

A
Project Report
On

Samaan
BTech-IT, Sem VI

Prepared By:
Preet Ketankumar Brahmabhatt (IT-116)
Ranipa Vraj Munesh (IT-120)

Guided By:
Prof. Kunal J Sahitya
Dept. of Information Technology



Department of Information Technology
Faculty of technology,
Dharmsinh Desai University
College road, Nadiad- 387001

April, 2025

CANDIDATE’S DECLARATION

We declare that 6th semester report entitled “Samaan” is our own work conducted under the supervision of the guide Prof. Kunal J Sahitya.

We further declare that to the best of our knowledge the report for B.Tech. VI semester does not contain part of the work which has been submitted either in this or any other university without proper citation.

Candidate’s Signature

Candidate’s Name: Preet Ketankumar Brahmbhatt

Student ID: 22ITUON075

Candidate’s Signature

Candidate’s Name: Ranipa Vraj Munesh

Student ID: 22ITUOS043

DHARMSINH DESAI UNIVERSITY

NADIAD-387001, GUJARAT



CERTIFICATE

This is to certify that the project carried out in the subject of Project-I, entitled “Samaan” and recorded in this report is a Bonafide report of work of

1) Preet Ketankumar Brahmbhatt Roll No: IT116 ID No:22ITUON075

2) Ranipa Vraj Munesh Roll No: IT120 ID No:22ITUOS043

of Department of Information Technology, semester VI. They were involved in Project work during academic year 2024 -2025.

Prof. Kunal J Sahitya

(Project Guide),

Department of Information Technology,

Faculty of Technology, Dharmsinh Desai University, Nadiad

Date:

Prof. (Dr.) V. K. Dabhi

Head, Department of Information Technology,

Faculty of Technology, Dharmsinh Desai University, Nadiad

Date:

ACKNOWLEDGEMENT

We extend our heartfelt gratitude to the **Department of Information Technology, Dharmsinh Desai University**, for providing us with the necessary resources and support throughout our project journey.

We sincerely thank our project guide, **Prof. Kunal J Sahitya**, for his invaluable guidance, encouragement, and insightful feedback, which played a crucial role in shaping our project "**Samaan**". His expertise and mentorship have been instrumental in our learning and the successful completion of this project.

We also express our deep appreciation to our Head of the Department, **Prof. (Dr.) V. K. Dabhi**, for fostering an environment of learning and innovation and for his constant support.

Lastly, we acknowledge our faculty members, peers, and everyone who directly or indirectly contributed to this project.

Preet Ketankumar Brahmbhatt (IT116)

Ranipa Vraj Munesh (IT120)

B. Tech Semester VI
Department of Information Technology
Dharmsinh Desai University

TABLE OF CONTENTS

1.0 Introduction.....	1
1.1 Project Details.....	1
1.2 Purpose.....	1
1.3 Scope.....	1
1.4 Objective.....	2
1.5 Technology and Literature Review.....	2
2.0 Project Management.....	3
2.1 Feasibility Study.....	3
2.1.1 Technical feasibility.....	3
2.1.2 Time schedule feasibility.....	3
2.1.3 Operational feasibility.....	3
2.1.4 Implementation feasibility.....	3
2.2 Project Planning.....	3
2.2.1 Project Development Approach and Justification	3
2.2.2 Project Plan	4
2.2.3 Milestones and Deliverables	4
2.2.4 Roles and Responsibilities	4
2.2.5 Group Dependencies	4
2.3 Project Scheduling.....	5
2.3.1 Project Scheduling chart	5
3.0 System Requirements Study	6
3.1 Study of Current System.....	6
3.2 Problems and Weaknesses of Current System.....	6
3.3 User Characteristics	6
3.4 Hardware and Software Requirements.....	6
3.5 Constraints.....	7
3.5.1 Regulatory Policies.....	7
3.5.2 Hardware Limitations.....	7
3.5.3 Interfaces to Other Applications.....	7
3.5.4 Parallel Operations.....	7
3.5.5 Higher Order Language Requirements.....	7
3.5.6 Reliability Requirements.....	7
3.5.7 Criticality of the Application.....	7
3.5.8 Safety and Security Consideration.....	7
3.6 Assumptions and Dependencies.....	8

4.0	System Analysis	9
4.1	Requirements of New System (SRS)	9
4.1.1	User Requirements	9
4.1.2	System Requirements	9
4.2	Features Of New System	10
4.3	Navigation Chart	11
4.4	Class Diagram	12
4.5	System Activity	13
4.6	Sequence Diagram	14
5.0	System Design	15
5.1	System Architecture Design	16
5.1.1	Component Diagram	16
5.1.2	Deployment Diagram	16
5.2	Input/Output and Interface Design	17
5.2.1	State Transition/UML Diagram	17
5.2.2	Samples Of Forms, Reports and Interface.....	17
6.0	Implementation Planning.....	19
6.1	Implementation Environment	19
6.2	Program/Modules Specification	20
6.3	Coding Standards	21
7.0	Testing	23
7.1	Testing Plan	23
7.2	Testing Strategy.....	24
7.3	Testing Methods.....	24
7.4	Test Cases	25
7.4.1	Purpose	25
7.4.2	Required Input	25
7.4.3	Expected Result	25
8.0	User Manual	27
8.1	Home Page	27
8.2	Register Page	28
8.3	Login Page	29
8.4	Carrier Dashboard Page	30
8.5	Add New Trip Page	31
8.6	Sender Chat List Page	32
8.7	Search Carrier Page	33
8.8	Sender Dashboard Page	34
9.0	Limitation and Future Enhancement.....	35

10.0	Conclusion and Discussion	36
10.1	Conclusions and Future Enhancement	36
10.2	Discussion	36
	10.2.1 Self-Analysis of Project Viabilities	36
	10.2.2 Problem Encountered and Possible Solutions	37
	10.2.3 Summary of Project work	37
11.0	References	38

ABSTRACT

The project Samaan is a peer-to-peer package delivery platform designed to connect individuals who wish to send packages with travellers (carriers) who have spare space in their vehicles. Unlike traditional courier services, Samaan offers a cost-effective, flexible, and user-friendly alternative by leveraging a community-driven logistics model. Developed using React.js for the frontend and Spring Boot for the backend, the platform incorporates real-time communication via WebSockets and stores data using MongoDB. The project follows Agile methodology to ensure iterative development, user feedback incorporation, and scalability. While the current implementation focuses on core delivery operations and communication, future enhancements such as live GPS tracking, AI-based trip matching, and multilingual support are proposed to further improve the user experience and reach. Samaan demonstrates the potential of decentralized logistics to reduce delivery costs and bridge gaps in existing package transport systems.

Key Features

- Role-based authentication for senders and carriers
- Trip creation, search, and booking system
- Real-time chat using WebSockets
- Interactive dashboards for both user roles
- Secure data handling with JWT and MongoDB

LIST OF FIGURES

Figure	Name	Page No.
2.1	Scheduling	5
4.1	Navigation Chart	11
4.2	Class Diagram	12
4.3	Use Case Diagram	13
4.4.1	Sequence Diagram (Chat)	14
4.4.2	Sequence Diagram (Add New Trip)	14
5.1	Component Diagram	16
5.2	Deployment Diagram	18
8.1.0	Home (1)	27
8.1.1	Home (2)	27
8.2	Register	28
8.3	Login	29
8.4	Carrier Dashboard	30
8.5	Add New Trip	31
8.6.0	Sender Chat List	32
8.6.1	Personal Chat	32
8.7	Search Carrier	33
8.8	Sender Dashboard	34

LIST OF TABLES

Table	Name	Page No.
6.1	Environmental Considerations	19
6.2	Core Modules	20
7.1	Responsibilities	23
7.2	Testing Methods	24
7.3	Registration Test	25
7.4	Search Testing	25
7.5	Chat Testing	26
7.6	Authentication Testing	26
10.1	Self-Analysis of Project Viabilities	36
10.2	Problems Encountered and Possible Solutions	37

1. Introduction

1.1 Project Details

Samaan is a full-stack package delivery platform that facilitates seamless connections between senders and carriers. It enables users to schedule package deliveries, search for available trips, and engage in real-time chat with carriers. The platform ensures secure transactions and a streamlined process for booking and tracking deliveries.

1.2 Purpose

The purpose of Samaan is to provide a digital solution for individuals and businesses looking to send packages conveniently. It aims to:

- Connect senders with carriers who have available space in their vehicles.
- Offer a cost-effective and efficient way to transport goods.
- Provide real-time chat and notifications for better coordination.
- Improve the logistics and delivery process for users.

1.3 Scope

Samaan covers the following functionalities:

- User authentication (Sender and Carrier roles).
- Trip creation and search based on origin, destination, and date.
- Cost estimation for package delivery.
- One-on-one chat between sender and carrier.
- Real-time notifications for new messages.

Out of Scope:

- Direct payment integration (handled externally).
- Insurance or liability coverage for package safety.

1.4 Objective

The objective of Samaan is to create a reliable and user-friendly package delivery platform that allows individuals to transport packages using available space in carriers' vehicles. It ensures:

- Transparency in trip availability.
- Secure communication between senders and carriers.
- Efficient trip management and tracking.

1.5 Technology and Literature Review

Technology Stack:

- Frontend: React.js (for a dynamic and responsive user interface)
- Backend: Java Spring Boot (for handling business logic)
- Database: MongoDB (for efficient data storage and retrieval)
- WebSocket: Used for real-time chat communication
- Hosting: Render (Frontend), Render (Backend)

Literature Review:

Existing logistics solutions focus primarily on traditional courier services, which can be costly and inflexible. Samaan introduces a peer-to-peer delivery model, reducing costs and improving efficiency. Studies show that decentralized logistics models can significantly reduce delivery time and operational expenses.

