

College Bus Tracking System

Final Project Report

Submitted by:

[Student Name]

Roll No: [Roll Number]

Supervisor:

[Supervisor Name]

Department:

Department of Computer Science & Engineering

[University/College Name]

Date: December 30, 2025



Abstract

The **College Bus Tracking System** is a comprehensive mobile and web-based solution designed to enhance the safety, efficiency, and management of college transportation. By leveraging Global Positioning System (GPS) technology and modern web frameworks, the system provides real-time location tracking of buses for students, parents, and faculty. It facilitates seamless communication through push notifications for arrival estimates, delays, and emergencies. For administrators and bus coordinators, the system offers robust tools for route management, driver assignment, and incident handling. This report documents the complete analysis, design, and architecture of the system, adhering to software engineering standards.

1. Introduction

1.1 Background

Transportation management in educational institutions often suffers from a lack of real-time visibility, leading to long wait times and safety concerns for students. Traditional methods of communicating delays are often inefficient and reactive.

1.2 Problem Statement

Existing systems lack real-time transparency, making it difficult for stakeholders to know the precise location of college buses. There is a need for a unified platform that connects drivers, students, and administrators to ensure timely and safe commutes.

1.3 Objectives

- To develop a cross-platform mobile application for real-time bus tracking.
 - To implement a secure role-based access control system for different stakeholders.
 - To ensure low-latency communication using WebSocket technology.
 - To provide administrative controls for fleet and route management.
-

2. System Overview & Architecture

The system follows a modern client-server architecture, utilizing a **Flutter** frontend for cross-platform mobile access and a **Node.js** backend for scalable API handling.

2.1 High-Level Architecture

The system comprises three main layers: the Presentation Layer (Mobile App), the Application Layer (API & Real-time Server), and the Data Layer (MongoDB).

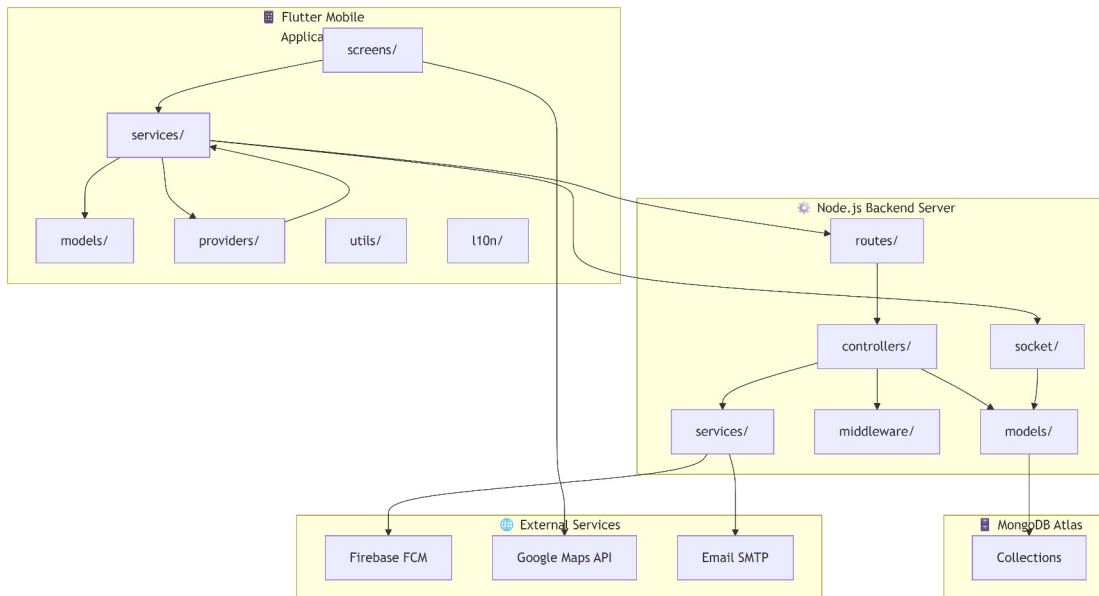


Figure 2.1: High-level package structure showing Mobile, Backend, and External Services interaction.

2.2 API Architecture

Communication between the mobile clients and the server occurs via RESTful APIs for transactional data and Socket.IO for real-time events.

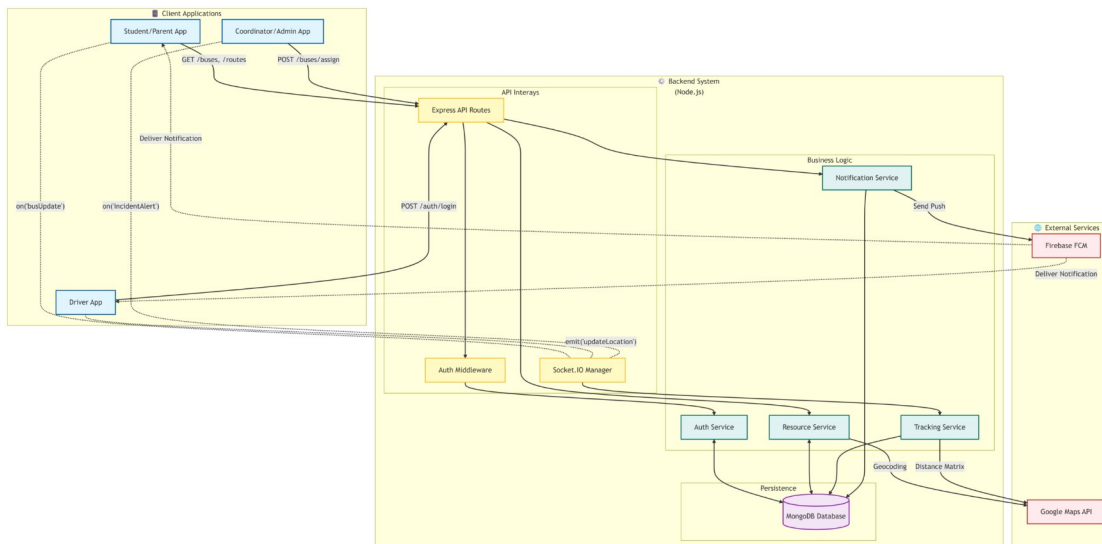


Figure 2.2: Client-Server communication patterns including REST and WebSockets.

2.3 Real-Time Architecture

Latencies are minimized using a persistent socket connection for location streaming.

