SC2006- SOFTWARE ENGINEERING

Lab 4 Deliverables

Lab Group- SCSA

Team- Apollo 11

Members:

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

Add headings (Format > Paragraph styles) and they will appear in your table of contents.

Use Case Model and Use Case Descriptions

Functional Requirement #1:Check Real-Time Congestion

Check Real-Time Congestion

Attribute	Details
Use Case Name	Check Real-Time Congestion
Created By	Ang Ming Yang
Last Updated By	Ang Ming Yang
Date Created	20th January
Date Last Updated	23rd February
Actor	Commuter

Description	Allows the user to check the congestion levels of MRT trains in real-time using the LTA DataMall API.

Preconditions	User has internet access and is logged in (if necessary).
Postconditions	Congestion data is displayed to the user.
Priority	High
Frequency of Use	High
Flow of Events	1. Users enter the starting and destination MRT stations. 2. System fetches real-time congestion data from the LTA API. 3. System displays congestion levels using a color-coded heatmap. 4. Users can either proceed with the journey or check alternative routes.

Alternative Flows	- If the API fails, the system displays cached historical congestion data - If the user enters an invalid station, the system prompts for re-entry.

Includes	Alternative route suggestions
Special Requirements	API must return data within 2 seconds.
Assumptions	User knows the MRT station names.

View Historical Congestion Trends

-	Details
Use Case Name	View Historical Congestion Trends
Created By	Agarwal Radhika
Last Updated By	Agarwal Radhika
Date Created	20th January
Date Last Updated	23rd February
Actor	Commuter
Description	Allows users to view past congestion trends to determine less crowded travel times.
Preconditions	User is logged in and selects a date range.

Postconditions	Historical congestion trends are displayed.
Priority	Medium
Frequency of Use	Medium
Flow of Events	1. User selects a past date range. 2. System retrieves historical congestion data from the database. 3. System displays congestion trends using graphs and heatmaps. 4. Users compare congestion levels across different times.
Alternative Flows	- If no historical data is available, the system notifies the user. - If the user enters an invalid date, the system prompts for re-entry.
Special Requirements	Data must be stored for at least 2 months.

Assumptions	Users understand congestion graphs.

Receive Congestion Notifications

Attribute	Details
Use Case Name	Receive Congestion Notifications
Created By	Gunda Sai Venkata Aaditya
Last Updated By	Gunda Sai Venkata Aaditya
Date Created	20th January
Date Last Updated	23rd February
Actor	Commuter
Description	Users can enable notifications for high congestion alerts.
Preconditions	Users enable congestion alerts in settings.

Postconditions	Notifications are sent when congestion exceeds the threshold.
Priority	Medium
Frequency of Use	Medium
Flow of Events	1. Users enable congestion notifications in settings. 2. User sets a congestion threshold (e.g., 80% train occupancy). 3. System monitors real-time congestion levels. 4. System sends a push notification when the congestion threshold is exceeded.
Alternative Flows	- If the user disables notifications, no alerts are sent. - If the network is down, the system retries sending alerts for 10 minutes before failing.
Includes	Customizable alert settings

Special Requirements	Push notifications must be delivered instantly.
Assumptions	Users have allowed notifications on their device.

Functional Requirement #2: Alternative Route Suggestions & Predictive Congestion Alerts

Find Alternative Routes

Attribute	Details
Use Case Name	Find Alternative Routes
Created By	Mohamed Yaseen Aboobucker Siddhique
Last Updated By	Mohamed Yaseen Aboobucker Siddhique
Date Created	20th January
Date Last Updated	23rd February
Actor	Commuter
Description	Suggests alternative MRT routes with lower congestion.

Preconditions	User is logged in and has searched for a train route.

Postconditions	Alternative route is displayed.
Priority	High
Frequency of Use	High
Flow of Events	1. User selects the “Find Alternative Route” option .2. System checks other MRT lines and congestion levels. 3. System displays less crowded alternative routes. 4. User selects the preferred alternative route.
Alternative Flows	- If no alternative route exists, the system suggests off-peak hours. - If the API fails, the system displays congestion estimates instead of real-time data.

Includes	Map view of suggested routes
Special Requirements	Congestion data must refresh every 5 minutes.

Assumptions	Users are willing to take alternative MRT routes.

Predictive Congestion Alerts

Attribute	Details
Use Case Name	Predictive Congestion Alerts
Created By	Titus Lua Jie Qing
Last Updated By	Titus Lua Jie Qing
Date Created	20th January
Date Last Updated	23rd February
Actor	Commuter
Description	Predicts train congestion levels using historical data and AI.
Preconditions	User has enabled congestion alerts.

Postconditions	System alerts the user before peak congestion occurs.
Priority	Medium
Frequency of Use	Medium
Flow of Events	1. System analyzes historical congestion trends. 2. System predicts congestion for upcoming hours. 3. If congestion is predicted to exceed 80%, the system sends an alert.
Alternative Flows	- If prediction fails, the system falls back to live congestion data. - If the user disables notifications, no alerts are sent.
Includes	AI-powered congestion forecasting
Special Requirements	System should predict congestion at least 30 minutes in advance.

Assumptions	Users will act based on congestion predictions.

Key Classes Description & Diagram

Class 1: User

Attribute	Type	Description
userID	String	Unique identifier for the user
username	String	User's name
email	String	User's email address
password	String	User's encrypted password
preferences	Object	Stores user preferences like alert settings

Method	Return Type	Description
viewCongestionData()	void	Displays real-time congestion levels

`setNotificationThreshold (int threshold)`	void	Allows user to set congestion alert threshold
`findAlternativeRoutes()`	List	Retrieves and suggests alternative routes
`viewHistoricalData (Date dateRange)`	List	Displays congestion trends for selected dates

Class 2: CongestionData

Attribute	Type	Description
`stationID`	String	Unique identifier for MRT stations
`congestionLevel`	int	Percentage of train occupancy (0-100%)
`timestamp`	DateTime	Time of congestion data retrieval

Method	Return Type	Description
`fetchRealTimeData()`	List	Retrieves congestion levels from LTA API
`analyzeTrends()`	Map<Date, int>	Analyzes congestion patterns over time
`predictCongestion()`	int	Uses AI/ML model to estimate future congestion

Class 3: AlternativeRoute

Attribute	Type	Description
`routeID`	String	Unique identifier for suggested routes
`stations`	List	List of MRT stations in the route
`travelTime`	int	Estimated travel time in minutes

`congestionScore`	int	Relative congestion level compared to default route

Method	Return Type	Description
`computeAlternativeRoutes()`	List	Suggests routes based on congestion and travel time
`getFastestRoute()`	AlternativeRoute	Returns the least congested and fastest route
`updateRouteData()`	void	Updates route suggestions based on new congestion data

Class 4: NotificationManager

Attribute	Type	Description

`notificationID`	String	Unique ID for each alert
`userID`	String	User receiving the notification
`threshold`	int	User-defined congestion limit
`status`	String	Notification status (sent/pending)

