KING SAUD UNIVERSITY

COLLEGE OF COMPUTER AND INFORMATION SCIENCES
DEPARTMENT OF SOFTWARE ENGINEERING

كلية علوم الحاسب والمعلومات
قسم هندسة البرمجيات

SWE321:
Software Design & Architecture

Project

Vehicle Tracking System

Software Design & Architecture

Version 1

9/10/2023

Anas Alsubaie
Lead Software Engineer

List of Authors:

Printed Name	Title	Date
Anas Alsubaie	Lead Software Eng.	9/4/23
Abdulmajeed Mohammed Alromaih	Software Eng.	9/4/23
Sultan Hammod Al-enzi	Software Eng.	9/4/23
Khalid Abdullah Alharbi	Software Eng.	9/4/23
Suliman Aljarbou	Software Eng.	9/4/23

Table of Contents

Contents

Revision History

Version#	Date	Description	Reason
1	10/9	Added system context view diagram	Instructed to do so.

1. Introduction

1.1 Description and Purpose

Our project is a sub-system of a broader fleet management application; specifically focusing on the Vehicle Tracking aspect. Using GPS technology this sub-system provides real-time(live) location data of vehicles. Enabling administrators to manage and allocate resources effectively.

This sub-system is essential as it forms the backbone of operational efficiency, enabling tasks such as driver attendance logging, vehicle dispatch based on location, and promoting driver accountability. Its implementation is crucial to businesses in the transportation and logistics sector, enhancing their service reliability and customer satisfaction.

The only change we have added is the system context view diagram, we have been given instructions about that.

2. Problem Domain Analysis

2.1 Domain:

Domain Description:
This project lies within the field of transportation and logistics. Focusing on improving vehicle management and service delivery efficiency in the taxi and delivery sectors through technological intervention.

Problem Statement:
The current issues we are facing in the domain include inefficient vehicle allocation for real time, safety concerns, inaccuracies in attendance, payroll management, lack of accountability, and transparency in operations.

Solution Strategy:
Our approach will involve the implementation of a GPS-based vehicle tracking system, facilitating real-time monitoring, increased safety measures, automated attendance, improved transparency, and accountability in operations.

Relevant Concepts and Vocabulary:
Geofencing: Virtual geographic boundaries triggering responses when a vehicle enters or leaves a predefined area.
Real-Time Monitoring: Continuous tracking of vehicle locations and status.
Route Optimization: Utilizing data to identify the most cost-efficient and quickest routes.
Telematics: An interdisciplinary field encompassing telecommunications and vehicular technologies for enhancing road safety.
User Authentication: A process requiring users to identify and verify themselves for system access.
Database Management: Storing, retrieving, and managing data including vehicle details and trip records.

This project aims to address existing challenges, revolutionizing vehicle management and enhancing efficiency and safety in transportation services.

3. The System Context View

3.1 Scenario 1

User: Admin Monitoring and Management
Scenario Title: Enhanced Fleet Management Through Admin Dashboard

Description:
A fleet manager wants to know the current location of all the vehicles in the fleet. The manager logs into the system using their admin ID and password. The system presents a visual dashboard showing the real-time location of each vehicle on a map**.**

Input: Admin id and password.

Output:
A real-time map displaying the location of each vehicle in the fleet.

Benefit:
The fleet manager can monitor the entire fleet in real-time, allowing for quick decision-making and more efficient fleet management.

3.2 Scenario 2

User: Driver
Scenario Title: Dynamic Route Planning for Drivers

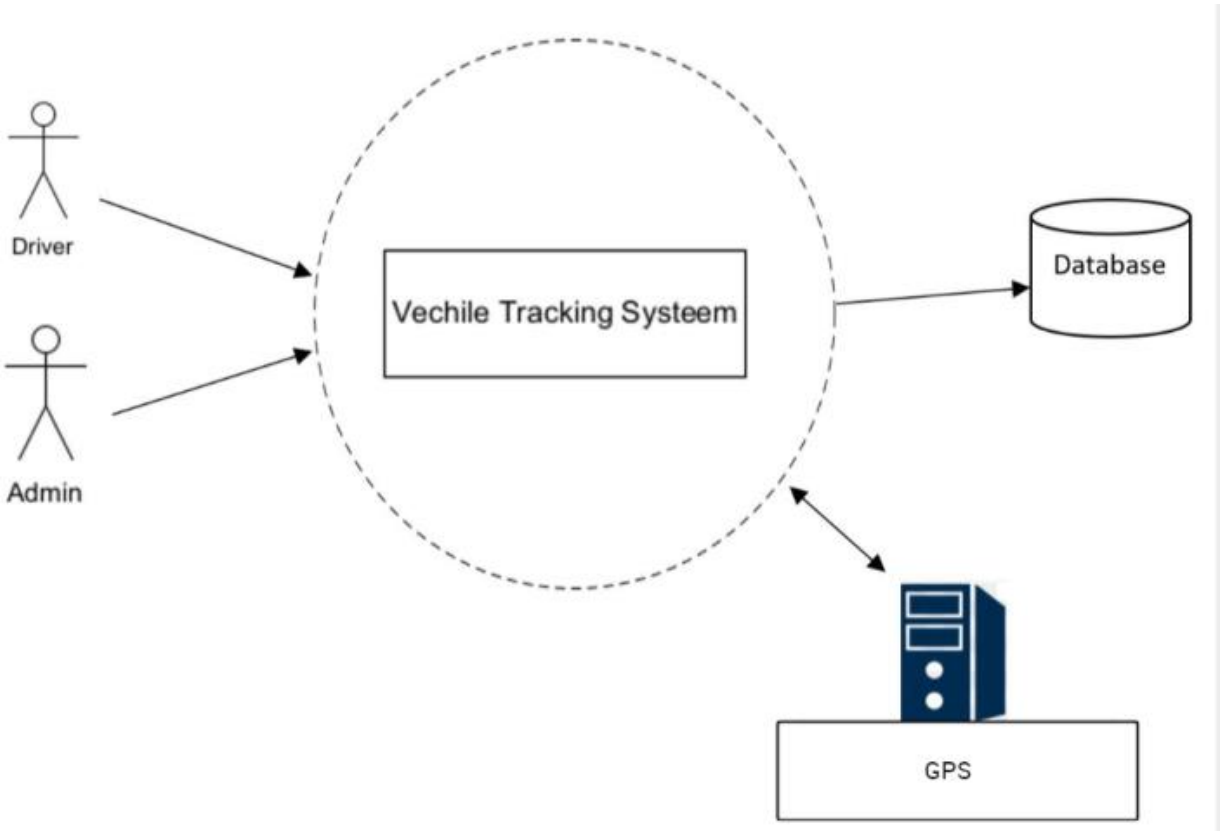
Description: Before starting a trip, the driver inputs the destination details into the system.

Input: Destination details and current location data from GPS.

Output: The system swiftly proposes the most efficient route, considering real-time traffic conditions, and estimates the time of arrival at the destination.

Benefit: Enables the driver to avoid delays and navigate the city efficiently, ensuring customer satisfaction and fuel savings.

System context view diagram:



4. Specific Requirements

4.1 Functional Requirements

4.1.1 The system shall allow the driver to log in using a unique user ID and password.
4.1.2 The system shall allow the admin to log in using a unique admin ID and password.
4.1.3 The system shall track the real-time location of each logged-in driver using GPS.
4.1.4 The system shall display the real-time location of all active drivers to the admin-accessible dashboard.
4.1.5 The system shall log the attendance of drivers based on their login and logout times.
4.1.6 The system shall allow the admin to view the status of the registered driver.
4.1.7 The system shall allow the driver to set the availability status for rides (available or unavailable).
4.1.8 The system shall maintain a history of all rides undertaken by each driver, including ride time, start and end locations.
4.1.9 The system shall provide a dashboard for admins to view summary statistics including total active drivers, total rides completed, and ongoing rides.
4.1.10 The system shall provide real-time updates to admins when a driver's status changes (e.g., when they become available for rides, start a ride, complete a ride).
4.1.11 The system shall allow the admin to generate reports for driver performance, total rides, revenue.
4.1.12 The system shall calculate the driver's salary based on their attendance and the number of rides they have completed.
4.1.13 The system shall allow the admin to edit the driver profile (update and delete the driver account.).
4.1.14 The system shall allow the admin to add driver account.

4.2 Non-Functional Requirements

4.2.1 Reliability

4.2.1.1 The system shall not crash if there is a failure.
4.2.1.2 The system shall be error free.

4.2.2 Availability

4.2.2.1 The system shall be available 99% of the time.
4.2.3 Performance
4.2.3.1 The system shall be capable of real-time tracking with a response time of less than a second.
4.2.3.2 The system shall be capable of processing a large quantities of data quickly to facilitate smoothed vehicle tracking

4.2.4 Portability

4.2.4.1 The system shall be compatible with any system.

4.2.5 Interoperability
4.2.5.1 The system shall be compact with various devices such as (smart phone, tablet, etc..).

4.2.6 Security
4.2.6.1 All data transmitted between the server and client shall be encrypted to be protected against unauthorized access.
4.2.7 Usability
4.2.7.1 The system shall have a user-friendly interface that can be operated without extensive training.

5 Challenges:

5.1 The system must be designed to handle growth in the number of users (drivers and admins) and vehicles. Achieving a design that is scalable and can accommodate increasing data volumes is a significant challenge.

5.2 Creating an intuitive and user-friendly interface for both drivers and admins can be challenging. The design must present complex data in an easily understandable manner and provide efficient navigation and controls.

5.3 Incorporating robust security measures into the system design to protect sensitive data like user credentials and location information can be complex. It involves careful consideration of encryption methods, access controls, and secure data transmission.

6 Projection:

6.1. Design of the Vehicle Tracking System: As a team, we will create a detailed design for the vehicle tracking system. This includes system architecture, user interface design, and data flow diagrams for features like real-time location tracking, driver attendance logging, and ride allocation based on location.

6.2 Understanding of System Design Principles: We expect to gain a deeper understanding of system design principles, particularly how they apply to real-time systems like vehicle tracking.

6.3 User Interface (UI) Design Skills: We aim to enhance our skills in designing intuitive and user-friendly interfaces, with a focus on a user-centric design approach.

6.4 Security Design Principles: We anticipate learning about incorporating security measures into the system design, which is crucial when handling sensitive user data.

6.5 Project Management: Managing this design project from start to finish will provide us with valuable experience in project management, including planning, task delegation, and meeting deadlines.

6.6 Teamwork and Collaboration: Throughout the process, we will strengthen our skills in team collaboration, conflict resolution, and effective communication.

SWE321:
Software Design & Architecture

PART 2

Project

Vehicle Tracking System

SYSTEM ARCHITECTURE DOCUMENT

Version 2

9/28/2023

Anas Alsubaie
Lead Software Engineer

1. Methodology

Plan

Modules	Estimated Time	Responsibility
Database	2 weeks	Khalid
Server	1 week	Anas
User/Driver	2 weeks	Suliman
Admin	1 week	Abdulmajed
Integration Testing and Debugging	3 weeks	Sultan

Database: Design the database schema and implement it using a database management system It can be tested using unit tests and integration tests with the database.

Server: interact with the database and handle incoming GPS data from drivers.

User/Driver: Develop the app that will run on the driver's device, capturing GPS data and sending it to the server. This module can be tested on a variety of devices and network conditions to ensure reliability.

Admin: This is likely the most complex component, consisting of several subsystems (Authentication, Driver Location Viewer, Task Allocator, Record Keeper). Each subsystem should be developed and tested individually before integrating them together.
Integration Testing and Debugging: Once all modules are developed, they should be tested together to ensure they work correctly as a complete system.

Schedule for the estimated project:

The estimated time for the project will take approximately 10 weeks to complete.
It would look like this:

- Week 1-2: **Database**
- Week 3: **Server**
- Week 4-5: **User/Driver**
- Week 6: **Admin**
- Week 7-8..: **Integration Testing and Debugging**

The prototype demonstration can be planned for the 5th or 6th week after GPS tracking and location reporting are ready. The design deadline can be set in the 8th week, and the final implementation, and final demonstration can be planned in the 10th week.

Key or Risk parts of the system:

The real-time tracking and reporting of vehicles' locations is a key part of the system. The risk we might face is in the accuracy and reliability of the GPS data. The accuracy in allocating the driver. Also, the ability to process the given data and display it in real time.

2. System Architecture

2.1 Design Decision

As a project designed for special purposes for vehicle tracking system, we should care about the design decisions to support and implement the non-functional requirement.
Usability & Accessibility

1- GPS Hardware: High-precision GPS tracker with 10-second location updates.
2- Data Storage: Firebase real-time storage Cloud-based NoSQL database for scalable storage.
3- GPS Tracking: using Google Maps Geolocation API as real-time GPS tracking with location updates every 10 seconds.
4- Backend Framework: Use Node.js, designed to handle up to 10,000 concurrent connections.
5- Admin interface: Use HTML, CSS, and JavaScript for the admin.
6- User interface: using flutter-based components to design the UI.
7- Real-Time Updates: Implement real-time data ensuring the user interface reflects the current state of the system with updates every 5 seconds.
8- Security: use HTTPS and data encryption with a target of zero data breaches or leaks, and strong authentication mechanisms with a target of zero unauthorized accesses.
9- API Integration: Ensure compatibility with necessary APIs, including map visualization, within a latency of 1 second.
10- Mobile Application: Develop a mobile app for drivers with a response time of under 2 seconds and compatibility with 99% of smartphones.

2.2 Domain Model

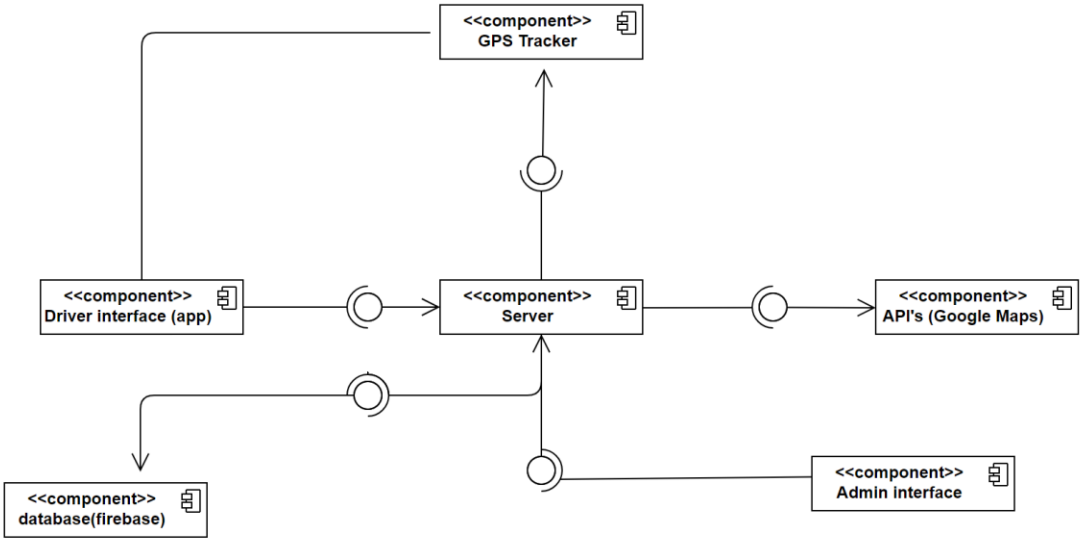


Figure 1 briefly description of the domain components

2.3 Architectural Style

The big picture(macro-level) of the system architecture style is layered architecture, where the top layer provides interface for clients(Admin, Driver), Middle layer provides the business logic processing, and the lower layer provides the utility specific services.

