**Online Ride-Sharing Platform**

**(SmartRide)**

**Software Engineering (SE)**

**By Students:**

1. **StudentID1 : Phạm Lê Anh Khôi**
2. **StudentID2 : Dương Thành Long**

|  |  |  |  |
| --- | --- | --- | --- |
| **Reference:** | **Team\_XX\_SE\_Case Study – Smart Ride \_v0.1** | | |
| **Audience:** | **Mr. Pham Thai Ky Trung** | **Document Version:** | **May, 2025** |
| **Outcome:** | **Online Ride-Sharing Platform (SmartRide)** | | |
| **Abstract:** | This document provides an in-depth analysis of a proposed urban ride-sharing business that connects customers with drivers using vehicles for transportation | | |

Contents

[I. Introduction 4](#_Toc199279312)

[II. Project Overview 5](#_Toc199279313)

[1. Project Objectives 5](#_Toc199279314)

[2. Domain Vocabulary 6](#_Toc199279315)

[3. Project Type 8](#_Toc199279316)

[4. Goals 9](#_Toc199279317)

[5. Project Incentives 10](#_Toc199279318)

[6. Assumptions 11](#_Toc199279319)

[7. Scope 12](#_Toc199279320)

[8. Functional Requirements 14](#_Toc199279321)

[9. Non-functional Requirements 15](#_Toc199279322)

[III. Problem Domain 16](#_Toc199279323)

[1. Pain Points 16](#_Toc199279324)

[2. Domain Entities 17](#_Toc199279325)

[3. Actors 18](#_Toc199279326)

[4. List of Tasks 19](#_Toc199279327)

[IV. Data Model 21](#_Toc199279328)

[V. Tasks and Support 22](#_Toc199279329)

[1. Task 1: User Registration & Authentication 22](#_Toc199279330)

[2. Task 2: Ride Booking 23](#_Toc199279331)

[3. Task 3: Driver Assignment 24](#_Toc199279332)

[4. Task 4: Ride Status Tracking 25](#_Toc199279333)

[5. Task 5: Payment Processing & Receipt Generation 26](#_Toc199279334)

[6. Task 6: Operational Reporting 27](#_Toc199279335)

[VI. Workflow 28](#_Toc199279336)

[1. User Registration & Authentication 28](#_Toc199279337)

[2. Ride Booking 29](#_Toc199279338)

[3. Payment Processing & Receipt Generation 30](#_Toc199279339)

[VII. Validation of Requirements 31](#_Toc199279340)

[1. Desk Reviews 31](#_Toc199279341)

[2. Traceability Matrix 32](#_Toc199279342)

[3. CRUD Check 33](#_Toc199279343)

[VIII. Possible Solutions 34](#_Toc199279344)

[1. Problem 1: Ride Matching Delays During Peak Hours 34](#_Toc199279345)

[2. Problem 2: Payment Failures or Discrepancies 35](#_Toc199279346)

[3. Problem 3: Fake Registrations or Fraudulent Use 36](#_Toc199279347)

# Introduction

- SmartRide is a transportation service company operating in a busy metropolitan city. The company connects customers with available drivers for a fee, using cars and motorbikes. Currently, SmartRide’s operations are managed manually, leading to inefficiencies such as long wait times, lost ride opportunities, and slow payment handling. To address these challenges and to prepare for future growth, SmartRide has commissioned the development of a new Online Ride-Sharing Platform (ORSP).

- SESoft Consulting has been tasked with designing and developing the ORSP to automate and enhance SmartRide’s operations, focusing on improving ride booking, driver assignment, real-time tracking, and online payments. This system is expected to deliver a faster, more reliable, and user-friendly experience for both customers and drivers, while also providing managerial insights to support business decision-making.

# Project Overview

## Project Objectives

* **Automate ride-booking** by allowing customers to easily request rides through a digital platform.
* **Reduce customer wait times** by efficiently matching available drivers to ride requests in real time.
* **Introduce secure online payment processing** to replace manual cash handling.
* **Provide real-time tracking** for customers to monitor driver location and estimated arrival times.
* **Enable account management** for customers and drivers through user profiles.
* **Generate operational reports** to help SmartRide’s management monitor service performance and demand trends.
* Initially, the system will focus on the core functionality, with the flexibility to add new features such as ride-sharing options, loyalty programs, and promotional offers in the future.

