HOSTEL ROOM ALLOTMENT SYSTEM

A Real Time Project report submitted in partial fulfilment of the requirements for the award of the degree of

Bachelor of Technology

In

Computer Science and Engineering

By

ANNADI CHANDANA (23011A0516)

KARLAKUNTA AKSHARA (23011A0544)

SUCHITRA DAS (23011A0566)

Under The Guidance of

Dr. K P SUPREETHI

Professor & Head of the Department

Department of Computer Science and Engineering,

JNTUH University College of Engineering, Science &Technology

Hyderabad

Kukatpally, Hyderabad - 500 085.

July 2025

Department of Computer Science and Engineering,

JNTUH University College of Engineering, Science &Technology

Hyderabad

Kukatpally, Hyderabad - 500 085.

July 2025

DECLARATION BY THE CANDIDATE

We, A.CHANDANA(23011A0516), K. AKSHARA(23011A0544),

SUCHITRA DAS (23011A0566), hereby declare that the real time project report entitled "**HOSTEL ROOM ALLOTMENT SYSTEM**", carried out by us under the guidance of **Dr. K P Supreethi,** is submitted in partial fulfilment of the requirements for the award of the degree of *Bachelor of Technology in Computer Science and Engineering.* This is a record of Bonafide work carried out by us and the results embodied in this project have not been reproduced /copied from any source. Whenever I have used materials (data, theoretical analysis, figures, and text) from other sources, I have given due credit to them by citing them in the text of the report and giving their details in the references. The results embodied in the project have not been submitted to any other University or Institute for the award of any other Degree or Diploma.

A. CHANDANA (23011A0516)

K. AKSHARA (23011A0544)

SUCHITRA DAS (23011A0566)

Date:

Department of Computer Science and Engineering,

JNTUH University College of Engineering, Science &Technology

Hyderabad

Kukatpally, Hyderabad - 500 085.

July 2025

CERTIFICATE BY THE SUPERVISOR

This is to certify that the project report entitled "**HOSTEL ROOM ALLOTMENT SYSTEM**", being submitted by A**.** CHANDANA (23011A0516), K. AKSHARA (23011A0544), SUCHITRA DAS (23011A0566), in partial fulfilment of the requirements for the award of the degree of *Bachelor of Technology* in *Computer Science and Engineering*, is a record of Bonafide work carried out by them.

The results of investigation enclosed in this report have been verified and found satisfactory. The results embodied in the project have not been submitted to any other University or Institute for the award of any other Degree or Diploma.

DR. K P SUPREETHI,

Professor & Head of the Department

Date:

Department of Computer Science and Engineering,

JNTUH University College of Engineering, Science &Technology

Hyderabad

Kukatpally, Hyderabad - 500 085.

July 2025

CERTIFICATE BY THE HEAD OF THE DEPARTMENT

This is to certify that the project report entitled "**HOSTEL ROOM ALLOTMENT SYSTEM**", being submitted by A**.** CHANDANA (23011A0516), K. AKSHARA (23011A0544), SUCHITRA DAS (23011A0566), in partial fulfilment of the requirements for the award of the degree of *Bachelor of Technology* in *Computer Science and Engineering*, is a record of Bonafide work carried out by them.

Dr. K P SUPREETHI,

Professor & Head of the Department

Date:

TABLE OF CONTENTS

LIST OF FIGURES AND SCREENSHOTS

LIST OF TABLES:

ACKNOWLEDGEMENTS

I sincerely thank my mentor, Dr. K P Supreethi, for her continuous support, invaluable feedback, and insightful guidance, which significantly contributed to the improvement and successful completion of this project. Her suggestions not only helped refine our work but also inspired us to push the boundaries of our capabilities.

I would also like to extend my heartfelt gratitude to my teammates for their dedication, collaborative spirit, and tireless efforts throughout the course of the project. Their contribution and teamwork were vital in ensuring that we achieved our goals efficiently and effectively.

I am truly grateful to the Department of Computer Science and Engineering, Jawaharlal Nehru Technological University Hyderabad, for providing the resources, infrastructure, and academic platform needed to complete this project. The department's support and encouragement made it possible for us to focus on learning and development without any hindrances.

A special thanks to all those who helped in various ways whether by offering suggestions, sharing knowledge, or simply being there for us. Their encouragement and constructive feedback were greatly appreciated.

This project has been a valuable learning experience, allowing me to gain hands-on experience and improve my technical, problem-solving, and teamwork skills. It has broadened my understanding and has helped me grow both academically and personally.

ABSTRACT

Generally, administrators who allot rooms to students rely on static lists or manual selection without a clear idea of the room's availability and amenities. With AR-based 3D modelling, they can virtually explore available rooms before allotting rooms. The system will display real time room availability, occupancy details, furniture, amenities, and other essential information when they search for rooms. This eliminates uncertainty and enhances the overall hostel allocation experience and it also reduces clashes. The Hostel Allocation System is a web-based application that helps allocate hostel rooms to students efficiently based on availability, preferences, and predefined rules. It aims to reduce manual effort and provide a smooth allocation process for both students and administrators. Our Hostel Room Allotment System uses Django (Python) for backend, MySQL for database, and Django Auth System for authentication. The frontend is built with HTML, CSS, and JavaScript. WebXR and Three.js enable AR-based 3D visualization, allowing students to explore their assigned rooms virtually.

1.INTRODUCTION

The hostel room allotment process is a critical administrative task in academic institutions.This project introduces a modern, web-based solution integrated with AR to enhance efficiency, transparency, and student satisfaction.

1.1 MOTIVATION:

In many colleges and universities, hostel room allotment is still managed manually, which leads to confusion, delays, and unfair distribution of rooms. Students often don't have a clear idea of room availability, and administrators struggle with managing room data efficiently. The motivation behind this project is to simplify and modernize the allotment process using web technologies and AR-based visualization. By building an intelligent, user-friendly platform, we aim to reduce manual workload, provide transparency, and offer students a better experience during room selection.

1.2 PROBLEM STATEMENT

Traditional hostel allotment systems lack real-time data, automation, and clarity, which often results in double bookings, data entry errors, and student dissatisfaction. Admins are burdened with repetitive tasks like verifying room status, updating records, and handling complaints manually. Students have no way of knowing what rooms are available or suitable for them before allotment. There is a clear need for an automated system that can manage student data, handle allotments based on predefined rules, and provide real-time room visibility.

1.3 OBJECTIVES:

The main objective of this project is to develop a web-based hostel room allotment Specific goals include system that streamlines the allocation process through automation and real-time updates.

- Providing separate login access for students and administrators
- Managing hostel room data (available, occupied, capacity)
- Allowing students to view and request rooms based on preferences
- Automating the room allotment logic
- Integrating AR-based visualization to explore rooms virtually
- Reducing manual work and improving the overall hostel management experience

1.4 PURPOSE AND SIGNIFICANCE:

The purpose of this project is to simplify and automate the hostel room allotment process in educational institutions. It aims to eliminate the manual and error-prone methods of room distribution by offering a digital platform where administrators can manage allotments efficiently and students can easily view room details and availability.

The system is significant because it introduces real-time data access, reduces human workload, and improves the transparency of room assignments. Features like AR-based 3D room visualization further enhance the student experience, allowing them to explore rooms virtually before allotment. This modern approach not only saves time but also minimizes room-related conflicts and ensures fair distribution.

1.5 SCOPE OF THE PROJECT

The scope of this project covers the development of a web-based hostel room allotment system that includes:

- •User Authentication for both admin and students
- •Student Registration and Login features
- •Room Management Module to track availability and occupancy
- •Room Allotment Logic based on student preferences and room capacity
- •Real-time status updates of available and occupied rooms
- •AR/3D Room Visualization using WebXR and Three.js
- •Email-based password reset using Django's authentication system
- •Admin Dashboard to manage students and room allocations

2 LITERATURE SURVEY

Several existing hostel management systems use technologies like PHP, MySQL, or Android apps to automate room allotment. However, most lack real-time updates, visual room layouts, and intelligent decision-making features. Studies on AR in educational settings show its potential to enhance spatial understanding, but its use in hostel allotment remains limited.

2.1 EXISTING SYSTEM:

Many colleges and universities currently rely on basic tools like spreadsheets or outdated local databases for hostel room allotment, which often results in inefficiencies and are often time-consuming, prone to human error, and lack transparency. In many cases, room preferences and capacity constraints are not handled efficiently. Additionally, there is no option for students to visually explore their allotted rooms in advance, and they are often unaware of their assigned roommates prior to moving in.Some institutions have introduced basic web portals to handle hostel registration processes. However, these systems are often limited in functionality — they primarily allow students to submit registration forms without offering real-time room availability, automated allotment logic, or interactive user interfaces. Such portals typically lack features like live updates, preference handling, room visualization, or secure communication between students and administrators, making the process only slightly better than traditional manual methods.

2.2 GAPS IDENTIFIED :

After studying existing systems and methods used in hostel room allotment, several limitations were observed. Most current solutions are either manual or only partially digital, lacking important features that would improve efficiency and user experience. These gaps highlight the need for a more advanced and complete system.

The existing systems suffer from several key limitations such as

- •No real-time room availability tracking.
- •Lack of user-friendly interfaces for students and admins.
- •Manual allotment leads to errors and delays.
- •No room preview or layout visualization for students.
- •Absence of role-based login access and secure authentication.
- •No integration of modern features like email notifications or AR.

2.2 OUR VALUE PROPOSITION:

Our Hostel Room Allotment System solves the common problems seen in manual and outdated hostel allocation methods. Instead of relying on spreadsheets or paper-based processes, our system offers a smart and automated solution. It allows students to check room availability in real-time, submit their preferences, and even explore rooms virtually using 3D or AR-based visualization. The admin can easily manage student data, room status, and allotments through a clean and easy-to-use dashboard. The system also sends email updates and supports password resets, making it smooth and user-friendly for both students and administrators. Overall, it brings more transparency, reduces workload, and improves the entire hostel allocation experience.

3. SYSTEM REQUIREMENTS

The system requires a web-enabled environment with support for database integration and secure user login. It can run on any platform that supports standard web technologies and internet access.

3.1 SOFTWARE REQUIREMENTS:

FRONTEND:

JavaScript

HTML5

CSS

BACKEND:

Component	Technology/Tool	Purpose
Framework	Django(Python)	Main web framework to build views,handle routing ,and manage logic.
Template Engine	Django Templates(.html)	For rendering dynamic HTML pages from views
Database	SQLite	Lightweight, file-based database used with Django ORM
ORM	Django ORM	Handles database interactions using Python (no raw SQL)
Authentication	Django Auth System	For secure login handling of admin and student roles
Email Service	Django Email(SMTP)	To send password reset emails to students
Data Visualization	Matplotlib	For generating dynamic bar charts of room status
File I/O	Python's Bytes IO	To create in-memory chart images for HTTP response

TOOLS AND UTILITIES:

VS Code or any preferred IDE

Cloud deployment platform (e.g., Render, Vercel)

npm or yarn (for package management)

3.2 FRONTEND:

The frontend of the project is developed using HTML5, CSS3, and JavaScript to create a clean, responsive, and interactive user interface. The design follows a modular approach with reusable components and a consistent layout structure. JavaScript is used to handle dynamic functionalities such as form validation, dropdown filters, and interactive elements on the dashboard.

Client-server communication is managed using the Fetch API, enabling asynchronous requests to the backend for data retrieval and updates. Navigation is handled using simple HTML-based routing structures. The frontend design ensures cross-browser compatibility and is optimized for both desktop and mobile displays, enhancing the overall user experience for administrators and students.

