COMP.SE.110–2024–2025–1 Software Design

1.- Introduction

This document aims to provide an overview of the architecture and design of the *Weather and Traffic* application. This application is designed to help users make informed decisions about public and urban transportation by providing them with real-time updates on weather conditions and how these may affect different services. It also provides instant alerts and updates on local traffic, all through an intuitive and easy-to-use interface.

2.- Design and System Components

The initial design phase of the project included the creation of several preliminary sketches that served as the basis for the development of the application. These first sketches allowed us to visualize the overall structure of the system and facilitated the identification of the key components required. From these sketches, we were able to make more informed decisions about the architecture and functionalities of the software, allowing us to start the project with a clear and well-defined direction.

2.1.- System Components:

- Front-end: We evaluated several front-end frameworks to build the application's interfaces. After reviewing the options, we selected React due to its ease of use, strong community support, and ability to integrate various libraries and components.
    - Key functionalities:
        - Real-time display of transportation routes and schedules.
        - Display of weather alerts and traffic conditions.
        - Notifications of service delays or disruptions.
        - Navigation menu to select different types of transportation or services
- Back-end: For the back-end development, we are using the programming language assigned in the course, which is Java. The choice of Java allows it to take advantage of its robustness, portability and extensive support community. In addition, its

extensive ecosystem of libraries and frameworks facilitates the implementation of various system functionalities.

- Key functionalities:
  - Process front-end requests to retrieve transport and weather data.
  - Manage user authentication and session.
  - Integrate and manage external APIs.
  - Apply business logic to determine alerts or recommendations based on real-time data.

– APIs: For the integration of functionalities, the following APIs were considered: *JourneysAPI / TransportAPI, OpenWeatherMap / OpenMeteo, Finnish Meteorological Institute API (FMI API), DigiTraffic API.* More on these APIs later

3.- Application Structure and Design Patterns

3.1.- Application Structure:

The following figure describes the back-end structure of the application on a high level. The components *TransportController* and *WeatherController* handle requests from the client side and route them to the service layer. They also forward responses back to the client. *TransportService* and *WeatherService* are service layer components, which handle the business logic of the application. They fetch data from the external APIs and map it to the data models *TransportStatus* and *WeatherStatus*, returning results back to the controller components.

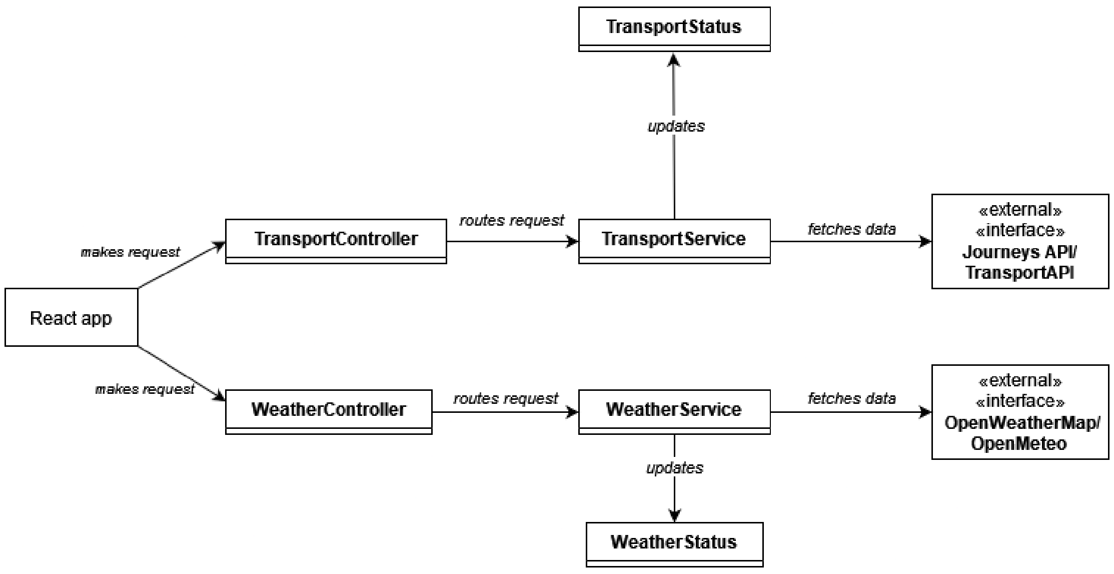

Figure 1: Back-end structure and interactions

3.2.- Design Patterns:

The application architecture follows the Model-View-Controller and Service Layer design patterns.

The *TransportStatus* and *WeatherStatus* classes represent the data in the back end, constituting the Model component. The React front-end application serves as the View, presenting information to users, while the *TransportController* and *WeatherController* function as the Controller, managing the interaction between the user interface and the service layer.

