

PROJECT REPORT

ON

**ISE TASK-2**

Comprehensive System Design and Agile Implementation for a Ride-Hailing Application

Submitted to

NMAM Institute of Technology, Nitte

(Off-Campus Centre, Nitte Deemed to be University, Nitte – 576110, Karnataka, India)

On partial fulfilment of the requirements for the award of the

Degree of Bachelor of Technology

In

INFORMATION SCIENCE AND ENGINEERING

By

Dhanush D Shetty

USN: NNM23IS048

Under the guidance of

Dr. JASON ELROY MARTIS , Associate Professor,

Department of Information Science and Technology,

NMAM Institute of Technology, Nitte, Karnataka,

India

ABSTRACT

Mobile technology lets ride share apps connect urban passengers directly with drivers through their digital platform. These apps use location tracking to pair riders with drivers in real-time and perform calculations to offer better automobile transport than regular taxi limos.

A ride-hailing app does basic operations including user sign-up and login while finding passengers and drivers near each other plus setting prices dynamically and rating passengers and drivers. The new systems improve user comfort by letting them see their trip details and by helping them split costs while offering plans to book ahead and integration among various transport modes.

The apps need a dependable foundation for managing many users at the same time while handling real-time processing and security features for payments along with their expandable service systems. Both driver and passenger apps connect to cloud services through APIs and this system heavily depends on processing spatial data.

Ride-hailing platforms succeed when they build an environment that helps all participants including riders, drivers, and platform managers to achieve their goals. Our platform needs user-friendly designs plus smart matching and routing systems that work alongside top-quality service tests.

The ride-hailing industry presents new challenges in processing many regional rules while taking care of drivers and Earth while working with transportation services and changing how people move. The solution needs regular improvements to technology plus updates to how companies run their business operations.

1.**INTRODUCTION**

Modern urban mobility began with ride-hailing applications which provided convenient and efficient and reasonably priced on-demand solutions for city transportation. Platform solutions ola and Lyft showcase the complete transformation power achieved from combining geolocation capabilities with real-time data analysis and payment integration into single-user experience flows. The development process requires detailed execution because it must unite advanced technology with official rules and upcoming user needs.

The document presents an Agile-designed ride-hailing application system which solves fundamental difficulties throughout its development process.

1. The system enables effortless connection with external third-party services such as payment gateways and mapping APIs for both functional needs and expanded capabilities.

2. A user-driven workflow approach should create logical processes for riders and drivers which span request bookings up to feedback operations.

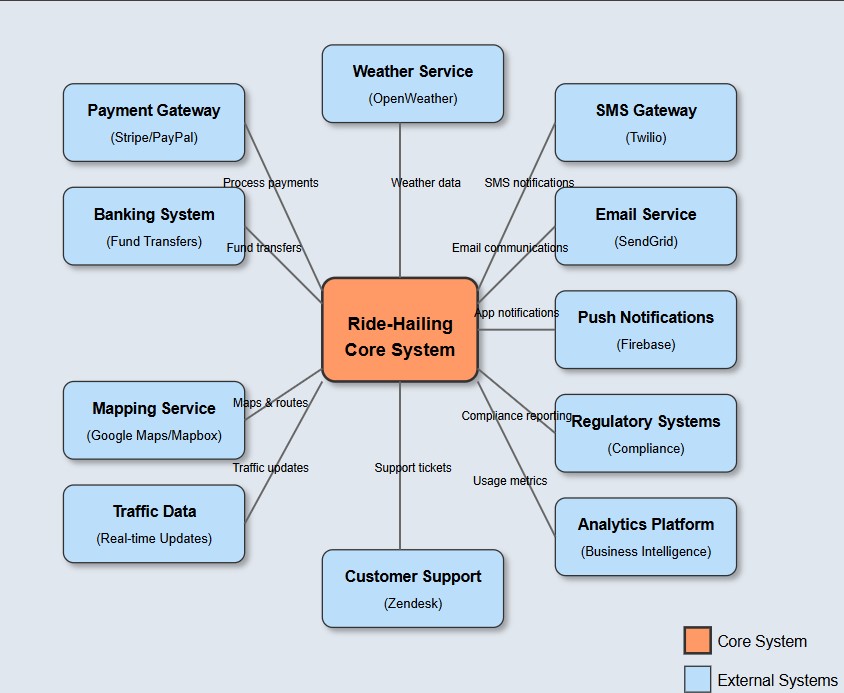
3. Agile Adaptability involves iterative development practices which enable refinements of features and risk prevention alongside delivering small incremental values.

