RSET
RAJAGIRI SCHOOL OF ENGINEERING & TECHNOLOGY
(AUTONOMOUS)

UNIRIDE: CAMPUS CARPOOLING PLATFORM

MINI PROJECT REPORT

SUBMITTED BY

JOHAN ABRAHAM TARUN, U2004043
MEEKHA MYNO BABU, U2004052
MEEVAL AUGUSTINE, U2004053
ROSE P JOSE, U2004060

DEPARTMENT OF INFORMATION TECHNOLOGY

UNIRIDE: CAMPUS CARPOOLING PLATFORM

A MINI PROJECT REPORT

Submitted by

Ms. Johan Abraham Tarun, U2004043

Ms. Meekha Myno Babu, U2004052

Ms. Meeval Augustine, U2004053

Ms. Rose P Jose , U2004060

to

the APJ Abdul Kalam Technological University in partial fulfillment of the requirements of the award of the Degree

of

Bachelor of Technology

in

Information Technology

Department of Information Technology

Rajagiri School of Engineering & Technology
Rajagiri Valley, Kakkanad, Kochi-39

JULY 2023

DEPARTMENT OF INFORMATION TECHNOLOGY

RAJAGIRI SCHOOL OF ENGINEERING & TECHNOLOGY (AUTONOMOUS)

RAJAGIRI VALLEY, KAKKANAD, KOCHI- 39

RSET
RAJAGIRI SCHOOL OF ENGINEERING & TECHNOLOGY
(AUTONOMOUS)

CERTIFICATE

This is to certify that the report entitled "**UNIRIDE : CAMPUS CARPOOLING PLATFORM**" is a bonafide record of the work done by **JOHAN ABRAHAM TARUN**(Reg No: **RET20IT041**), **MEEKHA MYNO BABU**(Reg No: **RET20IT049**), **MEEVAL AUGUSTINE**(Reg No: **RET20IT050**), **ROSE P JOSE**(Reg No: **RET20IT058**) in partial fulfillment of the award of the Degree of Bachelor of Technology in Information Technology at Rajagiri School of Engineering & Technology, Kakkanad, Kochi during the academic year 2022-23.

Ms. Viji Mohan	**Ms. Priya Mariam Raju**	**Dr.Neeba E A**
Project Guide	Project Coordinator	Head of the Department
Assistant Professor	Assistant Professor	HoD
Dept of IT	Dept of IT	Dept. of IT
RSET	RSET	RSET

Submitted for the practical examination conducted on……

Internal Examiner

ACKNOWLEDGEMENT

A strong foundation is necessary to step on and start building and expanding. We are thankful for the management and our Principal Dr. P. S. Sreejith who has ensured this strong foundation for us, students, to start building our lives on.

We thank Dr. Neeba E A, Head of the Department, Department of Information Technology, from the bottom of our hearts for being present at every stage of this journey to help and support us and make this project successful.

This project would not have seen completion if it was not for the guidance from our guide, Ms.Viji Mohan Asst. Professor, Department of Information Technology, who shared her knowledge and advice throughout the project. We extend our sincere gratitude to her for being the great mentor she is.

We also extend our sincere gratitude to our mini-project coordinator, Ms.Priya Mariam Raju, Asst. Professor, Department of Information Technology who coordinated us throughout the project.

Finally, we express our heartfelt gratitude to our teachers and friends for their invaluable assistance and unwavering support, which played a pivotal role in driving us to accomplish this task successfully. Their encouragement and guidance have been instrumental in our journey.

ABSTRACT

In college campuses, limited seating capacity in buses often leads to congestion and inconvenience for students. This innovative solution tackles multiple challenges, including limited parking space and congestion, by encouraging students to share rides and reducing the number of individual vehicles on the road. With an emphasis on sustainability, the application promotes environmental consciousness by minimizing pollution and carbon emissions.

Additionally, it offers an affordable and convenient transportation alternative for students who do not possess a driver's license or have limited access to private vehicles. By optimizing available resources and addressing the limitations of college bus services, our carpooling web application aims to enhance mobility and foster a more efficient transportation ecosystem within the campus community. This application provides a platform for students to connect, coordinate, and reserve car seats, promoting efficient transportation within the college community.

The users include administrators, car owners, and college students who need a ride from campus to nearby destinations. The administrator of the student carpooling web application is responsible for managing user accounts, overseeing the system's operation, and ensuring adherence to the platform's policies. It is important to note that the carpooling service operates on a cash-on-delivery and card payment basis, providing a convenient payment option for users. Students can search for available rides, and reserve seats in rides offered by other students. In addition, the system can be used by car owners who want to offer rides and share the cost of fuel and maintenance.

The front-end interface uses HTML, CSS, and JavaScript to provide a user-friendly experience.Firebase is used to create and manage the database for the carpooling web application, enabling real-time data storage and retrieval.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

1.1 GENERAL BACKGROUND

In today's fast-paced world, efficient transportation is essential for connecting individuals with their daily activities and responsibilities. For students, commuting to educational institutions can be challenging, especially in urban areas with crowded public transport and limited college bus services. Recognizing this need for a convenient and reliable transportation solution, our project aims to develop a specialized Ride-Sharing Reservation Application designed specifically for students.

The primary objective of the Ride-Sharing Reservation Application is to create a campus carpooling platform that facilitates rides for students, enabling them to share vehicles as both drivers and passengers. By bringing together students with cars and those without, the app optimizes transportation resources within the campus community, offering a dependable means of commuting to and from college.

One of the key advantages of the app is its accessibility for students who do not possess a driver's license or have their own vehicle. With limited college bus services and crowded public transportation, these students often face difficulties in reaching their classes on time. The app bridges this gap by connecting them with fellow students willing to share their rides, ensuring a reliable and punctual mode of transportation.

Moreover, the Ride-Sharing Reservation Application aims to address the issue of crowded college buses and public transport. By encouraging students to carpool, the number of vehicles on the road reduces, leading to decreased traffic congestion in and around the college campus. This not only results in a smoother commute for students but also contributes to a more sustainable and eco-friendly transportation system.

Safety and security are paramount concerns, and the app addresses them by

exclusively catering to college students. With a sense of trust and familiarity among peers sharing rides, students can feel comfortable and at ease during their journey, fostering a sense of community and camaraderie.

The app's unique feature of manually updated routes by drivers sets it apart from conventional map-based ride-sharing platforms. This allows drivers to offer more direct routes, accommodating nearby destinations or areas not covered by standard navigation services. Consequently, students can reach their destinations faster, saving valuable time for academic or personal pursuits.

Additionally, the Ride-Sharing Reservation Application serves as a reliable backup in emergencies or unexpected situations. Students can access emergency rides when needed, ensuring that no one is left stranded during critical times.

Beyond the practical advantages, the app enhances connectivity among students from diverse backgrounds and academic disciplines. By bringing together individuals in a shared transportation community, the platform fosters a sense of unity and inclusivity within the college campus.

1.2 PROJECT OBJECTIVE

Uniride: a campus carpooling platform project will establish an efficient and user-friendly digital system that facilitates carpooling among students. The platform aims to reduce traffic congestion, promote sustainability by lowering the carbon footprint, and offer cost savings through shared rides. It also seeks to enhance community building, improve accessibility for individuals without personal vehicles, and prioritize safety and security. With a user-friendly interface, the platform aims to simplify the process of finding carpool partners, collect and analyze data for system optimization, and foster collaboration with other campus services.

Ultimately, the project aims to create a comprehensive and sustainable transportation solution that benefits the campus community, reduces

environmental impact, and fosters a sense of interconnectedness among its users.

1.3 SCOPE

The scope of Uniride involves creating an intuitive platform where users can sign up, create profiles, and find potential carpool partners based on their commuting preferences and schedules. The application will utilize an efficient matching algorithm to optimize carpool arrangements, reducing traffic congestion and promoting sustainability on campus. Security measures will be implemented to protect user data and ensure a safe environment for carpooling. By fostering a sense of community and shared responsibility, Uniride endeavors to become a vital tool in streamlining campus transportation and promoting a greener campus ecosystem.

CHAPTER 2

LITERATURE SURVEY

Carpooling usually consists of two to four persons shuttling with a driver. Carpooling involves a group of people who have similar time slots and go to the same office or school such that they share a vehicle owned by the driver. Many studies were conducted to conclude the benefits and advantages of using carpooling services. There are benefits for individuals and the entire community. For individuals; the most important reason to use carpooling is to reduce transportation expenses and reduce travel time. Researchers found that the most important reason to use carpooling is to decrease the expenses of car fuel and transportation fees.

Carpooling is considered to be a good alternative to public transportation and taxi services. Furthermore, riding with a group of people with similar interests is a good way to spend time. It is a good way to reduce the total traveling time and driving stress. The work presented noted that being able to ride and socially interact with others rather than to drive is a cause for carpooling. The study of using carpooling on the Massachusetts Institute of Technology campus showed that travel distance was reduced from 9% to 27%.

For communities; carpooling is a means to reduce traffic congestion and pollution. Some studies identified environmental responsibility as a motivation for carpooling. However, others pointed out the benefits of carpooling from a social point of perspective; less fuel consumption, less CO2 production, less traffic congestion, and more social interaction. Gargiulo et al. showed how applying carpooling in Dublin would decrease the amount of CO2 in the air. The case study proposed by him showed the efficiency of carpooling in moderating congestion in Delhi.

To the best of our knowledge, extensive work and research have been conducted to produce a carpooling app for general purposes. However, identifying the optimal and shortest route presents a neat solution but with high complexity. For instance,

Bicocchi et al. developed an application for ride-sharing based on the extraction of suitable information from mobility traces and using a clustering algorithm that is applied to labeled trajectories. An android carpool application was developed to allow passengers to collaborate, plan for their journey, and share expenses. In this, the author presented two applications; the SplitCar application and the Buddy application for shared-use mobility in the metropolitan area of Bacău, Romania. The authors integrated different computing languages, standards, and technologies. Mallus et al. developed the CLACSOON application; a dynamic carpooling service in urban areas for real-time carpooling service using a multi-objective route matching algorithm.

Our proposed mobile carpooling application aims to generate a simple carpooling app that helps students of the Rajagiri School of Engineering and Technology (RSET) to find the most nearby drivers that match the passenger's preferences (i.e. gender) and thus to serve the ultimate goal of minimizing the overall travel distance, effort and the waiting time from campus to students homes located in Kochi city.

CHAPTER 3
METHODOLOGY

3.1 SYSTEM ANALYSIS AND DESIGN

The system analysis and design (SAD) of a campus carpooling platform involves a comprehensive and iterative process to develop an efficient and user-friendly system that facilitates carpooling to cater to the needs of students as both drivers and passengers. Car owners must register their vehicles, and the admin's approval is required based on car details, including license, registration number, and car make and model. This ensures a secure and trustworthy environment. Car owners can update seat availability and routes, enabling flexible rides for users. Students can easily reserve seats, simplifying the booking process. The platform is then implemented and thoroughly tested before deployment, with ongoing maintenance and support provided. The ultimate goal is to create a platform that optimizes campus transportation, reduces parking demand, promotes sustainability, and enhances community connectivity.

3.1.1 SYSTEM ARCHITECTURE

The proposed Uniride web application offers three distinct user types: the admin, and students who can act as both drivers and passengers. Figure 3.1.2 presents an overview of the system architecture, highlighting the functionalities available to each user. As an admin, their responsibilities include user management, car registrations, trip details storage, and handling reported issues. On the other hand, drivers have the capability to register their cars, update routes and seat availability, as well as manage their ride bookings. Users, who can be students acting as passengers, have the option to search for available rides based on specific preferences such as route, gender preferences, date, and the number of available seats. This multifaceted system aims to streamline carpooling and transportation options for students within the university community, providing a comprehensive and user-friendly experience.

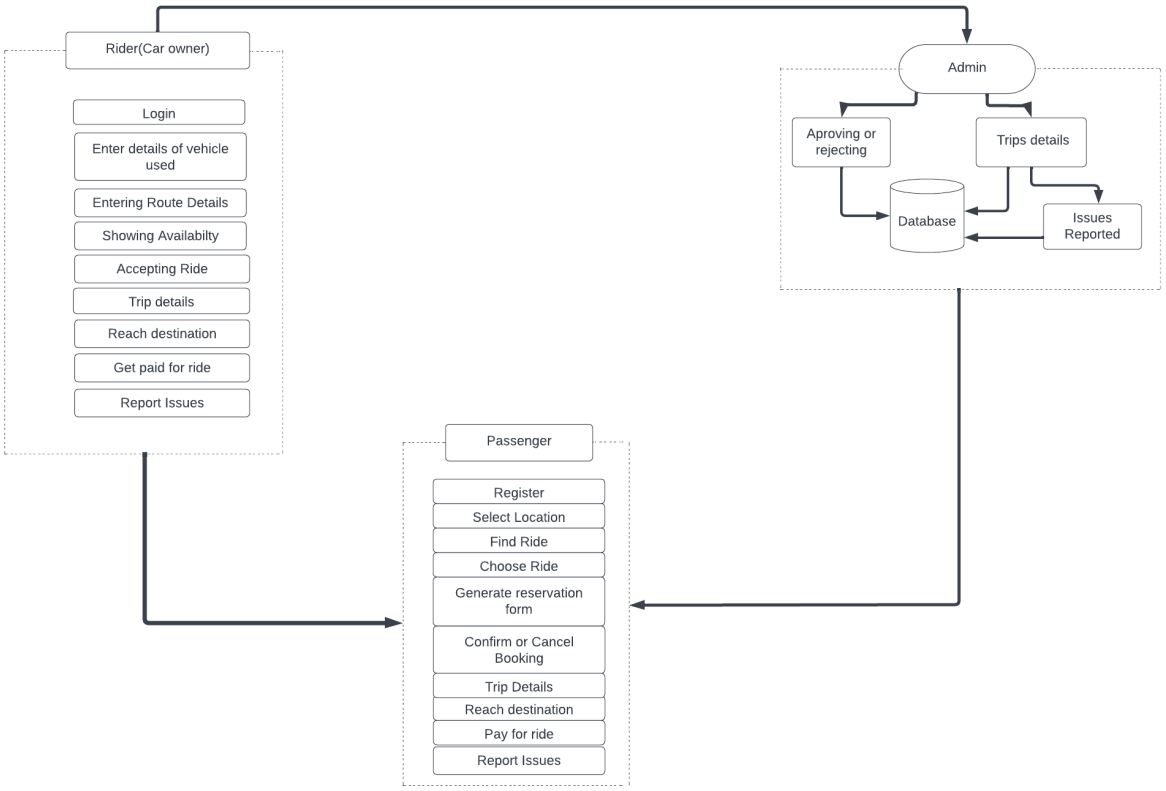


Figure 3.1.1 System Architecture

3.1.2 USE CASE DIAGRAM

The website allows the admin user to log into the system. Admin is also in charge of managing users and car registration, storing trip records, and dealing with reported concerns. The next user, the student, can be both a driver and a passenger. Drivers can register their vehicles by submitting relevant information such as license, registration number, and so on, after which they must wait for admin approval. If the administrator approves the registration, the related driver can adjust the number of available seats and the route. Passengers can reserve rides depending on their preferences, including route, gender, and seat availability. The admin, driver, and passenger can examine the trip details after confirming the ride. In case of any issues from the passenger or driver, it will be handled by the admin.

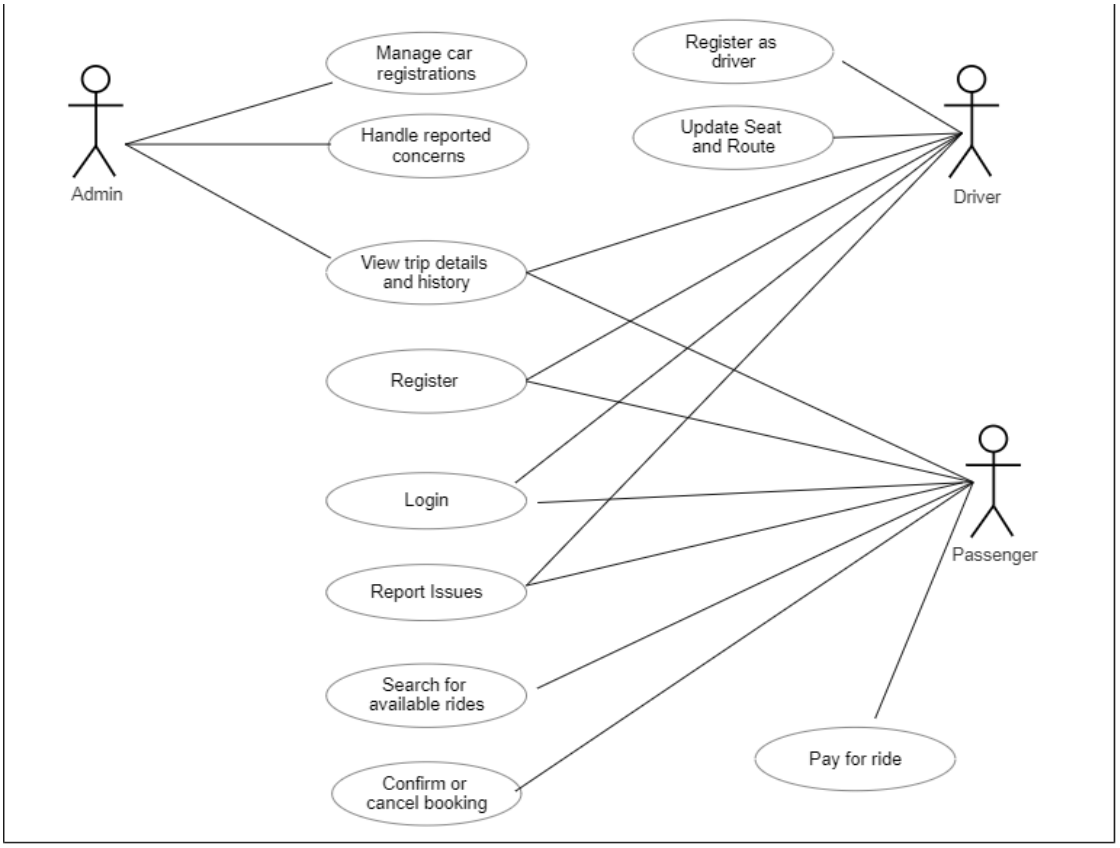


Figure 3.1.2:Use case diagram

3.1.3 ER DIAGRAM

Users log in through uid and password. They can enroll as riders and passengers. If they are riders they have to specify their name, gender,uid, semester, phone number, registration number, and car make and model. If they are passengers they have to specify uid, name, gender, and time. There will be an admin who manages the registration process of riders. They validate whether the given details are right or wrong. If the admin approves they can provide the bookings. The riders can also update their seats every day. Passengers can book their seats based on the available slots. The admin stores the booking details such as passenger uid, driver uid, location, price, time, and seats. If there is any complaint regarding a ride, the users can report it by specifying their uid, name, and the issue.

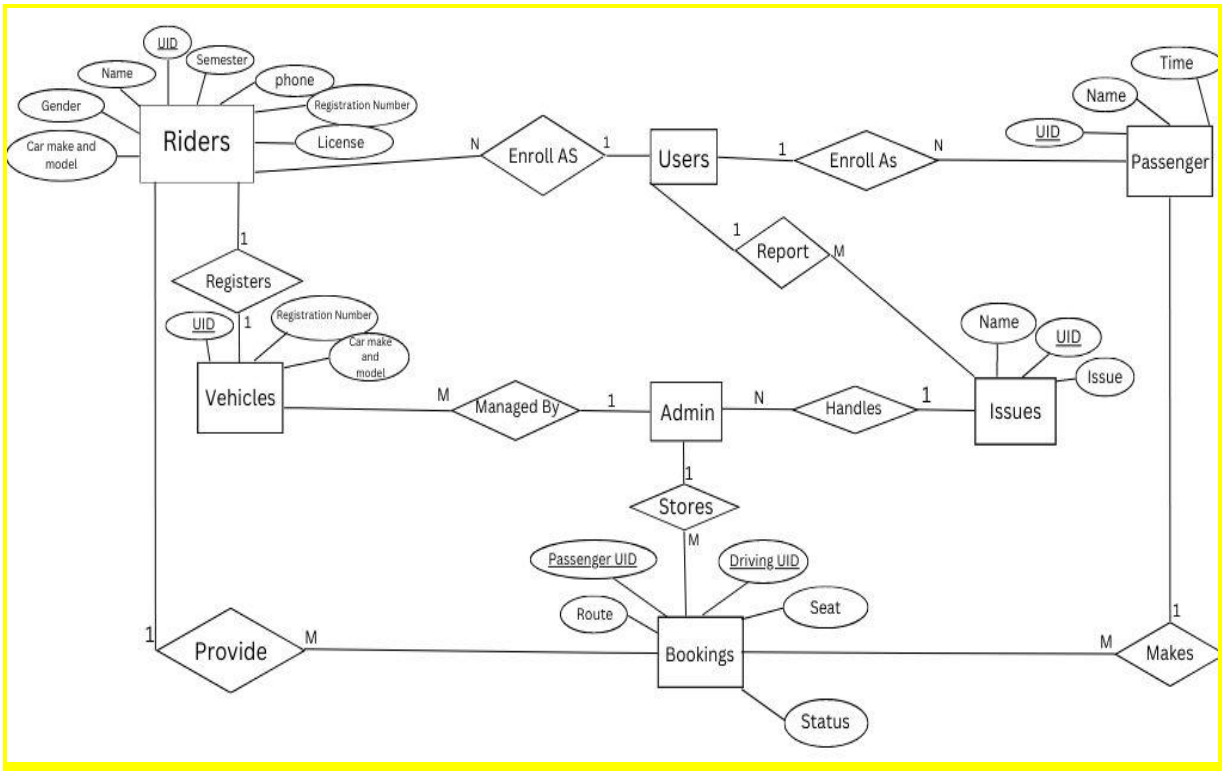


Figure 3.1.3 ER diagram

3.2 MODULE 1- ADMIN

- The Admin user can
- Approve or reject car registrations by accessing the data stored in the car_details database.
- View all the trip details and history stored in the bookings collection.
- View and handle all the issues reported by drivers and passengers stored in the reports collection.

3.3 MODULE 2- STUDENT DRIVER

The student driver user can

- Register themselves by entering their details into the registration_list collection in the real-time database.
- Log into the web application using the credentials provided by the driver upon registration which is stored in the registration_list database.
- Register themselves as drivers by entering details into the car-details collections inside Firestore.

- View their registration status by accessing details from the car-details collection.
- Update seat availability and route by entering details into the car-details collection.
- View trip bookings and history by accessing data stored in the trips collections.
- Start a trip using google maps and reach destination by connecting to google maps and get paid for the ride.
- Report issues about the trip to the admin by entering necessary details into the reports collection.

3.4 MODULE 3- STUDENT PASSENGER

The student passenger user can

- Register themselves by entering their details into the registration_list collection in the real-time database.
- Log into the web application using the credentials provided by the driver upon registration which is stored in the registration_list database.
- View the seat availability by accessing the details stored in the car_details collection and filtering it based on preferences such as gender and route.
- Confirm or cancel bookings and store details inside the bookings collection.
- View trip bookings and history by accessing data stored in the trips collections.
- Report issues about the trip to the admin by entering necessary details into the reports collection.

3.6 TABLE DESIGN

registration_list

The registration_list collection is a part of a database that stores user information including fields such as email, full name, password, and uid. The primary key in this collection is uid.

car_details

This collection stores details of cars owned by students. It includes information such as the car's make and model, owner's name, gender, car photo URL, registration number, route, available seats, semester, status, and uid. Uid is the primary key and it is the

foreign key to the primary key in registration_list.

bookings

It includes details of the car used for the ride, driver's information such as name, gender, photo URL, registration number, route, available seats, semester, status, and uid, as well as the passenger's name and uid. Each booking has an automatically generated unique identifier as the primary key.

reports

This collection records any issues or complaints related to a ride. It includes information about the passenger (uid and name), the driver (uid and name), the reported issue, and the date of the booking.

- **User Data:** Creates a "registration_list" collection to store user information, including fields such as names, email addresses, contact numbers, authentication credentials, and user preferences.

- **Car registration Data:** Creates a "car details" collection to store both accepted and rejected registrations. Accepted registrations include seat availability and designated routes for ride-sharing arrangements. Rejected registrations are kept for record-keeping purposes.

- **Booking Data:** Creates a "bookings" collection to track the bookings made by users. This collection can include details about the driver as well as the passenger, booking time, location and price.

- **Trips data:** Contains details about all the current trips as well as the trip history with the option to pay.

- **Reported Issue Data:** Creates a report collection to store reported passenger/rider details along with the issue faced.

- **Admin Data:** The admin data in the application includes details about registered vehicles, trip information, and user-reported issues.

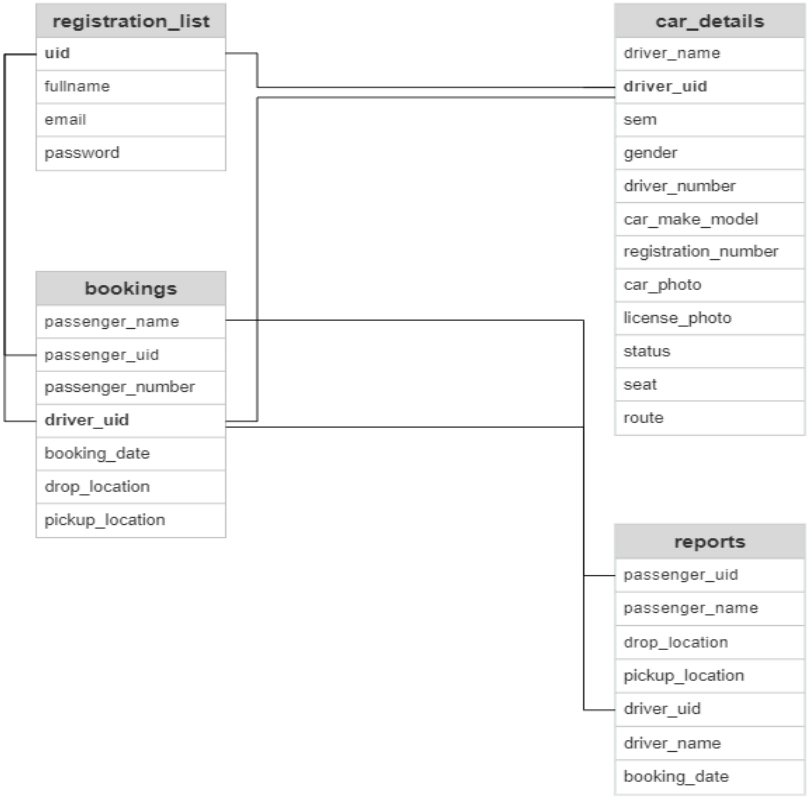


Figure 3.6: Depicts the Database design

3.7 GUI DESIGN

3.7.1. Registration page

This option enables students to register into the web application

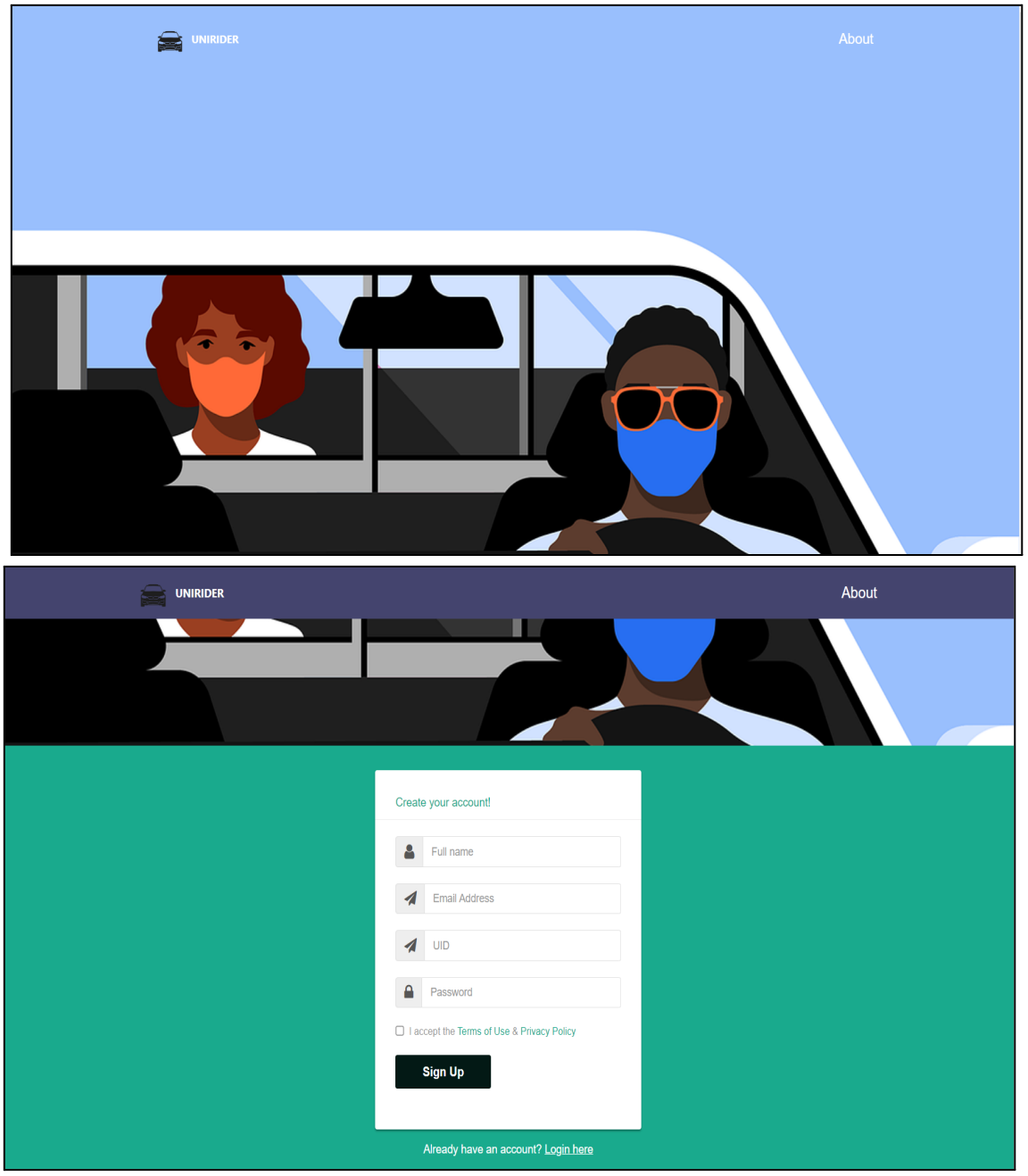


Figure 3.7.1.1 Depicts the registration page

3.7.2. Login page

This option enables users to login into the web application

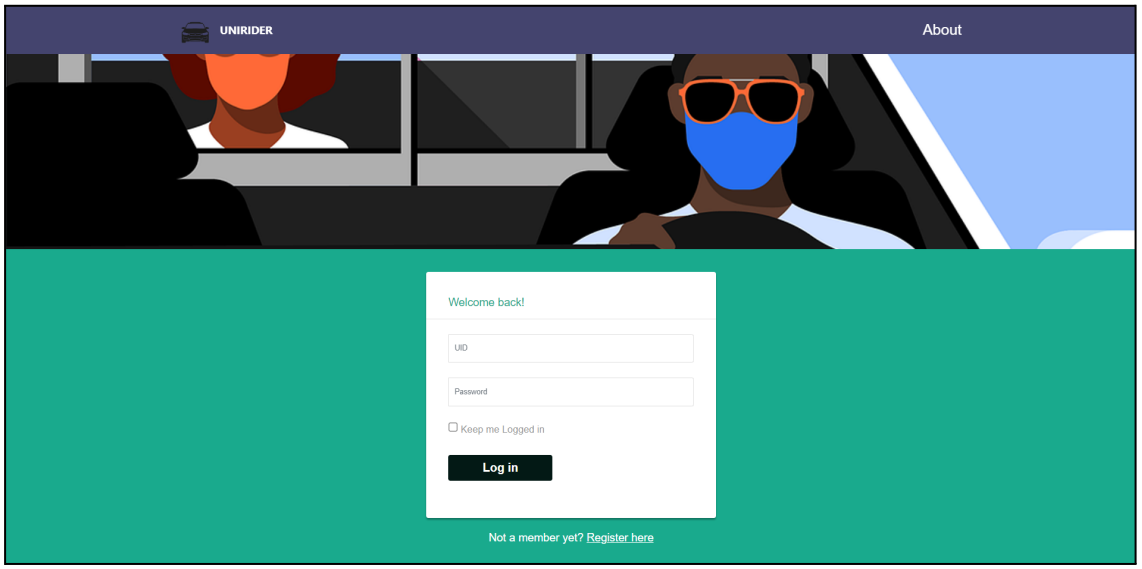


Figure 3.7.1.1 Depicts the Students' Login page

3.7.3. Home page

Figure 3.7.3.1 depicts the home page. The options include "Offer a ride" ie. for drivers to provide rides to students, "Book a ride" to book cars based on preferences and availability, "Trips" which displays all the current trip details and history, and a Logout button which redirects to the login page.

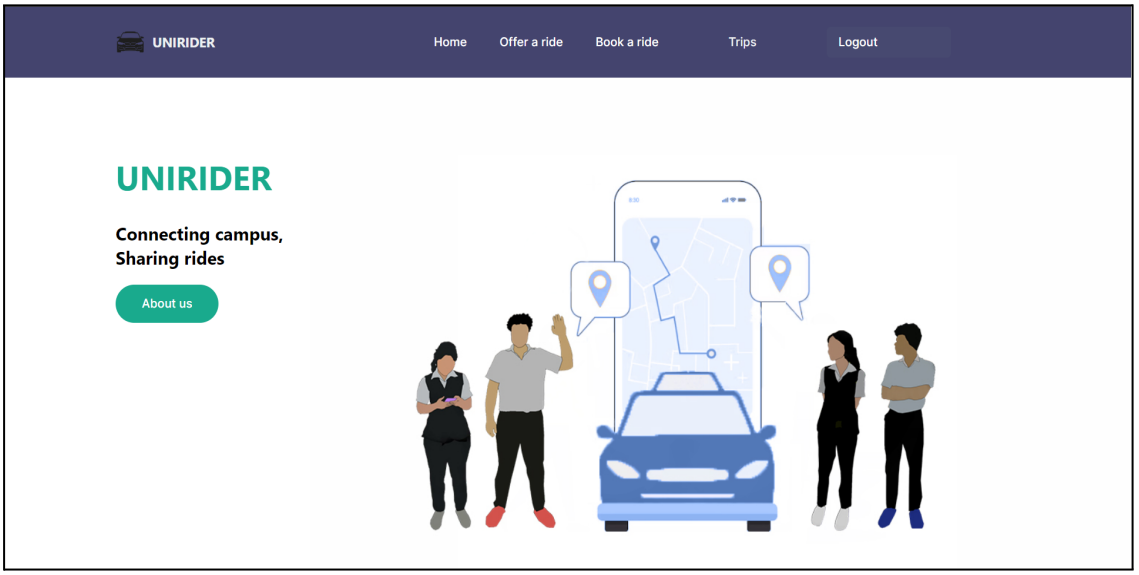


Figure 3.7.2.1 Depicts the home page

3.7.4. Depicts the Driver interface to provide rides

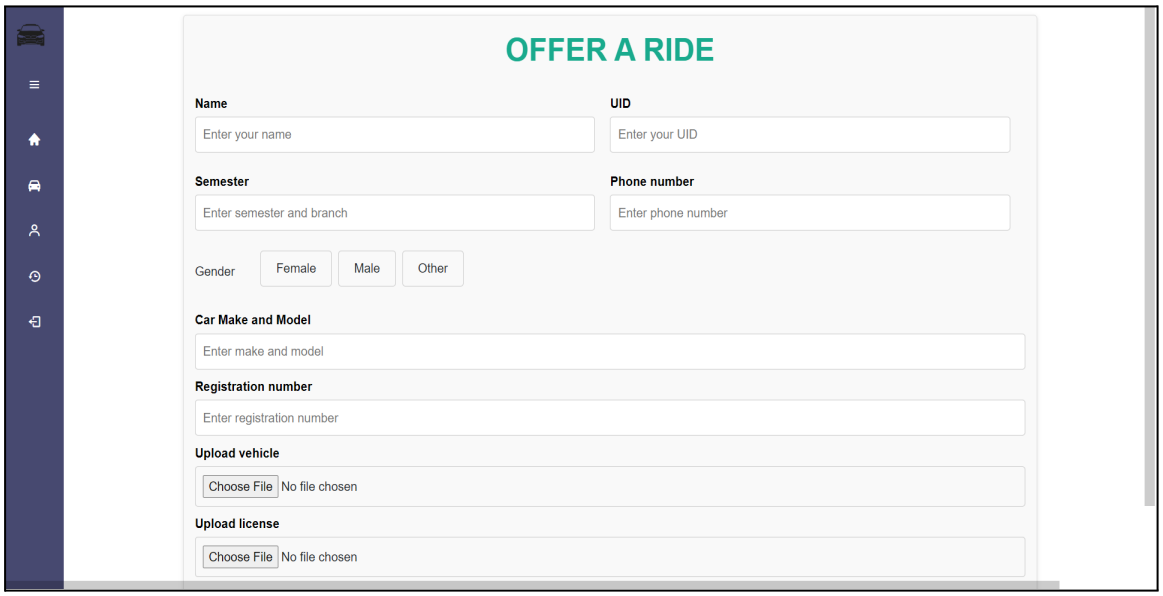


Figure 3.7.4.1 depicts the car registration form through which students can provide rides.

Figure 3.7.4.2 depicts the page to update seat and route that is only available if the admin accepts your car registrations. It also contains a button to "Turn off rides".

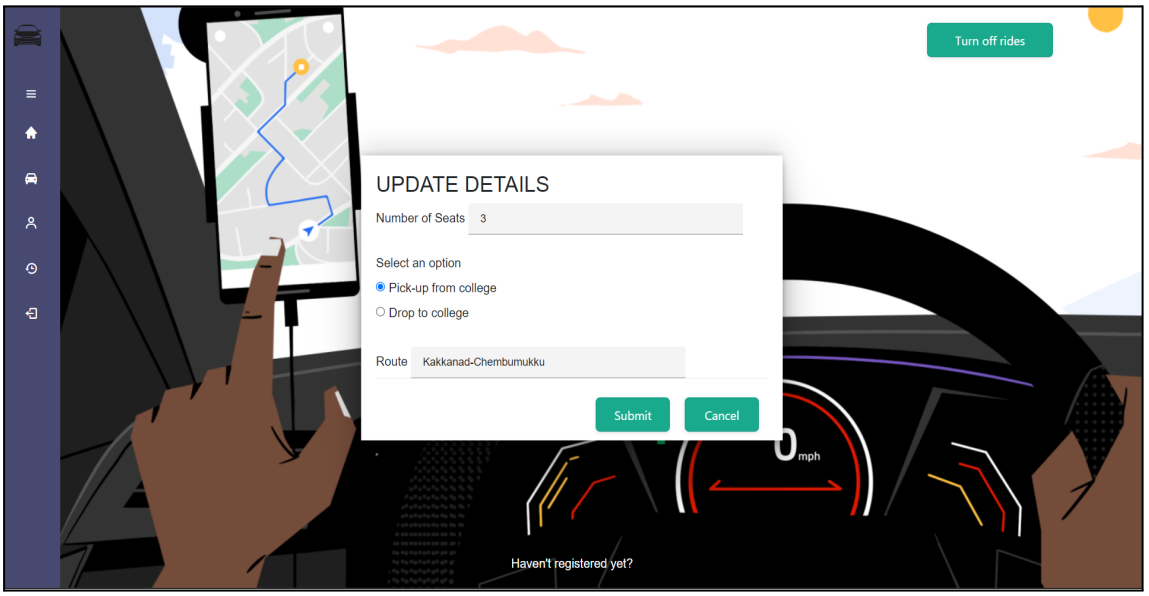


Figure 3.7.4.2 depicts the form with fields with seat availability, option to drop or pickup and route updation

3.7.5. Depicts the passenger interface to book rides

Figure 3.7.5.1 depicts the "Book a ride" page, where users can enter their UID, name, location, and phone number to get the closest ride based on their preferences and availability.

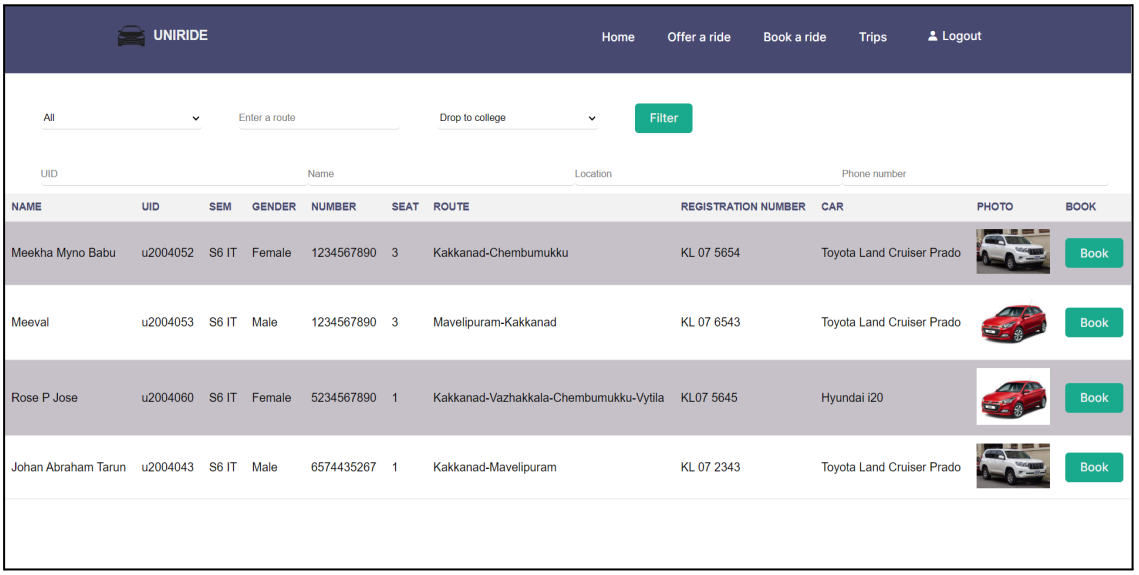


Figure 3.7.5.1 displays a screen containing input forms for the user's ID, name, address, phone number, and trip information, including the fare and the estimated arrival time, as well as filters for gender, route, and whether to pick up at or drop off at a college.

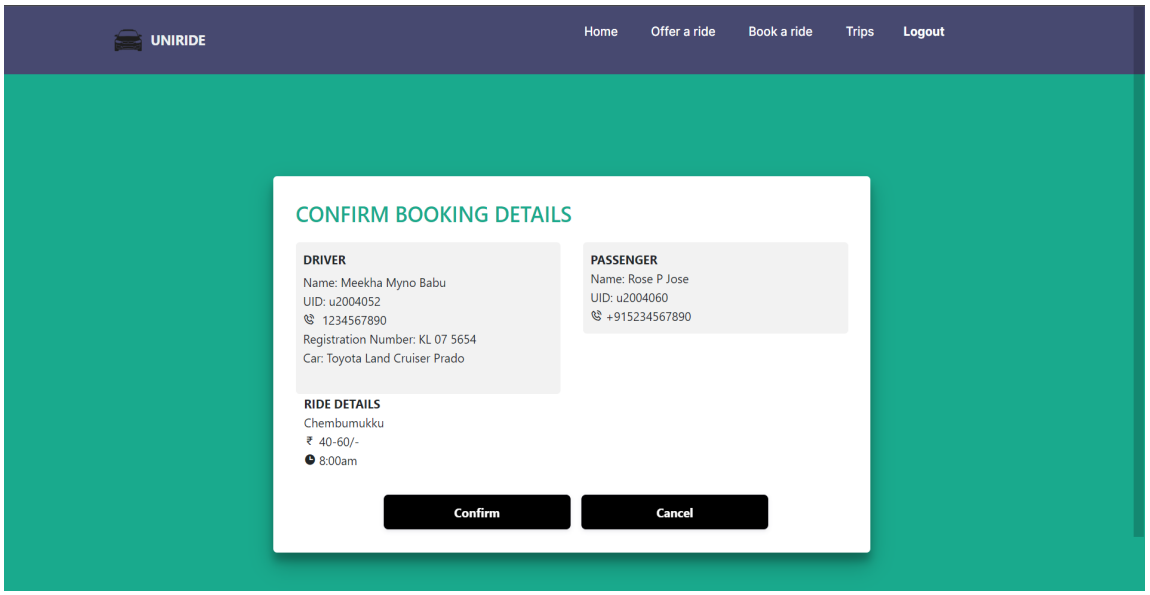


Figure 3.7.5.2 depicts the booking form with an option to confirm or cancel. By Clicking "Confirm booking", the passenger interface will have a decrement in the seat count.

3.7.6. Depicts the current trip details with the option to report and pay

Figure 3.7.6.1 depicts the current trip details. Details on passengers and rides are shown on the driver interface, and vice versa. Provides an option to pay, start a trip and also report in case of any problems.

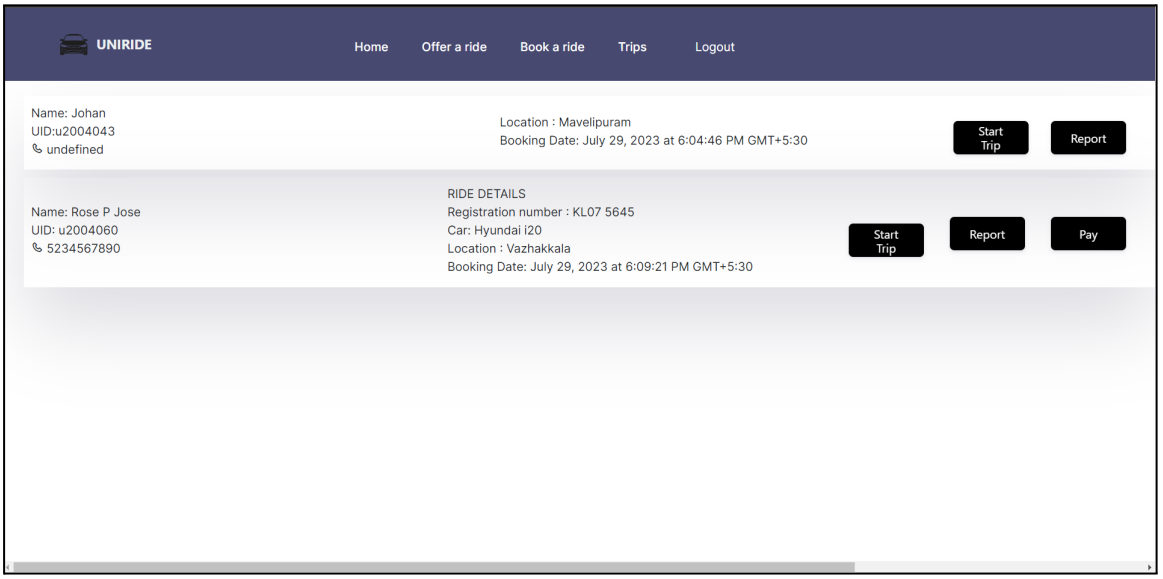


Figure 3.7.6.1 depicts trip details with name, uid, number, and ride details

Figure 3.7.6.1 depicts the Start trip button which when clicked redirects to google maps with the destination provided by the passenger

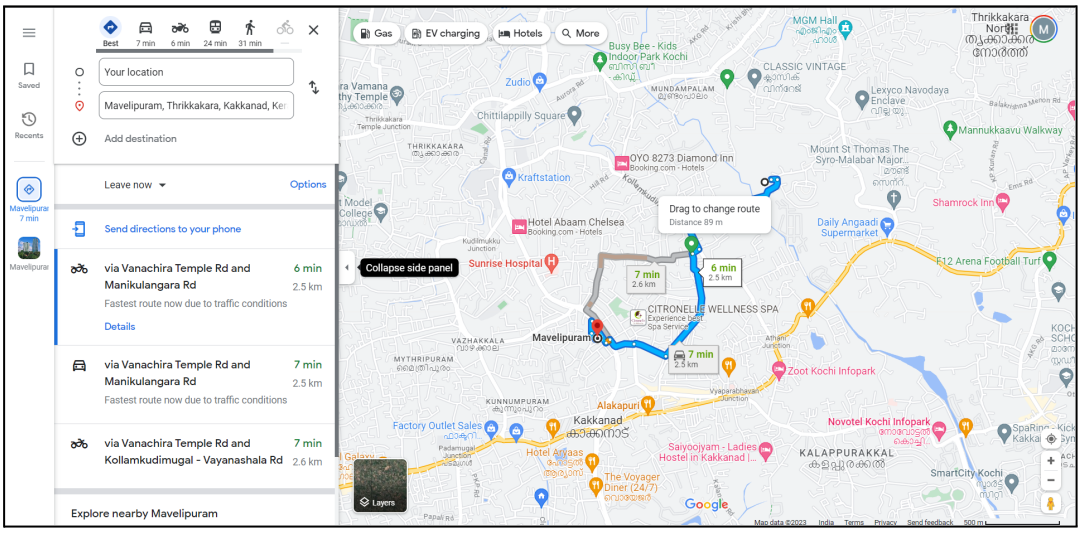
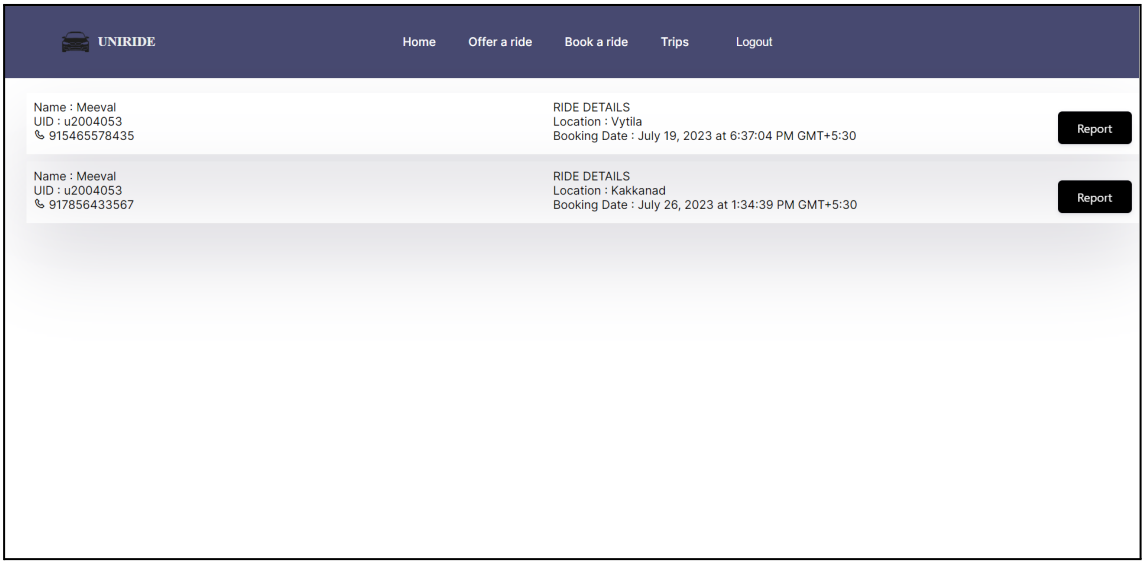


Figure 3.7.6.1 displays Google map with destination

3.7.7. Depicts the trip history with the option to report and pay

Figure 3.7.7.1 depicts the current trip details. Details on passengers and rides are shown on the driver interface, and vice versa. Provides an option to pay and also report in case of any problems.



` Figure 3.7.7.1 depicts trip details with name, uid, number and ride details

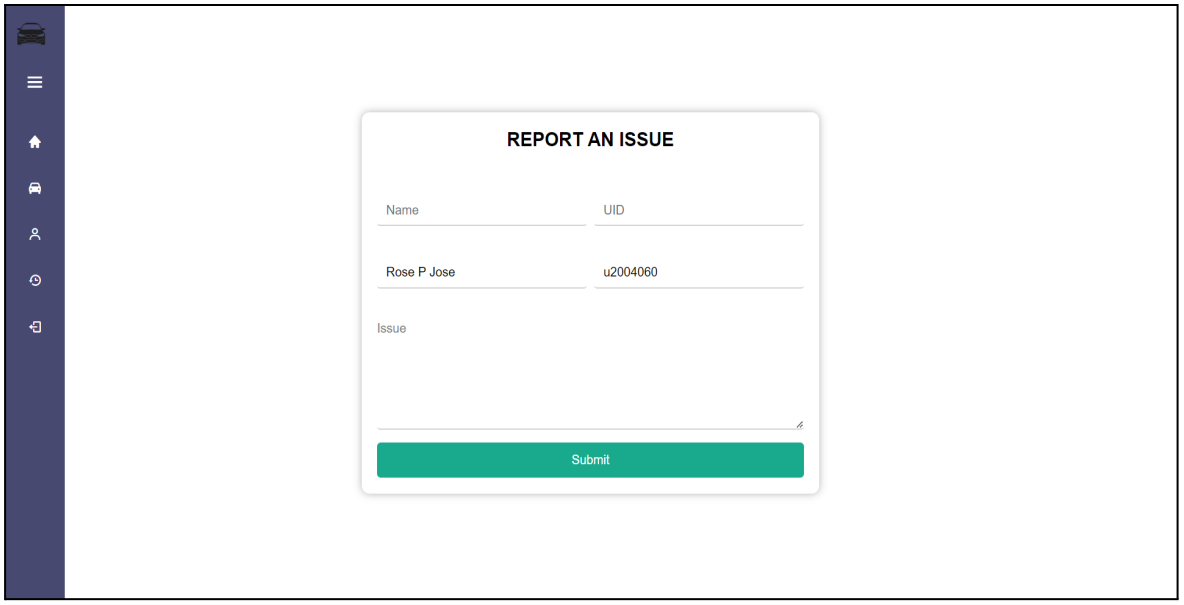
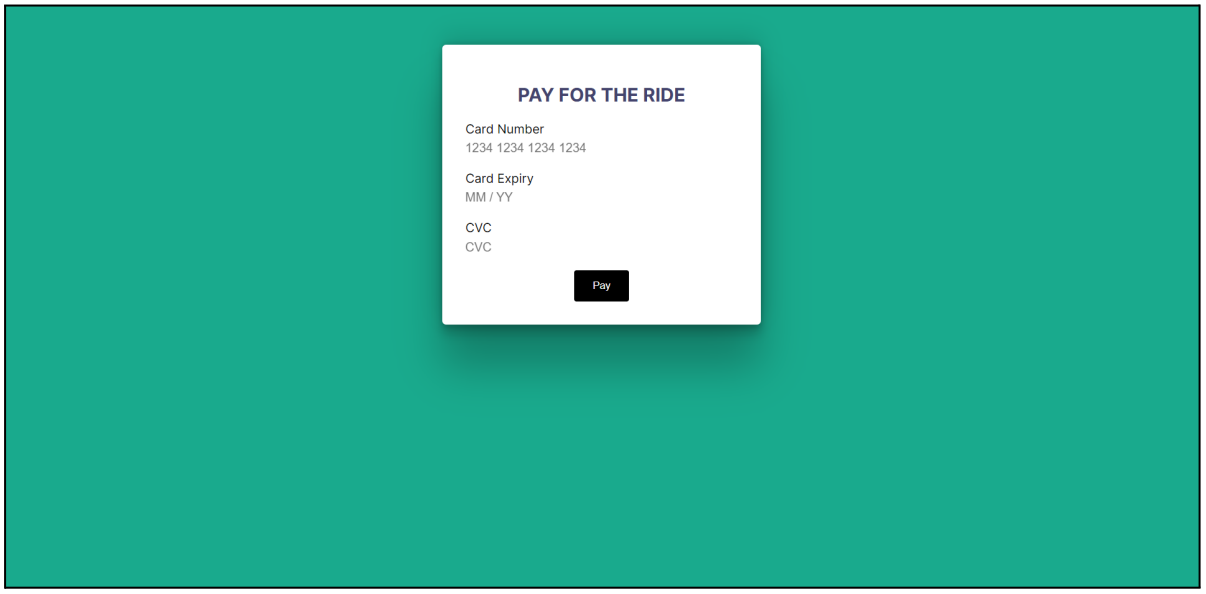


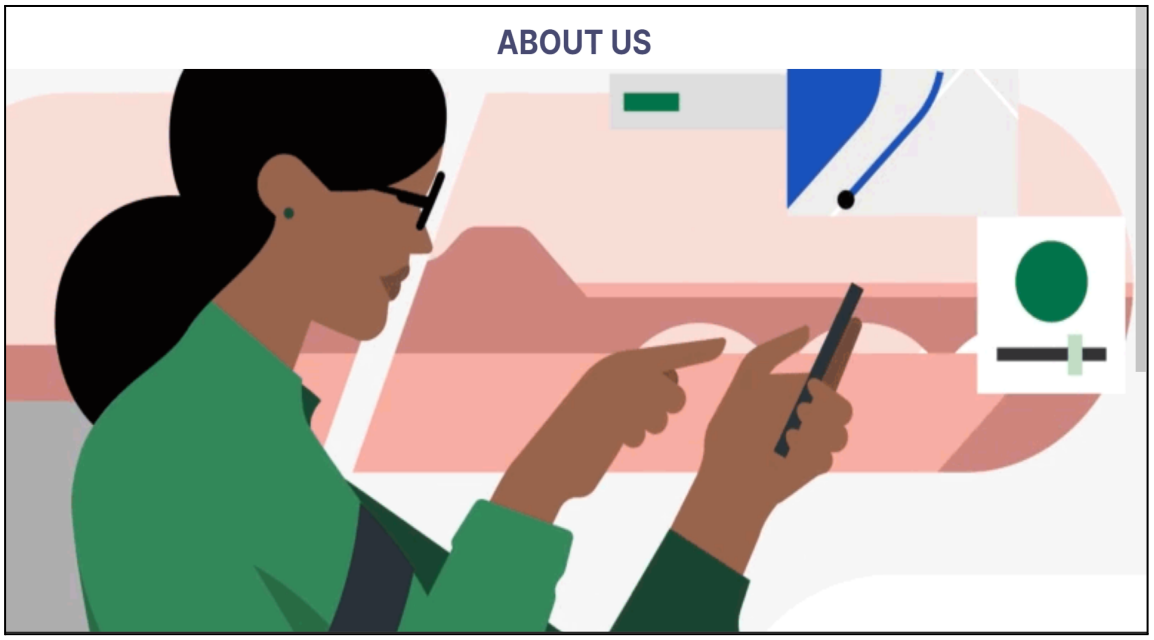
Figure 3.7.6.2 depicts the report form with driver and passenger uid, mail, and

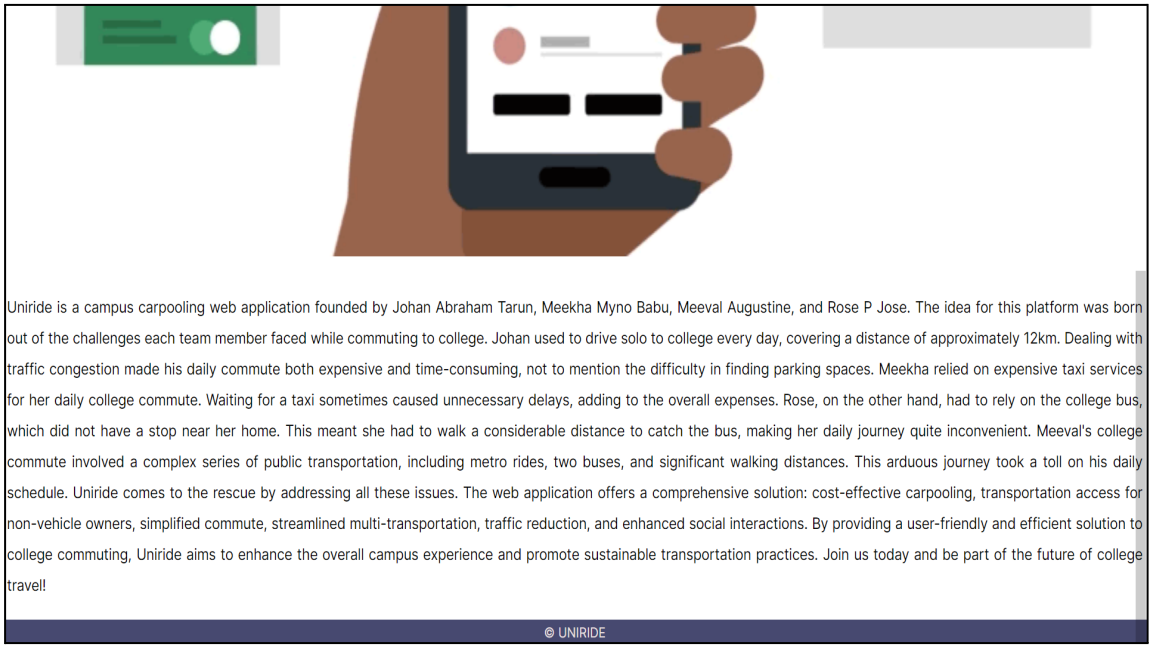


description about the issue

Figure 3.7.6.2 depicts the form to make card payments

3.7.8. Depicts the about us





Uniride is a campus carpooling web application founded by Johan Abraham Tarun, Meekha Myno Babu, Meeval Augustine, and Rose P Jose. The idea for this platform was born out of the challenges each team member faced while commuting to college. Johan used to drive solo to college every day, covering a distance of approximately 12km. Dealing with traffic congestion made his daily commute both expensive and time-consuming, not to mention the difficulty in finding parking spaces. Meekha relied on expensive taxi services for her daily college commute. Waiting for a taxi sometimes caused unnecessary delays, adding to the overall expenses. Rose, on the other hand, had to rely on the college bus, which did not have a stop near her home. This meant she had to walk a considerable distance to catch the bus, making her daily journey quite inconvenient. Meeval's college commute involved a complex series of public transportation, including metro rides, two buses, and significant walking distances. This arduous journey took a toll on his daily schedule. Uniride comes to the rescue by addressing all these issues. The web application offers a comprehensive solution: cost-effective carpooling, transportation access for non-vehicle owners, simplified commute, streamlined multi-transportation, traffic reduction, and enhanced social interactions. By providing a user-friendly and efficient solution to college commuting, Uniride aims to enhance the overall campus experience and promote sustainable transportation practices. Join us today and be part of the future of college travel!

© UNIRIDE

3.8 IMPLEMENTATION REQUIREMENTS

3.8.1 HARDWARE REQUIREMENTS

The selection of hardware configuration is a very important task related to software development, particularly inefficient RAM may adversely affect the speed and correspondingly the efficiency of the entire system. The processor should be powerful to handle all the operations. The hard disk should have sufficient capacity to solve the database and the application. The network should be efficient to handle the communication fast.

1. CPU: i3 Processor
2. Memory: 128 MB
3. Cache: 512KB
4. HardDisk: 4.3GB
5. Display: 15” Monitor
6. Keyboard: Standard 108 keys Enhanced KeyBoard
7. Mouse: Serial Mouse 3.8.2

SOFTWARE REQUIREMENTS:

The software requirements for the campus carpooling platform can be categorized into different components, including the development environment, backend, frontend, database, and additional tools. Here's a breakdown of the software requirements:

1. Operating System: WindowsXP, 7, 8 or above
2. Front End Tool: HTML, CSS, JavaScript
3. Back End Tool: Firebase, Node.js
4. IDE: VisualStudio

CHAPTER 4
RESULTS AND DISCUSSION

The development of the "Uniride" campus carpooling platform without the integration of maps but utilizing user-inputted routes has yielded promising results. During the implementation process, several key functionalities were successfully incorporated, including user registration, driver vehicle registration, passenger ride reservations based on specified destinations, and administrative management. The platform was designed to cater specifically to evening rides, making it available only when college classes are over. This unique approach has led to a focused and efficient carpooling system that meets the transportation needs of the campus community during peak commuting hours.

User Registrations and Route Input:
The user registration process was streamlined, allowing students to easily sign up using their college UID email addresses. Users were required to input their desired destination from their home for the morning rides, within the campus, or nearby areas for evening rides. This simplified approach allowed users to quickly find compatible carpooling options.

Driver Vehicle Registration and Admin Approval:
Drivers were required to submit relevant vehicle information, including license and registration numbers. Admin approval was granted after verifying the legitimacy of drivers and their vehicles. This ensured a pool of trustworthy drivers, enhancing the safety and reliability of the platform.

Passenger Ride Reservations and Route Matching:
Passengers could conveniently reserve rides based on their specified destination and find available drivers heading in the same direction.

