
PROJECT REPORT

for

HostelDesk

Version 1.0

Prepared by

Group No 3

Sanil Mishra
Daruri Kaustubh
Gowri B Kumar
Hari Narayanan J
Kaustubh Bahl
Juby Johnson
Elsa Mary Jibiachen
Manideep Reddy
Mohamed Afthab E K
Solanki Himalay Harijvan
Gigil James

B200057CS
B200825CS
B200699CS
B200741CS
B200833CS
B200051CS
B200049CS
B200823CS
B200719CS
B200855CS
B200771CS

sanil_b200057cs@nitc.ac.in
daruri_b200825cs@nitc.ac.in
gowri_b200699cs@nitc.ac.in
harinarayan_b200741cs@nitc.ac.in
kaustubh_b200833cs@nitc.ac.in
juby_b200051cs@nitc.ac.in
elsa_b200049cs@nitc.ac.in
belaganti_b200823cs@nitc.ac.in
mohamed_b200719cs@nitc.ac.in
solanki_b200855cs@nitc.ac.in
gigil_b200771cs@nitc.ac.in

Instructor: Dr Pranesh Das

Course: CS3002D: DATABASE MANAGEMENT
SYSTEMS

Date: 09-12-22

1 Table of Contents

1	Table of Contents	II
2	Acknowledgement	III
3	Purpose	1
4	Assumptions	3
5	Overall Description	4
6	Database-wide Design Decision	5
7	Database Administrative Functions	8
8	System Requirement Specification	11
9	Hardware Requirements	13
10	Graphical User Interface	14
11	Backend Code	27
12	References	55

2 Acknowledgement

We would like to extend our sincere gratitude to our Database and Management System faculty Dr Pranesh Das who has helped us in this endeavour and has always been very cooperative; without his help, cooperation, guidance and encouragement, the project couldn't have been what it evolved to be.

We would like to express our special gratitude to the Computer Science and Engineering Department which gave us this golden opportunity to do this wonderful project. It helped us learn many new topics and gave us a chance to research too.

We extend our heartfelt thanks to our faculty for their guidance and constant supervision, as well as, for providing us the necessary information regarding the project.

We are also thankful to our parents for their cooperation and encouragement.

3 Purpose

This Database Design Document for the Hostel and Mess allocation system establishes a target database management system identified from the analysis of the requirements of the software system, maintaining data consistency and integrity. The Entity-Relational model, thus created by analysing the use case diagram, is converted to a relational schema of the target Database Management System (DBMS).

3.1 Document Objectives

The Database Design Document has the following objectives:

1. To outline the software design and specification of the Hostel and Mess Allocation system database and the system architecture and components that users or system developers can access via a Database Management System.
2. To provide a fundamental approach for implementing the database and related software units, thus aiding in extracting details necessary for the application's software development.

3.2 Intended Audience and Document Overview

This document is written for the perusal of:

- Technical reviewers for assuring and evaluating the quality of the document.
- Architects whose overall architecture design must meet the requirements specified in this document.
- Designers whose design must meet the requirements specified in this document.
- Developers for implementing as per the software requirements specified in this document.
- Quality Assurance personnel for testing and validating the requirements given in this document.

The next section of the document, Assumptions and Constraints, explains certain preconceived notions and conditions that are assumed to be true. Constraints elucidate limitations such as user restrictions, proof of actions, etcetera.

The third section - Database wide design, focuses on describing the system's behaviour, explaining the significant roles/actions along with the details of the DBMS platform, security requirements, performance and availability decisions.

The fourth section, Database Administrative Functions, displays the Entity-Relationship Model created, the relational schema formed from the ER diagram with the normalization and data formats details.

4 Assumptions

1. The hostel and mess authority consist of three independent bodies: Administrator, Hostel Manager and Mess Manager.
2. The Administrator has the right to appoint the Hostel Manager and assign him as the hostel incharge.
3. The Administrator has the right to appoint the Mess Manager and assign him the mess incharge.
4. A Hostel Manager manages only one hostel.
5. A Mess Manager manages only one mess.
6. Students can apply for rooms by giving preferences; rooms will be allocated sequentially based on their requirements.
7. The Hostel Manager can confirm or reject the application.
8. Students can apply for the mess by giving preferences, and the allotment will be based on the availability of the mess.
9. The Mess Manager can confirm or reject the application.
10. A student can apply for the room and mess only once.

5 Overall Description

5.1 Project Overview

HostelDesk is a web application that mainly aims to automate the hostel room allocation and the process of choosing a mess. Currently, our students are filling up forms and submitting them in the respective hostel offices, which involves much paperwork and is less efficient.

5.2 Product Functionality

The Web Application has the following:

1. Administrators
2. Hostel Managers
3. Mess Managers
4. Students

Students can apply for room allocation and are allotted rooms according to vacancy. The Administrator appoints Hostel Managers. Students can also choose their mess through the portal. Mess Managers can, in return, see the students registered under their mess. The Administrator also appoints Mess Managers.

6 Database-Wide Design Decisions

6.1 Behaviour

6.1.1 Login and Sign Up

Users can log in or sign up for the website from the homepage. Students who don't have an account can sign up with their institute roll number and password under the signup tab. Mess and hostel managers, on the other hand, are not allowed to register on their own. They must be added to the system by the administrator.

A student logs in to the website using the institute roll number and password, whereas hostel managers, mess managers and the administrator use their employee ID and password. The users must select their roles as well. If the credentials and the role match with any user in the system, they get logged in successfully.

After logging in (or signing up), the users will be directed to their respective pages per their role.

The application provides the following roles with corresponding functionalities. These roles are as follows:

6.1.2 Administrator

- a. Manages the hostel and mess manager and is responsible for their appointment and removal
- b. Has view access over student details
- c. Can permanently expel students from the hostel
- d. Can change their password

6.1.3 Hostel Manager

- a. Manages the working of the hostel by monitoring various student requests
- b. Has view access to details of the students enrolled in the hostel by Roll number and Room number.
- c. Can vacate the rooms according to the requests made by the students.
- d. Can change their password

6.1.4 Mess Manager

- a. Has view access over the details of students enrolled in their corresponding mess
- b. Can download the CSV file containing the details of all enrolled students
- c. Can change their password

6.1.5 Student

- a. Apply for hostel allocation or vacation
- b. Choose their mess at the end of each month
- c. View their profile and edit details.
- d. Can change their password

6.2 DBMS Platform

HostelDesk is a web application that provides users with a clear and interactive experience. The design is simple, and all the interfaces follow a standard template. The web application is expected to work on web browsers. The application allows users to log into the system with corresponding credentials and is directed to different pages according to their roles. The functionalities extended to various users differ by their roles.

6.3 Security Requirements

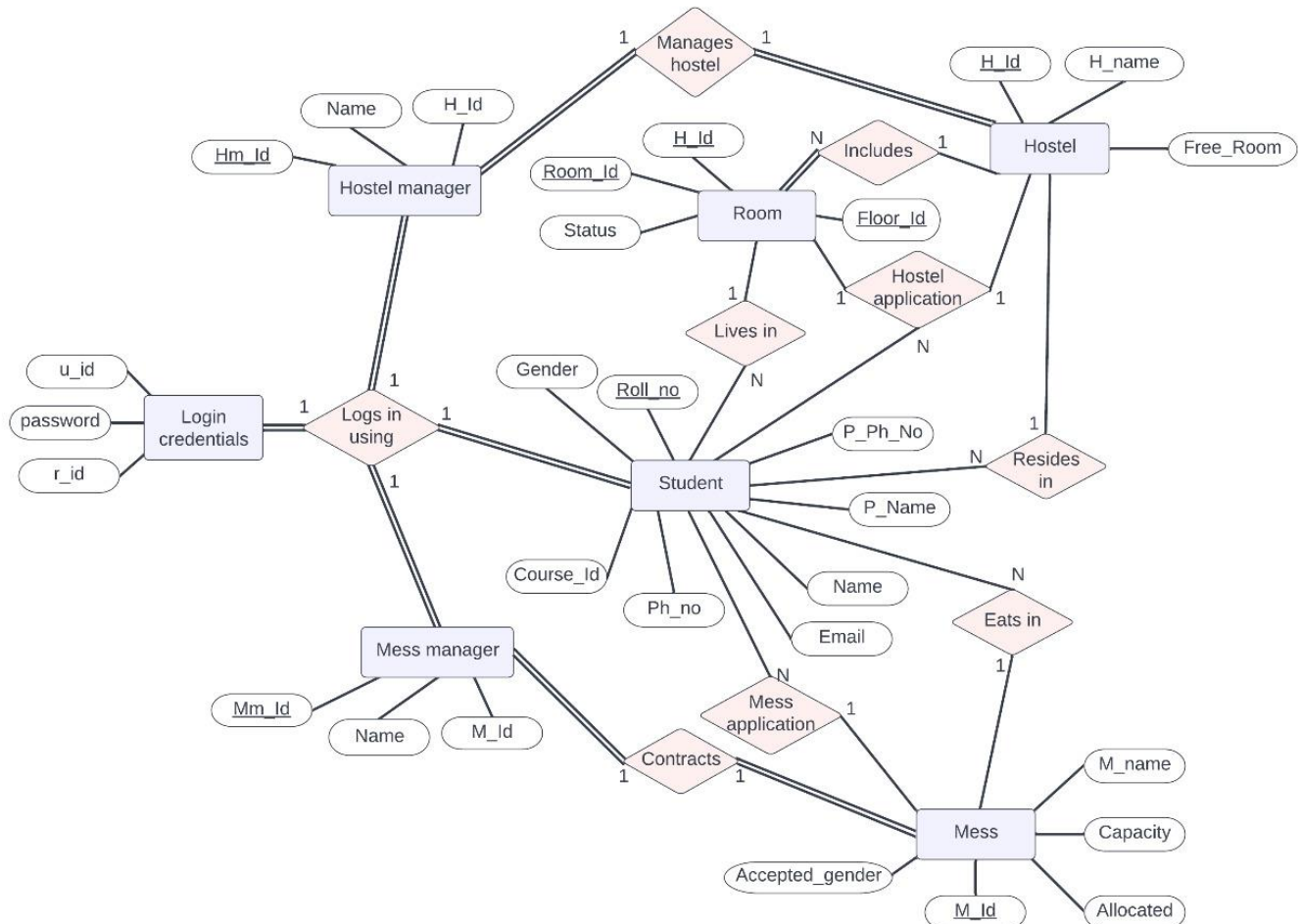
The system will store all the data in a secure database. The interacting users will be able to view information but will not have the privilege to modify/edit it. This privilege will be given to the chief warden (admin), and only they have the right to update the database. These are the two different types of accessors and have varying access constraints. In terms of the safety aspect, the system does not pose a threat to its users. To combat attacks by malware, backing up the database is advised.

