Bunab Software Requirements Specification for INSA

Emerging Technology (Jimma University)

Software Requirements Specification (SRS)

Project Name: BUNAB Ride Hailing Platform
Prepared For: INSA – Information Network Security Administration
Prepared By: Abdurahiman and Aboma Ride Service
Date: 24.04.2025
Version: 1.0

Standard SRS Document Outline for Ride-Hailing Platform

1. Introduction

1.1 Purpose
1.2 Scope
1.3 Definitions, Acronyms, and Abbreviations

2. Overall Description

2.1 Product Perspective
2.2 Product Functions
2.3 User Classes and Characteristics
2.4 Operating Environment
2.5 Design and Implementation Constraints
2.6 User Documentation
2.7 Assumptions and Dependencies

3. System Features

3.1 Feature Name

3.1.1 Description and Priority

3.1.2 Functional Requirements

3.1.3 Input/output Descriptions

3.1.4 Error Handling

4. External Interface Requirements

4.1 User Interfaces (Screenshots, mockups)

4.2 Hardware Interfaces

4.3 Software Interfaces (APIs, OS dependencies)

4.4 Communication Interfaces (e.g., HTTPS, SMS gateway)

6. System Models and Diagrams

6.1 Use Case Diagram

6.2 Activity Diagrams (e.g., Ride Booking Flow)

6.3 Data Flow Diagrams (DFD – Level 0 and Level 1)

1. Introduction

1.1 Purpose

This document specifies the requirements for the BUNAB Ride Hailing Platform, developed and registered in a trade name called Abdurahiman and Aboma Ride Service, to be certified by INSA. It defines the system's functionality, features, and constraints to ensure secure, scalable, and efficient transportation services in Ethiopia.

1.2 Scope

The platform includes three primary components:

- **Driver App** (Android)
- **Customer App** (Android)

- **Admin Panel** (Web-based)

It allows users to request rides, and drivers to accept and complete trips. Admins manage users, trips, complaints, and analytics.

1.3 Definitions, Acronyms, and Abbreviations

- **INSA:** Information Network Security Administration
- **BUNAB:** Brand name of the platform
- **SRS:** Software Requirements Specification
- **OTP:** One-Time Password
- **API:** Application Programming Interface

2. Overall Description

Great — here's a detailed write-up for each of the sub-sections under **Section 2: Overall Description** of your SRS document for the **BUNAB Ride Hailing Platform**, crafted in a professional tone for INSA certification.

2. Overall Description

2.1 Product Perspective

The BUNAB Ride Hailing Platform is an independent, full-stack application designed to facilitate transportation services through digital means. It consists of a **Customer App**, a **Driver App**, and an **Admin Dashboard**. The system is built using **Laravel** for the backend and **Flutter** for the mobile applications, following a service-oriented architecture (SOA). It communicates through RESTful APIs and is hosted on a secure server infrastructure. The platform is not a subsystem of any other system and is developed to cater specifically to urban and semi-urban transportation needs in Jimma, Ethiopia, with future scalability to include ride-sharing,.

2.2 Product Functions

The platform supports multiple core functionalities for various user types:

- **Customers** can register using their phone number, request rides by selecting pickup and drop-off locations, track rides in real-time, rate drivers, and make payments through cash.
- **Drivers** receive ride requests, accept or reject them, navigate to destinations using in-app map support, and receive payments directly in cash.
- **Administrators** can manage all platform data, verify drivers, monitor trip analytics, view logs, handle complaints, configure system settings, and export reports. Additional functions include notification systems, offline mode for drivers, fare estimation logic, and multilanguage support.

2.3 User Classes and Characteristics

The main user categories are as follows:

- **Customers**: General public users with basic smartphone literacy, primarily accessing the app via Android phones. They expect intuitive UI/UX and reliable GPS functionality.
- **Drivers**: Drivers registered with BUNAB, varying in digital literacy. The Driver App is designed with large buttons, simple navigation, and support for offline mode in areas with poor connectivity.
- **Administrators**: Staff with technical and operational roles. They use the web-based admin dashboard via desktop browsers and have access to sensitive data and system configurations. Role-based access control is implemented to distinguish between system admins and support staff.

