Software Engineering Project Milestone 3

About: System Design

Department of Computer Science
Namal University, Mianwali

Prepared by:

Mehroz Ali Khan	NUM-BSCS-2024-34
Ali Abbas	NUM-BSCS-2024-06
Laiba Taj	NUM-BSCS-2024-31

18 January 2026

Contents

1 Introduction

This Design Report presents the complete system design for the Electric Bus Location Tracker App, translating the approved Software Requirements Specification (SRS) into concrete design artifacts. The system enables real-time tracking of electric buses in Mianwali District, serving three main user groups: Passengers, Drivers, and Administrators.

1.1 Purpose

The purpose of this document is to:

- Provide comprehensive system architecture and design
- Visualize system behavior through behavioral and structural diagrams
- Establish clear traceability from requirements to design
- Validate design decisions through prototyping and stakeholder feedback

1.2 Scope

This design covers:

- Complete system architecture with layered design
- Behavioral models (Use Case, DFD, Sequence, Activity diagrams)
- Structural models (Class and Component diagrams)
- Interactive prototypes (paper-based and Figma)
- Requirements-to-design traceability

1.3 System Overview

The Electric Bus Location Tracker is a mobile application built using Flutter for cross-platform development, Node.js for backend services, MongoDB for persistent storage, and Redis for real-time data caching. The system integrates with Google Maps API for mapping services and GPS systems for location tracking.

2 Design Assumptions and Constraints

2.1 Design Assumptions

1. **User Capabilities**
   - All drivers possess smartphones with functional GPS
   - Passengers have smartphones with internet connectivity
   - All users have English language literacy
2. **Infrastructure**
   - Continuous internet connectivity available throughout Mianwali
   - GPS satellite coverage is adequate (± 10 meters accuracy)
3. **Stakeholder Support**
   - Full cooperation from Punjab Mass Transit Authority (PMA)
   - Government support for system deployment
   - Driver willingness to share location during duty hours
4. **Technical Environment**
   - AWS cloud services remain available and reliable
   - Google Maps API maintains current service levels
   - Mobile devices run Android 8.0 or higher

2.2 Design Constraints

1. **Regulatory Constraints**
   - Must comply with Pakistan's data protection regulations
   - Requires explicit user consent for location tracking
   - Must adhere to Google Maps API terms of service
2. **Technical Constraints**
   - Limited to Android platform only
   - Google Maps API quota limitations (rate limits apply)
   - Maximum 5-second latency for location updates
   - Database query response time under 500ms
3. **Performance Constraints**
   - Must support minimum 1000 concurrent users
   - Battery consumption limited to 5% per hour
   - Data usage capped at 10MB per hour per user
   - Map interface must load within 3 seconds on 4G

4. **Development Constraints**
    - Three-person development team
    - Budget limitations for cloud services
    - Limited access to physical buses for testing
    - Project completion by January 18, 2026
5. **Security Constraints**
    - All communications must use HTTPS/TLS 1.3
    - JWT token expiry set to 24 hours
    - Password hashing using bcrypt with 10 salt rounds
    - Account lockout after 5 failed login attempts

3 Key Design Decisions

3.1 DFD Decomposition Decisions

1. **Level 0 - Context Level**
   Decision: Single process representing entire system with external entities
   Rationale: Provides high-level view of system boundaries and data flows between users and external systems
2. **Level 1 - Major Processes**
   Decision: Decomposed into 4 main processes:
   - Authenticate Users (FR-001 to FR-009)
   - Track Bus Location (FR-010 to FR-048)
   - Route Management (FR-014 to FR-060)
   - Duty Management (FR-033 to FR-048, FR-067 to FR-071)

   Rationale: Each process represents a cohesive functional area with clear inputs and outputs
3. **Level 2 - Detailed Processes**
   Decision: Further decomposed core processes:
   - Track Bus Location → Validate Location, Update Bus Position, Calculate EAT, Display Real-Time Map
   - Route Management → Bus Data retrieval, Route Information processing

   Rationale: Breaks complex processes into manageable sub-processes while maintaining data flow consistency

3.2 Class Diagram Decisions

1. **User Hierarchy**
   Decision: Created abstract User class with Passenger, Driver, and Administrator as subclasses
   Rationale: Promotes code reuse through inheritance, centralizes common authentication attributes, enables polymorphic user handling
2. **Composition vs Aggregation**
   Decision:
   - Composition: Route-Stop relationship (stops cannot exist without routes)
   - Aggregation: Driver-Bus relationship (drivers can exist independently of buses)

   Rationale: Accurately models real-world dependencies and lifecycle management

3. **Association Classes**
   Decision: Introduced Duty as association class between Driver and Bus
   Rationale: Captures temporal assignment relationship with additional attributes (date, time, completion status)

3.3 Sequence Diagram Distribution

1. **Decision:** Created separate sequence diagrams for each major workflow
   Diagrams Created:
   - User Authentication (FR-001 to FR-008)
   - Bus Tracking and ETA Calculation (FR-010 to FR-028)
   - Driver Duty Management (FR-033 to FR-048)

   Rationale: Prevents overcrowding in single diagram, improves readability, allows focused analysis of specific interactions

3.4 Technology Stack Decisions

1. **Flutter for Mobile Development**
   Rationale: Single codebase for future iOS expansion, rich UI components, strong community support, excellent performance
2. **Node.js + Express for Backend**
   Rationale: Non-blocking I/O ideal for real-time applications, JavaScript consistency across stack, extensive library ecosystem
3. **JWT for Authentication**
   Rationale: Stateless authentication reduces server memory, enables horizontal scaling, industry-standard security

