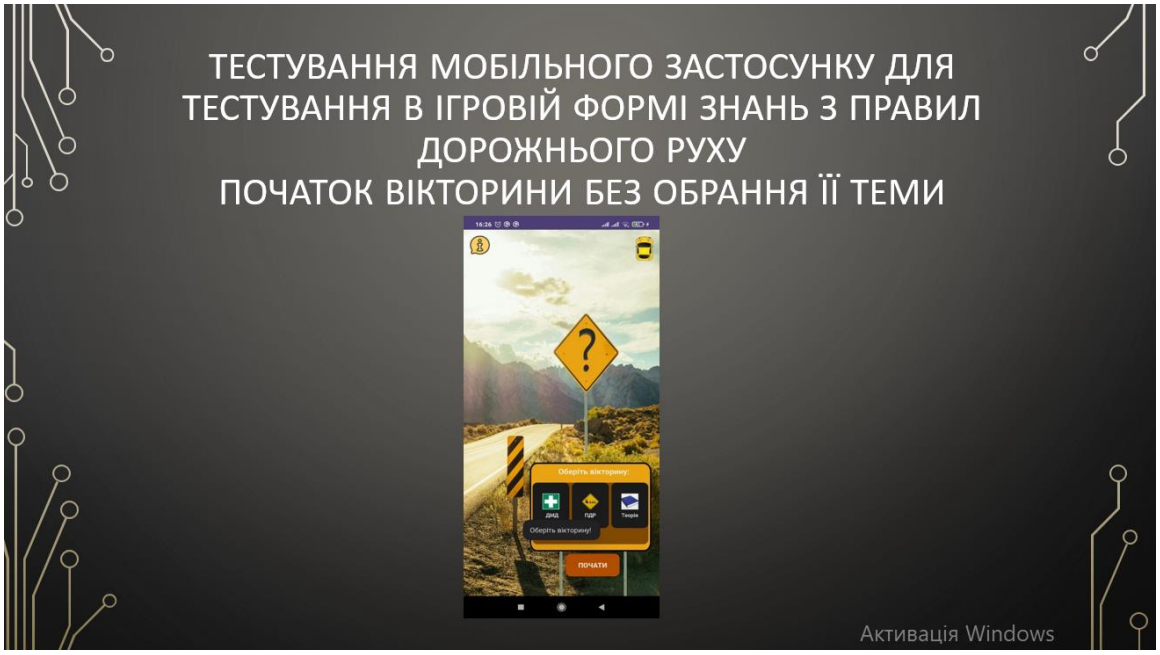

Рисунок Г.10 – Тестування застосунку

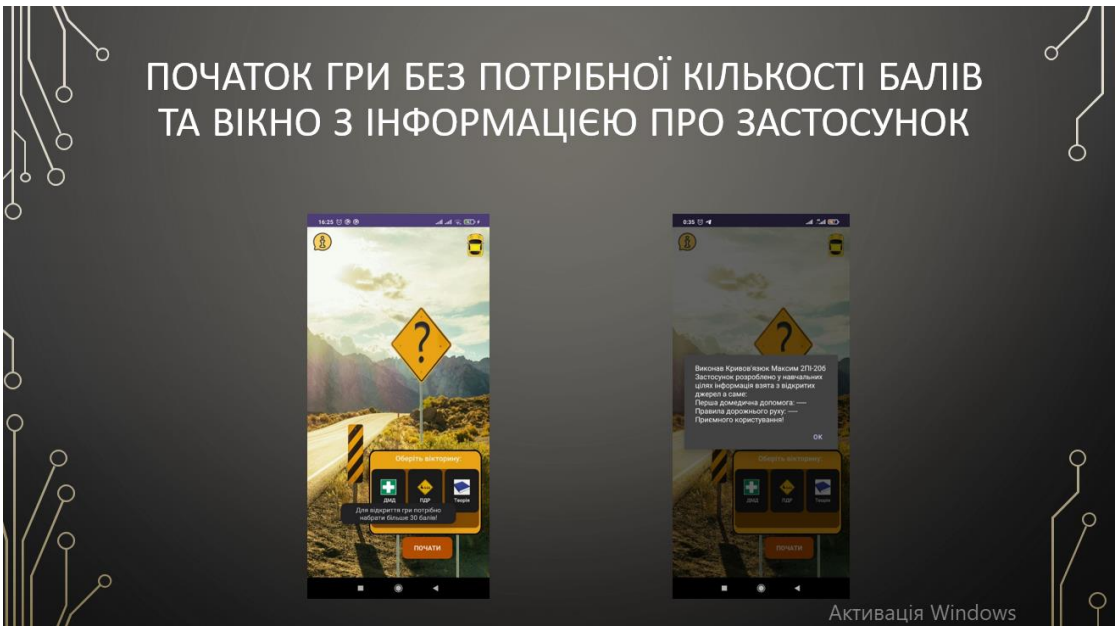

Рисунок Г.11 – Початок гри без потрібної кількості балів та вікно з інформацією про застосунок