**2.Context Model for Ride-Hailing App System Design**

The context model defines the boundaries of the ride-hailing core system and its interactions

with external third-party services. Below is a detailed breakdown of the external systemsand

their roles:



**2.1 Payment Processing**

• Payment Gateway (Stripe/PayPal):

The platform allows payment gateway systems to perform complete payment handling and managed subscription fee transactions.

o Purpose: Ensures secure and seamless transactions between riders, drivers, and the platform.

• Banking System:

The system enables financial fund transfers to driver accounts and validates all financial transaction records through interaction.

Through its Purpose system the solution executes payments to drivers and conducts currency operations for global deals.

**2.2 Communication Services**

SMS Gateway through Twilio serves two uses: first transmitting ride confirmation alerts and second sending OTP verification messages.

•Email Service (SendGrid):

Interaction: Delivers transactional emails (e.g., receipts, promotional offers).

•Push Notifications (Firebase):

The platform uses triggers for real-time app alerts which include driver arrival updates and ride requests among others.

**2.3 Mapping & Navigation**

* **Mapping Service (Google Maps/Mapbox)**:

o **Interaction**: Provides geolocation data, route optimization, and ETA calculations.

* **Traffic Data (Real-time Updates)**:
* **Interaction**: Integrates live traffic conditions to adjust routes dynamically.

**2.4 Regulatory & Compliance**

* **Regulatory Systems**:

o **Interaction**: Ensures adherence to local transport laws (e.g., driver licensing, fare caps).

* **Compliance Reporting**:

o **Interaction**: Automates generation of audit logs and regulatory reports.

**2.5 Analytics & Support**

* **Analytics Platform (BI Tools)**:

o **Interaction**: Aggregates usage metrics (e.g., ride frequency, peak hours) for business insights.

* **Customer Support (Zendesk)**:

o **Interaction**: Manages support tickets and user feedback through a centralized portal.

**2.6 Weather Service (OpenWeather)**

* **Interaction**: Fetches weather data to adjust surge pricing during adverse conditions (e.g., heavy rain).

**2.7 Key Interactions Overview**

* **Ride Booking**: Core system uses *Payment Gateway* for payment processing, *Mapping Service* for route calculation, and *SMS/Push Notifications* to notify users.
* **Driver Operations**: Relies on *Traffic Data* for route optimization and *Banking System* for payouts.
* **Compliance**: Automates reporting via *Regulatory Systems* and monitors data through *Analytics Platform*.

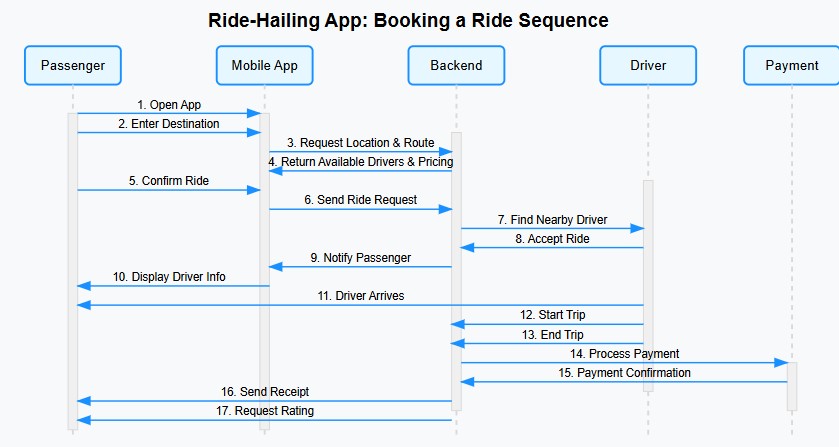
**2.8 Dependencies**

* The core system cannot function without critical services like *Payment Gateway* (revenue) and *Mapping Service* (navigation).
* Failures in *SMS Gateway* or *Push Notifications* would disrupt real-time user communication.

This model highlights the integration points essential for the app’s functionality while adhering to Agile principles (e.g., iterative development of modules like payment or mapping). The external systems are developed as loosely coupled components, enabling independent updates and scalability.

**3.Interaction Model for Ride-Hailing App: Booking a Ride Sequence**

This model outlines the step-by-step interactions between system components and users during the ride-booking process. The sequence aligns with Agile principles, enabling iterative development and modular testing of each interaction.



