Femimol Joseph, Kritika Kapoor, Prathiksha Rumale Vishwanath, Varun Tarikere Shankarappa
COMPSCI 520

EleNA Final Report

NOTE: SRS, Design Documentation and Evaluation are included in this final report. The recording link on google drive and GitHub link for the project are attached at the end of the final report.

1. OVERVIEW

Proposed Idea

Routing apps help to show routes from one place to the other, based on distance and time as deciding factors. But there should be the facility to customize route selection based on elevation gain for use cases such as hiking and intensive workouts.

For this we propose **EleNA**, an Elevation-based Navigation system. The project's goal is to create a software system that, given a start and an end location, determines a route that maximizes or minimizes elevation gain while limiting the total distance between the two locations to x% of the shortest path.

We will be developing a web application that will allow the user to enter source and destination, and the application will render visual routes on the map which will be customizable according to max or min elevation gain.

Stakeholders

- Users: They are the primary stakeholders who use the app to find directions, search for places, and get information about their surroundings. Our population of interest would be bikers, riders and hikers.
- Developers: The developers who create the app and continually work on improving its features and functionality.
- Investors: Investors who have invested in the development and growth of the app.

2. SOFTWARE REQUIREMENT SPECIFICATIONS (SRS)

List of features

- Interactive maps-based interface: The interactive app allows the user to enter the source and destination address with autocomplete functionality added for address completion.
- Multiple filters to select optimum route: Users have the choice to either select minimum or maximum elevation gain and adjust percentage increase from shortest path.
- Multiple modes of transport: The user will be able to choose between various modes of transport: walking, biking and driving.
- Route metrics: The app will return logistics of the optimal route selected with features like the distance travelled, time taken to travel the path and the elevation gain between the source and destination locations.
- Fitness tracker: The app will also show the calories burnt based on the mode of transport for walking and biking, taking other constraints set by the user into the consideration.
- Route rendering: Visual information is provided by rendering routes on the maps, which shows the optimal routes based on the conditions set.

Functional Requirements

- Search functionality: The application provides a search functionality that allows users to search for locations by entering a name, address, or point of interest.
- Geolocation: The app should be able to determine the user's location using GPS or other methods.
- Map rendering: The app should be able to render routes quickly and accurately, with features like zooming and panning.
- Route metrics: Once the optimum route selected, the statistics related to the selected route must be displayed for user reference.
- Navigation functionality: The application must provide navigation functionality that allows users to obtain the path from a specified source to a specified destination.

Non-functional Requirements

- Performance: The application must be responsive and provide fast performance, even in areas with slow internet connectivity.
- User Interface: The application must adhere to guidelines for a consistent user experience. The user interface must be intuitive and easy to navigate.
- Compatibility: The app should be compatible with a wide range of devices, operating systems, and internet browsers.
- Maintainability: The app should be easy to maintain and update, with a modular design and well-documented codebase.
- Reliability: The app should be always available and not crash or freeze during use.

3. DESIGN DOCUMENTATION

Architecture

EleNA is based on MVC architecture represented by the below diagram:

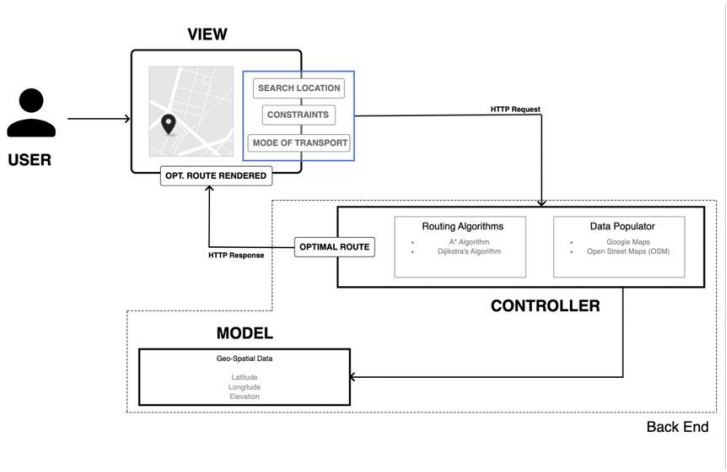


Fig. 1: Architecture

Major components of EleNA architecture are described below:

- Model – Our model will be populated with geodata, i.e, latitude, longitude and elevation related data which is required to render routes and the maps-based UI.
- Controller – Multiple components will make up the controller, where the logic of the application resides.
    a) Model population – We will make use of OSM and GMaps APIs to populate our model.
    b) Routing Algorithm - Using the inputs provided by the user (starting point and ending point, constraints like min/max elevation gain and percentage increase from shortest distance, travelling mode), the routing algorithm will generate the optimal route by extracting necessary information from the model.
    c) Route Rendering component – This component will be responsible for rendering the route and highlighting the path on the UI map screen.
- View – The view consists of a form and maps-based UI. It has the following components –
    a) Interactive maps screen – Allows the user to zoom in and zoom out of the locations on the map manually and renders the shortest path with specified user constraints.
    b) Address box– Allows the user to search and enter start and end locations with address autocomplete functionality.
    c) Percentage Increase constraint – Allows the user to adjust the percentage increase from shortest path, as a trade-off between elevation gain and distance.
    d) Elevation based constraint – Allows the user to choose maximum or minimum elevation gain based on the type of route needed.
    e) Mode of Transportation – Allows the user to choose between various modes of transport based on which the route information will be rendered.

