Software Requirements Specification

For

Apollo 11

Version 1.0 approved

Prepared by
Agarwal Radhika (U2323183J)
Ang Ming Yang (U2322946G)
Gunda Sai Venkata Aaditya (U2320456C)
Mohamed Yaseen Aboobucker Siddhique (U2323243C)
Titus Lua Jie Qing (U2222033A)

Lab Group – SCSA

13 April 2025

Members, Roles and Acknowledgements:

Ang Ming Yang

Source Code, Sequence Diagrams, Class Diagrams, System Architecture Diagrams, Documentation (secondary), Test Cases (secondary)

Agarwal Radhika

Documentation, Use Case Model, Class Diagrams, Design Model, Test Cases (secondary)

Gunda Sai Venkata Aaditya

Documentation, Test Cases, Use Case Model, State Machine Diagrams, Source Code (secondary)

Mohamed Yaseen Aboobucker Siddhique

Use Case Model, Class Diagrams, State Machine Diagrams, Design Model, Documentation (secondary), Source Code (secondary)

Titus Lua Jie Qing

Test Cases, Documentation, Use Case Model, Class Diagrams, Source Code (secondary)

1. Introduction

1.1 Purpose

Singapore plans to be a Smart Nation that is data-driven. Singapore has a relatively extensive rail network compared to many countries of its size and is constantly expanding with new stations. Many have speculated the future layouts of the railroad to be vast and extensive, such that there would be redundancies and multi-route to a particular destination. Youtuber bananasolid & Reddit post by plentk (plentk, 2024) presents a speculative map of Singapore's future rail network, envisioning a vast system with redundancies and multiple routes to each destination.

A significant portion of Google Maps data is reliant on crowd-sourced phone GPS data sharing. We aim to offer our insights to optimal route pathing primarily from LTA's data directly, LTA Datamall.

1.2 Primary Stakeholders

- **Commuters:** Aim to find the fastest route through the MRT System to reach a destination.
- **LTA Managers:** View congestion data to optimise the frequency of trains to meet user demand while minimising costs.
- **Map providing services:** Seek to integrate our system as a Software-as-a-Service (SaaS) to enhance their transit navigation and optimization features.

1.3 Documentation Conventions

- Font size 12 Arial
- Requirement Prioritising into: High, Medium, and Low
- Documentation: The GitHub will be constantly updated with various versions.

1.4 Intended Audience and Reading Suggestions

This document is intended for:

- **Developers**, for understanding the current state of implementation and testing approaches.
- **Supervisors**, to evaluate the progress and correctness of the system against requirements.

- **LTA Stakeholders**, as a future reference on how such a system can be implemented for public use.

1.5 Project Scope

With increasing MRT ridership in Singapore, especially during peak hours, train overcrowding remains a persistent issue. Commuters often lack access to timely information regarding train congestion levels, resulting in discomfort, travel delays, or inefficient route choices.

The Train Overcrowding Analysis and Mitigation System is designed to:

- Collect and display real-time crowd levels across MRT stations using LTA's DataMall API
- Use historical data and basic prediction techniques to offer congestion trend forecasts
- Provide smart route suggestions that avoid high-density stations
- Enable users to set congestion alerts when load levels exceed personal thresholds
- Support data analytics dashboards for transport authorities

This system empowers commuters with better decision-making tools and offers LTA a data-backed platform for planning and intervention.

2. Interface Overview

2.1 User Interface

The Train Overcrowding Analysis and Mitigation System features two primary user interfaces: one for **commuters** (general users) and another for **LTA administrators**. The user interface is designed to be mobile-responsive, intuitive, and accessible to the general public.

2.1.1 Commuter Interface

Screen	Description
Login/Register Screen	Allows users to create an account, log in, and set preferences (e.g. home/ work MRT stations, alert thresholds).
Home Screen	Shows a live map of Singapore's MRT network with color-coded indicators of crowd levels at different stations using data from the LTA API.
Real-Time Congestion Panel	Displays live crowd metrics for selected stations. Users can select a route and get crowd data for each station along the path.
Alternative Route Suggestion	Upon entering a start and end station, this module suggests the most efficient route that avoids high-congestion zones.
Congestion Alerts Panel	Allows users to set congestion thresholds for specific stations. Push/email alerts are sent when live data exceeds set values.
Historical Trends Dashboard	Graph-based visualization of congestion trends by time of day or date range. Helps users plan commutes ahead.

2.1.2 Admin Interface (LTA)

Screen	Description
Admin Dashboard	Visual analytics dashboard displaying overall congestion levels, frequently congested stations, and commuter behavior trends.
Alert Logs Viewer	Tracks how many users have received alerts, and which stations triggered them most often.
Data Export Panel	Enables LTA staff to export congestion logs and user interaction data for further analysis.

2.1.3 User Experience Highlights

- Clean tab navigation (Home, Alerts, History, Route)
- Color-coded congestion indicators (Green = Low, Red = High)
- Input validation on station selection and thresholds
- Responsive design for desktop and mobile
- Accessible color scheme for visually impaired users

2.2 Software Interface

2.2.1 API Used

The system integrates with the **LTA DataMall API**, which provides real-time data on MRT congestion levels across various train lines in Singapore.

- **API Name**: TrainCrowdData
- **Provider**: Land Transport Authority (LTA)
- **Endpoint**: https://datamall2.mytransport.sg/ltaodataservice/TrainCrowdData
- **Request Format**: HTTP GET with API key
- **Response Format**: JSON
- **Data Refresh Rate**: Every 10 minutes
- **Fields Returned**: Station code, load level, timestamp, line name, direction

The API is polled at regular intervals, and the data is stored and processed to generate real-time and predictive congestion alerts.

2.2.2 Database Used

The backend system uses a **PostgreSQL** relational database to persist system data. The following tables are used:

Table Name	Purpose
`users`	Stores user details, preferences, and alert thresholds
`station_data`	Stores congestion data (live + historical) by station and timestamp
`alerts`	Stores active and triggered alerts per user
`routes`	Stores predefined and suggested routes along with congestion levels
`logs`	Tracks

2.2.3 Key Product Functions

Commuters

Function	Description
Login and Account Creation & Password Reset	Create an account, log in securely, and set travel preferences
Edit Profile and Preferences	Update congestion thresholds, alert settings, and preferred stations

View Real-Time Congestion	Search for MRT stations and view live congestion levels
Find Best Routes	Get shortest route suggestions and estimate wait times based on current data
View Historical Trends	Analyze past congestion patterns to plan future trips
Travel History	Displays the past route predictions made

LTA Managers

Function	**Description**
Dashboard Access	View system-wide congestion and usage logs
Generate Monthly Report	Access last month's travel summaries for next month’s optimization and insights
Congestion Planning	Estimate optimal train frequency to meet demand, plus a 10% buffer for redundancy
Update API	Request to update all databases with new API data and refresh all viewers
Push Notifications	Send notifications to users if the current congestion exceeds their personal threshold settings

Admins

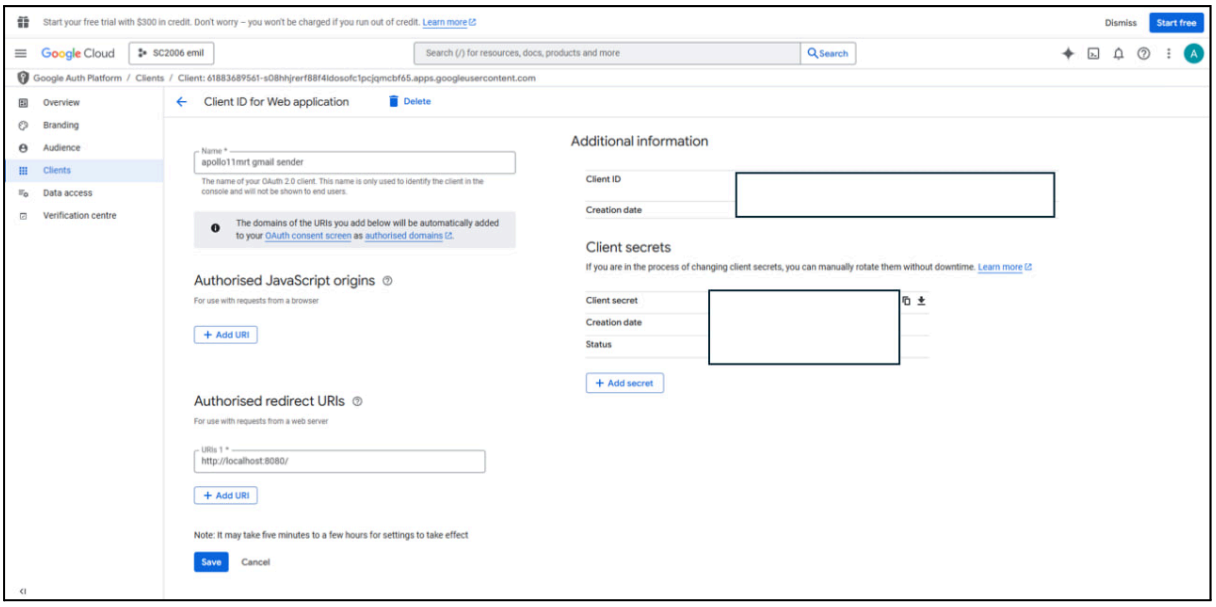
Function	Description
Reset User Password	Reset any user’s password without OTP
Force Refresh	Force a reload for users' UI for real-time data using Server-Sent Events (SSE)
Update All Databases and Force Refresh	Update all databases and simultaneously trigger a force refresh across all clients

2.2.4 Key System Functionality

Available to all users; run as background services)

Functionality	Description
Fetch Real-Time LTA Data	Polls the LTA DataMall API to retrieve real-time congestion data before rendering to users
Email Notifications	Sends congestion alerts to subscribed users via Gmail OAuth 2.0 and Google Cloud API
Log Events	Records user interactions and system-generated congestion statuses into the database

Update Prediction Models	Periodically refines congestion forecasts using historical data and trends
User Session Login Checks	Maintains active user sessions and validates authentication
Role-Based Access Control (RBAC)	Enforces access restrictions on all application routes based on user roles
Ngrok Public Web URL Hosting	Hosts the application publicly using dynamically generated Ngrok URLs
Server-Sent Events (SSE)	Pushes live data (e.g., congestion updates) to the frontend using JavaScript and SSE
Password Hashing with bcrypt	Hashes and stores user passwords securely using the bcrypt library
SQLite3 Databases	Manages all user and congestion data using lightweight SQL-based storage (SQLite3)



(Figure 1: email notification)

```python
# Decorator to require login for a view function
def login_required(f):
    @wraps(f)
    def decorated_function(*args, **kwargs):
        if "username" not in session:
            # If not logged in, redirect to the login page
            return redirect(url_for("login"))
        return f(*args, **kwargs)
    return decorated_function
```

(Figure 2: User Session Login Checks)

```python
# Commuter Home Route
@self.app.route("/commuter/home", methods=["GET"])
@login_required
def commuter_home():

    user = UserFactory.create_user(session["username"])
    if user.get_type() == "Commuter":  # Security check to make sure that only commuters can access this page
        return render_template("commuter/commuter_home.html", username=user.username)
    else:
        return redirect(url_for("login_error"))  # Redirect to error page if user is not Commuter
```

(Figure 3: Role-Based Access Control)

```
 * ngrok tunnel available at: NgrokTunnel: "https://ece3-138-75-5-235.ngrok-free.app" -> "http://localhost:5000"
Starting web server...
 * Serving Flask app 'Server'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
```

(Figure 4:Ngrok Public Web URL server hosting)

2.3 Hardware Interfaces

The Train Overcrowding Analysis and Mitigation System is designed to be compatible with a wide range of modern hardware devices without the need for specialized equipment.

Supported User Devices

- **Smartphones**
    - iOS (version 12 and above)
    - Android (API level 24 and above)
- **Tablets**
    - Both Android and iOS-based tablets are fully supported
- **Laptops and Desktop Computers**
    - Optimized for full-screen, desktop-first usage

Performance & Optimization

- The application is optimized for use on computers, providing a full-featured interface and data visualizations on larger screens.
- It retains full functional parity on mobile devices and tablets, ensuring accessibility and responsiveness across device types.

Hardware Requirements

- **Minimum Requirements:**
    - A modern web browser (Chrome, Firefox, Safari, or Edge – latest two versions)

    - Stable internet connectivity

- **No Specialized Hardware Required:**
  The system is designed to operate without any dependency on specialized hardware. All computations and interactions occur within the browser and server environment, making it lightweight and accessible for all users.

2.4 Software Interfaces

The Train Overcrowding Analysis and Mitigation System is designed for seamless cross-platform compatibility, requiring only a web browser for access. The application functions across both desktop and mobile environments, ensuring accessibility and responsiveness.

2.4.1 APIs Used

Multiple APIs are integrated to support data collection and automated notifications:

API Name	Purpose
LTA Data Mall API	Retrieves real-time and historical MRT train congestion and movement data
Gmail Google OAuth 2.0 API	Enables secure OAuth-based authentication and automated email notifications to users

2.4.2 Database Used

The application utilizes a lightweight and reliable **SQLite 3** database system, organized into separate schemas for user management and train-related data.

Database Tables

- **User Database:**
    - User – User credentials and metadata
    - User_Data – Profile and preference information
    - User_Notification – Notification settings and logs

- **Train Database:**
    - Trains_Data – Static and reference data for train lines
    - Train_Movement – Real-time and historical movement logs
    - Today_Congestion – Congestion levels updated daily
    - Processed_Data – Cleaned and derived data used for analysis

2.4.3 Environments, Frameworks, and Libraries Used

Python Environment

Tool/Library	Purpose
Flask	Web framework for backend logic and API routing
Ngrok	Exposes local Flask server to public internet for testing/demo

Frontend Stack

Technology	Purpose
HTML	Core structure of the web pages
CSS	Styling and visual formatting
JavaScript	Enables interactivity and real-time data refresh (e.g., graph updates)

2.4.4 Design Patterns

The system employs recognized object-oriented design patterns for scalability and real-time responsiveness:

- **Factory Pattern**
    - Used for dynamically creating user objects based on roles (admin, commuter, etc.) and loading role-specific homepage routes.
- **Observer Pattern (via Server-Sent Events - SSE)**
    - Enables real-time updates of graphs and dashboards. The server pushes congestion data to subscribed clients without requiring them to refresh manually.

2.5 Technology Stack Overview

Our codebase is primarily developed in Python, leveraging a variety of powerful libraries and frameworks to handle different aspects of data collection, processing, visualization, and web interaction. Below is a detailed overview of the tools and technologies in use:

1. **Data Collection**

- **requests**
  Used to make HTTP requests for API-based data collection. It facilitates communication with external data sources through RESTful APIs.

2. Data Processing

- **zipfile / zipp**
  Utilized to extract and process compressed `.zip` files containing structured datasets or logs.

- **pandas**
  Core library for reading, transforming, and migrating tabular data—especially Excel files—into structured formats compatible with databases.

3. Data Visualization

- **matplotlib**
  Used to generate static, interactive, and animated visualizations. Essential for plotting charts and illustrating data trends in reports and dashboards.

4. Graph Representation

- **networkx**
  Employed for representing and analyzing complex graph structures, including nodes, edges, and network relationships.

5. Web Application Framework

- **Flask**
  A lightweight WSGI web framework used to build and expose RESTful APIs and user interfaces. It provides the core logic for routing, session handling, and endpoint creation.

6. Public Server Access

- **ngrok**
  Integrated with Flask to expose local servers securely over the internet. This enables external clients to interact with locally hosted applications for development and testing purposes.

2.6 Python Import Dependencies

To host and run the server effectively, our project relies on multiple Python libraries that handle everything from API data collection to graph representation and public hosting.

Required PIP Dependencies

Install all required libraries using the following command:

pip install bcrypt flask google google_auth_oauthlib google-api-python-client pyngrok matplotlib networkx pandas requests

Dependency Descriptions

Package Name	Description
bcrypt	Used for securely hashing passwords and sensitive data
flask	Lightweight Python web framework for serving routes and APIs
google, google-api-python-client, google_auth_oauthlib	Enable integration with Google services and handle OAuth 2.0 authentication
pyngrok	Connects local Flask server to a public URL using ngrok tunnels
matplotlib	Generates charts and visual diagrams for data analysis
networkx	Used to model and analyze graph structures including nodes and edges
pandas	Handles data processing from Excel files and converts them to DB-ready formats

requests	Facilitates HTTP requests for collecting data from APIs

Automated Installation Script

To simplify dependency setup, a bash script named install_dependencies.sh is included. This script can be executed using **Git Bash** or any Bash-compatible terminal.

#!/bin/bash
 # install_dependencies.sh

pip install bcrypt flask google google_auth_oauthlib google-api-python-client pyngrok matplotlib networkx pandas requests

How to Run:

bash install_dependencies.sh

This ensures a consistent setup across all development environments and reduces manual configuration effort.

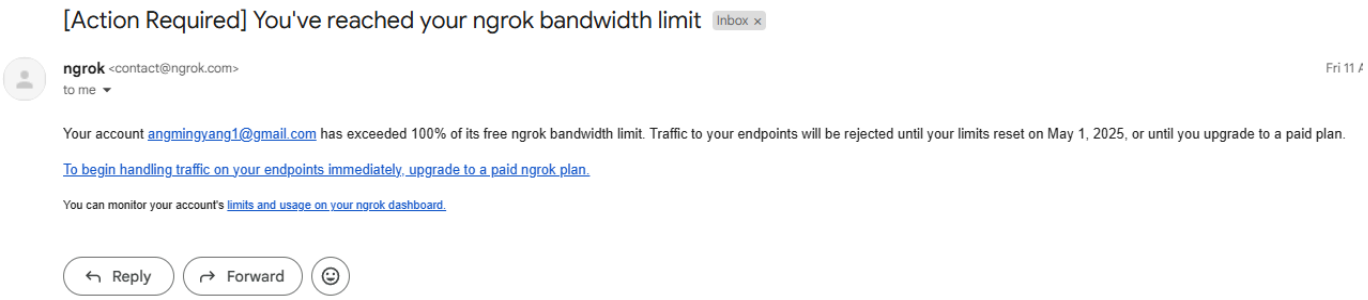
2.7 Design and Implementation Constraints

The Train Overcrowding Analysis and Mitigation System operates under certain constraints due to third-party services and infrastructure limitations. These constraints affect how the system is designed, implemented, and deployed.

