**Automatic Number Plate Recognition (ANPR) System**

Date: 23/9/2025

Supervision and Auditing by: Eng.Sameh Saras

Created by: Khaleel Issa



**Table of Contents**

Contents

[Figure 1: 1. Introduction 3](#_Toc209518956)

[1. Introduction 3](#_Toc209518957)

[2. System Overview 3](#_Toc209518958)

[3. Functional Requirements 3](#_Toc209518959)

[3.1 Event Capture 3](#_Toc209518960)

[3.2 Data Sync 3](#_Toc209518961)

[3.3 Alerts 4](#_Toc209518962)

[3.4 Reporting 4](#_Toc209518963)

[3.5 User Management 4](#_Toc209518964)

[4. Non-Functional Requirements 4](#_Toc209518965)

[5. Database Design (ER Diagram) 4](#_Toc209518966)

[6. API Endpoints 4](#_Toc209518967)

[7. System Architecture Diagram 5](#_Toc209518968)

[8. User Roles & Permissions 5](#_Toc209518969)

[10. Summary 5](#_Toc209518970)

[Figure 2: SequenceDiagram: 6](#_Toc209518971)

[Sequence Diagram: 6](#_Toc209518972)

[Figure 3: ERD 11](#_Toc209518973)

[ERD 11](#_Toc209518974)

[Figure 4: Class Diagram 16](#_Toc209518975)

[Class Diagram 16](#_Toc209518976)

[Figure 5: State Diagram 23](#_Toc209518977)

[State Diagram 24](#_Toc209518978)

[Figure 6: ActivityDiagram 28](#_Toc209518979)

[ActivityDiagram 28](#_Toc209518980)

[Detailed Explanation of the Activity Diagram 29](#_Toc209518981)

[Figure 7: Safety 31](#_Toc209518982)

[Figure 8: GUI 32](#_Toc209518983)

## Figure 1: 1. Introduction

Automatic Number Plate Recognition (ANPR) System –

Functional and Technical Requirements

## 1. Introduction

This document defines the functional and technical requirements for a Smart Vehicle Monitoring System across multiple locations in the Middle East. The system captures vehicle events using cameras, identifies vehicle country by license plate, stores videos/images, and provides alerts and reporting functionalities.

## 2. System Overview

The system includes the following main components:

* Local Site Layer: Cameras and local database at each location.
* OCR and Country Detection: Identifies vehicle license plate and country.
* Central System (Main DB): Oracle database storing all locations’ events and metadata.
* Object Storage: Centralized storage for video/image files.
* Back-End API: Endpoints for data access, alerts, and reporting.
* User Management: Roles: User, Supervisor, Admin.

## 3. Functional Requirements

### 3.1 Event Capture

* Capture vehicle image and 5-second video.
* Detect license plate and direction (In/Out).
* Identify country from license plate.
* Store event locally and sync to central database.

### 3.2 Data Sync

* Local database sends data to Main DB.
* Metadata stored in Oracle DB.
* Videos/images stored in central Object Storage.

### 3.3 Alerts

* Duplicate vehicle in same location.
* Impossible travel time between locations.
* Unknown country detection.

### 3.4 Reporting

* Average events per day, month, year.
* Filters: Location, Date, Vehicle Type, Country.

### 3.5 User Management

* User: View events in assigned location only.
* Supervisor: Manage users in supervised locations, view events, delete users.
* Admin: Full access, manage users (delete/change roles), manage system settings.

## 4. Non-Functional Requirements

* Video compression: H.265.
* Image compression: WebP or JPEG.
* Real-time sync and alerts.
* Scalability to multiple locations.
* High availability and backup.
* System Shall support optional interoperability with external system.
* Safety.

## 5. Database Design (ER Diagram)

ER Diagram - Tables: Locations, Cameras, CarEvents, Users, PlatePatterns - Relationships: - Location 1.. Cameras - Location 1.. CarEvents - Camera 1.. CarEvents - User assigned to Location (nullable for Admin)

## 6. API Endpoints

* CRUD/events - Add vehicle event
* CRUD /events - Retrieve events
* CRUD /alerts - Retrieve alerts
* CRUD /reports - Event statistics
* CRUD /users - List users
* CRUD /users/{id} - Delete user
* CRUD /users/{id}/role - Change role (Admin only)

# 7. System Architecture Diagram

Architecture Diagram Local Cameras -> Local DB -> Central Oracle DB + Object Storage -> API -> Users

