A MINI PROJECT REPORT

On

HOSTEL MANAGEMNET SYSTEM

Submitted by,

K.KAILASH	23J41A66R5
G.SATHWIK	23J41A66L5
M.MANOHAR	23J41A66N5
M.RAKESH	23J41A66P9

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

of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

(ARTIFICIAL INTELLIGENCE & MACHINE LEARNING)

Under the Guidance of

Mr. B. VENU MADHAV

Assistant Professor, Computer Science and Engineering

COMPUTER SCIENCE AND ENGINEERING(AI&ML)

MALLA REDDY ENGINEERING COLLEGE

Maisammaguda, Secunderabad, Telangana, India 500100

APRIL-2025

MALLA REDDY ENGINEERING COLLEGE

Maisammaguda,secunderabad,Telangana,india 500100

BONAFIDE CERTIFICATE

This is to certify that this Real time research project work entitled "**HOSTEL MANAGEMENT SYSTEM**", Submitted by **K.KAILASH(23J41A66R5),G.SATHWIK(23J41A66L5),M.MANOHAR(23J41A66N5),M.RAKESH(23J41A66P9),**to Malla Reddy Engineering College affiliated to JNTUH, Hyderabad in partial fulfillment for the award of **Bachelor of Technology in Computer Science and Engineering** is a bonafide record of project work carried out under my/our supervision during the academic year 2024 – 2025 and that this work has not been submitted elsewhere for a degree

SIGNATURE	**SIGNATURE**
Mr. B. VENU MADHAV	**Dr. U. MOHAN SRINIVAS**
Assistant Professor-CSE(AIML)	**HOD-CSE(AIML)**
Malla Reddy Engineering College	Malla Reddy Engineering College
Secunderabad, 500 100	Secunderabad, 500 100

Submitted for Real Time Project viva-voce examination held on _____

INTERNALEXAMINER **EXTERNAL EXAMINAR**

MALLA REDDY ENGINEERING COLLEGE

Maisammaguda, Secunderabad, Telangana, India 500100

DECLARATION

We hereby declare that the project titled **'HOSTEL MANAGEMENT SYSTEN'**, submitted to Malla Reddy Engineering College (Autonomous) and affiliated with JNTUH, Hyderabad, in partial fulfillment of the requirements for the award of a **Bachelor of Technology in Computer Science and Engineering** , represents my ideas in my own words. Wherever other's ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity, and I have not misrepresented, fabricated, or falsified any idea, data, fact, or source in our submission. I understand that any violation of the above will be a cause for disciplinary action by the Institute. It is further declared that the project report or any part thereof has not been previously submitted to any University or Institute for the award of degree or diploma.

Signature(s)

K.KAILASH	23J41A66R5	_____
G.SATHWIK	23J41A66L5	_____
M.MANOHAR	23J41A66N5	_____
M.RAKESH	23J41A66P9	_____

Secunderabad -500 100

Date:

MALLA REDDY ENGINEERING COLLEGE

Maisammaguda, Secunderabad, Telangana, India 500100

ACKNOWLEDGMENT

We sincerely thank our Principal, **Dr. A. Ramaswami Reddy**, who took a keen interest and encouraged us in every effort during the project work.

We express my heartfelt thanks **to Dr. U. MOHAN SRINIVAS**,Head of department of computer science and engineering, for his kind attention and valuable guidance throughout the project work.

We are thankful to my Project Coordinator _____**, Associate Professor**, Department of Computer Science and Engineering, for his cooperation during the project work.

We also thank all the teaching and non-teaching staff of the Department for their cooperation during the project work.

We also wish to thank all the teaching and non-teaching staff of the department for their valuable cooperation and support throughtout the project.

K.KAILASH 23J41A66R5

G.SATHWIK 23J41A66L5

M.MANOHAR 23J41A66N5

M.RAKESH 23J41A66P9

ABSTRACT

As the name specifies "**HOSTEL MANAGEMENT SYSTEM**" is software developed for managing various activities in the hostel. For the past few years the number of educational institutions is increasing rapidly.

Thereby the number of hostels is also increasing for the accommodation of the students studying in this institution. And hence there is a lot of strain on the person who are running the hostel and software's are not usually used in this context.

This particular project deals with the problems on managing a hostel and avoids the problems which occur when carried manually. Identification of the drawbacks of the existing system leads to the designing of computerized system that will be compatible to the existing system with the system Which is more user friendly and more GUI oriented.

We can improve the efficiency of the system, thus overcome the drawbacks of the existing system.

TABLE OF CONTENTS

CHAPTER -1

INTRODUCTION

The management of hostels in educational institutions has always been a challenging task, especially with the rising number of students seeking accommodation. Traditionally, hostel wardens and administrators relied on manual record-keeping, using registers and paper-based processes to manage student information, room allocation, fee collection, and complaint handling. However, with the increasing student population and the growing demand for efficiency and accuracy, manual systems have proven to be inadequate, time-consuming, and prone to human error.

The need for a streamlined and automated system to manage hostel operations has become more important than ever. In response to these challenges, the **Hostel Management System** has been developed as a comprehensive web-based solution. This system is designed to replace the manual processes with a digital platform where hostel administration and student services can be managed seamlessly. It not only improves operational efficiency but also enhances transparency, reduces paperwork, and minimizes delays in room allotment, grievance redressal, and fee tracking.

The Hostel Management System focuses on providing two distinct interfaces: one for the students and one for the administrators. Students can register, log in, book hostel rooms, update their profiles, and submit feedback or complaints through a user-friendly interface. Administrators can oversee the entire hostel operations, including managing room availability, allocating rooms to students, tracking complaints, managing fees, and viewing system logs. The real-time availability of data ensures that decisions are made quickly and accurately, enhancing the overall experience for both students and hostel management.

Developed using open-source technologies such as **PHP**, **MySQL**, **HTML5**, **CSS3**, and **JavaScript**, the system is cost-effective, scalable, and secure. The backend handles all the application logic and database operations, while the frontend ensures an intuitive and responsive user interface. MySQL serves as the relational database for storing student details, room information, complaints, and booking history. Security measures, such as password hashing and session-based authentication, have been implemented to ensure data protection and system integrity.One of the unique aspects of the system is its modular design. The Hostel Management System is built in a way that future enhancements can be easily incorporated without major

modifications. For example, features such as hostel fee online payments, biometric attendance systems, or even mobile app integrations can be seamlessly added. This flexibility ensures that the system remains relevant and scalable as institutional needs grow and technology evolves.

In addition to operational efficiency, the Hostel Management System promotes transparency and accountability. Students can easily check room availability, monitor their booking status, and submit grievances that are directly accessible to administrators. This reduces the communication gap between students and management and ensures that issues are addressed promptly. The inclusion of detailed access logs also helps administrators monitor system usage, enhancing security and traceability of activities within the hostel environment.

In conclusion, the Hostel Management System is a robust, modern, and practical solution to the challenges faced by educational institutions in managing hostel operations. By automating day-to-day activities and providing real-time access to information, the system not only saves time and reduces manual errors but also improves the quality of service provided to students. The adoption of this system leads to better management practices, increased student satisfaction, and the creation of a smart, technology-driven hostel environment.

CHAPTER 2

LITERATURE SURVEY

In today's educational environment, hostels are no longer just places of residence; they are critical components of the academic ecosystem. With the rapid growth of colleges and universities, particularly in urban and semi-urban areas, the need for efficient and well-organized hostel facilities has become increasingly important. Hostel management involves a variety of operations such as room allotment, fee collection, complaint handling, maintaining student records, and ensuring proper utilization of hostel resources. Traditionally, these operations were handled manually by hostel wardens and administrative staff, relying heavily on paper records and physical verification processes. However, as the number of students increased and expectations for service quality rose, manual systems have proven to be inefficient, error-prone, and unsuitable for managing modern hostels.

The growing reliance on technology across all sectors has driven institutions to seek digital solutions that can automate hostel operations. Initially, small digital adaptations like spreadsheet- based systems and desktop applications were introduced to reduce paperwork and assist with record management. While these early attempts were beneficial in reducing clerical work, they were limited in their capabilities. They lacked essential features like real-time room availability tracking, remote access for students and administrators, secure authentication methods, and integration with other institutional systems. Moreover, these desktop-based systems often depended on local machines, leading to data security risks and lack of accessibility outside campus premises.

With the advent of internet technologies and open-source development tools, there has been a significant shift towards web-based hostel management solutions. These modern systems utilize technologies such as PHP, MySQL, HTML5, and JavaScript to deliver powerful, interactive, and secure applications. Web-based platforms allow multiple users to interact with the system simultaneously, whether they are hostel administrators or students, ensuring real-time updates and efficient management of hostel activities. Furthermore, these systems are designed to be modular and scalable, allowing institutions to expand features based on their specific requirements without needing to overhaul the entire system.

The evolution of hostel management systems is also being influenced by broader trends in software engineering and campus automation. Institutions are increasingly looking for systems that can integrate with biometric access control, digital fee payment platforms, mobile applications, and cloud storage solutions. Research has shown that fully integrated hostel

management systems not only improve operational efficiency but also significantly enhance student satisfaction and trust in institutional processes. Effective hostel management contributes

Therefore, in the context of increasing student populations, growing technological adoption, and heightened expectations for service quality, it is essential to design a hostel management system that is robust, scalable, secure, and user-friendly. The proposed Hostel Management System addresses the gaps identified in traditional and semi-automated systems by offering a comprehensive, real-time, and accessible platform to manage hostel operations efficiently. This project is a step forward in leveraging technology to enhance institutional management and ensure a better residential experience for students.

Existing Hostel Management Systems

Many institutions continue to rely on manual processes such as paper-based records or simple spreadsheet management for hostel operations. This includes Room allocation,Complaint logging,Fee payment tracking,Visitor records.Attendance and discipline management.Some institutions have migrated to basic desktop-based applications or client-server models built using technologies like VB.NET, Java, and Microsoft Access. However, these solutions often lack scalability, data integrity, and remote accessibility.

Modern Web-Based Approaches

Recent studies and implementations emphasize the use of web-based platforms to enhance system availability and usability. Technologies such as PHP, MySQL, HTML5, and JavaScript enable the creation of responsive and robust hostel management systems.

Key features found in modern systems include:

- Role-based access for admin, wardens, and students.
- Real-time seat availability.
- Complaint and feedback management.
- Online payment gateways.
- Auto-generated reports for audits and inspections.

Web-based HMS examples include systems built with frameworks like Laravel, Django, or MERN stack, offering enhanced scalability, cloud support, and API integration.

Research Trends and Opportunities

Several academic papers and case studies suggest enhancements for hostel systems such as:

- **Integration with Biometric/RFID** for attendance.
- **Mobile apps** for better student engagement.

CHAPTER-3

SYSTEM ANALYSIS

3.1 Existing system

In the existing manual hostel management process, room allocations, student registrations, fee collection, and grievance handling are managed through physical registers or Excel sheets. Whenever a student needs accommodation, the process involves filling out manual forms, verifying documents, and allotting rooms through personal meetings. Similarly, complaints are recorded in complaint books, and their status tracking is often delayed or lost due to poor management practices.

Some institutions have tried to upgrade by using standalone desktop applications. However, these systems lack real-time data access, multi-user support, mobile accessibility, and often suffer from data loss risks due to system crashes or improper backup mechanisms. Existing System Disadvantages

Manual Dependency: Processes like room allotment or complaint handling are time- consuming and error-prone.

Lack of Transparency: Students often remain unaware of room availability or complaint status.

No Real-Time Data: Updates to room occupancy, fee status, or student records are not reflected in real time.

Poor Accessibility: Systems are not mobile-optimized or accessible outside the institutional premises.

3.2 Proposed System

student registrations, room allocations, fee tracking, complaint handling, and administrative reporting. Students can register, view available rooms, book accommodations, update their profiles, and file complaints through a user-friendly interface accessible from desktops or mobile devices. Simultaneously, hostel administrators can manage hostel facilities, monitor student bookings, handle grievances, and generate reports through a secure, intuitive admin dashboard. The system ensures real-time data updates, strong role-based access control, and protection of sensitive information through encrypted storage and secured sessions.

Developed using PHP, MySQL, HTML5, CSS3, and JavaScript, the system is cost-effective, scalable, and modular for future enhancements such as mobile app integration, biometric access,

and online payment gateways. It minimizes human errors, improves operational transparency, and significantly reduces the administrative burden. By offering real-time visibility into room availability, faster booking approvals, and efficient complaint resolution, the proposed Hostel Management System aims to create a modern, technology-driven residential experience that benefits both students and institutional management.

3.3 Proposed System Advantages

Automation of Manual Processes : Eliminates the need for physical paperwork.

Real-Time Room Availability : Students can instantly check available rooms and make bookings.

Centralized Data Management : All student data, room records, complaints, and feedback are stored in a single database.

Role-Based Access Control : Secure login system ensures only authorized users can access certain features.

Efficient Complaint and Feedback Handling : Students can submit complaints and suggestions online.

By implementing this proposed system, institutions can achieve improved operational efficiency, better student satisfaction, and more organized hostel management

CHAPTER-4

SYSTEM DESGIN

4.1 Input design

The system design is divided in to two portions. The Administrator section and the User (student's) section.

Administrator

1. The Administrator can allot different students to the different hostels.

2. He can vacate the students for the hostels.

3. He can control the status of the fee payement.

4. He can edit the details of the students.He can change their rooms, edit and delete the student records.

A process of converting user originated inputs to a computer-based format. Input design is an important part of development process since inaccurate input data are the most common cause of errors in data processing. Erroneous entries can be controlled by input design. It consists of developing specifications and procedures for entering data into a system and must be in simple format. The goal of input data design is to make data entry as easy, logical and free from errors as possible. In input data design, we design the source document that capture the data and then select the media used to enter them into the computer.