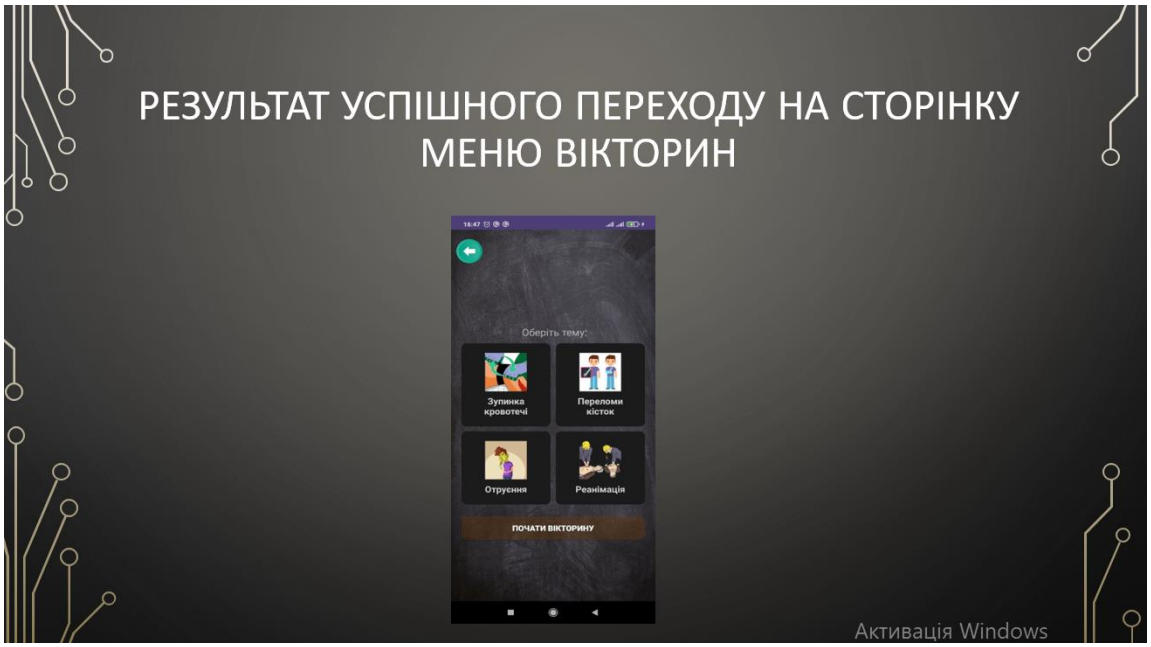

Рисунок Г.13 – Меню вікторин

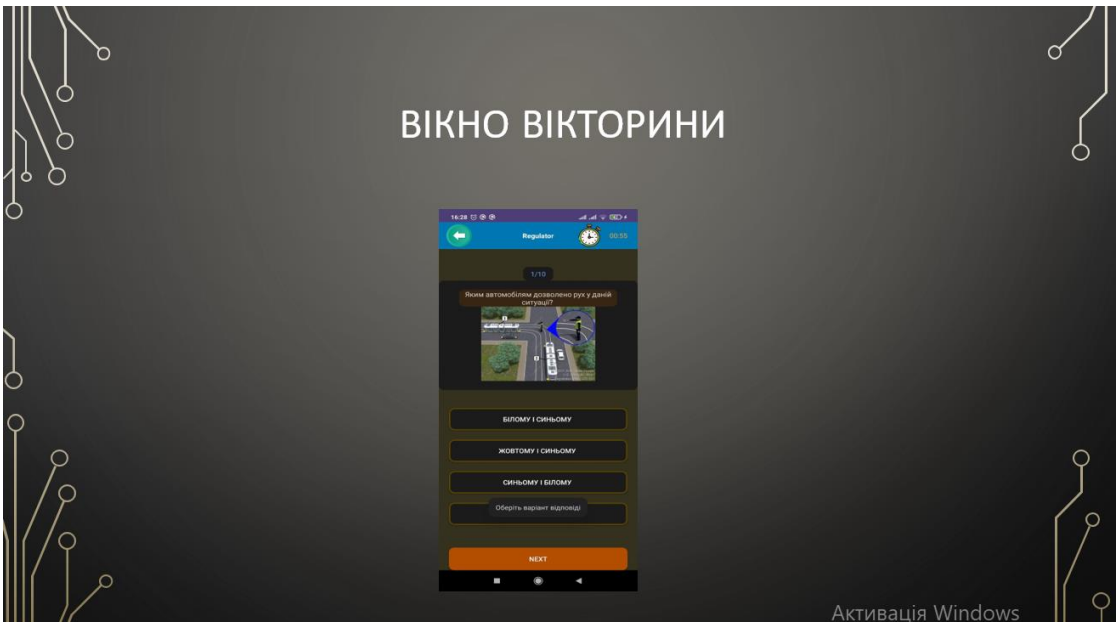

Рисунок Г.14 – Вікно вікторини

