UNIVERSITY OF BARISHAL

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

SOFTWARE REQIREMENT SPACIFICATION(SRS)

BUTrace-University Transportation Management System

CSE-3104: Software Engineering and Information System Design Lab

Submitted By:

Team No: 01

Team Name: **Hexagon**

Team Members:

Md Mahruf Alam (22 CSE 007)

SOURAV DEBNATH (22 CSE 009)

Utsojet Paticor (22 CSE 017)

ABDUS SAKUR (22 CSE 024)

Imam Hossen (22 CSE 040)

Md. Imam Hosen (22 CSE 051)

Submitted To:

Md. Samsuddhoa
Assistant Professor
Department of Computer Science and Engineering

Submission Date: 19/01/26

TABLE OF CONTENTS

Contributions

Name	Topics
Md Imam Hosen	**Inception**
Abdus Sakur	**Elicitation, ERD**
Utsojet Paticor	**Use Case, Activity Diagram**
Imam Hossen	**Swim Lane Diagram, Class Based Model**
Md Mahruf Alam	**Behavioral Model**
Sourav Debnath	**Behavioral Model**

BUTrace – Problem Statement

The **BUTrace – University Transportation Management System** is designed to automate and streamline the day-to-day transportation operations of the University of Barishal. Traditionally, university transportation activities such as bus scheduling, route management, and commuter coordination have been handled manually, which often leads to uncertainty, delays, poor communication, and inefficient use of transport resources. To overcome these limitations, BUTrace provides a centralized, digital, and real-time solution that ensures efficient transportation management and improved commuting experience for students, faculty members, drivers, and administrators.

In this system, students, faculty members, drivers, and administrators are registered as authorized users. Each user is provided with a unique system account that allows them to log in and access features based on their assigned role. Once logged in, students and faculty members can view bus routes, schedules, and live bus locations on an interactive map. The system instantly displays real-time bus positions along with estimated arrival times, helping users plan their commute more effectively and reduce waiting time.

Drivers interact with the system through a dedicated interface that allows them to view assigned routes and schedules and broadcast real-time bus location automatically using GPS integration. Drivers can also update trip status and report incidents such as breakdowns, traffic congestion, or route diversions. These updates are immediately communicated to administrators and affected users, ensuring timely responses during disruptions.

From an administrative perspective, BUTrace enables centralized monitoring and control of the entire transportation system. Administrators can manage buses, routes, schedules, and driver assignments, as well as monitor live fleet movement. The system also maintains historical transportation data, which supports report generation, performance analysis, and data-driven decision-making to optimize routes and improve service quality.

Overall, the **BUTrace – University Transportation Management System** ensures smooth and transparent transportation operations, reduces manual workload, improves communication among stakeholders, and enhances user satisfaction by providing a modern, reliable, and efficient university transportation solution.

1. Inception

1.1 Introduction

Inception is the initial phase of the requirements engineering process for the **BUTrace-University Transportation Management System**. This phase defines how the software project is initiated and determines the scope and nature of the transportation-related problems to be addressed. The primary objective of the inception phase is to identify common requirements as well as potential conflicts among the various stakeholders involved in the system.

During this phase, emphasis is placed on understanding stakeholder expectations, clarifying system objectives, and identifying the fundamental requirements needed to support an efficient and reliable university transportation system. To establish a strong foundation for the BUTrace project, the following key activities were carried out during the inception phase:

1. Identifying stakeholders involved in the transportation system
2. Recognizing multiple stakeholder viewpoints
3. Working towards collaboration among users and administrators
4. Asking the first set of questions to define system goals and constraints

1.2 Identifying Stakeholders

A stakeholder refers to any individual or group that is directly or indirectly affected by the BUTrace. Stakeholders include end users who interact with the system as well as authorities and organizations whose operational activities may be influenced by the system's development and deployment. Identifying stakeholders is essential to understand system requirements and manage differing expectations.

To identify the stakeholders of the BUTrace system, discussions were conducted with personnel involved in university transportation management, and the following key questions were considered:

- Who is funding or sponsoring the project?
- Who will use the system and benefit from its outcomes?
- Who has the authority to make decisions regarding the project?
- Who provides the resources required to develop and maintain the system?
- Whose work will be affected during and after the system implementation?

1.2.1 Concluding Thoughts on Stakeholders

Based on this analysis, the following stakeholders were identified for the BUTrace-University Transportation Management System:

1. **University Administration (Project Sponsor):**
   The university administration holds final authority over project approval, budget allocation, and policy decisions. It ensures that the system aligns with institutional regulations and objectives.
2. **Transport Authority / Transport Administrator:**
   The transport authority is responsible for managing transportation operations and has administrative control over the system, including routes, schedules, buses, and drivers.
3. **System Administrator (Admin):**
   The system administrator manages user accounts, system configurations, monitoring, and overall system control, making this role a key stakeholder.
4. **Bus Drivers:**
   Drivers directly interact with the system to receive route and schedule information, broadcast live location data, and report incidents. Their daily operations are significantly affected by the system.
5. **Students and Faculty Members:**
   Students and faculty members are primary end users who rely on the system for real-time bus tracking, schedule updates, and notifications to support daily commuting.
6. **System Developers:**
   Developers are responsible for designing, implementing, and maintaining the system. They handle system updates, bug fixes, and future enhancements.
7. **University:**
   The university finances the project and enforces rules and regulations that must be strictly followed during system development and operation, making it a critical stakeholder.

1.2.2 Recognizing Multiple Viewpoints

Different stakeholders of the **BUTrace** have different expectations and concerns regarding the system. Recognizing these multiple viewpoints helps in balancing usability, cost, security, and operational efficiency.

1. **University Administration (Project Sponsor) Viewpoints:**
    1. User-friendly and efficient transportation management system
    2. Minimum maintenance and operational cost
    3. Availability of expected system features within the approved budget

2. **Transport Authority / Systems Head Viewpoints:**
    1. Allow the system to be accessed via the Internet
    2. Restrict system functionalities based on user roles
    3. Ensure system accessibility from any Internet-enabled device
    4. Web-based user interfaces for ease of use
3. **Transport Administrator (System Administrator) Viewpoints:**
    1. Ability to manage buses, routes, schedules, and drivers efficiently
    2. Centralized system deployment and maintenance
    3. Strong authentication and access control mechanisms
    4. Ability to generate operational and performance reports
4. **System Operator Viewpoints:**
    1. Easy-to-operate and user-friendly system
    2. Automatic generation of reports related to transportation operations
    3. Availability of a user guide describing system usage
    4. Availability of a reference manual for system installation and configuration
5. **Commuter (Student and Faculty) Viewpoints:**
    1. System accessible via the Internet
    2. User-friendly and reliable interface
    3. Easy access to bus routes, schedules, and live tracking
    4. Ability to view transportation information after logging into the system
6. **Developer Viewpoints:**
    1. Clearly defined and unambiguous requirements
    2. Ease of system development, maintenance, and enhancement
7. **University Viewpoints:**
    1. System development within allocated budget
    2. Compliance with university rules, regulations, and policies

1.3 Towards Collaboration

Each stakeholder involved in the **BUTrace** has individual requirements and expectations. In this step, these requirements were analyzed and merged to form a unified and feasible set of system requirements. The following steps were followed to complete this task:

1. Identification of common and conflicting requirements

2. Categorization of the identified requirements
3. Collection of priority points for each requirement from stakeholders and prioritization based on voting
4. Making final decisions on system requirements

1.3.1 Common Requirements:

1. User-friendly and efficient system
2. Easy to operate
3. System accessible from any computer or device with Internet access

1.3.2 Conflicting Requirements:

1. Minimum system maintenance cost
2. Availability of all required features within the project budget
3. Easy access for users
4. Restriction of system functionality based on user roles
5. Clearly defined and unambiguous requirements

After categorizing and prioritizing these requirements, the following **final requirements** were established for the BUTrace system:

1.3.3 Final Requirements:

1. The system should be reliable and error-free (maximum 5% error tolerance).
2. The system shall be accessible via the Internet.
3. The system shall allow valid users to log in and log out.
4. The system shall restrict access to functionalities based on user roles.
5. The system shall allow administrators to change user roles and configure system parameters.
6. The system shall allow users to view bus routes, schedules, and live bus information without mandatory login.
7. The system shall allow logged-in users to track buses, receive notifications, and view trip-related information.
8. The system shall allow drivers to update trip status and report incidents.
9. The system shall allow administrators to generate reports related to fleet usage, routes, and incidents.
10. The system shall be centrally deployed and maintained.
11. The system shall allow valid users to access transportation services online after logging into the system.

1.4 Asking the First Questions

The initial set of context-free questions focused on the needs of customers and other stakeholders of BUTrace, as well as the overall project goals and expected benefits. These questions helped identify all relevant stakeholders, determine the measurable benefits of successful system implementation, and explore possible alternatives to developing a custom transportation management system.

The next set of questions aimed to achieve a clearer understanding of the transportation-related problems and allowed stakeholders to express their perceptions and expectations regarding the proposed solution. The final set of questions focused on evaluating the effectiveness of communication among stakeholders to ensure that requirements were clearly understood and properly documented.

2.Elicitation

2.0.1 Introduction

Elicitation is the process that helps stakeholders clearly define what is required from the **BUTrace**. During the elicitation phase, several challenges were encountered, including issues related to system scope, requirement volatility, and differences in stakeholder understanding.

Requirement elicitation is not a straightforward task. To address these challenges effectively, the elicitation activities for BUTrace were conducted in an organized and systematic manner, ensuring that stakeholder needs were accurately captured and documented.

2.0.2 Eliciting Requirements

Unlike the inception phase, where a question-and-answer approach is primarily used, the elicitation phase for the **BUTrace** employs a structured requirements elicitation process that combines problem solving, elaboration, negotiation, and specification. This process requires close cooperation between system users, administrators, and developers to accurately identify and refine system requirements.

To elicit the requirements for the BUTrace system, the following four activities were carried out:

1. Collaborative Requirements Gathering
2. Quality Function Deployment
3. Usage Scenarios
4. Elicitation Work Products

2.1 Collaborative Requirements Gathering

Several approaches to collaborative requirements gathering have been proposed, each utilizing different interaction scenarios. For the **BUTrace**, collaborative requirements gathering was carried out to ensure that stakeholder needs and expectations were clearly understood. The following steps were completed during this process:

- Meetings and discussions were conducted with personnel involved in university transportation management, including transport administrators and potential system users, to gather requirements and expectations from the proposed system.
- Stakeholders were asked about the difficulties and limitations they face with the existing manual and uncoordinated transportation system.
- Based on these discussions, a final and prioritized list of system requirements was selected.

2.2 Quality Function Deployment

Quality Function Deployment (QFD) is a technique used to translate customer needs into specific technical requirements for software development. It focuses on maximizing customer satisfaction throughout the software engineering process. With respect to BUTrace the system requirements were identified and prioritized using the QFD approach.

2.3 Normal Requirements

Normal requirements consist of objectives and goals that were identified during meetings with stakeholders of the **BUTrace**. The normal requirements of the system are as follows:

1. The system shall be accessible via the Internet.
2. The system shall allow users to view bus routes, schedules, and transportation information.
3. The system shall allow drivers to update trip status and broadcast bus location for valid users.
4. The system shall allow valid users to log in and log out.
5. The system shall restrict access to functionalities based on user roles.
6. The system shall allow logged-in users to track buses, receive notifications, and view trip-related information.
7. The system shall allow administrators to generate reports related to fleet usage, routes, and incidents.
8. The application shall be centrally deployed and maintained.
9. The system shall provide a help feature to assist users.

10. A product reference manual describing system installation, setup, and operation shall be provided.

2.4 Expected Requirements

Expected requirements are implicit system requirements that may not be explicitly stated by stakeholders but are essential for system satisfaction. The absence of these requirements may result in user dissatisfaction. The expected requirements of the BUTrace system are as follows:

1. The system shall maintain a centralized database of buses, routes, schedules, users, trips, and incidents.
2. The system shall enable the administrator to change user roles and access levels.
3. The system shall enable the administrator to reset or change user passwords.
4. The system shall enable the administrator to configure route schedules and timing parameters.
5. The system shall enable the administrator to assign and update driver-bus associations.
6. The system shall enable the administrator to configure notification rules and alert parameters.
7. The system shall support role-based access control for all user types.
8. The system shall allow users to log in using assigned credentials securely.
9. The system shall automatically update trip status based on driver inputs and GPS data.
10. The system shall automatically notify users about delays, route changes, or emergencies.
11. The system shall store historical trip and usage data.
12. The system shall generate logs for system activities and incidents.
13. The system shall manage bus status such as “Active,” “Delayed,” or “Out of Service.”
14. The system shall allow administrators to add, edit, or remove buses and routes.
15. The user interface shall be easy to use and make use of selection-based input fields wherever possible.

2.5 Exciting Requirements

Exciting requirements refer to system features that go beyond basic stakeholder expectations and significantly increase user satisfaction when implemented. The exciting requirements of the **BUTrace** are as follows:

1. The user interface shall provide clear error messages for invalid inputs along with tooltips and online help.

2. The user interface shall follow standard web design practices to ensure consistency with common Internet applications.
3. The system shall support login using mobile phone numbers.
4. System configuration details shall be properly documented and updated whenever patches or new releases are applied.
5. The system shall support social media–based or third-party authentication, where applicable.

2.6 Usage Scenarios

Initially, a user registers in the BUTrace system by creating an account. If the user already has an account, they log in using valid credentials. After successful authentication, students or faculty members search for transportation information such as bus routes and schedules. If no matching route is found, the system displays an appropriate message. Otherwise, the system retrieves and displays real-time bus availability and location information.

In Case of Viewing an Available Bus

- If a bus is available on the selected route, the system verifies the user's authentication status.
- For valid users, the system displays live bus location, estimated arrival time, and trip status.
- The system updates the user view dynamically based on real-time data received from GPS services.

In Case of Bus Delay or Incident

- If a bus is delayed or an incident is reported by the driver, the system updates the bus status accordingly.
- Users receive notifications regarding the delay or route change.
- Administrators are notified and may take appropriate action.

2.7 Elicitation Work Product

The output of the elicitation activities for the **BUTrace** varies depending on system size and complexity. The elicitation work products for this project include:

- A documented statement of requirements for the BUTrace system.
- A bounded statement defining the scope of the system.
- A list of customers, users, and other stakeholders who participated in requirements elicitation.
- A set of usage scenarios describing system interactions.
- A description of the system's technical and operational environment.

3.Use Case Scenario

This table presents the **Use Case Scenario** for the *BUTrace – University Transportation System*, organized into three hierarchical levels to clearly define system functionality and user interaction.

- **Level 0** represents the overall system domain, which is *BUTrace.*
- **Level 1** groups major functional modules of the system such as Authentication, User Management, Bus & Fleet Management, Route & Schedule Management, Bus Tracking, Trip Operations, Incident Reporting, Reports & Analytics, and Emergency Management.
- **Level 2** specifies detailed actions or operations available within each module, such as signing in, assigning drivers, viewing live bus locations, reporting incidents, and generating reports.

The following table summarizes the use cases of the system.

Level – 0	**Level – 1**	**Level – 2**	**Actors**
BUTrace-University Transportation	**Authentication**	Sign Up	Student, Faculty
		Sign In	Student, Faculty, Driver, Admin
		Sign Out	Student, Faculty, Driver, Admin
		Change Password	Student, Faculty, Driver, Admin
	User Management	Verify User Account	Admin
		Update User Profile	Student, Faculty, Driver
		Manage User Roles	Admin
		Add Bus	Admin

	Bus & Fleet Management	Edit Bus Information	Admin
		Remove Bus	Admin
		Assign Driver to Bus	Admin
	Route & Schedule Management	Add Route	Admin
		Update Route	Admin
		Define Schedule	Admin
		Modify Schedule	Admin
	Bus Tracking	View Live Bus Location	Student, Faculty, Admin
		Broadcast Bus Location	Driver
		View Estimated Arrival Time	Student, Faculty
	Trip Operations	Start Trip	Driver
		Update Trip Status	Driver
		Complete Trip	Driver
	Incident Reporting	Report Breakdown	Driver
		Report Traffic Issue	Driver
		Submit Complaint/Feedback	Student, Faculty

		Review Incident Report	Admin
	Reports & Analytics	Generate Fleet Report	Admin
		Generate Route Performance Report	Admin
		Generate Bus Utilization Report	Admin
		View Historical Data	Admin
	Emergency Management	Trigger Emergency Alert	Driver, Admin
		Notify Affected Users	System

Table: Use Case Scenario

Table: Use Case Scenario

Use Case Description and Use Case Diagram

Level 0-BUTrace - University Transportation Management System

Primary Actors: Student, Faculty, Driver, Admin

Goal in Context:

To provide a complete platform for managing, tracking, and ensuring the safety of university transportation.

Preconditions:

System is deployed and accessible.

Triggers:

User accesses the BUTrace system.

Action and Reply

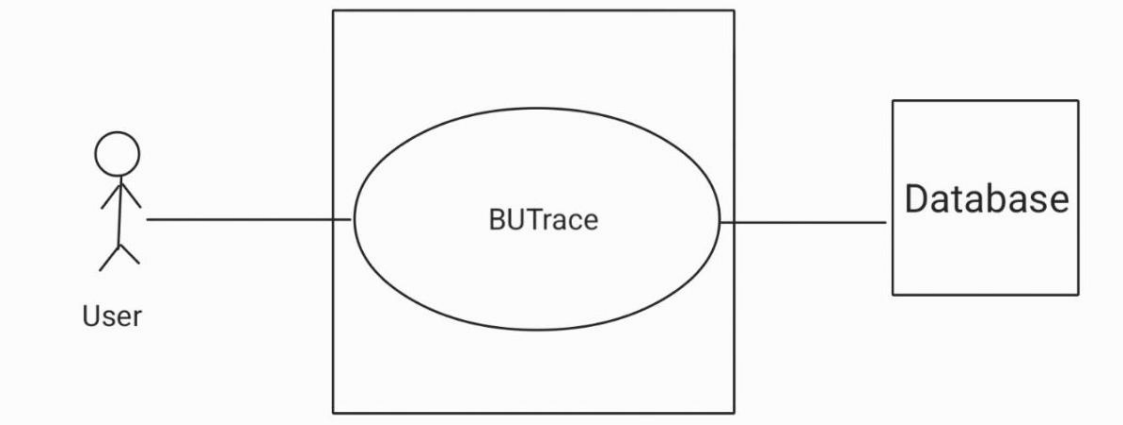
A1: User accesses the BUTrace system.
R1: System loads the main interface.

A2: User selects required service (authentication, tracking, reporting, etc.).
R2: System displays the corresponding module.

A3: User performs operations.
R3: System processes requests and returns results.

Priority: Critical

When Available: First increment



Level 0 - BUTrace

Level 1 - Core System Functionalities

Primary Actors: Student, Faculty, Driver, Admin

Includes: Authentication, User Management, Fleet, Routes, Tracking, Trip Operations, Incidents, Reports, Emergency

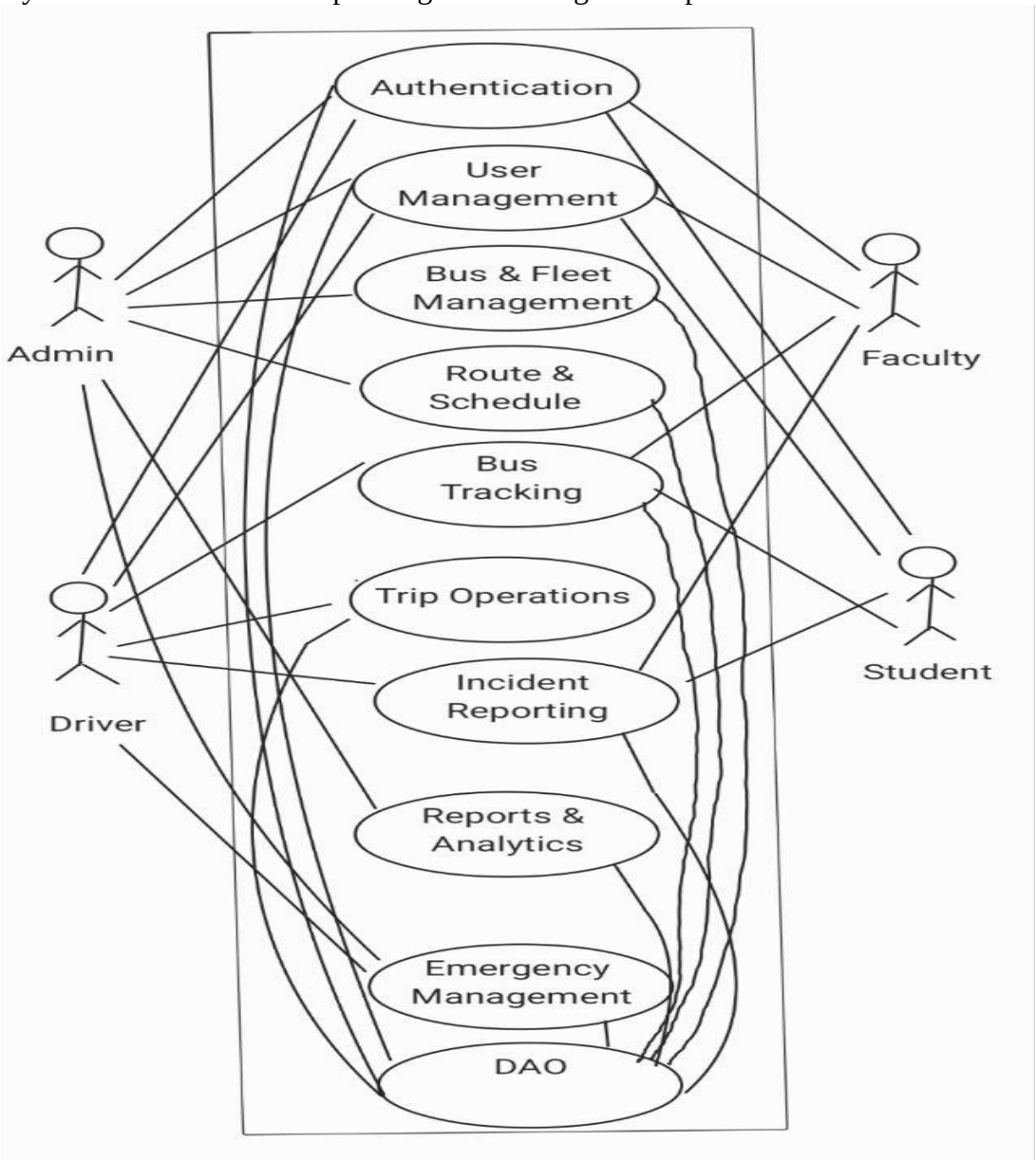
Action and Reply

A1: User selects a functional module from the dashboard.

R1: System loads the interface of the selected module.

A2: User performs operations within the module.

R2: System executes the corresponding business logic and updates the database.



Level 1 - Core System Functionalities of BUTrace

Level 1.1 - Authentication

Use Case: Sign Up / Sign In / Sign Out / Change Password

Primary Actors: Student, Faculty, Driver, Admin
Goal in Context: To allow secure access and account management.

Preconditions: User has valid credentials and a university ID.

Triggers: User selects authentication option.

Action and Reply

A1: User selects Sign Up / Sign In.
R1: System displays authentication form.

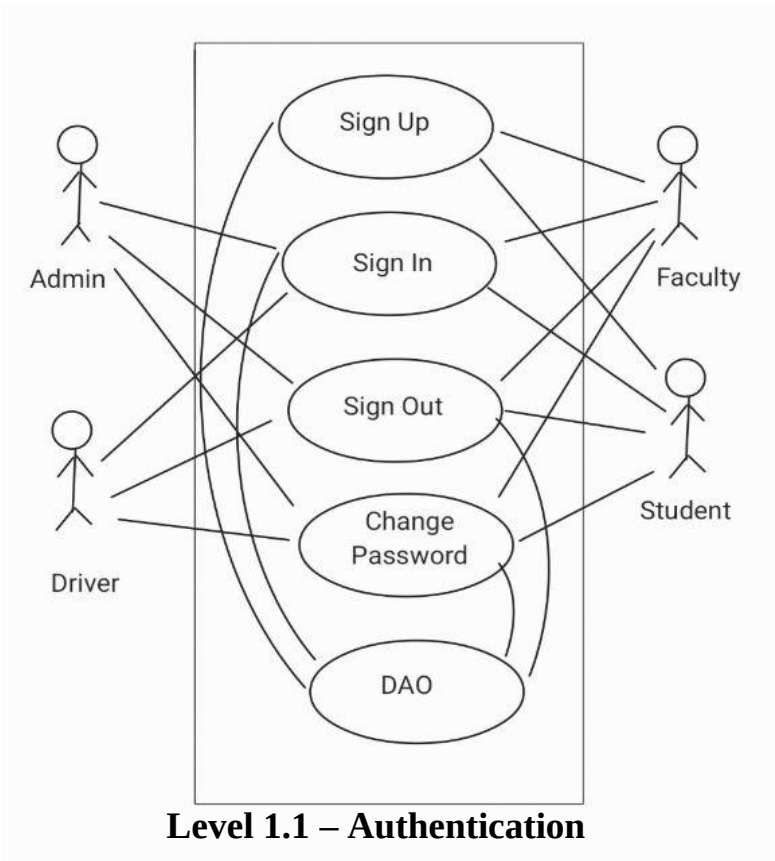
A2: User enters credentials.
R2: System validates input.

