Вінницький національний технічний університет
(повне найменування вищого навчального закладу)
Факультет інформаційних технологій та комп'ютерної інженерії
(повне найменування інституту, назва факультету (відділення))
Кафедра програмного забезпечення
(повна назва кафедри (предметної, циклової комісії))

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

на тему: «Розробка методів та програмних засобів навчальної гри з історії України для Android -додатку з використанням методів адаптивного навчання»

Виконав: студент 2-го курсу, групи ЗПІ-23м
спеціальності 121 – Інженерія програмного забезпечення
Лаба Діана Сергіївна

Керівник: к.т.н., доц. каф. ПЗ Войтко В.В.
«06» грудня 2023 р.

Опонент: к.т.н., доц. каф. ОТ Савицька Л.А.
«06» грудня 2023 р.

Допущено до захисту
Завідувач кафедри ПЗ
к.т.н., проф. Романюк О. Н.
(прізвище та ініціали)
«06» грудня 2023 р.

ВНТУ – 2023

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

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
Романюк О. Н.
« 19 » вересня 2023 р.

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТЦІ

Лабі Діані Сергіївні

1. Тема роботи – Розробка методів та програмних засобів навчальної гри з історії України для Android -додатку з використанням методів адаптивного навчання.

Керівник роботи: Войтко Вікторія Володимирівна, к.т.н., доц. кафедри ПЗ, затверджені наказом вищого навчального закладу від « 18 » вересня 2023 р. № 247.

2. Строк подання студентом роботи
5 грудня 2023 р.

3. Вихідні дані до роботи: середовище розробки Android Studio, мова розробки Java, система керування базами даних – Firebase, операційна система – Windows 10, метод адаптивного навчання.

4. Зміст текстової частини: вступ; аналіз стану розвитку навчальних ігор та постановка задач дослідження; розробка методу та моделей навчальної системи; розробка методів та програмних засобів навчальної гри з історії України для Android-додатку з використанням методів адаптивного навчання; тестування програми; економічна частина; висновки; перелік використаних джерел; додатки.

5. Перелік ілюстративного матеріалу: блок-схеми алгоритмів роботи мобільного додатку; моделі роботи навчального додатку; тестування мобільного застосунку.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	Войтко В.В., к.т.н., доц. кафедри ПЗ	19.09.23	05.12.23
5	Причепа І.В., к.е.н., доцент	22.11.23	01.12.23

7. Дата видачі завдання 19 вересня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз стану розвитку навчальних ігор та постановка задач дослідження	20.09.2023 30.09.2023	Вик
2	Розробка методу та моделей навчальної системи	01.10.2023 17.10.2023	Вик
3	Розробка програмних модулів навчальної гри з історії України для Android-додатку з використанням методів адаптивного навчання	18.10.2023 04.11.2023	Вик.
4	Тестування програми	05.11.2023 21.11.2023	Вик.
5.	Економічна частина	22.11.2023 01.12.2023	Вик

Студент _____ **Лаба Д. С.**
(підпис) (прізвище та ініціали)

Керівник магістерської кваліфікаційної роботи _____ **Войтко В.В.**
(підпис) (прізвище та ініціали)

Анотація

УДК 004.051

Лаба Д.С. Розробка методів та програмних засобів навчальної гри з історії України для Android-додатку з використанням методів адаптивного навчання. Магістерська кваліфікаційна робота зі спеціальності 121 – інженерія програмного забезпечення, освітня програма – інженерія програмного забезпечення. Вінниця: ВНТУ, 2023. 152 с.

На укр. мові. Бібліогр.: назв: 49; рисунків: 44; таблиць: 13.

У магістерській кваліфікаційній роботі розроблено додаток для вивчення історії України з адаптивним методом навчання. Було проведений детальний порівняльний аналіз існуючих аналогів системи, який дозволив визначити їхні переваги та недоліки. У роботі ретельно обґрунтовано доцільність розробки даного програмного продукту, вказано, які конкретні переваги та унікальні риси має навчальний додаток у порівнянні з існуючими аналогами.

Розроблено метод індивідуалізації навчального процесу, який спрямований на адаптацію навчання для кожного учня. Розроблено модель нарахування балів та отримання бонусів для навчальної гри, яка орієнтована на мотивацію, що надає бонусні бали за досягнення певних цілей або завдань та адаптує рівень складності завдань для кожного учня. Розроблено навчальний додаток з використанням інтегрованого середовища розробки Android Studio та мови програмування Java. Системою управління базами даних обрано Firebase.

Результати магістерської кваліфікаційної роботи можуть бути використані для самостійного навчання або для навчання в навчальних закладах.

Ключові слова: індивідуалізація, адаптація, навчальний додаток, бали, бонуси, навчальний процес.

Abstract

Laba D.S. Development of methods and software tools of educational game on the history of Ukraine for Android application using adaptive learning methods. Master's thesis on the specialty 121 - software engineering, educational program - software engineering. Vinnytsia: VNTU, 2023. 152 p.

In Ukrainian speech Bibliogr.: 49 title; drawings: 44; Tables 13.

In the master's qualification thesis, an application for studying the history of Ukraine with an adaptive learning method was developed. A detailed comparative analysis of existing analogues of the system was carried out, which made it possible to determine their advantages and disadvantages. The work carefully substantiates the feasibility of developing this software product, it is indicated what specific advantages and unique features the educational application has in comparison with existing analogues.

A method of individualizing the educational process has been developed, which is aimed at adapting education for each student. A model of scoring and receiving coins has been developed for the educational game, which is focused on motivation, which gives bonus points for achieving certain goals or tasks and adapts the level of difficulty of the tasks for everyone. An educational application was developed using the Android Studio integrated development environment and the Java programming language. Firebase was chosen as the database management system.

The results of the master's qualification work can be used for independent study or for study in educational institutions.

Keywords: individualization, adaptation, educational application, points, coins, educational process.

ЗМІСТ

ВСТУП

Обґрунтування вибору теми дослідження. Сьогодні з розвитком мобільних технологій смартфони та планшети стали важливою частиною нашого життя, а додатки для навчання та розваг на цих платформах стають все популярнішими. Сучасні технології відкривають нові можливості для покращення освітнього процесу як в школах, так і в університетах. Вони спрощують процес навчання завдяки безмежному доступу до інформації та полегшеній комунікації між викладачами та студентами [1].

Застосування принципів гейміфікації включає в себе використання ігрових концепцій у контексті навчання та не відокремлює увесь навчальний процес як гру. На уроках виключено використання іграшок та електронних пристроїв, за винятком використання їх для відеозв'язку. Гейміфікація не завжди означає проведення міжучнівських змагань із нагородами, але ці елементи можуть бути впроваджені. Основна мета методу полягає в мотивації учнів до певної активності, введенні елементів змагання у навчальний процес та допомозі учням усвідомлювати свій прогрес у вивченні матеріалу [2] .

Сьогодні історія України є важливою частиною національної культури та самоідентифікації. Вона є фундаментом для розуміння сучасного світу та формування громадянської позиції. Знання історії є важливим для збереження культурної спадщини та розуміння ключових подій. Однак, навчання історії може бути складним та нудним процесом. Навчальні ігри є ефективним способом зробити навчання більш цікавим та захопливим, а адаптивне навчання дозволяє персоналізувати навчання для кожного учня.

Отже, розробка навчальної гри є актуальною, оскільки існує недостатньо подібних додатків для вивчення історії, що свідчить про передбачуваний попит на такі ігри. Використання методів адаптивного навчання та створення навчальної гри для Android дозволяє впроваджувати інноваційні підходи до навчання, які можуть бути ефективними для різних категорій користувачів. Подібні навчальні додатки мають потенціал стати цінними ресурсами для освіти

та розваги, особливо в епоху високорозвинених технологій та популярності мобільних пристроїв.

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

Мета та завдання дослідження. Метою магістерської кваліфікаційної роботи є підвищення ефективності якості навчального процесу шляхом розробки навчальної гри для Android-додатку, спрямованої на вивчення історії України з використанням методів адаптивного навчання та створенням інтерактивного навчального середовища, яке автоматично пристосовується до поточного рівня знань та індивідуальних потреб користувачів, сприяючи ефективному і цікавому процесу вивчення історії України.

Основними задачами роботи є:

- визначити функціонал Android-додатку для вивчення історії України та засоби його реалізації;
- розробити метод індивідуалізації навчального процесу з використанням адаптивного методу навчання;
- розробити модель нарахування балів для навчальної гри;
- провести аналіз інформаційного забезпечення додатку;
- розробити моделі та алгоритми роботи навчальної гри;
- розробити програмне забезпечення навчальної гри;
- провести тестування роботи Android-додатку.

Об'єктом дослідження є процес розробки навчальної гри з використанням методу адаптивного навчання.

Предметом дослідження є методи та засоби розробки навчальної гри для мобільних пристроїв.

Методи дослідження. У процесі досліджень використовувались методи дослідження:

- методи розробки мобільного додатку для операційної системи Android, для реалізації навчальної гри;

- методи аналізу інформації для створення навчального матеріалу та формулювання завдань для тестування;
- методи індивідуалізації процесів, які включають розробку алгоритмів та використання системи рекомендацій, для адаптації процесу підбору завдань до рівня знань користувачів;
- методи тестування для перевірки працездатності додатку.

Наукова новизна отриманих результатів.

- Подальшого розвитку дістав метод індивідуалізації навчального процесу, який, на відміну від існуючих, спрямований на адаптацію навчання для кожного учня з метою оптимізації розуміння і запам'ятовування матеріалу та враховує особливості кожного учня, його потреби, рівень знань і темп навчання, що надає можливість налаштувати навчальний процес під конкретну ситуацію.
- Подальшого розвитку дістала модель нарахування балів та отримання бонусів для навчальної гри, яка, на відміну від існуючих, орієнтована на мотивацію, що надає бонусні бали за досягнення певних цілей або завдань та адаптує рівень складності завдань для кожного учасника на основі його рівня знань і навичок, забезпечуючи зручний режим роботи для користувачів.

Практична цінність отриманих результатів. Практична цінність результатів магістерської кваліфікаційної роботи полягає у можливості практично застосовувати розроблений Android-додаток як навчальну гру для ефективного вивчення історії України.

Особистий внесок здобувача. Усі наукові результати, викладені у магістерській кваліфікаційній роботі, отримані автором особисто. У науковій роботі [3], опублікованій у співавторстві, автору належить розробка алгоритму роботи навчальної ігрової програми.

Апробація матеріалів магістерської кваліфікаційної роботи. Результати магістерської кваліфікаційної роботи доповідалися та обговорювалися на міжнародній науково-практичній Інтернет-конференції

«Електронні інформаційні ресурси: створення, використання, доступ - 2023».

Публікації. Основні результати дослідження опубліковані в тезах доповіді на міжнародній науково-практичній Інтернет-конференції «Електронні інформаційні ресурси: створення, використання, доступ - 2023» [3].

Структура та обсяг роботи. Магістерська кваліфікаційна робота складається зі вступу, п’яти розділів, висновків, списку використаних джерел, що містить 33 найменування, 4 додатки. Робота містить 44 ілюстрації, 13 таблиць.

1 АНАЛІЗ СТАНУ РОЗВИТКУ НАВЧАЛЬНИХ ІГОР ТА ПОСТАНОВКА ЗАДАЧ ДОСЛІДЖЕННЯ

1.1 Аналіз стану розвитку навчальних ігор для Android-додатків

У сучасному світі, де мобільні пристрої стали неодмінною частиною нашого щоденного життя, розробка навчальних ігор для Android-додатків набуває все більшої актуальності. Ці ігри можуть використовуватися для навчання, отримання різноманітних навичок, можуть сприяти розвитку мислення та креативності, а також підвищувати мотивацію до навчання.

Розвиток ринку мобільних додатків надає розробникам можливість пристосовуватися до змінних потреб користувачів і залишатися конкурентоспроможними. Отримання інформації про тенденції та можливості мобільної розробки допомагає створювати інноваційні та сучасні рішення, які відповідають новим вимогам ринку.

У першому розділі проведемо аналіз стану розвитку навчальних ігор для Android-додатків, звертаючи увагу на їх оригінальність та джерела подальшого розширення. [4].

Платформа Android надає розмаїття програм для вдоволення різноманітних потреб і інтересів користувачів. Серед соціальних мереж особливо варто відзначити програми, які сприяють взаємодії та обміну мультимедійним вмістом. У фінансовому секторі присутні банківські додатки та електронні гаманці, що спрощують фінансовий менеджмент і безготівкові операції.

Мультимедійні розваги охоплюють музичні та відеоплеєри, а також різноманітні ігри – від головоломок до стратегій. Додатки для навчання пропонують уроки мови, головоломки та інтерактивні уроки. Картографічні служби, фоторедактори, інструменти підвищення продуктивності, засоби зв'язку, новинні програми та інструменти персоналізації розширюють можливості користувача в усіх аспектах.

Ця різноманітність визначає успіх Android як платформи, яка задовольняє вподобання та вимоги широкого кола користувачів, розширюючи їхні можливості у сферах розваг, навчання, фінансів і багатьох інших [5].

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. Із часом з'являлися все більш складні та захоплюючі навчальні ігри, які використовували різноманітні методи навчання та взаємодію.

Ці 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.

Зараз багато ігор використовують гейміфікацію, яка є популярним підходом в ігровій індустрії. 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.

У цілому, навчальні ігри для мобільних пристроїв стали потужним інструментом для навчання та розвитку навичок, який знаходить своє використання як у дитинстві, так і в дорослому житті [6].

У наш час навчальні ігри для Android-додатків представляють собою різноманітні та захопливі засоби, що дозволяють учням засвоювати нові знання та навички у захоплюючій формі. Ці ігри можуть бути використані для навчання різних предметів та навичок, і вони мають численні переваги перед традиційними методами навчання.

Навчальні ігри стають все більш популярними в освітньому контексті і використовуються як допоміжні засоби в класі, інструменти для самонавчання та засоби для оцінювання знань учнів. Технічні аспекти продуктів можуть вирізнятися за наявністю підтримки мобільних платформ, оптимізацією під різні

пристрої, а також можливістю використовувати сучасні технології, такі як штучний інтелект або адаптивне навчання.

Навчальні ігри можуть бути реалізовані для індивідуального навчання, враховуючи унікальні потреби та інтереси кожного учня, що дозволяє їм здобувати знання та навички у власному темпі. Ці ігри також виступають альтернативою традиційним методам оцінювання, надаючи інший спосіб визначити рівень знань учнів. Узагальнено, освітні ігри для Android є не лише доповненням, але і ефективним інструментом для розширення навчання та стимулювання цікавості до отримання знань.

При розробці навчальних ігор для додатків Android важливо враховувати оригінальність як один із ключових факторів. Цей процес може бути складним, але водночас вкрай корисним для навчання. Зосереджуючись на цільовій аудиторії, меті гри, геймплеї та освітньому контексті, розробники можуть створити унікальну гру, що буде цікавою та пізнавальною для користувачів.

Оригінальність є важливим фактором для того, щоб виділити свою гру серед конкурентів та привернути увагу потенційних користувачів. Багато розробників прагнуть створювати ігри з унікальними концепціями та ідеями, які роблять їх особливими. Деякі використовують головоломки, що вимагають від гравця застосування математичних або логічних розрахунків, тоді як інші створюють інтерактивні історії, де гравець приймає рішення, впливаючи на розвиток подій.

Це підкреслює важливість створення навчальних ігор, які не лише передають інформацію, але й роблять цей процес цікавим та викликаючим. Такі ігри можуть використовувати різні методи навчання, розвивати логічне мислення та творчість, тим самим створюючи ефективні інструменти для освіти[7].

Отже, індустрія освітніх ігор для Android-додатків є новою та сприятливою сферою. Освітні ігри сучасності включають різноманітні методи навчання та пропонують можливість розвивати низку навичок. Розробка унікальних та

інноваційних ігор є важливою складовою цієї галузі, оскільки вона виділяє серед конкурентів і привертає інтерес користувачів.

1.2 Порівняльний аналіз аналогів

Сьогодні існує безліч розвиваючих ігор на різні теми. Найпопулярнішими серед них є ігри з таких предметів, як математика, логіка та англійська мова. Українські історичні ігри теж популярні, але їх кількість значно менша, ніж ігор на іншу тематику.

Використання методу порівняння у дослідженні об'єктів є актуальним, особливо тоді, коли ці об'єкти мають аналогічні або спільні характеристики. Цей підхід включає індивідуальний аналіз кожного об'єкта, з виділенням критеріїв для подальшого порівняння. Під час безпосереднього порівняння визначаються як загальні, так і відмінні риси.

Процес порівняння дозволяє виявити переваги та недоліки об'єктів, аналізувати їхні схожі та відмінні характеристики. Отримані результати можуть бути використані для формулювання висновків щодо ефективності, якості або інших аспектів досліджуваних об'єктів. Важливо також визначити можливості для подальшого удосконалення або розвитку.

Метод порівняння є ефективним інструментом в дослідженнях, де необхідно об'єктивно оцінювати та ранжувати різні аспекти аналогічних об'єктів чи явищ. Його використання дозволяє отримати глибокий інсайт у властивості та характеристики об'єктів, сприяючи більш обґрунтованому прийняттю рішень та розробці подальших стратегій.

«ЗНО 2024. Історія України» (рис. 1.1) – це додаток, який допоможе користувачам ефективно вивчити всю необхідну інформацію, яка вимагається програмою ЗНО. Основною перевагою даного додатку є зрозумілий та цікавий інтерфейс, також додаток можна використовувати для підготовки до тестових завдань. Він містить банк тестових завдань, які дозволять користувачам потренуватися в розв'язанні завдань ЗНО. Картки поділені за темами та видами, що полегшує вивчення. Вони розроблені досвідченими фахівцями в галузі історії

України та регулярно оновлюються. Розробником даного додатку є Oleh Karmazyn. Але значним недоліком є те, що, оскільки додаток є частково безкоштовним, він розміщує багато реклами [8].

Рисунок 1.1 – Інтерфейс програми «ЗНО 2024. Історія України»

«Історія України» (рис. 1.2) – це мобільний додаток, який дозволяє користувачам вивчати історію України через ігри. Додаток має різноманітні ігри, які охоплюють різні періоди історії України, від козацьких часів до сучасності. Однією з переваг додатку є його захоплюючий та цікавий геймплей, ще додаток дозволяє закріпити отримані знання та навички за допомогою інтерактивних завдань і тестів. Це допомагає користувачам краще зрозуміти матеріал і навчитися розв'язувати тестові завдання. Гравці можуть випробувати свої знання з історії України, виконуючи завдання та розв'язуючи головоломки. Крім того, додаток має графічні ефекти та звукові ефекти, що робить гру ще цікавішою. Гра

розроблена компанією ExcitingGames.io. Проте, додаток також має деякі недоліки. Перш за все, він може бути відносно обмежений у своїй інформації про історію України, також немає реєстрації та адаптивного методу навчання, що є важливим аспектом для зацікавлення навчання у додатку [9].

Рисунок 1.2 – Інтерфейс програми «Історія України»

«ЗНО 2023: Історія України» (рис. 1.3) – цей додаток створено для того, щоб полегшити підготовку до ЗНО 2023 з історії України. Унікальна система вивчення дозволяє вчитися, розв'язуючи тести та отримуючи зворотний зв'язок при неправильних відповідях. Результати в тестах покращуються, що говорить про зростання знань та підготовки. Додаток також пропонує нестандартні тести,

такі як визначення правдивості фактів і "Вірю/Не вірю", які розвивають точність та уважність. Додаток використовує систему засвоєння матеріалу через вирішення тестів. При неправильній відповіді показується правильний варіант, що сприяє кращому запам'ятовуванню матеріалу. Гравець має можливість покращувати свої результати в тестах, намагаючись набрати якнайбільше балів за один раз. Це сприяє постійному вдосконаленню знань з історії та підготовці до екзаменів. Графічна частина виглядає також професійною, що і є відповідно зручною для використання. Додаток має простий і інтуїтивно зрозумілий інтерфейс. Виробником даної гри є D is a number!. Під час тестування додатку було знайдено недоліки та помилки у тестах, також даний додаток є частково безкоштовним, оскільки має платну частину та безліч реклами [10].

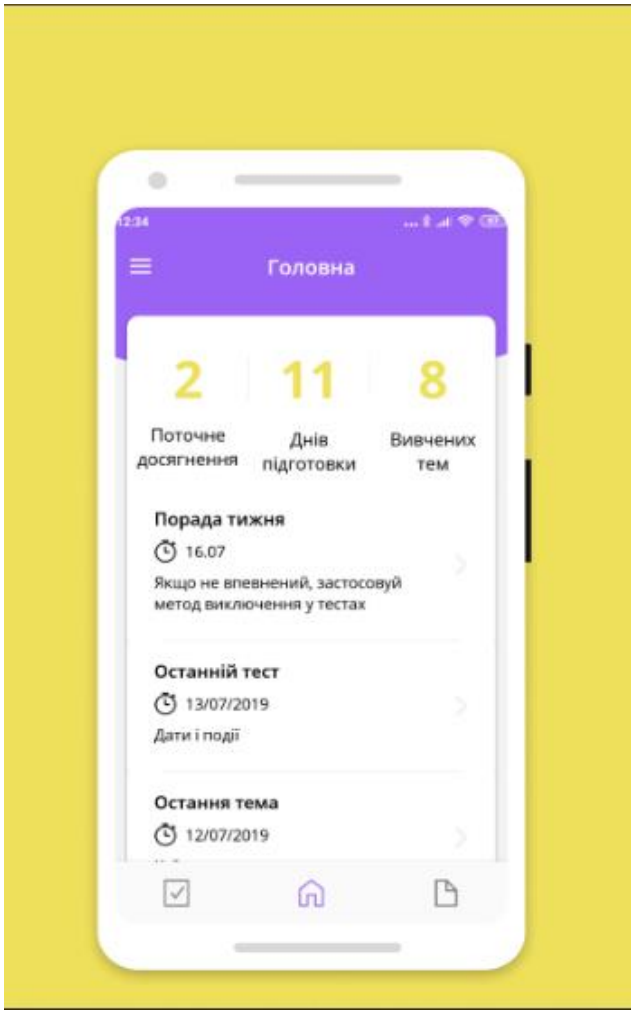

Рисунок 1.3 – Інтерфейс програми «ЗНО 2023: Історія України»

У результаті докладного порівняльного аналізу схожих додатків варто зауважити, що кожен з них володіє унікальними особливостями та відмінностями. Кожен додаток вирізняється своєю методологією та функціональністю, що відповідає конкретним потребам користувача. Важливо враховувати ці особливості при виборі найбільш підходящого додатка для конкретних цілей при вивченні історії України. Однак, з урахуванням того, що багато функцій є загальними, можна виділити ключові критерії для ефективного порівняння програмних продуктів. Результати аналізу аналогів зведено в таблицю 1.1.

Таблиця 1.1 – Порівняльна характеристика навчальних програмних продуктів для вивчення історії України

Критерії	ЗНО 2024. Історія України	Історія України	ЗНО 2023: Історія України	Власна реалізація
1	2	3	4	5
Реєстрація та авторизація	+	-	-	+
Різновидність режимів навчання	+	+	+	+
Наявність адаптивного методу навчання	-	-	-	+
Наявність онлайн бази даних	-	-	+	+
Безкоштовна ліцензія	+-	+	+-	+
Інтерфейс українською мовою	+	+	+	+
Підсумковий результат	3,5	3	3,5	6

Таблиця порівняння характеристик навчальних програмних додатків показала, що розробка власного програмного продукту є доцільною. По-перше, він використовує методи адаптивного навчання, які дозволяють підлаштовувати складність гри під рівень знань гравця. По-друге, він містить велику кількість навчального матеріалу з історії України. По-третє, він має зручний та інтуїтивно зрозумілий інтерфейс.

Таким чином, власна реалізація є більш ефективним і зручним інструментом для вивчення історії України, ніж існуючі аналоги.

1.3 Аналіз методів розв'язання задачі

Ефективність навчального процесу визначається розумінням, правильним вибором та кваліфікованим застосуванням методів, принципів і засобів навчання. Методи навчання, які сприяють підвищенню зацікавленості, активності, ефективності та результативності, є важливими компонентами освітнього процесу.

У сучасному світі мобільні додатки стають надзвичайно популярним інструментом використання інформаційних технологій. Кількість користувачів, які використовують додатки на своїх смартфонах і планшетах, постійно зростає. Це зумовлено поширенням смартфонів і планшетів, а також постійним розвитком технологій в галузі мобільних додатків.

Мобільні додатки в освіті можуть допомагати у покращенні навчального процесу, роблячи його більш доступним і цікавим. Вони можуть забезпечувати інтерактивні уроки, тестування, сприяти самонавчанню та забезпечувати доступ до освітніх ресурсів у будь-який час і в будь-якому місці. Використання мобільних додатків у навчанні може покращити залучення студентів та зробити процес більш ефективним.

Розробка мобільних додатків здійснюється на основі існуючих платформ, таких як Android або iOS. Для цього використовуються інтегровані середовища розробки (IDE), які надають розробникам широкий спектр інструментів і

можливостей, зокрема для створення коду, проведення тестування, вирішення завдань і планування впровадження [11].

Розробка інтерфейсу користувача — це процес покращення представлення та взаємодії веб-або мобільної програми, зосереджуючись на тому, як програма виглядає та як вона взаємодіє з користувачами. Інтерфейс користувача включає блоки тексту, які користувачі читають, кнопки, текстові поля, форми, зображення та інші візуальні елементи, які користувачі спостерігають і з якими взаємодіють під час використання програми.

Підхід MVVM розділяє додаток на три частини, щоб спростити розробку графічного інтерфейсу та бізнес-логіки. Модель відповідає за зберігання даних і бізнес-логіку, представлення - за відображення даних на екрані, а модель представлення - за зв'язок між ними. MVVM має ряд переваг, зокрема зручність у розробці та підтримці, масштабованість і придатність для тестування. ViewModel є зв'язком між моделлю та представленням. Він отримує дані з моделі та відображає їх у представленні. ViewModel також відповідає за оновлення даних у моделі. Візуальні компоненти, такі як кнопки, взаємодіють із ViewModel за допомогою команд.

Один із головних аргументів у виборі MVVM є розділення відповідальностей між компонентами додатку. Модель (Model) відповідає за управління даними та бізнес-логікою, представлення (View) забезпечує відображення графічного інтерфейсу користувача, а ViewModel діє як посередник між Model і View, забезпечуючи двосторонню зв'язок та обмін даними [12].

LiveData є ключовим компонентом архітектури Android, призначеним для зберігання даних у ViewModel та автоматичного відстеження їх змін за допомогою оновлень компонентів інтерфейсу користувача. LiveData можна описати як клас, який утримує дані та автоматично сповіщає спостерігачі про зміни цих даних.

Однією з ключових особливостей LiveData є його усвідомлення життєвого циклу компонентів програми, до яких він підключений. Це означає, що LiveData

автоматично припиняє оновлення спостерігачів, якщо компонент програми стає неактивним або переходить у стан зупинки. Такий підхід допомагає уникнути проблем, пов'язаних з витоком ресурсів та непотрібними оновленнями в неактивних компонентах.

LiveData забезпечує зручний механізм для реалізації архітектурного підходу "спостереження за даними", спрощуючи роботу розробників та забезпечуючи ефективний обмін даними між ViewModel та інтерфейсом користувача[13].

Є низка переваг у використанні LiveData, такі як:

- забезпечення відповідності інтерфейсу користувача стану даних;
- відсутність збоїв через припинення діяльності;
- завжди актуальні дані;
- обмін ресурсами;
- правильні зміни в конфігурації.

ViewBinding – це інструмент, який спрощує отримання доступу до елементів інтерфейсу користувача (UI) в коді Android-додатка. Його включення для модуля призводить до автоматичного створення файлу класу для кожного XML-файлу макету в цьому модулі. Кожен з цих класів містить посилання на всі елементи UI, які мають ідентифікатор у відповідному макеті.

ViewBinding використовується для автоматизації зв'язування макетів із кодом додатка в Android-розробці. Його використання дозволяє безпечний доступ до елементів користувацького інтерфейсу і допомагає уникнути проблем, пов'язаних з нульовими значеннями, оскільки він гарантує, що всі зв'язування відбуваються лише з існуючими елементами в макеті. Це полегшує розробку та підтримку коду, а також дозволяє уникнути можливих помилок, пов'язаних із некоректним доступом до елементів інтерфейсу.

Основною перевагою ViewBinding є його сумісність як з Java, так і з Kotlin. Використання цього інструменту спрощує код і зменшує його обсяг, оскільки не потрібно вручну виконувати зв'язування кожного елемента із XML-файлом макету. При роботі з ViewBinding слід дотримуватися правил іменування файлів

макетів, оскільки він автоматично створює класи для зв'язування на основі назв файлів. Це спрощує розуміння коду та його підтримку.

Узагальнюючи, ViewBinding є корисним інструментом, що підвищує продуктивність розробників, полегшуючи роботу з елементами інтерфейсу та сприяючи зниженню кількості коду. Використання цього підходу дозволяє ефективно працювати як у середовищі Java, так і у середовищі Kotlin, забезпечуючи зручний та чистий спосіб здійснення зв'язування між компонентами макету та програмною логікою додатка [14].

Інтерфейс програмного забезпечення можна визначати в файлах XML, що дозволяє відокремити його від коду і робить більш гнучким та легким для змін.

Оголошення інтерфейсу в XML полегшує візуалізацію та налагодження, а також спрощується тим, що елементи інтерфейсу в XML відповідають класам та методам у коді.

При створенні інтерфейсу в XML важливо дотримуватися деяких правил, таких як наявність одного кореневого елемента у кожному файлі розмітки, який представляє об'єкт View або ViewGroup.

У розглянутому програмному забезпеченні використовується компонент TableRow. Кожен об'єкт TableRow автоматично отримує ідентифікатор від tableRow1 до tableRow6, а їх властивість LayoutWidth встановлюється в match_parent. Це дозволяє рядкам займати весь доступний простір макета. Також, важливо відзначити, що використання XML для визначення інтерфейсу користувача дозволяє зберігати його в окремих файлах, відокремлюючи його від Java-коду.

Цей підхід має кілька переваг. По-перше, він спрощує розробку, оскільки інтерфейс може бути налаштований та змінюваний без необхідності втручання в Java-код. По-друге, він полегшує та прискорює процес змін у вигляді програми, оскільки зміни в XML-файлах відразу відображаються у візуальному представленні інтерфейсу без перекомпіляції коду. Такий підхід є зручним для розробників та дозволяє ефективно управляти інтерфейсом додатка.

Можна створити різні макети для різних типів пристроїв, орієнтацій екрану чи мов користувачів. Всі ці аспекти можуть бути легко налаштовані у файлах XML без потреби втручання у сам код програми. Крім того, XML-файли мають зрозумілу структуру, що полегшує їх читання та редагування, а також сприяє збереженню налагоджувальної зручності.

Компоненти GUI розташовані в шести рядах і чотирьох стовпцях, при цьому кожен рядок представлений об'єктом TableLayout. Кожен рядок може бути порожнім або містити інші компоненти, такі як макети, що включають інші елементи [15].

1.4 Постановка задач дослідження

Проаналізувавши питання розробки навчальної гри з історії України для Android-додатку з використанням методів адаптивного навчання, було визначено завдання, які необхідно виконати для розробки програмного додатку.

Основними задачами роботи є:

- визначити функціонал Android-додатку для вивчення історії України та засоби його реалізації;
- розробити метод індивідуалізації навчального процесу з використанням адаптивного методу навчання;
- розробити модель нарахування балів для навчальної гри;
- провести аналіз інформаційного забезпечення додатку;
- розробити моделі та алгоритми роботи навчальної гри;
- розробити програмне забезпечення навчальної гри;
- провести тестування роботи Android-додатку.

1.5 Висновки

У першому розділі було розглянуто стан розвитку навчальних ігор з історії України для Android-додатків з використанням методу адаптивного навчання.

Під час аналізу були розглянуті існуючі аналоги, такі як додатки «ЗНО 2024. Історія України», «Історія України», «ЗНО 2023: Історія України», і

проведено їх порівняння з власним програмним продуктом. У результаті цього порівняння було обґрунтовано доцільність розробки власного додатку.

Основні переваги власного програмного продукту включають його повну безкоштовність, можливість ведення статистики досягнень гравців та реалізацію контролю через тестування з різними режимами проходження тестів. На відміну від існуючих аналогів, додаток надає користувачам не лише теоретичні матеріали, але й активно взаємодіє з ними, забезпечуючи ефективний контроль.

Такий підхід до розробки програмного продукту вказує на його конкурентні переваги та спрямованість на задоволення потреб користувачів у навчанні та контролі знань з історії України.

Проведено аналіз існуючих методів для вирішення поставленої задачі. У результаті було обрано метод адаптивного навчання з використанням тестування для розробки навчальної гри з історії України для Android-додатку.

Сформульовано основні завдання, які необхідно виконати для розробки програмного додатку.

2 РОЗРОБКА МЕТОДУ ТА МОДЕЛЕЙ НАВЧАЛЬНОЇ СИСТЕМИ

2.1 Аналіз інформаційного забезпечення

У системі навчання, розробленій з орієнтацією на інформаційне забезпечення, увагу зосереджено на ключових аспектах для максимального зручності та ефективності роботи користувачів. Основний функціонал спрямований на глибоке освоєння теоретичних знань з історії України та підтримку інтерактивного навчання.

Система має базу даних. У базі даних є інформація про користувачів, їхні логіни та паролі, також перелік різноманітних запитань та відповідей на тестові завдання. Також дана система має модуль навчання, який може працювати в двох режимах. Перший із них адаптивний, що дозволяє проходити тести з підлаштовуванням під можливості користувача та його знання в процесі підбору наступних питань, щоб зацікавити гравця. Другий модуль є навчальним, який дозволяє проходити всі тести без адаптації до знань користувача.

Додаток також має бонусну програму – це нарахування балів та монет, що дозволяє дізнатися свої середні бали, та можливість використовувати монети при підказках та купівлі доступу до деяких тем.

Головним завданням є створення захоплюючого навчального середовища через застосування цікавих тестів, які розраховані на оцінку рівня складності. Адаптивний метод навчання забезпечує персоналізований підхід до кожного користувача, стимулюючи його активність та інтерес до навчання.

Система дозволяє користувачам вибирати свій власний шлях у навчанні: лінійно проходити тести або скористатися адаптивними тестами, що динамічно адаптуються до попередніх відповідей. Такий індивідуальний підхід допомагає уникнути нудьги та підвищує ефективність освоєння матеріалу.

Важливим аспектом є логічна послідовність у навчанні, яка дозволяє користувачам пройти завдання від простого до складного. Завдяки автоматизованій системі тестування, рівень знань користувача оцінюється об'єктивно, і студент може обрати свій варіант подальшого розвитку. Всі ці

аспекти об'єднуються з метою зробити процес навчання цікавим, ефективним та максимально адаптованим до потреб кожного користувача.

