Namal University, Mianwali

Department of Computer Science
Software Engineering

Software Requirements Specification

for

FareShare

A Ride-Sharing System

Team FareShare:

Name	Roll Number	Role
Muhammad Naveed	NUM-BSCS-2024-54	Group Lead
Munawar Ali	NUM-BSCS-2024-60	Group Member
Areeba Tahir	NUM-BSCS-2024-15	Group Member

Instructor: Asiya Batool
Requirement Provider: Rana Muhammad Adeel
Submission Date: December 28, 2025

Contents

List of Tables

List of Figures

1 Introduction

1.1 Purpose

The purpose of this Software Requirements Specification (SRS) is to provide a detailed description of the **FareShare** ride-sharing system. This document is intended to guide the development team, project managers, and quality assurance personnel through the software development lifecycle. It details the functional and non-functional requirements, interfaces, and system constraints.

This document adheres to the **IEEE Std 830-1984** guidelines for Software Requirements Specifications. It defines the complete behavior of the FareShare system, ensuring that the software delivered satisfies the needs of the stakeholders, particularly focusing on a **cash-based economy** operational model where no digital payment gateways are integrated.

1.2 Scope

FareShare is a mobile-based application ecosystem designed to facilitate transportation services in Pakistan. The system connects riders seeking transportation with nearby drivers willing to provide rides.

The **FareShare System** will:

- Provide separate Android/iOS applications for Riders and Drivers.
- Utilize GPS services for real-time location tracking and navigation.
- Implement an intelligent matching algorithm to pair riders with the nearest available drivers.
- Calculate fares dynamically based on distance, time, and vehicle category.
- Operate strictly on a **Cash-Only** basis for fare collection.
- Include safety features such as SOS alerts and ride monitoring.
- Provide an Administrative Web Panel for user management and document verification.

The system explicitly **excludes**:

- Integration with credit card processors (Stripe/Visa/Mastercard).
- In-app digital wallets for storing monetary value.
- Food delivery or courier services.

1.3 Definitions, Acronyms, and Abbreviations

Term	**Definition**
SRS	Software Requirements Specification.
GPS	Global Positioning System, used for geolocation.
ETA	Estimated Time of Arrival.
OTP	One-Time Password, used for phone verification.
API	Application Programming Interface.
Rider	The end-user requesting transportation.
Driver	The verified service provider operating a vehicle.
Admin	System administrator managing the backend operations.
CNIC	Computerized National Identity Card (Pakistan).
Cash-Only	A transaction model where payment is physically handed from Rider to Driver.

1.4 Overview

The remainder of this document is organized as follows:

- **Section 2** describes the general factors that affect the product and its requirements, including user characteristics and constraints.
- **Section 3** provides the specific functional, non-functional, and interface requirements in detail.
- **Section 4** contains the Appendices, including the Context Diagram and Use Case Diagram.

2 General Description

2.1 Product Perspective

FareShare is a standalone software system that interacts with several external entities. It operates within a client-server architecture:

- **Mobile Clients:** The Rider and Driver applications serve as the front-end interfaces.
- **Backend Server:** Hosted on a cloud infrastructure (e.g., AWS/DigitalOcean), handling business logic, database operations, and API requests.
- **External APIs:**
  - **Google Maps API:** For map rendering, geocoding, and route calculation.
  - **SMS Gateway:** For sending OTPs and emergency alerts.
  - **Firebase (FCM):** For push notifications.

The system functionality relies heavily on the availability and accuracy of these external interfaces.

2.2 Product Functions

The major functions of the system are grouped by module:

1. Account Management

- Registration via Phone Number.
- Profile Management (Photo, Name, Email).
- Driver Document Upload (License, CNIC, Registration).

2. Ride Operations

- Real-time Driver Availability toggle.
- Ride Booking (Pickup/Drop-off selection).
- Fare Estimation (Cash basis).
- Driver Matching Algorithm.
- Ride Cancellation logic.

3. Navigation & Tracking

- Live GPS tracking of the driver.
- Turn-by-turn navigation for drivers.
- ETA calculation.

