МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВОЛИНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ЛЕСІ УКРАЇНКИ
Кафедра комп'ютерних наук та кібербезпеки

На правах рукопису

ЛИТВИНЧУК ЯРОСЛАВ ВОЛОДИМИРОВИЧ

РОЗРОБКА ANDROID-ДОДАТКУ ДЛЯ ГЕНЕРАЦІЇ МАТЕМАТИЧНИХ ПРИКЛАДІВ НА МОВІ ПРОГРАМУВАННЯ KOTLIN

Спеціальність: 122 Комп'ютерні науки
Освітньо-професійна програма: Комп'ютерні науки та інформаційні технології
Кваліфікаційна робота на здобуття освітнього ступеня «бакалавр»

Науковий керівник:
МАМЧИЧ ТЕТЯНА ІВАНІВНА,
кандидат фізико-математичних наук, доцент
кафедри комп'ютерних наук та кібербезпеки

РЕКОМЕНДОВАНО ДО ЗАХИСТУ
Протокол №_____
засідання кафедри комп'ютерних наук
та кібербезпеки
від _____ 2024 р.
Завідувач кафедри

(_____) Гришанович Т. О.

ЛУЦЬК – 2024

ЗМІСТ

ВСТУП

Актуальність теми. Сучасний етап розвитку інформаційних технологій характеризується зростаючим впливом мобільних додатків на повсякденне життя та освітню діяльність. Мобільні платформи, такі як Android, є найпоширенішими серед користувачів, що зумовлює необхідність створення якісних додатків для різних сфер, включаючи освіту. Освітні мобільні додатки відіграють важливу роль у самостійному навчанні та підготовці вчителів до занять, особливо у вивченні таких базових дисциплін, як математика. Генерація математичних прикладів є важливим компонентом освітнього процесу, оскільки забезпечує практичне відпрацювання теоретичних знань та стимулює розвиток логічного мислення. Попри наявність значної кількості математичних додатків, існує потреба в розробці рішень, що забезпечують адаптивність, простоту використання, інтерактивність і відповідність сучасним стандартам розробки програмного забезпечення. У цьому контексті використання мови програмування Kotlin, яка є офіційною мовою розробки для платформи Android, дозволяє створювати продуктивні та зручні у використанні мобільні додатки. Kotlin забезпечує підвищену ефективність коду, зручність у роботі з бібліотеками та сучасними фреймворками, що робить його актуальним вибором для реалізації освітніх рішень. Таким чином, тема є актуальною, оскільки відповідає потребам сучасного суспільства у впровадженні інноваційних освітніх технологій, сприяє популяризації математики та покращує якість освітнього процесу завдяки використанню новітніх програмних підходів.

Мета роботи – розробити Android-додаток для генерації математичних прикладів на мові програмування Kotlin.

Для досягнення поставленої мети потрібно:

- провести аналіз існуючих рішень для генерації математичних прикладів у мобільних додатках з акцентом на їх функціональні можливості, недоліки та переваги;
- визначити вимоги до функціоналу, інтерфейсу та технічної реалізації Android-додатку;

- розробити архітектуру додатку з урахуванням специфіки роботи на мові програмування Kotlin;
- провести тестування розробленого додатку для виявлення помилок та оцінки його продуктивності;
- розробити рекомендації щодо використання додатку в освітньому процесі та можливості його подальшого розвитку.

Об’єкт дослідження – освітні Android-додатки.

Предмет дослідження – технології розробки Android-додатків для генерації математичних прикладів.

Апробація результатів роботи – тези було опубліковано у збірнику І Міжнародної науково-практичної конференції «Проблеми комп’ютерних наук, програмного моделювання та безпеки цифрових систем» (13 – 16 червня 2024 р).

РОЗДІЛ 1.
ТЕХНОЛОГІЇ РОЗРОБКИ ANDROID-ДОДАТКІВ

1.1 Особливості використання сучасних технологій Kotlin для розробки Android-додатків

Kotlin – це статично типізована мова програмування, розроблена компанією JetBrains і представлена у 2011 році. Вона створена для вирішення типових задач сучасного програмування, зокрема розробки мобільних додатків, серверних систем, веб-програм і навіть роботи з багатоплатформними проєктами. З 2017 року Kotlin став офіційною мовою для розробки Android-додатків, що підтвердило його значення в екосистемі мобільної розробки.

Kotlin має низку основних характеристик, що роблять її потужним інструментом для розробки на різних платформах, зокрема для Android. Однією з головних переваг Kotlin є її сумісність із Java. Оскільки Kotlin повністю інтегрується з Java, вона дозволяє програмістам використовувати всі існуючі Java-бібліотеки та фреймворки, що робить перехід на Kotlin плавним і поступовим. Це також дозволяє поєднувати код на Java і Kotlin в одному проєкті без необхідності переписувати існуючі компоненти, що значно знижує витрати на перехід на нову мову. Kotlin також дозволяє інтегруватися з іншими мовами і платформами, що робить її універсальним інструментом для розробки.

Однією з найбільших переваг Kotlin є її лаконічність. Код на Kotlin значно коротший і більш зрозумілий у порівнянні з Java, оскільки Kotlin підтримує сучасні синтаксичні конструкції, такі як data class (клас для збереження даних), який автоматично генерує методи toString(), equals(), hashCode(), а також copy() для копіювання об’єктів. Крім того, Kotlin має функції для роботи з колекціями, лямбда-вирази та розширення функцій, що дають можливість розширювати існуючі класи без їх зміни. Всі ці можливості дозволяють писати менш об’ємний, а отже, більш підтримуваний код.

Kotlin приділяє значну увагу безпеці коду, зокрема щодо обробки null-значень, що є однією з найбільших проблем у Java. Kotlin вводить концепцію

Nullable та Non-nullable типів, що дає змогу розробникам чітко визначати, які змінні можуть бути null, а які – не можуть. Ця особливість значно зменшує кількість помилок типу NullPointerException, які можуть виникати під час виконання програми. Завдяки цьому, код на Kotlin стає більш безпечним та стійким до помилок, пов’язаних із неправильним використанням null-значень.

Kotlin активно підтримує функціональне програмування, яке дозволяє використовувати лямбда-функції, функції вищого порядку, та працювати з колекціями через функції в стилі "map", "filter", "reduce". Це дозволяє створювати чистий і зручний код, що підтримує ідеї функціонального програмування, даючи змогу маніпулювати даними без зміни їхнього стану. Завдяки цьому програмісти можуть писати більш ефективний код, який легко тестується та підтримується.

Окрім того, Kotlin має властивість кросплатформеності завдяки Kotlin Multiplatform, що дозволяє розробляти один і той самий код для різних платформ, таких як Android, iOS, сервери (Kotlin/JS, Kotlin/Native). Це дає змогу створювати багатоплатформні програми, де загальна бізнес-логіка та алгоритми можуть бути використані на різних платформах, а специфічний для кожної платформи код може бути реалізований окремо. Це значно знижує витрати на розробку та підтримку багатоплатформних проєктів, дозволяючи програмістам писати код один раз і використовувати його на різних пристроях.

Kotlin також підтримує розширення класів (extension functions), що дозволяє додавати нову функціональність до існуючих класів без необхідності їх модифікації. Цей механізм дає змогу реалізувати додаткові функції для будь-якого класу, навіть якщо ви не маєте доступу до його вихідного коду. Наприклад, за допомогою extension functions можна додати нові методи для стандартних бібліотек Kotlin або Java. Це дає змогу робити код більш гнучким і зручним для подальшого розширення.

Також Kotlin підтримує роботу з асинхронними потоками через бібліотеку Coroutines, що дозволяє обробляти паралельні та асинхронні операції більш зручно і ефективно, ніж традиційний підхід через багатопоточність або обробку зворотних викликів. Coroutines спрощують синхронізацію коду, даючи змогу

писати асинхронний код у вигляді лінійного, без необхідності керувати складними потоками вручну.

Kotlin має низку переваг, які роблять його ідеальним вибором для Android-розробки. Однією з найбільш значущих переваг є офіційна підтримка Google. З 2017 року Google оголосила Kotlin однією з основних мов для розробки Android-додатків, що забезпечило мовою статус основного інструменту для мобільної розробки на платформі Android. Це означає, що Kotlin отримує постійну підтримку та оновлення від Google, а також інтеграцію з Android SDK, що дозволяє розробникам скористатися найсучаснішими можливостями для створення додатків. Офіційна підтримка Kotlin відкрила шлях до широкої екосистеми бібліотек і фреймворків, які забезпечують ефективний і зручний процес розробки. Крім того, розробники можуть скористатися активним ком'юніті Kotlin, яке швидко росте і надає велику кількість відкритих ресурсів та інструментів. Суттєвою перевагою Kotlin є підтримка сучасних фреймворків

1.1.1 Jetpack Compose: сучасний підхід до створення UI в Android

Jetpack Compose – це декларативний фреймворк для створення інтерфейсів користувача в Android, який розроблений компанією Google. Він дозволяє розробникам створювати користувацькі інтерфейси без необхідності писати значну кількість XML-коду, що є традиційним способом створення інтерфейсів у класичній Android-розробці. Jetpack Compose значно спрощує і прискорює процес створення UI, роблячи код більш зрозумілим, чистим і легко підтримуваним.

Основною ідеєю Jetpack Compose є декларативний підхід до створення інтерфейсу. Це означає, що замість того, щоб програмно створювати та оновлювати елементи інтерфейсу в коді (як це було в класичному Android за допомогою XML та класів, таких як TextView, Button, тощо), розробники описують, як інтерфейс має виглядати, а система сама стежить за його станом і відображенням. Це дозволяє зменшити складність коду, оскільки не потрібно

вручну оновлювати елементи інтерфейсу після кожної зміни в даних або стані програми.

Jetpack Compose працює на основі сумісності з Kotlin, що робить його інтеграцію з цією мовою зручною та логічною. Використовуючи Kotlin, Jetpack Compose дозволяє створювати функціональні компоненти інтерфейсу, що визначаються як функції. Вони описують, як виглядатиме UI для конкретного стану програми. Завдяки цьому підходу, весь інтерфейс можна написати без використання XML, що значно спрощує структуру додатка, знижує кількість помилок і дає більше контролю над виглядом інтерфейсу.

Однією з головних переваг Jetpack Compose є спрощення роботи з UI-компонентами. Наприклад, елементи інтерфейсу, такі як кнопки, текстові поля, списки, зображення тощо, можна створювати за допомогою простих Kotlin-функцій. Ці компоненти можна легко налаштовувати, змінювати їхній вигляд або стилі через параметри функцій. Наприклад, створення кнопки за допомогою Compose виглядатиме так:

```
Button(onClick = { /* дія при натисканні */ }) {
Text("Натисни мене")
}
```

Це просте й зрозуміле визначення елемента інтерфейсу, яке не потребує додаткового XML-коду або складної логіки для його відображення. Розробники можуть комбінувати ці функціональні елементи, створюючи складні інтерфейси з мінімальною кількістю коду.

Jetpack Compose також дозволяє легко працювати з станами інтерфейсу. Наприклад, за допомогою State та MutableState можна змінювати вигляд інтерфейсу залежно від введених даних або дій користувача. Завдяки цьому, коли змінюється стан даних або бізнес-логіки, Jetpack Compose автоматично оновлює відповідні частини інтерфейсу, що значно спрощує управління UI і підвищує його ефективність.

Іншою важливою характеристикою є інтеграція з іншими частинами Android Jetpack. Jetpack Compose тісно інтегрується з іншими бібліотеками Jetpack, такими як Navigation, LiveData, Paging, ViewModel тощо. Це дозволяє ефективно використовувати ці бібліотеки для управління станами, асинхронними операціями, навігацією між екранами, роботою з базами даних і обробкою великих обсягів даних.

Jetpack Compose також підтримує матеріальний дизайн (Material Design), що дозволяє розробникам легко створювати додатки з сучасним і зручним інтерфейсом. Бібліотека надає безліч готових компонентів, таких як кнопки, картки, меню, діалогові вікна, які повністю відповідають принципам матеріального дизайну і легко налаштовуються за допомогою параметрів функцій.

Ще однією перевагою є чудова підтримка анімацій. Jetpack Compose має зручні інструменти для створення плавних анімацій і ефектів для елементів інтерфейсу, таких як зміна розміру, кольору, позиції або прозорості. Це дозволяє розробникам створювати інтуїтивно зрозумілий і привабливий інтерфейс з анімованими переходами і ефектами без необхідності використовувати сторонні бібліотеки або складний код.

Завдяки своїй гнучкості, простоті та сумісності з Kotlin, Jetpack Compose дозволяє створювати високоякісні, масштабовані додатки для Android з мінімальними витратами часу на розробку та обслуговування коду. Він змінює підхід до розробки UI, спрощуючи процес створення інтерфейсів, одночасно підвищуючи їх продуктивність та зручність для кінцевого користувача. Jetpack Compose є надійним інструментом, який відкриває нові можливості для розробників, дозволяючи їм створювати інтуїтивно зрозумілі, швидкі та ефективні Android-додатки.

1.1.2 Coroutines в Kotlin: ефективне управління асинхронними операціями

Coroutines – це інструмент у Kotlin, що дозволяє розробникам ефективно та зручно працювати з асинхронними операціями, зберігаючи при цьому простоту і зрозумілість коду. Вони дозволяють реалізувати паралельне виконання задач без необхідності використовувати складні механізми багатопоточності, такі як Thread, що часто призводить до проблем із синхронізацією, управлінням потоками та помилками в коді. Coroutines спрощують написання асинхронного коду, роблячи його більш лінійним та зручним для розуміння.

Основна ідея Coroutines полягає в тому, щоб дозволити виконувати асинхронні операції у вигляді послідовних кроків, без блокування основного потоку. Вони використовують концепцію контексту та сфер виконання, що дає змогу керувати задачами та їх виконанням у різних потоках. Таким чином, співпрограми дозволяють писати асинхронний код, який виглядає як звичайний лінійний код, зберігаючи при цьому всі переваги асинхронного виконання, такі як неблокуючі операції.

Однією з головних переваг Coroutines є те, що вони не блокують потоки, а замість цього використовують саспенс-функції (suspending functions), які можуть призупиняти виконання співпрограм на певний час без блокування потоку. Коли співпрограма призупинена, інші співпрограми можуть продовжити виконання, що дозволяє ефективно використовувати ресурси системи і працювати з великою кількістю паралельних задач без необхідності створювати нові потоки для кожної з них. Співпрограма може бути знову відновлена, коли виконання операції буде завершено, і це відновлення відбудеться швидко та ефективно.

