Test Plan and Report: EasyRoute

Brandon Castro, Adora Vaz, Megan Nguy, Hazel Prasetya

Sprint 2

- **User Story 1:** As a user, I want to get a route between two locations. [7]

Sprint 3

- **User Story 1:** As a user, I want to customize my route to avoid stairs. [10]
- **User Story 4:** As a user, I want to search for a destination on the map. [8]
- **User Story 5:** As a user, I want to be able to choose whether I'm walking, driving, or using any other travel mode.
- **User Story 7:** As a user, I want to see turn-by-turn directions for my route. [5]

Sprint 4

- **User Story 1:** As a user, I need to be able to see where an elevator is to get to an upper level. [3]
- **User Story 4:** As a user, I want to be able to click on a location on the map and report any accessibility features in need of repair. [3]

Scenarios

Scenario 1: Display a route between two locations on the map. (Pass/Fail)

1. Select the start building by clicking on it with the mouse.
    - `selectedBuildings.push(selectedBuilding)` (only if a building is detected)
    - `highlightBuilding(selectedBuilding)` (function that sets the color of the material)
2. Select the end building.
    - `selectedBuildings.push(selectedBuilding)`
    - `highlightBuilding(selectedBuilding)`
3. Select the "Calculate Route" button.
4. Check if at least two buildings are selected.
5. Get the centroid coordinates of the two buildings and the user profile.
6. Call the direction API, wait for a response, then call the elevation API.
    - `makeDirections(route)` (this creates a route that will be added to the map)
7. Process the directions response and display it to the user.
8. Calculate the difference in elevation and show the user.
    - Users should see a yellow route and the directions of the route along with the difference in elevation.

Scenario 2: Customize route to avoid stairs (only works for walking profiles)

1. Select any two buildings on the map.
2. Toggle the "Avoid Stairs" button.
3. Change profile type to walking.
4. Press the "Calculate Route" button.
    - Same process as above: get a profile and a mode; the mode provides the API with the proper input.
    - `const options = avoidStairs ? { avoid_features: ['steps'] } : {};`
    - `const mode = document.getElementById('travelProfile').value;`
5. Wait for a response and process it.
    - Users should see the route and an augmented path if it's possible to avoid stairs. For some paths, avoiding stairs is not feasible.

Scenario 3: Search for a destination on the map

1. Click on the first input box and type in an address.
2. Process the current input and query the buildings to display to the user.
3. The user should be able to click on the boxes instead of manually selecting buildings.

Scenario 4: Choose whether I'm walking, driving, or using any other travel mode

1. Select the mode via a dropdown menu.
2. The mode is calculated the same as above.
3. A route should display that reflects the selected mode.

Scenario 5: See turn-by-turn directions for my route

1. Press the "Calculate Route" button (expects two buildings).
2. Wait for the API response, then call parse:
    - `const segments = json.features[0].properties.segments[0];`
    - `const routeTotal = { distance: segments.distance, duration: segments.duration };`
3. Use `segments` and `routeTotal` to update the UI.
    - Users should see the directions, total distance, and total duration. These are placed under the "Clear Route" button.

Scenario 6: See where an elevator is to get to an upper level

- The map should have the elevator icons in the correct places.

Scenario 7: Report any accessibility features in need of repair

1. Select a building.

2. Select "Report Repair" on the building's popup.
3. User enters a report in the input field.
4. Select the "Submit" button.
    - Users should be able to see an icon on the building, and their report should be available inside the popup.