A

Project Report

On

Online Mobile Recharge System

Submitted in partial fulfillment of the requirement for the degree of

Bachelor of Technology

In

Computer Science and Engineering

By

Nikhil Papola	2261392
Sagar Singh Bisht	2261501
Mayank Upadhyay	2261363
Vikram Singh Chilwal	2261606

Under the Guidance of

Mr. Anubhav Bewerwal

Lecturer

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

GRAPHIC ERA HILL UNIVERSITY, BHIMTAL CAMPUS

SATTAL ROAD, P.O. BHOWALI, DISTRICT-

NAINITAL-263132

2024-2025

STUDENT'S DECLARATION

We, Nikhil papola, Sagar singh bisht, Mayank upadhyay and Vikram singh chilwal hereby declare the work, which is being presented in the project, entitled**Online mobile recharge system** in partial fulfillment of the requirement for the award of the degree Bachelor of Technology (B.Tech.) in the session 2024-2025, is an authentic record of my work carried out under the supervision of Mr. Anubhav Bewerwal.

The matter embodied in this project has not been submitted by me for the award of any other degree.

Date:

Nikhil Papola

Sagar Singh Bisht

Mayank Upadhyay

Vikram Singh Chilwal

CERTIFICATE

The project report entitled "onlineRechargeSystem" being submitted by Nikhil Papola, Mayank upadhyay,Sagar singh bisht and Vikram singh chilwal D/o Mr. Narayan Singh,Mr. D.D Upadhyay, Mr. Kailash Singh and Mr. Rajendra Singh, University Roll Nos . 2261392, 2261363, 2261501 and 2261606 respectively of B.Tech.(CSE) to Graphic Era Hill University Bhimtal Campus for the award of bonafide work carried out by them. They have worked under my guidance and supervision and fulfilled the requirement for the submission of a report.

Mr. Anubhav Bewerwal
(Project Guide)

Dr. Ankur Singh Bisht
(Head, CSE)

ACKNOWLEDGEMENT

We take immense pleasure in thanking the Honorable Director 'Prof. (Col.) Anil Nair (Retd.)', GEHU Bhimtal Campus to permit me and carry out this project work with his excellent and optimistic supervision. This has all been possible due to his novel inspiration, able guidance, and useful suggestions that helped me to develop as a creative researcher and complete the research work, in time.

Words are inadequate in offering my thanks to GOD for providing me with everything that we need. We again want to extend thanks to our president 'Prof. (Dr.) Kamal Ghanshala' for providing us with all infrastructure and facilities to work in need without which this work could not be possible

Many thanks to 'Dr. Ankur Singh Bisht' (Head, Department of Computer Science and Engineering, GEHU Bhimtal Campus), our project guide 'Mr. Anubhav Bewerwal' (Lecturer, Department of Computer Science and Engineering, GEHU Bhimtal Campus) and other faculties for their insightful comments, constructive suggestions, valuable advice, and time in reviewing this report.

Finally, yet importantly, We would like to express my heartiest thanks to our beloved parents, for their moral support, affection, and blessings. We would also like to pay our sincere thanks to all my friends and well-wishers for their help and wishes for the successful completion of this project.

Nikhil Papola **2261392**
Sagar Singh Bisth **2261501**
Mayank Upadhyay **2261363**
Vikram Singh Chilwal **2261606**

ABSTRACT

This project titled **"Online Mobile Recharge System using MERN Stack"** is a full-stack web application that enables users to recharge their mobile phones online in a secure, efficient, and user-friendly manner. Built using the MERN stack—MongoDB, Express.js, React.js, and Node.js—the system provides real-time interaction between the user and the backend services.

The application allows users to register and log in securely, browse available mobile recharge plans based on different operators, and perform recharges using integrated payment gateways. All user data, transaction history, and recharge plans are managed using a cloud-based MongoDB database. The backend server, built with Node.js and Express.js, handles API requests, authentication, and database communication. The React.js frontend offers a dynamic and responsive interface for users to interact with the system.

Additionally, the system includes an admin panel that allows administrators to manage recharge plans, view transaction logs, and monitor system usage. The project emphasizes modern web development best practices, including JWT-based authentication, RESTful API design, and responsive UI components.

This system streamlines the mobile recharge process, reduces the need for physical vendors, and enhances user convenience by providing a digital alternative accessible 24/7.

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

MERN: MongoDB, Express.js, React.js, Node.js – A stack used for building full-stack web applications.

API: Application Programming Interface – A set of rules that allows software programs to communicate.

JWT: JSON Web Token – A compact, URL-safe method for representing claims between two parties

DB: Database –An organized collection of structured information or data.

UI: User Interface – The space where interactions between humans and machines occur.

