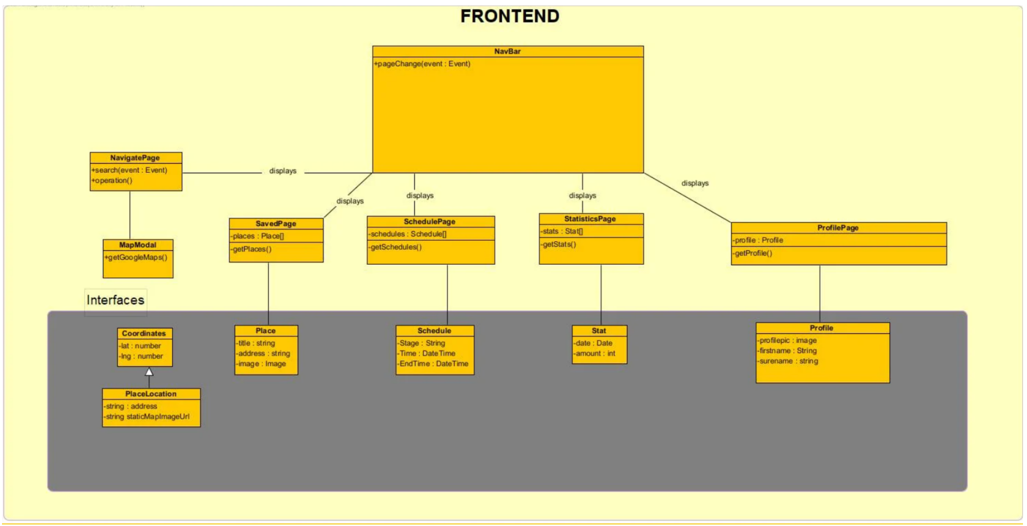
Software Requirements Specification

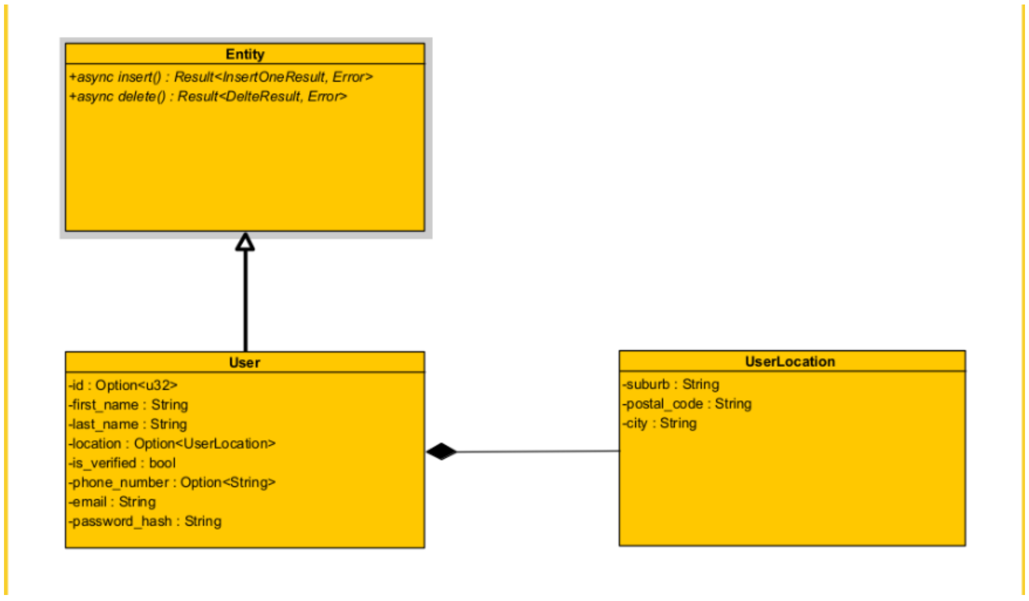
Introduction

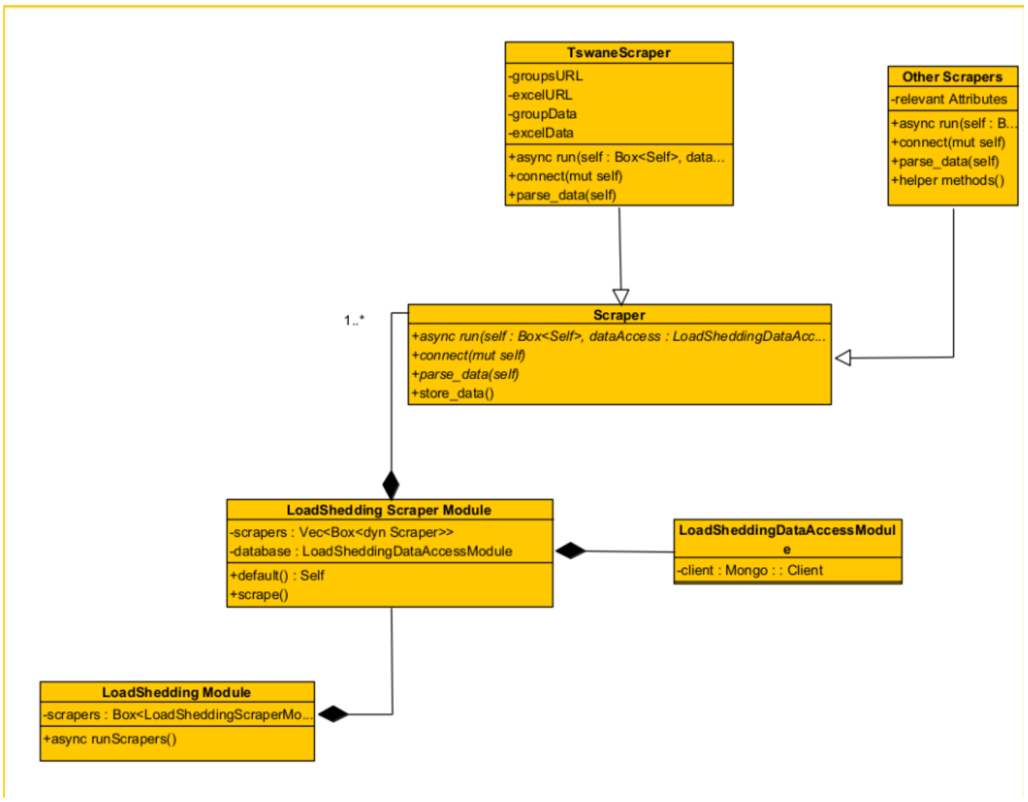
Where is the power, is a mobile and desktop app that navigates users through these dark times in South Africa. Loadshedding is inconveniencing a lot of South Africans, especially on the roads. Where is the power aims to assist South Africans avoid traffic, plan their day, find areas that have electricity, inform communities of surprise power outages in local areas and more.

By means of a map, areas will be coloured in on the map to indicate the status of loadshedding and any other power related issues. Navigating routes that are not impacted by loadshedding is a must, especially when trying to get home. Statistics will be used to identify what the areas' average uptime is and more.