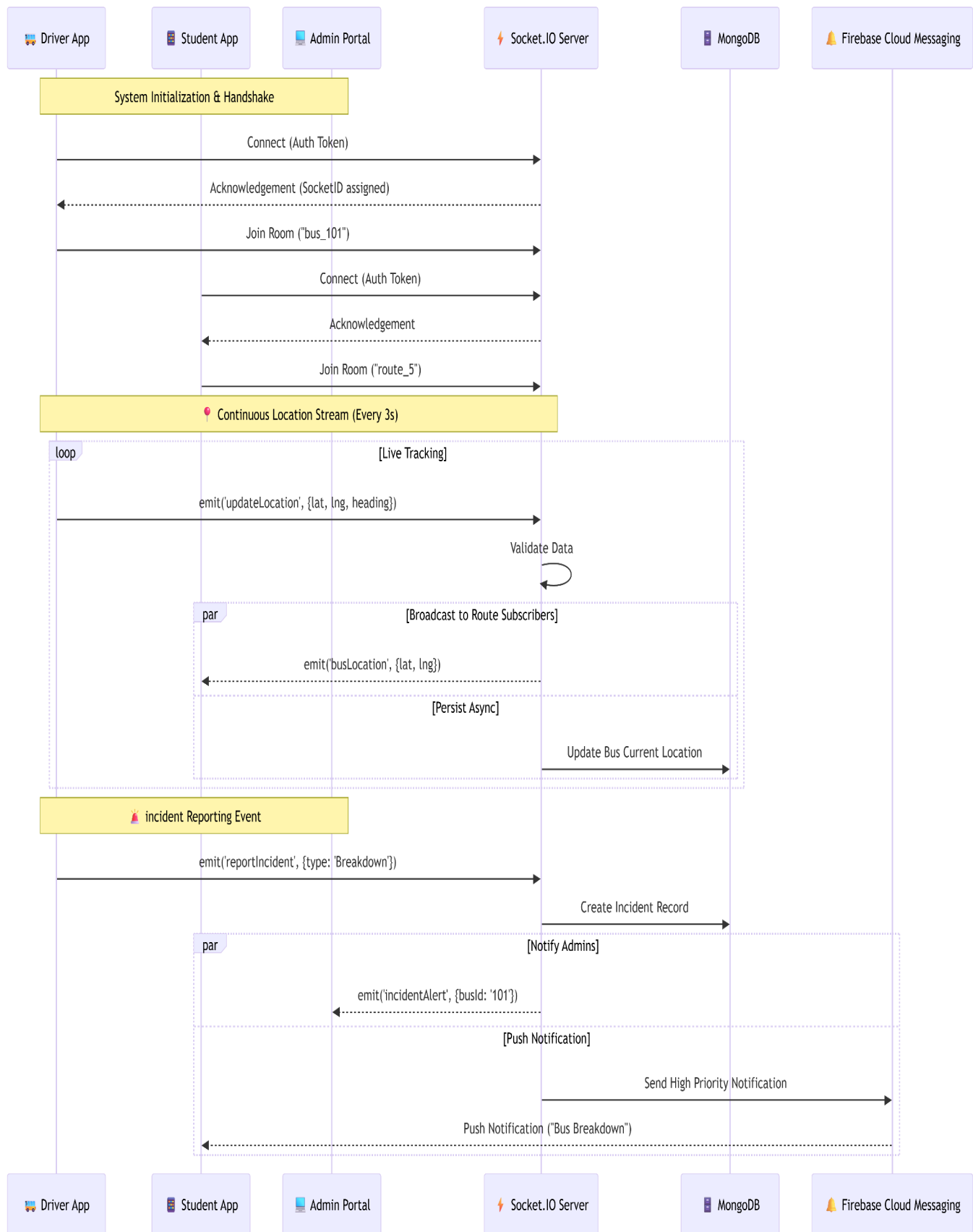


Figure 2.3: Event-driven architecture for live tracking and alerts.

3. Requirement Analysis

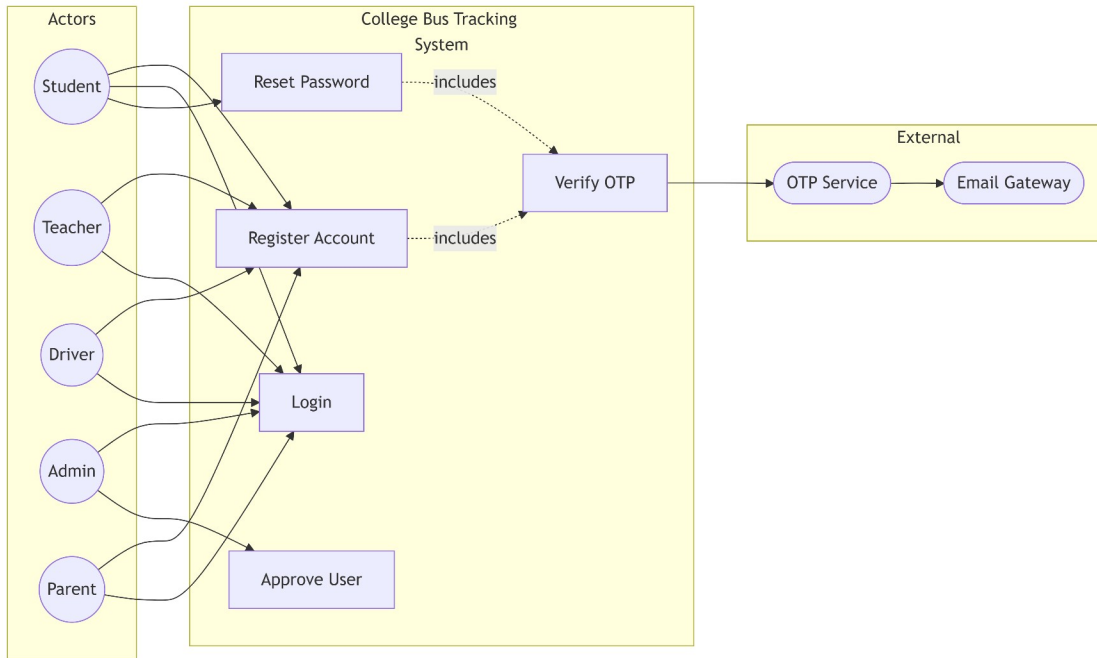
3.1 Functional Requirements (Use Cases)

The system supports multiple actors with distinct capabilities.

3.1.1 Authentication & User Management

Roles: All Users

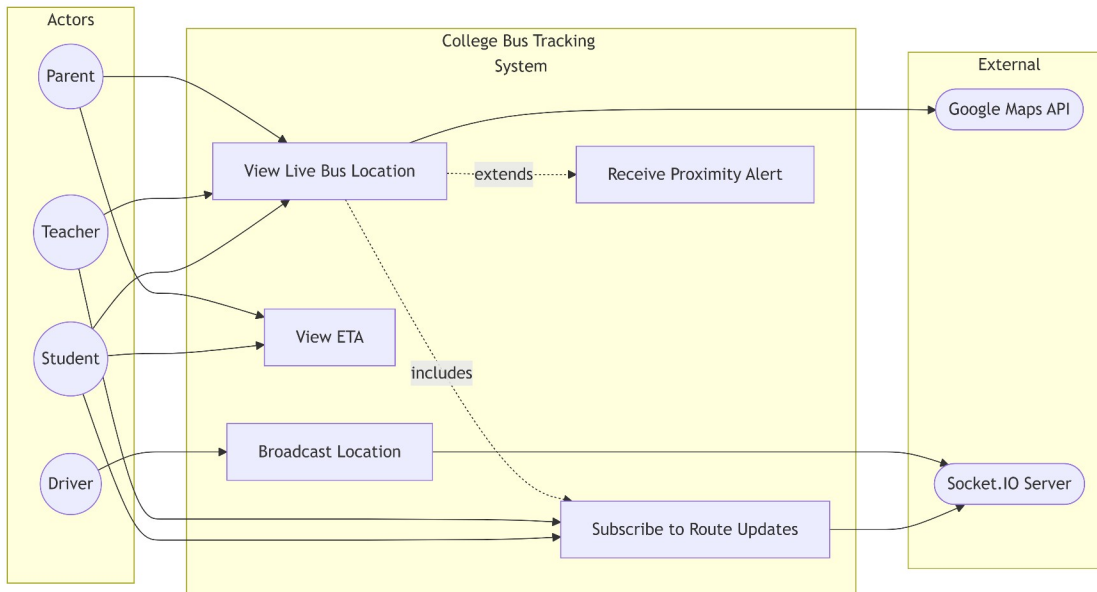
Description: Secure login and registration.