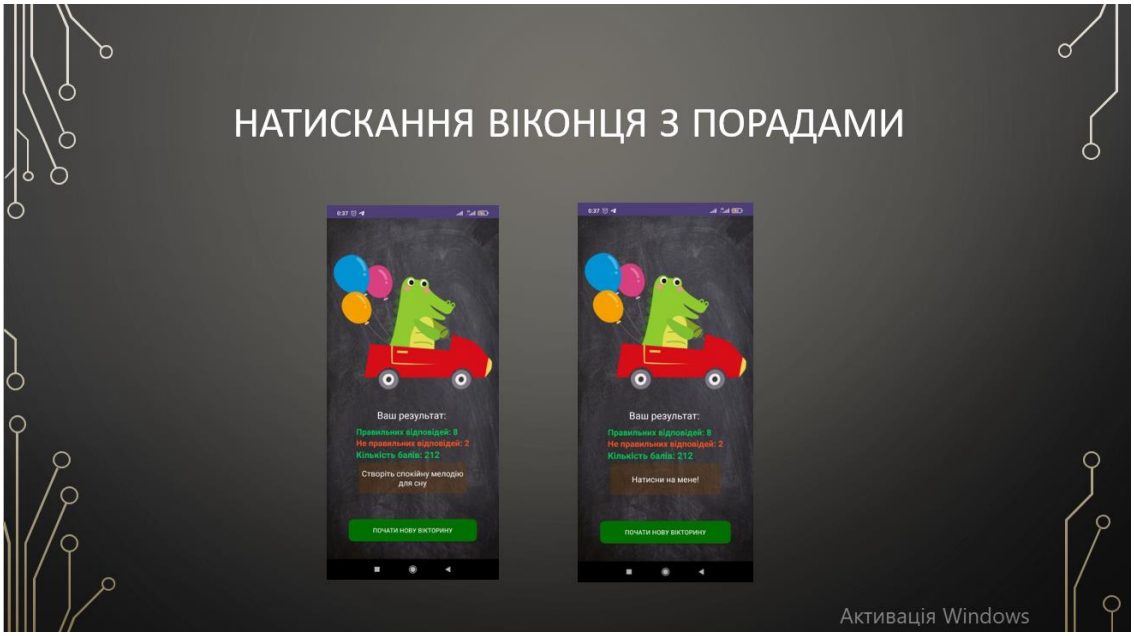

Рисунок Г.14 – Вікно з результатом 1

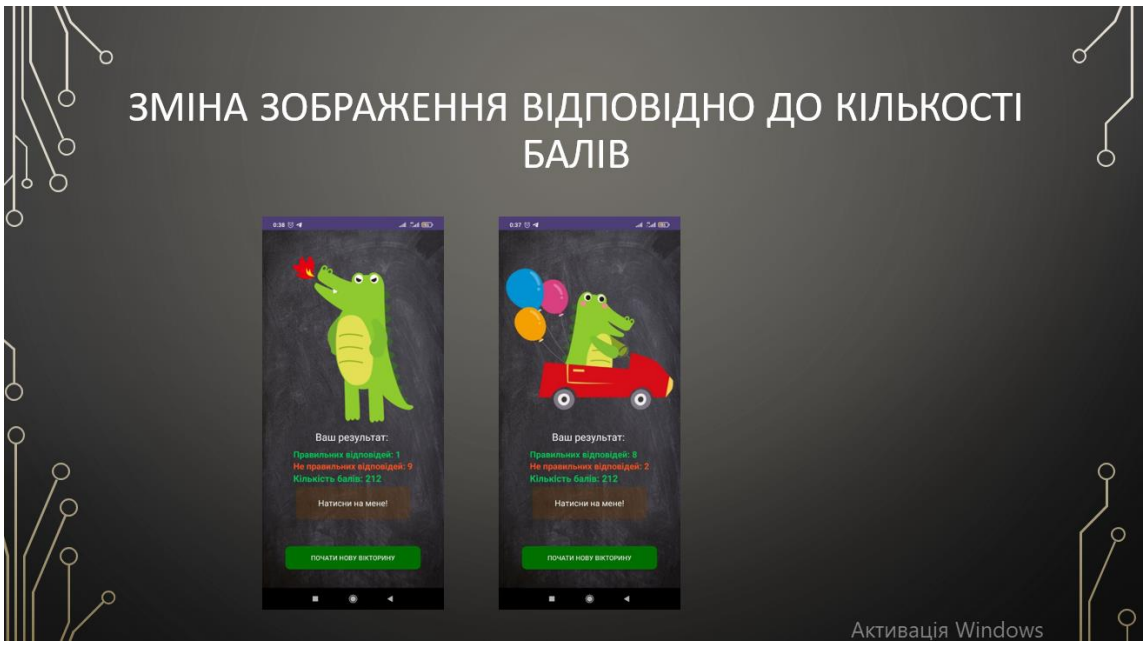

