SRS

SRS Revision History

2/17 - Document Created
2/26 - Updated Document
3/12 - Final Version

Concept of Operations / ConOps

Software Description

TrailGrade is a Python-based system that analyzes hiking trail data to generate multidimensional difficulty ratings using various components like cardio intensity, trail terrain, etc. Users can view nearby trail data and difficulty ratings as well as create their own trails.

Current System / Situation

There are currently a few versions of our system that are available to users. However, many users have expressed complaints over current systems with current information being unhelpful or unavailable. The goal of our system is to create a better system for users that provides them with accurate and helpful data.

Justification for a new System

Other versions of trail difficulty and information apps have a lack of depth and poor difficulty ratings that fail to accurately depict trail conditions for users, which has led to poor reviews. Our version will attempt to rectify current issues by implementing a better difficulty system and providing better information about each trail.

Operational Features of the Proposed System

A UI for users to filter and view trail difficulty, elevation change, distance, terrain, and location for trails nearby.

User Classes + Modes of Operation

Hikers/Anyone: The system is designed for anyone interested in hiking/going on a hike that wants to view nearby options for difficulty and information. There would be little variation regardless of who the user is.

Use Cases

1. A user wants to start hiking to get in shape. They know what trailhead they want to start at, but they need to make sure they pick a trail that won't be too tiring for them. They choose a trailhead / start location from the system and are presented with a list of options of trails sorted by difficulty, with easiest first. If they have other specifications as well, they can define them using a filter system.

2. A user frequently hikes a trail that isn't in the TrailGrade system. They'd like to see how its difficulty compares to other trails in the system.

Specific Requirements

External Interfaces (Inputs and Outputs)

Database
-Stores Trail Data
GET/INSERT/DELETE inputs
Json outputs

API
-Connects UI and Database
-Uses Queries to get information from Database

Functions

-Accesses User Location to produce recommendations for nearby trails
-User Requests Data - API - Accesses Database - Generates information/visualization

Usability Requirements

-Effectiveness based on data/location/visual accuracy
-Feedback from user opinions

Performance Requirements

100% of user information requests should be processed and delivered within 2 seconds. The user experience should be fluid and seamless, limited waiting.

Software System Attributes

The main attribute of the software should be its reliability for users. All information should be accurate and up to date. Difficulty ratings should be well-defined and accurate, to the satisfaction of the user. Individual profiles are not necessary as location for nearby data is the only requirement, so security shouldn't be an issue.

SDS

SDS Revision History

2/17 - Document Created
2/26 - Updated Document
3/12 - Final Version

System Overview

The system ingests GPS trail data, processes it through various analytical modules, stores the results in a structured database, and presents the information through interactive visualizations. The system is designed to be modular, for incremental development purposes.

Software Architecture

Components

1. Trail Data Ingestion Engine
   - Handles GPS data input
   - Validates and standardizes incoming formats
   - Performs initial data cleaning
   - Converts to internal data structure
   - Provides clean data to other components
2. Trail Database Manager
   - Manages database operations
   - Stores trail data and analysis results
   - Handles data persistence and retrieval
   - Provides data access interface
3. Geographic Analysis Core
   - Processes trail segments
   - Calculates elevation changes and metrics
   - Analyzes terrain characteristics
   - Generates difficulty ratings
4. API Service
   - Provides endpoints for data access
   - Handles request validation
   - Manages API response formatting
   - Coordinates between components
5. Visualization Generator
   - Creates interactive trail maps using Matplotlib
   - Generates elevation profiles and difficulty overlays
   - Produces visual analysis reports