This approach is widely recognized for introducing a clear separation of concerns in an application. The modularity of this design enhances maintainability and allows for future extensions, such as the integration of new APIs. Additionally, the Service Layer, implemented through *WeatherService* and *TransportService*, further refines this separation by encapsulating the business logic.

4.- APIs

In this section we present the APIs that were considered for the development of the application, alongside an analysis of their strengths and weaknesses based on our investigation and initial testing. These APIs were chosen to fulfill key functionalities such as providing real-time transportation information, weather forecasting, and traffic updates. By assessing each API's capabilities, we aimed to select the most suitable tools for creating a reliable and efficient user experience. This evaluation also includes considerations regarding ease of integration, data accuracy, geographical coverage, and potential limitations that may impact future scalability.

1. **JourneysAPI / TransportAPI:** This API provides real-time information on public and urban transport routes and schedules. It has been used to provide users with up-to-date details on transportation options, allowing them to plan their trips accurately.

    - **Pros:**

        - **Real-Time Updates:** The API provides real-time data, which is crucial for keeping users informed about delays or incidents.
        - **Extended Coverage:** The API has considerable coverage of multiple modes of transportation, which helps to offer versatile travel options to users.
        - **Flexibility:** Supports different data formats (JSON/XML), which facilitates integration with other backend components.

- **Cons:**

  - **Documentation Limitations:** Documentation can be somewhat limited or difficult to follow, which made the initial integration process more complex.

  - **Data Access Limitations:** Restrictions were sometimes encountered on the amount of data that can be requested within certain time intervals, which can be an obstacle if a large volume of information is to be obtained for analysis.

2. **OpenWeatherMap / OpenMeteo:** These APIs have been used to obtain weather data. OpenWeatherMap provides detailed data on current weather conditions and forecasts, while OpenMeteo was also used as an alternative, offering similar information.

   - **Pros (OpenWeatherMap):**

     - **High Accuracy and Coverage:** Provides very detailed data, including hourly forecasts, which is very useful for alerting users to conditions that may affect transportation.

     - **Extensive Forecasting Capabilities:** Provides a good variety of data, including temperature, wind, precipitation, and special conditions such as storms.

   - **Cons (OpenWeatherMap):**

     - **Query Limit:** The free version of the API has restrictions on the number of queries that can be made per minute, which can be a problem when the application has many users.

     - **Cost Associated with Additional Data:** To obtain advanced features, such as hyperlocal information, it is necessary to contract paid plans, which could increase the cost of the project.

   - **Pros (OpenMeteo):**

     - **Free and Easy to Use API:** OpenMeteo is a good free alternative, and its integration was easy due to a simple API structure.

     - **Specialized Data for Specific Locations:** It can provide customized and accurate weather information for specific locations without incurring high costs.

- **Cons (OpenMeteo):**

  – **Shallower Data Depth:** Compared to OpenWeatherMap, OpenMeteo has less data available for certain advanced indicators such as air quality.

  – **Update Frequency:** The frequency with which data is updated may be lower than OpenWeatherMap, which may affect the accuracy of real-time alerts.

3. **Finnish Meteorological Institute API (FMI API):** The API of the Finnish Meteorological Institute has been used to obtain accurate weather data specifically within Finland. The API provides detailed and official weather information for the country.

   - **Pros:**

     – **Official and Highly Accurate Data:** Being data provided by an official meteorological institute, the accuracy is very high, especially within Finland.

     – **Free for Users in Finland:** There are no significant limitations of use if the goal is to provide data locally in Finland, which is ideal for users within the country.

   - **Cons:**

     – **Limited Geographic Coverage:** It is focused only on Finland, which limits the capability of the application if you want to expand to international users.
     – **Complex Documentation:** Initial integration was a bit challenging due to documentation that is sometimes less accessible or lacking in detail for certain use cases.

4. **DigiTraffic API:** The DigiTraffic API has been used to obtain traffic data in Finland. This API provides detailed information about the current traffic situation, traffic events, and possible disruptions.

   - **Pros:**

     – **Real-Time Data:** DigiTraffic provides real-time traffic information, which is crucial for users to better plan their trips.

     – **Extensive Traffic Information:** Includes detailed information on incidents, road works, and traffic levels on specific routes, helping the app provide a comprehensive view of local traffic.

- **Cons:**
  - **Limited Coverage to Finland:** Similar to the FMI API, DigiTraffic has limited coverage to Finland, so it is not useful if the application wishes to have an international reach.
  - **Complexity in Data Integration:** The amount of data available is extensive, and while this is an advantage, it can also be a challenge to integrate it effectively and ensure that the most relevant data is what is displayed to the user.