EleNa: Elevation Based Navigation

By Group Kashikoi
Members: Astha Baranwal, Divya Sharma, Kavya Harlalka, Rohan Acharya
Github: https://github.com/kavyaharlalka/kashikoi-elena-navigation

The Problem

Navigation Systems typically provide the shortest path between any two given points. These are not optimal for a lot of scenarios where the user is interested in finding a path which has the least elevation gain. Moreover, some users might be interested in finding a path which has elevation gain so that they can partake in an intense and time-constrained workout. To extend navigation systems to solve the above problems, we have designed a software system that takes two points as input and finds the optimal route between them that maximizes or minimizes elevation gain while limiting the total path distance to n% of the shortest path between these two points.

The Problem

Stakeholders

Runners, bikers, hikers, tourists, fitness enthusiasts, event organizers, researchers, rescue teams.

Non Functional Requirements

Understandability, Readability, Testability, Reliability, Compatibility, Modularity, Extensibility and Usability

Functional Requirements

Map which renders the optimal route for the set of inputs sent by user, apply 5 algorithms, check and validate the inputs, ensure accuracy and performance of the algorithm is optimal etc.

User Stories

1. As the user, I want to consider the elevation gain while planning a route between two locations.
2. As the user, I want to select a location from dropdown.
3. As the user, I want to be able to choose minimize or maximize elevation gain.
4. As the user, I want to be able to limit the path to n% of the shortest path between two points.
5. As the user, I want to be able to choose which algorithm should be used.
6. As the user, I want to be able to choose which transportation mode I prefer.
7. As the user, I want the optimal route to be visible on the map.
8. As the user, I want the distance and elevation gain information for the optimal route to be visible.