## 8. User Roles & Permissions

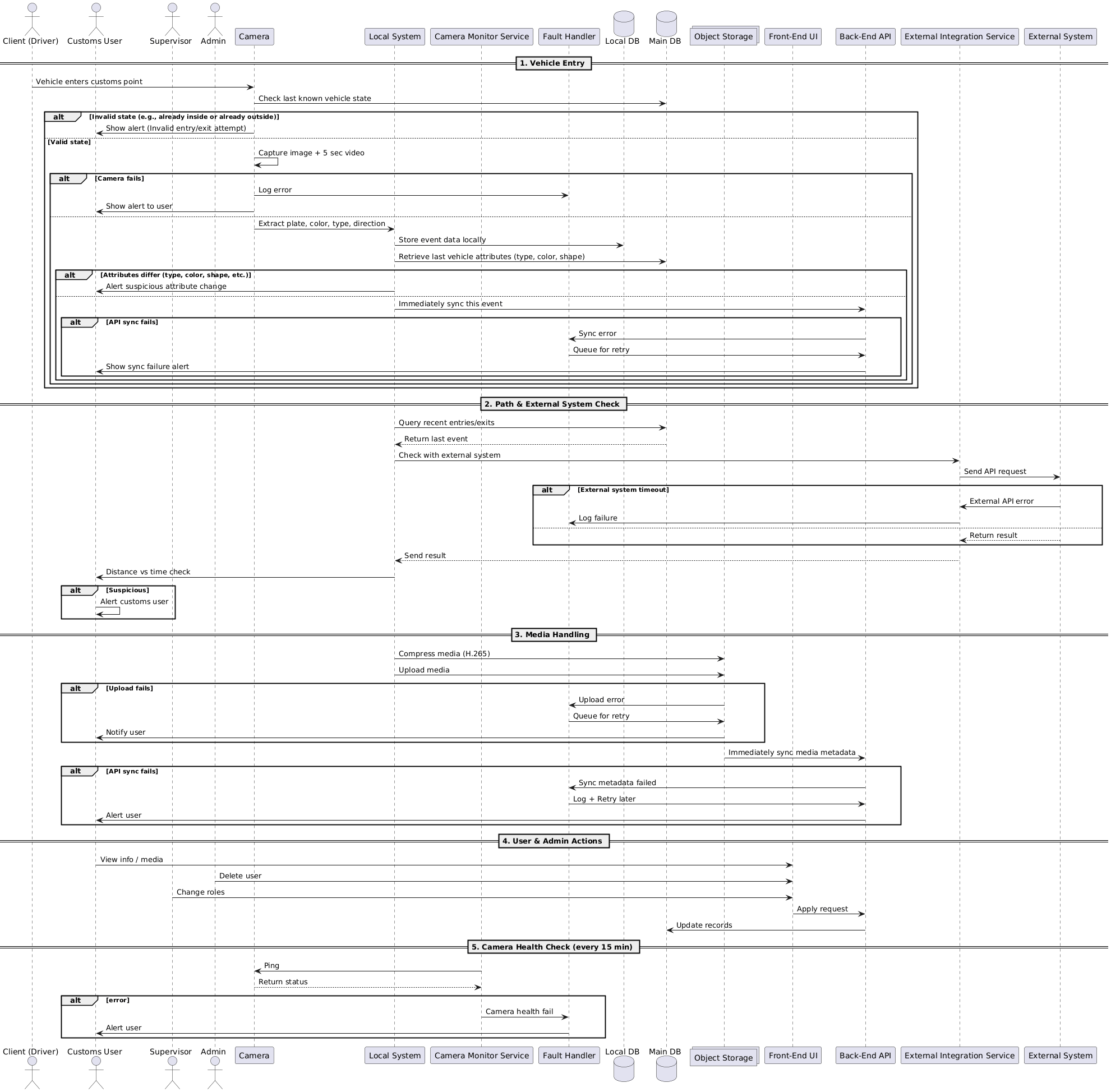
| Role | Permissions |
| --- | --- |
| User | View events in assigned location only |
| Supervisor | Manage users in supervised locations (delete/changelocation), view events |
| Admin | Full access: manage all users (delete/change role), manage system, view all events |

## 10. Summary

The system provides a complete solution for multi-location vehicle monitoring, including local capture, country detection, centralized storage, alerts, reporting, and role-based user management.

## Figure 2: SequenceDiagram:

# Sequence Diagram:



This diagram represents a detailed sequence diagram that illustrates the flow of events within a system. It outlines the interactions between different actors, such as users, camera systems, and other services. The sequence of actions is represented from the initial vehicle entry to various other checks, media handling, and system monitoring. Below is a comprehensive explanation of the diagram.

Overview:

The diagram presents a vehicle monitoring system with various interactions between different entities, such as cameras, databases, external systems, users, and other services. The key sections are:

1. Vehicle Entry Process

2. Path & External System Check

3. Media Handling

4. User & Admin Actions

5. Camera Health Check

Key Entities:

Client-Driven: The customer or the vehicle driver interacting with the system.

Customs User: Likely a user with specialized permissions for interacting with customs or checking vehicle details.

Supervisor: A higher-level user with oversight responsibilities.

Admin: An administrator who oversees system configuration and maintenance.

Camera System: Refers to the physical and software components responsible for capturing images/video.

Local System: The local system where data is processed or stored.

Camera Monitor Service: The service that monitors the camera and ensures its operational status.

