Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
Уфимский государственный нефтяной технический университет

Кафедра вычислительной техники и инженерной кибернетики

КУРСОВАЯ РАБОТА
по дисциплине
Мобильные приложения и программирование устройств

ПРИЛОЖЕНИЕ ДЛЯ ИЗУЧЕНИЯ ТЕХНИК СКОРОЧТЕНИЯ

Выполнил студент группы БПО-22-01 А.Р. Риянова
Принял ст. преподаватель Е.В. Дружинская

Дата представления работы: _____
Дата защиты: _____ Результат: _____

Уфа, 2025 г.

СОДЕРЖАНИЕ

ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

1. **Android** — мобильная операционная система, разработанная компанией Google, используемая в качестве целевой платформы для разработки приложения.
2. **API** — программный интерфейс приложения (Application Programming Interface), обеспечивающий взаимодействие между компонентами приложения.
3. **IDE** — интегрированная среда разработки (Integrated Development Environment), использованная для проектирования и программирования приложения (например, Android Studio).
4. **Kotlin** — язык программирования, применённый для реализации программной логики приложения.
5. **RSVP** — метод быстрого последовательного визуального представления текста (Rapid Serial Visual Presentation).
6. **UI** — пользовательский интерфейс (User Interface), включающий визуальные элементы приложения, такие как экраны и элементы управления.
7. **XML** — расширяемый язык разметки (Extensible Markup Language), используемый для создания макетов пользовательского интерфейса и хранения текстовых ресурсов.

ВВЕДЕНИЕ

В условиях стремительного роста информационных потоков способность быстро и эффективно воспринимать текстовую информацию становится ключевым навыком для людей любого возраста. Современные пользователи сталкиваются с перегрузкой графической информацией, что приводит к трудностям в чтении, обусловленным узким полем зрения, регрессией глаз, привычкой к артикуляции при чтении и недостаточной концентрацией внимания. Навык скорочтения позволяет ускорить обработку текстов, улучшить понимание прочитанного, развить память и повысить продуктивность. Разработка мобильного приложения для тренировки скорочтения представляет актуальную задачу, так как обеспечивает доступный и удобный инструмент для совершенствования навыков чтения. Использование платформы Android, доминирующей на рынке мобильных устройств, делает приложение широко доступным для пользователей.

Цель работы. Обосновать актуальность и значимость разработки мобильного приложения для тренировки скорочтения, а также спроектировать и реализовать программный продукт, направленный на повышение скорости и качества восприятия текстовой информации для широкой аудитории.

Задачи. Для достижения поставленной цели необходимо решить следующие задачи:

1. Изучить предметную область, включая особенности скорочтения и потребности пользователей в развитии этого навыка.
2. Провести анализ существующих программных продуктов для тренировки скорочтения, оценив их функциональность, дизайн и логику реализации.
3. Выполнить проектирование мобильного приложения, включающее функциональное моделирование, эскизирование пользовательского интерфейса и логическое моделирование.

4. Реализовать программную часть приложения, включая разработку логики и пользовательского интерфейса.

5. Провести тестирование приложения и сопоставить результаты с поставленной целью.

Объект исследования. Процесс разработки мобильных приложений для операционной системы Android, направленных на совершенствование навыков восприятия информации.

Предмет исследования. Мобильное приложение для тренировки скорочтения, включая его функциональные возможности, дизайн и программную реализацию.

Структура работы включает следующие разделы: обозначения и сокращения, введение, обзор предметной области, проектирование приложения, программная реализация, заключение и список использованных источников. Введение обосновывает актуальность темы, цель и задачи проекта. Раздел «Обзор предметной области» включает анализ особенностей скорочтения и существующих решений. Раздел «Проектирование приложения» охватывает функциональное моделирование, эскизирование интерфейса и логическое моделирование. В разделе «Программная реализация» описывается разработка и тестирование приложения. В заключении подводятся итоги работы и определяются перспективы дальнейшего развития приложения.

1 ОБЗОР ПРЕДМЕТНОЙ ОБЛАСТИ

1.1 Изучение техник скорочтения

Скорочтение — это совокупность техник, направленных на увеличение скорости чтения при сохранении высокого уровня понимания и усвоения текстовой информации. В условиях информационного общества, где люди ежедневно сталкиваются с большими объёмами текстов, такие техники становятся важным инструментом для повышения эффективности обработки информации. Ниже приведён обзор ключевых техник скорочтения, которые помогают развивать навыки быстрого чтения, улучшать концентрацию и память:

1. **Чтение блоками.** Эта техника предполагает восприятие текста не по отдельным словам, а целыми группами слов или фразами. Чтение блоками позволяет расширить поле зрения, охватывая больше текста за один взгляд, и сократить время, затрачиваемое на движение глаз. Это уменьшает количество фиксаций глаз на странице и ускоряет процесс чтения.

2. **Чтение по диагонали.** Данная техника направлена на быстрое сканирование текста по диагонали для улавливания общей структуры и ключевых идей без детального чтения каждого слова. Читатель перемещает взгляд по диагональным траекториям, фокусируясь на основных элементах текста, что позволяет быстро понять суть материала.

3. **Поиск ключевых слов.** Техника заключается в выделении наиболее значимых слов, несущих основную смысловую нагрузку, с игнорированием второстепенных элементов, таких как предлоги, союзы или описательные слова. Это помогает сосредоточиться на главном содержании текста, ускоряя его восприятие и понимание.

4. **Метод указки.** Использование указки, например пальца, ручки или другого ориентира, помогает направлять взгляд вдоль строки текста, поддерживая постоянный ритм чтения. Эта техника минимизирует регрессию

глаз (непроизвольное возвращение к уже прочитанным словам) и способствует более плавному и быстрому чтению.

5. **Слова наоборот.** Данная техника предполагает чтение слов в обратном порядке, что развивает гибкость восприятия текста и зрительное внимание. Практика чтения слов в обратной последовательности тренирует мозг быстрее обрабатывать текстовую информацию и адаптироваться к нестандартным форматам текста.

6. **Предложения наоборот.** Эта методика включает чтение предложений, написанных в обратном порядке, что помогает развивать память и способность анализировать структуру текста. Читатель учится восстанавливать смысл перевёрнутых предложений, что улучшает навыки реконструкции текста и понимания его логики.

Перечисленные техники скорочтения направлены на преодоление типичных препятствий, таких как узкое поле зрения, регрессия глаз, артикуляция (внутреннее проговаривание текста) и недостаточная концентрация. Их использование позволяет пользователям быстрее обрабатывать информацию, улучшать запоминание и повышать продуктивность в работе с текстами.

1.2 Анализ существующих решений

Для формирования концепции разрабатываемого приложения был проведён анализ существующих программных продуктов, предназначенных для тренировки скорочтения. Рассмотрены следующие приложения, доступные на платформе Android: Quickify, Spritz, Readmical и Spreeder. Анализ включает описание их функциональности, преимуществ и недостатков.

1 **Quickify**

Приложение предлагает различные упражнения для развития навыков скорочтения, включая визуальные тренажёры, слепое чтение, распознавание слов и букв, тренировки памяти и внимания. Интерфейс интуитивно понятный, есть встроенные рекомендации и статистика прогресса.

Преимущества: разнообразие упражнений, наглядная визуализация результатов, мотивационные элементы.

Недостатки: часть функций доступна только в платной версии, перегруженность интерфейса в отдельных модулях.

2 **Readmical**

Простое в использовании приложение, основное внимание в котором уделено технике чтения с помощью RSVP (Rapid Serial Visual Presentation) — последовательному отображению слов по одному на экране. Поддерживается загрузка собственных текстов.

Преимущества: минималистичный интерфейс, быстрая работа, возможность кастомизации скорости.

