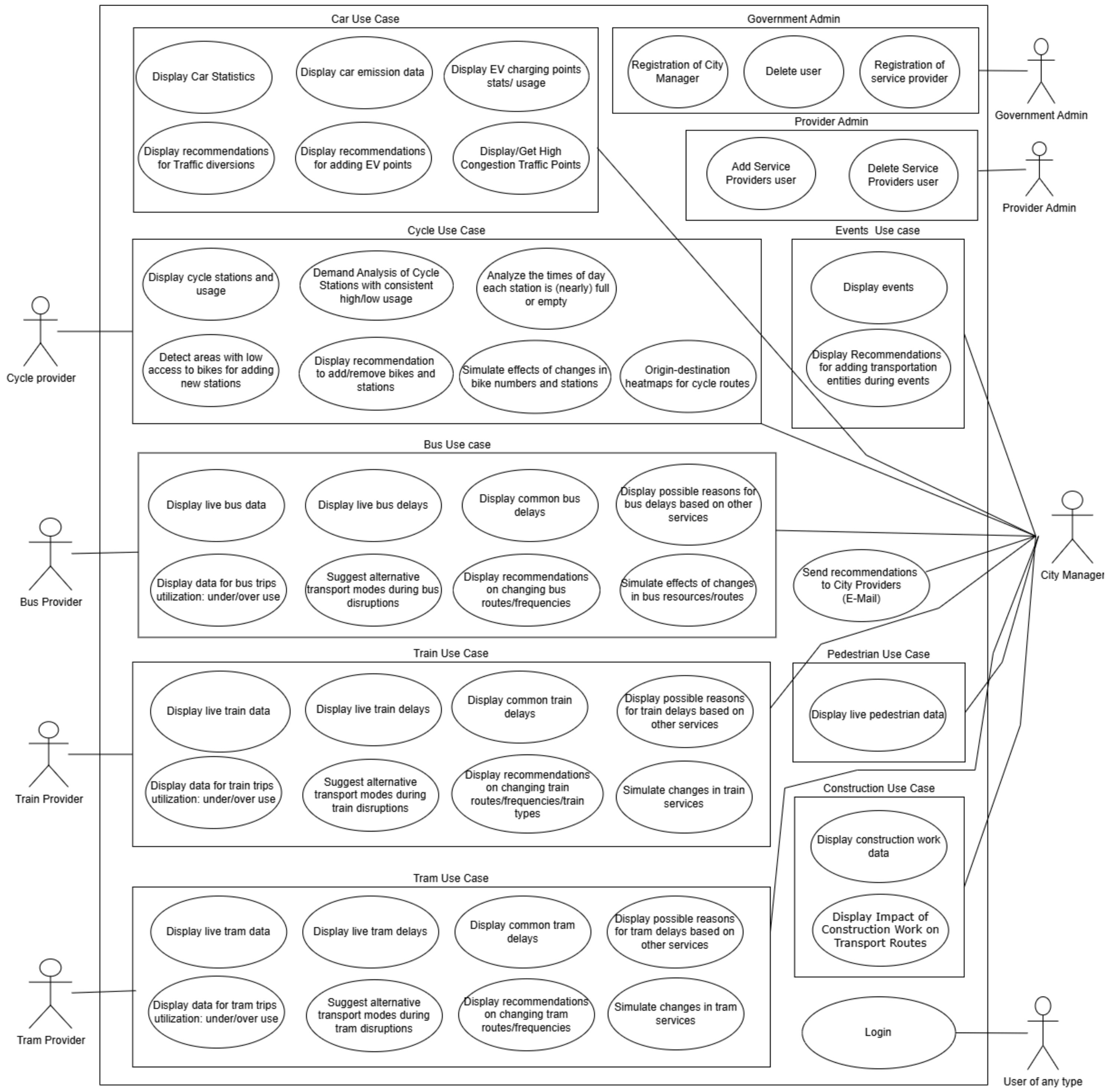
CS7CS3 Advanced Software Engineering Group Project

Requirements/Use Cases

Project Name: Sustainable City Management

Group: ***10***
IOANNIS CHRISTOFILOGIANNIS
AKHIL ELANGO
MUKUL GHARE
MOHAMMAD OWAIS KHAN
NADINE KIRNBAUER
NAMAN KUMAR
RAKESH LAKSHMANAN
SUDESH MURALIDHARAN
STEVIN JOSEPH SEBASTIAN
MICHAL BRONICKI

1. Use Case Diagram


Car Use Case
Display Car Statistics
Display car emission data
Display EV charging points stats/ usage
Display recommendations for Traffic diversions
Display recommendations for adding EV points
Display/Get High Congestion Traffic Points
Government Admin
Registration of City Manager
Delete user
Registration of service provider
Government Admin
Provider Admin
Add Service Providers user
Delete Service Providers user
Provider Admin
Cycle Use Case
Display cycle stations and usage
Demand Analysis of Cycle Stations with consistent high/low usage
Analyze the times of day each station is (nearly) full or empty
Detect areas with low access to bikes for adding new stations
Display recommendation to add/remove bikes and stations
Simulate effects of changes in bike numbers and stations
Origin-destination heatmaps for cycle routes
Cycle provider
Events Use case
Display events
Display Recommendations for adding transportation entities during events
Bus Use case
Display live bus data
Display live bus delays
Display common bus delays
Display possible reasons for bus delays based on other services
Display data for bus trips utilization: under/over use
Suggest alternative transport modes during bus disruptions
Display recommendations on changing bus routes/frequencies
Simulate effects of changes in bus resources/routes
Bus Provider
Send recommendations to City Providers (E-Mail)
City Manager
Train Use Case
Display live train data
Display live train delays
Display common train delays
Display possible reasons for train delays based on other services
Display data for train trips utilization: under/over use
Suggest alternative transport modes during train disruptions
Display recommendations on changing train routes/frequencies/train types
Simulate changes in train services
Train Provider
Pedestrian Use Case
Display live pedestrian data
Construction Use Case
Display construction work data
Display Impact of Construction Work on Transport Routes
Tram Use Case
Display live tram data
Display live tram delays
Display common tram delays
Display possible reasons for tram delays based on other services
Display data for tram trips utilization: under/over use
Suggest alternative transport modes during tram disruptions
Display recommendations on changing tram routes/frequencies
Simulate changes in tram services
Tram Provider
Login
User of any type

Government Admin Box:

Use Case: *Registration of City Manager*

1. Description
This use case describes the registration of a City Manager by the Government Admin. The goal is to enable internal users (City Managers) to access mobility data and insights for city management.
Actors:

- Government Admin: creates and manages city manager accounts.
- City Manager: receives access to the dashboard and data insights.

Triggers:

- Triggered manually by the Government Admin when a new city manager needs to be registered in the system.

Inputs:

- City manager details (name, email, role).

2. Flow of Events

Basic Flow			
Government Admin		System	
1	Government Admin logs into the system		
		2	Government Admin gets logged in
3	Navigates to user management and selects “Register City Manager”		
4	Enters manager’s details (name, email, role)		
		5	System validates data and creates account
		6	System displays confirmation message

Alternative Flow 1 - Incorrect Manager details			
User		System	
		6.1	Displays error if email already exists or validation fails

Alternative Flow 2 - Invalid login details			
User		System	
		2.1	The system shows an alert to the user saying that credentials are incorrect and denies the access to the system

3. Special Requirements
- Requires access to the user management system and authentication.

4. Preconditions
- Government Admin must be logged in and authorized.
- Government Admin must possess the detailed data of the user they want to provide access to.

5. Postconditions
- The City Manager account is created and can access the system.

Use Case: *Registration of Service Provider*

1. Description

This use case covers the registration of an external Service Provider Admin by the Government Admin. The goal is to allow external service providers to access relevant mobility data for their services.

Actors
- Government Admin: approves and registers service provider accounts.

Triggers
- Triggered when an external provider requests access.

Inputs
- Inputs include provider company information and contact details. (maybe number of users required)

2. Flow of Events

Basic Flow			
Government Admin		System	
1	Government Admin logs into the system		
		2	User gets logged in
3	Navigates to user management and selects “Register City Manager”		
4	Enters manager’s details (name, email, role)		
		5	System validates data and creates account
		6	System displays confirmation message

Alternative Flow 1			
Government Admin		System	
		6.1	Displays error if email already exists or validation fails

Alternative Flow 2			
Government Admin		System	
		2.1	The system shows an alert to the user saying that credentials are incorrect and denies the access to the system

3. Special Requirements
Dependencies: authentication service.
4. Preconditions
Government Admin must be logged in and authorized. Government Admin has to know which external service provider they want to provide access to *(Data collection, user data, verification)*
5. Postconditions
Service Provider Admin account is created and can log in. They can now add new users from their company.

Use Case: *Delete User*

1. Description
This use case describes how a Government Admin can delete a City Manager or Service Provider user from the system to revoke access.
Actor
- Government Admin: initiates and confirms user deletion.

Triggers
- Triggered by government admin when a user should no longer have access.

Inputs
- user ID or email

2. Flow of Events

Basic Flow			
Government Admin		System	
1	Admin navigates to user management page		
2	Selects a user and clicks “Delete”		
		3	System asks for confirmation
4	Admin confirms deletion		
		5	System removes user and displays confirmation

Alternative Flow 1			
Government Admin		System	
		5.1	Displays error if deletion fails

3. Special Requirements
Requires connection to user management database.
Dependencies: Authentication and authorization systems.
4. Preconditions
User exists and is listed in the system.
5. Postconditions
User account is permanently deleted and can't access the platform anymore.