Для того, щоб позначити, що функція є асинхронною і може бути призупинена, використовують ключове слово suspend. Співпрограма може викликати іншу співпрограму або саспенс-функцію, що дозволяє легко обробляти складні асинхронні операції в лінійному вигляді. Це дозволяє уникнути так званого "callback hell" (пекло зворотних викликів), яке часто

виникає при використанні традиційних механізмів зворотних викликів для обробки асинхронних операцій.

Крім того, співпрограми використовують механізм контексту співпрограми. Кожна співпрограма виконується в певному контексті, який визначає, який потік або диспетчер буде використовуватися для виконання цієї співпрограми. Це дозволяє розробникам контролювати, де саме буде виконуватися код, що є особливо корисним для Android-розробки, де важливо забезпечити, щоб певні операції виконувалися на головному потоці (наприклад, оновлення інтерфейсу користувача), а інші — у фонових потоках (наприклад, завантаження даних із мережі).

Для запуску співпрограм у Kotlin використовуються такі основні інструменти, як CoroutineScope та launch. CoroutineScope визначає область, в межах якої виконуються співпрограми, а launch дозволяє ініціювати нову співпрограму. Це дозволяє створювати нові співпрограми, що виконуються паралельно з основним потоком. Однак важливо зауважити, що GlobalScope не є найкращим варіантом для управління співпрограмами у великих програмах, оскільки він живе протягом усього життєвого циклу програми. Краще використовувати контексти, специфічні для компонентів програми, такі як viewModelScope для ViewModel у Android або lifecycleScope для Activity або Fragment.

Однією з основних переваг співпрограм є управління життєвим циклом. Замість того, щоб вручну обробляти завдання на основі життєвого циклу компонентів (наприклад, зупиняти або скасовувати потоки при знищенні активності або фрагмента), співпрограми дозволяють зручно працювати з цими аспектами за допомогою таких властивостей, як lifecycleScope або viewModelScope. Ці обмеження автоматично зупиняють виконання співпрограм, коли компоненти їх більше не потребують, що запобігає витокам пам'яті.

Для обробки помилок у співпрограмах Kotlin надає механізми, такі як try-catch блоки для обробки виключень, що виникають у співпрограмах. Це дає змогу зручно обробляти помилки, які можуть виникнути під час виконання

асинхронних задач, та виконувати відповідні дії у разі виникнення помилок без необхідності складної обробки потоків.

Coroutines також підтримують паралельне виконання за допомогою функцій, таких як async і await, що дозволяють запускати кілька співпрограм одночасно і чекати на їх завершення. Завдяки цьому розробники можуть обробляти кілька незалежних задач паралельно, що значно прискорює виконання операцій, таких як завантаження даних з кількох джерел або обробка великих наборів даних.

Завдяки всім цим можливостям, Coroutines в Kotlin є ідеальним рішенням для обробки асинхронних операцій і паралельних задач, надаючи розробникам функціональний і зручний інструмент для написання чистого, зрозумілого та ефективного асинхронного коду.

1.1.3 View Binding: ефективний спосіб взаємодії з елементами інтерфейсу в Android

View Binding — це інструмент, який спрощує доступ до елементів користувацького інтерфейсу (UI) в Android-додатках. Він дозволяє уникнути традиційного методу використання методу findViewById(), який часто був джерелом помилок і незручностей у великих проектах. Використання View Binding забезпечує більш безпечний, ефективний та зрозумілий спосіб роботи з елементами інтерфейсу, знижуючи ризик помилок під час виконання програми.

Принцип роботи View Binding:

View Binding автоматично генерує класи-зв’язки для кожного XML-файлу макета в проекті. Ці класи створюються під час компіляції і містять посилання на всі елементи, визначені в макеті. Ім'я класу-зв’язку формується шляхом перетворення назви XML-файлу в PascalCase і додавання слова "Binding". Наприклад, якщо макет має назву activity_main.xml, то View Binding згенерує клас ActivityMainBinding. Для використання View Binding у проекті необхідно увімкнути View Binding у файлі build.gradle додатку, додавши такі рядки до секції android:

```
viewBinding {
enabled = true
}
```

У коді активності або фрагмента створити екземпляр класу-зв’язку, використовуючи метод inflate або bind. Наприклад, для активності це виглядає так:

```
val binding = ActivityMainBinding.inflate(layoutInflater)
setContentView(binding.root)
```

Тепер елементи інтерфейсу з XML-файлу доступні безпосередньо через властивості об'єкта binding. Наприклад, якщо XML-файл містить елемент TextView з ідентифікатором textView, то доступ до нього здійснюється через binding.textView.

Переваги використання View Binding це: безпека, зручність і простота використання, відсутність помилок NullPointerException, легка інтеграція з фрагментами, сумісність з комплексними макетами та висока продуктивність

View Binding забезпечує безпечний доступ до елементів інтерфейсу. Це означає, що під час компіляції перевіряється правильність ідентифікаторів елементів і їх типів. У разі помилок розробник отримає попередження ще до запуску програми, що значно знижує кількість runtime-помилок. Використання View Binding позбавляє розробників необхідності вручну викликати findViewById() для кожного елемента. Це зменшує обсяг коду і робить його більш читабельним. Наприклад:

```
// Без View Binding
val textView: TextView = findViewById(R.id.textView)
textView.text = "Привіт, світ!"

// З View Binding
binding.textView.text = "Привіт, світ!"
```

Завдяки тому, що View Binding гарантує наявність усіх елементів, визначених у XML-файлі, значно знижується ризик помилок

NullPointerException. Наприклад, у випадку помилково видаленого елемента з XML View Binding автоматично вкаже на проблему під час компіляції. View Binding значно спрощує роботу з фрагментами. Створенням та знищенням об'єктів зв’язку можна легко управляти за допомогою життєвого циклу фрагмента. Наприклад:

```
private var _binding: FragmentExampleBinding? = null
private val binding get() = _binding!!

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

override fun onDestroyView() {
super.onDestroyView()
binding = null
}
```

Такий підхід запобігає витокам пам'яті, оскільки об'єкт зв’язку очищається, коли фрагмент знищується. View Binding підтримує складні макети з вкладеними структурами. Усі елементи макета, доступні в XML, автоматично включаються у згенерований клас зв’язку, що робить їх доступними для маніпуляцій у коді. Оскільки View Binding генерує класи-зв’язки під час компіляції, їх використання є ефективнішим у порівнянні з іншими підходами, такими як findViewById() або ButterKnife, які виконують пошук елементів під час виконання.

Загалом View Binding — це сучасний і зручний інструмент для роботи з елементами інтерфейсу в Android-додатках, що поєднує простоту використання з високою продуктивністю і безпекою. Завдяки безпеці, автоматизації

зв’язування елементів і легкій інтеграції в проекти View Binding став незамінним інструментом для розробників, які прагнуть створювати ефективні, чисті та надійні додатки.

1.1.4 RecyclerView: гнучкий інструмент для відображення динамічних списків в Android

RecyclerView – це один із найпотужніших і найбільш гнучких компонентів для роботи з динамічними списками та колекціями даних в Android. Він був розроблений як заміна ListView та GridView, які мали суттєві обмеження щодо продуктивності та кастомізації. RecyclerView надає широкий набір можливостей для створення складних, адаптивних інтерфейсів, дозволяючи ефективно працювати з великими обсягами даних.

Основний принцип роботи RecyclerView базується на повторному використанні представлень (views), що значно знижує навантаження на систему та покращує продуктивність. Замість створення нового об'єкта для кожного елемента списку, RecyclerView повторно використовує існуючі представлення, адаптуючи їх до нових даних. Це досягається завдяки використанню класу ViewHolder, який виступає в ролі контейнера для посилань на елементи інтерфейсу. ViewHolder дозволяє зберігати посилання на представлення, що мінімізує кількість викликів методу findViewById() і прискорює роботу додатку.

RecyclerView працює у тісному зв'язку з адаптером, який визначає, як дані відображаються в списку. Адаптер відповідає за створення нових ViewHolder і заповнення їх даними, які отримуються з колекції. Таким чином, розробник може повністю контролювати вигляд кожного елемента списку та його взаємодію з користувачем. RecyclerView також підтримує різноманітні макети, які визначають, як розташовуються елементи на екрані. Стандартні макети, такі як LinearLayoutManager, GridLayoutManager і StaggeredGridLayoutManager, дозволяють створювати вертикальні та горизонтальні списки, а також сітки або нестандартні розташування елементів.

Однією з ключових особливостей RecyclerView є його підтримка анімацій при додаванні, видаленні або переміщенні елементів. Це дозволяє створювати більш динамічні та інтерактивні інтерфейси. Для кастомізації анімацій можна використовувати спеціальні об'єкти ItemAnimator. Крім того, RecyclerView легко інтегрується з GestureDetector або іншими компонентами для реалізації складних жестів, таких як змахи або довге натискання.

RecyclerView також підтримує розділювачі між елементами списку за допомогою ItemDecoration. Це дозволяє розробникам додавати візуальні роздільники, відступи або інші графічні елементи між елементами списку. Додаткова функціональність, така як підвантаження даних під час перегляду або інтеграція з базами даних, реалізується через механізми, такі як ScrollListener або Paging Library, що забезпечує плавність роботи навіть із великими обсягами даних.

RecyclerView також добре масштабується і є оптимальним вибором для додатків, які вимагають високої продуктивності, кастомізації та інтерактивності у відображенні списків. Завдяки своїй модульній архітектурі він дозволяє розробникам створювати складні та адаптивні компоненти інтерфейсу, які відповідають потребам сучасних додатків.

1.1.5 Адаптери в Android: зв’язок даних із інтерфейсом

Адаптери (Adapters) є важливим елементом у розробці Android-додатків, які використовуються для відображення даних у компонентах інтерфейсу, таких як списки, сітки чи каруселі. Вони відіграють роль посередника між джерелом даних і елементами користувацького інтерфейсу, забезпечуючи перетворення даних у візуальні компоненти. Основна функція адаптера полягає в тому, щоб отримати дані, перетворити їх у вигляд, що відповідає інтерфейсу, і передати ці дані для відображення.

Адаптери працюють за наступною схемою: спочатку вони отримують вхідні дані, які можуть бути представлені у вигляді списків, масивів або складніших структур. Потім адаптер виконує логіку перетворення цих даних у

вигляд, що відповідає вимогам елементів інтерфейсу. Наприклад, адаптер створює макети для кожного пункту списку та прив’язує до них відповідні дані.

Основою будь-якого адаптера є його методи, які відповідають за створення і оновлення елементів списку. Метод onCreateViewHolder викликається для створення нового контейнера для елемента списку. Цей контейнер часто називають ViewHolder, і він містить посилання на компоненти макета одного елемента, наприклад, текстові поля, зображення або кнопки. Метод onBindViewHolder використовується для заповнення контейнера даними. У ньому відбувається прив’язка інформації з джерела даних до конкретного елемента списку. Крім того, метод getItemCount визначає кількість елементів, які потрібно відобразити.

Адаптери підтримують різні типи даних і макетів. Наприклад, стандартний ArrayAdapter використовується для роботи з масивами або списками рядків. Якщо розробник працює з більш складними макетами чи нестандартними типами даних, він може створити власний кастомний адаптер, наслідуючи клас BaseAdapter або RecyclerView.Adapter.

Головна перевага адаптерів полягає в їхній гнучкості. Вони дозволяють розробникам створювати різноманітні типи відображень даних, починаючи від простих списків тексту і закінчуючи багаторівневими складними інтерфейсами з інтерактивними елементами. Крім того, адаптери інтегруються з компонентами, такими як RecyclerView або Spinner, забезпечуючи динамічну роботу додатків. Наприклад, якщо джерело даних змінюється, адаптер може автоматично оновлювати інтерфейс, викликавши методи notifyDataSetChanged або notifyItemChanged.

Для оптимізації продуктивності адаптери підтримують концепцію повторного використання елементів. Це означає, що старі елементи, які виходять за межі видимості, можуть бути використані повторно для відображення нових даних. Цей підхід значно скорочує витрати на створення нових об’єктів і підвищує ефективність додатку, особливо при роботі з великими списками даних.

Однак робота з адаптерами може вимагати значної кількості коду і зусиль, особливо для створення кастомних рішень. Крім того, при реалізації складних списків або сіток важливо уважно налаштовувати оптимізацію, щоб уникнути перевантаження пам’яті чи уповільнення роботи програми.

Адаптери є незамінним інструментом для Android-розробки, дозволяючи створювати гнучкі та інтерактивні інтерфейси, що відображають динамічні дані. Вони забезпечують адаптивність додатків до різних типів контенту та розширюють можливості управління даними в мобільних програмах.

Таким чином, Kotlin поєднує в собі потужність статичної типізації, можливості функціонального програмування, безпеку коду, сумісність з Java та підтримку багатоплатформних проєктів. Це робить її однією з найкращих мов для розробки сучасних додатків, особливо для мобільних платформ, таких як Android.

1.2 Material Design 3 як новий етап у дизайні інтерфейсів Android

Material Design 3 (M3), також відомий як Material You, — це остання ітерація дизайну від Google, яка розширює концепції попередніх версій, зосереджуючись на персоналізації, адаптивності та інклюзивності. Цей підхід дозволяє створювати більш індивідуалізовані інтерфейси, що враховують уподобання користувача та забезпечують єдиний стиль на всіх пристроях Android.

Основна особливість Material Design 3 полягає у його здатності адаптуватися до кольорової палітри, обраної користувачем. Використовуючи алгоритм колірної гармонії, система автоматично генерує палітру, базуючись на шпалерах пристрою або інших джерелах кольору. Це створює унікальний, естетично привабливий вигляд кожного додатку, який гармонійно вписується у загальний дизайн системи.

Material Design 3 підтримує покращену адаптивність для різних типів пристроїв і розмірів екранів. Компоненти інтерфейсу автоматично

масштабуються і налаштовуються, забезпечуючи зручність використання як на мобільних телефонах, так і на планшетах, пристроях із великими екранами чи складаних гаджетах. Це дозволяє створювати більш універсальні та зручні додатки.

Одним із центральних елементів M3 є підтримка покращеної типографії. Нова система типографії використовує варіативні шрифти, які адаптуються до різних розмірів екранів і забезпечують кращу читабельність тексту. Крім того, Material Design 3 пропонує оновлені стилі шрифтів, що сприяють узгодженому вигляду всього інтерфейсу.

Material Design 3 також вдосконалює інтерактивні елементи, такі як кнопки, меню, слайдери та інші компоненти. Всі вони відповідають концепції динамічного дизайну, що реагує на дії користувача. Анімації стали більш плавними та природними, що покращує загальне враження від взаємодії з додатком.

Інклюзивність є важливим аспектом Material Design 3. Дизайн враховує потреби користувачів із різними рівнями фізичних можливостей, забезпечуючи підтримку висококонтрасних режимів, збільшених розмірів тексту та доступності для екранних читачів. Завдяки цьому M3 дозволяє створювати інтерфейси, доступні для максимальної кількості користувачів.