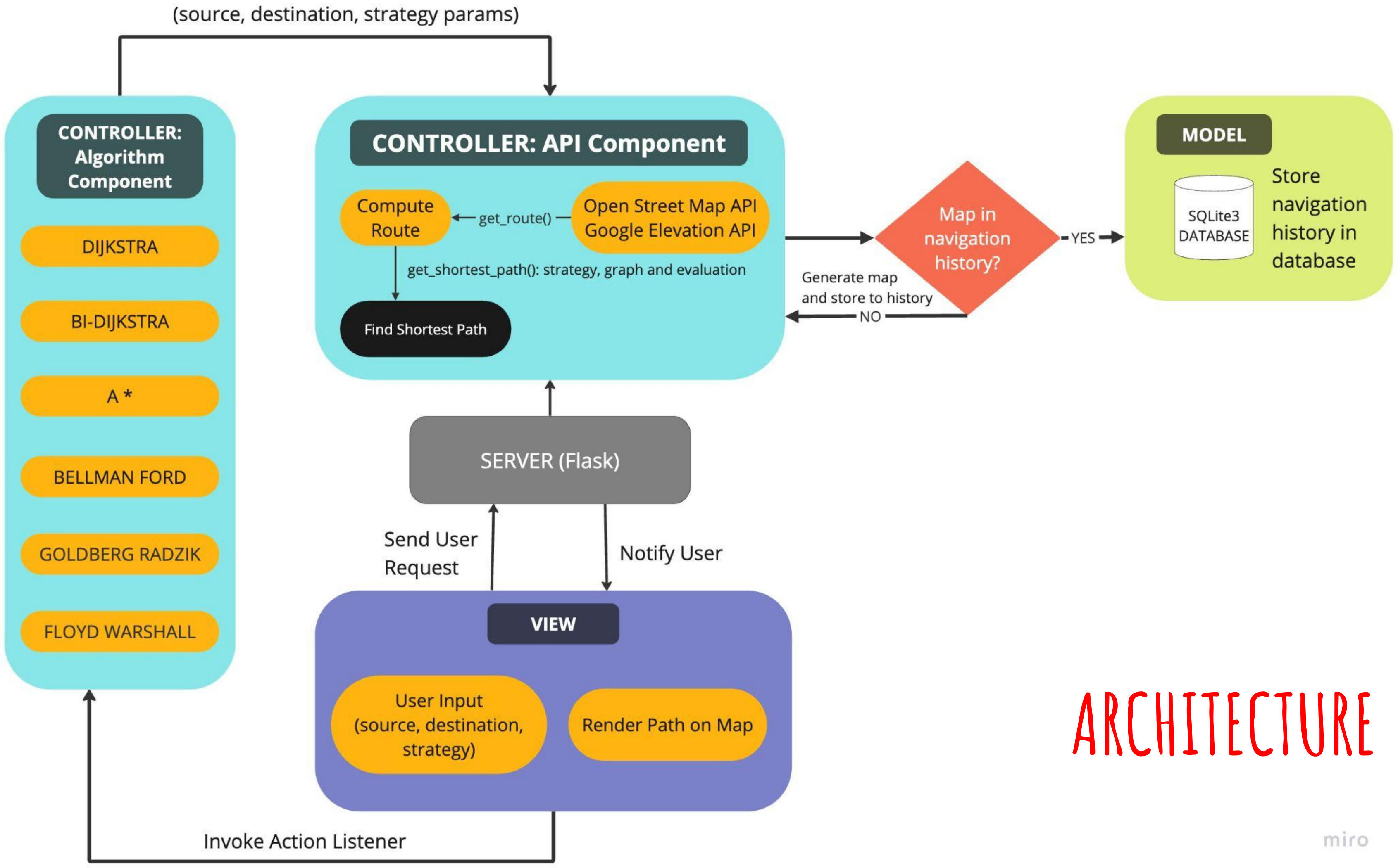
The Design

We built a fully functional system focusing on developing and experimenting with six routing algorithms for computing the route.

Architecture: Model View Controller (MVC)

Tech Stack:

- Python Flask for backend
- HTML, CSS, Javascript for frontend
- Google Maps API to display the final route to the user
- OpenStreetMap API to access the geographic database
- SQLite for storing navigation history

