HOSTEL MANAGEMENT SYSTEM

A PROJECT REPORT

Submitted by

SANTHOSH SHARMA M	**2303811710421136**
SARAVANAPRIYAN M	**2303811710421138**
SEENIVASAN S	**2303811710421139**
SIBI S	**2303811710421149**

in partial fulfillment of the requirements for the award degree of

Bachelor in Engineering

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY

(AUTONOMOUS)

SAMAYAPURAM - 621112

DECEMBER - 2025

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY
(AUTONOMOUS)
SAMAYAPURAM – 621112

BONAFIDE CERTIFICATE

The work embodied in the present project report entitled "**HOSTEL MANAGEMENT SYSTEM**" has been carried out by the students **SANTHOSH SHARAMA M, SARAVANAPRIYAN M, SEENIVASAN S, SIBI S.** The work reported here in is original and we declare that the project is their own work, except where specifically acknowledged, and has not been copied from other sources or been previously submitted for assessment.

Date of Viva Voce: ……

Mrs. V. KALPANA, M.E., (Ph.D.,)
SUPERVISOR
Assistant Professor
Department of CSE
K. Ramakrishnan College of Technology (Autonomous)
Samayapuram – 621 112.

Mr. R. RAJAVARMAN M.E., (Ph.D.,)
HEAD OF THE DEPARTMENT
Assistant Professor (Sr. Grade)
Department of CSE
K. Ramakrishnan College of Technology (Autonomous)
Samayapuram – 621 112.

INTERNAL EXAMINER **EXTERNAL EXAMNIER**

ABSTRACT

The Hostel Management System is a database-driven application designed to automate and streamline the management of hostel operations. This system uses SQL as the core technology for efficient data storage, retrieval, and management. It handles essential hostel activities such as student registration, room allocation, fee management, attendance tracking, and complaint handling. By maintaining structured records of students, rooms, staff, payments, and facilities in a centralized SQL database, the system reduces manual paperwork, minimizes errors, and improves data security. With the help of this system, hostel staff can easily assign rooms, keep track of student records, manage payments, and handle complaints. The system saves time, improves accuracy, and provides a better way to manage hostel operations smoothly and efficiently. The system provides quick access to real-time information, enabling hostel administrators to make effective decisions and maintain smooth operations. Overall, the Hostel Management System enhances the efficiency, transparency, and organization of hostel management, making it a reliable and user-friendly solution for educational institutions.

Keywords:

Hostel Management System, Database Management System (DBMS), Student Information, Room Allocation, Admission Management, Fee Management, SQL, User Authentication, Admin Module, Record Maintenance, Automation, Data Security, Report Generation, Software Engineering, Information System, Web/Application Development

ACKNOWLEDGEMENT

We thank our **Dr. N. VASUDEVAN**, Principal, for his valuable suggestions and support during the course of my research work.

We thank our **Mr. R. RAJAVARMAN**, Head of the Department, Assistant Professor (Sr. Grade), Department of Computer Science and Engineering, for his valuable suggestions and support during the course of my research work.

We wish to record my deep sense of gratitude and profound thanks to my Guide **Mrs. V. KALPANA,** Assistant Professor , Department of Computer Science and Engineering**,** for her keen interest, inspiring guidance, constant encouragement with my work during all stages, to bring this thesis into fruition.

We are extremely indebted to our project coordinator **Mrs. V. KALPANA,** Assistant Professor, Department of Computer Science and Engineering, for her valuable suggestions and support during the course of my research work.

We also thank the faculty and non-teaching staff members of the Department of Computer Science And Engineering, K .Ramakrishnan College Of Technology, Samayapuram, for their valuable support throughout the course of my research work.

Finally, we thank our parents, friends and our well wishes for their kind support.

SIGNATURE

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

SRS	–	System Requirement Specification
UML	–	Unified Modeling Language
UX	–	User Experience
API	–	Application Programming Interface
OTP	–	One-Time Password
PNG	–	Portable Network Graphics
SQL	–	Structured Query Language

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION ABOUT DOMAIN

The Hostel Management System domain focuses on the effective administration of hostel facilities within educational institutions such as schools, colleges, and universities. Hostels accommodate a large number of students and require systematic management of rooms, students, staff, and facilities. This domain involves handling daily operational tasks such as student registration, room allotment, fee tracking, attendance monitoring, maintenance, and complaint handling.With the growth in student populations, traditional manual systems are no longer efficient. Digital hostel management systems provide a structured way to store and access information, helping administrators maintain accuracy, security, and transparency. This domain plays a key role in improving the quality of hostel services and ensuring student comfort and safety.

1.2PROBLEM DESCRIPTION

Many hostels still rely on paper-based systems and manual record-keeping, which creates several challenges. These include delays in accessing information, difficulty in tracking room availability, errors in fee records, and misplaced documents. Managing large volumes of student data manually can lead to inconsistencies, redundancy, and inefficiency. Another major problem is the lack of proper data security and backup in manual systems. Unauthorized access to records, loss of important data due to accidents, and difficulty in generating reports are common issues. Communication between students and hostel administration is also slow, especially when handling complaints and maintenance requests. These problems highlight the need for an automated and reliable hostel management solution. hostel management systems mostly rely on manual processes such as paper registers, files, and spreadsheets. These methods are slow, inefficient, and highly prone to human errors. Finding student records, tracking room availability, managing fee payments, and handling complaints becomes difficult and time-consuming. Manual systems also increase the chances of data duplication, data loss, and lack of proper security. With an increasing number of students, it becomes challenging for administrators to manage hostel operations accurately and efficiently

1.3 OBJECTIVE OF THE PROJECT

The main objective of this project is to design and develop an automated Hostel Management System that improves the efficiency and reliability of hostel operations. The system aims to simplify the processes of student registration, room allocation, and fee management. It seeks to provide a user-friendly interface that can be easily used by hostel staff with minimum training.Additional objectives include reducing manual workload, minimizing errors, ensuring secure data storage, and enabling fast retrieval of information. The project also aims to generate accurate reports on student occupancy, fee status, and room availability, which will help administrators in decision-making. Overall, the system focuses on improving service quality and operational transparency.

.

1.3.1. Reduce Waiting Time:

The Hostel Management System helps reduce waiting time by automating tasks that are usually done manually, such as student registration, room allocation, and fee verification. Since all records are stored in a centralized digital database, administrators can quickly retrieve information without searching through files or registers. This speeds up the overall process and reduces delays for students and staff.

1.3.2. Improve Service Efficiency:

The system improves service efficiency by simplifying and organizing hostel operations. Tasks like updating student details, tracking room availability, and managing complaints can be done quickly and accurately. The automated system reduces human errors, saves time, and helps staff handle more tasks effectively, leading to better service quality and smoother hostel management.

1.3.3. Efficient Data Management:

To create a centralized database that stores all hostel-related information such as student details, room information, fee records, and staff data. This reduces paperwork and makes data access faster and more reliable.

1.3.4. Automation of Manual Processes:

To replace traditional manual procedures with an automated system that simplifies tasks like room allocation, student registration, and fee tracking, reducing human effort and errors.

1.3.5. Time Saving and Faster Operations:

To minimize the time required for completing hostel-related activities by enabling quick data entry, retrieval, and updates, thereby reducing waiting time for students and staff.

1.3.6. Improved Accuracy and Reliability:

To eliminate duplicate entries and reduce mistakes by maintaining structured and validated data, ensuring accurate records and trustworthy information.

1.3.7. Support for Decision Making:

To help hostel administrators generate reports on room occupancy, fee status, and resource usage, enabling informed and effective decision-making.

1.4 SCOPE OF THE PROJECT

The scope of the Hostel Management System covers all essential administrative functions required for smooth hostel operations. This includes managing student profiles, room details, staff details, fee records, attendance, and complaints. The system supports adding, updating, deleting, and viewing records through a centralized database.The project is designed to be scalable, allowing future enhancements such as online payment integration, mobile application support, biometric-based attendance, and SMS/email notifications. Although the system is primarily intended for hostel administrators, it can also be expanded to provide limited access to students for viewing their details and submitting requests. The system can be adapted for different types of hostels, including boys', girls', and co-educational hostels.

CHAPTER 2

SYSTEM REQUIREMENT SPECIFICATIONS (SRS)

2.1 FUNCTIONAL REQUIREMENTS

The Hostel Management System is designed to perform all essential operations required for the smooth functioning of a hostel. The system allows administrators to manage student records by adding, updating, deleting, and viewing student information such as personal details, academic information, and contact data. It supports room management by maintaining room details, capacity, and availability status, and enables easy room allocation and deallocation to students. The system handles fee management by recording payment details, tracking due amounts, and generating fee-related reports. It also manages staff information including wardens and maintenance staff. Students can submit complaints and maintenance requests through the system, and administrators can monitor and resolve these issues efficiently. The system includes secure user authentication to ensure that only authorized users can access and modify data. Additionally, it generates various reports related to room occupancy, student details, fee status, and complaints, helping administrators make informed decisions. All these functions work together to improve efficiency, accuracy, and transparency in hostel operations.

2.2 NON-FUNCTIONAL REQUIREMENTS

The Hostel Management System must be designed to deliver a reliable, secure, and user-friendly experience to all users. The system should provide quick response times when performing operations such as searching student records, allocating rooms, or updating fee details to ensure smooth workflow. It must maintain high levels of security by enforcing secure login procedures and restricting unauthorized access to sensitive data. The system should be scalable, meaning it can handle a growing number of users and records without significant delays or performance issues. It should be easy to use, requiring minimal training, and should offer consistent performance with minimal downtime. Additionally, the system should support data backup and recovery features to protect against accidental data loss or system failures.

The system should provide fast response times while retrieving or updating data, even when multiple users are accessing it simultaneously. It must ensure data security by restricting access through authentication and authorization. The system should be scalable so that it can handle an increasing number of students and records in the future. It should also be user-friendly, with a simple and intuitive interface that requires minimal training. Regular data backup and recovery mechanisms should be supported to prevent data loss. The system must be compatible with different devices and operate consistently without frequent system failures.

2.3 HARDWARE REQUIREMENTS

The Hostel Management System requires a basic computing environment for proper functioning. The recommended hardware includes a computer system with a minimum of an Intel i3 processor or its equivalent, at least 4 GB of RAM, and sufficient storage capacity such as 500 GB of hard disk space to store application files and database records. Input devices like a keyboard and mouse are required for user interaction, and a monitor is necessary for viewing the system interface. For network-based operations, a reliable internet connection and networking hardware such as routers or switches may be required. Printers can also be used for generating physical copies of reports and records when needed. Input devices like a keyboard and mouse are required for user interaction, and a monitor is necessary for viewing the system interface. For network-based operations, a reliable internet connection and networking hardware such as routers or switches may be required. A database management system like MySQL or SQL Server is required to store and manage data. A web browser such as Google Chrome, Mozilla Firefox, or Microsoft Edge is needed if the system is web-based. Development tools such as Visual Studio Code, PHP MyAdmin, or other database tools may be used during development. The system may also require a local server environment like XAMPP or WAMP for running the application. Printers can also be used for generating physical copies of reports and records when needed.