**3.1 Sequence Diagram Overview**

The booking process involves the following key actors and systems:

* **Passenger**: Initiates and completes the ride request.
* **Mobile App**: User interface for input and notifications.
* **Backend**: Central logic for processing requests, matching drivers, and managing data.
* **Driver**: Accepts and fulfills the ride request.
* **Payment System**: Handles transaction processing (via external gateways like Stripe/PayPal).

**3.2 Step-by-Step Interaction Breakdown**

1. **Open App** o **Actor**: Passenger
2. o **Action**: Launches the ride-hailing app.

* **System**: Mobile App initializes and fetches user data (e.g., saved

preferences, location).

1. **Enter Destination** **Actor**: Passenger
2. **Action**: Inputs pickup/drop-off locations.

o **System**: Mobile App sends this data to the Backend.

1. **Request Location & Route**

o **System**: Backend interacts with **Mapping Service (Google Maps/Mapbox)** to calculate routes and ETAs.

1. **Return Available Drivers & Pricing**
   * + - **System**: Backend identifies nearby drivers and calculates fare (incl. surge pricing if applicable).
       - **Mobile App**: Displays drivers and pricing to the Passenger.
2. **Confirm Ride**

o **Actor**: Passenger

o **Action**: Selects a ride option (e.g., UberX, Uber Black).

1. **Send Ride Request**

o **System**: Backend broadcasts the request to nearby Drivers via **Push Notifications (Firebase)**.

1. **Find Nearby Driver**

o **System**: Backend uses real-time location data to prioritize Drivers closest to the pickup point.

1. **Accept Ride** o **Actor**: Driver

o **Action**: Driver accepts the request via their app.

1. **Notify Passenger**

o **System**: Backend triggers **SMS (Twilio)** or **Push Notifications** to inform the Passenger of Driver details.

1. **Display Driver Info**

o **Mobile App**: Shows Driver name, vehicle details, license plate, and live location.

1. **Driver Arrives** o **Actor**: Driver o **Action**: Marks arrival in their app.

o **System**: Passenger receives an app notification.

1. **Start Trip** o **Actor**: Driver o **Action**: Begins the trip via their app. o **System**: Backend records trip start time and initiates live tracking.
2. **End Trip** o **Actor**: Driver o **Action**: Ends the trip at the destination.

o **System**: Backend calculates final fare and triggers payment.

1. **Process Payment**

o **System**: Backend charges the Passenger’s saved payment method via **Payment Gateway (Stripe/PayPal)**.

1. **Payment Confirmation** o **System**: Payment Gateway returns transaction status.

o **Mobile App**: Displays payment success/failure to Passenger and Driver.

1. **Send Receipt**

o **System**: Backend sends a receipt via **Email Service (SendGrid)** and in-app notification.

1. **Request Rating**

* **System**: Backend prompts Passenger and Driver to rate each other via **Push Notifications**.

**3.3 Key Dependencies & Agile Considerations**

 **Critical Interactions**:

* + - * + Payment processing (Step 14) depends on external gateways; failures here block revenue flow.
        + Real-time notifications (Steps 6, 9, 17) rely on **Firebase** and **Twilio**; redundant channels ensure reliability.

 **Agile Implementation**:

o Each step can be developed as a user story, enabling incremental testing (e.g., "Payment Integration Sprint"). o Loosely coupled systems (e.g., Mapping Service, SMS Gateway) allow parallel development and scalability.