A3: User submits form.
R3: System authenticates and creates session.

A4: User selects Sign Out / Change Password.
R4: System terminates session or updates password.

Exceptions: Invalid credentials, suspended account.

Priority: Essential
When Available: First increment



Level 1.1 – Authentication

Level 1.2 - User Management

Use Case: Verify Account, Update Profile, Manage Roles

Primary Actor: Admin
Goal in Context: To control user access and information.

Preconditions: User is registered.

Triggers: Admin reviews user list.

Action and Reply

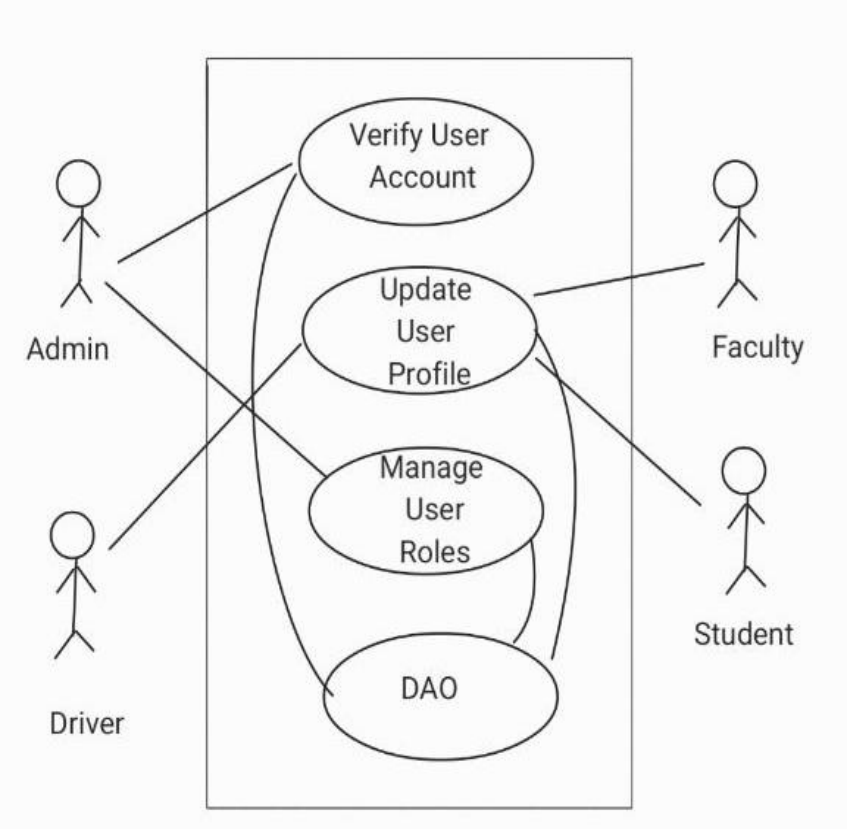
A1: Admin opens user list.
R1: System displays registered users.

A2: Admin selects a user.
R2: System shows user details.

A3: Admin verifies or updates role.
R3: System updates user status and permissions.

Exceptions: Invalid user data.

Priority: Essential
When Available: First increment



Level 1.2 - User Management

Level 1.3 - Bus & Fleet Management

Use Case: Add/Edit/Remove Bus, Assign Driver

Primary Actor: Admin
Goal in Context: To maintain the bus fleet.

Preconditions: Admin logged in.

Triggers: Admin manages fleet.

Action and Reply

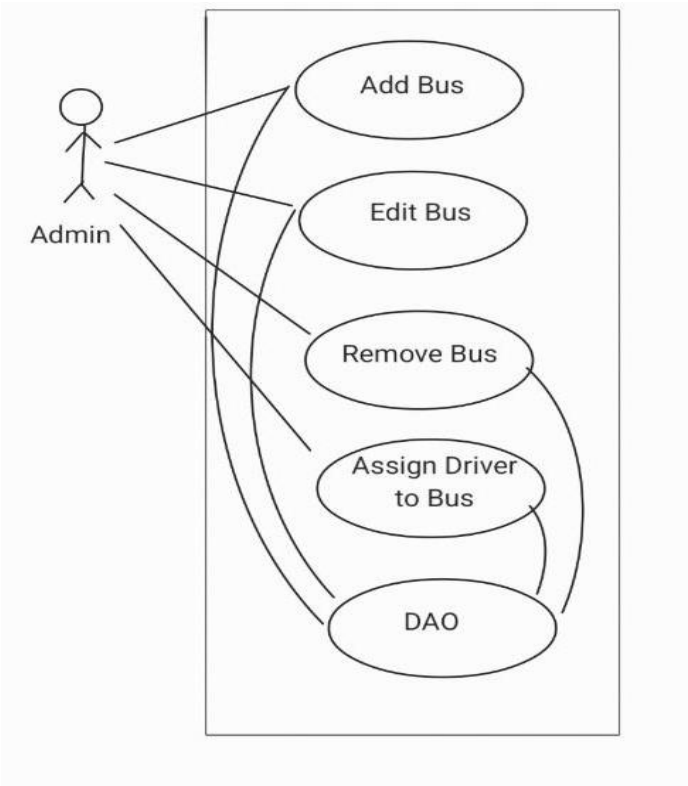
A1: Admin selects Add/Edit Bus.
R1: System shows bus information form.

A2: Admin enters or updates bus details.
R2: System validates and stores data.

A3: Admin assigns driver.
R3: System links driver with bus and updates schedule.

Exceptions: Duplicate bus ID, unavailable driver.

Priority: Essential
When Available: First increment



Level 1.3 - Bus & Fleet Management

Level 1.4 - Route & Schedule Management

Use Case: Add/Update Route, Define/Modify Schedule

Primary Actor: Admin

Goal in Context: To organize routes and timings.

Preconditions: Routes exist.

Triggers: Admin edits schedule.

Action and Reply

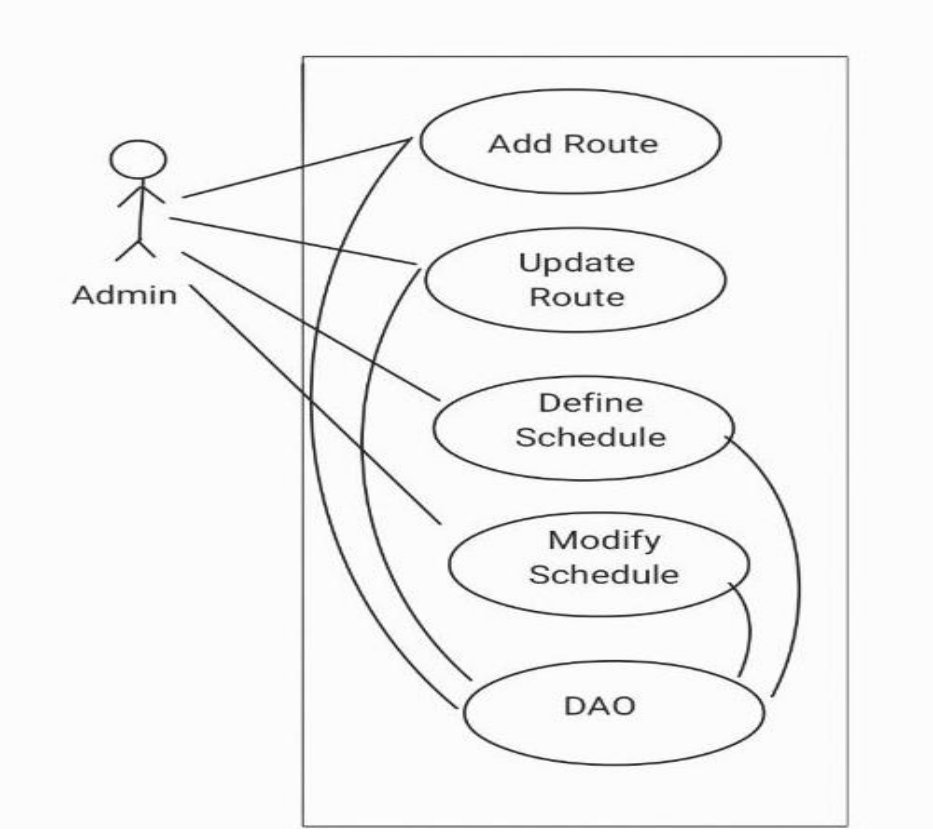
A1: Admin selects route or schedule option.
R1: System displays route and timetable editor.

A2: Admin updates route/time.
R2: System saves changes.

A3: System sends notification.
R3: Users receive updated schedule.

Exceptions: Invalid time, overlapping routes.

Priority: Essential
When Available: First increment



Level 1.4 - Route & Schedule Management

Level 1.5 - Bus Tracking

Use Case: View Live Location, Broadcast Location

Primary Actors: Student, Faculty, Driver

Goal in Context: To track buses in real time.

Preconditions: GPS enabled, trip active.

Triggers: User opens map / Driver starts trip.

Action and Reply

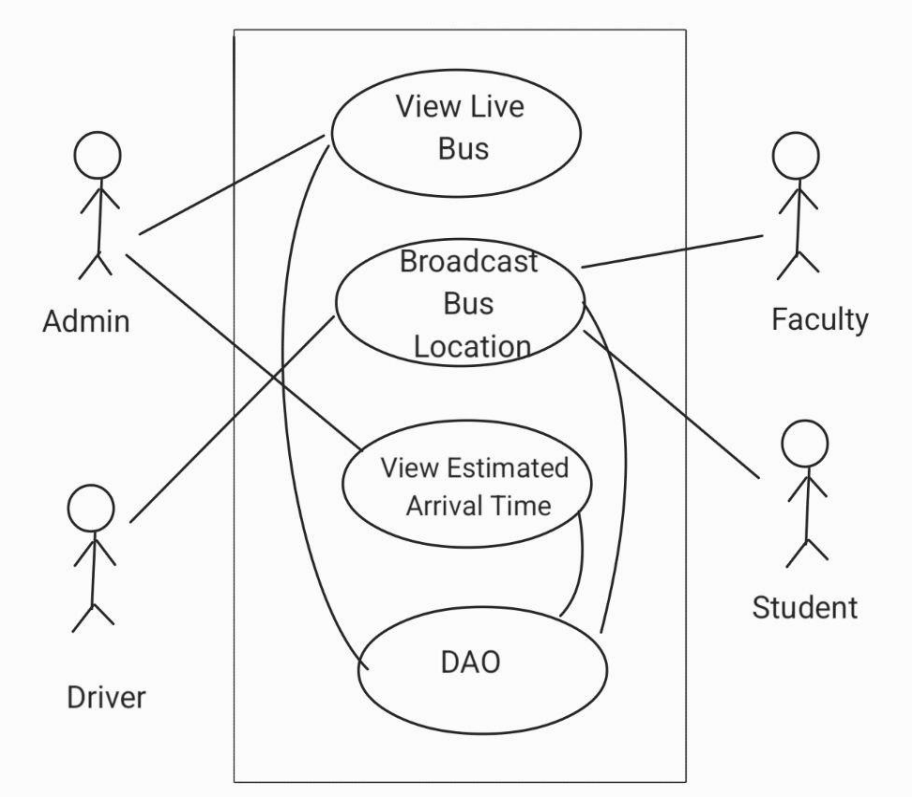
A1: Driver starts trip.
R1: System activates GPS tracking.

A2: GPS sends location data.
R2: System updates live map.

A3: Student selects route.
R3: System shows live bus and ETA.

Exceptions: GPS failure.

Priority: Essential
When Available: First increment



Level 1.5 - Bus Tracking

Level 1.6 - Trip Operations

Use Case: Start Trip, Update Status, Pause Status, Complete Trip

Primary Actor: Driver

Goal in Context: To manage the trip lifecycle.

Preconditions: Driver assigned to bus.

Triggers: Driver presses start/end.

Action and Reply

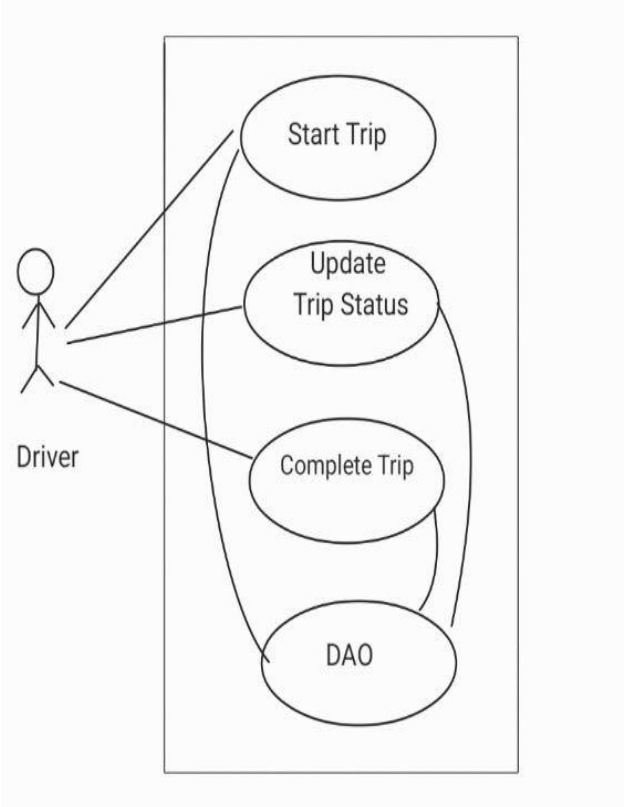
A1: Driver presses Start Trip.
R1: System records trip start and status.

A2: Driver updates trip status.
R2: System saves status and updates ETA.

A3: Driver presses End Trip.
R3: System closes trip and stores history.

Exceptions: Network failure, GPS error.

Priority: Essential
When Available: First increment



Level 1.6 - Trip Operations

Level 1.7 - Incident Reporting

Use Case: Report Breakdown, Traffic Issue, Complaint

Primary Actors: Driver, Student, Faculty

Goal in Context: To report problems.

Preconditions: User logged in.

Triggers: An incident occurs.

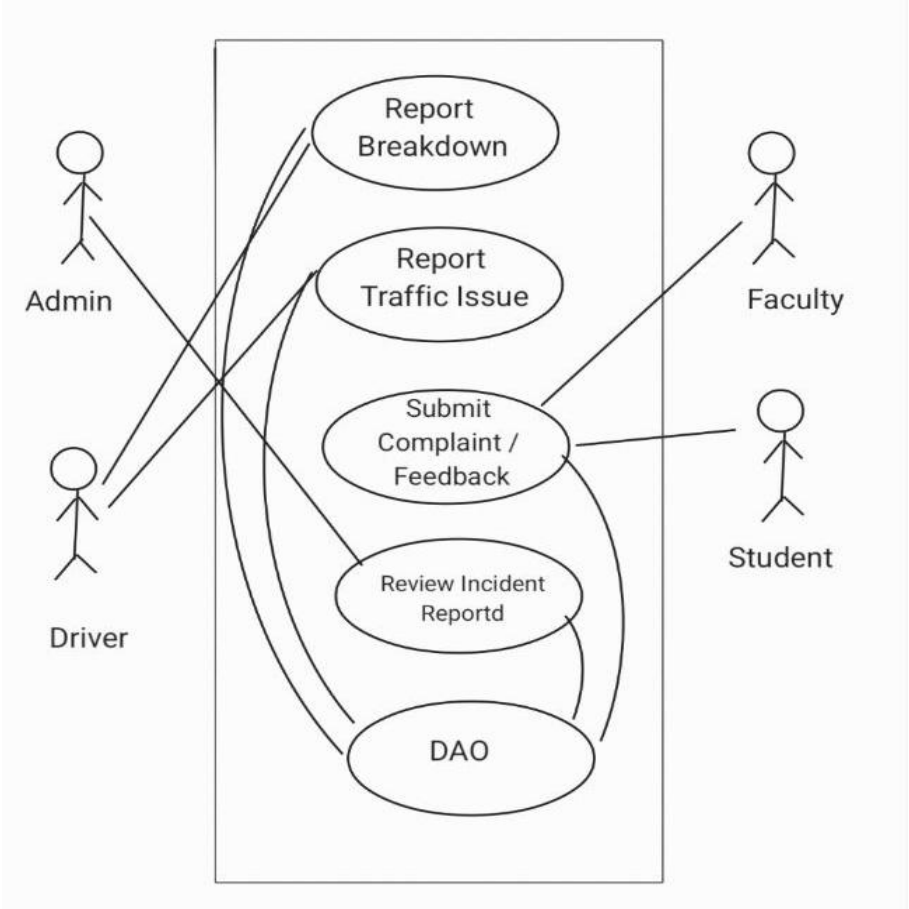
Action and Reply

A1: User selects Report Incident.
R1: System displays incident form.

A2: User submits report.
R2: System saves report and notifies admin.

Exceptions: Submission failure.

Priority: Essential
When Available: First increment



Level 1.7 - Incident Reporting

Level 1.8 - Reports & Analytics

Use Case: Generate Fleet, Route, Utilization Reports

Primary Actor: Admin

Goal in Context: To analyze performance.

Preconditions: Historical data available.

Triggers: Admin requests report.

Action and Reply

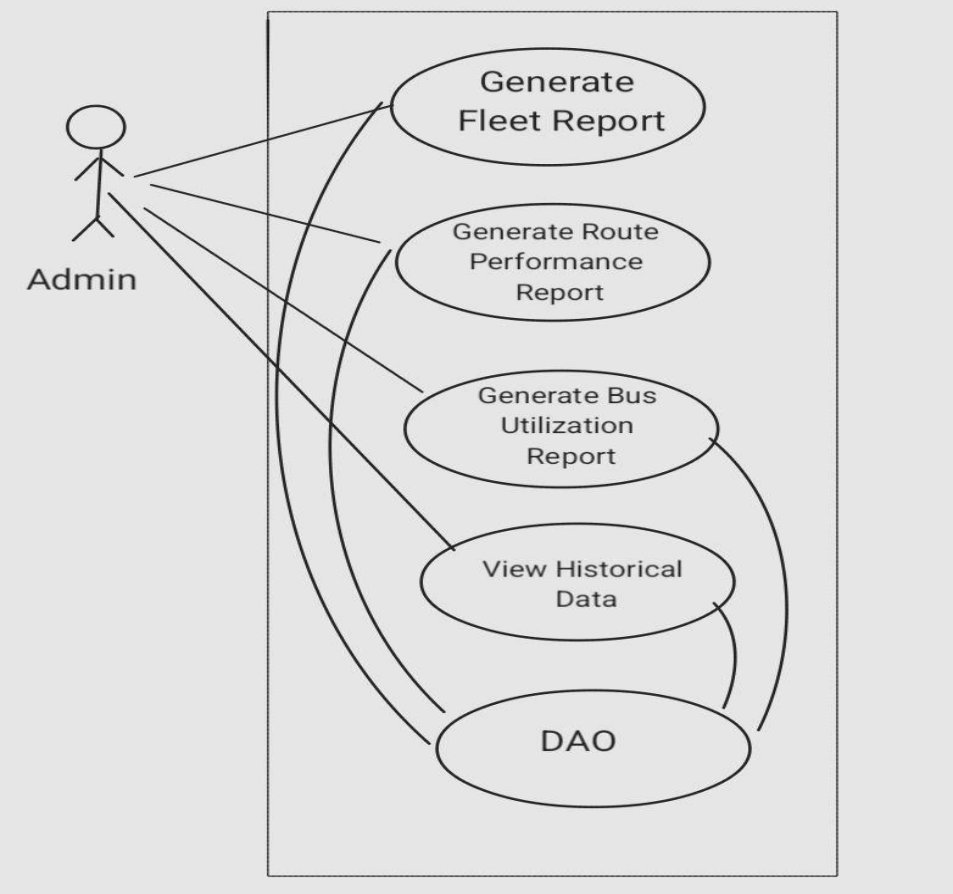
A1: Admin selects report type.
R1: System retrieves data.

A2: System processes statistics.
R2: System displays charts and tables.

Exceptions: No data found.

Priority: Important

When Available: Second increment



Level 1.8 - Reports & Analytics

Level 1.9 - Emergency Management

Use Case: Trigger Alert, Notify Users

Primary Actors: Driver, Admin
Secondary Actor: System

Goal in Context: To ensure safety during emergencies.

Preconditions: Emergency occurs.

Triggers: Emergency button pressed.

Action and Reply

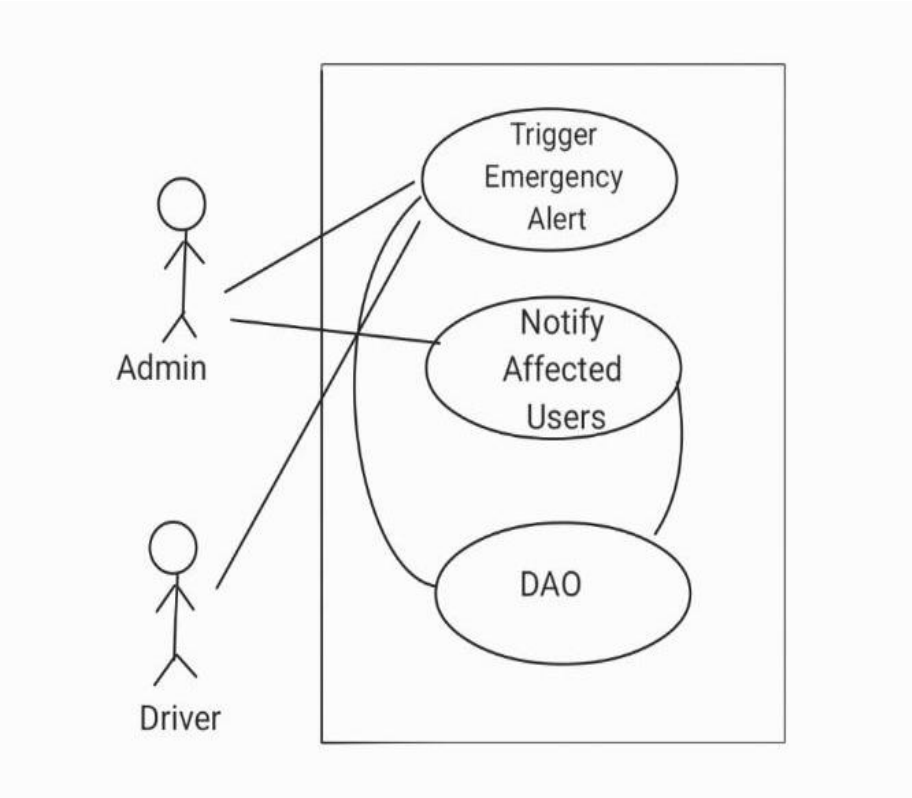
A1: Driver/Admin presses Emergency Alert.
R1: System activates emergency mode.

A2: System sends alerts to users.
R2: Users receive notification.

A3: Admin monitors emergency dashboard.
R3: System shows live status and responses.

Exceptions: Notification failure.