There are two major approaches for entering data in to the computer. They are

- Menus.
- Dialog Boxes.

Menus

A menu is a selection list that simplifies computer data access or entry. Instead of remembering what to enter, the user chooses from a list of options. A menu limits a user choice of response but reduce the chances for error in data entry.

Dialog Box

Dialog boxes are windows and these windows are mainly popup, which appear in response to certain conditions that occur when a program is run. It allows the display of bitmaps and pictures. It can have various controls like buttons, text boxes, list boxes and combo boxes. Using these controls we can make a 'dialog' with the program.

The proposed system has three major inputs. They are Machine Registration, Machine Scheduling

and Request Form.

4.2 Process Design

Process design plays an important role in project development. In order to understand the working procedure, process design is necessary. Data Flow Diagram and System Flow chart are the tools used for process design.

System Flow Chart is a graphical representation of the system showing the overall flow of control in processing at the job level; specifies what activities must be done to convert from a physical to logical model.

Data Flow Diagram is the logical representation of the data flow of the project. The DFD is drawn using various symbols. It has a source and a destination. The process is represented using circles and source and destination are represented using squares. The data flow is represented using arrows.

One reader can easily get the idea about the project through Data Flow Diagram

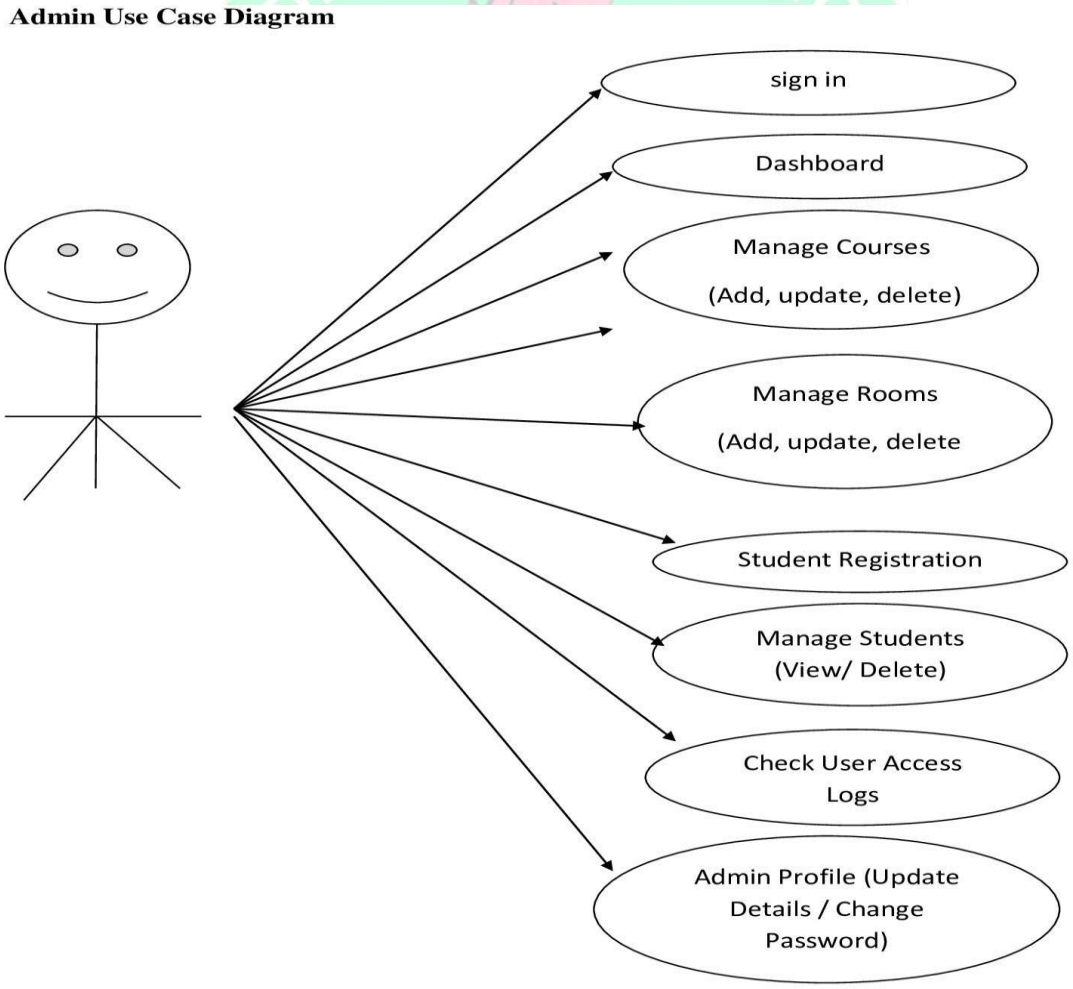


Fig-4.2.1

User Use Case Diagram

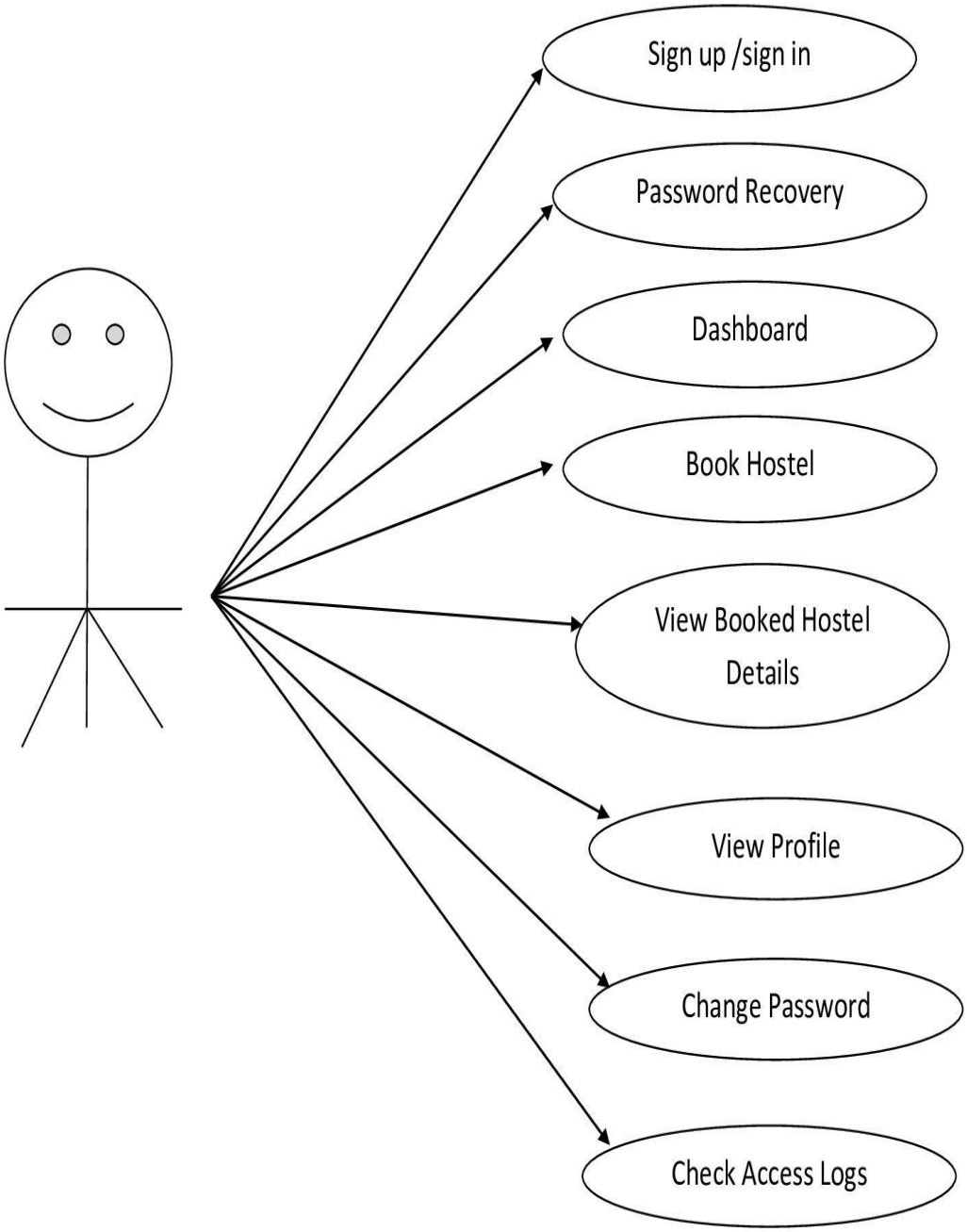


Fig-4.2.2

4.3 Database Design

The data in the system has to be stored and retrieved from database. Designing the database is part of system design. Data elements and data structures to be stored have been identified at analysis stage. They are structured and put together to design the data storage and retrieval system.

A database is a collection of interrelated data stored with minimum redundancy to serve many users quickly and efficiently. The general objective is to make database access easy, quick, inexpensive and flexible for the user. Relationships are established between the data items and unnecessary data items are removed. Normalization is done to get an internal consistency of data and to have minimum redundancy and maximum stability. This ensures minimizing data storage required, minimizing chances of data inconsistencies and optimizing for updates. The MySQL database has been chosen for developing the relevant databases

DATA TABLE RELATIONSHIP

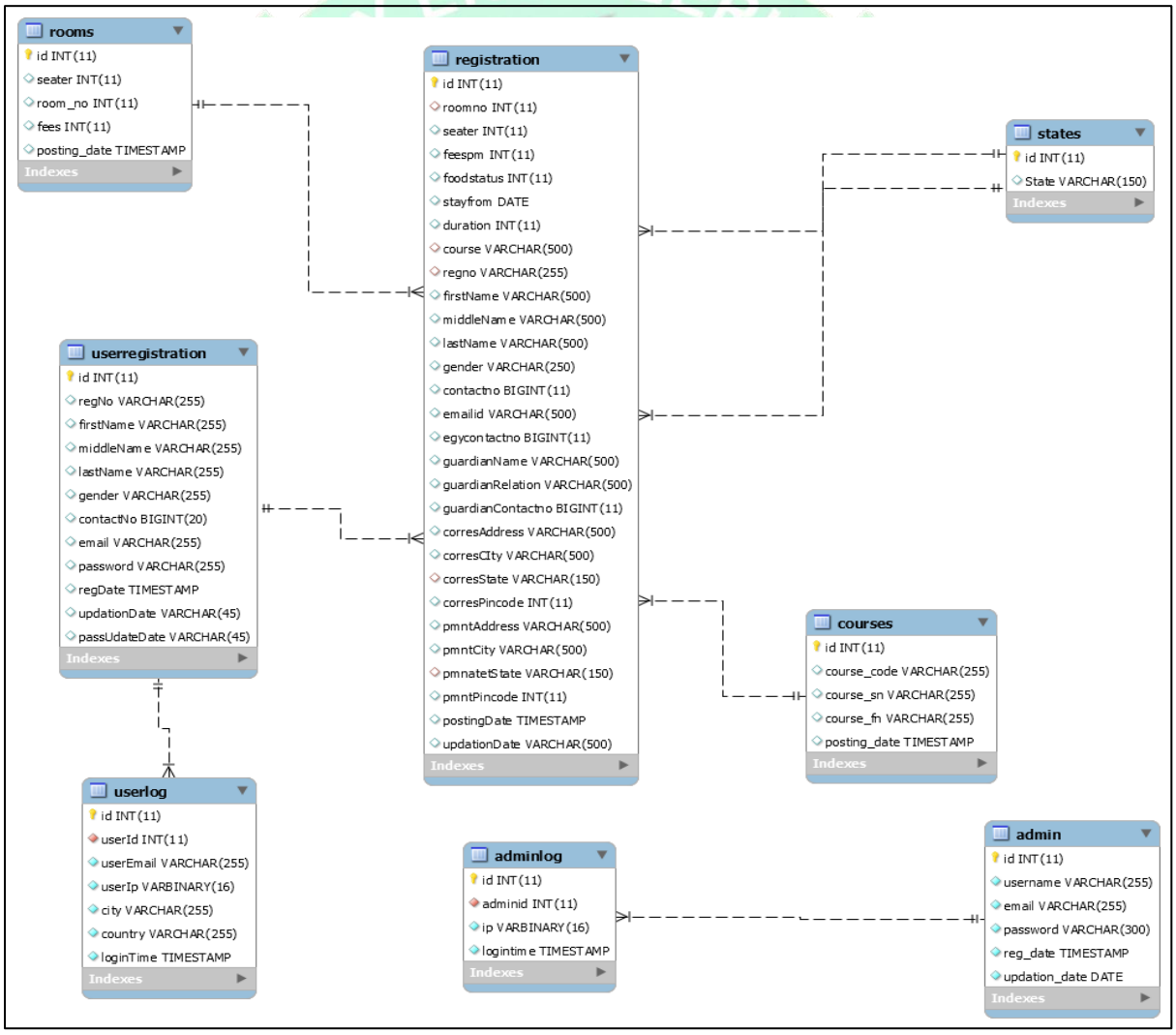


Fig 4.3.1

4.4 Output Design

Designing computer output should proceed in an organized, well throughout manner; the right output element is designed so that people will find the system whether or executed. When we design an output we must identify the specific output that is needed to meet the system. The usefulness of the new system is evaluated on the basis of their output.

Once the output requirements are determined, the system designer can decide what to include in the system and how to structure it so that the require output can be produced. For the proposed software, it is necessary that the output reports be compatible in format with the existing reports. The output must be concerned to the overall performance and the system's working, as it should. It consists of developing specifications and procedures for data preparation, those steps necessary to put the inputs and the desired output, ie maximum user friendly. Proper messages and appropriate directions can control errors committed by users.