Method	**Return Type**	**Description**
`sendNotification` `(userID, message)`	void	Sends congestion alert notification
`scheduleNotification()`	void	Predicts and sends alerts based on ML model
`cancelNotification` `(notificationID)`	void	Cancels a scheduled alert

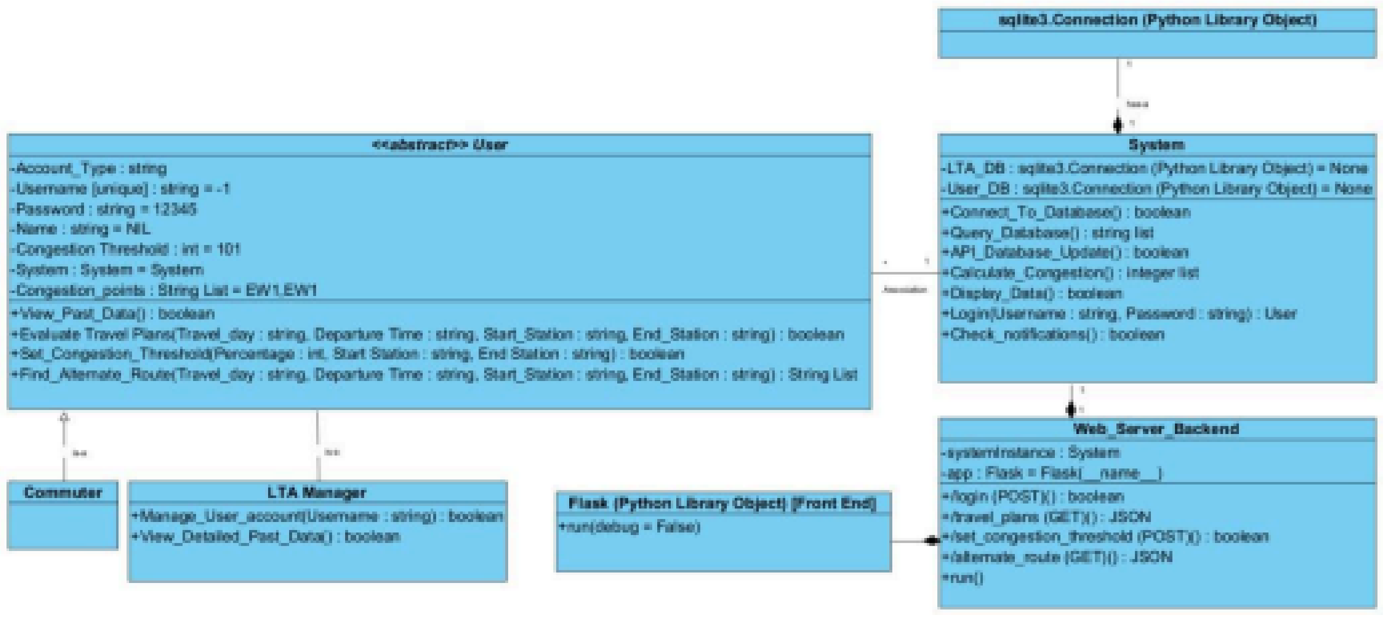
Class 5: ReportGenerator

Attribute	Type	Description
`reportID`	String	Unique identifier for each generated report
`dateGenerated`	Date	Timestamp of report generation
`reportType`	String	Type of report (daily, weekly, monthly)
`data`	List	Stores congestion and trend data

Method	Return Type	Description
`generateReport` `(reportType)`	String	Creates congestion reports for LTA/SMRT

`exportReport (format)`	File	Converts the report into CSV/PDF format
`retrievePastReports (dateRange)`	List	Fetches previously generated reports

Key Class Diagram


sqlite3.Connection (Python Library Object)
System
-LTA_DB : sqlite3.Connection (Python Library Object) = None
-User_DB : sqlite3.Connection (Python Library Object) = None
+Connect_To_Database() : boolean
+Query_Database() : string list
+API_Database_Update() : boolean
+Calculate_Congestion() : integer list
+Display_Data() : boolean
+Login(Username : string, Password : string) : User
+Check_notifications() : boolean
<<abstract>> User
-Account_Type : string
-Username [unique] : string = -1
-Password : string = 12345
-Name : string = NIL
-Congestion Threshold : int = 101
-System : System = System
-Congestion_points : String List = EW1,EW1
+View_Past_Data() : boolean
+Evaluate Travel Plans(Travel_day : string, Departure Time : string, Start_Station : string, End_Station : string) : boolean
+Set_Congestion_Threshold(Percentage : int, Start Station : string, End Station : string) : boolean
+Find_Alternate_Route(Travel_day : string, Departure Time : string, Start_Station : string, End_Station : string) : String List
Association
Commuter
LTA Manager
+Manage_User_account(Username : string) : boolean
+View_Detailed_Past_Data() : boolean
Flask (Python Library Object) [Front End]
+run(debug = False)
Web_Server_Backend
-systemInstance : System
-app : Flask = Flask(__name__)
+/login (POST)() : boolean
+/travel_plans (GET)() : JSON
+/set_congestion_threshold (POST)() : boolean
+/alternate_route (GET)() : JSON
+run()
Figure 1: Key Classes Diagram

Entity, Control, and Boundary Class Diagram & Description

Entity Classes

Class Name	Attributes	Methods	Description
User	userID: String username: String email: String password: String preferences: Object	viewCongestionData() setNotificationThreshold (t hreshold: int) findAlternativeRoutes () viewHistoricalData (dateRange: Date)	Stores user information and preferences for congestion tracking

CongestionData	stationID: String
	
	congestionLev el : int
	
	timestamp: DateTime
		fetchRealTimeData()
		
		analyzeTrends()
		
		predictCongest0ion()
			Stores congestion levels and analyzes trends

AlternativeRoute	routeID: String
	stations: List<String>
	travelTime: int
	congestionScore : int
		computeAlternativeRoutes() getFastestRoute()
		updateRouteData()
			Suggests less congested routes for users

ReportGenerator	reportID: String dateGenerated: Date reportType: String data: List<Congestion Data>	generateReport (reportType:String) exportReport (format:String) retrievePastReports (dateRange: Date)	Generates congestion reports for transport authorities

Control Classes

Class Name	Methods	Description
Congestion C ontroller	`getRealTimeCongestion (stationID: String): CongestionDatagetHistoricalData (dateRange: Date): List<CongestionData>`	Controls congestion tracking and data retrieval
RouteCont ro ller	`suggestAlternativeRoutes (source: String, destination: String): List<AlternativeRoute>`	Handles alternative route computation
Notification C ontroller	`sendAlert (userID: String, message: String) scheduleNotification (userID: String)`	Manages user notifications based on congestion levels

	cancelNotification (notificationID: String)	

ReportCo ntr oller	generateDailyReport()
	
	generateMonthlyReport()
	
	exportReport
	
	(reportID: String, format:String)
		Handles generation and retrieval of travel reports