3.3 BACKEND:

The backend of the project is implemented using the Django framework, leveraging Python for business logic and ORM-based database interaction. Django's powerful admin and view system simplifies server-side processing, form handling, session management, and rendering of dynamic content.Data is stored and managed using SQLite, a lightweight, file-based relational database that integrates seamlessly with Django. Core backend operations include user authentication, room allocation, real-time occupancy tracking, and automated email services for password recovery. The backend ensures data integrity, security, and smooth communication with the frontend through RESTful API views. It is deployed using cloud hosting platforms such as Render or Railway, offering high availability and scalability.

ADDITIONAL TOOLS AND DEPENDENCIES:

•Version Control (Git and GitHub): Used for tracking changes, collaborative development, and maintaining code history across development phases.

•Deployment Tools (Render/Vercel/GitHub Pages).

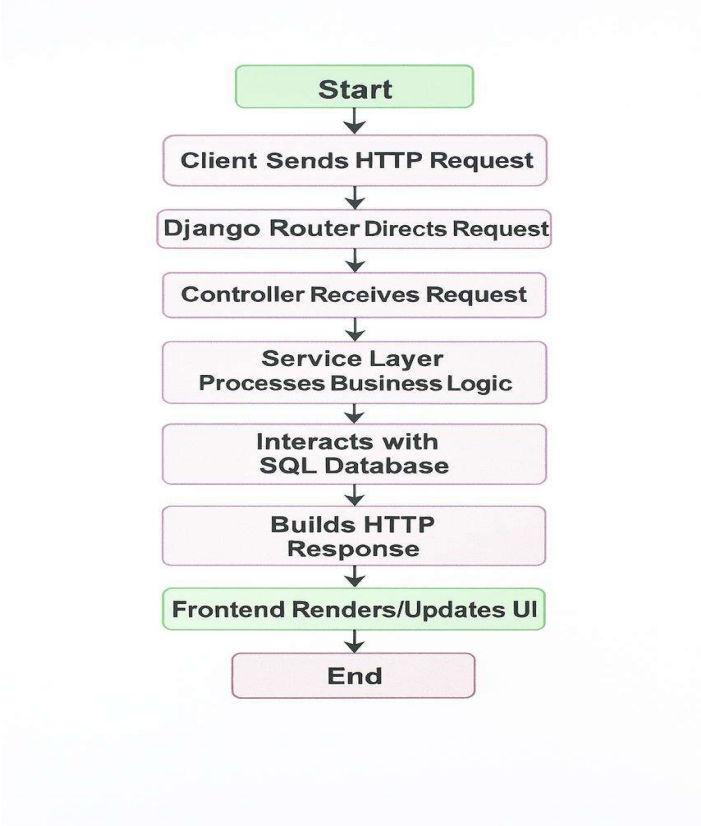

Fig 3.1: Workflow

Fig 3.1: Workflow and Tech Stack illustrates the system's architecture, outlining the end-to-end workflow from frontend input to backend processing and database interaction. It also highlights the technologies used at each stage, including frameworks, databases, and integration tools.

4. SYSTEM DESIGN AND METHODOLOGY

The system is designed using a modular client-server architecture with clear separation of frontend, backend, and database. An agile development approach was followed to ensure iterative progress and flexibility.

4.1 ARCHITECTURE:

The architecture of the Hostel Room Allotment System follows a classic client-server model. It is designed to support clean separation between the user interface, backend processing, and data storage. The overall system is modular, extensible, and built for efficient real-time room management.

1. Frontend (Presentation Layer)

•Built using HTML5, CSS3, and JavaScript.

•Handles all user interactions — including login, student dashboards, room viewing, and admin panel operations.

•Integrated with Three.js and WebXR for Augmented Reality (AR)-based 3D room previews.

•Communicates with the Django backend through standard form submissions and optionally via Fetch API.

2. Backend (Application Layer)

•Built using Django (Python), responsible for all server-side logic.

•Includes modules for:

•Student & admin login

•Room allocation

•Email-based password reset

•Dashboard data visualization (charts, summaries)

•Smart room suggestions (based on branch & availability)

•Handles request routing via Django's views.py and urls.py.

3. Database Layer

•Uses SQLite, a lightweight relational database.

•Stores:

•Student data

•Room inventory (room number, beds, occupancy)

•Room allocations

•Login credentials and history

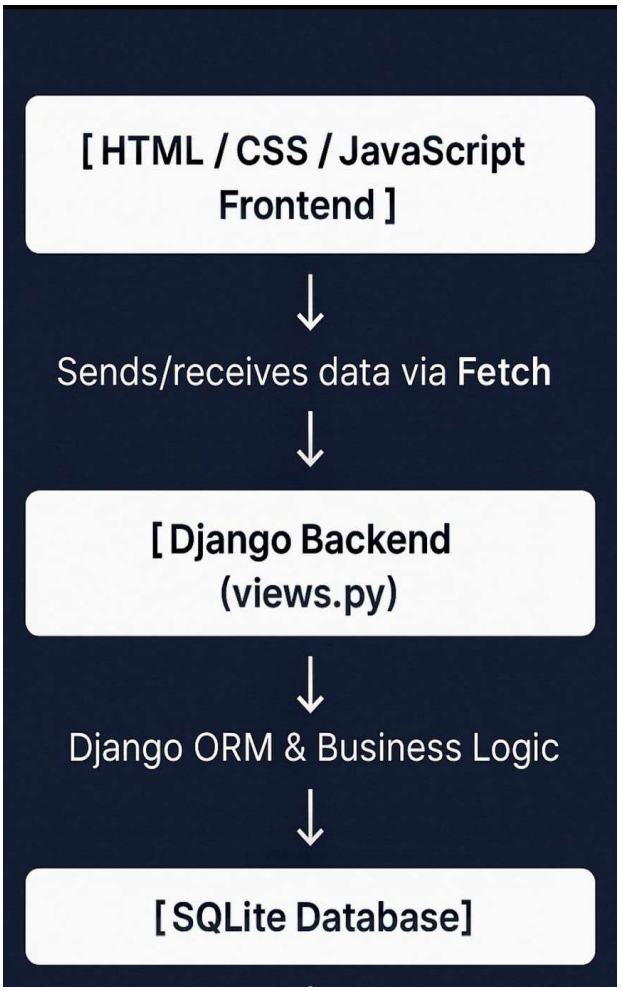

Fig 4.1: System Architecture Overview

Fig 4.1: The diagram illustrates the layered architecture of the Hostel Room Allotment System. The user interface, built using HTML, CSS, and JavaScript, interacts with the Django backend via Fetch API calls. The backend processes these requests using defined view functions and business logic in views.py. It then uses Django's Object-Relational Mapping (ORM) to communicate with the underlying SQLite database. This architecture ensures a clean separation of concerns, streamlined data flow, and efficient management of room allocation operations for a single-hostel setup.

4.2 DEVELOPMENT METHODOLOGY:

To build the Hostel Room Allotment System, we followed the Agile development method. This approach helped us build the project in small parts, test as we go, and improve things based on feedback. Agile is flexible, fast, and allows changes even in

the middle of development — which worked perfectly for our needs.

What We Did in Each Phase:

•Step-by-step Development (Sprints):

We broke the project into smaller parts or "sprints." In each sprint, we focused on one feature — like login, student info, room allocation, etc. After finishing each part, we tested it and improved based on the results.

•Modular Building:

We built the frontend, backend, and database separately. These parts were connected through APIs. This made it easier to test and update any part without affecting the others.

•Regular Testing & Integration:

We regularly saved our code to GitHub and checked that new features didn't break anything. Tools like Postman helped us test the backend APIs, and we used a browser to test the user interface.

•Easy Collaboration:

Our team used GitHub to share code, Google Docs for writing reports

Why We Chose Agile:

- We needed a working model early for reviews and demos.
- It allowed us to improve the system as we received feedback.
- We could easily add or remove features along the way.
- Team members could work on different parts at the same time.

4.3 DATABASE DESIGN:

The Hostel Room Allotment System utilizes a relational database model implemented using SQLite, which is tightly integrated with the Django ORM. This schema is designed for normalized, efficient, and scalable storage and retrieval of hostel data.

Core Tables and Relationships

1.Student Table

This table captures detailed information about each student.

•Fields: name, roll no, branch, admission_year, Email, phonenumber, password

•Constraints: roll no is assumed to be unique per student.

•Derived Field: current_year is calculated dynamically based on the admission year.

2.Rooms Table

This table maintains room-specific data and dynamically tracks occupancy.

•Fields: roomno, totalbeds

•Computed Fields:

•occupied – total number of allocated students

•remaining_beds – beds available for allotment

3.Room Allocation Table

Acts as a junction table mapping students to rooms.

•Fields: student (OneToOneField), room (ForeignKey), allocated_on

Relationships:

•A one-to-one relationship between Student and RoomAllocation ensures each student can be allocated only one room.

•A many-to-one relationship from RoomAllocation to Rooms means a room can have multiple allocations up to its capacity.

Relationship Summary

•1 Room → Many Students (up to bed capacity)

•1 Student → 1 Room (via one-to-one mapping)

Design Benefits

•Ensures data normalization and avoids redundancy.

•Enables complex queries for reporting, tracking allocations, and visual summaries.

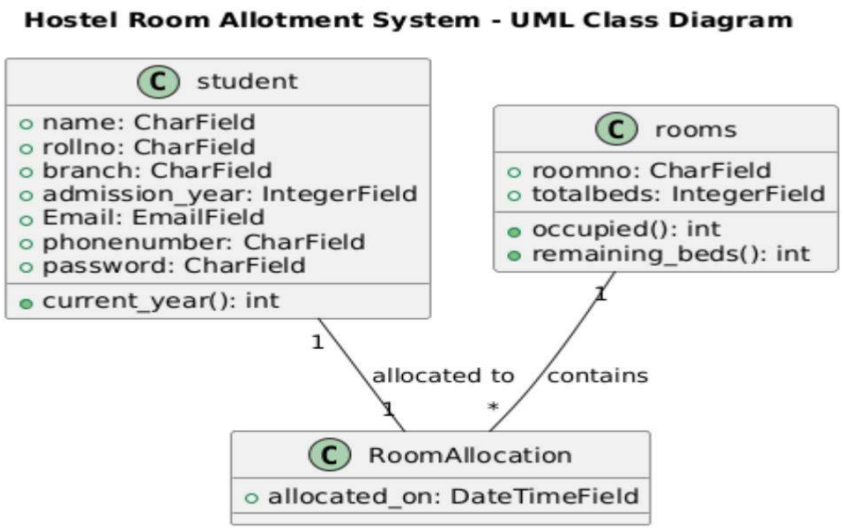

Fig 4.31 UML Class diagram

Fig 4.31: The UML Class Diagram illustrates the core structure of the Hostel Room Allotment System. It consists of three main classes: Student, Rooms, and RoomAllocation.