2. Project Management

2.1 Feasibility Study

2.1.1 Technical Feasibility

Samaan is technically feasible as it leverages modern web technologies like React.js, Spring Boot, and MongoDB. The architecture ensures scalability, performance, and security.

2.1.2 Time Schedule Feasibility

An iterative agile approach is followed, ensuring incremental feature development:

- Phase 1: Requirement gathering & system design
- Phase 2: Backend & frontend development
- Phase 3: WebSocket integration for real-time chat
- Phase 4: Testing & deployment

2.1.3 Operational Feasibility

The platform is user-friendly, requiring minimal training. Features like automated trip searches, real-time chat, and intuitive UI ensure smooth operation.

2.1.4 Implementation Feasibility

Samaan can be deployed in phases, allowing initial testing with a limited user base before a full-scale rollout.

2.2 Project Planning

2.2.1 Project Development Approach

An Agile development methodology is followed, allowing continuous feedback and rapid iterations.

2.2.2 Project Plan

- Week 1-2: Requirement analysis, UI/UX design
- Week 3-5: Backend and database setup
- Week 6-8: Frontend development and API integration
- Week 9-10: WebSocket-based chat implementation
- Week 11: Testing and bug fixes
- Week 12: Deployment and feedback

2.2.3 Milestones and Deliverables

- Milestone 1: Completion of database setup and backend APIs
- Milestone 2: Frontend UI implementation
- Milestone 3: Integration of WebSocket chat
- Milestone 4: System testing and deployment

2.2.4 Roles and Responsibilities

- Frontend Developer: UI development using React.js
- Backend Developer: API and database management with Spring Boot & MongoDB
- Tester: Ensures system functionality and bug fixing
- Project Manager: Coordinates task and ensures timely delivery

2.2.5 Group Dependencies

The frontend and backend teams work in parallel, with dependencies on API development and database integration.

2.3 Project Scheduling

Project Scheduling Chart :-

A Gantt chart is used to visualize project progress, ensuring efficient time management.

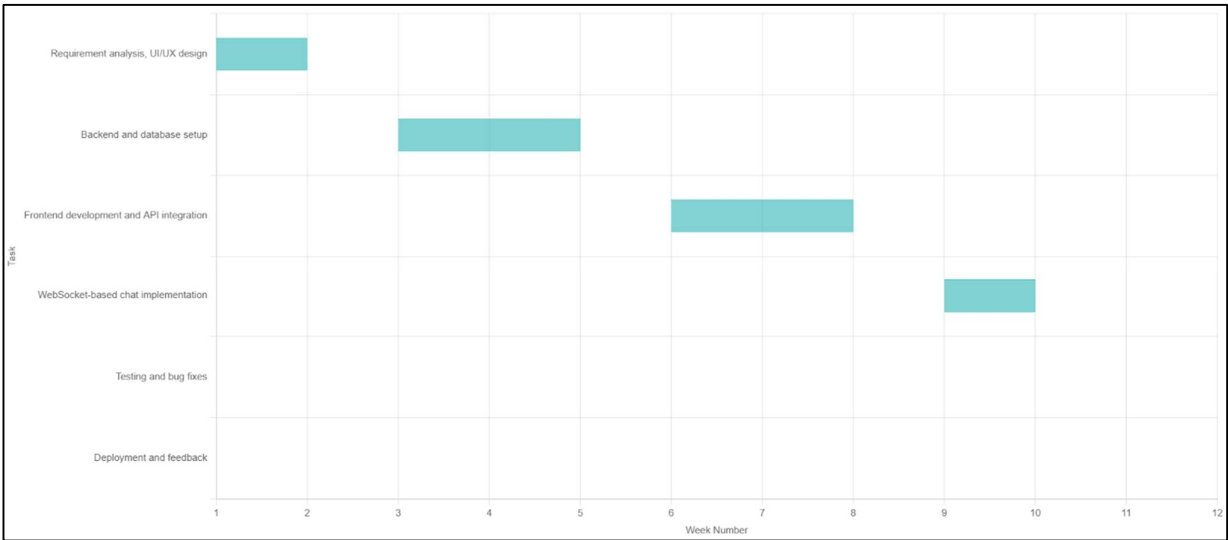


Figure 2.1 Scheduling

3. System Requirement Study

3.1 Study of Current System

Traditional package delivery systems rely on dedicated logistics companies, leading to high costs and delays. Current courier services lack flexible and cost-effective peer-to-peer delivery options.

3.2 Problems and Weaknesses of System

- High shipping costs for small deliveries.
- Inefficiency in utilizing available transport space.
- Lack of direct communication between senders and carriers.

3.3 User Characteristics

- Senders: Individuals or businesses looking to transport packages.
- Carriers: Vehicle owners willing to carry extra packages.
- Admins: Manage user registrations and monitor activity.

3.4 Hardware and Software Requirements

Hardware:

- Processor: Intel Core i3 or higher
- RAM: 4GB minimum
- Storage: 50GB free space

Software:

- Backend: Java Spring Boot
- Frontend: React.js
- Database: MongoDB
- Server: Render (backend hosting), Render (frontend hosting)

3.5 Constraints

3.5.1 Regulatory Policies

- Compliance with privacy regulations (e.g., GDPR).
- Ensuring user data protection.

3.5.2 Hardware Limitations

- The platform is optimized for standard consumer devices.

3.5.3 Interfaces to Other Applications

- Integration with third-party authentication services (Google Login).
- Potential API connections for payment processing (future scope).

3.5.4 Parallel Operations

- Multiple users should be able to interact with the system concurrently.

3.5.5 Higher Order Language Requirements

- Developed using Java Spring Boot and React.js.

3.5.6 Reliability Requirements

- 99.9% uptime with backup mechanisms.

3.5.7 Safety and Security Considerations

- End-to-end encryption for chat data.
- Secure storage of user credentials.

3.6 Assumptions and Dependencies

- Users will provide accurate information.
- The system will evolve based on feedback.
- Reliable internet connectivity is assumed.

4. System Analysis

4.1 Requirements of New System (SRS)

The new system, Samaan, is designed to facilitate seamless package delivery by connecting senders with carriers. It enables senders to find suitable carriers for their packages while ensuring real-time communication and efficient trip management.

4.1.1 User Requirements

- Senders should be able to register, search for available trips, and book a carrier for their package.
- Carriers should be able to register, list their trips with available capacity, and manage sender requests.
- The platform should provide a user-friendly interface with intuitive navigation.
- Senders and carriers should have a real-time chat feature to communicate about deliveries.

4.1.2 System Requirements

Functional Requirements

- User Authentication: Senders and carriers must log in to access the system.
- Trip Creation and Search:
 - Carriers can create trips by specifying source, destination, date, and available space.
 - Senders can search for trips matching their delivery needs.
- Real-Time Chat: Secure chat functionality for senders and carriers.

Non-Functional Requirements

- Security: The system must encrypt user data and messages.
- Scalability: Should support an increasing number of users and deliveries.
- Performance: Messages and trip data should be retrieved in real-time with minimal delays.
- Cross-Platform Accessibility: The system should work efficiently on both mobile and desktop devices.

4.2 Features of the New System

- User Registration and Authentication: Secure signup and login for senders and carriers.
- Trip Posting and Search:
 - Carriers can add trips, specifying their available space.
 - Senders can browse trips that match their package delivery requirements.
- Package Booking System: Senders can request a trip, and carriers can approve or decline requests.
- Real-Time Chat System: Facilitates smooth communication between senders and carriers for better coordination.
- Profile Management: Users can update their personal details and trip preferences.

