1. INTRODUCTION

Finding the perfect accommodation in the hustle and bustle of life can be an intimidating task. People may feel overwhelmed by academic responsibilities, social engagements, and overall transition to independent living. We are excited to introduce "Roomie" Room Rental Application as we have experienced the challenges of finding suitable housing.
Roomie is not just another housing platform; it is a customized solution designed to make securing accommodation easier. Our application aims at being the link between people and their ideal living spaces because we know what people want and need.

1.1 OBJECTIVES

The objectives of our Room Rental Application project can encompass various aspects, aligning with both business and user needs. Here is a list of potential objectives for our project:

I. User-Friendly Interface: Develop an intuitive and user-friendly interface to ensure ease of navigation and a positive user experience for individuals searching for rental accommodations.

II. Comprehensive Listings: Create a platform that offers a wide range of verified and comprehensive room listings, catering to different preferences, budgets, and needs of potential tenants.

III. Immersive Virtual Tours: Integrate virtual tour features to allow users to explore potential living spaces virtually, providing a realistic and immersive experience that aids in decision-making.

IV. Transparent Pricing and Policies: Ensure transparency in pricing and rental policies for each listed room, fostering trust and confidence among users and reducing the likelihood of disputes.

V. Secure Transactions: Implement a secure payment gateway to facilitate seamless and secure financial transactions between landlords and tenants within the application.

1.2 Advantages:

- **Convenience:** Online platforms offer convenience in finding, comparing, and booking rooms from anywhere at any time, eliminating the need for physical visits to multiple locations.

- **Variety of Options:** Users have access to a wide range of rooms, apartments, and houses across different locations and price ranges, providing more choices to suit individual preferences.

- **Transparent Reviews:** Platforms often feature reviews and ratings from previous tenants, helping renters make informed decisions based on others' experiences.

- **Ease of Payment**: Many online rental platforms offer secure payment options, making it convenient for both tenants and landlords to handle transactions electronically.

- **Time-Saving:** Renters can quickly filter through listings based on their preferences (price, location, amenities) rather than physically visiting numerous places.

1.3 Disadvantages:

- **Scams and Fraud:** There's a risk of encountering fraudulent listings or scams where the property doesn't match the description or may not even exist, leading to financial loss and inconvenience.

- **Inaccurate Information:** Sometimes, the information provided online about a property might be outdated or inaccurate, leading to misunderstandings or dissatisfaction upon arrival.

- **Limited Inspection:** Renters might not have the chance to inspect the property thoroughly before renting, leading to surprises or issues after moving in.

- **Dependency on Technology:** Technical glitches or website errors can disrupt the booking process or lead to misunderstandings between the tenant and landlord.

- **Lack of Personal Touch:** Online rentals lack the personal interaction that can sometimes help build a better understanding between tenants and landlords, potentially leading to communication issues or misunderstandings.

2. SYSTEM ANALYSIS

→ Here are some key aspects of system analysis:

- **Requirements Gathering**: Identify and document the functional and non-functional requirements of the Room Rental Application.
- **Use Case Analysis:** Define the various use cases and user scenarios the application should support. Identify primary actors (users, administrators, landlords).
- **Data Modeling:** Design the database schema to support the storage and retrieval of relevant data.
- **Non-functional Requirements:** Draft security measures, including user verification, data encryption, and secure payment processing.
- **Security Analysis:** Evaluate potential security threats and establish measures to mitigate risks. Conduct a security assessment, including penetration testing.

2.1 System Requirements:

A. Hardware Requirements:

- Processor: Dual-core processor (2.5 GHz or higher)
- RAM: 8GB DDR4 or higher
- Storage: 128GB SSD or higher
- Display: 1920x1080 resolution or higher

B. Software Requirements:

- Operating System: Windows 10, 11
- Web Server: XAMPP 7.4.3 or later
- Web Browser: Google Chrome
- Code Editor: Visual Studio Code 1.60.2

2.2 IMPLEMENTING TOOLS:

1. XAMPP:

XAMPP is a free and open-source cross-platform web server solution stack package developed by Apache Friends. The name "XAMPP" is an acronym that stands for:

X - Cross-platform (it works on various operating systems)

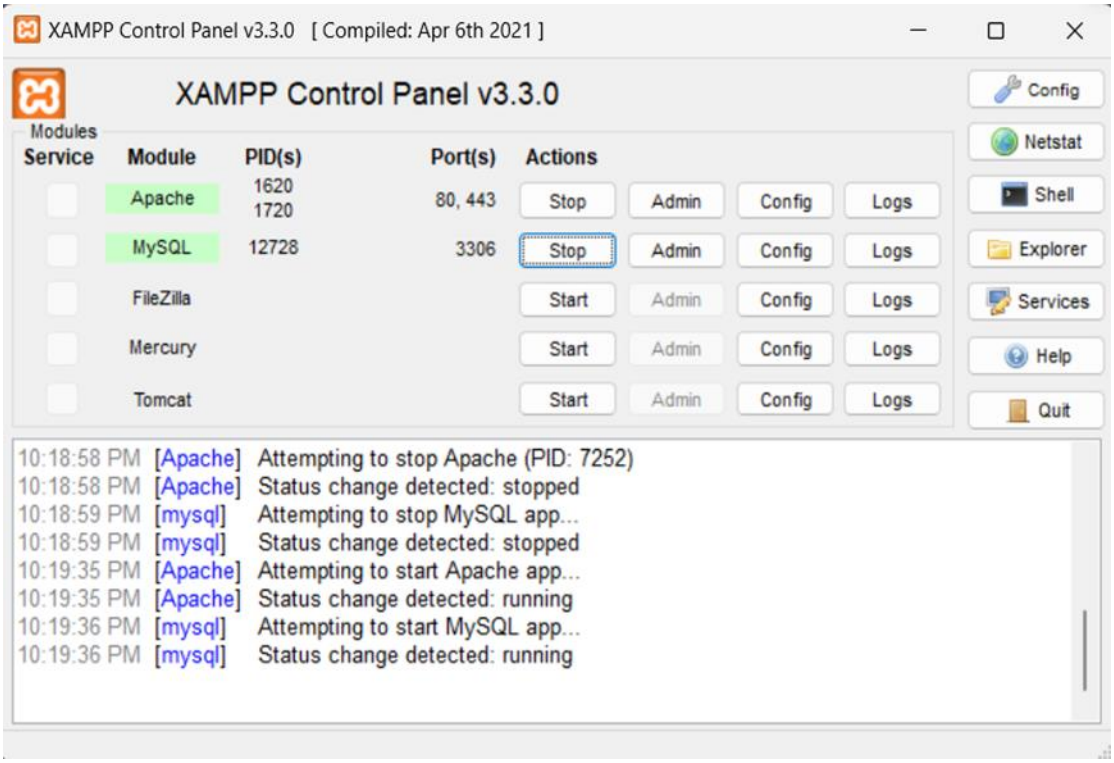
A - Apache (the web server)

M - MySQL (the database management system)

P - PHP (a server-side scripting language)

P - Perl (a programming language)

XAMPP is designed to make it easy to set up a local web development environment, allowing developers to test their websites or web applications on their own computer before deploying them to a live server. It includes all the necessary components to run a web server, such as Apache for handling HTTP requests, MySQL for database management, and PHP/Perl for server-side scripting.



2. HTML:

HTML, or HyperText Markup Language, is the standard markup language used to create and design documents on the World Wide Web. It is a key component of web development and provides a structured way to format content on the web. HTML documents are interpreted by web browsers to display text, images, links, forms, and other elements.

3. CSS:

CSS, or Cascading Style Sheets, is a style sheet language used to describe the presentation and layout of HTML documents. It defines how elements should be displayed on the screen, in print, or in other media. CSS enables web developers to control the appearance of web pages, ensuring consistency and flexibility in the design across different devices and screen sizes.

4. JavaScript:

JavaScript is a high-level, versatile programming language primarily known for its use in web development. It allows developers to add interactivity, manipulate the DOM (Document Object Model), handle events, and create dynamic content on websites. JavaScript is an essential component of modern web development and is supported by all major web browsers.

5. Security:

Security in the context of web development involves implementing measures to protect websites, web applications, and users from various threats and vulnerabilities. It's a critical aspect of the development process to ensure the confidentiality, integrity, and availability of data. HTTPS is implemented for secure communication, and security practices include input validation, parameterized queries, and protection against SQL injection and cross-site scripting (XSS).

6. Laravel:

Laravel is a free, open-source PHP web framework designed for the development of robust and maintainable web applications. It follows the Model-View-Controller (MVC) architectural pattern and provides elegant syntax and tools for tasks such as routing, caching, database interactions, and more. Laravel aims to make the development process enjoyable and efficient by emphasizing clean, expressive code.

7. MySQL:

MySQL is an open-source relational database management system (RDBMS) that is widely used for building and managing databases. It is a key component of the LAMP (Linux, Apache, MySQL, PHP/Python/Perl) and MERN (MongoDB, Express.js, React, Node.js) stacks and is often used in conjunction with web applications. MySQL is known for its performance, reliability, ease of use, and strong community support.

3. MODULE

Module is a type of diagram in which each of a set of standardized part or independent units that can be used to construct a more complex structure.

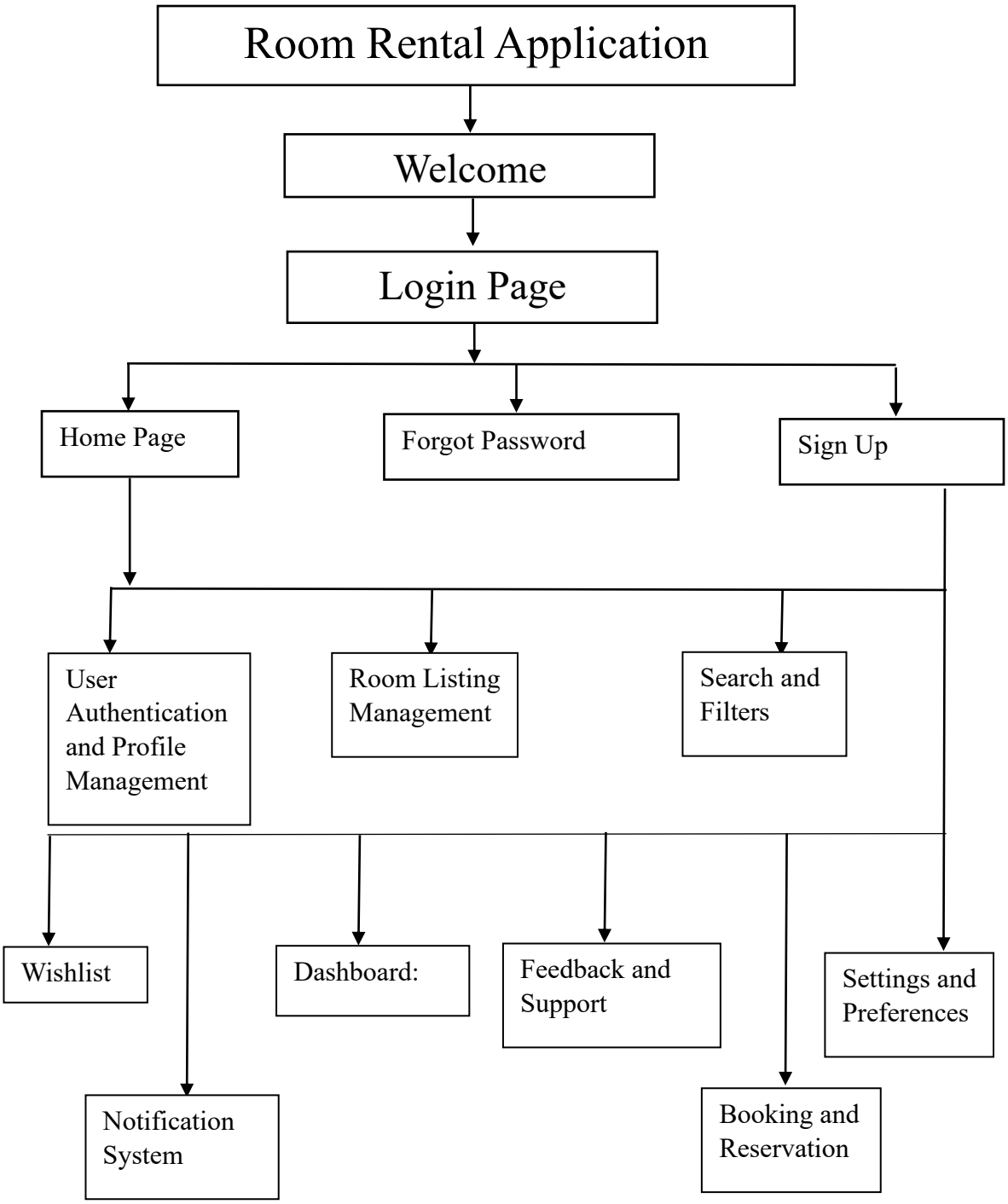


Fig 1: Module of Room Rental Application

4. FUNCTION ORIENTED DIAGRAM

The function-oriented diagram illustrates the step-by-step flow of processes within each module, detailing the inputs, processes, and outputs for comprehensive understanding.

