##### 

**APPENDIX 1**

**ONLINE MOBILE RECHARGE PORTAL**

## A PROJECT REPORT

### Submitted by

# VISHAL GV

# 

### in partial fulfillment for the award of the degree of

# BACHELOR OF TECHNOLOGY

**IN**

#### ARTIFICIAL INTELLIGENCE AND

#### DATA SCIENCE

## SRI KRISHNA COLLEGE OF ENGINEERING AND TECHNOLOGY

**(An Autonomous Institution, Affiliated to Anna University Chennai - 600 025)**

AUGUST / 2024



# BONAFIDE CERTIFICATE

Certified that this project report **“………ONLINE MOBILE RECHARGE…** **”**

is the bonafide work of “…………………………….VISHAL GV

#### who carried out the project work under my supervision.

##### SIGNATURE SIGNATURE

DR.VENKATA LAKSHMI SENTHIL KUMAR

##### HEAD OF THE DEPARTMENT SUPERVISOR

Submitted for the Project viva-voce examination held on

INTERNAL EXAMINER EXTERNAL EXAMINER

| **Chapter No.** | **Title** | **Page No.** | |
| --- | --- | --- | --- |
| **1** | **Introduction** | **i** | |
| 1.1 | Overview of the Online Mobile Recharge Portal | ii | |
| 1.2 | Objectives of the Project | iii | |
| 1.3 | Scope of the Project | iv | |
| 1.4 | Organization of the Report | V | |
| **2** | **Literature Survey** | **1** | |
| 2.1 | Existing Systems for Mobile Recharge | 2 | |
| 2.2 | Analysis of Current Solutions | 3 | |
| 2.3 | Identified Gaps and Challenges | 5 | |
| **3** | **System Design** | **7** | |
| 3.1 | System Architecture | 8 | |
| 3.2 | Module Descriptions | 10 | |
| 3.3 | Database Design | 13 | |
| 3.4 | User Interface Design | 15 | |
| **4** | **Implementation** | **18** | |
| 4.1 | Technology Stack Used | 19 | |
| 4.2 | Backend Development with Django | 21 | |
| 4.3 | Frontend Development with React | 23 | |
| 4.4 | Integration of frontend and backend | 25 | |
| 4.5 | Security Implementation | 27 | |
| **5** | Frontend code | **30** | |
| 5.1 | Backend code | 31 | |
| 5.2 | Frontend image | 33 | |
| 5.3 | Backend image | 36 | |
| **6** | **Results and Discussion** | **39** | |
| 6.1 | Evaluation of System Performance | 40 | |
| 6.2 | User Feedback and Analysis | 42 | |
|  |  |  | |
| **7** | **Conclusion and Future Work** | **47** | |
| 7.1 | Summary of Findings | 48 | |
| 7.2 | Recommendations for Future Enhancements | 50 | |
|  |  | |  |

**TABLE OF CONTENTS**

**Table 1.1** System Requirements for the Online Mobile Recharge Portal 2

**Table 2.1** Existing Systems for Handling Student Inquiries 3

**Table 2.2** Analysis of Current Solutions 5

**Table 3.1** System Architecture Overview 8

**Table 3.2** Module

**Table 3.3** Database Design Elements 13

**Table 3.4** User Interface Design Components 15

**Table 4.1** Technology Stack Used in Implementation 19

**Table 4.2** Backend Development with django 21

**Table 4.3** Frontend Development with React 23

**Table 4.4**Integration of Django 25

**Table 4.5** Security Measures Implemented in the Portal 27

**Table 6.1** Evaluation of System Performance 40

**Table 6.2** User Feedback and Analysis 42

**Table 6.3** Comparison with Existing Systems 44

**List of Abbreviations and Nomenclature**

**Abbreviations**

**API** - Application Programming Interface  
**DBMS** - Database Management System  
**UI** - User Interface  
**UX** - User Experience  
**JSON** - JavaScript Object Notation  
**HTTP** - Hypertext Transfer Protocol  
**SSL** - Secure Sockets Layer  
**CRUD** - Create, Read, Update, Delete  
**REST** - Representational State Transfer  
**MVC** - Model-View-Controller

**Nomenclature**

**Portal** - Student Inquiry Assistance Portal

**React** - JavaScript Library for Building User Interfaces

**DJANGO** – used for backend Database

**Frontend** - The Client-Side Part of the Application  
**Backend** - The Server-Side Part of the Application  
**Module** - A Functional Unit of the Software Application  
**Schema** - Structure of a Database or a Data Model  
**Inquiry** - A Question or Request Submitted by a Student  
**Administrator** - Person Managing the System and Handling Inquiries

ABSTRACT

In the digital age, the need for fast, secure, and convenient mobile recharge solutions has become increasingly important. This paper details the development of an Online Mobile Recharge System, a web-based platform designed to streamline the mobile recharge process for users across various networks. The system provides a seamless user experience, allowing users to select their mobile operator, browse through a variety of available recharge plans, and complete transactions with ease using multiple secure payment options, including credit/debit cards, mobile wallets, and online banking.

The frontend of the platform is built using React, offering a responsive and intuitive interface that is both user-friendly and visually appealing. The backend, developed with Django, ensures robust management of user data, secure transaction processing, and seamless integration with third-party services such as payment gateways and mobile operators. The system also features user authentication and authorization, enabling personalized experiences and ensuring the security of user information.

A key component of the platform is its ability to provide real-time updates on available recharge plans and offers, as well as a comprehensive transaction history for users to track their spending. To enhance user engagement, the system incorporates personalized notifications and reminders for upcoming recharge needs, ensuring users are always informed about the best available options. Additionally, the system is designed with scalability in mind, allowing for easy expansion to support additional mobile operators, international users, and various payment gateways. By integrating advanced encryption and data protection measures, the platform ensures the highest standards of security, safeguarding user data and financial transactions from potential threats.

The system also includes administrative tools that allow operators to manage plans, offers, and user accounts efficiently. By automating and digitizing the recharge process, the platform not only simplifies the user experience but also reduces operational costs for mobile operators. This online mobile recharge system aims to foster greater adoption of digital payment solutions, improve the overall mobile user experience, and support the growing demand for digital services in the telecommunications industry.

.

INTRODUCTION

The proliferation of mobile technology has significantly altered the way people communicate, access information, and perform daily tasks. Among these, the need for regular mobile recharges to maintain connectivity is a common requirement for millions of users globally. Traditionally, mobile recharges were performed through physical outlets or scratch cards, a process that could be time-consuming, inconvenient, and limited by location and availability. However, with the advancement of digital technology, online mobile recharge systems have emerged as a transformative solution, offering users a convenient, fast, and secure method to top up their mobile phones from anywhere, at any time.

An online mobile recharge system is a web-based platform that allows users to recharge their mobile phones by selecting their service provider, choosing from a variety of available plans, and completing the transaction through digital payment methods. This system eliminates the need for physical vouchers and enables instant recharges, thereby enhancing user convenience. Additionally, these platforms often provide additional features such as discounts, promotional offers, and the ability to track recharge history, making them an attractive alternative to traditional methods.

The rise of online mobile recharge platforms aligns with the broader trend of digital transformation in the telecommunications industry, where the emphasis is on improving user experience, operational efficiency, and accessibility. As mobile phone penetration continues to increase, particularly in developing regions, the demand for efficient and user-friendly recharge solutions is also expected to grow.

The development of the Online Recharge Portal is a collaborative effort by **Atimanyu , Saran,Vishal** and **Vishnuram**. Their combined expertise in frontend and backend technologies has resulted in a comprehensive solution that meets the needs of both students and administrators. By streamlining the process of selecting, purchasing, and managing mobile recharges, this platform fosters a more convenient and reliable user experience.

**Overview of the Online Mobile Recharge Portal**

The Online Mobile Recharge Portal is an advanced digital platform designed to streamline the process of mobile phone top-ups, offering a modern and efficient solution for users. This platform addresses the limitations of traditional recharge methods, providing a seamless and user-friendly experience for managing mobile recharges.

**Purpose and Goals**  
The primary goal of the portal is to simplify and enhance the way users recharge their mobile phones. By providing a centralized and accessible platform, the portal aims to improve user convenience, speed up transaction processing, and ensure a smooth and secure recharge experience. Key objectives include reducing the time and effort required for recharges, offering real-time updates on available plans, and providing a reliable alternative to physical recharge cards.

**Technological Framework**  
The portal utilizes React for its frontend and Django for its backend:

* **React**: This JavaScript library is employed to build a dynamic and interactive user interface. React’s component-based architecture enables the creation of responsive designs that enhance user experience and ease of navigation.
* **Django**: As the backend framework, Django manages server-side operations such as transaction processing, database management, and application logic. Its robust and secure design ensures efficient handling of recharge requests and the protection of user data.

**Key Features**

* **Recharge Functionality**: Users can effortlessly select their mobile operator, choose from a variety of recharge plans, and complete transactions through an intuitive interface.
* **Real-Time Updates**: The portal provides real-time information on available recharge plans, promotions, and the status of transactions, ensuring users have access to the latest details.
* **User Dashboard**: Users have access to a personal dashboard to manage their recharge history, view past transactions, and track current activities, enhancing transparency and control.
* **Administrative Tools**: Administrators can manage and monitor the portal’s operations, including overseeing transactions, managing recharge plans, and handling user support.

**Advantages**

* **Increased Efficiency**: Automates and simplifies the recharge process, reducing the need for physical cards and streamlining transactions.
* **Enhanced Convenience**: Offers users a 24/7 accessible platform for recharges, allowing them to perform transactions from any location at any time.
* **Improved Security**: Ensures secure transactions and protection of user data through advanced encryption and compliance with data protection standards.
* **Real-Time Information**: Provides up-to-date details on recharge plans and promotions, helping users make informed decisions quickly.

**Objectives of the Project**

**The Online Mobile Recharge Portal project is designed with several key objectives aimed at optimizing the mobile recharge process for users. The primary goals of the project are:**

1. **Streamline Recharge Management**  
   The portal aims to automate and simplify the mobile recharge process. By providing a centralized digital platform, it removes the need for physical recharge cards and manual processing, allowing users to quickly select plans and complete transactions more efficiently and accurately.
2. **Enhance User Experience**  
   A core objective is to improve the overall experience for users. The portal enables users to select their mobile operator, browse available plans, and complete recharges online. It also provides real-time updates on transaction status and available offers, ensuring users are always informed and satisfied with their recharge experience.
3. **Improve Operational Efficiency**  
   The portal is designed to facilitate better management of recharge activities. It includes features such as real-time plan updates, transaction tracking, and administrative tools for managing user accounts and transactions. This centralization helps streamline operations and allows for more effective management of recharge processes.
4. **Ensure Transaction Security**  
   A critical objective is to protect user data and financial transactions. The portal incorporates advanced security measures, including encryption and secure payment gateways, to safeguard sensitive information from unauthorized access and breaches. Ensuring robust security is vital for maintaining user trust and compliance with data protection standards.
5. **Provide Actionable Insights**  
   The portal includes reporting tools that offer insights into transaction trends, popular recharge plans, and user behavior. These insights help in analyzing performance, identifying user preferences, and making data-driven decisions to enhance service offerings and operational efficiency.
6. **Enhance Accessibility and Convenience**  
   The project focuses on creating an accessible and user-friendly interface for all users. By leveraging modern technologies like React for the frontend, the portal ensures a responsive and intuitive design that enhances ease of use. The goal is to make the recharge process as smooth and convenient as possible, leading to increased user satisfaction.

**Scope of the Project**

* **The Online Mobile Recharge Portal project encompasses a range of features and functionalities designed to optimize and streamline the mobile recharge process for users. The scope of the project is outlined as follows:**
* **Platform Development**  
  The project involves the creation of a digital platform accessible via web browsers. The platform will be developed using React for the frontend, ensuring a dynamic and responsive user interface, and Django for the backend, managing server-side logic and database operations. The scope includes the comprehensive design, development, and deployment of the entire system to provide a seamless recharge experience.
* **Recharge Functionality and Management**  
  Users will be able to select their mobile operator, choose from a variety of recharge plans, and complete transactions through the portal. Key features will include:
* **Recharge Selection**: Users can browse and choose from various recharge plans and promotions tailored to their mobile network.
* **Real-Time Processing**: Transactions will be processed instantly, ensuring immediate application of recharge credits to the users' mobile accounts.
* **Transaction Tracking**: Users will be able to track the status of their recharge transactions in real-time and receive notifications and updates on their transaction progress.
* **Administrative Management**  
  The project will provide a comprehensive administrative dashboard for managing the portal's operations. Features will include:
* **Plan Management**: Tools for adding, updating, and removing recharge plans and promotions.
* **User Management**: Capabilities for managing user accounts, handling customer support requests, and resolving issues.
* **Transaction Monitoring**: Monitoring and reviewing transaction histories, statuses, and user activity.
* **Reporting**: Generating reports on recharge volumes, popular plans, transaction trends, and user behavior to aid in decision-making and service improvements.
* **Security and Data Privacy**  
  Ensuring the security and privacy of user data is a crucial aspect of the project. The system will incorporate:
* **User Authentication**: Secure login mechanisms for users and administrators to ensure only authorized access.
* **Data Encryption**: Protection of sensitive information, such as payment details, during transmission and storage.
* **Access Control**: Restricting access to sensitive data and functionalities based on user roles and permissions to maintain data integrity and confidentiality.
* **User Interface and Experience**  
  The scope includes designing a user-friendly interface for both end-users and administrators.