Недостатки: ограниченная функциональность, отсутствуют комплексные тренировки памяти и внимания.

3 **Spritz**

Это приложение реализует фирменную технологию Spritz для показа текста с фиксацией ключевой буквы (Optimal Recognition Point), что позволяет сократить время на перемещение взгляда.

Преимущества: инновационная подача текста, высокая скорость восприятия при адаптации, компактный и удобный интерфейс.

Недостатки: ограниченные настройки, сложности в восприятии длинных и сложных предложений, особенно при высоких скоростях.

4 **Spreeder**

Многофункциональное приложение, предназначенное не только для тренировки скорочтения, но и для улучшения концентрации, памяти и когнитивных способностей. Имеет интеграцию с облачными сервисами и синхронизацию между устройствами.

Преимущества: широкие возможности персонализации, наличие обучающих программ и заданий, профессиональный подход.

Недостатки: англоязычный интерфейс, высокая цена подписки, перегруженность функционалом для неподготовленного пользователя.

2 ПРОЕКТИРОВАНИЕ ПРИЛОЖЕНИЯ

2.1 Функциональная модель

Функциональное моделирование определяет сценарии использования приложения и его основные возможности, обеспечивая соответствие потребностям пользователей. Приложение предназначено для тренировки скорочтения с использованием техник, таких как чтение блоками, чтение по диагонали, поиск ключевых слов, метод указки, слова наоборот и предложения наоборот.

Основные сценарии использования включают:

— **Выбор техники скорочтения**: Пользователь выбирает одну из доступных техник для тренировки (например, чтение блоками или метод указки).

— **Настройка параметров тренировки**: Пользователь задаёт уровень скорости отображения текста.

— **Выполнение упражнений**: Пользователь проходит тренировочные задания, соответствующие выбранной технике, с отображением текста в заданном формате (например, по диагонали или в обратном порядке).

— **Просмотр статистики**: Пользователь получает доступ к результатам тренировок, включая скорость чтения, количество правильных ответов.

Таким образом, функциональная модель приложения охватывает весь цикл взаимодействия пользователя — от выбора техники и настройки параметров до выполнения упражнений и анализа результатов. Такой подход обеспечивает индивидуализацию процесса обучения, способствует системному развитию навыков скорочтения и позволяет пользователю отслеживать личный прогресс, адаптируя тренировки под свои цели и уровень подготовки.

2.2 Эскизирование экранов

Эскизирование экранов является ключевым этапом проектирования пользовательского интерфейса мобильного приложения для тренировки скорочтения. Цель этого этапа — разработать интуитивно понятный, функциональный и визуально привлекательный интерфейс, который обеспечит удобное взаимодействие пользователя с приложением и поддержит выполнение всех сценариев использования, описанных в функциональной модели. На основе анализа потребностей пользователей и функциональных требований были спроектированы основные экраны приложения, каждый из которых соответствует определённому этапу взаимодействия: выбор техники, настройка параметров, выполнение упражнений, просмотр результатов и доступ к дополнительным материалам.

Эскизирование проводилось в среде Android Studio с использованием XML-макетов, что позволило одновременно проектировать визуальную структуру и интегрировать её с программной логикой. Макеты создавались с акцентом на простоту навигации, минимализм и поддержку всех описанных техник скорочтения: чтение блоками, чтение по диагонали, поиск ключевых слов, метод указки, слова наоборот и предложения наоборот. Для демонстрации интерфейса были подготовлены скриншоты экранов, которые иллюстрируют их структуру и функциональность (рисунки 1 - 8). Ниже описаны основные экраны приложения, их назначение и ключевые элементы интерфейса, реализованные в XML.

Основные экраны приложения

1. Экран главного меню.

Назначение: Обеспечивает навигацию по разделам приложения.

Элементы интерфейса: Нижняя панель навигации с вкладками «Упражнения», «Рейтинг», «Материалы».

XML-макет: activity_main.xml.

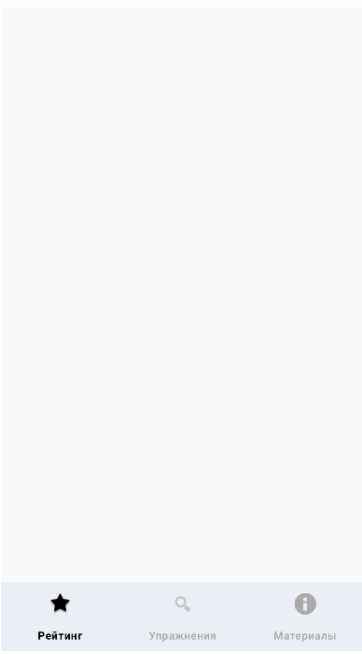

Рис. 1 – Экран главного меню

2. Экран выбора техники.

Назначение: Позволяет выбрать технику скорочтения.

Элементы интерфейса: Список техник (RecyclerView), иконка справки, открывающая подсказку.

XML-макет: fragment_exercises.xml.

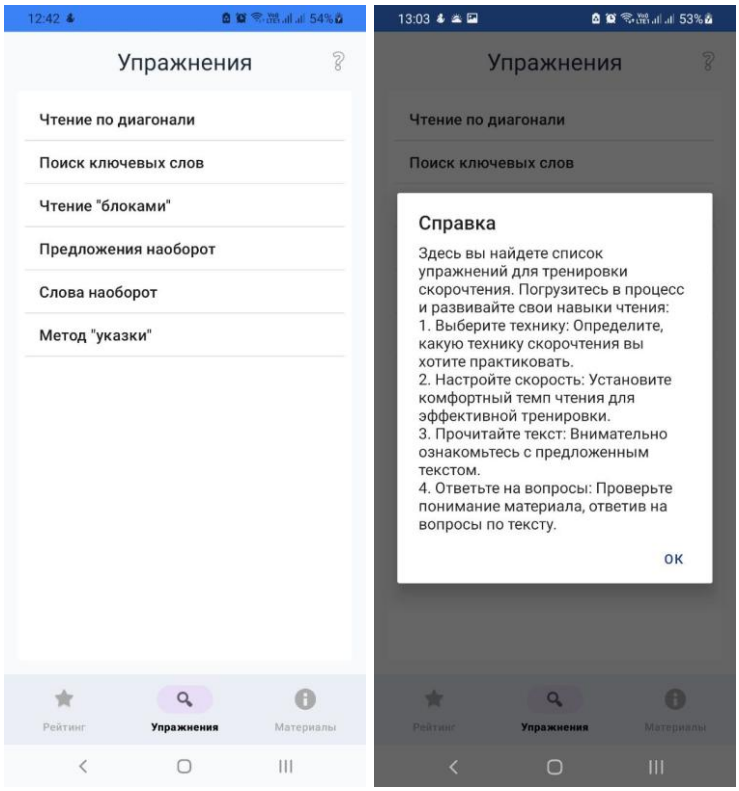

Рис. 2-3 – Экран выбора техники и открывающаяся подсказка

3. Экран выбора скорости.

Назначение: Настройка скорости чтения перед тренировкой.

Элементы интерфейса: Заголовок техники, кнопки для выбора скорости (200, 400, 600 слов/мин).

XML-макет: fragment_speed_selection.xml.