Розробникам Android-додатків Material Design 3 пропонує широкий набір інструментів і компонентів для створення сучасних інтерфейсів. Jetpack Compose, основний фреймворк для декларативного UI-програмування, має повну підтримку Material Design 3, що дозволяє легко інтегрувати нові елементи та функціональність. Удосконалена документація та готові шаблони сприяють швидкому впровадженню принципів M3 у проекти.

У порівнянні з попередніми версіями, Material Design 3 пропонує більше свободи у створенні індивідуальних дизайнів, водночас забезпечуючи узгодженість і функціональність. Персоналізація, адаптивність і доступність роблять M3 ключовим інструментом для створення інтерфейсів, які не лише естетично привабливі, а й орієнтовані на потреби користувача. Ця система задає

нові стандарти для мобільного дизайну, забезпечуючи сучасні додатки інструментами для інтеграції стильних, адаптивних і доступних інтерфейсів.

1.3 GitHub Libraries як основний інструмент для інтеграції сторонніх рішень у проекти Android

GitHub Libraries (або просто GitHub Libs) є незамінною частиною сучасної розробки Android-додатків, дозволяючи розробникам використовувати перевірені часом та спільнотою програмні рішення. Ці бібліотеки охоплюють широкий спектр функціональних можливостей, які допомагають спрощувати і прискорювати процес створення додатків, одночасно забезпечуючи високу якість реалізації.

GitHub виступає однією з найбільших платформ для розміщення відкритого вихідного коду, де можна знайти тисячі бібліотек для Android. Бібліотеки надають готові модулі, які можна інтегрувати у проєкти, зменшуючи необхідність розробки з нуля. Це можуть бути інструменти для роботи з інтерфейсом, обробки даних, управління мережевими запитами, а також для реалізації анімацій, роботи з базами даних або інтеграції хмарних сервісів.

Однією з ключових переваг використання GitHub Libs є їх активна підтримка та розвиток спільнотою розробників. Це забезпечує своєчасне оновлення бібліотек, виправлення помилок і додавання нових функціональних можливостей. Багато популярних бібліотек, таких як Retrofit для роботи з мережевими запитами, Glide або Picasso для завантаження і обробки зображень, а також Room для роботи з базами даних, отримують регулярні оновлення, які враховують останні зміни у платформі Android.

Інтеграція бібліотек із GitHub у проект Android реалізується за допомогою систем управління версіями, таких як Gradle. Це дозволяє легко додавати необхідні бібліотеки, вказуючи лише їх залежність у конфігураційному файлі build.gradle. Більш того, Gradle автоматично завантажує останню доступну версію бібліотеки або визначену розробником стабільну версію.

GitHub Libs мають кілька важливих переваг для розробників. По-перше, вони суттєво скорочують час розробки, оскільки більшість завдань вже вирішені у вигляді готових модулів. Наприклад, замість створення складної логіки для роботи з API, розробник може використовувати бібліотеку Retrofit, яка надає простий і зрозумілий інтерфейс для виконання HTTP-запитів. По-друге, використання перевірених бібліотек сприяє підвищенню надійності додатків, адже більшість із них мають великий обсяг тестів і широко використовуються у багатьох реальних проектах.

GitHub Libs дозволяють розробникам створювати якісні Android-додатки, використовуючи готові, добре протестовані рішення. Це економить час і ресурси, забезпечуючи при цьому можливість зосередитися на унікальності проєкту, а не на базовій інфраструктурі. Завдяки GitHub Libraries розробники отримують доступ до глобальної спільноти, новітніх технологій та ефективних інструментів, які допомагають створювати сучасні мобільні додатки.

1.4 Система автоматизації збірки та управління залежностями Gradle

Gradle є потужною системою автоматизації збірки, яка стала стандартом для розробки Android-додатків. Вона дозволяє автоматизувати процеси збірки, управління залежностями, тестування, а також запуску та розгортання програмних продуктів. Gradle спрощує і прискорює створення додатків, забезпечуючи можливість налаштування та оптимізації кожного етапу розробки. Основною особливістю Gradle є його здатність працювати з різними типами залежностей і забезпечувати гнучкість в управлінні ними. Завдяки підтримці мови скриптів на базі Groovy або Kotlin, Gradle дозволяє створювати конфігураційні файли, які визначають, як повинні бути побудовані проекти, і як будуть керуватися їхні залежності. Це дозволяє розробникам зручно описувати складні процеси збірки, зокрема налаштування для конкретних варіантів збірки для різних середовищ чи платформ.

Gradle дозволяє працювати з різними джерелами залежностей, такими як локальні бібліотеки, бібліотеки в репозиторіях Maven, JCenter чи Google Maven. За допомогою конфігураційного файлу build.gradle розробники можуть вказати зовнішні бібліотеки та плагіни, які автоматично завантажуються під час процесу збірки, що значно спрощує інтеграцію сторонніх рішень у проєкт. Окрім цього, Gradle підтримує механізм кешування, що дозволяє не завантажувати одну й ту ж бібліотеку щоразу, що підвищує ефективність збірки.

Gradle використовує концепцію задач (tasks), кожна з яких виконує певну дію, наприклад, компіляцію коду, запуск тестів, створення APK-файлів чи інші операції. Завдяки цій модульності, Gradle дозволяє максимально кастомізувати процес збірки, додаючи або змінюючи задачі, які виконуються на різних етапах. Це дає можливість організовувати складні, багатоступеневі процеси збірки без необхідності вручну виконувати кожен етап.

Крім того, Gradle є ефективним інструментом для підтримки багатьох варіантів збірки. Наприклад, для одного і того ж проекту можна налаштувати різні конфігурації для розробки, тестування та випуску. Це дозволяє автоматично підключати різні ресурси та налаштування залежно від середовища, в якому буде використовуватись додаток. Варіанти збірки можуть включати різні версії бібліотек, різні API-ключі, файли конфігурації тощо.

Gradle інтегрується з іншими інструментами Android-розробки, такими як Android Studio, що робить процес збірки максимально зручним. Наприклад, Android Studio надає вбудовану підтримку для Gradle, що дозволяє запускати задачі безпосередньо з середовища розробки, а також використовувати графічний інтерфейс для налаштування параметрів збірки.

Однією з ключових переваг Gradle є його масштабованість. Від простих проєктів до великих, складних додатків з багатьма модулями — Gradle може бути налаштований так, щоб задовольнити будь-які вимоги. Це дозволяє розробникам зберігати контроль над збіркою навіть у великих командах та на етапах, коли проєкт розростається.

Gradle є незамінним інструментом для розробки Android-додатків, забезпечуючи автоматизацію та оптимізацію процесу збірки. Його гнучкість, можливість масштабування, а також підтримка багатьох варіантів збірки роблять його основним інструментом для розробників Android, що прагнуть до ефективного та організованого процесу розробки.

1.5 Головний інструмент для розробки мобільних додатків - Android SDK

Android SDK (Software Development Kit) — це набір інструментів і бібліотек, призначених для розробки мобільних додатків на платформі Android. SDK є основною частиною екосистеми Android, надаючи розробникам усе необхідне для створення, тестування та оптимізації додатків для Android-пристроїв. Він включає в себе різноманітні компоненти, починаючи від інтерфейсів для роботи з пристроями та закінчуючи інструментами для налагодження і профілювання додатків.

Одним із основних елементів Android SDK є Android Studio — офіційне середовище розробки (IDE) для створення мобільних додатків для операційної системи Android. Розробка Android Studio була ініційована компанією Google у 2013 році, і з того часу це середовище стало основним інструментом для розробників Android-додатків, об’єднавши безліч функціональних можливостей для програмування, тестування та налагодження.

Android Studio побудоване на платформі IntelliJ IDEA від JetBrains, що надає потужний редактор коду з багатою підтримкою для програмування на мовах Java і Kotlin. Це середовище дозволяє автоматизувати багато аспектів розробки, значно спрощуючи процес створення мобільних додатків, що робить його ідеальним вибором для Android-розробників.

Однією з ключових функцій Android Studio є інтеграція з Android SDK, що забезпечує доступ до всіх необхідних інструментів і бібліотек для створення Android-додатків. Завдяки такій інтеграції, Android Studio автоматично пропонує

розробникам необхідні бібліотеки, оновлення SDK, плагіни та інструменти для оптимізації додатків.

Серед основних особливостей Android Studio можна виділити зручний редактор коду з автодоповненням, рефакторингом і підсвічуванням синтаксису. Це дає змогу зменшити кількість помилок у коді та прискорити написання програмного забезпечення. Розробники можуть зручно працювати з проєктами на Java, Kotlin та XML, а також мають змогу інтегрувати сторонні бібліотеки, що значно розширює функціональність їхніх додатків.

Однією з найбільш важливих можливостей Android Studio є інструмент для створення користувацьких інтерфейсів — Layout Editor. Це візуальний редактор, який дозволяє розробникам проєктувати інтерфейси для додатків за допомогою перетягування компонентів, таких як кнопки, текстові поля, списки, зображення та інші елементи. Розробники також можуть редагувати XML-код інтерфейсу, що дає повний контроль над дизайном додатка. Крім того, Android Studio підтримує сучасні фреймворки для створення UI, такі як Jetpack Compose, що дозволяє застосовувати декларативний підхід для створення інтерфейсів.

Android Studio також включає інструменти для тестування і налагодження. Завдяки Android Emulator, розробники можуть тестувати свої додатки на віртуальних пристроях, що дозволяє перевіряти сумісність з різними версіями Android та конфігураціями пристроїв. Крім того, IDE надає інструменти для профілювання продуктивності додатків, зокрема моніторинг використання пам’яті, процесорного часу, мережевого трафіку, а також фреймворки для автоматизованого тестування (Espresso, UI Automator та інші).

Ще однією важливою особливістю Android Studio є підтримка багатьох варіантів збірки та налаштування Gradle. Завдяки цьому розробники можуть створювати різні конфігурації для різних середовищ (наприклад, для розробки, тестування, продакшену) та автоматизувати процеси збірки, завантаження залежностей і публікації додатків. Це дозволяє легко управляти складними проектами і зменшити час на розгортання нових версій додатків.

У Android Studio також є підтримка для роботи з різними типами баз даних, такими як SQLite і Room, що спрощує інтеграцію з локальними базами даних на Android-пристрої. Крім того, інструменти Android Studio дозволяють розробникам інтегрувати свої додатки з хмарними сервісами, такими як Firebase, а також додавати можливості відправки push-сповіщень і аналітики.

Android Studio забезпечує зручну і ефективну роботу з Git-репозиторіями завдяки вбудованій підтримці системи контролю версій Git. Розробники можуть безпосередньо з IDE виконувати операції над репозиторіями, заливати зміни, відновлювати попередні версії коду, а також працювати в команді, синхронізуючи зміни з віддаленими репозиторіями.

Android Studio активно підтримує автоматичне оновлення та синхронізацію з останніми версіями Android SDK і бібліотек. Це гарантує, що розробники завжди працюють з найновішими інструментами і функціями, доступними на платформі Android.

SDK включає набір API (Application Programming Interface), які дозволяють додаткам взаємодіяти з операційною системою Android і апаратними засобами мобільних пристроїв. Ці API дозволяють виконувати безліч завдань, від базових операцій, таких як доступ до файлової системи, до складних, наприклад, робота з камерою, GPS, сенсорами пристрою та іншими функціями. SDK також містить інструменти для підтримки різноманітних версій Android, забезпечуючи сумісність додатків з різними поколіннями пристроїв.

Одним з важливих компонентів Android SDK є набір інструментів для емуляції Android-пристроїв. За допомогою Android Emulator розробники можуть тестувати свої додатки на різних типах пристроїв без необхідності мати фізичний доступ до кожного з них. Емулятор підтримує різні конфігурації екранів, обсягів пам’яті, а також дозволяє тестувати додатки в різних мережевих умовах, таких як 3G, 4G або Wi-Fi.

Ще одним важливим елементом SDK є набір інструментів для налагодження (debugging) та профілювання додатків. Android SDK надає інструменти для моніторингу продуктивності додатків, аналізу використання

пам’яті, процесорного часу і мережевого трафіку. Інструменти, такі як Android Debug Bridge (ADB), дозволяють розробникам підключати пристрій до комп’ютера і виконувати на ньому операції з налагодження, завантаження додатків, а також перевірки їхньої роботи.

SDK також включає в себе інструменти для автоматизації тестування. Для цього використовуються різні бібліотеки і фреймворки, такі як Espresso, Robolectric або UI Automator, які допомагають перевіряти функціональність додатків на різних рівнях — від одиничних тестів до комплексних тестів користувацького інтерфейсу. Це дозволяє значно скоротити час, необхідний для тестування додатків, і зменшити кількість помилок при випуску фінальних версій продукту.

Окрім базових компонентів для розробки додатків, Android SDK включає інструменти для роботи з різноманітними сервісами Google, такими як Firebase, Google Play Services, а також бібліотеки для інтеграції карт, аналізу даних і роботи з хмарними сервісами. Ці інструменти дають можливість додавати до додатків розширену функціональність, таку як відправка push-сповіщень, автентифікація через Google або аналіз даних за допомогою Firebase.

Важливою частиною Android SDK є підтримка багатьох мов програмування, серед яких найбільш популярні Java і Kotlin. Java була основною мовою для розробки Android-додатків протягом багатьох років, однак Kotlin, що був доданий до Android SDK в 2017 році, значно спростив процес програмування завдяки своєму більш зручному синтаксису та розширеним можливостям для роботи з асинхронними операціями.

Таким чином, Android SDK є невід’ємною частиною процесу створення Android-додатків. Він надає все необхідне для розробки, тестування, налагодження та оптимізації додатків, забезпечуючи розробникам доступ до інструментів і бібліотек для створення якісних мобільних продуктів.

1.6 Огляд існуючих аналогів

Додаток має лише кілька аналогів, що робить його унікальним інструментом для генерації математичних завдань із широкими можливостями налаштування та зручністю використання. Нижче розглянемо один із них, а саме «Математика: Генератор задач». Це інструмент призначений для генерування різноманітних математичних завдань за весь період вивчення математики в шкільній програмі (рис. 1.1).

