Try using this API for traffic stats:

https://www.tomtom.com/traffic-index/singapore-traffic/

https://move.tomtom.com/dashboard
Login details:

Username: kumar046@mymail.sim.edu.sg
PW: 1234567SJ

Hard code 1 super user for the govt that has the admin control for the other govt acc

Requirements for each actor:

Traffic Analyst:

- View Historical Trends ('Average Traffic Density', 'Road name', 'Month', 'Average/Annual Hotspots' for maybe the past 3 years) TA08

Name: View Historical Trends	**ID: TA-08**
Stakeholders and goals: Identify recurring congestion patterns for planning.	
Description: Plots daily, weekly, monthly, yearly average congestion levels.	
Actors: Traffic Analyst	
Trigger: User selects "Historical Trends."	
Normal Flow: 1. Analyst chooses time range. 2. System aggregates archived data. 3. Graphs & tables displayed for comparison.	
Sub-Flows: Drill-down per region	
Alternative/Exceptional Flows: No data → prompt to upload historical dataset	

- Detect Anomalies TA09 (define the anomalies: incidents on roads [road debri e.g. fallen tree, sinkhole/pothole, car accidents etc])

Name: Detect Anomalies	**ID: TA-09**
Stakeholders and goals: Flag unusual traffic behaviour for further investigation.	
Description: Identifies sudden deviations in traffic flow metrics.	
Actors: Traffic Analyst	
Trigger: Automatic after model run	
Normal Flow: 1. System applies statistical thresholds. 2. Outliers flagged for review. 3. Analyst examines and confirms findings.	
Sub-Flows: None	
Alternative/Exceptional Flows: All values normal → no alerts generated.	

- Schedule Automated Runs TA10

Name: Schedule Automated Runs	**ID: TA-10**
Stakeholders and goals: Automate routine daily analysis.	
Description: Schedules model execution at pre-defined intervals.	
Actors: Traffic Analyst	
Trigger: User sets scheduler time.	
Normal Flow: 1. Analyst opens scheduler 2. Selects dataset and frequency. 3. System adds job to cron 4. Job executes as planned.	
Sub-Flows: Automatic email notification on completion.	
Alternative/Exceptional Flows: Invalid time input → system rejects entry.	

- Filter the dataset by Region (focus analysis on specific areas - North, South, East, West) TA11 (try to implement this and if cant find relevant api for this feature, you can just ignore this then)

Name: Filter Dataset by Region	**ID: TA-11**
Stakeholders and goals: Focus analysis on specific areas	
Description: Applies regional filters before running models.	
Actors: Traffic Analyst	
Trigger: User chooses region dropdown.	
Normal Flow: 1. Analyst selects region. 2. System loads subset data. 3. Models applies to filtered data.	
Sub-Flows: None	
Alternative/Exceptional Flows: Region not found → prompt to upload regional map.	

Govt:

- Generate Historical Trends report (can be same as View Historical Trends from TA08) **:**
- Area analysis API from TomTom

Name: Generate Historical Congestion Report	**ID: GA-05**
Stakeholders and goals: Produce routine summaries for departmental review.	
Description: Compiles congestion indices, average delays, and hotspots.	
Actors: Government agencies	
Trigger: User selects “Time-based Report” button	

Normal Flow: 1. Officer sets date range. 2. System compiles data and generates graph//table for viewing. 3. Notification sent
Sub-Flows: None
Alternative/Exceptional Flows: Missing data → partial report with warning

- Run simulation for future event **: (You can test around to see if this feature is doable, and whether it can be implemented. If not, you can ignore this feature)**

Name: Simulate Event Scenario	**ID: GA-06**
Stakeholders and goals: Predict congestion impact before an upcoming event.	
Description: Runs model with temporary road restrictions or diversions.	
Actors: Government agencies	
Trigger: User chooses "Simulate Scenario" button	
Normal Flow: 1. Officer inputs event details (Number of people expected to attend, Area/s affected by event) 2. System runs simulation. 3. Visual forecast displayed. (Surrounding roads to be congested highlighted in orange,med congestion, or red, high congestion) 4. Officer can choose to save the results in pdf form.	
Sub-Flows: None	
Alternative/Exceptional Flows: Insufficient data → simulation cancelled	

- Simple Overlay of weather conditions over map (https://data.gov.sg/collections/1459/view)