<u>Tech Stack Justification</u>

EleNA is segmented into 2 layers of the stack.

- Front-end:

Frontend is browser-based application that is accessed by user to input location details such as source and destination address, percentage increase in shortest path that is acceptable, minimum, or maximum elevation gain, and the mode of transport for which the route needs to be rendered. The front-end code is written using HTML, CSS and ReactJS.

Front-end is a react web-app hosted on http://127.0.0.1:3000.
It gives the user the following choices to generate a path:

1. Source
2. Destination
3. Percentage increase from shortest path
4. Elevation gain (max or min)
5. Mode of transport

The input params are formatted as a JSON object and sent as body to the backend POST api (/get-route).
Repone consists of the following attributes:

1. The "path" key returns a list of coordinates comprising the optimum path that are rendered on the map.
2. Other details such as the distance, elevation gain form the shortest path, time required to travel between the source and destination are also returned from the response which are displayed in a tabular format within the form.

- Back-end:

Backend server is REST-based web server hosting both the routing algorithms Dijkstra and A*. It is written in Python using Flask framework which uses MVC Pattern as the application architecture. Backend, with reference to our design document, consists of the Model and the Controller.

- Controller:
  Controller is a flask REST application hosted on http://127.0.0.1:3003. It consists of the following API calls:
  1. /get-route, POST
     Body: Needs the params to run the algorithm properly – source, destination, % increase, elevation_gain, mode
     Response types: 200,success and 400,error
     Reponse_object: path: list of coordinates, distance
  2. /get-place, GET
     Query-params: text to be autocompleted
     Response types: 200 and 400
- Model:
  Model is a data model that consists of the Geo-Spatial data i.e. details of latitude, longitude and elevation. It is populated by an abstraction layer with third party geospatial data providers like Google maps API and OpenStreetMap which will be used by the backend server.

Design Decisions:

Design decisions that were considered during the development process are described below:

- Algorithms for calculating the optimum route: To develop the elevation-based navigation system, Dijkstra and A* are the two algorithms that were implemented due to their ability to find the shortest or the most efficient paths between the two locations, considering the elevation differences. The paths are obtained from each algorithm is compared with the elevation gain and percentage increase constraints provided by the user. The algorithm that returns the route which closely satisfies the user constraints is rendered on the map component in the UI.
- Flask Server: Although Django framework was initially explored, adopting Flask proved to be a superior option once it was realized that the model in our application does not store general data in object format but stores graphs due to the domain specific requirement. Additionally, Flask interfaces easily with a broad variety of python tools and libraries, which enabled us to use pre-existing geospatial, graph, or elevation-related frameworks.

- Open Topo API for calculating elevation gain: Open Topo API can be easily integrated, and it offers a simple interface for sending and receiving structured HTTP requests and elevation data, such as JSON. Because of this accessibility, it is simple to include the elevation gain computation that is required in the application.
- OSMnx library for map generation: Utilizing OSMnx, map data such as roads, paths, buildings, landmarks including elevation information can extracted and retrieved from the OpenStreetMap. The generated graphs can be used for network analysis and routing tasks efficiently.
- Caching methodology: Saving the generated graphs as .pkl files would assist to reduce the amount of space occupied by the large graphs. These take up a lot of storage space hence caching is implemented.

UI Design:

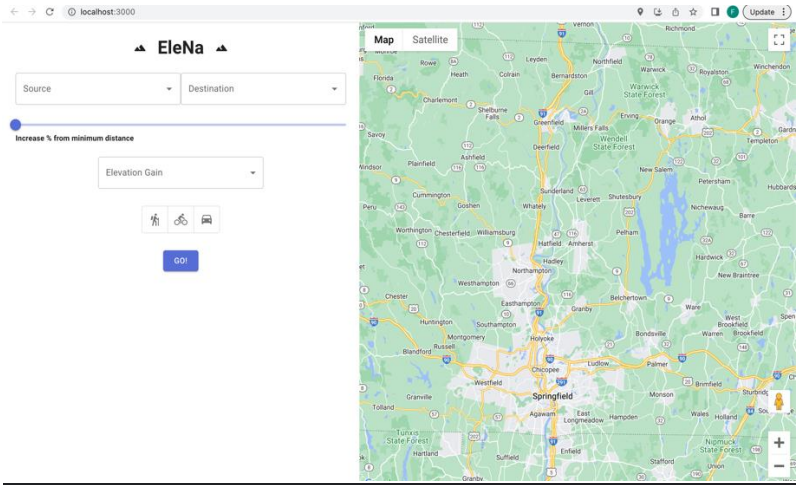


Fig. 2: Initial UI.

