Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра програмного забезпечення

Бакалаврська дипломна робота

на тему: «Розробка мобільного програмного додатку для вивчення математики дітьми молодшої школи»

Виконав: студент 4 курсу
групи ЗПІ-18б
спеціальності
121 – Інженерія програмного забезпечення
(шифр і назва напряму підготовки, спеціальності)
Деда В.П.
(прізвище та ініціали)
Керівник: к.т.н., доц. каф. ПЗ Войтко В.В.
(прізвище та ініціали)
Рецензент: к.т.н., доц. каф. КН Арсенюк І.Р.
(прізвище та ініціали)

Допущено до захисту
Зав. кафедри _____
«____» _____ 2022 р.

ВНТУ – 2022

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Ступінь вищої освіти – бакалавр
Спеціальність 121 – Інженерія програмного забезпечення

УЗГОДЖЕНО
Директор Деражнянського ліцею №1
імені П. Стрілецького
Сідлецький В.М.
"____"_____2022 р.

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
Романюк О.Н.
"_25_" __березня___2022 р.

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

Деді Владиславу Петровичу

1. Тема роботи – розробка мобільного програмного додатку для вивчення математики дітьми молодшої школи.

Керівник роботи: Войтко Вікторія Володимирівна, к.т.н., доц. кафедри ПЗ, затверджені наказом вищого навчального закладу від 24 березня 2022 року № 66.

2. Строк подання студентом роботи 13 червня 2022 року

3. Вихідні дані до роботи: середовище розробки Android Studio, мова розробки Kotlin, операційна система – Windows 10, базові алгоритми для генерації прикладів.

4. Зміст розрахунково-пояснювальної записки: вступ; аналіз та постановка задачі; аналіз предметної області; аналіз аналогів; розробка методу покрокової роботи додатку; розробка методу та алгоритму генерації нового прикладу; розробка методу та алгоритму зчитування задач з файлу; обгрунтування вибору інтерфейсу; розробка інтерфейсу; розробка програми; тестування додатку; висновки; перелік посилань; додатки.

5. Перелік графічного матеріалу: графічний інтерфейс мобільного додатку; модель роботи мобільного додатку; блок-схеми алгоритмів роботи мобільного додатку; тестування мобільного додатку.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	Войтко В. В., к.т.н., доцент кафедри ПЗ		

7. Дата видачі завдання _____20 лютого 2022 року_____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської дипломної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз стану питання та постановка задач розробки	26.03.2022 – 03.04.2022	Вик.
2	Розробка алгоритмів, структури та моделей	04.04.2022 – 16.04.2022	Вик.
3	Розробка програмного забезпечення	17.04.2022 – 27.05.2022	Вик.
4	Тестування програми	28.05.2022 – 10.06.2022	Вик.

Студент _____ Деда В.П.
(підпис) (прізвище та ініціали)

Керівник бакалаврської дипломної роботи _____ Войтко В.В.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Бакалаврська дипломна робота складається зі 103 сторінок формату А4, на яких є 42 рисунки, 2 таблиці, список використаних джерел містить 20 найменувань.

У бакалаврській дипломній роботі проведено аналіз стану програмних продуктів для вивчення математики дітьми молодшої школи. Було проведено аналіз аналогів, визначено їх переваги і недоліки та доведено доцільність розробки власного програмного продукту.

Розроблено метод генерації нових прикладів відповідно до обраного рівня. Генерація прикладів відбувається у певному діапазоні відповідно до обраної складності і не порушує його.

Виконане обґрунтування вибору мови та середовища програмування, а також технологій, які були використані під час розробки мобільного додатку. Розроблено програмний продукт, який являє собою мобільний додаток під платформу Android.

Робота реалізована за допомогою мови програмування Kotlin. В якості середовища для розробки програмного забезпечення було обрано Android Studio.

Отримані в бакалаврській дипломній роботі результати можна використати для навчання дітей математики з-за допомогою мобільного додатку.

Ключові слова: мобільний додаток, методи генерування.

ANNOTATION

The bachelor's thesis consists of 103 A4 pages, which have 42 figures, 2 tables, the list of sources used contains 20 items.

In the bachelor's thesis the analysis of a condition of software products for studying of mathematics by children of elementary school is carried out. The analysis of analogues was carried out, their advantages and disadvantages were determined and the expediency of developing one's own software product was proved.

A method for generating new examples according to the selected level has been developed. That is, the generation of examples occurs in a certain range, according to the chosen complexity, and does not violate it.

Substantiation of the choice of language and programming environment, as well as technologies that were used during the development of the mobile application. Developed a software product that is a mobile application for the Android platform.

The work is implemented using the Kotlin programming language. Android Studio was chosen as the software development environment.

ЗМІСТ

ВСТУП

Обґрунтування вибору теми дослідження. На сьогоднішній день використання смартфонів стає все більш і більш популярним. Уявити людину, яка б не користувалась своїм помічником у вигляді мобільного телефона, просто неможливо. Саме з цієї причини ринок мобільної розробки є досить насиченим, а головне – перспективним, і тому багато з розробників звертають свою увагу на цей напрямок. Пріоритетними є не просто мобільні веб-версії застосунків, а саме повноцінні мобільні додатки, використання яких є більш зручнішим. Тому кількість створених мобільних додатків, для вирішення будь-яких проблем користувача просто величезна.

Майже кожний учень молодшої школи в наш час має свій смартфон. Використання різноманітних мобільних додатків дозволяє спростити наше життя та вирішити певні проблеми в ньому. Вони застосовуються у багатьох сферах нашого життя. Навчання та освіта є одними з таких невід'ємних складових. Саме тому, поява додатків, які б дозволили зробити перевірку своїх знань та закріплення вивченого матеріалу було лише питанням часу.

Ігрові програми мають не тільки розважальну мету, вони також часто мають на меті практичне застосування в освітніх установах. На сьогодні, це як ніколи актуально, зробити час проведений дитиною в телефоні не витраченим марно. Діджиталізація освітнього процесу набирає все більше і більше обертів, адже саме введення мобільних технологій в навчальний процес може збільшити інтерес та мотивацію до самого навчання, як мінімум це цікавіше[1].

На сьогодні, ОС Android є найпопулярнішою операційною системою для мобільних пристроїв. Ця операційна система підтримує велику кількість інструментів розробки для пристроїв від різних виробників. Основною причиною популярності ОС Android є безкоштовні засоби розробки, а розробка під систему IOS вимагає більших втрат для початку роботи з нею.

Сучасні інструменти, які створені для розробки Android додатків, дають можливість розробникам реалізувати велику кількість цікавих ідей з найрізноманітнішими цілями. Ці інструменти дають можливість представити інформацію користувачу у різних зручних форматах. Наприклад таких як: текст, відео, аудіо, тощо.

Для сучасної системи освіти впровадження мобільних технологій для покращення навчального процесу є актуальним завданням. Під час розробки мобільного додатку слід приділити особливу увагу саме його інформаційному наповненню, орієнтованому на учнів молодшого шкільного віку. Тому актуальною є розробка мобільного програмного додатку для вивчення математики дітьми молодшої школи.

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

Мета та завдання дослідження. Метою роботи є розширення функціональних можливостей навчального процесу в молодшій школі шляхом створення мобільного додатку для вивчення дітьми математики, що підвищить зацікавленість учнів процесом навчання.

Відповідно до поставленої мети потрібно вирішити такі **завдання**:

– визначити найбільш ефективні засоби для реалізації навчального Android-додатку;

– розробити метод та модель роботи програми;

– розробити алгоритм роботи для генерації прикладів з різними діями і різними рівнями складності;

– визначити найефективніший спосіб зчитування задач різних рівнів складності з файлу засобами мови програмування Kotlin;

– розробити Android-додаток для вивчення та застосування знань на практиці з математики дітьми молодшої школи;

– провести тестування розробленого мобільного додатку.

Об’єктом дослідження є процеси розробки навчального мобільного додатку.

Предметом дослідження є методи і засоби розробки програмного продукту, принципи програмування мови Kotlin та засоби середовища Android Studio.

Методи дослідження. У процесі досліджень використовувались такі методи дослідження:

– метод побудови Android-додатку для побудови стабільного додатку;

– метод генерації нових прикладів для реалізації підбору завдань;

– метод сповіщення користувача про правильність розв’язання задачі чи прикладу для формування системи оцінювання результатів;

– методи тестування для перевірки працездатності програми.

Новизна отриманих результатів.

1. Подальшого розвитку дістав метод генерації нових прикладів відповідно до обраного рівня складності, який, на відміну від відомих, генерує приклади у певному діапазоні та повністю відповідає обраному рівню складності, що дозволяє забезпечити автоматичний підбір задач обраного рівня.

2. Подальшого розвитку дістала модель навчальної системи, яка, на відміну від відомих, використовує новий метод підбору задач і засоби комунікації з користувачем-дитиною у процесі виконання завдань, що дозволяє спростити користувацький інтерфейс і адаптувати програму під потреби навчального процесу молодшої школи.

Практична цінність отриманих результатів. Практична цінність одержаних результатів полягає у кінцевій реалізації мобільного Android-додатку для вивчення математики дітьми молодшої школи, що може використовуватись без допомоги дорослих.

Особистий внесок здобувача. Усі результати, викладені у бакалаврській дипломній роботі, отримані автором особисто. У науковій праці[2], опублікованій у співавторстві, автору належать такі результати:

розробка мобільного додатку «MathForKids», спрямованого на вивчення математики дітьми молодшої школи.

Апробація результатів роботи. Результати роботи були представлені на LI Науково-технічній конференції факультету інформаційних технологій та комп'ютерної інженерії (2022 р., м.Вінниця).

Публікації. Результати роботи були опубліковані в матеріалах конференції [2] – LI Науково-технічній конференції факультету інформаційних технологій та комп'ютерної інженерії (2022 р., м.Вінниця).

Аналіз. У пояснювальній записці до бакалаврської дипломної роботи було розглянуто 4 розділи та було використано 20 літературних джерел

1 АНАЛІЗ СТАНУ ПИТАННЯ ТА ПОСТАНОВКА ЗАДАЧ РОЗРОБКИ

1.1 Аналіз предметної області

Сьогоденний рівень розвитку інформаційних технологій, дає можливість використовувати сучасні методики навчання в освітньому процесі.

Термін «мобільне навчання», останнім часом став все частіше використовуватись у наші країні. Багато вчених та педагогів вважають, що саме від популярності смартфонів та наявності навчальних програм, залежить навчання з використанням мобільних технологій. За останні роки в Україні кількість користувачів мобільних телефонів зросла в три рази. Про це можна зробити висновки з результатів дослідження компанії TNS Infetest на замовлення Google.

Мобільне навчання вважається новою стадією розвитку електронного навчання, що використовує смартфони та безпровідний доступ до навчальних ресурсів як засіб навчання. Мобільне навчання можна поєднувати з традиційним навчанням, що дає можливість реалізувати змішане навчання (див. рис.1.1). Саме використання мобільного телефону як основного засобу для навчання і доступу до ресурсів – є основоною відмінністю мобільного навчання від інших систем навчання. Місцезнаходження самих учнів, під час того як проходить таке навчання може бути різним, як у класі, так і поза його межам.

Рисунок 1.1 - Мобільне навчання в системі змішаного типу

Сучасний темп використання та розширення функціональних можливостей смартфонів змушують освітню спільноту задуматись над використанням цих технологій з навчальною метою. Протягом останніх років активно обговорюються нові технології та методи навчання з-за допомогою мобільних телефонів і використання навчальних додатків для цього. Адже, застосування розвиваючих мобільних додатків здатне підвищити мотивацію учнів до навчання [3].

Мобільні додатки за ступенем мобільного впливу можуть бути розділені на наступні види:

– навчальні, які сприяють отриманню нових знань на навичок в тій чи іншій області знань;

– тренувальні, які виконують закріплюючу і контролюючу функцію;

– розвиваючі, які сприяють розвитку найбільш важливих навичок;

– комбіновані, які поєднують в собі в різних варіаціях всі три вище перераховані види.

Використання мобільних пристроїв в цілях навчання, загалом стимулюють школярів до роботи. Зокрема, доцільно виділити наступні позитивні процеси, які відбуваються в результаті застосування смартфонів в цілях отримання нових знань:

– відбувається активізація навчально-пізнавальної діяльності навіть у тих учнів, які до цього не сильно цікавились навчанням;

– за рахунок гнучкості та швидкому доступу до навчального матеріалу, підвищується продуктивність навчання;

– є можливість організації самостійного навчання, та більшої концентрації уваги на тій чи іншій темі;

– можна реалізувати змішане навчання.

Очевидно, що школярі різного віку полюбляють нові технології та всі можливості які вони надають, що звісно ж мотивує їх вчитись все більше і більше. Можна з впевненістю сказати, що мобільні додатки є досить ефективним інструментом для навчання, як в школах, так і вдома варто використовувати їх як доповнення до традиційного навчання. Мобільні додатки навчального характеру, мотивують школярів до досягнення нових цілей, у цікавій для них формі.

1.2 Аналіз аналогів розроблюваного програмного продукту

Додавання і віднімання, ділення і множення – саме ці прості математичні операції, діти вивчають зазвичай на простих прикладах, коли замість цифр використовуються різні палички, іграшки або інші предмети. Але для дітей XXI століття, які ще не впевнено тримаючись на ногах, але вже досить впевнено вміють вправлятись з телефоном чи планшетом, розвиваючі ігри стають дуже у нагоді.

Серед існуючих реалізацій Android-додатків для вивчення математики, та засвоєння цих знань на практиці, шляхом вирішення прикладів різної складності, найбільш близькими до створюваного додатку «MathForKids» є наступні [4]:

– DragonBox Algebra;

– Kids Number and Math Lite;

– Operation Math;

– Mathmateer;

– Geeksmath.

DragonBox Algebra – це ігровий мобільний додаток з головним героєм у вигляді дракончика. Додаток розрахований на дітей віком від 5 років. Головний герой допомагає дітям опанувати прості математичні дії. Сам додаток, був створений вчителем математики, і поєднує в собі навчання і гру. Також даний розвиваючий додаток передбачає 10 рівнів, а також 200 загадок. У додатку є ігрове поле, яке ділить екран користувача на дві рівні частини Екран вікна, у якому генеруються нові приклади, а також розв'язання цих згенерованих прикладів продемонстровано на рисунку 1.2 [5].

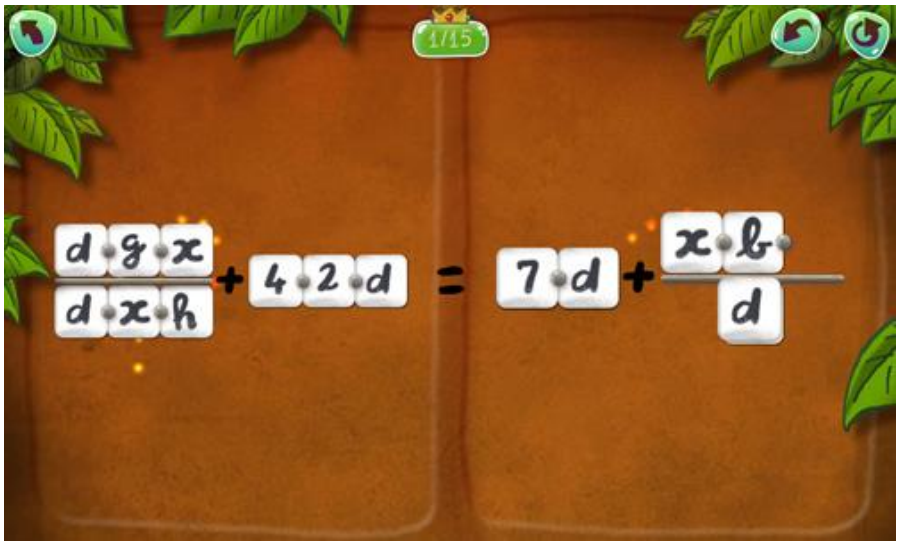


Рисунок 1.2 – Етап проходження гри DragonBox Algebra

Програма «Kids Numbers and Math Lite» була розроблена компанією «Intellijoy». Даний додаток розрахований на дітей дошкільного віку та учнів молодшої школи. Додаток містить декілька розділів, в яких діти мають можливість ознайомитись з числами, навчитись рахувати в умі, додавати і віднімати. Мобільний додаток має інтерфейс російською мовою та легко встановлюється на пристрої з ОС Android. Головне вікно програми «Kids Numbers and Math Lite» зображено на рисунку 1.3 [6].

Рисунок 1.3 - Головне вікно програми «Kids Numbers and Math Lite»

Маленьким дослідникам додаток «Operation Math» сподобається своїм досить захоплюючим та реалістичним ігровим сюжетом. Головна мета цієї гри є перемога над підступним і злим доктором Оддом. Це додаток містить три рівні, які є різні за складністю. Самі завдання представлені у цікавій та яскравій формі, створюючи у дитини відчуття, що вона не навчається, а грає. Додаток можна установити як на Андроїд так і на iOS [7]. Головне вікно програми зображено на рисунку 1.4.

Рисунок 1.4 - Головне вікно мобільного додатку «Operation Math»

Додаток «Mathmateer» ніби переносить дитину у світ космічних пригод, десь до далеких Галактик. Інтерфейс програми доступний лише англійською мовою. Цей мобільний додаток добре тренує логіку, увагу та пам'ять. На початку свого навчання дитині потрібно обрати свого провідника у світ космосу та математики серед 15 запропонованих героїв. Додаток розрахований на дітей віком від 9-ти років та доступний лише для сисеми iOS. Вікно, у якому користувач бачить згенеровані приклади, та вікно заміни вигляду ракети, яка супроводжує протягом всієї гри, зображені на рис.1.5 [8].

Рисунок 1.5 – Вікно з прикладами та вікно зміни ракети у додатку «Mathmateer»

Мобільний додаток «Geeksmath» – це програма, яка створена для учнів в якості репетитора. Дана програма також має можливість вибору різних рівнів складності, а також, якщо у користувача протягом гри виникнуть проблеми з розв’язанням того чи іншого завдання, доступна можливість запросити допомогу з вирішенням. Додаток дає можливість відслідковувати прогрес учня і зберігає всю статистику. Інтерфейс додатку написаний на російській мові [9]. Приклад роботи додатку «Geeksmath» зображено на рисунку 1.6.

Рисунок 1.6 - Приклад роботи додатку «Geeksmath»

Після аналізу усіх аналогів визначено їхні переваги та недоліки та проведено порівняння із розроблюваним Android-додатком під назвою «MathForKids». Результат порівняння зведено в таблицю 1.1.

Таблиця 1.1 – Порівняльні характеристики мобільних додатків

Назва гри / Критерій	DragonBox Algebra	Kids Number and Math Lite	Operation Math	Mathmateer	Geeksmath	MathForKids
Наявність рівнів складності завдань	-	-	+	+	+	+
Інтерфейс українською мовою	-	-	-	-	-	+
Зрозумілий інтерфейс	+	+	-	+	-	+
Кросплатформність	-	+	+	-	+	-
Підсумковий результат	1	2	2	2	2	3

Отже, можна зробити висновок, що у порівнянні з існуючими аналогами, розробка власного мобільного додатку є доцільною. В результаті роботи, отримаємо додаток, який буде покривати недоліки існуючих на даний момент аналогів та забезпечить вищу ефективність роботи.

1.3 Аналіз методів розв'язання поставленої задачі

Якщо століття тому люди в різних частинах світу перебували в золотій лихоманці, то тепер можна з упевненістю сказати, що сучасний світ повністю поглинений мобільною лихоманкою. На сьогоднішній день, популярність мобільних телефонів росте з неймовірною швидкістю, а в розвиток та маркетинг мобільних технологій їхній виробники інвестують досить великі кошти. Розробка додатків для смартфонів є дуже швидкозростаючою сферою програмування, оскільки кількість комп'ютерів, значно менша за кількість мобільних телефонів.

Існує два способи розробки мобільних додатків:

– нативна розробка;

– кросплатформенна, або гібридна розробка (ReactNative, Flutter, Xamarin).

Під поняттям «нативна розробка» розуміється, розробка програми для смартфона на конкретній мові програмування для конкретної операційної системи. Такі додатки є доволі продуктивними і не мають обмежень у розробці. Для нативної розробки під ОС Android використовується Java або Kotlin, а для iOS – Swift. Плюси такої розробки наступні:

– можливість отримати прямий доступ до апаратної частини смартфона;

– розробка звичного для користувача інтерфейсу для певної платформи;

– швидка реакція на дії користувача.