4.3 Navigation Chart

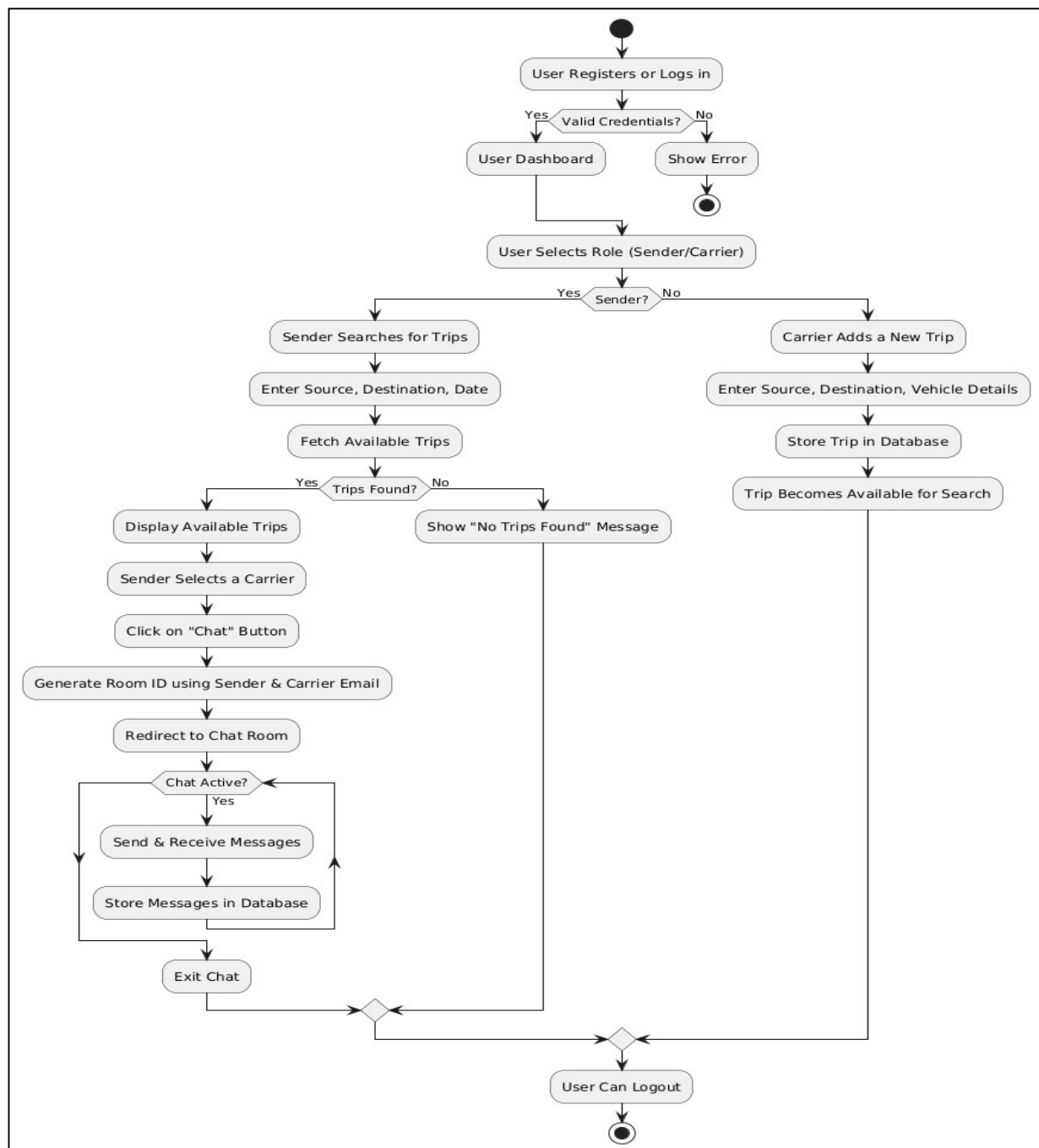


Figure 4.1 Navigation Chart

4.4 Class Diagram

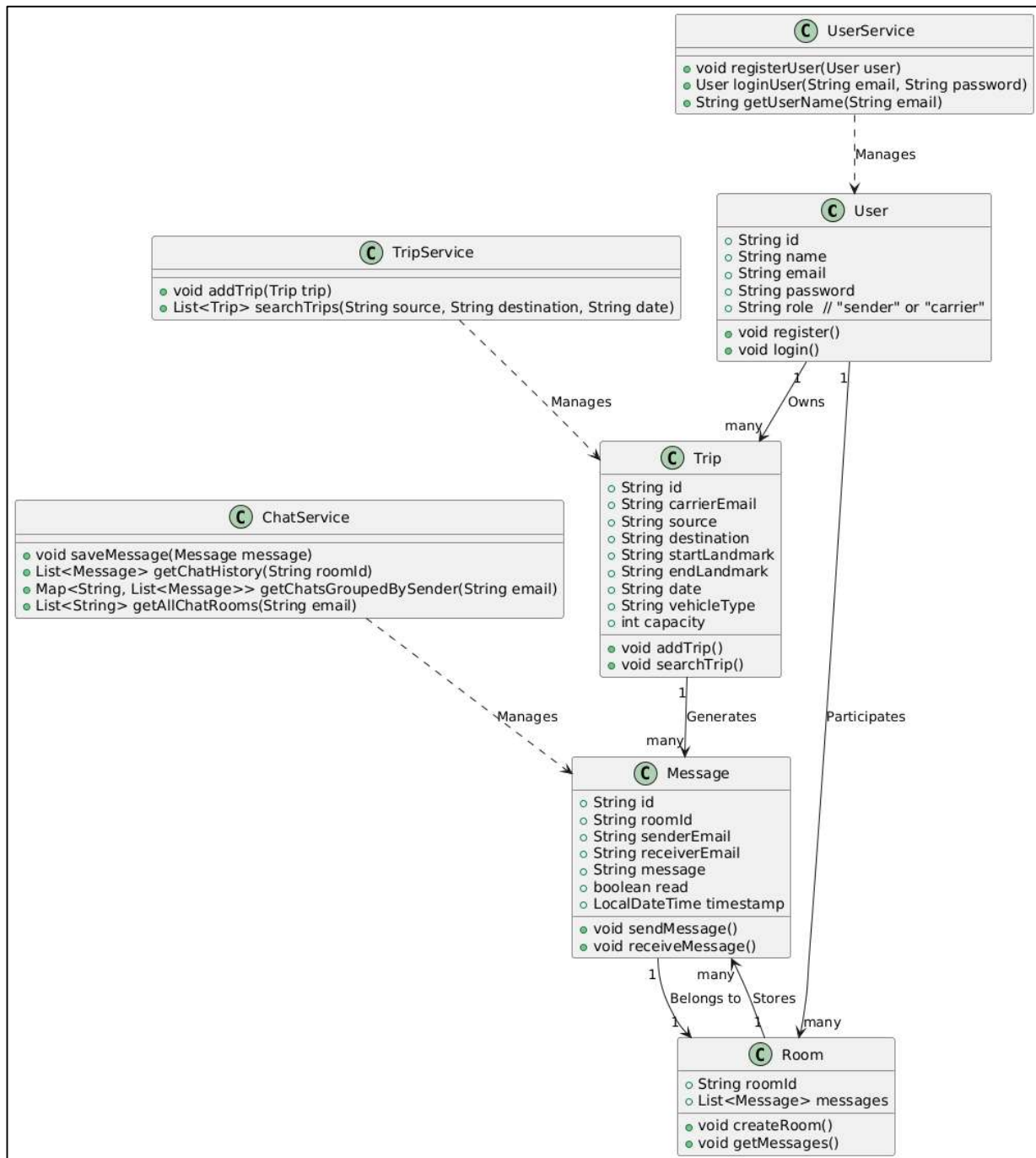


Figure 4.2 Class Diagram

4.5 Use Case Diagram

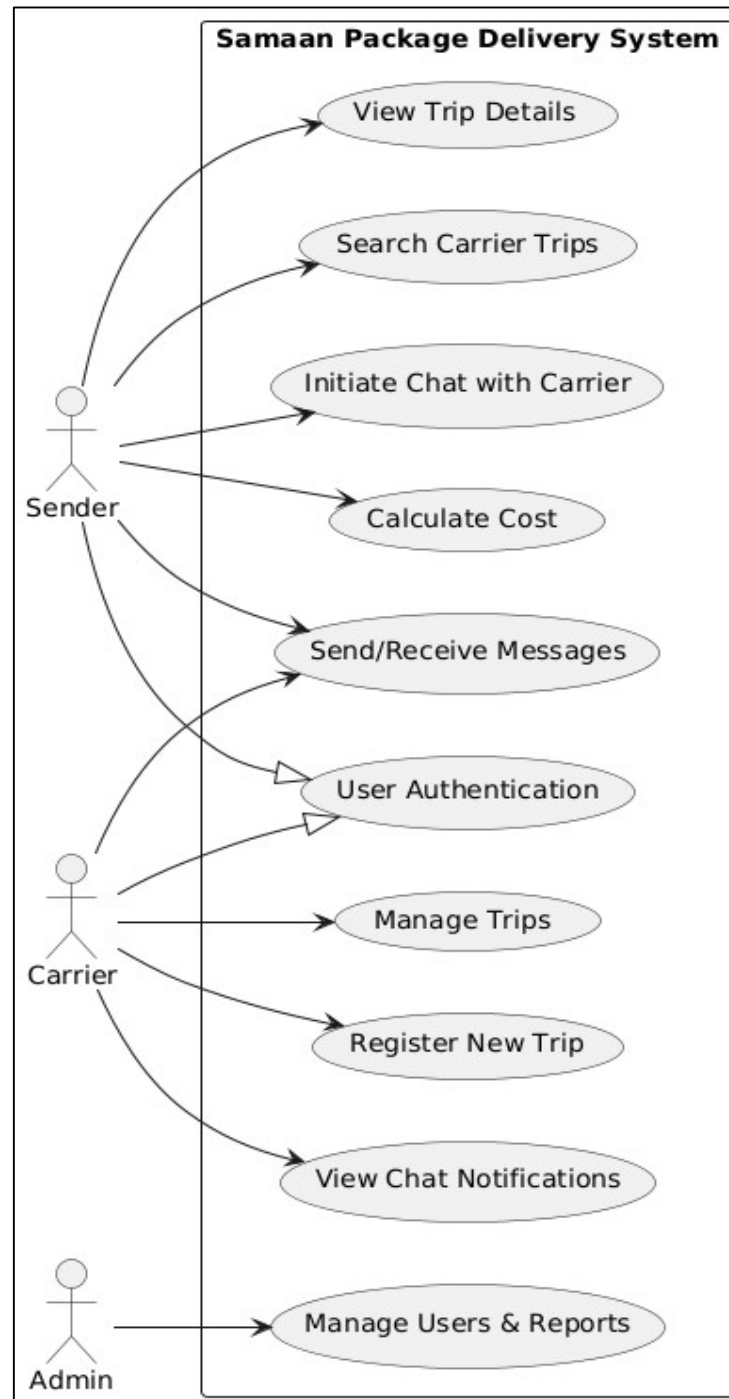


Figure 4.3 Use Case Diagram

4.6 Sequence Diagram

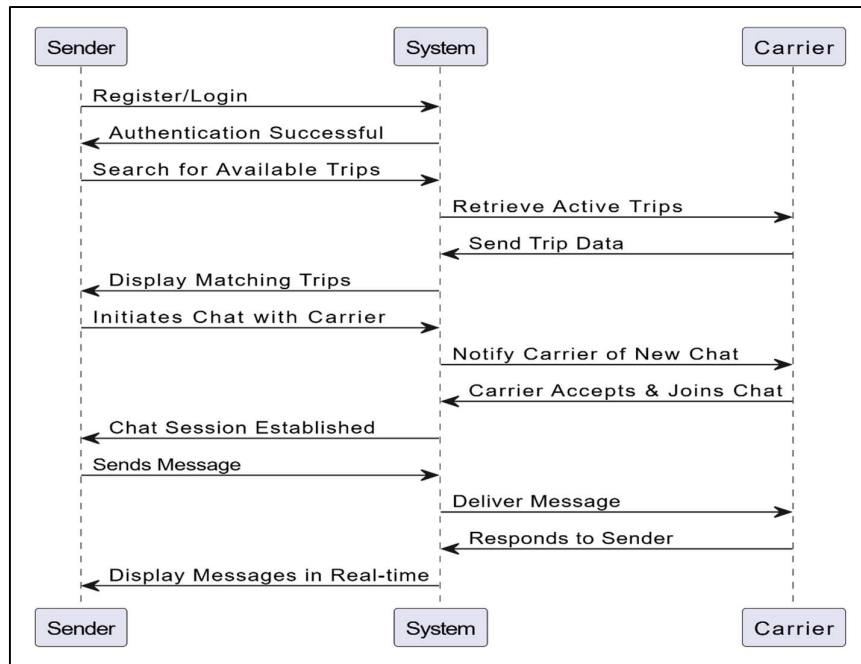


Figure 4.4.1 Sequence Diagram (Add new Trip)

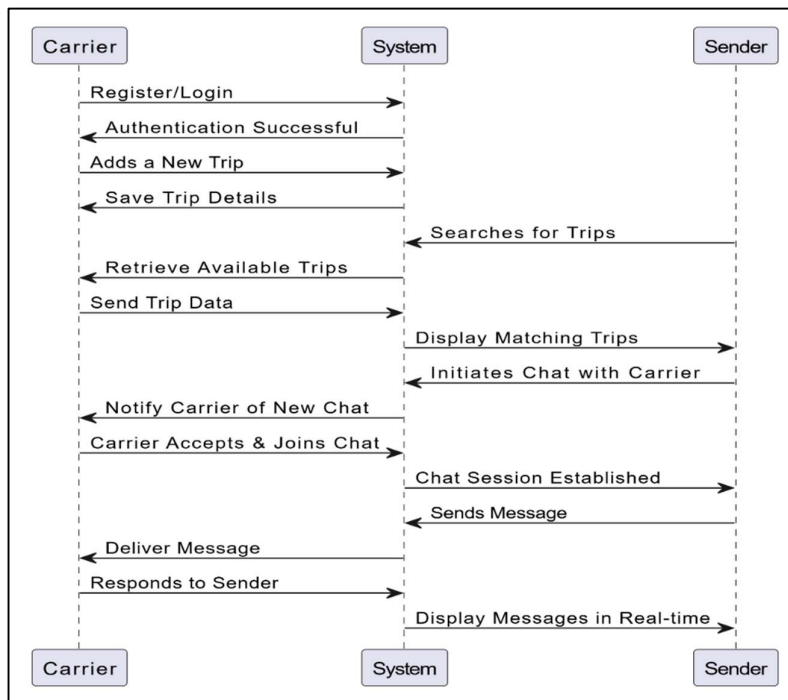


Figure 4.4.2 Sequence Diagram (Chat)

5. System Design

5.0 System Architecture Design

Overview of System Design and Architecture

Samaan follows the Model-View-Controller (MVC) architecture to ensure modularity, maintainability, and scalability. The system is a full-stack web application developed using React.js (Frontend) and Spring Boot (Backend) with MongoDB as the database. It also utilizes WebSockets for real-time chat functionality between senders and carriers.

Architecture Breakdown

- Frontend (View Layer): Developed using React.js, providing an interactive UI and seamless user experience.
- Backend (Controller Layer): Built using Spring Boot, handling authentication, data processing, business logic, and real-time chat functionality.
- Database (Model Layer): Uses MongoDB, ensuring flexible and scalable data storage for user data, trips, and chat history.
- WebSockets: Enables real-time chat between senders and carriers using STOMP over WebSockets.

Key Features of System Architecture

1. Separation of Concerns: The MVC architecture ensures a clear distinction between frontend, backend, and database.
2. Real-Time Communication: Uses WebSockets for instant chat messaging between senders and carriers.
3. Scalability: The backend supports RESTful APIs, allowing seamless future expansions such as mobile app integration.
4. Security: Implements JWT-based authentication to secure user sessions.
5. Performance Optimization: Utilizes caching mechanisms and asynchronous processing to improve system efficiency.
6. Reliability: Spring Boot ensures robust error handling and logging mechanisms for debugging and troubleshooting.