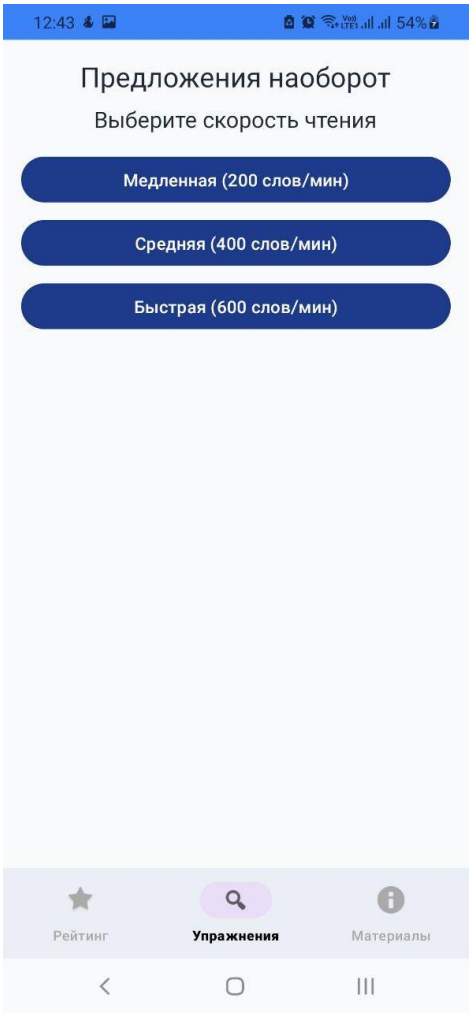

Рис. 4 – Экран выбора скорости

4. Экран выполнения упражнения.

Назначение: Отображает текст с применением выбранной техники.

Элементы интерфейса: Текстовая область (TextView) для отображения текста.

XML-макет: fragment_reading_test.xml.

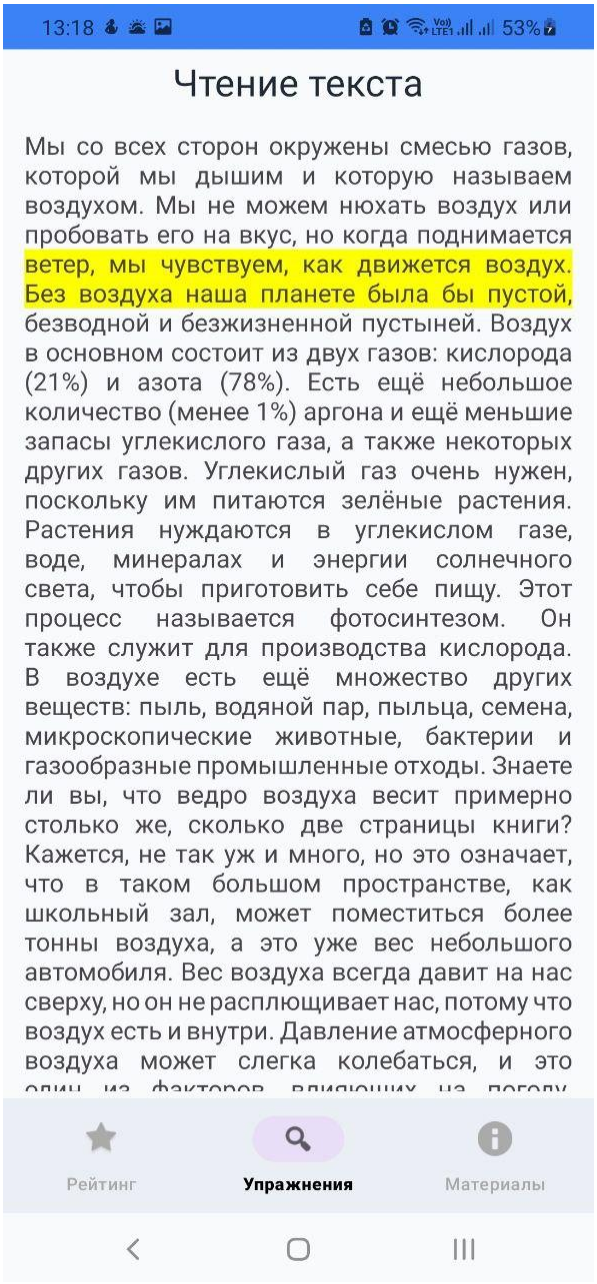

Рис. 5 – Экран выполнения упражнения

5. Экран тестирования.

Назначение: Проверяет понимание текста через вопросы.

Элементы интерфейса: Заголовок вопроса, текст вопроса, варианты ответа (RadioGroup), кнопка «Отправить».

XML-макет: fragment_test.xml.

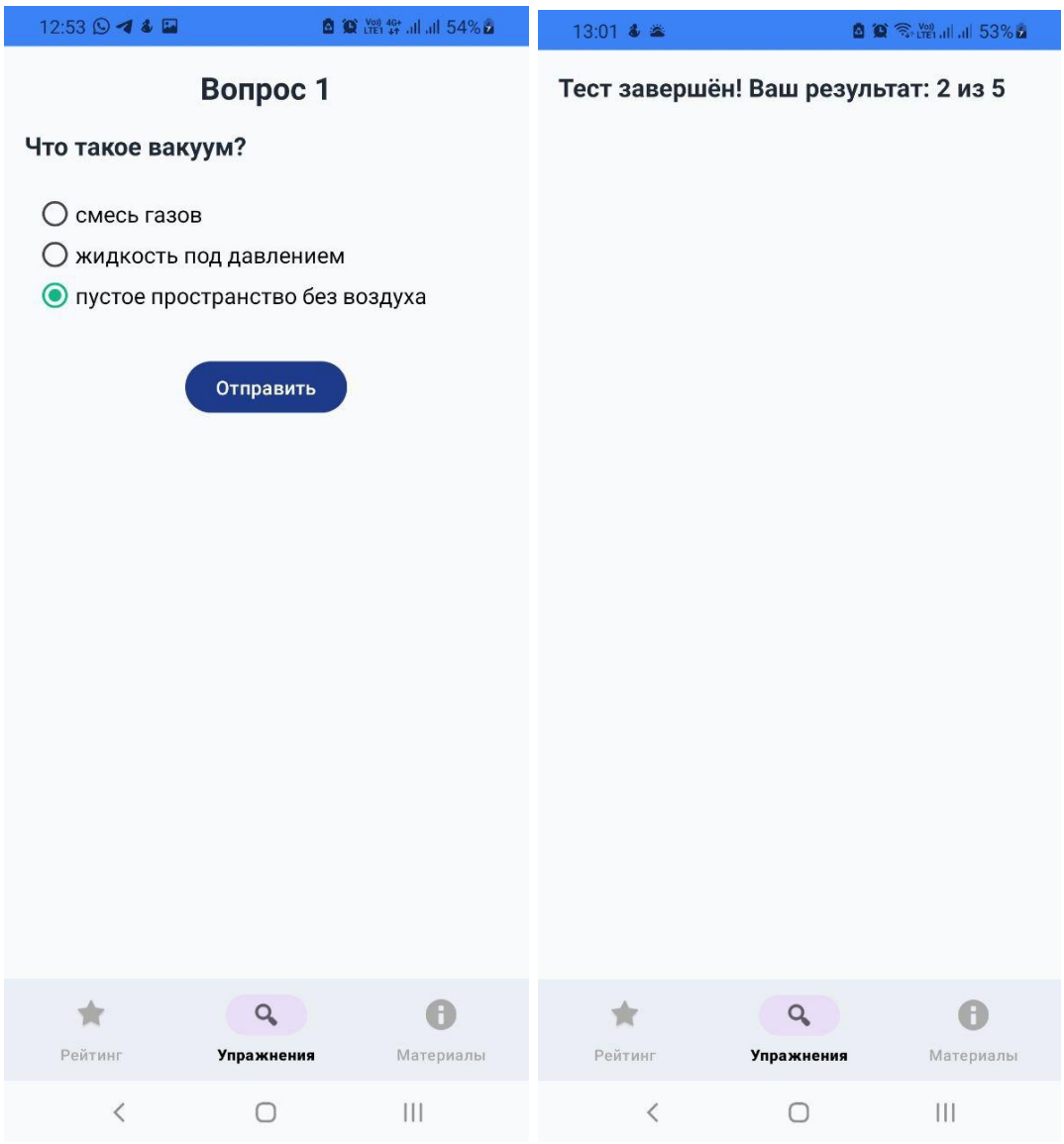

Рис. 6 – Экран тестирования и выведенного резуальта

6. Экран рейтинга.