Рисунок Г.15 – Вікно з результатом 2

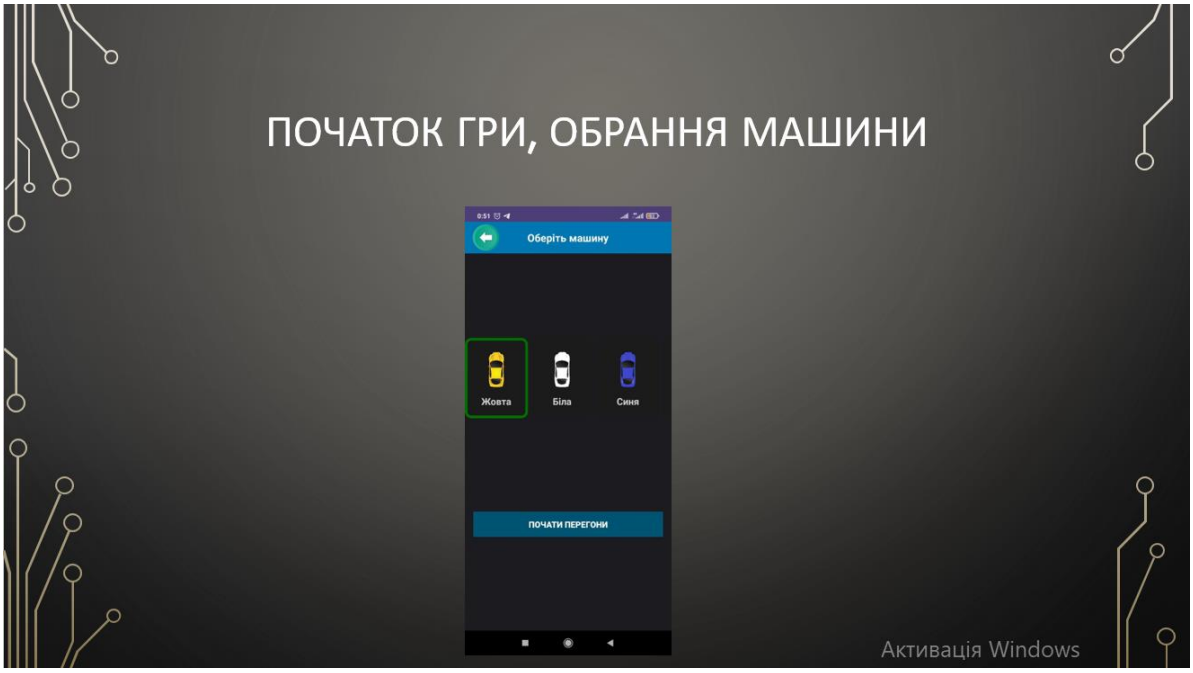

Рисунок Г.16 – Меню гри

Рисунок Г.17 – Гра Перегони

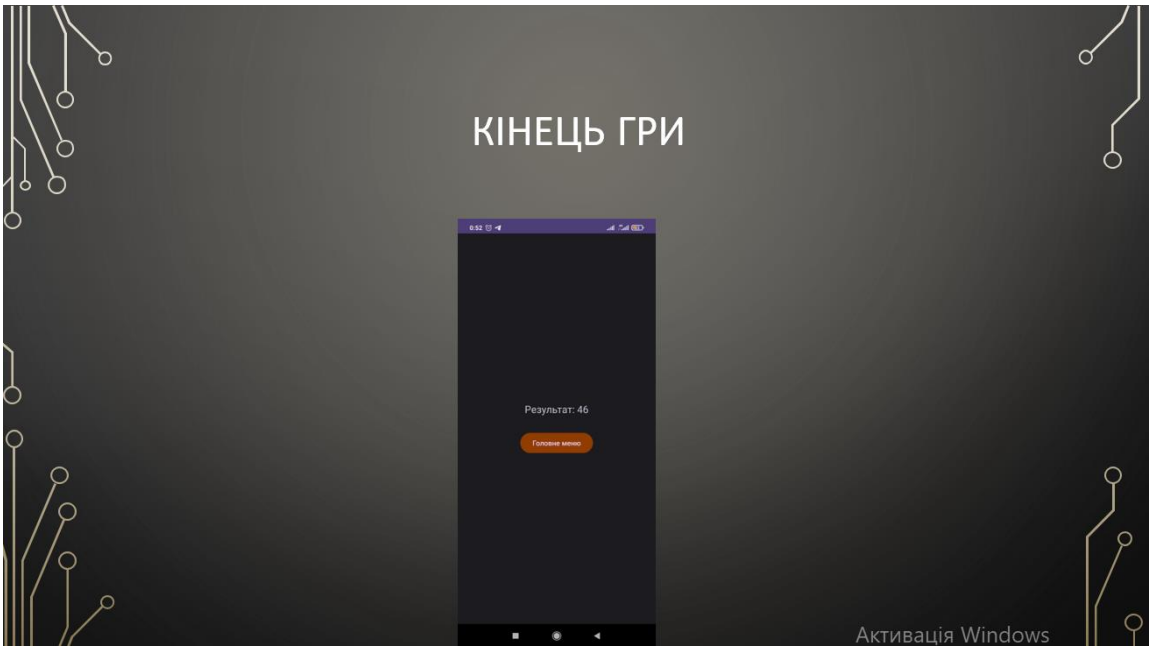