Priority: Critical
When Available: First increment



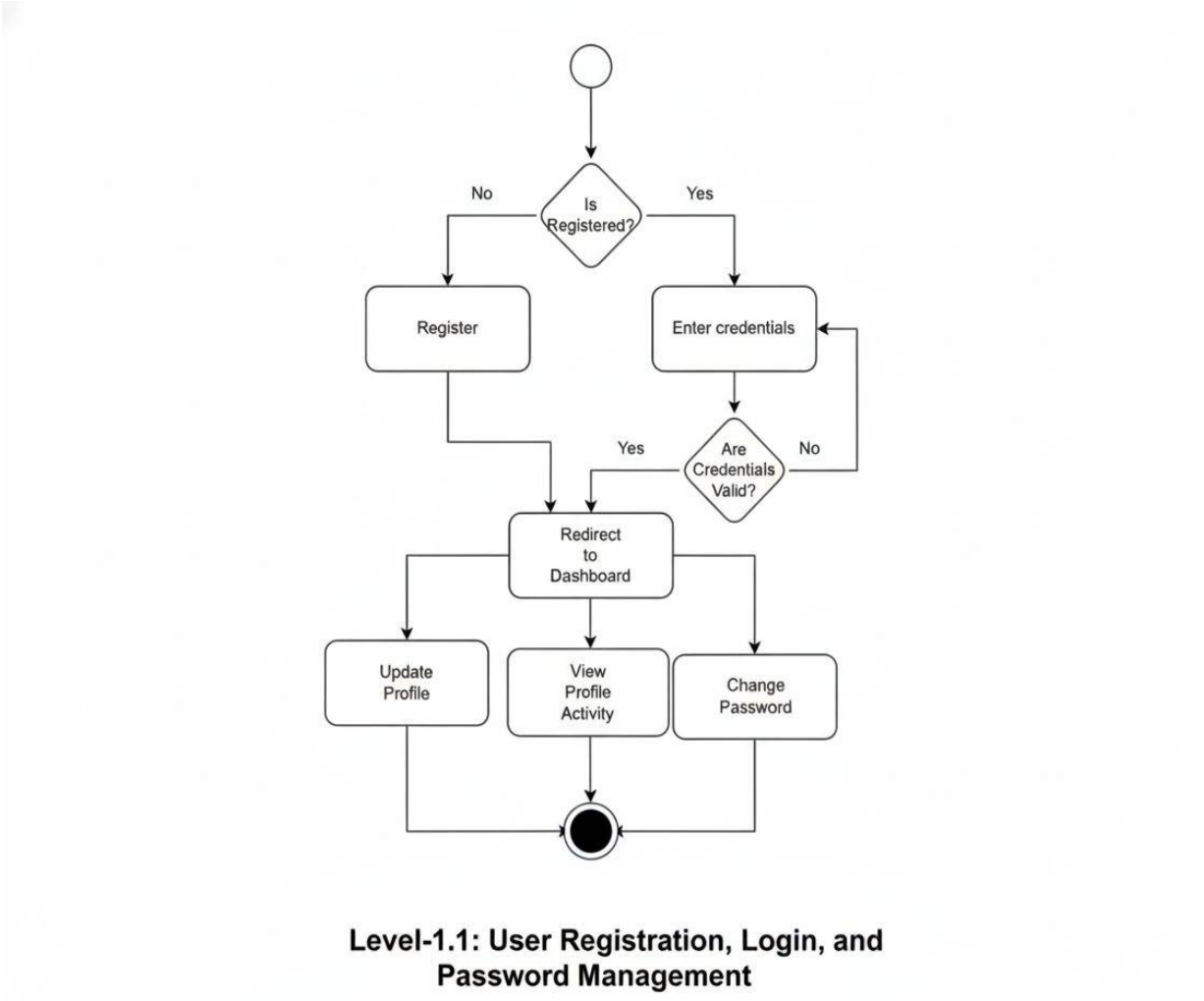
Level 1.9 - Emergency Management

Activity Diagram

Level 1.1:

Name: Authentication (Sign Up, Sign In, Sign Out, Change Password)

Reference: Use Case Level 1.1

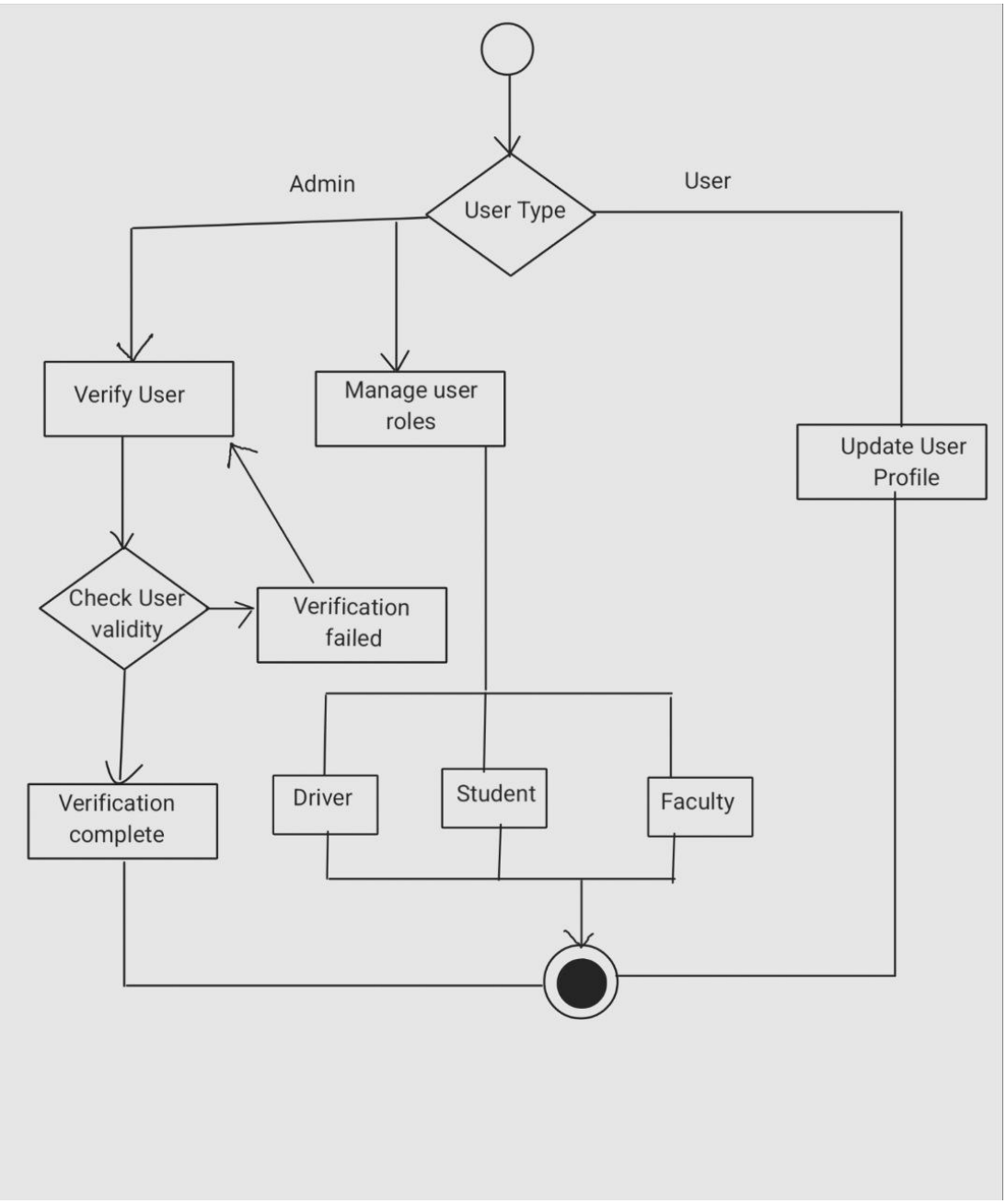


Level-1.1: User Registration, Login, and Password Management

Level 1.2:

Name: User Management

Reference: Use Case Level 1.2

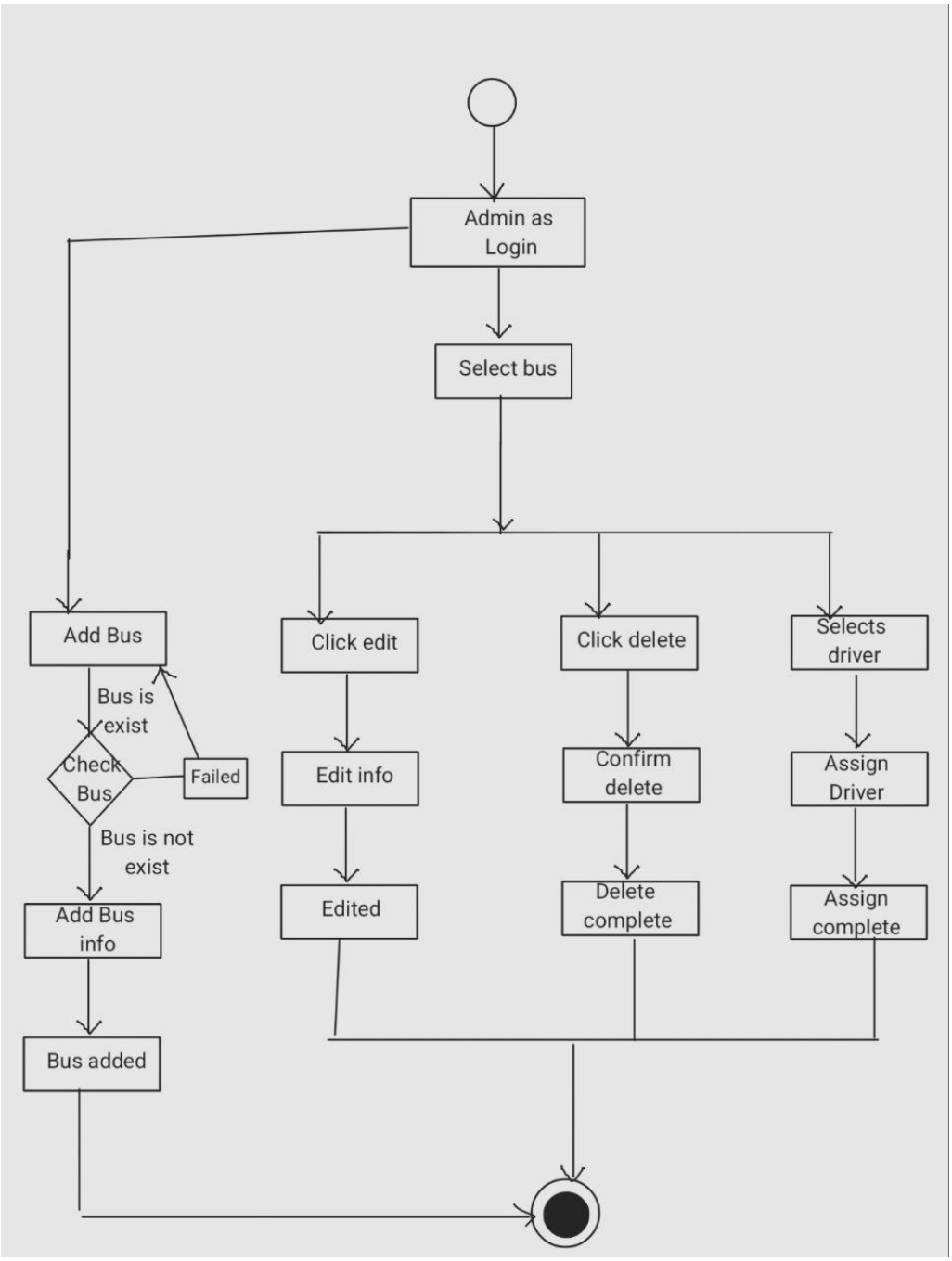


Level 1.2 - User Management

Level 1.3:

Name: Bus & Fleet Management

Reference: Use Case Level 1.3

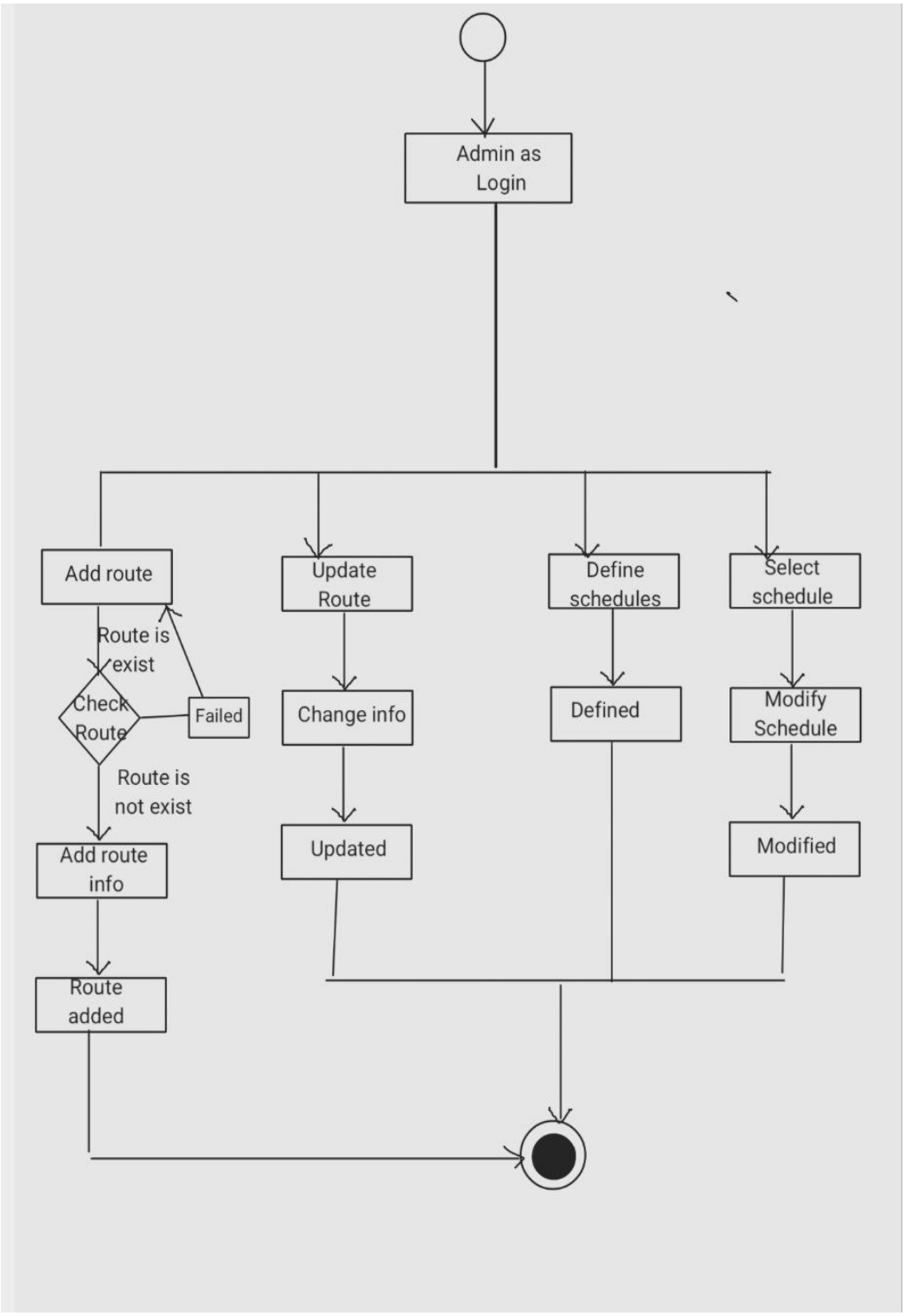


Level 1.3 - Bus & Fleet Management

Level 1.4:

Name: Route & Schedule Management

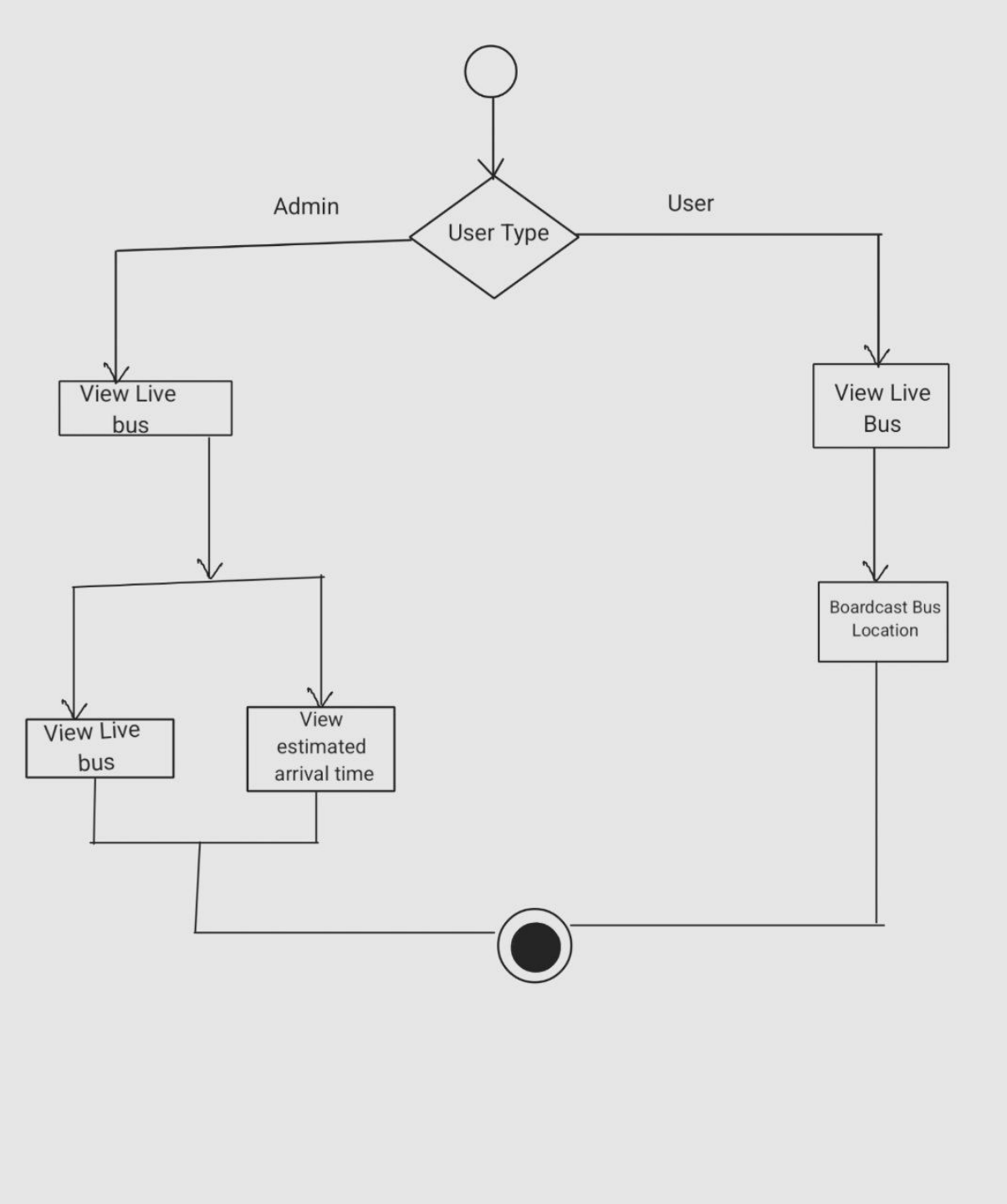
Reference: Use Case Level 1.4



Level 1.4 - Route & Schedule Management

Level 1.5:

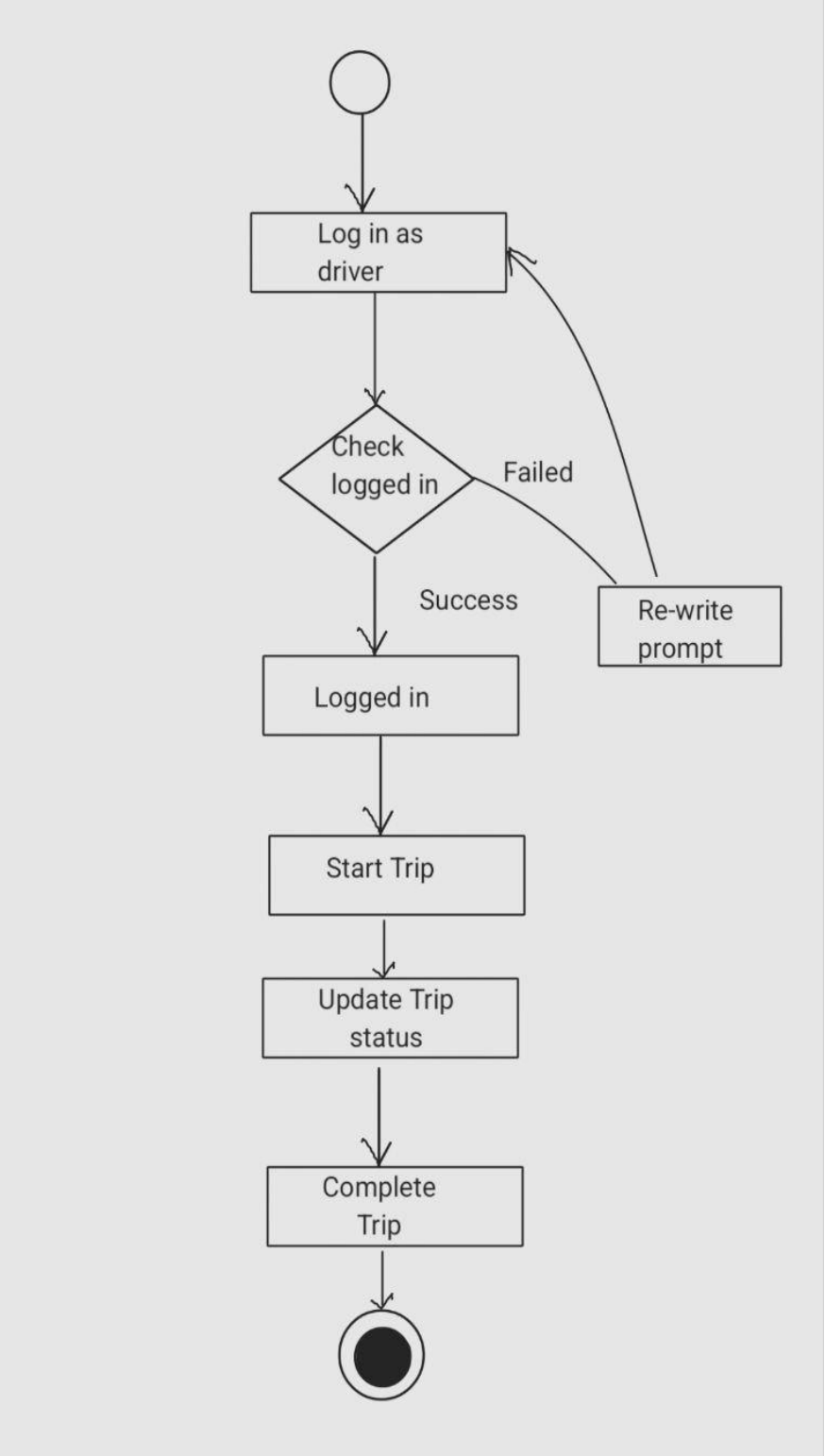
Name: Bus Tracking
Reference: Use Case Level 1.5



Level 1.5 - Bus Tracking

Level 1.6:

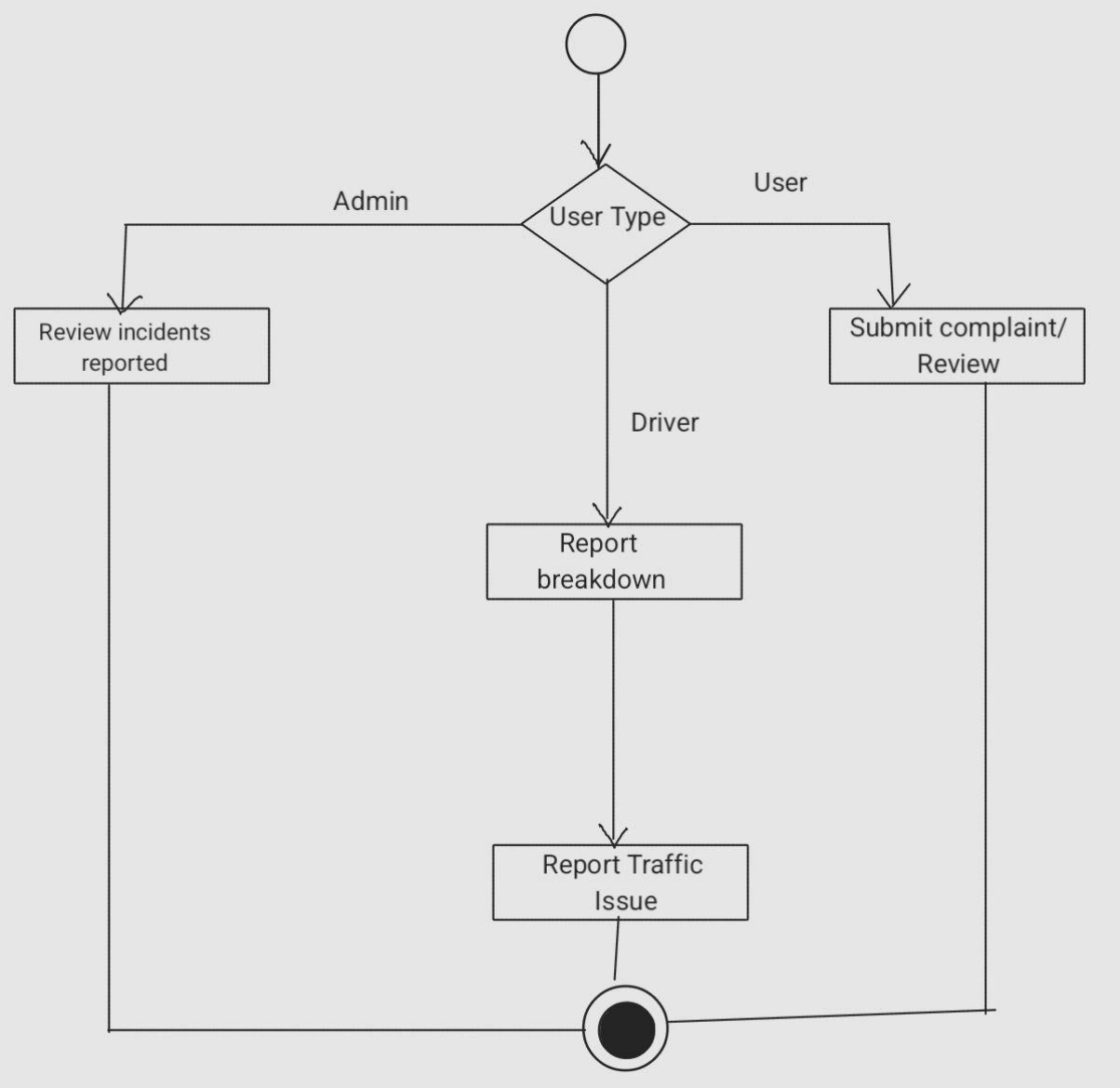
Name: Trip Operations
Reference: Use Case Level 1.6



Level 1.6 - Trip Operations

Level 1.7:

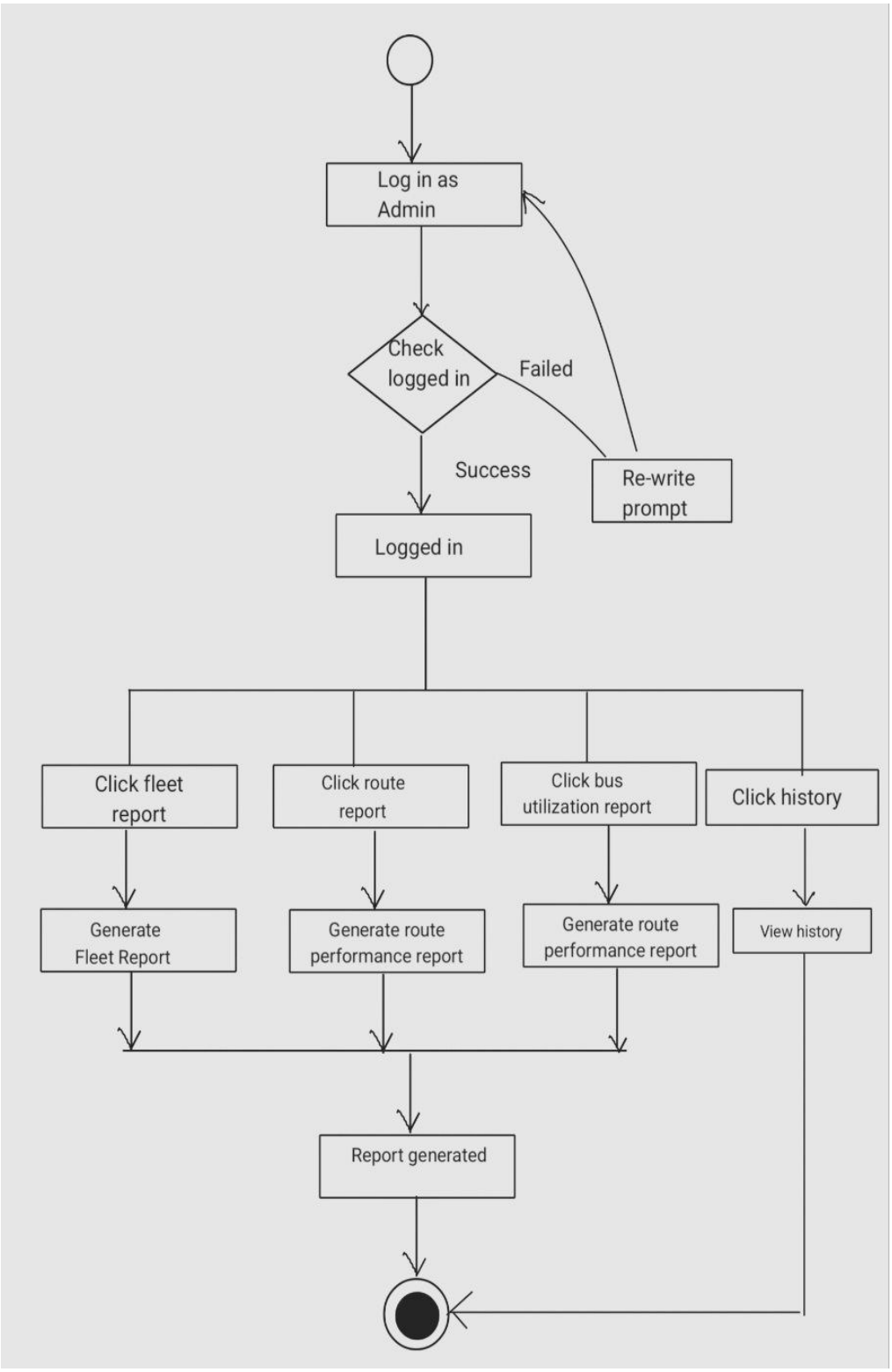
Name: Incident Reporting
Reference: Use Case Level 1.7



Level 1.7 - Incident Reporting

Level 1.8:

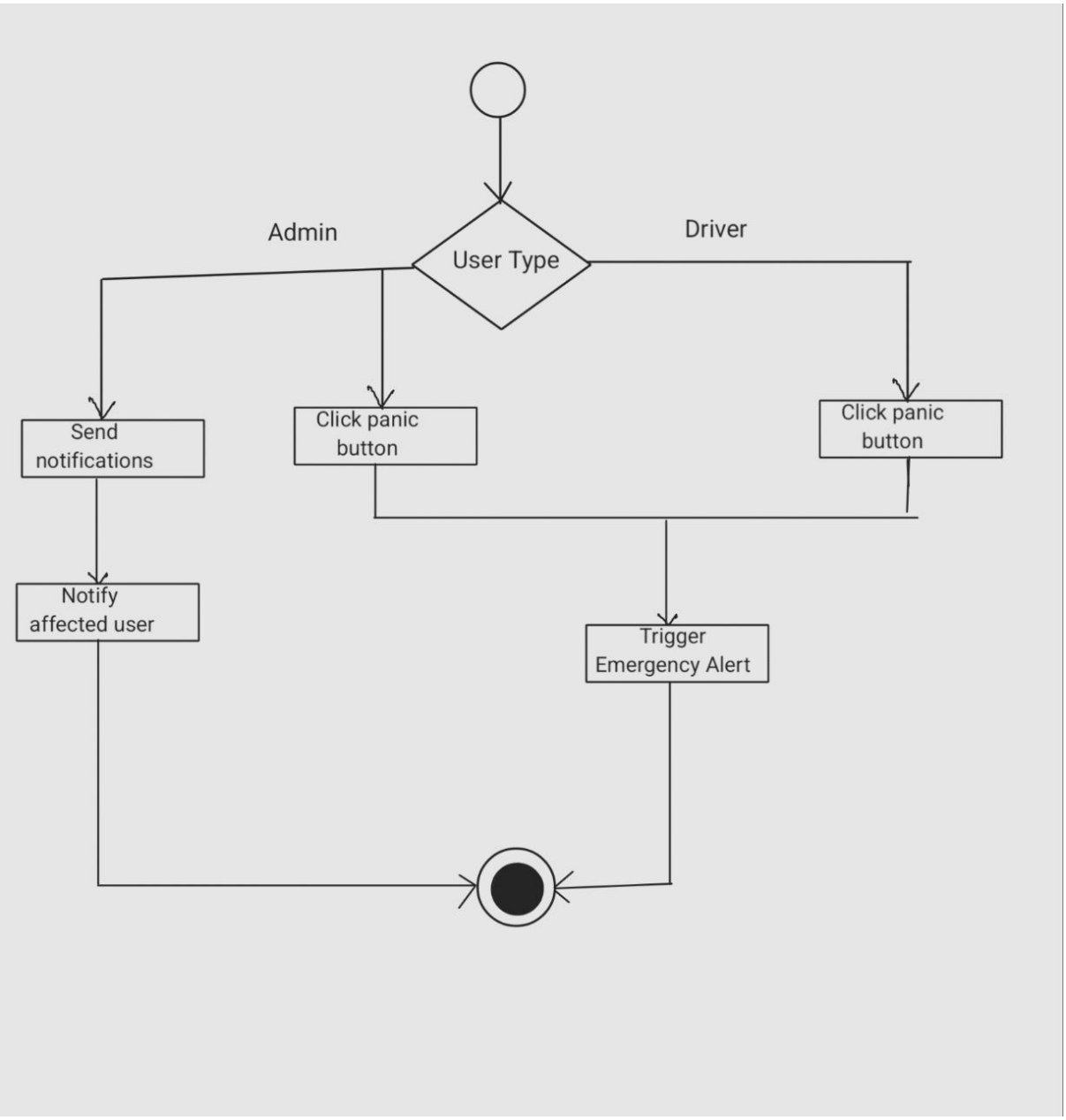
Name: Reports & Analytics
Reference: Use Case Level 1.8



Level 1.8 - Reports & Analytics

Level 1.9:

Name: Emergency Management
Reference: Use Case Level 1.9



Level 1.9 - Emergency Management

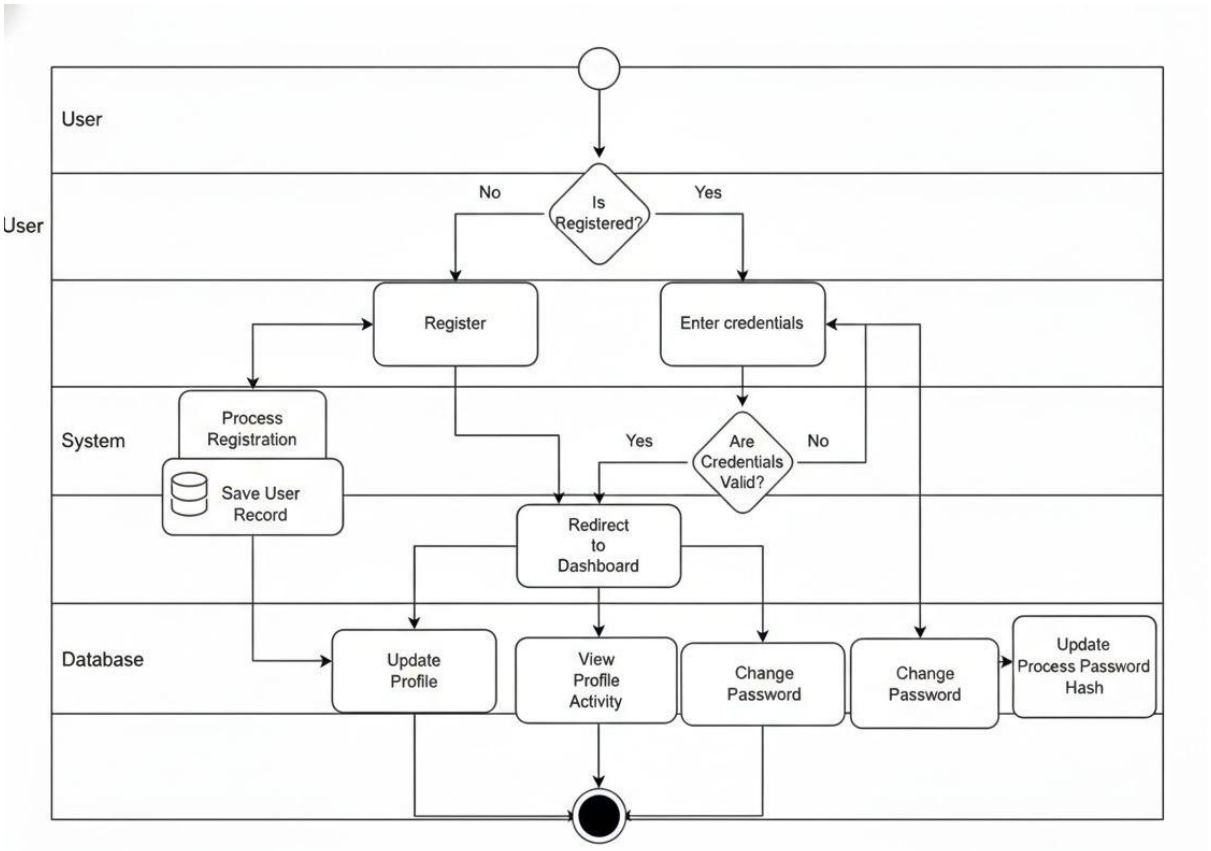
Swimlane Diagram

The swimlane diagrams illustrate the interaction between different **actors** (Student, Faculty, Driver, Admin, and System) and the **BUTrace** across its major functional modules. Each swimlane represents a specific actor and highlights their responsibilities, actions, and system interactions for a given use case. The diagrams are organized based on Level 1 use cases to provide clarity and traceability to the system requirements.

Level 1.1:

Name: Authentication (Sign Up, Sign In, Sign Out, Change Password)
Reference: Use Case Level 1.1

This swimlane diagram describes the authentication process for all system users. It shows how Students, Faculty, Drivers, and Admins interact with the system to create accounts, log in, log out, and manage password changes. The system validates credentials, manages sessions, and ensures secure access based on user roles.

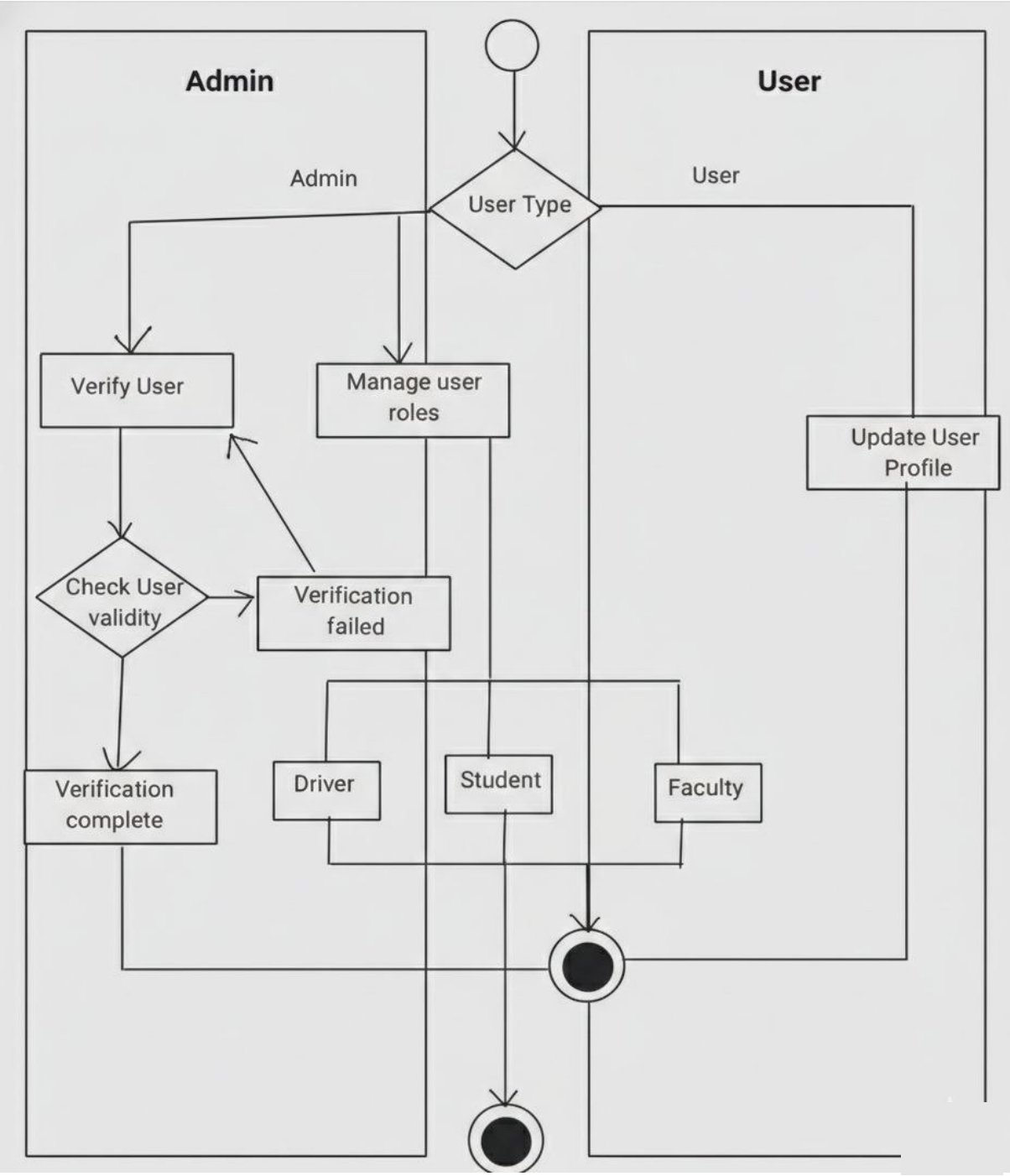


Level-1.1: User Registration, Login, and Password Management

Level 1.2:

Name: User Management
Reference: Use Case Level 1.2

This diagram illustrates administrative control over user accounts. It highlights how the Admin verifies user accounts, manages user roles, and oversees profile updates, while Students, Faculty, and Drivers can update their personal information. The system ensures role-based access and maintains accurate user records.

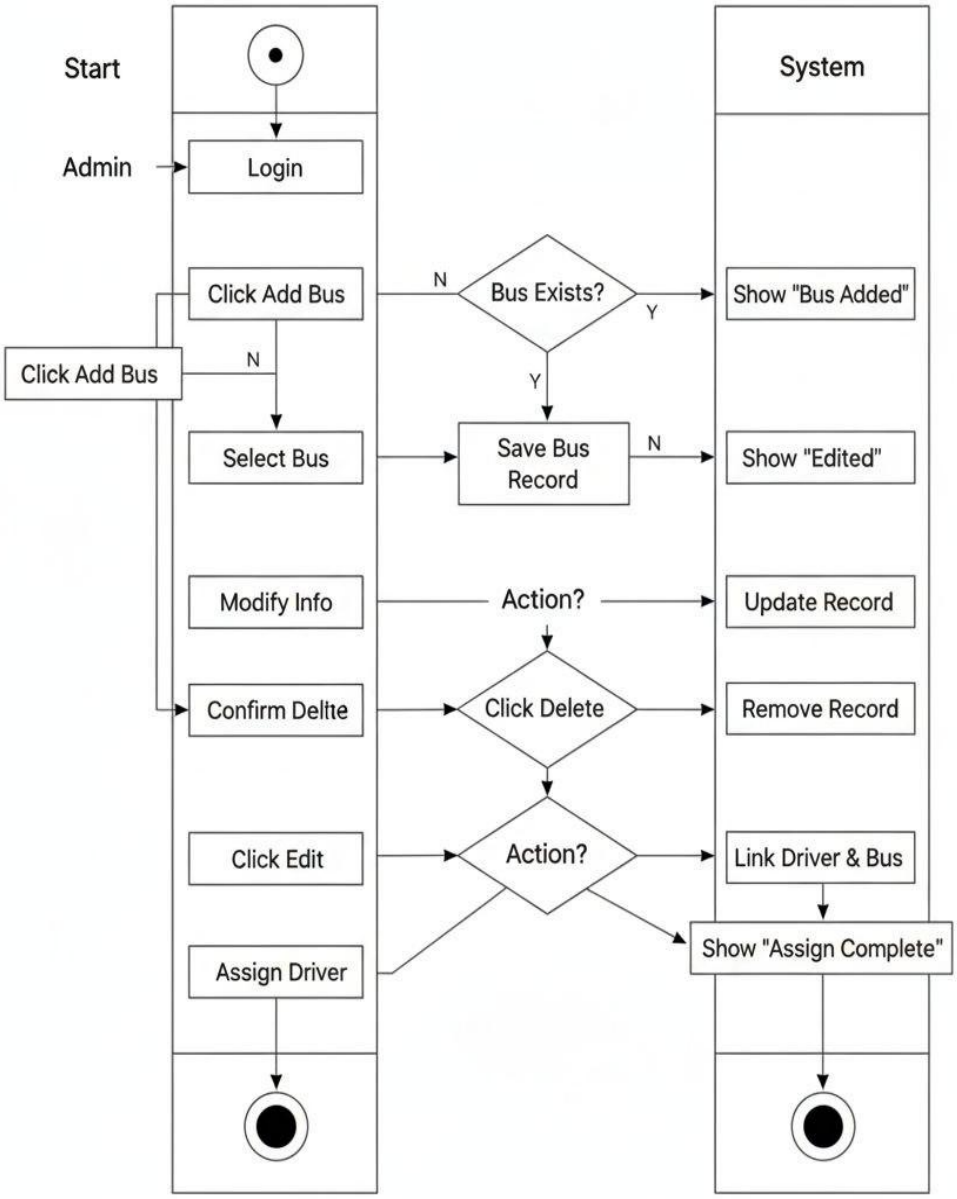


Level 1.2 - User Management

Level 1.3:

Name: Bus & Fleet Management
Reference: Use Case Level 1.3

The swimlane diagram for this level shows the Admin's responsibilities in managing the bus fleet. It includes adding new buses, editing bus information, removing buses, and assigning drivers. The system records and updates fleet data to ensure proper allocation and availability of resources.

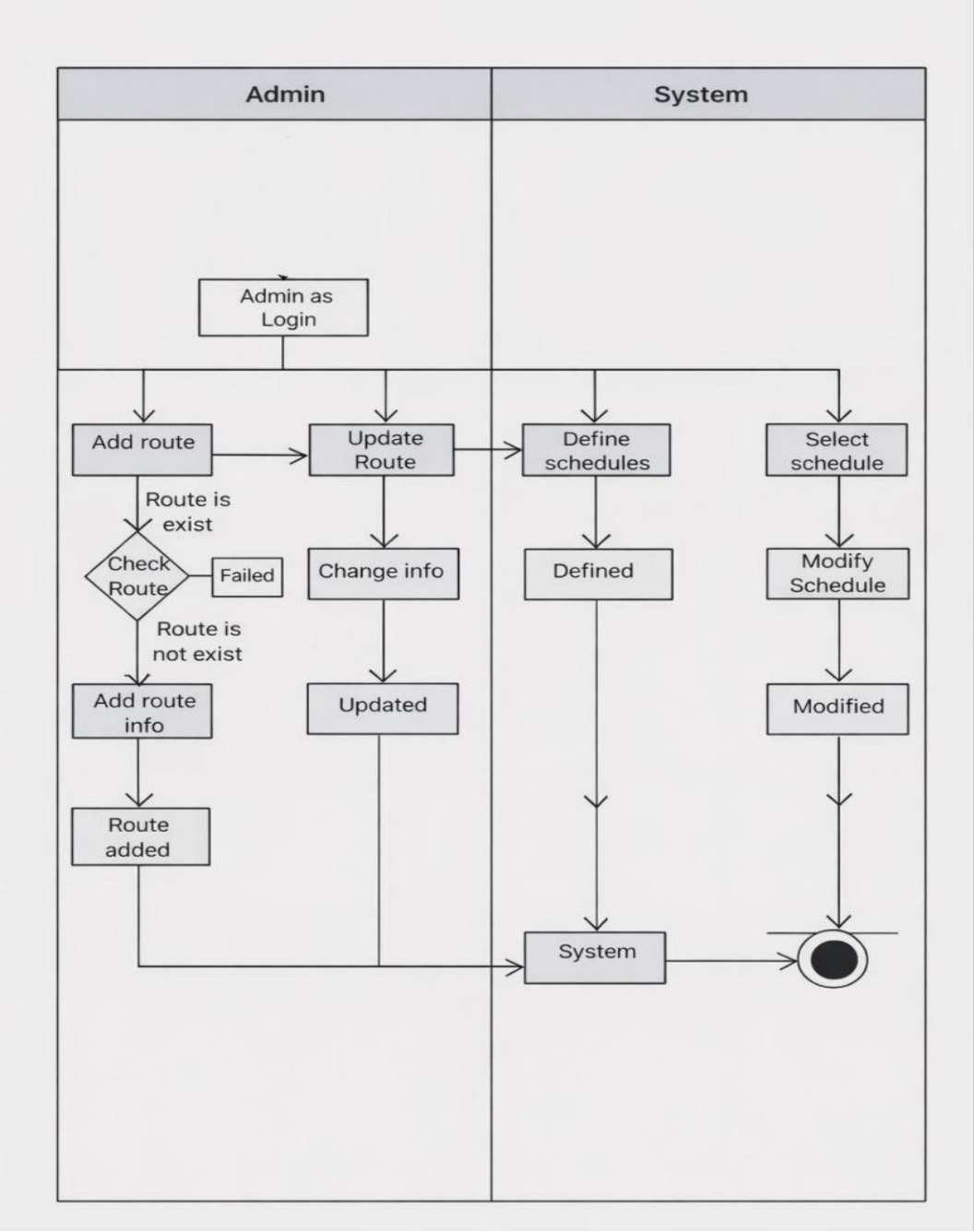


Level 1.3 - Bus & Fleet Management

Level 1.4:

Name: Route & Schedule Management
Reference: Use Case Level 1.4

This diagram represents how the Admin defines, updates, and modifies bus routes and schedules. It demonstrates the system's role in storing route data and making schedules available for tracking and trip operations, ensuring efficient transportation planning.