4 All System Design Diagrams

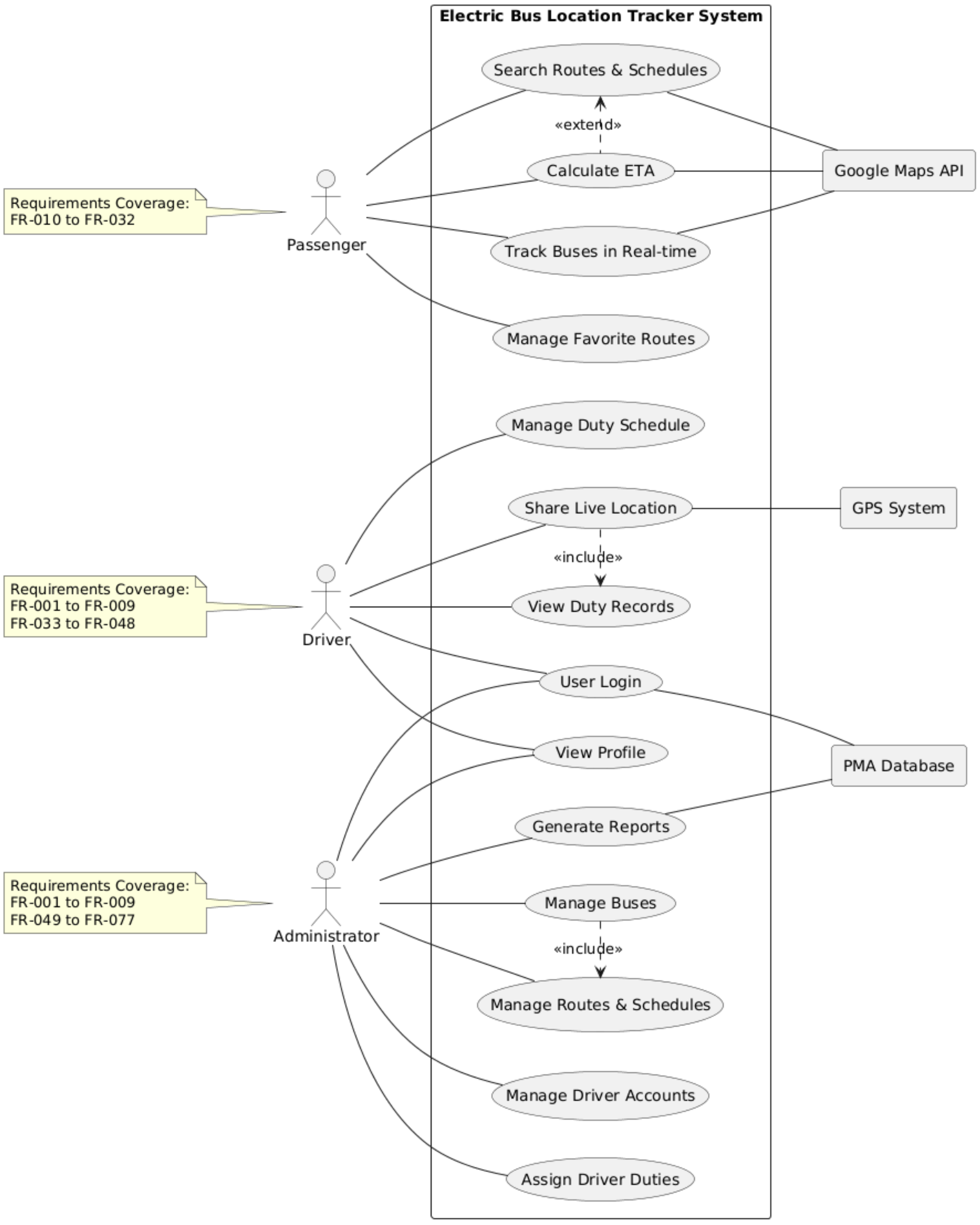
4.1 Use Case Diagram

Purpose: Illustrates all system actors and their interactions with system functionalities.

Actors:

- Passenger - Views bus locations, routes, schedules, and ETA
- Driver - Manages duties and shares live location
- Administrator - Manages system operations, buses, routes, and users
- GPS System - Provides location data
- Google Maps API - Supplies mapping and routing services
- PMA Database - External data source

Coverage: FR-001 to FR-077



Use Case Diagram shows three actors (Passenger, Driver, Administrator) with their respective use cases including authentication, tracking, route management, and duty management.

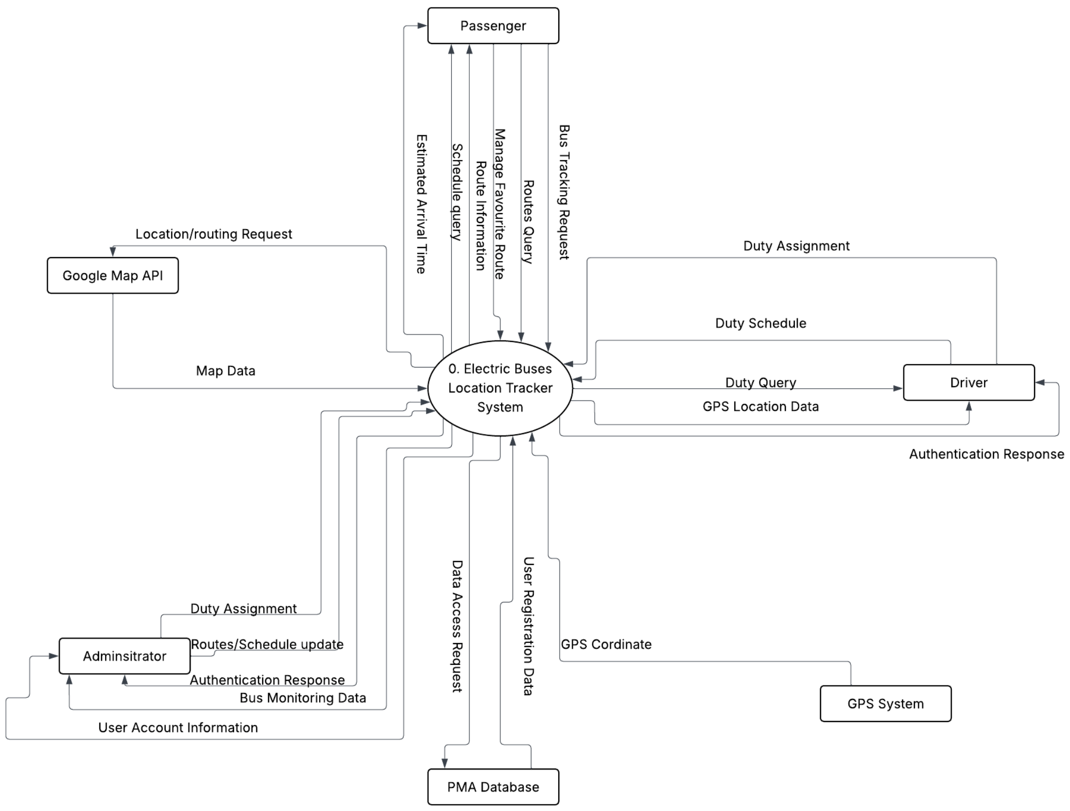
Figure 1: Use Case Diagram

4.2 Context Diagram (DFD Level 0)

Purpose: Shows the system as a single process with all external entities and data flows.

External Entities:

- Passenger, Driver, Administrator (users)
- GPS System, Google Map API, PMA Database (external systems)



Context diagram displays the Electric Bus Location Tracker System at center with bidirectional data flows to all external entities including user requests, GPS coordinates, map data, and authentication responses.

Figure 2: Context Diagram (Level 0 DFD)

4.3 Data Flow Diagram - Level 1

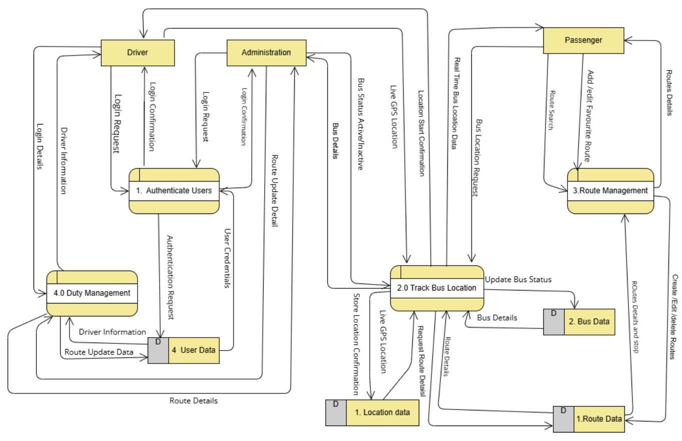
Purpose: Decomposes the system into major functional processes.

Major Processes:

1. Authenticate Users (FR-001 to FR-009)
2. Track Bus Location (FR-010 to FR-048, including real-time tracking and ETA)
3. Route Management (FR-014 to FR-022, FR-055 to FR-060)
4. Duty Management (FR-033 to FR-041, FR-067 to FR-071)

Data Stores:

- D1: Location data (Redis cache)
- D2: Bus Data (MongoDB)
- D3: Route Data (MongoDB)
- D4: User Data (MongoDB)



Level 1 DFD illustrates four main processes with data flows between users, processes, and data stores. Shows how authentication, tracking, route management, and duty management interact.