The implementation of the "Uniride" campus carpooling platform with user-inputted routes has proven to be a viable and effective solution for evening transportation needs. By

mainly focusing on carpooling exclusively during the time when college classes are over, the platform optimizes ride-sharing opportunities, reducing the number of individual vehicles on the road during peak commuting hours. This targeted approach not only eases traffic congestion but also contributes to a more sustainable and environmentally friendly campus community.

Furthermore, "Uniride" has the potential to strengthen the campus community by encouraging interpersonal connections among students and staff. As individuals share rides to common destinations, they have the opportunity to interact and build social bonds, thereby enhancing a sense of belonging on campus.

CHAPTER 5
CONCLUSION

The development of "Uniride", a comprehensive campus carpooling platform, has successfully addressed the transportation needs of students while promoting sustainable and efficient commuting on campus. By providing a user-friendly interface and robust features, Uniride has streamlined the process of carpooling, making it convenient for both drivers and passengers.

The platform's three main user roles - admin, driver, and passenger - have distinct functionalities that ensure smooth operations and user satisfaction. The admin plays a pivotal role in managing users and vehicle registrations, ensuring that only legitimate drivers with approved vehicles are allowed to participate in the carpooling system. This approval process enhances safety and trust within the community.

Drivers can easily register their vehicles by submitting relevant information, which is then subject to admin approval. This step ensures that the vehicles meet the necessary standards for carpooling, and drivers can efficiently adjust the number of available seats and route details to cater to passenger demand.

Passengers, on the other hand, have the flexibility to reserve rides based on their preferences, including route options, gender considerations, and seat availability. This empowers passengers to find suitable carpools that match their requirements, enhancing their overall experience.

The platform's trip recording feature enables all parties involved to access and examine trip details after confirming the ride, promoting transparency and accountability. In the event of any concerns or issues arising during the trip, the admin is readily available to address and resolve them, fostering a safe and secure carpooling environment for all users.

In conclusion, "Uniride" serves as a remarkable solution to alleviate transportation

challenges on campus, reduce traffic congestion, and minimize the carbon footprint. By fostering a collaborative and eco-friendly culture of carpooling, the platform contributes to a sustainable and greener campus community. Moving forward, continuous improvements, updates, and proactive user engagement will be essential to ensure the platform's long-term success and positive impact on campus transportation. Uniride is poised to become an integral part of campus life, enhancing connectivity, reducing costs, and promoting a sense of community among students and faculty alike.

REFERENCES

[1] Tehran Seyedehsan Seyedabrishamia et. al, "Impact of Carpooling on Fuel Saving in Urban Transportation: a Case Study"

[2] Kum Kum Dewan and Israr Ahmad, "Carpooling: A Step To Reduce Congestion (A Case Study of Delhi)"

[3]Mateo Mallus et.al, "Dynamic Carpooling in Urban Areas: Design and experimentation with a Multi-matching algorithm"

[4]https://escholarship.org/content/qt7jx6z631/qt7jx6z631_noSplash_f4a899522996dfe33264b8219a10ed7b.pdf?t=ph07of

[5] https://firebase.google.com

[6]Farkhanda Zafar et. al, "Carpooling in Connected and Autonomous Vehicles: Current Solutions and Future Directions"

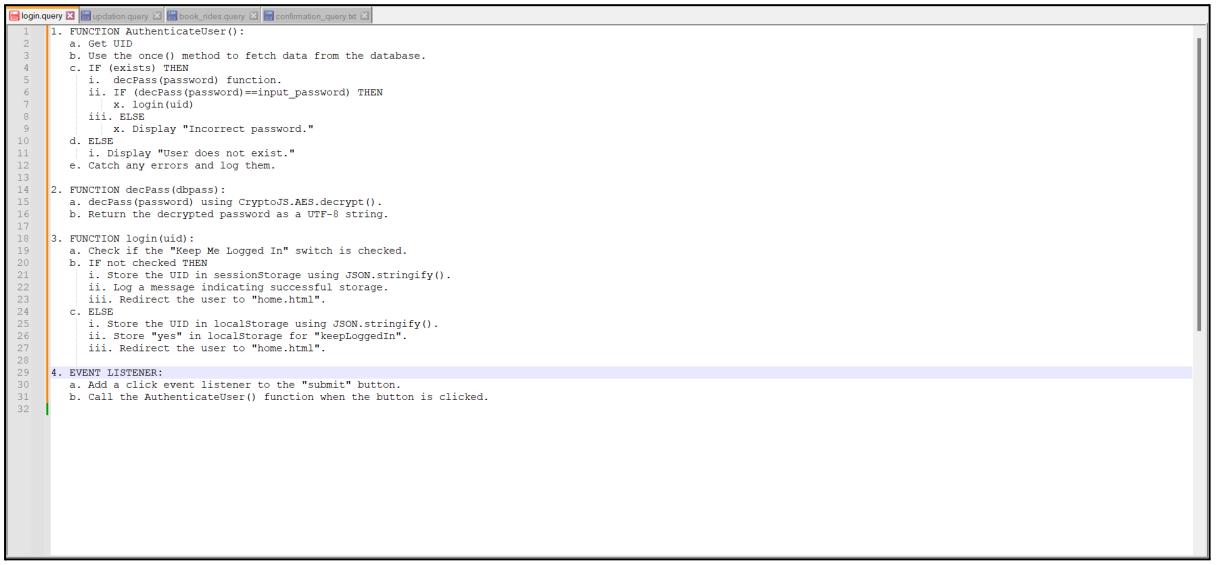
[7] Abeer Javed Syed,Sara Saba,Zain-ul-Abideen,Muhammad Noman and Adnan Talib, "CABTAB -A Factual Analysis Concerned with Travelling Issues, Specifically for An Organization", 2020 International Conference on Information Science and Communication Technology (ICISCT)

[8]https://www.academia.edu/35629949/Web_based_Carpooling_Android_Application

APPENDIX A

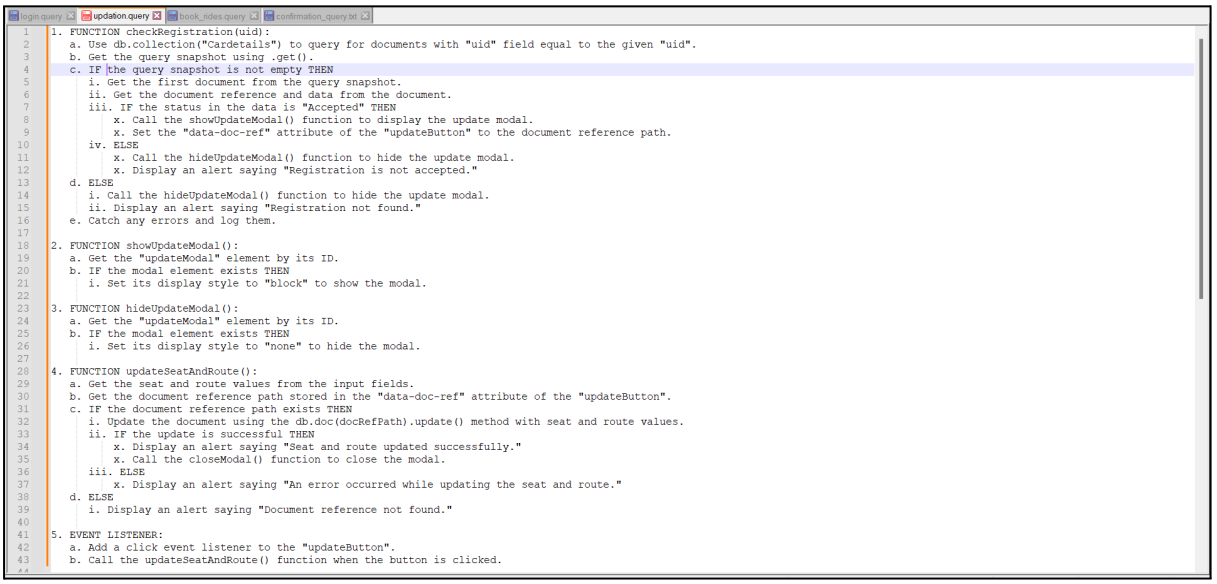
PSEUDO CODE

QUERY TO LOGIN TO THE APPLICATION PROVIDING CREDENTIALS



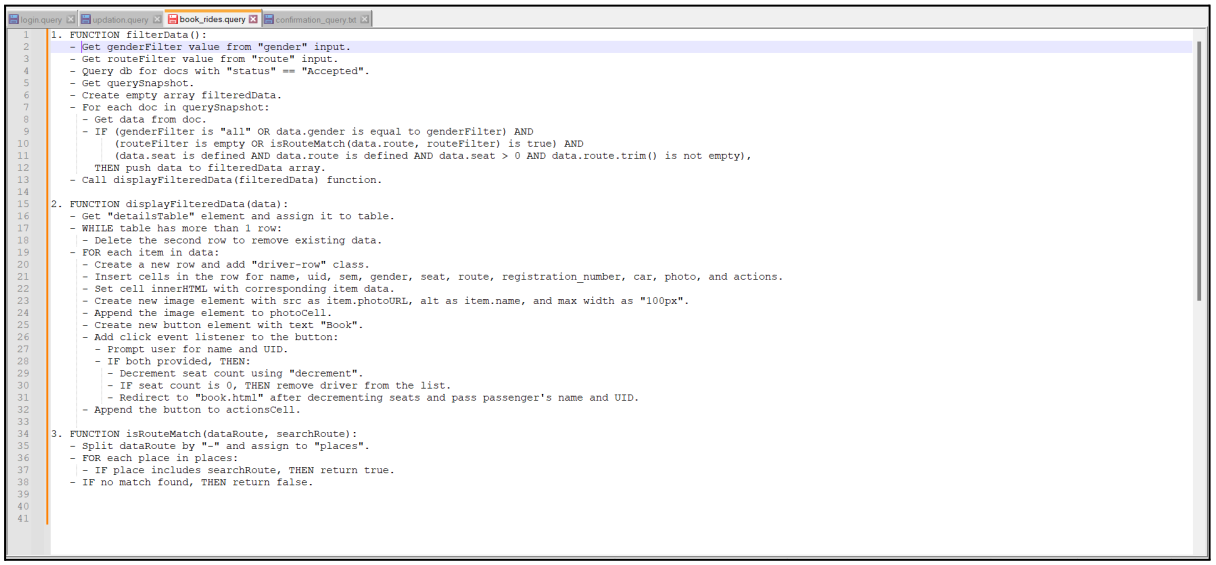
```
1. FUNCTION AuthenticateUser():
   a. Get UID
   b. Use the once() method to fetch data from the database.
   c. IF (exists) THEN
      i.  decPass(password) function.
      ii. IF (decPass(password)==input_password) THEN
          x. login(uid)
      iii. ELSE
          x. Display "Incorrect password."
   d. ELSE
      i. Display "User does not exist."
   e. Catch any errors and log them.

2. FUNCTION decPass(dbpass):
   a. decPass(password) using CryptoJS.AES.decrypt().
   b. Return the decrypted password as a UTF-8 string.

3. FUNCTION login(uid):
   a. Check if the "Keep Me Logged In" switch is checked.
   b. IF not checked THEN
      i. Store the UID in sessionStorage using JSON.stringify().
      ii. Log a message indicating successful storage.
      iii. Redirect the user to "home.html".
   c. ELSE
      i. Store the UID in localStorage using JSON.stringify().
      ii. Store "yes" in localStorage for "keepLoggedIn".
      iii. Redirect the user to "home.html".

4. EVENT LISTENER:
   a. Add a click event listener to the "submit" button.
   b. Call the AuthenticateUser() function when the button is clicked.
```

QUERY TO UPDATE SEAT AND ROUTE AFTER CHECKING THE STATUS OF THE CAR REGISTRATION



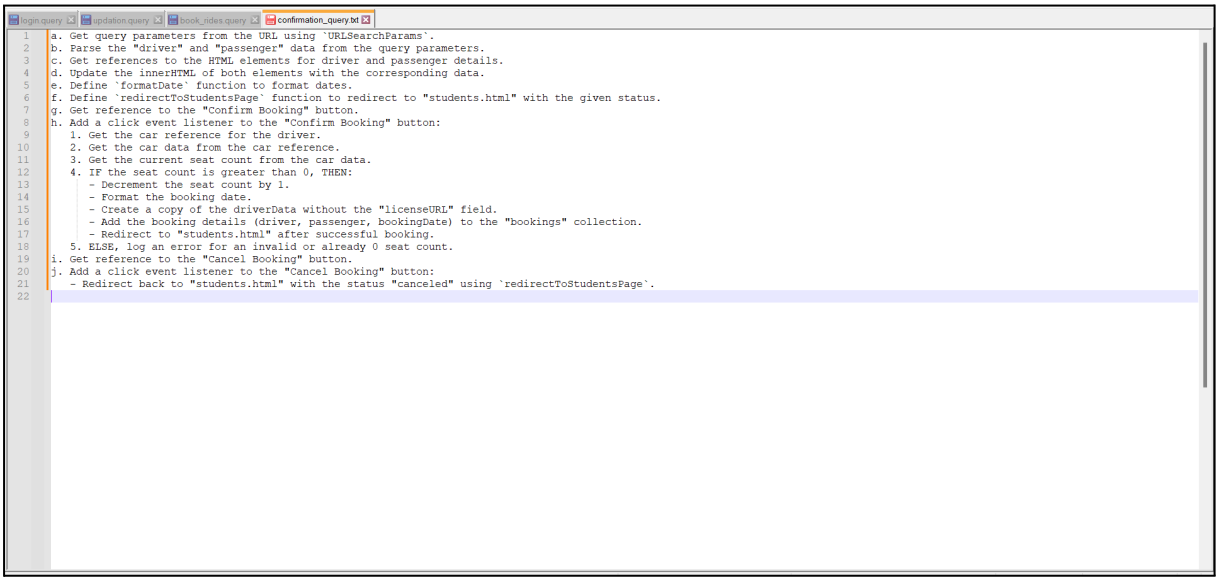
```
1. FUNCTION checkRegistration(uid):
   a. Use db.collection("Cardetails") to query for documents with "uid" field equal to the given "uid".
   b. Get the query snapshot using .get().
   c. IF the query snapshot is not empty THEN
      i. Get the first document from the query snapshot.
      ii. Get the document reference and data from the document.
      iii. IF the status in the data is "Accepted" THEN
          x. Call the showUpdateModal() function to display the update modal.
          x. Set the "data-doc-ref" attribute of the "updateButton" to the document reference path.
      iv. ELSE
          x. Call the hideUpdateModal() function to hide the update modal.
          x. Display an alert saying "Registration is not accepted."
   d. ELSE
      i. Call the hideUpdateModal() function to hide the update modal.
      ii. Display an alert saying "Registration not found."
   e. Catch any errors and log them.

2. FUNCTION showUpdateModal():
   a. Get the "updateModal" element by its ID.
   b. IF the modal element exists THEN
      i. Set its display style to "block" to show the modal.

3. FUNCTION hideUpdateModal():
   a. Get the "updateModal" element by its ID.
   b. IF the modal element exists THEN
      i. Set its display style to "none" to hide the modal.

4. FUNCTION updateSeatAndRoute():
   a. Get the seat and route values from the input fields.
   b. Get the document reference path stored in the "data-doc-ref" attribute of the "updateButton".
   c. IF the document reference path exists THEN
      i. Update the document using the db.doc(docRefPath).update() method with seat and route values.
      ii. IF the update is successful THEN
          x. Display an alert saying "Seat and route updated successfully."
          x. Call the closeModal() function to close the modal.
      iii. ELSE
          x. Display an alert saying "An error occurred while updating the seat and route."
   d. ELSE
      i. Display an alert saying "Document reference not found."

5. EVENT LISTENER:
   a. Add a click event listener to the "updateButton".
   b. Call the updateSeatAndRoute() function when the button is clicked.
```

QUERY TO BOOK A RIDE BASED ON FILTERING USING GENDER AND ROUTE



```
1. FUNCTION filterData():
   - Get genderFilter value from "gender" input.
   - Get routeFilter value from "route" input.
   - Query db for docs with "status" == "Accepted".
   - Get querySnapshot.
   - Create empty array filteredData.
   - For each doc in querySnapshot:
     - Get data from doc.
     - IF (genderFilter is "all" OR data.gender is equal to genderFilter) AND
          (routeFilter is empty OR isRouteMatch(data.route, routeFilter) is true) AND
          (data.seat is defined AND data.route is defined AND data.seat > 0 AND data.route.trim() is not empty),
       THEN push data to filteredData array.
   - Call displayFilteredData(filteredData) function.

2. FUNCTION displayFilteredData(data):
   - Get "detailsTable" element and assign it to table.
   - WHILE table has more than 1 row:
     - Delete the second row to remove existing data.
   - FOR each item in data:
     - Create a new row and add "driver-row" class.
     - Insert cells in the row for name, uid, sem, gender, seat, route, registration_number, car, photo, and actions.
     - Set cell innerHTML with corresponding item data.
     - Create new image element with src as item.photoURL, alt as item.name, and max width as "100px".
     - Append the image element to photoCell.
     - Create new button element with text "Book".
     - Add click event listener to the button:
       - Prompt user for name and UID.
       - IF both provided, THEN:
         - Decrement seat count using "decrement".
         - IF seat count is 0, THEN remove driver from the list.
         - Redirect to "book.html" after decrementing seats and pass passenger's name and UID.
     - Append the button to actionsCell.

3. FUNCTION isRouteMatch(dataRoute, searchRoute):
   - Split dataRoute by "-" and assign to "places".
   - FOR each place in places:
     - IF place includes searchRoute, THEN return true.
   - IF no match found, THEN return false.
```

QUERY TO CONFIRM RIDE AND DECREMENT SEAT COUNT

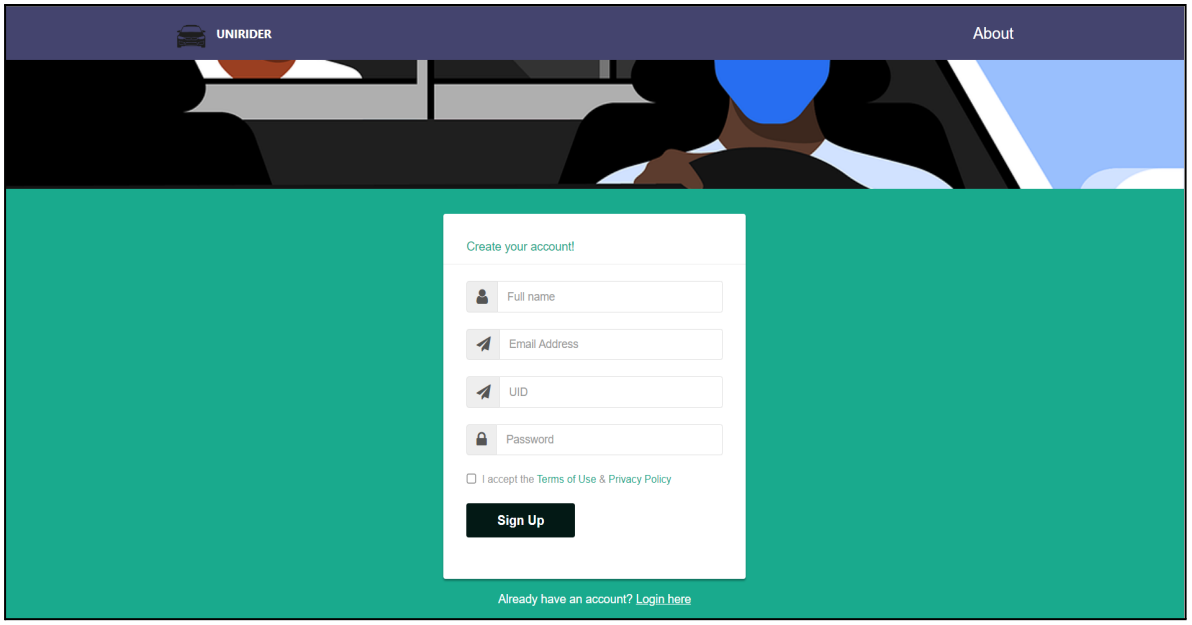


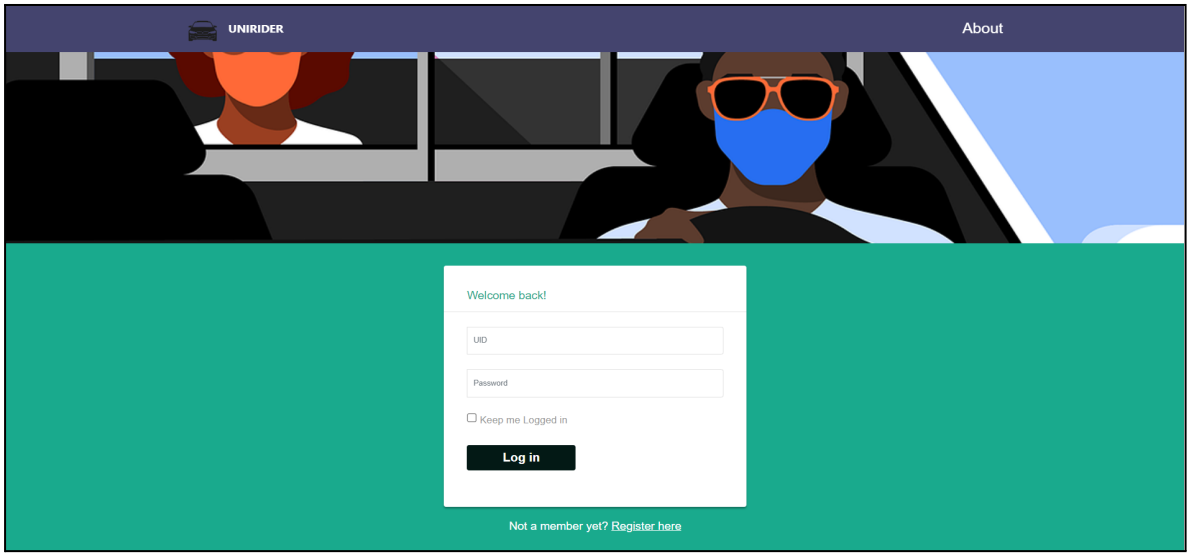
```
a. Get query parameters from the URL using `URLSearchParams`.
b. Parse the "driver" and "passenger" data from the query parameters.
c. Get references to the HTML elements for driver and passenger details.
d. Update the innerHTML of both elements with the corresponding data.
e. Define `formatDate` function to format dates.
f. Define `redirectToStudentsPage` function to redirect to "students.html" with the given status.
g. Get reference to the "Confirm Booking" button.
h. Add a click event listener to the "Confirm Booking" button:
   1. Get the car reference for the driver.
   2. Get the car data from the car reference.
   3. Get the current seat count from the car data.
   4. IF the seat count is greater than 0, THEN:
      - Decrement the seat count by 1.
      - Format the booking date.
      - Create a copy of the driverData without the "licenseURL" field.
      - Add the booking details (driver, passenger, bookingDate) to the "bookings" collection.
      - Redirect to "students.html" after successful booking.
   5. ELSE, log an error for an invalid or already 0 seat count.
i. Get reference to the "Cancel Booking" button.
j. Add a click event listener to the "Cancel Booking" button:
   - Redirect back to "students.html" with the status "canceled" using `redirectToStudentsPage`.
```

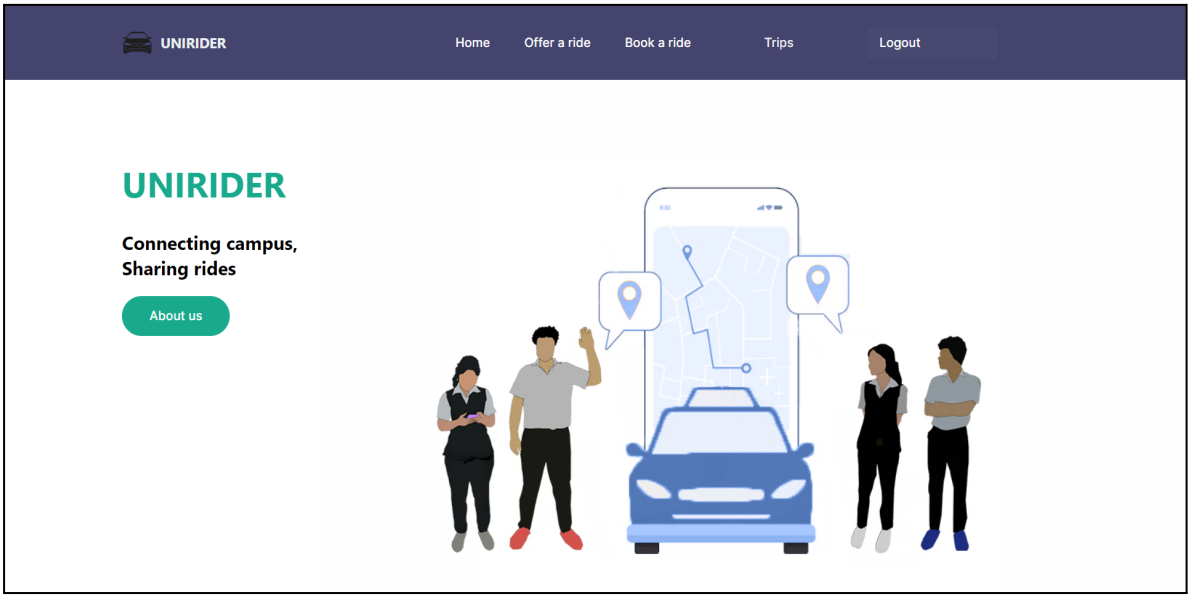
APPENDIX B

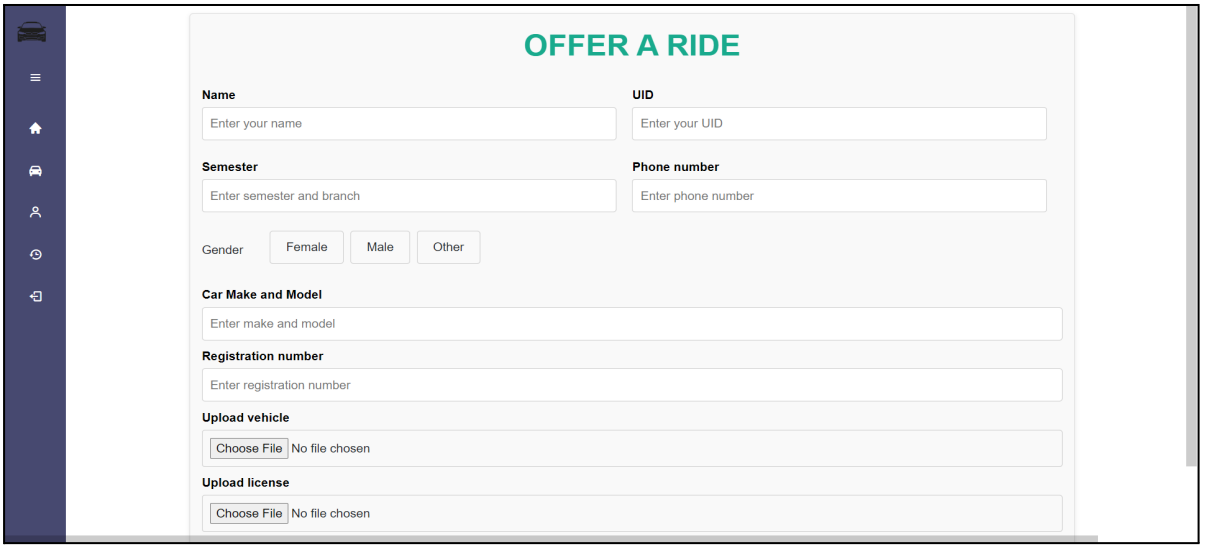
SAMPLE SCREENSHOTS

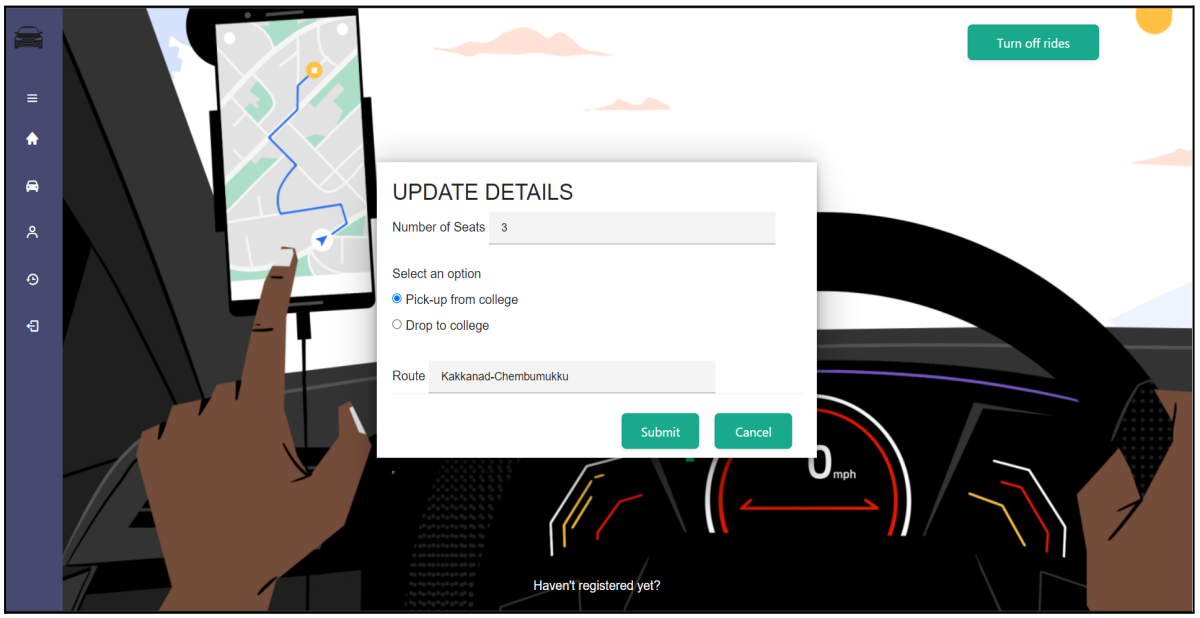
USER SIDE

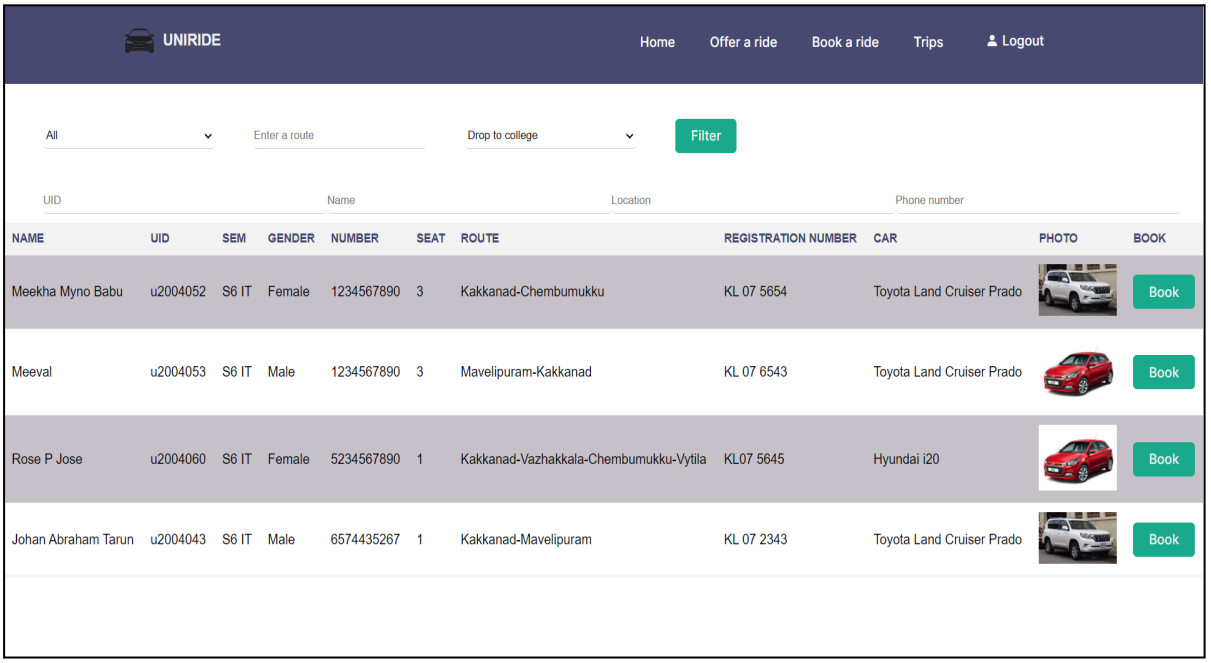


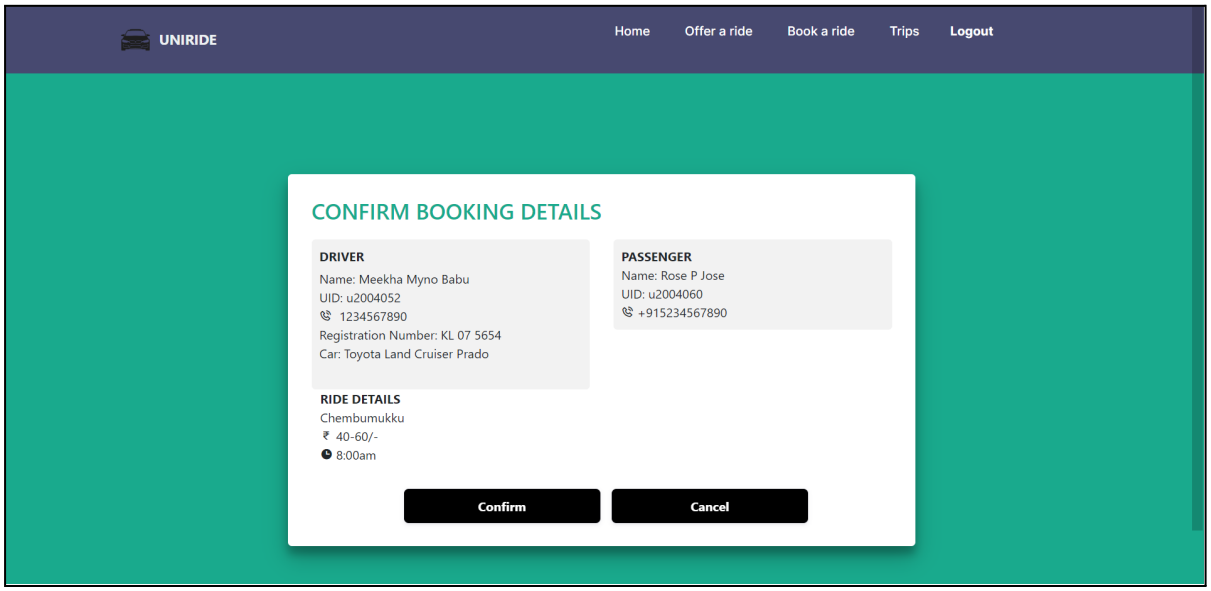

UNIRIDER
About
Welcome back!
UID
Password
Keep me Logged in
Log in
Not a member yet? Register here

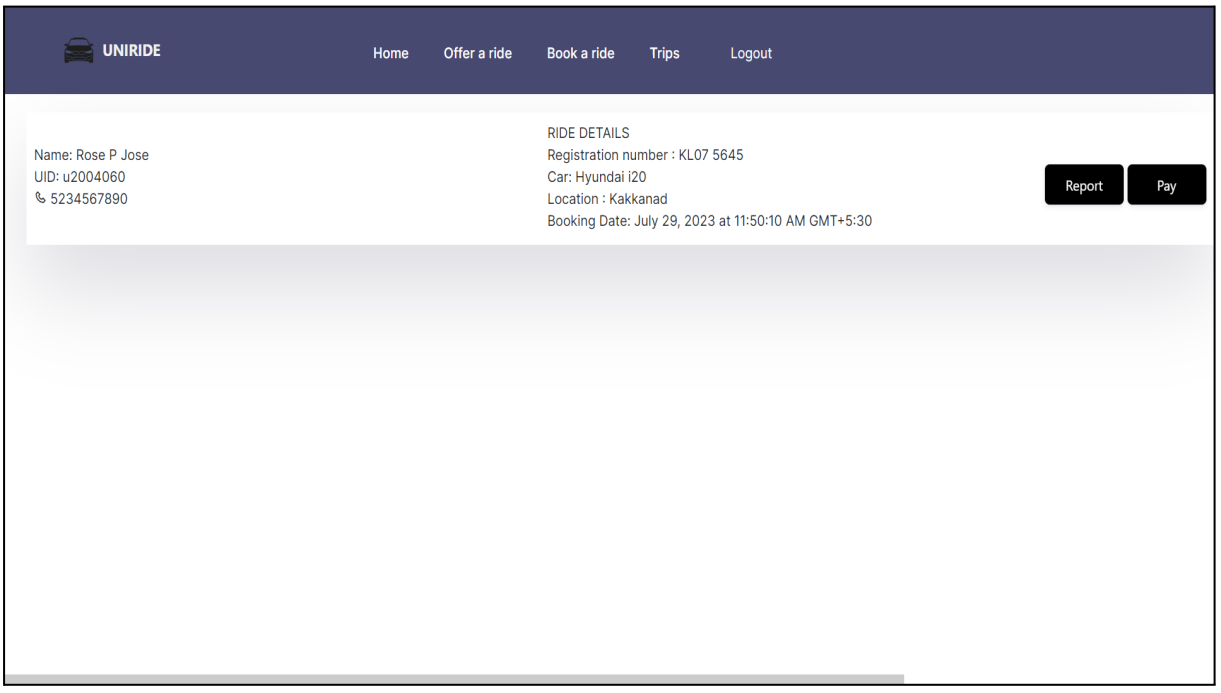

UNIRIDER
Home
Offer a ride
Book a ride
Trips
Logout
UNIRIDER
Connecting campus,
Sharing rides
About us

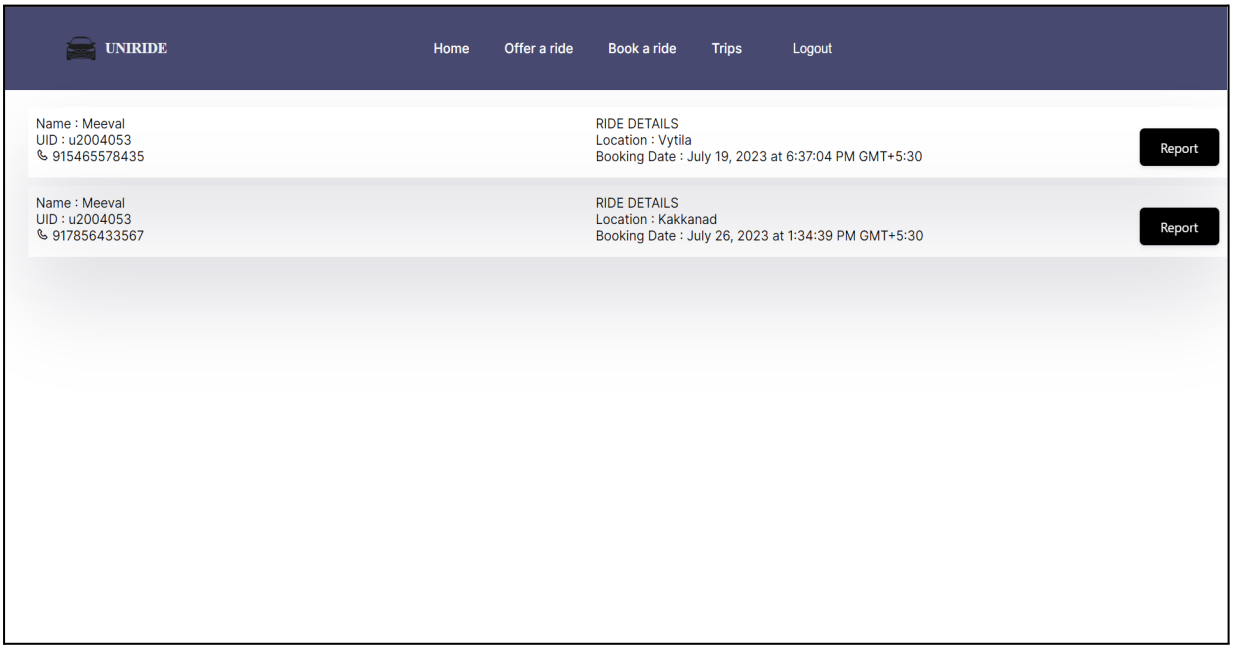

OFFER A RIDE
Name
Enter your name
UID
Enter your UID
Semester
Enter semester and branch
Phone number
Enter phone number
Gender
Female
Male
Other
Car Make and Model
Enter make and model
Registration number
Enter registration number
Upload vehicle
Choose File No file chosen
Upload license
Choose File No file chosen

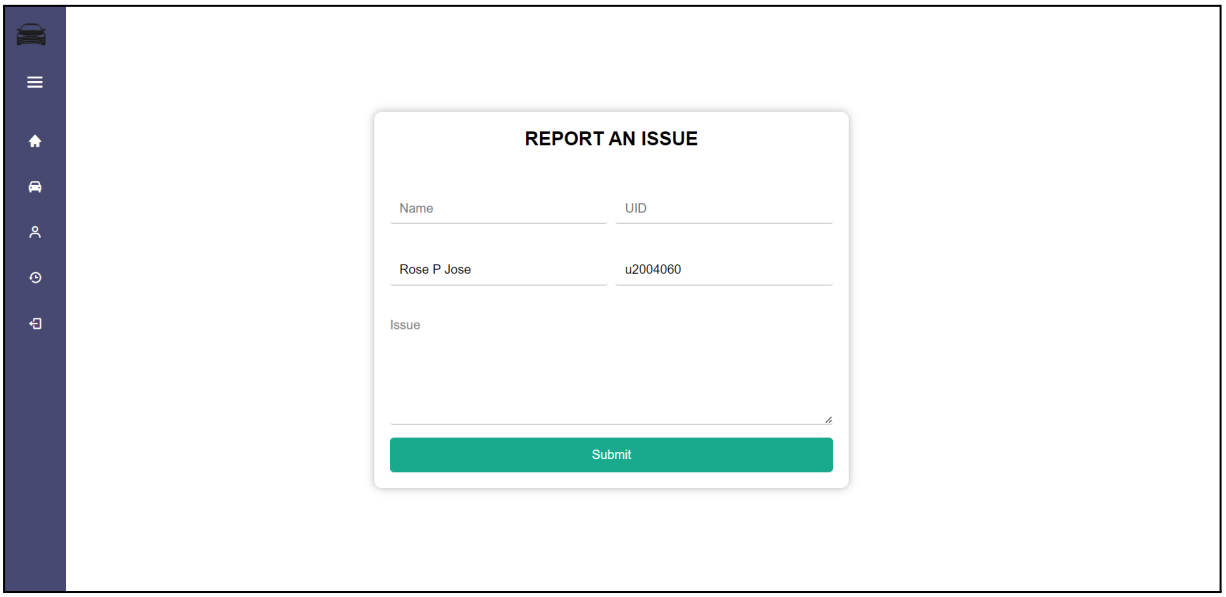

Turn off rides
UPDATE DETAILS
Number of Seats 3
Select an option
Pick-up from college
Drop to college
Route Kakkanad-Chembumukku
Submit
Cancel
0 mph
Haven't registered yet?

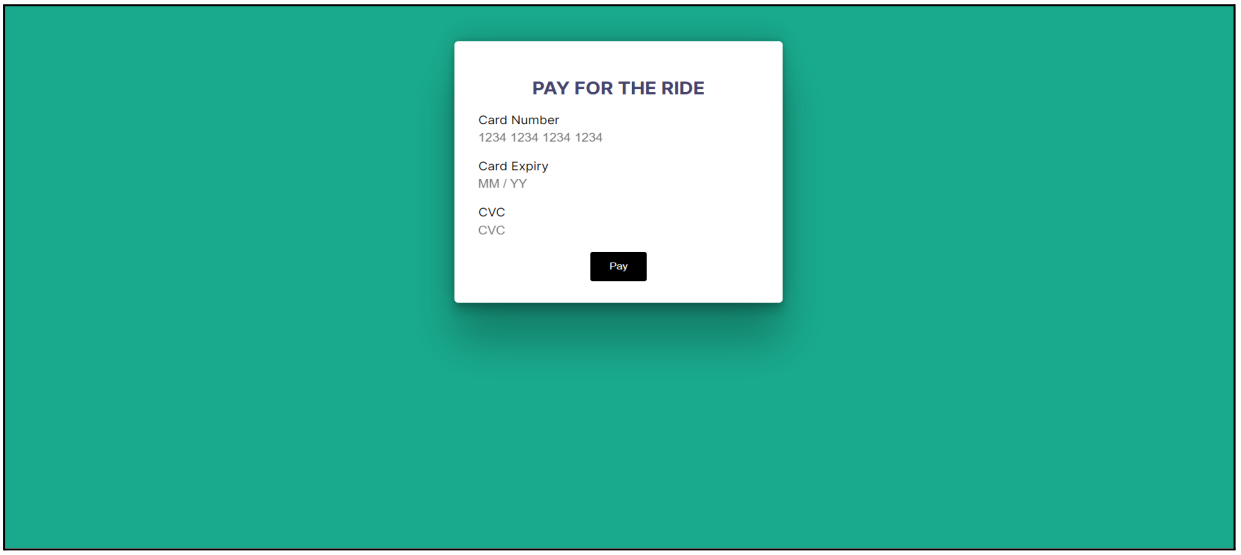

UNIRIDE
Home
Offer a ride
Book a ride
Trips
Logout
All
Enter a route
Drop to college
Filter
UID
Name
Location
Phone number
NAME UID SEM GENDER NUMBER SEAT ROUTE REGISTRATION NUMBER CAR PHOTO BOOK
Meekha Myno Babu u2004052 S6 IT Female 1234567890 3 Kakkanad-Chembumukku KL 07 5654 Toyota Land Cruiser Prado Book
Meeval u2004053 S6 IT Male 1234567890 3 Mavelipuram-Kakkanad KL 07 6543 Toyota Land Cruiser Prado Book
Rose P Jose u2004060 S6 IT Female 5234567890 1 Kakkanad-Vazhakkala-Chembumukku-Vytila KL07 5645 Hyundai i20 Book
Johan Abraham Tarun u2004043 S6 IT Male 6574435267 1 Kakkanad-Mavelipuram KL 07 2343 Toyota Land Cruiser Prado Book

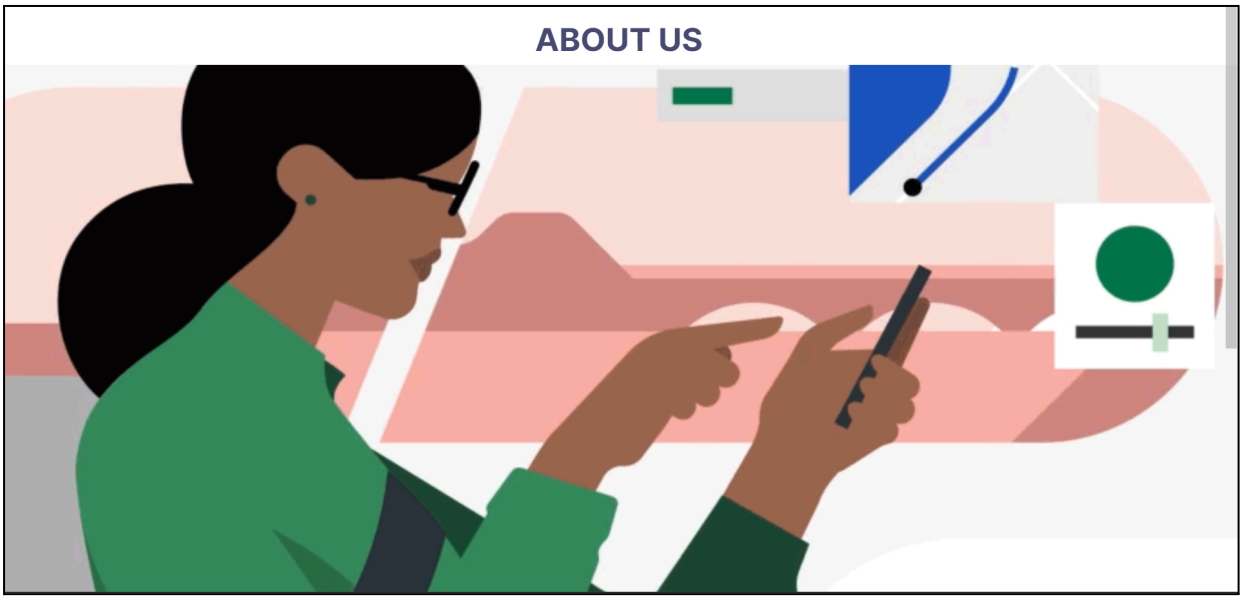

UNIRIDE
Home
Offer a ride
Book a ride
Trips
Logout
CONFIRM BOOKING DETAILS
DRIVER
Name: Meekha Myno Babu
UID: u2004052
1234567890
Registration Number: KL 07 5654
Car: Toyota Land Cruiser Prado
PASSENGER
Name: Rose P Jose
UID: u2004060
+915234567890
RIDE DETAILS
Chembumukku
40-60/-
8:00am
Confirm
Cancel

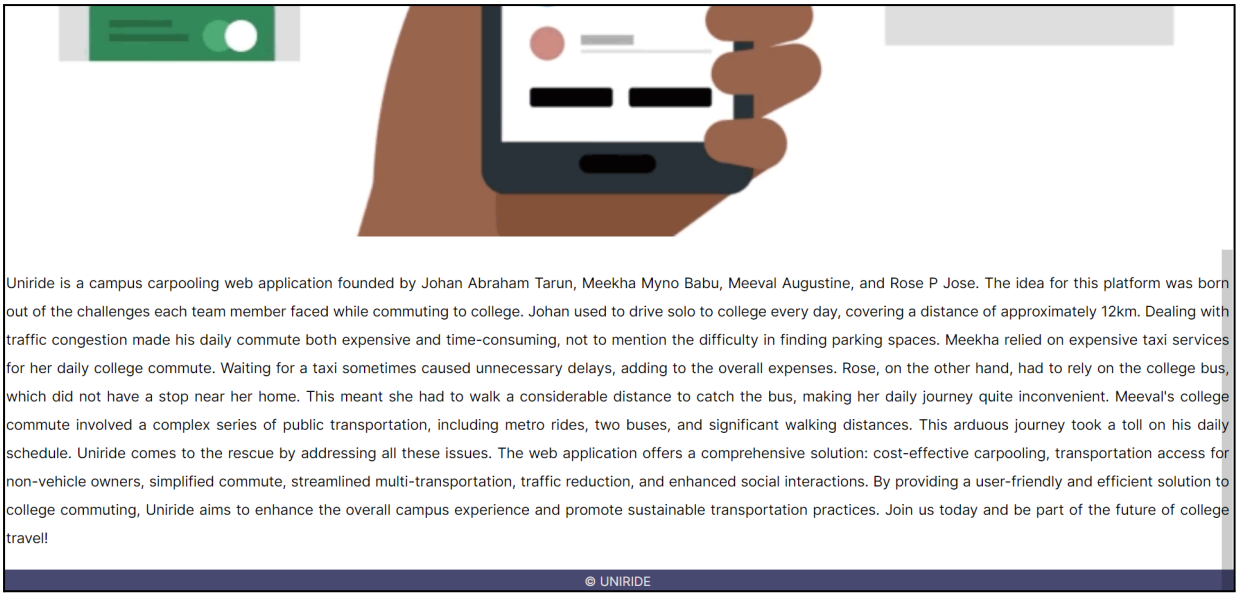

UNIRIDE
Home
Offer a ride
Book a ride
Trips
Logout
Name: Rose P Jose
UID: u2004060
5234567890
RIDE DETAILS
Registration number : KL07 5645
Car: Hyundai i20
Location : Kakkanad
Booking Date: July 29, 2023 at 11:50:10 AM GMT+5:30
Report
Pay


UNIRIDE
Home
Offer a ride
Book a ride
Trips
Logout
Name : Meeval
UID : u2004053
915465578435
RIDE DETAILS
Location : Vytila
Booking Date : July 19, 2023 at 6:37:04 PM GMT+5:30
Report
Name : Meeval
UID : u2004053
917856433567
RIDE DETAILS
Location : Kakkanad
Booking Date : July 26, 2023 at 1:34:39 PM GMT+5:30
Report

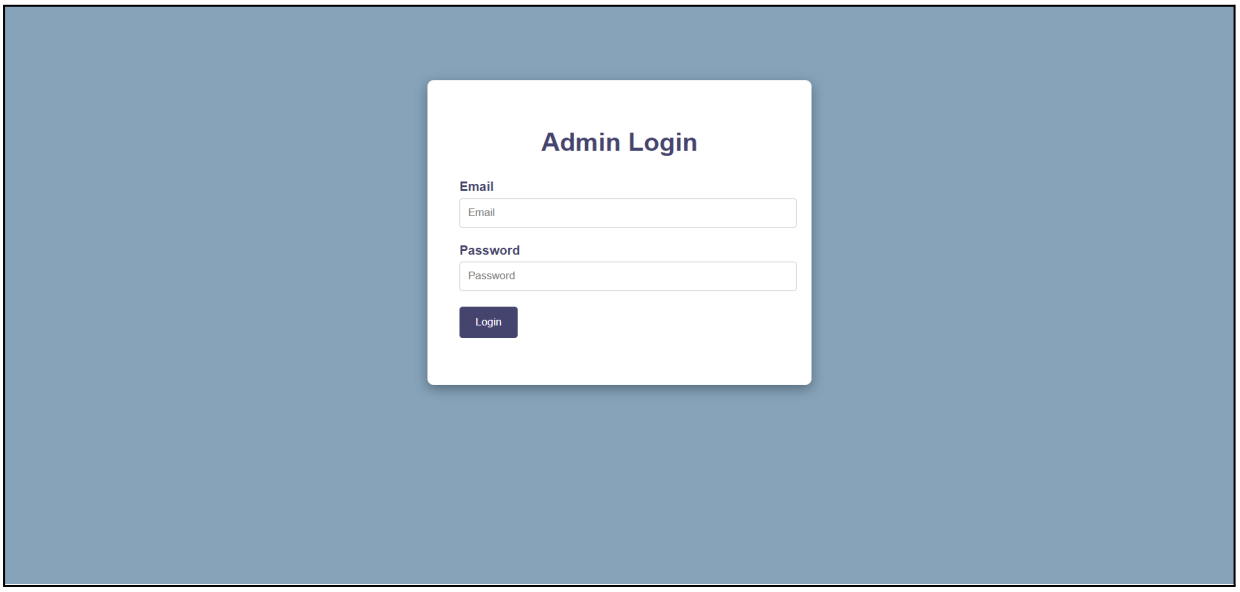

REPORT AN ISSUE
Name
UID
Rose P Jose
u2004060
Issue
Submit

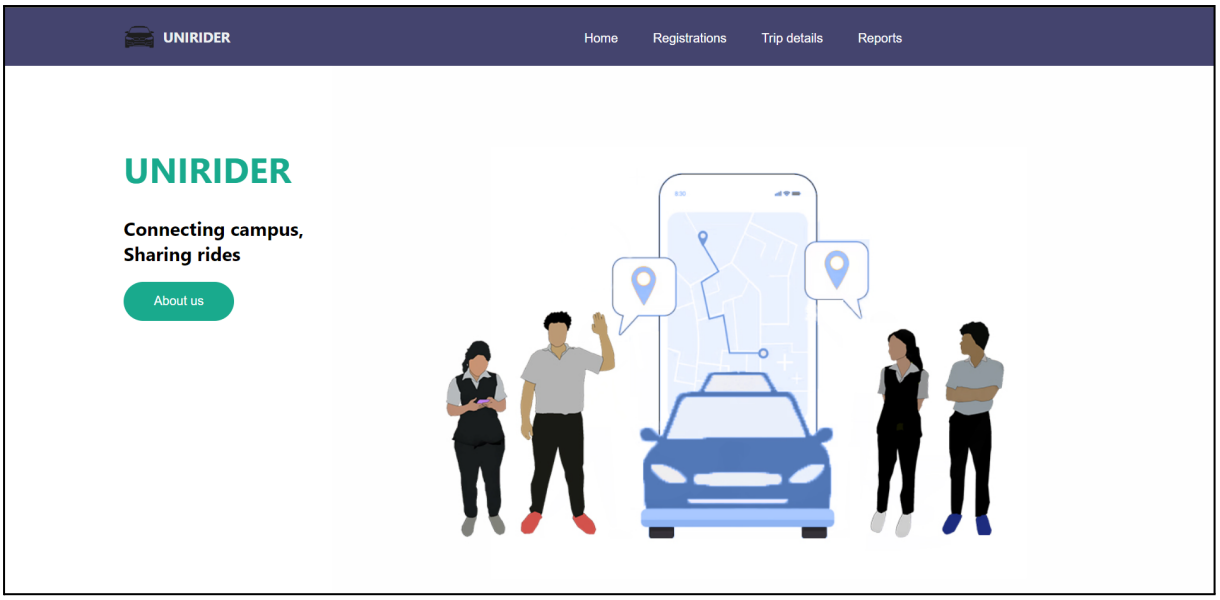

PAY FOR THE RIDE
Card Number
1234 1234 1234 1234
Card Expiry
MM / YY
CVC
CVC
Pay

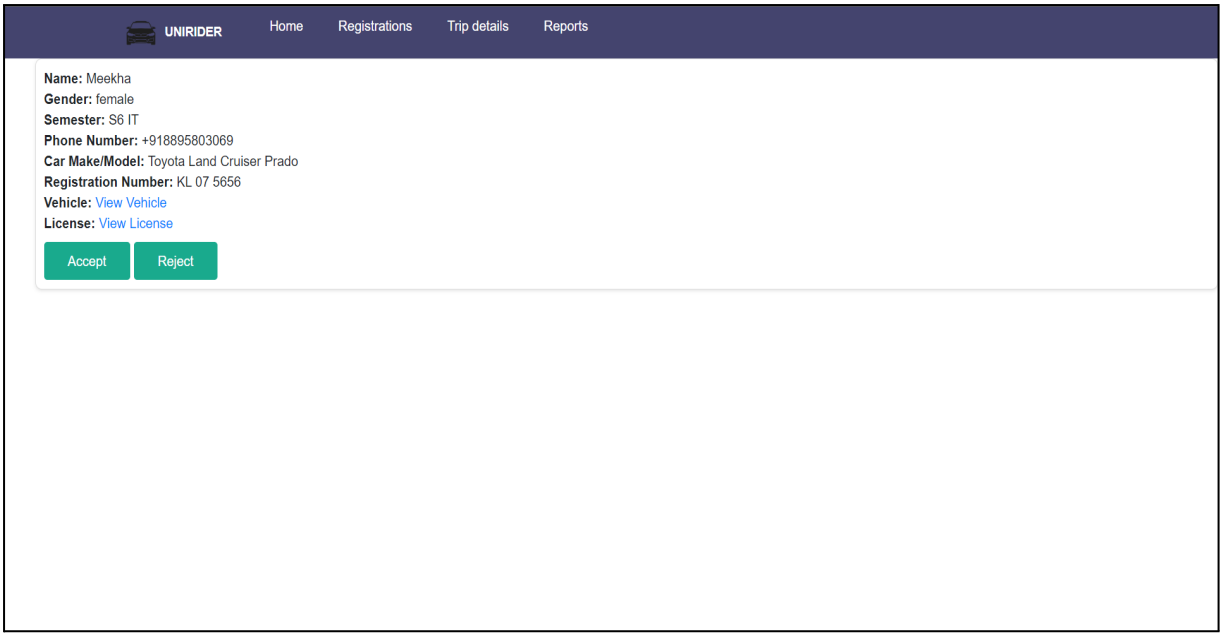

ABOUT US

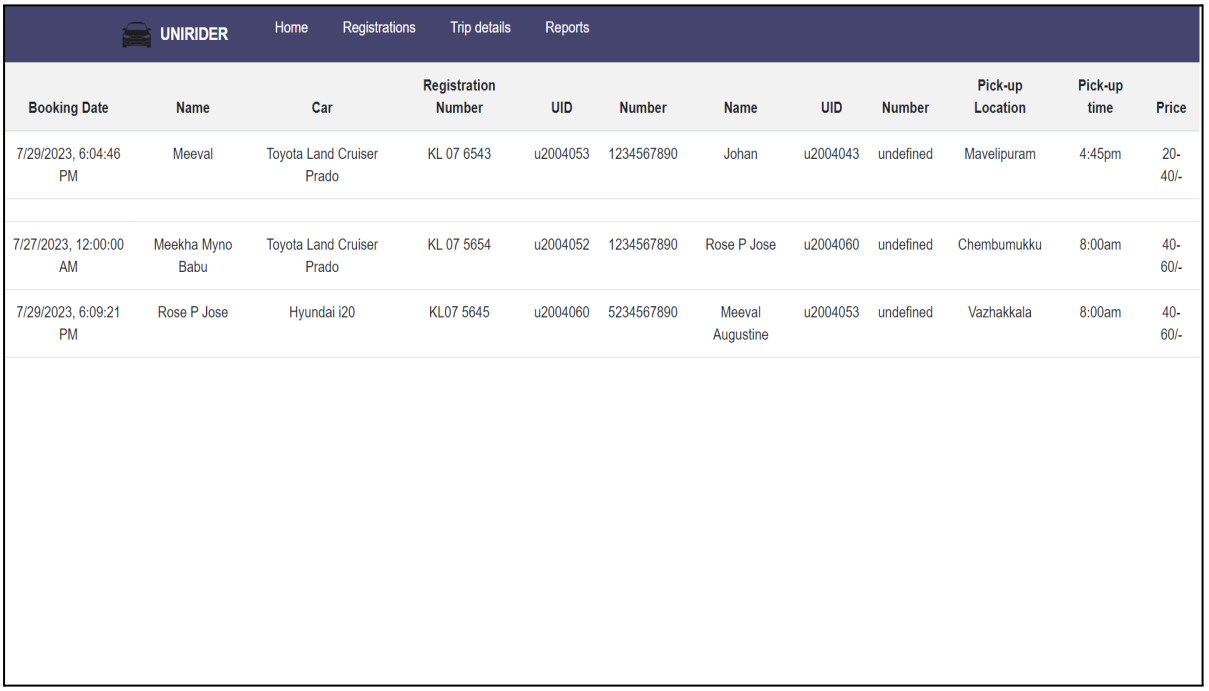

Uniride is a campus carpooling web application founded by Johan Abraham Tarun, Meekha Myno Babu, Meeval Augustine, and Rose P Jose. The idea for this platform was born out of the challenges each team member faced while commuting to college. Johan used to drive solo to college every day, covering a distance of approximately 12km. Dealing with traffic congestion made his daily commute both expensive and time-consuming, not to mention the difficulty in finding parking spaces. Meekha relied on expensive taxi services for her daily college commute. Waiting for a taxi sometimes caused unnecessary delays, adding to the overall expenses. Rose, on the other hand, had to rely on the college bus, which did not have a stop near her home. This meant she had to walk a considerable distance to catch the bus, making her daily journey quite inconvenient. Meeval's college commute involved a complex series of public transportation, including metro rides, two buses, and significant walking distances. This arduous journey took a toll on his daily schedule. Uniride comes to the rescue by addressing all these issues. The web application offers a comprehensive solution: cost-effective carpooling, transportation access for non-vehicle owners, simplified commute, streamlined multi-transportation, traffic reduction, and enhanced social interactions. By providing a user-friendly and efficient solution to college commuting, Uniride aims to enhance the overall campus experience and promote sustainable transportation practices. Join us today and be part of the future of college travel!
© UNIRIDE

ADMIN SIDE

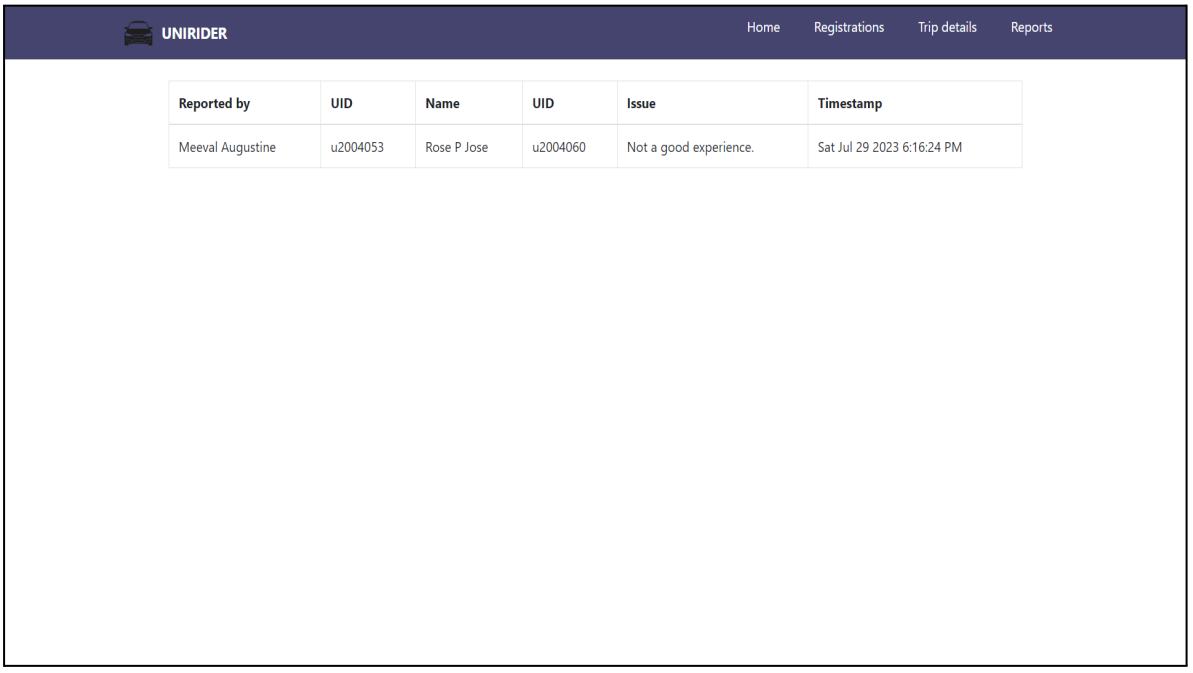

Admin Login
Email
Email
Password
Password
Login


UNIRIDER
Home
Registrations
Trip details
Reports
UNIRIDER
Connecting campus,
Sharing rides
About us


UNIRIDER
Home
Registrations
Trip details
Reports
Name: Meekha
Gender: female
Semester: S6 IT
Phone Number: +918895803069
Car Make/Model: Toyota Land Cruiser Prado
Registration Number: KL 07 5656
Vehicle: View Vehicle
License: View License
Accept
Reject



Booking Date	Name	Car	Registration Number	UID	Number	Name	UID	Number	Pick-up Location	Pick-up time	Price
7/29/2023, 6:04:46 PM	Meeval	Toyota Land Cruiser Prado	KL 07 6543	u2004053	1234567890	Johan	u2004043	undefined	Mavelipuram	4:45pm	20-40/-
7/27/2023, 12:00:00 AM	Meekha Myno Babu	Toyota Land Cruiser Prado	KL 07 5654	u2004052	1234567890	Rose P Jose	u2004060	undefined	Chembumukku	8:00am	40-60/-
7/29/2023, 6:09:21 PM	Rose P Jose	Hyundai i20	KL07 5645	u2004060	5234567890	Meeval Augustine	u2004053	undefined	Vazhakkala	8:00am	40-60/-



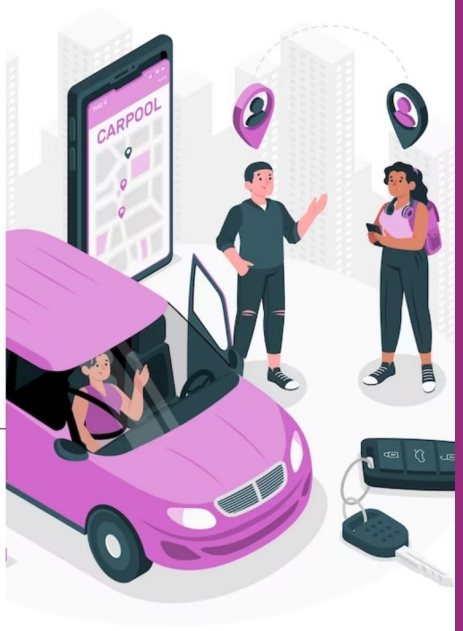
Reported by	UID	Name	UID	Issue	Timestamp
Meeval Augustine	u2004053	Rose P Jose	u2004060	Not a good experience.	Sat Jul 29 2023 6:16:24 PM

APPENDIX
SLIDES

UNIRIDE

CAMPUS CARPOOLING PLATFORM

Your Road to Sustainable Connections



Guided by
Ms.Viji Mohan
Assistant professor
DIT,RSET

Presented By
JOHAN ABRAHAM TARUN U2004043
MEEKHA MYNO BABU U2004052
MEEVAL AUGUSTINE U2004053
ROSE P JOSE U2004060

CONTENTS

Objective

The main objective of the campus carpooling platform project is to reduce traffic congestion, promote sustainable transportation, foster community connections, improve parking availability, enhance accessibility and affordability, and provide a user-friendly platform for convenient carpooling arrangements among campus members.