6.4 Performance and Availability Decisions

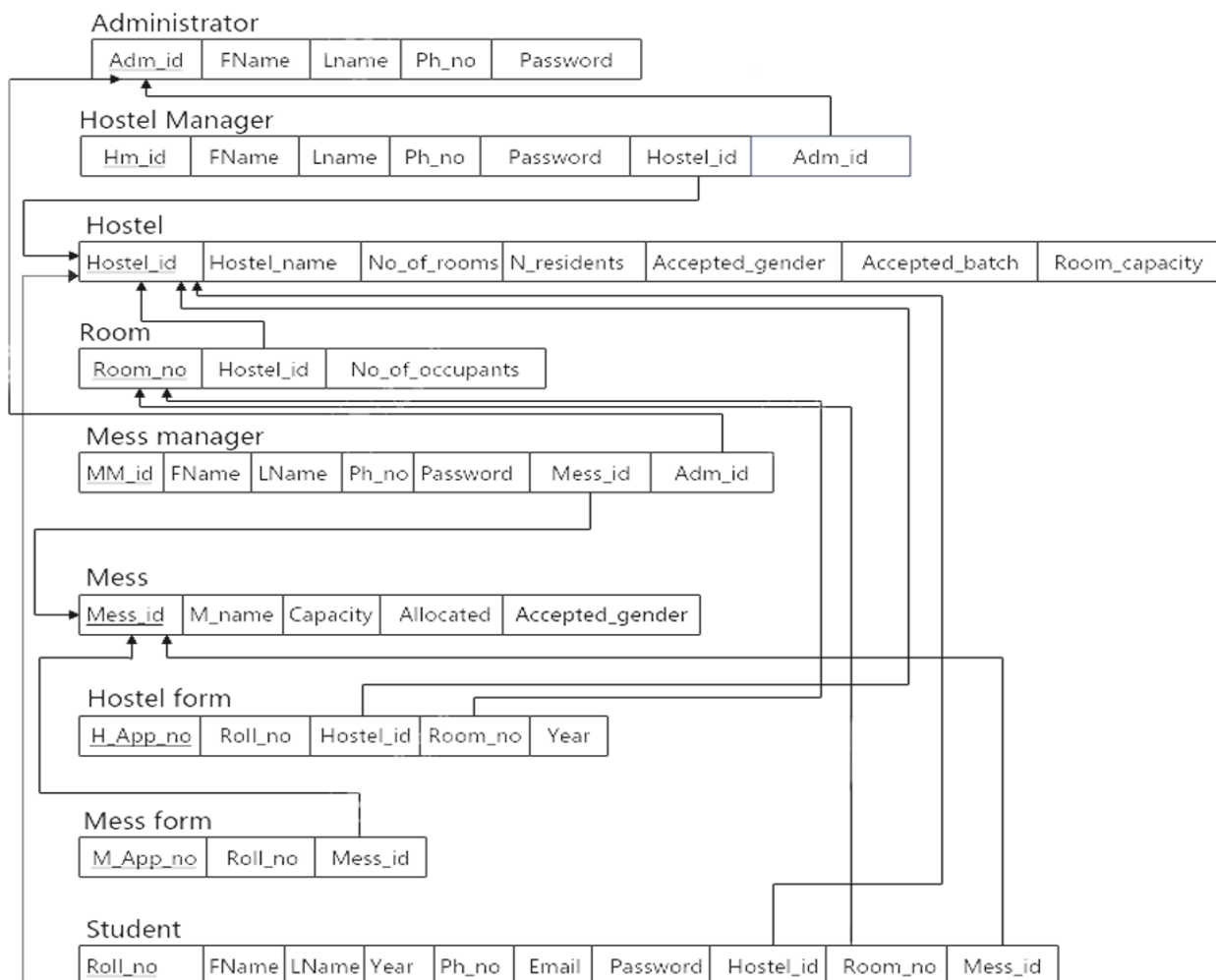
The search retrievals depend upon the updates made to the system. This system is designed to interact with students and staff. The system will respond to the user within less than a second of submitting a request. Overall, the performance will be fast and accurate. The system will be capable of handling a large amount of data and hence accommodate a high number of complaints, user credentials, etc.

7 DATABASE ADMINISTRATIVE FUNCTIONS

7.1 Entity-Relation Model



7.2 Relational Schema



7.3 Normalization

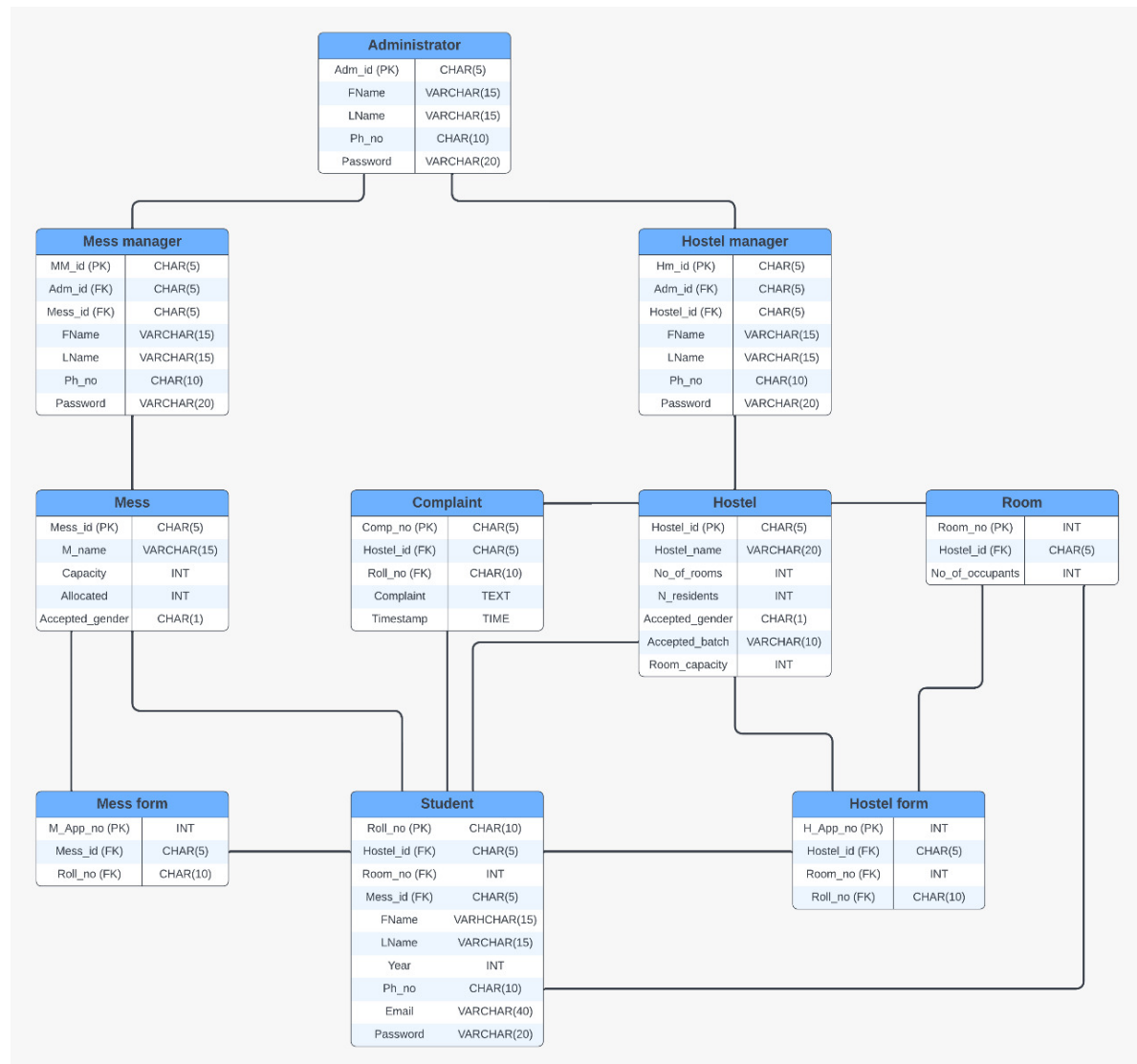
• 1NF - The tables are in 1NF, as there are no multivalued or composite attributes. Each table cell contains atomic values, and each record is unique. Hence the database is 1NF normalized.

1NF normalized.

• 2NF - The tables are already in 1NF as proved above. There are no partial Dependencies: No non-prime keys are solely dependent on only one part of a primary key in any of the tables. Hence the database is 2NF normalized.

• 3NF - The tables are already in 2NF as proved above. There are no transitive functional dependencies in the schema. There are no non-prime keys that are dependent on another non-prime key in any specific table. Hence the database is 3NF normalized.

7.4 Schema Description and Data Formats



8 System Requirement Specification

8.1 Functional System Requirements

This section gives the functional requirements that apply to the HostelDesk. These are the sub-modules in this phase.

8.1.1 Administrator Module

The Administrator can:

1. Appoint a Hostel Manager
2. Remove a Hostel Manager
3. Appoint a Mess Manager
4. Remove a Mess Manager
5. View the details of the students
6. Expel students permanently from the hostel

8.1.2 Hostel Manager Module

The Hostel Manager can:

1. Vacate students enrolled in their hostel
2. View details of students enrolled in their hostel

8.1.3 Mess Manager Module

The Mess Manager can:

1. View relevant details of students enrolled in their mess

8.1.4 Student Module

The students can:

1. Submit the application form for hostel allocation
2. Choose their mess

8.2 Non-Functional System Requirements

8.2.1 Performance Requirements

Some performance requirements needed are listed below:

1. The database should be capable of storing more than 15000 records.
2. The software should support multiple users at a time.

8.2.2 Safety Requirements

There are chances of crashes in a database at any time due to malware attacks or system failure. So, it is necessary to back up the database.

8.2.3 Security Requirements

Some factors identified to protect the software from accidental or malicious access, use, modification, destruction, or disclosure are described below.