2.4 Operating Environment

The BUNAB Ride Hailing Platform operates in the following environments:

- **Mobile Applications**: Android 8+ using Flutter.
- **Backend Server**: Laravel framework running on PHP 8+, hosted on a secure Linux-based server (Ubuntu 20.04 or higher), using Apache, and MySQL 8+ database.
- **Admin Dashboard**: Web application accessible via modern browsers (Chrome, Firefox, Edge).
- **Network**: Requires stable internet for real-time features.

2.5 Design and Implementation Constraints

- The platform must comply with **Ethiopian data protection and cybersecurity standards** as mandated by INSA.
- The system must ensure **end-to-end encryption** of sensitive data such as location and payment information.
- GPS functionality depends on third-party services (e.g., Google Maps API), which may incur usage fees and external data dependency.
- The app is primarily optimized for **low-end smartphones** commonly used in regional areas.
- Implementation must consider **multi-language support** (Amharic, Afaan Oromo).

2.6 User Documentation

The following documentation is or will be provided to support the platform:

- **User Guide** for customers (available in Amharic and English)

- **Driver Manual** detailing app usage and rules.
- **Admin Manual** explaining dashboard features, user management, and reports
- **System Maintenance Guide** for technical teams
- **API Documentation** for internal and potential third-party integrations All documents will be made available in PDF format and through a secured web portal for authorized users.

2.7 Assumptions and Dependencies

- The system assumes that all users have access to smartphones with basic GPS and internet capabilities.
- The ride matching and location services depend on third-party GPS and mapping APIs (Google Maps or similar).
- Legal and operational licenses must be maintained by BUNAB (under the trade license Abdurahiman and Aboma Ride Service) to continue platform operations.
- Regular updates will be required to maintain security, fix bugs, and introduce new features.

3. System Features

3.1 Ride Request and Matching

3.1.1 Description and Priority

This feature enables a customer to request a ride by selecting a pickup and drop-off location. The system identifies nearby drivers, calculates the fare, and dispatches the request to eligible drivers in real-time. Once accepted, both the driver and customer can track the trip.
Priority: High — This is a mission-critical feature.

3.1.2 Functional Requirements

- **FR-1**: The system shall allow users to input/select pickup and drop-off locations via GPS or manual input.
- **FR-2**: The system shall calculate estimated fare based on a distance-based algorithm.
- **FR-3**: The system shall dispatch ride requests to drivers within a configurable radius.
- **FR-4**: The system shall notify the customer when a driver accepts or declines the request.
- **FR-5**: The system shall allow cancellation from either party before the ride starts.
- **FR-6**: The system shall log all request and response data for future reference.

3.1.3 Input/output Descriptions

Input	Source	Description
Pickup Location	Customer	GPS or manual selection

Input	Source	Description
Drop-off Location	Customer	Destination address
Driver Availability	Driver App	Status and location of drivers
Ride Request	System	System-generated ride dispatch

Output	Destination	Description
Ride Confirmation	Customer App	Driver info and ETA
Ride Request	Driver App	Pickup and drop-off location
Status Updates	Both Apps	Real-time ride tracking and updates

3.1.4 Error Handling

- If no drivers are available, the system displays a retry message and logs the event.
- If the GPS is disabled or unavailable, the app prompts users to enable it.
- If a ride is not accepted within a timeout window, it is re-queued or canceled.
- If a cancellation occurs, the other party is immediately notified.

