МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Российский экономический университет им. Г.В. Плеханова»
Московский приборостроительный техникум

Курсовой проект

ПМ 01 Разработка модулей программного обеспечения для компьютерных систем

МДК 01.01 Разработка программных модулей

Специальность 09.02.07 «Информационные системы и программирование»

Квалификация: Программист

Тема: «Разработка мобильного приложения для развития детей»

Пояснительная записка

Листов: 0

Руководитель
_____ / Е.Ю.Бойцова
«____» _____ 2024 год

Исполнитель
_____ / А.В.Плахова
«____» _____ 2024 год

2024

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Российский экономический университет имени Г.В. Плеханова»
МОСКОВСКИЙ ПРИБОРОСТРОИТЕЛЬНЫЙ ТЕХНИКУМ

«УТВЕРЖДАЮ»
Заместитель директора по учебной работе
_____ Д.А. Клопов
«____» _____ 2024 г.

ЗАДАНИЕ

на выполнение курсового проекта (курсовой работы)

Плахова Анастасия Вадимовна

(фамилия, имя, отчество студента — полностью)

студенту группы П50-4-21 специальности 09.02.07 «Информационные системы и программирование» по МДК 01.01 «Разработка программных модулей»

1. Исходные данные к проекту (работе):
   1.1. Тема: «Разработка мобильного приложения для развития детей».
1.2. Состав курсового проекта:
   1.2.1. Задание КП
   1.2.2. Пояснительная записка
   1.2.3. Программа (исходные данные) на электронном носителе
   1.2.4. Презентация и инсталляционный пакет программы на электронном носителе
1.3. Содержание пояснительной записки:
   ВВЕДЕНИЕ
   1. ОБЩАЯ ЧАСТЬ
   1.1. Цель разработки
   1.2. Средства разработки
   2. СПЕЦИАЛЬНАЯ ЧАСТЬ
   2.1. Постановка задачи
      2.1.1. Входные данные предметной области
      2.1.2. Выходные данные предметной области
      2.1.3. Подробные требования к проекту
   2.2. Внешняя спецификация
      2.2.1. Описание задачи
      2.2.2. Входные и выходные данные
      2.2.3. Методы
      2.2.4. Тесты
      2.2.5. Контроль целостности данных
   2.3. Проектирование
      2.3.1. Схема архитектуры приложения
      2.3.2. Логическая схема данных
      2.3.3. Физическая схема данных
      2.3.4. Структурная схема
      2.3.5. Функциональная схема
      2.3.6. Код программы
      2.3.7. Схема тестирования
      2.3.8. Схема пользовательского интерфейса
   2.4. Результат работы программы
   3. ТЕХНОЛОГИЧЕСКАЯ ЧАСТЬ
   3.1. Инструментальные средства
   3.2. Отладка программы
   3.3. Защитное программирование
   3.4. Характеристики программы
   ЗАКЛЮЧЕНИЕ
   СПИСОК ИСПОЛЬЗУЕМЫХ МАТЕРИАЛОВ

Приложение А. Технический проект
Приложение Б. Эскизный проект
Приложение В. Сценарий и результаты тестовых испытаний
Приложение Г. Руководство пользователя
Приложение Д. Макет приложения
Приложение Е. Код программы
Приложение Ж. диаграммы прецедентов

2. Содержание задания по проекту (работе) — перечень вопросов, подлежащих разработке

	Разрабатываемый вопрос	Объем от всего задания, %	Срок выполнения
А	Описательная часть проекта (введение, общее описание и т. д.)	5	10.01.2024
1.	Введение	-	11.01.2024
2.	Цель разработки	-	15.01.2024
3.	Средства разработки	-	20.01.2024
Б	Анализ задачи и её постановка	15	21.01.2024
1.	Определение требований к программе	-	25.01.2024
2.	Спецификация программы (описание задачи, описание входных и выходных данных, методы)	-	05.02.2024
3.	Тесты, контроль целостности данных	-	10.02.2024
В	Проектирование и реализация	55	15.02.2024
1.	Схемы проекта (схема архитектуры, логическая схема данных, физическая схема данных, функциональная и структурная схемы, схема тестирования, код программы и схема пользовательского интерфейса)	-	16.02.2024
2.	Реализация в инструментальной среде	-	16.02.2024
Г	Технологическая часть проекта	5	20.03.2024
1.	Инструментальные средства разработки	-	21.04.2024
2.	Отладка программа	-	22.04.2024
3.	Защитное программирование	-	22.04.2024
4.	Характеристика программы	-	22.04.2024
Д	Программная документация	10	25.04.2024
1.	Приложение А. Технический проект	-	29.04.2024
2.	Приложение Б. Эскизный проект	-	30.04.2024
3.	Приложение В. Сценарий и результаты тестовых испытаний	-	01.05.2024
4.	Приложение Г. Руководство пользователя	-	01.05.2024
6.	Приложение Д. Макет приложения	-	04.05.2024
7.	Приложение Е. Код программы	-	05.05.2024
	Приложение Ж. диаграммы прецедентов		10.05.2024
Е	Экспериментальная часть проекта	10	25.05.2024
1.	Программа на машинном носителе. Информация на носителе разбита на разделы: эксплуатационный пакет, тексты программы, документация.	-	25.05.2024

Руководитель курсового проекта (работы) Екатерина Юрьевна Бойцова, преподаватель

«10» января 2024 года _____ / Е.Ю. Бойцова /

Дата выдачи курсового задания «10» января 2024 года
Срок сдачи законченного проекта (работы) «3» июня 2024 года

Задание принял к исполнению
«10» января 2024 года _____ / А.В. Плахова /

Развивашка

Техническое задание

ТЗ КП 02 01 П50-4-21 23 24-ЛУ

Листов 50

2024

АННОТАЦИЯ

В данном программном документе представлено техническое задание (ТЗ) на разработку программы «Развивашки». ТЗ оформлено в соответствии с требованиями ГОСТ 19.106-78 и ГОСТ 19.104-78. Документ содержит основные разделы, включая введение, основания для разработки, назначение, требования к программе, требования к программной документации, технико-экономические показатели, стадии и этапы разработки, порядок контроля и приемки, а также приложения.

Техническое задание представляет собой основной документ, определяющий требования и цели проекта, устанавливая критерии успешной приемки работы. Разработка ТЗ является важным этапом в создании программного продукта, и его правильное оформление способствует четкому определению задачи и облегчает процесс разработки и контроля за выполнением проекта.

1. Наименование и условное обозначение документа

1.1. Техническое задание оформлено в соответствии с ГОСТ 19.106-78 на листах формата 11 и 12 по ГОСТ 2.301-68, как правило, без заполнения полей листа. Номера листов (страниц) проставлены в верхней части листа над текстом.

1.2. Лист утверждения и титульный лист оформлены в соответствии с ГОСТ 19.104-78. Информационная часть (аннотация и содержание) и лист регистрации изменений не включены в документ.

1. Введение

В данном разделе читатель введен в контекст разрабатываемого программного продукта и содержание документа.

2. Основания для разработки

Здесь указаны документы, на основании которых велась разработка, наименование и условное обозначение темы разработки.

3. Назначение разработки

Описано функциональное и эксплуатационное назначение программы (программного изделия). Важно четко сформулировать, для чего предназначен разрабатываемый продукт.

5. Требования к программе

5.1. Требования к функциональным характеристикам: Здесь описаны функции и возможности программы, ее основные характеристики и особенности работы.

5.2. Требования к надежности: Включают в себя требования к стабильной работе программы и обеспечению безопасности данных.

5.3. Условия эксплуатации: Описаны условия, при которых программа должна работать корректно.

5.4. Требования к составу и параметрам технических средств: Определены минимальные требования к оборудованию, на котором будет выполняться программа.

5.5. Требования к информационной и программной совместимости: Установлены требования к совместимости программы с другими программами и информационными системами.

5.6. Специальные требования: Включают специфические требования, если такие имеются.

6. Требования к программной документации

В данном разделе указаны состав программной документации и специальные требования к ней.

7. Технико-экономические показатели

В разделе «Технико-экономические показатели» приведены ориентировочная экономическая эффективность, предполагаемая годовая потребность и экономические преимущества разработки. Дополнительно, в примечании отмечено, что анализ экономических преимуществ проведен без использования бюджетных средств.

8. Стадии и этапы разработки

В разделе «Стадии и этапы разработки» определены необходимые стадии разработки, этапы и содержание работ. Также определены сроки разработки и исполнители. Важно обратить внимание, что исполнители были определены на ходу разработки, как указано в примечании.

СОДЕРЖАНИЕ

1. ВВЕДЕНИЕ

1.1. Наименование программы

«Развивашка».

1.2. Краткая характеристика области применения программы

Программа предназначена для развития детей,

2. ОСНОВАНИЯ ДЛЯ РАЗРАБОТКИ

2.1. Основание для проведения разработки

Основанием для проведения разработки программы по «Развивашка» является курсовая работа от 2 февраля 2024 года.

2.2. Наименование и условное обозначение темы разработки

Наименование темы разработки: «Развивашка».

3. НАЗНАЧЕНИЕ РАЗРАБОТКИ

3.1. Функциональное назначение программы

Создание приложения, что поможет сильней развить навык мышления у детей. В основе у приложения лежит задумка о создание интуитивно понятной и простой тип ответы на вопросы, где дети будут отвечать на вопросы и развиваться.

3.2. Эксплуатационное назначение программы

Эксплуатационное назначение такой программы может быть в образовательных учреждениях, домашнем обучении, а также в дополнительных образовательных курсах для детей. Программа может быть использована для развития мыслительных навыков, логики, аналитического мышления, а также для развития эмоционального интеллекта.

4. ТРЕБОВАНИЯ К ПРОГРАММЕ ИЛИ ПРОГРАММНОМУ ИЗДЕЛИЮ

4.1. Требования к функциональным характеристикам

Авторизация и регистрация через почту:

- Функция регистрации с формой для ввода почты и пароля пользователя.
- Подтверждение регистрации по электронной почте.
- Функция авторизации, которая позволяет пользователям входить в учетную запись.

Создание различных типов тестов:

- Модуль для добавления тестов с выбором типа ответов (выпадающий список, комбо боксы, ручной ввод и другие).
- Возможность добавления вопросов и вариантов ответов к тестам.

Добавление ребёнка и просмотр результатов:

- Функция для связи учительской учетной записи с учетной записью ребенка по почте.
- Раздел для просмотра результатов ребенка, связанных с учительской учетной записью.

Несколько разделов для тестирования:

- Создание нескольких разделов (например, математика, грамматика, история) для проведения тестов.

Для каждой из этих функций потребуется разработать соответствующий интерфейс на фронтенде и логику на бэкенде, чтобы обеспечить их корректную работу и безопасность.

4.2. Требования к надежности

Программа не должна иметь ошибок сложности мажорные, критические и блокирующие.

4.3. Условия эксплуатации

Программа должна работать на Android.

4.4. Требования к составу и параметрам технических средств

Минимальные требования: Программа должна работать на Android.

4.5. Требования к информационной и программной совместимости

- Программа должна быть совместима с различными типами баз данных.
- Создание интерфейса, подходящего для эффективного образовательного процесса.

4.6. Требования к маркировке и упаковке

Программа будет доступна для скачивания на бесплатной основе на платформе GitHub. Физическая упаковка не предусмотрена.

4.7. Требования к транспортированию и хранению

Программа будет распространяться в электронном виде на платформе GitHub.

4.8. Специальные требования

- Программа должна быть интуитивно понятной и легкой в использовании, даже для пользователей без специальных навыков в области информационных технологий.
- ПО должно быть разработано с помощью технологии android studio на языке Java.

5. ТРЕБОВАНИЯ К ПРОГРАММНОЙ ДОКУМЕНТАЦИИ

5.1. Техническое задание (ТЗ)

 Этот документ описывает общие требования к программе, ее функциональность, особенности и ожидаемые результаты.

 Техническое задание может быть частью программной документации и обычно включает в себя описание требований заказчика к программе.

5.2. Спецификация

Спецификация детализирует требования из Технического задания и описывает, как должна быть реализована программа. Включает в себя технические характеристики, интерфейсы и структуру программы.

5.3. Текст программы

Этот документ может включать исходный код программы, комментарии и инструкции для разработчиков, а также спецификации программных модулей.

5.4. Описание программы

Этот раздел программной документации содержит общее описание программы, ее цели, функциональность и архитектурные особенности.

5.5. Программа и методики испытаний

Включает в себя информацию о том, как проводить тестирование программы, а также результаты тестирования.

5.6. Пояснительная записка

Этот документ может содержать дополнительные пояснения к программе, обоснование выбора технологий и архитектуры, а также прочие комментарии.

6. ТЕХНИКО-ЭКОНОМИЧЕСКИЕ ПОКАЗАТЕЛИ

- Не применимо.

7. СТАДИИ И ЭТАПЫ РАЗРАБОТКИ

Этап 1: Определение требований

- Стадия: Планирование и анализ
- Содержание работ: Определение функциональных и нефункциональных требований, составление Технического задания (ТЗ)
- Документы на выходе: Техническое задание (ТЗ)

Этап 2: Анализ и проектирование

- Стадия: Проектирование системы
- Содержание работ: Разработка архитектуры, создание спецификации, описание программы
- Документы на выходе: Спецификация, Описание программы

Этап 3: Разработка

- Стадия: Разработка
- Содержание работ: Написание кода, создание модулей и компонентов
- Документы на выходе: Текст программы, Описание применения (по мере необходимости), Руководство системного программиста (по мере необходимости), Руководство программиста (по мере необходимости), Руководство оператора (пользователя) (по мере необходимости)

Этап 4: Тестирование и верификация

- Стадия: Тестирование
- Содержание работ: Подготовка и проведение тестов, анализ результатов
- Документы на выходе: Программа и методики испытаний, Сценарий тестов, Результаты тестовых испытаний

Этап 5: Оценка и ревизия

- Стадия: Оценка и обратная связь
- Содержание работ: Проверка соответствия требованиям, выявление дефектов и недоработок, обновление документации
- Документы на выходе: Оценка соответствия требованиям, Дефектные отчеты (по мере необходимости), Корректировка документации (по мере необходимости)

Этап 6: Обратная связь и коррекция

- Стадия: Коррекция и улучшение
- Содержание работ: Учет обратной связи, коррекция и улучшение системы
- Документы на выходе: Обновленные версии документации (по мере необходимости)

Этап 7: Итерации (повторяется по необходимости)

- Стадия: Повторение
- Содержание работ: Повторение всех вышеперечисленных этапов при необходимости

Этап 8: Деплоймент и сопровождение

- Стадия: Внедрение и сопровождение
- Содержание работ: Внедрение программы, сопровождение и поддержка
- Документы на выходе: Развернутая система (готовая к внедрению), Обновленные версии документации (по мере необходимости), Пояснительная записка.

8. ПОРЯДОК КОНТРОЛЯ И ПРИЕМКИ

8.1. Виды испытаний:

Функциональное тестирование: Проверка выполнения функциональных требований.

Интеграционное тестирование: Проверка работоспособности программы на заданной платформе, правильное взаимодействие БД и программы, безошибочное взаимодействие между различными модулями программы.

8.2. Общие требования к приемке работы:

Выполнение всех требований, описанных в ТЗ, успешное прохождение испытаний и утверждение со стороны заказчика.

УТВЕРЖДЕН
ЭП КП 02.01 П50-4-21 21 24-ЛУ

Разработка информационной системы по программе “Развивашка”

Эскизное задание

ЭП КП 02 01 П50-4-21 21 24

Листов 50

2024

АННОТАЦИЯ

В данном программном документе представлено эскизный проект на разработку программы «Развивашки». Документ содержит основные разделы, включая введение, описание проблемы, цели и задачи, анализ аналогов, описание концепции проекта, технические аспекты, план работ, бюджет и ресурсы, ожидаемые результаты, а также заключение.

СОДЕРЖАНИЕ

1. ВВЕДЕНИЕ

1.1. Описание цели проекта и контекста, в котором он разрабатывается

Цель проекта заключается в разработке приложения, которое способствует развитию мыслительных навыков у детей. Оно будет предоставлять детям возможность отвечать на вопросы и развиваться, предлагая им интуитивно понятные и простые типы ответов.

Контекст, в котором разрабатывается проект, связан с потребностью в развитии когнитивных навыков у детей в современном образовательном процессе. Такое приложение может быть использовано в дополнительных образовательных курсах, домашнем обучении или в школе, чтобы помочь детям развить логическое мышление, аналитические способности и эмоциональный интеллект.

Таким образом, разрабатываемое приложение будет инструментом поддержки образовательных процессов, направленным на развитие у детей навыков мышления и способностей к анализу.

2. ОПИСАНИЕ ПРОБЛЕМЫ ИЛИ ЗАДАЧИ

2.1. Описание проблемы или задачи, которую решает проект

Проблема, которую решает проект, заключается в том, что современные дети сталкиваются с рядом проблем, связанных с развитием мышления. Возрастает количество заболеваний, связанных с плохим развитием мозга, а также снижается способность к анализу и критическому мышлению. Приложение направлено на решение этих проблем путем создания увлекательной и интерактивной платформы для развития мышления у детей.

Проект является актуальным, так как современные дети проводят много времени за экранами гаджетов, что не всегда способствует развитию мышления и критического мышления. Приложение предлагает новый подход к развитию умственных способностей детей, направляя их на путь к более эффективному и глубокому мышлению.

Таким образом, создание приложения, которое поможет улучшить умственные способности детей, является важной задачей, решение которой может иметь значительное влияние на будущее поколение.

2.2. Идентификация основных проблем и требований

2.2.1. Идентификация основных проблем

– Недостаточная стимуляция умственного развития: Сложно найти увлекательные и интерактивные способы развития мышления, особенно в мире, где дети проводят большую часть времени перед экранами гаджетов.

– Недостаток практического опыта: Дети имеют недостаточно возможностей для применения и развития своих мыслительных способностей на практике.

– Низкая мотивация: Многие упражнения по развитию мышления могут казаться скучными и однообразными для детей, что уменьшает их мотивацию к развитию мыслительных навыков.

2.2.2. Требования для проекта "Развивашка":

– Возможность ученикам проходить разные тесты и просматривать свои результаты, так же для учителей возможность просматривать результаты тестов своих учеников. Так же возможность создавать и удалять предмет и тему для админа, и создавать вопросы для менеджера.

3. ЦЕЛИ И ЗАДАЧИ

3.1. Определение основных целей проекта

Основная цель данного проекта заключается в создании приложения, направленного на развитие навыков мышления у детей. Это приложение должно обеспечивать интуитивно понятные и простые задания, которые помогут детям развиваться путем ответов на вопросы и выполнения различных упражнений.

3.2. Установление конкретных задач, необходимых для достижения целей

1. Для достижения цели создания приложения для развития навыков мышления у детей мы можем определить следующие конкретные задачи:

2. Исследовать основные принципы развития навыков мышления у детей и определить, какие виды упражнений и заданий будут наиболее эффективными для их развития.

3. Разработать интерфейс приложения, который будет доступен и понятен детям разного возраста.

4. Создать базу данных вопросов и заданий для приложения, учитывая возрастные особенности и уровень развития целевой аудитории.

5. Разработать систему обратной связи, которая будет помогать детям отслеживать свой прогресс и получать мотивацию для дальнейшего развития навыков мышления.

6. Провести тестирование и анализ эффективности приложения для корректировки и улучшения его функционала.

7. АНАЛИЗ АНАЛОГОВ

7.1. Обзор существующих аналогичных продуктов или решений на рынке

В настоящее время на рынке существует множество приложений и программ, направленных на развитие навыков мышления у детей. Ниже представлен обзор некоторых из них:

"Lumosity Kids" - это приложение, разработанное для детей. Оно содержит игры и упражнения, направленные на развитие памяти, внимания, логического мышления и других когнитивных навыков.

"CogniFit for Kids" - это приложение, предназначенное для улучшения когнитивных функций у детей. Оно включает в себя различные игры и упражнения, направленные на развитие внимания, памяти, умения обучаться, переключения внимания и других навыков.

"BrainPOP Jr." - это образовательная платформа, предназначенная для детей. Она предлагает анимационные видеоролики, интерактивные игры и упражнения, которые помогают развивать мышление, понимание причинно-следственных связей и другие когнитивные навыки.

