



## Software Design Specification

( GreenRoute )

**Project Code:**

BSCS-AWM-153435 -2025

**Internal Advisor:**

Dr. Hussam Ali

A handwritten signature in blue ink, appearing to read 'Hussam Ali', with the date '7-1-26' written below it.

**Project Manager:**

Prof. Dr. Muhammad Ilyas

**Project Team:**

Asma Khanum (TL) - BSCS51F22S015

Mahnoor (TM) - BSCS51F22S034

Wajeeha Safdar (TM) - BSCS51F22S035

**Submission Date:**

22 December 2025

---

Project Manager's Signature

## Document Information

Category	Information
Customer	University of Sargodha (SU)
Project	GreenRoute
Document	Software Design Specification
Document Version	2.0
Identifier	BSCS-AWM-153435 -2025
Status	Draft
Author(s)	Asma Khanum (TL) - BSCS51F22S015 Mahnoor (TM) - BSCS51F22S034 Wajeeha Safdar (TM) - BSCS51F22S035
Approver(s)	PM
Issue Date	Nov. 18, 2025
Document Location	
Distribution	1. Supervisor 2. PM 3. Project Team

## Definition of Terms, Acronyms and Abbreviations

Term	Description
PWA	Progressive Web Application
DFD	Design Specification
GPS	Global Positioning System
ETA	Estimated Time of Arrival
IOS	iPhone Operating System
JS	JavaScript
API	Application Programming Interface
RBAC	Role Base Access Control
HTTP	Hypertext Transfer Protocol
SNDA	Serverless Real-Time Data architecture
UI	User Interface
GUI	Graphical User Interface
ER	Entity Relation
HTML	HyperText Markup Language

## Table of Contents

---

<b>1. Introduction .....</b>	<b>4</b>
1.1 Purpose of Document.....	4
1.2 Project Overview .....	4
1.3 Scope .....	4
<b>2. Design Considerations .....</b>	<b>5</b>
2.1 Assumptions and Dependencies .....	5
2.2 Risks and Volatile Areas .....	6
<b>3. System Architecture .....</b>	<b>8</b>
3.1 System Level Architecture.....	8
3.2 Sub-System / Component / Module Level Architecture .....	9
3.3 Sub-Component / Sub-Module Level Architecture (1...n).....	9
<b>4. Design Strategies .....</b>	<b>10</b>
4.1 Strategy 1: Serverless, Real-Time Data Architecture (SNDA).....	10
4.2 Strategy 2: Modular, Tiered Application Structure .....	11
4.3 Strategy 3: Progressive Web Application (PWA) Client Design .....	11
<b>5. Detailed System Design.....</b>	<b>12</b>
5.1 Class Diagram Description.....	12
5.2 Sequence Diagram .....	17
5.3 State Transition Diagram .....	18
5.4 Logical data model (E/R model).....	19
5.5 Physical data models .....	20
5.6 GUIs .....	21
5.7 Data Flow Diagrams .....	22
<b>6. References .....</b>	<b>23</b>

## 1. Introduction

### 1.1 Purpose of Document

The purpose of this document is to describe what the Real-Time Web-Based Bus Tracking System will do and what is required to build it. It clearly explains the goals of the system, its main features, and how it should work so that everyone involved understands the same thing. This document helps guide the design, development, and testing of the project. It will be used to make sure the final system meets all the needs and expectations of its users.

This document will serve as a guide for all project stakeholders including the project manager, supervisor, development team and prospective users.

### 1.2 Project Overview

The project under study is a Real-Time Web-Based Bus Tracking System that allows users to view the live location of buses operating in Sargodha. The system uses GPS technology to collect the location of each bus and displays it on an interactive Google Map within a web application.

The software will be used by passengers to check the current position, route, and estimated arrival time of buses. Transport administrators can also use it to monitor bus movement, manage routes, and ensure that services are running on schedule. The system will be accessible through any device with an internet connection, such as a smartphone or computer.

#### Goals:

1. To provide accurate, real-time information about bus locations and arrival times.
2. To reduce passenger waiting time and improve transport reliability.
3. To support efficient route and fleet management for transport authorities.

#### Benefits:

1. Saves time and increases convenience for passengers.
2. Improves coordination between drivers and administrators.
3. Enhances trust and satisfaction in public transportation.
4. Contributes to the digital transformation of local transport systems.

### 1.3 Scope

The proposed system focuses on the development of a web-based real-time bus tracking platform designed specifically for the local E-Bus transport network launched under the initiative of Chief Minister Maryam Nawaz in Sargodha, Pakistan.

The boundaries of the proposed system are defined below to ensure clear deliverables.

#### In-Scope:

1. Designing and developing a responsive web application that displays real-time bus locations on an interactive Google Maps interface.
2. Utilizing GPS modules integrated within buses or drivers' mobile devices to capture live coordinate data.
3. Implementing a centralized backend database for continuous data synchronization and route management.
4. Providing users with essential transport details such as bus number, route, estimated arrival time (ETA), and current position.
5. Incorporating notification features to alert passengers of bus arrivals or route delays.
6. Ensuring the system's functionality and usability across desktop and mobile browsers through PWA design principles.

**Out-of-Scope:**

1. Integration with digital ticketing or payment systems.
2. Development of a dedicated native mobile application (Android/iOS).
3. Implementation of AI-based predictive analytics or route optimization algorithms.
4. Coverage of private, intercity, or non-governmental transport vehicles.
5. Operate offline; it will require an active internet connection for its services.

## 2. Design Considerations

The following section describes the key factors that influenced the system design decisions. It explains the assumptions made during development, the dependencies on external components and technologies, and the overall impact of these choices on system performance, scalability, and usability.

### 2.1 Assumptions and Dependencies

Following are assumptions and dependencies for the project that are already captured in the SRS document

#### Design Impact of Assumptions

The design must be capable enough of handling the limitations imposed by the core assumptions:

- **GPS-Enabled Devices and Active Tracking:**
  - **Design Choice:** The system must be battery efficient for the Driver's smartphone to ensure continuous GPS and active tracking throughout the travel experience.