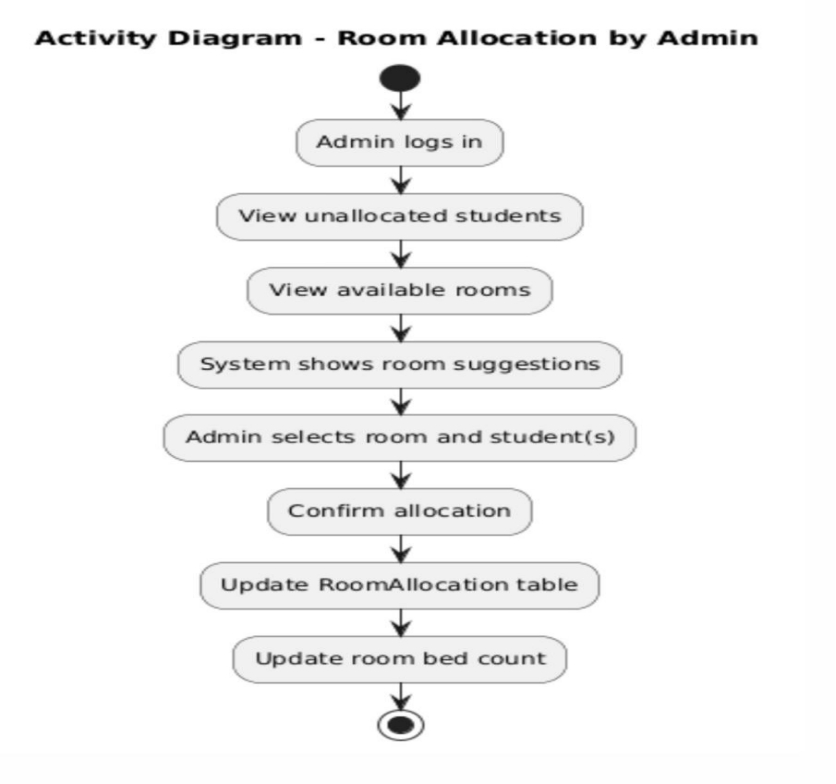

Fig 4.32 UML Activity diagram

Fig 4.32: Represents workflow or process logic using nodes and flows.Describes dynamic actions like login, room allocation, etc.Great for visualizing user interactions and system behavior.

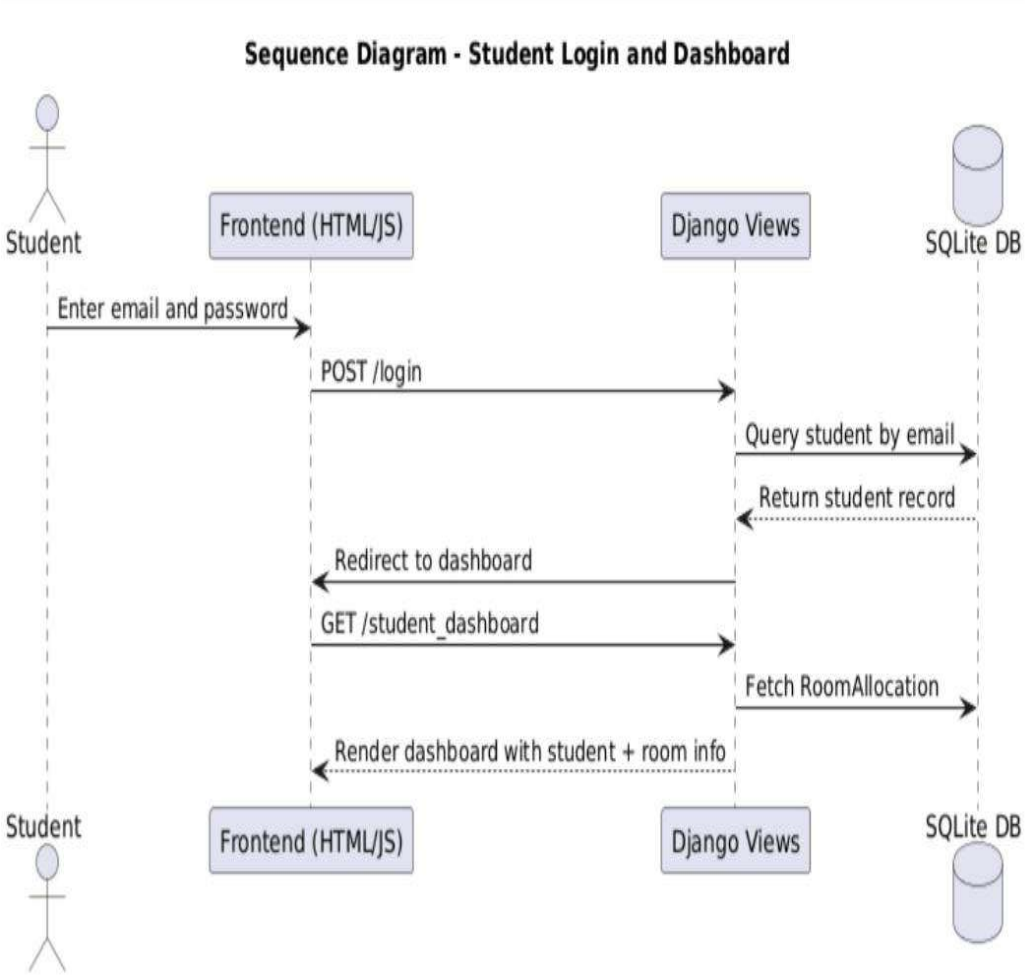

Fig 4.33 UML Sequence diagram

Fig 4.33: This sequence diagram represents the interaction between the student, frontend, Django backend, and SQLite database during the login and dashboard rendering process.

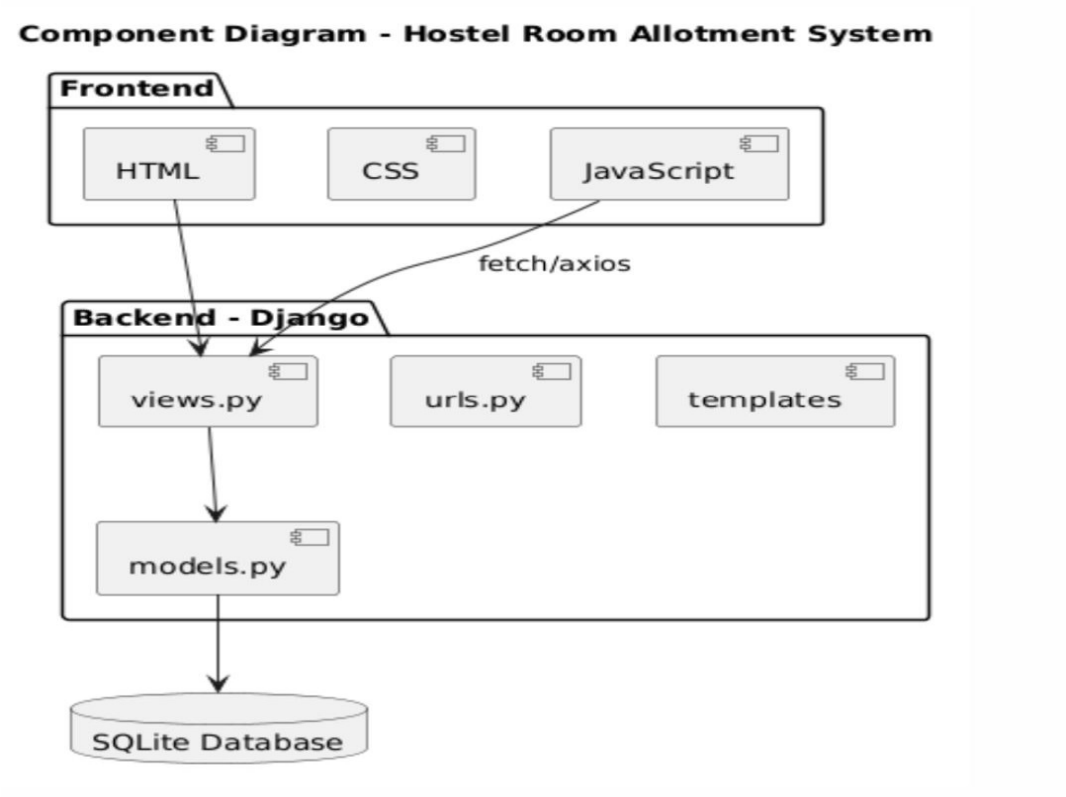

Fig 4.34 UML Component diagram

Fig 4.34: Breaks the system into functional parts or modules.Shows how components like login, room management, and dashboard connect.Helps with modular design and deployment planning.

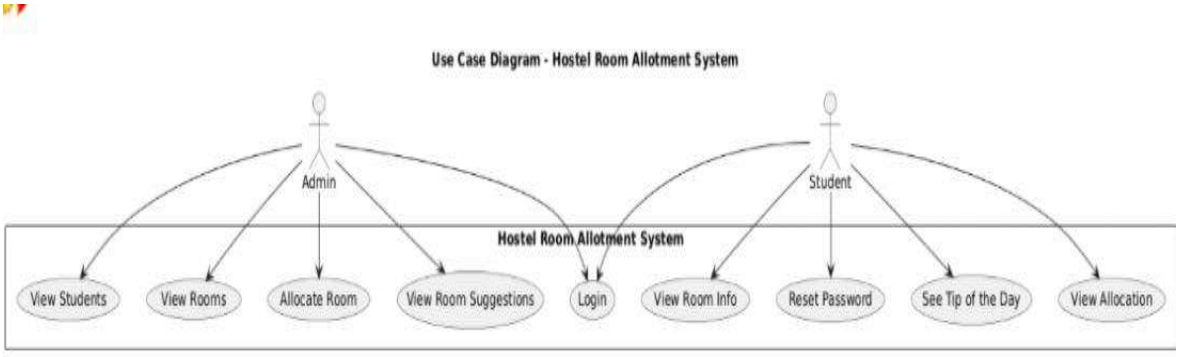

Fig 4.35 Use Case diagram

Fig 4.35: Illustrates system functionality from the user's perspective.Includes actors (Student, Admin) and their actions (allocate, view room).Helps clarify roles and what each user can do.

4.4 MODULES OVERVIEW:

The system is divided into the following main modules:

•Authentication Module: Handles login for both admin and students with role-based redirection.

•Student Module: Displays individual dashboards, room details, allocation info, and motivational tips.

•Admin Module: Includes student management, room management, room visualization, and automated room allocation.

•Room Allocation Module: Allows the admin to assign rooms to unallocated students with smart suggestions based on availability and department.

•Visualization Module: Displays bed occupancy status using interactive charts.

•Email Module: Sends password reset emails to students securely.

Each module is built with a focus on modularity, reusability, and separation of concerns to improve maintainability and scalability.

5. IMPLEMENTATION AND RESULTS

The hostel room allotment system was implemented using Django, React, and SQLite, with AR-based room visualization. It successfully enabled smooth login, smart room allocation, and interactive 3D viewing for both students and admins.

5.1 IMPLEMENTED FEATURES:

The Hostel Room Allotment System integrates a robust and modular architecture that combines user management, dynamic room allocation, and interactive dashboards to streamline the hostel management process. The application supports multiple user roles, primarily admin and students, with distinct access privileges and features tailored to each role.

