GoDeliver

Submitted By

Afsana Jannat

221-35-851

Supervised By

Khalid Been Md. Badruzzaman Biplob Senior

Lecturer

APPROVAL

This thesis titled on "**GoDeliver**", submitted by **Afsana Jannat (ID: 221-35-851)** to the Department of Software Engineering, Daffodil International University has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of Bachelor of Science in Software Engineering and approval as to its style and contents.

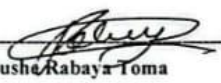
BOARD OF EXAMINERS

Chairman

Dr. S M Hasan Mahmud
Associate Professor
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University

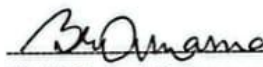
Internal Examiner 1

A.H.M Shahariar Parvez
Associate Professor
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University



Internal Examiner 2

Tapushe Rabaya Toma
Assistant Professor
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University



Internal Examiner 3

Khalid Been md. Badruzzaman Biplob
Lecturer (Senior Scale)
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University

External Examiner

Dr. Md Sazzadur Rahman
Professor
Institute of Information technology
Jahangirnagar University, Bangladesh

DAFFODIL INTERNATIONAL UNIVERSITY

DECLARATION OF THESIS AND COPYRIGHT

Author's Full Name	:	Afsana Jannat
Date of Birth	:	30-11-2003
Title	:	GoDeliver
Academic Session	:	2021-2025

I declare that this thesis is classified as:

☐	CONFIDENTIAL	(Contains confidential information under the Official Secret Act 1997)*
☐	RESTRICTED	(Contains restricted information as specified by the organization where research was done)*
☐	OPEN ACCESS	I agree that my project to be published as online open access (Full Text)

I acknowledge that Daffodil International University reserves the following rights:

1. The Project is the Property of Daffodil International University.
2. The Library of Daffodil International University has the right to make copies of the Project for the purpose of research only.
3. The Library of Daffodil International University has the right to make copies of the Project for academic exchange.

Certified by:

Afsana Jannat
_____
(Student's Signature)

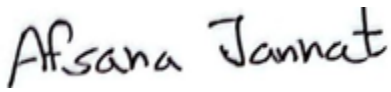
_____
(Supervisor's Signature)

221-35-851
_____
Student ID
Date: 12/24/2025

Khalid Been Md. Badruzzaman Biplob
_____
Name of Supervisor
Date: 12/24/2025

STUDENT'S DECLARATION

I hereby declare that the work in this project is based on my original work except for quotations and citations which have been duly acknowledged. I also declare that it has not been previously or concurrently submitted for any other degree at Daffodil International University or any other institution.



(Student's Signature)

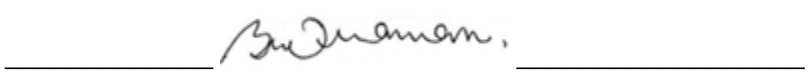
Full Name : **Afsana Jannat**

ID Number : 221-35-851

Date : 24 December 2025

SUPERVISOR'S DECLARATION

I/We* hereby declare that I/We* have checked this project and in my/our* opinion, this project is adequate in terms of scope and quality for the award of the degree of Bachelor of Science.



(Supervisor's Signature)

Full Name : Khalid Been Md. Badruzzaman Biplob
Date : 12/24/2025

GoDeliver

Afsana Jannat

Project submitted in fulfillment of the requirements for
the award of the degree of

Bachelor of Science/Master of Science

Department of Software Engineering

DAFFODIL INTERNATIONAL UNIVERSITY

Decembe 2025

ACKNOWLEDGEMENTS

First of all, we are grateful to The Almighty Allah for making us eligible to complete this thesis.

Then we would like to thank our supervisor "Khalid Been Md. Badruzzaman Biplob", Department of Software Engineering. We are extremely grateful and indebted to her for hers expert, sincere and valuable guidance and encouragement extended to us.

I would also like to thank them who were involved in the survey for this thesis project. Without their passionate participation and input, the validation survey could not have been successfully conducted.

We take this opportunity to record our sincere thanks to all the faculty members of the Department of Software Engineering for their help and encouragement.

Last but not least, we would like to thank to our parents, for their unconditional support, love and without this we would not have come this far.

ABSTRACT

GoDeliver project is a simple online parcel delivery system that connects customers, riders and admins on one platform. Customers can book parcels, pay online and track deliveries. Riders receive assigned tasks, update pickup and delivery statuses and admins manage users, parcels, payments and reports.

The system makes delivery faster, clearer and more organized by replacing manual work with digital processes.

How it works:

Customers: Register, book parcels, pay online and track parcels in real-time.

Riders: Accept tasks, mark pickups and deliveries and update parcel status.

Admin: Monitor the system, manage users and parcels, handle payments and generate reports.

Methodology:

The system uses modern web technologies with a front-end for users, a back-end server and a database for storing information. Real-time tracking shows the status of the parcel and secure payment gateways ensure safe transactions.

Overall, GoDeliver makes the delivery process efficient, transparent, and easy for everyone.

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

LIST OF APPENDICES

Appendix A: Coding

CHAPTER 1 — INTRODUCTION

1.1 Background

In today's busy world, people want fast and reliable parcel delivery services. Many courier systems are slow, difficult to understand and do not provide proper tracking or clear proof after delivery. Due to poor management, both businesses and ordinary people face delays and sometimes parcels get lost.

GoDeliver is an online parcel delivery system that tries to solve these problems by connecting customers, admins and riders in one place. It allows users to book parcels from home or office, pay online and track their parcels in real time. Admins can manage service centers and assign tasks to riders for pickup and delivery. Riders can update the status of parcels and confirm delivery using OTP or tracking number.

The goal of this project is to make delivery of parcel in BD more easier and faster.

1.1.1 Context and Relevance

In Bangladesh only ecommerce businesses are growing rapidly, and because of this the need of fast and Efficient delivery is increasing day by day. Even though we have some small courier companies still working manually, but this approach leads to delays and errors. This is the problem GoDeliver solves by Automates the delivery process by having a smart dashboard for riders, admins and users.

The main features of this project are the web tools and real time updates which makes managing delivery very simple and smooth. This is true for both inside and outside of the city, allowing all the users to send parcels anywhere in Bangladesh without much issues.

1.1.2 Problem Identification

Major problems current delivery system has are mentioned below:

- Payment issues and no digital receipts
- No proper way to track passenger earnings or responsibilities
- Neglect of clear information or digital tracking
- Poor communication and manual assignment of parcels.
- No real time updates on the parcel current status.

GoDeliver is designed to solve all these problems with automated processes, online payments and complete parcel tracking from pickup to delivery.

1.1.3 Purpose and Justification

The main goal of this project is to make parcel delivery simple, reliable and transparent for both customers and

service providers. It helps:

- Customers book and track parcels online
- Administrators manage parcels, payments and riders
- Riders confirm deliveries and get paid automatically

This project also supports the use of digital tools in the delivery business and helps Bangladesh move towards smart and online services.

1.1.4 Scope

The system includes:

- Online parcel booking and payment
- Real-time parcel tracking
- Admin dashboard to manage parcels and riders
- Rider dashboard to confirm pickup and delivery
- User dashboard to view parcel history, payments and give feedback

This version does not include warehouse automation or live GPS tracking in this system, but these features may be added later.

1.2 Project Planning and Initiation (Feasibility Study)

Phase 1: Preliminary Analysis & Project Scope Definition

The goal is to create a user-friendly website where anyone can send parcels, do online payment and check the status of their ordered parcels easily. Admins can control the full work process and riders will manage the main delivery of the parcels.

Phase 2: Market Feasibility Analysis

In Bangladesh services related to courier and parcel delivery are widely used. In recent years, with the rise of online shopping, people want highly reliable and fast delivery services which are easy to track. My project fulfills this role effectively.

Phase 3: Technical Feasibility Analysis

The project uses technologies:

- **Frontend:** React.js
- **Backend:** Node.js, Express.js
- **Database:** MongoDB Atlas
- **Payment Gateway:** Stripe
- **Authentication:** Firebase

These technologies are easy to use, open-source, and good for building web applications that can scale.

Phase 4: Financial Feasibility Analysis

As it's a public website the cost of hardware is very less, the main cost is in the cloud deployment and monthly subscriptions or payments and hosting. One of them is the Stripe payment system which charges a small fee per successful transaction, but in return it provides a secured and reliable payment process.

1.3 Target User Profile and Elicitation Process

1.3.1 Target Users

- General users who want to send parcels
- Admins who manage delivery tasks
- Riders who pick up and deliver parcels

1.3.2 User Profiles

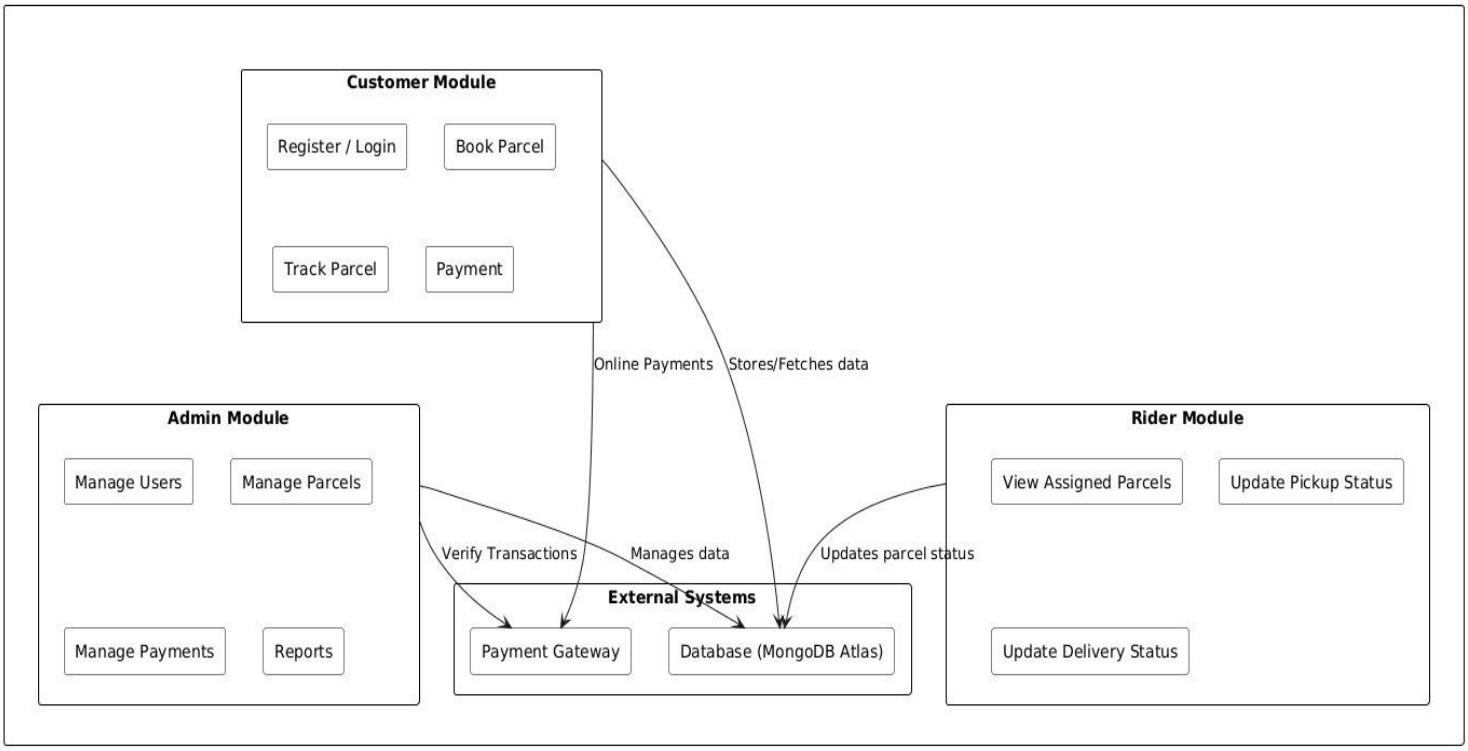
User Type	Age Range	Computer Experience	Frequency of Use	Education	Goal	Language	Training Needed
User	18–45	Medium	Frequent	Varies	Send and track parcels easily	Bangla/English	None
Admin	25–40	High	Daily	Graduate	Manage and monitor delivery system	English	None
Rider	18–35	Basic	Daily	Varies	Handle pickup and delivery tasks	Bangla	Short training

1.3.3 Elicitation Process

To gather user requirements, a few methods were used:

1. **Interviews:** Interviewed some of the local users and riders to find out their issues.
2. **Surveys:** Interviewed some of the local users and riders to find out their issues.
3. **Observation:** Observed how an actual delivery process executes, from user order to rider pickup to final delivery of the package.

1.4 Project Block Diagram



1.5 System Requirements

1.5.1 Hardware Requirements

- Processor: Core i3 or higher
- RAM: Minimum 4 GB
- Hard Disk: 2 GB free space
- Internet Connection: Required

- Display: 1366x768 resolution or higher

1.5.2 Software Requirements

- Operating System: Windows / macOS / Linux
- Development Tools: VS Code, Node.js
- Browser: Chrome, Firefox, or Edge
- Database: MongoDB Atlas
- Hosting: Vercel (frontend), Render (backend)

1.5.3 Constraints and Dependencies

- User needs to have a stable internet connection for real time tracking and payment process
- All online payments depends on availability of Stripe
- All authentication depends on Firebase
- Hosting needs to support both Nodejs and MongoDB

1.6 Project Scheduling

The project was divided into 6 stages:

Phase	Task	Duration
1	Requirement analysis & planning	2 weeks
2	UI/UX design	3 weeks
3	Backend development	6 weeks
4	Frontend integration	6 weeks
5	Testing & bug fixing	3 weeks
6	Final report & presentation	2 week

Total Duration: **22 weeks**

1.7 Summary

This section explains the purpose, importance and background of this system. It also discusses who will actually use this system, the projects feature and analyze if it is feasible to build in terms of cost and technologies. It shows the basic requirements and planning steps for the system.

The next chapter will describe the design of the system and how it is built, including use case diagrams, activity diagrams, ER diagrams, and coding structures.

CHAPTER 2 — DESIGN AND IMPLEMENTATION

2.1 Introduction

This chapter explains how the GoDeliver system is planned and built. It includes functional and non-functional requirements, UML design diagrams, database design, and major development details. The purpose of this chapter is to show how the system structure, user roles, and workflows are built for parcel booking, payment, tracking, and delivery tasks.

2.2 Functional Requirements

ID	Requirement Name	Description	Stakeholders
FR01	User Registration	Users must register using name, email, password	User
FR02	Login & Authentication	Login using Firebase Authentication.	User, Admin, Rider
FR03	Add Parcel	User can add parcel details (type, sender address, receiver address, etc.)	User
FR04	Calculate Delivery Cost	Calculates delivery cost dynamically based on parcel type, distance, and weight.	User
FR05	Payment System	User can pay using Stripe payment gateway.	User
FR06	Tracking System	Users and Admins can track parcel status in real-time.	User, Admin
FR07	Manage Riders	Admin can approve, reject, or edit rider details	Admin
FR08	Manage Parcel Delivery	Admin can assign riders and edit Parcel delivery stages.	Admin
FR09	Pickup & Delivery Confirmation	Riders can confirm pickup and delivery With tracking number/id.	Rider

2.3 Non-Functional Requirements

Category	Description
Performance	The system will respond to user actions within 3 seconds and support at least 100 active users simultaneously.
Reliability	All booking and payment data are stored securely in MongoDB Atlas to avoid data loss.
Scalability	The backend can handle many users and can grow using cloud hosting.
Portability	The web app can be run on any modern browser and mobile device.
Security	Firebase authentication keeps user login details and payments secure.
Maintainability	Modular code design makes updating and debugging easier.
Usability	The dashboard is simple, easy to use, and works well on mobile.

2.4 Object-oriented System design using UML

2.4.1 Use Case Diagram

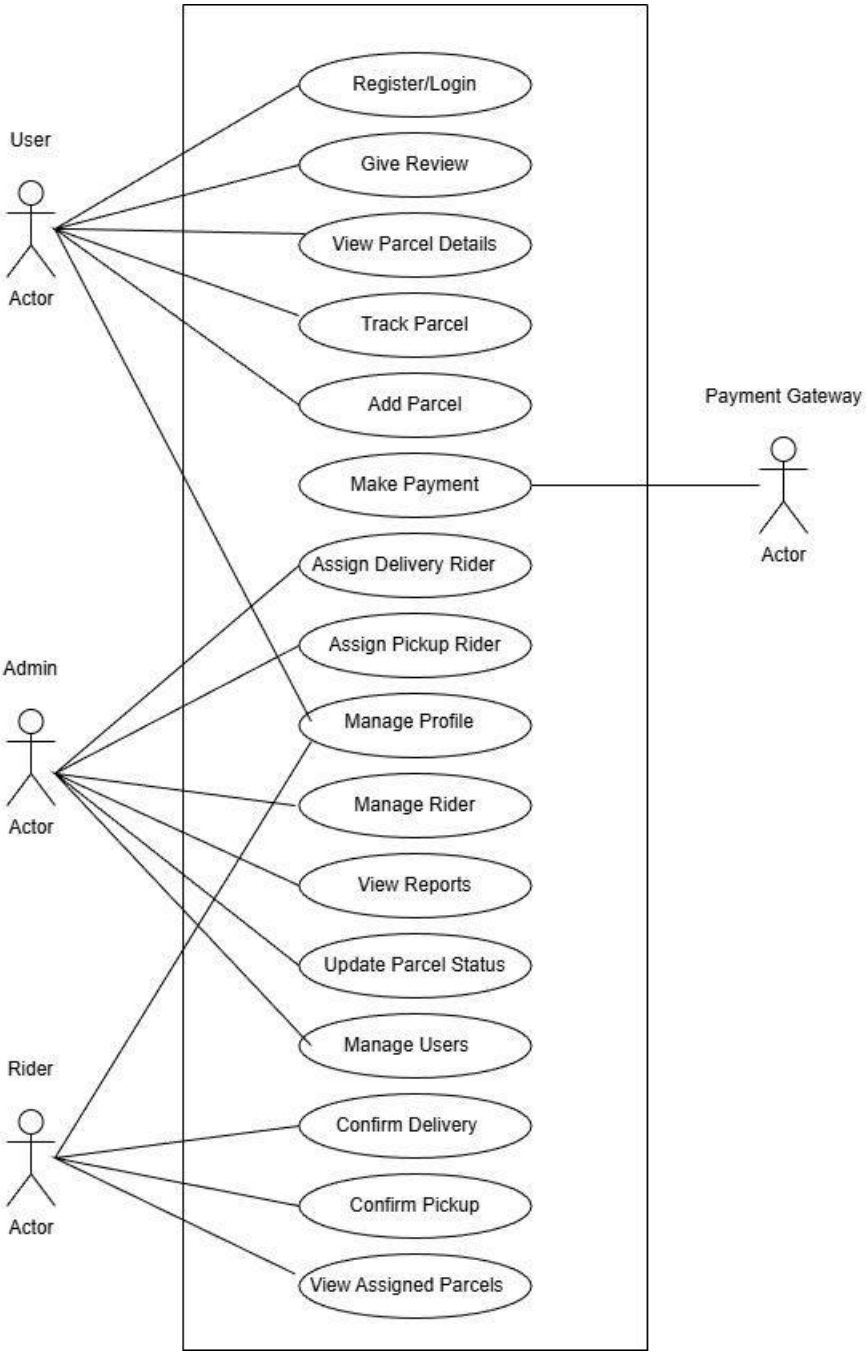


Figure 2: Use case Diagram

2.4.2 Case Description

Case Description – 01: Register

Use Case	User Register
Goal	Users can enter the system by the login.
Precondition	User must have a valid email address.
Success End Condition	System redirects user to their dashboard based on role (User/Rider/Admin).
Failed End Condition	User cannot register or login.
Primary Actors: **Secondary Actors:**	Customer System Database
Trigger	User clicks the **Register** or **Login** button from the website.
Description / Main Success Scenario	1. Press "register/log in" Bar 2. Enter user id and password. 3. Verified and login 4. Notification: "Login Successful"
Alternative Flows	1.1 System Error 1.1.a. Try again 2.1 The system Did not respond. 2.1. a. Show error message. 3.1 Information Error!! 3.1. a. Notification: "Enter the right User ID and Password."
Quality Requirements	Users fill up the Sign in info within 1 minutes.

Case Description – 02: Give Review

Use Case	Give Review
Goal	users to submit a review and rating successfully
Precondition	User must be logged in.
Success End Condition	Review is saved successfully in the database.
Failed End Condition	System shows an error message.
Primary Actors: **Secondary Actors:**	User System Database
Trigger	User clicks the **"Add Review"** button on the Review page.
Description / Main Success Scenario	1. User clicks the **"Add Review"** button. 2. User clicks **Submit**. 3. User writes review and selects a rating (1–5). 4. System saves the review to the database.
Alternative Flows	1.1 System Error 1.1.a. Try again 2.1 Database Error 2.1. a. Show error message. 3.1 Information Error!! 3.1. a. Notification: "Enter the right User ID and Password."
Quality Requirements	Rating must be between **1 and 5 stars**.

Case Description – 03: View Parcel Details

Use Case	View Parcel Details
Goal	Sell all parcel details
Precondition	User/Admin/Rider must be logged in.
Success End Condition	Parcel information is displayed correctly.
Failed End Condition	System shows an error message: **"Parcel not found."**
Primary Actors: **Secondary Actors:**	User, Admin, Rider System Database
Trigger	User/Admin/Rider clicks on **"View"** button from parcel list, payment history, or tracking section.
Description / Main Success Scenario	1. User/Admin/Rider clicks **"View" button** 2. System receives the parcel ID from the UI. 3. System allows user to go back to previous page.
Alternative Flows	1.1 Parcel Not Found 1.1.a. System shows: **"Parcel not found."** 2.1 User has No Permission 2.1. a. System checks role and detects restricted access. 3.1 Information Error!! 3.1. a. Notification
Quality Requirements	Parcel details must load within **2–3 seconds**.

Case Description – 04: Track Parcel

Use Case	Track Parcel
Goal	Allow users to track the real-time status
Precondition	User must be logged in.
Success End Condition	Parcel information is displayed correctly.
Failed End Condition	Parcel tracking cannot be found.
Primary Actors: **Secondary Actors:**	User Admin, Rider
Trigger	User goes to the **Tracking Page** or clicks **"Track"** from any parcel list.
Description / Main Success Scenario	1. User navigates to the **Track Parcel** page. 2. System fetches parcel data from the tracking collection. 3. System allows user to go back to previous page.
Alternative Flows	1.1 Invalid Tracking Number 1.1.a. System cannot find the parcel. 2.1 No Tracking Updates 2.1. a. Parcel exists but no tracking docs added yet. 3.1 Database/Network Error 3.1. a. Tracking data cannot be fetched.
Quality Requirements	Tracking page should load within **2–3 seconds**.

Case Description – 05: Add Parcel

Use Case	Add Parcel (User)
Goal	Users to create a new parcel booking
Precondition	User must be logged in.
Success End Condition	Parcel information is successfully saved in the database.
Failed End Condition	System shows validation or server error.
Primary Actors: **Secondary Actors:**	User System Database
Trigger	User clicks the **"Add Parcel"** button on the Add Parcel page.
Description / Main Success Scenario	1. User navigates to the **Add Parcel** page. 2. User selects parcel type (document / non-document). 3. User enters parcel title and weight (if non-document). 4. User clicks **Submit**.
Alternative Flows	1.1 System Error 1.1.a. System shows: **"Please fill all required fields."** 2.1 Missing Required Fields 2.1. a. User enters invalid weight (negative or letters). 3.1 Database Error 3.1. a. System shows: **"Failed to add parcel. Try again later."**
Quality Requirements	Cost must calculate instantly after clicking submit.