Додаткові програмні модулі для розширення функціональності навчальної системи можуть включати:

- авторизацію та реєстрацію користувача;
- блок теоретичного матеріалу;
- модуль перевірки знань шляхом тестування;
- використання адаптивного методу навчання для зручності проходження тестів;
- ведення статистики проходження тестів;
- модуль мотивації та нагородження.

Додавання таких модулів допоможе розширити можливості навчальної системи та зробити її більш привабливою та ефективною для користувачів.

2.2 Розробка загальної моделі навчальної системи

Розробка навчальної гри "Історія України" націлена на створення індивідуалізованого та автоматизованого навчального середовища для кінцевих користувачів. З технічної точки зору це передбачає швидкий та ефективний доступ до інформації, яка буде використовуватися для автоматизованого тестування. Важливим аспектом є здатність системи отримувати та оновлювати дані, необхідні для створення індивідуалізованих тестів.

Крім того, з технічної точки зору, гра розробляється з метою ефективного зберігання та обробки інформації про кожного користувача. Це включає в себе здатність системи відстежувати та зберігати прогрес користувачів у навчанні, що є ключовим елементом для створення індивідуальних навчальних траєкторій та надання користувачам персоналізованого досвіду. Технічна реалізація таких функцій передбачає використання ефективної бази даних, алгоритмів аналізу прогресу та забезпечення безпеки зберігання особистих даних користувачів.

Загальна модель роботи програми зображена на рисунку 2.1

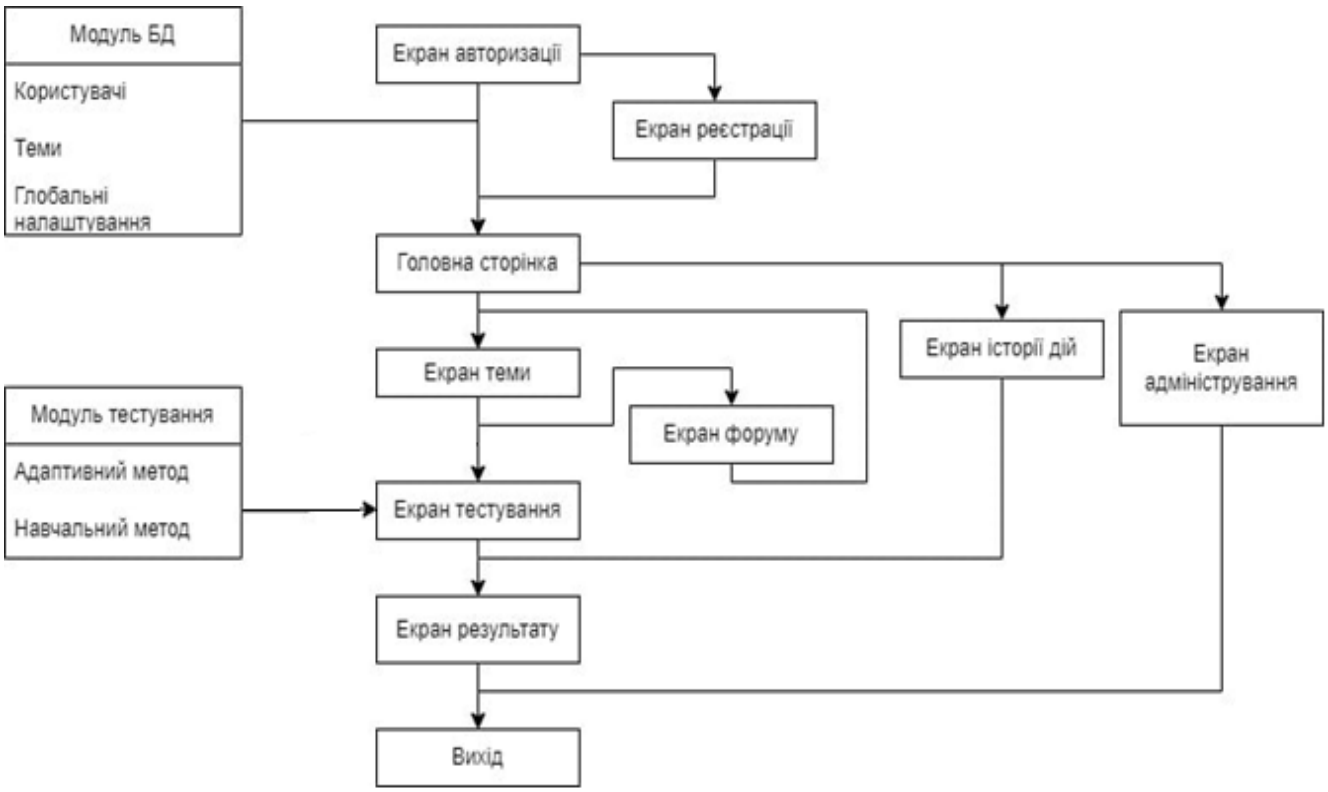

Рисунок 2.1 – Загальна модель роботи програми

Для управління базами даних використовуються спеціальні системи. До найпопулярніших систем управління базами даних можна віднести MySQL, PostgreSQL та Firebase.

MySQL – це система управління базами даних, яка ґрунтується на реляційній моделі і вирізняється своєю відкритістю та широким застосуванням у великих компаніях. Однією з її переваг є висока продуктивність, оскільки вона може ефективно обробляти великі обсяги даних і підтримує різноманітні типи таблиць. MySQL не обмежує кількість користувачів, які можуть одночасно взаємодіяти з базою даних.

Для адміністрування та керування базою даних зручно використовувати інтерфейс phpMyAdmin через браузер, що полегшує взаємодію з базою даних. Цей інструмент надає зручний спосіб виконання різноманітних операцій, таких як створення таблиць, додавання та видалення даних, виконання запитів SQL та багато іншого, що робить роботу з MySQL більш ефективною та доступною [16].

PostgreSQL – це об’єктно-реляційна система управління базами даних, доступна у вільному доступі. Вона славиться своєю гнучкістю, не обмежуючи розмір бази даних чи кількість записів. PostgreSQL відрізняється високою надійністю транзакцій та можливістю наслідування, що робить її популярною серед розробників. Детальна документація допомагає вирішити практично будь-які питання, пов'язані з використанням системи [17].

Firebase – це розширена платформа для розробки мобільних додатків. Однією з головних переваг є можливість уникнути створення власного backend, оскільки Firebase надає інтегровані можливості для серверного коду, бази даних, хостингу та аутентифікації. Firebase Realtime Database дозволяє синхронізувати дані між клієнтами та зберігати їх в хмарному сховищі, сприяючи ефективній та зручній роботі додатків [18].

Firebase Realtime Database має низку переваг, які роблять її привабливим вибором для розробки мобільних додатків:

1. Синхронізація в реальному часі. Firebase Realtime Database використовує протокол WebSockets для синхронізації даних між клієнтами в реальному часі. Це дозволяє розробникам створювати динамічні та взаємодіючі додатки, які завжди відображають останні дані.

2. Проста інтеграція. Firebase Realtime Database легко інтегрується з іншими продуктами Firebase, такими як Firebase Authentication і Firebase Cloud Functions. Це дозволяє розробникам створювати комплексні додатки, які використовують різні функції Firebase.

3. Гнучкість. Firebase Realtime Database є хмар-орієнтованою базою даних NoSQL. Це означає, що вона не має жорстких обмежень щодо структури даних. Це дозволяє розробникам створювати додатки, які відповідають їхнім конкретним потребам.

Для взаємодії навчальної гри з базою даних було обрано Firebase.

На рисунку 2.2 наведено узагальнену модель бази даних розроблюваного мобільного додатку.

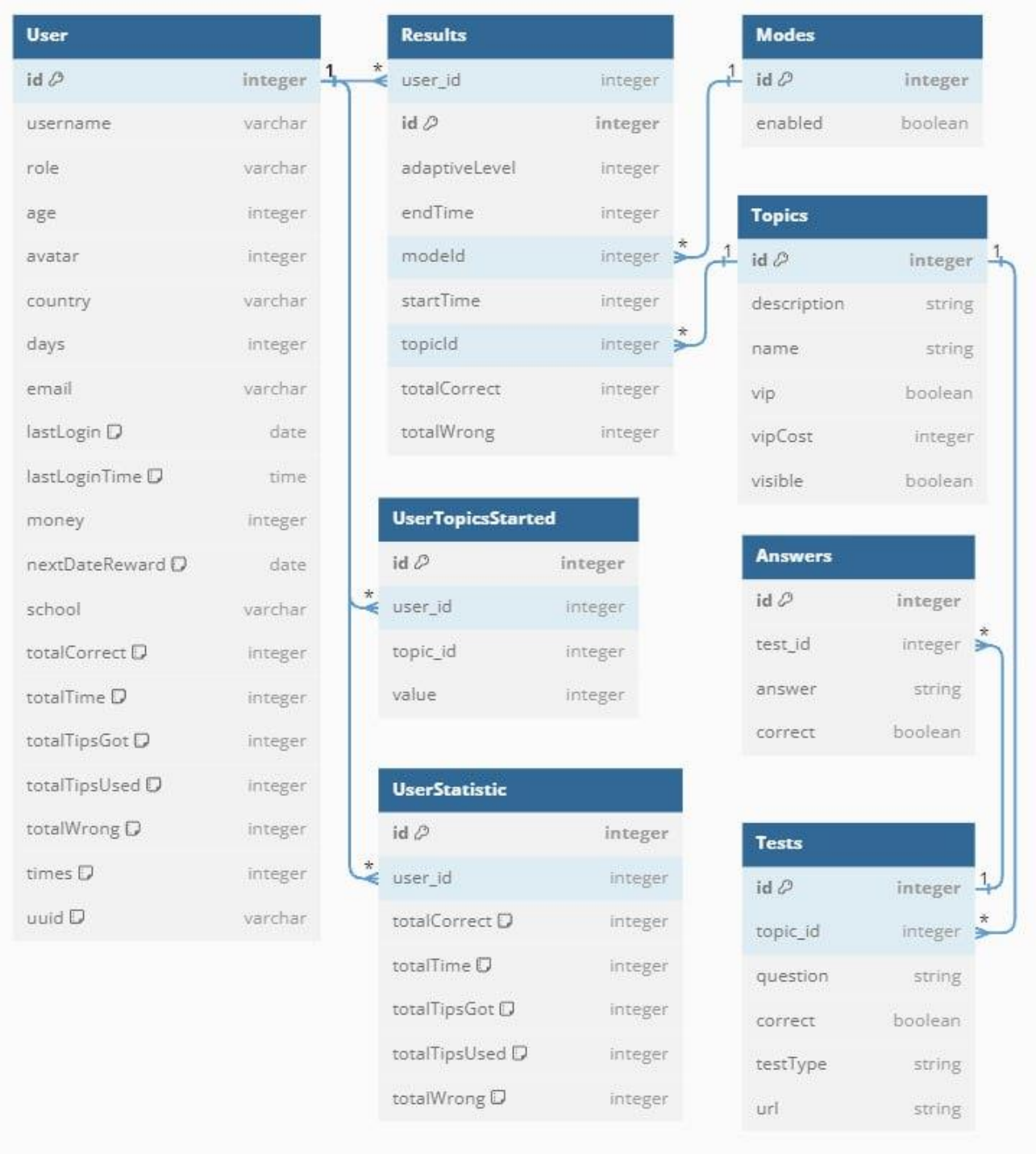

Рисунок 2.2 – Узагальнена модель бази даних програми

2.3 Розробка методу індивідуалізації навчального процесу для навчальної гри з адаптивним методом навчання

Індивідуалізація в освітньому процесі передбачає врахування унікальних характеристик кожного учня, таких як його особисті властивості та рівень готовності до навчання, при виборі методів, підходів і темпу навчання.

Індивідуалізація у навчанні включає в себе можливість учнів працювати з навчальними програмами власним темпом і здолати труднощі власними силами. Ця індивідуалізація також передбачає адаптацію навчальної програми до типу мислення кожного учня і відповідне вивільнення часу вчителя для індивідуальної роботи з учнями. Такий підхід спрямований на автоматизацію рутинних завдань у педагогічній роботі та дозволяє кожному учню навчатися відповідно до власних потреб.

У навчальному процесі принцип індивідуалізації виявляється через урахування індивідуально-психологічних особливостей учнів, які мають значний вплив на їх успішність у вивченні основ предметної області. Для забезпечення індивідуалізованого навчання необхідно, щоб вчитель був інформований про ці особливості своїх учнів та вмів використовувати методи індивідуалізованої освіти [19].

Адаптивне навчання (також відоме як адаптивне викладання) – це процес освіти, в якому застосовуються спеціальні алгоритми для індивідуалізації навчального процесу шляхом створення персоналізованого навчального шляху з використанням підібраних ресурсів і активностей, які відповідають унікальним потребам кожного студента.

Ця концепція базується на розумінні того, що люди мають різний темп розвитку та індивідуальні особливості.

Отже, для забезпечення оптимального професійного розвитку студента, навчання повинно бути адаптованим під його особисті запити, сприйняття, темп навчання, а також враховувати наявний досвід та вміння студента, а також виявляти і заповнювати прогалини в знаннях, які можуть бути як усвідомленими, так і неусвідомленими [20].

В додатку передбачено можливість вибору навчання, де є можливість користуватися звичайним тестом та з адаптивним методом навчання,відповідно до знань користувача. На рисунку 2.3 зображено перемикач (Toggle) для вибору навчання.

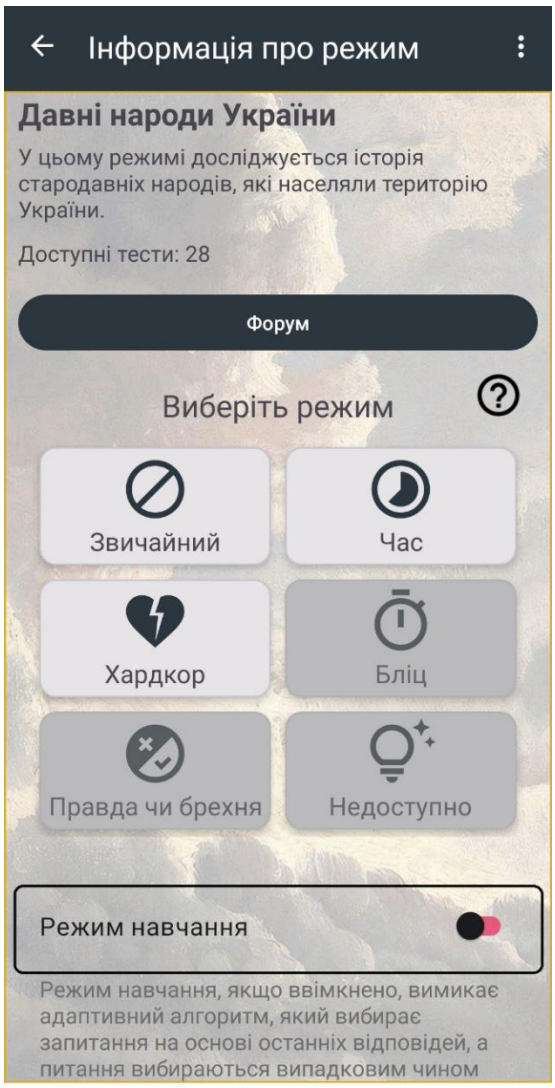

Рисунок 2.3 – Екран з перемикачем (Toggle) для вибору навчання

Розроблений метод індивідуалізації навчального процесу для навчальної гри з адаптивним методом навчання включає послідовність дій:

1.Для користування додатком користувач повинен зареєструватися та увійти до свого аккаунту за допомогою методу register, який приймає пошту та пароль, та методу логін, який також приймає пошту та пароль.

2. Користувач обирає тему і за допомогою слухача кліків ClickListener списку RecyclerView визначається, яка саме тема і питання будуть використані для проходження тесту.

3. У тестуванні визначенні методи onFail і onCorrect, які регулюють адаптивний режим збільшенням лічильника і в кінці яких викликається метод обмеження, щоб не перевищити рівень адаптивного режиму.

4. При проходженні тесту за адаптивним методом навчання користувач обирає відповіді, правильність яких фіксується функцією onCorrect, яка збільшує лічильник для адаптивного навчання на 1.

5. Якщо користувач надав у безперервному режимі 5 правильних відповідей, то в методі onCorrect збільшується лічильник на 2.

6. При тому, коли користувач дає неправильні відповіді, то викликається onFail і рахує кількість неправильних відповідей, отримані в безперервному режимі, якщо їх кількість більше або дорівнює 3, то значення лічильника зменшується на 1.

На рис. 2.4 зображено блок-схему алгоритму роботи адаптивного методу навчання.

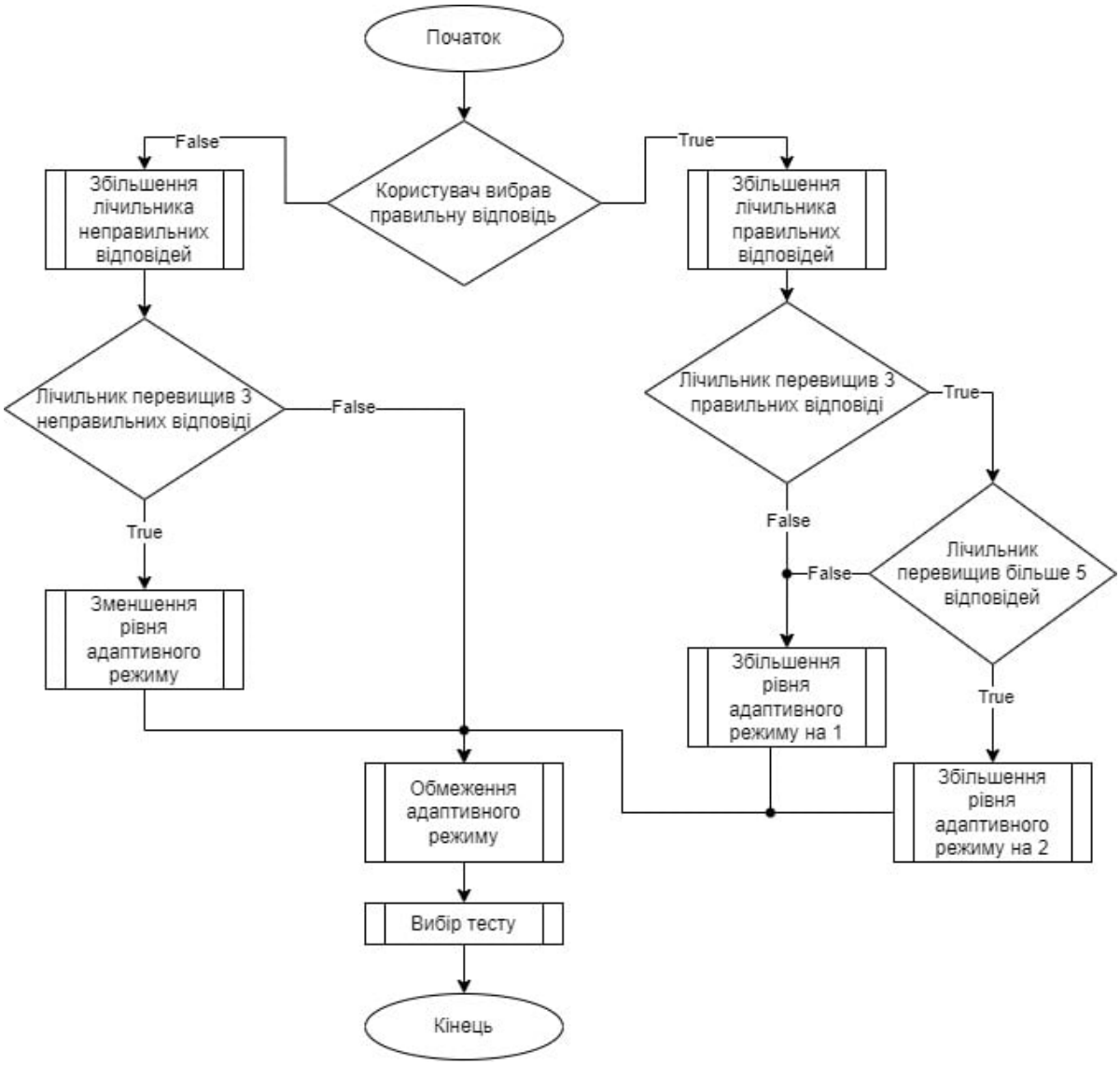

Рисунок 2.4 – Блок-схема алгоритму роботи адаптивного методу навчання

2.4 Розробка алгоритму тестування та моделі нарахування балів і бонусів

Тест – це набір завдань, які дозволяють оцінити рівень знань і навичок учня з певної навчальної дисципліни на основі отриманих результатів виконання цих завдань. Тестування являє собою метод, який дозволяє оцінити рівень знань та вмінь користувачів, використовуючи різноманітні тестові завдання, такі як питання, завдання на вибір, завдання з короткими відповідями тощо. Цей процес допомагає визначити, наскільки є зрозумілим матеріал і наскільки учні вміють застосовувати свої знання.

Адаптивне тестування – це форма тестування, де послідовність або складність запитань змінюється в залежності від правильності отриманих відповідей, наданих особою, яка проходить тест.

Використання адаптивних тестів має такі переваги:

1. Персоналізація. Тести можуть адаптуватися до індивідуальних здібностей учнів, уникати завдань, які для них є важкими чи простими.

2. Точність оцінки. Завдяки використанню більшого різноманіття питань різного рівня складності, можливо точніше оцінювати знання як сильних, так і слабких користувачів.

3. Зменшення обсягу тесту. Адаптивні тести можуть скоротити кількість питань, необхідних для отримання хорошої оцінки, що зекономить час користувачів.

4. Менша втомлюваність. Адаптивність допомагає зменшити втомлюваність користувачів, оскільки вони не витрачають час на питання, які їм можуть бути не під силу.

Адаптивне тестування може бути реалізоване різними способами з метою персоналізації процесу оцінювання. Один з можливих підходів – це встановлення складності для кожного запитання в тесті. Коли користувач правильно відповідає на питання, наступне запитання може бути більш складним, спрямовуючись на навички респондента і його знання на більш високому рівні. У випадку неправильної відповіді, система може запропонувати

менш важке запитання для підтримки користувача та підвищення його впевненості. Такий підхід допомагає адаптувати процес тестування до індивідуальних потреб кожного учня і забезпечити більш гнучке оцінювання знань [21].

Блок-схему узагальненого алгоритму тестування наведено на рисунку 2.5.

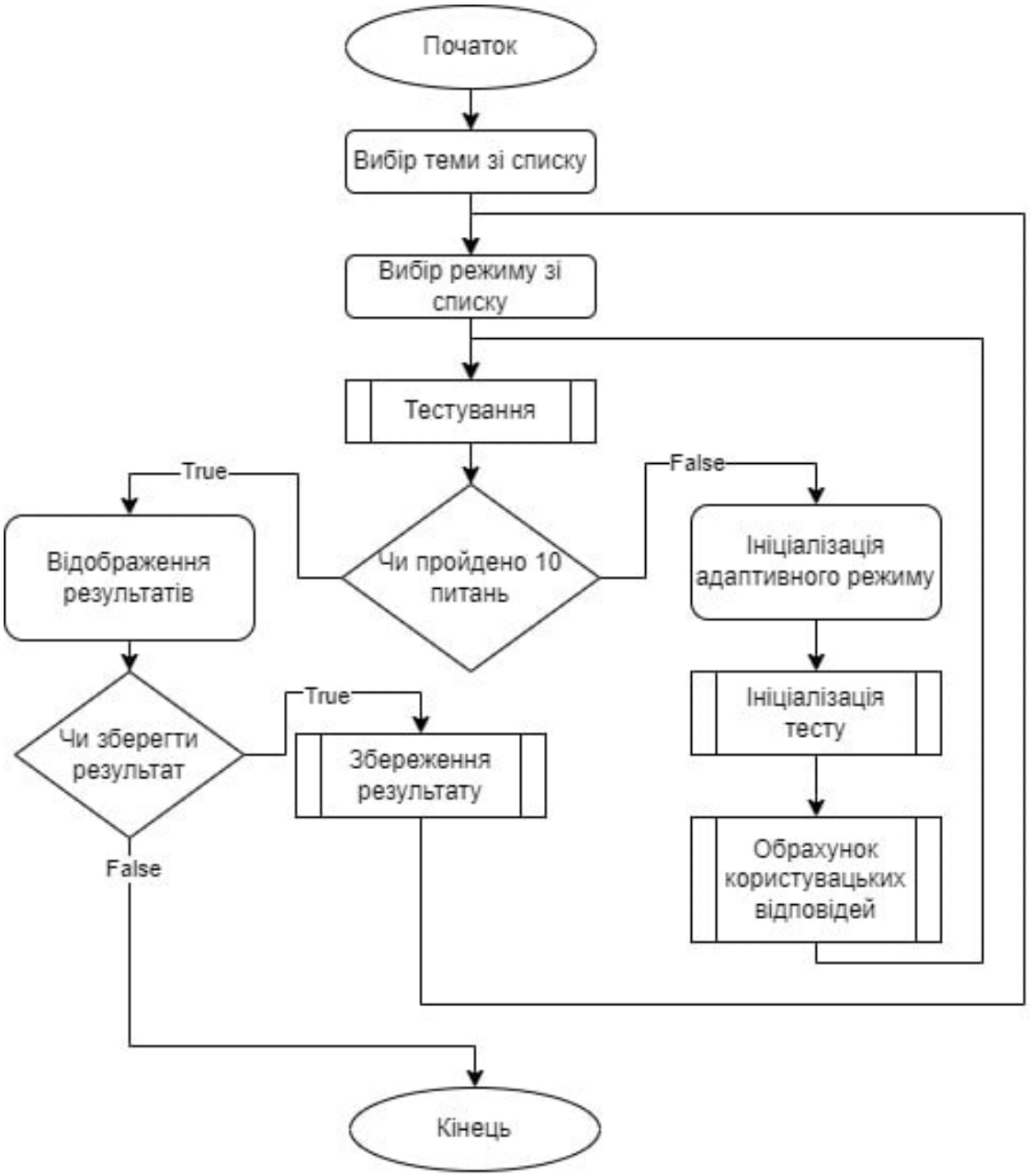

Рисунок 2.5 – Узагальнена блок-схема алгоритму тестування

Наступним етапом є модель нарахування балів та бонусів. Один із можливих варіантів нарахування балів – це використання адаптивної системи

оцінювання, де кількість балів за кожне завдання залежить від його складності. Наприклад, за правильну відповідь на питання вищої складності можна нараховувати більше балів, тоді як за правильну відповідь на менш складне питання – менше балів. Така система оцінювання дозволяє більш точно враховувати рівень знань користувача та надихає його на подальше вдосконалення. Застосування адаптивного методу навчання враховує індивідуальні особливості учня та створює більш ефективну систему оцінювання. Алгоритм нарахування балів зображено на блок-схемі (рис. 2.6).

Рисунок 2.6 – Блок-схема алгоритму нарахування балів

Іншим варіантом є нарахування бонусів у вигляді монет, що може бути використано для стимулювання користувачів до досягнення певних цілей,

заохочення активності, а також для проходження тестів, які потрібно купувати, або надає можливість використати підказку. Цей підхід може бути ефективним для мотивації користувачів до бажаної дії або навіть для створення змагань та рейтингів серед користувачів. Загалом, нарахування монет – це інструмент для створення стимулів та мотивації в різних сферах, який дозволяє визначити та винагородити досягнення. Алгоритм нарахування монет подано у вигляді блок-схеми (рис. 2.7).

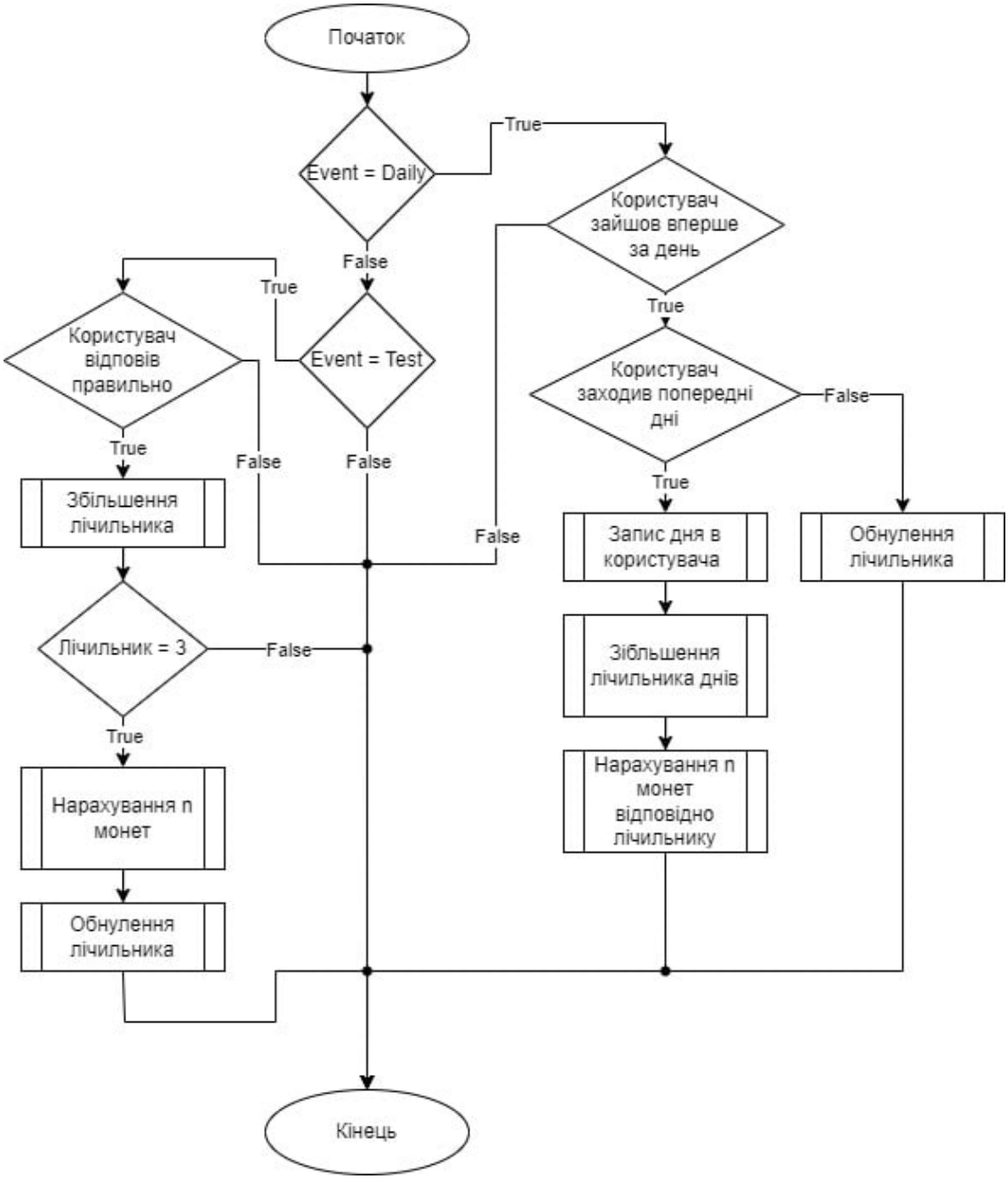

Рисунок 2.7 – Блок-схема алгоритму нарахування монет

Отже, в цьому підрозділі було розроблено та продемонстровано алгоритми тестування та алгоритми процесу нарахування балів та монет.

2.5 Розробка алгоритмів роботи додатку

У системі важливою рисою є наявність взаємозв'язків та відносин між її складовими. Для графічного подання цих взаємодій застосовується Unified Modeling Language (UML) – стандартна мова моделювання.

Діаграма прецедентів виступає як ефективний інструмент візуалізації взаємодії між акторами (користувачами) та сценаріями використання (прецедентами). Ця діаграма дозволяє отримати чітке уявлення про функціональність системи, ідентифікувати ролі, які беруть участь у взаємодії, та описати, як ці ролі взаємодіють між собою для досягнення конкретних цілей.

Іншими словами, діаграма прецедентів використовується для візуалізації та розуміння функціональних вимог до системи, що допомагає розробникам та зацікавленим сторонам чітко уявити, як система буде використовуватися та які функції вона повинна виконувати.

Модель навчальної гри з історії України, використовуючи адаптивний метод навчання, ілюструється на діаграмі прецедентів, зображеній на рис. 2.8. Ця діаграма дозволяє візуально презентувати взаємодію між різними прецедентами та акторами у системі, чітко визначаючи, як користувачі взаємодіють із навчальним додатком та як система реагує на їхні дії, забезпечуючи адаптивний процес навчання.

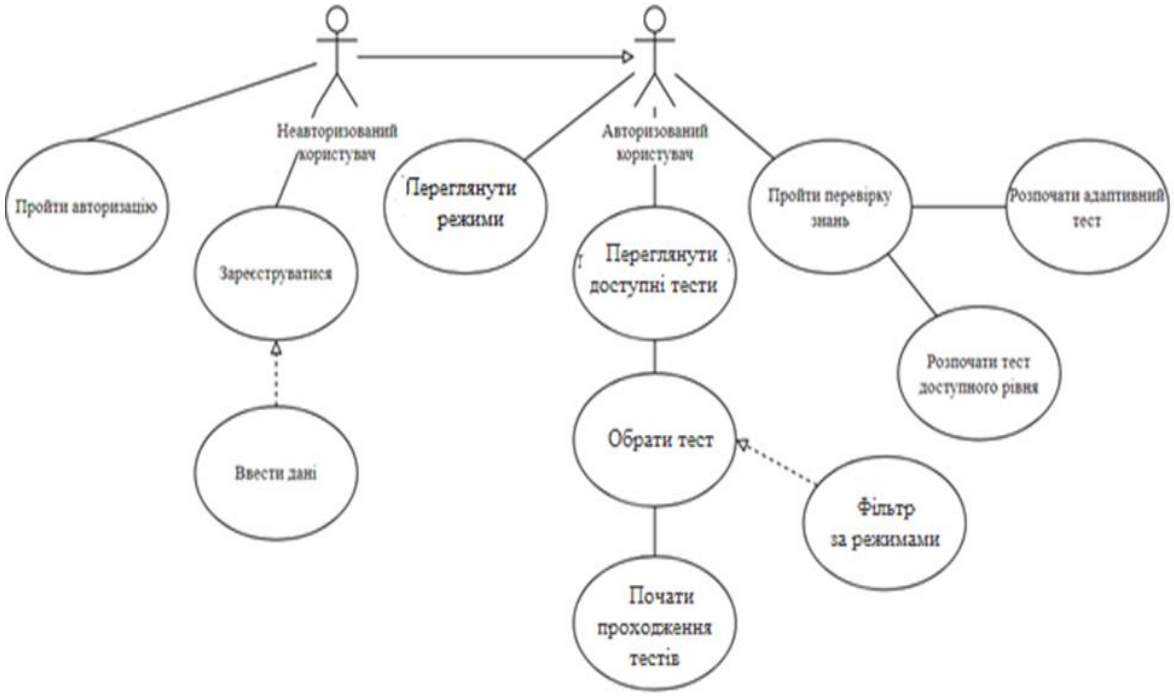

Рисунок 2.8– Модель навчальної гри у вигляді діаграми прецедентів

Перелік варіантів використання та їх короткий опис наведено в таблиці 2.1.