Boundary Classes

Class Name	UI Elements / Methods	Description

User Interface	displayRealTimeCongestion
	
	()
	
	showAlternativeRoutes()
	
	showHistoricalTrends()
	
	enableNotifications()
		Provides the UI for congestion tracking, route suggestions, and alerts

Admin Dashboard	`viewCongestionReports()` `exportReport` `(format: String)`	UI for LTA/SMRT officials to monitor congestion trends
Notification Service	`sendPushNotification` `(userID: String, message: String)`	Delivers real-time congestion alerts to users

Entity, Control & Boundary Diagram

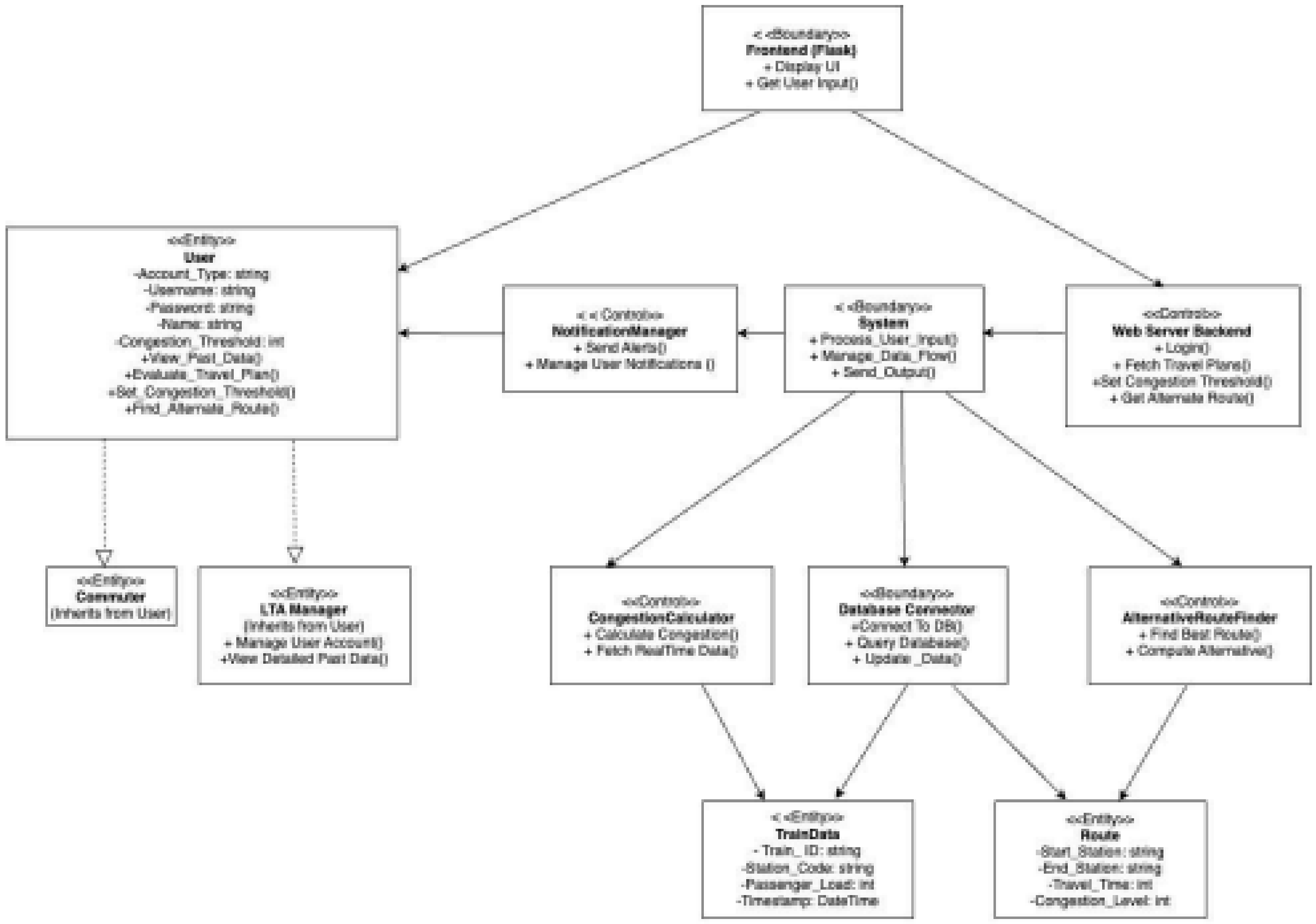


Figure 2: Entity, Control & Boundary Diagram

Sequence Diagrams of Use Cases

Checking Real-Time Congestion

Use Case: Check Real-Time Congestion

Component	**Action**
Commuter (User)	Requests real-time congestion for a specific MRT route
Frontend (Flask UI)	Sends congestion request to Web Server Backend

Web Server Backend	Calls `FetchRealTimeData()` from Database Connector
Database Connector & LTA API	Retrieves congestion data from the API
Database Connector & LTA API	Returns congestion data (Train Load, Status)
Web Server Backend	Sends processed congestion data to Frontend
Frontend (Flask UI)	Displays real-time congestion data (graphs, heatmaps)
Alternative Flow	If API data is not found, system fetches historical data
Alternative Flow	If no API and historical data exist, system displays error message