Case Description – 06: Make Payment

Use Case	Make Payment
Goal	Allow users to pay for their parcel
Precondition	User must be logged in.
Success End Condition	Payment is successfully completed.
Failed End Condition	Payment fails or is cancelled.
Primary Actors: **Secondary Actors:**	User System Database, Stripe/Payment Gateway
Trigger	User clicks the **"Add Parcel"** button on the Add Parcel page.
Description / Main Success Scenario	1. User navigates to the **Add Parcel** page. 2. User selects parcel type (document / non-document). 3. User enters parcel title and weight (if non-document). 4. User clicks **Submit**.
Alternative Flows	1.1 System Error 1.1.a. System shows: **"Please fill all required fields."** 2.1 Missing Required Fields 2.1. a. User enters invalid weight (negative or letters). 3.1 Database Error 3.1. a. System shows: **"Failed to add parcel. Try again later."**
Quality Requirements	Cost must calculate instantly after clicking submit.

Case Description – 07: Assign Parcel for Pickup (Admin)

Use Case	Assign Parcel for Pickup
Goal	Allow the Admin to assign a pickup rider for a paid parcel.
Precondition	Admin must be logged in.
Success End Condition	Pickup rider is assigned successfully.
Failed End Condition	Rider is not assigned.
Primary Actors: **Secondary Actors:**	Admin System Database, pickup Rider
Trigger	Admin clicks the **"Assign Rider"** button on the Assign Rider page.
Description / Main Success Scenario	1. Admin navigates to the **AssignRider** page. 2. Admin selects a **paid** parcel and clicks **Assign for Pickup**. 3. Admin clicks **Assign**. 4. System shows success message: → **"Pickup assigned successfully."**
Alternative Flows	1.1 No Available Rider 1.1.a. Admin cannot assign. 2.1 Admin Cancels Assignment 2.1.a. No update is saved. 3.1 Database Error 3.1. a. System fails to update parcel information.
Quality Requirements	Assignment should take **less than 10 seconds**.

Case Description – 08: Manage Riders (Admin)

Use Case	Manage Riders
Goal	Allow the admin to view, approve, or reject rider applications and change their status and system access level.
Precondition	Admin must be logged in.
Success End Condition	Rider status is successfully updated.
Failed End Condition	System does not update DB or role.
Primary Actors: **Secondary Actors:**	Admin System Database, Rider
Trigger	Admin clicks on **"Manage Riders"** page from the dashboard menu.
Description / Main Success Scenario	1. Admin navigates to the **Manage Riders** page. 2. Admin clicks desired action (Approve/Reject). 3. A confirmation alert is shown. 4. System updates the changes to the database.
Alternative Flows	1.1 Admin Cancels Action 1.1.a. Admin cancels confirmation popup. 2.1 Database Error 2.1. a. System fails to update parcel information.
Quality Requirements	Only Admin can modify rider role or status.

Case Description – 09: Manage Users (Admin)

Use Case	**Manage Users**	
Goal	Allow Admin to view all registered users, search/filter users, and update user roles between **User** and **Admin** when necessary to maintain proper system control and authorization.	
Precondition	Admin must be logged in.	
Success End Condition	User role is successfully updated in the system database.	
Failed End Condition	Role change fails.	
Primary Actors: **Secondary Actors:**	Admin System Database	
Trigger	Admin clicks **"Manage Users"** from dashboard navigation.	
Description / Main Success Scenario		
	1.	Admin navigates to the **Manage Users** page.
	2.	System displays confirmation popup.
	3.	Admin confirms the action.
Alternative Flows		
	1.1	Admin Cancels Operation
		1.1.a. Admin cancels confirmation popup
	2.1	Database Error
		2.1. a. Database fails to update user role.
Quality Requirements	Only Admin can access this page (role-based authorization).	

Case Description-10: Confirm Delivery (Rider)

Use Case	**Confirm Delivery**
Goal	allows a rider to confirm that a parcel has been successfully delivered
Precondition	Rider must be logged in.
Success End Condition	Parcel is marked as **Delivered** successfully.
Failed End Condition	Parcel status is not updated due to an error or failed delivery attempt.
Primary Actors: **Secondary Actors:**	Rider Admin, System Database
Trigger	Rider clicks the **"Confirm Delivery"** button on the parcel details page.
Description / Main Success Scenario	1. Rider navigates to the assigned parcels list. 2. Rider selects a parcel for delivery. 3. Rider selects a parcel for delivery.
Alternative Flows	1.1 Rider cancels confirmation: 1.1.a. Parcel status remains unchanged. 2.1 Database Error 2.1. a. Database fails to update delivery status.
Quality Requirements	Confirmation should take less than 10 seconds.

Case Description-12: Confirm Pickup (Rider)

Use Case	**Confirm** Pickup
Goal	Allow the Rider to confirm that a parcel has been successfully picked up from the sender.
Precondition	Rider must be logged in.
Success End Condition	Parcel is marked as **Picked Up** successfully
Failed End Condition	Parcel status is not updated due to an error or failed pickup attempt.
Primary Actors: **Secondary Actors:**	Rider Admin, System Database
Trigger	Rider clicks the **"Confirm Pickup"** button on the parcel details page.
Description / Main Success Scenario	1. Rider navigates to the assigned parcels list. 2. Rider selects a parcel to pick up. 3. Rider verifies sender and parcel details.
Alternative Flows	1.1 Sender not available: 1.1.a. Rider marks pickup as **Failed** or **Rescheduled**. 2.1 Database Error 2.1. a. Database fails to update pickup status.
Quality Requirements	Pickup confirmation should take less than 10 seconds.

Case Description-13: View Assigned Parcels

Use Case	**Confirm** Pickup
Goal	Allow the Rider to view all parcels assigned to them for pickup or delivery.
Precondition	Rider must be logged in.
Success End Condition	Rider can see a complete list of all assigned parcels with their details.
Failed End Condition	Assigned parcels are not displayed due to a system error or no parcels assigned.
Primary Actors: **Secondary Actors:**	Rider Admin, System Database
Trigger	Rider clicks on the **"Assigned Parcels"** menu in the app.
Description / Main Success Scenario	1. Rider navigates to the assigned parcels section. 2. System fetches all parcels assigned to the rider. 3. Rider can select a parcel to view more details or start pickup/delivery.
Alternative Flows	1.1 No parcels assigned: 1.1.a. System displays message: **"No assigned parcels at the moment."** 2.1 Database Error 2.1. a. System fails to fetch parcels and shows an error message.
Quality Requirements	Parcel list should load within 5 seconds.