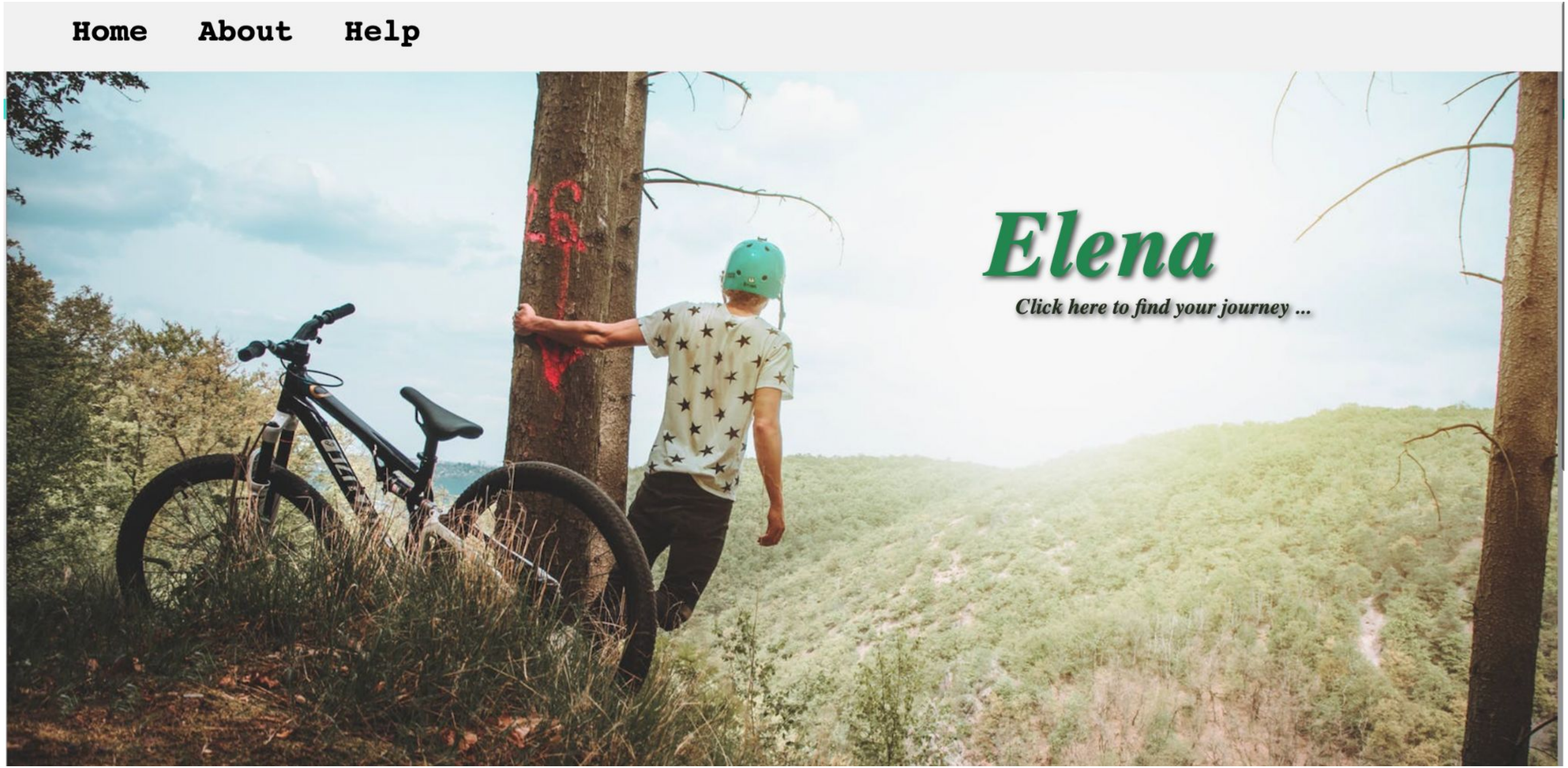

(source, destination, strategy params)
CONTROLLER: Algorithm Component
DIJKSTRA
BI-DIJKSTRA
A *
BELLMAN FORD
GOLDBERG RADZIK
FLOYD WARSHALL
CONTROLLER: API Component
Compute Route
get_route()
Open Street Map API
Google Elevation API
get_shortest_path(): strategy, graph and evaluation
Find Shortest Path
Map in navigation history?
YES
NO
Generate map and store to history
MODEL
SQLite3 DATABASE
Store navigation history in database
SERVER (Flask)
Send User Request
Notify User
VIEW
User Input (source, destination, strategy)
Render Path on Map
Invoke Action Listener
miro