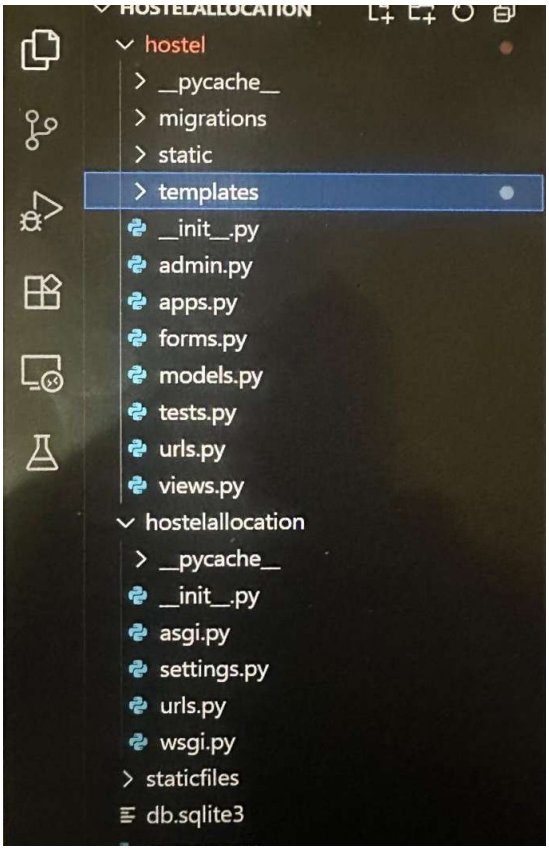

Fig 5.1.1 project Directory Structure

Fig 5.1.1 Project Directory Structure of Hostel Room Allotment System. This figure illustrates the modular organization of the Django-based application, where the hostel app contains business logic, templates, and models, while the hostel allocation folder manages project-level configurations. It follows Django's standard architecture and promotes maintainability and scalability.

```python
from django.urls import path
from .import views

urlpatterns = [
    path('',views.base,name='base'),
    path('homepage/', views.members, name="members"),
    path('login/', views.login_view, name='login'),
    path('admin_dashboard/', views.admin_dashboard, name='admin_dashboard'),
    path('student_dashboard/', views.student_dashboard, name='student_dashboard'),
    path('view-students/', views.view_students, name='view_students'),
    path('view-rooms/', views.view_rooms, name='view_rooms'),
    path('add-room/', views.add_room, name='add_room'),
    path('allocate-room/', views.allocate_room, name='allocate_room'),
    path('logout/', views.logout_view, name='logout'),
    path('view_details/',views.view_details,name='view_details'),
    path('forgot/',views.forgot,name='forgot'),
    path('room-chart/', views.room_chart, name='room_chart'),
    path('room_overview/', views.room_overview, name='room_overview'),
    path('room_info/', views.room_info, name='room_info'),
    path('manage-students/', views.manage_students, name='manage_students'),
    path('edit-student/', views.edit_student, name='add_student'),
    path('edit-student/<int:student_id>/', views.edit_student, name='edit_student'),
```

Fig 5.1.2 urls

Figure 5.1.2: Urls.py This figure shows the urls.py file, Entry point for routing and connects URL paths to the views(like login, dashboard..).

```python
class student(models.Model):
    name = models.CharField(max_length=50)
    rollno = models.CharField(max_length=50)
    branch = models.CharField(max_length=50)
    admission_year = models.IntegerField()
    Email = models.EmailField(help_text="Enter Your Email address")
    phonenumber = models.CharField(max_length=15)
    password = models.CharField(max_length=100, blank=True, null=True)

    @property
    def current_year(self):
        today = date.today()
        delta = today.year - self.admission_year
        if today.month < 7:
            delta -= 1
        return min(delta + 1, 4)
    def __str__(self):
        return f"{self.name} ({self.rollno})"

class rooms(models.Model):
    roomno=models.CharField(max_length=10,unique=True)
    totalbeds=models.IntegerField()
    @property
    def occupied(self):
        return RoomAllocation.objects.filter(room=self).count()
```

Fig 5.1.3 models

Fig 5.1.3 This figure shows models.py, It defines your database schema Student, Rooms, RoomAllocation.These models are used in views to fetch/store data.

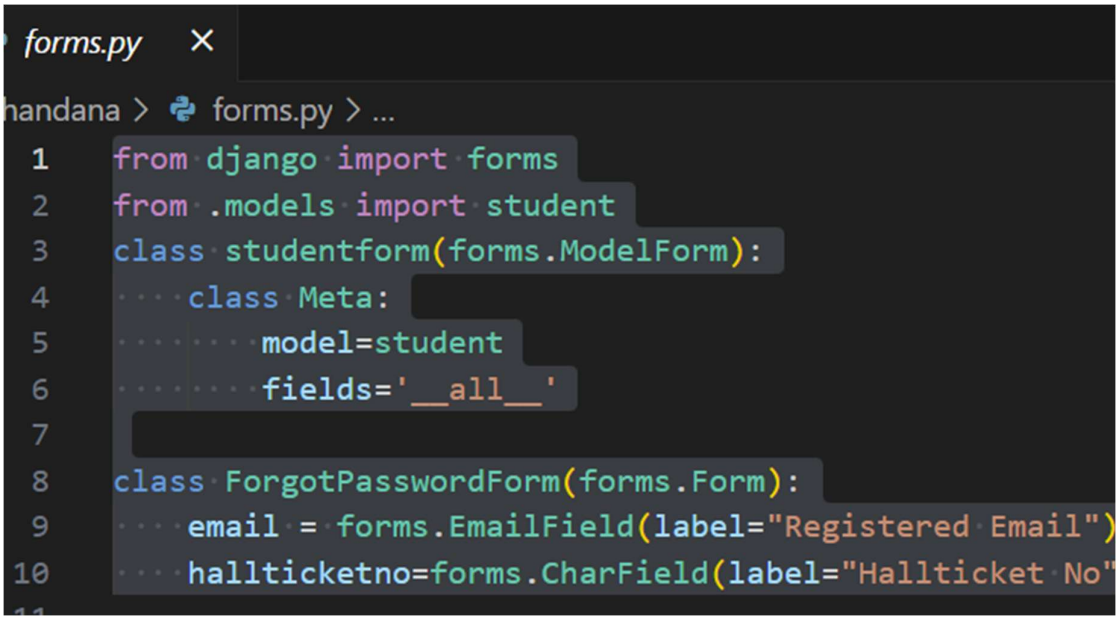

Fig 5.1.4:forms

Fig 5.1.4: This figure shows forms.py is used to create and manage forms using python classes and it connects models to HTML forms,handles input validation.

```
rom django.shortcuts import render, redirect, get_object_or_
rom django.contrib import messages
rom django.http import HttpResponse
rom .models import student, rooms, RoomAllocation
rom django.utils import timezone
mport matplotlib.pyplot as plt
mport matplotlib
rom .forms import studentform
atplotlib.use('Agg')
mport numpy as np
rom io import BytesIO
rom datetime import date
mport random
mport string
rom django.core.mail import send_mail
rom .forms import ForgotPasswordForm
rom django.contrib.auth import logout
```

Fig 5.1.5:views

Fig 5.1.5: In **views.py,** you import necessary modules like render, redirect, models, and forms to build logic for handling user requests and returning responses. Without these imports, Django can't recognize the functions or classes you're using**.**

```python
def base(request):
    return render(request, "base.html")
def members(request):
    return render(request, "homepage.html")
def admin_dashboard(request):
    return render(request, 'admin.html')
```

Fig 5.1.6:view function for page rendering

Fig 5.1.6: The **base(request)** view renders the main layout template shared across pages.The **members(request)** view displays student/member details fetched from the database.The **admin_dashboard(request)** view shows admin controls, room stats, and management tools.

```python
student_dashboard(request):
student_id = request.session.get('student_id')
student_obj = get_object_or_404(student, id=student_id)
allocation = RoomAllocation.objects.filter(student=student_obj).first()
return render(request, "student_dashboard.html", {
    "student": student_obj,
    "allocation": allocation
})
```

Fig 5.1.7 student_dashboard view function

Fig 5.1.7 : The **student_dashboard(request)** view loads the dashboard for logged-in students.It shows their room details, roommate info, and allotment status.This view helps students track and manage their hostel room information.

```python
def logout_view(request):
    logout(request)
    return redirect('base')
```

Fig 5.1.8 logout view function

Fig 5.1.8: The **logout_view(request)** logs out the current user securely. It clears the session and redirects the user to the login or homepage.This helps protect access and ensures proper session handling.

```python
def login_view(request):
    role = request.GET.get("role")
    if not role:
        return redirect('members')

    if request.method == "POST":
        username = request.POST.get("username", "").strip()
        password = request.POST.get("password", "").strip()

        if role == "admin":
            if username == "admin" and password == "admin123":
                request.session['admin_logged_in'] = True
                return redirect('admin_dashboard')
            else:
                messages.error(request, "Invalid admin credentials")

        elif role == "student":
            try:
                student_obj = student.objects.get(Email=username)
                if password == student_obj.rollno or password == student_obj.password:
                    request.session['student_id'] = student_obj.id
                    return redirect('student_dashboard')
                else:
                    messages.error(request, "Invalid student password")
            except student.DoesNotExist:
                messages.error(request, "Invalid student email")

    template = 'adminlogin.html' if role == "admin" else 'stulogin.h
    return render(request, template, {'role': role})
```

Snipping Tool

Screenshot copied to clipboard

Fig 5.1.9 login view function

Fig 5.1.9: The **login_view(request)** handles user login by verifying credentials. If valid, it creates a session and redirects based on the user role.If invalid, it shows an error and reloads the login page.

```python
def manage_students(request):
    students = student.objects.all()
    return render(request, 'manage_students.html', {'students': students})
def edit_student(request, student_id=None):
    if student_id:
        student_obj = get_object_or_404(student, id=student_id)
    else:
        student_obj = None

    if request.method == 'POST':
        form = studentform(request.POST, instance=student_obj)
        if form.is_valid():
            form.save()
            return redirect('manage_students')
    else:
        form = studentform(instance=student_obj)

    return render(request, 'edit_student.html', {'form': form, 'student_id': student_id})
```

Fig 5.1.10:manage students view function

Fig 5.1.10: The **manage_students(request)** view displays all students for the admin to review or update.The **edit_student(request, id)** view allows editing details of a specific student by ID.Together, they help the admin maintain accurate student records efficiently.

```python
def room_overview(request):
    all_rooms = rooms.objects.all()
    room_data = [{
        'sno': idx + 1,
        'id': room.id,
        'roomno': room.roomno,
        'occupied': room.occupied,
        'remaining': room.remaining_beds,
        'status': 'Full' if room.occupied == room.totalbeds else 'Available'
    } for idx, room in enumerate(all_rooms)]
    return render(request, 'roomdetvisual.html', {'rooms': room_data})
def room_chart(request):
    all_rooms = rooms.objects.all().order_by('roomno')
    labels = [room.roomno for room in all_rooms]
    occupied = [room.occupied for room in all_rooms]
    remaining = [room.remaining_beds for room in all_rooms]
    x = np.arange(len(labels))
    width = 0.35
    fig, ax = plt.subplots(figsize=(10, 5))
    ax.bar(x - width / 2, occupied, width, label='Occupied', color='#ff6666')
    ax.bar(x + width / 2, remaining, width, label='Remaining', color='#66b3ff')
    ax.set_ylabel('Beds')
    ax.set_title('Room Allocation Overview')
    ax.set_xticks(x)
    ax.set_xticklabels(labels, rotation=45)
    ax.legend()
    ax.grid(True, linestyle='--', alpha=0.4)
    plt.tight_layout()
    buffer = BytesIO()
    plt.savefig(buffer, format='png')
    plt.close(fig)
    buffer.seek(0)
    return HttpResponse(buffer.read(), content_type='image/png')
```

Fig 5.1.11:room overview view function

Fig 5.1.11: The **room_overview(request)** view presents a summary of all rooms in one place.It includes stats like total rooms, occupied, vacant, and capacity.Gives admins a quick visual of hostel room distribution.The **room_chart(request)** viewgenerates visual charts (bar graphs) of room data.

