

# 38883171 – Reflective Report Final

lu-lvb-JourneyPlanner

<https://github.com/JRamirezDD/lu-lvb-JourneyPlanner>

## Requirements:

My responsibilities for this sprint mainly revolved around preparing access to the lvb APIs from the UI components, providing the automated testing framework, and coordinating and organizing the application's development. As the project moved forward, I concluded that developing the component-communication framework would also fall into my responsibility, and not another group members. For a more detailed overview, refer to *Appendix 1.1 – Originally Assigned Requirements* and *Appendix 1.2 – Additional Completed Requirements*.

## Architecture:

An unrefined Use Case diagram was used for the initial definition of the required components, interactions, and external sources of data. Refer to *Appendix 2.1 – Use-Case Diagram Draft*.

The main architectural choice came when deciding how the UI Components would communicate with each other (marked as Data-Transfer in the Use-case diagram).

When looking at other OTP-UI Implementations, they all use a methodology named Props. Nevertheless, we believe that Context APIs are the superior approach, as they allow for increased modularity, easier extensibility, and simpler communication of objects via a “subscription” model. Refer to [Component-Communication Design Decision and Implementation](#) and *Appendix 2.2 – Data-Flow Diagram* for more details.

## Testing

Application tests are implemented via Jest. Ideally, we aim to create unit tests of all the minor parts, but core, of the application, such as data format conversion, layer generation, reactions to context changes, among others. With the time provided, this is very ambitious, so consider us satisfied if non-ui components are fully tested.

## Automated Tests

GitHub job (<https://github.com/JRamirezDD/lu-lvb-JourneyPlanner/blob/main/.github/workflows/node.CI.yml>) was created to ensure that commits to the main branch are compliant. Its goal is to ensure that the CI process of the application runs as expected via the following steps: Install dependencies, Run Jest tests, Build project

## Impediments and plans to address them

- Context APIs
  - o Goal was to hand over tasks to team member to give me time to work on supporting less capable team members, but he fell ill during the critical development time. I proceeded to develop the component-communication framework myself, and although some shared attributes are likely missing, it's finished for the most part.

- Map Layers
  - I had to involve myself a lot more than expected for the implementation of the Map Component. Team members are having issues standardizing the OTP data import into the MapBox component. Will create a framework with interfaces to simplify approach, and team members will just have to set-up filters for the imported OTP data.
- Map rendered by client
  - I'm not fully familiar with the correct approach to using and loading layers into MapBox, but currently the raw data is provided to the client, and then it renders the map. This might be an issue further on, and we might have to render the map first at the back-end, and then share it with the client.
- API-Key used by client
  - API-Key needed to access the LVB APIs is currently exposed to the client. This is not ideal, and it poses a security risk. We'll have to create a back-end API or proxy that injects the API-Key for accessing these resources.
- CORS
  - LVB APIs block requests from non-public IP addresses through CORS. Hopefully the same proxy as implemented for the API-Key injection will solve the issue. Currently using a public proxy for development.

## Contributions

- Task distribution.
- Requirement definition.
- Architectural approach.
- Component definition.
- Answering questions and helping team members implement their components.
- Implementation of Automated Tests on GitHub.
- Implementation of API-services to communicate with LVB-APIs.
- Implementation of Context Interface and Contexts.
- Implementation of Data Context Interface, standardized API access hook, and Data Contexts.
- Example components and pages for other team members to reference.
  - Components (under /components/\_sample)
    - autocompleteComponent
    - counterComponent
    - settingsComponent
  - Pages (under /app/examples/contexts)
    - autocomplete
    - multi-component
    - trigger-function
- Shared Contributions
  - Map Layers
    - Interface Layer
    - ItineraryLayer
  - Map Widget

# Appendices

## Appendix 1. - Requirements

### Appendix 1.1 – Originally Assigned Requirements

GH Issue #	Title	Estimated Size	Status
#1	<a href="#">Document APIs in shared PostMan Workspace</a>	XS	COMPLETED
#2	<a href="#">Create Use Case Diagram for Requirement Interaction Overview</a>	S	COMPLETED
#4	<a href="#">Create Repository Skeleton</a>	XS	COMPLETED
#5	<a href="#">Create App Skeleton with Plugins</a>	S	COMPLETED
#6	<a href="#">ESLint Formatting Rules</a>	XS	PENDING Basic rules implemented.
#7	<a href="#">Prettier Formatting Rules</a>	XS	PENDING
#8	<a href="#">autocompleteService API Implementation</a>	XS	COMPLETED
#11	<a href="#">Create Sprint 1 Requirement Tasks and Distribution</a>	S	COMPLETED
#12	<a href="#">nearbysearchsearchService API Implementation</a>	XS	CANCELLED Extra effort, not demanded in requirements.
#13	<a href="#">routingService API Implementation</a>	XS	COMPLETED
#14	<a href="#">stopmonitorService API Implementation</a>	XS	COMPLETED
#28	<a href="#">BaseMap Implementation</a>	L	PENDING

### Appendix 1.2 – Additional Completed Requirements

GH Issue #	Title	Estimated Size	Status
#15	<a href="#">Component-Communication Design Decision and Implementation</a>	L	COMPLETED
#18	<a href="#">Implement automated unit test execution for PRs</a>	M	COMPLETED
#19	<a href="#">Add interface toGeoJson for DTO objects</a>	XS	COMPLETED
#30	<a href="#">Create MapContext</a>	XS	COMPLETED
#31	<a href="#">Create UIContext</a>	XS	COMPLETED
#32	<a href="#">Create TripContext</a>	XS	CANCELLED
#33	<a href="#">Create DataContext Interface</a>	M	COMPLETED
#34	<a href="#">Create SettingsContext</a>	XS	COMPLETED
#35	<a href="#">Context API Interface</a>	M	COMPLETED
#40	<a href="#">Removing API-Key from client</a>	XS	PENDING
#41	<a href="#">Create autocompleteDataContext</a>	S	COMPLETED
#42	<a href="#">Create routingDataContext</a>	S	COMPLETED
#43	<a href="#">Create stopmonitorDataContext</a>	S	COMPLETED

## Appendix 2.1 – Use-Case Diagram Draft



## Appendix 2.2 – Data-Flow Diagram