А от недоліки у такого способу розробки наступні:

– висока вартість розробки і підтримки додатку;

– довший час необхідний на розробку.

Досить довгий час займає розробка одного і того ж самого додатку для різних операційних систем. Крім вартості, постає проблема і в тривалості самої розробки, адже кожен продукт, крім того що потрібно розробити, його потрібно протестувати. Тому, якщо потрібно створити простий додаток, який буде використовуватись для двох операційних систем, то у цьому випадку перевагу надають гібридному способу розробки.

Кросплатформна розробка здійснюється з-за допомогою таких веб-технологій як: HTML, CSS та JavaScript. Саме ці технології дозволяють розробку додатку на одразу на декілька платформ. Серед переваг такого способу розробки є низька вартість цієї розробки, тому що для цього, інколи буває достатньо одного фахівця. А ось якщо говорити про недоліки, то до них можна віднести можливу затримку реакції на дії користувача, а також сповільнення роботи самого додатку [10].

1.4 Постановка задач дослідження

Провівши аналіз питання розробки Android-додатку для вивчення дітьми молодшої школи математики «MathForKids», було визначено наступні завдання, які необхідно розв’язати у процесі розробки програмного продукту:

– розробити метод та модель роботи програми;

– розробити алгоритм генерації прикладів відповідно до обраної користувачем складності та генерації відповідей до нього;

– розробити інтерфейс всіх вікон додатку;

– розробити алгоритм зчитування задач з файлу, відповідно до того який рівень складності вибрав користувач;

– реалізувати збереження поточного балансу монет;

– розробити алгоритм збереження теми, яку користувач може змінювати протягом гри;

– провести тестування мобільного додатку.

Технічне завдання на розробку наведено в додатку А.

1.5 Висновки

У даному розділі було проведено аналіз з наявними програмами-аналогами і здійснено порівняння між собою та розроблюваною програмою «MathForKids» за певними критеріями. Також було детально проведено аналіз предметної області. Було проаналізовано способи розробки мобільних додатків і в результаті аналізу та порівнянь переваг та недоліків, було обрано метод «нативної розробки». Також, було поставлено основні задачі, які потрібно виконати протягом розробки мобільного мобільного додатку під платформу Android.

2 РОЗРОБКА МЕТОДУ, АЛГОРИТМІВ ТА МОДЕЛІ ПРОГРАМИ

2.1 Розробка методу побудови головного вікна додатку

При розробці мобільного Android-додатку було використано один з найпопулярніших шаблонів програмування Model-View-ViewModel, або скорочено MVVM. Даний шаблон програмування, використовується для розробки користувацького інтерфейсу і розділяє між собою бізнес логіку та логіку відображення.

MVVM складається з таких трьох компонентів:

– model, яка представляє собі дані, котрі необхідно показати користувачу;

– view – це клас, який відповідає за відображення даних, і в Android-додатках це зазвичай Activity, або Fragment;

– viewModel – це клас, який з’єднує View і Model, і цей клас підписаний на оновлення Model, а View підписана на ViewModel, і при цьому ViewModel не має явного посилання на View.

Для реалізації головного вікна програми, макет якого зображено на рисунку 1.2, а саме списку кнопок, які розташовані у вигляді плиток, з можливість прокручувати реалізовані за допомогою RecyclerView.

Рисунок 2.1 – Макет головного вікна мобільного додатку «MathForKids»

Елемент RecyclerView появився в Android 5.0 Lollipop і став покращеною версією ListView. Схематично роботу RecyclerView можна описати наступним чином. На екрані відображаються видимі елементи списку, і коли при прокрутці верхній елемент списку стає невидимим і виходить за його межі, його вміст очищується. При цьому, цей очищений елемент переміщується вниз екрану і наповнюється новими даними, тобто перевикористовується. За рахунок цього і була досягнута більша продуктивність і прокрутка стала більш плавною ніж у ListView.

У додатку «MathForKids» при виконані завдань правильно, користувач заробляє монети, і саме від правильних відповідей залежить кількість монет, адже при неправильній відповіді монети віднімаються. Для постійного зберігання балансу монет, було прийнято рішення зберігати це значення за допомогою SharedPreferences, адже саме воно призначене для постійного

збереження невеликої кількості даних. Значення примітивних типів даних зберігаються у вигляді пари ключ-значення.

Програма побудована за принципом Single-Activity Architecture, тобто було створено одне activity, яке, в свою чергу, вміщує в собі фрагменти, навігація між якими здійснювалась за допомогою Navigation Graph. Activity – це один з основних компонентів в Android, а також це екран, який бачить і з яким взаємодіє користувач. Може містити різні View та Fragment. В свою чергу fragment використовуються для побудови динамічних інтерфейсів і взаємодії з Activity. Головною перевагою фрагментів є спрощення роботи з UI. Одна activity може взаємодіяти з необмеженою кількістю fragments. Навігація між екранами fragments буде реалізована за допомогою Navigation Graph. Navigation Graph – це граф, який оприділяє всі можливі шляхи, які доступні у додатку користувачу. Також, ще одним великим плюсом у цього графа є можливість побачити візуально всі можливі шляхи, у мобільному додатку. Приклад такого вікна, зображено на рисунку 2.2.

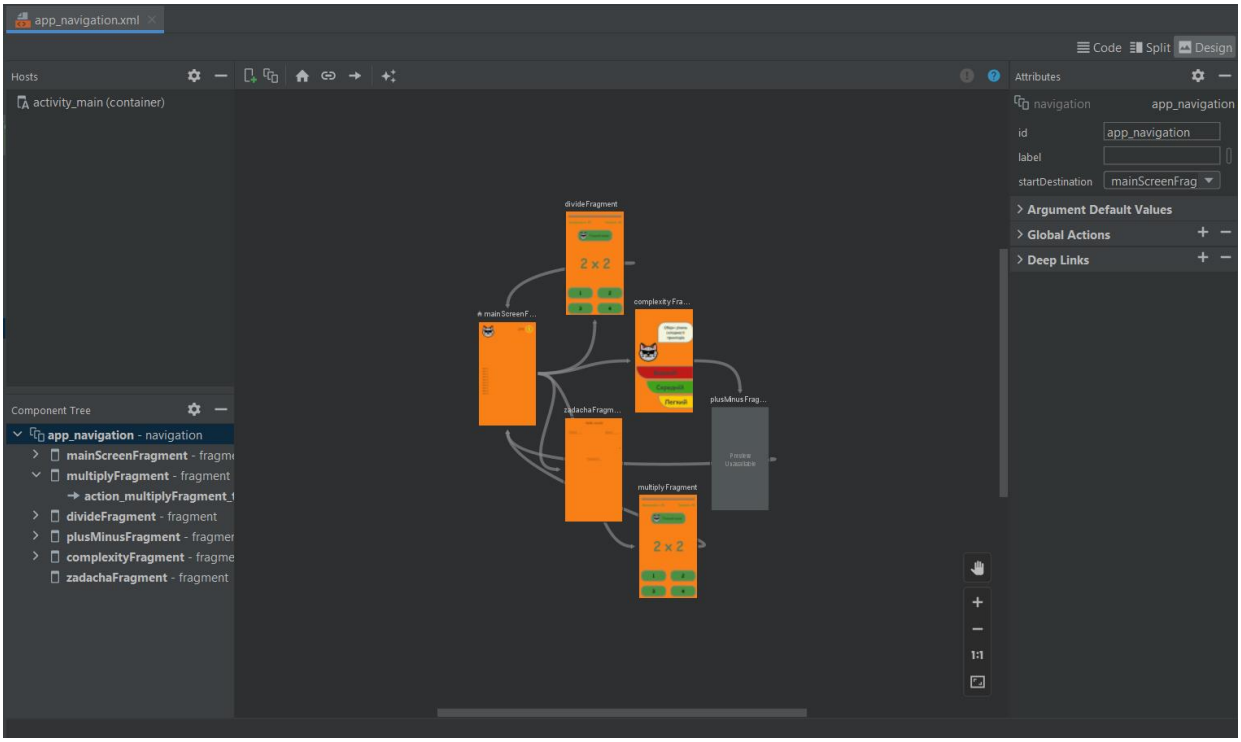


Рисунок 2.2 – Вікно XML файла з навігацією в Android studio

Таймер, який рахує кількість секунд, які залишились на виконання завдання, був реалізований за допомогою coroutines. Coroutine – це модель дизайну одночасності, який можна використовувати на Android для спрощення коду, що виконується асинхронно. Coroutine були додані до Котліна у версії 1.3 і засновані на встановлених концепціях інших мов. Coroutine допомагає керувати тривалими завданнями на Android, які в іншому випадку можуть заблокувати основний потік і змусити робочу програму не відповідати. Понад 50% професійних розробників, які використовують Coroutines, повідомили про підвищення продуктивності праці з використанням таймерів контролю робочого часу. Coroutines – це рекомендоване рішення від Google для асинхронного програмування на Android [11].

2.2 Розробка методу покрокової роботи мобільного додатку «MathForKids»

Кожна людина щодня має справу з різними алгоритмами, навіть не зовсім розуміючи це. Написати алгоритм можна навіть для звичайного походу в магазин, і при умові що алгоритм написано правильно, то покупець успішно вернеться додому з покупками. Адже алгоритм це є опис послідовних дій, який призводить до кінцевого результату. І якщо порушити черговість виконання, чи просто втратити якийсь крок, алгоритм може привести не до того результату якого очікувалось, або й взагалі перестати працювати [12].

Алгоритм роботи мобільного додатку «MathForKids» наступний:

1. Запуск Splash screen, або екрану запуску. Під час завантаження програми показується логотип і назва програми з анімацією миготіння.
2. Запуск головного вікна програми, на якому відображається тваринка, поточний баланс монет користувача, і список кнопок який прокручується горизонтально. Цей список, дає можливість користувачеві вибрати той чи інший тип завдання.

3. Можливість обрати тип завдання.
4. Запуск діалогового вікна з можливістю вибрати рівень складності завдання.
5. Запуск вікна програми з прикладами.
6. Генерація прикладів. При правильній відповіді, до рахунку додається один, в іншому випадку відповідно з рахунку віднімається один.
7. Асинхронна робота таймера, тобто робота таймера в іншому потоці, для уникнення блокування основного.
8. Після закінчення часу, на екрані появляється діалогове вікно, яке сповіщає про це, а також показує кількість зароблених монет.
9. Можливість обрати інший тип завдання.
10. Закінчення роботи програми.

Блок-схему алгоритму представлено на рисунку 2.3

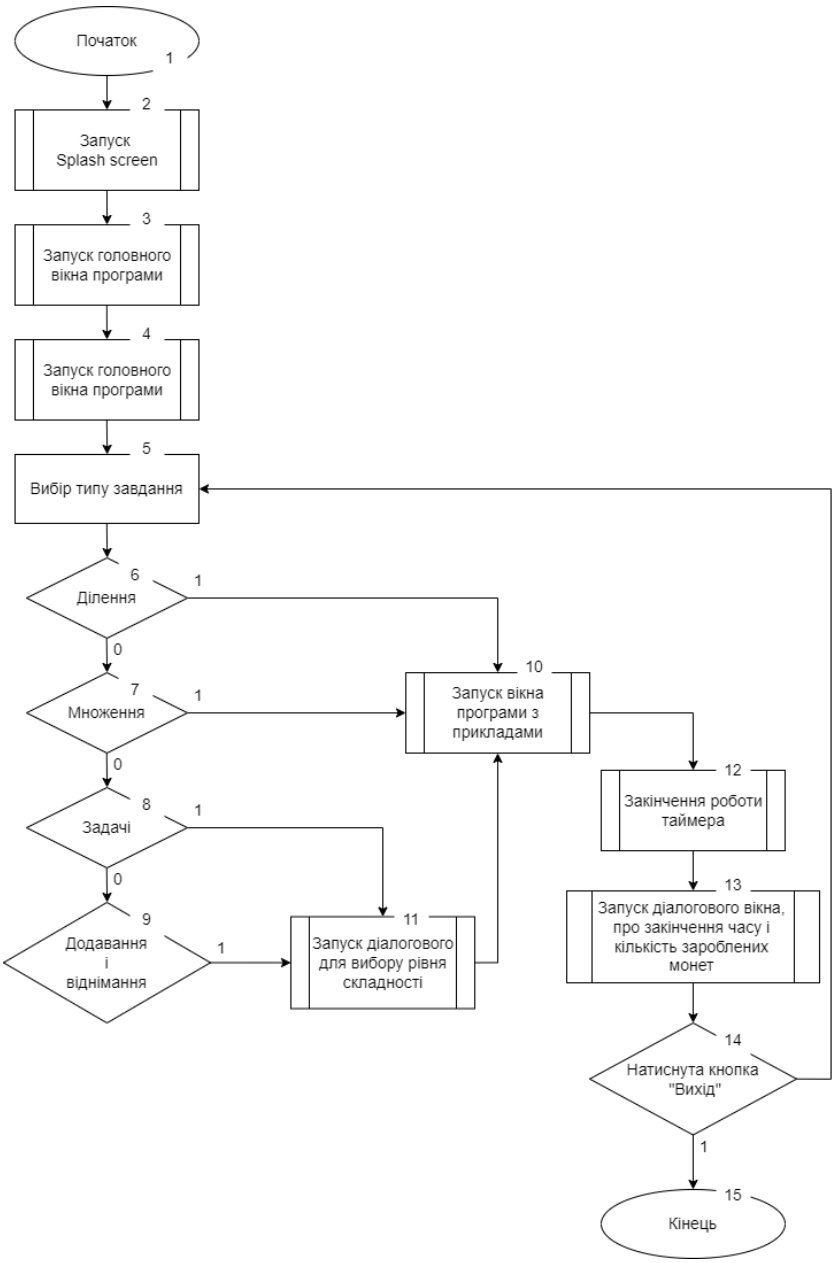


Рисунок 2.3 - Блок-схему алгоритму роботи мобільного додатку «MathForKids»

2.3 Розробка методу генерації нового прикладу

Алгоритм генерації нового прикладу є одним з основних алгоритмів в мобільному додатку «MathForKids». Адже саме від правильної побудови даного алгоритму, залежить коректність роботи функцій додатку.

Алгоритм генерації нового прикладу для дії додавання або віднімання, який складається з двох чисел, правильної відповіді і трьох неправильних, складається з наступних кроків:

1. Генерація пари чисел, відповідно до обраної складності прикладів. Тобто, коли обрана складність EASY, числа генеруються в діапазоні від 1 до 10, коли обрано рівень MEDIUM – від 1 до 100, і коли обрано рівень HARD верхня межа для генерації пари чисел є 1000.
2. Генерація boolean значення, і відповідно якщо буде true – дія у прикладі буде додавання, в іншому випадку – дія буде віднімання.
3. Якщо випало додавання, програма перевіряє, який рівень складності обрано. І якщо два числа при додаванні виходять за свою межу, яка складає відповідно 10, 100, 1000, пара чисел буде генеруватись доки не пройде цю умову.
4. Якщо випало віднімання, програма перевіряє, чи є перше число зі згенерованої пари меншим за друге. Якщо так, вони міняються місцями. Це зроблено для того, щоб запобігти від’ємних значень у відповіді.
5. Встановлення максимального значення, для верхньої межі генерації чисел неправильних відповідей, відповідно до правильної відповіді та значень виразу.
6. Якщо межа менша, ніж число 5, верхня межа встановлюється число 4. Це зроблено для того, щоб запобігти безкінечного генерування неправильних відповідей, тобто, щоб у діапазоні було мінімум 4 числа.
7. Генерація масиву з трьох значень типу int неправильних відповідей. Це відбувається, поки не заповниться масив, а значення генеруються від 1 до максимального значення, встановленого раніше. Якщо значення не дорівнює правильній відповіді і є унікальним в масиві, воно додається в цей масив.
8. Метод повертає повністю готовий приклад у вигляді data class-у з назвою MathExerciseModel (див. рис.2.4), який містить два значення, правильну відповідь, і три неправильні.

```
data class MathExerciseModel(
    val firstValue:Int,
    val secondValue:Int,
    val answerValue: Int,
    var wrongAnswers: ArrayList<Number>
)
```

Рисунок 2.4 – Код data class-у MathExerciseModel

Блок-схему алгоритму роботи розробленого методу подано на рис. 2.5.

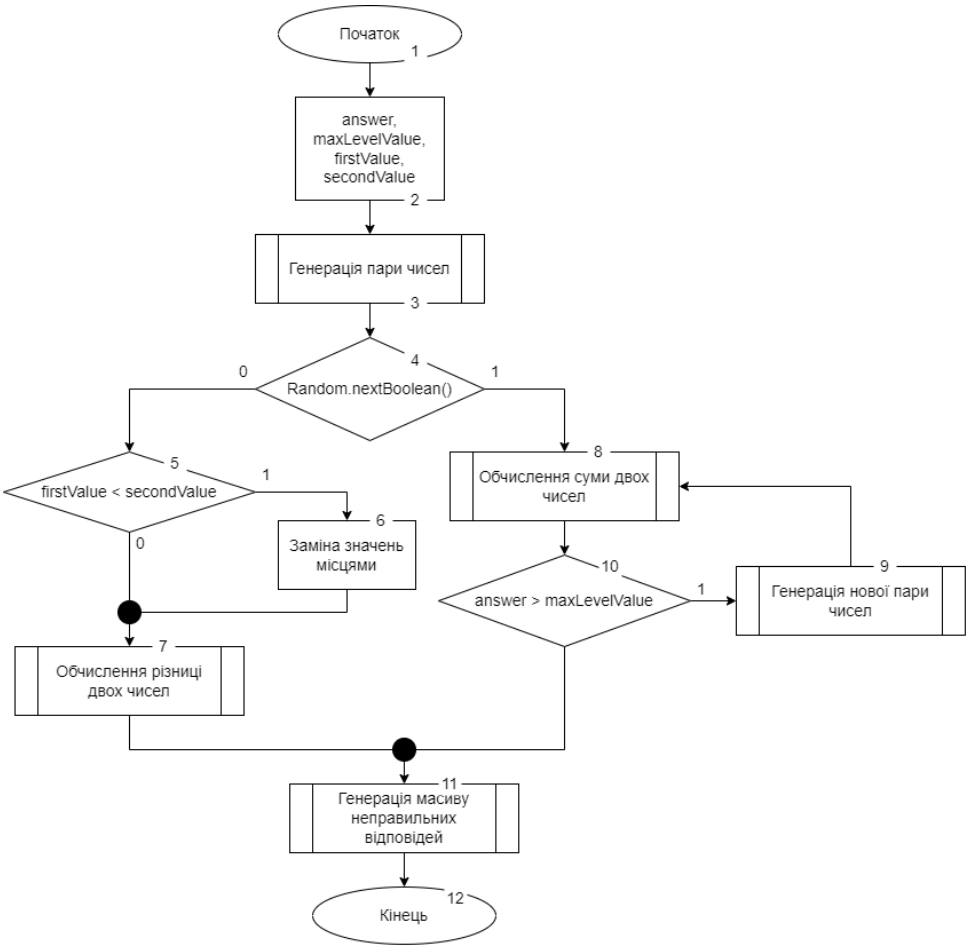


Рисунок 2.5 – Блок-схема алгоритму роботи генерації нового прикладу для дії додавання та віднімання

2.4 Розробка методу та алгоритму для зчитування задач з файлу

Алгоритм зчитування задач з файлу є одним з основних алгоритмів у мобільному додатку. Коли користувач вибирає задачі на головному вікні, появляється діалогове вікно, у якому користувач має обрати рівень складності задач. Існує три рівні складності: легкий, середній та важкий. Задачі зберігаються у трьох окремих файлах. Задача містить текст задачі і

відповідь до неї. Вони розділені між собою знаками «***». А далі у самі відповідь і текст задачі розділений за допомогою вертикальної лінії, для зручнішого зчитування файлу. Уривок файлу, для задач легкого рівня, під назвою «zadachi_easy.txt», зображено на рисунку 2.6.