1. Ngrok (Public Server Tunneling and Hosting)

Ngrok is used to expose the local Flask server to a public URL for development and testing. However, the free-tier and standard accounts introduce the following limitations:

Constraint	Description
Bandwidth Limits	Monthly bandwidth is capped based on account tier, which may impact prolonged testing
Random Public URLs	Ngrok generates a new random public URL on each startup unless using a paid plan with reserved domains



(Figure 5: bandwidth limit constraint of ngrok)

2. LTA Data Mall API Constraints

LTA Data Mall APIs are used for retrieving public transport data. However, API design and usage limitations include:

Constraint Type	Description
Rate Limits	Excessive API calls may result in daily bandwidth caps or temporary account blocks

Data Update Frequency	Not all APIs provide real-time data. Many endpoints have static refresh cycles

Specific Endpoint Constraints:

API Endpoint URL	Description	Update Frequency
`api_url_7`: `https://datamall2.mytransport.sg/ltaodataservice/PV/ODTrain`	Passenger Volume – Origin-Destination Train	Monthly (10th of every month)
`api_url_8`: `https://datamall2.mytransport.sg/ltaodataservice/PV/Train`	Passenger Volume – Per Train Line	Monthly (10th of every month)
`api_url_11`: `https://datamall2.mytransport.sg/ltaodataservice/TrainServiceAlerts?TrainLine=EWL`	Train Line Service Alerts	Ad hoc / Inconsistent refresh
`api_url_24`: `https://datamall2.mytransport.sg/ltaodataservice/PCDRealTime?TrainLine=EWL`	Real-time Platform Crowding Data	Every 10 minutes
`api_url_25`: `https://datamall2.mytransport.sg/ltaodataservice/PCDForecast`	Platform Crowding Forecast	Once daily at midnight

2.8 User Documentation

This section outlines the steps to set up and run the Train Overcrowding Analysis and Mitigation System server, as well as information regarding account access and Gmail configuration.

2.8.1. Server Setup Instructions

To run the system locally and expose it publicly using Ngrok:

Step 1: Install Dependencies

You can install the required Python libraries using one of the two methods:

Option A: Install from the provided markdown file:

```
pip install -r Pip_Requirements.md
```

Option B: Run the automated script using Git Bash:

```
bash install_dependencies.sh
```

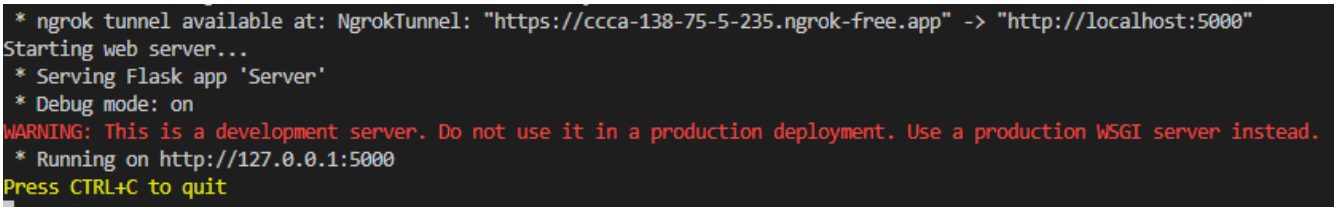
Step 1.1: Download the Databases Train database & User Database outside the Lab 4&5 folder.

Step 1.2: Use admin account to update all databases prior to running the commuter and LTA manager. (“Use Admin Only For The First Time”).

Step 2: Run the Server

Launch the server by executing:

```
python server.py
```



(Figure 6: Ngrok Tunnel and Flask Server Startup Output)

Upon execution, the terminal will display an **Ngrok-generated public URL**, e.g.:

https://ccca-138-75-5-235.ngrok-free.app/

This is the public address clients can access the system through.

2.8.2. Email Functionality & Gmail Configuration

The system includes email-based functions (e.g., password reset). During the first run, users may be prompted to sign in to a Google account depending on the validity of the `token.json` file.

Gmail Used in System:

- **Username:** Apollo11mrt@gmail.com
- **Password:** sc2006group1

Note:
Google Cloud Gmail API requires **2-Factor Authentication (2FA)**.

Therefore, initial setup may require assistance to obtain the 2FA verification.

For 2FA setup or if you wish to **use a different Gmail client**, update the `credentials.json` file with new credentials downloaded from the Google Cloud Console.

2.8.3. Enquiries & Support

For support or issues related to server setup or email configuration, contact:

- **angmingyang1@gmail.com**
- **angm0027@e.ntu.edu.sg**

2.8.4. Test User Accounts

Below is a list of pre-configured test accounts categorized by user roles.

Commuters

Username	Password	Email	Full Name	Cluster
User1	password123	bancrusher10@gmail.com	John Doe	101
User2	password123	bancrusher10@gmail.com	Alice Smith	80
User3	password123	bancrusher10@gmail.com	Bob Johnson	101

LTA Managers

Username	Password	Email	Full Name	Cluster
LTA1	password123	bancrusher10@gmail.com	Sarah Lee	101

LTA2	password123	bancrusher10@gmail.com	Michael Tan	101

Admins

Username	Password	Email	Full Name	Cluster
MY	password123	bancrusher10@gmail.com	Ming Yang Ang	101
Admin1	password123	bancrusher10@gmail.com	David Chong	101

Invalid User (for testing)

Username	Password	Email	Full Name	Cluster	User Type
fakeuser	password123	bancrusher10@gmail.com	fake fake	101	wrong_user_type

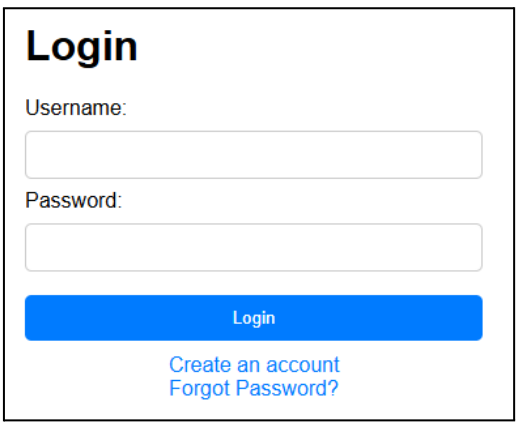
3. System Features

3.1 System Features - Main Menu

The *Train Overcrowding Analysis and Mitigation System* provides several key features designed to improve the commuting experience of MRT users and support transport authorities with data insights. The features are categorized into modules based on user roles: Commuters and LTA Administrators.

3.1.1 User Login and Registration

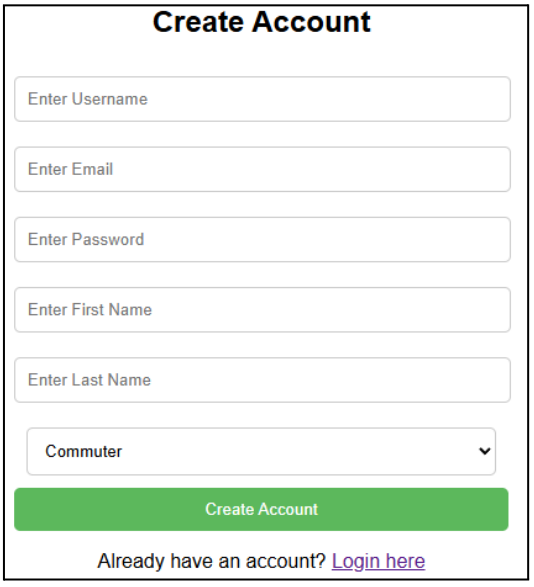
Attribute	Details
Description	Allows users to log in or register using their username and password.
Inputs	`Username`, `Password`
Outputs	Access to user-specific dashboard and issuance of session token
Actors	Any user (commuter, LTA manager, admin)



(Figure 7: User Login Interface)

3.1.2 Registration

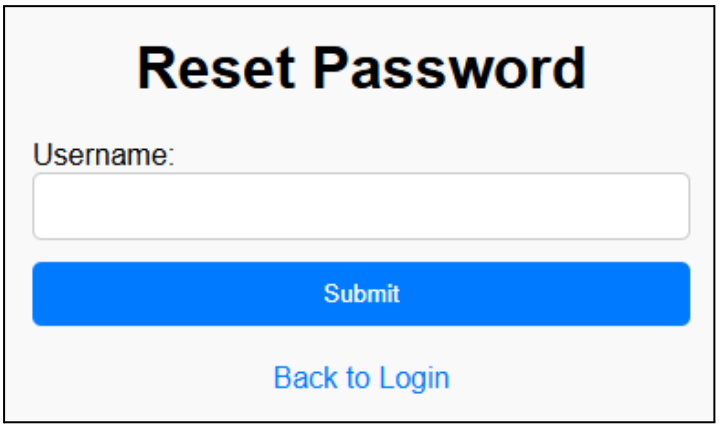
Attribute	Details
Description	Allows users to create a new account in the system.
Inputs	`Username`, `Email`, `Password`, `First Name`, `Last Name`, `User_type`
Outputs	Success message on valid registration, or failure message on validation error
Actors	Any user



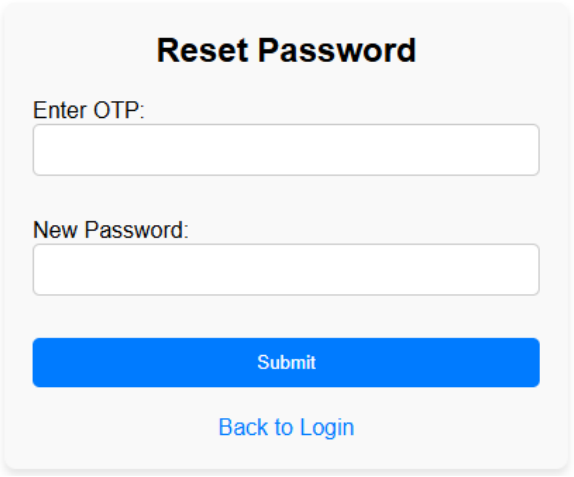
(Figure 8: User Registration Interface)

3.1.3 Reset Password

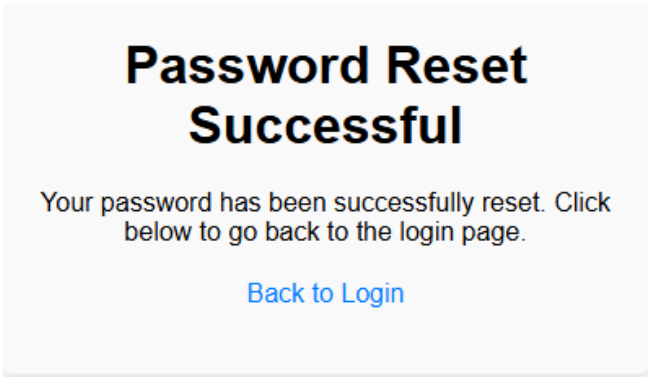
Attribute	Details
Description	Allows users to reset their account password using email verification.
Inputs	`Username`, `Email`
Outputs	One-time password (OTP) sent to registered email for verification, or error (invalid OTP/username)
Actors	Any user
Technologies	Gmail OAuth 2.0, Google Cloud API



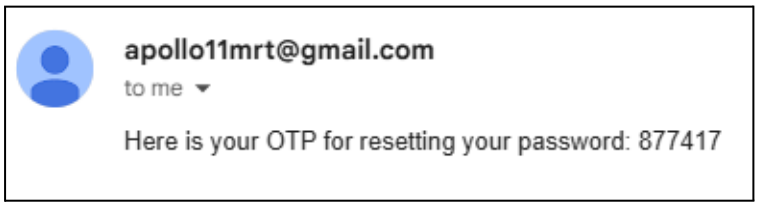
(Figure 9: Reset Password Login)



(Figure 10: Reset Password- New Password)



(Figure 11: Password Successfully Reset)



(Figure 12:Email OTP for Password Reset)

3.2 System Features – Commuter

Sample Account

- **Username:** User1
- **Password:** password123

3.2.1 User Settings

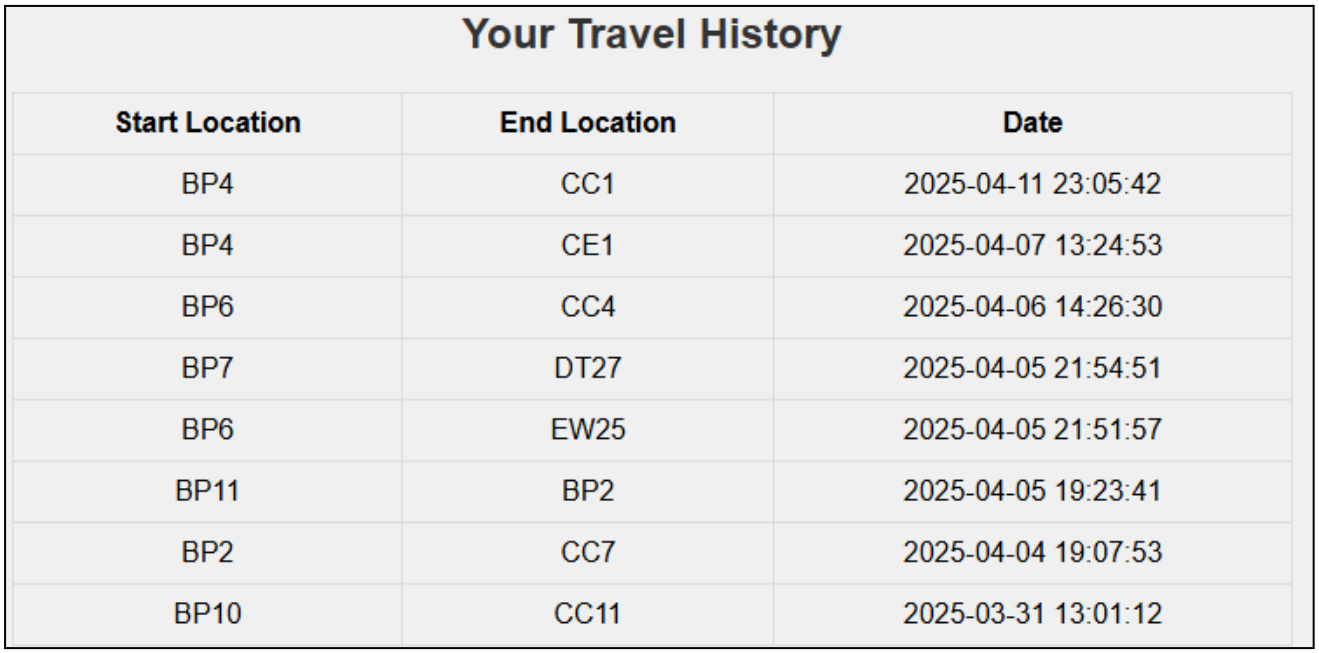
Attribute	Details
Description	Allows users to update personal details and set a congestion threshold value.
Inputs	`Email`, `First Name`, `Last Name`, `Threshold`
Outputs	Success confirmation or failure message
Actors	Commuter
Technologies	SQLite3 Database Queries – `User` Table

o

(Figure 13: User Updation Information Interface)

3.2.2 Travel History

Attribute	Details
Description	Displays all past route prediction history for the user.
Inputs	NIL
Outputs	History table showing Start Location, End Location, and Date
Actors	Commuter
Technologies	SQLite3 Database Queries – User_Data Table



Your Travel History

Start Location	End Location	Date
BP4	CC1	2025-04-11 23:05:42
BP4	CE1	2025-04-07 13:24:53
BP6	CC4	2025-04-06 14:26:30
BP7	DT27	2025-04-05 21:54:51
BP6	EW25	2025-04-05 21:51:57
BP11	BP2	2025-04-05 19:23:41
BP2	CC7	2025-04-04 19:07:53
BP10	CC11	2025-03-31 13:01:12

(Figure 14: User Travel History)

