VIETNAM GENERAL CONFEDERATION OF LABOR

**TON DUC THANG UNIVERSITY**

**FACULTY OF INFORMATION TECHNOLOGY**

A red and blue logo

AI-generated content may be incorrect.

**Instructor: Phạm Kỳ Thái Trung**

**Class: 23K50201**

**Students:**  
**523K0010 – Phạm Lê Anh Khôi**

**523K0013 – Dương Thành Long**

**Signature**



Phạm Lê Anh Khôi

A signature of a person

AI-generated content may be incorrect.

Dương Thành Long

**Intellectual Property**

***Copyright 2025 Team XX.***

The following documentation, the content therein and/or the presentation of its information are proprietary to and embodies the confidential processes, designs, technologies and otherwise of Team XX.  All copyright, trademarks, trade names, patents, industrial designs, and other intellectual property rights contained herein are, unless otherwise specified, the exclusive property of Team XX.

The ideas, concepts and/or their application embodied within this documentation remain and constitute items of intellectual property which nevertheless belong to Team XX.

The information (including, but by no means limited to, data, drawings, specification, documentation, software listings, source and/or object code) shall not be disclosed, manipulated, disseminated or otherwise in any manner inconsistent with nature and/or conditions under which this documentation has been issued.

The information contained herein is believed to be accurate and reliable. Team XX accepts no responsibility for its use in any way whatsoever. Team Five shall not be liable for any expenses, damage and/or related costs which may result from the use of the information contained herein.

The information contained herein is subject to change without notice.

All Rights Reserved. Copyright herein is expressly protected at common law, statute and under various International and Multi-National Treatises (including, but by no means limited to, the Berne Convention for the Protection of Literary and Artistic Works).

Contents

[Executive Summary 7](#_Toc199085469)

[I. Initial Activities 8](#_Toc199085470)

[1. System Vision Document 8](#_Toc199085471)

[2. Stakeholder Engagement 9](#_Toc199085472)

[3. Obtain Project Approval 10](#_Toc199085473)

[4. Business Modeling / Requirements 10](#_Toc199085474)

[Scoped business use case: Book Ride 11](#_Toc199085475)

[5. Business Processes / Flowchart of Requirements 12](#_Toc199085476)

[6. List of Requirements 13](#_Toc199085477)

[II. Plan your project. 14](#_Toc199085478)

[1. Project Scope 14](#_Toc199085479)

[2. Project Objectives 14](#_Toc199085480)

[3. Project Phases and Timeline 14](#_Toc199085481)

[4. Team Roles and Responsibilities 15](#_Toc199085482)

[5. Tools and Technologies 15](#_Toc199085483)

[6. Risk Management Plan 16](#_Toc199085484)

[7. Communication Plan 16](#_Toc199085485)

[III. Discovery and Understanding the Details 17](#_Toc199085486)

[1. System Narrative 17](#_Toc199085487)

[2. Actors and their goals 18](#_Toc199085488)

[3. List of Events 19](#_Toc199085489)

[4. List of Actors 20](#_Toc199085490)

[5. List of Use Cases 21](#_Toc199085491)

[6. Use Case Diagram 22](#_Toc199085492)

[i. Register Account Subsystem: 23](#_Toc199085493)

[ii. Book Ride Subsystem: 23](#_Toc199085494)

[iii. Process Payment Subsystem: 24](#_Toc199085495)

[iv. Track Driver Subsystem: 24](#_Toc199085496)

[7. Domain Class Model 25](#_Toc199085497)

[i. Register Account Subsystem: 25](#_Toc199085498)

[ii. Book Ride Subsystem: 26](#_Toc199085499)

[iii. Process Payment Subsystem: 26](#_Toc199085500)

[iv. Track Driver Subsystem: 27](#_Toc199085501)

[8. Use Case Descriptions 28](#_Toc199085502)

[i. Use Case: Register Account 28](#_Toc199085503)

[ii. Use Case 2: Book ride 30](#_Toc199085504)

[iii. Use Case 3: Process payment 32](#_Toc199085505)

[iv. Use Case 4: Track Driver 35](#_Toc199085506)

[9. Verifying use cases for Actor 38](#_Toc199085507)

[i. Verifying use cases for Customer 38](#_Toc199085508)

[ii. Verifying use cases for Driver 39](#_Toc199085509)

[iii. Verifying use case for Admin 40](#_Toc199085510)

[iv. Verifying use cases for Payment Gateway 41](#_Toc199085511)

[IV. Design System Components 42](#_Toc199085512)

[1. Design class for Register Account 42](#_Toc199085513)

[i. Domain Design Class 42](#_Toc199085514)

[ii. Controller 43](#_Toc199085515)

[iii. UI 43](#_Toc199085516)

[iv. Data Access 43](#_Toc199085517)

[2. Design class for Book Ride 44](#_Toc199085518)

[i. Domian Design Class 44](#_Toc199085519)

[ii. Controller 44](#_Toc199085520)

[iii. UI 45](#_Toc199085521)

[iv. Data Access 45](#_Toc199085522)

[3. Design class for Process Payment 46](#_Toc199085523)

[i. Domian Design Class 46](#_Toc199085524)

[ii. Controller 46](#_Toc199085525)

[iii. UI 47](#_Toc199085526)

[iv. Data Access 47](#_Toc199085527)

[4. Design class for Track Driver 48](#_Toc199085528)

[i. Domain Design Class 48](#_Toc199085529)

[ii. Controller 48](#_Toc199085530)

[iii. UI 49](#_Toc199085531)

[iv. Data Access 49](#_Toc199085532)

[V. Build, Test, Integrated System Component 50](#_Toc199085533)

[1. Package Diagram for Subsystem 50](#_Toc199085534)

[i. Register Account Subsystem 50](#_Toc199085535)

[ii. Book Ride Subsystem 51](#_Toc199085536)

[iii. Process Payment Subsystem 52](#_Toc199085537)

[iv. Track Driver 53](#_Toc199085538)

[2. Database Design 53](#_Toc199085539)

[i. Customer 53](#_Toc199085540)

[ii. Address 54](#_Toc199085541)

[iii. Account 55](#_Toc199085542)

[iv. Driver 56](#_Toc199085543)

[v. Ride 57](#_Toc199085544)

[vi. Payment 58](#_Toc199085545)

[vii. DriverLocation 59](#_Toc199085546)

[3. SQL Code 59](#_Toc199085547)

[4. UI Design 63](#_Toc199085548)

[i. Register Account 63](#_Toc199085549)

[ii. Book Ride 64](#_Toc199085550)

[iii. Process Payment 65](#_Toc199085551)

[iv. Track Ride 68](#_Toc199085552)

[5. Classes Code 69](#_Toc199085553)

[i. Account 69](#_Toc199085554)

[ii. Address 69](#_Toc199085555)

[iii. Customer 70](#_Toc199085556)

[iv. Driver 70](#_Toc199085557)

[v. DriverLocation 71](#_Toc199085558)

[vi. Location 71](#_Toc199085559)

[vii. Payment 71](#_Toc199085560)

[viii. Ride 71](#_Toc199085561)

[6. Test Plan 72](#_Toc199085562)

[VI. Complete System Testing and Deploy the System 73](#_Toc199085563)

[1. Test Case 73](#_Toc199085564)

[i. Register Account 73](#_Toc199085565)

[ii. Book Ride 74](#_Toc199085566)

[iii. Process Payment 75](#_Toc199085567)

[iv. Track Driver 76](#_Toc199085568)

[2. Deployment Plan 77](#_Toc199085569)

[3. Demonstration Plan 78](#_Toc199085570)

[Conclusion 79](#_Toc199085571)

# Executive Summary

SmartRide is an urban ride-sharing business that connects customers with drivers using vehicles for transportation. It currently relies on manual operations, which is inefficient and prone to delays. The business goal is to digitally transform into an Online Ride-Sharing Platform (ORSP) to streamline operations, improve customer experience, and support scalability.

# Initial Activities

## System Vision Document

**Project Name:**

SmartRide – Online Ride-Sharing Platform (ORSP)

**Business Problem:**

SmartRide currently operates a manual ride-matching and payment system which results in long wait times, inefficient driver assignment, and slow payment processing. This leads to customer dissatisfaction, lost revenue, and an inability to scale the business.

**Business Objectives:**

* Provide a digital platform for booking and managing rides.
* Reduce wait times and improve ride matching efficiency.
* Enable real-time GPS tracking for better transparency.
* Facilitate secure online payments and digital receipts.
* Generate data-driven reports for management decision-making.
* Lay a foundation for scalable and feature-rich expansion.

**Proposed Solution:**

Develop a web and mobile-based Online Ride-Sharing Platform where:

* Customers can create accounts, book rides, track drivers, and make payments online.
* Drivers can manage their availability, receive bookings, and get optimized routes.
* Admins can monitor system performance and generate analytics.

**Major Features:**

* User registration and login (customers and drivers)
* Ride booking and automatic driver assignment.
* Real-time GPS tracking and ETA updates
* Online payment system with receipts
* Driver navigation and route optimization
* Admin reporting dashboard

**Scope:**

The initial release (MVP) will include core ride-sharing functionalities. Future enhancements like shared rides, loyalty programs, and dynamic pricing are out of scope for this phase but may be considered for future releases.