- **Implication:** The mobile driver component (an internal feature, basically a PWA wrapper for the GPS transmission) must be a very lightweight design and have minimal background processing.
- **Internet Access and Modern Browser:**
  - **Design Choice:** The frontend (**React.js PWA**) must follow **Progressive Web Application** principles to ensure it works smoothly on different devices and modern browsers (Chrome, Edge, Safari).
  - **Implication:** Design must be naturally adaptive.
- **Availability of Institutional Data/Permissions:**
  - **Design Choice:** The **Admin Dashboard** must include comprehensive features for managing bus routes and associated data, given that the transport authority provides the initial dataset.

## Design Impact of Dependencies

The PWA's architecture is a cross-service integration, resulting in a design that depends on third party APIs and services:

- **Firebase Dependency:**
  - **Design Choice:** For live data handling the core architecture of our PWA must be centered on Firebase **Real time Database**.
  - **Implication:** Data in Firebase must be optimized for real-time reads and writes of latitude/longitude/timestamp data ensuring timely location updates.
- **Google Maps API Dependency:**
  - **Design Choice:** All functionalities related to map visualization, route drawing, and ETA computation will be based on the **Google Maps API**.
  - **Implication:** The React application needs a reliable and efficient component for managing API keys and handling map rendering.

## 2.2 Risks and Volatile Areas

Following are the potential areas of risks, that the system design must mitigate to make it efficient:

Risk/Volatile Area	Impact on Design	Contingency Path/Mitigation
<b>Google Maps/Firebase Free-Tier Usage Limits</b>	May experience service interruption or additional charges when traffic exceeds the free-tier limit.	<b>Mitigation:</b> Implement strict data transmission protocols to minimize unnecessary requests (e.g., sending updates only when the bus position significantly changes). <b>Contingency:</b> Design the backend data processing layer for seamless adaptability with both paid cloud services and self-hosted real-time databases if cost becomes a limitation.
<b>Real-Time Data Inaccuracy</b>	If GPS signal is weak or network is unstable, the bus location on the map will be delayed or varied irregularly.	<b>Mitigation:</b> Implement client-side smoothing algorithms on the frontend to estimate intermediate bus positions between official updates, providing smooth marker movement for the passenger. Implement a "Last Updated" timestamp display on each bus to inform users during poor connectivity areas.
<b>Security and Data Privacy</b>	Sensitive data (e.g., driver logs, passwords) could be compromised or exposed if security is not handled correctly.	<b>Mitigation:</b> Force HTTPS encryption for all client-server communication. Centralize authentication using Firebase Auth. Enforce Role-Based Access Control (RBAC) to restrict unauthorized users from admin/driver features.

Following are the potential upcoming developments, that we are considering for our system development:

Upcoming Developments	Design Impact	Recommended Design Strategy
<b>Future Integration of Digital Ticketing</b>	The current scope excludes payment systems, but are expected to be required in the future.	<b>Design Strategy (Modularity):</b> Design the Admin/User modules to be <b>modular</b> with clear separation of service layers. Ensure the user registration and login flow is independent from the tracking services so that a future payment module can be integrated as a separate microservice with minimum impact.
<b>Future Native Mobile App</b>	The current scope is PWA-only, but a	<b>Design Strategy (Technology Stack):</b> The use of Next.js for the frontend supports this approach, as it

<b>Development</b>	native app is a common next step for better performance.	can be leveraged with React Native for subsequent native mobile application development, ensuring the initial front-end codes remain valuable.
<b>Adding New Languages (English Only Constraint)</b>	Currently supports English only, but a local project may need Urdu or other languages support	<b>Design Strategy (Internationalization):</b> Build the frontend using a framework that supports <b>internationalization (i18n)</b> from the start. All user-facing text should be extracted into language resource files, rather than being hard-coded in the React components.

### 3. System Architecture

The following system architecture overview describes the decomposition of system responsibilities across its primary subsystems. The diagrams below provide a visual mapping of the system hierarchy, from high-level design to sub-component execution.

#### 3.1 System Level Architecture

The system architecture provides a top-level architecture providing a foundation for more detailed design work.

