38883171 – Reflective Report – Sprint 3

Requirements:

This sprint, my main deliverables were centered around improving the general user experience when using our application. The main objectives of the sprint were:

- Make the webapp usable for mobile users.
- Get user's location from browser and implement related functionality.
- Map reacts to selections in the ControlPanel.

Architecture:

Main View on mobile:

The design paradigm of the Main View had to be chosen carefully. This component contains both ControlPanel and map, and is in charge of placing them in a cohesive single view. On large horizontal resolutions, such as with desktops of tablets, the existing horizontal implementation works without issues. The ControlPanel and MapWidget are placed in a size-flexible single column, allowing it to adapt to varying degrees of variation in the x-axis resolution.

The issue came when the user was using a mostly y-axis resolution. The columnar approach stop being an effective way to display the information. To solve this problem, a new vertical mode was activated after the resolution reached x width.

The initial approach was to simply stack the components in a row. This didn't work very well due to the changing size of the control panel, often resulting in the map not being visible and it resulted in a bad experience for users. As an alternative, an overlay concept with an ControlPanelIsActive state was utilized. When inactive, the controlPanel was shrunk, leaving more space for the map. To enable and disable the ControlPanel both a click and a drag interaction was added, making it friendly for all kinds of users.

Retrieving User's Current Location:

A new locationContext was created to contain all data related to the user location. This meant the actual location, whether the feature is enabled or disabled, and any errors that may have been registered. From here the ControlPanel and MapWidget simply had to implement the desired internal functionality and error handling.

The implementation of the locationUpdater was very straightforward. Querying the user location is natively supported using the react navigator. The only challenge that arose was that the user location was being queried every millisecond. To prevent this, adding a simple time out was sufficient. An additional accuracy check was added to ensure that the user location wasn't being considered if the location estimate was too unreasonable (over 500m).

Map moves to ControlPanel selections:

3 types of interactions were implemented. When searching for a specific station, the map is moved to the specified coordinates, when the user location loads or is manually centered via an interaction button, the map is moved to the specified coordinates, and when an itinerary is selected, the features of the itinerary are retrieved, and the map will move the view so that all features are visible. This was done via the fitToFeatures function, which aims to comply with the following design patterns:
Open-Closed Pattern (OCP): Self-contained function that performs a specific functionality, and is swappable if required (ex. If the map framework is changed).
Adapter Pattern: Feature types represent coordinates in different ways. The function standardizes these into a format that is accepted by the mapLibre fitbounds function.

Testing

Support for the playwright testing platform was added to our CI-Testing job. When npm run test is triggered, both jest tests and playwright tests are executed. Given that playwright tests require the application to be ran separately, an additional step was added to the npm scripts that would run the application before running the tests and shut it down afterwards.

Idle State Performance Test – Implemented with Playwright, aims to ensure that a group of pre-configured context scenarios don't cause unexpected long-running performance issues, such as those caused by memory leaks or infinite recursive loops.

1. Configure desired context scenarios (set values of contexts to desired states).
2. Load page with these new scenarios, which the UI Components should react to.
3. Record memory usage for 3 minutes and ensure that memory growth isn't excessive.

LocationUpdater Test – Ensures that all functionality related to the location updater operates as expected, including updating location when a new location is available and handling low accuracy.

Impediments and plans to address them

Low-Quality icons on Map

At some stage from the process of converting an icon into a bitmap and loading it into the map, the outcome was noticeable low-quality images, as brought up by the LVB PO.
Solution: Issue originates when MapLibre resizes the icons. Leave the icons on their native resolution, and manually resize them to the desired size.