**Risks:**

* High demand could overload the system if scalability isn’t meticulously designed.
* GPS accuracy issues may impact customer trust.
* Payment integration challenges.
* Ensuring user data privacy and system security.

**Stakeholders:**

| **Stakeholder** | **Role / Interest** |
| --- | --- |
| **Customers** | **Request rides quickly and track drivers.** |
| **Drivers** | **Get ride requests, navigate efficiently, and get paid.** |
| **Business Managers** | **Monitor performance and improve service delivery.** |
| **Developers** | **Implement a scalable, reliable solution.** |
| **Investors** | **Ensure the platform generates ROI and supports growth.** |

**Tools used in creating diagram:**

* plantUML.
* Draw.io.
* Figma.

## Stakeholder Engagement

**Initial Stakeholder Interviews Conducted With:**

* SmartRide Business Owners
* Operations Manager
* Current Drivers
* Frequent Customers

**Purpose:**  
To understand:

* Pain points in the current system
* Expectations from the digital platform
* Feature priorities and critical use cases

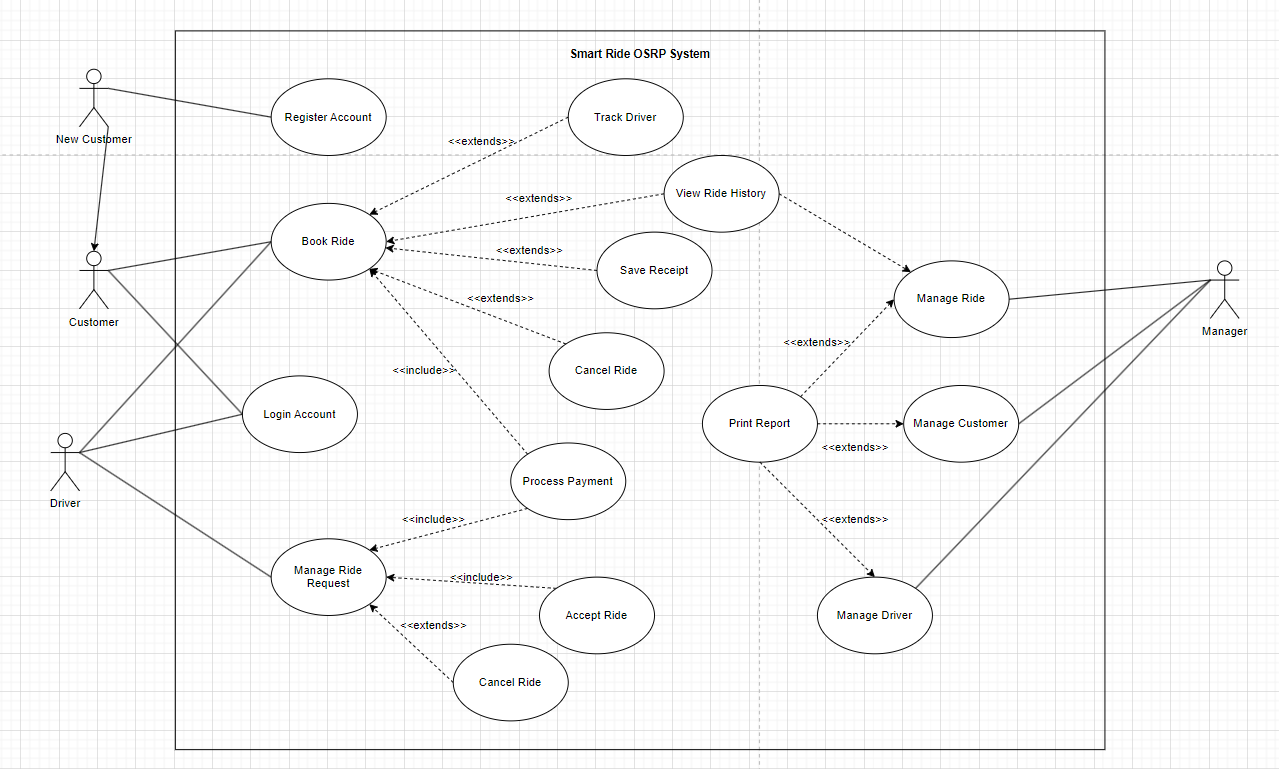
## Obtain Project Approval

**Actions Taken:**

* Shared System Vision Document with stakeholders.
* Conducted a review meeting with business owners.
* Presented timeline and resource estimates.
* Incorporated initial feedback and revised project goals.

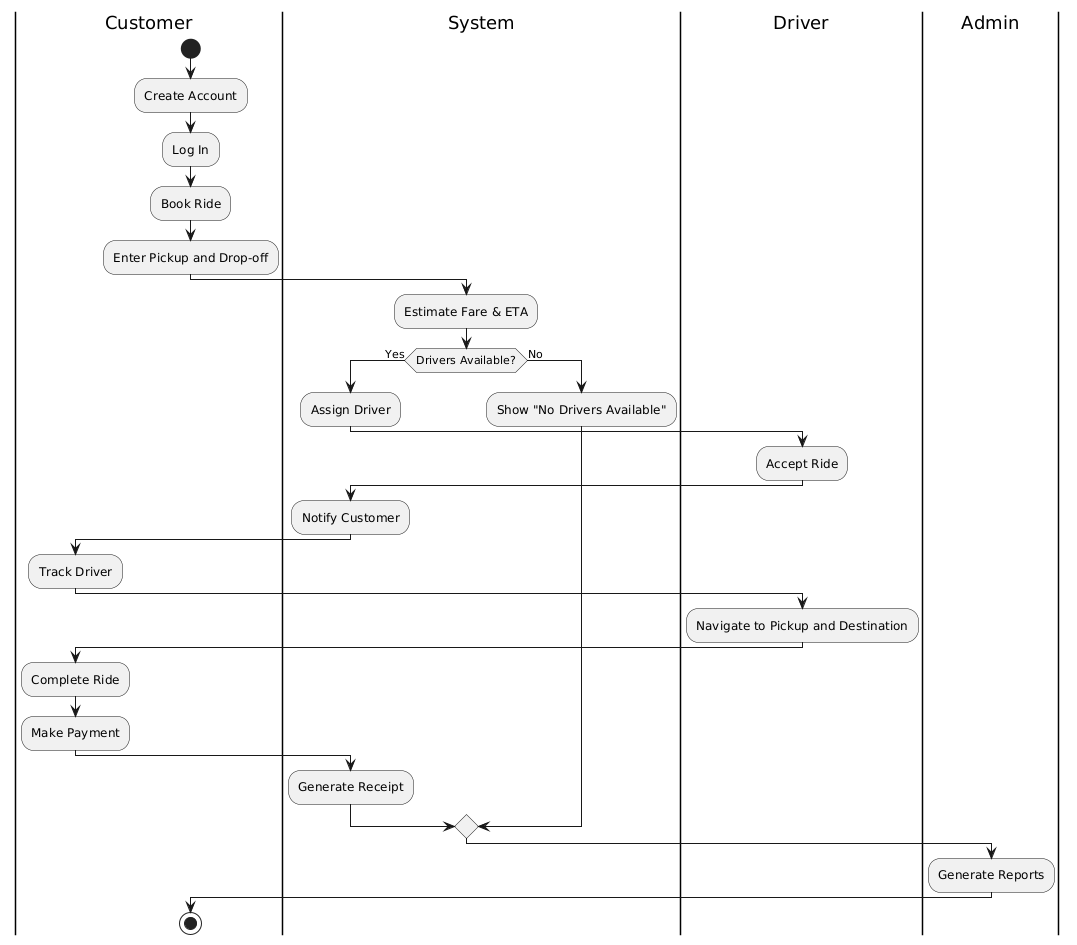
**Approval Outcome:** Project approved to proceed to planning and requirement gathering phases.

## Business Modeling / Requirements

****

Scoped business use case: Book Ride**A flowchart of a truck driver

AI-generated content may be incorrect.**

1. Business Processes / Flowchart of Requirements

## List of Requirements

**Functional Requirements:**

* Users (drivers/customers) can register and log in.
* Customers can request a ride by entering pickup and drop-off locations.
* The system shows nearby available drivers and assigns the nearest one.
* Customers can see ETA and track driver in real time.
* Drivers get navigation instructions to customer pickup and destination.
* Users can make online payments and receive receipts.
* Admins can generate reports on ride demand, peak hours, and operational stats.

**Non-Functional Requirements:**

* The system must manage high user loads during peak hours with low latency.
* Should support city-wide expansion and increase the user base.
* User data and payments must be protected using encryption and secure protocols.
* Interfaces should be user-friendly for drivers and customers alike.
* The service must be available 99.9% of the time.
* Modular architecture to support future enhancements.