3.2.3 View Real-Time Congestion

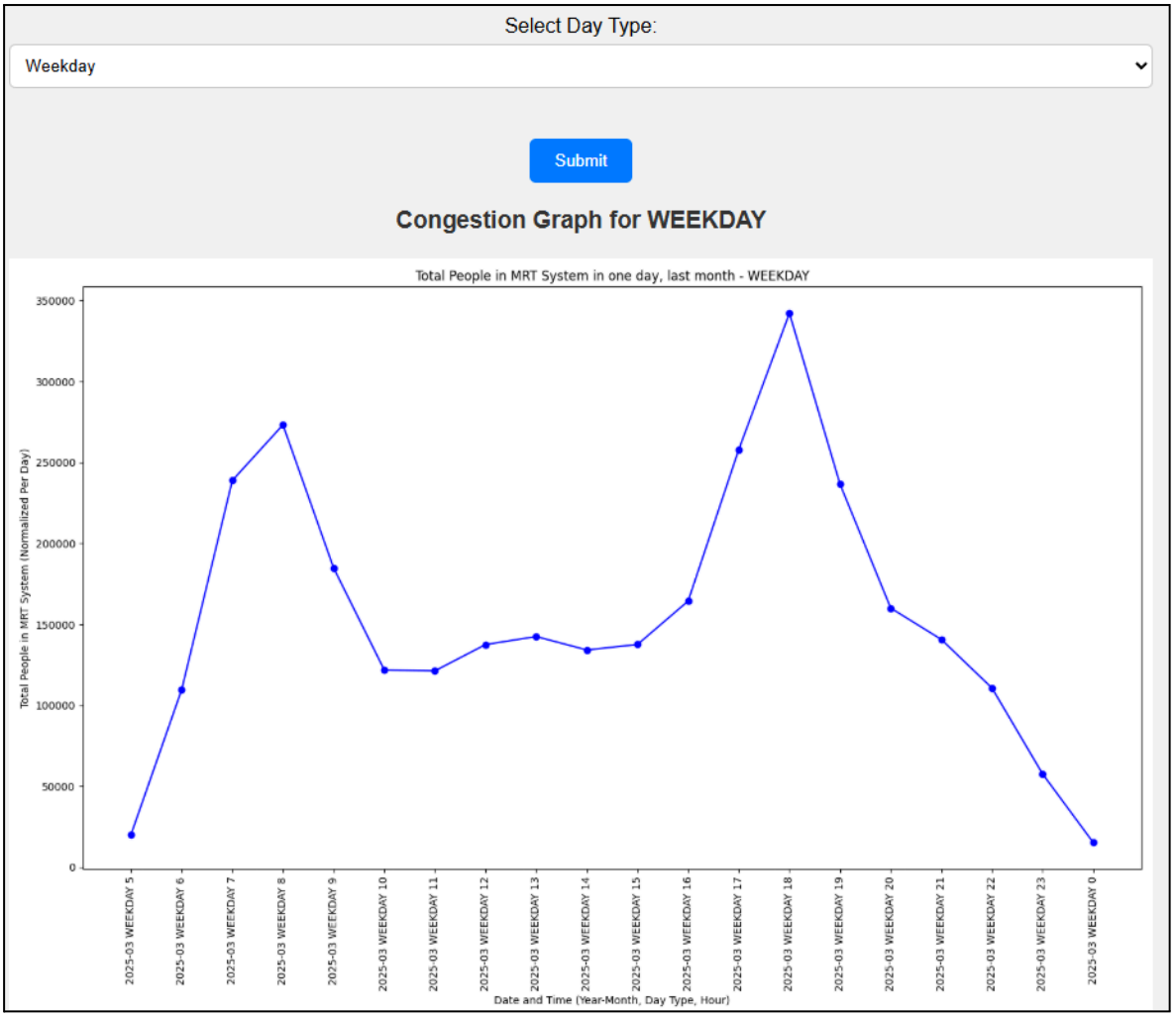
Attribute	Details
Description	Displays real-time crowd levels for MRT stations using LTA's TrainCrowdData API.
Inputs	Selected MRT Line
Outputs	Numerical table and congestion graph image
Actors	Commuter
Technologies	`requests`, LTA DataMall API, Server-Sent Events (SSE) for live refresh

(Figure 15: Real-Time Congestion)

^ *During Peek hours like 7am or 6pm* ^

3.2.4 Historical Data

Attribute	Details
Description	Displays last month's total MRT load per day, broken down by hour and categorized by weekday or weekend.
Inputs	`Weekday` or `Weekend`
Outputs	Congestion graph
Actors	Commuter
Technologies	SQLite3 Database Queries – `Train_Movement` Table, Load Calculation Algorithms
Assumptions	Load is calculated as: `(current_hour_cumulative_tap_in - previous_hour_cumulative_tap_out)`



(Figure 16: Historical Data for a Weekday)

3.2.5 Route Prediction

Attribute	Details
Description	Displays graphs and historical congestion data to help users plan optimal travel routes.
Inputs	Start Location, End Location
Outputs	List of train stations along the route and estimated travel time
Actors	Commuter
Technologies	Dijkstra's Algorithm, Graph Structures, networkx Library

Shortest Path:

['BP1', 'NS4', 'NS3', 'NS2', 'NS1', 'EW14', 'EW13', 'EW12', 'EW11', 'EW10', 'EW9', 'EW8', 'CC9', 'CC8', 'CC7', 'CC6']

Estimated Time:

40.0 minutes

Our MRT Map

(Figure 17: Route Prediction)

3.3 System Features – LTA

Sample Account

- **Username:** LTA1
- **Password:** password123

3.3.1 Get Monthly Report

Attribute	Details
Description	Generates a monthly report based on MRT usage data from the previous month.
Inputs	`MRT Line`, `Day Type` (Weekday/Weekend), `Hour`
Outputs	- Total Tap-Ins - Monthly Tap-In Proportions by MRT Line - Top 5 Busiest Stations (Tap-Ins) - Top 5 Busiest Stations (Tap-Outs) - Load Distribution Graph - Tap-In vs Tap-Out Discrepancy Chart
Actors	LTA Manager

Report Summary

Day Type: WEEKDAY

Total Tap-Ins: 81484167

Monthly Tap-In Proportions

BP: 3158437

CC: 13511679

DT: 14333828

EW: 20331635

NE: 7904873

NS: 18178072

TE: 4065643

Top 5 Busiest Stations (Tap-Ins) Last Month's Data

Rank	Station Code	Total Tap-Ins
1	EW24/NS1	2036073
2	TE14/NS22	1847154
3	NE12/CC13	1682755
4	EW12/DT14	1616099
5	NS21/DT11	1605219

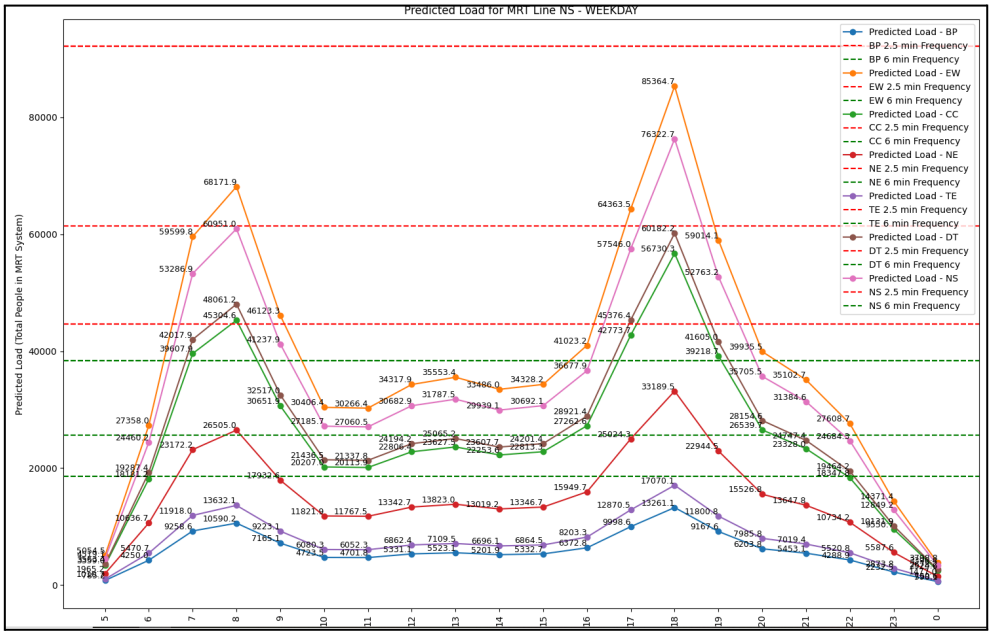
Top 5 Busiest Stations (Tap-Outs) Last Month's Data

Rank	Station Code	Total Tap-Outs
1	EW24/NS1	2025794
2	TE14/NS22	1953033
3	NS9/TE2	1897951
4	NE12/CC13	1725369
5	EW12/DT14	1681707

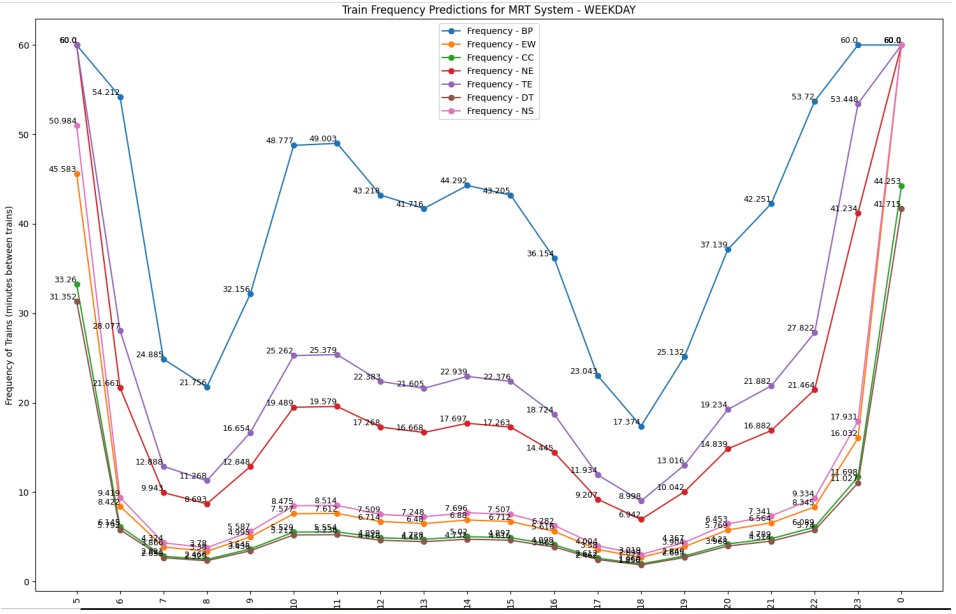
(Figure 18: Monthly Report Summary)

3.3.2 Congestion Planning

Attribute	Details
Description	Helps LTA Managers optimize train scheduling to reduce costs while meeting commuter demand.
Inputs	Day Type, Hour
Outputs	- Congestion Prediction Graph - Train Frequency Graph *(Based on actual load vs. theoretical capacity at varying frequencies)*
Actors	LTA Manager



(Figure 19: Predicted Load for Congestion Planning)



(Figure 20: Train Frequency Prediction on a Weekday)

Assumptions: Train Capacities

The following train capacities are used in the congestion predictions:

East-West Line (EW): 1920 passengers
North-South Line (NS): 1920 passengers
North-East Line (NE): 1920 passengers
Circle Line (CC): 931 passengers
Downtown Line (DT): 931 passengers
Thomson-East Coast Line (TE): 1280 passengers

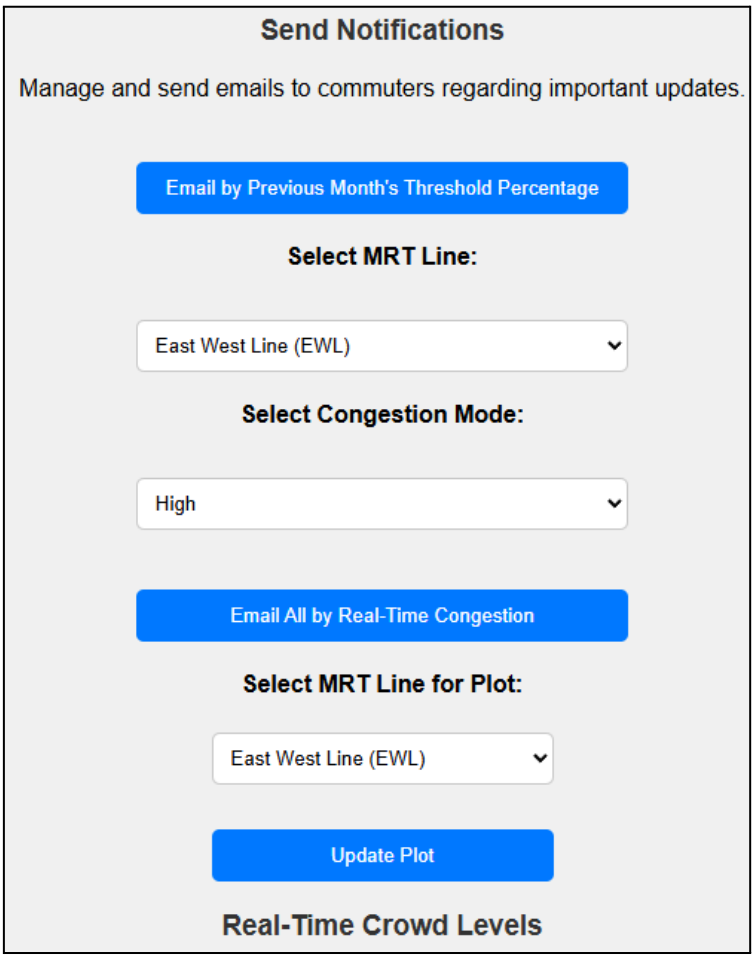
Source: Wikipedia

(Figure 21: Assumption for Train Capacities)

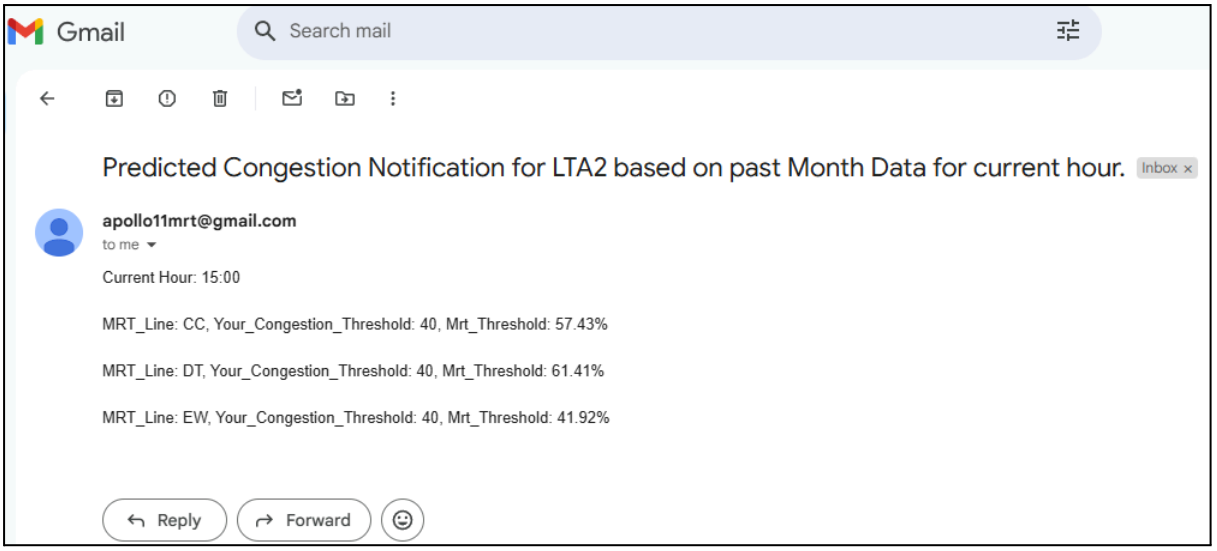
https://en.wikipedia.org/wiki/List_of_Singapore_MRT_and_LRT_rolling_stock

3.3.3 Email Notifications

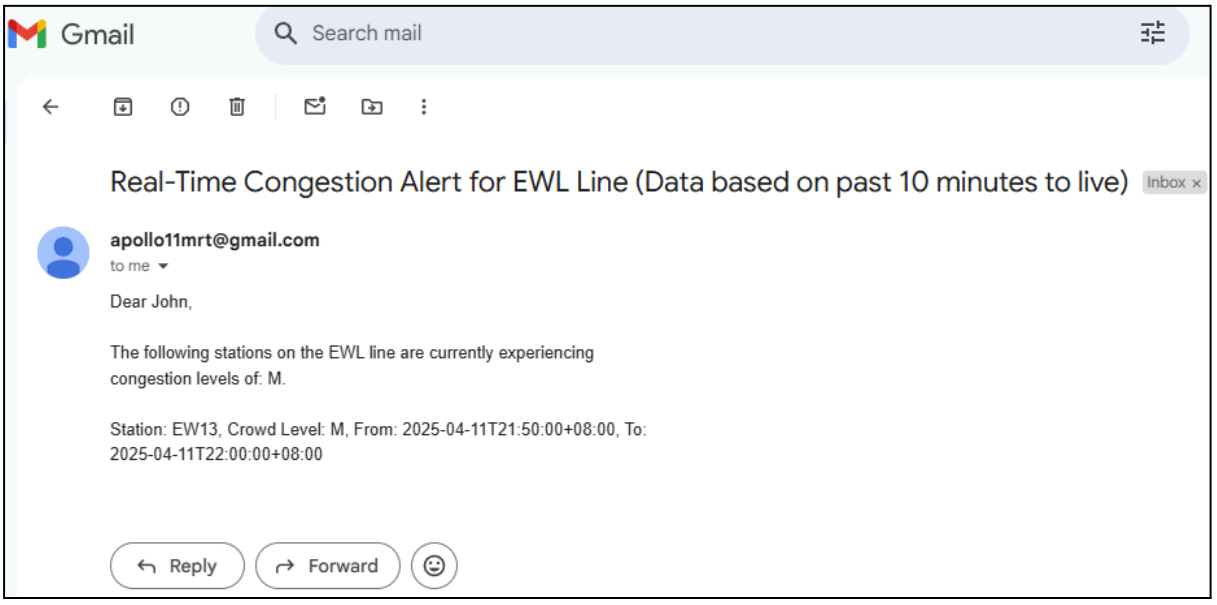
Attribute	Details
Description	Sends real-time congestion alerts via Gmail using OAuth 2.0 and Gmail API to all subscribed users. Notifications are triggered based on user-defined threshold percentages or system-detected congestion levels (High, Medium, Low). A real-time congestion graph is included for reference.
Inputs	MRT Line, Congestion Mode
Outputs	Email notifications sent to all subscribed users
Actors	LTA Manager



(Figure 22: Sending Email Notification)



(Figure 23: Email Notification)



(Figure 24: Email Notification)