Class Diagram







User Characteristics

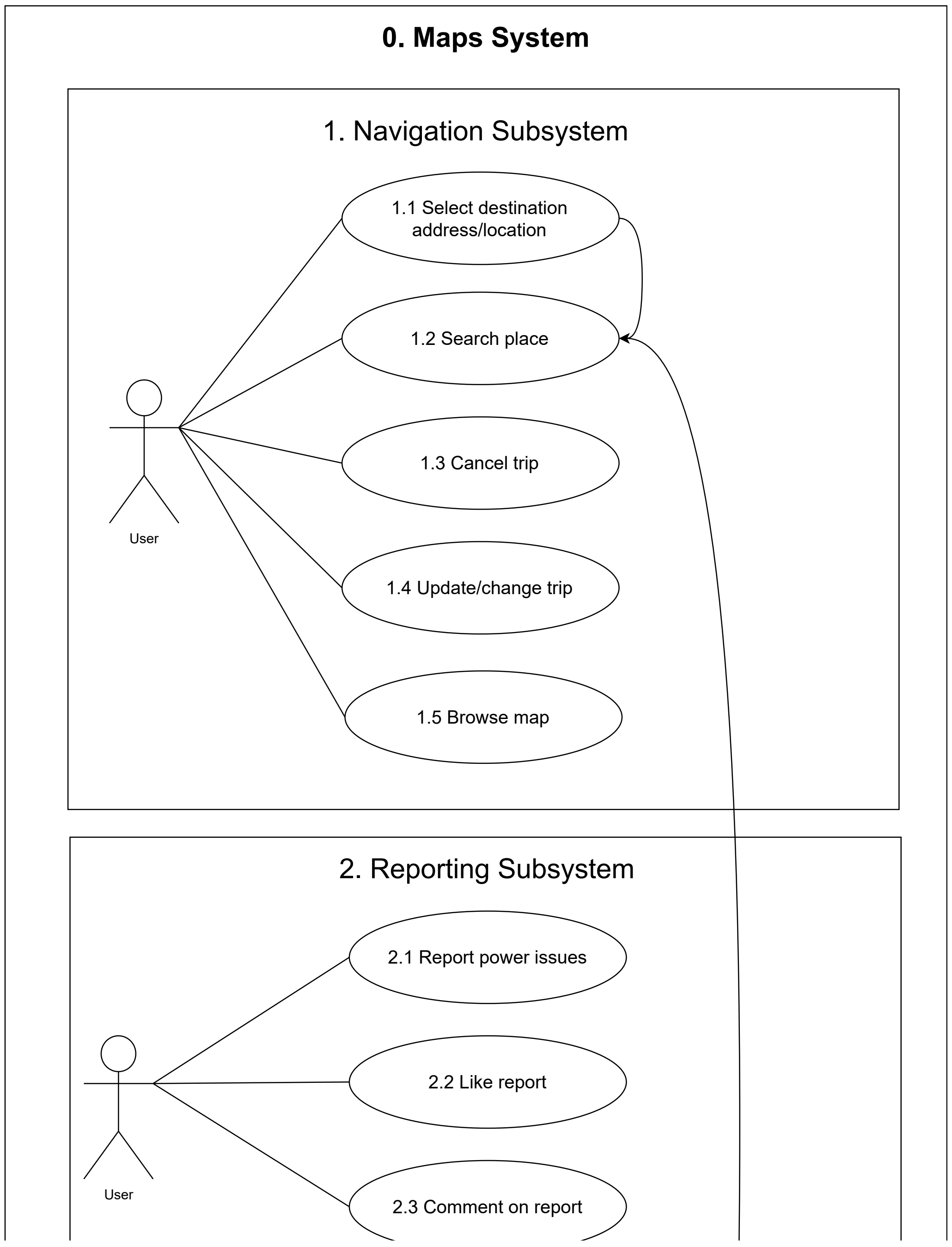
User Story

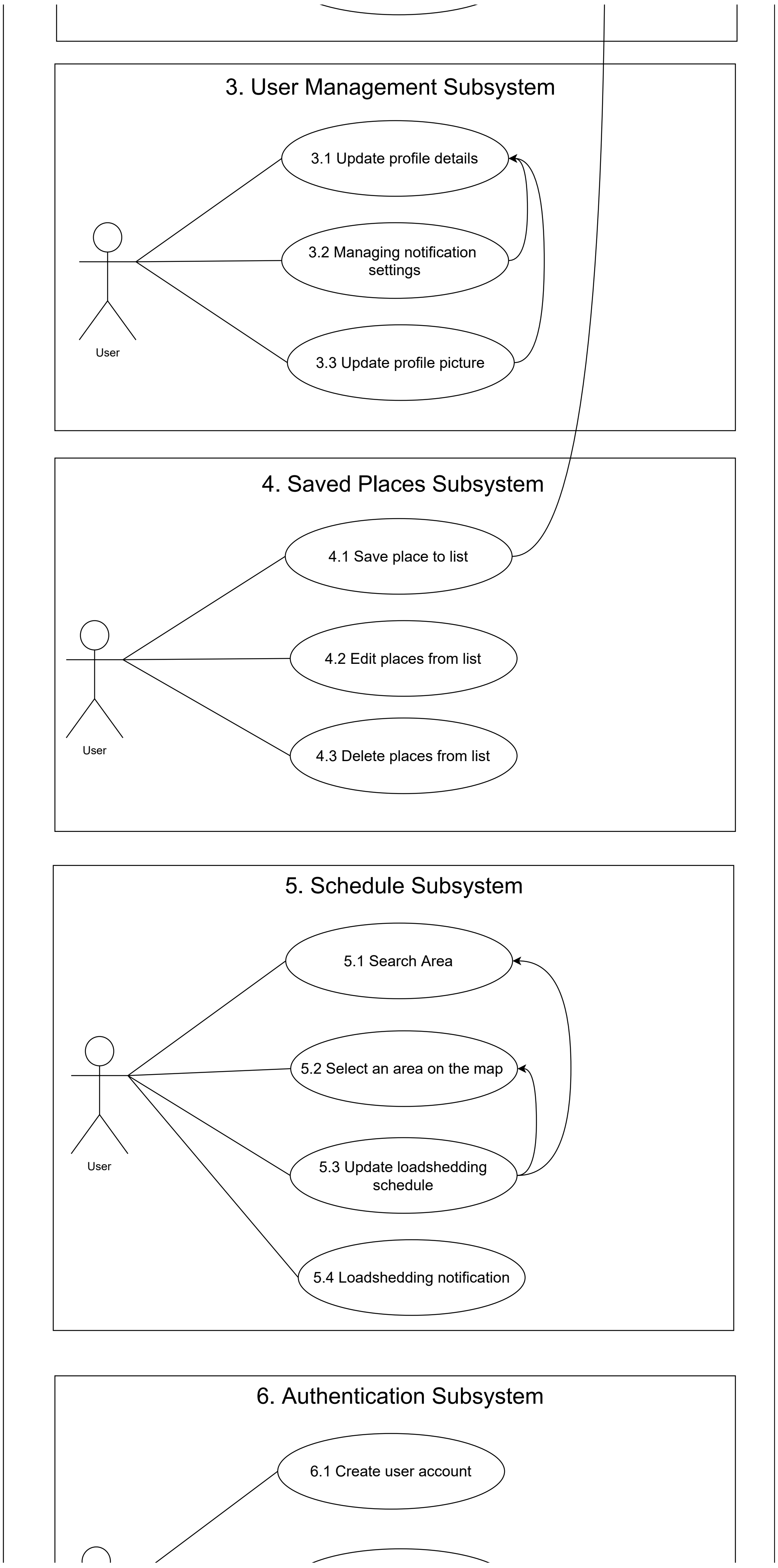
- Normal user
  - This user interacts with the maps.
  - Able to navigate through roads, where loadshedding is not happening.
  - View the statistics of areas around them.
  - See reports(e.g. Stolen cables) that are happening in Areas.
- Registered User
  - Everything a normal user can do.
  - This user can report problems in an area, while a normal user can not.
  - Receives push-notifications (mobile).
- Example user Stories
  - As a a commuter, who has to travel from one place to another, I want an application that finds the optimal route despite load shedding, so that I don’t have to manually look at the load shedding areas and schedules and plan out my route manually. Given the destination location, and the time of travel, the user may find the optimum route, avoiding load shedding. When the user clicks navigate, then they are shown a map with a search bar at the top, then the user can search for a place.
