Project: EleNa

Team Name: Redcoders

Members:

- Vijaya Gajanan Buddhavarapu
- Supriya Shreekant Jahagirdar
- Mrinal Tak
- Sirisha Annamraju
- Devyani Varma

Evaluation:

- Functional Requirements:
  - Given a start and end location, determine a route that maximizes or minimizes elevation gain, while limiting the total distance between the two locations to x% of the shortest path.
  - Algorithms: Dijkstra, A-star
  - Display route information, for instance, distance and elevation gain.

- Performance Analysis:
  - We implemented two different algorithms namely Dijkstra and A-star.
  - **Dijkstra is a special case of A star where we set the heuristic value to 0.**
  - **A star finds the shorter path faster than Dijkstra** as we use the heuristic of the distance till the destination node to reduce the time complexity.
  - But **Dijkstra is guaranteed to find the shortest path**.
  - To find the best heuristic value, we use binary search algorithm. Here we find the best scaling factor which is used to construct the weighted edges of the graph using below equation:

    ```
    edge_weight = scaling_factor * edge_length + elevation_gain.
    ```

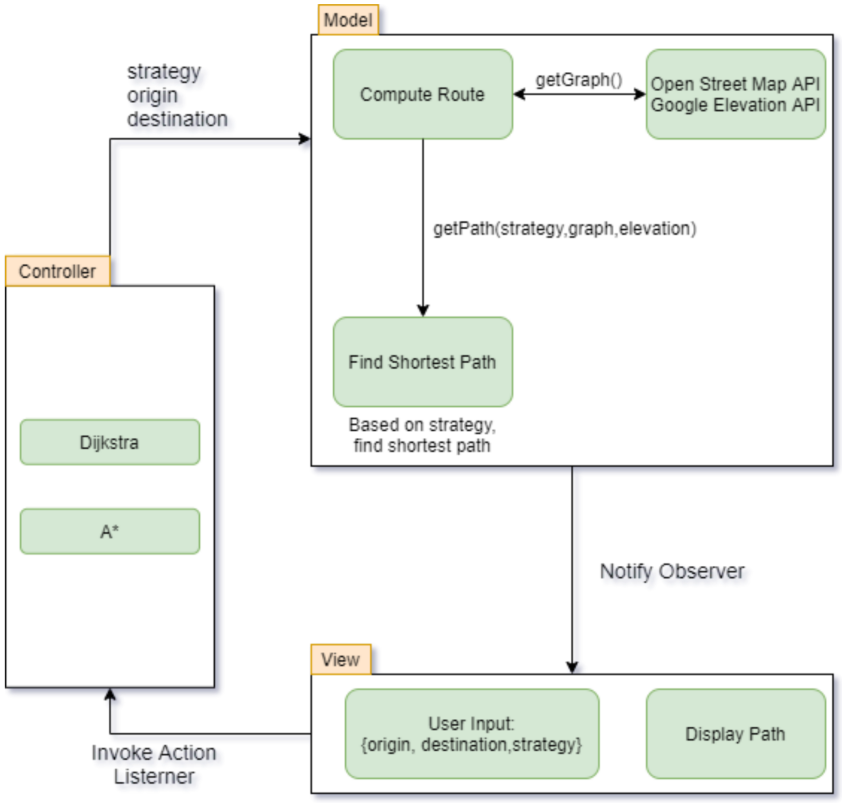
- Usability:
  - The current implementation is valid for **Amherst, Massachusetts** as the geographical area.
  - The implementation ensures usability. For instance,
    - User input: Ability to mark points on map or input address using text box.
    - Output: Shortest route as well as elevation considered route are displayed.
    - Reset: Set application to default configurations.

- Extensibility:
  - The use of **Strategy design pattern** makes it easy to extend the implementation to more algorithms.
  - The geographic region can be extended to wider regions by changing the radius considered.
  - The use of **Observer design pattern** renders the implementation efficient for concurrent access by multiple users.

- Testability:
  - The use of MVC design pattern ensures that the application is easily testable.
  - Use of unittest ensures automation in test-cases.
  - We also use UI white box testing

Architecture:

- Model-View-Controller (MVC) architecture pattern is used
- The View will take the input -
  - Origin and Destination location
  - Strategy corresponding to minimum or maximum elevation
  - Select one of the two algorithms - Dijkstra and A*
  - Path Limit
- Model calls Open Street API to generate the graph and Google Maps Elevation API to populate the nodes with elevation attributes. Model notifies the view to display the computed path.



Design Patterns used:

- Strategy design pattern: We use two different algorithms using strategy design pattern. We also use multiple heuristics thereby, using a common template.
- Observer design pattern: In our architecture, the model acts as observable and the view acts as observer.