2.3.1 GPS Tracking Subsystem:

Architectural Style: Data-Centric Architecture.

Why? This subsystem revolves around the GPS data generated by the trackers. The main function is to capture, process, and forward this data. The Data-Centric Architecture focuses on data generation, transformation, and storage, making it suitable for this subsystem.

Considered Styles: A distributed architecture was considered to handle data processing closer to the source (GPS trackers). However, considering the simplicity of the data involved (primarily location coordinates), a centralized data-centric system would be more efficient and manageable.

2.3.2 Server Subsystem:

Architectural Style: Layered Architecture.

Why? The server subsystem has distinct responsibilities like receiving data, processing it, storing it in the database, and serving it to clients (Admin and Driver interfaces). These responsibilities naturally form layers, making Layered Architecture suitable for this subsystem.

Considered Styles: A microservices architecture was considered. However, considering the scale and complexity of the system, the overhead of managing multiple services would outweigh the benefits. Therefore, a layered approach is more practical.

2.3.3 Admin Interface and Driver Interface Subsystem:

Architectural Style: Model-View-Controller (MVC).

Why? Both the Admin and Driver interfaces involve user interaction, data display, and data manipulation. The MVC architecture separates concerns (data, UI, and control logic), making it ideal for these interfaces.

Considered Styles: A Presentation-Abstraction-Control (PAC) architecture was considered, which also separates concerns. However, MVC is more widely used, better supported in most web and mobile development frameworks, and simpler to implement for the interfaces in this system.

2.3.4 Database Subsystem:

Architectural Style: Database-Centric Architecture.

Why? This subsystem's primary purpose is to store and retrieve data. A Database-Centric Architecture, where the database is the central component, is thus the most suitable.

Considered Styles: A distributed database architecture was considered for increased scalability and fault tolerance. However, given the nature and volume of data for this system, a single central database (Firebase, in this case) would be more cost-effective and simpler to manage.

2.3.5 APIs Subsystem:

Architectural Style: Service-Oriented Architecture (SOA).

Why? This subsystem involves integrating with external services (like Google Maps) through their APIs. SOA, where functionality is provided as services over a network, is the most fitting for this subsystem.

Considered Styles: A Resource-Oriented Architecture (ROA) was considered, which is a specific type of SOA used in RESTful web services. However, not all APIs used may follow REST principles, making a broader SOA approach more suitable.

2.4 Structural model

2.4.1 Components:

2.4.1.1 Admin Web Application: This is a web-based interface that allows admins to manage the system. They can view real-time and historical location data, manage users, and handle other administrative tasks.

2.4.1.2 Driver Application: This is a mobile app that allows drivers to interact with the system. They can view their routes, track their own location, and perform other tasks as required.

2.4.1.3 Server: This is the heart of the system. It receives GPS data from the GPS tracker, processes it, and stores it in the database. It also retrieves data from the database in response to requests from the Admin and Driver interfaces.

2.4.1.4 Firebase Database: This is where the system stores all the data. It includes GPS data, user data, and other necessary information.

2.4.1.5 GPS Tracker: This is based on the driver software collecting the GPS data.

2.4.1.6 Google Maps API: This is used by the server to convert GPS coordinates into readable addresses.

2.4.1.7 API Subsystem: This services as the interface between the server, the Admin Interface, the Driver Interface, and potentially other third-party systems. It allows these components to communicate with the server in a structured way.

2.4.2 Initial class diagram (high-level diagram showing the major components and their interaction)

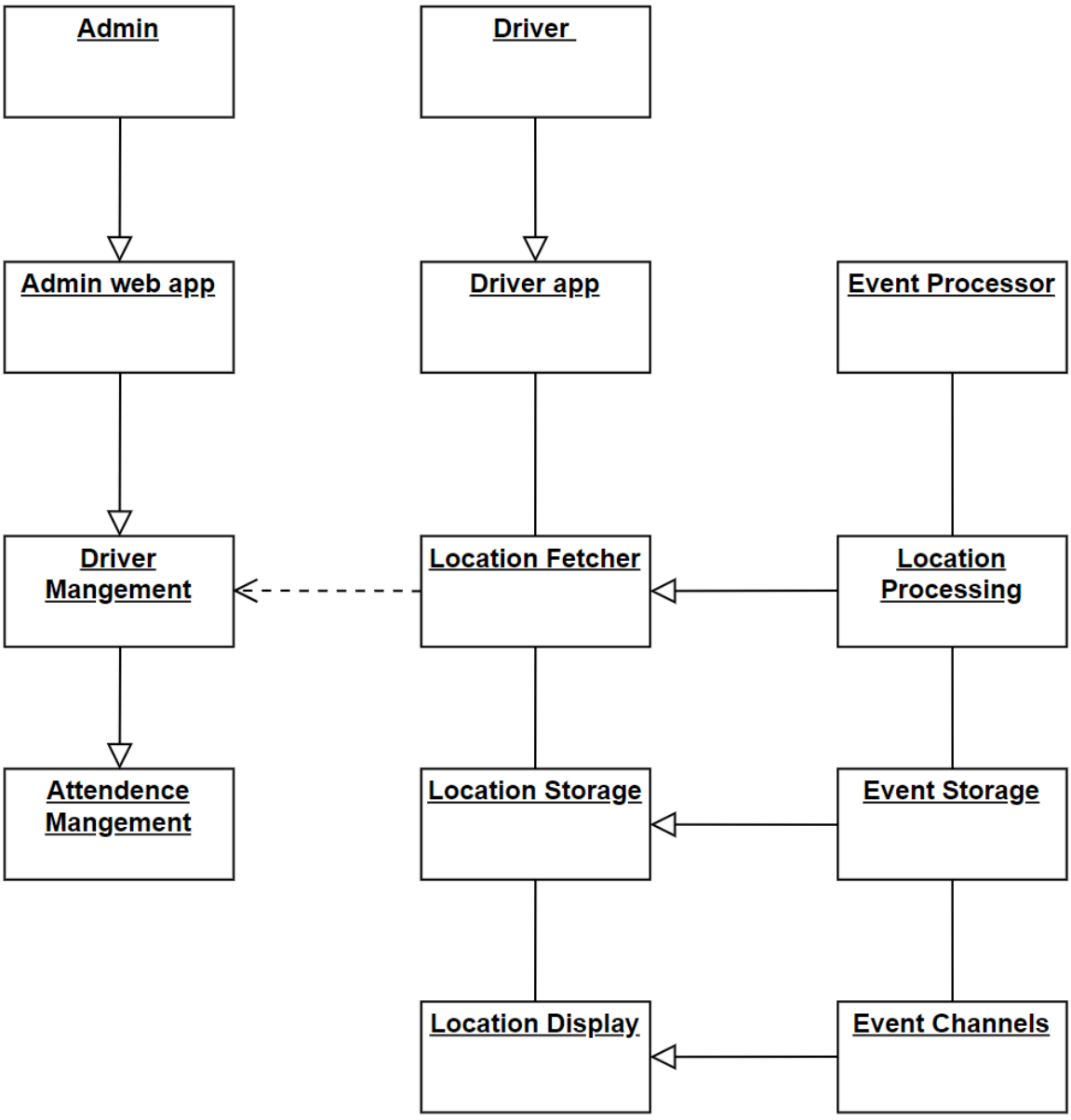


Figure 2 In this diagram, the rectangles represent the major components of the system, and the arrows represent the interactions between components.

2.4.3 How the Architecture Meets Non-Functional Properties

2.4.3.1 Real-time Updates: The EDA allows for real-time updates. When a driver's status or location changes, an event is generated, processed, and the admin interface is updated.

2.4.3.2 Scalability: The EDA can handle additions to the system (like adding more drivers or admin users) without affecting the existing system.

2.4.3.3 Reliability: With proper error and exception handling, the system can ensure that all events are properly produced, processed, and stored.

2.4.4 Use case Diagram.

Thie section represents the role of each admin and driver and their functionalities in the system using case diagram.

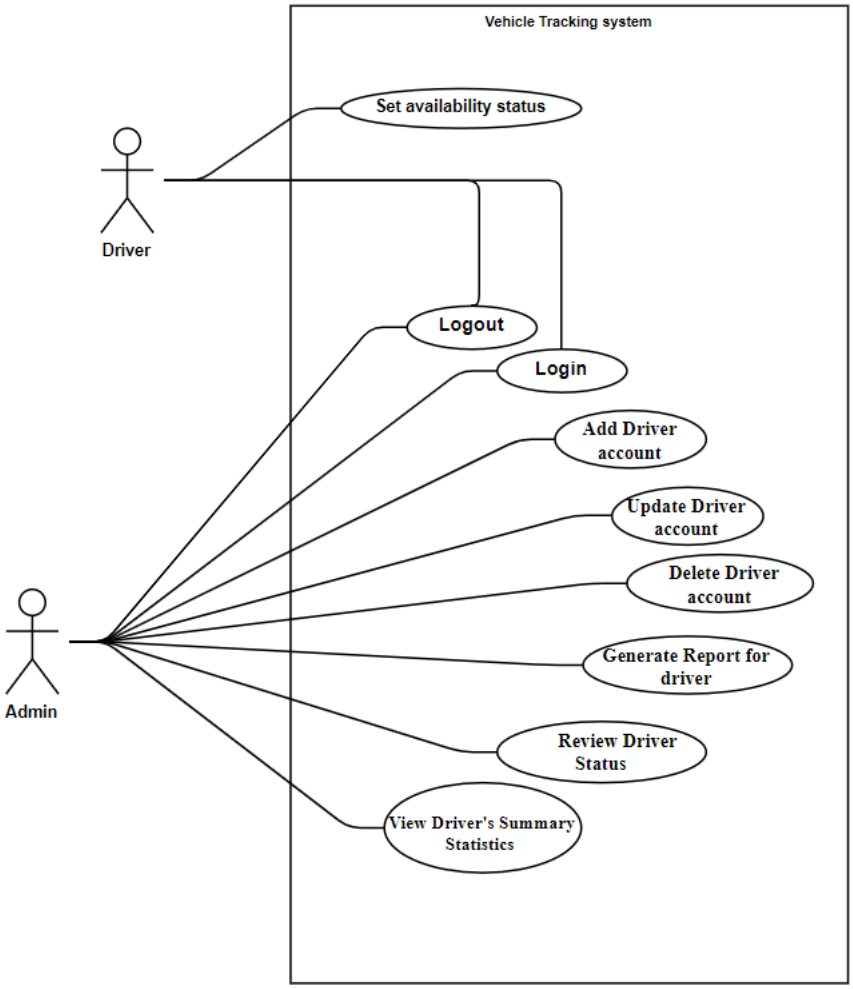


Figure 2 Use case diagram demonstrates the role or each Admin and driver.

2.4.5 Use case description

This section describes the variant of scenarios to do the functionalities of the system.

Use-case field	Description
Use case ID	UC1
Use Case Name	login
Trigger	This use case describes the scenario where the admin login into the system
Actors	admin
Preconditions	No precondition.
Main Course of actions	1- the admin asks the system to login 2- the system asks for admin information 3- the admin provides login information: A-ID B- password 4- the admin submits the login information 5- the system validates the login information 6- the system provides verification code field 7- the admin provides the verification code 8- the system validates the verification code 9- the system provides the home page
Alternate Courses	5.a: the admin provides invalid information. 5.a.1: the system asks the admin to provide valid information 5.a.2: the flow returns to step 2 9.a: the admin provides invalid verification code. 9.a.1: the system asks the admin to provide valid verification code 9.a.2: the flow retunes to step 7
Post Conditions	The home page is displayed

Use-case field	Description
Use case ID	UC2
Use Case Name	Add driver account
Trigger	This use case describes the scenario where the admin adds new driver account into the system
Actors	admin
Preconditions	- The admin is logged in
Main Course of actions	1- the admin asks the system to add driver 2- the system asks the admin to provide following info: (Full Name, Contact Information, Address, Driver's License Number) 3- the admin provides information 4- the admin submits the information. 5- the system validates the driver information 6- the driver info is valid 7- the system stores the driver info in database 8- the system shows success message 9- the system returns to the home page
Alternate Courses	6.a: the admin provides invalid information. 6.a.1: the system asks the admin to provide valid information 6.a.2: the flow returns to step 2
Post Conditions	The driver stored into the database

Use-case field	Description
Use case ID	UC3
Use Case Name	update driver account
Trigger	This use case describes the scenario where the admin update exist driver profile
Actors	admin
Preconditions	- The admin is logged in
Main Course of actions	1- the admin asks the system to update driver field 2- the system asks to provide the driver ID 3- the admin provides the driver ID 4- the system asks the admin to update following info: (Full Name, Contact Information, Address, Driver's License Number) 5- the admin provides information 6- the admin submits the information. 7- the system validates the driver information 8- the driver info is valid 9- the system stores the driver info in database 10- the system shows success message 11- the system returns to the home page
Alternate Courses	8.a: the admin provides invalid information. 8.a.1: the system asks the admin to provide valid information 8.a.2: the flow returns to step 4
Post Conditions	Update the driver field

Use-case field	Description
Use case ID	UC4
Use Case Name	delete driver account
Trigger	This use case describes the scenario where the admin deletes driver account
Actors	admin
Preconditions	- The admin is logged in
Main Course of actions	1- the admin asks the system to delete driver field 2- the system asks to provide the driver ID 3- the admin provides the driver ID 4- the system validates the driver 5- the driver ID is exist 6- the system asks the admin to confirm the request 7- the admin confirms the deletion 8- the system stores the driver info in archive 9- the system removes the driver field from database 10- the system shows success message 11- the system returns to the home page
Alternate Courses	5.a: The driver ID is not exist. 5.a.1: the system asks the admin to provide valid ID 5.a.2: the flow returns to step 2
Post Conditions	delete the driver field

Use-case field	Description
Use case ID	UC5
Use Case Name	Generate report for driver
Trigger	This use case describes the scenario where the admin Generate report for driver
Actors	admin
Preconditions	- The admin is logged in
Main Course of actions	1- the admin asks the system to generate driver report 2- the system asks to provide the driver ID 3- the admin provides the driver ID 4- the system validates the driver 5- the driver ID is existed 6- The system presents a list of available report options. These options typically include: a- Driver Performance Report b- Total Rides Report c- Revenue Report and choose the period time. 7- the admin selects the report and provide the period 8- the period is valid 9- the system shows the report
Alternate Courses	5.a: The driver ID is not existed. 5.a.1: the system asks the admin to provide valid ID 5.a.2: the flow returns to step 2 9.a the period is invalid. 9.a.1 the system shows indicated message 9.a.2 the flow return into step 6
Post Conditions	Generate the report

Use-case field	Description
Use case ID	UC7
Use Case Name	Review driver status
Trigger	This use case describes the scenario where the admin review driver status, to detect any dishonest actions by drivers, such as logging in without being available for rides.
Actors	admin
Preconditions	- The admin is logged in
Main Course of actions	1- the admin asks the system to review the driver's status 2- the system displays the list of drivers along with their login status 3- the admin filters the drivers list based on the status chosen 4- The system provides the admin with the ability to set rules for detecting dishonest actions, such as: a- Alert when a driver logs in without setting availability for rides. b- Alert when a driver frequently switches between "Available" and "Unavailable" status within a short time frame. 5- The system notifies the administrator of the detected dishonest action through a real-time notification. 6- Based on the information provided, the administrator may take appropriate actions, such as: a- contacting the driver. b-initiating an investigation.
Alternate Courses	None
Post Conditions	Provide a list of drivers based on their status