3.1.2 Live Tracking

Roles: Student, Parent, Teacher

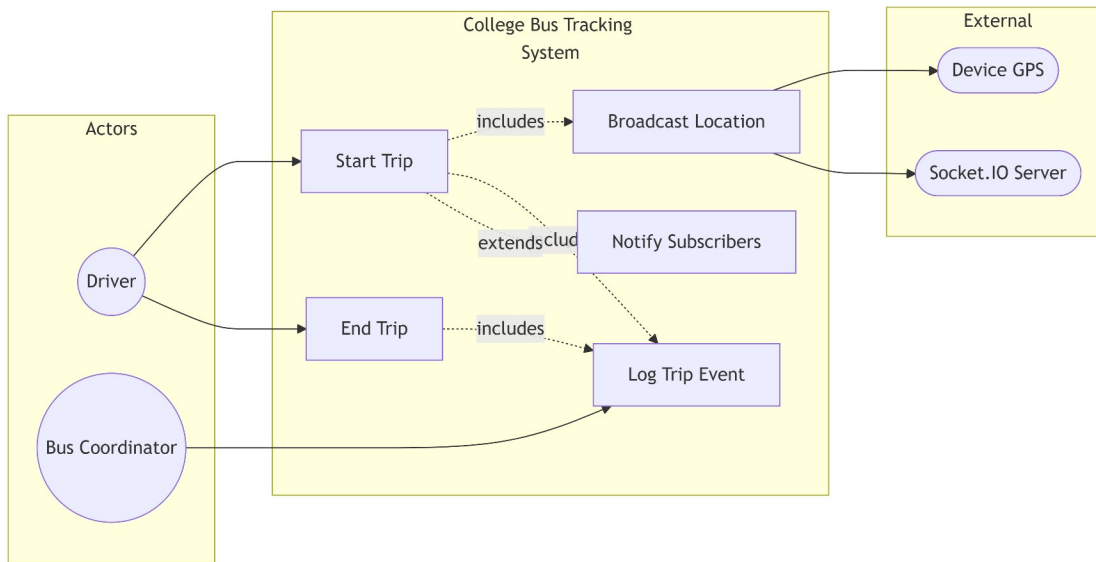
Description: View real-time location of assigned buses.



3.1.3 Trip Management

Roles: Driver

Description: Start and serve daily trips.



(Refer to Appendix A for full Use Case Diagrams UC4-UC9)

3.2 Data Flow Analysis (DFD)

Data movement is modeled from the context level down to specific processes.

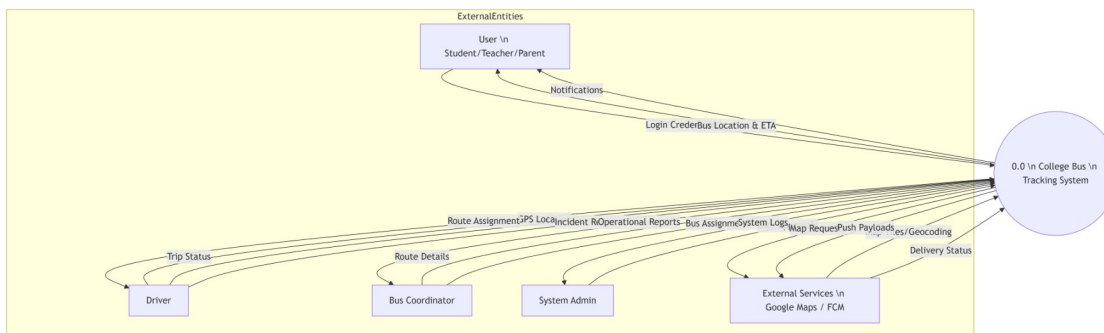


Figure 3.2: Level 0 DFD showing system boundaries and external entities.

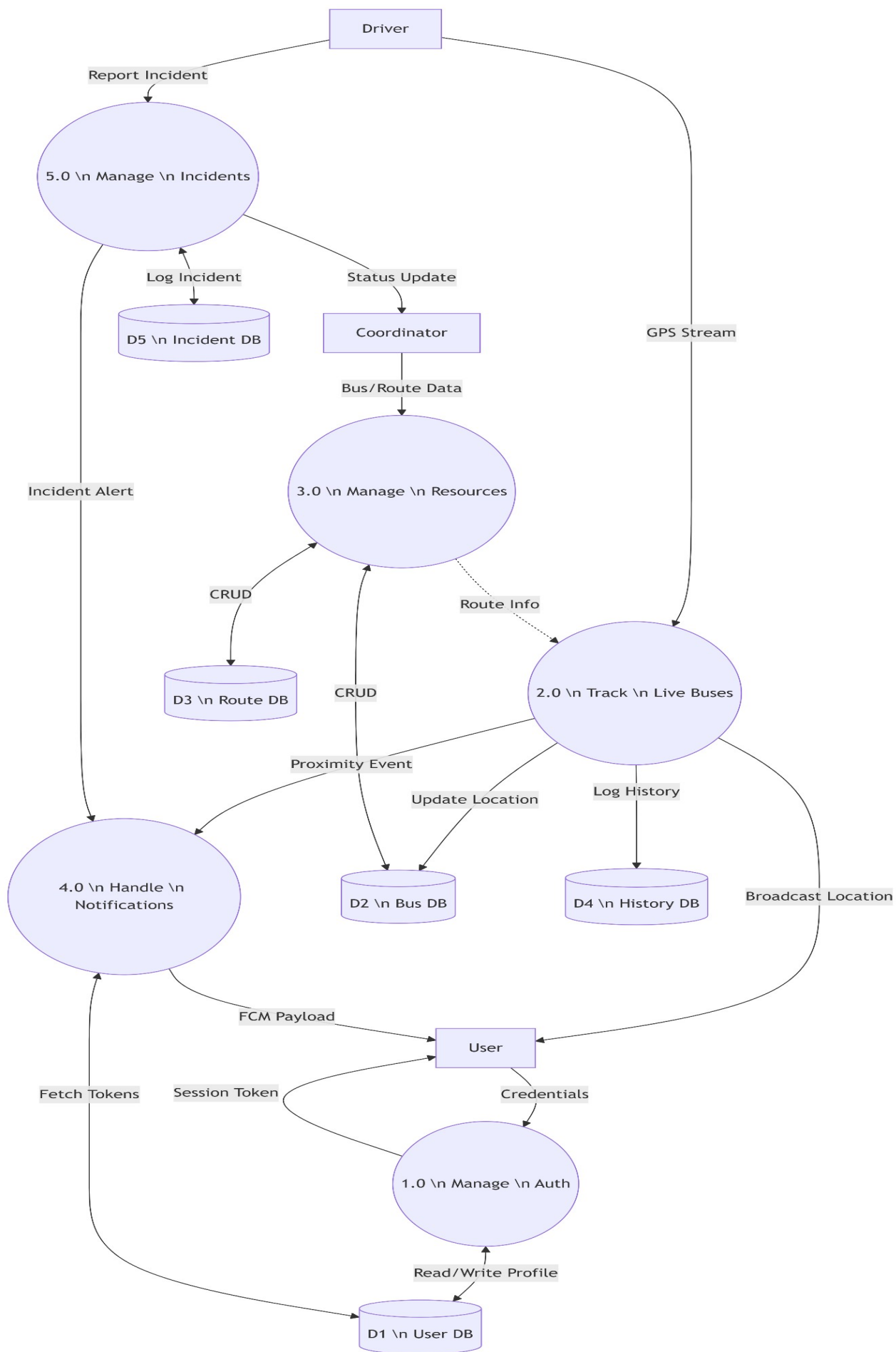


Figure 3.3: Level 1 DFD showing core data transformations.

4. System Design

4.1 Database Design (ER Diagram)

The database uses a document-oriented approach (MongoDB) to store Users, Buses, Routes, and Incidents.

