Elevate Jeff Muri Steve Borst Yifu Liu Kailash Srinivasan Eric Atwood

1. **The Problem.**

Given a starting and ending location, the Elevate program will allow the user to find a path between the two which either minimizes or maximizes elevation gain during the trip. When maximizing, user enters a time desired for the trip, and then the program determines the path of most change in elevation that covers that distance or time.

2. **The Design**.

We aim to make a desktop application and primarily use Python to build the program. Each component of the application is listed as follows:

Routing algorithm: The algorithm will take in starting and ending elevation as inputs (both latitude and longitude are used in coordinates), keep track of current elevation, and will compute the path of least change in elevation. Once the most efficient path is computed, the information will get sent to the data model. It will include a time constraint as an input for maximizing elevation change. If this option is selected, then the algorithm will compute the path of greatest change in elevation between the start and endpoint in which the user can still travel the path within the desired amount of time. We will initially assume the user travels at a fixed speed. We will employ Dijkstra’s algorithm to perform the computation.

Data model: The model will use the Open-Elevation API, which can return the elevation on the earth at any point given a set of input coordinates. The model will contain a data structure that will contain all the coordinate points (populated by the algorithm) to move on sequentially along the computed path.

UI: The user interface will display a 2d map of the entire area being traversed and display a line from the startpoint to the endpoint based on the results from the algorithm which will be retrieved from the data model. The interface will include inputs for start and end coordinates, an option to either minimize or maximize elevation, a time input when maximizing, and an option to reset the map and calculate a new route.

3. **The Evaluation**.

Algorithm testing: We will construct a graph with known routes to use for testing where each edge represents a change in elevation and each node represents current elevation; each edge can also include time for testing maximized elevation.

With this known graph, we can also test the data model to make sure all the points on the correct path are populated and in the right order.

The UI can also use this graph for testing its display, make sure it displays some sort of data (even if the data is inaccurate)

**Timeline notes:**

Nov 25: Have basic components realized: algorithm can do just elevation minimization (don’t worry about distance/time constraints or maximizing elevation). UI displays a window with basic user input for start/end coordinates, displays data such as elevation change, distance covered, estimated time; buttons for reset, close/restart, etc. data model at this point either doesn’t worry about obstructions at all (i.e. only concerned with raw gps and elevation data), or it only works off of streets; no offroading.