5.1 Component Diagram

Explanation of Components

The Component Diagram represents the logical structure of the system and how different modules interact in Samaan. It consists of:

- Authentication Module: Manages user registration, login, and authentication.
- Trip Management Module: Allows carriers to add, modify, and remove trips while allowing senders to search for trips.
- Chat Module: Enables real-time chat between senders and carriers using WebSockets.
- Notification Module: Sends chat notifications to carriers when a sender initiates a conversation.
- Database Module: Stores user data, trips, chat history, and notifications in MongoDB.

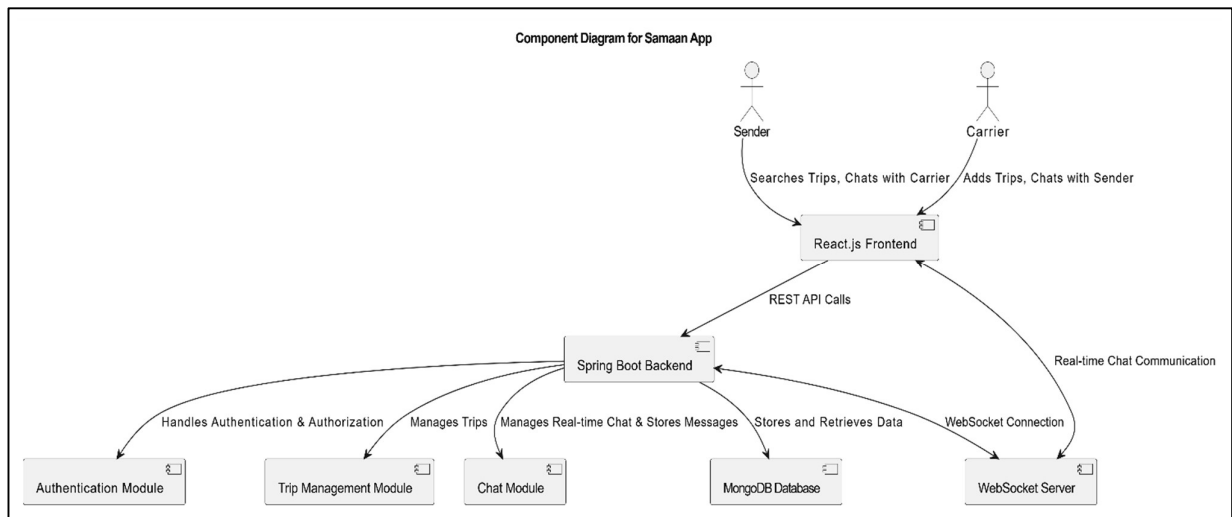


Figure 5.1 Component Diagram

Component Interaction Flow

1. The Frontend interacts with the Backend via REST APIs for authentication, trip search, and chat history retrieval.
2. The Backend processes requests and interacts with different modules.
3. The Chat Module uses WebSockets for real-time messaging.
4. The Database Module stores all data including user information, trips, chat logs, and notifications.

5.2 Deployment Diagram

Deployment Diagram Explanation

The Deployment Diagram illustrates how the software is deployed on cloud infrastructure, ensuring scalability and high availability.

Deployment Breakdown

1. Client (User's Device):
 - Users (Senders & Carriers) access the platform via web browsers.
 - The React.js UI is hosted on Render for easy deployment and performance optimization.
2. Application Server (Backend Server):
 - The Spring Boot backend is hosted on Render, handling API requests, authentication, and business logic.
 - WebSocket connections manage real-time messaging between senders and carriers.
3. Database Server:
 - MongoDB Atlas is used as the database service to store:
 - User information (Senders, Carriers).
 - Trip details (source, destination, date).
 - Chat history (message logs, timestamps).
 - Data is accessed via Spring Data MongoDB for efficient database interactions.

