Vivien Jamba
Aryan Dang
Aaryaman Mehta
Rahul Avadhanula

ELeNA (Elevation-based navigation system)

Software Requirements Specification (SRS)

1. Overview

EleNa is a map-based web application that enables users to decide routes between geographical locations with the inclusion of elevation as a factor. The application will have a 2D visualization of the map, in addition to labels to indicate the names of landmarks, areas, states, etc. The user may maximize and minimize elevation for the route chosen based on their preferences.

The primary focus for this project is to implement an interactable, accessible UI for a valuable application, so as to provide the user with a pleasant, confusion-free experience when navigating the interface. In this case, the stakeholders in this project will be purely the clients using the web application as well as the software developers creating it. Examples of such clients include anyone who needs to plan a route based on elevation preferences, such as hikers, bikers, runners, or any outdoor enthusiasts. It could also be useful for urban planners or architects who need to consider elevation when designing routes or structures. Additionally, this software could be used by transportation companies or logistics companies who need to optimize their routes based on elevation preferences to save time and reduce costs.

2. List of features

- Using OpenStreetMap’s API for autocomplete functionality, the user would input the origin and destination addresses.
- The user has several options to choose from when calculating the route from the origin and the destination. They may select the shortest path with no elevation. The user may also input a percentage value of the shortest path, essentially an “offset” value, that they are willing to travel extra, and can select whether to maximize or minimize elevation. The route will be displayed depending on the user’s preferences.
- The UI will display information about the chosen route, such as the elevation gain, elevation drop, distance, and approximate time taken to travel.
- Using OpenStreetMap’s API, the user will be able to interact with the map, such as viewing the area surrounding the route, zooming in/out, panning, etc.

3. Functional Requirements

- **Address input and autocomplete functionality:** The system must use OpenStreetMap's API to provide an autocomplete functionality for address input, allowing the user to input the origin and destination addresses.
- **Route calculation and options selection:** The system must allow the user to select from several options when calculating the route, including selecting the shortest path with no elevation, inputting a percentage value of the shortest path that they are willing to travel extra, and selecting whether to maximize or minimize elevation.
- **Route display:** The system must display the chosen route based on the user's preferences, including showing the elevation gain, elevation drop, distance, and approximate time taken to travel.
- **Shortest path algorithm:** The system must use GeoJSON formatted nodes in combination with Dijksta's and A* algorithms to calculate the shortest route with the elevation preferences from the user.
- **Map interaction:** The system must use OpenStreetMap's API to allow the user to interact with the map, such as viewing the area surrounding the route, zooming in/out, panning, etc.
- **Integration with OpenStreetMap's API:** The system must integrate with OpenStreetMap's API to access the necessary elevation and route data.
- **Data accuracy:** The system must ensure that the elevation and route data provided by OpenStreetMap's API is accurate and up-to-date.MVC design pattern: The system must follow the Model-View-Controller (MVC) design pattern architecture, with the view existing on React.JS, the model

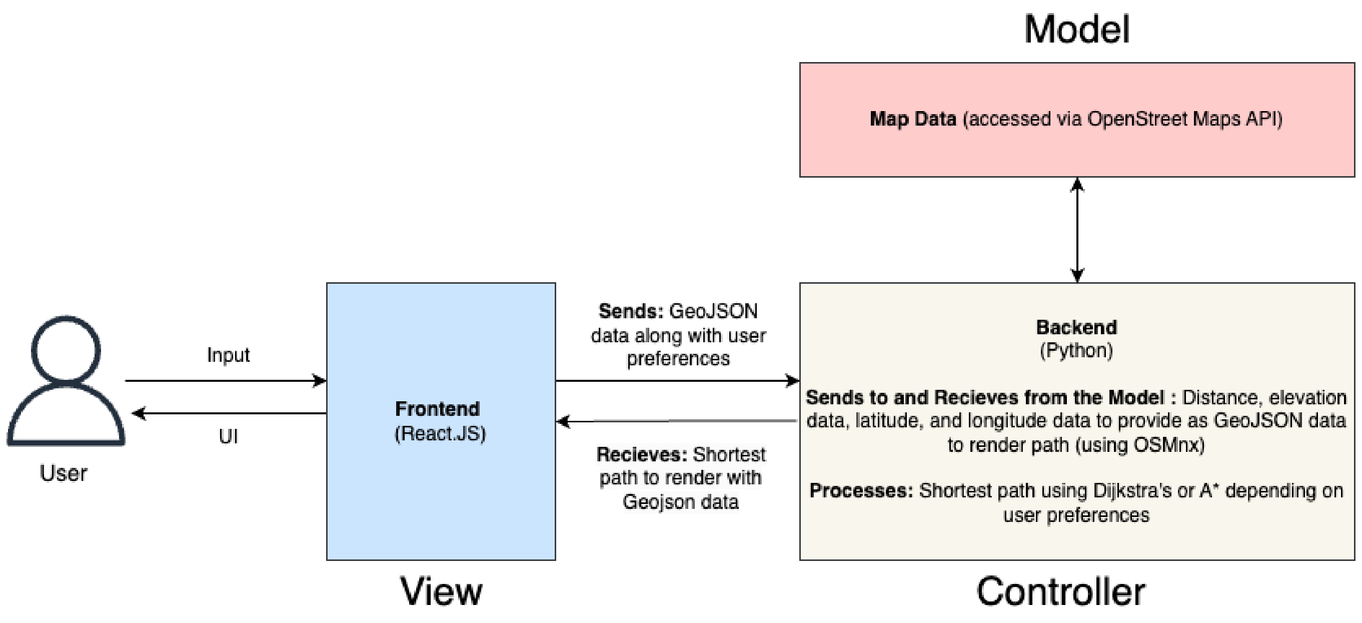
on a database accessed through OpenStreetMap's API, and the controller through a backend Python file.