Activity Diagram

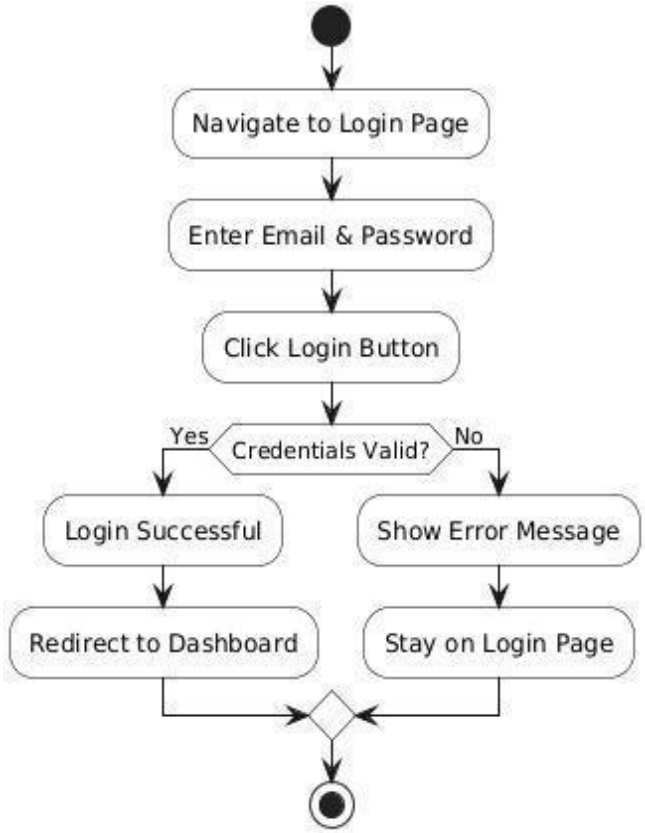


Figure 1: Login page

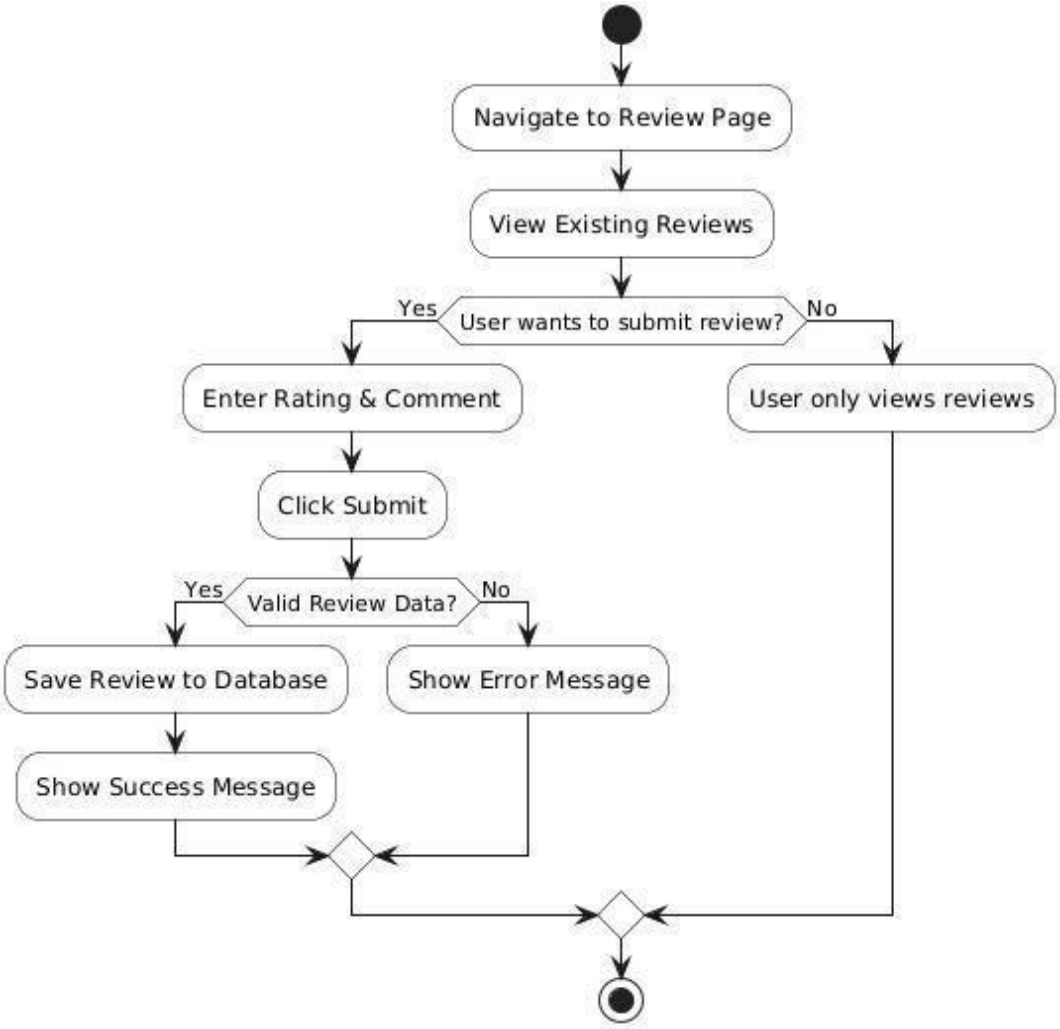


Figure 2: Review page

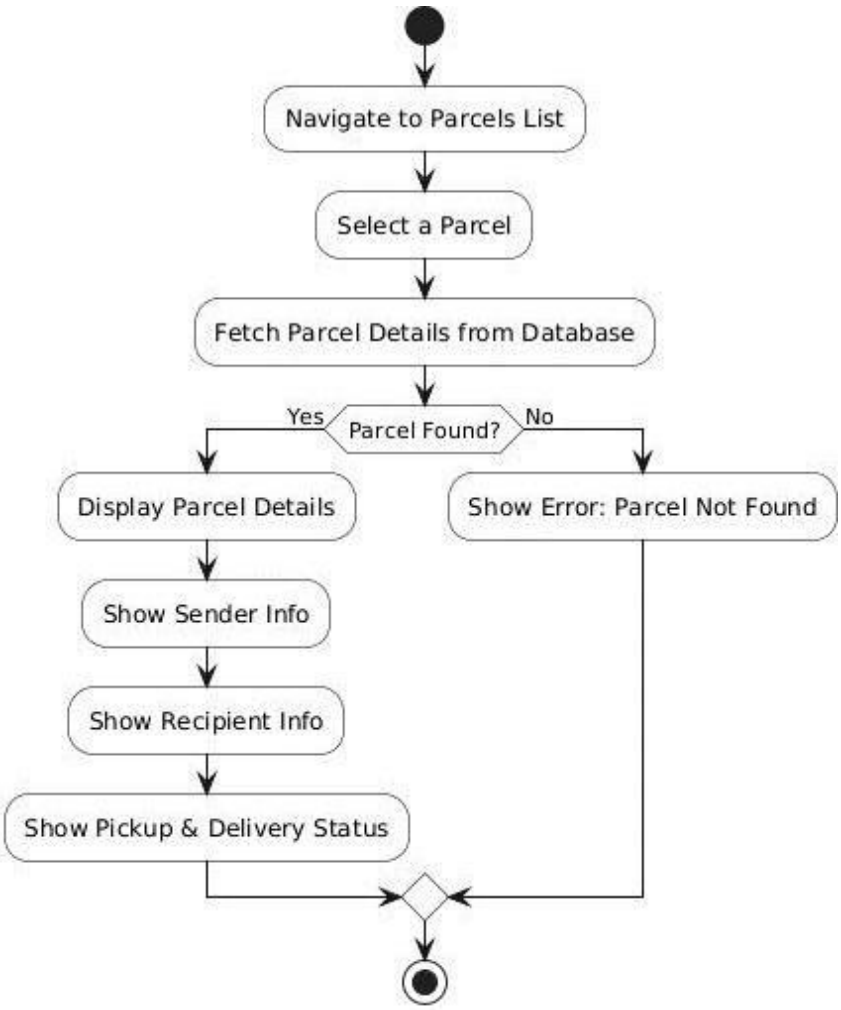


Figure 3: View Parcel Details

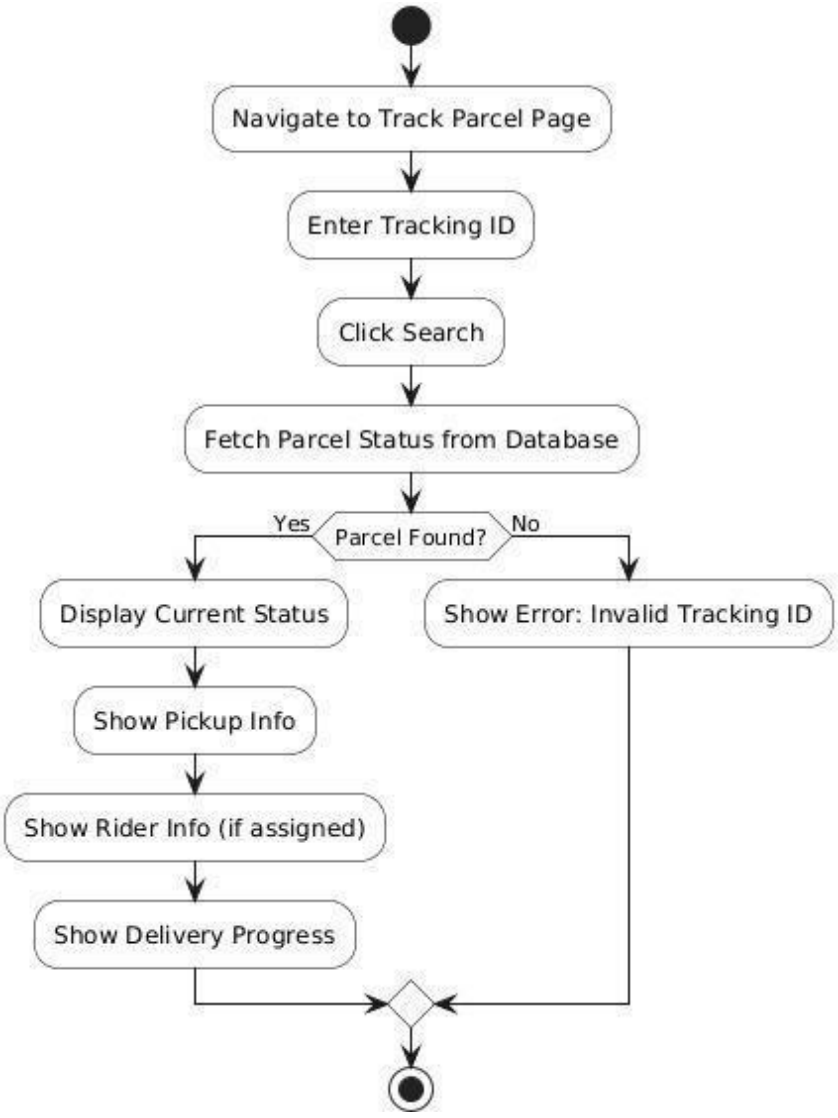


Figure 4: Track Parcel

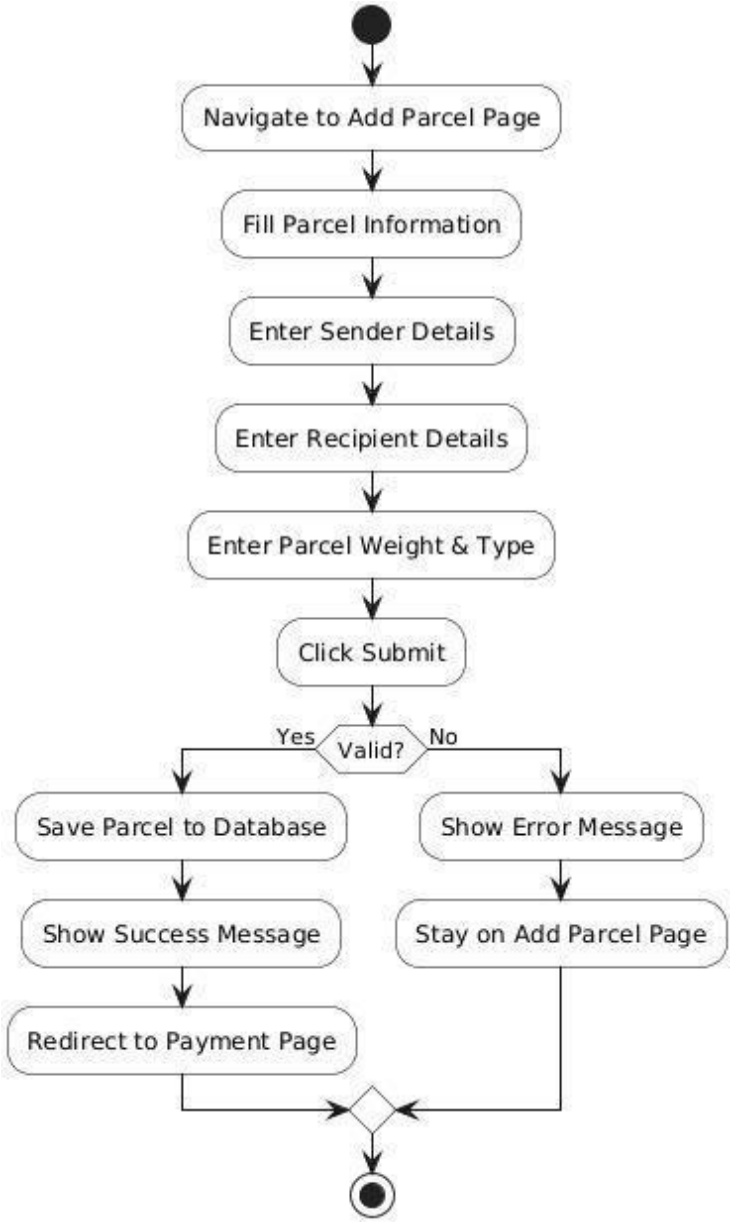


Figure 5: Add Parcel Page

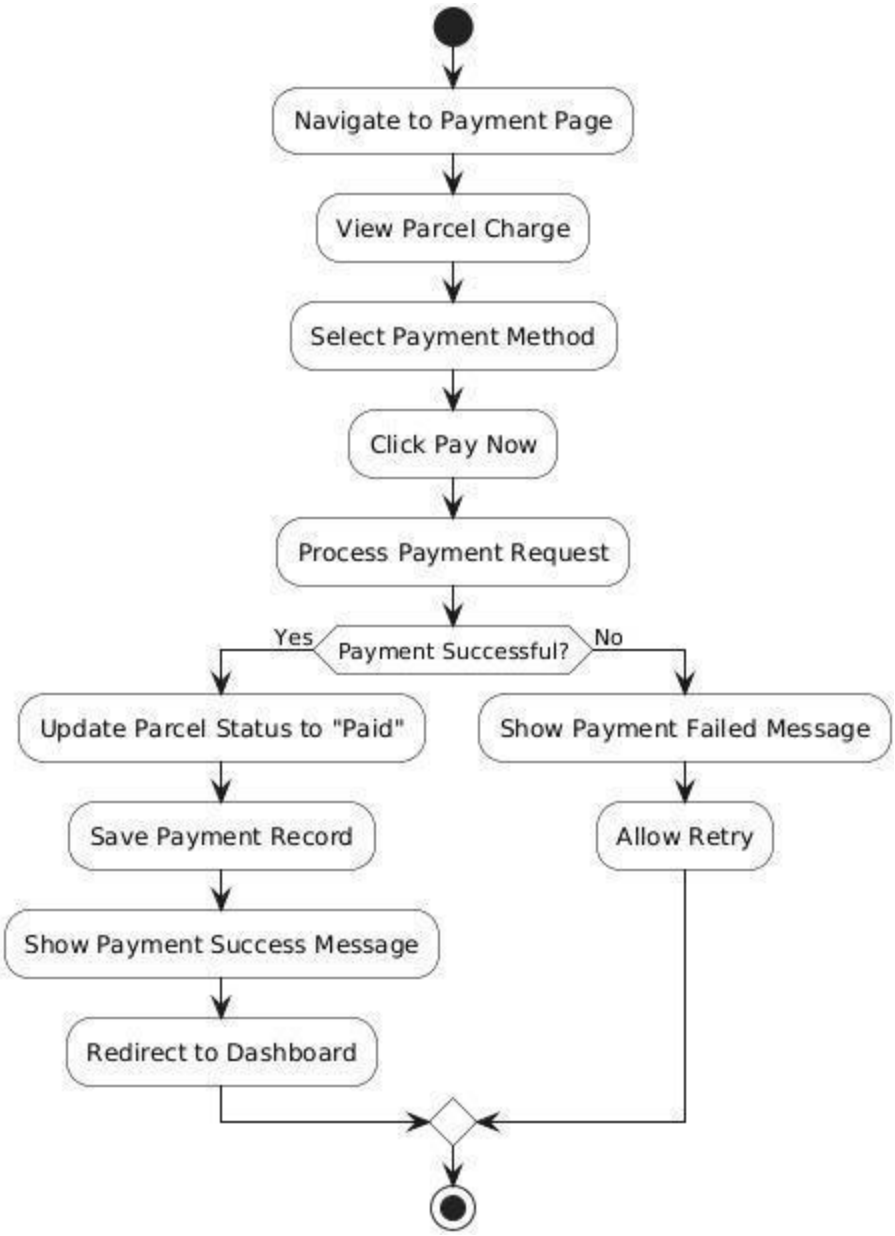


Figure 6: Payment Page

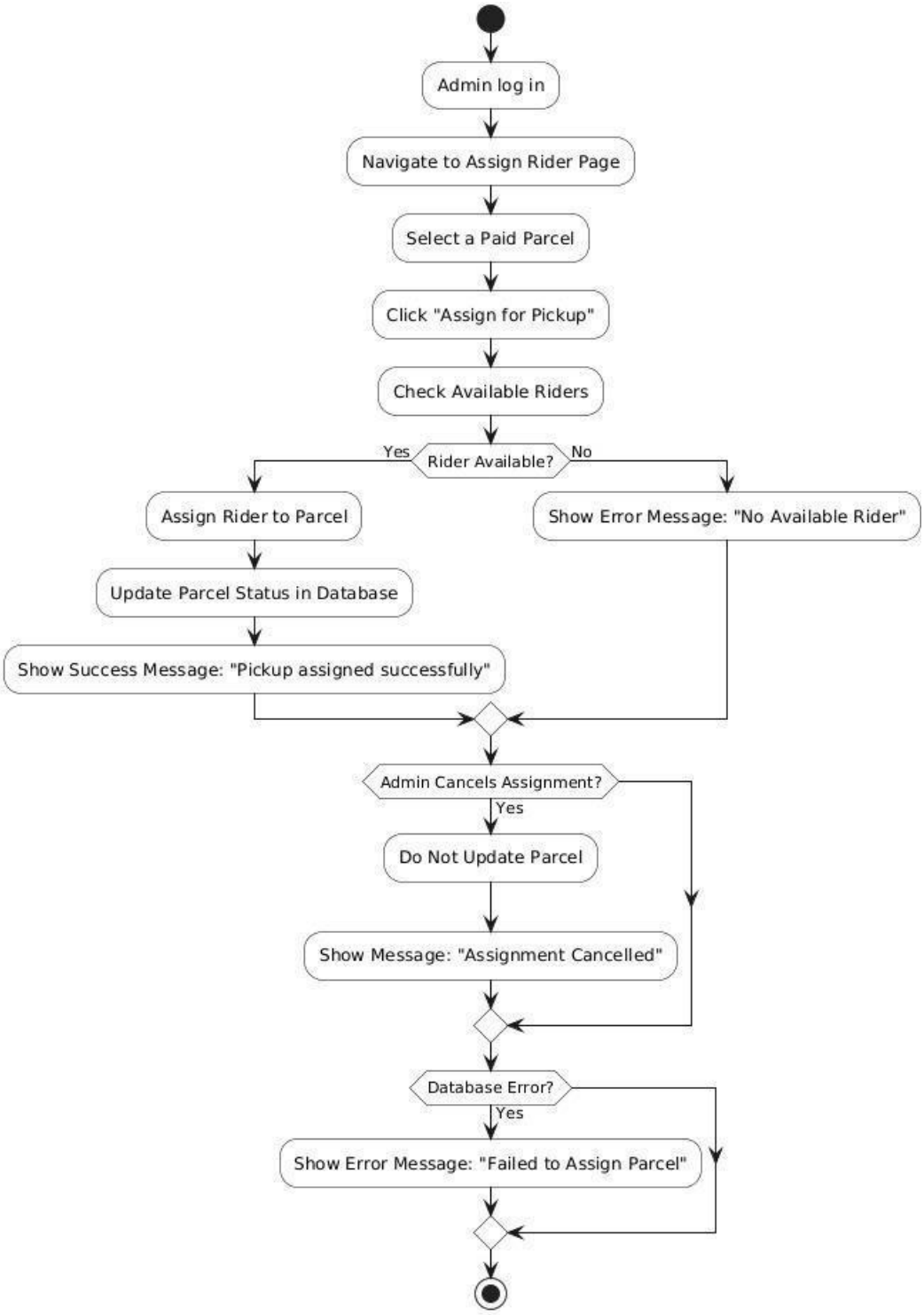


Figure 7: Assign Rider Parce

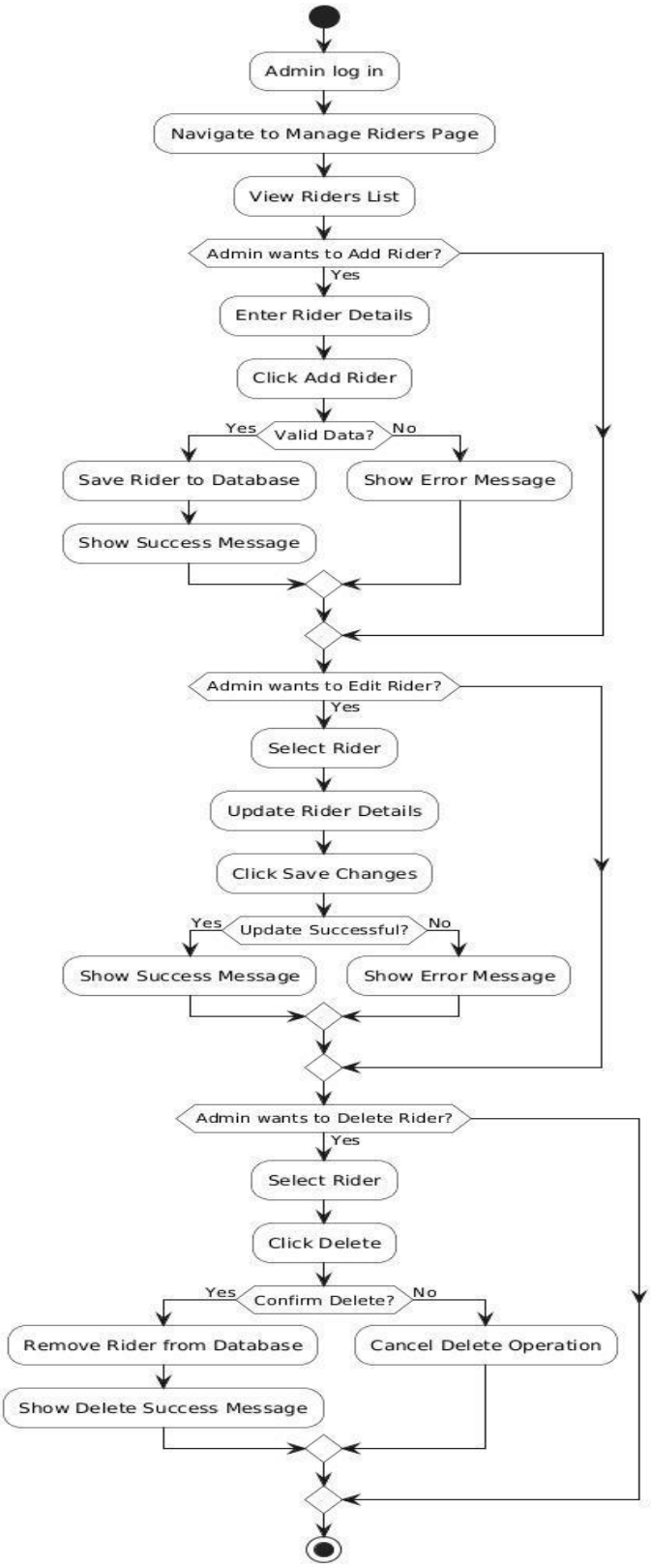


Figure 8: Manage Rider page

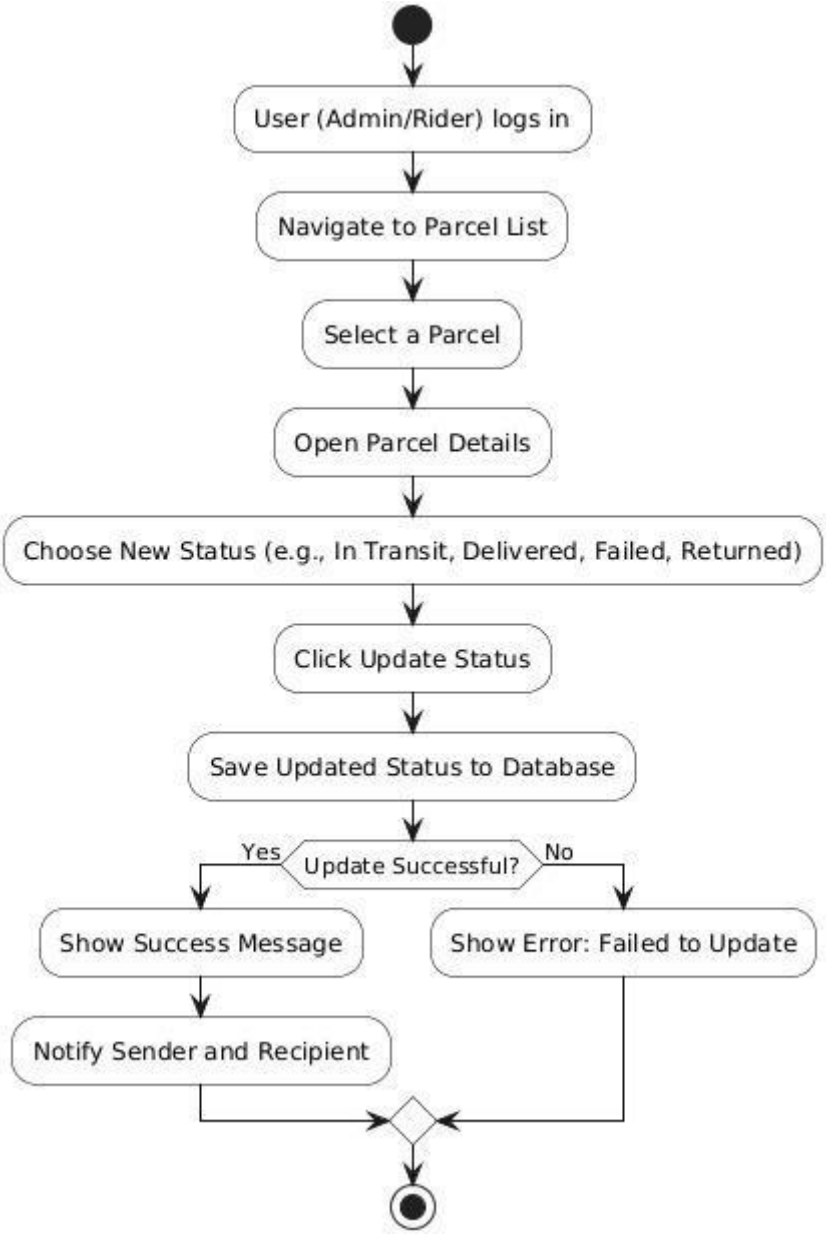


Figure 9: Update Parcel Status

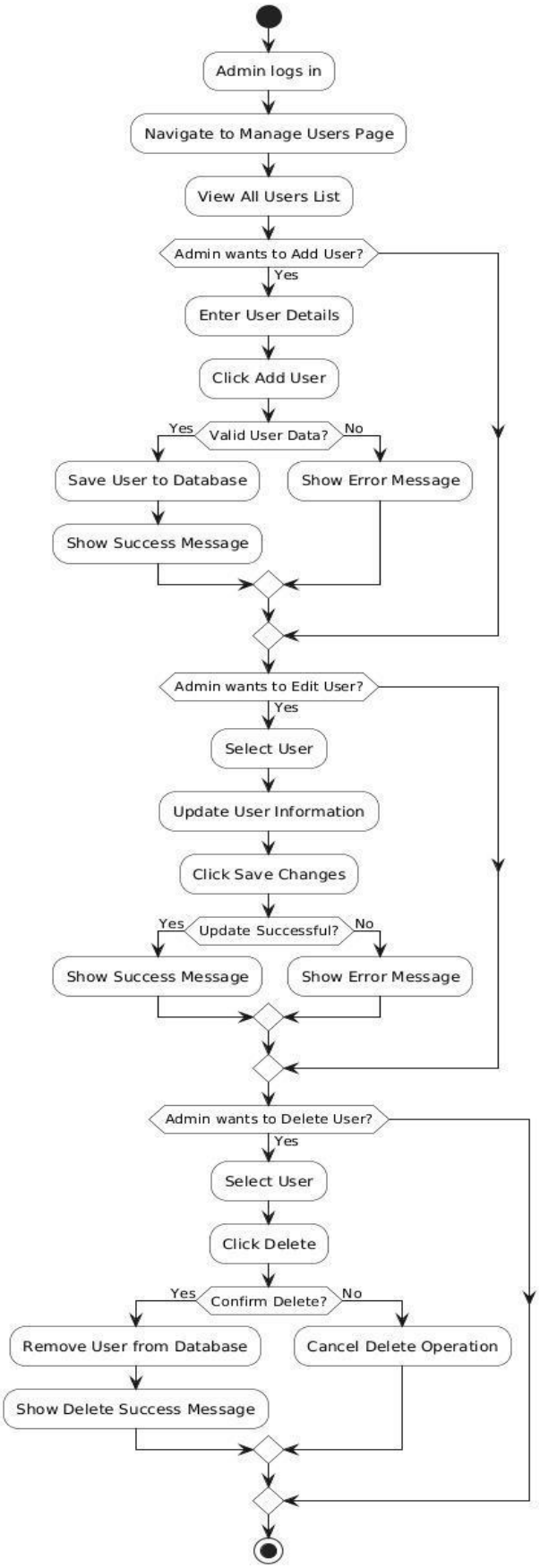

Admin logs in
Navigate to Manage Users Page
View All Users List
Admin wants to Add User?
Yes
Enter User Details
Click Add User
Yes
Valid User Data?
No
Save User to Database
Show Error Message
Show Success Message
Admin wants to Edit User?
Yes
Select User
Update User Information
Click Save Changes
Yes
Update Successful?
No
Show Success Message
Show Error Message
Admin wants to Delete User?
Yes
Select User
Click Delete
Yes
Confirm Delete?
No
Remove User from Database
Cancel Delete Operation
Show Delete Success Message

Figure 10: Manage Users

Figure 11: Confirm Delivery

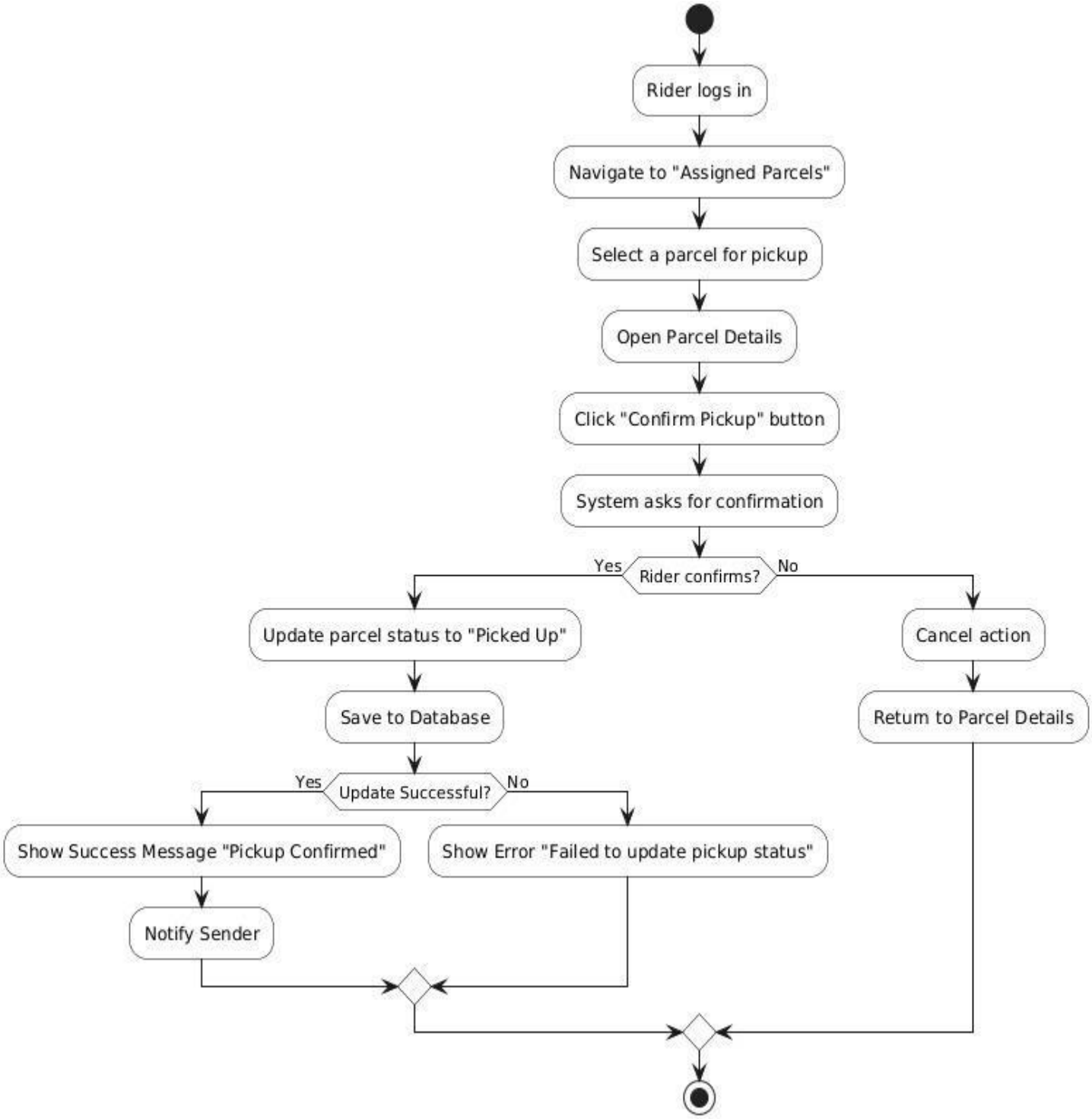

Rider logs in
Navigate to "Assigned Parcels"
Select a parcel for pickup
Open Parcel Details
Click "Confirm Pickup" button
System asks for confirmation
Rider confirms?
Yes
No
Update parcel status to "Picked Up"
Cancel action
Save to Database
Return to Parcel Details
Update Successful?
Yes
No
Show Success Message "Pickup Confirmed"
Show Error "Failed to update pickup status"
Notify Sender

Figure 12: Confirm Pickup


Rider logs in
Navigate to "Assigned Parcels" page
Request list of assigned parcels from Database
Parcels available?
Yes
No
Display list of parcels
Show message "No parcels assigned yet"
Show summary (Parcel ID, Status, Pickup & Delivery addresses)
Rider selects a parcel?
Yes
No
Open Parcel Details
Rider browses list
Show full parcel info
Allow actions (Confirm Pickup / Confirm Delivery)
Database Error?
Yes
Show Error Message
Allow Retry