1. Later version of the software will incorporate encryption
2. Keep specific log or history data sets.
3. Check data integrity for critical variables.
4. Restrict communication between some areas of the program

8.2.4 Software Quality Attributes

1. Less human error
2. Manual effort can be reduced
3. High Security
4. Data consistency
5. Easy to handle a large volume of data
6. Simpler updation and deletion of data
7. More organized form of maintaining records
8. Data redundancy can be avoided to some extent

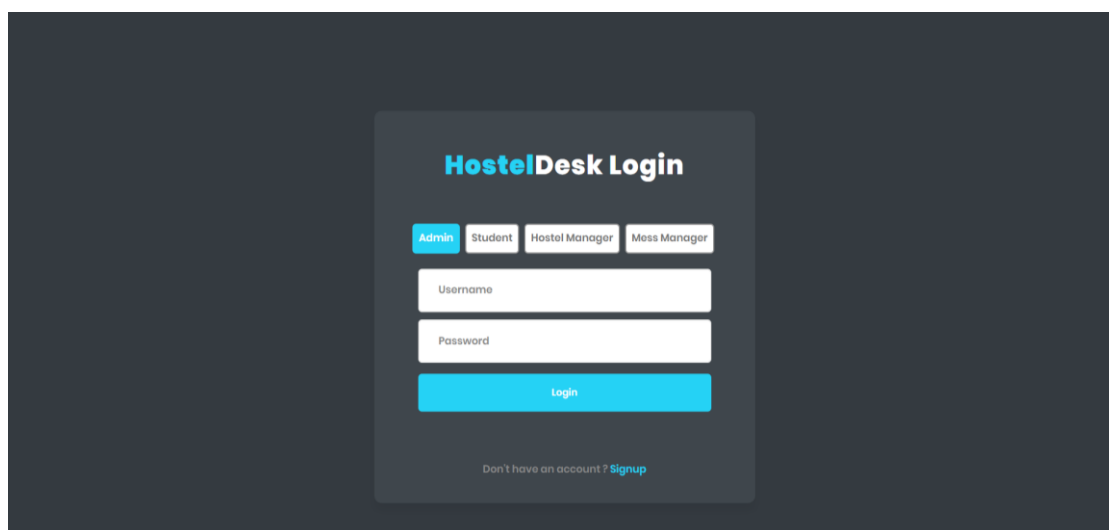
9 Hardware Requirements

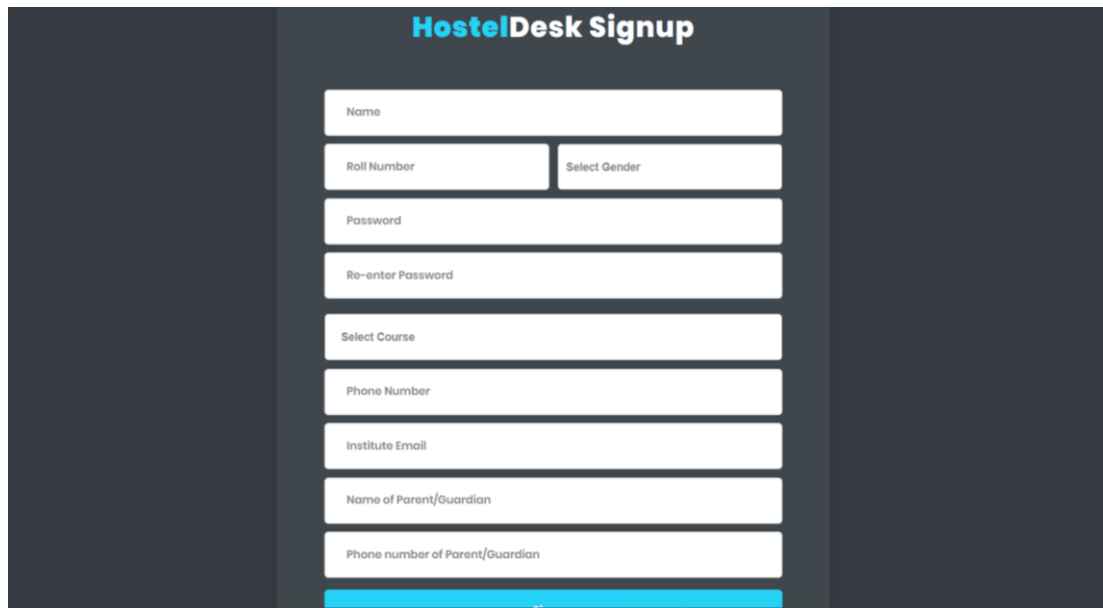
1. CPU: Intel Core or Xeon 3GHz (or Dual Core 2GHz) or equal AMD CPU
2. Cores: Single (Dual/Quad Core is recommended)
3. RAM: 4 GB (6 GB recommended)
4. Graphic Accelerators: Nvidia or ATI with support of OpenGL 1.5 or higher
5. Display Resolution: 1280x1024 is recommended, 1024x768 is minimum
6. Hard Disk: 64GB or more

10 Graphical User Interface

1. Login:

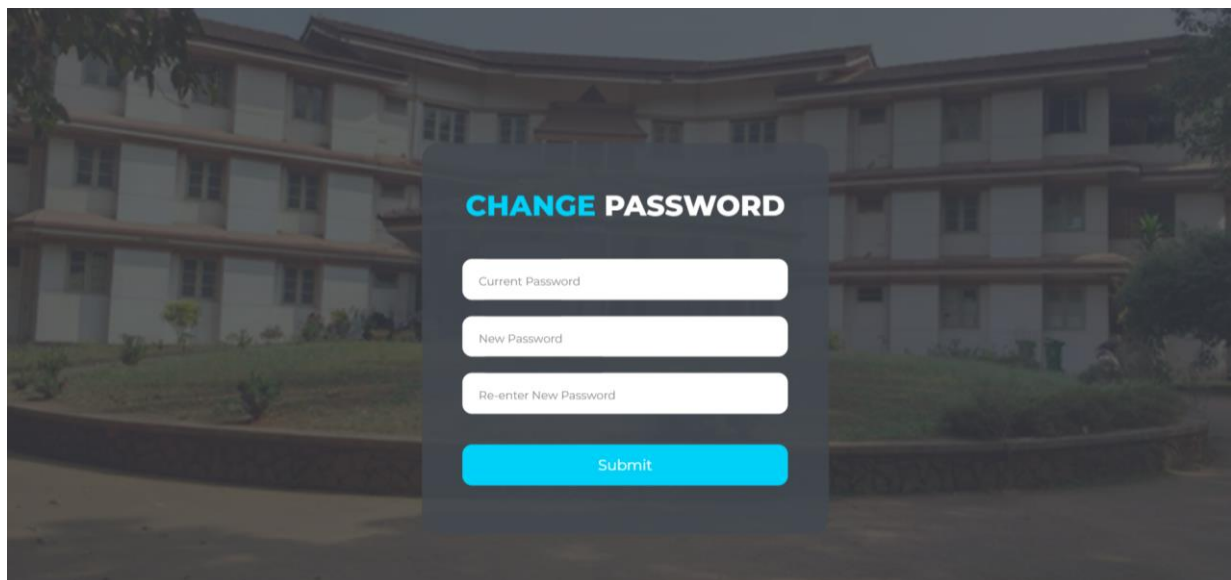
On first loading the website, the **homepage** is visible. Suppose you are already a registered student/ appointed Mess-Manager, Hostel Manager or Admin. In that case, you may log in via the “**Login**” option on the top left, enter the relevant details and be redirected to your respective module. If you haven’t registered yet (This option is for students only), you may click on the “**Signup**” option and enter your details and Course Details to do so.



A screenshot of the HostelDesk Signup form. The form is titled "HostelDesk Signup" in a blue and white font. It contains several input fields: Name, Roll Number, Select Gender (a dropdown menu), Password, Re-enter Password, Select Course (a dropdown menu), Phone Number, Institute Email, Name of Parent/Guardian, and Phone number of Parent/Guardian. A blue "Submit" button is at the bottom.

2. Change The Password

Existing users have the additional provision to **Change/Reset their password** by clicking on the change password Option visible on the signup page.

A screenshot of the Change Password form. The form is titled "CHANGE PASSWORD" in a blue and white font. It contains three input fields: Current Password, New Password, and Re-enter New Password. A blue "Submit" button is at the bottom. The background of the form is a blurred image of a hostel building.

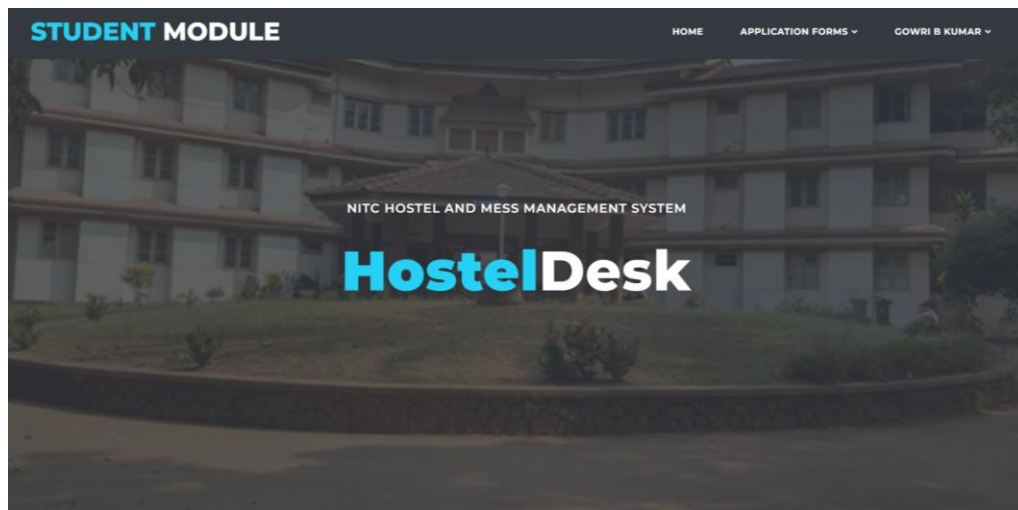
3. User Modules And Functionalities

Based on the ID entered by the user, they will be redirected to the various modules - **Student, Mess Manager, Hostel Manager,**

and Administrator. The functionalities of each of these modules are explained in the following subsections.

3.1 Student Module

The student module offers two main provisions to students registered under HostelDesk - **Application Forms** and **View/Edit Profile** (Represented on the Nav Bar as <STUDENTNAME>)

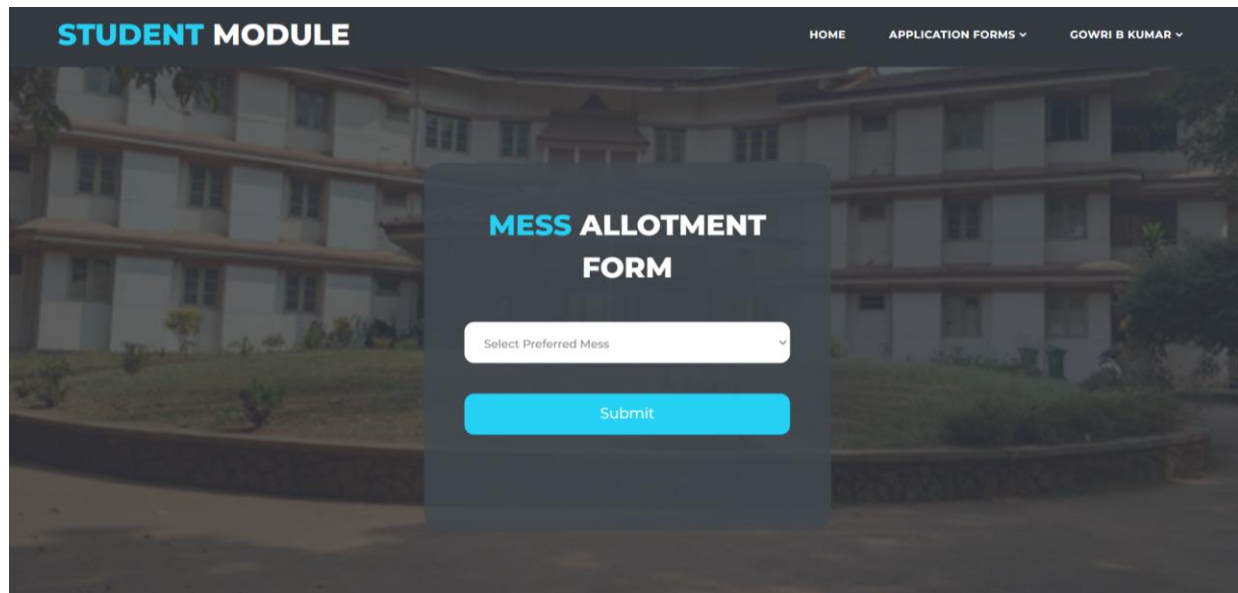
The screenshot displays the 'View/Edit Profile' form for a student named Gowri B Kumar. The form is titled 'GOWRI B KUMAR' and includes the following fields: Roll Number (B200699CS), Gender (F), Email (gowri_b200699cs@nitc.ac.in), Hostel (MLH), Floor (8), Room (001), and Mess (MLH Mess). Below these fields are input boxes for Name (Gowri B Kumar), Course (UG 3rd Year), Name of parent/guardian (B Kumar), Phone number (8520456277), and Alternate phone number of parent/guardian (7458962541). An 'Update' button is located at the bottom of the form.