4. **Post-Ride**

- Final Cash Fare Calculation.
- Rating and Review System.
- Ride History Log.

2.3 User Characteristics

- **Riders:** General public, students, and professionals. Technical literacy ranges from low to high. They prioritize ease of use, safety, and transparent pricing.
- **Drivers:** Vehicle owners seeking income. Educational backgrounds vary significantly. The Driver App interface must be highly visual, simple, and require minimal interaction while driving.
- **Administrators:** Technical staff responsible for vetting drivers and managing system health. High technical literacy is assumed.

2.4 General Constraints

- **Hardware:** The application requires smartphones with functional GPS, Camera, and Internet connectivity (4G/Wi-Fi).
- **Reliability:** The system must handle intermittent network connectivity gracefully (e.g., store ride data locally until reconnected).
- **Payment:** The system is constrained to **Cash-Only** transactions. No digital funds transfer is permitted within the app.
- **Geography:** Initially limited to the Mianwali region.

2.5 Assumptions and Dependencies

- It is assumed that all drivers possess valid legal documents.
- It is assumed that Google Maps API services will remain operational.
- It is assumed that users will grant location permissions to the application.

3 Specific Requirements

3.1 Functional Requirements

This section details the functional requirements of the system. Each requirement is defined with inputs, processing logic, and outputs.

3.1.1 Module: Authentication & Onboarding

Requirement ID	**FR-AUTH-01**
Feature Name	User Registration (Rider/Driver)
Description	The system shall allow new users to register an account using their mobile phone number.
Inputs	Mobile Number, Role Selection (Rider or Driver).
Processing Logic	1. Validate mobile number format (Pakistan format +92). 2. Check database for existing accounts. 3. Generate a random 6-digit OTP. 4. Invoke SMS Gateway API to send OTP.
Outputs	OTP Sent Screen; Database record created with status 'Unverified'.
Error Handling	If number exists: Prompt user to Login. If SMS fails: Display "Network Error".

Requirement ID	**FR-AUTH-02**
Feature Name	OTP Verification
Description	The system shall verify the user's mobile number via OTP.
Inputs	User entered 6-digit code.

Processing Logic	1. Retrieve stored OTP for the session. 2. Compare entered code with stored code. 3. Check if OTP is expired ($>$ 2 minutes). 4. If valid, generate JWT Access Token.
Outputs	User Dashboard (Rider or Driver Home). Session active.
Error Handling	Invalid OTP: Display ”Incorrect Code”. Expired OTP: Display ”Code Expired”.

Requirement ID	**FR-AUTH-03**
Feature Name	Driver Document Submission
Description	The system shall require drivers to upload legal documents before going online.
Inputs	Images of: CNIC (Front), CNIC (Back), Driving License, Vehicle Registration.
Processing Logic	1. Validate image format (JPG/PNG). 2. Upload images to secure cloud storage bucket. 3. Update Driver Profile status to ”Pending Approval”. 4. Create a verification ticket for Admin.
Outputs	Success Message: ”Documents Submitted for Review”.
Error Handling	File too large (>5MB); Upload timeout.

3.1.2 Module: Ride Booking & Matching

Requirement ID	**FR-BOOK-01**

Feature Name	Location Selection
Description	The Rider shall be able to set Pickup and Drop-off locations.
Inputs	Map Pin interaction or Text Search Address.
Processing Logic	1. If Text Search: Call Google Places API for suggestions. 2. If Pin Drop: Call Geocoding API to get address from Lat/Long. 3. Validate that location is within service area.
Outputs	Updated Map View with Route Line and Markers.
Error Handling	Service Area Error: ”FareShare is not available in this area”.

Requirement ID	**FR-BOOK-02**
Feature Name	Fare Estimation (Cash)
Description	The system shall estimate the cash fare before the ride is booked.
Inputs	Pickup Coords, Drop-off Coords, Selected Vehicle Type.
Processing Logic	1. Calculate Distance (km) and Time (min) via Directions API. 2. Retrieve Base Rate, Per Km Rate, Per Min Rate for vehicle type. 3. Compute: $Fare = Base + (Km \times Rate) + (Time \times Rate)$.
Outputs	Estimated Fare Range (e.g., ”PKR 200 - 230”).
Error Handling	Route Calculation Error.