Fault Handler: Handles any failures or issues in the system.

Local DB (Database): The local database where information is stored.

Main DB (Database): A central or primary database.

Object Storage: Refers to a storage system used for handling media objects such as images, videos, etc.

Front-End UI: The user interface presented to the user.

Back-End UI: The server-side processing and logic.

External Integration Service: Interfaces with external systems for additional processing or verification.

External System: An outside system that may interact with the system for various checks.

1. Vehicle Entry Process:

The sequence begins with the Client-Driven (vehicle) entering the system, which triggers several steps.

The Camera captures an image or a 5-second video upon vehicle entry.

This captured media is logged for further processing.

The system extracts key vehicle attributes such as type, color, direction, and

shape.

These attributes are used to immediately sync the event and queue it for entry in the system.

If a failure is detected, an alert is generated, and the system logs it for user notification. This could indicate a fault in capturing or analyzing the vehicle data.

2. Path & External System Check:

The system checks the recent entries and verifies the last recorded entry.

The external system is then queried for validation or confirmation. This could involve checking with an external source to confirm the validity or status of the entry.

If the external system is unavailable or returns an error (for example, external system timeout), the system logs the failure and processes the error.

After successfully receiving the results from the external system, the results are sent back for further actions.

3. Media Handling:

Compressing media (e.g., video or images): The media files (such as images and videos) are compressed to a manageable size for storage and transfer.

The system uploads the media to a centralized system or object storage.

If an error occurs during upload (e.g., upload failure), an alert is shown to the user, and the system retries the upload, queuing it for retry based on a defined schedule.

Additionally, metadata sync happens immediately after the media is uploaded, ensuring that the metadata about the media (e.g., size, date, location) is synchronized with the system's database.

Retry logic is triggered if there is a failure in syncing metadata or other aspects.

4. User & Admin Actions:

The admin or user can perform various administrative actions on the system.

For example, the Admin may delete certain entries, change user roles or permissions, and modify system configurations.

These actions are logged and processed for future tracking and auditing.

Users can also access the system to view info/media, allowing them to check previous logs or recorded media.

5. Camera Health Check:

The Camera Monitor Service regularly checks the health of the camera system. This check occurs periodically (every 15 minutes, in this case).

If the system detects a camera health failure, the camera's status is updated, and alerts are sent to the relevant users or system administrators to take corrective actions.

The system performs a ping to ensure that the cameras are responsive and operational. If a failure is detected, the camera system’s status is updated and communicated accordingly.

Summary of Key Interactions:

Client-Driven (Vehicle Entry):The process begins with vehicle entry, where images/videos are captured, attributes are extracted, and synced to the system.

Camera System: Responsible for capturing media and ensuring its health.

Path & External System Check: Validates the vehicle entry by checking recent entries and querying external systems.

Media Handling: Compresses, uploads, and syncs media while handling errors and retries.

User & Admin Actions: Enables users and administrators to manage data, view media, and modify system settings.

Camera Health Check: Monitors the health of the cameras at regular intervals.

This sequence diagram provides a clear and structured view of how the system functions in various operational stages, allowing stakeholders to understand the detailed flow of data and actions within the system. It ensures all components, from vehicle entry to system maintenance, are working cohesively to monitor and manage vehicle-related events efficiently.

## Figure 3: ERD

# ERD

A computer screen shot of a computer

AI-generated content may be incorrect.

Breakdown of the Schema:

1. Vehicle Table:

Attributes:

plate Number: The unique identifier for a vehicle (primary key).

color: The color of the vehicle.

type: The type of the vehicle (e.g., car, truck).

country: The country where the vehicle is registered.

lastKnownLocation: The most recent known location of the vehicle.

Purpose: This table stores basic details about the vehicle.

2. Event Table:

Attributes:

Event Id: A unique identifier for each event (primary key).

Vehicle plate Number: Foreign key referencing the plate Number in the vehicle table.

Camera camera Id: Foreign key referencing the camera Id in the camera table.

timestamp: The date and time when the event occurred.

direction: The direction in which the vehicle was moving.

location: The location where the event was recorded.

status: The current status of the event (e.g., completed, pending).

Purpose: This table records events related to vehicles, such as vehicle entry or status changes.

3. Media Table:

Attributes:

Media Id: A unique identifier for the media (primary key).

Event event Id: Foreign key referencing the event Id in the event table.

File Type: Type of media file (e.g., image, video).

url: The location where the media is stored.

Purpose: This table stores media files associated with specific events (images, videos).

4. Fault Table:

Attributes:

Fault Id: Unique identifier for the fault (primary key).

Camera camera Id: Foreign key referencing the camera Id in the camera table.

Error Code: A code identifying the type of error (e.g., camera failure).

description: A description of the fault.

Purpose: This table logs faults related to the camera systems, capturing error codes and descriptions of the issues.

5. External System Table:

Attributes:

System Id: Unique identifier for the external system (primary key).

name: The name of the external system (e.g., traffic control system).