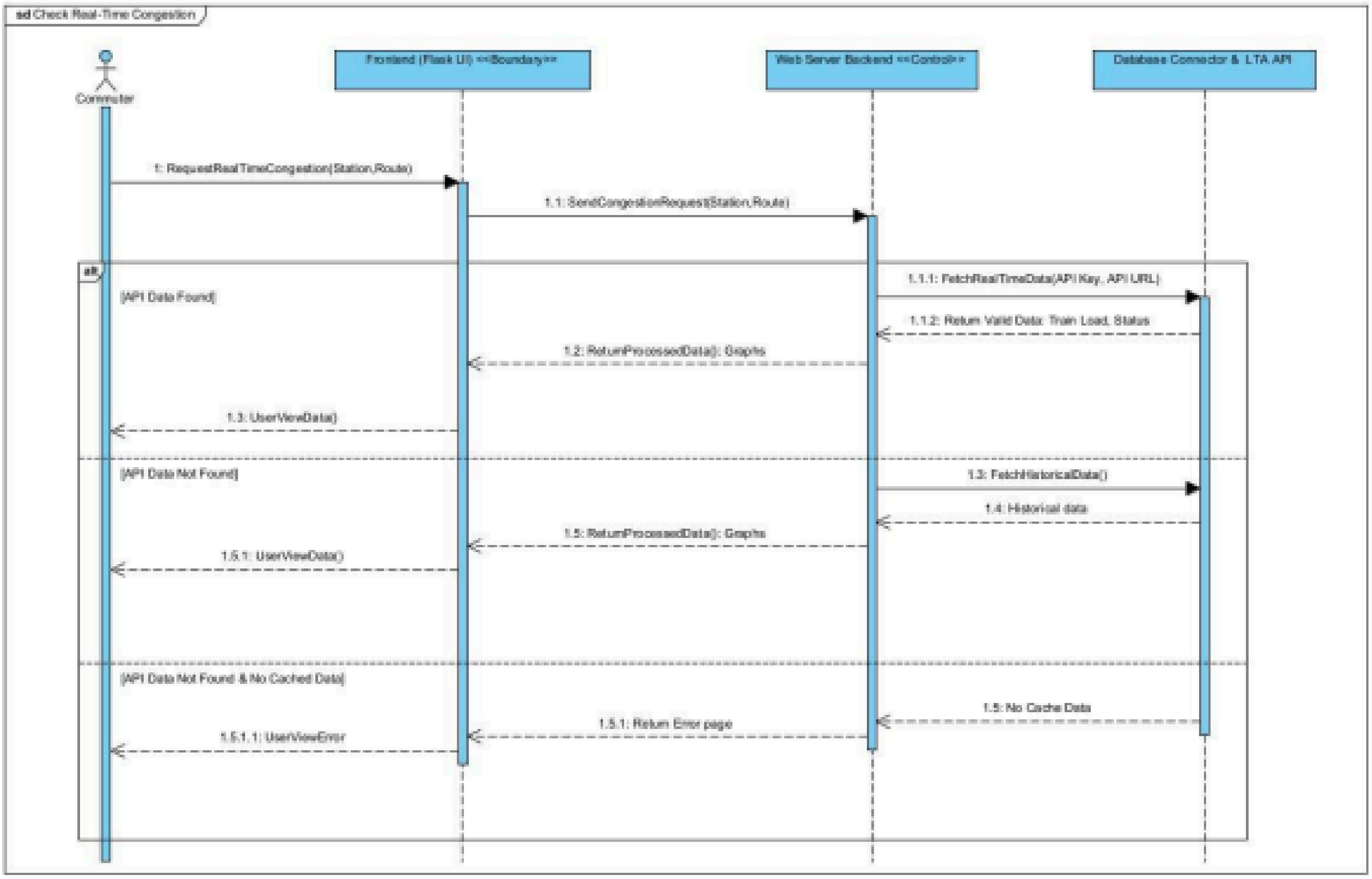

sd Check Real-Time Congestion
Commuter
Frontend (Flask UI) <<Boundary>>
Web Server Backend <<Control>>
Database Connector & LTA API
1: RequestRealTimeCongestion(Station,Route)
1.1: SendCongestionRequest(Station,Route)
alt
[API Data Found]
1.1.1: FetchRealTimeData(API Key, API URL)
1.1.2: Return Valid Data: Train Load, Status
1.2: ReturnProcessedData(): Graphs
1.3: UserViewData()
[API Data Not Found]
1.3: FetchHistoricalData()
1.4: Historical data
1.5: ReturnProcessedData(): Graphs
1.5.1: UserViewData()
[API Data Not Found & No Cached Data]
1.5: No Cache Data
1.5.1: Return Error page
1.5.1.1: UserViewError

Figure 3: Check Real-Time Congestion

Alternative Route Suggestions & Predictive Congestion Alerts

Use Case: Alternative Route Suggestions with Train Congestion

Component	**Action**
Commuter (User)	Enters alternative route request (Start, End, Date, Time)
Frontend (Flask UI)	Sends route request to Web Server Backend
Web Server Backend	Calls `FetchRouteData(Start, End, Date, Day)` from Database Connector

Database Connector & LTA API	Retrieves alternative route options based on congestion
Database Connector & LTA API	Returns list of alternative routes
Web Server Backend	Sends suggested routes to Frontend
Frontend (Flask UI)	Displays list of alternative routes to the user
Alternative Flow	If no alternative route is found, system displays error (Route Error 104)