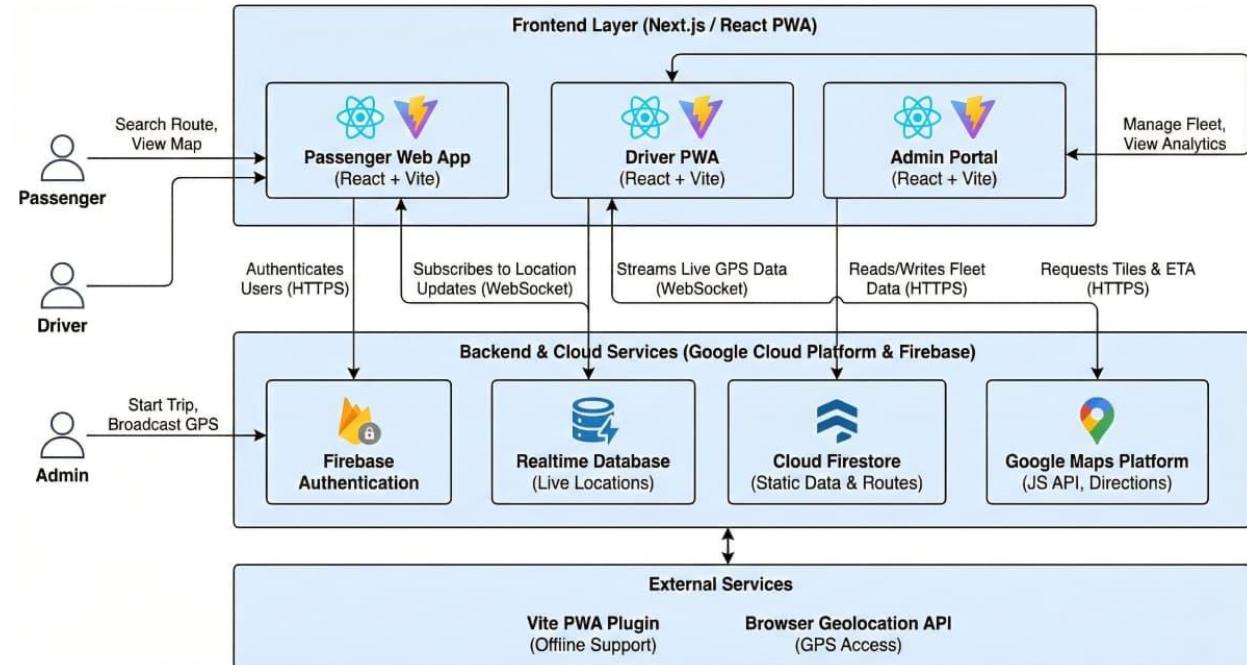


Figure 3.1: High-Level System Architecture of GreenRoute

## 3.2 Sub-System / Component / Module Level Architecture

The following diagrams present the subsystems, components, and modules, showing their relationships and interactions.

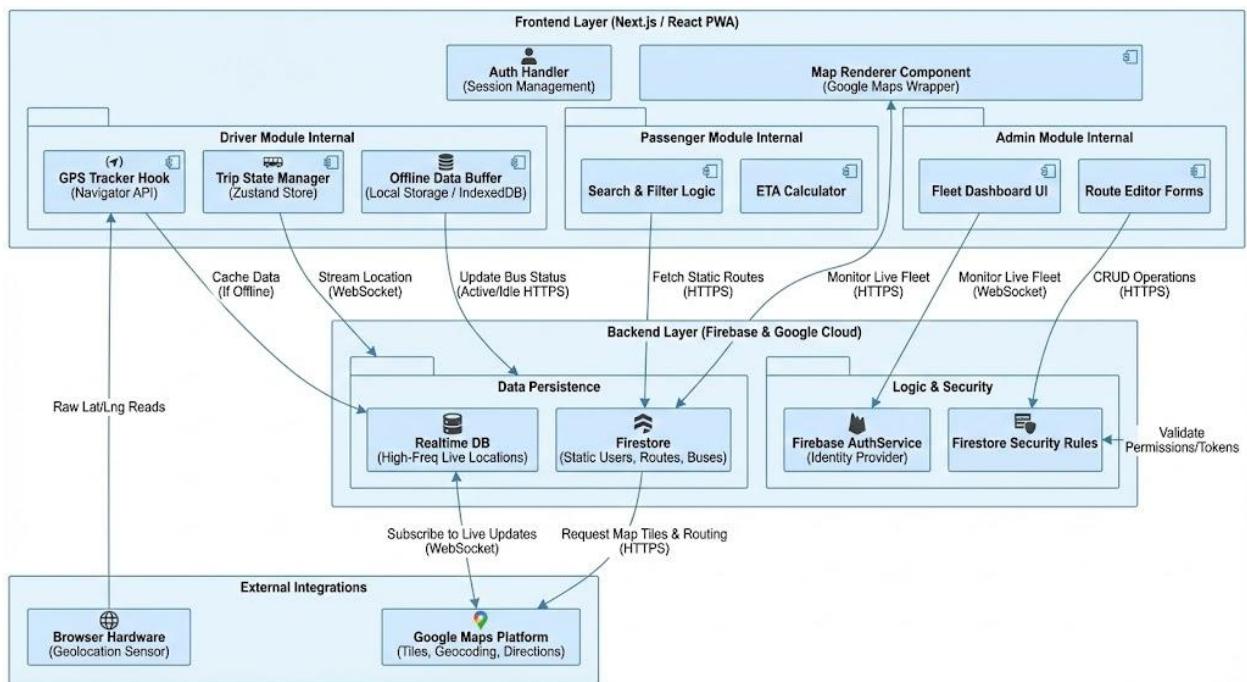


Figure 3.2: Component-Level Architecture

## 3.3 Sub-Component / Sub-Module Level Architecture (1...n)

The following diagrams illustrate the decomposition of modules and components into subcomponents and submodules and their relationships.

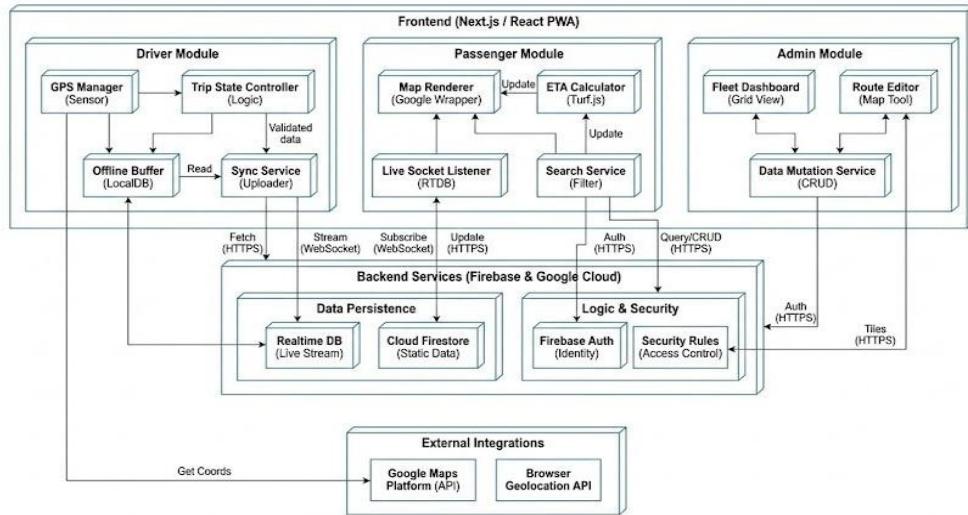


Figure 3.3: Detailed Sub-Component Level Architecture

## 4. Design Strategies

This section explains the key design strategies that govern the high-level structure and organization of the **GreenRoute**. These decisions ensure the project's non-functional requirements, such as performance, scalability, and maintainability.