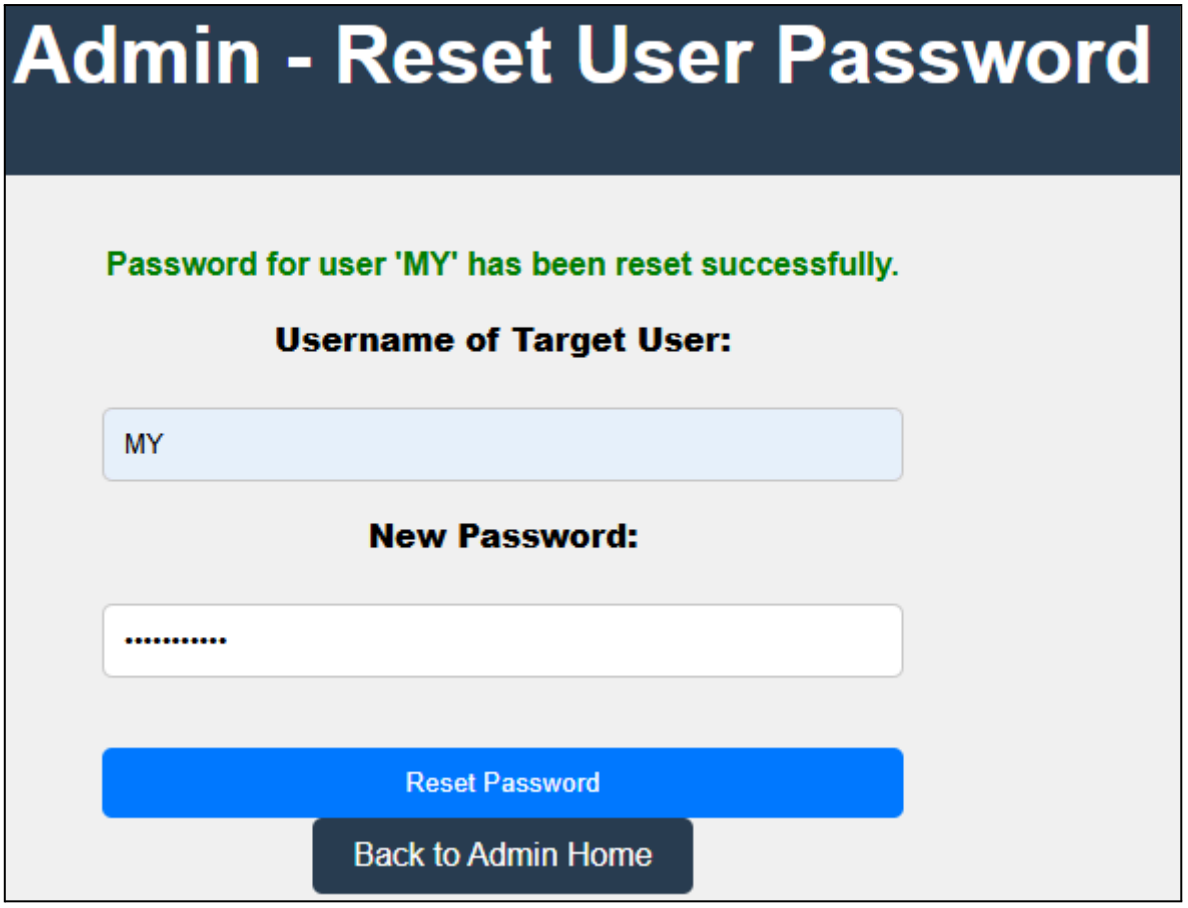
3.4 System Features – Admin

Sample Account

- **Username:** Admin1
- **Password:** password123

3.4.1 Reset User Password

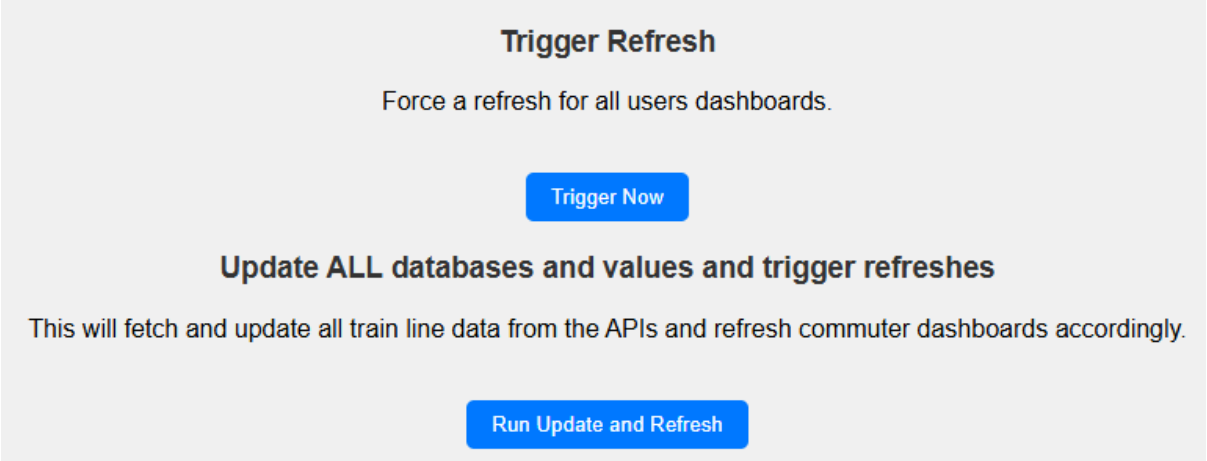
Attribute	Details
Description	Allows the admin to reset any user's password directly, bypassing OTP verification.
Inputs	`Username`, `New Password`
Outputs	Password reset success message
Actors	Admin



(Figure 25: Reset User Password)

3.4.2 Update All Databases and Trigger Refreshes

Attribute	Details
Description	Updates all four API data sources and pushes a live refresh to all active users using Server-Sent Events (SSE).
Inputs	Trigger Button
Outputs	- Client browsers refresh with updated data - Server updates all databases with new API data
Actors	Admin



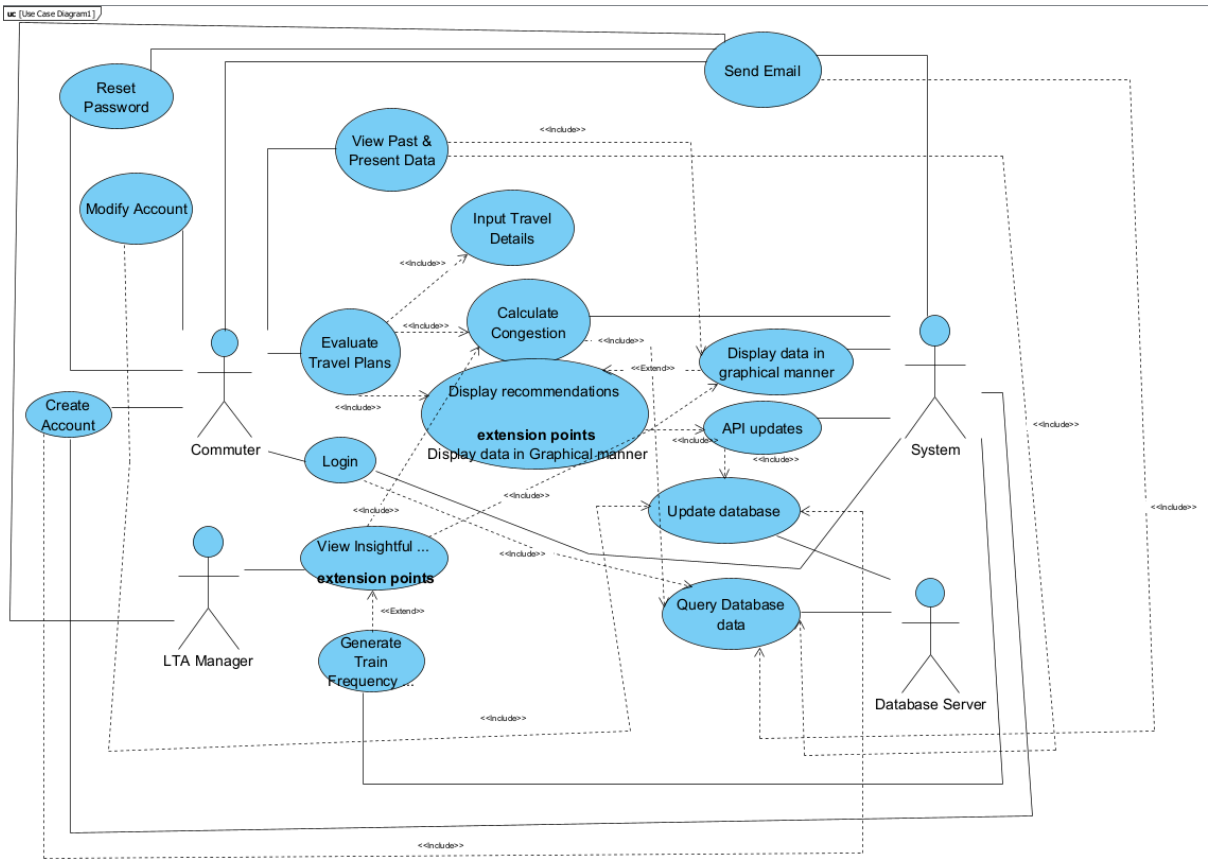
(Figure 26: Trigger Refresh for Updation of Database)

4. Diagrams and Mapping Requirements

4.1 Use Case

Our Product is broken down into 4 layers: Frontend (Presentation Layer), Control Layer (App Logic Layer), Interfaces (Data Access Layer) and the Persistent (Database).

4.1.1 Use Case Diagram



(Figure 27:Use Case Diagram)

4.1.2 Use Case Description

UC-001: Login

Field	Details
Use Case ID	UC-001
Use Case Name	Login
Actor	Commuter
Description	The initial starting page when the user enters
Flow of Events	1. The user enters a valid username and corresponding password 2. The user clicks on the enter login button. 3. System verifies credentials 4. The user is brought to the home page
Alternativ Flows	● Invalid Credentials: If the username or password is incorrect, the system displays an error message and prompts the user to try again. ● Unauthorized Access: If the user manually changes the URL to access restricted pages without authentication, the system redirects them to an authentication error page.
Exceptions	● System downtime or connectivity issues prevent login. ● The Server fails to respond within a given time frame. ● The user forgets their password and needs a reset.
Includes	Password Recovery [medium priority]
Assumptions	The user has an existing account.

UC-002: View Real-Time Congestion

Field	**Details**
Use Case ID	UC-002
Use Case Name	View Real-Time Congestion
Actor	Commuter
Description	Displays Real-Time Congestion levels for all stations from Low, Medium, and High.
Flow of Events	1. The user selects "View Real-Time Congestion" from the home page. 2. The system calls the API to retrieve real-time congestion data. 3. The system processes the data and displays the congestion levels for each station as Low, Medium, or High. 4. The user can view the congestion levels and make informed decisions about travel routes.
Alternativ Flows	NIL
Exceptions	● API returns no data: indicating it is not available, Display an “API not Available web page” ● Partial Data Retrieval: API retries 2 more times and Displays an “API not Available web page” if the issue persists
Includes	NIL.
Assumptions	The user is logged in. All users have permission to access Real-time Data.

UC-003: View Historical Congestion Data

Field	Details
Use Case ID	UC-003
Use Case Name	View Historical Congestion Data
Actor	Commuter
Description	The user selects a past date range to analyse congestion trends.
Flow of Events	1. The user selects “View Historical Congestion Data” 2. The user selects a past date range and stations. 3. The system fetches API congestion data. 4. The system displays trends using graphs or heat maps.
Alternative Flows	NIL
Exceptions	● If database retrieval is empty, the system will display a “no data found” ● If the data input is incorrect, the system will display "Invalid date format."
Includes	Check API for updates
Special Requirements	Data must be stored for at least 2 months.
Assumptions	The user understands how to interpret the congestion map.
Notes & Issues	Could be upgraded in the future for clicking on station icons instead of the dropdown box [low priority]

UC-004: Modify Congestion Account / Notifications

Field	Details
Use Case ID	UC-003
Use Case Name	Modify Account / Congestion Notifications
Actor	Commuter
Description	The user sets a notification threshold for congestion alerts based on predefined thresholds.
Flow of Events	1. The user selects the user settings button from the home page. 2. The users select Modify Congestion Notifications 3. The user sets a congestion threshold (e.g., 80% occupancy). 4. The System updates the database with the database interface 5. The system monitors real-time congestion levels. 6. The system sends push notifications when the threshold is exceeded.
Alternative Flows	If the threshold is set above 100%, notifications are disabled and alerts are not sent.
Exceptions	NIL
Includes	Update Database
Special Requirements	NIL
Assumptions	NIL
Notes & Issues	In the Future, allow users to set daily time range for allowed notification.

UC-005: Display Top Routes Suggestions with Predicted Congestions of the past 2 months

Field	Details
Use Case ID	UC-005
Use Case Name	Display Top Routes Suggestions with Predicted Congestions of the past 2 months
Actor	Commuter
Description	The system provides top routes to users, considering congestion levels and train swapping durations.
Flow of Events	1. The user selects “Display Top Routes Suggestions” from the home page. 2. The user selects their start and end point 3. The System updates the database with the database interface from the API 4. The System queries the database from the database interface for the relevant information. 5. The System calculates the shortest route using Dijkstra algorithm considering multiple MRT lines. 6. The System Displays the top 3 routes found
Alternative Flows	If no routes were returned, the system displays a "No viable routes found. Please try again later."
Exceptions	Empty Database, API failure, Network failure
Includes	Update Database, Query Database
Special Requirements	Data should display within 3 seconds of the prompt

Assumptions	The user is logged in
Notes & Issues	Future updates will display how the calculations are made for the specific route for transparency and feedback

UC-006: Generate Travel Report (LTA/SMRT)

Field	Details
Use Case ID	UC-006
Use Case Name	Generate Travel Report
Actor	LTA Manager
Description	Generates key insights like average passenger per cabin, average peak hours, and daily prediction graphs.
Flow of Events	1. LTA Manager clicks on “Generate Travel Report “ from the home page 2. The System updates the database with the database interface from the API 3. The System queries the database from the database interface for the relevant information. 4. The System calculates averages of the relevant information 5. The System displays a detailed report.
Alternative Flows	There is an error when collecting the relevant information. The layers will return None to the system for the system to display “Error generating report”
Exceptions	The API is down, The Database is empty.
Includes	Update Database, Query Database
Special Requirements	The report should generate within 5 seconds
Assumptions	User is logged in as a LTA manager

Notes & Issues	NIL

UC-007: Generate Recommended Train Scheduling (LTA/SMRT)

Field	Details
Use Case ID	UC-007
Use Case Name	Generate Optimise Train Scheduling
Actor	LTA Manager
Description	The system provides train schedule optimisation based on past congestion trends.
Flow of Events	1. LTA Manager clicks on “Generate Recommended Train Scheduling“ from the home page 2. The System updates the database with the database interface from the API 3. The System queries the database from the database interface for the relevant information. 4. The System calculates the recommended frequency for each hour for each day in a week 5. The system displays the data on a graph.
Alternative Flows	There is an error when collecting the relevant information. The layers will return -1 to the system for the system to display “Error generating report”
Exceptions	The API is down, The Database is empty.
Includes	Update Database, Query Database.
Special Requirements	NIL
Assumptions	The data used is from the past 2 months.
Notes & Issues	NIL

UC-008: Update Database

Field	Details
Use Case ID	UC-008
Use Case Name	Update Database
Actor	System
Description	Uses Datamall API to update the database
Flow of Events	1. The System receives an Update request 2. The System sends a request to the Datamall API 3. The System receives data from the API 4. The System checks the recency of the database data 5. The System sends the data to the database interface to update the data if not updated 6. The database is successfully updated.
Alternative Flows	There is an error when collecting or updating the relevant information. The layers will return None to the system for the system to display “Error updating data”
Exceptions	The API is down, The Database is unreachable, The data is already the latest
Includes	NIL
Special Requirements	NIL
Assumptions	NIL
Notes & Issues	NIL

UC-009: Query Database

Field	Details
Use Case ID	UC-009
Use Case Name	Query Database
Actor	System
Description	The system sends a query request to the database interface
Flow of Events	7. The System receives a query request 8. The System sends a request to the database interface 9. The database interface queries the relevant information 10. The database interface returns the relevant information 11. The System returns the data to the relevant method that calls for the data.
Alternative Flows	There is an error when collecting or updating the relevant information. The layers will return None to the system for the system to display “Error updating data”
Exceptions	The Database is unreachable
Includes	Update Database, Query Database
Special Requirements	NIL
Assumptions	NIL
Notes & Issues	NIL

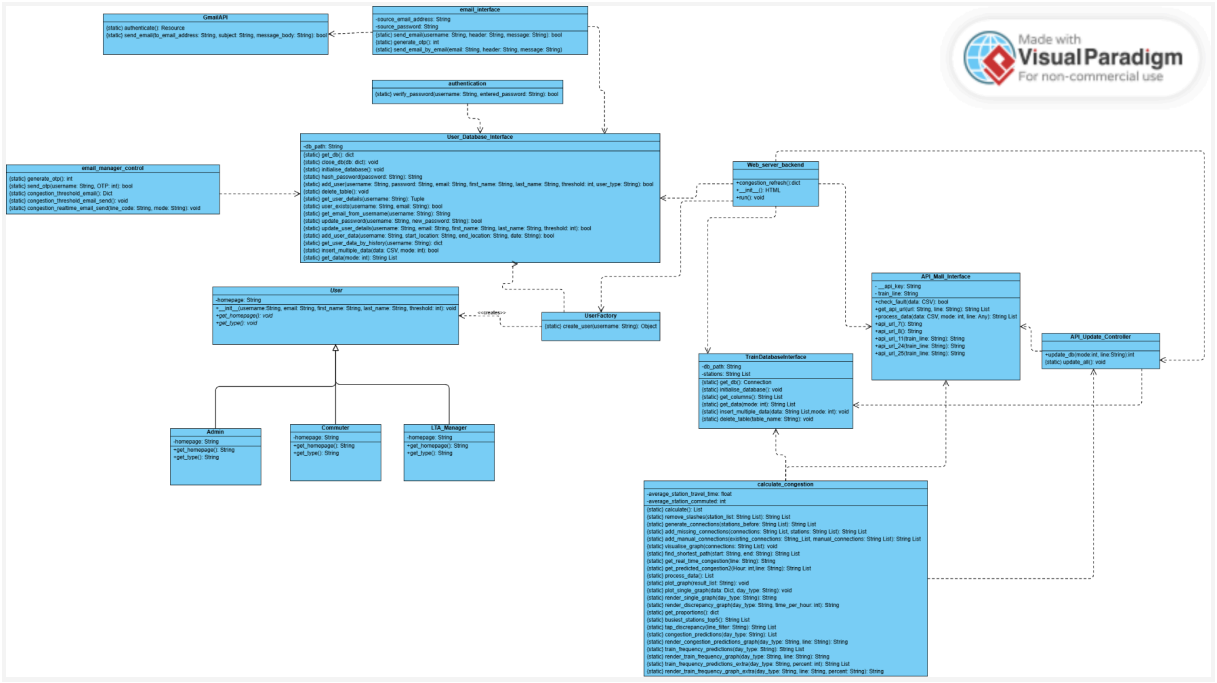
UC-010: Reset Password

Field	Details
Use Case ID	UC-010
Use Case Name	Reset Password
Actor	Commuter
Description	The User unable to log in, reset his/her password by email
Flow of Events	1. The user clicks on reset password in the login menu 2. The system prompts the user to input his username 3. The system sends a one-time password to the user's email 4. The user inputs the one-time password 5. The system prompts for a new password 6. The system updates the account with the new password
Alternative Flows	The One-time password imputed is invalid, Resend a new OTP and try again
Exceptions	The Database is unreachable, One time Password incorrect.
Includes	Update Database, Query Database, Send Email
Special Requirements	NIL
Assumptions	NIL
Notes & Issues	NIL