Figure 3: Data Flow Diagram - Level 1

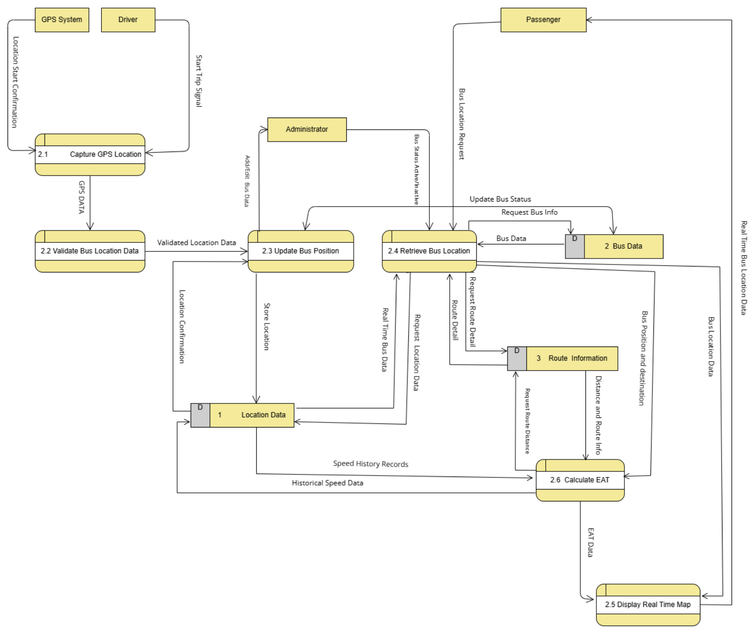
4.4 Data Flow Diagram - Level 2

Purpose: Provides detailed decomposition of core processes, particularly bus tracking and route management.

Level 2 Processes for Track Bus Location:

- 2.1 Capture GPS Location (FR-042 to FR-043)
- 2.2 Validate Bus Location Data (FR-044 to FR-045)
- 2.3 Update Bus Position (FR-046 to FR-048)

- 2.4 Retrieve Bus Location (FR-010 to FR-012)
- 2.5 Calculate EAT (FR-023 to FR-028)
- 2.6 Display Real-Time Map (FR-013, FR-018)



Level 2 DFD breaks down bus tracking into six sub-processes showing GPS capture, validation, position updates, retrieval, ETA calculation, and map display with detailed data flows.

Figure 4: Data Flow Diagram - Level 2 (Bus Tracking)

4.5 Sequence Diagrams

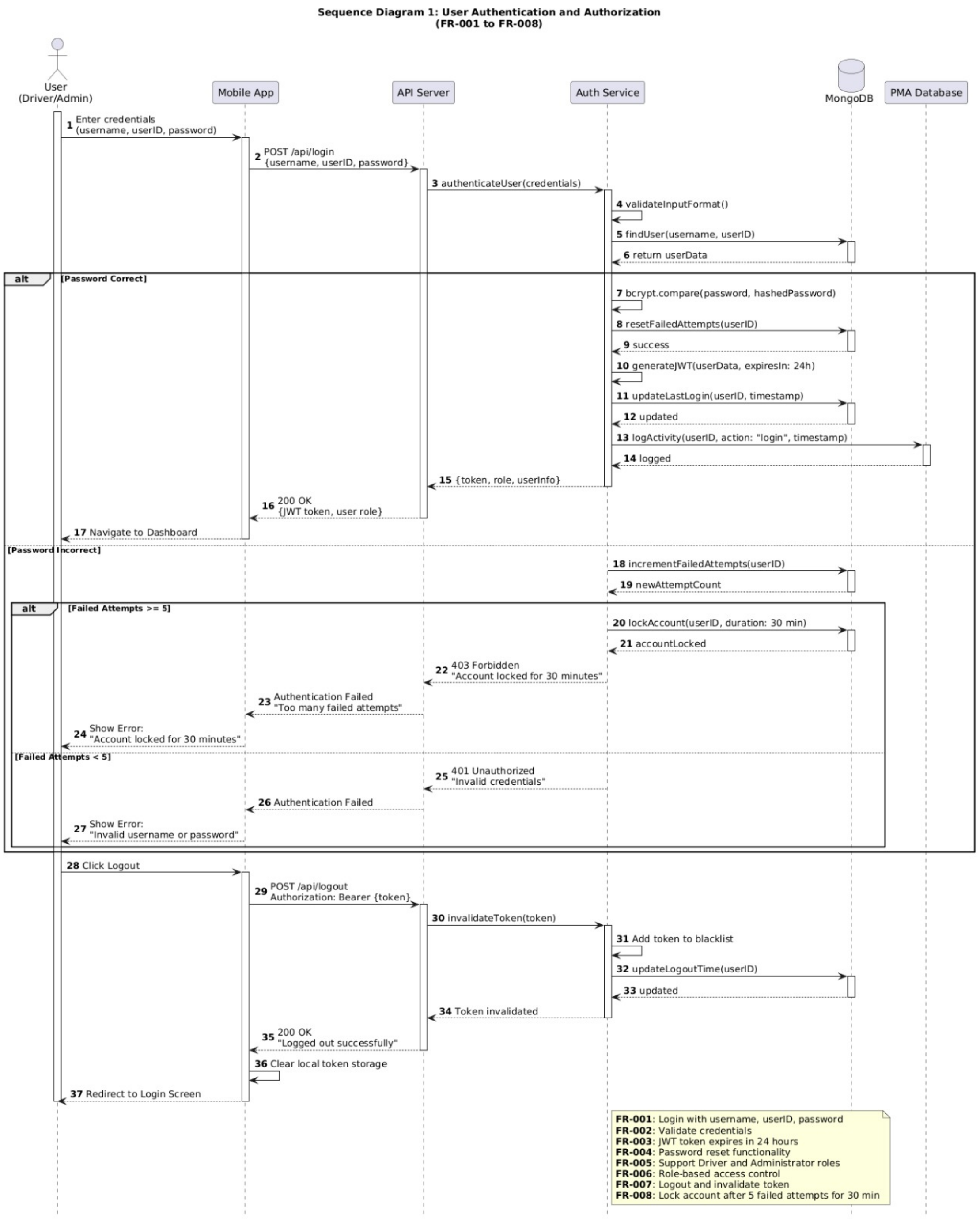
4.5.1 Sequence Diagram 1: User Authentication and Authorization

Coverage: FR-001 to FR-008

Participants: User (Driver/Admin), Mobile App, API Server, Auth Service, MongoDB, PMA Database

Key Flows:

- Successful login with JWT token generation
- Failed authentication with error handling
- Account lockout after 5 failed attempts
- Logout with token invalidation


Sequence Diagram 1: User Authentication and Authorization
(FR-001 to FR-008)
User (Driver/Admin)
Mobile App
API Server
Auth Service
MongoDB
PMA Database
1 Enter credentials (username, userID, password)
2 POST /api/login {username, userID, password}
3 authenticateUser(credentials)
4 validateInputFormat()
5 findUser(username, userID)
6 return userData
alt [Password Correct]
7 bcrypt.compare(password, hashedPassword)
8 resetFailedAttempts(userID)
9 success
10 generateJWT(userData, expiresIn: 24h)
11 updateLastLogin(userID, timestamp)
12 updated
13 logActivity(userID, action: "login", timestamp)
14 logged
15 {token, role, userInfo}
16 200 OK {JWT token, user role}
17 Navigate to Dashboard
[Password Incorrect]
18 incrementFailedAttempts(userID)
19 newAttemptCount
alt [Failed Attempts >= 5]
20 lockAccount(userID, duration: 30 min)
21 accountLocked
22 403 Forbidden "Account locked for 30 minutes"
23 Authentication Failed "Too many failed attempts"
24 Show Error: "Account locked for 30 minutes"
[Failed Attempts < 5]
25 401 Unauthorized "Invalid credentials"
26 Authentication Failed
27 Show Error: "Invalid username or password"
28 Click Logout
29 POST /api/logout Authorization: Bearer {token}
30 invalidateToken(token)
31 Add token to blacklist
32 updateLogoutTime(userID)
33 updated
34 Token invalidated
35 200 OK "Logged out successfully"
36 Clear local token storage
37 Redirect to Login Screen
FR-001: Login with username, userID, password
FR-002: Validate credentials
FR-003: JWT token expires in 24 hours
FR-004: Password reset functionality
FR-005: Support Driver and Administrator roles
FR-006: Role-based access control
FR-007: Logout and invalidate token
FR-008: Lock account after 5 failed attempts for 30 min

Sequence diagram shows complete authentication flow including credential validation, password verification using bcrypt, JWT token generation with 24-hour expiry, and account lockout mechanism after failed attempts.