# Plan your project.

## Project Scope

**In-Scope (MVP – Minimum Viable Product):**

* User Registration and Authentication (Customer, Driver, Admin)
* Ride Booking: Pickup/Drop-off input, fare estimation
* Real-time Driver Matching
* GPS Tracking & ETA Updates
* Online Payment Integration
* Driver Navigation
* Admin Reports and Ride Analytics

**Out-of-Scope (Future Enhancements):**

* Shared Rides (Carpooling)
* Discounts, Coupons, Loyalty Programs
* Ratings & Reviews
* Multi-language support
* Inter-city rides

## Project Objectives

* Digitize and streamline the ride-booking process.
* Improve customer satisfaction with faster and more reliable service.
* Enable secure, real-time payment and ride tracking.
* Provide business intelligence through data reporting.
* Lay the groundwork for future feature expansion.

## Project Phases and Timeline

| **Description** | **Duration** | **Estimated Timeframe** |
| --- | --- | --- |
| **Initial Activities & Vision** | **1 week** | **Week 1** |
| **Project Planning** | **1 week** | **Week 2** |
| **Requirements Gathering & Analysis** | **2 weeks** | **Weeks 3–4** |
| **System Design (Architecture, UI, DB)** | **2 weeks** | **Weeks 5–6** |
| **Build, Unit Test & Integrate Components** | **4 weeks** | **Weeks 7–10** |
| **System Testing & Deployment** | **2 weeks** | **Weeks 11–12** |

# Team Roles and Responsibilities

| **Role** | **Responsibilities** |
| --- | --- |
| **Project Manager** | **Oversees planning, progress, and resource allocation** |
| **Business Analyst** | **Gathers and models requirements, interfaces with stakeholders** |
| **UX/UI Designer** | **Designs intuitive interfaces for web and mobile** |
| **Backend Developer** | **Implements server-side logic and APIs** |
| **Frontend Developer** | **Develops responsive client interfaces** |
| **Mobile Developer** | **Builds mobile app versions (iOS/Android)** |
| **QA Engineer** | **Performs functional, integration, and system testing** |
| **DevOps Engineer** | **Sets up CI/CD pipelines, deployment, and infrastructure support** |

## Tools and Technologies

| **Area** | **Tools / Technologies** |
| --- | --- |
| **Project Management** | **Jira / Trello / Microsoft Project** |
| **Requirements Modeling** | **PlantUML, Lucidchart, Visual Paradigm** |
| **Version Control** | **Git, GitHub/GitLab** |
| **Development Stack** | **ASP.NET Core (Backend), React/Flutter (UI)** |
| **Database** | **SQL Server / PostgreSQL** |
| **Testing Tools** | **Postman, Selenium, xUnit** |
| **Deployment** | **Docker, Azure / AWS / Heroku** |
| **Communication** | **Slack, Microsoft Teams, Email** |

## Risk Management Plan

| **Risk** | **Mitigation Strategy** |
| --- | --- |
| **Feature creep** | **Lock MVP scope; introduce change request process** |
| **Resource unavailability** | **Assign backups or cross-train critical roles** |
| **Delay in third-party integrations (e.g., payment, GPS)** | **Use mocks initially; communicate early with vendors** |
| **System scalability problems** | **Design for modularity and cloud scaling from the start** |
| **Security and privacy issues** | **Use encryption, secure APIs, and follow best practices** |

## Communication Plan

* Weekly Status Meetings: Project Manager + Team Leads
* Stakeholder Demos: At the end of major milestones (P3, P4, P5)
* Issue Tracking: Through Jira/Trello
* Documentation: Stored in a shared repository (e.g., Notion, Confluence)

# Discovery and Understanding the Details

## System Narrative

**Booking a Ride with SmartRide**

A customer opens the SmartRide app and logs in. They enter their pickup and drop-off locations to request a ride. The system calculates the estimated fare and ETA based on traffic and distance, then displays this information to the customer.

The customer confirms the booking. The system automatically finds the nearest available driver and notifies them. The driver accepts the ride and heads to the pickup location. The customer can track the driver’s approach in real time.

Once the driver arrives, they begin the trip. After reaching the destination, the system calculates the final fare and charges the customer through their saved payment method. A digital receipt is generated, and both the customer and the driver can view it in their trip history. The ride details are logged for future reporting and analytics.

**Alternate scenario** – No drivers available

If there are no available drivers nearby when a customer requests a ride, the system notifies the customer and suggests trying again after some time.

## Actors and their goals

| Actor | Description | Primary Goals |
| --- | --- | --- |
| Customer | A city resident or visitor who uses SmartRide to book rides. | - Register and log in- Book a ride quickly and easily- Track driver in real time- Pay online securely- View trip history and receipts |
| Driver | A contracted individual who provides transportation using a car or motorbike. | - Register and verify driving credentials- Receive and accept ride requests- Navigate to locations- View trip and earnings history |
| Admin | A SmartRide staff member managing operations and overseeing system performance. | - Monitor customer and driver activity- Generate reports and insights- Ensure smooth operation and service quality |
| System (ORSP) | The automated online platform managing all ride-sharing operations. | - Match customers with drivers efficiently- Process payments- Maintain data integrity and security- Provide real-time tracking and notifications |

## List of Events

| **Event** | **Trigger** | **Source** | **Use Case** | **System Response** | **Destination** |
| --- | --- | --- | --- | --- | --- |
| User registration | New user submits registration form | Customer/Driver | Register Account | Validate input, create user account, send confirmation | Customer/Driver |
| User login | User submits login credentials | Customer/Driver | Log In | Authenticate user, start session | Customer/Driver |
| Ride request submitted | Customer inputs pickup and destination | Customer | Book Ride | Estimate fare/ETA, search for available driver | System |
| Driver assigned to ride | System finds closest available driver | System | Assign Driver | Notify driver and customer, update ride status | Driver and Customer |
| Driver accepts ride | Driver accepts incoming ride request | Driver | Accept Ride | Confirm assignment, start navigation | Customer |
| Ride begins | Driver marks arrival and starts ride | Driver | Start Ride | Update ride status to “In Progress” | System and Customer |
| Ride ends | Driver marks trip as completed | Driver | Complete Ride | Finalize fare, initiate payment | Customer and System |
| Payment processed | Ride is completed | System | Process Payment | Charge customer, update payment status, send receipt | Customer |
| Ride cancellation | User cancels ride before start | Customer/Driver | Cancel Ride | Update ride status, notify counterpart | Driver/Customer |
| No drivers available | System can't find driver in time | System | Book Ride | Notify customer, log unfulfilled request | Customer |
| Report generation | Admin requests usage reports | Admin | Generate Report | Fetch ride and system data, compile report | Admin |

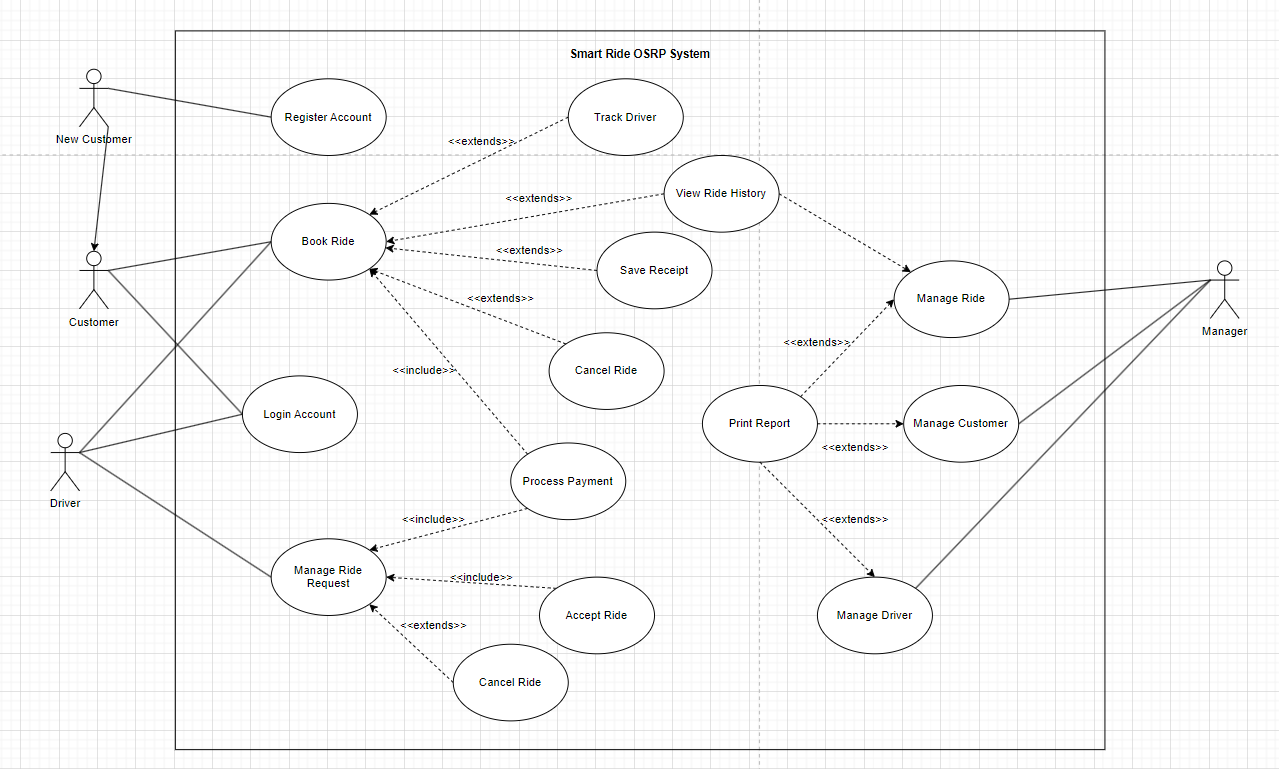
## List of Actors

* **Customer**: A person who uses the SmartRide platform to book rides. They can be a city resident or visitor.
* **Driver:** An individual who provides transportation services through SmartRide using their own car or motorbike.
* **Admin:** A staff member of SmartRide who manages the operations, monitors system performance, and generates reports.
* **System (ORSP):** The automated online platform that handles ride bookings, driver assignments, payments, and real-time tracking.
* **Payment Gateway:** Third-party system that processes payments from customers and drivers. It manages all transactions securely.
* **GPS Service:**  A third-party service responsible for providing location tracking and route optimization for both drivers and customers.