3.1.5 Non-Functional Requirements

- **NFR-1**: The ride request feature shall respond to customer actions (e.g., fare estimate, dispatch) within **3 seconds** under normal network conditions.
- **NFR-2**: The system shall maintain **99.9% uptime** for ride request and dispatch services.
- **NFR-3**: All trip-related data shall be stored securely and comply with **INSA cybersecurity guidelines**.
- **NFR-4**: Location services shall have an accuracy of at least **±10 meters** in urban areas.
- **NFR-5**: The system shall scale to support at least **10,000 concurrent ride requests**.
- **NFR-6**: Data related to ride requests shall be retained for a minimum of **one year** for audit purposes.
- **NFR-7**: The feature must support both **Amharic and English interfaces**, with fallback for offline mode when network access is lost.

3.2 User Registration and Login

3.2.1 Description and Priority

This feature allows both customers and drivers to create accounts and securely log into the system using phone numbers, OTPs, or passwords. It ensures that only verified users can access

ride-hailing features, ensuring a secure and accountable platform environment.
Priority: High — User authentication is essential for access control and data integrity.

3.2.2 Functional Requirements

- **FR-1**: The system shall allow new users to register using their phone number, name, and other required details.
- **FR-2**: The system shall send an OTP (One-Time Password) to the user's phone for verification during registration.
- **FR-3**: The system shall allow users to log in using their phone number and password or OTP.
- **FR-4**: The system shall securely store user credentials in hashed format.
- **FR-5**: The system shall differentiate between driver and customer roles during login and redirect them accordingly.
- **FR-6**: The system shall allow password reset functionality using SMS-based OTP.

3.2.3 Input/output Descriptions

Input	Source	Description
Phone Number	User	Used for account creation and login
OTP Code	SMS Gateway	Sent to user for verification
Name, Password	User	Account details provided during registration

Output	Destination	Description
OTP Message	User's phone	Used for identity verification
Login Token	Client App	Authentication token for session
Role-based Redirection	System	Takes users to driver or customer dashboard

3.2.4 Error Handling

- If an incorrect OTP is entered more than 3 times, the account will be temporarily locked.
- If the phone number is already registered, the system will prompt the user to log in.
- If the user enters an incorrect password, the system will allow 5 attempts before blocking the login temporarily.
- If the SMS gateway is down, the system will show an appropriate message and ask the user to try again later.

3.2.5 Non-Functional Requirements

- **NFR-1**: All passwords and sensitive data must be stored using **strong hashing algorithms (e.g., bcrypt)**.
- **NFR-2**: The system shall respond to login or registration requests within **2 seconds**.
- **NFR-3**: The platform shall ensure **multi-device login security**, invalidating previous sessions if a user logs in from a new device.
- **NFR-4**: OTPs shall be valid for only **5 minutes** and can only be used once.
- **NFR-5**: The platform shall log all authentication attempts for auditing and security analysis.
- **NFR-6**: The system must comply with **Ethiopian data protection laws** and **INSA digital identity guidelines**.
- **NFR-7**: The feature shall be available 24/7 with **at least 99.95% reliability**.

3.3 Trip History and Rating

3.3.1 Description and Priority

Allows users (drivers and customers) to view past trips and rate each other after completing a ride. This feature improves trust, transparency, and service quality.
Priority: Medium

3.3.2 Functional Requirements

- **FR-1**: Users shall be able to view a list of completed trips.
- **FR-2**: Each trip record shall show date, fare, origin, destination, and driver/customer details.
- **FR-3**: Users shall be prompted to rate and leave feedback after each ride.
- **FR-4**: Admins shall be able to access trip records and ratings.

3.3.3 Input/output Descriptions

Input	Source	Description
Trip Data	System	Automatically recorded after each ride

Input	Source	Description
Rating & Feedback	User	Optional comments and rating (1–5 stars)

Output	Destination	Description
Trip List	User App	User's ride history
Feedback Report	Admin Panel	Overview of ratings and comments

3.3.4 Error Handling

- If no trip history is available, the system shows a "No Trips Found" message.
- If feedback fails to submit, the system prompts a retry or logs it offline.

3.3.5 Non-Functional Requirements

- **NFR-1**: Trip records must be retained for **at least one year**.
- **NFR-2**: Feedback submission shall be processed within **1 second**.
- **NFR-3**: Rating data shall be encrypted and tamper-proof.
- **NFR-4**: The trip history interface must support **Amharic and English**.