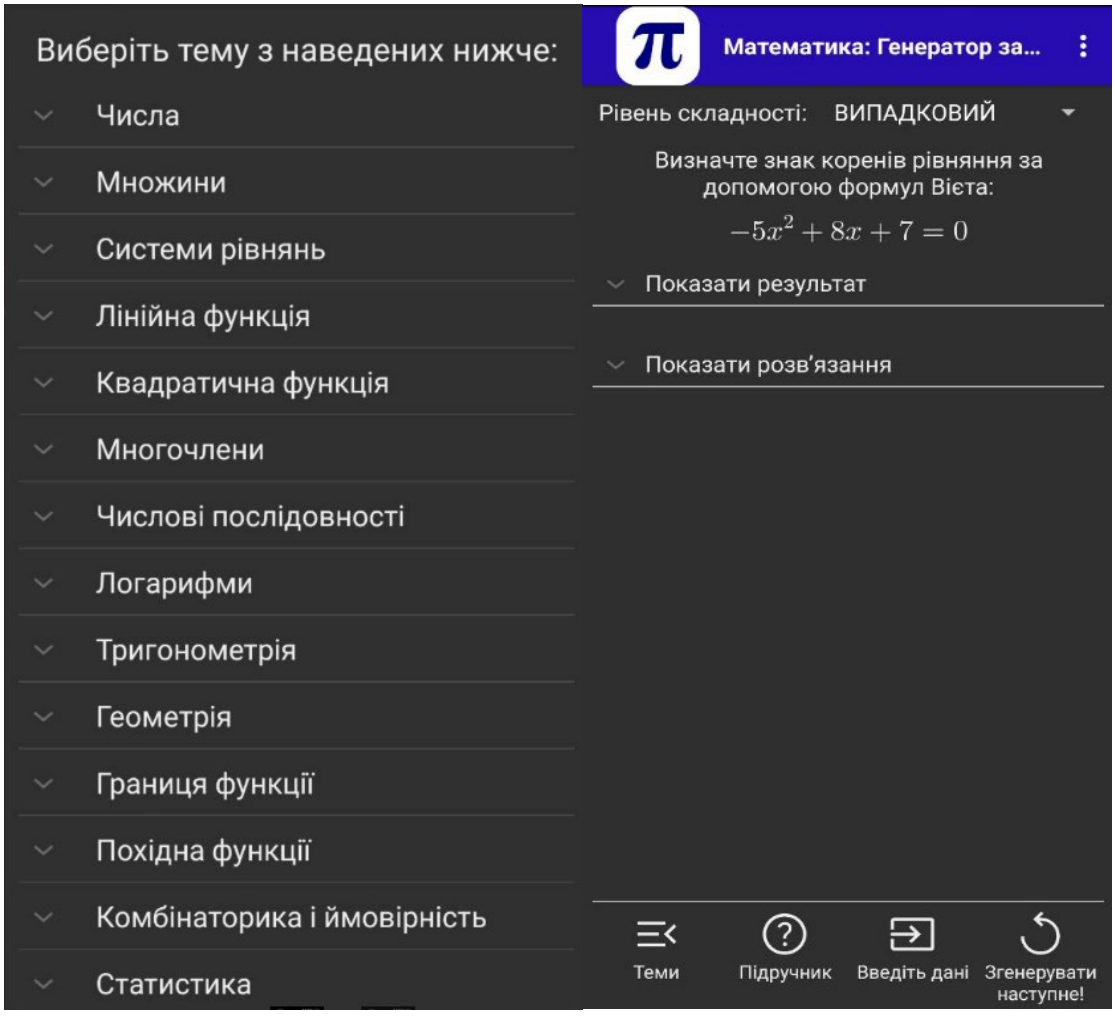

Рисунок 1.1 – Інтерфейс додатку «Математика: Генератор задач»

Переваги:

- більше тем для вивчення та практики;
- виведення розв’язків та шляхів вирішення завдання;

- теорія по темах;
- можливість перевірити себе у розв’язані.

Недоліки:

- генерація по 1 прикладу за раз;
- неможливість експорту згенерованих завдань .

РОЗДІЛ 2
ПРОЄКТУВАННЯ ТА РОЗРОБКА ANDROID-ДОДАТКУ "MATHGEN"

2.1. Постановка задачі, призначення та вимоги до програмного засобу

Серед усіх типів програм було обрано саме Android-додаток через його доступність і популярність. Android є найпоширенішою мобільною платформою у світі, що забезпечує широкий охоплення користувачів. Мобільний додаток надає можливість використовувати програму в будь-який час і в будь-якому місці, що особливо важливо для учнів і вчителів.

Вимоги:

- зручність;
- висока швидкодія;
- інтуїтивна зрозумілість;
- надійність;
- доступність.

Додаток повинен виконувати такі завдання:

- генерація рівнянь наступних видів: прості пропорції, лінійні рівняння, системи лінійних рівнянь з двома змінними, многочлени, квадратні рівняння, ірраціональні рівняння, тригонометричні рівняння та вирази, показникові рівняння, логарифмічні рівняння, складні системи рівнянь (системи з логарифмічними чи тригонометричними елементами);
- вибір користувачем виду рівняння, рівня складності, кількості завдань та кількості варіантів;
- можливість вибору відразу кількох видів рівнянь для генерації ;
- інтерактивний інтерфейс, сприятливий для використання як учителями, так і учнями;
- можливість експорту генерованих прикладів у форматі PDF.

Додаток повинен успішно працювати на операційних системах Android версії 7.0 та вище.

В додатку А подано технічне завдання на розробку Android-додатку для генерація математичних прикладів.

2.2. Вибір моделі розробки програмного засобу

Вибір моделі розробки програмного засобу є ключовим етапом, що визначає ефективність та якість процесу створення програмного забезпечення. Для розробки Android-додатку було обрано ітеративну модель розробки, яка є поєднанням елементів класичних водоспадних моделей і гнучких методологій. Ця модель забезпечує поетапний підхід до створення продукту, даючи змогу постійно вдосконалювати його, оперативно реагуючи на зміни вимог і впроваджуючи нові функції відповідно до зворотного зв’язку від користувачів та інших зацікавлених сторін.

Ітеративна модель передбачає поділ процесу розробки на кілька послідовних циклів (ітерацій), кожен з яких включає такі ключові етапи:

Планування – визначення цілей конкретної ітерації, розподіл завдань, оцінка необхідних ресурсів і часу. На цьому етапі формується розуміння того, які функції або компоненти будуть реалізовані.

Проєктування – створення архітектурних рішень, розробка моделей і підготовка технічної документації, яка описує, як саме буде реалізовано функціонал.

Розробка – написання коду та інтеграція функціональних модулів додатку. У цьому етапі активно використовуються сучасні інструменти та фреймворки, наприклад Kotlin, Jetpack Compose, View Binding, Coroutines тощо.

Тестування – перевірка реалізованого функціоналу на наявність помилок, оцінка його продуктивності та відповідності вимогам. Особливу увагу

приділяють тестуванню ключових функцій додатку, зокрема генерації математичних завдань та експорту у PDF-формат.

Аналіз – оцінка досягнутих результатів, порівняння їх із поставленими цілями, а також визначення проблемних зон і потенційних шляхів їх вирішення.

Завдяки такій структурі розробка є гнучкою, дозволяючи вносити зміни до функціоналу чи дизайну продукту на будь-якому етапі розробки. Це особливо важливо в умовах динамічного розвитку мобільного ринку, де вимоги користувачів можуть швидко змінюватися.

Однією з головних переваг ітеративної моделі є можливість створювати робочий прототип програмного забезпечення вже на початкових етапах. Це дає змогу раннього тестування функцій, що дозволяє отримувати зворотний зв’язок від кінцевих користувачів і вчасно враховувати їхні потреби. Крім того, ітеративний підхід сприяє ефективнішій роботі, оскільки завдання розподіляються рівномірно між ітераціями, уникаючи перевантаження на завершальних етапах проєкту.

Для розробки Android-додатку ця модель є оптимальною, оскільки дозволяє інтегрувати новітні технології та адаптувати продукт під сучасні вимоги. Таким чином, вибір ітеративної моделі дозволяє створювати якісний, продукт із мінімальними ризиками, забезпечуючи гнучкість, адаптивність та високий рівень задоволення потреб користувачів.

2.3. Загальний опис проєкту

UML (Unified Modeling Language) – це уніфікована мова моделювання, яка використовується для візуалізації, документування й проектування програмних систем. UML допомагає розробникам та аналітикам описати структуру і

поведінку системи, створити специфікації та спростити комунікацію між членами команди.

Діаграма варіантів використання – це один із видів UML-діаграм, що моделює функціональні можливості системи з точки зору взаємодії з нею користувачів або інших систем. Вона дозволяє зрозуміти, які задачі виконує система і як вона взаємодіє з актором або акторами (користувачами або іншими зовнішніми системами). У нашому випадку це є можливості вибору виду рівняння, рівня складності, кількості варіантів та кількості прикладів у варіанті, які обираються користувачем. Система ж генерує рівняння відповідно до налаштувань, вибраних користувачем, та експортує згенеровані приклади у PDF-файл для подальшого перегляду та інших дій з боку користувача (рис. 2.1).

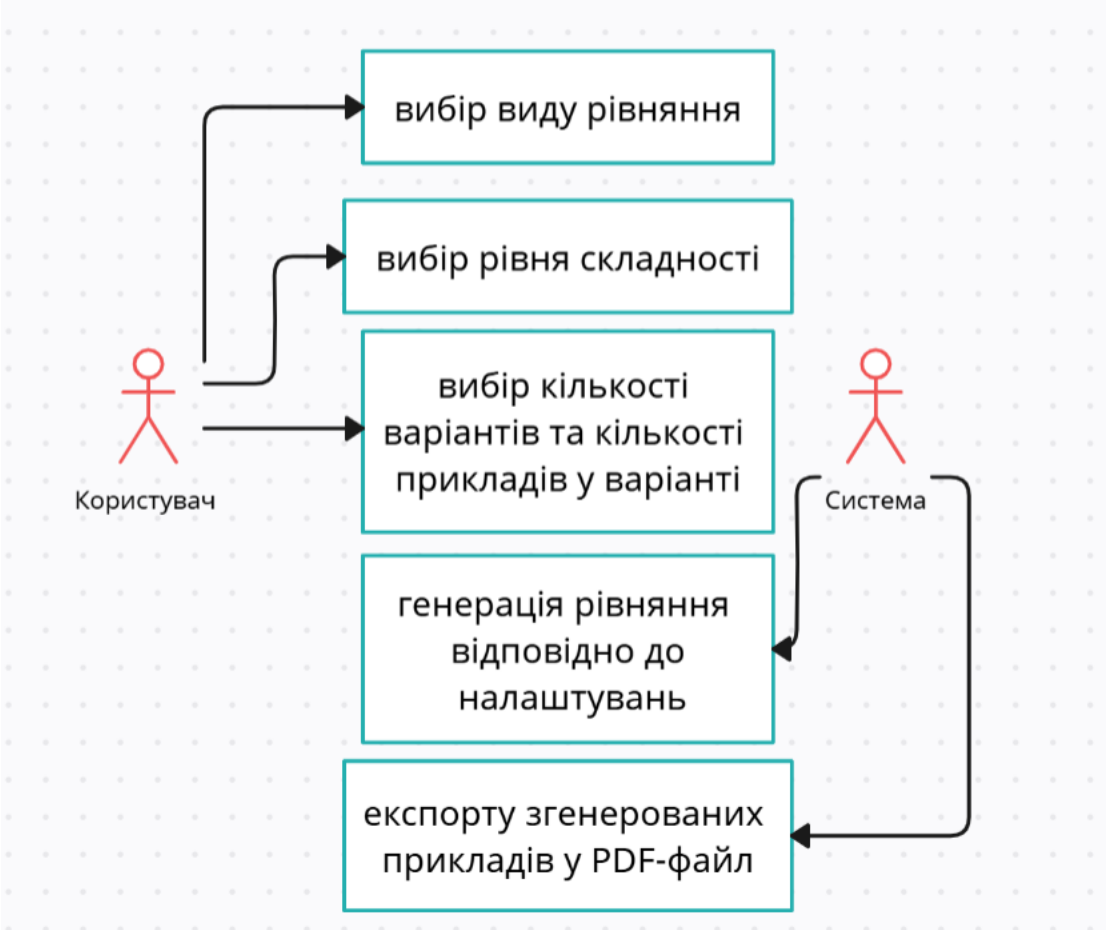

Рисунок 2.1 – Діаграма варіантів

Діаграма діяльності є одним із типів UML-діаграм, що використовується для моделювання потоку виконання в програмних системах. Вона описує процеси або дії, що виконуються в системі, відображаючи їх у вигляді послідовних кроків і взаємодій між ними. Діаграма діяльності дозволяє розглядати функціональність системи на більш високому рівні, зосереджуючи увагу на послідовності дій, умовах переходу та обробці різних варіантів виконання. У нашому випадку це послідовність дій після запуску додатку. Після старту користувач вибирає вид рівняння, рівень складності, кількість варіантів та кількість прикладів у варіанті та після цього має можливість перейти до перегляду результатів генерування або заново вибрати тему для подальшого генерування або закрити програму (рис. 2.2).

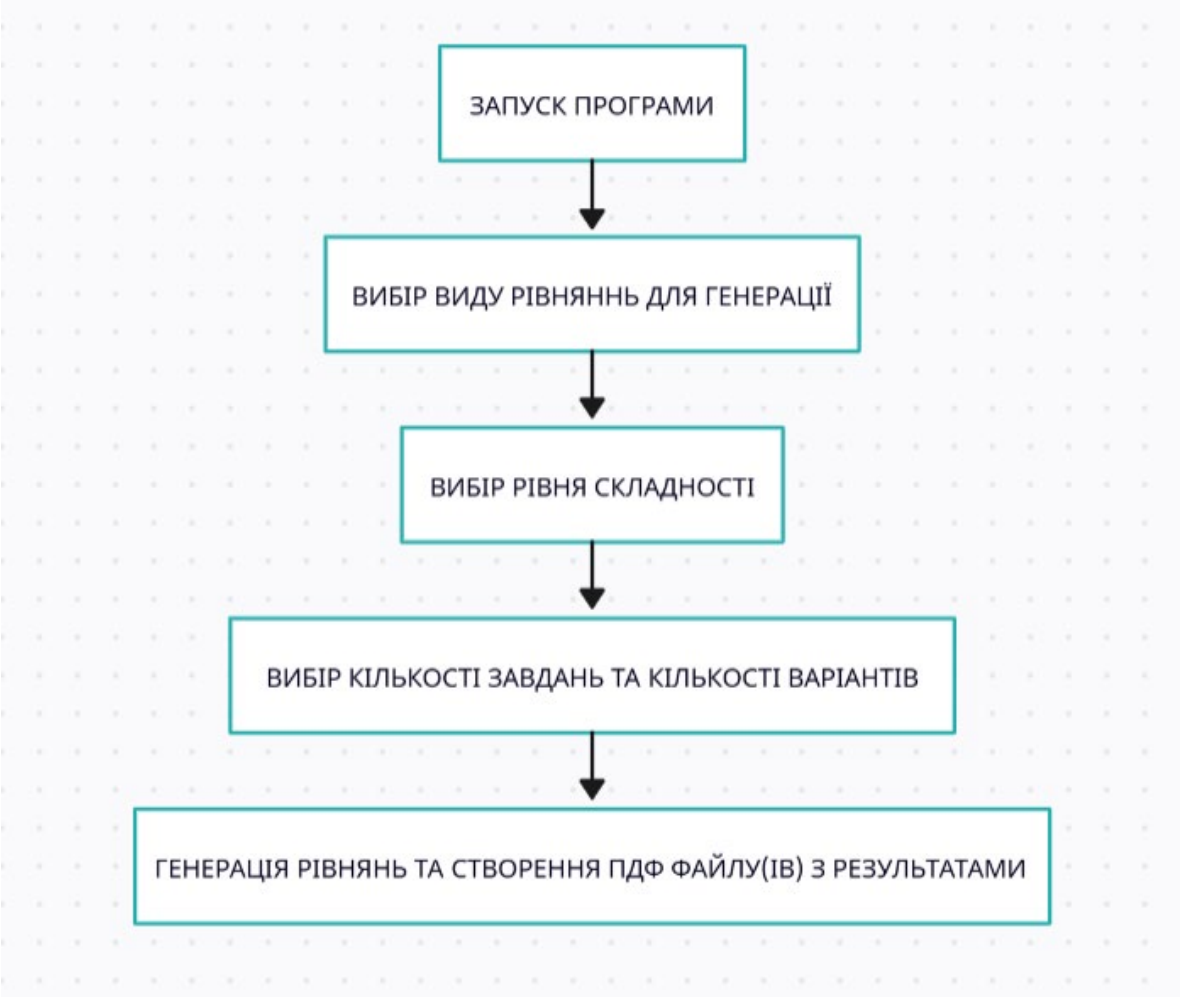

Рисунок 2.2 – Діаграма діяльності

Додаток починається зі стартового екрану завантаження, на якому відображається логотип. Після завершення завантаження користувач потрапляє на головний екран, де розміщений RecyclerView із елементами, згрупованими по два в ряду. Кожен елемент відповідає за певний тип математичних рівнянь.