**Organization of the Report**

**The report on the Online Mobile Recharge Portal is structured to provide a thorough understanding of the project, its development process, and its outcomes. The organization of the report is as follows:**

1. **Introduction**  
   This section offers an overview of the Online Mobile Recharge Portal, including its purpose, objectives, and significance. It introduces the challenges addressed by the project and outlines the scope of the report, detailing how the portal aims to streamline and enhance the mobile recharge process for users.
2. **Literature Survey**  
   The literature survey examines existing solutions and systems related to mobile recharge services. It includes:

* **Existing Systems**: An exploration of current mobile recharge platforms and their functionalities.
* **Analysis**: A critical assessment of these systems, identifying their strengths, weaknesses, and areas for improvement.
* **Identified Gaps**: Insights into the shortcomings and challenges that the Online Mobile Recharge Portal aims to address, providing context for its development.

1. **System Design**  
   This section details the design aspects of the portal, covering:

* **System Architecture**: An overview of the technical architecture, illustrating the interaction between the frontend and backend components.
* **Module Descriptions**: Detailed descriptions of the various modules within the system, such as recharge functionality, user management, and administrative tools.
* **Database Design**: The schema and structure of the database used to store and manage recharge data, including user information and transaction records.
* **User Interface Design**: The design principles and mockups for the user interface, focusing on creating an intuitive and engaging user experience.

1. **Implementation**  
   The implementation section describes the development process of the portal:

* **Technology Stack Used**: An overview of the technologies and tools employed in the project, including React for frontend development and Django for backend development.
* **Backend Development**: Details of the backend development process using Django, including server-side logic and database management.
* **Frontend Development**: Insights into the frontend development process using React, including the creation of a responsive and dynamic user interface.

1. **Testing and Validation**  
   This section covers the testing process and validation of the portal:

* **Testing Methodologies**: The approaches and techniques used to test the system, ensuring its functionality, performance, and security.
* **Test Cases and Results**: Specific test cases, their execution, and the results obtained, highlighting any issues found and their resolution.

**2. Literature Survey**

The literature survey for the Student Inquiry Assistance Portal examines existing systems and solutions for managing student inquiries, analyzing their effectiveness, and identifying gaps and challenges that the proposed portal aims to address.

**1. Existing Systems for Handling Student Inquiries**

Current systems used by educational institutions for managing student inquiries include traditional manual processes, as well as automated systems. Traditional methods often involve paper-based forms and in-person visits to administrative offices, which can be inefficient and time-consuming.

Automated systems, on the other hand, typically offer online submission forms and email-based communication. Examples include ticketing systems and inquiry management software that provide basic functionalities for tracking and responding to student requests. These systems often integrate with institutional databases to retrieve and update student information.

**2. Analysis of Current Solutions**

An analysis of these systems reveals several strengths and weaknesses. Automated systems can enhance efficiency by streamlining the inquiry process and providing centralized tracking. However, they often suffer from limitations such as:

* **Lack of Integration**: Many systems do not integrate seamlessly with other institutional software, leading to fragmented data and workflow inefficiencies.
* **Limited User Interface**: The user experience can be cumbersome, with non-intuitive interfaces that may discourage students from using the system.
* **Inadequate Reporting**: Reporting features are often basic, lacking advanced analytics that could help institutions better understand inquiry trends and improve their services.

**3. Identified Gaps and Challenges**

The survey identifies several gaps and challenges in current solutions:

* **Scalability Issues**: Many systems struggle to scale effectively with growing numbers of inquiries, leading to performance bottlenecks.
* **Data Security Concerns**: Ensuring the security of sensitive student information is a challenge, with some systems lacking robust security measures.
* **User Experience**: Current systems often fail to provide an optimal user experience, with complex navigation and limited accessibility options.

**Conclusion**

The literature survey highlights the need for a more efficient, integrated, and user-friendly solution for managing student inquiries. The Student Inquiry Assistance Portal aims to address these gaps by offering a modern, scalable platform with advanced features for improved communication, data security, and overall user experience.

**2.1 Existing Systems for Mobile Recharge**

In the online mobile recharge sector, various systems and platforms have been employed to facilitate recharge transactions. These can generally be categorized into traditional methods and modern automated solutions.

**1.1 Traditional Methods**

Traditional methods of mobile recharge include:

* **Physical Recharge Cards**: Users purchase physical recharge cards from retail outlets. These cards come with unique codes that users enter manually to top up their mobile accounts. While this method is straightforward, it is often inconvenient due to the need to visit stores and manually input codes.
* **In-Person Recharges**: Some service providers offer the option to recharge mobile accounts at physical locations such as kiosks or telecom stores. Although this provides immediate recharge, it is time-consuming and less accessible compared to online methods.

**1.2 Automated Systems**

To enhance efficiency and convenience, many platforms have adopted automated online recharge systems. These systems typically include:

* **Online Recharge Platforms**: Websites and mobile apps that allow users to recharge their mobile accounts directly through an online interface. Users can select their telecom operator, choose a plan, and complete transactions using various payment methods. These platforms offer convenience and immediate processing but vary in user experience and feature set.
* **Mobile Wallet Integration**: Recharge services integrated into mobile wallets and payment apps (e.g., Paytm, Google Pay). Users can perform recharges seamlessly from within their wallet apps, which often include features like saved payment methods and transaction history.
* **Subscription-Based Recharges**: Some systems offer subscription-based models where users can set up automatic recharges at regular intervals. This method simplifies the process for users who prefer not to manually initiate recharges each time.

**1.3 Strengths and Limitations**

**Strengths:**

* **Increased Convenience**: Automated systems provide users with the ability to recharge from anywhere at any time, significantly improving accessibility.
* **Instant Processing**: Online systems enable real-time processing of transactions, ensuring immediate application of recharge credits.

**Limitations:**

* **Integration Issues**: Some online recharge platforms may face challenges integrating with multiple telecom operators, leading to potential discrepancies or limitations in available plans.
* **Security Concerns**: Ensuring secure transactions is crucial, as online systems are vulnerable to fraud and data breaches. Implementing robust security measures is essential to protect user information and financial data.

**2.2 Analysis of Current Solutions**

Current solutions for managing mobile recharges range from traditional methods like physical recharge vouchers to modern digital platforms and mobile apps. Each approach has its own strengths and limitations, impacting the overall efficiency and user experience of the recharge process.

1. **Physical Recharge Vouchers**
   * **Strengths:**
     + **Accessibility:** Available at numerous retail locations, making them easy to purchase for users without internet access.
     + **Simplicity:** Users can recharge by simply entering a code, making it a straightforward process for those who prefer offline methods.
   * **Limitations:**
     + **Inconvenience:** Requires users to physically visit a store, which can be time-consuming and inconvenient.
     + **Limited Options:** Offers fewer customization options, such as choosing specific plans or applying discounts, compared to digital platforms.
2. **Telecom Provider Websites**
   * **Strengths:**
     + **Direct Access:** Provides a direct link to the service provider, ensuring that users are dealing with a trusted source.
     + **Plan Selection:** Offers a variety of recharge plans and packages tailored to different user needs.
   * **Limitations:**
     + **User Experience:** The quality of user experience can vary, with some websites being difficult to navigate or not optimized for mobile devices.
     + **Limited Integration:** These websites often lack integration with other payment platforms or third-party services, limiting the flexibility of payment options.
3. **Mobile Recharge Apps**
   * **Strengths:**
     + **Convenience:** Allows users to recharge their mobiles anytime and anywhere with just a few taps on their smartphone.
     + **Multiple Payment Options:** Supports various payment methods, including credit/debit cards, digital wallets, and UPI, providing flexibility.
   * **Limitations:**
     + **User Interface Complexity:** The multitude of features can make the interface cluttered and challenging to navigate for users who only want to perform a simple recharge.

**Conclusion**

Current solutions for mobile recharges offer a range of features to meet user needs, from convenience and accessibility to comprehensive integration with other services. However, these solutions are often constrained by limitations such as user experience, dependency on internet connectivity, and security concerns.

**2.3 Identified Gaps and Challenges**

Current solutions for managing online mobile recharges encompass a range of approaches, from traditional methods to modern automated systems. Each approach has its strengths and limitations, impacting its overall effectiveness in facilitating seamless recharge transactions.

**1. Online Recharge Platforms**

**Strengths:**

* **Convenience**: Allows users to recharge their mobile accounts from anywhere at any time, eliminating the need to visit physical stores or use physical cards.
* **Immediate Processing**: Transactions are processed in real-time, ensuring that recharge credits are applied instantly to the user’s mobile account.

**Limitations:**

* **User Experience**: The effectiveness of online platforms can vary based on their design and functionality. Poorly designed interfaces can lead to user frustration and errors, affecting the overall satisfaction.
* **Integration**: Some online recharge systems may operate independently from other financial or telecom systems, leading to potential issues with data synchronization and management.

**2. Mobile Wallet Integration**

**Strengths:**

* **Seamless Transactions**: Integrates recharge functionality within mobile wallet apps, allowing users to perform recharges easily using saved payment methods.
* **Enhanced Accessibility**: Users can perform recharges quickly without needing to navigate away from their wallet applications, streamlining the process.

**Limitations:**

* **Limited Plan Options**: Integration with mobile wallets may restrict the range of available recharge plans or promotions, depending on partnerships with telecom operators.
* **Security Risks**: Ensuring secure transactions is crucial, as mobile wallet integrations are vulnerable to potential fraud and data breaches.

**3. Subscription-Based Recharges**

**Strengths:**

* **Automatic Top-Ups**: Provides the convenience of automatic recharges at regular intervals, reducing the need for manual intervention by users.
* **User Retention**: Encourages user retention through recurring transactions, which can be beneficial for both users and service providers.

**Limitations:**

* **Flexibility**: Users may have limited flexibility in changing or canceling their subscription plans, which can be a drawback if their needs change.
* **Billing Errors**: There is a potential for billing errors or disputes if the subscription system is not accurately maintained or if users encounter issues with their accounts.

**Conclusion**

Current solutions for online mobile recharges offer various features to enhance the recharge experience but often face challenges such as integration issues, user experience limitations, and security concerns.

3.SYSTEM DESIGN

The System Design of the Online Mobile Recharge Portal outlines the architecture and key components that ensure an efficient, scalable, and user-friendly platform for handling mobile recharge transactions. This section covers the overall system architecture, module descriptions, database design, and user interface design.

**1. System Architecture**

The architecture of the Online Mobile Recharge Portal is designed to provide a robust and scalable solution. It consists of three main layers:

* **Frontend**: Built using React, the frontend offers a responsive and interactive user interface. It communicates with the backend via RESTful APIs to handle recharge requests, retrieve balance information, and update the user interface in real-time. The frontend ensures a seamless experience for users, providing easy navigation and intuitive interactions.
* **Backend**: Powered by Django, the backend handles business logic, transaction processing, and communication with the database. It provides API endpoints for the frontend to perform operations such as processing payments, managing recharge plans, and handling user authentication. The backend also ensures secure handling of sensitive payment information and transaction records.
* **Database**: PostgreSQL is used as the database to store and manage user profiles, recharge transactions, and other relevant information. Its relational design supports structured data and ensures consistency and reliability in transaction processing.

**2. Module Descriptions**

The portal is organized into several key modules:

* **Recharge Management**: This module handles the submission and processing of recharge requests. It allows users to select their telecom operator, choose recharge plans, and complete transactions. The module also tracks transaction status and updates users on successful or failed recharge attempts.
* **Administrative Dashboard**: Provides administrative staff with tools to manage recharge transactions, including monitoring transaction volumes, resolving issues, and generating reports. It includes functionalities for managing telecom operator partnerships and reviewing system performance metrics.
* **User Authentication**: Ensures secure access to the portal with features for user registration, login, and role-based access control. It protects user accounts and payment information through secure authentication mechanisms and data encryption.
* **Notification System**: Sends notifications to users regarding transaction status, promotional offers, and system updates. It enhances user engagement by keeping customers informed about their recharge activities and available offers.

**3. Database Design**

The database design for the portal includes:

* **Tables**: PostgreSQL tables are used to store various types of data, including user profiles, recharge transactions, telecom operator details, and promotional offers. Each table is designed to handle specific data types and relationships.

**3.1 System Architecture**

**Overview**

The System Architecture of the Student Inquiry Assistance Portal is designed to ensure a robust, scalable, and efficient system. It is structured into three main layers: the frontend, backend, and database. This multi-layered architecture promotes separation of concerns, making the system easier to manage and scale.

**1. Frontend**

* **Technology**: Built using **React**, a popular JavaScript library for building user interfaces.
* **Responsibilities**:
  + **User Interaction**: Handles all user interactions, including form submissions, data visualization, and real-time updates.
  + **API Integration**: Communicates with the backend through RESTful APIs to fetch and send data.
  + **Responsiveness**: Ensures the portal is accessible and functional across various devices and screen sizes.