Component Interactions

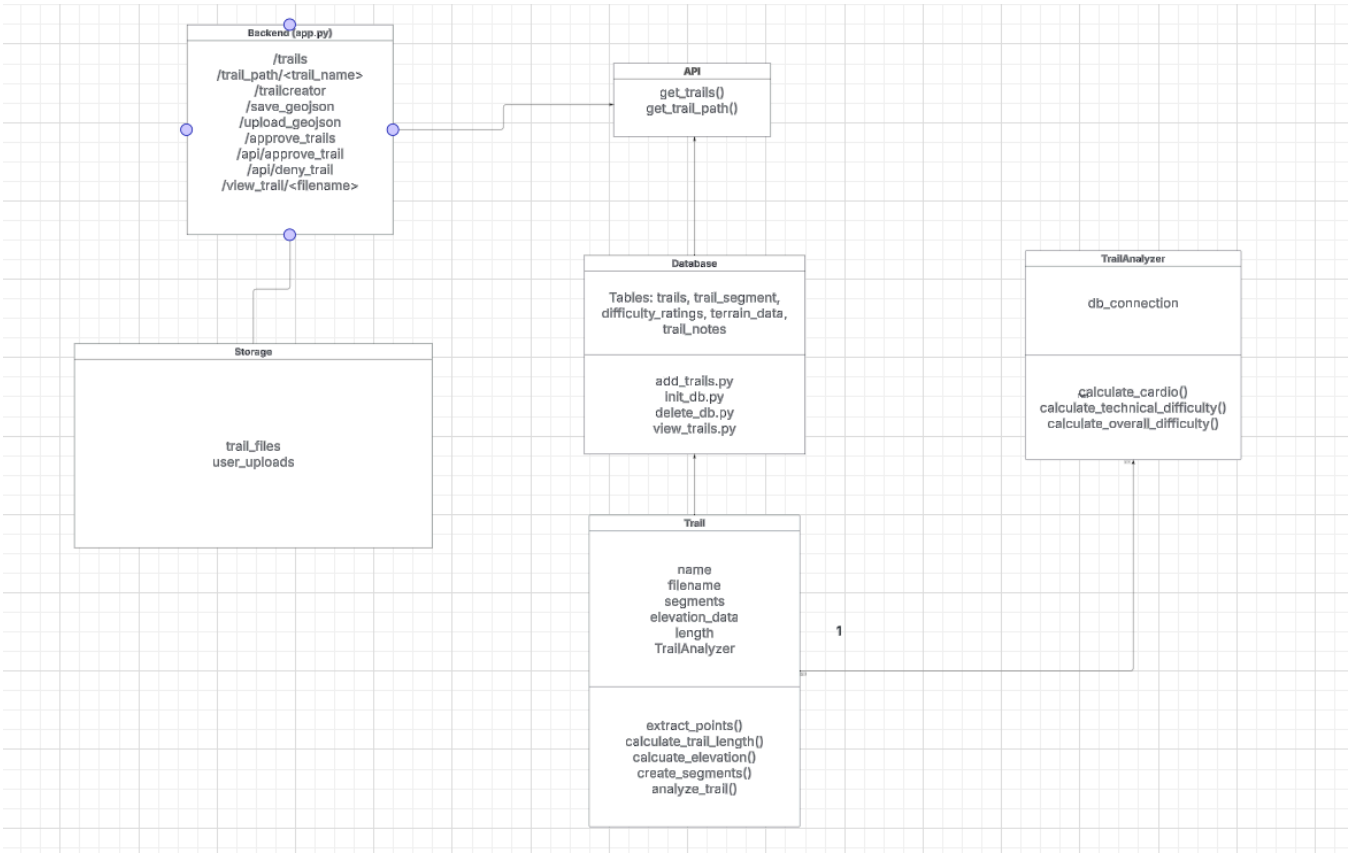
- The visualization generator interacts with other components through the API service

- The API service coordinates all data flow between components
- The geographic analysis core processes data from the ingestion engine and stores results through the database manager

Key Flows:

1. The ingestion engine processes raw gps data and passes it to the database
2. The analysis core pulls trail data for processing and returns difficulty ratings
3. The database manager maintains all persistent data and provides it to other components
4. The API service coordinates requests between components
5. The visualization generator creates visual representations based on analysis results