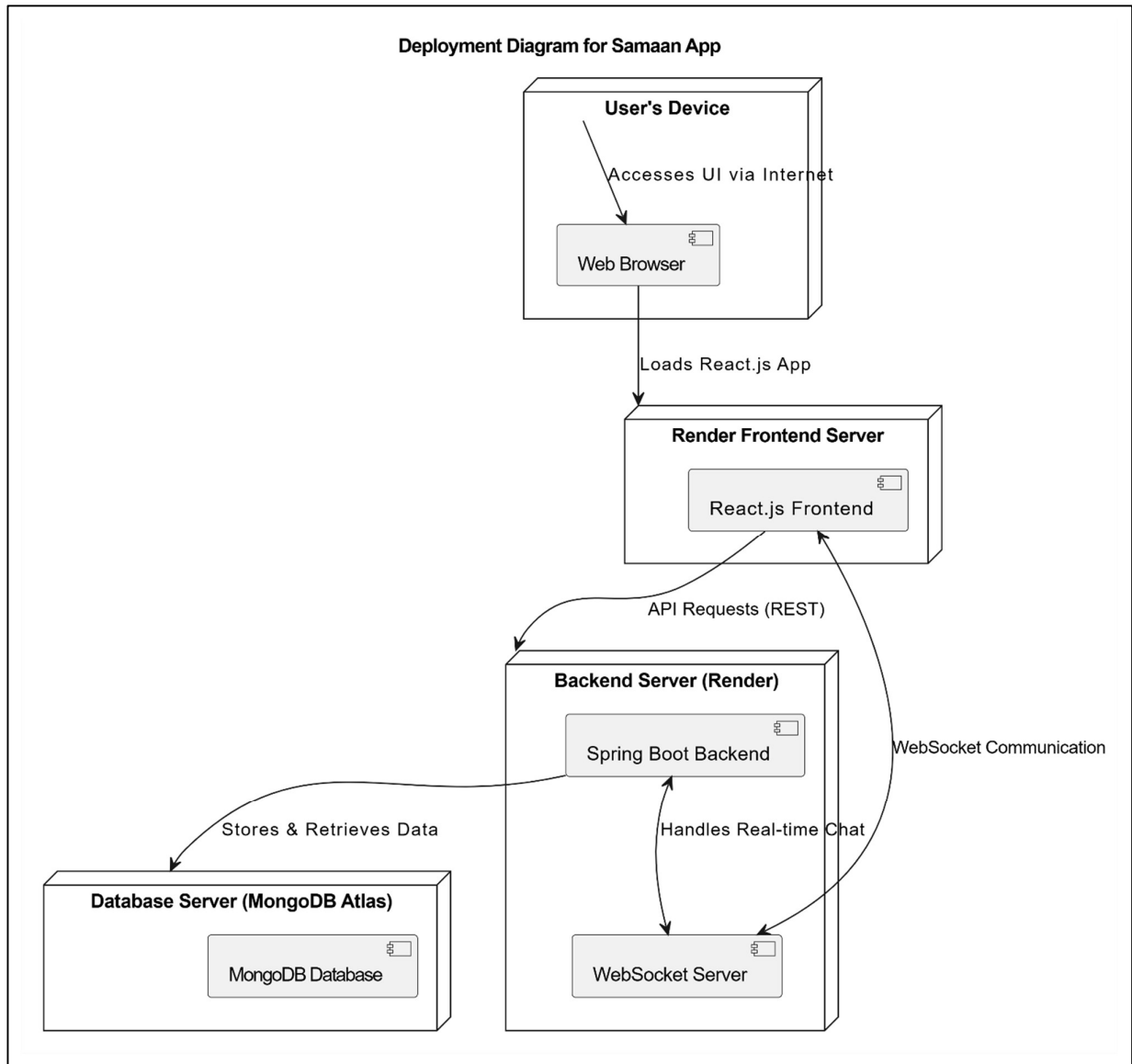


Figure 5.2 Deployment Diagram

Deployment Flow

1. Users access the platform via a web browser.
2. The React.js frontend is hosted on Render and communicates with the backend via REST APIs.
3. The Spring Boot backend on Render handles authentication, trip management, and chat operations.
4. The MongoDB database stores all critical application data, including trips and chat messages.
5. WebSockets handle real-time chat communication between senders and carriers.

6.0 Implementation Planning

This section outlines the implementation strategy for the **Samaan** platform, covering environment setup, module specifications, and coding standards to ensure maintainability, scalability, and efficiency.

6.1 Implementation Environment

The implementation environment defines the system setup required for deployment and execution. **Samaan** is developed as a **multi-user, web-based application** with a **GUI**.

Table 6.1 Environment Considerations

Aspect	Implementation Details
System Type	Multi-user
Interface	GUI (Web-based)
Backend	Java Spring Boot (REST API, WebSocket)
Frontend	React.js
Database	MongoDB Atlas
Hosting	Cloud-based (Frontend on Render, Backend on Render, Database on MongoDB Atlas)
Security	User authentication with JWT, Secure WebSockets

Why Multi-User GUI?

- Web-based GUI provides accessibility from any device.
- Multi-user system allows senders and carriers to interact concurrently.
- Spring Boot + React ensures scalability and a modern user experience.

6.2 Program/Modules Specification

The platform consists of various modules handling different functionalities. Below is an overview of the core modules and their responsibilities.

Table 6.2 Core Modules

Module Name	Description	Technologies Used
User Authentication	Manages user login, registration, and authentication (senders & carriers).	Spring Security
Trip Management	Allows carriers to create, update, and delete trips.	Java Spring Boot, MongoDB
Trip Search	Enables senders to search for trips based on source, destination, and date.	React.js, Axios
Chat System	Provides real-time chat between senders and carriers.	WebSockets, MongoDB
Notification System	Sends notifications when a new chat is initiated.	WebSockets
User Dashboard	Provides a personalized dashboard for senders and carriers.	React.js, Redux
Data Security	Ensures encrypted user credentials and secure API calls.	Spring Security

Module Interaction Flow

1. User registers/logs in → Authentication.
2. Carrier creates a trip → Data is stored in **MongoDB Atlas** via **Spring Boot**.
3. Senders search for trips → Query results are displayed based on source, destination, and date.
4. Sender clicks "Chat" → A chat room is automatically created using the sender's and carrier's email.
5. Carrier gets a notification → WebSocket-based real-time notification.
6. Users exchange messages → Chat messages are stored in MongoDB and delivered via WebSockets.

6.3 Coding Standards

Following coding standards ensures **readability, maintainability, and security**.

Backend (Spring Boot - Java) Coding Standards

- Follow Java naming conventions:
 - Classes → PascalCase (e.g., TripService)
 - Variables & Methods → camelCase (e.g., getTripDetails())
 - Constants → UPPER_CASE_SNAKE_CASE (e.g., MAX_RETRIES)
- Use RESTful API principles:
 - GET /trips/search → Retrieve available trips.
 - POST /trips/add → Create a new trip.
 - PUT /trips/{id} → Update trip details.
 - DELETE /trips/{id} → Delete a trip.
- Spring Boot Best Practices:
 - Use Service Layer to separate business logic from controllers.
 - Implement DTOs (Data Transfer Objects) for API responses.
 - Use JPA/Hibernate for structured database operations.

Frontend (React.js - JavaScript) Coding Standards

- Follow JavaScript naming conventions:
 - Components → PascalCase (e.g., ChatWindow.js)
 - Functions & variables → camelCase (e.g., fetchTripData())
- React Best Practices:
 - Use Functional Components with React Hooks (useState, useEffect).
 - Implement Redux for state management.
 - Use Axios for API calls.

Database Standards (MongoDB + Spring Boot)

- Use document-based schema design for flexibility.
- Use indexes on frequently queried fields (email, tripId) for performance optimization.
- Use a consistent naming convention for collections:
 - users → Stores user details (senders & carriers).
 - trips → Stores trip details.

- messages → Stores chat messages.
- rooms → Stores chat room details.

Security Standards

- Implement CORS to prevent unauthorized access.
- Use HTTPS for API communication.

Version Control & Documentation

- GitHub for version control.
- Code reviews before merging new features.
- JSDoc and JavaDocs for function and class documentation.

Summary of Implementation Plan

- Frontend: React.js, hosted on Render.
- Backend: Spring Boot, hosted on Render.
- Database: MongoDB Atlas for cloud storage.
- Real-time Chat: WebSockets for instant messaging between senders and carriers.
- Security: secure REST APIs.

7.0 Testing

Testing ensures the reliability, security, and performance of Samaan before deployment. This section outlines the testing plan, strategy, methods, and test cases to validate the system.

7.1 Testing Plan

The testing plan defines the scope, objectives, schedule, and responsibilities of testing.

Objectives

- Ensure all functionalities work as expected.
- Identify and fix bugs, vulnerabilities, and performance issues.
- Validate user interactions, database integrity, and API responses.
- Ensure cross-browser compatibility for a seamless experience.

Testing Scope

- Unit Testing (*Individual modules such as chat, trip search, and user authentication*).
- Integration Testing (*Interaction between chat, trip booking, and database operations*).
- System Testing (*Validates the entire package delivery and chat system*).
- User Acceptance Testing (UAT) (*Final validation by senders and carriers*).

Table 7.1:- Responsibilities

Role	Responsibility
Developers	Perform unit testing, fix bugs, and write test cases.
QA Testers	Conduct integration, system, and UAT testing.
End Users (Senders & Carriers)	Perform UAT and provide feedback.

7.2 Testing Strategy

The testing strategy defines how different testing phases will be executed.

Types of Testing Used

1. Unit Testing

- Tests individual methods, functions, and API endpoints.
- Tools Used: JUnit (Java), Jest (React.js).

2. Integration Testing

- Tests how different modules interact (*e.g., sender chatting with the carrier*).
- Tools Used: Postman, Spring Test Framework.

3. System Testing

- Validates the entire Samaan system.

Includes:

- Functional Testing – Ensures trip search, chat, and booking work correctly.
- Performance Testing – Measures speed, scalability, and concurrent usage handling.
- Security Testing – Tests authentication, role-based access, and chat security.