It helps visualize occupancy, vacancy.Useful for admins to analyze hostel usage trends at a glance.

```python
def allocate_room(request):
    if request.method == "POST":
        student_ids = request.POST.getlist("student_ids")
        room_id = request.POST.get("room_id")
        if not student_ids or not room_id:
            messages.error(request, "Please select students and a room.")
            return redirect('allocate_room')
        room_obj = get_object_or_404(rooms, id=room_id)
        new_students = []
        already_allocated = []
        for sid in student_ids:
            student_obj = get_object_or_404(student, id=sid)
            if RoomAllocation.objects.filter(student=student_obj).exists():
                already_allocated.append(student_obj.name)
            else:
                new_students.append(student_obj)
        if room_obj.remaining_beds < len(new_students):
            messages.error(request, f"Only {room_obj.remaining_beds} bed(s) available in Room {room_obj.r
            return redirect('allocate_room')
        for s in new_students:
            RoomAllocation.objects.create(student=s, room=room_obj, allocated_on=timezone.now())
        if new_students:
            messages.success(request, f"Room allocated to {len(new_students)} student(s).")
        if already_allocated:
            messages.warning(request, f"Skipped already allocated: {', '.join(already_allocated)}")
        return redirect('view_students')
    allocated_ids = RoomAllocation.objects.values_list('student_id', flat=True)
    students = student.objects.exclude(id__in=allocated_ids)
    rooms_list = [room for room in rooms.objects.all() if room.remaining_beds > 0]
    return render(request, 'allocate_room.html', {
        'students': students,
        'rooms': rooms_list,
    })
```

Fig 5.1.12 allocate room view function

Fig 5.1.12: The **allocate_room(request)** view lets the admin assign rooms to students. It checks room availability and updates both student and room records. Helps ensure each student gets a suitable, unoccupied room.

```python
ef add_room(request):
    if request.method == "POST":
        roomno = request.POST.get("room_no")
        totalbeds = request.POST.get("total_beds")
        rooms.objects.create(roomno=roomno, totalbeds=totalbeds)
        messages.success(request, "Room added successfully")
        return redirect('view_rooms')
    return render(request, 'add_room.html')
```

Fig 5.1.13 add room view function

Fig 5.1.13: The **add_room(request)** view allows the admin to add new rooms to the hostel database.

```python
def view_rooms(request):
    all_rooms = rooms.objects.all()
    room_data = [{
        'sno': idx + 1,
        'roomno': room.roomno,
        'occupied': room.occupied,
        'remaining': room.remaining_beds,
        'total': room.totalbeds
    } for idx, room in enumerate(all_rooms)]

    return render(request, 'view_rooms.html', {'rooms': room_dat
```

Fig 5.1.14 view rooms view function

Fig 5.1.14: The **view_rooms(request)** view shows all hostel rooms with details. It displays room number, capacity, and whether it's occupied. Useful for admins to track available and allotted rooms.

```python
def view_students(request):
    students = student.objects.all()
    student_data = []

    for s in students:
        allocation = RoomAllocation.objects.filter(student=s).first()
        student_data.append({
            'name': s.name,
            'rollno': s.rollno,
            'branch': s.branch,
            'year': s.current_year,
            'email': s.Email,
            'phone': s.phonenumber,
            'room': allocation.room.roomno if allocation else None
        })

    return render(request, 'view_students.html', {'student_data': student_data})
```

Fig 5.1.15:view students view function

Fig 5.1.15: The **view_students(request)** view lists all student records.It shows each student's details and assigned room (if any).Helps admins review and manage student allotment status.

```python
def forgot(request):
    if request.method == 'POST':
        form = ForgotPasswordForm(request.POST)
        if form.is_valid():
            email = form.cleaned_data['email'].strip()
            hno=form.cleaned_data['halltickетno'].strip()
            try:
                student_obj = student.objects.get(Email=email,rollno=hno)
                new_password = ''.join(random.choices(string.ascii_letters + string.digits, k=8))
                student_obj.password = new_password
                student_obj.save()

                send_mail(
                    subject='Your New Password',
                    message=f"Hello {student_obj.name},\n\nYour new password is: {new_password}\nUse it t
                    from_email='annadichandana744@gmail.com',
                    recipient_list=[email],
                    fail_silently=False,
                )

                messages.success(request, 'A new password has been sent to your email.')
                return render(request, 'password_reset_done.html')

            except student.DoesNotExist:
                messages.error(request, 'No student found with this email.')
            except Exception as e:
                messages.error(request, 'Error while sending the email.')
    else:
        form = ForgotPasswordForm()

    return render(request, 'forgot.html', {'form': form})
```

Fig 5.1.16:forgot password view function

Fig 5.1.16 The **forgot(request)** view handles password reset requests from users.It verifies the email and sends a reset link or code securely.Allows users to regain access if they forget their login credentials.

```python
def room_info(request):
    student_id = request.session.get('student_id')
    student_obj = get_object_or_404(student, id=student_id)
    allocation = RoomAllocation.objects.filter(student=student_obj).first()
    room = allocation.room if allocation else None
    roommates = RoomAllocation.objects.filter(room=room).exclude(student=student_obj) if room else []
    return render(request, "room_info.html", {
        "student": student_obj,
        "room": room,
        "roommates": roommates
    })
```

Fig 5.1.17 room info view function

Fig 5.1.17: The **room_info(request)** view displays detailed data about each room.It includes room number, capacity, occupants, and status.

```
    student_id = request.session.get('student_id')
    student_obj = get_object_or_404(student, id=student_id)
    allocation = RoomAllocation.objects.filter(student=student_obj).first()

    tips = [
        "Stay hydrated throughout the day.",
        "Attend all your classes for maximum benefit.",
        "Back up your work regularly.",
        "Use the library for peaceful study.",
        "Don't forget to network with your batchmates.",
        "Organize your tasks with a planner.",
        "Sleep well before exams – it improves memory!"
    ]

    tip_of_the_day = tips[random.randint(1,7)]

    return render(request, "view_details.html", {
        "student": student_obj,
        "allocation": allocation,
        "tip_of_the_day": tip_of_the_day
    })
```

Fig 5.1.18 view details view function

Fig 5.1.18: The **view_details(request)** view shows detailed information of the logged-in user.It includes profile, room, and roommate data in a single page.Helps users quickly review their hostel-related information.

```
def edit_profile(request):
    student_id = request.session.get('student_id')
    if not student_id:
        return redirect('login')

    student_obj = student.objects.get(id=student_id)

    if request.method == "POST":
        student_obj.name = request.POST.get("name")
        student_obj.rollno = request.POST.get("rollno")
        student_obj.branch = request.POST.get("branch")
        student_obj.admission_year = request.POST.get("admission_year")
        student_obj.Email = request.POST.get("email")
        student_obj.phonenumber = request.POST.get("phone")
        student_obj.save()
        return redirect('student_dashboard')

    return render(request, 'edit_profile.html', {'student': student_obj})
```

Fig 5.1.19 edit student profile view function

Fig 5.1.19: The **edit_profile(request)** view allows a logged-in user to update their personal details.It loads a form with existing data and saves changes upon submission.Ensures students or admins can keep their profile info accurate.

5.2 RESULTS:

The Hostel Room Allotment System successfully achieved its core objectives and delivered the following key results:

•Students were able to log in, view their personal and room details, and even see their roommates.

•Admins were able to:

•View/add/edit students

•Add rooms and track room occupancy

•Allocate rooms based on real-time bed availability

•An interactive occupancy chart was generated showing the current room-wise status.

•Students received password reset emails via the built-in forgot password feature.

• Role-based login and logout worked seamlessly, ensuring session security.

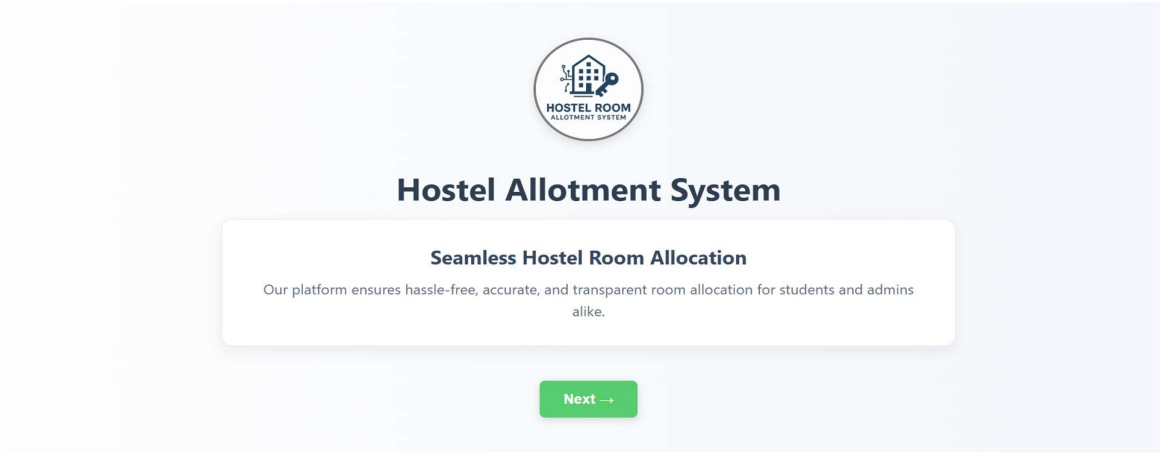

Fig 5.2.1 Landing page

Fig. 5.2.1: Landing Page displays the entry screen of the project, featuring a concise project introduction.It offers an intuitive, user- friendly interface that encourages smooth navigation and engagement with the system.

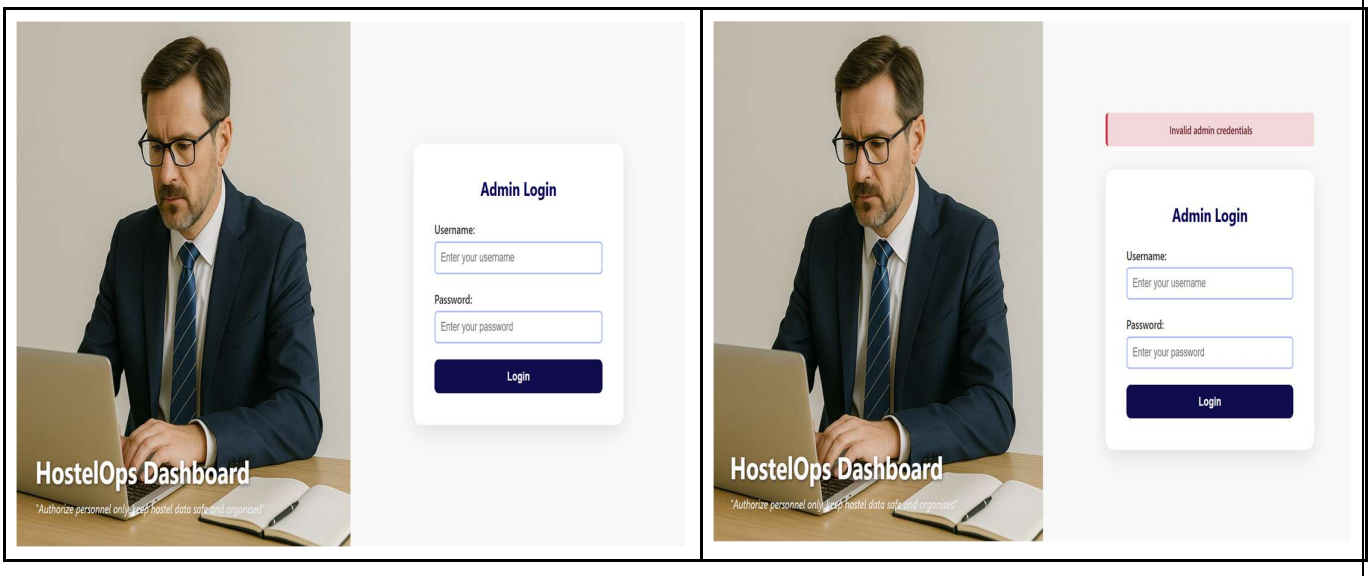

Fig. 5.2.2: Admin Login Page

Fig. 5.2.2: Admin Login Page The system includes a secure, role-based login page for administrators. Admins authenticate using a valid username and password, verified through Django's authentication system. Upon successful login, they are redirected to the Admin Dashboard. Session-based login ensures only authorized users can access administrative features.