Requirement ID	**FR-BOOK-03**

Feature Name	Ride Request Broadcasting
Description	The system shall find the nearest available drivers.
Inputs	Rider Location, Vehicle Type.
Processing Logic	1. Query Geospatial Database for drivers with status "Online" within 3km. 2. Filter by Vehicle Type. 3. Sort by Distance (Ascending). 4. Send FCM Push Notification to the top candidate.
Outputs	Notification on Driver App: "New Ride Request".
Error Handling	No Drivers Found: Display "No drivers available, try again later".

Requirement ID	**FR-BOOK-04**
Feature Name	Ride Acceptance
Description	The Driver shall accept a ride request to initiate the trip.
Inputs	Accept Button Press.
Processing Logic	1. Check if ride is still pending (not taken by another driver). 2. Assign Driver ID to Ride Record. 3. Change Ride Status to "Accepted". 4. Send Confirmation to Rider.
Outputs	Navigation Screen loaded for Driver. Driver Profile loaded for Rider.
Error Handling	Ride Expired/Taken: Display "Ride no longer available".

3.1.3 Module: Ride Execution Tracking

Requirement ID	**FR-EXEC-01**
Feature Name	Real-Time Tracking
Description	The Rider shall view the Driver's location in real-time.
Inputs	Driver GPS Stream (Lat, Long, Heading, Speed).
Processing Logic	1. Driver App emits location event via WebSocket every 3 seconds. 2. Server relays data to the specific Rider's socket room. 3. Rider App animates the car marker on the map.
Outputs	Smooth movement of vehicle icon on Rider Map.
Error Handling	Connection Lost: ”Reconnecting...” spinner.

Requirement ID	**FR-EXEC-02**
Feature Name	Start Ride
Description	The Driver shall mark the ride as started upon picking up the rider.
Inputs	Swipe ”Start Ride”.
Processing Logic	1. Validate Driver distance to Pickup (Geofence check). 2. Record Start Time and Start Coordinates. 3. Change Status to ”In Progress”.
Outputs	UI update: ”Ride in Progress”.
Error Handling	Driver too far from pickup: ”You must be at pickup location”.

Requirement ID	**FR-EXEC-03**
Feature Name Final Calculation	End Ride
Description	The system shall calculate the final cash fare upon completion.
Inputs	Swipe ”End Ride”, GPS Logs.
Processing Logic	1. Calculate actual distance traveled and duration. 2. Apply pricing formula. 3. Generate Final Cash Amount.
Outputs	Rider Screen: ”Please Pay PKR [Amount]”. Driver Screen: ”Collect PKR [Amount]”.
Error Handling	GPS Drift: System uses estimated distance if actual GPS logs are corrupted.

Requirement ID	**FR-EXEC-04**
Feature Name	Cash Collection Confirmation
Description	The Driver must confirm they received the cash payment.
Inputs	Button ”Cash Received”.
Processing Logic	1. Update Ride Status to ”Completed”. 2. Create Transaction Record (Cash). 3. Release Driver to ”Available” pool.
Outputs Rating Screen.	Ride Summary
Error Handling	N/A.

3.1.4 Module: Safety Emergency

Requirement ID	**FR-SAFE-01**
Feature Name	SOS Alert
Description	Users can trigger an emergency alert during a ride.
Inputs	SOS Button Press.
Processing Logic	1. Capture current GPS location. 2. Construct SMS with ”Help! I am in danger. Track me here: [Link]”. 3. Send to pre-configured emergency contacts.
Outputs	SMS Sent Confirmation. Backend Admin Alert.
Error Handling	SMS Gateway Failure.

3.2 External Interface Requirements

3.2.1 User Interfaces

The system will provide a graphical user interface (GUI) compatible with Android and iOS touchscreens.