Use-case field	Description
Use case ID	UC8
Use Case Name	Review driver statistics
Trigger	This use case describes the scenario where the admin review driver statistics such as total active drivers, total rides completed, ongoing rides.
Actors	admin
Preconditions	- The admin is logged in
Main Course of actions	1- the admin asks the system to review the driver's statistics 2- the system asks to provide driver ID 3- the admin provides admin ID 4- the system validates the driver ID 5- the driver ID is existed 6- the system provides 3 sections including: a- Total rides completed. b- ongoing rides c- total active drivers 7- the admin chooses one of the sections 8- the system shows the related information
Alternate Courses	5.a: The driver ID is not existed. 5.a.1: the system asks the admin to provide valid ID 5.a.2: the flow returns to step 2
Post Conditions	Provide the statistics for the admin

Use-case field	Description
Use case ID	UC9
Use Case Name	Logout
Trigger	This use case describes the scenario where the admin logout from the system.
Actors	admin
Preconditions	- The admin is logged in
Main Course of actions	1- the admin asks the system to logout 2- the system asks the admin to confirm the logout 3- the admin confirms the logout 4- the system flags the admin as logged out 5- the system logs the admin out of the system 6- the system shows success message 7- the system redirects to login page
Alternate Courses	None
Post Conditions	Logout from the system

Use-case field	Description
Use case ID	UC10
Use Case Name	Set availability status
Trigger	This use case describes the scenario where the driver to set the availability status.
Actors	driver
Preconditions	- The driver is logged in
Main Course of actions	1- the driver asks the system to set the availability status 2- the system asks the driver to choose of the status including: a- available b- un-available 3- the driver chooses one of the statuses 4- the system flags the driver as chooses status 5- the system updates the driver status 6- the system shows success message
Alternate Courses	None
Post Conditions	Set the status for the driver

2.4.6 User interface

2.4.6.1 Admin interface

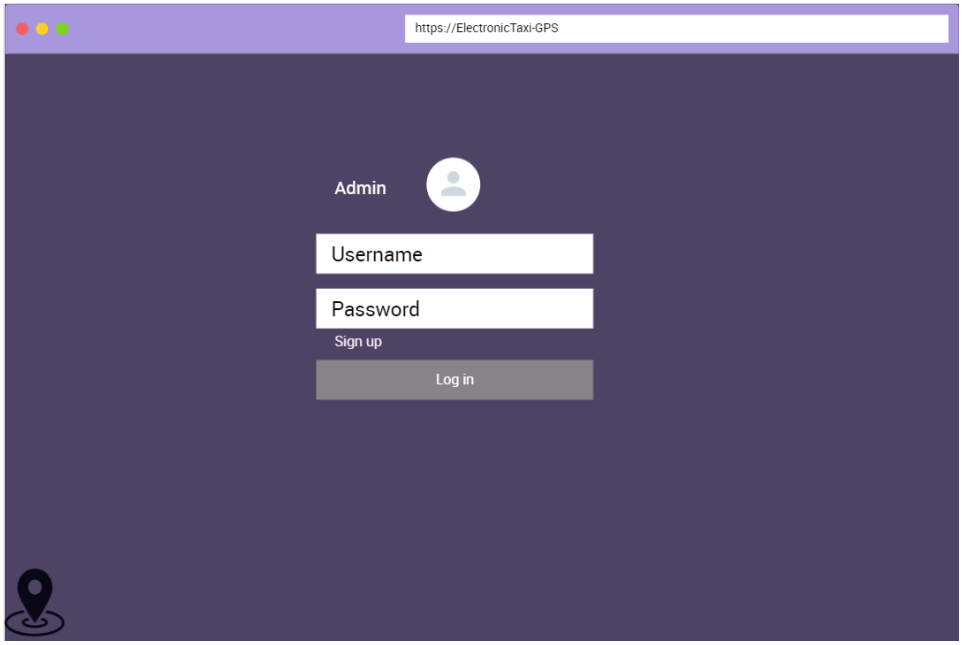


Figure 1 This is admin login page.

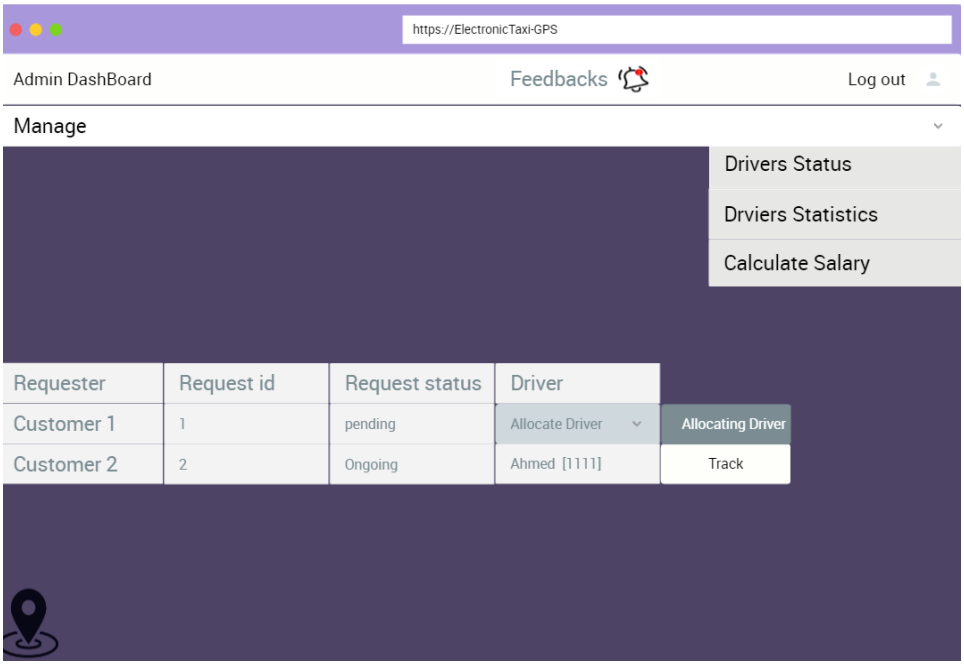


Figure 2 Admin's Dashboard where he can see the requests, allocate drivers, track ongoing requests.

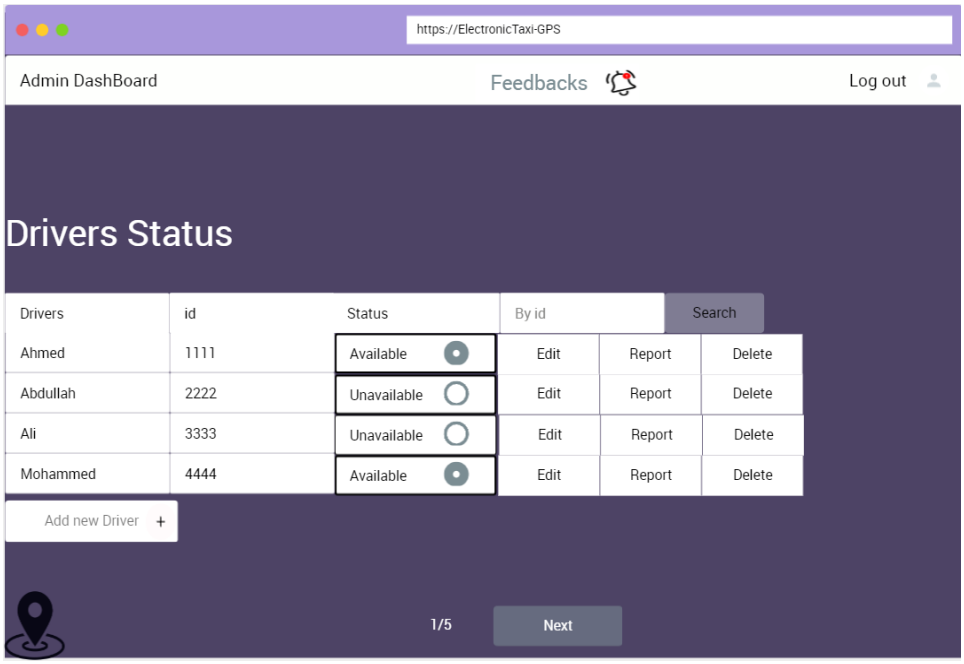


Figure 3 Admin can review Status, Edit, Report, Delete, Search by id.

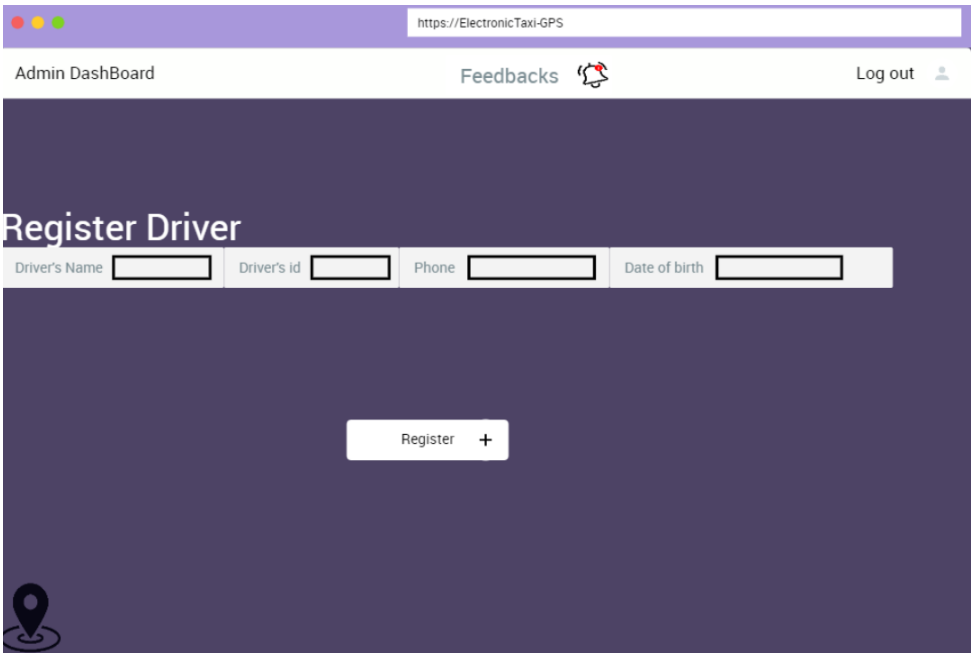


Figure 4 Admin can Register a new Driver.

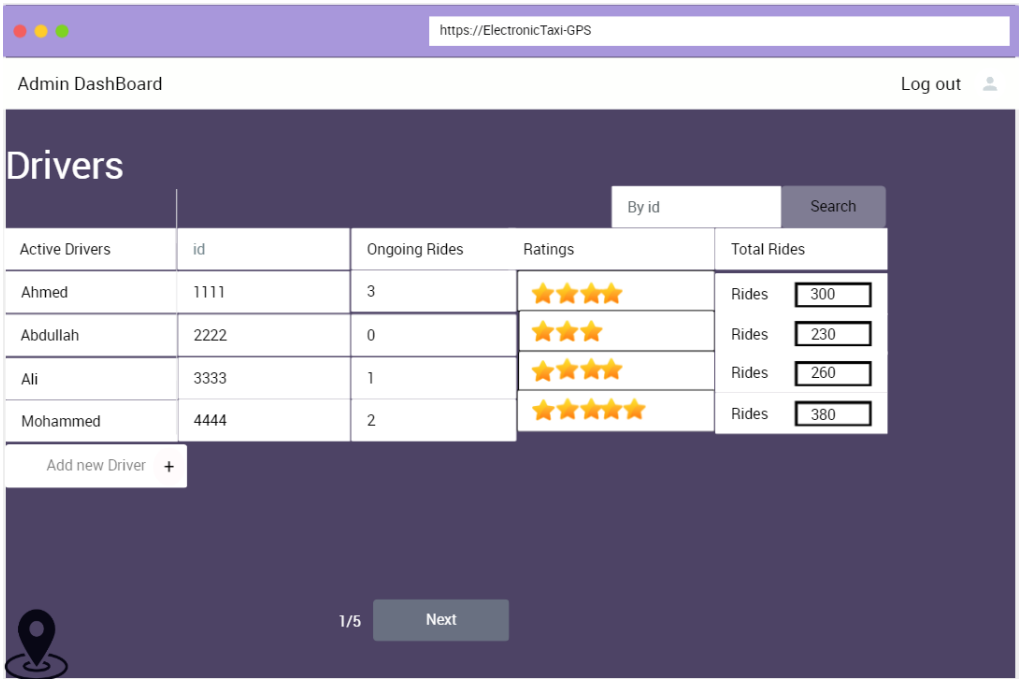


Figure 5 Admin can view all active driver's summary Rides, Ratings, Ongoing rides.

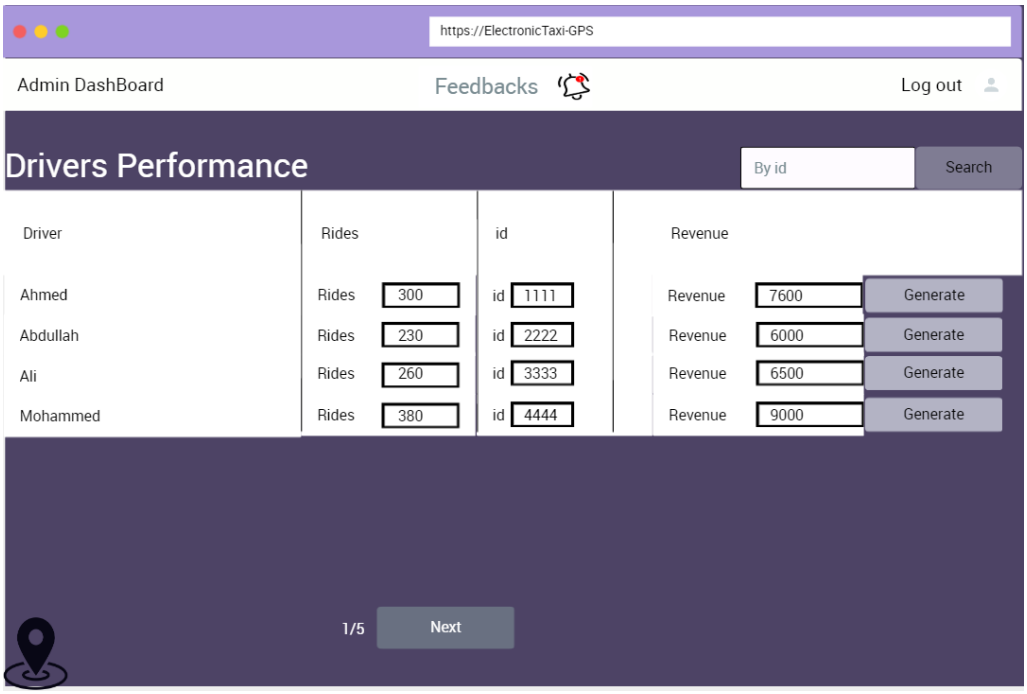


Figure 6 Admin can view the Performance of Drivers performance Rides, Revenue, Generate report.