Таблиця 2.1 – Реєстр варіантів використання

Актор	Прецедент	Опис
Неавторизований користувач	Пройти авторизацію	Введення логіну та паролю для здійснення входу в навчальну гру.
	Зареєструватися	Введення та збереження особистих даних нового користувача для подальшої можливості авторизації.
Авторизований користувач	Переглянути особисту інформацію	Перегляд актуальної інформації про рівень, «статус» користувача та кількість завершених тестів.
	Переглянути режими	Переглянути основні режими проходження гри

Продовження таблиці 2.1

	Пройти перевірку знань	Можливість обрати та пройти один з доступних для користувача тестів.
	Переглянути доступні тести	Можливість перегляду доступних тестів для проходження з адаптивним методом навчання та без адаптації до знань користувача.

На першому етапі розробки навчальної гри "Історія України" з використанням адаптивного методу навчання, основною метою є оволодіння користувачем частковими теоретичними знаннями з історії України та їх подальша перевірка. Головною ідеєю є надання користувачеві можливості вчитися та перевіряти свої знання через виконання тестових завдань.

Після успішного завершення тесту користувач отримує доступ до інших тестів, охоплюючи різні теми з історії України. Кожен тест включає в себе різні завдання та питання, спрямовані на перевірку різних аспектів знань.

Кожна відповіді на питання повинні бути розглянуті крок за кроком. Такий підхід сприяє глибокому засвоєнню матеріалу та забезпечує ефективний процес навчання. Програмний продукт розробляється в різні етапи, і перший з них - алгоритмізація. Алгоритм визначає чітку послідовність конкретних дій, необхідних для розв'язання завдання, що допомагає систематизувати та послідовно виконувати процеси в грі [22].

Для того, щоб побачити деякі алгоритми вирішення задач, при написанні додатку з адаптивним методом навчання було створено блок-схеми.

Для початку користування додатком, користувач повинен увійти за допомогою свого логіну та паролю. Проте, якщо користувач ще не має облікового запису в додатку, йому необхідно зареєструватися. Реєстраційний процес включає в себе введення особистих даних гравця, що може включати в

себе електронну пошту та пароль, а після цього можна ввести ім'я та іншу інформацію. Ця інформація може використовуватися для ідентифікації та взаємодії користувача в додатку. Додаток може забезпечувати захист цих особистих даних та дотримуватися стандартів приватності, щоб забезпечити конфіденційність і безпеку інформації користувачів. Алгоритм реєстрації та авторизації наведено на рисунках 2.9 та 2.10.

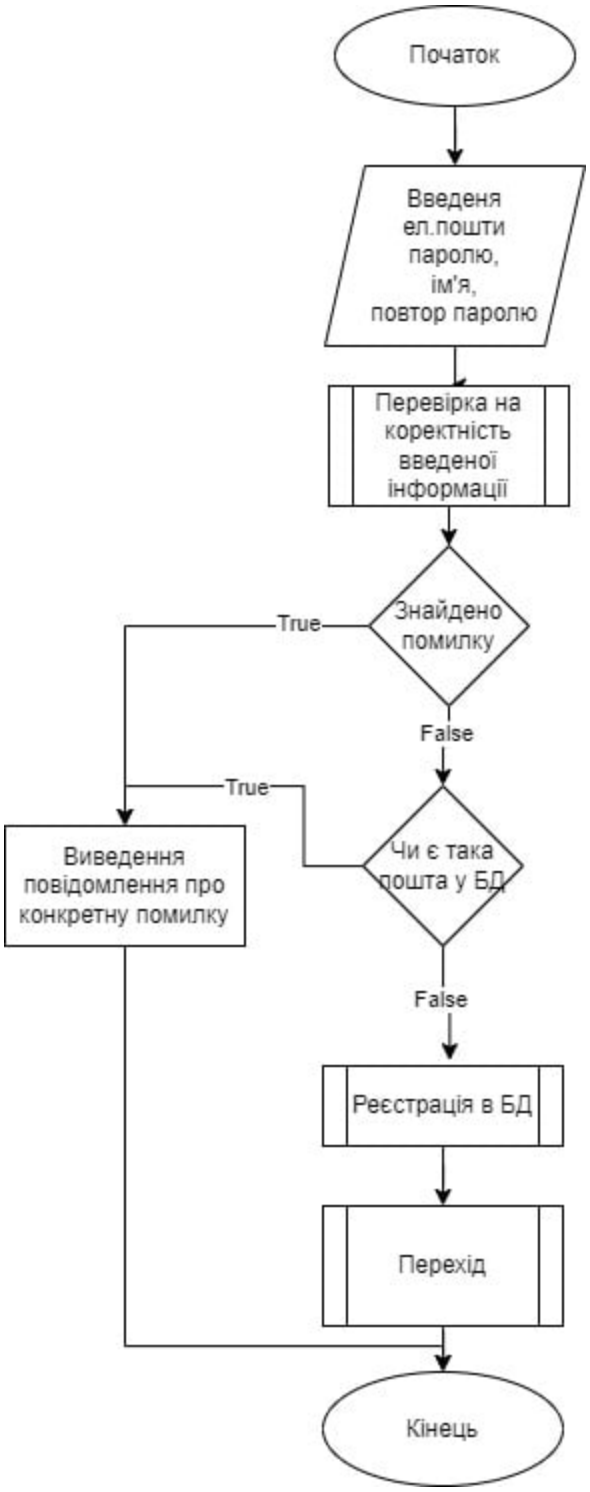

Рисунок 2.9 – Блок-схема алгоритму реєстрації

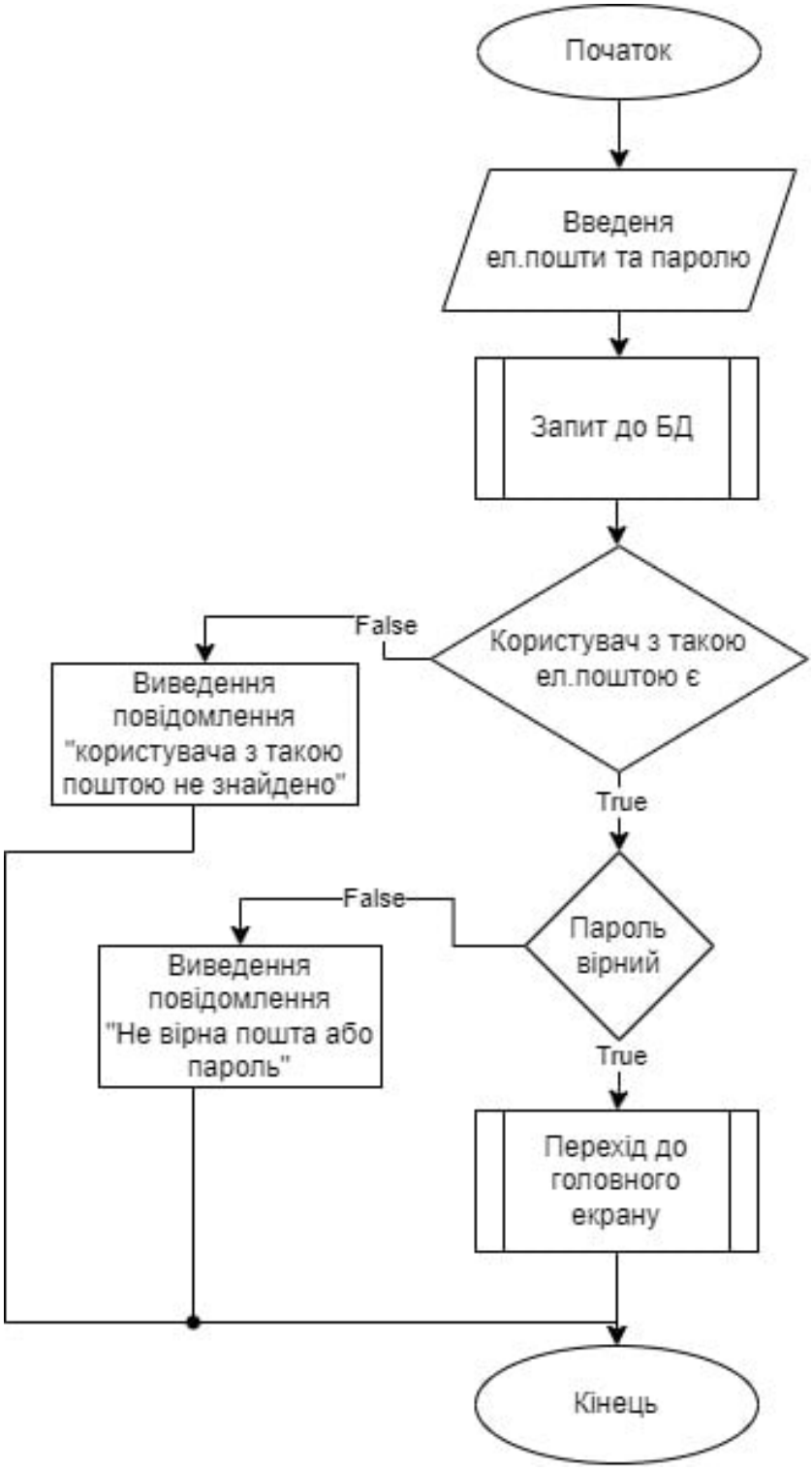

Рисунок 2.10 – Блок-схема алгоритму авторизації

Кожен крок в алгоритмі є доволі логічним, використовується база даних і йдуть перевірки за кожним пунктом: чи зареєстрований користувач, чи вірно він ввів пароль і т.д.

У додатку були розроблені алгоритми для проходження VIP-тесту. Цей тест передбачає можливість користувачів придбати його за допомогою монет, які вони отримують під час гри. VIP-тест містить більше запитань та цікавіший матеріал, що робить його більш вимогливим та захопливим для користувачів. Такий підхід дозволяє стимулювати користувачів до активності та надає можливість отримати більше знань та задоволення від гри.

На рисунку 2.11 зображено блок-схему алгоритму тестування.

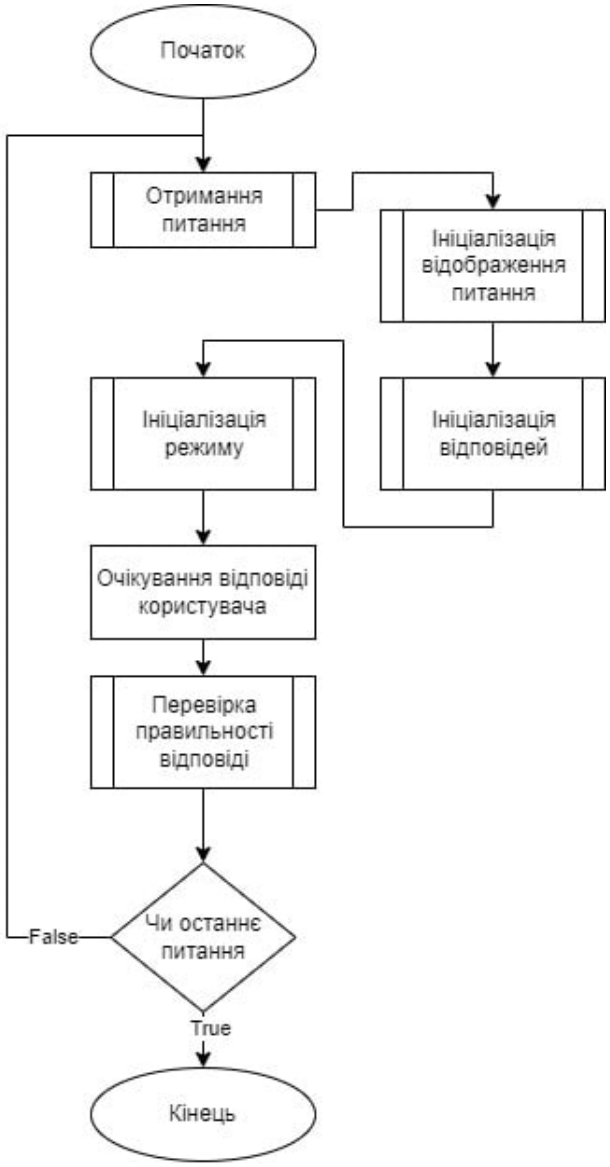

Рисунок 2.11 – Блок-схема алгоритму проходження тесту

2.6 Висновки

Отже, в другому розділі дослідження було розроблено метод

індивідуалізації навчального процесу для навчальної гри з використанням адаптивного методу навчання. Також були створені моделі системи, алгоритми роботи системи, алгоритми для авторизації та реєстрації користувачів у додатку. Розроблено алгоритми для проходження VIP-тесту та адаптивного тестування користувачів. Усі ці елементи допомагають створити індивідуалізований навчальний процес, який адаптується до потреб і здібностей кожного користувача, забезпечуючи ефективне та цікаве навчання.

3 РОЗРОБКА ПРОГРАМНИХ МОДУЛІВ НАВЧАЛЬНОЇ ГРИ З ІСТОРІЇ УКРАЇНИ ДЛЯ ANDROID -ДОДАТКУ З ВИКОРИСТАННЯМ МЕТОДУ АДАПТИВНОГО НАВЧАННЯ

3.1 Варіантний аналіз і обґрунтування вибору засобів для реалізації програмного засобу

Вибір мови програмування для реалізації будь-якої програми є складним і відповідальним рішенням. У випадку розробки мобільної гри вибір мови програмування залежатиме від різноманітних факторів. Важливими з них є рівень складності проєкту, доступність ресурсів, можливості мови програмування, а також вимоги до швидкості, надійності та масштабованості програми. Рішення повинно бути обґрунтованим і враховувати потреби та особливості самого проєкту. Наприклад, для розробки мобільної системи на Android-платформі можуть використовуватися мови програмування, такі як Java або Kotlin, у той час як для iOS-платформи використовуються мови, такі як Swift або Objective-C. Однак важливо також враховувати, що може знадобитися інтеграція з вже існуючими системами, що також вплине на вибір мови програмування [23].

Kotlin – це нова мова програмування, розроблена компанією JetBrains. Вона була випущена в 2011 році і швидко набрала популярності серед розробників. Kotlin призначений для створення додатків на платформі Java, але він також може використовуватися для інших цілей, таких як Android-розробка, веб-розробка та багато іншого [24].

Kotlin має низку переваг, які роблять його привабливим вибором для широкого кола завдань і проєктів. По-перше, Kotlin є дуже виразною мовою. Це означає, що код Kotlin часто може бути написаний більш компактно і зрозуміло, ніж код на інших мовах. По-друге, Kotlin має сильну систему безпеки типів. Це означає, що компілятор може виявити багато помилок ще до того, як програма буде запущена. По-третє, Kotlin є дуже зручною мовою в використанні. Вона має

простий синтаксис і вбудовані функції, які можуть допомогти розробникам швидко і ефективно створювати програми [25].

Але як і всі мови програмування Kotlin має і свої недоліки. Перехід на Kotlin може бути складним для деяких розробників, які звикли до Java. Це пов'язано з тим, що Kotlin має інший синтаксис і підходи до кодування. Крім того, Kotlin може не підтримуватися в деяких галузях розробки, наприклад, в системному програмуванні або розробці ігор.

Java – це одна з найпопулярніших мов програмування серед розробників в Україні та одна з найпоширеніших серед Back-end розробників у світі. Вона є відмінним вибором для початківців завдяки великій спільноті користувачів, високій популярності на ринку праці та багатому набору навчальних ресурсів.

Java – це загальнопризначена мова програмування, створена компанією Sun Microsystems у 1995 році. Вона знайшла застосування в розробці програмного забезпечення, веб-сервісів, ігор і різноманітних додатків. Java користується великою популярністю завдяки своєму простому синтаксису, гнучкості, високому рівню безпеки, портативності (можливості працювати на різних платформах) та можливості масштабування для великих проектів [26].

Java – мова програмування, яка має велику кількість переваги [27]:

1. Простий і зрозумілий синтаксис, що полегшує вивчення та використання. Це особливо корисно для новачків у програмуванні.

2. Портативність: Java дозволяє запускати код на різних операційних системах, якщо є підтримка віртуальної машини Java (JVM). Це спрощує розробку та розгортання додатків.

3. Велика стандартна бібліотека, яка включає безліч готових класів і методів для вирішення різних завдань, що сприяє швидкості та зручності програмування.

4. Висока кібербезпека завдяки механізму перевірки типів та відсутності покажчиків на пам'ять, що робить її привабливим вибором для розроблення застосунків, що потребують високого рівня захисту даних.

5. Підтримка багатопотоковості і масштабованість, що дозволяє опрацьовувати великі обсяги даних та виконувати багато завдань одночасно.

6. Висока продуктивність завдяки віртуальній машині та оптимізації коду.

7. Велика спільнота розробників, яка надає підтримку та доступ до різноманітних матеріалів для вивчення та розвитку.

8. Ефективна обробка винятків та помилок, що сприяє створенню надійного та стабільного програмного забезпечення.

9. Використовується в різних сферах, включаючи розробку мобільних застосунків, веб-сервісів, ігор, фінансових систем і систем для наукових досліджень.

10. Підтримка рефлексії, що дозволяє програмам аналізувати та модифікувати свою структуру під час виконання, що полегшує адаптацію до змінних вимог.

Android Studio – інтегроване середовище розробки від Google для розробки додатків на платформі Android OS. Це середовище може бути встановлене на різних операційних системах, включаючи Windows, Mac і Linux. Android Studio базується на платформі IntelliJ IDEA і надає розробникам засоби для створення різних типів додатків для Android, включаючи додатки для смартфонів, планшетів та інших контекстуальних модулів [28].

Android Studio призначене для розробників різного рівня досвіду – від початківців до професіоналів. Розробники можуть використовувати Java або C++ для створення Android-додатків.

Android Studio підтримує безперервну інтеграцію, що дозволяє виявляти проблеми в реальному часі. Це сприяє ранньому виявленню і виправленню помилок та прискорює процес розгортання додатків в Google Play App Store.

Android Studio має інструменти для оцінки продуктивності, включаючи аналіз файлів з пакетом програми, візуалізацію графіки, вимірювання використання пам'яті та аналіз батареї пристрою. Такі інструменти допомагають розробникам оптимізувати продуктивність своїх додатків.

Android Studio також підтримує інтеграцію з Google App Engine та іншими сервісами Google, а також підтримує всі версії платформи Android, починаючи з Android 1.6.

Серед основних функцій Android Studio можна виділити [29]:

- розширений редактор макетів з можливістю роботи з UI компонентами за допомогою Drag-and-Drop;
- використання Gradle для збірки додатків;
- підтримка різних видів збірок та генерація кількох .apk файлів;
- рефакторинг коду та статичний аналізатор коду (Lint);
- вбудований ProGuard для оптимізації та захисту коду;
- підтримка розробки додатків для Android Wear, Android TV та інших платформ;
- інтеграція з Google Cloud Platform та різними API;
- підтримка мови Kotlin в нових версіях Android Studio.

Android Studio – це потужний інструмент для розробки Android-додатків з багатьма корисними функціями та інструментами для підтримки різних типів проектів.

Отже, для виконання магістерської кваліфікаційної роботи було обрано мову програмування Java та інтегроване середовище розробки Android Studio. Використання мови програмування Java дозволяє користуватися численними перевагами, такими як підтримка функціонального програмування, компактний синтаксис та інші.

Android Studio є потужним інструментом для розробки мобільних додатків для платформи Android, що надає широкі можливості для створення, тестування та налагодження програмного коду. Використання обох цих інструментів дозволить здійснити якісну розробку додатку.

3.2 Розробка програмного модуля для реєстрації та авторизації користувачів

Використання платформи Firebase для реалізації авторизації та реєстрації може спростити розробку та забезпечити надійний та безпечний механізм управління користувачами. Firebase надає зручний інтерфейс для автентифікації, а також забезпечує можливості зберігання даних та інші сервіси в хмарі.

Код починається з визначення класу DataNetHandler, який містить методи для спілкування з інтернетом. Цей клас не є обов'язковим, але він може бути корисним для розробки веб-додатків, які потребують доступу до Інтернету.

Далі код визначає метод login(), який є основною функцією авторизації користувачів. Цей метод приймає два параметри: електронну пошту та пароль.

Метод login() спочатку перевіряє, чи є така електронна пошта в базі даних. Якщо такої електронної пошти немає, він видає повідомлення, що такого користувача з поштою немає.

Якщо користувач вже зареєстрований, код авторизує його за допомогою методу signIn().

Якщо користувач вже зареєстрований, код відправляє запит до БД для отримання користувача відповідно до пошти. Результатом методу є об'єкт користувача або null якщо авторизація невдала.

На рисунку 3.1 зображено класу DataNetHandler.

```
public class DataNetHandler {

    // >>>>> AUTH PART
    public void login(String email, String password, ICallback<User> callback) {
        Task<AuthResult> resultTask = FirebaseAuth.getInstance().signInWithEmailAndPassword(email, password);
        resultTask.addOnSuccessListener(authResult -> {
            FirebaseDatabase.getInstance().getReference().get().addOnSuccessListener(dataSnapshot -> {
                if (dataSnapshot.exists()) {
                    String uuid = authResult.getUser().getUid();
                    DataSnapshot snapshot = dataSnapshot.child( path: "Users").child(uuid);
                    if (snapshot.exists()) {
                        call(callback, snapshot.getValue(User.class));
                    }
                }
            });
        });
        resultTask.addOnFailureListener(command -> call(callback,  value: null));
    }
```

Рисунок 3.1 – Створення класу DataNetHandler

На наступній частині коду зображено два методи логіну та реєстрації.

Метод login() спочатку використовує Firebase Authentication для входу в систему користувача за його електронною адресою та паролем. Якщо операція входу в систему успішна, метод отримує UID користувача та використовує його для отримання даних користувача з бази даних. Якщо дані користувача існують, метод викликає метод call() інтерфейсу callback з даними користувача.

У логіні викликаємо метод signInWithEmailAndPassword з параметрами пошти та паролю, далі отримуємо callback, який повертає об’єкт, якщо є такий у БД, після чого ми отримуємо об’єкт User який містить всю інформацію

У реєстрації також досить проста реалізація – виклик методу createUserWithEmailAndPassword. Цей метод приймає чотири параметри: ім'я користувача, email, пароль та callback. Спочатку він використовує екземпляр FirebaseAuth для створення нового користувача з заданим email та паролем. Якщо користувач створений успішно, метод створює новий об'єкт User і встановлює його властивості. Метод потім викликає метод updateUser() для збереження даних користувача в Firebase Realtime Database. Нарешті, метод викликає callback зі значенням true.

Цей метод використовується для реєстрації нового користувача. Він приймає чотири параметри: ім'я користувача, адресу електронної пошти, пароль і іконку (рис.3.2).

```
// >>>>> AUTH PART
public void login(String email, String password, ICallback<User> callback) {
    Task<AuthResult> resultTask = FirebaseAuth.getInstance().signInWithEmailAndPassword(email, password);
    resultTask.addOnSuccessListener(authResult -> {
        FirebaseDatabase.getInstance().getReference().get().addOnSuccessListener(dataSnapshot -> {
            if (dataSnapshot.exists()) {
                String uuid = authResult.getUser().getUid();
                DataSnapshot snapshot = dataSnapshot.child("Users").child(uuid);
                if (snapshot.exists()) {
                    call(callback, snapshot.getValue(User.class));
```

Рисунок 3.2 – Методи логіну та реєстрації

Наступний метод використовується для оновлення даних користувача в Firebase Realtime Database. Він приймає один параметр: об'єкт User із даними, які потрібно оновити.

Якщо об'єкт User не є null, метод отримує екземпляр DatabaseReference для кореня Firebase Realtime Database. Потім метод встановлює значення в зазначеному місці в базі даних на значення об'єкта User (рис3.3).

```
public void updateUser(@Nullable User user) {
    if (user != null) {
        DatabaseReference database = FirebaseDatabase.getInstance().getReference().child( pathString: "Users");
        database.child(user.getUuid()).setValue(user);
    }
}
```

Рисунок 3.3 – Метод для оновлення даних користувача в Firebase Realtime Database

Схема обробника кліків екрану авторизації – це алгоритм, який визначає, що робити, коли користувач натискає кнопку або інший елемент керування на екрані авторизації. Першим кроком є обробка події кліка. Цей крок виконується в обробнику подій елемента екрану, який був натиснутий користувачем. Обробник подій повинен визначити, який елемент екрану був натиснутий, і викликати відповідний метод обробки кліка. Другим кроком є перевірка доступу. Цей крок виконується методом обробки кліка, який визначає, чи має користувач доступ до елемента екрану, який був натиснутий. Якщо користувач не має доступу, метод обробки кліка повинен відхилити запит. Якщо користувач має доступ, метод обробки кліка повинен обробити запит користувача. Цей крок може включати в себе такі дії, як: перевірка введених даних, здійснення запиту до бази даних, виконання операції на пристрої. Після обробки запиту метод обробки кліка повинен відобразити результат користувачеві. Блок-схема алгоритму обробника кліків екрану авторизації зображена на рисунку 3.4.

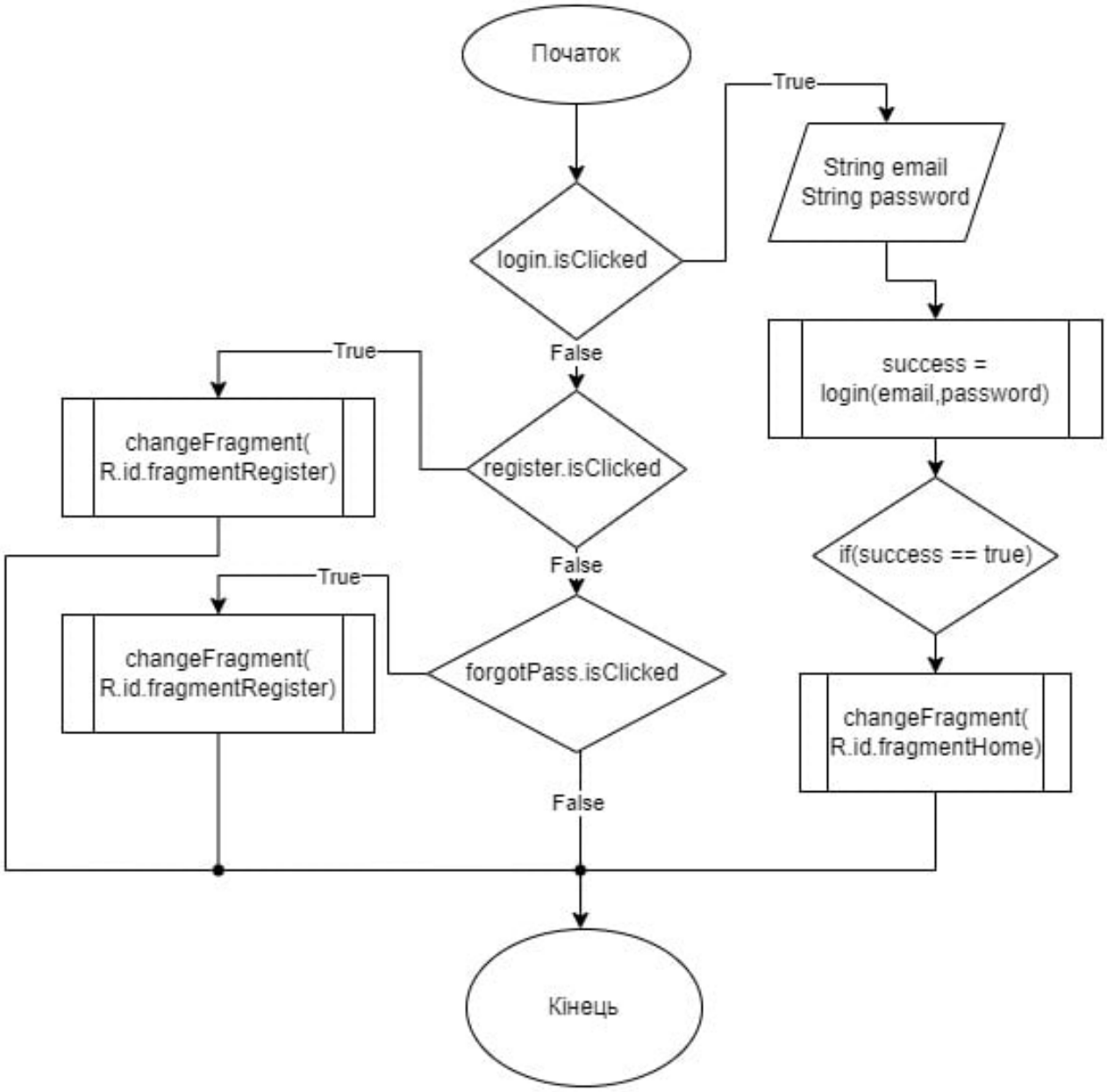

Рисунок 3.4 –Блок-схема алгоритму обробника кліків екрану авторизації

3.3 Розробка програмного модуля індивідуалізації з адаптивним тестуванням

Адаптивний режим реалізовано за технологією адаптивного підбору питань в процесі тестування. Кожній темі призначені питання, і кожне питання має рівень складності від 1 до 5 або більше. Після правильних відповідей лічильник перевіряє, чи на певну кількість питань поспіль отримано правильну відповідь, чи неправильну. Якщо лічильник перевищує 3 правильні відповіді

підряд, то вибираються складніші питання. У випадку неправильних відповідей лічильник скидається, і підбираються легші питання.

Методи реалізовані в класі FragmentTest (onCorrect і onFail), які викликаються після кнопки підтвердити.

Метод onFail виконує такі дії як збільшення кількості неправильних відповідей користувача на одиницю. Це робиться за допомогою методу setTotalWrong() об'єкта result, також збільшує кількість поспіль неправильних відповідей користувача на одиницю. Це робиться за допомогою методу setStreakWrong() об'єкта result (рис 3.5).

Підсвічує запитання червоним кольором. Це робиться за допомогою методу blinkColor(). Якщо кількість отриманих поспіль неправильних відповідей користувача досягає 3, то, якщо режим навчання не активний, то зменшує рівень адаптації користувача на одиницю. Це робиться за допомогою методу setAdaptiveLevel() об'єкта result також виводить повідомлення про те, що рівень адаптації зменшився. Це реалізується за допомогою методу System.out.println().

```
private void onFail() {
    result.setTotalWrong(result.getTotalWrong() + 1);
    result.setStreakWrong(result.getStreakWrong() + 1);
    user.getStatistic().setTotalWrong(user.getStatistic().getTotalWrong() + 1);

    blinkColor(binding.question, R.color.red);

    if (result.getStreakWrong() >= 3) {
        if (!localSettings.isLearningMode()) {
            result.setAdaptiveLevel(Math.max(result.getAdaptiveLevel() - 1, 0));
            System.out.println("Adaptive level decreased");
        }
    }
```

Рисунок 3.5 – Створення методу onFail

Метод onCorrect() виконує такі ді,ї як підсвічує запитання зеленим кольором. Це реалізується за допомогою методу blinkColor().

Збільшує кількість правильних відповідей користувача на одиницю. Це робиться за допомогою методу setTotalCorrect() об'єкта result, також збільшує кількість поспіль правильних відповідей користувача на одиницю. Це робиться за допомогою методу setStreakCorrect() об'єкта result.

Якщо кількість поспіль правильних відповідей користувача досягає 3, то режим навчання не активний, та збільшує рівень адаптації користувача на одиницю. Це реалізується за допомогою методу setAdaptiveLevel() об'єкта result. Додає користувачеві гроші в розмірі settings.getMultiplierMoneys(). Це робиться за допомогою методу addMoney() класу UserUtils.

Приклад коду методу onCorrect() наведено на рисунку 3.6.

```
private void onCorrect() {
    blinkColor(binding.question, R.color.green);
    result.setTotalCorrect(result.getTotalCorrect() + 1);
    result.setStreakCorrect(result.getStreakCorrect() + 1);
    user.getStatistic().setTotalCorrect(user.getStatistic().getTotalCorrect() + 1);
    if (result.getStreakCorrect() >= 3) {
        result.setStreakCorrect(0);
        if (!localSettings.isLearningMode()) {
            result.setAdaptiveLevel(Math.min(result.getAdaptiveLevel() + 1, 5));
            System.out.println("Adaptive level increased");
        }
        int addMoney = (int) (1 * settings.getMultiplierMoneys());
        UserUtils.addMoney(user, addMoney);
    }

    result.setStreakWrong(Math.max(0, result.getStreakWrong() - (int) (1 + Math.random() * 1)));

    mainViewModel.getUtils().waitAsync( msTime: 1000, this::getNewTest);
}
```

Рисунок 3.6 – Створення методу onCorrect()

Блок-схема алгоритму поведінки адаптивного режиму під час тестування наведена на рисунку 3.7.