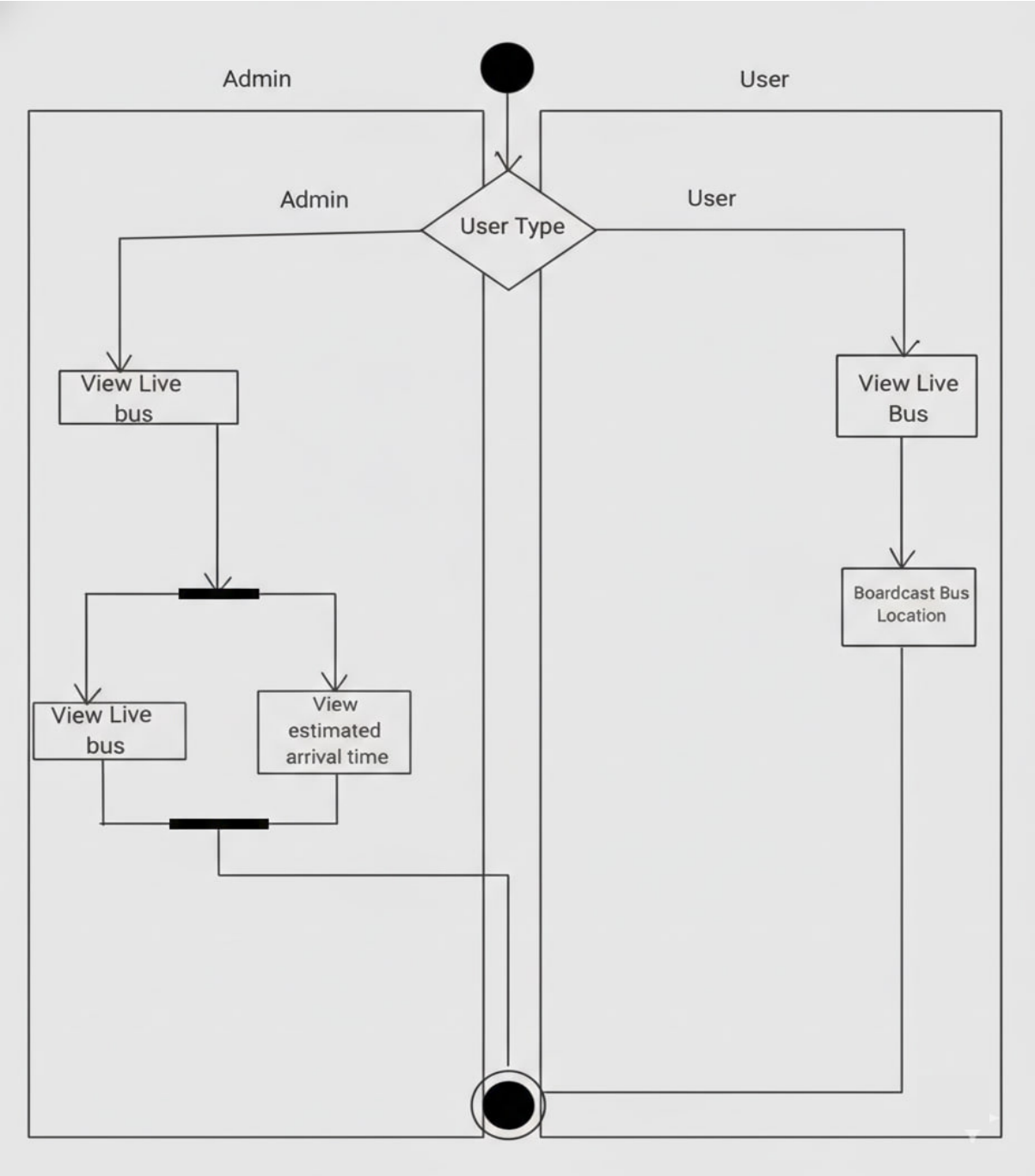


Level 1.4 - Route & Schedule Management

Level 1.5:

Name: Bus Tracking
Reference: Use Case Level 1.5

The bus tracking swimlane diagram shows real-time interaction between Drivers, Users, and the System. Drivers broadcast live bus locations, while Students, Faculty, and Admins view live tracking information and estimated arrival times. The system continuously updates and displays location data.

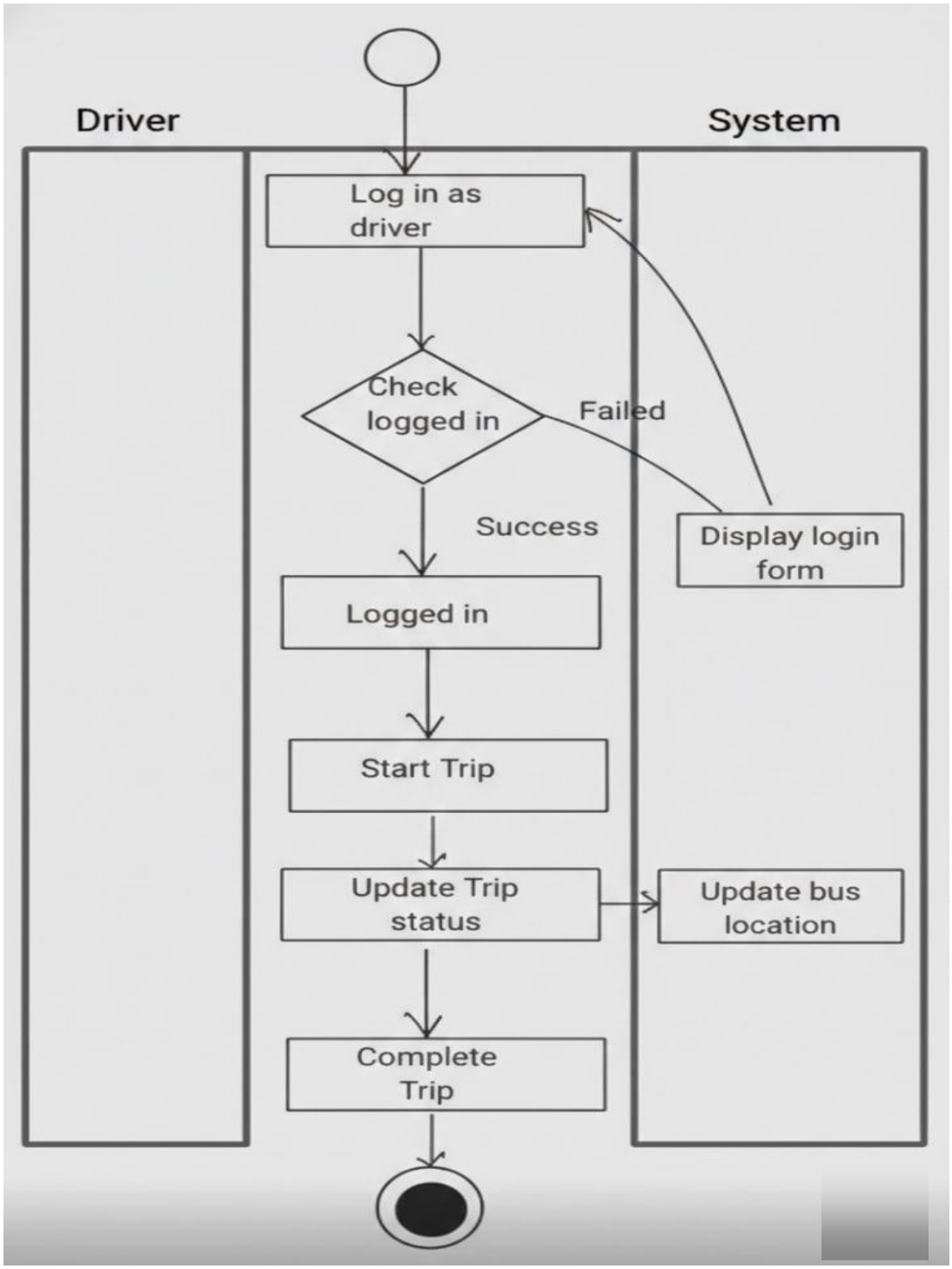


Level 1.5 - Bus Tracking

Level 1.6:

Name: Trip Operations
Reference: Use Case Level 1.6

This diagram focuses on the Driver's role in managing trips. It includes starting a trip, updating trip status, and completing a trip. The system records trip progress and updates availability for tracking, reporting, and analytics purposes.

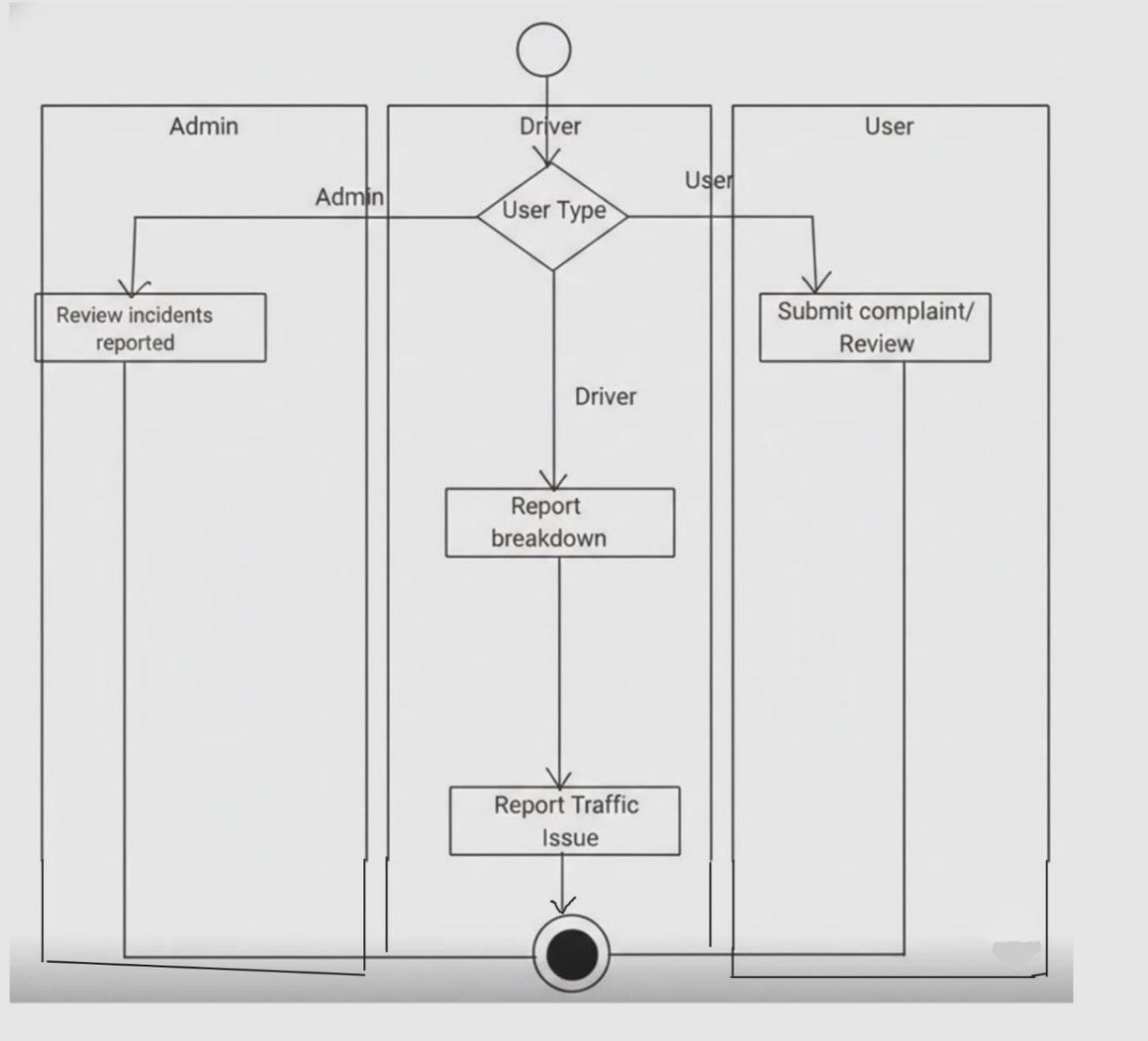


Level 1.6 - Trip Operations

Level 1.7:

Name: Incident Reporting
Reference: Use Case Level 1.7

The swimlane diagram for incident reporting illustrates how Drivers report breakdowns and traffic issues, while Students and Faculty submit complaints or feedback. The Admin reviews and manages reported incidents, and the system logs all reports for monitoring and resolution.

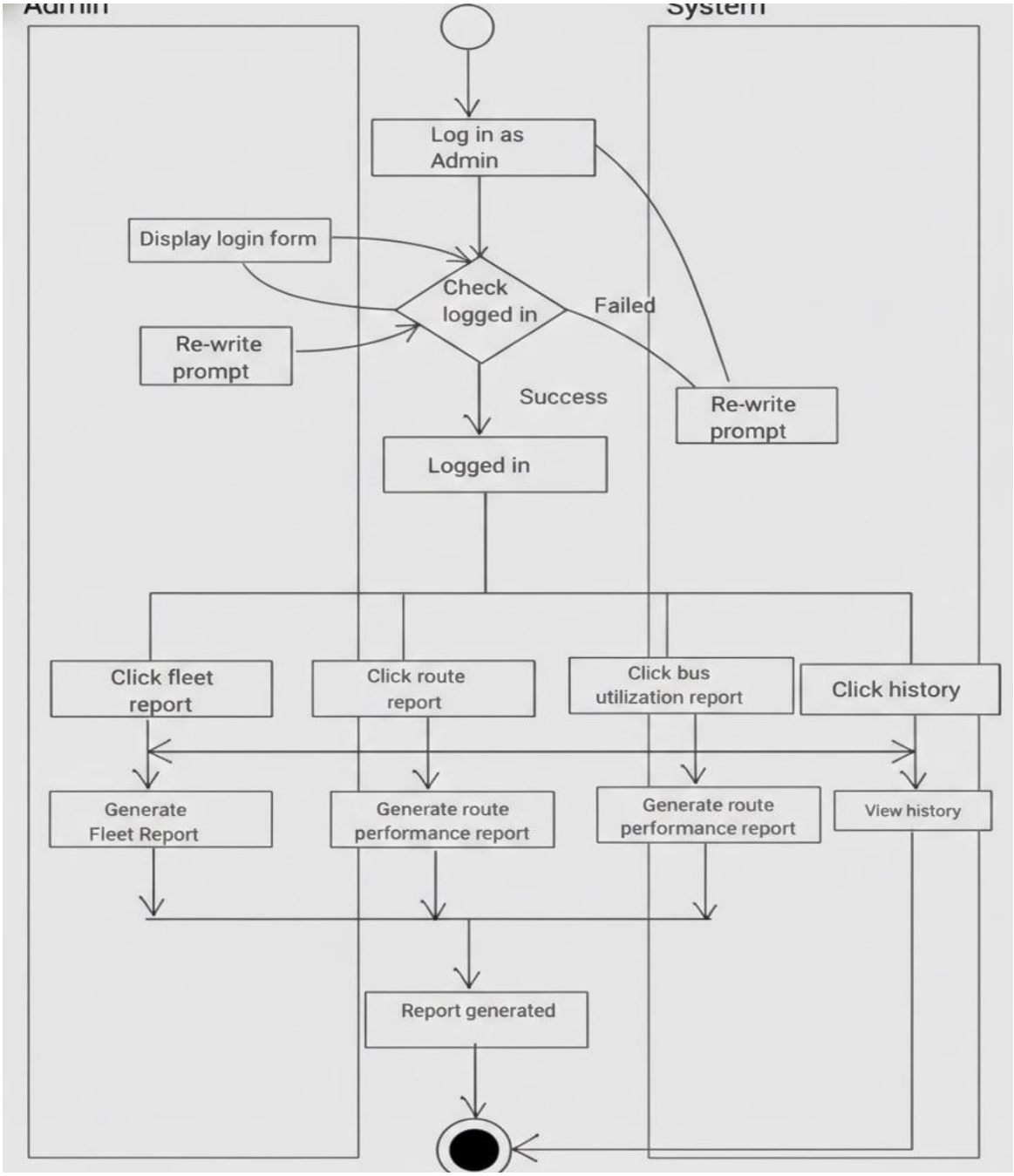


Level 1.7 - Incident Reporting

Level 1.8:

Name: Reports & Analytics
Reference: Use Case Level 1.8

This diagram shows the Admin generating analytical reports such as fleet reports, route performance reports, and bus utilization reports. It also includes viewing historical data. The system processes stored data and presents meaningful insights to support decision-making.

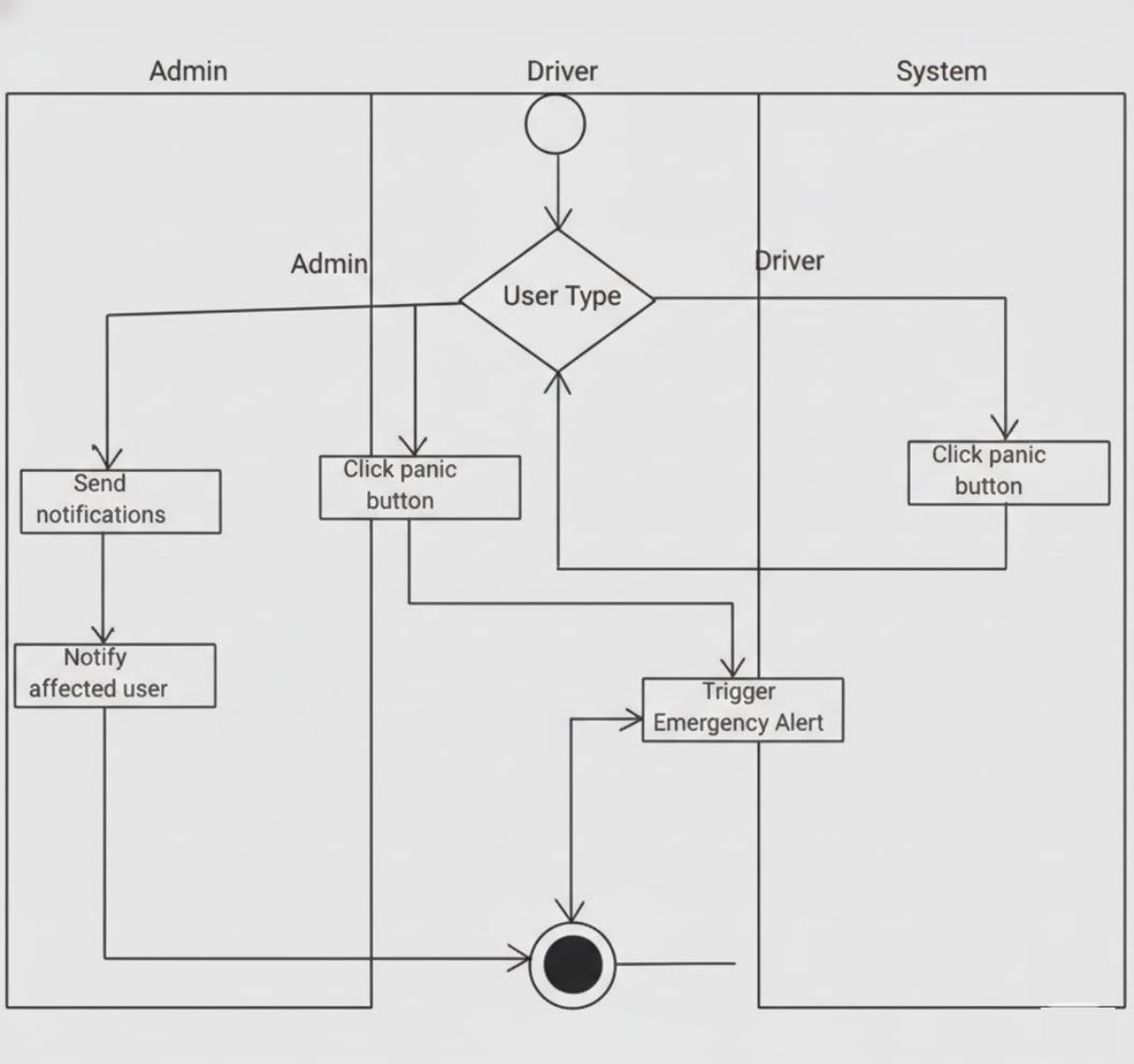


Level 1.8 - Reports & Analytics

Level 1.9:

Name: Emergency Management
Reference: Use Case Level 1.9

The emergency management swimlane diagram illustrates how Drivers or Admins trigger emergency alerts during critical situations. The system automatically notifies affected users, ensuring timely communication and improved safety response.



Level 1.9 - Emergency Management

Relationship between objects

Figure 1 (A): Entity Relationships Description

The diagram represents the **Entity-Relationship (ER) model** of a **University Bus Transport Management System**. It shows the relationships among different entities such as User, Admin, Driver, Bus, Route, Schedule, Notifications, Issue, Stoppage, and Bus Location.

1. User Relationships

1.1 User — *Receives* — Notifications

- A **User** receives **Notifications**.
- One user can receive multiple notifications.
- Each notification belongs to a specific user.
- **Relationship Type:** One-to-Many (1:N)

1.2 User — *Is_a* — Driver

- A **Driver** is a specialized type of **User**.
- This represents **generalization/specialization (inheritance)**.
- Every driver must be a user, but not every user is a driver.

1.3 User — *Is_a* — Admin

- An **Admin** is also a specialized type of **User**.
- Admin inherits all attributes of User.
- Not all users are admins.

1.4 User — *Views* — Bus Location

- A **User** can view the **Bus Location**.
- Multiple users can view the same bus location.
- **Relationship Type:** Many-to-Many (M:N)

1.5 User — *Reports* — Issue

- A **User** can report an **Issue**.
- One user can report multiple issues.
- Each issue is reported by one user.
- **Relationship Type:** One-to-Many (1:N)

2. Bus Relationships

2.1 Bus — *Generated_for* — Notifications

- A **Bus** generates notifications (e.g., delay, arrival, breakdown).
- One bus can generate multiple notifications.
- Each notification is related to a specific bus.
- **Relationship Type:** One-to-Many (1:N)

2.2 Driver — *Drives* — Bus

- A **Driver** drives a **Bus**.
- One driver may drive one or multiple buses (depending on system design).
- A bus is driven by one driver at a time.
- **Relationship Type:** One-to-Many or One-to-One (depending on implementation)

2.3 Bus — *Runs_on* — Route

- A **Bus** runs on a specific **Route**.
- One route can have multiple buses.
- Each bus runs on one route.
- **Relationship Type:** Many-to-One (N:1)

2.4 Bus — *Follows* — Schedule

- A **Bus** follows a specific **Schedule**.
- One schedule can be followed by multiple buses.
- Each bus follows one schedule.
- **Relationship Type:** Many-to-One (N:1)

2.5 Bus — *Picked_at* — Stoppage

- A **Bus** picks up passengers at multiple **Stoppages**.
- One stoppage can serve multiple buses.
- **Relationship Type:** Many-to-Many (M:N)

2.6 Bus — *Tracked_at* — Bus Location

- A **Bus** is tracked at a specific **Bus Location**.
- One bus can have multiple location records (real-time tracking).
- Each location record belongs to one bus.
- **Relationship Type:** One-to-Many (1:N)

3. Admin Relationships

3.1 Admin — *Manages* — Route

- An **Admin** manages routes.
- One admin can manage multiple routes.
- **Relationship Type:** One-to-Many (1:N)

3.2 Admin — *Creates* — Schedule

- An **Admin** creates schedules.
- One admin can create multiple schedules.
- **Relationship Type:** One-to-Many (1:N)

3.3 Admin — *Manages* — Bus

- An **Admin** manages buses.
- One admin can manage multiple buses.
- **Relationship Type:** One-to-Many (1:N)

4. Issue Relationships

4.1 Issue — *Related_to* — Bus

- An **Issue** is related to a specific **Bus**.
- One bus can have multiple issues.
- Each issue is associated with one bus.
- **Relationship Type:** Many-to-One (N:1)

5. Route and Schedule Relationship

5.1 Route — *Has* — Schedule

- A **Route** has one or more **Schedules**.
- Each schedule belongs to one route.
- **Relationship Type:** One-to-Many (1:N)