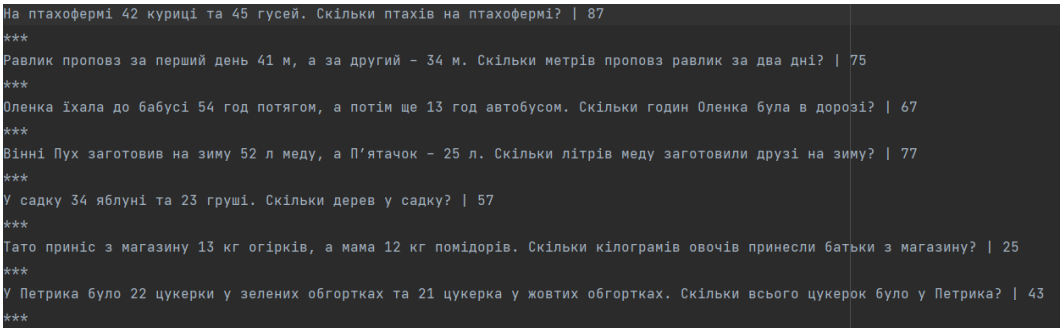


Рисунок 2.6 – Уривок файлу «zadachi_easy.txt»

Алгоритм зчитування з файлу наступний:

1. Визначення обраного рівня задач, який передається в даний фрагмент задач у аргументах. Значення передається типу String, і має назву level, яка прописана у класі констант.
2. Визначення імені файлу який буде зчитуватись, відповідно до обраного користувачем рівня. Наприклад, коли обрано середній рівень задач, ім'я файлу буде «zadachi_medium.txt».
3. Відповідний файл, дістається з папки «assets». Даний «package» використовується для зберігання інформації у вигляді картинок, аудіо і в тому числі текстових файлів.
4. Вміст файлу зчитується у вигляді однієї стрічки.
5. Стрічка розбивається за допомогою функції «split», на масив типу «String». Функція має «delimiter», це те по чому вона розділяє. В цьому випадку це «***».
6. Далі береться кожна стрічка масиву і знову за допомогою функції «split» розділяється на два значення, текст задачі і відповідь.

7. За допомогою функції «map», ці два значення зберігаються в масив у вигляді об’єкту «Zadacha», який містить ці два значення, а саме текст задачі і відповідь до неї.
8. Встановлення відповідних значень задачі у відповідні view на фрагменті

Блок-схему алгоритму представлено на рисунку 2.7.

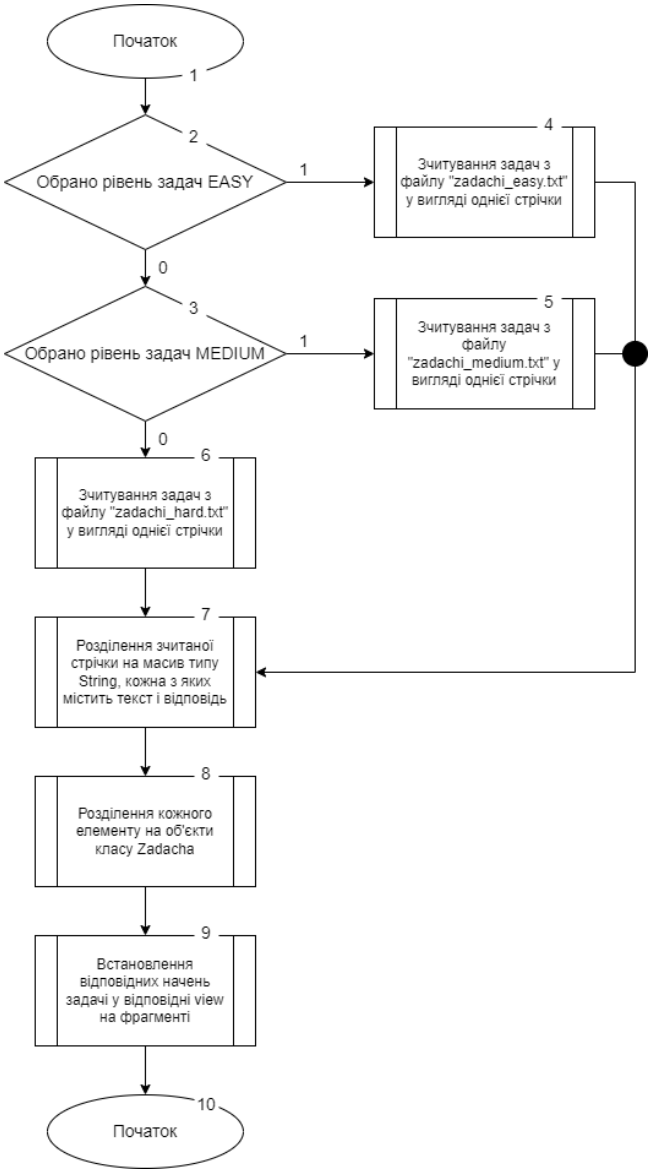


Рисунок 2.6 – Блок-схема алгоритму зчитування задач з файлу

Об’єкти, у вигляді яких зберігаються задачі у масиві – це data class, основне призначення якого і є збереження даних, який містить два поля,

перше типу String, в якому зберігається текст задачі, а друге типу Int, яке зберігає правильну відповідь для неї.

2.5 Розробка моделі роботи навчальної системи

Щоб краще зрозуміти взаємодію користувача із навчальною системою у вигляді Android-додатка, було прийнято рішення розробити модель роботи методом User Flow (див. рис. 2.7). Дана діаграма дає можливість краще описати взаємодію користувача з мобільним додатком.

Відповідно до діаграми, першим кроком користувача є перегляд головної сторінки. Наступним кроком є вибір завдання користувачем.

Наступний крок це вибір рівня складності прикладу або задачі, відповідно до якого будуть генеруватися приклади і задачі ввідповідного рівня.

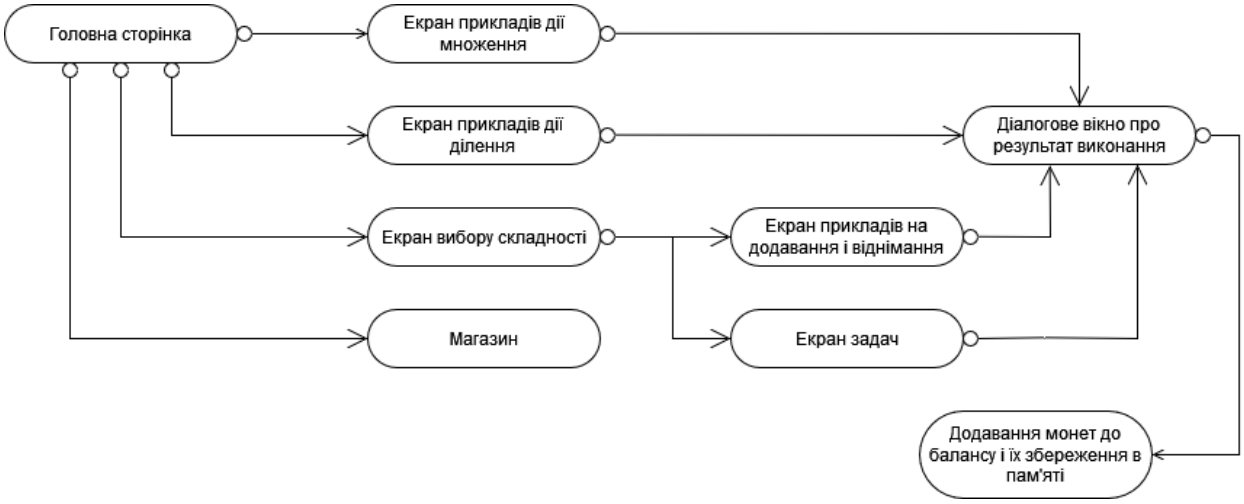


Рисунок 2.7 – Модель User Flow взаємодії користувача із навчальною системою

2.6 Висновки

У даному розділі було розроблено алгоритм побудови головного вікна мобільного додатку, описано алгоритм роботи розробленої програми, що дозволяє швидко зрозуміти весь цикл роботи та побачити основні операції програми. Розроблено та описано алгоритм генерації нового прикладу та алгоритм зчитуваввння задач з файлу.

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Варіантний аналіз та обґрунтування вибору засобів реалізації програмного продукту

На сьогоднішній день, технології розробки програмного забезпечення розвиваються з блискавичною швидкістю. Тому, не є дивиною, що кількість мов програмування так само зростає.

Вибір мови програмування, для розробки того чи іншого програмного забезпечення, є дуже важливим питанням яке постає перед кожним початком розробки програми. Для розробки мобільного додатку, який буде спрямований для роботи на смартфонах під управлінням ОС Android, в пріоритеті буде мова яка має відповідний функціонал для роботи з графічним інтерфейсом, є можливість обробляти дії користувача, є можливість легко та швидко працювати з файлами. Розглянемо найпопулярніші мови програмування, які використовуються для розробки мобільних Android-додатків, а саме Java та Kotlin.

Java — це універсальна, об'єктно-орієнтована мова програмування, яка заснована на класах, розроблена для менших залежностей реалізації. Це платформа для розробки додатків. Тому Java є швидкою та надійною. Ця мова широко використовується для розробки Java-додатків для ноутбуків, центрів обробки даних, ігрових консолей, наукових суперкомп'ютерів, мобільних телефонів тощо.

Платформа Java — це набір програм, які допомагають програмістам ефективно розробляти й запускати програми програмування на Java. Платформа включає в себе механізм виконання, компілятор і набір бібліотек. Це набір комп’ютерного програмного забезпечення та специфікацій. Джеймс Гослінг розробив платформу Java в Sun Microsystems, а пізніше її придбала корпорація Oracle [13].

Kotlin — це статично типізована мова програмування із відкритим вихідним кодом, яка підтримує як об’єктно-орієнтоване, так і функціональне програмування. Kotlin надає подібний синтаксис та концепції як інші мови, включаючи C#, Java та Scala. Kotlin не прагне бути унікальним — натомість він черпає натхнення з десятиліть розвитку мови. Він існує у варіантах, націлених на JVM (Kotlin/JVM), JavaScript (Kotlin/JS) і рідний код (Kotlin/Native).

Kotlin був розроблений і розроблений JetBrains, чеською компанією, відомою своєю популярною IDE - IntelliJ IDEA. Команда Google Android нещодавно оголосила, що офіційно додає підтримку програмування Kotlin. Певні API Android, такі як наприклад Android KTX, специфічні для Kotlin, але більшість з них написані на Java і можуть бути викликані з Java або Kotlin. Сумісність Kotlin з Java робить цю мову такою популярною серед Android-розробників. Це означає, що є можливість викликати код Java з Kotlin і навпаки, використовуючи всі наявні бібліотеки Java. Популярність Kotlin призводить до кращого досвіду розробки на Android, але розробка фреймворків для Android продовжується з урахуванням як Kotlin, так і Java. Сумісність Kotlin з Java означає, що не потрібно використовувати тільки Kotlin у проекті. Можна також мати проекти з кодом Kotlin і Java

До травня 2017 року єдиними офіційно підтримуваними мовами програмування для Android були Java та C++. Google оголосила про офіційну підтримку Kotlin для Android на Google I/O 2017, і, починаючи з Android Studio 3.0, Kotlin вбудований у набір інструментів розробки Android. Kotlin можна додати до попередніх версій Android Studio за допомогою плагіна. [14].

Результати порівняння розглянутих мов програмування за певними критеріями наведені у таблиці 3.1.

Таблиця 3.1 – Порівняльні характеристики мобільних додатків

Мова програмування / Критерій	Java	Kotlin
Об’єктно-орієнтованість	+	+
Null-безпека	-	+
Extensions	-	+
Можливість розробки для ОС Android	+	+
Підсумковий результат	2	4

Отже, можна зробити висновок, що мова програмування Kotlin найкраще підходить серед порівнюваних мов, так як задовольняє усі потреби необхідні для розробки мобільного Android-додатку «MathForKids».

3.2. Обґрунтування вибору інтерфейсу

Для того, щоб відкрити улюблену гру на комп’ютері, використовуємо мишку, щоб переключити канал на телевізорі потрібен пульт, а для підтвердження замовлення і інтернет-магазині натискаємо відповідну кнопку. Все це є прикладами різних видів інтерфейсів, які допомагають людям взаємодіяти з пристроями.

Інтерфейс – це місце, в якому з’єднуються два функціональні об’єкти. Або простіше кажучи, це «міст» між користувачем і програмою. Це інструменти взаємодії, з-за допомогою якої, одна система налаштовую контакт з іншою.

Інтерфейс допомагає віддавати команди програмам і пристроям. Вони в свою чергу роблять відповідні дії і видають відповідь. Ось які задачі допомагають виконувати інтерфейси:

– робити ввід команд;

– отримувати відповідь у зрозумілій для користувача формі, наприклад текст, зображення чи звук;

– обмінюватись інформацією між пристроями і програмами;

– взаємодія людини в операційною системою;

– отримання інформації про помилку і можливих варіантах їх виправлення.

Інтерфейси бувають різних типів. По-перше, системи які взаємодіють між собою мають відмінність по характеру реалізації. По-друге, вони відрізняють по характеру можливостей інтерфейсів. Одні, наприклад, дають можливість доступу до програмної частини пристрою, але щоб їх використовувати потрібні спеціальні навички. А інші – можуть бути зручними у використанні, але функціонал який вони надають є досить обмеженим.

Інтерфейс користувача – це тип інтерфейсу, який призначений для організації взаємодії між людиною і програмним забезпеченням. Взаємодіяти з операційними системами і програмами, які знаходять під її керуванням можна саме з-за допомогою цього типу інтерфейсу. Інструментами, для реалізації інтерфейсу користувача найчастіше є наступні:

– клавіатура;

– комп’ютерна мишка;

– джойстик;

– дисплей.

При цьому, в залежності від операційної системи інтерфейс користувача може бути реалізованим в декількох формах. Найпопулярнішим є графічний інтерфейс [15].

Графічний інтерфейс – це система взаємодії, яка надає доступ користувачам до системних об’єктів та інструментів управління шляхом візуалізації інформації, тобто через графічні елементи зображені на моніторі. Найчастіше, використовуються програмні вікна для екранного відображення команд і результатів їх виконання. Елементи управління, всередині цих програмних вікон, відображаються в наступних формах:

– значки;

– поля вводу і виводу;

– кнопки;

– зображення;

– меню;

– список.

З-за допомогою комп’ютерної миші чи сенсорного дисплею відбувається управління вище вказаними елементами. Так як різні програми використовують схожі візуальні елементи, то процес ознайомлення нових програм для користувача є максимально спрощеним [16].

Інший тип інтерфейсу це текстовий інтерфейс, який є системою для взаємодії людини з комп’ютером чи іншим пристроєм за допомогою буквено-числових виразів. Він характеризується низькою потребою в ресурсах системи і високою швидкістю відповіді. Але такий тип інтерфейсу має свої недоліки. Один з яких, що його можуть використовувати лишу підготовлені користувачі, які мають спеціальні навички [17].

Отже, для написання даного мобільного додатку було обрано графічний інтерфейс користувача, адже при реалізації програми буде використано багато графічних елементів, для кращої взаємодії дитини з смартфоном.

3.3. Розробка інтерфейсу

Розробка інтерфейсу програми є важливим етапом у розробці мобільного додатку. При написанні мобільного додатку «MathForKids» було використано чимало різних типів View кожне з яких має своє призначення і свої функції. Програма побудована за принципом Single-Activity Architecture, тобто було створено одне activity, яке, в свою чергу, вміщує в собі фрагменти, навігація між якими здійснювалась за допомогою Navigation Graph.

Запускаючи додаток, користувач переходить на Splash screen, який зображено на рисунку 3.1, в якому бачить два елементи: ImageView і TextView. ImageView використовується для відображення зображень на екрані, а TextView в свою чергу призначений для виведення тексту.

Рисунок 3.1 – Приклад роботи Splash screen

Далі користувач переходить на головний екран, який зображений на рисунку 3.2, на якому він бачить набір view у формі «плиток», які мають можливість прокручуватись по горизонталі. Для цього було використано RecyclerView.

Рисунок 3.2 – Приклад роботи головного екрану програми

У користувача є можливість обрати дію в прикладах, яку він хотів би попрактикувати, а також рівень складності цих прикладів (див.рис.3.3). Також є можливість обирати розв’язування задач та рівень їх складності. При виборі рівня складності завдання, було використано таку view як button, натискання кожної з яких передає в аргументах, яка складність прикладів була обрана. Тому, наприклад, коли користувач натискає кнопку «Легко», передається рівень складності easy, і на наступному екрані приклади генеруються відповідно до обраної складності.

Рисунок 3.3 – Приклад роботи екрану вибору рівня складності прикладів

Обравши дію для прикладів, користувач переходить до самого екрану з прикладом (див.рис.3.4). На цьому фрагменті, користувач бачить такі UI-елементи як: Button, TextView та ImageView Окрім цих UI-елементів, на екрані також реалізований таймер, який починає свій відлік з 40 секунд. Це час, за який користувач повинен розв’язати якомога більше прикладів правильно, адже за це нараховуються монети, а за кожний неправильний розв’язок монети віднімаються.

Рисунок 3.4 – Приклад роботи екрану з прикладами

Програма має цікавий для дитини інтерфейс. Коли користувач відповідає правильно, тваринка інформує його, що приклад розв’язано правильно, і генерується новий приклад. Якщо ж приклад було розв’язано неправильно, тваринка теж в цьому випадку сповістить користувача про помилку, а також буде віднято одну монетку з рейтингу досягнень. Для демострації поточної кількості монеток, було використано два UI-елементи: TextView який відповідав за відображення кількості зароблених монет, і ImageView який мав функцію показувати зображення монетки.

Кількість монеток, яку користувач заробляє протягом гри зберігаються навіть після повного закриття додатку. Це реалізовано, з-за допомогою SharedPreferences, яке дає можливість постійно зберігати невеликі об’єми даних.

Коли таймер закінчиться, на екрані з’явиться діалогове вікно, в якому надається інформація про те, скільки було правильних відповідей і відповідно скільки монеток було зароблено користувачем (див.рис.3.5).

Рисунок 3.5 – Приклад роботи діалогового вікна з результатом

Код виклику діалогового вікна через viewModel даного класу продемонстровано на рисунку 3.6.

```
activity?.runOnUiThread {
    binding.progressbar.progress = 0
    "Твій результат: {viewModel.score} Бажаю успіхів!"
        ?.let { String.format(it) }?.let { viewModel.showDialog(context, it) }
}
```

Рисунок 3.6 – Код виклику діалогового вікна

Реалізація методу showDialog(), який у аргументи приймає context і рядок типу String, який буде відображатись у діалоговому вікні, прописана у базовій ViewModel, і код реалізації зображено на рисунку 3.7.

```
fun showDialog(context: Context?, text: String) {
    val dialog = context?.let { Dialog(it) }
    dialog?.let { dlg ->
        dlg.apply {  this: Dialog
            requestWindowFeature(Window.FEATURE_NO_TITLE);
            setCancelable(false)
            setContentView(R.layout.dialog_first_speach_layout)
            window?.setBackgroundDrawable(ColorDrawable(Color.TRANSPARENT))
            findViewById<TextView>(R.id.tv_main_text).text = text
            findViewById<Button>(R.id.speach_dialog_ok_button).setOnClickListener {  it: View!
                dismiss()
                actionBackToMainScreen()
            }
            show()
        }
    }
}
```

Рисунок 3.7 – Код реалізації діалогового вікна

Таймер, який рахує кількість секунд на виконання завдання, був реалізований за допомогою coroutines. Код, для реалізації цього таймера зображено на рисунку 3.8. Coroutine – це модель дизайну одночасності, який можна використовувати на Android для спрощення коду, що виконується асинхронно. Coroutine були додані до Котліна у версії 1.3 і засновані на встановлених концепціях інших мов [2].

```
scope.launch {  this: CoroutineScope
    val tickSeconds = 0
    for (second in 40 downTo tickSeconds) {
        activity?.runOnUiThread {
            binding.apply {  this: ExerciseFragmentBinding
                progressbar.progress = second
                tvSecondsLeft.text =
                    "Залишилось: {second}"
                        ?.let { String.format(it) }
            }
        }
        delay( timeMillis: 1000)
    }
```

Рисунок 3.8 – Код реалізації таймера

Coroutine допомагає керувати тривалими завданнями на Android, які в іншому випадку можуть заблокувати основний потік і змусити робочу програму не відповідати. Понад 50% професійних розробників, які використовують Coroutines, повідомили про підвищення продуктивності праці з використанням таймерів контролю робочого часу. Coroutines – це рекомендоване рішення від Google для асинхронного програмування на Android [18].

3.4. Розробка програми

Під час розробки Android-додатку «MathForKids» було створено багато змінних, методів і класів для реалізації даної програми. Для viewModel-ей екранів множення, ділення, додавання та віднімання було реалізовано базову viewModel, яку відповідно інші наслідували. Ця viewModel містила в собі ряд змінних та методів.