Applications/Relevance

- **Reducing traffic congestion**: By sharing a ride with other students, the number of cars on the road is reduced, which helps alleviate traffic congestion.
- **Cost savings:** Carpooling allows individuals to share the cost of gas and parking expenses, which can lead to significant cost savings.
- **Environment-friendly:** Fewer cars on the road mean reduced emissions, which can help improve air quality and reduce the carbon footprint of the campus.
- **Better use of parking spaces:** Carpooling can help optimize parking spaces on campus, especially during peak hours when parking spots are in high demand.
- **Social interaction:** Carpooling provides an opportunity for students to get to know each other and build a sense of community on campus.

Existing/Related Works

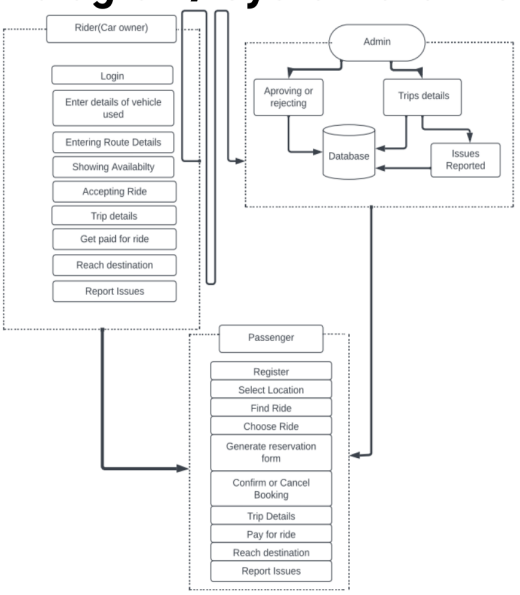
Uber: Uber is a popular ride-sharing platform that connects riders with drivers. It allows users to request rides, track their drivers, and make cashless payments.

BlaBlaCar: BlaBlaCar is a long-distance carpooling service that connects drivers traveling from one city to another with passengers heading in the same direction. It provides a platform for users to offer or book seats in shared rides.

Main components

Students/passenger	Car Owners	Admin

Block diagram/ System architecture



Modules names

1- ADMIN

- The Admin user can
- Approve or reject car registrations by accessing the data stored in the car_details database.
- View all the trip details and history stored in the bookings collection.
- View and handle all the issues reported by drivers and passengers stored in the reports collection.

Modules names

2- STUDENT DRIVER

The student driver user can

- Register themselves by entering their details into the registration_list collection in the real-time database.
- Log into the web application using the credentials provided by the driver upon registration which is stored in the registration_list database.
- Register themselves as drivers by entering details into the car-details collections inside Firestore.
- View their registration status by accessing details from the car-details collection.
- Update seat availability and route by entering details into the car-details collection.
- View trip bookings and history by accessing data stored in the trips collections.
- Report issues about the trip to the admin by entering necessary details into the reports collection.

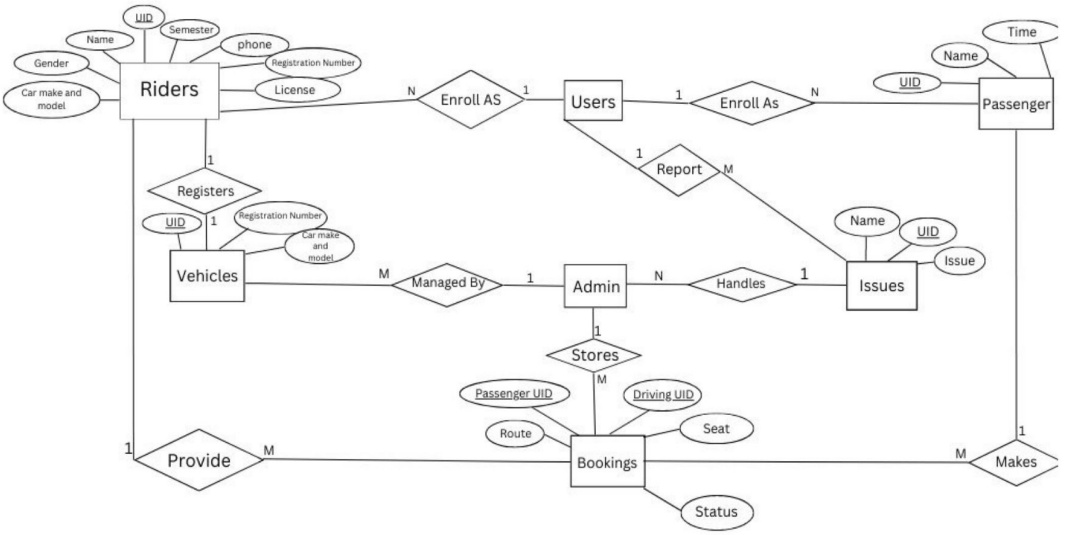
Modules names

3- STUDENT PASSENGER

The student passenger user can

- Register themselves by entering their details into the registration_list collection in the real-time database.
- Log into the web application using the credentials provided by the driver upon registration which is stored in the registration_list database.
- View the seat availability by accessing the details stored in the car_details collection and filtering it based on preferences such as gender and route.
- Confirm or cancel bookings and store details inside the bookings collection.
- View trip bookings and history by accessing data stored in the trips collections.
- Report issues about the trip to the admin by entering necessary details into the reports collection.

ER Diagram



Implementation details

HTML/CSS: Used for structuring and styling the web pages.

JavaScript: Used for client-side interactivity and dynamic functionality.

Firebase: widely used Backend as a Service (BaaS) platform provided by Google.It enable real-time data updates, handle user authentication, store user-generated content, execute backend logic with cloud functions, track analytics, send notifications, and provide hosting and deployment capabilities. These features enhance the platform's functionality, user experience, and scalability.

Node.js: for efficient server-side development, handling concurrent requests, scalability, leveraging a rich ecosystem of libraries, supporting real-time features with WebSockets, serving as a proxy server/API gateway, and enabling a microservices architecture.

Implementation details

VS Code : a lightweight and versatile IDE used for developing the campus carpooling platform. It offers a rich extension ecosystem, integrated terminal, IntelliSense, Git integration, debugging capabilities, and cross-platform support.

Database design

1. **User Data:** Create a "registration_list" collection to store user information, including fields such as names, email addresses, contact numbers, authentication credentials, and user preferences.
2. **Car registration Data:** Creates a "car details" collection to store both accepted and rejected registrations. Accepted registrations include seat availability and designated routes for ride-sharing arrangements. Rejected registrations are kept for record-keeping purposes.
3. **Booking Data:** Creates a "bookings" collection to track the bookings made by users. This collection can include details about driver as well as the passenger, booking time, location and price.

Database design

4. **Admin Data:** The admin data in the application includes details about registered vehicles, trip information, and user-reported issues.
5. **Trips data:** Contains details about all the current trips as well as the trip history with the option to pay.
6. **Reported Issue Data:** Creates a report collection to store reported passenger/rider details along with the issue faced.

APPENDIX D

RAJAGIRI SCHOOL OF ENGINEERING AND TECHNOLOGY (AUTONOMOUS)

DEPARTMENT OF INFORMATION TECHNOLOGY

PROGRAMME: INFORMATION TECHNOLOGY

VISION

To evolve into a department of excellence in information technology by the creation and exchange of knowledge through leading-edge research, innovation and services, which will in turn contribute towards solving complex societal problems and thus building a peaceful and prosperous mankind.

MISSION

To impart high-quality technical education, research training, professionalism and strong ethical values in the young minds for ensuring their productive careers in industry and academia so as to work with a commitment to the betterment of mankind.

PROGRAM EDUCATIONAL OBJECTIVES (PEO)

Graduates of Information Technology program shall

PEO 1: Have strong technical foundation for successful professional careers and to evolve as key-players / entrepreneurs in the field of information technology.

PEO 2: Excel in analyzing, formulating and solving engineering problems to promote life-long learning, to develop applications, resulting in the betterment of the society.

PEO 3: Have leadership skills and awareness on professional ethics and codes.

PROGRAM OUTCOMES (PO)

Information Technology program students will be able to:

PO 1. Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO 2. **Problem analysis**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO 3. **Design/development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO 4. Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO 5. Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

PO 6. The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO 7. Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO 8. Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO 9. Individual and teamwork: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO 10. Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO 11. Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO 12. Life-long learning: Recognize the need for and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

PROGRAM SPECIFIC OUTCOMES (PSO)

Information Technology program students will be able to:

PSO1: Acquire skills to design, analyze and develop algorithms and implement them using high-level programming languages**.**

PSO2: Contribute their engineering skills in computing and information engineering domains like network design and administration, database design and knowledge engineering.

PSO3: Develop strong skills in systematic planning, developing, testing implementing and providing IT solutions for different domains which helps in the betterment of life.

COURSE OBJECTIVES:

This course is designed for enabling the students to apply the knowledge to address the real-world situations/problems and find solutions. The course is also intended to estimate the

ability of the students in transforming theoretical knowledge studied as part of the curriculum so far into a working model of a software system. The students are expected to design and develop a software/hardware project to innovatively solve a real-world problem.

COURSE OUTCOMES:

After completion of the course the student will be able to

Sl.NO	DESCRIPTION	Blooms' Taxonomy Level
CO1	Make use of acquired knowledge within the selected area of technology for project development.	Level 3: Apply
CO2	Identify, discuss and justify the technical aspects and design aspects of the project with a systematic approach.	Level 3: Apply
CO3	Interpret, improve and refine technical aspects for engineering projects.	Level 3: Apply
CO4	Associate with a team as an effective team player for the development of technical projects.	Level 3: Apply
CO5	Report effectively the project related activities and findings.	Level 2: Understand

CO-PO AND CO-PSO MAPPING

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO 10	PO 11	PO 12	PSO 1	PSO 2	PSO3
CO1	3	3	3	3	3	3	3	3				3	3	3	3

CO2	3	3	3	3	3		2	3		3	2	3	3		3
CO3	3	3	3	3	3	2	3	3		2	3	3	2	2	2
CO4	3	3	2	2				3	3	3	3	3			
CO5	3				2			3	2	3	2	3			

3/2/1: high/medium/low

RSET
RAJAGIRI SCHOOL OF ENGINEERING & TECHNOLOGY
(AUTONOMOUS)

CLOVER: E- Commerce Book Service

MINI PROJECT REPORT

SUBMITTED BY

Ms. ANGELINA RAJU, U2004014

Mr. AJIL P Y, U2004007

Mr.ESSACK RAFEEK, U2004034

Ms. T A LAKSHMI, U2004065

DEPARTMENT OF INFORMATION TECHNOLOGY

CLOVER: E- Commerce Book Service

A MINI PROJECT REPORT

Submitted by

Ms. Angelina Raju

Mr. Ajil P Y

Mr. Essack Rafeek

Ms. T A Lakshmi

to

the APJ Abdul Kalam Technological University in partial fulfillment of the requirements of the award of the Degree

of

Bachelor of Technology

in

Information Technology

Department of Information Technology

Rajagiri School of Engineering & Technology

Rajagiri Valley, Kakkanad, Kochi-39

JULY 2023

DEPARTMENT OF INFORMATION TECHNOLOGY

RAJAGIRI SCHOOL OF ENGINEERING & TECHNOLOGY

(AUTONOMOUS)

RAJAGIRI VALLEY, KAKKANAD, KOCHI- 39

CERTIFICATE

This is to certify that the report entitled “Clover:E-commerce Book Service” is a bonafide record of the work done by Ms. Angelina Raju,Mr. Ajil P Y,Mr. Essack Rafeek,Ms. T A Lakshmi,University Register Numbers U2004007, U2004014, U2004034, U2004065 in partial fulfillment of the award of the Degree of Bachelor of Technology in Information Technology at Rajagiri School of Engineering & Technology, Kakkanad, Kochi during the academic year 2022-2023.

Dr. Nikhila T Bhuvan	Dr. Priya Mariam Raju	Dr.Neeba E A
Project Guide Assistant Professor Dept. of IT RSET	Project Coordinator Assistant Professor Dept. of IT RSET	Head of the Department Associate Professor Dept. of IT RSET

Submitted for the practical examination conducted on……

Internal Examiner

ACKNOWLEDGEMENT

A strong foundation is a necessity to step on and to start building and expanding. We are thankful for the management and our Principal,Dr. P. S. Sreejith who has ensured this strong foundation for us, students, to start building our lives on.

We thank Dr. Neeba E A, Head of the Department, Department of Information Technology, from the bottom of our hearts for being present at every stage of this journey to help and support us and make this project successful.

We also extend our sincere gratitude and appreciation to our mini project coordinators, Dr.Priya Mariam Raju, Asst. Professor, Department of Information Technology who coordinated us throughout the project.

This project would not have seen completion if it was not for the guidance from our guide, Dr. Nikhila T Bhuvan, Asst. Professor, Department of Information Technology, who shared her knowledge and advice throughout the project. We extend our sincere gratitude to her for being the great mentor she is.

Last but certainly not least we thank our teachers and friends whose help and support gave us the momentum to complete this work

ABSTRACT

An e-commerce book is a website which is mainly used for selling and buying any kind and genre of any book. The major problem faced by book readers is buying books at a high price and selling them at a lower price.To solve this problem, we have this website where you can buy books at an affordable price and sell them for the price the book is worth for. There are mainly four categories. After reading a new book, you can sell the book in the first category and get seventy percent of the book. The second category is the second hand books where the books have been used for a quite amount of time. The third category is books which are usable but can have some damages. The fourth category books are the ones which require recycling which can be given to the binman. When we buy these books, we can also refer to the reviews of the books in the review section. This way we can get a basic idea of what the book is about.The users of this website are the students who want to sell their old books to other students , for book readers and for research purposes. The benefits of this website are:

1. Affordable Prices
2. Fair Selling Prices
3. Diverse Categories
4. Sustainability and Recycling
5. Building learning Opportunities

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

1.1 GENERAL BACKGROUND

The world of books has long been dominated by the production and sale of new books, often leading to high costs for readers and considerable environmental impact. The relentless demand for new copies contributes to deforestation, excessive paper consumption, and mounting book waste. As a response to these challenges, the E-commerce Book Services project endeavors to challenge the status quo and transform the way books are bought and sold.

The project draws inspiration from the idea of creating a sustainable and affordable ecosystem for book lovers while ensuring that publishers and sellers can continue to operate profitably. By categorizing books based on their condition and promoting the reuse and recycling of books, the project envisions fostering a community of environmentally conscious readers.

Categorization System:

The cornerstone of the E-commerce Book Services project lies in its innovative categorization system. Books are divided into four distinct categories, each catering to different customer preferences and budgets:

Category A - New Books: This category comprises brand new books, sold at the full retail price, suitable for readers who prefer the pristine condition and the latest publications.

Category B - Second-Hand Books: Books that were initially categorized as Category A and were previously purchased by customers, fall into this category. These gently used books are priced at 80% of the original price, making them an attractive and affordable option for budget-conscious readers.

Category C - Readable Third-Hand and More: Books that have seen multiple owners and exhibit moderate wear and tear are placed in Category C. These books are priced at 60% of the original price, offering substantial savings while encouraging readers to focus on the

content rather than the book's physical condition.

Category D - Recyclable Books: This category includes books that might no longer be suitable for reading due to significant damage.

Cashback Policy:

The E-commerce Book Services project further incentivizes sustainable book consumption through a well-designed cashback policy for book returns:

Category A to B: If a customer purchases a book from Category A and returns it in a condition suitable for Category B, they receive a 45% cashback.

Category A to C: If the returned book is deemed suitable for Category C, the customer receives a 30% cashback.

Category A to D: If the book is recyclable and falls into Category D, the customer receives a 10% cashback.

Category B to C: For customers returning Category B books that can be categorized as Category C, they receive a 40% cashback.

Category B to D: If the returned book is suitable for Category D, the customer receives a 10% cashback.

Category C to C or D: Customers returning Category C books that are suitable for recycling in Category D or for Category C receive a 55% cashback.

1.2 PROJECT OBJECTIVE

The objective of the "E-commerce Book Services" project is to create an innovative online platform that promotes sustainable and affordable reading by categorizing books based on their condition and offering varying pricing structures. The project aims to encourage responsible book consumption, reduce waste, and incentivize readers to return and recycle books, fostering a greener and more eco-conscious approach to the book industry. Through the implementation of the categorization system and cashback policy, the project seeks to provide readers with diverse book options while ensuring profitability for sellers, ultimately creating a win-win situation for all stakeholders involved.

1.3 SCOPE

The scope of the "E-commerce Book Services" project is focused on creating an innovative online platform that promotes sustainable and affordable reading, offering benefits that extend beyond the obvious advantages. The project aims to revolutionize the traditional book-selling approach and create a win-win situation for both readers and sellers. The key aspects within the project scope include:

Accessibility and Convenience: The primary benefit of "E-commerce Book Services" lies in its accessibility and convenience for book lovers. Customers can easily access the platform from the comfort of their homes and choose from a diverse range of books categorized into four different categories based on condition and pricing.

Sustainable Book Consumption: The project's categorization system encourages readers to opt for second-hand books , reducing waste and promoting environmental sustainability. By offering affordable alternatives to new books, the project aims to reduce the overall environmental impact of book consumption.

Affordability: "E-commerce Book Services" offers an affordable and budget-friendly option for readers to enjoy their favorite literature. The pricing structure allows readers to make well-informed choices based on their preferences and financial constraints.

Cashback Policy: The cashback policy incentivizes readers to return books, contributing to the circular economy. Customers receive cashback based on the suitability of the returned book for other categories (B, C, or D), encouraging responsible book sharing and recycling.

User-Friendly Interface: The website will have an intuitive and user-friendly interface, enabling easy navigation, book search, and a smooth checkout process. User accounts will allow customers to manage their purchases, returns, and cashback status conveniently.

Promoting Community Engagement: The project aims to foster a sense of community among readers who share and cherish well-loved books. By exchanging books within the platform, readers can engage with like-minded individuals and create a vibrant book-sharing culture.

Win-Win for Sellers: The "E-commerce Book Services" platform provides a profitable opportunity for sellers to cater to a broader audience and effectively manage their book inventory. The project aims to create a sustainable ecosystem for both readers and sellers.

The "E-commerce Book Services" project aspires to go beyond the traditional book-buying experience, providing readers with an eco-conscious and budget-friendly alternative. The platform's accessibility, convenience, and commitment to sustainability create a unique value proposition, making it an indispensable solution for book enthusiasts and promoting positive change in the book industry.

CHAPTER 2

LITERATURE SURVEY

This chapter discusses the buying and selling of books at various platforms and how the system is being managed . It mainly describes the working of the system.

2.1 E-COMMERCE PLATFORM SYSTEM

Electronic commerce (e-commerce) refers to companies and individuals that buy and sell goods and services over the internet. E-commerce operates in different types of market segments and can be conducted over computers, tablets, smartphones, and other smart devices. Nearly every imaginable product and service is available through e-commerce transactions, including books, music, plane tickets, and financial services such as stock investing and online banking. As such, it is considered a very disruptive technology.

E-commerce has helped businesses (especially those with a narrow reach like small businesses) gain access to and establish a wider market presence by providing cheaper and more efficient distribution channels for their products or services. Target (TGT) supplemented its brick-and-mortar presence with an online store that allows customers to purchase everything from clothes and coffeemakers to toothpaste and action figures right from their homes.

2.1.1 LIMITATIONS

- Limited Customer Service: If you shop online for a computer, you cannot simply ask an employee to demonstrate a particular model's features in person. And although some websites let you chat online with a staff member, this is not a typical practice.
- Lack of Instant Gratification: When you buy an item online, you must wait for it to be shipped to your home or office. However, e-tailers like Amazon make the waiting game a little bit less painful by offering same-day delivery as a premium option for select products.

- Inability to Touch Products: Online images do not necessarily convey the whole story about an item, and so e-commerce purchases can be unsatisfying when the products received do not match consumer expectations. Case in point: an item of clothing may be made from shoddier fabric than its online image indicates.
- Reliance on Technology: If your website crashes, garners an overwhelming amount of traffic, or must be temporarily taken down for any reason, your business is effectively closed until the e-commerce storefront is back.
- Higher Competition: Although the low barrier to entry regarding low cost is an advantage, this means other competitors can easily enter the market. E-commerce companies must have mindful marketing strategies and remain diligent on SEO optimization to ensure they maintain a digital presence.

2.2 BOOK DATABASE SYSTEM

A comprehensive book database is essential for an e-commerce bookstore. It includes information about book titles, authors, ISBNs, descriptions, cover images, categories, and other relevant metadata. The database allows customers to search and filter books easily.
ISBNdb gathers data from hundreds of libraries, publishers, merchants and other sources around the globe to compile a vast collection of unique book data searchable by ISBN, title, author, or publisher. Get a FREE 7 day trial and get access to the full database of 33 plus million books and all data points including title, author, publisher, publish date, binding, pages, list price, and more.

ISBNdb Stats

Books: 34,717,253

Authors: 12,008,386

Publishers: 1,850,282

2.2.1 LIMITATIONS

1. Increased costs

- As database management systems require advanced hardware, software, and skilled employees, it is often associated with higher costs.
- The cost of maintaining the resources to operate a DBMS can include training, licensing, regulatory compliance, etc.
- DBMS also requires a high-speed processor as well as a large memory size to store safely and securely. They can be expensive solutions too.

2. Complexity

- To cover a lot of requirements and solve many data problems, the DBMS has complex functionality that makes it a complicated software.
- It is critical for developers, designers, and database users to have an appropriate skill set in order to use the database successfully and unlock its power.
- If they don’t understand the DBMS, then it may lead to loss of data or database failure.

3. Higher impact of a failure

- The fact that the DBMS is a central place for all of your data increases the vulnerability of the system.
- Since all users rely on one centralized place, the failure of any component can have a severe negative impact on operations or permanent damage to the database.

2.3 RELATED WORKS

2.3.1. Half Price Books

Half Price Books, Records, Magazines, Incorporated is a chain of new and used bookstores in the United States. The company's original motto is "We buy and sell anything printed or recorded except yesterday's newspaper", and many of the used books, music, and movies for sales in each location are purchased from local residents.

Half Price Books publishes some of the books it sells, inexpensively reprinting non-copyrighted titles or acquiring the U.S. or English language rights from another publisher. Half Price Books reprints these titles under its publishing arm, Hackberry Press.

Among Hackberry Press titles is the Half Price Books children's storybook series Say Good Night to Illiteracy, with 13 editions in print. All proceeds from the series benefit family literacy organizations such as Reach Out and Read and the National Center for Family Literacy project was axed in 2005.

2.3.2. Alibris

Alibris is an online store that sells new books, used books, out-of-print books, rare books,and other media through an online network of independent booksellers.

Booksellers list their inventories on Alibris which in turn offers the books on its retail website, a separate library services site, and business-to-business partners such as Barnes & Noble, Half Price Books, and eBay. It also offers services in the UK through the Alibris.co.uk website. It offers more than 250 million books from a network of over 6000 booksellers in 65 countries.

Most sales made through Alibris are fulfilled by the bookseller directly to the end customer. Sales to libraries or other institutions or books needing transoceanic shipping are consolidated in a distribution center in Sparks, Nevada. Alibris also has a similar network for music (albums, cassette tapes, and CDs) and movies (VHS or DVD).

Alibris allows customers to buy and sell at the same time. Alibris charges a starting fee that varies based on what is being sold and what kind of commission is charged.
Alibris was a charter member of the Google eBooks service when it was announced by Google on December 6, 2010.

2.3.3. AbeBooks

AbeBooks is an e-commerce global online marketplace with seven websites that offer books, fine art, and collectables from sellers in over 50 countries. Launched in 1996, it specializes in used, rare and out-of-print books. AbeBooks has been a subsidiary of Amazon since 2008.

AbeBooks is known as an online marketplace where high-end rare and collectible items change hands. In February 2015 it recorded the most expensive sales ever at the time: a rare illustrated ornithology book was sold online for $191,000. The website periodically reports their recent high value sales.

Sellers pay a monthly subscription to list their books on the site, ranging from $25 to $500, depending on how many books they list. This subscription fee has been in place since at least April 2008. In addition, sellers pay a percentage fee for each book sold via the websites.

2.3.4. BookChor

BookChor is an online platform used for buying and selling used books in India. The website serves as a marketplace where book enthusiasts can browse through a diverse collection of second-hand books across various genres and purchase them at discounted prices.

Users can explore the BookChor website to find a wide range of pre-owned books, including fiction, non-fiction, academic, self-help, and more. The books are often sold at lower prices compared to new copies, making it an affordable option for readers to expand their collection and read more without breaking the bank.

2.3.5. Amazon Books

Amazon Books was a chain of retail bookstores owned by online retailer Amazon. The first store opened on November 2, 2015, in Seattle, Washington. On March 2, 2022, it

was reported that all Amazon Books would close on various dates in the future.

2.3.6 LIMITATIONS

- Affordable Prices: The given websites are not providing affordable prices as much as we provide. We provide maximum profit with affordable prices.
- Fair Selling Prices: The books that we sell are sold in different categories and these categories have fair selling prices. These are provided in the above websites.
- Diverse Categories: As I mentioned above, there are different categories such as class A, class B, class C and class D. These categories each have different prices and are affordable by the users
- Sustainability and Recycling: The category D is used to recycle the books which have been used multiple times and are worn and torn. These books are recycled which provide for a sustainable environment.
- Building learning Opportunities: Our website is very useful for learning students who can afford these books at a low price and can get a cashback. This way the students are encouraged to learn and afford books at a fair price. The above given websites don’t provide this benefit.

CHAPTER 3
MODELLING/EXPERIMENTATION

3.1 SYSTEM ANALYSIS AND DESIGN

System analysis and design for an e-commerce bookstore involves a structured approach to understanding, defining, and designing the various components and functionalities of the online platform. The process encompasses several stages, each aimed at ensuring that the e-commerce bookstore meets the needs of its users and operates efficiently.

3.1.1 SYSTEM ARCHITECTURE

The proposed Clover has two user types- Admin and User(Customer). Figure 3.1.1 shows that the admin can manage all other users in the system and can also view, add and delete users.The users can signup and later login to the site. The user, then, can select the necessary books and add them to the cart. Now, the book is sold to the admin directly. If the user wants to return the book, they can directly access the shop and return the book. The admin will check the condition of the book and add the book to the appropriate class.

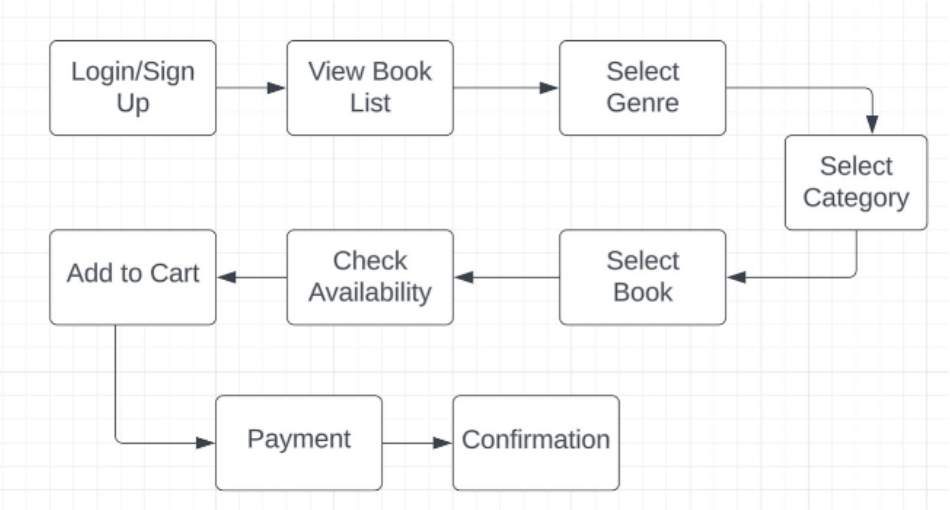


Figure 3.1 System Architecture

3.2 MODULES

There are mainly 5 modules:

- Module 1- Customer
- Module 2- Admin
- Module 3- Book
- Module 4- Category
- Module 5- Cart

3.2.1 MODULE 1- CUSTOMER

The Customer user can:

- Sign up/login in
- Edit profile
- View book
- Select category
- Select genre
- Add to cart

3.2.2 MODULE 2- ADMIN

The Admin user can

- Manage book price
- Add books to each category according to the conditions.
- Keep track of the book in the cart.

3.2.3 MODULE 3- BOOK

The book user can

- Select different categories
- Each has different genre
- Price will be given according to the categories

3.2.4 MODULE 4- CATEGORY

There are mainly 4 categories:

1. Class A:
    - The books are new and not being reused.
    - It is selled in original price.
2. Class B:
    - The books have been used once.
    - Sold at 80% of original price
3. Class C:
    - The books have been used multiple times and there will be some damages.
    - Sold at 60% of original price
4. Class D:
    - The books have been used too many times and have to be recycled.

3.2.5 MODULE 5- CART

The cart page contains:

- Contains the orders.

- This is monitored by the admin.

3.3 TABLE DESIGN

The given figure 3.7, shows the database design that is used in the project.The following is the detail of each of the databases used.

Sign_up

The sign up database contains the name, number, address, email, create_pass and conf_pass of the user. These values will be stored in the database when the user signs up into the website. The email is the primary key.

Login

The login page is connected to the sign up database. The user can log into the home page using the email and password which was given in the sign up page. Here, we are using the email from the sign up page as a foreign key as reference.

Book

The book database contains the Book_name, Image Path, Author_name, Book_price, Genre and Category. The Book_id is given by the mongodb database, which is the primary key.

Genre

The genre page contains three types of genre, Fictional Fantasy, Novel and Tragedy. The genre is connected to the Book database and will add the books to the specific page automatically. The genre is a foreign key.

Category

The category pages are of three types: Class A , Class B and Class C. Each of these classes is connected to the book database, ie, the category is a foreign key here.

Cart

The cart contains the book title, Book Quantity, Book_price, Image Path and the total_price. These values are acquired from the book database after certain calculations.

Admin

The admin handles the book_payment, the Upload of the books and the return of the books. The admin uses the Book_title to do so. The book_id is the primary key.

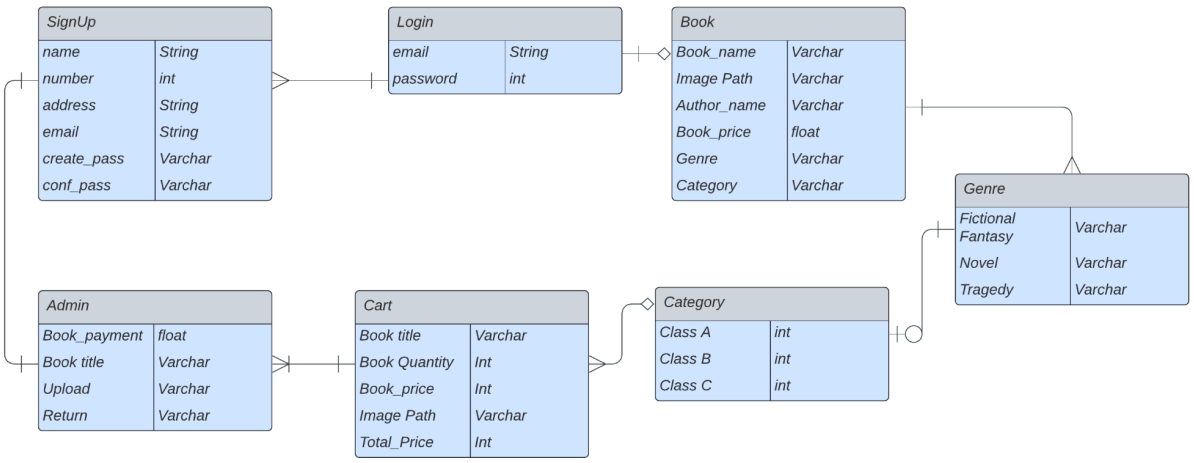


Figure 3.2 Depicts the Database design

3.4 E R DIAGRAM

In the given figure 3.8, shows the ER Diagram which is used to implement the website. The following is the detailed description of the diagram:

1. **Entities:**

   The entities in the entity diagram are:

   - Customer
   - Sign up
   - Login
   - Book
   - Admin
   - Category
   - Genre
   - Cart

2. **Attributes:**

   The attributes for each of the entities are:

   2.1 Customer has no attributes.

   2.2 Sign up entity has the attributes: Name, email, Phone, Address, create password and confirm password.

   2.3 Login entity has the attributes: email, Password.

   2.4 Book entity has attributes: Book_id, Book_title, imagePath,price,author,cname,genre.

   2.5 Category entities have attributes: Class A,Class B,Class C, and Class D.

   2.6 Genre entity has these attributes: Novel, Fictional Fantasy and Tragedy

   2.7 Admin entity has attributes: Book_id, Book_title, imagePath,price,author,cname,genre.

   2.8 Cart: Book_title, Price, Quantity, imagePath and Total Price.

3. **Relationship between the entities and attributes:**

   The relational between each of the entities are:

   - Customer is related to the signup which means the customer has to sign up first then only they can access the website.

- The sign up is related to the login entity, which means after the user has signed into the website, can use the email and password to login to the website.
- The customer can select the books. The book entity contains the category and the genre entity.
- The admin can return the books and upload the books to the Book entity.
- The customer can order a Book to the Cart. The cart is accessed by the Customer.

4. Primary Keys:

The primary key for each entity are:

- Sign up: email is the primary Key.
- Login page: has the foreign key which is used from the sign up entity.
- Book : Book_id is the primary key.
- The entities have the Book_id as their foreign key.

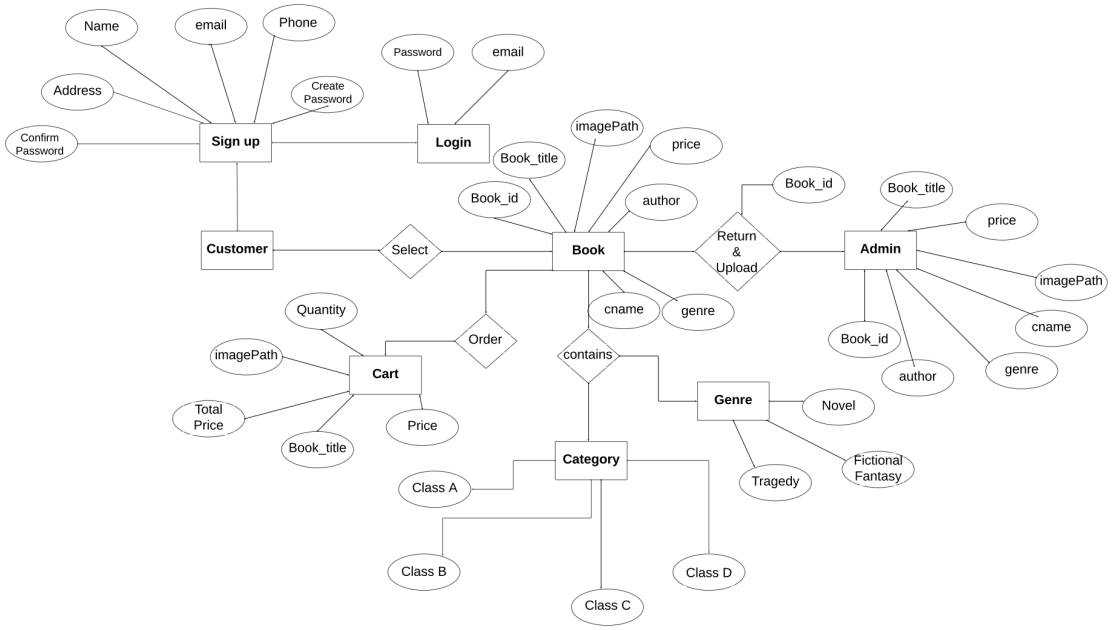


Figure 3.3 Depicts the E R Diagram

3.5 IMPLEMENTATION REQUIREMENTS AND SCHEDULE

3.5.1 HARDWARE REQUIREMENTS

The selection of hardware configuration is a very important task related to software development, particularly inefficient RAM may adversely affect the speed and correspondingly the efficiency of the entire system. The processor should be powerful to handle all the operations. The hard disk should have sufficient capacity to solve the database and the application. The network should be efficient to handle the communication fast.

1. CPU: Intel(R) Core(™) i5-10210U Processor
2. Memory: 8GB
3. Cache: 1017 MB
4. HardDisk: 239 GB
5. Display: 1920 x 1080
6. Mouse: Lenovo

3.5.2 SOFTWARE REQUIREMENTS

1. Operating System: WindowsXP 11
2. Front End Tool: HTML, CSS, JavaScript
3. Back End Tool: Node.js and Express.js
4. IDE: Visual Studio Code

CHAPTER 4
RESULTS AND DISCUSSION

4.1.CASHBACK CALCULATION SYSTEM

The technical aspect of the cashback policy involves developing a sophisticated cashback calculation system that automatically determines the appropriate cash back percentage for returned books based on their condition and suitability for other categories. The system efficiently processes book returns and provides cashback to customers, enhancing user experience and satisfaction.

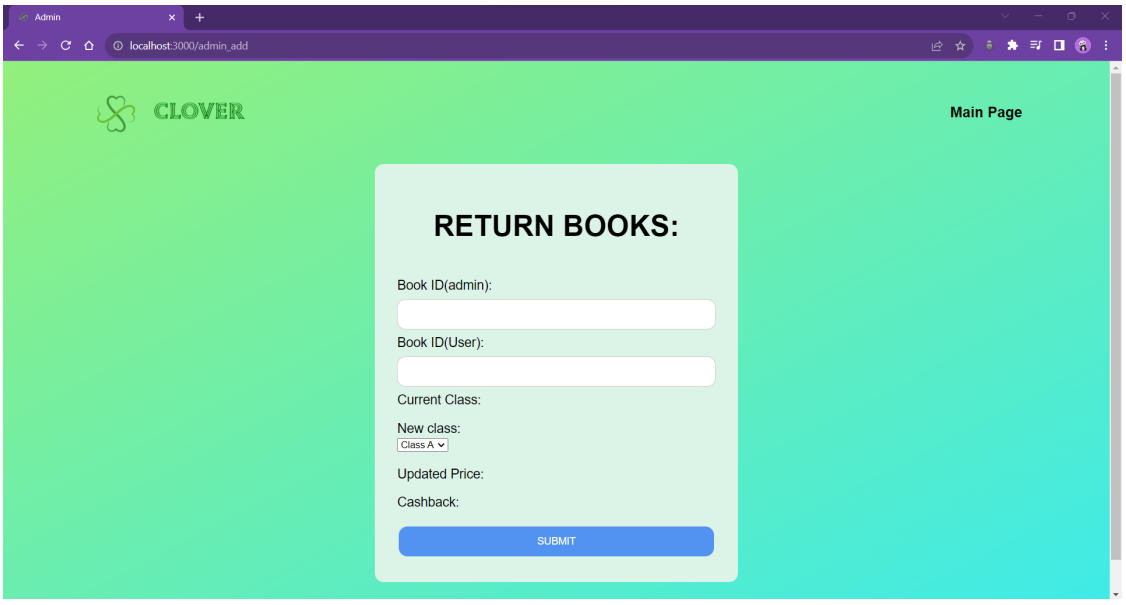


Figure 4.1 Return page

The above figure shows the return of the books. Here, the administrator can return the books and change the category of the book according to the current class. The updated price of the book and the cashback to the user will be calculated and displayed when the class is updated.

4.2. UPLOAD SYSTEM

The administrator can upload books to the user's page and to the admin's page easily.

This is implemented using a book_id. The administrator can upload a set of books using a form as shown in figure 4.2 and figure 4.3.

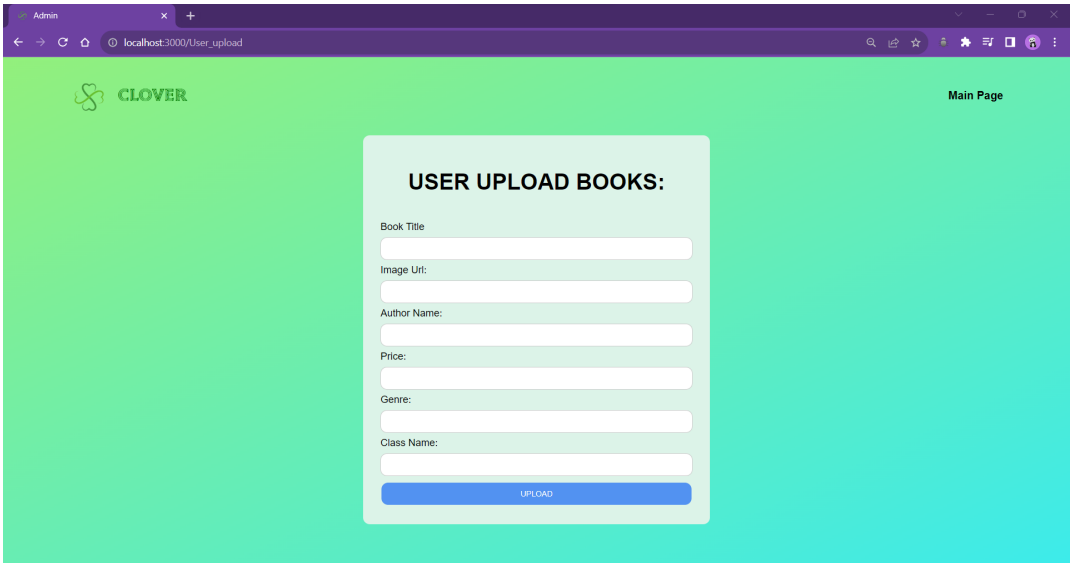


Figure 4.2 User's upload page

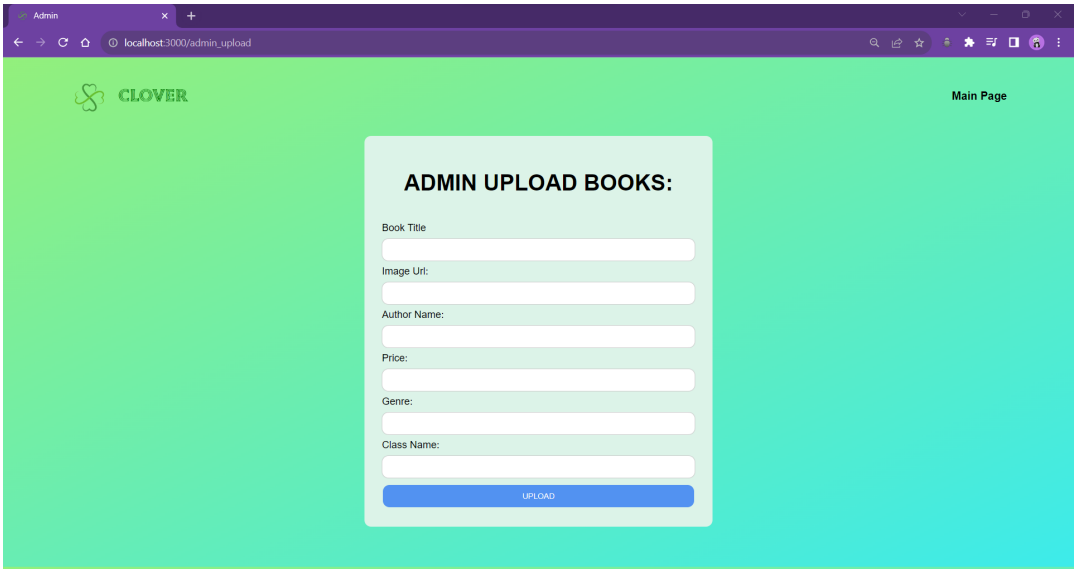


Figure 4.3 Admin's upload page

4.3. USER ACCOUNT MANAGEMENT:

The project's technical success includes the development of a secure and user-friendly user account management system. This feature allows customers to create accounts and order books.

The following figures show how the user can sign up and login into the system. The user can check their profile and also change their password.

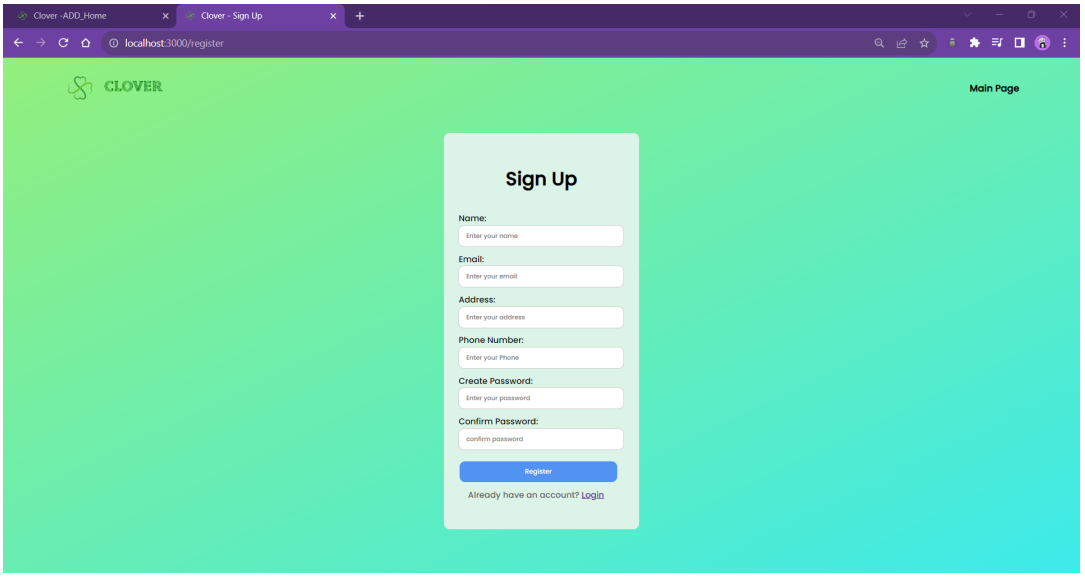


Figure 4.4 Sign up page

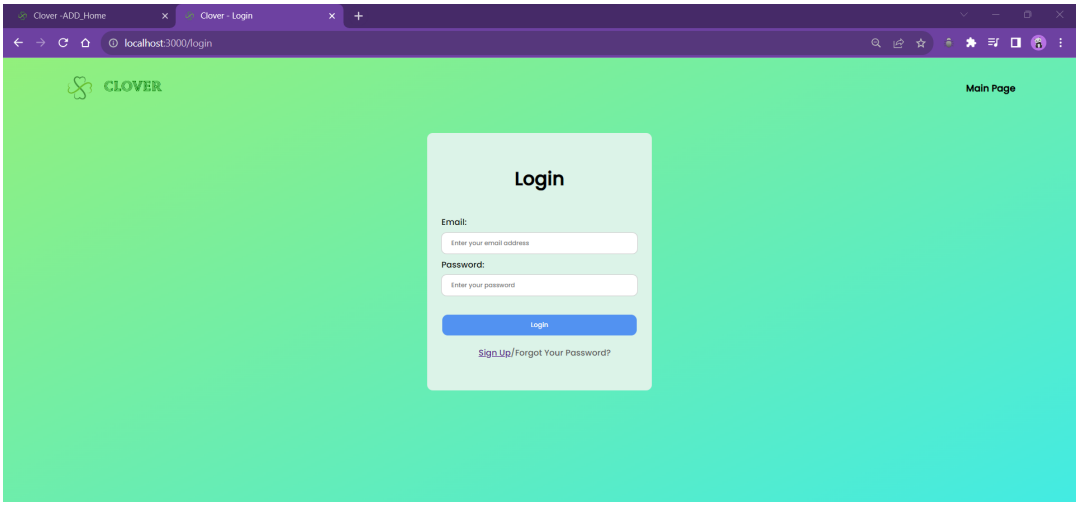


Figure 4.5 Login page

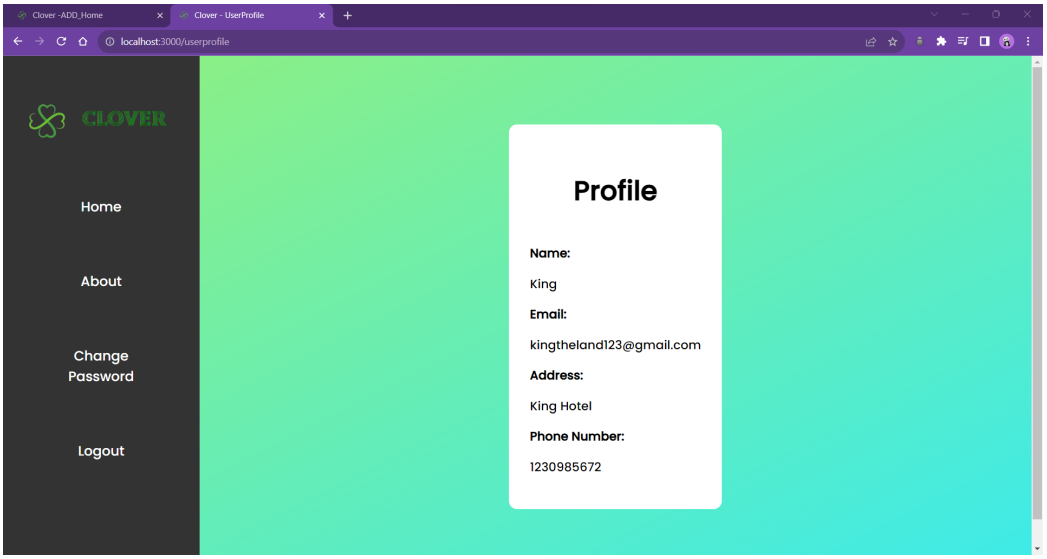


Figure 4.6 Profile page

4.4.GUI DESIGN

The GUI design features that are mainly used are bootstrap 3.4.1 and the shopping cart has a proper listing of the images, title and prices of the books. The Navigation menu which is implemented contains all the necessary information and links to the pages.

4.4.1. Display of login and signup in the main page

The customer can easily access the signup page and the login page which are displayed in the main page.

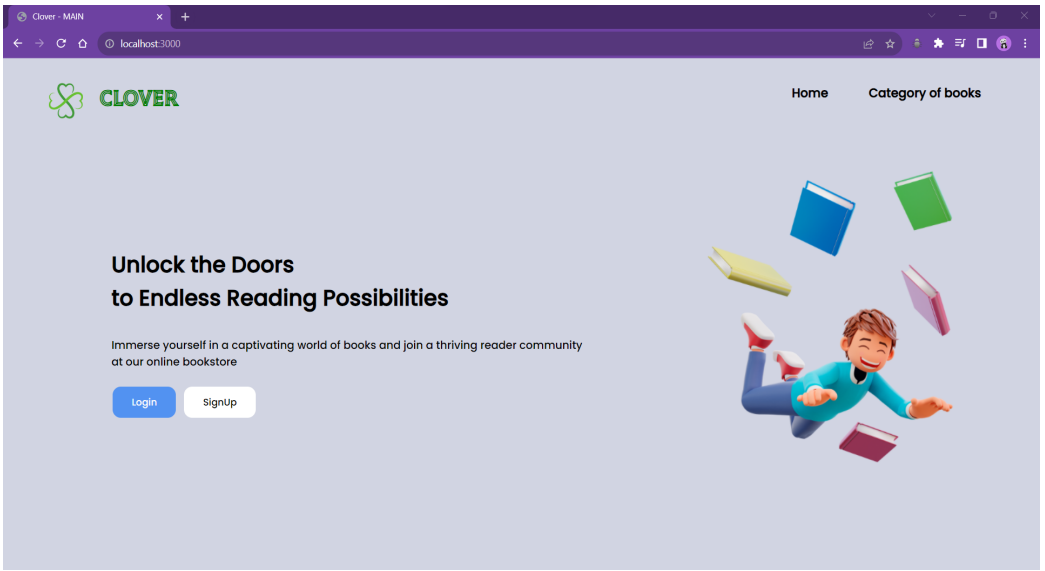


Figure 4.7 Mainpage

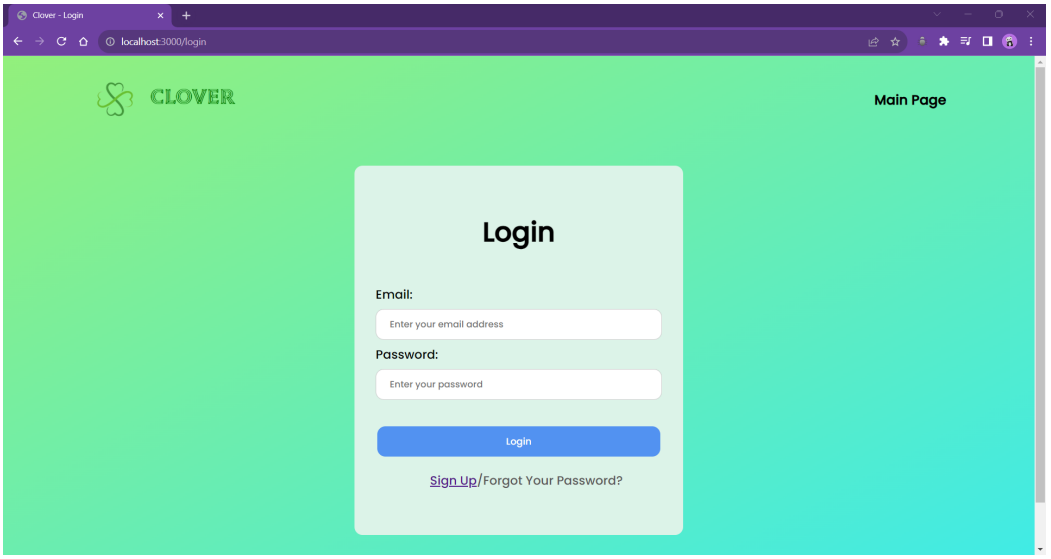


Figure 4.8 User's login page

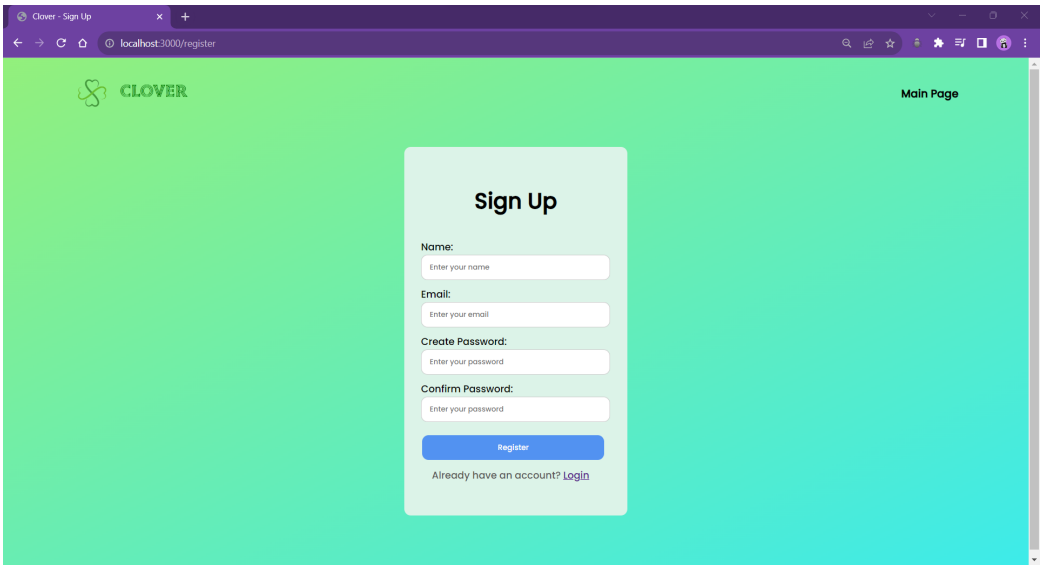


Figure 4.9 User's sign up page

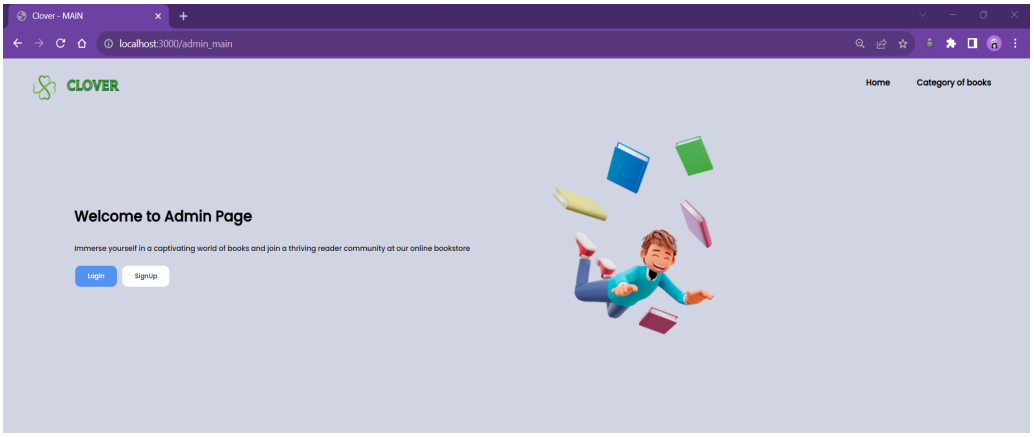


Figure 4.10 Admin's MainPage

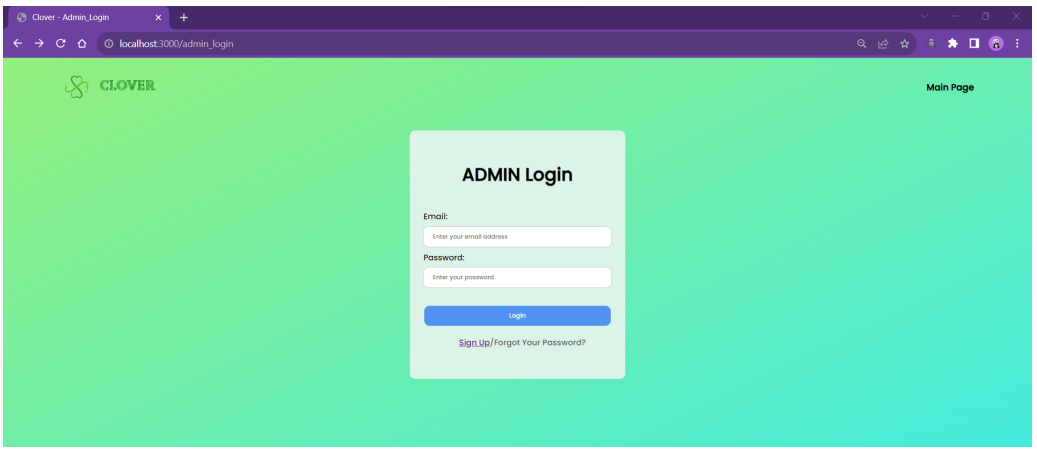


Figure 4.11 Admin's Login page

4.4.2. The Navbar

Figure 4.12 depicts the navbar which is common for almost all of the pages. The user can access the main page, home page, genre, category, their accounts and can logout to the main page.

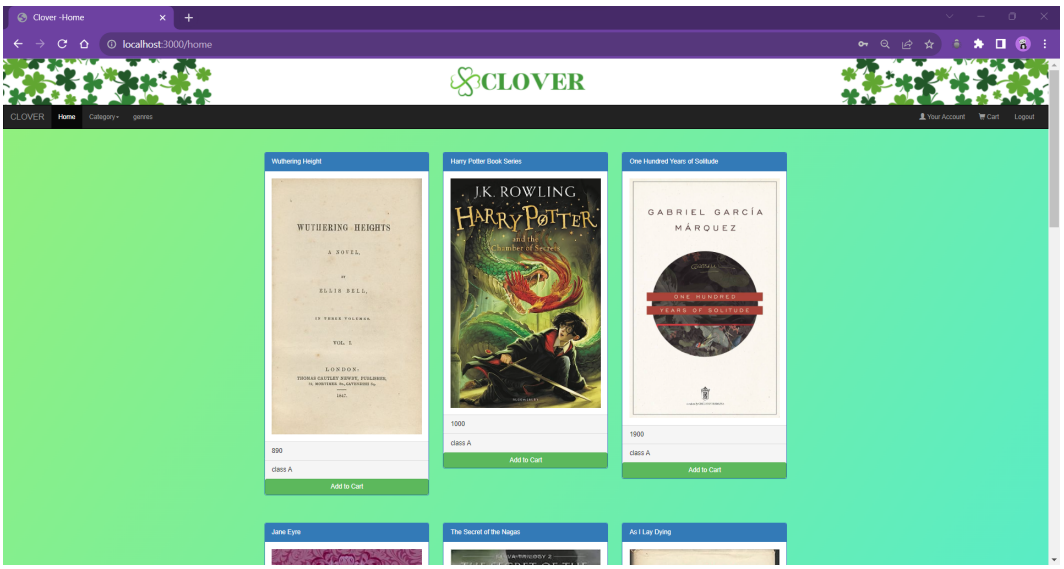


Figure 4.12 Navbar

4.4.3. Cart

Figure 4.13 depicts the home page which has access to the cart. The users can add their book from the home page and also access the cart at the same time to check if the book is added to the cart.