UX: User Experience – Refers to a person's emotions and attitudes about using a particular product or service.

CRUD: Create, Read, Update, Delete – Basic operations of persistent storage.

SDK: Software Development Kit – A collection of software tools and programs for development

SSL: Secure Sockets Layer – A protocol for encrypting information over the internet .

CDN: Content Delivery Network – A system of servers that deliver web content based on the user's location.

CMS: Content Management System – A software that helps users create, manage, and modify content on a website.

DOM: Document Object Model – A programming interface for web documents.

.

INTRODUCTION

1.1 Prologue

In the digital era, mobile users seek fast and reliable online recharge solutions. This project, built using the MERN stack, addresses that need by offering a secure, user-friendly platform for mobile recharges. It combines real-time functionality, efficient data handling, and responsive design, providing valuable experience in full-stack development while solving a relevant real-world problem.

1.2 Background and Motivations

With the rapid growth of mobile users and digital payments, the need for an efficient online recharge system has become essential. Traditional recharge methods are time-consuming and inconvenient. This project is motivated by the goal to simplify the recharge process through a seamless web platform. Leveraging the MERN stack allows for modern, scalable development, ensuring users enjoy quick, secure, and hassle-free mobile recharges anytime, anywhere.

1.3 Problem Statement

Traditional mobile recharge methods often involve delays, lack convenience, and require physical presence or third-party vendors. Users face difficulties in accessing updated plans and ensuring secure payments. This project aims to solve these issues by developing a webbased system using the MERN stack, enabling users to perform secure, real-time mobile recharges with updated plans and transaction tracking.

1.4 Objectives and Research Methodology

Objective:

- Develop a full-stack web application for mobile recharge using the MERN stack (MongoDB, Express.js, React.js, Node.js).
- Implement secure user authentication to ensure safe access and transaction handling.
- Integrate real-time recharge functionality with dynamic plan selection based on operators.
- Provide an admin panel for managing users, plans, and viewing transaction logs.

Research Methodology:

- Literature Review: Studied existing mobile recharge systems and current industry practices to understand user needs and system requirements.

- Requirement Analysis: Identified key functional and non-functional requirements for user and admin roles.
- Design and Development: Used modular design patterns and MERN stack technologies to build the system in phases.
- Testing and Evaluation: Conducted unit testing, integration testing, and user feedback analysis to ensure system reliability and performance.

This document is structured as follows:

- **Chapter 1** introduces the project, motivation, goals, and methodology.
- **Chapter 2** covers the system requirements — both hardware and software.
- **Chapter 3** presents the core logic, coding functions, and backend integration.
- **Chapter 4** includes interface snapshots and a walkthrough of key screens.
- **Chapter 5** outlines limitations and known issues in the current version.
- **Chapter 6** suggests potential future enhancements.
- **Chapter 7** concludes with a summary and project reflection.
- The **References** section lists all tools, frameworks, and resources used.

PHASES OF SOFTWARE DEVELOPMENT CYCLE

2.1. Hardware Requirement:

Component	Windows	macOS	Linux
Operating System	Windows 10 or later (64-bit)	macOS 10.13 (High Sierra) or later	Ubuntu 20.04 or later / GNOME, KDE, or Unity desktop
RAM	Minimum: 8 GBRecommended: 16 GB (for dev tools, local servers, and image handling)	Minimum: 8 GBRecommended: 16 GB	Minimum: 8 GBRecommended: 16 GB
Storage	At least 5 GB free (includes Node.js modules, image cache, DB storage)	At least 5 GB free	At least 5 GB free
Processor	Intel i5 or higher / AMD equivalent with virtualization support	Apple Silicon (M1/M2) or Intel Core i5/i7	Intel i5 or higher / AMD Ryzen
Optional Requirements	Intel VT-x, EM64T, XD Bit for Docker or emulator-based testing	Virtualization support (for Docker/containers)	Glibc 2.17+ and virtualization enabled if needed
Display	Minimum 1280×800 resolution	Minimum 1280×800 resolution	Minimum 1280×800 resolution

2.2 Software Requirement:

Node.js (v14 or higher)
MongoDB (Cloud-based using MongoDB Atlas)
Visual Studio Code or any code editor
Git (Version Control System)
React.js (Frontend Library)

Express.js (Web Framework for Node.js)
Cloudinary (for Image Storage and Management)
Stripe (for Payment Gateway Integration)
Postman (for API testing)
Any modern browser (Chrome, Firefox, Edge, etc.)