Перший метод є абстракним і має назву actionBackToMainScreen(), тобто даний метод реалізовується в кожному класі по різному. Але з назви методу зрозуміло, що при виклику даного методу, користувач повернеться на головний екран. Але так як користувач переходить з різних екранів, саме тому було вирішено зробити цей метод абстрактним. Наприклад, у класу DivideViewModel даний метод повертає користувача з DivideFragment на головний екран програми. Код реалізації даного методу у класі DivideViewModel зображено на рисунку 3.9.

```
override fun actionBackToMainScreen() = binding.root.findNavController()
    .navigate(R.id.action_divideFragment_to_mainScreenFragment)
```

Рисунок 3.9 – Код реалізації методу actionBackToMainScreen() у класі DivideViewModel

Наступний метод є також абстрактний і назва даного методу є makeMathExerciseModel(). Цей метод приймає в параметри значення типу

String з рівнем складності задачі. Даний метод створений для генерації нового прикладу, відповідно до обраної дії та обраного рівня складності. Даний метод генерує саме приклад, і повертає значення у вигляді моделі, яка містить два числа прикладу, ще одне число яке є правильною відповіддю і масив неправильних трьох чисел. Код даного методу, який генерує приклад для дії додавання і віднімання зображено на рисунку 3.10.

```
override fun makeMathExerciseModel(
    level: String?
): MathExerciseModel {
    if (this.level == null) {...}
    if (level.equals(Constants.HARD_CHAR)) {...}
    generatePair(this.level)
    if (Random.nextBoolean()) {
        answer = firstValue + secondValue
        when (this.level) {
            Constants.EASY_CHAR -> {...}
            Constants.MEDIUM_CHAR -> {...}
            Constants.HARD_CHAR -> {...}
        }

        max = answer + firstValue
        while (max < 5) {...}
        binding.exercise.symbol.text = "+"
    } else {
        if (firstValue < secondValue) {...}
        answer = firstValue - secondValue
        max = firstValue + secondValue
        while (max < 5) {
            max++
        }
        binding.exercise.symbol.text = "-"
    }
    val min = 1
    while (wrongAnswers.size < 3) {...}
    return MathExerciseModel(firstValue, secondValue, answer, wrongAnswers)
}
```

Рисунок 3.10 – Код методу makeMathExerciseModel() для класу PlusMinusViewModel

Наступний метод є загальним для всіх класів-наслідників і назва даного методу generateNewExercise(). Даний метод приймає в аргументах значення рівня складності прикладів. Даний метод відповідає саме за генерацію позиції правильної відповіді, встановлення clickListener-ів для кнопок,

сповіщення користувача про правильне чи навпаки неправильне розв’язання прикладів, а також встановлення всіх чисел на свої місця відповідно до згенерованих позицій. Код цього методу зображено на рисунку 3.11.

```
fun generateNewExercise(level: String? = null) {
    val mathExercise = makeMathExerciseModel(level)
    val indexOfTrueAnswer: Int = Random.nextInt( from: 0,  until: 3)
    val arrayOfButtons = ArrayList<Button>()
    updateScore(score)
    arrayOfButtons.apply {  this: ArrayList<Button>
        add(binding.setOfFourButtons.button1)
        add(binding.setOfFourButtons.button2)
        add(binding.setOfFourButtons.button3)
        add(binding.setOfFourButtons.button4)
        get(indexOfTrueAnswer).apply {  this: Button
            text = mathExercise.answerValue.toString()
            setOnClickListener {...}
        }
        var counter = 0
        for (i in 0..3) {
            if (indexOfTrueAnswer != i) {
                arrayOfButtons[i].apply {...}
                counter++
            }

        }
    }

    binding.exercise.apply {  this: ExerciseBinding
        firstValue.text = mathExercise.firstValue.toString()
        secondValue.text = mathExercise.secondValue.toString()
    }
}
```

Рисунок 3.11 – Код методу generateNewExercise()

Для оновлення рахунку було використано клас, який зберігає дані реалізовує паттерн Observable – LiveData. Дане поле, яке має назву «scoreLiveData» і зберігає числове значення типу Int, при зміні якого, змінюється і значення на екрані. Код реалізації підписки на зміни даного поля зображено на рисунку 3.12.

```
viewModel.scoreLiveData.observe(viewLifecycleOwner) {  it: Int!
    activity?.runOnUiThread {
        binding.tvScore.text = activity?.getString(R.string.score, it)
    }
}
```

Рисунок 3.12 – Код реалізації підписки на зміни поля «scoreLiveData»

Для оновлення даного поля, клас базової viewModel містить метод з відповідною назвою updateScore(). Даний метод має єдину функцію – оновлювати значення зароблених монет користувачем. Метод updateScore() має приватний модифікатор доступу, адже викликається лиш у класі в якому створений, тобто у BaseViewModel. Код реалізації даного методу зображено на рисунку 3.13.

```
private fun updateScore(score: Int) = scoreLiveData.postValue(score)
```

Рисунок 3.13 – Код реалізації методу updateScore()

Після кожного разу, коли сплив час на таймері програма перевіряє, чи заробив користувач хоча б одну монету, і якщо це правда, програма додає ці зароблені монети до числа загальних і зберігає їх у місці їх постійного зберігання SharedPreferences. Код реалізації перевірки та додавання монет до місця їх постійного збереження зображено на рисунку 3.14.

```
if (viewModel.score > 0) {
    balance += viewModel.score
    sharedPref.edit().putInt(Constants.BALANCE, balance).apply()
}
```

Рисунок 3.14 –Код реалізації перевірки та додавання монет до SharedPreferences

Для зчитування задач з файлу, насамперед визначається ім’я файлу, відповідно до його складності. Для визначення, імені файлу, з якого будуть зчитуватись, а потім генеруватись задачі, викликається метод getFileName(), код якого зображено на рисунку 3.15.

```
fun getFileName(level: String?): String = when (level) {
    Constants.EASY_CHAR -> PREMIUM_ZADACHA_EASY
    Constants.MEDIUM_CHAR -> PREMIUM_ZADACHA_MEDIUM
    Constants.HARD_CHAR -> PREMIUM_ZADACHA_HARD
    else -> PREMIUM_ZADACHA_MEDIUM
}
```

Рисунок 3.15 – Код реалізації методу getFileName()

Після того як було визначено ім’я файлу, з якого будуть зчитуватись задачі, відбувається зчитування файлу в масив задач. Код реалізації цього зчитування зображено на рисунку 3.16.

```
level = arguments?.getString( key: "level")
fileName = viewModel.getFileName(level)
file = context?.let { context ->
    context.assets.open(fileName).bufferedReader().use {  it: BufferedReader
        it.readText()
    }
}.toString()
arrayListZadach = file.split( ...delimiters: "***").map {  it: String
    it.split( ...delimiters: "|")
}.map {  it: List<String>
    Zadacha(it[0], it[1].trim().toInt())
}
```

Рисунок 3.16 – Код реалізації зчитування задач з файлу в масив

Всі вище перелічені та описані методи, поля та класи є важливою частиною розробленої програми. Адже без правильної їх реалізації робота

головного функціоналу даного мобільного додатку була б не зовсім коректною.

3.5. Висновки

У даному розділі було проведено аналіз мов програмування які призначення для розробки мобільних додатків Kotlin та Java. І після порівнянь всіх переваг і недоліків цих мов програмування, було прийнято рішення використовувати мову програмування Kotlin для розробки мобільного Android-додатку «MathForKids». Також було проведено аналіз та розроблено дизайн інтерфейсу, який є орієнтованим на користувачів меншого віку. Також було описано та продемонстровано код методів, полів і класів, які є найважливішими у програмі.

4 ТЕСТУВАННЯ ПРОГРАМИ

4.1 Аналіз методів тестування програмного забезпечення

Перше програмне забезпечення розроблялося або для потреб військової сфери, або з ціллю наукових досліджень. Тестування такого типу програмного забезпечення проводилось строго із записом усього тестового процесу. Записувались також тестові дані, а також отримані результати. Процес тестування починався після завершення процесу кодування, і в більшості випадків проводився тими ж самим спеціалістами.

Протягом процесу тестування, слід перевіряти не лише саму зібрану програму, а також потрібно перевірити архітектуру, вимоги і самі тести. «Традиційне» тестування, стосувалося лише скомпільованої та готової програми і існувало таке тестування до початку 80-х років. В подальшому, було вирішено виконувати тестування на різних стадіях розробки програмного забезпечення. Саме це дозволило знаходити помилки раніше, і тим самим скорочувались терміни написання проекту та бюджет його розробки.

Існує багато визначень слова «тестування», але одне з них говорить, що це є визначення відповідності об’єкту тестування відповідно до заданих специфікацій та характеристик, та опрацювання програмою послідовності різних наборів тестів з наперед прописаними результатами. Тобто це перевірка відповідності продукту відповідно до початкових специфікацій, які були задані технічним завданням. Для того, аби охопити найрізноманітніші випадки, підбираються такі ж найрізноманітніші тести.

Якщо ж дивитись на поняття «тестування» в ширшому сенсі, то можна сказати, що це є процес експериментального дослідження функціональності продукту. Кожне тестування має дві дійових особи, якими є суб’єкт і об’єкт тестування. Суб’єктом тестування є «тестувальник», тобто людина, в чиї обов’язки входить виконання тестування. Об’єктом тестування являється

досліджувана система. У ролі досліджуваної системи, можу бути як весь готовий продукт, так і окремі його частини [19].

Наявність або відсутність доступу до коду програмного забезпечення є однією з основних класифікацій тестування. Згідно цього типу параметру, виділяють наступні типи тестування:

– white-box testing;

– black-box testing;

– grey-box testing.

У методі тестування «white-box testing» спеціаліст, який тестує програмне забезпечення, має доступ до програмного коду продукту, який тестує. Це необхідно для того, щоб зрозуміти роботу даного продукту від початку і до кінця. До переваг такого тестування можна віднести наступні

– до створення користувацького інтерфейсу вже може проводитися тестування програмного забезпечення;

– тестувальник допомагає усувати помилки, ще на стадії розробки, адже глибше інтегрований в розробку додатку.

А от щодо недоліків даного типу тестування, то вони наступні:

– необхідні висококваліфіковані спеціалісти, які мають навички написання коду і його структури;

– вартість є надто високою;

– на тестування великих додатків можуть бути потрачені тижні або й навіть місяці, адже необхідне глибоке занурення в код, а це вимагає більше часу.

У методі тестування «black-box testing» спеціаліст, який тестує програмне забезпечення, навпаки немає доступу до програмного коду продукту, який тестує. Більшість видів тестування працюють за цим методом, адже в цьому методі тестувальник взаємодіє з продуктом і перевіряє його так, як і реальні користувачі під час реального використання. Серед переваг цього типу тестування є наступні:

– темп роботи пришвидшується, адже розробник і тестувальник незалежні один від одного;

– тестувальнику достатньо розбиратись в специфікаціях і не потрібно вміти писати код;

– написання тест-кейсів можна починати одразу після отримання специфікацій.

Попри меншу вартість та вищу швидкість даний тип має наступні недоліки:

– тест-кейсів не може бути без специфікацій;

– важко спроектувати і написати всі тест-кейси;

– коректність роботи усього додатку перевірити важко, адже щось точно буде знайдено користувачем.

Метод тестування «grey-box testing» є комбінацією двох попередніх і полягає в тому, що у тестувальника є доступ до певної частини коду, а до іншої – немає. Зазвичай, про методи «black-box» та «white-box» мова йде відносно окремих частин продукту, тобто в загальному продукт тестується за методом «grey-box» [20].

Отже, після аналізу кожного з методів тестування, визначення їхніх переваг і недоліків, було прийнято рішення використати метод «black-box testing». Використання саме цього методу, дасть можливість виявити помилки у функціонуванні мобільного додатку чи в роботі якогось з алгоритмів вже на ранній стадіїї.

4.2 Тестування розробленого програмного продукту

Тестування Android-додатку «MathForKids» використовуючи методику «black-box testing» означає перевірку правильності функціонування додатку при виконанні основних його алгоритмів та порівняння наявного результату з очікуваним. Саме такі алгоритми використання програмного

додатку називають тест-кейсами. Для тестування розробленого Android-додатку «MathForKids» було розроблено такі тест-кейси:

Тест-кейс №1 – Запуск діалогового вікна при запуску програми:

1. Відкрити мобільний додаток «MathForKids».

2. Отримати у відповідь відкриття діалогового вікна після закриття екрану «Splash screen».

Очікуваним результатом даного тест-кейсу є отримання від мобільного додатку відкритого діалогового вікна на початку запуску програми. Результат виконання тест-кейсу наведено на рисунку 4.1.

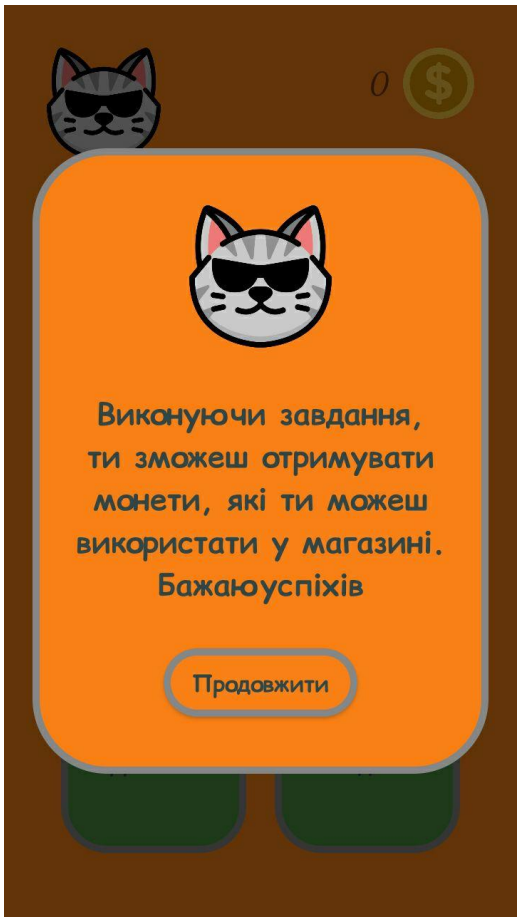


Рисунок 4.1 – Результат виконання тест-кейсу №1

Наявний результат виконання повністю відповідає очікуваному, отже тест-кейс №1 вважається успішно пройденим.

Тест-кейс №2 – Список кнопок показується в два рядки і має можливість прокручуватись по горизонталі:

1. Відкрити мобільний додаток «MathForKids».
2. Натиснути кнопку «Продовжити» у діалоговому вікні.
3. Отримати відповідь програми.

У даному тест-кейсі, очікується відображення кнопок у два рядки з можливістю прокручувати по горизонталі. Результат виконання даного тест-кейсу зображено на рисунку 4.2.

Рисунок 4.2 – Результат виконання тест-кейсу №2

У результаті виконання тест-кейсу №2, мобільним додатком було показано кнопки, у вигляді двох рядків. Також є можливість прокручувати зміст. Тому, наявний результат відповідає очікуваному і можна вважати, що тест-кейс №2 успішно пройдено.

Тест-кейс №3 – Можливість обрати рівень виконання прикладів «Додавання і віднімання»:

1. Відкрити мобільний додаток «MathForKids».

2. Натиснути кнопку «Продовжити» у діалоговому вікні.

3. Натиснути кнопку «Додавання і віднімання».

4. Очікувати відкриття екрану з можливістю обрати важкість прикладів.

У даному тест-кейсі, очікується відображення трьох кнопок, які мають назви «Важкий», «Середній» та «Легкий», і означають відповідний рівень важкості прикладів. Результат виконання даного тест-кейсу зображено на рисунку 4.3.

Рисунок 4.3 – Результат виконання тест-кейсу №3

У результаті виконання тест-кейсу №3, мобільним додатком було показано три кнопки, з можливістю вибрати рівень важкості. Тому, наявний

результат відповідає очікуваному і можна вважати, що тест-кейс №3 успішно пройдено.

Тест-кейс №4 – Генерування прикладів «Додавання і віднімання» легкого рівня складності:

1. Відкрити мобільний додаток «MathForKids».
2. Натиснути кнопку «Продовжити» у діалоговому вікні.
3. Натиснути кнопку «Додавання і віднімання».
4. Обрати рівень складності «Легкий».
5. Очікувати відкриття екрану з можливістю обрати важкість прикладів.

У даному тест-кейсі, очікується відкриття екрану «Додавання і віднімання» і генерація прикладу легкої складності, тобто сума чисел є не більше 10. Результат виконання даного тест-кейсу зображено на рисунку 4.4.

Рисунок 4.4 – Результат виконання тест-кейсу №4

У результаті виконання тест-кейсу №4, мобільним додатком було згенеровано приклад легкого рівня складності, адже сума чисел не є більшою за 10. Тому, наявний результат відповідає очікуваному і це означає, що тест-кейс №4 успішно пройдено.

Тест-кейс №5 – Сповіщення користувача про вибір правильної відповіді:

1. Відкрити мобільний додаток «MathForKids».

2. Натиснути кнопку «Продовжити» у діалоговому вікні.

3. Натиснути кнопку «Додавання і віднімання».

4. Обрати рівень складності «Легкий».

5. Розв’язати приклад правильно.

6. Очікувати сповіщення від програми про правильно розв’язаний приклад.

У даному тест-кейсі, очікується відкриття екрану «Додавання і віднімання» і після правильного вирішення прикладу, користувачу з’являється повідомлення «Молодець» на зеленому фоні і добавляється плюс один до рахунку. Результат виконання даного тест-кейсу зображено на рисунку 4.5.

Рисунок 4.5 – Результат виконання тест-кейсу №5

У результаті виконання тест-кейсу №5, після правильно розв’язаного прикладу, на екрані з’явилося повідомлення яке сповіщає про правильність розв’язання і добавився плюс один до рахунку. Тому, наявний результат відповідає очікуваному і це означає, що тест-кейс №5 успішно пройдено.

Тест-кейс №6 – Сповіщення користувача про вибір неправильної відповіді:

1. Відкрити мобільний додаток «MathForKids».

2. Натиснути кнопку «Продовжити» у діалоговому вікні.

3. Натиснути кнопку «Додавання і віднімання».

4. Обрати рівень складності «Легкий».

5. Розв’язати приклад неправильно.

6. Очікувати сповіщення від програми про неправильно розв’язаний приклад.

У даному тест-кейсі, очікується відкриття екрану «Додавання і віднімання» і після неправильного вирішення прикладу, користувачу з’являється повідомлення «Подумай краще» на червоному фоні і

віднімається один бал з рахунку. Результат виконання даного тест-кейсу зображено на рисунку 4.6.

Рисунок 4.6 – Результат виконання тест-кейсу №6

У результаті виконання тест-кейсу №6, після неправильно розв’язаного прикладу, на екрані з’явилося повідомлення яке сповіщає про неправильність розв’язання і віднявся один бал з рахунку. Тому, наявний результат відповідає очікуваному і це означає, що тест-кейс №6 успішно пройдено.

Тест-кейс №7 – Сповіщення користувача про закінчення часу на виконання прикладів:

1. Відкрити мобільний додаток «MathForKids».
2. Натиснути кнопку «Продовжити» у діалоговому вікні.
3. Натиснути кнопку «Додавання і віднімання».
4. Обрати рівень складності «Легкий».
5. Розв’язати 15 прикладів правильно протягом відведеного часу.
6. Очікувати сповіщення від програми про закінчений час і результат 15.

У даному тест-кейсі, очікується відкриття діалогового вікна, яке сповіщає про закінчення часу, і показує результат за цей раунд, у поточному

тест-кейсі це 15. Результат виконання даного тест-кейсу зображено на рисунку 4.7.

Рисунок 4.7 – Результат виконання тест-кейсу №7

У результаті виконання тест-кейсу №7, мобільний додаток сповістив користувача про закінчення часу, тобто було відображено вікно, яке показало результат 15, як і очікувалось на початку. Тому, наявний результат відповідає очікуваному і це означає, що тест-кейс №7 успішно пройдено.

Тест-кейс №8 – Сповіщення користувача про правильно розв’язану задачу:

1. Відкрити мобільний додаток «MathForKids».

2. Натиснути кнопку «Продовжити» у діалоговому вікні.

3. Натиснути кнопку «Задачі».

4. Обрати рівень складності «Легкий».

5. Розв’язати задачу правильно.

6. Очікувати сповіщення від програми про правильно розв’язану задачу.

У даному тест-кейсі, очікується відкриття екрану «Задачі» і після правильного вирішення задачі, користувачу з’являється повідомлення

«Молодець» на зеленому фоні і добавляється плюс один до рахунку, а далі на екрані показується наступна задача. Результат виконання даного тест-кейсу зображено на рисунку 4.8.