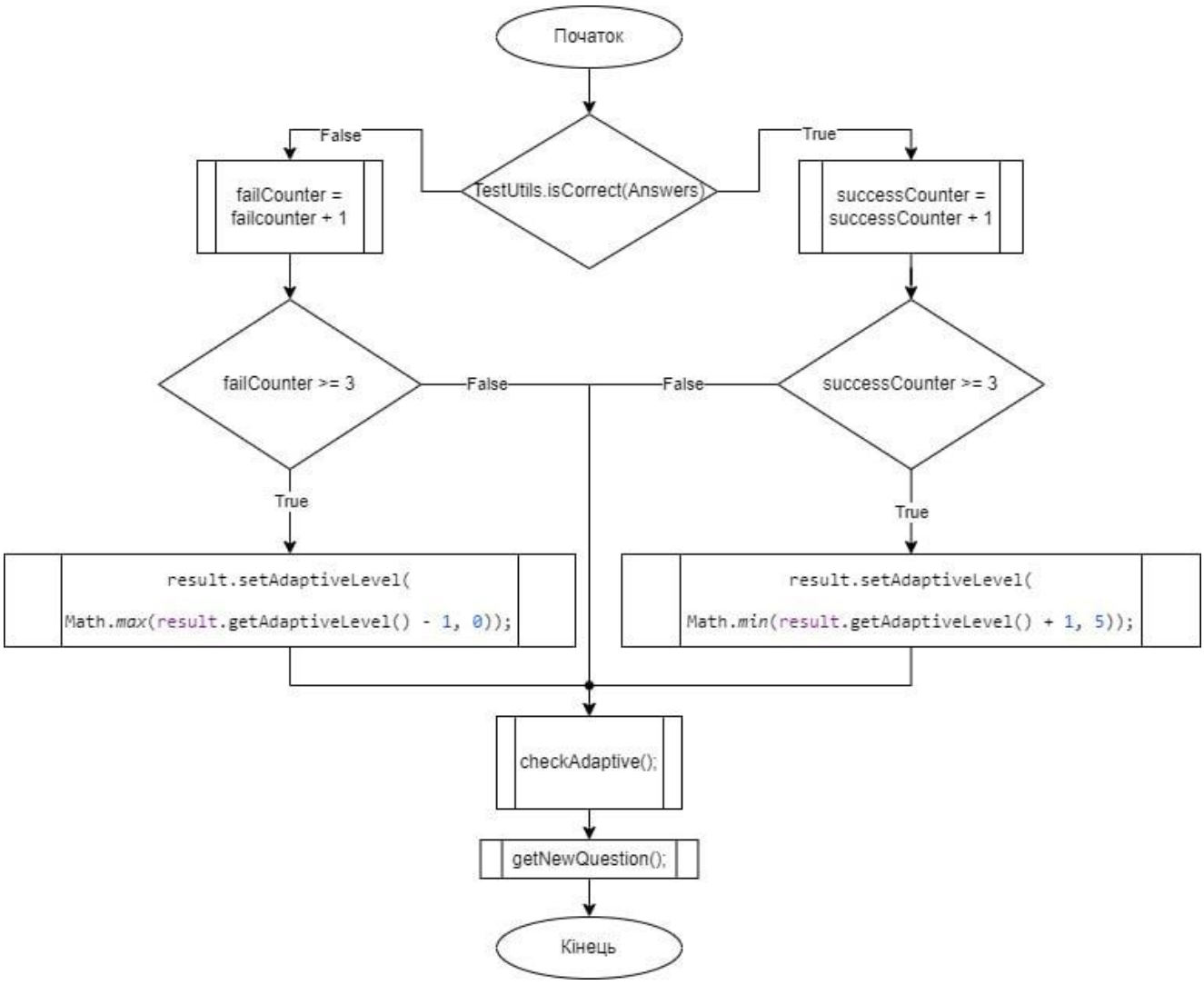

Рисунок 3.7 – Блок-схема алгоритму поведінки адаптивного режиму під час тестування

3.4 Розробка модуля нарахування балів та отримання бонусів для навчальної гри

На етапі розробки не менш важливим етапом є розробка моделі нарахування балів та отримання монет для навчальної гри.

Цей код є методом з назвою onCalculate(). Він викликається, коли користувач відповідає на питання. Метод виконує такі дії:

1. Створює об'єкт UserTest для зберігання інформації про відповідь користувача.

2. Встановлює значення correct в об'єкті UserTest відповідно до того, чи була відповідь користувача правильною.

3. Встановлює значення points в об'єкті UserTest відповідно до рівня тесту та множника грошей, встановленого в налаштуваннях.

4. Додає об'єкт UserTest до списку тестів користувача.

Отримується список нагород та для користувача ініціалізується візуальний список з нагородами.

У даному випадку, метод createUserTest() повертає об'єкт UserTest з такими значеннями:

1. correct: isCorrect – значення, яке вказує, чи була відповідь користувача правильною.

2. points: (int) (test.getLevel() == 0 ? 1 * settings.getMultiplierPoints() : test.getLevel() * settings.getMultiplierPoints()) – кількість балів, отриманих користувачем за відповідь. Рівень тесту визначається методом getLevel(), а множник грошей – методом getMultiplierPoints().

Фрагмент коду подано на рисунку 3.8.

```
// core || calculate -> fail/correct
private void onCalculate(boolean isCorrect) {
    UserTest resultAnswer = UserUtils.createUserTest(test);
    resultAnswer.setCorrect(isCorrect);
    resultAnswer.setPoints(isCorrect ? (int) (test.getLevel() == 0 ? 1 * settings.getMultiplierPoints() : test.getLevel() * settings.getMultiplierPoints()) : 0);
    result.getTests().add(resultAnswer);
}
```

Рисунок 3.8 – Створення методу onCalculate()

Наступний метод, який отримує налаштування глобальної системи, а потім використовує їх для відображення списку нагород для користувача (рис.3.9).

Метод починається з ініціалізації змінної rewards значенням settings.getDailyRewards(). Ця змінна містить список нагород, які користувач може отримати щодня. Потім метод перевіряє, чи не дорівнює rewards null. Якщо ні, то метод отримує доступ до користувача за допомогою змінної user. Ця змінна містить інформацію про користувача, наприклад, його ім'я, адресу електронної пошти та кількість днів, які він користується програмою. Якщо user також не дорівнює null, то метод створює новий адаптер RewardAdapter. Цей адаптер використовується для відображення списку нагород на екрані.

Далі метод встановлює адаптер adapter для візуального елемента binding.itemsRV. Цей візуальний елемент являє собою список, який буде відображати нагороди.

Нарешті, метод викликає метод setItems() адаптера adapter, щоб встановити список нагород, який він повинен відображати. Якщо користувачу доступна нагорода, то метод викликає метод waitAsync() класу Utils. Цей метод виконує блок коду асинхронно, з затримкою в 1000 мілісекунд.

У блоці коду, який виконується асинхронно, метод отримує кількість днів, які користувач користується програмою. Потім він перевіряє, чи більше ця кількість, ніж кількість нагород, доступних користувачу. Якщо так, то метод встановлює кількість днів на значення, яке дорівнює кількості нагород, доступних користувачу.

```
mainViewModel.getGlobalSettings().observe(getViewLifecycleOwner(), settings -> {
    ArrayList<Reward> rewards = settings.getDailyRewards();

    if (rewards != null) {
        User user = mainViewModel.getUser().getValue();
        if (user != null) {
            RewardAdapter adapter = new RewardAdapter(user.getDays(), settings.getMultiplierMoneys());
            binding.itemsRV.setAdapter(adapter);
            binding.itemsRV.setLayoutManager(new LinearLayoutManager(requireContext()));
            adapter.setItems(rewards);
            if (canGive) {
                mainViewModel.getUtils().waitAsync( msTime: 1000, () -> {
                    int days = user.getDays();
                    if (days >= settings.getDailyRewards().size()) {
                        days = settings.getDailyRewards().size() - 1;
                    }
                    int count = (int) (settings.getDailyRewards().get(days).getAmount() * settings.getMultiplierMoneys());
                    UserUtils.addMoney(user, count);

                    mainViewModel.getDataNetHandler().updateUser(user);

                    if (isVisible()) {
                        Snackbar.make(view, String.format("%s %s %s", getString(R.string.you_get_tip), count, StringHelper.getTipsWord(requireContext(),
                    }
                });
            }
        }
    }
```

Рисунок 3.9 – Створення методу для відображення списку нагород для користувача

Наступним кроком метод обчислює значення нагороди, яку отримає користувач. Це значення розраховується як добуток кількості днів, які користувач користується програмою, і коефіцієнта множення, який визначається налаштуваннями глобальної системи.

Після цього метод викликає метод addMoney() класу UserUtils. Цей метод додає нагороду до балансу користувача. Нарешті, метод викликає метод updateUser() класу DataNetHandler. Цей метод оновлює інформацію про користувача в базі даних.

Якщо метод isVisible() повертає true, то метод викликає метод Snackbar.make(). Цей метод відображає сповіщення на екрані користувача. Сповіщення містить інформацію про те, що користувач отримав нагороду.

Також на рисунку 3.10 зображено блок-схему діалогу нарахування монет.

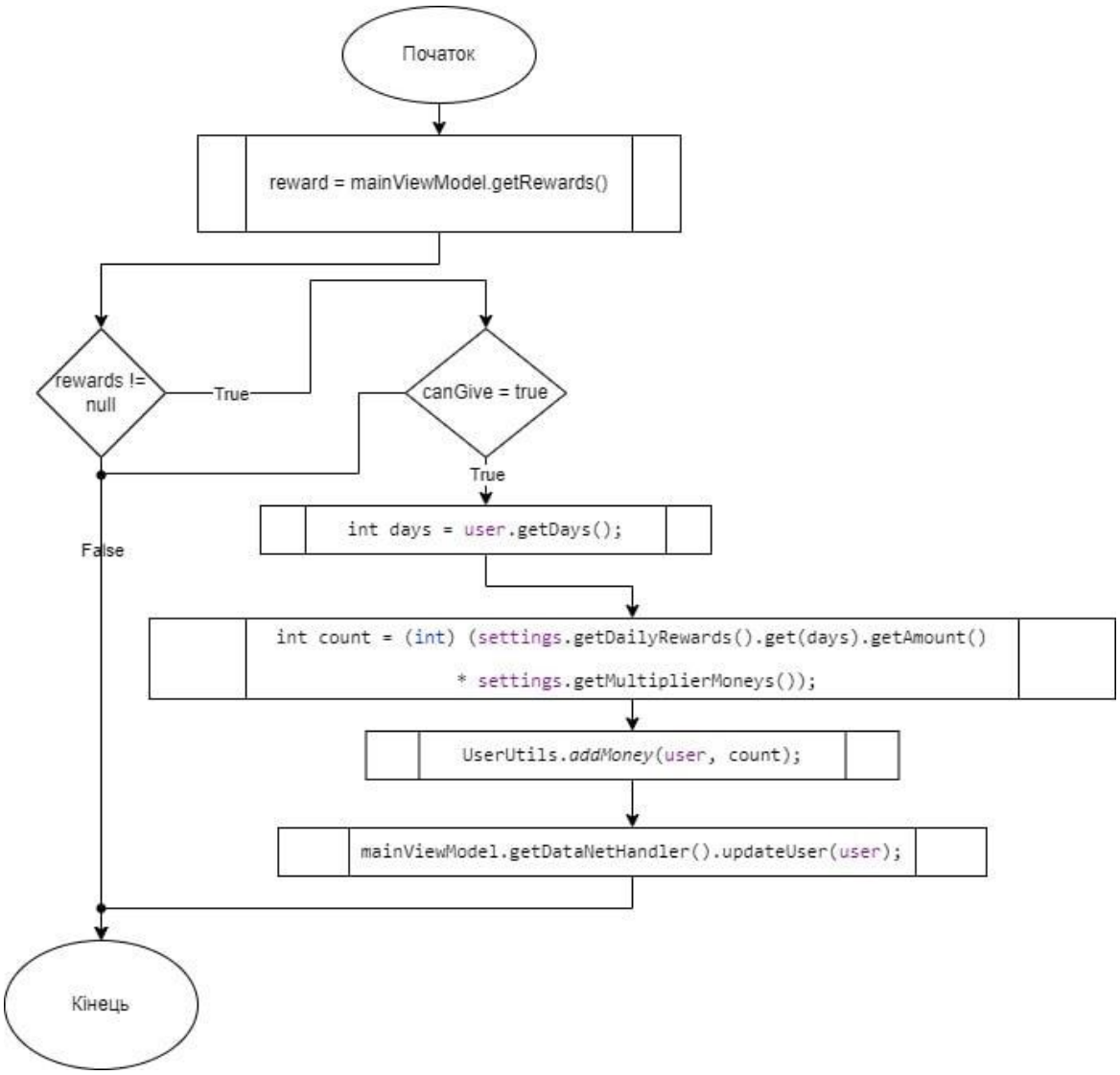

Рисунок 3.10 – Блок-схема діалогу нарахування монет

3.5 Висновки

У третьому розділі обґрунтовано вибір мови програмування та технологій для розробки програмного продукту. Основними елементами цього вибору стали мова програмування Java і використання інтегрованого середовища розробки Android Studio, що є оптимальними для створення мобільних додатків для платформи Android.

Крім того, визначено систему управління базами даних, і вибір припав на Firebase, забезпечуючи надійне та ефективне зберігання даних. Ці рішення були зроблені на підставі переваг проаналізованого інструменту та відповідно до вимог розробленого програмного продукту.

Розроблено навчальну гру з історії України, яка включає модуль для реєстрації та авторизації користувачів, метод індивідуалізації з адаптивним тестуванням та модель нарахування балів та отримання монет для навчальної гри.

4 ТЕСТУВАННЯ ПРОГРАМИ

4.1 Аналіз методів тестування програмного забезпечення

Тестування програмного забезпечення (ТПЗ) – це важливий процес, який включає в себе перевірку та оцінку якості програми з метою виявлення помилок, дефектів і проблем. Метою тестування є забезпечення того, що програмне забезпечення працює правильно, відповідає вимогам та очікуванням користувачів, а також забезпечує надійність, безпеку та ефективність його роботи. Тестування є важливою частиною процесу розробки програмного забезпечення і допомагає виявити та виправити проблеми ще до випуску програми.

Процес тестування програмного забезпечення включає кілька основних етапів. Спочатку проводять юніт-тестування, де перевіряють окремі компоненти програми. Потім виконується інтеграційне тестування для перевірки взаємодії компонентів. Наступний етап – системне тестування, коли перевіряється робота всієї системи з використанням різних сценаріїв. На завершальному етапі проводиться приймальне тестування для перевірки відповідності програми вимогам та готовності до впровадження. Ці етапи допомагають забезпечити якість та надійність програмного забезпечення перед його випуском [30]. Для досягнення цих цілей розробники використовують різні методи тестування:

1. Тестування продуктивності (Performance Testing) є важливою складовою нефункціонального тестування, спрямованого на оцінку реакції системи під впливом очікуваного або підвищеного навантаження при різних сценаріях використання. Метою цього виду тестування є перевірка швидкодії, працездатності, та стабільності системи. Параметри оцінки продуктивності включають час відгуку, надійність, використання ресурсів та масштабованість. Тестування продуктивності допомагає виявити можливі вразливості та недоліки у програмному забезпеченні з метою запобігання їх впливу на роботу системи під час реального використання.

2. Функціональне тестування (Functional Testing) займається аналізом функціональних характеристик програмного забезпечення та перевіркою відповідності реальної роботи реалізованих функцій очікуваній поведінці згідно зі специфікацією та бізнес-вимогами. Це тестування спрямоване на імітацію фактичного використання системи і може проводитися на різних рівнях тестування, включаючи компонентний, інтеграційний, системний та приймальний. Функціональні тести базуються на функціях, описаних у вимогах, функціональних специфікаціях або сценаріях використання системи (Use case).

3. Тестування безпеки (Security Testing) – це процес, спрямований на оцінку і перевірку системи з точки зору її стійкості до зовнішніх атак та забезпечення цілісності, доступності, конфіденційності та надійності програмного забезпечення. В рамках тестування безпеки проводяться перевірки автентифікації, авторизації, доступності, конфіденційності та безвідмовності програми під впливом різноманітних вторгнень і атак.

4. Тестування білої скриньки (White Box Testing) – це метод, в якому тестувальники аналізують внутрішню структуру програмного забезпечення та розуміють його внутрішній механізм роботи. Вони перевіряють вірність виконання функцій, логіку програми та відповідність вимогам.

5. Тестування чорної скриньки (Black Box Testing) – це метод, при якому тестувальники не розглядають внутрішню структуру програми і не мають доступу до її внутрішнього коду. Вони аналізують програму як незнайому "чорну скриньку", концентруючись лише на вхідних даних, обробці та вихідних результатів, з метою переконатися, що вона працює відповідно до специфікацій та заданих вимог.

Після ретельного аналізу різних методів тестування програмного забезпечення, було обрано метод тестування чорної скриньки. Цей вибір обумовлений переконанням, що тестування чорної скриньки надає більше об'єктивних результатів і дозволяє виявити потенційні проблеми, які можуть виникнути при фактичному використанні програмного забезпечення в реальних умовах.

4.2 Тестування програмного продукту

Для того, щоб розпочати тестування навчальної гри, було обрано метод «чорної скриньки», оскільки він не розглядають внутрішню структуру програми і не має доступу до її внутрішнього коду.

Для того, щоб розпочати роботу в додатку, потрібно пройти реєстрацію в системі, щоб дані користувача збереглися в базі даних та після того він міг без проблем заходити в додаток. Запустивши додаток, перед користувачем з’являється вікно реєстрації, в якому він може ввести свої дані та обрати аватар (рис. 4.1).

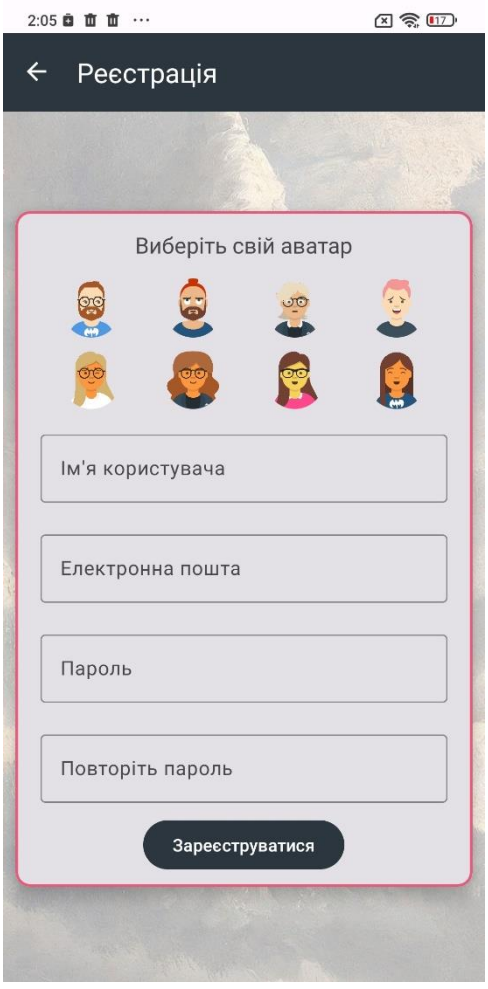

Рисунок 4.1 – Вікно реєстрації

Користувач, який вже є зареєстрований, після наступного входу в додаток просто вводить свої ідентифікуючі дані.

Вікно авторизації в програмі зображено на рисунку 4.2.

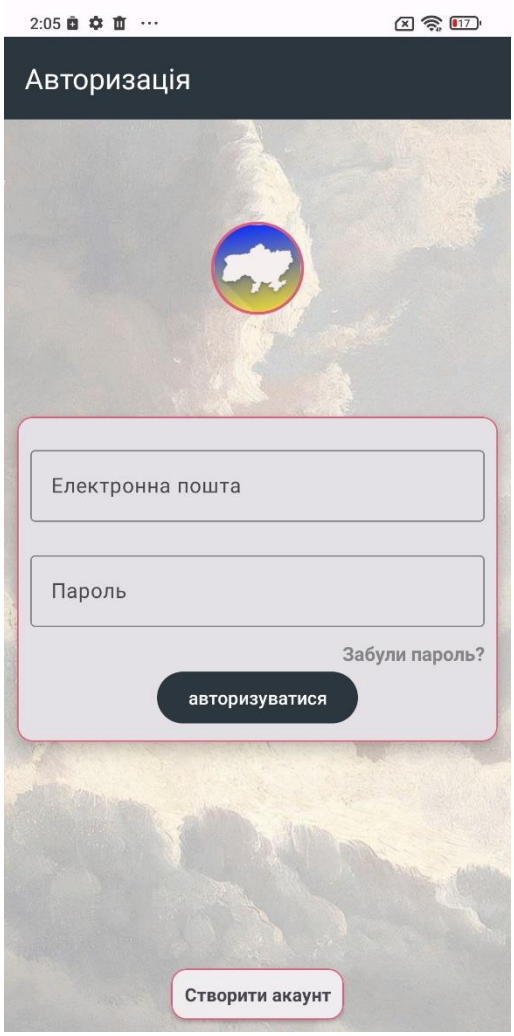

Рисунок 4.2 – Вікно авторизації

Якщо користувач введе невірні дані під час авторизації, система надсилатиме повідомлення про помилку, повідомляючи його про невдачу при вході. У випадку повторних невдалих спроб користувача спонукають використовувати опцію "Забули пароль?". Натискання на цю кнопку веде до процедури відновлення паролю через вказану електронну пошту користувача. Цей механізм забезпечує простий і безпечний спосіб відновлення доступу до облікового запису у випадку забутого пароля.

На рисунку 4.3 зображено вікно із сповіщенням та іконкою відновлення паролю.

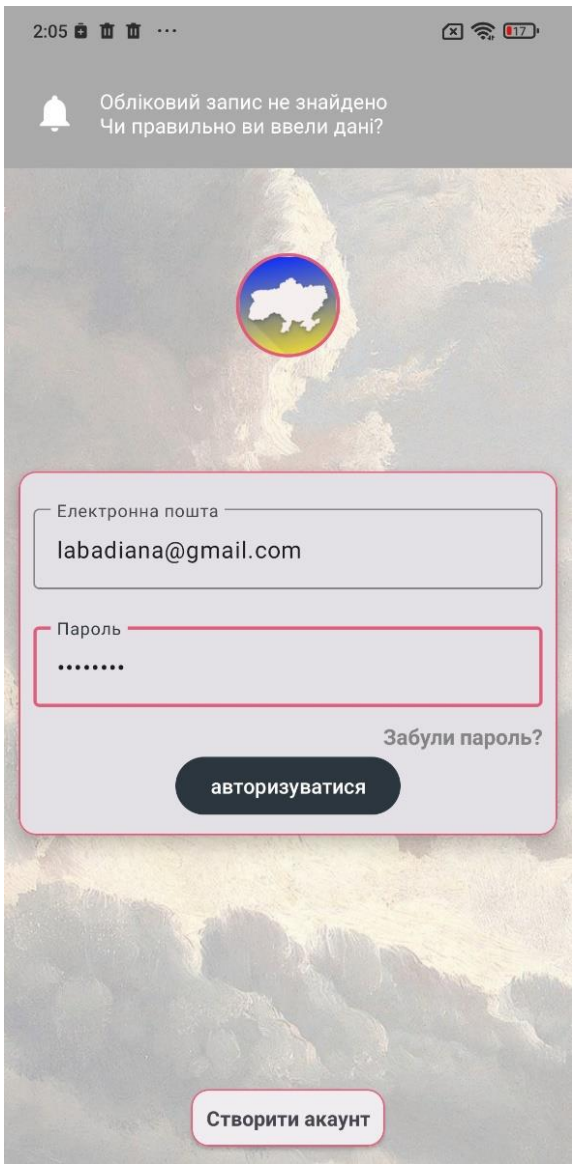

Рисунок 4.3 – Вікно авторизації із забутим паролем

Після успішної реєстрації та авторизації гравця відбувається відкриття головного вікна додатку, де наявні важливі особисті дані користувача, такі як ім'я, кількість монет у його власності та середній бал його гри, що створює повну та інформативну обстановку для подальшого взаємодії з додатком. А ще гравець бачить перед собою різноманітні теми, які є різними за періодами в історії України.

На рисунку 4.4 зображено головне вікно додатку.

Рисунок 4.4 – Головне вікно додатку

Якщо натиснути кнопку « + більше», можна побачити декілька важливих підпунктів (рис.4.5).

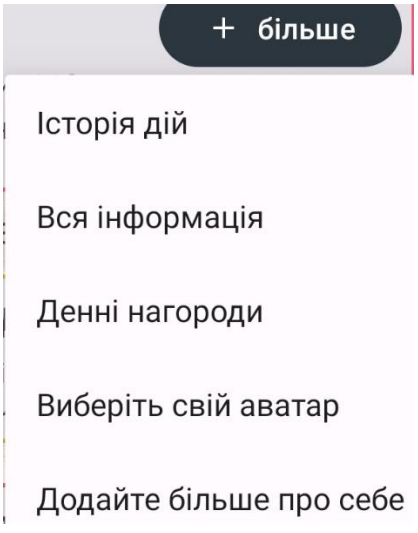

Рисунок 4.5 – Меню кнопки «+більше»

При випадному списку є інформація, яку можна переглянути. Перше – це історія дій користувача. Оскільки деяким користувачам інколи цікаво переглянути свою активність у додатку, то саме ця можливість є реалізованою в додатку. При виборі «Історія дій», гравець може подивитися свою активність протягом дня чи місяця, оскільки на кожній дії стоїть дата. Також при натисканні на одну із дій можемо перейти і подивитися, що саме виконували (рис4.6).

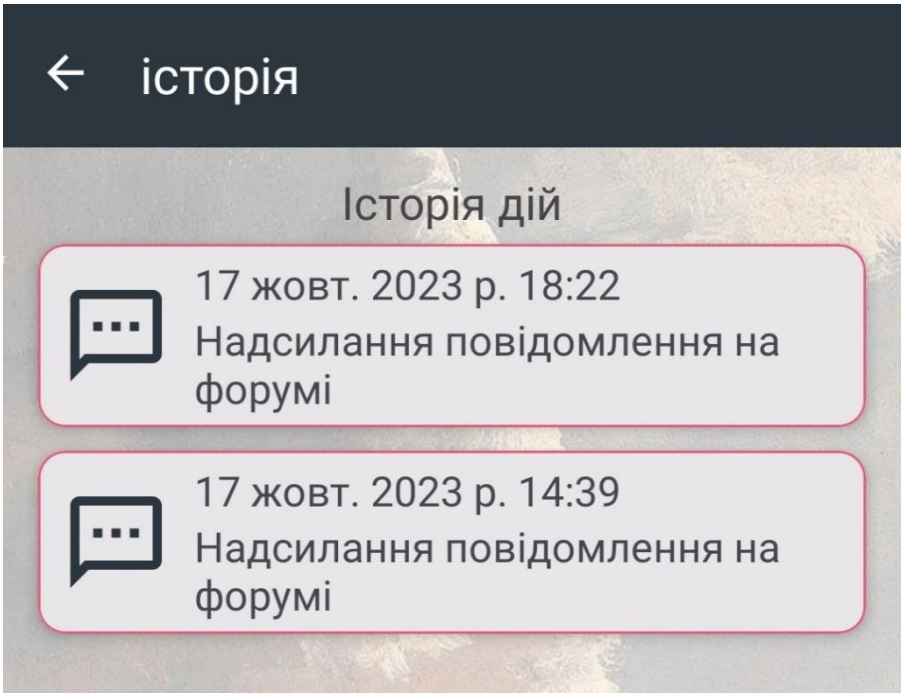

Рисунок 4.6 – Вікно «Історія дій»

Також у меню можна переглянути інформацію про користувача його загальний час проведення в додатку, також середній бал, вік, країну, місце навчання, також не менш важливим є смуга днів, що означа,є скільки днів підряд користувач займався в програмі, та останній вхід, який було здійснено гравцем, а також, де в інформації можна подивитися унікальний ідентифікатор користувача.

На рисунку 4.7 зображено екран з інформацією про користувача.

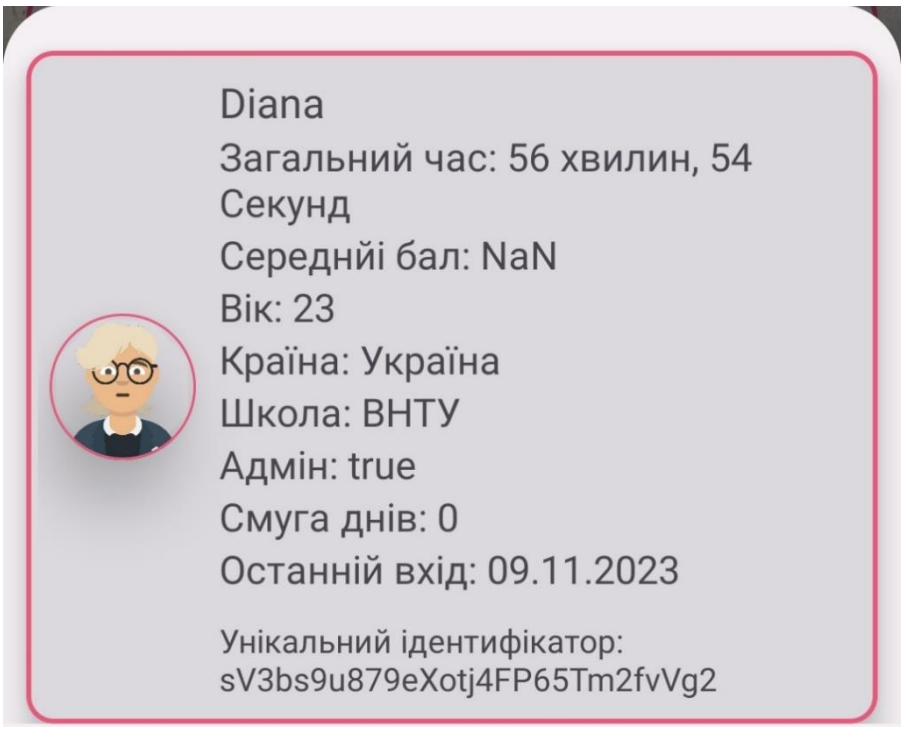

Рисунок 4.7 – Вікно «Вся інформація»

У грі можна заробити монети не тільки при проходженні тестів, а й при постійній активності, коли користувач кожного дня заходить у гру. Кнопка в меню «Денні нагороди» дозволяє зрозуміти, скільки монет у грі було зароблено сьогодні (рис.4.8).

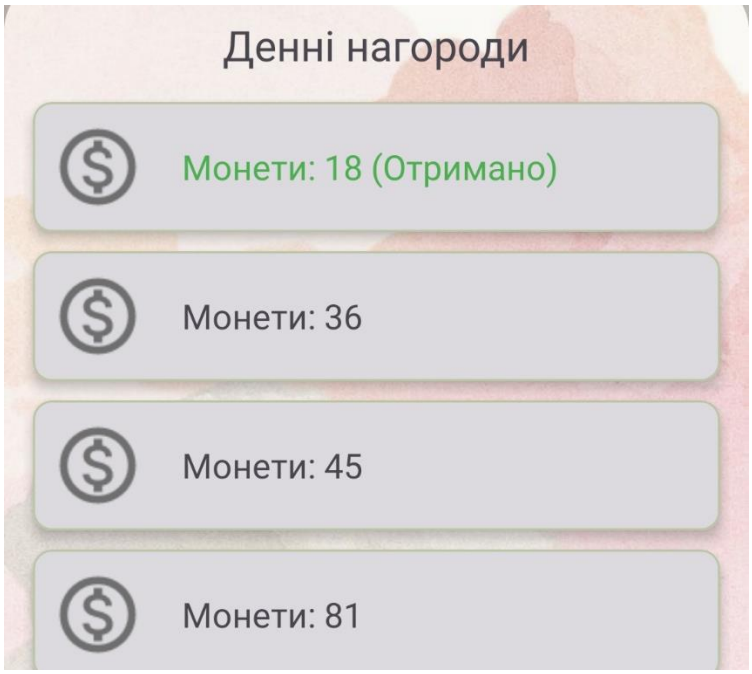

Рисунок 4.8 – Вікно «Денних нагород»

Щоб дізнатися, як користувачі проходять завдання, які тести їм більше до вподоби і т.д., можна подивитися статистику, яка доступна лише для адміністратора додатку. На рисунку 4.9 зображено дві статистики: одна з них показує, скільки було зроблено правильних і не правильних відповідей, інша показує відсоток користувачів, які запускають теми.

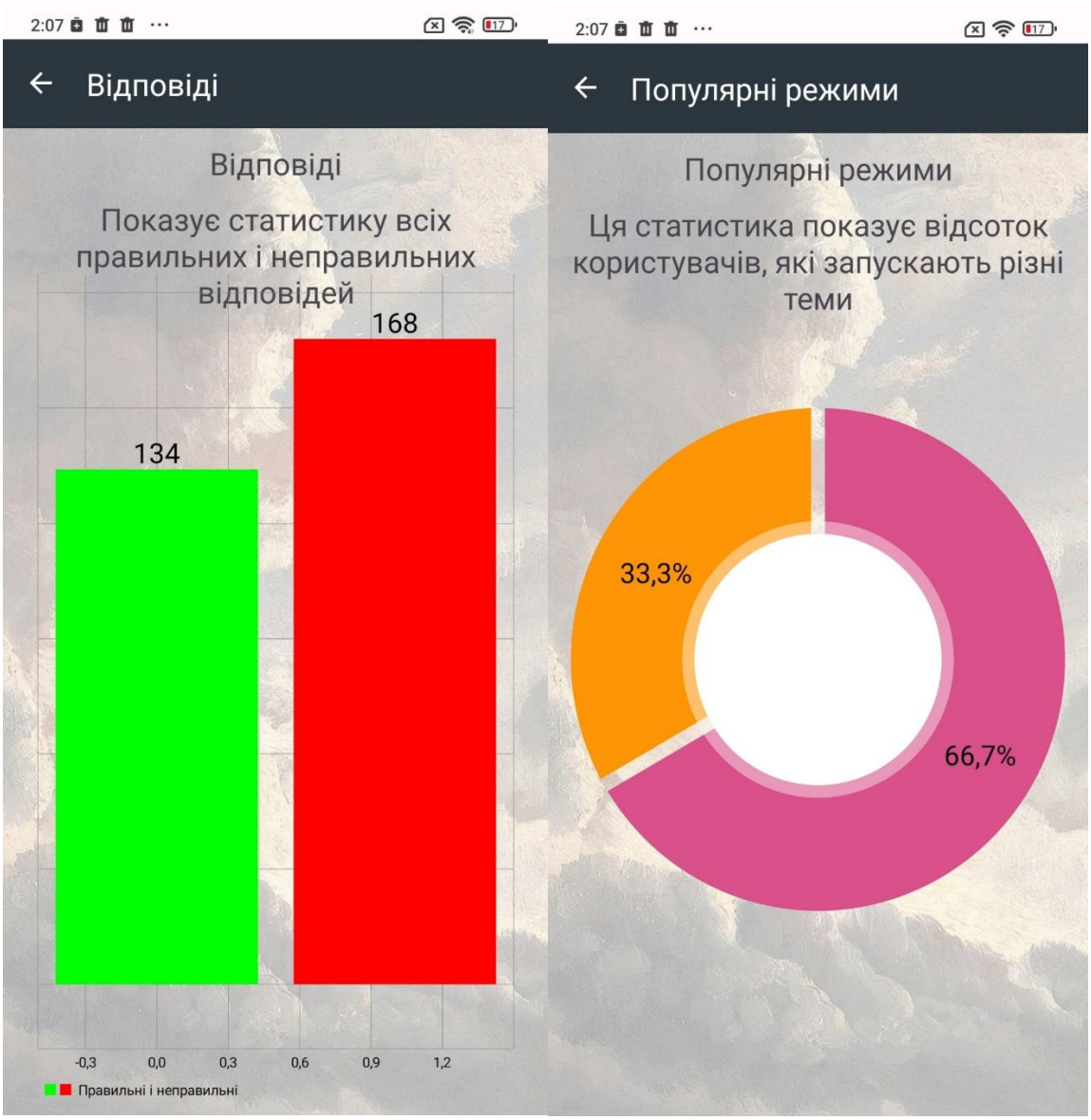

Рисунок 4.9 – Вікно «Статистик»

Після того, як було отримано денні нагороди та було переглянуто статистику, гравець може обрати тему, яка поділяється на види тестів: VIP- тести та звичайні. Вікно звичайної теми наведене на рис 4.10.