Figure 13: view Assigned Parcel

2.4.4 Sequence Diagram

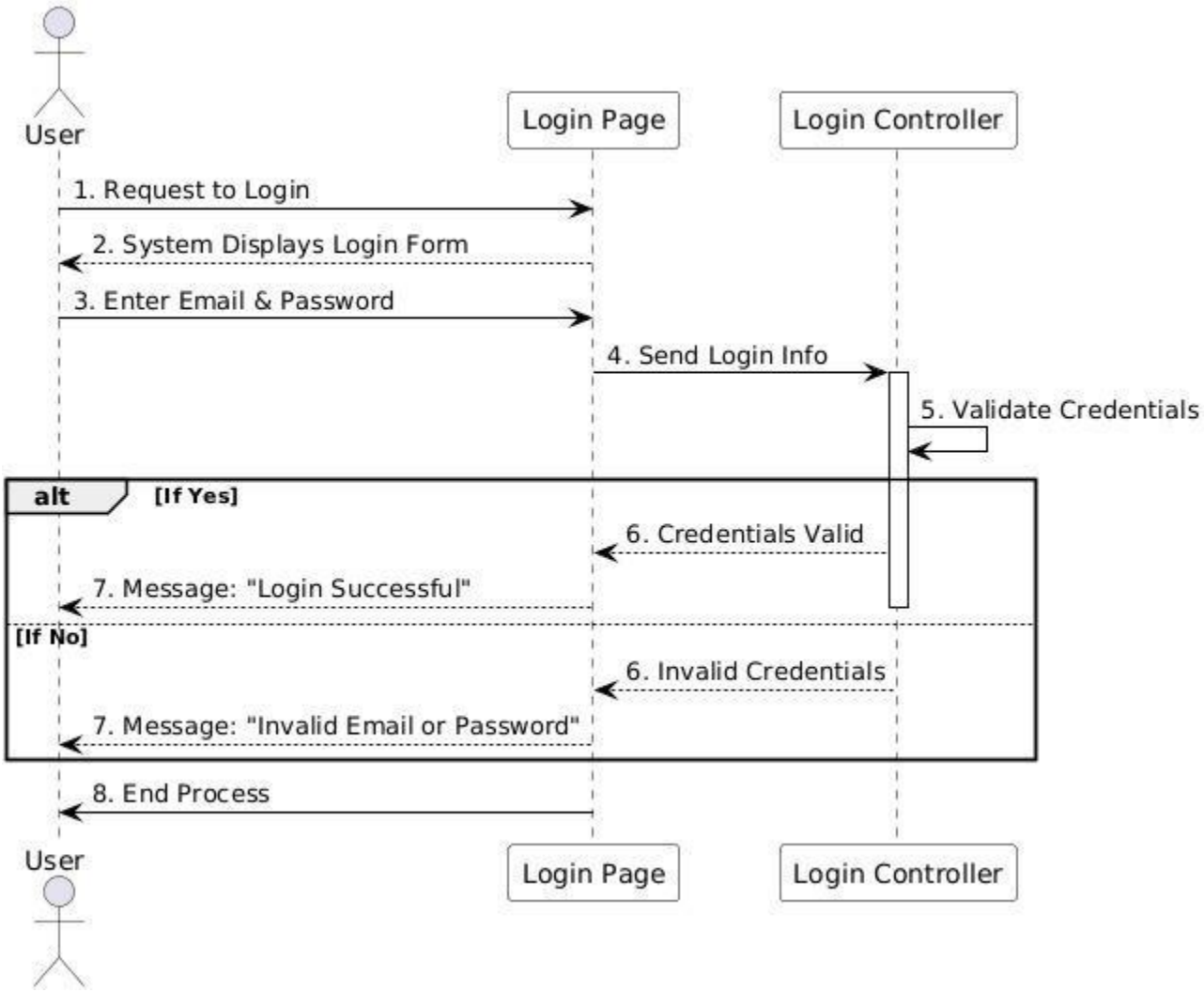


Figure 1: Login pag

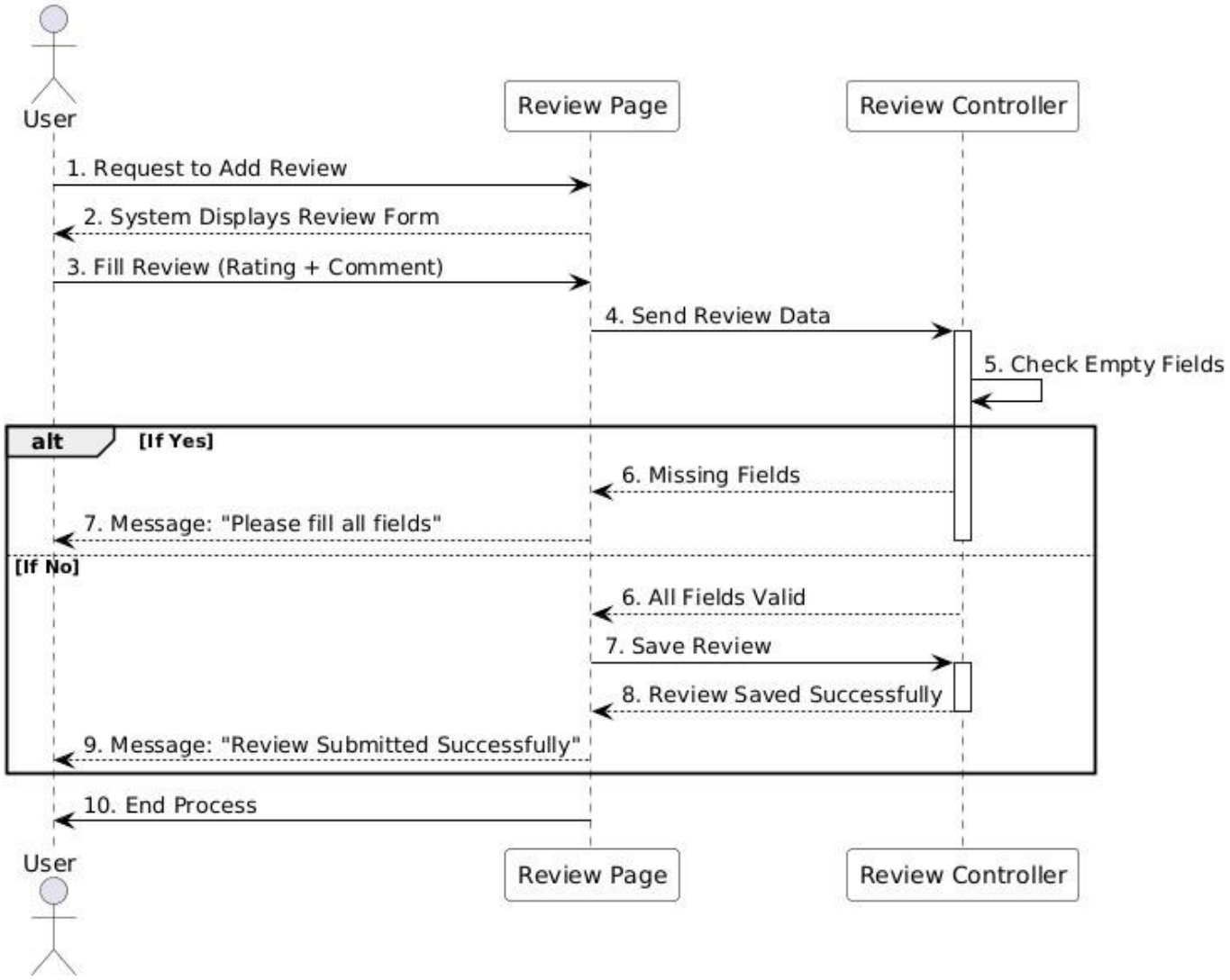


Figure 2: Review page

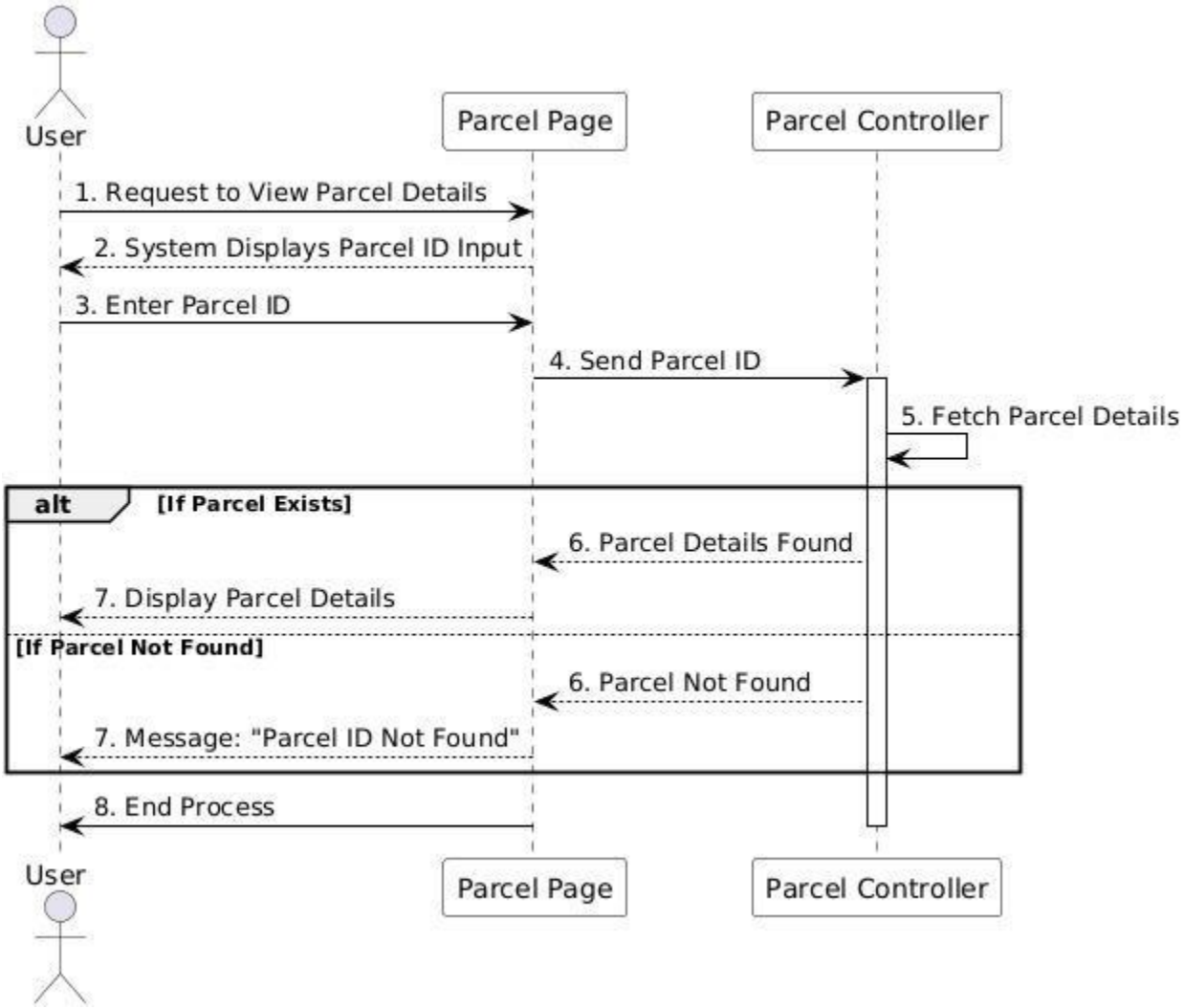


Figure 3: View Parcel Detaile

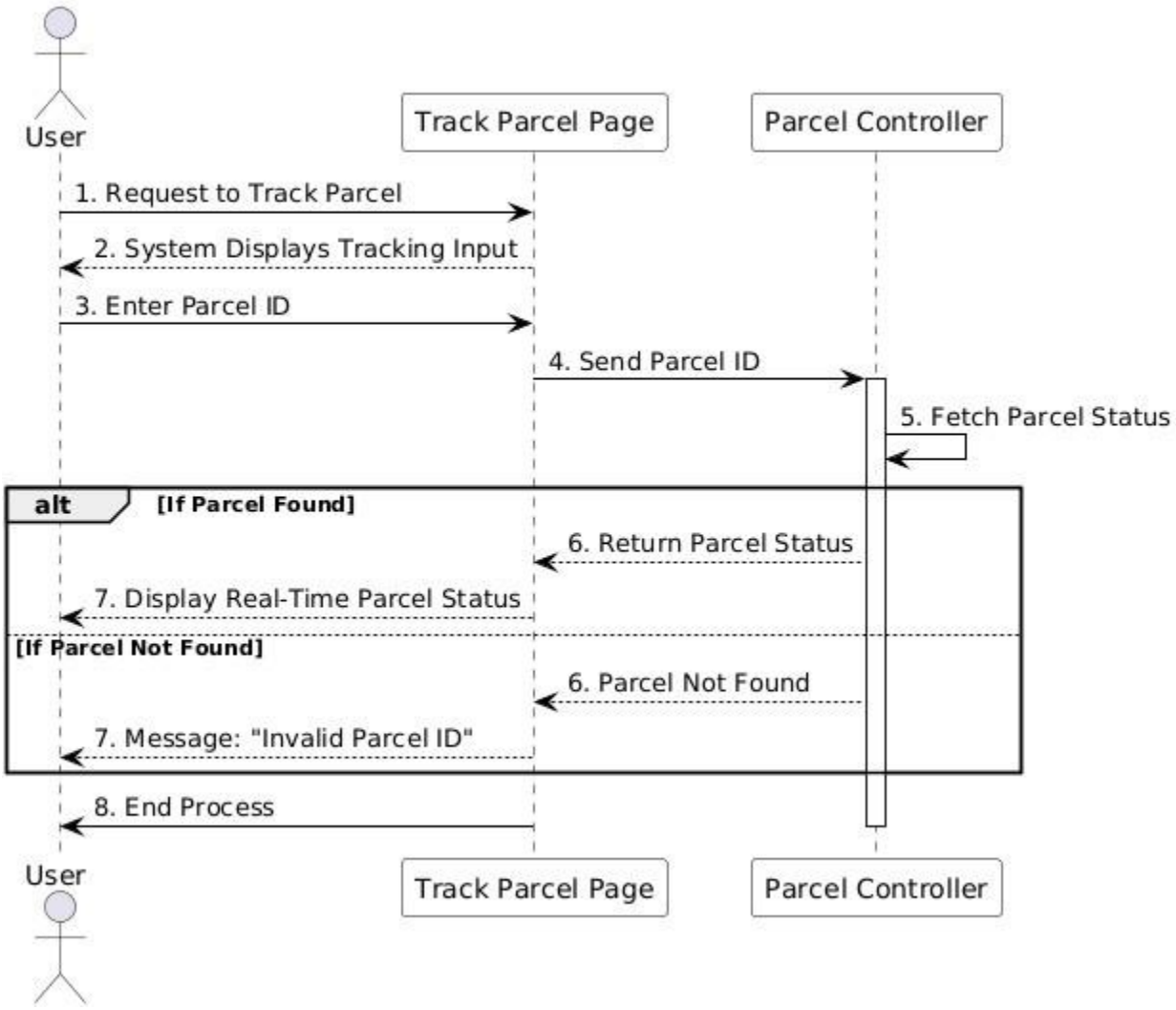


Figure 4: Track Parcel

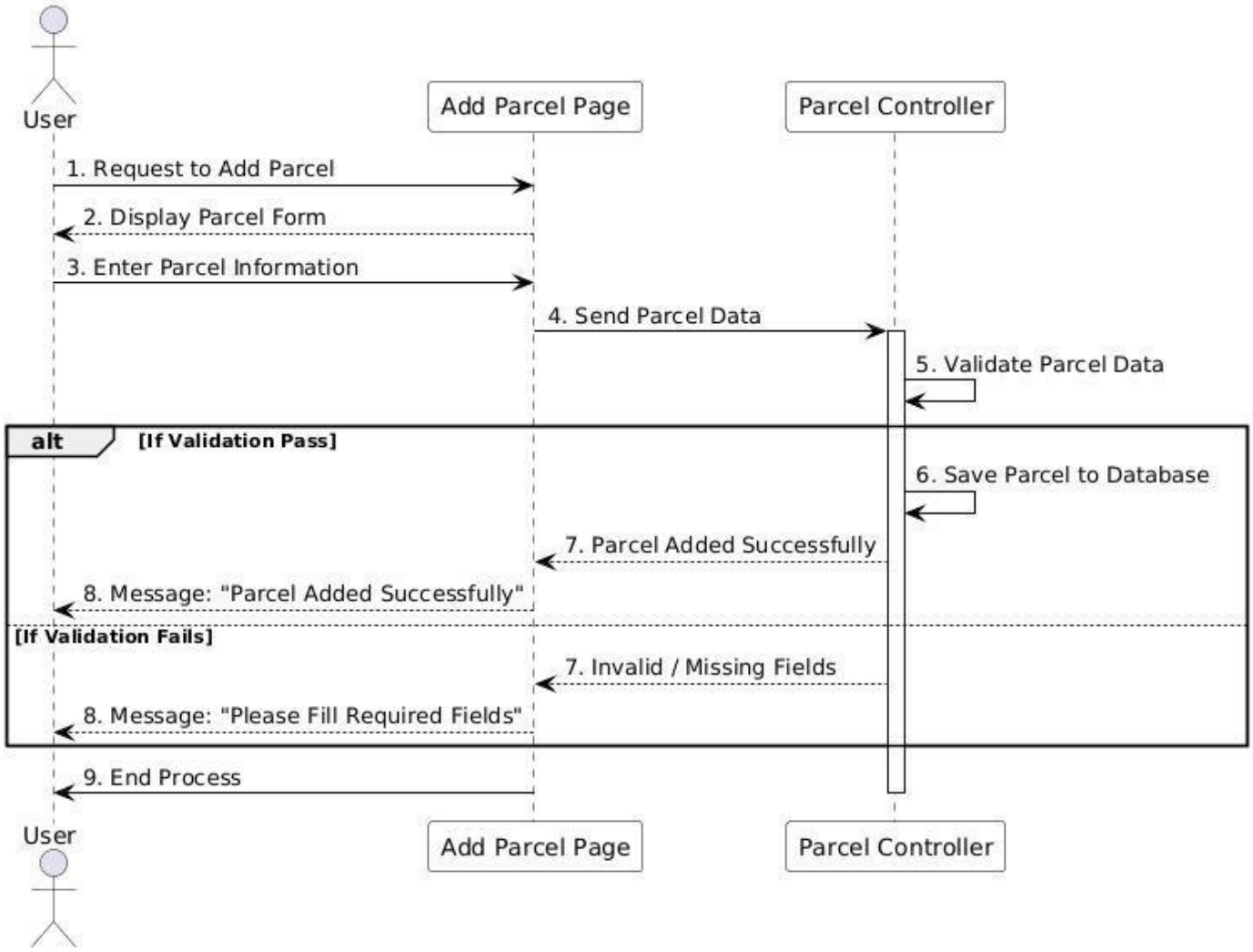


Figure 5: Add Parcel Page

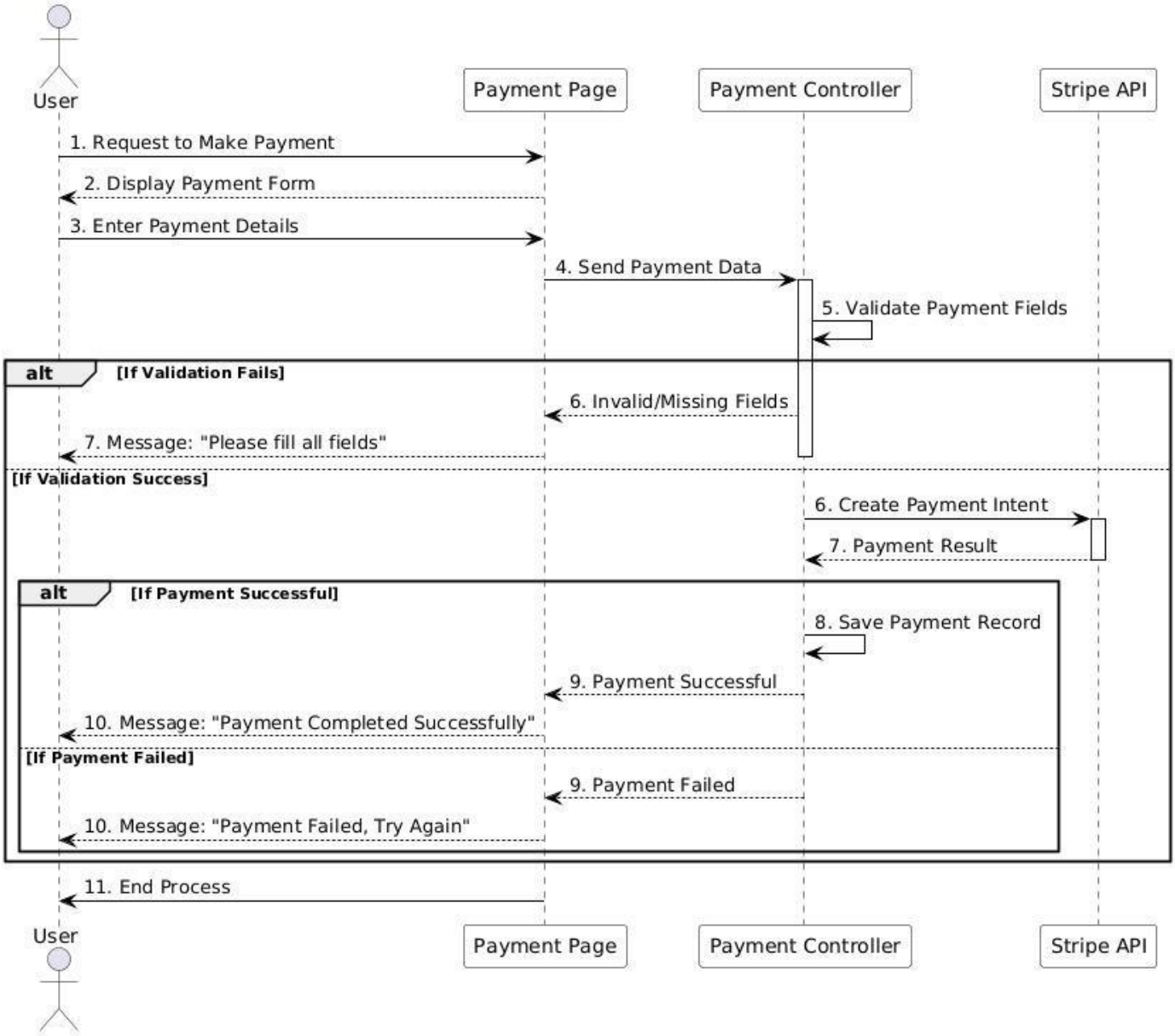


Figure 6: Payment Page

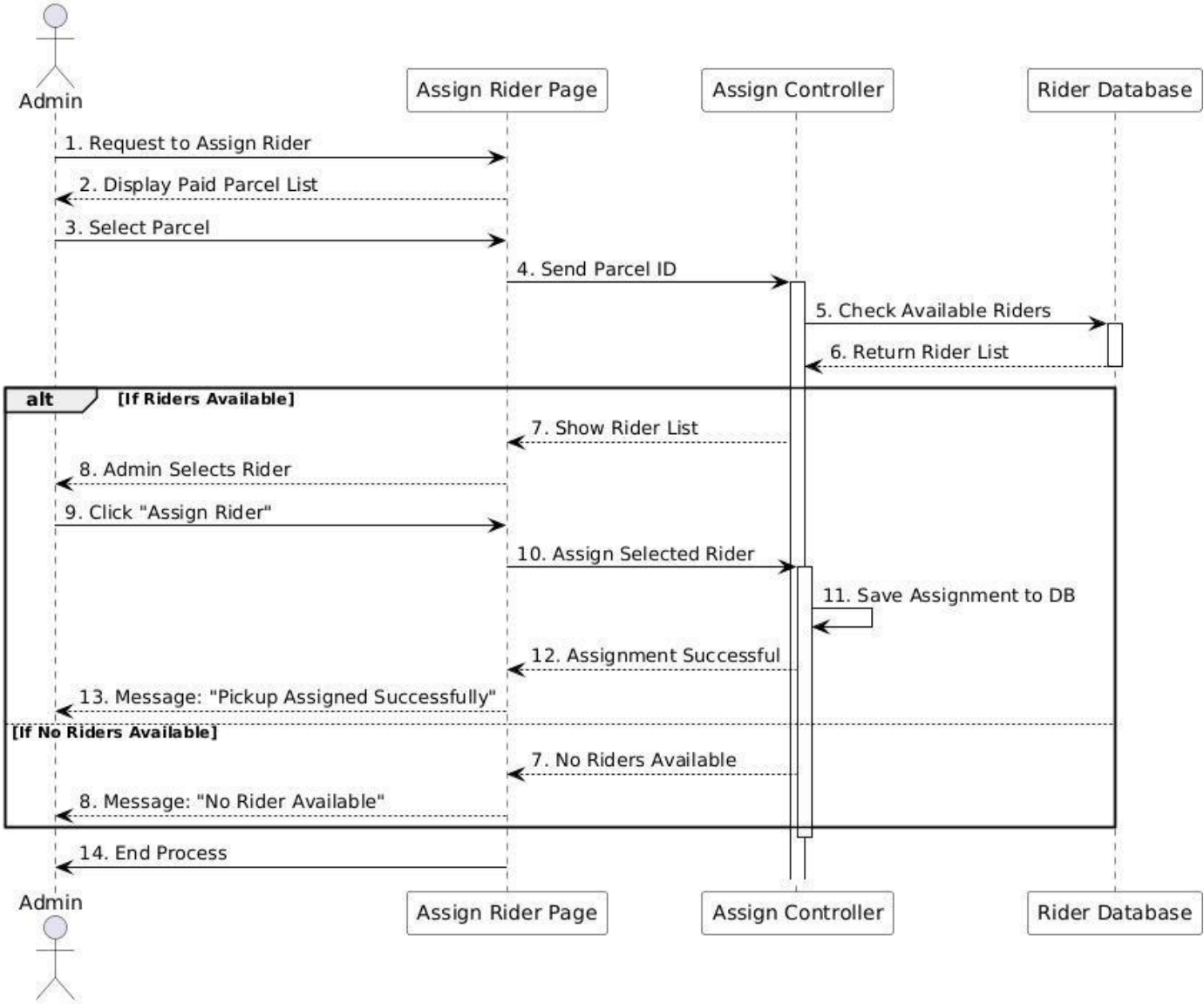


Figure 7: Assign Rider Parcel

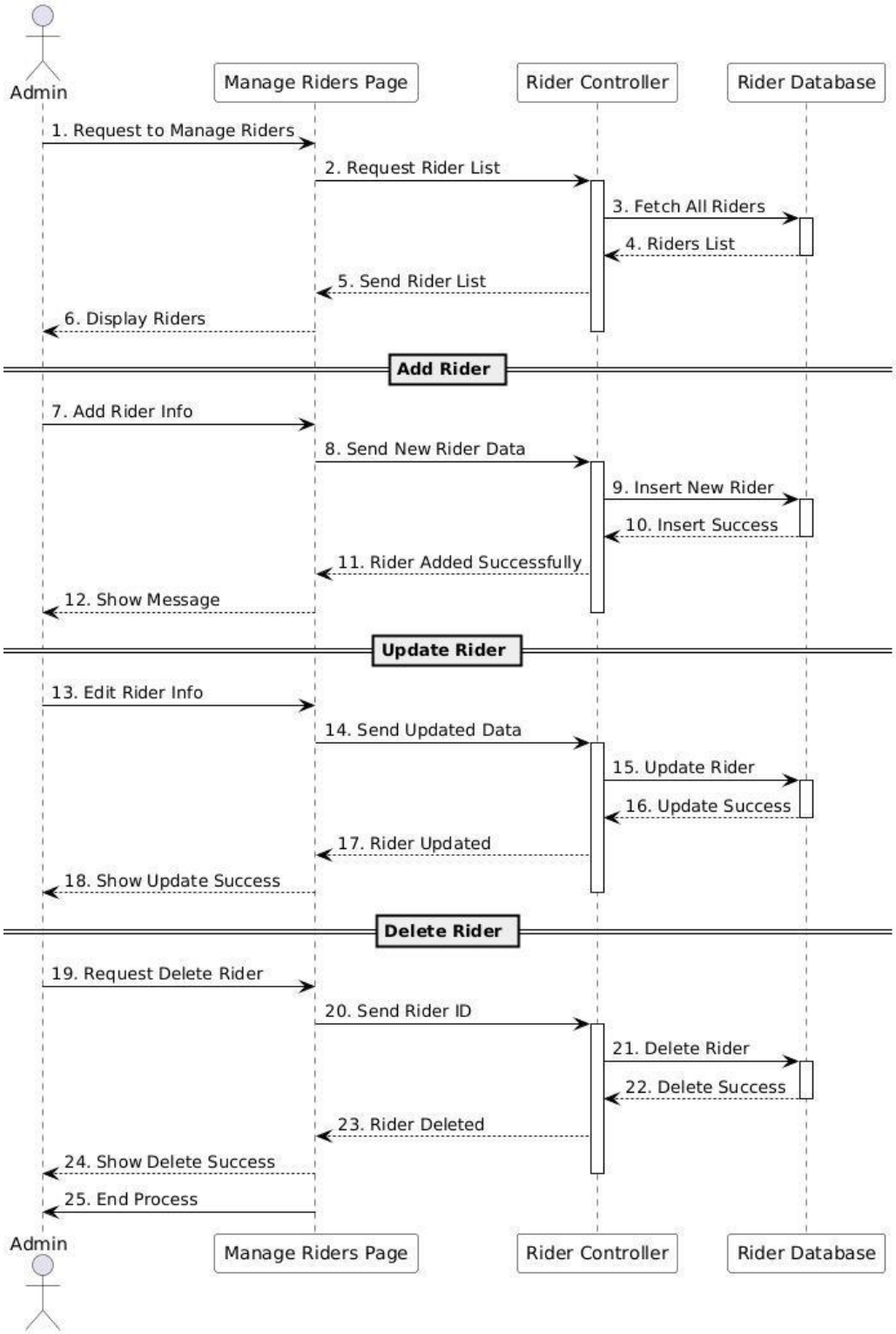


Figure 8: Manage Rider

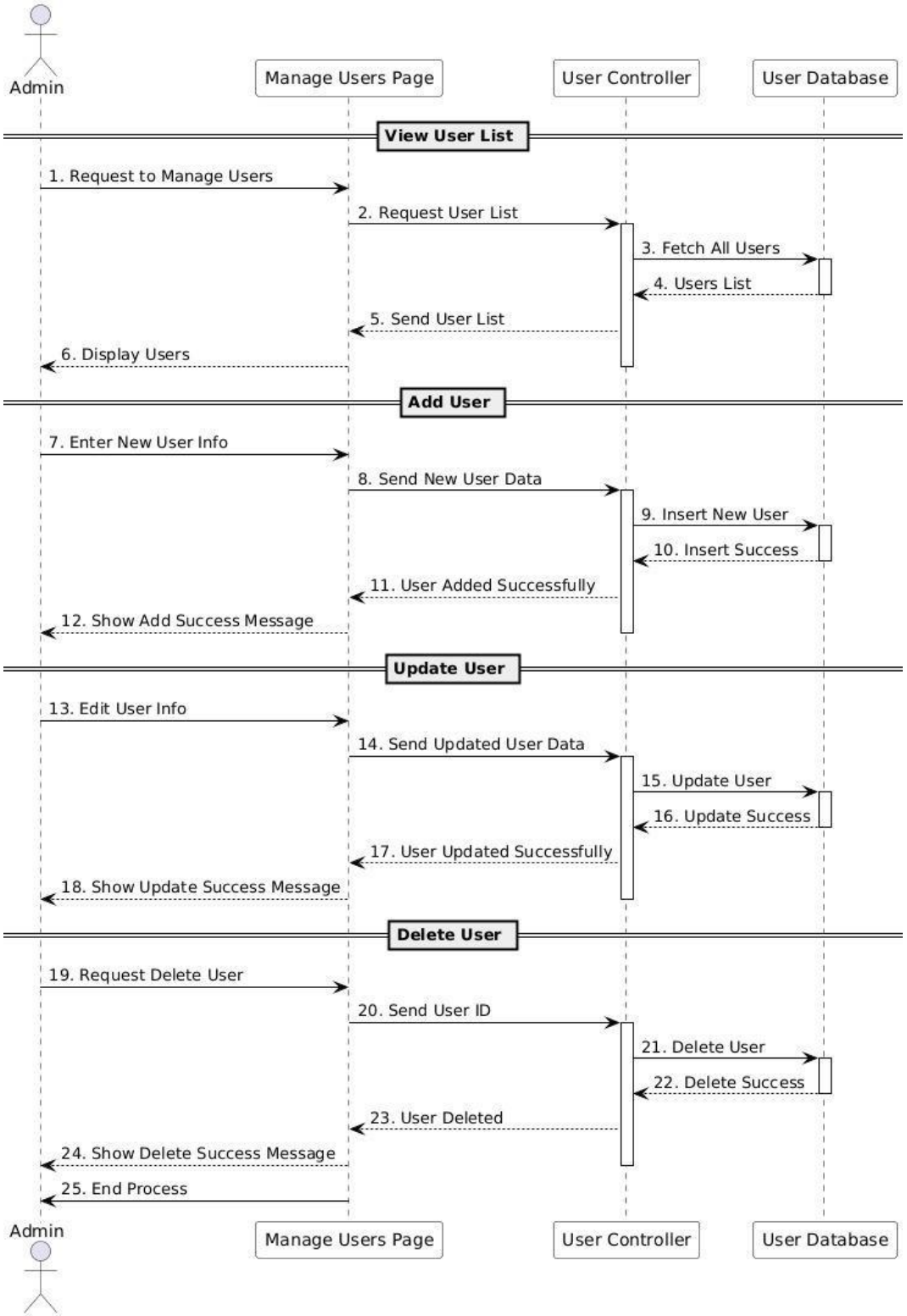


Figure 09: Manage Users

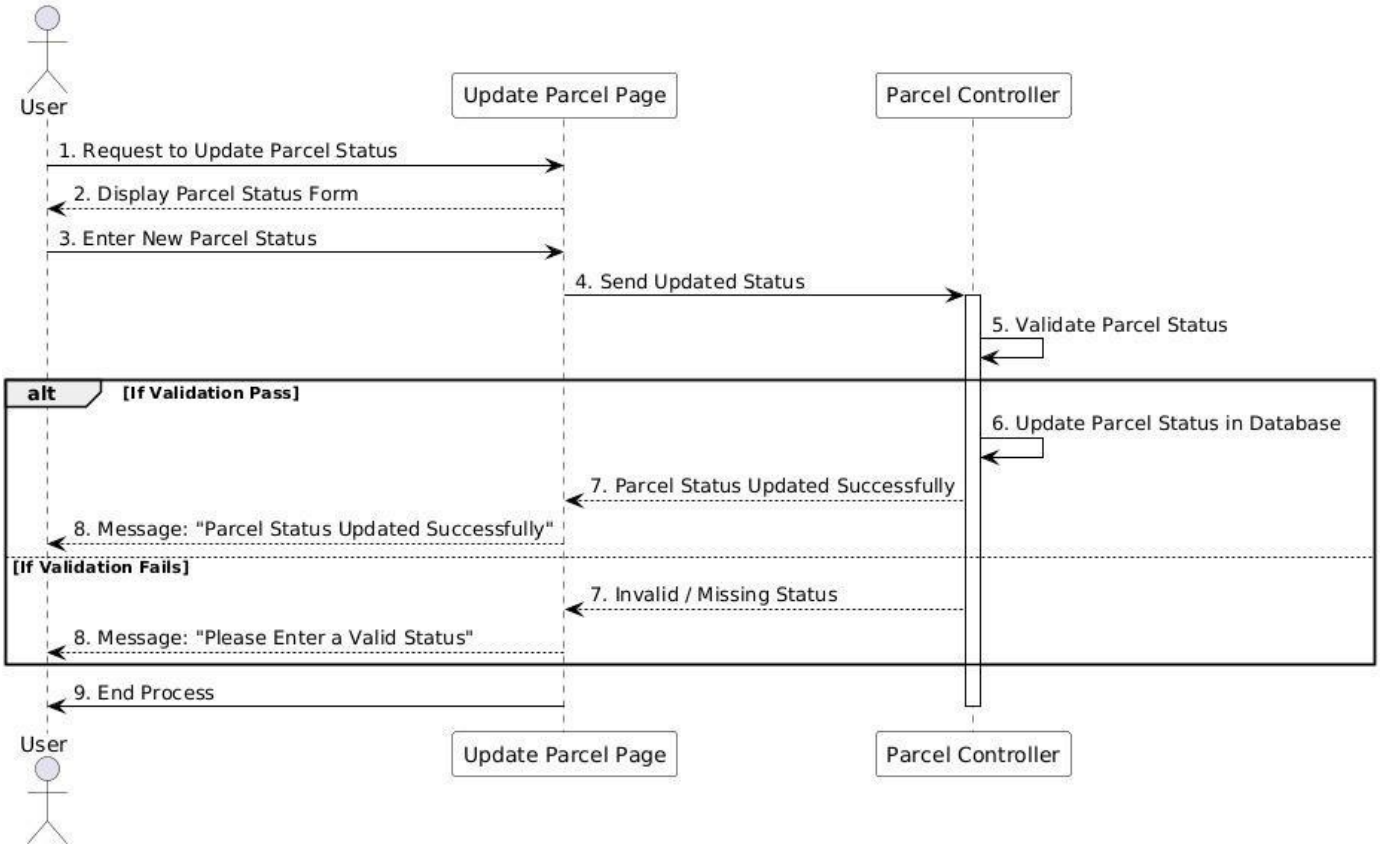


Figure 10: Update Parcel Status

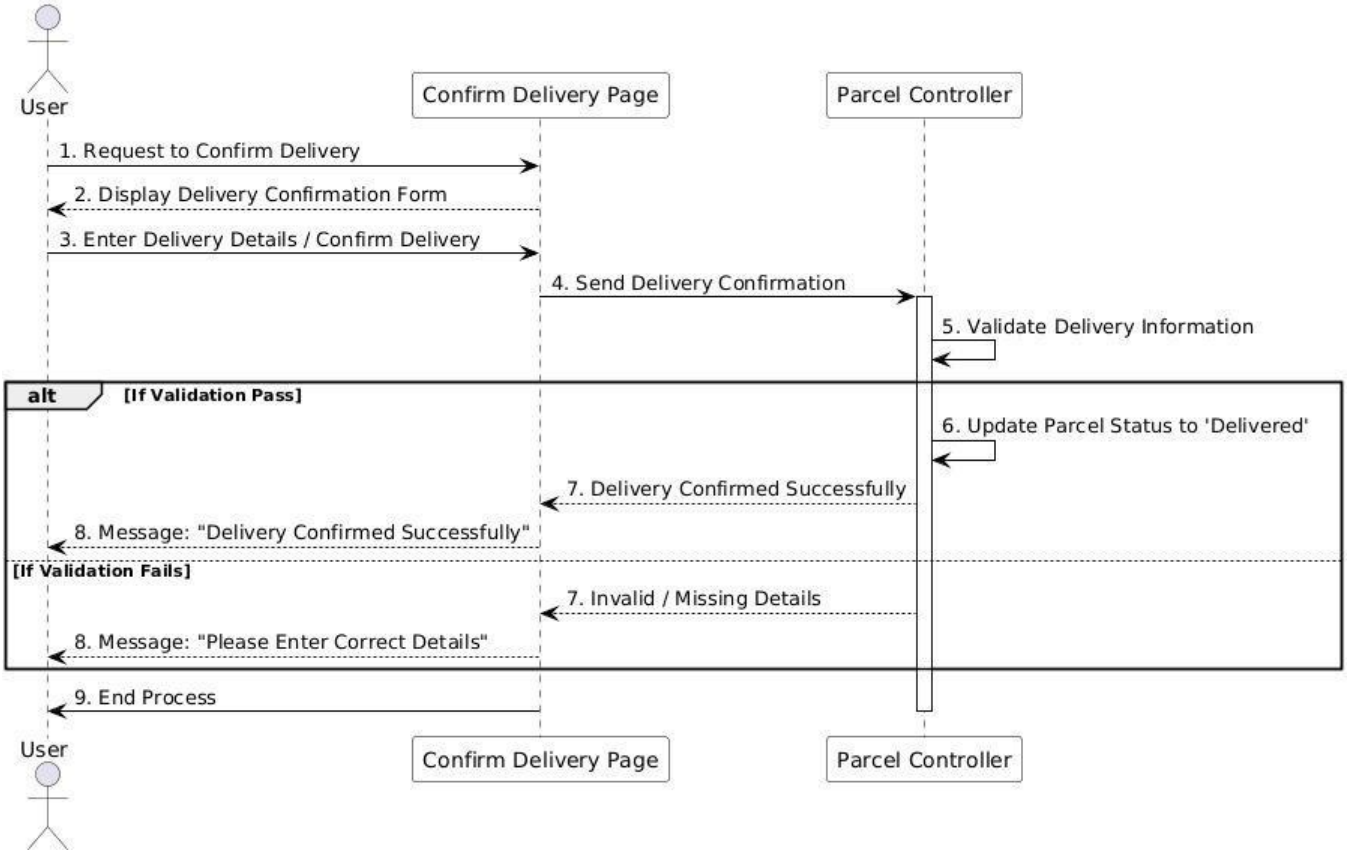


Figure 11: Confirm Delivery

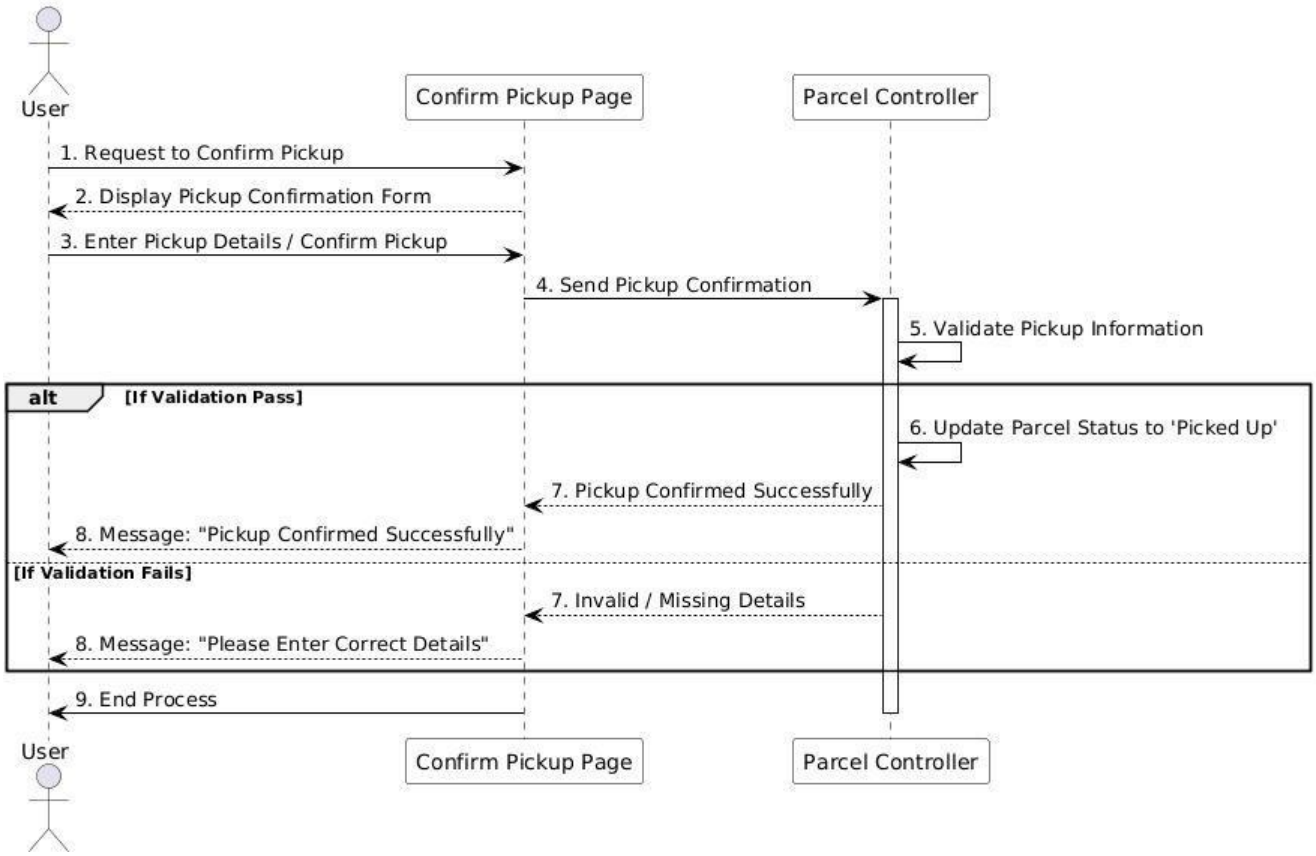


Figure 12: Confirm Pickup

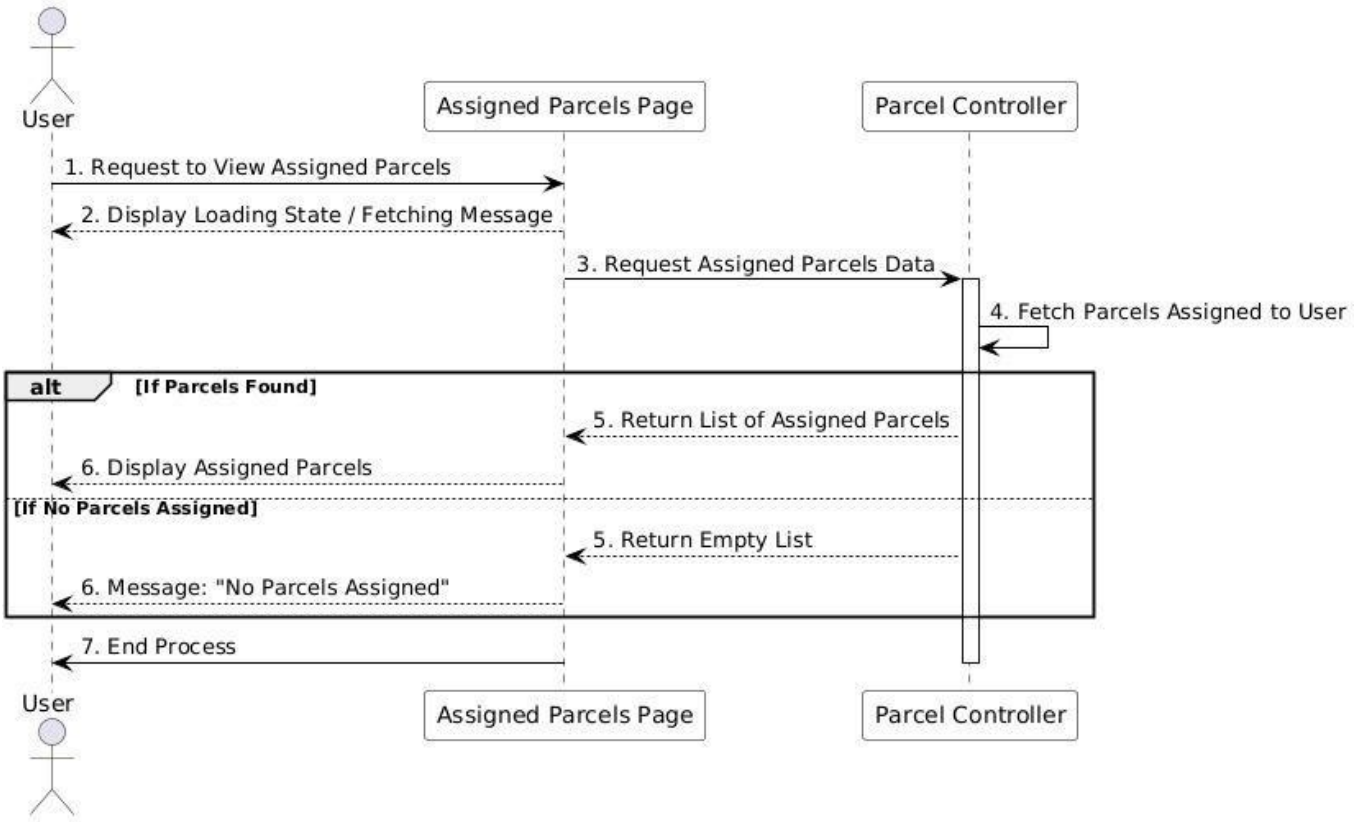


Figure 13: view Assigned Parcel

2.4.5 Class Diagram

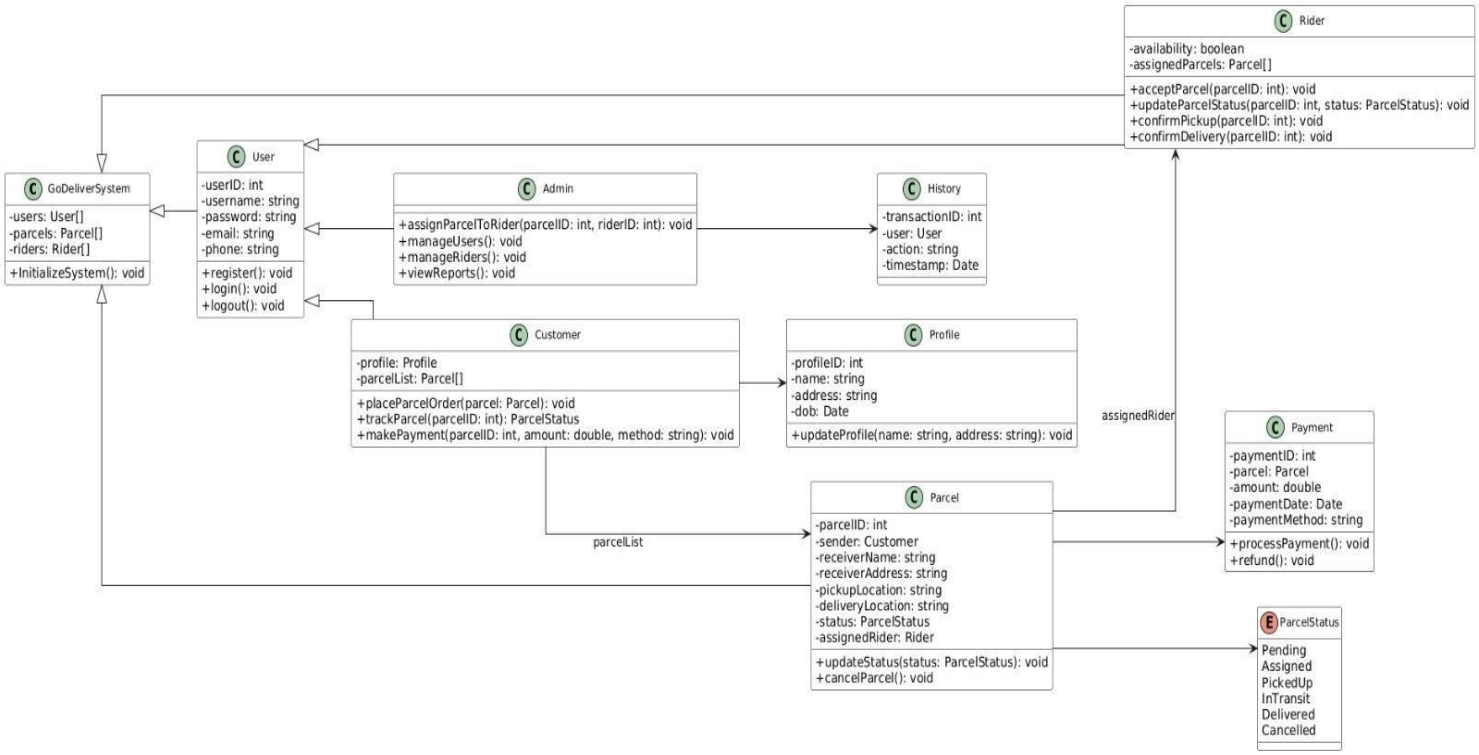


Figure 14: Class Diagram

2.4.6 ER Diagram

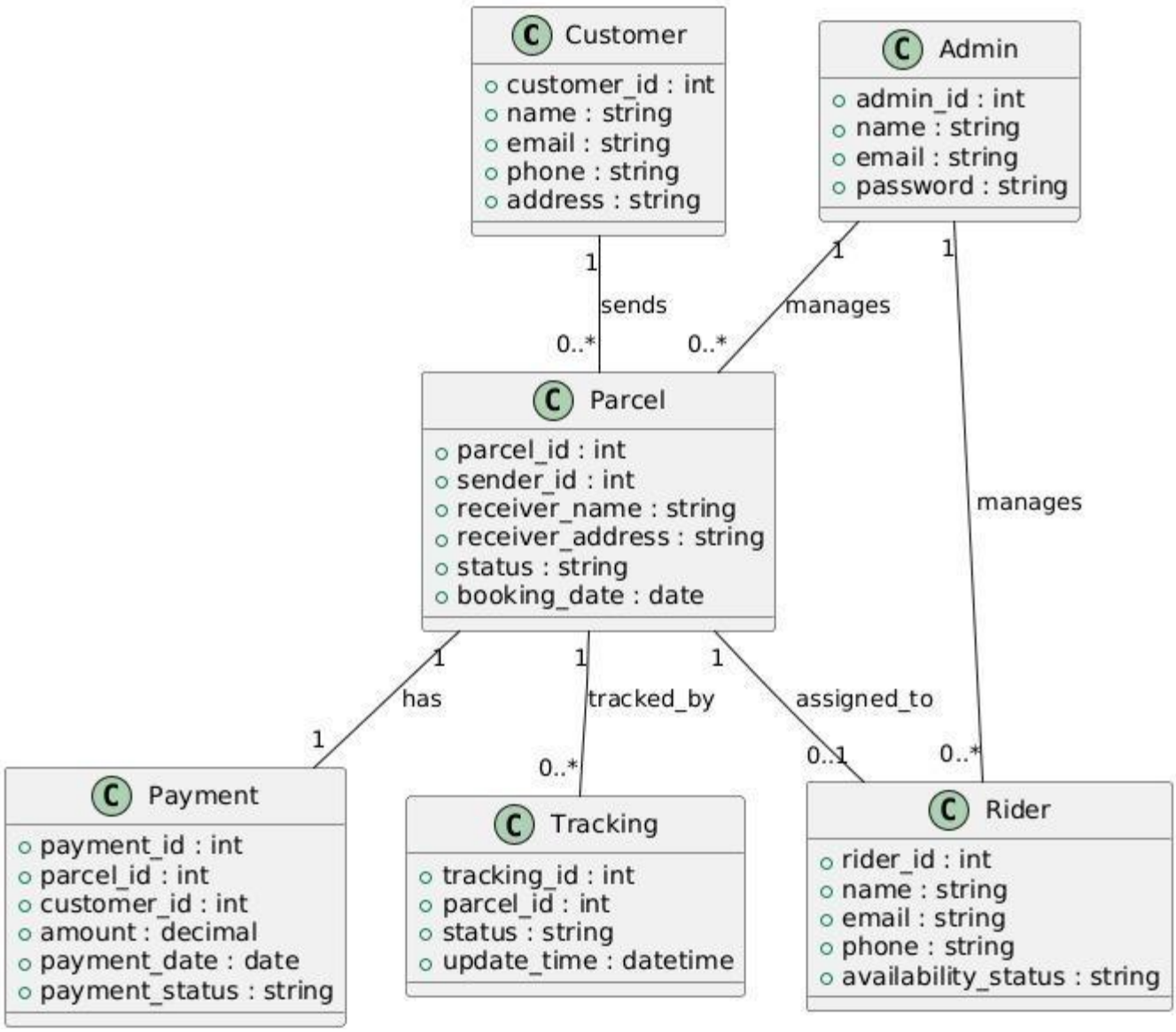


Figure 15: ER Diagram

2.5 Coding: Appendix

```
3

4    import { createBrowserRouter } from "react-router";
5    import RootLayout from "../layout/RootLayout";
6    import Home from "../pages/Home/Home/Home";
7    import Login from "../pages/Authentication/Login/Login";
8    import Register from "../pages/Authentication/Register/Register";
9    import Coverage from "../pages/Coverage/Coverage";
10   import PrivateRoute from "../routes/PrivateRoute";
11   import SendParcel from "../pages/SendParcel/SendParcel";
12   import AuthLayout from "../layout/AuthLayout/AuthLayout";
13   import DashboardLayout from "../layout/DashboardLayout";
14   import MyParcels from "../pages/DashBoardPages/MyParcels/MyParcels";
15   import Payment from "../pages/DashBoardPages/Payment/Payment";
16   import PaymentHistory from
     "../pages/DashBoardPages/PaymentHistory/PaymentHistory";
17   import BeARider from "../pages/DashBoardPages/BeARider/BeARider";
18   import PendingRiders from "../pages/DashBoardPages/PendingRiders/PendingRiders";
19   import ActiveRiders from "../pages/DashBoardPages/ActiveRiders/ActiveRiders";
20   import MakeAdmin from "../pages/DashBoardPages/MakeAdmin/MakeAdmin";
21   import Forbidden from "../pages/Forbidden/Forbidden";
22   import AdminRoute from "../routes/AdminRoute";
23   import AssignRider from "../pages/DashBoardPages/AssignRider/AssignRider";
24   import Contact from "../pages/Contact/Contact";
25   import DashboardHome from "../pages/DashBoardPages/DashboardHome";
26   import RiderRoute from "../routes/riderRoute";
27   import PendingDeliveries from
     "../pages/DashBoardPages/PendingDeliveries/PendingDeliveries";
28   import CompletedDeliveries from
     "../pages/DashBoardPages/CompletedDeliveries/CompletedDeliveries";
29   import MyEarnings from "../pages/DashBoardPages/MyEarnings/MyEarnings";
30   import TrackPickup from "../pages/DashBoardPages/TrackPickup/TrackPickup";
31   import ParcelTracking from
     "../pages/DashBoardPages/ParcelTracking/ParcelTracking";
32   import BookParcel from "../pages/DashBoardPages/BookParcel/BookParcel";
33   import ParcelDetails from "../pages/DashBoardPages/MyParcels/ParcelDetails";
34   import PickupScan from "../pages/DashBoardPages/PickupScan/PickupScan";
35
36   export const router = createBrowserRouter([
37       {
38           path: "/",
39           Component: RootLayout,
```

```
            children: [
                {
                    index: true,
                    Component: Home,
                },
                {
                    path: "coverage",
                    Component: Coverage,
                    loader: () => fetch("/districtsData.json"),
                },
                {
                    path: "contact",
                    Component: Contact
                },
                {
                    path: "forbidden",
                    Component: Forbidden,
                },
                {
                    path: 'beARider',
                    element: <PrivateRoute><BeARider></BeARider></PrivateRoute>,
                    loader: () => fetch("/districtsData.json"),
                },
                {
                    path: "sendParcel",
                    element: (
                        <PrivateRoute>
                            <SendParcel />
                        </PrivateRoute>
                    ),
                    loader: () => fetch("/districtsData.json"),
                },
            ],
        },
        {
            path: "/",
            Component: AuthLayout,
            children: [
                {
                    path: "login",
                    Component: Login,
                },
                {
                    path: "register",
```

```
                    Component: Register,
                },
            ],
        },

        // rider route
        {
            path: "/dashboard",
            element: <DashboardLayout />,
            children: [
                {
                    path: "pending-deliveries",
                    element: <RiderRoute><PendingDeliveries /></RiderRoute>
                },
                {
                    path: "completed-deliveries",
                    element: <RiderRoute><CompletedDeliveries /></RiderRoute>
                },
                {
                    path: "my-earnings",
                    element: <RiderRoute><MyEarnings /></RiderRoute>
                },
                {
                    path: "pickup/:trackingId",
                    element: <RiderRoute><PickupScan></PickupScan></RiderRoute>
                },
            ]
        },



        // admin route

        {
            path: '/dashBoard',
            element: <PrivateRoute>
                <DashboardLayout></DashboardLayout>
            </PrivateRoute>,
            children: [
                {
                    index: true,
                    Component: DashboardHome
                },
                {
```

```
126                 path: "myParcels",
127                 Component: MyParcels,
128             },
129             {
130                 path: "parcels/:id",
131                 element: <ParcelDetails></ParcelDetails>
132             },
133             {
134                 path: "payment/:parcelId",
135                 Component: Payment,
136             },
137             {
138                 path: "paymentHistory",
139                 Component: PaymentHistory,
140             },
141             {
142                 path: "track-pickup",
143                 Component: TrackPickup,
144             },
145             {
146                 path: "parcel-tracking",
147                 Component: ParcelTracking,
148             },
149             {
150                 path: 'assignRider',
151                 element: <AdminRoute><AssignRider></AssignRider></AdminRoute>
