**PROJECT REPORT**

**ON**

**System Modelling and Agile Software Development for a Ride-hailing App (Mini-Uber Design)**

**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**  
**Chirag Rai**  
**NNM23IS037**

**Under the guidance of**  
**Dr. Jason Elroy Martis, Associate Professor,**  
Department of Information Science and Technology,  
NMAM Institute of Technology, Nitte, Karnataka, India

{chiragrai2005@gmail.com, nnm23is037@nmamit.in}

**Keywords:** System Modelling, Agile Software Development, Scrum Methodology, Ride-Hailing App, UML Diagrams, Sprint Planning

**TABLE OF CONTENTS**

|  |  |
| --- | --- |
| Titles | Page Numbers |
| Abstract | 3 |
| Unit 1: System Modelling for Mini-Uber | 4-6 |
| 1.1 Context Model | 4-5 |
| 1.2 Interaction Model | 5-6 |
| 1.3 Behavioural Model | 6 |
|  |  |
| Unit 2: Scrum Development Plan for Mini-Uber | 7-12 |
| 1.1 Sprint Structure with Key Deliverables | 7-8 |
| 1.2 User Stories | 9 |
| 1.3 Definition of Done (DoD) for Each User Story | 10-12 |
|  |  |
| Conclusion | 13 |
| References | 13 |

**Abstract**

This report explores the design and development of a simple ride-hailing application, which we call Mini-Uber as an example, by integrating system modelling techniques with an Agile-based development approach. The goal is to construct a structured system architecture while maintaining flexibility in development. The report is divided into two major sections:

**System models** that represent key functional aspects of the app

An **Agile plan** that outlines an iterative development strategy.

The system modelling part includes three key UML diagrams.

A **Context Model** highlights the app’s interaction with external systems such as payment processors and location services.

An **Interaction Model,** depicted through a sequence diagram, provides a step-by-step visualization of the ride-booking process.

A **Behavioural Model** in the form of a state diagram illustrates the different stages of a ride, from request initiation to payment processing.

These diagrams serve as a blueprint for understanding how different components of the system interact.

The Agile methodology is implemented using the **Scrum framework**, ensuring a structured yet adaptable development cycle.

The report includes a **Sprint plan**, defining key deliverables at each stage, and a **backlog with user stories** for both passengers and drivers, covering various features of the app.

A well-defined **Definition of Done (DoD)** ensures that each development task meets quality and functionality standards.

By combining structured system modelling with Agile-driven development, this approach provides a scalable, adaptable, and user-focused blueprint for building a ride-hailing application. The models and methodologies detailed in this report serve as a roadmap for a well-organized and efficiently managed development process.

**Unit 1: System Modelling for Mini-Uber**

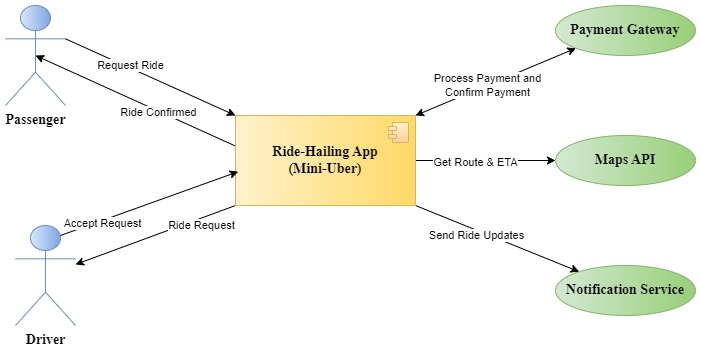
System modelling is a technique used to represent the structure and behavior of a system in a simplified manner. It helps developers and stakeholders understand how different components interact and function together. These models serve as a blueprint for system design, aiding in documentation, analysis, and problem-solving.

The types of system models discussed for the ride hailing service are:

* **Context Models** – Illustrate the system’s interactions with external entities such as users, services, and databases.
* **Interaction Models** – Depict the communication between system components, often using sequence diagrams.
* **Behavioural Models** – Show how the system responds to different events or transitions, typically represented with state diagrams.

