**ТЕХНИЧЕСКОЕ ЗАДАНИЕ  
на разработку клиент-серверного приложения**

**AI Assistant**

**Исполнитель:** \_Катлюшкин С.Р.\_\_Янченко М.А.\_

**Заказчик:** \_Буркацкий\_К.А.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Дата утверждения:** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Введение

1.1. Назначение программы

Настоящее ТЗ определяет требования к разработке клиент-серверного приложения для поиска схожих музыкальных треков по текстам песен с использованием методов искусственного интеллекта. Система предназначена для автоматического сбора текстов песен, их анализа и поиска схожих треков на основе семантического соответствия.

1.2. Цель разработки

Создание высокопроизводительной, безопасной и масштабируемой системы, обеспечивающей:

* Сбор текстов песен из открытых источников (например, Genius API, Lyrics.ovh API);
* Анализ семантического содержания текстов песен;
* Поиск схожих треков по содержанию текста;
* Генерацию отчетов с результатами поиска;
* Шифрование данных при передаче и хранении;
* Логирование операций и аналитика запросов.

2. Общие положения

2.1. Состав приложения

- Серверный компонент на Python с помощью Flask с интеграцией внешних API.

- База данных SQLite для хранения логов запросов, пользовательских данных и метаинформации

- Клиентские приложения с графическим интерфейсом для Windows

2.2. Основные функции приложения

- Получение текстов песен из открытых источников;

- Анализ текста с использованием NLP;

- Поиск схожих песен по содержанию;

- Выдача результатов в JSON и CSV;

- Логирование запросов и действий пользователей.

2.3. Основные функции компонентов

Серверный компонент:

- Многопоточное Python-приложение на FastAPI для обработки данных и ML-анализа.

База Данных:

- SQLite для хранения текстов песен, результатов анализа и пользователей

Клиентское приложение:

- Составление запроса, его шифрование и отправка на сервер.  
- Получение зашифрованного ответа с сервера, его дешифровка и показ клиенту.

3. Функциональные требования  
Для более точного рассмотрения кода и его функционала представлен Git данного проекта (https://github.com/RBXPrOfIX/YP2025KY). В README.MD основная информация.

3.1. Серверная часть

- Обработка запросов. Поддержка REST API с эндпоинтами: /get\_lyrics, /find\_similar, /generate\_report  
- Интеграция с источниками данных. Доступ к Genius API, Lyrics.ovh API  
- Многопоточность. Поддержка ≥10 одновременных запросов  
- Логирование в server.log и client.log. Фиксация в БД информации об устройстве в зависимости от клиента (информация о ПК, аккаунте Телеграмм, используемый браузер).  
- Логирование. Фиксация времени запроса, IP-адреса, типа операции, статуса выполнения  
- Docker контейнер для автономной работы сервера.

3.2. Клиентская часть

- Обработка Post запросов аналогично Основному Функционалу на сервере.  
- Приложение для Windows с интерфейсом на PyQt5.   
- Взаимодействие с REST API, визуализация результатов анализа

3.2. Часть БД

- Хранение дешифрованных запросов с клиента и ответов с сервера.

4. Этапы разработки

Этап 1: Разработка Технического Задания (ТЗ)

- Срок: 1 неделя.

- Результат: готовое и одобренное ТЗ.

Этап 2: Настройка Git и документации

- Срок: 3 неделя.

- Результат: готовая документация и Git.

Этап 3: Разработка серверной части и его API

- Срок: 5 неделя.

- Результат: готовая и функционирующая серверная часть.

Этап 4: Создание клиентского приложения с GUI

- Срок: 7 неделя.

- Результат: готовая и функционирующая клиентская часть.

Этап 5: Интеграция БД

- Срок: 9 неделя.

- Результат: настроенная БД для приложения.

Этап 6: Взаимодействие БД и приложения

- Срок: 11 неделя.

- Результат: функционирующая связь между БД и приложением

Этап 7: Интеграция Метода Шифрования в приложение

- Срок: 13 неделя.

- Результат: настроенная шифрованная связь в приложении.

Этап 8: Развертывание и Тестирование проекта

- Срок: 15 неделя.

- Результат: полностью готовое и функционирующее приложение.