Students may update their personal information in case of any errors. They cannot edit Hostel Information, Roll Number and Mess information under this provision.

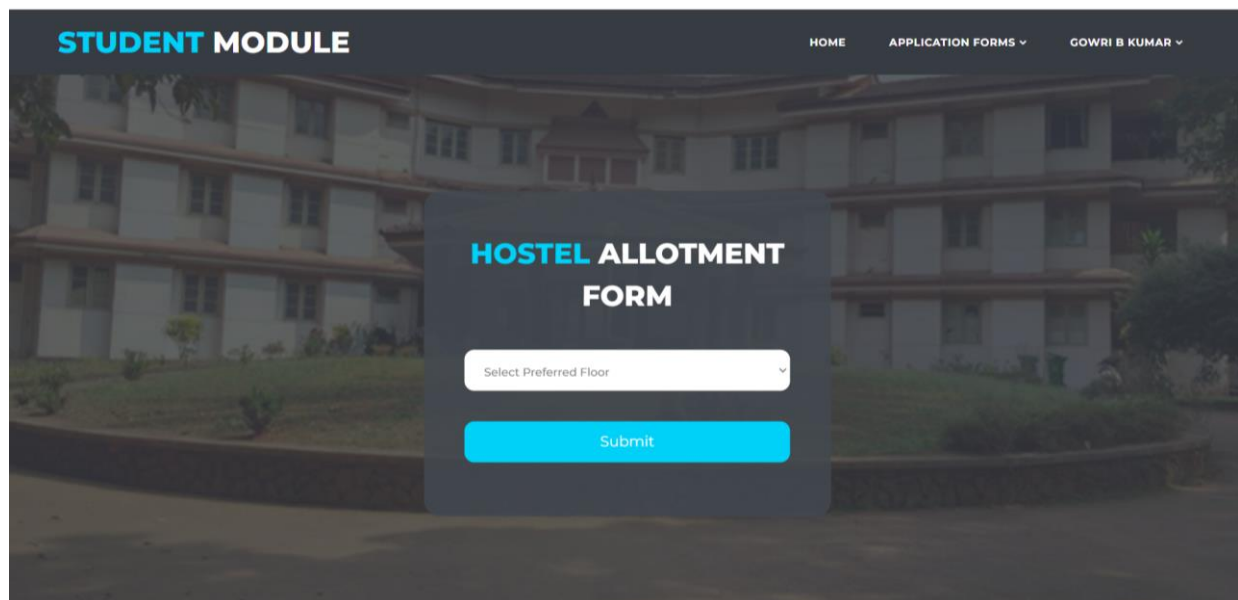
Under the **Application Forms** button in the NavBar, Students may fill out applications for Mess Allocation and Hostel Allocation. The **Mess Form** requires students to select their mess for that

particular month. Once a specific mess has reached its maximum capacity, it will be unavailable in the Drop-Down Menu.

Students must also fill out the Hostel Application form to be allotted a hostel room. The form is shown based on the student's year and the hostel to which the student is permitted to be assigned. It collects the student's floor preferences and allots the room based on availability.



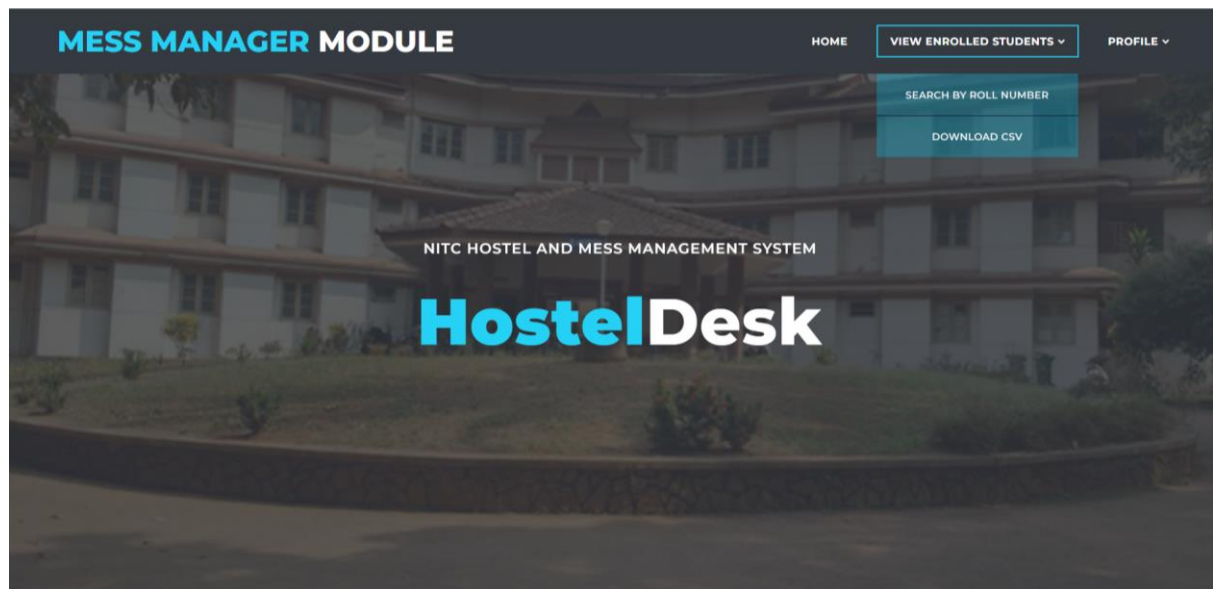
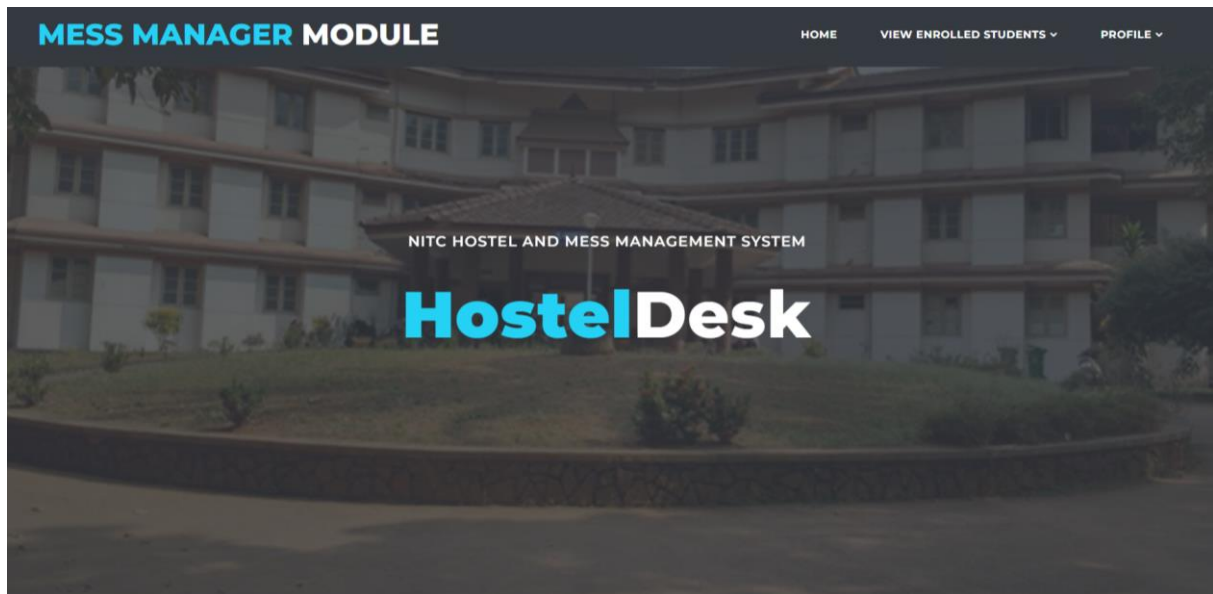
The screenshot shows the 'MESS ALLOTMENT FORM' within the 'STUDENT MODULE'. The form is centered on a dark background with a blurred image of a hostel building. It features a dropdown menu labeled 'Select Preferred Mess' and a blue 'Submit' button. The navigation bar at the top includes 'HOME', 'APPLICATION FORMS', and 'GOWRI B KUMAR'.