2.4 SOFTWARE REQUIREMENTS

The Hostel Management System requires a compatible operating system such as Windows, Linux, or macOS to run efficiently, along with a database management system like MySQL or SQL Server for storing and managing data. A web server environment such as XAMPP or WAMP may be required if the system is developed as a web application. Web browsers like Google Chrome, Mozilla Firefox, or Microsoft Edge are needed to access the system. Development and management tools such as Visual Studio Code, phpMyAdmin, or database administration tools are used to create, maintain, and update the system, ensuring smooth functioning and compatibility across platforms. . A database management system such as MySQL or Microsoft SQL Server is necessary to store and manage hostel data securely. If the system is web-based, a web server environment like XAMPP or WAMP may be used to host the application. Web browsers such as Google Chrome, Mozilla Firefox, or Microsoft Edge are required to access the system interface. Development and administration tools like Visual Studio Code, phpMyAdmin, or similar software are used for system development, testing, and maintenance. The system is designed to work smoothly with commonly available industry-standard software to ensure compatibility and ease of deployment.

2.5 USER CHARACTERISTICS

The intended users of the Hostel Management System include hostel administrators, wardens, clerks, and other authorized staff members who are responsible for managing hostel operations. These users are expected to have basic computer literacy, such as the ability to use a keyboard, mouse, and navigate application menus. The system is also designed to be accessible to students who may use it to view their personal information, room details, or submit complaints. Since users may come from non-technical backgrounds, the system features a simple and intuitive interface that minimizes complexity and reduces the learning curve. The design focuses on making operations easy and efficient, even for first-time users.Users are expected to have basic computer knowledge such as using a keyboard, mouse, and navigating software applications. No advanced technical skills are required. Students may also interact with the system for viewing their details or submitting complaints. The system is designed to be simple so that users from non-technical backgrounds can use it easily without extensive training.

Overall, Users are expected to have basic computer knowledge such as using a keyboard, mouse, and navigating software applications. No advanced technical skills are required. Students may also interact with the system for viewing their details or submitting complaints. The system is designed to be simple so that users from non-technical backgrounds can use it easily without extensive training.

2.6 CONSTRAINTS

The system is subject to certain limitations such as hardware capacity, network availability, and institutional policies. Performance may be affected if the system is used on low-specification computers or with unstable internet connectivity. The system must comply with data protection and privacy regulations. Budget constraints may limit the use of advanced features such as biometric authentication or mobile app development. Additionally, the system depends on proper data entry by users, and incorrect input may affect the accuracy of the information stored in the system.The Hostel Management System operates under certain constraints that may affect its performance and functionality. Hardware limitations such as low memory or processing power can slow down system operations. Network constraints, including slow or unstable internet connectivity, may impact data synchronization and system access, especially in a web-based environment. Budget limitations may restrict the implementation of advanced features such as real-time notifications, biometric authentication, or mobile application development. The system must comply with institutional policies and data protection laws to ensure the privacy and security of user information. Additionally, the system depends heavily on accurate data input by users, and incorrect or incomplete entries may affect the overall reliability of the system..

CHAPTER 3

ANALYSIS AND DESIGN

3.1 CLASS DIAGRAM

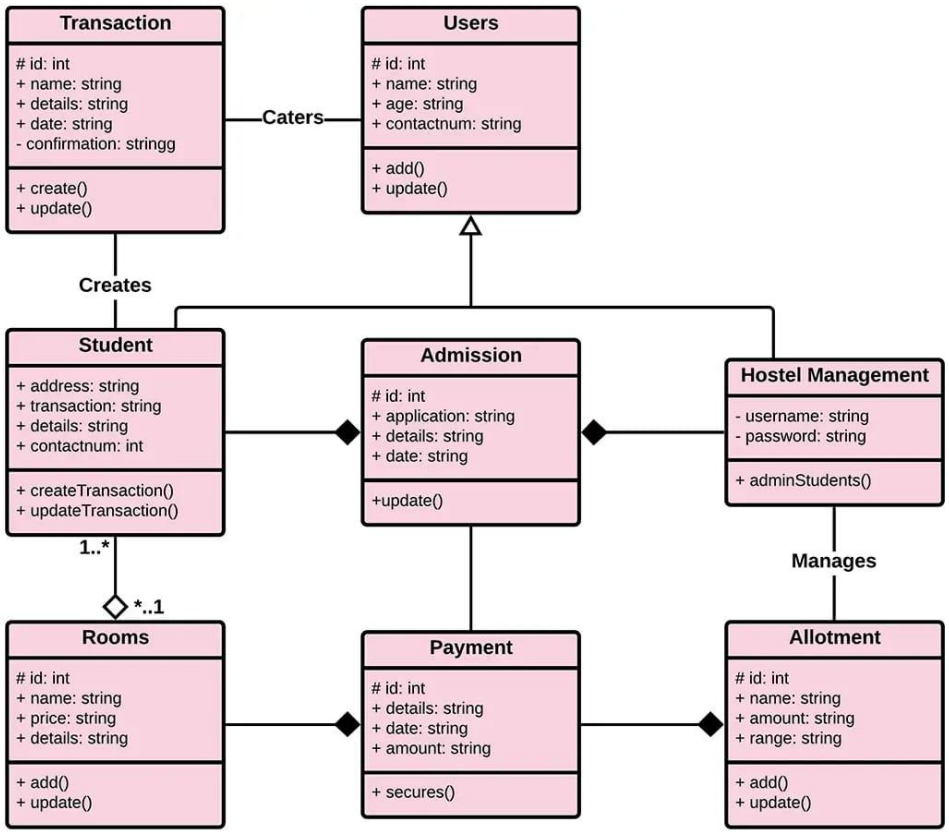

Figure 3.1 Class Diagram

3.1.1 Use Case Description

The hostel management system class diagram represents the interaction between different components involved in managing hostel activities. At the core, the Users class serves as the base class, from which the Student and Hostel Management classes derive. Students undergo the admission process, represented by the Admission class, which stores details such as application information and admission dates. Once admitted, students are allotted rooms through the Allotment class, which works alongside the *Rooms* class that maintains room details like pricing and availability. Financial activities are handled through the Transaction and Payment classes, where students' payments create secure transactions linked to their accounts. The Hostel Management class oversees the entire process, managing student records, admissions, allotments, and updates. The relationships between these classes ensure smooth coordination between admission, room allotment, and financial management, forming a complete and efficient hostel administration workflow.

3.2 USE CASE DIAGRAM

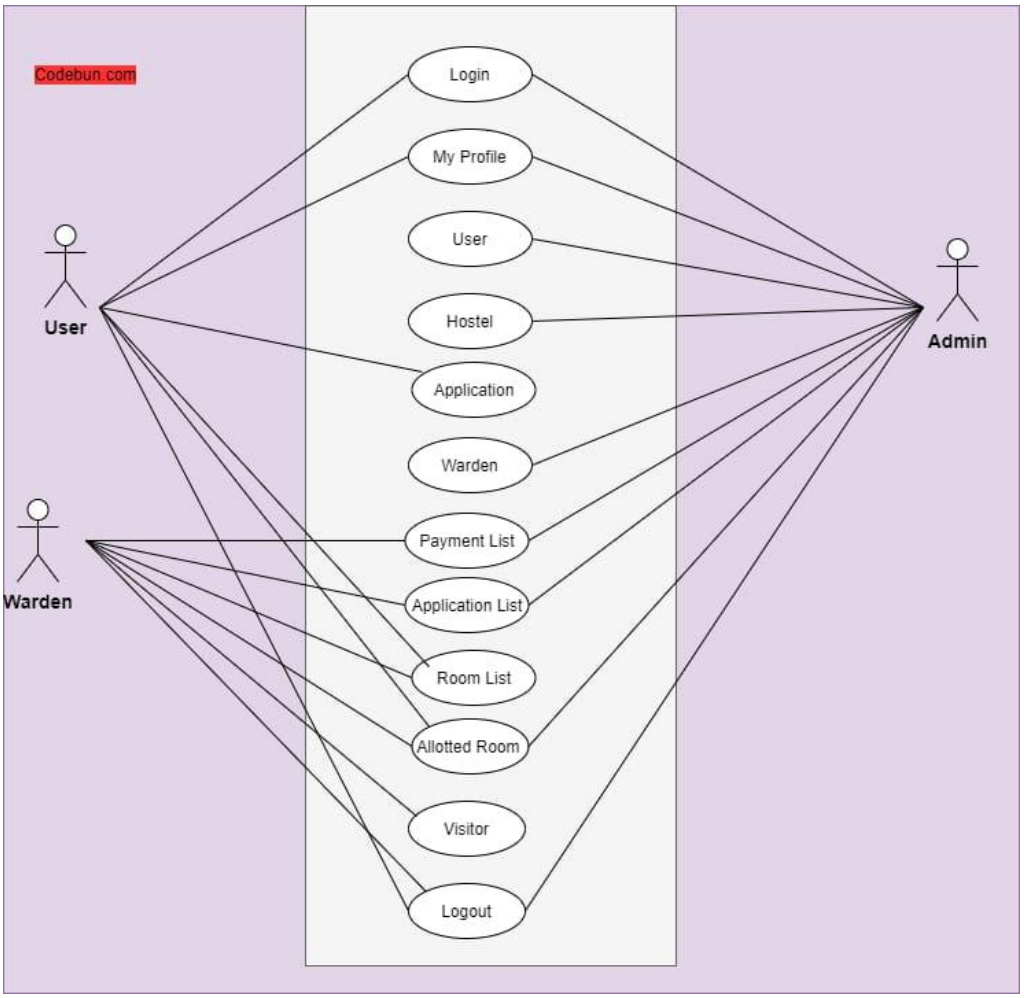

Figure 3.2 Use Case Diagram

3.2.1 Use Case Description

The use case diagram represents the overall functionality of the Hostel Management System and the interactions between its three main actors: User, Warden, and Admin. The User can perform basic operations such as logging in, viewing or updating their profile, applying for hostel accommodation, and checking allotted room details or visitor information. The Warden has additional responsibilities and can access the payment list, application list, room list, allotted rooms, and manage visitor entries, along with basic login and profile functions. The Admin has full system control and can manage users, hostels, wardens, applications, rooms, payments, and all allotment processes. The central system contains all use cases like Login, Application, Hostel, Room List, Payment List, Visitor, and Logout, which different actors access based on their roles. This diagram clearly shows how each actor interacts with the system and highlights the administrative hierarchy, where the Admin has the highest privileges, followed by the Warden, and then the regular User.