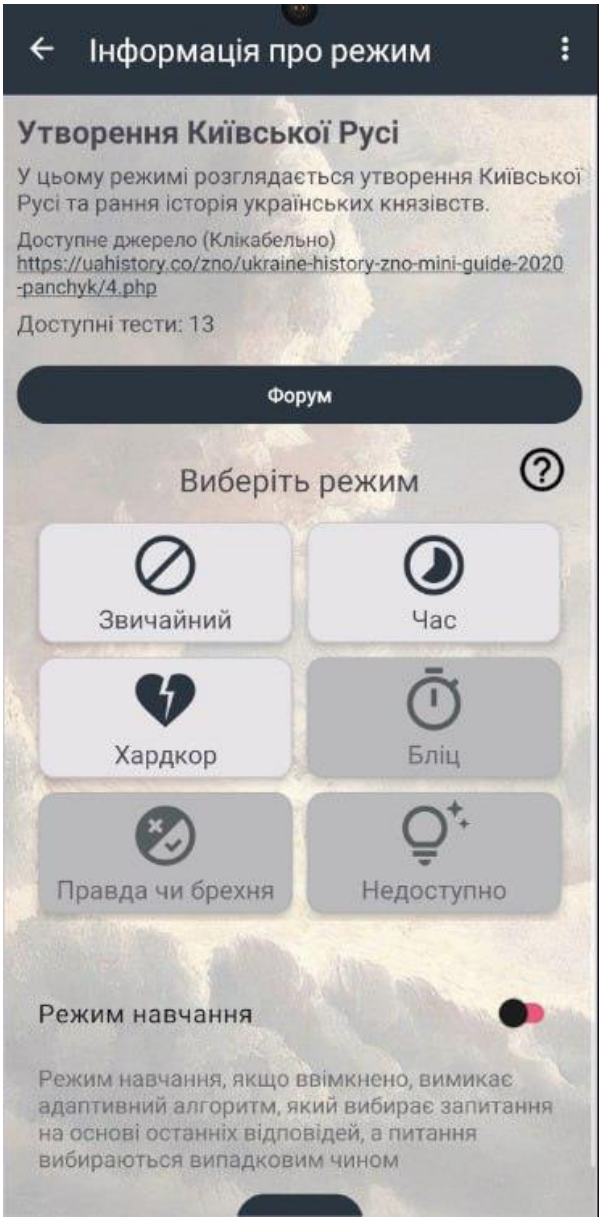

Рисунок 4.10 – Вікно теми

На рисунку 4.11 подане вікно безкоштовної теми, де доступна докладна інформація про тему, а також можливість переходу за відповідним посиланням. Крім того, в цьому вікні користувачі можуть перейти до розділу "Форум", де вони можуть обговорювати тему, спілкуватися та домовлятися про можливі міжусобні змагання. Це доповнює гейміфікований досвід додатка, надаючи спільноті гравців платформу для обміну думками, домовленостей та організації цікавих подій, як показано на рисунку 4.11.

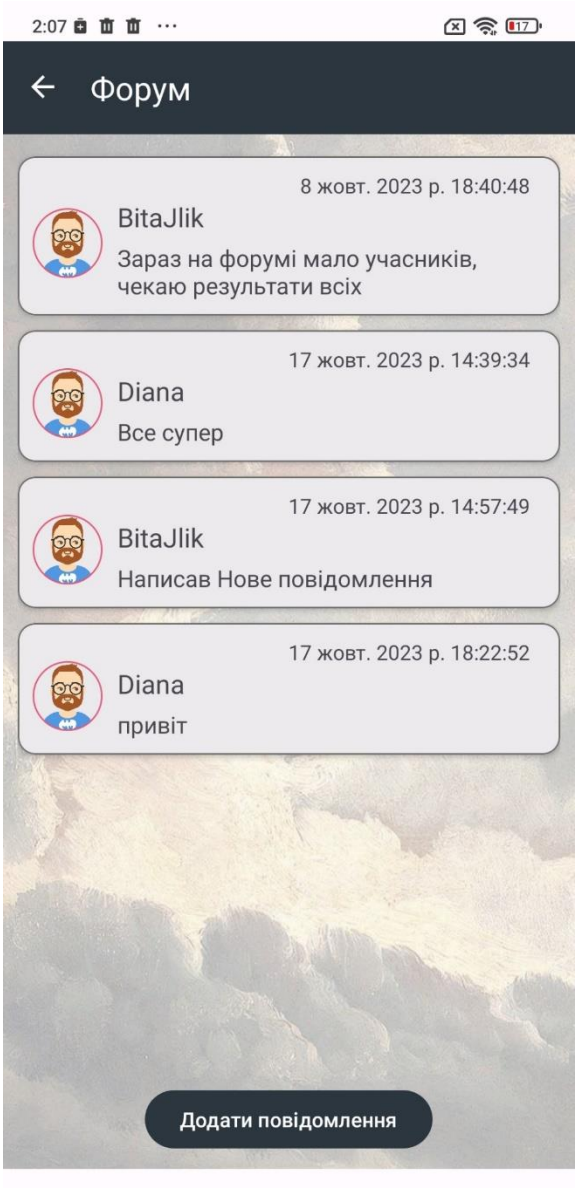

Рисунок 4.11 – Вікно «Форуму»

На рисунку 4.12 відображено можливість користувача обрати відповідний режим перед початком проходження тесту. Це дозволяє користувачеві вибрати режим з різною складністю в залежності від його вподобань. Додаткову інформацію про режими можна отримати, натиснувши на знак запитання. Крім того, користувач може вибрати режим проходження тестів, такий як режим навчання, де відсутній адаптивний підхід, чи, навпаки, вибрати адаптивний режим. Це надає користувачеві більше контролю над його досвідом та сприяє персоналізованому підходу до навчання.

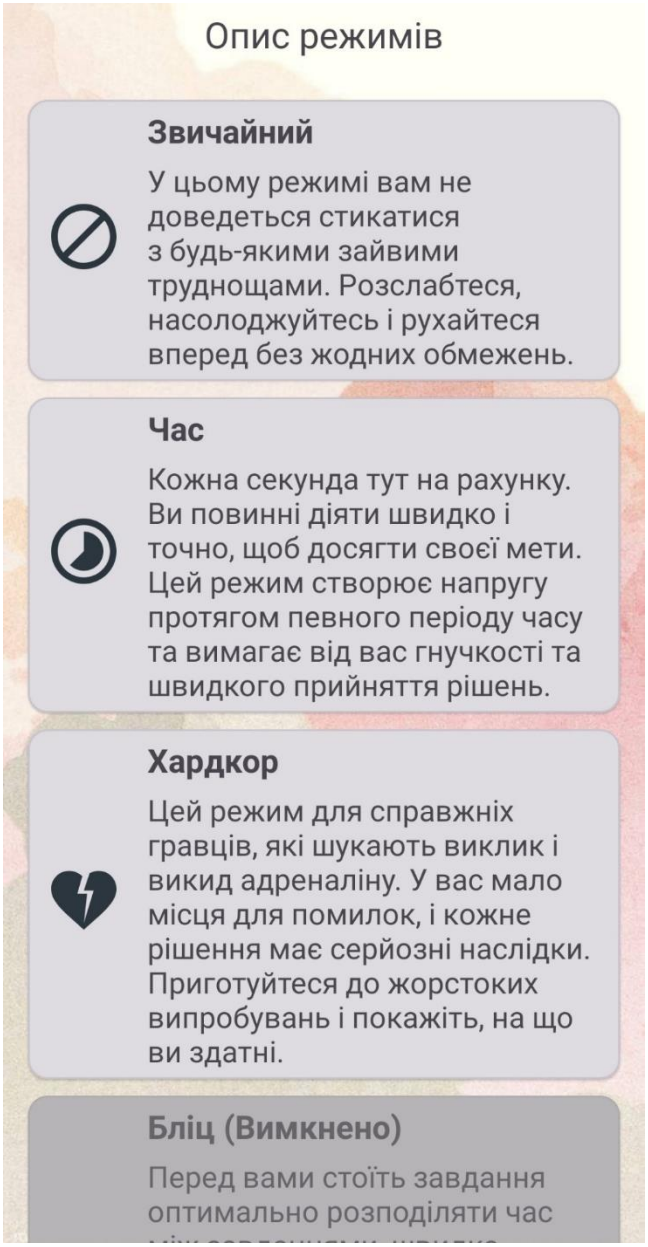

Рисунок 4.12 – Вікно «Опис режимів»

Після того, як користувач ознайомився з різними режимами та обрав оптимальний для себе, він може розпочати проходження тестування. Це включає в себе вибір конкретної теми або тесту, залежно від його інтересів чи потреб в оцінці знань. Під час тестування користувачеві буде запропоновано питання або завдання з вибраної теми, а його відповіді будуть оцінені для надання його знань чи вмінь у даній області. Такий підхід дозволяє персоналізувати досвід користувача та забезпечити ефективний процес навчання чи оцінки.

На рисунку 4.13 зображено вікно тестування.

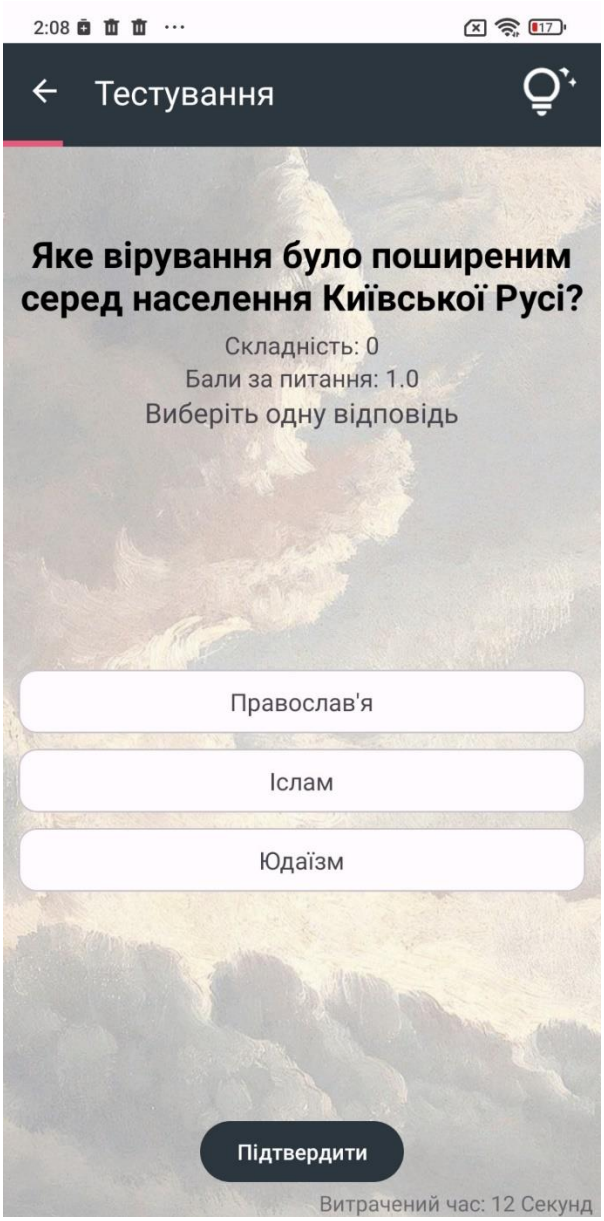

Рисунок 4.13 – Вікно «Тестування»

Тестування в додатку включає в себе різноманітні типи завдань, спрямованих на ефективну перевірку рівня знань користувача. Це можуть бути звичайні тести з однією або декількома відповідями, а також завдання з введенням тексту з клавіатури. Такий різноманітний підхід дозволяє більш точно визначити здатність користувача реагувати на різні варіації завдань та надає більше можливостей для гнучкості у навчанні та оцінюванні.

На рисунку 4.14 подано завдання, в якому користувач повинен вручну ввести відповідь. Цей тип тесту дозволяє оцінити користувача за його здатність

правильно формулювати відповіді та висловлювати свої думки без обмежень вибору з певних варіантів

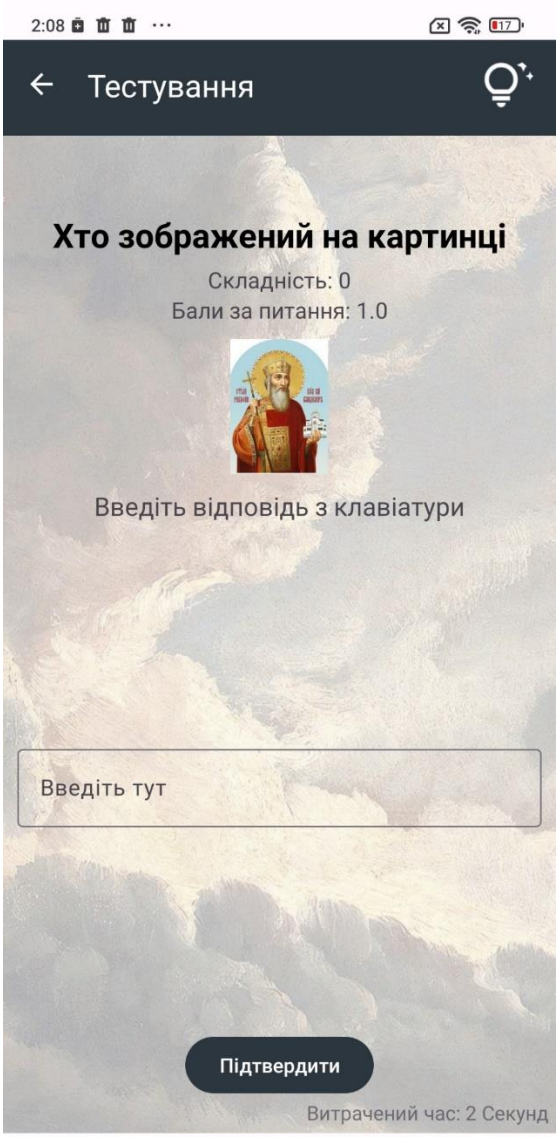

Рисунок 4.14 – Вікно «Тестування із введенням відповіді»

Коли користувач дає правильну відповідь, запитання підсвічується зеленим кольором, вказуючи на успішне вирішення завдання. У випадку неправильної відповіді, запитання підсвічується червоним кольором, що вказує на необхідність перевірки відповіді та можливої корекції. Така візуальна реакція допомагає користувачеві легко розрізнити правильні та неправильні відповіді, сприяючи зрозумінню його успішності в тестуванні (рис 4.15).

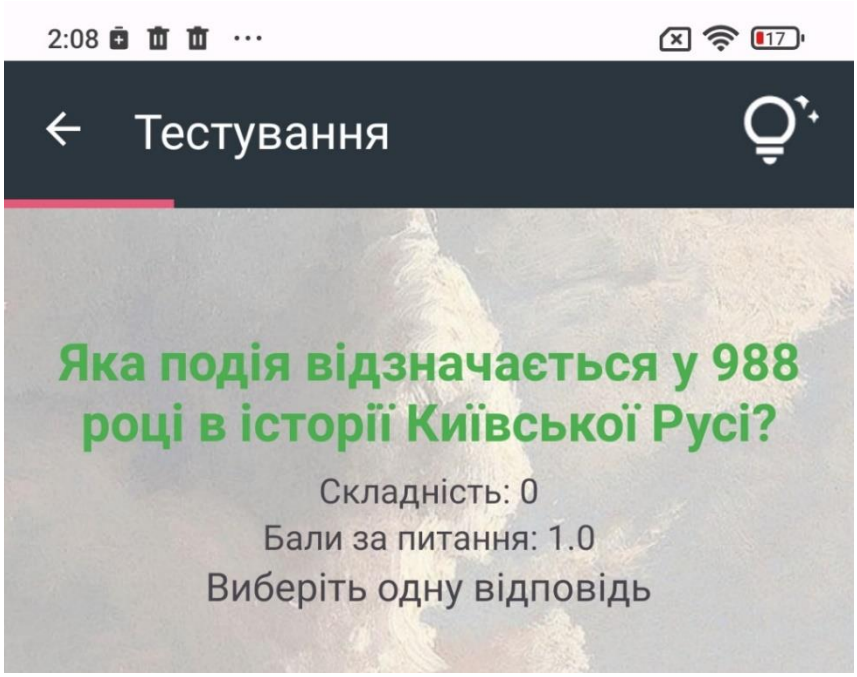

Рисунок 4.15 – Індикатор питання зелений, відповідь вірна

Користувач може використовувати підказки для полегшення проходження тесту. Щоб скористатися підказкою, потрібно натискати на кнопку «ліхтарика», що викликає меню із доступними опціями. Серед підсказок є такі варіанти, як «Видалити половину», яка дозволяє зменшити кількість варіантів відповіді, та «Мінус один», яка видаляє один із варіантів. Важливо зауважити, що використання кожної підказки є платним сервісом, що може стимулювати користувачів до заробітку монет та обдуманого використання підказок (рис.4.16).

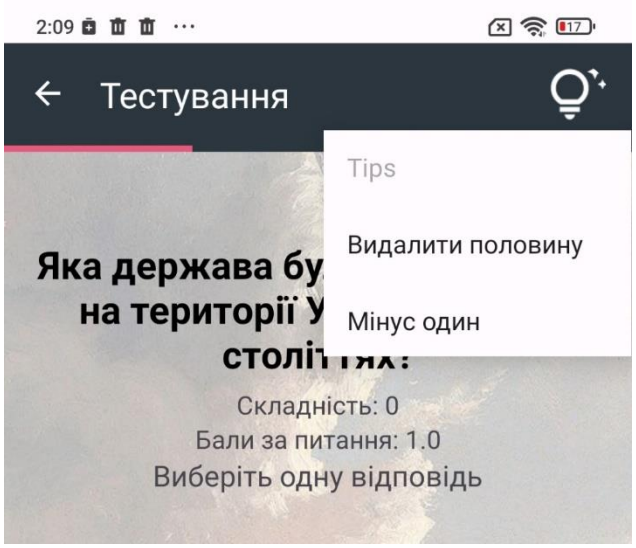

Рисунок 4.16 – Меню підказок

Якщо користувач вже вирішив не проходити тест до кінця, у нього є можливість завершити тест. Ця можливість надає користувачеві гнучкість та контроль над процесом тестування. Він може припинити тестування в будь-який момент, вибрати альтернативні дії, такі як відміна або перехід до інших тем, навіть не дивлячись на результат. Такий підхід покликаний забезпечити зручність використання додатка та можливість враховувати індивідуальні потреби користувачів (4.17).

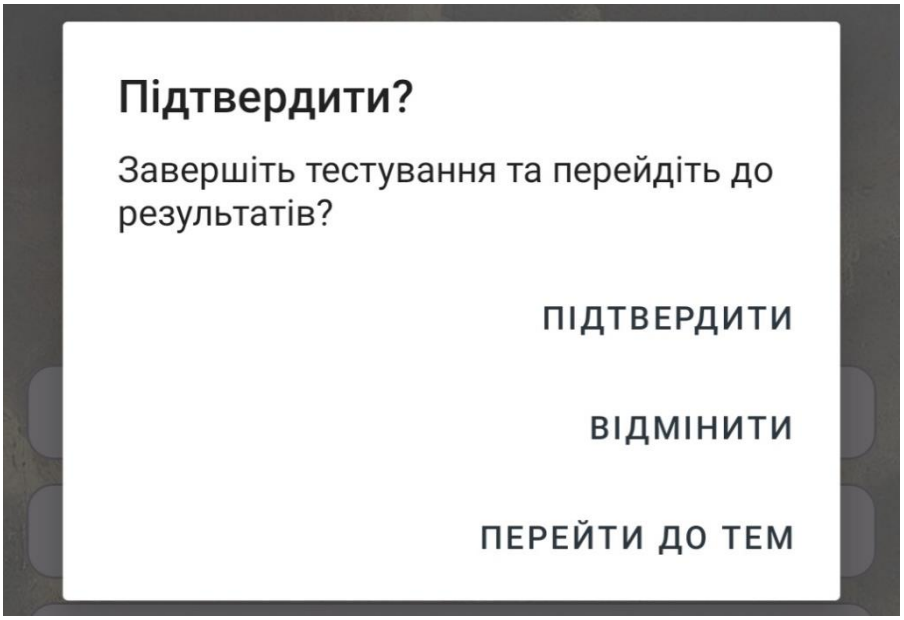

Рисунок 4.17 – Вікно із виходом з тесту

Під час завершення тесту користувачеві надається детальний огляд його прогресу під час тестування, що сприяє кращому розумінню сильних і слабких сторін у своїх знаннях. Статистика включає інформацію про кількість правильних та неправильних відповідей, загальний бал та витрачений час на тестування. Такий докладний звіт допомагає користувачам оцінити свої досягнення, а також може бути використаний для вдосконалення навичок і збереження результатів для подальшого аналізу чи порівняння. Функція також надає можливість зберегти результат для подальшого перегляду або порівняння середнього балу (рис 4.18).

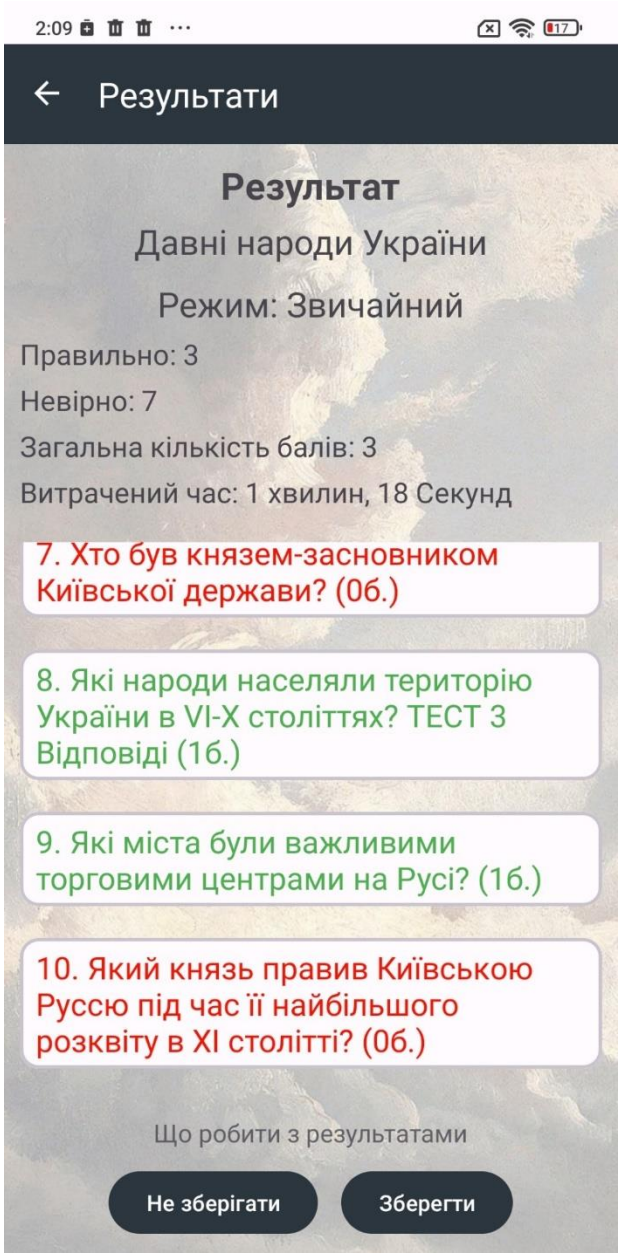

Рисунок 4.18 – Вікно результатів

У ході тестування навчальної гри з історії України було підтверджено, що програма функціонує коректно та відповідає вимогам, визначеним у технічному завданні. Тестування дозволило переконатися у надійності та ефективності розробленого програмного продукту, підтвердивши його готовність до використання та задоволення потреб користувачів.

4.3 Розробка інструкції користувача

Інструкція для користувача передбачає ознайомлення з мінімальними

технічними характеристиками для встановлення застосунку на смартфон. Детальний опис характеристик наведений у таблиці 4.1. Додатково можна вказати, що відповідність цим характеристикам дозволить користувачам насолоджуватися оптимальним та стабільним функціонуванням гри на їхньому смартфоні.

Таблиця 4.1 – Мінімальні технічні характеристики

Характеристика	Опис
Операційна система	Android 5.0 або вище
Процесор	Мінімум 1,0 ГГц, двохядерний або вище
Оперативна пам'ять	Мінімум 512 мегабайт RAM
Вільне місце	Мінімум 40 МБ
Інтерфейси	Wi-Fi, LTE

Для встановлення програмного продукту потрібно запустити файл «WorldLeaner.apk». Після цього з'явиться вікно із встановленням, де користувачеві слід слідувати вказівкам на екрані та підтвердити процес встановлення (рис.4.19).

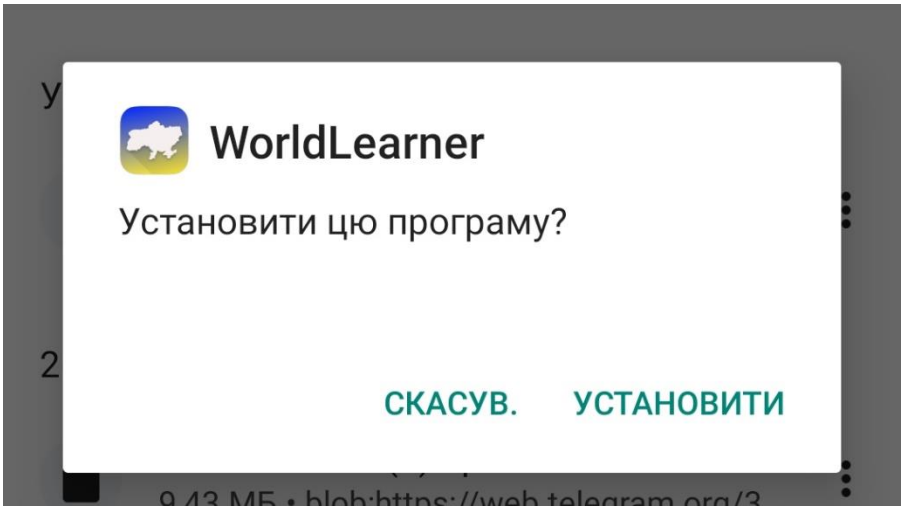

Рисунок 4.19 – Вікно установки

Коли програма успішно встановиться, з'явиться вікно підтвердження цього. Після цього додаток буде повністю готовий до використання, і користувач зможе відкривати його та користуватися всіма доступними функціями (рис.4.20).

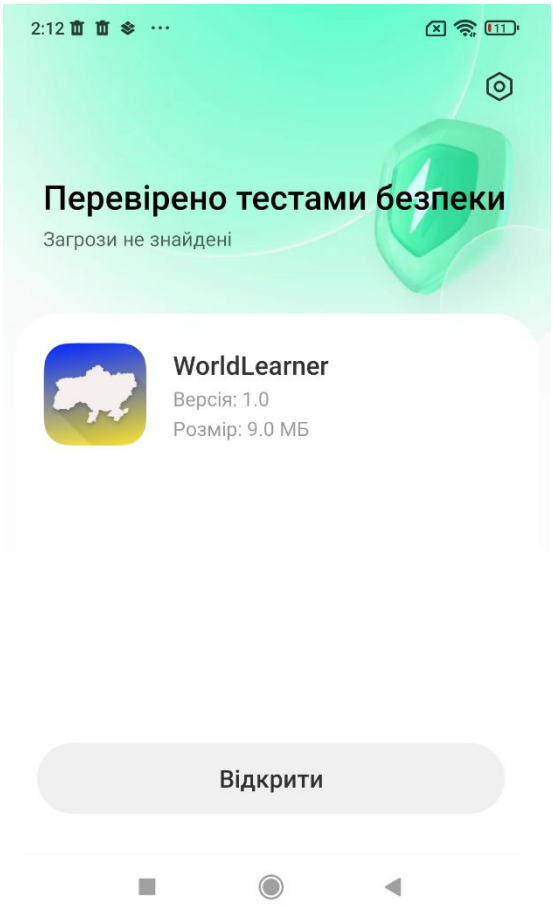

Рисунок 4.20 – Вікно завершення процесу

4.4 Висновки

У четвертому розділі було ретельно проаналізовано методики тестування програмного забезпечення, в результаті чого було обрано тестування методом «чорного ящика», оскільки цей метод дозволяє провести тестування системи в умовах, максимально наближених до ситуацій кінцевого використання.

Проведено комплексне тестування системи, яке підтвердило її працездатність і відповідність поставленим вимогам. Отримані результати свідчать про ефективність функціонування системи під умовами реального використання.

Визначені технічні вимоги для розгортання системи, забезпечуючи її стабільну роботу та високий рівень продуктивності. Крім того, розроблено інструкцію користувача щодо встановлення компонентів системи, що сприяє зручності та ефективності процесу.

5 ЕКОНОМІЧНА ЧАСТИНА

Науково-технічна розробка має право на існування та впровадження, якщо вона відповідає вимогам часу, як в напрямку науково-технічного прогресу та і в плані економіки. Тому для науково-дослідної роботи необхідно оцінювати економічну ефективність результатів виконаної роботи.

Магістерська кваліфікаційна робота за темою «Розробка методів та програмних засобів навчальної гри з історії України для Android-додатку з використанням методів адаптивного навчання» відноситься до науково-технічних робіт, які орієнтовані на виведення на ринок (або рішення про виведення науково-технічної розробки на ринок може бути прийнято у процесі проведення самої роботи), тобто коли відбувається так звана комерціалізація науково-технічної розробки. Цей напрямок є пріоритетним, оскільки результатами розробки можуть користуватися інші споживачі, отримуючи при цьому певний економічний ефект. Але для цього потрібно знайти потенційного інвестора, який би взявся за реалізацію цього проекту і переконати його в економічній доцільності такого кроку.

Для наведеного випадку нами мають бути виконані такі етапи робіт:

1) проведено комерційний аудит науково-технічної розробки, тобто встановлення її науково-технічного рівня та комерційного потенціалу;

2) розраховано витрати на здійснення науково-технічної розробки;

3) розрахована економічна ефективність науково-технічної розробки у випадку її впровадження і комерціалізації потенційним інвестором і проведено обґрунтування економічної доцільності комерціалізації потенційним інвестором.

5.1 Проведення комерційного та технологічного аудиту науково-технічної розробки

Метою проведення комерційного і технологічного аудиту дослідження за темою «Розробка методів та програмних засобів навчальної гри з історії України для Android -додатку з використанням методів адаптивного навчання» є

оцінювання науково-технічного рівня та рівня комерційного потенціалу розробки, створеної в результаті науково-технічної діяльності.

Оцінювання науково-технічного рівня розробки та її комерційного потенціалу рекомендується здійснювати із застосуванням 5-ти бальної системи оцінювання за 12-ма критеріями, наведеними в табл. 5.1 [31].

Таблиця 5.1 – Рекомендовані критерії оцінювання науково-технічного рівня і комерційного потенціалу розробки та бальна оцінка

Бали (за 5-ти бальною шкалою)					
	0	1	2	3	4
Технічна здійсненність концепції					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено працездатність продукту в реальних умовах
Ринкові переваги (недоліки)					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в	Технічні та споживчі влас-тивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в	Технічні та споживчі властивості продукту значно кращі, ніж в
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає

Продовження таблиці 5.1

Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуютьс я у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь- які регламентні обмеження на виробництво та реалізацію продукту

Результати оцінювання науково-технічного рівня та комерційного потенціалу науково-технічної розробки потрібно звести до таблиці 5.2.

Таблиця 5.2 – Результати оцінювання науково-технічного рівня і комерційного потенціалу розробки експертами

Критерії	Експерт (ПІБ, посада)		
	1	2	3
	Бали:		
1. Технічна здійсненність концепції	4	4	3
2. Ринкові переваги (наявність аналогів)	4	4	3
3. Ринкові переваги (ціна продукту)	4	3	4
4. Ринкові переваги (технічні властивості)	3	3	3
5. Ринкові переваги (експлуатаційні витрати)	3	3	3
6. Ринкові перспективи (розмір ринку)	4	4	3
7. Ринкові перспективи (конкуренція)	3	3	4
8. Практична здійсненність (наявність фахівців)	3	3	3
9. Практична здійсненність (наявність фінансів)	4	3	4
10. Практична здійсненність (необхідність нових матеріалів)	3	3	3
11. Практична здійсненність (термін реалізації)	3	3	4
12. Практична здійсненність (розробка документів)	3	3	3
Сума балів	41	39	40
Середньоарифметична сума балів $СБ_c$	40		

За результатами розрахунків, наведених в таблиці 5.2, зробимо висновок щодо науково-технічного рівня і рівня комерційного потенціалу розробки. При цьому використаємо рекомендації, наведені в табл. 5.3 [31].

Таблиця 5.3 – Науково-технічні рівні та комерційні потенціали розробки

Середньоарифметична сума балів СБ ,розрахована на основі висновків експертів	Науково-технічний рівень та комерційний потенціал розробки
41…48	Високий
31…40	Вище середнього
21…30	Середній
11…20	Нижче середнього
0…10	Низький

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Розробка методів та програмних засобів навчальної гри з історії України для Android -додатку з використанням методів адаптивного навчання» становить 40 балів, що, відповідно до таблиці 5.3, свідчить про комерційну важливість

проведення даних досліджень (рівень комерційного потенціалу розробки вище середнього).

5.2 Розрахунок витрат на проведення науково-дослідної роботи

Витрати, пов’язані з проведенням науково-дослідної роботи на тему «Розробка методів та програмних засобів навчальної гри з історії України для Android -додатку з використанням методів адаптивного навчання», під час планування, обліку і калькулювання собівартості науково-дослідної роботи групуємо за відповідними статтями.

5.2.1 Витрати на оплату праці

До статті «Витрати на оплату праці» належать витрати на виплату основної та додаткової заробітної плати керівникам відділів, лабораторій, секторів і груп, науковим, інженерно-технічним працівникам, конструкторам, технологам, креслярам, копіювальникам, лаборантам, робітникам, студентам, аспірантам та іншим працівникам, безпосередньо зайнятим виконанням конкретної теми, обчисленої за посадовими окладами, відрядними розцінками, тарифними ставками згідно з чинними в організаціях системами оплати праці.