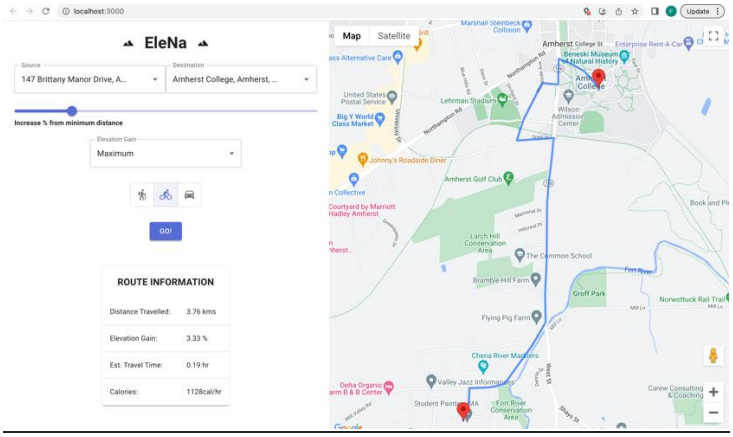


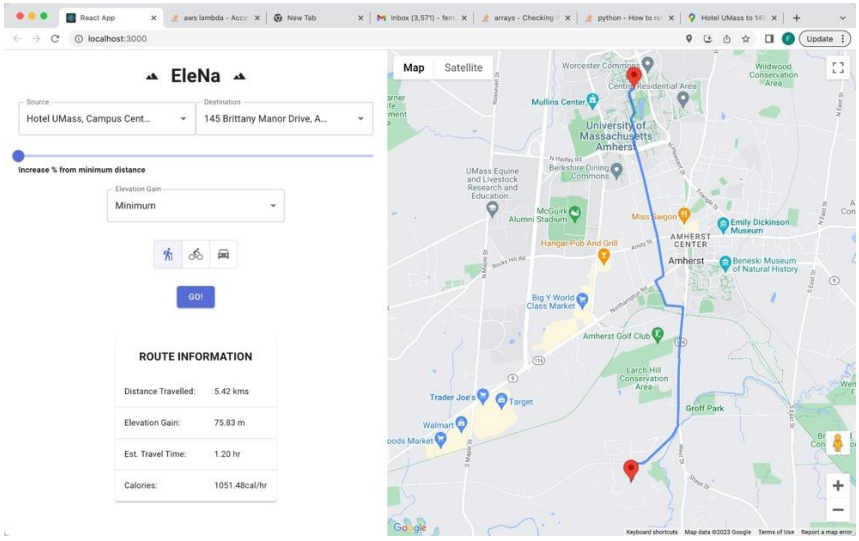
Fig. 3: UI after rendering the optimum path.

4. EVALUATION

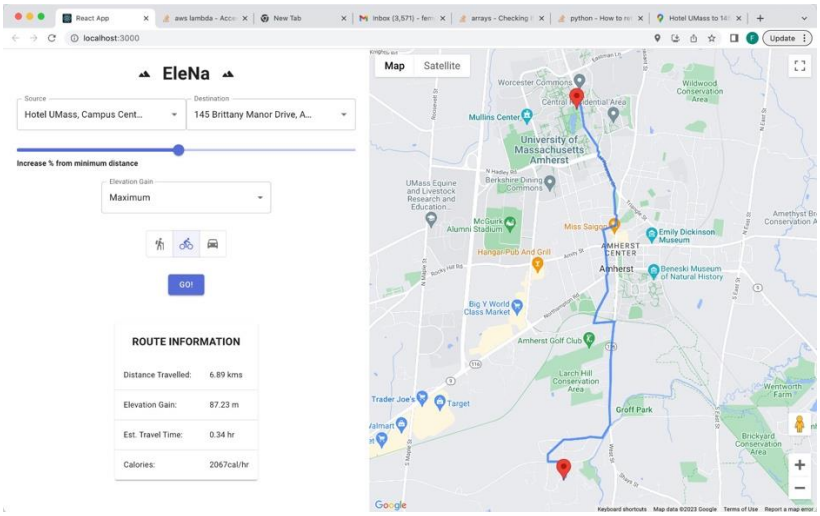
Four evaluation strategies are performed to verify the application and its functionalities:

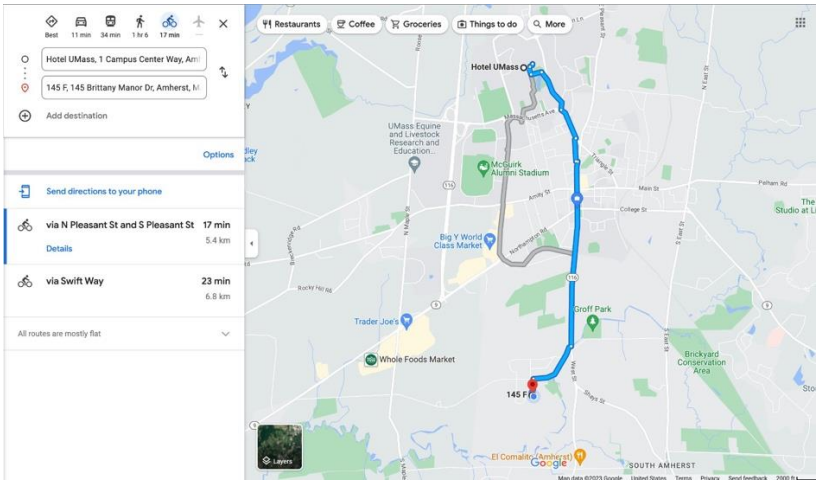
- **Correctness of the algorithm:**
  To validate the correctness of the algorithm the following steps were performed:
  a) We consider a path/distance calculated by Google maps as source of truth.
  b) First, for correctness, we will randomly select N=5 locations including starting point and the destination and calculate the optimal route based on our algorithm.
  c) The path and distance for all the above points will be calculated using google maps API.
  d) The difference between EleNa's output and Googles output are compared based on distance and estimated time for travel.

Walking:

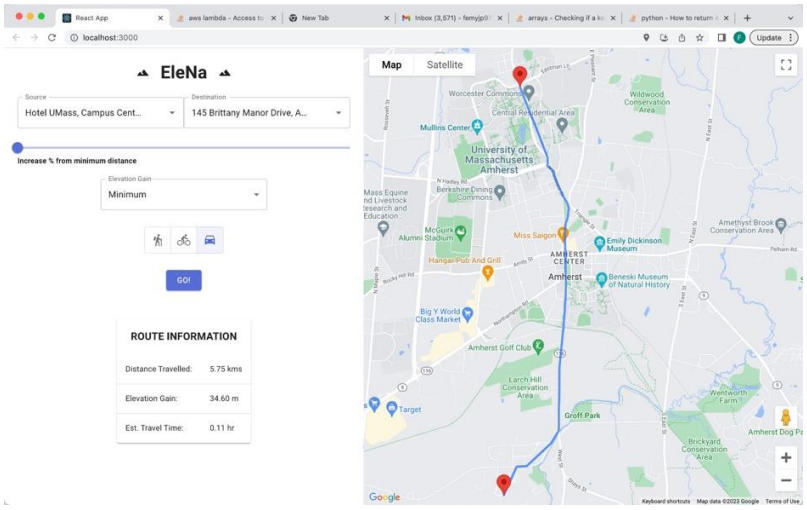


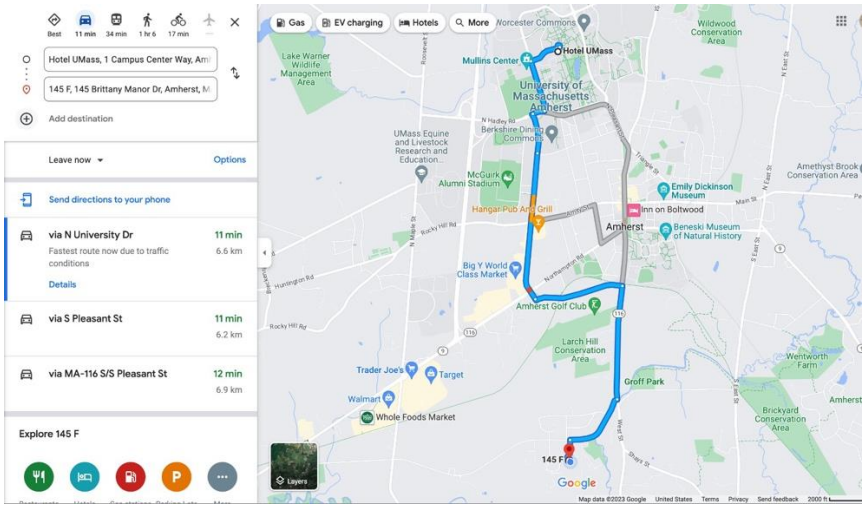
Biking:





Driving:

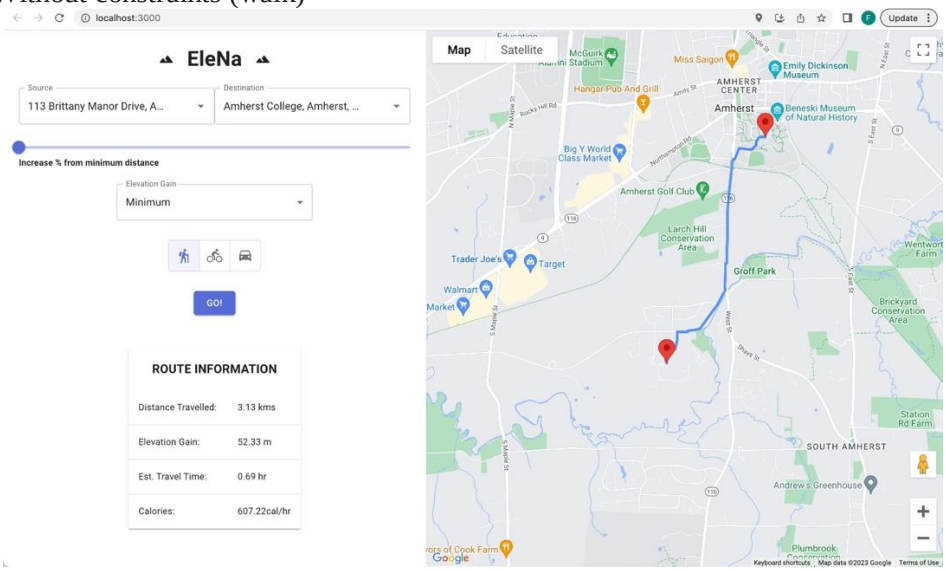




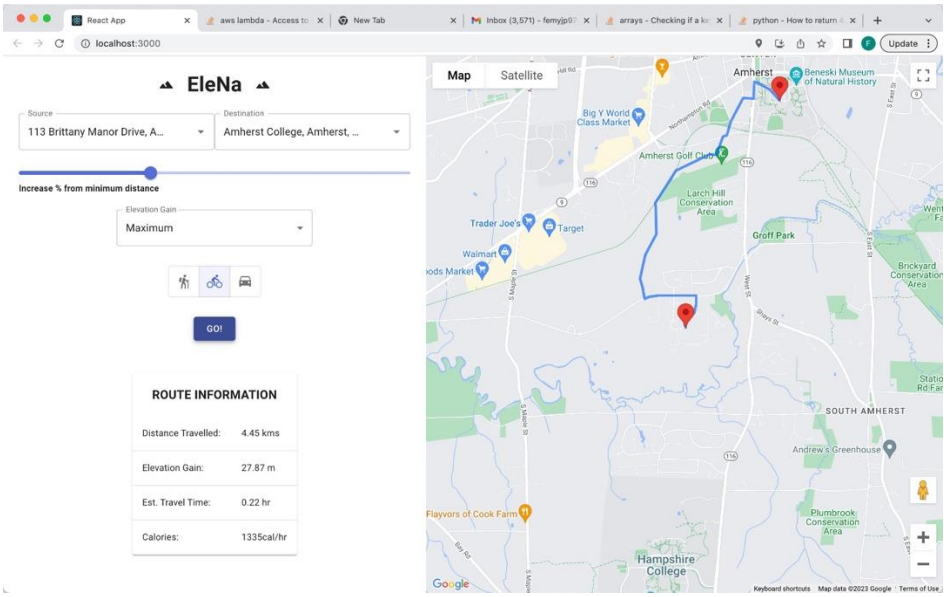
- **Sanity Testing for all modes:**

  We tested all cases of all possible constraints, with the available options. All the cases rendered a route and populated route statistics correctly.

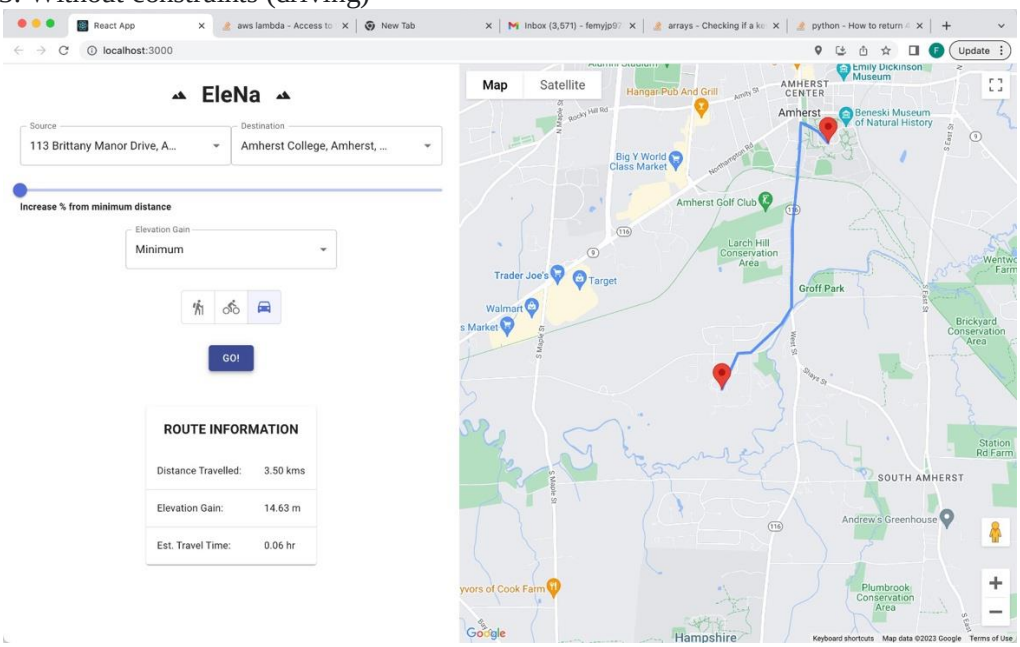
- Case 1: Without constraints (walk)