3.3 SEQUENCE DIAGRAM

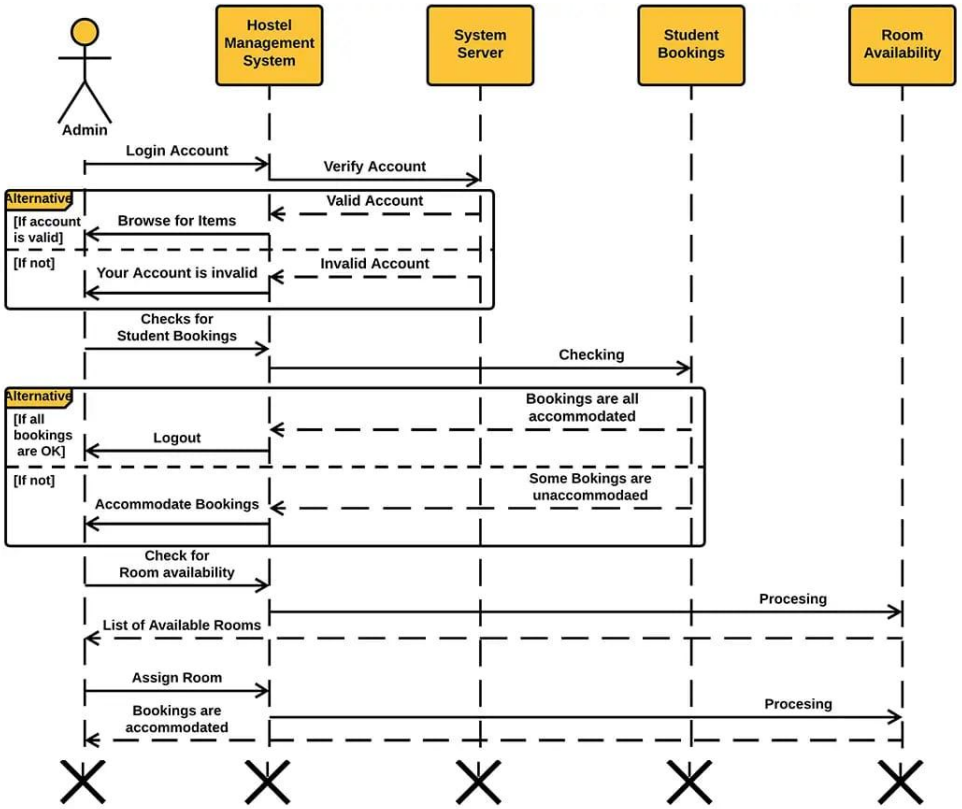

Figure 3.3 Sequence Diagram

3.3.1 Sequence Diagram Description

The sequence diagram illustrates the workflow of the hostel management system when an admin interacts with the system to manage student bookings and room availability. The process begins with the admin logging into the system, after which the Hostel Management System sends the login details to the System Server for verification. If the account is valid, the system allows the admin to browse items; otherwise, it displays an invalid account message. Once the admin is logged in, the system checks the Student Bookings to identify whether all students are properly accommodatedNext, the system checks room availability by communicating with the Room Availability module, retrieves the list of available rooms, and assigns a suitable room to each pending booking. After assigning rooms, the system updates the status to confirm that all bookings have been accommodated. This sequence diagram clearly shows the interaction flow between the admin and the various system components involved in verifying accounts, managing bookings, and assigning rooms efficiently.

3.4 ACTIVITY DIAGRAM

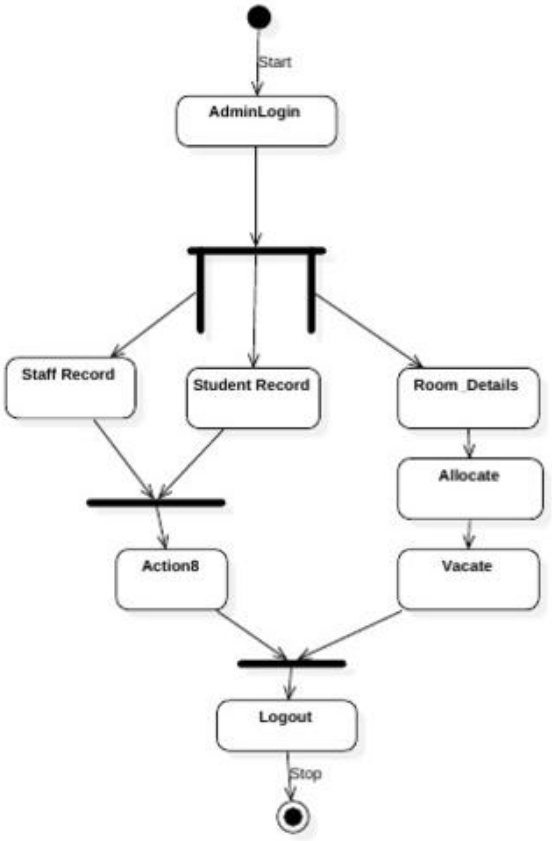

Figure 3.4 Activity Diagram

3.4.1 Activity Diagram Description

The activity diagram illustrates the step-by-step flow of activities performed by the Admin in the Hostel Management System. It shows how the admin navigates through different tasks such as managing staff records, student details, room allocation, and vacating rooms before finally logging out..The first activity is AdminLogin, where the admin enters credentials to gain access to the system. This is the entry point to all administrative functionalities. If the admin selects Staff Record, they can view, update, or manage hostel staff details.After completing this, the workflow joins with other activities through a join node, meaning the next step depends on completing the task.

3.5 COMPONENT DIAGRAM

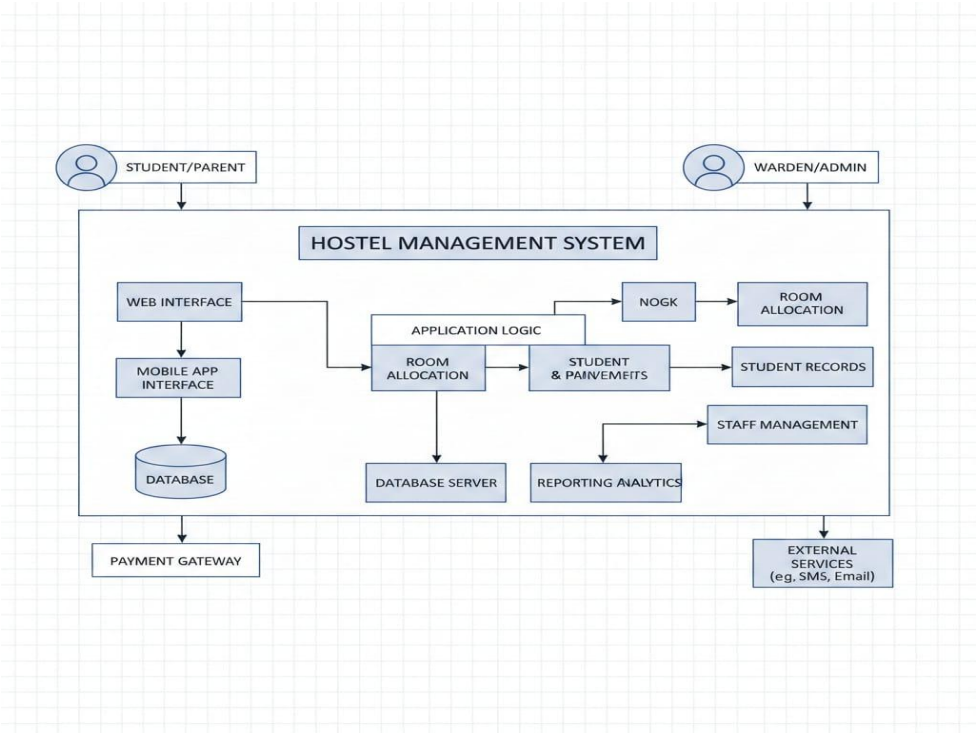

Figure 3.5 Component Diagram

3.5.1 Component Diagram Description

The component diagram illustrates the overall architecture of the Hostel Management System (HMS) by showing how different functional modules, users, and external services interact with each other. It highlights the major components such as interfaces, application logic, database layers, and integrated external systems that work together to deliver a complete hostel management solution. The system has two primary users: Students/Parents and Warden/Admin. Students access the system mainly for room booking, fee payments, and information updates, while the admin manages student records, room allocation, and staff-related activities. Overall, this component diagram shows a structured hostel management system where user interfaces, application logic, database layers, and external integrations work together. Each component has a specific responsibility, creating an efficient, automated, and user-friendly environment for both students and administrators.

3.6 DEPLOYMENT DIAGRAM

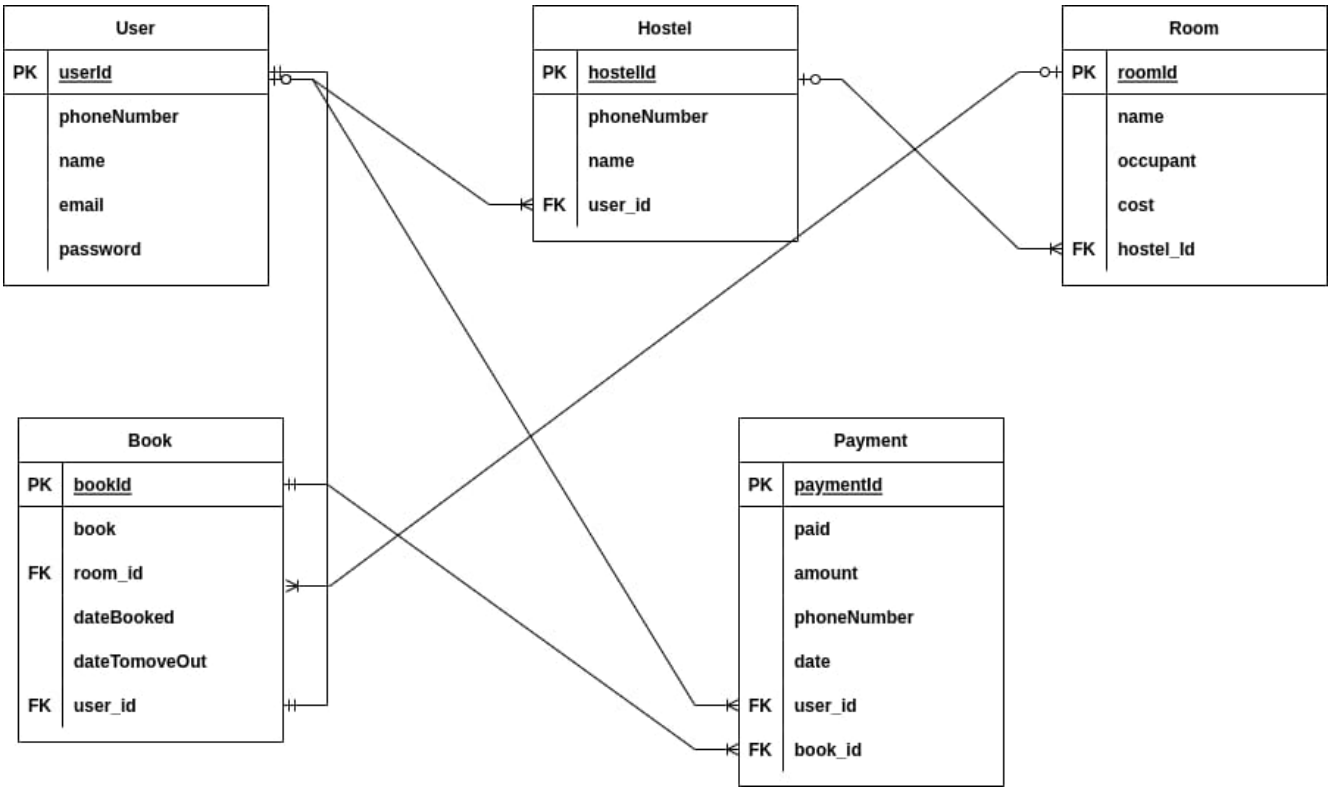

Figure 3.6 Deployment Diagram

3.6.1 Deployment Diagram Description

The deployment diagram illustrates the physical architecture of the Hostel Management System by showing how software components are deployed across different hardware nodes. The system is primarily divided into three major nodes: the Client Device, the Application Server, and the Database Server. Each node hosts specific components required for the system to function efficiently.The Client Device node represents the devices used by students, parents, wardens, and administrators to access the system. This includes desktops, mobile phones, and tablets through which users interact using the web or mobile interface. These devices communicate with the backend server using standard internet protocols. The client node mainly runs the browser or the mobile application that sends requests to the server.