На головному екрані також є елемент для вибору відразу кількох різних тем рівнянь і кнопка, яка відкриває папку в провіднику, де зберігаються файли із завданнями. При натисканні на будь-який елемент відкривається спливаюче вікно (pop-up), де можна вибрати кількість завдань, кількість варіантів і рівень складності.

Усі рівняння генеруються за одним принципом. Коли в pop-up вікні натискається кнопка "Генерувати", додаток зчитує обрані теги складності, теми, кількості завдань і варіантів. Залежно від обраної теми визначається відповідний клас для генерації завдань. Наприклад, для теми "диференційні рівняння" використовується відповідний клас. Потім, на основі обраного рівня складності, визначається набір макетів завдань, які будуть використані. Кількість завдань обмежується тегом, і для кожного завдання випадково вибирається шаблон, у який генеруються коефіцієнт. Кількість варіантів визначає, скільки PDF-файлів із завданнями буде створено. Кожен PDF-файл містить номер варіанту, тему та самі завдання.

У pop-up вікні для вибору відразу кількох тем встановлено перемикачі (switchers). Користувач може вибрати кілька тем, після чого завдання генеруються шляхом випадкового звернення до відповідних класів. Рівень складності обирається за допомогою радіо-кнопок, а кількість завдань і варіантів вводиться вручну в EditText.

Процес створення PDF-файлів у додатку побудований наступним чином:

Спочатку ми отримуємо всі введені користувачем параметри: кількість завдань, кількість варіантів, рівень складності та тему. Ці параметри зберігаються у вигляді тегів для подальшої обробки.

Залежно від обраної теми визначається відповідний клас, який відповідає за генерацію завдань для цієї теми. Кожен клас містить шаблони завдань (макети), які відповідають різним рівням складності.

На основі рівня складності завантажується відповідний список шаблонів. Для кожного завдання із заданої кількості випадковим чином вибирається шаблон, у який вставляються унікальні числа (щоб уникнути повторів). Це дозволяє забезпечити різноманітність у завданнях.

Після формування списку завдань для одного варіанту ми додаємо до контенту PDF-файлу:

- номер варіанту;
- назву теми;
- згенеровані завдання із використанням обраних шаблонів.

Якщо користувач обрав кілька варіантів, процес генерації повторюється для кожного варіанту. Кожен варіант містить унікальні завдання на основі обраних параметрів.

Для створення PDF використовується бібліотека, яка дозволяє додавати текст і зображення до документів. Ми створюємо документ, заповнюємо його контентом і зберігаємо у локальній файловій системі.

Готовий PDF-файл зберігається у папку додатку, до якої користувач має доступ через спеціальну кнопку на головному екрані. Файл автоматично отримує унікальну назву, яка включає тему, рівень складності та номер варіанту.

Для генерації рівнянь у додатку було реалізовано різноманітні алгоритми, загалом для кожного виду рівняння було створено шаблони для кожного рівня складності по 5 у кожному, з випадковою генерацією коефіцієнтів або генерацією при якій рівняння має цілий розв’язок, залежно від типу.

Для лінійних рівнянь це виглядає наступним чином.

Легкий рівень:

На легкому рівні рівняння будуть містити прості операції з однією змінною. Рівняння можуть мати форму:

- розв'яжіть рівняння: $a\,x + b = 0$;

- знайдіть x: a x = b;
- розв'яжіть рівняння: a x - b = 0;
- знайдіть корінь рівняння: x / a = b;
- розв'яжіть рівняння: a x = b x + c.

Середній рівень:

На середньому рівні рівняння стають складнішими, додаються дроби та змінні з обох сторін:

- розв'яжіть рівняння: a x + b = c x + d;
- знайдіть x: (a x + b)/c = (d x + e)/f;
- розв'яжіть рівняння: (a x - b)/(c x + d) = e;
- знайдіть корінь рівняння: a(x - b) = c(x + d);
- розв'яжіть рівняння: a(x + b) - c(x - d) = e.

Складний рівень:

На складному рівні рівняння містять дроби з кількома змінними та більш складні операції:

- розв'яжіть рівняння: (a x + b)/(c x - d) = (e x + f)/(g x + h);
- знайдіть x: a x + b = c (d x + e);
- розв'яжіть рівняння: (a x + b)/(c x + d) = e / (f x + g);
- знайдіть корінь рівняння: a(x + b) - c(x + d) + e = 0;
- розв'яжіть рівняння: a(x + b) + c(x - d) = e x + f.

Після цього для кожного шаблону лінійного рівняння ми генеруємо коефіцієнти a, b та c так, щоб одна зі змінних була кратною іншій, що дозволить отримати ціле значення для x.

Для генерації квадратних рівнянь, ми також використовуємо шаблони рівнянь, поділені за рівнем складності. Для кожного рівня складності шаблони рівнянь включають різноманітні варіанти операцій з коефіцієнтами. Важливим аспектом є те, що потрібно генерувати коефіцієнти так, щоб рівняння мали цілі числа в розв'язку. Для цього треба враховувати дискримінант та інші умови, які дозволяють отримувати цілі корені.

Легкий рівень:

На легкому рівні рівняння будуть міститись прості квадратичні операції з однією змінною:

- розв'яжіть рівняння: $a x^2 + b x + c = 0$;
- знайдіть x: $x^2 + a x + b = 0$;
- розв'яжіть рівняння: $a x^2 + b = 0$;
- знайдіть корінь рівняння: $x^2 = a x + b$;
- розв'яжіть рівняння: $a x^2 = b x + c$.

Середній рівень:

На середньому рівні рівняння стають більш складними, включаючи різні операції з коефіцієнтами:

- розв'яжіть рівняння: $a x^2 + b x + c = 0$;
- знайдіть x: $a x^2 - b x = c$;
- розв'яжіть рівняння: $(a x + b)(c x + d) = 0$;
- знайдіть корінь рівняння: $a x^2 + b x = c x + d$;
- розв'яжіть рівняння: $a x^2 + b x + c = 0$.

Складний рівень:

На складному рівні рівняння будуть більш заплутаними, тобто містити змінні з високими степенями:

- розв'яжіть рівняння: $a x^4 + b x^2 + c = 0$;
- знайдіть x: $(a x^2 + b x + c)(d x^2 + e x + f) = 0$;
- розв'яжіть рівняння: $a x^4 - b x^2 + c = 0$;
- розв'яжіть рівняння: $(x^2 + a x + b)^2 = c$.

Було створено алгоритм генерації коефіцієнтів для цілих коренів. Для кожного рівняння генеруємо випадкові значення для коефіцієнтів a, b, і c і тд. і перевіряємо, чи задовольняють ці значення умову, що дискримінант є квадратом числа. Якщо це не так, генеруємо нові коефіцієнти. Цей підхід дозволяє забезпечити генерацію рівнянь, які мають цілі корені для кожного шаблону на всіх рівнях складності.

Такі шаблони різних рівнів складності та алгоритми генерації коефіцієнтів для цілих коренів було розроблено та запрограмовано для всіх видів рівнянь.

2.4 Обґрунтування вибору інструментальних засобів розробки

Вибір інструментальних засобів розробки Android-додатків у даній роботі обумовлений необхідністю використання сучасних, ефективних та надійних технологій для створення програмного забезпечення. Кожен з обраних інструментів має свою специфіку та особливості, що забезпечує високий рівень продуктивності, зручність у розробці та підтримку різноманітних функцій.

Kotlin був вибраний як основна мова програмування для розробки Android-додатків, оскільки вона є офіційно підтримуваною мовою для Android і має безліч переваг перед Java, включаючи зручний синтаксис, розширену підтримку функціональних можливостей, таких як лямбда-вирази, розширення функцій, null-безпечність і більш сучасну модель обробки помилок. Завдяки своїй інтеграції з усіма основними Android-інструментами, Kotlin забезпечує ефективну розробку без втрат на продуктивності, що робить його ідеальним для створення мобільних додатків.

Jetpack Compose було вибрано як сучасний інструмент для розробки інтерфейсів у додатках. Цей фреймворк дозволяє створювати UI за допомогою декларативного підходу, що робить процес побудови інтерфейсів простим та зрозумілим. На відміну від традиційних XML-компонентів, Jetpack Compose дозволяє швидко й ефективно змінювати вигляд інтерфейсу без необхідності зайвих операцій з перерисовуванням елементів, що суттєво підвищує продуктивність.

Coroutines є важливим інструментом для управління асинхронними операціями в Kotlin. Завдяки цій бібліотеці можна легко створювати ефективні, читабельні та безпечні потоки виконання, що дозволяє працювати з багатозадачністю без блокування основного потоку. Це надзвичайно корисно для розробки додатків, які потребують паралельної обробки даних або взаємодії з мережею чи базами даних без втрати продуктивності.

View Binding був обраний для ефективного та безпечного взаємодії з елементами інтерфейсу користувача в Android-додатках. Завдяки цьому

інструменту можна безпосередньо звертатися до елементів UI без необхідності використання методів на кшталт findViewById(), що не тільки спрощує код, але й робить його більш надійним і зручним для підтримки, оскільки не потрібно вручну перевіряти ідентифікатори елементів.

RecyclerView вибрано для відображення динамічних списків, оскільки цей компонент є універсальним та дуже гнучким інструментом для обробки великих обсягів даних. Взаємодія з адаптерами дозволяє створювати списки, що можуть динамічно змінюватися без необхідності перерисовувати всі елементи списку при кожній зміні.

Адаптери в Android використовуються для зв’язку даних з інтерфейсами. Вони дозволяють відображати інформацію у вигляді списків чи інших комплексних елементів UI, що забезпечує гнучкість у роботі з великими обсягами даних. Адаптери беруть на себе роль організації даних для відображення в різних компонентах, таких як RecyclerView, забезпечуючи зручний механізм обробки змін і оновлень.

Material Design 3 був обраний для розробки інтерфейсів, оскільки цей стандарт надає інтуїтивно зрозумілий, привабливий і зручний дизайн для користувачів Android. Завдяки широкому набору компонентів та адаптивним стилям цей стандарт дозволяє створювати естетичні та зручні інтерфейси, що забезпечує гарний користувацький досвід на різних пристроях.

GitHub Libraries були вибрані як основний інструмент для інтеграції сторонніх рішень у проекти. Використання бібліотек з GitHub дозволяє швидко знаходити рішення для стандартних задач та інтегрувати готові компоненти, що значно зменшує час на розробку та покращує ефективність створення програм.

Gradle – потужний інструмент автоматизації збірки та управління залежностями, є основою для створення і підтримки Android-додатків. Завдяки простоті використання і гнучкості у конфігураціях, Gradle дозволяє налаштувати процеси збірки для різних варіантів розгортання додатка і автоматизувати більшість задач, пов’язаних з підготовкою проєкту до публікації.

Android SDK є головним інструментом для створення, тестування та налагодження мобільних додатків для Android. SDK забезпечує всі необхідні інструменти для розробки, тестування та впровадження додатків, включаючи емулятори, дебаггери та інші корисні компоненти, що дозволяють розробникам створювати якісні мобільні додатки з мінімальними витратами часу.

2.5 Особливості програмної реалізації та основні режими функціонування

Перш за все створимо спливаюче вікно для відображення всіх видів рівнянь. Перевіримо чи вже відображається спливаюче вікно, і якщо так, то буде виконуватись вихід з функції. Якщо вікно ще не відображено, змінна встановлюється в значення true, щоб уникнути повторного відображення вікна. Далі створимо вигляд спливаючого вікна на основі шаблону. Визначимо розміри вікна. Потім створимо об'єкт, в якому задаємо створений вигляд і визначаємо параметри: вікно можна натискати, але воно не закривається при натисканні поза вікном, а також встановимо прозорий фон. (рис. 2.3).

```
private fun showSpellPopup(spell: Spell) {
    try {
        if (isPopupDisplayed) {
            return
        }
        isPopupDisplayed = true

        val inflater = getSystemService(Context.LAYOUT_INFLATER_SERVICE) as LayoutInflater
        val popupView = inflater.inflate(R.layout.pop_selected_spell_craft_item, root: null)

        val width = (resources.displayMetrics.widthPixels * 0.9).toInt()
        val height = ViewGroup.LayoutParams.WRAP_CONTENT

        val popupWindow = PopupWindow(popupView, width, height)

        popupWindow.isFocusable = true
        popupWindow.isOutsideTouchable = false
        popupWindow.isTouchable = true
        popupWindow.setBackgroundDrawable(ColorDrawable(Color.TRANSPARENT))
```

Рисунок 2.3 – Спливаюче вікно для головного меню

Далі встановимо напівпрозорість для всього екрану. Тепер за допомогою findViewById будемо знаходити необхідні елементи інтерфейсу у вигляді спливаючого вікна: текстове поле для назви типу, кнопка закриття, група радіокнопок для вибору складності (легкий, середній, складний), а також поля для введення кількості завдань і варіантів. Радіокнопка для легкого рівня активується за замовчуванням. Задамо текст для назви типу в spellNameTextView.