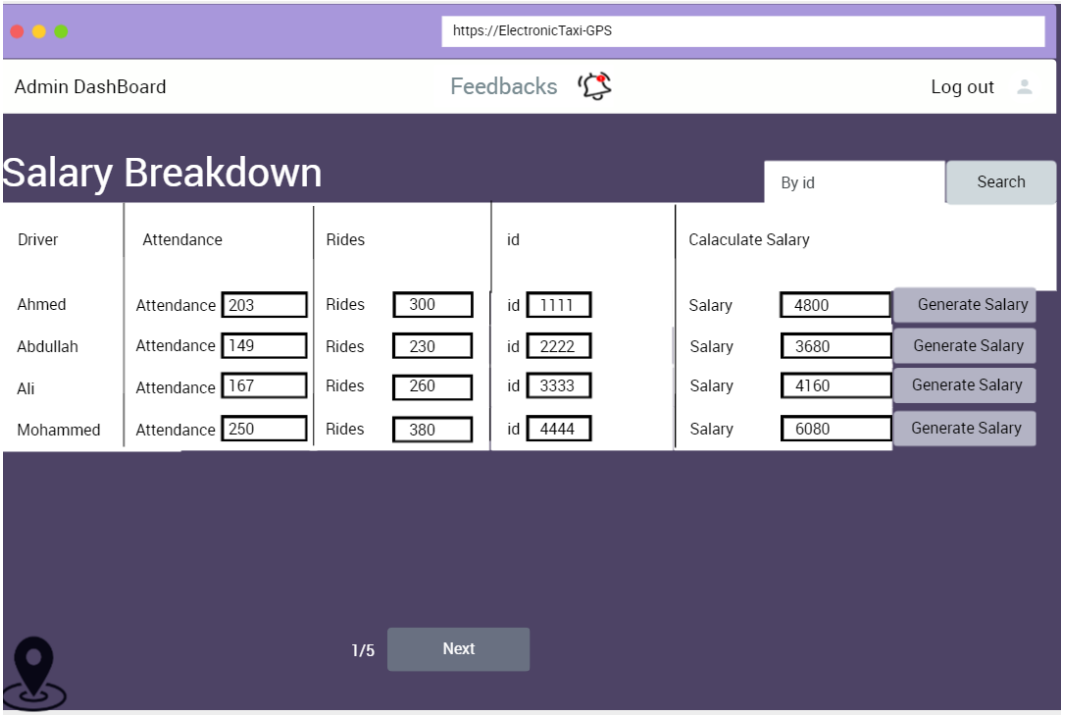


Figure 7 Admin can view the attendance, Salary breakdown for drivers, calculate, Generate Salary.

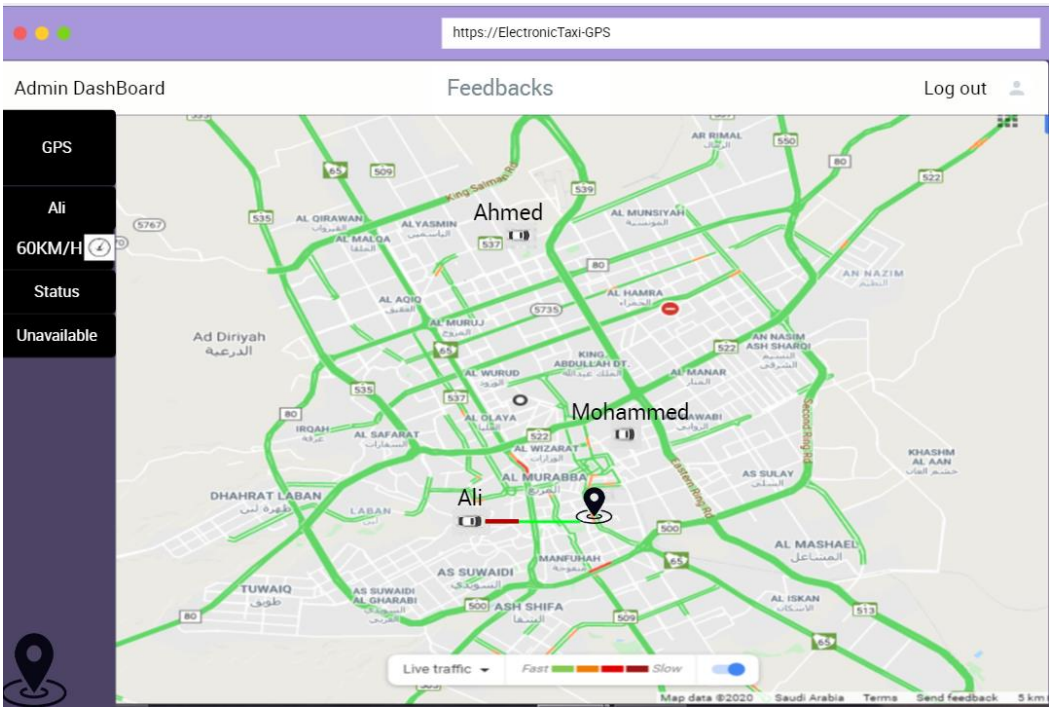


Figure 8 Admin can view the Location of Drivers Speed, Status.

3 Non-Functional Properties

3.1 Reliability:

Reliability is paramount in a Vehicle Tracking System because it ensures that the system consistently provides accurate and dependable location information. Reliability is crucial for safety, as administrators and users must trust the system's data for decision-making.

3.2 Availability:

Availability is essential because the system needs to be operational 24/7. Vehicle tracking is often a mission-critical service, and downtime can result in significant disruptions.

3.3 Portability:

Portability is vital for a Vehicle Tracking System because it ensures that the system can be easily accessed and used on various devices and platforms. This flexibility is essential for accommodating different GPS devices in vehicles and providing administrators and users with the freedom to access the system from their preferred devices.

3.4 interoperability:

Interoperability is critical for a Vehicle Tracking System because it enables seamless communication and data exchange between different hardware and software components. This ensures that the system can work cohesively with various GPS devices, mapping services, and third-party applications. Interoperability simplifies integration efforts, allowing for the use of diverse tools and technologies while maintaining system functionality. It also promotes collaboration and data sharing among different stakeholders.

3.5 Security:

Security is a critical quality attribute as the system deals with sensitive location data and user information. Unauthorized access or data breaches can have severe consequences. Implementing robust security measures, including encryption and user authentication, is essential to protect the system.

3.6 Usability:

Usability is essential for both administrators and end-users. A user-friendly interface and intuitive design make it easier for administrators to manage the system and for end-users to access location information. Usability can reduce the learning curve and improve overall satisfaction.

4 Quality Assurance

4.4 Reviews

The type of the reviews that will be made is software peer review:

1- Code Review: Source code will be investigated, and we will check for bugs/errors, and will be removed. This code review will take place after the implementation phase.

2- Walkthrough reviews: it's useful for higher level documents like requirement specifications, and to achieve common understanding and gather feedback.

4.5 Verification

1- Testing: helps to ensure that the software meets the specified requirement. It aims to make certain that the software meets its intended or required functionality.
2- Unit testing: it will be used for individual modules of the SW. It's to ensure that it meets the requirements & specifications. Write different scenarios with different inputs and expected outputs like a test case for individual functions.

3- Integrated testing: will be useful for the interaction between software modules to detect error/defects from different modules.

4.6 Validation

To establish the confidence for the client in terms of requirements, security, reusability and expandability, and performance are met we would use validation testing:

1- User Acceptance testing
2- Security testing
3- Performance testing
4- compatibility testing

4.4 Acceptance criteria

4.4.1 Procedures: We will provide the user step-by-step clear instructions/installation on how to install the software. The installation is simple with no ambiguity.
4.4.2 Testing: The software should be tested to ensure that the functionality and requirements meet the user needs.
4.4.3 Training: The software provider must offer comprehensive training sessions to users (both drivers and administrators) covering system usage, navigation, key features.

4.4.4 Documentation: technical documentation should be provided user manual, admin manual, and installation guide, The user manual should detail the functionalities and operations of the software, The admin manual should provide guidance on system setup, configuration, and administrative tasks.

5 Future Considerations

Our system is designed with future scalability, flexibility, and maintainability in mind. Key parts include the Event Processor, the database (Event Storage), and the Event Channels. The system can handle an increase in user volume, addition of new features, and integration with other systems. A potential risk lies in the reliance on the GPS API, which can be mitigated with fallback mechanisms and thorough testing. Regular review and testing will ensure the system's continued effectiveness and reliability.

SWE321:
SYSTEM DESIGN DOCUMENT

PART 3

Project

Vehicle Tracking System

SYSTEM ARCHITECTURE DOCUMENT

Version 3

9/28/2023

Anas Alsubaie
Lead Software Engineer

1.Introduction:

This System Design Document (SDD) is a detailed guide for implementing our project. It refines the system architecture, provides a structural model, and outlines the dynamic behavior of the system, serving as a practical guide for developers.

2. Detailed Design:

2.1 Detailed System Architecture

2.1.1 Component diagram

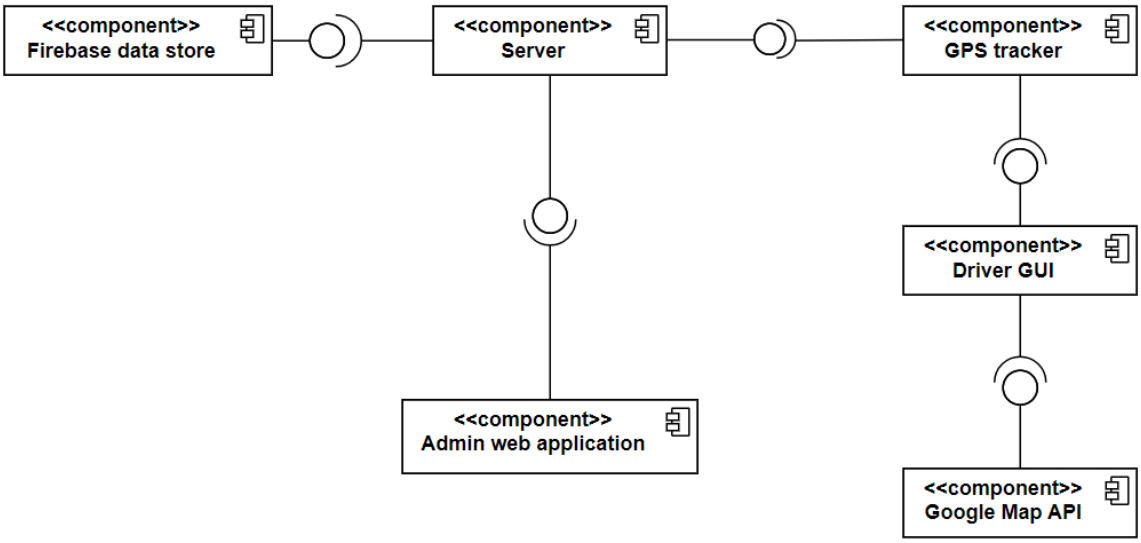


Figure 1 This figure illustrates the Components of the system and their interactions.

x

Component Name:	**Admin Web Application**
Description:	A web-based interface dedicated for administrators to manage the vehicle tracking system, providing functionalities to monitor vehicle locations, manage users, and execute administrative tasks
Properties/data:	• **Admin data:** The ID and credentials of each administrator. • **Vehicle data:** The ID and real-time location data of each vehicle. • **Historical location data** • **Driver data:** The ID, credentials, status, and attendance of each driver.
Behavior/functionality:	• **Manages admin accounts:** Allows administrators to log in. • **Admin Authentication:** Verify admin credentials upon login. • **Dashboard Display:** Overview of all vehicles and their current locations. • **Historical Data View:** Allows selection of date/time range to view past vehicle locations. • **Manages driver data:** Allows admin to add, update, and delete driver data, and monitor driver status. Manage driver credentials and monitor attendance. • **Notifications & Alerts:** View alerts related to any anomalies or issues with vehicle tracking. • **Report Generation:** Generate reports for vehicle movement, driver attendance.
Connectors & Interface:	• **Interaction with the Server:** Fetching and sending data. • **Web Browser:** For the graphical user interface display and interactions. • **Database:** To request and store data.
Dependencies:	• **Server:** For fetching, storing data, process requests and manage system operations. • **Internet connection:** For real-time data updates and communications with the server. • **Browser**: Modern web browsers that support the latest web standards (like Chrome) • **Database:** Subsystem to store and retrieve data.
Resources:	• **Web server:** To host the web application. • **Database Connection:** For fetching and updating data. • **Web Development Frameworks** • **Network resources:** To connect to the Server and Database.

Component Name:	**Driver Application**
Description:	A mobile application tailored for drivers, allowing them to view their assignment and navigate routes, see their current location, and undertake necessary operations pertaining to their tasks, update their status, and check in and out of their vehicles.
Properties/data:	• **Driver Data: The ID, credentials, and status of each driver.** • **Current location data** (latitude, longitude) • **Planned route details.** • **Notifications and alerts for new assignment**
Behavior/functionality:	• **User Authentication**: Authenticate driver upon login. • **Driver logs in/out:** Allows drivers to log in and log out of their accounts. • **Current Location Display**: Show the driver's real-time location using a map interface. • **Route Navigation**: Display the planned route, provide turn-by-turn navigation, and estimated arrival time. • **Notifications & Alerts**: Receive notifications about route changes, traffic updates, or system-related information. • **Feedback/Reporting**: Allows the driver to send feedback or report issues. • **View task/assignment:** Lets drivers view their current tasks or assignments. • **Update status:** Allows drivers to update their status (e.g., available, unavailable, on-duty, etc.). • **Shift check-in/check-out:** Lets drivers check in when they start their shift and check out when they finish.
Connectors & Interface:	• **Interaction with the Server**: Fetching and sending data. • **GPS Tracking**: To provide real-time location data. • **APIs:** (Google Maps) For maps and navigation.
Dependencies:	• **GPS Tracking**: To determine the real-time location. • **Depend on the Server:** to process requests and manage system operations.
Resources:	• **Mobile device:** To run the driver application. • **Network resources:** To connect to the Server Subsystem and GPS Tracking Subsystem.

Component Name:	**Server**
Description:	The Server component is a backend service that processes requests and manages operations for the entire system, including the Manager Application and Driver Application.
Properties/data:	• **Raw GPS data input queue** • **Processed location data** • **Connection configurations for the database and other services** • **User data requests:** Requests related to both manager and driver data, which encompasses user IDs, credentials, roles, status (active, inactive), user profiles, and any other data that is specific to each user. • **Task/Assignment data requests:** Pertaining to task or assignment details, including IDs, descriptions, associated drivers, status, timestamps, and location data. • **Route data requests:** Related to generated routes for each assignment, including travel distance, estimated travel time, and actual travel time. • **Log data requests:** Involving logging of system activity, such as user logins and logouts, state changes of assignments, and system errors. • **Notification data requests:** Pertaining to notifications sent to managers and drivers, including the content and status of notifications. • **Feedback data requests:** If applicable, requests related to the system's feedback mechanism, such as feedback IDs, associated users, feedback content, and timestamps.