Provider Admin Box:

Use Case: *Add Service Provider user*

1. Description
This use case allows a Service Provider Admin to register internal users within their organization for access to company-specific data.
Actors

- Service Provider Admin: creates new internal users.
- Internal Users: access the dashboard of the specific service provider they belong to

Triggers and Inputs
Triggered when the service provider wants to add internal users.
Inputs: user name, email, role within company.
2. Flow of Events

Basic Flow			
Service Provider Admin		System	
1	*Service Provider* Admin logs into the system		
		2	User gets logged in
3	Navigates to user management and selects "Register User"		
4	Enters manager's details (name, email, role)		
		5	System validates data and creates account
		6	System displays confirmation message

Alternative Flow 1			
User		System	
		6.1	Displays error if email already exists or validation fails

Alternative Flow 2			
User		System	
		2.1	The system shows an alert to the user saying that credentials are incorrect and denies the access to the system

3. Special Requirements
Requires access to the service provider module.
Dependencies: Centralized authentication system.
4. Preconditions
Service Provider Admin must be logged in and authorized.
5. Postconditions
Internal user account is created successfully and can access platform and perform actions.

Use Case: *Delete Service Provider user*

1. Description
This use case covers how a Service Provider Admin removes an internal user from their company's access list.
Actors
- Service Provider Admin: selects and deletes internal users.

Triggers and Inputs
Triggers: Triggered when an employee leaves the organization or no longer requires access.
Inputs: internal user ID or email.

2. Flow of Events

Basic Flow			
User		System	
1	Admin navigates to user management page		
2	Selects a user and clicks "Delete"		
		3	System asks for confirmation
4	Admin confirms deletion		
		5	System removes user and displays confirmation

Alternative Flow 1	
User	System

		5.1	Displays error if deletion fails due to dependencies

3. Special Requirements
Requires proper authentication and database access.
Dependencies: Company-specific users information data
4. Preconditions
Service Provider Admin must be logged in and authorized. User account must exist
5. Postconditions
The internal user account is deleted and access revoked. Users can't access the platform anymore.

Car Use Case Box:

Use case: Display Car Statistics

1. Description**:**
This use case enables the city manager to view the data on cars in the city. The information will include statistics like total cars, cars added in the last 6 months to 1 year, stats of fuel based cars and EVs, Trends for different car types. The system will identify the trends in EVs and fuel based cars for the city manager.
Actors

- City Manager**:** Evaluates the usage of cars.

Triggers:

- Analysis report shows a steep increase in the number of registered cars in the last 2-3 months.

Inputs:

- Car registration data including information about year of registration, car type (Petrol, Diesel or EV)

2. Flow of Events:

Basic Flow			
User : City Manager		System	
1	User logs into the system.		
2	User selects "Car Indicator"		
		3	System opens the dashboard for cars with different tabs.
4	User select "Car Statistics" Tab		
		5	System retrieves the data from the database, performs the analysis and renders the statistics in the form of charts, graphs and tables.

6	User reviews cars usages and the trends of car registrations and identifies potential issues.		

Alternative flow			
User : City Manager		System	
1	User logs into the system.		
2	User selects “Car Indicator”		
		3	System opens the dashboard for cars with different tabs.
4	User select “Car Statistics” Tab		
		5	System displays an error saying “ Car statistics data unavailable please check back after few mins”

3. Special Requirements**:** None
4. Pre- Conditions
   - User authenticated to view the dashboard.
   - Car Statistics data available in Database.
   - Dashboard already rendered with the visualizations i.e. all calculation for the visualization already done in the background and cached.
5. Post Conditions
   - Car usage for different fuel type and Trends displayed on the page.

Use case : *Display recommendations for Traffic diversions*

1. Description

The goal of this use case is to generate data-driven recommendations to optimise traffic flow in areas with regular congestion. Show recommendations to reroute vehicles via alternate routes, adding/removing lanes, or adjusting signal timings to minimize delays and improve mobility efficiency across the city.

Actors
- City Managers - Review and Authorize recommendations

Triggers
- Analysis report shows recurring congestion beyond defined thresholds on a route.
- Scheduled event is expected to increase/decrease traffic in an area

Inputs
- Historical traffic data from APIs
- Insights from data analysis report

2. Flow of Events

Basic Flow			
User : City Manager		System	
1	Monitor Dashboard for recommendations on Bikes data.		
		2	Whenever a trigger fires, generate and display a recommendation.
3	User reviews the system generated recommendation and approve it.		
		4	Change recommendation status to approved.

Alternative Flow 1			
User : City Manager		System	
3.1	Rejects system generated recommendation		
		3.2	Changes recommendation status to Rejected

3. Special Requirements
    - N/A

4. Preconditions
    - User authentication and Login
    - Data Analysis system should have provided latest insights from historical data
    - Threshold values for congestion severity must be defined
    - Events data should be updated and synchronized with the system

5. Postconditions
    - Recommendation status has been updated

Use Case : *Display Car Emission Data*

1. Description: This use case enables the user to view the data on emissions from cars. This information will include CO2 emissions from cars and trends related to emissions.

Actors:
- City Manager: Review the recommendation by the system.

Triggers
- Analysis report shows a steep increase in the number of fuel based cars in the last 2-3 months and the potential increase in the CO2 emissions.

Input:
- Car emission data i.e. CO2 emission and other gases emission.

2. Flow events:

Basic Flow			
User : City Manager		System	
1	User selects "Car Indicator"		
		2	System opens the dashboard for cars with different tabs.
3	User select "Emissions" Tab		
		4	System retrieves the data from the database, performs the analysis and renders the statistics in the form of charts, graphs and tables.
5	User reviews emissions data and the trends of emissions and identifies potential issues.		

Alternate Flow			
User : City Manager		System	
		4.1	System displays an error saying " Emission Data unavailable please check back after few mins"

Special Requirements: N/A
Pre-conditions
- Users must be authenticated and logged in.
- Emissions data available in Database.
- Dashboard already rendered with the visualizations i.e. all calculation for the visualization already done in the background and cached.

Post Conditions
- Emissions data and trends displayed on the page.

Use Case: *Display EV Charging Points, Stats/Usage*

1. Description**:**
This use case provides the city manager information on the locations of EV charging stations currently present in the city and their usage. This information combined with the data on EV car statistics, the city manager can assess the future demand for EV charging stations.
Actors:
- City Manager: Review the recommendation by the system.

Triggers
- Analysis reports show a steep increase in usage of charging stations and an increase in the number of EV cars.

Input
- EV charging point location and usage data.

2. Flow of Events

Basic Flow			
User : City Manager		System	
1	User selects “Car Indicator”		
		2	System opens the dashboard for cars with different tabs.
3	User select “EV Charging Usage” Tab		
		4	System retrieves the data from the database, performs the analysis and renders the statistics in the form of charts, graphs and tables.
5	User reviews EV Charging Usage and the trends and identifies potential issues.		

Alternate Flow			
User : City Manager		System	
		4.1	System displays an error saying “EV Station data unavailable please check back after few mins”

- Special Requirements: N/A
- Pre-Conditions
  - Users must be authenticated and logged in
  - EV stations location and usage data available in Database.
  - Dashboard already rendered with the visualizations i.e. all calculation for the visualization already done in the background and cached.
- Post- Conditions
  - EV charging station location and usage data displayed on the page.

Use Case: *Display recommendations for adding EV Charging points*

1. Description: This use case provides the city manager recommendations on adding new EV Charging points based on the EV usage stats and the increase in EV cars in the city.

Actors:
- City Manager: Review the recommendation by the system.

Triggers:
- When the values for EV cars and station usage exceeds a certain threshold. This triggers the system to generate the recommendations.

Input:
- EV charging point usage data and EV cars data

2. Flow of Events

Basic Flow			
User : City Manager		System	
1	User selects “Car Indicator”		
		2	System opens the dashboard for cars with different tabs.
		3	System retrieves the recommendations from the pre-generated list of recommendations.
4	User gets the notification for recommendations		
5	User reviews recommendations.		

Alternate Flow			
User : City Manager		System	
		3.1	System fails to retrieve the recommendations from the pre-generated list of recommendations.

- Special Requirements: N/A
- Pre-Conditions
    - EV usage and EV cars data in Database.
    - Backend Engine already processed the data and recommendations already generated and stored.
- Post- Conditions
    - Recommendation visible to the user.

Use Case Name: *Display/Get High Congestion Traffic Points*

1. Description

Provides city manager a live map/list of current and predicted vehicular congestion hotspots, including severity, duration, and contributing factors (incidents, events, weather).

Actors:

- City Manager

Triggers:

- The user manually opens the “Traffic Congestion” panel.

Inputs:

- Live congestion data from relevant APIs.
- Historical congestion data based on the recorded live data.
- Predicted congestion data based on the output of a prediction algorithm.

2. Flow of Events

Basic Flow			
User		System	
1	User opens “Traffic Congestion” panel		
2	User chooses live/historical/prediction mode		
		3	The system fetches relevant data from external API/database/prediction model
		4	The system displays the congestion data on a map and as a list of high-congestion areas

Alternative Flow 1 - Data not available			
User		System	
		4.1	The system shows an alert to the user if the data is not available

3. Special Requirements:
- Availability of external live traffic congestion data.
- Periodic background data fetching and storage for historical mode.
- Availability of the prediction model trained on historical data, and taking into account incidents, events, etc. The prediction model should be robust to make relevant predictions, responsive to changes in current conditions, and sufficiently fast.