Purpose: This table stores information about external systems that interact with the camera system.

6. Camera Table:

Attributes:

Camera Id: Unique identifier for the camera (primary key).

location: The location where the camera is installed.

status: The current operational status of the camera (e.g., active, inactive).

External System system Id: Foreign key referencing the system Id in the external System table.

Purpose: This table contains information about the cameras used in the system, including their location, status, and connection to external systems.

7. Report Table:

Attributes:

Report Id: Unique identifier for the report (primary key).

User username: Foreign key referencing the username in the user table.

Event event Id: Foreign key referencing the event Id in the event table.

timestamp: The date and time the report was generated.

parameters: Additional parameters or details associated with the report.

Purpose: This table logs reports generated by users in relation to events.

8. User Table:

Attributes:

username: Unique identifier for the user (primary key).

role: The role or permission level of the user (e.g., admin, supervisor).

Purpose: This table stores information about the users who interact with the system, including their roles and permissions.

Relationships:

Vehicle to Event: A one-to-many relationship between the vehicle and event tables. Each vehicle can have multiple events associated with it, but each event is related to only one vehicle.

Event to Media: A one-to-many relationship between the event and media tables. Each event can have multiple media files associated with it.

Camera to Event: A many-to-one relationship where each event is captured by a single camera, but a camera can capture multiple events.

Fault to Camera: A one-to-many relationship between the camera and fault tables. Each camera can have multiple faults logged.

Event to Report: A one-to-many relationship where each event can have multiple reports associated with it.

User to Report: A one-to-many relationship where each user can generate multiple reports.

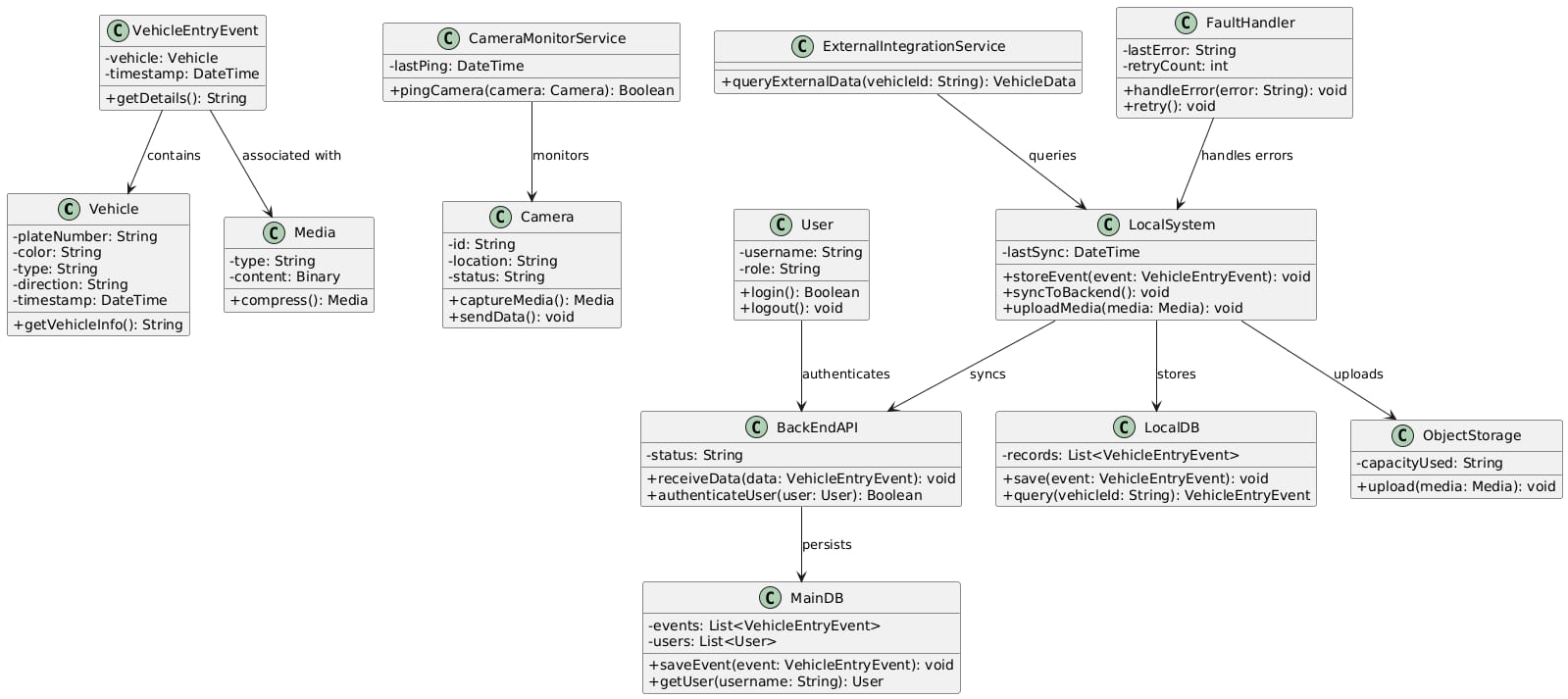
Summary:

The database schema models key entities and relationships in the vehicle monitoring system. It includes vehicles, events, cameras, media, faults, external systems, reports, and users. The schema is designed to allow the system to track vehicle activities, capture related media, detect faults in cameras, and generate reports for further analysis.