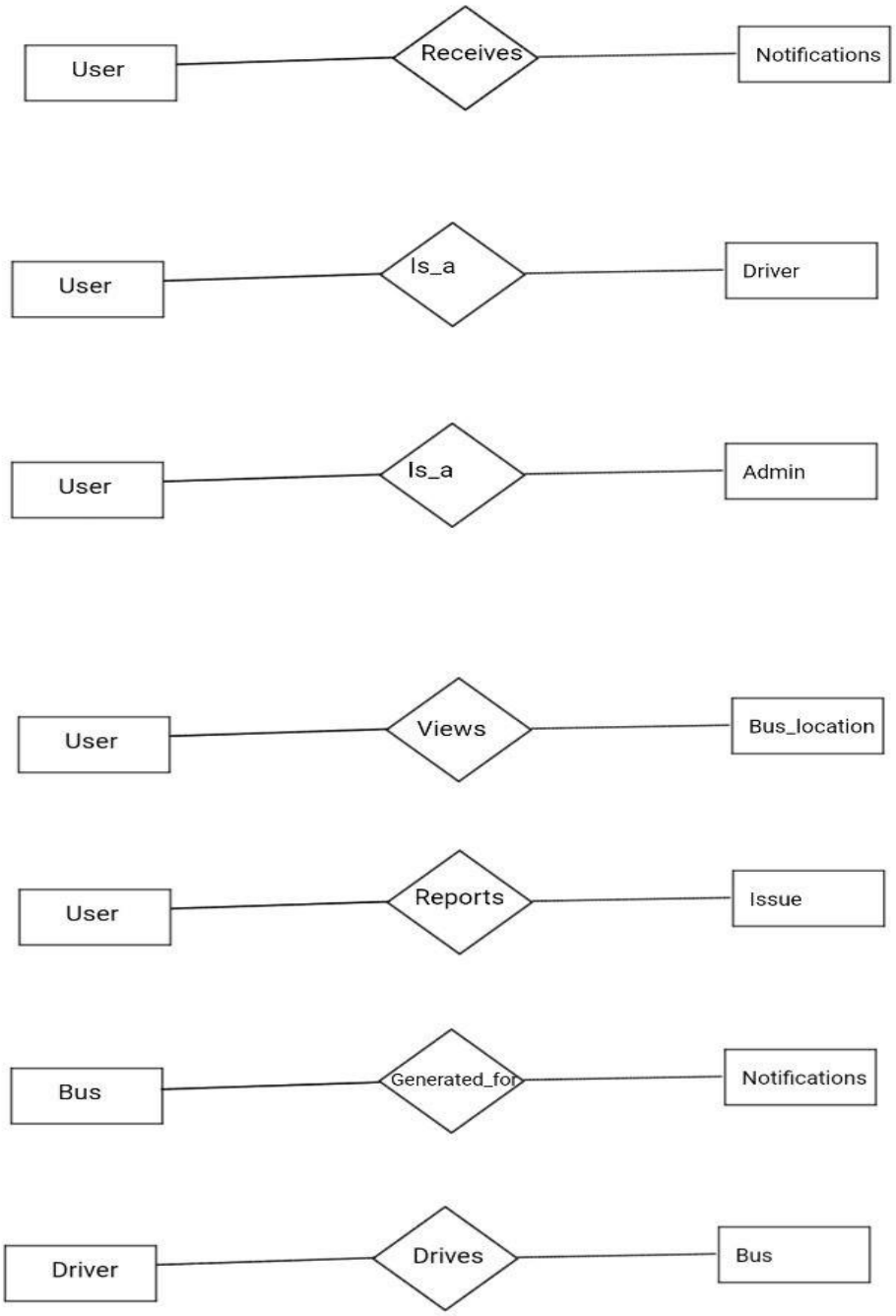


Figure 1 (A): Entity Relations

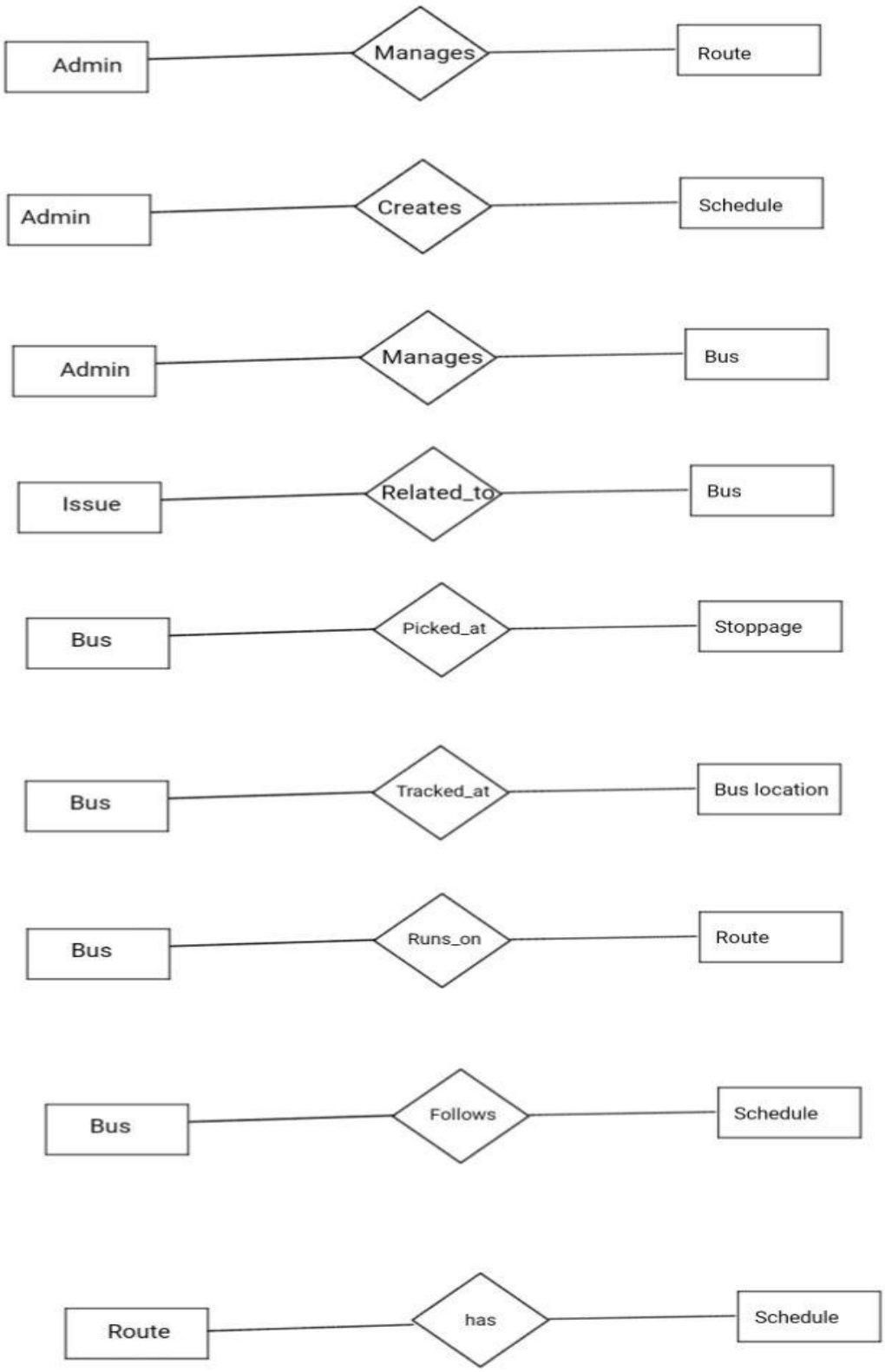


Figure 1 (A): Entity Relations

Entity Relation Diagram

Figure 1: Entity Relationship Diagram (ERD) Description
University Bus Transport Management System

1. User
The User entity represents all individuals who access the system.

Attributes:
1.1 name
1.2 email
1.3 password
1.4 role

Relationships:
1.5 A user receives notifications.
1.6 A user reports issues.
1.7 A user views bus locations.
1.8 User is specialized into Admin and Driver.

2. Admin
Admin is a specialized entity derived from User.

Relationships:
2.1 An admin manages routes.
2.2 An admin manages buses.
2.3 An admin creates schedules.

Admin controls overall system operations including route, bus, and schedule management.

3. Driver
Driver is also a specialized entity derived from User.

Relationships:
3.1 A driver drives a bus.

The driver operates assigned buses in the system.

4. Bus
The Bus entity represents vehicles operating in the transport system.

Attributes:
4.1 name
4.2 seat_availability

Relationships:
4.3 A bus runs on a route.
4.4 A bus follows a schedule.
4.5 A bus is parked at a stoppage.

4.6 A bus is tracked at a bus location.
4.7 A bus generates notifications.
4.8 A bus is related to issues.
4.9 A bus is managed by an admin.
4.10 A bus is driven by a driver.

5. Route
The Route entity represents predefined paths followed by buses.

Attributes:
5.1 name
5.2 status

Relationships:
5.3 A route has multiple stoppages.
5.4 A route has multiple schedules.
5.5 A route is managed by an admin.
5.6 Buses run on routes.

6. Stoppage
The Stoppage entity represents pickup and drop-off locations.

Attributes:
6.1 name
6.2 latitude
6.3 longitude

Relationships:
6.4 A stoppage belongs to a route.
6.5 Buses are parked at stoppages.

7. Schedule
The Schedule entity represents bus timing information.

Attributes:
7.1 start_time
7.2 departure_time
7.3 day

Relationships:
7.4 A schedule is created by an admin.
7.5 A schedule belongs to a route.
7.6 A schedule is followed by buses.

8. Bus_Location
The Bus_Location entity stores real-time tracking information.

Attributes:
8.1 route_id
8.2 bus_id

8.3 latitude
8.4 longitude
8.5 speed

Relationships:
8.6 A bus location record belongs to a bus.
8.7 Bus locations are viewed by users.

9. Issue
The Issue entity represents complaints or reported problems.

Attributes:
9.1 title
9.2 description
9.3 status

Relationships:
9.4 An issue is reported by a user.
9.5 An issue is related to a bus.

10. Notifications
The Notifications entity represents alerts generated by the system.

Attributes:
10.1 title
10.2 message
10.3 status

Relationships:
10.4 Notifications are received by users.
10.5 Notifications are generated for buses.

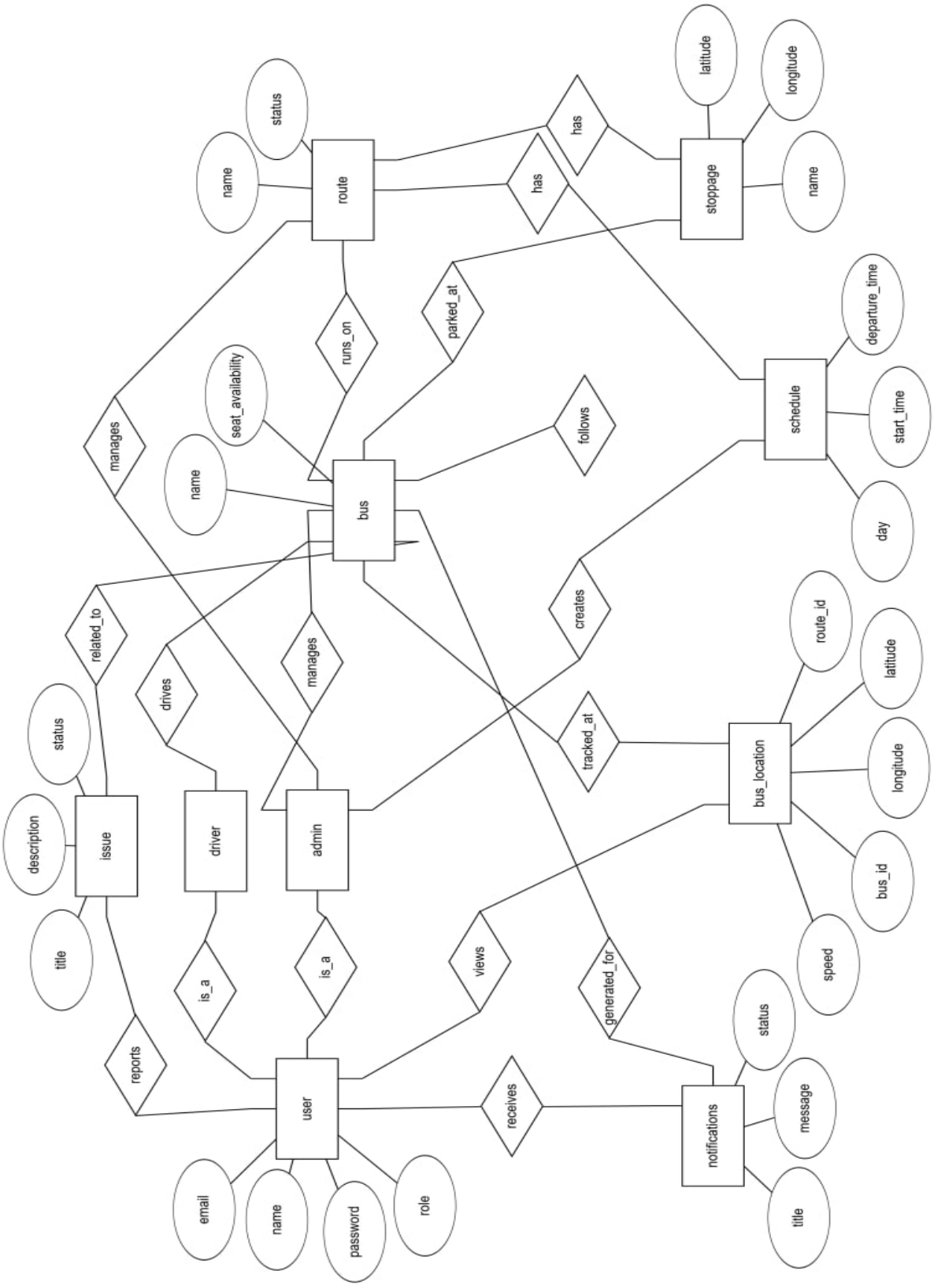


Figure 2: ER Diagram

Class-Based Modeling

The Class-Based Modeling approach identifies the major classes in the solution space of the University Bus Transport Management System. It categorizes system elements based on their roles, responsibilities, attributes, and operations. This modeling technique ensures structured design, modular development, and clear separation of concerns.

Class-Based Modeling Codes and Meaning

1 – External System/Service
2 – Attribute/Data
3 – Process/Operation
4 – Actor/User Role
5 – Core Domain Entity
6 – Control/Transaction
7 – Information/Record

List of Nouns in the Solution Space and Their General Classification

The solution space analysis identifies key nouns that represent actors, entities, services, and records within the system. These nouns are classified based on their roles in the system architecture.

For example, User, Student, Faculty, Driver, and Admin are classified as Actor/User Role (4), Core Domain Entity (5), and Information/Record (7) because they represent system participants and maintain persistent information.

Entities such as Bus, Route, and Trip are classified as Core Domain Entities (5), Control/Transaction (6), and Information/Record (7) since they form the operational backbone of the transport system.

Items like GPS are categorized as External System/Service (1) because they represent third-party tracking services.

Processes such as Authentication, Complaint, Notification, and Report are classified under Process/Operation (3) and Control/Transaction (6), as they handle system actions and workflows.

Potential Classes Identified

After analyzing the nouns, the following potential classes were identified:

1. User
2. Student
3. Faculty
4. Driver
5. Admin
6. Bus
7. Route

8. Schedule
9. Trip
10. Incident
11. Notification
12. Emergency
13. Complaint
14. Account

Selection Criteria

The following criteria were used to finalize the classes:

1. Retain information
2. Needed services
3. Multiple attributes
4. Common attributes
5. Common operations
6. Essential requirements

Only the classes satisfying most of these criteria were retained in the final design.

Final List of Classes

1. User
2. Student
3. Faculty
4. Driver
5. Admin
6. Bus
7. Route
8. Schedule
9. Trip
10. Incident
11. Notification
12. Complaint

These classes represent the structural and behavioral foundation of the system.

Class-Based Modeling Code Description

In Class-Based Modeling, system elements are categorized using numerical codes to clearly define their roles and responsibilities within the system. Code 1 represents External Systems or Services that interact with the system but are not internally controlled, such as GPS services. Code 2 refers to Attributes or Data, which store descriptive information about entities. Code 3 denotes Processes or Operations, meaning the actions or functions performed by the system. Code 4 identifies Actors or User Roles who interact with the system, such as students or administrators. Code 5 represents Core Domain Entities, which are the main business objects central to the system's functionality. Code 6 indicates Control or Transaction elements that manage workflows and system operations. Finally, Code 7 represents Information or Records, which store persistent data for tracking and documentation purposes.

Code	Meaning
1	External System/Service
2	Attribute/Data
3	Process/Operation
4	Actor/User Role
5	Core Domain Entity
6	Control/Transaction
7	Information/Record

List of Nouns in the Solution Space and Their General Classification:

Sl. No	Noun	General Classification
1	User	4, 5, 7
2	Student	4, 5, 7
3	Faculty	4, 5, 7
4	Driver	4, 5, 7
5	Admin	4, 5, 7
6	Bus	5, 6, 7
7	Route	5, 6, 7

8	**Schedule**	**6, 7**
9	**Trip**	**5, 6, 7**
10	**Location**	**2, 6**
11	**GPS**	**1**
12	**Notification**	**3, 6**
13	**Incident**	**5, 6, 7**
14	**Emergency**	**6, 7**
15	**Complaint**	**3, 6, 7**
16	**Feedback**	**3, 6, 7**
17	**Report**	**3, 6**
18	**Account**	**5, 6**
19	**Authentication**	**3**
20	**Dashboard**	**7**

Potential Classes Identified

1. User

2. Student
3. Faculty
4. Driver
5. Admin
6. Bus
7. Route
8. Schedule
9. Trip
10. Incident
11. Notification
12. Emergency
13. Complaint
14. Account

Selection Criteria

1. Retain information
2. Needed services
3. Multiple attributes
4. Common attributes
5. Common operations
6. Essential requirements

Potential Class Criteria

Potential Class	**Selection Criteria**

User	**1, 2, 4, 5, 6**
Student	**1, 2, 4, 5, 6**
Faculty	**1, 2, 4, 5, 6**
Driver	**1, 2, 4, 5, 6**
Admin	**1, 2, 4, 5, 6**
Bus	**1, 2, 3, 4, 5, 6**
Route	**1, 2, 3, 4, 5, 6**
Schedule	**1, 2, 3**
Trip	**1, 2, 3, 4, 5, 6**
Incident	**1, 2, 3, 4, 6**
Notification	**1, 2, 3, 4, 6**
Emergency	**1, 2, 3, 4, 6**
Complaint	**1, 2, 3, 4**
Account	**1, 2, 3, 4**

Final List of Classes

1. **User**
2. **Student**
3. **Faculty**
4. **Driver**
5. **Admin**
6. **Bus**
7. **Route**
8. **Schedule**
9. **Trip**
10. **Incident**
11. **Notification**
12. **Complaint**

Class Cards

User	
Attributes	**Methods**
-user_id -full_name -email -mobile_number -password -role	+register() +login() +logout() +updateProfile()
Responsibilities	**Collaborators**
1. Manage user registration and authentication 2. Maintain user profile information	1. Notification

Student	
Attributes	**Methods**
-student_id -department	+viewBusSchedule() +trackBus() +submitComplaint()
Responsibilities	**Collaborators**
1. View bus routes, schedules, and live tracking 2. Submit complaints and feedback	1. Trip 2. Notification 3. Complaint

Faculty	
Attributes	**Methods**
-faculty_id -designation	+viewBusSchedule() +trackBus()
Responsibilities	**Collaborators**
1. Access reliable real-time transport information	1. Trip 2. Notification

Driver	
Attributes	**Methods**
-driver_id -license_number	+startTrip() +updateTripStatus() +reportIncident()
Responsibilities	**Collaborators**
1. Operate assigned buses 2. Report incidents and emergencies	1. Bus 2. Trip 3. Incident

Admin	
Attributes	**Methods**
-admin_id	+manageUsers() +manageFleet() +generateReport()
Responsibilities	**Collaborators**
1. Oversee system operations 2. Manage users, buses, and routes	1. Bus 2. Route 3. Report

Bus	
Attributes	**Methods**
-bus_id -registration_number -capacity -status	+assignDriver() +updateStatus()
Responsibilities	**Collaborators**
1. Maintain bus details 2. Support trip execution	1. Driver 2. Route 3. Trip

Route	
Attributes	**Methods**
-route_id -route_name	+updateRoute()
Responsibilities	**Collaborators**
1. Define and manage bus routes	1. Schedule 2. Bus

Schedule	
Attributes	**Methods**
-schedule_id -departure_time -arrival_time	+updateSchedule()
Responsibilities	**Collaborators**
1. Manage route timing and updates	1. Route 2. Notification

Trip	
Attributes	**Methods**
-trip_id -start_time -end_time -trip_status	+startTrip() +updateTripStatus() +endTrip()
Responsibilities	**Collaborators**
1. Control the trip lifecycle	1. Bus 2. Driver

Incident	
Attributes	**Methods**
-incident_id -incident_type -description -status	+reportIncident() +updateStatus()
Responsibilities	**Collaborators**
1. Record and manage transport-related incidents	1. Driver 2. Admin 3. Notification

Complaint	
Attributes	**Methods**
-complaint_id -complaint_text -status	+submitComplaint() +updateStatus()
Responsibilities	**Collaborators**
1. Handle user complaints and feedback	1. Student 2. Admin

Notification	
Attributes	**Methods**
-notification_id -message -status	+sendNotification()
Responsibilities	**Collaborators**
1. Deliver system notifications and alerts	1. User

Class Diagram

The class diagram represents the object-oriented structure of the University Bus Transport Management System. The **User** class is the base class containing common attributes and authentication methods. **Student**, **Driver**, and **Admin** inherit from User, adding specialized functionalities such as viewing schedules, operating buses, and managing the system. The **Bus** class stores vehicle details and is associated with **Route** and **Trip**. The **Route** class defines transport paths, while **Schedule** manages departure and arrival times. The **Trip** class controls the lifecycle of each journey, connecting Bus and Driver. Overall, the diagram shows inheritance, associations, and system responsibilities in a structured and modular design.

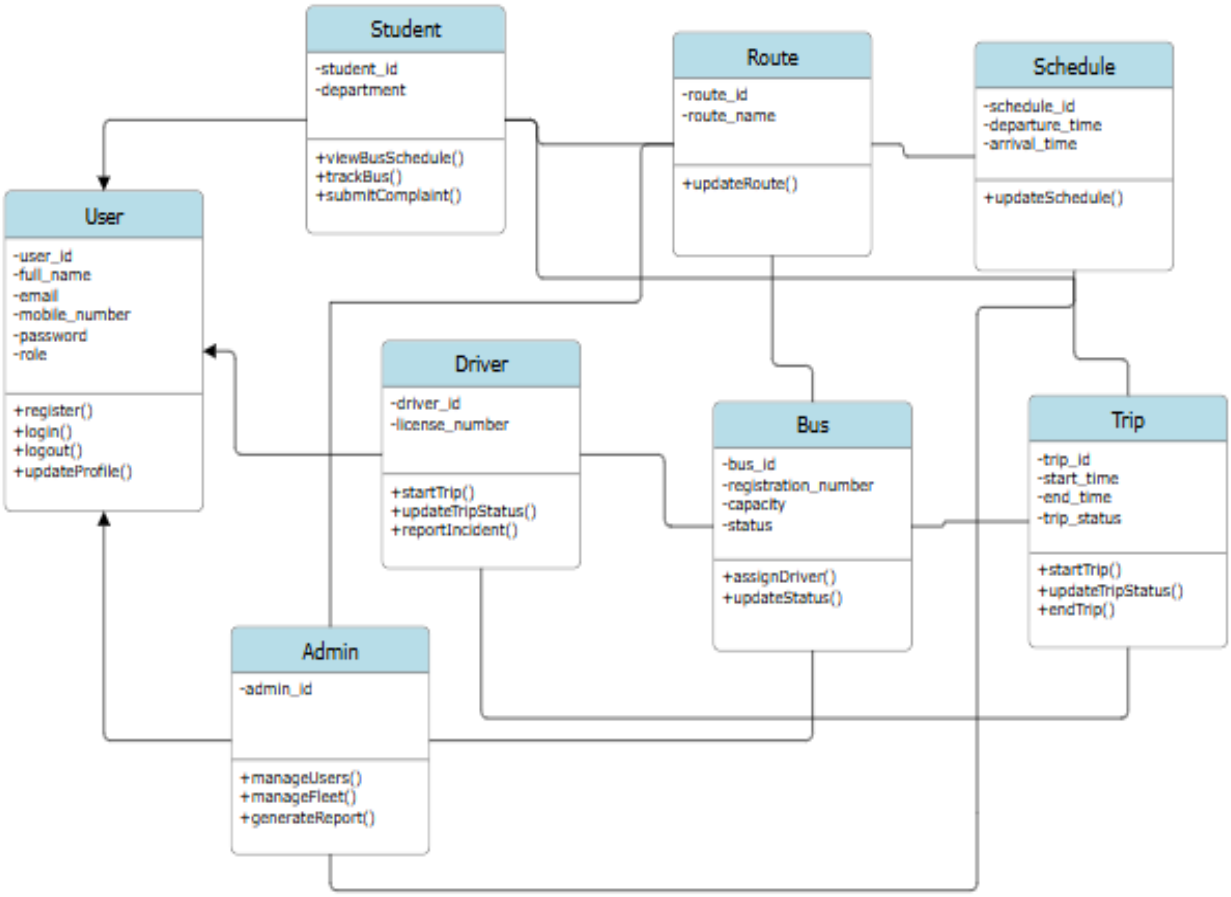


Figure: Class Diagram

Behavioural Model

Behavioural Model describes how a software system acts and reacts to external stimuli (like user inputs or system events). While functional requirements tell you *what* a system does, the behavioural model illustrates it. It can be formulated with

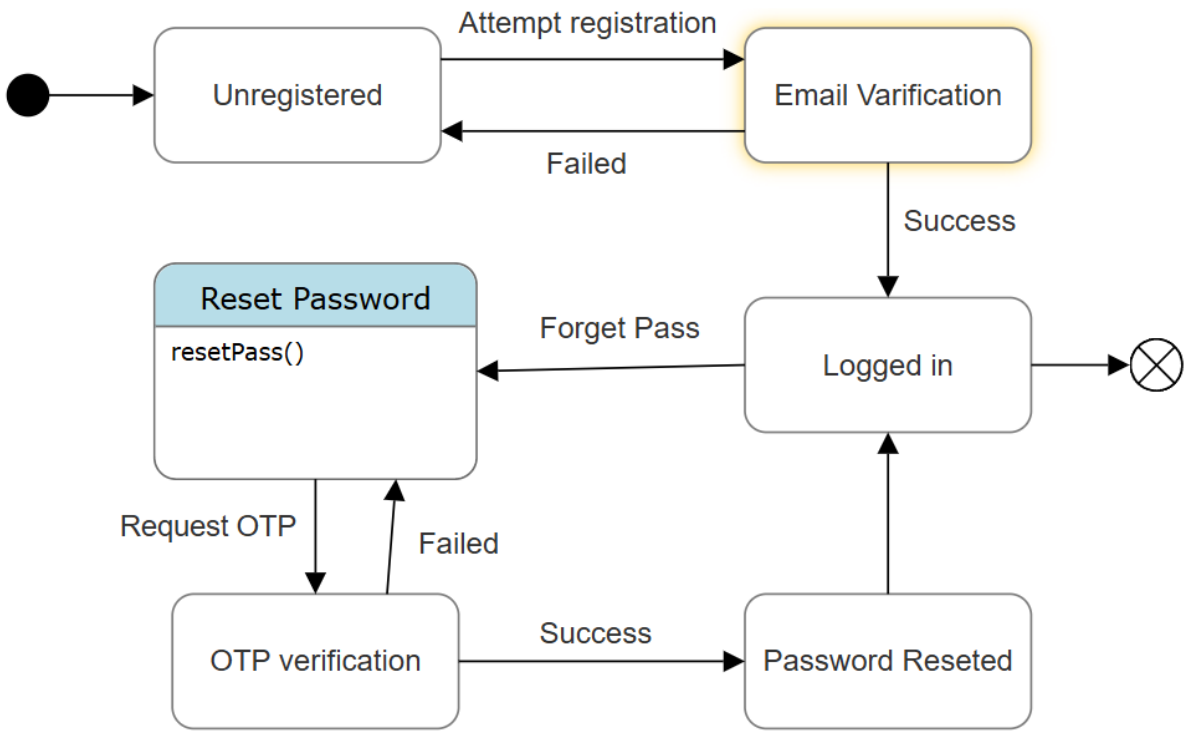
1. State Transition Diagram
2. Data Flow Diagram
3. Sequence Diagram