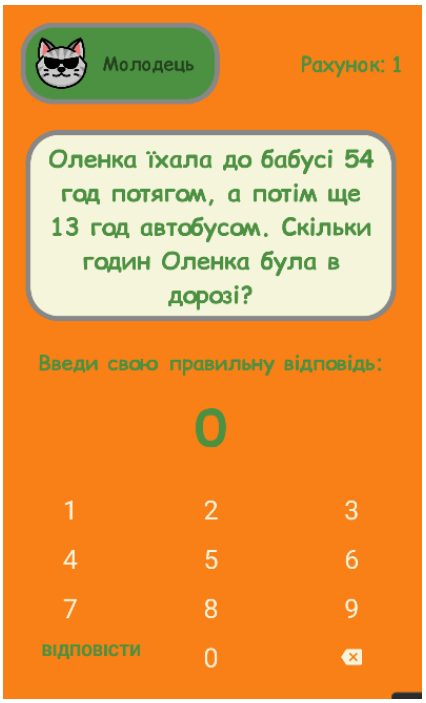


Рисунок 4.8 – Результат виконання тест-кейсу №8

У результаті виконання тест-кейсу №8, після правильно розв’язаної задачі, на екрані з’явилося повідомлення яке сповіщає про правильність розв’язання і добавився плюс один до рахунку. А також нова задача була показана на екрані. Тому, наявний результат відповідає очікуваному і це означає, що тест-кейс №8 успішно пройдено.

Тест-кейс №9 – Сповіщення користувача про неправильно розв’язану задачу:

1. Відкрити мобільний додаток «MathForKids».
2. Натиснути кнопку «Продовжити» у діалоговому вікні.
3. Натиснути кнопку «Задачі».
4. Обрати рівень складності «Легкий».
5. Розв’язати задачу неправильно.
6. Очікувати сповіщення від програми про неправильно розв’язану задачу та віднімання одного балу з рахунку.

У даному тест-кейсі, очікується відкриття екрану «Задачі» і після неправильного вирішення задачі, користувачу з’являється повідомлення «Подумай краще» на червоному фоні і віднімається один бал з рахунку. а також задача на екрані змінюватись не повинна. Результат виконання даного тест-кейсу зображено на рисунку 4.9.

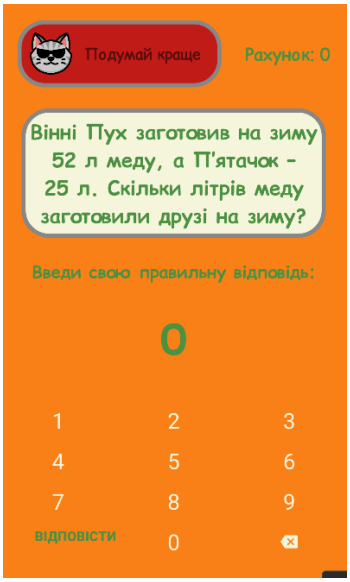


Рисунок 4.9 – Результат виконання тест-кейсу №9

У результаті виконання тест-кейсу №9, після неправильно розв’язаної задачі, на екрані з’явилося повідомлення яке сповіщає про неправильність розв’язання і віднявся один бал з рахунку. А також, задача на екрані залишилась та сама. Тому, наявний результат відповідає очікуваному і це означає, що тест-кейс №9 успішно пройдено.

Тест-кейс №10 – Збереження балансу монет після повного закриття додатку:

1. Відкрити мобільний додаток «MathForKids».
2. Натиснути кнопку «Продовжити» у діалоговому вікні.
3. Натиснути кнопку «Множення».
4. Розв’язати 10 прикладів правильно.
5. Перейти на головний екран.
6. Повністю закрити додаток.
7. Відкрити мобільний додаток «MathForKids».

8. Поточний баланс монет повинен бути 10.

У даному тест-кейсі, очікується зберігання балансу монет, відповідно до кількості розв’язаних прикладів. Для даного тест-кейсу це 10 монет. Результат виконання даного тест-кейсу зображено на рисунку 4.10.

Рисунок 4.10 – Результат виконання тест-кейсу №10

У результаті виконання тест-кейсу №10, розв’язання десяти прикладів і повного закриття додатку, а потім відкривши додаток баланс монет був такий, який був до закриття, тобто 10. Тому, наявний результат відповідає очікуваному і це означає, що тест-кейс №10 успішно пройдено.

4.3 Розробка інструкції користувача

Для запуску мобільного додатку необхідно запустити файл MathForKids, зображення іконки в головному меню має вигляд, як зображено на рисунку 4.11.

Рисунок 4.11 – Зображення іконки файлу програми в головному меню

Відкриваючи програму, користувач переходить на головний екран, на якому є список в два ряди, з можливістю прокручувати його горизонтально. Якщо користувач обрав «Додавання і віднімання» або «Задачі», тоді відкриється екран з можливість обрати рівень складності майбутніх прикладів чи задач.

Після вибору складності, відкривається наступний екран на якому власне генеруються приклади, або беруться задачі відповідно до обораного рівня. Якщо ж користувач обирає іншу дію, він образу переходить на екран з прикладами.

На екрані з прикладами, також присутній таймер, який обмежує користувача в часі на 40 секунд. Тобто, це означає, що за даний час користувач має розв’язати якомога більше прикладів. Розв’язавши правильно приклад, чи задачу, користувач отримує сповіщення про це. Тобто, коли приклад, або задачу розв’язано правильно, на екрані з’являється сповіщення, яке зображено на рисунку 4.12, з текстом «Молодець» на зеленому фоні.

Рисунок 4.12 – Сповіщення про правильно розв’язаний приклад або задачу

Коли навпаки, приклад, або задачу розв’язано неправильно, на екрані з’являється сповіщення, яке продемонстровано на рисунку 4.13, з текстом «Подумай краще» на червоному фоні.

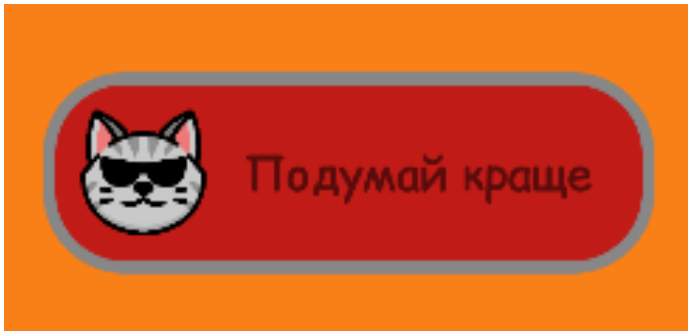


Рисунок 4.13 – Сповіщення про неправильно розв’язаний приклад або задачу

При чому, при кожній правильній плюс одна монета до рахунку, і відповідно при неправильній одна монета забирається.

Після закінчення таймера, користувач бачить діалогове вікно, яке сповіщає про закінчення часу. Дане діалогове вікно містить інформацію про те, скільки монет було зароблено протягом попереднього раунду.

4.4 Висновки

Для того, щоб провести тестування мобільного Android-додатку «MathForKids», було обрано методику «black-box testing», яка передбачає перевірку функціональності додатку за допомогою тест-кейсів. У результаті тестування було доведено, що даний програмний продукт є повністю працездатним та відповідає поставленому технічному завданню. Також було

розроблено інструкцію користувача, з-за допомогою якої користувачу буде простіше розпочати використання додатку та швидше ознайомитись зі способами використання функціоналу даного додатку.

ВИСНОВКИ

У бакалаврській дипломній роботі було розроблено мобільний програмний додаток для вивчення математики дітьми молодшої школи «MathForKids», шляхом розв’язування прикладів та задач з різними діями і різними рівнями складності.

Було проведено аналіз предметної області даного питання. Також, був проведений аналіз аналогів розроблюваного програмного продукту для

вивчення математики дітьми молодшої школи, а також застосування цих знань на практиці. В результаті порівняння аналогів із власним програмним продуктом, було доведено доцільність розробки власного програмного продукту для вивчення дітьми математики. Також було виконано постановку задач розробки програмного продукту.

Було розроблено методи та модель роботи програми, розроблено алгоритм побудови головного вікна мобільного додатку. Також, було розроблено покроковий алгоритм роботи програми, алгоритм генерації нового прикладу та алгоритм зчитування задач з файлу. Крім того, було розроблено блок-схеми цих алгоритмів.

Було проведено варіантний аналіз та обґрунтування вибору засобів реалізації програмного продукту. В результаті даного аналізу, в якості мови програмування було обрано мову програмування Kotlin, а в якості середовища розробки Android Studio.

В процесі розробки було обґрунтовано та реалізовано інтерфейс мобільного додатку «MathForKids». Також, було описано програмну реалізацію основних класів програми.

Було проаналізовано основні методи тестування програмного забезпечення, та в результаті даного аналізу було обрано методику «black-box testing». Тестування, яке проводилось за допомогою тест-кейсів, довело повну працездатність розробленого програмного додатку та його відповідність поставленому технічному завданню. Крім того, було розроблено інструкцію користувача.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Мобільне навчання як сучасна технологія в освіті [Електронний ресурс] – Режим доступу до ресурсу: https://naurok.com.ua/mobilne-navchannya-yak-suchasna-tehnologiya-v-osviti-239094.html.

2. Войтко В.В. Розробка мобільного додатку “MATH FOR KIDS”, спрямованого на вивчення математики дітьми молодшої школи / Войтко В.В., Круподьорова Л. М., Денисюк А. В., Гаврилюк О. В., Барчук Н. Є., Деда В.П. Матеріали LI Науково-технічна конференція факультету інформаційних технологій та комп'ютерної інженерії (2022) ВНТУ, Факультет інформаційних технологій та комп'ютерної інженерії, Секція програмного забезпечення. [Електронний ресурс] – Режим доступу до ресурсу: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2022/paper/view/15098/12734

3. Мобільні програми для освіти [Електронний ресурс] – Режим доступу до ресурсу: https://www.formacionyestudios.com/uk/aplicaciones-moviles-para-educacion.html.

4. 5 мобільних додатків для вивчення математики [Електронний ресурс]– Режим доступу до ресурсу: https://kidsvisitor.com/uk/news/136-5-mobilnikh-dodatkiv-dlia-vivchennia-matematiki/.

5. Build a Greater Understanding of Algebra [Електронний ресурс]– Режим доступу до ресурсу: https://dragonbox.com/products/algebra-12.

6. Kids Numbers [Електронний ресурс]– Режим доступу до ресурсу: https://play.google.com/store/apps/details?id=zok.android.numbers&hl=uk&gl=US

7. Operation Math: Addition, Subtraction [Електронний ресурс] – Режим доступу до ресурсу: https://www.commonsense.org/education/app/operation-math-addition-subtraction-multiplication-and-division.

8. Mathmateer [Електронний ресурс]– Режим доступу до ресурсу: https://learningworksforkids.com/playbooks/mathmateer/.

9. Geeksmath [Електронний ресурс]– Режим доступу до ресурсу: https://apkpure.com/ru/geeksmath/com.cloudcomputingcenter.matan.

10. Розробка мобільних додатків від А до Я: повний гайд [Електронний ресурс] – Режим доступу до ресурсу: https://dan-it.com.ua/uk/blog/rozrobka-mobilnih-dodatkiv-vid-a-do-ja-povnij-gajd/.

11. Kotlin coroutines on Android [Електронний ресурс] – Режим доступу до ресурсу: https://developer.android.com/kotlin/coroutines.

12. Що таке алгоритм? [Електронний ресурс] – Режим доступу до ресурсу: https://uk.education-wiki.com/7432851-what-is-an-algorithm.

13. What is Java? Definition, Meaning & Features of Java Platforms [Електронний ресурс] – Режим доступу до ресурсу: https://www.guru99.com/java-platform.html.

14. Kotlin overview [Електронний ресурс] – Режим доступу до ресурсу: https://developer.android.com/kotlin/overview.

15. Користувацький інтерфейс [Електронний ресурс] – Режим доступу до ресурсу: https://uk.upwiki.one/wiki/User_interface.

16. Графічний інтерфейс користувача [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Графічний_інтерфейс користувача

17. Текстовий інтерфейс користувача [Електронний ресурс] – Режим доступу до ресурсу: https://wblog.wiki/uk/Text_user_interface

18. Kotlin coroutines on Android [Електронний ресурс] – Режим доступу до ресурсу: https://developer.android.com/kotlin/coroutines.

19. Авраменко А.С., Авраменко В.С., Косенюк Г.В. Тестування програмного забезпечення. Навчальний посібник. – Черкаси: ЧНУ імені Богдана Хмельницького, 2017.

20. Типи тестування: пам'ятка початківцю [Електронний ресурс] – Режим доступу до ресурсу: https://training.epam.ua/#!/News/469?lang=ua.

ДОДАТКИ

Додаток А – Технічне завдання

Міністерство освіти і науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп’ютерної інженерії

УЗГОДЖЕНО
Директор Деражнянського ліцею №1
імені П. Стрілецького
Сідлецький В.М.
“____”_____2022 р.

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
Романюк О.Н.
“_25_”__березня__2022 р.

Технічне завдання

на бакалаврську дипломну роботу «Розробка мобільного програмного додатку для вивчення математики дітьми молодшої школи»

за спеціальністю

121 – Інженерія програмного забезпечення

Керівник бакалаврської дипломної роботи:

к.т.н., доц. В.В. Войтко

“_25_”__березня__2022 року.

Виконав:

студент гр. 3ПІ-18б В.П. Деда

“_25_”__березня__2022 року.

Вінниця – 2022 року

1. Найменування та галузь застосування

Бакалаврська дипломна робота: «Розробка мобільного програмного додатку для вивчення математики дітьми молодшої школи».

Галузь застосування – мобільні-технології.

2. Підстава для розробки.

Завдання на роботу, яке затверджене на засіданні кафедри програмного забезпечення – протокол №13 від «07» лютого 2022 р.

3. Мета та призначення розробки.

Метою роботи є підвищення можливостей навчального процесу в молодшій школі шляхом створення мобільного додатку для вивчення дітьми математики, що підвищить зацікавленість учнів навчальним процесом.

Призначення роботи – покращення процесу вивчення математики дітьми молодшої школи шляхом залучення до цього мобільних технологій.

4. Вихідні дані для проведення НДР

1. Мобільне навчання як сучасна технологія в освіті [Електронний ресурс] – Режим доступу до ресурсу: https://naurok.com.ua/mobilne-navchannya-yak-suchasna-tehnologiya-v-osviti-239094.html.
2. Мобільні програми для освіти [Електронний ресурс] – Режим доступу до ресурсу: https://www.formacionyestudios.com/uk/aplicaciones-moviles-para-educacion.html.
3. Розробка мобільних додатків від А до Я: повний гайд [Електронний ресурс] – Режим доступу до ресурсу: https://dan-it.com.ua/uk/blog/rozrobka-mobilnih-dodatkiv-vid-a-do-ja-povnij-gajd/.
4. Kotlin coroutines on Android [Електронний ресурс] – Режим доступу до ресурсу: https://developer.android.com/kotlin/coroutines.

5. Технічні вимоги

Вхідні дані – інформація від користувача про складність прикладів чи задач; вихідні дані – графічний інтерфейс із прикладами чи задачами відповідно до обраної складності.

6. Конструктивні вимоги.

Конструкція пристрою повинна відповідати естетичним та ергономічним вимогам, повинна 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 бакалаврської дипломної роботи;
- технічне завдання;
- лістинги програми.

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

Додаток Б – Акт впровадження (Деражнянський ліцей №1 імені П. Стрілецького)

АКТ

про впровадження результатів наукової роботи

«Розробка мобільного програмного додатку для вивчення математики дітьми молодшої школи»

студента ВНТУ Деди Владислава Петровича

Результати бакалаврської кваліфікаційної роботи «Розробка мобільного програмного додатку для вивчення математики дітьми молодшої школи» студента ВНТУ Деди В.П., що полягають в розробці:

- методу генерації нових прикладів відповідно до обраного рівня складності, який генерує приклади у певному діапазоні, та повністю відповідає обраному рівню складності;
- методу сповіщення користувача про правильність розв’язання прикладів, який сповіщає користувача про це прямо під час гри, при чому сповіщення мають різні кольори, щоб краще та швидше розуміти та відрізняти чи правильно приклад чи задача були вирішені. Тому сповіщення про правильне вирішення має зелений колір, а про не правильне – червоний;
- програмного додатку MathForKids, який призначений для вивчення математики дітьми молодшої школи, а також застосування цих знань на практиці впроваджено в Деражнянському ліцеї №1 імені П. Стрілецького.

Директор Деражнянського ліцею Сідлецький В. М.

№1 імені П. Стрілецького

Додаток В – Протокол перевірки на плагіат

ПРОТОКОЛ

ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Розробка мобільного програмного додатку для вивчення математики дітьми молодшої школи

Тип роботи: БДР

Підрозділ : кафедра програмного забезпечення, ФІТКІ

Науковий керівник: к.т.н., доц. каф. ПЗ Войтко В.В.

Оригінальність	97.3%
Схожість	2.7%

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.

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck

Автор роботи _____ Деда В.П

Керівник роботи _____ Войтко В.В.

Додаток Г – Лістинг програми

MainActivity.kt

```
package com.example.mathwithfinik

import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.view.WindowManager
import com.example.mathwithfinik.databinding.ActivityMainBinding
import com.example.mathwithfinik.ui.mainscreen.FirstSpeachDialog
import com.example.mathwithfinik.ui.mainscreen.MainScreenFragment

class MainActivity : AppCompatActivity() {
    private val binding: ActivityMainBinding by lazy {
        ActivityMainBinding.inflate(layoutInflater)
    }
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(binding.root)
        window.setFlags(
            WindowManager.LayoutParams.FLAG_FULLSCREEN,
            WindowManager.LayoutParams.FLAG_FULLSCREEN
        )
        if (savedInstanceState == null) {
            supportFragmentManager.beginTransaction()
                .replace(R.id.container, MainScreenFragment.newInstance())
                .commitNow()
        }
        if (intent.getBooleanExtra(Constants.FIRST_TIME, false)) {
            FirstSpeachDialog.showDialog(this)
        }
    }
}
```

MainScreenAdapter.kt

```
package com.example.mathwithfinik

import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.ImageView
import android.widget.TextView
import androidx.recyclerview.widget.RecyclerView
import com.example.mathwithfinik.models.MenuItem

class MainScreenAdapter(private val menuItems: ArrayList<MenuItem>) :
    RecyclerView.Adapter<MainScreenAdapter.MainScreenViewHolder>() {
    var onItemClick: ((MenuItem) -> Unit)? = null
    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int):
MainScreenViewHolder {
        val view = LayoutInflater.from(parent.context)
```

```
            .inflate(R.layout.msf_item, parent, false)
        return MainScreenViewHolder(view)
    }

    override fun getItemCount(): Int = menuItems.size

    override fun onBindViewHolder(holder: MainScreenViewHolder, position:
Int) {
        val item = menuItems[position]
        holder.msfItemImage.setImageDrawable(item.icon)
            holder.msfNameTv.text = item.name

    }

    inner class MainScreenViewHolder(v: View) : RecyclerView.ViewHolder(v) {
        val msfItemImage: ImageView = v.findViewById(R.id.msf_item_image)
        val msfNameTv: TextView = v.findViewById(R.id.msf_tv)

        init {
            v.setOnClickListener {
                onItemClick?.invoke(menuItems[adapterPosition])
            }
        }

    }
}
```

ComplexityFragment.kt

```
package com.example.mathwithfinik

import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.fragment.app.Fragment
import androidx.navigation.fragment.findNavController
import com.example.mathwithfinik.databinding.FragmentComplexityBinding

class ComplexityFragment : Fragment() {

    private lateinit var binding: FragmentComplexityBinding

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View {
        binding = FragmentComplexityBinding.inflate(inflater, container,
false)
        return binding.root
    }

    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        binding.apply {
            tvEasyLevel.setOnClickListener {
                navigate(Constants.EASY_CHAR)
            }
            tvMediumLevel.setOnClickListener {
                navigate(Constants.MEDIUM_CHAR)
            }
```

```
            tvHardLevel.setOnClickListener {
                navigate(Constants.HARD_CHAR)
            }
        }
    }

    private fun navigate(level: String) = findNavController().navigate(

ComplexityFragmentDirections.actionComplexityFragmentToPlusMinusFragment(
            level
        )
    )
}
```