UC-011: Send Email

Field	Details
Use Case ID	UC-011
Use Case Name	Send Email
Actor	System
Description	The System Pushes an email notification
Flow of Events	1. System is prompt to send a notification. 2. The system pushes a notification to user's email.
Alternative Flows	Email Parameters are invalid.
Exceptions	NIL
Includes	Query Database
Special Requirements	NIL
Assumptions	NIL
Notes & Issues	NIL

4.3 UML Class Diagram

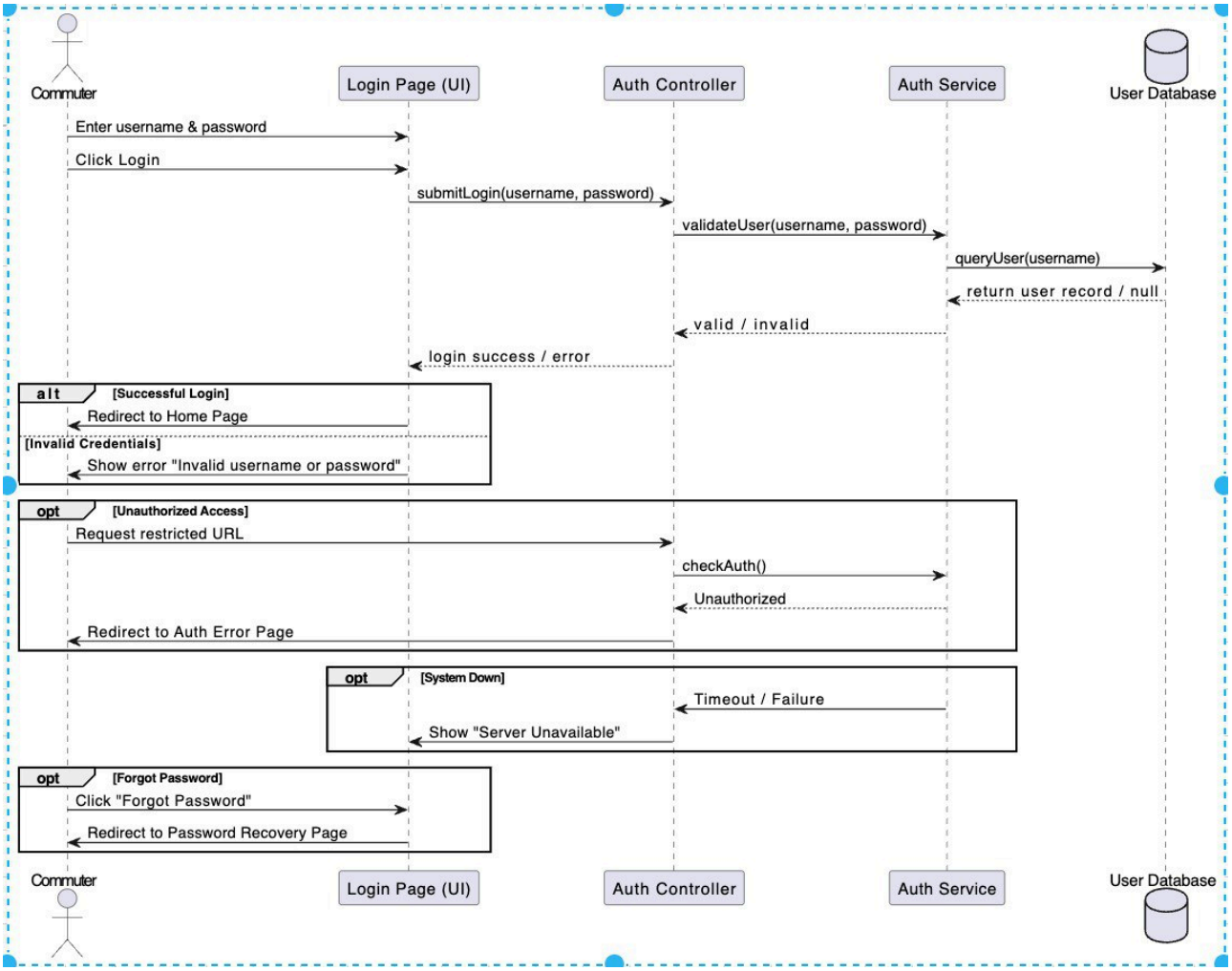


(Figure 28:UML Class Diagram)

Home page, -> other functions
Home page -> settings

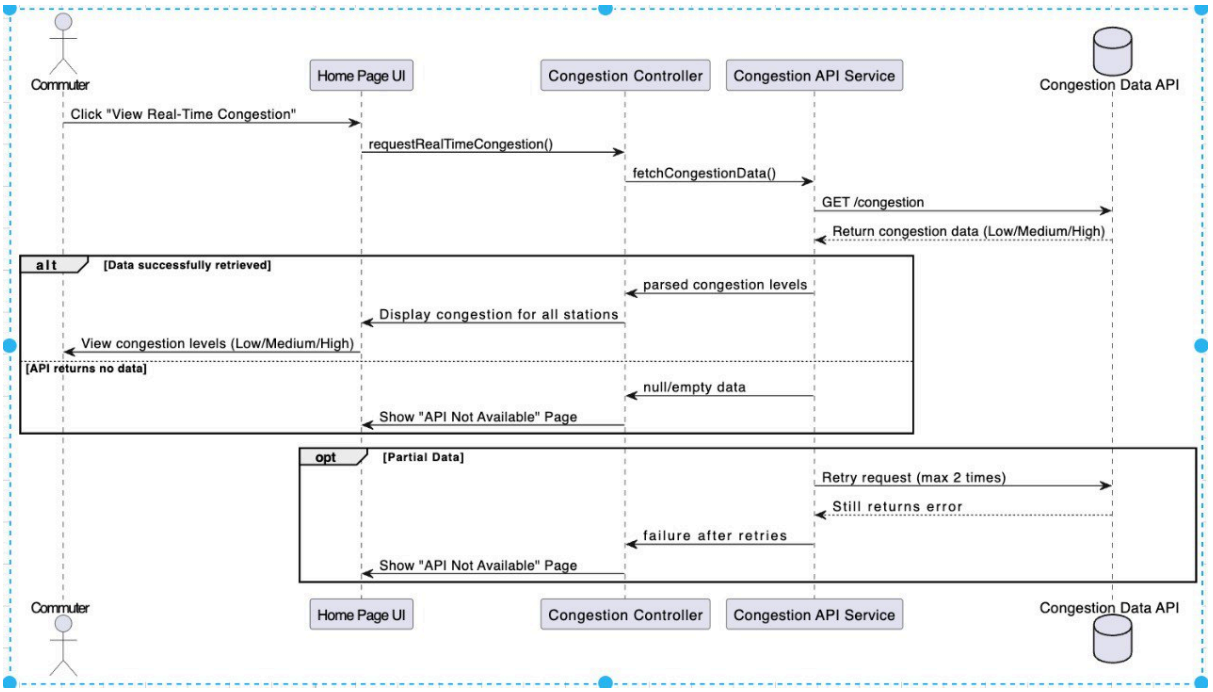
4.4 Sequence Diagram

S1:Login



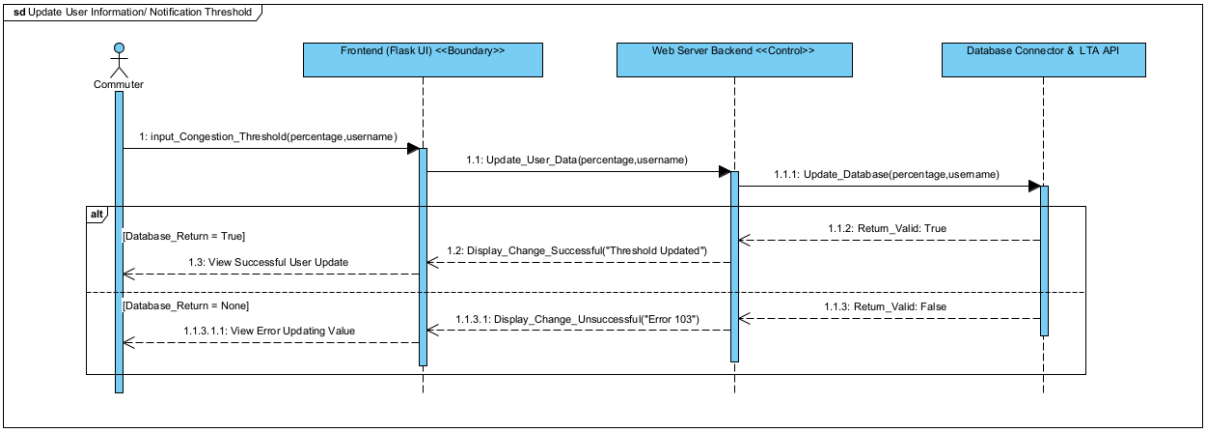
(Figure 29:Sequence Diagram for Login)

S2: Check Real-Time Congestion



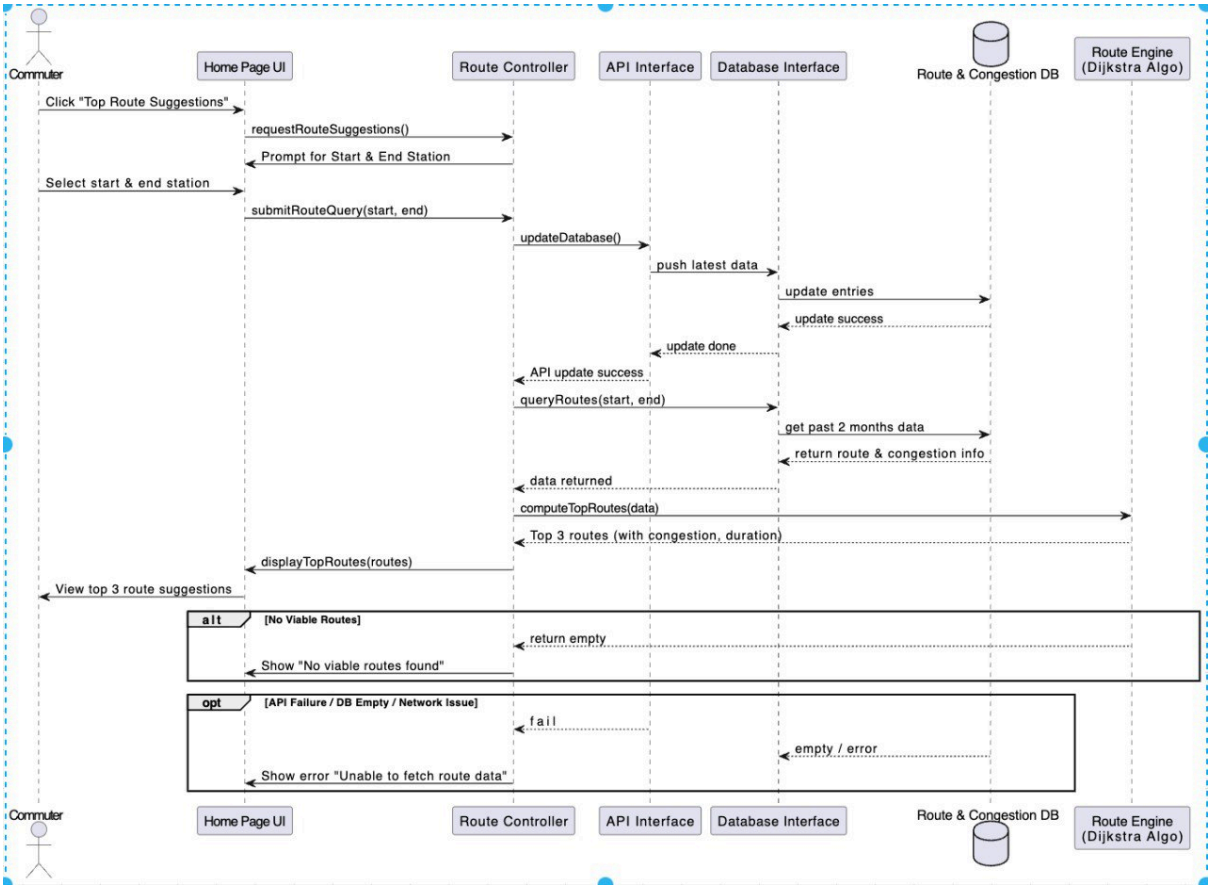
(Figure 30:Sequence Diagram for Checking Real-Time Congestion)

S3: Update User Information / Notification threshold



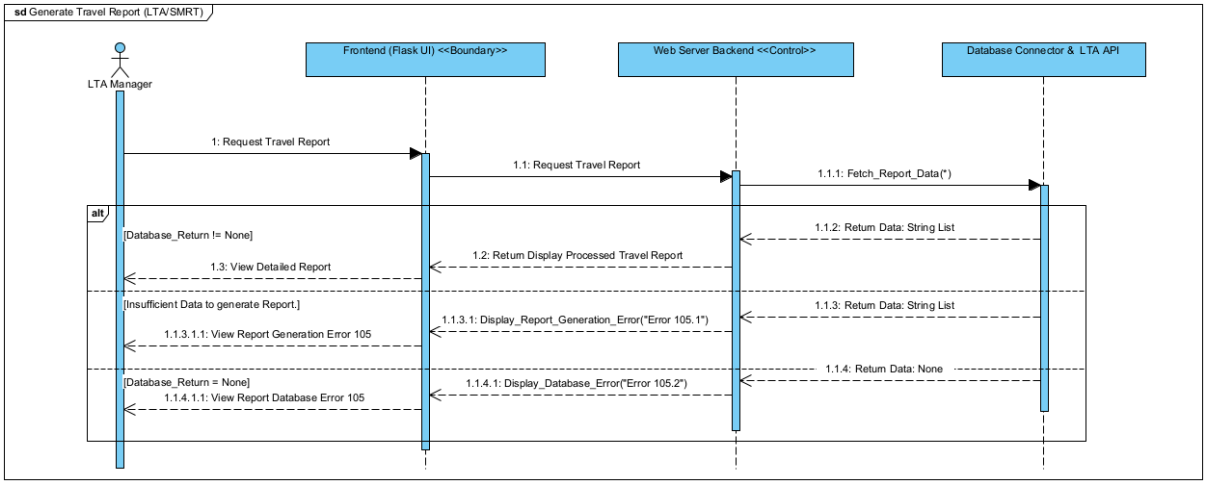
(Figure 31:Sequence Diagram for Notification Threshold)

S4: Alternative Routes Suggestion



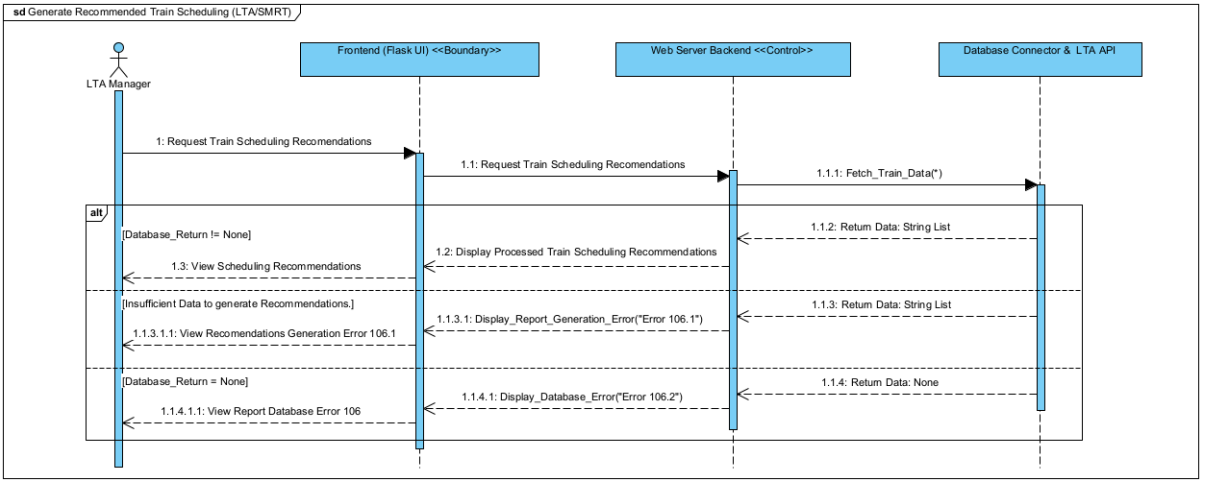
(Figure 32:Sequence Diagram for Alternative Route Suggestion)

S5: Generate Travel Report (LTA/SMRT)