  - As a member of the community, I want to make awareness to any cables stolen, substations, blown or any other related power outage reasons, so that I can inform others and they can plan out their routes with more insight using the app. Given the user has something to report, when

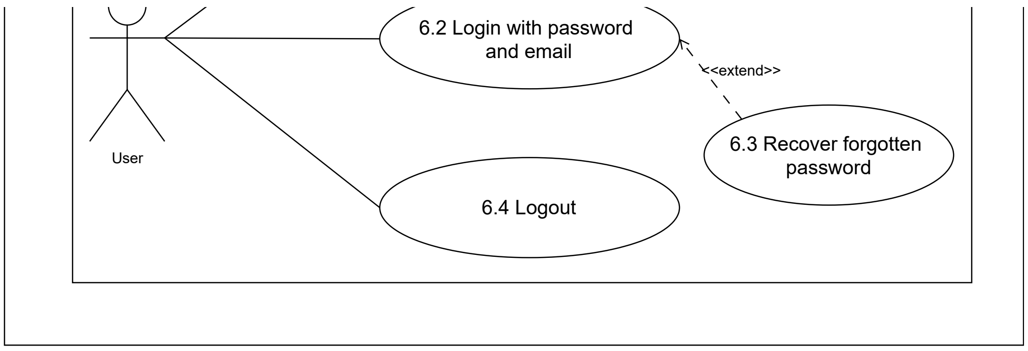
they click the report button, then they can select the type of report issue and fill in other details and send off a report.
  - As a member of the community, I want to know the stages of load shedding for a particular area,so that I don't have to use a separate app to do so. Given the user's desire to view the schedules, when the user clicks the schedule tab, then the user will be shown all the schedules for a particular area they wish to see.
  - As a South African, I would like to see on my map which areas have loadshedding so I can plan my trip.
  - As a South African, I would like to report on issues in my area regarding stolen cables such that I can inform my community on outages
  - As a user I would like to save places on the map so that I can easily access them