7.2. Анализ их сильных и слабых сторон

Вышеперечисленные аналоги имеют свои сильные и слабые стороны. Сильные стороны включают широкий функционал, интеграцию с другими продуктами, мощные аналитические возможности, простой интерфейс, облачное решение и регулярные обновления.

Однако, у этих программ также есть свои слабые стороны, такие как высокая стоимость внедрения и поддержки, сложность настройки и интеграции с другими системами, ограниченный функционал по сравнению с более крупными системами и некоторые пользователи отмечают недостаточные аналитические возможности.

8. ОПИСАНИЕ КОНЦЕПЦИИ ПРОЕКТА

8.1. Подробное описание того, как будет работать проект

1. Открытие системы:

– Пользователь (учитель) входит в систему с помощью своих учетных данных (почта и пароль).

– После успешного входа учитель попадает на главный экран системы.

2. Создание нового вопроса:

– Администратор выбирает опцию "Добавить новый товар" и вводит основные данные о товаре (название, описание, категория, цена, количество и т.д.).

– Система самостоятельно создает учетную запись для нового товара.

3. Управление запасами:

– Администратор отслеживает поступление и расход товаров, обновляя информацию о запасах при поступлении и продаже товаров.

4. Учет продаж:

– Система автоматически регистрирует продажи, учитывая количество и стоимость проданных товаров.

5. Учет поставщиков:

– Администратор системы вносит информацию о поставщиках и связывает их с товарами, управляя поставками и ценами.

6. Администрирование категорий и атрибутов товаров:

– Администратор системы создает и управляет категориями товаров, а также определяет атрибуты, по которым можно фильтровать товары.

7. Закрытие системы:

– После завершения работы администратор выходит из системы, обеспечивая безопасность и конфиденциальность данных.

8.2. Описание основных функций и характеристик

Программа "Развивашка" должна предоставлять следующие функции:

– Окно регистрации с двумя 3 для ввода логина,почты и пароля

– Возможность отсортировать предмет и тему для прохождения теста

– Возможность создать предмет, тему для администратора

– Возможность создать несколько типов вопроса для роли менеджера

– Возможность просмотреть свои результаты тестов для каждого пользователя

– Возможность просмотреть результаты тестов в качестве учителя по запросу

– Возможность удалить предмет и тему для администратора

Условия эксплуатации:

– Доступ к интернету для хранения данных в облачной базе данных.

9. ТЕХНИЧЕСКИЕ АСПЕКТЫ

9.1. Описание используемых технологий и платформ

Для реализации проекта Android studio планируется использование языка программирования java.

9.2. Определение необходимых ресурсов и инфраструктуры

В качестве базы данных предусмотрено Firebase для эффективного хранения и управления информацией.

10. БЮДЖЕТ И РЕСУРСЫ

10.1. Определение бюджета проекта и распределение ресурсов

Не предусмотрено.

10.2. Установление ограничений и оценка затрат

Проведение практики ориентировочной экономической эффективности осуществляется без привлечения бюджетных средств в рамках курсового проекта по Технологии разработки программного обеспечения (КП 02.01). Анализ экономических преимуществ проводится в рамках курсового проекта по Технологии разработки программного обеспечения (КП 02.01) и не предполагает использования бюджетных средств.

11. ОЖИДАЕМЫЕ РЕЗУЛЬТАТЫ

Ожидаемые результаты включают успешное внедрение приложения для развития детей, и отслеживания результатов тестов.

Повышение уровня развития мышления у детей: Приложение разработано с учетом когнитивных особенностей детей и предоставляет им интерактивные упражнения, способствующие развитию логики, памяти, внимания, воображения и творческого мышления.

Улучшение аналитических и критических способностей: Упражнения в приложении требуют от детей анализа информации, решения проблем и выработки собственных суждений, тем самым развивая их аналитические и критические способности.

Повышение мотивации к обучению: Благодаря увлекательному и интерактивному формату приложение делает обучение более интересным и увлекательным для детей, повышая их мотивацию к познанию нового.

Улучшение школьной успеваемости: Развитые мышление и аналитические способности являются основой для успешного обучения в школе. Приложение помогает детям заложить прочный фундамент для дальнейшего академического успеха.

Повышение самооценки: Успешное выполнение упражнений в приложении дает детям чувство удовлетворения и повышает их самооценку.

12. ЗАКЛЮЧЕНИЕ

Автоматизированная система учета запасов и товаров для розничных магазинов позволит значительно повысить эффективность процессов управления запасами, улучшить аналитику и отчетность, соблюдать законодательство в области налогообложения, повысить производительность труда сотрудников и улучшить обслуживание клиентов. Благодаря удобному интерфейсу и более эффективным процессам, данная система способствует оптимизации управления запасами и содействует росту бизнеса организации.

УТВЕРЖДЕН
ТП КП 02.01 П50-4-21 23 24-ЛУ

АННОТАЦИЯ

В данном программном документе представлен технический проект на разработку программы «Развивашка». Документ содержит основные разделы, включая введение, где ожидается представление проекта и его контекста, а также обоснование его необходимости.

Цель проекта, где ожидается определение основной цели проекта, то есть то, чего ожидается достижения в результате его реализации.

Технические аспекты, где будет описание использованных технологий, инфраструктуры и других технических аспектов проекта, описание входных и выходных данных, включая ERD (схему сущность-связь) и словарь данных.

Схема интерфейса, где ожидается представление схемы интерфейса проекта.

Структурную схему, где ожидается представление структурной схемы проекта.

функциональную схему, где ожидается представление функциональной схемы проекта.

Бюджет и ресурсы, где ожидается определение бюджета проекта и распределение ресурсов, а также оценка затрат.

План работ, где ожидается определение этапов разработки проекта, сроков выполнения и распределение задач между членами команды.

Ожидаемые результаты, где ожидается описание конечных результатов проекта и ожидаемых достижений.

Риски и меры по их снижению, где ожидается описание потенциальных рисков проекта и ожидаемых мер по их снижению.

Также заключение, где ожидается подведение итогов и обоснование целесообразности продолжения проекта.

СОДЕРЖАНИЕ

1. ВВЕДЕНИЕ

1.1. Представление проекта и его контекста

Проект “Развивашка” разработана для прохождения тестов учеников, так же для учителей, что бы учителя могли просматривать результаты пользователей, а ученики могли проходить различные тесты и развиваться в разных областях. Эта система предоставляет возможность улучшить свои знания в той или иной области. Так же в данной системе возможно добавлять тесты, вопросы, предметы и темы.

2. ЦЕЛЬ ПРОЕКТА

2.1. Определение основной цели проекта

Основная цель проекта - разработка информационной системы. Ее основная цель - обеспечить персонализированный подход к обучению: ученики могут проходить тесты по различным предметам и темам, чтобы улучшить свои знания, а учителя могут отслеживать прогресс своих учеников, просматривая результаты тестов. “Развивашка” позволяет проходить тесты по различным предметам и темам, получать обратную связь и анализировать результаты, а также создавать и редактировать учебный материал. Учителя могут просматривать результаты учеников по каждому тесту и предмету, анализировать прогресс и выявлять области, требующие дополнительного внимания.

3. ЗАДАЧИ ПРОЕКТА

3.1. Установление конкретных задач, необходимых для достижения целей

Создать следующие для роли администратор.

Раздел “Страница администратора”.

1. На странице " Страница администратора" администратор сможет видеть кнопку по переходу на создание и удаления предмета и темы.

2. Для удобства поиска администратор может воспользоваться функцией поиска по имени или по email адресу пользователя.

Раздел “Удаление и создание предмета и темы”

1. На странице “Удаление и создание предмета и темы” администратор может создавать, удалять сам предмет или тему.

Раздел "Роль менеджера":

1. При авторизации менеджера он попадает на свою личную страницу, где он может перейти на страницы для создания вопросов по разным кнопкам.

2. Так же он может перейти на страницу для создания теста.

Раздел "Страница ученика":

1. Здесь пользователь авторизуется за ученика и может перейти на страницу для просмотра своих результаты или пройти тесты.

2. Нажав на кнопку просмотреть тесты то он переходит на тест где может выбрать предмет и тему для теста.

Раздел “Учитель”

1. Здесь пользователь после авторизации попадает на свою страницу и узнав номер ученика может просмотреть его результаты по тестам.

4. ТЕХНИЧЕСКИЕ АСПЕКТЫ

4.1. Описание использованных технологий и инфраструктуры

Для разработки программы планируется использование java Android studio. В качестве базы данных предусмотрено FireStore.

4.2. ERD-диаграмма

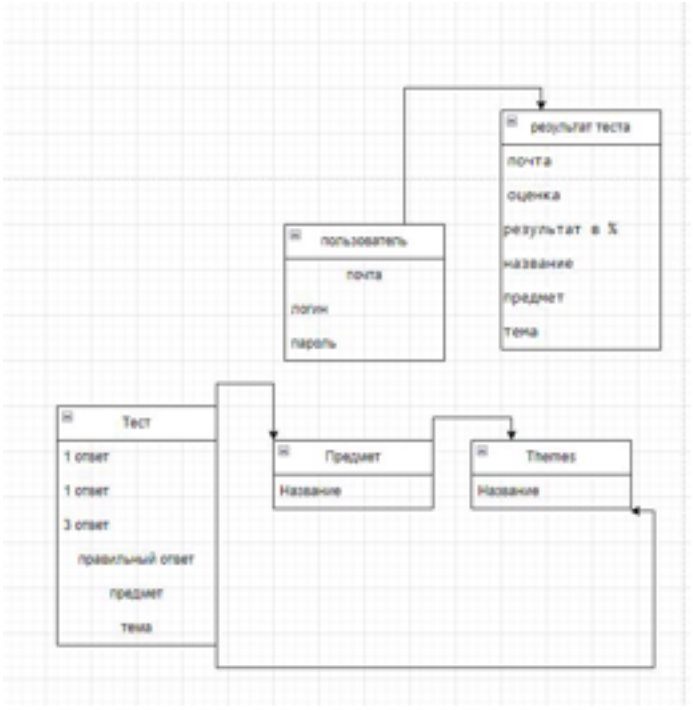

Рисунок 1 инфологическая модель данных

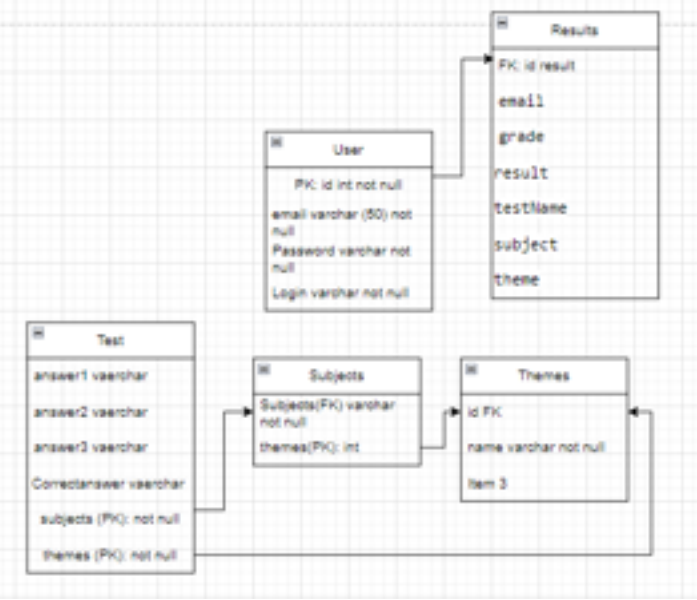

Рисунок 2 - Даталогическая модель данных

Таблица 1 - Словарь данных

Наименование поля	Ключ	Тип данных	Описание

1	3	4	5
User			
id	PK	int	индефекатор
Email		varchar	почта
Password		varchar	почта
Login		varchar	Логин
grade		varchar	Оценка
result		varchar	Сколько баллов
subject		varchar	Название предмета
testName		varchar	Название теста
Theme		varchar	Тема
name		varchar	Название предмета
themes		varchar	Темы
Answer1		Varchar	1 вариант ответа
Answer2		Varchar	2 вариант ответа
Answer3		Varchar	3 вариант ответа
CorrectAnswer		varchar	Правильный вариант ответа
Qvestion_type		varchar	Тип вопроса

4.3. Входные данные

Таблица 2 - Входные данные приложения

Имя	Ограничения	Обязательное	Структура, формат	Форма ввода
1	2	3	4	5
Входные данные				
Почта	Максимальная длина строки 255 символов	ДА	Строка. Формат почты:[a-zA-Z0 9.!#$%&’*+/=?^_ { |}~-]+@[a-zA-Z0-9-]+(?:\.[a-zA-Z0-9-]+)3, 255}	Текстовое поле
Логин	Максимальная длина строки 255 символов	Да	Строка	Текстовое поле
Пароль	Максимальная длина строки 255 символов	Да	Строка	Текстовое поле

Оценка	Максимальная длина 1 символов	Да	Строка	Текстовое поле
Результат	Максимальная длина строки 255 символов	Да	Строка	Текстовое поле
Предмет	Максимальная длина строки 255 символов	Да	Строка	Текстовое поле
Тема	Максимальная длина строки 255 символов	Да	Строка	Текстовое поле
Название теста	Максимальная длина строки 255 символов	Да	Строка	Текстовое поле
1 вариант ответ	Максимальная длина строки 255 символов	Да	Строка	Текстовое поле

2 вариант ответ	Максимальная длина строки 255 символов	Да	Строка	Текстовое поле
3 вариант ответ	Максимальная длина строки 255 символов	Да	Строка	Текстовое поле
Правильный вариант ответ	Максимальная длина строки 255 символов	Да	Строка	Текстовое поле
Тип теста	Максимальная длина строки 255 символов	Да	Строка	Текстовое поле

4.4. Выходные данные

Таблица 3 - Выходные данные приложения

Имя	Обязательное	Структура, формат	Форма вывода
1	3	4	5
Выходные данные			
Вопросы	Да	строка	Текстовое поле

Личный кабинет	да	строка	Текстовое поле
Результаты тестов	Да	строка	Текстовое поле
Тесты	Да	строка	Текстовое поле