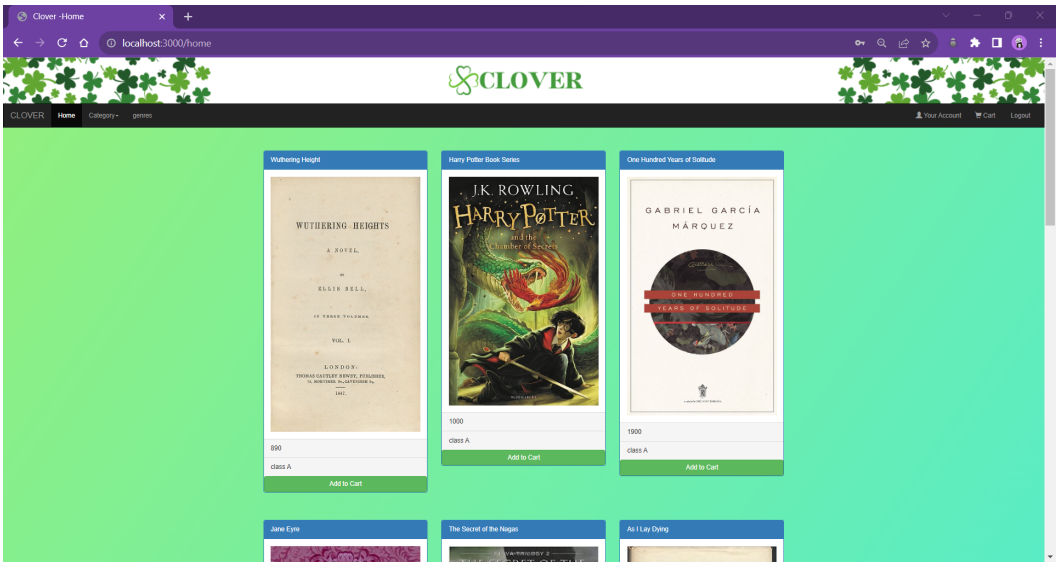


Figure 4.13 Cart

4.4.4. User Profile and Logout

Figure 4.14 depicts the logout and password change for a user. The user can change their account details in the user profile page. The user then can also logout accordingly. This will be later accessed by the admin, which is accessed by the admin page.

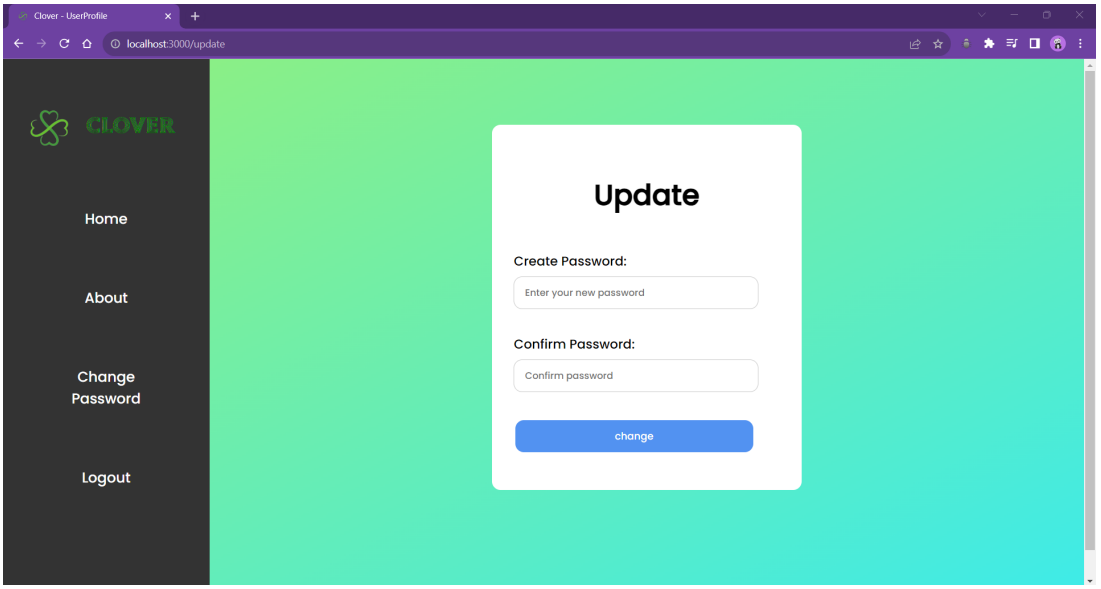


Figure 4.14 Logout and password change for a user

4.5. CART MANAGEMENT:

The user's can easily add the item to the cart. The figure 4.15, shows when the cart is empty. The figure 4.16, shows when the user has added the item. When the user clicks proceed to checkout, a mail will be send to the user and the admin as shown in figure 4.18 and figure 4.17

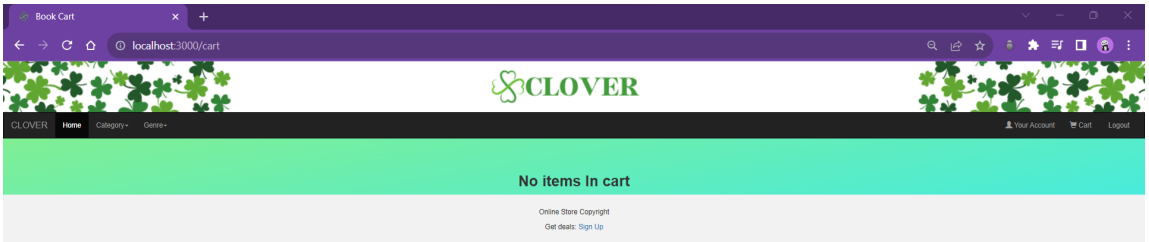


Figure 4.15 Cart page without items

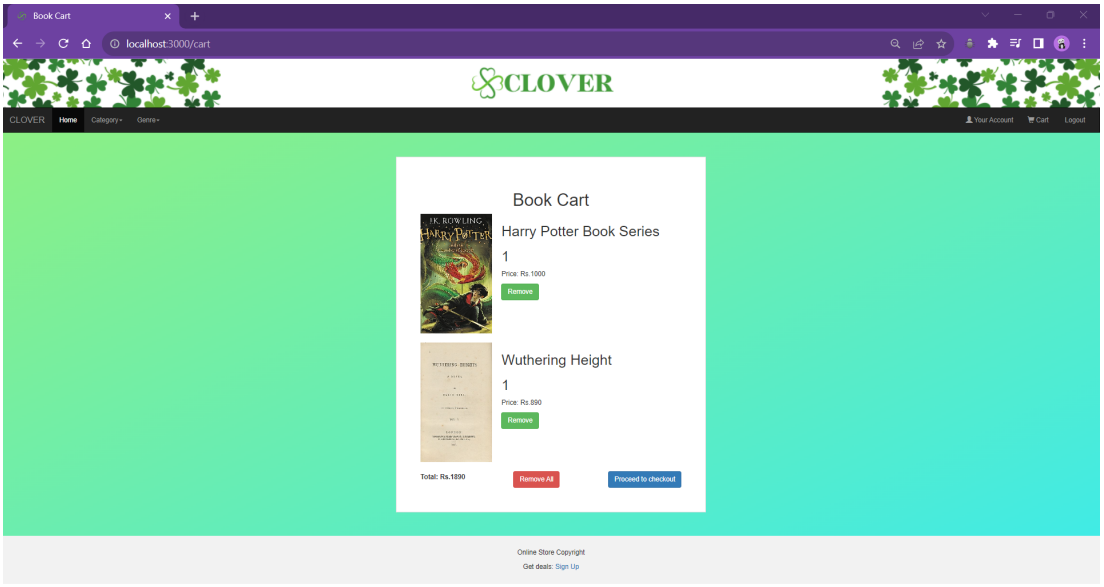


Figure 4.16 Cart page with items

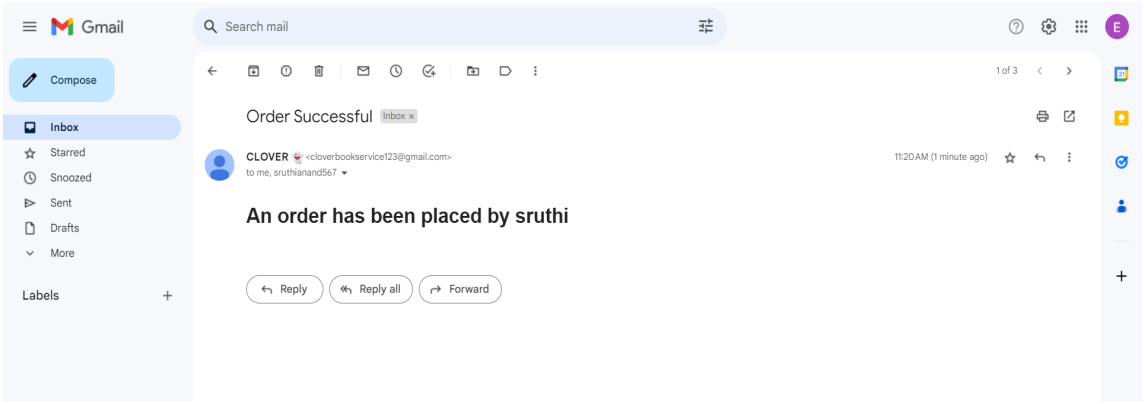


Figure 4.17 Email received by admin

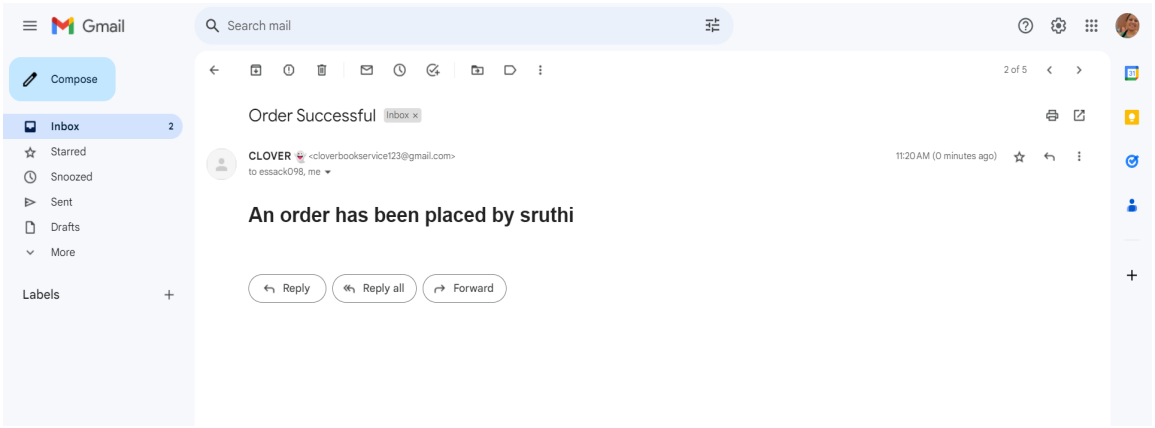


Figure 4.18 Email received by user

CHAPTER 5
CONCLUSION

The "E-commerce Book Service" project has achieved its primary objective of promoting sustainable and affordable reading through an innovative e-commerce platform. The project's implementation of a categorization system, cashback policy, and user-friendly interface has revolutionized the traditional book-buying experience, offering benefits that extend beyond the surface.

The success of the categorization ensures accurate and consistent classification of books based on their condition, allowing customers to make well-informed choices according to their preferences and budgets. This categorization approach has not only provided affordable alternatives for readers but also significantly contributed to reducing book waste and promoting environmental sustainability.

The cashback policy has been instrumental in incentivizing readers to return books, actively engaging in the circular economy, and fostering responsible book disposal. By rewarding customers for recycling books or opting for second-hand options, the project has encouraged positive environmental practices while building a sense of community among book enthusiasts.

The user-friendly interface and user account management system have enhanced the overall user experience, making it convenient for customers to access the platform

The "E-commerce Book Service" project has not only empowered readers to make sustainable and budget-conscious choices but has also provided a profitable opportunity for sellers to reach a broader audience and effectively manage their book inventory.

CHAPTER 6
REFERENCES

[1] Thapa Technical,(17 Nov, 2020) "Complete Registration Form using HTML, CSS, Node JS, Express, and MongoDB in Hindi in 2022," YouTube, 17-Nov-2020. [Online]. Available: https://youtu.be/XZiFBlfluZk.(visited on 16/5/2023)

[2] Thapa Technical,(18 Nov, 2020) "Signin Form || Login Form using HTML, CSS, Node JS, Express, & MongoDB in Hindi in 2020," YouTube, 18-Nov-2020. [Online]. Available: https://youtu.be/gX5vsKUO7QE.(visited on 18/7/2023)

[3] Academind,(26 May 2016) "NodeJS / Express / MongoDB - Build a Shopping Cart - #4 Seeding Data," YouTube, 26-May-2016. [Online]. Available: https://youtu.be/V30Rpqi6kYE.(visited on 24/7/2023)

[4] W3Schools, "CSS Tutorial," [Online]. Available: https://www.w3schools.com/css/.(visited on 23/5/2023)

[5] W3Schools, "HTML Tutorial," [Online]. Available: https://www.w3schools.com/html/.(visited on 22/5/2023)

[6] Academind, "NodeJS / Express / MongoDB - Build a Shopping Cart - #11 Adding a Session Store," YouTube, 25-Jun-2016. [Online]. Available: https://youtu.be/g32awc4HrLA.(visited on 25/7/2023)

[7] W3Schools, "Node.js Tutorial," [Online]. Available: https://www.w3schools.com/nodejs/.(visited on 24/7/2023)

[8] Express.js, "Session Middleware," [Online]. Available: https://expressjs.com/en/resources/middleware/session.html.(visited on 24/7/2023)

[9] Express.js, [Online]. Available: https://expressjs.com/.(visited on 7/7/2023)

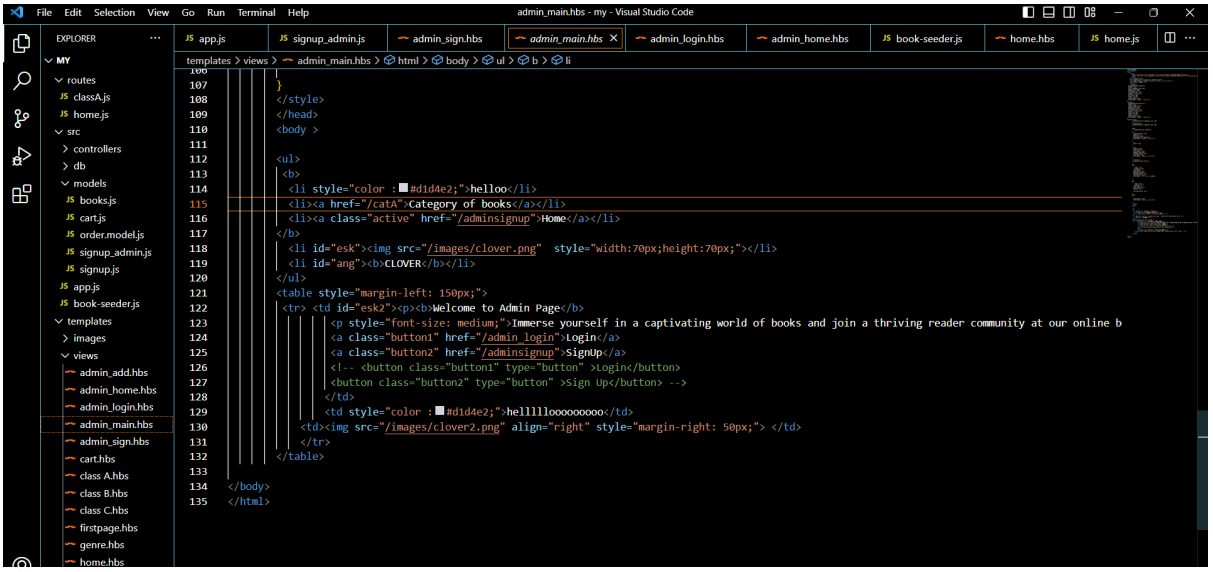
[10] Nodemailer, "About Nodemailer," [Online]. Available: https://nodemailer.com/about/.(visited on 7/7/2023)

APPENDIX A
PSEUDO CODE

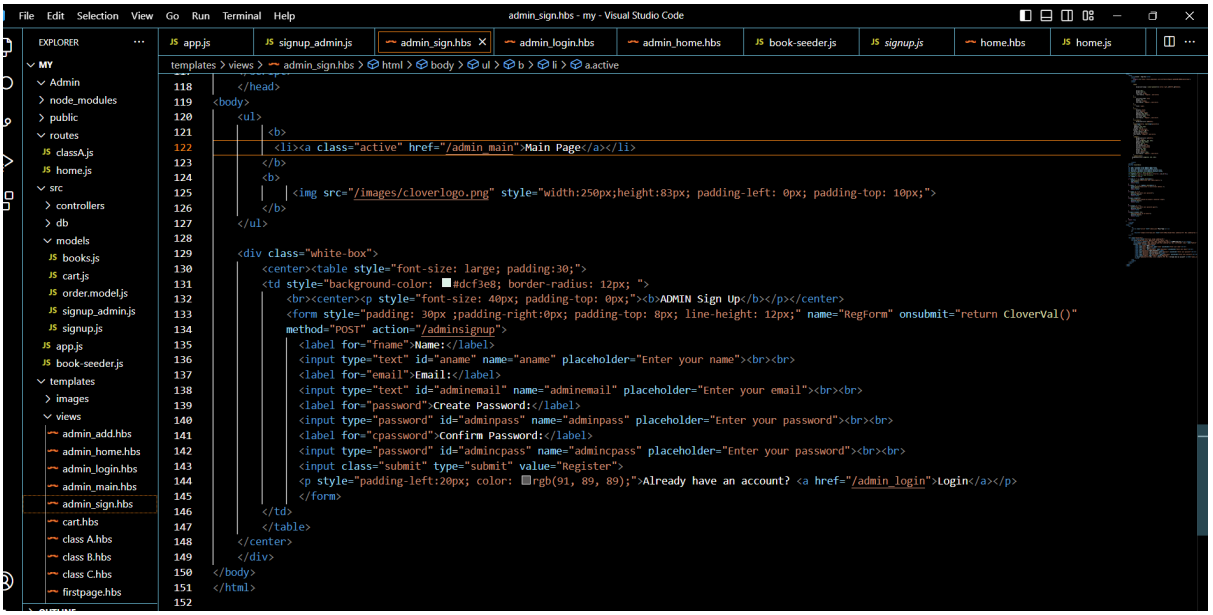
MAIN APPLICATION

ADMIN'S PAGES

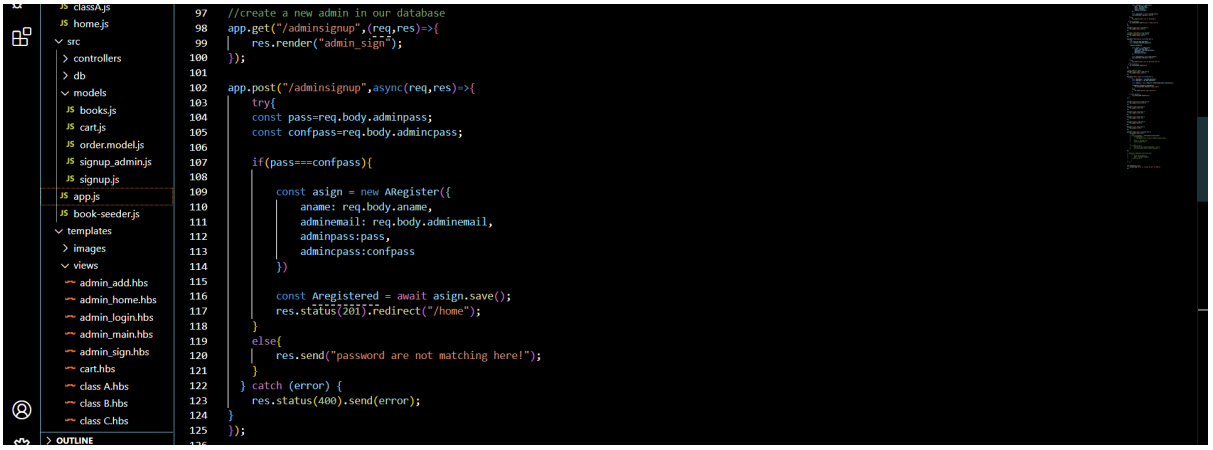
ADMIN'S MAIN PAGE



ADMIN'S SIGN UP

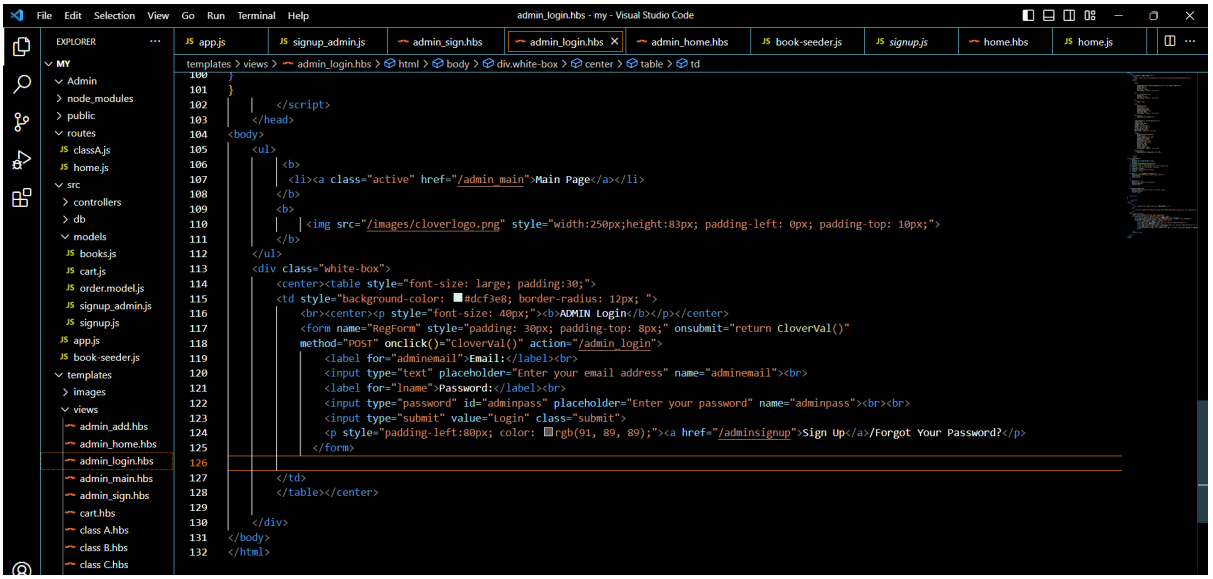


```
</head>
<body>
    <ul>
        <b>
            <li><a class="active" href="/admin_main">Main Page</a></li>
        </b>
        <b>
            <img src="/images/cloverlogo.png" style="width:250px;height:83px; padding-left: 0px; padding-top: 10px;">
        </b>
    </ul>

    <div class="white-box">
        <center><table style="font-size: large; padding:30;">
        <td style="background-color: #dcf3e8; border-radius: 12px; ">
            <br><center><p style="font-size: 40px; padding-top: 0px;"><b>ADMIN Sign Up</b></p></center>
            <form style="padding: 30px ;padding-right:0px; padding-top: 8px; line-height: 12px;" name="RegForm" onsubmit="return CloverVal()"
            method="POST" action="/adminsignup">
              <label for="fname">Name:</label>
              <input type="text" id="aname" name="aname" placeholder="Enter your name"><br><br>
              <label for="email">Email:</label>
              <input type="text" id="adminemail" name="adminemail" placeholder="Enter your email"><br><br>
              <label for="password">Create Password:</label>
              <input type="password" id="adminpass" name="adminpass" placeholder="Enter your password"><br><br>
              <label for="cpassword">Confirm Password:</label>
              <input type="password" id="admincpass" name="admincpass" placeholder="Enter your password"><br><br>
              <input class="submit" type="submit" value="Register">
              <p style="padding-left:20px; color: rgb(91, 89, 89);">Already have an account? <a href="/admin_login">Login</a></p>
              </form>
        </td>
        </table>
    </center>
    </div>
</body>
</html>
```

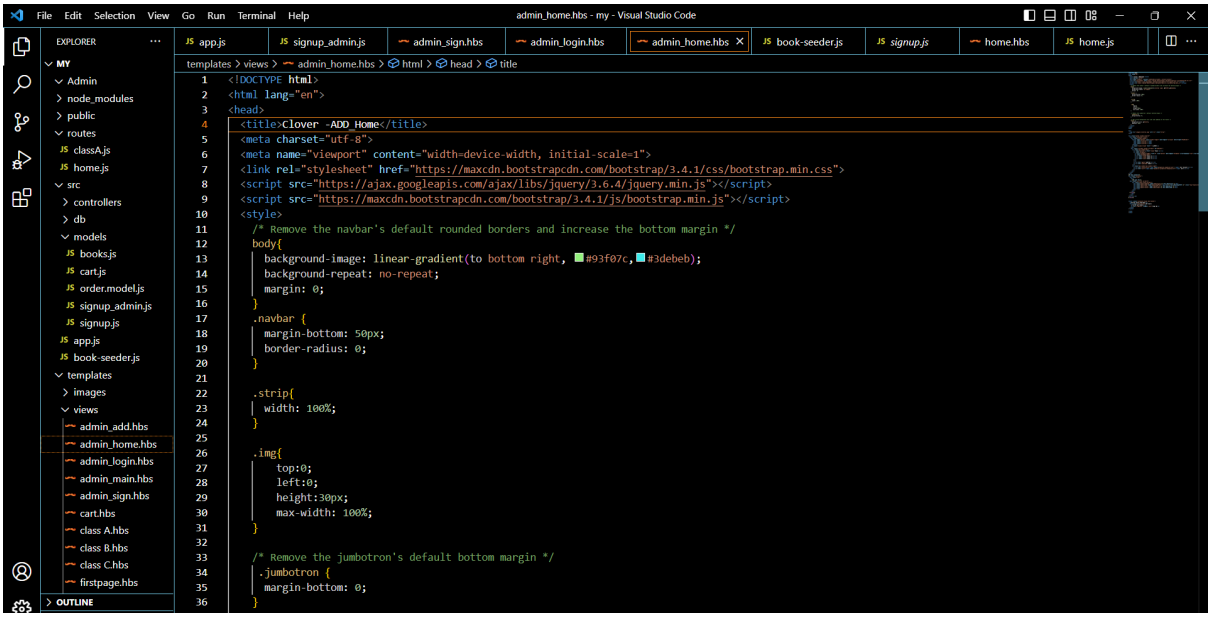


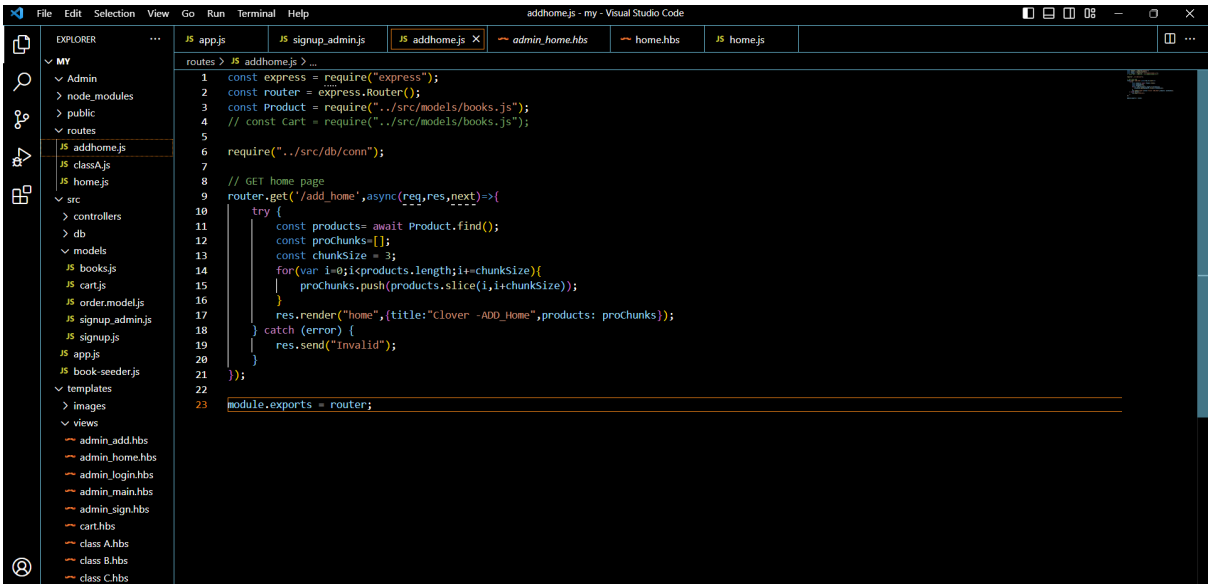
```
//create a new admin in our database
app.get("/adminsignup",(req,res)=>{
    res.render("admin_sign");
});

app.post("/adminsignup",async(req,res)=>{
    try{
    const pass=req.body.adminpass;
    const confpass=req.body.admincpass;

    if(pass===confpass){

        const asign = new ARegister({
            aname: req.body.aname,
            adminemail: req.body.adminemail,
            adminpass:pass,
            admincpass:confpass
        })

        const Aregistered = await asign.save();
        res.status(201).redirect("/home");
    }
    else{
        res.send("password are not matching here!");
    }
  } catch (error) {
    res.status(400).send(error);
}
});
```

ADMIN'S LOGIN

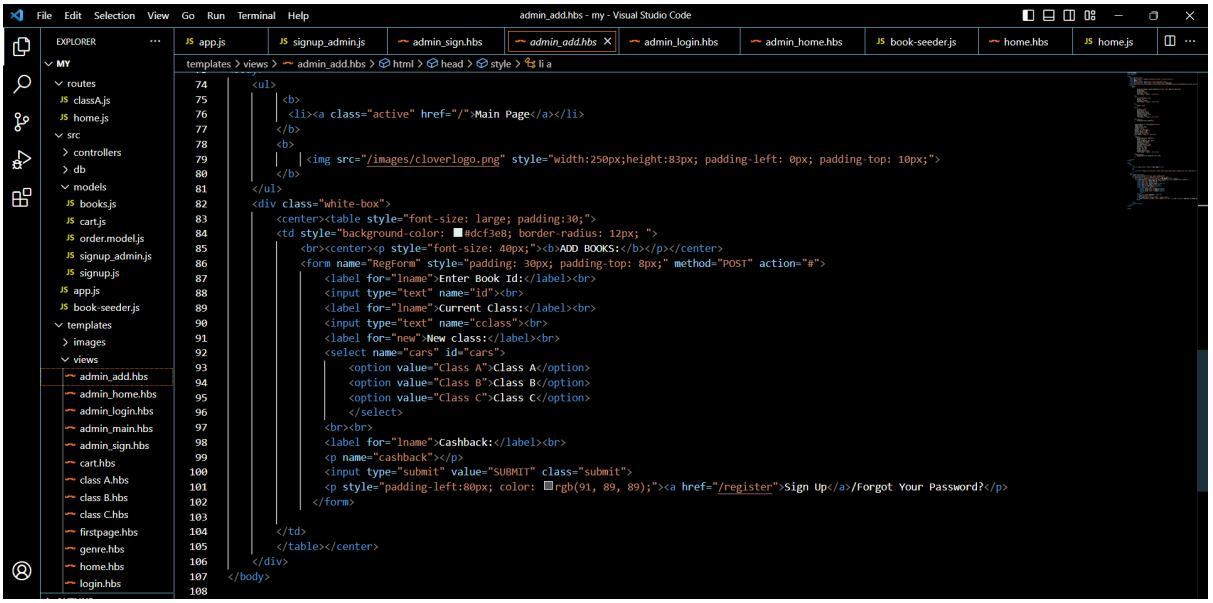


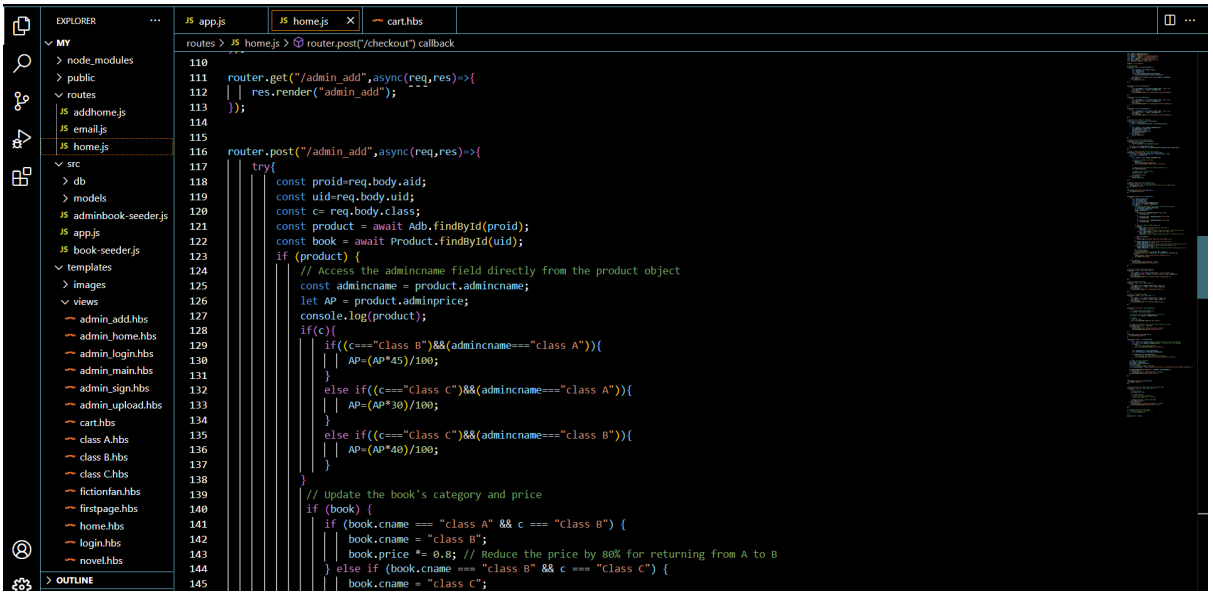
ADMIN'S HOME PAGE



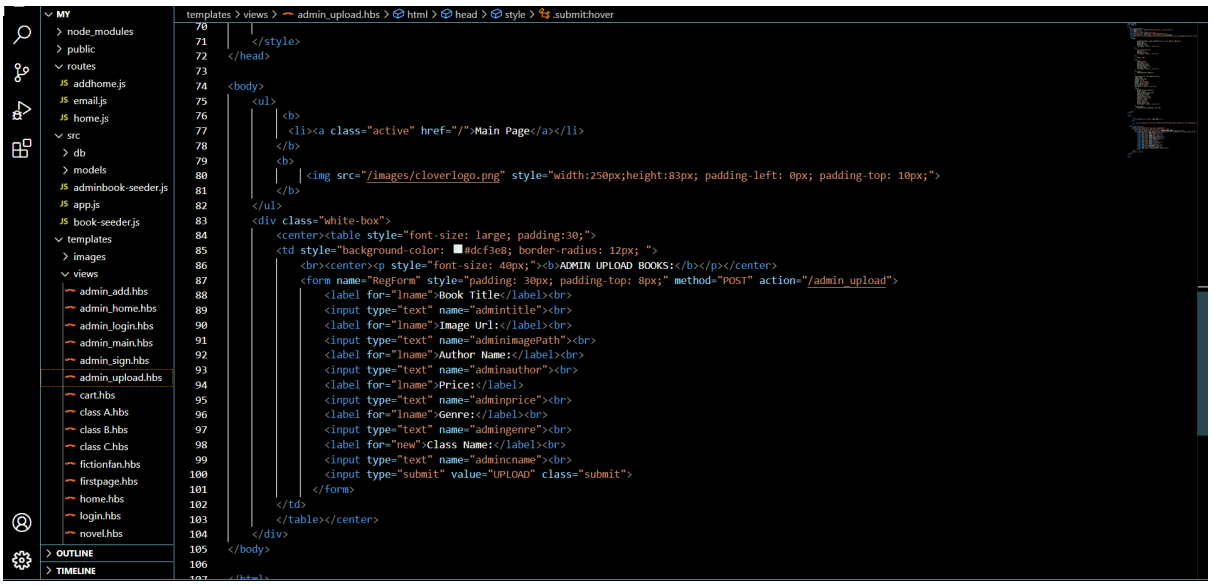


PAGE TO RETURN THE BOOK TO USER'S HOME PAGE





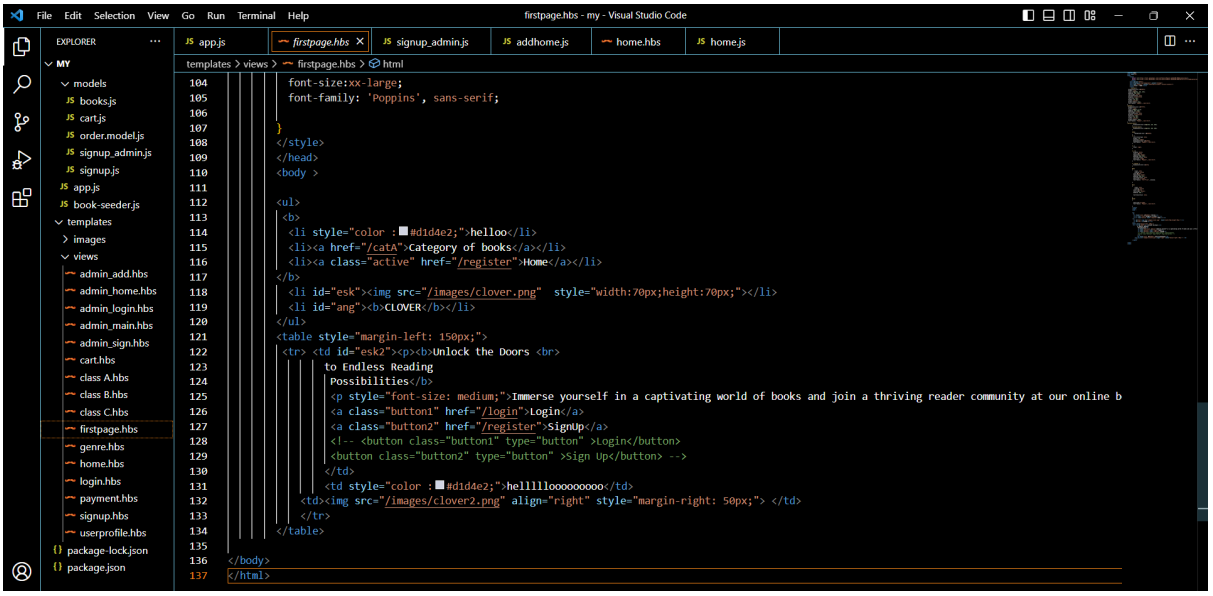
PAGE TO UPLOAD THE BOOK TO USER'S HOME PAGE



```
</style>
</head>

<body>
    <ul>
        <b>
          <li><a class="active" href="/">Main Page</a></li>
        </b>
        <b>
            <img src="/images/cloverlogo.png" style="width:250px;height:83px; padding-left: 0px; padding-top: 10px;">
        </b>
    </ul>
    <div class="white-box">
        <center><table style="font-size: large; padding:30;">
        <td style="background-color: #dcf3e8; border-radius: 12px; ">
            <br><center><p style="font-size: 40px;"><b>ADMIN UPLOAD BOOKS:</b></p></center>
            <form name="RegForm" style="padding: 30px; padding-top: 8px;" method="POST" action="/admin_upload">
                <label for="lname">Book Title</label><br>
                <input type="text" name="admintitle"><br>
                <label for="lname">Image Url:</label><br>
                <input type="text" name="adminimagePath"><br>
                <label for="lname">Author Name:</label><br>
                <input type="text" name="adminauthor"><br>
                <label for="lname">Price:</label>
                <input type="text" name="adminprice"><br>
                <label for="lname">Genre:</label><br>
                <input type="text" name="admingenre"><br>
                <label for="new">Class Name:</label><br>
                <input type="text" name="admincname"><br>
                <input type="submit" value="UPLOAD" class="submit">
            </form>
        </td>
        </table></center>
    </div>
</body>
```

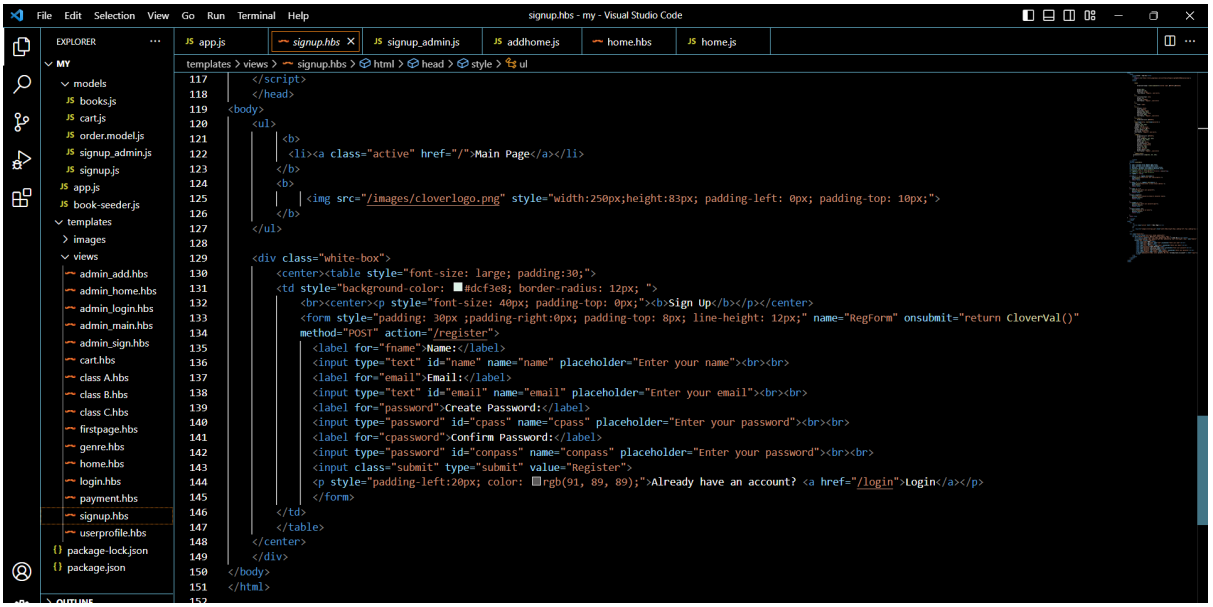
USER'S PAGES

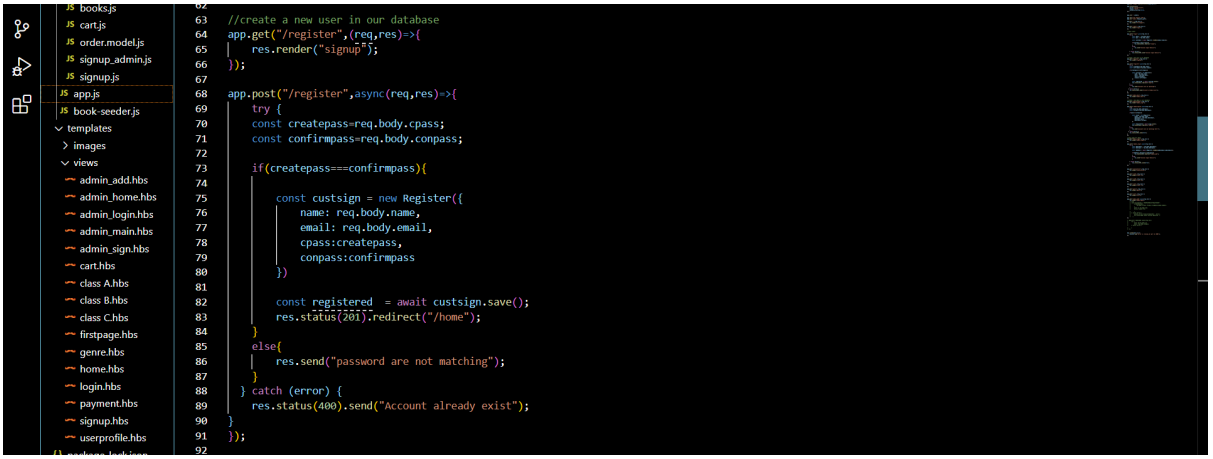
USER'S MAIN PAGE



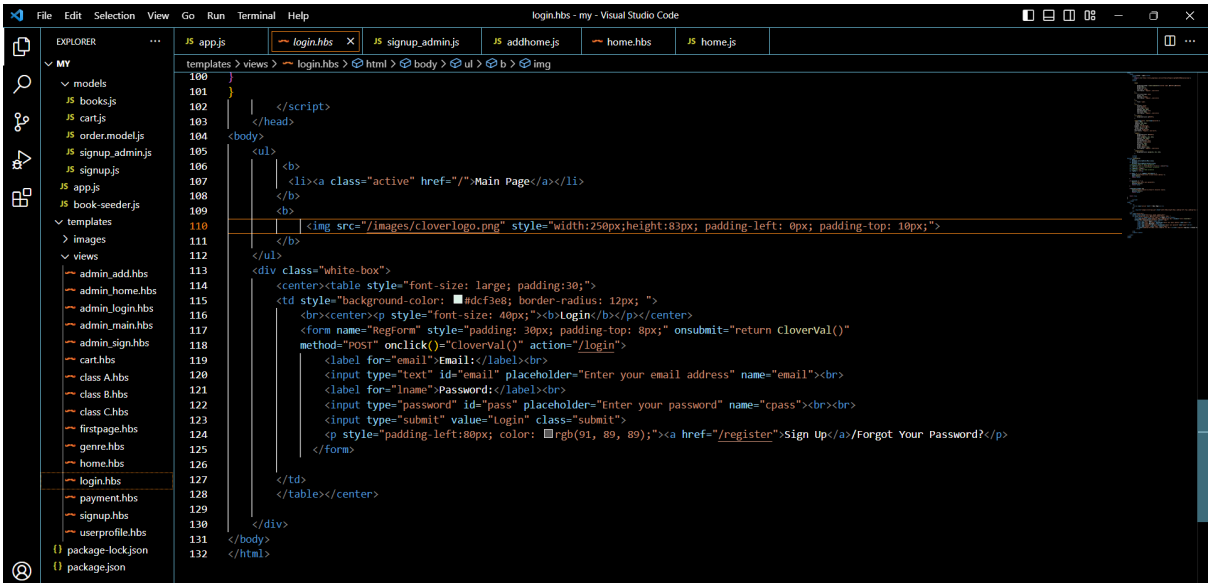
```
        font-size:xx-large;
        font-family: 'Poppins', sans-serif;

    }
    </style>
    </head>
    <body >

    <ul>
      <b>
        <li style="color :#d1d4e2;">helloo</li>
        <li><a href="/catA">Category of books</a></li>
        <li><a class="active" href="/register">Home</a></li>
      </b>
        <li id="esk"><img src="/images/clover.png"  style="width:70px;height:70px;"></li>
        <li id="ang"><b>CLOVER</b></li>
      </ul>
      <table style="margin-left: 150px;">
        <tr> <td id="esk2"><p><b>Unlock the Doors <br>
                to Endless Reading
                Possibilities</b>
                <p style="font-size: medium;">Immerse yourself in a captivating world of books and join a thriving reader community at our online b
                <a class="button1" href="/login">Login</a>
                <a class="button2" href="/register">SignUp</a>
                <!-- <button class="button1" type="button" >Login</button>
                <button class="button2" type="button" >Sign Up</button> -->
                </td>
                <td style="color :#d1d4e2;">hellllloooooooo</td>
            <td><img src="/images/clover2.png" align="right" style="margin-right: 50px;"> </td>
            </tr>
        </table>

</body>
</html>
```

USER'S SIGN UP PAGE





USER'S LOGIN PAGE



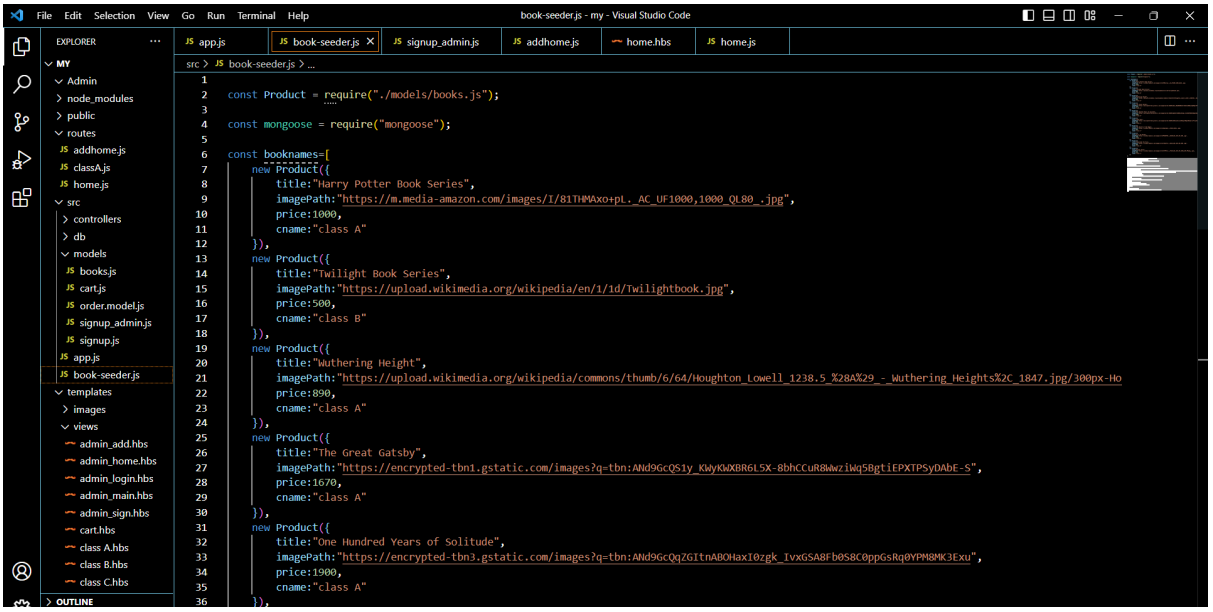
```
}
}
    </script>
  </head>
<body>
  <ul>
    <b>
      <li><a class="active" href="/">Main Page</a></li>
    </b>
    <b>
      <img src="/images/cloverlogo.png" style="width:250px;height:83px; padding-left: 0px; padding-top: 10px;">
    </b>
  </ul>
  <div class="white-box">
    <center><table style="font-size: large; padding:30;">
    <td style="background-color: #dcf3e8; border-radius: 12px; ">
      <br><center><p style="font-size: 40px;"><b>Login</b></p></center>
      <form name="RegForm" style="padding: 30px; padding-top: 8px;" onsubmit="return CloverVal()"
      method="POST" onclick()="CloverVal()" action="/login">
        <label for="email">Email:</label><br>
        <input type="text" id="email" placeholder="Enter your email address" name="email"><br>
        <label for="lname">Password:</label><br>
        <input type="password" id="pass" placeholder="Enter your password" name="cpass"><br><br>
        <input type="submit" value="Login" class="submit">
        <p style="padding-left:80px; color: rgb(91, 89, 89);"><a href="/register">Sign Up</a>/Forgot Your Password?</p>
      </form>

    </td>
    </table></center>

  </div>
</body>
</html>
```

```
app.get("/login",(req,res)=>{
    res.render("login");
});

//login check

app.post("/login",async(req,res)=>{
    try {
        const email = req.body.email;
        const pass = req.body.cpass;

        const useremail = await Register.findOne({email:email});

        if(useremail.cpass===pass){
            res.status(201).redirect("/home");
        }
        else{
            res.send("Invalid login Details");
        }

    } catch (error) {
        res.status(400).send("Invalid login Details");
    }
});
```

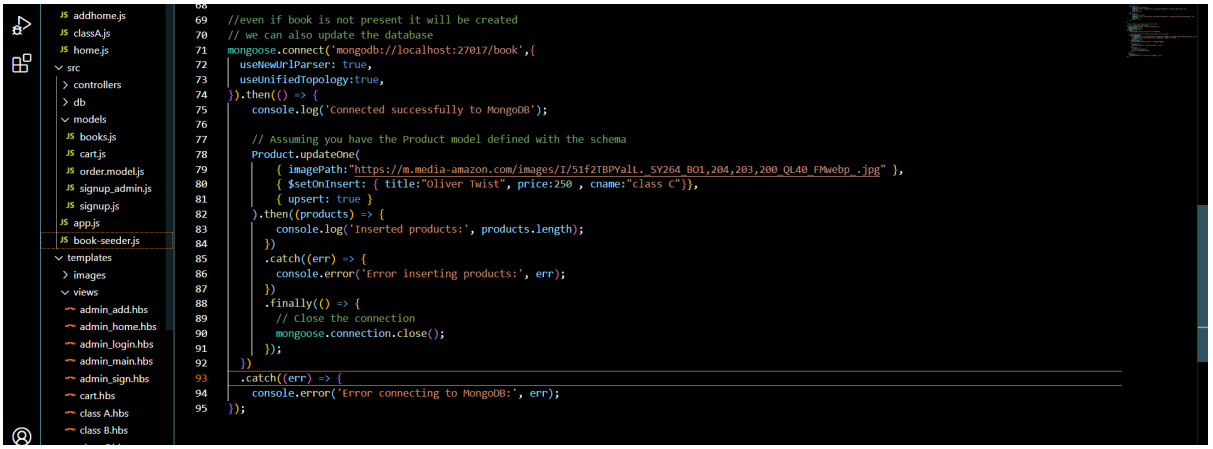
USER'S HOME PAGE

```
{{# each products}}
<div class="container">
  <div class="row">
    <div>
      {{# each this}}
  <div class="col-xs-6 col-md-4">
      <div class="panel panel-primary">
        <div class="panel-heading" name="title">{{ this.title }}</div>
        <div class="panel-body" name="imagePath"><img src="{{ this.imagePath }}" class="img-responsive" style="width:100%" alt="Image"></div>
        <div class="panel-footer" name="price">{{ this.price }}</div>
        <div class="panel-footer" name="cname">{{ this.cname }}</div>
        {{!-- <a href="/cart/{{this._id}}" class="btn btn-success btn-block">Add to Cart</a> --}}
        <div></div>
      </div>
    </div>
  {{/each}}
    </div>
  </div>
</div><br><br>
{{/each}}


<footer class="container-fluid text-center">
  <p>Online Store Copyright</p>
  <form class="form-inline">Get deals:
    <p>To Get more deals:</p>
    <a href="/register" target="_self">Sign Up</a>
  </form>
</footer>
```

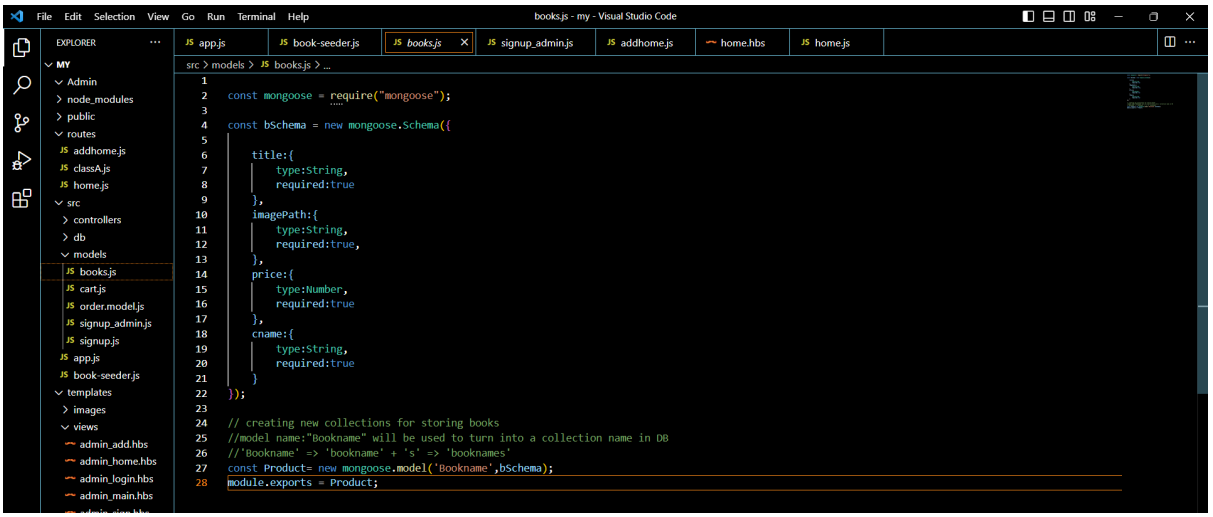
```
const express = require("express");
const router = express.Router();
const Product = require("../src/models/books.js");
const Cart = require("../src/models/books.js");

require("../src/db/conn");

// GET home page
router.get('/home',async(req,res,next)=>{
    try {
        const products= await Product.find();
        const proChunks=[];
        const chunkSize = 3;
        for(var i=0;i<products.length;i+=chunkSize){
            proChunks.push(products.slice(i,i+chunkSize));
        }
        res.render("home",{title:"Clover -Home",products: proChunks});
    } catch (error) {
        res.send("Invalid");
    }
});
```



```
const Product = require("./models/books.js");

const mongoose = require("mongoose");

const booknames=[
    new Product({
        title:"Harry Potter Book Series",
        imagePath:"https://m.media-amazon.com/images/I/81THMAxo+pL._AC_UF1000,1000_QL80_.jpg",
        price:1000,
        cname:"class A"
    }),
    new Product({
        title:"Twilight Book Series",
        imagePath:"https://upload.wikimedia.org/wikipedia/en/1/1d/Twilightbook.jpg",
        price:500,
        cname:"class B"
    }),
    new Product({
        title:"Wuthering Height",
        imagePath:"https://upload.wikimedia.org/wikipedia/commons/thumb/6/64/Houghton_Lowell_1238.5_%28A%29_-_Wuthering_Heights%2C_1847.jpg/300px-Ho
        price:890,
        cname:"class A"
    }),
    new Product({
        title:"The Great Gatsby",
        imagePath:"https://encrypted-tbn1.gstatic.com/images?q=tbn:ANd9GcQS1y_KWyKWXBR6L5X-8bhCCuR8WwziWq5BgtiEPXTPSyDAbE-S",
        price:1670,
        cname:"class A"
    }),
    new Product({
        title:"One Hundred Years of Solitude",
        imagePath:"https://encrypted-tbn3.gstatic.com/images?q=tbn:ANd9GcQqZGItnABOHaxI0zgk_IvxGSA8Fb0S8C0ppGsRq0YPM8MK3Exu",
        price:1900,
        cname:"class A"
    }),
```



```
//even if book is not present it will be created
// we can also update the database
mongoose.connect('mongodb://localhost:27017/book',{
  useNewUrlParser: true,
  useUnifiedTopology:true,
}).then(() => {
    console.log('Connected successfully to MongoDB');

    // Assuming you have the Product model defined with the schema
    Product.updateOne(
        { imagePath:"https://m.media-amazon.com/images/I/51f2TBPYaLL._SY264_BO1,204,203,200_QL40_FMwebp_.jpg" },
        { $setOnInsert: { title:"Oliver Twist", price:250 , cname:"class C"}},
        { upsert: true }
    ).then((products) => {
        console.log('Inserted products:', products.length);
      })
      .catch((err) => {
        console.error('Error inserting products:', err);
      })
      .finally(() => {
        // Close the connection
        mongoose.connection.close();
      });
  })
  .catch((err) => {
    console.error('Error connecting to MongoDB:', err);
});
```



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

const bSchema = new mongoose.Schema({

    title:{
        type:String,
        required:true
    },
    imagePath:{
        type:String,
        required:true,
    },
    price:{
        type:Number,
        required:true
    },
    cname:{
        type:String,
        required:true
    }
});

// creating new collections for storing books
//model name:"Bookname" will be used to turn into a collection name in DB
//'Bookname' => 'bookname' + 's' => 'booknames'
const Product= new mongoose.model('Bookname',bSchema);
module.exports = Product;
```

CATEGORY PAGE IN HTML

CLASS A

```
</nav>
<center><b style="font-size:xx-large;">Class A Books</b></center>
<br><br><br>

{{#each books}}
<div class="container">
  <div class="row">
    <div>
  <div class="col-xs-6 col-md-4">
      <div class="panel panel-primary">
        <div class="panel-heading" name="title">{{ this.title }}</div>
        <div class="panel-body" name="imagePath"><img src="{{ this.imagePath }}" class="img-responsive" style="width:100%" alt="Image"></div>
        <div class="panel-footer" name="price">{{ this.price }}</div>
        <div class="panel-footer" name="cname">{{ this.cname }}</div>
        <a href="/cart/{{this._id}}" class="btn btn-success btn-block">Add to Cart</a>
        <div></div>
      </div>
    </div>
    </div>
  </div>
</div><br><br>
{{/each}}

<footer class="container-fluid text-center">
  <p>Online Store Copyright</p>
  <form class="form-inline">Get deals:
    <a href="/register" target="_self">Sign Up</a>
  </form>
</footer>

</body>
```

CLASS B

```
adminbook-seeder.js
app.js
book-seeder.js
templates
  images
  views
    admin_add.hbs
    admin_home.hbs
    admin_login.hbs
    admin_main.hbs
    admin_sign.hbs
    admin_upload.hbs
    cart.hbs
    class A.hbs
    class B.hbs
    class C.hbs
    fictionfan.hbs
    firstpage.hbs
    home.hbs
    login.hbs
    novel.hbs
    payment.hbs
    signup.hbs
    tra.hbs
    updatepass.hbs
    user_upload.hbs
```

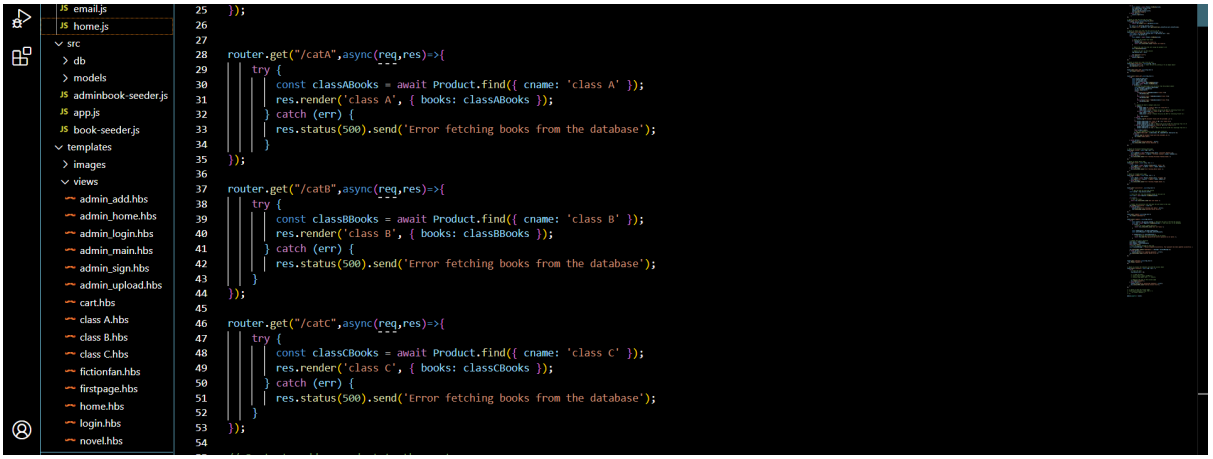
```
    </div>
  </div>
</nav>
<center><b style="font-size:xx-large;">Class B Books</b></center>
<br><br><br>

{{#each books}}
<div class="container">
  <div class="row">
    <div>
  <div class="col-xs-6 col-md-4">
      <div class="panel panel-primary">
        <div class="panel-heading" name="title">{{ this.title }}</div>
        <div class="panel-body" name="imagePath"><img src="{{ this.imagePath }}" class="img-responsive" style="width:100%" alt="Image"></div>
        <div class="panel-footer" name="price">{{ this.price }}</div>
        <div class="panel-footer" name="cname">{{ this.cname }}</div>
        <a href="/cart/{{this._id}}" class="btn btn-success btn-block">Add to Cart</a>
        <div></div>
      </div>
    </div>
    </div>
  </div>
</div><br><br>
{{/each}}