Behavior/functionality:	• **Data Ingestion**: Accept and queue raw GPS data from trackers. • **Data Processing**: Convert raw GPS data into actionable location information. • **Data Storage**: Store processed location information in the database. • **Data Retrieval**: Fetch location and route details in response to requests from the interfaces. • **Authentication & Authorization**: Verify credentials and permissions of Admin and Driver interfaces. • **Notifications & Alerts Handling**: Send out notifications or alerts based on certain triggers or events. • **Processes requests:** Handles requests from the Manager and Driver Applications, such as logins, updates, and assignments. •
Connectors & Interface:	• **Database Connection**: To store and retrieve data. • **API Endpoints**: Exposed endpoints for communication with external interfaces. • **Communication protocols** for receiving data from GPS trackers. • **Manager Application:** To receive and send data to. • **Driver Application:** To receive and send data to. • **Database Management System:** To send data requests to and receive responses from. • **Google Maps API:** To calculate routes and estimated times of arrival.
Dependencies:	• **Database (Firebase**): To persistently store and retrieve data. • **Internet connection**: For external communications. • **Depend on the Manager and Driver Applications** • **Depend on the Google Maps API for geolocation services.**
Resources:	• **Server Hardware** • **Server Operating System and Middleware** • **Backend Development Frameworks** (e.g., Node.js, Django, Flask) • **Network resources.**

Component Name:	**GPS Tracker**
Description:	The GPS Tracker is a component that continuously monitors and updates the real-time geographical location of drivers. It interacts with the GPS of the driver's device to determine their current location and communicates this information to the Server Subsystem.
Properties/data:	• **Current location:** The real-time geographical coordinates (latitude and longitude) of the driver. • **Timestamp of the last data sent.** **Update frequency:** The frequency at which the GPS Tracker updates the driver's location.
Behavior/functionality:	• **Monitors location:** Interacts with the GPS hardware of the driver's device to continuously monitor the driver's real-time location. • **Data Transmission**: Sends the gathered GPS data to the server at regular intervals or upon request. • **Update location**: Sends updates on the driver's location to the Server Subsystem at regular intervals determined by the update frequency. • **Self-diagnostics**: Periodic checks to ensure the tracker is functioning properly.
Connectors & Interface:	• **Wireless communication interface** (e.g., 4G, 5G, or satellite communication) to transmit data. • **Driver's GPS hardware:** To monitor the driver's real-time location. • **Server Subsystem:** To send updates on the driver's location.
Dependencies:	• **Server Subsystem**: Depends on the Server Subsystem to receive and process location updates. • **Driver's GPS hardware:** Depends on the GPS hardware of the driver's device to provide accurate, real-time location data.
Resources:	• **GPS hardware:** The GPS Tracker needs access to the GPS hardware of the driver's device to determine the driver's location. **Network resources:** To communicate with the Server Subsystem.

Component Name:	**Google Maps API**
Description:	**A third-party service provided by Google that allows the conversion of raw GPS coordinates into human-readable addresses, often termed as "reverse geocoding".**
Properties/data:	• **API Key:** The unique identifier used to authenticate requests associated with the project for usage and billing. • **Rate limits**: Number of requests allowed in a specific time frame. • **Quotas:** Total allowed requests in a billing period.
Behavior/functionality:	• **Processes requests:** Receives requests from the Server Subsystem, processes these requests, and returns the requested data (like routes, distances, estimated travel times). • **Rate limiting:** It manages the number of requests that can be made within a certain timeframe to ensure the service's availability and prevent abuse.
Connectors & Interface:	• **Server Subsystem:** Receives requests from and sends responses to.
Dependencies:	• **Internet connection**: For accessing the Google Maps API service. • **Server Subsystem**: Depends on the Server Subsystem to send requests for data. •
Resources:	• **Documentation for the Google Maps API, detailing endpoints, request/response formats, and best practices**. • **API Key:** The Google Maps API requires an API Key to authenticate requests. • **Network resources:** To communicate with the Server Subsystem.

Component Name:	**API's**
Description:	**A software layer facilitating communication between the server and various interfaces, ensuring data flow in a structured and secured manner.**
Properties/data:	• **API Endpoint URLs** (Different endpoints for different functionalities) • **Request/Response format** (e.g., JSON, XML) • **Authentication tokens or API keys** • **Rate limits and quotas for different interfaces or users**
Behavior/functionality:	• **Data Transmission**: Sends and receives data between server and interfaces. • **Authentication & Authorization**: Validates requests and determines access levels. • **Data Validation**: Checks the integrity and structure of incoming data. • **Error Handling**: Provides appropriate responses in case of errors or invalid requests. • **Integration with Third-party Systems**: Facilitates interactions with other systems, e.g., Google Maps API.
Connectors & Interface:	• **HTTPS protocol for secured communication.**
Dependencies:	• **Internet connectivity for external API interactions**. • **Server: To fetch or save data**. • **External APIs (like Google Maps API): For specific functionalities.**
Resources:	• **API documentation detailing endpoints, methods, request/response formats, and examples.**

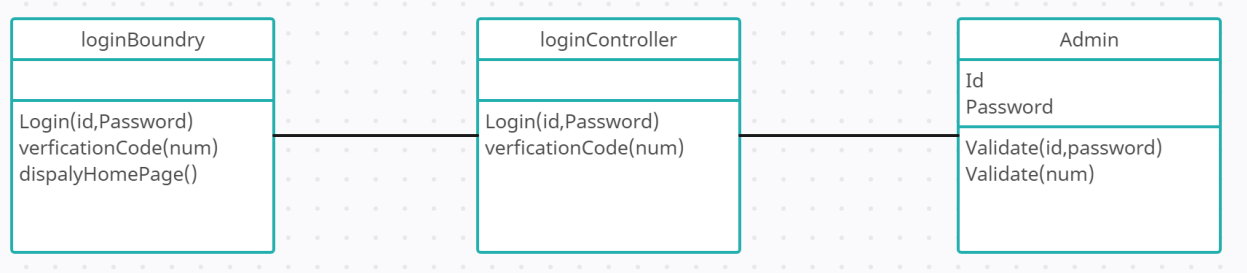
2.2 Detailed Structural Model:

2.2.1 Detailed Object Diagram

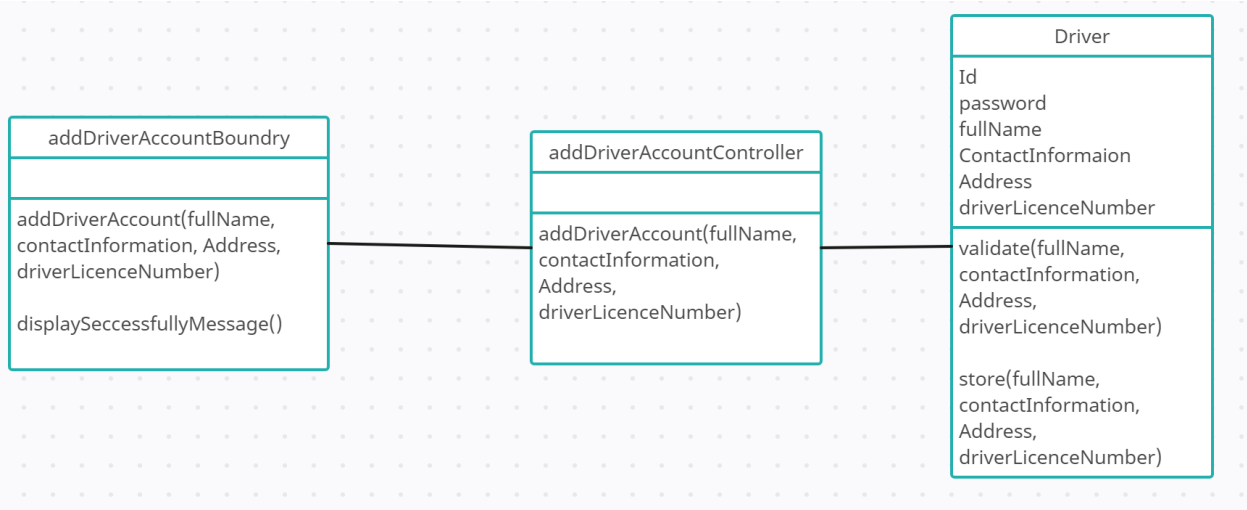
2.2.2 Detailed Analysis diagram

2.2.3 VOPC

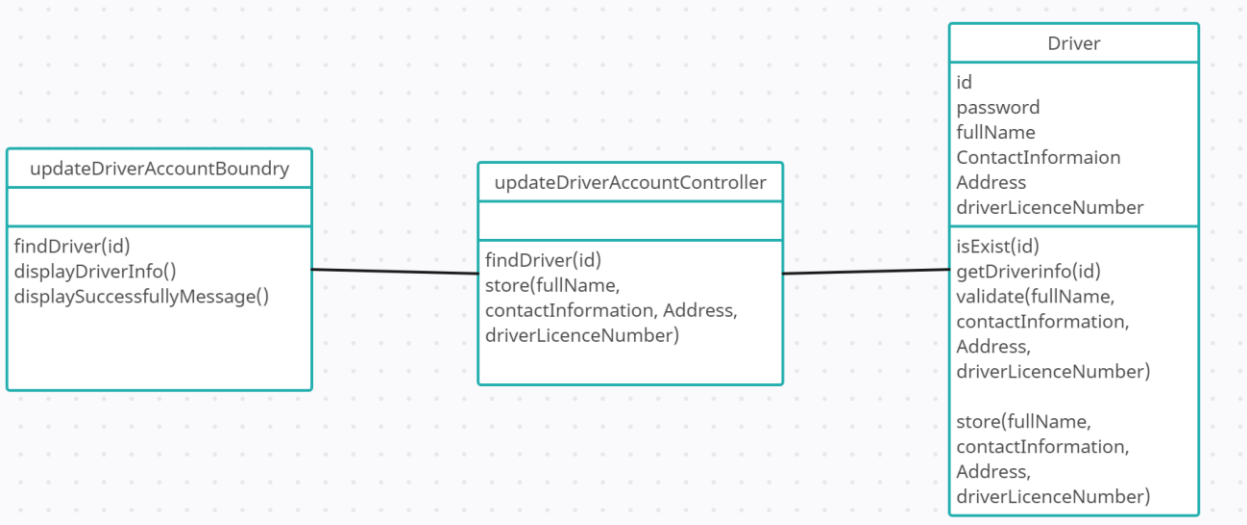
2.2.3.1 Login



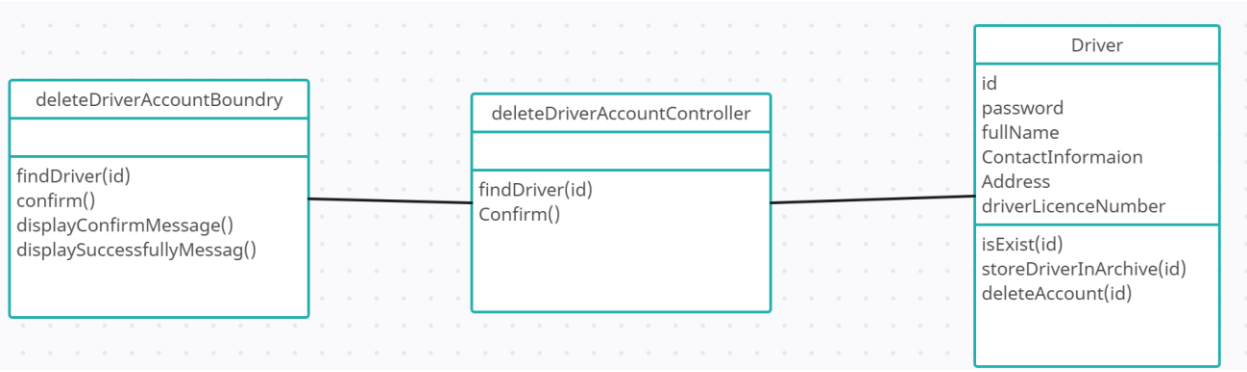
2.2.3.2 Add driver account



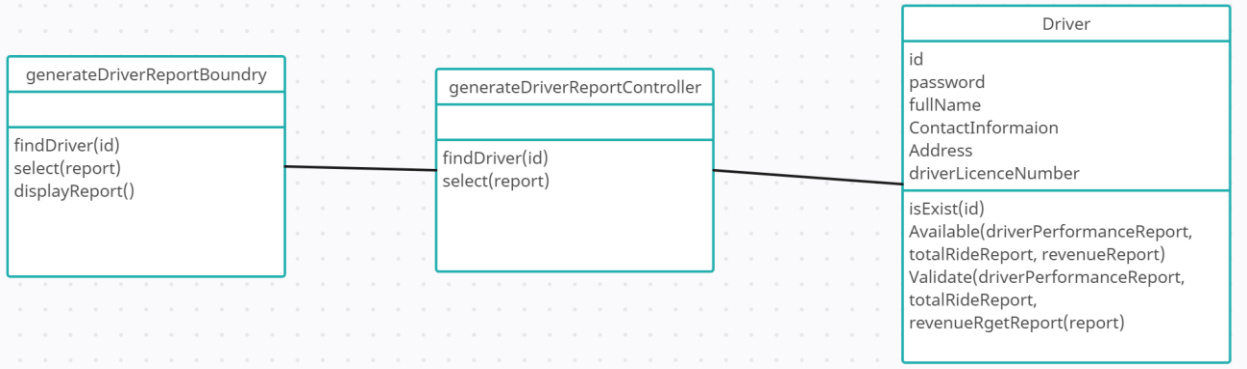
2.2.3.3 Update driver account



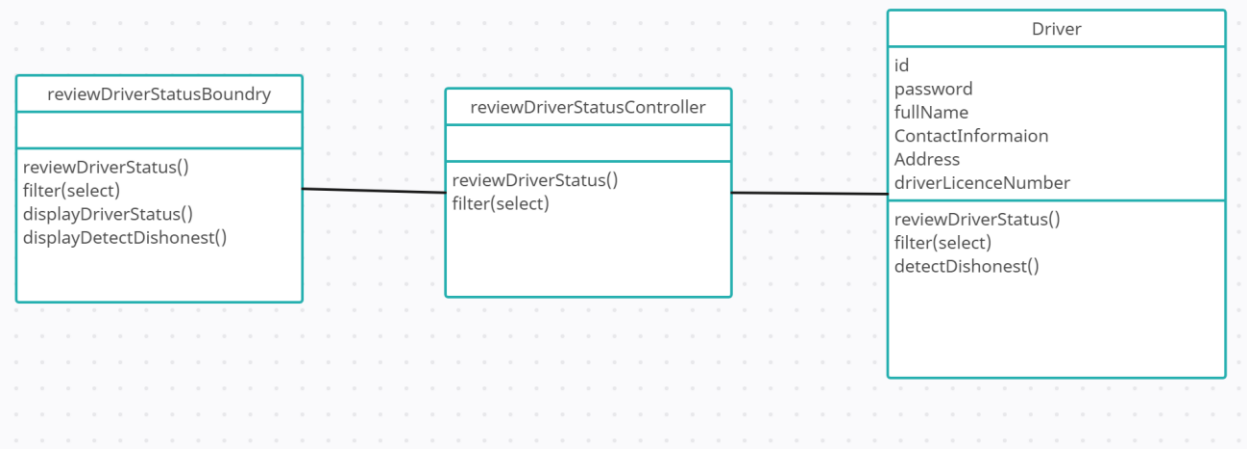
2.2.3.4 Delete driver account.