152             },
153             {
154                 path: "pendingRiders",
155                 element: <AdminRoute><PendingRiders></PendingRiders></AdminRoute>
156             },
157             {
158                 path: "activeRiders",
159                 element: <AdminRoute><ActiveRiders></ActiveRiders></AdminRoute>
160             },
161             {
162                 path: "bookParcel",
163                 element: <AdminRoute><BookParcel></BookParcel></AdminRoute>
164             },
165             {
166                 path: "makeAdmin",
167                 element: <AdminRoute><MakeAdmin></MakeAdmin></AdminRoute>
168             },
169         ]
```

```
170       }
171   ]);
172
```

```
import { Link, NavLink } from 'react-router';
import useAuth from '../../../hooks/useAuth';
import { useTranslation } from "react-i18next";

const Navbar = () => {
    const { user, logOut } = useAuth();
    const { t, i18n } = useTranslation();

    const handleSignOut = () => {
        logOut()
            .then(() => console.log("sign out successfully"))
            .catch(error => console.log(error));
    };

    // Change Language
    const changeLanguage = (lang) => {
        i18n.changeLanguage(lang);
    };

    const navItems = (
        <>
            <li><NavLink to="/">{t("home")}</NavLink></li>
            <li><NavLink to="/sendParcel">{t("sendParcel")}</NavLink></li>
            <li><NavLink to="/coverage">{t("coverage")}</NavLink></li>

            {user && (
                <li><NavLink to="/dashBoard">{t("dashboard")}</NavLink></li>
            )}

            <li><NavLink to="/beARider">{t("beARider")}</NavLink></li>
            <li><NavLink to="/contact">{t("contact")}</NavLink></li>
        </>
    );

    return (
        <div className="navbar text-white bg-[#103963] shadow-sm">
            <div className="navbar-start">
                <div className="dropdown text-black">
```

```
                    <div tabIndex={0} role="button" className="btn btn-ghost lg:hidden">
                        <svg xmlns="http://www.w3.org/2000/svg" className="h-5 w-5" fill="none"
viewBox="0 0 24 24" stroke="currentColor">
                            <path strokeLinecap="round" strokeLinejoin="round" strokeWidth="2"
d="M4 6h16M4 12h8m-8 6h16" />
                        </svg>
                    </div>
                    <ul tabIndex="-1" className="menu menu-sm dropdown-content bg-base-100
rounded-box z-1 mt-3 w-52 p-2 shadow">
                        {navItems}
                    </ul>
                </div>

                <div className="flex">
                    <img className="w-[60px] h-[60px]" src="https://cdn-icons-
png.flaticon.com/512/10053/10053703.png" alt="" />
                    <a className="text-3xl font-extrabold mt-3">GoDeliver</a>
                </div>
            </div>

            <div className="navbar-center hidden lg:flex">
                <ul className="menu menu-horizontal px-1">{navItems}</ul>
            </div>

            <div className="navbar-end flex gap-2">
                {/* Language Switch */}
                {/* <button onClick={() => changeLanguage("en")} className="btn btn-
sm">EN</button>
                <button onClick={() => changeLanguage("bn")} className="btn btn-sm">BN</button>
*/}

                {/* Language Switch Toggle */}
                <label className="flex items-center cursor-pointer gap-2">
                    <span className="text-sm font-semibold">EN</span>
                    <input
                        type="checkbox"
                        className="toggle toggle-info text-gray-400"
                        onChange={(e) =>
                            e.target.checked ? changeLanguage("bn") : changeLanguage("en")
                        }
                        checked={i18n.language === "bn"}
                    />

                    <span className="text-sm font-semibold">BN</span>
```

```
                </label>

                {/* Login / Logout */}
                {user ? (
                    <button onClick={handleSignOut} className="btn btn-
sm">{t("signOut")}</button>
                ) : (
                    <Link to="/login" className="btn btn-sm">{t("login")}</Link>
                )}
            </div>
        </div>
    );
};

export default Navbar;
```

Chapter 3

3.1 Introduction

Software testing is a very important step when building any system. It helps to check whether the app works properly and meets the needs of the users. In the case of GoDeliver, which is a parcel delivery system, all the components are tested to ensure that they work well, run fast, and remain reliable.

We have three types of users in our system. Customers, admins and riders. Each group has their own unique tasks. Users can order parcels using our website, admins can manage all the ordered parcels and assign to corresponding riders, and riders responsibility is to deliver the parcels to the allocated location.

This section explains the various testing methods used in this system. How each part of the website was tested, different types of tests that were performed and how this helps to ensure a reliable and easier to use website for all types of users.

3.2 Testing Features

The testing features that are used in this system are mentioned in this section. By doing proper testing we ensure a reliable and high performance website for the users. This system strives for smooth, safe and accurate delivery experience for all types of users.

The testing mainly checks whether all features work independently and then it checks whether they work together. We did both unit and integration testing. We first test all the single functions and analyze their response with the expected result, then we test various features all together. The final test is a complete user from, starts from signing up to order a parcel online.

3.1.1 Features to Be Tested

The features of the GoDeliver system that need to be tested are listed below:

1. **User Registration and Login**
    - Checks whether all users (customers, riders, admins) are able to sign up without any issues.
    - Tests the login system using both correct and incorrect login details.
2. **Customer Functions**
    - Sending parcel orders with correct sender and recipient information.
    - Tracking parcels to see if status updates are showing correctly.
    - Making payments and receiving receipts after payment..
3. **Rider Functions**

- Accepting parcel and confirming that the pickup has been completed.
- Changing the status of the parcel at each stage (PickedUp → InTransit → Delivered). Checking the assigned parcels and updating the rider's availability.

4. **Admin Functions**
    - Deliver parcels to the correct customers without any errors.
    - Manage users and passengers, such as adding new parcels, editing details, or removing them.

3.2 Testing Strategies