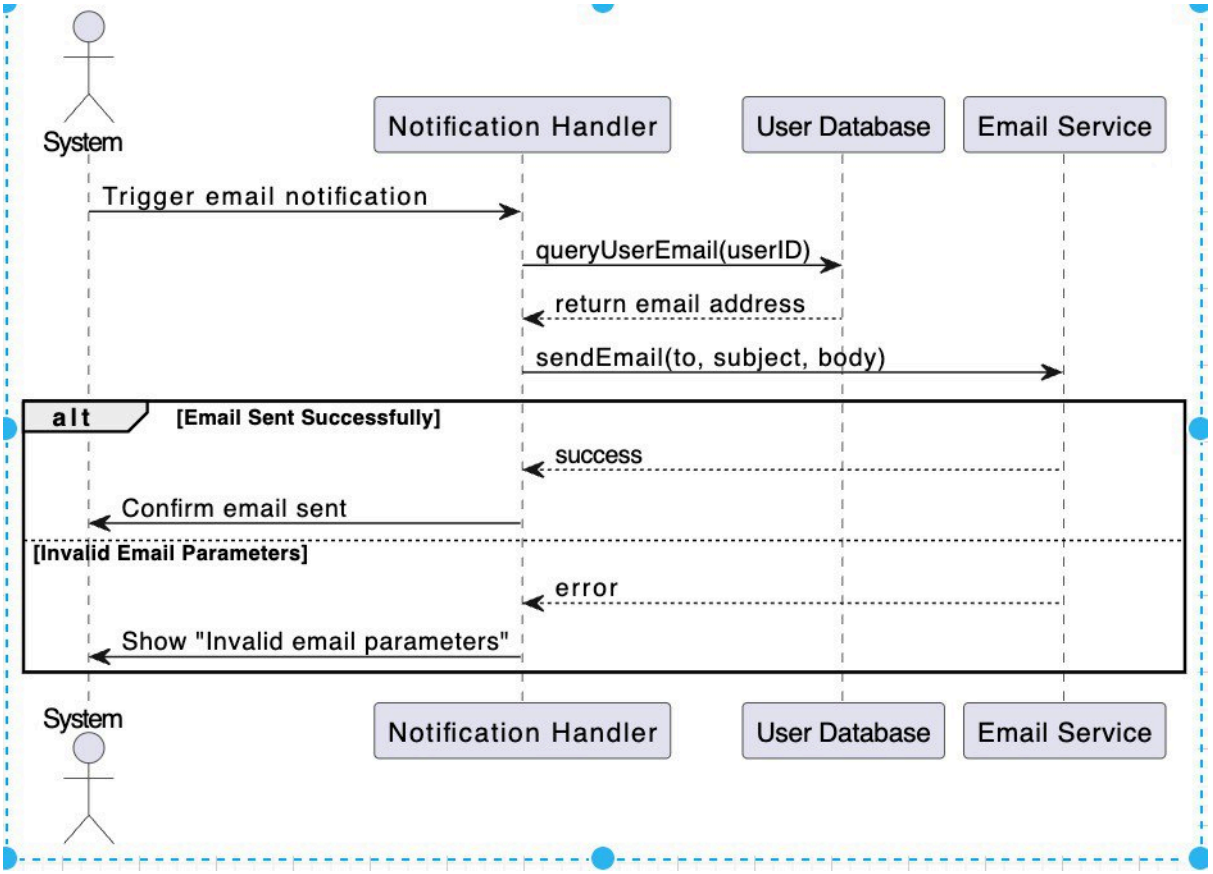
(Figure 33:Sequence Diagram for Generating Travel Report)

S6: Generate Recommended Train Scheduling(LTA/SMRT)



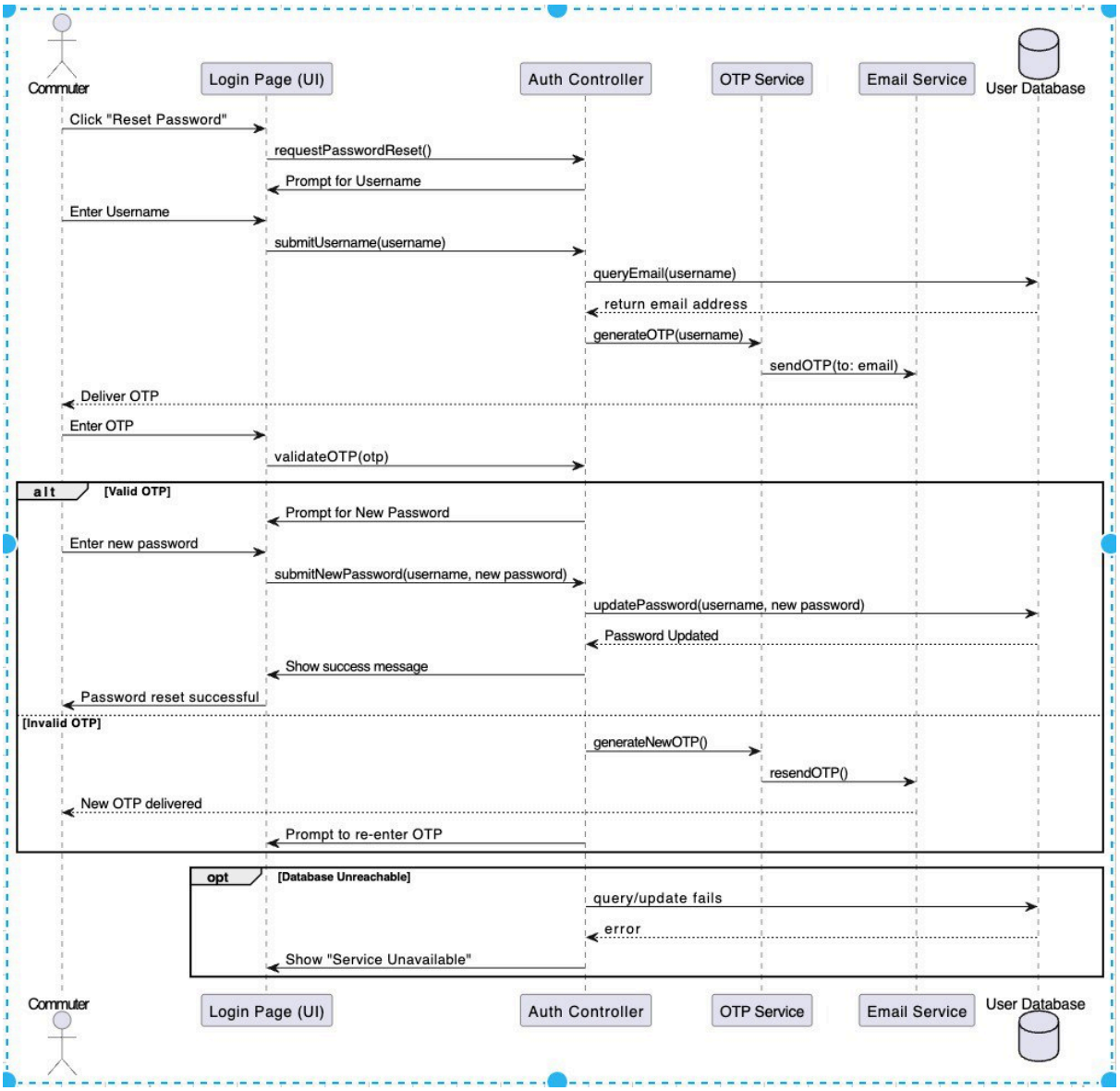
(Figure 34:Sequence Diagram for Generate Recommended Train Scheduling)

S7: Send Email Notification



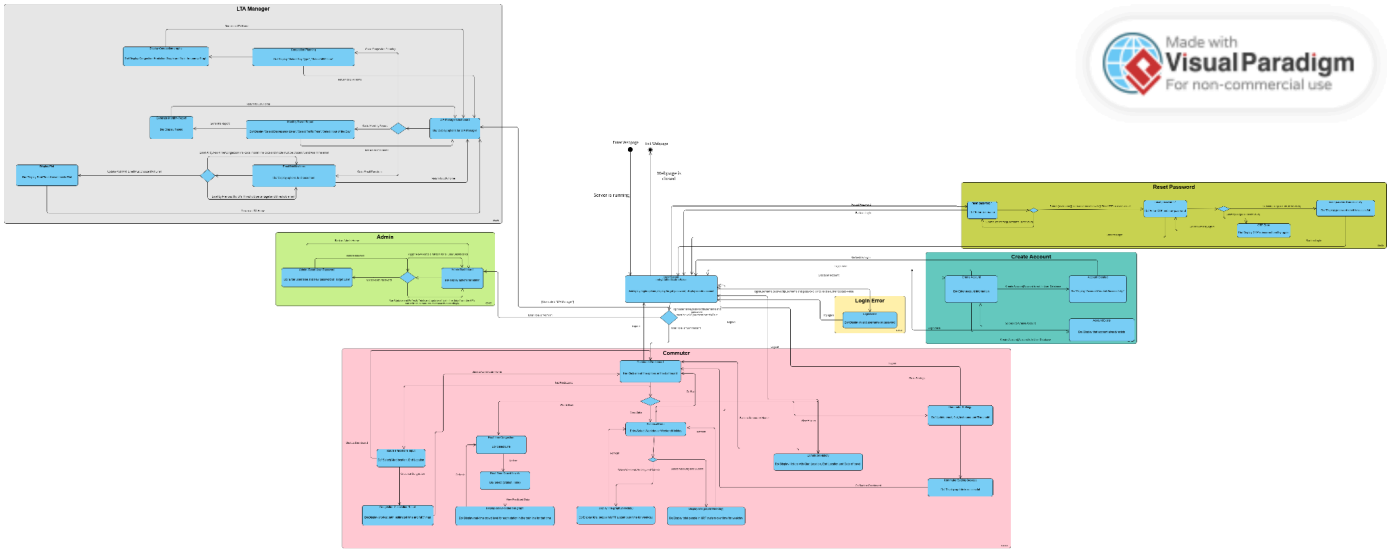
(Figure 35:Sequence Diagram for Sending Email Notification)

S8: Reset User Password



(Figure 36 :Sequence Diagram for Reset User Password)

4.5 Dialog Map (State Machine Diagram)

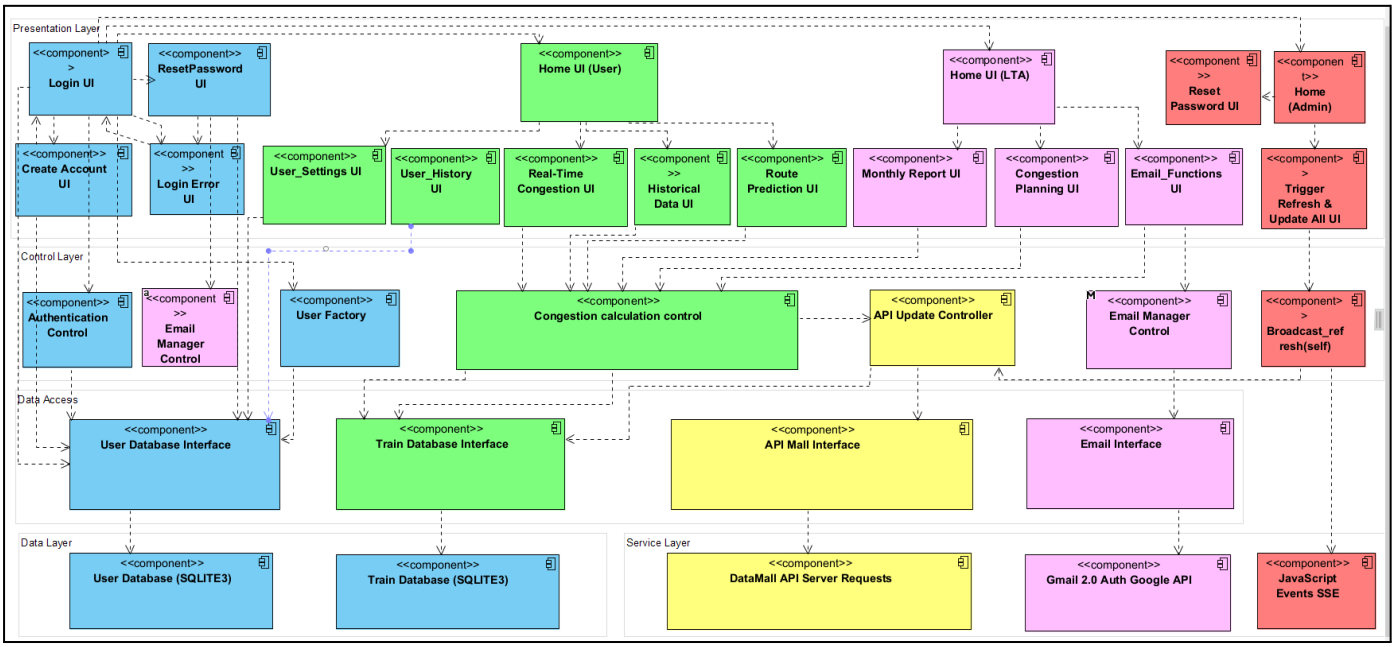


(Figure 37 :Dialog Map)

4.6 System Architecture

Using N-Type Architecture breaking down to

- Presentation Layer
- Control Layer
- Data Access Layer
- Data Layer
- Service Layer



(Figure 38 : System Architecture)

5. Testing

5.1 Black Box Testing

5.1.1 Selected Control Class: AuthController

The selected control class for black box testing is the AuthController, which handles user authentication including sign up and login processes.

- During sign up, the user must input:
    - email
    - password
    - name
- On successful sign up:
    - Password is hashed
    - User details are stored in the database
- During login, the user provides:
    - Registered email
    - Corresponding password
- The system authenticates the user and redirects to the home screen if the credentials match.

Test Case 1:Login Function (Apollo MRT)

Test scenario	Login to the system
Test case ID	Login
Test description	Invalid inputs result in unsuccessful login.
Prerequisite	Username must already be registered

Login Test Cases

#	Description	Action	Inputs	Expected Output	Actual Output	Test Result
1	Valid credentials	1. Go to Login page 2. Enter correct username and password 3. Press "Login"	Username: aaditya001 Password: abc12345	Redirect to user home page	User is redirected to home page	Pass
2	Incorrect password	Same as above, but wrong password	Username: aaditya001 Password: wrongpass	Show error message: "Login Failed — Invalid username or password. Try again"	Error message displayed	Pass

3	Non-existent username	Try with a username that doesn't exist	Username: notarealuser Password: abc12345	Show error message: "Login Failed — Invalid username or password. Try again"	Error message displayed	Pass
4	Empty username field	Submit form without entering a username	Username: (empty) Password: abc12345	Show message: "Please fill out this field"	Message shown	Pass

Test Case 2: Signup Function (Apollo MRT)

Test scenario	User creates a new account
Test case ID	signup
Test description	Invalid or incomplete inputs should prevent account creation.
Prerequisite	User must not already exist in the system

Signup Test Cases

#	Description	Action	Inputs	Expected Output	Actual Output	Test Result
1	Valid signup details	1. Go to Signup page 2. Fill all required fields with unique values 3. Click 'Create Account'	Username: newuser001 Email: newuser@email.com Password: strongPass123 First Name: John Last Name: Doe Type: Commuter	Account Created Successfully and redirected to success page	Account Created Successfully and redirected to success page	Pass

2	Empty username field	1. Go to Signup page 2. Leave username empty 3. Fill the rest and click 'Create Account'	Username: (empty) Other fields filled	Show message: "Please fill out this field"	Message is shown	Pass
3	Duplicate username	1. Use an existing registered username 2. Fill all fields and submit	Username: aaditya001 (already taken)	Account Already Exists The username or email you entered is already registered. Please try a different one.	User is redirected to 'account already exists' page	Pass

Test Case 3: Real-Time Congestion

Test scenario	View real-time congestion levels on selected MRT line
Test case ID	realtime_congestion
Test case description	User selects a line to view real-time congestion information.
Prerequisite	User must be logged in as Commuter

Real-Time Congestion - Black Box Test Cases

#	Description	Action	Inputs	Expected Output	Actual Output	Test Result
1	Valid line selected	1. Login as commuter 2. Go to Real-Time Congestion page 3. Select 'EWL' and click 'Submit'	Line: EWL	Bar chart with crowd levels for EWL stations appears	Chart displayed with station names and crowd levels	Pass
2	Valid line selected	1. Login as commuter 2. Go to Real-Time Congestion page 3. Select 'NSL' and click 'Submit'	Line: NSL	Bar chart with crowd levels for NSL stations appears	Chart displayed with station names and crowd levels	Pass

Test Case 4:Route Prediction

Test scenario	Find the shortest route between two MRT stations
Test case ID	route_prediction
Test case description	User selects two valid stations and views the predicted route and estimated travel time.
Prerequisite	User must be logged in as Commuter

Route Prediction - Black Box Test Cases

#	Description	Action	Inputs	Expected Output	Actual Output	Test Result
1	Valid start and end stations	1. Login as commuter 2. Navigate to route prediction 3. Select valid start and end stations	Start: EW21 End: EW16	Shortest route is displayed with estimated time	Route and time displayed	Pass
2	Same station selected for start and end	Select same station for both fields and submit	Start: EW 13 End: EW13	Message or logic to handle redundant route	Route with 0 min shown or error avoided	Pass
3	One or both stations not selected	Submit the form with empty fields	Start: (empty) End: EW16	Show form error or prevent submission	Form submission blocked	Pass

Test Case 5: Commuter Home Page - Black Box Testing (Apollo MRT)

Test scenario	Access and interact with commuter home interface
Test case ID	commuter_home_ui
Test case description	Ensure commuter is correctly routed to the homepage and UI components are accessible.
Prerequisite	User must be logged in as a Commuter

Commuter Home Page - Black Box Test Cases

#	Description	Action	Inputs	Expected Output	Actual Output	Test Result
1	Correct UI displayed after login	1. Login with valid commuter account 2. Observe redirection	Username: aaditya001 Password: abc12345	Redirect to /commuter/home with welcome message and options shown	Commuter homepage loaded with links (settings, history, etc.)	Pass
2	User tries to access without logging in	1. Open /commuter/ home directly in a private window	No session data	Redirect to login page	User is redirected to login page	Pass
3	Verify all links are present	Login and observe home screen for link options	Valid commuter session	Links/buttons for Route Prediction, Real-Time, History, Settings are visible	All links visible and clickable	Pass

Test Case 6: LTA Home Page - Black Box Testing (Apollo MRT)