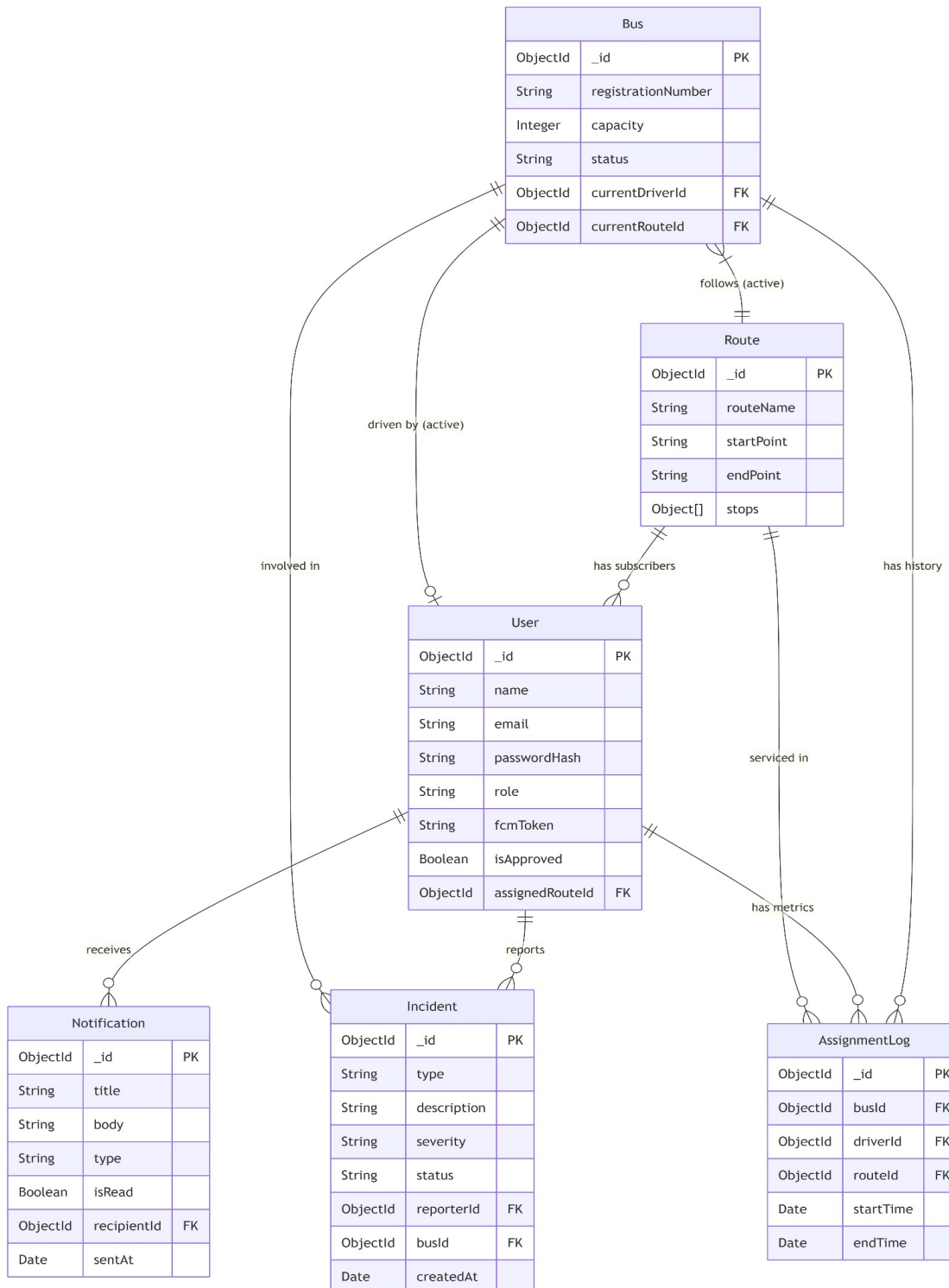


Figure 4.1: ER Diagram showing entities, attributes, and cardinality.

4.2 Class Design

The internal structure of the software is organized into models, services, and controllers.

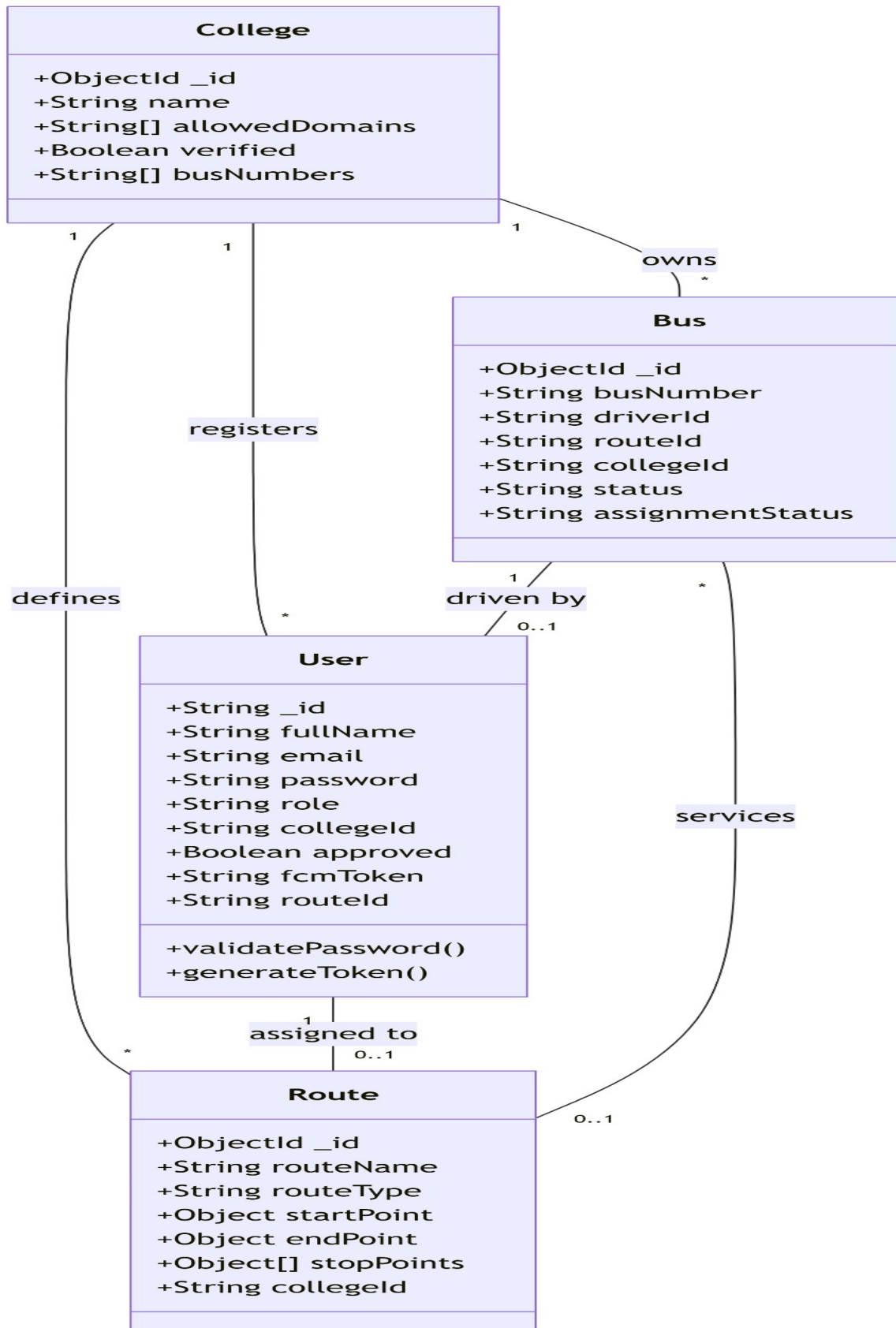


Figure 4.2: Class Diagram for core domain entities.