<footer class="container-fluid text-center">
  <p>Online Store Copyright</p>
  <form class="form-inline">Get deals:
    <a href="/register" target="_self">Sign Up</a>
  </form>
</footer>
```

CLASS C

```
images
views
  admin_add.hbs
  admin_home.hbs
  admin_login.hbs
  admin_main.hbs
  admin_sign.hbs
  admin_upload.hbs
  cart.hbs
  class A.hbs
  class B.hbs
  class C.hbs
  fictionfan.hbs
  firstpage.hbs
  home.hbs
  login.hbs
  novel.hbs
  payment.hbs
  signup.hbs
  tra.hbs
  updatepass.hbs
  user_upload.hbs
  userprofile.hbs
```

```
    </ul>
    </div>
  </div>
</nav>
<center><b style="font-size:xx-large;">Class C Books</b></center>
<br><br><br>

{{#each books}}
<div class="container">
  <div class="row">
    <div>
  <div class="col-xs-6 col-md-4">
      <div class="panel panel-primary">
        <div class="panel-heading" name="title">{{ this.title }}</div>
        <div class="panel-body" name="imagePath"><img src="{{ this.imagePath }}" class="img-responsive" style="width:100%" alt="Image"></div>
        <div class="panel-footer" name="price">{{ this.price }}</div>
        <div class="panel-footer" name="cname">{{ this.cname }}</div>
        <a href="/cart/{{this._id}}" class="btn btn-success btn-block">Add to Cart</a>
        <div></div>
      </div>
    </div>
    </div>
  </div>
</div><br><br>
{{/each}}
```

EXPRESS.JS CODE(CATEGORY)



```
email.js
home.js
src
  db
  models
  adminbook-seeder.js
  app.js
  book-seeder.js
templates
  images
  views
    admin_add.hbs
    admin_home.hbs
    admin_login.hbs
    admin_main.hbs
    admin_sign.hbs
    admin_upload.hbs
    cart.hbs
    class A.hbs
    class B.hbs
    class C.hbs
    fictionfan.hbs
    firstpage.hbs
    home.hbs
    login.hbs
    novel.hbs
```

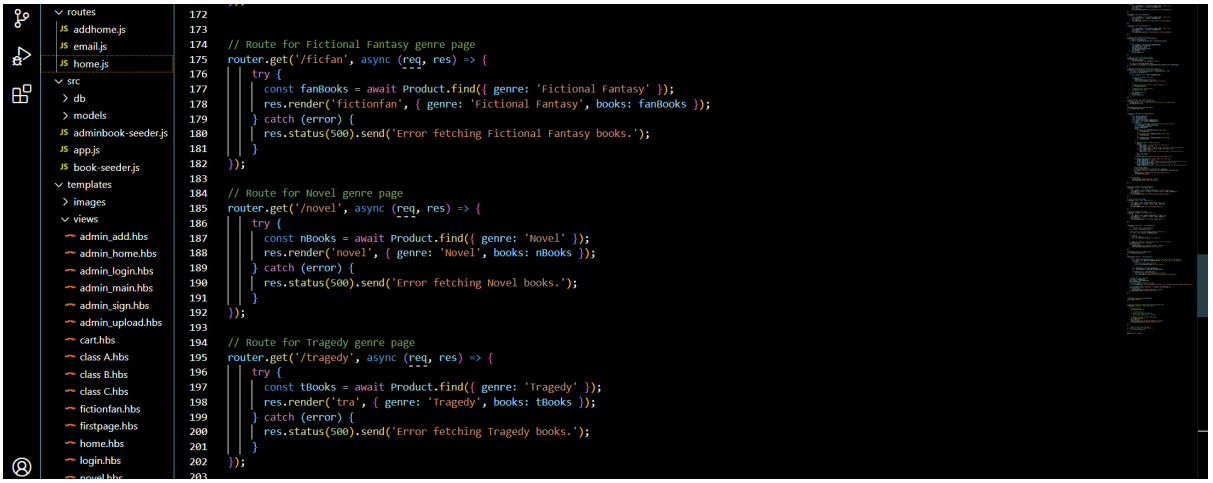
```
});


router.get("/catA",async(req,res)=>{
    try {
        const classABooks = await Product.find({ cname: 'class A' });
        res.render('class A', { books: classABooks });
    } catch (err) {
        res.status(500).send('Error fetching books from the database');
    }
});

router.get("/catB",async(req,res)=>{
    try {
        const classBBooks = await Product.find({ cname: 'class B' });
        res.render('class B', { books: classBBooks });
    } catch (err) {
        res.status(500).send('Error fetching books from the database');
    }
});

router.get("/catC",async(req,res)=>{
    try {
        const classCBooks = await Product.find({ cname: 'class C' });
        res.render('class C', { books: classCBooks });
    } catch (err) {
        res.status(500).send('Error fetching books from the database');
    }
});
```

CART PAGE

```
</ul>
<ul class="nav navbar-nav navbar-right">
<li><a href="/userprofile"><span class="glyphicon glyphicon-user"></span> Your Account</a></li>
<li><a href="/cart"><span class="glyphicon glyphicon-shopping-cart"></span> Cart</a></li>
<li><a href="/"><span class="glyphicon glyphicon-logout"></span> Logout</a></li>
</ul>
</div>
</div>
</nav>
{{# if products}}
<div class="cart-container">
<h1>Book Cart</h1>
{{# each products}}
<div class="book-item">
<img src=" {{this.item.imagePath }} " alt="Book Cover">
<div class="book-details">
<h2>{{this.item.title}}</h2>
<h2>{{this.qty}}</h2>
<p>Price: Rs.{{this.price}}</p>
<a href="/cart/remove/{{ this.item._id }}" class="btn btn-success">Remove</a>
</div>
</div>
{{/each}}
<div class="total-price">
<p><b> Total: Rs.{{totalPrice}}</b></p>
<a href="/clear-cart" class="btn btn-danger">Remove All</a>
<a href="/checkout" class="btn btn-primary">Proceed to checkout</a>
</div>
</div>
{{else}}
<div class="row">
<div>
<h2><b><center>No items In cart</center></b></h2>
</div>
</div>
{{/if}}
```

GENRE

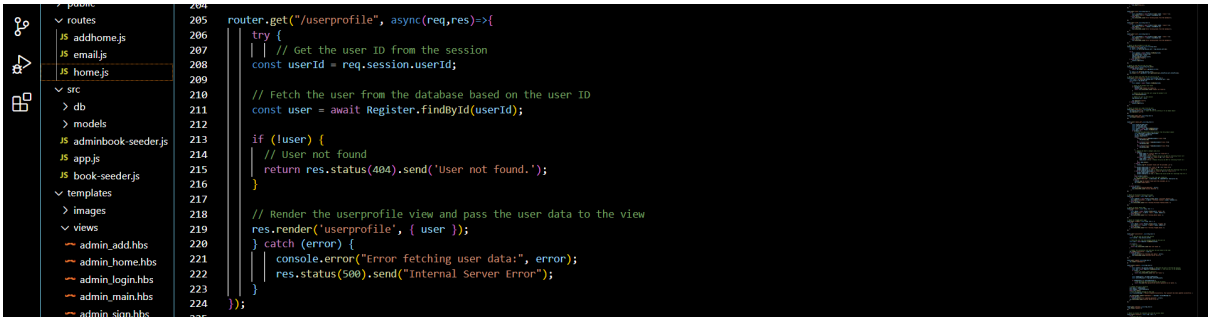
```
{{#each books}}
<div class="container">
<div class="row">
<div>
<div class="col-xs-6 col-md-4">
<div class="panel panel-primary">
<div class="panel-heading" name="title">{{ this.title }}</div>
<div class="panel-body" name="imagePath"><img src="{{ this.imagePath }}" class="img-responsive" style="width:100%" alt="Image"></div>
<div class="panel-footer" name="price">{{ this.author }}</div>
<div class="panel-footer" name="price">{{ this.price }}</div>
<div class="panel-footer" name="price">{{ this.genre }}</div>
<div class="panel-footer" name="cname">{{ this.cname }}</div>
<a href="/cart/{{this._id}}" class="btn btn-success btn-block">Add to Cart</a>
<div></div>
</div>
</div>
</div>
</div>
</div><br><br>
{{/each}}

<footer class="container-fluid text-center">
<p>Online Store Copyright</p>
<form class="form-inline">Get deals:
<p>To Get more deals:</p>
<a href="/register" target="_self">Sign Up</a>
```



```
// Route for Fictional Fantasy genre page
router.get('/ficfan', async (req, res) => {
    try {
        const fanBooks = await Product.find({ genre: 'Fictional Fantasy' });
        res.render('fictionfan', { genre: 'Fictional Fantasy', books: fanBooks });
    } catch (error) {
        res.status(500).send('Error fetching Fictional Fantasy books.');
    }
});

// Route for Novel genre page
router.get('/novel', async (req, res) => {
    try {
        const nBooks = await Product.find({ genre: 'Novel' });
        res.render('novel', { genre: 'Novel', books: nBooks });
    } catch (error) {
        res.status(500).send('Error fetching Novel books.');
    }
});

// Route for Tragedy genre page
router.get('/tragedy', async (req, res) => {
    try {
        const tBooks = await Product.find({ genre: 'Tragedy' });
        res.render('tra', { genre: 'Tragedy', books: tBooks });
    } catch (error) {
        res.status(500).send('Error fetching Tragedy books.');
    }
});
```

USERPROFILE

```
</head>
<body>
    <div class="container">
    <div class="sidebar">
      <h2><img src="/images/cloverlogo.png" style="width:250px;height:83px; padding-left: 0px; padding-top: 10px;"></h2>
      <ul>
        <li><a href="/home">Home</a></li>
        <li><a href="/userprofile">About</a></li>
        <li><a href="/update">Change Password</a></li>
        <li><a href="/">Logout</a></li>
      </ul>
    </div>
    <div class="white-box">
      <center>
        <table style="font-size: large; padding: 50;">
          <td style="background-color: #ffffff; border-radius: 12px; ">
          <br><center><p style="font-size: 40px;"><b>Profile</b></p></center>
          <form name="RegForm" style="padding: 30px; padding-top: 8px;" method="POST" action="#">
            <label for="fname"><b>Name:</b></label>
            <p name="name">{{ user.name }}</p>
            <label for="email"><b>Email:</b></label><br>
            <p name="email">{{ user.email }}</p>
            <label for="address"><b>Address:</b></label>
            <p name="address">{{ user.address }}</p>
            <label for="Phone"><b>Phone Number:</b></label>
            <p name="Phone">{{ user.Phone }}</p>
            </form>
        </td>
        </table>
        </center>
    </div>
    </div>
</body>
</html>
```



```
router.get("/userprofile", async(req,res)=>{
    try {
        // Get the user ID from the session
    const userId = req.session.userId;

    // Fetch the user from the database based on the user ID
    const user = await Register.findById(userId);

    if (!user) {
        // User not found
        return res.status(404).send('User not found.');
    }

    // Render the userprofile view and pass the user data to the view
    res.render('userprofile', { user });
    } catch (error) {
        console.error("Error fetching user data:", error);
        res.status(500).send("Internal Server Error");
    }
});
```

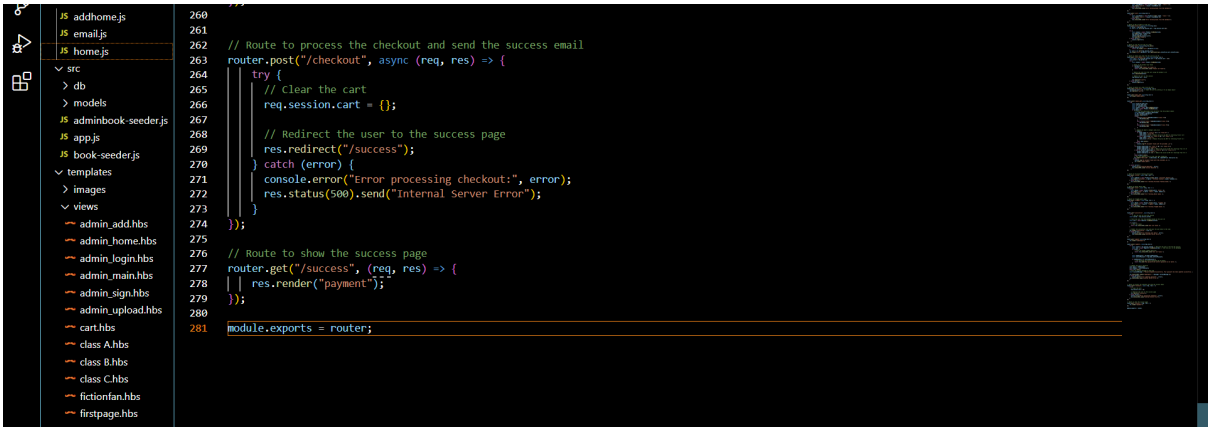
UPDATE PASSWORD

```
</head>
<body>
    <div class="container">
    <div class="sidebar">
      <h2><img src="/images/cloverlogo.png" style="width:250px;height:83px; padding-left: 0px; padding-top: 10px;"></h2>
      <ul>
        <li><a href="/home">Home</a></li>
        <li><a href="/userprofile">About</a></li>
        <li><a href="/update">Change Password</a></li>
        <li><a href="/">Logout</a></li>
      </ul>
    </div>
    <div class="white-box">
        <center>
          <table style="font-size: large; padding: 50;">
            <td style="background-color: #ffffff; border-radius: 12px; ">
            <br><center><p style="font-size: 40px;"><b>Update</b></p></center>
            <form name="Regform" style="padding: 30px; padding-top: 8px;" method="POST" action="/update">
                <label for="newPassword">Create Password:</label>
                <input type="password" id="newPassword" name="newPassword" placeholder="Enter your new password"><br><br>
                <label for="confirmPassword">Confirm Password:</label>
                <input type="password" id="confirmPassword" name="confirmPassword" placeholder="Confirm password"><br><br>
                <input class="submit" type="submit" value="change">
                <p>{{ message }}</p>
            </form>
          </td>
          </table>
          </center>
    </div>
    </div>
</body>
</html>
```

```
router.get("/update",async(req,res)=>{
    res.render("updatepass");
})

router.post("/update", async(req,res)=>{
    try {
        const userId = req.session.userId; // Retrieve the user's ID from the session
        const user = await Register.findById(userId); // Find the user in the database
        if (!user) {
            // User not found, handle the error
            return res.status(404).send('User not found.');
        }

        const newPassword = req.body.newPassword;
        const confirmPassword = req.body.confirmPassword;

        if (newPassword !== confirmPassword) {
            // New password and confirm password do not match
            return res.send('New password and confirm password do not match.');
        }

    // Update the user's password
    user.cpass = newPassword;
    user.conpass = confirmPassword;
    await user.save();
    // Pass the success message to the view
    const successMessage = 'Password Updated Successfully. Your password has been updated successfully.';

    res.status(200).render("updatepass", { message: successMessage });
    } catch (error) {
        console.error('Error updating password:', error);
    res.status(500).send('Internal Server Error');
    }
});
```

SUCCESS PAGE

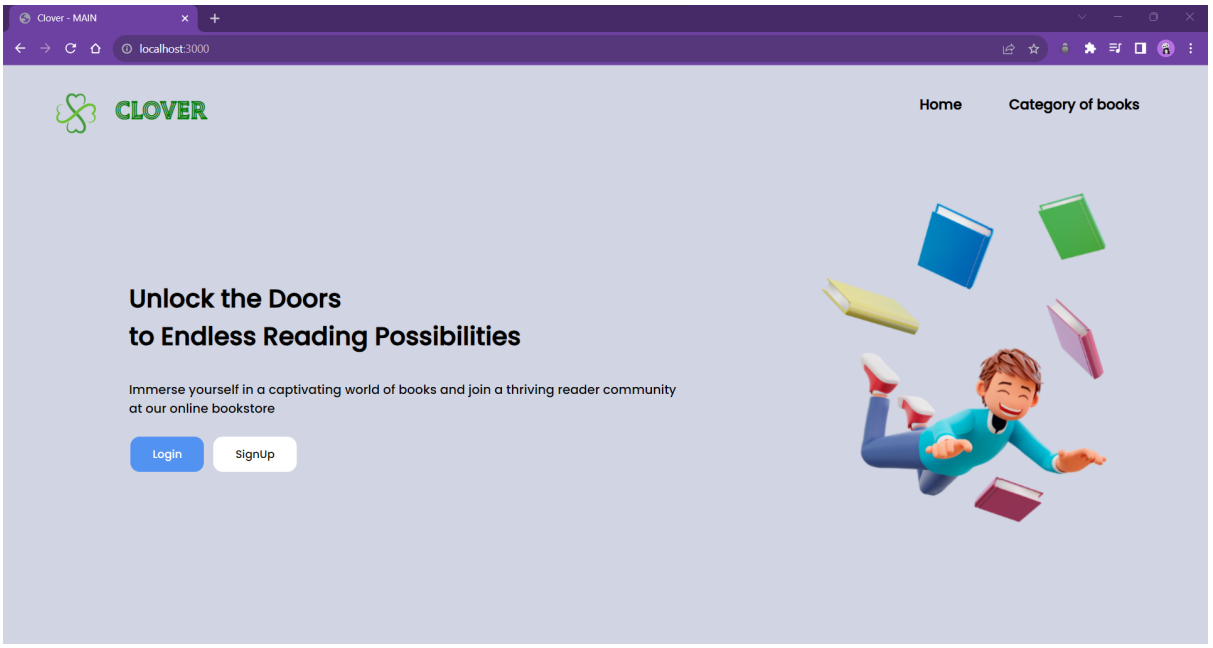
```
                display: block;
                color: black;
                text-align: center;
                padding: 40px 50px;
                text-decoration: none;
                font-size: 20px;
                font-family: 'Poppins', sans-serif;
            }
            li a:hover {
                background-color: #93f07c;
            }