4. Non-functional Requirements

- **Understandability:** The system must be easy to understand, with clear documentation and well-organized code. The system should be designed to allow for future improvements and enhancements. This includes having a README file to understand how to run the application, aptly-named functions and files, and appropriate codebase organization.
- **Debuggability:** The system must log and report any errors that occur during usage to allow for quick resolution and maintenance.
- **Performance:** The system must be able to calculate routes and display them in a timely manner, even when the user selects complex options, such as routes that maximize or minimize elevation. The system should be optimized for efficiency and speed.
- **Usability:** The system must be easy to use and intuitive for all types of users, including those with little technical expertise. The interface should be clear and well-designed, with easy-to-understand icons and labels, striving to follow Schneiderman's 8 Golden Rules.
- **Reliability:** The system must be reliable and accurate, providing correct route information to the user at all times. The system should be able to handle large volumes of user requests without crashing or slowing down.

Design

1. Architecture


Model
Map Data (accessed via OpenStreet Maps API)
User
Input
UI
Frontend (React.JS)
View
Sends: GeoJSON data along with user preferences
Recieves: Shortest path to render with Geojson data
Backend (Python)
Sends to and Recieves from the Model : Distance, elevation data, latitude, and longitude data to provide as GeoJSON data to render path (using OSMnx)
Processes: Shortest path using Dijkstra's or A* depending on user preferences
Controller

