**Техническое задание (ТЗ)**

**На разработку мобильного приложения «CodeMentor»**

**Общие сведения**

* Наименование проекта: CodeMentor.
* Краткое описание: Интерактивный мобильный тренажёр для изучения языков программирования C++ и Assembler (NASM) с использованием AI-ассистента для анализа кода, предоставления обратной связи и генерации решений.
* Целевая платформа: Android.
* Основная цель проекта: Создать образовательный инструмент, который помогает пользователям не просто решать задачи, а учиться писать чистый, эффективный и правильный код, получая мгновенную и персонализированную обратную связь от AI-ментора.
* Целевая аудитория:
  + Студенты технических специальностей, изучающие низкоуровневое программирование.
  + Начинающие разработчики, желающие улучшить качество кода.
  + Энтузиасты программирования, ищущие современный способ практики.

**Функциональные требования**

2.1 Главный экран и выбор задачи

* Пользователь должен иметь возможность выбрать язык программирования:
  + C++
  + Ассемблер (NASM)
* Пользователь должен иметь возможность выбрать уровень сложности задачи (“Легкий” "Нормальный", "Средний", "Профессиональный").
* После выбора параметров приложение должно сгенерировать и отобразить случайную задачу соответствующую выбранным языку и сложности.

2.2. Экран решения задачи

* Интерфейс должен содержать:
  + Текстовое поле с условием задачи.
  + Редактор кода для выбранного языка.
  + Кнопку для отправки кода на проверку AI-ассистенту.

2.3. Механизм анализа кода AI-ассистентом

* После отправки кода пользователя, приложение передает его и условие задачи на анализ AI-модели.
* В случае правильного решения:
  + AI подтверждает, что решение верное.
  + AI предлагает для сравнения эталонное (оптимальное или альтернативное) решение с комментариями (если это возмонжно).
* В случае неправильного решения:
  + AI указывает на конкретную логическую или синтаксическую ошибку в коде.
  + AI предоставляет объяснение сути ошибки.

2.4. Интерактивный чат с AI-ассистентом

* На экране решения задачи должна быть доступна функция чата.
* Пользователь должен иметь возможность задать AI-ассистенту уточняющие вопросы по:
  + Условию задачи.
  + Предложенному AI решению или подсказке.
  + Общим концепциям, связанным с задачей.
* История чата должна сохраняться в рамках текущей сессии решения задачи.

2.5. Хранение прогресса и статистика

* Приложение должно сохранять прогресс пользователя локально на устройстве.
* После решения каждой задачи приложение должно обновлять рейтинг пользователя по данному языку программирования.
* Расчет рейтинга:
  + Рейтинг должен отображаться в процентах (например, в виде круговой диаграммы).
  + На рейтинг влияют: сложность решенной задачи и количество попыток (решена с ошибками или с первого раза).
* Должен быть экран "Профиль" или "Статистика", где пользователь может увидеть свой общий прогресс и рейтинг по каждому языку.

**Технические требования**

* Платформа: Android (минимальная поддерживаемая версия API уточняется на этапе проектирования).
* Язык программирования: Kotlin.
* Среда разработки: Android Studio.
* Пользовательский интерфейс (UI):
  + Разметка экранов должна быть реализована с использованием XML.
  + Для отображения списков (задачи, история чата) использовать RecyclerView.
* Хранение данных:
  + Для хранения базы задач и прогресса пользователя использовать локальную базу данных SQL (рекомендуется SQLite).
* Взаимодействие с AI:
  + Интеграция с AI-моделью должна осуществляться через API. Для сетевых запросов использовать библиотеку Retrofit.
  + Ядро AI-ассистента: Google Gemini API (или аналогичная модель, способная к анализу кода и генерации текста).

**Этапы разработки**

1. Проектирование:
   * Создание макетов всех экранов приложения.
   * Проектирование схемы базы данных SQL для хранения задач и прогресса.
2. Настройка окружения и базовой архитектуры:
   * Создание проекта в Android Studio.
   * Подключение необходимых библиотек (Retrofit, RecyclerView и т.д.).
3. Реализация UI и модели данных:
   * Верстка XML-макетов.
   * Настройка базы данных.
4. Интеграция API:
   * Настройка Retrofit для взаимодействия с Google Gemini API.
   * Реализация запросов на анализ кода и получение ответов.
5. Реализация бизнес-логики:
   * Написание кода для генерации задач, управления состоянием.
   * Реализация логики отправки кода на проверку, обработки ответов от AI и их отображения.
   * Реализация механики чата.
   * Реализация системы подсчета рейтинга.
6. Тестирование и отладка:
   * Проверка всех функций приложения.
   * Тестирование взаимодействия с нейросетью на разных типах кода (правильный, с ошибками).
   * Отладка и исправление выявленных дефектов.
7. Подготовка к релизу:
   * Сборка релизной версии приложения (.apk или .aab).
   * Подготовка иконок и сопроводительных материалов.