BaseViewModel.kt

```
package com.example.mathwithfinik

import android.app.Dialog
import android.content.Context
import android.graphics.Color
import android.graphics.drawable.ColorDrawable
import android.view.View
import android.view.Window
import android.view.animation.AnimationUtils
import android.widget.Button
import android.widget.TextView
import androidx.lifecycle.MutableLiveData
import androidx.lifecycle.ViewModel
import com.example.mathwithfinik.databinding.ExerciseFragmentBinding
import com.example.mathwithfinik.models.MathExerciseModel
import kotlinx.android.synthetic.main.exercise_fragment.view.*
import kotlin.random.Random


abstract class BaseViewModel(open val binding: ExerciseFragmentBinding) :
ViewModel() {
    var score = 0
    val scoreLiveData = MutableLiveData<Int>()
    val anim by lazy {
        AnimationUtils.loadAnimation(binding.root.context,
R.anim.scale_alpha_notification)
    }

    fun generateNewExercise(level: String? = null) {
        val mathExercise = makeMathExerciseModel(level)
        val indexOfTrueAnswer: Int = Random.nextInt(0, 3)
        val arrayOfButtons = ArrayList<Button>()
        updateScore(score)
        arrayOfButtons.apply {
            add(binding.setOfFourButtons.button1)
            add(binding.setOfFourButtons.button2)
            add(binding.setOfFourButtons.button3)
            add(binding.setOfFourButtons.button4)
            get(indexOfTrueAnswer).apply {
                text = mathExercise.answerValue.toString()
                setOnClickListener {
                    score++
                    binding.notification.tv_notification.apply {
                        binding.notification.background =
```

```
                            context?.getDrawable(R.drawable.back_for_item)
                        visibility = View.VISIBLE
                        text = "Молодець"
                        this.startAnimation(anim)
                    }

                    updateScore(score)
                    this@BaseViewModel.generateNewExercise(level)
                }
            }
            var counter = 0
            for (i in 0..3) {
                if (indexOfTrueAnswer != i) {
                    arrayOfButtons[i].apply {
                        text = mathExercise.wrongAnswers[counter].toString()
                        setOnClickListener {
                            binding.notification.tv_notification.apply {
                                binding.notification.background =
                                    context?.getDrawable(R.drawable.back_red)
                                visibility = View.VISIBLE
                                text = "Подумай краще"
                                this.startAnimation(anim)
                            }
                            if (score > 0) score--
                            updateScore(score)
                        }
                    }
                    counter++
                }

            }
        }

        binding.exercise.apply {
            firstValue.text = mathExercise.firstValue.toString()
            secondValue.text = mathExercise.secondValue.toString()
        }
    }

    abstract fun makeMathExerciseModel(level: String? = null):
MathExerciseModel

    abstract fun actionBackToMainScreen()

    private fun updateScore(score: Int) = scoreLiveData.postValue(score)

    fun showDialog(context: Context?, text: String) {
        val dialog = context?.let { Dialog(it) }
        dialog?.let { dlg ->
            dlg.apply {
                requestWindowFeature(Window.FEATURE_NO_TITLE);
                setCancelable(false)
                setContentView(R.layout.dialog_first_speach_layout)

window?.setBackgroundDrawable(ColorDrawable(Color.TRANSPARENT))
                findViewById<TextView>(R.id.tv_main_text).text = text

findViewById<Button>(R.id.speach_dialog_ok_button).setOnClickListener {
                    dismiss()
                    actionBackToMainScreen()
                }
                show()
            }
```

```
        }
    }
}
```

ZadachaViewModel.kt

```
package com.example.mathwithfinik.zadacha

import android.annotation.SuppressLint
import android.content.Context
import android.view.View
import android.view.animation.AnimationUtils
import androidx.lifecycle.MutableLiveData
import androidx.lifecycle.ViewModel
import com.example.mathwithfinik.Constants
import com.example.mathwithfinik.R
import com.example.mathwithfinik.databinding.ZadachaFragmentBinding
import com.example.mathwithfinik.models.Zadacha
import kotlinx.android.synthetic.main.exercise_fragment.view.*
import kotlin.random.Random

class ZadachaViewModel(val binding: ZadachaFragmentBinding) : ViewModel() {


    var score = 0
    val scoreLiveData = MutableLiveData<Int>()
    val anim by lazy {
        AnimationUtils.loadAnimation(binding.root.context,
R.anim.scale_alpha_notification)
    }

    private fun increaseScore() = scoreLiveData.postValue(++score)

    private fun decreaseScore() {
        if (score > 0) scoreLiveData.postValue(--score)
    }

    private fun getRandom() = Random.nextInt(0, 9)

    @SuppressLint("UseCompatLoadingForDrawables")
    fun generateZadacha(context: Context?, arrayListZadach: List<Zadacha>,
isFirstly: Boolean = false) {
        var intAnswerFromET = 0
        if (binding.etAnswer.text.toString().isNotEmpty()) {
            intAnswerFromET = binding.etAnswer.text.toString().toInt()
        }
        var random = getRandom()
        val answer = arrayListZadach[random].answer

        binding.etAnswer.text.clear()
        if (intAnswerFromET == answer) {
            binding.notification.background =
context?.getDrawable(R.drawable.back_for_item)
            binding.notification.tv_notification.apply {
                visibility = View.VISIBLE
                text = "Молодець"
                this.startAnimation(anim)
            }
            increaseScore()
```

```
            binding.tvZadacha.text = arrayListZadach[random].text.trim()

        } else {
            binding.notification.background =
context?.getDrawable(R.drawable.back_red)
            decreaseScore()
            binding.notification.tv_notification.apply {
                visibility = View.VISIBLE
                text = "Подумай краще"
                this.startAnimation(anim)
            }
        }
    }

    fun getFileName(level: String?): String = if (Constants.isPremium) {
        when (level) {
            Constants.EASY_CHAR -> PREMIUM_ZADACHA_EASY
            Constants.MEDIUM_CHAR -> PREMIUM_ZADACHA_MEDIUM
            Constants.HARD_CHAR -> PREMIUM_ZADACHA_HARD
            else -> PREMIUM_ZADACHA_MEDIUM
        }
    } else {
        when (level) {
            Constants.EASY_CHAR -> FREE_ZADACHA_EASY
            Constants.MEDIUM_CHAR -> FREE_ZADACHA_MEDIUM
            Constants.HARD_CHAR -> FREE_ZADACHA_HARD
            else -> FREE_ZADACHA_MEDIUM
        }
    }

    companion object {
        const val FREE_ZADACHA_EASY = "free_zadachi_easy.txt"
        const val FREE_ZADACHA_MEDIUM = "free_zadachi_easy.txt"
        const val FREE_ZADACHA_HARD = "free_zadachi_easy.txt"
        const val PREMIUM_ZADACHA_EASY = "free_zadachi_easy"
        const val PREMIUM_ZADACHA_MEDIUM = "free_zadachi_easy"
        const val PREMIUM_ZADACHA_HARD = "free_zadachi_easy"
    }
}
```

ZadachaFragment.kt

```
package com.example.mathwithfinik.zadacha

import android.annotation.SuppressLint
import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.fragment.app.Fragment
import com.example.mathwithfinik.R
import com.example.mathwithfinik.databinding.ZadachaFragmentBinding
import com.example.mathwithfinik.models.Zadacha
import kotlin.random.Random


class ZadachaFragment : Fragment() {
```

```
    private lateinit var viewModel: ZadachaViewModel
    private lateinit var file: String
    private lateinit var fileName: String
    private var level: String? = null
    lateinit var binding: ZadachaFragmentBinding
    lateinit var arrayListZadach: List<Zadacha>

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View {
        binding = ZadachaFragmentBinding.inflate(inflater, container, false)
        viewModel = ZadachaViewModel(binding)
        return binding.root
    }

    @SuppressLint("UseCompatLoadingForDrawables")
    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)
        level = arguments?.getString("level")
        fileName = viewModel.getFileName(level)
        file = context?.let { context ->
            context.assets.open(fileName).bufferedReader().use {
                it.readText()
            }
        }.toString()
        arrayListZadach = file.split("***").map {
            it.split("|")
        }.map {
            Zadacha(it[0], it[1].trim().toInt())
        }


        binding.apply {
            tvZadacha.text = arrayListZadach[Random.nextInt(0,
9)].text.trim()
            tvScore.text = activity?.getString(R.string.score, 0)
        }

        binding.buttonAnswer.setOnClickListener {
            viewModel.generateZadacha(context, arrayListZadach)
        }

        viewModel.scoreLiveData.observe(viewLifecycleOwner) {
            activity?.runOnUiThread {
                binding.tvScore.text = activity?.getString(R.string.score,
it)
            }
        }

    }
}
```

MainScreenAdapter.kt

```
package com.example.mathwithfinik.ui.mainscreen

import android.annotation.SuppressLint
import android.app.Dialog
import android.content.Context
```

```
import android.content.SharedPreferences
import android.graphics.Color
import android.graphics.drawable.ColorDrawable
import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.view.Window
import android.widget.Button
import androidx.appcompat.app.AppCompatActivity
import androidx.fragment.app.Fragment
import androidx.lifecycle.ViewModelProvider
import androidx.navigation.fragment.findNavController
import androidx.recyclerview.widget.GridLayoutManager
import androidx.recyclerview.widget.LinearLayoutManager
import com.example.mathwithfinik.Constants
import com.example.mathwithfinik.MainScreenAdapter
import com.example.mathwithfinik.R
import com.example.mathwithfinik.databinding.MainScreenFragmentBinding
import com.example.mathwithfinik.models.MenuItem


class MainScreenFragment : Fragment() {

    companion object {
        fun newInstance() = MainScreenFragment()
    }

    private lateinit var sharedPref: SharedPreferences
    private lateinit var viewModel: MainScreenViewModel
    lateinit var binding: MainScreenFragmentBinding
    private val items = ArrayList<MenuItem>()
    private val adapterMenu = MainScreenAdapter(items)


    override fun onAttach(context: Context) {
        super.onAttach(context)
        sharedPref = context.getSharedPreferences(
            Constants.SHARED_PREFS_NAME,
            AppCompatActivity.MODE_PRIVATE
        )
    }

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View {
        binding = MainScreenFragmentBinding.inflate(inflater, container,
false)
        return binding.root
    }

    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {

        binding.mspRv.apply {
            if (items.isEmpty()) {
                initArray()
            }
            layoutManager = GridLayoutManager(activity, 2,
LinearLayoutManager.HORIZONTAL, false)
            adapter = adapterMenu
        }
```

```
        adapterMenu.onItemClick = { item ->
            when (item.name) {
                context?.getString(R.string.multiply) -> {

findNavController().navigate(R.id.action_mainScreenFragment_to_multiplyFragme
nt)
                }
                context?.getString(R.string.divide) -> {

findNavController().navigate(R.id.action_mainScreenFragment_to_divideFragment
)
                }
                context?.getString(R.string.plus_minus) -> {

findNavController().navigate(R.id.action_mainScreenFragment_to_complexityFrag
ment)
                }
                context?.getString(R.string.zadachi) -> {
                    showDialog(context)
                }
                else -> {
                }
            }

        }
        binding.mspTvMoneyBalance.text = sharedPref.getInt(Constants.BALANCE,
0).toString()

    }

    fun showDialog(context: Context?) {
        val dialog = context?.let { Dialog(it) }
        dialog?.let { dlg ->
            dlg.requestWindowFeature(Window.FEATURE_NO_TITLE);
            dlg.setCancelable(false);
            dlg.setContentView(R.layout.dialog_level_layout)

dlg.window?.setBackgroundDrawable(ColorDrawable(Color.TRANSPARENT))
            dlg.findViewById<Button>(R.id.button_hard).setOnClickListener {
                findNavController().navigate(

MainScreenFragmentDirections.actionMainScreenFragmentToZadachaFragment(
                        Constants.HARD_CHAR
                    )
                )

MainScreenFragmentDirections.actionMainScreenFragmentToZadachaFragment(Consta
nts.HARD_CHAR)
                dlg.dismiss()
            }
            dlg.findViewById<Button>(R.id.button_medium).setOnClickListener {
                findNavController().navigate(

MainScreenFragmentDirections.actionMainScreenFragmentToZadachaFragment(
                        Constants.MEDIUM_CHAR
                    )
                )
                dlg.dismiss()
            }
            dlg.findViewById<Button>(R.id.button_easy).setOnClickListener {
                findNavController().navigate(

MainScreenFragmentDirections.actionMainScreenFragmentToZadachaFragment(
```

```
                        Constants.EASY_CHAR
                    )
                )
                dlg.dismiss()
            }
            dlg.show()
        }
    }

    @SuppressLint("UseCompatLoadingForDrawables")
    fun initArray() {
        items.add(
            MenuItem(
                resources.getDrawable(R.drawable.icon_plus_minus),
                getString(R.string.plus_minus)
            )
        )
        items.add(
            MenuItem(resources.getDrawable(R.drawable.icon_divide),
getString(R.string.divide))
        )
        items.add(
            MenuItem(resources.getDrawable(R.drawable.icon_multiply),
getString(R.string.multiply))
        )
        items.add(
            MenuItem(
                resources.getDrawable(R.drawable.icon_zadachi),
                getString(R.string.zadachi)
            )
        )
        items.add(
            MenuItem(resources.getDrawable(R.drawable.icon_shopping_cart),
"Магазин", true)
        )
        items.add(
            MenuItem(resources.getDrawable(R.drawable.icon_world), "Табличка
\n множення9", true)
        )
    }


    override fun onActivityCreated(savedInstanceState: Bundle?) {
        super.onActivityCreated(savedInstanceState)
        viewModel = ViewModelProvider(this)[MainScreenViewModel::class.java]
    }
}
```

FirstSpeachDialog.kt

```
package com.example.mathwithfinik.ui.mainscreen

import android.app.Dialog
import android.content.Context
import android.graphics.Color
import android.graphics.drawable.ColorDrawable
import android.view.Window
import android.widget.Button
import com.example.mathwithfinik.R
```

```
object FirstSpeachDialog {

    fun showDialog(context: Context?) {
        val dialog = context?.let { Dialog(it) }
        dialog?.let { dialog ->
            dialog.requestWindowFeature(Window.FEATURE_NO_TITLE);
            dialog.setCancelable(false);
            dialog.setContentView(R.layout.dialog_first_speach_layout)

dialog.window?.setBackgroundDrawable(ColorDrawable(Color.TRANSPARENT))

dialog.findViewById<Button>(R.id.speach_dialog_ok_button).setOnClickListener
{
                dialog.dismiss()
            }
            dialog.show()
        }
    }

}
```

PlusMinusViewModel.kt

```
package com.example.mathwithfinik.plus_minus_screen

import androidx.navigation.findNavController
import com.example.mathwithfinik.BaseViewModel
import com.example.mathwithfinik.Constants
import com.example.mathwithfinik.R
import com.example.mathwithfinik.databinding.ExerciseFragmentBinding
import com.example.mathwithfinik.models.MathExerciseModel
import kotlin.random.Random

class PlusMinusViewModel(override val binding: ExerciseFragmentBinding) :
BaseViewModel(binding) {

    private var firstValue = 0
    private var secondValue = 0
    private var answer = 0
    private val wrongAnswers = ArrayList<Number>()
    private var max = 0
    private var level: String? = null

    override fun makeMathExerciseModel(
        level: String?
    ): MathExerciseModel {
        if (this.level == null) {
            this.level = level
        }
        if (level.equals(Constants.HARD_CHAR)) {
            binding.exercise.apply {
                firstValue.textSize = 60F
                secondValue.textSize = 60F
                symbol.textSize = 60F
            }
        }
        generatePair(this.level)
        if (Random.nextBoolean()) {
            answer = firstValue + secondValue
            when (this.level) {
```

```
                Constants.EASY_CHAR -> {
                    while (answer > 10) {
                        generatePair(this.level)
                        answer = firstValue + secondValue
                    }
                }
                Constants.MEDIUM_CHAR -> {
                    while (answer > 100) {
                        generatePair(this.level)
                        answer = firstValue + secondValue
                    }
                }
                Constants.HARD_CHAR -> {
                    while (answer > 1000) {
                        generatePair(this.level)
                        answer = firstValue + secondValue
                    }
                }
            }

            max = answer + firstValue
            while (max < 5) {
                max++
            }
            binding.exercise.symbol.text = "+"
        } else {
            if (firstValue < secondValue) {
                val tmp = firstValue
                firstValue = secondValue
                secondValue = tmp
            }
            answer = firstValue - secondValue
            max = firstValue + secondValue
            while (max < 5) {
                max++
            }
            binding.exercise.symbol.text = "-"
        }
        val min = 1
        while (wrongAnswers.size < 3) {
            val value = Random.nextInt(min, max)
            if (value != answer && !wrongAnswers.contains(value)) {
                wrongAnswers.add(value)
            }
        }
        return MathExerciseModel(firstValue, secondValue, answer,
wrongAnswers)
    }

    private fun generatePair(level: String?): Pair<Int, Int> {
        when (level) {
            Constants.EASY_CHAR -> {
                firstValue = Random.nextInt(1, 10)
                secondValue = Random.nextInt(1, 10)
            }
            Constants.MEDIUM_CHAR -> {
                firstValue = Random.nextInt(1, 100)
                secondValue = Random.nextInt(1, 100)
            }
            Constants.HARD_CHAR -> {
                firstValue = Random.nextInt(1, 1000)
                secondValue = Random.nextInt(1, 1000)
            }
```

```
            else -> {
                firstValue = 0
                secondValue = 0
            }
        }
        return Pair(firstValue, secondValue)
    }

    override fun actionBackToMainScreen() {
        binding.root.findNavController()
            .navigate(R.id.action_plusMinusFragment_to_mainScreenFragment)
    }
}
```

PlusMinusFragment.kt

```
package com.example.mathwithfinik.plus_minus_screen

import android.content.Context
import android.content.SharedPreferences
import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.appcompat.app.AppCompatActivity
import androidx.fragment.app.Fragment
import com.example.mathwithfinik.Constants
import com.example.mathwithfinik.R
import com.example.mathwithfinik.databinding.ExerciseFragmentBinding
import kotlinx.coroutines.*

class PlusMinusFragment : Fragment() {

    private lateinit var sharedPref: SharedPreferences
    private lateinit var binding: ExerciseFragmentBinding
    private lateinit var viewModel: PlusMinusViewModel
    private var job = Job()
    private val scope = CoroutineScope(Dispatchers.Main + job)
    private var balance = 0
    private var result: String? = ""


    override fun onAttach(context: Context) {
        super.onAttach(context)
        sharedPref = context.getSharedPreferences(
            Constants.SHARED_PREFS_NAME,
            AppCompatActivity.MODE_PRIVATE
        )
    }

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View {
        binding = ExerciseFragmentBinding.inflate(inflater, container, false)
        viewModel = PlusMinusViewModel(binding)
        return binding.root
    }

    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
```

```
        balance = sharedPref.getInt(Constants.BALANCE, 0)
        binding.progressbar.max = 40
        result = arguments?.getString("level_plus_minus")
        scope.launch {
            val tickSeconds = 0
            for (second in 40 downTo tickSeconds) {
                binding.apply {
                    progressbar.progress = second
                    tvSecondsLeft.text =
                        activity?.getString(R.string.left_time, second)
                            ?.let { String.format(it) }
                }

                delay(1000)
            }
            if (viewModel.score > 0) {
                balance += viewModel.score
                sharedPref.edit().putInt(Constants.BALANCE, balance).apply()
            }
            binding.progressbar.progress = 0
            activity?.getString(R.string.result, viewModel.score)
                ?.let { String.format(it) }?.let {
viewModel.showDialog(context, it) }

        }

        viewModel.generateNewExercise(result)

        viewModel.scoreLiveData.observe(viewLifecycleOwner) {
            activity?.runOnUiThread {
                binding.tvScore.text = "Твій рахунок: $it"
            }
        }
    }

    override fun onDestroy() {
        super.onDestroy()
        job.cancel()
    }
}
```

MultiplyViewModel.kt

```
package com.example.mathwithfinik.multiply

import androidx.navigation.findNavController
import com.example.mathwithfinik.BaseViewModel
import com.example.mathwithfinik.R
import com.example.mathwithfinik.models.MathExerciseModel
import com.example.mathwithfinik.databinding.ExerciseFragmentBinding
import kotlin.random.Random

class MultiplyViewModel(override val binding: ExerciseFragmentBinding) :
BaseViewModel(binding) {

    override fun makeMathExerciseModel(
        level: String?
    ): MathExerciseModel {
        val firstValue: Int = Random.nextInt(1, 10)
        val secondValue: Int = Random.nextInt(1, 10)
```

```
        val answer = firstValue * secondValue;
        val wrongAnswers = ArrayList<Number>()
        var max = answer + answer
        while (max < 5) {
            max += answer
        }
        val min = 1
        while (wrongAnswers.size < 3) {
            val value = Random.nextInt(min, max)
            if (value != answer && !wrongAnswers.contains(value)) {
                wrongAnswers.add(value)
            }
        }
        return MathExerciseModel(firstValue, secondValue, answer,
wrongAnswers)
    }

    override fun actionBackToMainScreen() {
        binding.root.findNavController()
            .navigate(R.id.action_multiplyFragment_to_mainScreenFragment)
    }
}
```