3

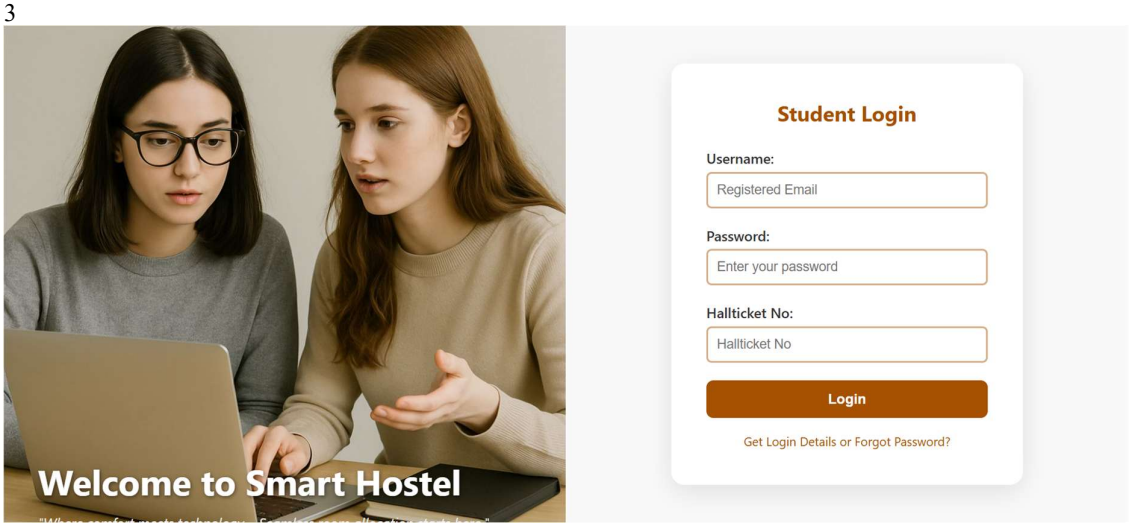

Fig. 5.2.3 : Student Login page

Fig. 5.2.3: Login page Students log in using their registered email and either their hall ticket number (for first-time login) or a personal password. The system checks these against stored records to ensure secure access. A “Forgot Password” option allows users to reset credentials via email. First-time users receive login details using Get Login Details, enabling smooth onboarding.

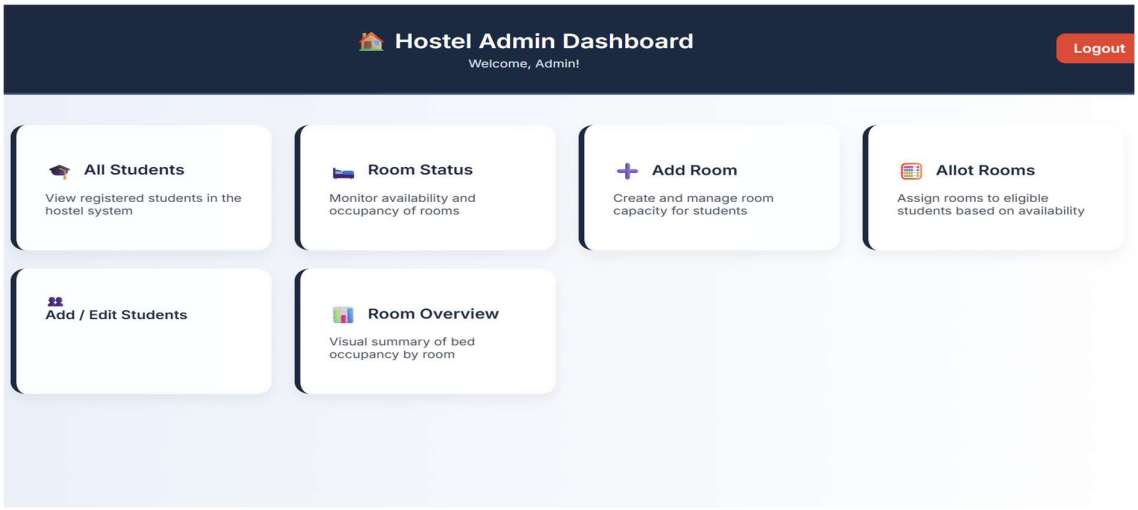

Fig. 5.2.4: ADMIN DASHBOARD

Fig 5.2.4: The Admin Dashboard provides an overview of all hostel activities in one place.displays all registered students with filters like branch, year, and allocation status, along with a keyword search option. Admins can view room status, add new rooms, allot rooms, and edit student details. A room overview section shows occupancy and availability visually.

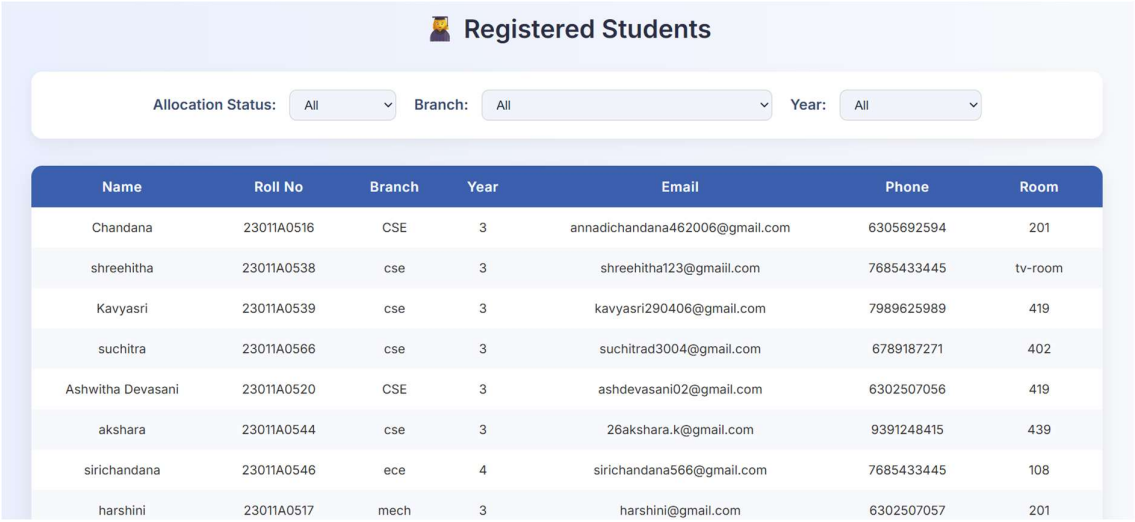

Fig. 5.2.5: Registered students

Fig. 5.2.5: Registered students have information about all the registered students in the hostel info like(Name,Rollno,etc…)**.**

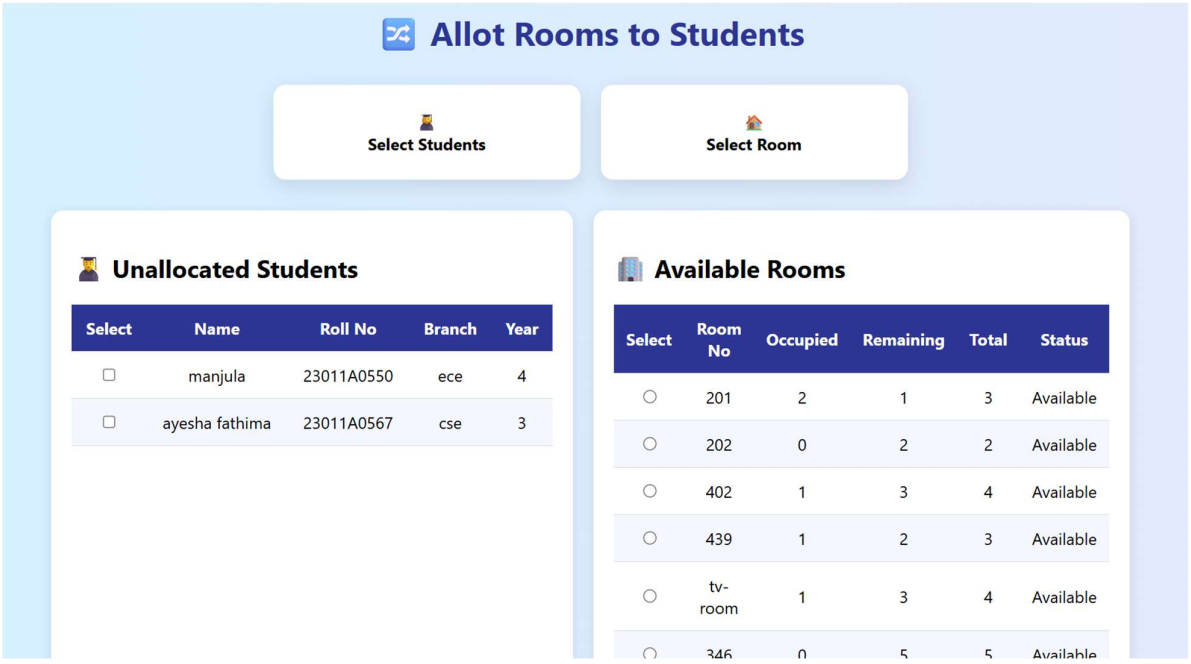

Fig. 5.2.6: Allot Rooms

Fig. 5.2.6: In Allot Rooms The admin can allocate rooms by first selecting unallocated students from a list. A table displays students who haven't been assigned a room yet. then the admin selects a room from the available rooms list, which shows details like room number and remaining beds. The system ensures only rooms with enough beds are selectable. Once confirmed, the selected students are assigned to the room immediately**.**

Fig 5.2.7:Manage Students

Fig 5.2.7:Manage Students allows the admin to view, search, add, and edit student profiles efficiently.Admins can filter students by branch, year, or allocation status for better control.This ensures smooth handling of all student data required for room allotment.