3.7 PACKAGE DIAGRAM

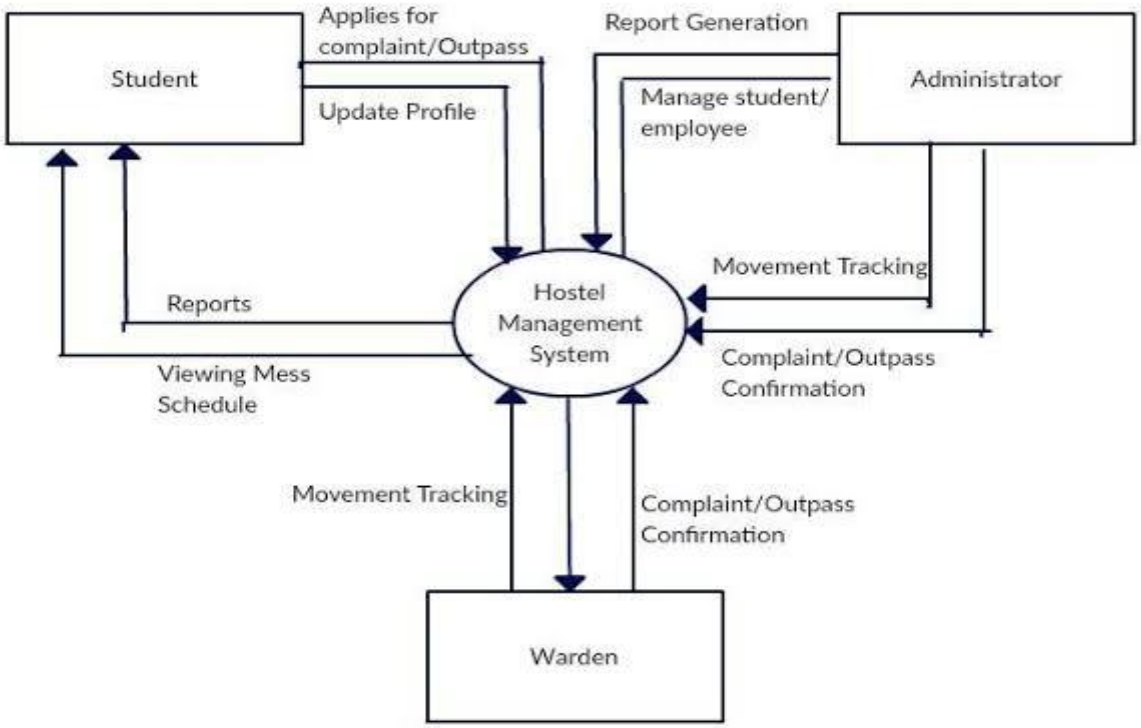

Figure 3.7 Package Diagram

3.7.1 Package Diagram Description

A package diagram for this Hostel Management System would organize the listed features into logical, modular groups based on user roles and shared functionality. The system can be structured into three main packages: Student, Warden, and Common. The Student package would contain modules for profile updates, complaint or outpass applications, mess schedule viewing, movement tracking, and report generation—all tailored for student access. The Warden package would include features for managing students and employees, confirming complaints or outpasses, overseeing hostel operations, generating administrative reports, and monitoring student movements. This modular approach ensures separation of concerns, reduces duplication, and maintains clear dependencies, with the Student and Warden packages both relying on the Common package for core functionalities while operating independently within their respective domains.

3.8 COLLABORATION DIAGRAM

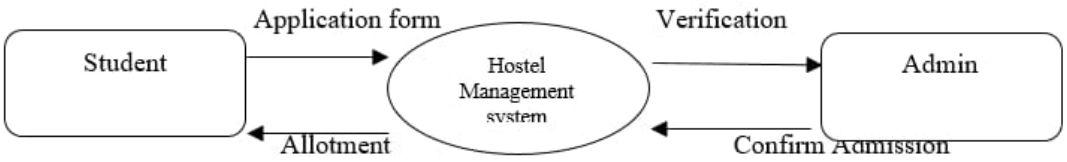

Fig 1 DFD For Allotment Process

Fig 2 DFD For Student Login

Fig 3 DFD For Student Registration

Figure 3.8 Collaboration Diagram

3.8.1 Collaboration Diagram Description

Hostel Management System (HMS), and Admin during different hostel-related processes. In the allotment process, the student submits an application form to the Hostel Management System. The system forwards the details to the admin for verification, and after confirmation of admission by the admin, the hostel allotment information is sent back to the student. In the student login process, the student provides a username and login details, which are validated by the system. Upon successful authentication, the HMS allows access to the student register. In the student registration process, the student enters personal details through the application filling module, which is reviewed and approved by the admin. Overall, this collaboration diagram shows how different components communicate and coordinate with each other to ensure smooth registration, login, verification, and hostel allotment within the Hostel Management System.

CHAPTER 4

IMPLEMENTATION

4.1 MODULE DESCRIPTION

4.1.1 Login and Authentication Module:

This module ensures that only authorized users can access the hostel management system. It verifies user credentials such as username and password and provides secure login for students, wardens, and administrators. Based on the authenticated user role, the system grants appropriate privileges and displays the corresponding dashboard. It also includes logout functionality and security measures like password validation and session handling to protect user information.

4.1.2 Student Management Module

The Student Management Module handles all information related to students who apply for or stay in the hostel. It maintains records such as personal details, contact information, applications, and profile updates. Administrators and wardens can easily add new students, edit existing data, and view complete student profiles when needed. This module ensures organized data storage and simplifies the management of student information.

4.1.3 Room Allocation Module

The Room Allocation Module manages the distribution and tracking of hostel rooms. It records room availability, assigns rooms to newly admitted students, and updates room status after allotment or shifting. The module ensures that rooms are allocated fairly and efficiently by matching available rooms with student requirements. It also helps administrators generate allotment lists and maintain proper room occupancy records.

.4.1.4 Fee Management Module

The Fee Management Module oversees all financial transactions related to hostel fees. It allows students to pay their fees and enables administrators to record payments, track pending dues, and maintain detailed payment histories. This module ensures transparency by generating receipts and keeping accurate financial records. It helps both students and administrators monitor fee-related activities easily.

4.1.5 Attendance Management Module

The Attendance Management Module tracks the daily presence of students within the hostel. It records entry and exit timings, monitors absences, and identifies irregular attendance patterns. Wardens use this module to maintain discipline and ensure student safety. By storing attendance logs and generating reports, it supports effective hostel administration and enhances security.

4.2 TECHNOLOGY DESCRIPTION

The Hostel Management System is developed using a combination of modern and reliable technologies that ensure efficient performance, easy maintenance, and user-friendly interaction. The application is primarily built using HTML, CSS, and JavaScript on the frontend to create responsive web pages and provide smooth user interaction. For the backend, PHP (or Java/Python, if your project uses a different language) is used to handle server-side logic, manage requests, and connect the user interface with the database. The system's data is stored in a MySQL database, which maintains student records, room details, fee transactions, attendance logs, and administrative information in a structured and secure manner. SQL queries are used for storing, retrieving, and updating database values. Additionally, the project follows the XAMPP/WAMP server environment (if applicable) to host the database and run PHP scripts locally. The overall design uses UML diagrams, including use case, class, and sequence diagrams, to model system behavior and architecture. These technologies collectively ensure that the system is scalable, secure, and easy to deploy for real hostel management operations.

CHAPTER 5
TESTING

5.1 TESTING STRATEGY

The testing strategy for the Hostel Management System focuses on ensuring that all modules function correctly, efficiently, and reliably under different conditions. The system is tested using a combination of black-box testing, white-box testing, and user acceptance testing (UAT). Black-box testing is used to verify the functionality of each module—such as login, student management, room allocation, fee payment, and attendance—without looking into the internal code. White-box testing is applied to check the internal logic, input validation, and database queries in order to identify hidden errors or performance issues. Integration testing ensures that modules interact correctly, for example, verifying that room allotment updates student records or payment transactions are stored in the database. System testing is carried out to evaluate the behavior of the entire application as a complete unit. Finally, user acceptance testing is conducted with the help of students, wardens, and admins to confirm that the system meets real-world requirements and is easy to use. Through this multi-level testing strategy, the project ensures high quality, accuracy, and reliability.

5.2 SAMPLE TEST CASE

TC01 - Login Module

Test Case: Enter valid credentials → System logs in successfully.

Description:

Allows users (Student/Warden/Admin) to enter their username and password to access the system. It ensures that only authorized users are allowed to log in securely.

Input: Valid username & password.

Expected Output: Login successful.

Status: Pass.

TC02 – Registration Module

Test Case: Submit valid student details → New account created.

Description:

This module lets new students register by entering personal details such as name, email, phone, and student ID. The system stores this information for future login and hostel processes.

Input: New student registration details.

Expected Output: Account created.

Status: Pass.

TC03 – Room Allocation Module

Test Case: Allocate room to student → Room appears in student profile.

Description:

Admin or Warden assigns an available room to a registered student. The module updates room status and displays the allotted room in the student's account.

Input: Student ID and available room number.

Expected Output**:** Room allocated successfully.

Status: Pass.

TC04 – Fee Payment Module

Test Case: Enter valid fee details → Payment is recorded.

Description:

Allows students to pay hostel fees through online or offline methods. The module verifies payment information and stores the transaction in the system.

Input: Valid payment details (transaction ID/UPI/card).

Expected Output: Payment successful and recorded.

Status: Pass.

TC05 – Student Attendance Module

Test Case: Mark attendance → Attendance updated.

Description:

Wardens mark daily attendance of hostel students. The module updates the attendance record and stores the entry for reports and monitoring.

Input: Student ID and attendance status.

Expected Output: Attendance marked successfully.

Status: Pass.

TC06 – Visitor Management Module

Test Case: Add visitor entry → Visitor is recorded.

Description:

Allows wardens to register visitors by noting name, purpose, and student they are visiting. The system stores this entry for security and tracking.

Input: Visitor details.

Expected Output: Visitor entry added.

Status: Pass.

TC07 – Complaint Management Module

Test Case: Submit complaint → Complaint appears in admin dashboard.

Description:

Students can submit complaints regarding hostel issues such as maintenance or food. The module stores the complaint and forwards it to the admin for action.

Input: Complaint form details.

Expected Output: Complaint recorded successfully.

Status: Pass.

TC08 – Report Module

Test Case: Select date range → System generates report.

Description:

Generates daily, weekly, or monthly reports for fees, attendance, room allotment, or complaints. Helps admin analyze hostel activities.

Input: Date range.

Expected Output: Report generated.

Status: Pass.

TC09 – Notification Module

Test Case: New event occurs (room allotment/payment/complaint update) → Notification sent.

Description:

Sends alerts to students and staff about room allotments, fee confirmations, complaint updates, or important hostel announcements.

Input: Trigger event.

Expected Output: Notification delivered.Status: Pass.

5.3 Test results

The testing of the Smart Canteen Management System showed that all major modules worked correctly. The Login module successfully allowed users to access the system with valid credentials. The Registration module created new user accounts without any issues. Menu Management worked properly by adding and updating food items. The Order Placement module accepted selected items and generated orders correctly. The Payment module processed online payments successfully using valid details. Order Tracking displayed the correct status for each order. Inventory Management updated stock levels accurately whenever changes were made. The Report Generation module produced sales reports without errors, and the Notification module successfully sent alerts to users when their order was ready. Overall, all modules performed as expected and passed the test.