Rider App Screens:

1. **Home Map:** Full-screen map with a search bar at the top and a ”Current Location” FAB.
2. **Vehicle Selection:** A bottom sheet displaying vehicle types (Bike, Rickshaw, Mini, Sedan) with icons and estimated prices.
3. **Ride Status Panel:** A card overlay showing Driver Name, Photo, Vehicle Plate, and Color. Includes ”Call” and ”Message” buttons.
4. **Payment Screen:** A full-screen prompt displaying the large Cash Amount to be paid.

Driver App Screens:

1. **Online/Offline Switch:** A prominent toggle to set availability.
2. **Request Modal:** A time-sensitive pop-up (15 seconds) showing Pickup Address, Distance, and Estimated Earning.
3. **Navigation Mode:** A 3D map view utilizing the Google Navigation SDK.
4. **Collection Screen:** A high-contrast screen showing the amount of cash to collect.

3.2.2 Hardware Interfaces

- **GPS Module:** The system requires access to the device’s Location Services (Fine Location) for tracking.
- **Camera:** Access is required for capturing profile photos and scanning driver documents.
- **Network Radio:** The system uses the 4G/LTE/Wi-Fi radio for API communication.

3.2.3 Software Interfaces

- **Operating System:** Android 8.0+ / iOS 14.0+.
- **Database:** MongoDB (NoSQL) for storing user profiles and ride logs.
- **Maps API:** Google Maps Platform for geocoding and rendering.
- **Push Notification Service:** Firebase Cloud Messaging (FCM).

3.2.4 Communication Interfaces

- **REST API:** Client-server communication over HTTPS using JSON format.
- **WebSockets:** Socket.io protocol for real-time bi-directional event streaming (Location updates, Status changes).

3.3 Non-Functional Requirements

3.3.1 Performance Requirements

1. **Latency:** The Ride Matching algorithm must return a driver match within 10 seconds under normal load.
2. **Throughput:** The backend system must support at least 500 concurrent ride requests per minute.
3. **App Start Time:** The mobile application must launch and render the map within 4 seconds.
4. **Tracking Update Rate:** Driver location must update on the Rider's screen at least every 5 seconds.

3.3.2 Safety Requirements

1. **Data Backup:** User data and ride logs must be backed up incrementally every 24 hours.
2. **Driver Vetting:** The system must prevent drivers from accepting rides until their documents have been manually verified by an Administrator.

3.3.3 Security Requirements

1. **Encryption:** All data in transit must be encrypted using TLS 1.2 or higher.
2. **Password Storage:** Passwords (if used) must be hashed using Bcrypt.
3. **Privacy:** User phone numbers should be masked when making calls through the app (if VoIP is implemented) or strictly controlled.
4. **Authorization:** API endpoints must be protected using JWT (JSON Web Tokens) to ensure only authenticated users can access data.

3.3.4 Software Quality Attributes

- **Reliability:** The system availability target is 99.9% uptime during business hours.
- **Usability:** The UI must be intuitive enough for a user with basic smartphone literacy to book a ride within 3 clicks.
- **Maintainability:** The code must adhere to modular architecture standards to allow for easy updates and feature additions.

4 Appendices

4.1 Appendix A: Context Diagram

The following Context Diagram illustrates the boundary of the FareShare system and its interaction with external entities (Rider, Driver, Admin, Payment System, Maps API).