Name: Overlay Weather Data	**ID: GA-07**
Stakeholders and goals: Assess environmental impact on traffic	
Description: Merges rainfall / haze data with traffic maps	

Actors: Government agencies
Trigger: User presses "Weather Overlay" button
Normal Flow: 1. User presses "Weather Overlay" button 2. System retrieves weather feed and matches data to time intervals. 3. Map is overlaid with color gradient showing weather conditions and corresponding traffic congestion (On top of current map, overlap with a heat map ranging from blue to red, can change colour scheme if need, to show varying weather conditions)
Sub-Flows: Legend for heatmap auto-generated
Alternative/Exceptional Flows: API offline → overlay skipped

- Cross refer to public transport routes (API available in LTA mall website provided in previous document)

Name: Cross-Reference Public Transport Data	**ID: GA-08**
Stakeholders and goals: Coordinate road and transit planning	
Description: Correlates congestion hotspots with bus/MRT routes	
Actors: Government agencies	
Trigger: Officer selects "Transport Overlay" button	
Normal Flow: 1. User requests transport overlay. 2. System retrieves congestion data. 3. System retrieves transit route data. 4. System composes overlay. 5. System displays overlay. (So basically highlighted routes of buses with slightly less opacity so we can see if congestion on particular roads is linked to the congestion to the presence or lack of buses)	
Sub-Flows: None	
Alternative/Exceptional Flows: None	

- Manage User account control (Only available for sub-user group Government Admin, which is still under Govt User) (sidenote: if this is overly complicated to implement, please focus on the other features first then.)
- **Hard code (fixed/default username and password) 1 super user for the govt that has the admin control for the other govt acc (for demo purposes to the assessor)**

Name: Manage User Accounts	**ID: GA-10**
Stakeholders and goals: Ensure only authorized personnel access data	
Description: Add, edit, or deactivate government user accounts	
Actors: Administrator (Govt)	
Trigger: Admin selects "User Management" button	
Normal Flow: 1. Admin open user management. 2. System loads user list. 3. Admin performs an action. (Admin can see list of active user accounts and choose to deactivate/suspend/activate)	
Sub-Flows: None	
Alternative/Exceptional Flows: Duplicate email → reject entry.	

Public:

- Filter congestion views based on Region (Divide the map between North, South, East, West, and Central regions, Map only displays data of that region)
- Apply this to both the Live data view and the Prediction data view (try to implement this and if cant find relevant api for this feature, you can just ignore this then)

Name: Filter by Region	
Stakeholders and goals: User wants to focus on specific districts.	

Description: Allows filtering congestion view by region.
Actors: Public User
Trigger: User selects region filter.
Normal Flow: 1. Map has a hover trigger for the different regions (Mouse cursor hover over region to filter it) 2. User selects a region. 3. System applies region filter. 4. Map refreshes with filtered view. (Only shows the data of that region, applies to both Live and Prediction data for the map)
Sub-Flows: None.
Alternative/Exceptional Flows: No roads in region → display empty state.

- Toggle between live congestion data and prediction data (~30 mins later) for the main map, Just a different view of the main map
- (try to implement this and if cant find relevant api for this feature, you can just ignore this then)

Name: Toggle Map Modes	
Stakeholders and goals: User prefers different visual views (live vs predictive).	
Description: Switches between real-time and forecasted congestion views.	
Actors: Public User	
Trigger: User toggles view button.	
Normal Flow: 1. System opens map in current mode. (Default) 2. User toggles to predictive mode. 3. System checks forecast availability. (Runs prediction algorithms) 4. Map re-renders in predictive view.	
Sub-Flows: None	

Alternative/Exceptional Flows: Forecast data unavailable → remain in live mode.

System Developer:

2: View Algo Mods

Name: View Algorithm Modules	
Stakeholders and goals: System Developers - want to view all available algorithm modules so they can understand, maintain, or enhance the system.	
Description: View a list of all algorithm modules used by this system.	
Actors: System Developers	
Trigger: The System Developer clicks the "Algorithm" button on the Admin Dashboard Page.	
Normal Flow: 1. The System Developer clicks on the **"**Algorithm**"** button. 2. The system loads the Admin Dashboard Page. 3. The system checks what algorithm modules are available. 4. The system retrieves the list of algorithm modules from its database. 5. The system returns the list to the dashboard. 6. The System Developer sees all available algorithm modules displayed on the page.	
Sub-Flows: None	
Alternative/Exceptional Flows: No algorithm modules found → returns null	

3: Suspend Algo Mods

Name: Suspend Algorithm Modules	**ID: SD-03**
Stakeholders and goals: System Developers - want to temporarily disable an algorithm module so it will not be used by the system.	
Description: Suspend an algorithm module of choice without entirely deleting its data.	
Actors: System Developers	
Trigger: The System Developer selects an algorithm and clicks the "Suspend" button.	
Normal Flow: 1. The System Developer chooses which algorithm they want to suspend.	