5. СХЕМА ИНТЕРФЕЙСА

5.1. Схема интерфейса

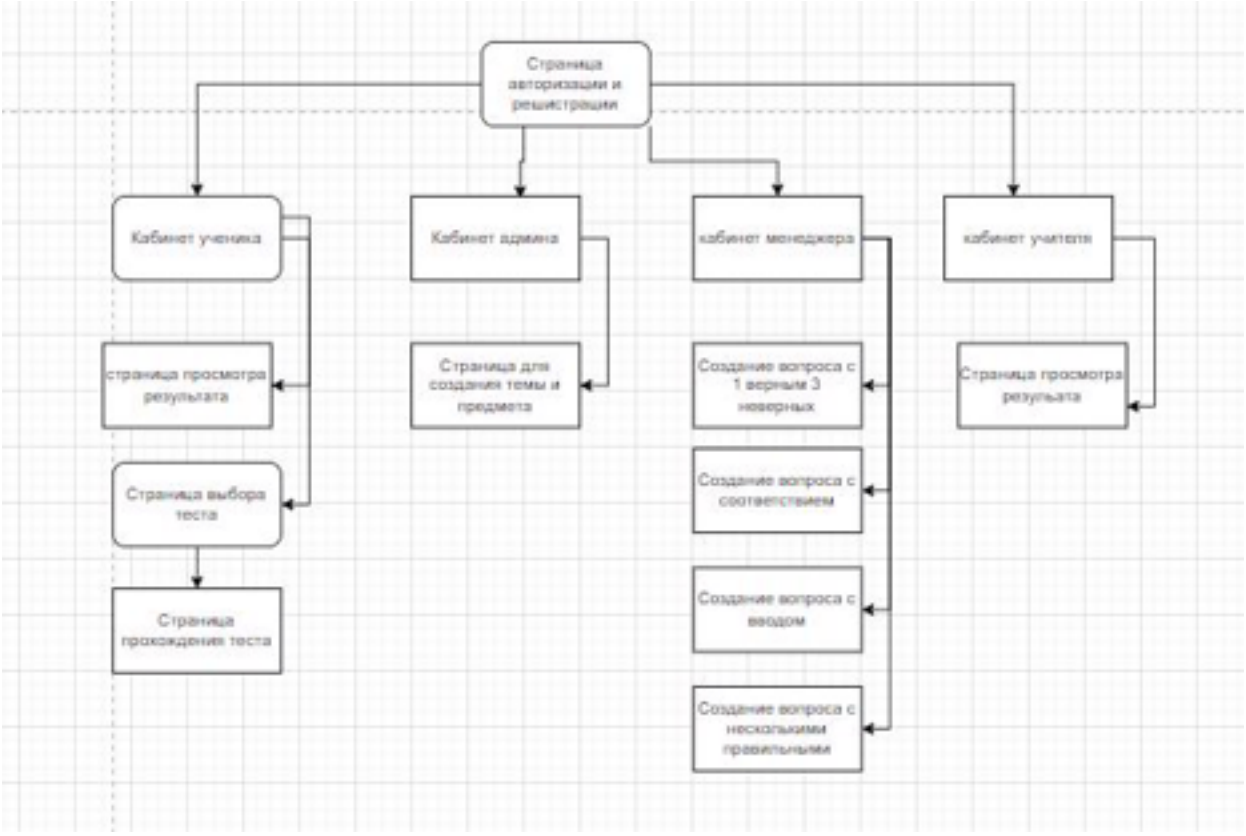

Рисунок 3 - Схема интерфейса

6. СТРУКТУРНАЯ СХЕМА

6.1. Структурная схема

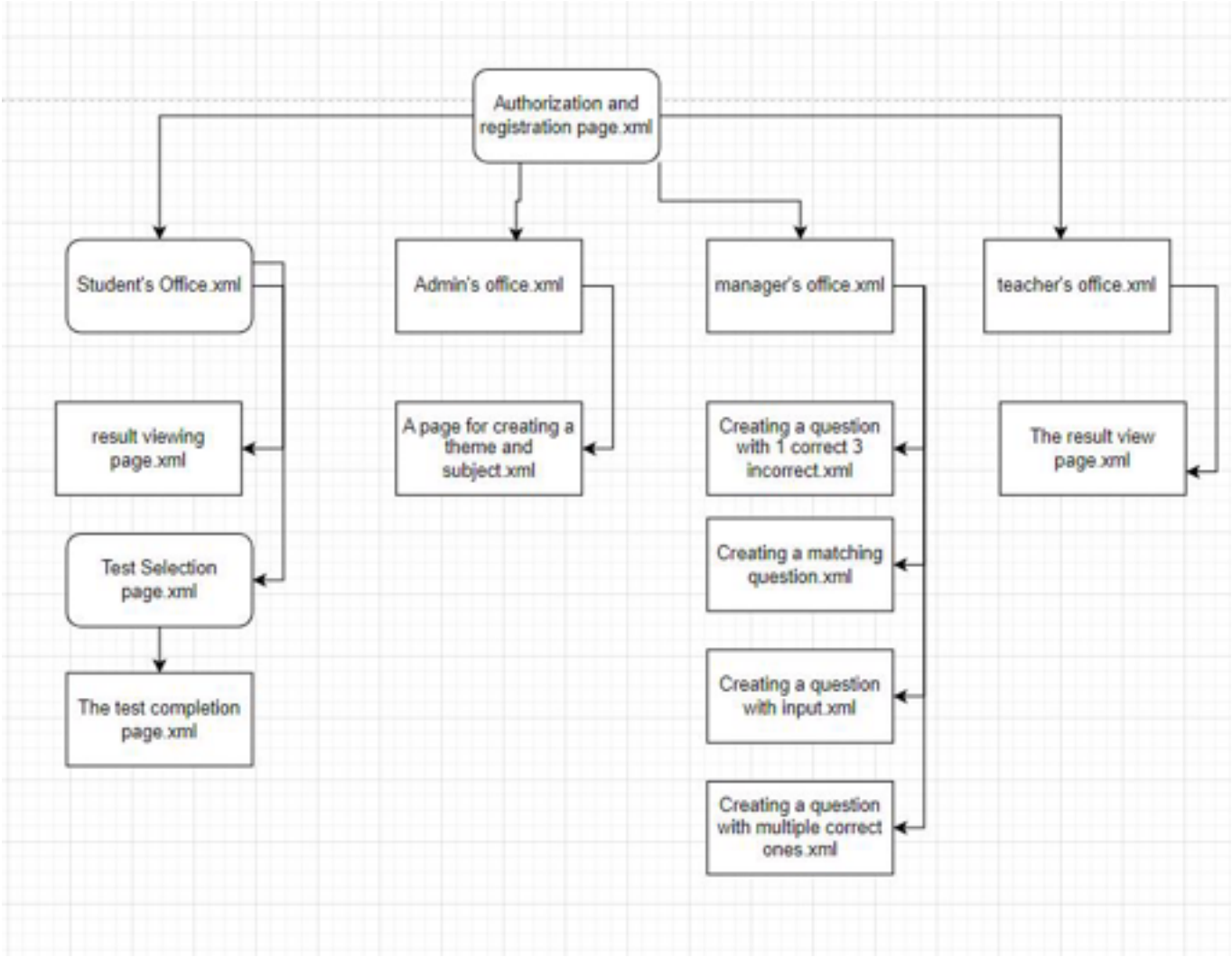

Рисунок 4 - Структурная схема

7. ФУНКЦИОНАЛЬНАЯ СХЕМА

7.1. Функциональная схема программы

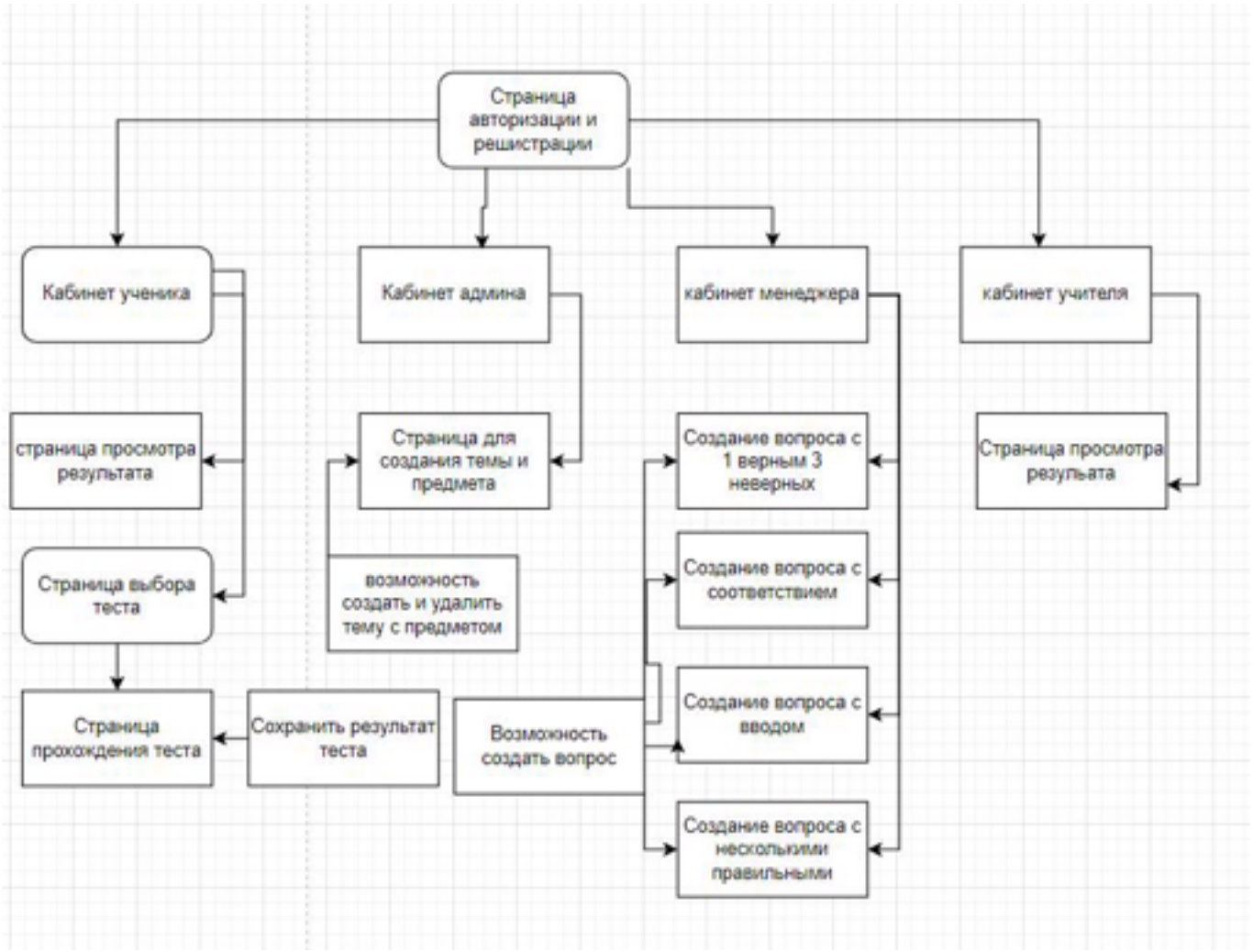

Рисунок 5 - Функциональная схема программы

8. БЮДЖЕТ И РЕСУРСЫ

8.1. Определение бюджета

Проведение практики ориентировочной экономической эффективности осуществляется без привлечения бюджетных средств в рамках курсового проекта по Технологии разработки программного обеспечения (КП 02.01). Анализ экономических преимуществ проводится в рамках курсового проекта по Технологии разработки программного обеспечения (КП 02.01) и не предполагает использования бюджетных средств.

9. ОЖИДАЕМЫЕ РЕЗУЛЬТАТЫ

В итоге завершения проекта, будет предоставлено полнофункциональное приложение, доступное для загрузки из USB-флешки с инструкцией по установке.

10. РИСКИ И МЕРЫ ПО ИХ СНИЖЕНИЮ

Ожидается, что одним из рисков проекта может быть задержка в разработке из-за технических сложностей. Для снижения этого риска предпримутся следующие меры: увеличение числа разработчиков и регулярное отслеживание прогресса.

11. ЗАКЛЮЧЕНИЕ

В заключении, разработка приложения "Развивашка" является оправданной и целесообразной задачей. Необходимость в эффективной обучаемости учеников и результатов тестов. Ценность заключается в следующим функцианале.

Поиск результатов учеников по email: Учителя могут легко найти результаты тестов своих учеников, введя их электронную почту.

Отображение результатов в удобном формате: Результаты тестов представлены в виде списка, что позволяет учителям быстро оценить прогресс каждого ученика.

Интеграция с Firestore: Результаты тестов хранятся в централизованном хранилище Firestore, что обеспечивает удобство доступа и синхронизации данных.

Так же есть возможность ученикам пройти тест с разными вариантами вопросов.

Так же есть возможность менеджерам добавлять новые вопросы разных типов

Удобный дизайн, простой интерфейс и надежные меры безопасности сделают приложение “Развивашка” ценным инструментом для улучшения процесса обучения и отслеживания прогресса учеников.

УТВЕРЖДЕН КП 02.01 П50-4-21 21 24

ПРИЛОЖЕНИЕ Г. Сценарий тестовых испытаний

Программа для развития детей

Сценарий тестов испытаний

УП 02.01 П50-4-21 23 24

Листов: 12

2024

АННОТАЦИЯ

В данном разделе представлены результаты тестов для «Развивашка». Результаты тестов включают проверку всех тест-кейсов который были описаны в предыдущей работе.

СОДЕРЖАНИЕ

1. СХЕМА ТЕСТИРОВАНИЯ

1.1. Схема тестирования

Тестирование разработанного программного решения проводилось по схеме тестирования, представленной на рисунке 1.

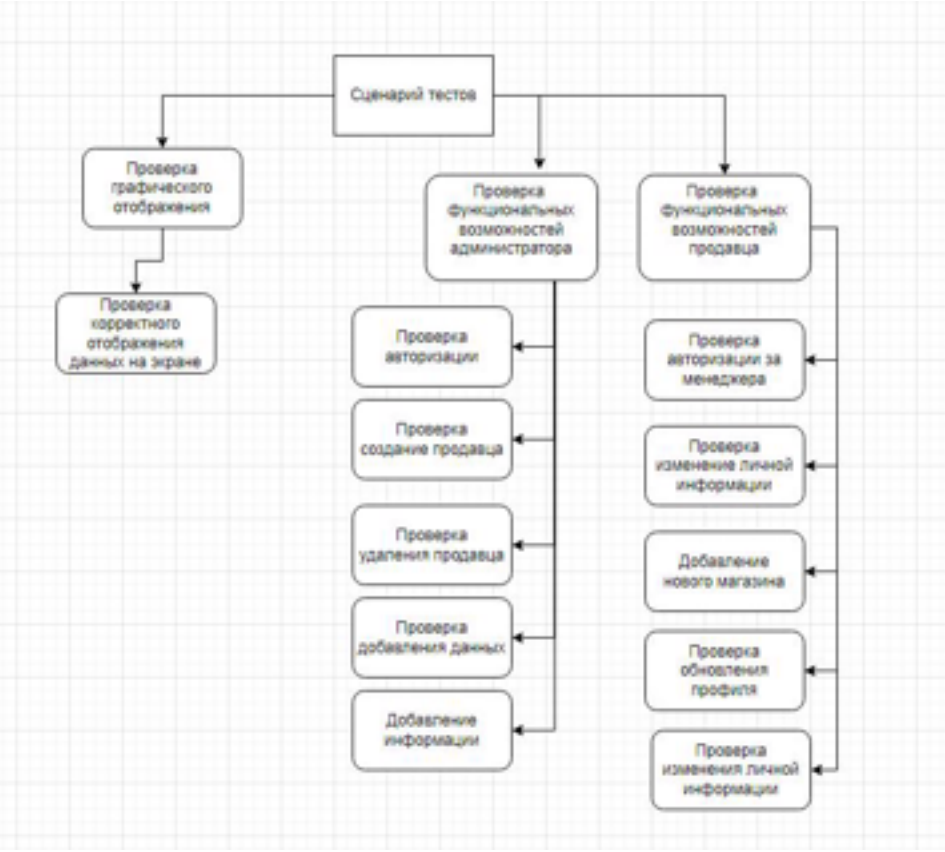

Рисунок 1 – Схема тестирования

2. ТЕСТИРОВАНИЕ ГРАФИЧЕСКОГО ИНТЕРФЕЙСА ПОЛЬЗОВАТЕЛЯ

2.1. Проверка графического интерфейса пользователя

Таблица 1. Проверка графического интерфейса пользователя

№ п/п	Предпосылки	Шаги тестирования	Данные тестирования	Ожидаемый результат	Фактический результат	Статус теста (пройден/ не пройден)
1	2	3	4	5	6	7
Проверка корректного отображения данных на экране						
1	Ни администратор, ни менеджер не должны быть авторизованы.	Открыть главное окно приложения	Логин – «test1@mail.ru», Пароль – «Testik.01», Роль - Ученик	Страница отображается кабинет ученика	Страница отображается кабинет ученика	Пройден
2	Администратор должен быть авторизован	Открыть окно создание Теста и предмет	Данные не предусмотрены.	Страница отображается корректно данные не должны	Страница отображается корректно данные не выходить за пределы полей	Пройден

				выходить за пределы полей		
3	Менеджер должен быть авторизован	Открыть страницу создание для вопроса	Выбрать предмет, темы, Ввести данные в поля answer1, answer2,answer3,correct answer, qvwstion_text	Страница отображается корректно данные не должны выходить за пределы полей, вопросы должны быть созданы успешно	Страница отображается корректно, тест был создан	Пройден
4	За ученика выбрать тест и нажать на него.	Открыть страницу для прохождения теста, нажать на кнопку следующий вопрос	Выбрать один вариант из нескольких представленных	Должно вывести сообщение верный или неверный ответ и показ следующего вопроса	Вывод сообщение о правильности ответа, вывод следующего вопроса.	Пройден
5	За ученика выбрать посмотреть результаты тестов	Нажать на кнопку просмотр результатов		Должно вывести список пройденных тестов и результат оценки	Вывод пройденных тестов и их результат	Пройден

6	Проверка на регистрацию	Нажать на кнопку регистрация	Ввести логин, пароль и почта, выбрать роль ученик	Должно вывести сообщение, что пользователь успешно зарегистрирован	Вывод сообщения что пользователь успешно зарегистрирован	Пройден
7	Проверка на авторизацию пользователя	Нажать на кнопку авторизация пользователя	Ввести логин пароль и почту.	В зависимости от роли переход на соответствующие кабинеты, для пользователя	Переход на страницу для пользователя	Пройден
8	Окно авторизации	Запустить приложение	Данные не предусмотрены.	Доступно неавторизованному пользователю	Доступно неавторизованному пользователю	Тест пройден
9	Окно личного кабинета администратора	Авторизация за ролью администратор	Данные не предусмотрены.	Доступно авторизованному пользователю с	Доступно авторизованному пользователю с ролью	Тест пройден

				ролью администратор	администратор	
10	Окно личного кабинета менеджер	Авторизация за ролью менеджер	Данные не предусмотрены.	Доступно авторизованному пользователю с ролью менеджер	Доступно авторизованному пользователю с ролью менеджер	Тест пройден
11	Окно личного кабинета учитель	Авторизация за ролью учитель	Данные не предусмотрены.	Доступно авторизованному пользователю с ролью учитель	Доступно авторизованному пользователю с ролью учитель	Тест пройден
12	Окно личного кабинета ученик	Авторизация за ролью ученик	Данные не предусмотрены.	Доступно авторизованному пользователю с ролью ученик	Доступно авторизованному пользователю с ролью ученик	Тест пройден
13	Окно вывода данных о результате теста	Авторизация за ролью ученик, переход на страницу просмотреть результаты тестов	Данные не предусмотрены.	Вывод информации о пройденных тестов	Вывод информации о пройденных тестов	Тест пройден

14	Создание вопроса типа 1 верный, 3 неверных	Авторизоваться за менеджера, зайти на страницу создания вопроса 1 верный, 3 неверных	Данные не предусмотрены.	Страница отображается корректно данные не должны выходить за пределы полей	Страница отображается корректно данные не выходить за пределы полей	Пройден
15	Создание вопроса типа соответствие	Авторизоваться за менеджера, зайти на страницу создания соответствия	Данные не предусмотрены.	Страница отображается корректно данные не должны выходить за пределы полей	Страница отображается корректно данные не выходить за пределы полей	Пройден
16	Никто не авторизован	Открыть главное окно приложения	Данные не предусмотрены.	Страница отображает форму авторизации Страница отображает форму авторизации	Страница отображает форму авторизации Страница отображает форму авторизации	Пройден
17	Авторизоваться за учителя	Зайти на страницу где выводятся результаты	Ввести логин ученика	Отображает список пройденных тестов	Отображает список пройденных тестов	Пройден

		тестов учеников				
18	Авторизоваться за учителя	Зайти на страницу где выводятся результаты тестов учеников	Данные не предусмотрены.	Корректно отображается интерфейс	Корректно отображается интерфейс	Пройден
19	Авторизоваться за ученика	Зайти на страницу тесты	Данные не предусмотрены.	Корректно отображаются данные и тесты	Корректно отображаются данные и тесты	Пройден
20	Авторизоваться за ученика	Зайти на страницу тестов, выбрать другой предмет	Данные не предусмотрены.	Должны поменяться темы и сам список вопросов	Должны поменяться темы и сам список вопросов	Пройден
21	Авторизоваться за ученика	Зайти на страницу тестов, выбрать один тест из списка, нажать на него	Данные не предусмотрены.	Вывод списка ответов и самого вопроса. Данные должны выводится корректно	Вывод списка ответов и самого вопроса. Данные должны выводится корректно	Пройден

22	Авторизоваться за ученика	Зайти на страницу тестов, выбрать один тест из списка, нажать на него, нажать на кнопку “Следующий вопрос”	Данные не предусмотрены.	Смена вопроса на другой вопрос, отображение сообщение о верном ответе или неверном ответе	Смена вопроса на другой вопрос, отображение сообщение о верном ответе или неверном ответе	Пройден
23	Авторизоваться за ученика	Зайти на страницу тестов, выбрать один тест из списка, нажать на него, нажать на кнопку “Следующий вопрос” и так пока не выведется сообщение	Данные не предусмотрены.	Вывод сообщения что тест закончен, возможность сохранить результат теста или нет	Вывод сообщения что тест закончен, возможность сохранить результат теста или нет	Пройден
24	Авторизоваться за ученика	Зайти на страницу тестов, выбрать	Данные не предусмотрены.	Выведется сообщение что тест сохранён, выведется	Выведется сообщение что тест сохранён, выведется	Пройден

		один тест из списка, нажать на него, нажать на кнопку “Следующий вопрос” и так пока не выведется сообщение, нажать на кнопку “сохранить тест”		оценка и результат в процентах	оценка и результат в процентах	
25			Данные не предусмотрены.			Пройден

Приложение Д. Руководство пользователя

АННОТАЦИЯ

В данном программном документе приведено руководство пользователя для приложения Android studio java для развития детей и отслеживанию прогресса.

В разделе «Условия выполнения программы» описаны рекомендуемая и минимальная конфигурация устройства для использования данной программы. Также описано какие программные средства необходимы для пользования данной программой.

В разделе «Выполнение программы» находятся подразделы:

1. Действия для запуска программы – описывает действия для корректного запуска программы.

2. Выполнение программы с описанием функций – описывает возможности программы и результат действий.

СОДЕРЖАНИЕ

1. УСЛОВИЯ ИСПОЛЬЗОВАНИЯ ПРОГРАММЫ

В таблице 1 представлены максимальные или рекомендуемые и минимальные технические средства для использования приложения на мобильном устройстве.

Таблица 1. Максимальные или рекомендуемые и минимальные требования

№	Тип оборудования	Наименование оборудования
1	2	3
Максимальные или рекомендуемые требования		
1	Android API Level	34 (Android 13)
2	OpenAI Play Services	Требуется
3	Версия Java	8
4	Операционная система	Android
Минимальные требования		
1	Android API Level	27 (Android 8.1 Oreo)
2	Версия Java	8
3	OpenAI Play Services	Требуется
4	Операционная система	Android

В Таблице 2 представлены программные средства для использования программы.

Таблица 2. Программные средства

№	Тип средства	Название средства	Назначение
1	2	3	4
1	Операционная система	Android	Организация взаимодействия программ и пользователя

2. ВЫПОЛНЕНИЕ ПРОГРАММЫ

2.1. Действия для запуска программы

Для открытия приложение включить телефон и запустить приложение.

2.2. Выполнение программы с описанием функций

2.2.1. Выполнение программы «Неавторизированный пользователь»

2.2.4 Авторизация

После запуска приложения сразу открывается окно авторизации-регистрации, куда надо написать логин, почиу и сам пароль(Рисунок 1).

На данной странице необходимо ввести логин, email пользователя и пароль. Появится сообщение об успешной регистрации (Рисунок 1).

Рисунок 1 Страница регистрации и авторизации

После авторизация открытие следующего окна будет зависеть от роли пользователя, который он занимает.

2.2.2. Выполнение программы «Администратор»

2.2.4 Окно админа

После того как пользователь ввёл данные, то он если администратор перейдёт на такую страницу (Рисунок 3).

Рисунок 2 страница администратора

В этом окне есть возможность перейти на страницу для создания и удаления предметов и темы.

2.2.4 Страница добавления и удаления предмета и темы

После того как пользователь(администратор) нажал на кнопку блюда его перебросит на данную страницу (Рисунок 4).

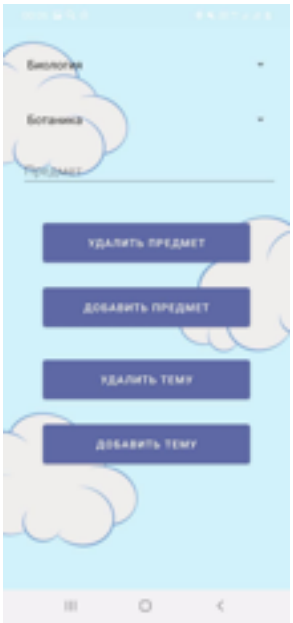

Рисунок 3 Страница добавления и удаления предмета и темы

В этом окне есть возможность добавить или удалить предмет или темы, для этого сначала надо ввести название предмета или выбрать предмет, для удаления темы.