### 4.1 Strategy 1: Serverless, Real-Time Data Architecture (SNDA)

The following sections present the design strategies considered for the system, detailing the reasoning behind each and areas of impact.

Area	Discussion and Reasoning	Trade-Offs
<b>Data Management</b>	The system is designed around the <b>Firebase Realtime Database</b> . This dictates a NoSQL data model optimized for speed and real-time synchronization. GPS data (latitude, longitude, timestamp) is pushed directly from the Driver application to Firebase, which acts as the central hub.	<b>Vendor Lock-In:</b> The system's dependency on Firebase may hinder flexibility and high costs for transitioning to a different hosting platform in future.
<b>Concurrency &amp; Synchronization</b>	Firebase automatically handles updates from multiple buses drivers and broadcasts the latest locations to all connected users, avoiding the need to manage server-side real-time connections.	<b>Cost Sensitivity:</b> High-frequency updates from multiple buses and users may lead to increase in Firebase usage cost.

<b>System Reuse</b>	The Firebase backend services are reusable for any future related application , as they provide a unified API.	<b>Data Complexity:</b> NoSQL data models (like Firebase's) complicates relational queries (e.g., finding all buses near a specific bus stop and calculating their ETA), requiring denormalization or extra cloud functions.
---------------------	--	--

## 4.2 Strategy 2: Modular, Tiered Application Structure

The system is divided into self-contained modules for each main user role—Admin, Driver, and User and a core set of shared services. This structure ensures separation of concerns and facilitates future development.

Area	Discussion and Reasoning	Trade-Offs
<b>Future System Extension</b>	The system treats the <b>Admin Dashboard</b> , <b>User Portal</b> , and the <b>Driver Service</b> as separate modules, enabling new features like driver analytics—while maintaining the integrity of the core GPS tracking functionality.	<b>Increased Initial Complexity:</b> Developing the system with strict module boundaries and interfaces adds overhead to the initial setup compared with a single-tier architecture
<b>System Reuse</b>	The <b>Location Data Service</b> (the layer that reads from Firebase and interacts with the Google Maps API) is isolated. This service can be reused by any client (web, mobile, or third-party display boards) that needs real-time bus locations	<b>Inter-Module Communication:</b> Ensuring secure and reliable communication between the client apps and Firebase requires strict enforcement of security rules and consistent data formatting.
<b>User Interface Paradigms</b>	Each tier has a tailored interface: <b>Admins</b> have a desktop dashboard with full features, <b>Users</b> have a mobile-centric map view; and <b>Drivers</b> have a lightweight interface focused on GPS submissions	<b>Maintenance Overhead:</b> Maintaining three separate user interfaces, have same underlying frontend frameworks, increases testing and deployment overhead.

## 4.3 Strategy 3: Progressive Web Application (PWA) Client Design

The system will be delivered as a Progressive Web Application (PWA) using React.js. This addresses the requirement for a "Web-Based" solution while mitigating the performance and user experience shortcomings of a traditional web application.

Area	Discussion and Reasoning	Trade-Offs
<b>User Interface Paradigms</b>	PWA technology enables the system to be installed directly on a user's mobile device or desktop, offering an <b>app-like experience</b> with splash screens, icons, and full-screen mode, without relying on app stores.	<b>Limited Native Integration:</b> PWAs have reduced access to advanced device features as compared to native apps
<b>Data Management</b>	PWA capabilities allow <b>offline caching</b> of static assets, ensures the app shell loads quickly, enhancing perceived performance.	<b>Caching Management:</b> Managing service worker lifecycles and application updates requires careful version control, making it complex than conventional web development.
<b>Concurrency and Synchronization</b>	The use of <b>React's component model</b> and state management facilitates efficient UI updates. Upon receiving real-time updates from Firebase, only the components displaying bus locations update, ensuring a smooth and non-blocking user experience.	<b>Initial Load Time:</b> The React bundle may take longer to load initially, potentially increasing first-load latency.

## 5. Detailed System Design

The following section presents all detailed system design diagrams and their corresponding descriptions.

### 5.1 Class Diagram Description

The system is structured around a central **User** class which is inherited by three specific actors: **Driver**, **Admin**, and **Passenger**. The operational logic involves **Bus**, **Route**, **Stop**, and **Notification** classes interacting to manage the transit network.

#### 1. User Management Classes

- **Class: User (Abstract Class)**
  - **Description:** The base class representing any authenticated entity in the system.
  - **Attributes:**
    - **userID:** String {PK}: Primary key uniquely identifying the user.

- **email:** String: User's contact email.
    - **passwordHash:** String: Secured storage of the user's password.
    - **fullName:** String: The full legal name of the user.
    - **createdAt:** DateTime: Timestamp of account creation.
  - **Methods:**
    - login(): Boolean: Authenticates the user into the system.
    - logout(): Void: Ends the current user session.
    - resetPassword(): Void: Initiates the password recovery process.
- **Class: Driver**
  - **Description:** Represents the bus operator responsible for trips.
  - **Attributes:**
    - driverID: String: Unique identifier for the driver.
    - licenseNumber: String: The driver's official license ID.
    - shiftStatus: Enum {ON\_DUTY, OFF\_DUTY}: Current working state.
    - currentBusID: String {FK}: Foreign key linking to the bus being driven.
  - **Methods:**
    - startTrip(routeID: String): Boolean: Begins a new journey on a specific route.
    - stopTrip(): Boolean: Ends the current journey.
    - broadcastGPS(lat: Float, lng: Float): Void: Transmits real-time location data.
- **Class: Admin**
  - **Description:** Represents the system manager.
  - **Attributes:**
    - adminID: String: Unique identifier for the administrator.
    - permissions: List<String>: A collection of access rights allowed for this admin.
  - **Methods:**
    - addBus(busDetails: Bus): Boolean: Registers a new bus in the fleet.
    - removeBus(busID: String): Boolean: Deletes a bus from the fleet.
    - updateRoute(routeDetails: Route): Void: Modifies existing route information.
    - viewSystemLogs(): List<Log>: Retrieves system activity records.
- **Class: Passenger**
  - **Description:** Represents the commuter using the app.
  - **Attributes:**
    - passengerID: String: Unique identifier for the passenger.
    - preferences: JSON: User-specific settings (e.g., saved routes).