The output design is the key to the success of any system. Output is the key between the user and the sensor. The output must be concerned to the system's working, as it should.

Output design consists of displaying specifications and procedures as data presentation. User never left with the confusion as to what is happening without appropriate error and acknowledges message being received. Even an unknown person can operate the system without knowing anything about the system.

CHAPTER-5

SYSTEM REQUIREMENTS

5.1 Hardware Requirements

The hardware components required for the successful deployment and operation of the Hostel Management System are as follows:

Minimum Configuration

- Processor: Intel Pentium IV or equivalent
- RAM: 512 MB
- Hard Disk: 40 GB
- Monitor: 1024 × 768 Resolution Color Monitor
- Input Devices: Standard Keyboard and Mouse
- Network: LAN/Wi-Fi connection (for intranet or internet use)

Recommended Configuration

- Processor: Intel i3 / i5 (2.0 GHz or higher)
- RAM: 4 GB or higher
- Hard Disk: 250 GB HDD / SSD for faster access
- Monitor: LED monitor with Full HD resolution
- Network: High-speed Internet with a stable connection

These configurations ensure smooth performance during local development or small-scale deployment of the system within an educational institution.

5.2 Software Requirements

The software stack used for developing and running the Hostel Management System consists of both open-source and widely supported platforms:

Operating System

- Windows XP / 7 / 10 (For local development)
- Linux (Ubuntu/CentOS) (Recommended for production server)

Server and Development Tools

- PHP Triad Bundle (Includes Apache, PHP 5.6, MySQL, phpMyAdmin)
- XAMPP / WAMP / LAMP as an alternative stack

Languages & Technologies

- Frontend: HTML, CSS, JavaScript
- Backend: PHP (Hypertext Preprocessor)
- Database: MySQL
- Web Server: Apache HTTP Server
- Admin Interface: phpMyAdmin (for MySQL administration)

Other Tools (Optional but Useful)

- Text Editor/IDE: VS Code, Sublime Text, or Notepad++
- Version Control: Git (for code management)
- Browsers: Google Chrome, Mozilla Firefox (for testing and usage)

CHAPTER-6

IMPLEMENTATION

6.1 Modules

The Hostel Management System is structured into multiple modules to ensure clarity, modular development, and scalability. Each module is responsible for a specific functionality and collectively contributes to a seamless user and admin experience.

1. User Authentication Module

Description:

Handles registration, login, session management, and password recovery for both users (students) and administrators.

Key Features:

- User sign-up and sign-in interface
- Password hashing for secure login
- Session tracking and logout functionality
- Forgot password functionality via security verification

2. Room Management Module

Description:

Administers room details including room creation, seater configuration, and room availability tracking.

Key Features:

- Add/edit/delete rooms
- Set number of seaters and associated fee
- Mark availability dynamically upon student booking
- Display available rooms to users in real-time

3. Student Registration and Booking Module

Description:

Allows students to register, update their profiles, and book rooms.

Key Features:

- New student registration form with validation
- Admin-side room allotment to students
- Booked room details display in user dashboard
- Profile update and change password options

4. Complaint and Feedback Module

Description:

Enables users to lodge complaints or suggestions, which the admin can view and resolve.

Key Features:

- Online complaint form submission
- Feedback rating or suggestion box
- Admin dashboard to monitor and respond to issues

5. Access Logs and Audit Module

Description:

Logs and displays login sessions and actions performed by users.

Key Features:

- Displays last login time and session duration
- Admin can view system access logs of users
- Useful for activity monitoring and security audits

6. Admin Dashboard Module

Description:

The control panel for administrators to manage students, rooms, and all back-end processes.

Key Features:

- Admin login and profile management
- Manage room and course entries
- View/delete registered students
- View feedback, complaints, and access logs
- Password update and recovery tools

7. Module Integration and Testing

After developing each module individually:

- **Unit Testing** was performed to ensure standalone module functionality.
- **Integration Testing** was conducted to verify that modules worked in coordination.
- **User Acceptance Testing** was carried out with test users (students and faculty) to validate practical usability.

8. Deployment Process

The following steps were followed during implementation:

1. **Local Environment Setup**:
    - Installed XAMPP with Apache and MySQL
    - Copied the source code into htdocs/hostel/ directory
    - Created and imported the hostel.sql database via phpMyAdmin
2. **System Launch**:
    - The application was accessed locally via http://localhost/hostel/
    - Admin and student accounts were created for demonstration
3. **Documentation and Demo**:
    - Prepared user and admin manuals
    - Conducted live demonstration with real-time booking and complaint filing

9. Benefits of Implementation

- Automation of time-consuming manual processes
- Real-time room availability and student data access
- Secure login system with role-based control
- Organized complaint handling and monitoring
- Easy future scalability with modular design

Login Page Php Code:-

```
<?php

session_start();

include('includes/config.php');
```

```
if(isset($_POST['login']))

{

$email=$_POST['email'];

$password=$_POST['password'];

$stmt=$mysqli->prepare("SELECT email,password,id FROM userregistration WHERE
email=? and password=? ");

        $stmt->bind_param('ss',$email,$password);

        $stmt->execute();

        $stmt -> bind_result($email,$password,$id);

        $rs=$stmt->fetch();

        $stmt->close();

        $_SESSION['id']=$id;

        $_SESSION['login']=$email;

        $uip=$_SERVER['REMOTE_ADDR'];

        $ldate=date('d/m/Y h:i:s', time());

        if($rs)

        {

      $uid=$_SESSION['id'];

      $uemail=$_SESSION['login'];

$ip=$_SERVER['REMOTE_ADDR'];

$geopluginURL='http://www.geoplugin.net/php.gp?ip='.$ip;

$addrDetailsArr = unserialize(file_get_contents($geopluginURL));

$city = $addrDetailsArr['geoplugin_city'];

$country = $addrDetailsArr['geoplugin_countryName'];

$log="insert into userLog(userId,userEmail,userIp,city,country)
values('$uid','$uemail','$ip','$city','$country')";

$mysqli->query($log);

if($log)
```

```
{
header("location:dashboard.php");
            }
}
            else
            {
                echo "<script>alert('Invalid Username/Email or password');</script>";
            }
          }
            ?>


<!doctype html>
<html lang="en" class="no-js">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1, minimum-scale=1,
maximum-scale=1">
    <meta name="description" content="">
    <meta name="author" content="">
    <meta name="theme-color" content="#3e454c">
    <title>Student Hostel Registration</title>
    <link rel="stylesheet" href="css/font-awesome.min.css">
    <link rel="stylesheet" href="css/bootstrap.min.css">
    <link rel="stylesheet" href="css/dataTables.bootstrap.min.css">>
    <link rel="stylesheet" href="css/bootstrap-social.css">
    <link rel="stylesheet" href="css/bootstrap-select.css">
```

```
    <link rel="stylesheet" href="css/fileinput.min.css">
    <link rel="stylesheet" href="css/awesome-bootstrap-checkbox.css">
    <link rel="stylesheet" href="css/style.css">
<script type="text/javascript" src="js/jquery-1.11.3-jquery.min.js"></script>
<script type="text/javascript" src="js/validation.min.js"></script>
<script type="text/javascript" src="http://code.jquery.com/jquery.min.js"></script>
<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>
    <?php include('includes/header.php');?>
    <div class="ts-main-content">
        <?php include('includes/sidebar.php');?>
        <div class="content-wrapper">
            <div class="container-fluid">
                <div class="row">
                  <div class="col-md-12">
```

```
<h2 class="page-title">User Login </h2>
<div class="row">
<div class="col-md-6 col-md-offset-3">
<div class="well row pt-2x pb-3x bk-light">
<div class="col-md-8 col-md-offset-2">

<form action="" class="mt" method="post">
<label for="" class="text-uppercase text-sm">Email</label>
<input type="text" placeholder="Email" name="email" class="form-control mb">
<label for="" class="text-uppercase text-sm">Password</label>
<input type="password" placeholder="Password" name="password" class="form-control mb">

<input type="submit" name="login" class="btn btn-primary btn-block" value="login" >
</form>
</div>
</div>
<div class="text-center text-light">
<a href="forgot-password.php" class="text-light">Forgot password?</a>
</div>
</div>
</div>
</div>
</div>
```

```
                    </div>
                </div>
            </div>
        </div>
    </div>
    <script src="js/jquery.min.js"></script>
    <script src="js/bootstrap-select.min.js"></script>
    <script src="js/bootstrap.min.js"></script>
    <script src="js/jquery.dataTables.min.js"></script>
    <script src="js/dataTables.bootstrap.min.js"></script>
    <script src="js/Chart.min.js"></script>
    <script src="js/fileinput.js"></script>
    <script src="js/chartData.js"></script>
    <script src="js/main.js"></script>
</body>


</html>
```