Далі додамо обробник події закриття спливаючого вікна через setOnDismissListener, в якому відновлюється прозорість екрану. Після цього оновлюємо атрибути вікна, а змінна isPopupDisplayed встановлюється в false, щоб вказати, що вікно закрите. Далі спливаюче вікно відображається по центру екрану за допомогою showAtLocation, і застосовується анімація для масштабування вікна до початкового розміру.

Тепер напишемо код для обробки натискання на кнопку генерації castSpell. Спочатку перевіряється, чи вибрано рівень складності за допомогою перевірки selectedDifficultyId. Якщо рівень не вибраний, відображається повідомлення з Toast, що вказує на необхідність вибору складності. Потім перевіримо, чи введено кількість завдань і варіантів. Якщо ці поля порожні, з'являється Toast з повідомленням про необхідність заповнити ці поля. Якщо всі перевірки пройдені, за допомогою оператора when визначається, який рівень складності вибрано (легкий, середній або складний). (рис. 2.4).

Після цього створимо відповідність вибору генератора задач для певного типу рівнянь. Спочатку визначимо кількість задач та варіантів, конвертуючи значення з типу String до Int. Далі, залежно від назви типу рівняння (значення spell.name), створюється відповідний об'єкт генератора задач, наприклад, для "Прості пропорції" обирається генератор SimpleProportionsSpell, для "Лінійні рівняння" — LinearEquationsSpell і так далі для інших типів рівнянь. Якщо тип не знайдено, за умовчанням вибирається генератор для простих пропорцій.

```
castSpell.setOnClickListener {
    val selectedDifficultyId = difficultyRadioGroup.checkedRadioButtonId

    val numberOfTasks = numberOfTasksEditText.text.toString()
    val numberOfVariants = numberOfVariantsEditText.text.toString()

    if (selectedDifficultyId == -1) {
        Toast.makeText( context: this,  text: "Виберіть рівень складності", Toast.LENGTH_SHORT).show()
    } else if (numberOfTasks.isEmpty() || numberOfVariants.isEmpty()) {
        Toast.makeText( context: this,  text: "Вкажіть кількість завдань та варіантів", Toast.LENGTH_SHORT
    } else {
        val difficulty = when (selectedDifficultyId) {
            R.id.easyRadioButton -> "легкий"
            R.id.mediumRadioButton -> "середній"
            R.id.hardRadioButton -> "складний"
            else -> "легкий"
        }
```

Рисунок 2.4 – Обробки натискання на кнопку генерації

Далі напишемо код, який буде відповідати за створення PDF-файлів за допомогою методу generateTasks і відображення спливаючого вікна з результатами. Спочатку генеруватимуться файли, і якщо список шляхів не порожній, то спливаюче вікно закривається. Далі буде створюватись діалогове вікно з повідомленням про успішне створення файлів, і в ньому відображатиметься список імен файлів. Користувач зможе вибрати один з файлів, після чого запуститься відкриття обраного PDF-файлу за допомогою відповідного додатку. Для цього буде створюватись URI для файлу через FileProvider, і тоді файл відкриватиметься. (рис. 2.5)

Тепер створимо діалогове вікно, яке повідомляє користувача про успішне створення PDF-файлів. Якщо файли були успішно згенеровані, в діалоговому вікні відображатиметься список імен файлів, отриманих зі шляху до файлів. Користувач зможе вибрати файл, і тоді за допомогою FileProvider створиться URI для файлу, після чого відкриється PDF-файл в зовнішньому додатку для перегляду. У діалоговому вікні є кнопка "OK", яка буде закривати вікно. Якщо виникла помилка при створенні PDF, відображатиметься повідомлення про помилку за допомогою Toast.

```
val pdfPaths = spellGenerator.generateTasks( context: this, difficulty, tasksCount, variantsCount)

if (pdfPaths.isNotEmpty()) {
    popupWindow.dismiss()
    isPopupDisplayed = false

    val builder = AlertDialog.Builder( context: this)
    builder.setTitle("PDF файли успішно створені!")

    val fileNames = pdfPaths.map { File(it).name }
    builder.setItems(fileNames.toTypedArray()) { dialog, which ->
        val file = File(pdfPaths[which])
        val uri = FileProvider.getUriForFile( context: this,  authority: "$packageName.provider", file)
        val openIntent = Intent(Intent.ACTION_VIEW)
        openIntent.setDataAndType(uri,  type: "application/pdf")
        openIntent.addFlags(Intent.FLAG_GRANT_READ_URI_PERMISSION)
        startActivity(openIntent)
```

Рисунок 2.5 – Створення PDF-файлів

Далі створимо обробку події натискання на вікно виду рівняння. Якщо ім'я типу дорівнює "Створити власне завдання", то відкривається нова активність OwnSpellActivity. Якщо ім'я типу — "Папка з файлами", викликається метод для відкриття директорії з PDF-файлами. В інших випадках, буде викликатись метод для відображення спливаючого вікна з інформацією щодо вибраного виду рівняння. (рис. 2.6).

```
override fun onSpellClicked(spell: Spell, anchor: View) {
    if (spell.name == "Створити власне завдання") {
        val intent = Intent( packageContext: this, OwnSpellActivity::class.java)
        startActivity(intent)
    } else if (spell.name == "Папка з файлами") {
        openPdfDirectory()
    } else {
        showSpellPopup(spell)
    }
}
```

Рисунок 2.6 – Обробка натискання на вікно з видом рівняння

Тепер визначимо інтерфейс OnSpellClickListener, який містить метод onSpellClicked, що викликається при натисканні на вид рівняння. Клас SpellsAdapter є адаптером для RecyclerView, який відображає список видів у вигляді елементів інтерфейсу. У конструкторі адаптера буде передаватись список видів, об'єкт KonfettiView для анімації та відслідковувач натискань. Метод onCreateViewHolder буде створювати та повертати SpellViewHolder, який відповідає за відображення елементу списку. У методі onBindViewHolder будуть зв’язуватись дані рівняння з відповідним елементом у ViewHolder.

Далі визначимо внутрішній клас SpellViewHolder, який є частиною адаптера для RecyclerView. У методі bind() відбуватиметься прив'язка даних до елементів інтерфейсу: встановлюватиметься текст і зображення для рівняння на основі переданого об'єкта spell. Також, при натисканні на елемент списку, застосовуватиметься анімація зменшення розміру елемента за допомогою ScaleAnimation, де зменшується масштаб елемента, з фіксованим центром анімації (pivotX і pivotY). Задамо тривалість анімації і після її виконання елемент залишатиметься зменшеним завдяки параметру fillAfter. (рис. 2.7).

```
inner class SpellViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {
    fun bind(spell: Spell) {
        val spellName = itemView.findViewById<TextView>(R.id.spellName)
        val spellImage = itemView.findViewById<ImageView>(R.id.spellImage)

        spellName.text = spell.name
        spellImage.setImageResource(spell.imageResId)

        itemView.setOnClickListener {
            val scaleDown = ScaleAnimation(
                fromX: 1f, toX: 0.9f, fromY: 1f, toY: 0.9f,
                Animation.RELATIVE_TO_SELF, pivotXValue: 0.5f,
                Animation.RELATIVE_TO_SELF, pivotYValue: 0.5f
            ).apply {
                duration = 100
                fillAfter = true
            }
```

Рисунок 2.7 – Прив'язка даних до елементів інтерфейсу

Тепер створимо анімацію масштабування елемента з початковим масштабом 0.9x і кінцевим 1x як по горизонталі, так і по вертикалі. Використаємо ScaleAnimation, де параметри fromX і toX будуть визначати горизонтальне масштабування, а fromY і toY — вертикальне. Анімація прив'язуватиметься до центру елемента через pivotXValue і pivotYValue, щоб масштабування відбувалося навколо середини. Задамо тривалість анімації, і після її завершення елемент буде залишатись в кінцевому стані (fillAfter = true). Анімація запускатиметься на itemView. Крім того, вкажемо пустий відслідковувач подій анімації для обробки початку, завершення та повтору анімації.

Створимо функцію generateCustomSpell, яка спочатку перевірятиме введені користувачем значення для кількості завдань і варіантів, конвертуючи їх у цілі числа. Якщо будь-яке з цих значень є некоректним або меншим, або рівним нулю, буде показано повідомлення про помилку через Toast. Далі, для кожного з перемикачів, що відповідають різним типам рівнянь, буде створений список пар, що включатимуть відповідні перемикачі та об'єкти, які генеруватимуть завдання для кожного типу рівнянь. Після цього буде встановимо напівпрозорість для всього екрану. За допомогою findViewById програма знайде елементи інтерфейсу для спливаючого вікна, такі як текстове поле для назви типу, кнопка закриття, група радіокнопок для вибору рівня складності, а також поля для введення кількості завдань і варіантів. (рис. 2.8).

Далі напишемо код, де спочатку будуть фільтруватись вибрані типи, використовуючи метод filter для відбору елементів, у яких позначено прапорець, після чого ці типи додаватимуться до списку через map. Якщо список вибраних типів порожній, виводитиметься повідомлення Toast, що користувач повинен вибрати хоча б один тип. Потім буде робитись вибір складності через checkedRadioButtonId для групи радіокнопок, і в залежності від вибору користувача встановлюватиметься відповідна складність: легка, середня чи складна. Якщо жоден варіант не вибрано, встановлюється значення за замовчуванням ("легкий"). Далі буде створюватись порожній список pdfPaths, до якого буде додаватися шлях до файлів PDF.

```
private fun generateCustomSpell() {
    val tasksCount = putTasks.text.toString().toIntOrNull()
    val variantsCount = putVariants.text.toString().toIntOrNull()
    if (tasksCount == null || variantsCount == null || tasksCount <= 0 || variants
        Toast.makeText( context: this,  text: "Вкажіть коректну кількість завдань та ва
        return
    }

    val themeSwitches = listOf(
        Pair(switch1, SimpleProportionsSpell()),
        Pair(switch11, LinearEquationsSpell()),
        Pair(switch111, PolynomialsSpell()),
        Pair(switch1111, QuadraticEquationsSpell()),
        Pair(switch11111, IrrationalEquationsSpell()),
        Pair(switch12, TrigonometricEquationsSpell()),
        Pair(switch122, ExponentialEquationsSpell()),
        Pair(switch1222, LogarithmicEquationsSpell()),
        Pair(switch13, ComplexSystemsSpell()),
        Pair(switch133, DifferentialEquationsSpell())
```

Рисунок 2.8 – Функція generateCustomSpell

Тепер зробимо перевірку, чи доступне зовнішнє сховище для запису. Якщо сховище недоступне, з'являтиметься повідомлення про помилку. Далі буде створено директорію "MathGen" у публічному зовнішньому сховищі, якщо така ще не існує. Потім для кожного варіанту завдання буде створюватись новий PDF-документ. Визначимо розмір сторінки, а також використаємо об'єкт Paint для налаштування кольору тексту на чорний. Після цього задамо інформацію для нової сторінки PDF-документа з вказаними розмірами сторінки та номером. Далі створиться сама нова сторінка за допомогою startPage, і буде отримуватись об'єкт canvas, на якому буде малюватися контент. Потім викличемо функцію drawHeader, щоб створити заголовок сторінки. Після цього цикл буде проходити через всі завдання, і для кожного завдання перевірятиметься, чи вистачить місця на поточній сторінці. Якщо місце закінчується, поточну сторінку завершуємо, збільшуємо номер сторінки та створюємо нову сторінку, повторюючи процес заголовка і виведення завдань. Позиція для завдань на сторінці

оновлюватиметься кожного разу після перевірки. При завершені запису до PDF-документа буде створений файл із назвою, що містить номер варіанта. Далі додаток запише PDF у вказаний файл, використовуючи FileOutputStream, і збереже шлях до файлу у списку pdfPaths. Якщо виникне помилка під час запису, буде зафіксовано виняток, виведеться повідомлення про помилку через Toast, і виконання функції завершиться. У фінальному блоці закриємо документ для звільнення ресурсів. (рис. 2.9).

```
        pdfDocument.finishPage(page)

        val fileName = "Власне закляття. Варіант $variant.pdf"
        val file = File(docsDir, fileName)

        try {
            pdfDocument.writeTo(FileOutputStream(file))
            pdfPaths.add(file.absolutePath)
        } catch (e: Exception) {
            e.printStackTrace()
            Toast.makeText( context: this,  text: "Помилка запису PDF: ${e.message}", Toast.LENGTH_SHORT)
            return
        } finally {
            pdfDocument.close()
        }
    }
```

Рисунок 2.9 – Запис та збереження PDF-файлу

Далі створимо вікно сповіщення користувача щодо успішного генерації PDF-файлів. У разі наявності згенерованих файлів, відобразиться діалогове вікно із заголовком "PDF файли успішно створені!" та списком імен файлів. Користувач зможе вибрати файл для перегляду, після чого додаток відкриє його за допомогою зовнішнього PDF-читача, використовуючи Intent із відповідним MIME-типом і дозволом на читання URI.

Далі реалізуємо створення PDF-документів для кожного варіанту завдань. У межах циклу для кожного варіанту ініціалізуємо об'єкт PdfDocument, а також об'єкт Paint для налаштування кольору та стилю тексту. Визначимо параметри сторінки за допомогою PdfDocument. PageInfo, включаючи розміри сторінки та

номер. На сторінку через об'єкт canvas буде виводитися текст із заданими параметрами, де текст заголовка буде розміщено по центру сторінки. Завдяки обчисленню горизонтального положення тексту (topicX), заголовок "Складні системи рівнянь" буде вирівняно по центру зверху сторінки.

Після цього напишемо код для генерації тексту завдань та їх відображення на PDF-сторінці. Встановимо параметри шрифту, такі як розмір і жирність. Текст з номером варіанта, наприклад, "Варіант № X", буде центруватися по ширині сторінки та розміщуватися у верхній частині. Потім для кожного завдання генеруватимуться унікальні задачі за допомогою функції generateComplexSystemTask. Завдання перевірятимуться на унікальність, і якщо такого завдання ще немає у списку, воно додаватиметься та відображатиметься на сторінці з відповідним номером і координатами для вертикального розташування.

Тепер напишемо код для створення PDF-файл із назвою, що містить інформацію про складні системи рівнянь і номер варіанта. Після завершення роботи над сторінкою PDF-документ буде збережено у вказаному каталозі. У разі виникнення помилки запису буде виведено повідомлення про помилку, а список шляхів до PDF-файлів залишиться порожнім. Наприкінці документ буде закриватись і повертати список успішно згенерованих шляхів до PDF-файлів.