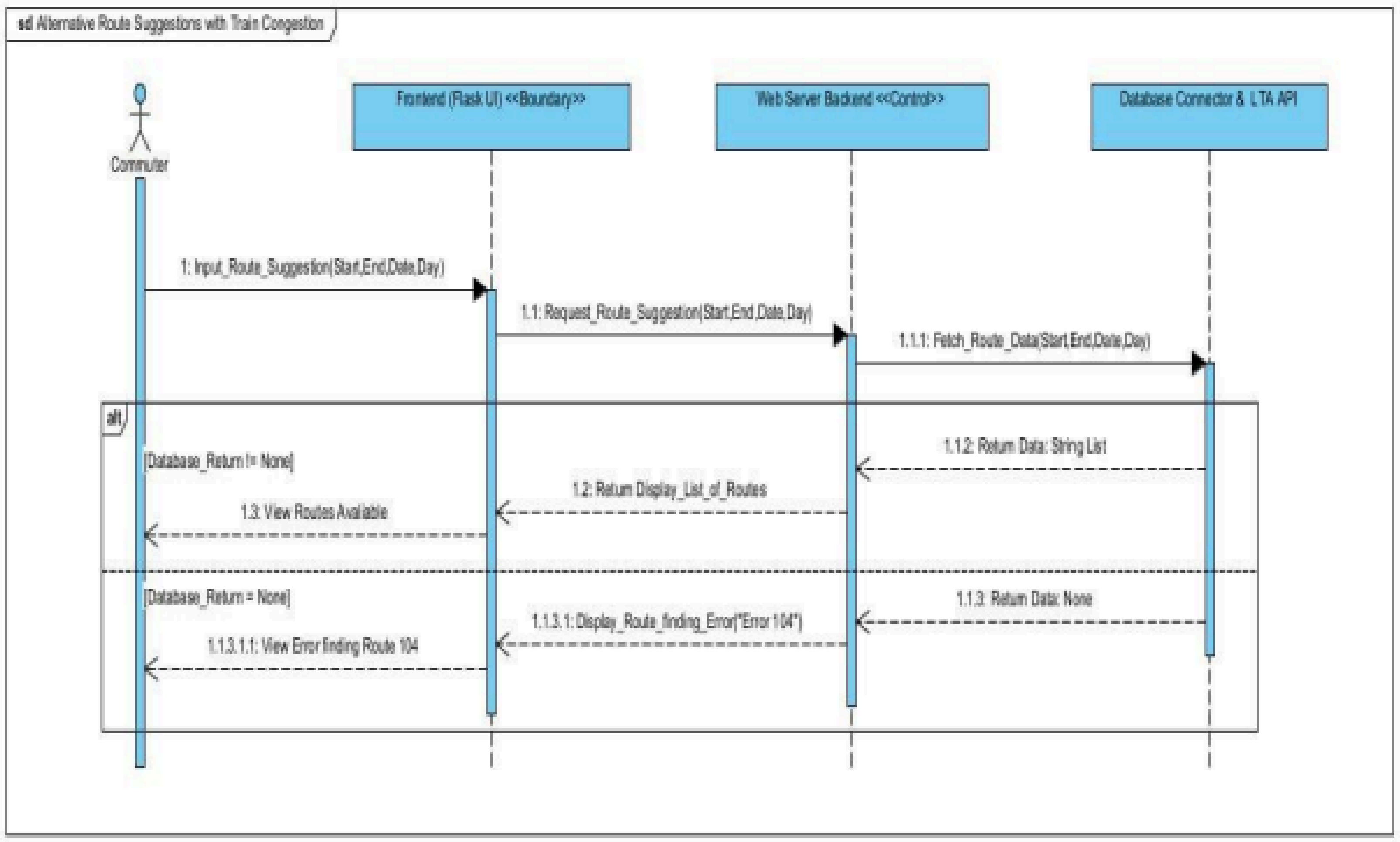

sd Alternative Route Suggestions with Train Congestion
Commuter
Frontend (Flask UI) <<Boundary>>
Web Server Backend <<Control>>
Database Connector & LTA API
1: Input_Route_Suggestion(Start,End,Date,Day)
1.1: Request_Route_Suggestion(Start,End,Date,Day)
1.1.1: Fetch_Route_Data(Start,End,Date,Day)
alt
[Database_Return != None]
1.1.2: Return Data: String List
1.2: Return Display_List_of_Routes
1.3: View Routes Available
[Database_Return = None]
1.1.3: Return Data: None
1.1.3.1: Display_Route_finding_Error("Error 104")
1.1.3.1.1: View Error Finding Route 104

Figure 4: Alternative Route Suggestions with Train Congestion

Receiving Congestion Notifications

Use Case: Receive Congestion Notifications

Component	**Action**
Commuter (User)	Inputs congestion threshold to receive alerts
Frontend (Flask UI)	Sends threshold update request to Web Server Backend
Web Server Backend	Calls `UpdateDatabase(Threshold, Username)`
Database Connector & LTA API	Updates congestion threshold value in DB
Database Connector & LTA API	Returns confirmation of valid update
Web Server Backend	Sends success message to Frontend
Frontend (Flask UI)	Displays "Congestion Threshold Updated" message
Alternative Flow	If database update fails, system displays error (Threshold Error 103)

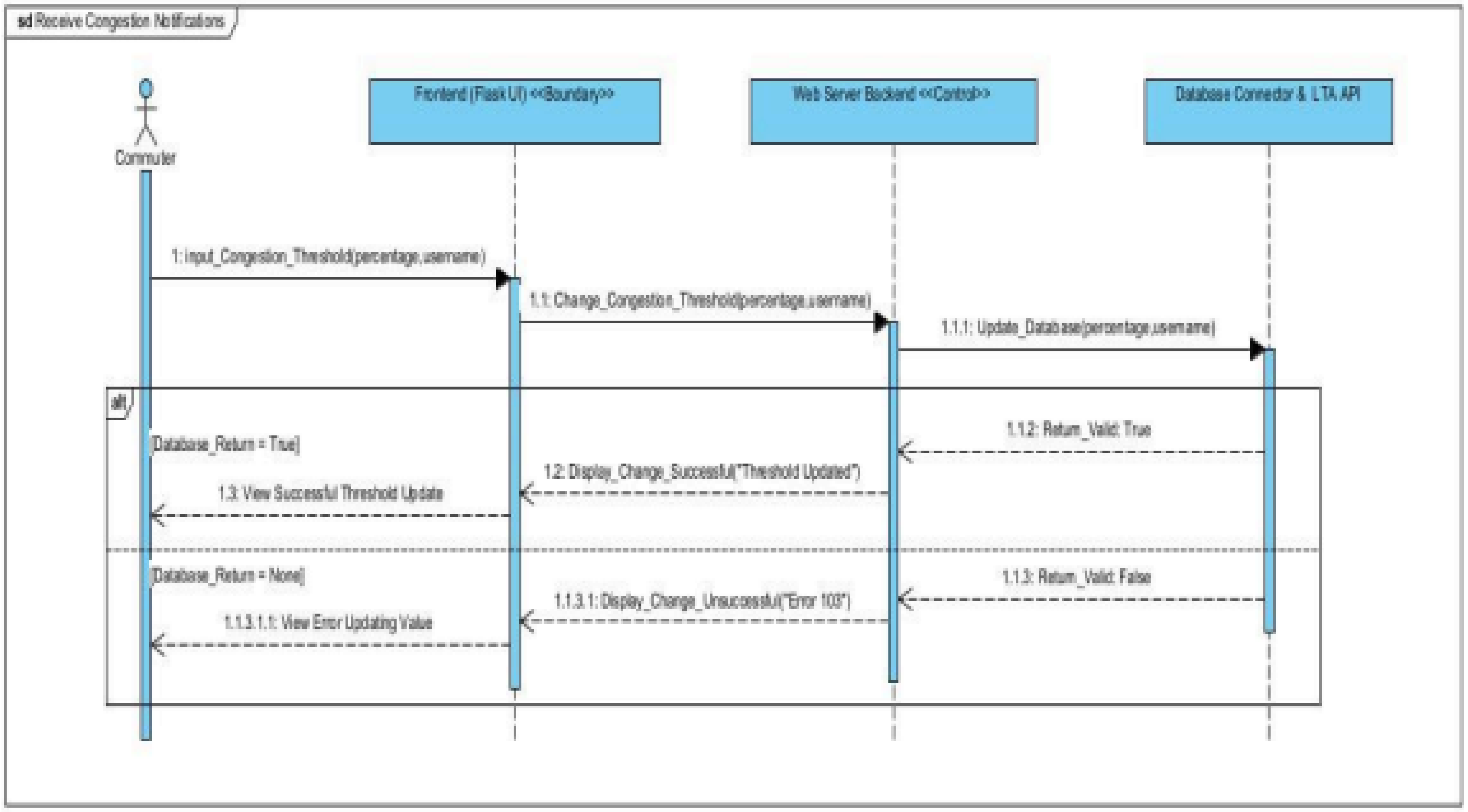

sd Receive Congestion Notifications
Commuter
Frontend (Flask UI) <<Boundary>>
Web Server Backend <<Control>>
Database Connector & LTA API
1: Input_Congestion_Threshold(percentage,username)
1.1: Change_Congestion_Threshold(percentage,username)
1.1.1: Update_Database(percentage,username)
alt
[Database_Return = True]
1.1.2: Return_Valid: True
1.2: Display_Change_Successful("Threshold Updated")
1.3: View Successful Threshold Update
[Database_Return = None]
1.1.3: Return_Valid: False
1.1.3.1: Display_Change_Unsuccessful("Error 103")
1.1.3.1.1: View Error Updating Value

Figure 5: Receive Congestion Notifications

Generating Travel Reports for LTA/SMRT

Use Case: Generate Travel Report (LTA/SMRT)

Component	Action
LTA Manager (User)	Requests a congestion report
Frontend (Flask UI)	Sends travel report request to Web Server Backend
Web Server Backend	Calls `Fetch_Report_Data()` from Database Connector
Database Connector & LTA API	Retrieves historical congestion and train data
Database Connector & LTA API	Returns processed congestion report data