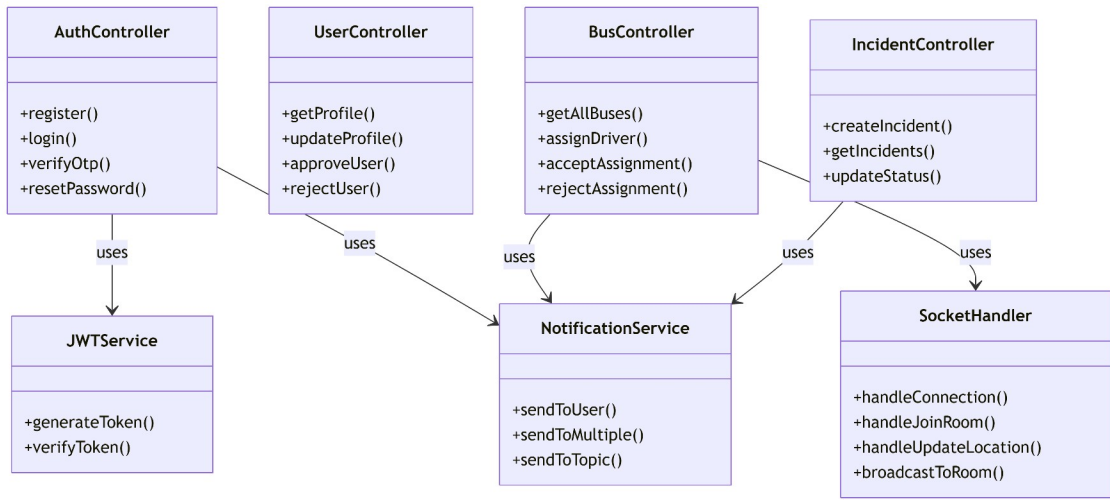
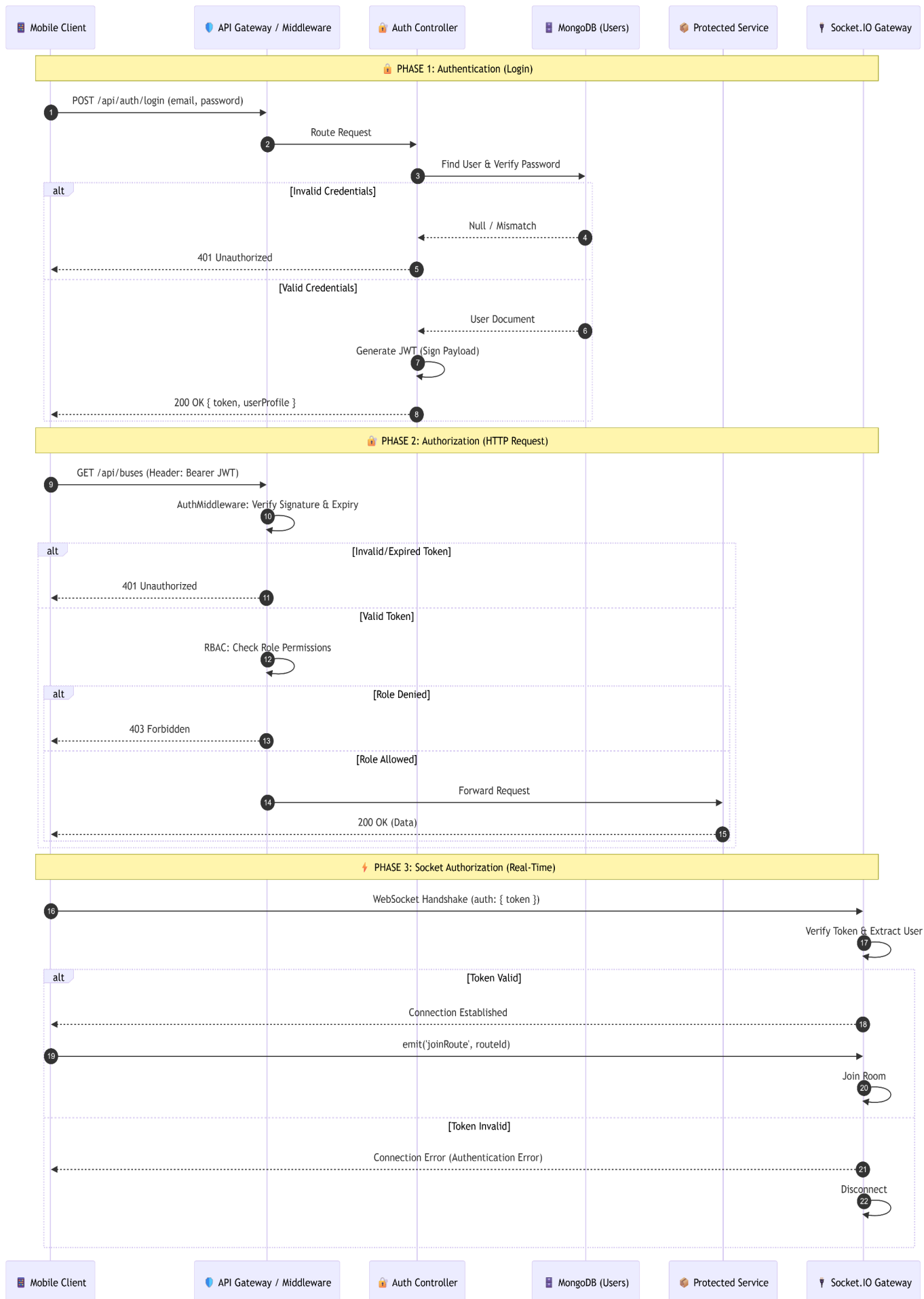


Figure 4.3: Class Diagram for backend controller-service pattern.

4.3 Security Design

Authentication is handled via JWT, with Role-Based Access Control (RBAC).