Назначение: Показывает лучшие результаты по техникам.

Элементы интерфейса: Список результатов (RecyclerView), иконка справки с подсказкой.

XML-макет: fragment_rating.xml.

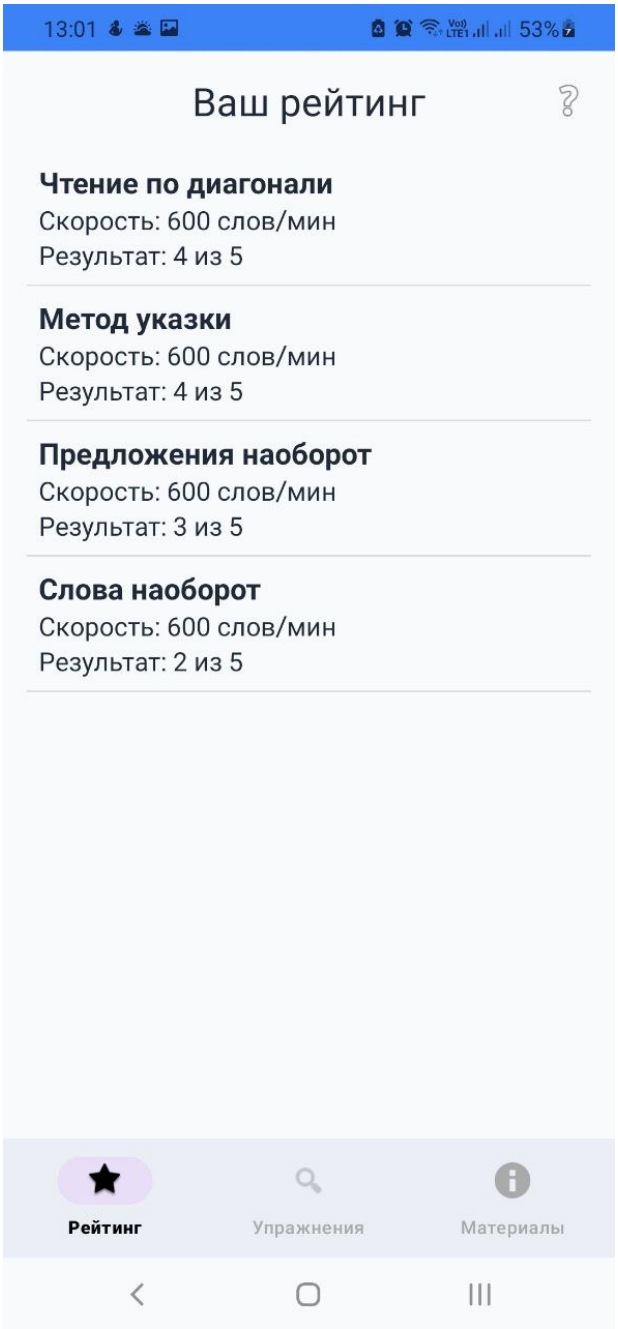

Рис. 7 – Экран рейтинга

7. Экран дополнительных материалов.

Назначение: Предоставляет информацию о техниках.

Элементы интерфейса: Список техник (RecyclerView), иконка справки с подсказкой.

XML-макет: fragment_materials.xml.

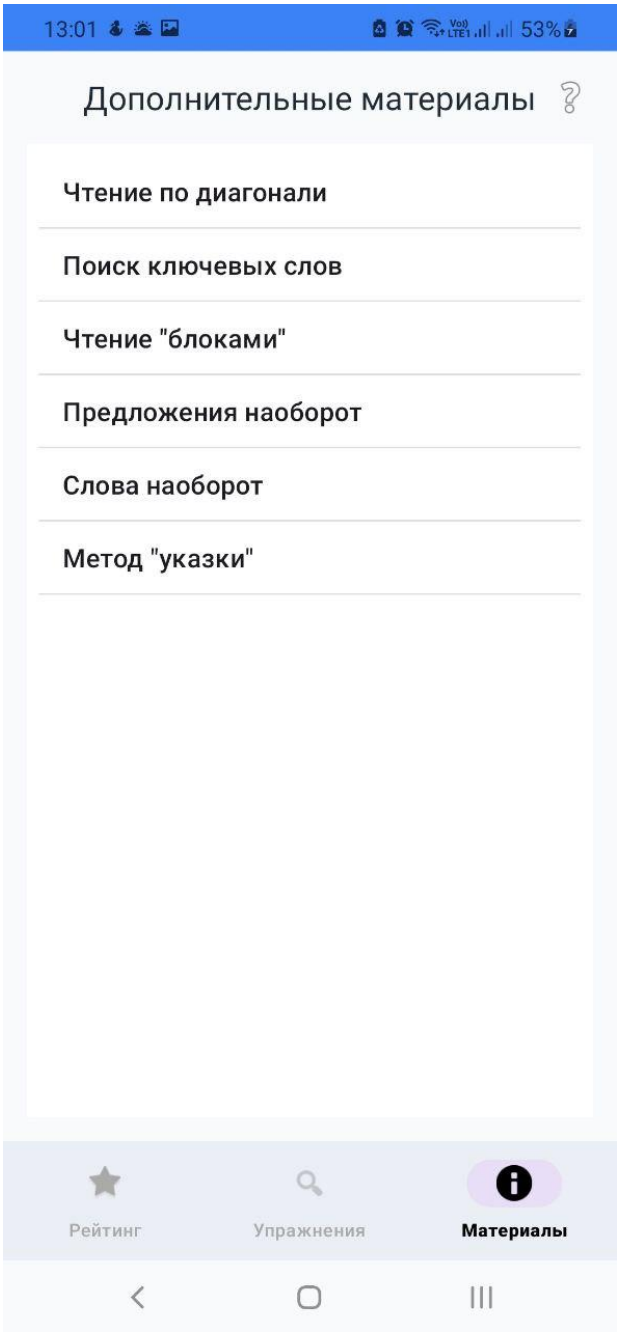

Рис. 8 – Экран дополнительных материалов

8. Экран описания техники.

Назначение: Показывает детали техники и запускает тренировку.

Элементы интерфейса: Кнопка «<» (Назад), описание техники, кнопка «Старт», контейнеры для предварительного просмотра.

XML-макет: fragment_technique_detail.xml.

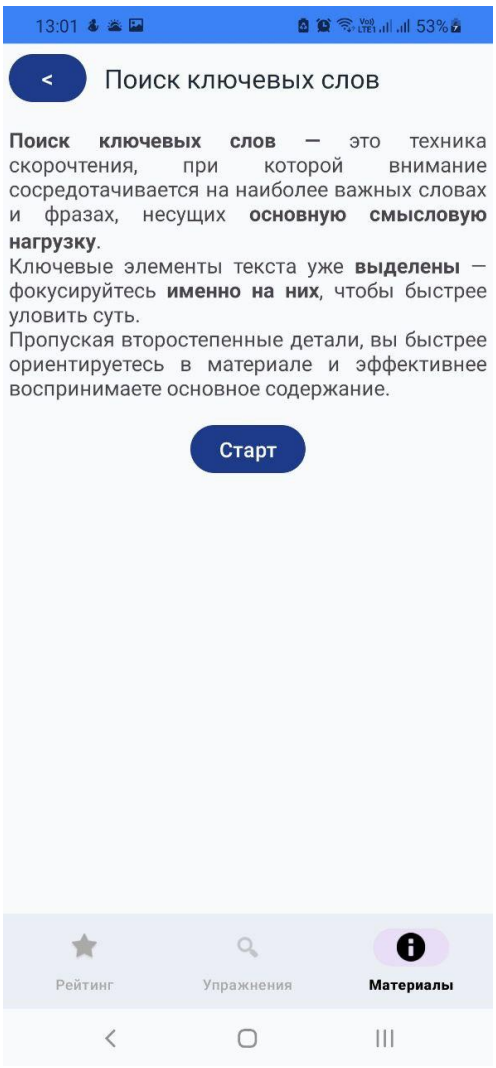

Рис. 9 – Экран описания техники

Принципы дизайна

- **Удобство использования**: Элементы управления (кнопки, списки) и иконки справки размещены интуитивно, обеспечивая лёгкий доступ к функциям.
- **Минимализм**: Экраны содержат только необходимые элементы, упрощая навигацию и фокусируя внимание на задаче.
- **Целевая функциональность**: Каждый экран чётко реализует свой сценарий: выбор техники, тренировка, тестирование или просмотр результатов.
- **Динамичность**: Подсказки и анимации (например, для метода указки или диагонального чтения) делают взаимодействие более живым и понятным.