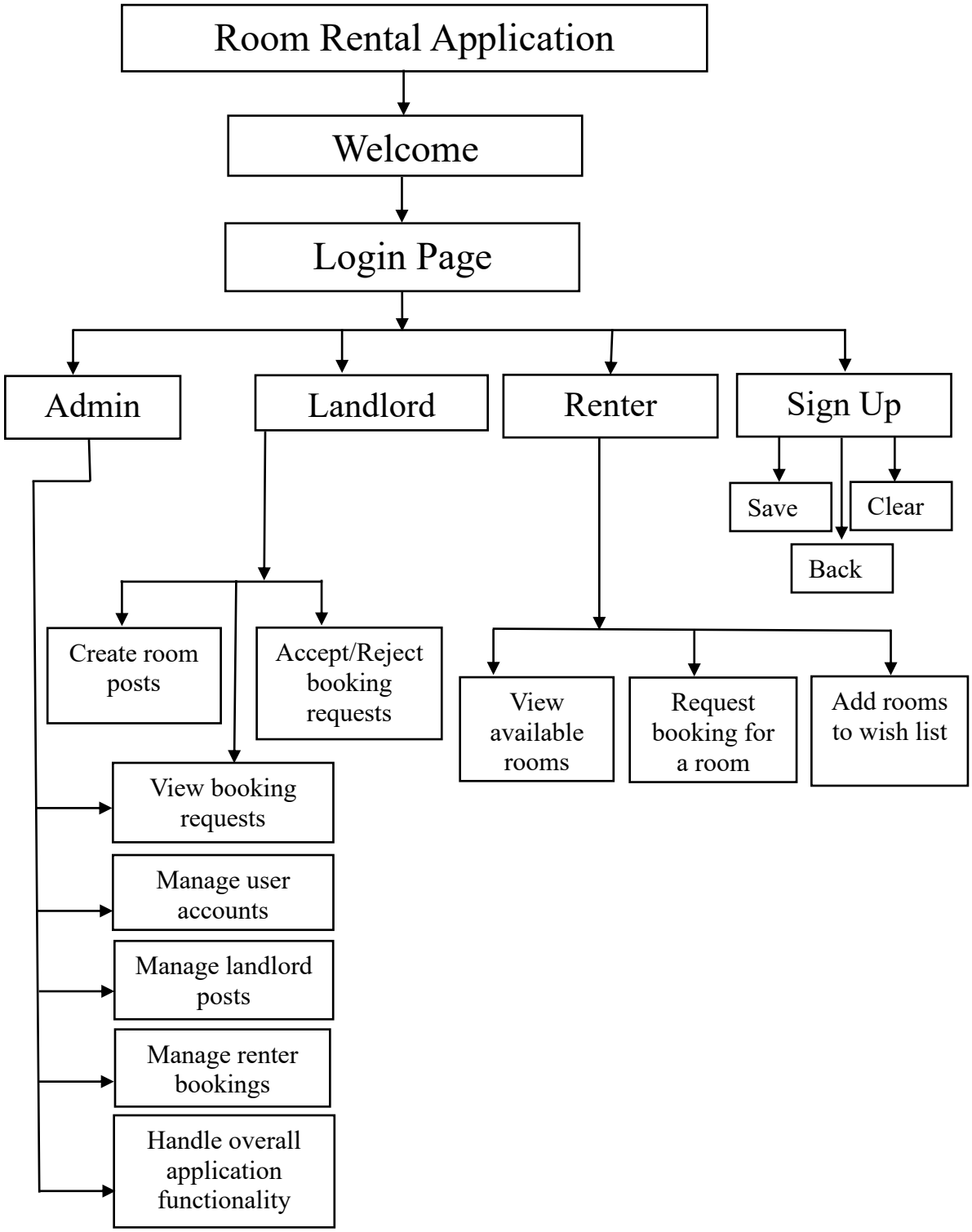
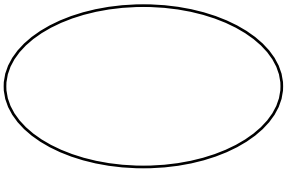
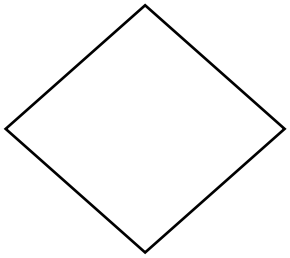
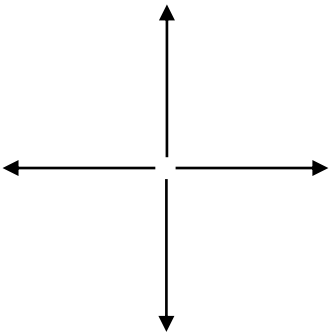


Fig 2: Function Oriented Diagram of Room Rental Application

5. Flowchart:

Flowchart can be defined as "the graphical representation of an algorithm." The flowchart is an easy way to understand and analyze the problem.

Flowchart uses some geometric figures to specifies operation. They are given below with their name and functions:

Geometric Figures	Names	Functions
	Oval / Ellipse	It is used as starting and stopping of the flowchart.
	Parallelogram	It is used as to accept input data and display the output result.
	Diamond	It is used to decision making process.
	Rectangle	It is used to process the data.
	Flow Lines	It is used as connectors.