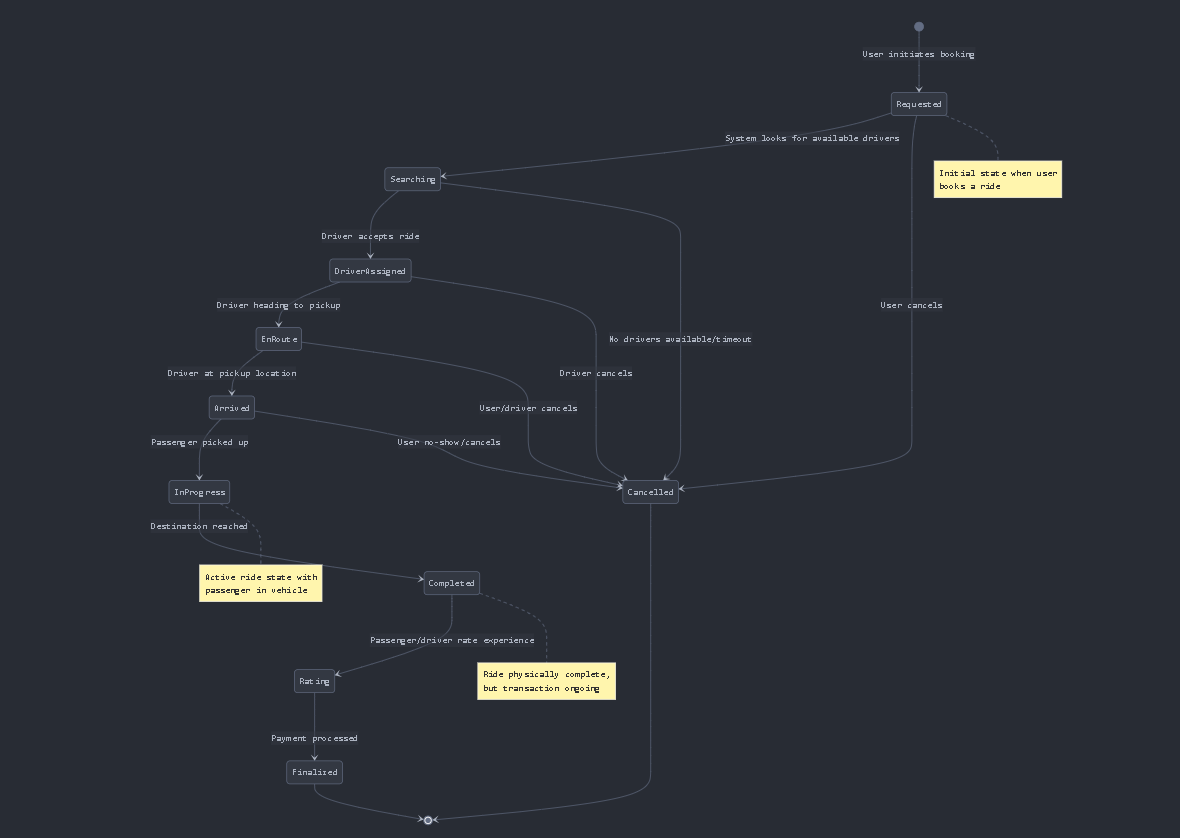
**4.Behavioral Model: Ride-Hailing System Interactions**

The section elaborates on the behavioral model by investigating the functions of main actors

and processes which interact with external systems throughout ride cycles. The framework

follows Agile methods while it supports modular breakdown and step-by-step development

processes.



**4.1.This state diagram illustrates the complete lifecycle of a ride from booking to completion.**

The diagram shows:

1.Initial States:

The system enters the requested state because a user starts the booking process.

The Available Driver Search System operates to identify suitable drivers for service.

2.Active States:

A driver receives guest requests at this stage.

EnRoute: Driver traveling to pickup location

The driver attains the designated pickup location.

During the InProgress state the passenger enters the vehicle while the ride continues.

3.Completion States:

The ride process terminates after reaching the journey destination.

Rating: Post-ride feedback collection

The process reaches completion after payment is processed while the ride achieves its complete state.

4.Cancellation Paths:

The process contains various points that lead to cancellation according to the illustration.

The ride can be canceled at any moment by either passenger or driver throughout the process before it finishes.

5.Transitions:

Clear paths between states representing user and driver actions

The transportation service follows a systematic procedure beginning with booking and continuing until execution and final payment completion.

**4.2 Agile Implementation Strategies**

1. **User Stories**:

*"As a passenger, I want to request a ride so I can reach my destination."*

*"As a driver, I need navigation guidance to efficiently reach pickup and drop-off points."*

2. **Sprint Breakdown**:

**Sprint 1**: Develop *Request Ride* and *Assign Driver* logic.

**Sprint 2**: Integrate **Payment Gateway** for end-to-end payment processing.

**Sprint 3**: Implement **Mapping Service** APIs for route optimization

**3 Dependencies & Risks**

 **Critical Dependencies**:

*Payment Gateway*: Revenue flow halts if integration fails.

*Mapping Service*: Incorrect ETAs or routes degrade user experience.

 **Mitigation**:

Use fallback payment methods (e.g., cached card details).

Implement redundant mapping providers (e.g., switch to Mapbox if Google Maps fails).