4. Preconditions
- The user is authenticated and has sufficient permissions to access the traffic congestion panel.
- The external APIs supplying the live data are available.
- Historical data has been recorded by the system for a relevant period of time.
- The prediction model has been trained on relevant past data and has access to necessary auxiliary data (e.g. events).

5. Postconditions
- A map showing traffic congestion is displayed (in relevant live/historical/prediction mode).
- A list of currently congested areas is displayed.

Cycle Use Case Box:

Use case : *Display recommendation to add/remove bikes and stations*

1. Description
The goal of this use case is to generate recommendations to match supply-demand for usage of bikes across the city. Show recommendations to add/remove bikes from a station, deploy/remove stations so that rider demand is met efficiently.
Actors

- City Managers - Review and Authorize recommendations
- Bike Service Providers - Execute Recommendations

Triggers

- Station bike count falls below minimum availability threshold
- Analysis report shows under/over use of bikes at different stations
- Scheduled event is expected to increase/decrease demand

Inputs

- Real time Bike data from Bike APIs
- Insights from data analysis report

2. Flow of Events

Basic Flow			
User : City Manager		System	
1	Monitor Dashboard for recommendations on Bikes data.		
		2	Whenever a trigger fires, generate and display a recommendation.
3	Review the system generated recommendation		

3. Special Requirements
N/A
4. Preconditions

- User authentication and Login
- Live Bike data via APIs should be available
- Data Analysis system should have provided latest insights from historical data
- Threshold values for minimum/maximum bike availability must be defined

5. Postconditions

- Recommendation status has been updated
- Bike service providers should have executed the approved actions

Use case: *Simulation effects of changes in bike numbers and stations*

1. Description

The main objective is to allow city managers and bike service providers to simulate the impact of changing the number of bikes or bike station location. By running the simulation, managers and bike service providers can see how redistribution of bikes or increasing the capacity of the bike stations can affect the bike availability and user demand.

Actors

- City managers
- Bike service providers

Trigger

- Recommendations of change in bike numbers or stations are prompted to the city managers and bike service providers
- City managers can also choose to simulate the changes in bike numbers and station location.

Inputs

- Current bike availability and their usages
- Historical data of bike usages.
- Real time data of bike availability in each stations

2. Flow of events

User - City Manager /Service provider		System	
1	Logged in and go to suggestion page		
		2	Shows recommendations on new bike addition or stations
3	User will review these recommendations and simulate the changes to see the effects		

3. Special requirements

N/A

4. Precondition

- The system has access to historical and real time data of bike usages.
- The simulation module should be active and ready to process scenarios
- The city manager and bike service provider logged in and have access to the bike changes simulation.

5. Postcondition

- The simulation results are shown on the dashboard showing projected bike availability and congestion patterns.
- City managers can compare different scenarios by adding bikes and stations and make changes if necessary.

Use Case: *Display Cycle Stations and Usage*

1. Description

This use case enables the Cycle Provider to view real-time information about bike-sharing stations across the city, including their locations, current bike availability, and usage patterns. The goal is to provide visibility into the cycle network's operational status to support decision-making for station management, bike redistribution, and service optimization.

Actors

- Cycle Provider: Reviewing, managing
- City Manager: Reviewing, managing

Triggers:

- Cycle Provider logs into the system and navigates to the cycle stations dashboard
- Scheduled automatic refresh of station data
- Cycle Provider requests updated station information

Inputs:

- User authentication credentials
- Optional filters (e.g., specific geographic area, time period, station ID)
- Refresh/update request from the Cycle Provider

2. Flow of events

Basic Flow			
User		System	
1	Cycle Provider logs into the system and selects "Display cycle stations and usage" option		
		2	The system displays all available stations on a map interface with bike availability and usage data
4	Cycle Provider reviews the displayed information		

Alternative Flow 1			
User		System	
		1.1	The system detects that station data is unavailable or connection to stations is lost, displays error message indicating data retrieval issue and suggests user retry or contact system administrator

3. Special Requirements

- Real-Time Data Feed**:** Requires a continuous and reliable data feed from cycle stations to ensure current status accuracy.

4. Preconditions

- User authentication

- Database containing station information and usage history must be accessible
- Map service API must be accessible and functional

5. Postconditions

- Cycle Provider has successfully viewed current station locations and real-time usage data and has the information needed to make operational decisions (e.g., bike redistribution, maintenance scheduling)

USE CASE: *Demand Analysis of Cycle Stations with consistent high/low usage*

1. Description

Goals and Responsibilities Analyze 60-90 days of historical demand data to identify stations with consistent high or low usage patterns. Segments stations by demand type (commute corridors, tourist areas, residential). Enables city managers to understand demand drivers and make network optimization decisions.

Actors

- City Manager / Network Planner: Analyzes demand, plans network optimization
- Rebalancing Coordinator: Uses demand forecasts to optimize distribution routes
- System: Aggregates historical data, calculates demand metrics, identifies patterns

Triggers and Inputs

- User navigates to demand analysis dashboard
- System completes daily/weekly demand aggregation from historical data

Inputs: 60-90 days historical availability, real-time Dublin Bikes API, static station data, checkout/return transaction data

2. Flow of Events

Basic Flow

User - City Manager		System	
1	Navigates to demand analysis dashboard		
		2	Queries 60-90 days of historical availability data
		3	Calculates demand metrics: avg checkouts/hour, peak hours, day-of-week patterns, occupancy trends
		4	Segments stations: high-demand (commute), tourist, residential (low/steady), mixed
		5	Displays stations color-coded by usage intensity
6	User selects station to view demand profile		
		7	Shows hourly demand curve, day-of-week breakdown, month-over-month growth, checkout vs. return patterns
8	City Manager identifies: peak commute corridors, tourist hotspots, residential zones		
		9	Calculates demand variability (coefficient of variation) and forecasts upcoming peak hours

10	City Manager models scenarios: consolidate low-demand stations, expand high-demand corridor		

Alternative Flow 1: High-Demand Growth

User - City Manager		System	
		1.1	Detects demand increasing >10% month-over-month
		1.2	Recommends capacity increase or new satellite station
1.3	City Manager sees growth alert		

Alternative Flow 2: Declining Demand

User - City Manager		System	
		2.1	Detects sustained demand <10% occupancy
		2.2	Recommends capacity increase or new satellite station
2.3	City Manager reviews low-demand station		

3. Special Requirements
Dependencies
- 60-90 days historical availability data aggregated and stored
- JCDecaux API key

4. Preconditions
- User logged in
- 60-90 days historical data collected and stored
- Demand metric algorithms implemented
- Geographic clustering capabilities available
- Static station data loaded

5. Postconditions
- Demand analysis dashboard segments stations by usage intensity
- Usage trends visible: growing, declining, stable patterns
- Hourly demand curves available per station
- Demand profiles identified: commute, tourist, residential, mixed
- Month-over-month and year-over-year growth metrics tracked
- Directional flow imbalances visible
- Rebalancing forecasts guide staff optimization
- Capacity planning decisions informed by historical trends

Use Case: *Analyze the times of day each station is (nearly) full or empty*

1. Description

Show per-station patterns/times when docks are nearly full or nearly empty (historical), and forecast upcoming risk windows.

Actors:

- City manager
- Cycle provider

Triggers:

- The user manually opens the "Analyze bike station usage" panel.

Inputs:

- Static bike station location data periodically updated from relevant APIs.
- Live usage data from relevant APIs.
- Historical usage data based on the recorded live data.
- Predicted usage data based on the output of a prediction algorithm.

2. Flow of Events

Basic Flow			
User		System	
1	User opens "Analyze bike station usage" panel		
2	User chooses historical/prediction mode		
		3	The system fetches relevant data from database/prediction model
		4	The system displays the usage data on a map and as a list of underutilised or overutilised bike stations

Alternative Flow 1 - Data not available			
User		System	
		4.1	The system shows an alert to the user if the data is not available

3. Special Requirements:

- Availability of external live bike station usage data.

4. Preconditions

- The user is authenticated
- The static bike station location data has been recorded by the system.
- The external APIs supplying the live data are available.
- Historical data has been recorded by the system for a relevant period of time.

- The prediction model has been trained on relevant past data and has access to necessary auxiliary data (e.g. events).

5. Postconditions
- A map showing bike stations and their usage is displayed (in relevant live/historical/prediction mode).
- A list of currently underutilised or overutilised bike stations is displayed.

Use Case: *Detect areas with low access to bikes for adding new stations*

1. Description
Here, we identify city locations where bike-sharing access is lacking, enabling data-driven recommendations for adding new bike stations. The system analyses historic cycle usage data (bike checkouts/returns), existing station locations, and pedestrian footfall trends. By integrating these datasets, it locates areas with consistently high pedestrian traffic but few or no bike access points, prioritizing such zones for station expansion.
Actors
- City Manager
- Cycle Provider

Triggers & Inputs:
- The user navigates to the low-access area detection filter in the cycle dashboard.

2. Flow of Events:

Basic Flow			
User		System	
1	User Opens low-access analysis dashboard		
		2	System loads and processes historic cycle and footfall data
		3	Retrieves static station positions.
		4	Calculates coverage gaps (areas with >X foot traffic and >Y distance from station)
		5	Displays city map highlighting underserved zones
		6	System shows data-driven suggestions for new station locations
7	User reviews identified areas		

Alternative Flow

User		System	
1	User adjusts threshold settings (e.g., minimum footfall, access distance) to refine results		
		2	Displays city map highlighting underserved zones calculated using the given threshold settings.