**2. Backend**

* **Technology**: Developed using **Django**, a high-level Python web framework.
* **Responsibilities**:
  + **Business Logic**: Manages the core functionality of the portal, including inquiry processing and user management.
  + **API Services**: Provides RESTful endpoints for the frontend to interact with the backend.
  + **Authentication**: Handles user authentication and authorization, ensuring secure access to the portal.

**3. Database**

* **Technology**: Utilizes **MongoDB**, a NoSQL database that supports flexible schema designs.
* **Responsibilities**:
  + **Data Storage**: Stores all data related to inquiries, users, and system logs.
  + **Scalability**: Provides a scalable solution to handle increasing volumes of data and users.
  + **Query Performance**: Optimized for fast data retrieval and efficient query processing.

**Conclusion**

The architecture ensures that each component works harmoniously, providing a seamless experience for users while maintaining system performance and scalability.

**3.2 Module Descriptions**

**1. Inquiry Management**

* **Functionality**: Manages the entire lifecycle of student inquiries, from submission to resolution.
* **Features**:
  + **Submission**: Allows students to submit inquiries via a web form.
  + **Categorization**: Automatically categorizes inquiries based on predefined criteria.
  + **Tracking**: Provides real-time tracking of inquiry status and history.
  + **Resolution**: Facilitates the resolution process, including updates and feedback.

**2. Administrative Dashboard**

* **Functionality**: Provides tools for administrative staff to manage and oversee inquiries.
* **Features**:
  + **Inquiry Management**: Allows staff to view, sort, and prioritize inquiries.
  + **Reports**: Generates detailed reports on inquiry trends, response times, and staff performance.
  + **Monitoring**: Offers real-time insights into system activity and user interactions.

**3. User Authentication**

* **Functionality**: Manages user access and security.
* **Features**:
  + **Registration**: Allows new users to register and create accounts.
  + **Login**: Provides secure login mechanisms with password protection.
  + **Role-Based Access**: Controls access to different functionalities based on user roles (e.g., students, staff).

**4. Notification System**

* **Functionality**: Keeps users informed about the status of their inquiries and system updates.
* **Features**:
  + **Alerts**: Sends notifications for important updates, such as status changes and system announcements.
  + **Customization**: Allows users to customize notification preferences.

**Conclusion**

Each module is designed to address specific needs of the system, ensuring a comprehensive and user-centric approach to managing student inquiries.

**3.3 Database Design**

**1. Collections**

* **Users Collection**: Stores user profiles, including credentials, roles, and contact information.
* **Inquiries Collection**: Contains details of all submitted inquiries, including metadata, status, and resolution history.
* **Logs Collection**: Maintains system logs for monitoring and troubleshooting purposes.

**2. Schema**

* **Dynamic Fields**: The flexible schema allows for varying inquiry types and user attributes, accommodating future changes and additions.
* **Relationships**: Defines relationships between different data entities, such as linking inquiries to specific users or administrative actions.

**3. Indexes**

* **Performance Optimization**: Indexes are used to speed up query performance, particularly for frequently accessed data.
* **Types of Indexes**: Includes single-field indexes for simple queries and compound indexes for more complex search requirements.

**4. Security**

* **Data Protection**: Implements encryption and other security measures to protect sensitive information.
* **Access Control**: Ensures that only authorized users can access or modify specific data.

**Conclusion**

The database design is structured to efficiently manage data, support scalable operations, and ensure security and performance. Each module is designed to address specific needs of the system, ensuring a comprehensive and user-centric approach to managing student inquiries.

**3.4 User Interface Design**

**1. Design Principles**

* **User-Centric**: Focuses on creating an intuitive and engaging experience for users.
* **Simplicity**: Emphasizes clear navigation and minimalistic design to reduce user confusion.
* **Consistency**: Maintains a consistent look and feel across the portal to enhance usability.

**2. Layout**

* **Dashboard**: Provides an overview of key functionalities and quick access to common tasks.
* **Forms**: Designed with user input in mind, including validation and error handling to guide users through the submission process.
* **Views**: Offers different views for students and administrative staff, tailored to their specific needs and roles.

**3. Accessibility**

* **Compliance**: Adheres to accessibility standards such as WCAG (Web Content Accessibility Guidelines).
* **Features**: Includes keyboard navigation, screen reader support, and adjustable text sizes to accommodate various user needs.

**4. Responsiveness**

* **Adaptability**: Ensures that the portal is fully functional on a range of devices, from desktops to mobile phones.
* **Testing**: Regularly tested across different screen sizes and devices to ensure consistent performance.

**Conclusion**

The user interface design is crafted to deliver a seamless and inclusive experience, making it easy for users to navigate and interact with the portal.

**4. Implementation**

The implementation of the Student Inquiry Assistance Portal involves translating the design into a functional system using a combination of frontend and backend technologies. This section outlines the key aspects of the implementation, focusing on technology stack, backend and frontend development, integration, and security.

**1. Technology Stack Used**

The project leverages a modern technology stack to ensure performance and scalability:

* **Frontend**: **React** is used to build a dynamic and responsive user interface. It offers a component-based architecture, enhancing modularity and reuse.
* **Backend**: **Django**, a high-level Python web framework, handles server-side logic and API management. It provides robust features for rapid development and security.
* **Database**: **MongoDB**, a NoSQL database, manages data with flexibility, supporting diverse data structures and scaling as needed.

**2. Backend Development with Django**

The backend development focuses on creating a reliable and secure server-side application:

* **Architecture**: Utilizes Django's Model-View-Template (MVT) pattern for efficient code organization.
* **API Development**: Employs Django REST framework to build RESTful APIs, facilitating communication between the frontend and backend.
* **Database Management**: Uses Django’s ORM to interact with MongoDB, ensuring smooth data operations and schema management.
* **Security**: Implements authentication, authorization, and data protection measures to safeguard user information.

**3. Frontend Development with React**

The frontend is designed to provide an intuitive and engaging user experience:

* **Component-Based Design**: Develops reusable components for different parts of the UI, such as forms, dashboards, and notifications.
* **State Management**: Utilizes React’s built-in state management and Redux for handling complex state requirements.
* **Routing**: Manages navigation between different views and pages using React Router.
* **Performance Optimization**: Incorporates techniques like code splitting and lazy loading to enhance application performance.

**4. Integration of Frontend and Backend**

Integration ensures seamless communication between the user interface and server-side functionalities:

* **API Integration**: Connects React components with Django APIs to handle data submission, retrieval, and updates.
* **Authentication**: Utilizes JWT for secure user sessions and role-based access control.
* **Error Handling**: Implements mechanisms for handling and displaying errors, ensuring a smooth user experience.

**4.1 Technology Stack Used**

**Overview**

The Student Inquiry Assistance Portal is built using a modern technology stack that ensures performance, scalability, and maintainability. The selected technologies include React for the frontend, Django for the backend, and MongoDB for the database. This combination leverages the strengths of each technology to create a seamless and efficient system.

**1. Frontend Technology: React**

* **Purpose**: React is used to build a dynamic and responsive user interface. It provides a component-based architecture that allows for modular and reusable UI elements.
* **Features**: React’s virtual DOM enhances performance by minimizing direct manipulation of the actual DOM. Its rich ecosystem includes libraries like React Router for navigation and Redux for state management.
* **Advantages**: Offers a smooth user experience with fast updates and interactive features. The component-based approach facilitates easy maintenance and scalability.

**2. Backend Technology: Django**

* **Purpose**: Django is employed to handle the server-side logic, data processing, and API management. It is a high-level Python web framework that encourages rapid development and clean, pragmatic design.
* **Features**: Includes a robust ORM (Object-Relational Mapping) for database interactions, built-in admin interface for easy management, and a comprehensive authentication system.
* **Advantages**: Django’s “batteries-included” philosophy provides a wide range of built-in features, reducing the need for third-party libraries and ensuring rapid development and security.

**3. Database Technology:**

* **Purpose**: DJANGO, a NoSQL database, is chosen for its flexibility in handling unstructured data and its ability to scale horizontally.
* **Features**: Supports a schema-less design, allowing for dynamic and diverse data structures. Its document-oriented model is well-suited for managing inquiry data.
* **Advantages**: Provides high performance and scalability, making it suitable for handling large volumes of data and complex queries efficiently.

**4. Additional Tools and Libraries**

* **APIs**: RESTful APIs are used for communication between the frontend and backend, enabling data exchange and integration.
* **Authentication**: JWT (JSON Web Tokens) and OAuth2 are used to secure user authentication and authorization.

**Conclusion**

The chosen technology stack combines the strengths of React, Django, and MongoDB, creating a powerful and adaptable platform for managing student inquiries.

**4.2 Backend Development with Django**

**Overview**

The backend of the Student Inquiry Assistance Portal is developed using Django, a robust and versatile Python web framework. Django’s features streamline backend development, providing a solid foundation for handling business logic, data management, and user interactions.

**1. Django Framework Overview**

* **Architecture**: Follows the Model-View-Template (MVT) architecture pattern. The Model defines the data structure, the View handles the business logic, and the Template manages the presentation layer.
* **Advantages**: Django’s built-in features, such as an ORM, admin interface, and security mechanisms, significantly accelerate development and enhance security.

**2. API Development**

* **RESTful APIs**: Django REST framework (DRF) is used to build and manage APIs, facilitating communication between the frontend and backend.
* **Endpoints**: APIs are designed for handling inquiries, user management, and other critical functionalities. Examples include endpoints for creating, retrieving, updating, and deleting inquiries.
* **Serialization**: DRF serializers are used to convert complex data types, such as querysets, into JSON format for frontend consumption.

**3. Database Interaction**

* **ORM**: Django’s ORM simplifies database operations by allowing developers to interact with the database using Python code instead of raw SQL queries.
* **Models**: Define the structure of the database tables. Models for inquiries, users, and other entities are created to manage data effectively.
* **Migrations**: Django’s migration system tracks and applies changes to the database schema, ensuring consistency between the models and the actual database.

**4. Security**

* **Authentication**: Django provides built-in authentication features, including user registration, login, and password management.
* **Authorization**: Role-based access control is implemented to restrict access to different parts of the system based on user roles and permissions.
* **Data Protection**: Includes measures such as input validation, CSRF protection, and encryption to safeguard sensitive information.

**Conclusion**

Django’s comprehensive framework offers a streamlined approach to backend development, enabling efficient handling of business logic, data management, and security.

**4.3 Frontend Development with React**

**Overview**

The frontend of the Student Inquiry Assistance Portal is developed using React, a powerful JavaScript library for building user interfaces. React’s component-based architecture and state management capabilities are leveraged to create a dynamic and responsive user experience.

**1. React Framework Overview**

* **Component-Based Architecture**: React’s core concept is the use of reusable components. Each component represents a piece of the UI, encapsulating its own logic and rendering.
* **Virtual DOM**: React uses a virtual DOM to optimize performance by minimizing direct manipulation of the actual DOM. Changes are first applied to the virtual DOM and then efficiently updated in the real DOM.

**2. User Interface Design**

* **Components**: The UI is built using a set of React components, including forms for inquiry submission, dashboards for administrators, and notifications for users.
* **State Management**: React’s built-in state management is used for handling component-level state. For more complex state needs, libraries like Redux are employed to manage global state and synchronize data across components.
* **Routing**: React Router is used to manage navigation and routing within the application, allowing users to switch between different views and pages seamlessly.

**3. Integration with Backend**

* **API Calls**: Axios or Fetch API is used to make HTTP requests to the Django backend. These calls are used to fetch data, submit inquiries, and handle other interactions with the server.
* **Error Handling**: Proper error handling mechanisms are implemented to manage and display error messages, ensuring a smooth user experience even when issues occur.

**4. Performance Optimization**

* **Code Splitting**: Implements code splitting to load only the necessary components and reduce the initial load time of the application.
* **Lazy Loading**: Uses React’s lazy loading to defer the loading of non-essential components until they are needed, enhancing performance.

**Conclusion**

React’s flexible and efficient framework allows for the creation of a dynamic and responsive frontend, providing users with an engaging and intuitive experience.

**4.4 Integration of Frontend and Backend**

**Overview**

The integration of the frontend and backend of the Student Inquiry Assistance Portal involves connecting React with Django to enable seamless communication and data exchange. This integration is crucial for the overall functionality of the portal.

**1. API Integration**

* **RESTful APIs**: Django REST framework provides the necessary APIs for the React frontend to interact with the backend. These APIs handle requests for creating, retrieving, updating, and deleting data.
* **Endpoints**: Defined endpoints for different functionalities, such as submitting inquiries, fetching user information, and retrieving inquiry statuses.

**2. Data Flow**

* **Requests and Responses**: The frontend sends HTTP requests to the backend APIs using libraries like Axios or Fetch. The backend processes these requests, performs necessary operations, and sends back responses in JSON format.
* **State Management**: React’s state management is used to handle and display data received from the backend. This includes updating the UI based on the data returned by API calls.