2. The developer clicks the “Suspend” button. 3. The system processes the suspension request. 4. The system updates the chosen algorithm so it becomes suspended. 5. The Admin Dashboard Page reloads. 6. The System Developer sees the updated page showing the algorithm has been suspended.
Sub-Flows: None
Alternative/Exceptional Flows: Suspension fails → returns null and error message

5: Optimize System Perf (try to implement this; if too complicated, can just move onto others first)

Name: Perform Routine Maintenance	**ID: SD07**
Stakeholders and goals: System Developers - want to keep the system healthy by performing routine maintenance tasks such as log cleanup, backups, and system updates.	
Description: Carry out regular system maintenance activities. Includes cleaning old logs, creating backups, and performing server or system updates.	
Actors: System Developers	
Trigger: The System Developer opens the Maintenance Dashboard to perform scheduled or manual system maintenance.	
Normal Flow: 1. The System Developer accesses the maintenance dashboard. 2. The system displays available maintenance options (log cleanup, backup, update). 3. The developer chooses to run the maintenance tasks. 4. The system cleans up old logs based on retention settings. 5. The system creates a system backup. 6. The system applies any required system or server updates. 7. After completing all tasks, the system returns a success result. 8. The developer sees the maintenance result displayed on the page.	
Sub-Flows: Maintenance steps executed separately	
Alternative/Exceptional Flows: Maintenance task fails (e.g., disk full, backup error, update failure) → generate failure report & return error message	

8: Monitor Sys Logs (try to implement this; if too complicated, can just move onto others first)

Name: Monitor System Logs	**ID: SD09**
Stakeholders and goals: System Developers - want to monitor system logs to check overall system health, identify anomalies, and resolve flagged issues.	
Description: Access and view system logs, check summaries of log activity, identify flagged or unusual events, and resolve any issues detected.	
Actors: System Developers	
Trigger:	
Normal Flow: 1. The System Developer accesses the system logs dashboard. 2. The system retrieves recent log entries and generates a log summary. 3. The system displays the overall system health and activity summary to the developer. 4. The developer views any flagged or unusual log entries. 5. The developer selects a flagged issue and chooses to resolve it. 6. The system applies the corrective action and marks the issue as resolved. 7. The system shows the resolution status to the developer.	
Sub-Flows: Regenerating corrupted logs	
Alternative/Exceptional Flows: Issue resolution fails → generate error report & return error message Log corruption detected → attempts to repair/regenerate the logs, if fail, returns error	

10: Create Access Perms

Name: Create Access Permission	**ID: SD-12**
Stakeholders and goals: System Developers - want to create new access permissions so that users or system components have the correct authorization levels to perform actions securely.	
Description: Create new access permissions within the system for the different user groups (e.g., Traffic Analyst, Government/Transport Authorities, General Public).	
Actors: System Developers	
Trigger: The System Developer opens the Create Permission Form from the permissions page.	
Normal Flow: 1. The System Developer opens the permission creation form. 2. The system displays the permission form and matrix of available rules. 3. The developer fills in the new permission details.	

4. The developer submits the permission form. 5. The system validates the permission syntax and rules. 6. The system saves the new permission into the database. 7. A confirmation message is shown indicating the permission was created successfully.
Sub-Flows: None
Alternative/Exceptional Flows: Permission creation fails → returns error message

11: Update Access Perms

Name: Update Access Permission	
Stakeholders and goals: System Developers - want to modify existing access permissions so users or system components maintain correct and secure authorization levels.	
Description: Update or modify existing access permissions.	
Actors: System Developers	
Trigger: The System Developer selects an existing permission and chooses to update it.	
Normal Flow: 1. The System Developer opens the permissions page. 2. The system loads and displays the current permission matrix. 3. The developer selects a permission to update. 4. The developer edits the permission rules and submits the update. 5. The system validates the updated permission syntax. 6. The system saves the updated permission to the database. 7. The system confirms the update was successful. 8. The developer sees the update success message.	
Sub-Flows: None	
Alternative/Exceptional Flows: Invalid permission syntax → return error message & update not applied.	