3. Special Requirements

- Must integrate historic data APIs for pedestrian footfall data, and cycle data.
- Geographic heatmap generation and spatial clustering/analysis
- Dashboard capable of real-time visualization and low-latency updates.

4. Preconditions

- Dashboard already rendered with the visualizations i.e. all calculation for the visualization already done in the background and cached.
- User authenticated and authorized
- At least 30 days of recent cycle & footfall data available
- Geographic/mapping capabilities enabled

5. Postconditions

- Dashboard shows map of low-access areas with prioritization for new stations
- List or map view of recommended new locations for city manager action
- Insights stored for simulation and recommendation modules.

Use Case: *Demand Analysis of Cycle Stations with consistent high/low usage*

1. Description

Identify underserved areas with consistently low bike or dock availability. The system tracks availability trends over 30-90 days to distinguish systemic access problems from temporary stockouts. Enables city managers to optimize station placement and improve network coverage equity.

Actors

- City Manager: Identifies coverage gaps, plans new station locations
- Service Provider: Monitors station health, rebalances bikes
- System: Integrates real-time bike data, analyzes historical patterns, calculates coverage metrics

Triggers and Inputs

- Trigger 1: Station shows <20% bike availability for 40%+ of peak hours over rolling 7 days
- Trigger 2: Station consistently has 0 free docks during peak hours
- Trigger 3: User navigates to low-access detection dashboard
- Inputs: Real-time bike/dock data (Dublin Bikes API), historical 30-90 day trends, static station positions and capacities

2. Flow of Events

Basic Flow			
User : City manager		System	
1	Navigates to the low-access tab, in the cycle dashboard		
		2	Retrieves real-time Dublin Bikes API data (updates every minute)
		3	Loads static station data (positions, stand counts, capacity)
		4	Queries historical availability data (past 30-90 days) on demand
		5	Calculates low-access metrics: percentage of bikes available, 0 free docks frequency
		6	Displays map showing stations color-coded by access level (green/yellow/red)
7	User reviews underperforming stations and selects one		
		8	Shows availability trend, peak shortage hours, day-of-week patterns, nearby stations
9	City Manager identifies coverage gaps and proposes new station location		

Alternative Flow 1: Geographic Coverage Gap

Alternative Flow 1: Geographic Coverage Gap			
User : City manager		System	
		1.1	Identifies neighborhoods where nearest functioning station is >800m away
		1.2	Flags as critical gap requiring new capacity
1.3	City Manager sees gap alert and views recommended new station locations		

Alternative Flow 2: Imbalanced Network

Alternative Flow 2: Imbalanced Network			
User : City manager		System	
		2.1	Detects stations consistently empty while others overflow
		2.2	Recommends rebalancing strategy or new satellite station
2.3	City Manager receives alert and can authorize emergency rebalancing		

3. Special Requirements
3.1 Dependencies

- JCDecaux -Dublinked (or similar) API integration (contract_name="dublin") with 1-minute update frequency
- Static station data cached and accessible via API or CSV
- Historical data storage: 30-90 days minimum for pattern identification
- Geographic analysis: coverage radius mapping, nearest-station distance calculation
- Heatmap generation: demand patterns by location and time of day
- Rebalancing impact tracking: pre/post comparison

4. Preconditions
- JCDecaux API key obtained from https://developer.jcdecaux.com/
- Dublin Bikes API access registered
- Static station data (positions, capacities) loaded
- 30+ days historical availability data collected
- Geographic/mapping capabilities available
- User authenticated

5. Postconditions
- Low-access dashboard displays underserved areas with color-coded severity
- Coverage gap map identifies neighborhoods requiring new stations
- Geographic heatmaps show areas with persistent access problems
- Recommendations generated for new station placement
- All analysis logged for tracking network improvements
- City Manager can propose backed-by-data station placement decisions

Use Case: *Origin-Destination Heatmaps for Cycle Routes*

1. Description
The visualized heatmap displays the most common cycle routes by shading station locations and connecting paths according to trip frequency. Areas and routes with higher bike activity appear more prominently, allowing users to instantly identify popular origins, destinations, and travel corridors. This intuitive visualization aids in understanding usage patterns and supports effective decision-making for resource allocation and infrastructure planning.

Actors
- City Manager
- Cycle Provider

Triggers and Inputs
- The user initiates a request to view the heatmap through the system interface.

2. Flow of Events

Basic Flow			
User : City manager		System	
1	User selects “Cycle Dashboard”		
2	User selects “View Cycle Heatmap”		
		3	System retrieves historical and real-time data for individual cycle trips from APIs and databases.

		4	System tracks the frequency of each unique trip path
		5	System visualizes these frequencies in a map overlay
6	User reviews heatmap and identifies potential optimization areas.		

Alternative Flow 1			
User : City manager		System	
1	User selects "Cycle Dashboard"		
2	User selects "View Cycle Heatmap"		
		3	The system displays an error : "Origin-Destination Heatmap unavailable."

3. Special Requirements
- Must integrate Dublin bikes real-time and historic APIs.
- Dashboard capable of real-time visualization and low-latency updates.

4. Preconditions
- Real-time and historical cycle data accessible.
- Dashboard already rendered with the visualizations i.e. all calculation for the visualization already done in the background and cached.
- Geographic/mapping capabilities enabled
- User is authenticated.

5. Postconditions
- Origin-Destination heatmap is displayed.
- Popular trip paths, origins, and destinations, and high usage areas are identified.
- Insights stored for simulation and recommendation modules.

Bus Use Case Box:

Use Case: Display Live Bus Data

1. Description

The system displays live data for each bus, including its origin (source), current location, and intended destination. The goal is to provide the User with up-to-the-minute visibility into bus movements and positions, enabling effective monitoring, route management, and operational oversight.

Actors
- Bus Provider: Reviewing, managing
- City Manager: Reviewing, managing

Trigger:
- The Bus Provider initiates a request to view live bus data through the system interface.

Inputs:

- Selection of specific bus routes or all routes
- Time range for tracking data
- Filters for specific routes, bus number, destinations, or operational status

2. Flow of Events

Basic Flow			
User		System	
1	Bus Provider logs into the system and navigates to the bus fleet monitoring section and selects "Display live bus data" option		
		2	System retrieves real-time GPS and operational data from all active buses, processes the location data, matches it with route information and displays live bus data.
		3	System continuously updates the display as buses move (auto-refresh)
4	Bus Provider reviews the displayed live bus information		

Alternative Flow 1			
User		System	
		1.1	The system detects that data is unavailable or connection to several buses is lost due to connection outage, displays error message indicating data retrieval issue, suggests user retry or contact system administrator, identifies one or more buses with lost GPS connection, marks affected buses with warning indicator, displays last known position with timestamp and continues to display data for buses with active connections
1.2	User can refresh or check back later		

Alternative Flow 2	
User	System

2.1	User selects specific route number from filter options		
		2.2	System retrieves data only for buses operating on selected route and displays all buses on that route with their respective positions

3. Special Requirements
Real-time Bus data (routes, individual train, schedules, station, delays)
4. Preconditions
- User authentication
- Bus route information (source, destination, stops) must be pre-loaded in the system database
- Map service API must be accessible and functional
- Map topology (stops, bus icons with numbers, etc.) must be preloaded to the map

5. Postconditions
- The Bus Provider has successfully viewed real-time bus location data including source, current position, and destination and has the information needed to make operational decisions

USE CASE: *Display Live Bus delays*

1. Description
Goals and Responsibilities Real-time Dublin Bus delay visibility using National Transport API GTFSR v2. Calculate impact using historical passenger volume. Enable city managers to optimize routes and frequency; enable service providers to coordinate real-time responses.
Actors
- City Manager: Strategic route optimization based on patterns
- Service Provider / Dispatcher: Real-time operations and driver coordination

Triggers
- Bus running behind schedule (2-5 min threshold for local routes)
- User requests live delays dashboard
- System receives National Transport API update (every 30-60 sec)

Inputs
- Real-time bus positions (GTFSR v2), static schedules (GTFS)

2. Flow of Events
Basic Flow

User - City Manager/Service Provider		System	
1	Navigates to live delays dashboard		
		2	Retrieves real-time bus data
		3	Loads static GTFS data (stops, routes, schedules)
		4	Calculates delay for each bus vs. schedule

5	User reviews delay: route, location, delay duration, affected passengers		
		6	Continues retrieving updates every minute

Alternative Flow 1: No Delays

User - City Manager/Service Provider		System	
	-	1.1	Shows “all buses on schedule”
		1.2	Displays on-time performance statistics and network metrics

Alternative Flow 2: API Unavailable

User - City Manager/Service Provider		System	
	-	3.1	API returns error or timeout
		3.2	Uses cached data with warning "Data from X minutes ago
		3.3	Retries connection at exponential backoff intervals

3. Special Conditions
- National Transport API GTFSR v2 integration with error handling and caching
- Static GTFS data cached locally (updated weekly)
- Impact scoring algorithm: delay × hourly passenger percentage
- Pattern detection analytics for trend identification
- Route optimization tools with historical data modeling

4. Preconditions
- National Transport API registered access and API key
- GTFS static data, Passenger volume data and historical data loaded.
- User authenticated

5. Postconditions
- Real-time dashboard updated with accurate delays
- Impact-prioritized view focuses on high-effect delays
- Pattern insights available for strategic planning
- All events logged for performance analysis
- Service optimization recommendations available

Use Case Name: *Display common bus delays*

1. Description
Identify recurring bus delay patterns by route, stop, and time-of-day.
Actors:
- City manager(s)
- Bus provider(s)

Triggers:

- The user manually opens the “Analyze common bus delays” panel.