Web Server Backend	Sends processed report to Frontend
Frontend (Flask UI)	Displays travel report to LTA Manager
Alternative Flow	If insufficient data is available, system displays report generation error (Error 105.1)

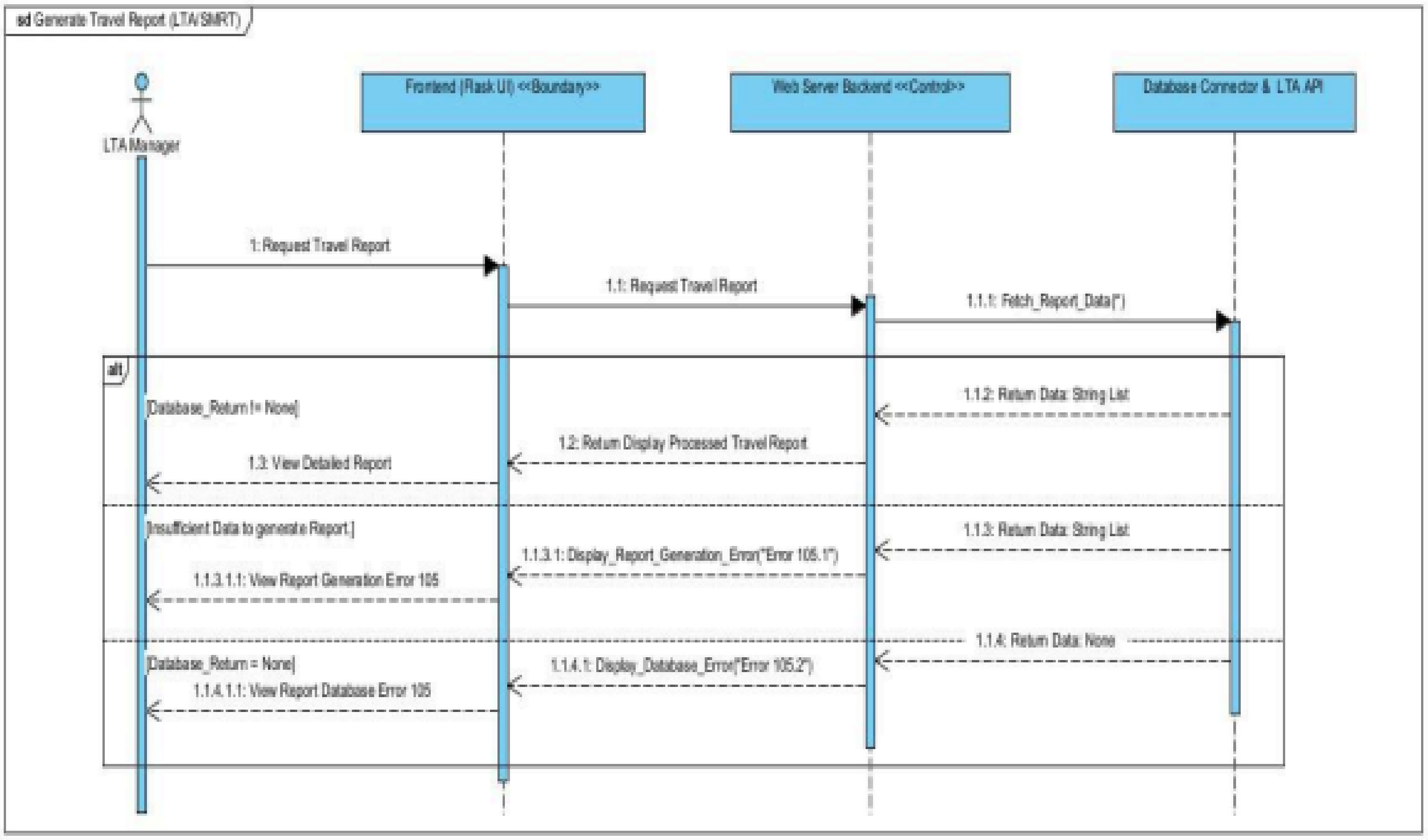

sd Generate Travel Report (LTA/SMRT)
LTA Manager
Frontend (Flask UI) <<Boundary>>
Web Server Backend <<Control>>
Database Connector & LTA API
1: Request Travel Report
1.1: Request Travel Report
1.1.1: Fetch_Report_Data(*)
alt
[Database_Return != None]
1.1.2: Return Data: String List
1.2: Return Display Processed Travel Report
1.3: View Detailed Report
[Insufficient Data to generate Report.]
1.1.3: Return Data: String List
1.1.3.1: Display_Report_Generation_Error("Error 105.1")
1.1.3.1.1: View Report Generation Error 105
1.1.4: Return Data: None
[Database_Return = None]
1.1.4.1: Display_Database_Error("Error 105.2")
1.1.4.1.1: View Report Database Error 105

Figure 6: Generate Travel Report (LTA/SMRT)

Generating Recommended Train Scheduling for LTA/SMRT

Use Case: Generate Recommended Train Scheduling (LTA/SMRT)

Component	Action
LTA Manager (User)	Requests recommended train scheduling

Frontend (Flask UI)	Sends scheduling request to Web Server Backend
Web Server Backend	Calls `Fetch_Train_Data()` from Database Connector
Database Connector & LTA API	Retrieves scheduling information based on congestion patterns
Database Connector & LTA API	Returns optimized train schedule recommendations
Web Server Backend	Sends schedule recommendations to Frontend
Frontend (Flask UI)	Displays train scheduling recommendations
Alternative Flow	If insufficient data exists, system displays scheduling error (Error 106.1)