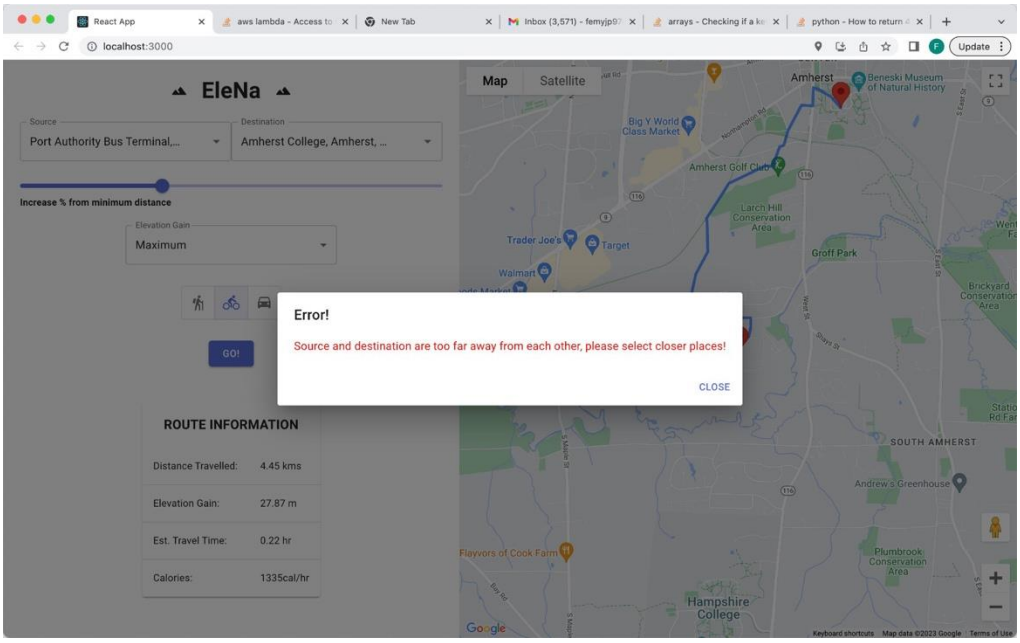
- Case 2: With constraints (biking)



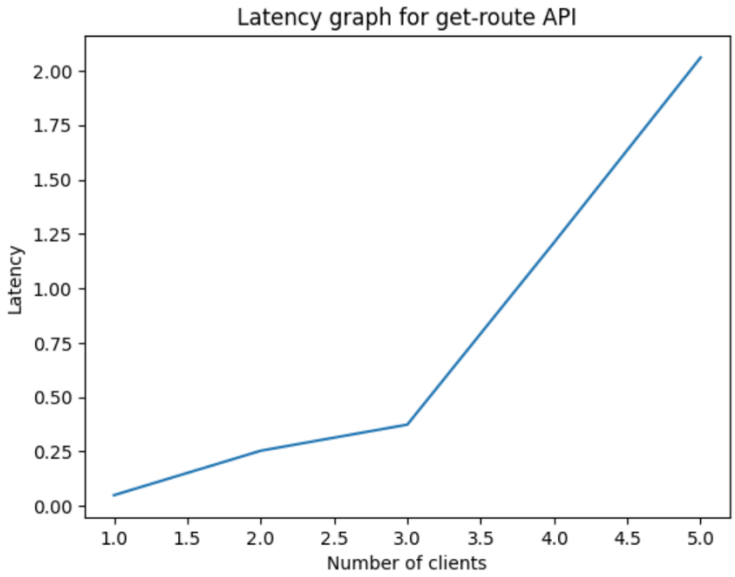
- Case 3: Without constraints (driving)