5.1 Flowchart Diagram of Login Form:

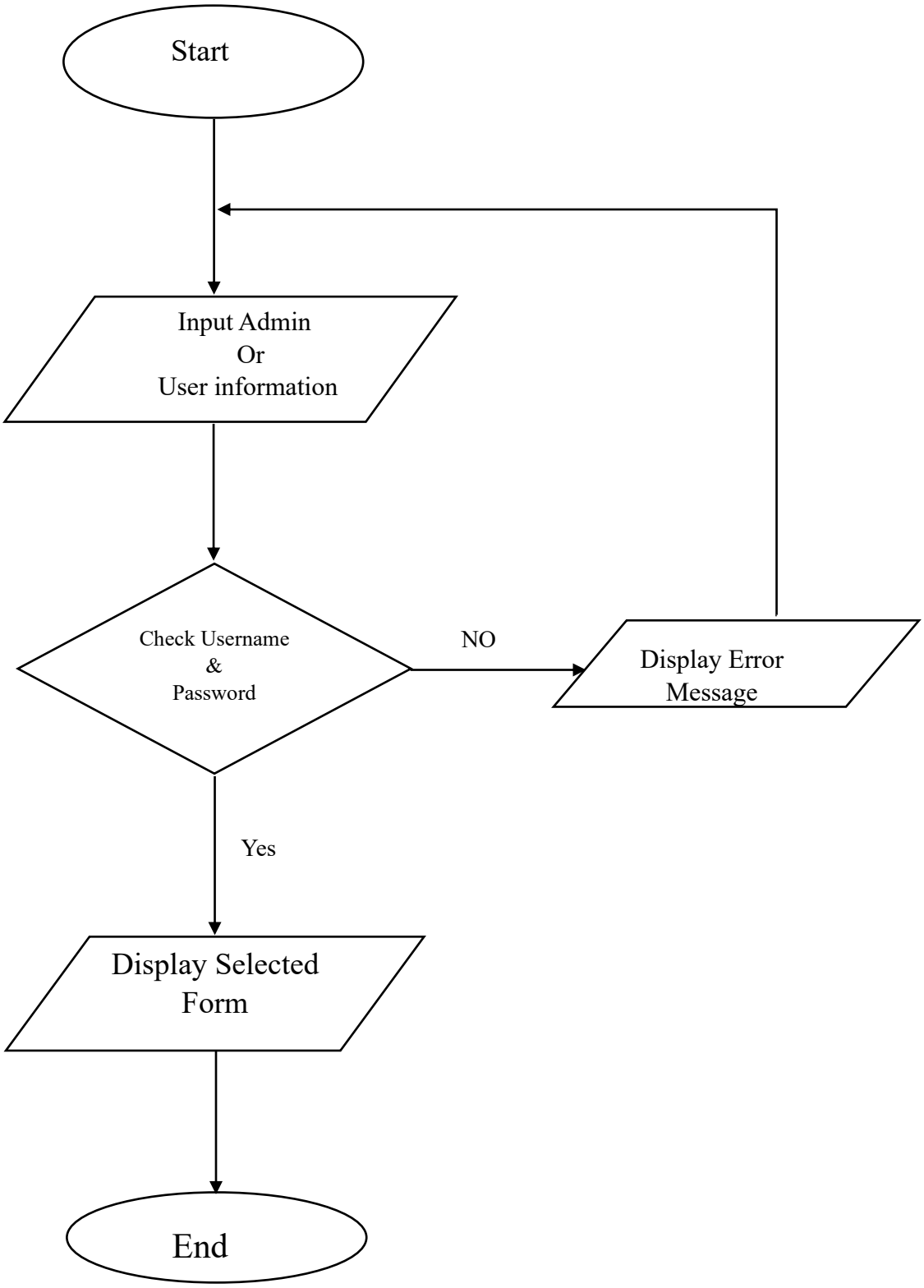


Fig 3.1: Flowchart Diagram of Login Form

5.2 Flowchart Diagram of Register Form:

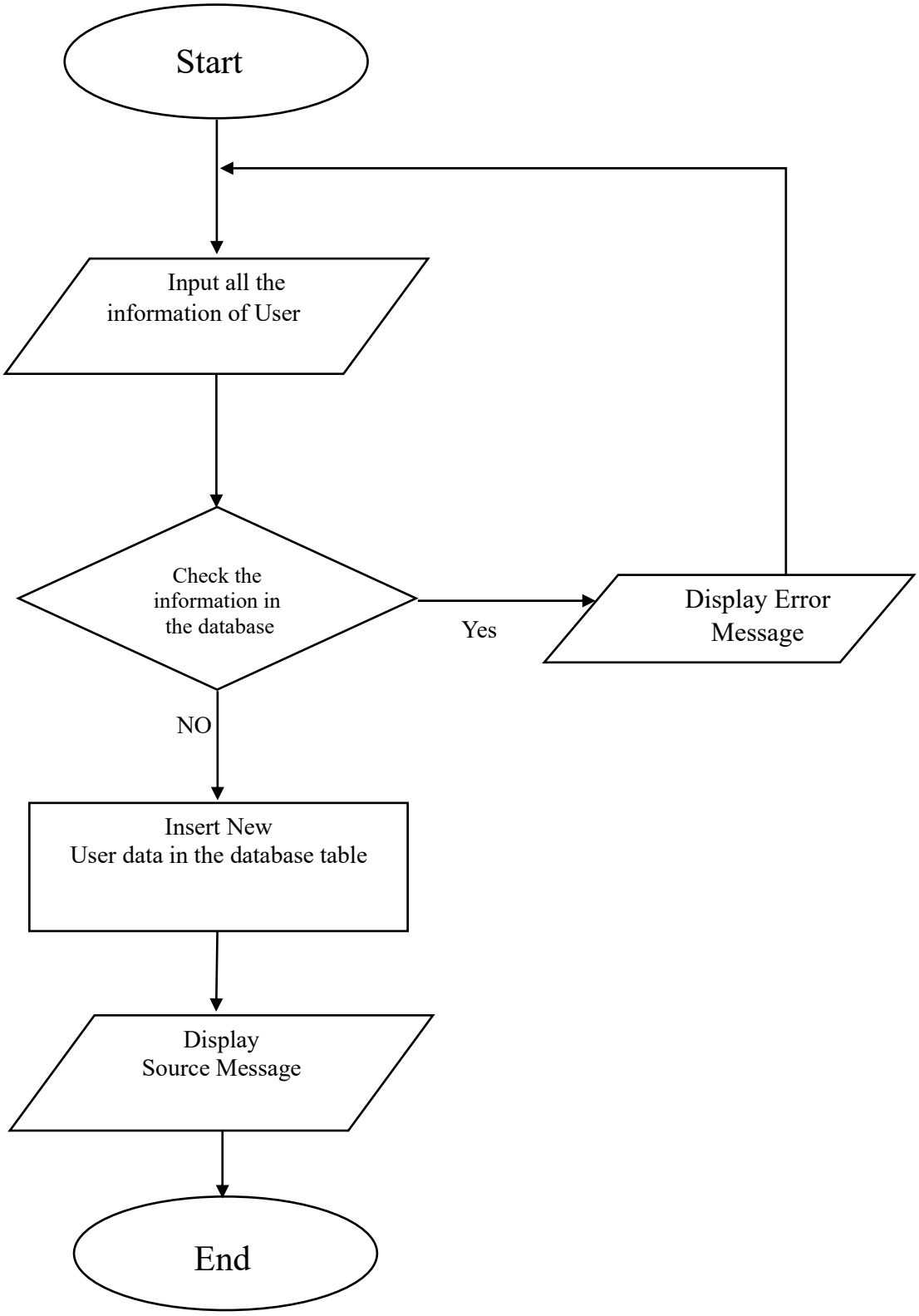


Fig 3.2: Flowchart Diagram of Register Form

5.3 Flowchart Diagram of Product Detail Form:

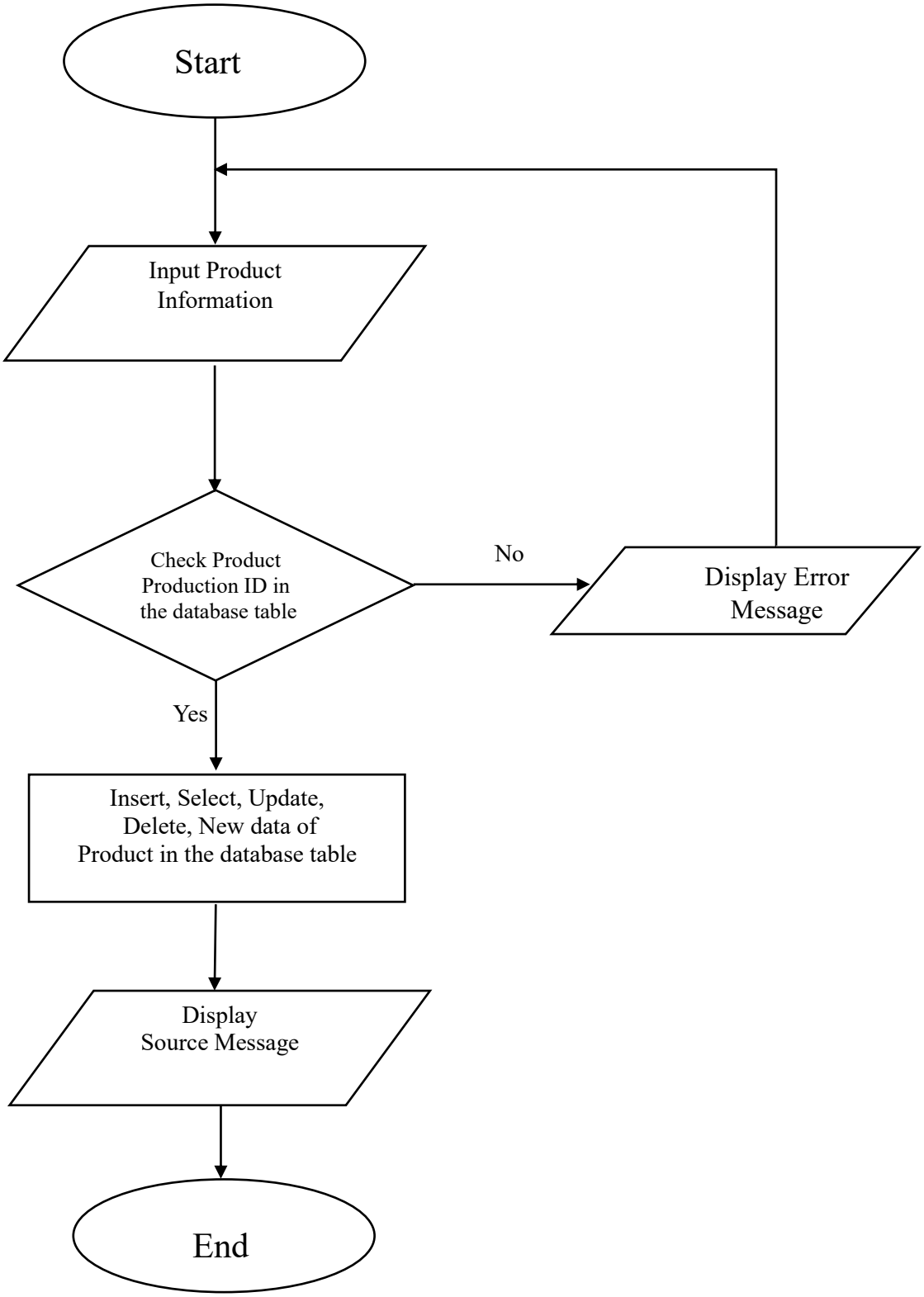


Fig 3.3: Flowchart Diagram of Product Detail Form

5.4 Flowchart Diagram of Category Detail Form:

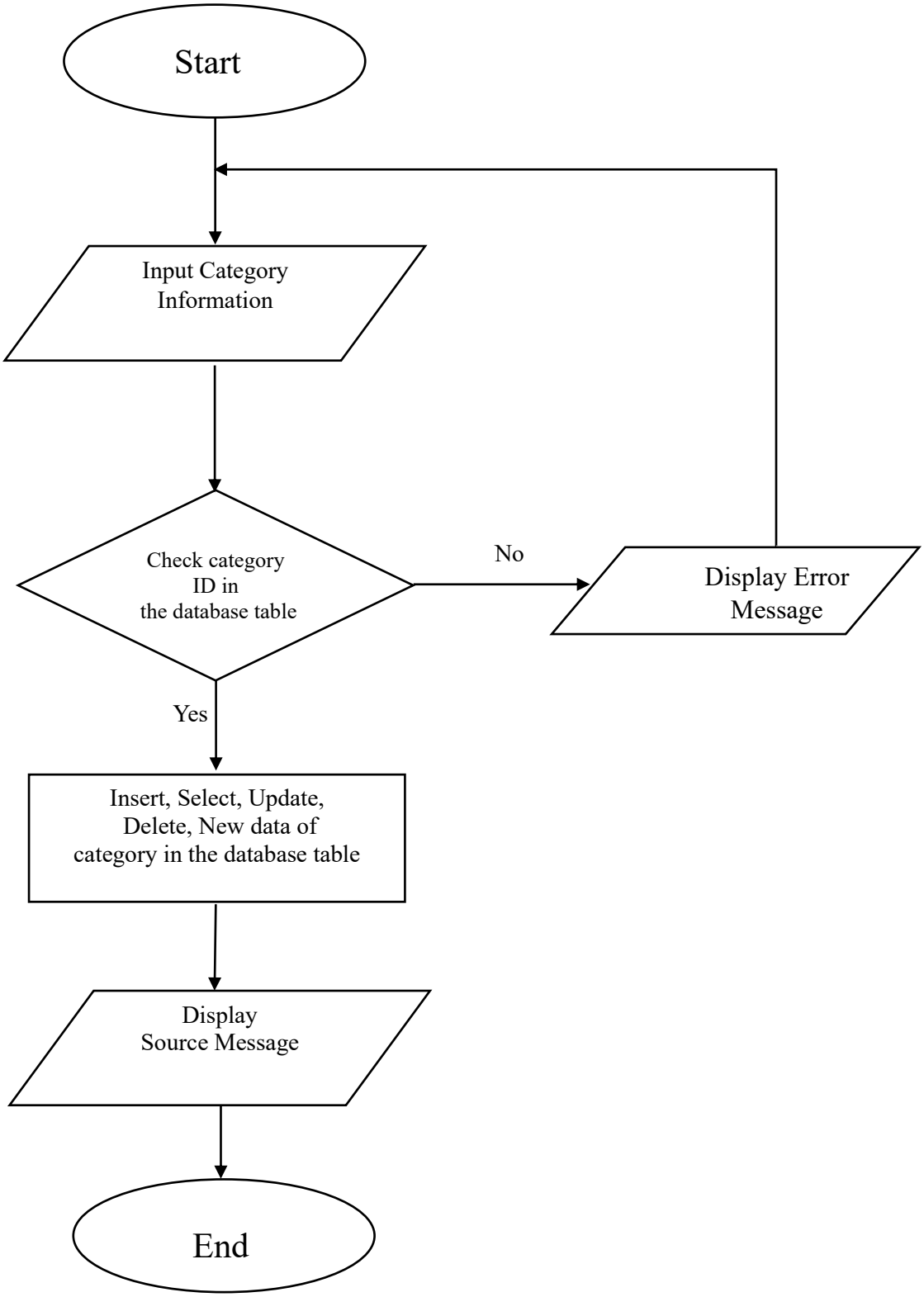


Fig 3.4: Flowchart Diagram of Category Detail Form

6. ER-DIAGRAM

An entity relationship diagram is a data modeling technique that graphically illustrate an information system entity and the relationships between those entities. An ER-Diagram and sensation model of data used to represent entity framework infrastructure.

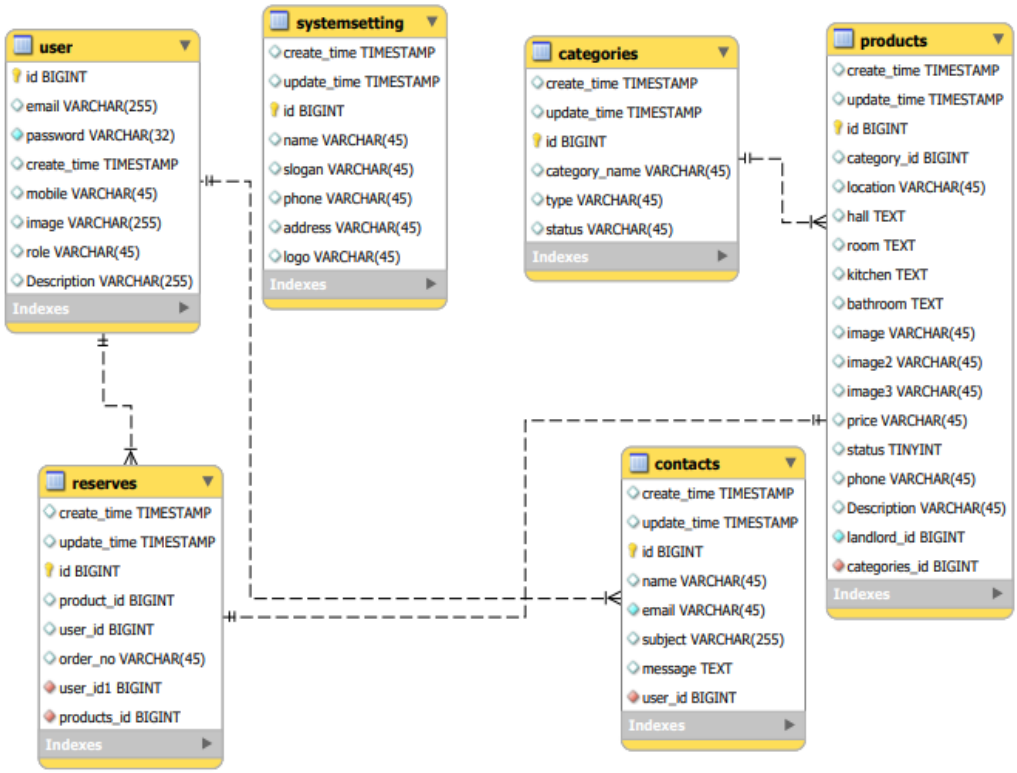
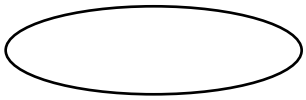
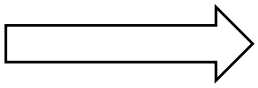


Fig 4: ER-Diagram of Room Rental Application

7. DFD

A data flow diagram (DFD) is a graphical representation of the flow of data through an information system. It is a tool for visualizing the movement of data through a system, and for identifying and understanding the relationships between different components of the system.

Geometric Figure	Names	Function
	Rectangle	A rectangle defines source or destination of the system. It is also called entity.
	Oval	It represents as a process that gives us information. It is also called processing box.
	Arrow	An arrow identifies the data flow i.e. it gives information to the data that is in motion.
	Open Rectangle	Data is store either temporary or permanently.

7.1 0-level DFD:

It is also known as a context diagram. It's designed to be an abstraction view, showing the system as a single process with its relationship to external entities. It represents the entire system as a single bubble with input and output data indicated by incoming/outgoing arrows.