Functional Requirements

Use Cases




3. User Management Subsystem
3.1 Update profile details
3.2 Managing notification settings
3.3 Update profile picture
User
4. Saved Places Subsystem
4.1 Save place to list
4.2 Edit places from list
4.3 Delete places from list
User
5. Schedule Subsystem
5.1 Search Area
5.2 Select an area on the map
5.3 Update loadshedding schedule
5.4 Loadshedding notification
User
6. Authentication Subsystem
6.1 Create user account


6.2 Login with password and email
<<extend>>
6.3 Recover forgotten password
User
6.4 Logout

Service Contracts

Requirements

- Open maps with loadshedding colours overlay
  - Colour overlay
  - Getting load shedding times for areas
  - Average uptime of an area
  - Click on area
  - Display load shedding INFO
  - Transition colour area for when load shedding
- Report power issues for an area
  - Report area down
  - Show where power is having trouble unrelated to Load shedding
  - Heatmap of reports
  - A report is valid for 30 minutes and requires a new report
  - Ask user if report is still valid
- Navigation
  - ETA based on loadshedding
  - Route alterations to avoid load shedding
  - Plan trip
  - Cancel trip
  - Updating trip
- Statistics
  - Display Daily loadshedding graph
  - Display weekly loadshedding graph

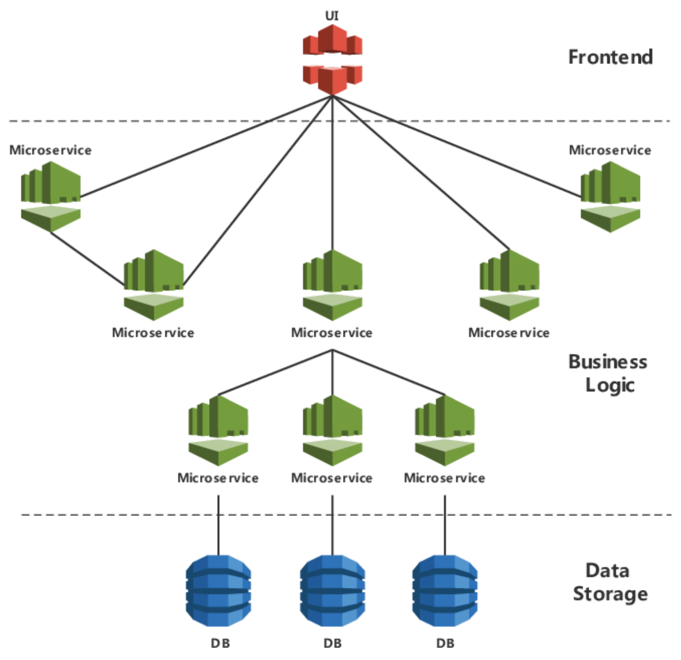
Architectural Requirements

Quality requirements

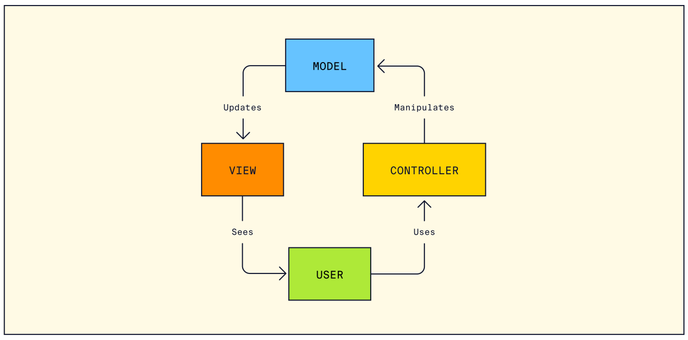
- Reliability
- Consistency
- Usability
- Maintainability
- Scalability
- Mobility
- Reusability