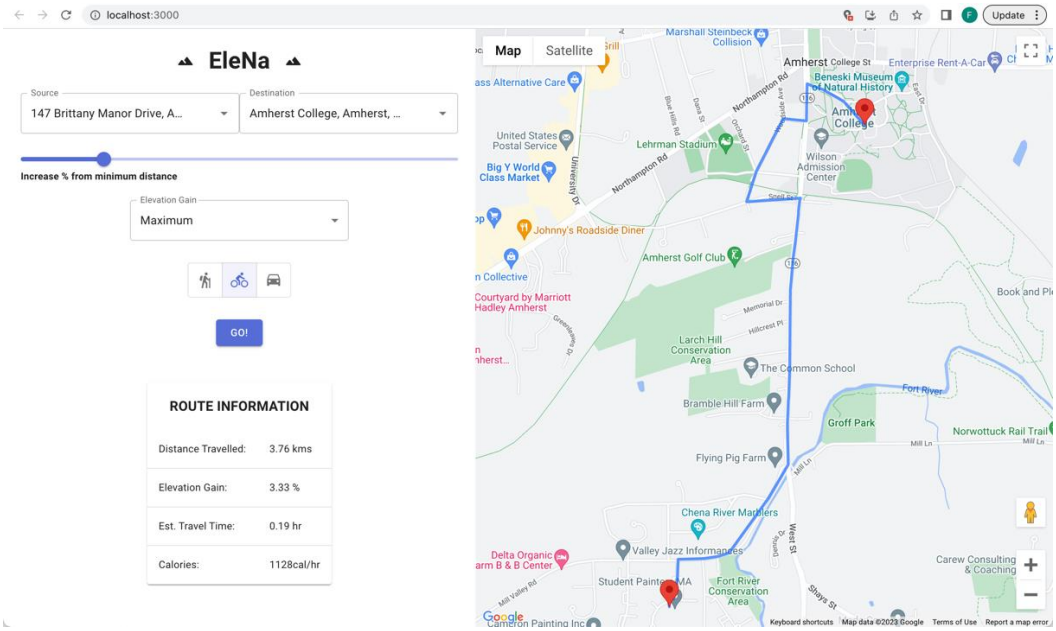
- As a Limitation, when the source and destination are far apart, the following error message is displayed in the UI.