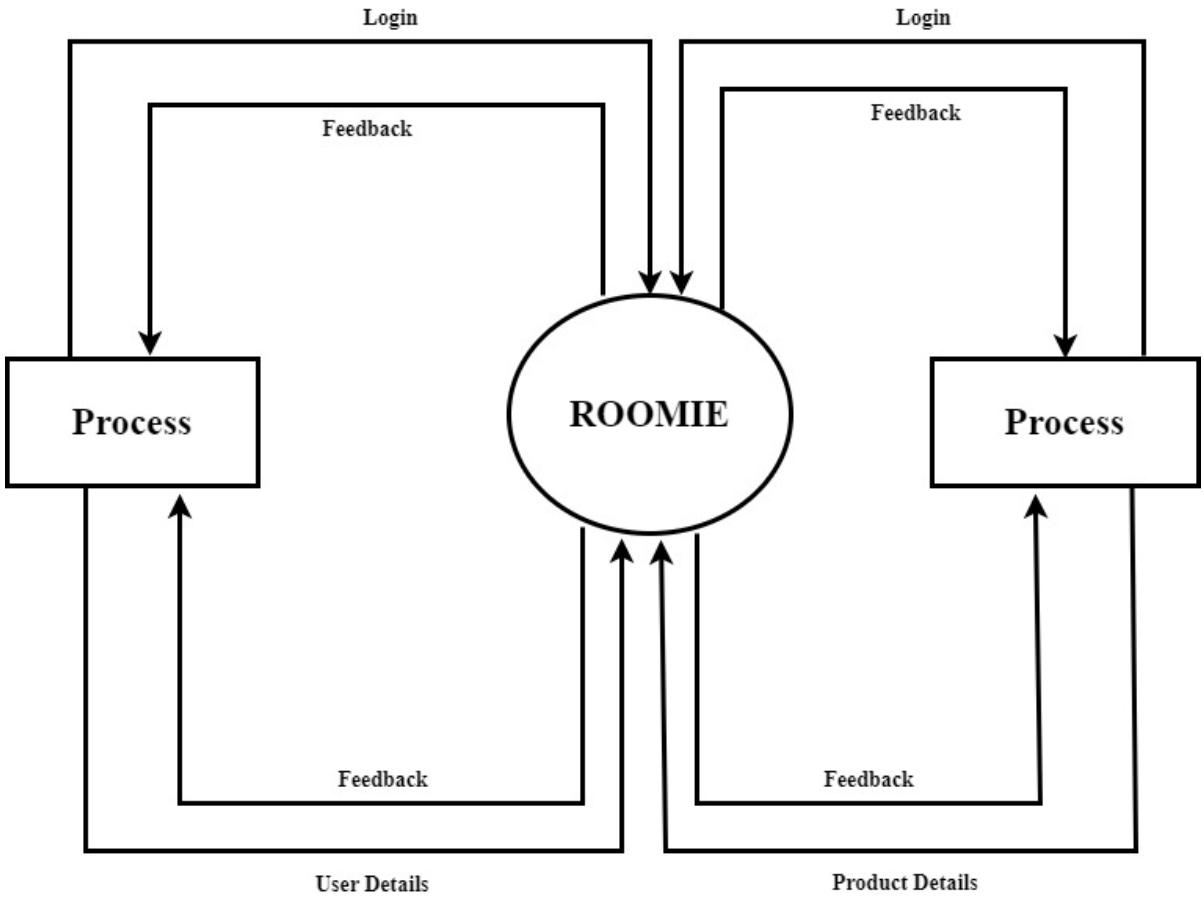


Fig 5.1: 0-level DFD

7.2 1-level DFD:

In 1-level DFD, the context diagram is decomposed into multiple bubbles/processes. In this level, we highlight the main functions of the system and breakdown the high-level process of 0-level DFD into subprocesses.