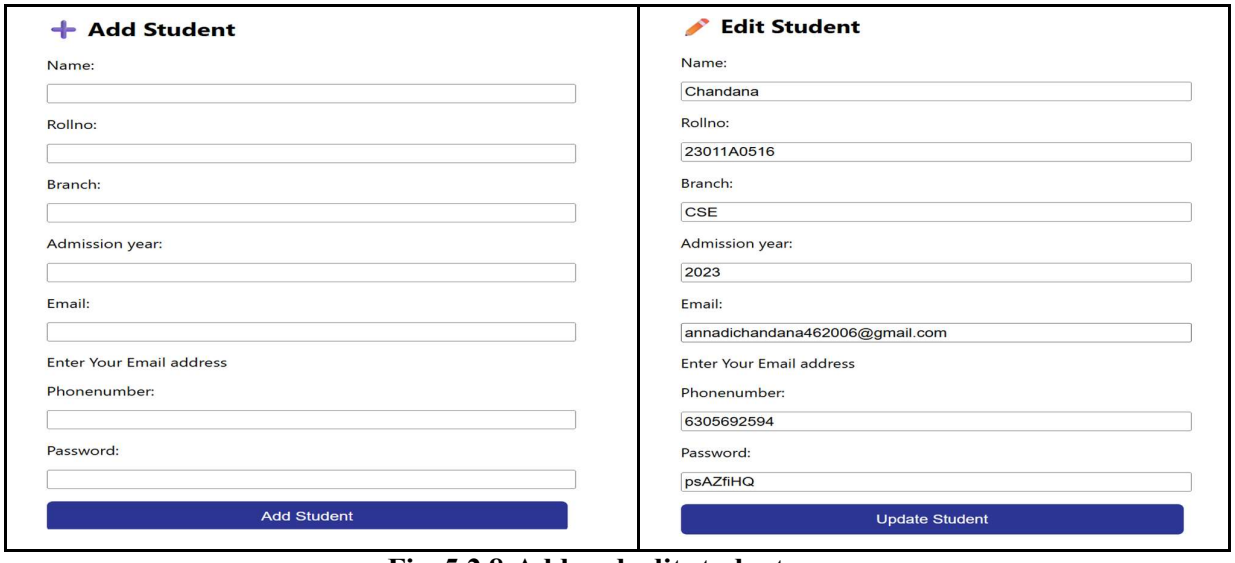

Fig. 5.2.8:Add and edit students

Fig. 5.2.8: Admins can add students by filling in details like name, roll number, branch, and contact info.They can also edit existing student records to correct or update any information.This helps keep the student database accurate and up-to-date**.**

Room Details

S.No	Room No	Occupied	Remaining	Total Beds	Status
1	201	2	1	3	Partially Occupied
2	202	0	2	2	Empty
3	402	1	3	4	Partially Occupied
4	439	1	2	3	Partially Occupied
5	tv-room	1	3	4	Partially Occupied
6	346	0	5	5	Empty
7	219	0	3	3	Empty
8	234	0	5	5	Empty
9	220	0	3	3	Empty
10	217	0	3	3	Empty

Fig 5.2.9: Room Details

Fig 5.2.9: Room Details displays all available rooms with information like room number, total beds, occupied beds, and remaining capacity.Admins can add new rooms and monitor occupancy status in real-time.This helps ensure efficient room usage and avoids overbooking.

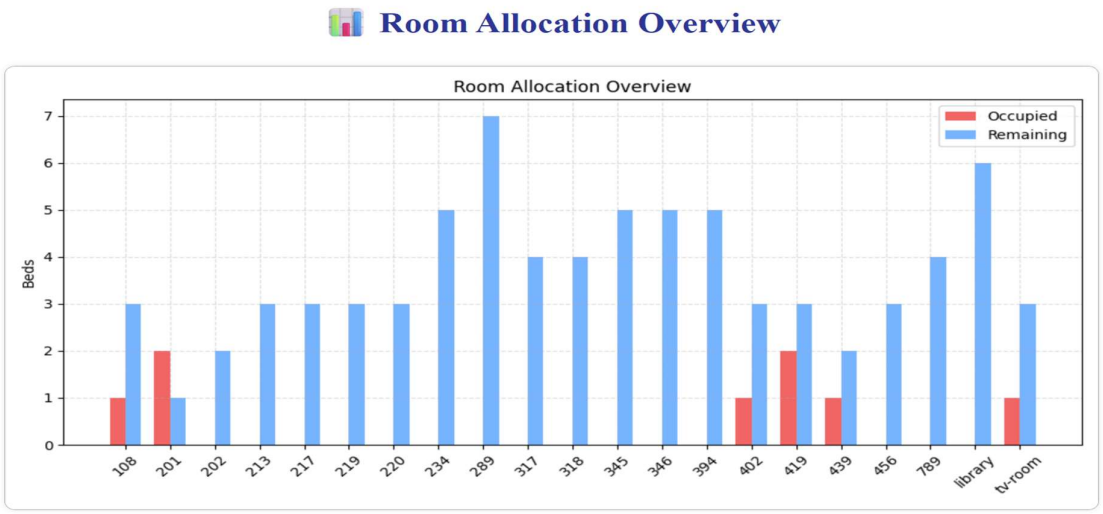

Fig 5.2.10: Room Overview

Fig 5.2.10: Room-Overview will give details of occupied vs remaining as bar chart.

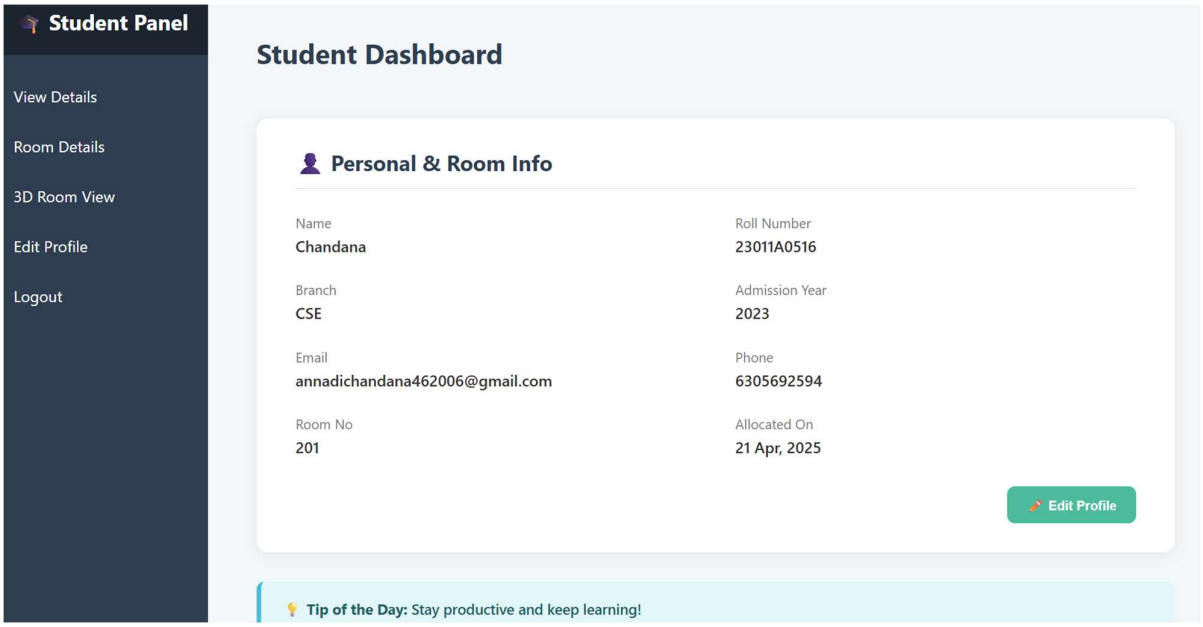

Fig 5.2.11 Student Dashboard

Fig. 5.2.11: Student Dashboard shows the student's personal details and their allotted room information after login.It also displays roommate names and includes a helpful academic tip of the day.The dashboard offers a simple, personalized view for students to stay informed.as shown in fig 7.11.

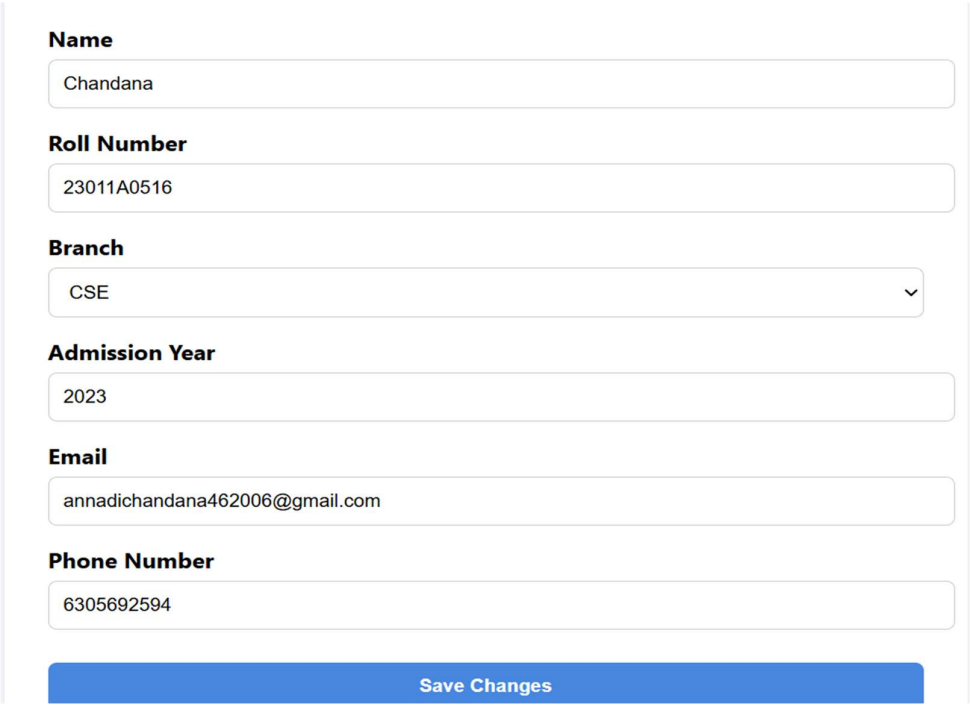

Fig. 5.2.12:Edit Profile

Fig. 5.2.12:Edit Profile where students edit their details and save changes .

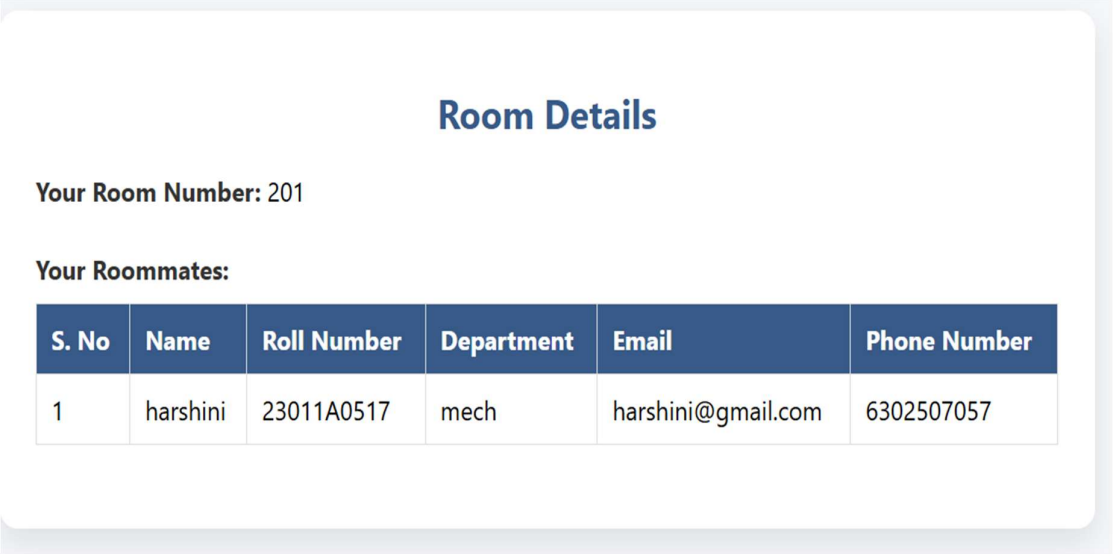

Fig. 5.2.13: Room Details

Fig. 5.2.13: Room Details includes Student Room Number and the roommates of the student who are allotted to the same room so that the student can have contact with the roommates.