Рисунок Г.18 – Кінець гри

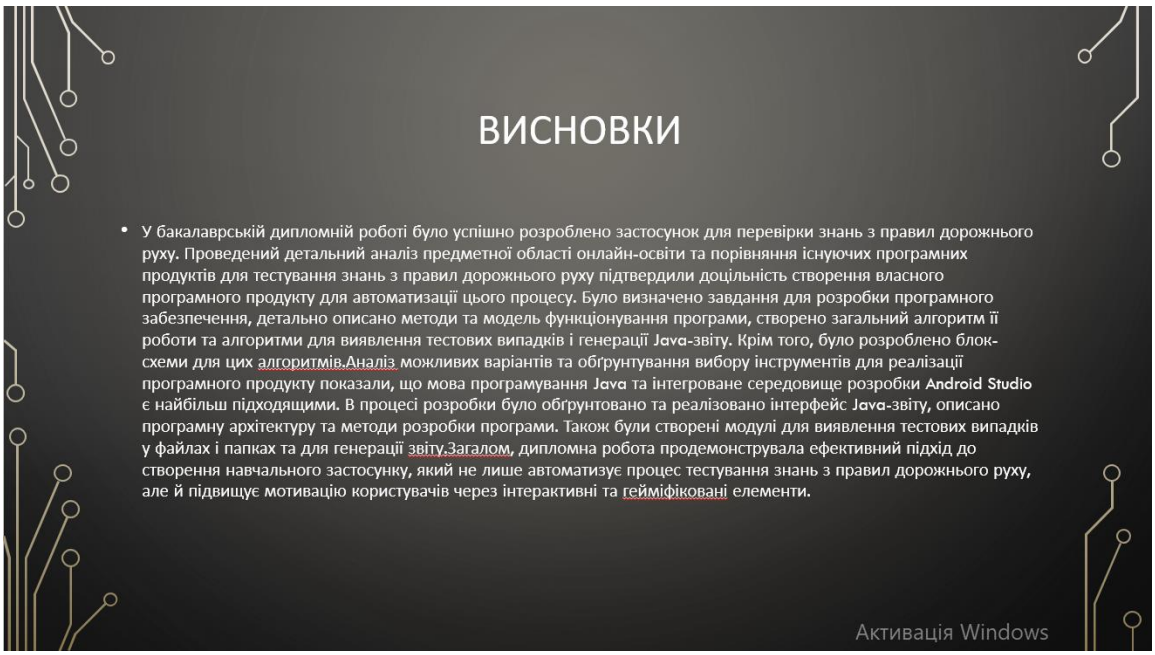

Рисунок Г.19 – Висновок

Рисунок Г.20 – Дякую за увагу