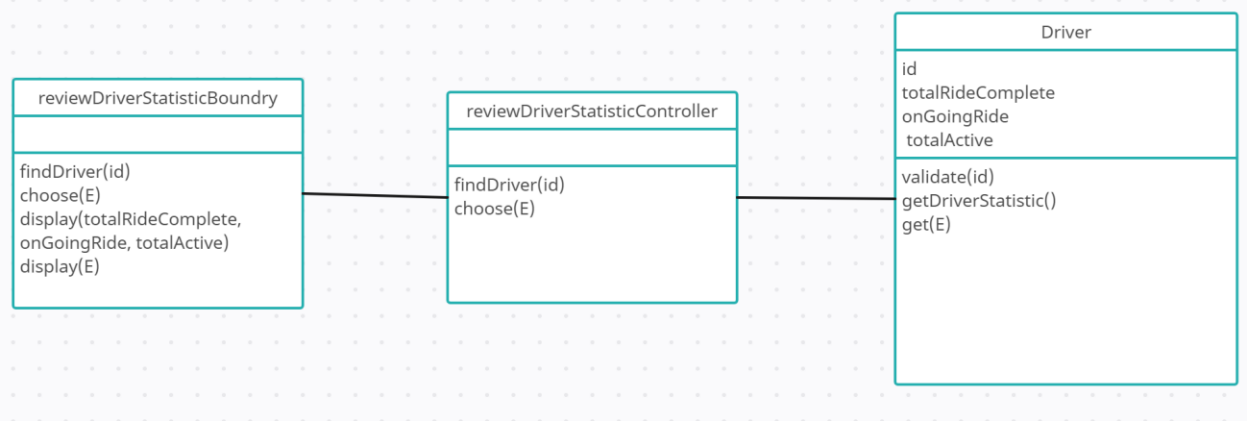
2.2.3.5 Generate driver report



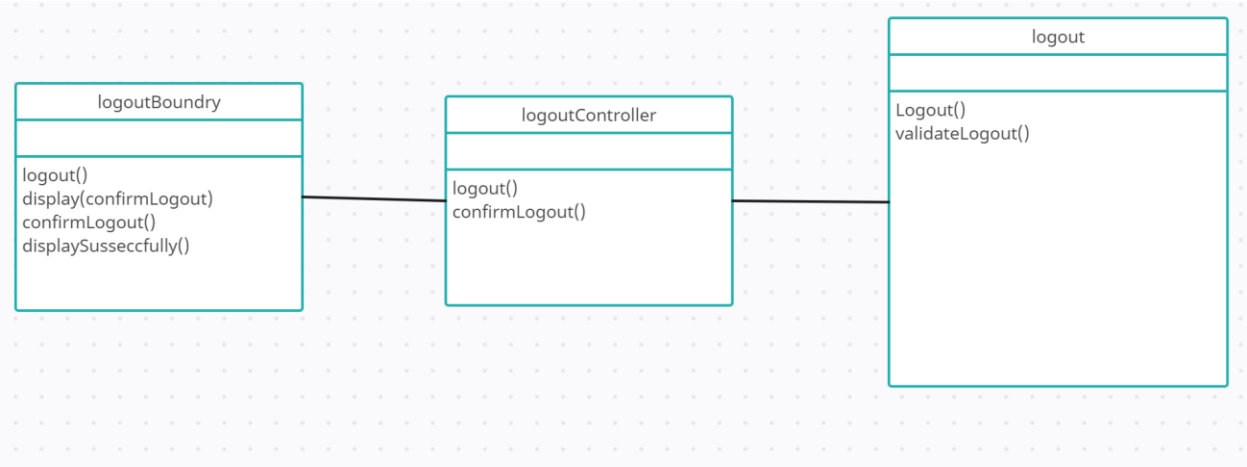
2.2.3.6 Review driver status



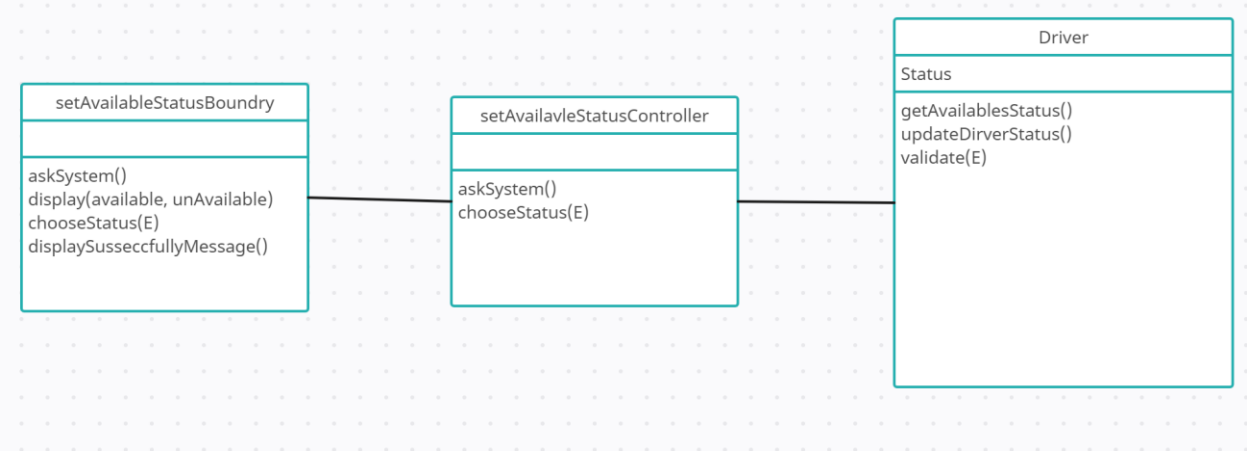
2.2.3.7 Review driver statistic



2.2.3.8 Logout



2.2.3.9 Set available status

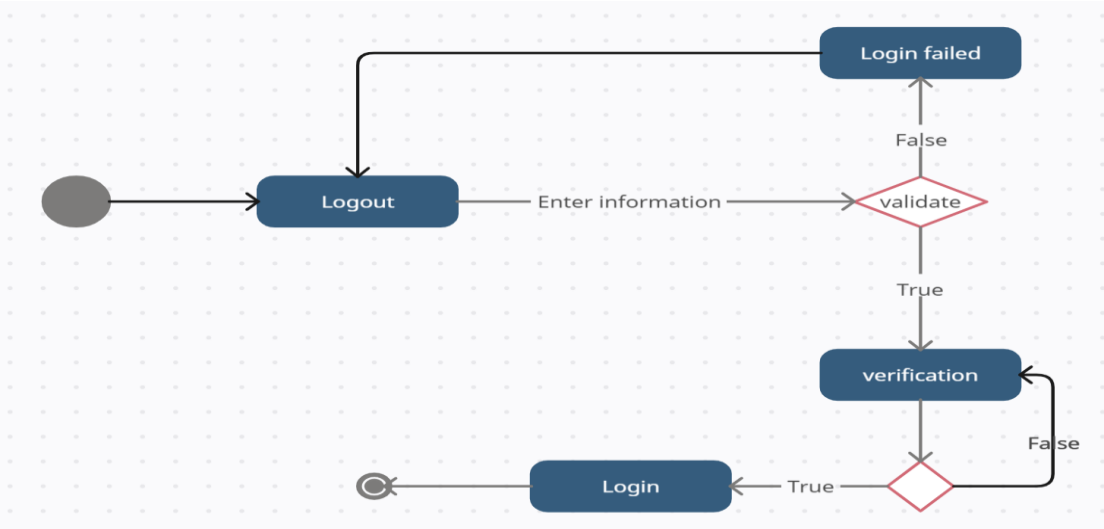


3.3 Dynamic Model:

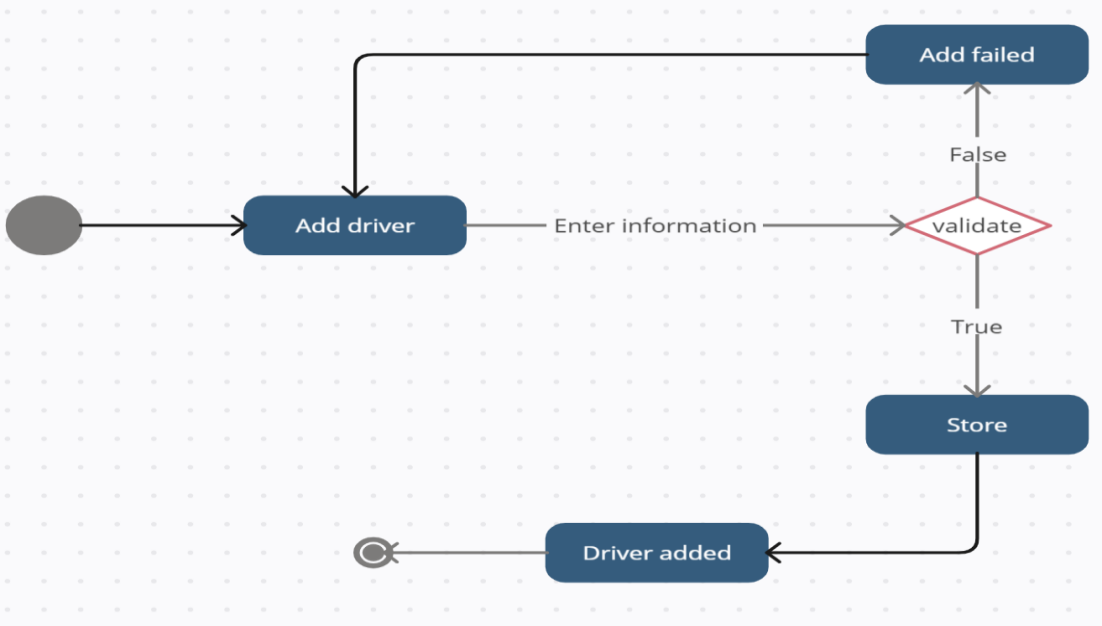
Refine system behavior in phase 2. Describe behavior of the system

3.3.1 State Diagram

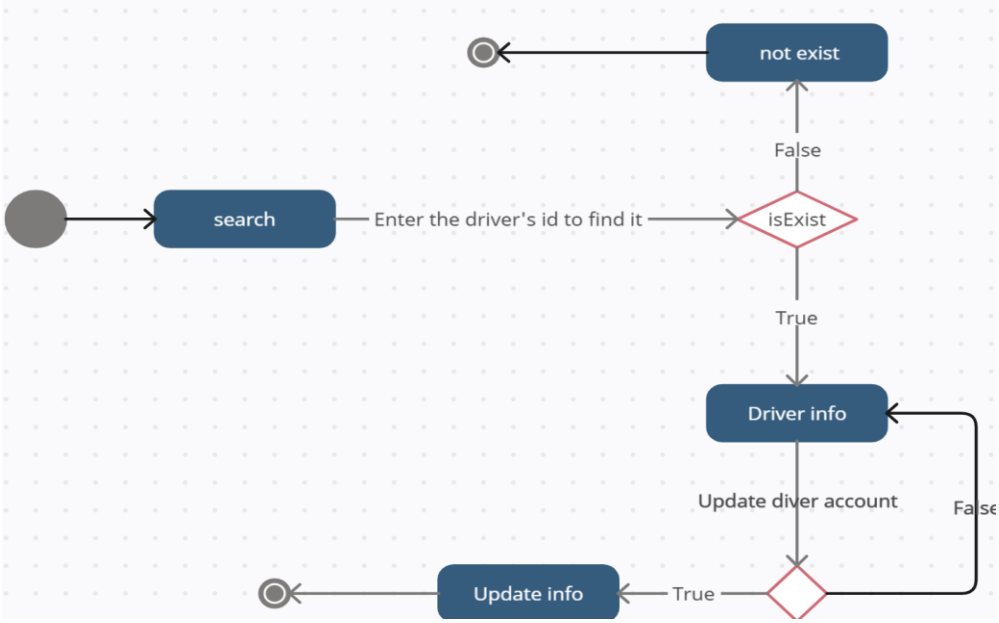
3.3.1.1 use case : Login



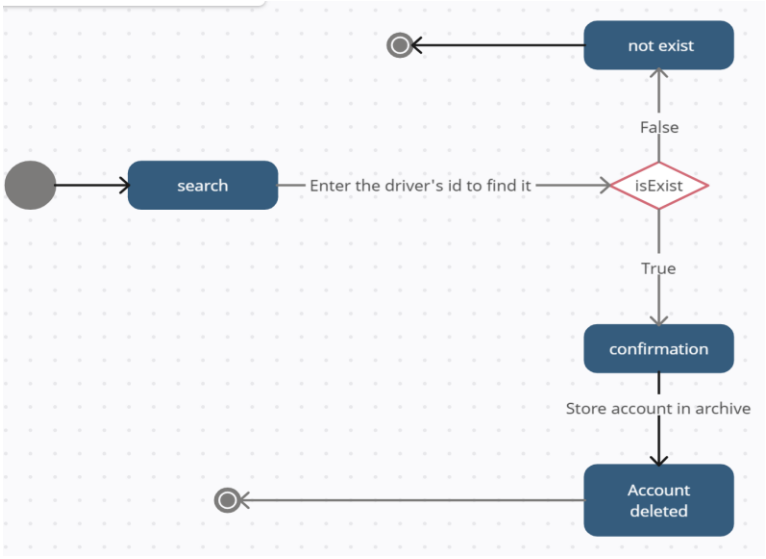
3.3.1.2 Use case : Add driver.



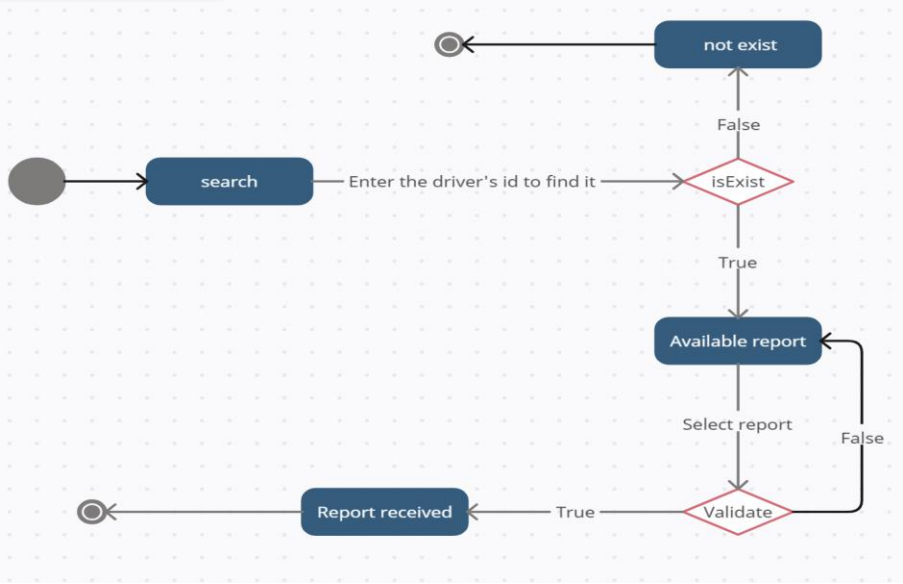
3.3.1.3 Use case : Update driver account



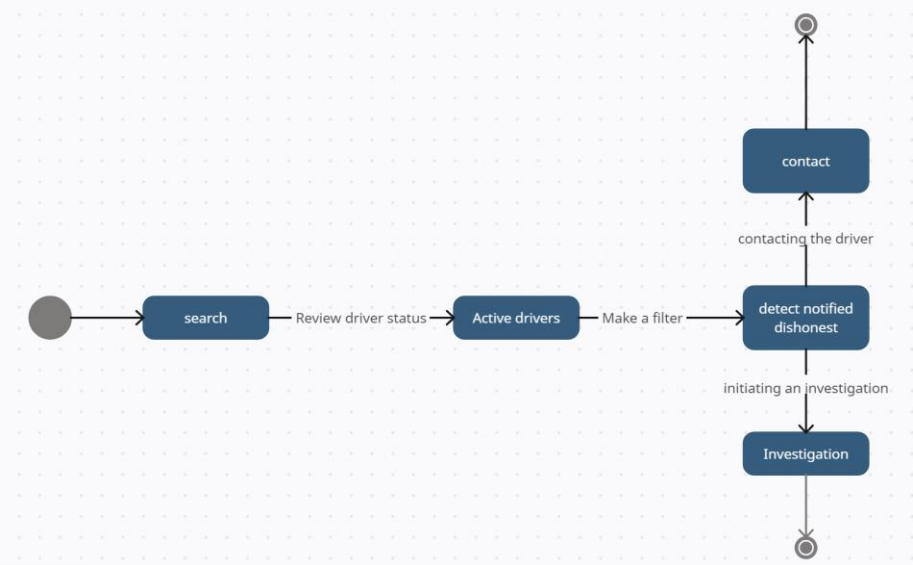
3.3.1.4 Use case : Delete driver account