- **Methods:**

- searchRoute(query: String): List<Route>: Finds routes matching user input.
- viewLiveMap(busID: String): MapView: Opens the tracking interface for a bus.
- getETA(stopID: String): Time: Calculates arrival time for a specific stop.

## 2. Operational Classes

- **Class: Bus**

- **Description:** Represents the physical vehicle in the fleet.

- **Attributes:**

- busID: String {PK}: Primary Key unique to the vehicle.
- plateNumber: String: The vehicle's license plate.
- capacity: Integer: Maximum number of passengers.
- status: Enum {ACTIVE, MAINTENANCE, IDLE}: Current operational condition.
- currentLat: Float: Current latitude coordinate.
- currentLng: Float: Current longitude coordinate.

- **Methods:**

- updateLocation(lat: Float, lng: Float): Void: Updates the bus's position in the system.
- assignDriver(driverID: String): Void: Links a driver to the bus.

- **Class: Route**

- **Description:** Defines the path and schedule a bus follows.

- **Attributes:**

- routeID: String {PK}: Primary Key for the route.
- routeName: String: Descriptive name of the route.
- totalDistance: Float: The full length of the route.
- googlePolyline: String: Encoded string for map visualization.

- **Methods:**

- addStop(stop: Stop): Void: links a new stop to the route.
- calculateTotalTime(): Time: Computes the estimated duration of the route.

- **Class: Stop**

- **Description:** Represents a physical location where the bus halts.

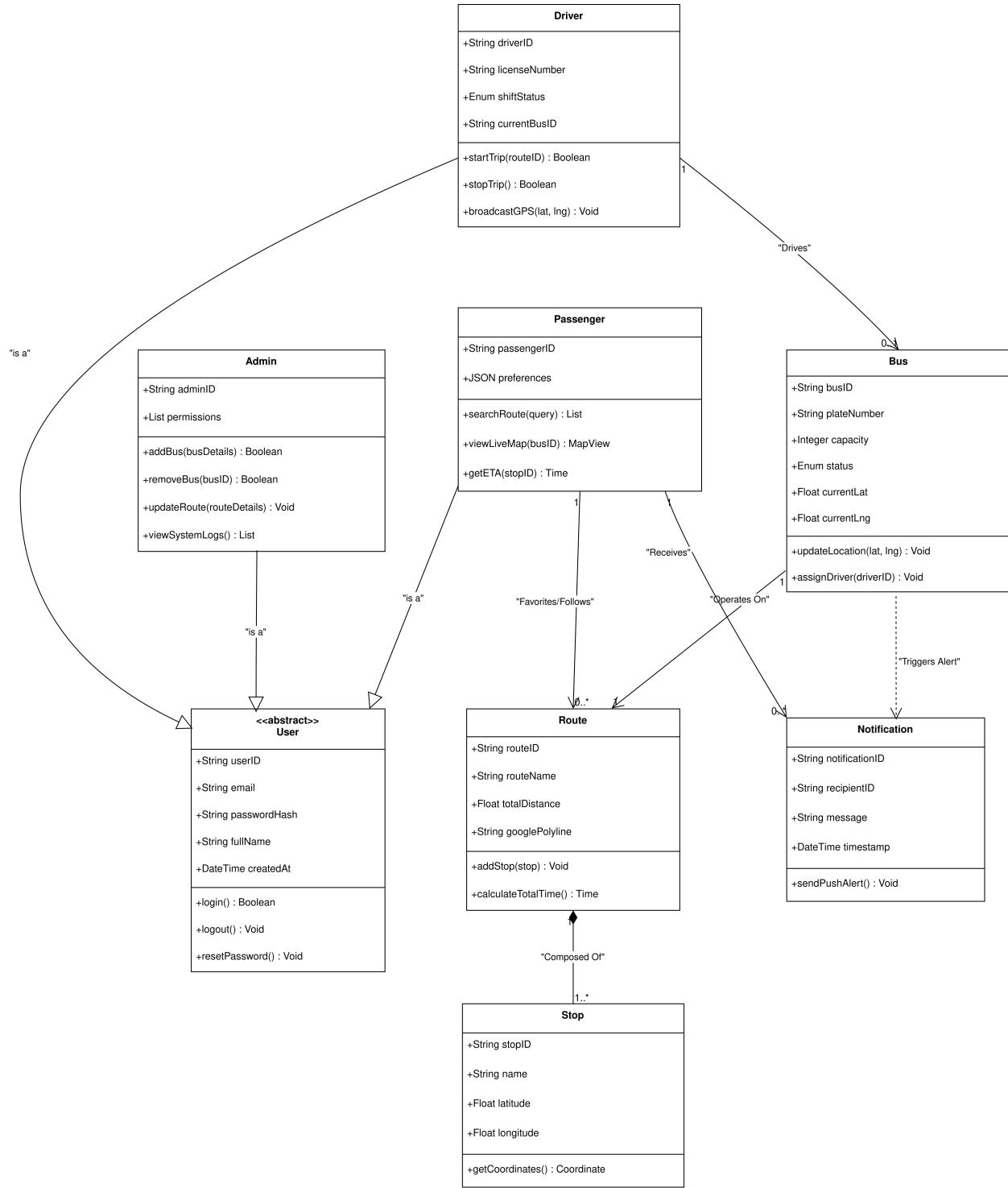
- **Attributes:**

- stopID: String {PK}: Primary Key for the stop.
- name: String: The human-readable name of the stop.