ARCHITECTURE

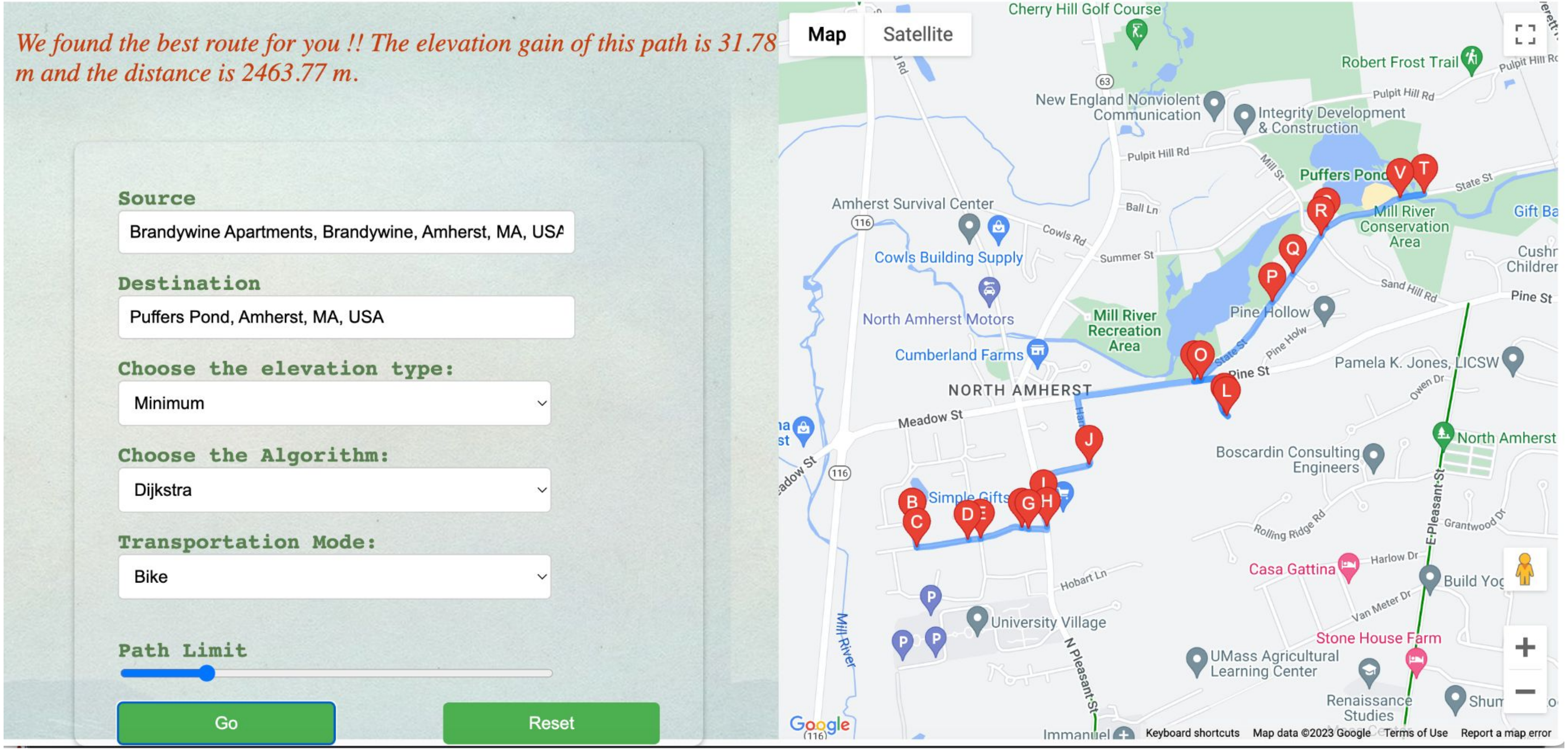
ARCHITECTURE

- View will take the input:
  - Origin and Destination location
  - Strategy corresponding to the minimum or maximum elevation
  - One of the two algorithms (Dijkstra/A*)
  - Path limit (Maximum percentage of the shortest path)
- Controller will call OpenStreetAPI to generate the graph and GoogleMaps Elevation API to populate the nodes with elevation attributes.
- Controller will send the coordinates and strategy to Model.
- Model will store the coordinates and strategy in history.
- Finally, Controller notifies the view to display the computed path.

FRONTEND



FRONTEND



We found the best route for you !! The elevation gain of this path is 31.78 m and the distance is 2463.77 m.

Source

Brandywine Apartments, Brandywine, Amherst, MA, USA

Destination

Puffers Pond, Amherst, MA, USA

Choose the elevation type: Minimum

Choose the Algorithm: Dijkstra

Transportation Mode: Bike

Path Limit

Demo: Code Walkthrough

- We have created an independent API that can be used externally by other users as well for their own applications.
- This API is capable of finding the best and shortest routes and responds with their details.
- We will now walkthrough the code and show the key components of the Controller and Model.
- Then we will use Postman to demonstrate a request sent to our API and its response.

The Evaluation

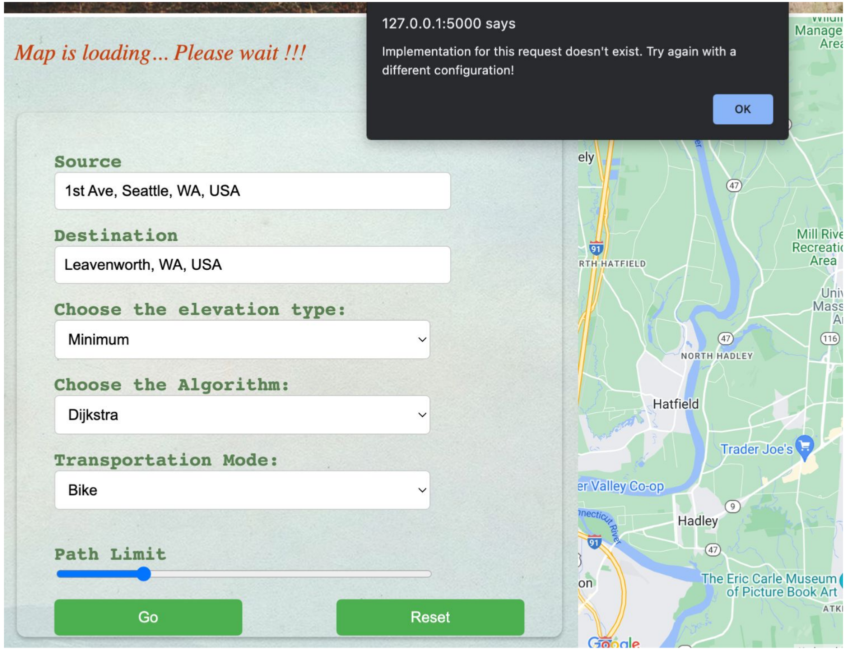
We have done testing to to ensure that all the functions and modules are tested and all corner cases are covered:

1. UI Test
2. Manual Test
3. Performance Test
4. Unit Test
5. Integration/API Test

We achieved a test coverage of 100% (evaluated using pytest-cov)

UI TEST EXAMPLE

MANUAL TEST EXAMPLE



127.0.0.1:5000 says

Implementation for this request doesn't exist. Try again with a different configuration!

Map is loading... Please wait !!!

Source: 1st Ave, Seattle, WA, USA

Destination: Leavenworth, WA, USA

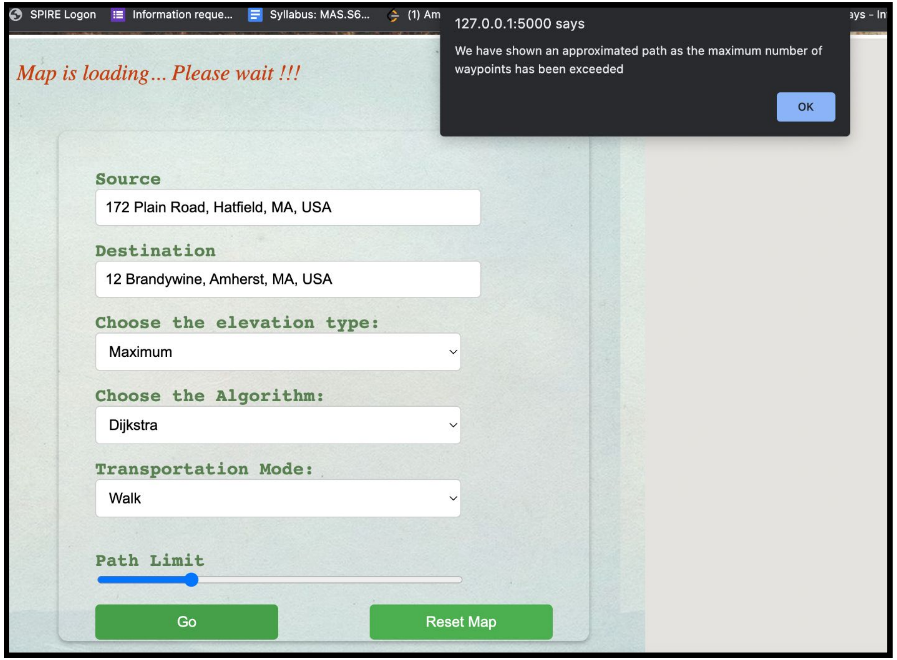
Choose the elevation type: Minimum

Choose the Algorithm: Dijkstra

Transportation Mode: Bike

Path Limit

OK
Go
Reset
NORTH HADLEY
Hatfield
Trader Joe's
Hadley
The Eric Carle Museum of Picture Book Art
Mill River Recreation Area



127.0.0.1:5000 says

We have shown an approximated path as the maximum number of waypoints has been exceeded

Map is loading... Please wait !!!