Issues with data source for fitToFeatures

After a layer's data is loaded into the map, the generated features need to be extracted for then using them to determine their coordinates and moving the map to fit them all. When attempting to move to a large itinerary, only a partial section of it was fit into the screen. When attempting to load again a larger section would be fit, and by repeating the whole layer would be visible.
Solution: When using MapLibre's queryRenderedFeatures or querySourceFeatures, only the features in the current viewport (ie. map area). This explains the issue described above. To circumvent it, the geoJSON data used to generate the layer was stored raw, and when the action was triggered, instead of using MapLibre's functions, the raw data was directly submitted to the fitToFeatures function.

Appendix 1: Sample CI-Tests result

```
  2 failed
    testing\playwright\functionality\Map.spec.ts:6:5 › test ──────────────────────
    testing\playwright\performance\uiContext.spec.ts:33:9 › Performance of app › should
aintain performance during route planning
  20 passed (4.5m)
```

Appendix 2: Sample Execution of Idle Performance Test

```
  ✓  1 testing\playwright\performance\mainView.spec.ts:13:7 › MainView memory performance - DEFAULT (1.6m)
[DEFAULT] Heap at 10s: 97.36 MB
[DEFAULT] Heap at 20s: 97.37 MB
[DEFAULT] Heap at 30s: 97.37 MB
[DEFAULT] Heap at 40s: 97.37 MB
[DEFAULT] Heap at 50s: 97.38 MB
[DEFAULT] Heap at 60s: 89.97 MB
[DEFAULT] Heap at 70s: 89.79 MB
[DEFAULT] Heap at 80s: 89.79 MB
[DEFAULT] Heap at 90s: 89.80 MB
[DEFAULT] Heap at 100s: 89.80 MB
[DEFAULT] Heap at 110s: 89.80 MB
[DEFAULT] Heap at 120s: 89.80 MB
[DEFAULT] Heap at 130s: 89.81 MB
[DEFAULT] Heap at 140s: 89.81 MB
[DEFAULT] Heap at 150s: 89.81 MB
[DEFAULT] Heap at 160s: 89.81 MB
[DEFAULT] Heap at 170s: 89.81 MB
[DEFAULT] Heap at 180s: 89.82 MB
[DEFAULT] Total Memory Growth: -7.55 MB
  ✓  2 testing\playwright\performance\mainView.spec.ts:13:7 › MainView memory performance - ITINERARY_VIEW_SCENARIO (1.6m)
[ITINERARY_VIEW_SCENARIO] Heap at 10s: 107.00 MB
[ITINERARY_VIEW_SCENARIO] Heap at 20s: 107.00 MB
[ITINERARY_VIEW_SCENARIO] Heap at 30s: 107.01 MB
[ITINERARY_VIEW_SCENARIO] Heap at 40s: 107.01 MB
[ITINERARY_VIEW_SCENARIO] Heap at 50s: 107.01 MB
[ITINERARY_VIEW_SCENARIO] Heap at 60s: 92.86 MB
[ITINERARY_VIEW_SCENARIO] Heap at 70s: 92.67 MB
[ITINERARY_VIEW_SCENARIO] Heap at 80s: 92.67 MB
[ITINERARY_VIEW_SCENARIO] Heap at 90s: 92.67 MB
[ITINERARY_VIEW_SCENARIO] Heap at 100s: 92.68 MB
[ITINERARY_VIEW_SCENARIO] Heap at 110s: 92.68 MB
[ITINERARY_VIEW_SCENARIO] Heap at 120s: 92.68 MB
[ITINERARY_VIEW_SCENARIO] Heap at 130s: 92.68 MB
[ITINERARY_VIEW_SCENARIO] Heap at 140s: 92.68 MB
[ITINERARY_VIEW_SCENARIO] Heap at 150s: 92.69 MB
[ITINERARY_VIEW_SCENARIO] Heap at 160s: 92.69 MB
[ITINERARY_VIEW_SCENARIO] Heap at 170s: 92.70 MB
[ITINERARY_VIEW_SCENARIO] Heap at 180s: 92.70 MB
[ITINERARY_VIEW_SCENARIO] Total Memory Growth: -14.30 MB
```