State Transition Diagram

It shows how the states change between each action. Major actors play roles in change of states

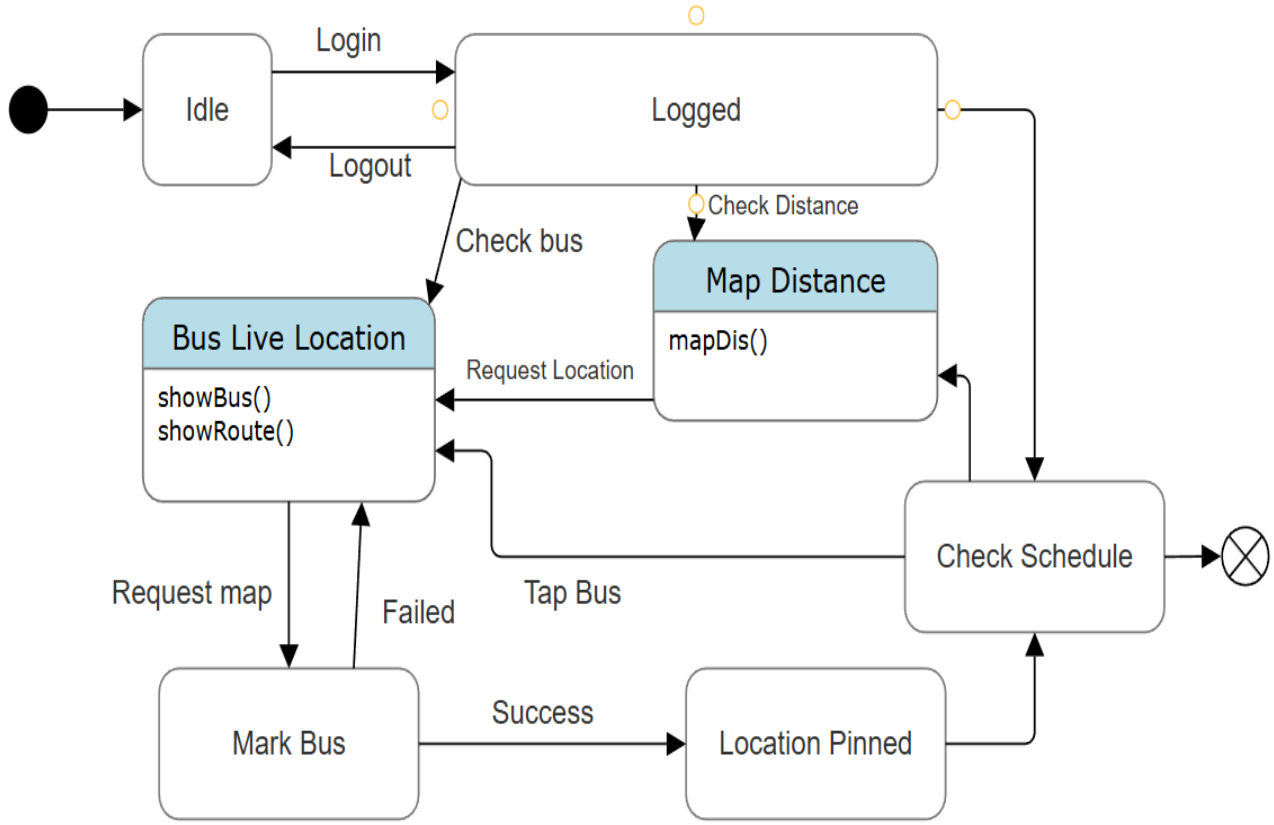
State 01: User First Steps

The state diagram illustrates the user registration and password reset process. A user starts in the **Unregistered** state and moves to **Email Verification** during registration. If verification succeeds, the user becomes **Logged In**; if it fails, they return to the Unregistered state. From the Logged In state, a user can request a password reset, leading to **Reset Password** and then **OTP Verification**. If OTP verification is successful, the system transitions to **Password Reset** and finally back to **Logged In**. If verification fails, the user returns to the Reset Password state.



State 02: Student Using BUTrace

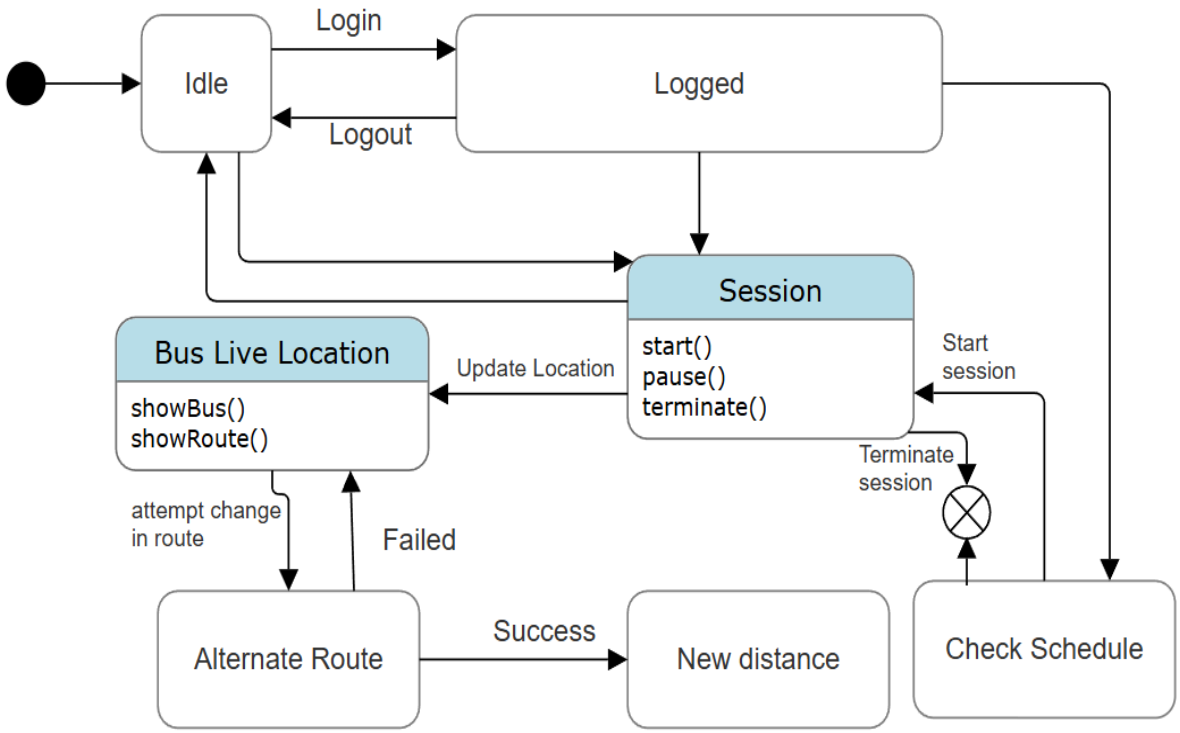
This diagram illustrates the interaction between system components for tracking buses. The user starts in an **Idle** state and logs in to enter the **Logged** state. From there, the system can **Check Bus** locations and distances via **Bus Live Location** and **Map Distance** modules. Users can **Tap Bus** to mark it, which updates **Location Pinned**. Finally, the **Check Schedule** module confirms bus timings, completing the workflow. Arrows indicate the flow of requests, responses, and state transitions.



State 03: Driver Operating with BUTrace

This diagram shows the session and route management process. The user starts in the **Idle** state and moves to **Logged** after login. From Logged, a **Session** begins where start(), pause(), and terminate() operations are managed. During the session, the system updates **Bus Live Location**.

The user can attempt an **Alternate Route**; if successful, a **New Distance** is calculated, otherwise it returns to the previous state. The user may also check the schedule, and the session ends when terminated. Overall, the diagram represents login, session control, live tracking, route change, and schedule checking flow.

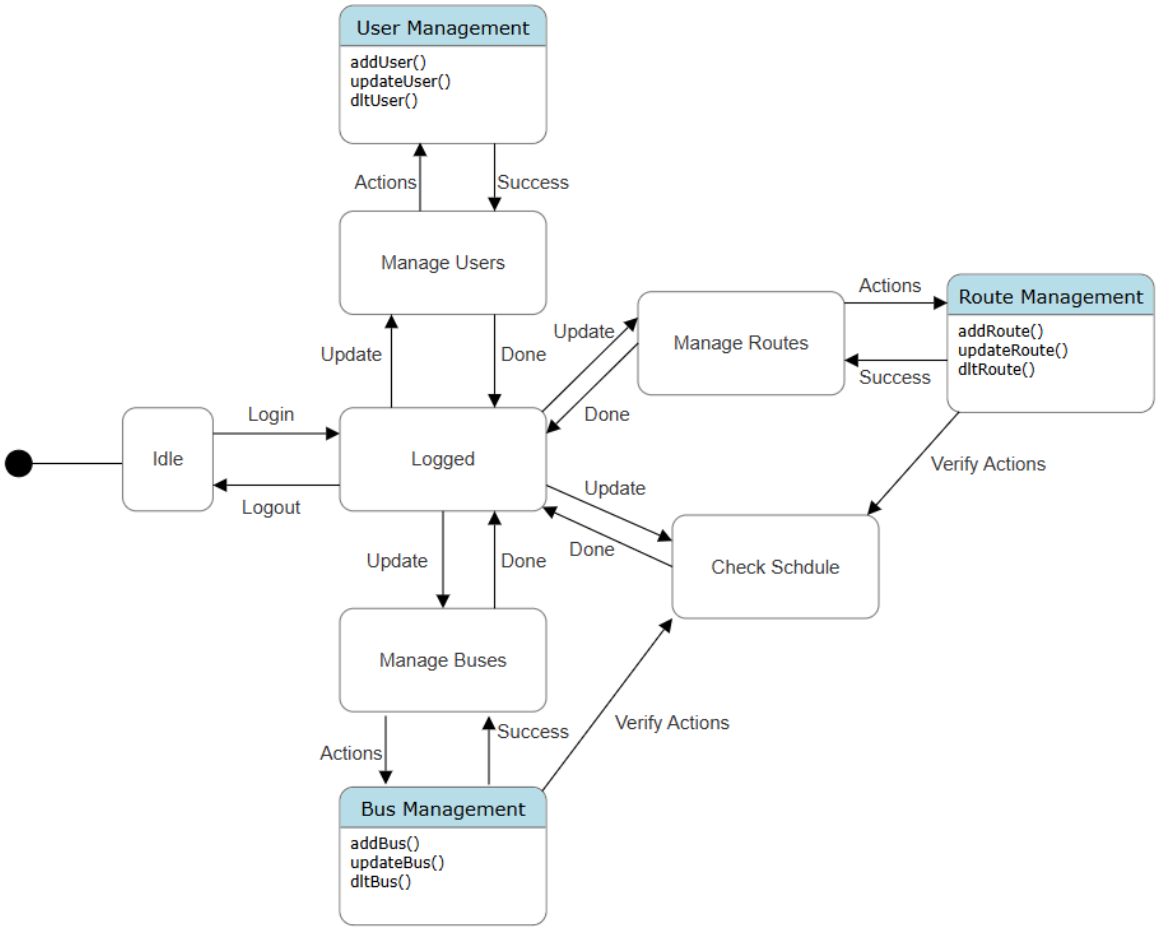


State 04: Admin Action

This diagram represents the administrative management workflow of the system. The process starts in the **Idle** state and moves to **Logged** after login. From the Logged state, the admin can manage users, routes, buses, and schedules.

Through **Manage Users**, the system performs addUser(), updateUser(), and deleteUser() operations under **User Management**. Similarly, **Manage Routes** and **Route Management** handle route-related actions, while **Manage Buses** and **Bus Management** control bus operations such as addBus(), updateBus(), and deleteBus().

Each management action requires verification and, upon success, returns to the Logged state. The admin can also check schedules during the process. The workflow ends when the admin logs out and returns to the Idle state.



Data Flow Diagram

It shows the flow of data between major actors and databases

The Data Flow Diagram (DFD) illustrates how data moves between major actors, processes, and databases within the BUTrace system.

Figure: Level 0 for BUTrace
The Level 0 DFD (Context Diagram) presents an overall view of the system. It shows BUTrace as a single process interacting with external actors such as Admin, Student, and Driver. These actors send requests (login, route check, trip updates, complaints) and receive responses (notifications, schedules, reports). It highlights the system boundary and primary data exchanges.

Figure: Level 1.1 (Admin) for BUTrace
This level expands the Admin process. It shows how the Admin manages users, routes, buses, and schedules. Data flows between the Admin interface and system databases for storing and updating records. It also includes report generation and verification of actions.

Figure: Level 1.2 (Student) for BUTrace
This diagram details the Student's interaction with the system. Students log in, view routes and schedules, track live bus locations, and submit complaints. Data flows between the student interface and databases such as Trip, Schedule, and Complaint records.

Figure: Level 1.3 (Driver) for BUTrace
This level focuses on Driver operations. Drivers log in, start or update trips, report incidents, and update bus status. Data flows between the driver interface and Trip, Bus, and Incident databases.

Figure: Level 2.1 (Admin) for BUTrace
This diagram further decomposes Admin processes into detailed sub-processes such as add/update/delete user, manage fleet, verify routes, and generate reports. It shows precise data interactions with respective databases.

Figure: Level 2.2 (Student) for BUTrace
This level provides detailed student operations including live tracking requests, schedule checks, complaint submission, and notification handling. It illustrates how each action interacts with relevant data stores.

Figure: Level 2.3 (Driver) for BUTrace
This diagram breaks down driver activities into detailed steps such as starting a trip, updating trip status, reporting incidents, and ending trips. It clearly shows how operational data is stored and retrieved from the system databases.

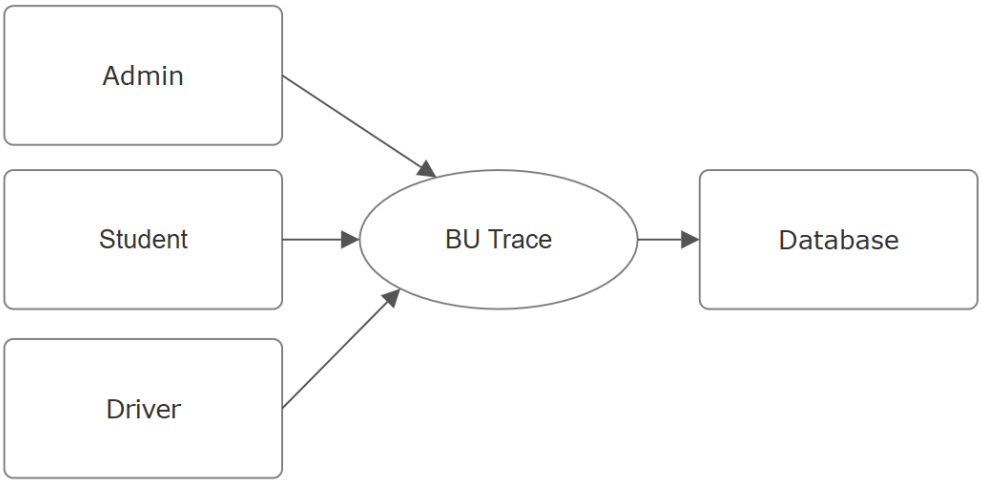


Figure: Level 0 for BUTrace

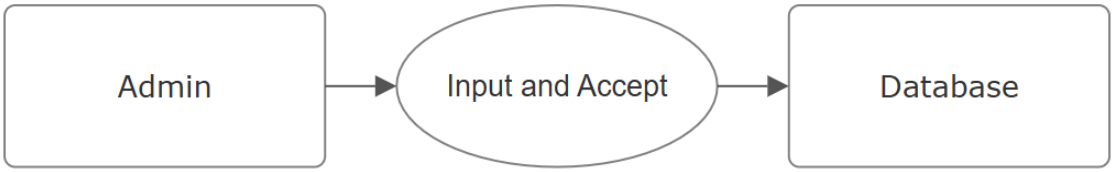


Figure: Level 1.1 (Admin) for BUTrace

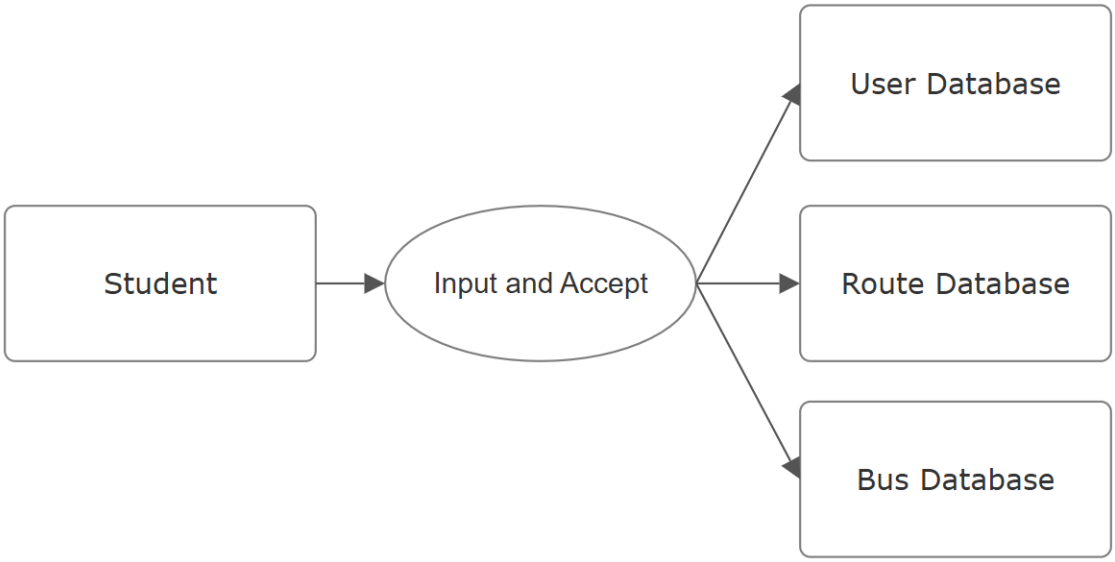


Figure: Level 1.2 (Student) for BUTrace

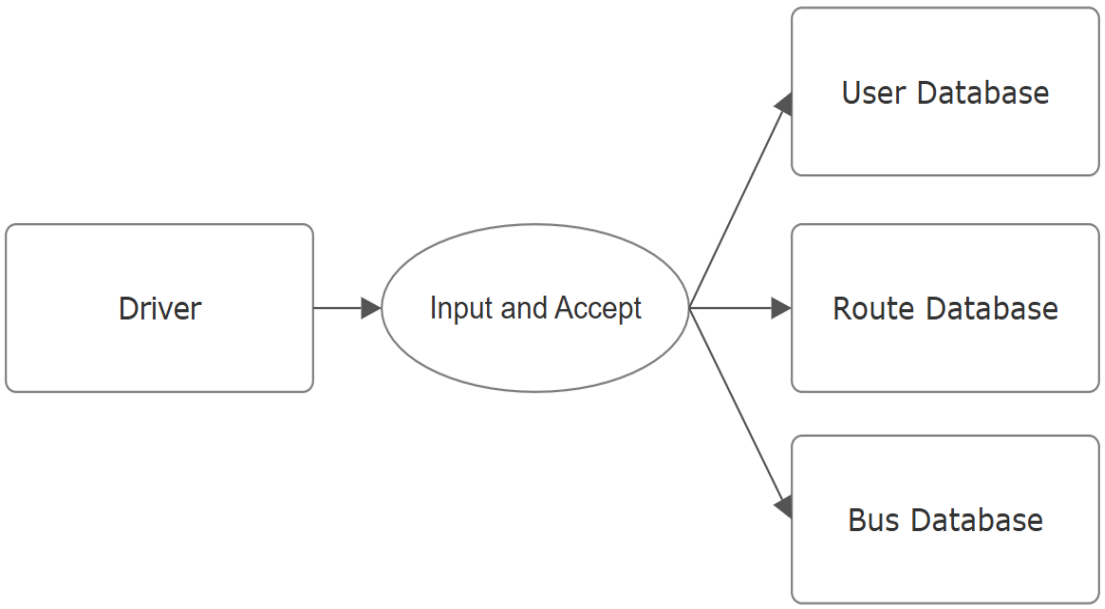


Figure: Level 1.3 (Driver) for BUTrace

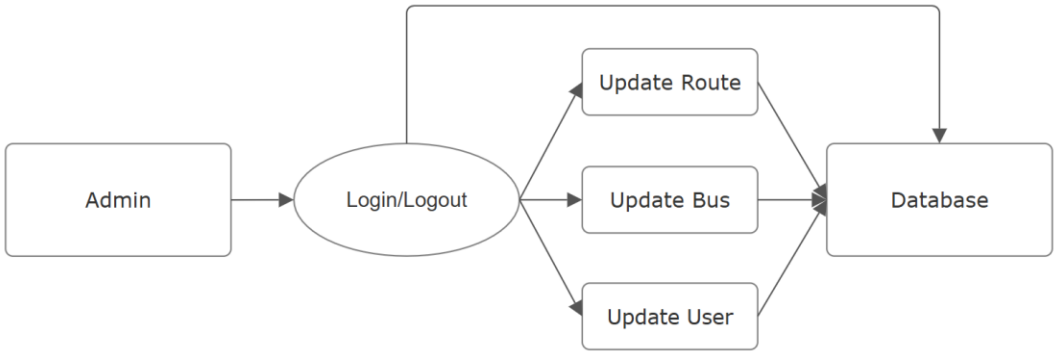


Figure: Level 2.1 (Admin) for BUTrace

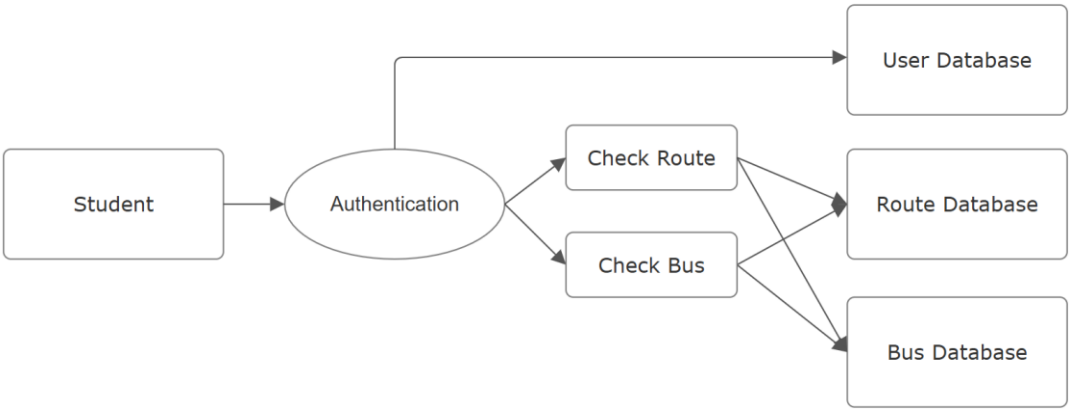


Figure: Level 2.2 (Student) for BU Trace

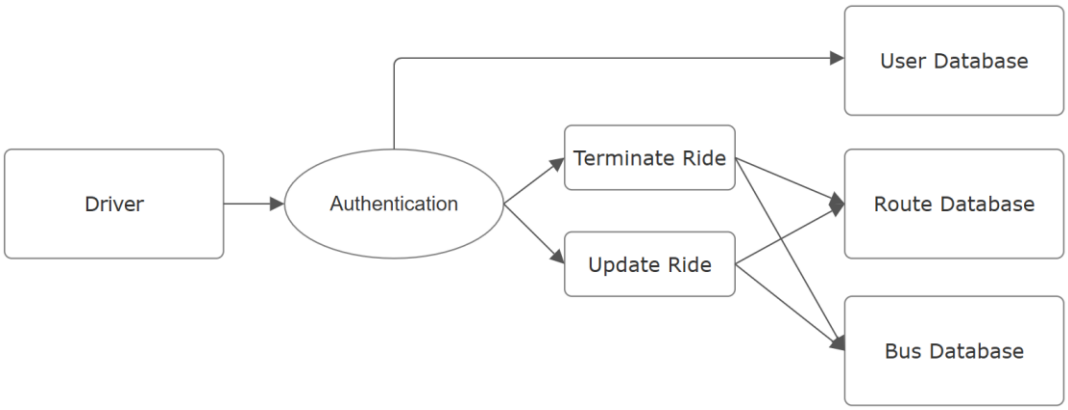


Figure: Level 2.3 (Driver) for BUTrace

Sequence Diagram

It shows how processes or objects operate with one another and in what **chronological order**. This sequence diagram shows the interaction between Student, Faculty, Driver, Admin, and system components such as Server, GPS, and Notification. Users log in, and the Server validates credentials before granting access. Drivers start trips and share live location through GPS, allowing students and faculty to view real-time bus location and ETA. Admin can update routes and schedules, which are broadcast to users. If a delay occurs, the driver updates the status, and the system sends delay notifications to users.