Figure 5: Sequence Diagram - User Authentication

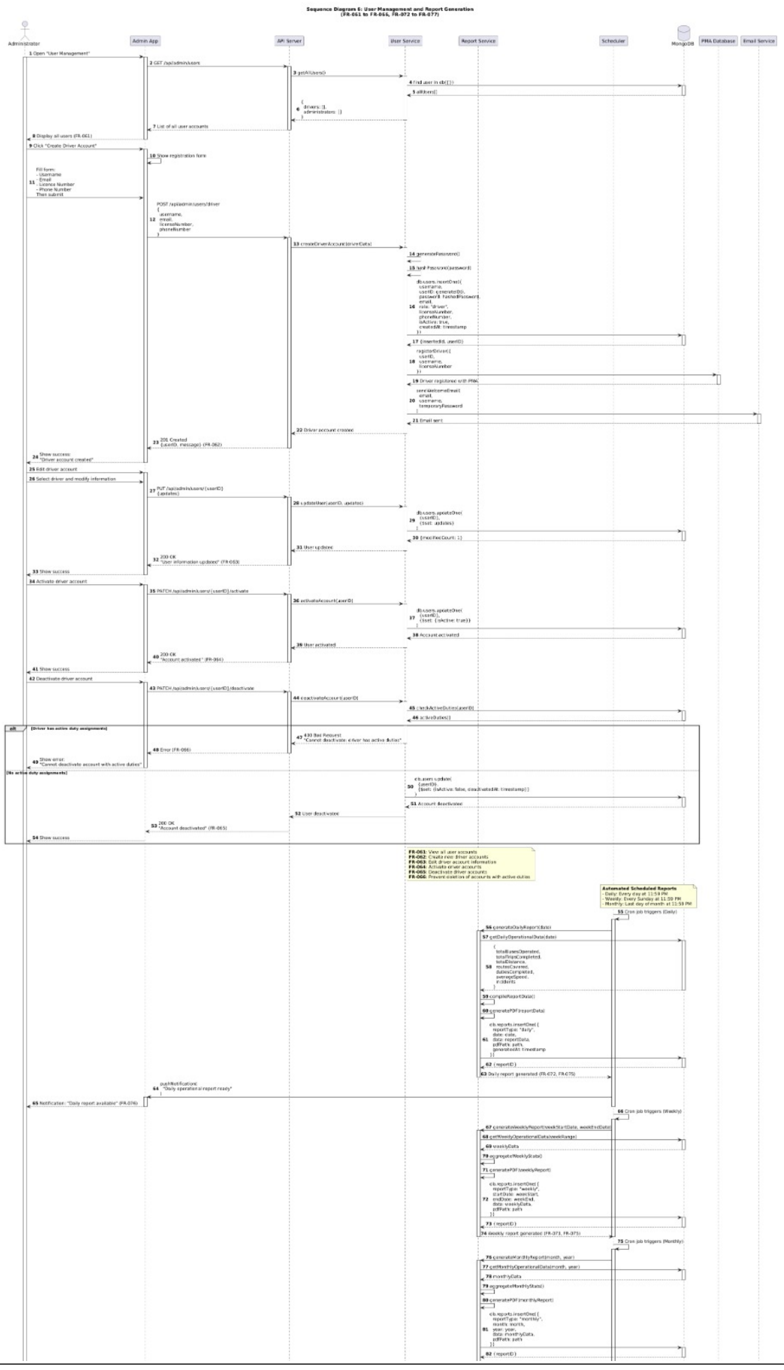
4.5.2 Sequence Diagram 2: Real-Time Bus Tracking and ETA Calculation

Coverage: FR-010 to FR-028, FR-042 to FR-048

Participants: Passenger, Driver, Mobile App, API Server, GPS Service, Report Service, Bus Tracking, Route Information, Location Data, Google Maps API

Key Interactions:

- GPS location capture and validation
- Real-time location updates via WebSocket
- ETA calculation using historical speed data
- Automatic 30-second ETA updates
- Bus arrival/departure notifications



Complex sequence diagram illustrating dual flows: driver location sharing through GPS and passenger real-time tracking with ETA updates. Shows WebSocket communication for live updates and Google Maps integration.

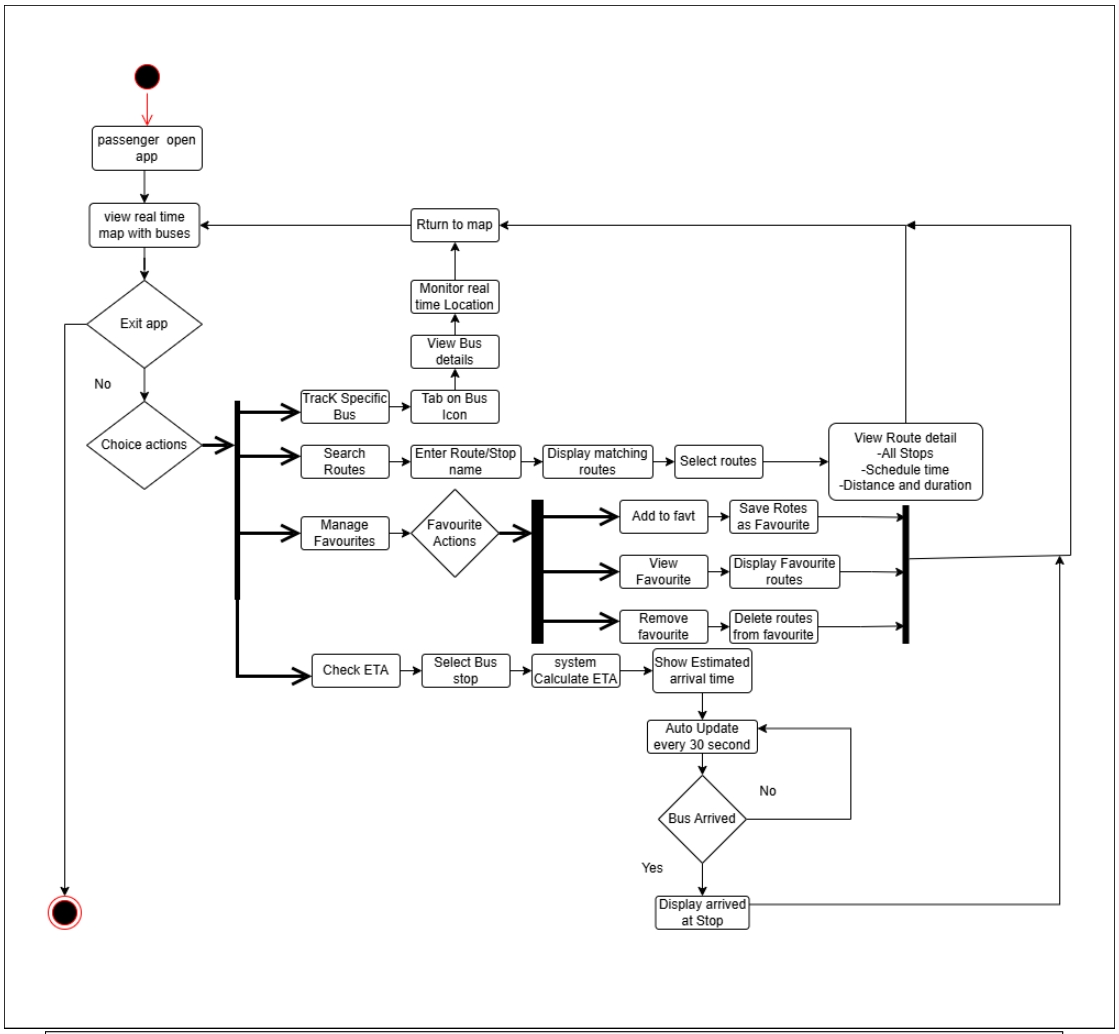
Figure 6: Sequence Diagram - Bus Tracking and ETA

4.6 Activity Diagrams

4.6.1 Activity Diagram 1: Passenger - Track Bus and View Routes

Coverage: FR-010 to FR-032
Main Flows:

- Login authentication
- Real-time bus tracking on map
- Route search by name or stop
- Favorite route management (add, view, remove)
- ETA checking with automatic updates
- Bus arrival/departure status monitoring



Activity diagram with swim lanes showing passenger journey from login through various tracking and route management activities, including parallel paths for different features.