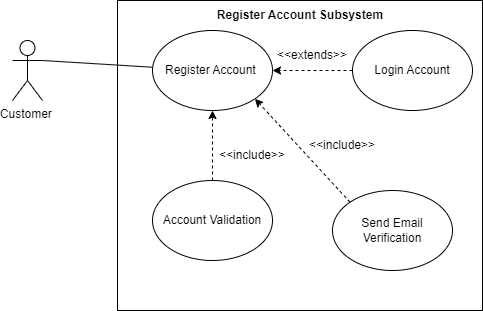
## List of Use Cases

* **Register Account:** Users create an account by providing personal details (e.g., name, email, phone number) and credentials.
* **Log In:** Customer Users log into the system with their credentials to access the platform.
* **Book Ride:** Customer inputs pickup and drop-off locations, receives fare estimate, and confirms booking.
* **Search for Available Driver**: System searches for available drivers based on the customer's location and request.
* **Accept Ride Request**: A driver receives and accepts a ride request from a customer.
* **Start Ride:** begins the ride by marking the start of the trip once they reach the customer’s location.
* **Complete Ride:** marks the trip as completed once they reach the destination, and fare is calculated.
* **Process Payment:** Customer’s payment is processed after the ride is completion.
* **Cancel Ride:** Either the customer or driver can cancel the ride before it begins.
* **View Ride History:** Users (customer, driver, or admin) can view historical ride data, including completed rides and earnings.
* **Generate Reports**: Admin generates reports on system usage, ride data, and financials to monitor performance and trends.
* **Track Driver:** Customer can track their assigned driver in real time via GPS integration.
* **Update Account Information:** Users can update personal information, including phone number, payment method, etc.
* **Manage System Configuration:** Admin manages the settings and configurations for the platform, such as service areas or driver eligibility.

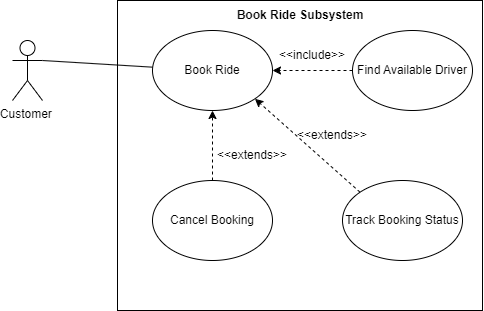
## Use Case Diagram

****

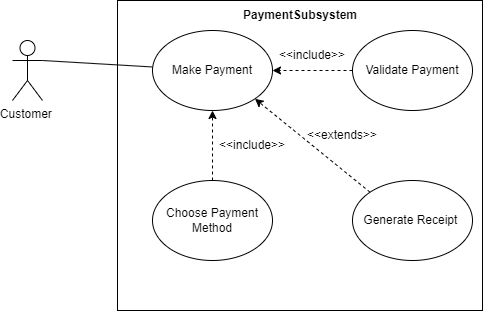
### Register Account Subsystem:



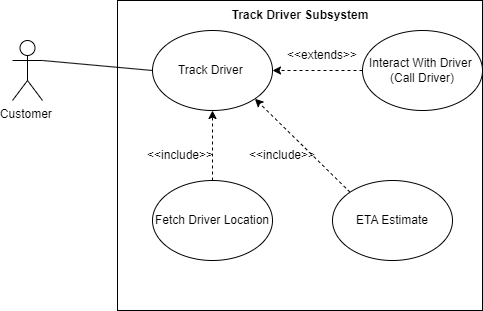
### Book Ride Subsystem:



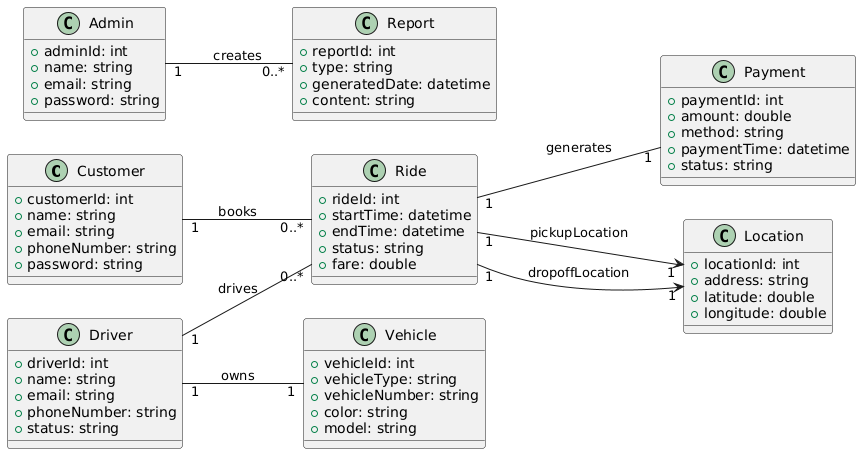
### Process Payment Subsystem:



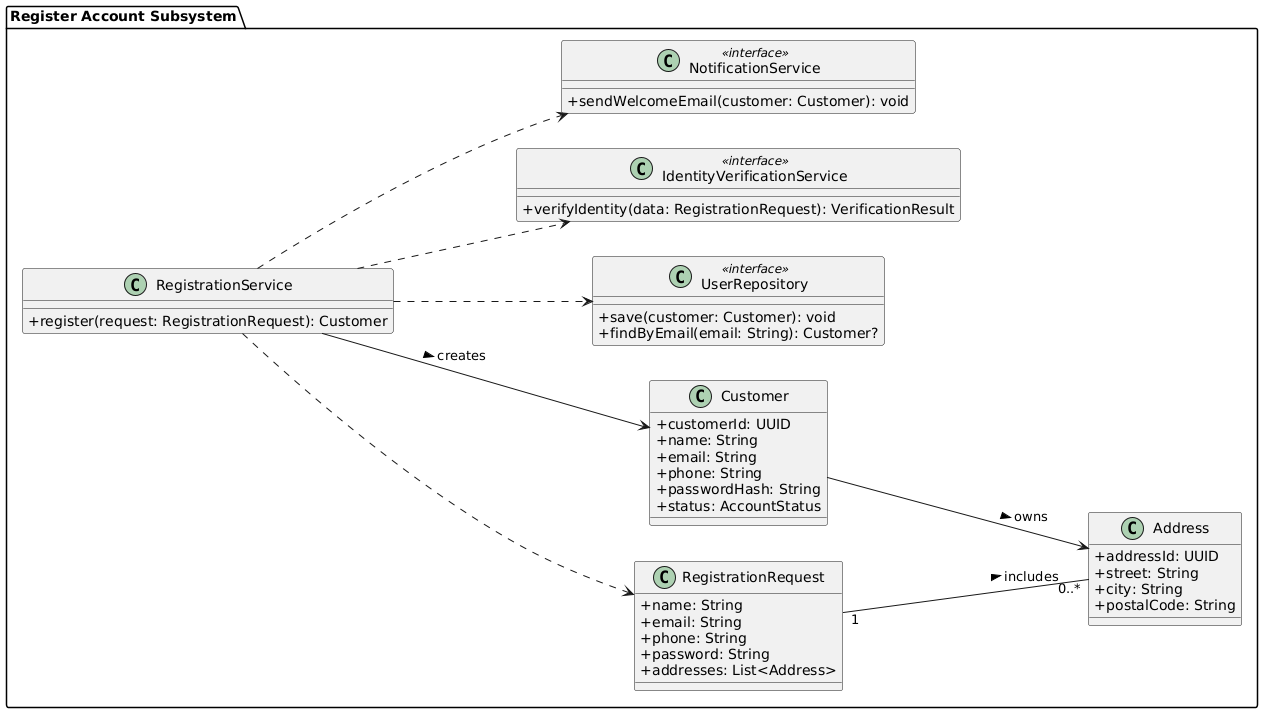
### Track Driver Subsystem:



## Domain Class Model



### Register Account Subsystem:

****

### Book Ride Subsystem:

A screenshot of a computer

AI-generated content may be incorrect.

### Process Payment Subsystem:

A screenshot of a computer

AI-generated content may be incorrect.