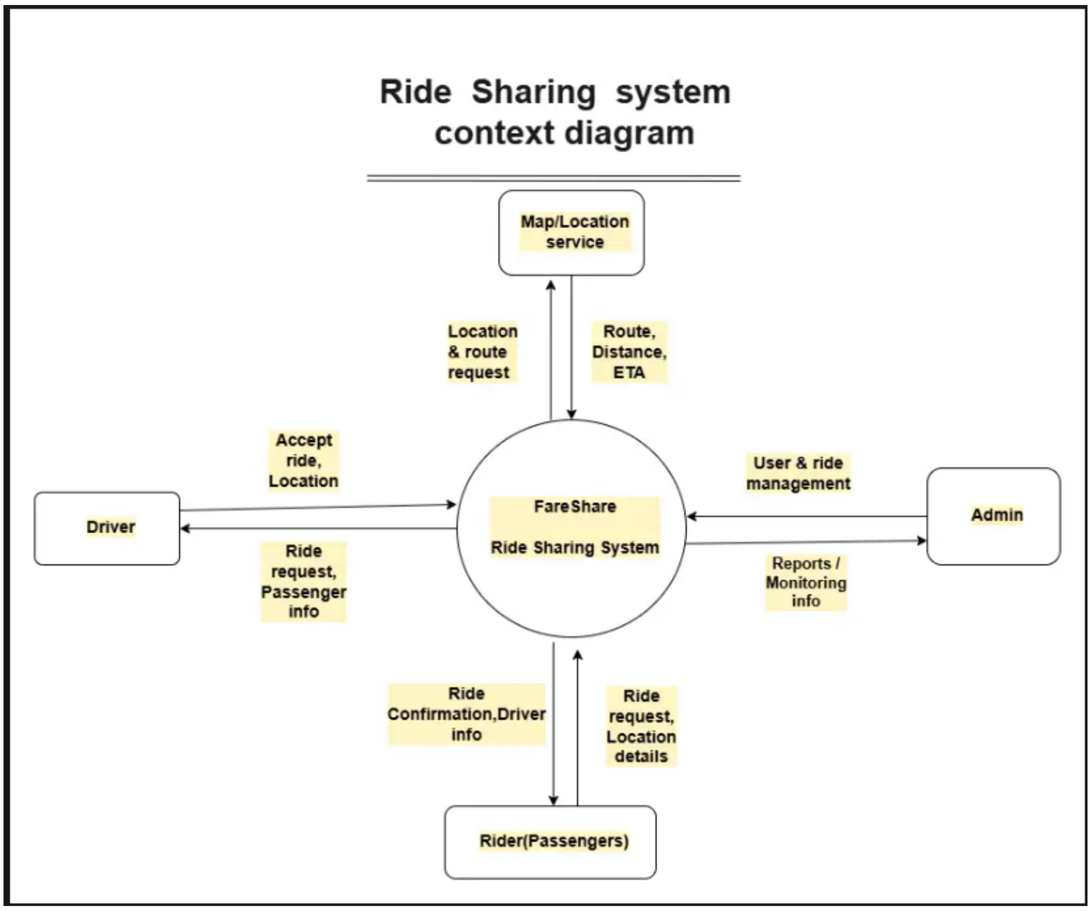
Ride Sharing system context diagram
Map/Location service
Location & route request
Route, Distance, ETA
Accept ride, Location
Driver
Ride request, Passenger info
FareShare Ride Sharing System
User & ride management
Admin
Reports / Monitoring info
Ride Confirmation,Driver info
Ride request, Location details
Rider(Passengers)

Figure 1: Context Diagram of FareShare System

4.2 Appendix B: Use Case Diagram

The Use Case Diagram depicts the primary actors and their interactions with the system's use cases.