Figure 7: Activity Diagram - Passenger Workflow

4.6.2 Activity Diagram 2: Driver - Duty Management and Location Sharing

Coverage: FR-001 to FR-009, FR-033 to FR-048

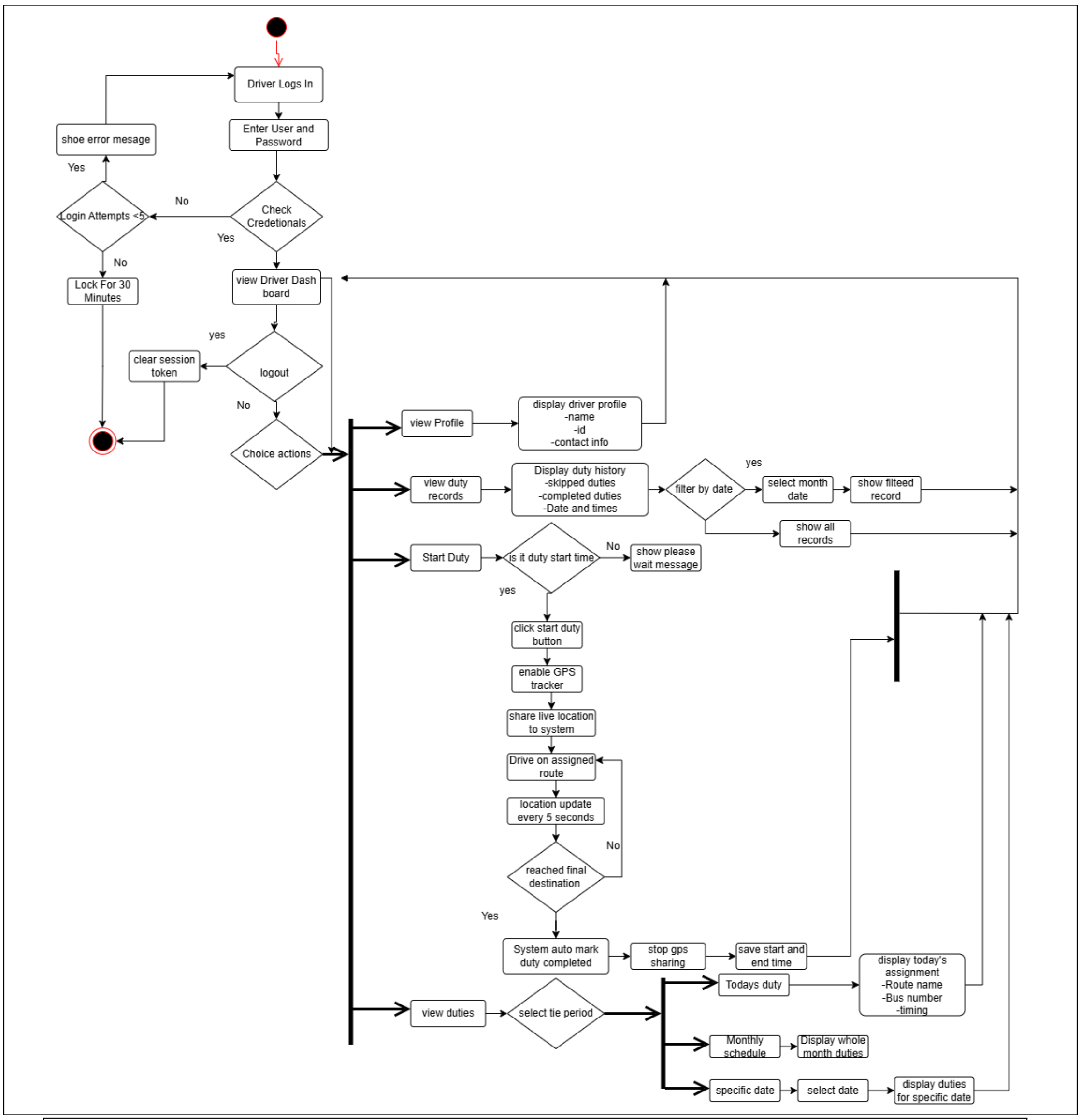
Main Flows:

- Driver authentication with account lockout
- View profile information
- Check duty records (completed/skipped) with date filtering

- Start duty at scheduled time
- Enable GPS tracker and share live location
- Drive assigned route with automatic location updates (5-second intervals)
- Automatic duty completion upon reaching destination
- View duties (today's, monthly, specific date)

Decision Points:

- Login attempts validation (max 5)
- Duty start time check
- Destination reached verification
- Duty view period selection
- Filter by date option



Comprehensive activity diagram showing driver workflows including login security, duty viewing with filters, GPS-based location sharing during active duty, and automatic duty completion mechanism.