CODING OF FUNCTIONS

1. Backend

1.1 index.js

```
const express = require("express");
const session = require("express-session");
const passport = require("passport");
const cors = require("cors");
const mongoose = require("mongoose");
require("dotenv").config();

require("./services/passport"); // Google OAuth config

const authRoutes = require("./routes/auth");
const paymentRoute = require("./routes/payment");
const rechargeRoute = require("./routes/recharge");
const plansRoute = require("./routes/plans"); // ✅ ADD THIS

const app = express();

// ✅ CORS
app.use(
  cors({
    origin: "http://localhost:3000",
    credentials: true,
  })
);

// ✅ JSON parser
app.use(express.json());

// ✅ Session
app.use(
  session({
    secret: process.env.COOKIE_KEY || "some-secret-key",
    resave: false,
    saveUninitialized: false,
    cookie: {
      maxAge: 24 * 60 * 60 * 1000,
      secure: false,
      sameSite: "lax",
    },
  })
```

```
);

// ✅ Passport
app.use(passport.initialize());
app.use(passport.session());

// ✅ Routes
app.use("/auth", authRoutes);
app.use("/api/payment", paymentRoute);
app.use("/api/recharge", rechargeRoute);
app.use("/api/plans", plansRoute); // ✅ ADD THIS LINE

// ✅ Root
app.get("/", (req, res) => res.send("Server is running"));

console.log("Connecting to MongoDB at:", process.env.MONGODB_URI);

// ✅ MongoDB Connect + Start Server
mongoose.connect(process.env.MONGODB_URI, {
  useNewUrlParser: true,
  useUnifiedTopology: true,
})
.then(() => {
  console.log("✅ Connected to MongoDB");
  app.listen(process.env.PORT || 5000, () =>
    console.log(🚀 Server running on port ${process.env.PORT || 5000})
  );
})
.catch((err) => {
  console.error("❌ MongoDB connection error:", err);
});
```

1.2 payment/route

```
const express = require("express");
const Razorpay = require("razorpay");
const router = express.Router();
const Recharge = require("../models/Recharge");

const razorpay = new Razorpay({
  key_id: process.env.RAZORPAY_KEY_ID,
  key_secret: process.env.RAZORPAY_KEY_SECRET,
});

// ✅ Step 1: Create Razorpay order
router.post("/create-order", async (req, res) => {
  const { amount } = req.body;

  const options = {
    amount: amount * 100, // Amount in paisa
    currency: "INR",
    receipt: "receipt_order_" + Date.now(),
  };

  try {
    const order = await razorpay.orders.create(options);
    res.json(order); // ✅ Send order back to frontend
  } catch (err) {
    console.error("Order creation failed:", err);
    res.status(500).json({ error: "Failed to create Razorpay order" });
  }
});

// ✅ Step 2: Save recharge after successful payment
router.post("/verify", async (req, res) => {
  const {
    number,
    provider,
    planId,
    amount,
    paymentId,
    orderId,
    userEmail,
  } = req.body;

  try {
```

```
    const newRecharge = new Recharge({
      number,
      provider,
      planId,
      amount,
      razorpay_payment_id: paymentId,
      razorpay_order_id: orderId,
      userEmail,
      date: new Date(),
    });

    await newRecharge.save();
    res.json({ success: true, message: "Recharge saved successfully" });
  } catch (error) {
    console.error("Recharge save error:", error);
    res.status(500).json({ error: "Recharge save failed" });
  }
});
module.exports = router;
```

1.3 recharge/models

```
const mongoose = require("mongoose");

const RechargeSchema = new mongoose.Schema({
 number: String,
 provider: String,
 planId: Number,
 amount: Number,
 razorpay_payment_id: String,
 razorpay_order_id: String,
 userEmail: String,
 date: { type: Date, default: Date.now },
});

module.exports = mongoose.model("Recharge", RechargeSchema);
```

1.4 .env

```
GOOGLE_CLIENT_ID=871396694177-
bill4cje92efgud7p8fmhumhnqjpn312.apps.googleusercontent.com
GOOGLE_CLIENT_SECRET=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
```

```
COOKIE_KEY=your_random_secure_cookie_key
BASE_URL=http://localhost:5000

RAZORPAY_KEY_ID=rzp_test_CayIxTstjZievD
RAZORPAY_KEY_SECRET=v9sEvD3NFQyI7FdyekyhqvGo

MONGODB_URI=mongodb+srv://bishtsagar8958:bishtsagar8958@cluster0.irl3r0g.mongodb.net/?retryWrites=true&w=majority&appName=Cluster0
```