2.2.3. Выполнение программы «Ученик»

После того как пользователь авторизовался, если он имеет роль ученик, то он попадает на страницу ученик (Рисунок 6), где он может выбрать посмотреть результаты тестов или перейти на страницу с тестами.

Рисунок 4 окно ученика

Если пользователь выбрал пройти тест открывается страница с выбором тестов, тут он может выбрать предмет и тему для теста (Рисунок 5).

Рисунок 5 страница для выбора теста

Так же со страницы ученика можно просмотреть результаты тестов (Рисунок 6).

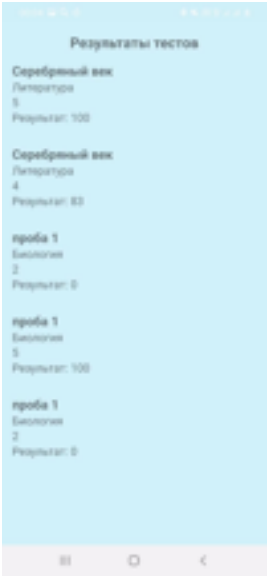

Рисунок 6 просмотр результатов теста

2.2.4. Выполнение программы «Учитель»

После того как пользователь авторизовался, если он имеет роль учитель, то он попадает на страницу учитель (Рисунок 7), где он может выбрать для просмотров результаты тестов учеников.

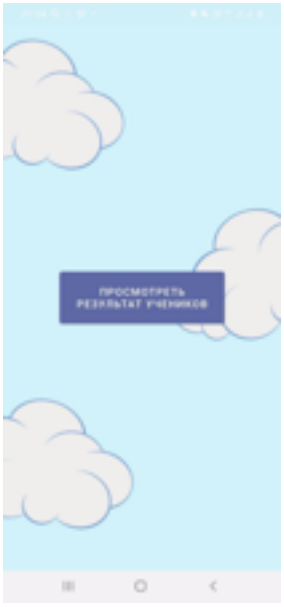

Рисунок 7 личная страница учителя

После перехода на страницу просмотра результата учеников нужно ввести email и нажать на кнопку “Поиск” (Рисунок 8).

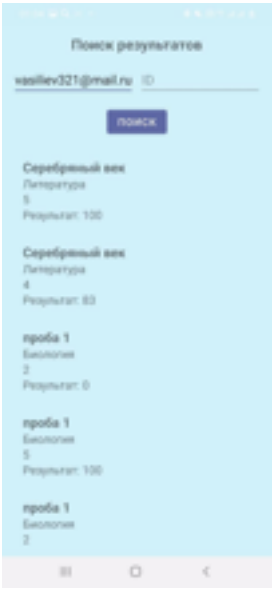

Рисунок 8 поиск по почте

2.2.4. Выполнение программы «Менеджер»

После того как пользователь авторизовался, если он имеет роль менеджер, то он попадает на страницу менеджера (Рисунок 9), где он может перейти на создание вопросов (разных типов) и на создания тестов.

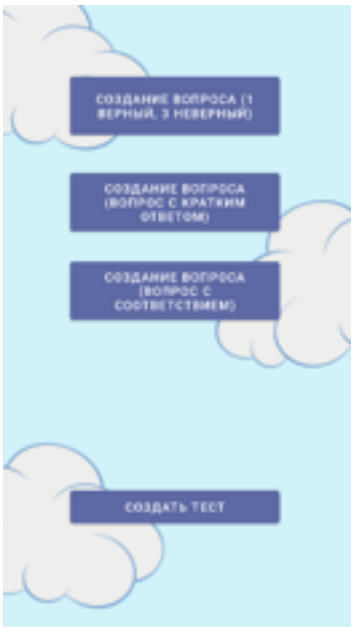

Рисунок 9 страница менеджера

Если менеджер нажимает на 1 кнопку(сверху), то он попадёт на страницу для создания вопроса. Сюда надо ввести данные и выбрать тему и предмет и нажать на кнопку “создать вопрос” (Рисунок 10).

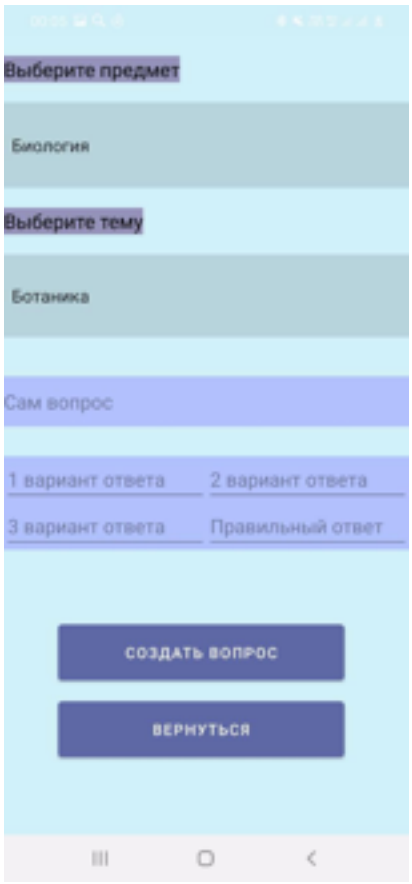

Рисунок 10 создание 1 типа вопроса

Если менеджер выбирает создать вопрос с вводимым ответом, то открывается окно где нужно ввести сам вопрос и правильный ответ (Рисунок 11).

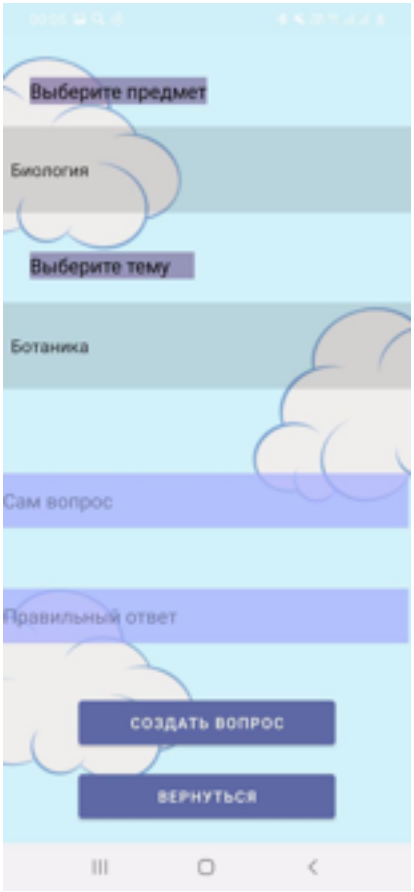

Рисунок 11 создание вопроса 2 типа

Если менеджер выбирает создать вопрос с соответствие от открывается новое окно (Рисунок 12). Здесь пользователь выбирает тему и предмет, задаёт вопрос и пишет правильные ответы, например, город и достопримечательности относящиеся к этому городу.

Рисунок 12 создание 3 типа вопроса

Если пользователь выберет создать тест, то открывается новый вопрос (Рисунок 13). Здесь пользователь выбирает тему и предмет пишет название теста и выбирает вопросы для этого теста, если такой тест есть то он пишет что такой уже тест есть хотите добавить новый вопрос, если пользователь отвечает да, то к этому тесту добавляется вопрос.

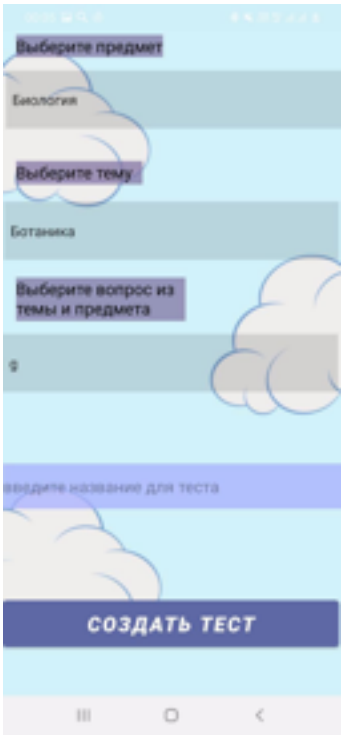

Рисунок 13 создание теста

УТВЕРЖДЕН
ТЗ КП 02.01 П50-4-21 21-24-ЛУ

Приложения для развития детей

Макет приложения

МП КП 02.01 П50-4-21 21-24

Листов 10

2024

АННОТАЦИЯ

В данном документе представлен макет приложения (МП) на создание программы под названием "Развивашка” МП разработано с учетом стандартов ГОСТ 19.106-78 и ГОСТ 19.104-78.

В данном разделе представлены рекомендации подготовки макета для настольного приложения, представляющего общий дизайн и структуру продукта в соответствии с требованиями проекта. Макет создавался с использованием инструмента Figma и соответствует требованиям, выдвигаемым к интерфейсу приложения.

1. Окна авторизации и регистрации будут окнами входа в программу. Сначала пользователю представлена регистрация и авторизация. Если у него есть аккаунт, то для этого создана кнопка для авторизации. Если не вводит данные нажимает регистрация.

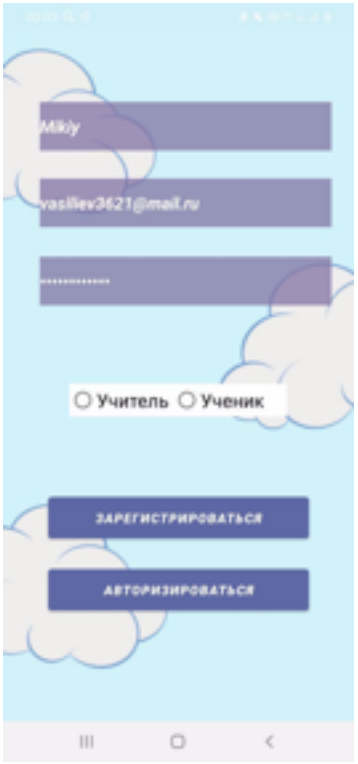

Рисунок 1 – окно регистрации и авторизации

Рисунок 2 – страница ученика

Рисунок 3 – прохождение тестов

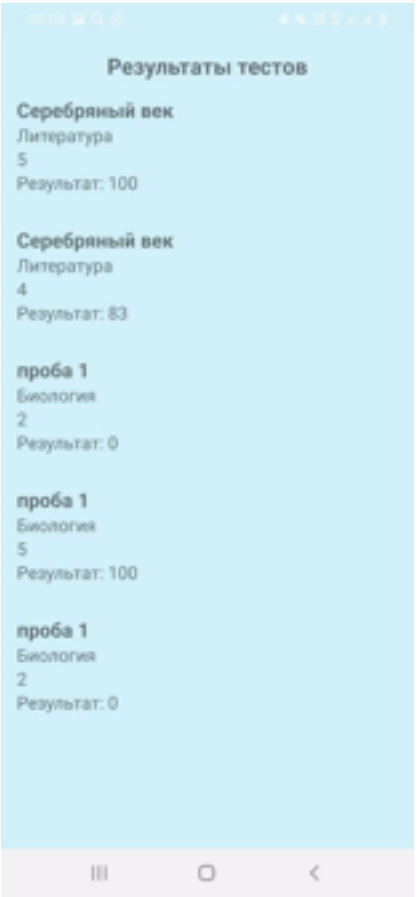

Рисунок 4 – просмотр результатов тестов

2. Если пользователь входит в систему за менеджера, то ему доступны

следующие окна, которые представлены ниже.

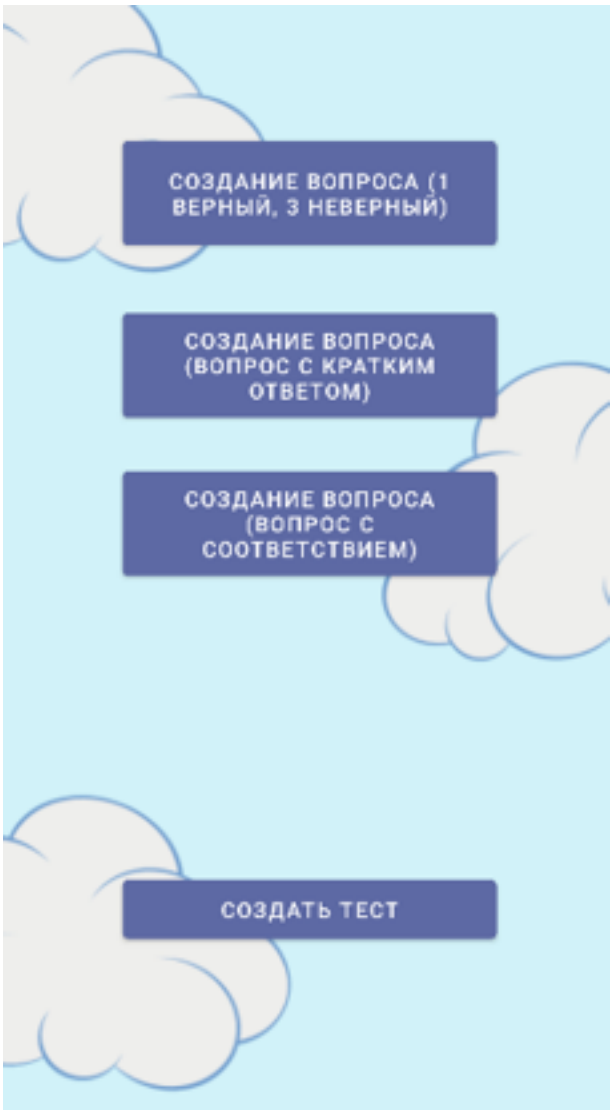

Рисунок 5 – главная страница

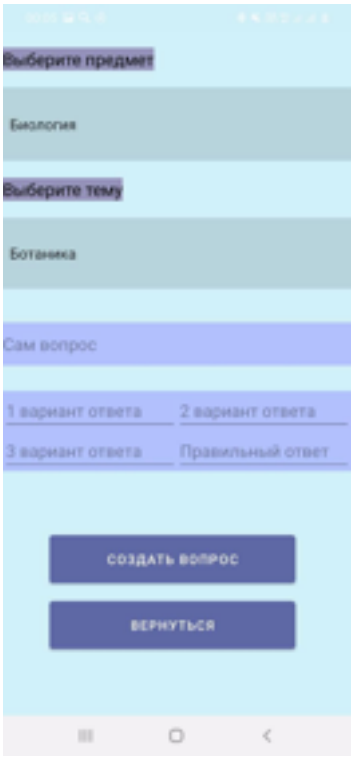

Рисунок 6 – создание вопроса с 1 правильным 3 неправильными

Рисунок 7 – создание вопроса с вводом ответа

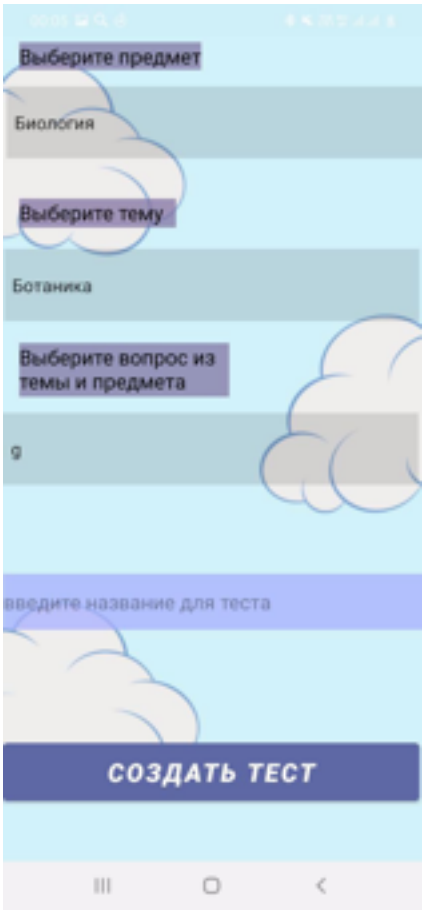

Рисунок 8 создание тестов

Если пользователь входит в систему за админа, то ему доступны следующие окна, которые представлены ниже.

Рисунок 9 – главная страница

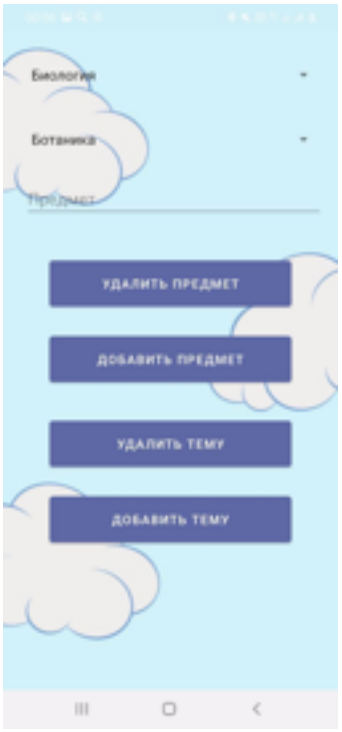

Рисунок 10 – Создание и удаление предмета и теста

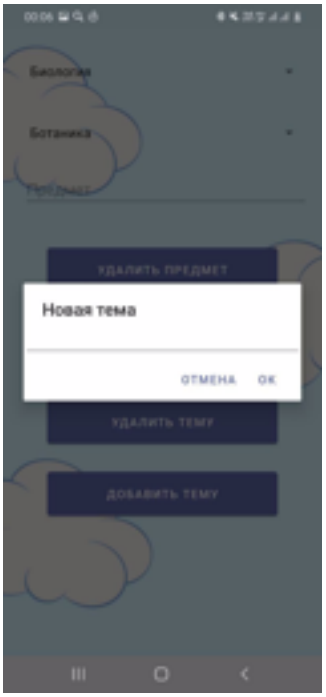

Рисунок 11 создание темы

Если пользователь входит в систему за учителя, то ему доступны следующие окна, которые представлены ниже.

Рисунок 12 – главная страница

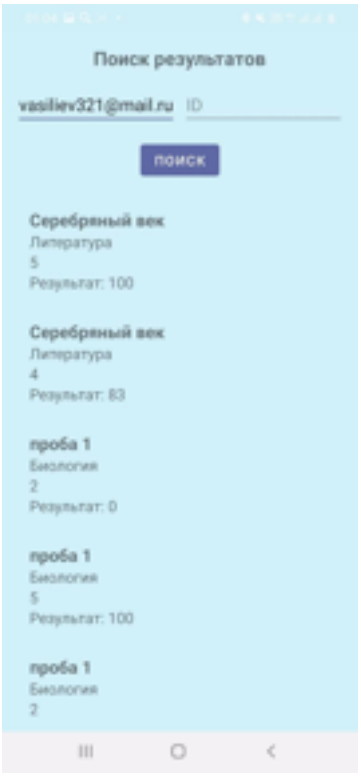

Рисунок 13 – Просмотр результат детей

ПРИЛОЖЕНИЕ Ж. КОД ПРОГРАММЫ

АННОТАЦИЯ

В данном разделе представлен код программы «Развивашка», который содержит 22 модуль для образования логики работы задуманной на этапе проектирования.

СОДЕРЖАНИЕ

1. ТЕКСТ ПРОГРАММЫ

1.1. Наименование программы

Наименование «Развивашка».

1.2 Область применения

Программа предназначена для равзыития детей путём прохождения различных тестов.