**3. Authentication**

* **Token-Based Authentication**: Implements JWT (JSON Web Tokens) for secure communication between the frontend and backend. Tokens are used to authenticate users and authorize access to certain resources.
* **Session Management**: Manages user sessions and ensures that authenticated requests are properly handled by the backend.

**4. Error Handling**

* **Error Messages**: Handles errors gracefully by displaying appropriate messages to users and logging issues for further analysis. This includes handling network errors, server errors, and validation errors.

**5. Testing and Debugging**

* **Integration Testing**: Conducts tests to ensure that the frontend and backend work together as expected. This includes testing API endpoints, data flow, and user interactions.
* **Debugging Tools**: Uses tools like Postman for API testing and browser developer tools for debugging frontend issues.

**4.5 Security Implementation**

**Overview**

Security is a critical aspect of the Online Recharge Mobile portal , protecting sensitive data and ensuring secure user interactions. Various security measures are implemented to safeguard the system against potential threats.

**1. Authentication and Authorization**

* **User Authentication**: Utilizes Django’s built-in authentication system and JWT for secure user login and session management. Users are required to authenticate before accessing certain features.
* **Role-Based Access Control**: Implements role-based permissions to restrict access to specific functionalities based on user roles (e.g., students, administrative staff).

**2. Data Protection**

* **Encryption**: Sensitive data, such as passwords and personal information, is encrypted using industry-standard encryption algorithms. This ensures data confidentiality and protection.
* **HTTPS**: Enforces HTTPS to secure data transmission between the client and server. SSL/TLS certificates are used to encrypt communication and protect against eavesdropping.

**3. Input Validation and Sanitization**

* **Input Validation**: Validates user inputs to prevent malicious data from being processed. Ensures that all inputs meet the expected format and constraints.
* **Sanitization**: Sanitizes inputs to prevent attacks such as SQL injection, cross-site scripting (XSS), and cross-site request forgery (CSRF).

**4. Security Monitoring**

* **Logging**: Implements logging to monitor system activity and detect potential security incidents. Logs are reviewed regularly to identify and address security issues.
* \*\*Vulnerability Scanning

FRONTEND REACT CODE:

APP.JS

import React, { useState, useEffect } from 'react';

import { BrowserRouter as Router, Route, Routes, Navigate } from 'react-router-dom';

import NavHead from './NavHead/NavHead';

import Login from './Login/Login';

import Register from './Register/Register';

import Home from './Home/Home';

import Plans from './Plans/Plans';

import Service from './Service/Service';

import ContactUs from './ContactUs/ContactUs';

import Account from './Account/Account';

import Profile from './Profile/Profile';

import NavDown from './NavDown/NavDown';

import Recharge from './Recharge/recharge';

import AdminLogin from './Admin/AdminLogin';

import AdminDashboard from './Admin/AdminDashboard';

function App() {

const [isLoggedIn, setIsLoggedIn] = useState(false);

const [isAdminLoggedIn, setIsAdminLoggedIn] = useState(false);

useEffect(() => {

// Check local storage for login status

const user = localStorage.getItem('user');

setIsLoggedIn(user ? true : false);

const admin = localStorage.getItem('admin');

setIsAdminLoggedIn(admin ? true : false);

}, []);

return (

<Router>

<NavHead isLoggedIn={isLoggedIn} setIsLoggedIn={setIsLoggedIn} />

<Routes>

<Route path="/Login" element={!isLoggedIn ? <Login setIsLoggedIn={setIsLoggedIn} /> : <Navigate to="/Home" />} />

<Route path="/Register" element={!isLoggedIn ? <Register /> : <Navigate to="/Home" />} />

<Route path="/Home" element={<Home />} />

<Route path="/recharge" element={isLoggedIn ? <Recharge /> : <Navigate to="/Login" />} />

<Route path="/Plans" element={isLoggedIn ? <Plans /> : <Navigate to="/Login" />} />

<Route path="/Service" element={isLoggedIn ? <Service /> : <Navigate to="/Login" />} />

<Route path="/ContactUs" element={isLoggedIn ? <ContactUs /> : <Navigate to="/Login" />} />

<Route path="/Account" element={isLoggedIn ? <Account /> : <Navigate to="/Login" />} />

<Route path="/Profile" element={isLoggedIn ? <Profile /> : <Navigate to="/Login" />} />

<Route path="/" element={<Navigate to="/Home" />} />

<Route path="/admin/login" element={<AdminLogin setIsAdminLoggedIn={setIsAdminLoggedIn} />} />

<Route path="/admin/dashboard" element={isAdminLoggedIn ? <AdminDashboard /> : <Navigate to="/admin/login" />} />

</Routes>

<NavDown />

</Router>

);

}

export default App;

**LOGIN.JS**

import React, { useState } from 'react';

import axios from 'axios';

import { useNavigate } from 'react-router-dom';

import './Login.css';

const Login = ({ setIsLoggedIn }) => {

  const [mobileNo, setMobileNo] = useState('');

  const [password, setPassword] = useState('');

  const [error, setError] = useState('');

  const navigate = useNavigate();

  const handleLogin = async (event) => {

    event.preventDefault();

    if (mobileNo === '' || password === '') {

      setError('Please enter both mobile number and password.');

      return;

    }

    try {

      const response = await axios.post('http://localhost:8000/login/', {

        mobile: mobileNo,

        password: password

      });

      if (response.status === 200) {

        localStorage.setItem('token', response.data.token);

        setIsLoggedIn(true);

        navigate('/Home');

      } else {

        setError('Invalid mobile number or password.');

      }

    } catch (err) {

      console.error('Error logging in:', err);

      setError('An error occurred. Please try again.');

    }

  };

  return (

    <div className="login-container">

      <h2>Login</h2>

      <form onSubmit={handleLogin}>

        <div className="form-group">

          <label htmlFor="mobileNo">Mobile No:</label>

          <input

            type="text"

            id="mobileNo"

            value={mobileNo}

            onChange={(e) => setMobileNo(e.target.value)}

            required

          />

        </div>

        <div className="form-group">

          <label htmlFor="password">Password:</label>

          <input

            type="password"

            id="password"

            value={password}

            onChange={(e) => setPassword(e.target.value)}

            required

          />

        </div>

        {error && <p className="error">{error}</p>}

        <button type="submit">Login</button>

      </form>

    </div>

  );

};

export default Login;

**PROFILE PAGE .JS**

import React, { useEffect, useState } from 'react';

import './Profile.css';

const Profile = () => {

  const [recharges, setRecharges] = useState([]);

  useEffect(() => {

    const storedRecharges = JSON.parse(localStorage.getItem('recharges')) || [];

    setRecharges(storedRecharges);

  }, []);

  return (

    <div className="profile-container">

      <h1>Your Profile</h1>

      <div className="recharge-history">

        <h2>Recharge History</h2>

        {recharges.length > 0 ? (

          <table className="recharge-table">

            <thead>

              <tr>

                <th>Mobile Number</th>

                <th>Price</th>

                <th>Validity</th>

                <th>Data</th>

                <th>Calls</th>

                <th>Date</th>

              </tr>

            </thead>

            <tbody>

              {recharges.map((recharge, index) => (

                <tr key={index}>

                  <td>{recharge.mobileNumber}</td>

                  <td>₹{recharge.plan.price}</td>

                  <td>{recharge.plan.validity}</td>

                  <td>{recharge.plan.data}</td>

                  <td>{recharge.plan.calls}</td>

                  <td>{new Date(recharge.date).toLocaleString()}</td>

                </tr>

              ))}

            </tbody>

          </table>

        ) : (

          <p className="no-history">No recharge history found.</p>

        )}

      </div>

    </div>

  );

};

export default Profile;

**NAVBAR.JS**

import React from 'react';

import './NavHead.css';

import AccountCircleIcon from '@mui/icons-material/AccountCircle';

import { Link, NavLink } from 'react-router-dom';

import jio from '../Assests/jio.png';

const NavHead = ({ isLoggedIn, setIsLoggedIn }) => {

  const handleLogout = () => {

    localStorage.removeItem('user');

    setIsLoggedIn(false);

  };

  return (

    <nav>

      <Link to="/home" className='logo'>

        <img src={jio} alt="logo" className='toggle-icon' />

      </Link>

      <ul>

        <li>

          <NavLink to="/plans" className={({ isActive }) => (isActive ? 'active' : '')}>Plans</NavLink>

        </li>

        <li>

          <NavLink to="/service" className={({ isActive }) => (isActive ? 'active' : '')}>Service</NavLink>

        </li>

        <li>

          <NavLink to="/contactus" className={({ isActive }) => (isActive ? 'active' : '')}>Contact Us</NavLink>

        </li>

        <li className="dropdown">

          <NavLink to="/account" className={({ isActive }) => (isActive ? 'active' : '')}>

            <AccountCircleIcon />

          </NavLink>

          <div className="dropdown-content">

            {isLoggedIn ? (

              <>

                <NavLink to="/Profile" className={({ isActive }) => (isActive ? 'active' : '')}>Profile</NavLink>

                <NavLink to="/settings" className={({ isActive }) => (isActive ? 'active' : '')}>Settings</NavLink>

                <NavLink to="/login" className={({ isActive }) => (isActive ? 'active' : '')} onClick={handleLogout}>Logout</NavLink>

              </>

            ) : (

              <>

                <NavLink to="/login" className={({ isActive }) => (isActive ? 'active' : '')}>Log In</NavLink>

                <NavLink to="/register" className={({ isActive }) => (isActive ? 'active' : '')}>Register</NavLink>

                <NavLink to="/admin/login" className={({ isActive }) => (isActive ? 'active' : '')}>Admin Login</NavLink>

              </>

            )}

          </div>

        </li>

      </ul>

    </nav>

  );

};

export default NavHead;

**PLAN DETAILS**

import React, { useState } from 'react';

import { useNavigate } from 'react-router-dom';

import Button from 'react-bootstrap/Button';

import Card from 'react-bootstrap/Card';

import CIcon from '@coreui/icons-react';

import { cibNetflix, cibAmazon, cibEpicGames } from '@coreui/icons';

import './Plans.css';

const Plans = () => {

  const [isPrepaid, setIsPrepaid] = useState(true);

  const navigate = useNavigate();

  const handleToggle = () => {

    setIsPrepaid(!isPrepaid);

  };

  const prepaidPlans = [

    { price: 375, validity: '28 days', data: '1GB/day', calls: 'Unlimited', icon: cibNetflix },

    { price: 456, validity: '56 days', data: '2GB/day', calls: 'Unlimited', icon: cibNetflix },

    { price: 674, validity: '84 days', data: '3GB/day', calls: 'Unlimited', icon: cibAmazon },

    { price: 3599, validity: '365 days', data: '2.5GB/day', calls: 'Unlimited', icon: cibNetflix },

    { price: 1029, validity: '84 days', data: '2GB/day', calls: 'Unlimited', icon: cibNetflix },

    { price: 674, validity: '84 days', data: '3GB/day', calls: 'Unlimited', icon: cibEpicGames },

  ];

  const postpaidPlans = [

    { price: 449, validity: 'Bill Cycle', data: '40GB/month', calls: 'Unlimited', icon: cibEpicGames },

    { price: 649, validity: 'Bill Cycle', data: '30GB/month', calls: 'Unlimited', icon: cibNetflix },

    { price: 749, validity: 'Bill Cycle', data: '50GB/month', calls: 'Unlimited', icon: cibEpicGames },

    { price: 1549, validity: 'Bill Cycle', data: '300GB/month', calls: 'Unlimited', icon: cibEpicGames },

    { price: 349, validity: 'Bill Cycle', data: '30GB/month', calls: 'Unlimited', icon: cibNetflix },

    { price: 849, validity: 'Bill Cycle', data: '75GB/month', calls: 'Unlimited', icon: cibEpicGames },

  ];

  const handleRecharge = (plan) => {

    navigate('/recharge', { state: { plan } });

  };

  const plans = isPrepaid ? prepaidPlans : postpaidPlans;

  return (

    <div className="plans-container">

      <h1 className="script-plan">Popular Mobile Plans</h1>

      <div className='toggle'>

        <label className="switch">

          <input type="checkbox" checked={isPrepaid} onChange={handleToggle} />

          <span className="slider"></span>

        </label>

        <span className="toggle-label">{isPrepaid ? "Prepaid" : "Postpaid"}</span>

      </div>

      <div className="card-container">

        {plans.map((plan, index) => (

          <Card className="plan-card" key={index}>

            <Card.Body>

              <Card.Text className='plan-card-text'>

                <strong className="highlighted-price">₹{plan.price}</strong><br />

                <strong>Validity:</strong> {plan.validity}<br />

                <CIcon icon={plan.icon} className="plan-icon" />

                <hr />

                <div className="plan-details">

                  <span><strong>Data:</strong> {plan.data}</span>

                  <span><strong>Calls:</strong> {plan.calls}</span>

                </div>

              </Card.Text>

              <Button variant="primary" className="plan-card-button" onClick={() => handleRecharge(plan)}>Recharge</Button>