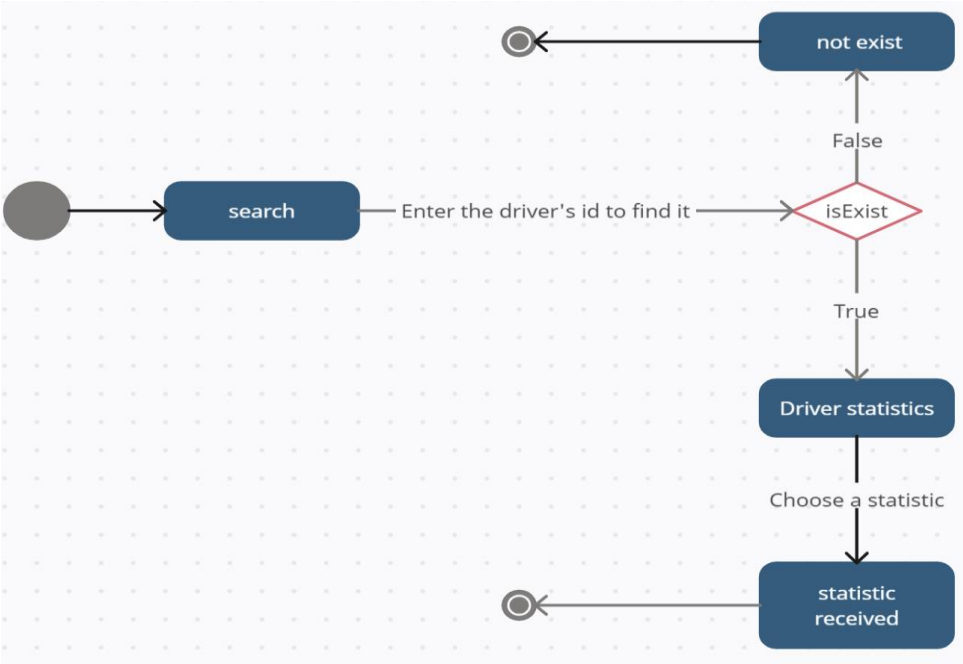
3.3.1.5 use case: Generate Driver Report