1.3 Модули

Модуль	Описание	Количество строк кода	Размер в (Кбайтах)
1	2	3	4
Admin_cabinet	Личная страница для администратора	25	1
Admin_dobavlenie_dannix	Страница для добавления и удаления предмета и темы	259	12
Change_the_test	Выбор теста для ученика	204	9
Dobavlenie	Первоначальная страница для добавления темы и предмета, позже заменена страницей Admin_cabinet	80	4
MainActivity	Страница для авторизации и регистрации пользователей	262	15
Menedger_cabinet	Личная страница менеджера	55	2
Meneger_create_a_tests	Страница для менеджер что бы можно было создать тест	333	17
Meneger_qvestion_complence	Страница для добавления вопроса с соответствием	175	9

prohogdenie_Test	Страница для ученика для прохождения теста	369	21
Quvestion_meneger	Страница для создания вопроса с 1 правильным и 3 неправильными ответами	71	9
Qvestion_vvod	Страница для создания вопроса с рукописным вводом	149	7
RecyclerItemClickListener	Обработка вывода списка	57	2
Result_ycheniki	Страница для отображения рузультатов тестов для учеников	136	5
Result_ychitel	Страница для отображения рузультатов тестов учеников для учителя	73	3
ychenik_cabinet	Личная страница для ученика	39	2
ycitel_cabinet	Личная страница для учителя	24	1

1.4 Код модуля Admin_cabinet

```
package com.example.razvivaska_plahova;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;

public class Admin_cabinet extends AppCompatActivity {

    private Button Per;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_admin_cabinet);
```

```
        Per = findViewById(R.id.Perehod);
        Per.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                startActivity(new Intent(Admin_cabinet.this, Admin_dobavlenie_dannix.class));
            }
        });
    }
}}
```

1.6 Код модуля Admin_dobavlenie_dannix

```
package com.example.razvivaska_plahova;

import static android.content.ContentValues.TAG;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;

import android.annotation.SuppressLint;
import android.content.DialogInterface;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Spinner;
import android.widget.TextView;
import android.widget.Toast;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.OnFailureListener;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.firestore.DocumentReference;
import com.google.firebase.firestore.DocumentSnapshot;
import com.google.firebase.firestore.FieldValue;
import com.google.firebase.firestore.FirebaseFirestore;
import com.google.firebase.firestore.QueryDocumentSnapshot;
import com.google.firebase.firestore.QuerySnapshot;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

public class Admin_dobavlenie_dannix extends AppCompatActivity {

    private Spinner spinnerSubject;
```

```
    private Spinner spinnerTheme;
    private EditText editTextQuestionTitle;
    private EditText editTextQuestionText;
    private EditText editTextAnswer1;
    private EditText editTextAnswer2;
    private EditText editTextAnswer3;
    private EditText editTextAnswer4;
    private EditText editTextSubjectName;
    private EditText editTextThemeName;
    private Button buttonCreateQuestion;
    private Button
buttonAddSubject,button_updateSubject,button_updateTheme,button_deleteSubject,button_delet
eTheme;
    private Button buttonAddTheme;
    private ArrayAdapter<Subject> adapter;
    private FirebaseFirestore db;
    private ArrayList<Subject> subjects;
    private ArrayList<String> themes;

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

        db = FirebaseFirestore.getInstance();

        spinnerSubject = findViewById(R.id.spinner_subject);
        spinnerTheme = findViewById(R.id.spinner_theme);
        editTextSubjectName = findViewById(R.id.editTextSubjectName);
        buttonAddSubject = findViewById(R.id.button_add_subject);
        buttonAddTheme = findViewById(R.id.button_add_theme);
        button_deleteSubject = findViewById(R.id.button_deleteSubject);
        button_deleteTheme  = findViewById(R.id.button_deleteTheme);
        subjects = new ArrayList<>();
        themes = new ArrayList<>();

        adapter = new ArrayAdapter<Subject>(this, android.R.layout.simple_spinner_item,
subjects) {
            @Override
            public View getView(int position, View convertView, ViewGroup parent) {
                View view = super.getView(position, convertView, parent);
                TextView textView = (TextView) view.findViewById(android.R.id.text1);
                textView.setText(((Subject) getItem(position)).getName());
                return view;
            }
        };
        spinnerSubject.setAdapter(adapter);
        buttonAddSubject.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                addSubject();
            }
```

```
        });
        button_deleteSubject.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                deleteSubject();
            }
        });
        button_deleteTheme.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                deleteTheme();
            }
        });
        buttonAddTheme.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                addTheme();
            }
        });

        loadSubjects(); // Call loadSubjects() at the end
        spinnerSubject.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
            @Override
            public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {
                Subject selectedSubject = (Subject) spinnerSubject.getSelectedItem();
                themes.clear();
                List<String> themesList = selectedSubject.getThemes();
                if (themesList != null && !themesList.isEmpty()) {
                    themes.addAll(themesList);
                    ArrayAdapter<String> adapter = new
ArrayAdapter<>(Admin_dobavlenie_dannix.this, android.R.layout.simple_spinner_item,
themes);

adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
                    spinnerTheme.setAdapter(adapter);
                } else {
                    // Handle the case when there are no themes
                    spinnerTheme.setAdapter(null); // or some other default adapter
                }
            }

            @Override
            public void onNothingSelected(AdapterView<?> parent) {
                // Do nothing
            }
        });


    }
    private void loadSubjects() {
        db.collection("subjects")
                .get()
```

```
            .addOnCompleteListener(new OnCompleteListener<QuerySnapshot>() {
                @Override
                public void onComplete(@NonNull Task<QuerySnapshot> task) {
                    if (task.isSuccessful()) {
                        subjects.clear(); // Clear the list before adding new data
                        for (QueryDocumentSnapshot document : task.getResult()) {
                            Subject subject = document.toObject(Subject.class);
                            subjects.add(subject);
                        }
                        adapter.notifyDataSetChanged(); // Notify the adapter of data changes
                    } else {
                        Log.d(TAG, "Error getting subjects: ", task.getException());
                    }
                }
            });
    }
    private void addSubject() {
        String subjectName = editTextSubjectName.getText().toString();
        Subject subject = new Subject(subjectName);
        db.collection("subjects").document(subjectName).set(new HashMap<String, String>() {{
            put("name", subjectName);
        }});
        subjects.add(subject);
        adapter.notifyDataSetChanged(); // Now this should work
        spinnerSubject.setAdapter(adapter);
    }
    private void deleteSubject() {
        String subjectName = editTextSubjectName.getText().toString();

        db.collection("subjects").document(subjectName).delete()
                .addOnSuccessListener(new OnSuccessListener<Void>() {
                    @Override
                    public void onSuccess(Void aVoid) {
                        // Обновляем список subjects после удаления документа
                        // Пример, предполагая, что subjects - это ArrayList
                        subjects.remove(subjectName);

                        Toast.makeText(Admin_dobavlenie_dannix.this, "Предмет успешно удалён",
Toast.LENGTH_SHORT).show();
                        adapter.notifyDataSetChanged();
                    }
                })
                .addOnFailureListener(new OnFailureListener() {
                    @Override
                    public void onFailure(@NonNull Exception e) {
                        Toast.makeText(Admin_dobavlenie_dannix.this, "Ошибка при удаление
предмета: " + e.getMessage(), Toast.LENGTH_SHORT).show();
                    }
                });
    }
    private void addTheme() {
        // Создаем всплывающее окно с полем ввода для имени темы
```

```
        AlertDialog.Builder builder = new AlertDialog.Builder(Admin_dobavlenie_dannix.this);
        builder.setTitle("Новая тема");

        // Создаем поле ввода для имени темы
        final EditText input = new EditText(Admin_dobavlenie_dannix.this);
        builder.setView(input);

        // Устанавливаем кнопки "ОК" и "Отмена"
        builder.setPositiveButton("ОК", new DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int id) {
                String themeName = input.getText().toString();
                Subject selectedSubject = (Subject) spinnerSubject.getSelectedItem();

                // Проверяем, существует ли тема в списке тем предмета
                if (selectedSubject.getThemes().contains(themeName)) {
                    // Если тема уже есть, выводим сообщение об ошибке
                    Toast.makeText(Admin_dobavlenie_dannix.this, "Тема уже существует!",
Toast.LENGTH_SHORT).show();
                } else {
                    // Если темы нет, добавляем ее в список тем предмета
                    selectedSubject.addTheme(themeName);
                    // Обновляем список тем в Firestore
                    db.collection("subjects").document(selectedSubject.getName()).update("themes",
FieldValue.arrayUnion(themeName));
                    // Обновляем список тем в адаптере
                    themes.clear();
                    themes.addAll(selectedSubject.getThemes());
                    ArrayAdapter<String> adapter = new
ArrayAdapter<>(Admin_dobavlenie_dannix.this, android.R.layout.simple_spinner_item,
themes);

adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
                    spinnerTheme.setAdapter(adapter);
                }
            }
        });
        builder.setNegativeButton("Отмена", new DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int id) {
                dialog.cancel();
            }
        });

        // Создаем и показываем диалоговое окно
        AlertDialog dialog = builder.create();
        dialog.show();
    }
    private void deleteTheme() {
        Subject selectedSubject = (Subject) spinnerSubject.getSelectedItem();
        String themeName = (String) spinnerTheme.getSelectedItem();

        List<String> themes = selectedSubject.getThemes();
```

```
        // Проверяем, что тема действительно существует в списке
        if (themes.contains(themeName)) {
            themes.remove(themeName); // Удаляем тему по имени
            // Обновляем данные
            try {
                db.collection("subjects").document(selectedSubject.getName()).update("themes",
themes)
                        .addOnSuccessListener(new OnSuccessListener<Void>() {
                            @Override
                            public void onSuccess(Void aVoid) {
                                Toast.makeText(Admin_dobavlenie_dannix.this, "Тема успешно удалена",
Toast.LENGTH_SHORT).show();
                                adapter.notifyDataSetChanged();
                            }
                        });
            } catch (Exception e) {
                Toast.makeText(Admin_dobavlenie_dannix.this, "Ошибка удалении темы: " +
e.getMessage(), Toast.LENGTH_SHORT).show();
            }
        } else {
            Toast.makeText(Admin_dobavlenie_dannix.this, "Тема не найдена",
Toast.LENGTH_SHORT).show();
        }
    }
}
```

1.7 Код модуля Change_the_test

```
package com.example.razvivaska_plahova;

import static android.content.ContentValues.TAG;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;

import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import android.text.Editable;
import android.text.TextWatcher;
import android.util.Log;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.EditText;
import android.widget.Spinner;
import android.widget.TextView;
import android.widget.Toast;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.firestore.DocumentSnapshot;
import com.google.firebase.firestore.FirebaseFirestore;
import com.google.firebase.firestore.QueryDocumentSnapshot;
```

```
import com.google.firebase.firestore.QuerySnapshot;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

public class Change_the_test extends AppCompatActivity {

    private ArrayAdapter<Subject> subjectAdapter;
    private FirebaseFirestore db;
    private ArrayList<Subject> subjects;
    private ArrayList<String> themes;
    private Spinner spinnerSubject;
    private Spinner spinnerTheme;
    private RecyclerView testRecyclerView;
    private TestAdapter testAdapter;
    private List<Test> testList = new ArrayList<>();
    private List<Test> allTests = new ArrayList<>();

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_change_the_test);
        spinnerSubject = findViewById(R.id.subject_spinner);
        spinnerTheme = findViewById(R.id.theme_spinner);
        testRecyclerView = findViewById(R.id.test_recyclerview);

        subjects = new ArrayList<>();
        themes = new ArrayList<>();

        // Инициализируйте db
        db = FirebaseFirestore.getInstance();

        // Настройка RecyclerView
        testRecyclerView.setLayoutManager(new LinearLayoutManager(this));
        testAdapter = new TestAdapter(testList);
        testRecyclerView.setAdapter(testAdapter);

        subjectAdapter = new ArrayAdapter<Subject>(this, android.R.layout.simple_spinner_item, subjects) {
            @Override
            public View getView(int position, View convertView, ViewGroup parent) {
                View view = super.getView(position, convertView, parent);
                TextView textView = (TextView) view.findViewById(android.R.id.text1);
                textView.setText(((Subject) getItem(position)).getName());
                return view;
            }
        };
        spinnerSubject.setAdapter(subjectAdapter);

        // Загружаем все тесты из Firestore
        loadTests();
        // Загружаем список предметов
        loadSubjects();

        // Настройка Spinner'ов
        spinnerSubject.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
            @Override
            public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {
                Subject selectedSubject = (Subject) spinnerSubject.getSelectedItem();
                themes.clear();
                List<String> themesList = selectedSubject.getThemes();
                if (themesList != null && !themesList.isEmpty()) {
```

```
                themes.addAll(themesList);
                ArrayAdapter<String> adapter = new ArrayAdapter<>(Change_the_test.this,
android.R.layout.simple_spinner_item, themes);
                adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
                spinnerTheme.setAdapter(adapter);
            } else {
                // Handle the case when there are no themes
                spinnerTheme.setAdapter(null); // or some other default adapter
            }
        }

        @Override
        public void onNothingSelected(AdapterView<?> parent) {
            // Do nothing
        }
    });

    spinnerTheme.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
        @Override
        public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {
            // Фильтруем тесты по выбранному предмету и теме
            filterTestsBySubjectAndTheme();
        }

        @Override
        public void onNothingSelected(AdapterView<?> parent) {
            // Do nothing
        }
    });

    // Обработчик нажатия на элемент в RecyclerView
    testRecyclerView.addOnItemTouchListener(
            new RecyclerItemClickListener(this, testRecyclerView, new
RecyclerItemClickListener.OnItemClickListener() {
                @Override
                public void onItemClick(View view, int position) {
                    // Переход на новую активность при нажатии на карточку
                    Intent intent = new Intent(Change_the_test.this, prohogdenie_Test.class);

                    // Передача данных о тесте в новую активность
                    intent.putExtra("testName", testList.get(position).getname());
                    intent.putExtra("subject", testList.get(position).getSubject());
                    intent.putExtra("theme", testList.get(position).getTheme());

                    startActivity(intent);
                }

                @Override
                public void onLongItemClick(View view, int position) {
                    // Ничего не делаем
                }
            })
    );
}

private void loadSubjects() {
    db.collection("subjects")
            .get()
            .addOnCompleteListener(new OnCompleteListener<QuerySnapshot>() {
                @Override
                public void onComplete(@NonNull Task<QuerySnapshot> task) {
                    if (task.isSuccessful()) {
                        subjects.clear();
```

```
                    for (QueryDocumentSnapshot document : task.getResult()) {
                        Subject subject = document.toObject(Subject.class);
                        subjects.add(subject);
                    }
                    subjectAdapter.notifyDataSetChanged();
                } else {
                    Log.d(TAG, "Error getting subjects: ", task.getException());
                }
            }
        });
    }

    private void loadTests() {
        db.collection("tests")
                .get()
                .addOnCompleteListener(new OnCompleteListener<QuerySnapshot>() {
                    @Override
                    public void onComplete(@NonNull Task<QuerySnapshot> task) {
                        if (task.isSuccessful()) {
                            allTests.clear();
                            for (DocumentSnapshot document : task.getResult().getDocuments()) {
                                Test test = document.toObject(Test.class);
                                allTests.add(test);
                            }
                            // Изначально отображаем все тесты
                            testList.addAll(allTests);
                            testAdapter.notifyDataSetChanged();
                        } else {
                            Log.w(TAG, "Error getting documents", task.getException());
                            Toast.makeText(Change_the_test.this, "Ошибка загрузки тестов",
Toast.LENGTH_SHORT).show();
                        }
                    }
                });
    }

    private void filterTestsBySubjectAndTheme() {
        String selectedSubject = spinnerSubject.getSelectedItem().toString();
        String selectedTheme = spinnerTheme.getSelectedItem().toString();

        testList.clear();
        for (Test test : allTests) {
            if (test.getSubject().equals(selectedSubject) &&
                    test.getTheme().equals(selectedTheme)) {
                testList.add(test);
            }
        }
        testAdapter.notifyDataSetChanged();
    }

}
```

1.8 Код модуля Dobavlenie

```
package com.example.razvivaska_plahova;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;

import android.annotation.SuppressLint;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
```

```
import android.widget.Button;
import android.widget.EditText;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.OnFailureListener;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.firestore.CollectionReference;
import com.google.firebase.firestore.DocumentReference;
import com.google.firebase.firestore.DocumentSnapshot;
import com.google.firebase.firestore.FirebaseFirestore;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

public class Dobavlenie extends AppCompatActivity {
    FirebaseFirestore db = FirebaseFirestore.getInstance();
    EditText subjectEditText;
    Button saveButton;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_dobavlenie);
        Button addButton = findViewById(R.id.button);
        EditText userInputEditText = findViewById(R.id.textView);
        DocumentReference documentRef =
FirebaseFirestore.getInstance().collection("Subjects").document("Subjects");

        addButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String userInputData = userInputEditText.getText().toString();

                // Get the current list of data from the document
                documentRef.get().addOnCompleteListener(new OnCompleteListener<DocumentSnapshot>() {
                    @Override
                    public void onComplete(@NonNull Task<DocumentSnapshot> task) {
                        if (task.isSuccessful()) {
                            DocumentSnapshot document = task.getResult();
                            if (document.exists()) {
                                // Get the current list of data from the document
                                List<String> dataList = (List<String>) document.get("dataList");

                                // Add the new data to the list
                                dataList.add(userInputData);

                                // Update the document with the new list
                                Map<String, Object> updateData = new HashMap<>();
                                updateData.put("dataList", dataList);
                                documentRef.update(updateData);
                            } else {
                                // If the document doesn't exist, create a new one with the initial list
                                List<String> dataList = new ArrayList<>();
                                dataList.add(userInputData);
                                Map<String, Object> data = new HashMap<>();
                                data.put("dataList", dataList);
                                documentRef.set(data);
                            }
                        } else {
                            Log.w("Firestore", "Error getting document", task.getException());
                        }
```

```
                }
            });
        }
    });
  }
}
```

1.10 Код модуля MainActivity

```
package com.example.razvivaska_plahova;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.OnFailureListener;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.firestore.DocumentReference;
import com.google.firebase.firestore.DocumentSnapshot;
import com.google.firebase.firestore.FirebaseFirestore;
import java.util.HashSet;
import java.util.Random;
import java.util.Set;
// RegisterActivity.java
import android.widget.RadioButton;
import android.widget.RadioGroup;
public class MainActivity extends AppCompatActivity {
    private EditText emailEditText, passwordEditText, loginEditText;
    private RadioButton teacherRadioButton, studentRadioButton;
    private RadioGroup roleRadioGroup;
    private Button registerButton,loginButton;
    private FirebaseAuth mAuth;
    private FirebaseFirestore db;

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

        mAuth = FirebaseAuth.getInstance();
        db = FirebaseFirestore.getInstance();

        emailEditText = findViewById(R.id.email_edit_text);
        passwordEditText = findViewById(R.id.password_edit_text);
        loginEditText = findViewById(R.id.login_edit_text);
        teacherRadioButton = findViewById(R.id.teacher_radio_button);
        studentRadioButton = findViewById(R.id.student_radio_button);
        roleRadioGroup = findViewById(R.id.role_radio_group);
        registerButton = findViewById(R.id.register_button);

        registerButton.setOnClickListener(new View.OnClickListener() {
            @Override
```

```
        public void onClick(View v) {
            String email = emailEditText.getText().toString();
            String password = passwordEditText.getText().toString();
            String login = loginEditText.getText().toString();
            String role;
            RandomIdGenerator generator = new RandomIdGenerator();
            String IDS = generator.generateRandomString(10);

            if (teacherRadioButton.isChecked()) {
                role = "Учитель";
            } else if (studentRadioButton.isChecked()) {
                role = "Ученик";
            }else if (studentRadioButton.isChecked()) {
                role = "Админ";
            }else if (studentRadioButton.isChecked()) {
                role = "Менеджер";
            }else {
                Toast.makeText(MainActivity.this, "Пожалуйста выберите роль", Toast.LENGTH_SHORT).show();
                return;
            }

            mAuth.createUserWithEmailAndPassword(email, password)
                    .addOnCompleteListener(new OnCompleteListener<AuthResult>() {
                        @Override
                        public void onComplete(@NonNull Task<AuthResult> task) {
                            if (task.isSuccessful()) {
                                // Get the current user's email
                                String email = emailEditText.getText().toString();

                                // Create a new user document in Firestore
                                FirebaseFirestore db = FirebaseFirestore.getInstance();
                                DocumentReference userRef = db.collection("users").document(email);

                                User user = new User(login, role, email, IDS);

                                // Set the user document with the User object
                                userRef.set(user)
                                        .addOnSuccessListener(new OnSuccessListener<Void>() {
                                            @Override
                                            public void onSuccess(Void aVoid) {
                                                Toast.makeText(MainActivity.this, "Регистрация прошла успешно",
Toast.LENGTH_SHORT).show();
                                            }
                                        })
                                        .addOnFailureListener(new OnFailureListener() {
                                            @Override
                                            public void onFailure(@NonNull Exception e) {
                                                Toast.makeText(MainActivity.this, "Упс, ошибка на стороне сервера, проверьте
подключены ли вы к интернету", Toast.LENGTH_SHORT).show();
                                            }
                                        });
                            } else {
                                Toast.makeText(MainActivity.this, "Упс, регистрация провалилась. Возможно аккаунт с
такой почтой уже есть", Toast.LENGTH_SHORT).show();
                            }
                        }
                    });

        }
    });
    loginButton = findViewById(R.id.login_button);

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