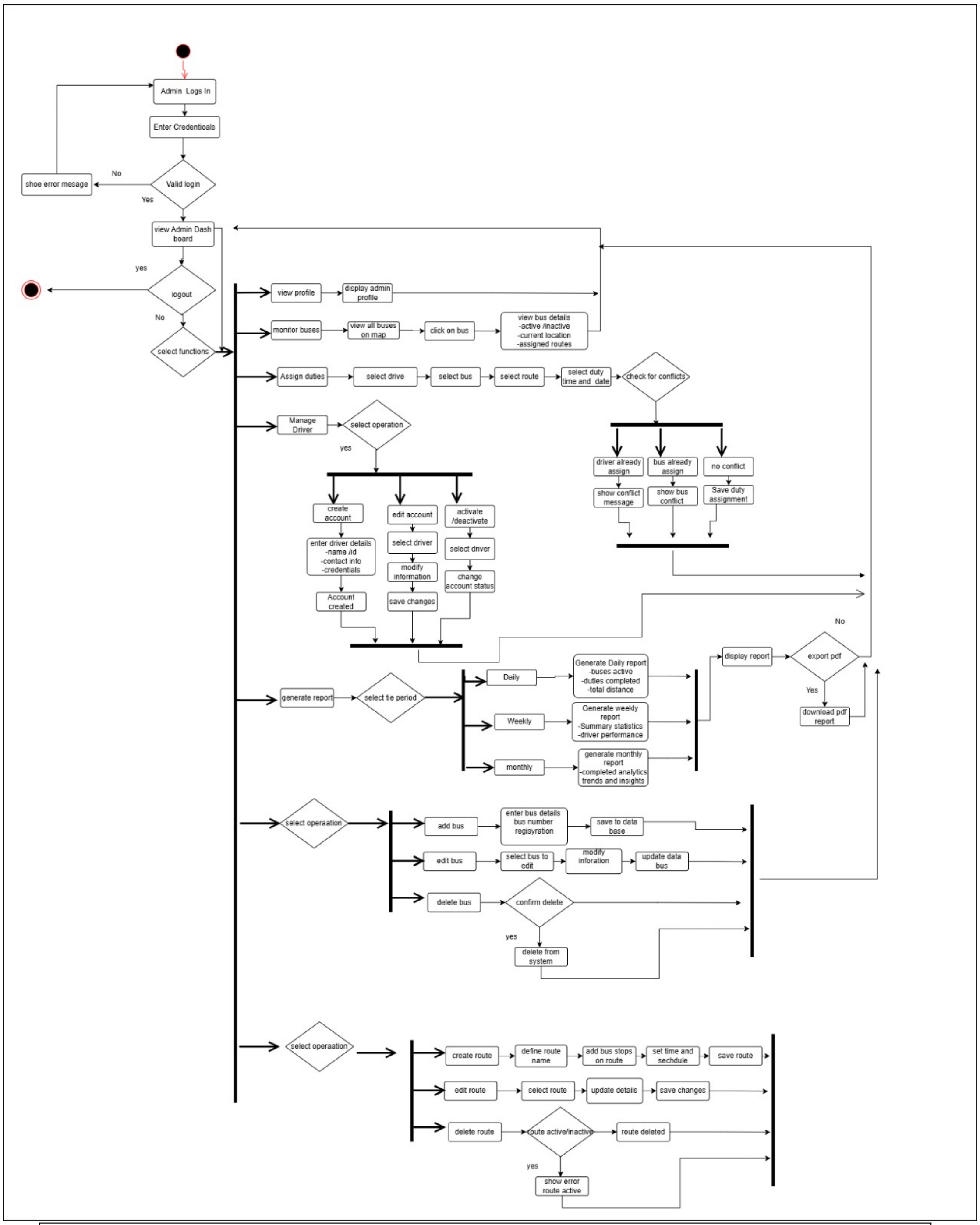
Figure 8: Activity Diagram - Driver Duty Management

4.6.3 Activity Diagram 3: Admin - Adding duty, Adding routes, Adding drivers besides with also editing them and report generation

Coverage: FR-001 to FR-009, FR-049 to FR-077

Main Flows:

- Admin authentication with account lockout
- View profile information
- Generate monthly report
- Add and edit duties
- Add and edit driver information
- Delete driver account
- Add routes to the map and assigned stops
- Edit and add stops for the route
- View Detail of all the routes



Comprehensive activity diagram showing admin workflows including system management, user account administration, route and duty management, and report generation capabilities.

Figure 9: Activity Diagram - Admin Management

4.7 Class Diagram

Purpose: Represents the object-oriented structure of the system with classes, attributes, operations, and relationships.

Key Classes and Relationships:

1. **User (Abstract Base Class)**
   - Attributes: userId, username, password, email, phoneNumber, createdAt, isActive
   - Operations: login(), resetPassword(), viewProfile(), logout()
   - Subclasses: Passenger, Driver, Administrator
2. **Passenger → User**
   - Attributes: passengerId, favoriteRoutes
   - Operations: viewRealTimeBusLocation(), searchRoutes(), calculateETA(), manageFavoriteRoutes(), viewRouteDetails()
   - Relationships: uses Route (0..*), checks ETA (0..*)
3. **Driver → User**
   - Attributes: driverId, licenseNumber
   - Operations: startDuty(), viewCurrentDutyDetail(), viewMonthlyDutyDetail(), viewDutyRecordsByDate(), shareLiveLocation(), completeDuty()
   - Relationships: assigned to Bus (0..1), performs Duty (0..*), shares Location (1)
4. **Administrator → User**
   - Attributes: adminId, role, permissions
   - Operations: viewAllBuses(), addBus(), editBusDetails(), deleteBus(), createRoute(), modifyRoute(), defineSchedule(), createDriverAccount(), assignDuty(), modifyDutyAssignment(), exportReports()
   - Relationships: manages Bus (*), manages Route (*), creates Duty (*), generates Report (*)
5. **Bus**
   - Attributes: busId, busNumber, capacity, status, createdAt, updatedAt
   - Operations: updateStatus(), getBusDetails(), isActive()
   - Relationships: composition with Tracking (1), aggregation with Driver (0..1), associated with Route (0..1)
6. **Route**
   - Attributes: routeId, routeName, startPoint, endPoint, totalDistance, estimatedTime, isActive, createdAt

- Operations: getRouteDetails(), addStop(), removeStop(), getStopsList(), calculateTotalDistance()
- Relationships: composition with Stop (2..*), associated with Schedule (0..1)

7. **Stop**
   - Attributes: stopId, stopName, sequence, arrivalTime
   - Operations: getStopDetails(), updateArrivalTime()
   - Relationships: part of Route
8. **Duty (Association Class)**
   - Attributes: dutyId, date, startTime, endTime, status, actualStartTime, completionStatus
   - Operations: completeDuty(), getDutyDetails(), isCompleted()
   - Relationships: links Driver to Bus and Route
9. **Tracking**
   - Attributes: trackingId, latitude, longitude, speed, startedAt, stoppedAt, startTracking(), stopTracking()
   - Operations: updateLocation(), getLocationHistory(), getCurrentLocation()
10. **ETA (Estimated Arrival Time)**
    - Attributes: etaId, estimatedTime, distanceRemaining, lastCalculatedAt, arrivalStatus
    - Operations: calculateETA(), updateETA(), getETADetails(), hasArrived()
11. **Location**
    - Attributes: locationId, latitude, longitude, altitude, speed, bearing, timestamp, accuracy
    - Operations: getDistance(), calculateDistanceOf(), isValid()
12. **Authentication**
    - Attributes: sessionId, userId, tokenId, expiresAt
    - Operations: generateToken(), validateToken(), revokeToken()
13. **Report**
    - Attributes: reportId, type, generatedAt, period, format
    - Operations: generateDailyReport(), generateWeeklyReport(), generateMonthlyReport(), exportToPDF()
14. **Schedule**

- Attributes: scheduleId, departureTime, arrivalTime, frequency, daysOfWeek, isActive
- Operations: getScheduleDetails(), updateSchedule()