**1.1 Context Model**

The **Contextual model** defines the external entities interacting with the system and their relationships. It provides a high-level view of how the system connects with users and external services.

**

*Fig. 1.1 UML diagram of Contextual Model*

The above diagram shows how different components work together to enable the ride-hailing service.

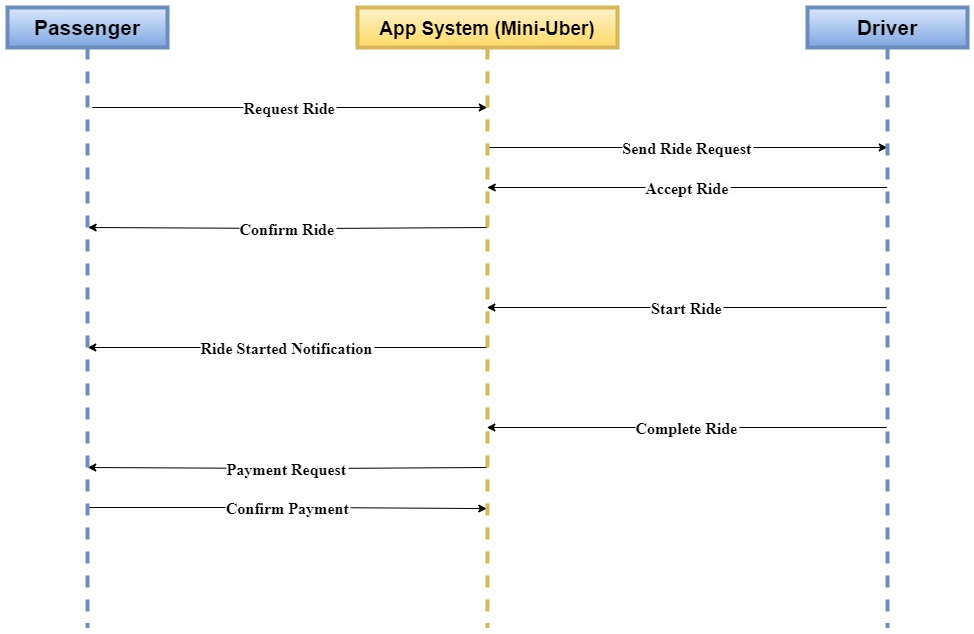
The **Ride-Hailing App** is the central component that interacts with different entities.

**Actors:**

* **Passenger:** Requests a ride and receives a confirmation.
* **Driver:** Receives ride requests and accepts rides.
* **Payment Gateway:** Processes ride payments and confirms transactions.
* **Maps API:** Provides route details and estimated arrival times.
* **Notification Service:** Sends ride updates to passengers and drivers.

**1.2 Interaction Model**

The **Interaction model** describes how different system components communicate through messages, focusing on user interactions and system processes.



*Fig. 1.2 UML Sequence Diagram of Interaction Model*

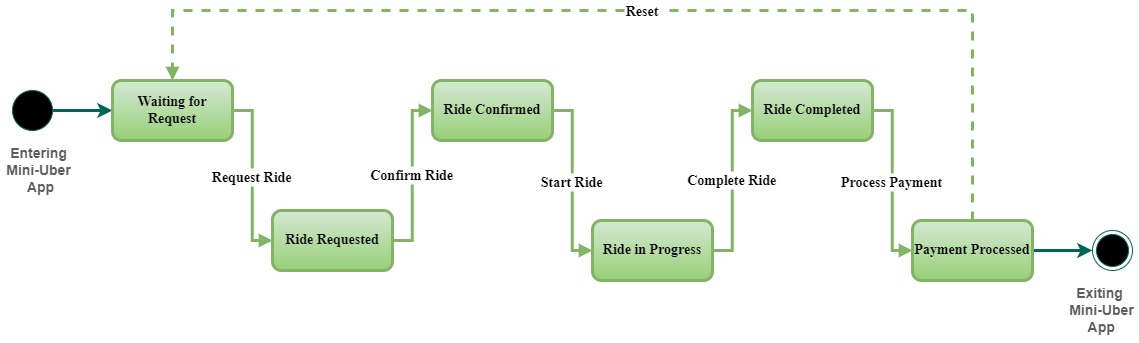
The above sequence diagram ensures that all ride-related events happen in the correct order.