Architectural Patterns

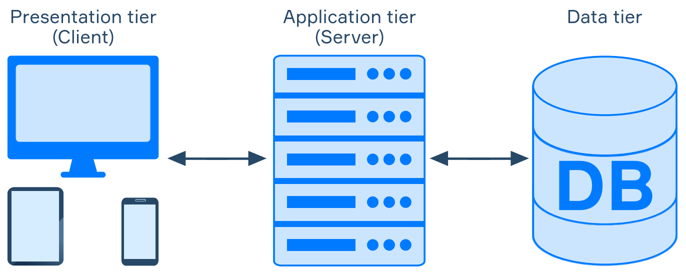
Microservices: The application is divided into microservices, each responsible for specific functionalities.



Model-View-Controller (MVC): The MVC pattern is implemented within each microservice to further organize and separate concerns.



Three-tier: Front-end + API + Backend



- Maps and Geolocation Microservice
  - **Model**: `Represents the data structure and logic related to maps and geolocation. This includes storing and retrieving map data, handling`

geolocation-related operations, and integrating with external map services.
    - **View**: `Defines the presentation layer responsible for rendering the maps and related UI component.`
    - **Controller**: `Handles user interactions, manages the communication between the model and view, and triggers actions based on user input.`

- Reporting Microservice

    - **Model**: `Contains the data structures and business logic for problem reporting. This includes storing problem reports, managing their lifecycle, and integrating with other services for notifications or processing.`
    - **View**: `Displays the problem reporting forms, user interface for browsing problems, and any related visualizations`
    - **Controller**: `Handles user input, validates and processes problem reports, and interacts with the model and view accordingly.`

- Time Schedules Microservice

    - **Model**: `Manages time schedules, including storing schedules, validating and processing schedule data, and integrating with other services for notifications or reminders.`
    - **View**: `Presents the schedule information, UI components for managing schedules, and any relevant visualizations.`
    - **Controller**: `Handles user interactions related to time schedules, triggers actions based on user input, and communicates with the model and view.`

- Saved Places Microservice

    - **Model**: `Stores and manages saved places data, including CRUD operations on saved places, categorization, and integration with map services.`
    - **View**: `Displays the list of saved places, provides UI for adding, editing, and deleting saved places.`
    - **Controller**: `Handles user interactions related to saved places, manages communication between the model and view, and triggers actions based on user input.`

- Power Outage Statistics Microservice

    - **Model**: `The model in the power outage statistics microservice would include the data structures and business logic for managing power outage statistics.`
    - **View**: `Have a view component responsible for fetching the statistics from the backend API and passing them to the frontend for display.`
    - **Controller**: `The controller in the power outage statistics microservice would handle the business logic and expose endpoints for retrieving and processing power outage statistics.`

- Backend API

Use the Rust Rocket framework to build the backend API that exposes endpoints for communication between the frontend and the microservices. It handles requests from the frontend, performs necessary data processing and integration with the microservices, and returns responses.

- Database

  Utilize MongoDB as your database to store the relevant data for each microservice. Each microservice can have its own collections or schemas in the database.

- Three-Tier Architecture

  The Three-Tier Architecture gets implemented at a higher level to structure the app into three layers

  - **Presentation Tier**: `Use Ionic and Angular to build the frontend user interface (UI). This layer handles UI rendering, user interactions, and communication with the Application Layer through APIs.`
  - **Logic Tier**: `Use Rust Rocket as the API framework to build the middle layer. This layer receives requests from the Presentation Layer, processes them, and interacts with the appropriate Microservices or backend services. It acts as the bridge between the frontend and backend services, handling business logic, request routing, and authentication.`
  - **Data Tier**: `Utilize Rust and MongoDB for the backend services and database. Implement the necessary Microservices to handle specific functionalities such as maps, problem reporting, time schedules, and saved places. Each Microservice can have its own Models, Views, and Controllers, following the MVC pattern, to handle specific logic related to that functionality. The Microservices interact with the MongoDB database to retrieve and store data.`

Design Patterns

- Dependency Injection
- Template Method
- Observer
- Facade

Constraints

- Display one loadsheading time sheet at a time.
- Show loadshedding statistics for one area at a time.
- Load *n* areas around the currently located area/searched are of the user.

Technology Requirements

ANGULAR IONIC RUST MONGODB AMAZON AWS