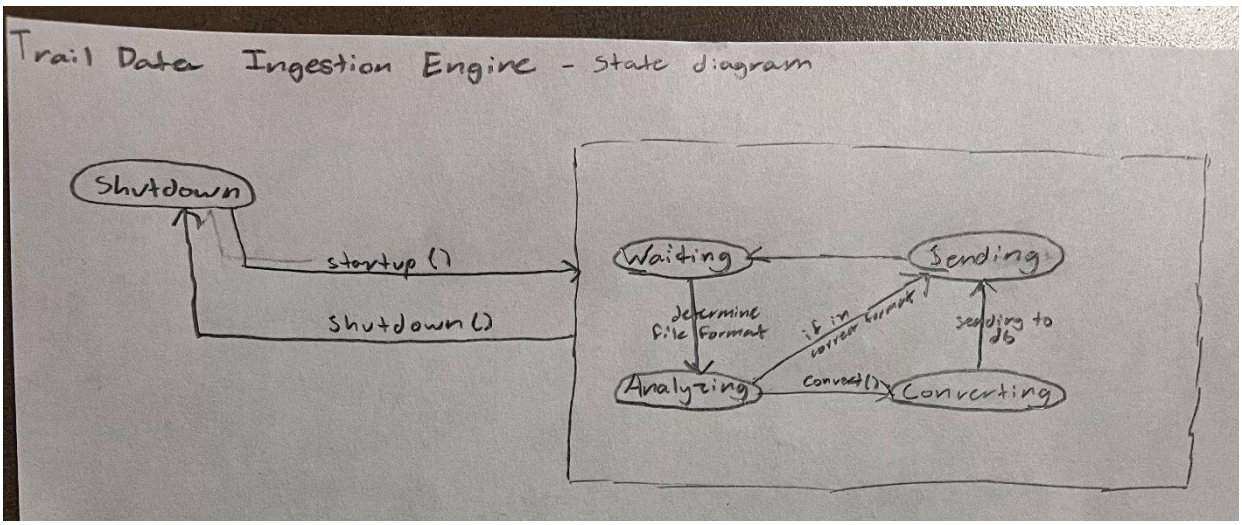
Project-Wide UML Class Diagram



Software Modules

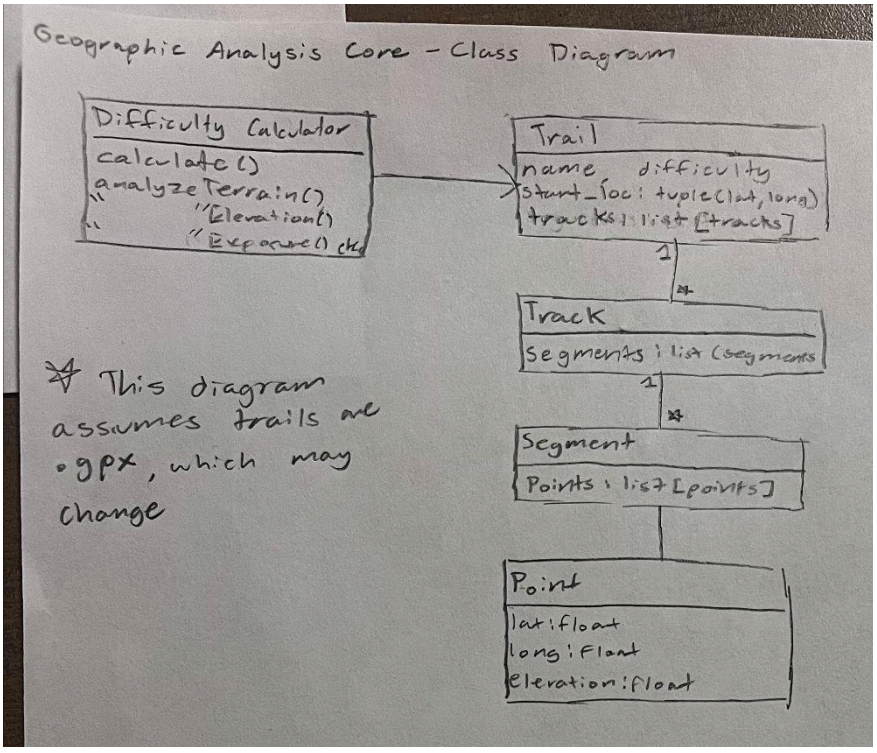
Trail Data Ingestion Engine

Serves as the system's data intake processor, accepts various formats of GPS trail data and converts them into a standardized internal format