Основна заробітна плата дослідників

Витрати на основну заробітну плату дослідників ($З_о$) розраховуємо у відповідності до посадових окладів працівників, за формулою 5.1 [31]:

$$З_о = \sum_{i=1}^{k} \frac{M_{ni} \cdot t_i}{T_p}, \qquad (5.1)$$

де k – кількість посад дослідників залучених до процесу досліджень;

M_{ni} – місячний посадовий оклад конкретного дослідника, грн;

t_i – число днів роботи конкретного дослідника, дні;

T_p – середнє число робочих днів в місяці, T_p=21 день.

$$З_о = 22000{,}00 \cdot 70 / 21 = 73333{,}33 \text{ грн.}$$

Проведені розрахунки зведемо до таблиці 5.4.

Таблиця 5.4 – Витрати на заробітну плату дослідників

Найменування посади	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Число днів роботи	Витрати на заробітну плату, грн
Керівник проекту	22000	1047,62	70	73333,33
Інженер-розробник програмного забезпечення	20000	952,38	70	66666,67
Інженер-розробник програмного забезпечення	20000	952,38	70	66666,67
Консультант	18000	857,14	70	60000,00
Всього				266666,67

Основна заробітна плата робітників

Витрати на основну заробітну плату робітників ($З_р$) за відповідними найменуваннями робіт НДР розраховуємо за формулою 5.2:

$$З_р = \sum_{i=1}^{n} C_i \cdot t_i \,, \tag{5.2}$$

де C_i – погодинна тарифна ставка робітника відповідного розряду, за виконану відповідну роботу, грн/год;

t_i – час роботи робітника при виконанні визначеної роботи, год.

Погодинну тарифну ставку робітника відповідного розряду C_i можна визначити за формулою 5.3:

$$C_i = \frac{M_M \cdot K_i \cdot K_c}{T_p \cdot t_{зм}} \,, \tag{5.3}$$

де M_M – розмір прожиткового мінімуму працездатної особи, або мінімальної місячної заробітної плати (в залежності від діючого законодавства), приймемо $Мм$=6700,00 грн;

K_i – коефіцієнт міжкваліфікаційного співвідношення для встановлення тарифної ставки робітнику відповідного розряду [31];

K_c – мінімальний коефіцієнт співвідношень місячних тарифних ставок робітників першого розряду з нормальними умовами праці виробничих

об’єднань і підприємств до законодавчо встановленого розміру мінімальної заробітної плати.

$T_р$ – середнє число робочих днів в місяці, приблизно $T_р$ = 21 день;

$t_{зм}$ – тривалість зміни, год.

$C_1 = 6700{,}00 \cdot 1{,}10 \cdot 1{,}65 / (21 \cdot 8) = 72{,}38$грн.
$З_{р1} = 72{,}38 \cdot 10{,}00 = 723{,}84$грн.
Величина витрат на основну заробітну плату робітників наведена в табл.5.5.

Таблиця 5.5 – Величина витрат на основну заробітну плату робітників

Найменування робіт	Тривалість роботи, год	Розряд роботи	Тарифний коефіцієнт	Погодинна тарифна ставка, грн	Величина оплати на робітника грн
Підготовка робочого місця дослідника	10	2	1,1	72,38	723,84
Інсталяція програмного забезпечення	2	5	1,7	72,38	144,77
Тестування системи	10	5	1,7	111,87	1118,66
Всього					1987,27

Додаткова заробітна плата дослідників та робітників

Додаткову заробітну плату розраховуємо як 10 … 12% від суми основної заробітної плати дослідників та робітників за формулою 5.4:

$$З_{дод} = (З_о + З_р) \cdot \frac{Н_{дод}}{100\%}, \qquad (5.4)$$

де $Н_{дод}$ – норма нарахування додаткової заробітної плати. Приймемо 11%.

$З_{дод} = (266666{,}67 + 1987{,}27) \cdot 11 / 100\% = 29551{,}93$грн.

5.2.2 Відрахування на соціальні заходи

Нарахування на заробітну плату дослідників та робітників розраховуємо як 22% від суми основної та додаткової заробітної плати дослідників і робітників за формулою 5.5:

$$З_н = (З_о + З_р + З_{дод}) \cdot \frac{Н_{зп}}{100\%} \qquad (5.5)$$

де $Н_{зп}$ – норма нарахування на заробітну плату. Приймаємо 22%.

$Зн = (266666{,}67 + 1987{,}27 + 29551{,}93) \cdot 22 / 100\% = 65605{,}29081$грн.

5.2.3 Сировина та матеріали

До статті «Сировина та матеріали» належать витрати на сировину, основні та допоміжні матеріали, інструменти, пристрої та інші засоби і предмети праці, які придбані у сторонніх підприємств, установ і організацій та витрачені на проведення досліджень.

Витрати на матеріали (*М*), у вартісному вираженні розраховуються окремо по кожному виду матеріалів за формулою 5.6:

$$M = \sum_{j=1}^{n} H_j \cdot Ц_j \cdot К_j - \sum_{j=1}^{n} B_j \cdot Ц_{вj}\ , \qquad (5.6)$$

де H_j – норма витрат матеріалу *j*-го найменування, кг;

n – кількість видів матеріалів;

$Ц_j$ – вартість матеріалу *j*-го найменування, грн/кг;

$К_j$ – коефіцієнт транспортних витрат, ($К_j$ = *1,1 … 1,15*);

B_j – маса відходів *j*-го найменування, кг;

$Ц_{вj}$ – вартість відходів *j*-го найменування, грн/кг.

$М_1 = 3 \cdot 200{,}00 \cdot 1{,}1 - 0{,}000 \cdot 0{,}00 = 660{,}0$ грн.

Проведені розрахунки зведемо до таблиці 5.6.

Таблиця 5.6 – Витрати на матеріали

Найменування матеріалу, марка, тип, сорт	Ціна за од	Норма витрат, од	Величина відходів, кг	Ціна відходів, грн/кг	Вартість витраченого матеріалу, грн
Офісний папір А4-500-80	200	3	0	0	660
Канцелярське приладдя (набір)	150	2	0	0	330
Flesh-пам'ять Kingston 32 GB	180	1	0	0	198
					1188

5.2.4 Розрахунок витрат на комплектуючі

Витрати на комплектуючі *($К_в$)*, які використовують при проведенні НДР на тему «Розробка методів та програмних засобів навчальної гри з історії України для Android -додатку з використанням методів адаптивного навчання» відсутні.

5.2.5 Спецустаткування для наукових (експериментальних) робіт

До статті «Спецустаткування для наукових (експериментальних) робіт» належать витрати на виготовлення та придбання спецустаткування необхідного для проведення досліджень, також витрати на їх проектування, виготовлення, транспортування, монтаж та встановлення.

Балансову вартість спецустаткування розраховуємо за формулою 5.7:

$$B_{спец} = \sum_{i=1}^{k} Ц_i \cdot C_{пр.i} \cdot K_i \quad , \qquad (5.7)$$

де $Ц_i$ – ціна придбання одиниці спецустаткування даного виду, марки, грн;

$C_{пр.i}$ –кількість одиниць устаткування відповідного найменування, які придбані для проведення досліджень, шт.;

K_i – коефіцієнт, що враховує доставку, монтаж, налагодження устаткування тощо, ($K_i = 1,10…1,12$);

k – кількість найменувань устаткування.

$B_{спец} = 18000 \cdot 2 \cdot 1,1 = 39600$ грн.

Отримані результати зведемо до таблиці 5.7:

Таблиця 5.7 – Витрати на придбання спецустаткування по кожному виду

Найменування устаткування	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
Моноблок ARTLINE Business G44v18	2	18 000	39600
Моноблок Acer Aspire C24-1300 (DQ.BL0ME.00L)	2	22 000	48400
Всього			88000

5.2.6 Програмне забезпечення для наукових (експериментальних) робіт

До статті «Програмне забезпечення для наукових (експериментальних) робіт» належать витрати на розробку та придбання спеціальних програмних засобів і програмного забезпечення, (програм, алгоритмів, баз даних) необхідних для проведення досліджень, також витрати на їх проектування, формування та встановлення.

Балансову вартість програмного забезпечення розраховуємо за формулою5.8:

$$B_{прг} = \sum_{i=1}^{k} Ц_{iпрг} \cdot C_{прг.i} \cdot K_i \ , \qquad (5.8)$$

де $Ц_{iпрг}$ – ціна придбання одиниці програмного засобу даного виду, грн;

$C_{прг.i}$ – кількість одиниць програмного забезпечення відповідного найменування, які придбані для проведення досліджень, шт.;

K_i – коефіцієнт, що враховує інсталяцію, налагодження програмного засобу тощо, ($K_i = 1,10…1,12$);

k – кількість найменувань програмних засобів.

$B_{прг} = 9500 \cdot 4 \cdot 1,1 = 42560$ *грн.*

Отримані результати зведемо до таблиці 5.8:

Таблиця 5.8– Витрати на придбання програмних засобів по кожному виду

Найменування програмного засобу	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
ОС Windows 11	4	9500	42560
Прикладний пакет Microsoft Office 2021 Pro	4	5000	22400
Всього			64960

5.2.7 Амортизація обладнання, програмних засобів та приміщень

В спрощеному вигляді амортизаційні відрахування по кожному виду обладнання, приміщень та програмному забезпеченню тощо, розраховуємо з використанням прямолінійного методу амортизації за формулою 5.9:

$$A_{обл} = \frac{Ц_б}{T_в} \cdot \frac{t_{вик}}{12}, \quad (5.9)$$

де $Ц_б$ – балансова вартість обладнання, програмних засобів, приміщень тощо, які використовувались для проведення досліджень, грн;

$t_{вик}$ – термін використання обладнання, програмних засобів, приміщень під час досліджень, місяців;

$T_в$ – строк корисного використання обладнання, програмних засобів, приміщень тощо, років.

$A_{обл}$ = (39600,00 · 3) / (5 · 12) = 1980 грн.

Проведені розрахунки зведемо до таблиці 5.9.

Таблиця 5.9 – Амортизаційні відрахування по кожному виду обладнання

Найменування обладнання	Балансова вартість, грн	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн
Моноблок ARTLINE Business G44v18	39 600	5	3	1980,00
Моноблок Acer Aspire C24-1300 (DQ.BL0ME.00L)	48 400	3	3	4033,33
Оргтехніка	6000	4	2	250,00
Приміщення лабораторії	200000	22	3	2272,73
ОС Windows 11	28500	3	3	2375,00
Прикладний пакет Microsoft Office 2021 Pro	15000	3	3	1250,00
Всього				12161,06

5.2.8 Паливо та енергія для науково-виробничих цілей

Витрати на силову електроенергію ($B_е$) розраховуємо за формулою 5.10:

$$B_е = \sum_{i=1}^{n} \frac{W_{уі} \cdot t_i \cdot Ц_е \cdot K_{впі}}{\eta_i}, \quad (5.10)$$

де W_{yi} – встановлена потужність обладнання на визначеному етапі розробки, кВт;

t_i – тривалість роботи обладнання на етапі дослідження, год;

$Ц_е$ – вартість 1 кВт-години електроенергії, грн; (вартість електроенергії визначається за даними енергопостачальної компанії), приймемо $Ц_е = 7,50$ грн;

$K_{вп i}$ – коефіцієнт, що враховує використання потужності, $K_{вп i} < 1$;

η_i – коефіцієнт корисної дії обладнання, $\eta_i < 1$.

$B_е = 0,1 \cdot 560,0 \cdot 7,50 \cdot 0,95 / 0,97 = 411,34$ *грн.*

Проведені розрахунки зведемо до таблиці 5.10.

Таблиця 5.10 – Витрати на електроенергію

Найменування обладнання	Встановлена потужність, кВт	Тривалість роботи, год	Сума, грн
Моноблок ARTLINE Business G44v18	0,1	560	411,34
Моноблок Acer Aspire C24-1300 (DQ.BL0ME.00L)	0,1	400	293,81
Робоче місце дослідника	0,15	560	617,01
Оргтехніка	0,45	20	66,11
Всього			1388,27

5.2.9 Службові відрядження

До статті «Службові відрядження» дослідної роботи належать витрати на відрядження штатних працівників, працівників організацій, які працюють за договорами цивільно-правового характеру, аспірантів, зайнятих розробленням досліджень, відрядження, пов’язані з проведенням випробувань машин та приладів, а також витрати на відрядження на наукові з’їзди, конференції, наради, пов’язані з виконанням конкретних досліджень.

Витрати за статтею «Службові відрядження» розраховуємо як 20…25% від суми основної заробітної плати дослідників та робітників за формулою 5.11:

$$B_{св} = (З_о + З_р) \cdot \frac{Н_{св}}{100\%}, \qquad (5.11)$$

де $Н_{св}$ – норма нарахування за статтею «Службові відрядження», приймемо $Н_{св}$ = 20%.

$В_{св}$ = (266666,67+ 1987,27) · 20 / 100% = 53730,79 грн.

5.2.10 Витрати на роботи, які виконують сторонні підприємства, установи і організації

Витрати за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації» розраховуємо як 30…45% від суми основної заробітної плати дослідників та робітників за формулою 5.12:

$$В_{сп} = (З_о + З_р) \cdot \frac{Н_{сп}}{100\%}, \quad (5.12)$$

де $Н_{сп}$ – норма нарахування за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації», приймемо $Н_{сп}$= 40%.

$В_{сп}$ = (266666,67+ 1987,27) · 40 / 100% = 107461,57 грн.

5.2.11 Інші витрати

До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками.

Витрати за статтею «Інші витрати» розраховуємо як 50…100% від суми основної заробітної плати дослідників та робітників за формулою 5.13:

$$І_в = (З_о + З_р) \cdot \frac{Н_{ів}}{100\%}, \quad (5.13)$$

де $Н_{ів}$ – норма нарахування за статтею «Інші витрати», приймемо $Н_{ів}$ = 50%.

$І_в$ = (266666,67+ 1987,27) · 50 / 100% 134326,97 грн.

5.2.12 Накладні (загальновиробничі) витрати

До статті «Накладні (загальновиробничі) витрати» належать: витрати, пов’язані з управлінням організацією; витрати на винахідництво та раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів;

витрати, пов'язані з набором робочої сили; витрати на оплату послуг банків; витрати, пов'язані з освоєнням виробництва продукції; витрати на науково-технічну інформацію та рекламу та ін.

Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуємо як 100…150% від суми основної заробітної плати дослідників та робітників за формулою 5.14:

$$B_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%}\ , \qquad (5.14)$$

де $Н_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати», приймемо $Н_{нзв}$ = 120%.

$B_{нзв} = (266666,67 + 1987,27) \cdot 100 / 120\% = 322\,384,72$ *грн.*

Витрати на проведення науково-дослідної роботи розраховуємо як суму всіх попередніх статей витрат за формулою 5.15:

$$B_{заг} = З_о + З_р + З_{дод} + З_н + М + К_в + B_{спец} + B_{прг} + A_{обл} + B_е + B_{св} + B_{сп} + І_в + B_{нзв}. \qquad (5.15)$$

$B_{заг} = 1149412,54$ *грн.*

Загальні витрати *ЗВ* на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховується за формулою 5.16:

$$ЗВ = \frac{B_{заг}}{\eta}, \qquad (5.16)$$

де η - коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи, приймемо η=0,9.

$ЗВ = 1149412,547 / 0,9 = 1277125,05$ *грн.*

5.3 Розрахунок економічної ефективності науково-технічної розробки при її можливій комерціалізації потенційним інвестором

В ринкових умовах узагальнюючим позитивним результатом, що його може отримати потенційний інвестор від можливого впровадження результатів тієї чи іншої науково-технічної розробки, є збільшення у потенційного інвестора величини чистого прибутку.

Результати дослідження передбачають комерціалізацію протягом 3-х років реалізації на ринку.

В цьому випадку майбутній економічний ефект буде формуватися на основі таких даних:

ΔN – збільшення кількості споживачів продукту, у періоди часу, що аналізуються, від покращення його певних характеристик;

1-й рік – 6000 користувачів;

2-й рік –8000 користувачів;

3-й рік – 4000 користувачів.

N – кількість споживачів які використовували аналогічний продукт у році до впровадження результатів нової науково-технічної розробки, приймемо 30000 користувачів;

$Ц_б$ – вартість програмного продукту у році до впровадження результатів розробки, приймемо 2220 грн як платна підписка для одного користувача на рік;

$\pm\Delta Ц_o$ – зміна вартості програмного продукту від впровадження результатів науково-технічної розробки, приймемо 500,00 грн.

Можливе збільшення чистого прибутку у потенційного інвестора $\Delta П_i$ для кожного із 3-х років, протягом яких очікується отримання позитивних результатів від можливого впровадження та комерціалізації науково-технічної розробки, розраховуємо за формулою 5.17 [32]:

$$\Delta П_i = (\pm\Delta Ц_o \cdot N + Ц_o \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}), \qquad (5.17)$$

де λ – коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість. У 2023 році ставка податку на додану вартість складає 20%, а коефіцієнт $\lambda = 0,8333$;

ρ – коефіцієнт, який враховує рентабельність інноваційного продукту. Приймемо $\rho = 40\%$;

ϑ – ставка податку на прибуток, який має сплачувати потенційний інвестор, у 2023 році ϑ =*18%*;

Збільшення чистого прибутку 1-го року:

$\Delta П_1$ =(500·30000,00+2720·6000)·0,83·0,4·(1-0,18/100%)= 8557375,68 грн.

Збільшення чистого прибутку 2-го року:

$\Delta П_2$ =(500·30000,00+2720·(6000+8000))·0,83·0,4(1-0,18/100%)= 9 14502729,92грн.

Збільшення чистого прибутку 3-го року:

$\Delta П_3$ =(500·30000,00+2720·(6000+8000+4000))·0,83·0,4·(1-0,18/100%)= 17475407,04 грн.

Приведена вартість збільшення всіх чистих прибутків *ПП*, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки, розраховується за формулою 5.18:

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \quad (5.18)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному з років, протягом яких виявляються результати впровадження науково-технічної розробки, грн;

T – період часу, протягом якого очікується отримання позитивних результатів від впровадження та комерціалізації науково-технічної розробки, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, τ =0,15;

t – період часу (в роках) від моменту початку впровадження науково-технічної розробки до моменту отримання потенційним інвестором додаткових чистих прибутків у цьому році.

$ПП$ = 8557375,68/(1+0,15)^1+ 14502729,92/(1+0,15)^2+ 1 17475407,04/(1+0,15)^3= 29897707,43грн.

Величина початкових інвестицій PV, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки, розраховується за формулою 5.19:

$$PV = k_{інв} \cdot ЗВ, \quad (5.19)$$

де $k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію, приймаємо $k_{інв}$=2;

$ЗВ$ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, приймаємо 1277125,05грн.

$PV = k_{інв} \cdot ЗВ$ = 2· 1277125,05= 2554250,092грн.

Абсолютний економічний ефект $E_{абс}$ для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки за формулою 5.20 становитиме:

$$E_{абс} = ПП - PV \quad (5.20)$$

де $ПП$ – приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації науково-технічної розробки, 29897707,43грн;

PV – теперішня вартість початкових інвестицій, 2554250,092 грн.

$E_{абс} = ПП - PV$ = 29897707,43- 2554250,092= 27343457,34грн.

Внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки, розраховується за формулою 5.21:

$$E_в = \sqrt[Tж]{1 + \frac{E_{абс}}{PV}} - 1, \quad (5.21)$$

де $E_{абс}$ – абсолютний економічний ефект вкладених інвестицій, 27343457,34грн;

PV – теперішня вартість початкових інвестицій, 2554250,092грн;

$T_{ж}$ – життєвий цикл науково-технічної розробки, тобто час від початку її розробки до закінчення отримування позитивних результатів від її впровадження, 3 роки.

$$E_{в} = \sqrt[T_{ж}]{1 + \frac{E_{абс}}{PV}} - 1 = (1 + 27343457{,}34 / 2554250{,}092)^{1/3} = 1{,}27.$$

Мінімальна внутрішня економічна дохідність вкладених інвестицій $\tau_{мін}$, розраховується за формулою 5.22:

$$\tau_{мін} = d + f, \tag{5.22}$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2023 році в Україні d =0,1;

f – показник, що характеризує ризикованість вкладення інвестицій, приймемо 0,18.

$\tau_{мін}$ = 0,1+0,18 = 0,28 < 1,27 свідчить про те, що внутрішня економічна дохідність інвестицій $E_{в}$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки вища мінімальної внутрішньої дохідності. Тобто інвестувати в науково-дослідну роботу за темою «Розробка методів та програмних засобів навчальної гри з історії України для Android -додатку з використанням методів адаптивного навчання» доцільно.

Період окупності інвестицій $T_{ок}$ які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки, розраховується за формулою 5.23:

$$T_{ок} = \frac{1}{E_{в}}, \tag{5.23}$$

де $E_{в}$ – внутрішня економічна дохідність вкладених інвестицій.

$T_{ок}$ = 1 / 1,27 = 0,79 року.

$T_{ок} <$ 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

5.4 Висновки

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Розробка методів та програмних засобів навчальної гри з історії України для Android-додатку з використанням методів адаптивного навчання» становить 40 балів, що, свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки вище середнього).

Також термін окупності становить 0,79 р., що менше 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

Отже можна зробити висновок про доцільність проведення науково-дослідної роботи за темою «Розробка методів та програмних засобів навчальної гри з історії України для Android-додатку з використанням методів адаптивного навчання».

ВИСНОВКИ

Під час виконання магістерської кваліфікаційної роботи було розроблено навчальний додаток з історії України з адаптивним методом навчання. Для розробки було використано середовище програмування Android Studio. Робота оформлена згідно методичних вказівок [33].

Був проведений аналіз поточного стану даного питання, описані основні аналоги, виокремлені їхні особливості та недоліки. Надано детальне порівняння цих аналогів із розробленим власним програмним продуктом. У результаті такого порівняння розкриті основні аспекти, які внесені в розроблений продукт у сферу вивчення історії України в порівнянні з існуючими аналогами. Зазначено, яким чином розроблений додаток вирішує проблеми, які є властиві іншим системам, і як він забезпечує покращення у процесі навчання користувачів. Було обрано мову програмування Java та систему управління базами даних обрано Firebase.

Було розроблено навчальну гру з історії України, яка включає модуль для реєстрації та авторизації користувачів, метод індивідуалізації з адаптивним тестуванням та модель нарахування балів й отримання бонусів для навчальної гри.

Подальшого розвитку дістав метод індивідуалізації навчального процесу, який, на відміну від існуючих, спрямований на адаптацію навчання для кожного учня з метою оптимізації розуміння і запам'ятовування матеріалу, та враховує особливості кожного учня, їхні потреби, рівень знань і темп навчання, також надає можливість налаштувати навчання під кожну конкретну ситуацію.

Подальшого розвитку дістала модель нарахування балів та отримання бонусів для навчальної гри, яка, на відміну від існуючих, орієнтована на мотивацію, що надає бонусні бали за досягнення певних цілей або завдань та адаптує рівень складності завдань для кожного учасника на основі їхнього рівня знань і навичок, забезпечуючи оптимальний розвиток користувачів у зручному режимі роботи.

Було проведено тестування додатку, під час якого була підтверджена його повна працездатність і відповідність поставленому технічному завданню. Результати тестування свідчать про ефективність функціоналу та стабільну роботу програмного забезпечення.

Розроблена інструкція користувача, яка детально описує процес використання програми. Інструкція спрощує взаємодію користувачів з програмним продуктом, надаючи їм чіткі та послідовні кроки для використання різноманітних функцій та можливостей. Такий підхід сприяє зручності використання та позитивному досвіду користувачів під час взаємодії з розробленим додатком.

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. Войтко В.В. Розробка навчальної гри з історії України як Android-додатку з використанням методі адаптивного навчання / В.В. Войтко, Н.Є. Барчук, О.В. Гаврилюк, Д.С. Лаба // Електронні інформаційні ресурси: створення, використання, доступ. Збірник матеріалів Міжнародної науково-практичної Інтернет конференції 20-21 листопада 2023 р. – Суми/Вінниця: НІКО/ 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. «ЗНО 2024. Історія України» [Електронний ресурс]. – Режим доступу до ресурсу: https://play.google.com/store/apps/details?id=solid.icon.znoapp&hl
9. «Історія України» [Електронний ресурс]. – Режим доступу до ресурсу: https://play.google.com/store/apps/details?id=io.exgames.timeline.ukraine&hl=

10.«ЗНО 2023: Історія України» [Електронний ресурс]. – Режим доступу до ресурсу: https://play.google.com/store/apps/details?id=disanumber.historyzno2018&hl=
11.Розробка мобільних додатків і їх види [Електронний ресурс]. – Режим доступу до ресурсу: http://apeps.kpi.ua/rozrobka-mobilnykh-dodatkiv-i-yii-vidy
12.Model-View-ViewModel [Електронний ресурс]. – Режим доступу до ресурсу: https://www.wikidata.uk-ua.nina.az/Model-View-ViewModel.html
13.LiveData overview [Електронний ресурс]. – Режим доступу до ресурсу: https://developer.android.com/topic/libraries/architecture/livedata
14.View Binding in Android Jetpack [Електронний ресурс]. – Режим доступу до ресурсу: https://www.geeksforgeeks.org/view-binding-in-android-jetpack/
15. Reto Meier Professional Android 4 Application Development 3rd Edition / Reto Meier – Wrox, 2012. – 816 p.
16.MySQL Crash Course: A Hands-on Introduction to Database Development / Rick Silva // No Starch Press – May 23, 2023 – p. 352
17.Бази даних та СУБД. [Електронний ресурс]. Режим доступу до ресурсу: https://timeweb.com/ru/community/articles/bazy-dannyh-i-subd-1
18.Що таке FIREBASE? [Електронний ресурс]. – Режим доступу до ресурсу: https://avada-media.ua/ua/services/firebase/
19.Принцип індивідуалізації і колективності навчання [Електронний ресурс]. – Режим доступу до ресурсу: https://studfile.net/preview/5722650/page:11/
20.Westwood P. Inclusive and Adaptive Teaching: Meeting the Challenge of Diversity in the Classroom. Taylor & Francis Group, 2018. 128 p.
21.Федорук П.І. Адаптивні тести: статистичні методи аналізу результатів тестового контролю знань //Математичні машини і системи. – 2007. – № 3,4. – С. 122-138.
22. Новотарський М. А. Алгоритми та методи обчислень: навч. посіб. для студ. спеціальностей 121 «Інженерія програмного забезпечення», спеціалізації «Програмне забезпечення високопродуктивних комп’ютерних систем та

мереж» та 123 «Комп'ютерна інженерія», спеціалізації «Комп'ютерні системи та мережі». Київ : КПІ ім. Ігоря Сікорського, 2019, 407 с.

23. Kotlin і Java: яку мову краще обирати для вивчення [Електронний ресурс]. – Режим доступу до ресурсу: https://foxminded.ua/kotlin-abo-java/
24. Розробка мобільних додатків TECHNOLOGIES вивчення [Електронний ресурс]. – Режим доступу до ресурсу: https://brander.ua/technologies/kotlin
25. Мова програмування Kotlin [Електронний ресурс]. – Режим доступу до ресурсу: https://lemon.school/blog/mova-programuvannya-kotlin
26. Benjamin Evans, Jason Clark, Martijn Verburg The Well-Grounded Java Developer, 2nd Edition, Manning 2022, 704 p.
27. Переваги мови програмування Java [Електронний ресурс]. – Режим доступу до ресурсу: https://np.pl.ua/2021/03/perevahy-movy-prohramuvannia-java/
28. Різниця між Android Studio та Eclipse [Електронний ресурс]. – Режим доступу до ресурсу: https://uk.strephonsays.com/android-studio-and-vs-eclipse-13109
29. Android Studio: переваги та особливості [Електронний ресурс] – Режим доступу до ресурсу: https://qagroup.com.ua/publications/android-studio-perevagy-ta-osoblyvosti
30. Тестування програмного забезпечення: типи, види та застосування [Електронний ресурс]. – Режим доступу до ресурсу: https://foxminded.ua/testuvannia-prohramnoho-zabezpechennia/
31. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт / Уклад. : В. О. Козловський, О. Й. Лесько, В. В. Кавецький. Вінниця : ВНТУ, 2021. 42 с.
32. Кавецький В. В. Економічне обґрунтування інноваційних рішень: практикум / В. В. Кавецький, В. О. Козловський, І. В. Причепа. Вінниця : ВНТУ, 2016. 113 с.
33. Методичні вказівки до виконання магістерської кваліфікаційної роботи для студентів спеціальності 121 «Інженерія програмного забезпечення» / уклад. : О. Н. Романюк, Г. О. Черноволик. – Вінниця : ВНТУ, 2022. – 50 с.

ДОДАТКИ

Додаток А – Технічне завдання

Міністерство освіти і науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ

к.т.н., проф. О. Н. Романюк

"19" вересня 2023 р.

Технічне завдання

на магістерську кваліфікаційну роботу «Розробка методів та програмних засобів навчальної гри з історії України для Android -додатку з використанням методів адаптивного навчання» за спеціальністю 121 – Інженерія програмного забезпечення

Керівник магістерської кваліфікаційної роботи:

_____ к.т.н., доц. В.В. Войтко

"19" вересня 2023 р.

Виконав:

_____ студент гр.3ПІ-22м Д.С. Лаба

"19" вересня 2023 р.

Вінниця – 2023 року

1. Найменування та галузь застосування

Магістерська кваліфікаційна робота: «Розробка методів та програмних засобів навчальної гри з історії України для Android -додатку з використанням методів адаптивного навчання».

Галузь застосування – Android-додатки для навчальних систем.

2. Підстава для розробки.

Підставою для виконання магістерської кваліфікаційної роботи (МКР) є індивідуальне завдання на МКР та наказ ректора № 247 від 18 вересня 2023 р. по ВНТУ про закріплення тем МКР.

3. Мета та призначення розробки.

Метою роботи є підвищення ефективності якості навчального процесу шляхом розробки навчальної гри для Android-додатку, спрямованої на вивчення історії України з використанням методів адаптивного навчання та створенням інтерактивного навчального середовища, яке автоматично пристосовується до поточного рівня знань та індивідуальних потреб користувачів, сприяючи ефективному і цікавому процесу вивчення історії України.

Призначення роботи – розробка методів і засобів реалізації навчальної гри з історії України з використанням методів адаптивного навчання.

4 Вихідні дані для проведення НДР

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

1. Гейміфікація та ігрове навчання, у чому різниця? [Електронний ресурс]. – Режим доступу до ресурсу: https://www.futureschool.online
2. Коваленко О. О. Моделі гейміфікації в системах управління навчанням: монографія / О. О. Коваленко, Є. А. Паламарчук. – Вінниця : ВНТУ, 2023. – 85 с.
3. Westwood P. Inclusive and Adaptive Teaching: Meeting the Challenge of Diversity in the Classroom. Taylor & Francis Group, 2018. 128 p.

4. Федорук П.І. Адаптивні тести: статистичні методи аналізу результатів тестового контролю знань //Математичні машини і системи. – 2007. – № 3,4. – С. 122-138.

5. Технічні вимоги

Вихідні дані до роботи: середовище розробки Android Studio, мова розробки Java, система керування базами даних – Firebase, операційна система – Windows 10, метод адаптивного навчання.

6. Конструктивні вимоги.

Інтерфейс Android-додатку повинен відповідати естетичним та ергономічним вимогам, програма має бути бути зручною в обслуговуванні та керуванні.

Графічна та текстова документація повинна відповідати діючим стандартам України.

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

– пояснювальна записка до МКР;

– технічне завдання;

– лістинги програми.

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

9. Стадії та етапи розробки:

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи
1	Аналіз стану розвитку навчальних ігор та постановка задач дослідження	20.09.2023 – 30.09.2023
2	Розробка методу та моделей навчальної системи	01.10.2023 – 17.10.2023
3	Розробка програмних модулів навчальної гри з історії України для Android-додатку з використанням методів адаптивного навчання	18.10.2023 – 04.11.2023
4	Тестування програми	05.11.2023 – 21.11.2023
5	Економічна частина	22.11.2023 – 01.12.2023

10. Порядок контролю та прийняття.

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

105

Додаток Б – Протокол перевірки на плагіат

ПРОТОКОЛ ПЕРЕВІРКИ НАВЧАЛЬНОЇ (КВАЛІФІКАЦІЙНОЇ) РОБОТИ

Назва роботи: **Розробка методів та програмних засобів навчальної гри з історії України для Android -додатку з використанням методів адаптивного навчання**

Тип роботи: магістерська кваліфікаційна робота

Підрозділ : кафедра програмного забезпечення, ФІТКІ, ЗПІ – 22м

Науковий керівник: Войтко В.В.

Unicheck	
Оригінальність	**96,7 %**
Схожість	3,3 %

Аналіз звіту подібності

■ Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.

□ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності 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 приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Черноволик Г. О.

Опис прийнятого рішення: **допустити до захисту**

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck

Автор роботи _____ Лаба Д.С.

Керівник роботи _____ Войтко В.В.