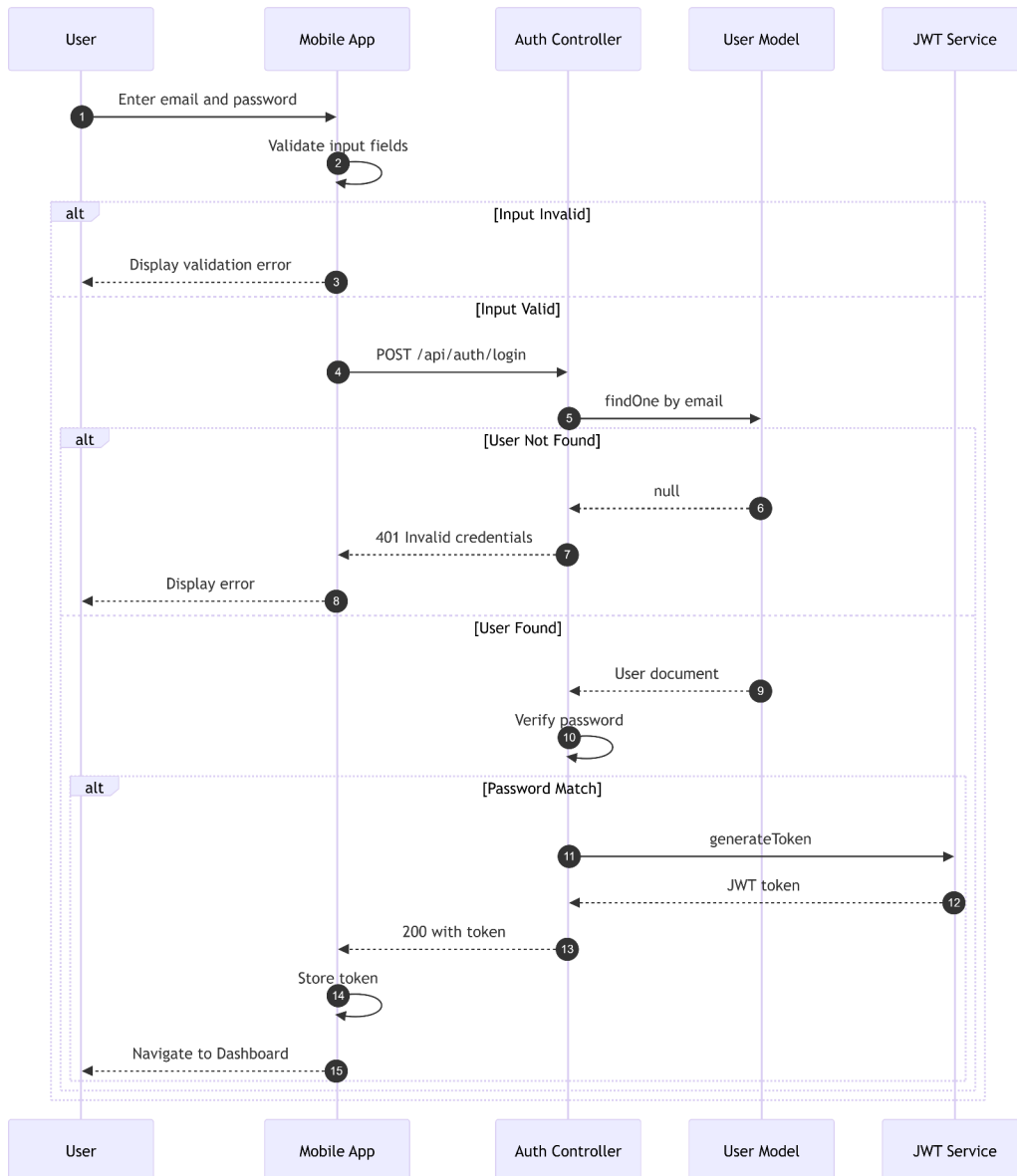


5. behavioral Modeling

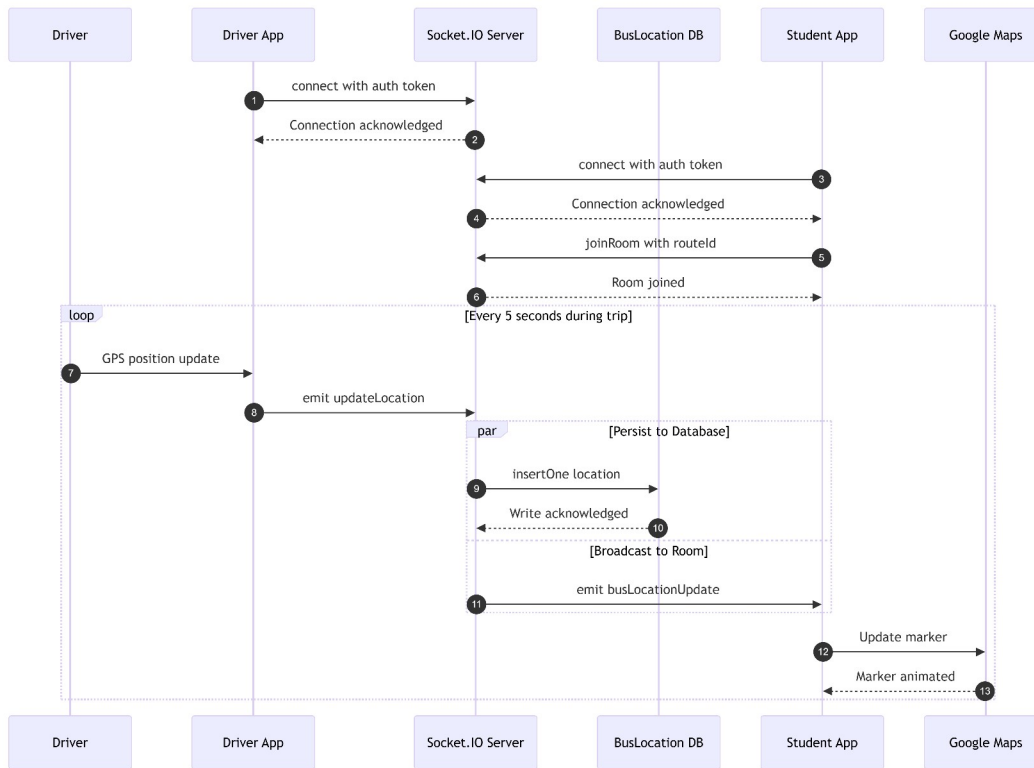
5.1 Sequence Diagrams

Detailed interactions for key workflows.

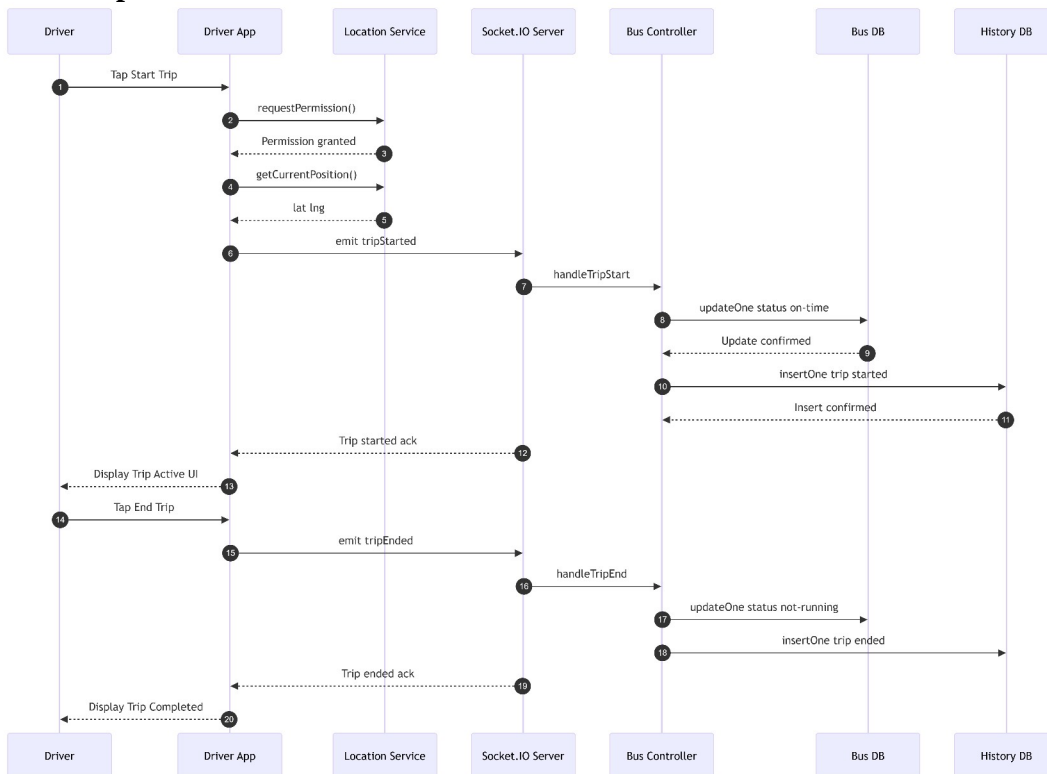
User Authentication Flow:



Real-Time Tracking Flow:



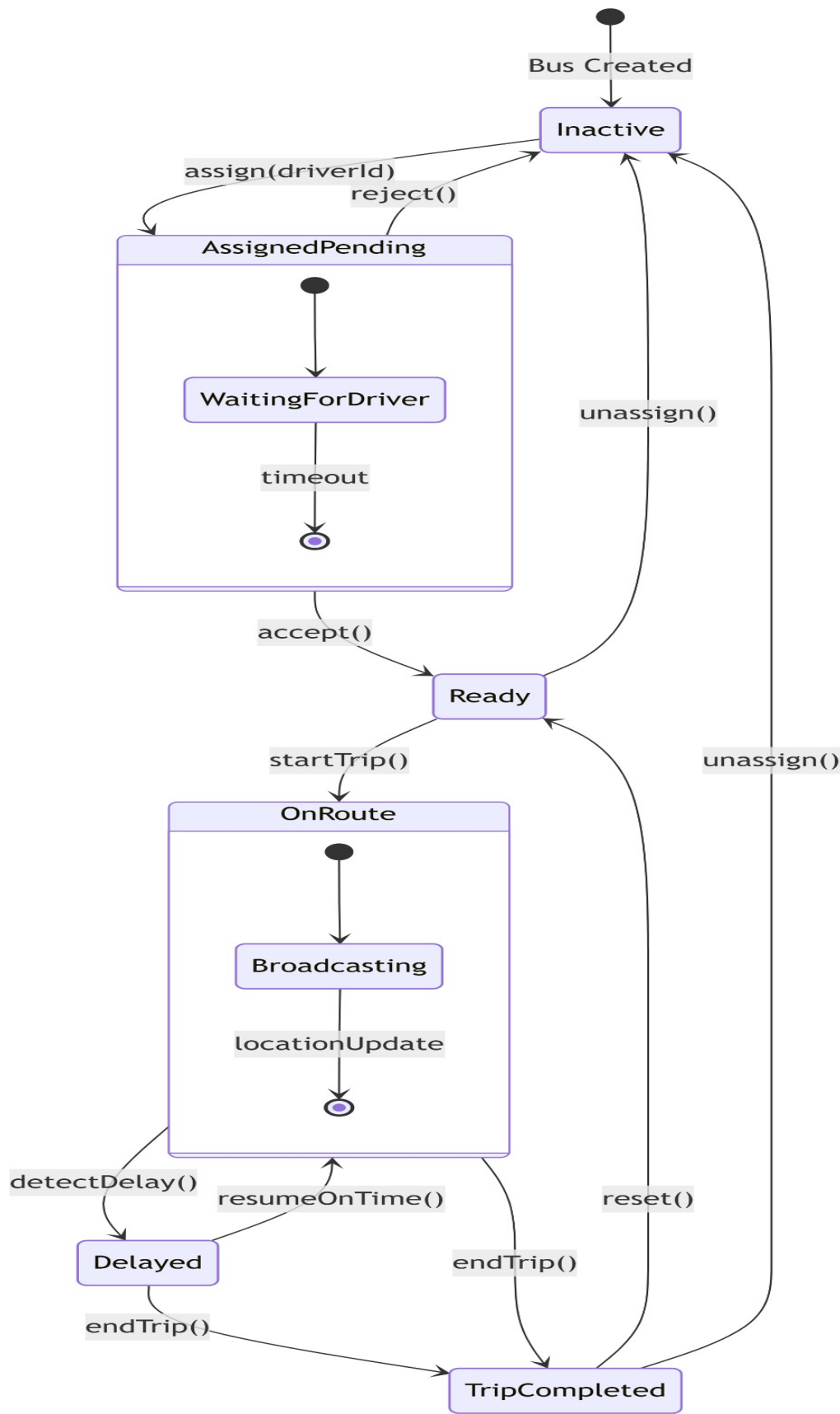
Driver Trip Workflow:



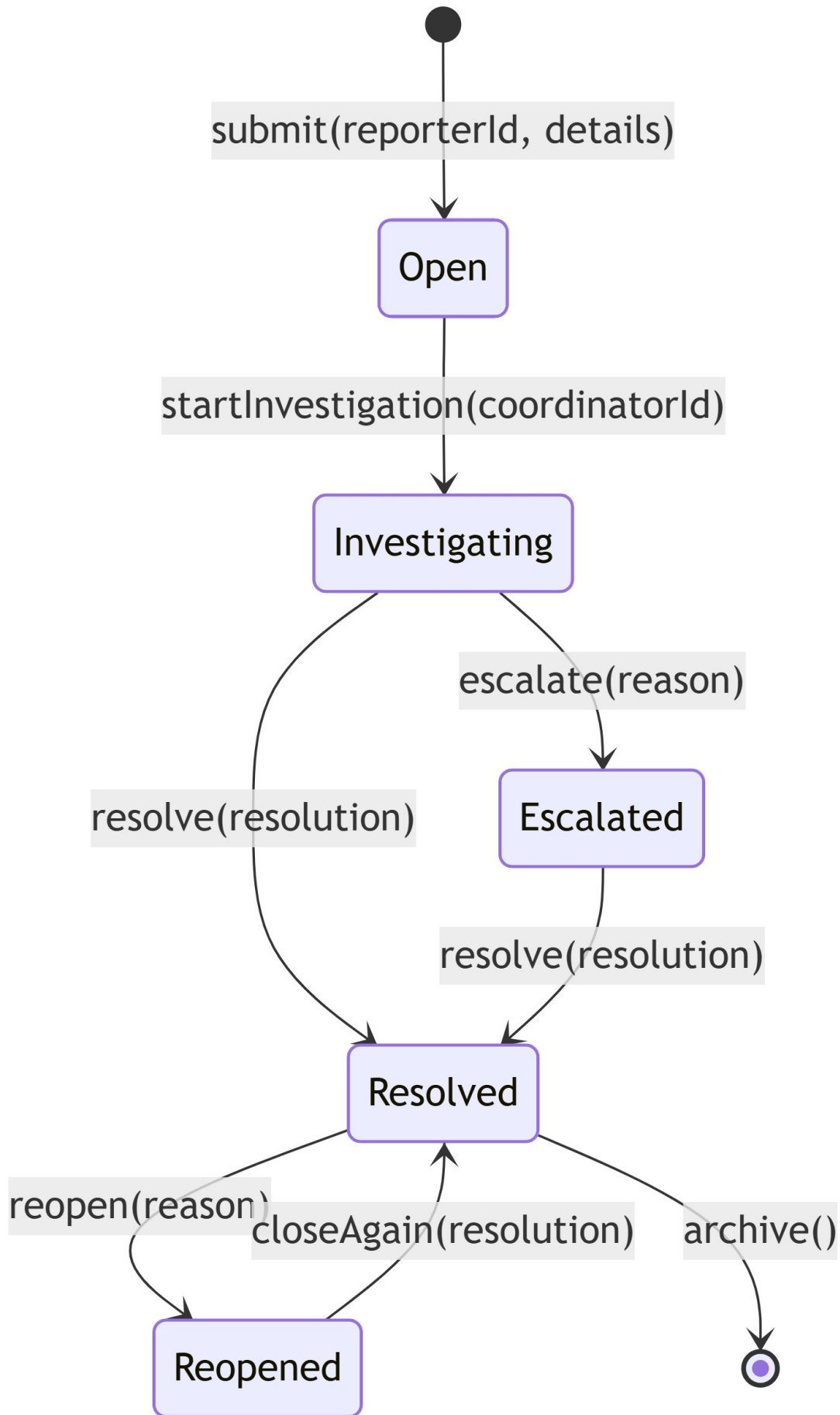
5.2 State Machine Diagrams

Lifecycle modeling for complex entities.

Bus Entity States:



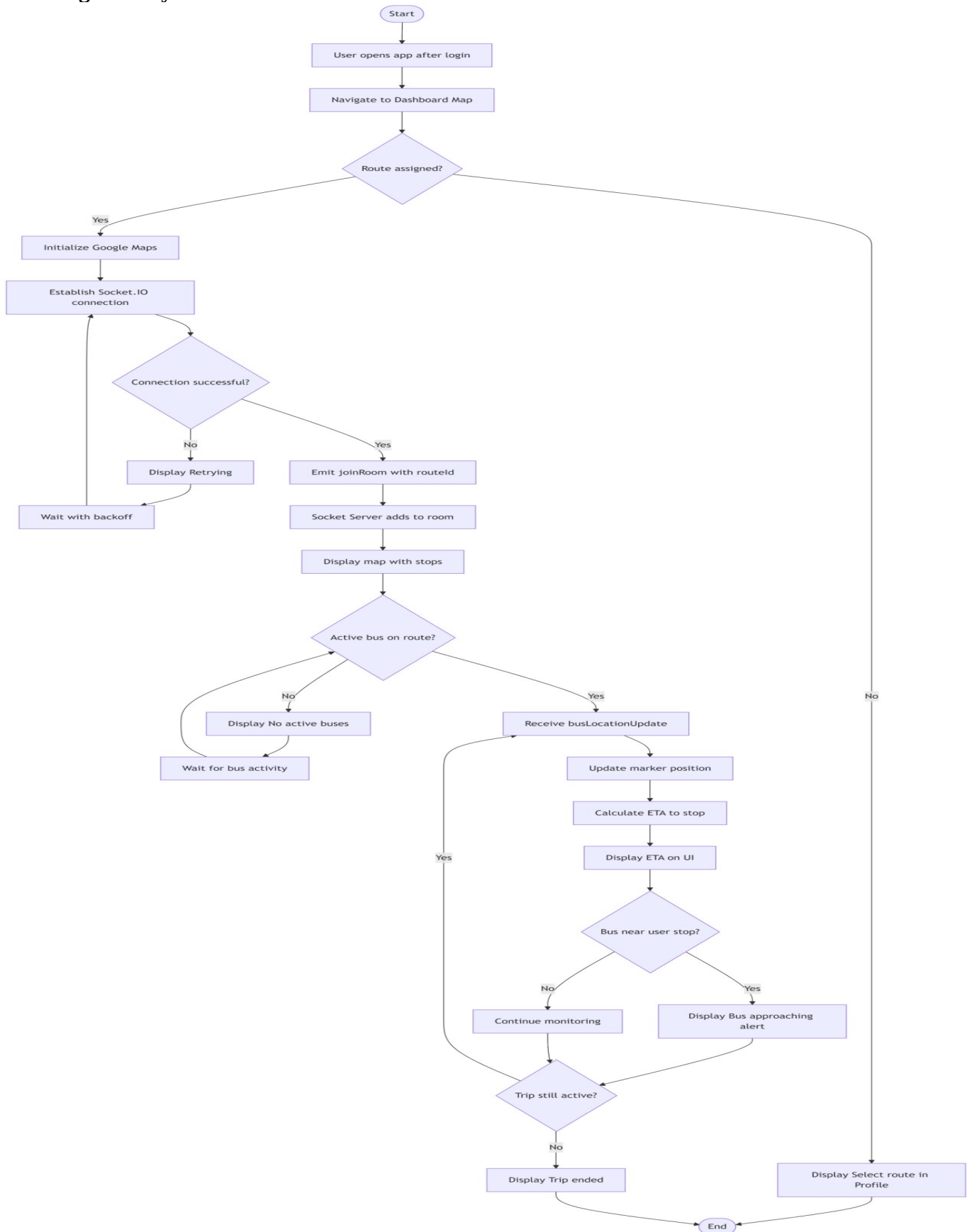
Incident Lifecycle:



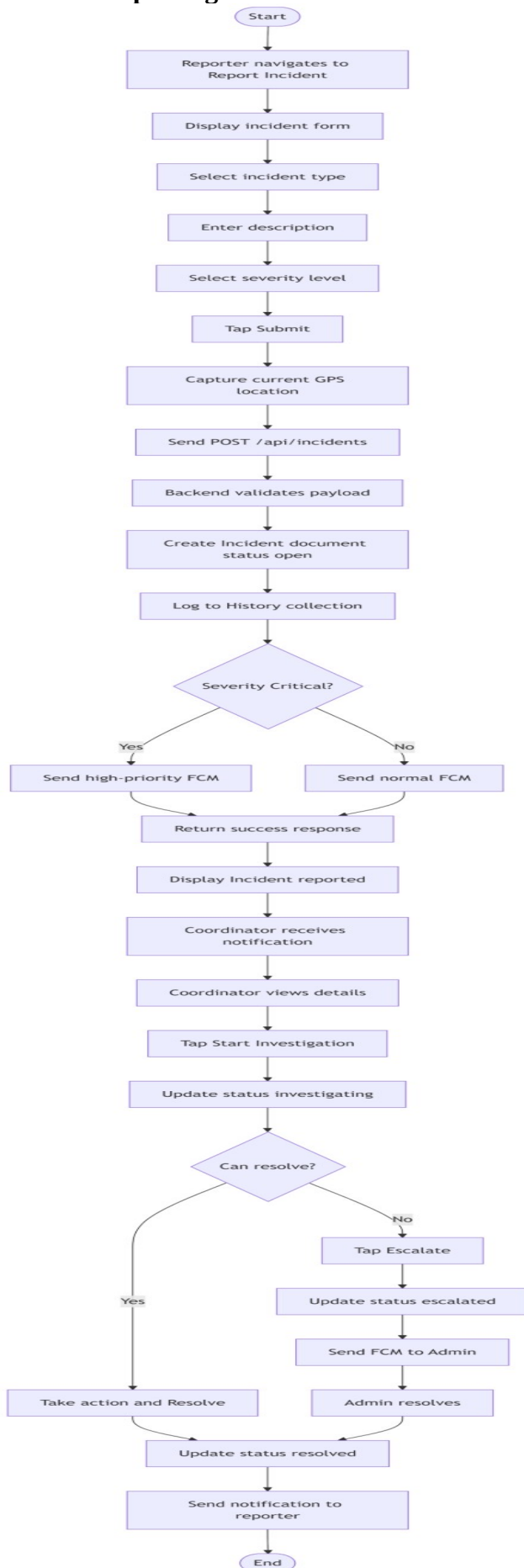
5.3 Activity Diagrams

Workflow logic for business processes.

Tracking Activity:



Incident Reporting:



6. Implementation Environment

6.1 Technology Stack

- **Frontend:** Flutter (Dart) - For Android and iOS delivery.
- **Backend:** Node.js (Express, TypeScript) - For API logic.
- **Database:** MongoDB Atlas - For cloud data storage.
- **Real-time:** Socket.IO - For bidirectional event streams.
- **Maps:** Google Maps Flutter Plugin & Directions API.
- **Notifications:** Firebase Cloud Messaging (FCM).

6.2 Deployment Structure

The system is designed for cloud deployment (e.g., AWS/Heroku).

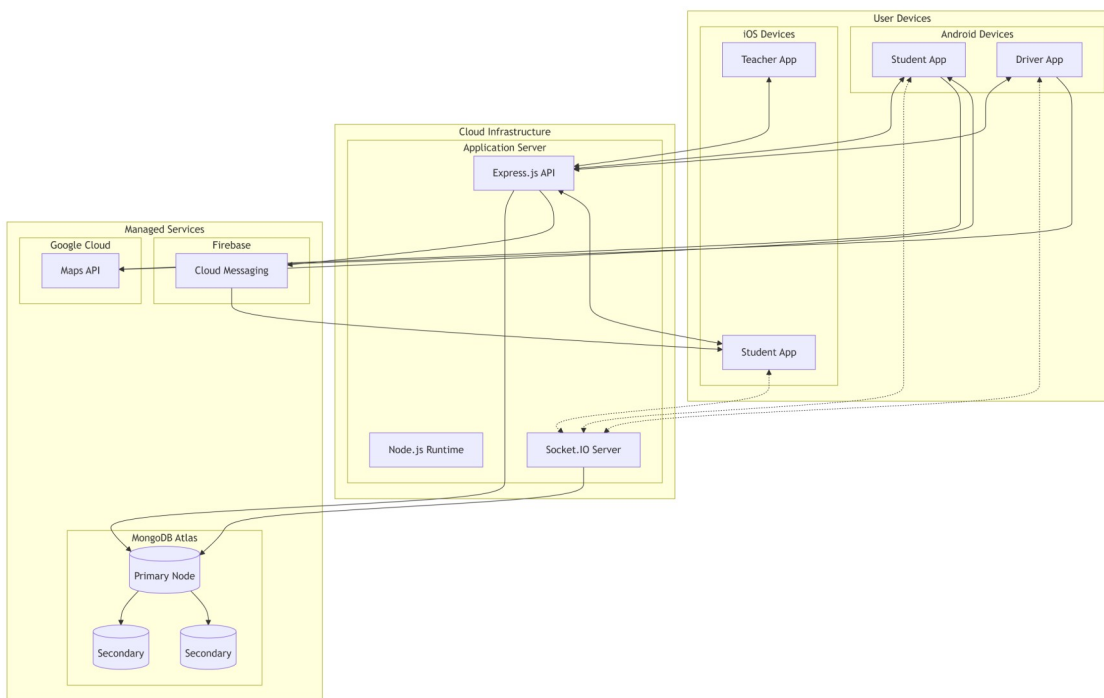


Figure 6.1: Deployment Diagram showing cloud infrastructure.

7. Conclusion & Future Work

The **College Bus Tracking System** successfully addresses the need for real-time visibility in daily campus commutes. By integrating modern mobile and web technologies, it ensures safety, reliability, and ease of management.

Future Enhancements:

- AI-based route optimization to save fuel.
 - Integration with college attendance systems (RFID).
 - Offline mode support for drivers in low-network areas.
-

8. Appendices

Appendix A: Additional Use Case Diagrams

- UC4 Manage Bus Routes
- UC5 Push Notifications
- UC6 Report Incident
- UC7 Assign Driver
- UC8 User Approvals
- UC9 View Trip History

Appendix B: Additional Behavioral Diagrams

- SD4 Push Notification Delivery
- SD5 Incident Reporting
- SD6 Driver Assignment
- SM3 User Account
- SM4 Bus Assignment
- SM5 Driver Session
- SM6 Notification