3.4 Driver Wallet

3.4.1 Description and Priority

The Driver Wallet allows drivers to track their earnings, view transaction history, and monitor available balance and pending withdrawals.
Priority: High

3.4.2 Functional Requirements

- **FR-1**: The system shall update the driver's wallet balance after each completed trip.
- **FR-2**: Drivers shall be able to view daily, weekly, and monthly earnings.
- **FR-3**: The system shall record commission deductions automatically.
- **FR-4**: Drivers shall be able to request payouts to their linked bank or wallet accounts.

3.4.3 Input/output Descriptions

Input	Source	Description
Trip Fare	System	Auto-calculated earnings from rides
Withdrawal Request	Driver	Initiates fund transfer to bank/mobile money

Output	Destination	Description
Wallet Balance	Driver App	Real-time view of earnings and deductions
Transaction History	Driver App	Record of completed trips and payouts

3.4.4 Error Handling

- If payout fails, the system shall log the issue and notify the driver.
- If balance is insufficient, the system blocks the withdrawal request.

3.4.5 Non-Functional Requirements

- **NFR-1**: Wallet updates shall occur within **5 seconds** after trip completion and the driver makes the confirmation on payment Received button.
- **NFR-2**: All wallet transactions must be stored securely and traceable for **audit compliance**.
- **NFR-3**: The system shall handle up to **100 concurrent wallet requests**.
- **NFR-4**: Drivers shall have access to wallet data via driver home dashboard.

4. System Models and Diagrams

This section presents graphical representations of the system to provide a clear understanding of how various components interact and how data flows within the ride-hailing platform.

4.1 Use Case Diagram

The Use Case Diagram shows the interactions between the main system actors (Customer, Driver, Admin) and the key functions of the platform, such as ride requests, trip management, wallet updates, and admin monitoring.

Actors:

- Customer

- Driver
- Admin

Use Cases:

- Register/Login
- Request Ride
- Accept Ride
- Track Ride
- Rate Trip
- View Wallet
- Manage Users (Admin)
- Generate Reports (Admin)

Actors:

Customer

Driver

Admin

Use Cases:

Register/Login

Request Ride

Accept Ride

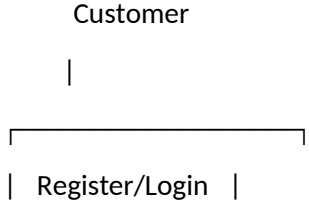
Track Ride

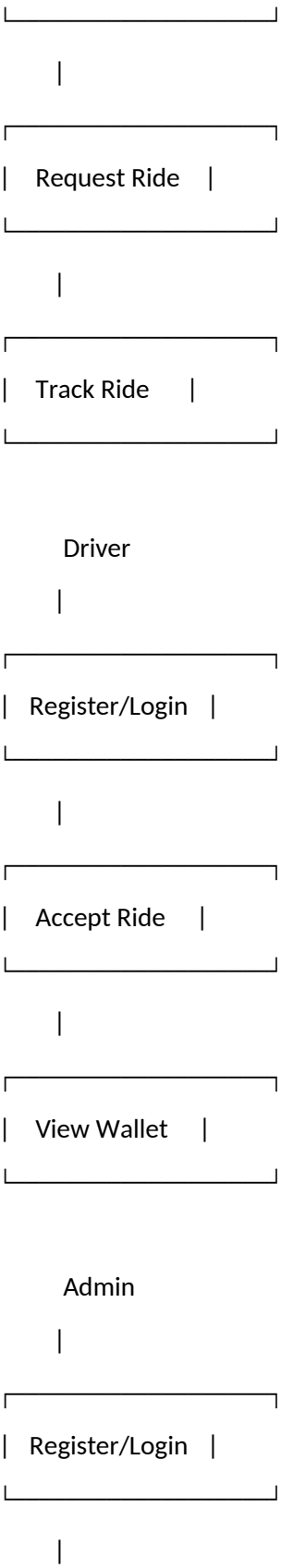
View Wallet (Driver)