2. Frontend

2.1 App.js

```
import { useEffect, useState } from "react";
import axios from "axios";
import { BrowserRouter as Router, Routes, Route } from "react-router-dom";

import Plans from "./components/Plans";
import Dashboard from "./components/Dashboard";
import Login from "./components/Login";
import History from "./components/History";
import RechargeForm from "./components/RechargeForm"; // ✅ Import RechargeForm
import AdminDashboard from "./components/AdminDashboard";



function App() {
  const [user, setUser] = useState(null);

  useEffect(() => {
    axios
      .get("http://localhost:5000/auth/user", { withCredentials: true })
      .then((res) => setUser(res.data))
      .catch((err) => console.log("Not logged in"));
  }, []);

  if (!user) {
    return <Login />;
  }

  const userEmail = user.emails[0].value;
```

```
  return (
    <Router>
      <div>
        <header style={{ padding: "10px", backgroundColor: "#f0f0f0" }}>
          <p>Welcome, {user.displayName}</p>
          <p>Email: {userEmail}</p>
          <a href="http://localhost:5000/auth/logout">
            <button>Logout</button>
          </a>
        </header>

        <Routes>
          <Route path="/dashboard" element={<Dashboard userEmail={userEmail} />} />
          <Route path="/plans" element={<Plans userEmail={userEmail} />} />
          <Route path="/recharge" element={<RechargeForm />} /> {/* ✅ Added Recharge
Route */}
          <Route path="/history" element={<History userEmail={userEmail} />} />
          <Route path="*" element={<Dashboard userEmail={userEmail} />} />
