**Отчёт по выполнению проекта**

Для лучшего понимания были созданы несколько файлов в которых есть дополнительная информация. Ниже их название. Все файлы залиты на GhitHub.  
1. Замечания к структуре базы данных.docx

2. Техническое задание на проект Mountain Pass Application.docx

3. план реализации.docx

4. Отчёт по выполнению проекта.docx (Текущий документ)

5. Команды для запуска тестов.docx

**🔹 Анализ существующей базы данных**

Перед началом разработки была **исследована существующая база данных** ФСТР.  
В ходе анализа **были выявлены следующие недостатки**:

* **Отсутствует отдельная таблица пользователей** – данные о пользователях хранились в таблице перевалов.
* **Отсутствует нормализованная таблица координат** – координаты хранились в таблице перевалов вместо отдельной сущности.
* **Данные об уровнях сложности хранились в виде JSON** – что усложняло работу с ними.
* **Изображения хранились в виде JSON-объекта в поле images** – это не соответствует принципам нормализации.
* **Отсутствует механизм хранения GPS-треков** – невозможно загружать и обрабатывать маршруты в формате GPX/KML.
* **Отсутствует система статусов модерации** – невозможно отслеживать статус рассмотрения перевала.

Исходя из выявленных недостатков, **была разработана и реализована новая структура базы данных**.

**🔹 Что было сделано**

✅ **Этап 1. Улучшение структуры БД**

* Разработана **новая структура БД** с выделением отдельных таблиц User, Coords, PerevalImages, PerevalGpsTracks.
* Добавлена **таблица статусов модерации** (status).
* Все связи в БД **приведены к нормальной форме**.

✅ **Этап 2. Разработка класса работы с БД**

* Реализован **класс DatabaseService** для работы с БД.
* Добавлены методы:
  + add\_pereval() – сохранение данных о перевале.
  + add\_image() – сохранение изображений.
  + add\_gps\_track() – загрузка GPS-треков.
  + add\_related\_objects() – связывание перевалов с вершинами/хребтами.
  + get\_weather() – получение сохранённой информации о погоде.

✅ **Этап 3. Разработка REST API**

* Реализованы **основные API-маршруты**:
  + **POST /submitData** – приём данных о перевале.
  + **POST /uploadImage** – загрузка изображений перевала.
  + **POST /uploadTrack** – загрузка GPS-треков.
* Интеграция с внешними API:
  + **Google Maps API** – генерация ссылки на карту.
  + **OpenWeather API** – получение прогноза погоды.
  + **Yandex Maps API** – построение маршрута к перевалу.
  + **Strava/Wikiloc API** – работа с GPS-треками.
* **Переменные API-ключей загружаются из .env**.

✅ **Этап 4. Тестирование API**

* Написаны **автоматические тесты (pytest)** для:
  + **POST /submitData** – проверка корректности данных.
  + **POST /uploadImage** – проверка загрузки изображений.
  + **POST /uploadTrack** – проверка загрузки GPS-треков.
* Все тесты **успешно пройдены** (3 passed).

✅ **Этап 5. Работа с Git**

* **Вся работа велась в ветке submitData**.
* Все изменения фиксировались отдельными коммитами:

*- Добавлена структура базы данных и класс работы с БД.*

*- Реализован REST API для приёма данных о перевале (POST /submitData).*

*- Реализован API для загрузки изображений (POST /uploadImage).*

*- Реализован API для загрузки GPS-треков (POST /uploadTrack).*

*- Реализована интеграция с внешними API (Google Maps, OpenWeather, Yandex Maps, Strava).*

*- Добавлены автоматические тесты для REST API.*

*- Объединение ветки `submitData` с `master`.*

* После завершения разработки **ветка submitData была слита в master**.

**Итог**

🔹 **Все требования выполнены**.  
🔹 **REST API разработано, протестировано и зафиксировано в Git**.  
🔹 **Все данные передаются и обрабатываются в соответствии с требованиями**.  
🔹 **Интеграции с картами и прогнозом погоды настроены (но не тестировались из-за отсутствия API-ключей)**.

**Проект завершён и готов к передаче на проверку.**