Testing methods are crucial to ensure an error and bug free website which will lead to good user expeience. Another goal is to ensure all the features work smoothly in all types of cases. We use a mix of unit and integration testing and at the end use full system testing to ensure the bug free process.

3.2.1 Test Approach

The GoDeliver project uses a step-by-step testing approach to test each feature and part of the system. The approach includes:

1. **Unit Testing**
    - Each part, such as user login, parcel handling, and payment processing, is tested separately to ensure that it is working properly.

- Unit testing finds problems in one part, before they are connected to other parts.

2. **Integration Testing**
   - After testing each component separately, integration testing is done to ensure that all components work well together.
   - For example, when a rider updates the status of a parcel, it should be displayed correctly for both the customer and admin parts.
3. **System Testing**
   - The entire GoDeliver system is tested together to ensure that everything works from start to finish.
   - This includes testing all user types, steps, and features in real-life scenarios.
4. **User Acceptance Testing (UAT)**
   - Selected users use it to test whether the system works properly and is easy to use.
   - Feedback from these users makes the system better and easier to use before release.

3.2.2 Pass/Fail Criteria

To clearly determine whether each test case is successful, the GoDeliver system uses the following pass/fail rules:

- **Pass Criteria**
  - The system works as expected without any errors.
  - All inputs are handled correctly and outputs are output correctly.
  - User screens respond correctly to actions and data is stored correctly.
  - Security features such as login and access control work correctly.
- **Fail Criteria**
  - Any part of the system gives incorrect results or stops working.
  - User input is not properly checked, resulting in incorrect or incorrect data.
  - Parcel status, payments, or reports are not updated properly.
  - People can access the system without permission or breach security.

3.3 System Testing (Test Cases with Report)

System testing is the stage where the entire GoDeliver system is tested together to ensure that all parts and features are working properly with each other. The main goal of system testing is to check whether the system meets the functionality, ease of use, and security requirements set during the project.

3.3.1 Sample Test Cases

Test Case ID	Module	Test Description	Test Steps	Expected Result	Actual Result	Pass/Fail
TC01	Customer	Register new customer	Enter valid details and submit registration form	Account created successfully, confirmation message displayed	As expected	Pass
TC02	Customer	Place a parcel order	Login → Enter sender & receiver details → Submit order	Parcel order saved, order ID generated	As expected	Pass
TC03	Customer	Track parcel	Login → Enter parcel ID → View status	Current parcel status displayed accurately	As expected	Pass
TC04	Customer	Make payment	Select parcel → Enter payment details → Submit	Payment successful	As expected	Pass
TC05	Rider	Accept assigned parcel	Login → View assigned parcels → Accept parcel	Parcel status changes to "Assigned" for Rider	As expected	Pass
TC06	Rider	Update parcel status	Login → Select parcel → Update status to "PickedUp" / "InTransit"	Parcel status updated properly	As expected	Pass
TC07	Rider	Confirm delivery	Login → Select delivered parcel → Confirm	Parcel status changes to "Delivered" for Customer & Admin	As expected	Pass
TC08	Admin	Assign parcel to rider	Login → Select parcel → Assign to rider	Rider assigned successfully, notification sent	As expected	Pass
TC09	Admin	Generate report	Login → Generate report for parcel delivery	Report generated correctly, displaying all relevant details	As expected	Pass
TC10	System Security	Login with invalid credentials	Enter wrong username/password	Error message displayed, login denied	As expected	Pass
TC11	System	Cancel parcel	Customer selects pending parcel → Cancel	Parcel status updated to "Cancelled"	As expected	Pass

3.3.2 System Testing Report Summary

- **Total Test Cases Executed:** 11
- **Passed:** 11
- **Failed:** 0
- **Remarks:** All major features of the GoDeliver system have been tested and are working well, including user registration, parcel handling, payments, rider tasks, and admin functions. The system works as expected, is easy to use, and no major issues were found.

3.4 Summary

This chapter explains in detail the software testing process for the GoDeliver system. This testing verifies that all parts of the system - such as customer operations, rider tasks, admin features, parcel handling, and payment processing - are working correctly and reliably.

We started by introducing software testing and explaining why it is important to improve the system by finding defects. We identified the main features to be tested, which included all users and parts of the system. A clear test plan was used, which included unit testing, integration testing, system testing and user acceptance testing, with clear pass or fail rules to check whether the tests were successful.

The system testing phase had detailed test cases for each part, to ensure that each function was tested as expected. The test report showed that all major features worked as they should, the system handled incorrect inputs correctly, and no major issues were found.

Overall, the tests show that GoDeliver is a reliable, secure, and easy-to-use parcel delivery system, ready to use. Careful testing ensures that users can confidently perform their tasks - such as placing orders, delivering parcels, or managing the system as an admin - without any errors.

Chapter 4 Deployment and Maintenance

4.1 Introduction

This section explains how we deployed our projects into the cloud, which platform we used and the steps to run the system on the cloud server, how users can access it and how we make it work effectively in the cloud.

4.2 The SRLC (software release life cycle)

Deployment Process:

The deployment process was handled following the software release life cycle, ensuring smooth release and proper maintenance of our system.

The first test was done locally to prevent any disaster in live, fixing major bugs and ensuring that the main features works without any errors.

Before going live we first deployed the project in firebase, tested the website live for a couple of days with various test users, to check for any errors.

After full testing, the system was deployed to firebase production for the users to use. Necessary settings such as database integration and security were implemented.

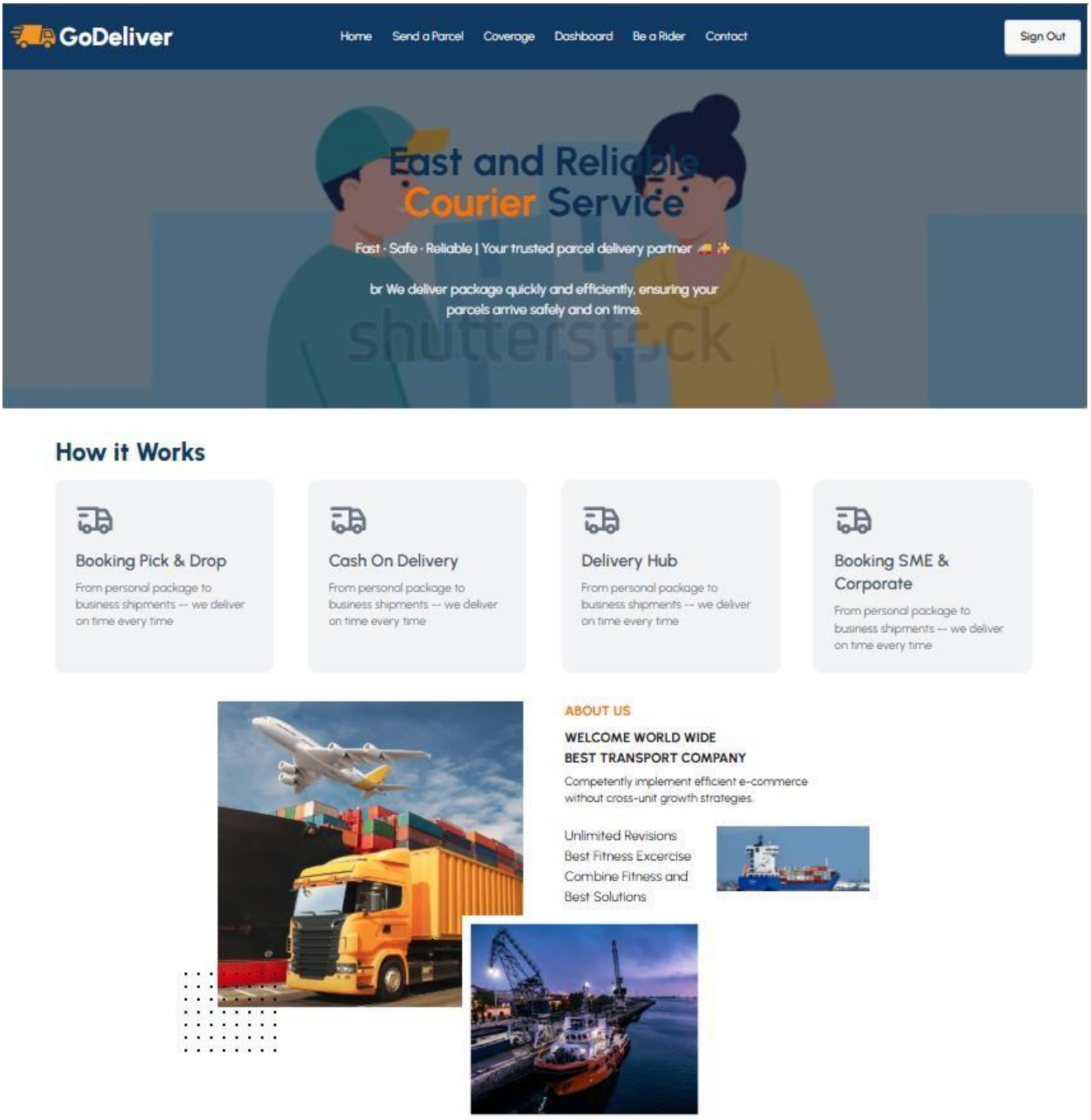
After the final deployment, monitoring of the system has started to check if any error occurs while users are using it. If any issue occurs the issues are quickly resolved and then redeployed following all the previous steps.