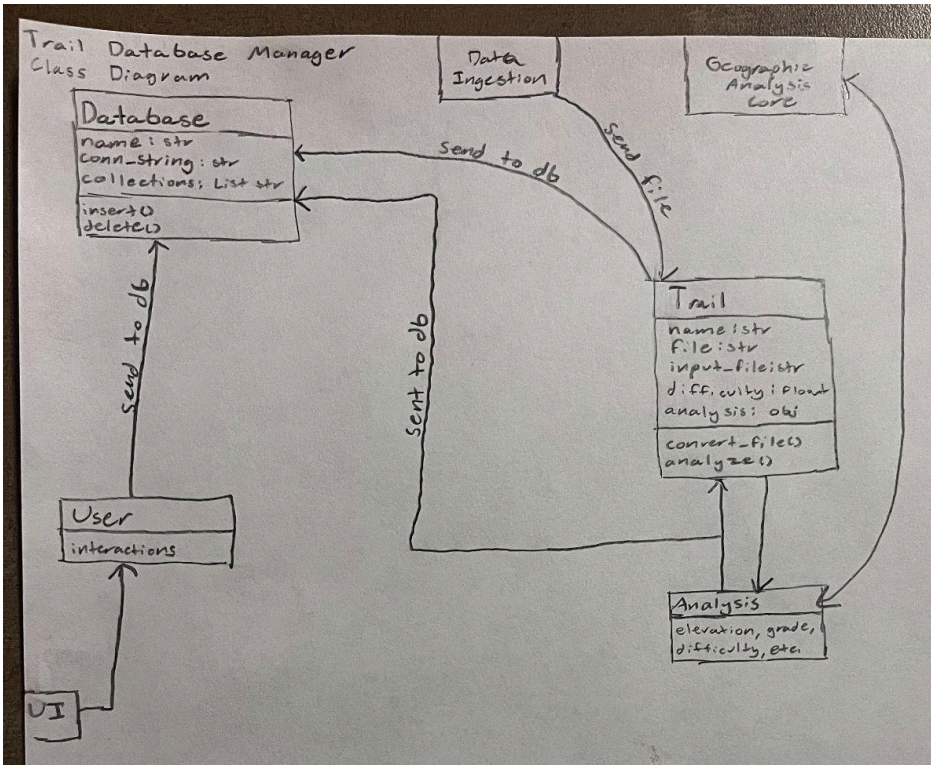
Geographic Analysis Core

Processes trail segments and generates difficulty ratings based on factors including elevation change, terrain, exposure, etc.