12: Suspend Access Perms

Name: Suspend Access Permission	
Stakeholders and goals: System Developers - want to temporarily disable an access permission to prevent unauthorized or risky actions, or to comply with administrative/security requirements.	

Description: Suspend an existing access permission.
Actors: System Developers
Trigger: The System Developer selects an active permission and chooses the “Suspend Permission” action.
Normal Flow: 1. The System Developer opens the permissions page. 2. The system retrieves and displays the list of active permissions. 3. The developer selects a permission to suspend. 4. The developer enters a suspension reason and submits the request. 5. The system updates the permission status to suspended. 6. The system creates an audit log stating that the permission was suspended. 7. The system confirms that the suspension was successful. 8. The developer sees the suspension success message.
Sub-Flows: None
Alternative/Exceptional Flows: Suspension fails → returns failure response and message

13: Backup & Restore Data

- You can test and see whether this feature is able to be implemented or not; if not, you can ignore this feature.

Name: Backup and Restore Data	**ID: SD-16**
Stakeholders and goals: System Developers - want to securely back up system data and restore it when needed to ensure data integrity, availability, and recovery readiness.	
Description: Perform a full system data backup and restore previous backups when required.	
Actors: System Developers	
Trigger: The System Developer selects “Backup & Restore Data” from the maintenance dashboard.	
Normal Flow: 1. The System Developer opens the Backup & Restore page. 2. The system shows two options: Backup or Restore. 3. The developer clicks on Backup. 4. The system starts collecting all the data that needs to be saved. 5. The system finishes gathering the data and prepares it for backup. 6. The system safely packages and stores the backup file. 7. The system records that a new backup has been created.	

8. The system confirms that the backup was saved successfully. 9. The system sends a simple confirmation message to the developer. 10. The system displays “Backup completed successfully.”
Sub-Flows: Restore data → select “Restore” and choose backup ID, controller requests restoration from entity, entity restores the data from the selected backup, returns success message.
Alternative/Exceptional Flows: Backup/Restore failure → returns failure message

15: View Feedback

Name: View Feedback	**ID: SD-18**
Stakeholders and goals: System Developers - want to review all user feedback to understand issues, improvements, and overall system experience.	
Description: View all feedback submitted by public users (registered or not).	
Actors: System Developers	
Trigger: The System Developer opens the Feedback Dashboard from the system menu.	
Normal Flow: 1. The System Developer opens the Feedback Dashboard page. 2. The system requests all feedback records from the database. 3. The system retrieves the list of feedback entries. 4. The system displays the full list of feedback to the developer. 5. The developer reviews the displayed feedback on the dashboard.	
Sub-Flows: Views specific feedback details → display full message, date, & user information.	
Alternative/Exceptional Flows: No feedback available → returns empty list with message “No feedback found.” Database error → retrieval fails, returns error message “Unable to load feedback. Please try again later.”	

16: Update Feedback

Name: Update Feedback	**ID: SD-19**
Stakeholders and goals: System Developers - want to update user feedback messages and push them as alerts/pop-ups to all users so that developers do not need to reply individually.	

Description: Update feedback responses or announcements from users, and have the updated feedback be displayed as system-wide alerts/pop-ups to users. Instead of replying individually, the system broadcasts the messages to all relevant users.

Actors: System Developers

Trigger: The System Developer opens the Feedback/Notification Manager to update a selected feedback message.

Normal Flow:

1. The system developer opens the Notification/Feedback Manager page.
2. The system displays the list of pending or existing feedback entries.
3. The developer selects a feedback item to update.
4. The developer enters the updated feedback response or announcement message.
5. The system validates the update request.
6. The update is saved in the database.
7. The system generates a broadcast alert/pop-up version of the updated feedback.
8. The system sends the updated alert to the user interface for all affected users to see.
9. The system developer receives a confirmation message that the update was successful

Sub-Flows: View pending feedback → filter or search feedback entries before selecting one to update

Alternative/Exceptional Flows: Notification service unavailable → alert broadcasting fails, the system logs the attempted update for retry, the system informs the developer that the alert broadcast could not be delivered.
Invalid update input → returns error message & requests corrections.