```
        @Override
        public void onClick(View v) {
            String email = emailEditText.getText().toString();
            String password = passwordEditText.getText().toString();
            final String login = loginEditText.getText().toString();

            // Проверка на пустоту логина
            if (login.isEmpty()) {
                Toast.makeText(MainActivity.this, "Введите логин", Toast.LENGTH_SHORT).show();
                return;
            }

            // Проверка на пустоту email и password
            if (email.isEmpty() || password.isEmpty()) {
                Toast.makeText(MainActivity.this, "Введите email и пароль", Toast.LENGTH_SHORT).show();
                return;
            }

            mAuth.signInWithEmailAndPassword(email, password)
                    .addOnCompleteListener(new OnCompleteListener<AuthResult>() {
                        @Override
                        public void onComplete(@NonNull Task<AuthResult> task) {
                            if (task.isSuccessful()) {
                                // Проверка роли пользователя и логина
                                db.collection("users").document(email)
                                        .get()
                                        .addOnCompleteListener(new OnCompleteListener<DocumentSnapshot>() {
                                            @Override
                                            public void onComplete(@NonNull Task<DocumentSnapshot> task) {
                                                if (task.isSuccessful()) {
                                                    DocumentSnapshot document = task.getResult();
                                                    String role = document.getString("role");
                                                    String userLogin = document.getString("login"); // Получение логина из
документа
                                                    Intent intent = null;
                                                    if (role != null && userLogin != null) {
                                                        if (role.equals("Ученик") && userLogin.equals(login)) { // Сравнение логина
                                                            // Открыть страницу 1 для учеников
                                                            Toast.makeText(MainActivity.this, "Добро пожаловать, ученик!",
Toast.LENGTH_SHORT).show();
                                                            intent = new Intent(MainActivity.this, ychenik_cabinet.class);
                                                            //startActivity(new Intent(MainActivity.this, ychenik_cabinet.class));
                                                        } else if (role.equals("Учитель") && userLogin.equals(login)) { // Сравнение
логина
                                                            // Открыть страницу 2 для учителей
                                                            Toast.makeText(MainActivity.this, "Добро пожаловать, учитель!",
Toast.LENGTH_SHORT).show();
                                                            intent = new Intent(MainActivity.this, ycitel_cabinet.class);
                                                            // startActivity(new Intent(MainActivity.this, ycitel_cabinet.class));
                                                        } else if (role.equals("Менеджер") && userLogin.equals(login)) { //
Сравнение логина
                                                            // Открыть страницу 2 для менеджеров
                                                            Toast.makeText(MainActivity.this, "Добро пожаловать, менеджер!",
Toast.LENGTH_SHORT).show();
                                                            intent = new Intent(MainActivity.this, Menedger_cabinet.class);
                                                            /// startActivity(new Intent(MainActivity.this, Menedger_cabinet.class));
                                                        } else if (role.equals("Админ") && userLogin.equals(login)) { // Сравнение
логина
                                                            // Открыть страницу 2 для админов
                                                            Toast.makeText(MainActivity.this, "Добро пожаловать, админ!",
Toast.LENGTH_SHORT).show();
                                                            intent = new Intent(MainActivity.this, Admin_cabinet.class);
                                                            //startActivity(new Intent(MainActivity.this, Admin_cabinet.class));
```

```
                                        }if (intent != null) {
                                            // Добавляем данные в Intent перед открытием нового окна
                                            intent.putExtra("email", email);
                                            intent.putExtra("login", login);
                                            startActivity(intent);
                                        } else {
                                            Toast.makeText(MainActivity.this, "Неверный логин или пароль",
Toast.LENGTH_SHORT).show();
                                        }
                                    } else {
                                        Toast.makeText(MainActivity.this, "Ошибка получения данных
пользователя", Toast.LENGTH_SHORT).show();
                                    }

                                } else {
                                    Toast.makeText(MainActivity.this, "Ошибка получения данных
пользователя", Toast.LENGTH_SHORT).show();
                                }
                            }
                        });
                    } else {
                        Toast.makeText(MainActivity.this, "Ошибка входа", Toast.LENGTH_SHORT).show();
                    }
                }
            });
        }
    });


}

public class RandomIdGenerator {
    private final Random RANDOM = new Random();


    private String generateRandomString(int length) {
        StringBuilder sb = new StringBuilder();
        for (int i = 0; i < length; i++) {
            int randomChar = RANDOM.nextInt(62);
            if (randomChar < 26) {
                sb.append((char) (randomChar + 'a'));
            } else if (randomChar < 52) {
                sb.append((char) (randomChar - 26 + 'A'));
            } else {
                sb.append((char) (randomChar - 52 + '0'));
            }
        }
        return sb.toString();
    }
}

public class User {
    private String login;
    private String role;
    private String email;
    private String IDS;
    public User(String login, String role, String email, String IDS) {
        this.login = login;
        this.role = role;
        this.email = email;
        this.IDS = IDS;
    }
```

```
    public String getLogin() {
        return login;
    }

    public void setLogin(String login) {
        this.login = login;
    }

    public String getRole() {
        return role;
    }

    public void setRole(String role) {
        this.role = role;
    }
    public String getemail() {
        return email;
    }

    public void setemail(String role) {
        this.email = email;
    }

    public String getID() {
        return IDS;
    }

    public void setID(String ID) {
        this.IDS = IDS;
    }
  }
}
```

1.11 Код модуля Menedger_cabinet

```
package com.example.razvivaska_plahova;


import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;

public class Menedger_cabinet extends AppCompatActivity {

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

      // Инициализируйте кнопки
      Button quvestion = findViewById(R.id.button_Quvestion);
```

```
        Button Test_button = findViewById(R.id.Test_button);
        Button q1 = findViewById(R.id.button_Quvestion_kratki);
        Button q3 = findViewById(R.id.button_Quvestion_3);

        quvestion.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                startActivity(new Intent(Menedger_cabinet.this, Quvestion_meneger.class));
            }
        });
        q1.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                startActivity(new Intent(Menedger_cabinet.this, Qvestion_vvod.class));
            }
        });
        q3.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                startActivity(new Intent(Menedger_cabinet.this, Meneger_qvestion_complence.class));
            }
        });
        Test_button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                startActivity(new Intent(Menedger_cabinet.this, Meneger_create_a_tests.class));
            }
        });
    }
}
```

1.12 Код модуля Meneger_create_a_tests

```
package com.example.razvivaska_plahova;

import static android.content.ContentValues.TAG;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Spinner;
import android.widget.TextView;
import android.widget.Toast;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.firestore.DocumentReference;
import com.google.firebase.firestore.DocumentSnapshot;
import com.google.firebase.firestore.FieldValue;
import com.google.firebase.firestore.FirebaseFirestore;
import com.google.firebase.firestore.QueryDocumentSnapshot;
```

```
import com.google.firebase.firestore.QuerySnapshot;

import java.util.ArrayList;
import java.util.List;
import java.util.ArrayList;
import java.util.List;

public class Meneger_create_a_tests extends AppCompatActivity {
    private Spinner spinnerSubjects;
    private Button buttonCreateTest;
    private Spinner spinnerThemes;
    private Spinner spinnerQuestions;

    private EditText name_test;
    private FirebaseFirestore db;
    private ArrayAdapter<Subject> adapter;
    private ArrayList<Subject> subjects;
    private ArrayList<String> themes;
    private ArrayList<String> qvestion;
    private ArrayList<Questions> questionsList; // Список вопросов для Spinner

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_meneger_create_atests);
        spinnerSubjects = findViewById(R.id.spinner_subjects);
        spinnerThemes = findViewById(R.id.spinner_themes);
        spinnerQuestions = findViewById(R.id.spinner_questions);
        name_test = findViewById(R.id.edit_text_question_title);
        buttonCreateTest = findViewById(R.id.button_create_test);
        subjects = new ArrayList<>();
        themes = new ArrayList<>();
        questionsList = new ArrayList<>(); // Инициализация списка вопросов

        adapter = new ArrayAdapter<Subject>(this, android.R.layout.simple_spinner_item, subjects) {
            @Override
            public View getView(int position, View convertView, ViewGroup parent) {
                View view = super.getView(position, convertView, parent);
                TextView textView = (TextView) view.findViewById(android.R.id.text1);
                textView.setText(((Subject) getItem(position)).getName());
                return view;
            }
        };
        spinnerSubjects.setAdapter(adapter);
        db = FirebaseFirestore.getInstance();

        buttonCreateTest.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                createTest();
            }
        });

        // Загрузка предметов
        loadSubjects();

        // Обработчик выбора предмета
        spinnerSubjects.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
            @Override
            public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {
                Subject selectedSubject = (Subject) parent.getItemAtPosition(position);
                loadThemes(selectedSubject);
            }
```

```
        @Override
        public void onNothingSelected(AdapterView<?> parent) {
            // Nothing selected
        }
    });

    // Обработчик выбора темы
    spinnerThemes.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
        @Override
        public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {
            String selectedTheme = (String) parent.getItemAtPosition(position);
            loadQuestions(selectedTheme);
        }

        @Override
        public void onNothingSelected(AdapterView<?> parent) {
            // Nothing selected
        }
    });
}

// Метод для загрузки тем, принимающий выбранный предмет
private void loadThemes(Subject selectedSubject) {
    themes.clear(); // Очищаем список тем
    List<String> themesList = selectedSubject.getThemes();
    if (themesList != null && !themesList.isEmpty()) {
        themes.addAll(themesList);
        ArrayAdapter<String> adapter = new ArrayAdapter<>(Meneger_create_a_tests.this,
android.R.layout.simple_spinner_item, themes);
        adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
        spinnerThemes.setAdapter(adapter);
    } else {
        // Handle the case when there are no themes
        spinnerThemes.setAdapter(null); // or some other default adapter
    }
}

// Метод для загрузки вопросов
private void loadQuestions(String selectedTheme) {
    // Get the selected subject and theme
    String selectedSubject = null;
    if (spinnerSubjects.getSelectedItem() != null) {
        selectedSubject = spinnerSubjects.getSelectedItem().toString();
    }

    if (selectedSubject != null && selectedTheme != null) {
        // Query Firestore for questions based on selected subject and theme
        db.collection("Questions")
                .whereEqualTo("subject", selectedSubject)
                .whereEqualTo("theme", selectedTheme)
                .get()
                .addOnCompleteListener(new OnCompleteListener<QuerySnapshot>() {
                    @Override
                    public void onComplete(@NonNull Task<QuerySnapshot> task) {
                        if (task.isSuccessful()) {
                            questionsList.clear(); // Очищаем список вопросов
                            for (QueryDocumentSnapshot document : task.getResult()) {
                                Questions question = document.toObject(Questions.class);
                                questionsList.add(question); // Добавляем объект Questions
                            }

                            // Create and set the adapter for spinnerQuestions
```

```
                    ArrayAdapter<Questions> adapter = new ArrayAdapter<Questions>(getApplicationContext(),
android.R.layout.simple_spinner_item, questionsList) {
                        @Override
                        public View getView(int position, View convertView, ViewGroup parent) {
                            View view = super.getView(position, convertView, parent);
                            TextView textView = (TextView) view.findViewById(android.R.id.text1);
                            textView.setText(((Questions) getItem(position)).getQuestionText()); // Изменяем текст
                            return view;
                        }

                        // Optional: Customize the appearance of the dropdown view
                        @Override
                        public View getDropDownView(int position, View convertView, ViewGroup parent) {
                            View view = super.getDropDownView(position, convertView, parent);
                            TextView textView = (TextView) view.findViewById(android.R.id.text1);
                            textView.setText(((Questions) getItem(position)).getQuestionText()); // Изменяем текст
                            return view;
                        }
                    };
                    adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
                    spinnerQuestions.setAdapter(adapter);
                } else {
                    Log.w("Error", "Error loading questions", task.getException());
                    // Handle the error (e.g., display an error message to the user)
                }
            }
        });
    } else {
        Log.w("Error", "Subject or Theme is null");
        // Handle the situation where either subject or theme is null (e.g., display an error message)
    }
}

private void loadSubjects() {
    db.collection("subjects")
            .get()
            .addOnCompleteListener(new OnCompleteListener<QuerySnapshot>() {
                @Override
                public void onComplete(@NonNull Task<QuerySnapshot> task) {
                    if (task.isSuccessful()) {
                        subjects.clear(); // Clear the list before adding new data
                        for (QueryDocumentSnapshot document : task.getResult()) {
                            Subject subject = document.toObject(Subject.class);
                            subjects.add(subject);
                        }
                        adapter.notifyDataSetChanged(); // Notify the adapter of data changes
                    } else {
                        Log.d(TAG, "Error getting subjects: ", task.getException());
                    }
                }
            });
}

private void createTest() {
    // Get the selected subject, theme, and questions
    String selectedSubject = spinnerSubjects.getSelectedItem().toString();
    String selectedTheme = spinnerThemes.getSelectedItem().toString();
    Questions selectedQuestion = (Questions) spinnerQuestions.getSelectedItem();
    String name_ = name_test.getText().toString();
    String Names = name_;

    // Проверка на выбранные поля
    if (selectedSubject.isEmpty() || selectedTheme.isEmpty() || selectedQuestion == null || Names.isEmpty()) {
```

```
        Toast.makeText(Meneger_create_a_tests.this, "Выберите предмет, тему и вопрос!",
Toast.LENGTH_SHORT).show();
        return;
    }

    // Check if a test with the same name already exists, but only if the subject and theme are the same
    db.collection("tests")
            .whereEqualTo("name", Names)
            .whereEqualTo("subject", selectedSubject)
            .whereEqualTo("theme", selectedTheme)
            .get()
            .addOnCompleteListener(new OnCompleteListener<QuerySnapshot>() {
                @Override
                public void onComplete(@NonNull Task<QuerySnapshot> task) {
                    if (task.isSuccessful()) {
                        if (!task.getResult().isEmpty()) {
                            // Test with the same name exists, show a dialog
                            showConfirmationDialog(Names, selectedQuestion, selectedSubject, selectedTheme);
                        } else {
                            // Test with the same name does not exist, create a new test
                            createTestWithNewQuestion(Names, selectedQuestion, selectedSubject, selectedTheme);
                        }
                    } else {
                        Log.w("TestCreation", "Error checking for existing test", task.getException());
                        Toast.makeText(Meneger_create_a_tests.this, "Ошибка, повторите ещё раз",
Toast.LENGTH_SHORT).show();
                    }
                }
            });
  }
  private void showConfirmationDialog(String testName, Questions selectedQuestion, String selectedSubject,
String selectedTheme) {
    AlertDialog.Builder builder = new AlertDialog.Builder(Meneger_create_a_tests.this);
    builder.setTitle("Тест уже существует");
    builder.setMessage("Вы хотите добавить этот вопрос к тесту " + testName + "?");
    builder.setPositiveButton("Да", (dialog, which) -> {
        // Add the question to the existing test
        addQuestionToExistingTest(testName, selectedQuestion, selectedSubject, selectedTheme);
    });
    builder.setNegativeButton("Нет", (dialog, which) -> {
        // Do nothing, user canceled
    });
    builder.show();
  }

  // Create a new test with the selected question
  private void createTestWithNewQuestion(String testName, Questions selectedQuestion, String selectedSubject,
String selectedTheme) {
    DocumentReference testRef = db.collection("tests").document();
    Test test = new Test(testName, selectedSubject, selectedTheme, new ArrayList<>());
    test.getQuestions().add(selectedQuestion);

    testRef.set(test)
            .addOnSuccessListener(documentReference -> {
                Log.d("TestCreation", "Test created successfully");
                Toast.makeText(Meneger_create_a_tests.this, "Тест успешно создан",
Toast.LENGTH_SHORT).show();
            })
            .addOnFailureListener(e -> {
                Log.w("TestCreation", "Error creating test", e);
                Toast.makeText(Meneger_create_a_tests.this, "Ошибка при создания теста",
Toast.LENGTH_SHORT).show();
            });
```

```
    }

    // Add the selected question to an existing test
    private void addQuestionToExistingTest(String testName, Questions selectedQuestion, String selectedSubject,
String selectedTheme) {
        db.collection("tests")
                .whereEqualTo("name", testName)
                .get()
                .addOnCompleteListener(new OnCompleteListener<QuerySnapshot>() {
                    @Override
                    public void onComplete(@NonNull Task<QuerySnapshot> task) {
                        if (task.isSuccessful()) {
                            if (!task.getResult().isEmpty()) {
                                for (DocumentSnapshot document : task.getResult().getDocuments()) {
                                    DocumentReference testRef = document.getReference();
                                    testRef.update("questions", FieldValue.arrayUnion(selectedQuestion))
                                            .addOnSuccessListener(aVoid -> {
                                                Log.d("TestCreation", "Question added to existing test successfully");
                                                Toast.makeText(Meneger_create_a_tests.this, "Вопрос успешно добавлен",
Toast.LENGTH_SHORT).show();
                                            })
                                            .addOnFailureListener(e -> {
                                                Log.w("TestCreation", "Error adding question to existing test", e);
                                                Toast.makeText(Meneger_create_a_tests.this, "Ошибка при добавлении вопроса",
Toast.LENGTH_SHORT).show();
                                            });
                                }
                            }
                        } else {
                            Log.w("TestCreation", "Error updating test", task.getException());
                            Toast.makeText(Meneger_create_a_tests.this, "Ошибка обновления теста",
Toast.LENGTH_SHORT).show();
                        }
                    }
                });
    }
}
class Test {
    private  String name;
    private String subject;
    private String theme;
    private List<Questions> questions;
    public Test(String name,String subject, String theme, ArrayList<Questions> questions) {
        this.name = name;
        this.subject = subject;
        this.theme = theme;
        this.questions = questions;
    }

    public String getSubject() {
        return subject;
    }
    public Test() {
    }
    public String getname() {
        return name;
    }
    public String getTheme() {
        return theme;
    }

    public List<Questions> getQuestions() {
        return questions;
```

```
    }

}
```

1.13 Код модуля Meneger_qvestion_complence

```
package com.example.razvivaska_plahova;

import static android.content.ContentValues.TAG;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Spinner;
import android.widget.TextView;
import android.widget.Toast;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.OnFailureListener;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.firestore.DocumentReference;
import com.google.firebase.firestore.FirebaseFirestore;
import com.google.firebase.firestore.QueryDocumentSnapshot;
import com.google.firebase.firestore.QuerySnapshot;

import java.util.ArrayList;
import java.util.List;

public class Meneger_qvestion_complence extends AppCompatActivity {
    private Spinner spinnerSubject;
    private Spinner spinnerTheme;
    private Button exit;
    private Button buttonCreateQuestion;
    private EditText editTextQuestionText;
    private ArrayAdapter<Subject> adapter;
    private FirebaseFirestore db;
    private ArrayList<Subject> subjects;
    private ArrayList<String> themes;
    private EditText editTextAnswer1;
    private EditText editTextAnswer2;
    private EditText editTextAnswer3;

    private EditText editTextAnswer4;
    private EditText editTextAnswer1_otvet;
    private EditText editTextAnswer2_otvet;
    private EditText editTextAnswer3_otvet;
    private EditText editTextAnswer4_otvet;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_meneger_qvestion_complence);
        db = FirebaseFirestore.getInstance();
        editTextAnswer1 = findViewById(R.id.edit_text_answer1);
```