SQL Code:-

```
-- phpMyAdmin SQL Dump
-- version 5.2.1
-- https://www.phpmyadmin.net/
--
-- Host: 127.0.0.1
-- Generation Time: Jan 14, 2025 at 12:14 PM
-- Server version: 10.4.32-MariaDB
```

```
-- PHP Version: 8.2.12


SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";

START TRANSACTION;

SET time_zone = "+00:00";


/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;

/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS
*/;

/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;

/*!40101 SET NAMES utf8mb4 */;


--

-- Database: `hostel`

--


- - ----------------------------------------------------------


--

-- Table structure for table `admin`

--


CREATE TABLE `admin` (

  `id` int(11) NOT NULL,

  `username` varchar(255) NOT NULL,

  `email` varchar(255) NOT NULL,

  `password` varchar(300) NOT NULL,
```

```
  `reg_date` timestamp NOT NULL DEFAULT current_timestamp(),
  `updation_date` date NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1_swedish_ci;

--
-- Dumping data for table `admin`
--

INSERT INTO `admin` (`id`, `username`, `email`, `password`, `reg_date`, `updation_date`)
VALUES
(1, 'admin', 'admin@gmail.com', 'Test@1234', '2024-01-31 20:31:45', '2024-01-01');

--------------------------------------------------------

--
-- Table structure for table `adminlog`
--

CREATE TABLE `adminlog` (
  `id` int(11) NOT NULL,
  `adminid` int(11) NOT NULL,
  `ip` varbinary(16) NOT NULL,
  `logintime` timestamp NOT NULL DEFAULT current_timestamp()
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1_swedish_ci;

--------------------------------------------------------
```

```
--
-- Table structure for table `complainthistory`
--

CREATE TABLE `complainthistory` (
  `id` int(11) NOT NULL,
  `complaintid` int(11) DEFAULT NULL,
  `compalintStatus` varchar(255) DEFAULT NULL,
  `complaintRemark` mediumtext DEFAULT NULL,
  `postingDate` timestamp NULL DEFAULT current_timestamp()
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;

--
-- Dumping data for table `complainthistory`
--

INSERT INTO `complainthistory` (`id`, `complaintid`, `compalintStatus`, `complaintRemark`, `postingDate`) VALUES
(1, 1, 'In Process', 'Electrician assigned.', '2025-01-14 11:11:12'),
(2, 1, 'Closed', 'Switch changed', '2025-01-14 11:11:31');

--------------------------------------------------------

--
-- Table structure for table `complaints`
--
```

```
CREATE TABLE `complaints` (
  `id` int(11) NOT NULL,
  `ComplainNumber` bigint(12) DEFAULT NULL,
  `userId` int(11) DEFAULT NULL,
  `complaintType` varchar(255) DEFAULT NULL,
  `complaintDetails` mediumtext DEFAULT NULL,
  `complaintDoc` varchar(255) DEFAULT NULL,
  `complaintStatus` varchar(255) DEFAULT NULL,
  `registrationDate` timestamp NULL DEFAULT current_timestamp()
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;

--
-- Dumping data for table `complaints`
--

INSERT INTO `complaints` (`id`, `ComplainNumber`, `userId`, `complaintType`,
`complaintDetails`, `complaintDoc`, `complaintStatus`, `registrationDate`) VALUES
(1, 389685413, 5, 'Electrical', 'Switch not working.', NULL, 'Closed', '2025-01-14 11:10:24');

--
-- Table structure for table `courses`
--

CREATE TABLE `courses` (
  `id` int(11) NOT NULL,
```

```
  `course_code` varchar(255) DEFAULT NULL,
  `course_sn` varchar(255) DEFAULT NULL,
  `course_fn` varchar(255) DEFAULT NULL,
  `posting_date` timestamp NULL DEFAULT current_timestamp()
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1_swedish_ci;

--
-- Dumping data for table `courses`
--

INSERT INTO `courses` (`id`, `course_code`, `course_sn`, `course_fn`, `posting_date`)
VALUES
(1, 'B10992', 'B.Tech', 'Bachelor  of Technology', '2025-01-01 19:31:42'),
(2, 'BCOM1453', 'B.Com', 'Bachelor Of commerce ', '2025-01-01 19:31:42'),
(3, 'BSC12', 'BSC', 'Bachelor  of Science', '2025-01-01 19:31:42'),
(4, 'BC36356', 'BCA', 'Bachelor Of Computer Application', '2025-01-01 19:31:42'),
(5, 'MCA565', 'MCA', 'Master of Computer Application', '2025-01-01 19:31:42'),
(6, 'MBA75', 'MBA', 'Master of Business Administration', '2025-01-01 19:31:42'),
(7, 'BE765', 'BE', 'Bachelor of Engineering', '2025-01-01 19:31:42');

-- --------------------------------------------------------

--
-- Table structure for table `feedback`
--

CREATE TABLE `feedback` (
```

```
  `id` int(11) NOT NULL,
  `AccessibilityWarden` varchar(255) DEFAULT NULL,
  `AccessibilityMember` varchar(255) DEFAULT NULL,
  `RedressalProblem` varchar(255) DEFAULT NULL,
  `Room` varchar(255) DEFAULT NULL,
  `Mess` varchar(255) DEFAULT NULL,
  `HostelSurroundings` varchar(255) DEFAULT NULL,
  `OverallRating` varchar(255) DEFAULT NULL,
  `FeedbackMessage` varchar(255) DEFAULT NULL,
  `userId` int(11) DEFAULT NULL,
  `postinDate` timestamp NULL DEFAULT current_timestamp()
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;

--
-- Dumping data for table `feedback`
--

INSERT INTO `feedback` (`id`, `AccessibilityWarden`, `AccessibilityMember`,
`RedressalProblem`, `Room`, `Mess`, `HostelSurroundings`, `OverallRating`,
`FeedbackMessage`, `userId`, `postinDate`) VALUES
(1, 'Very Good', 'Very Good', 'Excellent', 'Very Good', 'Excellent', 'Average', 'Good', 'NA', 5,
'2025-01-14 11:12:43');

--------------------------------------------------------------------

--
-- Table structure for table `registration`
--
```

```
CREATE TABLE `registration` (
  `id` int(11) NOT NULL,
  `roomno` int(11) DEFAULT NULL,
  `seater` int(11) DEFAULT NULL,
  `feespm` int(11) DEFAULT NULL,
  `foodstatus` int(11) DEFAULT NULL,
  `stayfrom` date DEFAULT NULL,
  `duration` int(11) DEFAULT NULL,
  `course` varchar(500) DEFAULT NULL,
  `regno` int(11) DEFAULT NULL,
  `firstName` varchar(500) DEFAULT NULL,
  `middleName` varchar(500) DEFAULT NULL,
  `lastName` varchar(500) DEFAULT NULL,
  `gender` varchar(250) DEFAULT NULL,
  `contactno` bigint(11) DEFAULT NULL,
  `emailid` varchar(500) DEFAULT NULL,
  `egycontactno` bigint(11) DEFAULT NULL,
  `guardianName` varchar(500) DEFAULT NULL,
  `guardianRelation` varchar(500) DEFAULT NULL,
  `guardianContactno` bigint(11) DEFAULT NULL,
  `corresAddress` varchar(500) DEFAULT NULL,
  `corresCIty` varchar(500) DEFAULT NULL,
  `corresState` varchar(500) DEFAULT NULL,
  `corresPincode` int(11) DEFAULT NULL,
  `pmntAddress` varchar(500) DEFAULT NULL,
  `pmntCity` varchar(500) DEFAULT NULL,
```

```
  `pmnatetState` varchar(500) DEFAULT NULL,
  `pmntPincode` int(11) DEFAULT NULL,
  `postingDate` timestamp NULL DEFAULT current_timestamp(),
  `updationDate` varchar(500) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1_swedish_ci;

--
-- Dumping data for table `registration`
--

INSERT INTO `registration` (`id`, `roomno`, `seater`, `feespm`, `foodstatus`, `stayfrom`,
`duration`, `course`, `regno`, `firstName`, `middleName`, `lastName`, `gender`, `contactno`,
`emailid`, `egycontactno`, `guardianName`, `guardianRelation`, `guardianContactno`,
`corresAddress`, `corresCIty`, `corresState`, `corresPincode`, `pmntAddress`, `pmntCity`,
`pmnatetState`, `pmntPincode`, `postingDate`, `updationDate`) VALUES
(2, 100, 5, 8000, 1, '2025-01-06', 6, 'Bachelor  of Technology', 10806121, 'Anuj', '', 'kumar',
'male', 1234567890, 'ak@gmail.com', 1236547890, 'ABC', 'XYZ', 98756320000, 'ABC 12345
XYZ Street', 'New Delhi', 'Delhi (NCT)', 110001, 'ABC 12345 XYZ Street', 'New Delhi', 'Delhi
(NCT)', 110001, '2025-01-03 14:58:26', NULL),
(3, 200, 2, 6000, 1, '2025-01-10', 12, 'Bachelor  of Science', 108061233, 'John', '', 'Doe', 'male',
1425362514, 'hohn@gmail.com', 456312058, 'Alex Doe', 'father', 1234567890, '123 Xyz
apartment ', 'New Delhi', 'Delhi (NCT)', 110001, '123 Xyz apartment ', 'New Delhi', 'Delhi
(NCT)', 110001, '2025-01-03 14:58:26', NULL),
(4, 200, 2, 6000, 0, '2025-01-08', 9, 'Bachelor Of commerce ', 10806121, 'Anuj', '', 'kumar',
'male', 1234567890, 'test@gmail.com', 546456546, 'ytrrtyrt', 'yrtyrty', 46456456, 'ttyrytryr',
'yrty', 'Andhra Pradesh', 123123, 'ttyrytryr', 'yrty', 'Andhra Pradesh', 123123, '2025-01-03
14:58:26', NULL),
(5, 100, 5, 8000, 1, '2025-01-15', 3, 'Bachelor  of Technology', 14563213, 'John', '', 'Matthew ',
'male', 8956237845, 'john@gmail.com', 7845123698, 'Mrs. Jacob Mattew', 'Uncle',
5623894178, 'H-899, Gauri Apartment', 'Kanpur', 'Uttar Pradesh', 551007, 'H-899, Gauri
Apartment', 'Kanpur', 'Uttar Pradesh', 551007, '2025-01-03 14:58:26', NULL),
```

(6, 132, 5, 2000, 1, '2025-01-15', 12, 'Bachelor of Technology', 14563213, 'Amit', 'kumar', 'Singh', 'male', 9632587412, 'amit123@gmail.com', 8563145621, 'Ram Kumar Singh', 'Father', 4563245631, 'Hno 181/1 Mayur Vihar ', 'New Delhi', 'Delhi (NCT)', 110092, 'Hno 181/1 Mayur Vihar ', 'New Delhi', 'Delhi (NCT)', 110092, '2025-01-03 14:58:26', NULL);

--

-- Table structure for table `rooms`

--

CREATE TABLE `rooms` (

`id` int(11) NOT NULL,

`seater` int(11) DEFAULT NULL,

`room_no` int(11) DEFAULT NULL,

`fees` int(11) DEFAULT NULL,

`posting_date` timestamp NULL DEFAULT current_timestamp()

) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1_swedish_ci;

--

-- Dumping data for table `rooms`

--

INSERT INTO `rooms` (`id`, `seater`, `room_no`, `fees`, `posting_date`) VALUES

(1, 5, 100, 8000, '2025-01-01 22:45:43'),

(2, 2, 201, 6000, '2025-01-01 22:45:43'),

(3, 2, 200, 6000, '2025-01-01 22:45:43'),

(4, 3, 112, 4000, '2025-01-01 22:45:43'),

(5, 5, 132, 2000, '2025-01-01 22:45:43'),

(6, 3, 145, 3000, '2025-01-01 22:45:43');

--

--

-- Table structure for table `states`

--

CREATE TABLE `states` (

`id` int(11) NOT NULL,

`State` varchar(150) DEFAULT NULL

) ENGINE=MyISAM DEFAULT CHARSET=latin1 COLLATE=latin1_swedish_ci;

--

-- Dumping data for table `states`

--

INSERT INTO `states` (`id`, `State`) VALUES

(1, 'Andaman and Nicobar Island (UT)'),

(2, 'Andhra Pradesh'),

(3, 'Arunachal Pradesh'),

(4, 'Assam'),

(5, 'Bihar'),

(6, 'Chandigarh (UT)'),

(7, 'Chhattisgarh'),

(8, 'Dadra and Nagar Haveli (UT)'),

(9, 'Daman and Diu (UT)'),

(10, 'Delhi (NCT)'),

(11, 'Goa'),

(12, 'Gujarat'),

(13, 'Haryana'),

(14, 'Himachal Pradesh'),

(15, 'Jammu and Kashmir'),

(16, 'Jharkhand'),

(17, 'Karnataka'),

(18, 'Kerala'),

(19, 'Lakshadweep (UT)'),

(20, 'Madhya Pradesh'),

(21, 'Maharashtra'),

(22, 'Manipur'),

(23, 'Meghalaya'),

(24, 'Mizoram'),

(25, 'Nagaland'),

(26, 'Odisha'),

(27, 'Puducherry (UT)'),

(28, 'Punjab'),

(29, 'Rajastha'),

(30, 'Sikkim'),

(31, 'Tamil Nadu'),

(32, 'Telangana'),

(33, 'Tripura'),

(34, 'Uttarakhand'),

(35, 'Uttar Pradesh'),

(36, 'West Bengal');

-- --

--

-- Table structure for table `userlog`

--

CREATE TABLE `userlog` (

`id` int(11) NOT NULL,

`userId` int(11) NOT NULL,

`userEmail` varchar(255) NOT NULL,

`userIp` varbinary(16) NOT NULL,

`city` varchar(255) NOT NULL,

`country` varchar(255) NOT NULL,

`loginTime` timestamp NOT NULL DEFAULT current_timestamp()

) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1_swedish_ci;

--

-- Dumping data for table `userlog`

--

INSERT INTO `userlog` (`id`, `userId`, `userEmail`, `userIp`, `city`, `country`, `loginTime`) VALUES

(1, 5, 'john@gmail.com', 0x3132372e302e302e31, '', '', '2025-01-14 11:09:34');

-- --

```
--
-- Table structure for table `userregistration`
--

CREATE TABLE `userregistration` (
  `id` int(11) NOT NULL,
  `regNo` varchar(255) DEFAULT NULL,
  `firstName` varchar(255) DEFAULT NULL,
  `middleName` varchar(255) DEFAULT NULL,
  `lastName` varchar(255) DEFAULT NULL,
  `gender` varchar(255) DEFAULT NULL,
  `contactNo` bigint(20) DEFAULT NULL,
  `email` varchar(255) DEFAULT NULL,
  `password` varchar(255) DEFAULT NULL,
  `regDate` timestamp NULL DEFAULT current_timestamp(),
  `updationDate` varchar(45) DEFAULT NULL,
  `passUdateDate` varchar(45) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1_swedish_ci;

--
-- Dumping data for table `userregistration`
--

INSERT INTO `userregistration` (`id`, `regNo`, `firstName`, `middleName`, `lastName`,
`gender`, `contactNo`, `email`, `password`, `regDate`, `updationDate`, `passUdateDate`)
VALUES
```

(3, '10806121', 'Anuj', '', 'kumar', 'male', 1234567890, 'test@gmail.com', 'Test@123', '2025-01-02 14:56:18', NULL, NULL),

(4, '108061233', 'John', '', 'Doe', 'male', 1425362514, 'hohn@gmail.com', 'Test@123', '2025-01-02 14:56:18', NULL, NULL),

(5, 'BE123', 'John', '', 'Matthew ', 'male', 8956237845, 'john@gmail.com', '123', '2025-01-02 14:56:18', NULL, NULL),

(6, '14563213', 'Amit', 'kumar', 'Singh', 'male', 9632587412, 'amit123@gmail.com', 'Test@123', '2025-01-02 14:56:18', '17-04-2024 05:12:02', NULL);

--

-- Indexes for dumped tables

--

--

-- Indexes for table `admin`

--

ALTER TABLE `admin`

ADD PRIMARY KEY (`id`);

--

-- Indexes for table `complainthistory`

--

ALTER TABLE `complainthistory`

ADD PRIMARY KEY (`id`);

--

-- Indexes for table `complaints`

--

```
ALTER TABLE `complaints`
  ADD PRIMARY KEY (`id`);


--
-- Indexes for table `courses`
--
ALTER TABLE `courses`
  ADD PRIMARY KEY (`id`);


--
-- Indexes for table `feedback`
--
ALTER TABLE `feedback`
  ADD PRIMARY KEY (`id`);


--
-- Indexes for table `registration`
--
ALTER TABLE `registration`
  ADD PRIMARY KEY (`id`);


--
-- Indexes for table `rooms`
--
ALTER TABLE `rooms`
  ADD PRIMARY KEY (`id`),
  ADD KEY `room_no` (`room_no`);
```

```
--
-- Indexes for table `states`
--
ALTER TABLE `states`
  ADD PRIMARY KEY (`id`);

--
-- Indexes for table `userlog`
--
ALTER TABLE `userlog`
  ADD PRIMARY KEY (`id`);

--
-- Indexes for table `userregistration`
--
ALTER TABLE `userregistration`
  ADD PRIMARY KEY (`id`),
  ADD KEY `email` (`email`);

--
-- AUTO_INCREMENT for dumped tables
--

--
-- AUTO_INCREMENT for table `admin`
--
```

```
ALTER TABLE `admin`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=2;

--
-- AUTO_INCREMENT for table `complainthistory`
--
ALTER TABLE `complainthistory`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=3;

--
-- AUTO_INCREMENT for table `complaints`
--
ALTER TABLE `complaints`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=2;

--
-- AUTO_INCREMENT for table `courses`
--
ALTER TABLE `courses`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=9;

--
-- AUTO_INCREMENT for table `feedback`
--
ALTER TABLE `feedback`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=2;
```

```
--
-- AUTO_INCREMENT for table `registration`
--
ALTER TABLE `registration`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=7;


--
-- AUTO_INCREMENT for table `rooms`
--
ALTER TABLE `rooms`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=7;


--
-- AUTO_INCREMENT for table `states`
--
ALTER TABLE `states`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=37;


--
-- AUTO_INCREMENT for table `userlog`
--
ALTER TABLE `userlog`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=2;


--
-- AUTO_INCREMENT for table `userregistration`
--
```

```
ALTER TABLE `userregistration`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=7;
COMMIT;

/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
```

CHAPTER-7

SYSTEM TESTING

7.1 INTRODUCTION

System testing is the stage of implementation, which is aimed at ensuring that the system works accurately and efficiently before live operation commences. Testing is the process of executing the program with the intent of finding errors and missing operations and also a complete verification to determine whether the objectives are met and the user requirements are satisfied. The ultimate aim is quality assurance.