MultiplyFragment.kt

```
package com.example.mathwithfinik.multiply

import android.content.Context
import android.content.SharedPreferences
import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.appcompat.app.AppCompatActivity
import androidx.fragment.app.Fragment
import com.example.mathwithfinik.Constants
import com.example.mathwithfinik.R
import com.example.mathwithfinik.databinding.ExerciseFragmentBinding
import kotlinx.coroutines.*

class MultiplyFragment : Fragment() {

    private lateinit var sharedPref: SharedPreferences
    lateinit var binding: ExerciseFragmentBinding
    lateinit var viewModel: MultiplyViewModel
    private var job = Job()
    val scope = CoroutineScope(Dispatchers.IO + job)
    private var balance = 0

    override fun onAttach(context: Context) {
        super.onAttach(context)
        sharedPref = context.getSharedPreferences(
            Constants.SHARED_PREFS_NAME,
            AppCompatActivity.MODE_PRIVATE
        )
    }

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
```

```
    ): View {
        binding = ExerciseFragmentBinding.inflate(inflater, container, false)
        viewModel = MultiplyViewModel(binding)
        return binding.root
    }

    @OptIn(DelicateCoroutinesApi::class)
    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        balance = sharedPref.getInt(Constants.BALANCE, 0)
        binding.progressbar.max = 40
        binding.exercise.symbol.text = "*"
        scope.launch {
            val tickSeconds = 1
            for (second in 40 downTo tickSeconds) {
                activity?.runOnUiThread {
                    binding.apply {
                        progressbar.progress = second
                        tvSecondsLeft.text =
                            activity?.getString(R.string.left_time, second)
                                ?.let { String.format(it) }
                    }
                }
                delay(1000)
            }
            if (viewModel.score > 0) {
                balance += viewModel.score
                sharedPref.edit().putInt(Constants.BALANCE, balance).apply()
            }
            activity?.runOnUiThread {
                binding.progressbar.progress = 0
                activity?.getString(R.string.result, viewModel.score)
                    ?.let { String.format(it) }?.let {
viewModel.showDialog(context, it) }
            }
        }

        viewModel.generateNewExercise()

        viewModel.scoreLiveData.observe(viewLifecycleOwner) {
            activity?.runOnUiThread {
                binding.tvScore.text = "Твій рахунок: $it"
            }
        }
    }

    override fun onDestroy() {
        super.onDestroy()
        job.cancel()
    }
}
```

MathExerciseModel.kt

```
package com.example.mathwithfinik.models

data class MathExerciseModel(
    val firstValue:Int,
    val secondValue:Int,
    val answerValue: Int,
    //three numbers of wrong answers
```

```
    var wrongAnswers: ArrayList<Number>
)
```

MenuItem.kt

```
package com.example.mathwithfinik.models

import android.graphics.drawable.Drawable

data class MenuItem(val icon: Drawable, val name: String, val blocked:
Boolean? = false)
```

DivideViewModel.kt

```
package com.example.mathwithfinik.divide

import androidx.navigation.findNavController
import com.example.mathwithfinik.BaseViewModel
import com.example.mathwithfinik.R
import com.example.mathwithfinik.databinding.ExerciseFragmentBinding
import com.example.mathwithfinik.models.MathExerciseModel
import kotlin.random.Random

class DivideViewModel(override val binding: ExerciseFragmentBinding) :
BaseViewModel(binding) {

    override fun makeMathExerciseModel(
        level: String?
    ): MathExerciseModel {
        val answer: Int = Random.nextInt(2, 10)
        val secondValue: Int = Random.nextInt(2, 10)
        val firstValue = secondValue * answer
        val wrongAnswers = ArrayList<Number>()
        var max = answer * 2
        while (max < 5) {
            max++
        }
        val min = 1
        while (wrongAnswers.size < 3) {
            val value = Random.nextInt(min, max)
            if (value != answer && !wrongAnswers.contains(value)) {
                wrongAnswers.add(value)
            }
        }
        return MathExerciseModel(firstValue, secondValue, answer,
wrongAnswers)
    }

    override fun actionBackToMainScreen() = binding.root.findNavController()
        .navigate(R.id.action_divideFragment_to_mainScreenFragment)

}
```

DivideFragment.kt

```
package com.example.mathwithfinik.divide

import android.content.Context
import android.content.SharedPreferences
import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.appcompat.app.AppCompatActivity
import androidx.fragment.app.Fragment
import com.example.mathwithfinik.Constants
import com.example.mathwithfinik.R
import com.example.mathwithfinik.databinding.ExerciseFragmentBinding
import kotlinx.coroutines.*

class DivideFragment : Fragment() {

    private lateinit var sharedPref: SharedPreferences
    lateinit var binding: ExerciseFragmentBinding
    private lateinit var viewModel: DivideViewModel
    var job = Job()
    val scope = CoroutineScope(Dispatchers.Default)
    private var balance = 0

    override fun onAttach(context: Context) {
        super.onAttach(context)
        sharedPref = context.getSharedPreferences(
            Constants.SHARED_PREFS_NAME,
            AppCompatActivity.MODE_PRIVATE
        )
    }

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View {
        binding = ExerciseFragmentBinding.inflate(inflater, container, false)
        viewModel = DivideViewModel(binding)
        return binding.root
    }

    @OptIn(DelicateCoroutinesApi::class)
    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        balance = sharedPref.getInt(Constants.BALANCE, 0)
        binding.progressbar.max = 40
        binding.exercise.symbol.text = ":"
        scope.launch {
            val tickSeconds = 0
            for (second in 40 downTo tickSeconds) {
                activity?.runOnUiThread {
                    binding.apply {
                        progressbar.progress = second
                        tvSecondsLeft.text =
                            activity?.getString(R.string.left_time, second)
                                ?.let { String.format(it) }
                    }
                }
                delay(1000)
            }
            if (viewModel.score > 0) {
                balance += viewModel.score
```

```
                sharedPref.edit().putInt(Constants.BALANCE, balance).apply()
            }
            activity?.runOnUiThread {
                binding.progressbar.progress = 0
                activity?.getString(R.string.result, viewModel.score)
                    ?.let { String.format(it) }?.let {
viewModel.showDialog(context, it) }
            }
        }

        viewModel.generateNewExercise()

        viewModel.scoreLiveData.observe(viewLifecycleOwner) {
            activity?.runOnUiThread {
                binding.tvScore.text = activity?.getString(R.string.score,
it)
            }
        }

    }

    override fun onDestroy() {
        super.onDestroy()
        job.cancel()
    }
}
```

MainScreenFragment.kt

```
package com.example.mathwithfinik.ui.mainscreen

import android.annotation.SuppressLint
import android.app.Dialog
import android.content.Context
import android.content.SharedPreferences
import android.graphics.Color
import android.graphics.drawable.ColorDrawable
import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.view.Window
import android.widget.Button
import androidx.appcompat.app.AppCompatActivity
import androidx.fragment.app.Fragment
import androidx.lifecycle.ViewModelProvider
import androidx.navigation.fragment.findNavController
import androidx.recyclerview.widget.GridLayoutManager
import androidx.recyclerview.widget.LinearLayoutManager
import com.example.mathwithfinik.Constants
import com.example.mathwithfinik.MainScreenAdapter
import com.example.mathwithfinik.R
import com.example.mathwithfinik.databinding.MainScreenFragmentBinding
import com.example.mathwithfinik.models.MenuItem


class MainScreenFragment : Fragment() {

    companion object {
        fun newInstance() = MainScreenFragment()
```

```
    }

    private lateinit var sharedPref: SharedPreferences
    private lateinit var viewModel: MainScreenViewModel
    lateinit var binding: MainScreenFragmentBinding
    private val items = ArrayList<MenuItem>()
    private val adapterMenu = MainScreenAdapter(items)


    override fun onAttach(context: Context) {
        super.onAttach(context)
        sharedPref = context.getSharedPreferences(
            Constants.SHARED_PREFS_NAME,
            AppCompatActivity.MODE_PRIVATE
        )
    }

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View {
        binding = MainScreenFragmentBinding.inflate(inflater, container,
false)
        return binding.root
    }

    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {

        binding.mspRv.apply {
            if (items.isEmpty()) {
                initArray()
            }
            layoutManager = GridLayoutManager(activity, 2,
LinearLayoutManager.HORIZONTAL, false)
            adapter = adapterMenu
        }

        adapterMenu.onItemClick = { item ->
            when (item.name) {
                context?.getString(R.string.multiply) -> {

findNavController().navigate(R.id.action_mainScreenFragment_to_multiplyFragme
nt)
                }
                context?.getString(R.string.divide) -> {

findNavController().navigate(R.id.action_mainScreenFragment_to_divideFragment
)
                }
                context?.getString(R.string.plus_minus) -> {

findNavController().navigate(R.id.action_mainScreenFragment_to_complexityFrag
ment)
                }
                context?.getString(R.string.zadachi) -> {
                    showDialog(context)
                }
                else -> {
                }
            }

        }
        binding.mspTvMoneyBalance.text = sharedPref.getInt(Constants.BALANCE,
```

```
0).toString()

    }

    fun showDialog(context: Context?) {
        val dialog = context?.let { Dialog(it) }
        dialog?.let { dlg ->
            dlg.requestWindowFeature(Window.FEATURE_NO_TITLE);
            dlg.setCancelable(false);
            dlg.setContentView(R.layout.dialog_level_layout)

dlg.window?.setBackgroundDrawable(ColorDrawable(Color.TRANSPARENT))
            dlg.findViewById<Button>(R.id.button_hard).setOnClickListener {
                findNavController().navigate(

MainScreenFragmentDirections.actionMainScreenFragmentToZadachaFragment(
                        Constants.HARD_CHAR
                    )
                )

MainScreenFragmentDirections.actionMainScreenFragmentToZadachaFragment(Consta
nts.HARD_CHAR)
                dlg.dismiss()
            }
            dlg.findViewById<Button>(R.id.button_medium).setOnClickListener {
                findNavController().navigate(

MainScreenFragmentDirections.actionMainScreenFragmentToZadachaFragment(
                        Constants.MEDIUM_CHAR
                    )
                )
                dlg.dismiss()
            }
            dlg.findViewById<Button>(R.id.button_easy).setOnClickListener {
                findNavController().navigate(

MainScreenFragmentDirections.actionMainScreenFragmentToZadachaFragment(
                        Constants.EASY_CHAR
                    )
                )
                dlg.dismiss()
            }
            dlg.show()
        }
    }

    @SuppressLint("UseCompatLoadingForDrawables")
    fun initArray() {
        items.add(
            MenuItem(
                resources.getDrawable(R.drawable.icon_plus_minus),
                getString(R.string.plus_minus)
            )
        )
        items.add(
            MenuItem(resources.getDrawable(R.drawable.icon_divide),
getString(R.string.divide))
        )
        items.add(
            MenuItem(resources.getDrawable(R.drawable.icon_multiply),
getString(R.string.multiply))
        )
        items.add(
```

```
            MenuItem(
                resources.getDrawable(R.drawable.icon_zadachi),
                getString(R.string.zadachi)
            )
        )
        items.add(
            MenuItem(resources.getDrawable(R.drawable.icon_shopping_cart),
getString(R.string.shop)
        )
    }


    override fun onActivityCreated(savedInstanceState: Bundle?) {
        super.onActivityCreated(savedInstanceState)
        viewModel = ViewModelProvider(this)[MainScreenViewModel::class.java]
    }
}
```

Додаток Д.