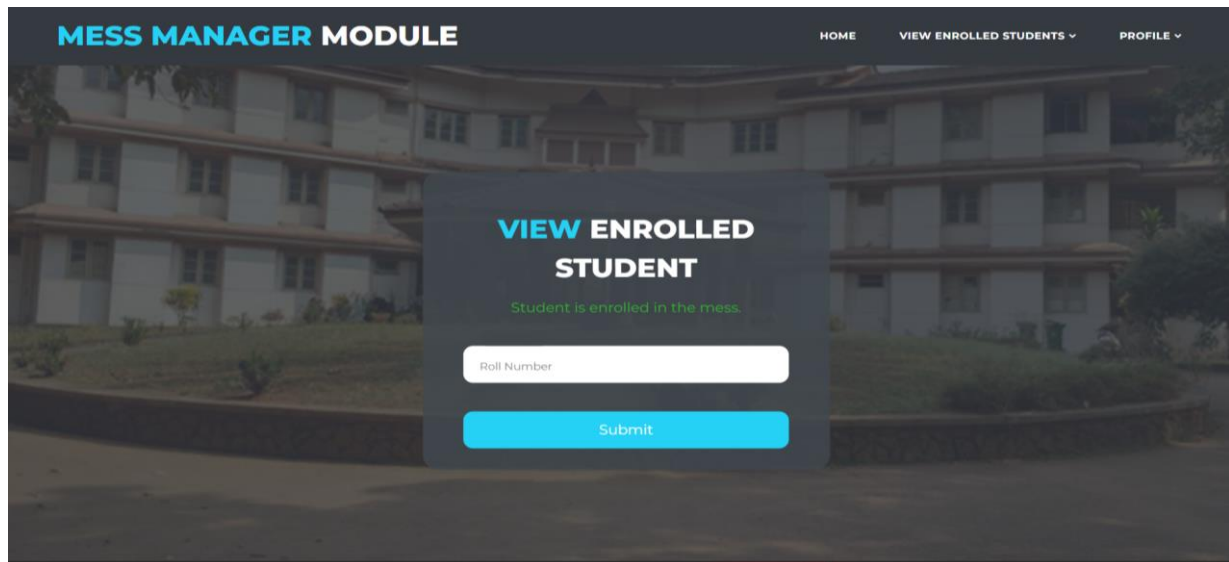


The screenshot shows the 'HOSTEL ALLOTMENT FORM' within the 'STUDENT MODULE'. The form is centered on a dark background with a blurred image of a hostel building. It features a dropdown menu labeled 'Select Preferred Floor' and a blue 'Submit' button. The navigation bar at the top includes 'HOME', 'APPLICATION FORMS', and 'GOWRI B KUMAR'.

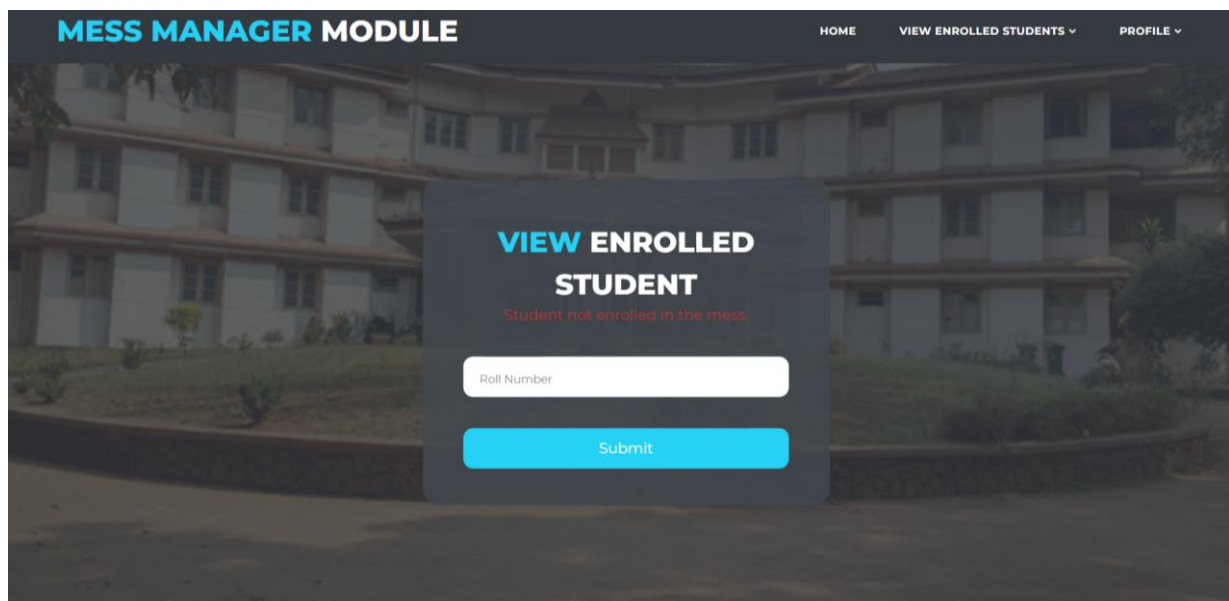
3.2 Mess Manager Module

This module offers Mess-Managers the provision to view the students enrolled in the mess they are currently undertaking. Under this provision, they are provided with the option to check whether a particular student is enrolled in their mess and to download the entire database of students enrolled in their mess using the “**Download CSV**” option.



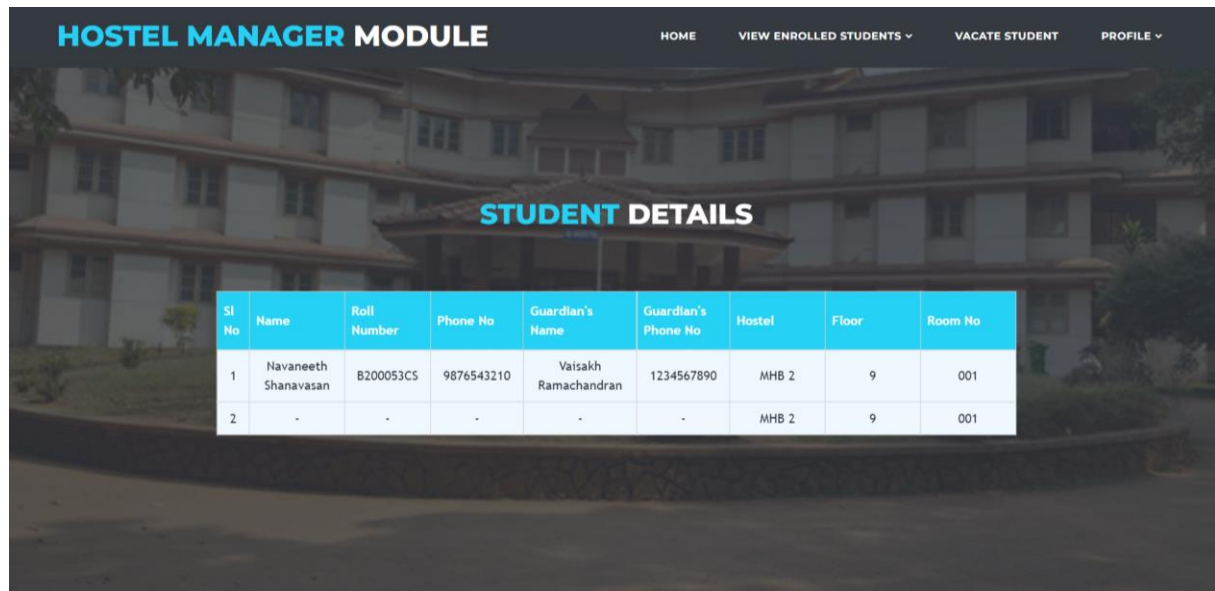
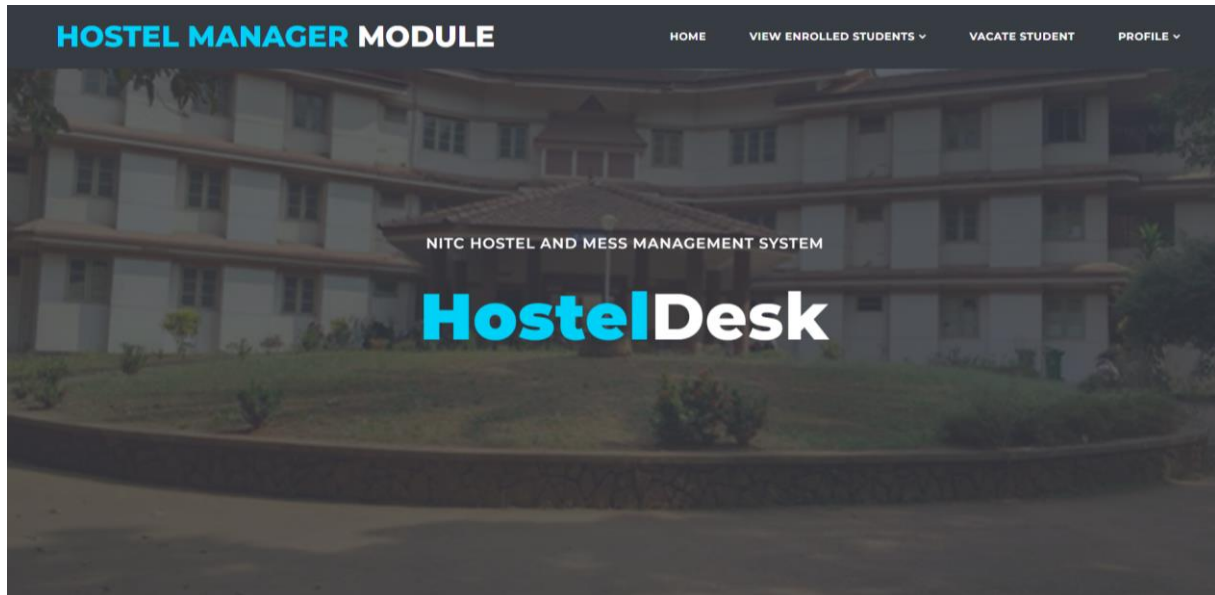


As seen in the screenshots above, on entering the roll number of a student, the system checks whether the student is enrolled in that particular mess and shows the result.

