MyPersonalizedRoute Front-End Development Documentation

The front-end UI of MPR is developed using the Vue.js framework and Vuetify material design component framework (v1.5.16). The UI is divided into two large components, one component dedicated to the overlay (i.e. the toolbar, menus, navigation drawers, etc.) and one component dedicated to the interactive map. The functions of which will be further described on pp. 2-5.

To use the website, start by installing the dependencies by using

npm install

in the console while in the directory of the application. To start the server use

npm run serve

List of APIs/ Plugins/ Libraries being used

- Leaflet – Javascript library to build an interactive map.
    - Documentation available: https://leafletjs.com/
    - Uses the default OSM tileset and satellite tileset from MapBox
- Leaflet Color Markers – Adds color variations of the standard Leaflet markers
    - Documentation available: https://github.com/pointhi/leaflet-color-markers
- Leaflet Routing Machine – API for the Leaflet that adds routing
    - Documentation available: https://www.liedman.net/leaflet-routing-machine/api/
    - Several supported routing engines
        - Open Source Routing Machine (OSRM)
            - We are using this engine for map matching but because it uses an OSRM demo server, we are limited to 100 points per map matching request
            - Documentation available: http://project-osrm.org/docs/v5.5.1/api/#match-service
        - TomTom (Engine currently being used)
            - Documentation available: https://developer.tomtom.com/routing-api/routing-api-documentation
            - Note: free users are only granted 2,500 credits daily, each HTTP request sent to this service will consume credits
- Leaflet.heat – Leaflet heatmap plugin
    - Documentation available: https://github.com/Leaflet/Leaflet.heat
- Leaflet.measure – Leaflet plugin that adds a distance measuring tool
    - Documentation available: https://github.com/aprilandjan/leaflet.measure
- Nominatim – Geocoding/ reverse-geocoding API for Leaflet
    - Documentation available: http://nominatim.org/release-docs/latest/api/Overview/
    - Note: Functionality such as auto-complete is explicitly forbidden as mentioned in their usage policy available: https://operations.osmfoundation.org/policies/nominatim/

- Axios – Promise based HTTP client that is used to handle HTTP requests (i.e. to handle responses from the back-end)
    - Documentation available: https://github.com/axios/axios
- PROJ4 – Library to perform coordinate transformations
    - Documentation available: https://www.npmjs.com/package/proj4

MPR Functionality

As mentioned above the application is separated into two components, toolbar.vue, which contains all the overlays and the UI that users are interacting with, and leaflet.vue, which contains everything related to the interactive map. For the toolbar to communicate with the leaflet map, events are used to pass data between components. Functions, or *methods* in Vue, and the events/ actions are explained for each component below.

toolbar.vue

Function	Input	Output
grabTime() Function that gets the local time for the system and displays it in navigation drawer	None	Local time on user's machine
findLoc(searchQuery, option) Function that will find a location (address) for the user	• ***searchQuery*** This is the address that a user is trying to lookup • ***option*** This dictates which field the user is searching from (i.e. from the search bar, from the "origin" text field, from the "destination" text field)	Emits "**search-submit**" event to leaflet.vue, passing the searchQuery and option
findRoute() Function to find a route between two waypoints	None	Emits "**find-route**" event, passing lat and long of origin/ destination, option (i.e. fastest, shortest distance, personalized, etc), and time (currently time isn't used)
showSpatial() Function that activates when the spatial coverage or temporal coverage is toggled on/ off	None	Emits "**show-spatial**", passing spatial (Boolean) and temporal (Boolean)
clearRoute() Function that clears a route off map	None	Emits "**clear-route**", does not pass any data

Event Handlers	**Input**	**Output**
"set-field" Sets the coordinates and address of the origin and destination. This is for handling interactions with the map	• ***mark.location*** This has the latitude and longitude to be reverse geocoded • ***mark.choice*** This dictates if it's the origin or destination	• ***origin*** Address of origin/ starting pint • ***dest*** Address of destination • ***origin_latlng*** Lat and long of the origin • ***dest_latlng*** Lat and long of the destination
"fill-field" Fills the text fields (i.e. origin, destination, and search bar fields) with address after reverse geocoding. This is for handling search results	• ***result.name*** This has the latitude and longitude to be reverse geocoded • ***result.choice*** This dictates if it's the origin, destination, or search bar • ***result.coords*** Lat and long of the search result	Same as "set-fields"

leaflet.vue

Function	Input	Output
createButton(label, container) Creates a button for the map	• *label* Label for the button (i.e. "start") • *container* What you are putting the button in (i.e.)	Returns ***btn***
initMap() Function to initialize the interactive map	None	Adds map tiles/ layers (OSM and Mapbox), layer control, scale, measuring tool, geocoder, router
onLocationError(e) Alerts user if there is an error getting their location when first entering website (i.e. if user does not allow website permission to use location	"e" is just an event object	Error message

onLocationFound(e) Adds marker to map where the user is and save their coordinates	Same as above	Sets marker on map and emits "**fill-fields**" to set the origin (for routing) to the user's current location
locateUser() Function that uses built-in Leaflet function "locate" and "setView" to position the map to the user's location	None	Positions the map so that the user's location is centered
setField(latlng, choice) Function that runs when user clicks on map and choses to set their origin or destination to that location		Emits "**set-field**", passes location (lat and long) and choice
Event Handlers	**Input**	**Output**
"search-submit" Takes the address submitted by a user and geocodes it to find the lat and long, also add marker to map	• *search.query* User's search query (i.e. University of Calgary) • *search.choice* Dictates if user selected the location as their origin or destination	Location's lat, long, and complete address. Emits "**fill-fields**", passes complete address, choice, and coordinates
"find-route" Finds route given two points given any of the four options: fastest, shortest distance, personalized, or most popular	• *query.orig* – User's origin • *query.dest* User's destination • *query.option* Dictates which routing option the user selected • *query.time* User's current time, used for personalized and most popular route only	Visualization of route on the map along with the directions
"show-spatial" If spatial coverage or temporal coverage is toggled on/ off by user, display/ hide the UI (i.e. heatmap and time slider). Note that only	• onoff.spatial and onoff.temporal (both Booleans) Used to see if spatial coverage or temporal coverage has been toggled on or off	Enable/ disable heatmap layer for spatial coverage on map. Enable/ disable temporal coverage layer on map.

one can be toggled on at a time		
“clear-route”	None	Remove route, and origin/ destination waypoint markers