Inputs:

- Static routes and stop location data periodically updated from relevant APIs.
- Live delay from relevant APIs.
- Historical delay data based on the recorded live data.

2. Flow of Events

Basic Flow			
User		System	
1	User opens “Bus indicators” panel and clicks “Analyze common bus delays”		
2	User chooses historical time window and optional route/stop/time-of-day filters		
		3	The system queries the relevant data from the database
		4	The system displays a list of common bus delays with information on routes, stops, and time of day

Alternative Flow 1 - Data not available			
User		System	
		4.1	The system shows an alert to the user if the data is not available

3. Special Requirements:

- Availability of external live bus delay data.
- Periodic background data fetching and storage for historical mode.
- Periodic (less frequent) background static bus route/stop location data fetching and storage.
- Internal data aggregation to extract common/recurring bus delay patterns.

4. Preconditions

- The user is authenticated and has sufficient permissions to access the common bus delays panel.
- The static route/stop location data has been recorded by the system.
- Historical data has been recorded by the system for a relevant period of time.

5. Postconditions

- A list of common bus delays with information on routes, stops, and time of day is displayed.

Use Case: Display data for bus trips utilization: under/over use

1. Description

The goal of this use case is to allow City Managers and Bus Service Providers to view real-time and historical bus trip utilization data. The system highlights underused or overused bus routes to optimize service delivery and resource allocation.

Actors

- City Manager – Reviews utilization data and identifies routes requiring changes.
- Bus Service Provider – Monitors operational usage to adjust fleet deployment.

Triggers and Inputs

- Trigger: User selects "Bus Trips Utilization" from the dashboard.
- Scheduled refresh of real-time or historical bus data.
- Inputs: Bus occupancy, route information, historical usage data, thresholds for under/overuse.

2. Flow of Events

Basic Flow			
User : Bus service provider		System	
1	User selects "Display Bus Trips Utilization.		
		2	System retrieves historical and real-time bus data from APIs and databases.
		3	System calculates utilization per route and classifies them as underused, overused, or optimal.
		4	System visualizes data on a dashboard with maps and charts.
5	User reviews utilization and identifies potential optimization areas.		

Alternative Flow 1			
User : Bus Service Provider		System	
1	Requests bus utilization analysis		
		2	Real-time feed unavailable
		3	Predicts utilization from historical patterns
		4	Displays predicted utilization and alerts user

5	The user reviews utilization and identifies potential optimization areas.		

3. Special Requirements

Dependencies

- Must integrate heterogeneous data sources (JSON, XML, REST APIs).

4. Preconditions

- Historical and real-time bus data is available and Must predict data using historical trends if live sources fail.
- Thresholds for overuse/underuse are predefined.
- User is authenticated.

5. Postconditions

- Dashboard displays up-to-date bus utilization.
- Overused and underused routes are identified and logged.
- Insights available for further recommendations.

Use Case: *Display recommendations on changing bus routes/frequencies*

1. Description

This use case mainly focuses on optimizing the city bus service by analyzing the historical data (Static data) and real-time data to detect congestion, delays, overused and underused routes/buses. The System will generate recommendations to increase, decrease or reroute these services.

Actors

- City Manager
- Bus Service provider

Triggers

- Increase in congestion in particular areas.
- Delays in bus services.
- Underuse of particular routes
- Overuse of particular routes

Inputs

- Clicking on show recommendations in the respective services dashboard.

2. Flow of events

User - City Manager / Bus Service provider		System	
1	Logged in		
		2	Analyzes the historical data, real time data and usage of buses and shows recommendations like addition/reduction of buses.

3	City manager will review these recommendations and send these to bus service provider		
4	Bus service provider reviews the recommendation request and approve if its possible		

3. Special Requirements
N/A
4. Precondition
- The system has access to both historical and real time data sources.
- The predictive and route optimization modules must be active.

5. Postcondition
- The system generates recommendations based on the analysis of congestion levels, usage and delays.
- Recommendations like increase in bus services in high demand areas, reducing or reallocation of underused services and modifying routes in congested areas.
- The city managers reviews and approves these recommendations and then forwards them to bus service providers.
- The bus service providers review these recommendations and take necessary action.

Use case: *Simulate effects of changes in bus resources/routes*

1. Description
The main objective of this use case is to allow city managers and bus service providers to simulate the impact of recommended changes like adding, removing or rerouting the bus services before actually implementing them. This will allow them to take an informed decision about the changes.
Actors
- City managers
- Bus service providers

Triggers
- The city manager chooses to simulate the recommended changes before approving
- The bus service providers can also simulate by choosing this simulate option.

Inputs
- Reviewing all the recommendations provided by the system
- Clicking on the simulate button on the promising recommendations.

2. Flow of events:

User - City Manager		System	
1	Logged in		

		2	Shows recommendations on any of the services
3	City manager/ Bus service provider will review these recommendations and simulate the changes to see the effects		

3. Special requirements
N/A
4. Precondition
- Recommendation for bus changes generated by the system.
- The simulation module is operational and has access to historical and real time data.
- The city manager or bus service provider logged into the dashboard with access to simulation features.

5. Postconditions
- The simulation results are shown on the dashboard showing projected effects on congestion and delays.
- The city manager and respective service providers can review based on the simulation results and make informed decisions.

Use case: *Suggest alternative transport modes during bus disruptions*
1. Description
The goal of this use case is to suggest alternate transport options to passengers affected by bus disruptions. Show alternate transport services nearby which passengers can use instead.

Actors
- City Service Providers – Receive alternate routes options and convey to passengers

Triggers
- A Bus disruption is reported.

Inputs
- Real Time data for reported disruptions having information about incident location.

2. Flow of Events

Basic Flow			
User: Bike Service Provider		System	
		1	Update Dashboard when there is a bus disruption reported.
		2	Generate alternate transport options passengers can use according to their current location

		3	Send recommendation to bus service provider
		4	Change disruption status to in progress.
5	Receive alternate routes and convey to Passengers		
6	Update disruption status to Solved		
		7	Change disruption status to Solved.

3. Special Requirements
- Real-time bus disruption data.

4. Preconditions
- User authentication and Login

5. Postconditions
- Disruption status has been updated

Use Case: *Display Possible Reasons for Bus Delays based on Other Services*

1. Description

This use case enables the dashboard to inform users about probable causes for bus delays by aggregating and analysing multiple external datasets. The system utilizes live construction site data, public event schedules, real-time traffic congestion reports, and live transport data feeds (cars, tram, etc.).

Actors
- Bus Provider
- City Manager

Trigger:
- The threshold requirement for being classified as a delay or disruption is met. (ex. Live Traffic congestion level, traffic accident reports, etc)

Inputs:
- Live delay status from bus data feed
- Construction site locations and schedules
- Major event data (venues, time windows)
- Real-time traffic congestion data
- Live feeds from train and tram services (for disruptions or delays)

2. Flow of Events

Basic Flow	
User	System

		1	The system continuously scans the data feed from multiple APIs to check for the bus delay threshold conditions.
		2	When threshold is met, highlight the bus, the route, and the live journey along with other details.
		3	System identifies the relevant threshold factors and displays the details of the causes.
		4	Visualize the disruption on a map overlay.
		5	System shows an alert on the dashboard.
6	User receives the alert and is directed to the bus dashboard.		
7	User reviews the map with the highlighted bus along with its details along with relevant causes.		

Alternative Flow			
User		System	
		1	System's continuous threshold check for bus delays is not met.
		2	System displays the map view without any highlighted buses and a message "No external delay factors detected".

3. Special Requirements
- Must integrate real-time APIs for bus data.
- Must integrate real-time APIs for data of other indicators to perform threshold check.
- Dashboard capable of real-time visualization and low-latency updates.

4. Preconditions
- Dashboard already rendered with the visualizations i.e. all calculation for the visualization already done in the background and cached.
- User authenticated and authorized

5. Postconditions
- An alert is sent to the city manager and the bus provider.
- A map is displayed in the bus dashboard highlighting the relevant bus.
- Insights stored for simulation and recommendation modules.

Train Use Case Box:

Use Case: *Display Live Train Data*

1. Description

This use case allows the Train Provider to access and view real-time information about their train fleet. The system displays comprehensive live data for each train, including its origin station (source), current location along the route, and intended destination station. The goal is to provide the Train Provider with up-to-the-minute visibility into train movements and positions, enabling effective fleet monitoring, schedule management, and operational coordination across the rail network.

Actors
- Train Provider
- City Manager

Trigger:
- The Train Provider initiates a request to view live train data through the system interface.