- latitude: Float: GPS Latitude.
- longitude: Float: GPS Longitude.
- **Methods:**
- getCoordinates(): Coordinate: Returns the location object of the stop.

### 3. Support Classes

- **Class: Notification**
  - **Description:** Manages alerts sent to users.
  - **Attributes:**
    - notificationID: String: Unique ID for the alert.
    - recipientID: String {FK}: Foreign Key linking to the receiving User.
    - message: String: Text content of the alert.
    - timestamp: DateTime: Time the alert was created.
  - **Methods:**
    - sendPushAlert(): Void: Dispatches the notification to the device.



**Figure 5.1:** UML Class Diagram depicting the inheritance hierarchy of User actors (Admin, Driver, Passenger) and their relationships with operational entities (Bus, Route, Stop)

## 5.2 Sequence Diagram

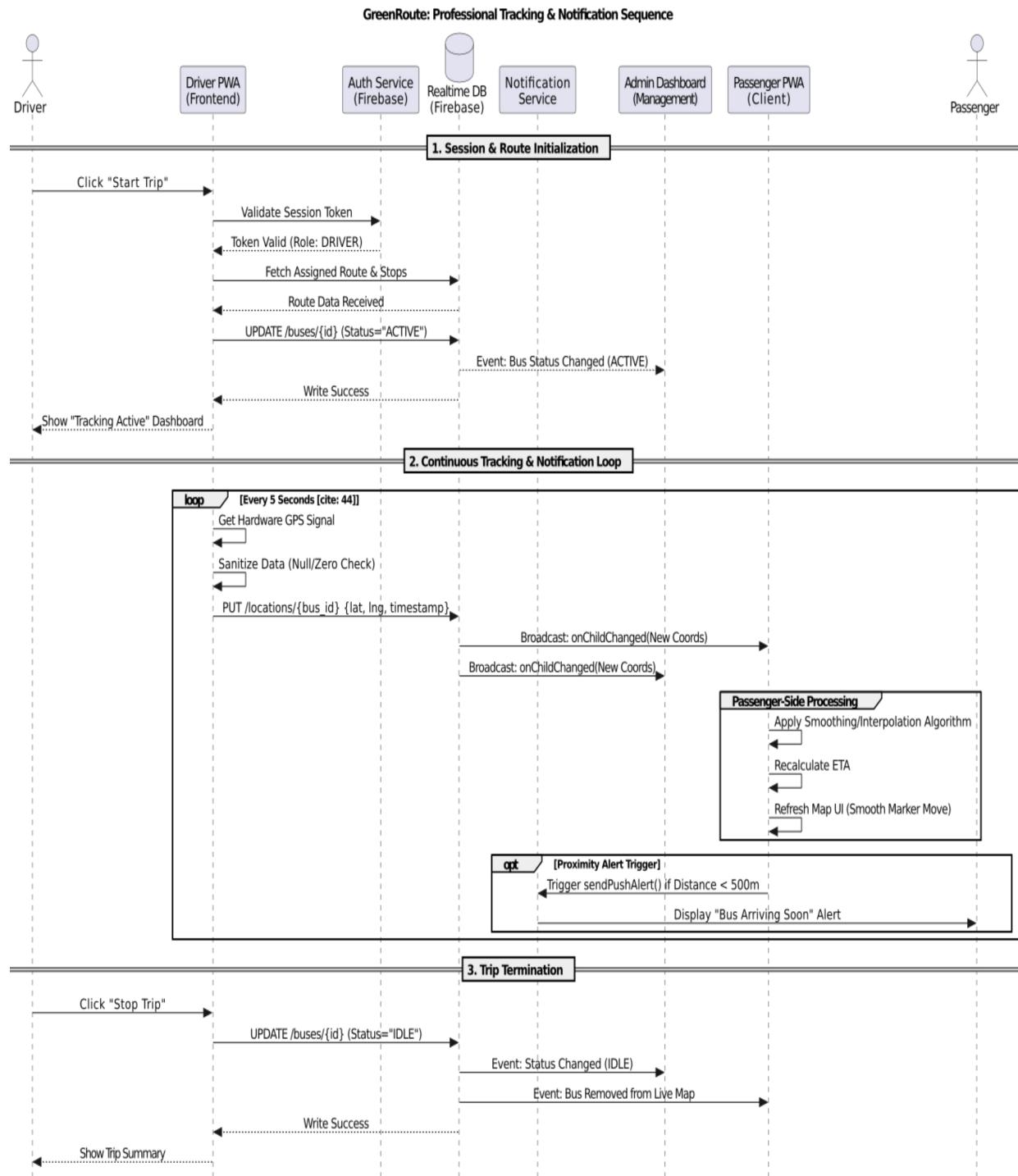
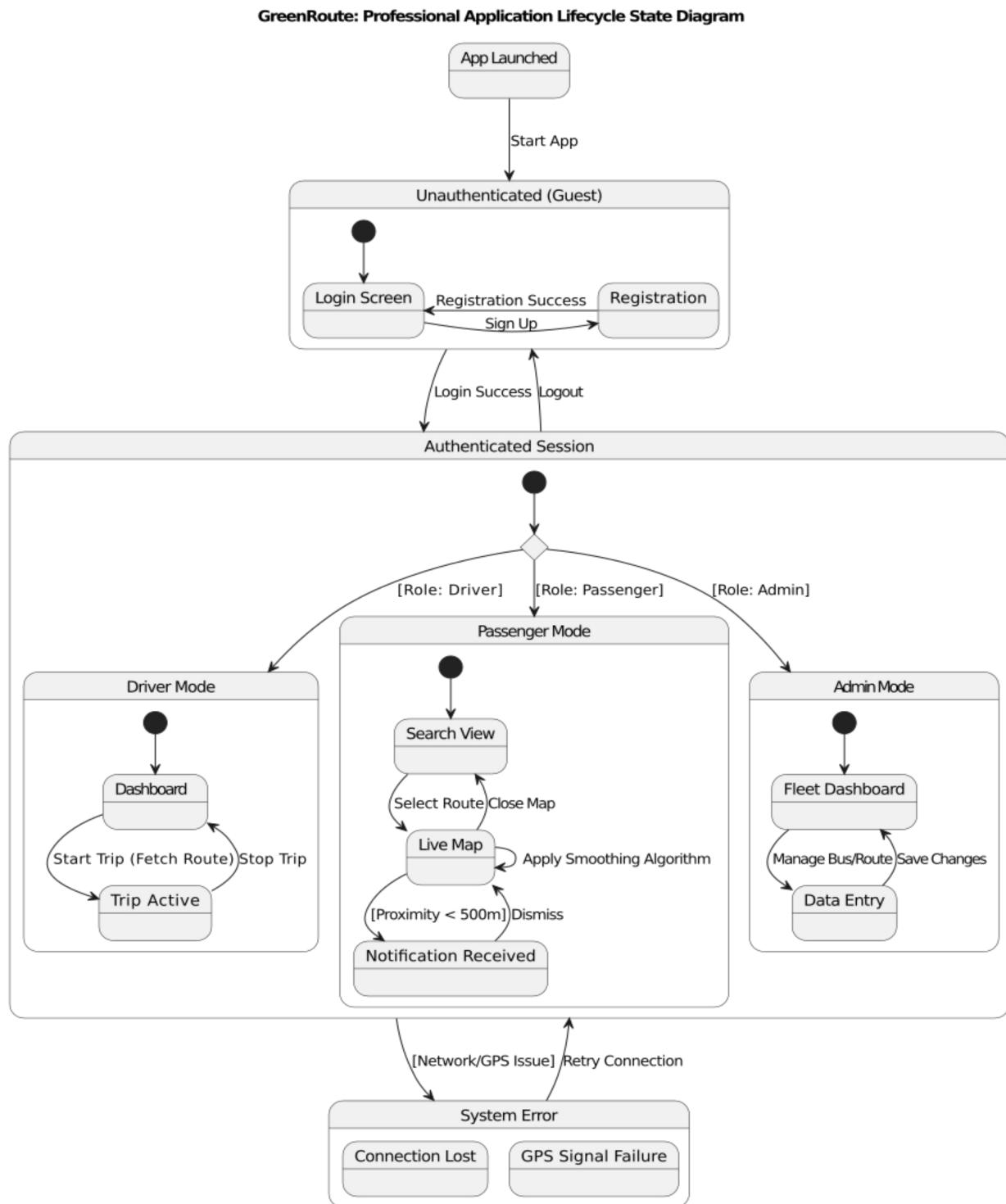