Додаток В – Лістинг програми

```
package com.vn.worldlearner.ui.fragment.test;

import android.animation.Animator;
import android.animation.AnimatorListenerAdapter;
import android.animation.ArgbEvaluator;
import android.animation.ObjectAnimator;
import android.animation.ValueAnimator;
import android.os.Bundle;
import android.view.Menu;
import android.view.View;
import android.widget.PopupMenu;
import android.widget.TextView;

import androidx.annotation.ColorRes;
import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.core.content.ContextCompat;
import androidx.core.view.ViewCompat;
import androidx.lifecycle.MutableLiveData;
import androidx.recyclerview.widget.LinearLayoutManager;

import com.github.abdularis.civ.CircleImageView;
import com.google.android.material.appbar.AppBarLayout;
import com.google.android.material.snackbar.Snackbar;
import com.squareup.picasso.Picasso;
import com.tapadoo.alerter.Alerter;
import com.vn.worldlearner.R;
import com.vn.worldlearner.data.models.Answer;
import com.vn.worldlearner.data.models.GlobalSettings;
import com.vn.worldlearner.data.models.ResultAnswer;
```

```
import com.vn.worldlearner.data.models.ResultData;
import com.vn.worldlearner.data.models.User;
import com.vn.worldlearner.data.models.test.TestData;
import com.vn.worldlearner.data.models.test.TestDataUser;
import com.vn.worldlearner.data.models.test.TestMode;
import com.vn.worldlearner.data.models.test.TestModeDataUser;
import com.vn.worldlearner.data.models.test.TestModeType;
import com.vn.worldlearner.data.utils.ModeUtils;
import com.vn.worldlearner.data.utils.TestUtils;
import com.vn.worldlearner.data.utils.TipUtils;
import com.vn.worldlearner.databinding.FragmentTestBinding;
import com.vn.worldlearner.ui.fragment.BaseFragment;
import com.vn.worldlearner.ui.utils.AlertFragmentDialog;
import com.vn.worldlearner.ui.utils.IArrowBackPressed;
import com.vn.worldlearner.ui.utils.IBackPressed;
import com.vn.worldlearner.ui.utils.ICallback;
import com.vn.worldlearner.ui.utils.StringHelper;

import java.util.ArrayList;
import java.util.Locale;
import java.util.Random;

// Init models +
// Init progressbar
// Get testData +
// Shuffle answers +
// Clear positions +
// initVisible question and answers
// Catch start +
// After confirm check answers +
// Give success or fail +
```

```
// Implement correctly ModeTypes
public class FragmentTest extends BaseFragment<FragmentTestBinding> implements IBackPressed,
IArrowBackPressed {
    // Core
    private TestData testData;
    private ResultData resultData;
    private final ArrayList<Integer> answerPositions = new ArrayList<>();
    private final MutableLiveData<TestData> testLiveData = new MutableLiveData<>(null);
    // Models
    private User user;
    private TestMode mode;
    private TestModeDataUser dataUser;
    private GlobalSettings settings;
    private TestModeType modeType;
    // Misc
    private ValueAnimator animator;
    private boolean initProgress;
    private int counterTest = 1;

    public FragmentTest() {
        super(FragmentTestBinding.class);
    }

    @Override
    public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {
        mainViewModel.getMode().observe(getViewLifecycleOwner(), mode -> {
            this.mode = mode;
            mainViewModel.getModeType().observe(getViewLifecycleOwner(), modeType -> {
                this.modeType = modeType;
                mainViewModel.getResultData().setValue(this.resultData = new ResultData());
                mainViewModel.getSettings().observe(getViewLifecycleOwner(), settings -> {
                    this.settings = settings;
                    mainViewModel.getUser().observe(getViewLifecycleOwner(), user -> {
```

```
                this.resultData.setUserBefore(new User(this.user = user));
                this.dataUser = ModeUtils.getUserMode(user, mode);
                this.getNewTest();
                testLiveData.observe(getViewLifecycleOwner(), testData -> {
                   this.testData = testData;
                   if (this.testData != null) {
                      this.initMisc();
                      this.initTimer();
                      this.initQuestion();
                      this.initAnswers();
                      this.startTimer();
                      final long startTime = System.currentTimeMillis();

                      this.setEndCallback(value -> {
                         onCalculate(false, startTime, System.currentTimeMillis());
                         onFail();
                      });
                      binding.confirm.setOnClickListener(v -> {
                         binding.confirm.setEnabled(false);
                         pauseTimer();
                         final long endTime = System.currentTimeMillis();
                         if (checkOnAnswer()) {
                            boolean isCorrect = TestUtils.checkCorrectAnswers(testData, answerPositions) ||
                                  TestUtils.checkCorrectAnswer(testData,
StringHelper.getString(binding.inputEditText.getText()));

                            onCalculate(isCorrect, startTime, endTime);

                            if (isCorrect) {
                               onCorrect();
                            } else {
                               onFail();
                            }
```

```
                        }
                        mainViewModel.getUtils().waitUi(400, () -> {
                            binding.confirm.setEnabled(true);
                        });
                    });
                } else {
                    changeFragment(R.id.fragmentResult);
                }
            });
        });
    });
});
    }

    // Core || CALCULATE -> FAIL/CORRECT
    private void onCalculate(boolean isCorrect, long startTime, long endTime) {
        ResultAnswer resultAnswer = new ResultAnswer();
        resultAnswer.setTestData(testData);
        resultAnswer.setCorrectAnswer(isCorrect);
        resultData.getAnswers().add(resultAnswer);

        int points = (int) (TestUtils.getPoints(startTime, endTime, mode.getLevel()) *
settings.getPointsMultiplier());
        TestDataUser dataUser = TestUtils.getTestDataUser(this.testData, this.dataUser);
        dataUser.setPoints(isCorrect ? points : 0);
        resultAnswer.setPoints(isCorrect ? points : 0);
    }
    private void onFail() {
        resultData.setWrongAnswers(resultData.getWrongAnswers() + 1);
        resultData.setFailStreak(resultData.getFailStreak() + 1);
        user.getStatistic().setTotalWrong(user.getStatistic().getTotalWrong() + 1);
```

```
    blinkColor(binding.question, R.color.red);

    if (resultData.getFailStreak() >= 3) {
        Alerter.create(requireActivity())
                .setDuration(5000)
                .setText(getString(R.string.maybe_need_lower_level))
                .addButton(getString(R.string.to_modes), 0, vClick -> {
                    Alerter.hide();
                    changeFragment(R.id.fragmentHomePage);
                })
                .setIcon(R.drawable.ic_help)
                .show();
    }

    if (modeType.getName().equals("hardcore")) {
        Alerter.create(requireActivity())
                .setIcon(R.drawable.ic_hardcore)
                .setIconColorFilter(ContextCompat.getColor(requireContext(), R.color.red))
                .setText(R.string.type_hardcore_fail)
                .setDismissable(false)
                .show();
        mainViewModel.getUtils().waitUi(1500, () -> {
            changeFragment(R.id.fragmentResult);
        });
        return;
    }

    mainViewModel.getUtils().waitAsync(1000, () -> {
        this.getNewTest();
    });
}
```

```
    private void onCorrect() {
        blinkColor(binding.question, R.color.green);
        resultData.setCorrectAnswers(resultData.getCorrectAnswers() + 1);
        resultData.setCorrectStreak(resultData.getCorrectStreak() + 1);
        user.getStatistic().setTotalCorrect(user.getStatistic().getTotalCorrect() + 1);

        if (resultData.getCorrectStreak() >= 3) {
            resultData.setCorrectStreak(0);
            int tipCount = (int) (1 * settings.getTipsMultiplier());
            TipUtils.addTips(user, tipCount);
            Snackbar.make(requireView(), String.format("%s %s %s", getString(R.string.you_get_tip), tipCount,
StringHelper.getTipsWord(requireContext(), tipCount)), Snackbar.LENGTH_SHORT).show();
        }

        resultData.setFailStreak(resultData.getFailStreak() - (int) (1 + Math.random() * 1));
        if (resultData.getFailStreak() < 0) {
            resultData.setFailStreak(0);
        }

        mainViewModel.getUtils().waitAsync(1000, () -> {
            this.getNewTest();
        });
    }

    private void getNewTest() {
        ui(() -> {
            this.testLiveData.setValue(this.getTestData());
        });
    }


    private TestData getTestData() {
        if (counterTest-- < 1) {
```

```
            return null;
        }
        TestData data = TestUtils.getNotDoneTest(this.mode, this.dataUser);
        if (data != null) {
            if (data.equals(this.testData) && this.mode.getTests().size() !=
this.dataUser.getTestDataUsers().size()) {
                System.out.println("Question is equal trying choose other");
                data = getTestData();
            }
            if (data != null) {
                counterTest = 1;
                return data.clone();
            }
        }
        counterTest = 1;
        return null;
    }

    private boolean checkOnAnswer() {
        boolean flag = true;
        switch (this.testData.getTestType()) {
            case CHOOSE:
            case IMAGE:
                if (answerPositions.isEmpty()) {
                    flag = false;
                    Alerter.create(requireActivity()).setText(getString(R.string.answer_cant_be_empty)).show();
                }
                break;
            case INPUT:
                if (StringHelper.getString(binding.inputEditText.getText()).isEmpty()) {
                    flag = false;
                    Alerter.create(requireActivity()).setText(getString(R.string.answer_cant_be_empty)).show();
                }
```

```
            break;
        case MULTIPLY:
            if (answerPositions.size() <= 1) {
                flag = false;
                Alerter.create(requireActivity()).setText(getString(R.string.answer_need_more)).show();
            }
            break;
    }
    return flag;
}

// progressBar etc.
private void initMisc() {
    this.answerPositions.clear();
    binding.progress.setMax(mode.getTests().size());
    binding.question.setTextColor(ContextCompat.getColor(requireContext(), R.color.black));

    if (modeType.getName().equals("default")) {
        int size = ModeUtils.getUserMode(user, mode).getTestDataUsers().size();
        binding.progress.setProgressCompat(size, initProgress);
        initProgress = true;
    }
}

// Set question by modeType
private void initQuestion() {
    switch (testData.getTestType()) {
        case IMAGE:
            binding.itemsRV.setVisibility(View.VISIBLE);
            binding.inputLayout.setVisibility(View.GONE);
            binding.questionImage.setVisibility(View.VISIBLE);
            if (testData.getUrlImage() != null) {
```

```
            Picasso.get().load(testData.getUrlImage()).into(binding.questionImage);
        }
        break;
    case CHOOSE:
        binding.itemsRV.setVisibility(View.VISIBLE);
        binding.inputLayout.setVisibility(View.GONE);
        binding.questionImage.setVisibility(View.GONE);
        break;
    case INPUT:
        binding.inputLayout.setVisibility(View.VISIBLE);
        binding.itemsRV.setVisibility(View.GONE);
        binding.questionImage.setVisibility(View.GONE);
        break;
}

binding.question.setText(testData.getQuestion());
String type = "UNDEFINED TYPE";
switch (testData.getTestType()) {
    case CHOOSE:
        type = getString(R.string.type_choose);
        break;
    case MULTIPLY:
        type = getString(R.string.type_multiply);
        break;
    case INPUT:
        type = getString(R.string.type_input);
        break;
    case IMAGE:
        type = getString(R.string.type_image);
        break;
}
binding.textType.setText(type);
```

```
}

// Show list answers or input field
private void initAnswers() {
    AnswerAdapter adapter = new AnswerAdapter();
    adapter.setCallback(position -> {
        if (answerPositions.contains(position)) {
            answerPositions.remove(position);
        } else {
            answerPositions.add(position);
        }
    });
    binding.itemsRV.setLayoutManager(new LinearLayoutManager(requireContext()));
    binding.itemsRV.setAdapter(adapter);
    adapter.setItems(testData.getAnswers());
}

// Start timer by modeType
private void initTimer() {
    if (animator == null) {
        if (modeType.getName().equals("time")) {
            animator = ValueAnimator.ofInt(settings.getTypeTimeTest(), 0);
            animator.setDuration(settings.getTypeTimeTest() * 1000L);
        } else if (modeType.getName().equals("max_time")) {
            animator = ValueAnimator.ofInt(settings.getTypeTimeMode(), 0);
            animator.setDuration(settings.getTypeTimeMode() * 1000L);
        }
    }
}

private void setEndCallback(ICallback<Boolean> callback) {
    if (animator != null) {
```

```
        animator.addListener(new AnimatorListenerAdapter() {
            @Override
            public void onAnimationEnd(Animator animation) {
                super.onAnimationEnd(animation);
                callback.call(true);
            }
        });
    }
}

private void startTimer() {
    if (animator != null) {
        animator.removeAllListeners();
        animator.addUpdateListener(animation -> {
            int value = (int) animation.getAnimatedValue();
            binding.progress.setProgressCompat(value, true);
        });

        animator.start();
    }
}

private void endTimer() {
    if (animator != null) {
        animator.removeAllListeners();
        animator.end();
    }
}

private void resumeTimer() {
    if (animator != null) {
        animator.resume();
```

```
    }
}

private void pauseTimer() {
    if (animator != null) {
        animator.pause();
    }
}

// Enable menu tips
private void initTips(boolean enabled) {
    AppBarLayout layout = requireActivity().findViewById(R.id.appBarLayout);
    User user = mainViewModel.getUser().getValue();
    if (layout != null) {
        CircleImageView tips = layout.findViewById(R.id.tips);
        tips.setVisibility(enabled ? View.VISIBLE : View.GONE);
        ViewCompat.setTooltipText(tips, getString(R.string.tips));
        if (enabled) {
            PopupMenu menu = new PopupMenu(requireContext(), tips);
            menu.getMenu().add(Menu.NONE, -1, 0, getString(R.string.tips)).setEnabled(false);
            if (testData != null) {
                switch (testData.getTestType()) {
                    case CHOOSE:
                    case MULTIPLY:
                    case IMAGE:
                        menu.getMenu().add(0, 0, 0, R.string.tip_remove_half);
                        menu.getMenu().add(0, 1, 0, R.string.tip_remove_one);
                        break;
                    case INPUT:
                        menu.getMenu().add(0, 2, 0, R.string.tip_half_answer);
                        menu.getMenu().add(0, 3, 0, R.string.tip_one_letter);
                        break;
```

```
}
tips.setOnClickListener(v -> menu.show());

menu.setOnMenuItemClickListener(item -> {
   if (item.getItemId() == 0) {
      int tipCost = 4;
      String fText = formatTips(user.getTips(), tipCost);
      new AlertFragmentDialog.Builder()
            .withTitle(getString(R.string.use_tip) + "?")
            .withMessage(fText + getString(R.string.tip_description_remove_half))
            .withYes(getString(R.string.confirm))
            .withActionYes(() -> {
               if (TipUtils.hasTips(user, tipCost)) {
                  ArrayList<Answer> answers = testData.getAnswers();
                  if (answers != null) {
                     if (answers.size() > 2) {
                        int countToRemove = answers.size() / 2;
                        while (countToRemove > 0) {
                           Answer answer = answers.get(new Random().nextInt(answers.size()));
                           if (!answer.isCorrect()) {
                              countToRemove--;
                              answers.remove(answer);
                           }
                        }
                        initAnswers();
                        TipUtils.drawTips(user, tipCost);
                     }
                  }
               }
            }).build().show(getChildFragmentManager(), this.getTag());
   }
   if (item.getItemId() == 1) {
```

```
    int tipCost = 1;
    String fText = formatTips(user.getTips(), tipCost);
    new AlertFragmentDialog.Builder()
            .withTitle(getString(R.string.use_tip) + "?")
            .withMessage(fText + getString(R.string.tip_description_remove_one))
            .withYes(getString(R.string.confirm))
            .withActionYes(() -> {
                if (TipUtils.hasTips(user, tipCost)) {
                    ArrayList<Answer> answers = testData.getAnswers();
                    if (answers != null) {
                        if (answers.size() > 1) {
                            while (true) {
                                Answer answer = answers.get(new Random().nextInt(answers.size()));
                                if (!answer.isCorrect()) {
                                    answers.remove(answer);
                                    break;
                                }
                            }
                            initAnswers();
                            TipUtils.drawTips(user, tipCost);
                        }
                    }
                }
            }).build().show(getChildFragmentManager(), this.getTag());
    }
    if (item.getItemId() == 2) {
        if (TipUtils.hasTips(user, 1)) {
            // TODO: 16.09.2023
            int tipCost = 4;
            String fText = formatTips(user.getTips(), tipCost);
            new AlertFragmentDialog.Builder()
                    .withTitle(getString(R.string.use_tip) + "?")
```

```
            .withMessage(fText + getString(R.string.tip_description_remove_half))
            .withYes(getString(R.string.confirm))
            .withActionYes(() -> {
              if (TipUtils.hasTips(user, tipCost)) {
                ArrayList<Answer> answers = testData.getAnswers();
                if (answers != null) {


                }
              }
            }).build().show(getChildFragmentManager(), this.getTag());
        TipUtils.drawTips(user, 1);
    }
}
if (item.getItemId() == 3) {
    if (TipUtils.hasTips(user, 1)) {
        // TODO: 16.09.2023
        int tipCost = 4;
        String fText = formatTips(user.getTips(), tipCost);
        new AlertFragmentDialog.Builder()
            .withTitle(getString(R.string.use_tip) + "?")
            .withMessage(fText + getString(R.string.tip_description_remove_half))
            .withYes(getString(R.string.confirm))
            .withActionYes(() -> {
              if (TipUtils.hasTips(user, tipCost)) {
                ArrayList<Answer> answers = testData.getAnswers();
                if (answers != null) {


                }
              }
            }).build().show(getChildFragmentManager(), this.getTag());
        TipUtils.drawTips(user, 1);
    }
```

```
                }
                initAnswers();
                return true;
            });
        }
    }
}

// Utils
private String formatTips(int tipsCount, int tipsCost) {
    String tipsWord = StringHelper.getTipsWord(requireContext(), tipsCount);
    String tipsWordCost = StringHelper.getTipsWord(requireContext(), tipsCost);
    String fText = String.format(Locale.getDefault(), "%s: %d %s\n", getString(R.string.you_have),
tipsCount, tipsWord);
    fText += String.format("%s: %s %s\n\n", getString(R.string.use_cost), tipsCost, tipsWordCost);
    return fText;
}

private void blinkColor(TextView textView, @ColorRes int color) {
    ObjectAnimator colorAnimator = ObjectAnimator.ofInt(textView, "textColor",
textView.getCurrentTextColor(), ContextCompat.getColor(requireContext(), color));
    colorAnimator.setEvaluator(new ArgbEvaluator());
    colorAnimator.setDuration(200);
    colorAnimator.start();
}

@Override
public void onResume() {
    initTips(true);
    resumeTimer();
    super.onResume();
}
```

```
    @Override
    public void onPause() {
        pauseTimer();
        initTips(false);
        super.onPause();
    }

    private void showBackDialog() {
        new AlertFragmentDialog.Builder()
                .withTitle(getString(R.string.confirm) + "?")
                .withMessage(getString(R.string.end_test) + "?")
                .withYes(getString(R.string.confirm))
                .withNo(getString(R.string.cancel))
                .withNeutral(getString(R.string.to_modes))
                .withActionYes(() -> changeFragment(R.id.fragmentResult))
                .withActionNeutral(() -> {
                    mainViewModel.getUser().setValue(resultData.getUserBefore());
                    changeFragment(R.id.fragmentHomePage);
                })
                .build()
                .show(getChildFragmentManager(), getTag());
    }

    @Override
    public boolean onBackPressed() {
        showBackDialog();
        return true;
    }
}
public class AnswerAdapter extends BaseAdapter<Answer> {
    private int defaultColor = 0;
```

```
    private ICallback<Integer> callback;

    public void setCallback(ICallback<Integer> callback) {
        this.callback = callback;
    }

    @NonNull
    @Override
    public BaseHolder<Answer> onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
        return createHolder(parent, R.layout.item_answer, AnswerHolder.class);
    }

    @Override
    public void onBindViewHolder(@NonNull BaseHolder<Answer> holder, int position) {
        super.onBindViewHolder(holder, position);
        if (holder instanceof AnswerHolder) {
            AnswerHolder answerHolder = (AnswerHolder) holder;
            holder.itemView.setOnClickListener(v -> {
                callback.call(position);
                answerHolder.chosen = !answerHolder.chosen;
                int mainColor = ContextCompat.getColor(answerHolder.itemView.getContext(),
R.color.main_color);
                if (defaultColor == 0) {
                    defaultColor = answerHolder.binding.cardAnswer.getStrokeColorStateList().getDefaultColor();
                }
                answerHolder.binding.cardAnswer.setStrokeColor(answerHolder.chosen ? mainColor :
defaultColor);
            });
        }
    }

    public static class AnswerHolder extends BaseHolder<Answer> {
        private final ItemAnswerBinding binding;
```

```
        boolean chosen;

        public AnswerHolder(@NonNull View view) {
            super(view);
            binding = ItemAnswerBinding.bind(view);
        }

        @Override
        public void bind(Answer answer) {
            binding.text.setText(answer.getAnswer());
        }
    }
}
public class FragmentMode extends BaseMenuFragment<FragmentModeBinding> {
    private final MutableLiveData<Boolean> statisticVisible = new MutableLiveData<>(false);
    private int doneTests = 0;

    public FragmentMode() {
        super(FragmentModeBinding.class);
    }

    @Override
    public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {
        mainViewModel.getMode().observe(getViewLifecycleOwner(), mode -> {
            binding.textMode.setText(mode.getName());
            String fLevel = String.format("%s: %s", getString(R.string.level), mode.getLevel());
            binding.textLevel.setText(fLevel);
            binding.textDescription.setText(mode.getDescription());
            mainViewModel.getUser().observe(getViewLifecycleOwner(), user -> {
                if (user != null) {
                    TestModeDataUser modeDataUser = ModeUtils.getUserMode(user, mode);
                    if (modeDataUser != null) {
```

```
                doneTests = ModeUtils.countModeDoneTests(modeDataUser);
            }
            String fModeInfo = String.format("%s: %s/%s", getString(R.string.done), doneTests,
mode.getTests().size());
            binding.textModeInfo.setText(fModeInfo);
        }
        statisticVisible.observe(getViewLifecycleOwner(), statisticValue -> {
            if (statisticValue) {
                binding.statistics.setText(getString(R.string.questions));
                binding.textAvailableTest.setText(getString(R.string.current_statistic_mode));
                UserAdapter adapter = new UserAdapter();
                adapter.setSettings(mainViewModel.getSettings().getValue());
                adapter.setMode(mode);
                adapter.setCallback(userCallback -> {
                    if (user != null) {
                        if (UserUtils.isInputtedAddInfo(user) || user.isAdmin()) {
                            DialogUserInfo userInfo = new DialogUserInfo();
                            userInfo.setUser(userCallback);
                            userInfo.show(getChildFragmentManager(), getTag());
                        } else {

Alerter.create(requireActivity()).setIcon(R.drawable.ic_secure).setIconColorFilter(ContextCompat.getColor(
requireContext(), R.color.red)).setText(R.string.need_input_information).show();
                        }
                    }
                });
                binding.itemsRV.setAdapter(adapter);
                binding.itemsRV.setLayoutManager(new LinearLayoutManager(requireContext()));
                ArrayList<User> usersInMode = new ArrayList<>();
                ArrayList<User> users = mainViewModel.getUsers().getValue();
                if (users != null) {
                    for (int i = 0; i < users.size(); i++) {
                        User userI = users.get(i);
```

```
                    if (ModeUtils.hasTestModeUser(userI, mode) && !ModeUtils.isEmptyMode(userI, mode))
{
                        usersInMode.add(userI);
                    }
                }
                Collections.sort(usersInMode, (user1, user2) -> {
                    int u1 = ModeUtils.getTotalPointsMode(ModeUtils.getUserMode(user1, mode));
                    int u2 = ModeUtils.getTotalPointsMode(ModeUtils.getUserMode(user2, mode));
                    return Integer.compare(u2, u1);
                });
                adapter.setItems(usersInMode);
            }


        } else {
            binding.statistics.setText(getString(R.string.statistics));
            binding.textAvailableTest.setText(getString(R.string.available_tests));
            QuestionAdapter adapter = new QuestionAdapter(mode, user);
            binding.itemsRV.setAdapter(adapter);
            binding.itemsRV.setLayoutManager(new LinearLayoutManager(requireContext()));
            adapter.setItems(mode.getTests());
        }
    });
});

this.setMenuProvider(new MenuProvider() {
    @Override
    public void onCreateMenu(@NonNull Menu menu, @NonNull MenuInflater menuInflater) {
        menu.add(getString(R.string.reset_mode));
    }

    @Override
    public boolean onMenuItemSelected(@NonNull MenuItem menuItem) {
```

```
            if (menuItem.getItemId() == 0) {
                AlertFragmentDialog fragmentDialog = new
AlertFragmentDialog(getString(R.string.reset_mode) + "?", getString(R.string.reset_mode_text),
getString(R.string.confirm), getString(R.string.cancel), null, null, null, () -> {
                    User user = mainViewModel.getUser().getValue();
                    if (user != null) {
                        if (user.getTestModeDataUsers().remove(ModeUtils.getUserMode(user, mode))) {
                            mainViewModel.getDataNetHandler().updateUser(user);
                            statisticVisible.setValue(false);
                            Snackbar.make(view, getString(R.string.success_reset),
Snackbar.LENGTH_SHORT).show();
                        }
                    }
                });
                fragmentDialog.show(getChildFragmentManager(), fragmentDialog.getTag());
            }
            return false;
        }
    });

    binding.statistics.setOnClickListener(v -> {
        statisticVisible.postValue(Boolean.FALSE.equals(statisticVisible.getValue()));
    });

    binding.modeStart.setOnClickListener(v -> {
        if (doneTests == mode.getTests().size()) {
            Snackbar.make(view, getString(R.string.mode_is_finish_restart),
Snackbar.LENGTH_SHORT).show();
            return;
        }
        DialogSelectMode dialogSelectMode = new DialogSelectMode();
        dialogSelectMode.show(getChildFragmentManager(), getTag());
        dialogSelectMode.setCallback(selected -> {
            mainViewModel.getModeType().observe(getViewLifecycleOwner(), modeType -> {
```

```
                changeFragment(R.id.fragmentTest);
            });
        });
    });
});

}
} public class UserAdapter extends BaseAdapter<User> {
    private GlobalSettings settings;
    private ICallback<User> callback;
    private TestMode mode;

    public void setCallback(ICallback<User> callback) {
        this.callback = callback;
    }

    public void setMode(TestMode mode) {
        this.mode = mode;
    }

    public void setSettings(GlobalSettings settings) {
        this.settings = settings;
    }

    @NonNull
    @Override
    public BaseHolder<User> onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
        return createHolder(parent, R.layout.item_user, UserHolder.class);
    }

    @Override
    public void onBindViewHolder(@NonNull BaseHolder<User> holder, int position) {
```

```
        super.onBindViewHolder(holder, position);

        UserHolder userHolder = (UserHolder) holder;

        User user = getItems().get(position);

        int userPoints = ModeUtils.getTotalPointsUser(user);

        userHolder.binding.textName.setText(user.getUsername());

        String fLevel = String.format("%s: %s", userHolder.itemView.getContext().getString(R.string.level),
UserUtils.getLevel(userPoints, settings));

        userHolder.binding.textLevel.setText(fLevel);

        String fPoints = String.format("%s: %s",
userHolder.itemView.getContext().getString(R.string.totalPoints),
ModeUtils.getTotalPointsMode(ModeUtils.getUserMode(user, mode)));

        userHolder.binding.textPoints.setText(fPoints);


        userHolder.itemView.setOnLongClickListener(v -> {

            callback.call(user);

            return true;

        });

    }


    public static class UserHolder extends BaseHolder<User> {

        private final ItemUserBinding binding;


        public UserHolder(@NonNull View view) {

            super(view);

            binding = ItemUserBinding.bind(view);

        }


        @Override

        public void bind(User user) {

            int id = itemView.getContext().getResources().getIdentifier("a" + user.getIcon(), "drawable",
itemView.getContext().getPackageName());

            binding.avatar.itemImage.setImageDrawable(ContextCompat.getDrawable(itemView.getContext(),
id));

        }
```

```
    }

} public class DialogSelectMode extends BottomSheetDialogFragment {
    private DialogSelectModeBinding binding;
    private MainViewModel mainViewModel;
    private int selectedPosition;

    private ICallback<Boolean> callback;
    private boolean selectedMode;

    public DialogSelectMode() {
        super(R.layout.dialog_select_mode);
    }

    public void setCallback(ICallback<Boolean> callback) {
        this.callback = callback;
    }

    @Override
    public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {
        super.onViewCreated(view, savedInstanceState);
        binding = DialogSelectModeBinding.bind(view);
        mainViewModel = new ViewModelProvider(requireActivity()).get(MainViewModel.class);

        mainViewModel.getSettings().observe(getViewLifecycleOwner(), settings -> {
            ArrayList<TestModeType> types = settings.getModeTypes();
            if (types != null && !types.isEmpty()) {
                ModeAdapter adapter = new ModeAdapter();
                binding.itemsRV.setAdapter(adapter);
                binding.itemsRV.setLayoutManager(new GridLayoutManager(requireContext(), 2));
                adapter.setItems(types);
                adapter.setCallback(selectedPosition -> {
```

```
            for (int i = 0; i < adapter.getItems().size(); i++) {
                ModeAdapter.ModeTypeHolder holder = (ModeAdapter.ModeTypeHolder)
this.binding.itemsRV.findViewHolderForAdapterPosition(i);
                if (holder != null) {
                    if (i == selectedPosition) this.selectedPosition = selectedPosition;
                    holder.update(i == selectedPosition);
                }
            }
            this.selectedMode = true;
        });

        binding.help.setOnClickListener(v -> {
            DialogSelectModeHelp dialogSelectModeHelp = new DialogSelectModeHelp();
            dialogSelectModeHelp.show(getChildFragmentManager(), getTag());
        });

        binding.confirm.setOnClickListener(v -> {
            if (!selectedMode) {
                Alerter.create(requireDialog()).setText(R.string.select_mode).show();
            } else {
                mainViewModel.getModeType().setValue(types.get(selectedPosition));

                UserStatistic statistic = mainViewModel.getUser().getValue().getStatistic();
                UserUtils.incrementValue(statistic.getTotalModeType(), types.get(selectedPosition).getId());
                UserUtils.incrementValue(statistic.getTotalModeStarted(),
mainViewModel.getMode().getValue().getId());

                dismiss();
                callback.call(selectedMode);
            }
        });
    }
});
```

```
    }

}
public class FragmentHome extends BaseMenuFragment<FragmentHomePageBinding> {
    private static boolean updatingDaily = true;

    public FragmentHome() {
        super(FragmentHomePageBinding.class);
    }

    @Override
    public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {
        mainViewModel.getSettings().observe(getViewLifecycleOwner(), settings -> {
            mainViewModel.getUser().observe(getViewLifecycleOwner(), user -> {
                // Check if user deleted or banned
                if (user == null) {
                    String fText = String.format("%s: %s", getString(R.string.reason),
getString(R.string.reason_cheating));
                    fText += "\n" + getString(R.string.application_will_be_closed);
                    Alerter.create(requireActivity()).setTitle(R.string.your_acc_was_deleted).setText(fText).show();
                    mainViewModel.getUtils().waitUi(5000, () -> {
                        System.exit(0);
                    });
                    return;
                }
                // Daily
                if (updatingDaily) {
                    this.checkDailyReward(user, userCallback -> showDailyDialog(true));
                    updatingDaily = false;
                }
                // Init
                binding.userName.setText(user.getUsername());
                int userPoints = ModeUtils.getTotalPointsUser(user);
```

```
        int userLevel = UserUtils.getLevel(userPoints, settings);
        String fLevel = String.format("%s: %s", getString(R.string.level), userLevel);
        binding.userLevel.setText(fLevel);
        String fPoints = String.format("%s: %s", getString(R.string.totalPoints), userPoints);
        binding.userTotalPoints.setText(fPoints);
        int idAvatar = requireActivity().getResources().getIdentifier("a" + user.getIcon(), "drawable",
requireContext().getPackageName());
        binding.userImage.setImageResource(idAvatar);
        if (user.isAdmin()) {
            binding.buttonAdmin.setVisibility(View.VISIBLE);
        }
        DateFormat format = SimpleDateFormat.getTimeInstance();
        user.setLastLoginTime(format.format(new Date()));
        binding.buttonMore.setOnClickListener(v -> {
            PopupMenu menu = new PopupMenu(requireContext(), binding.buttonMore);
            menu.getMenu().add(0, 2, 0, R.string.daily_rewards);
            menu.getMenu().add(0, 0, 0, R.string.choose_your_avatar);
            menu.getMenu().add(0, 1, 0, R.string.add_more_about_yourself);
            menu.show();

            menu.setOnMenuItemClickListener(item -> {
                if (item.getItemId() == 0) {
                    AvatarView avatarView = new AvatarView(requireContext());
                    AlertDialog alertDialog = new AlertDialog.Builder(requireContext())
                            .setPositiveButton(getString(R.string.confirm), (dialog, which) -> {
                                user.setIcon(avatarView.getSelectedIcon());
                                mainViewModel.getDataNetHandler().updateUser(user);
                            })
                            .setView(avatarView)
                            .create();
                    alertDialog.show();
                } else if (item.getItemId() == 1) {
                    changeFragment(R.id.fragmentAddInformation);
```

```
                } else if (item.getItemId() == 2) {
                    showDailyDialog(false);
                }
                return true;
            });
        });

        binding.buttonAdmin.setOnClickListener(v -> {
            PopupMenu menu = new PopupMenu(requireContext(), binding.buttonMore);
            menu.getMenu().add(0, 0, 0, R.string.admin_graphics);
            String enableDisable = !settings.isDebugUsers() ? getString(R.string.enable) :
getString(R.string.disable);
            menu.getMenu().add(0, 1, 0, String.format("%s %s", enableDisable,
getString(R.string.debugUsers)));
            menu.setOnMenuItemClickListener(item -> {
                if (item.getItemId() == 0) {
                    changeFragment(R.id.fragmentAdminGraphics);
                }
                if (item.getItemId() == 1) {
                    settings.setDebugUsers(!settings.isDebugUsers());
                }
                return true;
            });

            menu.show();
        });
    });
    mainViewModel.getModes().observe(getViewLifecycleOwner(), modes -> {
        ModeAdapter modeAdapter = new ModeAdapter();
        binding.itemsRV.setAdapter(modeAdapter);
        binding.itemsRV.setLayoutManager(new LinearLayoutManager(requireContext()));
        modeAdapter.setPositionCallback(position -> {
```

```
            mainViewModel.getMode().postValue(modes.get(position));
            mainViewModel.getTestData().postValue(null);
            changeFragment(R.id.fragmentMode);
        });
        modeAdapter.setItems(modes);
    });
    // Menu | Reset all modes
    super.setMenuProvider(new MenuProvider() {
        @Override
        public void onCreateMenu(@NonNull Menu menu, @NonNull MenuInflater menuInflater) {
            menu.add(0, 1, 0, getString(R.string.reset_all_modes));
        }

        @Override
        public boolean onMenuItemSelected(@NonNull MenuItem menuItem) {
            if (menuItem.getItemId() == 1) {
                AlertFragmentDialog fragmentDialog = new
AlertFragmentDialog(getString(R.string.reset_all_modes) + "?", getString(R.string.reset_all_modes_text),
getString(R.string.confirm), getString(R.string.cancel), null, null, null, () -> {
                    User user = mainViewModel.getUser().getValue();
                    if (user != null) {
                        user.getTestModeDataUsers().clear();
                        mainViewModel.getDataNetHandler().updateUser(user);
                        Snackbar.make(view, getString(R.string.success_reset), Snackbar.LENGTH_SHORT).show();
                    }
                });
                fragmentDialog.show(getChildFragmentManager(), fragmentDialog.getTag());
            }
            return true;
        }
    });
}
```

```
private void checkDailyReward(User user, ICallback<User> callback) {
    SimpleDateFormat dateFormat = new SimpleDateFormat("dd.MM.yyyy", Locale.getDefault());
    long currentTimeMillis = System.currentTimeMillis();
    Calendar calendar = Calendar.getInstance();
    calendar.setTimeInMillis(currentTimeMillis);
    String currentDate = dateFormat.format(calendar.getTime());
    calendar.add(Calendar.DAY_OF_MONTH, 1);
    String nextDate = dateFormat.format(calendar.getTime());

    if (user.getNextDateReward() == null) {
        // First time
        user.setDays(0);
        user.setLastLogin(currentDate);
        callback.call(user);
    } else if (user.getNextDateReward().equals(currentDate)) {
        // Other times
        user.setDays(user.getDays() + 1);
        user.setLastLogin(currentDate);
        callback.call(user);
    }

    if (user.getLastLogin() == null || !user.getLastLogin().equals(currentDate)) {
        user.setDays(0);
        user.setLastLogin(currentDate);
    }

    user.setNextDateReward(nextDate);
}
private void showDailyDialog(boolean canGive) {
    DialogDailyReward dialogDailyReward = new DialogDailyReward(canGive);
    dialogDailyReward.show(getChildFragmentManager(), dialogDailyReward.getTag());
}
```