Source: 172 Plain Road, Hatfield, MA, USA

Destination: 12 Brandywine, Amherst, MA, USA

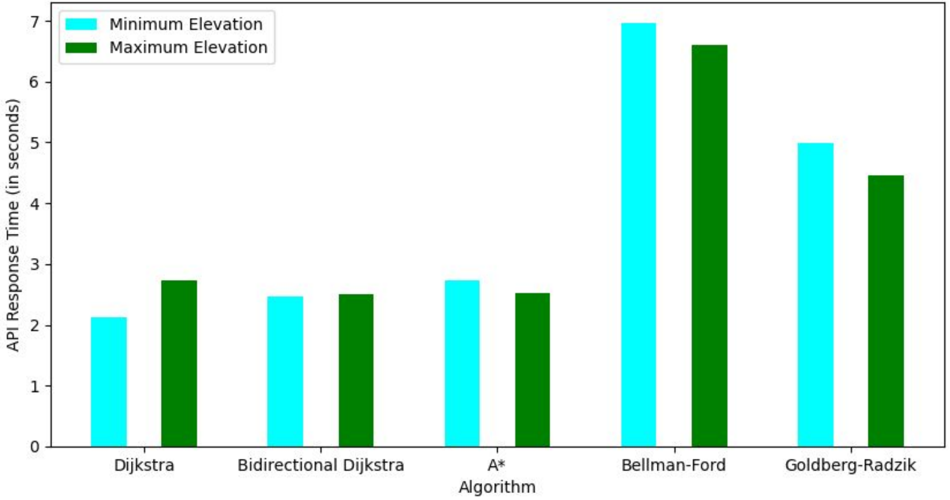
Choose the elevation type: Maximum

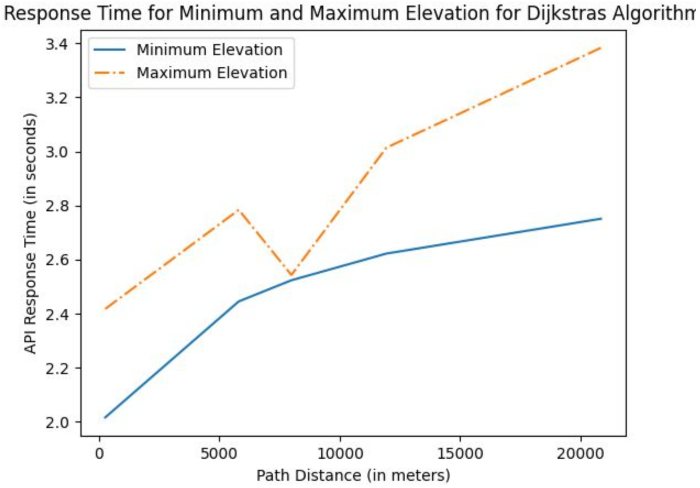
Choose the Algorithm: Dijkstra

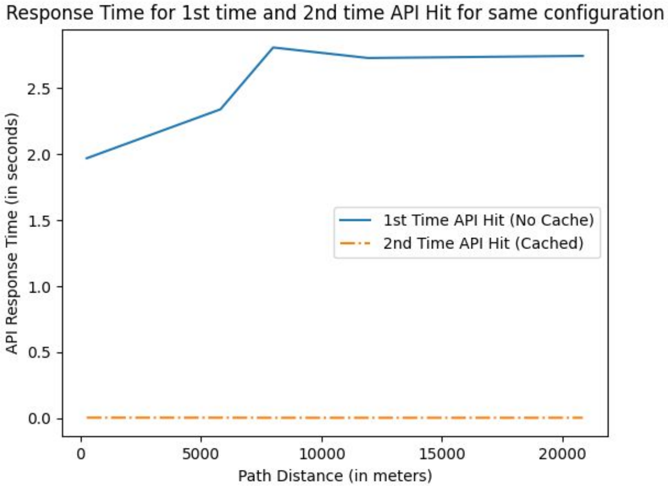
Transportation Mode: Walk

Path Limit

Performance test results




Response Time for Minimum and Maximum Elevation for Dijkstras Algorithm


Response Time for 1st time and 2nd time API Hit for same configuration

UNIT TEST

- Testing with invalid values to ensure error is thrown
- Testing the sql db manager methods to:
    - ensure that insertion and fetch are working
    - throw data check errors in case of invalid data

INTEGRATION TEST

- Illegal Source Address
- Illegal Destination Address
- Null Source Address
- Null Destination Address
- Illegal Algorithm ID
- Illegal Path Percentage
- Illegal Transportation Mode

CONTRIBUTIONS

Group Member Responsibilities

We implemented the entire application using pair programming.

Frontend:
Map component - Divya, Rohan
Input component - Divya, Rohan

Backend:
Algorithms - Kavya, Astha
All other components - Kavya, Astha

Test suites: Everyone

Documentation: Everyone

THANK YOU!