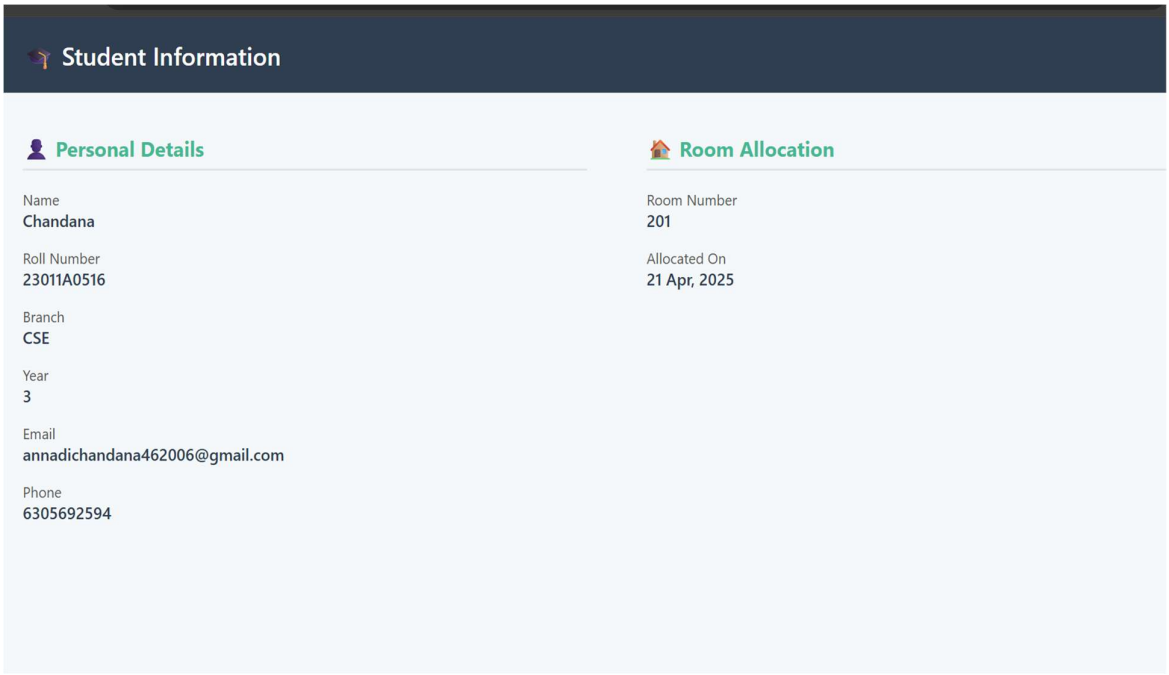

Fig 5.2.14:Student Information

Fig 5.2.14:Student Information includes key personal details such as name, roll number, email, phone number, branch, and academic year.These details are visible in the student dashboard and are also used during room allocation.Maintaining accurate student data ensures proper room assignments and communication.

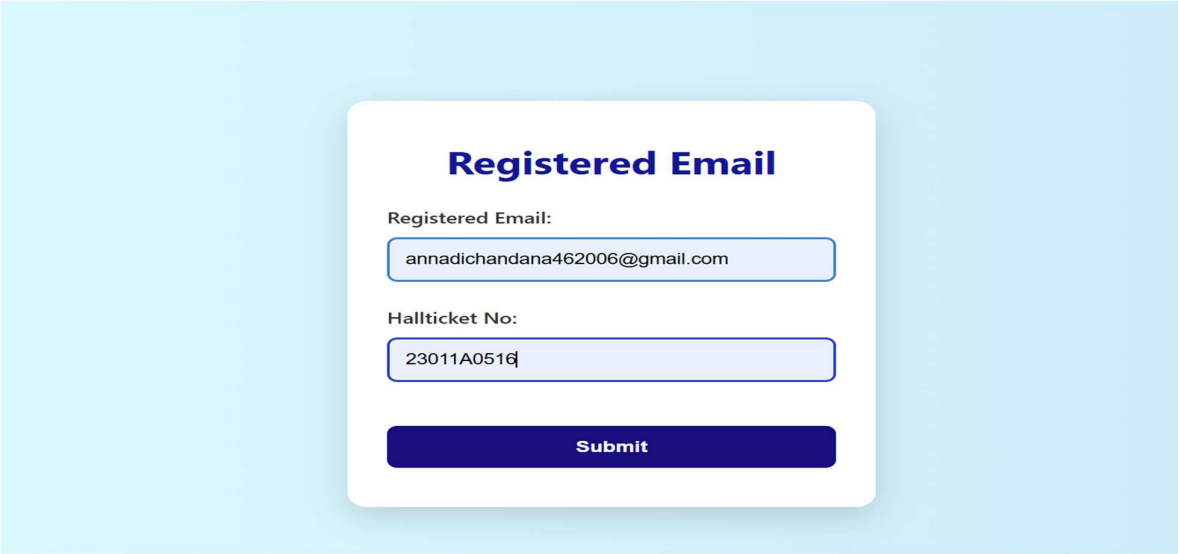

Fig 5.2.15:forgot password

Fig 5.2.15: This screen prompts the user to enter their registered email address to initiate the password reset process. Once submitted, the system verifies the email and sends a reset link if it's valid.

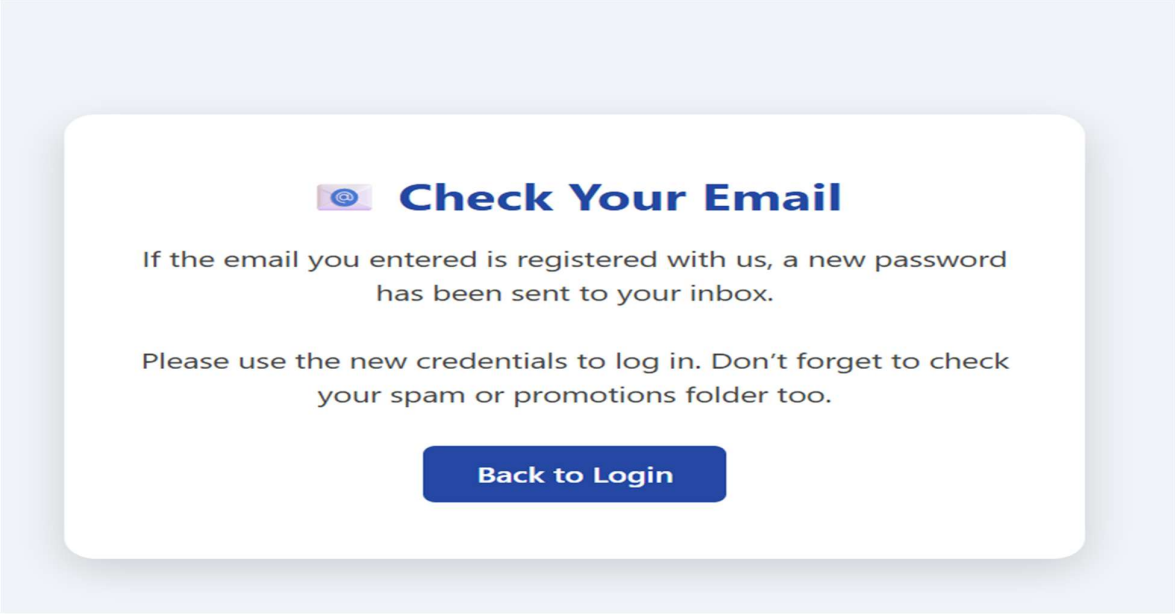

Fig 5.2.16:Confirmation Message box

Fig 5.2.16: This Screen then confirms that the reset email has been sent and instructs the user to check their inbox.

Fig 5.2.17:mail box

Fig 5.2.17: This figure shows the password reset email sent to the user after submitting a valid registered email. It contains a newly generated password for login and advises the user to change it later. This ensures secure account recovery in the Hostel Allocation System.

6.CONCLUSION AND FUTURE SCOPE

The system efficiently automates hostel room allotment, reducing manual work and errors. In the future, features like real-time notifications and AI-based room suggestions can further enhance usability.

6.1 CONCLUSION

The Hostel Room Allotment System was developed to simplify and digitalize the process of assigning hostel rooms to students in an educational institution. Traditional methods—often manual and paper-based—are time-consuming, prone to human error, and lack transparency. Our project resolves these issues by providing an automated, centralized platform that streamlines every aspect of the room allocation process. Built using Django (Python) for the backend and SQLite as the database, the system ensures robust data handling, scalability, and quick development. The frontend, designed using HTML, CSS, and JavaScript, offers a clean, responsive, and user-friendly interface for both administrators and students.

The platform features role-based login, allowing students to securely view their room details, personal information, and roommate information. For administrators, the system provides tools to add new rooms, track room occupancy, view all student records, and visualize room status using interactive charts. Additionally, password recovery through email enhances the user experience and ensures account security.

The project meets its core objective of replacing manual room allocation with a modern, intuitive, and efficient web-based system. It provides real-time data, minimizes errors, and improves transparency for all stakeholders involved. The system is particularly well-suited for individual hostels and can be further extended to accommodate multiple hostels or additional features in the future.

6.2 UNIQUE SELLING POINT (USP)

Unlike traditional allotment systems that rely on manual processes or static forms, our Hostel Room Allotment System stands out by offering:

•Automated & Smart Allocation: Reduces admin workload by suggesting optimal room allocations based on seat availability and student details like branch or year.

•Real-time Room Insights: Displays up-to-date occupancy, availability, and roommate info, ensuring transparency for students and staff.

•Simple, Role-based Access: Clean and secure login system with personalized dashboards for students and admin.

•Visual Dashboards: Integrated charts and tables for room occupancy and student distribution, aiding faster decisions.

•AR/3D Room Preview (Optional Extension): An immersive way for students to explore their allocated rooms virtually using WebXR and Three.js.

•Smart Notifications: Email-based password recovery and future potential for allotment alerts and reminders.

6.3 . 🔭 FUTURE SCOPE

•**Automated Smart Allotment**: Add intelligent algorithms to suggest optimal room assignments based on branch, year, preferences, or gender.

•**Multi-hostel support:** Extend the system to manage multiple hostels or campuses under a unified admin dashboard.

•**AR-based Room Tour:** Enhance room visualization using WebXR and 3D walkthroughs for an immersive student preview experience.

•**Admin Analytics Dashboard:** Add insights like allocation efficiency, vacancy trends, or heatmaps for decision-making.

•**Notification System:** Integrate email/SMS alerts for room allotment, deadline reminders, and announcement

REFERENCES:

The References that we used throughout this project:

Django Documentation. Available at: https://docs.djangoproject.com/

Python Documentation (v3.x). Available at: https://docs.python.org/3/

SQLite Documentation. Available at: https://www.sqlite.org/docs.html

Git Documentation. Available at: https://git-scm.com/doc

GitHub Docs. Version Control and Collaboration Guide. Available at: https://docs.github.com/