          <Route path="/admin" element={<AdminDashboard userEmail={userEmail} />} />


        </Routes>
      </div>
    </Router>
  );
}

export default App;
```

2.2 AdminPanel/components

```
import React, { useEffect, useState } from "react";
import axios from "axios";

function AdminPanel() {
  const [recharges, setRecharges] = useState([]);
  const adminEmail = "admin@example.com"; // Match this with backend

  useEffect(() => {
    const fetchRecharges = async () => {
      try {
        const res = await
axios.get(http://localhost:5000/api/admin/recharges?email=${adminEmail});
        setRecharges(res.data);
      } catch (err) {
        console.error("Failed to load admin data", err);
      }
    };

    fetchRecharges();
  }, []);

  return (
    <div>
      <h2>All Recharge Transactions</h2>
      <table border="1">
        <thead>
          <tr>
            <th>User Email</th>
            <th>Amount</th>
            <th>Provider</th>
            <th>Validity</th>
            <th>Data</th>
            <th>Date</th>
          </tr>
        </thead>
        <tbody>
          {recharges.map((r) => (
            <tr key={r._id}>
              <td>{r.userEmail}</td>
              <td>₹{r.amount}</td>
              <td>{r.provider}</td>
              <td>{r.planDetails?.validity}</td>
              <td>{r.planDetails?.data}</td>
```

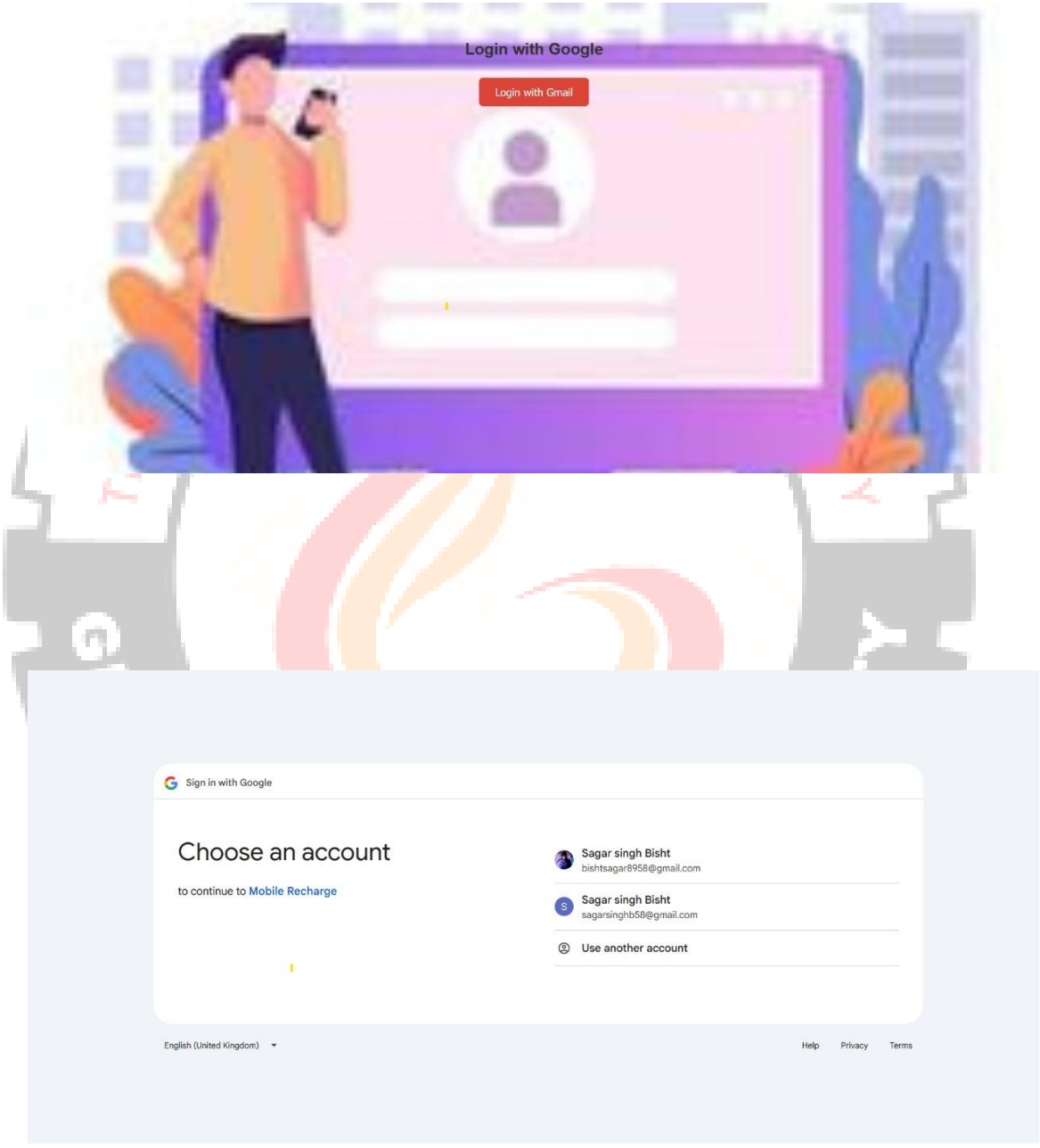
```
            <td>{new Date(r.date).toLocaleString()}</td>
          </tr>
        ))}
      </tbody>
    </table>
  </div>
 );
}

export default AdminPanel;
```