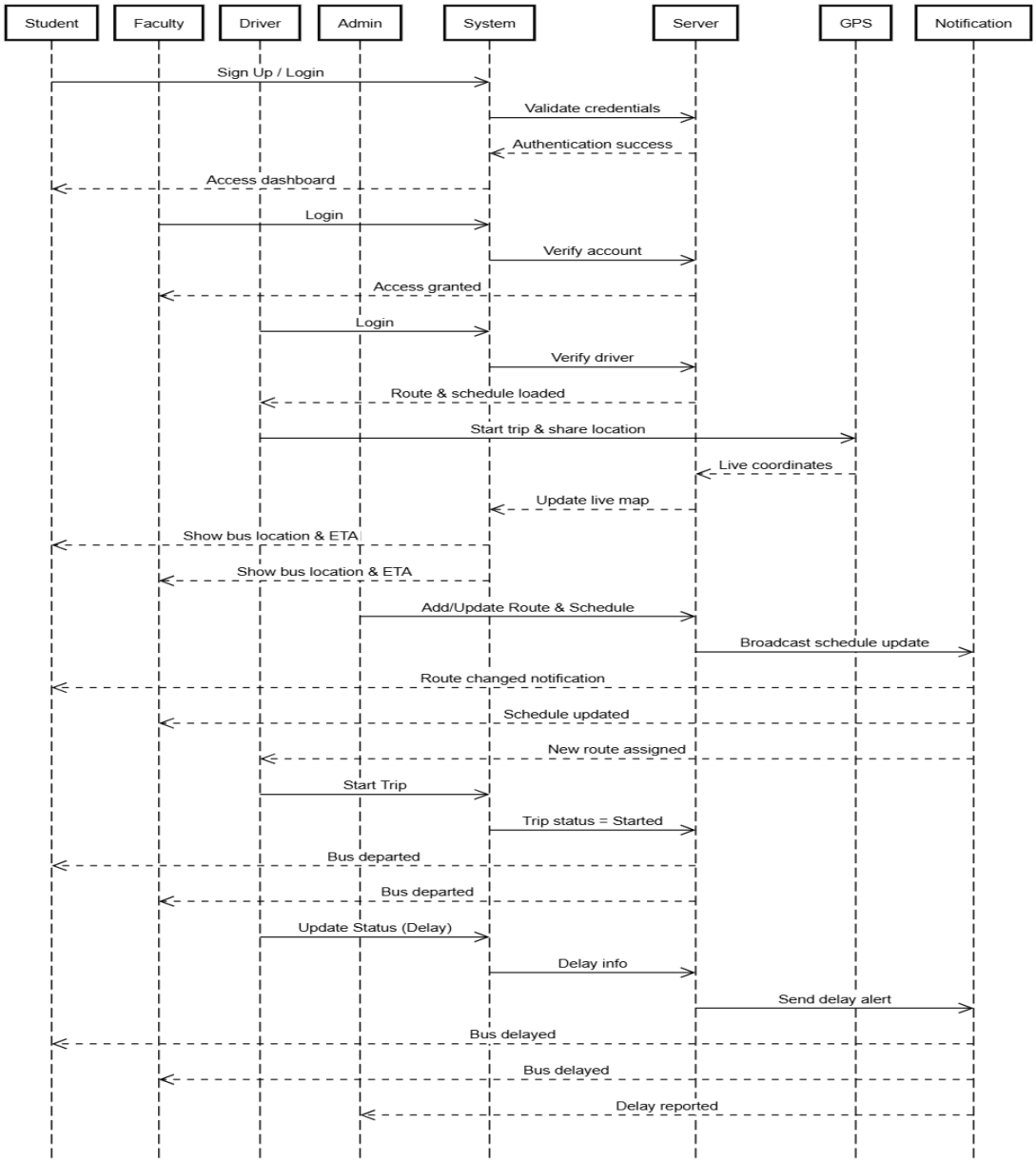


Figure 1(A): BUTrace

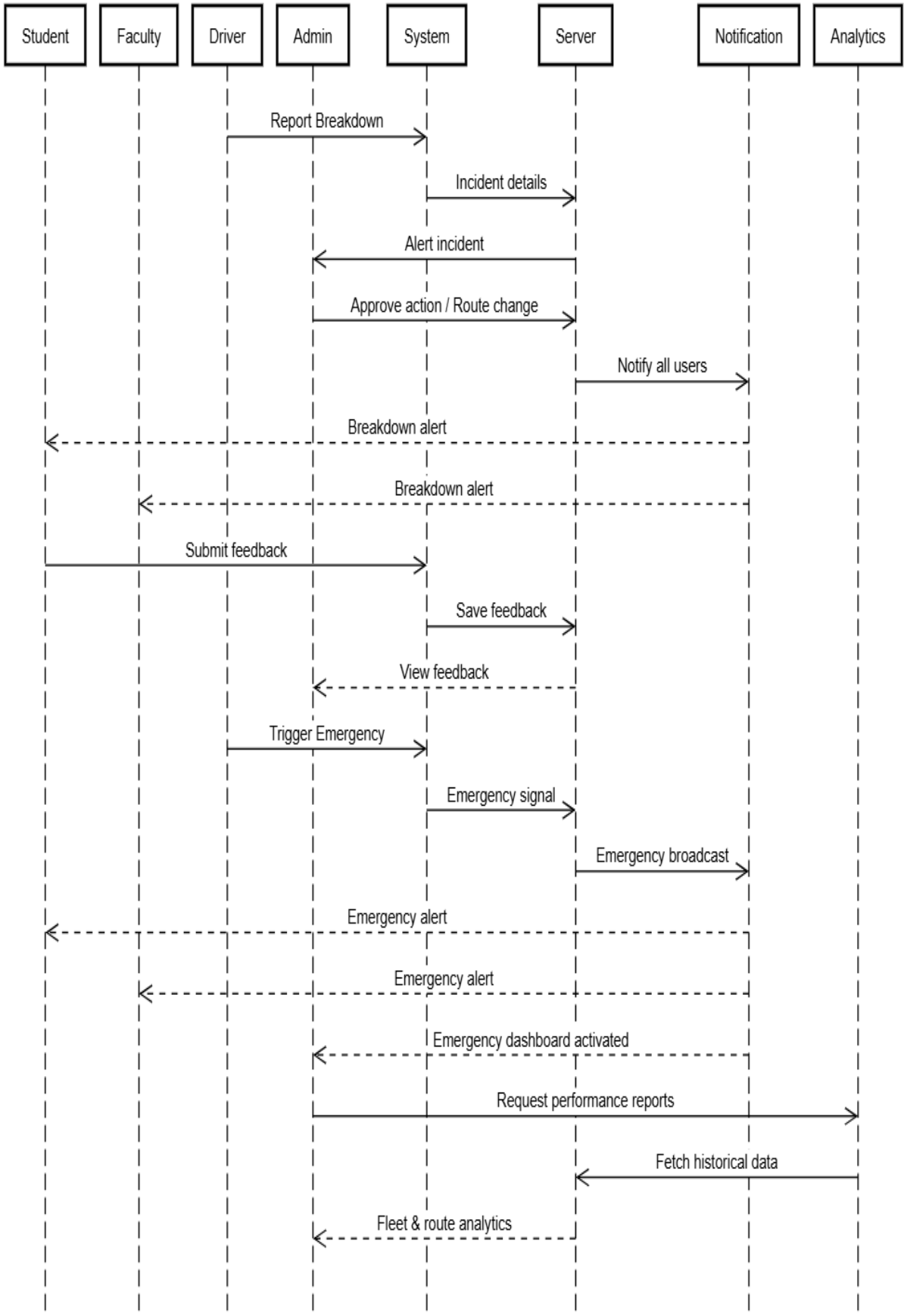


Figure 1(B): BUTrace

Sequence Diagram by Parts of BUTrace

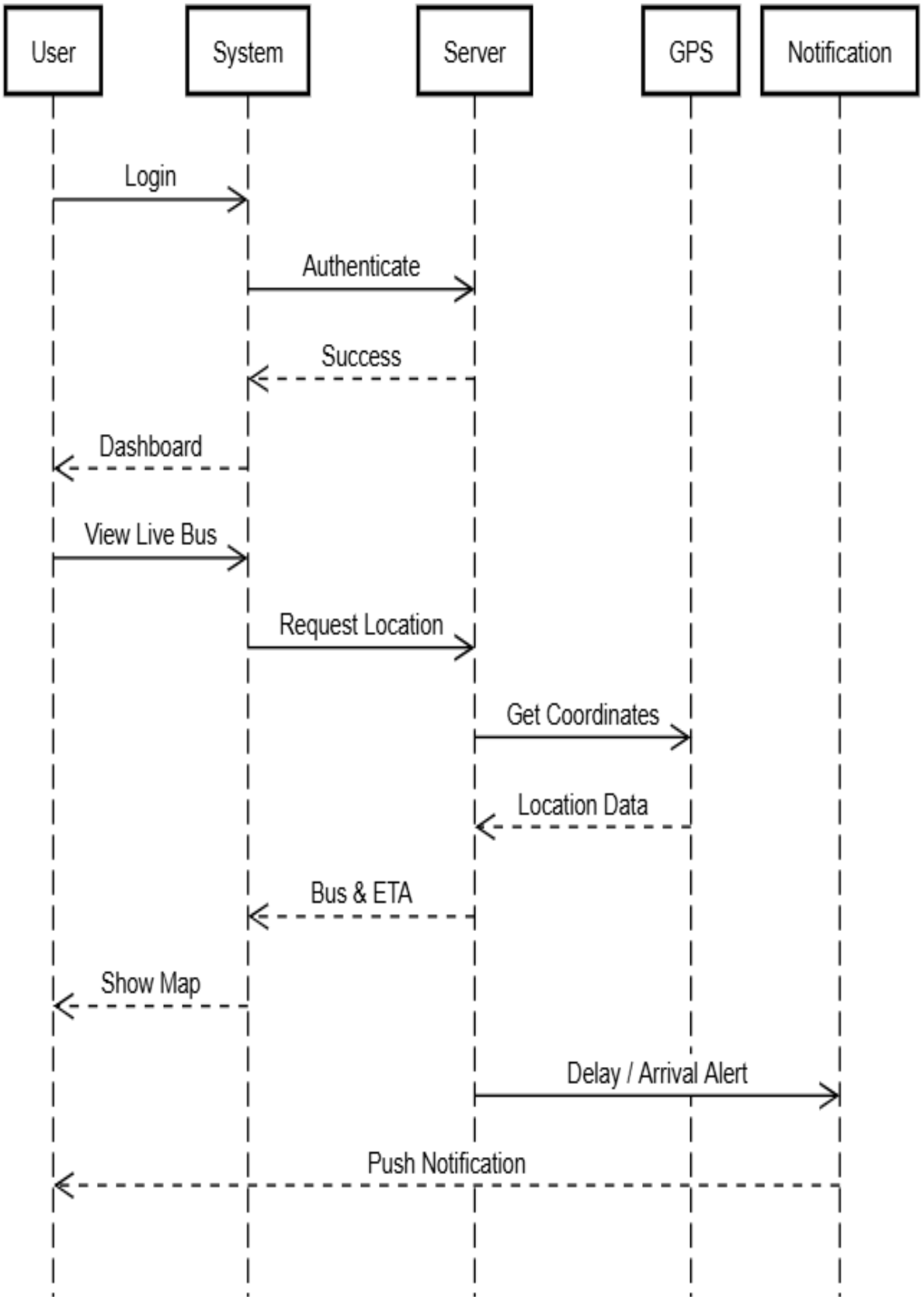


Figure 2: Users Bus Tracking & Notification

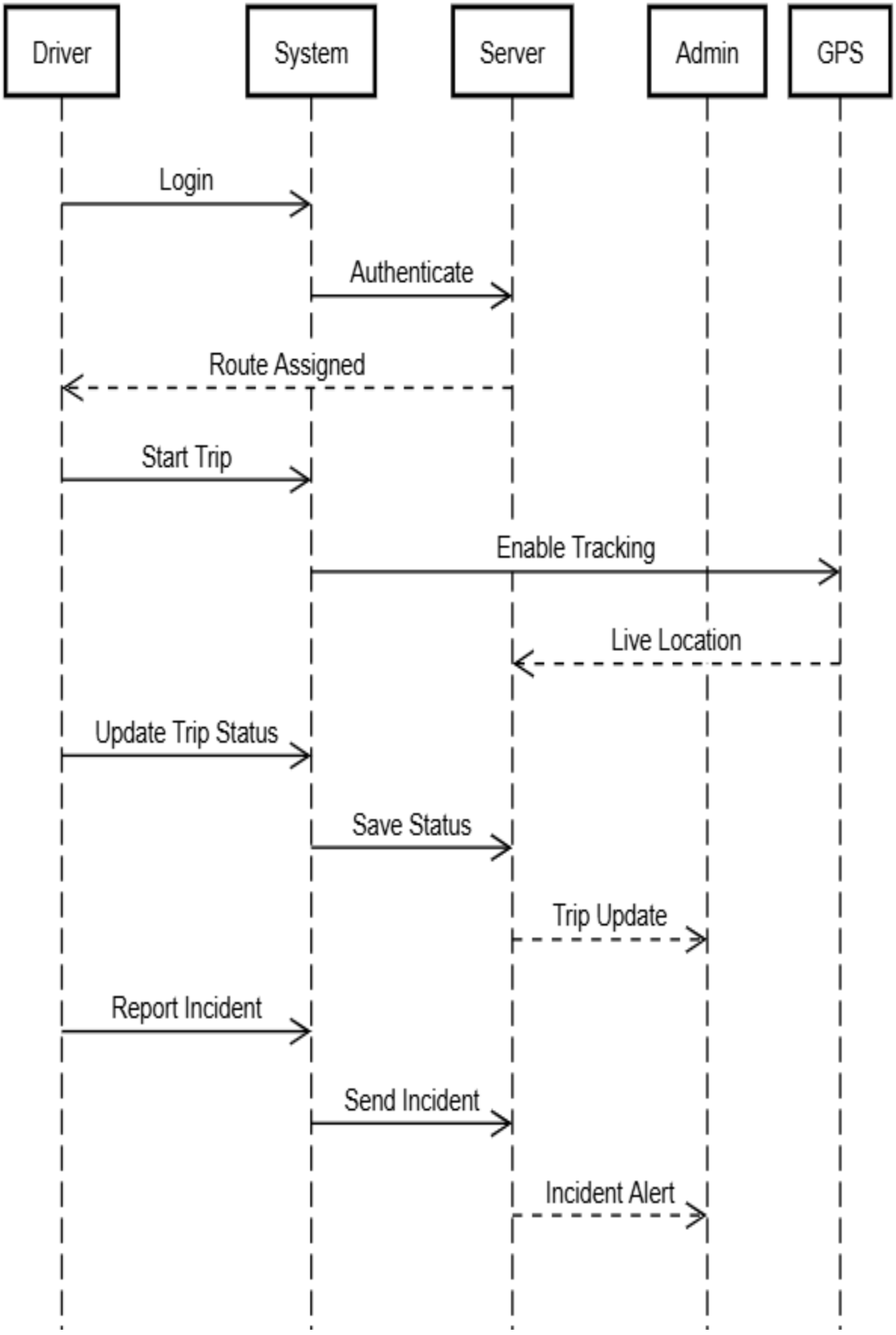


Figure 3: Driver-Trip & Incident Reporting

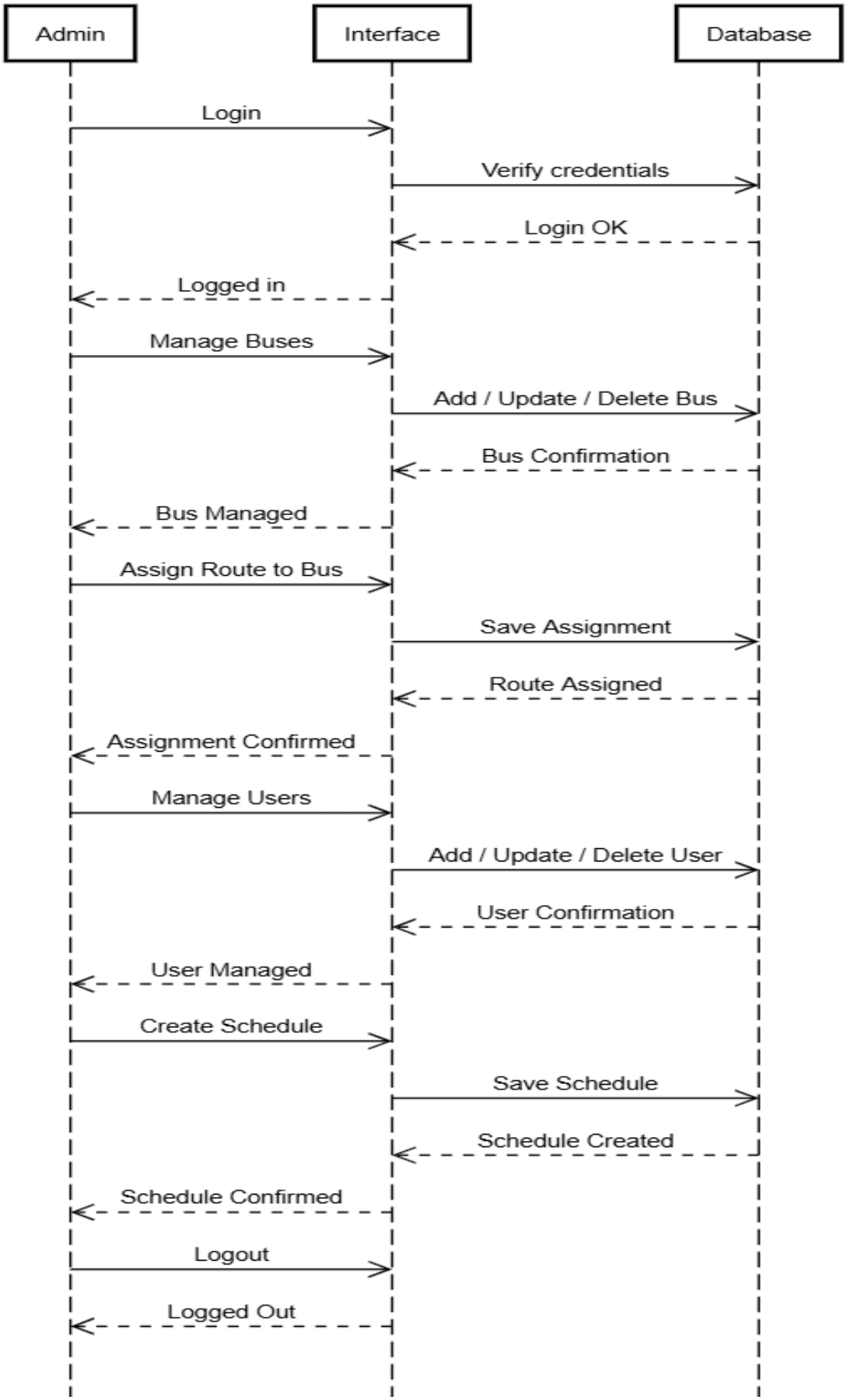


Figure 4: Admin-Fleet & User Management

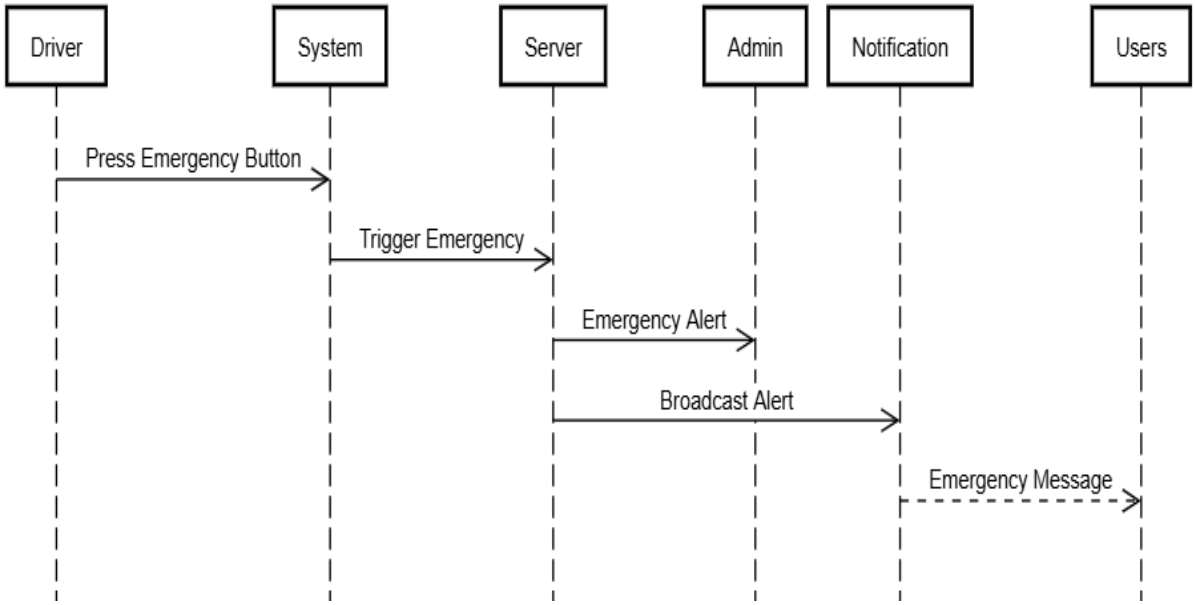


Figure: Emergency Management

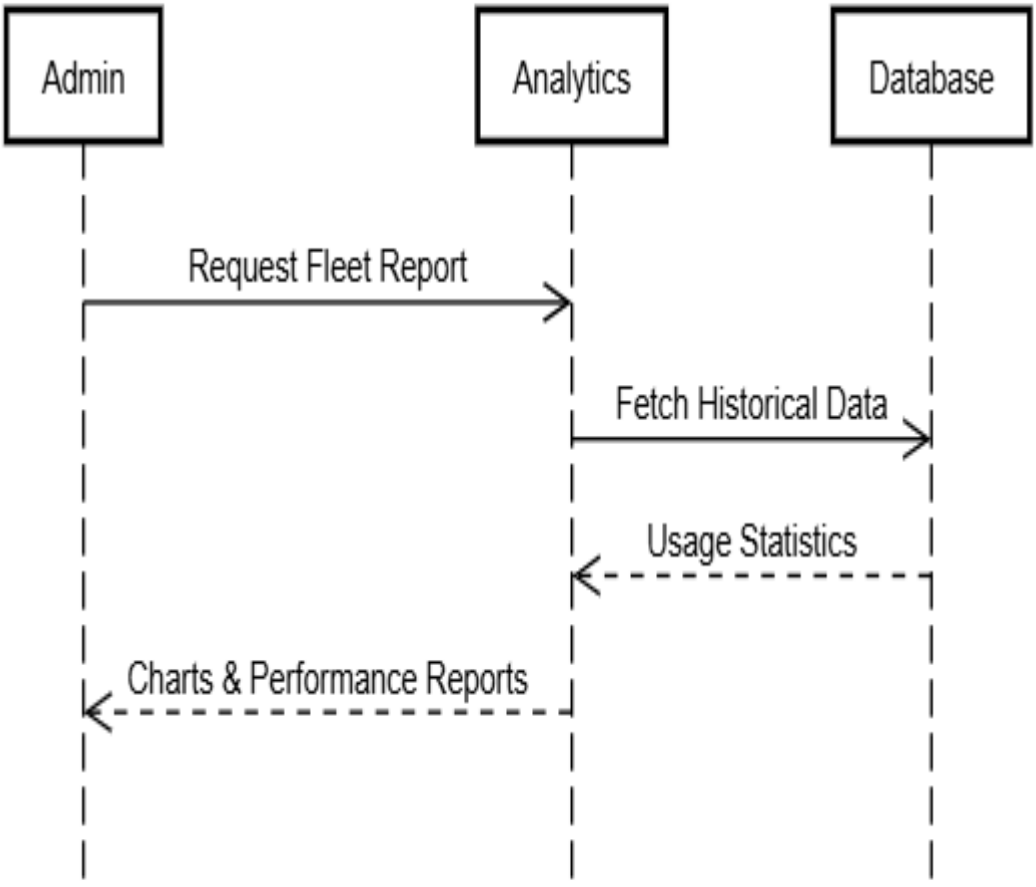


Figure 4: Analytics & Reports