**Step-by-step flow of the Model:**

1. The Passenger requests a ride.
2. The App System sends the request to the Driver.
3. The Driver accepts the ride.
4. The system confirms the ride to the passenger.
5. The driver starts the ride, and a notification is sent to the passenger.
6. After ride completion, the system sends a payment request.
7. The passenger confirms the payment, and the ride is marked as complete.

**1.3 Behavioural Model**

The **Behavioural model** explains how the system changes states based on user actions and system events.



*Fig. 1.3 UML State Diagram of Behavioural Model*

The above diagram represents different states of a ride from **entry to exit**.

This state diagram ensures that every ride follows a structured flow from request to payment.

**State transitions:**

1. **Waiting for Request**: The app is idle until a passenger requests a ride.
2. **Ride Requested:** The system processes the request and waits for driver confirmation.
3. **Ride Confirmed:** A driver accepts the ride, and the system moves to the next phase.
4. **Ride in Progress:** The driver starts the ride.
5. **Ride Completed:** The driver finishes the trip.
6. **Payment Processed:** The system finalizes the payment and exits.

**Unit 2: Scrum Development Plan for Mini-Uber**

A Scrum Plan serves as a structured roadmap for guiding the development of a project using the Scrum framework, a widely used Agile methodology. It defines how the development process is managed through iterative cycles, known as Sprints, ensuring continuous progress and adaptability.

**2.1 Sprint Structure with Key Deliverables**

The development of the ride-hailing app follows the Scrum methodology, divided into **four sprints** of two weeks each. Each sprint focuses on incremental feature development and testing, ensuring continuous integration and improvement.  
A Sprint is a fixed-duration development phase for the app where the team focuses on completing a defined set of tasks to deliver a functional increment of the software.

**Sprint Breakdown:**

**Sprint 1: Core System Setup & User Authentication**

* + Develop user registration & login (both passengers & drivers)
  + Implement profile management
  + Set up database & backend infrastructure
  + Basic UI design for login & dashboard
  + Testing & review

**Sprint 2: Ride Booking & Matching System**

* + Develop ride request & confirmation flow
  + Implement driver matching algorithm
  + Develop ride status tracking (Idle, Requested, Confirmed, In Progress, Completed)
  + UI/UX improvements
  + Initial user feedback & testing

**Sprint 3: Payment & Rating System**

* + Implement fare calculation & payment processing
  + Enable rating & feedback system
  + Improve app security features
  + Performance optimization
  + Bug fixes & testing

**Sprint 4: Final Enhancements & Deployment**

* + Final UI/UX refinements
  + Stress testing & scalability improvements
  + Deployment & launch preparation
  + Documentation & final review

**2.2 User Stories**

User stories define the key functionalities required by both passengers and drivers. Each story follows the format: **As a [user], I want to [goal] so that [reason].**

**Passenger User Stories:**

1. **Ride Request:** *As a passenger, I want to request a ride to the nearest driver so that I can quickly get my driver.*
2. **Ride Tracking:** *As a passenger, I want to track my ride in real-time so that I know my driver’s location.*
3. **Fare Estimation:** *As a passenger, I want to see an estimated fare before booking so that I can plan my expenses.*
4. **Payment Options:** *As a passenger, I want to pay via multiple options (card, cash, wallet) so that I have flexibility.*
5. **Rate & Review:** *As a passenger, I want to rate and review my driver so that I can help improve service quality.*

**Driver User Stories:**

1. **Accept Rides:** *As a driver, I want to accept or decline ride requests so that I can choose trips that suit me.*
2. **Navigation Assistance:** *As a driver, I want integrated navigation so that I can reach passengers efficiently.*
3. **Earnings Tracking:** *As a driver, I want to view my daily earnings so that I can manage my finances.*
4. **Availability Toggle:** *As a driver, I want to toggle my availability so that I can work when I choose.*
5. **Passenger Rating:** *As a driver, I want to rate passengers so that I can provide feedback on their behaviour.*