## Domain Vocabulary

| **Term** | | **Definition** |
| --- | --- | --- |
| **ORSP** | | Online Ride-Sharing Platform — the digital system enabling ride booking, tracking, payment, and reporting. |
| **Customer** | | A user of the ORSP who requests rides, tracks driver location, and makes payments. |
| **Driver** | | A service provider registered with the ORSP who accepts ride requests and transports customers. |
| **Ride** | | A single trip from pickup to drop-off, representing the core business transaction. |
| **Vehicle** | | A data holder for a driver’s mode of transport (car or motorbike), including its details. |
| **Location** | | Geographical coordinates and address information for pickup and drop-off points. |
| **ETA** | | Estimated Time of Arrival — the calculated time for a driver to reach the pickup location. |
| **Payment** | | The process and record of transferring fare from customer to driver, managed via a third-party gateway. |
| **Account** | | Authentication and profile information for all users (customers, drivers, managers). |
| **Manager** | | Administrative user who oversees operations, accesses reports, and manages system settings. |
| **Report** | | Aggregated data summaries (e.g., ride volumes, revenues) provided for managerial insight. |
| **Payment Gateway** | | External service integrated into the ORSP for secure payment processing (e.g., Stripe, PayPal). |
| **GPS Navigation Service** | | Third-party system providing real-time location tracking and routing for drivers. |
| **Notification** | | Real-time messages sent to customers and drivers (e.g., ride status updates, payment confirmations). |
| **Availability** | | A driver’s current status (online/offline) indicating readiness to accept new ride requests. |
| **Scalability** | The system’s ability to handle increasing numbers of users, rides, and transactions without performance degradation. | |

## Project Type

The SmartRide ORSP engagement is a **custom software development** project with the following characteristics:

* **Domain:** Transportation / Ride-Sharing Platform
* **Application Type:**
  + **Web Application**: Responsive portal for managers and administrative users
  + **Mobile Applications**: Native or cross-platform apps for iOS and Android, used by customers and drivers
* **Architecture Style:** Client-server, cloud-hosted microservices
* **Technology Stack (Indicative):**
  + Backend: ASP.NET Core with Entity Framework Core
  + Frontend: React (web) and React Native (mobile)
  + Database: Microsoft SQL Server
  + Hosting: Azure or AWS

## Goals

|  |
| --- |
| * Enable customers to create accounts, book rides, and pay online. |
| * Enable drivers to create profiles, accept ride requests, and navigate to customers. |
| * Provide real-time GPS tracking and ETA updates to customers. |
| * Automate payment handling, receipts, and confirmations. |
| * Allow managers to access reports on rides, revenue, and demand trends. |
| * Ensure system security, reliability, and scalability for future growth. |

## Project Incentives

* **Reduce Customer Churn:** By cutting down wait times and offering real-time tracking, SmartRide aims to prevent lost bookings and improve overall customer loyalty.
* **Increase Ride Completion Rate:** Automated driver matching and dynamic dispatch will help fulfill more ride requests, directly boosting revenue.
* **Streamline Operations:** Eliminating manual processes for dispatch and payments will lower administrative overhead and reduce error rates.
* **Enhance Driver Utilization:** Faster, fairer assignment of rides will keep drivers active and reduce idle time, improving their earnings and retention.
* **Enable Data-Driven Growth:** Built-in reporting and analytics will empower management to spot demand trends, optimize pricing, and plan for peak periods.
* **Future-Proof the Platform:** A modular, scalable system foundation will allow SmartRide to roll out new features (shared rides, loyalty programs, new vehicle types) with minimal rework.