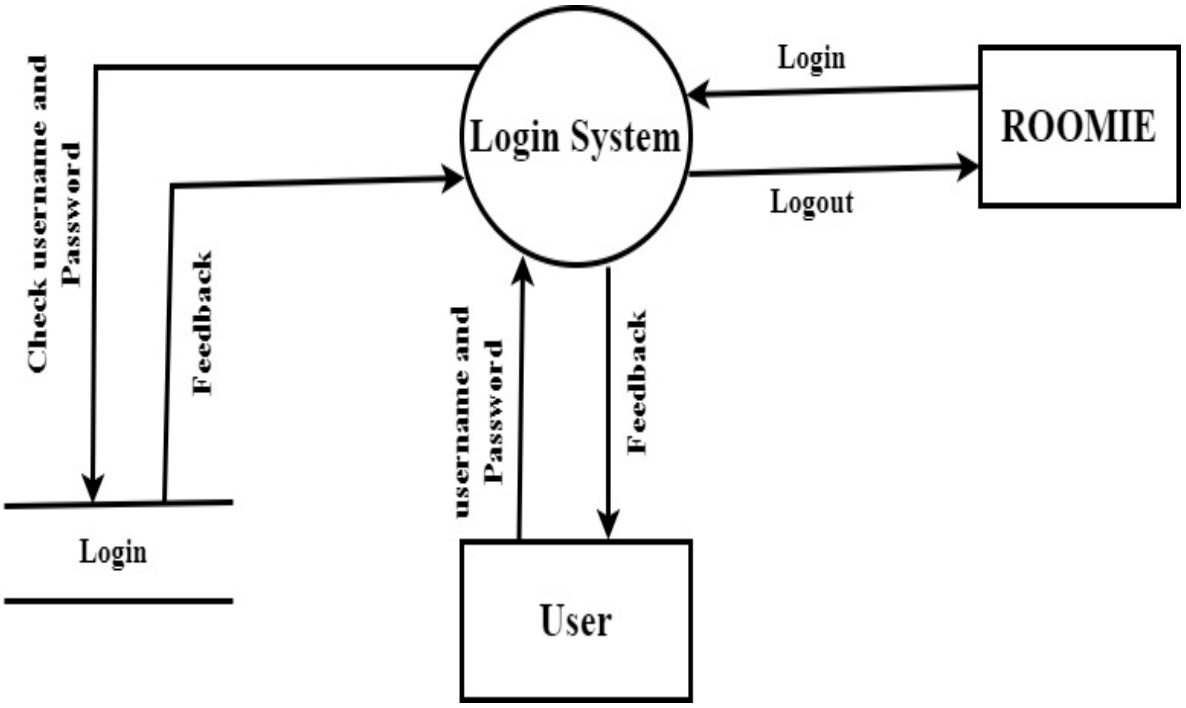


Fig 5.2: 1-level DFD

7.3 Dataflow Diagram of Admin Login System

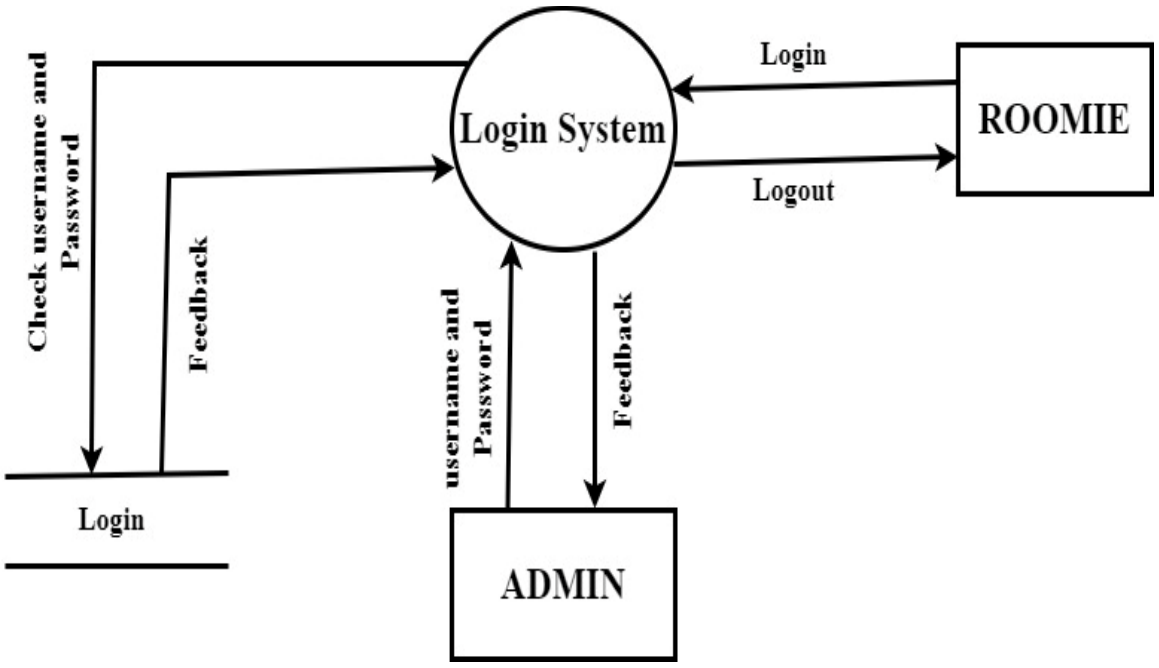


Fig 5.3: 1-level DFD

8.1 Home Page

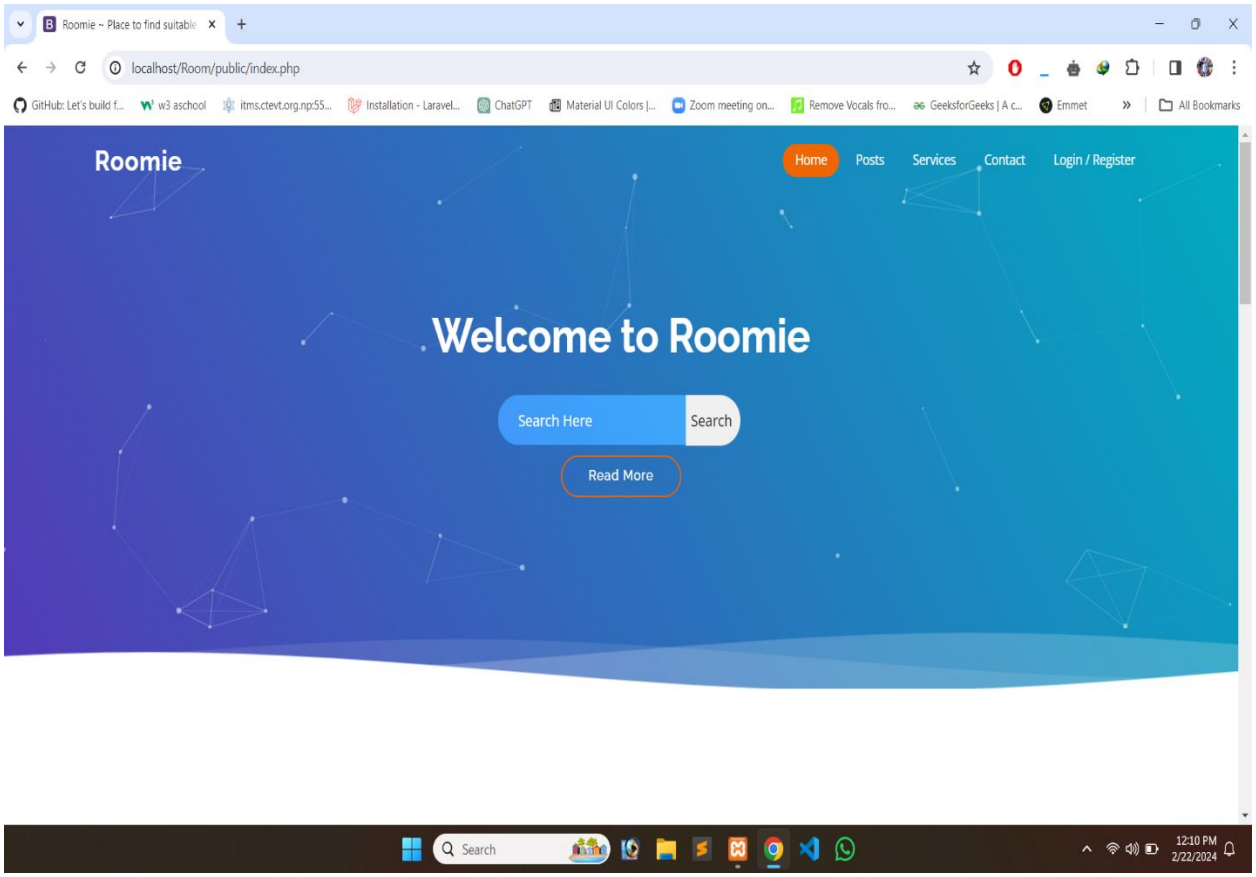


Fig 6.1: Home Page of Room Rental Application

8.2 Login Form

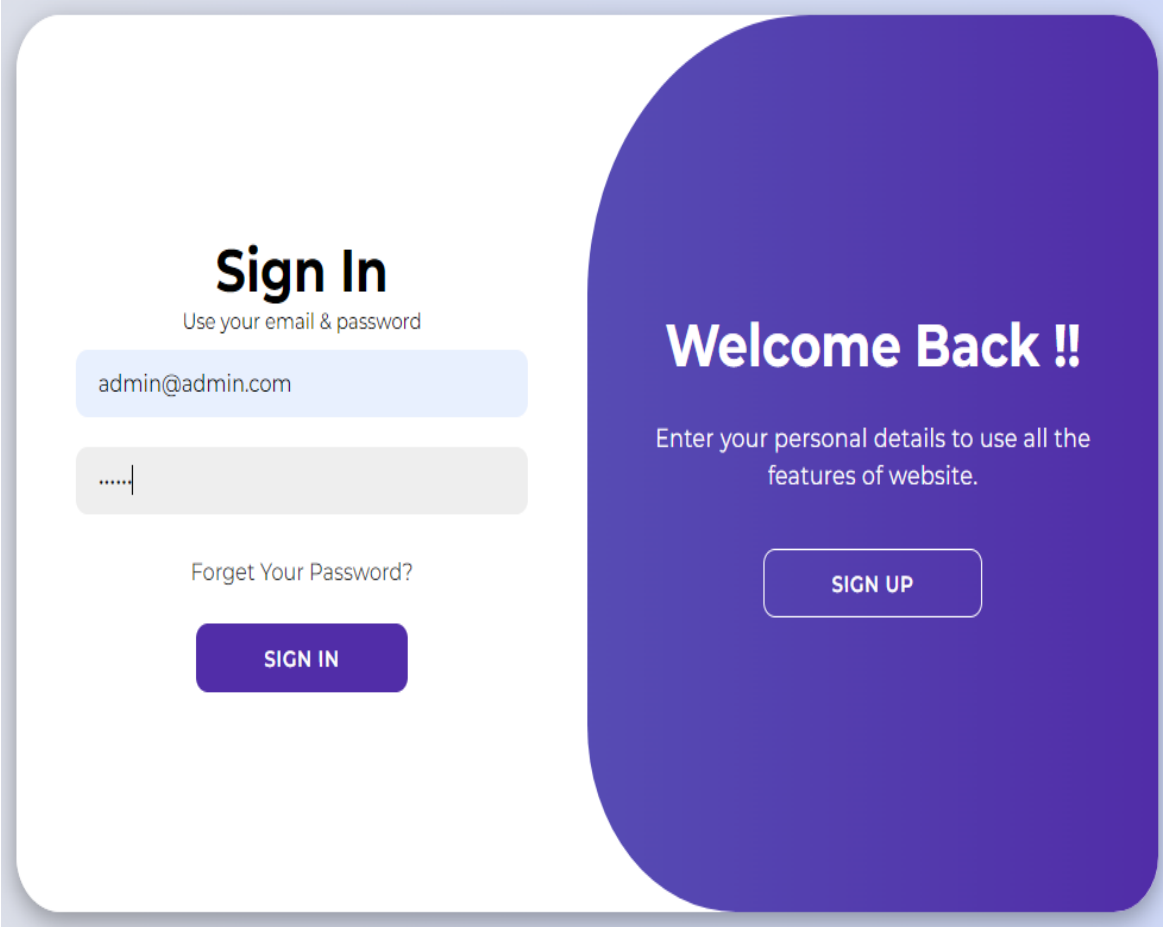


Fig 6.2: Login Form of Room Rental Application

8.3 Register Form

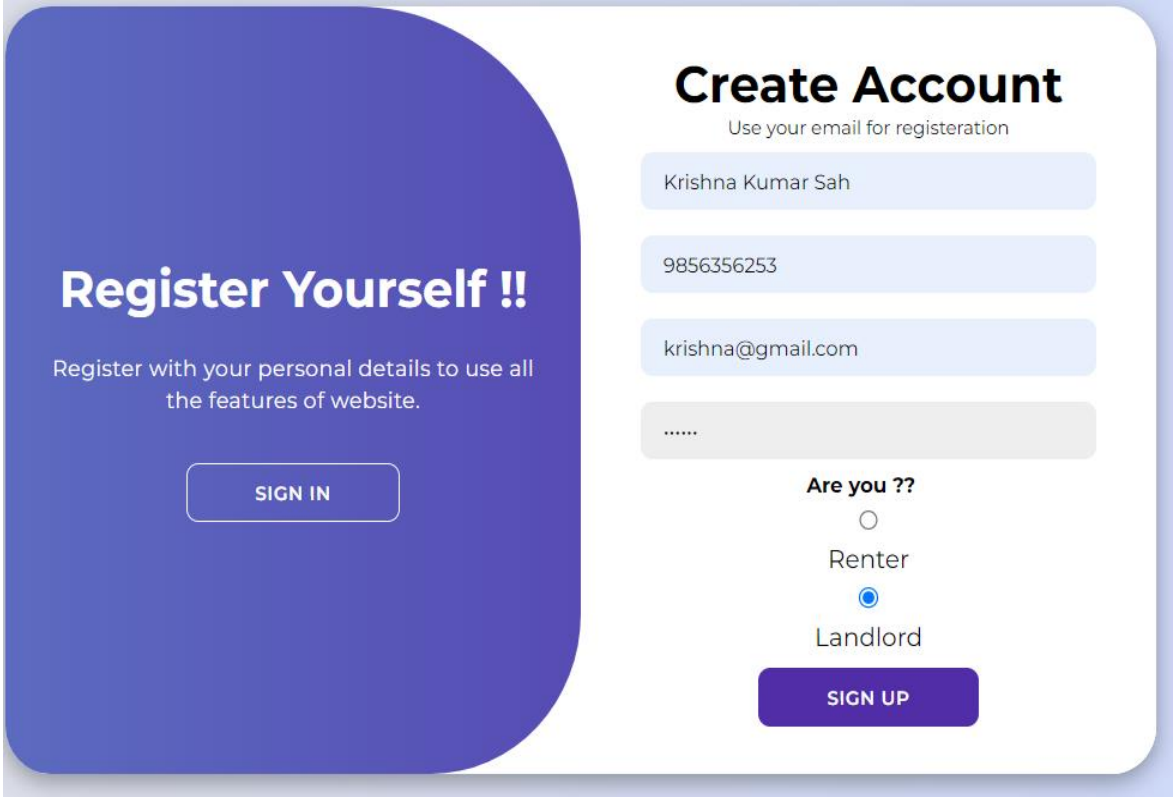


Fig 6.3: Register Form of Room Rental Application

8.4 Admin Dashboard

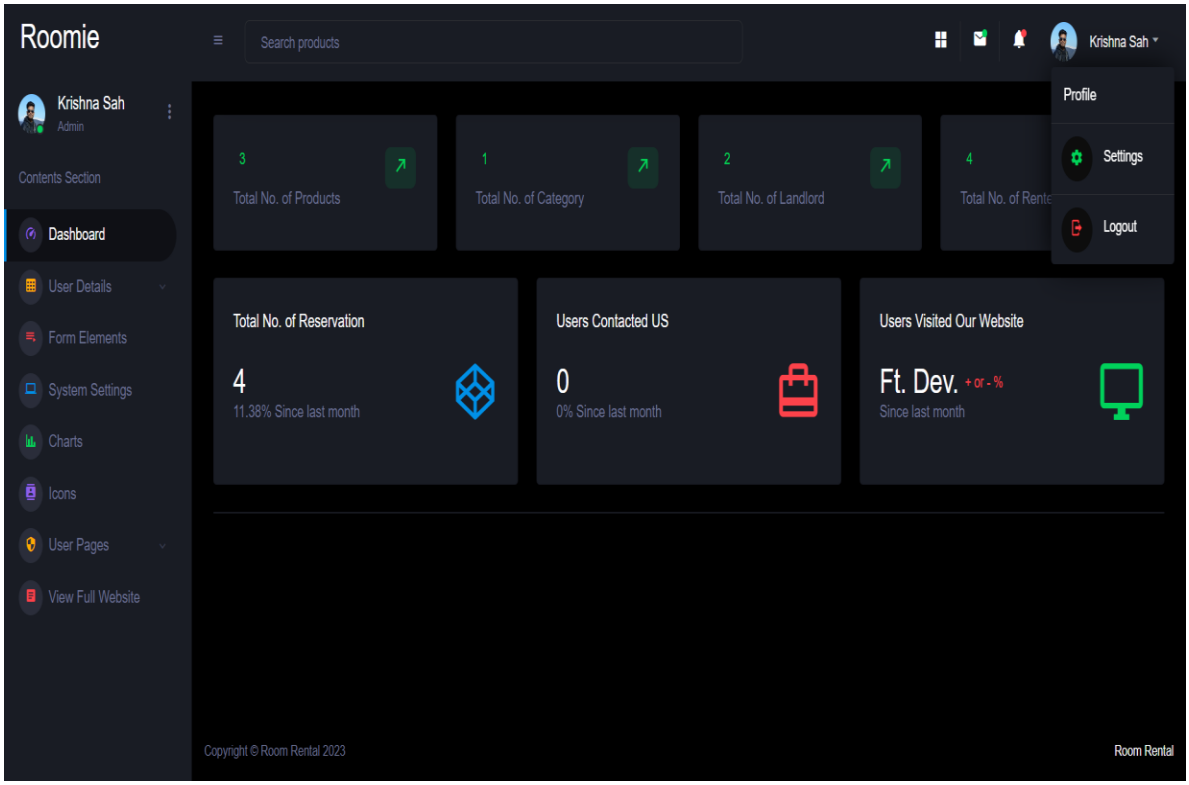


Fig 6.4: Admin Dashboard of Room Rental Application

8.5 Room Details

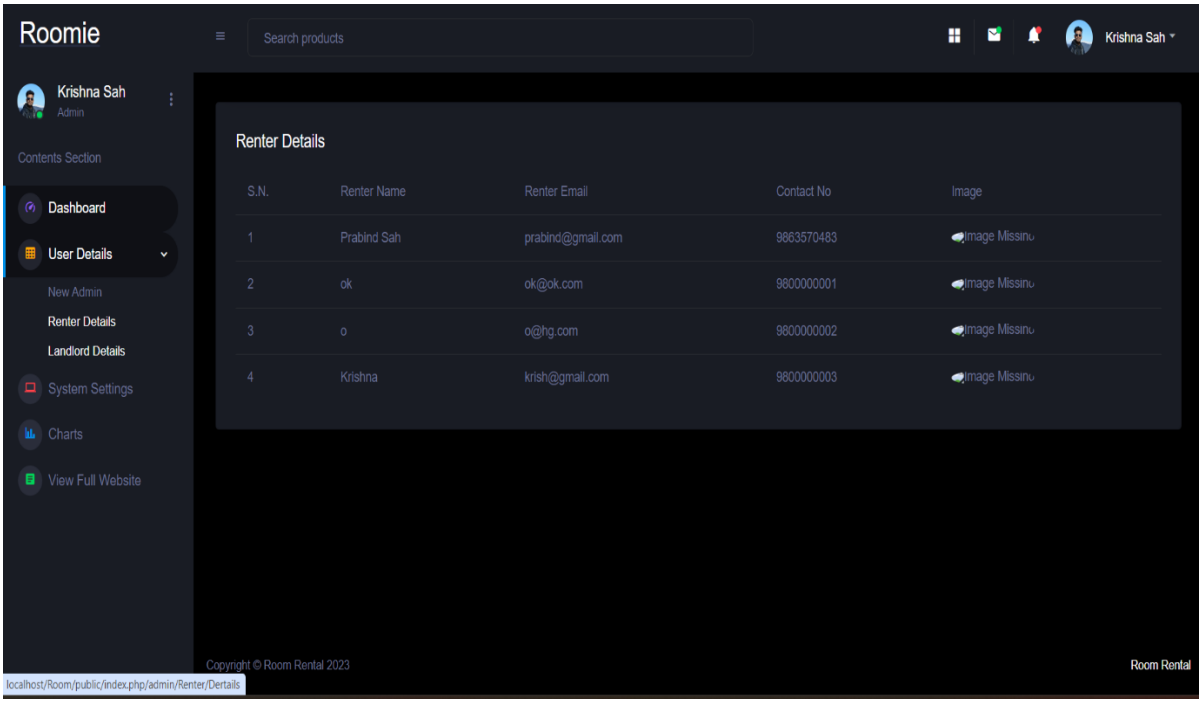


Fig 6.5: Room Details of Room Rental Application

8.6 Add Room Details Form

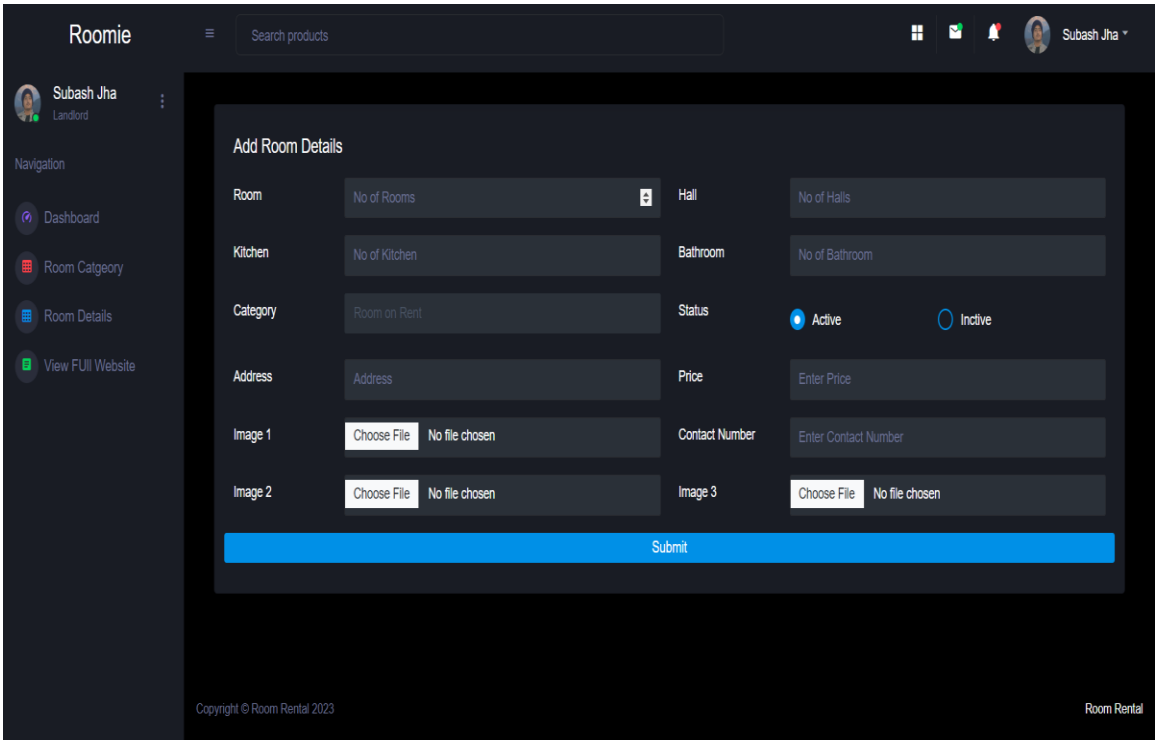


Fig 6.6: Add Room Details Form of Room Rental Application

8.7 Category Details

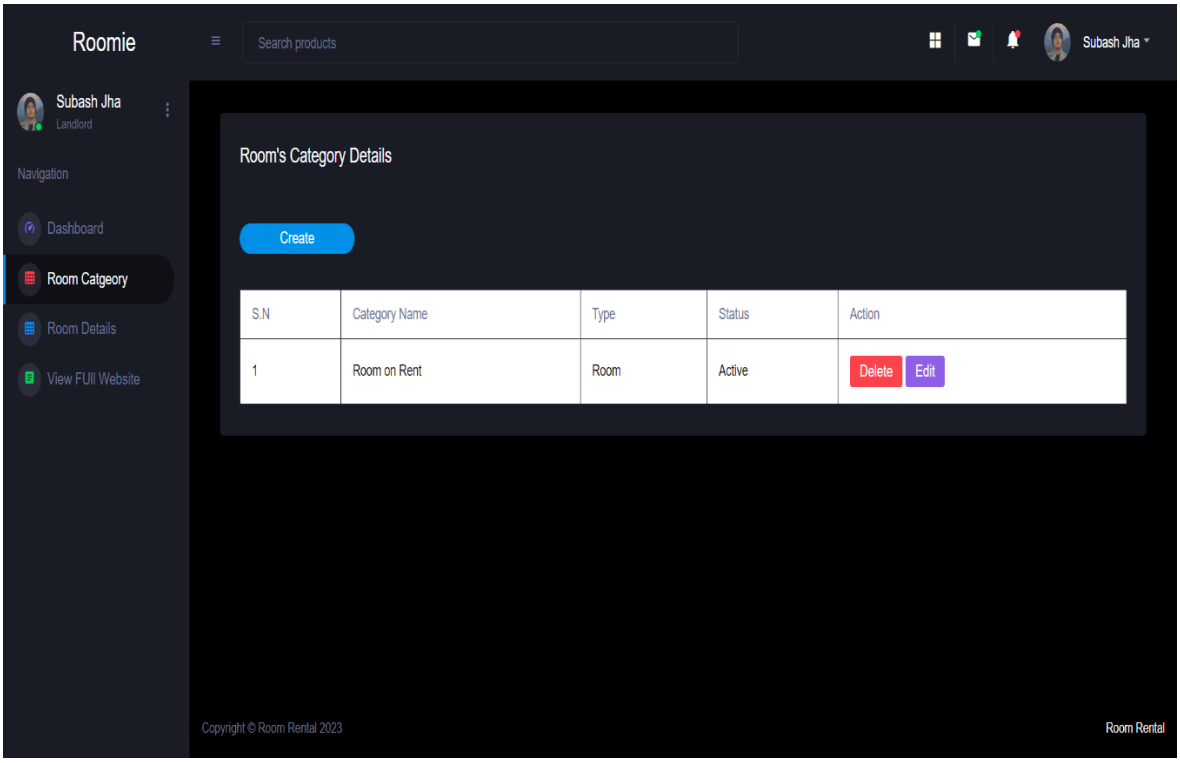


Fig 6.7: Category Details of Room Rental Application

8.8 Add Category Form

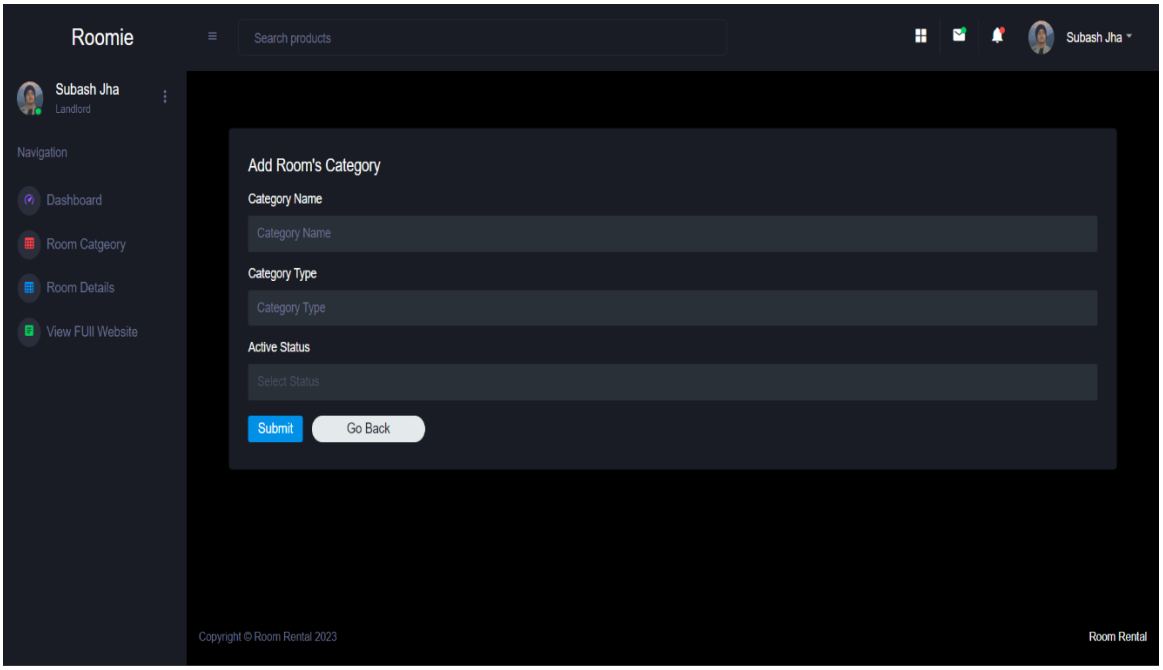


Fig 6.8: Add Category Form of Room Rental Application

8.9 Admin Setting for Updating System Setting