Тепер напишемо окремо всі класи для генерації рівнянь які будуть відповідати кожному окремому виду рівняння. В кожному класі будуть міститися шаблони для кожного рівня складності та алгоритми генерації коефіцієнтів для цілих коренів.

Після цього перейдемо до створення дизайну теми для Android-додатку з використанням Jetpack Compose. Він створює кольорові схеми для темної та світлої теми, а також підтримує динамічні кольори на Android 12+ (API 31 і вище). Якщо динамічні кольори увімкнені, система автоматично застосовує відповідну кольорову схему на основі налаштувань пристрою. В іншому випадку використовується заздалегідь визначена кольорова схема. Функція MathTheme

забезпечує правильне застосування теми, передаючи кольорову схему та типографію в MaterialTheme для всього вмісту додатку.

За допомогою XML-коду створимо макет для нашого Android-додатку з використанням ConstraintLayout як основного контейнера. В контейнері знаходяться два елементи: RecyclerView з ідентифікатором listOfSpells, який займає весь екран і відображає список всіх видів рівнянь (або інших елементів), та KonfettiView, що також займає весь екран і відображає ефекти конфетті у випадку успішної генерації. Обидва елементи прив'язані до всіх сторін контейнера за допомогою обмежень (constraints). (рис. 2.10).

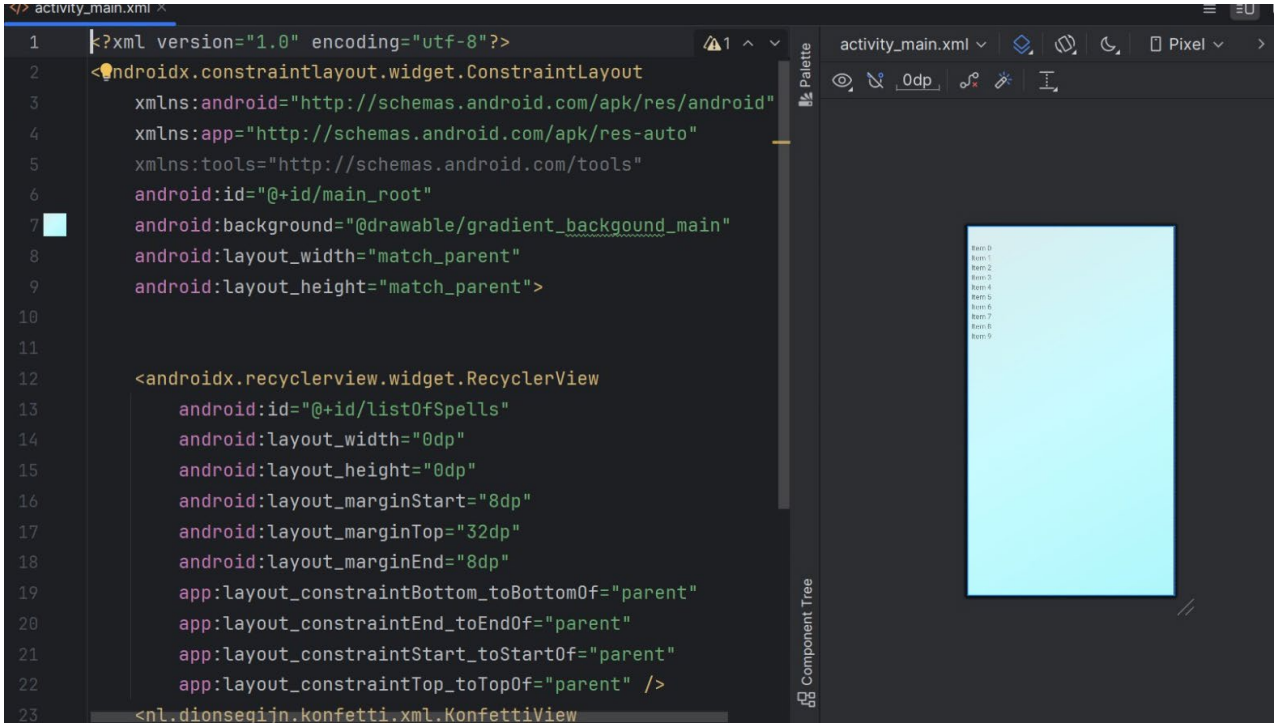

Рисунок 2.9 – Дизайн головного меню

Далі створимо дизайн для всіх інших сторінок та вікон нашого додатку, вікон для виду рівняння, спливаючих вікон для вибору опції та після генерації рівнянь так окремого вікна для створення власного набору з доступних видів рівнянь.

Режим функціонування – це сукупність станів в яких може перебувати додаток та умови функціонування його складових в залежності від цих станів. У нашого додатку є лише один основний режим функціонування На рис. 2.10 подано вигляд режиму та остаточний вигляд додатку.

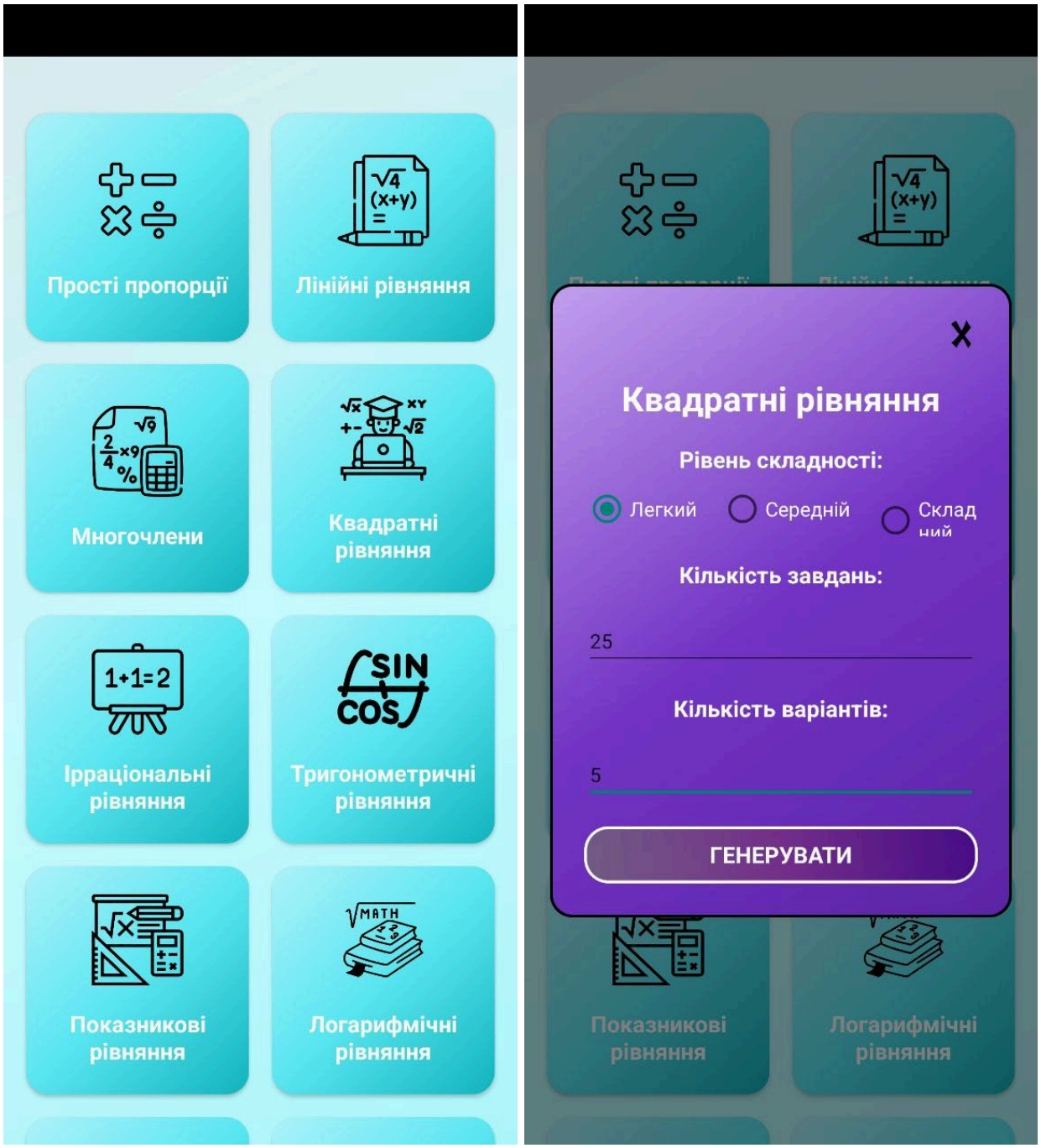

Рисунок 2.10 – Основний режим функціонування

2.6. Організація тестування та налагодження програмного засобу

Головною метою тестування розробленого Android-додатку є виявлення помилок та дефектів у його роботі, а також оцінка якості продукту у відповідному контексті використання. Тестування є невід'ємною частиною процесу розробки і включає перевірку функціональності, сумісності, продуктивності та інтерфейсу програми.

У процесі тестування були проведені наступні види тестів:

- тестування функціональності: перевірка роботи основних функцій додатку, таких як генерація рівнянь, вибір рівня складності, коректність обчислень та інтерфейсні взаємодії;
- тестування інтерфейсу: перевірка правильності відображення елементів на різних розмірах екрану, відповідність дизайну вимогам Material Design 3, зручність використання інтерфейсу;
- тестування сумісності: перевірка працездатності додатку на різних пристроях, операційних системах (Android 7.0 та вище);
- тестування продуктивності: оцінка швидкості роботи додатку, часу завантаження, швидкості генерації рівнянь та обробки великих обсягів даних.

Протягом ручного тестування не було виявлено критичних помилок, але були зафіксовані кілька дрібних недоліків, які були оперативно виправлені під час розробки. Додаток продемонстрував стабільну роботу на пристроях із різними технічними характеристиками, включаючи смартфони з процесорами середнього класу, а також на емуляторах різних версій Android.

Зокрема, тестування сумісності показало, що додаток без помилок працює на пристроях із різними версіями операційної системи Android (від 7.0 до останніх релізів). Також перевірена адаптивність інтерфейсу, яка забезпечує правильне відображення елементів на різних типах екранів, від смартфонів до планшетів.

Програму тестували на кількох пристроях з різними характеристиками: смартфони з процесорами серії Qualcomm Snapdragon та MediaTek, з

оперативною пам'яттю від 3 до 16 ГБ. В результаті тестування продуктивності було встановлено, що додаток працює без збоїв, генерація рівнянь відбувається швидко, а додаток стабільно реагує на введення користувача.

Підсумовуючи, можна сказати, що проведене тестування дозволило виявити та усунути незначні проблеми, що суттєво підвищило якість додатку. В результаті, додаток працює стабільно та коректно на всіх підтримуваних пристроях, забезпечуючи безперебійну генерацію математичних рівнянь без збоїв.

2.7. Рекомендації по використанню та впровадженню програмного засобу

Оскільки наша розробка генерує математичні рівняння, її можна успішно використовувати в освітньому процесі для покращення розуміння та практики з математичних дисциплін. Додаток має значний потенціал для впровадження в школах як інструмент для вчителів і учнів. Він дозволяє учням генерувати завдання різної складності, що дає змогу адаптувати навчальний процес під індивідуальні потреби користувачів. Для педагогів додаток стане корисним інструментом для створення навчальних матеріалів і тестів, що дозволяє зекономити час та отримати різноманітні варіанти задач для перевірки знань.

Впровадження додатку може бути доцільним у таких навчальних закладах, де проводяться курси з алгебри, геометрії, початкам математичного аналізу тощо. Завдяки гнучкості налаштувань складності, учні можуть працювати на тому рівні, який відповідає їхнім знанням і навичкам. Також додаток буде корисним для підготовки до іспитів та тестів, оскільки дає змогу згенерувати практично нескінченну кількість варіантів завдань.

Для подальшого розвитку додатку можна впроваджувати додаткові функції, такі як інтеграція з онлайн-курсами, можливість експорту результатів до систем управління навчанням (LMS), або створення адаптивних тестів, які динамічно змінюють рівень складності відповідно до результатів користувача.

З огляду на перспективи розвитку технологій та програмного забезпечення, додаток може бути вдосконалений шляхом додавання нових типів рівнянь, а також створення інтерактивних функцій, що дозволяють учням отримувати покрокові розв’язки або пояснення до кожного етапу вирішення задачі.

ВИСНОВКИ

У роботі розглянуто сучасні технології та інструменти для розробки Android-додатків. Проаналізовано переваги мови Kotlin та її інтеграцію з фреймворками, такими як Jetpack Compose для створення UI, і бібліотекою Coroutines для управління асинхронними операціями. Описано можливості View Binding для роботи з елементами інтерфейсу та RecyclerView для динамічного відображення списків. Досліджено роль адаптерів у зв'язку даних з інтерфейсом, а також значення Material Design 3 як стандарту для створення сучасних UI. Розглянуто використання GitHub Libraries для розширення функціоналу проєктів, Gradle для автоматизації збірки та Android SDK як основного інструменту розробки додатків. Огляд існуючих аналогів дозволив виділити їх переваги та недоліки. Поставлено задачі, сформульовано призначення та вимоги до програмного засобу. Обґрунтовано вибір інструментальних засобів розробки. Описано реалізацію алгоритмів генерації математичних рівнянь. Надано загальний опис проєкту за допомогою діаграм та схем, докладно описано особливості програмної реалізації додатку для генерації математичних прикладів, зазначено основні режими функціонування створеного додатку. Розроблено власний Android-додаток для генерації математичних рівнянь наступних видів: прості пропорції, лінійні рівняння, системи лінійних рівнянь з двома змінними, многочлени, квадратні рівняння, ірраціональні рівняння, тригонометричні рівняння та вирази, показникові рівняння, логарифмічні рівняння. Реалізовано вибір користувачем рівня складності, кількості завдань та кількості варіантів та можливість експорту генерованих прикладів у форматі PDF. Розроблено рекомендації щодо використання додатку в освітньому процесі та можливості його подальшого розвитку.