### Track Driver Subsystem:

A screenshot of a computer

AI-generated content may be incorrect.

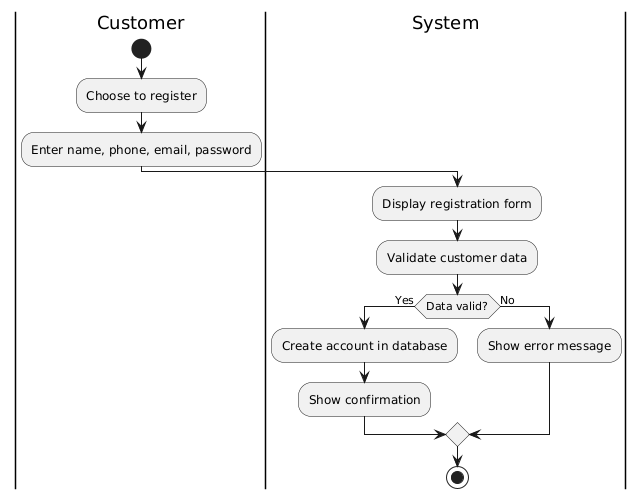
## Use Case Descriptions

### Use Case: Register Account

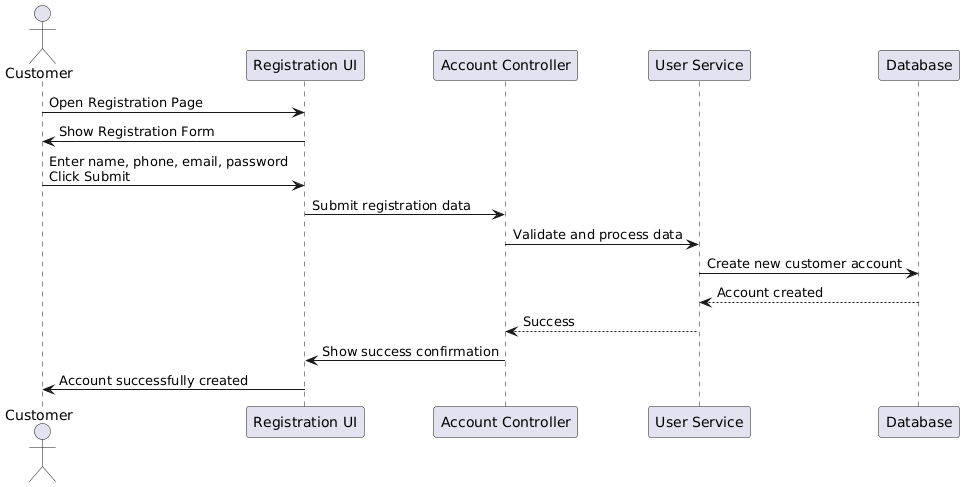
#### Use Case Description

| Use case name: | Register Account |
| --- | --- |
| Scenario: | Customer registers for a SmartRide account. |
| Triggering event: | A new customer wants to use the ride booking service. |
| Brief description: | Customer provides personal information and login details to create an account. |
| Actors: | Customer |
| Related use cases: | Login, Book Ride |
| Stakeholders: | Customer Support, Marketing |
| Preconditions: | Registration service must be available. |
| Postconditions: | Account is created and stored in the system. |
| Flow of activities: |  |
| Actor | System |
| 1. Customer chooses to register. | 1.1 System displays registration form. |
| 2. Customer fills in name, phone, email, and password. | 2.1 System validates input fields. |
| 3. Customer submits registration. | 2.2 System creates new accounts and confirms registration. |
| Exception conditions: | 2.1 Missing or invalid data. Email already exists. |

#### Activity Diagram:



#### Sequence Diagram:

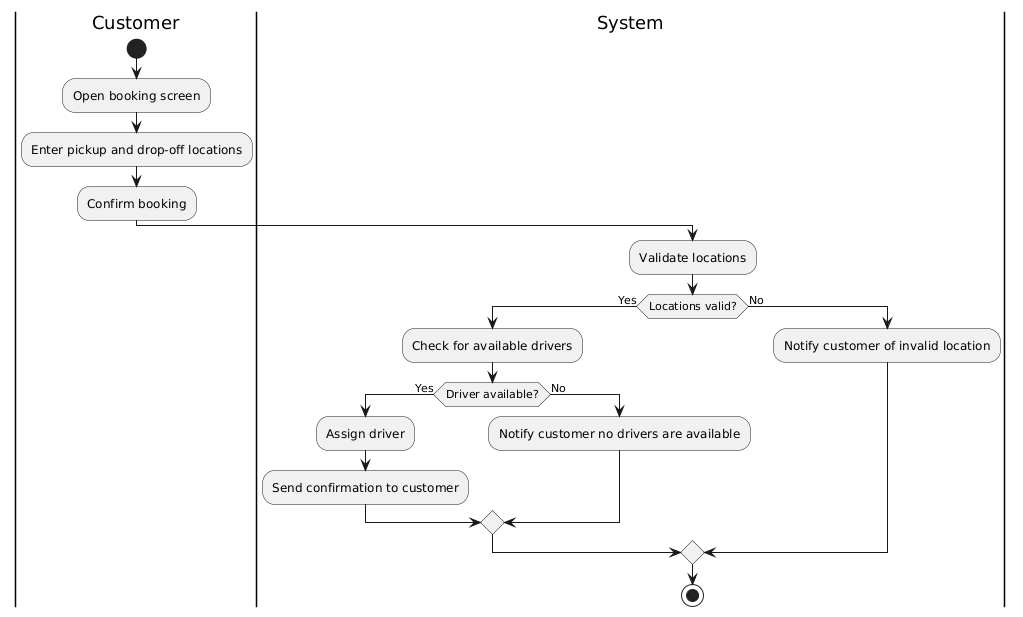


### Use Case 2: Book ride

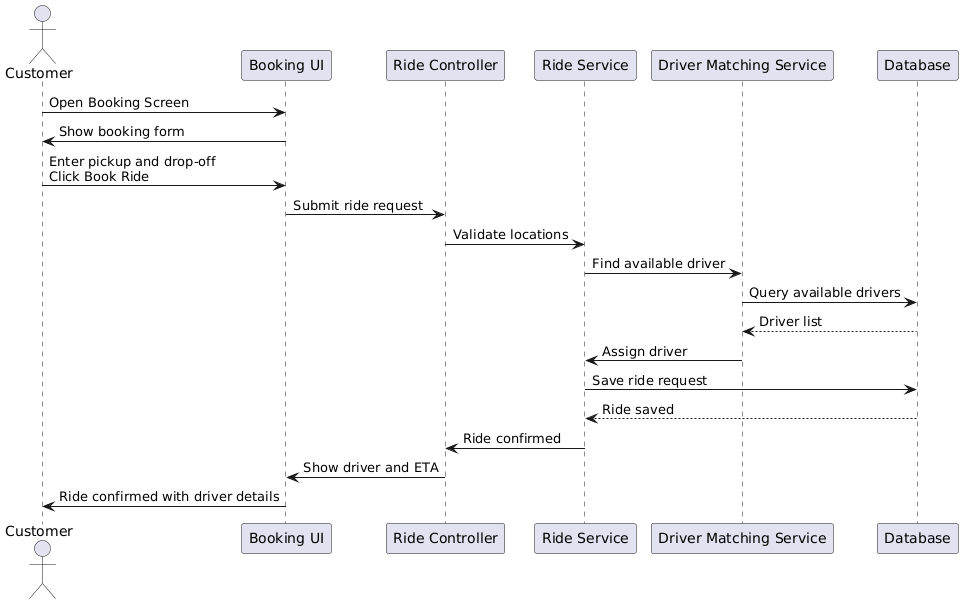
#### Use case description:

| Use case name: | Book ride |
| --- | --- |
| Scenario: | Customer books a ride via the mobile app or website. |
| Triggering event: | Customer opens the booking screen and enters trip details. |
| Brief description: | The system processes the ride request and assigns an available driver. |
| Actors: | Customer, System, Driver |
| Related use cases: | Track Driver, Make Payment |
| Stakeholders: | Customers, Operations Team |
| Preconditions: | User must be logged in. Location services must be available. |
| Postconditions: | Ride is booked and driver is notified. |
| Flow of activities: |  |
| Actor | System |
| 1. Customer enters pickup and dropoff location. | 1.1 System verifies location validity. |
| 2. Customer confirms ride. | 1.2 System matches with available driver.  1.3 System notifies driver and confirms booking to customer. |
| Exception conditions: | 1.1 No drivers available.  1.2 Invalid location. |