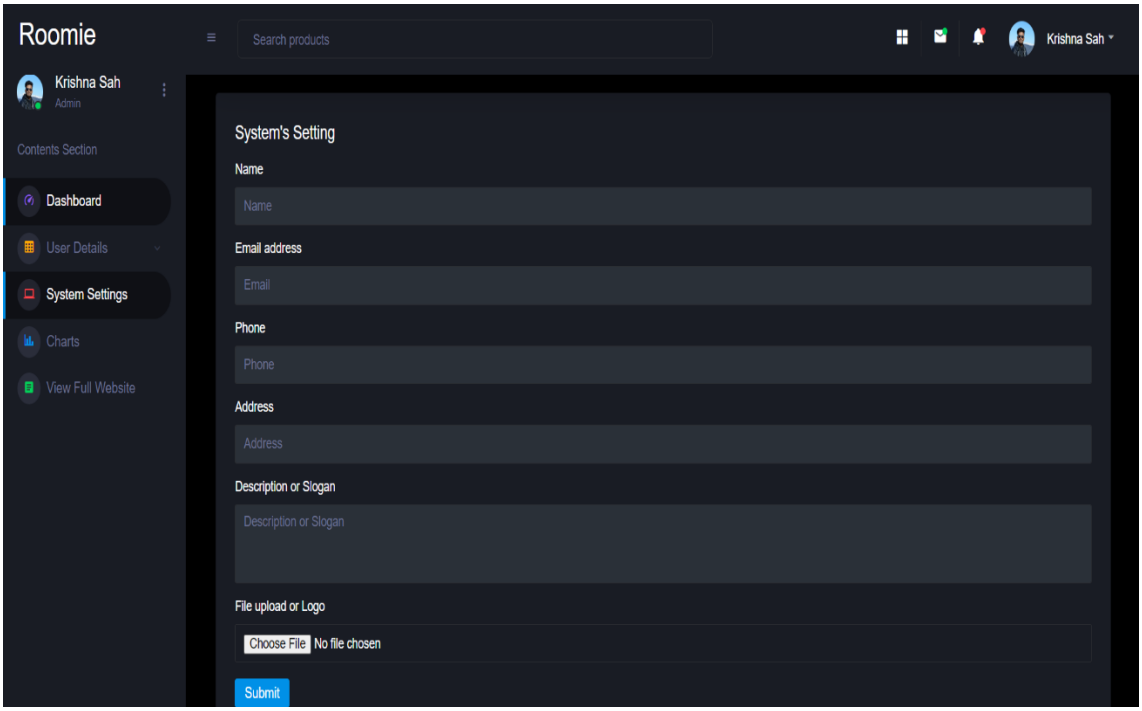


Fig 6.9: System Setting of Room Rental Application

8.10 New Admin Creation

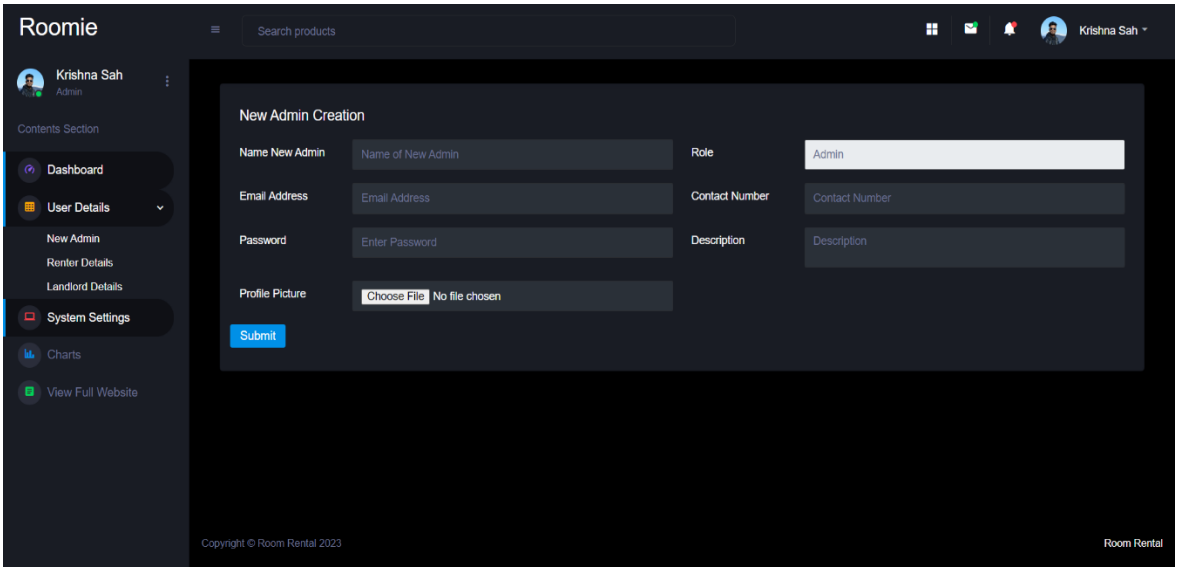


Fig 6.10: New Admin Creation of Room Rental Application

8.11 Renter Details

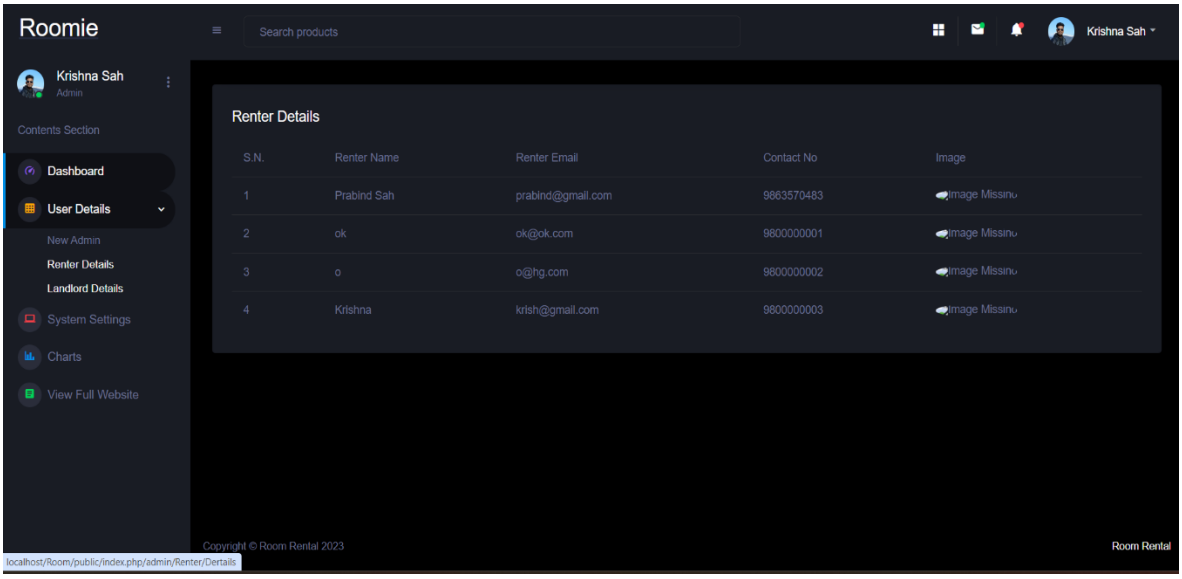


Fig 6.11: Renter Details of Room Rental Application

8.12 Room Reservation Details

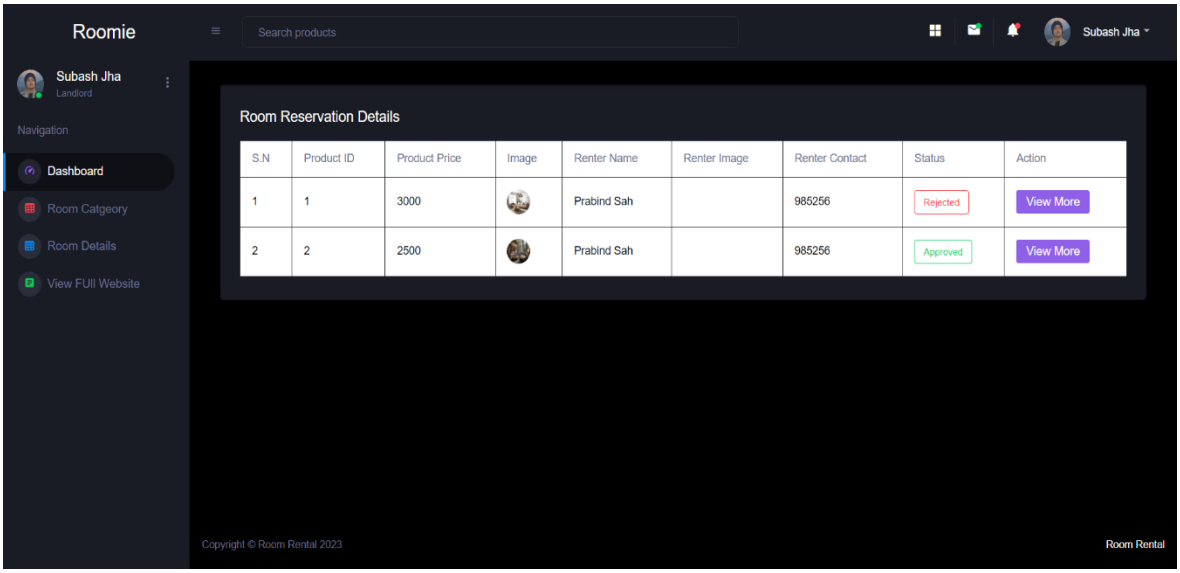


Fig 6.12: Room Reservation Details of Room Rental Application

8.13 Room Reservation Update

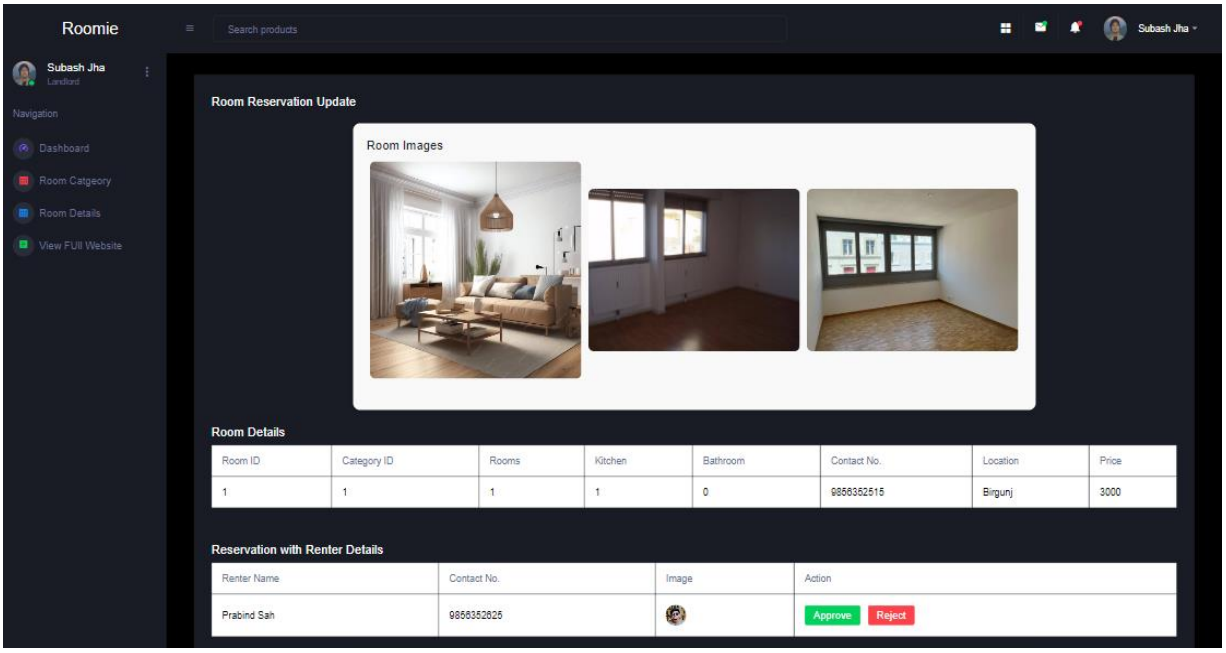


Fig 6.13: Room Reservation Update

8.14 Landlord Dashboard

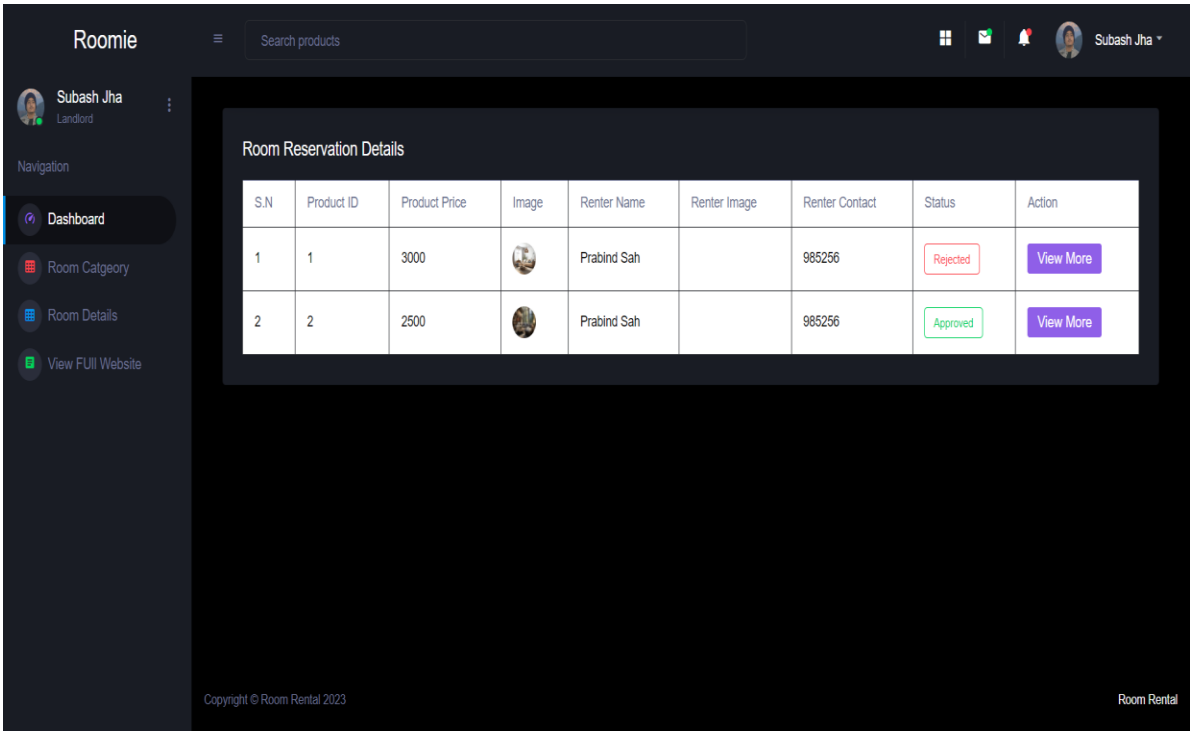


Fig 6.14: Admin Dashboard of Room Rental Application

8.15 Renter Dashboard

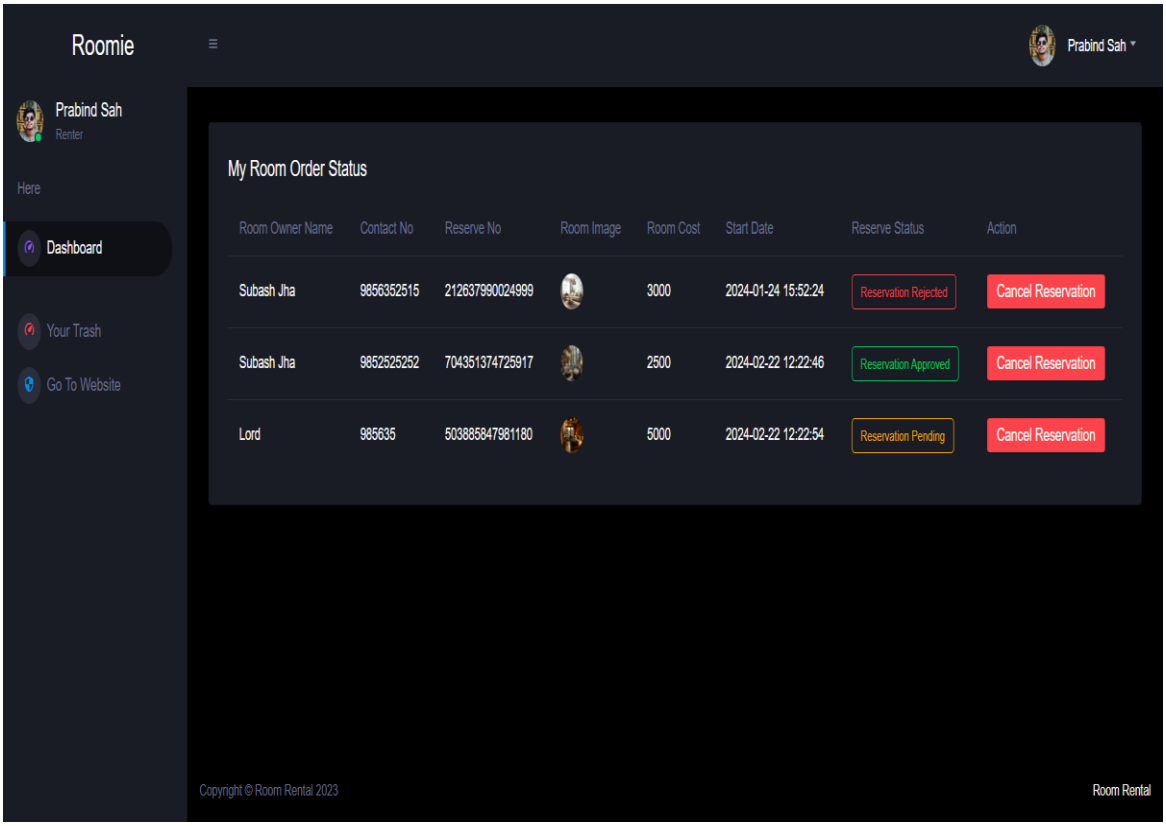


Fig 6.15: Admin Dashboard of Room Rental Application

9.1 Login Code

```
{
    public function login(Request $request){

        $request->validate([
            'email' => 'required | email',
            'password' => 'required | min:6',
        ]);
        $user = User::where('email',$request->email)->first(); //finding
user
        if($user){
            if($user->password){
                if(Hash::check($request->password, $user->password)){
                    Auth::login($user);
                      $userRoles = Auth::user()->role;
                      $userid = Auth::user()->id;
                      $request = session()->put('id',Auth::user()->id);
                    switch ($userRoles) {
                        case 'Admin':

                            return redirect()->route('admin.dashboard');
                            break;

                        case 'Landlord':

                            return redirect()->route('landlord.dashboard');
                            break;

                        case 'Renter':
                            return redirect()->route('renter.dashboard');
                            break;

                        default:
                            return redirect()->back();
                    }
                }
                $request->session()->flash('error','Incorrect Password');
                return redirect()->back();
            }
        }
        $request->session()->flash('error','User Not Found');
        return redirect()->back();
    }
```