## Assumptions

* **City Coverage:** SmartRide initially operates within a single metropolitan area; no immediate multi-city expansion is planned.
* **User Technology Proficiency:** All customers and drivers possess basic smartphone literacy and can install and use mobile apps for booking rides, navigation, and payments.
* **Hardware Availability:** Drivers use their own GPS-enabled smartphones. Server infrastructure (cloud or on-premises) will be provisioned separately and is outside the scope of this project.
* **Ride Type Stability:** Only cars and motorbikes are supported at launch. Any new vehicle types (e.g., vans, e-bikes) will be added via future system updates.
* **Customer Interactions:** The system handles ride requests, real-time tracking, and payments. Customer support inquiries (e.g., refunds, complaints) are managed through external channels.
* **Third-Party Payment Gateway:** SmartRide integrates with an existing PCI-compliant payment provider; the platform does not process or store raw payment card data.
* **External Navigation Service:** A reliable GPS/navigation API is available for real-time location tracking and ETA calculations.
* **User Roles:** The system supports three primary user roles—Customer, Driver, and Manager—and each must register an Account before using the platform.

## Scope

**Included**

* **User Account Management:**
  + Registration, login, and profile management for Customers, Drivers, and Managers.
* **Ride Lifecycle:**
  + Ride request submission with pickup and drop-off locations.
  + Automatic driver matching and assignment.
  + Real-time GPS-based tracking and ETA updates.
  + Ride status updates (requested, accepted, in-progress, completed).
* **Vehicle Management:**
  + Registration and maintenance of Driver’s Vehicle details (type, license plate, status).
* **Payment Processing:**
  + Integration with a third-party, PCI-compliant payment gateway.
  + Fare calculation, transaction execution, and digital receipt generation.
* **Reporting & Analytics:**
  + Generation of operational reports (ride volumes, revenue, driver activity) for Managers.
* **Notifications:**
  + Real-time alerts to Customers and Drivers for ride status changes and payment confirmations.

**Excluded**

* **Hardware Procurement:** Server infrastructure, driver devices, or in-vehicle hardware.
* **Customer Support Workflows:** Ticketing system, dispute resolution, or refund management.
* **Dynamic Pricing & Promotions:** Surge pricing algorithms, loyalty rewards, or discount campaigns.
* **Shared Rides & Pooling:** Carpooling or multi-passenger matching.

## Functional Requirements

* **Enable Customer Account Management:** Customers must be able to create, update, and manage their user accounts, including personal details, payment information, and ride history.
* **Enable Driver Account Management:** Drivers must be able to register, update, and manage their profiles, vehicle information, and availability status.
* **Support Ride Booking:** Customers must be able to input pickup and drop-off locations and submit ride requests through the platform.
* **Automate Driver Assignment:** The system must automatically locate available drivers based on proximity, availability, and estimated arrival times, and assign them to incoming ride requests.
* **Provide Real-time Tracking:** Both customers and drivers must have access to live GPS tracking to monitor driver location, route progress, and estimated arrival times.
* **Enable Secure Online Payments:** Customers must be able to securely complete ride payments through an integrated online payment gateway, with digital receipts generated upon successful transactions.
* **Generate Operational Reports:** Managers must be able to access system-generated reports summarizing ride volumes, driver activity, peak usage periods, and other relevant operational metrics.
* **Support Notifications and Updates:** The system must send real-time updates to customers and drivers about ride status changes, estimated arrival times, and payment confirmations.
* **Ensure Ride History Management:** Customers and drivers must be able to view their respective ride histories, including trip details, payment records, and ratings.
* **Facilitate System Scalability and Maintenance:** The system must be designed to easily accommodate future expansions, such as adding new cities, new vehicle types, or new customer loyalty programs.

## Non-functional Requirements

* **Performance and Scalability**  
  The system must handle high volumes of concurrent users, especially during peak hours, without significant performance degradation. It must be scalable to support future expansion to additional cities and increased numbers of customers and drivers.
* **Reliability and Availability**  
  The platform must maintain a minimum uptime of 99.9% to ensure customers and drivers can access the service at all times. System recovery mechanisms must be in place to handle unexpected failures with minimal service interruption.
* **Security**  
  The system must protect user data through encryption and secure authentication methods. Integration with third-party payment gateways must comply with industry standards for financial data protection (e.g., PCI DSS compliance).
* **Usability**  
  The platform must provide a clean, intuitive user interface that is easy to navigate for customers, drivers, and managers. Key tasks (such as booking a ride, accepting a ride, or making a payment) must be easily completed with minimal steps and training.
* **Maintainability**  
  The system must be designed with a modular architecture to allow easy updates, bug fixes, and the addition of new features without significant system downtime.
* **Interoperability**  
  The ORSP must integrate seamlessly with external systems such as GPS navigation services and third-party payment gateways without requiring manual intervention.
* **Responsiveness**  
  Real-time features such as ride tracking, driver assignment, and notification delivery must occur with minimal latency to enhance the user experience.