Додаток Д – Ілюстративна частина

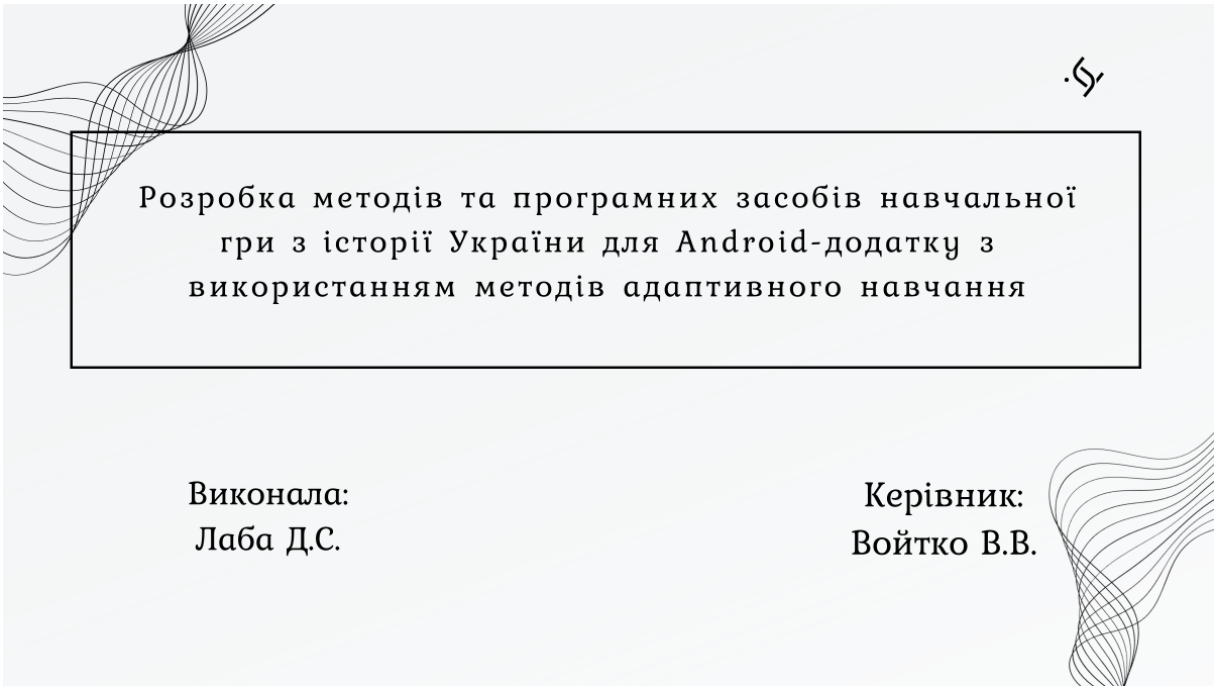

Рисунок Г.1 – Назва роботи

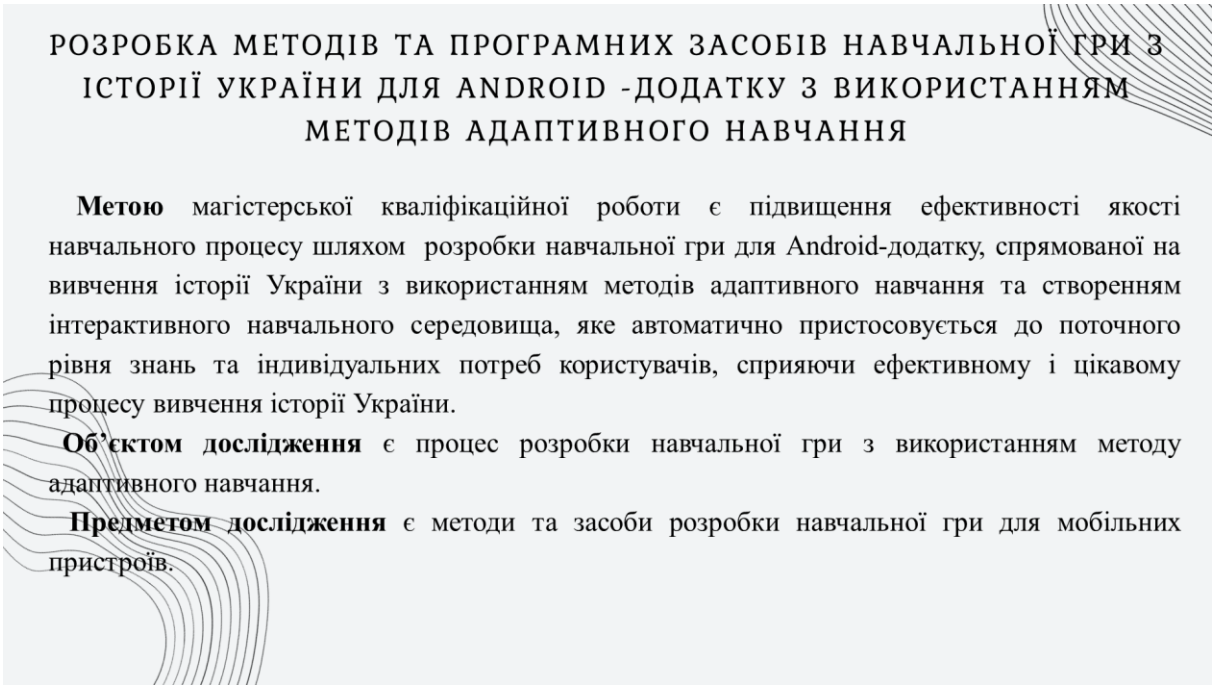

Рисунок Г.2 – Мета, об'єкт і предмет дослідження

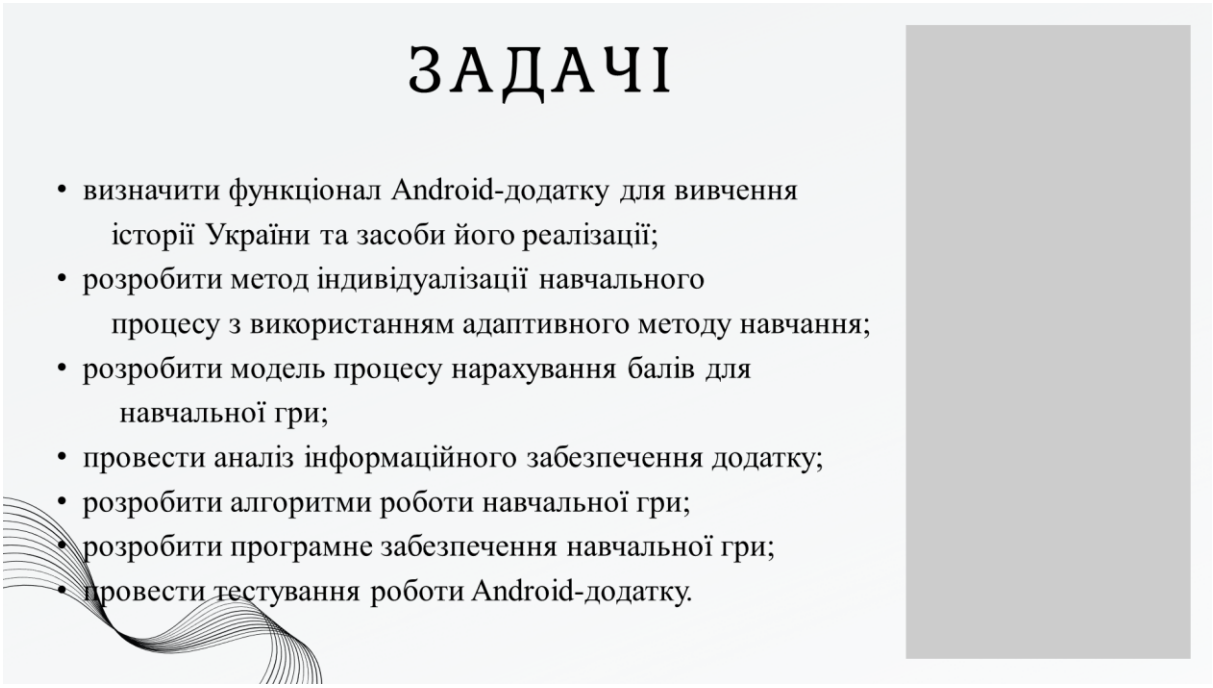

Рисунок Г.3 – Задачі

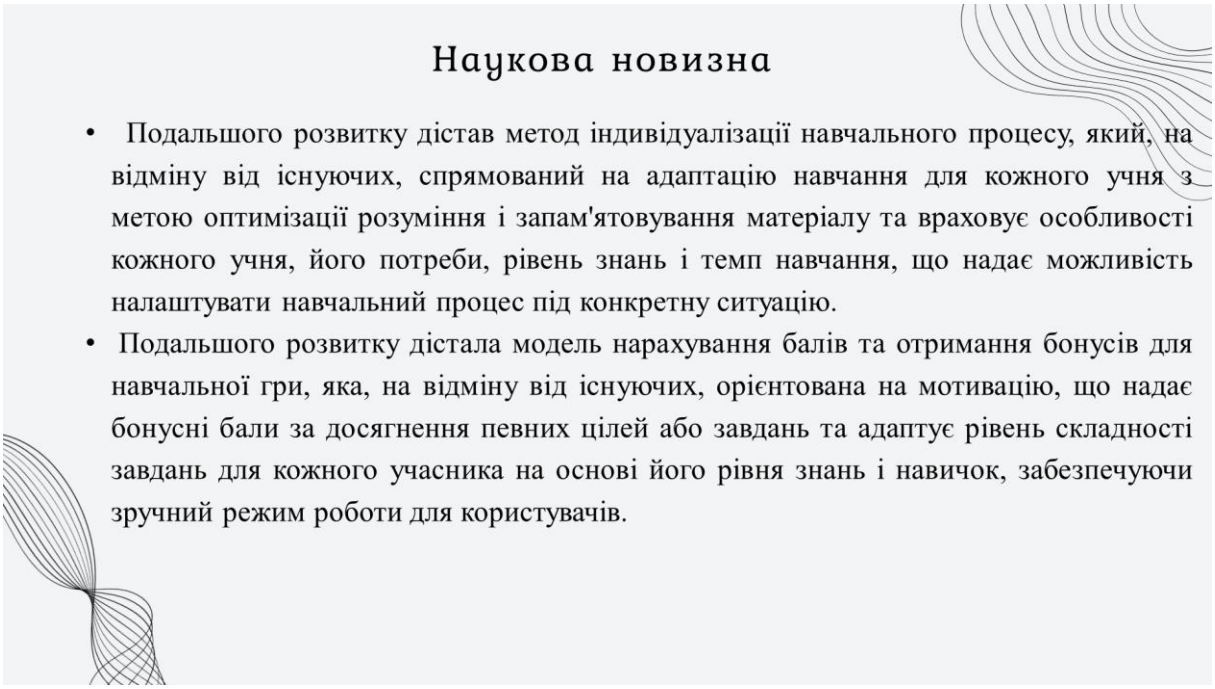

Рисунок Г.4 – Наукова новизна

Практична цінність

- **Практична цінність** результатів магістерської кваліфікаційної роботи полягає у здатності практично застосовувати розроблений Android-додаток як навчальну гру для ефективного вивчення історії України.

Рисунок Г.5 – Практична цінність

ПОРІВНЯННЯ АНАЛОГІВ

Критерії	ЗНО 2024. Історія України	Історія України	ЗНО 2023: Історія України	Власна реалізація
1	2	3	4	5
Реєстрація та авторизація	+	-	-	+
Різновидність режимів навчання	+	+	+	+
Наявність адаптивного методу навчання	-	-	-	+
Наявність онлайн бази даних	-	-	+	+
Безкоштовна ліцензія	+-	+	+-	+
Інтерфейс українською мовою	+	+	+	+
Підсумковий результат	3,5	3	3,5	6

Рисунок Г.6 – Порівняння аналогів

Рисунок Г.7 – Загальна модель роботи програми

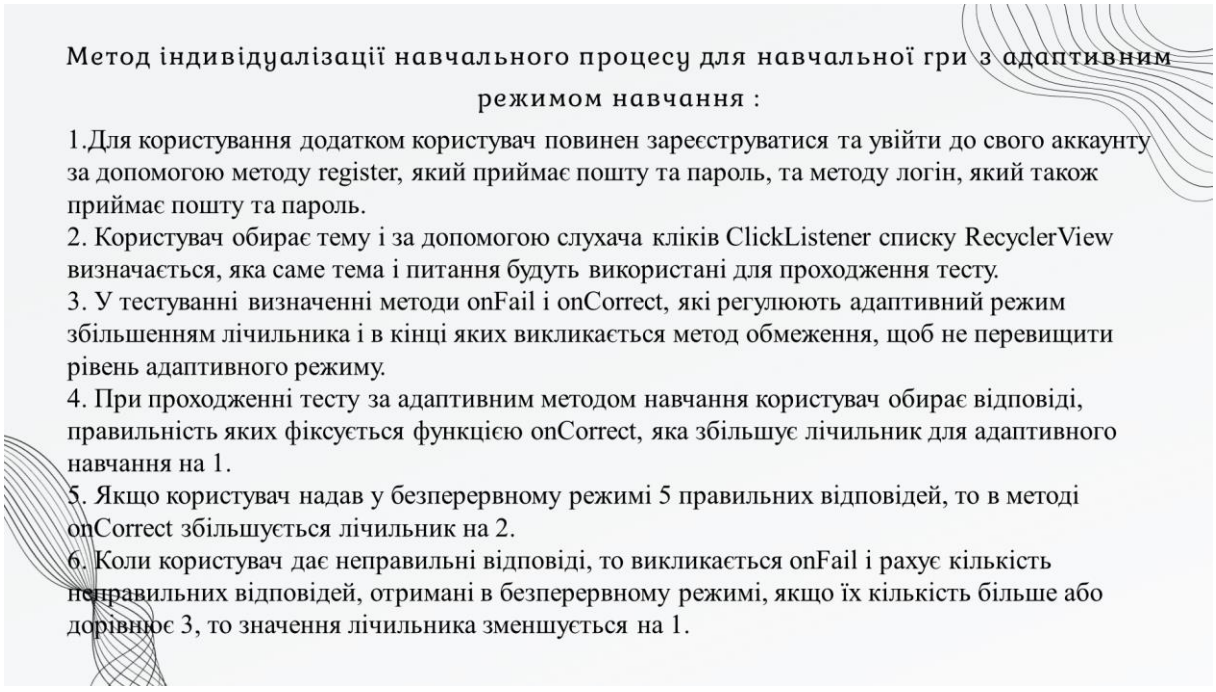

Рисунок Г.8 – Метод індивідуалізації навчального процесу з адаптивним методом навчання

Рисунок Г.9 – Блок-схема алгоритму роботи методу індивідуалізації навчального процесу в режимі адаптивного навчання

Рисунок Г.10 – Модель навчальної системи

Рисунок Г.11 – Блок-схеми алгоритмів реєстрації та авторизації

Рисунок Г.12 –Узагальнена блок-схема алгоритму тестування

Рисунок Г.13 – Модель нарахування балів та отримання бонусів для навчальної гри

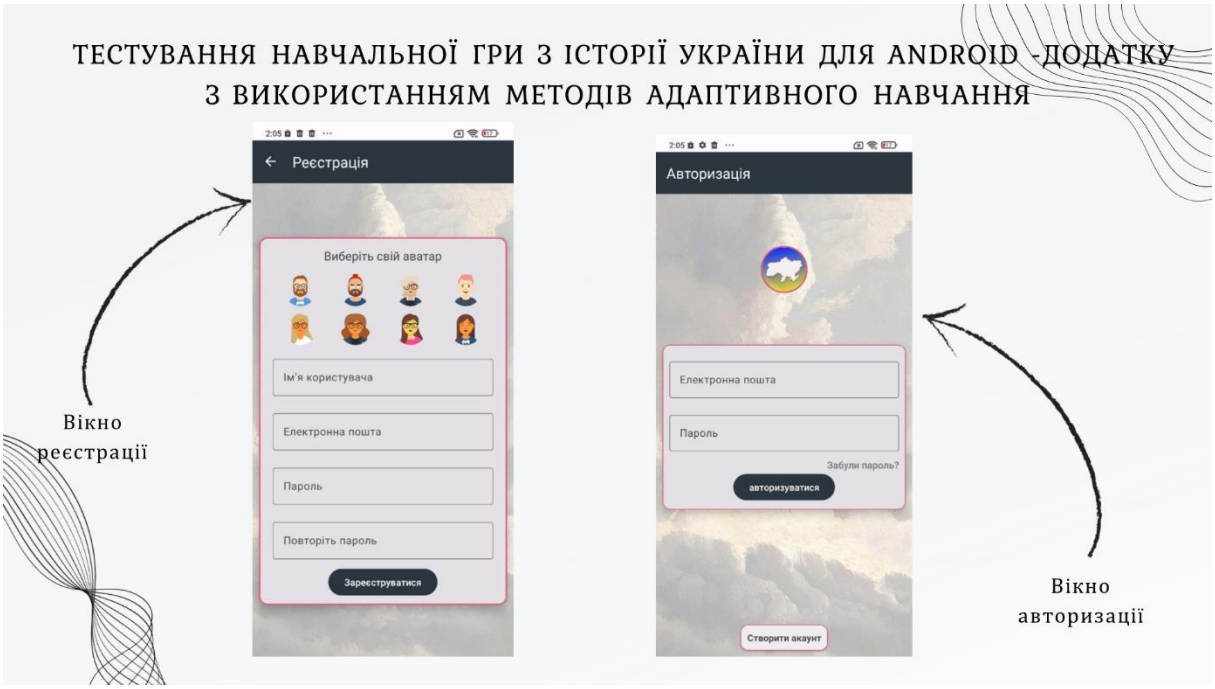

Рисунок Г.14 – Вікна реєстрації та авторизації

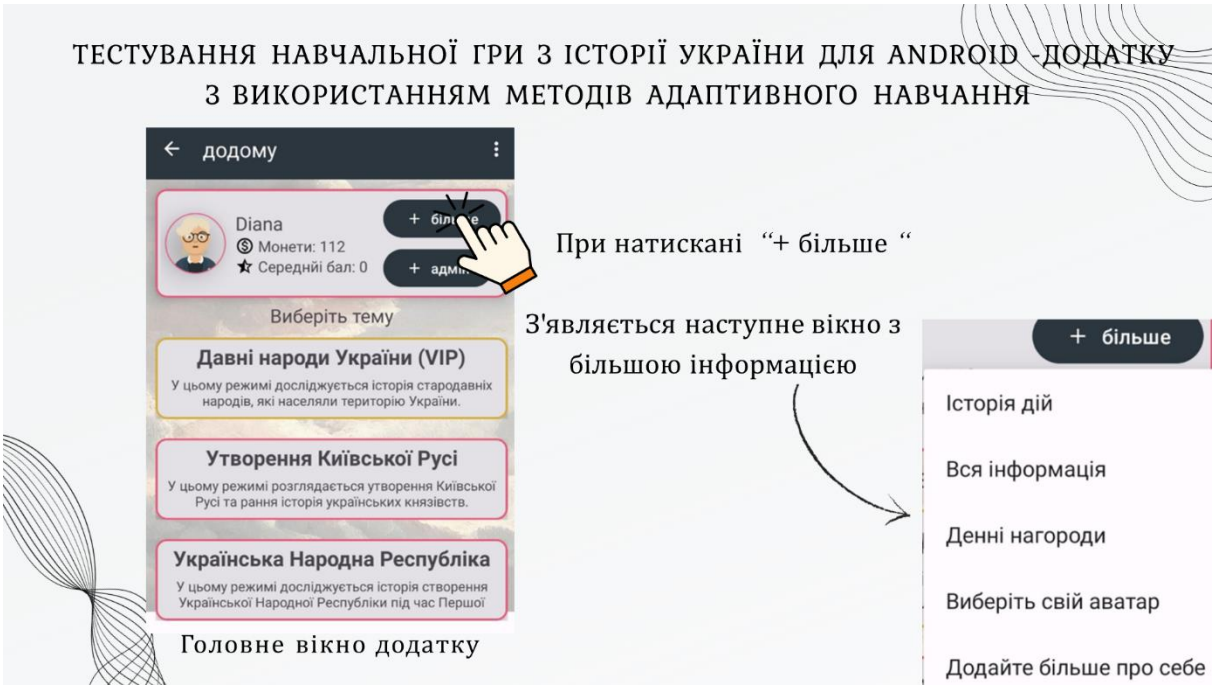

Рисунок Г.15 – Головне вікно та меню кнопки «+більше»

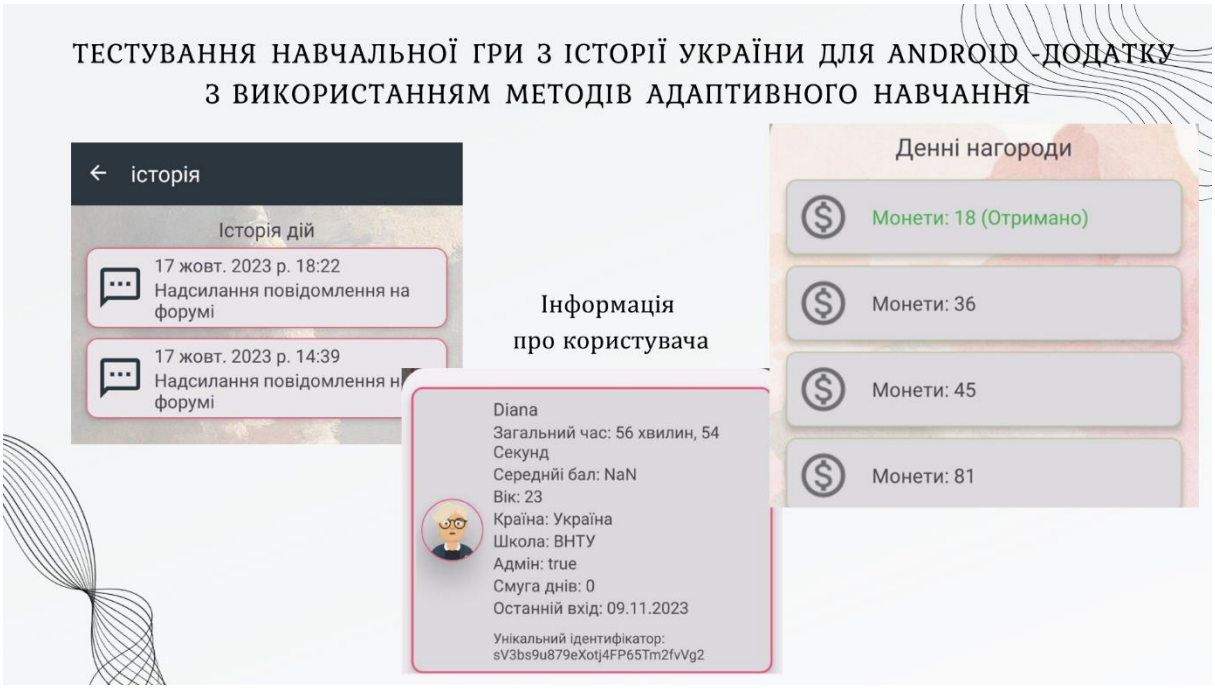

Рисунок Г.16 – Вікна історії дій, інформації про користувача та денні нагороди

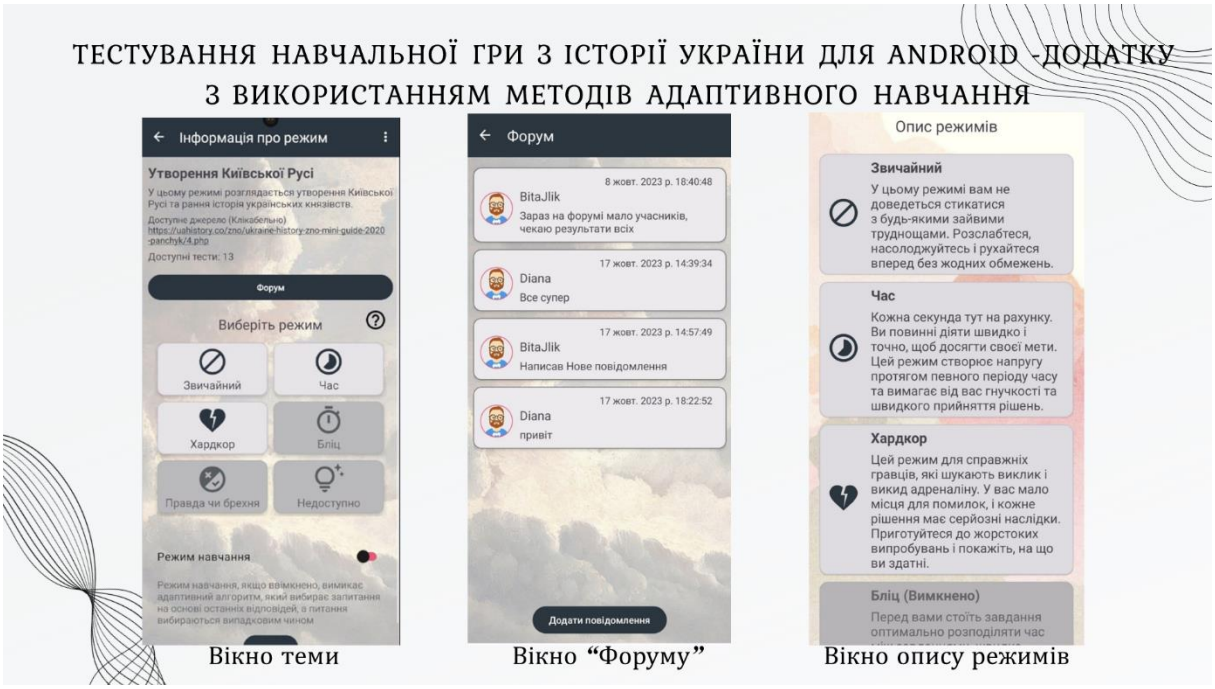

Рисунок Г.17 – Вікна теми, форуму та опису режимів

Рисунок Г.18 –Вікна статистики

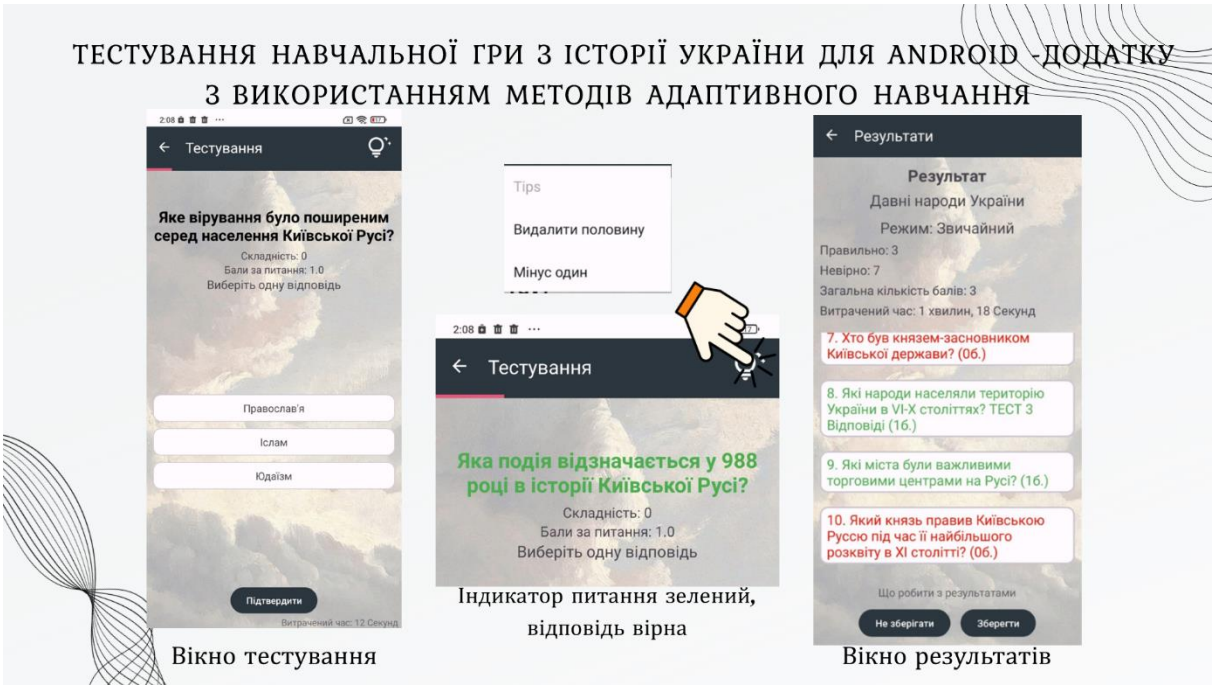

Рисунок Г.19 – Вікна тестування, результатів тесту та приклад індикатору відповіді

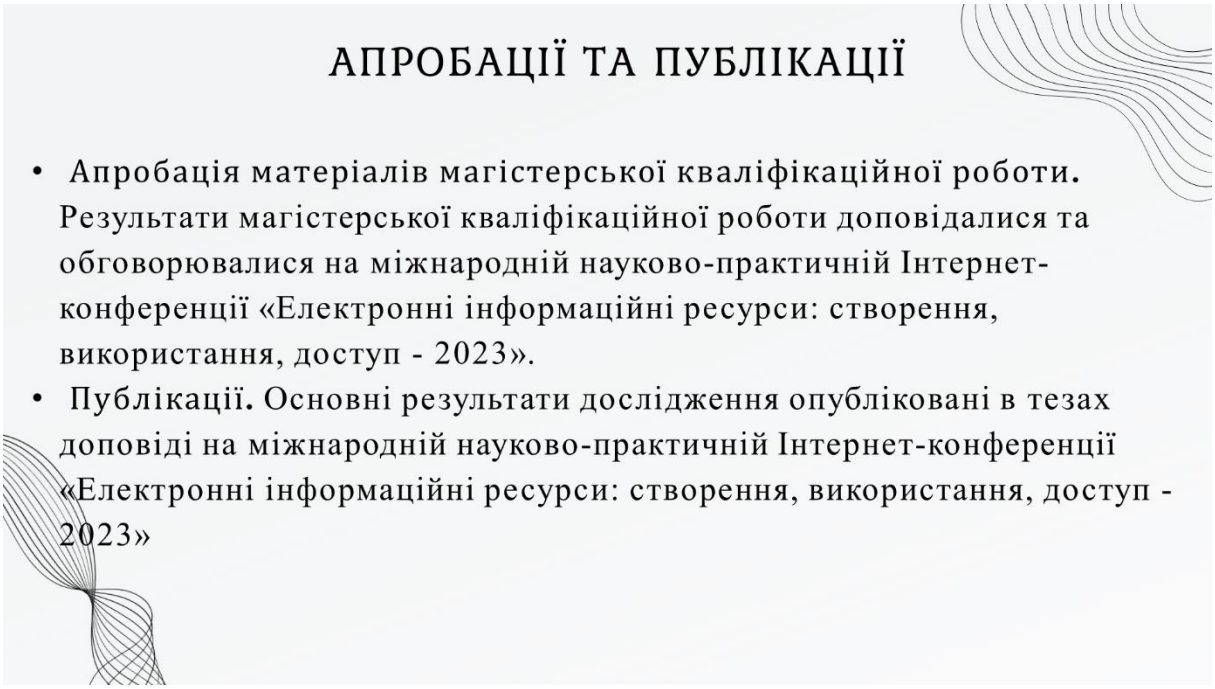

Рисунок Г.20 –Апробації та рублікації

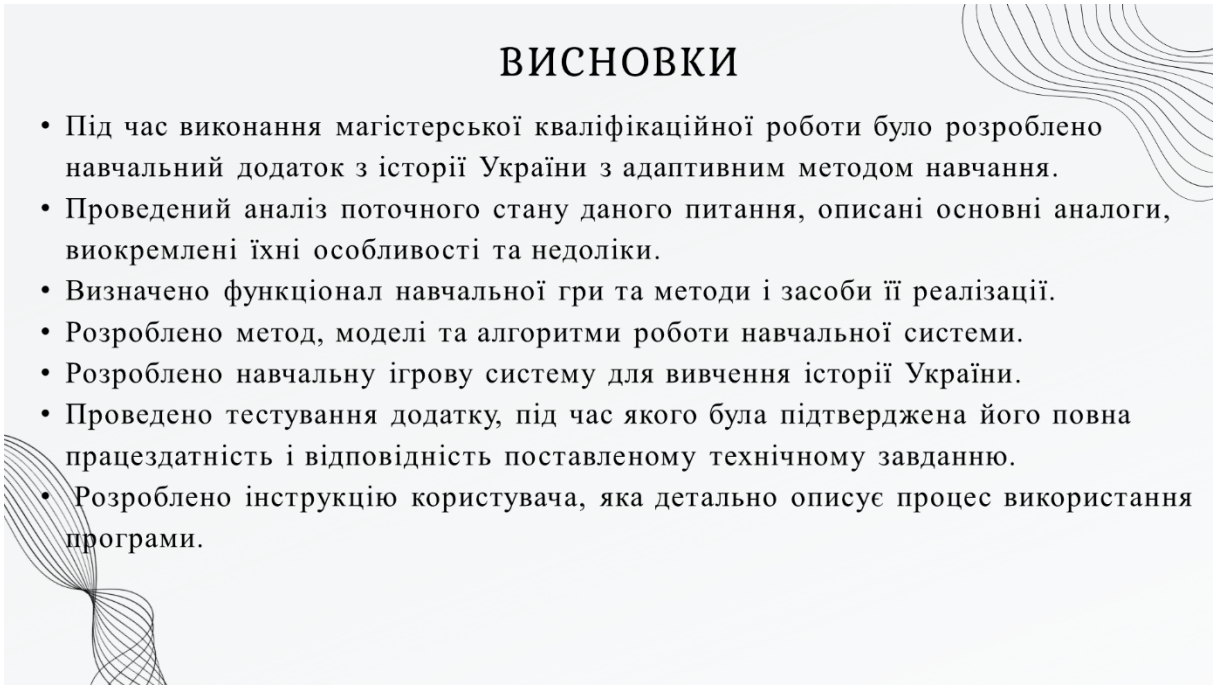

Рисунок Г.21 – Висновки

Рисунок Г.22 –Дякую за увагу