CHAPTER 6

CONCLUSION & FUTURE SCOPE

6.1 CONCLUSION

The Hostel Management System successfully automates and streamlines various hostel-related operations such as student registration, room allocation, fee management, attendance tracking, and administrative control. By replacing manual processes with a centralized digital system, it improves accuracy, reduces paperwork, and saves time for both students and administrators. The system ensures secure login, maintains organized records, and provides quick access to essential information. Through proper testing, each module was verified to function efficiently and reliably. Overall, the project achieves its objective of creating a user-friendly, secure, and efficient solution for hostel administration and enhances the overall management experience.

6.2 FUTURE SCOPE

The system can be further enhanced with several advanced features to improve performance and user convenience. Integration of biometric or RFID-based attendance can automate entry and exit tracking more accurately. Adding mobile app support will allow students and wardens to access services on the go. Online payment gateways can be extended to include more digital modes and automatic invoice generation. The system can also incorporate advanced analytics for predicting room usage, fee payment patterns, and student behavior. Additionally, AI-based chatbots can be included for answering common student queries. With cloud deployment, the system can support remote access and scalable data storage. These enhancements will make the hostel management system more powerful, flexible, and future-ready.

5.4 APPENDIX A – Source Code

PHP CODE:

```
<?php
   session_start();
   include('../includes/dbconn.php');
   if(isset($_POST['login'])){
   $username=$_POST['username'];
   $password=$_POST['password'];
   $password = md5($password);
   $stmt=$mysqli->prepare("SELECT username,email,password,id FROM admin WHERE (userName=?|| 
email=?) and password=? ");
                    $stmt->bind_param('sss',$username,$username,$password);
                    $stmt->execute();
                    $stmt -> bind_result($username,$username,$password,$id);
                    $rs=$stmt->fetch();
                    $_SESSION['id']=$id;
                    $uip=$_SERVER['REMOTE_ADDR'];
                    $ldate=date('d/m/Y h:i:s', time());
                    if($rs){
                              header("location:dashboard.php");
                                        } else {
                                                  echo "<script>alert('Invalid Username/Email or password');</script>";
                                        }
          }
?>

<!DOCTYPE html>
<html dir="ltr">

<head>
   <meta charset="utf-8">
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <!-- Tell the browser to be responsive to screen width -->
   <meta name="viewport" content="width=device-width, initial-scale=1">
   <meta name="description" content="">
```

```
    <meta name="author" content="">
    <!-- Favicon icon -->
    <link rel="icon" type="image/png" sizes="16x16" href="../assets/images/favicon.png">
    <title>Hostel Management System</title>
    <!-- Custom CSS -->
    <link href="../dist/css/style.min.css" rel="stylesheet">

    <script type="text/javascript">
    function valid() {
    if(document.registration.password.value!= document.registration.cpassword.value){
       alert("Password and Re-Type Password Field do not match  !!");
    document.registration.cpassword.focus();
    return false;
       }
    return true;
       }
    </script>

</head>

<body>
    <div class="main-wrapper">
        <!-- ============================================================== -->
        <!-- Preloader - style you can find in spinners.css -->
        <!-- ============================================================== -->
        <div class="preloader">
            <div class="lds-ripple">
                <div class="lds-pos"></div>
                <div class="lds-pos"></div>
            </div>
        </div>
        <!-- ============================================================== -->
        <!-- Preloader - style you can find in spinners.css -->
        <!-- ============================================================== -->

        <!-- ============================================================== -->
        <!-- Login box.scss -->
        <!-- ============================================================== -->
```

```
<div class="auth-wrapper d-flex no-block justify-content-center align-items-center position-relative"
    style="background:url(../assets/images/big/auth-bg.jpg) no-repeat center center;">
    <div class="auth-box row">
        <div class="col-lg-7 col-md-5 modal-bg-img" style="background-image:
url(../assets/images/adimg.jpg);">
        </div>
        <div class="col-lg-5 col-md-7 bg-white">
            <div class="p-3">
                <div class="text-center">
                    <img src="../assets/images/big/icon.png" alt="wrapkit">
                </div>
                <h2 class="mt-3 text-center">Admin Login</h2>

                <form class="mt-4" method="POST">
                    <div class="row">
                        <div class="col-lg-12">
                            <div class="form-group">
                                <label class="text-dark" for="uname">Email or Username</label>
                                <input class="form-control" name="username" id="uname" type="text"
                                    placeholder="Email or Username" required>
                            </div>
                        </div>
                        <div class="col-lg-12">
                            <div class="form-group">
                                <label class="text-dark" for="pwd">Password</label>
                                <input class="form-control" name="password" id="pwd" type="password"
                                    placeholder="Enter your password" required>
                            </div>
                        </div>
                        <div class="col-lg-12 text-center">
                            <button type="submit" name="login" class="btn btn-block btn-
danger">LOGIN</button>
                        </div>
                        <div class="col-lg-12 text-center mt-5">
                            <a href="../index.php" class="text-danger">Go Back</a>
                        </div>
                    </div>
                </form>
```

```
                </div>
            </div>
        </div>
    </div>
    <!-- ============================================================== -->
    <!-- Login box.scss -->
    <!-- ============================================================== -->
</div>
<!-- ============================================================== -->
<!-- All Required js -->
<!-- ============================================================== -->
<script src="../assets/libs/jquery/dist/jquery.min.js "></script>
<!-- Bootstrap tether Core JavaScript -->
<script src="../assets/libs/popper.js/dist/umd/popper.min.js "></script>
<script src="../assets/libs/bootstrap/dist/js/bootstrap.min.js "></script>
<!-- ============================================================== -->
<!-- This page plugin js -->
<!-- ============================================================== -->
<script>
    $(".preloader ").fadeOut();
</script>
</body>

</html>

<?php
    session_start();
    include('../includes/dbconn.php');
    include('../includes/check-login.php');
    check_login();

    if(isset($_GET['del']))
    {
        $id=intval($_GET['del']);
        $adn="DELETE from rooms where id=?";
            $stmt= $mysqli->prepare($adn);
            $stmt->bind_param('i',$id);
            $stmt->execute();
```

```
        $stmt->close();
        echo "<script>alert('Record has been deleted');</script>" ;
    }
?>

<!DOCTYPE html>
<html dir="ltr" lang="en">

<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <!-- Tell the browser to be responsive to screen width -->
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <meta name="description" content="">
    <meta name="author" content="">
    <!-- Favicon icon -->
    <link rel="icon" type="image/png" sizes="16x16" href="../assets/images/favicon.png">
    <title>Hostel Management System</title>
    <!-- Custom CSS -->
    <link href="../assets/extra-libs/c3/c3.min.css" rel="stylesheet">
    <link href="../assets/libs/chartist/dist/chartist.min.css" rel="stylesheet">
     <!-- This page plugin CSS -->
     <link href="../assets/extra-libs/datatables.net-bs4/css/dataTables.bootstrap4.css" rel="stylesheet">
    <!-- Custom CSS -->
    <link href="../dist/css/style.min.css" rel="stylesheet">

</head>

<body>
    <!-- ============================================================== -->
    <!-- Preloader - style you can find in spinners.css -->
    <!-- ============================================================== -->
    <div class="preloader">
        <div class="lds-ripple">
            <div class="lds-pos"></div>
            <div class="lds-pos"></div>
        </div>
    </div>
```

```
    <!-- ============================================================== -->
    <!-- Main wrapper - style you can find in pages.scss -->
    <!-- ============================================================== -->
    <div id="main-wrapper" data-theme="light" data-layout="vertical" data-navbarbg="skin6" data-
sidebartype="full"
        data-sidebar-position="fixed" data-header-position="fixed" data-boxed-layout="full">
        <!-- ============================================================== -->
        <!-- Topbar header - style you can find in pages.scss -->
        <!-- ============================================================== -->
        <header class="topbar" data-navbarbg="skin6">
            <?php include 'includes/navigation.php'?>
        </header>
        <!-- ============================================================== -->
        <!-- End Topbar header -->
        <!-- ============================================================== -->
        <!-- ============================================================== -->
        <!-- Left Sidebar - style you can find in sidebar.scss  -->
        <!-- ============================================================== -->
        <aside class="left-sidebar" data-sidebarbg="skin6">
            <!-- Sidebar scroll-->
            <div class="scroll-sidebar" data-sidebarbg="skin6">
                <?php include 'includes/sidebar.php'?>
            </div>
            <!-- End Sidebar scroll-->
        </aside>
        <!-- ============================================================== -->
        <!-- End Left Sidebar - style you can find in sidebar.scss  -->
        <!-- ============================================================== -->
        <!-- ============================================================== -->
        <!-- Page wrapper  -->
        <!-- ============================================================== -->
        <div class="page-wrapper">
            <!-- ============================================================== -->
            <!-- Bread crumb and right sidebar toggle -->
            <!-- ============================================================== -->
            <div class="page-breadcrumb">
                <div class="row">
                    <div class="col-7 align-self-center">
```

```
            <h4 class="page-title text-truncate text-dark font-weight-medium mb-1">Rooms
Management</h4>
                <div class="d-flex align-items-center">
                  <!-- <nav aria-label="breadcrumb">

                  </nav> -->

                </div>
              </div>

            </div>

        </div>
        <!-- ============================================================== -->
        <!-- End Bread crumb and right sidebar toggle -->
        <!-- ============================================================== -->
        <!-- ============================================================== -->
        <!-- Container fluid  -->
        <!-- ============================================================== -->
        <div class="container-fluid">

          <!-- Table Starts -->
          <div class="row">
            <div class="col-12">
              <div class="card">
                <div class="card-body">
                <a href="add-rooms.php"><button type="button" class="btn btn-block btn-md btn-
success">Add New Room Details</button></a>
                <hr>
                  <div class="table-responsive">
                    <table id="zero_config" class="table table-striped table-hover table-bordered no-wrap">
                    <thead class="thead-dark">
                        <tr>
                          <th>#</th>
                          <th>Room No.</th>
                          <th>Seater</th>
                          <th>Fees Per Month</th>
                          <th>Published On</th>
```

```
                                    <th>Actions</th>
                                </tr>
                            </thead>
                            <tbody>
                            <?php
                                $aid=$_SESSION['id'];
                                $ret="SELECT * from rooms";
                                $stmt= $mysqli->prepare($ret) ;
                                //$stmt->bind_param('i',$aid);
                                $stmt->execute() ;//ok
                                $res=$stmt->get_result();
                                $cnt=1;
                                while($row=$res->fetch_object())
                                    {
                                        ?>
                            <tr><td><?php echo $cnt;;?></td>
                            <td><?php echo $row->room_no;?></td>
                            <td><?php echo $row->seater;?></td>
                            <td>$<?php echo $row->fees;?></td>
                            <td><?php echo $row->posting_date;?></td>
                            <td><a href="edit-room.php?id=<?php echo $row->id;?>" title="Edit"><i class="icon-
note"></i></a>  
                            <a href="manage-rooms.php?del=<?php echo $row->id;?>" title="Delete"
onclick="return confirm("Do you want to delete");"><i class="icon-close" style="color:red;"></i></a></td>
                            </tr>
                                <?php
                                    $cnt=$cnt+1;
                                } ?>
                                                                                    </tbody>
                        </table>
                    </div>
                </div>
            </div>
        </div>
    </div>

    <!-- Table Ends -->
```

```
        </div>
        <!-- ============================================================== -->
        <!-- End Container fluid  -->
        <!-- ============================================================== -->
        <!-- ============================================================== -->
        <!-- footer -->
        <!-- ============================================================== -->
        <?php include '../includes/footer.php' ?>
        <!-- ============================================================== -->
        <!-- End footer -->
        <!-- ============================================================== -->
    </div>
    <!-- ============================================================== -->
    <!-- End Page wrapper  -->
    <!-- ============================================================== -->
</div>
<!-- ============================================================== -->
<!-- End Wrapper -->
<!-- ============================================================== -->
<!-- End Wrapper -->
<!-- ============================================================== -->
<!-- All Jquery -->
<!-- ============================================================== -->
<script src="../assets/libs/jquery/dist/jquery.min.js"></script>
<script src="../assets/libs/popper.js/dist/umd/popper.min.js"></script>
<script src="../assets/libs/bootstrap/dist/js/bootstrap.min.js"></script>
<!-- apps -->
<!-- apps -->
<script src="../dist/js/app-style-switcher.js"></script>
<script src="../dist/js/feather.min.js"></script>
<script src="../assets/libs/perfect-scrollbar/dist/perfect-scrollbar.jquery.min.js"></script>
<script src="../dist/js/sidebarmenu.js"></script>
<!--Custom JavaScript -->
<script src="../dist/js/custom.min.js"></script>
<!--This page JavaScript -->
<script src="../assets/extra-libs/c3/d3.min.js"></script>
<script src="../assets/extra-libs/c3/c3.min.js"></script>
<script src="../assets/libs/chartist/dist/chartist.min.js"></script>
```

```
    <script src="../assets/libs/chartist-plugin-tooltips/dist/chartist-plugin-tooltip.min.js"></script>
    <script src="../dist/js/pages/dashboards/dashboard1.min.js"></script>
    <script src="../assets/extra-libs/datatables.net/js/jquery.dataTables.min.js"></script>
    <script src="../dist/js/pages/datatable/datatable-basic.init.js"></script>

</body>

</html>

<?php
    session_start();
    include('../includes/dbconn.php');
    include('../includes/check-login.php');
    check_login();

    if(isset($_GET['del']))
    {
        $id=intval($_GET['del']);
        $adn="DELETE from rooms where id=?";
            $stmt= $mysqli->prepare($adn);
            $stmt->bind_param('i',$id);
            $stmt->execute();
            $stmt->close();
            echo "<script>alert('Record has been deleted');</script>" ;
    }
?>

<!DOCTYPE html>
<html dir="ltr" lang="en">
<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <!-- Tell the browser to be responsive to screen width -->
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <meta name="description" content="">
    <meta name="author" content="">
    <!-- Favicon icon -->
    <link rel="icon" type="image/png" sizes="16x16" href="../assets/images/favicon.png">
```

```
    <title>Hostel Management System</title>
    <!-- Custom CSS -->
    <link href="../assets/extra-libs/c3/c3.min.css" rel="stylesheet">
    <link href="../assets/libs/chartist/dist/chartist.min.css" rel="stylesheet">
     <!-- This page plugin CSS -->
     <link href="../assets/extra-libs/datatables.net-bs4/css/dataTables.bootstrap4.css" rel="stylesheet">
    <!-- Custom CSS -->
    <link href="../dist/css/style.min.css" rel="stylesheet">

</head></div>
    <!-- ============================================================== -->
    <!-- End Wrapper -->
    <!-- ============================================================== -->
    <!-- End Wrapper -->
    <!-- ============================================================== -->
    <!-- All Jquery -->
    <!-- ============================================================== -->
    <script src="../assets/libs/jquery/dist/jquery.min.js"></script>
    <script src="../assets/libs/popper.js/dist/umd/popper.min.js"></script>
    <script src="../assets/libs/bootstrap/dist/js/bootstrap.min.js"></script>
    <!-- apps -->
    <!-- apps -->
    <script src="../dist/js/app-style-switcher.js"></script>
    <script src="../dist/js/feather.min.js"></script>
    <script src="../assets/libs/perfect-scrollbar/dist/perfect-scrollbar.jquery.min.js"></script>
    <script src="../dist/js/sidebarmenu.js"></script>
    <!--Custom JavaScript -->
    <script src="../dist/js/custom.min.js"></script>
    <!--This page JavaScript -->
    <script src="../assets/extra-libs/c3/d3.min.js"></script>
    <script src="../assets/extra-libs/c3/c3.min.js"></script>
    <script src="../assets/libs/chartist/dist/chartist.min.js"></script>
    <script src="../assets/libs/chartist-plugin-tooltips/dist/chartist-plugin-tooltip.min.js"></script>
    <script src="../dist/js/pages/dashboards/dashboard1.min.js"></script>
    <script src="../assets/extra-libs/datatables.net/js/jquery.dataTables.min.js"></script>
    <script src="../dist/js/pages/datatable/datatable-basic.init.js"></script>

</body></html>
```