Fig 7.1: Login Code of Room Rental Application

9.2 Register Code

```
public function register(Request $request)
   {

       // Validate the form data
       $request->validate([
           'name' => 'required|string',
           'email' => 'required|email|unique:users',
           'mobile' => 'required|max:14',
           'password' => 'required|min:6',
           'role' => 'required|in:Renter,Landlord',
       ]);

       // Create a new user
     $data = [
           'name' => $request->name,
           'email' => $request->email,
           'mobile' => $request->mobile,
           'password' => bcrypt($request->password),
           'role' => $request->role,
           ];

        User::insert($data);

       return redirect()->route('login.page')->with('success', 'Registration successful. Please log
in.');
   }
}
```

Fig 7.2: Register Code of Room Rental Application

9.3 Code for Create Room Details

```
public function create(Request $request){
        $request->validate([
                        'location' => 'required',
                        'price' =>'required',
                        'status' => 'required',
                        'room' => 'required',
                        'category_id' => 'required',
                        'phone' => 'required',
                    ]);

                     $images = array();

                    if($files = $request->file('image') && is_array($request->file('image'))){

                        for ($i = 0; $i <= 2; $i++) {
                            $imageKey = 'image' . $i;
                            if ($request->file('image')[$i]) {
                                $file =  $request->file('image')[$i];
                                $newName = time().'_'. rand(10,9999999999999).'_'.$file-
>getClientOriginalName();
                                $newPath = public_path()."/Room_Images/";
                                $file->move($newPath, $newName);
                                $images[$imageKey] = $newName;
                            }
                        }

                }
                    $data = [
                    'landlord_id' => session()->get('id'), // Save the user ID in the data array
                     'category_id' =>$request->category_id,
                     'location' =>$request->location,
                     'price' =>$request->price,
                     'room' =>$request->room,
                     'hall' =>$request->hall,
                     'kitchen' =>$request->kitchen,
                     'bathroom' =>$request->bathroom,
                     'phone' =>$request->phone,
                     'status' =>$request->status,
                     'image' =>$images['image0'] ?? "",
                     'image2'=>$images['image1'] ?? "",
                     'image3'=>$images['image2'] ?? ""
                 ];

                Product::insert($data);
                     return redirect()->route('Room.Details');

    }
```

Fig 7.3: Code for Creating Room Details in Room Rental Application

9.4 Code for Delete Room

```
public function roomdelete($id){
        if(!$id){
            return redirect()->route('Room.Details');
        }
        try{
            $product = Product::find($id);
            $product->delete();
            return redirect()->route('Room.Details');
        }catch(\Exception $e){
          return redirect()->route('Room.Details');
        }
    }

    public function ProductDetails($id){
        if(!$id){
            return redirect()->back();
        }

        $product = Product::find($id);

        if($product){
            return view('frontend.showdetails',
compact(}product'));

        if(!$id){
            session()->flash('error', 'Product NOt found !!');
            return redirect()->back();
        }
    }
```

Fig 7.4: Code for Deleting Room in Room Rental Application

9.5 Code for Update Category

```
public function catupdate(Request $request,$id){
        if(!$id){
            return redirect()->route('Room.Category');
          }
          $data['category'] = Category:: find($id);

          if($data['category']){

            $request->validate([
                'category_name' =>'required',
                'type'=>'required ',
                'status'=>'required'
            ]);

            $updatecategory_data =[
                'category_name' => $request-
>get('category_name'),
                'type' => $request->type,
                'status' => $request->status ?? 1,
            ];
            $data['category']->update($updatecategory_data);
            return redirect()->route('Room.Category');
          }
          return redirect()->route('Room.Category');
    }
```

Fig 7.5: Code for Update Category in Room Rental Application

9.6 Code for Search Room

```
public function search(Request $request){

        $searchTerm = $request->search;
        $query = Product::query();
        $data['systems'] = Systemsetting::find(1);
        $_SESSION['setting'] = $data['systems'];
        if($searchTerm){
            $query-> where('location','LIKE','%'.$searchTerm.'%')
                    ->orWhere('price','LIKE','%'.$searchTerm.'%')
                    ->orWhere('kitchen','LIKE','%'.$searchTerm.'%')
                    ->orWhere('hall','LIKE','%'.$searchTerm.'%')
                    -
>orWhere('Description','LIKE','%'.$searchTerm.'%')->get();

                    return view('frontend.result',$data);

        }else{
            return redirect() -> back();

        }
```

Fig 7.6: Code for Search Room in Room Rental Application

9.7 Code for Reservation of Room

```
public function reserve(Request $request,$id){
        if($ids = $user = Auth::id()){
            $role = User::find($user);
            if($role->role == 'Renter'){
                $product = Product::find($id);
                $P_id = $product->id;
                $landlordid = $product->landlord_id;
                $orderNo = rand(10,999999999999999);

                $data =[
                    'order_no' => $orderNo,
                    'product_id'=>$P_id,
                    'landlord_id'=>$landlordid,
                    'user_id'=>Auth::id()
                ];

                  Reserve::insert($data);
                  Mail::to(\auth()->user()->email)->send(new ReserveMail($orderNo,
$product));     return redirect()->route('renter.dashboard');

            }else{
                return redirect()->back();
            }

        }else{
            return redirect('login.page');
        }
    }
```

Fig 7.7: Code for Reservation of Room in Room Rental Application

9.8 Code for Delete Reservation

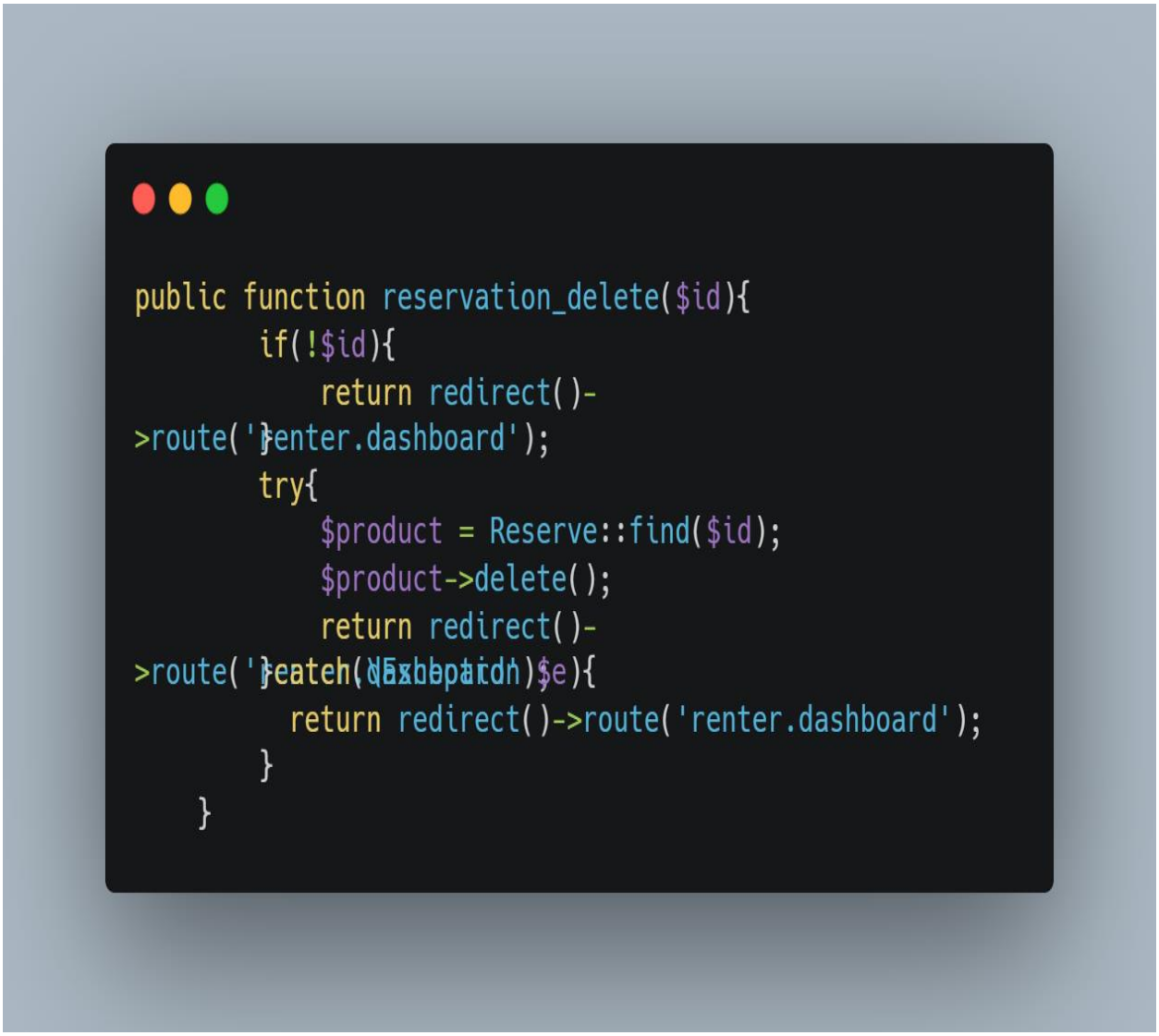


Fig 7.8: Code for Delete Reservation in Room Rental Application

10. System Testing

- System testing for our Room Rental Application involves evaluating the integrated system to ensure that it meets the specified requirements and functions as intended. Here is an overview of key aspects that are considered in the system testing phase:

I. **Functional Testing:** Verify that all functional requirements are implemented correctly. Test user registration and authentication processes. Ensure the virtual tour feature works seamlessly.

II. **User Interface (UI) Testing:** Evaluate the user interface for usability, responsiveness, and adherence to design specifications. Verify that buttons, links, and interactive elements function as expected.

III. **Performance Testing:** Assess the performance of the application under various conditions. Measure response times for key functionalities.

IV. **Security Testing:** Identify and address security vulnerabilities to protect user data. Verify secure communication, especially during payment transactions.

V. **Compatibility Testing:** Confirm that the Room Rental Application works correctly across different browsers, devices, and operating systems. Verify compatibility with various devices, including desktops, laptops, tablets, and smartphones.

VI. **Database Testing:** Ensure the integrity and reliability of the database. Test data retrieval and storage operations. Verify that database queries are optimized for efficiency. Check the handling of concurrent transactions and data consistency.

11. System Maintenance

System maintenance is a crucial aspect of ensuring the ongoing functionality, security, and usability of the Roomie project. It involves monitoring, updating, and enhancing the system to meet changing requirements and address issues that may arise over time.

1. **Monitoring and Incident Response:**
    - Implement continuous monitoring to detect and address any system anomalies or security incidents promptly.
    - Establish an incident response plan to efficiently handle and resolve issues as they arise.
2. **Regular Software Updates:**
    - Keep the application's software stack, including operating systems, frameworks, and third-party libraries, up to date to address security vulnerabilities and benefit from the latest features and optimizations.
3. **Database Maintenance:**
    - Perform routine database maintenance tasks, such as optimizing queries, updating indexes, and managing data backups to ensure data integrity and efficient performance.
4. **User Feedback and Support:**
    - Establish channels for users to provide feedback on their experiences with the application.
    - Address user-reported issues promptly and provide ongoing support to maintain high user satisfaction.
5. **Scalability and Performance Optimization:**
    - Monitor system performance and scalability to anticipate and address any bottlenecks as user traffic grows.
    - Optimize code and database queries to enhance overall system performance.
6. **Regulatory Compliance:**
    - Stay informed about changes in relevant legal and regulatory requirements, ensuring ongoing compliance with data protection laws and other industry-specific regulations.
7. **Backup and Disaster Recovery:**
    - Regularly test backup and disaster recovery procedures to ensure the ability to recover data and restore operations in the event of system failures or data loss.

12. CONCLUSION:

The conclusion of the Roomie project marks the culmination of extensive planning, development, and testing efforts aimed at creating a Room Rental Application that addresses the diverse needs of users in search of suitable accommodations. This concluding phase encapsulates the achievements, highlights the key features, and reinforces the project's overarching goals.

The project has successfully adopted a user-centric approach, ensuring that the Roomie application is not only functional but also intuitive and responsive to the needs of its users. The incorporation of innovative features, such as the smart matching algorithm and immersive virtual tours, demonstrates a commitment to leveraging cutting-edge technology to enhance the overall user experience.

Transparent pricing, clear rental policies, and a robust verification process contribute to building trust among users, fostering a reliable and transparent housing marketplace. Roomie goes beyond a transactional platform, fostering community engagement through forums and networking features. This creates a collaborative environment and a sense of belonging among users.

In conclusion, the Roomie project not only represents the successful development of a Room Rental Application but also embodies a commitment to excellence, user satisfaction, and continuous improvement. As Roomie is introduced to the market, it is poised to make a meaningful contribution to the housing ecosystem by simplifying the search for accommodations, fostering community connections, and setting new standards for transparency and innovation in the room rental space. Welcome to the future of housing solutions—welcome to Roomie!

13. REFERENCES:

- **www.google.com**
- **www.youtube.com/**
- **www.github.com/**
- **https://chat.openai.com/**
- **https://stackoverflow.com/**
- **https://codepen.io/**
- **https://carbon.now.sh/**
- **Old Projects**