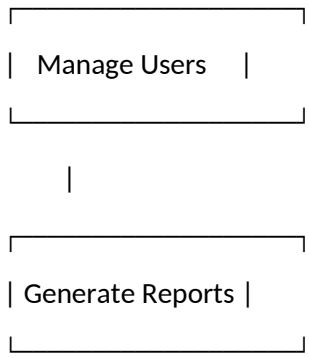
Manage Users (Admin)

Generate Reports (Admin)

Diagram:







4.2 Activity Diagram – Ride Booking Flow

This diagram outlines the sequence of actions in the ride-booking process, from the moment a customer initiates a ride to trip completion and payment.

Steps:

1. Customer logs in
2. Enters pickup and drop-off location
3. System matches with nearby driver
4. Driver accepts the ride
5. Trip starts
6. Trip ends
7. Payment is processed
8. Customer/Driver rate each other

Start

↓

Customer opens App

↓

Enter Pickup and Drop-off Locations

↓

System searches for nearby Drivers

↓

Driver receives Ride Request

↓

Driver Accepts or Rejects

(Reject) *(Accept)*

Search for another Driver *Driver Assigned*

↓ ↓

Ride Starts

↓

Ride Ends

↓

Payment Processed

↓

Customer and Driver Rate Each Other

↓

End

4.3 Data Flow Diagrams

4.3.1 Level 0 DFD (Context Level)

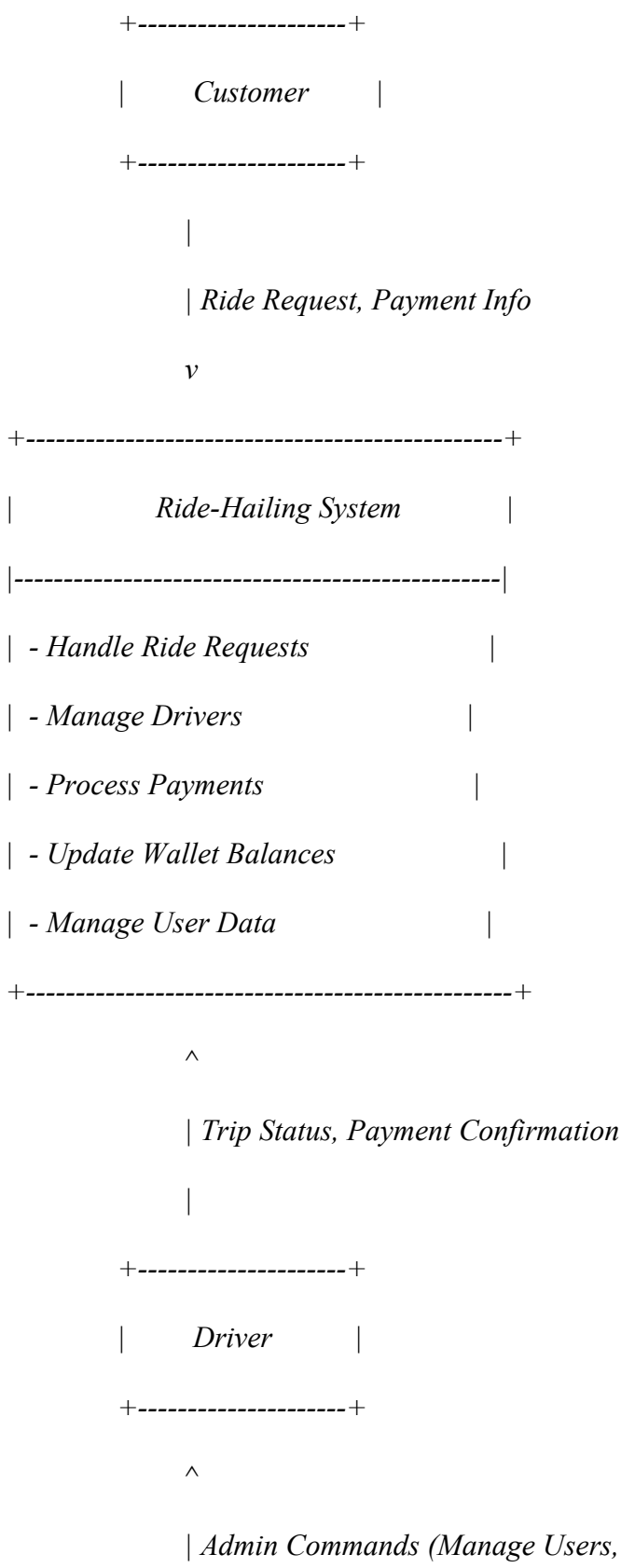
Shows the overall interaction between external entities (Users and Admin) and the system as a single process.