Tests are carried out and the results are compared with the expected document. In the case of erroneous results, debugging is done. Using detailed testing strategies a test plan is carried out on each module. The various tests performed in "Network Backup System" are unit testing, integration testing and user acceptance testing.

7.2 Types of Testing

7.2.1 Unit Testing

Unit testing focuses on verifying the correctness of individual modules within the system. Each component, such as user registration, login, room booking, complaint submission, and admin operations, was tested in isolation. These tests were designed to ensure that the fundamental functionality of each module works as expected under both valid and invalid input scenarios.

Test cases included:

- Successful and unsuccessful user registration with both valid and invalid inputs.
- Valid and invalid login attempts.
- Correct functionality of room booking with various booking statuses (available, booked, etc.).
- Complaint submission with required data fields.

The aim was to catch issues early in specific parts of the code, ensuring each unit behaves as expected before integration.

7.2.2 Integration Testing

After individual modules were verified, integration testing was conducted to assess how well the modules work together. This phase validated the data flow between different system components and ensured that one module's output is correctly passed as input to the next.

Key integration test scenarios included:

- **Post-registration workflow**: After successful student registration, the room booking functionality was tested to ensure seamless data transition.
- **Login and complaint submission**: The complaint submission functionality was tested after successful login, ensuring data integrity and correct processing.
- **Admin room management**: After users made bookings, the admin's ability to manage room availability and approve/reject bookings was validated.

The goal was to ensure that the interaction between modules did not introduce data inconsistencies or functional errors.

7.2.3 System Testing

System testing was conducted on the entire Hostel Management System, ensuring that the integrated system met all functional, non-functional, and security requirements. Tests were performed to verify system behavior under different real-world conditions.

The following scenarios were tested:

- **User Registration and Login**: Successful registration and login flows were tested.
- **Room Booking and Availability**: Booking a room and ensuring the system accurately reflected real-time availability.
- **Complaint and Feedback Submission**: Testing of the complaint submission feature, ensuring data is handled correctly.
- **Admin Approval Process**: Validation of admin's ability to approve or reject booking requests.
- **Password Change and Session Management**: Ensuring that users could change their passwords, and that session timeouts and logouts functioned correctly.

This phase helped verify the system's performance and functionality across multiple use cases.

7.2.4 User Acceptance Testing (UAT)

User Acceptance Testing (UAT) involved real users, such as students and admin volunteers, who interacted with the system in a controlled environment. Their feedback was used to ensure that the system met user expectations and was ready for deployment.

Key areas evaluated included:

- **Ease of Room Booking**: Students tested the booking flow to ensure it was straightforward and intuitive.
- **Complaint Submission Process**: Users assessed the simplicity of submitting complaints and feedback.
- **User Interface (UI) Usability**: The overall ease of navigation and user-friendly design of the system.
- **Overall Satisfaction**: Feedback on the system's performance, responsiveness, and usability.

UAT confirmed that the system was user-friendly, efficient, and met the requirements of the stakeholders, indicating its readiness for live deployment.

7.3 Testing Strategy

The overall testing approach was designed to address all critical aspects of the system's performance, security, and functionality.

Testing Techniques Used

- **Black Box Testing**: Focused purely on validating system functionality without any examination of the internal code structure. This method ensured that users' interactions with the system met their expectations.
- **Functional Testing**: This involved checking that specific functional modules (registration, login, booking, feedback) worked as intended.
- **Performance Testing**: Ensured the system could handle high usage loads without performance degradation, especially under peak usage scenarios (e.g., many users attempting to book rooms simultaneously).
- **Security Testing**: Tested to ensure that the system properly handled security concerns such as secure login processes, session management, and protection against unauthorized access.

This structure helps ensure that the Hostel Management System is not only functional but also secure, user-friendly, and capable of handling expected real-world scenarios.

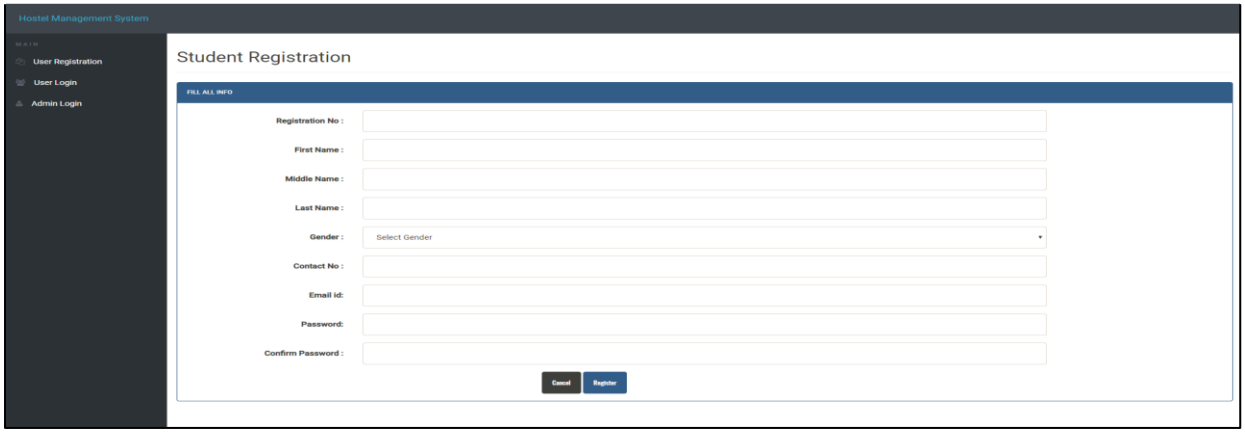
CHAPTER-9

OUTPUT SCREENS

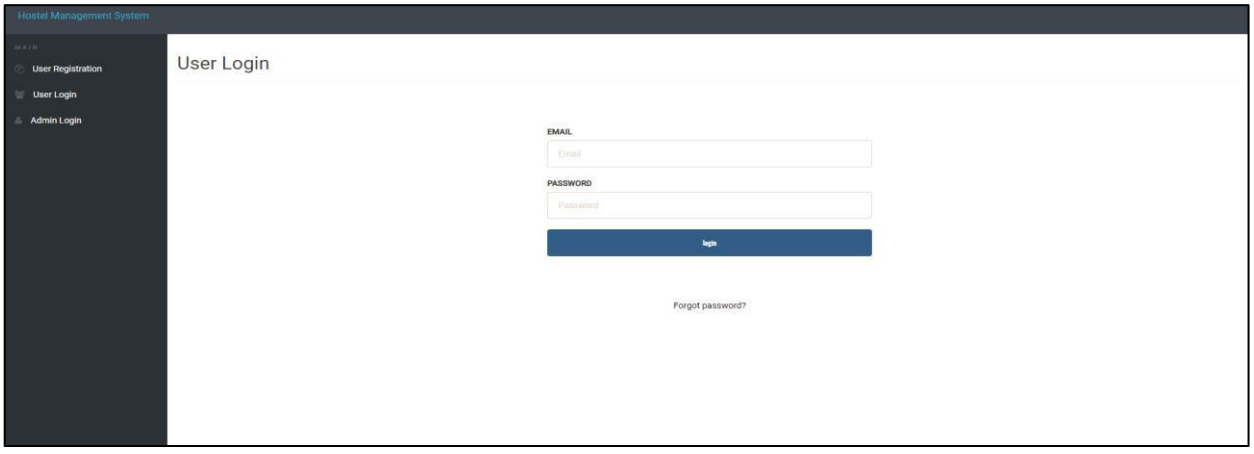
The hostel management system provides several interactive screens designed for both administrators and students (users)

9.1 USER MODULE

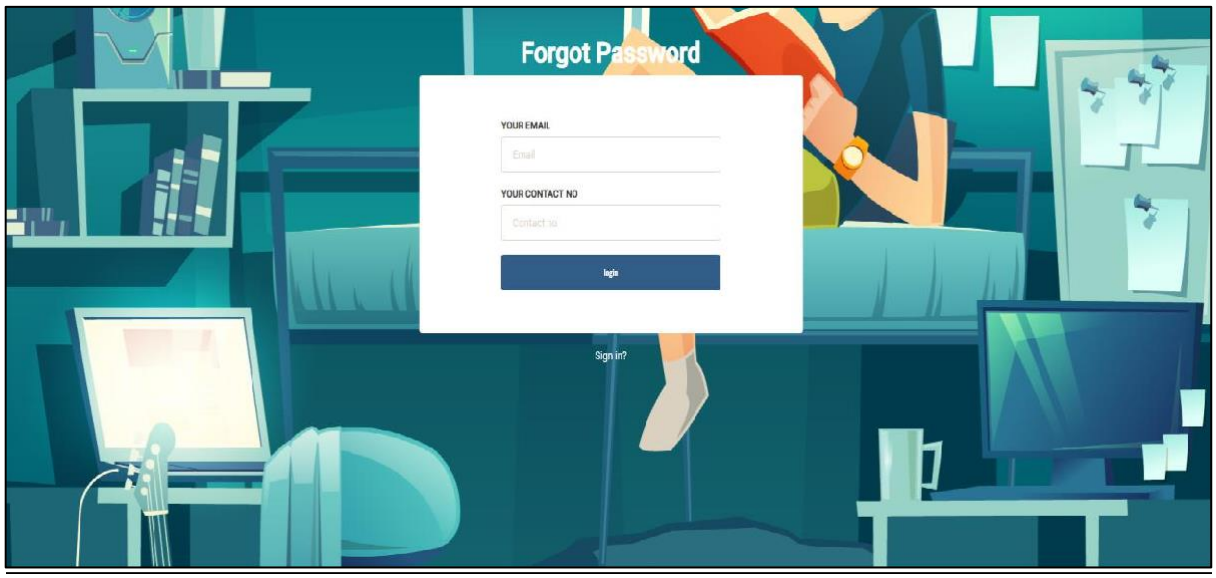
User signup: allows new users (students) to create an account



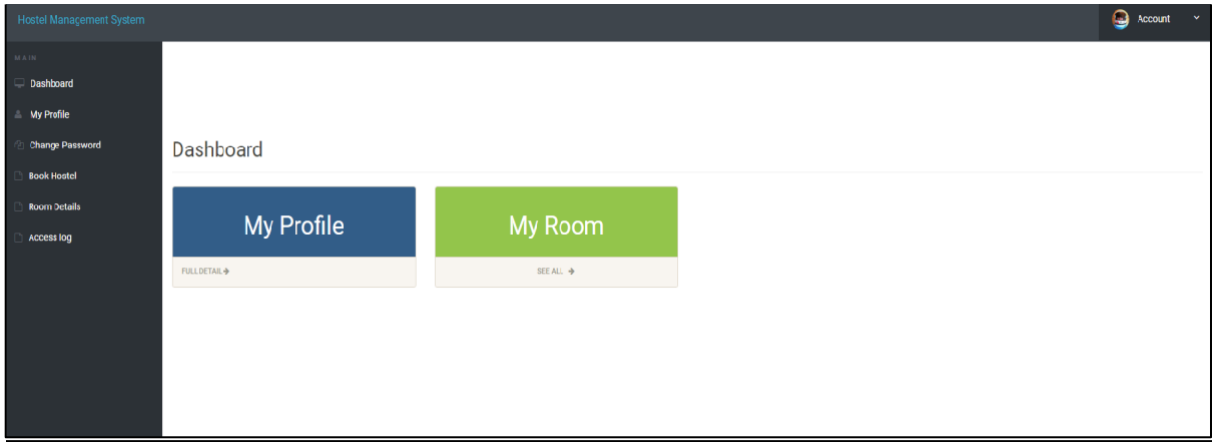
User sign in: allows existing users to login using credentials



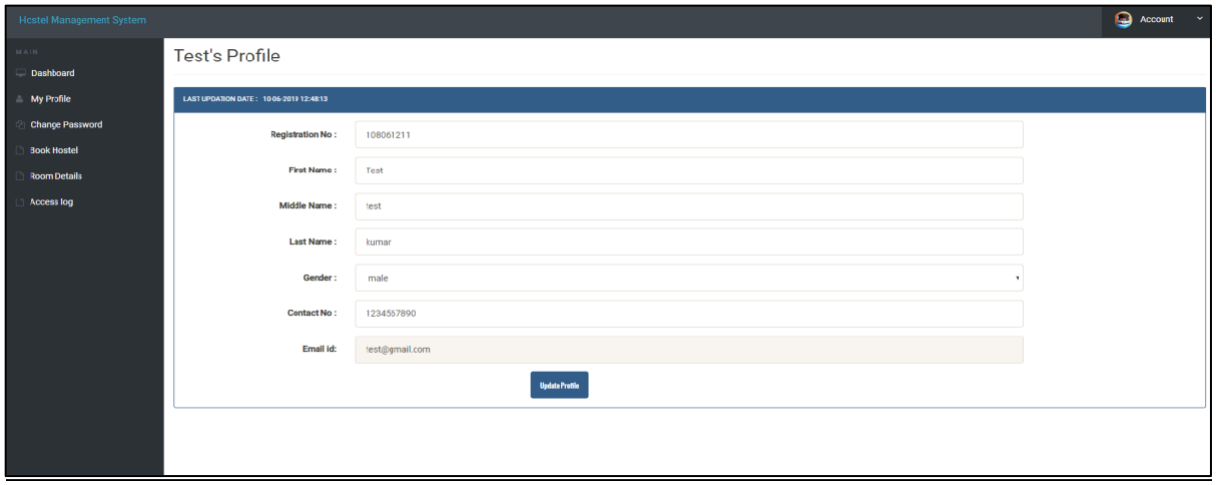
User recovery password: helps users recover their password via a security question or OTP