        </style>
    </head>
<body>
    <ul>
        <b>
          <li><a class="active" href="/">Main Page</a></li>
        </b>
        <b>
            <img src="/images/cloverlogo.png" style="width:250px;height:83px; padding-left: 0px; padding-top: 10px;">
        </b>
    </ul>
    <div>
        <h1><center>Order has been placed. Will be send to your house.</center></h1>
    </div>
    </div>
</body>
</html>
```

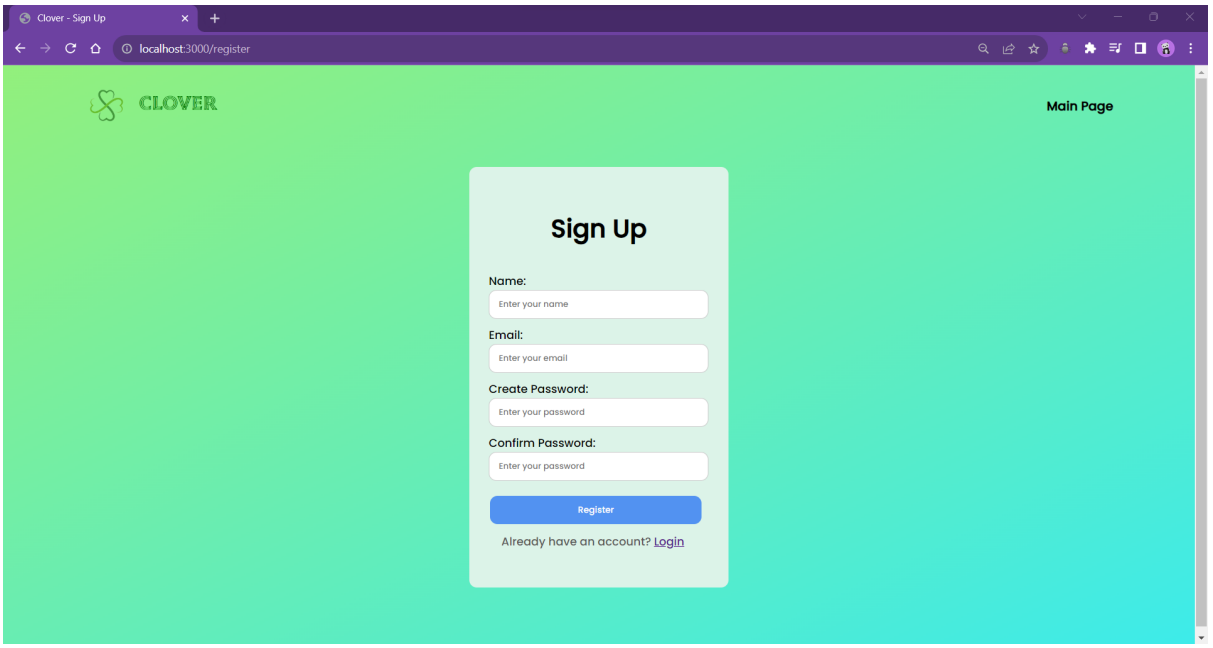

addhome.js
email.js
home.js
src
db
models
adminbook-seeder.js
app.js
book-seeder.js
templates
images
views
admin_add.hbs
admin_home.hbs
admin_login.hbs
admin_main.hbs
admin_sign.hbs
admin_upload.hbs
cart.hbs
class A.hbs
class B.hbs
class C.hbs
fictionfan.hbs
firstpage.hbs
260
261
262 // Route to process the checkout and send the success email
263 router.post("/checkout", async (req, res) => {
264 try {
265 // Clear the cart
266 req.session.cart = {};
267
268 // Redirect the user to the success page
269 res.redirect("/success");
270 } catch (error) {
271 console.error("Error processing checkout:", error);
272 res.status(500).send("Internal Server Error");
273 }
274 });
275
276 // Route to show the success page
277 router.get("/success", (req, res) => {
278 res.render("payment");
279 });
280
281 module.exports = router;

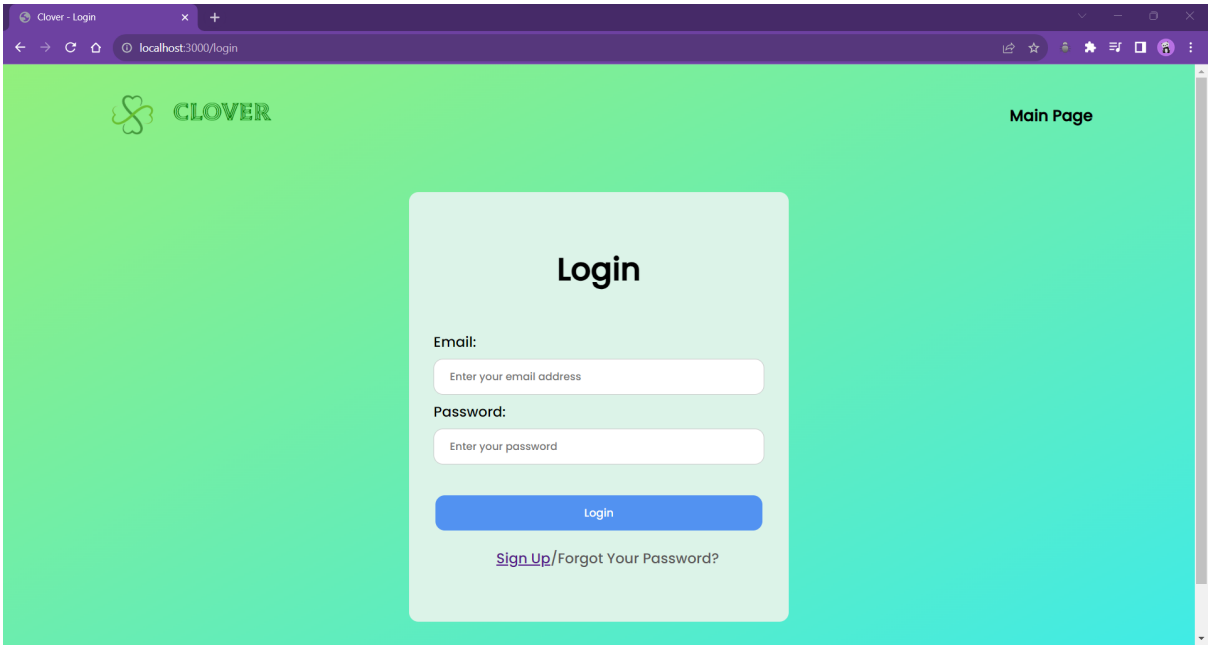
APPENDIX B

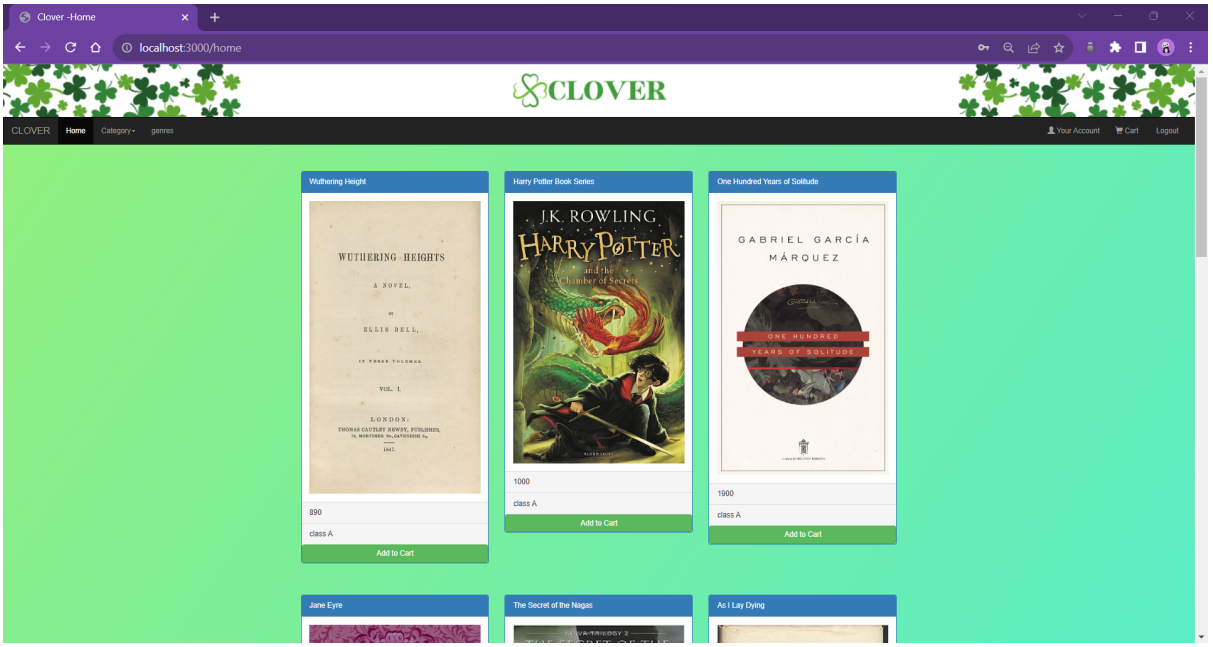
SAMPLE SCREENSHOT

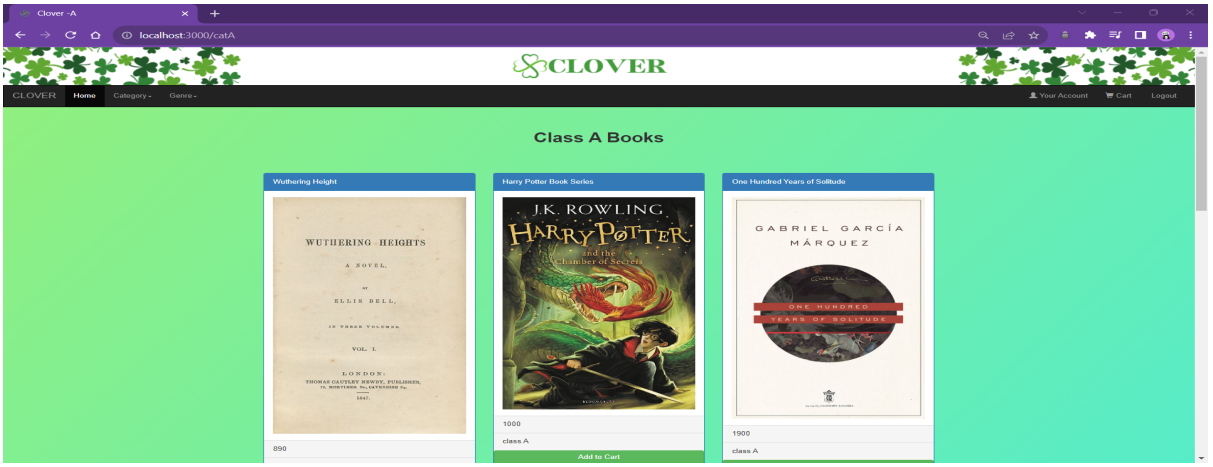
USER'S VIEW PAGES

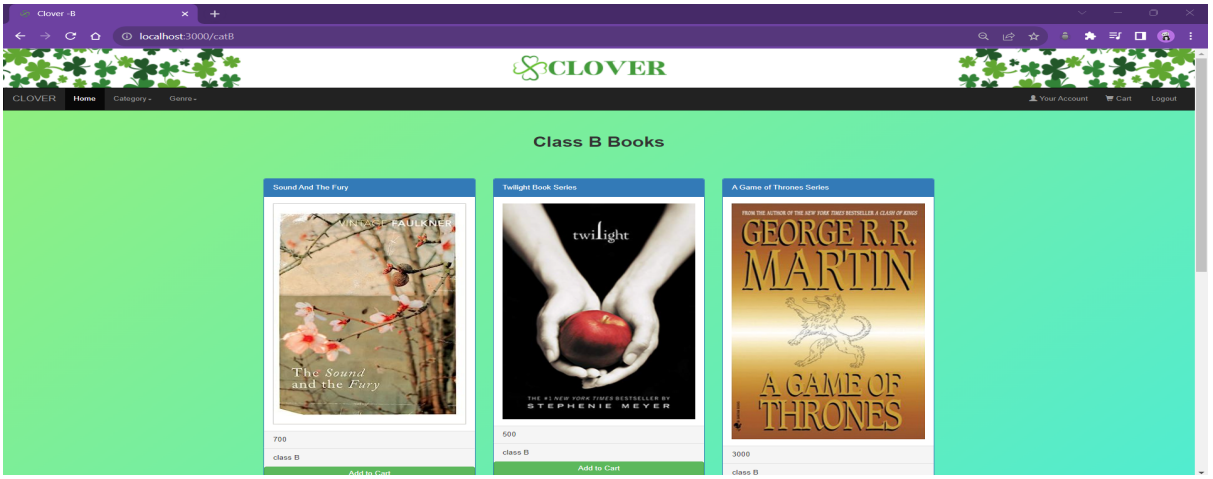


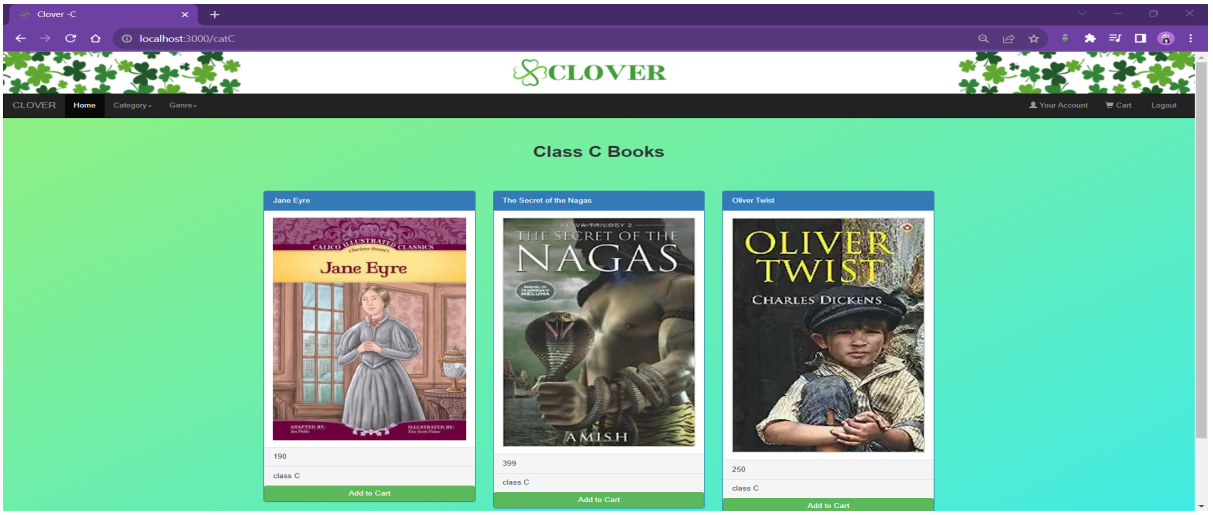


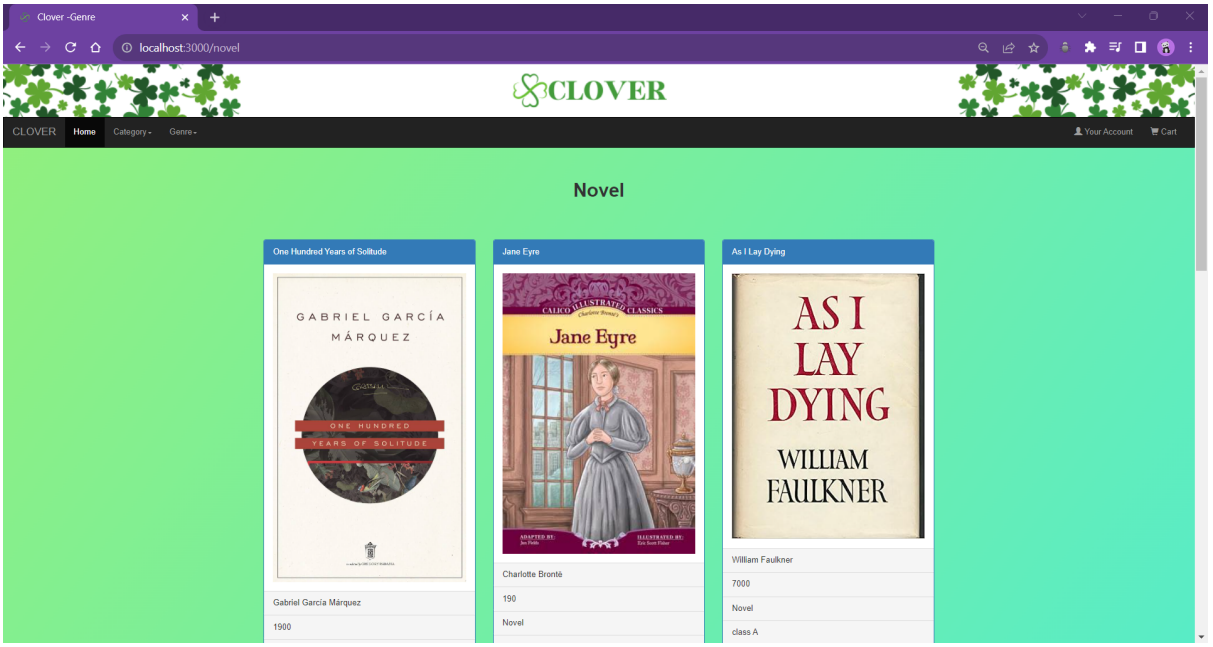

Clover - Login
localhost:3000/login
CLOVER
Main Page
Login
Email:
Enter your email address
Password:
Enter your password
Login
Sign Up/Forgot Your Password?

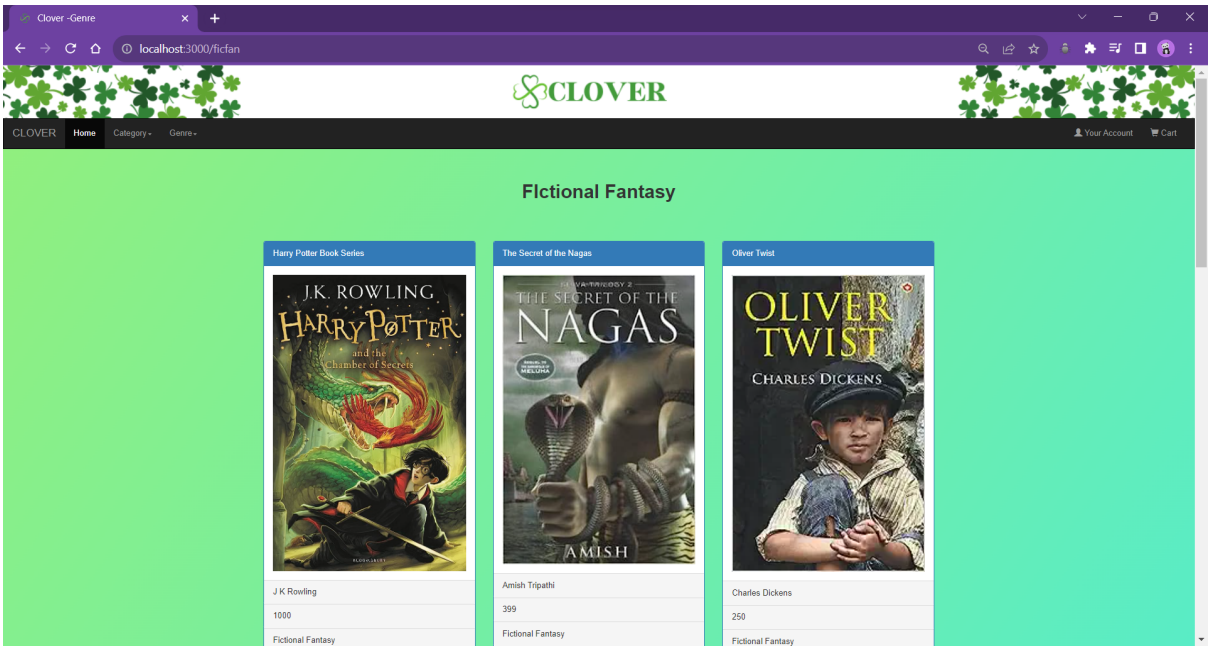

Clover -Home
localhost:3000/home
CLOVER
CLOVER
Home
Category
genres
Your Account
Cart
Logout
Wuthering Height
WUTHERING HEIGHTS
A NOVEL
ELLIS BELL
890
class A
Add to Cart
Harry Potter Book Series
J.K. ROWLING
HARRY POTTER
1000
class A
Add to Cart
One Hundred Years of Solitude
GABRIEL GARCÍA MÁRQUEZ
ONE HUNDRED YEARS OF SOLITUDE
1900
class A
Add to Cart
Jane Eyre
The Secret of the Nagas
As I Lay Dying

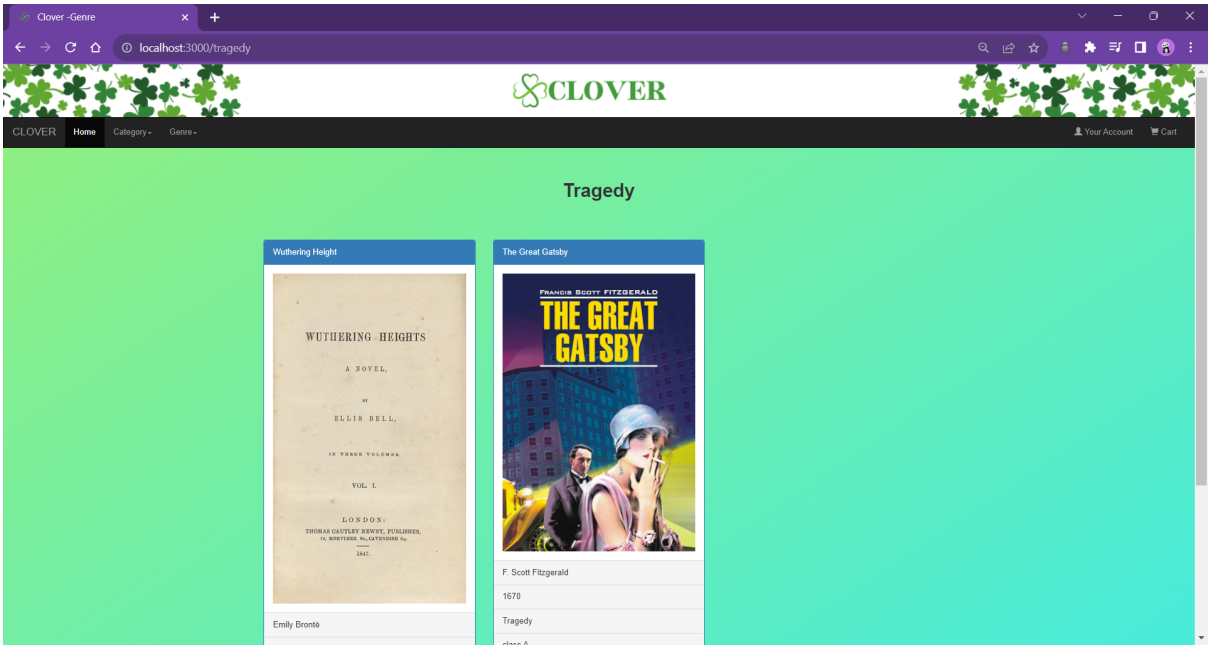

Clover -A
localhost:3000/catA
CLOVER
CLOVER
Home
Category
Genre
Your Account
Cart
Logout
Class A Books
Wuthering Height
WUTHERING HEIGHTS
A NOVEL
ELLIS BELL
890
Harry Potter Book Series
J.K. ROWLING
HARRY POTTER
and the Chamber of Secrets
1000
class A
Add to Cart
One Hundred Years of Solitude
GABRIEL GARCÍA MÁRQUEZ
ONE HUNDRED YEARS OF SOLITUDE
1900
class A

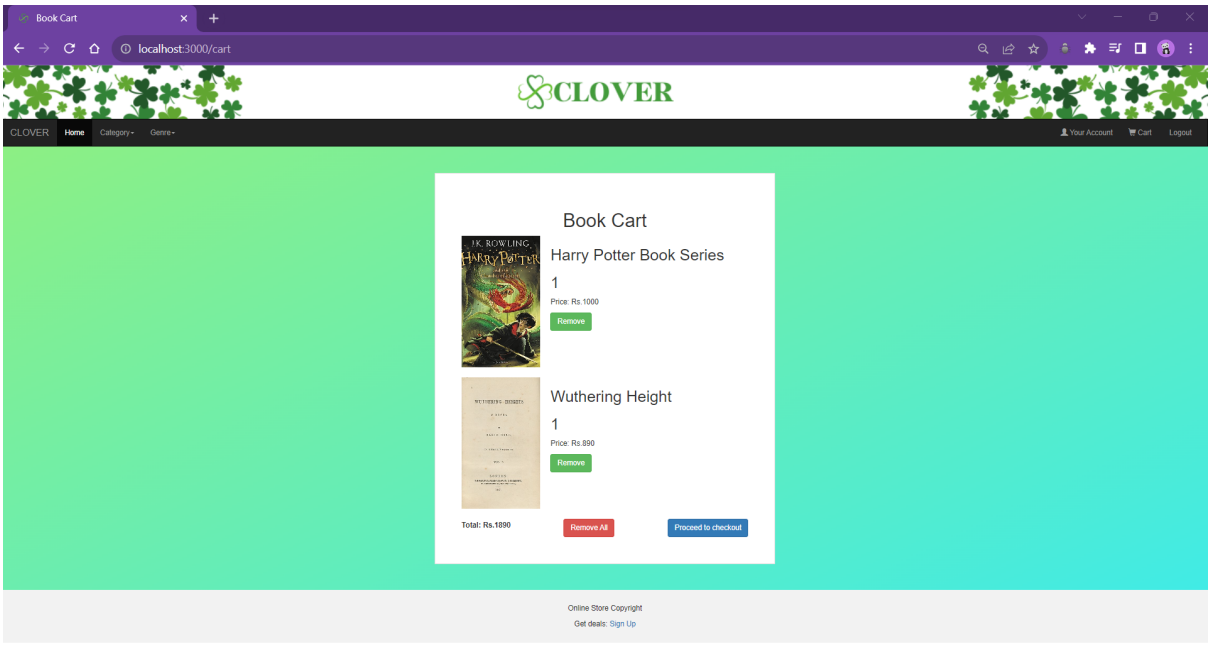

Clover -B
localhost:3000/catB
CLOVER
CLOVER
Home
Category
Genre
Your Account
Cart
Logout
Class B Books
Sound And The Fury
WILLIAM FAULKNER
The Sound and the Fury
700
class B
Twilight Book Series
twilight
STEPHENIE MEYER
500
class B
Add to Cart
A Game of Thrones Series
GEORGE R. R. MARTIN
A GAME OF THRONES
3000
class B

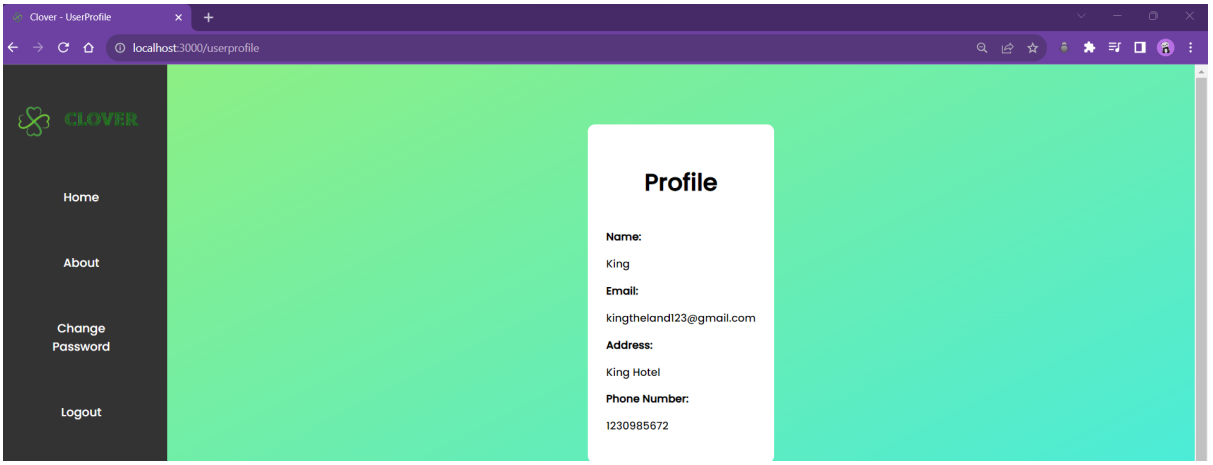

Clover -C
localhost:3000/catC
CLOVER
CLOVER
Home
Category
Genre
Your Account
Cart
Logout
Class C Books
Jane Eyre
Jane Eyre
190
class C
Add to Cart
The Secret of the Nagas
THE SECRET OF THE NAGAS
AMISH
399
class C
Add to Cart
Oliver Twist
OLIVER TWIST
CHARLES DICKENS
250
class C
Add to Cart

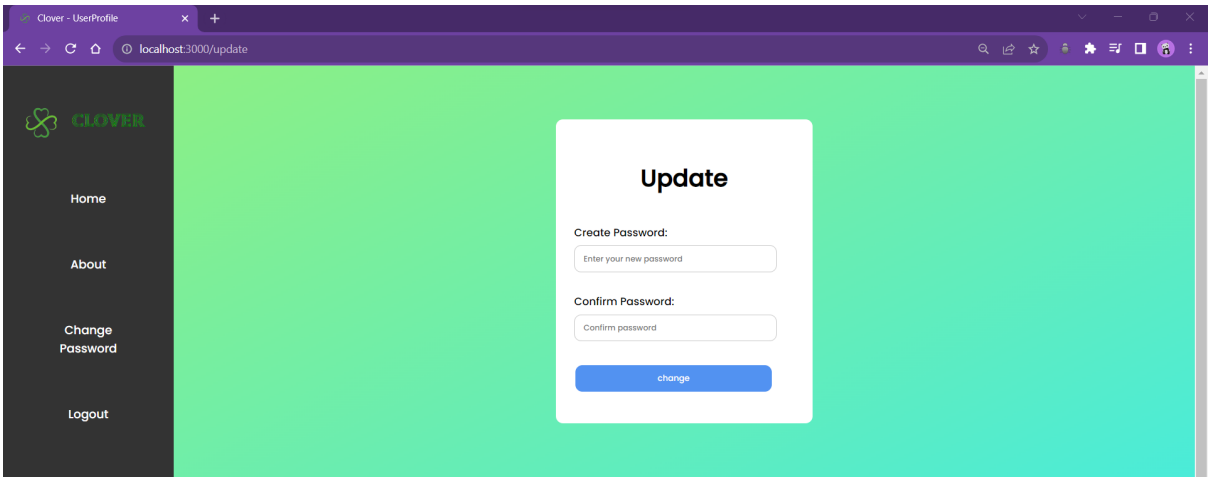

Clover -Genre
localhost:3000/novel
CLOVER
CLOVER
Home
Category
Genre
Your Account
Cart
Novel
One Hundred Years of Solitude
GABRIEL GARCÍA MÁRQUEZ
ONE HUNDRED YEARS OF SOLITUDE
Gabriel García Márquez
1900
Jane Eyre
CALICO ILLUSTRATED CLASSICS
Jane Eyre
Charlotte Brontë
190
Novel
As I Lay Dying
AS I LAY DYING
WILLIAM FAULKNER
William Faulkner
7000
Novel
class A


Clover -Genre
localhost:3000/ficfan
CLOVER
CLOVER
Home
Category
Genre
Your Account
Cart
Fictional Fantasy
Harry Potter Book Series
J.K. ROWLING
HARRY POTTER
and the Chamber of Secrets
J K Rowling
1000
Fictional Fantasy
The Secret of the Nagas
THE SECRET OF THE NAGAS
AMISH
Amish Tripathi
399
Fictional Fantasy
Oliver Twist
OLIVER TWIST
CHARLES DICKENS
Charles Dickens
250
Fictional Fantasy

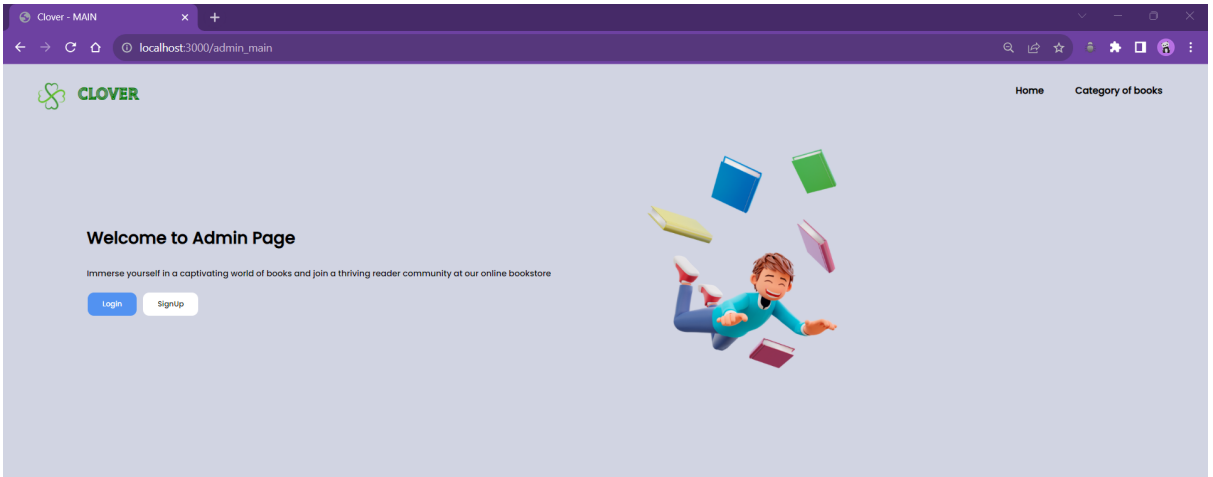

Clover -Genre
localhost:3000/tragedy
CLOVER
CLOVER
Home
Category
Genre
Your Account
Cart
Tragedy
Wuthering Height
WUTHERING HEIGHTS
A NOVEL,
ELLIS BELL,
IN THREE VOLUMES.
VOL. I.
LONDON:
Emily Brontë
The Great Gatsby
FRANCIS SCOTT FITZGERALD
THE GREAT GATSBY
F. Scott Fitzgerald
1670
Tragedy

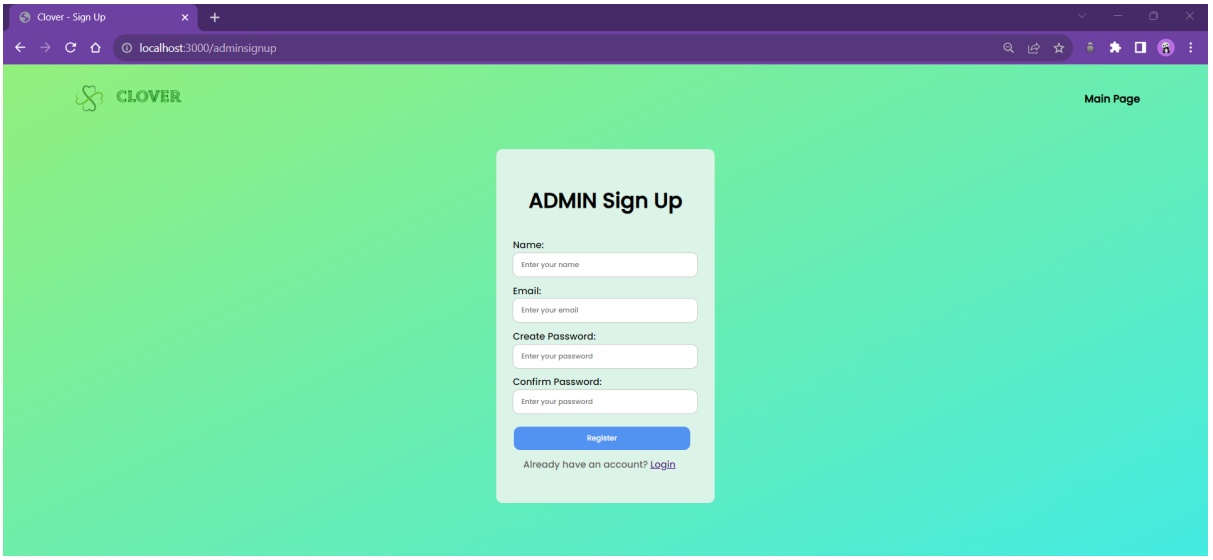

Book Cart
localhost:3000/cart
CLOVER
CLOVER
Home
Category
Genre
Your Account
Cart
Logout
Book Cart
Harry Potter Book Series
1
Price: Rs.1000
Remove
Wuthering Height
1
Price: Rs.890
Remove
Total: Rs.1890
Remove All
Proceed to checkout
Online Store Copyright
Get deals: Sign Up

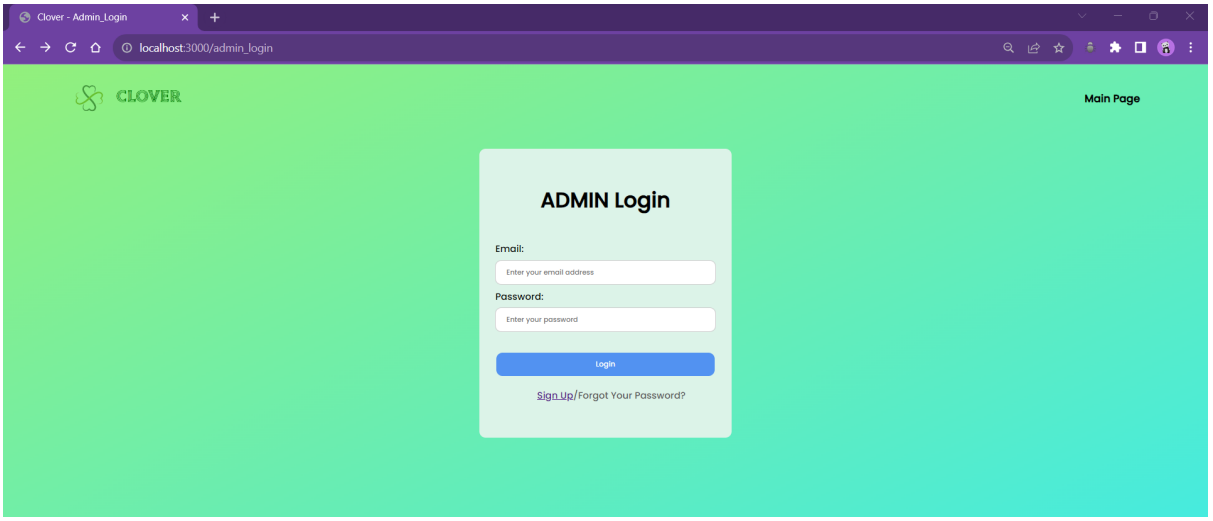


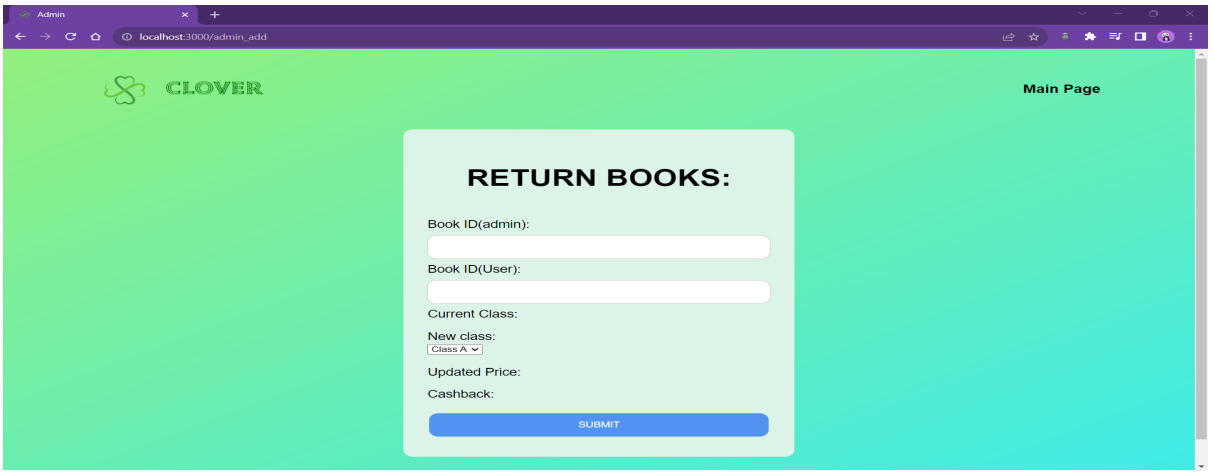


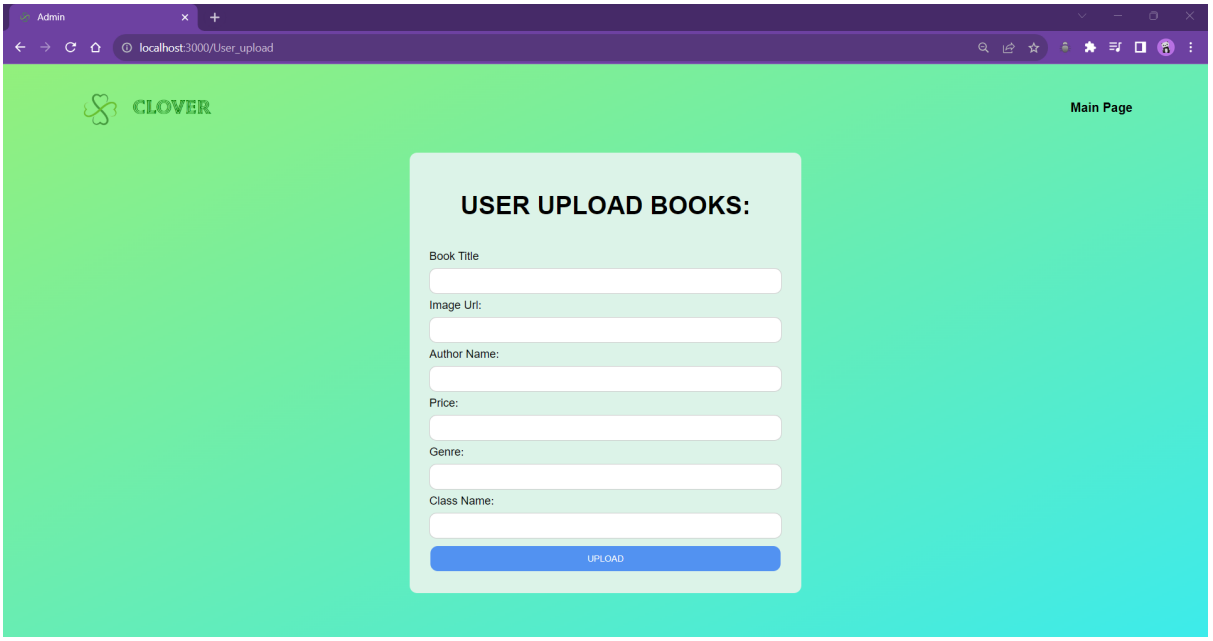
ADMIN'S VIEW PAGES

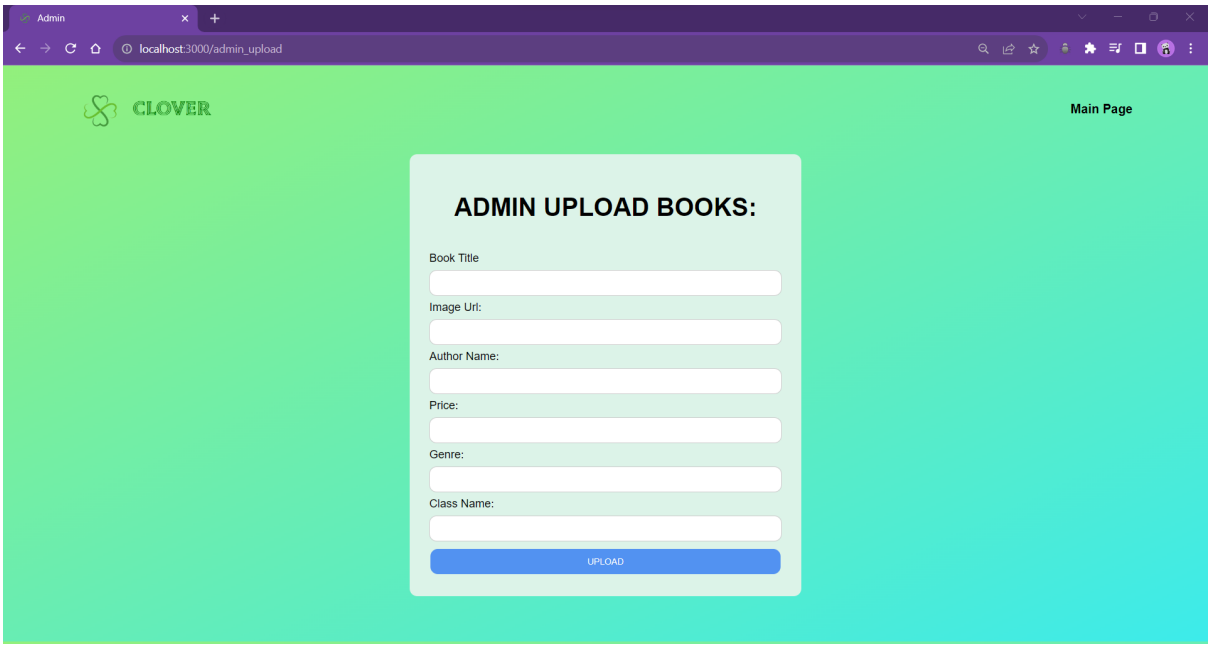



Clover - Sign Up
localhost:3000/adminsignup
CLOVER
Main Page
ADMIN Sign Up
Name:
Enter your name
Email:
Enter your email
Create Password:
Enter your password
Confirm Password:
Enter your password
Register
Already have an account? Login

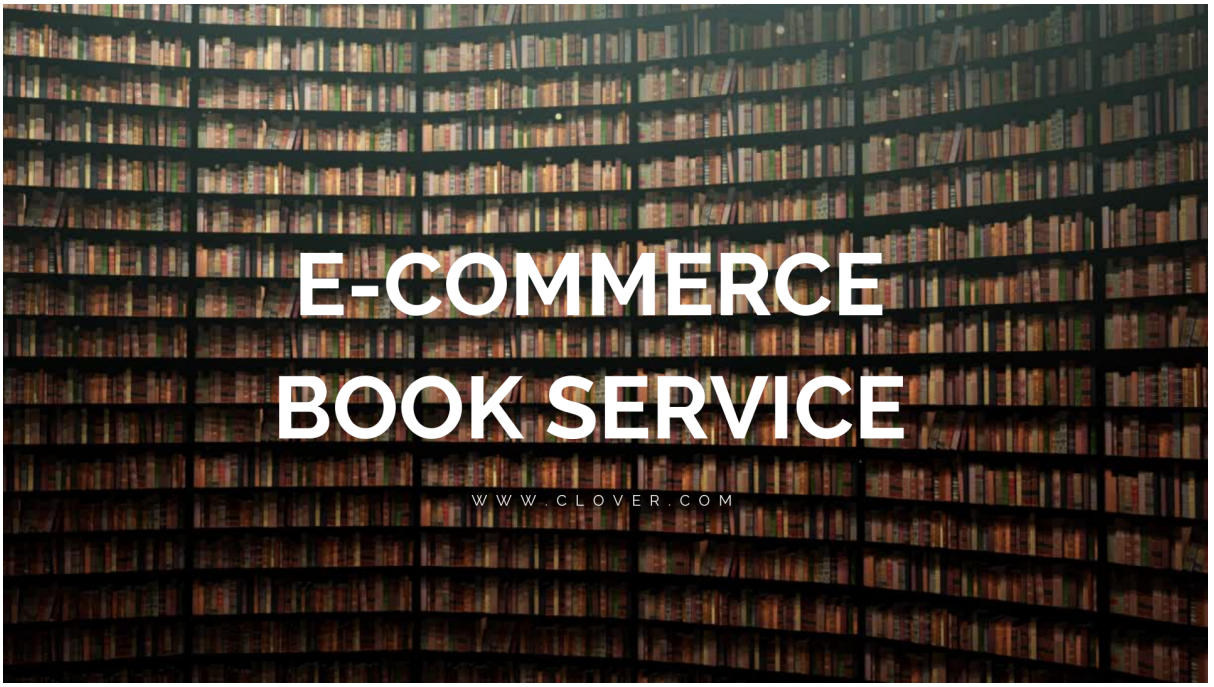

Clover - Admin_Login
localhost:3000/admin_login
CLOVER
Main Page
ADMIN Login
Email:
Enter your email address
Password:
Enter your password
Login
Sign Up/Forgot Your Password?


Admin
localhost:3000/admin_add
CLOVER
Main Page
RETURN BOOKS:
Book ID(admin):
Book ID(User):
Current Class:
New class:
Class A
Updated Price:
Cashback:
SUBMIT


Admin
localhost:3000/User_upload
CLOVER
Main Page
USER UPLOAD BOOKS:
Book Title
Image Url:
Author Name:
Price:
Genre:
Class Name:
UPLOAD


Admin
localhost:3000/admin_upload
CLOVER
Main Page
ADMIN UPLOAD BOOKS:
Book Title
Image Url:
Author Name:
Price:
Genre:
Class Name:
UPLOAD

APPENDIX C
SLIDES



PROBLEM STATEMENT (OBJECTIVE)

- Create an e-commerce platform for sustainable and affordable reading.
- Implement dynamic book categorization (Categories A, B, C, D).
- Incentivize responsible book consumption with cashback policy.
- Foster a win-win situation for readers and sellers.
- Measure success by adoption rate and positive impact on the environment

APPLICATIONS/RELEVANCE

- Sustainable reading solutions, reducing book waste and environmental impact.
- Innovative categorization system for efficient book classification.
- Encourages responsible book consumption through cashback policy.
- User-friendly interface for seamless book browsing and purchasing.
- Community-driven book sharing fosters engagement among readers.
- Profitability for sellers, reaching a broader audience.
- Effective marketing strategies to promote sustainable reading.
- Measured success through adoption rate and positive impact.

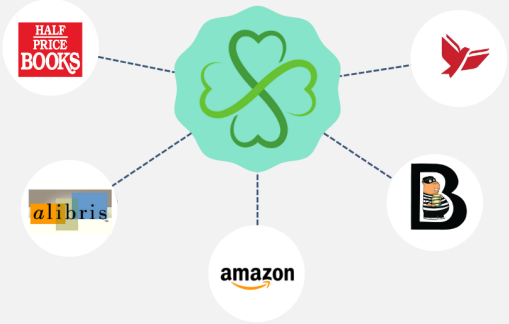
EXISTING / RELATED WORKS

Half Price Books

Half Price Books, Records, Magazines, Incorporated is a chain of new and used bookstores in the United States

Alibris

Alibris is an online store that sells new books, used books, out-of-print books, rare books



Amazon

Sells book at discount price

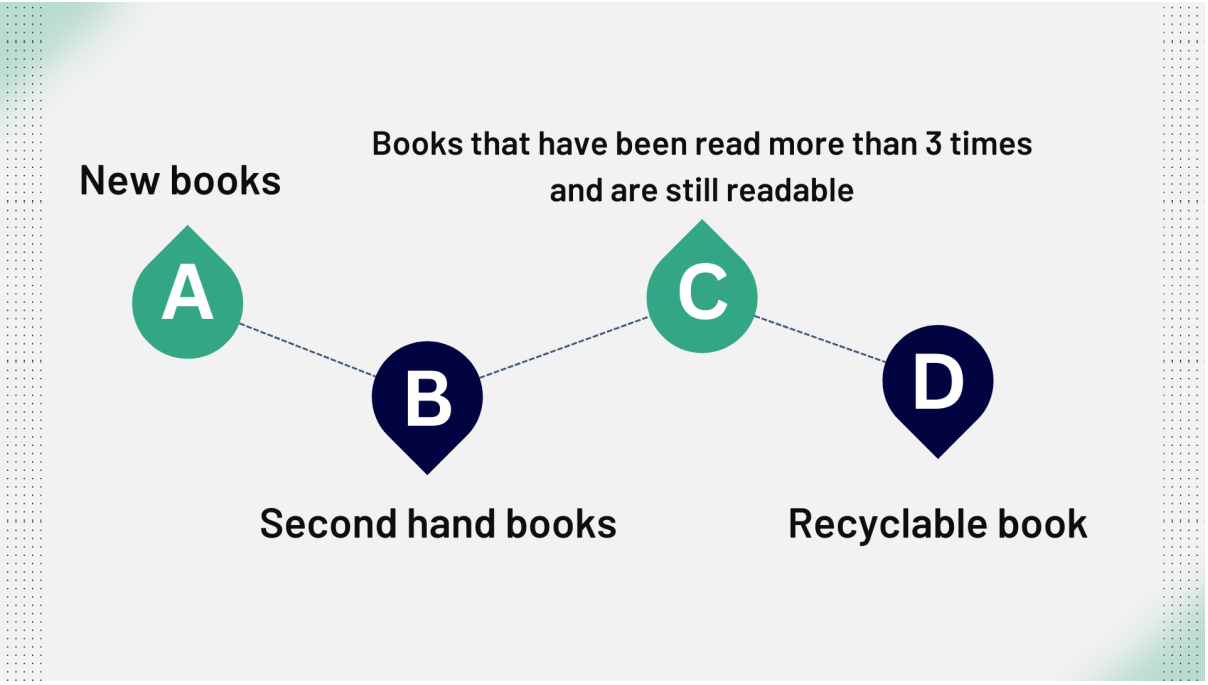
AbeBooks

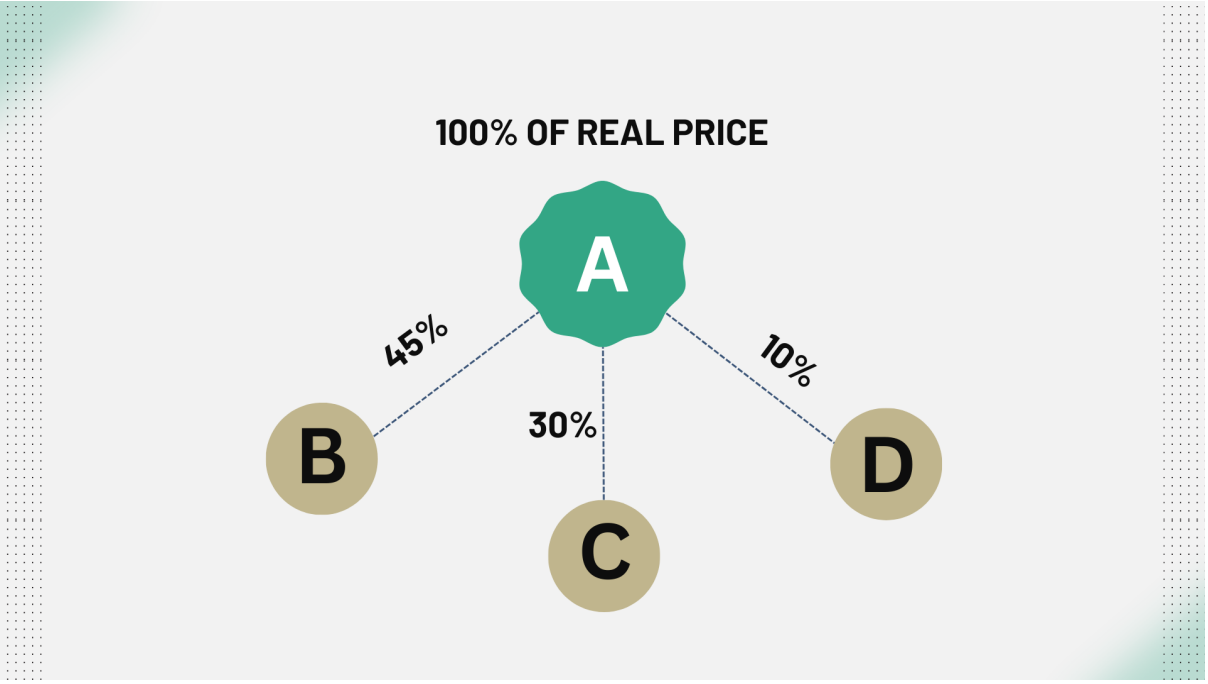
known for their epic selection of new and rare books, they also sell fine art & collectibles

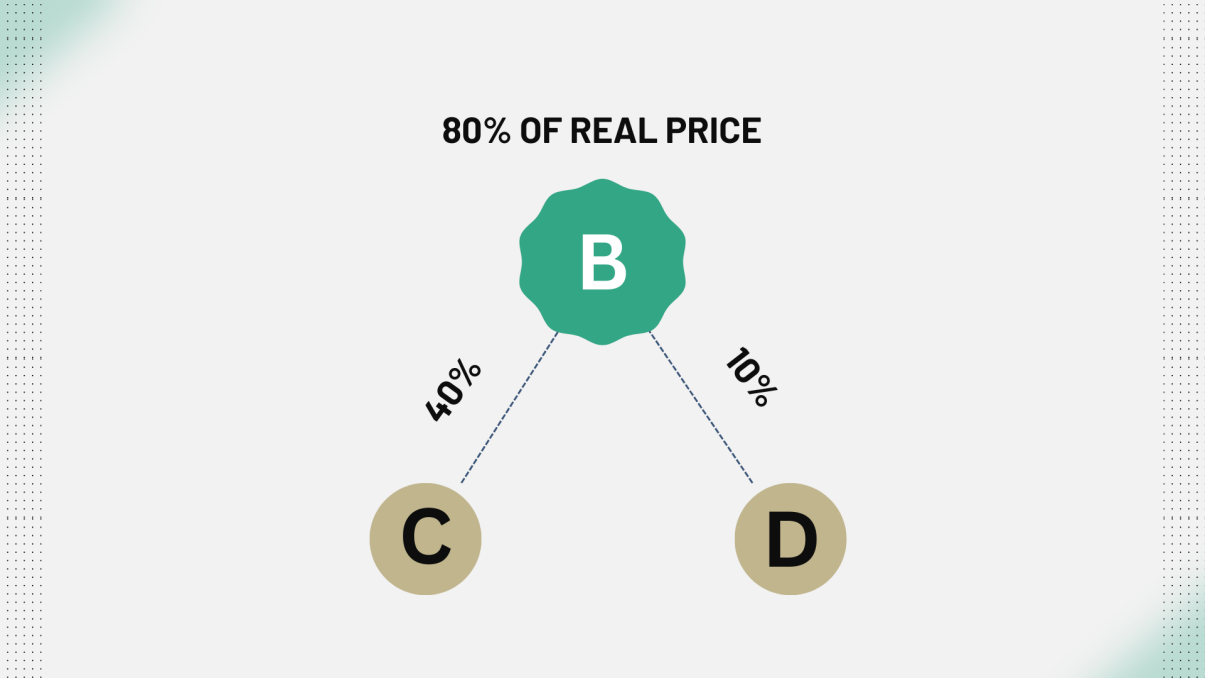
BookChor

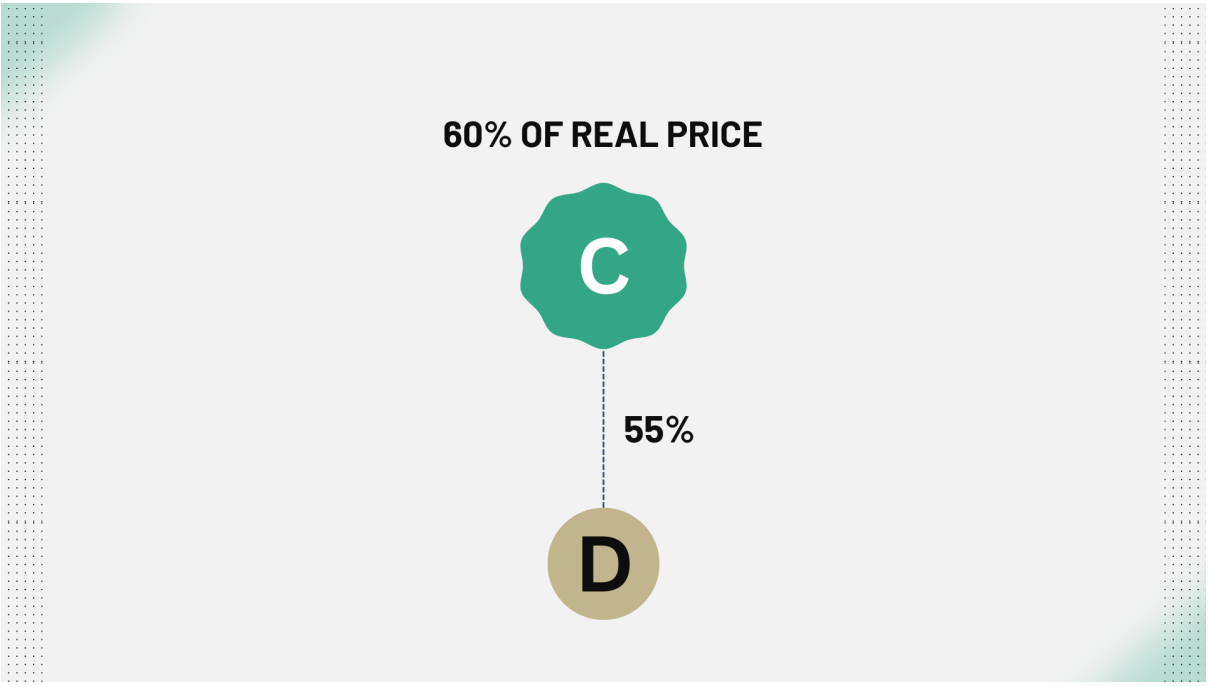
India's 1st Second Hand Books Online bookstore that offers a wide range of second-hand books, Old Books

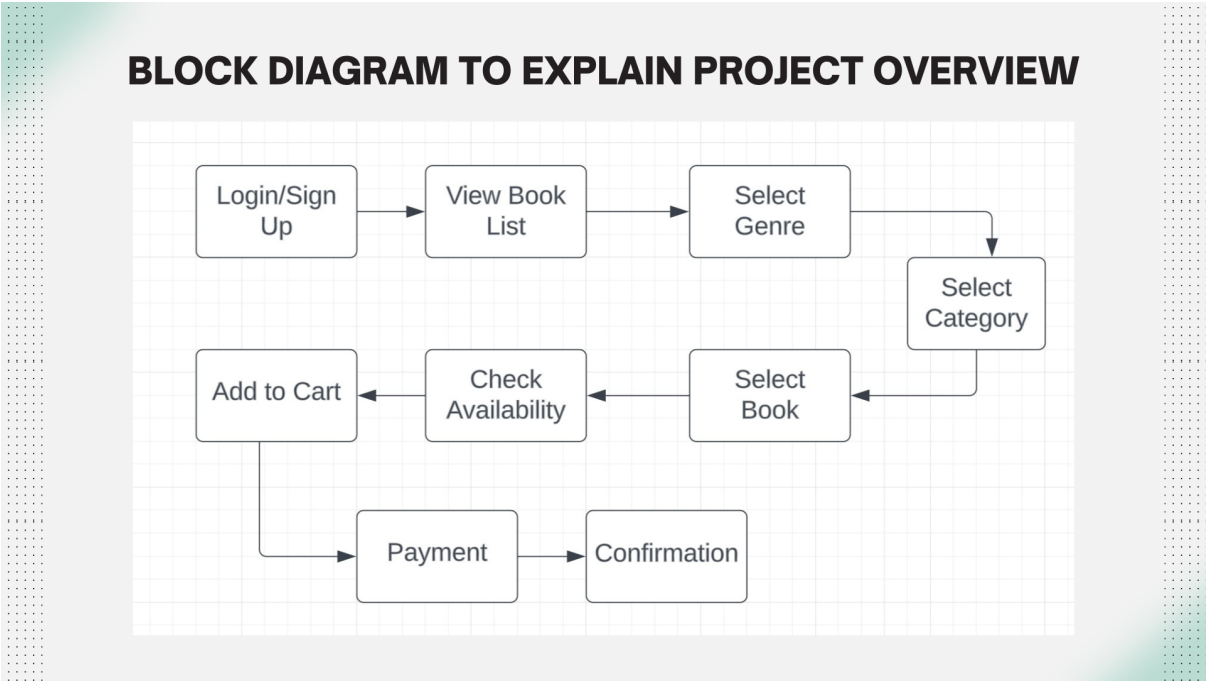
WHY?


New books
Books that have been read more than 3 times and are still readable
A
B
C
D
Second hand books
Recyclable book


100% OF REAL PRICE
A
45%
30%
10%
B
C
D


80% OF REAL PRICE
B
40%
10%
C
D


60% OF REAL PRICE
C
55%
D


BLOCK DIAGRAM TO EXPLAIN PROJECT OVERVIEW
Login/Sign Up
View Book List
Select Genre
Select Category
Add to Cart
Check Availability
Select Book
Payment
Confirmation

Modules name with description in a line

1. Customer

- Sign up/login in
- Edit profile
- View book
- Select category
- Select genre
- Add to cart

2. Admin

- Manage book price
- Add books to each category according to the conditions.
- Keep track of the book in the cart.

3. Book

- Has different categories
- Each has different genre
- Price will be given according to the categories

Modules name with description in a line

4. Category

Class A

- The books are new and not being reused.
- It is selled in original price.

Class B

- The books have been used once.
- Sold at 80% of orginal price

Class C

- The books have been used multiple times and there will be some damages.
- Sold at 60% of orginal price

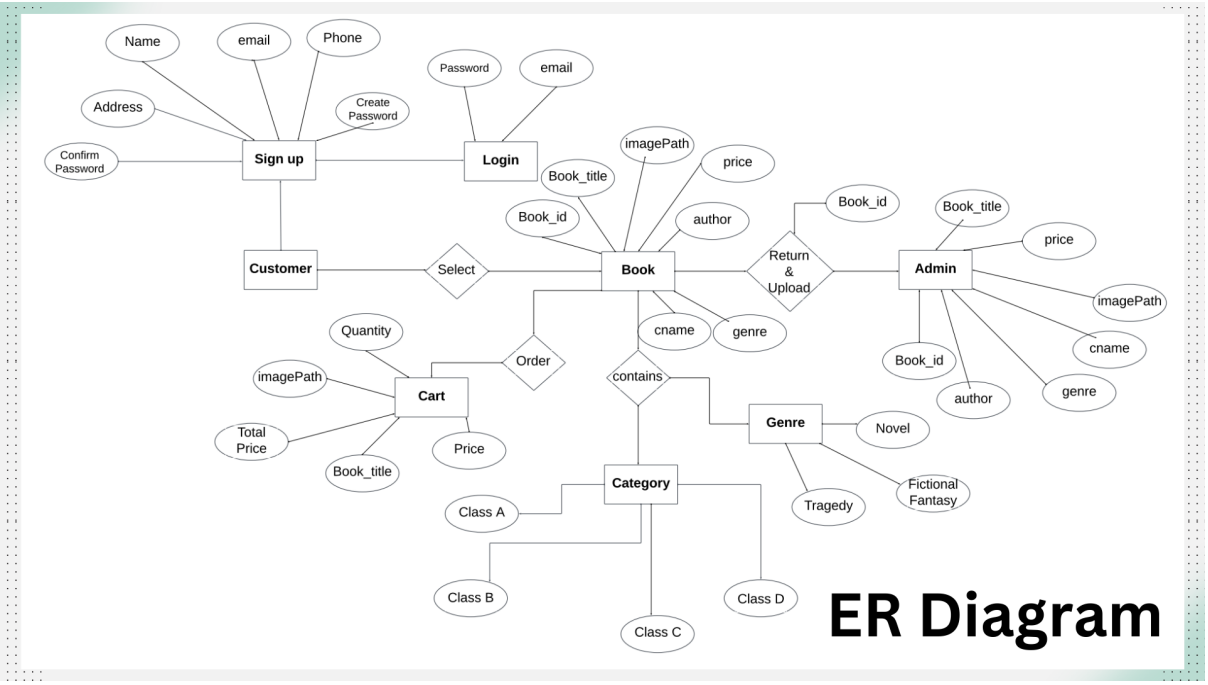
Class D

- The books have been used too many times and have to be recycled.

Modules name with description in a line

5. Cart

- Contains the orders.
- This is monitored by the admin



IMPLEMENTATION DETAILS LIKE PLATFORMS, IDE,DATABASE

- **IDE : VisualStudio**
- Visual Studio is an IDE from Microsoft.
- It is used to develop computer programs including websites, web apps, web services and mobile apps.
- It uses Microsoft software development platforms such as Windows API, Windows Forms, Windows Presentation Foundation, Windows Store and Microsoft Silverlight.

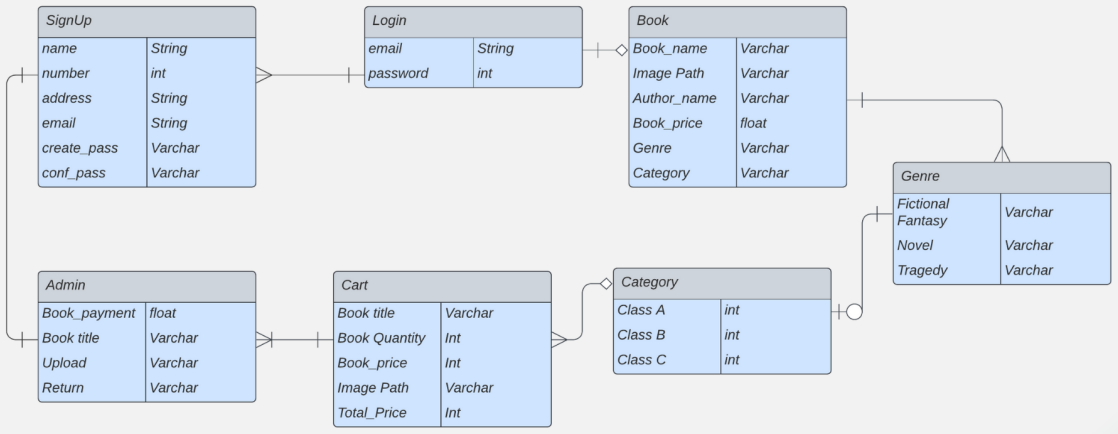
- **Database : MongoDB**
- It is a source-available cross-platform document-oriented database program. Classified as a NoSQL database program, MongoDB uses JSON-like documents with optional schemas.

IMPLEMENTATION DETAILS LIKE PLATFORMS, IDE,DATABASE

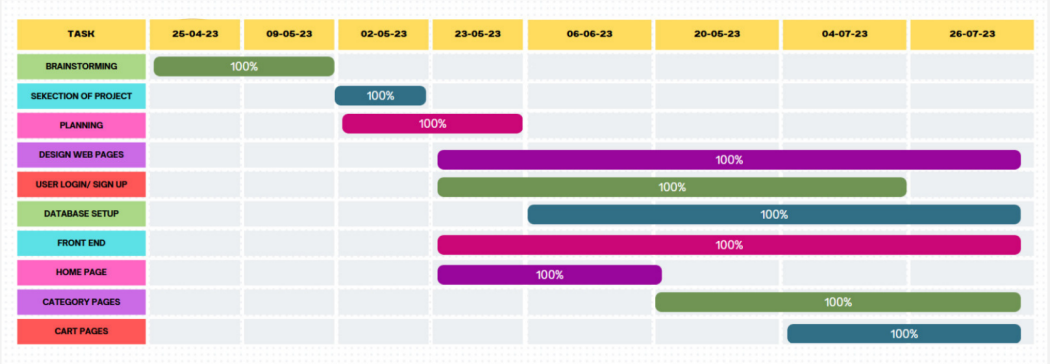
- **Platform : Node.js**
- Node.js is a cross-platform, open-source server environment that can run on Windows, Linux, Unix, macOS, and more.
- Node.js is a back-end JavaScript runtime environment, runs on the V8 JavaScript Engine, and executes JavaScript code outside a web browser.

- **Back-end Framework: Express.js**
- Express.js, or simply Express,is a back end web application framework for building RESTful APIs with Node.js, released as free and open-source software under the MIT License. It is designed for building web applications and APIs.

DATABASE DESIGN



PROJECT SCHEDULE WITH A GANTT CHART AND MAIN RESPONSIBILITIES OF EACH MEMBER (TASK ASSIGNMENT)



THANK YOU

APPENDIX D

RAJAGIRI SCHOOL OF ENGINEERING AND TECHNOLOGY (AUTONOMOUS)

DEPARTMENT OF INFORMATION TECHNOLOGY

PROGRAMME: INFORMATION TECHNOLOGY

VISION

To evolve into a department of excellence in information technology by the creation and exchange of knowledge through leading-edge research, innovation and services, which will in turn contribute towards solving complex societal problems and thus building a peaceful and prosperous mankind.

MISSION

To impart high-quality technical education, research training, professionalism and strong ethical values in the young minds for ensuring their productive careers in industry and academia so as to work with a commitment to the betterment of mankind.

PROGRAM EDUCATIONAL OBJECTIVES (PEO)

Graduates of Information Technology program shall

PEO 1: Have strong technical foundation for successful professional careers and to evolve as key-players / entrepreneurs in the field of information technology.

PEO 2: Excel in analyzing, formulating and solving engineering problems to promote life-long learning, to develop applications, resulting in the betterment of the society.

PEO 3: Have leadership skills and awareness on professional ethics and codes.

PROGRAM OUTCOMES (PO)

Information Technology program students will be able to:

PO 1. Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO 2. **Problem analysis**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO 3. **Design/development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO 4. Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO 5. Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

PO 6. The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO 7. Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO 8. Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO 9. Individual and teamwork: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO 10. Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO 11. Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO 12. Life-long learning: Recognize the need for and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

PROGRAM SPECIFIC OUTCOMES (PSO)

Information Technology program students will be able to:

PSO1: Acquire skills to design, analyze and develop algorithms and implement them using high-level programming languages**.**

PSO2: Contribute their engineering skills in computing and information engineering domains like network design and administration, database design and knowledge engineering.

PSO3: Develop strong skills in systematic planning, developing, testing implementing and providing IT solutions for different domains which helps in the betterment of life.

COURSE OBJECTIVES:

This course is designed for enabling the students to apply the knowledge to address the real-world situations/problems and find solutions. The course is also intended to estimate the ability of the students in transforming theoretical knowledge studied as part of the curriculum so far into a working model of a software system. The students are expected to design and develop a software/hardware project to innovatively solve a real-world problem.

COURSE OUTCOMES:

After completion of the course the student will be able to

Sl.NO	**DESCRIPTION**	**Blooms' Taxonomy Level**
CO1	Make use of acquired knowledge within the selected area of technology for project development.	Level 3: Apply
CO2	Identify, discuss and justify the technical aspects and design aspects of the project with a systematic approach.	Level 3: Apply
CO3	Interpret, improve and refine technical aspects for engineering projects.	Level 3: Apply
CO4	Associate with a team as an effective team player for the development of technical projects.	Level 3: Apply
CO5	Report effectively the project related activities and findings.	Level 2: Understand

CO-PO AND CO-PSO MAPPING

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO 10	PO 11	PO 12	PSO 1	PSO 2	PSO3
CO1	3	3	3	3	3	3	3	3				3	3	3	3
CO2	3	3	3	3	3		2	3		3	2	3	3		3
CO3	3	3	3	3	3	2	3	3		2	3	3	2	2	2
CO4	3	3	2	2				3	3	3	3	3			
CO5	3				2			3	2	3	2	3			

3/2/1: high/medium/low

RSET
RAJAGIRI SCHOOL OF ENGINEERING & TECHNOLOGY
(AUTONOMOUS)

MINI PROJECT REPORT

STUDENT PLACEMENT MANAGEMENT SYSTEM

SUBMITTED BY

Mr. ASHWIN SAJI KUMAR

Ms. MARY MISHAL FRANCIS

Ms. ASHLEEN SUNIL MATHEW

DEPARTMENT OF INFORMATION TECHNOLOGY

STUDENT PLACEMENT MANAGEMENT SYSTEM

A MINIPROJECT REPORT

Submitted by

Mr. Ashwin Saji Kumar

Ms. Mary Mishal Francis

Ms. Ashleen Sunil Mathew

to

the APJ Abdul Kalam Technological University in partial fulfillment of the requirements of the award of the Degree

of

Bachelor of Technology

in

Information Technology

Department of Information Technology

Rajagiri School of Engineering & Technology
Rajagiri Valley, Kakkanad, Kochi-39

JULY 2023

DEPARTMENT OF INFORMATION TECHNOLOGY

RAJAGIRI SCHOOL OF ENGINEERING & TECHNOLOGY (AUTONOMOUS)

RAJAGIRI VALLEY, KAKKANAD, KOCHI- 39

CERTIFICATE

This is to certify that the report entitled "Student Placement Management System" is a bonafide record of the work done by Mr. Ashwin Saji Kumar, Ms. Mary Mishal Francis and Ms. Ashleen Sunil Mathew, University Register Number RET20IT022, RET20IT048, RET20IT020 in partial fulfillment of the award of the Degree of Bachelor of Technology in Information Technology at Rajagiri School of Engineering & Technology, Kakkanad, Kochi during the academic year 2022-23.

Dr. Ranju S Kartha	Dr. Priya Mariam Raju	Dr. Neeba E A
Project Guide	Project Coordinator	Head of the Department
Assistant Professor	Assistant Professor	Associate Professor
Dept of IT	Dept of IT	Dept of IT
RSET	RSET	RSET

Submitted for the practical examination conducted on……

Internal Examiner

ACKNOWLEDGEMENT

A strong foundation is a necessity to step on and to start building and expanding. We are thankful for the management and our Principal Dr. P. S. Sreejith who has ensured this strong foundation for us, students, to start building our lives on.

We thank Dr. Neeba E A, Head of the Department, Department of Information Technology, from the bottom of our hearts for being present at every stage of this journey to help and support us and make this project successful.

We also extend our sincere gratitude to our miniproject coordinator, Dr. Priya Mariam Raju, Asst. Professor, Department of Information Technology who coordinated us throughout the project.

This project would not have seen completion if it was not for the guidance from our guide, Dr Ranju S Kartha, Asst. Professor, Department of Information Technology, who shared her knowledge and advice throughout the project. We extend our sincere gratitude to her for being the great mentor she is.

Last but certainly not least we thank our teachers and friends whose help and support gave us the momentum to complete this work

ABSTRACT

A placement management system is a software tool designed to facilitate the process of managing student placements within an organization. This system is commonly used by educational institutions and companies that offer job opportunities to students. It can help streamline the hiring process and match students with companies that fit their interests, resulting in better placement outcomes for both the students and the companies.

The system provides a more seamless and efficient placement experience for students, employers, and administrators, enhancing overall satisfaction. The users involved in the placement management system include Administrators/Placement Officers who are responsible for managing the placement process, including creating job postings, scheduling interviews, and managing student resumes and applications. Companies or organizations offer job opportunities to students through the placement system, while students seek placement opportunities through the system. Faculty Advisors/Teachers provide guidance and support to students throughout the placement process and verify the data that students enter.

Technologies such as HTML, CSS, and JavaScript are used to create the user interface and improve the user experience. A database management system such as MongoDB is used to store and manage data, while Node.js is used as the runtime environment. This model is cross-platform and can be integrated with desktop/web applications as per the client's requirements.

TABLE OF CONTENTS

CHAPTER 1
INTRODUCTION

1.1 GENERAL BACKGROUND

The Placement Management System can be traced back to the evolution of recruitment and human resources practices. As organizations grew in size and complexity, the need for efficient management of the placement process became evident.In the past, recruitment and placement processes were primarily manual and paper-based. Organizations relied on physical job postings, printed resumes, and manual record-keeping to manage applicants and hiring procedures. These processes were time-consuming, prone to errors, and lacked centralized data management. The emergence of computer technology in the late 20th century led to the development of Applicant Tracking Systems (ATS). ATS software aimed to streamline applicant management by digitizing resumes, automating candidate tracking, and providing basic reporting capabilities. ATS marked a significant advancement in managing placements, but they primarily focused on the early stages of the recruitment process.

Over time, HR technology continued to evolve, driven by the need for more comprehensive and efficient placement management. HR software solutions began to incorporate features beyond ATS, such as interview scheduling, candidate evaluation, and reporting. These solutions provided more holistic support for the entire placement process.

Placement Management Systems started integrating with other HR systems, such as HRIS (Human Resource Information Systems) and payroll software, to create a seamless flow of data between various HR functions. This integration facilitated a more cohesive and efficient HR ecosystem. The digital age brought with it a wealth of data, and advancements in data analytics allowed organizations to make data-driven decisions in the placement process. Placement Management Systems started incorporating analytics tools to evaluate recruitment strategies, identify bottlenecks, and optimize hiring outcomes.

1.2 PROJECT OBJECTIVE

The objective of a Placement Management System is to efficiently acquire top talent, enhance candidate experience, centralize and automate data management, enable data-driven decision-making, ensure compliance and data security, foster collaboration among stakeholders, integrate with other HR systems, manage a talent pool, save time through automation, and scale with organizational growth. By achieving these goals, the system optimizes the recruitment process, streamlines operations, and facilitates successful talent acquisition for the organization.

1.3 SCOPE

The scope of a Placement Management System includes job posting and applicant management, interview scheduling, candidate evaluation and selection, analytics, and reporting. It streamlines the placement process, manages applicants, facilitates interviews, evaluates candidates, ensures compliance, maintains data security, integrates with HR systems, screens resumes, promotes collaboration, and enables data-driven decision-making. It encompasses various functionalities and features tailored to meet the needs of both students and recruiters.

The system allows students to create and manage their profiles, including academic information, skills, achievements, and resumes. This ensures that their profiles are updated and readily available for potential employers.

Additionally, any changes in the students' personal details or marks can be easily updated by the students themselves within their profiles, eliminating the need to contact the placement cell.

The system enables employers to post job opportunities and manage their company profiles, providing a platform for recruiters to connect with students for internships, part-time jobs, and full-time positions. Students can apply for job postings and internships directly through the system, streamlining the application process for efficient review by recruiters.

Moreover, the system facilitates interview scheduling between students and recruiters, manages the placement process, and tracks student placements. Some systems may also include career guidance and counseling modules to assist students in exploring career paths, preparing for interviews, and enhancing their employability.

The system captures data on placements, placement rates and other relevant metrics. It offers reporting capabilities to provide insights into the effectiveness of the college's placement efforts. Additionally, the system may offer support for alumni seeking job opportunities, enabling them to access job postings and career services even after graduation.

CHAPTER 2
LITERATURE SURVEY

This chapter discusses the various technologies which have been used to create a student placement management system.

2.1 Applicant Tracking Systems

A Standalone Applicant Tracking System (ATS) is a specialized software designed to streamline the recruitment and hiring process by managing the application and candidate tracking. These systems focus primarily on applicant management, from the initial submission of resumes to the final stages of candidate selection.

2.1.1 LIMITATIONS

- Limited Functionality: Standalone ATS may lack comprehensive features beyond applicant management. They may not provide functionalities for interview scheduling, candidate evaluation, onboarding, or integration with other HR systems
- Lack of Flexibility: Some Standalone ATS may have rigid workflows that may not align with an organization's unique hiring process.

2.2 Spreadsheet-Based Systems

Referring to the use of spreadsheets for managing the placement process, these systems rely on manual data entry and formulas to track and organize information related to job postings, applicants, interviews, and candidate selection. Spreadsheets allow for organizing data in tabular formats, making it possible to create separate sheets or tabs for different aspects of the placement process, such as job postings, applicant details, and interview schedules. They offer flexibility and customization options, enabling organizations to design their layouts and workflows based on their specific placement requirements. Organizations can define columns, labels, and formulas according to their needs, but they rely on

manual data entry for recording and updating information. Recruiters and hiring managers need to manually input data, such as candidate names, contact details, work experience, and interview feedback, into the spreadsheet.

2.2.1 LIMITATIONS

- Limited Automation: Spreadsheet-Based Systems rely heavily on manual data entry and lack advanced automation capabilities. Tasks such as data calculations, sorting, and filtering require manual intervention and can be time-consuming.
- Data Integrity and Accuracy: Human error is a significant risk in Spreadsheet-Based Systems. Typos, incorrect formulas, or accidental data deletions can lead to data inconsistencies and inaccurate information, impacting decision-making and the overall reliability of the system.

CHAPTER 3
METHODOLOGY/THEORY/MODELLING/ EXPERIMENTATION

3.1 SYSTEM ANALYSIS AND DESIGN

The design of a Student Placement System involves creating a structured and efficient platform that facilitates the entire campus placement process for students, recruiters, and administrators. The system is designed with different user roles, such as students, recruiters, placement coordinators, and administrators.

Students are given the ability to create and update their profiles, providing essential academic details, skills, achievements, and resumes to showcase their qualifications effectively.

Recruiters, on the other hand, can create job postings, manage their company profiles, and specify the requirements for each job opportunity they offer to students.

The design of the system emphasizes scalability to accommodate the growing needs of the institution, ensuring it can handle increasing data and user demands effectively. Additionally, it provides a user-friendly interface for easy navigation and interaction, making it accessible and intuitive for all users.

Overall, the design of the Student Placement System aims to bridge the gap between students and potential employers, facilitating successful placements, and empowering students with valuable career-related resources and opportunities for their future professional endeavors.

3.1.1 SYSTEM ARCHITECTURE

The proposed placement management system has four user types- admin, staff, student and company. Figure 3.1.2 shows the overall system architecture with the functionalities of each user. Figure 3.1.2 shows that the admin can manage all other users in the system.

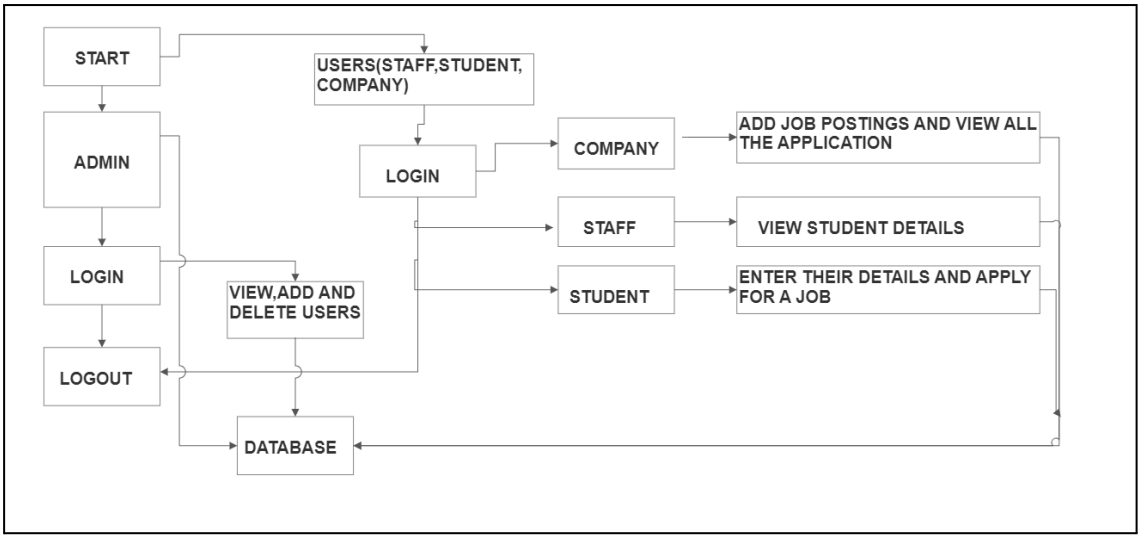


Figure 3.1.1 System Architecture

3.1.2 USE CASE DIAGRAM

The website allows the Admin user to log into the system. Admin is also responsible for managing staff,admin, and students.The Admin uploads csv file and creates accounts for all users. The next user, which is the student, can add their own details, and also apply for the job posted by the company. The user-staff can also register themselves and view the details of students.The next user, company will add the job listings and check all job applications.

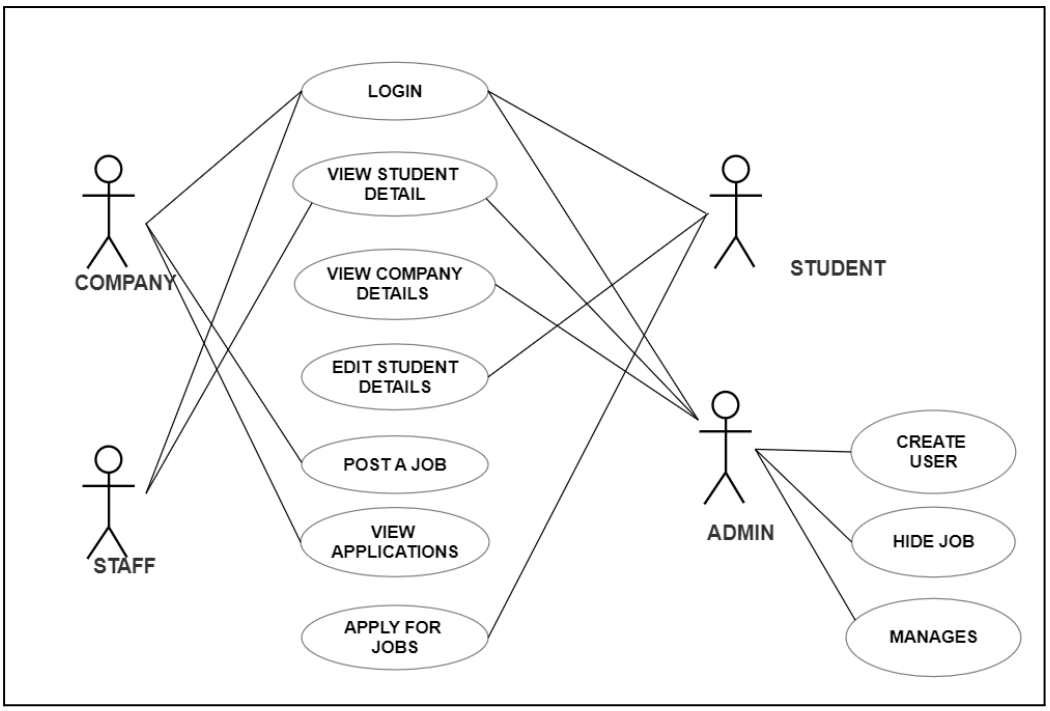


Figure 3.1.2: Use case diagram

3.2 MODULE 1- ADMIN

- Manage all details of students,staff and company.
- Assign accounts for each user .
- Admin can log in as staff or admin.
- Admin manages previous year statistics
- Admin imports csv file and then creates a account for user

3.3 MODULE 2-STAFF

The Staff user can

- Staff can login and then view details of students of their department

3.4 MODULE 3- STUDENT

The student user can

- Add student details into the student_details database.
- Apply for jobs required
- When a student user logins then we can see the placement statistics.

3.5 MODULE 3- COMPANY

The company user can

- Perform tests by uploading the photo of their skin into the patient_report database.

3.6 TABLE DESIGN

Student

Student table contains information about students after they create an account. The information includes student id, username, password, first name, last name, email and image.

Company

Company Table consists of information about the company. Information includes company id, name and its image.

Auth_user

Auth_user consists of information about the admin. It includes details of admin like id, password.

Company_job

Company_job includes information about the company's criteria. Company's criteria which include specific 10th mark,12th mark, cgpa, no of backpackers, role, city and other details which are companies requirements.

Job_application

Displays details of job application.

Student_detail

The details of students which are required while creating an account for the student. The details include id, uid, phone no., and all other details required for creating an account.

Staff

The fields given to staff is userid, branch and name

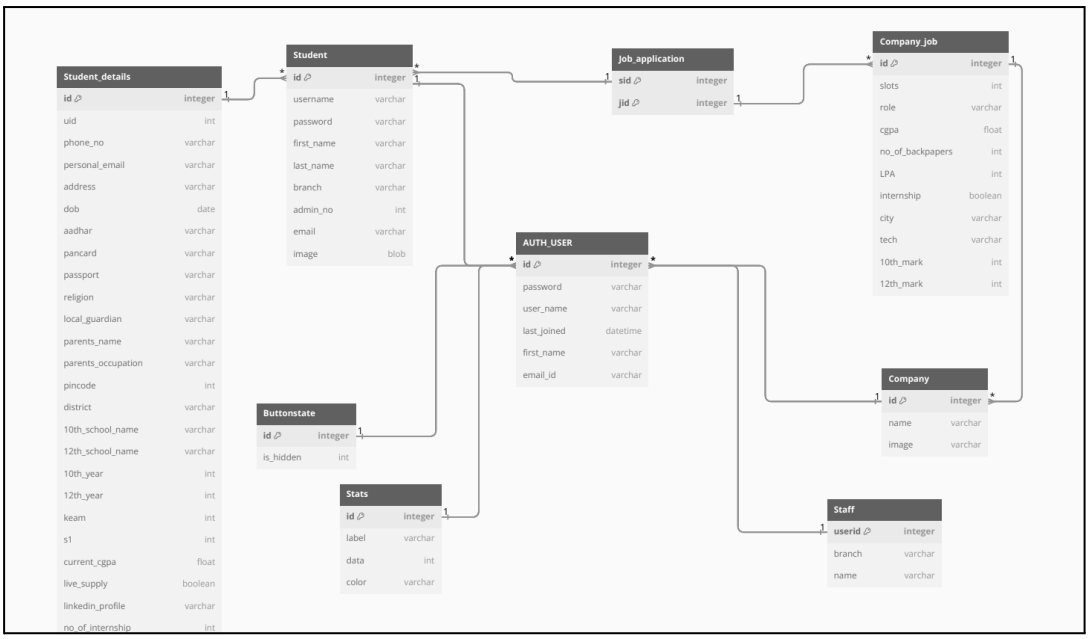


Figure 3.6: Database design

3.7 GUI DESIGN

3.7.1. Homepage

It enables 4 users (admin,staff, student and company) to access their login page.

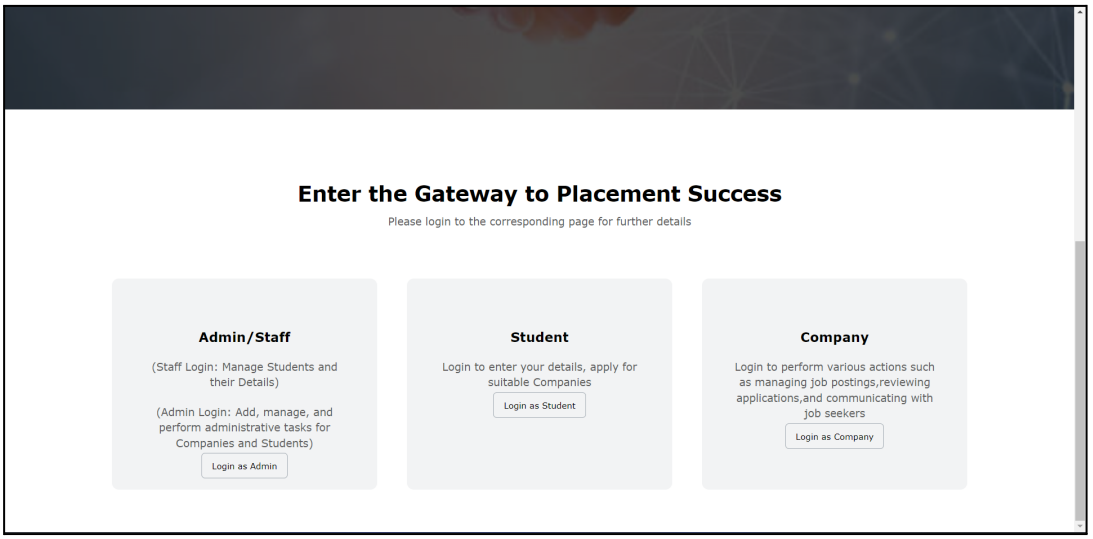


Figure 3.7.1.1 Depicts the Homepage

3.7.2. Student Login Page

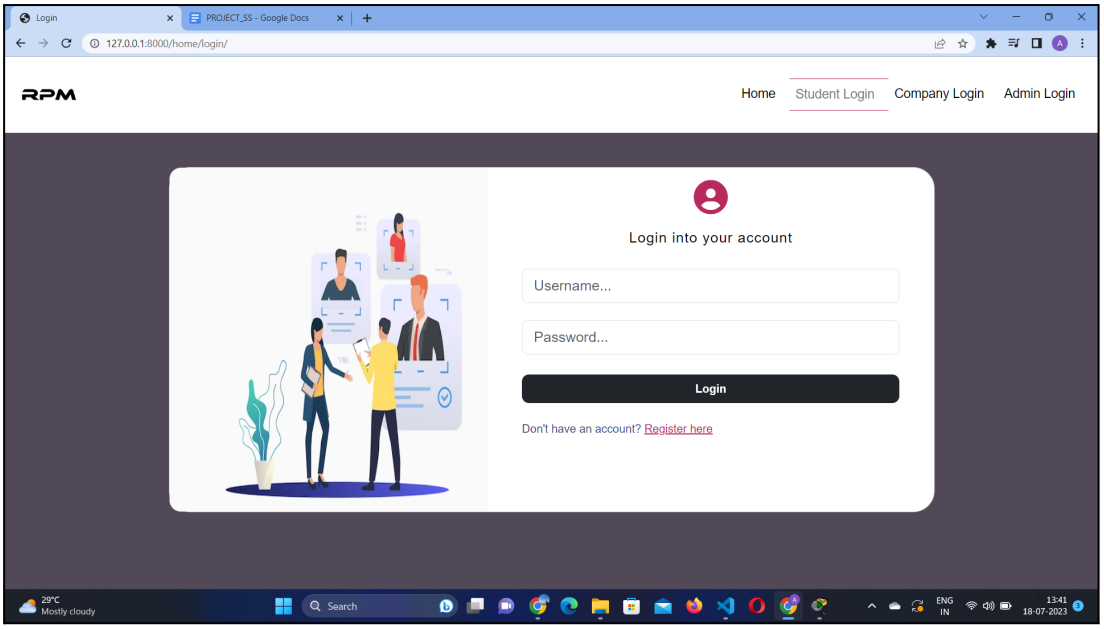


Figure 3.7.1.3 Depicts the Student Login Page

3.7.1.3 Company Login Page

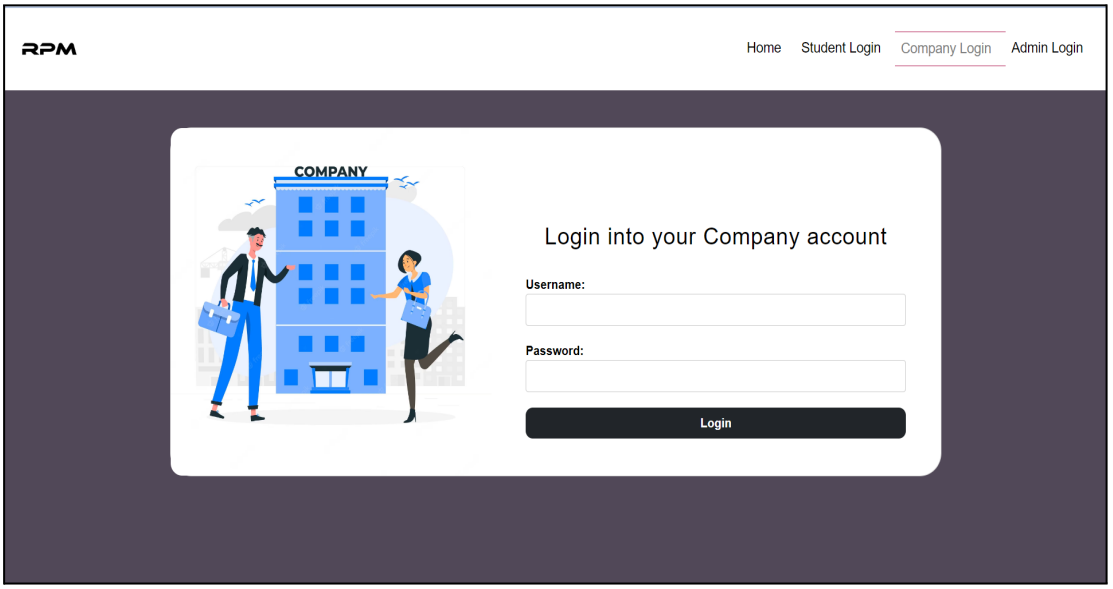


Figure 3.7.1.3 Depicts the Company Login Page

3.7.2. STATISTICS OF PLACEMENT

Figure 3.7.2.1 depicts the statistics of previous year placements.It shows the number of Students placed.

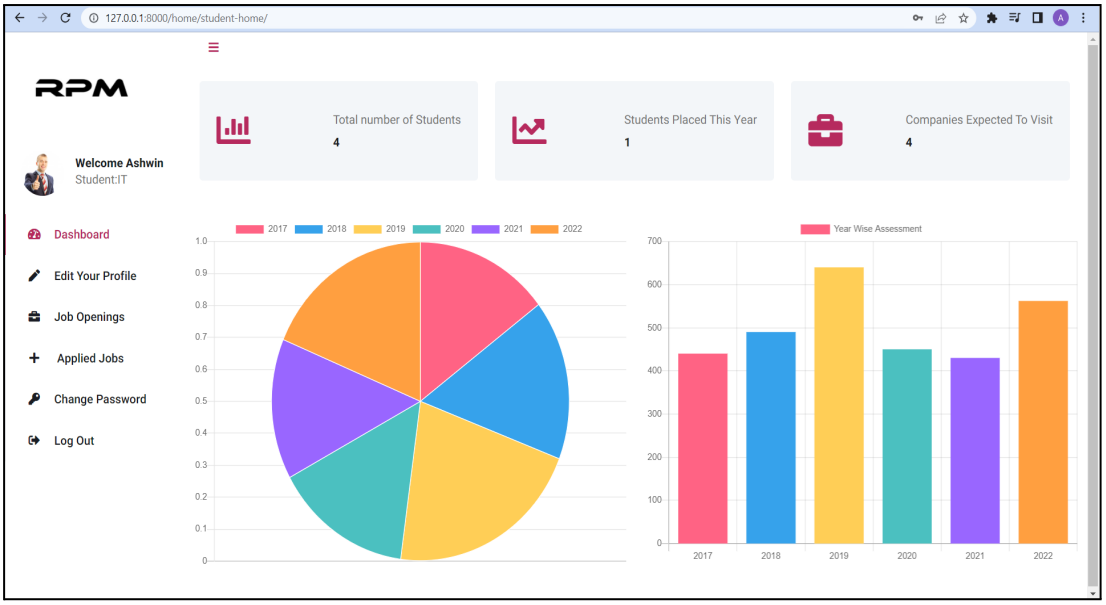


Figure 3.7.2.1 Depicts the statistics of placements

3.7.3. Job lists

Figure 3.7.3.1 depicts the list of jobs that a student can apply for.We can apply for this job by clicking the button.

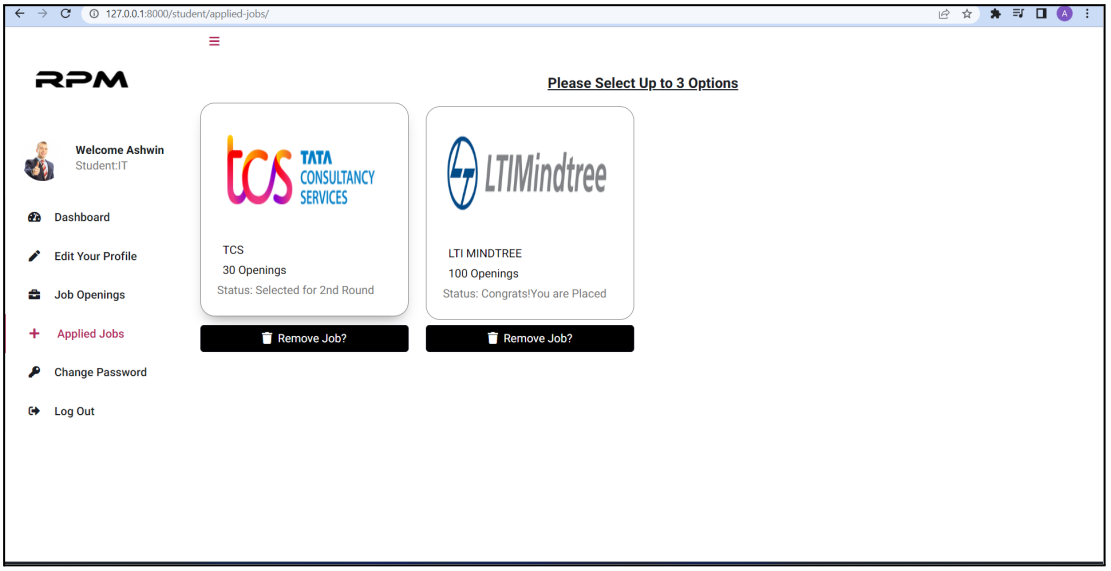


Figure 3.7.3.1 Depicts the job lists

3.7.4. Student details Entry

Figure 3.7.4.1 depicts the entry of details of students.These details include first name, last name and other details of the student.

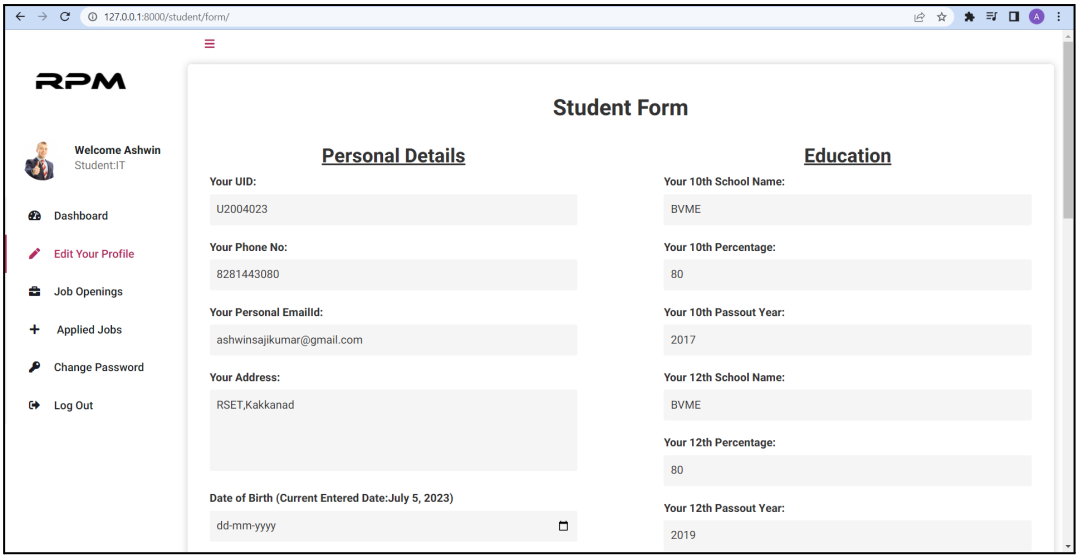


Figure 3.7.4.1 Depicts the entry of student details

3.8 IMPLEMENTATION REQUIREMENTS AND SCHEDULE

3.8.1 HARDWARE REQUIREMENTS

The selection of hardware configuration is a very important task related to software development, particularly inefficient RAM may adversely affect the speed and correspondingly the efficiency of the entire system. The processor should be powerful to handle all the operations. The hard disk should have sufficient capacity to solve the database and the application. The network should be efficient to handle the communication fast.

1. CPU: i3 Processor
2. Memory: 128 MB
3. Cache: 512KB
4. Floppy Disk. :1.44MB
5. HardDisk:4.3GB
6. Display:15” Monitor
7. Keyboard: Standard108keysEnhancedKeyBoard
8. Mouse: MSSerialMouse

3.8.2 SOFTWARE REQUIREMENTS

1. Operating System: WindowsXP, 7, 8 or above
2. Front End Tool: Python
3. Back End Tool: SQLite
4. IDE: Visual Studio Code

CHAPTER 4
RESULTS AND DISCUSSION

The Student Placement Management System is a comprehensive software application designed to streamline and enhance the process of managing student placements in educational institutions. This section presents the results and discussion of the system's implementation and its impact on the placement process. The system successfully automated various manual tasks involved in the placement process, such as student registration, company profile management, job posting, and student-company matching. This automation significantly reduced the administrative workload, enabling staff to focus on more strategic aspects of the placement process. The algorithm implemented for student-company matching proved to be effective in suggesting suitable job opportunities to students based on their qualifications and preferences. This resulted in improved student-company matches, increasing the likelihood of successful placements and enhancing the overall satisfaction of both students and companies.

Students received timely updates on placement activities, interview schedules, and company responses, enhancing their engagement and reducing anxiety during the placement process. The Student Placement Management System significantly improved the overall efficiency of the placement process. By automating time-consuming tasks and providing quick access to information, the system streamlined the workflow for staff members, leading to smoother operations and faster decision-making.With the system's transparent communication and efficient matching algorithm, students reported a more positive experience during the placement process. They felt more informed about available job opportunities and appreciated the personalized job recommendations based on their skills and preferences. The streamlined job posting and application tracking features made it easier for recruiters to engage with potential candidates, leading to increased company interest in campus placements. The system's data analytics capabilities provided valuable insights into placement trends and student performance. This

data-driven approach allowed placement coordinators to make informed decisions and tailor placement strategies to better meet students' and companies' needs.

Based on user feedback and system performance, future enhancements can be made to further improve the system. These may include incorporating AI-based features for better student-company matching, integrating interview scheduling tools, and enhancing the reporting and analytics functionalities.

CHAPTER 5
CONCLUSION

The Django-based Student Placement Management System successfully streamlined the placement process. With rapid development and scalability, the user-friendly interface facilitated seamless navigation for students, companies, and staff. Django's strong security features ensure data confidentiality and system reliability.

The system's automation reduced administrative overhead and improved efficiency. The algorithm-based student-company matching enhanced student experience, while the company interface attracted more recruiters. Future directions include continuous improvement, AI integration, and mobile accessibility.

The centralized database management and data analytics capabilities empowered staff with valuable insights for data-driven decision-making. With Django's continuous updates and best practices, the system remains up-to-date and robust. The secure and reliable nature of Django provided a safe environment for sensitive student and company information.

Overall, the system has positively impacted the placement process, fostering successful student careers and productive company partnerships.

CHAPTER 6
REFERENCES

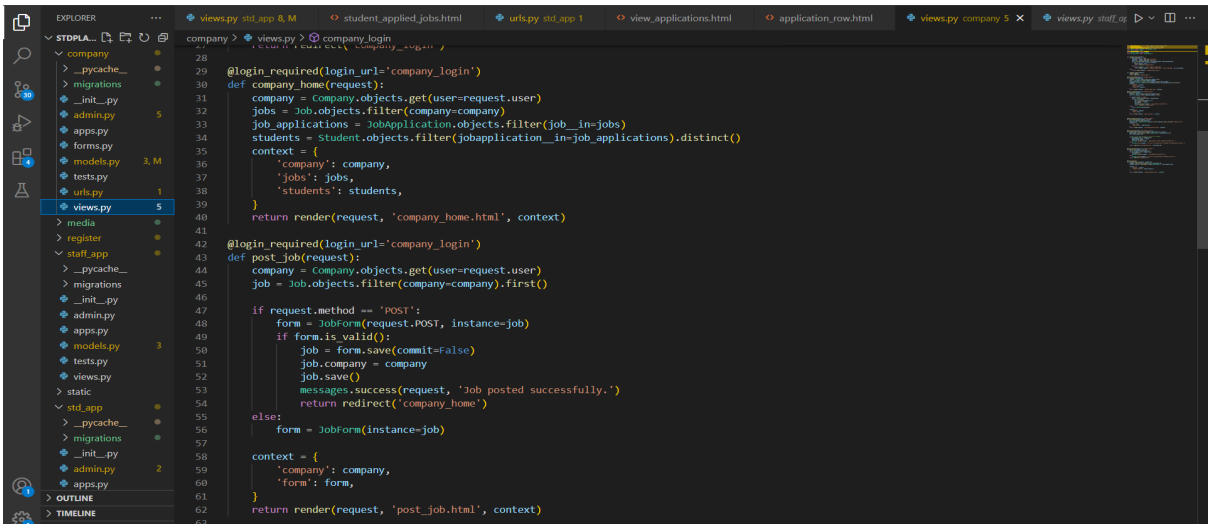
- Python-https://www.python.org/doc/
- Chartjs -https://github.com/chartjs/Chart.js helped in creating the statistics chart for placements
- Bootstrap-https://getbootstrap.com/docs/5.1/getting-started/introduction/ helped in enhancing the user interface
- W3schools-https://www.w3schools.com/django/ used as reference for django
- Open AI -https://chat.openai.com/auth/login
- DBdiagram - https://dbdiagram.io/d to create Database design
- College Training and Placement System Project Report: https://www.scribd.com/document/440794070/College-Training-And-Placement-System-Project-Report-docx
- Web Based Placement Management System: https://ijcsit.com/docs/Volume%207/vol7issue2/ijcsit2016070268.pdf
- Placement Management System for Campus Recruitment: https://ijisrt.com/assets/upload/files/IJISRT20MAY826.pdf
- Placement management system: https://www.slideshare.net/mehul53/placement-management-system
- A Research on Placement Management System: https://www.ijraset.com/research-paper/placement-management-system

APPENDIX A
PSEUDO CODE

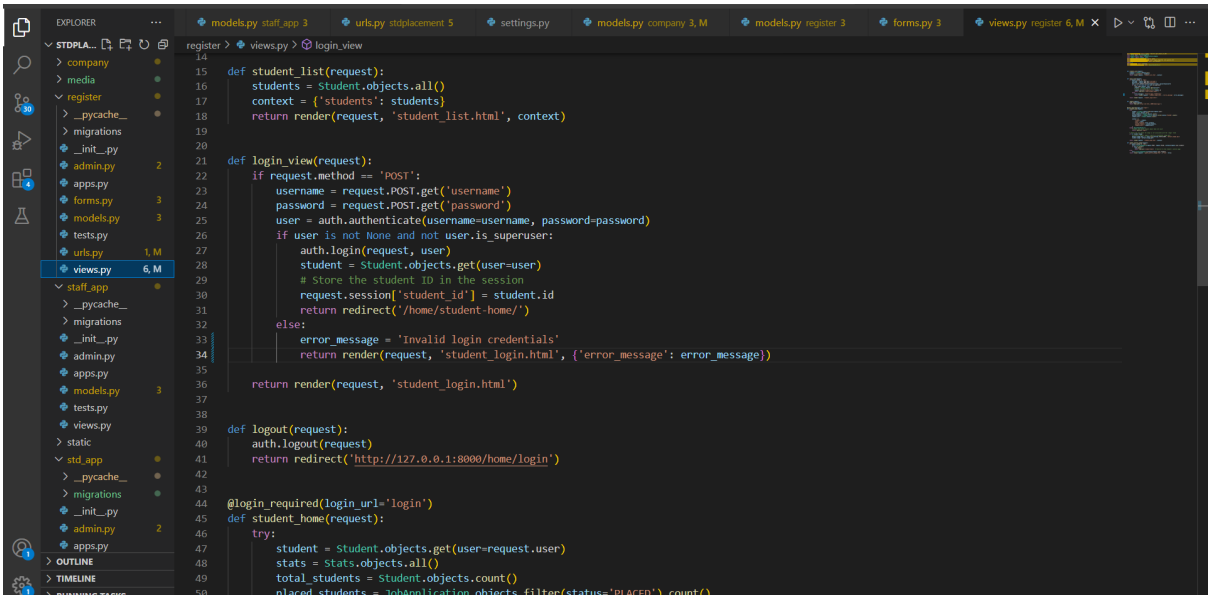
STUDENT APPLYING FOR A JOB