            </Card.Body>

          </Card>

        ))}

      </div>

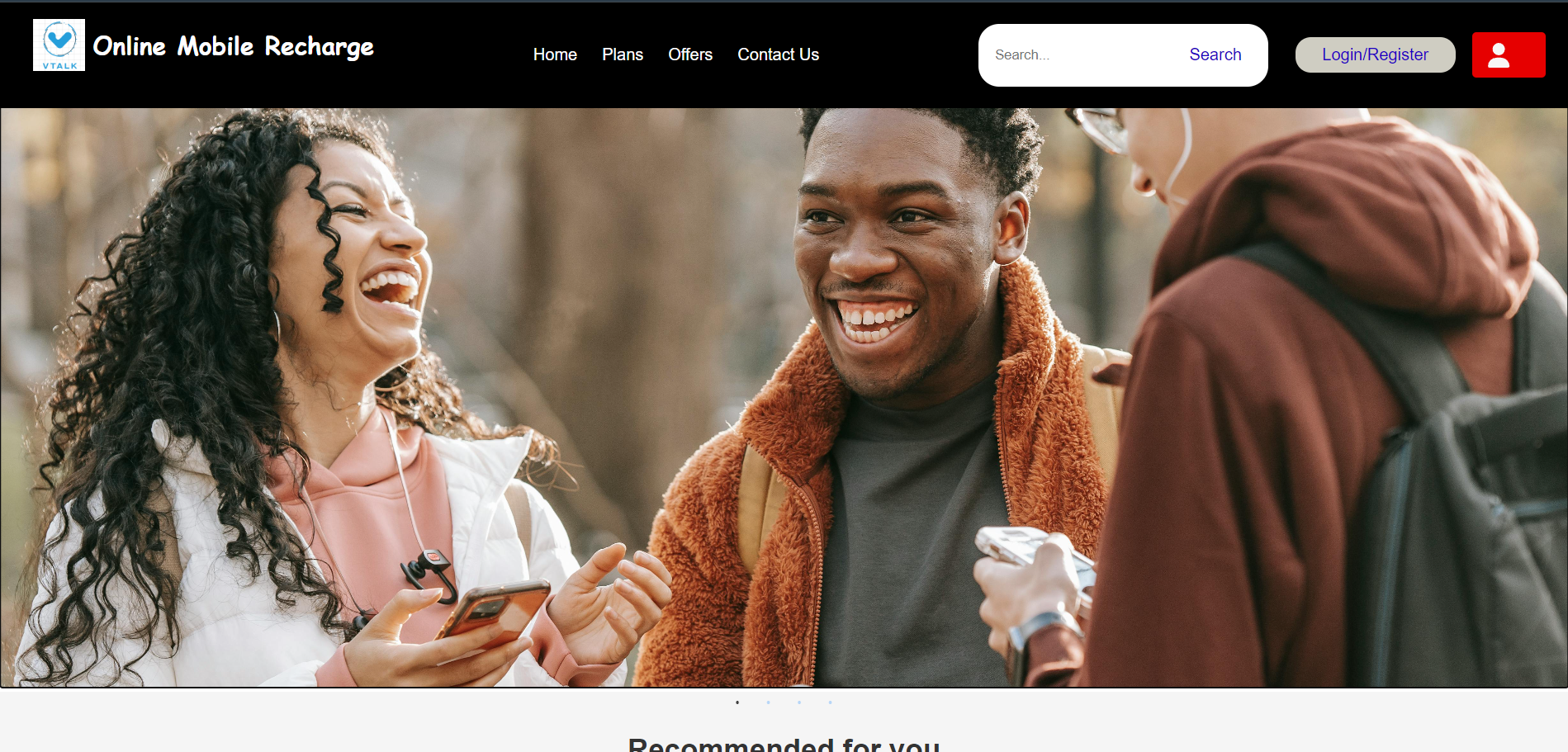
    </div>

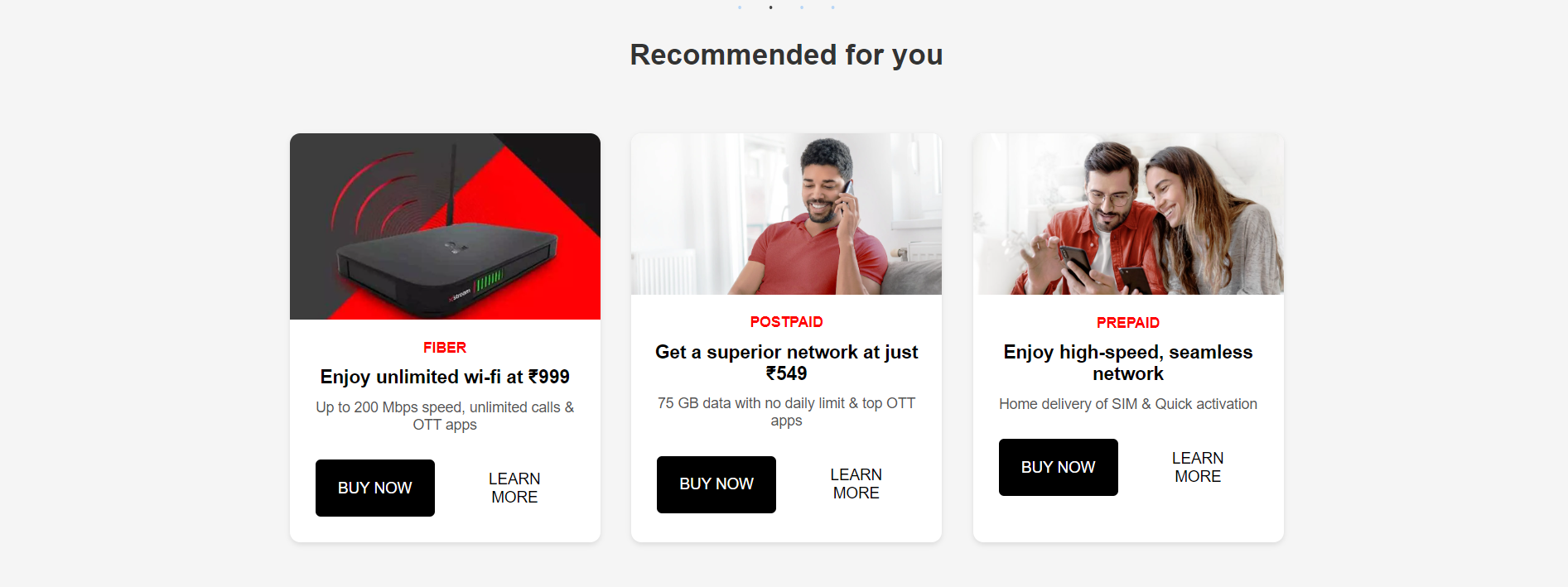
  );

};

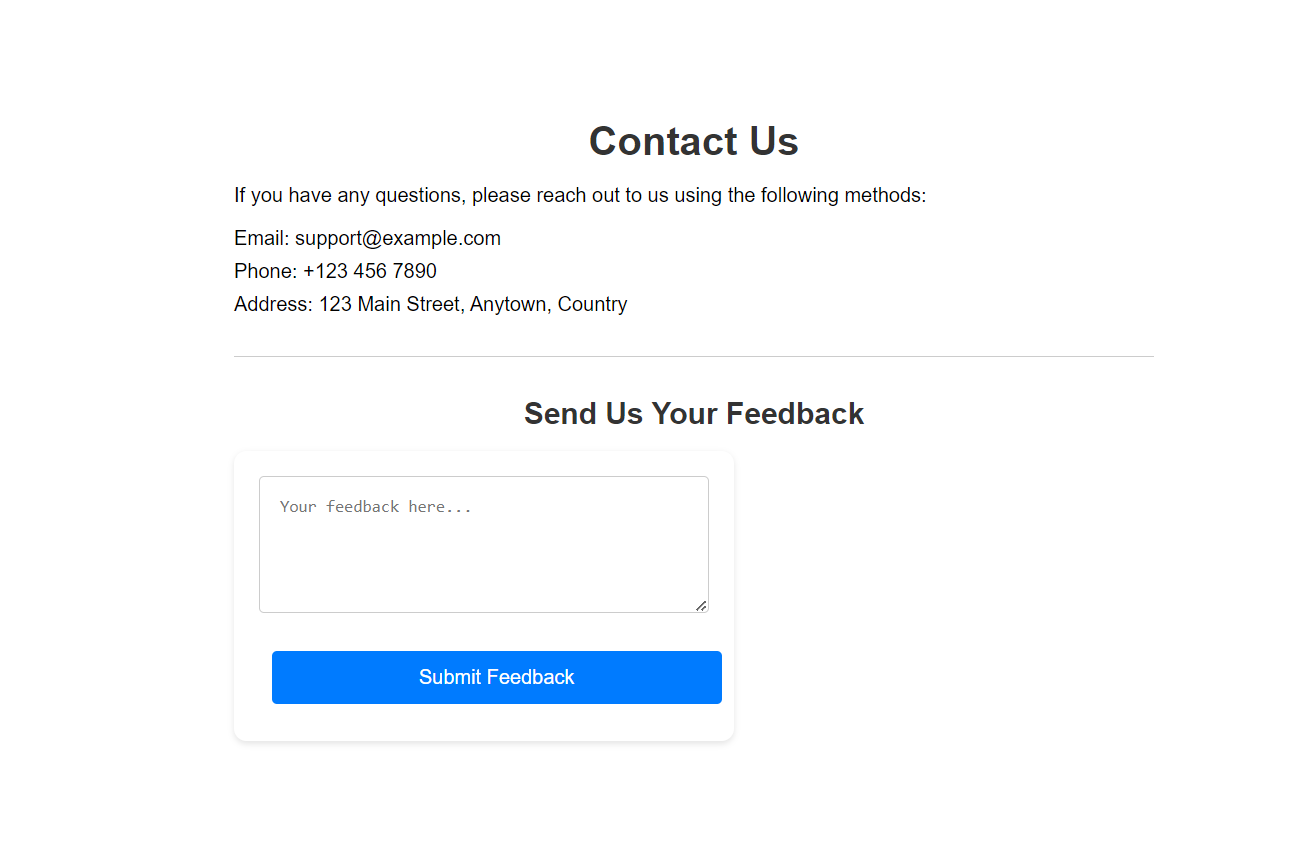
export default Plans;

FRONTEND IMAGE:

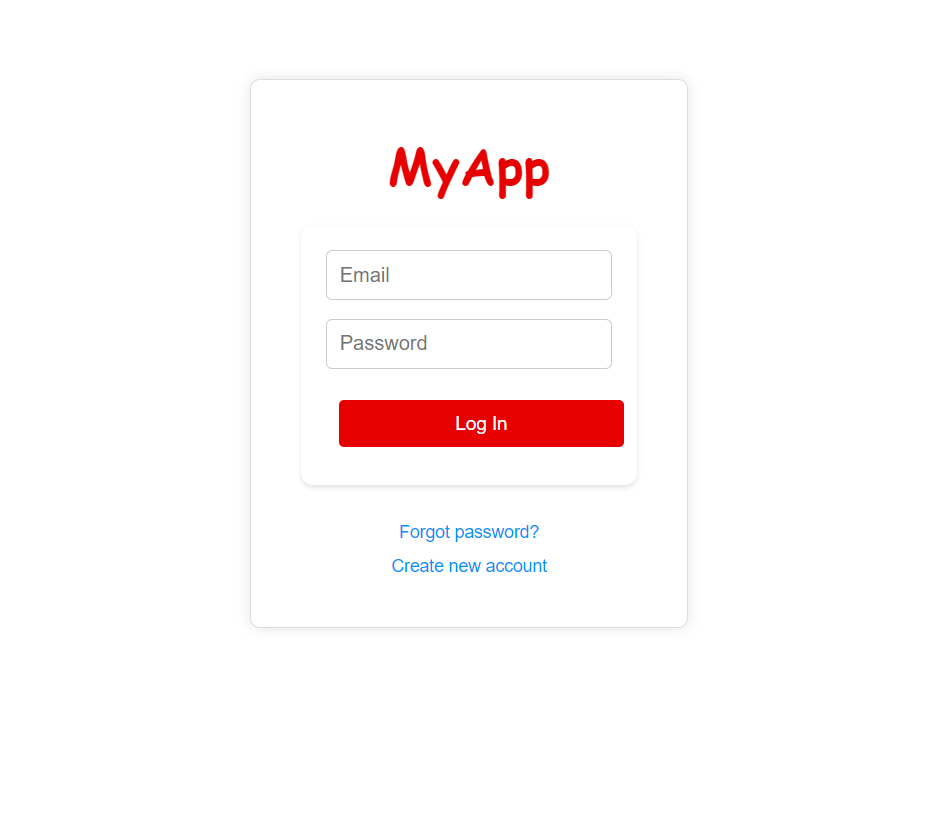




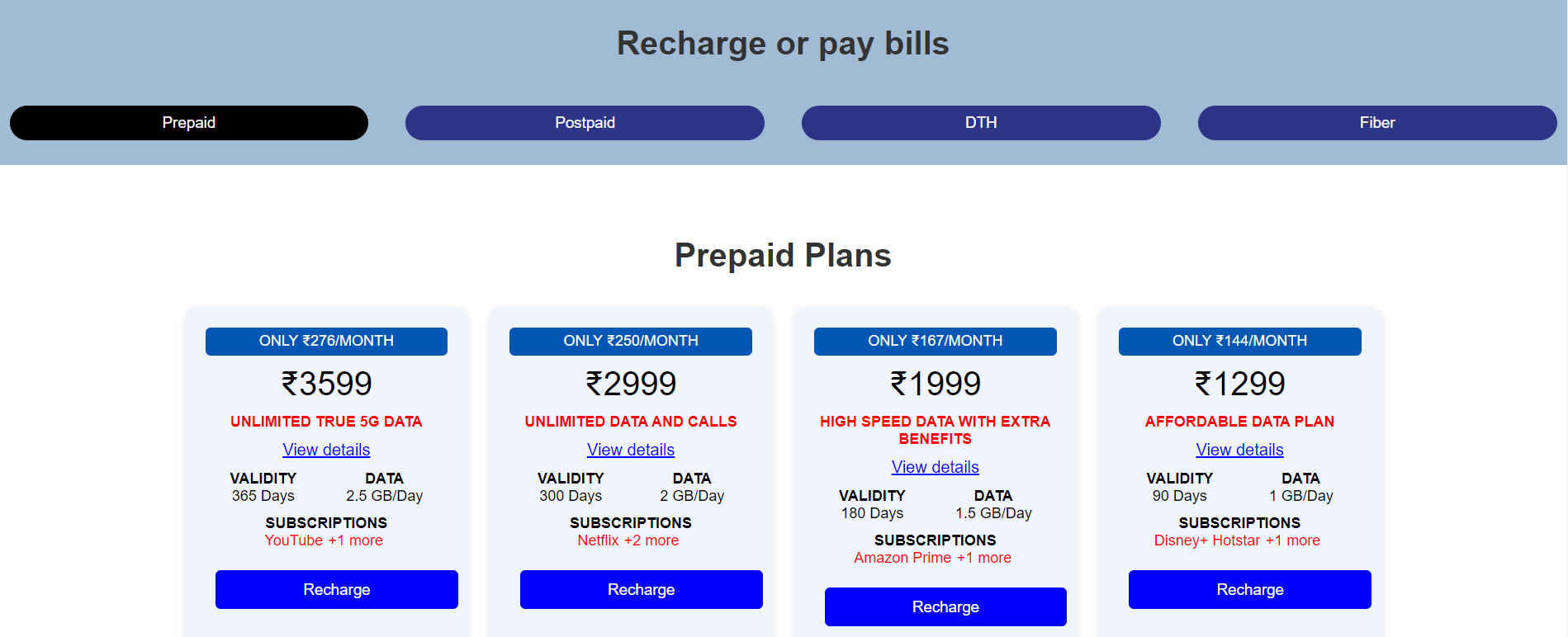
ENQUIRY FORM:



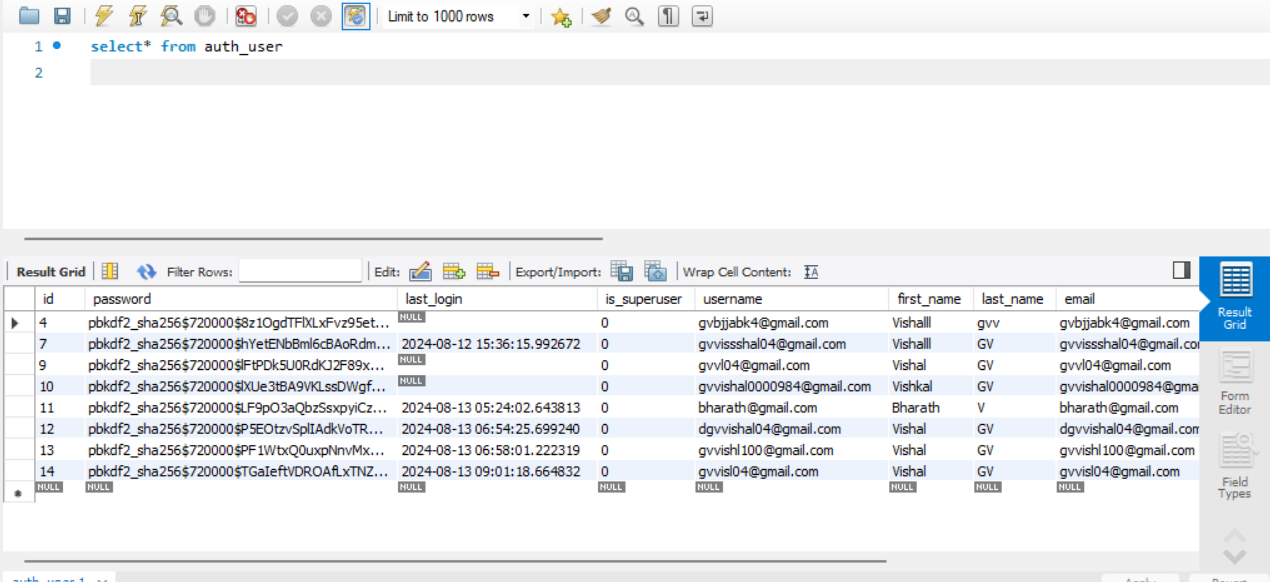
SIGN IN PAGE:

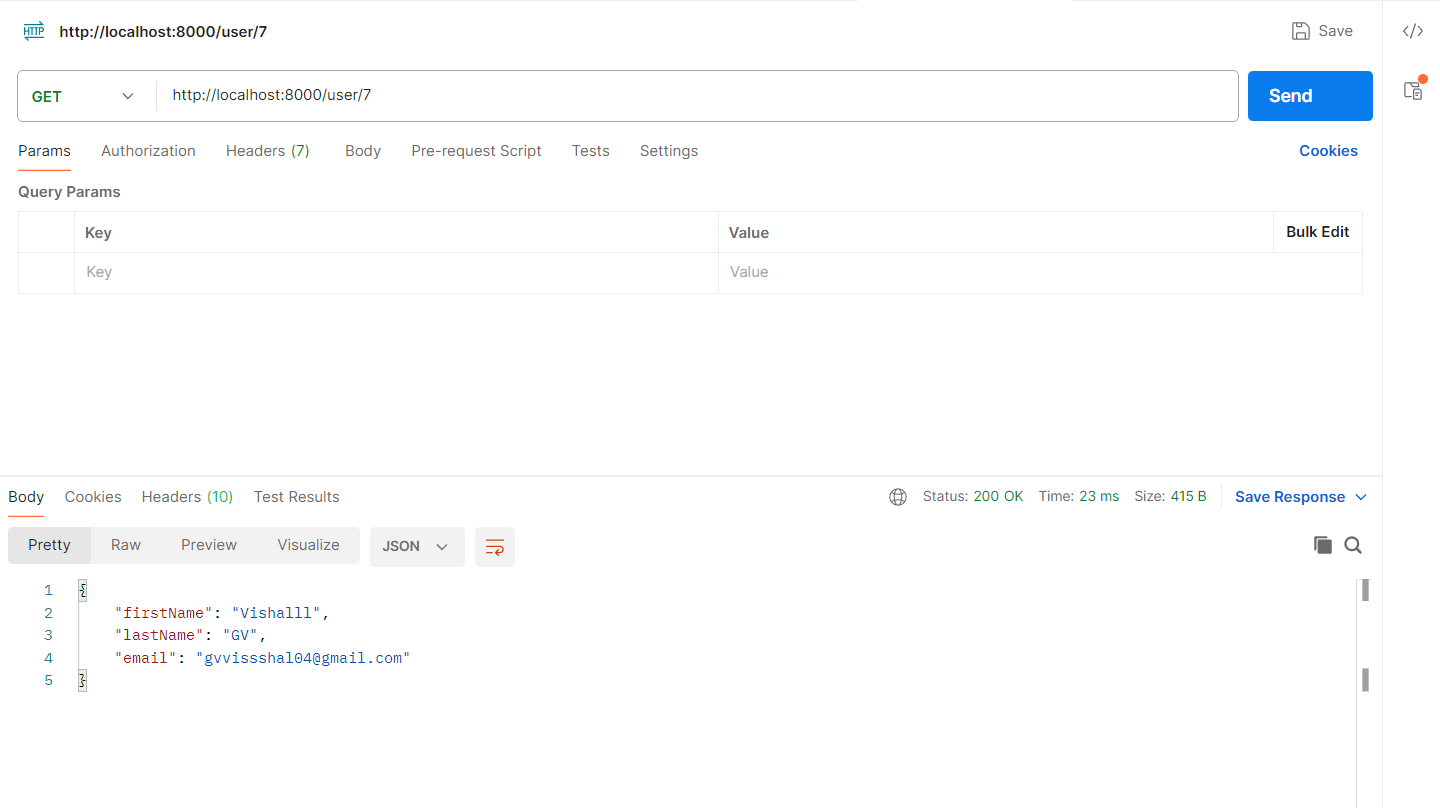


PLANS LIST:

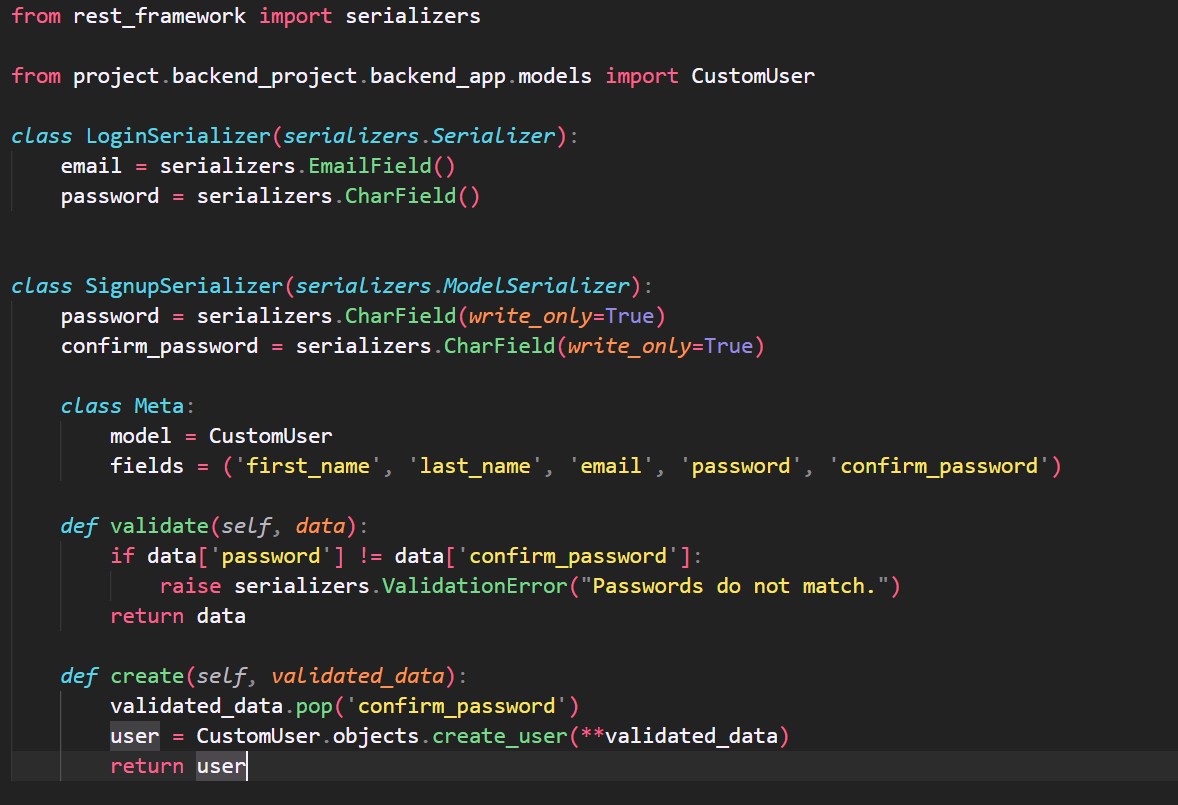
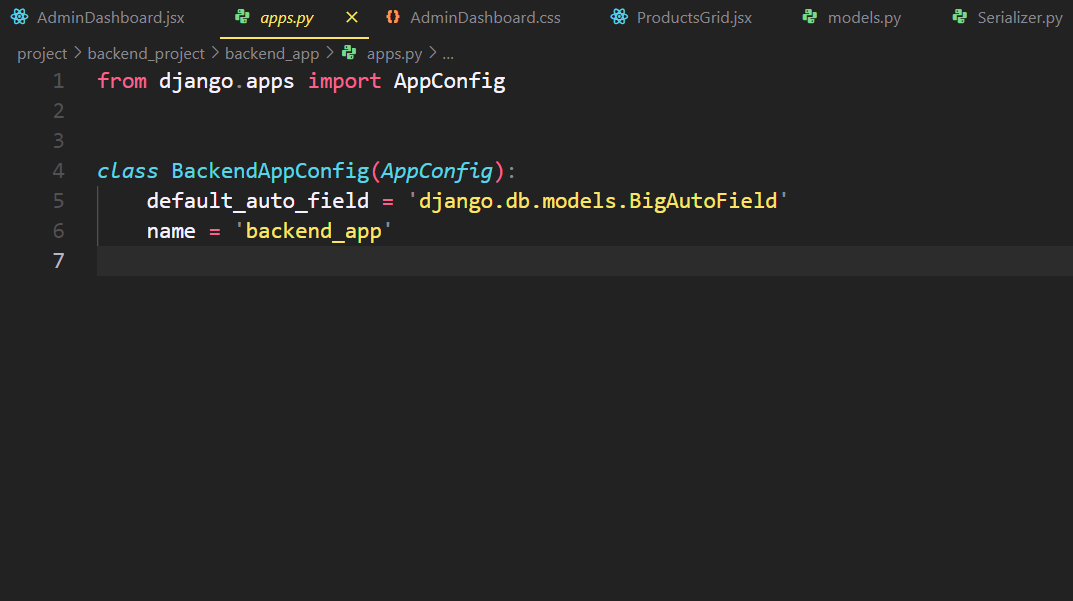