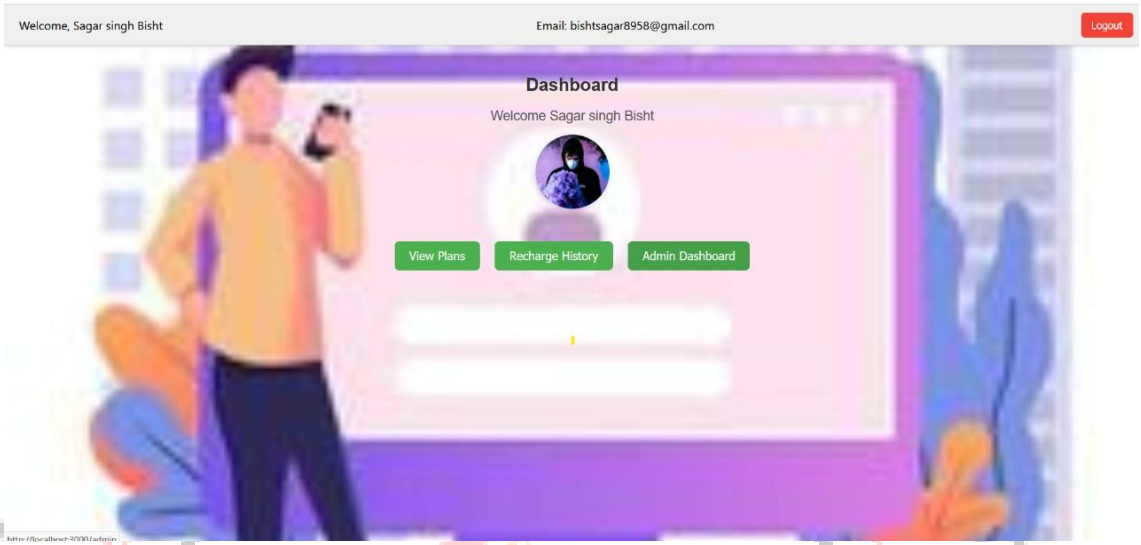
2.3 AdminDashboard/components

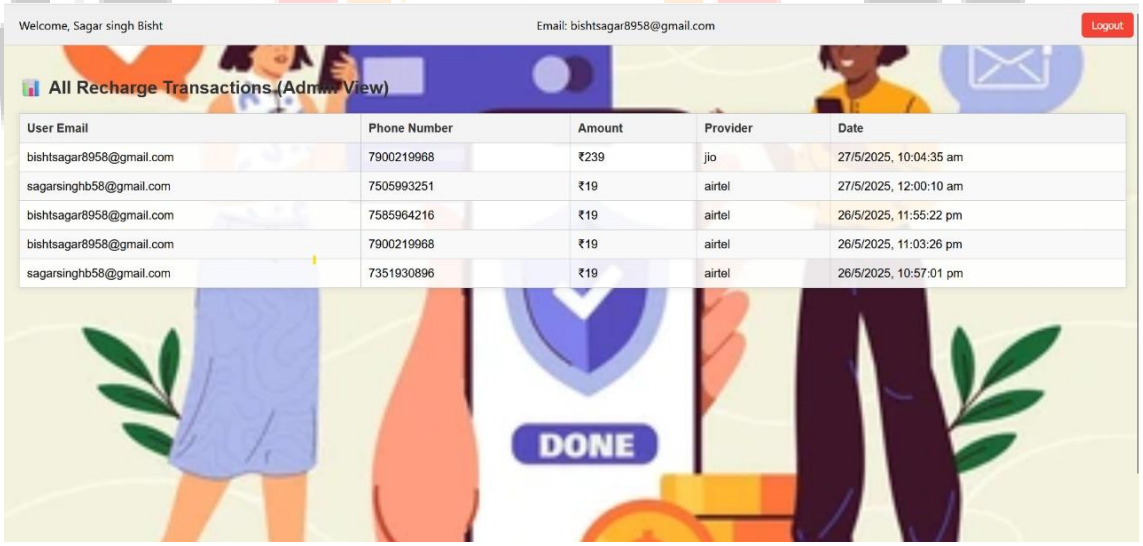
```
import { useEffect, useState } from "react";
import axios from "axios";

function AdminDashboard({ userEmail }) {
 const [recharges, setRecharges] = useState([]);
 const [accessDenied, setAccessDenied] = useState(false);

 useEffect(() => {
 if (!userEmail) return;

 const cleanedEmail = userEmail.trim().toLowerCase();
 if (cleanedEmail !== "bishtsagar8958@gmail.com") {
  setAccessDenied(true);
  return;
 }

 axios.get("http://localhost:5000/api/recharge/all")
  .then(res => setRecharges(res.data))
  .catch(err => console.error("Error loading recharges", err));
}, [userEmail]);


 if (accessDenied) {
  return (
   <div style={{ padding: "20px", color: "red" }}>
    <h2>⛔ Access Denied</h2>
    <p>You are not authorized to view this page.</p>
   </div>
  );
 }
```

```
 return (
   <div style={{ padding: "20px" }}>
     <h2>📊 All Recharge Transactions (Admin View)</h2>
     {recharges.length === 0 ? (
       <p>No transactions found.</p>
     ) : (
       <table border="1" cellPadding="10" style={{ width: "100%", marginTop: "20px" }}>
         <thead>
           <tr>
             <th>User Email</th>
             <th>Phone Number</th>
             <th>Amount</th>
             <th>Provider</th>
             <th>Date</th>
           </tr>
         </thead>
         <tbody>
           {recharges.map((r, i) => (
             <tr key={i}>
               <td>{r.userEmail}</td>
               <td>{r.number}</td>
               <td>₹{r.amount}</td>
               <td>{r.provider}</td>
               <td>{new Date(r.date).toLocaleString()}</td>
             </tr>
           ))}
         </tbody>
       </table>
     )}
   </div>
 );
}

export default AdminDashboard;
```