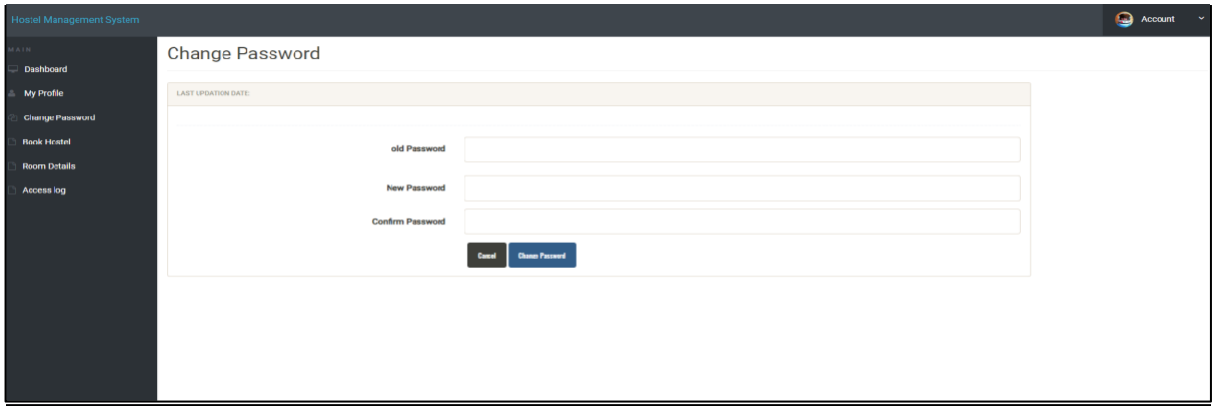
User dashboard:displays an overview of user actions



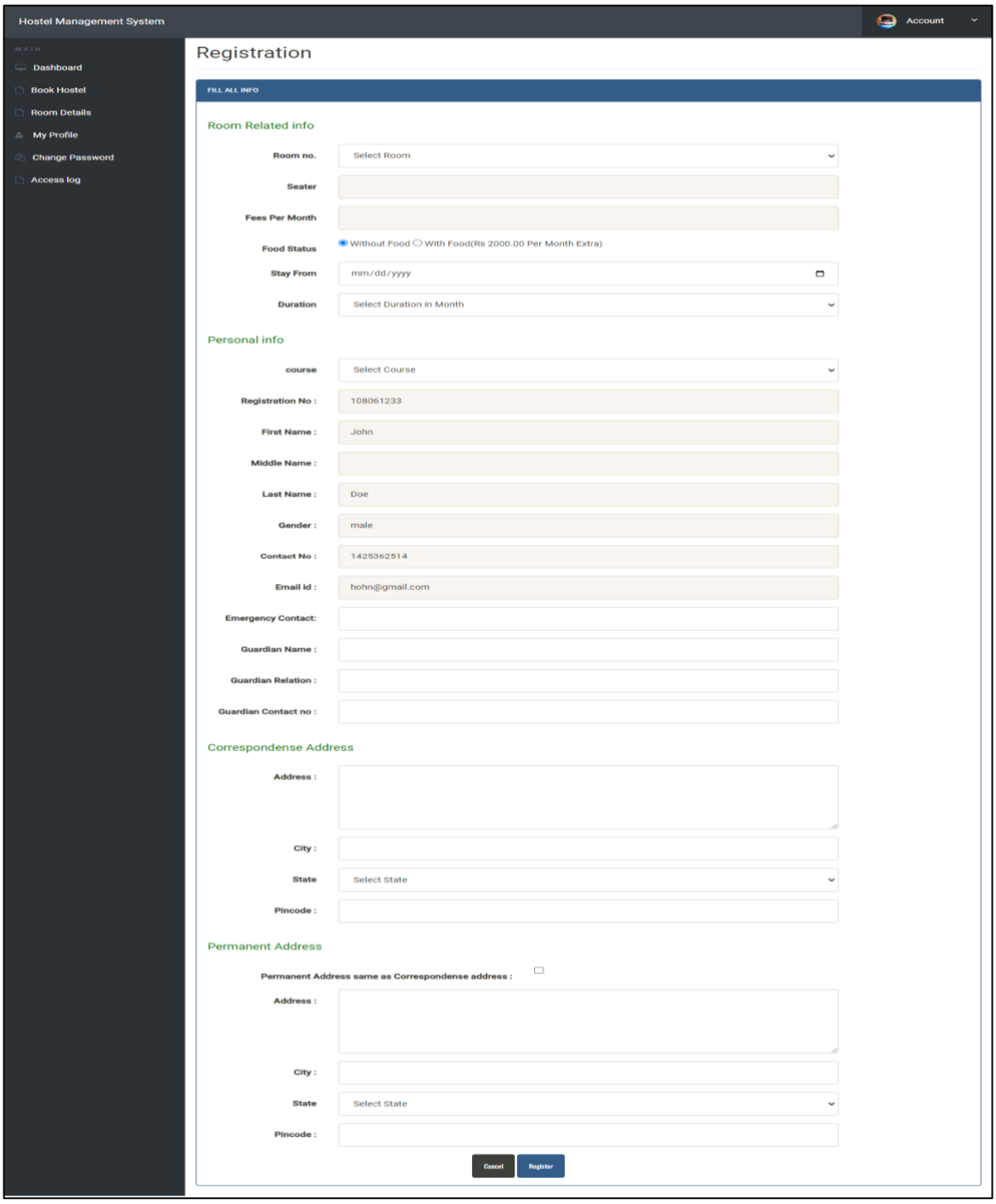
User profile:displays users personal information



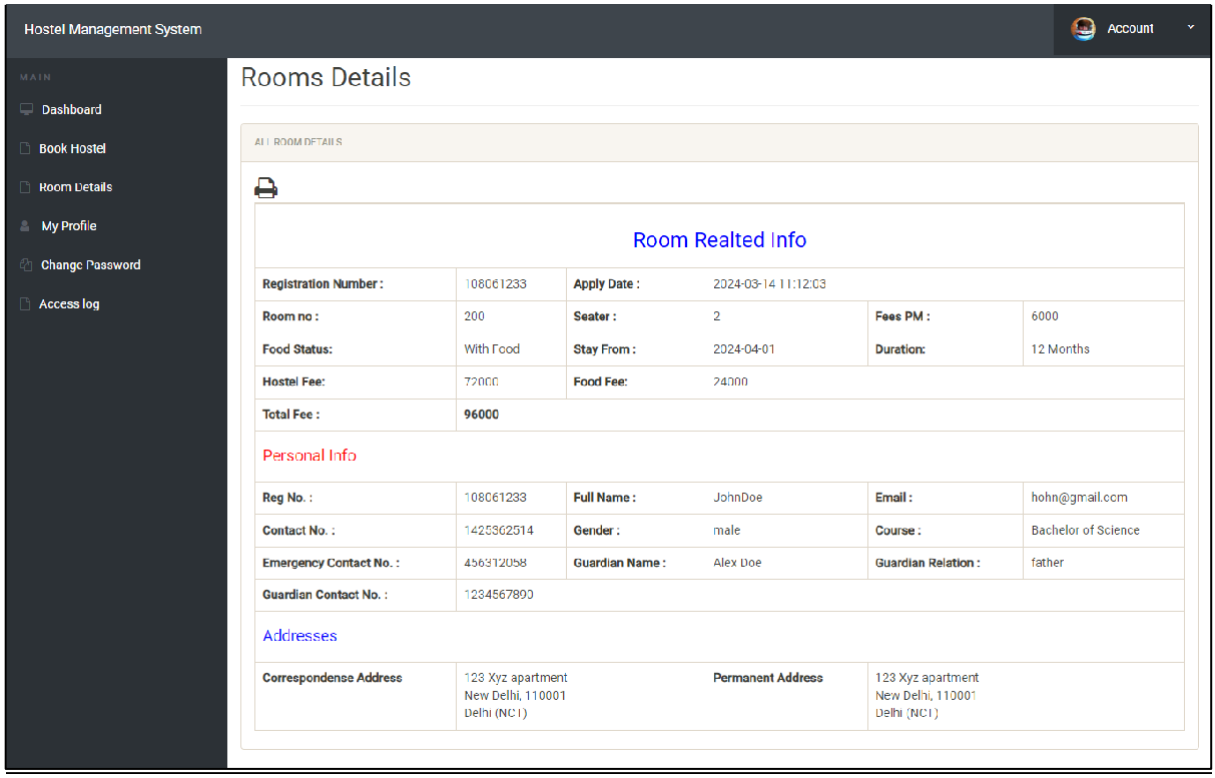
User change password:allows users to security update their passeord



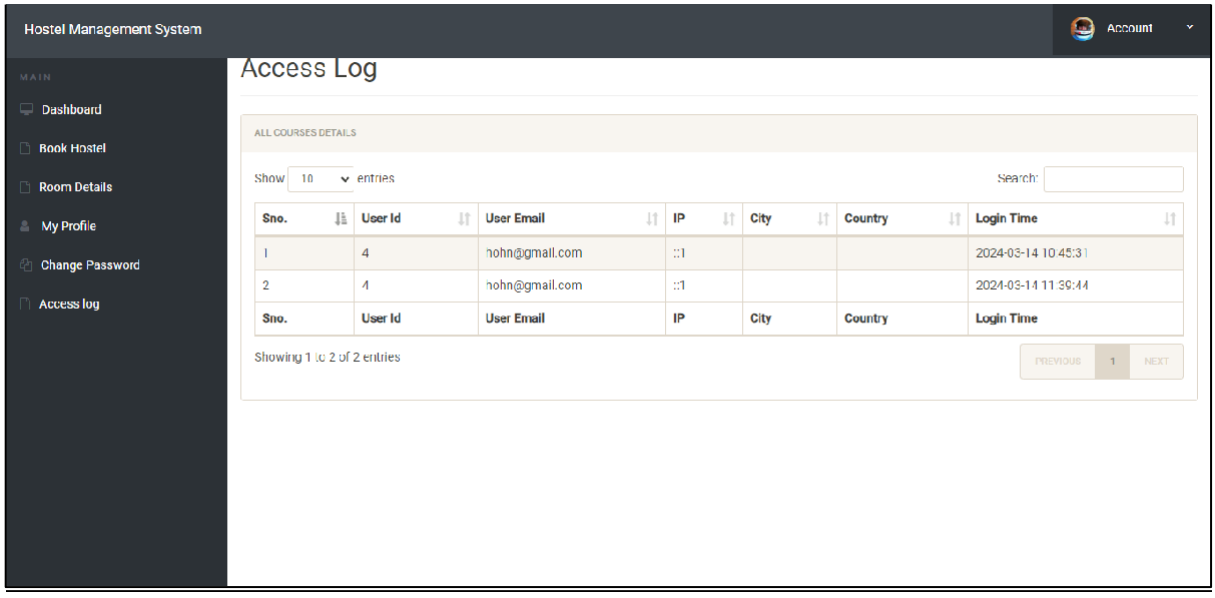
User hostel booking:enables students to book available rooms