2.3 Логическая модель

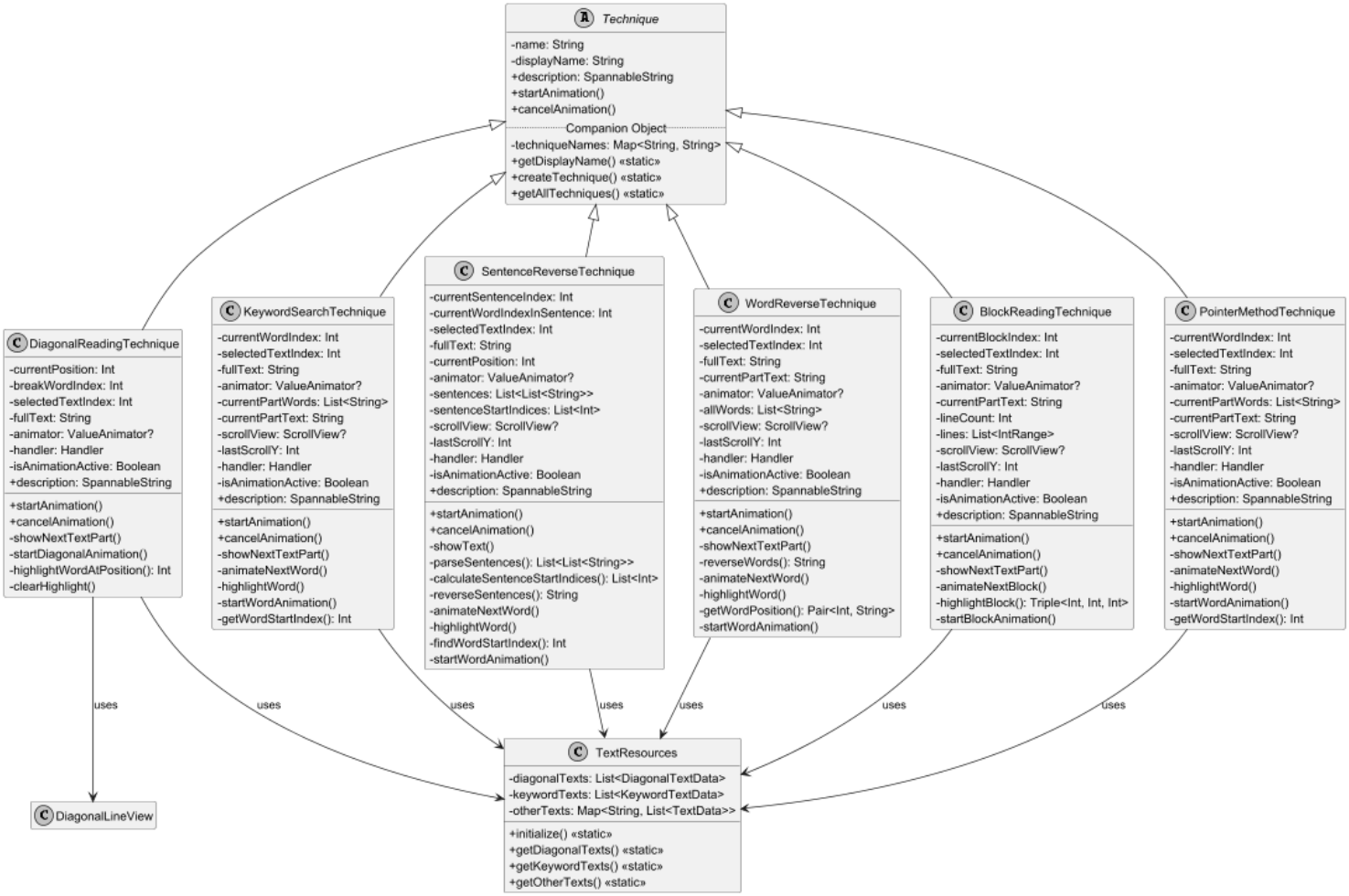

Рис. 10 – Диаграмма классов техник скорочтения и текстовых ресурсов

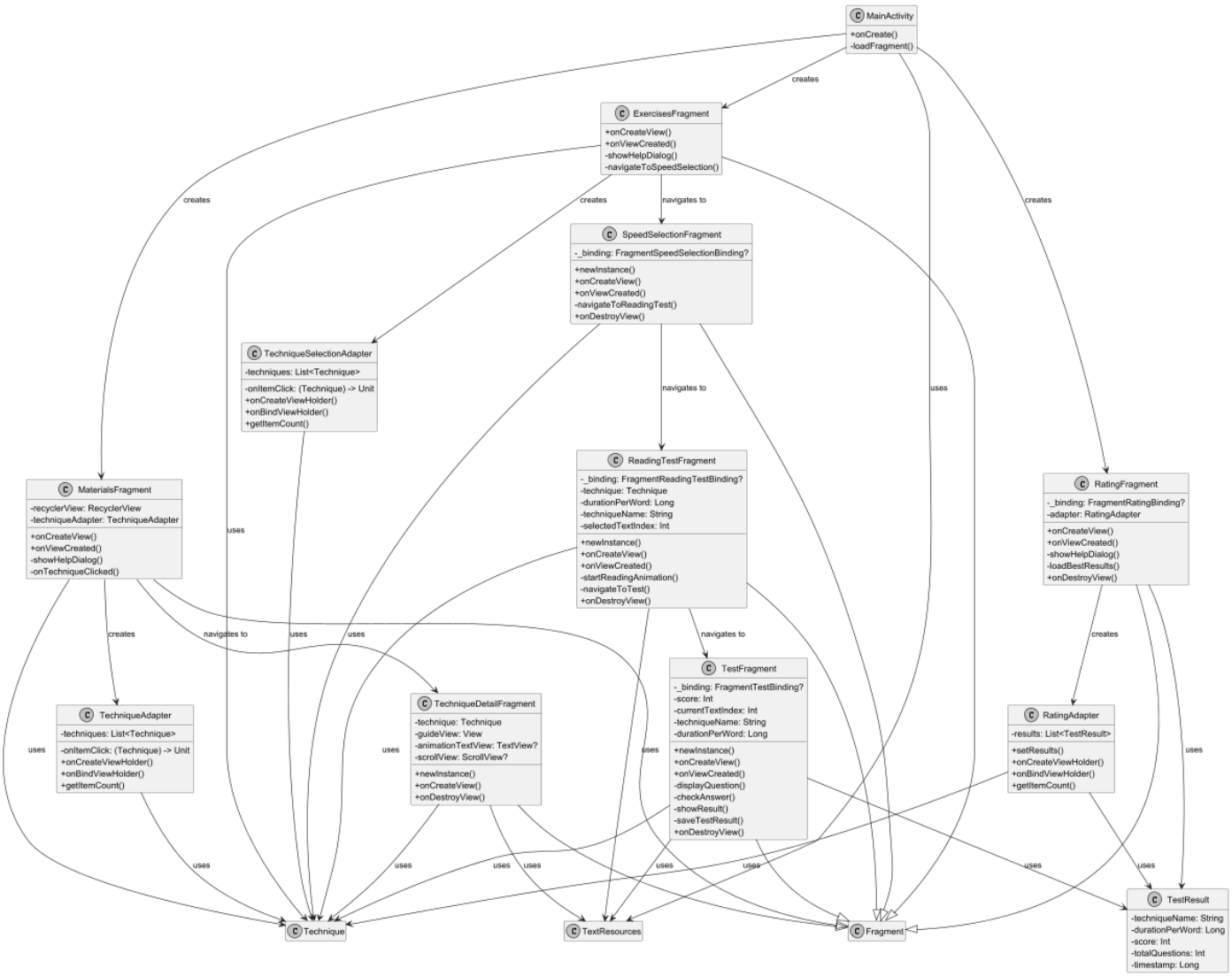

Рис. 11 – Диаграмма классов пользовательского интерфейса и логики приложения

3 ПРОГРАММНАЯ РЕАЛИЗАЦИЯ

3.1 Реализация техник скорочтения

Разработаны четыре техники скорочтения: KeywordSearchTechnique, SentenceReverseTechnique, WordReverseTechnique и BlockReadingTechnique. Все классы наследуются от абстрактного класса Technique, реализуя методы startAnimation, cancelAnimation и свойство description для обработки текста, анимации и предоставления описания техники.

1. KeywordSearchTechnique:

Назначение: Подсвечивает ключевые слова жирным шрифтом и цветом, а текущие слова - жёлтым фоном, способствуя фокусировке на значимых частях текста.

Описание реализации:

- Для подсветки текста используется SpannableString с применением BackgroundColorSpan и ForegroundColorSpan.
- Анимация реализована с помощью ValueAnimator, управляющего перемещением указки (guideView) над текущим словом.
- Метод highlightWord обеспечивает подсветку ключевых слов, загруженных из TextResources, и текущего слова.

Фрагмент кода:

```
private fun highlightWord(textView: TextView) {
    val spannable = SpannableString(currentPartText)

    val existingBackgroundSpans = spannable.getSpans(0, spannable.length,
BackgroundColorSpan::class.java)
    for (span in existingBackgroundSpans) {
        spannable.removeSpan(span)
    }
    val existingStyleSpans = spannable.getSpans(0, spannable.length,
StyleSpan::class.java)
    for (span in existingStyleSpans) {
        spannable.removeSpan(span)
    }
    val existingForegroundSpans = spannable.getSpans(0, spannable.length,
android.text.style.ForegroundColorSpan::class.java)
    for (span in existingForegroundSpans) {
        spannable.removeSpan(span)
    }

    val keyWords =
TextResources.getKeywordTexts().getOrNull(selectedTextIndex)?.keyWords ?:
```

```
emptyList()
    keyWords.forEach { keyWord ->
        var startIndex = currentPartText.indexOf(keyWord, ignoreCase = false)
        while (startIndex != -1) {
            val endIndex = startIndex + keyWord.length
            spannable.setSpan(
                StyleSpan(Typeface.BOLD),
                startIndex,
                endIndex,
                Spannable.SPAN_EXCLUSIVE_EXCLUSIVE
            )
            spannable.setSpan(

android.text.style.ForegroundColorSpan(ContextCompat.getColor(textView.context,
R.color.keyword_color)),
                startIndex,
                endIndex,
                Spannable.SPAN_EXCLUSIVE_EXCLUSIVE
            )
            startIndex = currentPartText.indexOf(keyWord, startIndex + 1,
ignoreCase = false)
        }
    }
```

2. SentenceReverseTechnique:

Назначение: Отображает слова в предложении в обратном порядке для тренировки правильного чтения и профилактики «зеркального» чтения.

Описание реализации:

- Метод reverseSentences перестраивает текст, инвертируя порядок слов в каждом предложении с сохранением пунктуации и коррекцией регистра.

Фрагмент кода:

```
private fun reverseSentences(text: String): String {
    val sentenceRegex = Regex("([^.!?]+)([.!?])")
    return sentenceRegex.findAll(text).joinToString(" ") { matchResult ->