5.5 APPENDIX B -SCREENSHOTS

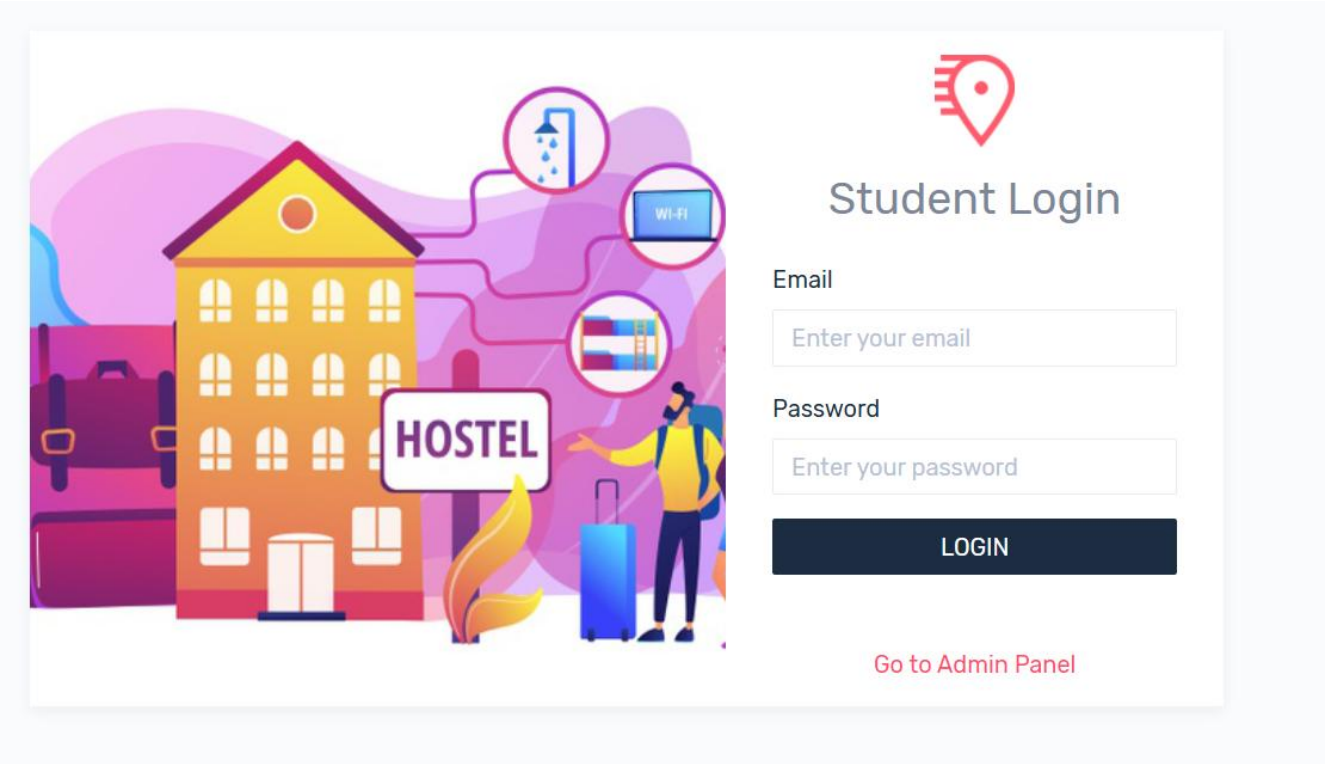

FIGURE B.1 STUDENT LOGIN

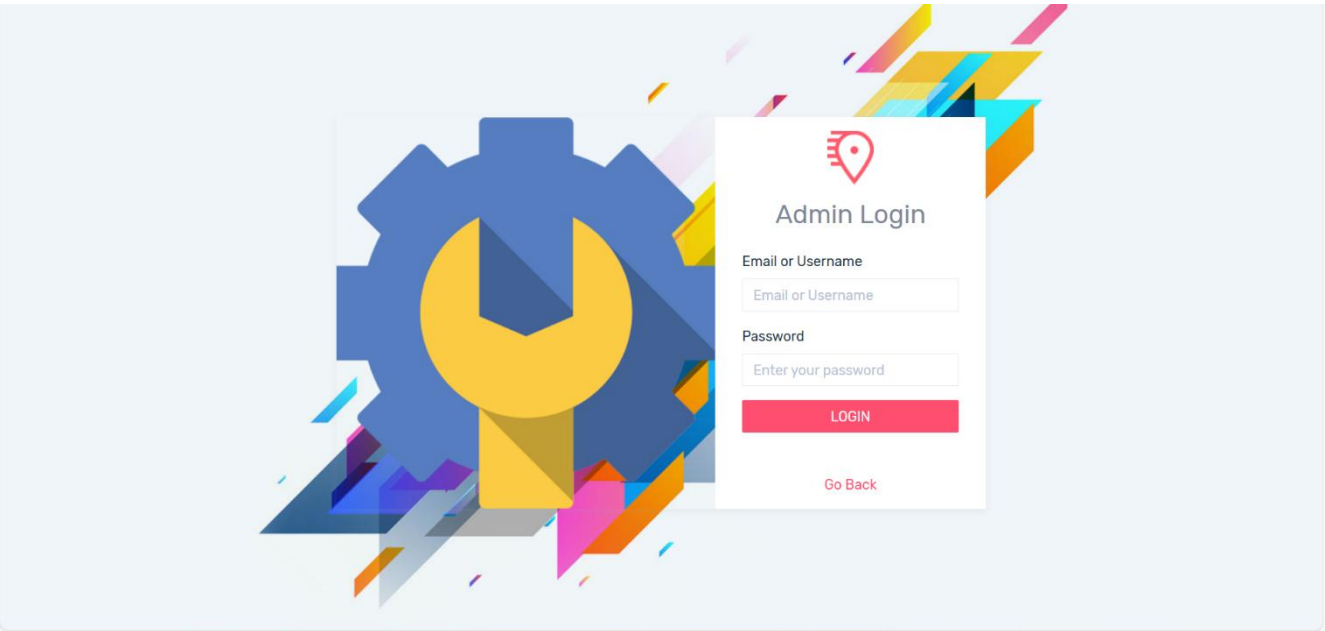

FIGURE B.2 ADMIN LOGIN

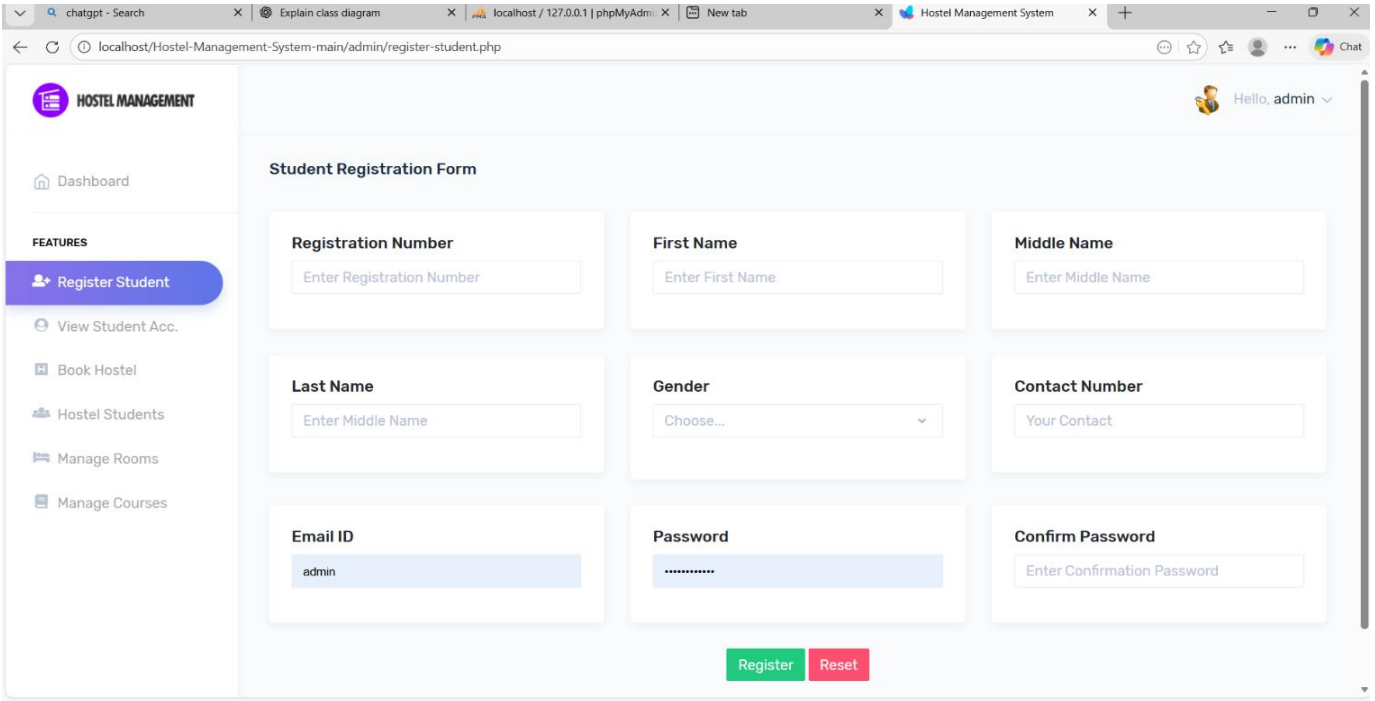

FIGURE B.3 REGISTER STUDENT

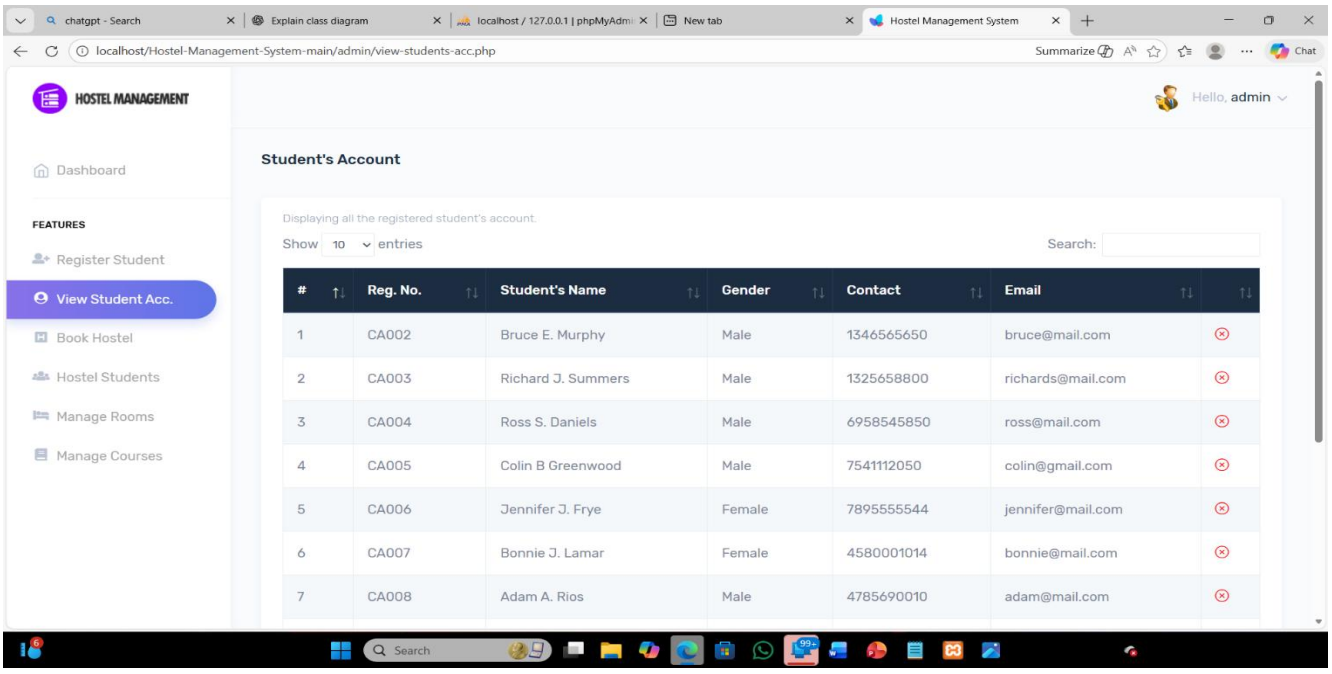

FIGURE B.4 VIEW STUDENT

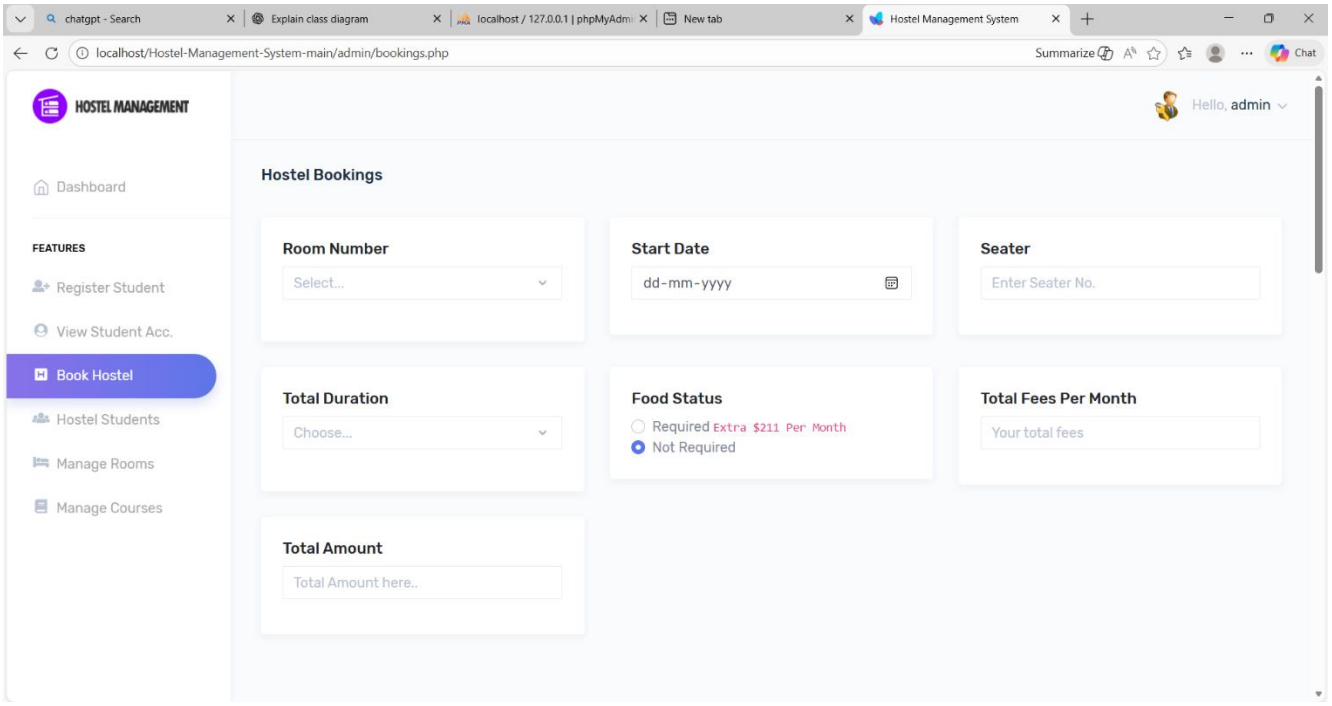

FIGURE B.5 BOOK HOSTEL