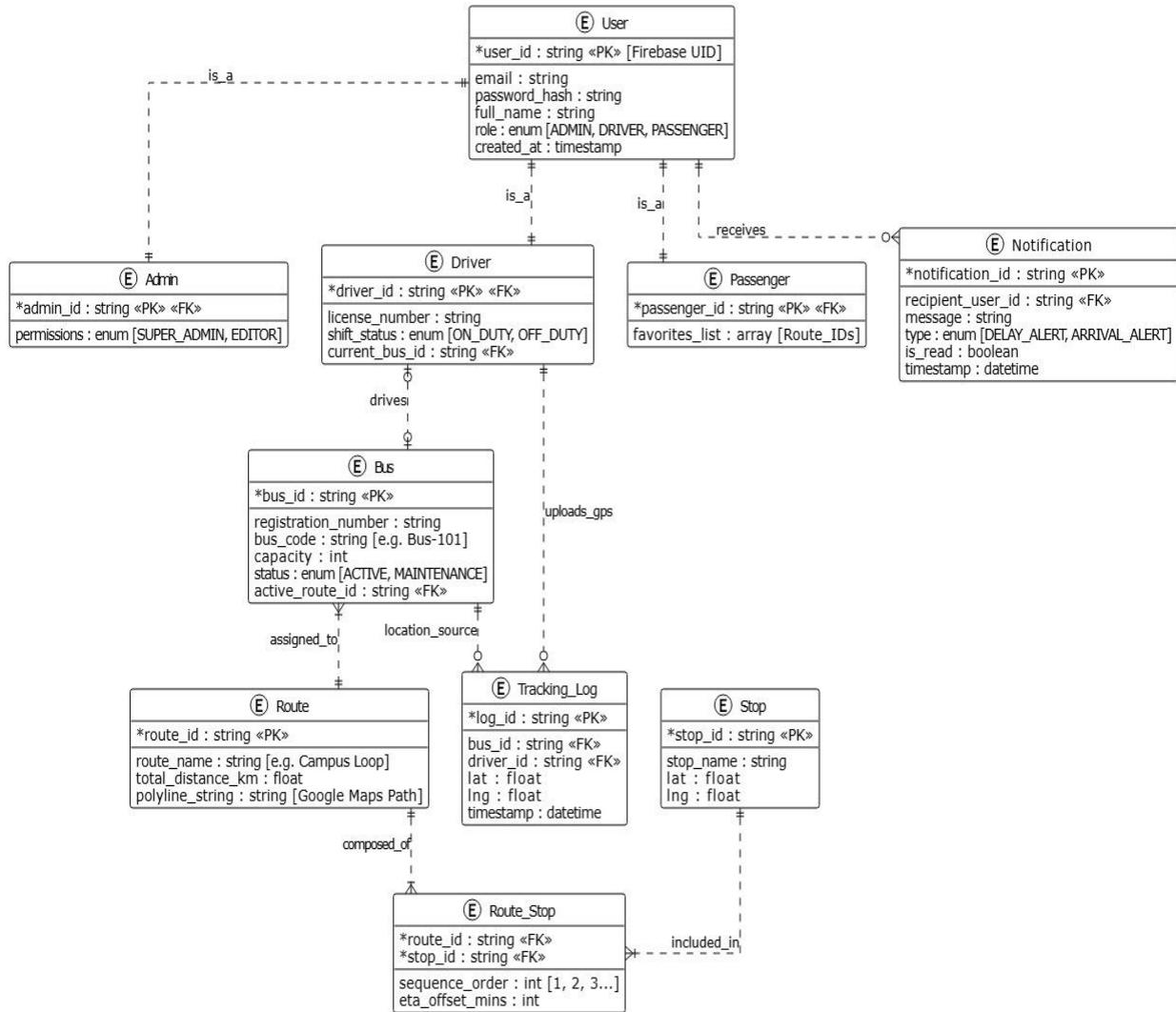
Figure 5.2: Detailed Sequence Diagram

## 5.3 State Transition Diagram



**Figure 5.3:** State Transition Diagram representing the lifecycle changes of a Bus entity and Driver Duty cycles

## 5.4 Logical data model (E/R model)



**Figure 5.4:** Entity-Relationship (ER) Diagram showing the logical associations and cardinality between Users, Buses, Routes, Stops, and Notifications.

## 5.5 Physical data models



**Figure 5.5: Physical Data Model illustrating the NoSQL schema implementation within the Firebase Realtime Database**

## 5.6 GUIs

Graphical User Interface (GUI) mockups displaying the Desktop Admin Dashboard for fleet management and the Mobile PWA interface for Passenger real-time tracking.

The image displays two screenshots of the GreenRoute mobile application interface. Both screenshots feature a dark theme with green accents.

**Top Screenshot (Passenger View):**

- Header: GreenRoute, Live Map, Routes, Schedules, Driver Login, Download App.
- Banner: New Fleet | 33 Buses Active.
- Title: Sargodha Moves with GreenRoute.
- Text: Real-time tracking for the University Road to Company Bagh route. Zero uncertainty.
- Button: Track My Bus Now →

**Bottom Screenshot (Admin/Driver View):**

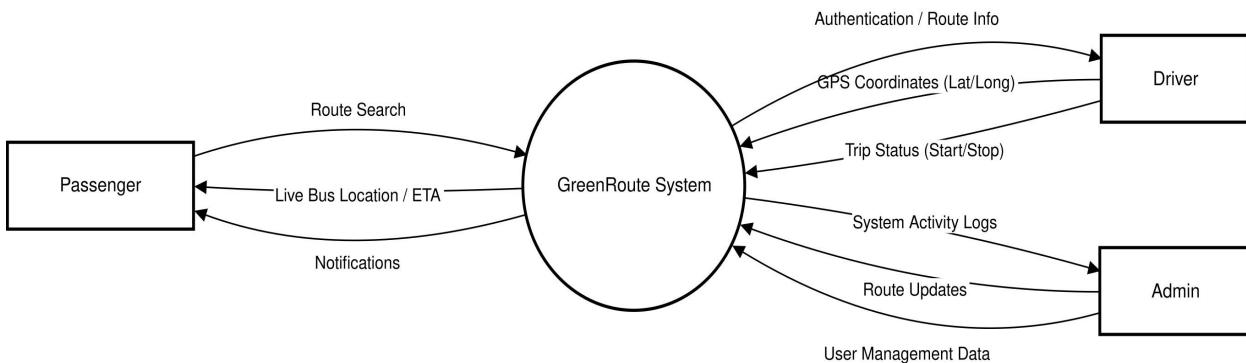
- Header: GreenRoute, Live Map, Routes, Schedules, Driver Login, Download App.
- Title: System Status.
- Text: Real-time metrics from Sargodha's transit network.
- Card 1: 33 Active Buses (with location icon).
- Card 2: Live Map Preview showing a route from University Rd to Company Bagh.
- Card 3: Zero Carbon Eco Transit (with bus icon).