This structure ensures that all necessary data points related to vehicles, cameras, events, and system health are captured and can be queried efficiently. The relationships between the tables reflect the real-world interactions between these entities, such as a vehicle triggering events, events generating media, and faults being associated with cameras.

## Figure 4: Class Diagram

# Class Diagram



Class Diagram Breakdown:

1. VehicleEntryEvent Class:

Attributes:

vehicle: An instance of the Vehicle class, which contains details of the vehicle involved in the event.

timestamp: A DateTime object indicating when the event occurred.

Methods:

getDetails(): Returns a string containing details of the vehicle entry event.

Relationships:

Contains a Vehicle object, linking the event to a specific vehicle.

2. Vehicle Class:

Attributes:

plateNumber: The vehicle’s unique identifier (e.g., license plate).

color: The color of the vehicle.

type: The type or category of the vehicle (e.g., car, truck).

direction: The direction the vehicle is moving.

timestamp: The time when the vehicle's information was last updated.

Methods:

getVehicleInfo(): Returns a string with the vehicle's information (plate number, color, etc.).

3. Media Class:

Attributes:

type: The type of media (e.g., image, video).

content: The binary content of the media file.

Methods:

compress(): Compresses the media content to a more efficient format or size.

4. Camera Class:

Attributes:

id: A unique identifier for the camera.

location: The location of the camera.

status: The current status of the camera (active, inactive).

Methods:

captureMedia(): Captures media (image/video) using the camera.

sendData(): Sends the captured data to a designated service or storage.

5. CameraMonitorService Class:

Attributes:

lastPing: The time of the last successful communication with a camera.

Methods:

pingCamera(camera: Camera): Pings the camera to check its operational status and returns a boolean indicating whether the camera is operational.

Relationships:

Monitors the status of cameras in the system.

6. ExternalIntegrationService Class:

Attributes:

lastError: Describes the last error encountered by the external integration service.

retryCount: Tracks how many retry attempts were made to resolve an issue.

Methods:

queryExternalData(vehicleId: String): Queries external systems for vehicle data using the vehicle's unique identifier.

handleError(error: String): Handles errors encountered during external data queries.

retry(): Attempts to retry a failed external query.

Relationships:

Queries external systems for vehicle data.

7. User Class:

Attributes:

username: The unique identifier for the user.

role: The user’s role within the system (e.g., admin, user).

Methods:

login(): Authenticates the user and returns a boolean indicating whether login was successful.

logout(): Logs the user out of the system.

Relationships:

Authenticates a user for system access.

8. LocalSystem Class:

Attributes:

lastSync: A DateTime object indicating the last synchronization time with external systems or databases.

Methods:

storeEvent(event: VehicleEntryEvent): Stores vehicle entry events locally in the system.

syncToBackend(): Syncs the stored event data to the backend system.

uploadMedia(media: Media): Uploads media to a storage or server.

Relationships:

Stores events and media locally and uploads them when necessary.

Syncs data with the backend.

9. BackEndAPI Class:

Attributes:

status: The current status of the backend system.

Methods:

receiveData(data: VehicleEntryEvent): Receives vehicle entry data from the local system.

authenticateUser(user: User): Authenticates a user for access to the backend.

Relationships:

Persists data from the local system.

10. LocalDB Class:

Attributes:

records: A list of VehicleEntryEvent objects, representing all the vehicle entry events stored in the local database.

Methods:

save(event: VehicleEntryEvent): Saves a new vehicle entry event to the local database.

query(vehicleId: String): Queries the local database for vehicle entry events based on a given vehicle ID.

Relationships:

Stores vehicle entry events locally.

11. MainDB Class:

Attributes:

events: A list of VehicleEntryEvent objects stored in the main database.

users: A list of User objects, representing the users of the system.

Methods:

saveEvent(event: VehicleEntryEvent): Saves a vehicle entry event to the main database.

getUser(username: String): Retrieves a user based on the provided username.

Relationships:

Persists vehicle entry events and user information.

12. ObjectStorage Class:

Attributes:

capacityUsed: A string that tracks the amount of storage used.

Methods:

upload(media: Media): Uploads media (e.g., images, videos) to the object storage system.

Summary of Class Interactions:

VehicleEntryEvent: Contains a Vehicle and a Media object and is processed by other classes such as LocalSystem and BackEndAPI.

CameraMonitorService: Monitors the status of cameras and pings them regularly.

ExternalIntegrationService: Handles interactions with external systems to retrieve additional data about vehicles.

User: Authenticates and interacts with the system to log in and out.