**5.Agile Development with Scrum**

Ride-Sharing App Scrum Development Plan

Sprint Structure and Key Deliverables

Sprint 0 (1 week): Project Setup

Deliverables:

* Development environment setup
* Architecture design documentation
* Product backlog creation
* Team onboarding
* Design system foundation

Sprint 1 (2 weeks): Core User Authentication

Deliverables:

* User registration and login functionality
* Users at both ends of the platform must create their profiles
* Basic navigation structure
* User authentication API
* Data schema implementation

Sprint 2 (2 weeks): Basic Ride Booking

Deliverables:

* Location selection mechanism
* Basic ride request functionality
* Driver availability system
* Simple ride matching algorithm
* Payment integration foundation

Sprint 3 (2 weeks): Driver Experience

Deliverables:

* Driver app interface
* Ride acceptance mechanism
* Navigation integration
* Driver status management
* Earnings tracking

Sprint 4 (2 weeks): Enhanced Passenger Experience

Deliverables:

* Ride tracking interface
* ETA calculations
* Ride history
* Favorite locations
* Driver rating system

Sprint 5 (2 weeks): Payment and Reviews

Deliverables:

* Complete payment processing
* Tipping functionality
* Comprehensive review system
* Receipt generation
* Payment dispute handling

Sprint 6 (2 weeks): Advanced Features

Deliverables:

* Ride scheduling
* Multiple stop capability
* Fare splitting
* Ride preferences
* Notifications system

Sprint 7 (2 weeks): Quality and Performance

Deliverables:

* Performance optimization
* Edge case handling
* Accessibility compliance
* Comprehensive testing
* Bug fixes

**6. Passenger Stories**

1. Ride Booking

* Story: As a passenger, I want to request a ride by setting my pickup location and destination so I can travel to my desired location.
* Definition of Done:
* Passenger can enter or select current location and destination
* App displays estimated fare and arrival time before confirmation
* Ride request is successfully sent to available drivers
* Passenger receives confirmation when a driver accepts
* All screens are responsive and follow design guidelines
* Feature passes all automated tests
* Product Owner approves functionality

2.Ride Tracking

* Story: As a passenger, I want to track my driver's location in real-time so I know when to expect them.
* Definition of Done:
* Map displays driver's current location and movement
* ETA is continuously updated
* Driver's information and vehicle details are visible
* Option to contact driver is available
* Feature works when app is in background
* Performance tested for minimal battery drain
* Tracking accuracy verified in multiple locations

3.Ride Payment

* Story: As a passenger, I want to pay for my ride seamlessly using my preferred payment method so I don't need to use cash.
* Definition of Done:
* Multiple payment methods can be saved and selected
* Automatic payment processing when ride completes
* Receipt is generated and accessible
* Tipping option is available
* Payment processing meets security standards
* Error handling for failed payments
* Transaction history is accessible

4.Ride History

* Story: As a passenger, I want to view my past rides so I can reuse routes and track my spending.
* Definition of Done:
* Complete list of past rides with details (date, time, route, cost)
* Option to repeat a previous ride
* Filtering and sorting capabilities
* Receipt access for each ride
* Data loaded efficiently with pagination
* Search functionality implemented
* Export option available

5.Driver Rating

* Story: As a passenger, I want to rate and review my driver after the ride so I can provide feedback on my experience.
* Definition of Done:
* Rating prompt appears after ride completion
* Star rating system (1-5) implemented
* Optional text feedback field available
* Pre-defined feedback categories available
* Anonymous feedback option
* Previous ratings viewable in ride history
* Ratings stored and linked to specific rides

6.Ride Scheduling

* Story: As a passenger, I want to schedule a ride in advance so I can ensure transportation for future needs.
* Definition of Done:
* Calendar/time picker for scheduling
* Confirmation of scheduled ride
* Reminder notifications
* Option to cancel or modify scheduled rides
* Driver assignment process starts before scheduled time
* Scheduled rides appear in upcoming section
* Fare estimate provided at scheduling time

**7. Driver Stories**

1.Ride Acceptance

* Story: As a driver, I want to receive and accept ride requests so I can provide service to passengers.
* Definition of Done:
* Driver receives notifications for nearby ride requests
* Request shows pickup location, destination, and estimated fare
* Timer for response decision
* Accept/decline functionality
* Acceptance rate tracking
* Clear navigation to pickup after acceptance
* Proper handling of simultaneous requests