# Problem Domain

## Pain Points

SmartRide’s current manual operations present several critical challenges that negatively impact customer satisfaction, driver efficiency, and overall business growth:

* **Long Wait Times**: Customers often experience significant delays in securing a ride, especially during peak hours, due to the lack of an automated driver assignment process.
* **Missed Ride Opportunities**: Without an efficient system to match customers with available drivers, many potential rides are lost, leading to lost revenue and frustrated customers.
* **Manual Payment Processing**: Payments are currently handled manually between drivers and customers, which is time-consuming, error-prone, and inconvenient for both parties.
* **Driver Assignment Inefficiencies**: During periods of high demand, there is no organized method for quickly identifying and assigning nearby available drivers to customers.
* **Limited Visibility for Customers**: Customers have no way to track where their driver is or receive accurate arrival time estimates, leading to uncertainty and dissatisfaction.
* **High Administrative Overhead**: Manual ride tracking, payment recording, and business reporting require significant administrative effort, slowing down operations and decision-making.
* **Scalability Limitations**: The current manual approach is not scalable. As customer demand grows, the company struggles to handle increased ride requests without compromising service quality.

## Domain Entities

| **Entity** | **Description** |
| --- | --- |
| **Customer** | A user who requests and pays for rides via the platform. |
| **Driver** | A service provider who accepts ride requests and transports customers. |
| **Ride** | A record of a single trip, including pickup/drop-off locations, status, and associated users. |
| **Vehicle** | Details about a driver’s transport (car or motorbike), including type and registration data. |
| **Location** | Geographic information (latitude, longitude, address) for ride pickup and drop-off points. |
| **Payment** | Transaction details for a ride, including fare, method, status, and receipt information. |
| **Account** | Authentication and profile information for all users (customers, drivers, managers). |
| **Manager** | An administrative user who oversees operations and accesses reports and analytics. |
| **Report** | Aggregated summaries of rides, payments, and performance metrics for managerial review. |

## Actors

* **Customer:** An individual who uses the platform to book rides, track driver locations, make payments, and manage their ride history and personal profile.
* **Driver:** A registered individual who uses the platform to receive ride requests, accept bookings, navigate to pickup and drop-off locations, and manage their availability and profile.
* **Manager:** A SmartRide administrative user who oversees system operations, reviews reporting and analytics, manages customer or driver accounts when necessary, and monitors service performance trends.
* **Payment Gateway** *(External Actor):* A third-party service responsible for securely processing customer payments. SmartRide will integrate with an external gateway to manage financial transactions without directly handling sensitive payment data.
* **GPS Navigation Service** *(External Actor):* A third-party service used to provide real-time location tracking, route optimization, and estimated arrival times for both customers and drivers.

## List of Tasks

**User Account Management**

* Register new Customer, Driver, and Manager accounts
* Log in and log out of the system
* Update and manage user profiles and credentials

**Ride Booking**

* Enter pickup and drop-off locations
* Submit ride requests
* Display available vehicle types and ETAs

**Driver Assignment**

* Locate nearest available Driver
* Notify Driver of incoming ride request
* Allow Driver to accept or decline a ride

**Real-Time Tracking**

* Continuously update Driver’s GPS location
* Recalculate and display ETA to Customer
* Notify Customer and Driver of status changes

**Ride Status Management**

* Transition a Ride through statuses (Requested → Accepted → In-Progress → Completed → Rated)
* Record timestamps for lifecycle events

**Navigation Support**

* Provide turn-by-turn directions to Driver
* Re-route dynamically based on traffic or route changes

**Payment Processing**

* Calculate fare based on distance, time, and vehicle type
* Integrate with third-party payment gateway for secure transactions
* Generate and deliver digital receipts

**Notifications & Alerts**

* Send real-time notifications for ride confirmations, arrivals, and completions
* Alert users to payment status and account changes

**Reporting & Analytics**

* Aggregate ride and payment data
* Generate on-demand operational reports for Managers (e.g., daily summaries, peak-hour trends)

**Error Handling & Validation**

* Validate user inputs (locations, payment details, profile data)
* Gracefully handle and report system or integration failures

# Data Model

A diagram of a company

AI-generated content may be incorrect.