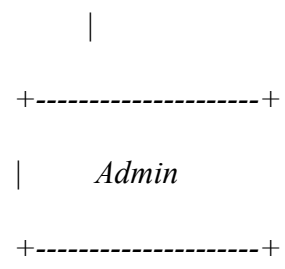
Entities:

- Customer
- Driver
- Admin

Processes:

- Ride Request Processing
- Payment Handling
- User Management

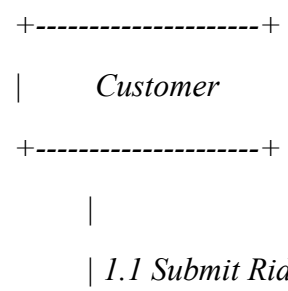


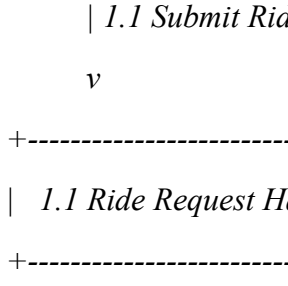


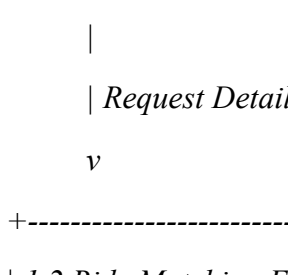
4.3.2 Level 1 DFD

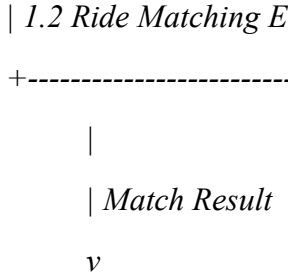
Expands the main processes into sub-processes like:

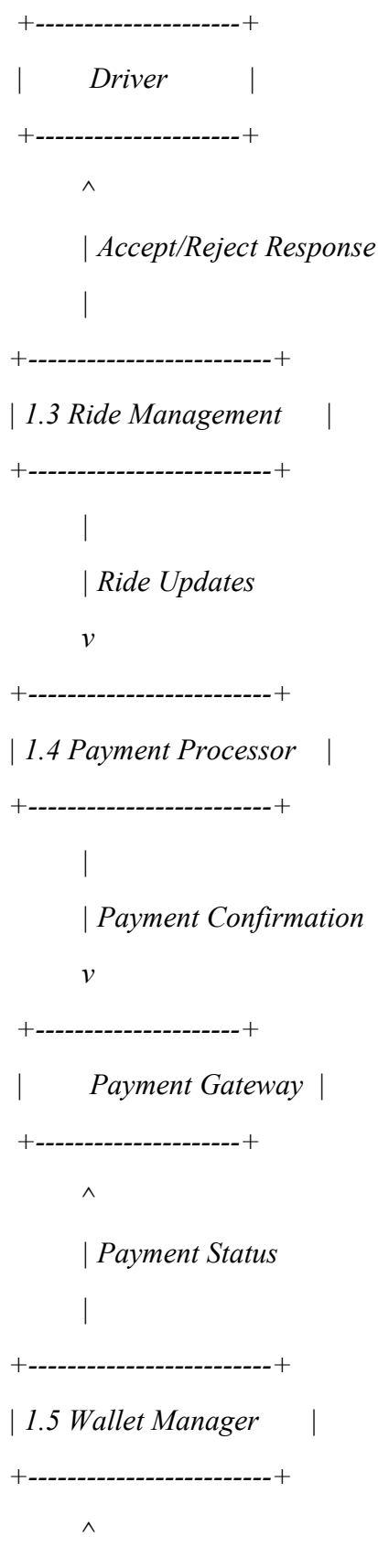
- Registration/Login
- Trip Matching Engine
- Wallet Processing
- Feedback System