LocalSystem: Stores events locally and synchronizes them with external systems, managing media uploads as well.

BackEndAPI: Receives data from the local system and authenticates users.

ObjectStorage: Manages storage of media files related to vehicle entry events.

## Figure 5: State Diagram

This class diagram illustrates how the system components work together to manage vehicle entry events, media processing, camera monitoring, user interactions, and synchronization with external systems. The system’s architecture is designed to handle vehicle data efficiently, provide storage for media, and ensure fault tolerance through the FaultHandler.

# State Diagram

A diagram of a system

AI-generated content may be incorrect.

Overview of the Vehicle Monitoring and Access Control System

This state machine diagram outlines the operational logic for a comprehensive Vehicle Monitoring and Access Control System. The system is designed to track vehicles, validate their entry and exit, and identify suspicious activities. It processes vehicle information from detection to final exit, ensuring a secure and efficient flow. The diagram uses standard UML notation to show the various states, transitions, and actions within the system's lifecycle.

Detailed State-by-State Breakdown

The system's process can be broken down into distinct phases, each represented by a specific state.

1. Vehicle Detection and Initial Checks

WaitingForDetection: This is the initial state where the system is idle and continuously monitoring for vehicles.

Detected: The system transitions to this state when a vehicle is detected by a camera. This triggers the process of querying the vehicle's last known state to determine if it has already been recorded.

CheckingState: In this state, the system queries a database to retrieve the last known status of the detected vehicle.

InvalidState: If the vehicle's state is not valid for an entry or exit transaction (e.g., the system has no prior record, or the last known state is inconsistent), the process is terminated.

Capturing: If the vehicle's state is valid, the system transitions here to begin capturing data, including the vehicle's license plate, color, and type.

2. Data Analysis and Path Verification

AnalyzingAttributes: After capturing data, the system analyzes the vehicle's current attributes. This state involves comparing the newly captured data with the last known attributes stored in the database.

AttributesMatch: If the captured attributes match the stored attributes, the system proceeds to verify the vehicle's movement.

SuspiciousChange: If the attributes differ (e.g., a different license plate is detected on the same vehicle), this state is triggered. The system raises an alert, and the vehicle's data is sent for manual review.

VerifyingPath: This state is critical for security. The system analyzes the time and distance between the vehicle's last known position and its current location.

NormalMovement: If the time and distance are within expected parameters, the movement is considered normal.

SuspiciousMovement: If the time and distance are inconsistent (e.g., the vehicle moved too fast or too slow for the distance), the system flags it as suspicious. This state triggers an alert to customs or security personnel.

3. Investigation and System Actions

UnderInvestigation: Both SuspiciousChange and SuspiciousMovement states lead to this state. The system requires manual intervention to clear the vehicle. This is a holding state where security personnel can investigate the flagged vehicle.

UploadingMedia: Once a vehicle's entry is confirmed (either through NormalMovement or after being manually cleared from UnderInvestigation), the system begins uploading media and associated data to the central database.

SyncingData: This state ensures that all data related to the vehicle's entry is synchronized with the main system. This ensures data integrity and consistency.

Inyard: Once the entry process is complete, the vehicle's state is updated to Inyard, indicating that it is inside the controlled area.

4. Vehicle Exit Process

AttemptingExit: The system transitions to this state when a request for a vehicle's exit is detected.

ValidatingExit: The system checks the vehicle's current state to validate if it is eligible to exit. A valid state would be Inyard.

ExitDenied: If the vehicle's state is invalid (e.g., it was never registered as entering the area), the exit is denied, and the process is terminated.

CapturingExit: If the exit is valid, the system captures media and data related to the exit event.

UploadingExitMedia: The captured exit data is uploaded to the central database.