У результаті виконаної роботи було опановано технологію створення Android-додатків та розроблено Android-додаток для генерації математичних прикладів на мові програмування Kotlin.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Що таке Android: все що ви повинні знати. *FutureNow*. URL: https://futurenow.com.ua/shho-take-android-vse-shho-vy-povynni-znaty/ (дата звернення: 04.12.2024).
2. Розробити додаток на андроїд: інструкція зі створення. *FoxmindEd*. URL: https://foxminded.ua/rozrobyty-dodatok-na-android/ (дата звернення: 04.12.2024).
3. Розробка мобільних додатків на KOTLIN: швидко, професійно, під ключ | студія Brander. *Створення і розробка інтернет-магазинів від Brander*. URL: https://brander.ua/technologies/kotlin (дата звернення: 04.12.2024).
4. Kotlin або Java: порівняння продуктивності та застосування. *FoxmindEd*. URL: https://foxminded.ua/kotlin-abo-java/ (дата звернення: 04.12.2024).
5. Все про kotlin: ваш перший крок у розробці на android в 2024. *Game UA*. URL: https://www.procoding.com.ua/kotlin/ (дата звернення: 04.12.2024).
6. JetBrains: Essential tools for software developers and teams. *JetBrains*. URL: https://www.jetbrains.com/ (дата звернення: 04.12.2024).
7. Що таке веб додаток? | Блог WEBCASE. *Webcase*. URL: https://webcase.com.ua/uk/blog/cho-takoe-web-prilozhenie-vse-vidy/ (дата звернення: 04.12.2024).
8. ПО Java. *fw_error_www*. URL: https://www.oracle.com/cis/java/ (дата звернення: 04.12.2024).
9. Kotlin | Null и nullable-типы. *METANIT.COM - Сайт о программировании*. URL: https://metanit.com/kotlin/tutorial/5.1.php (дата звернення: 04.12.2024).
10. Найкращі кросплатформні фреймворки для розробки мобільних додатків у 2025 році | Stfalcon.com. *Custom Software Development Company | Stfalcon.com*. URL: https://stfalcon.com/uk/blog/post/cross-platform-app-development-frameworks (дата звернення: 04.12.2024).

11. Кросплатформова розробка мобільних додатків. *Webbook Studio*. URL: https://webbookstudio.com/ua/mobile-development/cross-platform-app-development/ (дата звернення: 04.12.2024).
12. Переваги та Недоліки Кросплатформної та Нативної Розробки Мобільних Додатків - Merehead. *Merehead*. URL: https://merehead.com/ua/blog/cross-platform-native-mobile-development/ (дата звернення: 04.12.2024).
13. Flutter чи Kotlin: що вибрати для розробки мобільного додатка? | URL: https://wezom.com.ua/ua/blog/flutter-kotlin-native-ili-pwa (дата звернення: 04.12.2024).
14. Coroutine в Kotlin керівництво від FoxmindEd. *FoxmindEd*. URL: https://foxminded.ua/coroutine-kotlin/ (дата звернення: 04.12.2024).
15. SDK для мобільних додатків. *eSputnik*. URL: https://esputnik.com/support/sdk-dlya-mobilnyh-prilozhenij (дата звернення: 04.12.2024).
16. SDK для Android і iOS? - ASO World. *Increase your App Ranking in Google Play & iOS & - ASO World*. URL: https://asoworld.com/ua/aso-glossary/sdk-integration/ (дата звернення: 04.12.2024).
17. NEWS P. Елегантний спосіб інтеграції API та SDK - ProIT. *ProIT: медіа для профі в IT*. URL: https://proit.ua/ielieghantnii-sposib-intieghratsiyi-api-ta-sdk/ (дата звернення: 04.12.2024).
18. XML для новачків - Підтримка від Microsoft. *Microsoft Support*. URL: https://support.microsoft.com/uk-ua/office/xml-для-новачків-a87d234d-4c2e-4409-9cbc-45e4eb857d44 (дата звернення: 04.12.2024).
19. Що таке XML? – Опис (XML) – AWS. *Amazon Web Services, Inc.* URL: https://aws.amazon.com/ua/what-is/xml/ (дата звернення: 04.12.2024).
20. Словник основних UI/UX елементів. *Онлайн-курси від компанії QATestLab | Головна сторінка*. URL: https://training.qatestlab.com/blog/technical-articles/dictionary-of-basic-ui-ux-elements/ (дата звернення: 04.12.2024).

21. Створення компонентної дизайн системи UI. *Medium*. URL: https://medium.com/nuances-of-programming/создание-компонентной-дизайн-системы-ui-29f357919eac (дата звернення: 04.12.2024).
22. Composing suspending functions | Kotlin. *Kotlin Help*. URL: https://kotlinlang.org/docs/composing-suspending-functions.html (дата звернення: 04.12.2024).
23. What does the suspend function mean in a Kotlin Coroutine?. *Stack Overflow*. URL: https://stackoverflow.com/questions/47871868/what-does-the-suspend-function-mean-in-a-kotlin-coroutine (дата звернення: 04.12.2024).
24. Життєвий цикл програми на Kotlin. | Android Developers. *Android Developers*. URL: ua (дата звернення: 04.12.2024).
25. How to get lifecycleScope from Context or Activity?. *Stack Overflow*. URL: https://stackoverflow.com/questions/74012465/how-to-get-lifecyclescope-from-context-or-activity (дата звернення: 04.12.2024).
26. Zhdanov A. Ljdujjxsredfyysq View Binding в Android. *Хабр*. URL: https://habr.com/ua/articles/467295/ (дата звернення: 04.12.2024).
27. Привязка | Android Developers. *Android Developers*. URL: ua (дата звернення: 04.12.2024).
28. Material Design 3 для Android | Stfalcon. *Custom Software Development Company | Stfalcon.com*. URL: https://stfalcon.com/uk/blog/post/android-material-design-3 (дата звернення: 04.12.2024).
29. Що таке Material Design (Матеріальний дизайн) - Wizeclub Education. *Wizeclub Education*. URL: https://wizeclub.education/blog/shho-take-material-design-materialnij-dizajn/ (дата звернення: 04.12.2024).
30. Frank82. Retrofit – библиотека для обмена с интернетом (Часть 1). *Хабр*. URL: https://habr.com/ua/articles/478374/ (дата звернення: 04.12.2024).
31. Retrofit. *Освой програмування граючись*. URL: https://developer.alexanderklimov.ua/android/library/retrofit.php (дата звернення: 04.12.2024).

32. 16.4. Коротко про Gradle – Про Програмування. *Про Програмування – ABout IT And Programming*. URL: https://abitap.com/16-4-korotko-pro-gradle/ (дата звернення: 04.12.2024).
33. ЩО ТАКЕ GRADLE ЯКІ ПРОБЛЕМИ ВІН ВИРІШУЄ. *Joomladom*. URL: https://joomladom.com/vidpovid/sho-take-gradle-yaki-problemi-vin-virishuye.html (дата звернення: 04.12.2024).
34. ЩО ТАКЕ GRADLE ЯКІ ПРОБЛЕМИ ВІН ВИРІШУЄ. *REPORTER | Information portal*. URL: https://reporter.zp.ua/shho-take-gradle-yaki-problemy-vin-vyrishuye.html (дата звернення: 04.12.2024).
35. Android Studio emulator – IT Master - електроніка та програмування. *Головна – IT Master - електроніка та програмування*. URL: https://itmaster.biz.ua/programming/android/android-emulator.html (дата звернення: 04.12.2024).
36. UML. *KDE Documentation* -. URL: https://docs.kde.org/trunk5/uk/umbrello/umbrello/uml-basics.html (date of access: 04.12.2024).
37. Використання уніфікованої мови моделювання UML. *iwanoff.inf.ua*. URL: http://iwanoff.inf.ua/oop_kn/LabTraining05.html (дата звернення: 04.12.2024).
38. МІЖНАРОДНА НАУКОВО-ПРАКТИЧНА КОНФЕРЕНЦІЯ "ПРОБЛЕМИ КОМП'ЮТЕРНИХ НАУК, ПРОГРАМНОГО МОДЕЛЮВАННЯ ТА БЕЗПЕКИ ЦИФРОВИХ СИСТЕМ". *Прикладні проблеми комп'ютерних наук, безпеки та математики*. URL: https://apcssm.vnu.edu.ua/index.php/conf/index (дата звернення: 04.12.2024).

ДОДАТОК А

1. Підстави для розробки

Роботи з розроблення гри здійснюються на підставі написання кваліфікаційної роботи.

2. Призначення розробки

Програма призначена для:

- освітніх цілей. Надання допомоги у вивченні математики шляхом генерації математичних задач різної складності, що дозволяє користувачам покращувати свої навички розв'язування рівнянь та інших математичних виразів;
- самостійної практики. Надання можливість учням та студентам тренуватися на численних варіантах завдань, адаптуючи складність до своїх можливостей;
- підготовки до іспитів. Забезпечення генерацію тестів та вправ для підготовки до різних видів перевірок, включаючи контрольні, екзамени та тести.

3. Вимоги до програми чи програмного продукту

3.1 Вимоги до функціональних характеристик

Створена гра повинна виконувати такі завдання:

- генерація рівнянь наступних видів: прості пропорції, лінійні рівняння, системи лінійних рівнянь з двома змінними, многочлени, квадратні рівняння, ірраціональні рівняння, тригонометричні рівняння та вирази, показникові рівняння, логарифмічні рівняння, складні системи рівнянь (системи з логарифмічними чи тригонометричними елементами);
- вибір користувачем виду рівняння, рівня складності , кількості завдань та кількості варіантів;
- можливість вибору відразу кількох видів рівнянь для генерації ;
- інтерактивний інтерфейс, сприятливий для використання як учителями, так і учнями;

- можливість експорту генерованих прикладів у форматі PDF.

3.2 Вимоги до надійності

Система повинна відповідати наступним параметрам:

- коефіцієнт готовності повинен становити не менше 0,95 (визначається як результат відношення середнього напрацювання на відмову до суми середнього напрацювання на відмову та середнього часу відновлення);

3.3 Умови експлуатації

Програма може використовуватись на Android-пристроях з різними технічними характеристиками.

3.4 Вимоги до інформаційної і програмної сумісності

Android-додаток повинен успішно працювати на всіх пристроях, що використовують операційну систему Android версії 7.0 та вище. Додаток має бути сумісним з різними моделями смартфонів і планшетів, що підтримують ці версії ОС.

3.5 Вимоги до програмної документації

До складу документів обов’язково входять:

- кваліфікаційна робота;
- технічне завдання;
- інструкція користувачу.

4. Техніко-економічні показники

Додаток є у вільному доступі.

5. Стадії і етапи розробки

1. Передпроєктна підготовка (вивчення і аналіз поточних процесів, які необхідно реалізувати).

2. Специфікація вимог до системи (узгодження вимог до системи, що розробляється).

3. Проєктування, розробка і тестування програмного забезпечення (виконання робіт по проєктуванню, визначенню архітектури і розробці програмного забезпечення на підставі складеного технічного завдання).

4. Реалізація програми (впровадження системи, тестування в робочому режимі і здійснення необхідних допрацювань, виявлених в процесі дослідної експлуатації).

6. Порядок контролю і приймання

Приймання результатів надання послуг здійснюється в процесі здачі кваліфікаційної роботи.

ДОДАТОК Б

Інструкція користувачу

1. **Загальні відомості**

   Android-додаток “MathGen”.

2. **Функціональне призначення**

   Програма призначена для освітніх цілей, самостійної практики та підготовки до іспитів.

3. **Умови застосування програми**

   Програма призначена для використання на мобільних пристроях з операційною системою Android. Для коректної роботи додатка необхідно, щоб пристрій мав мінімум Android 7.0 та 100мб вільної пам’яті.

4. **Повідомлення оператору**

   Натиснути на іконку додатку для запуску.

5. **Опис роботи програми**

   Натиснути на іконку додатку для запуску. Відкриється головне меню, де користувач зможе вибрати вид рівняння або створити набір із кількох видів або відкрити папку для перегляду уже згенерованих файлів. При виборі певного виду рівняння з’явиться нове діалогове вікно, з вибором рівня складності, кількості завдань у варіанті та кількості варіантів. Після вибору всіх налаштувань натиснути кнопку «Генерувати» . Програма згенерує PDF-файл з вибраними видами рівнянь та з’явиться спливаюче вікно про це, після цього можна натиснути на назву файлу, щоб перейти до його перегляду або повернутись до генерації наступного виду або закрити програму.

АНОТАЦІЯ

Литвинчук Я.В. Розробка Android-додатку для генерації математичних прикладів на мові програмування Kotlin . Рукопис

Кваліфікаційна робота на здобуття освітнього ступеня «магістр» за спеціальністю 122 Комп’ютерні науки. Волинський національний університет імені Лесі Українки, Луцьк, 2024 р.

Робота присвячена дослідженню сучасних технологій розробки Android-додатків та практичній реалізації мобільного додатку "MathGen". У роботі проаналізовано особливості використання мови програмування Kotlin, бібліотек Jetpack Compose для створення інтерфейсу користувача, Coroutines для управління асинхронними операціями, View Binding для ефективної взаємодії з елементами інтерфейсу, а також RecyclerView і адаптерів для відображення та обробки динамічних списків.

Також розглянуто принципи використання Material Design 3 для дизайну інтерфейсів, системи автоматизації збірки Gradle, Android SDK як основного інструменту розробки та популярних бібліотек GitHub для інтеграції сторонніх рішень.

Розроблений додаток "MathGen" є мобільним рішенням для генерації математичних завдань, яке включає постановку задачі, визначення функціональних вимог, вибір моделі розробки та обґрунтування інструментів. Реалізовано ключові функції: генерація рівнянь 10 видів; вибір користувачем виду рівняння, рівня складності, кількості завдань та кількості варіантів; можливість вибору відразу кількох видів рівнянь для генерації; інтерактивний інтерфейс, сприятливий для використання як учителями, так і учнями; можливість експорту генерованих прикладів у форматі PDF.

Результатом роботи є повнофункціональний додаток, що відповідає сучасним вимогам до розробки мобільних застосунків.

Ключові слова: Android, Kotlin, Jetpack Compose, MathGen, мобільний додаток, Gradle, Material Design, розробка, тестування.

ABSTRACT

Lytvynchuk Y.V. Development of an Android application for generating mathematical examples in the Kotlin programming language. Manuscript

Qualification work for obtaining the degree of "Master" in the specialty 122 Computer Science. Lesya Ukrainka Volyn National University, Lutsk, 2024

The work is devoted to the study of modern technologies for developing Android applications and the practical implementation of the "MathGen" mobile application. The work analyzes the features of using the Kotlin programming language, Jetpack Compose libraries for creating a user interface, Coroutines for managing asynchronous operations, View Binding for effective interaction with interface elements, as well as RecyclerView and adapters for displaying and processing dynamic lists.

The principles of using Material Design 3 for interface design, the Gradle build automation system, Android SDK as the main development tool, and popular GitHub libraries for integrating third-party solutions are also considered.

The developed application "MathGen" is a mobile solution for generating mathematical tasks, which includes problem formulation, definition of functional requirements, selection of a development model and justification of tools. Key functions have been implemented: generation of 10 types of equations; user selection of the type of equation, level of complexity, number of tasks and number of options; ability to select several types of equations for generation at once; interactive interface, convenient for use by both teachers and students; ability to export generated examples in PDF format.

The result of the work is a fully functional application that meets modern requirements for mobile application development.

Keywords: Android, Kotlin, Jetpack Compose, MathGen, mobile application, Gradle, Material Design, development, testing.