```
        editTextAnswer2 = findViewById(R.id.edit_text_answer2);
        editTextAnswer3 = findViewById(R.id.edit_text_answer3);
        editTextAnswer4 = findViewById(R.id.edit_text_answer4);
        editTextAnswer1_otvet = findViewById(R.id.edit_text_answer1_otvet);
        editTextAnswer2_otvet = findViewById(R.id.edit_text_answer2_otvet);
        editTextAnswer3_otvet = findViewById(R.id.edit_text_answer3_otvet);
        editTextAnswer4_otvet = findViewById(R.id.edit_text_answer4_otvet);
        editTextQuestionText = findViewById(R.id.edit_text_question_text);
        spinnerSubject = findViewById(R.id.spinner_subject);
        spinnerTheme = findViewById(R.id.spinner_theme);
        exit = findViewById(R.id.button_exit);
        buttonCreateQuestion = findViewById(R.id.button_create_question);
        subjects = new ArrayList<>();
        themes = new ArrayList<>();

        adapter = new ArrayAdapter<Subject>(this, android.R.layout.simple_spinner_item, subjects) {
            @Override
            public View getView(int position, View convertView, ViewGroup parent) {
                View view = super.getView(position, convertView, parent);
                TextView textView = (TextView) view.findViewById(android.R.id.text1);
                textView.setText(((Subject) getItem(position)).getName());
                return view;
            }
        };
        spinnerSubject.setAdapter(adapter);
        exit.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                startActivity(new Intent(Meneger_qvestion_complence.this, Menedger_cabinet.class));
            }
        });
        buttonCreateQuestion.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
              createQuestion();

            }
        });
//
        loadSubjects(); // Call loadSubjects() at the end
        spinnerSubject.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
            @Override
            public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {
                Subject selectedSubject = (Subject) spinnerSubject.getSelectedItem();
                themes.clear();
                List<String> themesList = selectedSubject.getThemes();
                if (themesList != null && !themesList.isEmpty()) {
                    themes.addAll(themesList);
                    ArrayAdapter<String> adapter = new ArrayAdapter<>(Meneger_qvestion_complence.this,
android.R.layout.simple_spinner_item, themes);
                    adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
                    spinnerTheme.setAdapter(adapter);
                } else {
                    // Handle the case when there are no themes
                    spinnerTheme.setAdapter(null); // or some other default adapter
                }
            }

            @Override
            public void onNothingSelected(AdapterView<?> parent) {
                // Do nothing
            }
        });
```

```
    }

    private void loadSubjects() {
        db.collection("subjects")
                .get()
                .addOnCompleteListener(new OnCompleteListener<QuerySnapshot>() {
                    @Override
                    public void onComplete(@NonNull Task<QuerySnapshot> task) {
                        if (task.isSuccessful()) {
                            subjects.clear(); // Clear the list before adding new data
                            for (QueryDocumentSnapshot document : task.getResult()) {
                                Subject subject = document.toObject(Subject.class);
                                subjects.add(subject);
                            }
                            adapter.notifyDataSetChanged(); // Notify the adapter of data changes
                        } else {
                            Log.d(TAG, "Error getting subjects: ", task.getException());
                        }
                    }
                });
    }
    private void createQuestion() {
        Subject selectedSubject = (Subject) spinnerSubject.getSelectedItem();
        String theme = spinnerTheme.getSelectedItem().toString();
        String questionText = editTextQuestionText.getText().toString();
        String correctAnswer = editTextAnswer4.getText().toString();
        String incorrectAnswer1 = editTextAnswer1.getText().toString();
        String incorrectAnswer2 = editTextAnswer2.getText().toString();
        String incorrectAnswer3 = editTextAnswer3.getText().toString();
        String correctAnswer_otvet = editTextAnswer4_otvet.getText().toString();
        String incorrectAnswer1_otvet = editTextAnswer1_otvet.getText().toString();
        String incorrectAnswer2_otvet = editTextAnswer2_otvet.getText().toString();
        String incorrectAnswer3_otvet = editTextAnswer3_otvet.getText().toString();

        String answer1 = incorrectAnswer1 + " " + incorrectAnswer1_otvet;
        String answer2 = incorrectAnswer2 + " " + incorrectAnswer2_otvet;
        String answer3 = incorrectAnswer3 + " " + incorrectAnswer3_otvet;
        String CorrectAncwer = correctAnswer + " " + correctAnswer_otvet;
        String Qvestion_type = "Вопрос с соответствием";
        Questions question = new Questions(selectedSubject.getName(), theme, questionText, answer1, answer2,
answer3, CorrectAncwer,Qvestion_type);
        Toast.makeText(Meneger_qvestion_complence.this, "Question", Toast.LENGTH_SHORT).show();

        db.collection("Questions").add(question).addOnSuccessListener(new
OnSuccessListener<DocumentReference>() {
            @Override
            public void onSuccess(DocumentReference documentReference) {
                Toast.makeText(Meneger_qvestion_complence.this, "Вопрос успешно создан",
Toast.LENGTH_SHORT).show();
            }
        }).addOnFailureListener(new OnFailureListener() {
            @Override
            public void onFailure(@NonNull Exception e) {
                Toast.makeText(Meneger_qvestion_complence.this, "Ошибка создания вопроса",
Toast.LENGTH_SHORT).show();
            }
        });
    }
} }
```

1.14 Код модуля prohogdenie_Test

```
package com.example.razvivaska_plahova;
import androidx.annotation.NonNull;
import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;
import androidx.constraintlayout.widget.ConstraintLayout;
import androidx.constraintlayout.widget.ConstraintSet;

import android.content.DialogInterface;
import android.os.Bundle;
import android.content.Intent;
import android.text.Editable;
import android.text.TextWatcher;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.LinearLayout;
import android.widget.RadioButton;
import android.widget.RadioGroup;
import android.widget.TextView;
import android.widget.Toast;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.firestore.DocumentReference;
import com.google.firebase.firestore.DocumentSnapshot;
import com.google.firebase.firestore.FirebaseFirestore;
import com.google.firebase.firestore.QuerySnapshot;

import java.util.ArrayList;
import java.util.Collections;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Random;

public class prohogdenie_Test extends AppCompatActivity {
    private TextView testNameTextView, subjectTextView, themeTextView;
    private TextView questionTextView;
    private RadioGroup answerRadioGroup;
    ConstraintLayout sootvetstvieConstraintLayout;
    ConstraintLayout vvod;
    private Button nextButton;
    private FirebaseAuth mAuth;
    private FirebaseFirestore db = FirebaseFirestore.getInstance();
    private List<Questions> questions = new ArrayList<>(); // Список вопросов
    private int currentQuestionIndex = 0; // Индекс текущего вопроса
    private int score = 0;
    private boolean isAnswerChecked = false; // Флаг, чтобы проверять, был ли выбран ответ
    private TextView questionText;
    private LinearLayout answerContainer;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_prohogdenie_test);
            mAuth = FirebaseAuth.getInstance();
            testNameTextView = findViewById(R.id.testNameTextView);
            subjectTextView = findViewById(R.id.subjectTextView);
            themeTextView = findViewById(R.id.themeTextView);
            questionTextView = findViewById(R.id.questionTextView);
```

```
        answerRadioGroup = findViewById(R.id.answerRadioGroup);
        nextButton = findViewById(R.id.nextButton);
        sootvetstvieConstraintLayout = findViewById(R.id.Sootvetstvie);
        vvod = findViewById(R.id.vvods);
        // Получение данных о тесте из Intent
        Intent intent = getIntent();
        String testName = intent.getStringExtra("testName");
        String subject = intent.getStringExtra("subject");
        String theme = intent.getStringExtra("theme");

        testNameTextView.setText("Название теста: " + testName);
        subjectTextView.setText("Предмет: " + subject);
        themeTextView.setText("Тема: " + theme);

        // Загрузка теста из Firestore
        loadTest(testName, subject, theme);

        // Настройка слушателя для кнопки "Следующий"
        nextButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

//                if (isAnswerChecked) {
//                    checkAnswer();
//                    isAnswerChecked = false;
//                } else {
//                    Toast.makeText(prohogdenie_Test.this, "Выберите ответ", Toast.LENGTH_SHORT).show();
//                }
                checkAnswer();
            }
        });

        // Настройка слушателя для RadioGroup
        answerRadioGroup.setOnCheckedChangeListener(new RadioGroup.OnCheckedChangeListener() {
            @Override
            public void onCheckedChanged(RadioGroup group, int checkedId) {
                // Если выбран какой-то RadioButton
                if (checkedId != -1) {
                    isAnswerChecked = true;
                } else {
                    isAnswerChecked = false;
                }
            }
        });
    }

    private void loadTest(String testName, String subject, String theme) {
        db.collection("tests")
                .get()
                .addOnCompleteListener(new OnCompleteListener<QuerySnapshot>() {
                    @Override
                    public void onComplete(@NonNull Task<QuerySnapshot> task) {
                        if (task.isSuccessful()) {
                            for (DocumentSnapshot document : task.getResult().getDocuments()) {
                                if (document.getString("name").equals(testName) &&
                                        document.getString("subject").equals(subject) &&
                                        document.getString("theme").equals(theme)) {
                                    // Найден нужный тест
                                    List<Map<String, Object>> questionsFromFirestore = (List<Map<String, Object>>)
document.get("questions");

                                    if (questionsFromFirestore != null && !questionsFromFirestore.isEmpty()) {
                                        // Преобразование вопросов из Firestore в объекты Questions
```

```
                        for (Map<String, Object> questionData : questionsFromFirestore) {
                            String questionText = (String) questionData.get("questionText");
                            String answer1 = (String) questionData.get("answer1");
                            String answer2 = (String) questionData.get("answer2");
                            String answer3 = (String) questionData.get("answer3");
                            String correctAnswer = (String) questionData.get("correctAnswer"); //  или  (String)
questionData.get("правильный_ответ");
                            String type = (String) questionData.get("qvestion_type"); //  или  (String)
questionData.get("правильный_ответ");

                            questions.add(new Questions(questionText, answer1, answer2, answer3,
correctAnswer, type));
                        }

                        Collections.shuffle(questions, new Random());
                        // Отобразите первый вопрос
                        showNextQuestion();
                        return; // Выходим из цикла, так как тест найден
                    } else {
                        // Тест не содержит вопросов
                        Toast.makeText(prohogdenie_Test.this, "Тест не содержит вопросов",
Toast.LENGTH_SHORT).show();
                    }
                }
            }

            // Тест не найден
            Toast.makeText(prohogdenie_Test.this, "Тест не найден", Toast.LENGTH_SHORT).show();
        } else {
            // Ошибка при чтении данных
            Toast.makeText(prohogdenie_Test.this, "Ошибка загрузки теста",
Toast.LENGTH_SHORT).show();
        }
    }
});
}

private void showNextQuestion() {
    if (currentQuestionIndex < questions.size()) {
        // Получаем текущий вопрос
        Questions currentQuestion = questions.get(currentQuestionIndex);
        questionTextView.setText(currentQuestion.getQuestionText());
        if (currentQuestion.getqvestion_type().equals("1 правильный, 3 неправильных")) {
            sootvetstvieConstraintLayout.setVisibility(View.GONE);
            vvod.setVisibility(View.GONE);
            answerRadioGroup.setVisibility(View.VISIBLE); // Отображаем RadioGroup// Отобразите варианты
ответов
            RadioButton answerOption1 = findViewById(R.id.answerOption1);
            answerOption1.setText(currentQuestion.getAnswer1());
            RadioButton answerOption2 = findViewById(R.id.answerOption2);
            answerOption2.setText(currentQuestion.getAnswer2());
            RadioButton answerOption3 = findViewById(R.id.answerOption3);
            answerOption3.setText(currentQuestion.getAnswer3());
            RadioButton answerOption4 = findViewById(R.id.answerOption4);
            answerOption4.setText(currentQuestion.getCorrectAnswer());
        } else if (currentQuestion.getqvestion_type().equals("Вопрос с соответствием")) {
            answerRadioGroup.setVisibility(View.GONE); // Скрываем RadioGroup
            sootvetstvieConstraintLayout.setVisibility(View.VISIBLE);
            vvod.setVisibility(View.GONE);
            String[] answers = currentQuestion.getAnswer1().split(" ");
            String[] answers2 = currentQuestion.getAnswer2().split(" ");
            String[] answers3 = currentQuestion.getAnswer3().split(" ");
            String[] answers4 = currentQuestion.getCorrectAnswer().split(" ");
```

```
            // Устанавливаем текст в RadioButtons
            TextView answerOption1 = findViewById(R.id.textView16);
            answerOption1.setText(answers[0]);
            TextView answerOption1_1 = findViewById(R.id.text16);
            answerOption1_1.setText(answers[1]);
            TextView answerOption2 = findViewById(R.id.text7);
            answerOption2.setText(answers2[0]);
            TextView answerOption2_2 = findViewById(R.id.text3);
            answerOption2_2.setText(answers2[1]);
            TextView answerOption3 = findViewById(R.id.text5);
            answerOption3.setText(answers3[0]);
            TextView answerOption3_3 = findViewById(R.id.text1);
            answerOption3_3.setText(answers3[1]);
            TextView answerOption4 = findViewById(R.id.text6);
            answerOption4.setText(answers4[0]);
            TextView answerOption4_4 = findViewById(R.id.text4);
            answerOption4_4.setText(answers4[1]);
            EditText editTextOtvet = findViewById(R.id.text_otvet);
            editTextOtvet.addTextChangedListener(new TextWatcher() {
                @Override
                public void beforeTextChanged(CharSequence s, int start, int count, int after) {
                    // Не используется
                }

                @Override
                public void onTextChanged(CharSequence s, int start, int before, int count) {
                    // Не используется
                }

                @Override
                public void afterTextChanged(Editable s) {
                    // Проверяем ответ, когда пользователь вводит текст
                    checkAnswer();
                }
            });
            //answerOption4.setVisibility(View.VISIBLE); // Отображаем 4-й вариант ответа (текстовое поле)
            // ... реализуйте показ текстового поля для свободного ответа
        }else if (currentQuestion.getqvestion_type().equals("Вопрос с соответствием")) {
            answerRadioGroup.setVisibility(View.GONE);
            sootvetstvieConstraintLayout.setVisibility(View.GONE);
        }
        // Сбросьте выделенный RadioButton
        answerRadioGroup.clearCheck();

        currentQuestionIndex++;
    } else {
        // Тест завершен
        showResults();
    }
}

private void checkAnswer() {
    EditText textOtvet1 = findViewById(R.id.text_otvet);
    EditText textOtvet2 = findViewById(R.id.text_otvet1);
    EditText textOtvet3 = findViewById(R.id.text_otvet3);
    EditText textOtvet4 = findViewById(R.id.text_otvet4);
    Questions currentQuestion = questions.get(currentQuestionIndex - 1); // Исправлено: берем предыдущий
вопрос
    String userAnswer = findViewById(R.id.text_otvet).getContext().toString();
    // Проверяем тип вопроса
    if (currentQuestion.getqvestion_type().equals("Вопрос с соответствием")) {
        String userAnswer1 = textOtvet1.getText().toString().trim();
        String userAnswer2 = textOtvet2.getText().toString().trim();
```

```
            String userAnswer3 = textOtvet3.getText().toString().trim();
            String userAnswer4 = textOtvet4.getText().toString().trim();

            // Проверяем правильность введенных данных
            if (!userAnswer1.isEmpty() && !userAnswer2.isEmpty() && !userAnswer3.isEmpty() &&
!userAnswer4.isEmpty()) {
                // Проверяем, совпадают ли ответы пользователя с правильными ответами
                boolean isCorrect =
                        userAnswer1.equalsIgnoreCase(currentQuestion.getAnswer1().trim()) &&
                                userAnswer2.equalsIgnoreCase(currentQuestion.getAnswer2().trim()) &&
                                userAnswer3.equalsIgnoreCase(currentQuestion.getAnswer3().trim()) &&
                                userAnswer4.equalsIgnoreCase(currentQuestion.getCorrectAnswer().trim());

                // Выводим результат проверки
                if (isCorrect) {
                    score++;
                    Toast.makeText(this, "Правильно!", Toast.LENGTH_SHORT).show();
                } else {
                    Toast.makeText(this, "Неверно", Toast.LENGTH_SHORT).show();
                }
                showNextQuestion();
            } else {
                Toast.makeText(this, "Заполните все поля ответа.", Toast.LENGTH_SHORT).show();
            }
        } else if (currentQuestion.getqvestion_type().equals("1 правильный, 3 неправильных")) {
            int selectedId = answerRadioGroup.getCheckedRadioButtonId();
            if (selectedId != -1) {
                RadioButton selectedRadioButton = findViewById(selectedId);
                String selectedAnswer = selectedRadioButton.getText().toString();

                if (selectedAnswer.equalsIgnoreCase(currentQuestion.getCorrectAnswer())) {
                    score++;
                    Toast.makeText(this, "Правильно!", Toast.LENGTH_SHORT).show();
                } else {
                    Toast.makeText(this, "Неверно", Toast.LENGTH_SHORT).show();
                }
                showNextQuestion();
            } else {
                Toast.makeText(this, "Выберите ответ", Toast.LENGTH_SHORT).show();
            }
        } else if (currentQuestion.getqvestion_type().equals("Вопрос с вводимом ответом")) {
            EditText Otvet1_vvesti = findViewById(R.id.Otvet_vvesti);
            if (Otvet1_vvesti.getText().toString().trim().isEmpty()) {
                Toast.makeText(this, "Поле ответа не может быть пустым!", Toast.LENGTH_SHORT).show();
                return; // Выход из функции, если поле пустое
            }
            String userAnswers = Otvet1_vvesti.getText().toString().trim();
            if (userAnswers.equalsIgnoreCase(currentQuestion.getCorrectAnswer().trim())) {
                // Ответ верный
                score++;
                Toast.makeText(this, "Правильно!", Toast.LENGTH_SHORT).show();
            } else {
                // Ответ неверный
                Toast.makeText(this, "Неверно", Toast.LENGTH_SHORT).show();
            }
        }
    }


    private void showResults() {
        // Вычисление процента правильных ответов
        int percentage = (score * 100) / questions.size();
```

```
        // Отображение результатов
        Toast.makeText(this, "Тест завершен! Ваш результат: " + percentage + "%",
Toast.LENGTH_LONG).show();

        // Создание диалогового окна с вопросом о сохранении результата
        new AlertDialog.Builder(this)
                .setTitle("Сохранить результат?")
                .setMessage("Хотите сохранить результат теста?")
                .setPositiveButton(android.R.string.yes, new DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog, int which) {
                        // Сохраняем результат в Firestore
                        saveResult(percentage);
                    }
                })
                .setNegativeButton(android.R.string.no, null)
                .setIcon(android.R.drawable.ic_dialog_alert)
                .show();
    }

    private void saveResult(int percentage) {
        String email = mAuth.getCurrentUser().getEmail(); // Получаем почту пользователя
        Intent intent = getIntent();
        String subject = intent.getStringExtra("subject");
        String theme = intent.getStringExtra("theme");
        String testName = intent.getStringExtra("testName");
        // Создаем Map с данными для сохранения
        String grade = calculateGrade(percentage);

        // Создаем Map с данными для сохранения
        Map<String, Object> resultData = new HashMap<>();
        resultData.put("email", email);
        resultData.put("subject", subject);
        resultData.put("theme", theme);
        resultData.put("testName", testName);
        resultData.put("result", percentage);
        resultData.put("grade", grade);

        // Сохраняем данные в коллекцию "results" в Firestore
        db.collection("results")
                .add(resultData)
                .addOnCompleteListener(new OnCompleteListener<DocumentReference>() {
                    @Override
                    public void onComplete(@NonNull Task<DocumentReference> task) {
                        if (task.isSuccessful()) {
                            Toast.makeText(prohogdenie_Test.this, "Результат сохранен!",
Toast.LENGTH_SHORT).show();
                            startActivity(new Intent(prohogdenie_Test.this, ychenik_cabinet.class));
                        } else {
                            Toast.makeText(prohogdenie_Test.this, "Ошибка сохранения результата!",
Toast.LENGTH_SHORT).show();
                            startActivity(new Intent(prohogdenie_Test.this, ychenik_cabinet.class));
                        }
                    }
                });
    }
    private String calculateGrade(int percentage) {
        if (percentage < 70) {
            return "2";
        } else if (percentage >= 70 && percentage <= 80) {
            return "3";
        } else if (percentage >= 81 && percentage <= 90) {
            return "4";
        } else if (percentage >= 91 && percentage <= 100) {
```