# Tasks and Support

## Task 1: User Registration & Authentication

| **Aspect** | **Description** |
| --- | --- |
| **Purpose** | Allow new users (Customers, Drivers, Managers) to create an account and log in securely. |
| **Precondition** | User has downloaded the SmartRide app or accessed the web portal. |
| **Frequency** | Once per user (registration), then at each subsequent app launch or session (login). |
| **Critical** | Insecure or failed authentication exposes privacy/data or prevents use of the system. |
| **Work area** | Registration & Login screens (mobile/web). |
| **Subtask** | Example Solution |
| **1. Collect user details** | Present form for name, email/phone, password, (driver: license) |
| **2. Validate input** | Check mandatory fields, email format, password strength |
| **3. Create account record** | Invoke Account.create() → store credentials in user database |
| **4. Send confirmation (optional)** | Email/SMS with verification link or code |
| **5. Authenticate on login** | Account.login() → check credentials → issue session token |
| **Variants** |  |
| 1a. **Social Login** | User chooses Google/Facebook SSO → OAuth flow → create or link local account. |
| 1b. **Password Reset** | User forgot password → requests reset link → completes password change. |

## Task 2: Ride Booking

| **Aspect** | **Description** |
| --- | --- |
| **Purpose** | Enable a Customer to request a new ride by specifying pickup and drop-off locations. |
| **Precondition** | User is authenticated and has a valid payment method on file. |
| **Frequency** | Every time a Customer needs a trip. |
| **Critical** | Invalid locations or missing payment info block ride creation and frustrate users. |
| **Work area** | “Book a Ride” screen with map and form. |
| **Subtask** | Example Solution |
| **1. Enter pickup location** | Map-based search / GPS “use my current location” |
| **2. Enter drop-off location** | Address autocomplete + map pin drop |
| **3. Validate service area** | Check coordinates against supported city polygon |
| **4. Show vehicle types & ETAs** | Query ETA service for cars/motorbikes |
| **5. Confirm ride request** | Ride.create() + persist pickup/drop-off and customer ID |
| **Variants** |  |
| **2a. Schedule for later** | Customer selects future date/time → system enqueues ride request. |
| **2b. Fare estimate only** | Customer previews fare and ETA without booking. |

## Task 3: Driver Assignment

| **Aspect** | **Description** |
| --- | --- |
| **Purpose** | Automatically match a new Ride request to the most suitable available Driver. |
| **Precondition** | A Ride has been created and at least one Driver is marked available nearby. |
| **Frequency** | Once per new Ride request. |
| **Critical** | Poor matching leads to long waits or ride cancellations. |
| **Work area** | Backend dispatch service (RideManager). |
| **Subtask** | Example Solution |
| **1. Locate nearby drivers** | Spatial query: Driver.location within radius of pickup point |
| **2. Filter by availability** | Exclude drivers marked “busy” or “offline” |
| **3. Rank by ETA & fairness** | Sort by estimated arrival time and recent assignment count |
| **4. Dispatch assignment** | Ride.assignDriver(driverId) → notify driver via push/SMS |
| **5. Await driver response** | Timeout after N seconds → retry or escalate to next candidate |
| **Variants** |  |
| **3a. Manual override** | Manager forces assignment to a specific driver in special cases. |
| **3b. Shared ride matching** | (Future) match multiple customers to one driver. |

## Task 4: Ride Status Tracking

| **Aspect** | **Description** |
| --- | --- |
| **Purpose** | Provide both Customer and Driver with real-time updates on ride progress and ETA. |
| **Precondition** | Driver has accepted the Ride and GPS updates are being sent. |
| **Frequency** | Continuous, throughout the lifecycle of each Ride. |
| **Critical** | Delays or inaccuracies in tracking decrease user trust and can cause missed pickups. |
| **Work area** | Mobile apps (Customer & Driver), backend location service. |
| **Subtask** | Example Solution |
| **1. Driver sends GPS heartbeat** | App → Location.update(latitude, longitude) |
| **2. Update Ride record** | Ride.updatePosition() & recalculate ETA |
| **3. Push notification to Customer** | Real-time socket message or push notification |
| **4. Display map & ETA** | Mobile UI map redraw with driver marker and time label |
| **Variants** |  |
| **4a. Intermittent connectivity** | Buffer location updates locally → sync when online. |
| **4b. Alternative routing** | Driver manually updates route → recalculate ETA. |