- Card 4: 5 min Avg Wait Time (with clock icon).

The screenshot shows the 'Routes & Schedules' section of the GreenRoute app. At the top, there are navigation links for 'Live Map', 'Routes', and 'Schedules', along with 'Driver Login' and 'Download App' buttons. A notification bar indicates '3 Active Routes'. Below this, the title 'Routes & Schedules' is displayed in large, bold letters. A subtitle below it reads 'Plan your journey with our comprehensive route information, timetables, and fare details.' Underneath, a section titled 'Available Routes' lists three routes: Route A, Route B, and Route C. Each route card includes a small icon, the route name, the number of stops, the start and end locations, duration, frequency, and first bus time.

Route	Stops	From	To	Duration	Frequency	First Bus
Route A	12 stops	University Road	Company Bagh	45 min	Every 10 min	6:00 AM
Route B	8 stops	Railway Station	Satellite Town	30 min	Every 15 min	6:30 AM
Route C	10 stops	Faisal Colony	City Center	35 min	Every 12 min	6:00 AM

## 5.7 Data Flow Diagrams

### 5.7.1 DFD Level 0



**Figure 5.7.1:** Level 0 Data Flow Diagram (DFD) visualizing the high-level input and output flows between the GreenRoute system and external entities

## 6. References

Ref. No.	Document Title	Date of Release/ Publication	Document Source
1	Project Proposal	Oct 6, 2025	<a href="https://github.com/Asma-Khanum1722/FYP-Project.git">https://github.com/Asma-Khanum1722/FYP-Project.git</a>
2	Software Requirements Specification	Oct 20, 2025	<a href="https://github.com/Asma-Khanum1722/FYP-Project.git">https://github.com/Asma-Khanum1722/FYP-Project.git</a>