        val body = matchResult.groupValues[1].trim()
        val endPunct = matchResult.groupValues[2]
        val tokenRegex = Regex("""\w+|[^\s\w]""")
        val tokens = tokenRegex.findAll(body).map { it.value }.toList()

        val wordChunks = mutableListOf<WordChunk>()
        var i = 0
        while (i < tokens.size) {
            val tok = tokens[i]
            if (tok.any { it.isLetterOrDigit() }) {
                val puncts = mutableListOf<String>()
                var j = i + 1
                while (j < tokens.size && !tokens[j].any { it.isLetterOrDigit()
}) {
                    puncts.add(tokens[j])
                    j++
                }
                wordChunks.add(WordChunk(tok, puncts))
                i = j
            } else {
```

```
                if (wordChunks.isNotEmpty()) {
                    wordChunks.last().punctuation.add(tok)
                }
                i++
            }
        }

        val sb = StringBuilder()
        for ((word, puncts) in wordChunks.asReversed()) {
            for (p in puncts) {
                sb.append(p)
            }
            if (sb.isNotEmpty() && sb.last() != ' ' && sb.last() !in listOf('-',
'–', '—', ',', '.', '!', '?')) {
                sb.append(' ')
            }
            sb.append(word)
            sb.append(' ')
        }

        var sent = sb.toString().trim()
        val wordPattern = Regex("""\b([a-zA-Za-яА-ЯёЁ]+)\b""")
        val lastWordMatch = wordPattern.findAll(sent).lastOrNull()
        if (lastWordMatch != null) {
            val lastWord = lastWordMatch.value
            val range = lastWordMatch.range
            val correctedLastWord = lastWord.replaceFirstChar {
it.lowercaseChar() }
            sent = sent.substring(0, range.start) + correctedLastWord +
sent.substring(range.endInclusive + 1)
        }

        val finalSent = sent.replaceFirstChar { it.uppercaseChar() }
        "$finalSent$endPunct"
    }
}
```

3. WordReverseTechnique:

Назначение: Переворачивает буквы в каждом слове, сохраняя порядок предложений, для тренировки внимания и движения глаз.

Описание реализации:

- Метод reverseWords переворачивает буквы в словах, сохраняя пунктуацию.

Фрагмент кода:

```
private fun reverseWords(text: String): String {
    val tokenRegex = Regex("""\w+|[^\s\w]""")
    val tokens = tokenRegex.findAll(text).map { it.value }.toList()

    val result = StringBuilder()
    var i = 0

    while (i < tokens.size) {
        val token = tokens[i]

        if (token.any { it.isLetterOrDigit() }) {
```

```
            val word = token
            val punctuations = mutableListOf<String>()

            var j = i + 1
            while (j < tokens.size && !tokens[j].any { it.isLetterOrDigit() }) {
                punctuations.add(tokens[j])
                j++
            }

            val reversedWord = word.reversed()
            result.append(reversedWord)
            punctuations.forEach { result.append(it) }

            if (j < tokens.size) {
                result.append(" ")
            }

            i = j
        } else {
            i++
        }
    }

    return result.toString().trim()
}
```

4. **BlockReadingTechnique**:

Назначение: Выделяет текст блоками по две строки, развивая навык чтения целыми фразами.