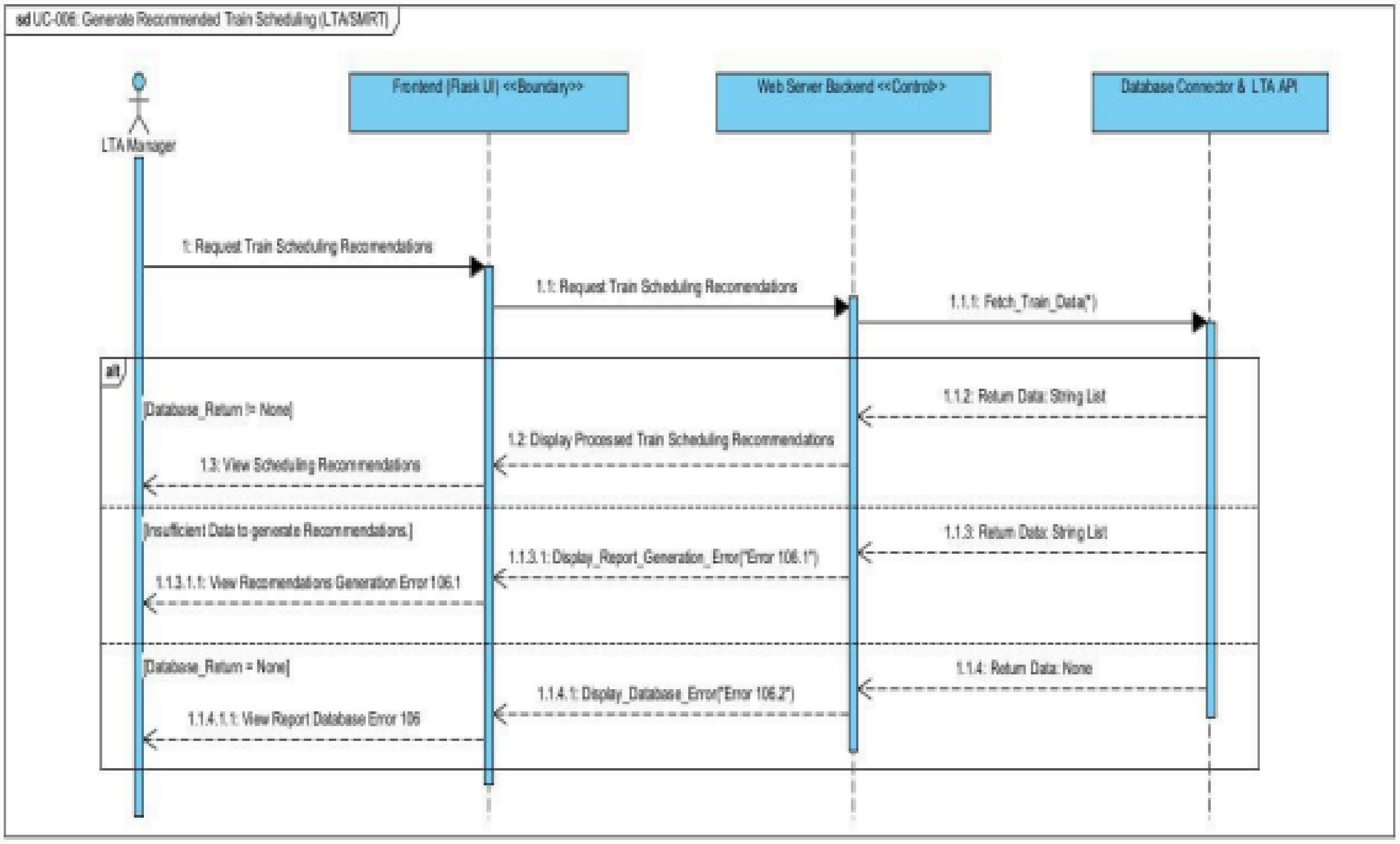

sd UC-006: Generate Recommended Train Scheduling (LTA/SMRT)
LTA Manager
Frontend (Flask UI) <<Boundary>>
Web Server Backend <<Control>>
Database Connector & LTA API
1: Request Train Scheduling Recomendations
1.1: Request Train Scheduling Reconendations
1.1.1: Fetch_Train_Data(*)
alt
[Database_Return != None]
1.1.2: Return Data: String List
1.2: Display Processed Train Scheduling Recommendations
1.3: View Scheduling Recommendations
[Insufficient Data to generate Recommendations.]
1.1.3: Return Data: String List
1.1.3.1: Display_Report_Generation_Error("Error 106.1")
1.1.3.1.1: View Recomendations Generation Error 106.1
[Database_Return = None]
1.1.4: Return Data: None
1.1.4.1: Display_Database_Error("Error 106.2")
1.1.4.1.1: View Report Database Error 106

Figure 7: Generate Recommended Train Scheduling (LTA/SMRT)

User Login System

Use Case: User Login Process

Component	Action
Commuter (User)	Enters login credentials (Username, Password)
Frontend (Flask UI)	Sends `sendLoginRequest(Username, Password)` to Web Server Backend
Web Server Backend	Calls `queryUserCredentials(Username)` in Database
Database	Authenticates user and returns login status

Web Server Backend	Sends login status (success/fail) to Frontend
	ask UI) Displays login success or failure message

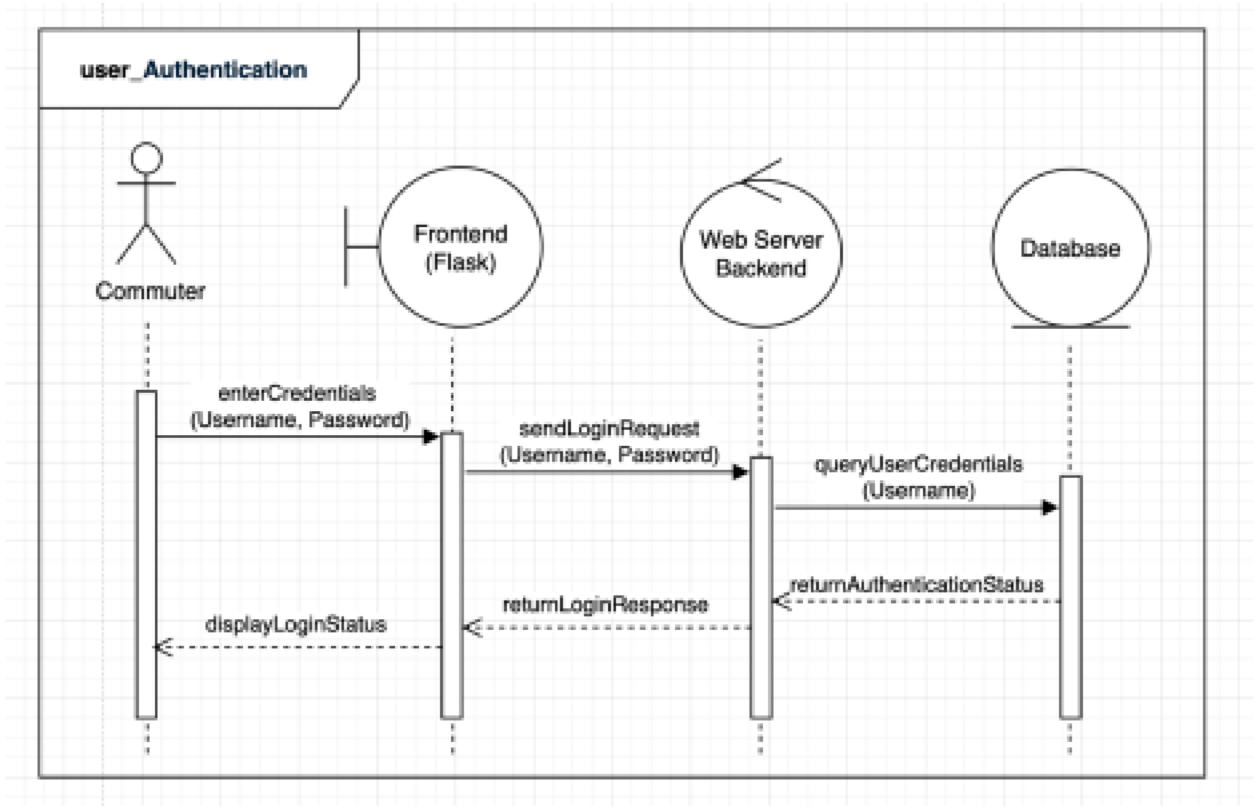

user_Authentication
Commuter
Frontend (Flask)
Web Server Backend
Database
enterCredentials (Username, Password)
sendLoginRequest (Username, Password)
queryUserCredentials (Username)
returnAuthenticationStatus
returnLoginResponse
displayLoginStatus