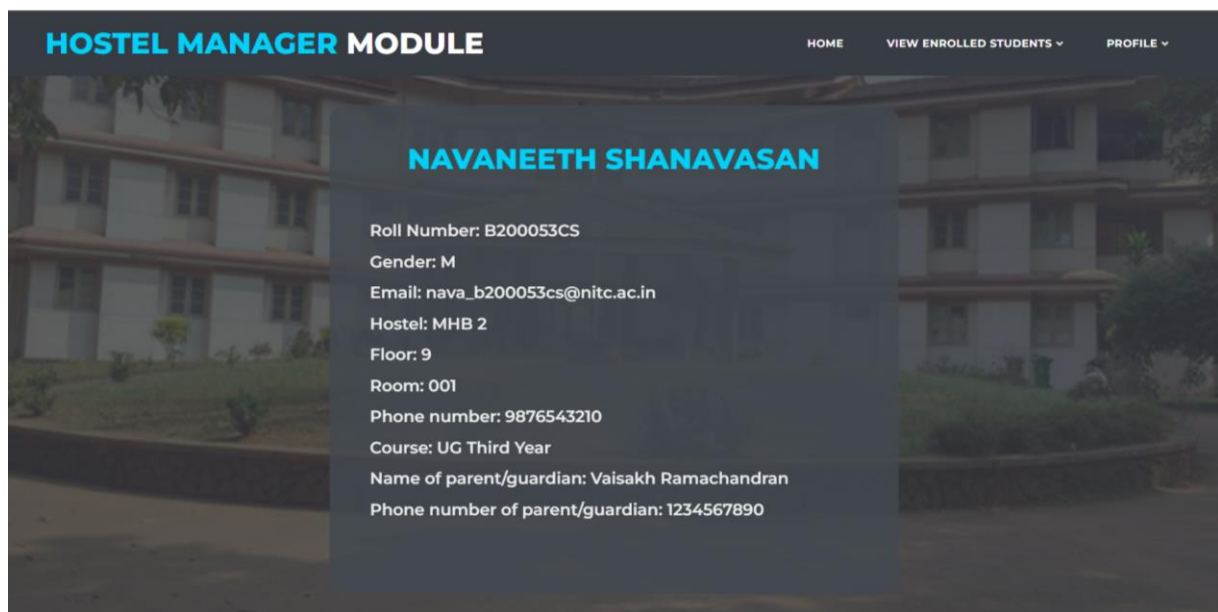
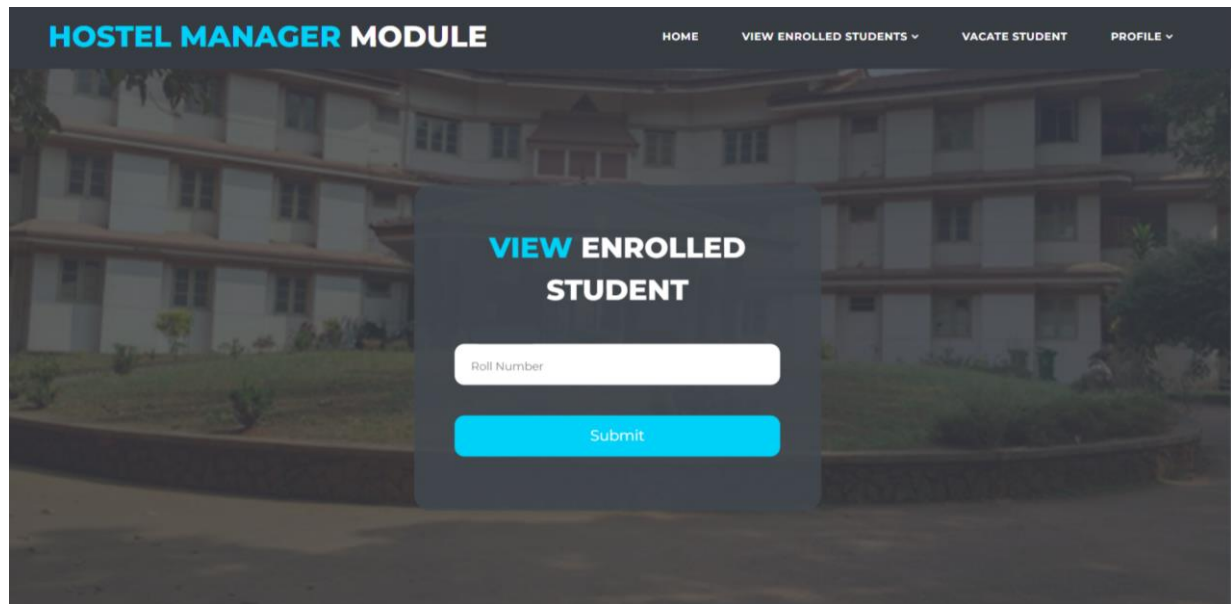


3.3 Hostel Manager

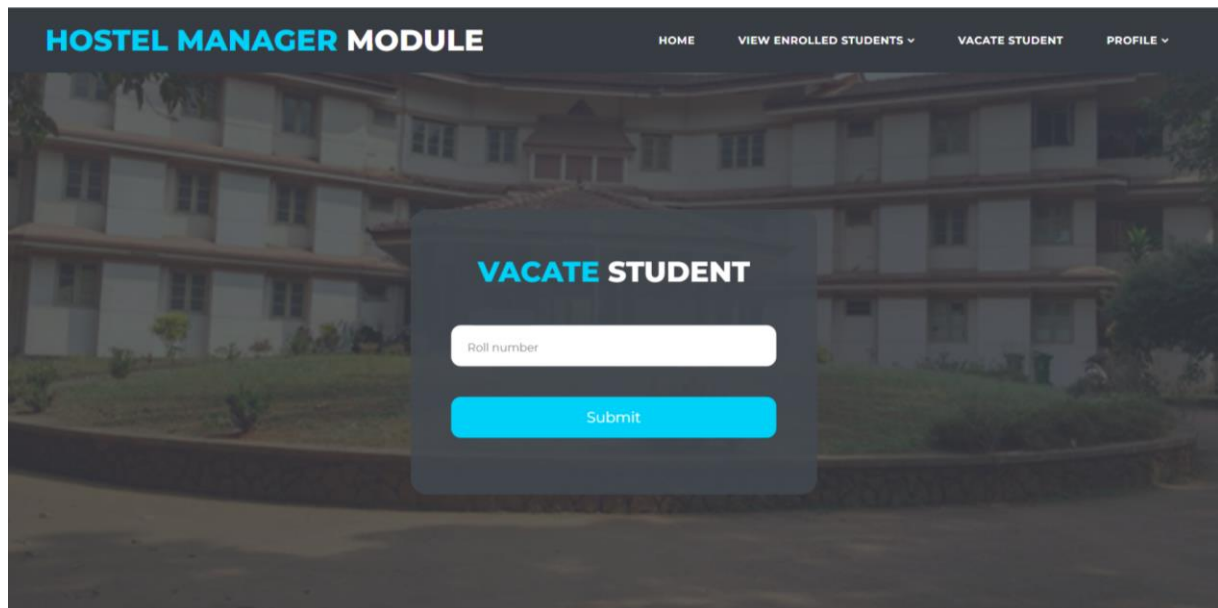
This module offers Hostel-Managers the provision to view students enrolled in the respective hostel they are in charge of, vacate a student who is currently a resident of that hostel and view their profile.



The Hostel Managers may view the details of a particular student by searching for them using the student's roll number. The respective student's details will then be visible on the screen.

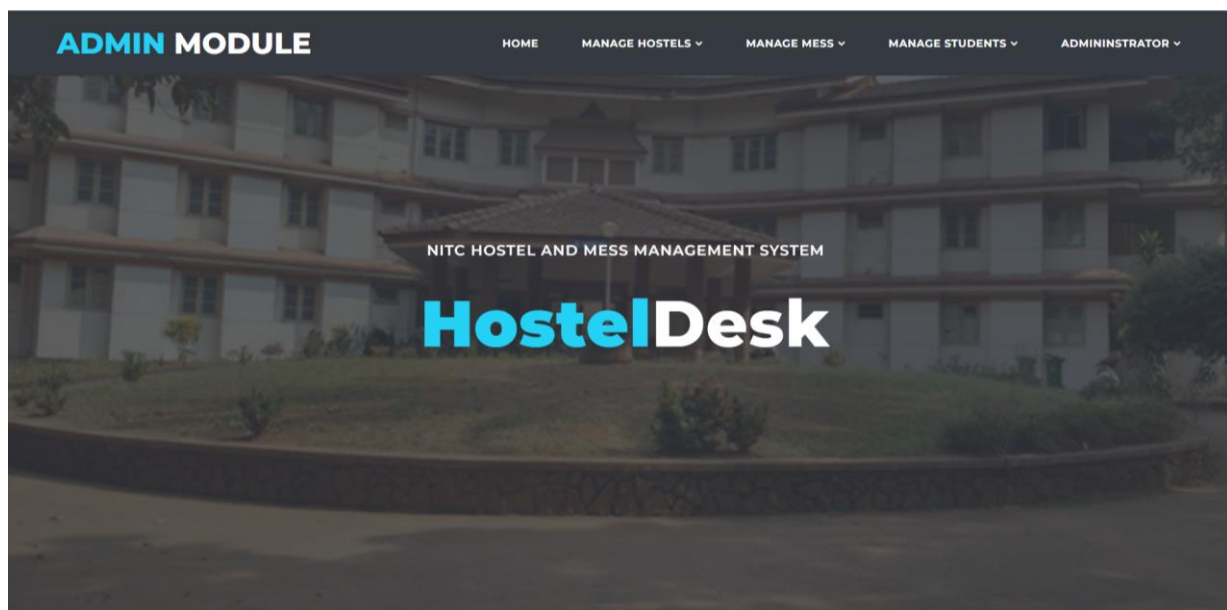


The Hostel Manager can also vacate a student with the help of the **vacate student** option by entering the student's roll number.



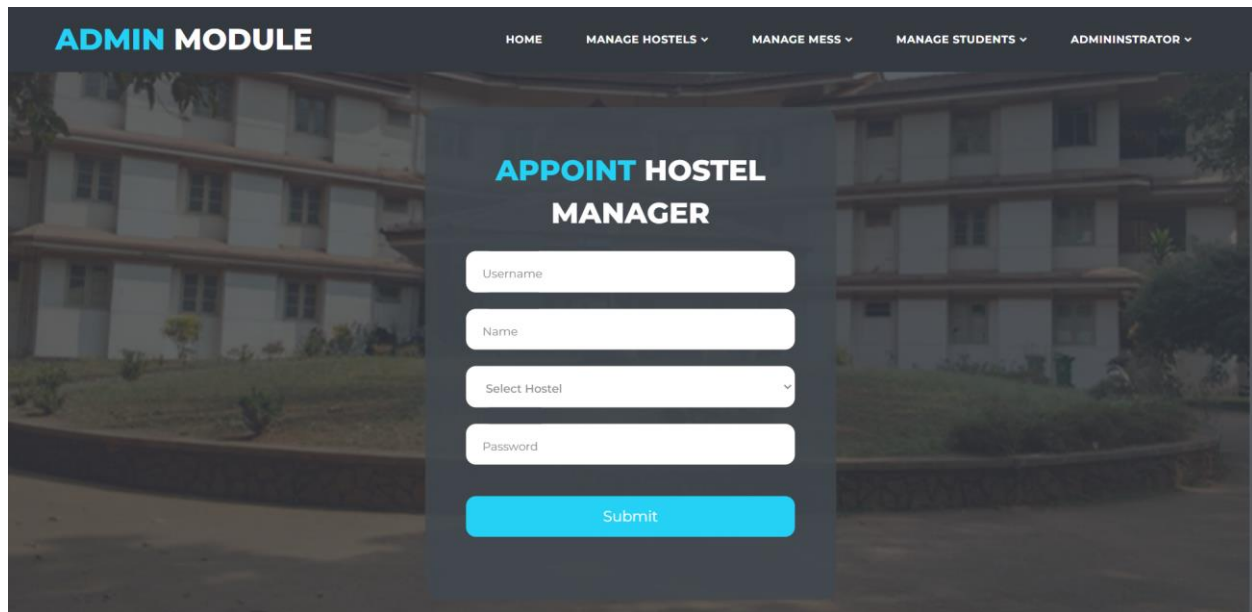
3.4 Administrator

This module offers administrators the provision to appoint and remove hostel-managers and mess-managers, expel students from their respective hostels, and view details of the students.