4. User Acceptance Testing (UAT)

- Conducted by senders and carriers before launch.
- Ensures real-world usability and feedback implementation.

7.3 Testing Methods

Table 7.2 Testing Methods

Testing Method	Description	Example
Black Box Testing	Tests without knowing the internal code.	Checking if senders can search for carriers and start a chat.
White Box Testing	Tests with internal code knowledge.	Verifying database queries for trip search.
Manual Testing	Testers manually execute test cases.	Checking UI responsiveness of the chat feature.
Automated Testing	Scripts run tests automatically.	Running API tests for chat and trip booking using Postman.

7.4 Test Cases

7.4.1 Purpose

Each test case ensures a specific feature functions correctly.

7.4.2 Required Input

Inputs include user actions, API calls, and database queries.

7.4.3 Expected Result

Defines the expected system behaviour for each input.

Sample Test Cases

1. User Registration Test

Table 7.3 Registration Test

Test Case ID	TC-001
Feature	User Registration
Input	New user enters valid details
Expected Output	User successfully registered & redirected to the dashboard
Actual Output	Pass
Status	Pass / Fail

2. Sender Searching for a Trip

Table 7.4 Search Testing

Test Case ID	TC-002
Feature	Search for carrier trips
Input	Sender enters source, destination, and date
Expected Output	List of available carriers is displayed
Actual Output	Pass
Status	Pass / Fail

3. Chat Initialization & Message Sending

Table 7.5 Chat Testing

Test Case ID	TC-003
Feature	Real-time chat between sender and carrier
Input	Sender clicks on "Chat" with a carrier
Expected Output	Chat room is created, messages are sent and received
Actual Output	Pass
Status	Pass / Fail

4. Authentication Security Test

Table 7.6 Authentication Testing

Test Case ID	TC-004
Feature	Unauthorized login attempt
Input	User enters incorrect password 5 times
Expected Output	Account is not allowed to be accessed
Actual Output	Pass
Status	Pass

Summary of Testing Plan

- Unit Testing ensures all components function correctly.
- Integration Testing ensures modules interact properly.
- System Testing validates real-world conditions.
- User Acceptance Testing (UAT) ensures Samaan meets user needs.

8.0 User Manual

8.1 Home Page

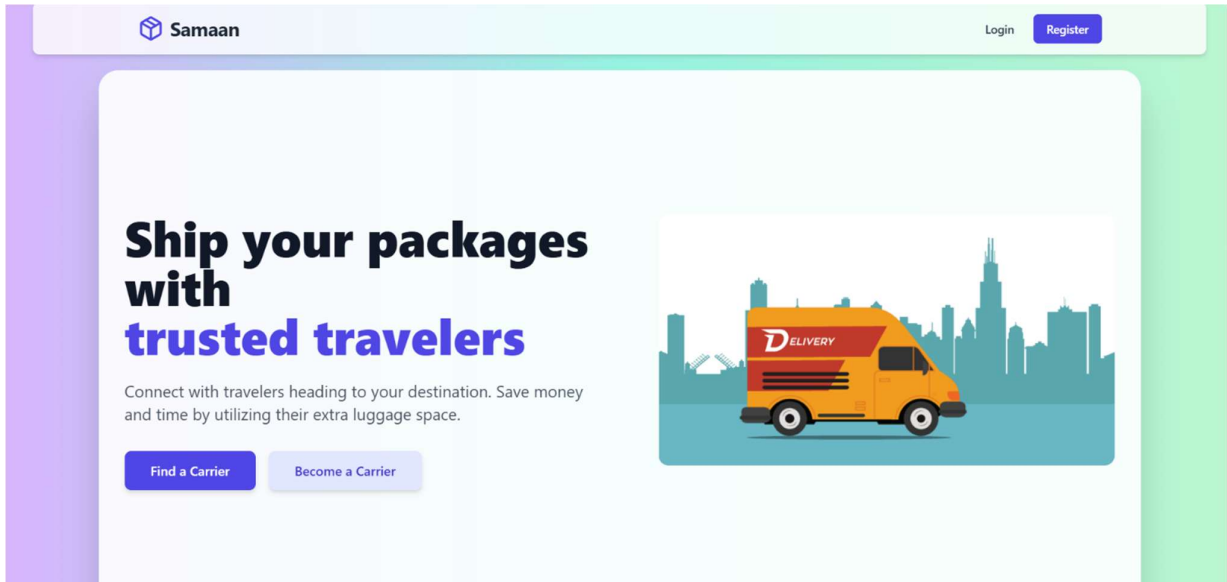


Figure 8.1.0 Home (1)

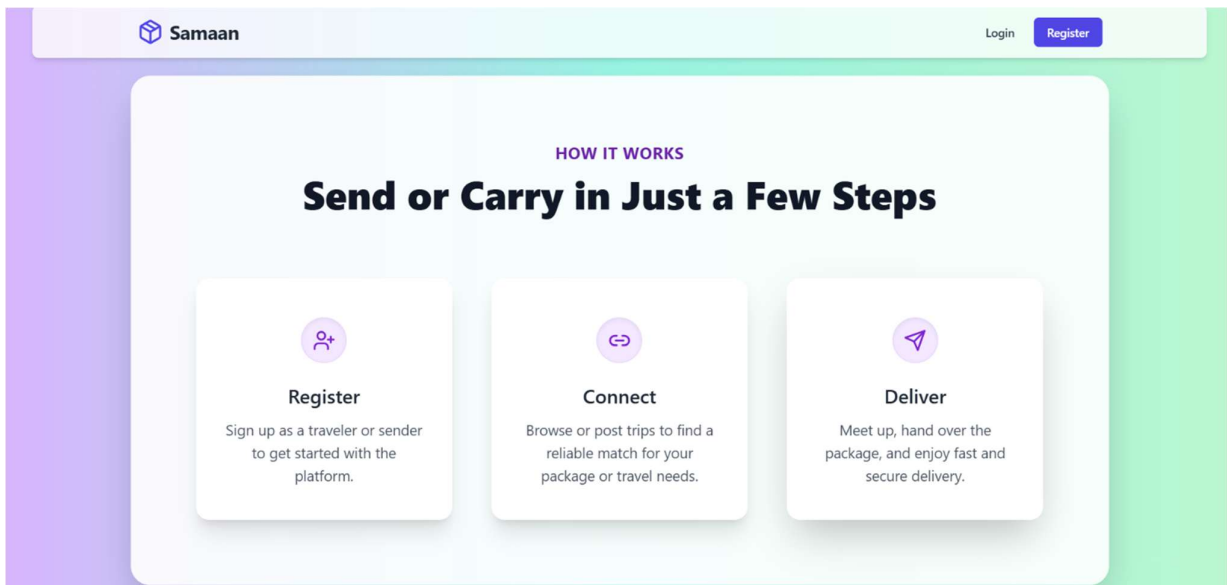


Figure 8.1.1 Home (2)

Description

The homepage of the Samaan application provides users with a clear entry point to the platform. It offers two primary options: "Find a Carrier" for senders looking to ship packages, and "Become a Carrier" for users willing to transport packages. The design emphasizes secure and cost-effective delivery through trusted travellers.

8.2 Register Page

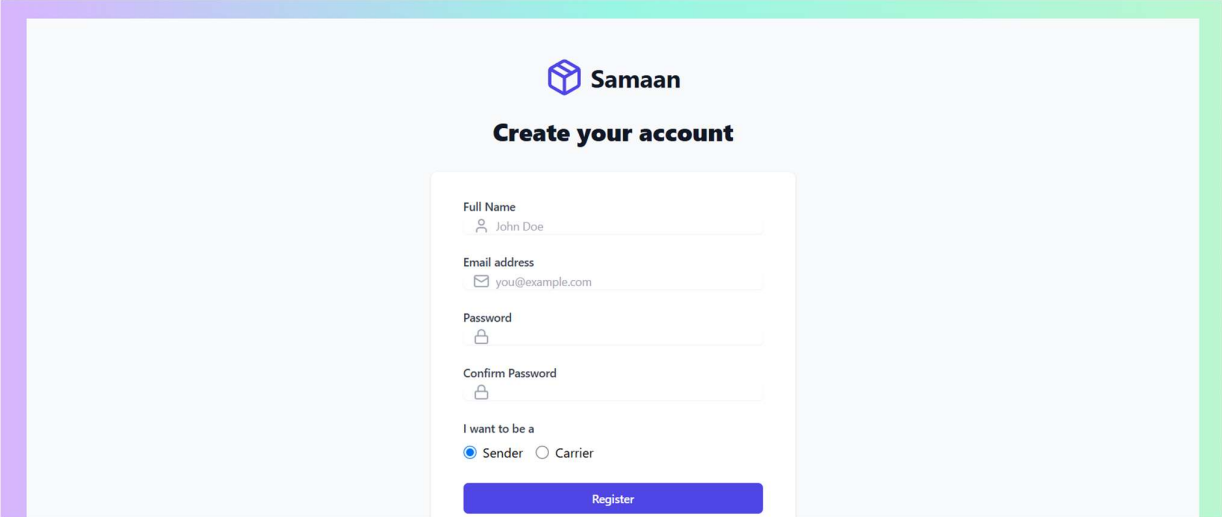
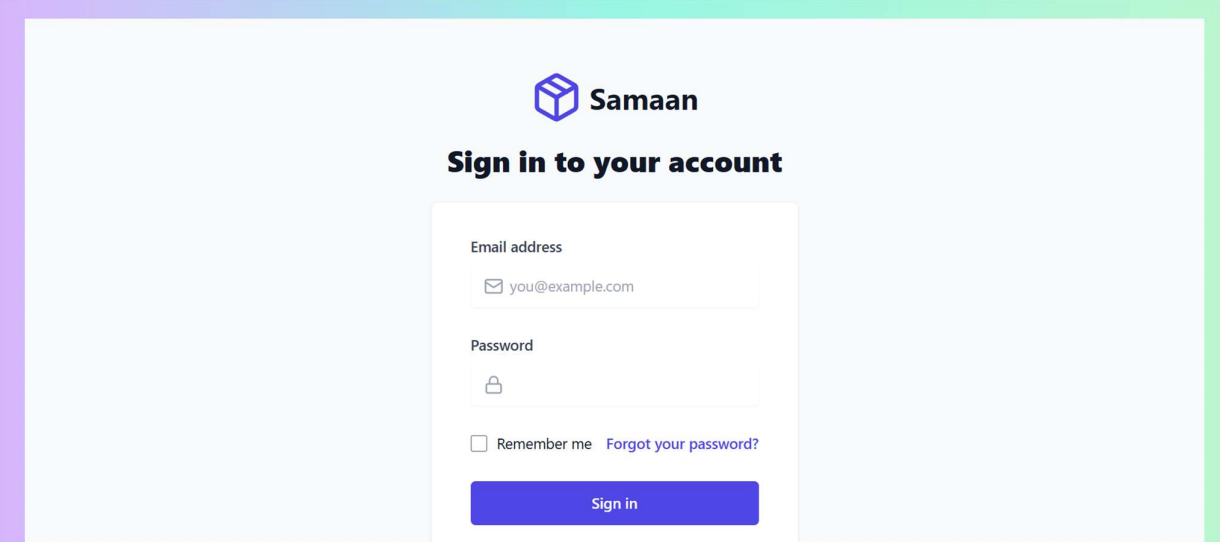
The image shows a web registration form for an application named 'Samaan'. The form is titled 'Create your account' and is set against a light blue background with a purple and green border. It contains several input fields: 'Full Name' with a person icon and the text 'John Doe', 'Email address' with an envelope icon and the text 'you@example.com', 'Password' with a lock icon, and 'Confirm Password' with a lock icon. Below these fields is a section 'I want to be a' with two radio button options: 'Sender' (which is selected) and 'Carrier'. At the bottom of the form is a blue button labeled 'Register'.