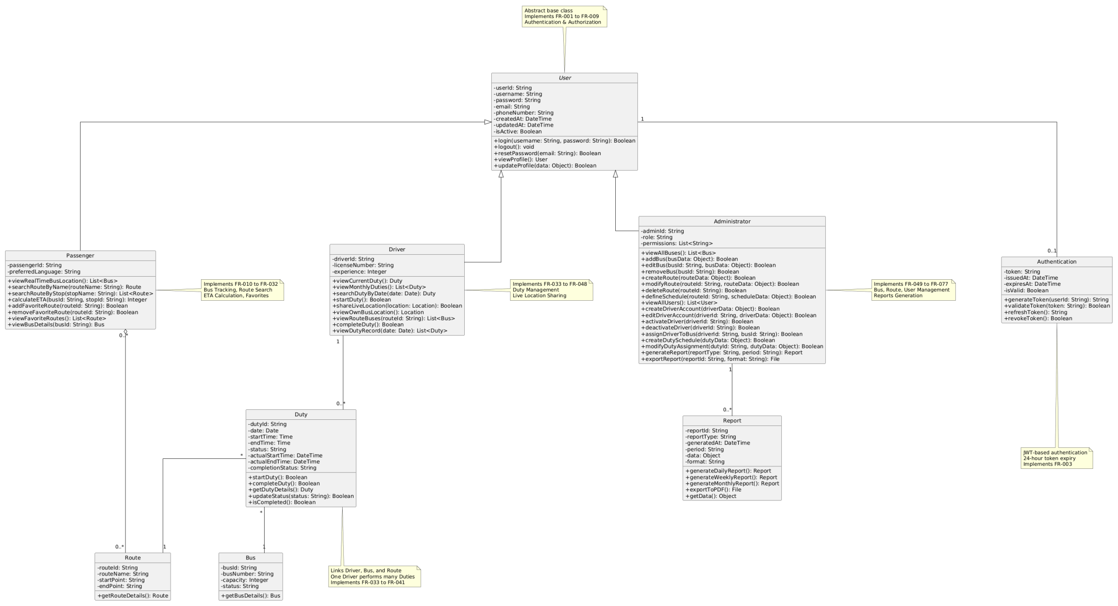

Abstract base class
Implements FR-001 to FR-009
Authentication & Authorization
User
-userId: String
-username: String
-password: String
-email: String
-phoneNumber: String
-createdAt: DateTime
-updatedAt: DateTime
-isActive: Boolean
+login(username: String, password: String): Boolean
+logout(): void
+resetPassword(email: String): Boolean
+viewProfile(): User
+updateProfile(data: Object): Boolean
1
Passenger
-passengerId: String
-preferredLanguage: String
+viewRealTimeBusLocation(): List<Bus>
+searchRouteByName(routeName: String): Route
+searchRouteByStop(stopName: String): List<Route>
+calculateETA(busId: String, stopId: String): Integer
+addFavoriteRoute(routeId: String): Boolean
+removeFavoriteRoute(routeId: String): Boolean
+viewFavoriteRoutes(): List<Route>
+viewBusDetails(busId: String): Bus
Implements FR-010 to FR-032
Bus Tracking, Route Search
ETA Calculation, Favorites
Driver
-driverId: String
-licenseNumber: String
-experience: Integer
+viewCurrentDuty(): Duty
+viewMonthlyDuties(): List<Duty>
+searchDutyByDate(date: Date): Duty
+startDuty(): Boolean
+shareLiveLocation(location: Location): Boolean
+viewOwnBusLocation(): Location
+viewRouteBuses(routeId: String): List<Bus>
+completeDuty(): Boolean
+viewDutyRecord(date: Date): List<Duty>
Implements FR-033 to FR-048
Duty Management
Live Location Sharing
Administrator
-adminId: String
-role: String
-permissions: List<String>
+viewAllBuses(): List<Bus>
+addBus(busData: Object): Boolean
+editBus(busId: String, busData: Object): Boolean
+removeBus(busId: String): Boolean
+createRoute(routeData: Object): Boolean
+modifyRoute(routeId: String, routeData: Object): Boolean
+deleteRoute(routeId: String): Boolean
+defineSchedule(routeId: String, scheduleData: Object): Boolean
+viewAllUsers(): List<User>
+createDriverAccount(driverData: Object): Boolean
+editDriverAccount(driverId: String, driverData: Object): Boolean
+activateDriver(driverId: String): Boolean
+deactivateDriver(driverId: String): Boolean
+assignDriverToBus(driverId: String, busId: String): Boolean
+createDutySchedule(dutyData: Object): Boolean
+modifyDutyAssignment(dutyId: String, dutyData: Object): Boolean
+generateReport(reportType: String, period: String): Report
+exportReport(reportId: String, format: String): File
Implements FR-049 to FR-077
Bus, Route, User Management
Reports Generation
0..1
Authentication
-token: String
-issuedAt: DateTime
-expiresAt: DateTime
-isValid: Boolean
+generateToken(userId: String): String
+validateToken(token: String): Boolean
+refreshToken(): String
+revokeToken(): Boolean
JWT-based authentication
24-hour token expiry
Implements FR-003
0..*
1
0..*
1
0..*
Report
-reportId: String
-reportType: String
-generatedAt: DateTime
-period: String
-data: Object
-format: String
+generateDailyReport(): Report
+generateWeeklyReport(): Report
+generateMonthlyReport(): Report
+exportToPDF(): File
+getData(): Object
Duty
-dutyId: String
-date: Date
-startTime: Time
-endTime: Time
-status: String
-actualStartTime: DateTime
-actualEndTime: DateTime
-completionStatus: String
+startDuty(): Boolean
+completeDuty(): Boolean
+getDutyDetails(): Duty
+updateStatus(status: String): Boolean
+isCompleted(): Boolean
*
*
0..*
1
1
Route
-routeId: String
-routeName: String
-startPoint: String
-endPoint: String
+getRouteDetails(): Route
Bus
-busId: String
-busNumber: String
-capacity: Integer
-status: String
+getBusDetails(): Bus
Links Driver, Bus, and Route
One Driver performs many Duties
Implements FR-033 to FR-041

Figure 10: Class Diagram - Part 1: User Hierarchy and Core Entities

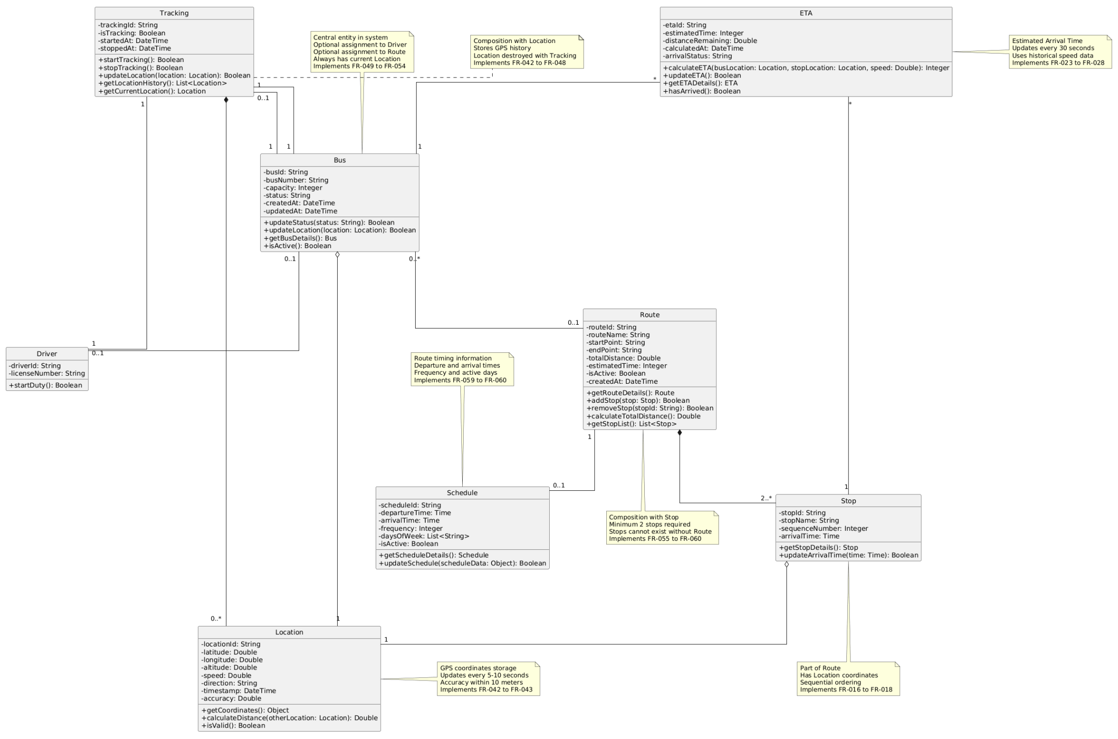


Figure 11: Class Diagram - Part 2: Supporting Classes and Relationships

Key Relationship Types:

- **Inheritance:** Passenger, Driver, Administrator inherit from User
- **Composition:** Route contains Stops (strong ownership), Bus contains Tracking
- **Aggregation:** Driver assigned to Bus (weak ownership)
- **Association:** Duty links Driver, Bus, and Route with temporal attributes

4.8 Component Diagram

Purpose: Illustrates high-level system components and their dependencies.