Test scenario	Access and interact with LTA Manager home interface
Test case ID	lta_home_ui
Test case description	Ensure LTA Manager is routed to the home page and has access to key management features.
Prerequisite	User must be logged in as LTA_Manager

LTA Home Page - Black Box Test Cases

#	Description	Action	Inputs	Expected Output	Actual Output	Test Result
1	Correct UI after LTA login	1. Login with valid LTA account 2. Observe redirection	Username: lta_admin Password: lta12345	Redirect to /lta/home and display dashboard	Dashboard shown with summary and navigation options	Pass
2	User tries to access without logging in	Access /lta/home directly in incognito/ private window	No session	Redirect to login error page	Redirected to /login/error	Pass
3	Non-LTA user attempts access	Login as commuter/ admin and access /lta/home	Session: commuter/admin	Redirect to login error page	Redirected to /login/error	Pass
4	UI components check	Login and check for LTA tools: Monthly Report,	Valid LTA session	All tools visible and accessible via buttons or links	Monthly Report, Congestion Planning,	Pass

		Planning, Email			Email tabs displayed	

Test Case 7: Reset Password - Black Box Testing (Apollo MRT)

Test scenario	Reset a user account password
Test case ID	reset_password
Test case description	User enters valid or invalid credentials to reset password via OTP.
Prerequisite	User must have an existing registered account

Reset Password - Black Box Test Cases

#	Description	Action	Inputs	Expected Output	Actual Output	Test Result
1	Valid username and OTP entered	1. Go to /reset 2. Enter valid username 3. Receive and enter correct OTP 4. Enter new password and submit	Username: aaditya001 OTP: 123456 New Password: newpass123	Password is reset and redirected to success page	Redirected to /reset/ password/ success	Pass
2	Unregistered username entered	Enter a username that is not	Username: ghostuser	Display message: 'Username does not exist'	Error message displayed on screen	Pass

		registered and submit				
3	Incorrect OTP entered	Use valid username, but enter wrong OTP	Username: aaditya001 OTP: 999999	Redirect to OTP error page	Redirected to /reset/ password/ OTP_error	Pass
4	Empty username field	Submit reset form with username left blank	Username: (empty)	Show validation message: 'Please fill out this field'	Browser validation prevents submission	Pass
5	Empty OTP field on second page	Leave OTP input empty and try to submit	OTP: (empty)	Show validation message: 'Please fill out this field'	Browser validation prevents submission	Pass

5.1.3 Test Cases and Testing Results

5.1.3.1 Login

Input Parameters:

1. Email
2. Password

Test Case Name	Test Input	Expected Output	Actual Output	Test Result

Login-01	**(Valid)** Email: commuter@test.com Password: correctpassword123	Login Success	Login Success	Pass
Login-02	**(Valid)** Email: commuter@test.com **(Invalid)** Password: wrongpass	Login Failed: "Incorrect password"	Login Failed: "Incorrect password"	Pass
Login-03	**(Valid)** Email: commuter@test.com **(Invalid)** Password: ""	Login Failed: "Password is required"	Login Failed: "Password is required"	Pass
Login-04	**(Invalid)** Email: nonuser@test.com Password: somepassword	Login Failed: "Email not registered"	Login Failed: "Email not registered"	Pass
Login-05	**(Invalid)** Email: invalidemail.com Password: testpass123	Login Failed: "Invalid email format"	Login Failed: "Invalid email format"	Pass
Login-06	**(Invalid)** Email:	Login Failed: "Email and	Login Failed: "Email and	Pass

	" "Password: " "	password required"	password required"	

5.1.3.2 Sign Up

Input Parameters:

1. Email
2. Password
3. Full Name

Equivalence Classes – Sign Up

Class Type	Input Condition	Expected Output
Valid	Proper email format, non-empty password, valid name	Account created successfully
Invalid	Email field empty	Error: "Email is required"
Invalid	Password field empty	Error: "Password is required"
Invalid	Name field empty	Error: "Name is required"
Invalid	Email format invalid	Error: "Invalid email format"
Invalid	Duplicate email already exists	Error: "Email already registered"

Test Cases and Testing Results – Sign Up

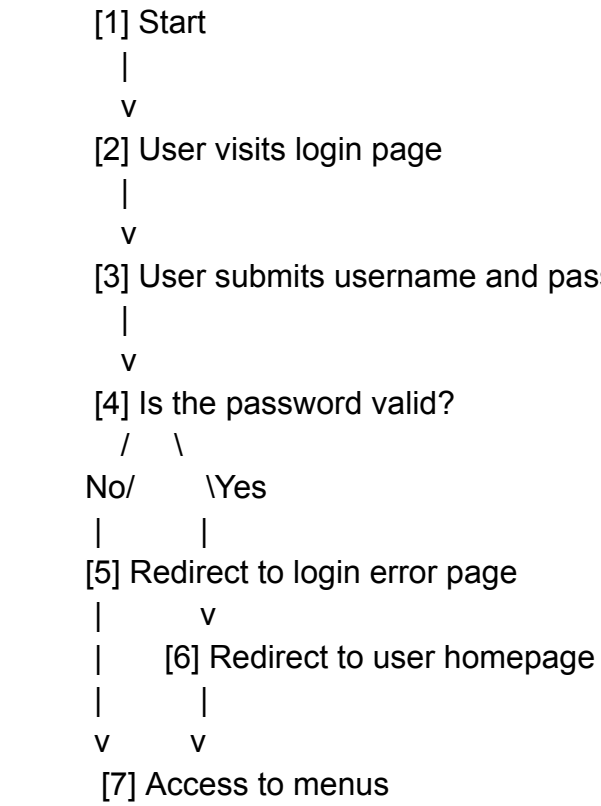
Test Case Name	Test Input	Expected Output	Actual Output	Test Resu
Signup-01	**(Valid)** Email: newuser@test.com Password: strongPass123 Name: Alex Tan	Sign Up Success	Sign Up Success	Pass
Signup-02	**(Invalid)** Email: "" Password: strongPass123 Name: Alex Tan	Error: "Email is required"	Error: "Email is required"	Pass
Signup-03	**(Invalid)** Email: newuser@test.com Password: "" Name: Alex Tan	Error: "Password is required"	Error: "Password is required"	Pass
Signup-04	**(Invalid)** Email: newuser@test.com Password: strongPass123 Name: ""	Error: "Name is required"	Error: "Name is required"	Pass

Signup-05	**(Invalid)** Email: `invalidemail.com` Password: `strongPass123` Name: `Alex Tan`	Error: "Invalid email format"	Error: "Invalid email format"	Pass
Signup-06	**(Invalid)** Email: `existinguser@test.com` Password: `strongPass123` Name: `Alex Tan`	Error: "Email already registered"	Error: "Email already registered"	Pass

5.2 White Box Testing

Test Case 1: Login Function

Control Diagram:



Paths:

1. Path 1: Username and password are valid
    a. 1 → 2 → 3 → 4 → 6 → 7

2. Path 2: Username or password is invalid
    a. 1 → 2 → 3 → 4 → 5 → 7

Test Cases:

1. Test Case 1 (Path 1): Correct credentials (valid username and password)
    a. Input: User submits valid username and password.

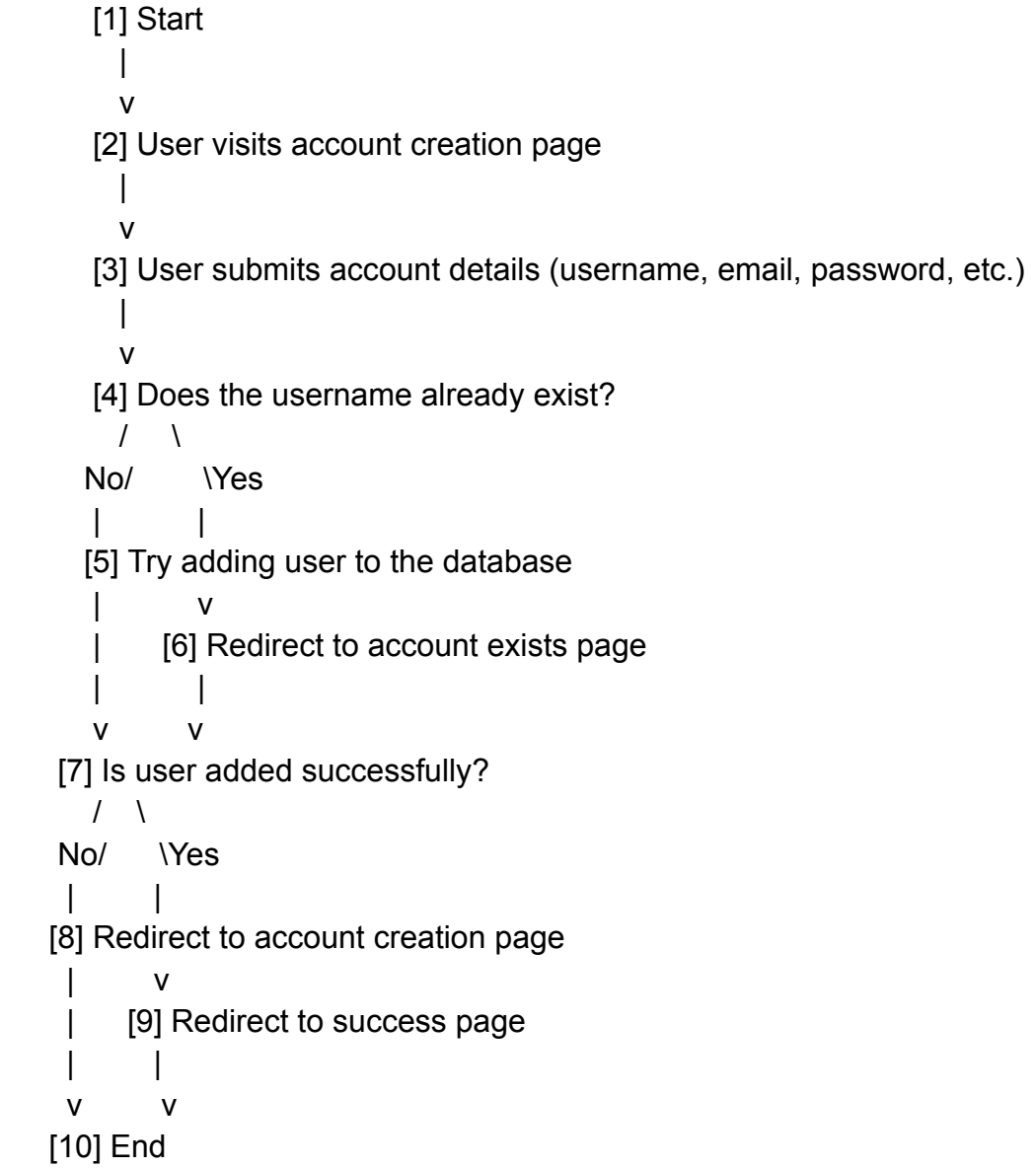
    b. Expected Output: User is redirected to the homepage (based on user type).
2. Test Case 2 (Path 2): Incorrect credentials (invalid username or password)
    a. Input: User submits invalid username or password.
    b. Expected Output: User is redirected to the login error page.

Cyclomatic Complexity Calculation:

- V(G) = 7−6+2(1) = 3

Test Case 2: Account Creation

Control Diagram:



Paths:

1. Path 1: Username exists, redirect to error page
    a. 1 → 2 → 3 → 4 → 5 → 6 → 10
2. Path 2: Username doesn't exist, user added successfully
    a. 1 → 2 → 3 → 4 → 5 → 7 → 9 → 10
3. Path 3: Username doesn't exist, user addition failed
    a. 1 → 2 → 3 → 4 → 5 → 7 → 8 → 10

Test Cases:

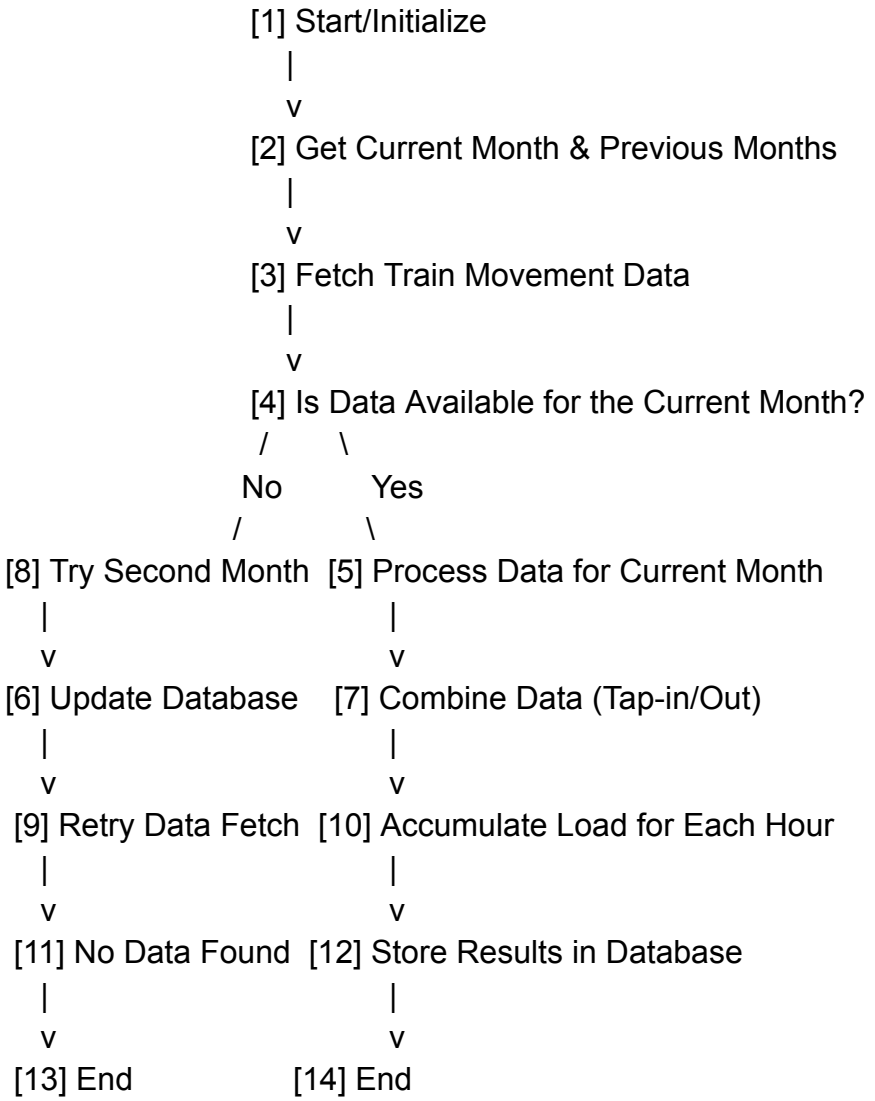
1. Test Case 1 (Path 1): Username already exists
    a. Input: User submits a username that already exists.
    b. Expected Output: User is redirected to the account exists error page.
2. Test Case 2 (Path 2): Account created successfully
    a. Input: User submits valid, unique account details.
    b. Expected Output: User is redirected to the account created success page.
3. Test Case 3 (Path 3): Account creation fails (e.g., due to database issues)
    a. Input: User submits valid, unique account details, but account creation fails.
    b. Expected Output: User is redirected back to the account creation page to retry.

Cyclomatic Complexity Calculation:

- $V(G) = 9-7+2(1) = 4$

Test Case 3: Calculate Congestion

Control Diagram: (Load Diagram)



Paths:

1. Path 1: Current month data is available, processed, and stored.
    a. 1 → 2 → 3 → 4 → Yes → 5 → 7 → 10 → 12 → 14
2. Path 2: No data for current month, but data is available for the second month, processed, and stored.
    a. 1 → 2 → 3 → 4 → No → 8 → 5 → 7 → 10 → 12 → 14
3. Path 3: No data for both current and second months, database is updated, retry data fetch.
    a. 1 → 2 → 3 → 4 → No → 8 → 6 → 9 → 3 → 4 → No → 11 → 13
4. Path 4: No data for both current and second months after retry, no data found.

a. 1 → 2 → 3 → 4 → No → 8 → 6 → 9 → 3 → 4 → No → 11 → 13

Test Cases:

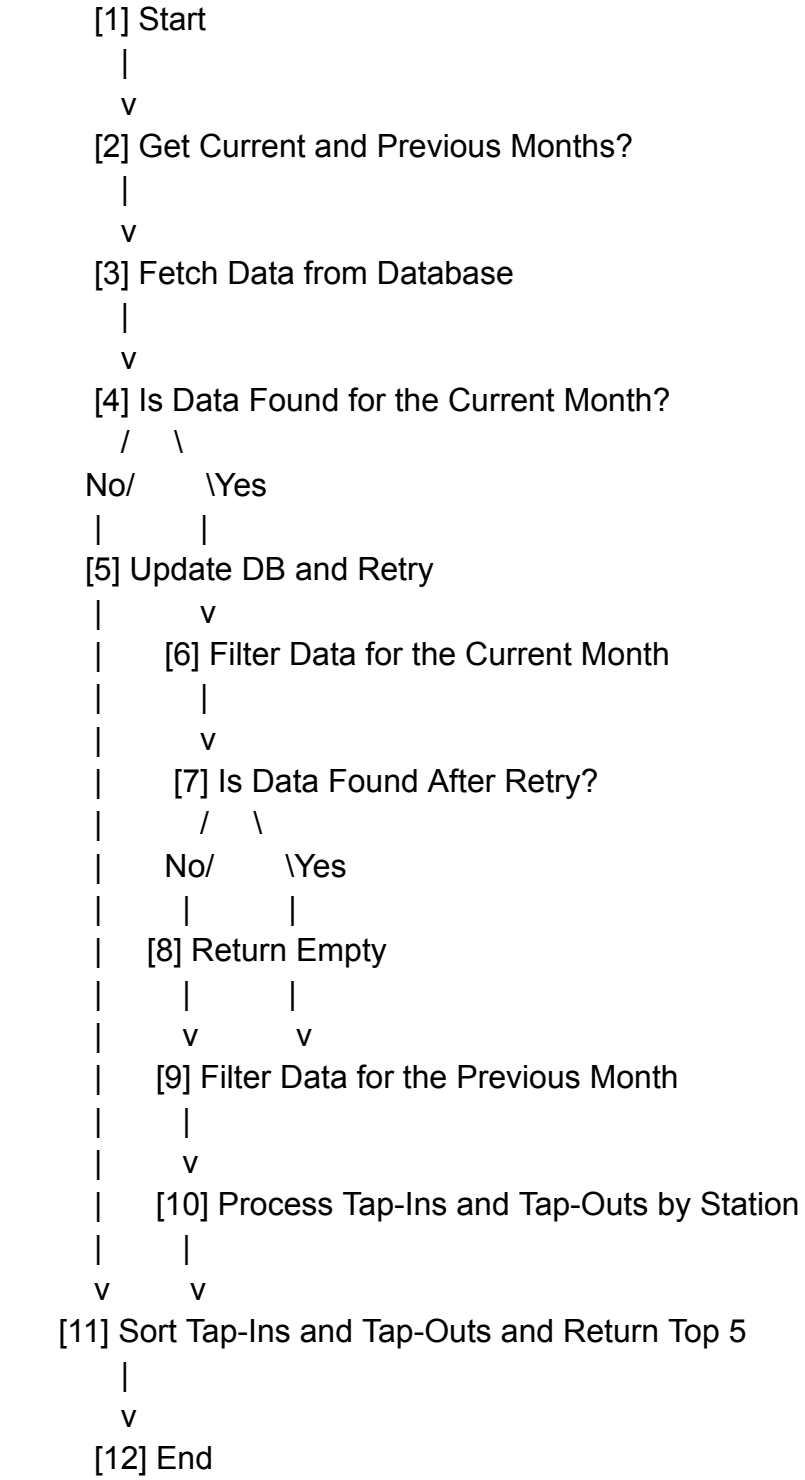
1. Test Case 1 (Path 1): Current month data is available
    a. Input: Data exists for the current month.
    b. Expected Output: Data is processed, combined, and stored in the database.
2. Test Case 2 (Path 2): No data for current month, but data exists for second month
    a. Input: No data for the current month, but data exists for the second last month.
    b. Expected Output: Data for the second last month is processed, combined, and stored in the database.
3. Test Case 3 (Path 3): No data for both months, database update is triggered
    a. Input: No data for the current and second last month, triggering a database update.
    b. Expected Output: Database is updated and retried fetching data. No data is found after retry.
4. Test Case 4 (Path 4): No data after retry, failure
    a. Input: No data found after retrying the data fetch.
    b. Expected Output: The system logs the error, and no data is processed.