Описание реализации:

- Метод highlightBlock подсвечивает блок текста, вычисляя количество слов в блоке.

Фрагмент кода:

```
private fun highlightBlock(textView: TextView): Triple<Int, Int, Int> {
    if (!isAnimationActive) return Triple(0, 0, 0)

    val spannable = SpannableString(currentPartText)
    val existingSpans = spannable.getSpans(0, spannable.length,
BackgroundColorSpan::class.java)
    for (span in existingSpans) {
        spannable.removeSpan(span)
    }

    val firstLineIndex = currentBlockIndex * 2
    val secondLineIndex = min(firstLineIndex + 1, lineCount - 1)
    val startIndex = lines[firstLineIndex].first
    val endIndex = lines[secondLineIndex].last

    val blockText = currentPartText.substring(startIndex, endIndex)
    val wordCountInBlock = blockText.split("\\s+".toRegex()).filter {
it.isNotEmpty() }.size

    val firstLineText = currentPartText.substring(lines[firstLineIndex].first,
lines[firstLineIndex].last)
```

```
    val firstLineWordCount = firstLineText.split("\\s+".toRegex()).filter {
it.isNotEmpty() }.size

    val secondLineText = if (secondLineIndex > firstLineIndex) {
        currentPartText.substring(lines[secondLineIndex].first,
lines[secondLineIndex].last)
    } else {
        ""
    }
    val secondLineWordCount = secondLineText.split("\\s+".toRegex()).filter {
it.isNotEmpty() }.size

    if (startIndex < spannable.length && endIndex <= spannable.length) {
        spannable.setSpan(
            BackgroundColorSpan(Color.YELLOW),
            startIndex,
            endIndex,
            Spannable.SPAN_EXCLUSIVE_EXCLUSIVE
        )
    }

    textView.text = spannable
    return Triple(wordCountInBlock, firstLineWordCount, secondLineWordCount)
}
```

Общие характеристики техник:

- Анимация реализована с использованием ValueAnimator для подсветки текста и перемещения указки.
- Прокрутка текста осуществляется через ScrollView с анимацией (ValueAnimator.ofInt).
- Асинхронная обработка задач выполняется с помощью Handler.

3.2 Управление текстовыми ресурсами

TextResources:

Назначение: Обеспечивает централизованное хранение текстов для техник и тестов, загружая данные из XML-файла (res/xml/texts.xml).

Описание реализации:

- Метод initialize парсит XML, формируя списки DiagonalTextData, KeywordTextData и TextData для различных техник.

Фрагмент кода:

```
fun initialize(context: Context) {
    try {
        val parser = context.resources.getXml(R.xml.texts)
        val diagonalList = mutableListOf<DiagonalTextData>()
        val keywordList = mutableListOf<KeywordTextData>()
```

```
        val otherMap = mutableMapOf<String, MutableList<TextData>>()

        var currentTechnique: String? = null
        var currentText: StringBuilder? = null
        var currentBreakWords: MutableList<String>? = null
        var currentKeyWords: MutableList<String>? = null
        var currentQuestions: MutableList<Pair<String, List<String>>>? = null
        var currentQuestionText: StringBuilder? = null
        var currentAnswers: MutableList<String>? = null

        var eventType = parser.eventType
        while (eventType != XmlPullParser.END_DOCUMENT) {
            when (eventType) {
                XmlPullParser.START_TAG -> {
                    when (parser.name) {
                        "technique" -> {
                            currentTechnique = parser.getAttributeValue(null,
"name")
                            otherMap[currentTechnique] = mutableListOf()
                        }
                        "text" -> {
                            currentText = StringBuilder()
                            currentBreakWords = mutableListOf()
                            currentKeyWords = mutableListOf()
                            currentQuestions = mutableListOf()
                        }
                        "content" -> {
                            currentText?.append(parser.nextText().trim())
                        }
                        "breakWords" -> {
                            currentBreakWords = mutableListOf()
                        }
                        "keyWords" -> {
                            currentKeyWords = mutableListOf()
                        }
                        "word" -> {
                            val word = parser.nextText().trim()
                            currentBreakWords?.add(word)
                            currentKeyWords?.add(word)
                        }
                        "questions" -> {
                            currentQuestions = mutableListOf()
                        }
                        "question" -> {
                            currentQuestionText =
StringBuilder(parser.getAttributeValue(null, "text"))
                            currentAnswers = mutableListOf()
                        }
                        "answer" -> {
                            val answer = parser.nextText().trim()
                            currentAnswers?.add(answer)
                            if (parser.getAttributeValue(null, "correct") ==
"true") {
                            }
                        }
                    }
                }
                XmlPullParser.END_TAG -> {
                    when (parser.name) {
                        "text" -> {
                            if (currentTechnique != null && currentText != null
&& currentQuestions != null) {
                                when (currentTechnique) {
                                    "Чтение по диагонали" -> {
```

```
                                        diagonalList.add(
                                            DiagonalTextData(
                                                text = currentText.toString(),
                                                breakWords = currentBreakWords
?: emptyList(),
                                                questionsAndAnswers =
currentQuestions.toList()
                                            )
                                        )
                                    }
                                    "Поиск ключевых слов" -> {
                                        keywordList.add(
                                            KeywordTextData(
                                                text = currentText.toString(),
                                                keyWords = currentKeyWords ?:
emptyList(),
                                                questionsAndAnswers =
currentQuestions.toList()
                                            )
                                        )
                                    }
                                    else -> {
                                        otherMap[currentTechnique]?.add(
                                            TextData(
                                                text = currentText.toString(),
                                                questionsAndAnswers =
currentQuestions.toList()
                                            )
                                        )
                                    }
                                }
                            }
                            currentText = null
                            currentBreakWords = null
                            currentKeyWords = null
                            currentQuestions = null
                        }
                        "question" -> {
                            if (currentQuestionText != null && currentAnswers !=
null) {
                                currentQuestions?.add(
                                    currentQuestionText.toString() to
currentAnswers.toList()
                                )
                            }
                            currentQuestionText = null
                            currentAnswers = null
                        }
                    }
                }
            }
            eventType = parser.next()
        }

        diagonalTexts = diagonalList
        keywordTexts = keywordList
        otherTexts = otherMap
    } catch (e: Exception) {
        e.printStackTrace()
        diagonalTexts = emptyList()
        keywordTexts = emptyList()
        otherTexts = emptyMap()
    }
}
```

3.3 Реализация пользовательского интерфейса

Разработаны фрагменты и адаптеры, обеспечивающие демонстрацию техник, проведение тестов и отображение рейтинга результатов.

1. ReadingTestFragment:

Назначение: Запускает анимацию чтения для выбранной техники и текста, после чего перенаправляет на тестирование.

Описание реализации:

- Метод startReadingAnimation настраивает контейнер и вызывает анимацию техники.

Фрагмент кода:

```
private fun startReadingAnimation(textView: TextView) {
    val guideView = View(requireContext()).apply {
        visibility = View.INVISIBLE
        layoutParams = FrameLayout.LayoutParams(20, 2)
        setBackgroundColor(android.graphics.Color.BLACK)
    }

    val container = if (technique is DiagonalReadingTechnique)
binding.diagonalContainer else binding.scrollContainer
    container.addView(guideView)

    technique.startAnimation(textView, guideView, durationPerWord,
selectedTextIndex) {
        if (isAdded && !isDetached && !isRemoving) {
            container.removeView(guideView)
            navigateToTest()
        }
    }
}
```

2. TechniqueDetailFragment:

Назначение: Отображает анимацию техники в разделе «Обучение» с фиксированной скоростью (200 мс/слово).

Описание реализации:

- Метод onCreateView настраивает элементы интерфейса и анимацию.