Inputs:
- Selection of specific train lines or all lines
- Time range for tracking data
- Filters for specific routes, train IDs, destinations, train types, or operational status

2. Flow of Events

Basic Flow			
User		System	
1	User selects train Dashboard		
		2	The system retrieves real-time tracking and operational data from all active trains.
		3	System continuously updates the display as trains move (auto-refresh)
4	Train Provider reviews the displayed live train information		

Alternative Flow 1 - Data not available			
User		System	
		2.1	The system shows an alert to the user if the data is not available

Alternative Flow 2			
User		System	
1.1	User enters specific train ID/number/route or selects from dropdown		
		2.1	System retrieves data only for the selected train/route, displays detailed view with complete route path and all

			scheduled stops highlighted/all trains scheduled for that path and shows detailed journey progress with estimated arrival times for remaining stops/detailed schedule for each train on that route

3. Special Requirements
- Real-time Train data (routes, individual train, schedules, station, delays)

4. Preconditions
- User authentication
- Train route information (source station, all intermediate stops, destination station) must be pre-loaded in the system database and railway network topology (stations, tracks, junctions) must be accurately mapped in the system
- Map service API must be accessible and functional

5. Postconditions:
- The Train Provider has successfully viewed real-time Train location data including source, current position, and destination and has the information needed to make operational decisions

Use Case: *Display live train delays*

1. Description
The system displays live data on train delays, including cascading effects.
Actors
- Train Provider
- City Manager

Triggers and Inputs
- Train running behind schedule
- Delay affects connecting train
- Inputs: Real-time train positions, static schedule.

2. Flow of Events

Basic Flow			
User		System	
1	Navigates to live train delays dashboard		
		2	Retrieves real-time train data
3	User reviews: line, location, delay, connected service impacts		
		4	Updates every 30 seconds
5	User selects train to view all affected stops, estimated delays		

	at each, connected services impact		

Alternative Flow 1: Cascading Delays

User - City Manager/Service Provider		System	
		4.1	Detects delayed train on Line A prevents on-time departure of connecting train Line B
		4.2	Calculates ripple effects across network (Dependency graph, ML)
4.3	Dashboard highlights dependency chain and all affected connections		

Alternative Flow 2: Critical Delay

User - City Manager/Service Provider		System	
		3.1	Detects delay >30 minutes
		3.2	Generates alerts
3.3	The user views the alert		

Alternative Flow 3: Data unavailable

User - City Manager/Service Provider		System	
		2.1	Data is unavailable
		2.2	An error message is displayed

3. Special Requirements
Dependencies

- Real-time train position data
- Comprehensive schedule data with station stop details
- Cascading delay calculation engine tracking connection dependencies

4. Preconditions

- User is authenticated and has sufficient permissions
- Live train delay data is available through the external API
- Static data on train stops and schedules is available

5. Postconditions

- Real-time dashboard shows accurate delays on all trains
- Cascading impacts visible (dependency chains)

Use Case: *Display common train delays*

1. Description
Identify recurring train delay patterns by route, stop, and time-of-day.
Actors:

- City manager

- Train provider

Triggers:
- The user manually opens the “Analyze common train delays” panel.

Inputs:
- Static routes and stop location data periodically updated from relevant APIs.
- Live delay from relevant APIs.
- Historical delay data based on the recorded live data.

2. Flow of Events

Basic Flow			
User		System	
1	User opens “Train indicators” panel and clicks “Analyze common train delays”		
2	User chooses historical time window and optional route/stop/time-of-day filters		
		3	The system queries the relevant data from the database
		4	The system displays a list of common train delays with information on routes, stops, and time of day

Alternative Flow 1 - Data not available			
User		System	
		4.1	The system shows an alert to the user if the data is not available

3. Special Requirements:
- Availability of external live train delay data.
- Periodic background data fetching and storage for historical mode.
- Periodic (less frequent) background static train route/stop location data fetching and storage.
- Internal data aggregation to extract common/recurring train delay patterns.

4. Preconditions
- The user is authenticated and has sufficient permissions to access the common train delays panel.
- The static route/stop location data has been recorded by the system.
- Historical data has been recorded by the system for a relevant period of time.

5. Postconditions
- A list of common train delays with information on routes, stops, and time of day is displayed.

Use Case: *Display data for train trips utilization: under/over use*

1. Description
This use case allows City Managers and Train Service Providers to monitor train trip utilization. Overused and underused train routes are identified to optimize service schedules and operational efficiency.

Actors
- City Manager – Analyzes usage patterns to optimize mobility.
- Train Service Provider – Monitors route occupancy to adjust train schedules.

Triggers and Inputs
- Trigger: User selects “Train Trips Utilization” on the dashboard.
- Inputs: Real-time train positions, historical ridership, route data, thresholds for under/overuse.

2. Flow of Events

Basic Flow			
User : Train service provider		System	
1	User selects “Display Train Trips Utilization.		
		2	System retrieves historical and real-time train data from APIs and databases.
		3	System calculates utilization per route and classifies them as underused, overused, or optimal.
		4	System visualizes data on a dashboard with maps and charts.
5	User reviews utilization and identifies potential optimization areas.		

Alternative Flow 1			
User : Train Service Provider		System	
1	Requests train utilization analysis		
		2	Real-time feed unavailable
		3	Predicts utilization from historical patterns
		4	Displays predicted utilization and alerts user

5	User reviews utilization and identifies potential optimization areas.		

3. Special Requirements
- Integration with multiple heterogeneous train data sources (APIs, government open data).
- Predictive module for missing data.

4. Preconditions
- Train route, schedule, and occupancy data accessible.
- User is authenticated.
- Thresholds for utilization defined.

5. Postconditions
- Real-time and historical train utilization displayed.
- Overused and underused routes identified.
- Data stored for further recommendations.

Use Case: *Display recommendations on changing train routes/frequencies/train types*

1. Description

This use case mainly focuses on optimizing the city train service by analyzing the historical data (static data) and real-time data to detect delays, overused and underused routes. The System will generate recommendations to increase and decrease services.

Actors
- City Manager
- Train Service provider

Triggers
- Delays in train services.
- Underuse of particular routes
- Overuse of particular routes

Inputs
- Clicking on show recommendations in the train services dashboard.

2. Flow of events

User - City Manager / Train service provider		System	
1	Logged in		
		2	Analyzes the historical data, real time data and usage of trains and shows recommendations like addition/reduction of trains.

3	City manager and Train service providers will review these recommendations.		

3. Special Requirements
N/A
4. Precondition
- The system has access to both historical and real time data sources.
- The predictive and route optimization modules must be active.

5. Postcondition
- The system generates recommendations based on the analysis of usage and delays.
- Recommendations like increase in train services in high demand areas, reducing of underused services.
- The city managers review these recommendations.

Use case: *Simulate changes in train services*

1. Description

The main objective of this use case is to allow city managers and train service providers to simulate the impact of recommended changes like adding or removing the train services before actually implementing them. This will allow them to take an informed decision about the changes.
Actors
- City managers
- Train service providers

Triggers
- The city manager chooses to simulate the recommended changes before approving
- The train service providers can also simulate by choosing this simulate option.

Inputs
- Reviewing all the recommendations provided by the system
- Clicking on the simulate button on the promising recommendations.

2. Flow of events:

User - City Manager / Train Service providers		System	
1	Logged in		
		2	Shows recommendations on any of the services
3	City manager and train service providers will review these recommendations and simulate the changes to see the effects		

3. Special requirements
N/A

4. Precondition
- Recommendation for train changes generated by the system.
- The simulation module is operational and has access to historical and real time data.
- The city manager or train service provider logged into the dashboard with access to simulation features.

5. Postconditions
- The simulation results are shown on the dashboard showing projected effects addition and reduction of train services
- The city manager and train service providers can review based on the simulation results.

Use case: *Suggest alternative transport modes during train disruptions*

1. Description

The goal of this use case is to suggest alternate transport options to passengers affected by train disruptions. Show alternate transport services nearby which passengers can use instead.

Actors
- City Service Providers – Receive alternate routes options and convey to passengers

Triggers
- A train disruption is reported.

Inputs
- Real Time data for reported disruptions having information about incident location.

2. Flow of Events

Basic Flow			
User: Service Providers		System	
		1	Update Dashboard when there is a train disruption reported.
		2	Generate alternate transport options passengers can use according to their current location
		3	Send recommendation to service providers
4	Receive disruption information and alternate modes		

3. Special Requirements
- Real-time train disruption data.

4. Preconditions
- User is authenticated

5. Postconditions
- Disruption status has been updated

Use Case: *Display Possible Reasons for Train Delays based on Other Services*

1. Description

This use case enables the dashboard to inform users about probable causes for train delays by aggregating and analysing multiple external datasets. The system utilizes live construction site data, public event schedules, real-time traffic congestion reports, and live transport data feeds (cars, tram, etc.).

Actors
- Train Provider
- City Manager

Trigger:
- The threshold requirement for being classified as a delay or disruption is met. (ex. Live Traffic congestion level, traffic accident reports, etc)

Inputs:
- Live delay status from train data feed
- Construction site locations and schedules
- Major event data (venues, time windows)
- Real-time traffic congestion data
- Live feeds from bus and tram services (for disruptions or delays)

2. Flow of Events

Basic Flow			
User		System	
		1	The system continuously scans the data feed from multiple APIs to check for the train delay threshold conditions.
		2	When threshold is met, highlight the train, the route, and the live journey along with other details.
		3	System identifies the relevant threshold factors and displays the details of the causes.
		4	Visualize the disruption on a map overlay.
		5	System shows an alert on the dashboard.
6	User receives the alert and is directed to the train dashboard.		
7	User reviews the map with the highlighted train along with its details along with relevant causes.		

Alternative Flow			
User		System	
		1	System's continuous threshold check for train delays is not met.
		2	System displays the map view without any highlighted trains and a message "No external delay factors detected".

3. Special Requirements
- Must integrate real-time APIs for train data.
- Must integrate real-time APIs for data of other indicators to perform threshold check.
- Dashboard capable of real-time visualization and low-latency updates.