User Booked Hostel Details:displays current room allocation

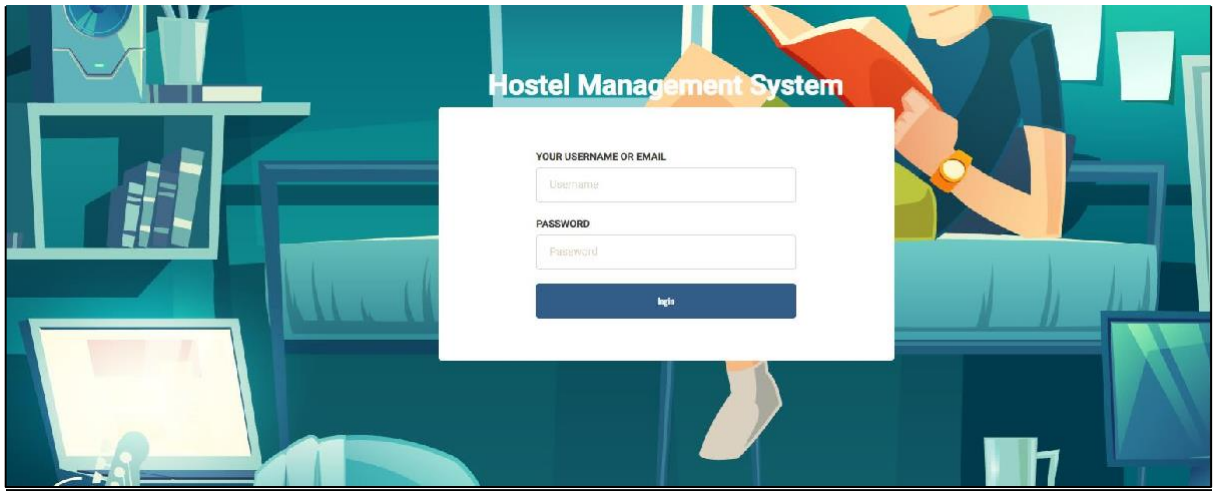


User access log details:show login history

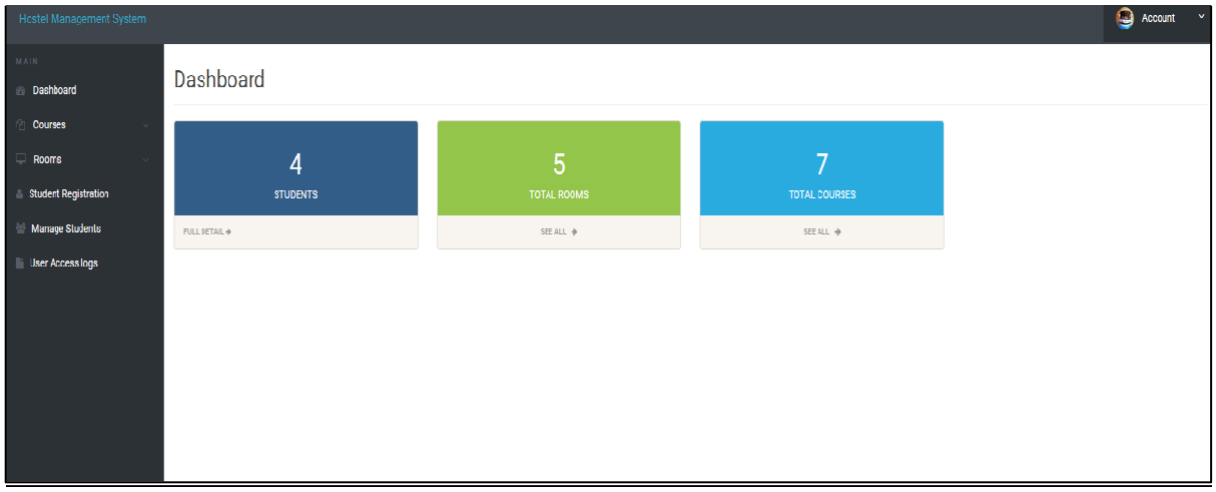


9.2 ADMIN MODULE

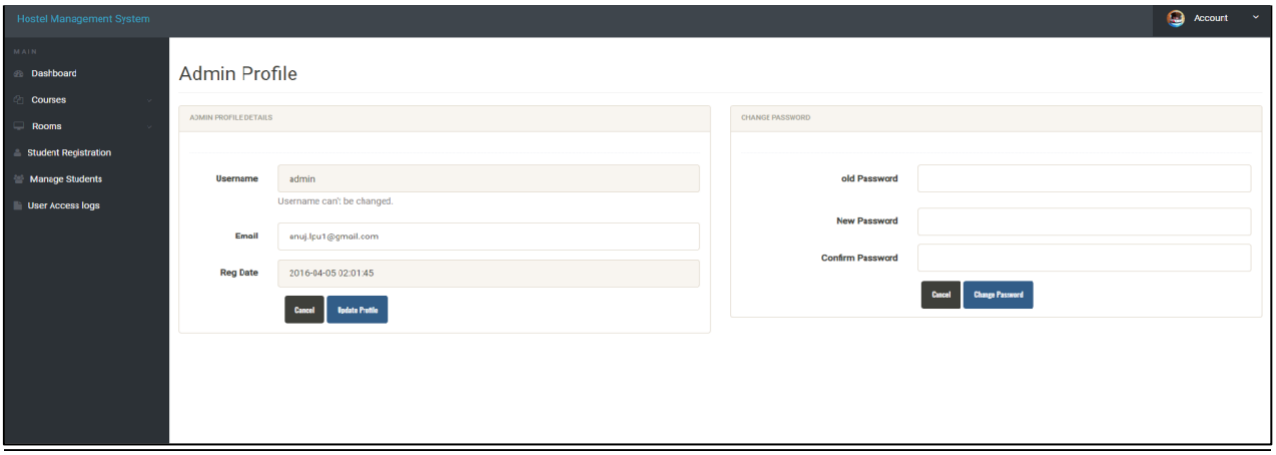
Admin Login: secure entry for administrator



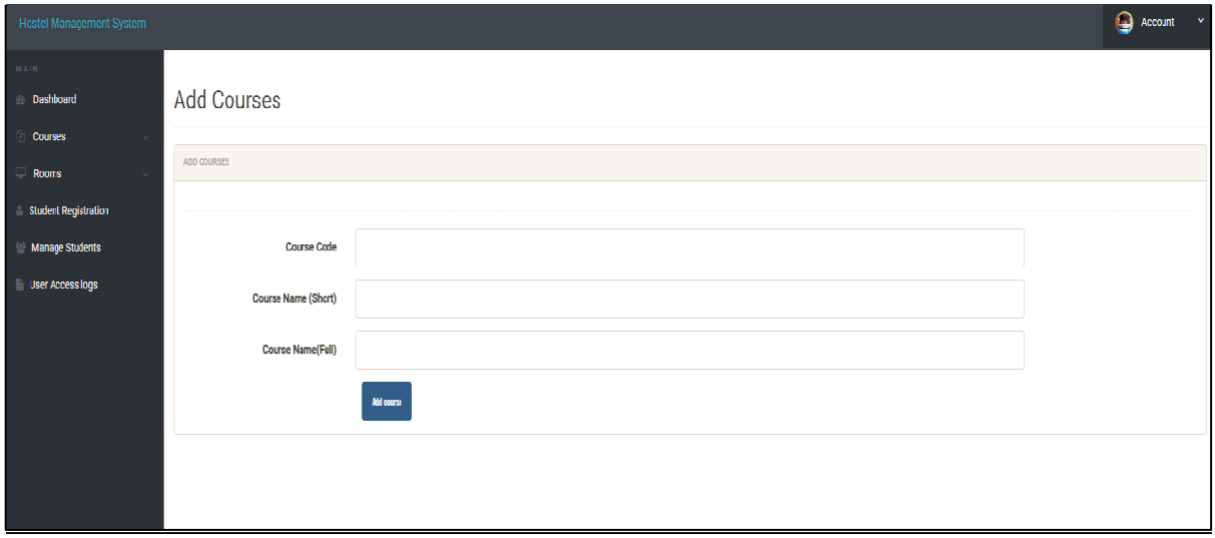
Admin Dashboard:admin control panel ,displaying system status



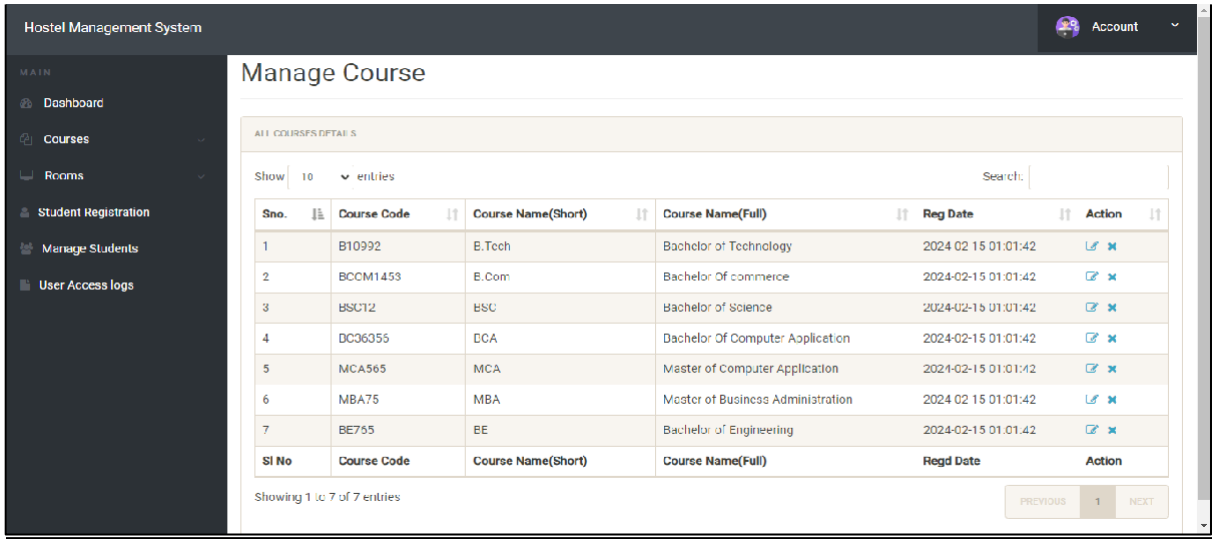
Admin Profile:displays and allows update of admin info



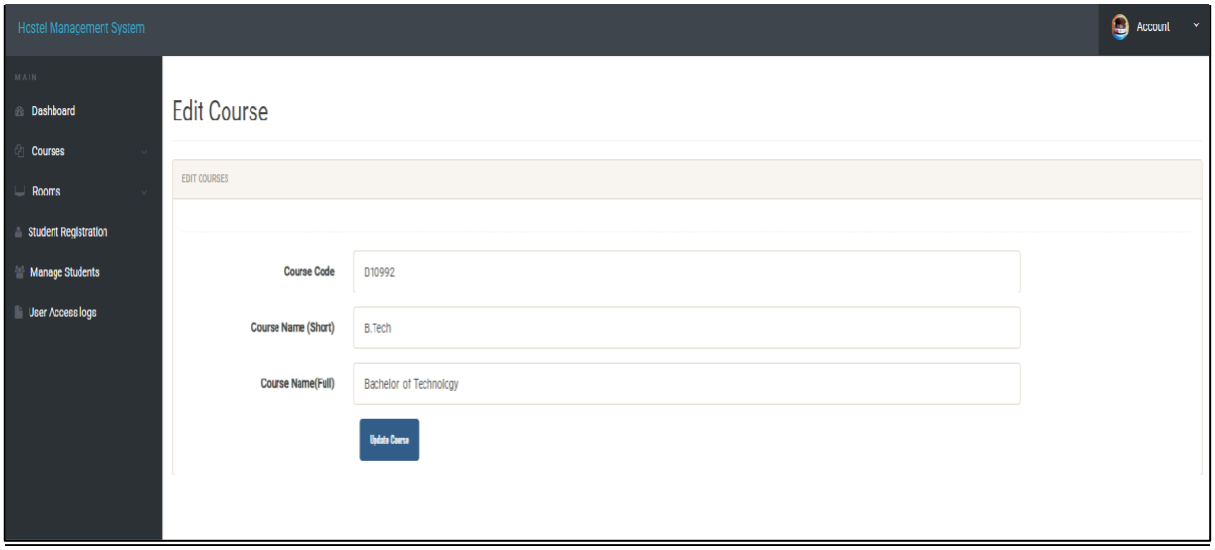
Admin Add Course :admin can add academic courses



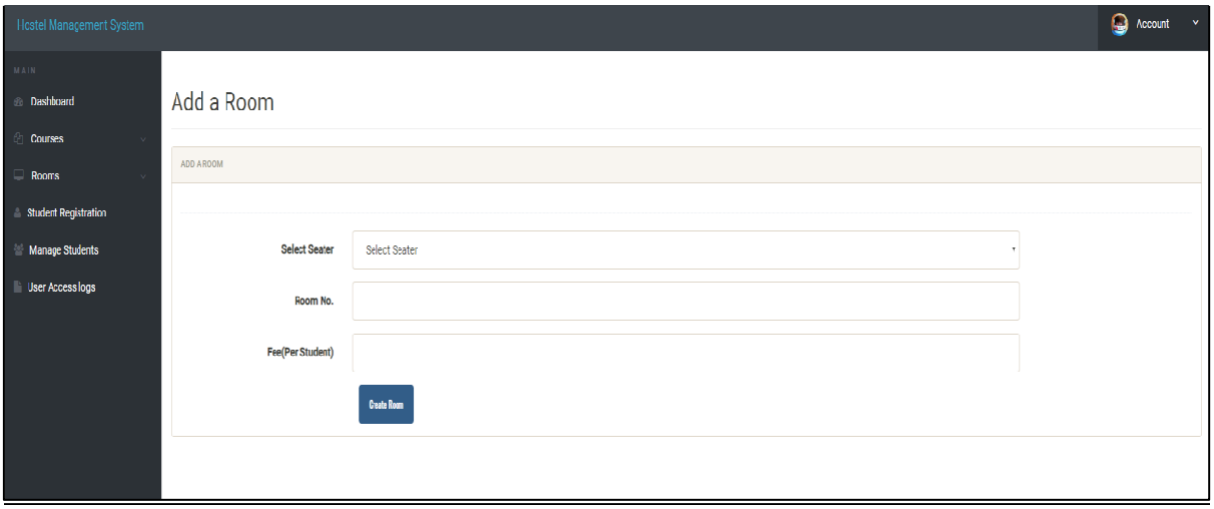
Admin Manage Courses: view, update and delete existing courses



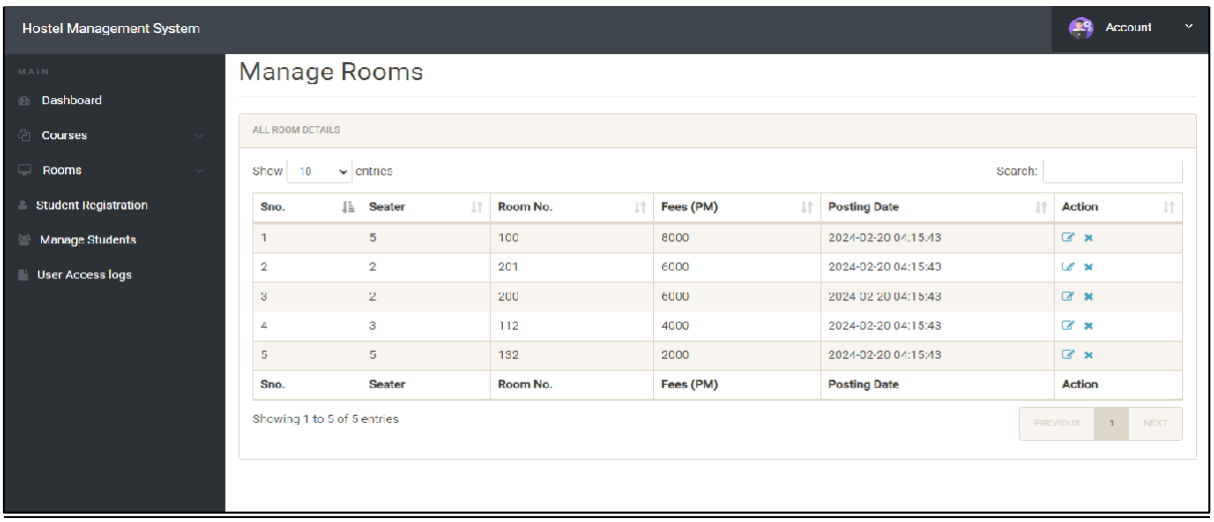
Admin edit Course Details: update course name, duration, or code