#### Activity Diagram



#### Sequence Diagram

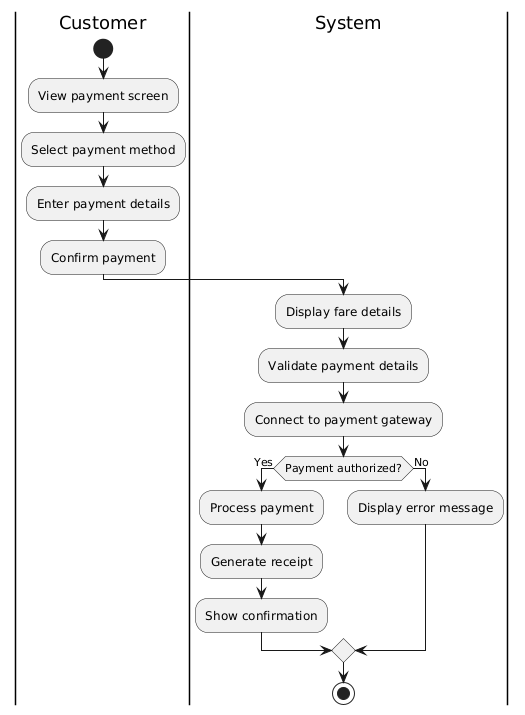


### Use Case 3: Process payment

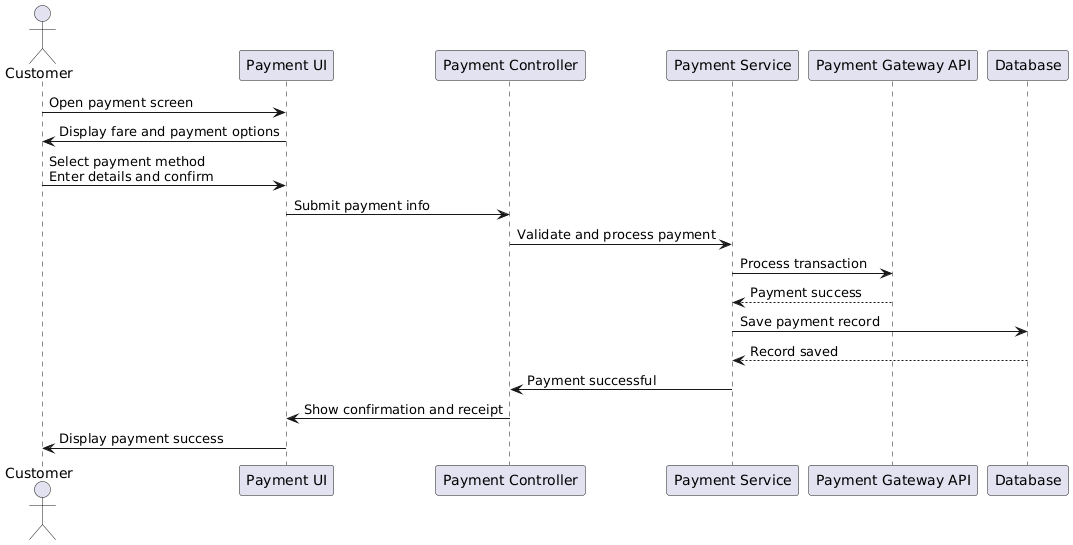
#### Use case description:

| Use case name: | Process payment |
| --- | --- |
| Scenario: | Customer pays for a completed ride. |
| Triggering event: | Ride is completed and fare is calculated. |
| Brief description: | Customer chooses a payment method and the system processes the transaction. |
| Actors: | Customer, Payment Gateway |
| Related use cases: | Book Ride |
| Stakeholders: | Customers, Finance Department |
| Preconditions: | Ride must be completed. Payment gateway must be operational. |
| Postconditions: | Payment is successful and receipt is issued. |
| Flow of activities: |  |
| Actor | System |
| 1. Customer chooses to pay. | 1.1 System displays fare and payment options. |
| 2. Customer selects payment method and confirms. | 1.2 System connects to payment gateway.  1.3 System processes payment.  1.4 System issues receipt. |
| Exception conditions: | 1.2 Invalid card or insufficient funds.  1.3 Payment gateway fails. |

#### Activity Diagram:



#### Sequence Diagram:

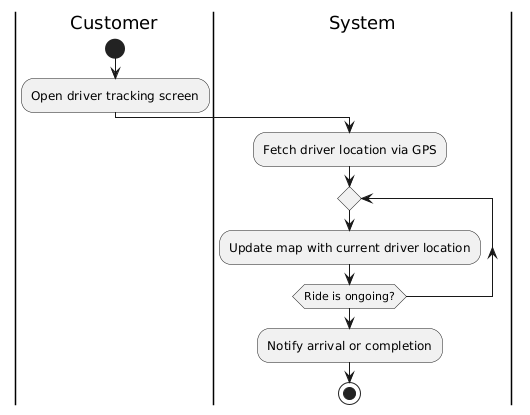


### Use Case 4: Track Driver

#### Use Case Description:

| Use case name: | Track Driver |
| --- | --- |
| Scenario: | Customer wants to monitor their ride’s progress. |
| Triggering event: | A ride has been accepted by a driver. |
| Brief description: | The system provides real-time GPS tracking of the driver’s location. |
| Actors: | Customer, Driver |
| Related use cases: | Book Ride |
| Stakeholders: | Customers, Operations Team |
| Preconditions: | A driver has accepted the ride. GPS service is available. |
| Postconditions: | Customer can view updated driver location until arrival. |
| Flow of activities: |  |
| Actor | System |
| 1. Customer opens track screen. | 1.1 System retrieves and displays current driver location. |
| 2. Customer refreshes or waits. | 1.2 System updates location in real-time using GPS. |
| Exception conditions: | 1.1 GPS service fails or is unavailable. |

#### Activity Diagram:



#### Sequence Diagram:PlantUML diagram

## Verifying use cases for Actor

### Verifying use cases for Customer

| **Data entity/domain class** | **C R U D** | **Verified use case** |
| --- | --- | --- |
| **Customer** | **Create** | **Create customer account** |
|  | **Read/report** | **Look up customerView ride history** |
|  | **Update** | **Update customer accountUpdate payment info** |
|  | **Delete** | **Deactivate customer account (soft delete)** |
| **Ride** | **Create** | **Book ride** |
|  | **Read** | **Track rideView ride history** |
|  | **Update** | **Cancel ride** |
|  | **Delete** | **Cancel ride (remove future scheduled ride)** |
| **Payment** | **Create** | **Make payment** |
|  | **Read** | **View payment history** |
|  | **Update** | **Update payment method** |
|  | **Delete** | **Remove payment method** |

### Verifying use cases for Driver

| **Data entity/domain class** | **C R U D** | **Verified use case** |
| --- | --- | --- |
| **Driver** | **Create** | **Register as driver** |
|  | **Read/report** | **View assigned ridesCheck payment status** |
|  | **Update** | **Update driver profileUpdate vehicle info** |
|  | **Delete** | **Deactivate driver account** |
| **Ride** | **Read** | **View assigned rides** |
|  | **Update** | **Accept/decline ride requestMark ride as complete** |
| **Vehicle** | **Create** | **Register vehicle** |
|  | **Read** | **View registered vehicle** |
|  | **Update** | **Update vehicle details** |
|  | **Delete** | **Remove vehicle (on deactivation)** |
| **Payment** | **Read** | **View earnings summaryView completed payments** |

### Verifying use case for Admin

| **Data entity/domain class** | **C R U D** | **Verified use case** |
| --- | --- | --- |
| **Customer** | **Read** | **View customer profileGenerate usage reports** |
|  | **Update** | **Process account adjustments** |
|  | **Delete** | **Archive/deactivate customer account** |
| **Driver** | **Read** | **View driver profile** |
|  | **Update** | **Approve/reject driver registrationModify driver details** |
|  | **Delete** | **Remove/deactivate driver account** |
| **Ride** | **Read** | **Monitor ongoing ridesView ride history** |
|  | **Update** | **Reassign ride (if needed)Cancel ride** |
| **Payment** | **Read** | **Review transaction historyGenerate financial reports** |
|  | **Update** | **Adjust/refund transactions** |
| **Report** | **Create** | **Generate ride usage reportsGenerate financial summaries** |
|  | **Read** | **View past generated reports** |
|  | **Update** | **Refresh or modify report filters** |
|  | **Delete** | **Remove outdated reports** |

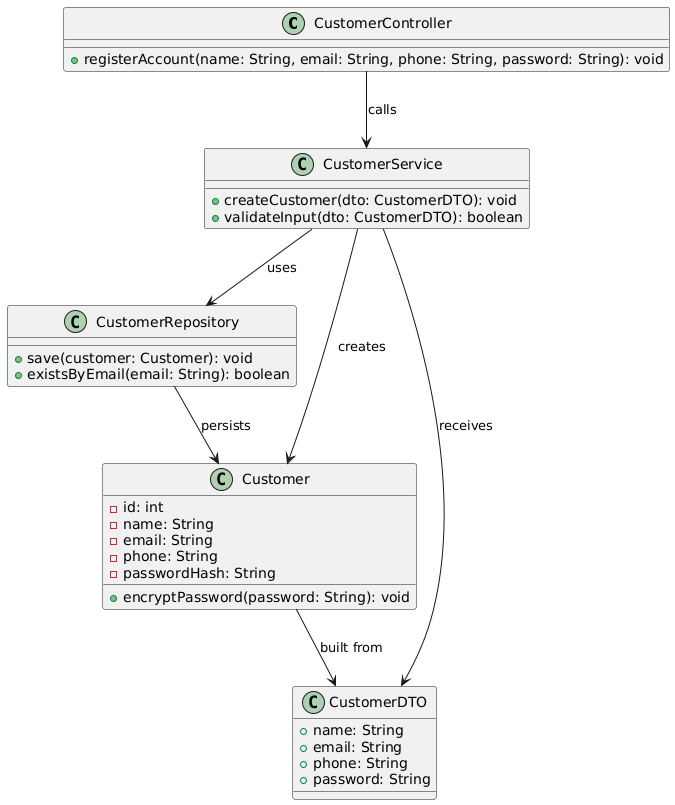
### Verifying use cases for Payment Gateway

| **Data entity/domain class** | **C R U D** | **Verified use case** |
| --- | --- | --- |
| **Payment** | **Create** | **Process customer payment** |
|  | **Read** | **Verify transaction status** |
|  | **Update** | **Retry failed transactionApply partial refunds** |
|  | **Delete** | **Cancel a pending/unverified payment** |

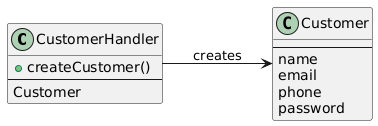
# Design System Components

## Design class for Register Account

### Domain Design Class



### Controller



### UI

A screenshot of a computer screen

AI-generated content may be incorrect.

