Software Requirements Specification (SRS) – RideNow App

# 1. Introduction

Purpose: Define requirements for RideNow, a ride-hailing app connecting customers and drivers.  
Scope: Customers request rides; drivers accept and complete rides; payments and ratings are recorded.  
Stakeholders: Customers, Drivers, Operations Team, Finance.

# 2. Overall Description

Product Perspective: Mobile-first app, with backend database & APIs.  
Product Functions: Ride booking, driver assignment, payment processing, rating system.  
User Characteristics: Customers (any age group with smartphones), drivers (licensed drivers).  
Constraints: Must support 10,000 concurrent users, integrate with GPS APIs, secure payments.

# 3. Functional Requirements

* FR1 – User Management: Users can register/login with email, phone, and password. Drivers must upload license info.
* FR2 – Ride Request: Customer submits pickup and drop-off locations. System assigns nearest available driver.
* FR3 – Ride Acceptance: Drivers can accept/reject rides. Once accepted, ride status = 'accepted'.
* FR4 – Ride Completion: Driver updates ride status = 'completed'. System calculates fare.
* FR5 – Payment: Customer pays by cash or card. Payment record created in DB.
* FR6 – Rating: After ride completion, customer can rate driver (1–5 stars).

# 4. Non-Functional Requirements

* Performance: 2-second response time.
* Scalability: 100,000 active users.
* Security: Encrypted payment info.
* Reliability: 99.5% uptime.
* Usability: Intuitive UI.

# 5. Database Design (Simplified Schema)

* customers(customer\_id, full\_name, email, phone)
* drivers(driver\_id, full\_name, license\_number, status)
* rides(ride\_id, customer\_id, driver\_id, pickup, dropoff, status)
* payments(payment\_id, ride\_id, amount, method, status)
* ratings(rating\_id, ride\_id, rating, comments)

# 6.SQL Queries for Table Creation

* -- Customers
* use testdb;
* CREATE TABLE customers (
* customer\_id INT PRIMARY KEY AUTO\_INCREMENT,
* full\_name VARCHAR(100) NOT NULL,
* email VARCHAR(100) UNIQUE NOT NULL,
* phone VARCHAR(15) UNIQUE NOT NULL
* );
* -- Drivers
* CREATE TABLE drivers (
* driver\_id INT PRIMARY KEY AUTO\_INCREMENT,
* full\_name VARCHAR(100) NOT NULL,
* license\_number VARCHAR(50) UNIQUE NOT NULL,
* status ENUM('available','on\_trip') DEFAULT 'available'
* );
* -- Rides
* CREATE TABLE rides (
* ride\_id INT PRIMARY KEY AUTO\_INCREMENT,
* customer\_id INT NOT NULL,
* driver\_id INT,
* pickup\_location VARCHAR(255) NOT NULL,
* dropoff\_location VARCHAR(255) NOT NULL,
* status ENUM('requested','accepted','completed','cancelled') DEFAULT 'requested',
* FOREIGN KEY (customer\_id) REFERENCES customers(customer\_id),
* FOREIGN KEY (driver\_id) REFERENCES drivers(driver\_id)
* );
* -- Payments
* CREATE TABLE payments (
* payment\_id INT PRIMARY KEY AUTO\_INCREMENT,
* ride\_id INT NOT NULL,
* amount DECIMAL(10,2) NOT NULL,
* method ENUM('cash','card') NOT NULL,
* status ENUM('pending','paid','failed') DEFAULT 'pending',
* FOREIGN KEY (ride\_id) REFERENCES rides(ride\_id) ON DELETE CASCADE
* );
* -- Ratings
* CREATE TABLE ratings (
* rating\_id INT PRIMARY KEY AUTO\_INCREMENT,
* ride\_id INT NOT NULL,
* rating INT CHECK (rating BETWEEN 1 AND 5),
* comments TEXT,
* FOREIGN KEY (ride\_id) REFERENCES rides(ride\_id) ON DELETE CASCADE

);

# 6. Use Cases

UC01 – Request Ride  
Actor: Customer  
Flow: Login → Enter pickup/drop → Submit request → System assigns driver.

UC02 – Accept Ride  
Actor: Driver  
Flow: View ride → Accept/Reject → System updates status.

UC03 – Payment  
Actor: Customer  
Flow: After completion → System generates fare → Customer pays.

UC04 – Rate Driver  
Actor: Customer  
Flow: After ride → Submit rating → Stored in DB.

# 7. Traceability Matrix

|  |  |  |
| --- | --- | --- |
| Requirement | Use Case | Test Case |
| FR1 – Register/Login | UC01 | TC01, TC02 |
| FR2 – Request Ride | UC01 | TC03 |
| FR3 – Accept Ride | UC02 | TC04 |
| FR5 – Payment | UC03 | TC05, TC06 |
| FR6 – Rating | UC04 | TC07 |