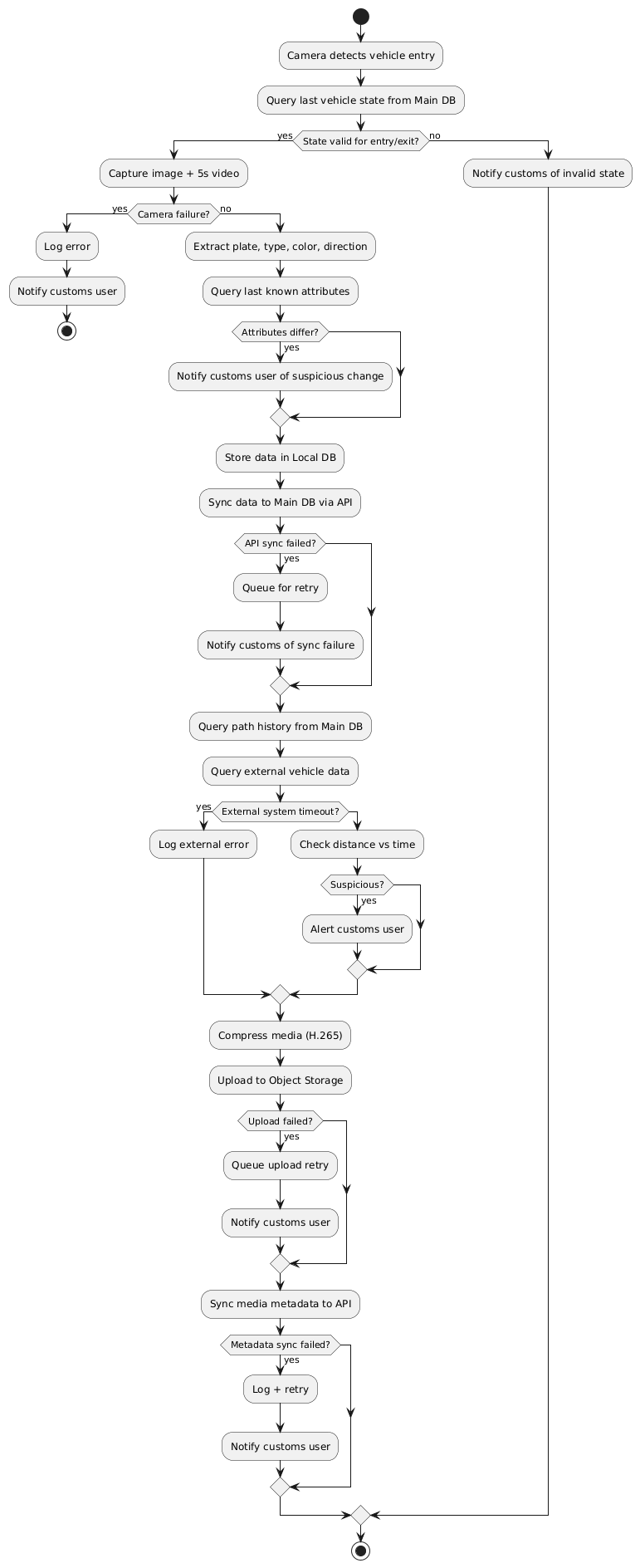
SyncingExitData: This state ensures all exit data is fully synchronized.

Exited: Once the synchronization is complete, the vehicle's state is officially updated to Exited, and the process concludes.

The diagram also shows a final state, represented by a black circle with a white border, which signifies the end of the process for a single vehicle transaction. The loop back from Exited and ExitDenied to the initial state indicates that the system is ready to monitor for the next vehicle.

## Figure 6: ActivityDiagram

# ActivityDiagram



# Detailed Explanation of the Activity Diagram

The diagram represents a comprehensive, multi-step process for handling a transaction, designed to ensure security, integrity, and proper error handling. Each shape and arrow has a specific meaning in the Unified Modeling Language (UML) notation, detailing the flow of control and data.

1. Initiation and Transaction Validation

The process begins with the Start Node, a solid black circle, signifying the entry point of the workflow. The very first action is Query transaction from DB, where the system retrieves the details of the transaction from a database.

Following this, the flow arrives at the first Decision Node, a diamond shape labeled transaction valid?. This is a critical check to verify the basic integrity of the transaction.

If the answer is No, the system executes the Log invalid transaction action. This step is crucial for auditing and troubleshooting, as it records a permanent record of the invalid attempt. The process then concludes at a Final Node, a black circle with a white outline.

If the answer is Yes, the process continues to the next phase, indicating the transaction has passed its initial validation.

2. User and Account Verification

The next action, Query user information from DB, retrieves the user's data associated with the transaction. This is a crucial security step.

The system then enters another Decision Node labeled user verified?. This node determines if the user's identity or status is valid and trustworthy.

If the answer is No, the system flags the user for manual inspection by performing the Flag user for review action. It then proceeds to Notify Admin, sending an alert to the system administrator or a security team. This path is designed to prevent fraudulent activities. The process then terminates.

If the answer is Yes, the flow proceeds to the final checks, confirming the user is authorized and trustworthy.

3. Financial Check and Transaction Processing

The process moves to Check user balance, which is an essential financial validation. A subsequent Decision Node, balance sufficient?, checks if the user has enough funds to cover the transaction amount.

If the balance is No, the system executes the Decline transaction action. It then notifies the user via Notify User, providing a clear reason for the decline. The process concludes here.

If the balance is Yes, all preliminary checks are passed, and the system is ready to proceed with the core operation.

The Process transaction action is then executed. This represents the central activity of the diagram, where the financial transfer or operation actually takes place.

4. Finalization and Error Handling

After processing the transaction, the system enters the final Decision Node, transaction successful?. This check confirms the outcome of the processing step.

If the transaction is No, meaning it failed for some technical reason (e.g., a network error or a system timeout), the system performs two actions: Log error and Notify Admin. This ensures that the failure is documented and an administrator is alerted to investigate the issue. The process terminates.

If the transaction is Yes, indicating a successful operation, the system proceeds with finalization steps. This includes Update DB to reflect the new state of the user's account and the transaction record. Finally, the Send confirmation to user action sends a notification to the user about the successful transaction.