### Data Access

A diagram of a customer account

AI-generated content may be incorrect.

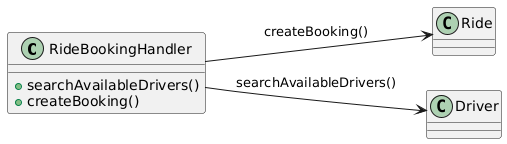
## Design class for Book Ride

### Domian Design Class

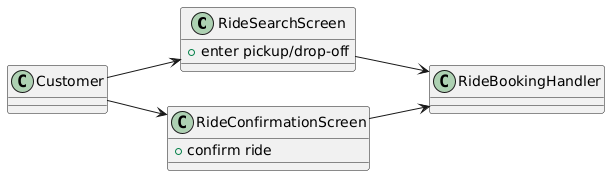
A diagram of a design class

AI-generated content may be incorrect.

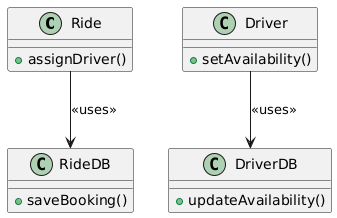
### Controller

****

### UI

****

### Data Access

****

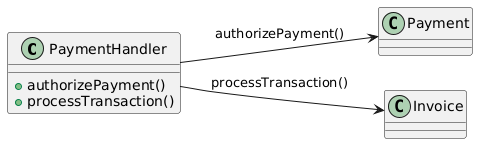
## Design class for Process Payment

### Domian Design Class

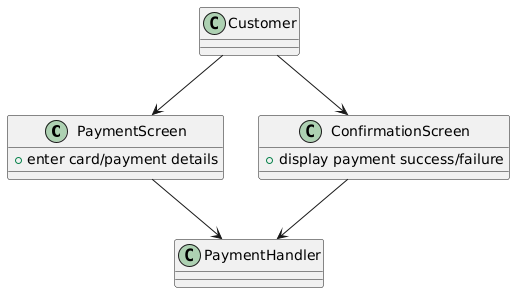
A diagram of a payment

AI-generated content may be incorrect.

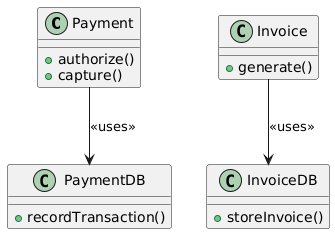
### Controller

****

### UI

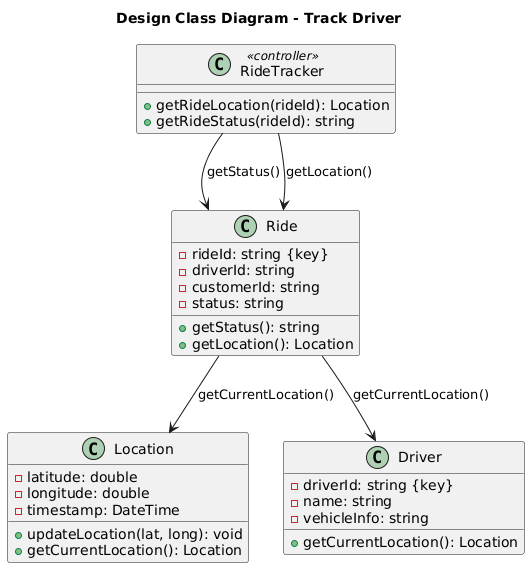
****

### Data Access

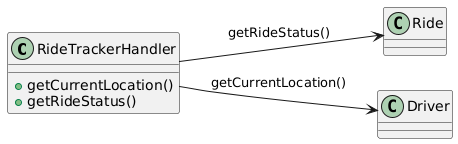
****

## Design class for Track Driver

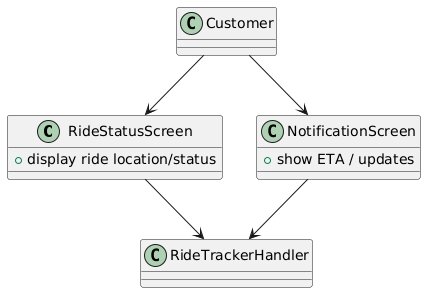
### Domain Design Class

****

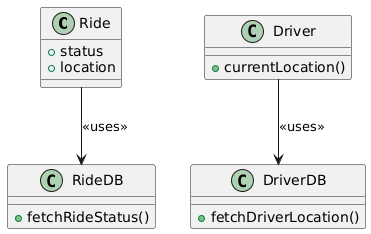
### Controller

****

### UI

****

### Data Access

****

# Build, Test, Integrated System Component

## Package Diagram for Subsystem

### Register Account Subsystem

A screenshot of a diagram

AI-generated content may be incorrect.

### Book Ride Subsystem

A screenshot of a computer

AI-generated content may be incorrect.

### Process Payment Subsystem

A diagram of a payment system

AI-generated content may be incorrect.

### Track Driver

A diagram of a track driver

AI-generated content may be incorrect.

## Database Design

### Customer

| **Column** | **Data Type** | **Constraints** |
| --- | --- | --- |
| **CustomerID** | **INT** | **PK, IDENTITY(1,1)** |
| **Name** | **NVARCHAR(100)** | **NOT NULL** |
| **PhoneNumber** | **NVARCHAR(20)** | **NOT NULL** |
| **Email** | **NVARCHAR(100)** | **NOT NULL, UNIQUE** |
| **CreatedDate** | **DATETIME** | **NOT NULL, DEFAULT GETDATE()** |

### Address

| **Column** | **Data Type** | **Constraints** |
| --- | --- | --- |
| **AddressID** | **INT** | **PK, IDENTITY(1,1)** |
| **CustomerID** | **INT** | **FK → Customer(CustomerID), ON DELETE CASCADE** |
| **AddressType** | **NVARCHAR(50)** | **NOT NULL** |
| **Street** | **NVARCHAR(200)** | **NOT NULL** |
| **City** | **NVARCHAR(100)** | **NOT NULL** |
| **State** | **NVARCHAR(100)** | **NOT NULL** |
| **PostalCode** | **NVARCHAR(20)** | **NOT NULL** |

### Account

| **Column** | **Data Type** | **Constraints** |
| --- | --- | --- |
| **AccountID** | **INT** | **PK, IDENTITY(1,1)** |
| **CustomerID** | **INT** | **FK → Customer(CustomerID), ON DELETE CASCADE** |
| **AccountType** | **NVARCHAR(50)** | **NOT NULL** |
| **AccountNumber** | **NVARCHAR(100)** | **NOT NULL** |
| **ExpiryMonth** | **TINYINT** | **NULL** |
| **ExpiryYear** | **SMALLINT** | **NULL** |
| **CreatedDate** | **DATETIME** | **NOT NULL, DEFAULT GETDATE()** |

### Driver

| **Column** | **Data Type** | **Constraints** |
| --- | --- | --- |
| **DriverID** | **INT** | **PK, IDENTITY(1,1)** |
| **Name** | **NVARCHAR(100)** | **NOT NULL** |
| **PhoneNumber** | **NVARCHAR(20)** | **NOT NULL** |
| **VehicleType** | **NVARCHAR(50)** | **NOT NULL** |
| **VehicleNumber** | **NVARCHAR(20)** | **NOT NULL** |
| **Availability** | **BIT** | **NOT NULL, DEFAULT 1** |
| **CreatedDate** | **DATETIME** | **NOT NULL, DEFAULT GETDATE()** |