```
            return "5";
        } else {
            return "Неизвестно"; // Добавьте обработку для некорректных значений
        }
    }
}}
```

1.15 Код модуля Quvestion_meneger

```
package com.example.razvivaska_plahova;

import static android.content.ContentValues.TAG;

import android.util.Log;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Spinner;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.TextView;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.OnFailureListener;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.firestore.DocumentReference;
import com.google.firebase.firestore.FirebaseFirestore;
import com.google.firebase.firestore.QueryDocumentSnapshot;
import com.google.firebase.firestore.QuerySnapshot;

import java.util.ArrayList;
import java.util.List;



public class Quvestion_meneger extends AppCompatActivity {
    private Spinner spinnerSubject;
    private Spinner spinnerTheme;
```

```
private EditText editTextQuestionTitle;
private EditText editTextQuestionText;
private EditText editTextAnswer1;
private EditText editTextAnswer2;
private EditText editTextAnswer3;
private EditText editTextAnswer4;
private EditText editTextSubjectName;
private EditText editTextThemeName;
private Button buttonCreateQuestion;
private Button buttonAddSubject;
private Button buttonAddTheme, exit;
private ArrayAdapter<Subject> adapter;
private FirebaseFirestore db;
private ArrayList<Subject> subjects;
private ArrayList<String> themes;

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

    db = FirebaseFirestore.getInstance();

    spinnerSubject = findViewById(R.id.spinner_subject);
    spinnerTheme = findViewById(R.id.spinner_theme);
    editTextQuestionText = findViewById(R.id.edit_text_question_text);
    editTextAnswer1 = findViewById(R.id.edit_text_answer1);
    editTextAnswer2 = findViewById(R.id.edit_text_answer2);
    editTextAnswer3 = findViewById(R.id.edit_text_answer3);
    editTextAnswer4 = findViewById(R.id.edit_text_answer4);
    exit = findViewById(R.id.button_exit);
    buttonCreateQuestion = findViewById(R.id.button_create_question);

    subjects = new ArrayList<>();
    themes = new ArrayList<>();

    adapter = new ArrayAdapter<Subject>(this, android.R.layout.simple_spinner_item, subjects) {
        @Override
        public View getView(int position, View convertView, ViewGroup parent) {
            View view = super.getView(position, convertView, parent);
            TextView textView = (TextView) view.findViewById(android.R.id.text1);
            textView.setText(((Subject) getItem(position)).getName());
```

```
            return view;
        }
    };
    spinnerSubject.setAdapter(adapter);
    exit.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            startActivity(new Intent(Quvestion_meneger.this, Menedger_cabinet.class));
        }
    });
    buttonCreateQuestion.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            createQuestion();

        }
    });

    loadSubjects(); // Call loadSubjects() at the end
    spinnerSubject.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
        @Override
        public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {
            Subject selectedSubject = (Subject) spinnerSubject.getSelectedItem();
            themes.clear();
            List<String> themesList = selectedSubject.getThemes();
            if (themesList != null && !themesList.isEmpty()) {
                themes.addAll(themesList);
                ArrayAdapter<String> adapter = new ArrayAdapter<>(Quvestion_meneger.this,
android.R.layout.simple_spinner_item, themes);
                adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
                spinnerTheme.setAdapter(adapter);
            } else {
                // Handle the case when there are no themes
                spinnerTheme.setAdapter(null); // or some other default adapter
            }
        }

        @Override
        public void onNothingSelected(AdapterView<?> parent) {
            // Do nothing
        }
    });
```

```
    }
    private void loadSubjects() {
        db.collection("subjects")
                .get()
                .addOnCompleteListener(new OnCompleteListener<QuerySnapshot>() {
                    @Override
                    public void onComplete(@NonNull Task<QuerySnapshot> task) {
                        if (task.isSuccessful()) {
                            subjects.clear(); // Clear the list before adding new data
                            for (QueryDocumentSnapshot document : task.getResult()) {
                                Subject subject = document.toObject(Subject.class);
                                subjects.add(subject);
                            }
                            adapter.notifyDataSetChanged(); // Notify the adapter of data changes
                        } else {
                            Log.d(TAG, "Error getting subjects: ", task.getException());
                        }
                    }
                });
    }
//    private void addSubject() {
//        String subjectName = editTextSubjectName.getText().toString();
//        Subject subject = new Subject(subjectName);
//        db.collection("subjects").document(subjectName).set(new HashMap<String, String>() {{
//            put("name", subjectName);
//        }});
//        subjects.add(subject);
//        adapter.notifyDataSetChanged(); // Now this should work
//        spinnerSubject.setAdapter(adapter);
//    }
//    private void addTheme() {
//        String themeName = editTextThemeName.getText().toString();
//        Subject selectedSubject = (Subject) spinnerSubject.getSelectedItem();
//        selectedSubject.addTheme(themeName);
//                                db.collection("subjects").document(selectedSubject.getName()).update("themes",
FieldValue.arrayUnion(themeName));
//        themes.clear();
//        themes.addAll(selectedSubject.getThemes());
//                        ArrayAdapter<String> adapter = new ArrayAdapter<>(Quvestion_meneger.this,
android.R.layout.simple_spinner_item, themes);
//        adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
```

```
//        spinnerTheme.setAdapter(adapter);
//    }
    private void createQuestion() {
        Subject selectedSubject = (Subject) spinnerSubject.getSelectedItem();
        String theme = spinnerTheme.getSelectedItem().toString();
        String questionText = editTextQuestionText.getText().toString();
        String correctAnswer = editTextAnswer4.getText().toString();
        String incorrectAnswer1 = editTextAnswer1.getText().toString();
        String incorrectAnswer2 = editTextAnswer2.getText().toString();
        String incorrectAnswer3 = editTextAnswer3.getText().toString();

        String answer1 = incorrectAnswer1;
        String answer2 = incorrectAnswer2;
        String answer3 = incorrectAnswer3;
        String CorrectAncwer = correctAnswer;
        String Qvestion_type = "1 правильный, 3 неправильных";
        Questions question = new Questions(selectedSubject.getName(), theme, questionText, answer1, answer2, answer3, CorrectAncwer,Qvestion_type);
        Toast.makeText(Quvestion_meneger.this, "Question", Toast.LENGTH_SHORT).show();

        db.collection("Questions").add(question).addOnSuccessListener(new OnSuccessListener<DocumentReference>() {
            @Override
            public void onSuccess(DocumentReference documentReference) {
                Toast.makeText(Quvestion_meneger.this, "Вопрос успешно создан", Toast.LENGTH_SHORT).show();
            }
        }).addOnFailureListener(new OnFailureListener() {
            @Override
            public void onFailure(@NonNull Exception e) {
                Toast.makeText(Quvestion_meneger.this, "Ошибка при создании вопроса", Toast.LENGTH_SHORT).show();
            }
        });
    }
}
```

1.16 Код модуля Qvestion _vvod

```
package com.example.razvivaska_plahova;

import static android.content.ContentValues.TAG;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
```

```
import android.content.Intent;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Spinner;
import android.widget.TextView;
import android.widget.Toast;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.OnFailureListener;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.firestore.DocumentReference;
import com.google.firebase.firestore.FirebaseFirestore;
import com.google.firebase.firestore.QueryDocumentSnapshot;
import com.google.firebase.firestore.QuerySnapshot;

import java.util.ArrayList;
import java.util.List;

public class Qvestion_vvod extends AppCompatActivity {
    private Spinner spinnerSubject;
    private Spinner spinnerTheme;
    private EditText editTextAnswer4;
    private Button exit;
    private Button buttonCreateQuestion;
    private EditText editTextQuestionText;
    private ArrayAdapter<Subject> adapter;
    private FirebaseFirestore db;
    private ArrayList<Subject> subjects;
    private ArrayList<String> themes;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_qvestion_vvod);
        spinnerSubject = findViewById(R.id.spinner_subject);
        spinnerTheme = findViewById(R.id.spinner_theme);
        editTextQuestionText = findViewById(R.id.edit_text_question_text);
        editTextAnswer4 = findViewById(R.id.edit_text_answer4);
        exit = findViewById(R.id.button_exit);
        buttonCreateQuestion = findViewById(R.id.button_create_question);
        db = FirebaseFirestore.getInstance();
        subjects = new ArrayList<>();
        themes = new ArrayList<>();
        adapter = new ArrayAdapter<Subject>(Qvestion_vvod.this, android.R.layout.simple_spinner_item, subjects) {
            @Override
            public View getView(int position, View convertView, ViewGroup parent) {
                View view = super.getView(position, convertView, parent);
                TextView textView = (TextView) view.findViewById(android.R.id.text1);
                textView.setText(((Subject) getItem(position)).getName());
                return view;
            }
        };
        spinnerSubject.setAdapter(adapter);
        exit.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
```

```
            startActivity(new Intent(Qvestion_vvod.this, Menedger_cabinet.class));
        }
    });
    buttonCreateQuestion.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            createQuestion();

        }
    });

    loadSubjects(); // Call loadSubjects() at the end
    spinnerSubject.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
        @Override
        public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {
            Subject selectedSubject = (Subject) spinnerSubject.getSelectedItem();
            themes.clear();
            List<String> themesList = selectedSubject.getThemes();
            if (themesList != null && !themesList.isEmpty()) {
                themes.addAll(themesList);
                ArrayAdapter<String> adapter = new ArrayAdapter<>(Qvestion_vvod.this,
android.R.layout.simple_spinner_item, themes);
                adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
                spinnerTheme.setAdapter(adapter);
            } else {
                // Handle the case when there are no themes
                spinnerTheme.setAdapter(null); // or some other default adapter
            }
        }

        @Override
        public void onNothingSelected(AdapterView<?> parent) {
            // Do nothing
        }
    });
}
private void loadSubjects() {
    db.collection("subjects")
            .get()
            .addOnCompleteListener(new OnCompleteListener<QuerySnapshot>() {
                @Override
                public void onComplete(@NonNull Task<QuerySnapshot> task) {
                    if (task.isSuccessful()) {
                        subjects.clear(); // Clear the list before adding new data
                        for (QueryDocumentSnapshot document : task.getResult()) {
                            Subject subject = document.toObject(Subject.class);
                            subjects.add(subject);
                        }
                        adapter.notifyDataSetChanged(); // Notify the adapter of data changes
                    } else {
                        Log.d(TAG, "Error getting subjects: ", task.getException());
                    }
                }
            });
}

private void createQuestion() {
    Subject selectedSubject = (Subject) spinnerSubject.getSelectedItem();
    String theme = spinnerTheme.getSelectedItem().toString();
    String questionText = editTextQuestionText.getText().toString();
    String correctAnswer = editTextAnswer4.getText().toString();

    String answer1 = "";
```

```
        String answer2 = "";
        String answer3 = "";
        String CorrectAncwer = correctAnswer;
        String Qvestion_type = "Вопрос с вводимом ответом";
        Questions question = new Questions(selectedSubject.getName(), theme, questionText, answer1, answer2,
answer3, CorrectAncwer,Qvestion_type);
        Toast.makeText(Qvestion_vvod.this, "Question", Toast.LENGTH_SHORT).show();

        db.collection("Questions").add(question).addOnSuccessListener(new
OnSuccessListener<DocumentReference>() {
            @Override
            public void onSuccess(DocumentReference documentReference) {
                Toast.makeText(Qvestion_vvod.this, "Вопрос успешно создан", Toast.LENGTH_SHORT).show();
            }
        }).addOnFailureListener(new OnFailureListener() {
            @Override
            public void onFailure(@NonNull Exception e) {
                Toast.makeText(Qvestion_vvod.this, "Ошибка при создании вопроса",
Toast.LENGTH_SHORT).show();
            }
        });
    }
}
```

1.17 Код модуля RecyclerItemClickListener

```
package com.example.razvivaska_plahova;

import android.content.Context;
import android.view.GestureDetector;
import android.view.MotionEvent;
import android.view.View;

import androidx.recyclerview.widget.RecyclerView;

public class RecyclerItemClickListener implements RecyclerView.OnItemTouchListener {

    private OnItemClickListener mListener;

    public interface OnItemClickListener {
        void onItemClick(View view, int position);
        void onLongItemClick(View view, int position);
    }

    GestureDetector mGestureDetector;

    public RecyclerItemClickListener(Context context, final RecyclerView recyclerView, OnItemClickListener
listener) {
        mListener = listener;
        mGestureDetector = new GestureDetector(context, new GestureDetector.SimpleOnGestureListener() {
            @Override
            public boolean onSingleTapUp(MotionEvent e) {
                return true;
            }

            @Override
            public void onLongPress(MotionEvent e) {
                View child = recyclerView.findChildViewUnder(e.getX(), e.getY());
                if (child != null && mListener != null) {
                    mListener.onLongItemClick(child, recyclerView.getChildAdapterPosition(child));
                }
            }
        });
```

```
    }

    @Override
    public boolean onInterceptTouchEvent(RecyclerView rv, MotionEvent e) {
        View child = rv.findChildViewUnder(e.getX(), e.getY());
        if (child != null && mListener != null && mGestureDetector.onTouchEvent(e)) {
            mListener.onItemClick(child, rv.getChildAdapterPosition(child));
            return true;
        }
        return false;
    }

    @Override
    public void onTouchEvent(RecyclerView rv, MotionEvent e) {
    }

    @Override
    public void onRequestDisallowInterceptTouchEvent(boolean disallowIntercept) {
    }
}
```

1.18 Код модуля Result_ycheniki

```
package com.example.razvivaska_plahova;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.content.Context;
import android.widget.TextView;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.firestore.FirebaseFirestore;
import com.google.firebase.firestore.QueryDocumentSnapshot;
import java.util.ArrayList;
import java.util.List;

// Класс для хранения данных о результате теста
class Result {
    private String email;
    private String grade;
    private int result;
    private String subject;
    private String testName;
    private String theme;

    public Result(String email, String grade, int result, String subject, String testName, String theme) {
        this.email = email;
        this.grade = grade;
        this.result = result;
        this.subject = subject;
        this.testName = testName;
        this.theme = theme;
    }

    // Геттеры
    public String getEmail() { return email; }
    public String getGrade() { return grade; }
```

```
    public int getResult() { return result; }
    public String getSubject() { return subject; }
    public String getTestName() { return testName; }
    public String getTheme() { return theme; }
}

public class Result_ycheniki extends AppCompatActivity {
    private RecyclerView recyclerView;
    private ResultsAdapter adapter;
    private List<Result> results = new ArrayList<>();

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

        recyclerView = findViewById(R.id.recyclerView);
        recyclerView.setLayoutManager(new LinearLayoutManager(this));
        adapter = new ResultsAdapter(this, results);
        recyclerView.setAdapter(adapter);

        FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();
        if (user != null) {
            String userEmail = user.getEmail();
            FirebaseFirestore db = FirebaseFirestore.getInstance();
            db.collection("results")
                    .whereEqualTo("email", userEmail)
                    .get()
                    .addOnSuccessListener(queryDocumentSnapshots -> {
                        results.clear();
                        for (QueryDocumentSnapshot document : queryDocumentSnapshots) {
                            String grade = document.getString("grade");
                            int result = document.getLong("result").intValue();
                            String subject = document.getString("subject");
                            String testName = document.getString("testName");
                            String theme = document.getString("theme");

                            Result resultItem = new Result(userEmail, grade, result, subject, testName, theme);
                            results.add(resultItem);
                        }
                        adapter.notifyDataSetChanged();
                    })
                    .addOnFailureListener(e -> {
                        // Обработка ошибки загрузки данных
                    });
        }
    }
}

class ResultsAdapter extends RecyclerView.Adapter<ResultsAdapter.ViewHolder> {
    private List<Result> results;
    private Context context;

    public ResultsAdapter(Context context, List<Result> results) {
        this.context = context;
        this.results = results;
    }

    @NonNull
    @Override
    public ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
        View itemView = LayoutInflater.from(parent.getContext())
                .inflate(R.layout.result_item, parent, false);
```

```
        return new ViewHolder(itemView);
    }

    @Override
    public void onBindViewHolder(@NonNull ViewHolder holder, int position) {
        Result result = results.get(position);
        holder.testName.setText(result.getTestName());
        holder.subject.setText(result.getSubject());
        holder.grade.setText(result.getGrade());
        holder.result.setText("Результат: " + result.getResult());
    }

    @Override
    public int getItemCount() {
        return results.size();
    }

    public static class ViewHolder extends RecyclerView.ViewHolder {
        public TextView testName;
        public TextView subject;
        public TextView grade;
        public TextView result;

        public ViewHolder(@NonNull View itemView) {
            super(itemView);
            testName = itemView.findViewById(R.id.testName);
            subject = itemView.findViewById(R.id.subject);
            grade = itemView.findViewById(R.id.grade);
            result = itemView.findViewById(R.id.result);
        }
    }
} }
```

1.19 Код модуля Result_ychitel

```
package com.example.razvivaska_plahova;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.os.Bundle;
import android.widget.EditText;
import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.firestore.FirebaseFirestore;
import com.google.firebase.firestore.QueryDocumentSnapshot;

import java.util.ArrayList;
import java.util.List;
public class Result_ychitel extends AppCompatActivity {

    private EditText emailInput;
    private EditText idInput;
    private RecyclerView recyclerView;
    private ResultsAdapter adapter;
    private List<Result> results = new ArrayList<>();
```

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_result_ychitel);
        emailInput = findViewById(R.id.emailInput); // ID вашего EditText для email
        idInput = findViewById(R.id.idInput); // ID вашего EditText для ID

        recyclerView = findViewById(R.id.recyclerView);
        recyclerView.setLayoutManager(new LinearLayoutManager(this));
        adapter = new ResultsAdapter(this, results);
        recyclerView.setAdapter(adapter);

        findViewById(R.id.searchButton).setOnClickListener(view -> {
            String email = emailInput.getText().toString().trim();
            String id = idInput.getText().toString().trim();

            loadResultsFromFirestore(email, id);
        });
    }

    private void loadResultsFromFirestore(String email, String id) {
        FirebaseFirestore db = FirebaseFirestore.getInstance();
        db.collection("results")
                .whereEqualTo("email", email)
                .get()
                .addOnSuccessListener(queryDocumentSnapshots -> {
                    results.clear();
                    for (QueryDocumentSnapshot document : queryDocumentSnapshots) {
                        String grade = document.getString("grade");
                        int result = document.getLong("result").intValue();
                        String subject = document.getString("subject");
                        String testName = document.getString("testName");
                        String theme = document.getString("theme");

                        Result resultItem = new Result(email, grade, result, subject, testName, theme);
                        results.add(resultItem);
                    }
                    adapter.notifyDataSetChanged();
                })
                .addOnFailureListener(e -> {
                    // Обработка ошибки загрузки данных
                    Toast.makeText(this, "Ошибка загрузки данных", Toast.LENGTH_SHORT).show();
                });
    }
}
```

1.20 Код модуля ychenik_cabinet

```
package com.example.razvivaska_plahova;

import androidx.appcompat.app.AppCompatActivity;

import android.annotation.SuppressLint;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

public class ychenik_cabinet extends AppCompatActivity {
    private Button buttonCreateTest, result;
    private TextView email_view,log_view;
    @Override
```

```
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_ychenik_cabinet);
        Intent intent = getIntent();
        buttonCreateTest = findViewById(R.id.test_baton);
        result = findViewById(R.id.test_res_baton);
        email_view = findViewById(R.id.pochta);
        log_view = findViewById(R.id.log);
        log_view.setText(intent.getStringExtra("login"));
        email_view.setText(intent.getStringExtra("email"));
        buttonCreateTest.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                startActivity(new Intent(ychenik_cabinet.this, Change_the_test.class));
            }
        });
        result.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                startActivity(new Intent(ychenik_cabinet.this, Result_ycheniki.class));
            }
        });
    }
}
```

1.22 Код модуля ycitel_cabinet

```
package com.example.razvivaska_plahova;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;

public class ycitel_cabinet extends AppCompatActivity {
    private Button result;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_ycitel_cabinet);
        result = findViewById(R.id.res_yc);
        result.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                startActivity(new Intent(ycitel_cabinet.this, Result_ychitel.class));
            }
        });
    }
}
```

Приложение 3. диаграммы прецедентов