Фрагмент кода:

```
override fun onCreateView(
    inflater: LayoutInflater, container: ViewGroup?,
    savedInstanceState: Bundle?
): View? {
    val view = inflater.inflate(R.layout.fragment_technique_detail, container,
false)
```

```
    val techniqueName = arguments?.getString(ARG_TECHNIQUE_NAME) ?: ""

    // Используем Technique.createTechnique для создания техники
    technique = Technique.createTechnique(techniqueName)

    val titleTextView = view.findViewById<TextView>(R.id.technique_title)
    val descriptionTextView =
view.findViewById<TextView>(R.id.technique_description)
    val scrollContainer = view.findViewById<FrameLayout>(R.id.scroll_container)
    val diagonalContainer =
view.findViewById<FrameLayout>(R.id.diagonal_container)
    val startButton = view.findViewById<Button>(R.id.start_button)
    val backButton = view.findViewById<Button>(R.id.back_button)

    titleTextView.text = technique.displayName
    descriptionTextView.text = technique.description

    guideView = View(requireContext()).apply {
        visibility = View.INVISIBLE
        layoutParams = FrameLayout.LayoutParams(20, 2).apply {
            setMargins(0, 0, 0, 0)
        }
        setBackgroundColor(android.graphics.Color.BLACK)
    }
```

3. TestFragment:

Назначение: Проводит тестирование, отображая вопросы по прочитанному тексту и сохраняя результаты.

Описание реализации:

- Метод saveTestResult сохраняет лучший результат в SharedPreferences, сравнивая с предыдущими.

Фрагмент кода:

```
private fun saveTestResult(normalizedTechniqueName: String, totalQuestions: Int)
{
    val sharedPreferences = requireContext().getSharedPreferences("TestResults",
Context.MODE_PRIVATE)
    val editor = sharedPreferences.edit()
    val key = "result_$normalizedTechniqueName"

    val existingResultJson = sharedPreferences.getString(key, null)
    var shouldSave = true

    if (existingResultJson != null) {
        try {
            val existingResult = JSONObject(existingResultJson)
            val existingScore = existingResult.getInt("score")
            val existingDuration = existingResult.getLong("durationPerWord")

            if (score < existingScore || (score == existingScore &&
durationPerWord >= existingDuration)) {
                shouldSave = false
            }
        } catch (e: Exception) {
            Log.e("TestFragment", "Failed to parse existing result JSON:
```

```
$existingResultJson", e)
        }
    }
```

4. RatingFragment:

Назначение: Отображает рейтинг лучших результатов по техникам в виде списка.

Описание реализации:

- Метод loadBestResults загружает результаты из SharedPreferences, группирует их по техникам и сортирует по времени.

Фрагмент кода:

```
private fun loadBestResults() {
    val sharedPreferences = requireContext().getSharedPreferences("TestResults",
Context.MODE_PRIVATE)
    val allEntries = sharedPreferences.all

    val results = mutableListOf<TestResult>()
    for (entry in allEntries) {
        val jsonString = entry.value as? String ?: continue
        try {
            val json = JSONObject(jsonString)
            val result = TestResult(
                techniqueName = json.getString("techniqueName"),
                durationPerWord = json.getLong("durationPerWord"),
                score = json.getInt("score"),
                totalQuestions = json.getInt("totalQuestions"),
                timestamp = json.optLong("timestamp", 0L)
            )
            results.add(result)
        } catch (e: Exception) {
            e.printStackTrace()
        }
    }

    val bestResults = results.groupBy { it.techniqueName }
        .mapValues { entry ->
            entry.value.maxByOrNull { it.score }!!
        }
        .values
        .sortedByDescending { techniqueResults ->
            results.filter { it.techniqueName == techniqueResults.techniqueName
}
                .maxOfOrNull { it.timestamp } ?: 0L
        }

    adapter.setResults(bestResults)
}
```

5. RatingAdapter:

Назначение: Обеспечивает отображение списка результатов в RatingFragment.

Описание реализации:

- Метод bind в RatingViewHolder привязывает данные результата к элементам интерфейса.

Фрагмент кода:

```
class RatingViewHolder(private val binding: ItemRatingBinding) :
RecyclerView.ViewHolder(binding.root) {
    fun bind(result: TestResult) {
        binding.tvTechniqueName.text =
Technique.getDisplayName(result.techniqueName)
        binding.tvSpeed.text = "Скорость: ${result.durationPerWord} слов/мин"
        binding.tvScore.text = "Результат: ${result.score} из
${result.totalQuestions}"
    }
}
```

6. TestResult:

Назначение: Представляет модель данных для хранения результатов теста (название техники, скорость, баллы, общее количество вопросов, временная метка).

Описание реализации:

- Реализован как дата-класс, используемый в RatingFragment и TestFragment для передачи и хранения данных.

Фрагмент кода:

```
data class TestResult(
    val techniqueName: String,
    val durationPerWord: Long,
    val score: Int,
    val totalQuestions: Int,
    val timestamp: Long
)
```

ЗАКЛЮЧЕНИЕ

В результате выполнения курсовой работы разработано мобильное приложение для операционной системы Android, предназначенное для тренировки навыков скорочтения. Работа направлена на решение актуальной задачи - повышение скорости и качества восприятия текстовой информации в условиях роста информационных потоков. Достижение поставленной цели обеспечено последовательным выполнением всех сформулированных задач.

В рамках изучения предметной области проанализированы особенности скорочтения и ключевые техники, такие как чтение блоками, чтение по диагонали, поиск ключевых слов, метод указки, чтение слов и предложений в обратном порядке. Проведён обзор существующих программных продуктов (Quickify, Readmical, Spritz, Spreeder), что позволило выявить их преимущества и недостатки, а также определить функциональные требования к разрабатываемому приложению.

На этапе проектирования создана функциональная модель, включающая сценарии выбора техник, настройки параметров, выполнения упражнений и просмотра статистики. Разработаны эскизы экранов пользовательского интерфейса с использованием XML-макетов в среде Android Studio, обеспечивающие интуитивную навигацию и поддержку всех предусмотренных техник. Логическая модель представлена в виде диаграмм классов, описывающих структуру техник скорочтения, текстовых ресурсов и компонентов интерфейса.

Тестирование приложения подтвердило его соответствие поставленным требованиям: корректное отображение техник, точность подсчёта результатов и удобство взаимодействия с интерфейсом. Приложение предоставляет пользователям эффективный инструмент для развития навыков скорочтения, позволяя настраивать параметры тренировок и отслеживать прогресс.

В дальнейшем для развития приложения планируется реализация следующих направлений:

1. Введение системы последовательного открытия техник, где изначально доступна только самая лёгкая техника, а последующие становятся доступны после успешного прохождения предыдущей, что обеспечит постепенное усложнение тренировок и повысит мотивацию пользователей.

2. Расширение набора техник скорочтения, включая:

- Технику «зашумлённый» текст, предполагающую добавление визуальных помех (например, наложение случайных символов или искажений), для тренировки концентрации и выделения значимой информации.
- Технику «текст за шторкой», где текст отображается постепенно через движущуюся область видимости, развивая навык быстрого восприятия ограниченного фрагмента текста.
- Технику «текст с закрытой частью строк», при которой часть строк скрыта, заставляя пользователя предугадывать содержание и улучшать контекстуальное понимание.

3. Добавление настроек для отображения текста, позволяющих пользователю изменять размер текста и цветовую схему для повышения комфорта чтения и адаптации под индивидуальные предпочтения.

4. Реализация гибкой настройки скорости анимации, предоставляющей возможность задавать произвольные значения скорости отображения текста, что обеспечит более точную персонализацию тренировок.

Таким образом, поставленные в работе цели и задачи были достигнуты, а разработанное приложение может служить основой для дальнейшего совершенствования полноценного мобильного решения в области тренировки скорочтения.