4. Preconditions
- Dashboard already rendered with the visualizations i.e. all calculation for the visualization already done in the background and cached.
- User authenticated and authorized

5. Postconditions
- An alert is sent to the city manager and the train provider.
- A map is displayed in the train dashboard highlighting the relevant train.
- Insights stored for simulation and recommendation modules.

Tram Use Case Box:

Use Case: *Display Live Tram Data*

1. Description

This use case displays live data for trams, including its origin stop (source), current location along the line, and destination stop. This should enable effective monitoring, headway management, service coordination, and operational response across the tram network.

Actors
- Tram Provider: A light rail/tram transportation service provider who operates and manages tram services.
- City Manager: Reviewing, managing and auditing.

Trigger:
- The tram Provider or City Manager initiates a request to view live tram data through the system interface.

Inputs:
- Selection of specific tram lines or all lines
- Time range for tracking data
- Filters for specific Tram/Fleet ID, routes, destinations, tram types, or operational status

2. Flow of Events

Basic Flow			
User		System	
1	Tram Provider or City Manager navigates to the tram section and selects "Display live tram data" option		
		2	System retrieves real-time tracking and operational data from all active trams, processes the location data, displays live tram data showing.
		3	System continuously updates the display as trains move (auto-refresh)
4	User reviews the displayed live tram information		

Alternative Flow 2			
User		System	
		2.1	API data unavailable, display error message

Alternative Flow 3			
User		System	
1.1	User enters specific tram ID/number/line (red or green) or selects from dropdown		
		2.2	System retrieves data only for the selected tram/line, displays detailed view with complete route path and all scheduled stops highlighted/all trams scheduled for that path and shows detailed journey progress with estimated arrival times for remaining stops/detailed schedule for each tram on that route

3. Special Requirements
   - Real-time tram data (routes, individual tram, schedules, station, delays)

4. Preconditions
   - User authorised
   - Tram route information (source stop, all intermediate stops, destination stop) schedule and timetable data

- Map service API must be accessible and functional

5. Postconditions:
- User has successfully viewed real-time tram location data including source, current position, and destination and has the information needed to make operational decisions

Use case: *Display live tram delays*

1. Description

Display Real-time tram delays using Luas Forecasting API. Enable operators to manage real-time incidents.

Actors
- City Manager: Frequency and capacity optimization
- Tram Service Provider: Real-time tram management

Triggers and Inputs
- Trigger: User requests live delays dashboard
- Inputs: Real-time tram positions (Forecasting API), static schedule

2. Flow of Events

Basic Flow

User - City Manager/Service Provider		System	
1	Navigates to live tram delays dashboard		
		2	Retrieves real-time tram data from Luas Forecasting API (Red/Green lines)
		3	Determines current hour and queries hourly passenger distribution
4	User reviews: line, location, delay, estimated affected passengers		
		5	Updates every 20-30 seconds
6	User selects line to view affected stations, hourly distribution curve, capacity status (synthetic data)		

Alternative Flow 1 - Data not available			
User		System	
		2.1	The system shows an alert to the user if the data is not available

3. Special Requirements

Dependencies

- Luas Forecasting API integration for Red and Green lines

4. Preconditions
- Luas Forecasting API registered access
- Static schedule data loaded
- User authenticated

5. Postconditions
- Real-time dashboard updated with accurate delays by line

Use Case: *Display common Tram delays*

1. Description
Identify recurring tram delay patterns by route, stop, and time-of-day.
Actors:
- City manager
- Tram provider

Triggers:
- The user manually opens the “Analyze common tram delays” panel.

Inputs:
- Static routes and stop location data periodically updated from relevant APIs.
- Live delay from relevant APIs.
- Historical delay data based on the recorded live data.

2. Flow of Events

Basic Flow			
User		System	
1	User opens “Tram indicators” panel and clicks “Analyze common tram delays”		
2	User chooses historical time window and optional route/stop/time-of-day filters		
		3	The system queries the relevant data from the database
		4	The system displays a list of common tram delays with information on routes, stops, and time of day

Alternative Flow 1 - Data not available			
User		System	
		4.1	The system shows an alert to the user if the data is not available

3. Special Requirements
- Availability of external live tram delay data.

4. Preconditions
- The user is authenticated and has sufficient permissions to access the common tram delays panel.
- The static route/stop location data has been recorded by the system.
- Historical data has been recorded by the system for a relevant period of time.

5. Postconditions
- A list of common tram delays with information on routes, stops, and time of day is displayed.

Use Case: *Display Tram Trips Utilization: Under/Overuse*

1. Description
This use case allows City Managers and Tram Service Providers to view tram usage patterns. The system identifies overused or underused tram routes and informs service adjustments.
Actors
- City Manager – Evaluates utilization for operational planning.
- Tram Service Provider – Adjusts tram frequency or routes.

Triggers and Inputs
- Trigger: User selects "Tram Trips Utilization" from dashboard.
- Inputs: Real-time tram positions, historical trip data, event data, thresholds.

2. Flow of Events

Basic Flow			
User : Tram service provider		System	
1	User selects "Display Tram Trips Utilization.		
		2	System retrieves historical and real-time Tram data from APIs and databases.
		3	System calculates utilization per route and classifies them as underused, overused, or optimal.
		4	System visualizes data on a dashboard with maps and charts.
5	User reviews utilization and identifies potential optimization areas.		

Alternative Flow 1	
User : Tram Service Provider	System

		3.1	API or database unavailable, display error to user

3. Special Requirements
- Must integrate Tram real-time APIs.
- Dashboard capable of low-latency, real-time visualization with interactive maps and charts.

4. Preconditions
- Real-time and historical tram data accessible.
- Thresholds for utilization defined.
- User logged in with proper credentials.

5. Postconditions
- Tram trip utilization displayed.
- Overused and underused routes identified.
- Insights stored for simulation and recommendation modules.

Use Case: *Display recommendations on changing Tram routes/frequencies*

1. Description

This use case mainly focuses on optimizing the city tram service by analyzing the historical data (static data) and real-time data to detect congestion, delays, overused and underused routes. The System will generate recommendations to increase and decrease services.

Actors
- City Manager
- Tram Service provider

Triggers
- Delays in tram services.
- Underuse of particular routes
- Overuse of particular routes

Inputs
- Clicking on show recommendations in the tram service dashboard.

2. Flow of events

User - City Manager		System	
1	Logged in		
		2	Analyzes the historical data, real time data and usage of tram and shows recommendations like addition/reduction of trams.
3	City manager and tram Service provider will review these recommendations.		

3. Special Requirements
N/A
4. Precondition
- The system has access to both historical and real time data sources.
- The predictive and route optimization modules must be active.

5. Postcondition
- The system generates recommendations based on the analysis of congestion levels, usage and delays.
- Recommendations like increase in tram services in high demand areas, reducing or reallocation of underused services.
- The users may review the recommendations.

Use case: *Simulate changes in tram services*

1. Description
The main objective of this use case is to allow city managers and tram service providers to simulate the impact of recommended changes like adding or removing the tram services before actually implementing them. This will allow them to take an informed decision about the changes.
Actors
- City managers
- Tram service providers

Triggers
- The city manager chooses to simulate the recommended changes
- The tram service providers can also simulate by choosing this simulate option.

Inputs
- Clicking on the simulate button on the recommendations provided by the system.

2. Flow of events:

User - City Manager		System	
1	Logged in		
		2	Shows recommendations on service
3	City manager and Tram Service providers will review these recommendations and simulate the changes to see the effects		

3. Special requirements
N/A
4. Precondition
- Recommendation for tram changes generated by the system.

- The simulation module is operational and has access to historical and real time data.
- The city manager or tram service provider logged into the dashboard with access to simulation features.

5. Postconditions

- The simulation results are shown on the dashboard showing projected effects addition and reduction of tram services
- The city manager and tram service providers can review based on the simulation results and make informed decisions.

Use case: *Suggest alternative transport modes during tram disruptions*

1. Description

The goal of this use case is to suggest alternate transport options to passengers affected by tram disruptions. Show alternate transport services nearby which passengers can use instead.

Actors

- City Service Providers – Receive alternate routes options and convey to passengers.

Triggers

- A tram disruption is reported.

Inputs

- Real Time data for reported disruption having information about incident location.

2. Flow of Events

Basic Flow			
User: Service Providers		System	
		1	Update Dashboard when there is a tram disruption reported.
		2	Generate alternate transport options passengers can use according to their current location
		3	Send recommendation to service providers
		4	Change disruption status to in progress.
5	Receive disruption information and recommendations		

3. Special Requirements

- Real-time tram disruption data.

4. Preconditions
- User authentication and Login

5. Postconditions
- Disruption status has been updated

Use Case: *Display Possible Reasons for Tram Delays based on Other Services*

1. Description
This use case enables the dashboard to inform users about probable causes for tram delays by aggregating and analysing multiple external datasets. The system utilizes live construction site data, public event schedules, real-time traffic congestion reports, and live transport data feeds (cars, tram, etc.).

Actors
- Tram Provider
- City Manager

Trigger:
- The threshold requirement for being classified as a delay or disruption is met. (ex. Live Traffic congestion level, traffic accident reports, etc)

Inputs:
- Live delay status from tram data feed
- Construction site locations and schedules
- Major event data (venues, time windows)
- Real-time traffic congestion data
- Live feeds from bus and tram services (for disruptions or delays)

2. Flow of Events

Basic Flow			
User		System	
		1	The system continuously scans the data feed from multiple APIs to check for the tram delay threshold conditions.
		2	When threshold is met, highlight the tram, the route, and the live journey along with other details.
		3	System identifies the relevant threshold factors and displays the details of the causes.
		4	Visualize the disruption on a map overlay.
		5	System shows an alert on the dashboard.
6	User receives the alert and is directed to the tram dashboard.		