**2.3 Definition of Done (DoD) for Each User Story**

**Passenger User Stories:**

1. **Ride Request:**
   * Feature allows passengers to request a ride and matches them with the nearest available driver.
   * Pickup and drop-off locations are accurately processed.
   * Ride request status updates are displayed to the passenger.
   * System handles different scenarios (e.g., no available drivers).
   * Functionality is fully tested and approved.
2. **Ride Tracking:**
   * Real-time location tracking is enabled using GPS.
   * Passenger can view the driver’s live position on a map.
   * Location updates are accurate and refresh in real-time.
   * System undergoes performance testing to ensure reliability.
   * Documentation includes tracking implementation details.
3. **Fare Estimation:**
   * Fare calculation logic is correctly implemented.
   * Estimated fare is displayed before confirming the ride.
   * System considers distance, time, and dynamic pricing factors.
   * Feature is tested for different trip scenarios.
   * Stakeholders review and approve the final implementation.
4. **Payment Options:**
   * Multiple payment methods (cash, card, wallet) are fully integrated.
   * Transactions are processed securely and confirmed instantly.
   * System correctly handles failed and successful payments.
   * Security measures ensure user data protection.
   * UI is tested for seamless user experience.
5. **Rate & Review:**
   * Passengers can provide ratings and reviews after each ride.
   * Reviews are stored and displayed appropriately.
   * Ratings contribute to driver reputation.
   * System ensures feedback submission without errors.
   * Feature is reviewed and tested for fairness and accuracy.

**Driver User Stories:**

1. **Accept Rides:**
   * Drivers receive real-time ride requests with passenger details.
   * Option to accept or decline a request is functional.
   * System updates ride status upon driver’s selection.
   * Feature is tested under different traffic and availability conditions.
   * User experience is validated for responsiveness.
2. **Navigation Assistance:**
   * Integrated GPS provides route guidance for drivers.
   * Routes adjust dynamically based on real-time traffic.
   * Turn-by-turn navigation is clear and accurate.
   * Testing ensures smooth navigation without glitches.
   * UI is designed for ease of use while driving.
3. **Earnings Tracking:**
   * Drivers can view daily, weekly, and monthly earnings.
   * Payment breakdown per ride is accurately recorded.
   * Transaction details are securely stored and accessible.
   * Testing confirms earnings data accuracy.
   * User interface is simple and easy to navigate.
4. **Availability Toggle:**
   * Drivers can set their status as "Available" or "Unavailable."
   * Status updates reflect in real-time on the system.
   * Offline drivers do not receive ride requests.
   * Feature functions smoothly under different network conditions.
   * System undergoes usability testing to confirm ease of use.
5. **Passenger Rating:**
   * Drivers can rate passengers after a completed trip.
   * Rating system is fair and prevents duplicate reviews.
   * Passenger ratings are stored and affect their profile.
   * System ensures anonymity and data integrity.
   * Feature is tested for security and usability.

**Conclusion**

This document serves as a guide of **System Modeling and Agile Software Development** to create a System Design for a simple ride-hailing application such as Mini-Uber and implements Agile practices in the design process. The use of **Scrum methodology** structured the development process into well-defined **Sprints**, promoting continuous improvement and adaptability. **UML diagrams** play a crucial role in representing system interactions, user workflows, and behavioural processes, ensuring clarity in system design.

**References**

Scrum Guide – https://scrumguides.org/

Agile Manifesto – https://agilemanifesto.org/

Unified Modeling Language (UML) Documentation – https://www.uml.org/

IEEE Software Engineering Standards – https://standards.ieee.org/

System Modeling Concepts (IBM) – https://www.ibm.com/docs/en/rational

Agile Software Development (Atlassian) – https://www.atlassian.com/agile

Ride-Hailing App Architecture (Medium Articles) – https://medium.com/