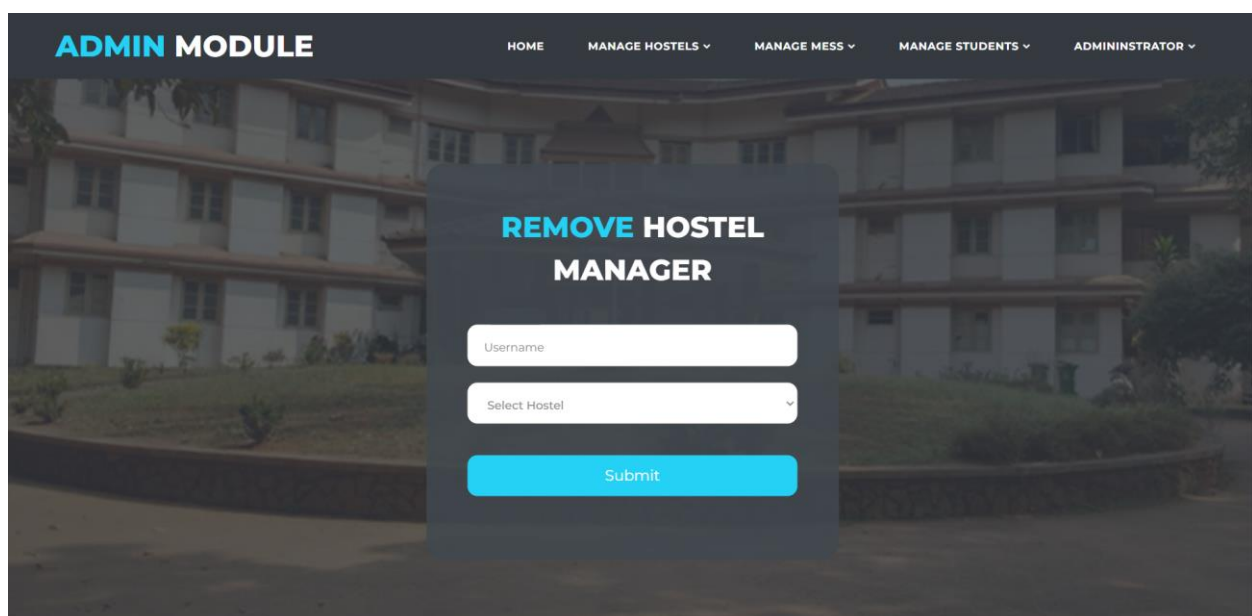
This module offers administrators the provision to appoint and remove hostel managers and mess managers, expel students from their respective hostels, and view details of the students.

The administrator may appoint hostel managers by using the manage hostels section. They can appoint a hostel manager by setting their username, name, hostel and password.



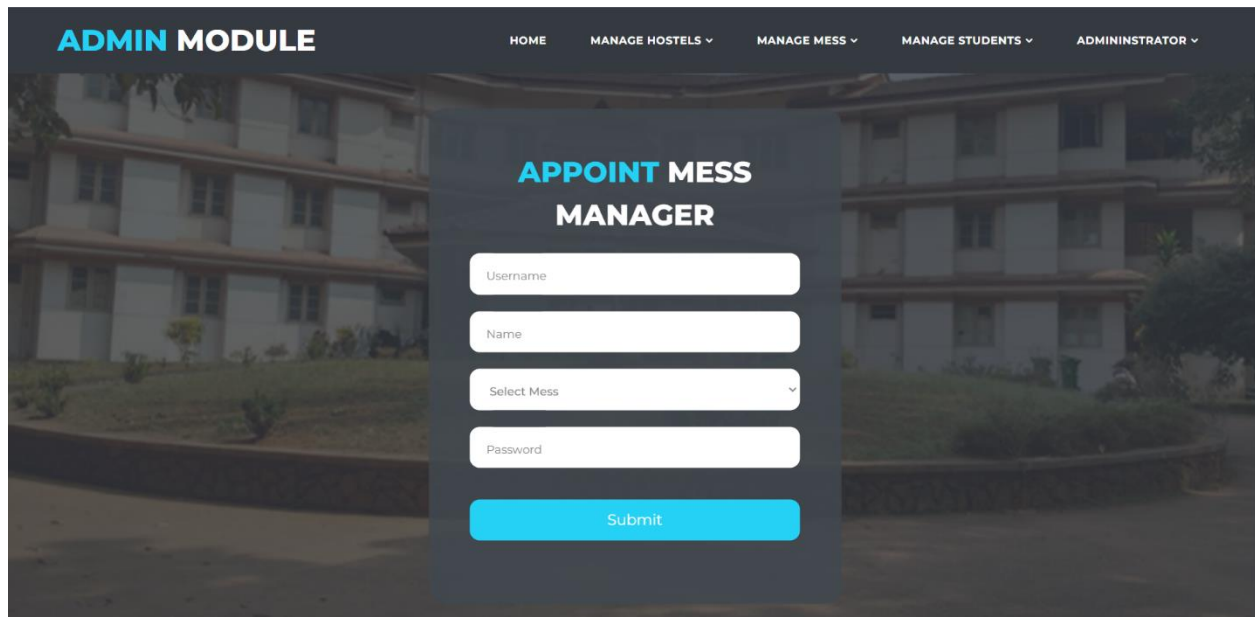
The screenshot shows the 'ADMIN MODULE' interface with a navigation bar containing 'HOME', 'MANAGE HOSTELS', 'MANAGE MESS', 'MANAGE STUDENTS', and 'ADMINISTRATOR'. The main content area features a form titled 'APPOINT HOSTEL MANAGER' overlaid on a background image of a hostel building. The form includes four input fields: 'Username', 'Name', 'Select Hostel' (a dropdown menu), and 'Password'. A blue 'Submit' button is located at the bottom of the form.

The administrator can remove hostel managers by entering their corresponding usernames and hostel number in the remove hostel-manager section.

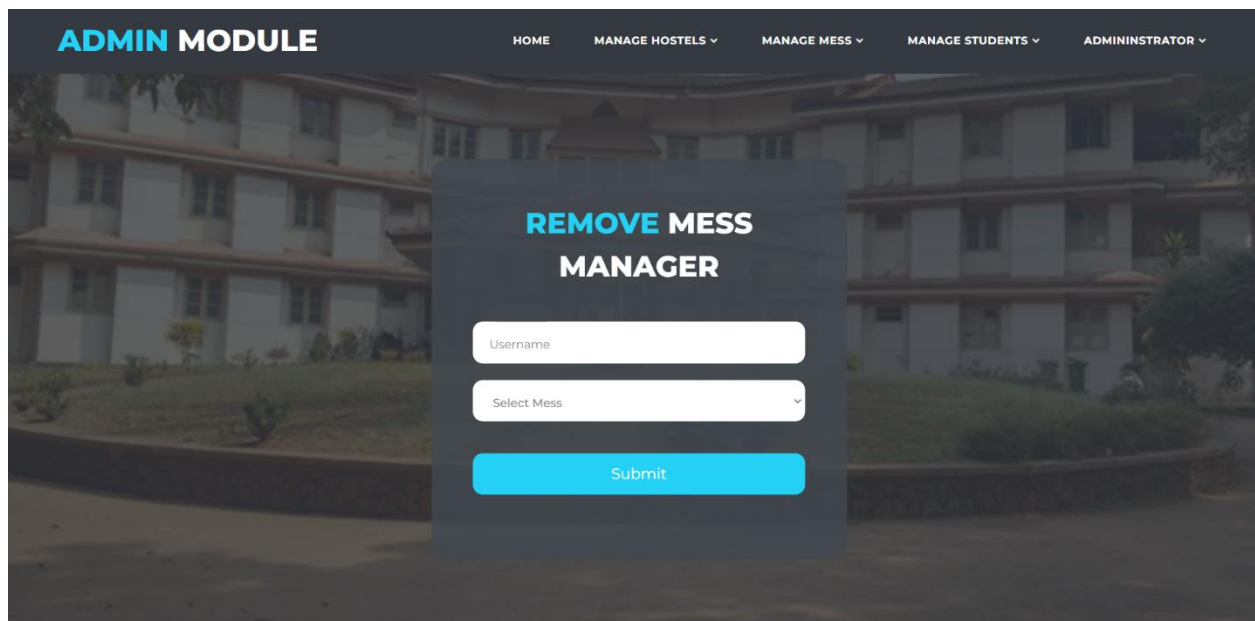


The screenshot shows the 'ADMIN MODULE' interface with a navigation bar containing 'HOME', 'MANAGE HOSTELS', 'MANAGE MESS', 'MANAGE STUDENTS', and 'ADMINISTRATOR'. The main content area features a form titled 'REMOVE HOSTEL MANAGER' overlaid on a background image of a hostel building. The form includes two input fields: 'Username' and 'Select Hostel' (a dropdown menu). A blue 'Submit' button is located at the bottom of the form.

Mess-managers can also be appointed and removed in the same way using the manage mess section.

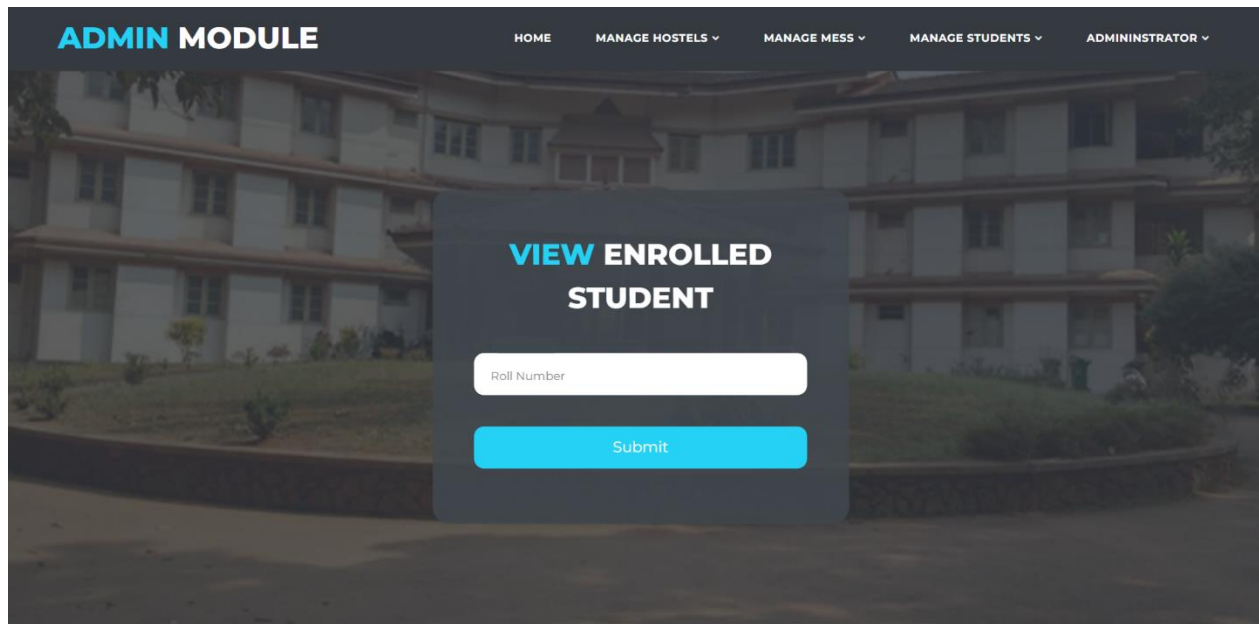


The screenshot shows the 'ADMIN MODULE' interface with a navigation bar containing 'HOME', 'MANAGE HOSTELS', 'MANAGE MESS', 'MANAGE STUDENTS', and 'ADMINISTRATOR'. The main content area features a form titled 'APPOINT MESS MANAGER'. The form includes four input fields: 'Username', 'Name', 'Select Mess' (a dropdown menu), and 'Password'. A blue 'Submit' button is located at the bottom of the form. The background of the page shows a blurred image of a multi-story building.