```
    return render(request, 'student_view_jobs.html', context)

@login_required(login_url='login')
def try_job(request, job_id):
    job = Job.objects.get(id=job_id)
    student = Student.objects.get(user=request.user)
    student_details = StudentDetails.objects.filter(user=request.user).first()

    if student_details is None:
        # Redirect to a page indicating that the form hasn't been filled
        return render(request, 'form_not_filled.html')

    if (
        student_details.Current_CGPA >= job.cgpa and
        student_details.Supply_Count <= job.Backpaper and
        student_details.TenthPercentage >= job.tenthper and
        student_details.TwelthPercentage >= job.twelthper and
        student_details.Internship >= job.internship
    ):
        # Check the number of existing job applications
        existing_applications = JobApplication.objects.filter(student=student).count()
        if existing_applications >= 3:
            # Display a prompt and deny the request
            return render(request, 'exceeded_applications.html')
        else:
            # Check if the student has already applied for the job
            existing_application = JobApplication.objects.filter(job=job, student=student).first()
            if existing_application:
                # Display a message that the student has already applied for the job
                messages.warning(request, 'You have already applied for this job.')
            else:
                # Create a new job application
                application = JobApplication(job=job, student=student, company=job.company)  # Set the company field
                application.save()
                # Display a success message
                messages.success(request, 'Job application submitted successfully.')
```

2.COMPANY LOGINS AND POSTS JOB

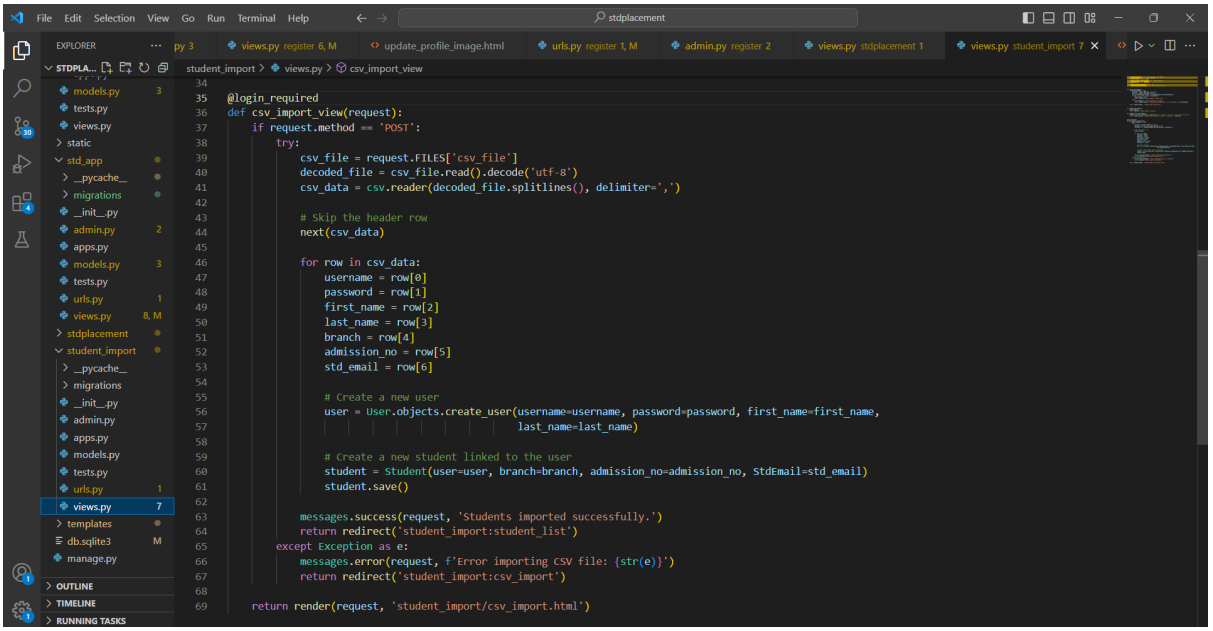


STUDENT LOGIN



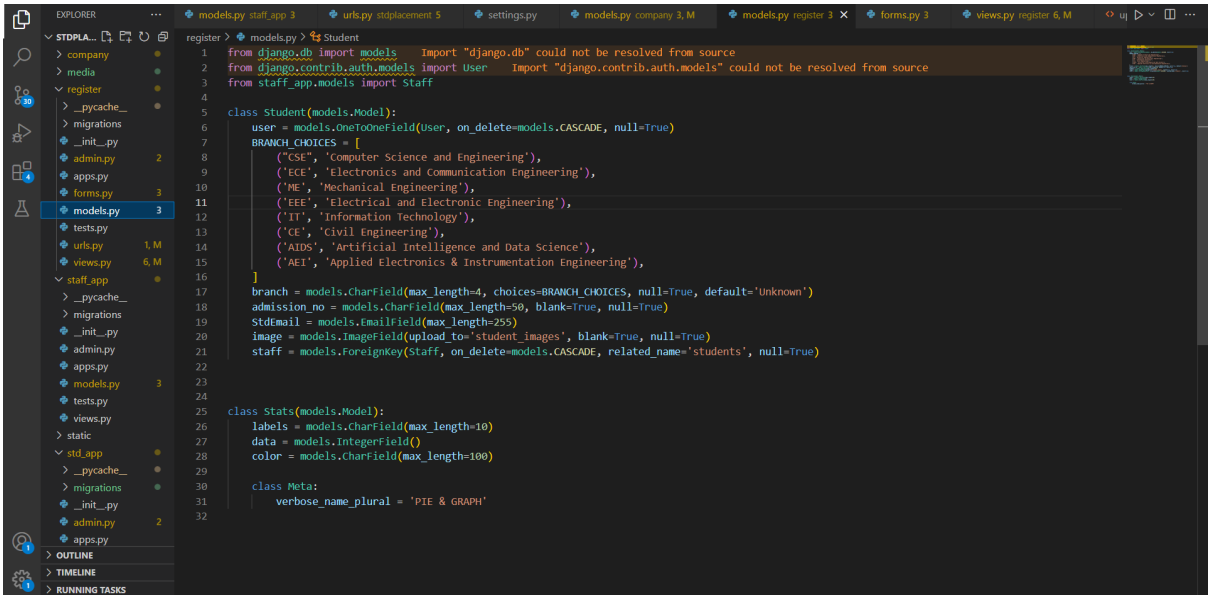
```python
def student_list(request):
    students = Student.objects.all()
    context = {'students': students}
    return render(request, 'student_list.html', context)


def login_view(request):
    if request.method == 'POST':
        username = request.POST.get('username')
        password = request.POST.get('password')
        user = auth.authenticate(username=username, password=password)
        if user is not None and not user.is_superuser:
            auth.login(request, user)
            student = Student.objects.get(user=user)
            # Store the student ID in the session
            request.session['student_id'] = student.id
            return redirect('/home/student-home/')
        else:
            error_message = 'Invalid login credentials'
            return render(request, 'student_login.html', {'error_message': error_message})

    return render(request, 'student_login.html')


def logout(request):
    auth.logout(request)
    return redirect('http://127.0.0.1:8000/home/login')


@login_required(login_url='login')
def student_home(request):
    try:
        student = Student.objects.get(user=request.user)
        stats = Stats.objects.all()
        total_students = Student.objects.count()
        placed_students = JobApplication.objects.filter(status='PLACED').count()
```

ADMIN CREATES ACCOUNT BY IMPORTING CSV FILE



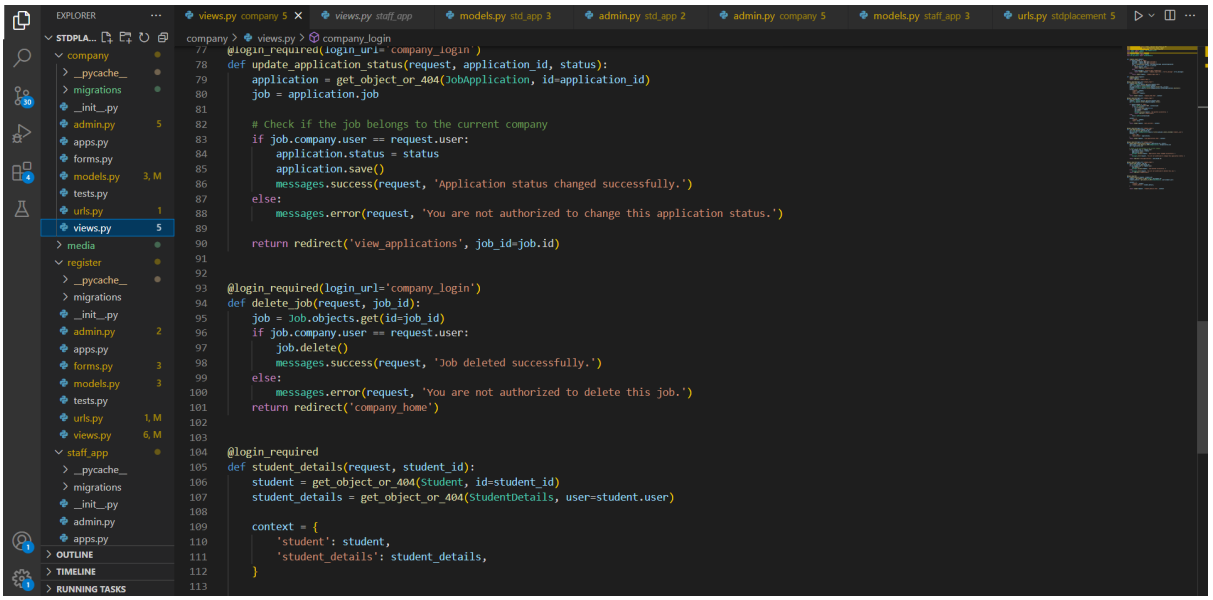
```python
@login_required
def csv_import_view(request):
    if request.method == 'POST':
        try:
            csv_file = request.FILES['csv_file']
            decoded_file = csv_file.read().decode('utf-8')
            csv_data = csv.reader(decoded_file.splitlines(), delimiter=',')

            # Skip the header row
            next(csv_data)

            for row in csv_data:
                username = row[0]
                password = row[1]
                first_name = row[2]
                last_name = row[3]
                branch = row[4]
                admission_no = row[5]
                std_email = row[6]

                # Create a new user
                user = User.objects.create_user(username=username, password=password, first_name=first_name,
                                                last_name=last_name)

                # Create a new student linked to the user
                student = Student(user=user, branch=branch, admission_no=admission_no, StdEmail=std_email)
                student.save()

            messages.success(request, 'Students imported successfully.')
            return redirect('student_import:student_list')
        except Exception as e:
            messages.error(request, f'Error importing CSV file: {str(e)}')
            return redirect('student_import:csv_import')

    return render(request, 'student_import/csv_import.html')
```

STUDENT MODEL CREATION -



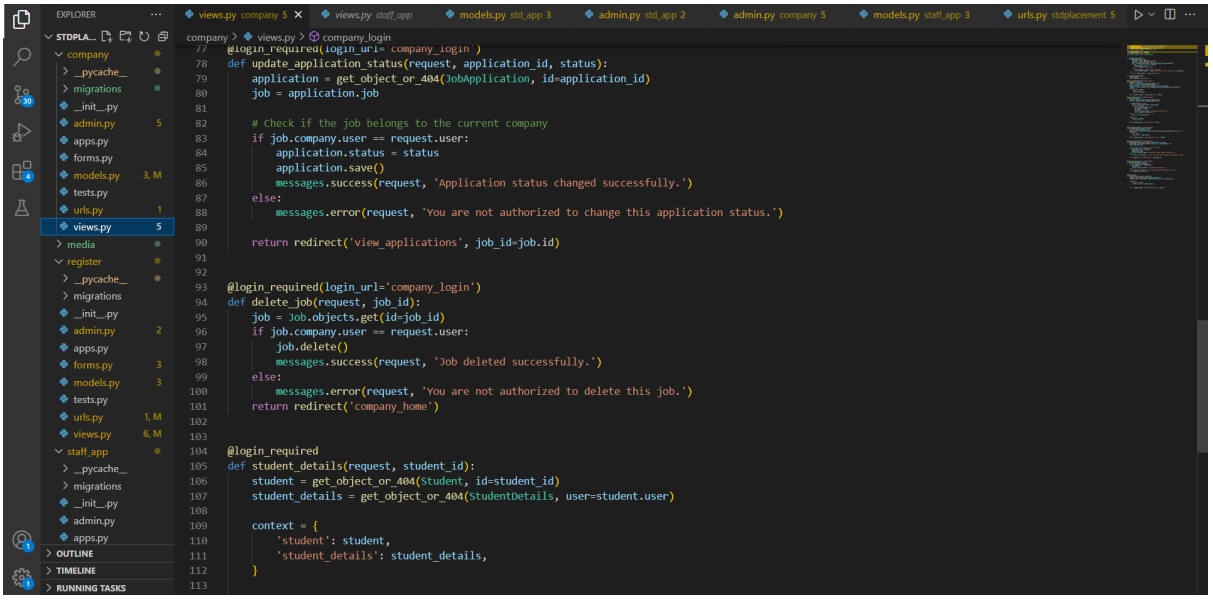
```python
from django.db import models    Import "django.db" could not be resolved from source
from django.contrib.auth.models import User    Import "django.contrib.auth.models" could not be resolved from source
from staff_app.models import Staff

class Student(models.Model):
    user = models.OneToOneField(User, on_delete=models.CASCADE, null=True)
    BRANCH_CHOICES = [
        ('CSE', 'Computer Science and Engineering'),
        ('ECE', 'Electronics and Communication Engineering'),
        ('ME', 'Mechanical Engineering'),
        ('EEE', 'Electrical and Electronic Engineering'),
        ('IT', 'Information Technology'),
        ('CE', 'Civil Engineering'),
        ('AIDS', 'Artificial Intelligence and Data Science'),
        ('AEI', 'Applied Electronics & Instrumentation Engineering'),
    ]
    branch = models.CharField(max_length=4, choices=BRANCH_CHOICES, null=True, default='Unknown')
    admission_no = models.CharField(max_length=50, blank=True, null=True)
    StdEmail = models.EmailField(max_length=255)
    image = models.ImageField(upload_to='student_images', blank=True, null=True)
    staff = models.ForeignKey(Staff, on_delete=models.CASCADE, related_name='students', null=True)


class Stats(models.Model):
    labels = models.CharField(max_length=10)
    data = models.IntegerField()
    color = models.CharField(max_length=100)

    class Meta:
        verbose_name_plural = 'PIE & GRAPH'
```

ROUTING -routing is used to enable the creation of dynamic and interactive websites that can help navigate between different pages or sections seamlessly



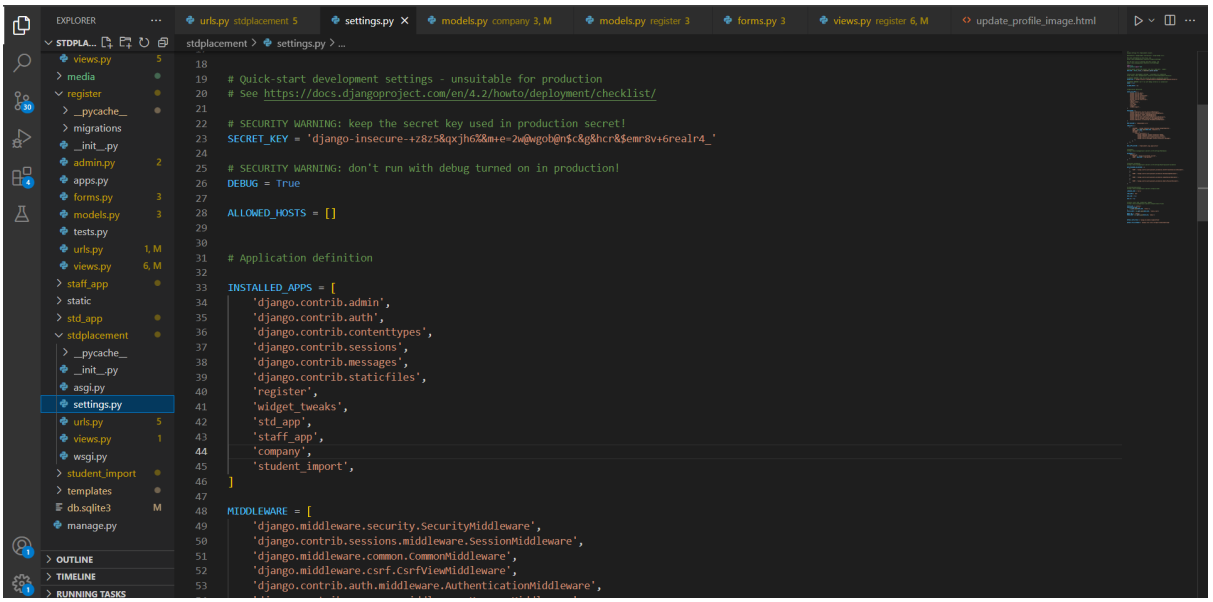
```python
@login_required(login_url='company_login')
def update_application_status(request, application_id, status):
    application = get_object_or_404(JobApplication, id=application_id)
    job = application.job

    # Check if the job belongs to the current company
    if job.company.user == request.user:
        application.status = status
        application.save()
        messages.success(request, 'Application status changed successfully.')
    else:
        messages.error(request, 'You are not authorized to change this application status.')

    return redirect('view_applications', job_id=job.id)


@login_required(login_url='company_login')
def delete_job(request, job_id):
    job = Job.objects.get(id=job_id)
    if job.company.user == request.user:
        job.delete()
        messages.success(request, 'Job deleted successfully.')
    else:
        messages.error(request, 'You are not authorized to delete this job.')
    return redirect('company_home')


@login_required
def student_details(request, student_id):
    student = get_object_or_404(Student, id=student_id)
    student_details = get_object_or_404(StudentDetails, user=student.user)

    context = {
        'student': student,
        'student_details': student_details,
    }
```

DELETE OR HIDE JOB WHEN TIME PERIOD OF JOB POSTING IS OVER



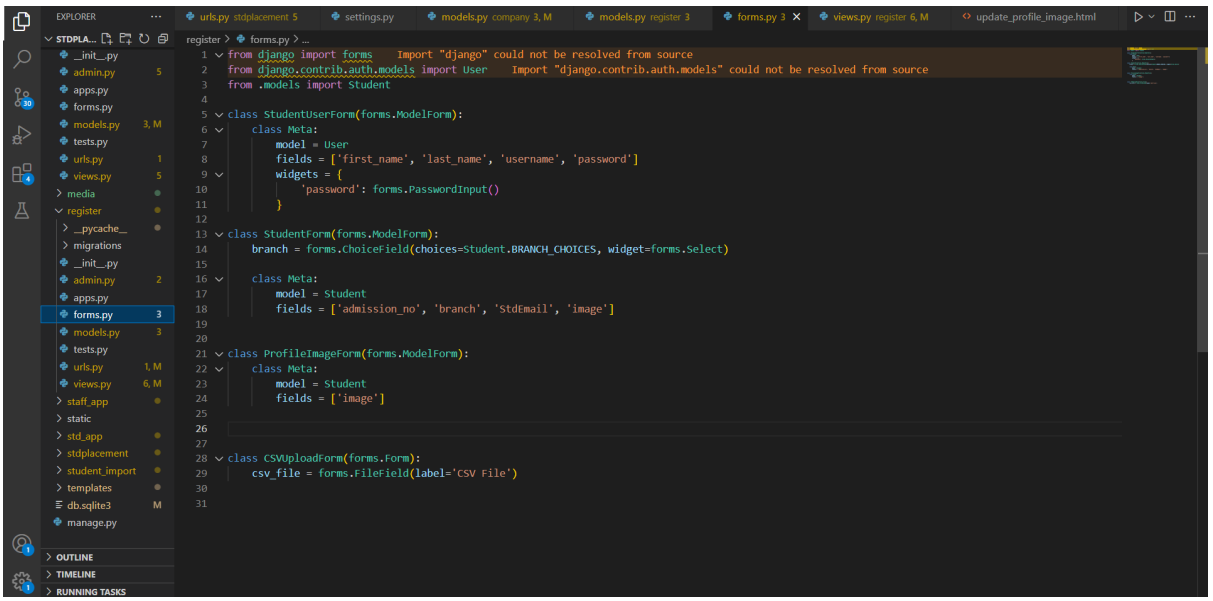
```python
@login_required(login_url='company_login')
def update_application_status(request, application_id, status):
    application = get_object_or_404(JobApplication, id=application_id)
    job = application.job

    # Check if the job belongs to the current company
    if job.company.user == request.user:
        application.status = status
        application.save()
        messages.success(request, 'Application status changed successfully.')
    else:
        messages.error(request, 'You are not authorized to change this application status.')

    return redirect('view_applications', job_id=job.id)


@login_required(login_url='company_login')
def delete_job(request, job_id):
    job = Job.objects.get(id=job_id)
    if job.company.user == request.user:
        job.delete()
        messages.success(request, 'Job deleted successfully.')
    else:
        messages.error(request, 'You are not authorized to delete this job.')
    return redirect('company_home')


@login_required
def student_details(request, student_id):
    student = get_object_or_404(Student, id=student_id)
    student_details = get_object_or_404(StudentDetails, user=student.user)

    context = {
        'student': student,
        'student_details': student_details,
    }
```

TEMPLATES



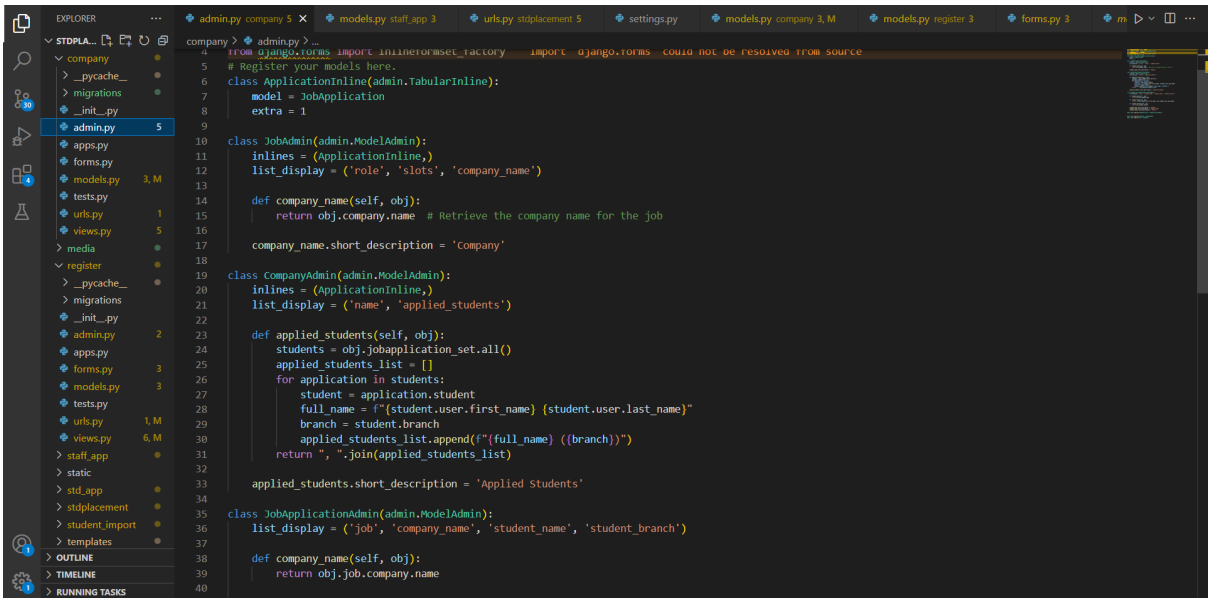
```python
# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/4.2/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'django-insecure-+zBz5&qxjh6%&m+e=2w@wgob@n$c&g&hcr&$emr8v+6realr4_'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = []


# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'register',
    'widget_tweaks',
    'std_app',
    'staff_app',
    'company',
    'student_import',
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
```

THIS IS USED AS TEMPLATE FOR FORMS AND WHEN REQUIRED IT CREATES A NEW FORM



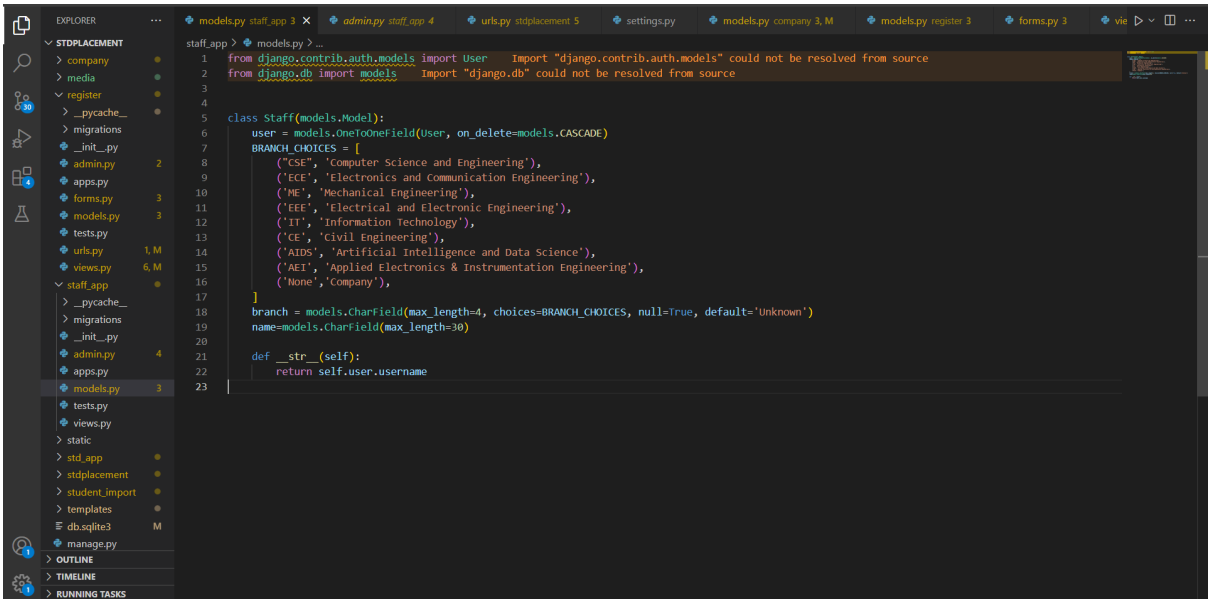
```python
from django import forms    Import "django" could not be resolved from source
from django.contrib.auth.models import User    Import "django.contrib.auth.models" could not be resolved from source
from .models import Student

class StudentUserForm(forms.ModelForm):
    class Meta:
        model = User
        fields = ['first_name', 'last_name', 'username', 'password']
        widgets = {
            'password': forms.PasswordInput()
        }

class StudentForm(forms.ModelForm):
    branch = forms.ChoiceField(choices=Student.BRANCH_CHOICES, widget=forms.Select)

    class Meta:
        model = Student
        fields = ['admission_no', 'branch', 'StdEmail', 'image']


class ProfileImageForm(forms.ModelForm):
    class Meta:
        model = Student
        fields = ['image']


class CSVUploadForm(forms.Form):
    csv_file = forms.FileField(label='CSV File')
```

ADMIN CAN VIEW DETAILS OF COMPANY AND DETAILS OF STUDENTS WHO HAVE APPLIED FOR JOBS



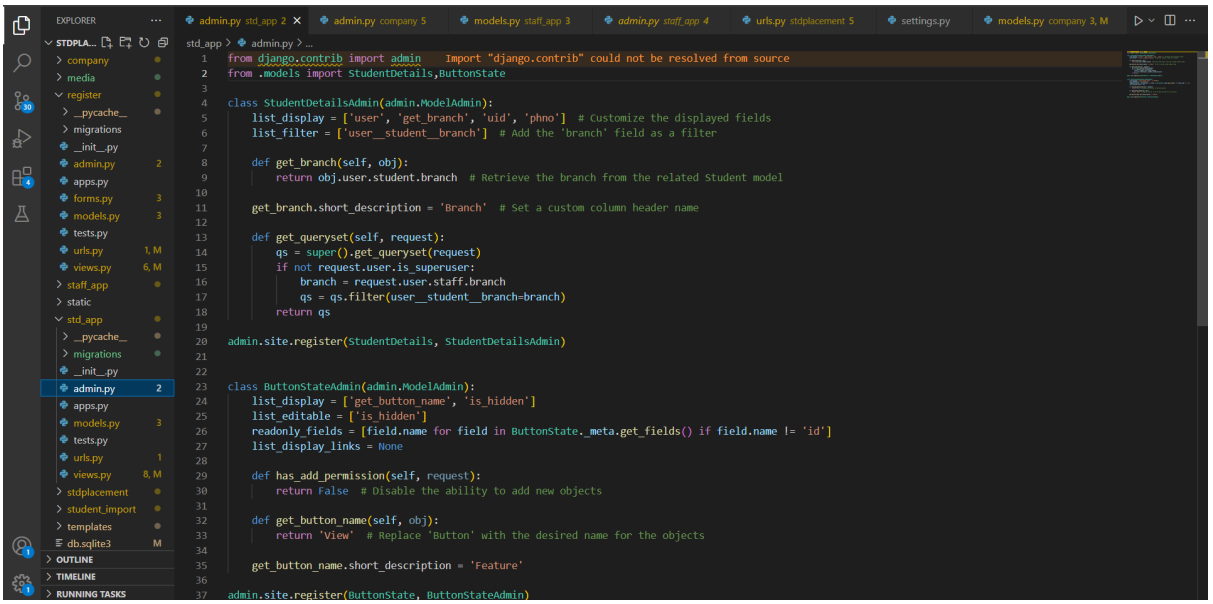
```python
from django.forms import inlineformset_factory    Import "django.forms" could not be resolved from source
# Register your models here.
class ApplicationInline(admin.TabularInline):
    model = JobApplication
    extra = 1

class JobAdmin(admin.ModelAdmin):
    inlines = (ApplicationInline,)
    list_display = ('role', 'slots', 'company_name')

    def company_name(self, obj):
        return obj.company.name  # Retrieve the company name for the job

    company_name.short_description = 'Company'

class CompanyAdmin(admin.ModelAdmin):
    inlines = (ApplicationInline,)
    list_display = ('name', 'applied_students')

    def applied_students(self, obj):
        students = obj.jobapplication_set.all()
        applied_students_list = []
        for application in students:
            student = application.student
            full_name = f"{student.user.first_name} {student.user.last_name}"
            branch = student.branch
            applied_students_list.append(f"{full_name} ({branch})")
        return ", ".join(applied_students_list)

    applied_students.short_description = 'Applied Students'

class JobApplicationAdmin(admin.ModelAdmin):
    list_display = ('job', 'company_name', 'student_name', 'student_branch')

    def company_name(self, obj):
        return obj.job.company.name
```

ACCOUNT CREATION OF STAFF USING MODELS FILE



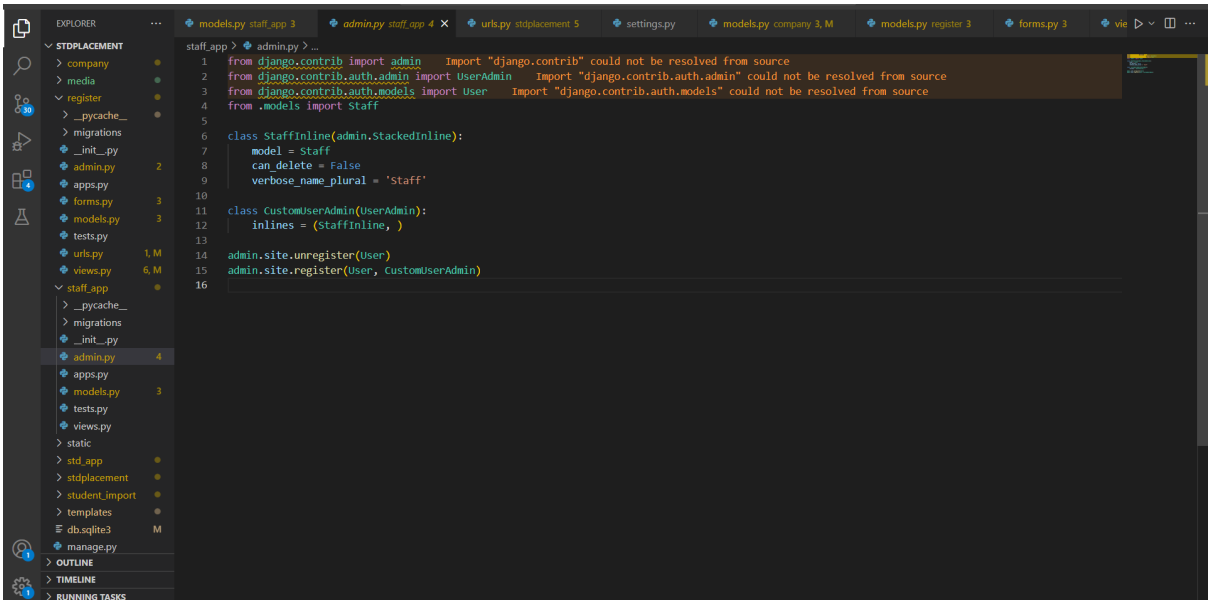
```python
from django.contrib.auth.models import User    Import "django.contrib.auth.models" could not be resolved from source
from django.db import models    Import "django.db" could not be resolved from source


class Staff(models.Model):
    user = models.OneToOneField(User, on_delete=models.CASCADE)
    BRANCH_CHOICES = [
        ("CSE", 'Computer Science and Engineering'),
        ('ECE', 'Electronics and Communication Engineering'),
        ('ME', 'Mechanical Engineering'),
        ('EEE', 'Electrical and Electronic Engineering'),
        ('IT', 'Information Technology'),
        ('CE', 'Civil Engineering'),
        ('AIDS', 'Artificial Intelligence and Data Science'),
        ('AEI', 'Applied Electronics & Instrumentation Engineering'),
        ('None','Company'),
    ]
    branch = models.CharField(max_length=4, choices=BRANCH_CHOICES, null=True, default='Unknown')
    name=models.CharField(max_length=30)

    def __str__(self):
        return self.user.username

```

CODE WHICH REPRESENTS HOW ADMIN CAN VIEW STUDENT DETAILS



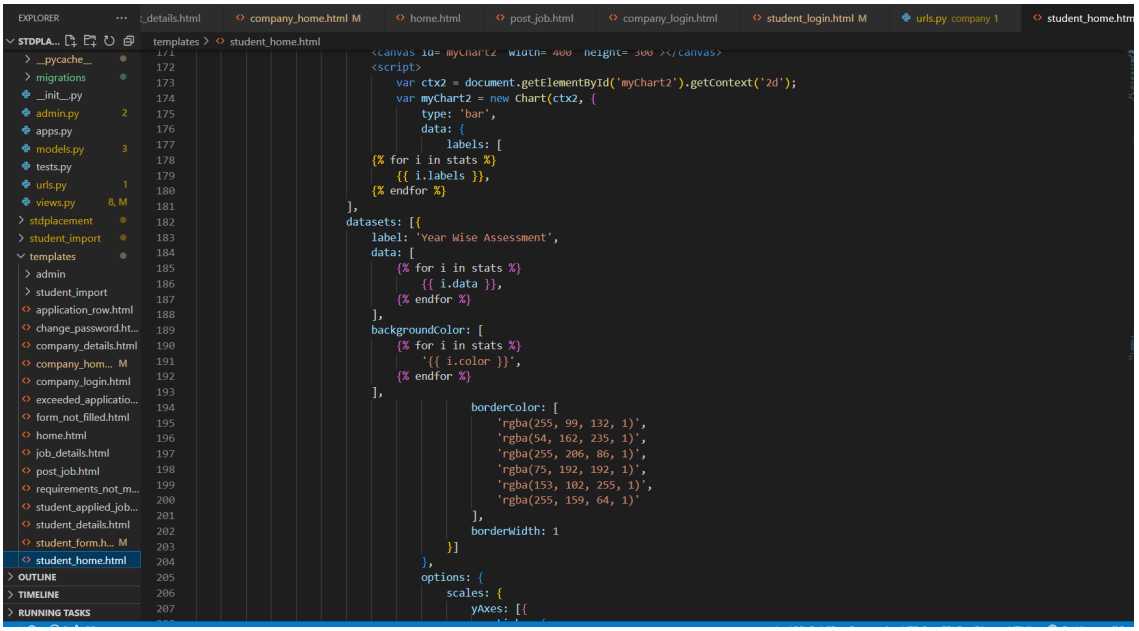
```python
from django.contrib import admin    Import "django.contrib" could not be resolved from source
from .models import StudentDetails,ButtonState

class StudentDetailsAdmin(admin.ModelAdmin):
    list_display = ['user', 'get_branch', 'uid', 'phno']  # Customize the displayed fields
    list_filter = ['user__student__branch']  # Add the 'branch' field as a filter

    def get_branch(self, obj):
        return obj.user.student.branch  # Retrieve the branch from the related Student model

    get_branch.short_description = 'Branch'  # Set a custom column header name

    def get_queryset(self, request):
        qs = super().get_queryset(request)
        if not request.user.is_superuser:
            branch = request.user.staff.branch
            qs = qs.filter(user__student__branch=branch)
        return qs

admin.site.register(StudentDetails, StudentDetailsAdmin)


class ButtonStateAdmin(admin.ModelAdmin):
    list_display = ['get_button_name', 'is_hidden']
    list_editable = ['is_hidden']
    readonly_fields = [field.name for field in ButtonState._meta.get_fields() if field.name != 'id']
    list_display_links = None

    def has_add_permission(self, request):
        return False  # Disable the ability to add new objects

    def get_button_name(self, obj):
        return 'View'  # Replace 'Button' with the desired name for the objects

    get_button_name.short_description = 'Feature'

admin.site.register(ButtonState, ButtonStateAdmin)
```

CODE WHICH REPRESENTS HOW ADMIN CAN VIEW STAFF DETAILS



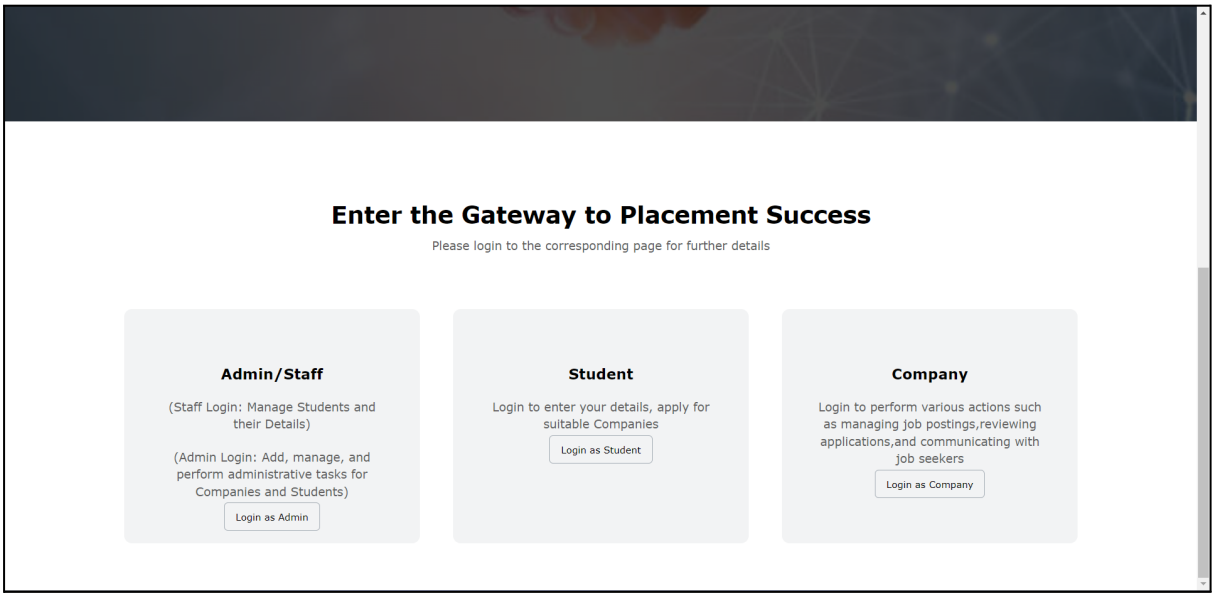
```
from django.contrib import admin    Import "django.contrib" could not be resolved from source
from django.contrib.auth.admin import UserAdmin    Import "django.contrib.auth.admin" could not be resolved from source
from django.contrib.auth.models import User    Import "django.contrib.auth.models" could not be resolved from source
from .models import Staff

class StaffInline(admin.StackedInline):
    model = Staff
    can_delete = False
    verbose_name_plural = 'Staff'

class CustomUserAdmin(UserAdmin):
    inlines = (StaffInline, )