| *Wallet Update*

|

+--------------------+

| *Admin* |

+--------------------+

|

| *1.6 Admin Dashboard (Reports, Manage Users)*

4.4 Entity Relationship Diagram (ERD)

The ERD visualizes the database structure, showing the relationships between key data entities like Users, Rides, Vehicles, Wallet Transactions, and Feedback.

Entities:

- User (Customer, Driver)
- Ride
- Vehicle
- Payment
- Wallet
- Rating

Relationships:

- One User can have many Rides
- Each Ride links one Customer and one Driver
- One Driver has one Vehicle
- One Wallet per User

[Customer]

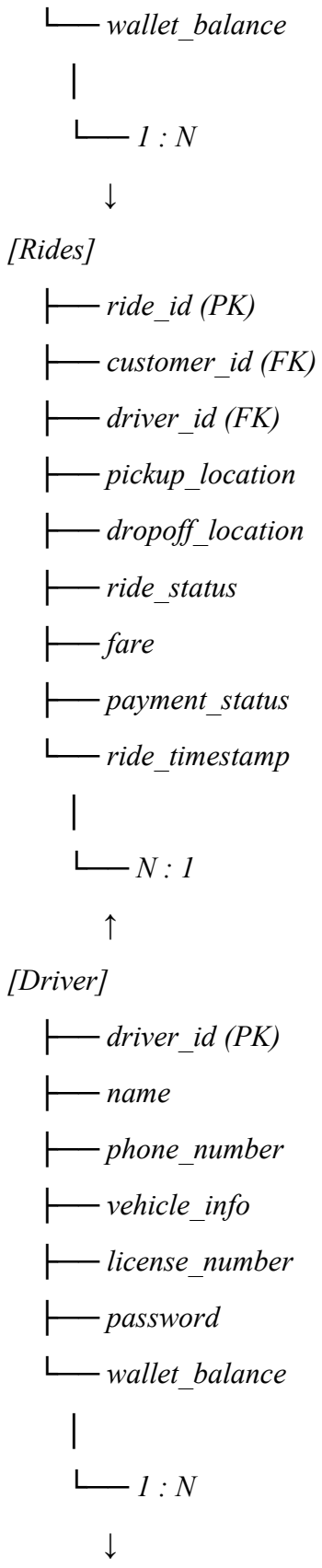
├── *customer_id (PK)*

├── *name*

├── *phone_number*

├── *email*

├── *password*



[Wallet_Transactions]

- ├── *transaction_id (PK)*
- ├── *driver_id (FK)*
- ├── *amount*
- ├── *transaction_type (credit/debit)*
- ├── *transaction_date*

[Admin]

- ├── *admin_id (PK)*
- ├── *username*
- ├── *password*

4.5 System Architecture Diagram (Client-Server Model)

Illustrates the system's technical architecture, highlighting the flow between mobile apps, backend server, database, and admin dashboard.

Components:

- Client Apps (Driver App, Customer App – Android/iOS)
- Backend Server (Laravel/PHP)
- Database (MySQL/PostgreSQL)
- Admin Panel (Web-Based)
- Notification Server (FCM or SMS Gateway)

(Diagram to be inserted)

4. Appendices

A. Trade License (Copy Attached)

B. Taxpayer Registration Certificate

C. Company Profile

D. Screenshots of Application Interfaces

E. Sample API Endpoints

F. Source Code Repository Summary (if required by INSA)