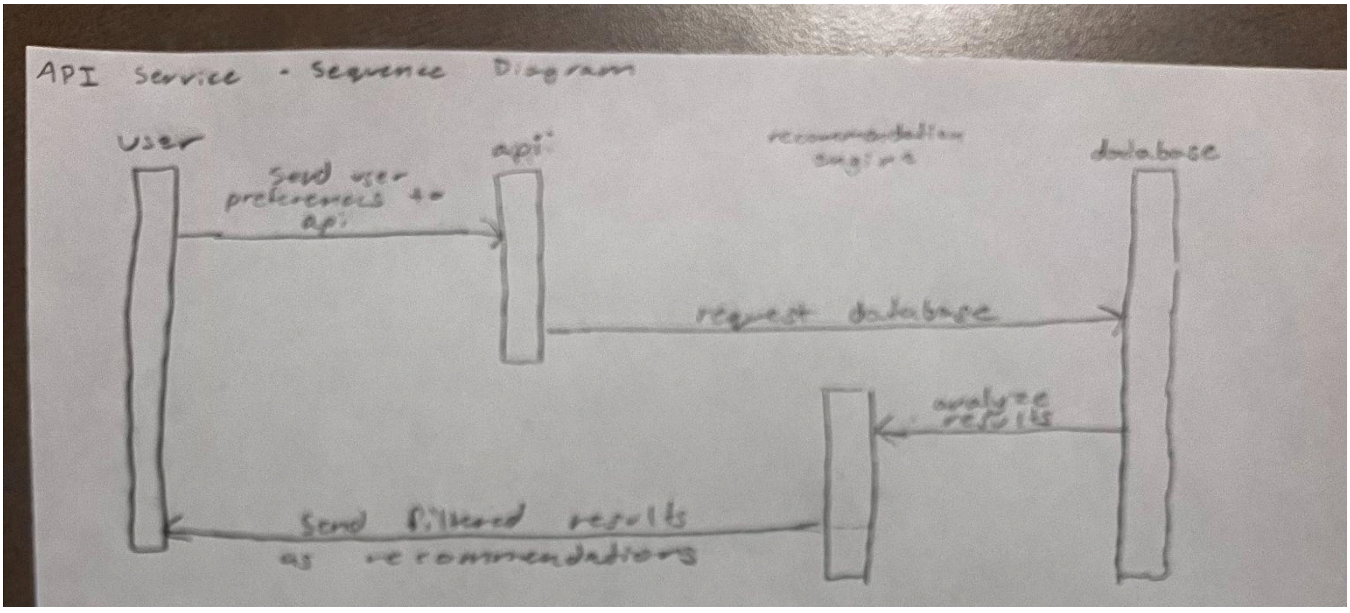
Trail Database Manager

Handles all storage operations, maintains trail data, analysis results, and user interaction history in a structured database



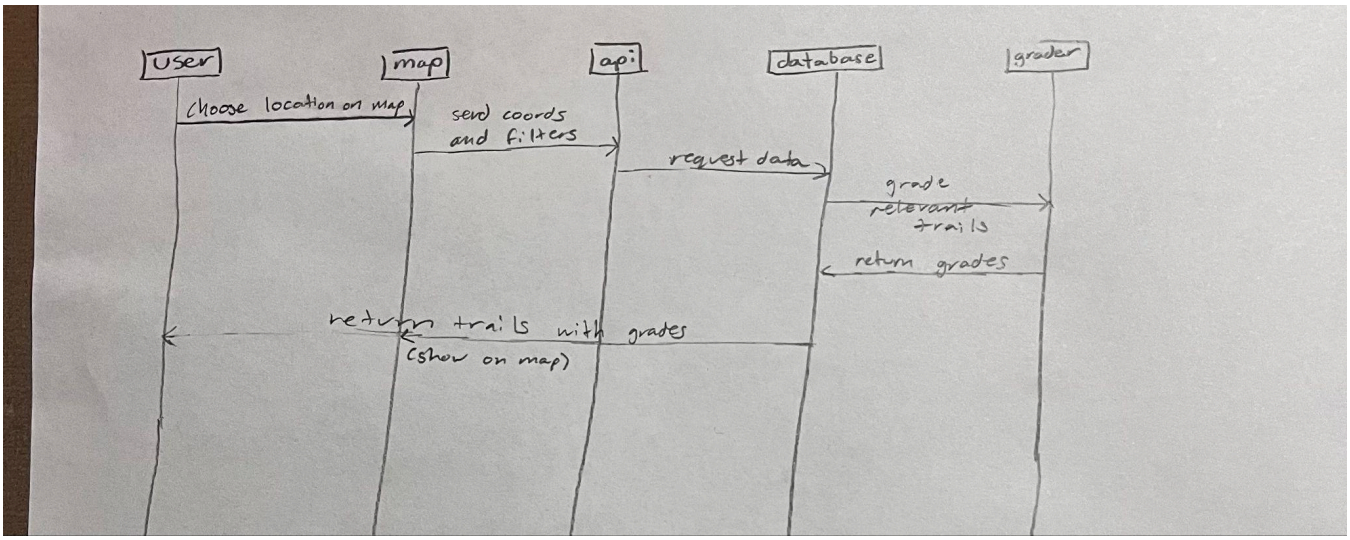
API Service

Provides an interface for external applications to access trail data, analyses, and recommendations



Dynamic Models

Sequence diagram of a user picking a location on a map and being given nearby trails



Project Timeline (February 19 - March 12):

Milestone	Tasks	Time
Research Data	Study GPS data, algorithms, and database structures	2 days
Database Setup	Configure database, design structure, CRUD operations, and initial tests	3 days
Frontend Development	Create skeleton, design pages, and setup basic interactions	4 days
API Development	Create all API endpoints, request validation, and initial testing	5 days
Algorithm Development	Implement trail difficulty ratings and optimizations	5 days
Data Analysis and Geographic	Process real trail data, test	4 days

System	calculations, and visualize endpoints	
Final Testing and Optimization	Debug API, improve UI, optimize performance, and test integrations	4 days
Submission and Presentation	Prepare slides, demo project, and finalize documentation	4 days

Roles

Caden - Database, Data Ingestion Engine
Jake - Core Analysis, Frontend Polishing
Parker - API
Logan - Project Management/Organization
Giovanni - Initial Frontend