## Task 5: Payment Processing & Receipt Generation

| **Aspect** | **Description** |
| --- | --- |
| **Purpose** | Charge the Customer for a completed Ride and generate a digital receipt. |
| **Precondition** | Ride status = “Completed” and Customer has a valid payment method. |
| **Frequency** | Once per Ride completion. |
| **Critical** | Failed transactions lead to revenue loss and require manual follow-up. |
| **Work area** | Backend payment service (PaymentProcessor) and Customer app. |
| **Subtask** | Example Solution |
| **1. Calculate fare** | Formula: base fare + distance × rate + time × rate |
| **2. Initiate transaction** | Payment.process(paymentInfo) → call third-party gateway API |
| **3. Handle gateway response** | On success → Payment.markPaid(); on failure → retry/alert |
| **4. Generate digital receipt** | Payment.generateReceipt() → PDF or in-app view |
| **5. Notify Customer** | Push notification + email with receipt link |
| **Variants** |  |
| **5a. Wallet top-up** | Deduct from in-app wallet balance before fallback to card. |
| **5b. Split fare** | Customer splits payment among multiple riders. |

## Task 6: Operational Reporting

| **Aspect** | **Description** |
| --- | --- |
| **Purpose** | Provide Managers with insights on ride volumes, revenue, and driver performance over time. |
| **Precondition** | System has processed multiple Rides and Payments to aggregate data. |
| **Frequency** | On-demand or scheduled (e.g., daily at midnight). |
| **Critical** | Inaccurate or delayed reports hinder strategic decision-making. |
| **Work area** | Manager portal / reporting dashboard. |
| **Subtask** | Example Solution |
| **1. Query ride/payment data** | ReportGenerator.fetchRides(startDate, endDate) |
| **2. Aggregate metrics** | Calculate totals, averages, peak-hour counts |
| **3. Format report** | Render as table/chart in dashboard |
| **4. Export or schedule delivery** | CSV/PDF export or automated email dispatch |
| **Variants** |  |
| **6a. Custom date range** | Manager selects arbitrary start/end dates. |
| **6b. Performance alerts** | System automatically flags underperforming metrics via email. |

# Workflow

## User Registration & Authentication

A diagram of a company

AI-generated content may be incorrect.

## Ride Booking

A diagram of a flowchart

AI-generated content may be incorrect.

## Payment Processing & Receipt Generation

A screenshot of a diagram

AI-generated content may be incorrect.

# Validation of Requirements

## Desk Reviews

Conduct systematic, internal reviews of the requirements specification to ensure completeness, consistency, clarity, and alignment with stakeholder needs before broader stakeholder involvement.

**Participants:**

* **Lead Business Analyst** (chairs the review)
* **Domain Expert** (ride-sharing operations specialist)
* **Technical Architect** (assesses technical feasibility)
* **QA Lead** (identifies testability and ambiguity issues)
* **UX Designer** (reviews usability and workflow clarity)

**Scope of Review:**

* **Completeness:** Verify that all major business processes (registration, booking, assignment, tracking, payment, reporting) are covered by requirements.
* **Consistency:** Ensure terminology is used uniformly (e.g., “Ride,” “Customer,” “Driver,” “ETA”).
* **Clarity:** Check that each requirement is unambiguous, phrased in active voice, and has clear acceptance criteria.
* **Feasibility:** Assess whether non-functional targets (performance, availability, security) are realistic given the technology stack.
* **Traceability:** Confirm that each functional requirement can be traced back to a business goal or pain point.