Cyclomatic Complexity Calculation:

- V(G = 15−14+2=3

Test Case 4: Obtaining Busiest Station

Control Diagram:



Paths:

1. Path 1: Current month data is available
    a. 1 → 2 → 3 → 4 → 6 → 10 → 11 → 12
2. Path 2: No data for current month, but data exists for second month
    a. 1 → 2 → 3 → 4 → 5 → 3 → 4 → 6 → 10 → 11 → 12
3. Path 3: No data for both months, database update is triggered
    a. 1 → 2 → 3 → 4 → 7 → 10 → 11 → 12
4. Path 4: No data after retry, failure
    a. 1 → 2 → 3 → 4 → 7 → 9 → 10 → 11 → 12

Test Cases:

1. Test Case 1 (Path 1): Current month data is available
    a. Input: Data exists for the current month.
    b. Expected Output: Data is processed, combined, and stored in the database.
2. Test Case 2 (Path 2): No data for current month, but data exists for second month
    a. Input: No data for the current month, but data exists for the second last month.
    b. Expected Output: Data for the second last month is processed, combined, and stored in the database.
3. Test Case 3 (Path 3): No data for both months, database update is triggered
    a. Input: No data for the current and second last month, triggering a database update.
    b. Expected Output: Database is updated and retried fetching data. No data is found after retry.
4. Test Case 4 (Path 4): No data after retry, failure
    a. Input: No data found after retrying the data fetch.
    b. Expected Output: The system logs the error, and no data is processed.

Cyclomatic Complexity Calculation:

- $V(G) = 13-12+2(1) = 3$

5.3 Assumptions & Parameters

Assumptions:

- For historical congestion calculations, we delay the cumulative_out by one cycle (hour) to account for passengers who tap in and tap out within the same hour. This adjustment ensures that such journeys are not excluded from the effective load computation of the hour.

- We assume that the proportion of commuters tapping into a specific MRT line reflects the proportional demand load on that line during any given hour.

- The average MRT commute distance in Singapore is estimated at 11.7 km, which equates to approximately 7 stations per trip (Statista, Wikipedia).

Parameters:

Parameter	Value / Description
Total MRT Track Length	242.6 km across 143 stations
Average Commute Distance	11.7 km ≈ 7 stations per trip
Average Peak Hour Demand (EW Line)	95,541 people/hour (estimated for entire East-West line)
Train Capacity	1,920 passengers per train (based on MRT rolling stock max load)
Peak Frequency	Every 2.5 minutes → 24 trains/hour/direction → Max Capacity: 92,160 people/hr
Off-Peak Frequency	Every 6 minutes → 10 trains/hour/direction → Max Capacity: 38,400 people/hr

Train Requirement Formula:

We estimate the number of trains needed per hour using:

Number of Trains Needed= Train Capacity / Peak Hour Demand

Example Calculation:

- Peak Hour Demand (EW Line): 95,541 people/hour
- Train Capacity: 1,920 people/train

Trains Needed= 95541 / 1920 ≈ 49.7

Approximately 50 trains per hour are required (or 25 trains per direction).

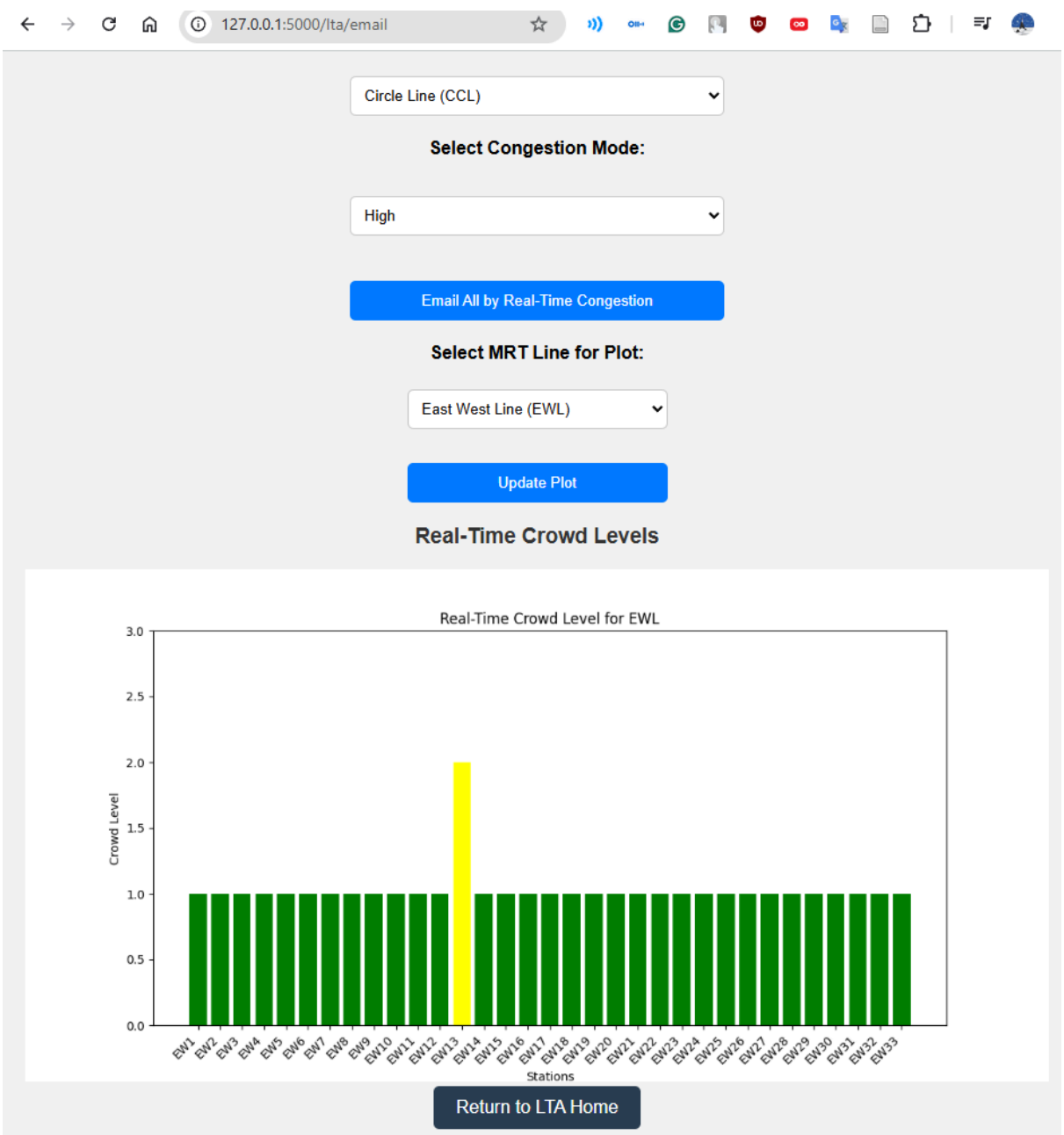
Capacity vs. Demand Comparison:

Scenario	Trains/hr (per direction)	Total Capacity/hr	Meets 95k Demand?
Peak (2.5 min)	24 x 2 = 48 trains/hr	48 x 1,920 = 92,160	Slightly under by ~3,400
Actual Required	25 x 2 = 50 trains/hr	96,000	Just enough
Off-Peak (6 min)	10 x 2 = 20 trains/hr	20 x 1,920 = 38,400	Far below requirement

- https://www.statista.com/statistics/1232854/singapore-average-number-of-stops-on-commute-trip/
- https://en.wikipedia.org/wiki/Mass_Rapid_Transit_%28Singapore%29
- https://www.google.com.sg/maps/dir/Pasir+Ris+Central,+Pasir+Ris+MRT+Station+(EW1),+Singapore/Tuas+West+Drive,+Tuas+Link+MRT+Station+(EW33),+Singapore/@1.3249691,103.7134989,12z/data=!3m2!4b1!5s0x31da0f55acf49f93:0x27edb502ede309a5!4m14!4m13!1m5!1m1!1s0x31da3d4802ba5673:0x9b4b232b5dcb0936!2m2!1d103.9493411!2d1.3731143!1m5!1m1!1s0x31da

0f71431c9faf:0x4cc2439dc086eeae!2m2!1d103.6366883!2d1.3397689!3e3?entry=ttu&g_ep=EgoyMDI1MDQwMi4xIKXMDSoASAFQAw%3D%3D

- https://en.wikipedia.org/wiki/List_of_Singapore_MRT_and_LRT_rolling_stock



(Figure 39: Real-Time MRT Congestion Dashboard)

Real-Time Congestion Alert for EWL Line (Data based on past 10 minutes to live) Inbox ×

apollo11mrt@gmail.com
to me

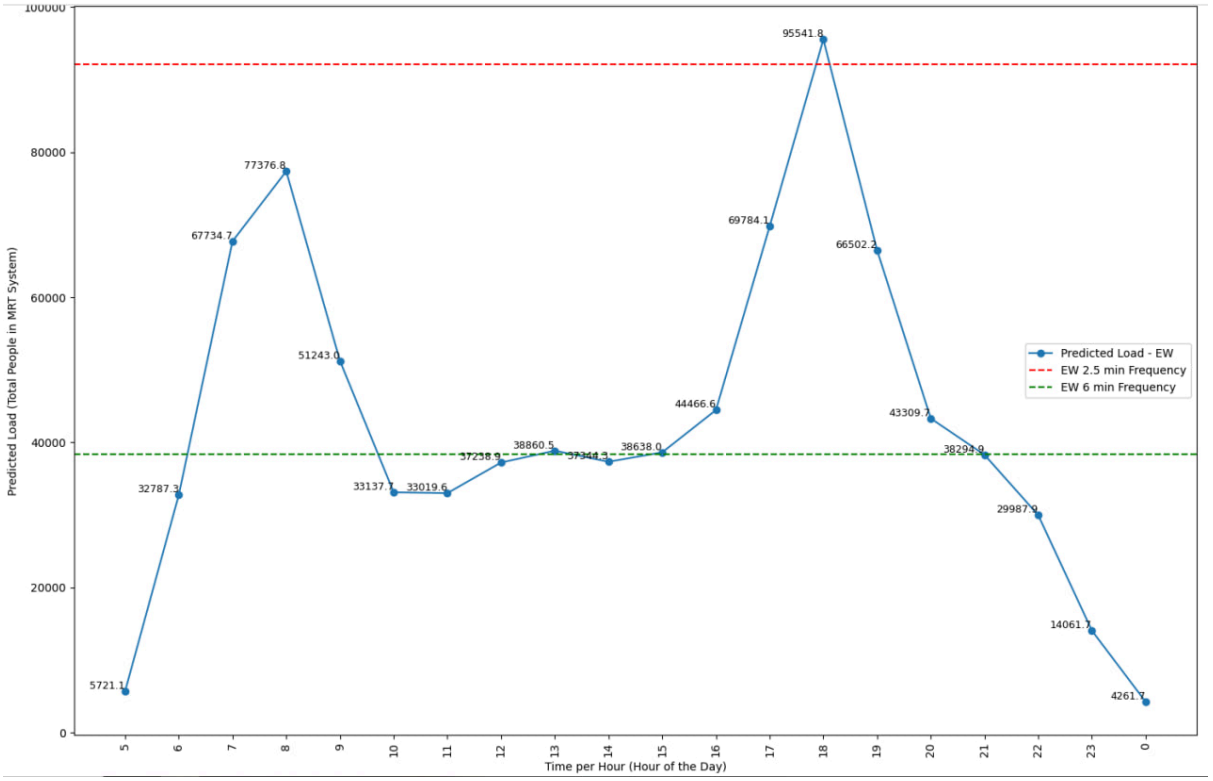
Dear John,

The following stations on the EWL line are currently experiencing congestion levels of: M.

Station: EW13, Crowd Level: M, From: 2025-04-11T21:50:00+08:00, To: 2025-04-11T22:00:00+08:00

Reply Forward

(Figure 40: Real Time Congestion Alert on Email)



(Figure 41: Predicted Load)

6. Requirements

6.1 Functional Requirements

1. User Registration and Authentication: allowing users to create new accounts using their email and password.
2. Users should be able to reset their passwords via email OTP.
3. Real Time Congestion Monitoring: users will be able to view current congestion levels at any MRT station.
4. Users should be able to view their past search history.
5. Users can select specific lines for sorting of data for the Congestion Graphs
6. Historical Congestion Trends: Users shall be able to select an MRT station and view historical congestion data.
7. Congestion Alert: Users shall be able to set congestion thresholds for selected stations.
8. Alternate Route Suggestions: The Route suggestion should be adaptable to new MRT stations as well.
9. Displayed information should be refreshed when the data is updated.
10. The system should be capable of sending users alerts when real-time congestion is high via email.

6.2 Non-Functional Requirements

Responsive User Interface

- Users should experience minimal delays (<5 seconds) when performing common tasks such as login, viewing live congestion, or setting alerts.

- UI Should indicate an error if unable to load

Mobile Responsive

- The application shall be fully functional on all major mobile devices, including both Android and iOS phones.

- Users must be able to view **1**00% of the content regardless of device screen size.

- UI elements like buttons and inputs must be accessible and touch-friendly.

Data Retrieval Efficiency

- The application shall retrieve and display congestion data or analytics (e.g., nearby stations, peak times) within 30 seconds under standard load and network conditions.

- System should query local database and API only if the data of the date is not found to prevent overloading the api data rate

- The System should be able to support 100 users at once.

Quick App Startup

- After launching or rebooting the application, it must load fully and be usable within 5 seconds.

- Startup time includes loading the login screen and all static assets required for navigation.

Security

- The system should not let users of the wrong type access web pages that are not for that data type

- The System should store passwords in hashed forms and compare them in hashed forms to ensure password security

Appendix

Appendix A: User Manual

DATA DICTIONARY

Term	Definition
Account	A registered commuter profile in the application. Includes personal information such as email, password, preferences, and alert configurations.
LTA Manager	A user with elevated privileges who can monitor alert logs, generate reports, and oversee system performance and congestion data.
Admin	Someone who can modify user's credentials, update all databases and refresh users
Web Application	A mobile or web-based tool that provides real-time train congestion data, alternate route suggestions, and congestion alerts to users.
Alert	A notification trigger set by a user that activates when a selected MRT station exceeds a defined congestion threshold.

Load	Amount of people in the entire MRT ecosystem for a particular hour measured from the (current_hour_cumulative_tap_in - previous_hour_Cumulative_tap_out).
Dashboard	A screen displaying real-time and historical congestion levels, alert settings, and suggested routes personalized to the user.
Data Source	External APIs integrated with the app, such as LTA's TrainCrowdData API, used to fetch real-time congestion data across MRT stations.
Historical Trends	Past congestion data stored and visualized to help users understand average load at different times and stations.
Journey	A trip taken by the user from one MRT station to another. The app analyzes journey segments to suggest lower congestion routes.
Notification	An automated email sent to the user when real-time congestion exceeds the user's predefined threshold at a monitored station.
Real-Time Data	Live congestion data fetched periodically from LTA's APIs and used to update the UI and trigger user alerts.
Route	A recommended path from a selected origin to a destination MRT station, ranked by congestion levels and travel efficiency.
Station	A physical MRT stop. Each station has a unique code and associated congestion values that update every minute from the LTA DataMall.

Threshold	The crowd percentage value (e.g. 70%) set by the user which, when exceeded, triggers an alert for the selected MRT station.
User/Commuter	A general user of the application, typically a daily MRT commuter who uses the app to monitor congestion and optimize their travel.
Visualization	Charts and are used in the app to help users understand congestion patterns across stations and times.