ГРАФІЧНА ЧАСТИНА

РОЗРОБКА МОБІЛЬНОГО ПРОГРАМНОГО ДОДАТКУ ДЛЯ ВИВЧЕННЯ МАТЕМАТИКИ ДІТЬМИ МОЛОДШОЇ ШКОЛИ

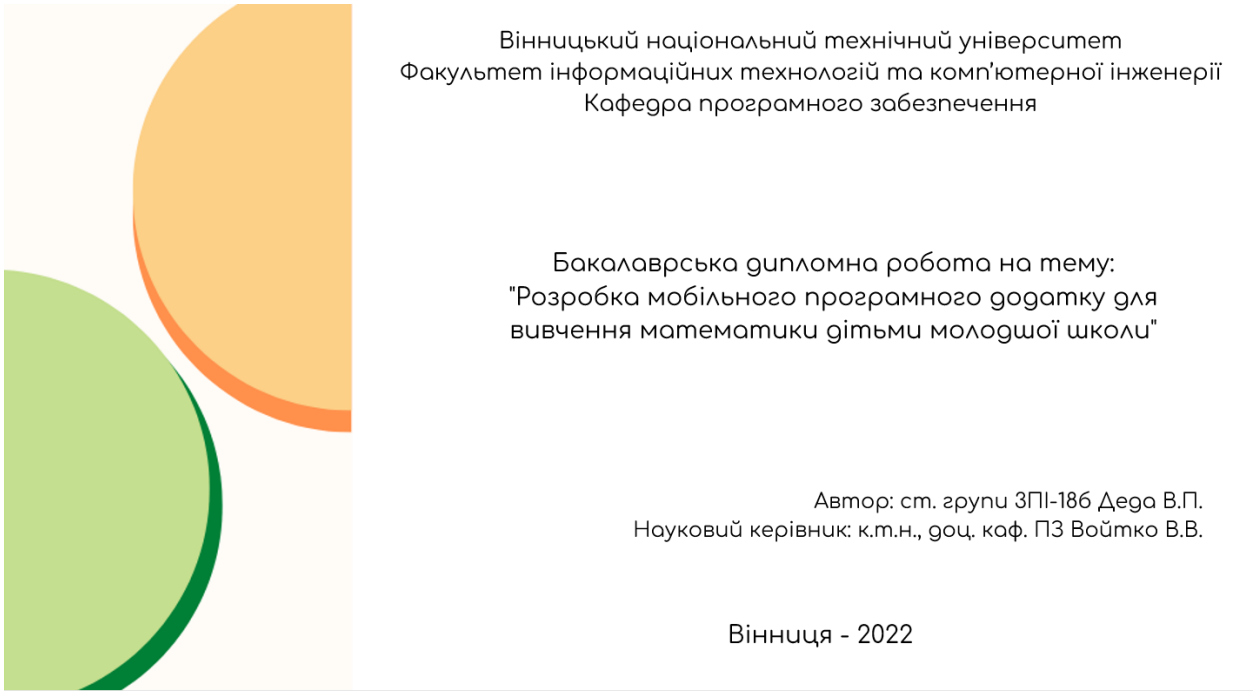


Рисунок Д.1 – Титульний слайд

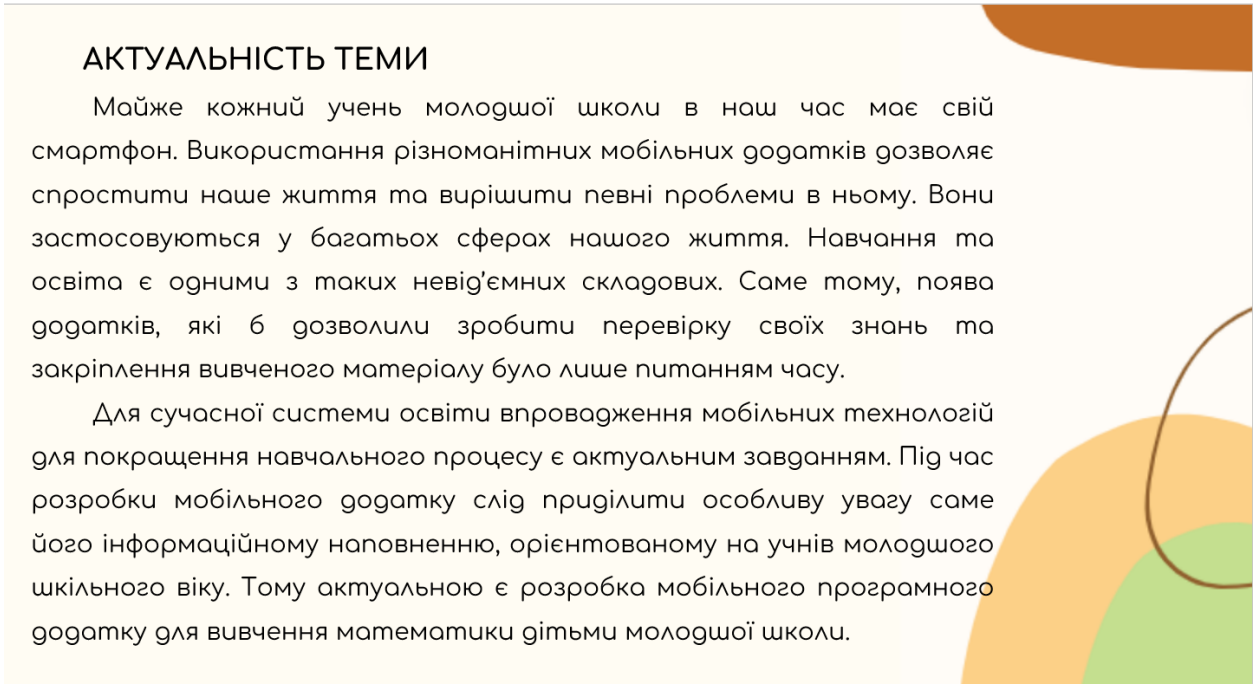


Рисунок Д.2 – Актуальність теми

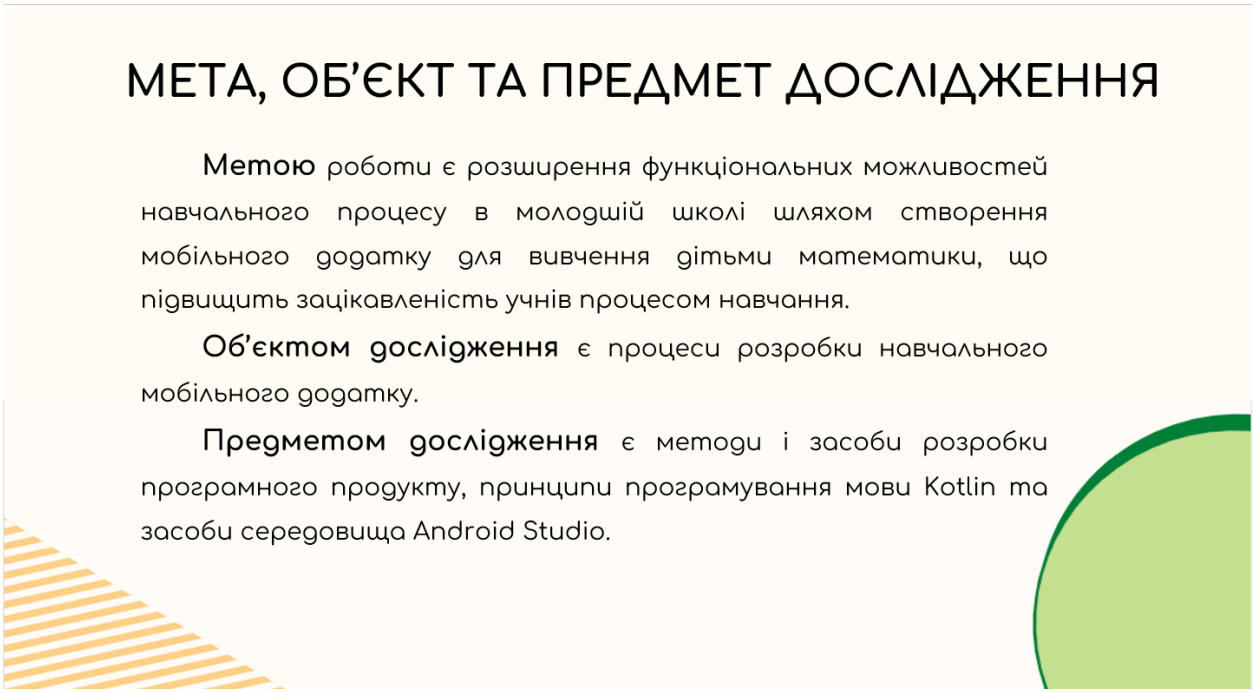


Рисунок Д.3 – Мета, об'єкт та предмет дослідження

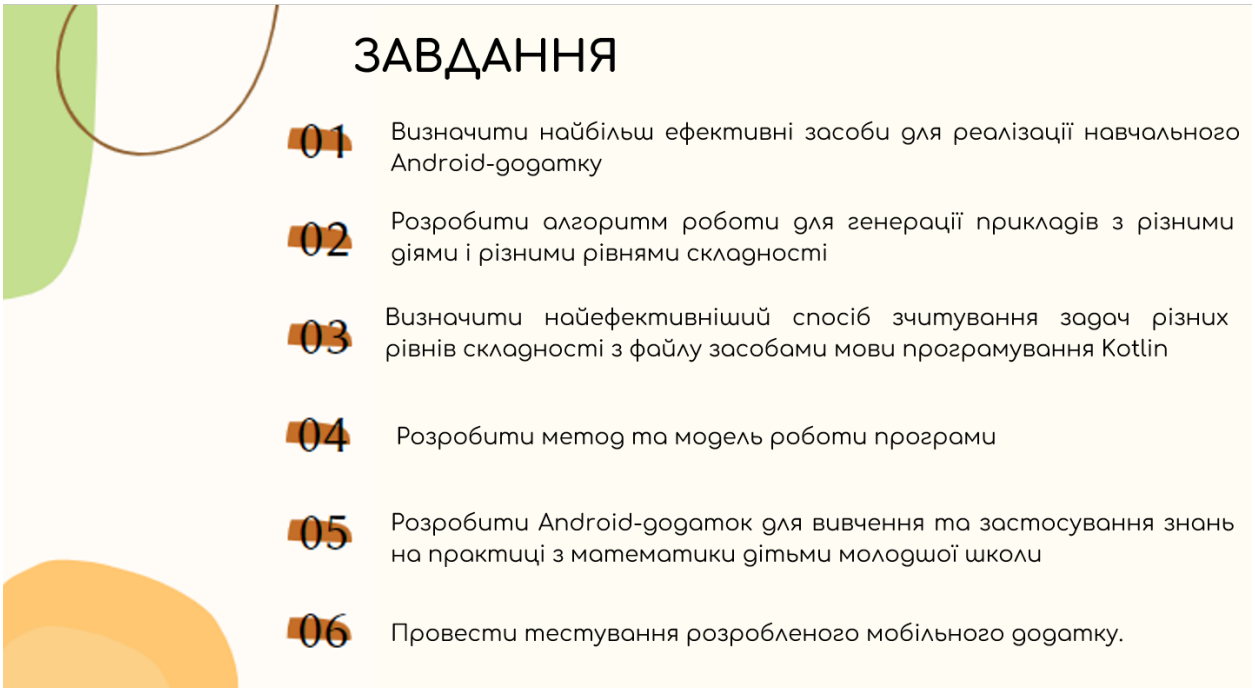


Рисунок Д.4 – Задачі дослідження

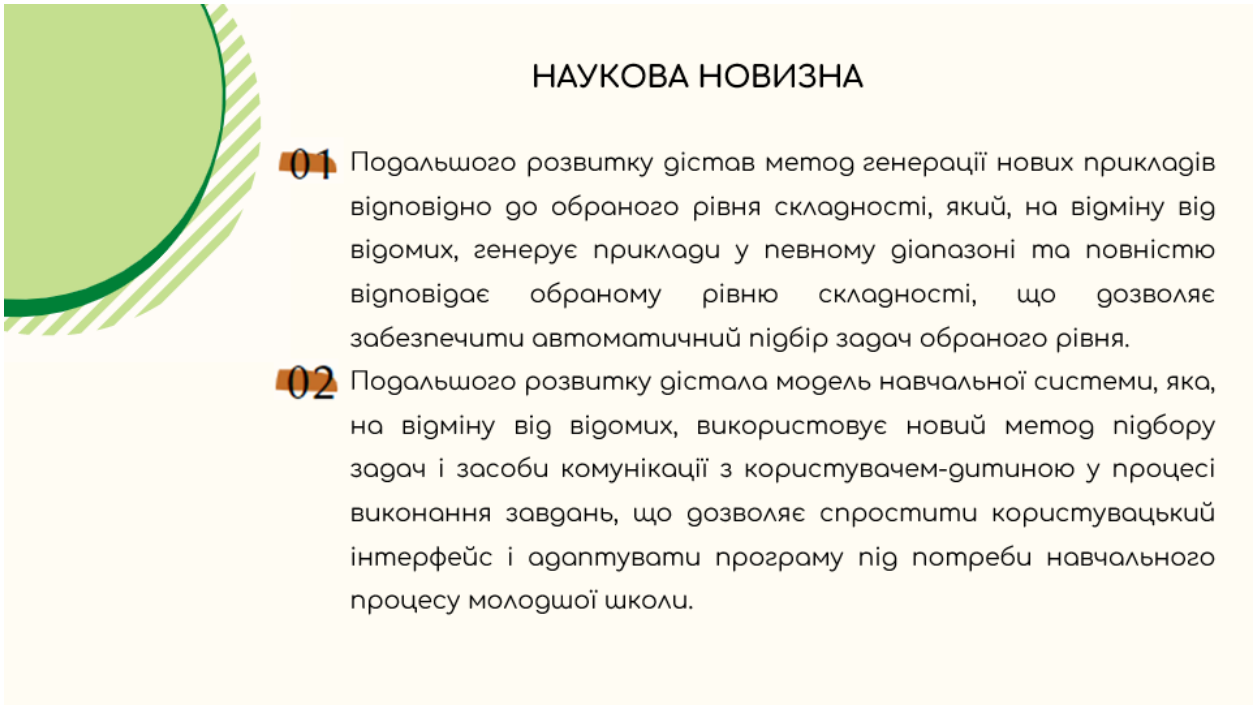


Рисунок Д. 5 – Наукова новизна

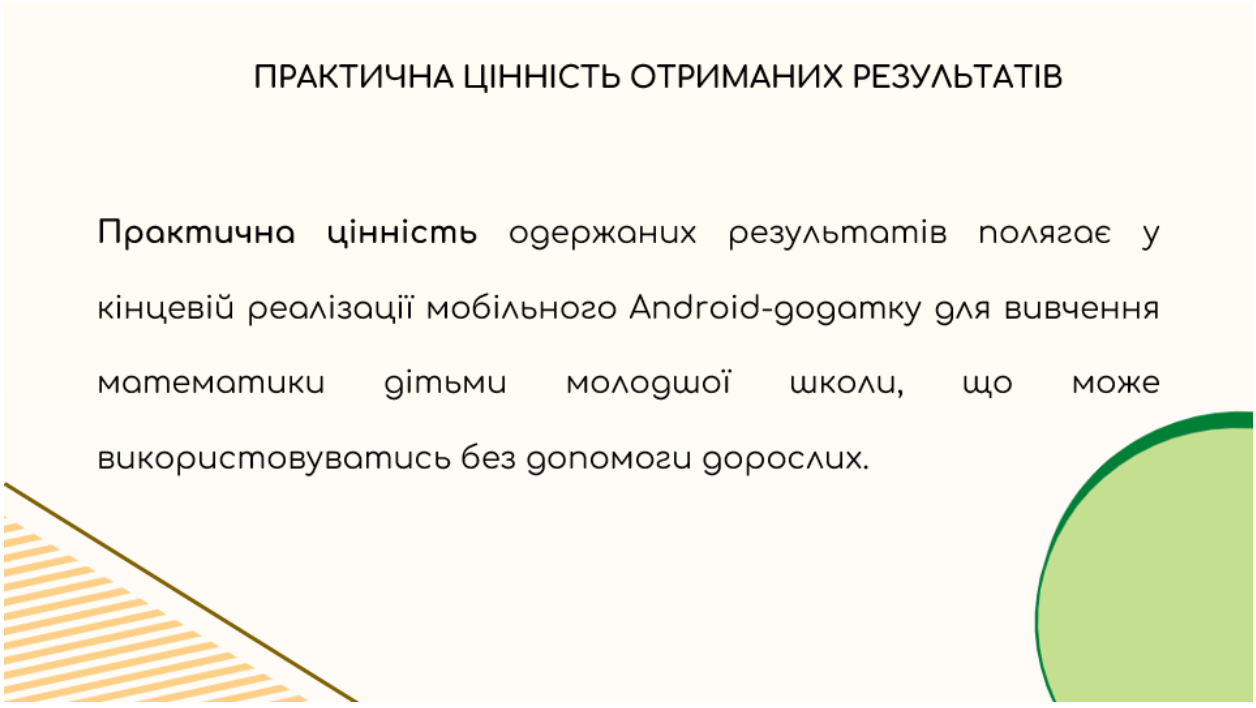


Рисунок Д.6 – Практична цінність одержаних результатів

АКТ

про впровадження результатів наукової роботи «Розробка мобільного програмного додатку для вивчення математики дітьми молодшої школи»

Результати бакалаврської кваліфікаційної роботи «Розробка мобільного програмного додатку для вивчення математики дітьми молодшої школи», що полягають в розробці програмного додатку MathForKids, який призначений для вивчення математики дітьми молодшої школи, а також застосування цих знань на практиці впроваджено в Деражнянському ліцеї №1 імені П. Стрілецького.

Додаток Б – Акт впровадження (Деражнянський ліцей №1 імені П. Стрілецького)

АКТ
про впровадження результатів наукової роботи
«Розробка мобільного програмного додатку для вивчення математики дітьми молодшої школи»
студента ВНТУ Деди Владислава Петровича

Результати бакалаврської кваліфікаційної роботи «Розробка мобільного програмного додатку для вивчення математики дітьми молодшої школи» студента ВНТУ Деди В.П., що полягають в розробці:

- методу генерації нових прикладів відповідно до обраного рівня складності, який генерує приклади у певному діапазоні, та повністю відповідає обраному рівню складності;
- методу сповіщення користувача про правильність розв'язання прикладів, який сповіщає користувача про це прямо під час гри, при чому сповіщення мають різні кольори, щоб краще та швидше розуміти та відрізняти чи правильно приклад чи задача були вирішені. Тому сповіщення про правильне вирішення має зелений колір, а про не правильне – червоний;
- програмного додатку MathForKids, який призначений для вивчення математики дітьми молодшої школи, а також застосування цих знань на практиці впроваджено в Деражнянському ліцеї №1 імені П. Стрілецького.

Директор Деражнянського ліцею №1 імені П. Стрілецького — Сідлецький В. М.

Рисунок Д.7 – Акт про впровадження наукової роботи

ПОРІВНЯЛЬНИЙ АНАЛІЗ АНАЛОГІВ

Критерій \ Назва гри	DragonBox Algebra	Kids Number and Math Lite	Operation Math	Mathmateer	Geeksmath	MathForKids
Наявність рівнів складності завдань	-	-	+	+	+	+
Інтерфейс українською мовою	-	-	-	-	-	+
Зрозумілий інтерфейс	+	+	-	+	-	+
Кросплатформність	-	+	+	-	+	-
Підсумковий результат	1	2	2	2	2	3

Рисунок Д.8 – Порівняльний аналіз аналогів

Метод покрокової роботи мобільного додатку «MathForKids»

1. Запуск Splash screen, або екрану запуску. Під час завантаження програми показується логотип і назва програми з анімацією миготіння.

2. Запуск головного вікна програми, на якому відображається тваринка, поточний баланс монет користувача, і список кнопок який прокручується горизонтально. Цей список, дає можливість користувачеві вибрати той чи інший тип завдання.

3. Можливість обрати тип завдання.

4. Запуск діалогового вікна з можливістю вибрати рівень складності завдання.

5. Запуск вікна програми з прикладами.

6. Генерація прикладів. При правильній відповіді, до рахунку додається один, в іншому випадку відповідно з рахунку віднімається один.

7. Асинхронна робота таймера, тобто робота таймера в іншому потоці, для уникнення блокування основного.

8. Після закінчення часу, на екрані появляється діалогове вікно, яке сповіщає про це, а також показує кількість зароблених монет.

9. Можливість обрати інший тип завдання.

10. Закінчення роботи програми.

Рисунок Д.9 – Метод покрокової роботи мобільного додатку «MathForKids»

Алгоритм генерації нового прикладу для дії додавання та віднімання

1. Генерація пари чисел, відповідно до обраної складності прикладів. Тобто, коли обрана складність EASY, числа генеруються в діапазоні від 1 до 10, коли обрано рівень MEDIUM – від 1 до 100, і коли обрано рівень HARD верхня межа для генерації пари чисел є 1000.

2. Генерація boolean значення, і відповідно якщо буде true – дія у прикладі буде додавання, в іншому випадку – дія буде віднімання.

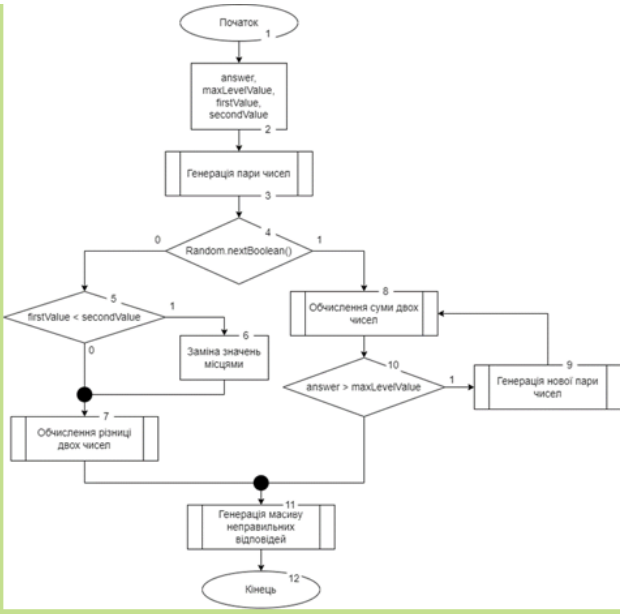
3. Якщо випало додавання, програма перевіряє, який рівень складності обрано. І якщо два числа при додаванні виходять за свою межу, яка складає відповідно 10, 100, 1000, пара чисел буде генеруватись доки не пройде цю умову.

4. Якщо випало віднімання, програма перевіряє, чи є перше число зі згенерованої пари меншим за друге. Якщо так, вони міняються місцями. Це зроблено для того, щоб запобігти від'ємних значень у відповіді.

5. Встановлення максимального значення, для верхньої межі генерації чисел неправильних відповідей, відповідно до правильної відповіді та значень виразу.

6. Якщо межа менша, ніж число 5, верхня межа встановлюється число 4. Це зроблено для того, щоб запобігти безкінечного генерування неправильних відповідей, тобто, щоб у діапазоні було мінімум 4 числа.

7. Генерація масиву з трьох значень типу int неправильних відповідей. Це відбувається, поки не заповниться масив, а значення генеруються від 1 до максимального значення, встановленого раніше. Якщо значення не дорівнює правильній відповіді і є унікальним в масиві, воно додається в цей масив.



8. Метод повертає повністю готовий приклад у вигляді data class-у з назвою MathExerciseModel, який містить два значення, правильну відповідь, і три неправильні.

Рисунок Д.10 – Алгоритм генерації нового прикладу для дії додавання та віднімання

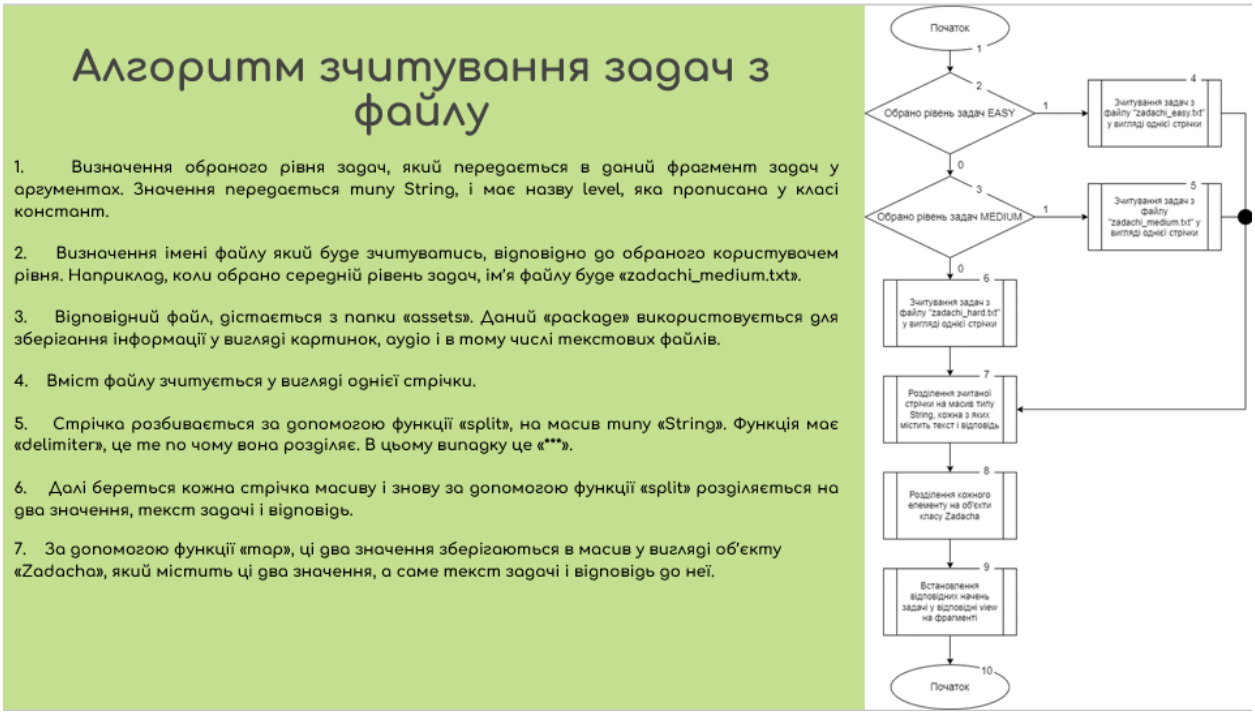


Рисунок Д.11 – Алгоритм зчитування задач з файлу

Рисунок Д.12 – Тестування програми

Апробація результатів роботи. Результати роботи були представлені на LI Науково-технічній конференції факультету інформаційних технологій та комп'ютерної інженерії (2022 р., м.Вінниця).

Публікації. Войтко В.В. Розробка мобільного додатку "MATH FOR KIDS", спрямованого на вивчення математики дітьми молодшої школи / Войтко В.В., Круподьорова Л. М., Денисюк А. В., Гаврилюк О. В., Барчук Н. Є., Деда В.П. Матеріали LI Науково-технічна конференція факультету інформаційних технологій та комп'ютерної інженерії (2022) ВНТУ, Факультет інформаційних технологій та комп'ютерної інженерії, Секція програмного забезпечення. [Електронний ресурс] – Режим доступу до ресурсу: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2022/paper/view/15098/12734

Рисунок Д.13 – Апробація результатів роботи та публікації

ВИСНОВКИ

У бакалаврській дипломній роботі було розроблено мобільний програмний додаток для вивчення математики дітьми молодшої школи «MathForKids», шляхом розв'язування прикладів та задач з різними діями і різними рівнями складності.

Було проведено аналіз предметної області даного питання. Також, був проведений аналіз аналогів розроблюваного програмного продукту для вивчення математики дітьми молодшої школи, а також застосування цих знань на практиці. В результаті порівняння аналогів із власним програмним продуктом, було доведено доцільність розробки власного програмного продукту для вивчення дітьми математики. Також було виконано постановку задач розробки програмного продукту.

Було розроблено методи та модель роботи програми, розроблено алгоритм побудови головного вікна мобільного додатку. Також, було розроблено покроковий алгоритм роботи програми, алгоритм генерації нового прикладу та алгоритм зчитування задач з файлу. Крім того, було розроблено блок-схеми цих алгоритмів.

Рисунок Д.14 – Висновки (перша частина)

ВИСНОВКИ

Було проведено варіантний аналіз та обґрунтування вибору засобів реалізації програмного продукту. В результаті даного аналізу, в якості мови програмування було обрано мову програмування Kotlin, а в якості середовища розробки Android Studio.

В процесі розробки було обґрунтовано та реалізовано інтерфейс мобільного додатку «MathForKids». Також, було описано програмну реалізацію основних класів програми.

Було проаналізовано основні методи тестування програмного забезпечення, та в результаті даного аналізу було обрано методику «black-box testing». Тестування, яке проводилось за допомогою тест-кейсів, довело повну працездатність розробленого програмного додатку та його відповідність поставленому технічному завданню. Крім того, було розроблено інструкцію користувача.

Рисунок Д.15 – Висновки (друга частина)