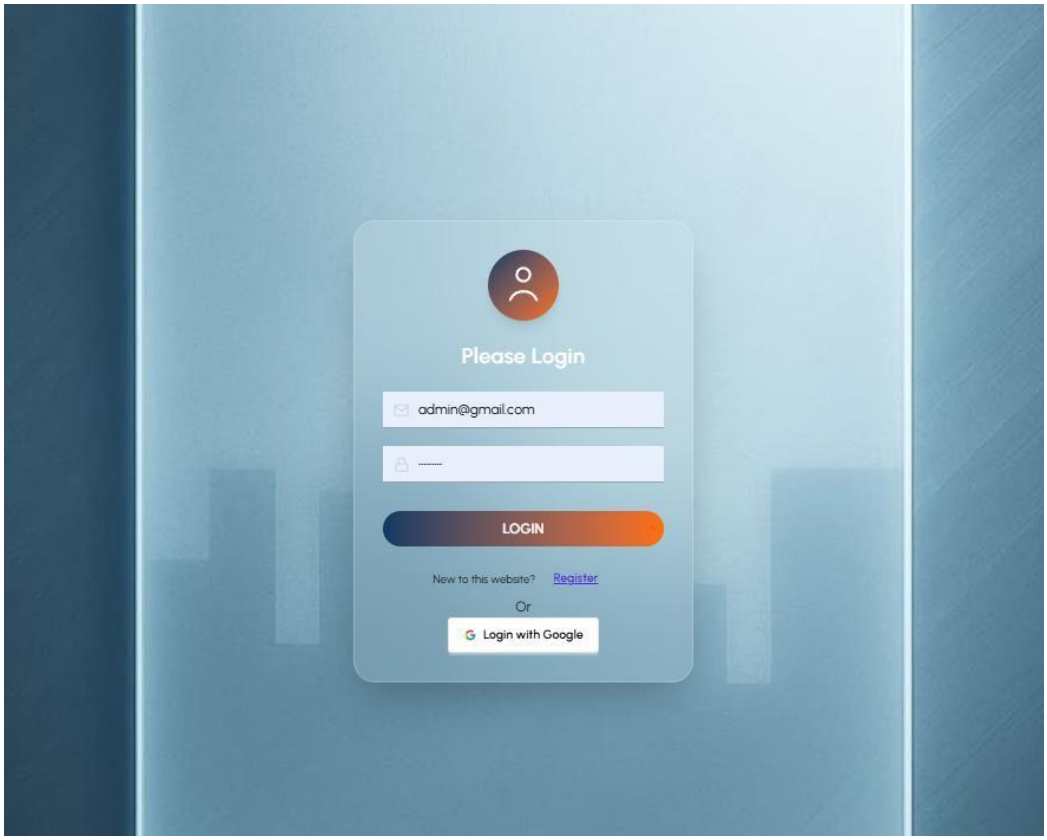
CHAPTER 5: USER MANUAL

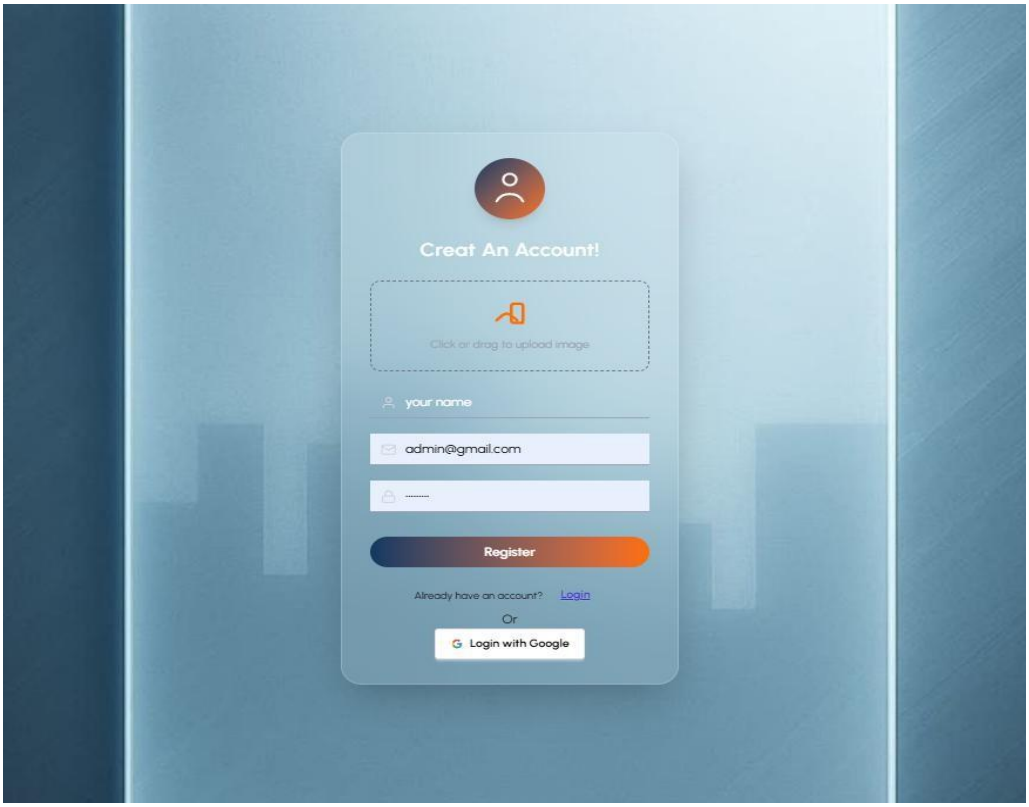
5.1 Introduction

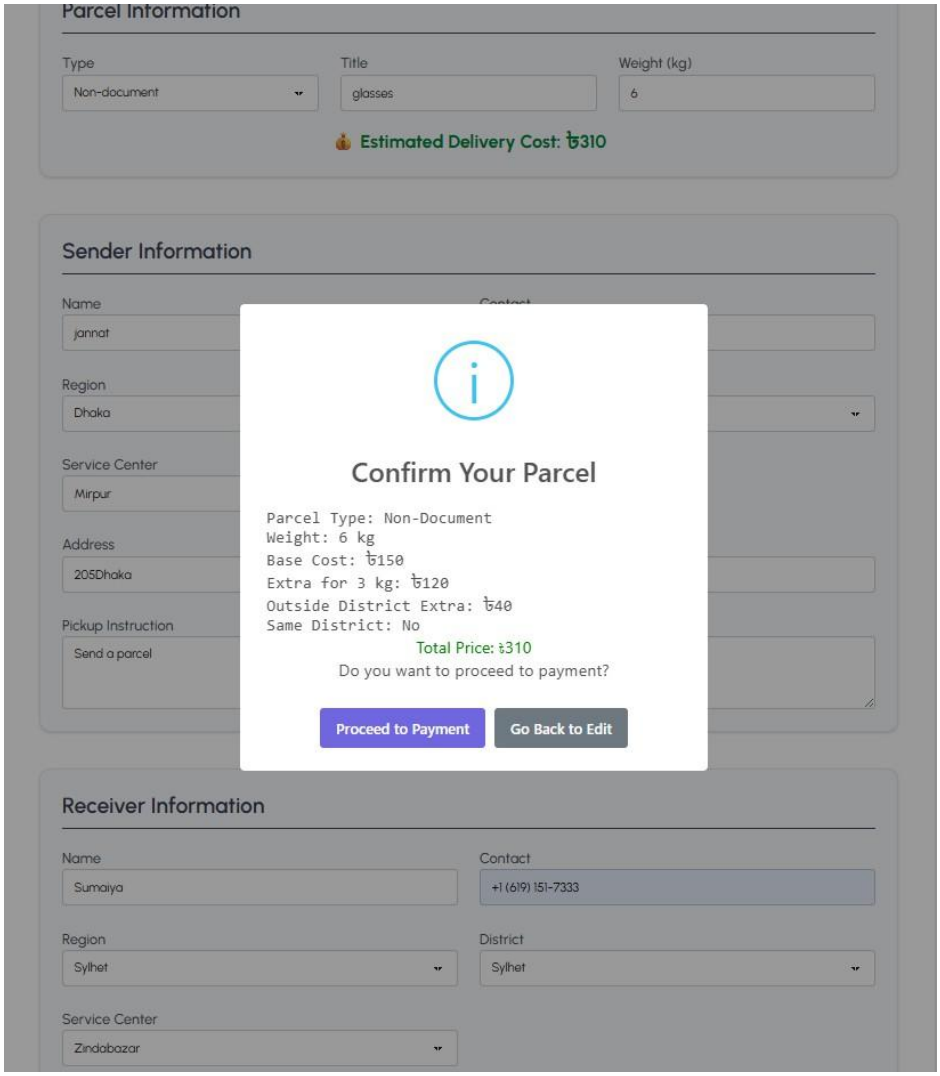
This section provides an easy to follow step by step manual on how to use this system for different roles. This guide helps all three kinds of users (users, riders and admins). They can easily understand what they need to do, where they need to go, what task they need to finish. The manual is designed in a way so that any kind of user can understand it easily. This ensures our system can be used by any kinds of users regarding their background.

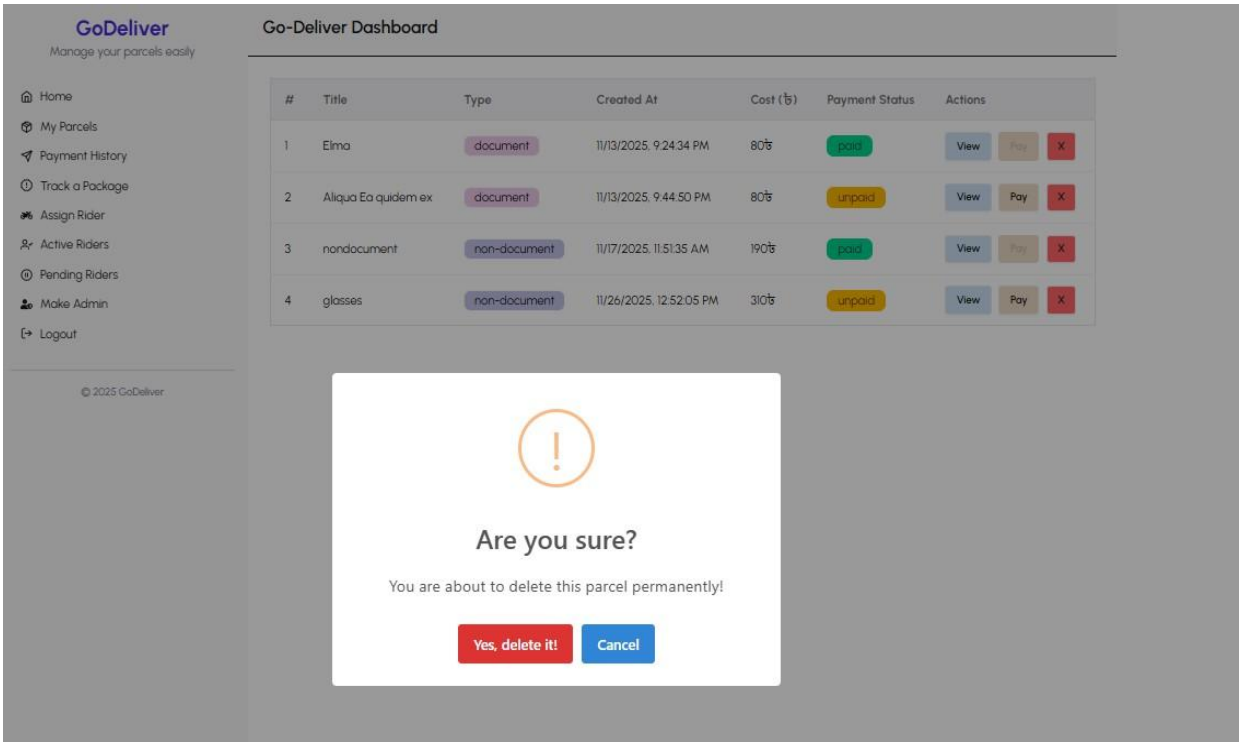
5.2 Project Functionalities

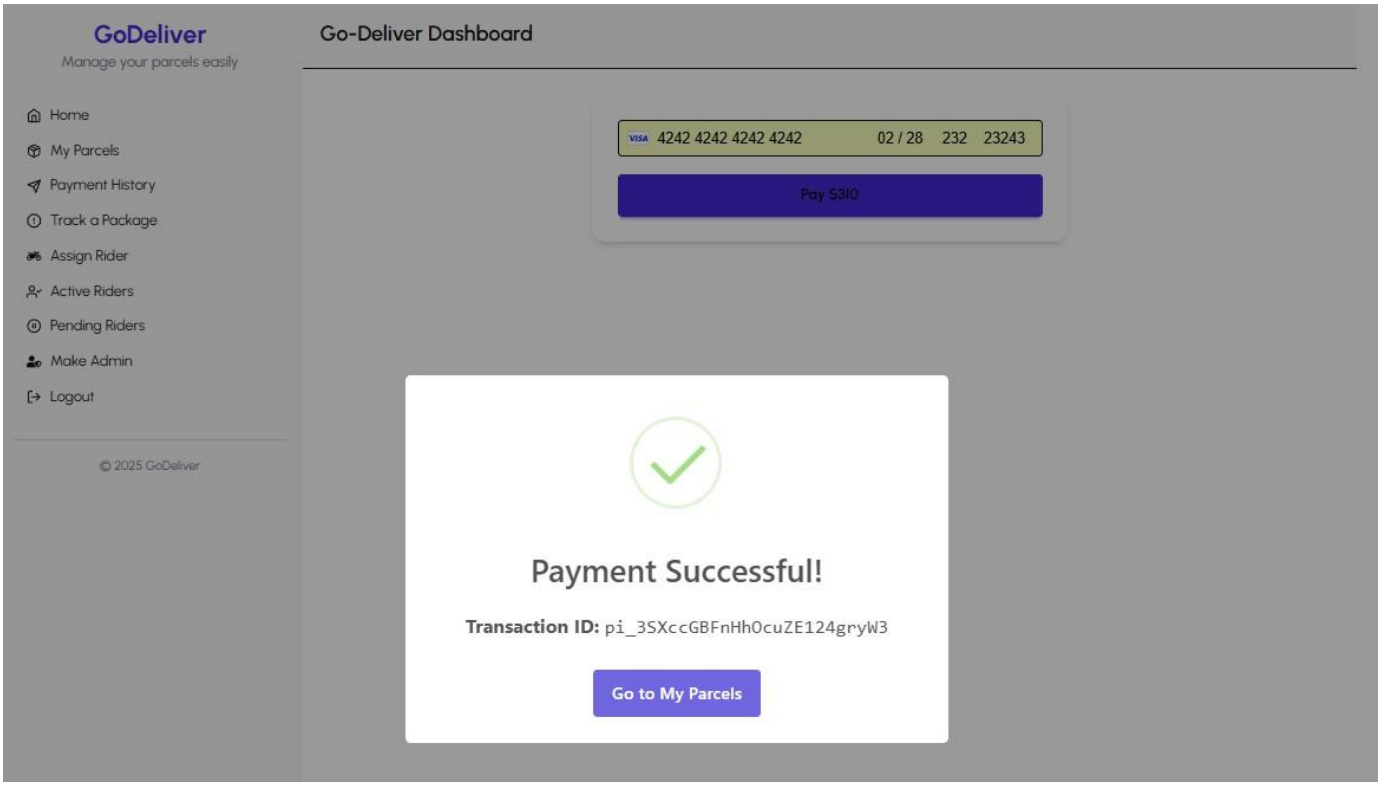


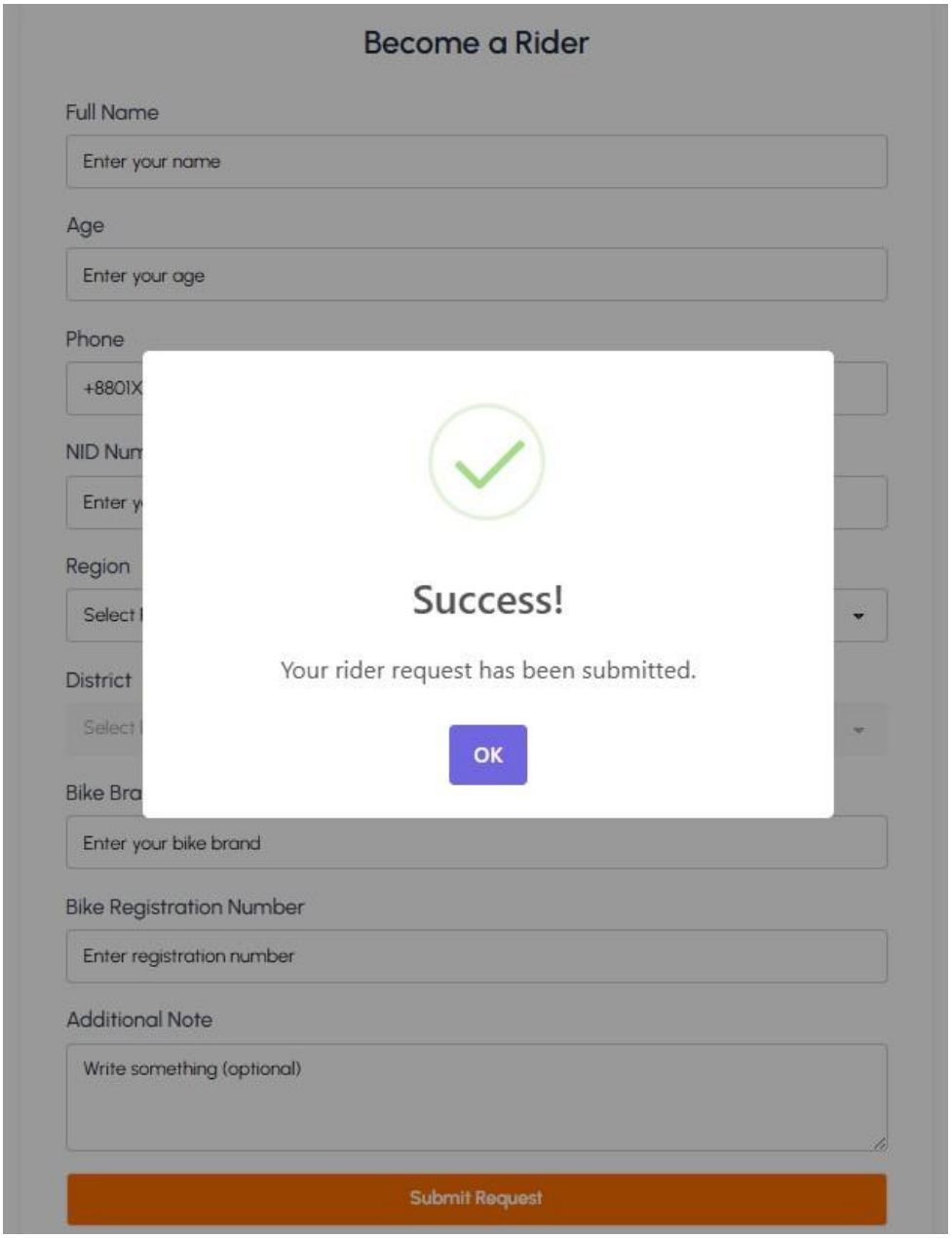

Please Login
admin@gmail.com
LOGIN
New to this website? Register
Or
Login with Google


Creat An Account!
Click or drag to upload image
your name
admin@gmail.com
Register
Already have an account? Login
Or
Login with Google


Parcel Information
Type
Non-document
Title
glasses
Weight (kg)
6
Estimated Delivery Cost: ৳310
Sender Information
Name
jannat
Region
Dhaka
Service Center
Mirpur
Address
205Dhaka
Pickup Instruction
Send a parcel
Confirm Your Parcel
Parcel Type: Non-Document
Weight: 6 kg
Base Cost: ৳150
Extra for 3 kg: ৳120
Outside District Extra: ৳40
Same District: No
Total Price: ৳310
Do you want to proceed to payment?
Proceed to Payment
Go Back to Edit
Receiver Information
Name
Sumaiya
Contact
+1 (619) 151-7333
Region
Sylhet
District
Sylhet
Service Center
Zindabazar


GoDeliver
Manage your parcels easily
Go-Deliver Dashboard
Home
My Parcels
Payment History
Track a Package
Assign Rider
Active Riders
Pending Riders
Make Admin
Logout
© 2025 GoDeliver
Title Type Created At Cost (৳) Payment Status Actions
1 Elma document 11/13/2025, 9:24:34 PM 80৳ paid View Pay X
2 Aliqua Ea quidem ex document 11/13/2025, 9:44:50 PM 80৳ unpaid View Pay X
3 nondocument non-document 11/17/2025, 11:51:35 AM 190৳ paid View Pay X
4 glasses non-document 11/26/2025, 12:52:05 PM 310৳ unpaid View Pay X
Are you sure?
You are about to delete this parcel permanently!
Yes, delete it!
Cancel