admin.site.unregister(User)
admin.site.register(User, CustomUserAdmin)
```

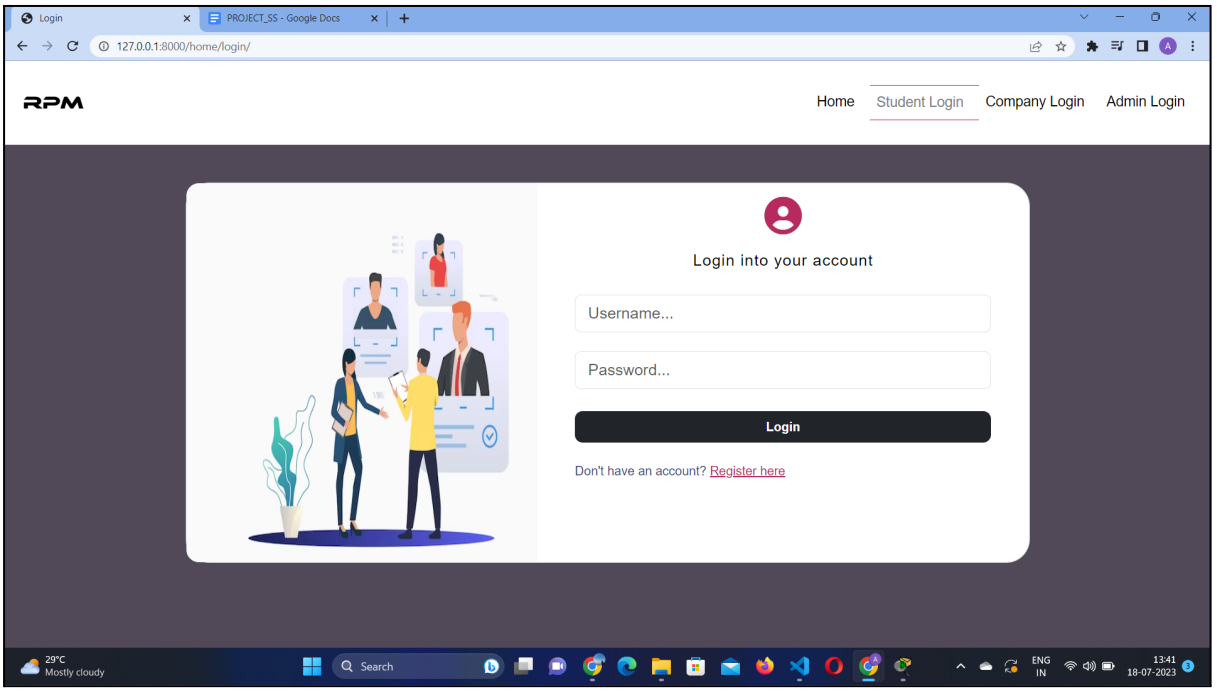
USER STUDENT LOGINS THE ACCOUNT AND ENTERS PAGE WHICH CONTAIN PREVIOUS YEAR STATISTICS.THIS STATISTICS IS UPDATED THROUGH THE ADMIN SIDE.

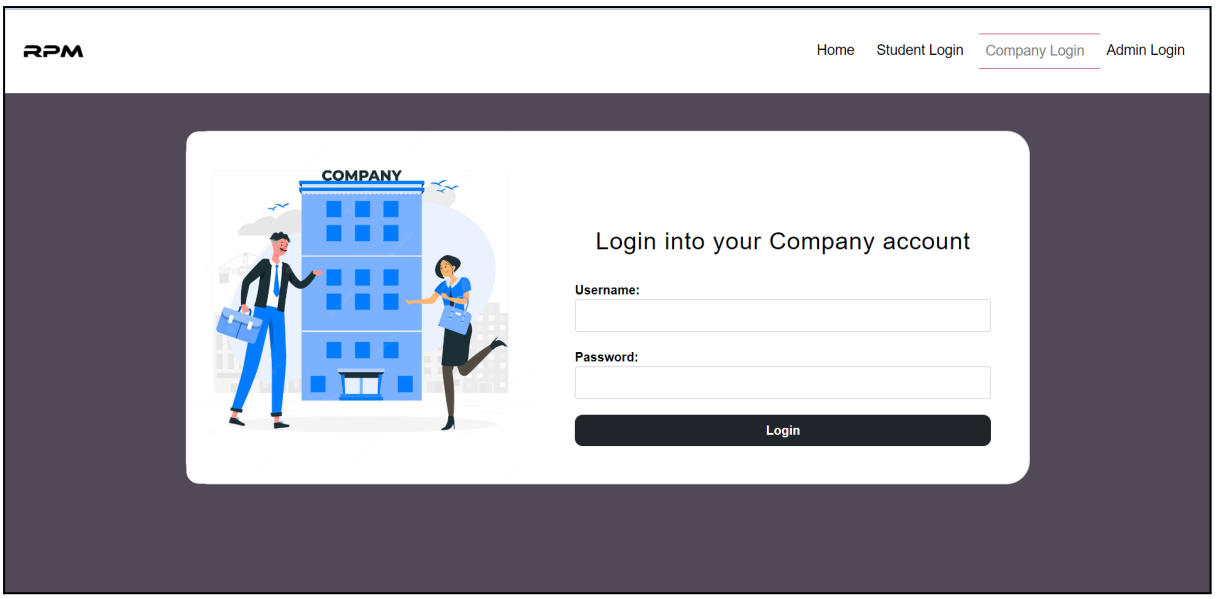


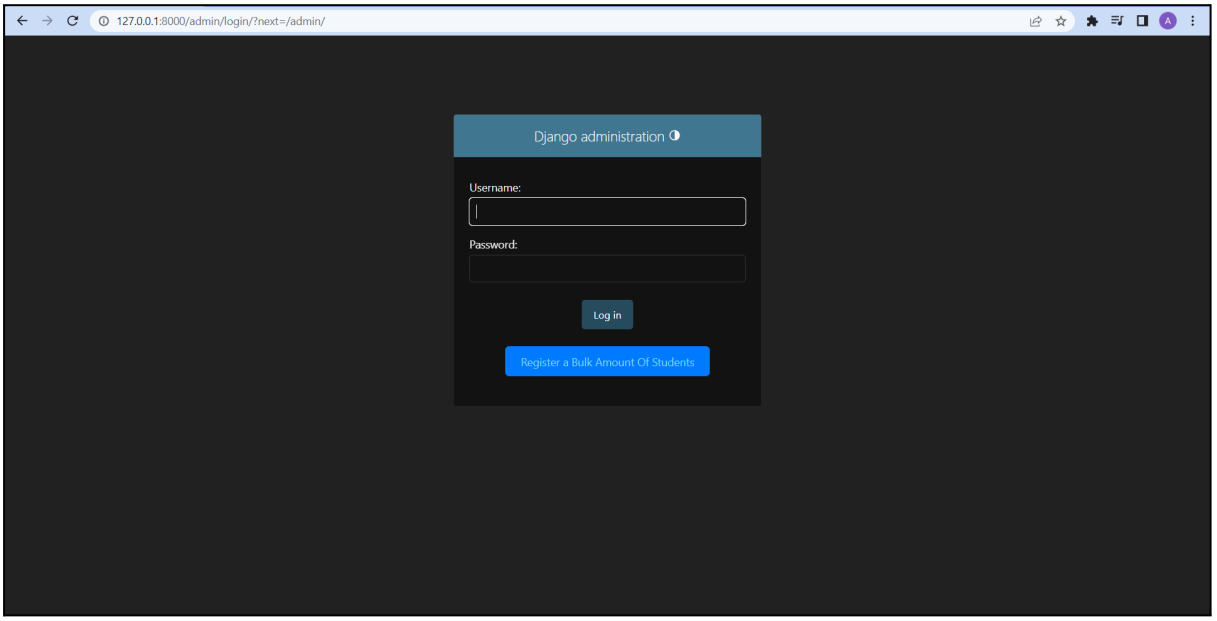
```
<canvas id="myChart2" width="400" height="300"></canvas>
<script>
    var ctx2 = document.getElementById('myChart2').getContext('2d');
    var myChart2 = new Chart(ctx2, {
        type: 'bar',
        data: {
            labels: [
{% for i in stats %}
    {{ i.labels }},
{% endfor %}
],
datasets: [{
    label: 'Year Wise Assessment',
    data: [
        {% for i in stats %}
            {{ i.data }},
        {% endfor %}
    ],
    backgroundColor: [
        {% for i in stats %}
            '{{ i.color }}',
        {% endfor %}
    ],
                borderColor: [
                    'rgba(255, 99, 132, 1)',
                    'rgba(54, 162, 235, 1)',
                    'rgba(255, 206, 86, 1)',
                    'rgba(75, 192, 192, 1)',
                    'rgba(153, 102, 255, 1)',
                    'rgba(255, 159, 64, 1)'
                ],
                borderWidth: 1
            }]
        },
        options: {
            scales: {
                yAxes: [{
```

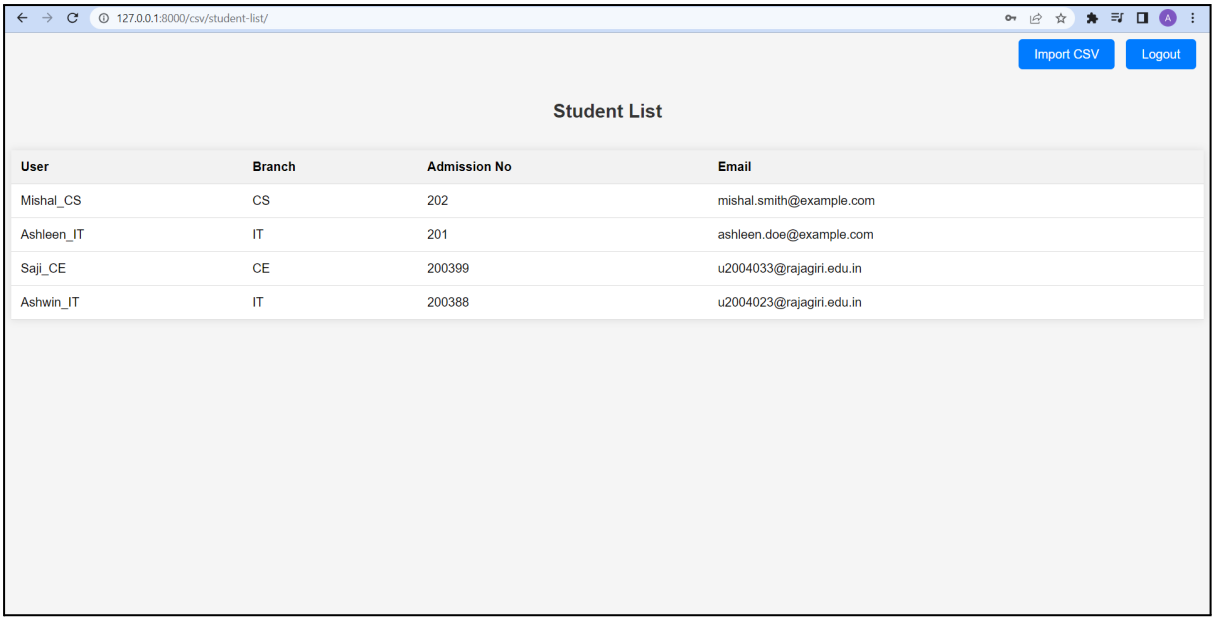
APPENDIX B
SAMPLE SCREENSHOTS

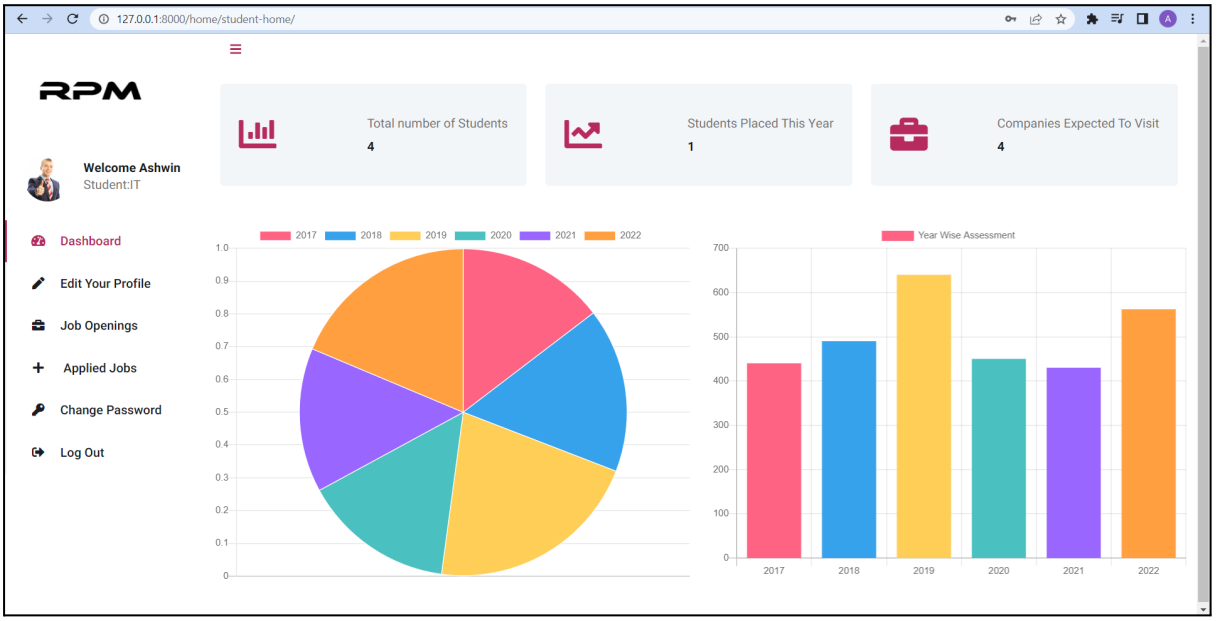


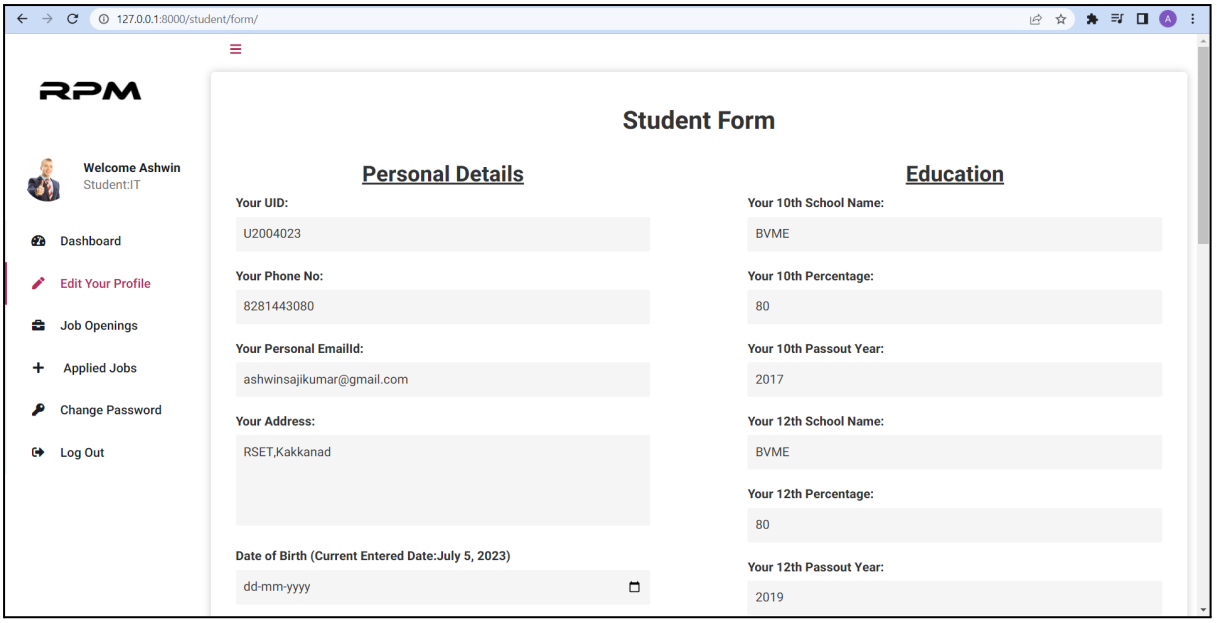



RPM
Home
Student Login
Company Login
Admin Login
COMPANY
Login into your Company account
Username:
Password:
Login

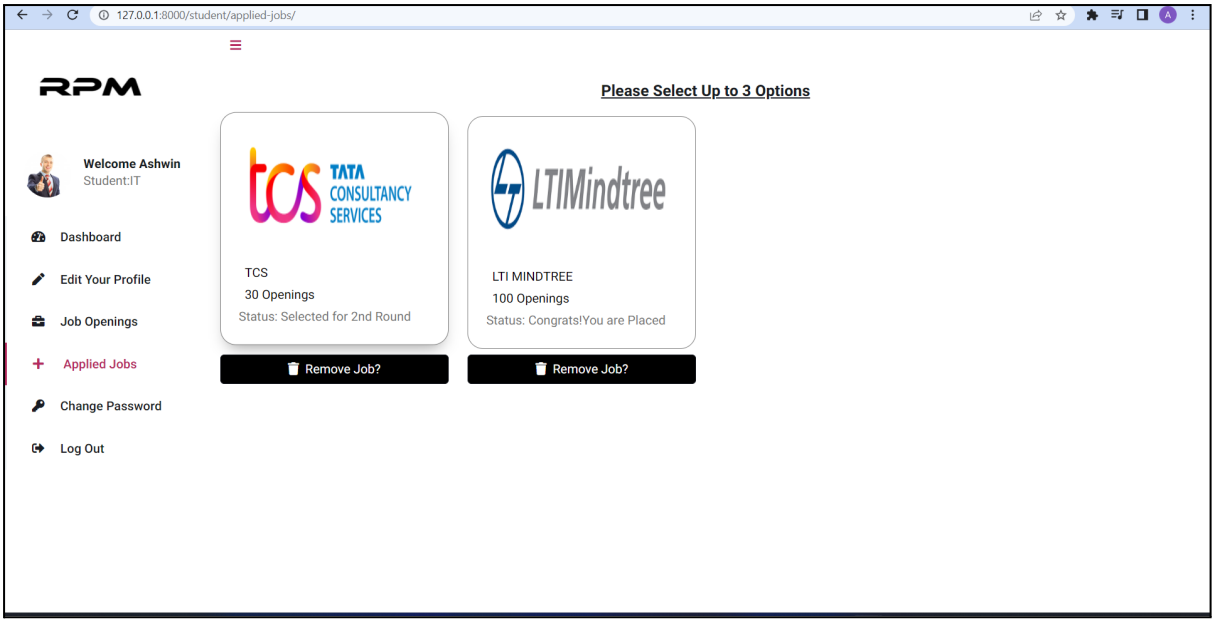

127.0.0.1:8000/admin/login/?next=/admin/
Django administration
Username:
Password:
Log in
Register a Bulk Amount Of Students

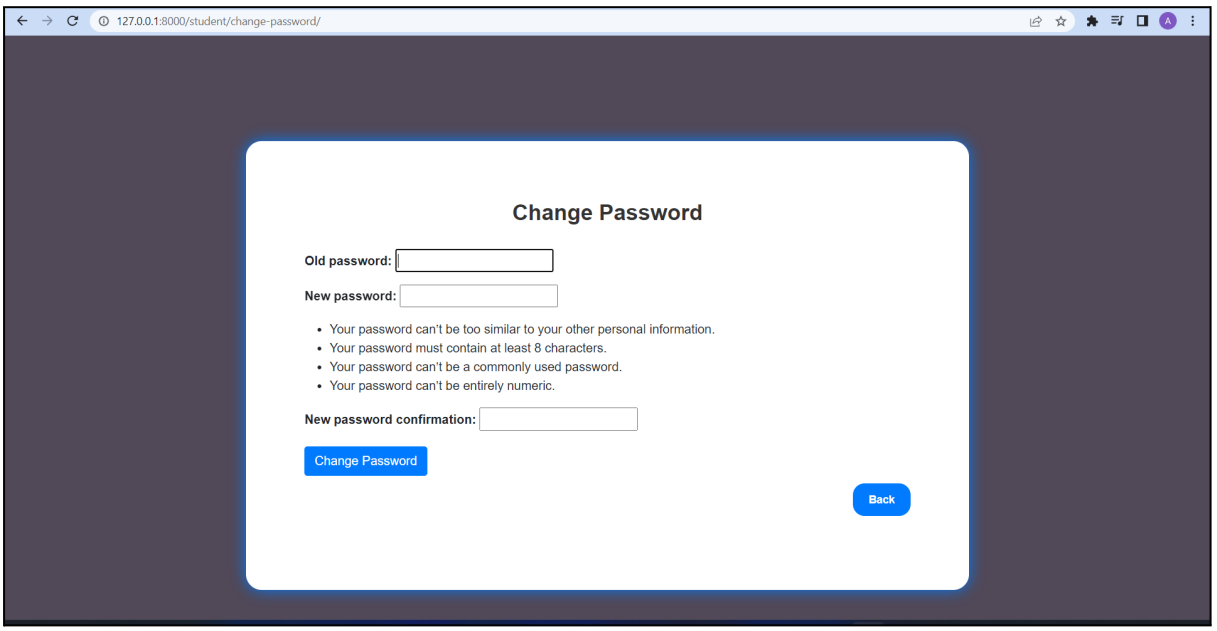

127.0.0.1:8000/csv/student-list/
Import CSV
Logout
Student List
User
Branch
Admission No
Email
Mishal_CS
CS
202
mishal.smith@example.com
Ashleen_IT
IT
201
ashleen.doe@example.com
Saji_CE
CE
200399
u2004033@rajagiri.edu.in
Ashwin_IT
IT
200388
u2004023@rajagiri.edu.in

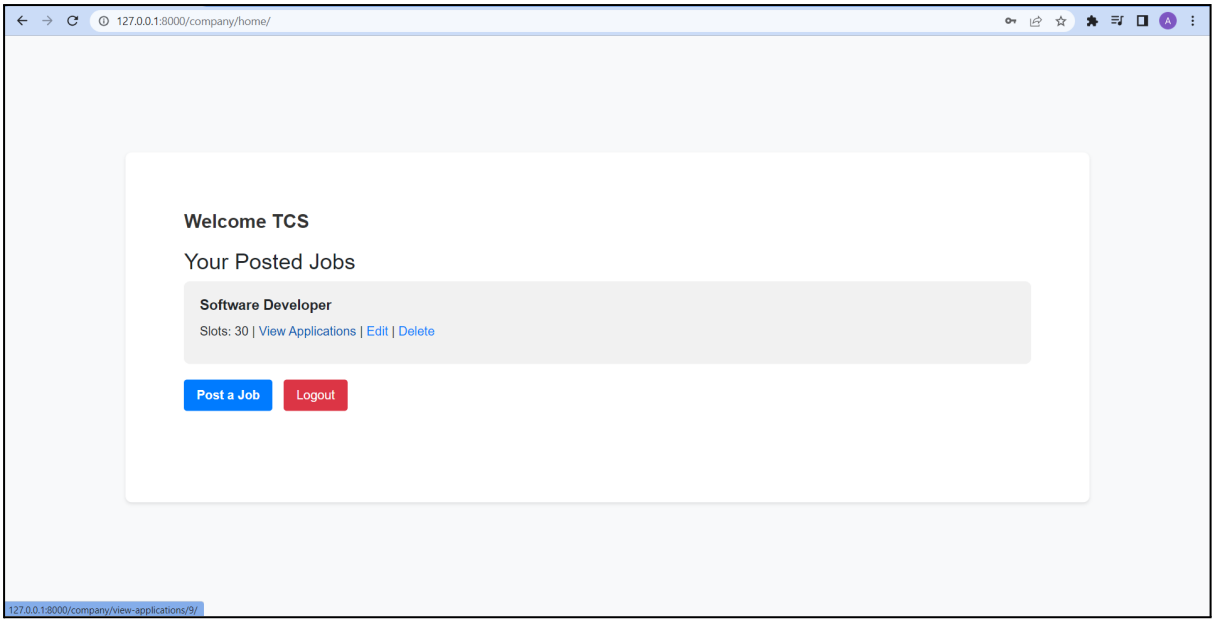

127.0.0.1:8000/home/student-home/
RPM
Welcome Ashwin
Student:IT
Dashboard
Edit Your Profile
Job Openings
Applied Jobs
Change Password
Log Out
Total number of Students
4
Students Placed This Year
1
Companies Expected To Visit
4
2017
2018
2019
2020
2021
2022
Year Wise Assessment

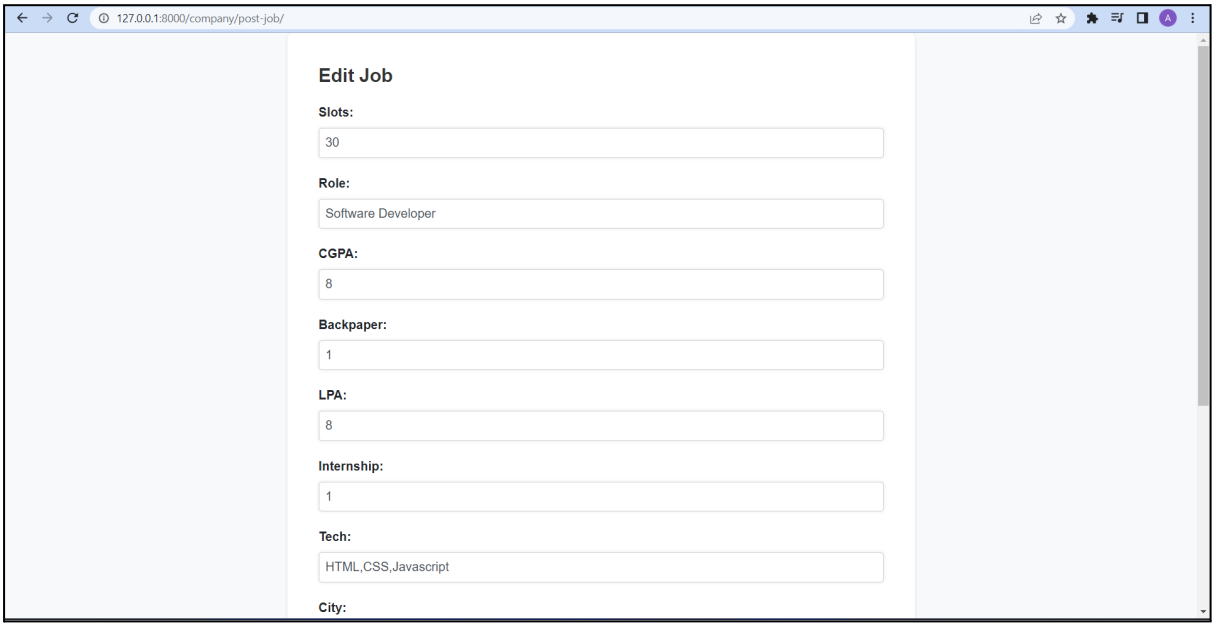

127.0.0.1:8000/student/form/
RPM
Welcome Ashwin
Student:IT
Dashboard
Edit Your Profile
Job Openings
Applied Jobs
Change Password
Log Out
Student Form
Personal Details
Your UID:
U2004023
Your Phone No:
8281443080
Your Personal EmailId:
ashwinsajikumar@gmail.com
Your Address:
RSET,Kakkanad
Date of Birth (Current Entered Date:July 5, 2023)
dd-mm-yyyy
Education
Your 10th School Name:
BVME
Your 10th Percentage:
80
Your 10th Passout Year:
2017
Your 12th School Name:
BVME
Your 12th Percentage:
80
Your 12th Passout Year:
2019

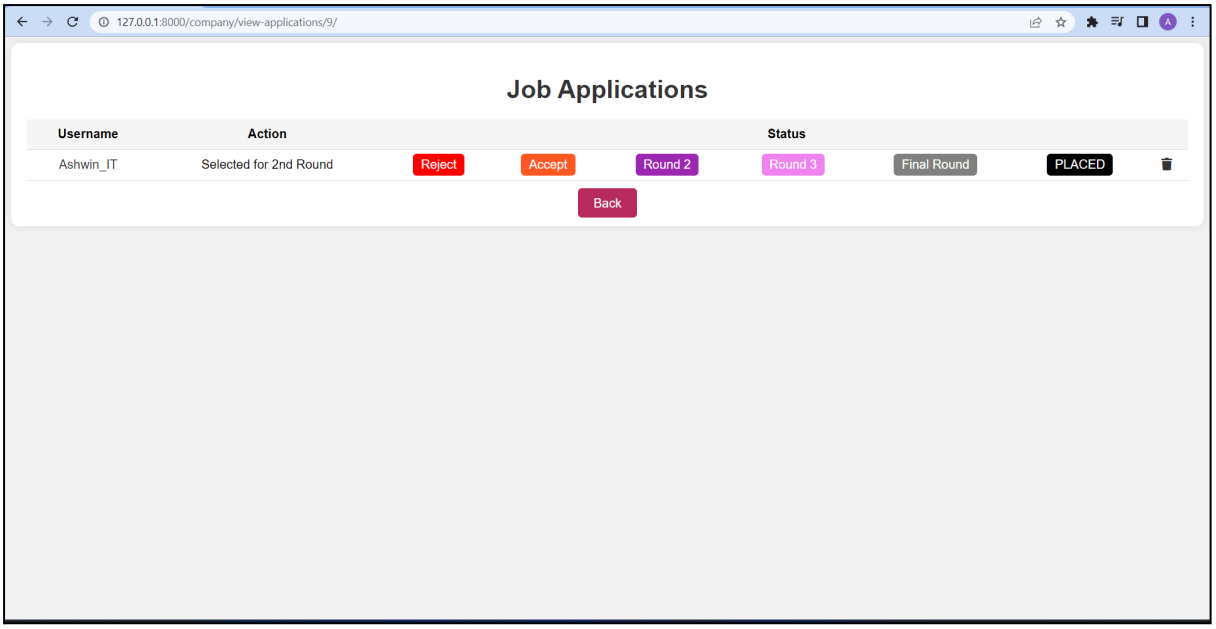

127.0.0.1:8000/student/view-jobs/
RPM
Welcome Ashwin
Student:IT
Dashboard
Edit Your Profile
Job Openings
Applied Jobs
Change Password
Log Out
Please Select Up to 3 Options
Infosys
Navigate your next
INFOSYS
100 Openings
Interested?

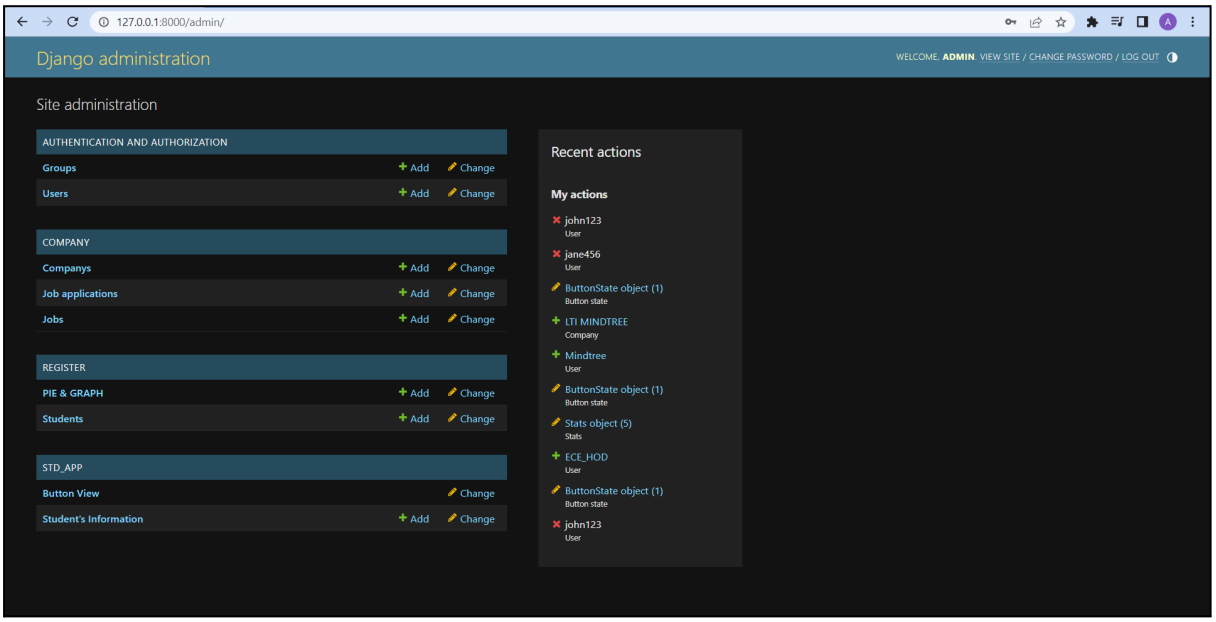

127.0.0.1:8000/student/applied-jobs/
RPM
Please Select Up to 3 Options
Welcome Ashwin
Student:IT
Dashboard
Edit Your Profile
Job Openings
Applied Jobs
Change Password
Log Out
TATA CONSULTANCY SERVICES
TCS
30 Openings
Status: Selected for 2nd Round
Remove Job?
LTIMindtree
LTI MINDTREE
100 Openings
Status: Congrats!You are Placed
Remove Job?

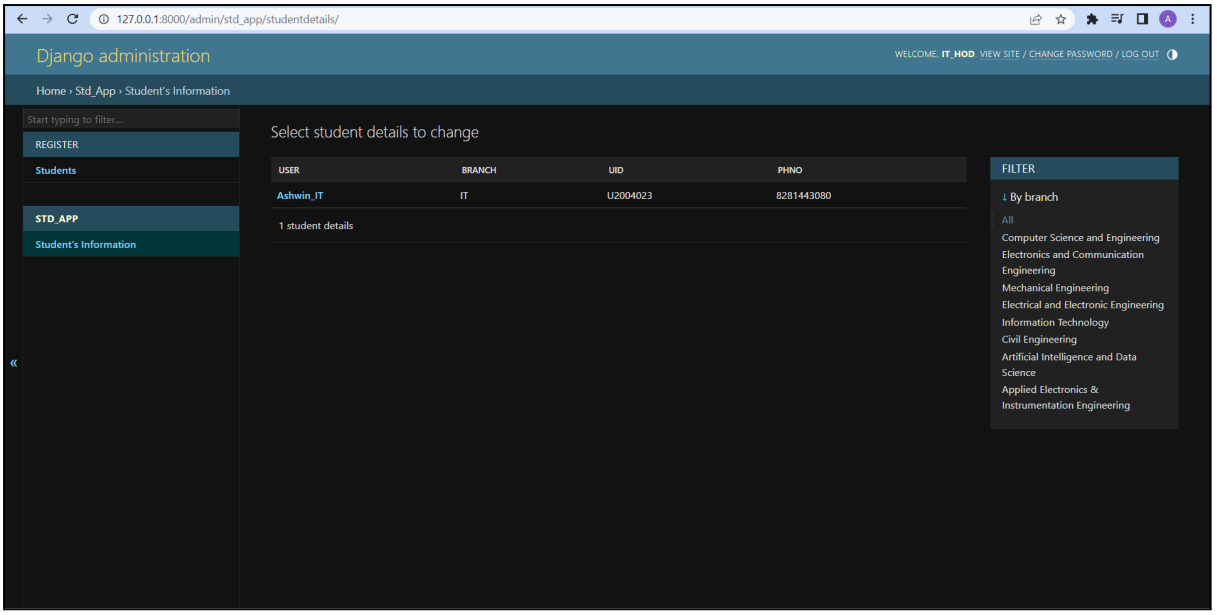

127.0.0.1:8000/student/change-password/
Change Password
Old password:
New password:
Your password can't be too similar to your other personal information.
Your password must contain at least 8 characters.
Your password can't be a commonly used password.
Your password can't be entirely numeric.
New password confirmation:
Change Password
Back

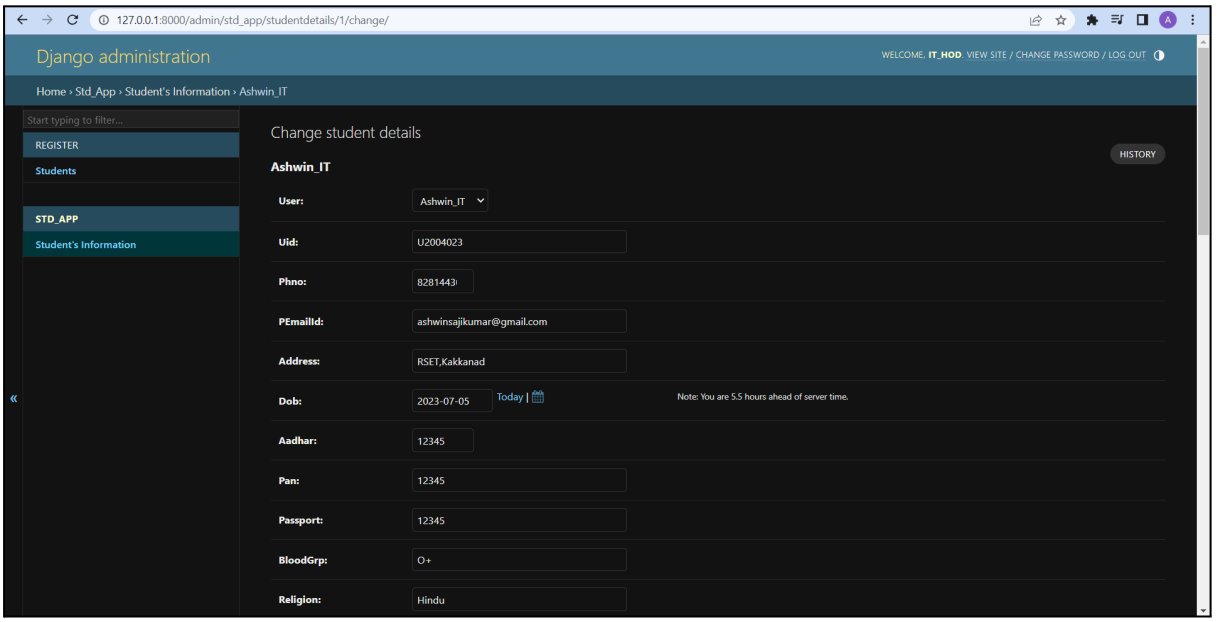

127.0.0.1:8000/company/home/
Welcome TCS
Your Posted Jobs
Software Developer
Slots: 30 | View Applications | Edit | Delete
Post a Job
Logout
127.0.0.1:8000/company/view-applications/9/


127.0.0.1:8000/company/post-job/
Edit Job
Slots:
30
Role:
Software Developer
CGPA:
8
Backpaper:
1
LPA:
8
Internship:
1
Tech:
HTML,CSS,Javascript
City:

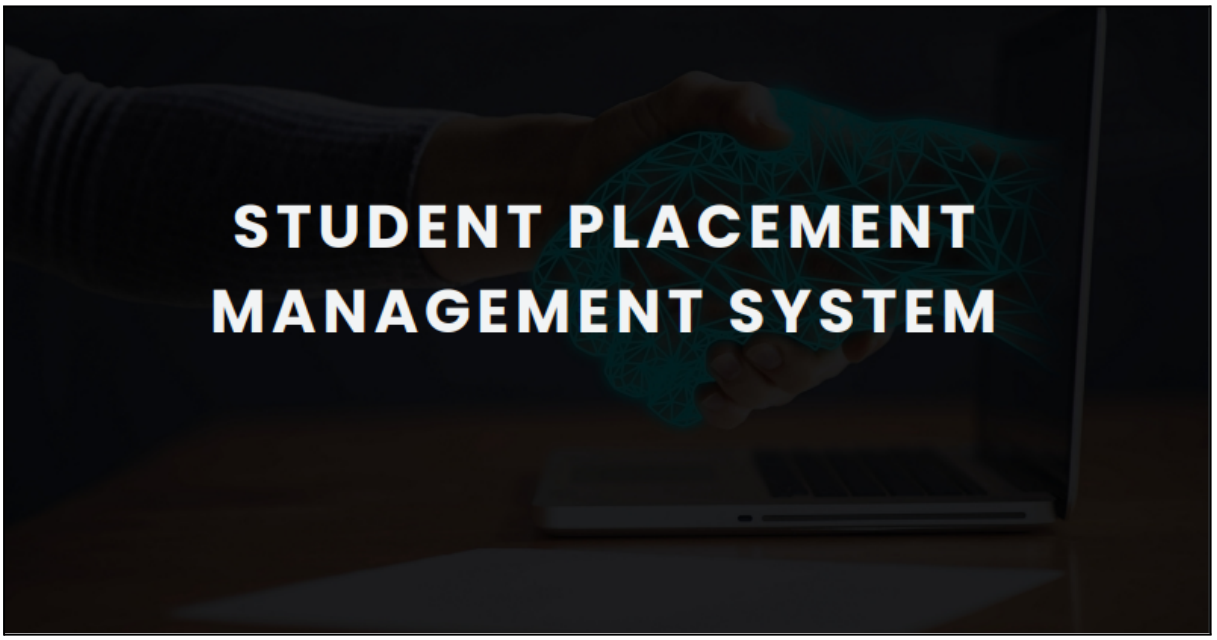

127.0.0.1:8000/company/view-applications/9/
Job Applications
Username
Action
Status
Ashwin_IT
Selected for 2nd Round
Reject
Accept
Round 2
Round 3
Final Round
PLACED
Back


127.0.0.1:8000/admin/
Django administration
WELCOME, ADMIN. VIEW SITE / CHANGE PASSWORD / LOG OUT
Site administration
AUTHENTICATION AND AUTHORIZATION
Groups
Users
COMPANY
Companys
Job applications
Jobs
REGISTER
PIE & GRAPH
Students
STD_APP
Button View
Student's Information
Add
Change
Recent actions
My actions
john123
jane456
ButtonState object (1)
LTI MINDTREE
Mindtree
Stats object (5)
ECE_HOD


127.0.0.1:8000/admin/std_app/studentdetails/
Django administration
WELCOME, IT_HOD VIEW SITE / CHANGE PASSWORD / LOG OUT
Home › Std_App › Student's Information
Start typing to filter...
REGISTER
Students
STD_APP
Student's Information
Select student details to change
USER
BRANCH
UID
PHNO
Ashwin_IT
IT
U2004023
8281443080
1 student details
FILTER
By branch
All
Computer Science and Engineering
Electronics and Communication Engineering
Mechanical Engineering
Electrical and Electronic Engineering
Information Technology
Civil Engineering
Artificial Intelligence and Data Science
Applied Electronics & Instrumentation Engineering


127.0.0.1:8000/admin/std_app/studentdetails/1/change/
Django administration
WELCOME, IT_HOD VIEW SITE / CHANGE PASSWORD / LOG OUT
Home › Std_App › Student's Information › Ashwin_IT
Start typing to filter...
REGISTER
Students
STD_APP
Student's Information
Change student details
HISTORY
Ashwin_IT
User: Ashwin_IT
Uid: U2004023
Phno: 8281443
PEmailId: ashwinsajikumar@gmail.com
Address: RSET,Kakkanad
Dob: 2023-07-05 Today |
Note: You are 5.5 hours ahead of server time.
Aadhar: 12345
Pan: 12345
Passport: 12345
BloodGrp: O+
Religion: Hindu

APPENDIX C
SLIDES



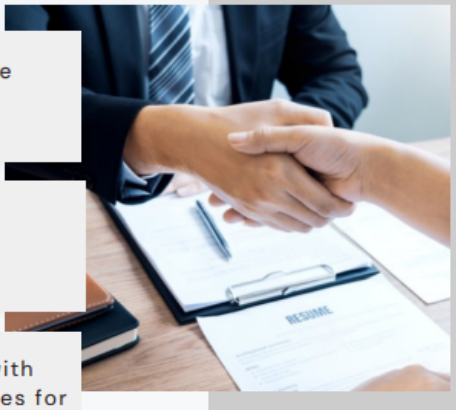
CONTENT

INTRODUCTION

Placement management system is a software for managing student placements within an organization

Streamlines hiring process and reduces manual efforts.

Matches students with companies aligned with their interests. Improves placement outcomes for students and companies.



PROBLEM STATEMENT

Traditional placement processes are often inefficient, time-consuming, leading to challenges for both employers and candidates.There is a need for a robust and user-friendly placement management system that addresses these challenges, providing a seamless and streamlined experience for employers and candidates. Such a system should automate and simplify the placement process, facilitating efficient communication, transparent tracking, and seamless coordination.

RELEVANCE

Efficiency

The system reduces manual effort, paperwork and enables faster processing of applications, improves communication, simplifies the overall placement management for institutions.

Centralized

The system provides a centralized platform for managing applications, scheduling interviews, and tracking placement offers, ensuring a smooth and organized experience for students.

Suitability

The system helps identify suitable placements for students based on their qualifications and skills. This increases the likelihood of students securing placements that align with their career goals, enhancing their employability and boosting placement success rates

EXISTING RELATED WORKS

Bullhorn

Bullhorn is a cloud-based recruitment CRM and placement management system. It offers features such as candidate sourcing, applicant tracking, resume parsing, interview management, and placement tracking.

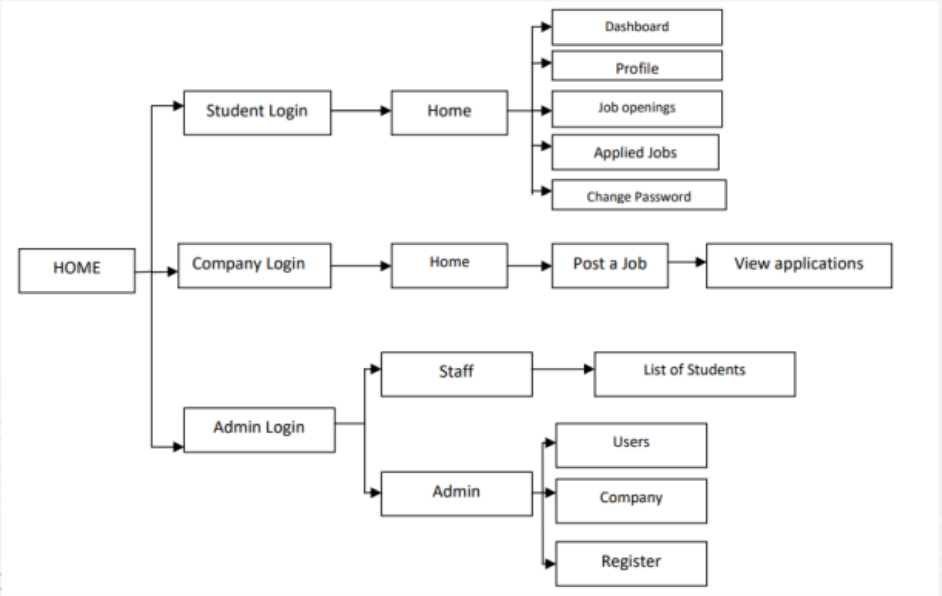
JOBVITE

Jobvite is a popular placement management system that offers end-to-end recruitment solutions. It provides features such as job posting, candidate screening, applicant tracking, and interview management.

Taleo

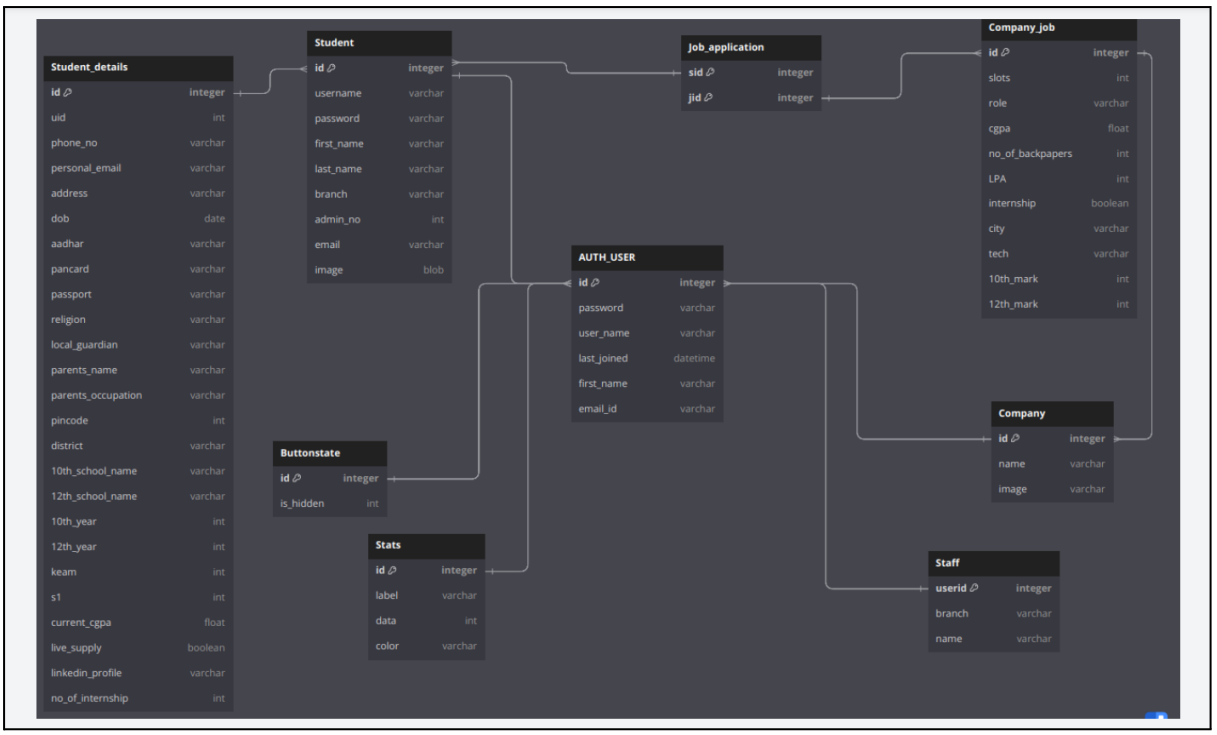
Taleo, now a part of Oracle, is a widely used talent management system that includes a comprehensive module for placement management. It offers features like job requisition management, candidate sourcing, resume parsing.

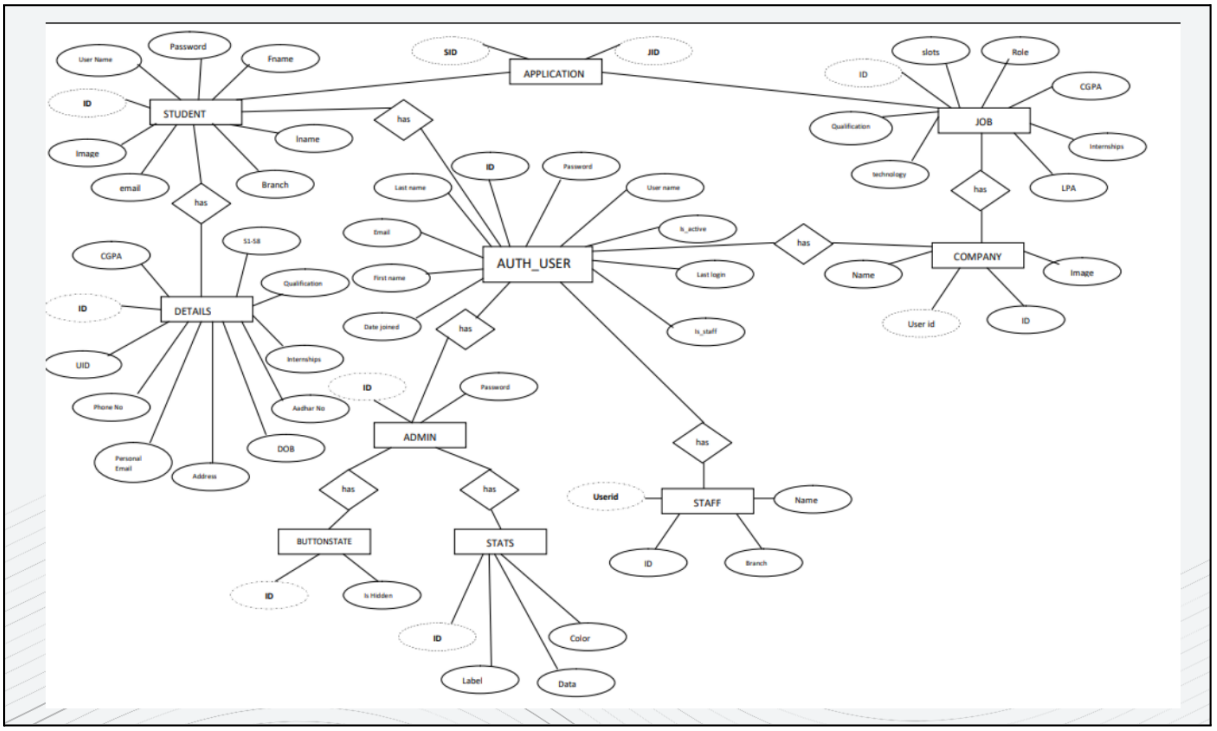
BLOCK DIAGRAM



Modules

1	Home Page	Login page for all users
2	Student Login	Student can login to their profile with user id and password given by admin
3	Admin/Staff Login	Admin/Staff can login to their profile with their user id and password
4	Company Login	Company can login to their profile with their user id and password
5	Dashboard	Students can view the previous year placement Statistics which is updated by the admin
6	Edit Your Profile	Students are supposed to fill their personal and academic details
7	Job openings	Student can view the available job openings if they are interested they can apply.
8	Applied Jobs	Students can view the details of applied jobs like level of selection etc
9	Change Password	Students are supposed to change their default password given by the admin
10	Company - Home	Company can Post a new Job with eligibility criteria and can see the applications of students to that job
11	Company Login	Company can login to their profile with their user id and password
12	Admin - Home	Site administration and also creates account and manages for all the users
13	Staff - Home	Can access to students information of their corressponding department only


Student_details
id integer
uid int
phone_no varchar
personal_email varchar
address varchar
dob date
aadhar varchar
pancard varchar
passport varchar
religion varchar
local_guardian varchar
parents_name varchar
parents_occupation varchar
pincode int
district varchar
10th_school_name varchar
12th_school_name varchar
10th_year int
12th_year int
keam int
s1 int
current_cgpa float
live_supply boolean
linkedin_profile varchar
no_of_internship int
Student
id integer
username varchar
password varchar
first_name varchar
last_name varchar
branch varchar
admin_no int
email varchar
image blob
Job_application
sid integer
jid integer
Company_job
id integer
slots int
role varchar
cgpa float
no_of_backpapers int
LPA int
internship boolean
city varchar
tech varchar
10th_mark int
12th_mark int
AUTH_USER
id integer
password varchar
user_name varchar
last_joined datetime
first_name varchar
email_id varchar
Buttonstate
id integer
is_hidden int
Stats
id integer
label varchar
data int
color varchar
Company
id integer
name varchar
image varchar
Staff
userid integer
branch varchar
name varchar


APPLICATION
SID
JID
STUDENT
Password
Fname
Lname
Image
email
Branch
has
DETAILS
CGPA
S1-S8
Qualification
UID
Internships
Phone No
Aadhar No
DOB
Personal Email
Address
AUTH_USER
Email
First name
Last name
Date joined
Is_active
Last login
Is_staff
User name
JOB
slots
Role
CGPA
Internships
technology
LPA
COMPANY
Name
Image
User id
ADMIN
BUTTONSTATE
Is Hidden
STATS
Color
Label
Data
STAFF
Userid
Branch

IMPLEMENTATION DETAILS

Visual studio Code

- DE : Visual studio Code : is very powerful with a range of features and a fairly user friendly interface. It supports multiple programming languages and is a must have tool for developers who require a robust development environment.

Database : SQLite -SQLite is a C-language library that implements a small, fast, self-contained, high-reliability, full-featured, SQL database engine.

SQLite

IMPLEMENTATION DETAILS

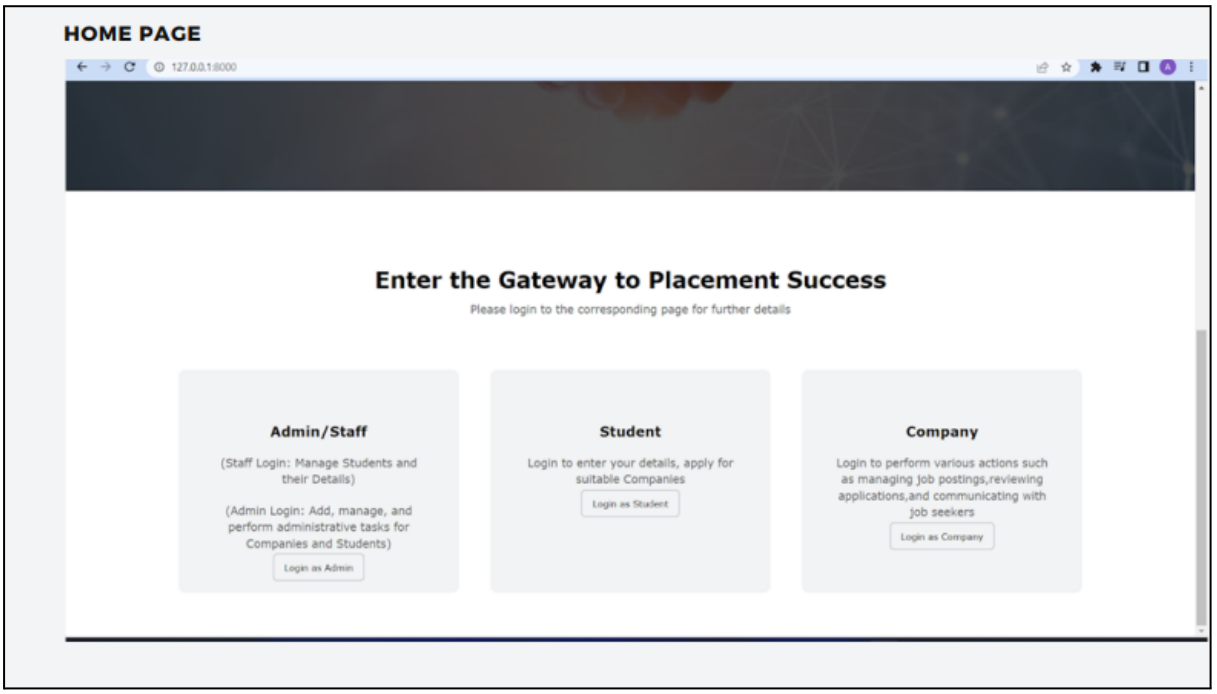
Python Django

Platform : Python Django -High level python web framework that enables rapid development of secure and maintainable websites.

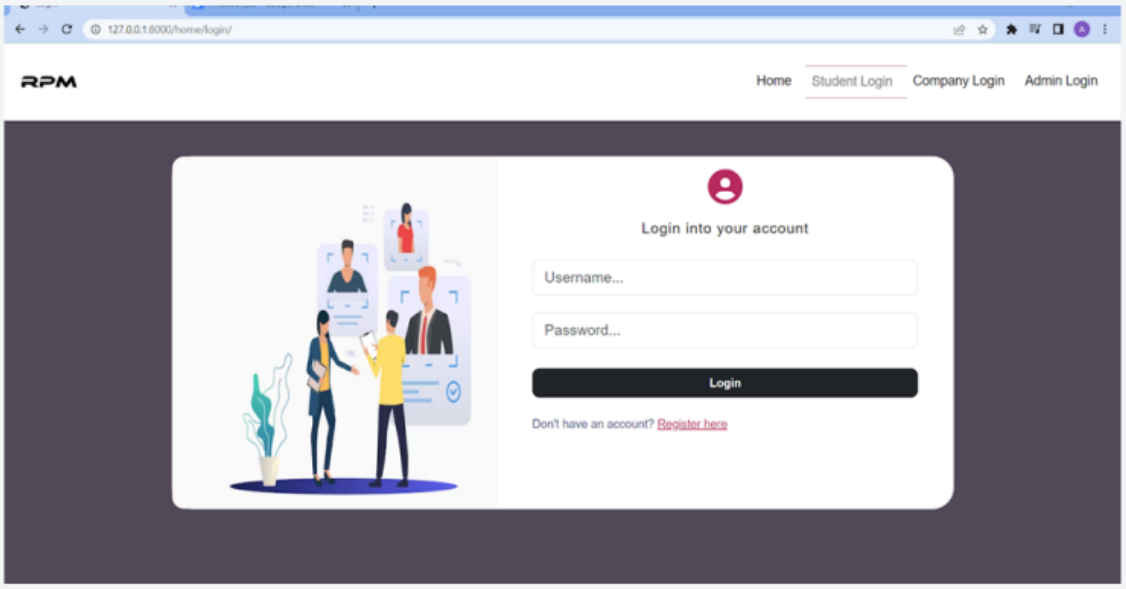
Programming Languages :
Javascript , Python ,HTML,CSS

Web Development

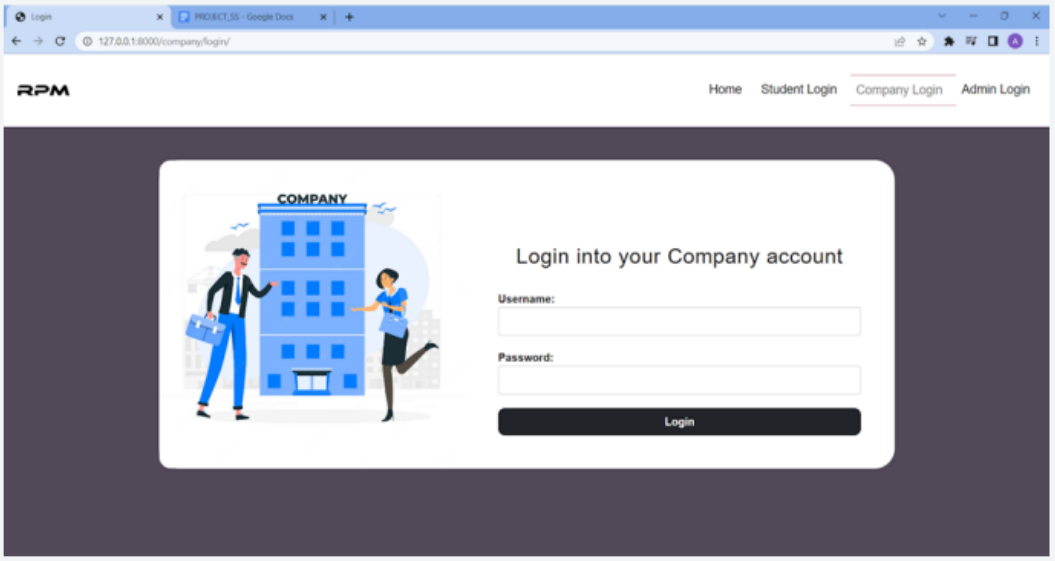

SCREENSHOT OF WORK DONE


HOME PAGE
127.0.0.1:8000
Enter the Gateway to Placement Success
Please login to the corresponding page for further details
Admin/Staff
(Staff Login: Manage Students and their Details)
(Admin Login: Add, manage, and perform administrative tasks for Companies and Students)
Login as Admin
Student
Login to enter your details, apply for suitable Companies
Login as Student
Company
Login to perform various actions such as managing job postings,reviewing applications,and communicating with job seekers
Login as Company

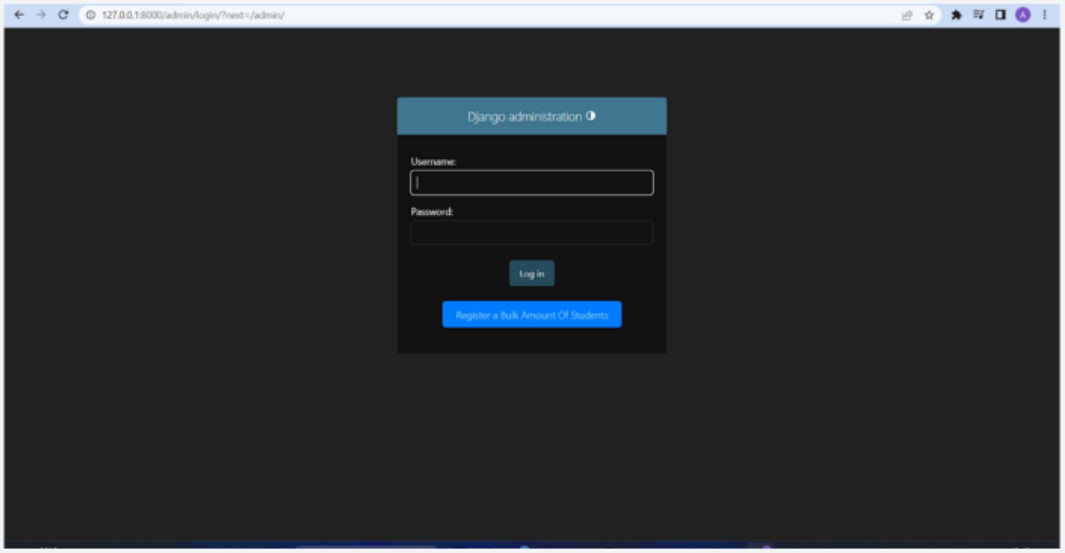
STUDENT LOGIN PAGE



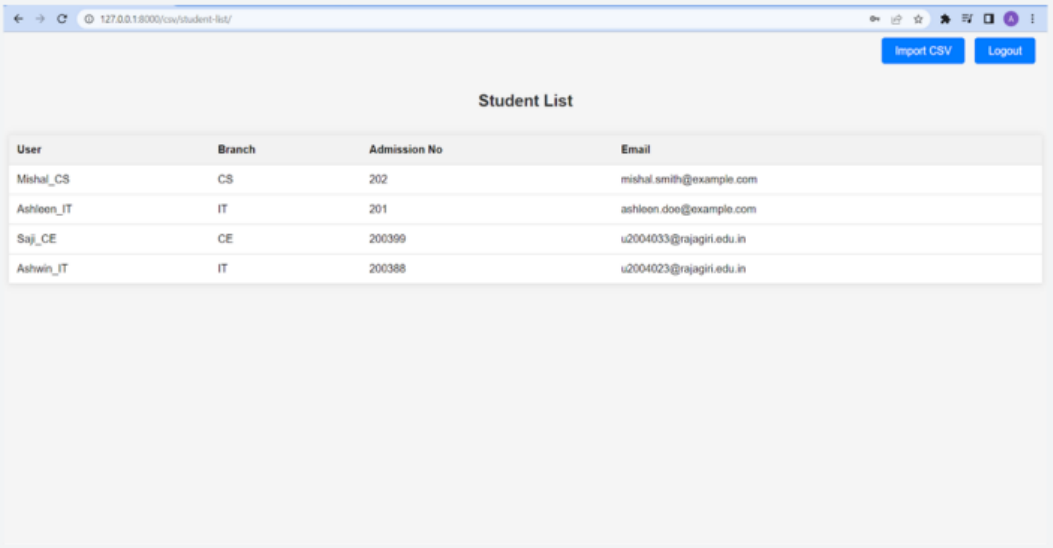
COMPANY LOGIN PAGE



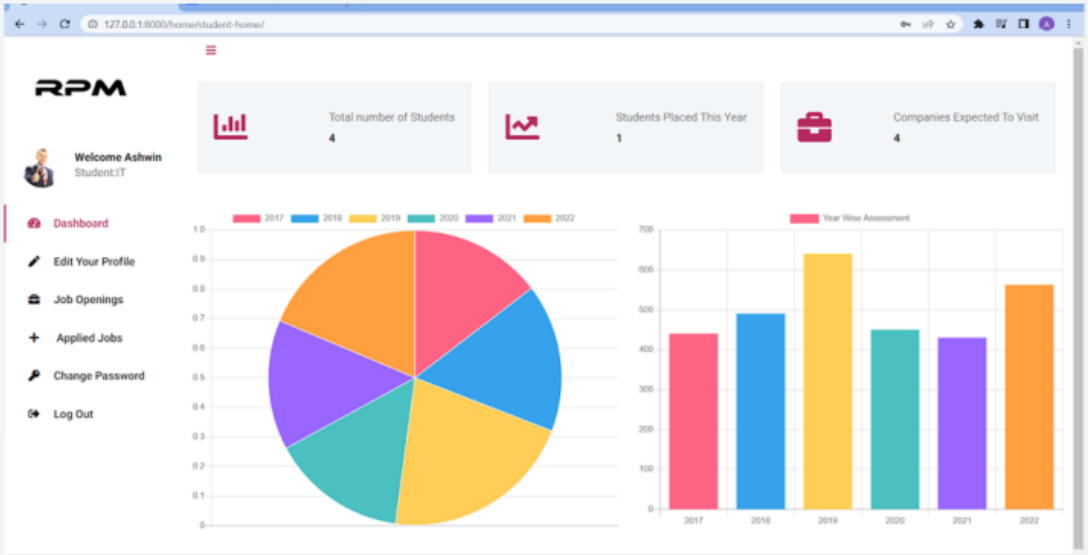
ADMIN/STAFF LOGIN PAGE



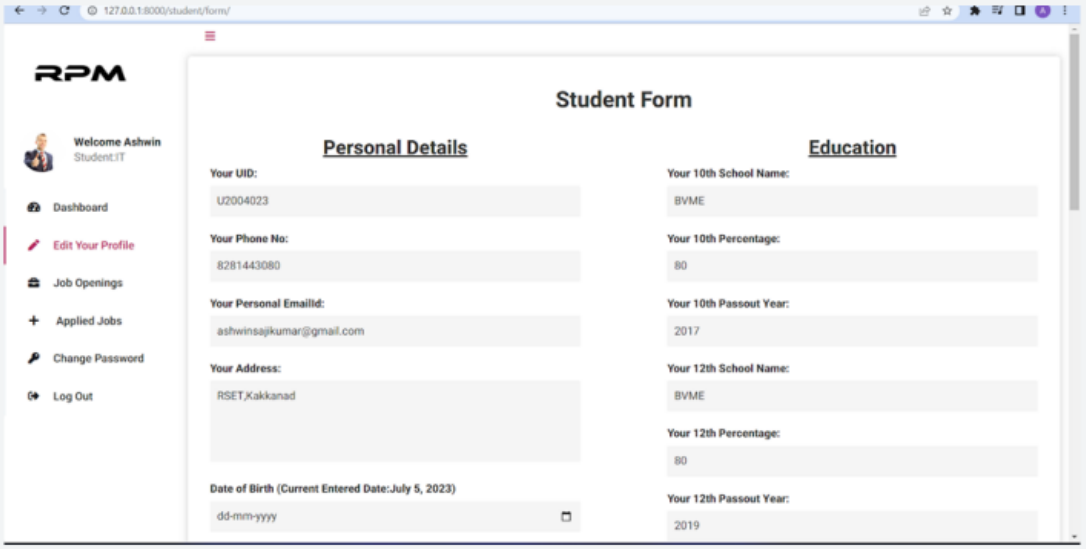
IMPORTING THE CSV FILE TO REGISTER NEW STUDENTS



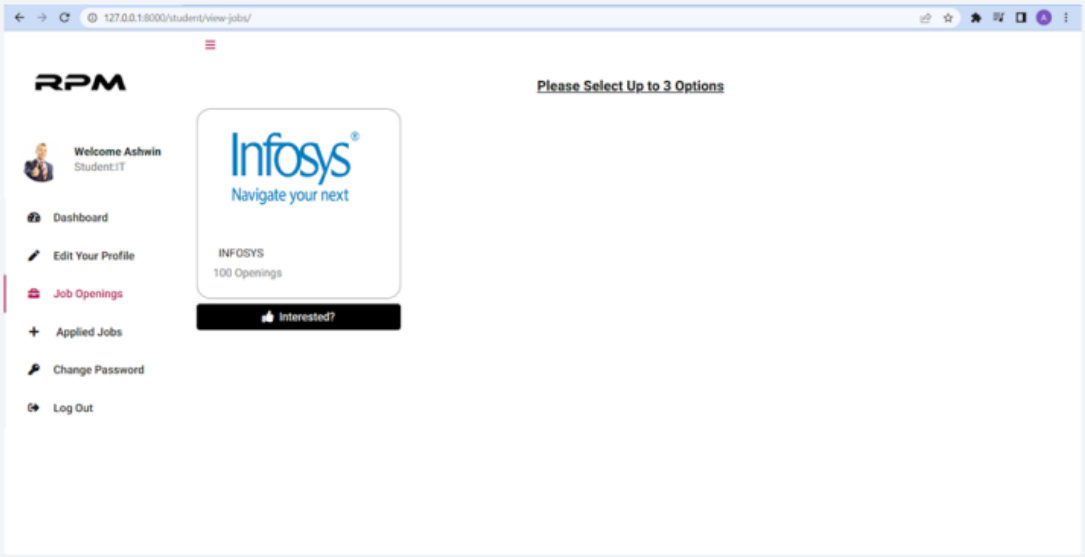
STUDENT HOME PAGE



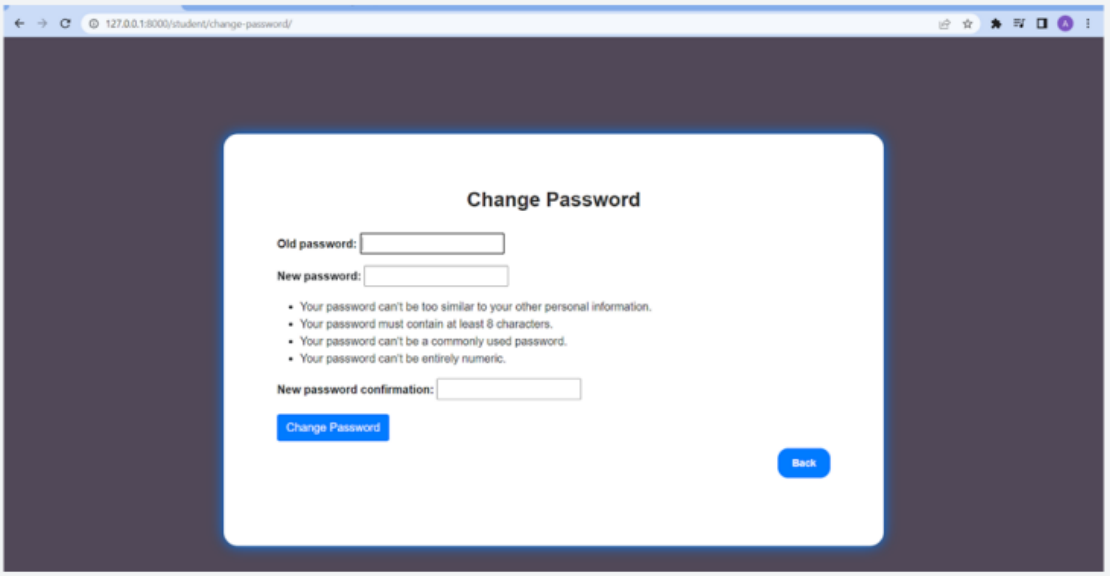
STUDENT FORM



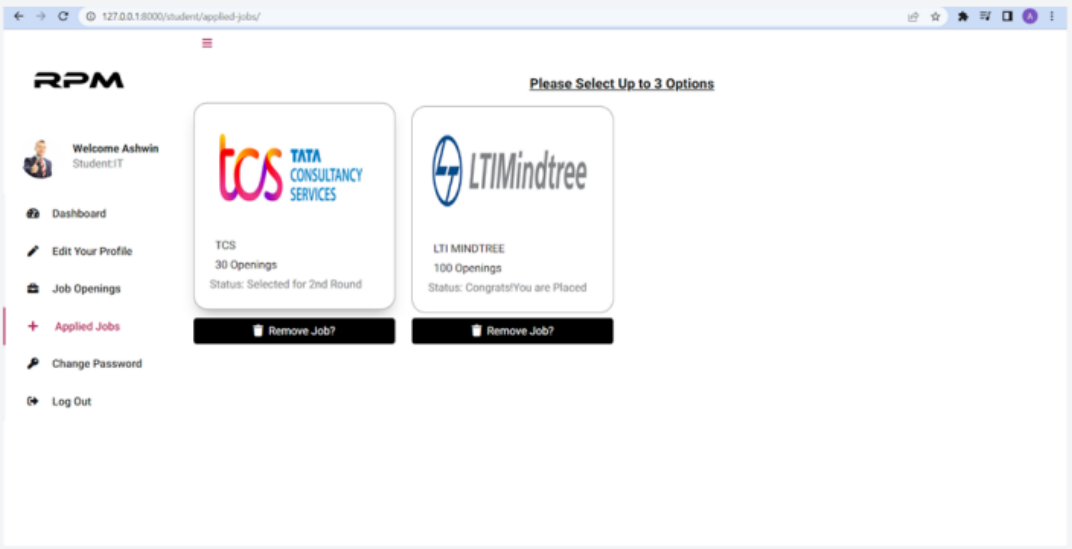
STUDENT JOB LIST



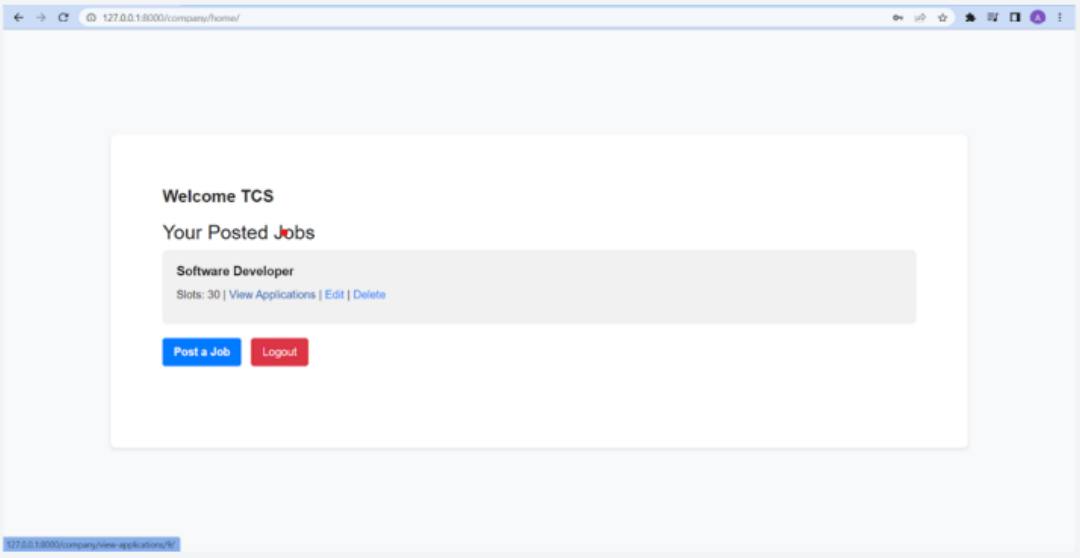
STUDENT PASSWORD CHANGE OPTION

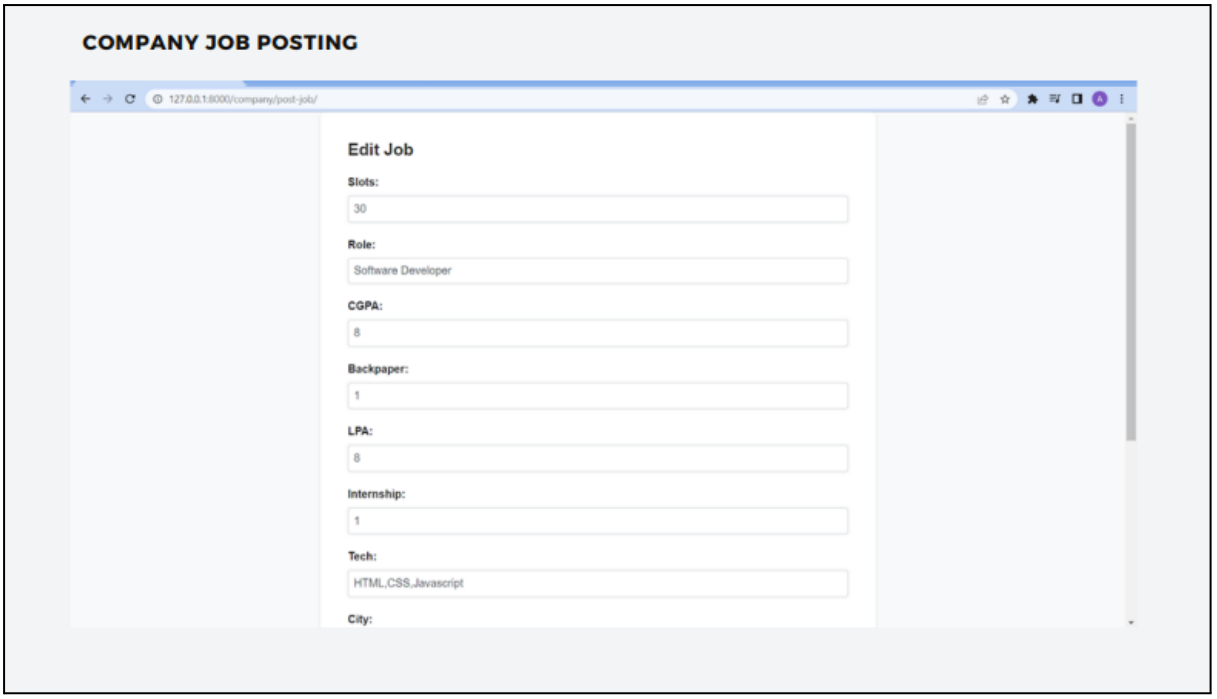


STUDENT SELECTED JOBS



COMPANY HOME PAGE





TASK ASSIGNMENT

Research	Ashleen,Mishal
Database setup	Ashwin Saji Kumar
Developed API routes for students and Company	Ashwin Saji Kumar
Build APIs for admin/staff	Ashwin Saji Kumar
User login and registration	Ashwin Saji Kumar
Documentation	Ashleen Sunil
Web page designing	Mishal

Events	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10	Week 11
Researching											
Selection of Project											
Planning											
Users Login Authentication											
Database Setup											
Build API for Student and Company											
Build API for Admin and Staff											
Webpage Designing											
Implementation											
Documentation											

THANKYOU

The student placement management website is a comprehensive and user-friendly platform that is efficient, centralized, and suitable for students and institutions.

APPENDIX D

RAJAGIRI SCHOOL OF ENGINEERING AND TECHNOLOGY (AUTONOMOUS)

DEPARTMENT OF INFORMATION TECHNOLOGY

PROGRAMME: INFORMATION TECHNOLOGY

VISION

To evolve into a department of excellence in information technology by the creation and exchange of knowledge through leading-edge research, innovation and services, which will in turn contribute towards solving complex societal problems and thus building a peaceful and prosperous mankind.

MISSION

To impart high-quality technical education, research training, professionalism and strong ethical values in the young minds for ensuring their productive careers in industry and academia so as to work with a commitment to the betterment of mankind.

PROGRAM EDUCATIONAL OBJECTIVES (PEO)

Graduates of Information Technology program shall

PEO 1: Have strong technical foundation for successful professional careers and to evolve as key-players / entrepreneurs in the field of information technology.

PEO 2: Excel in analyzing, formulating and solving engineering problems to promote life-long learning, to develop applications, resulting in the betterment of the society.

PEO 3: Have leadership skills and awareness on professional ethics and codes.

PROGRAM OUTCOMES (PO)

Information Technology program students will be able to:

PO 1. Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO 2. **Problem analysis**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO 3. **Design/development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO 4. Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO 5. Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

PO 6. The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO 7. Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO 8. Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO 9. Individual and teamwork: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO 10. Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO 11. Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO 12. Life-long learning: Recognize the need for and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

PROGRAM SPECIFIC OUTCOMES (PSO)

Information Technology program students will be able to:

PSO1: Acquire skills to design, analyze and develop algorithms and implement them using high-level programming languages**.**

PSO2: Contribute their engineering skills in computing and information engineering domains like network design and administration, database design and knowledge engineering.

PSO3: Develop strong skills in systematic planning, developing, testing implementing and providing IT solutions for different domains which helps in the betterment of life.

COURSE OBJECTIVES:

This course is designed for enabling the students to apply the knowledge to address the real-world situations/problems and find solutions. The course is also intended to estimate the ability of the students in transforming theoretical knowledge studied as part of the curriculum so far into a working model of a software system. The students are expected to design and develop a software/hardware project to innovatively solve a real-world problem.

COURSE OUTCOMES:

After completion of the course the student will be able to

Sl.NO	**DESCRIPTION**	**Blooms' Taxonomy Level**
CO1	Make use of acquired knowledge within the selected area of technology for project development.	Level 3: Apply
CO2	Identify, discuss and justify the technical aspects and design aspects of the project with a systematic approach.	Level 3: Apply
CO3	Interpret, improve and refine technical aspects for engineering projects.	Level 3: Apply
CO4	Associate with a team as an effective team player for the development of technical projects.	Level 3: Apply
CO5	Report effectively the project related activities and findings.	Level 2: Understand

CO-PO AND CO-PSO MAPPING

	P O1	P O2	P O3	P O4	P O5	P O6	P O7	P O8	P O9	PO 10	PO 11	PO 12	PSO 1	PSO 2	PS O3

CO1	3	3	3	3	3	3	3	3				3	3	3	3
CO2	3	3	3	3	3		2	3		3	2	3	3		3
CO3	3	3	3	3	3	2	3	3		2	3	3	2	2	2
CO4	3	3	2	2				3	3	3	3	3			
CO5	3				2			3	2	3	2	3			

3/2/1: high/medium/low