3.3.1.6 Use Case: Review Driver Status

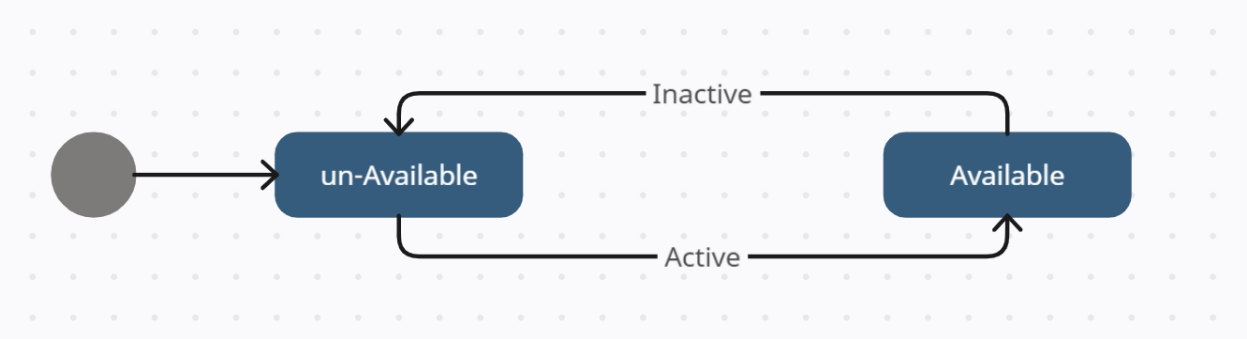


3.3.1.7 Use Case: Review Driver Statistics



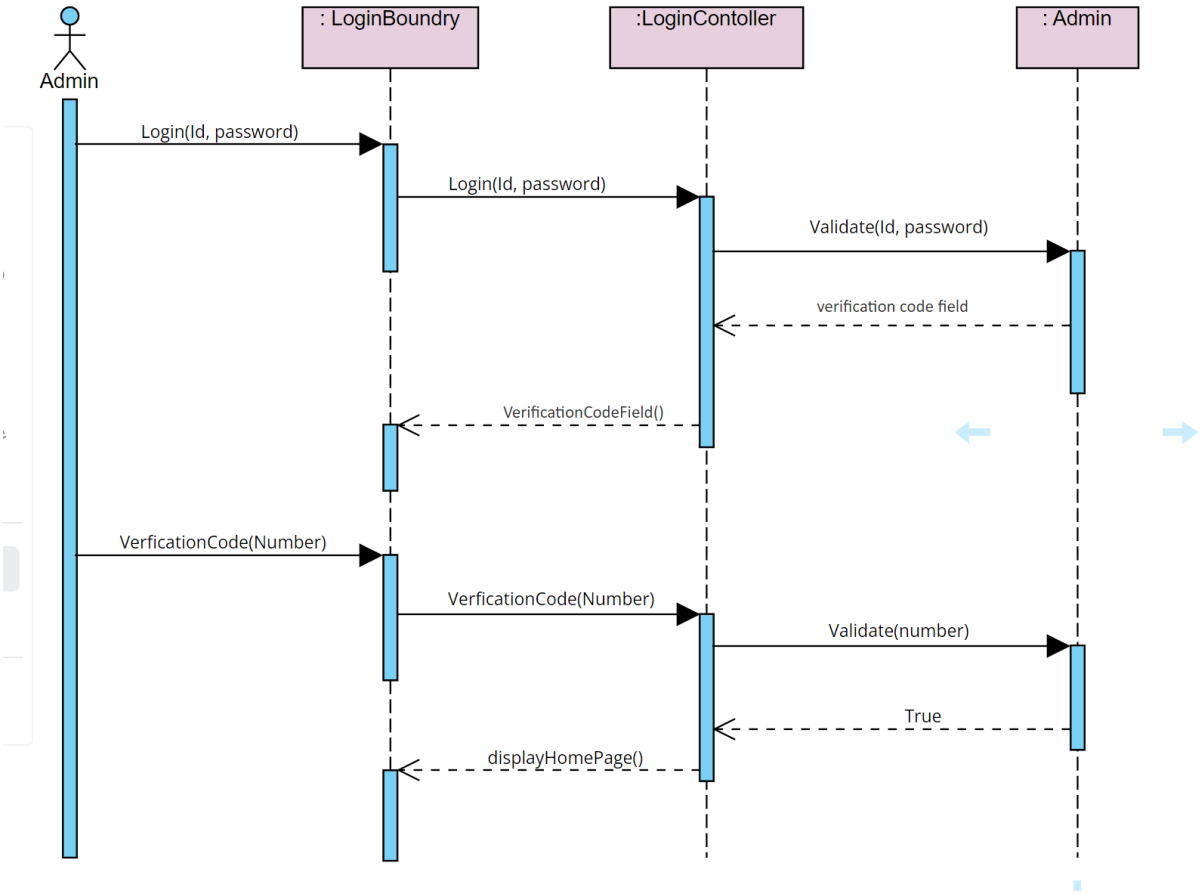
3.3.1.8 Use Case: Logout

3.3.1.9 Use Case: Set Available Status

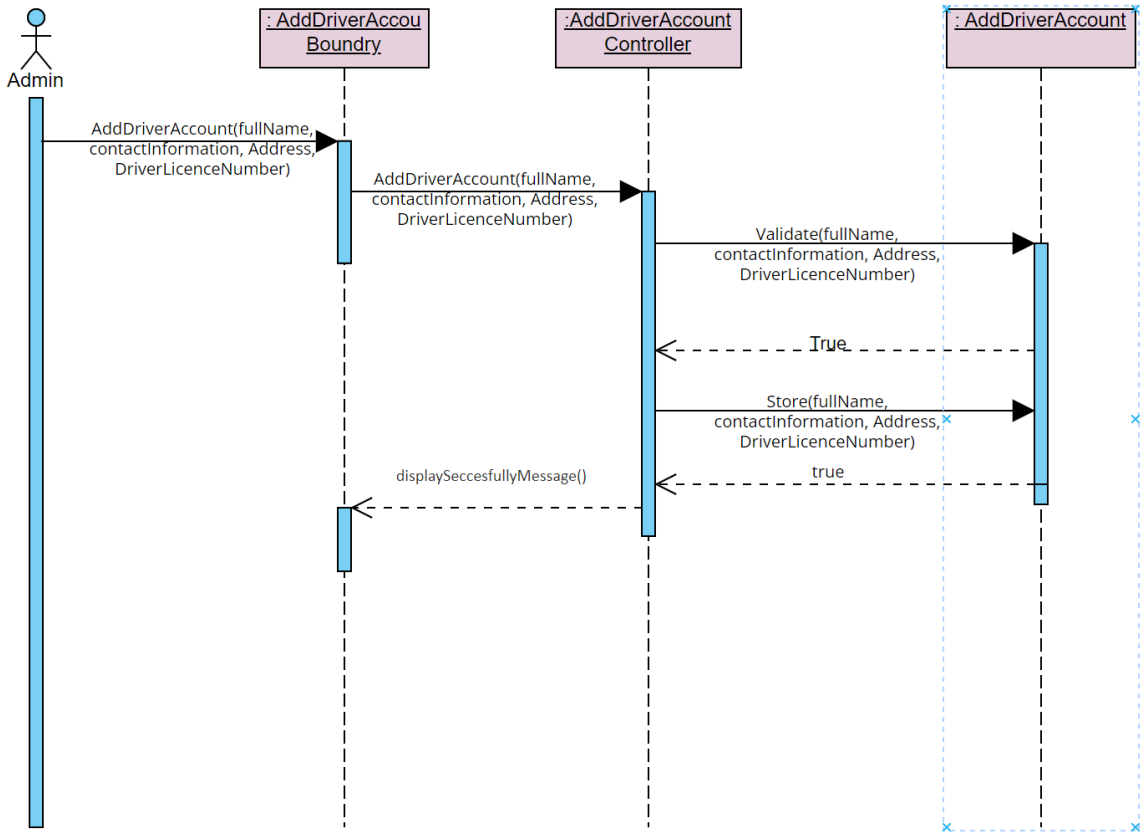


3.3.2 Sequence Diagram

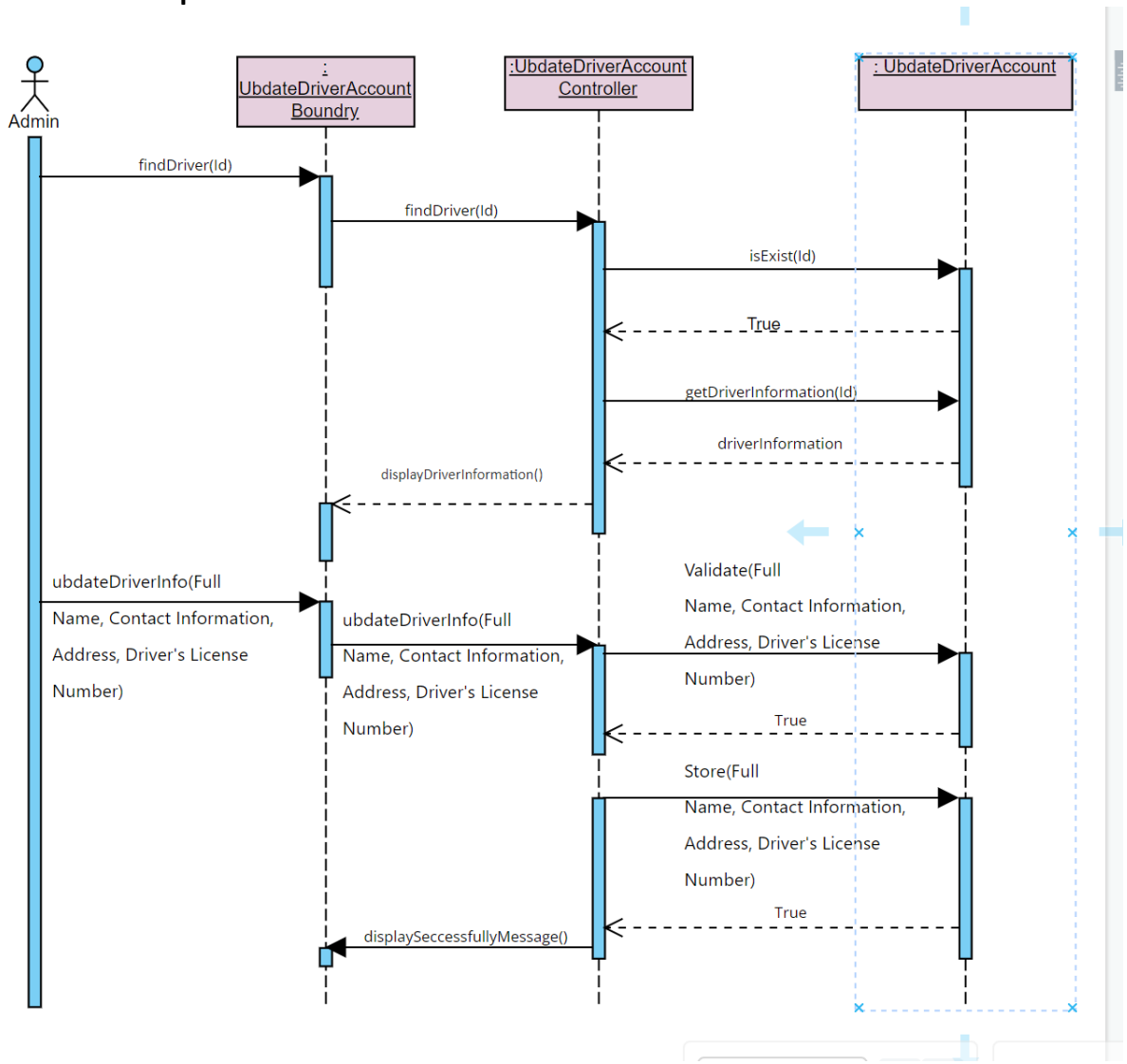
3.3.2.1 Use Case: Login



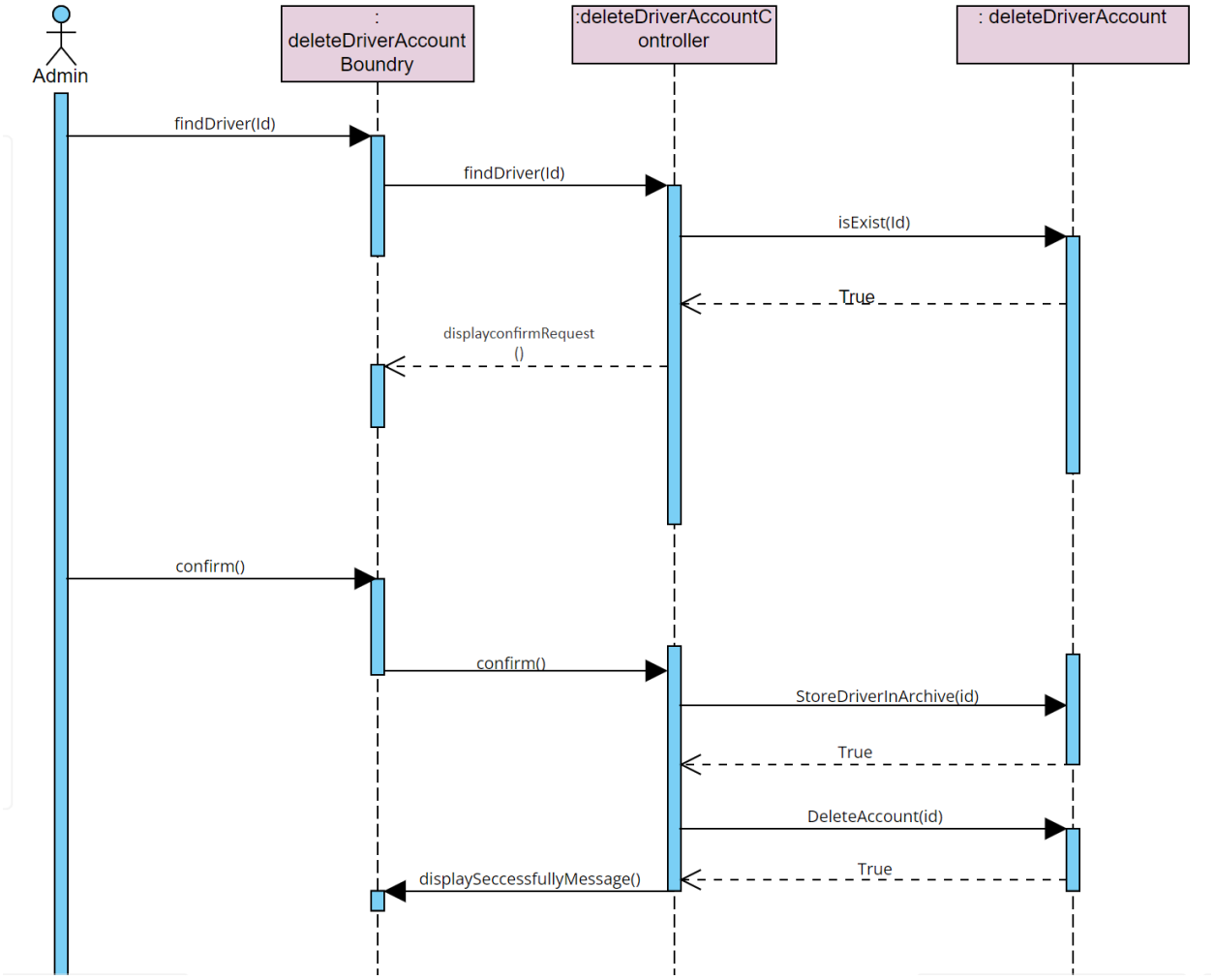
3.3.2.2 Use Case: Add Driver Account



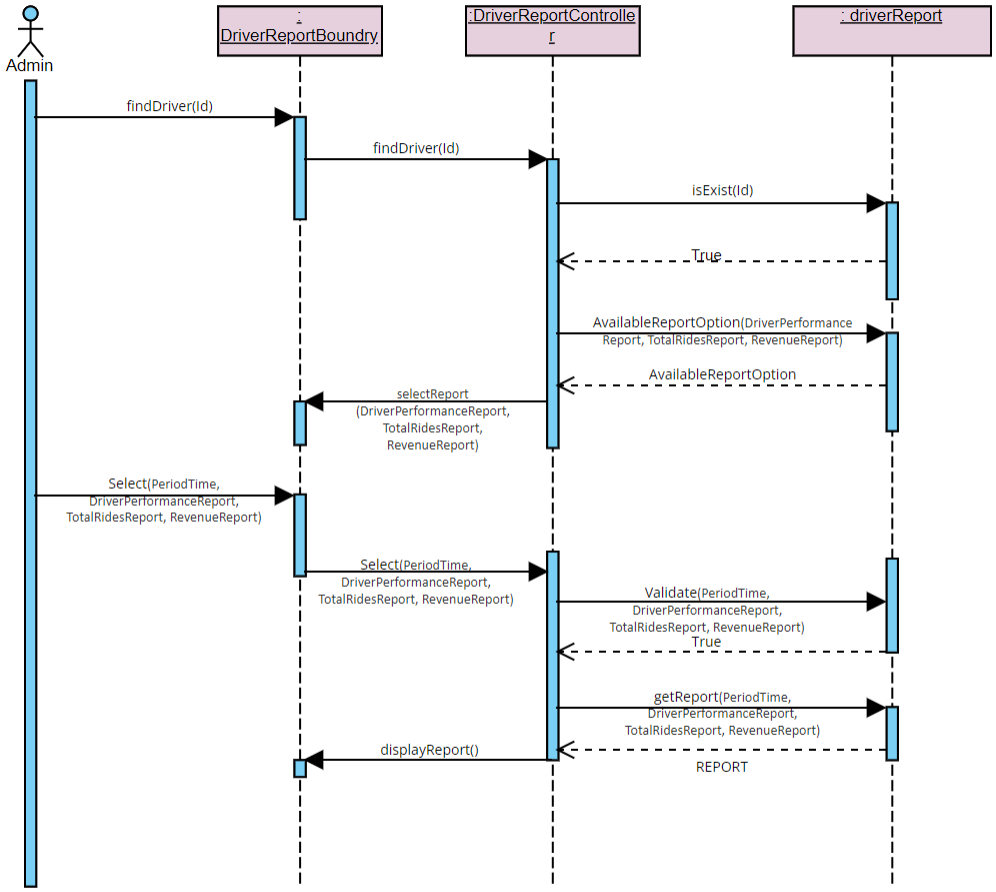
3.3.2.3 Use Case: Update Driver Account



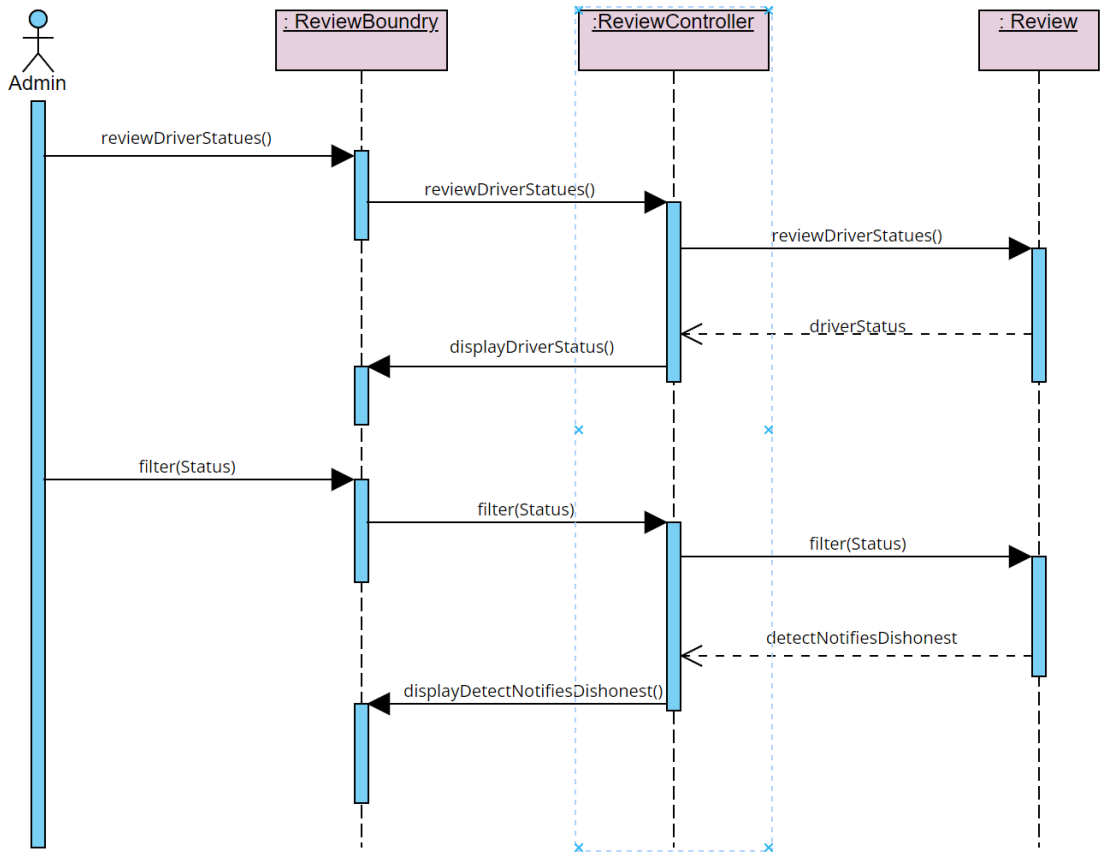
3.3.2.4 Use Case: Delete Driver Account



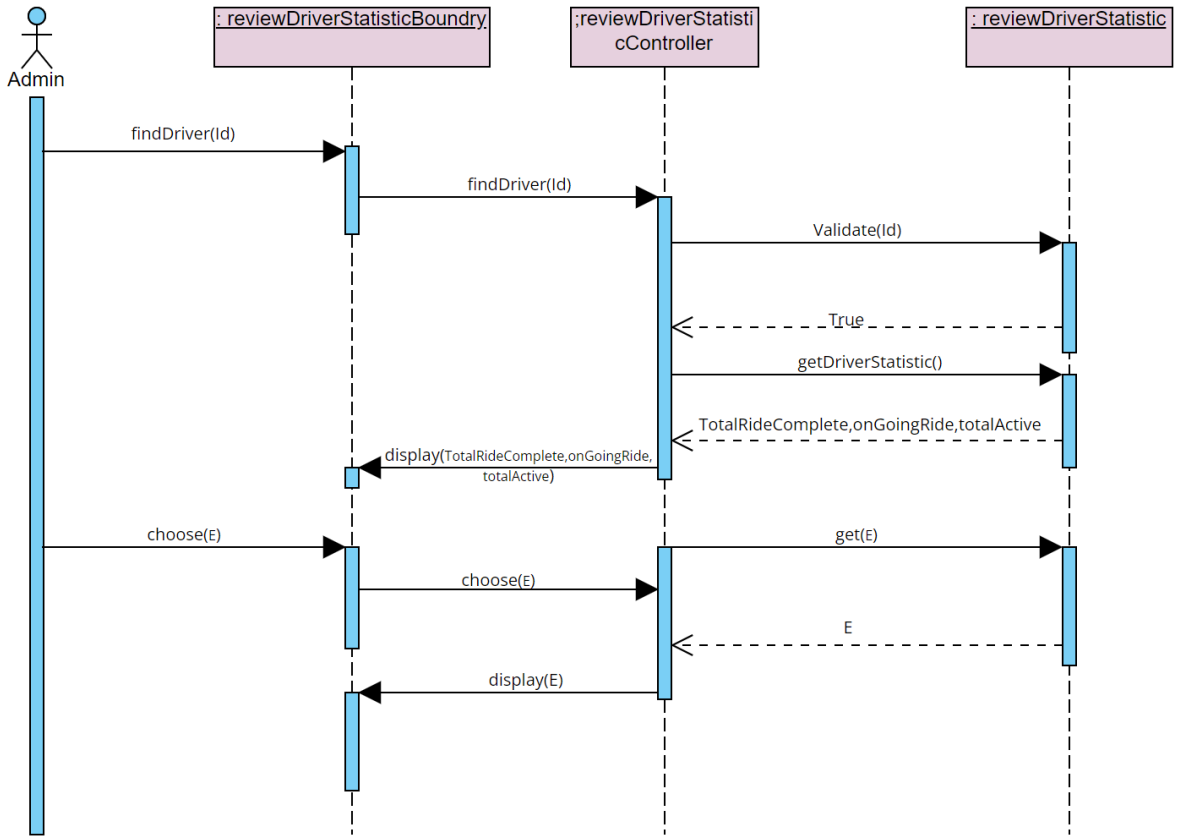
3.3.2.5 Use Case: Generate Driver Report



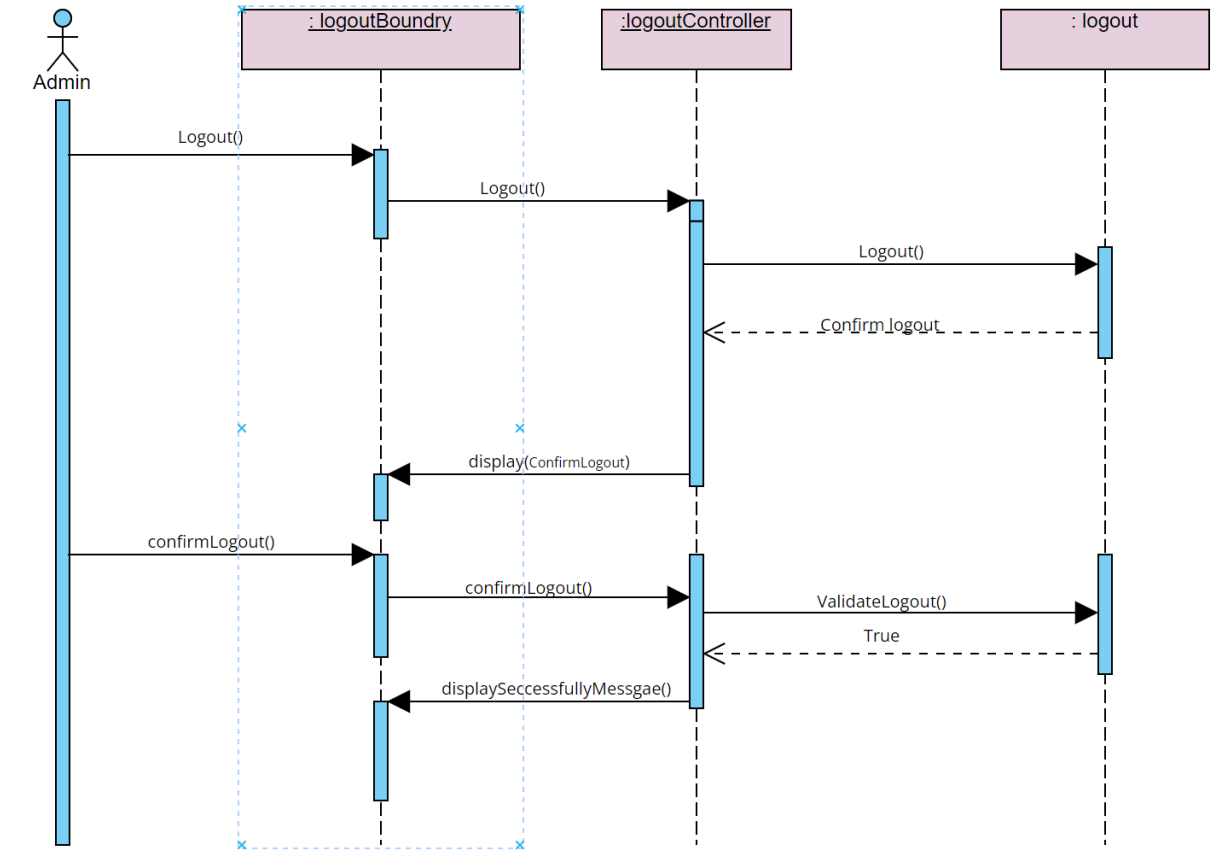
3.3.2.6 Use Case: Review Driver Status



3.3.2.7 Use Case: Review Driver statistics



3.3.2.8 Use Case: Logout



3.3.2.9 Use Case: Set Available Status