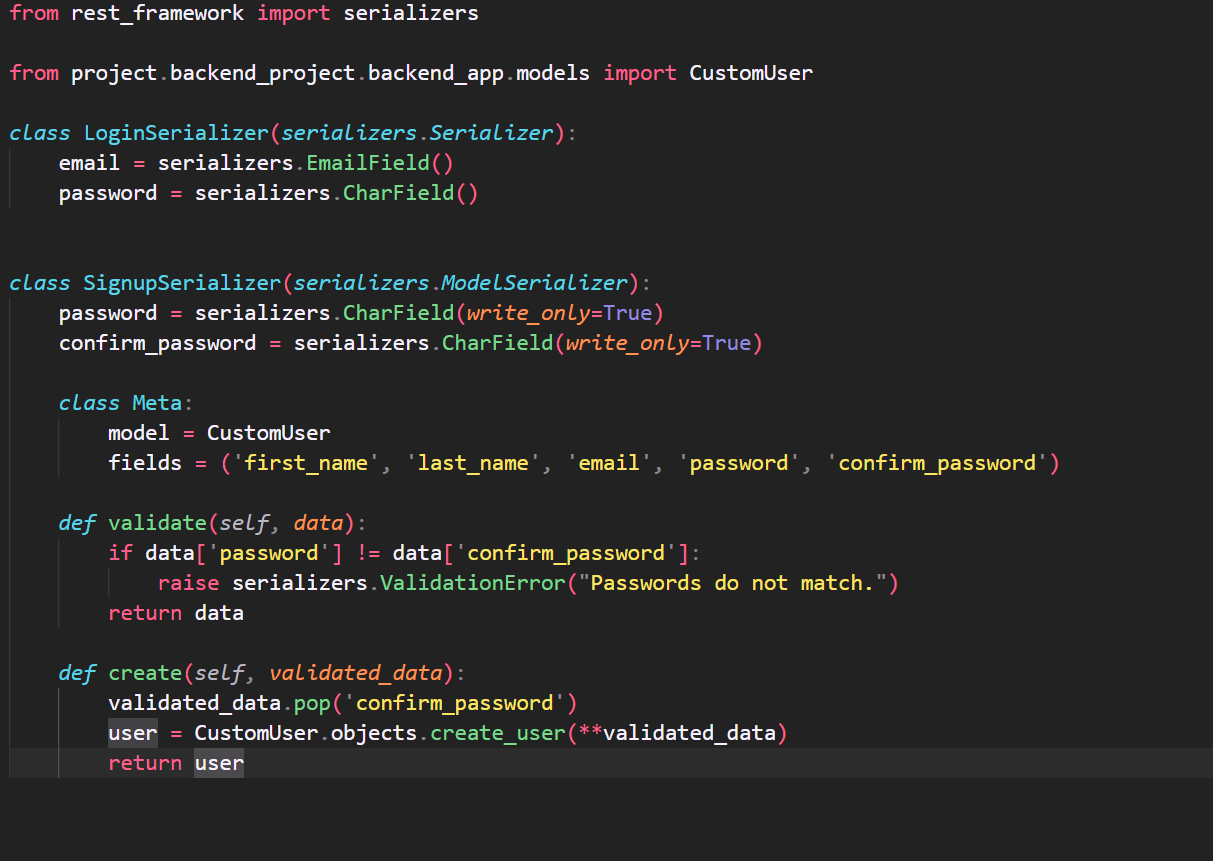
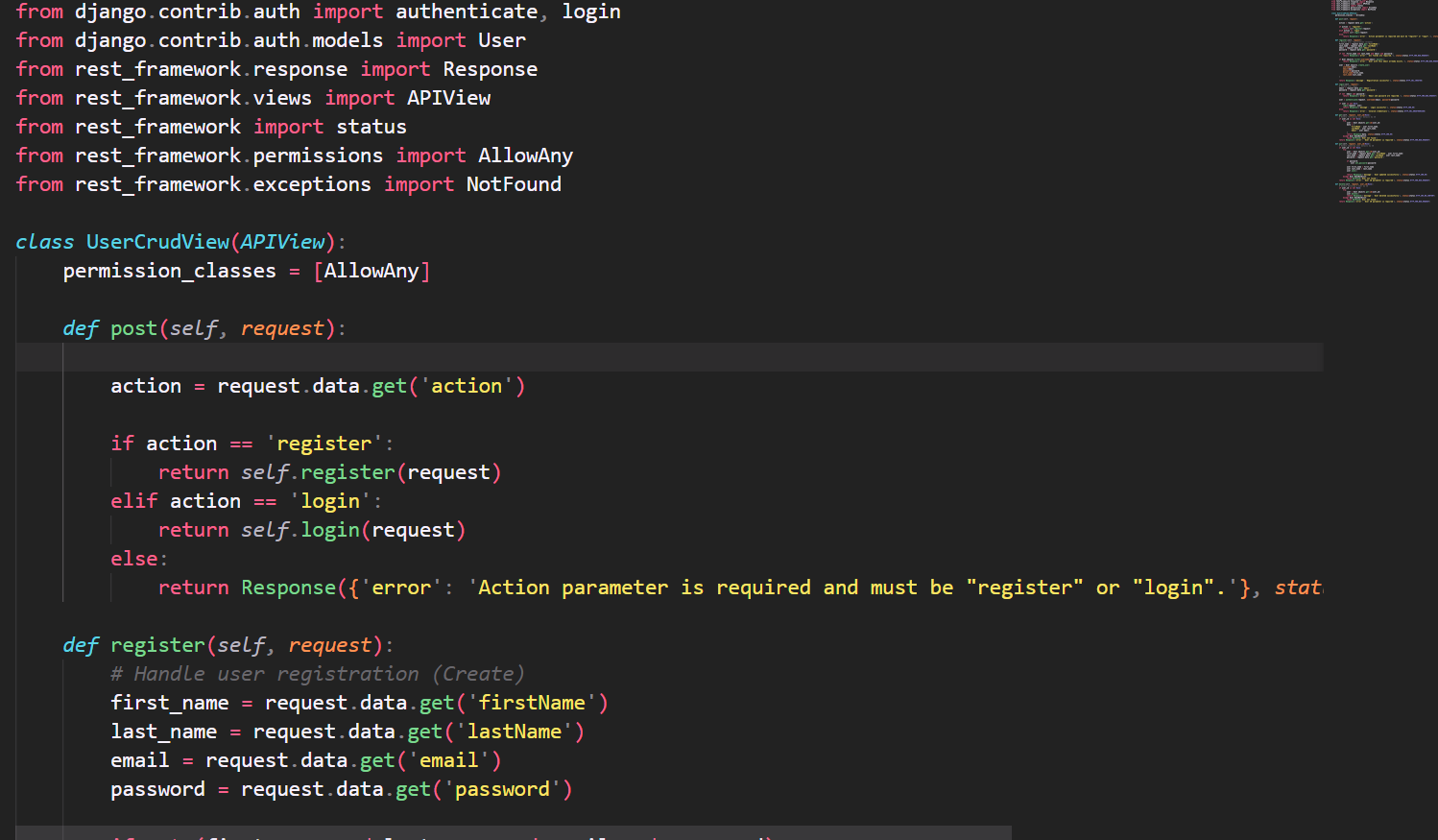
DATABASE IMAGE:

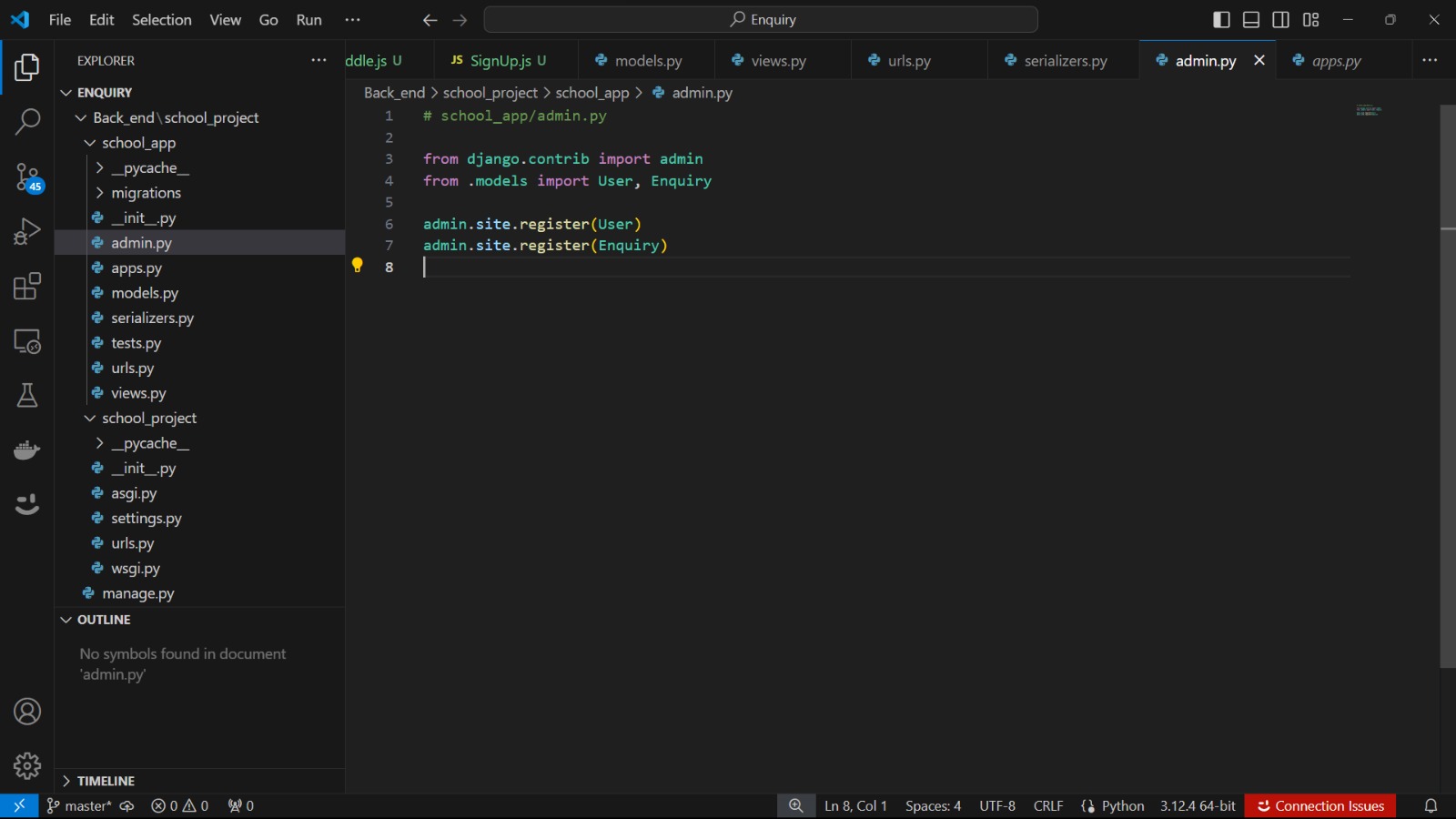


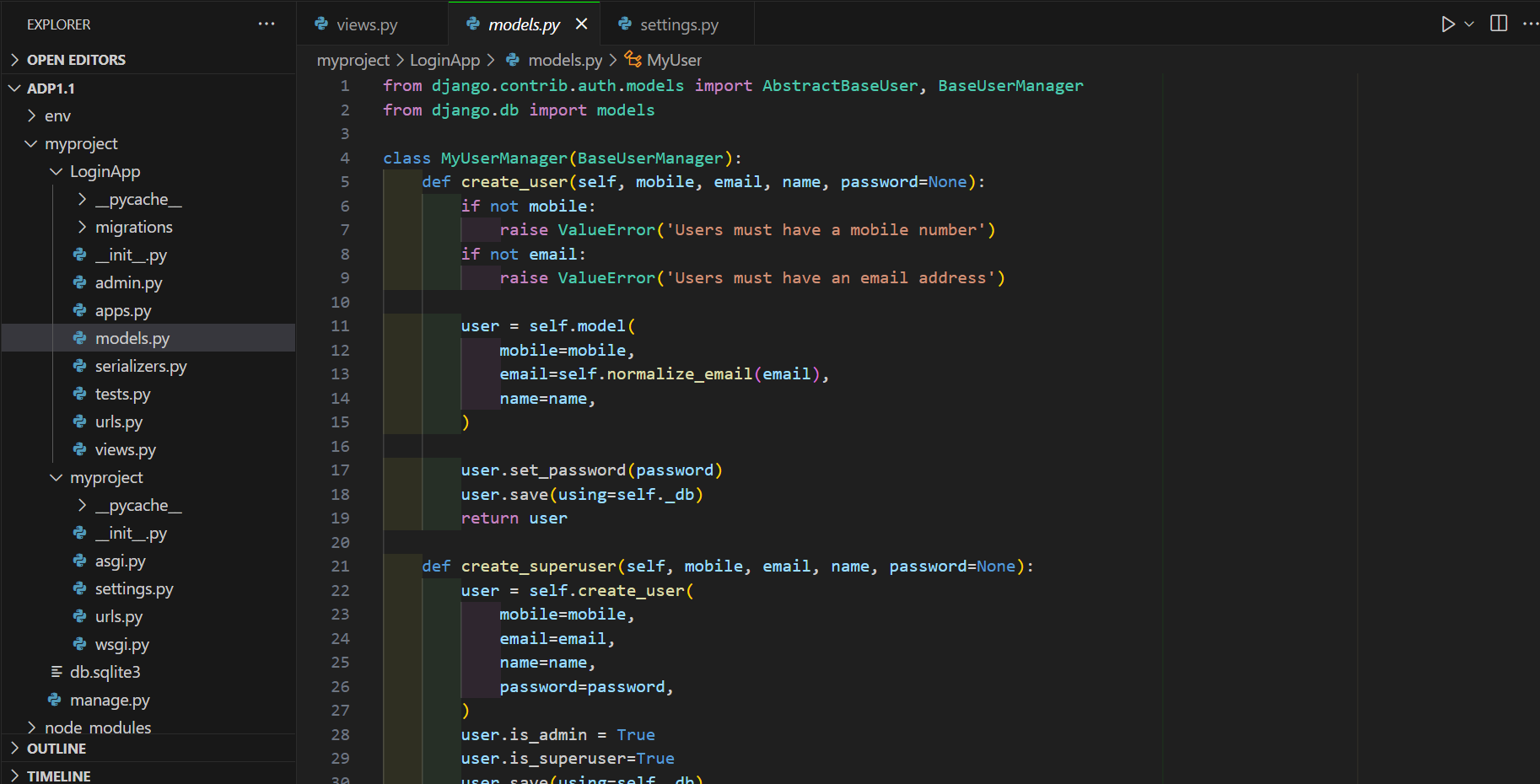


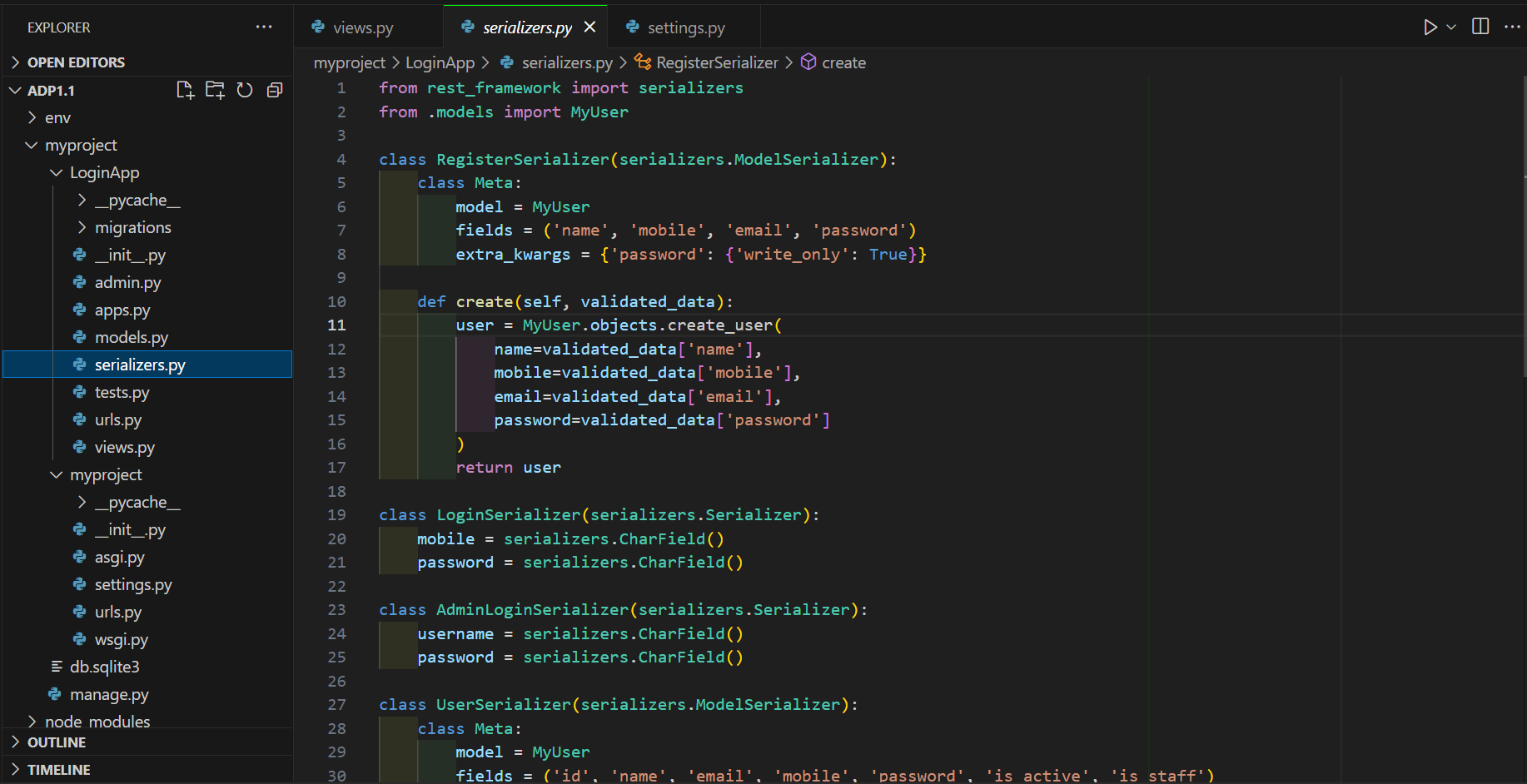
BACKEND IMAGES:











**6. Evaluation of System Performance**

**Overview**

The performance evaluation of the Online Mobile Recharge Portal is critical in ensuring that the system operates efficiently, responds quickly to user interactions, and remains reliable under varying loads. This evaluation uses both quantitative and qualitative measures to assess the system's overall performance.

1. **Performance Metrics**
   * **Response Time:** Measures the time taken for the system to respond to user requests, including recharge transactions and page navigations. Benchmarks are set to ensure response times remain within acceptable limits for a smooth user experience.
   * **Throughput:** Assesses the system's ability to handle numerous concurrent transactions and users. This is done through load testing to ensure the portal can manage peak usage effectively.
   * **Scalability:** Evaluates the portal's ability to scale with increasing user demand. This involves testing the system's capacity to handle additional users and transactions without compromising performance.
2. **Testing Methods**
   * **Load Testing:** Simulates multiple users accessing the portal simultaneously to evaluate how well it manages high traffic. Tools like Apache JMeter or Locust are used to conduct these tests.
   * **Stress Testing:** Pushes the system beyond its normal operating capacity to determine its breaking point and to assess its behavior under extreme conditions.
3. **Results**
   * **Response Time:** The average response time for transactions and page loads is measured. For example, recharge transactions should have an average response time of less than 500 milliseconds, with page load times under 2 seconds.
   * **Throughput:** The system's capability to process a specified number of transactions per second is evaluated. The portal should handle up to 1000 concurrent users without significant performance degradation.
4. **Optimization**
   * **Caching:** Implements caching strategies, such as using Redis or Memcached, to reduce load times and enhance performance by storing frequently accessed data.
   * **Code Optimization:** Involves refactoring code to remove inefficiencies and improve execution speed. This includes optimizing database queries and minimizing the time taken to render frontend components.

**Conclusion**

The performance evaluation confirms that the Online Mobile Recharge Portal meets the necessary performance standards, providing a fast, reliable, and scalable user experience. Ongoing monitoring and optimization are recommended to sustain high performance as the system continues to evolve and adapt to new user requirements.

**6 .1 User Feedback and Analysis**

**Overview**

User feedback is vital for assessing the effectiveness of the Online Mobile Recharge Portal and identifying areas for enhancement. This analysis centers on gathering, evaluating, and responding to user feedback to improve the portal's functionality and overall user experience.

1. **Feedback Collection Methods**
   * **Surveys:** Deploys surveys to collect both quantitative and qualitative feedback from users. The surveys cover aspects such as user satisfaction, usability, and the effectiveness of key features.
   * **Interviews:** Conducts one-on-one interviews with a representative sample of users to gain deeper insights into their experiences and gather detailed suggestions for improvement.
   * **Feedback Forms:** Provides in-portal feedback forms allowing users to submit their comments, suggestions, and report any issues directly within the portal.
2. **Key Findings**
   * **Usability:** Users generally find the portal easy to navigate and user-friendly. Positive feedback often highlights the intuitive design and straightforward instructions, while some users suggest minor adjustments for an even smoother experience.
   * **Features:** Core features, such as recharge processing and transaction history tracking, receive positive feedback for their effectiveness. However, users also suggest additional features, such as personalized offers or expanded payment options, to enhance the portal.
3. **Analysis**
   * **Satisfaction Levels:** Analyzes survey responses to gauge overall user satisfaction. Metrics like Net Promoter Score (NPS) and satisfaction ratings are employed to quantify user sentiments.
   * **Common Issues:** Identifies recurring issues and concerns mentioned by users, including usability challenges, feature requests, and performance-related problems.

**Conclusion**

The analysis of user feedback provides valuable insights into the strengths and areas for improvement of the Online Mobile Recharge Portal. By systematically collecting and acting on user feedback, the portal can continue to evolve, ensuring it meets the needs and expectations of its users while maintaining high levels of satisfaction and engagement.

**7. Conclusion and Future Work**

The Online Mobile Recharge Portal has successfully achieved its core objectives of providing users with a fast, reliable, and user-friendly platform for managing their mobile recharges. Utilizing a modern technology stack comprising React, Django, and MongoDB, the portal has created a robust, scalable, and secure system that significantly enhances the mobile recharge experience for users.

**Key Achievements:**

* **Streamlined Recharge Process:** The portal has simplified the mobile recharge process, allowing users to quickly and efficiently manage their mobile services with just a few clicks.
* **User Experience:** The user-friendly interface, built with React, offers an intuitive experience for users, with features like real-time transaction updates and easy access to recharge history, contributing to a seamless user journey.
* **Backend Efficiency:** Django’s powerful backend framework ensures that transactions are processed efficiently and securely, with RESTful APIs facilitating smooth data communication between the frontend and backend.
* **Security Measures:** The portal employs comprehensive security practices to protect user data and ensure that all transactions are conducted securely and reliably.

**Future Work:** While the Online Mobile Recharge Portal has achieved significant milestones, there are several areas for future development and enhancement:

1. **Feature Enhancements**
   * **Advanced Analytics:** Implement advanced reporting and analytics features to provide users with insights into their recharge patterns, spending, and offers.
   * **Personalized Experience:** Develop personalized user dashboards with customized recommendations and offers based on user preferences and behavior.
2. **Performance Optimization**
   * **Scalability Improvements:** Continue to enhance the system’s scalability to accommodate increasing user numbers and transaction volumes. Explore additional optimization techniques to maintain high performance during peak usage.
   * **User Feedback Integration:** Regularly gather user feedback to refine the user interface and further improve usability, ensuring that the portal addresses any emerging needs or usability challenges.
3. **Integration and Interoperability**
   * **Third-Party Integrations:** Integrate with various payment gateways, mobile service providers, and loyalty programs to offer users a more comprehensive and convenient recharge experience.
   * **Mobile Accessibility:** Enhance mobile accessibility by developing dedicated mobile applications or improving responsive design to ensure the portal is fully functional across all devices and screen sizes.
4. **Security Enhancements**
   * **Continuous Monitoring:** Implement advanced security monitoring tools to detect and respond to potential threats in real-time, ensuring the ongoing protection of user data.
   * **Compliance:** Regularly update security practices to comply with evolving data protection regulations, maintaining user trust and safeguarding sensitive information.

**Conclusion:**

The Online Mobile Recharge Portal has established a strong foundation as a vital tool for users seeking a convenient and secure way to manage their mobile recharges. Moving forward, the focus will be on expanding features, optimizing performance, and ensuring the portal continues to meet the growing and evolving needs of its users. By addressing these future work areas, the portal can solidify its position as an indispensable resource in the digital service landscape.