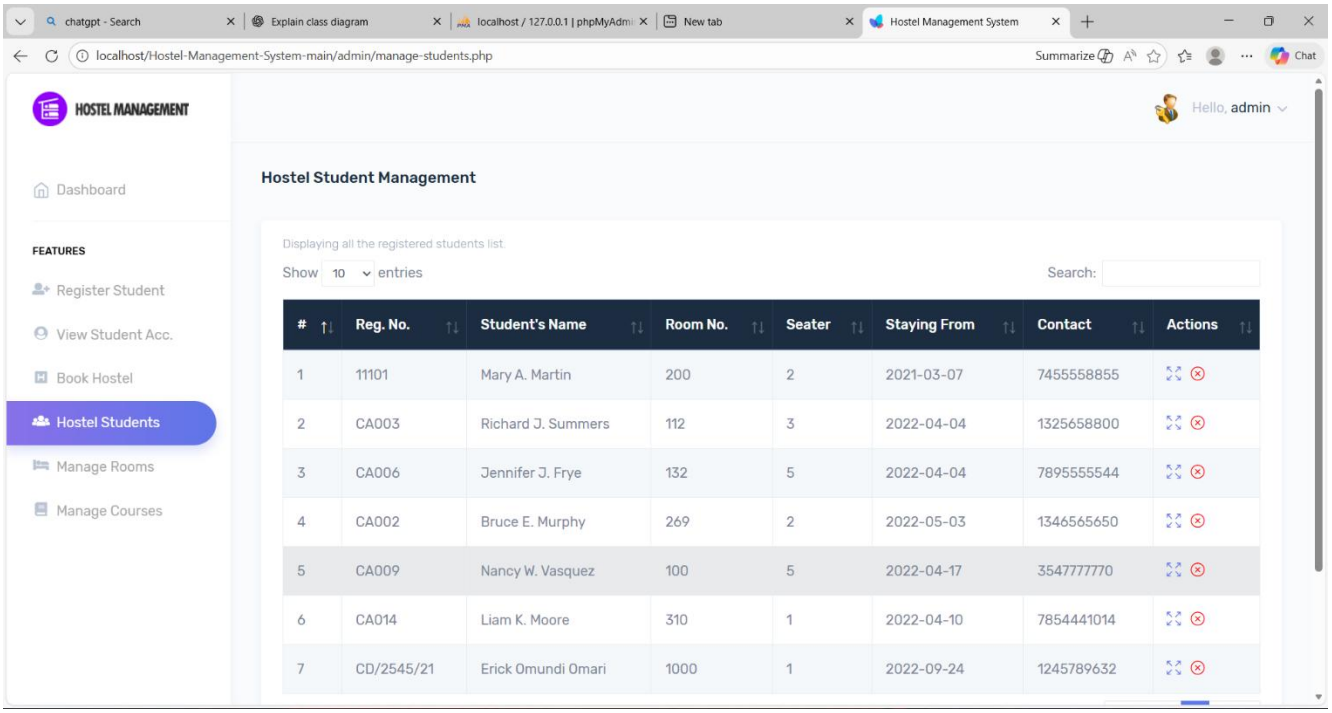

FIGURE B.6 HOSTEL STUDENTS

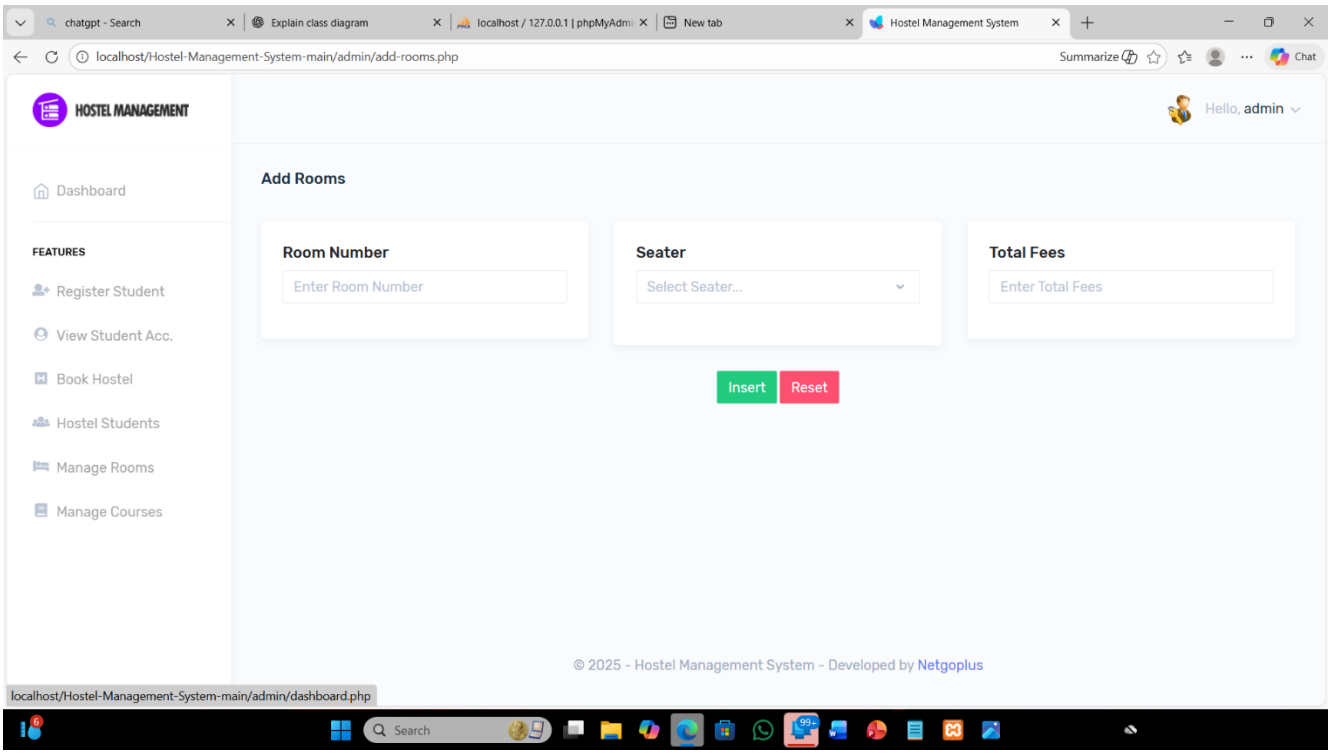

FIGURE B.7 ADD ROOMS

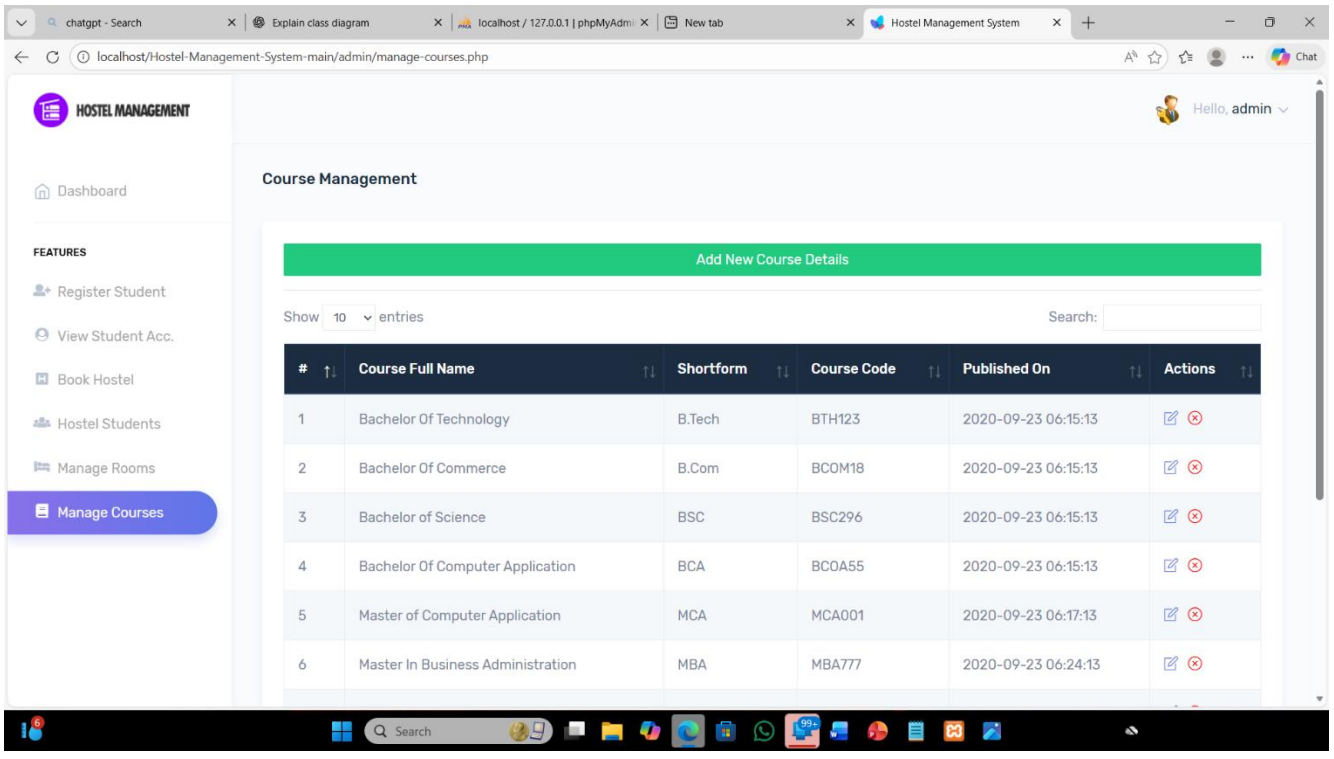

FIGURE B.8 MANAGE COURSES

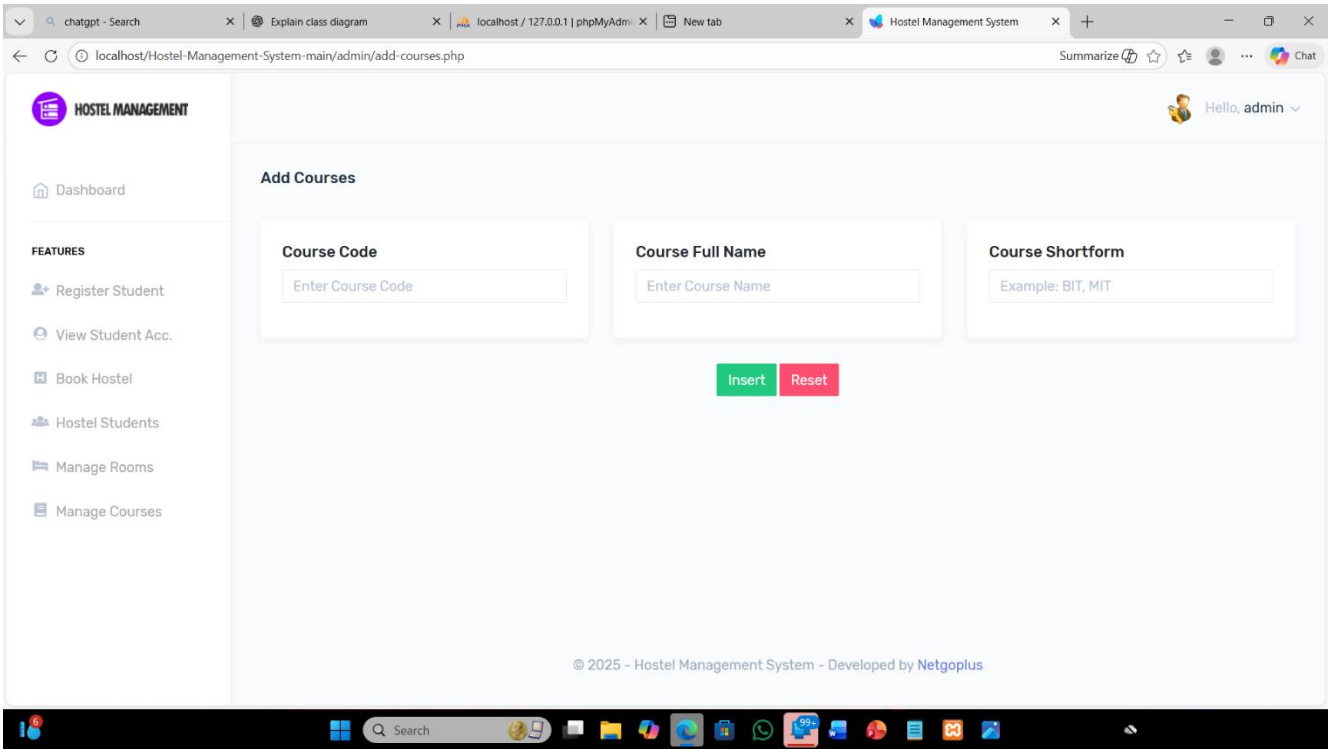

FIGURE B.9 MANAGE ROOMS

REFERENCES

REFERENCES BOOKS:

1. Silberschatz, A., Korth, H. F., & Sudarshan, S.
   Database System Concepts, 6th Edition, McGraw-Hill Education, 2019.

2. Elmasri, R., & Navathe, S. B.
   Fundamentals of Database Systems, 7th Edition, Pearson Education, 2016.

3. Pressman, R. S.
   Software Engineering: A Practitioner's Approach, 8th Edition, McGraw-Hill, 2015.

4. Sommerville, I.
   Software Engineering, 10th Edition, Pearson Education, 2016.