2.Navigation

* Story: As a driver, I want turn-by-turn navigation to pickup locations and destinations so I can efficiently complete rides.
* Definition of Done:
* Integration with navigation services
* Voice-guided directions
* Alternative route suggestions
* Traffic information incorporated
* Option to use preferred navigation app
* Works in background mode
* Minimal battery and data usage

3.Earnings Tracking

* Story: As a driver, I want to track my earnings in real-time so I can monitor my income.
* Definition of Done:
* Daily, weekly, and monthly earnings summary
* Breakdown by ride
* Tips and bonuses clearly indicated
* Visualization of earnings trends
* Payment schedule information
* Tax reporting information
* Export functionality for financial records

4.Status Management

* Story: As a driver, I want to easily set my availability status so I can control when I receive ride requests.
* Definition of Done:
* Simple toggle for online/offline status
* Option to set availability duration
* Scheduled availability feature
* Break time functionality
* Status change confirmations
* Current status clearly displayed
* Automatic offline mode after extended inactivity

5.Passenger Rating

* Story: As a driver, I want to rate passengers after rides so I can provide feedback on my experience.
* Definition of Done:
* Rating prompt appears after ride completion
* Star rating system implemented
* Predefined issue categories available
* Text comment option
* Previous passenger ratings visible on new requests
* Block feature for problematic passengers
* Rating history accessible

**6.Route Optimization**

* **Story:** As a driver, I want to receive optimal route suggestions to maximize my earnings and minimize downtime.
* **Definition of Done:**
  + Heat map of high-demand areas
  + Suggestions based on historical data
  + Time-based recommendations
  + Event-based demand predictions
  + Opt-in/out of suggestions
  + Personalized based on driver preferences
  + Balance maintained across service area

**8. Scrum Ceremonies**

**1.Sprint Planning (Beginning of each sprint)**

* Review and prioritize backlog items
* Estimate user stories using story points
* Determine sprint capacity and commit to deliverables
* Break down stories into tasks
* Define clear acceptance criteria

**2.Daily Scrum (15 minutes each day)**

* Each team member shares:
  + What they did yesterday
  + What they plan to do today
  + Any impediments they're facing

**3.Sprint Review (End of each sprint)**

* Demonstrate completed user stories to stakeholders
* Gather feedback on functionality
* Assess what was completed vs. planned
* Update product backlog based on feedback

**4.Sprint Retrospective (After sprint review)**

* Discuss what went well
* Identify areas for improvement
* Agree on actionable items for the next sprint
* Track continuous improvement metrics

**Definition of Ready**

Before a user story can be accepted into a sprint, it must:

* Be clearly written from user perspective
* Have clear acceptance criteria
* Be independently valuable to users
* Be estimated by the development team
* Be small enough to complete in a single sprint
* Have all external dependencies identified
* Have necessary design assets prepared
* Have testability considerations addressed

**Team Structure**

* Product Owner: Prioritizes backlog, represents stakeholders
* Scrum Master: Facilitates process, removes impediments
* Development Team:
  + 2 Frontend Developers
  + 2 Backend Developers
  + 1 QA Engineer
  + 1 UI/UX Designer

**Tools**

* Version Control: GitHub
* Project Management: Jira
* Communication: Slack
* CI/CD: Jenkins
* Documentation: Confluence

**9.github:**

**10. Conclusion**

This report presents a holistic system design for a ride-hailing application,

integrating **contextual**, **interactional**, and **behavioral models** to define its architecture, workflows, and dependencies. By adopting **Agile methodologies**, the design emphasizes iterative development, modularity, and adaptability, ensuring alignment with real-world complexities and user needs. Key insights include:

* 1. **System Scalability**:
     + The integration of external systems (e.g., Stripe for payments, Google Maps for navigation) ensures scalability while maintaining loose coupling. This allows independent updates and minimizes downtime risks.
  2. **User-Centric Workflows**:
     + The interaction and behavioral models prioritize seamless user experiences, from ride booking to post-trip feedback. Real-time communication (SMS, push notifications) and dynamic pricing adjustments enhance reliability and transparency.
  3. **Agile Success**:
     + Sprint-based development, user stories, and retrospectives enabled incremental delivery and risk mitigation. Tools like GitHub facilitated collaboration, version control, and conflict resolution, demonstrating the value of structured teamwork.
  4. **Resilience & Compliance**:
     + The system accounts for edge cases (e.g., payment failures, cancellations) and regulatory requirements through automated compliance reporting and redundant service integrations.