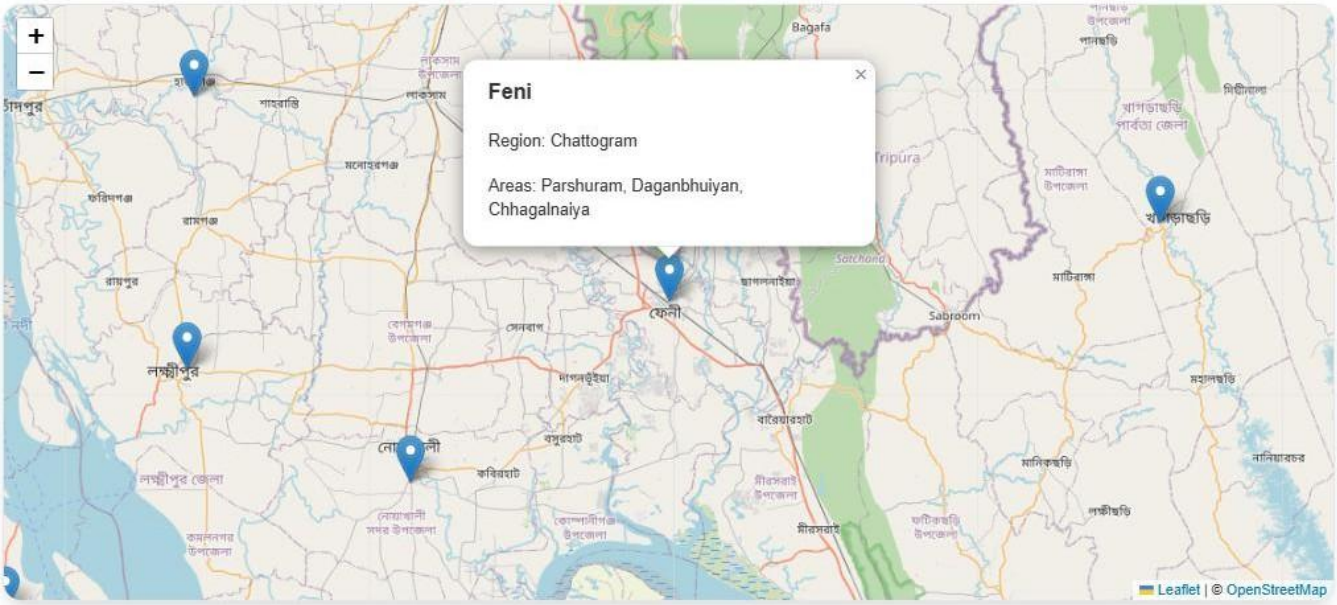

GoDeliver
Manage your parcels easily
Go-Deliver Dashboard
Home
My Parcels
Payment History
Track a Package
Assign Rider
Active Riders
Pending Riders
Make Admin
Logout
© 2025 GoDeliver
4242 4242 4242 4242 02 / 28 232 23243
Pay $310
Payment Successful!
Transaction ID: pi_3SXccGBFnHhOcuZE124gryW3
Go to My Parcels

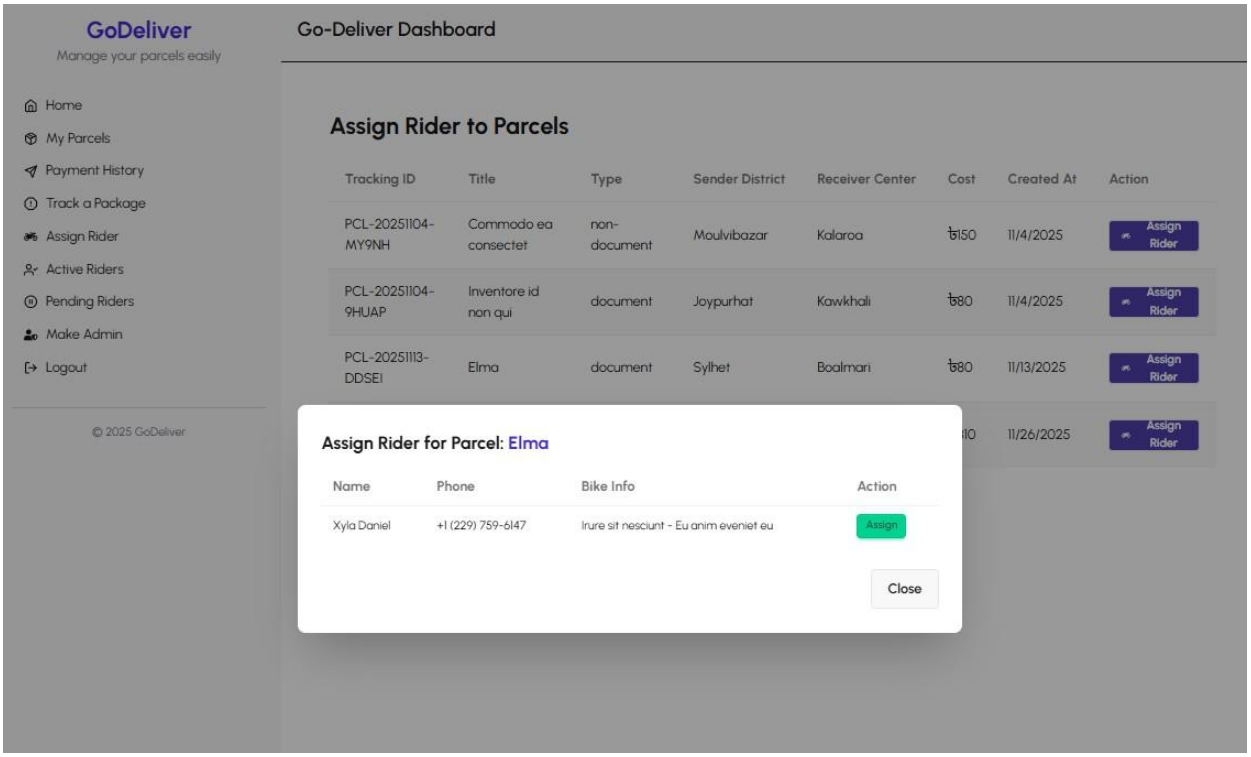

Become a Rider
Full Name
Enter your name
Age
Enter your age
Phone
+8801X
NID Num
Enter y
Region
Select
District
Select
Bike Bra
Enter your bike brand
Bike Registration Number
Enter registration number
Additional Note
Write something (optional)
Submit Request
Success!
Your rider request has been submitted.
OK

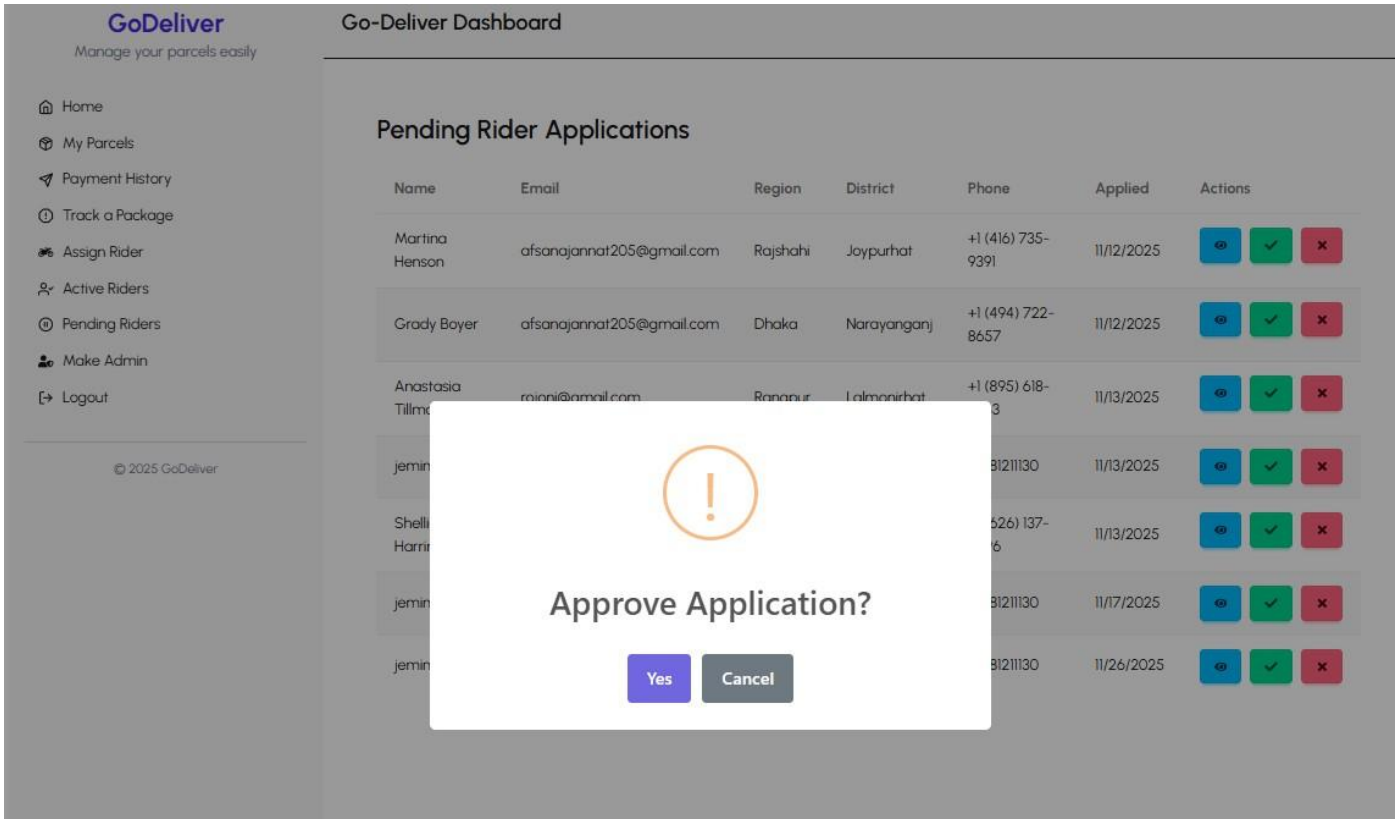
We are available in 64 districts

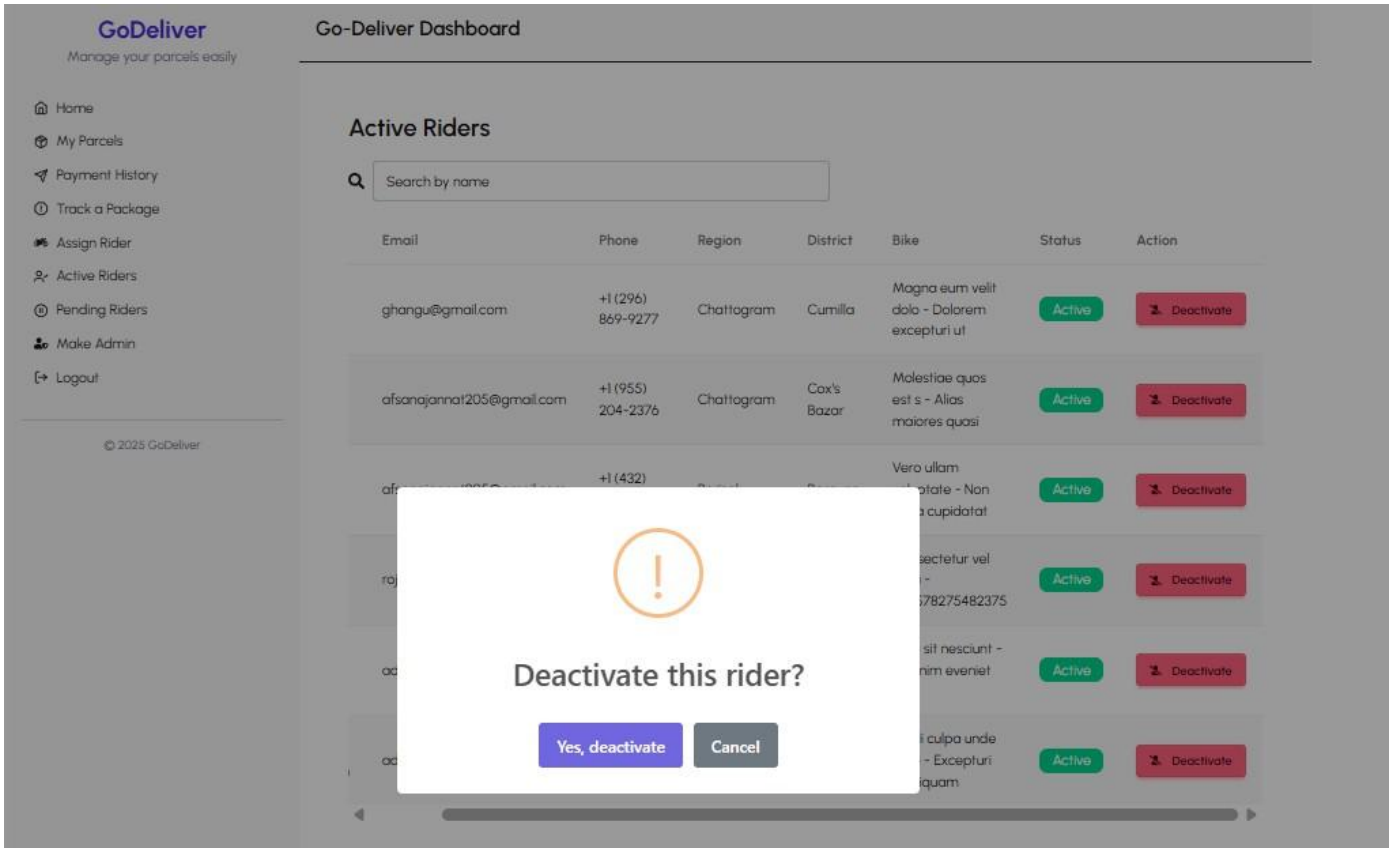
feni | Search

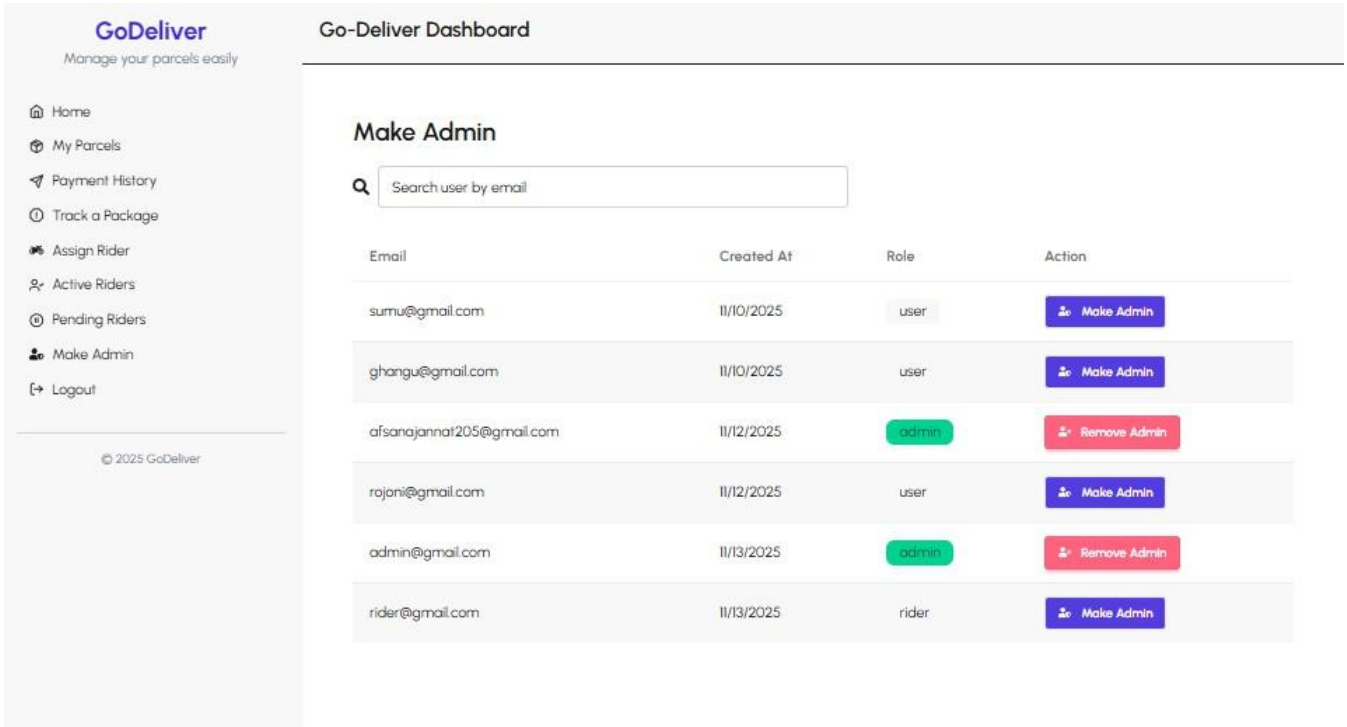
We deliver almost all over Bangladesh






GoDeliver
Manage your parcels easily
Home
My Parcels
Payment History
Track a Package
Assign Rider
Active Riders
Pending Riders
Make Admin
Logout
© 2025 GoDeliver
Go-Deliver Dashboard
Pending Rider Applications
Name
Email
Region
District
Phone
Applied
Actions
Martina Henson
afsanajannat205@gmail.com
Rajshahi
Joypurhat
+1 (416) 735-9391
11/12/2025
Grady Boyer
afsanajannat205@gmail.com
Dhaka
Narayanganj
+1 (494) 722-8657
11/12/2025
Anastasia
11/13/2025
11/13/2025
11/13/2025
11/17/2025
11/26/2025
Approve Application?
Yes
Cancel


GoDeliver
Manage your parcels easily
Home
My Parcels
Payment History
Track a Package
Assign Rider
Active Riders
Pending Riders
Make Admin
Logout
© 2025 GoDeliver
Go-Deliver Dashboard
Active Riders
Search by name
Email
Phone
Region
District
Bike
Status
Action
ghangu@gmail.com
+1 (296) 869-9277
Chattogram
Cumilla
Magna eum velit dolo - Dolorem excepturi ut
Active
Deactivate
afsanajannat205@gmail.com
+1 (955) 204-2376
Chattogram
Cox's Bazar
Molestiae quos est s - Alias maiores quasi
Active
Deactivate
Vero ullam
tate - Non
cupidatat
Active
Deactivate
ectetur vel
78275482375
Active
Deactivate
sit nesciunt -
nim eveniet
Active
Deactivate
culpa unde
- Excepturi
quam
Active
Deactivate
Deactivate this rider?
Yes, deactivate
Cancel


GoDeliver
Manage your parcels easily
Home
My Parcels
Payment History
Track a Package
Assign Rider
Active Riders
Pending Riders
Make Admin
Logout
© 2025 GoDeliver
Go-Deliver Dashboard
Make Admin
Search user by email
Email
Created At
Role
Action
sumu@gmail.com
11/10/2025
user
Make Admin
ghangu@gmail.com
11/10/2025
user
Make Admin
afsanajannat205@gmail.com
11/12/2025
admin
Remove Admin
rojani@gmail.com
11/12/2025
user
Make Admin
admin@gmail.com
11/13/2025
admin
Remove Admin
rider@gmail.com
11/13/2025
rider
Make Admin

5.3 **Summary**

After learning the various user roles, any one can use this website without any previous technical knowledge. The instructions help the users to understand and navigate the website effectively, how to order a parcel, how to complete a delivery, how to manage and assign a delivery to a designated rider, everything is implemented by keeping in mind good UI/UX rules.

Chapter 6 Project Summary

6.1 Introduction

This section talks about the overview of the GoDeliver project. How the system works, how each program was implemented, the future goal and the feature which was built currently, the constraints.

6.2 Project Limitation

Even though the project is complete, there are some limitations while developing this project. The main one was time, which hindered the process to add more advanced features that were planned initially. As the time was short, some features were normalized and simplified to ensure a complete delivery process system.

Another limitation was the budget of this project, as this project was completely done by a student, the budget is completely 0. All technologies used here are freely available to the public. As a result implementing features such as GPS tracking which required subscription wasn't added due to lack of funds.

Some features which were planned at start such as live chat support, multi language interface and rider assignment automation were not implemented due to time constraints.

6.3 Scope

The project GoDeliver mainly focuses on creating a simple and user-friendly platform for handling parcels online throughout Bangladesh.User can create accounts, book one or more parcels, make online payments and all. Riders can view their pending deliveries, update the current status of the delivery etc. Admin can manage the full process from a centralized account.

We also implement important modules, for example signin and signup for user authentication, booking of a parcel and tracking. These features ensure a smooth delivery process from booking to final delivery to the user.

Some features are not yet implemented and not within the scope, such as GPS tracking in real time, routing optimization which will happen automatically, live chart support, AI based forecasting of packages or integration with other courier services. Currently it does not handle cash on delivery service.

6.4 Future Work

In the upcoming future, our main goal is to add a GPS tracking system which will work in real time, introduction of mobile apps, AI integration, rider assignment automation, chart support and multi language capability. The goal is to scale the system horizontally by implementing new features day by day.

6.5 Conclusion

This project successfully developed a helpful and easy to understand parcel delivery management system. Users can book parcels, admins can then assign those parcels to designated riders and finally riders will deliver the parcel to the designated location. Completing a full online delivering process.

REFERENCES

- MongoDB, MongoDB Documentation from https://www.mongodb.com/docs
- Stripe. (n.d.). *Stripe API Reference* from https://stripe.com/docs/api
- Firebase. from https://firebase.google.com/docs/admin
- Node.js. *Node.js Documentation*. from https://nodejs.org/en/docs
- Express.js. *Express.js Guide*. from https://expressjs.com/en/guide

Afsana Jannat

221-35-851

Quick Submit

Quick Submit

Daffodil International University

Document Details

Submission ID

trn:oid:::1:3450436746

Submission Date

Dec 24, 2025, 11:54 AM GMT+6

Download Date

Dec 24, 2025, 12:30 PM GMT+6

File Name

GodeliverModify1.pdf

File Size

5.3 MB

73 Pages

5,963 Words

30,904 Characters

0% detected as AI

The percentage indicates the combined amount of likely AI-generated text as well as likely AI-generated text that was also likely AI-paraphrased.

Caution: Review required.

It is essential to understand the limitations of AI detection before making decisions about a student's work. We encourage you to learn more about Turnitin's AI detection capabilities before using the tool.

Detection Groups

0 AI-generated only **0%**
Likely AI-generated text from a large-language model.

0 AI-generated text that was AI-paraphrased **0%**
Likely AI-generated text that was likely revised using an AI-paraphrase tool or word spinner.

Disclaimer
Our AI writing assessment is designed to help educators identify text that might be prepared by a generative AI tool. Our AI writing assessment may not always be accurate (i.e., our AI models may produce either false positive results or false negative results), so it should not be used as the sole basis for adverse actions against a student. It takes further scrutiny and human judgment in conjunction with an organization's application of its specific academic policies to determine whether any academic misconduct has occurred.

Frequently Asked Questions

How should I interpret Turnitin's AI writing percentage and false positives?
The percentage shown in the AI writing report is the amount of qualifying text within the submission that Turnitin's AI writing detection model determines was either likely AI-generated text from a large-language model or likely AI-generated text that was likely revised using an AI paraphrase tool or word spinner.

False positives (incorrectly flagging human-written text as AI-generated) are a possibility in AI models.

AI detection scores under 20%, which we do not surface in new reports, have a higher likelihood of false positives. To reduce the likelihood of misinterpretation, no score or highlights are attributed and are indicated with an asterisk in the report (*%).

The AI writing percentage should not be the sole basis to determine whether misconduct has occurred. The reviewer/instructor should use the percentage as a means to start a formative conversation with their student and/or use it to examine the submitted assignment in accordance with their school's policies.

What does 'qualifying text' mean?
Our model only processes qualifying text in the form of long-form writing. Long-form writing means individual sentences contained in paragraphs that make up a longer piece of written work, such as an essay, a dissertation, or an article, etc. Qualifying text that has been determined to be likely AI-generated will be highlighted in cyan in the submission, and likely AI-generated and then likely AI-paraphrased will be highlighted purple.

Non-qualifying text, such as bullet points, annotated bibliographies, etc., will not be processed and can create disparity between the submission highlights and the percentage shown.