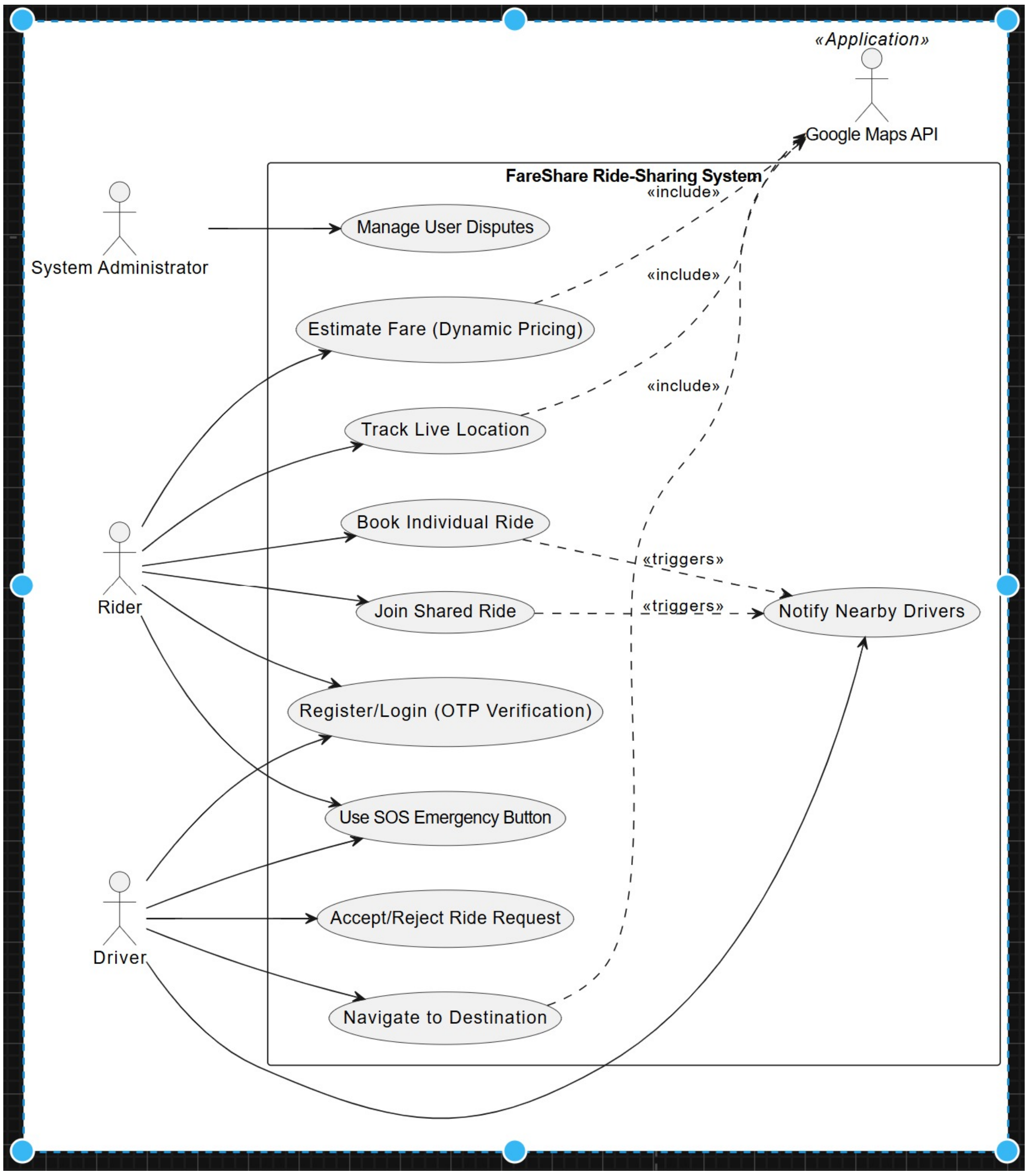

Figure 2: Use Case Diagram of FareShare System

4.3 Appendix C: Detailed Use Case Narratives

4.3.1 Use Case 1: Book a Ride

Use Case ID	UC-01
Use Case Name	Book a Ride
Primary Actor	Rider
Pre-conditions	Rider is logged in; GPS is enabled; Internet is active.
Main Flow	1. Rider opens the application. 2. System displays current location on map. 3. Rider enters destination ”Namal College”. 4. System calculates route and displays ”Estimated Cash Fare: PKR 300”. 5. Rider selects vehicle type ”Car”. 6. Rider clicks ”Book Now”. 7. System searches for nearby drivers. 8. System matches a driver and displays Driver Profile.
Alternative Flow	7a. No drivers are found. 7b. System displays ”No drivers available”. 7c. Rider retries or cancels.
Post-conditions	Ride status is ”Accepted”. Driver is en route.

4.3.2 Use Case 2: Accept a Ride

Use Case ID	UC-02
Use Case Name	Accept Ride Request
Primary Actor	Driver
Pre-conditions	Driver is Online; Driver status is Available.
Main Flow	1. System sends ride request notification to Driver. 2. Driver views Pickup Location and Estimated Fare. 3. Driver taps ”Accept” within 15 seconds. 4. System assigns the ride to the Driver. 5. System launches navigation to pickup point.
Alternative Flow	3a. Driver ignores request or taps ”Decline”. 3b. System forwards request to the next available driver.
Post-conditions	Driver status changes to ”Busy”.