Figure 8: User Authentication

Initial Dialog Flow & Map

Commuter Dialog Flow

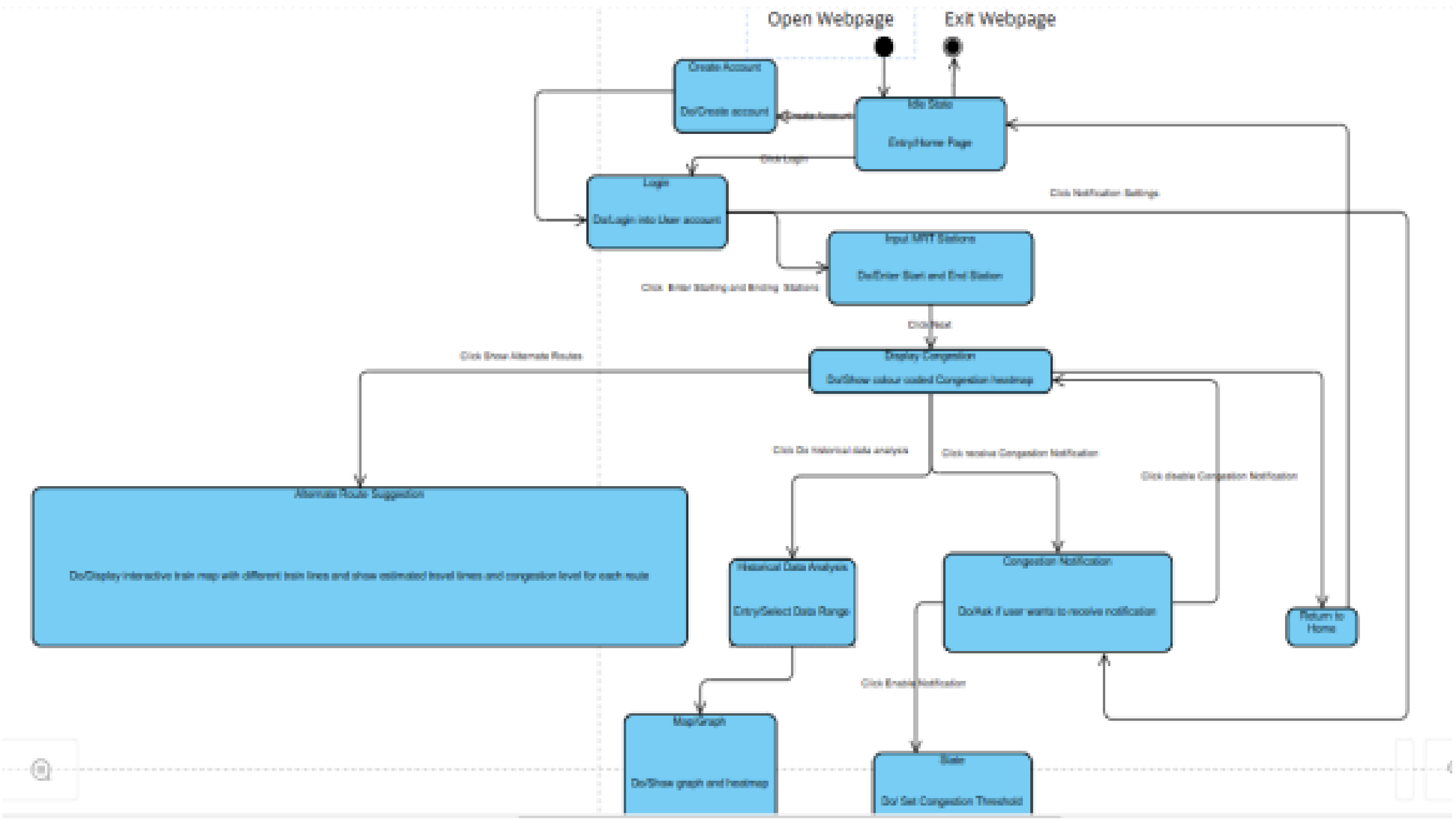
Step	User Action	System Response
1	Open Webpage	System displays the Entry/Home Page
2	Click Create Account	System opens account creation form

3	Click Login	System authenticates user and navigates to home dashboard
4	Click Enter MRT Stations	System prompts user to input starting and ending stations
5	Click Next	System displays congestion data in a color-coded heatmap
6	Click Show Alternate Routes	System shows alternative train routes with congestion levels and estimated travel times
7	Click Do Historical Data Analysis	System prompts user to select data range
8	Select Date Range	System generates a graph and heatmap showing congestion trends
9	Click Receive Congestion Notification	System asks if user wants to enable notifications
10	Click Enable Notification	System sets user congestion threshold and enables alerts
11	Click Disable Congestion Notification	System turns off congestion alerts
12	Click Return to Home	System redirects user to the main home page
13	Click Exit Webpage	System logs user out and closes session

LTA Manager Dialog Flow

Step	User Action	System Response
1	Login to Admin Portal	System authenticates LTA Manager and redirects to admin dashboard
2	Click Generate Travel Report	System requests report data from Database Connector
3	Select Date Range for Report	System fetches historical congestion data
4	Click View Travel Report	System generates and displays travel congestion trends
5	Click Export Report	System provides download options in CSV or PDF format
6	Click Generate Recommended Train Schedule	System analyzes past congestion data and suggests optimized train schedules
7	Click Return to Home	System redirects to admin dashboard
8	Click Exit Webpage	System logs out and ends admin session

Dialog Map


Open Webpage
Exit Webpage
Create Account
Do/Create account
Create Account
Idle State
Entry/Home Page
Click Login
Login
Do/Login into User account
Click Notification Settings
Input MRT Stations
Do/Enter Start and End Station
Click Enter Starting and Ending Stations
Click Next
Click Show Alternate Routes
Display Congestion
Do/Show colour coded Congestion heatmap
Click Do historical data analysis
Click receive Congestion Notification
Click disable Congestion Notification
Alternate Route Suggestion
Do/Display interactive train map with different train lines and show estimated travel times and congestion level for each route
Historical Data Analysis
Entry/Select Data Range
Congestion Notification
Do/Ask if user wants to receive notification
Return to Home
Click Enable Notification
Map/Graph
Do/Show graph and heatmap
State
Do/ Set Congestion Threshold