### Ride

| **Column** | **Data Type** | **Constraints** |
| --- | --- | --- |
| **RideID** | **INT** | **PK, IDENTITY(1,1)** |
| **CustomerID** | **INT** | **FK → Customer(CustomerID)** |
| **DriverID** | **INT** | **FK → Driver(DriverID), NULL until assigned** |
| **PickupAddress** | **NVARCHAR(200)** | **NOT NULL** |
| **DropoffAddress** | **NVARCHAR(200)** | **NOT NULL** |
| **RequestTime** | **DATETIME** | **NOT NULL, DEFAULT GETDATE()** |
| **StartTime** | **DATETIME** | **NULL** |
| **EndTime** | **DATETIME** | **NULL** |
| **Status** | **NVARCHAR(50)** | **NOT NULL, DEFAULT 'Requested'** |
| **EstimatedFare** | **DECIMAL(10,2)** | **NULL** |
| **ActualFare** | **DECIMAL(10,2)** | **NULL** |

### Payment

| **Column** | **Data Type** | **Constraints** |
| --- | --- | --- |
| **PaymentID** | **INT** | **PK, IDENTITY(1,1)** |
| **RideID** | **INT** | **FK → Ride(RideID), ON DELETE CASCADE** |
| **AccountID** | **INT** | **FK → Account(AccountID)** |
| **Amount** | **DECIMAL(10,2)** | **NOT NULL** |
| **PaymentMethod** | **NVARCHAR(50)** | **NOT NULL** |
| **TransactionID** | **NVARCHAR(100)** | **NOT NULL** |
| **PaymentTime** | **DATETIME** | **NOT NULL, DEFAULT GETDATE()** |
| **Status** | **NVARCHAR(50)** | **NOT NULL** |

### DriverLocation

| **Column** | **Data Type** | **Constraints** |
| --- | --- | --- |
| **LocationID** | **INT** | **PK, IDENTITY(1,1)** |
| **DriverID** | **INT** | **FK → Driver(DriverID)** |
| **RideID** | **INT** | **FK → Ride(RideID), ON DELETE CASCADE** |
| **Latitude** | **DECIMAL(9,6)** | **NOT NULL** |
| **Longitude** | **DECIMAL(9,6)** | **NOT NULL** |
| **Timestamp** | **DATETIME** | **NOT NULL, DEFAULT GETDATE()** |

## SQL Code

A screenshot of a computer program

AI-generated content may be incorrect.

A computer screen shot of a program

AI-generated content may be incorrect.

A computer screen shot of a computer program

AI-generated content may be incorrect.

## UI Design

### Register Account

A screenshot of a login form

AI-generated content may be incorrect.

### Book Ride

A screenshot of a phone

AI-generated content may be incorrect.

### Process Payment

#### Start page

A screen shot of a payment form

AI-generated content may be incorrect.

#### Loading page

A screenshot of a computer

AI-generated content may be incorrect.

#### Success Page

A screenshot of a payment

AI-generated content may be incorrect.

### Track Ride

A screenshot of a computer

AI-generated content may be incorrect.

## Classes Code

### Account

A computer code on a black background

AI-generated content may be incorrect.

### Address

A screen shot of a computer screen

AI-generated content may be incorrect.

### Customer

A screen shot of a computer program

AI-generated content may be incorrect.

### Driver

A computer screen with text and numbers

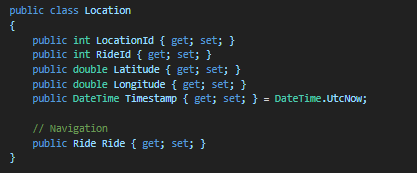
AI-generated content may be incorrect.

### DriverLocation

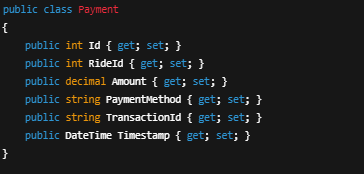
A screen shot of a computer code

AI-generated content may be incorrect.

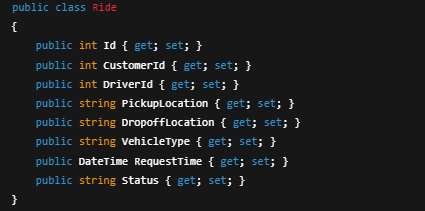
### Location

****

### Payment

****

### Ride

****

## Test Plan

| **Section** | **Description** |
| --- | --- |
| **System** | **SmartRide Online Ride-Sharing Platform** |
| **Scope** | **Covers 4 core use cases: Register Account, Book Ride, Make Payment, Track Ride** |
| **Testing Type** | **Functional, Integration, Usability, Acceptance Testing** |
| **Test Environment** | **ASP.NET Core MVC with SQL Server, hosted on local dev server and staging** |
| **Tools Used** | **Postman, Selenium (UI Testing), Visual Studio Test, SQL Profiler** |
| **Pass/Fail Criteria** | **Test passes if actual result = expected result, and system behaves as intended** |
| **Resources** | **1 QA Tester, 1 Developer, 1 Project Lead** |
| **Schedule** | **3 Days Test Cycle: Unit Test (Day 1), Integration Test (Day 2), UAT (Day 3)** |

# Complete System Testing and Deploy the System

## Test Case

### Register Account

| **Field** | **Value** |
| --- | --- |
| **Test Case ID** | **TC01** |
| **Use Case** | **Register Account** |
| **Description** | **Verify customer can register with valid details** |
| **Preconditions** | **Customer is not already registered** |
| **Test Steps** | **1. Navigate to registration page**  **2. Fill all required fields**  **3. Submit form** |
| **Expected Result** | **Customer account is created, data is saved to database, confirmation is shown** |
| **Postconditions** | **New Customer, Account, and optionally Address records created in the system** |

### Book Ride

| **Field** | **Value** |
| --- | --- |
| **Test Case ID** | **TC02** |
| **Use Case** | **Book Ride** |
| **Description** | **Verify ride booking flow and driver assignment** |
| **Preconditions** | **Customer is logged in and has valid location details** |
| **Test Steps** | **1. Enter pickup/dropoff location**  **2. Request ride**  **3. System assigns a driver** |
| **Expected Result** | **Ride is recorded with Requested or Confirmed status, driver is notified** |
| **Postconditions** | **New Ride record is created with reference to Customer and Driver (if assigned)** |

### Process Payment

| **Field** | **Value** |
| --- | --- |
| **Test Case ID** | **TC03** |
| **Use Case** | **Process Payment** |
| **Description** | **Verify that payment is processed after ride completion** |
| **Preconditions** | **Ride has Completed status, and valid payment method is configured** |
| **Test Steps** | **1. View completed ride**  **2. Choose payment method**  **3. Submit payment** |
| **Expected Result** | **Payment is recorded, transaction ID is generated, and status is Paid** |
| **Postconditions** | **New Payment record linked to Ride and Account is created** |

### Track Driver

| **Field** | **Value** |
| --- | --- |
| **Test Case ID** | **TC04** |
| **Use Case** | **Track Driver** |
| **Description** | **Ensure customer can track driver’s current location** |
| **Preconditions** | **Ride has been accepted and driver is en route** |
| **Test Steps** | **1. Open app during active ride**  **2. View driver's live location on map** |
| **Expected Result** | **System shows updated driver location in real-time** |
| **Postconditions** | **Location data is polled from DriverLocation and presented to the customer** |

## Deployment Plan

| **Step** | **Activity** | **Details** |
| --- | --- | --- |
| **1** | **Prepare Production Environment** | **Configure cloud server (e.g., Azure or AWS), SQL Server, SSL setup** |
| **2** | **Code Freeze and Final Build** | **Create final build from development branch (e.g., main)** |
| **3** | **Data Migration (if applicable)** | **Set up schema and seed basic data on production DB** |
| **4** | **Deploy Web Application** | **Use CI/CD tools like GitHub Actions or Azure Pipelines** |
| **5** | **Run Sanity Tests** | **Test basic functionality (login, registration, DB connection)** |
| **6** | **Monitor Logs and Metrics** | **Integrate with Application Insights or similar for real-time health** |
| **7** | **Rollback Plan** | **Retain backup and previous stable build for emergency restore** |

## Demonstration Plan

**Scenario: "Customer Registers, Books Ride, Pays and Tracks"**

| **Step** | **Action** | **System Response** |
| --- | --- | --- |
| **1** | **Go to SmartRide home page** | **Landing page appears with login/register** |
| **2** | **Register as new customer** | **Customer form is shown; submits data** |
| **3** | **Log in and select “Book Ride”** | **Ride booking screen appears** |
| **4** | **Enter pickup and dropoff locations** | **List of available drivers and ETA is shown** |
| **5** | **Confirm ride** | **System assigns a driver and starts tracking** |
| **6** | **View map to track driver** | **Driver location updates in real-time** |
| **7** | **Complete ride and choose payment method** | **Secure payment form appears** |
| **8** | **Pay and view receipt** | **Payment confirmation and downloadable receipt shown** |

# Conclusion

The SmartRide system is a functional and streamlined solution designed to address key challenges faced by the ride-sharing business. While the processes involved are not overly complex at this stage, the system effectively integrates essential features that significantly improve the customer and driver experience. By following structured design principles and using modern architectural patterns, the platform is scalable and ready to evolve with future enhancements such as loyalty programs, ride-sharing options, or advanced route optimization. The system architecture ensures that each component is clearly defined and easy to maintain.

Overall, the SmartRide Online Ride-Sharing Platform lays a solid foundation for digital transformation, helping the business reduce manual inefficiencies, increase customer satisfaction, and support future growth opportunities in the urban transport space.