7	User reviews the map with the highlighted tram along with its details along with relevant causes.		

Alternative Flow			
User		System	
		1	System's continuous threshold check for tram delays is not met.
		2	System displays the map view without any highlighted tram and a message "No external delay factors detected".

3. Special Requirements
- Must integrate real-time APIs for tram data.
- Must integrate real-time APIs for data of other indicators to perform threshold check.
- Dashboard capable of real-time visualization and low-latency updates.

4. Preconditions
- Dashboard already rendered with the visualizations i.e. all calculation for the visualization already done in the background and cached.
- User authenticated and authorized

5. Postconditions
- An alert is sent to the city manager and the tram provider.
- A map is displayed in the tram dashboard highlighting the relevant tram.
- Insights stored for simulation and recommendation modules.

Events Use Case Box:

Use Case: *Display Events*

1. Description

This use case displays all the events that are about to happen in the city.
Events data will be visualized on the city map within the events dashboard.
Actors:
- City Managers
- Service Providers

Triggers and inputs:
- Clicks on the events dashboard

2. Flow of Events

Basic Flow			
City Manager		System	
1	The user clicks on the events tab.		

		2	The System will retrieve the data from the data source and display details like type of event, location, timing on the map.
3	The user reviews the displayed events details		

Alternative Flow 1			
User		System	
		2.1	The system detects that event data is unavailable, it displays error message indicating data retrieval issue and suggests user retry or contact system administrator

3. Special Requirements
The system must have access to real-time or scheduled event data.
4. Precondition
- Events data must be properly obtained from private APIs like TicketMaster API
- The user must be properly authenticated to view the events details

5. Postcondition
- The dashboard displays event on the map
- Users can review these details

Use Case: *Display Recommendations for adding transportation entities during events*
1. Description
This use case focuses on keeping track of events happening across the city and help plan travel accordingly. For instance, when a large event is expected to draw big crowds, the system can recommend adding extra buses, tram services, or trains so that people can reach their destinations comfortably and without delays.
Heatmaps on the map will be generated based on event crowd density and the corresponding demand for additional transport services.

Actors
- City Managers
- Respective service providers (Bus, Tram, Train)

Triggers
- An alert will be sent to the city manager informing a large event that is about to take place in the coming days.

2. Flow of events

User	System

		1	Whenever a large event is about to take place, a recommendation is shown to add new transport in that area.
2	City managers and City service providers will review these recommendations.		

3. Special Requirements
N/A
4. Precondition
- Events data must be properly obtained from private APIs like TicketMaster API and transport information must also be properly obtained from government data source platforms.
- The user must be properly authenticated to view the recommendations

5. Postcondition
- The dashboard displays event information and recommendations.
- Recommendations are reviewed by the users.

Constructions Use Cases

Use Case: *Display Construction Work Data*

1. Description
This use case enables users to see the current locations of ongoing construction works across the city via a map overlay in the dashboard. The system pings both a database and external APIs to retrieve up-to-date information about active construction sites. The retrieved data such as site coordinates, project details, and status, is visualized interactively as markers or overlays on a live city map embedded in the dashboard.
Actors:
- City Manager
- Transport Provider

Triggers and Inputs:
- The user opens the construction work locations panel on the dashboard.
- Scheduled/automatic refresh for live updates.

Basic Flow			
User		System	
1	User selects “Construction Dashboard”		
		2	Queries database and APIs for current construction site data

		3	Receives, validates, and aggregates construction site data
		3	Displays real-time map overlay with markers for each site along with their details, highlighting any surrounding paths affected.
4	User clicks through site markers to identify problem areas affected by construction work.		

Alternative Flow			
User		System	
1	User selects "Construction Dashboard"		
		2	System displays an error : "Construction Dashboard unavailable."

3. Special Requirements
- Must integrate relevant real-time and historic APIs.
- Dashboard capable of real-time visualization and low-latency updates.
- Map overlay integration.

4. Preconditions
- Real-time and historical construction site data accessible.
- Dashboard already rendered with the visualizations i.e. all calculation for the visualization already done in the background and cached.
- User is authenticated.

5. Postconditions
- User has a clear, map-based view of active city construction sites
- Sites can be reviewed in detail via dashboard for further analysis or planning

Use Case: *Display Impact of Construction Work on Transport Routes*

1. Description
This use case explains how the City Manager views the operational impact of ongoing or planned construction work on public transport routes. The goal is to identify route disruptions, delays, or detours affecting buses, trains, or bike stations and support timely communication or re-routing decisions.
Actors:
- City Manager: Monitors and analyzes affected routes.

Triggers and Inputs:

- Triggered when the City Manager selects the “Construction Impact” view on the dashboard.

User - City Manager		System	
1	Selects the “Construction Impact” layer or filter.		
		2	Correlates construction zones with affected routes or stations and highlights impacted routes

3. Special Requirements
- Integration with city works or road authority APIs.

4. Preconditions
- Live or planned construction data available.
- Transport route and service datasets loaded.

5. Postconditions
- Dashboard displays all routes affected by construction work.
- City Manager can identify disruptions and coordinate with service providers or planners.

Further Use Cases:

Use Case: *Login*

1. Descriptions
   This use case describes the process by which any user accesses the system by providing valid credentials.
   Actors:
   - City Manager
   - Cycle Provider
   - Train Provider
   - Tram Provider
   - Bus Provider
   - Government Admin
   - Provider Admin

   Triggers and Inputs:
   - Username
   - Password

2. Flow of Events

Basic Flow	
User	System

1	Navigate to login page		
		2	Display login form (username/email and password fields)
	Enter credentials and submit	3	
		4	Validate provided credentials
		5	Credentials are valid, grant access.
6	User is redirected to dashboard.		

Alt. Flow No.	**Condition/Trigger**	**Actor Action/Choice**	**System Response**
1	Not registered	Click “Register”/“Sign up”	Display registration form
2	Credentials invalid after several attempts	Click “Forgot password”	Show recovery/reset password interface
3	MFA enabled	After password entry	Prompt for additional verification
4	Inactive/deactivated account	Try to login	Show account inactive/deactivated message
5	System unavailable	Try to login	Show system/service error message
6	Multiple MFA failures or login abuse	Fail authentication	Lock account/display security warning

3. Special Requirements
N/A
4. Preconditions
- User has an account (for registered users)
- User knows their login credentials
5. Postconditions
- On success: Session begins, permissions assigned
- On failure: User remains on login page with error displayed

Use Case: *Send Recommendations to City Providers (E-Mail)*

2. Descriptions
   This use case describes how the City Manager sends insights emails to transport city providers (Bus, Train, Bike, etc.) through a button that will be provided on the dashboard.

The goal is to enable city managers to share actionable demand insights such as high or low usage, demand imbalances, or forecasted surges directly with relevant providers to improve operational coordination.

Actors:

- City Manager: Reviews the data on the dashboard and clicks a button provided on the dashboard to send mails to the city providers.
- Transport Provider: Receives the e-mail and can take appropriate action.

Triggers and Inputs:

- Manually triggered when the City Manager clicks the button to send the e-mail on the dashboard.

2. Flow of Events

Basic Flow

User - City Manager		System	
1	Clicks the e-mail button on the dashboard based on the insights gauged from the dashboard.		
		2	Automatically compiles a message using predefined templates and retrieves the associated provider contact from the directory and sends the notification
		3	Displays a confirmation message and lo the action with timestamp, user id etc.

3. Special Requirements

- The system must have a working email or notification service integrated (Amazon SES etc).

4. Preconditions

- The City Manager is authenticated and authorized to send notifications.
- Demand data (historical and real-time) has been aggregated and is available in the dashboard.
- Email templates containing station/route details, metrics, and issue summary must be defined.

5. Postconditions

- Notification (email or message) is sent successfully to the appropriate transport provider.
- Confirmation is displayed to the City Manager.
- The system logs the action for record-keeping and traceability.

Use Case Name: *Display Live Pedestrian Data*

1. Description

This use case describes how the City Manager views live pedestrian traffic data on the dashboard.

The goal is to provide real-time visibility into pedestrian flow across the city, such as footfall at major intersections, tourist zones, or transport hubs, to support operational and planning decisions.

Actors:

- City Manager: Views and monitors pedestrian flow data.
- System: Fetches, processes, and visualizes live pedestrian sensor data from connected sources.

Triggers and Inputs:

Triggered when the City Manager navigates to the “Pedestrian Data” section of the dashboard.

2. Flow of Events

Basic Flow			
User - City Manager		System	
1	Navigates to “Live Pedestrian Data.”		
		2	Fetches real-time data from sensors/APIs and displays live footfall and density metrics.

Alternative Flow - Data Unavailable			
User - City Manager		System	
1	Navigates to “Live Pedestrian Data.”		
		2	Displays no data available at this time.

3. Special Requirements

- Active connection to live data APIs or IoT sensors.
- Automatic refresh or polling for near real-time updates.

4. Preconditions

- Live pedestrian data sources connected and operational.

5. Postconditions

- Dashboard displays current pedestrian activity by area.
- City Managers can analyze live movement patterns or receive “data unavailable” notice.