The diagram's use of distinct Final Nodes at different points of failure and a single Final Node at the successful conclusion highlights the system's design for comprehensive outcome management. This structured approach is vital for building a reliable and secure financial or e-commerce platform.

## Figure 7: Safety

Safety

The system shall ensure operational safety through:

Automatic data backup and recovery mechanisms.

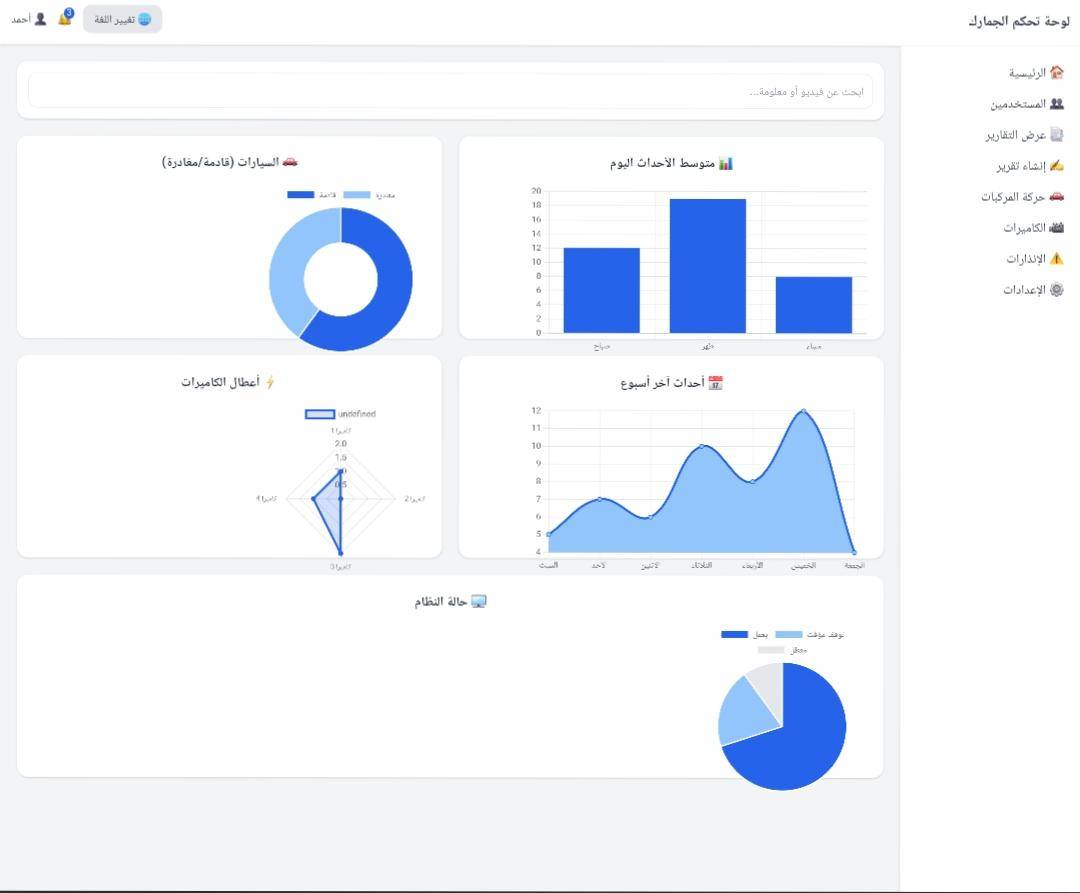
Protection of critical vehicle and user data against unauthorized access.

Failure-handling procedures to prevent data loss during unexpected shutdowns.

Alerts and logs for abnormal system behavior to allow proactive action.

This requirement is classified as Non-Functional, since it relates to how the system protects itself rather than a direct user function

## Figure 8: GUI



The proposed Graphical User Interface (GUI) is designed to be simple, clear, and user-friendly, enabling customs officers to easily monitor and manage their daily tasks. The image below represents a conceptual mock-up of the dashboard. It is intended for illustration purposes only and may contain inaccuracies or omissions; it does not reflect the final product.

Main Components of the Interface:

Top Bar (Header): Includes quick access to language switching, notifications, and the user profile (with login/logout options).

Sidebar (Navigation Menu): Provides access to the main functional areas such as: Home (Dashboard overview) Users management Reports (view existing or create new) Vehicle movement (arrivals and departures) Cameras (live view and status) Alerts and alarms System settings Search Section: A dedicated search field that allows the user to look up videos or related information directly from the dashboard.

Visualization Area (Charts & Graphs): Displays key system metrics in a simple, consistent layout, such as: Daily average events Vehicle traffic (arrivals vs. departures) Weekly events trend Camera failures or downtime Overall system status The overall design emphasizes clarity, simplicity, and usability, using neutral colors (white and gray) with highlights in blue to maintain a professional and calm appearance.