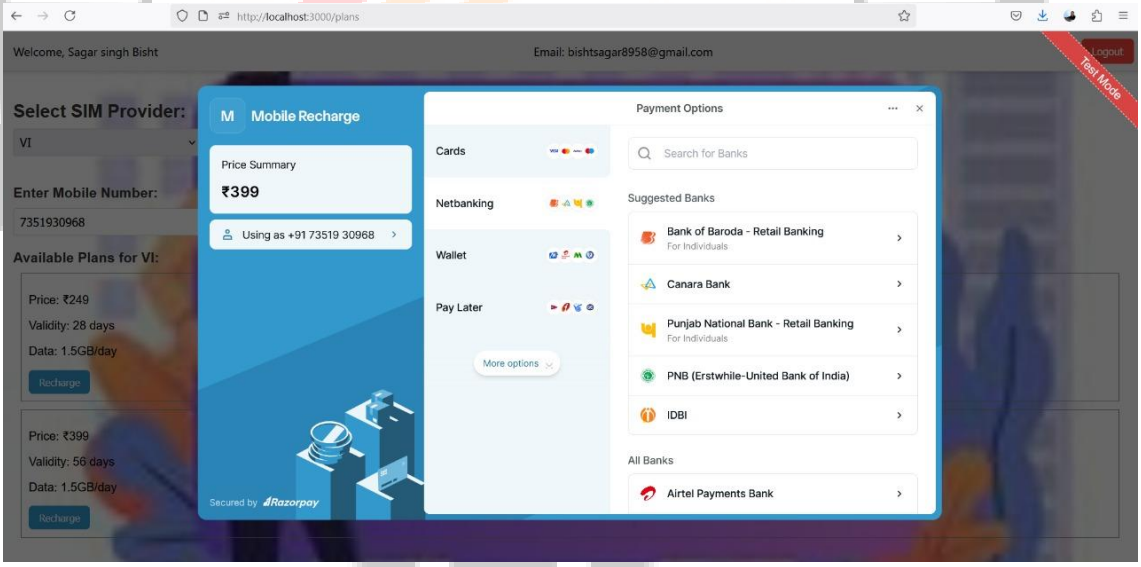
SNAPSHOTS

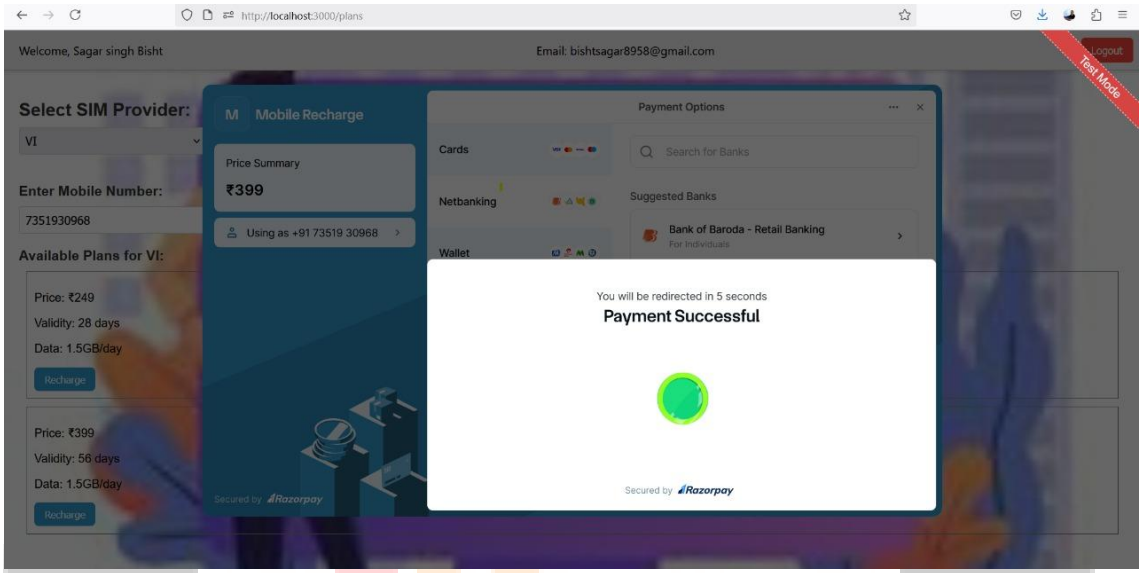

Login with Google
Login with Gmail
Sign in with Google
Choose an account
to continue to Mobile Recharge
Sagar singh Bisht
bishtsagar8958@gmail.com
Sagar singh Bisht
sagarsinghb58@gmail.com
Use another account
English (United Kingdom)
Help
Privacy
Terms


Welcome, Sagar singh Bisht
Email: bishtsagar8958@gmail.com
Logout
Dashboard
Welcome Sagar singh Bisht
View Plans
Recharge History
Admin Dashboard


Welcome, Sagar singh Bisht
Email: bishtsagar8958@gmail.com
Logout
All Recharge Transactions (Admin View)
User Email
Phone Number
Amount
Provider
Date
bishtsagar8958@gmail.com
7900219968
₹239
jio
27/5/2025, 10:04:35 am
sagarsinghb58@gmail.com
7505993251
₹19
airtel
27/5/2025, 12:00:10 am
bishtsagar8958@gmail.com
7585964216
₹19
airtel
26/5/2025, 11:55:22 pm
bishtsagar8958@gmail.com
7900219968
₹19
airtel
26/5/2025, 11:03:26 pm
sagarsinghb58@gmail.com
7351930896
₹19
airtel
26/5/2025, 10:57:01 pm
DONE


Welcome, Sagar singh Bisht
Email: bishtsagar8958@gmail.com
Logout
Select SIM Provider:
Jio
Enter Mobile Number:
Enter 10-digit number
Available Plans for JIO:
Price: ₹239
Validity: 28 days
Data: 1.5GB/day
Recharge
Price: ₹399
Validity: 56 days
Data: 1.5GB/day
Recharge


http://localhost:3000/plans
Welcome, Sagar singh Bisht
Email: bishtsagar8958@gmail.com
Logout
Test Mode
Select SIM Provider:
VI
Enter Mobile Number:
7351930968
Available Plans for VI:
Price: ₹249
Validity: 28 days
Data: 1.5GB/day
Recharge
Price: ₹399
Validity: 56 days
Data: 1.5GB/day
Recharge
M Mobile Recharge
Price Summary
₹399
Using as +91 73519 30968
Payment Options
Cards
Netbanking
Wallet
Pay Later
More options
Search for Banks
Suggested Banks
Bank of Baroda - Retail Banking
For Individuals
Canara Bank
Punjab National Bank - Retail Banking
For Individuals
PNB (Erstwhile-United Bank of India)
IDBI
All Banks
Airtel Payments Bank
Secured by Razorpay