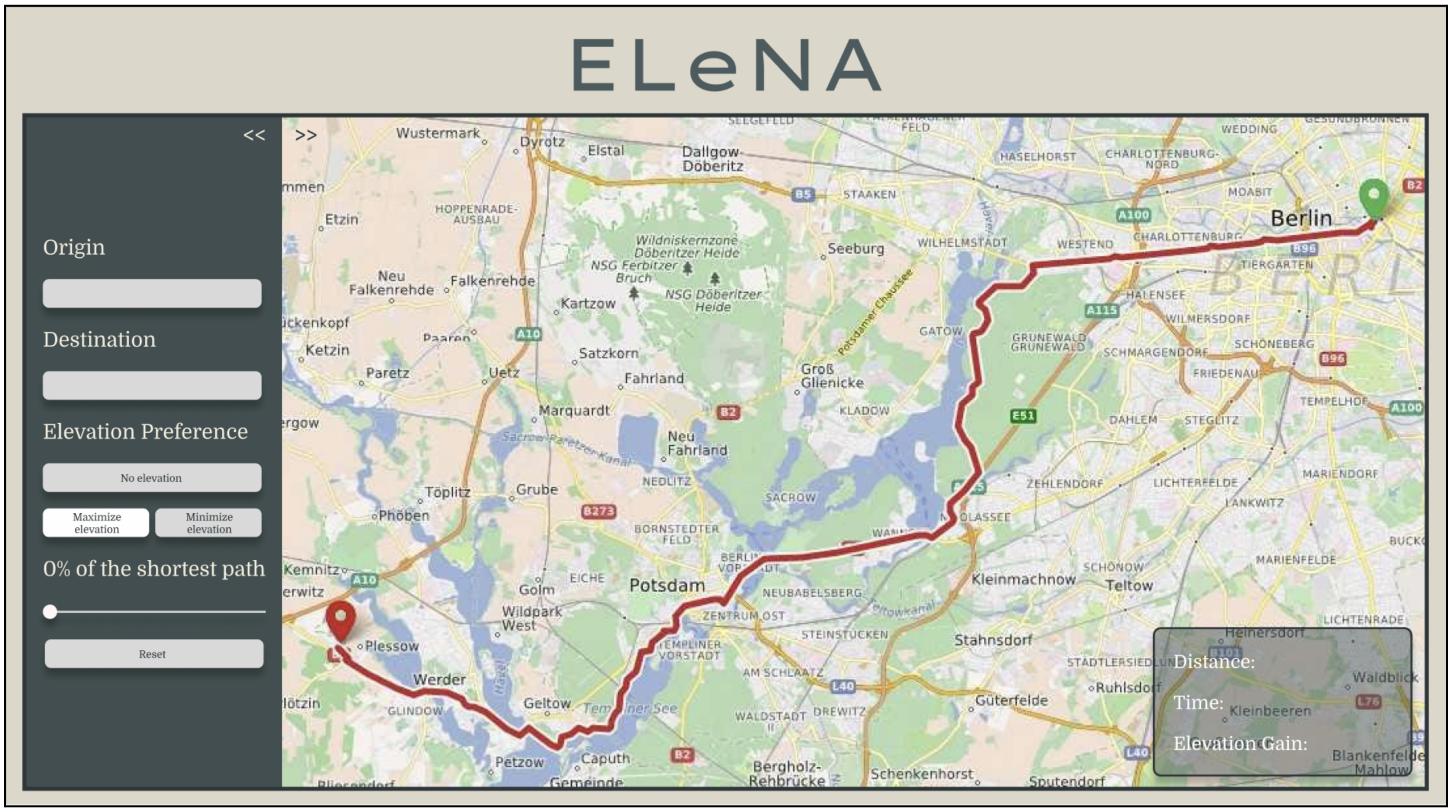
2. Tech stack

Front-end: ReactJS - ReactJS is a popular and powerful front-end JavaScript library that enables the creation of complex user interfaces. It has a large developer community, great documentation, and a large ecosystem of libraries and tools that can make development faster and more efficient. For the EleNa app, ReactJS would be ideal for creating an interactive, responsive, and user-friendly UI that can handle a variety of input types and display information in real-time.

Back-end: Python - Python is a high-level, general-purpose programming language that is widely used in web development, data science, machine learning, and other domains. It is known for its simplicity, readability, and ease of use. For the EleNa app, Python would be an excellent choice for the back-end because it is fast, efficient, and has a large number of libraries and frameworks that can handle various tasks such as web scraping, data manipulation, and database management. Additionally, Python has great support for APIs and can handle the interaction with the OpenStreetMap API with ease. Particularly, we intend to use OSMnx, which is a Python library that can be used to download, analyze, and visualize OpenStreetMap data. It can be used to download and preprocess the necessary data from OpenStreetMap, such as the elevation data and road network information, and perform additional analysis on the data as needed.

Mapping and routing: OpenStreetMap and Leaflet - OpenStreetMap is a collaborative project that provides free and editable maps of the world. It is a great alternative to proprietary mapping services like Google Maps and has a vast community of contributors who help to maintain and improve the data. For the EleNa app, OpenStreetMap would be a great choice for providing the base maps and routing data. Leaflet is an open-source JavaScript library that can be used to create interactive maps on the web. It works well with OpenStreetMap data and has a large number of plugins and extensions that can be used to enhance the map's functionality and appearance.

3. UI design mockup



Evaluation Plan

1. Evaluating functional requirements

Alpha/Beta Testing: We will be collecting feedback/bug reports on our product with friends/family in the alpha phase, and throughout campus in the beta phase. This will allow any issues or bugs to be identified and resolved before the final release.

Unit Test Suites: Unit tests can be created to ensure that each function performs as expected and that the code modules are integrated correctly. This will involve testing each of the functions that perform different tasks in the system, including the address input and autocomplete functionality, route calculation, route display, shortest path algorithm, and map interaction. Our team will focus on passing core unit tests including:

- tests which confirm our intended functionality (for instance, the appropriate route is recommended)
- edge case testing (for example, if the user wants no change in elevation, we must gracefully let them know this is impossible)