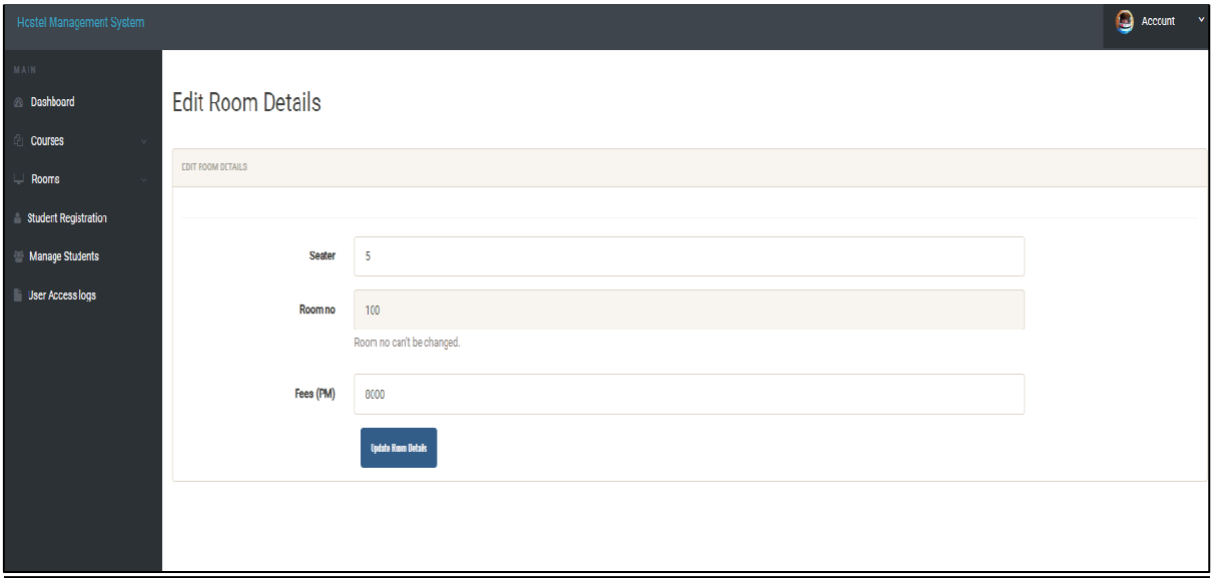
Admin add room:adds a new hostel room



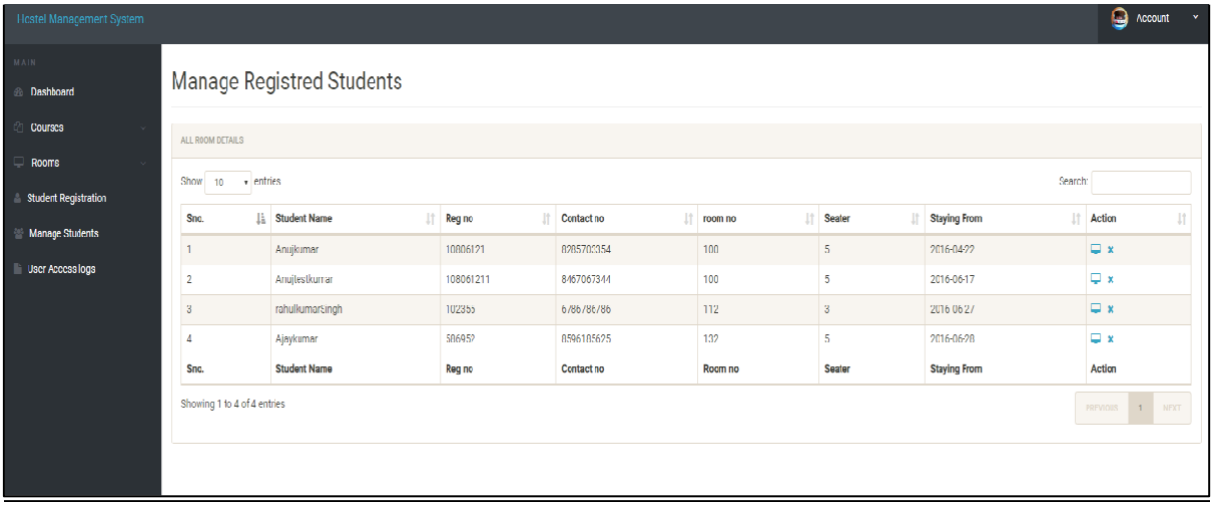
Admin Manage Rooms:list,edit or delete rooms



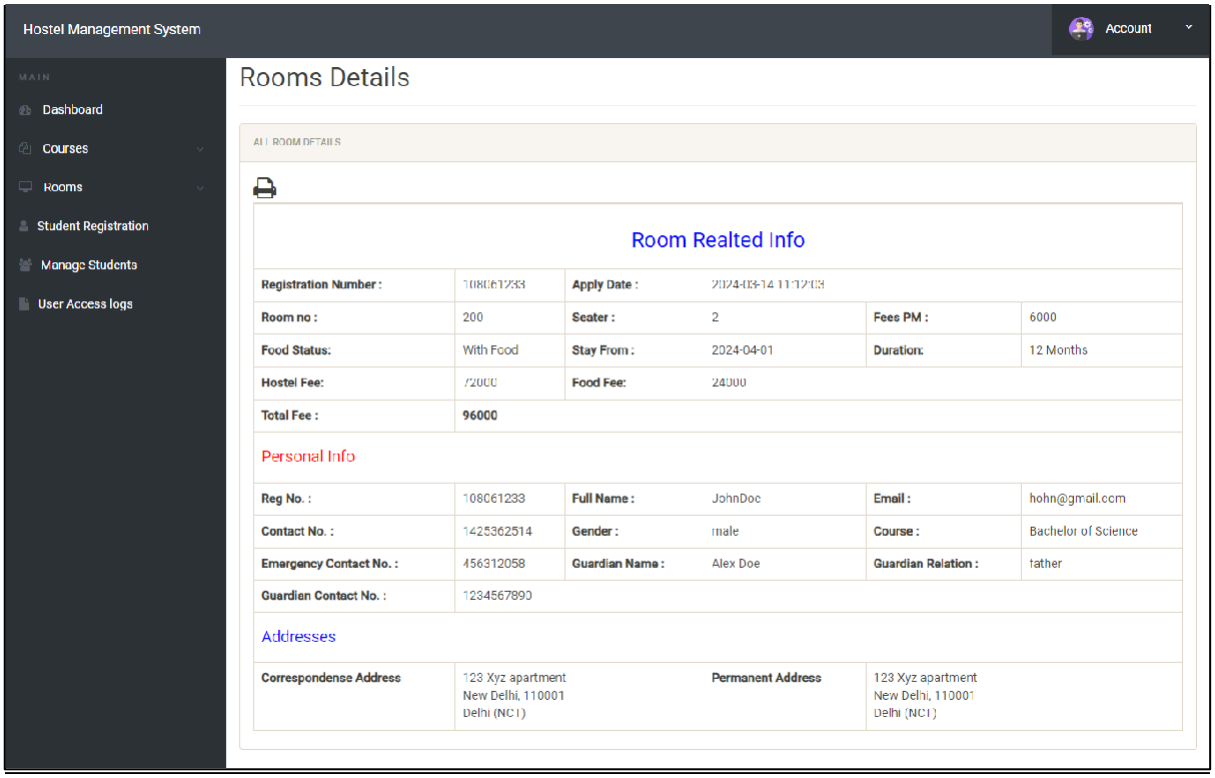
Admin Edit Room Details:allows editing of room records(number,fee,seater)



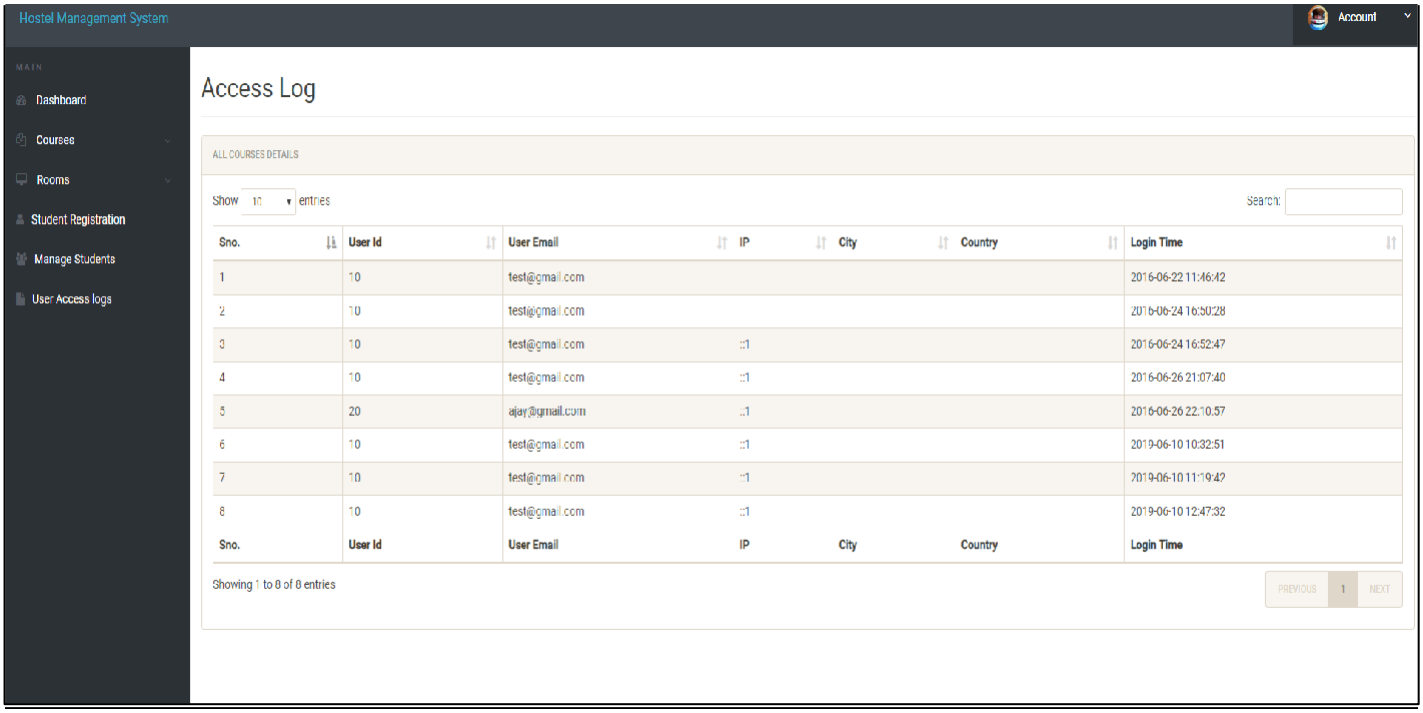
Manage Registered Students: displays all registered students



Registered Student Details:detailed student profile view



User Access logs :logs and display student login data



CHAPTER-7

CONCLUSION

To conclude the description about the project: The project, developed using PHP and MySQL is based on the requirement specification of the user and the analysis of the existing system, with flexibility for future enhancement.

The expanded functionality of today's software requires an appropriate approach towards software development. This hostel management software is designed for people who want to manage various activities in the hostel. For the past few years the number of educational institutions are increasing rapidly.

Thereby the number of hostels are also increasing for the accommodation of the students studying in this institution. And hence there is a lot of strain on the person who are running the hostel and software's are not usually used in this context. This particular project deals with the problems on managing a hostel and avoids the problems which occur when carried manually.

Identification of the drawbacks of the existing system leads to the designing of computerized system that will be compatible to the existing system with the system which is more user friendly and more GUI oriented.

CHAPTER-8

REFERENCES

Ahmed, S.R., 2012. DesktopGI application for hostel management of Punjab University Lahore. *Journal of Himalayan Earth Science,* 45 (2).

ZIBRA,2018. *Thedigitization of the Italianhospitality industry an empirical analysis*.

Kumar, N., 2015. *InformationSeekingBehaviorof Faculty Members and Students in the Digital Environment Great Lakes Institute of Management: A Case Study. Historv*. 21 (72),pp. 285–294.

Horrall, J. and Merrall, J., 1998. *Selecting Research Collections for Digitization*.

IntelligentsecuritysystemforgirlsinhostelAmolSopkall, **S Amikesha U. Katait 2, PoojaS. Ingole 3, Neha Dumre 4, *PreethiUghade IJARSE*,** Vol. No. 4, Special Issue (01), March 2015.

Marshall, Esther and Lynette Mackenzie. "Adjustment to residential care the experience of newly admitted residents to hostel accommodation in Australia." *Australian occupational therapy journa* 1552 (2008): 123–32.