4.3.3 Use Case 3: Complete Ride & Collect Cash

Use Case ID	UC-03
Use Case Name	Complete Ride
Primary Actor	Driver
Pre-conditions	Ride is ”In Progress”.
Main Flow	1. Driver arrives at destination. 2. Driver swipes ”End Ride”. 3. System calculates final fare based on GPS logs. 4. System displays ”Collect PKR 300”. 5. Rider pays cash. 6. Driver taps ”Cash Received”.
Post-conditions	Ride is closed. Transaction recorded.

4.3.4 Use Case 4: Cancel Ride (Rider)

Use Case ID	UC-04
Use Case Name	Cancel Ride
Primary Actor	Rider
Pre-conditions	Ride has been booked but not started.
Main Flow	1. Rider taps ”Cancel Ride”. 2. System prompts for a cancellation reason. 3. Rider selects a reason (e.g., ”Driver too far”). 4. System notifies the Driver of cancellation. 5. System updates ride status to ”Cancelled”.
Alternative Flow	2a. Driver has already arrived. 2b. System warns Rider of potential cancellation fee. 2c. Rider confirms cancellation.
Post-conditions	Ride is cancelled. Driver is returned to Available pool.

4.3.5 Use Case 5: Rate Driver

Use Case ID	UC-05
Use Case Name	Rate Driver
Primary Actor	Rider
Pre-conditions	Ride has been completed.

Main Flow	1. System presents Rating Screen. 2. Rider selects Star Rating (1-5). 3. Rider enters optional comment. 4. Rider taps ”Submit”. 5. System updates Driver’s average rating.
Post-conditions	Rating is saved in database.

4.3.6 Use Case 6: View Ride History

Use Case ID	UC-06
Use Case Name	View Ride History
Primary Actor	Rider or Driver
Pre-conditions	User is logged in.
Main Flow	1. User navigates to ”History” tab. 2. System retrieves list of past rides from database. 3. System displays summary (Date, Time, Cost, Route). 4. User taps a specific ride for details.
Post-conditions	Detailed receipt view is displayed.

4.3.7 Use Case 7: Update User Profile

Use Case ID	UC-07
Use Case Name	Update Profile
Primary Actor	Rider or Driver
Pre-conditions	User is logged in.
Main Flow	1. User navigates to ”Settings” -> ”Profile”. 2. User edits Name or Email. 3. User uploads new Profile Picture. 4. User taps ”Save”. 5. System validates inputs and updates database.
Post-conditions	User profile is updated across the app.

4.3.8 Use Case 8: Toggle Availability Status

Use Case ID	UC-08
Use Case Name	Toggle Online Status
Primary Actor	Driver

Pre-conditions	Driver is logged in and documents are verified.
Main Flow	1. Driver taps the ”Online/Offline” switch. 2. System checks GPS signal strength. 3. System updates Driver status in geospatial index. 4. UI updates to show ”You are Online”.
Post-conditions	Driver is visible to nearby riders.

4.4 Appendix D: Data Dictionary

Data Element	**Type**	**Description**
UserID	UUID	Unique identifier for every user.
PhoneNumber	String	Unique mobile number used for login.
Role	Enum	'Rider', 'Driver', or 'Admin'.
VehicleType	Enum	'Bike', 'Rickshaw', 'Mini', 'Sedan'.
RideStatus	Enum	'Searching', 'Accepted', 'Arrived', 'In_Progress', 'Completed', 'Cancelled'.
GeoPoint	JSON	{ lat: Float, lng: Float }.
FareAmount	Decimal	The calculated cost of the trip in PKR.
DriverDocument	String (URL)	Link to the stored image of legal docs.
VerificationStatus	Enum	'Unverified', 'Pending', 'Verified', 'Rejected'.