System Architecture - Three Tiers:

1. **Presentation Tier**
   - Mobile App (Flutter)
   - Interfaces: IAuth, ITracking, IRoute, IDuty, IMap
2. **Business Logic Tier**
   - Auth Component - User authentication and authorization

- Bus Tracking Component - Real-time location processing
- Route Management Component - Route and schedule operations
- Duty Management Component - Driver duty assignments
- Data Repository Component - Data access abstraction

3. **Data Access Tier (Persistence Tier)**
   - MongoDB - Persistent storage (users, buses, routes, duties)
   - Redis - Real-time location cache

External Systems:

- Google Maps API - Mapping and routing services
- GPS System - Location data provider

Communication Protocols:

- REST API (HTTP/HTTPS) - Client-server communication
- WebSocket (Socket.io) - Real-time updates
- JDBC interfaces for database connections

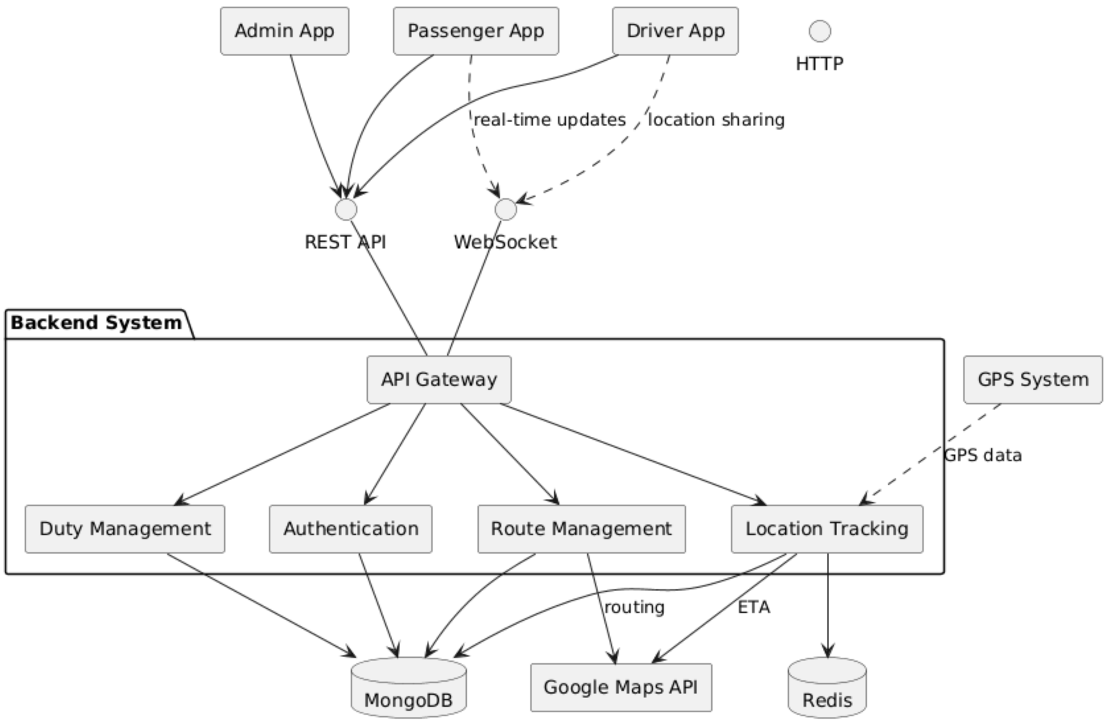
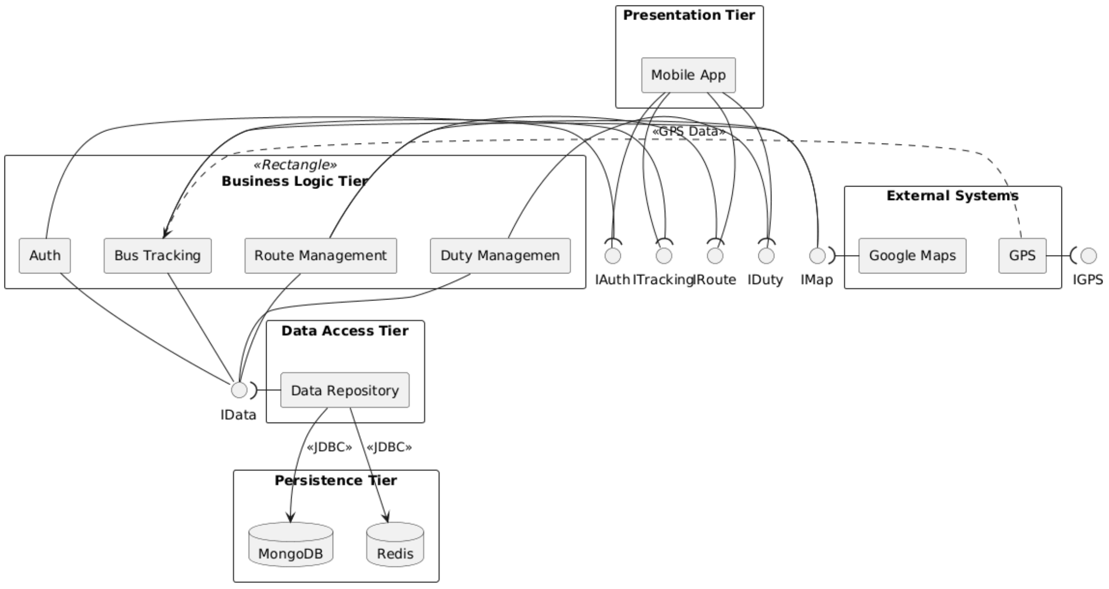


Figure 12: Component Diagram - Part 1: System Architecture Overview



Component diagram displaying three-tier architecture with Presentation layer (Mobile App), Business Logic layer (Auth, Bus Tracking, Route Management, Duty Management, Data Repository), and Persistence layer (MongoDB, Redis). Shows interface dependencies and external system integrations.

Figure 13: Component Diagram - Part 2: Detailed Component Interactions

5 Requirements–Design Traceability Table

This table maps functional requirements from the SRS to their corresponding design artifacts, ensuring complete coverage and traceability.

Link to Google sheet of Traceability Table:
`https://docs.google.com/spreadsheets/d/1vjEYH0nXio0jEPt5wevlva5RH6Ut9zjqmzuHLO7zVf`
`edit?usp=sharing`

6 GitHub and Figma Links

6.1 GitHub Repository

Repository URL: `https://github.com/Mehrozalikhan01/Electric-buses-location-tracker`
`git`

6.2 Figma Prototype

Figma Link: `https://www.figma.com/design/BoCShru8p7isvqPn427vHa/Untitled?`
`node-id=13-6881&t=kgQ1WCa5wS2Oaihv-1`

Prototype Coverage:

- **Passenger Interface:**
  - Real-time bus tracking map
  - Route search and display
  - ETA calculation screen
  - Favorite routes management
- **Driver Interface:**
  - Login and authentication
  - Duty schedule view (daily, monthly, specific date)
  - Start duty and location sharing
  - Duty completion confirmation
  - Profile view
- **Administrator Interface:**
  - Dashboard with system overview
  - Bus management (add, edit, delete, monitor)
  - Route and schedule management
  - Driver account management
  - Duty assignment interface
  - Reports generation

Interactive Features:

- Clickable navigation between screens
- Real-time update simulations
- Map interaction prototypes

6.3 Meeting Minutes Data

Google Sheet Link: `https://docs.google.com/spreadsheets/d/1WLFOwPuELaKaJOVSwRVZz3am`
`edit?usp=drivesdk`