- unit testing for different areas of the world
- code coverage tests

2. Evaluating nonfunctional requirements

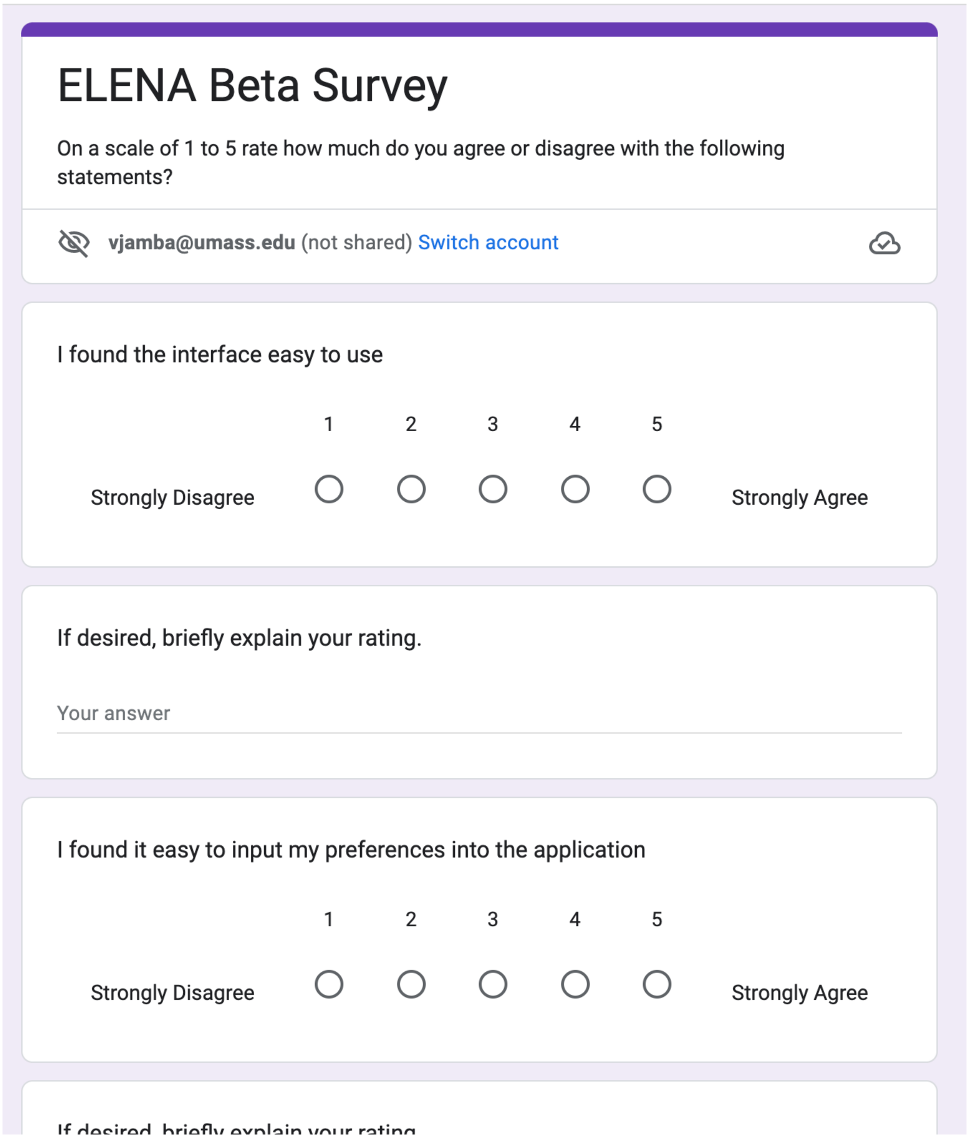
Alpha/Beta Testing: Given our priority on accessibility, Alpha/Beta testing can also be used to test the usability of the system. This will allow users to provide feedback on the system's ease of use, intuitive interface, and clear labels and icons.

Performance Testing: Performance testing can be done to check if the system can handle a large volume of user requests without slowing down or crashing. This will involve testing the system with a large number of concurrent users and checking if the system can still calculate routes and display them in a timely manner.

A/B (Split) Testing: During our beta phase, we will be collecting feedback on various versions of our UI using A/B testing. Some cases we hope to use A/B test for are:

- sliding scale elevation input vs multiple choice
- display layout changes (i.e. full screen vs multiple displays)
- aesthetic changes (colors, shapes)

Likert Scale Surveys: During our beta phase, we will be collecting UI/UX feedback through modified Likert scale surveys as shown below:



ELENA Beta Survey

On a scale of 1 to 5 rate how much do you agree or disagree with the following statements?

vjamba@umass.edu (not shared) Switch account

I found the interface easy to use

	1	2	3	4	5	
Strongly Disagree	○	○	○	○	○	Strongly Agree

If desired, briefly explain your rating.

Your answer

I found it easy to input my preferences into the application

	1	2	3	4	5	
Strongly Disagree	○	○	○	○	○	Strongly Agree

If desired, briefly explain your rating.