Figure 8.2 Register

Description

The Register Page allows users to create an account by entering their details and clicking the "Register" button to submit the information.

8.3 Login Page



Samaan

Sign in to your account

Email address

you@example.com

Password

☐ Remember me [Forgot your password?](#)

Sign in

Figure 8.3 Login

Description

The login page allows users to securely access their accounts by entering their registered email and password. After login, users are redirected based on their role (Sender or Carrier) associated with the registered email. After login as sender users are redirected to search carrier page and after login as carrier users are redirected to carrier dashboard page.

8.4 Carrier Dashboard page

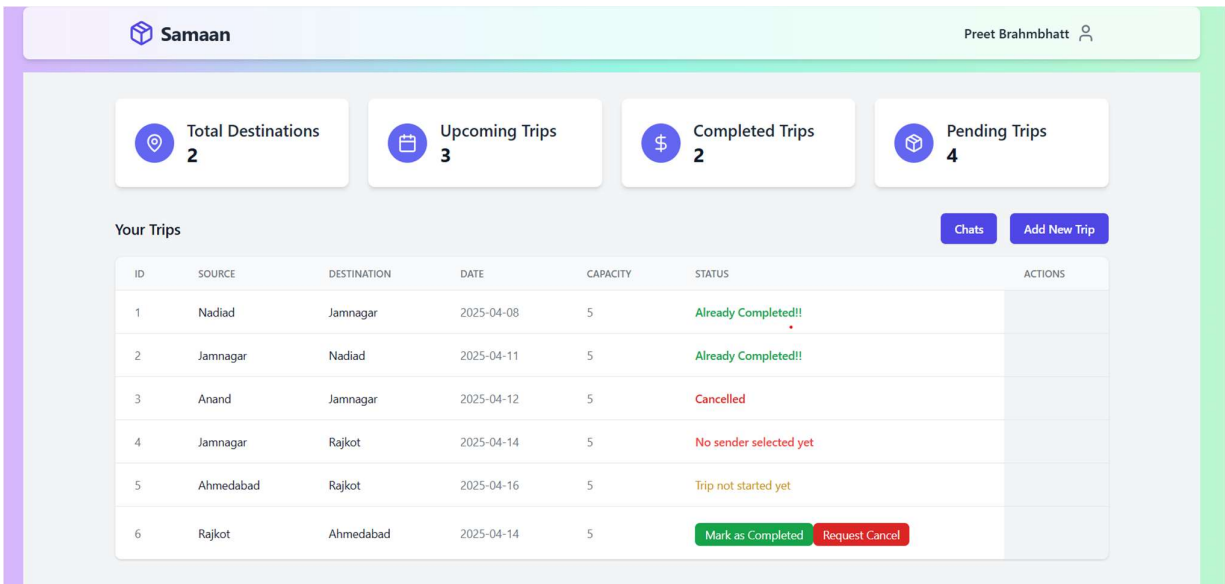


Figure 8.4.0 Carrier Dashboard

Description

This dashboard provides a comprehensive overview of all the trips managed by the carrier. The page displays key statistics at the top in the form of informative cards:

- Total Destinations: Number of unique destination cities covered.
- Upcoming Trips: Number of trips scheduled for future dates.
- Completed Trips: Number of trips that have already been completed.
- Pending Trips: Number of trips yet to be completed or confirmed.

Your Trips Table

Below the summary cards, a detailed table lists all the trips created by the carrier. Each row in the table shows:

- ID: Trip identification number.
- Source & Destination: Start and end cities of the trip.
- Date: Scheduled date of the trip.
- Capacity: Available package capacity for that trip.
- Status: Current status of the trip. Statuses include:

- Already Completed!! – trip is finished.
- Cancelled – trip was cancelled.
- No sender selected yet – no sender has chosen this trip.
- Trip not started yet – trip is scheduled for a future date.
- Actions: Based on the status and date, the carrier can take specific actions like marking a trip as completed or to request cancellation.
- Chats Button: Allows the carrier to view and respond to chat messages from senders.
- Add New Trip Button: Redirects to a form where the carrier can create and publish a new trip.

8.5 Add New Trip Page

The screenshot shows the 'Add New Trip' form in the Samaan application. The form is titled 'Add New Trip' and contains several input fields: Email (pr@gmail.com), Carrier Name (Preet Brahmhatt), Source (Anand), Destination (Ahmedabad), Start Landmark (sanket chowkdi), End Landmark (palladium mall), Vehicle Type (car), Capacity (in kg) (10), and Date (16-04-2025). A 'Submit' button is located at the bottom right of the form.

Figure 8.5 Add New Trip

Description

The Add New Trip form allows carriers to enter trip details including source, destination, vehicle type, capacity, and date. Upon submission, the trip is saved and becomes visible to potential senders searching for delivery options.

8.6 Sender Chat List



Figure 8.6.0 Sender Chat List

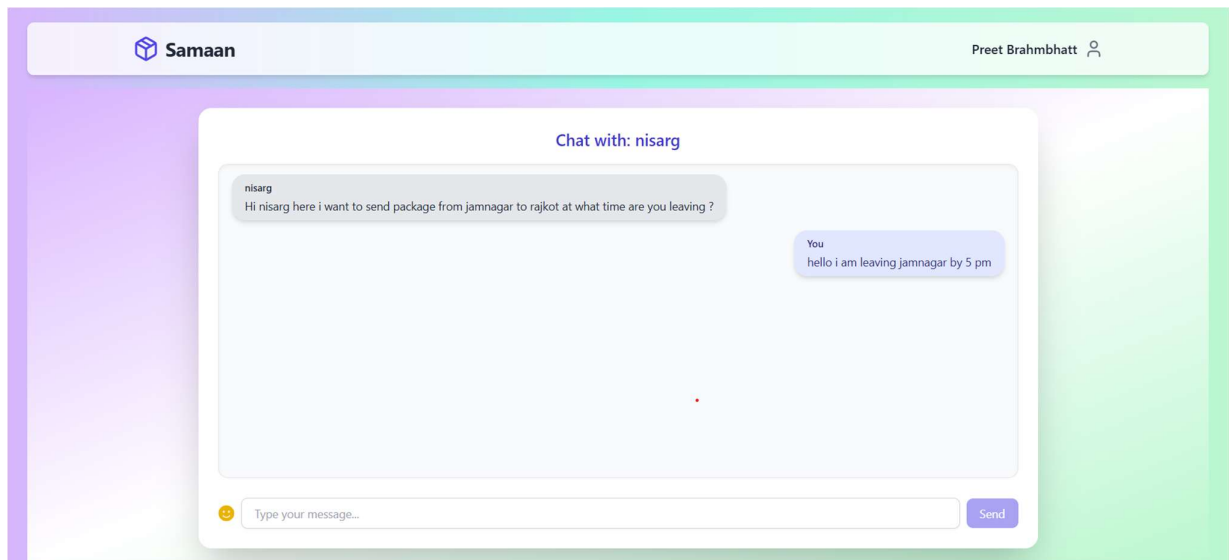
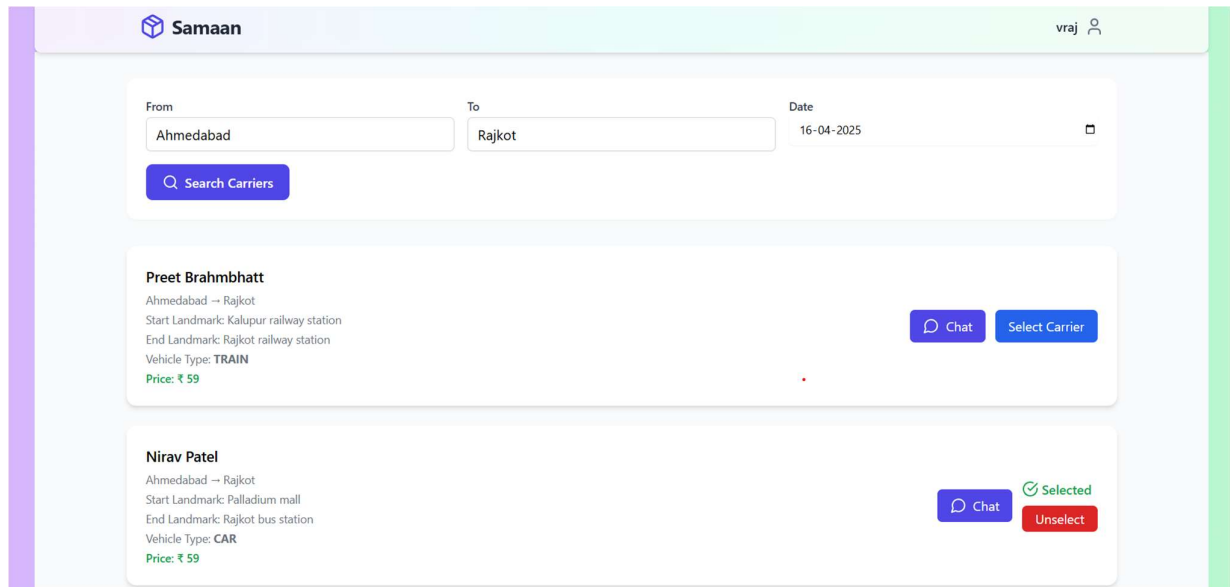