## Traceability Matrix

| **Requirement** | **Quality Attribute(s)** | **Assessment Metric** |
| --- | --- | --- |
| Customer can request and book a ride via the app/web | • Usability  • Performance  • Reliability | • <2 s for ride-request submission• ≥ 95% successful bookings• <1% booking errors |
| Driver can receive and accept ride requests | • Performance  • Usability | • Notification delivered <1 s after assignment• ≥ 98% driver-app uptime |
| Real-time driver tracking and ETA updates for customers | • Performance  • Reliability | • GPS location updated every 5 s• ≥ 99% location accuracy• <3 s ETA refresh |
| Customer can complete secure online payment | • Security  • Performance | • PCI-DSS compliance• <3 s transaction processing time• 0% unencrypted PII |
| Manager can generate operational reports on demand | • Scalability  • Performance | • Reports for 30 days of data generated <5 s• ≥ 99% data accuracy |
| System must support ≥ 10 000 concurrent users | • Scalability  • Availability | • Sustained throughput at 10 000 sessions/minute• 99.9% uptime in 30-day window |
| Notifications of ride status changes reach users reliably | • Reliability  • Performance | • ≥ 99.5% notification delivery success• Delivery latency <2 s |

## CRUD Check

| **Task / Actors** | **Customer** | **Driver** | **Cashier** | **System** | **Manager** |
| --- | --- | --- | --- | --- | --- |
| Register Account | C, R, U, D | C, R, U, D | R |  |  |
| Book a Ride | C, R | R |  | R, U | R |
| Accept Ride Request |  | C, R, U |  | R, U | R |
| Track Ride in Real-Time | R, U | R, U |  | R, U |  |
| Make Payment | C, R |  | R, U | C, R, U | R, U |
| Generate Receipt | R |  | R, U | C, R | R, U |
| Generate Reports |  |  |  | C, R | R, U, D |
| Manage Driver Information |  | C, R, U, D |  |  | R, U, D |

# Possible Solutions

## Problem 1: Ride Matching Delays During Peak Hours

**Description:**  
During high-demand periods (e.g., rush hour, weekends), customers may experience delays in being matched with available drivers. This can result in poor user experience, ride cancellations, and reduced customer retention.

**Possible Solutions:**

1. **Dynamic Pricing (Surge Pricing):**
   * Implement surge pricing algorithms that increase fare rates during peak hours to incentivize more drivers to become available.
   * Communicate surge pricing transparently to users before ride confirmation.
2. **Driver Incentive System:**
   * Offer bonuses or higher commissions for drivers accepting rides during high-demand windows.
   * Introduce gamification (e.g., reward points) to encourage driver availability.
3. **AI-based Demand Forecasting:**
   * Use machine learning models to predict demand spikes based on historical data, location, weather, and events.
   * Preemptively alert or position drivers in expected high-demand zones.

## Problem 2: Payment Failures or Discrepancies

**Description:**  
Users may face issues during payment processing due to connectivity errors, gateway downtime, or inconsistent fare calculations. This can lead to failed transactions, overcharge, or underpayment.

**Possible Solutions:**

1. **Multi-Gateway Payment Support:**
   * Integrate multiple payment gateway providers (e.g., Stripe, PayPal, Razorpay) with failover mechanisms to ensure continuity if one service fails.
2. **Pre-Authorization of Fare Estimate:**
   * Lock a fare estimate with a margin before ride starts; finalize fare at end with fare breakdown.
   * Handle discrepancies with automated dispute resolution or escalation to customer service.
3. **Payment Retry and Wallet Option:**
   * Offer retry mechanisms and fallback to in-app wallet balance for failed transactions.
   * Notify the user via app and email with detailed receipts and refund process if needed.

## Problem 3: Fake Registrations or Fraudulent Use

**Description:**  
The system may be vulnerable to botted accounts or malicious users creating fake rider or driver accounts to exploit promotions or conduct fraud.

**Possible Solutions:**

1. **Phone and Email Verification:**
   * Enforce OTP-based verification during registration.
   * Require verified email addresses and phone numbers for account creation.
2. **Document and Face Verification for Drivers:**
   * Implement KYC (Know Your Customer) checks including driver's license, ID proof, and real-time selfie matching.
   * Use third-party ID verification services (e.g., Onfido, Jumio).
3. **Behavioral Analysis & Fraud Detection Algorithms:**
   * Monitor suspicious patterns like high ride cancellation rate, location spoofing, or frequent promo code use.
   * Apply AI-driven fraud detection systems to flag and block such activities automatically.