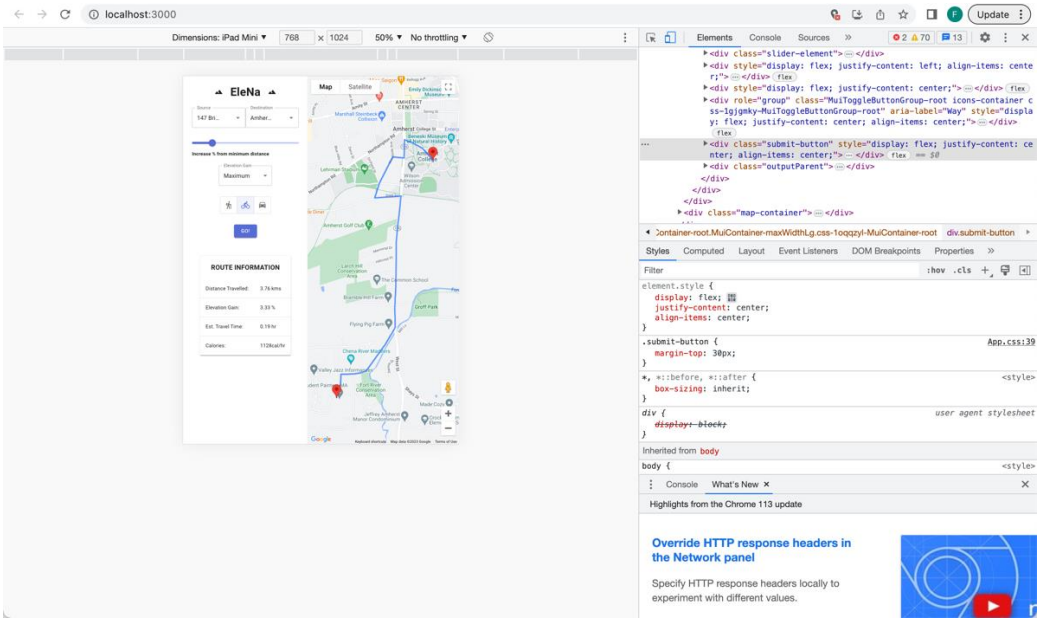


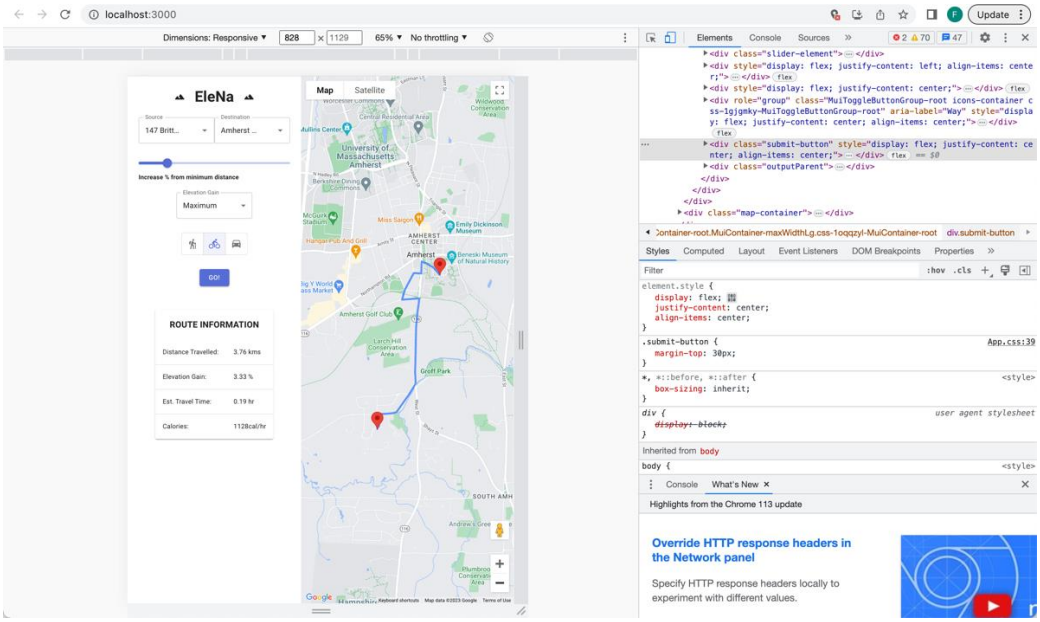
- **Performance testing:**
  To validate performance testing of the backend server, a latency graph is plotted w.r.t number of clients currently using the application. As expected, the latency increases as the number of users using the application increase.



- **Responsive UI validation:** The web application was tested across multiple resolutions to test the responsive UI feature which was implemented in the code. Both the form and the map components were responsive to varying dimensions of desktop, varying resolution and ipad mini.


localhost:3000
Update
EleNa
Source
147 Brittany Manor Drive, A...
Destination
Amherst College, Amherst, ...
Increase % from minimum distance
Elevation Gain
Maximum
GO!
ROUTE INFORMATION
Distance Travelled: 3.76 kms
Elevation Gain: 3.33 %
Est. Travel Time: 0.19 hr
Calories: 1128cal/hr
Map
Satellite
Amherst College
Lehrman Stadium
Johnny's Roadside Diner
Amherst Golf Club
Courtyard by Marriott Hadley Amherst
Larch Hill Conservation Area
The Common School
Bramble Hill Farm
Groff Park
Flying Pig Farm
Chena River Madders
Valley Jazz Informances
Fort River Conservation Area
Norwottuck Rail Trail
Google


localhost:3000
Update
Elements
Console
Sources
Styles
Computed
Layout
Event Listeners
DOM Breakpoints
Properties
Override HTTP response headers in the Network panel
Specify HTTP response headers locally to experiment with different values.



- **Test suites and coverage reports**

Test Suite Report for Frontend:

```
PASS src/App.test.js
PASS src/__tests__/Form.test.js

Test Suites: 5 passed, 5 total
Tests:       5 passed, 5 total
Snapshots:   5 passed, 5 total
Time:        1.35 s
Ran all test suites.
```

Unit Test Suite Report for Backend:

```
----------------------------------------------------------------------
Ran 14 tests in 0.362s

OK
```

Coverage Report for Backend Test Suite:

Coverage report: 98%

coverage.py v7.2.5, created at 2023-05-22 22:01 -0400

Module	statements	missing	excluded	coverage
/Users/femimoljoseph/IdeaProjects/TSE-520/Elena/CS_520-Elena/backend/controller/__init__.py	0	0	0	100%
/Users/femimoljoseph/IdeaProjects/TSE-520/Elena/CS_520-Elena/backend/controller/a_star.py	42	0	0	100%
/Users/femimoljoseph/IdeaProjects/TSE-520/Elena/CS_520-Elena/backend/controller/djikistra.py	90	6	0	93%
/Users/femimoljoseph/IdeaProjects/TSE-520/Elena/CS_520-Elena/backend/controller/utils.py	39	1	0	97%
backend_test.py	193	0	0	100%
Total	**364**	**7**	**0**	**98%**

coverage.py v7.2.5, created at 2023-05-22 22:01 -0400

LINK TO RECORDING: https://drive.google.com/drive/folders/1sw2gOlAdSy-Pgdlxo_pC-WjdTqOu0d39?usp=sharing

LINK TO GITHUB: https://github.com/varunsdevang/CS_520-Elena