Figure 8.6.1 Personal Chat

Description

This page displays all active chats between the logged in carrier and all the senders who initiated chat. Each chat card shows the sender's name and their most recent message. Carriers can tap on any chat to continue the conversation in real-time.

8.7 Search Carrier Page



The screenshot shows the Samaan Search Carrier Page. At the top, there is a header with the Samaan logo on the left and a user profile icon labeled 'vraj' on the right. Below the header, there is a search form with three input fields: 'From' (Ahmedabad), 'To' (Rajkot), and 'Date' (16-04-2025). A blue button labeled 'Search Carriers' is positioned below the 'From' field. Below the search form, there are two carrier listings. The first listing is for 'Preet Brahmhatt', showing the route 'Ahmedabad → Rajkot', start and end landmarks, vehicle type 'TRAIN', and price '₹ 59'. It includes a 'Chat' button and a 'Select Carrier' button. The second listing is for 'Nirav Patel', showing the route 'Ahmedabad → Rajkot', start and end landmarks, vehicle type 'CAR', and price '₹ 59'. It includes a 'Chat' button, a 'Selected' status indicator with a green checkmark, and an 'Unselect' button.

Samaan vraj

From: Ahmedabad To: Rajkot Date: 16-04-2025

[Search Carriers](#)

Preet Brahmhatt
Ahmedabad → Rajkot
Start Landmark: Kalupur railway station
End Landmark: Rajkot railway station
Vehicle Type: TRAIN
Price: ₹ 59

[Chat](#) [Select Carrier](#)

Nirav Patel
Ahmedabad → Rajkot
Start Landmark: Palladium mall
End Landmark: Rajkot bus station
Vehicle Type: CAR
Price: ₹ 59

[Chat](#) Selected [Unselect](#)

Figure 8.7 Search Carrier

Description

This page allows the sender to search for available carriers by selecting the source, destination, and date. The sender can view trip details, initiate a chat with the carrier, and select or unselect a carrier for their package.

8.8 Sender Dashboard

The screenshot shows the Samaan Sender Dashboard. At the top, there is a header bar with the Samaan logo on the left and the user name 'vraj' with a profile icon on the right. Below the header, the main section is titled 'Your Selected Trips' and includes a 'Your Chats' button. The trips are listed in a table with the following columns: #, Trip ID, Carrier, Source, Destination, Status, and Rating & Feedback.

#	Trip ID	Carrier	Source	Destination	Status	Rating & Feedback
1	#67fce07c38183c4fc9c7c30	Nirav Patel	Ahmedabad	Rajkot	Pending	Request Cancellation
2	#67fd0a49e557977012c9664b	Preet Brahmhatt	Rajkot	Ahmedabad	Completed	★★★★★ very nice Submit Rating & Feedback
3	#67f913c50402430c9a82eb2e	Preet Brahmhatt	Anand	Jamnagar	Cancelled	
4	#67f0c01462a594419903cc54	Preet Brahmhatt	Nadiad	Jamnagar	Completed	★★★★★ Very good Feedback Submitted

Figure 8.8 Sender Dashboard

Description

This page displays all the trips selected by the sender, along with their current status (Pending, Completed, or Cancelled). The sender can request cancellations, view trip details, and submit ratings and feedback for completed trips.

9.0 Limitations and Future Enhancements

9.1 Limitations

Despite its robust features, Samaan has some limitations:

- No Package Insurance Feature – The platform connects senders and carriers but does not provide insurance for lost or damaged packages.
- Limited Real-Time Tracking – The system does not include live GPS tracking of packages; carriers only update their status manually.
- No Automated Price Estimation Based on Distance – The price estimation is based on fixed logic rather than real-time distance calculations.
- Basic Notification System – Notifications for new chats and trip updates exist but lack real-time push notification functionality.
- No Multi-Language Support – The app is currently available only in English, limiting accessibility for non-English speakers.

9.2 Future Enhancements

To improve Samaan, the following features can be added:

- Live GPS Tracking – Enable real-time tracking of packages using carrier GPS data.
- Automated Pricing Algorithm – Implement AI-based pricing based on real-time distance and demand.
- Push Notifications – Enhance notifications with real-time push alerts for new messages and trip updates.
- Package Insurance Integration – Offer optional insurance plans for senders to protect their shipments.
- Multi-Language Support – Expand accessibility by adding support for multiple languages.
- Rating & Review System – Allow senders and carriers to rate each other based on their experience.
- AI-Based Trip Matching – Optimize trip suggestions using machine learning to match senders with the best carriers.

10.0 Conclusion and Discussion

10.1 Conclusions and Future Enhancement

Samaan successfully connects senders and carriers, allowing users to search for trips, send packages, and communicate in real-time through a chat system. The platform ensures secure transactions and enhances user experience with a smooth carrier selection process.

While the current implementation is functional and efficient, adding real-time tracking, automated pricing, and better notification features will improve its usability and efficiency.

10.2 Discussion

10.2.1 Self-Analysis of Project Viabilities

Table 10.1 Self-Analysis of Project Viabilities

Aspect	Analysis
Technical Feasibility	Built with React.js (frontend) and Spring Boot (backend), ensuring high scalability.
Operational Feasibility	User-friendly interface allows senders to find trips and chat easily with carriers.
Economic Feasibility	No need for complex third-party integrations, reducing costs.
Market Potential	Helps individuals efficiently send packages via independent carriers.

10.2.2 Problems Encountered and Possible Solutions

Table 10.2 Problems Encountered and Possible Solutions

Problem	Solution
Missing Room ID during chat initialization	Implemented a room generation system based on sender and carrier emails.
Messages not being saved correctly in the database	Ensured correct mapping of sender and carrier emails while saving messages.
Chat system was duplicating messages on UI	Implemented a message filtering mechanism to prevent duplicate display.
Notification delays	Plan to add push notification support in future updates.

10.2.3 Summary of Project Work

The project analysed existing package delivery platforms, designed an efficient web-based solution, and implemented key features like trip search, chat functionality, and user authentication using Spring Boot (backend) and React.js (frontend).

The system was rigorously tested to ensure reliability, and future enhancements such as real-time tracking, push notifications, and AI-based trip matching were proposed to improve scalability and user experience.

11.0 References

1. MongoDB Inc. (n.d.). MongoDB Documentation.

Retrieved from <https://www.mongodb.com/docs/>

Last Accessed on: 15th April 2025

Official documentation used for implementing schema design and performance optimizations.

2. Spring Team. (n.d.). Spring Boot Documentation.

Retrieved from <https://docs.spring.io/spring-boot/docs/current/reference/html/>

Last Accessed on: 15th April 2025

Used for backend development, security configurations, and RESTful API implementations.

3. Meta (React Team). (n.d.). *React* – A JavaScript library for building user interfaces.

Retrieved from <https://reactjs.org/>

Last Accessed on: 15th April, 2025

Reference for building the responsive and component-based frontend.

4. Render Documentation. (n.d.). Deploying Web Applications on Render.

Retrieved from <https://render.com/docs>

Last Accessed on: 15th April, 2025

Resource used for deploying both frontend and backend services.

5. OpenCage Geocoding API. OpenCage GmbH.

Retrieved from <https://opencagedata.com>

Last Accessed on: 15th April, 2025

Used for converting user-provided localities and landmarks into geographic coordinates (latitude and longitude) to calculate distances between users.

6. Roadie

Retried from <https://www.roadie.com/>

Last Accessed on: 15th April, 2025

A UPS company offering same-day delivery by matching people who need to send items with drivers already heading that way.