LIMITATIONS

- **No Real-Time Operator Integration**
  The system does not actually connect to telecom operator APIs. Recharges are simulated and do not reflect in real mobile accounts.

- **Limited Payment Gateway Options**
  The system may only integrate with one payment provider (e.g., Razorpay), and lacks support for multiple payment options such as UPI, Net Banking, or Wallets.

- **No SMS Notification System**
  Users are not notified via SMS upon successful or failed recharges, limiting real-time feedback.

- **No Support for International Numbers**
  The system is designed for domestic users only; it does not support international mobile recharges.

- **Basic Error Handling**
  Some edge cases (e.g., payment timeout, interrupted transactions) may not be with robust recovery mechanisms.

- **No Offline Mode**
  The system requires a constant internet connection and cannot process recharges offline or queue them for later.

- **Single User Role**
  The current implementation may support only basic user and admin roles, without fine-grained access controls.

- **Limited Recharge Plans**
  Recharge plans are manually entered and may not reflect real-tim availability or promotional offers from telecom providers.

- **Security at Prototype Level**
  While basic authentication and input validation arein place, the system may lack enterprise-level security measures such as encryption at rest, rate limiting, or penetration testing.

ENHANCEMENTS

- **Integration with Real Telecom APIs**

Connect the system with actual telecom operators (e.g., Airtel, Jio) through their APIs to perform real mobile recharges.

- **Multi-Payment Gateway Support**

Add support for various payment methods such as UPI, PayPal, Net Banking, Credit/Debit cards, and digital wallets.

- **SMS and Email Notifications**

Send real-time recharge confirmations, OTPs, and transaction alerts via SMS and email to enhance user trust.

- **Recharge Plan Auto-Sync**

Use telecom APIs to auto-update available recharge plans so users always see the latest options.

- **Downloadable Invoices**

Allow users to download recharge receipts in PDF format for record-keeping or reimbursement.

- **Wallet System**

Implement a wallet feature where users can load money in advance and use it for quick recharges.

- **Mobile App Version**

Develop a mobile app using React Native or Flutter for a more convenient, on-the-go user experience.

- **Scheduled or Auto Recharge**

Enable users to schedule monthly recharges or set up auto-recharge for specific plans.

- **Admin Analytics Dashboard**

Provide visual charts and reports showing recharge trends, user activity, revenue, and system performance.

- **Multi-Language and Accessibility Support**

Add support for multiple languages and make the platform accessible for differently-abled users.

CONCLUSION

The **Online Mobile Recharge System** offers a streamlined, user-friendly platform for users to perform mobile recharges quickly and securely. By leveraging the **MERN stack (MongoDB, Express.js, React, Node.js)**, the system ensures efficient data handling, responsive user interfaces, and robust backend functionality. This project successfully demonstrates how modern web technologies can be combined to build a full-stack solution for real-world use cases.

While the current system handles core functionalities like user authentication, recharge plan selection, and payment processing, there is room for future enhancements such as real-time telecom API integration, wallet systems, and advanced user analytics. Overall, the project fulfills its objective of simplifying mobile recharge processes and lays a solid foundation for scaling into a more feature-rich, production-ready application.

REFERENCES

- **MongoDB Documentation**
https://www.mongodb.com/docs/

- **Express.js Official Guide**
https://expressjs.com/

- **React.js Documentation**
https://reactjs.org/docs/getting-started.html

- **Node.js Documentation**
https://nodejs.org/en/docs/

- **Razorpay Payment Gateway Integration**
https://razorpay.com/docs/

- **JWT (JSON Web Tokens) Authentication Guide**
https://jwt.io/introduction

- **BCrypt for Password Hashing**
https://www.npmjs.com/package/bcrypt

- **MERN Stack Tutorial by freeCodeCamp**
https://www.freecodecamp.org/news/learn-the-mern-stack/

- **Bootstrap Documentation (for UI components)**
https://getbootstrap.com/docs/

- **MDN Web Docs – JavaScript, HTML, CSS**
https://developer.mozilla.org/

- **Telecom API Providers (for future integration)**

- https://www.mondialtelecom.com/
- https://developer.airtel.in/
- https://www.vodafone.in/business/developer