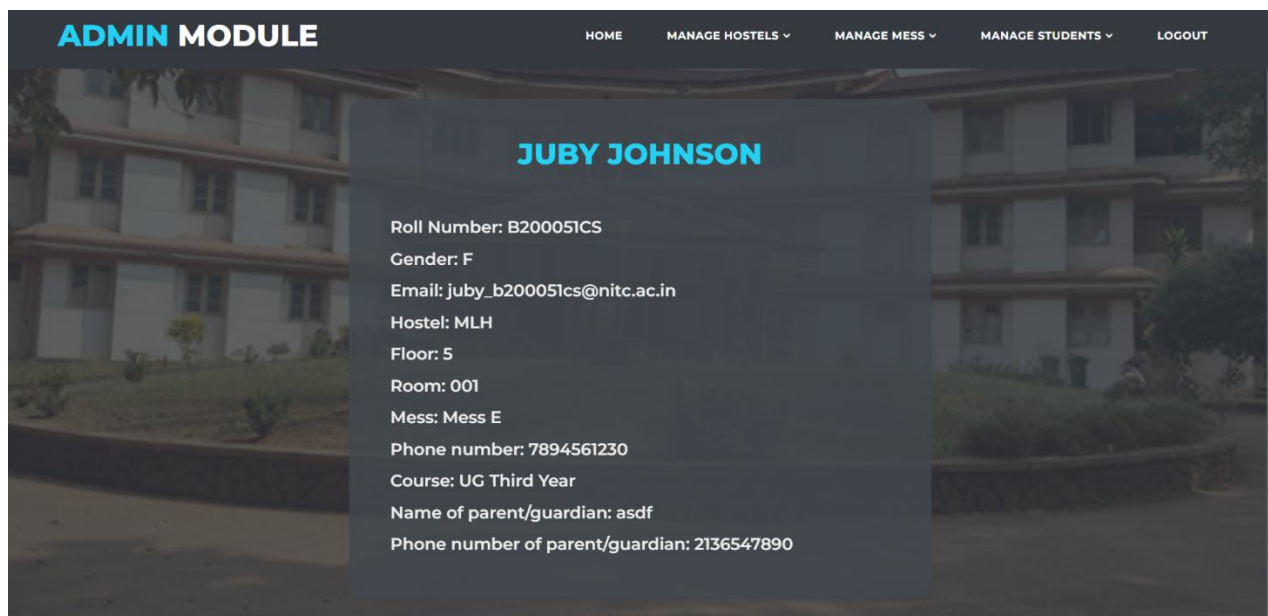
The screenshot shows the 'ADMIN MODULE' interface with a navigation bar containing 'HOME', 'MANAGE HOSTELS', 'MANAGE MESS', 'MANAGE STUDENTS', and 'ADMINISTRATOR'. The main content area features a form titled 'REMOVE MESS MANAGER'. The form includes two input fields: 'Username' and 'Select Mess' (a dropdown menu). A blue 'Submit' button is located at the bottom of the form. The background of the page shows a blurred image of a multi-story building.

The administrator may view the details of a student by entering their roll number in the view student records section.



The screenshot shows the 'ADMIN MODULE' interface with a navigation bar containing 'HOME', 'MANAGE HOSTELS', 'MANAGE MESS', 'MANAGE STUDENTS', and 'ADMINISTRATOR'. The main content area features a form titled 'VIEW ENROLLED STUDENT' with a text input field for 'Roll Number' and a 'Submit' button. The background is a blurred image of a hostel building.

View of the student record.

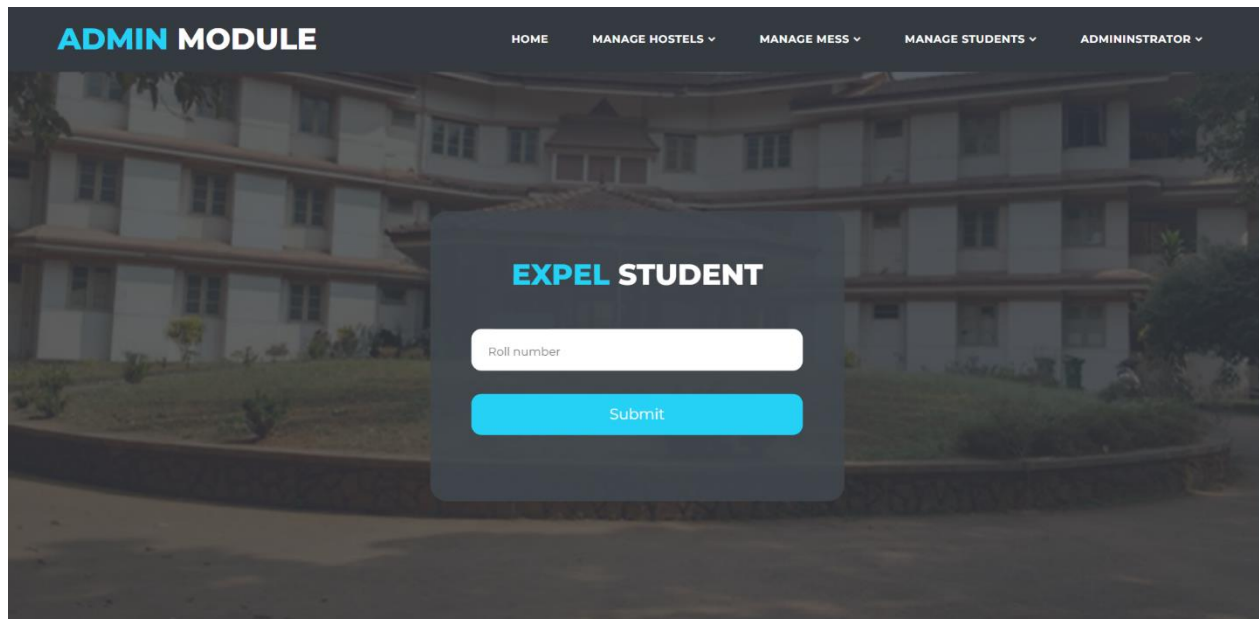


The screenshot shows the 'ADMIN MODULE' interface with a navigation bar containing 'HOME', 'MANAGE HOSTELS', 'MANAGE MESS', 'MANAGE STUDENTS', and 'LOGOUT'. The main content area displays the student record for 'JUBY JOHNSON' with the following details:

- Roll Number: B200051CS
- Gender: F
- Email: juby_b200051cs@nitc.ac.in
- Hostel: MLH
- Floor: 5
- Room: 001
- Mess: Mess E
- Phone number: 7894561230
- Course: UG Third Year
- Name of parent/guardian: asdf
- Phone number of parent/guardian: 2136547890

The background is a blurred image of a hostel building.

The administrator can permanently expel a student from the hostel using the manage student section.



The screenshot displays the 'ADMIN MODULE' interface. The top navigation bar includes links for HOME, MANAGE HOSTELS, MANAGE MESS, MANAGE STUDENTS, and ADMINISTRATOR. The main content area features a dark overlay with the title 'EXPEL STUDENT' in bold. Below the title is a text input field labeled 'Roll number' and a blue 'Submit' button. The background of the interface shows a photograph of a multi-story hostel building.

11 Backend Code

```
from django.http import HttpResponseRedirect
from django.shortcuts import render
from django.shortcuts import redirect
# from .forms import LoginForm, SignupForm
from django.db import connection
from django.db import IntegrityError
from .models import *
from datetime import datetime, timedelta
from django.db import connection
import random
import csv
import os

def redirect_modules(role):
    if role == 1:
        # return render(request, "admin_module.html")
        return redirect("../admin_module")
    elif role == 2:
        # return render(request, "student_module.html")
        return redirect("../student_module")
    elif role == 3:
        # return render(request, "hostel_manager_module.html")
        return redirect("../hostel_manager_module")
    elif role == 4:
        # return render(request, "mess_manager_module.html")
        return redirect("../mess_manager_module")

def rollno_validation(rollno):
    rollno = rollno.lower()
    masters = ['ar', 'ch', 'ce', 'cs', 'ca', 'ee', 'ec', 'me', 'mt', 'ma', 'cy', 'ph', 'ms']
    bachelors = ['cs', 'ee', 'ec', 'me', 'ep', 'ch', 'ar', 'bt', 'ce', 'mt', 'pe']
    if len(rollno)==9 and rollno[0].isalpha() and rollno[1:7].isdigit() and
rollno[7:10].isalpha() and rollno[0] in ['b', 'm'] :
        if (rollno[0]=='b' and rollno[7:] in bachelors):
            return 1
        elif (rollno[0]=='m' and rollno[7:] in masters):
            return 1
    else:
        return -1
    return -1

def email_validation(email, rollno):
    rollno = rollno.lower()
    mail_format = '_' + rollno + '@nitc.ac.in'
    if len(email)<= 22:
        return -1
```

```

        elif email[-21:] != mail_format:
            return -1
        else:
            return 1

def ph_no_validation(ph_no):
    ph_no = str(ph_no)
    if (ph_no.isnumeric() and len(ph_no)==10):
        return 1
    else:
        return -1

# Create your views here.
def index(request):
    if request.session.has_key('user'):
        #Deleting session variable
        del request.session['user']
    return render(request,"index.html")

'''*****
*****'''
#LOGIN
def login(request):
    dict2 = {'admin':1,'student':2,'h_manager':3,'m_manager':4}
    dict1={'name':'','password':'','role':''}
    if request.method=="POST":
        dict1=request.POST
        cursor = connection.cursor()
        query = "select password,r_id from login_cred where u_id = '{}'"
        query = query.format(dict1['name'])
        cursor.execute(query)
        y = cursor.fetchall()
        expel_password = "ulsjy@lt$bj#nk$adf^nkiB307SGBD"
        if (len(y)!=0):
            if y[0][0] == expel_password:
                return render(request, "login.html",{'error':"User is
expelled."})
            if (y[0][0]==dict1['password'] and y[0][1]==dict2[dict1['role']]):
                role = dict2[dict1['role']]
                request.session['user'] = [dict1['name'],role]
                return redirect_modules(role)
            else:
                # print("Wrong credentials or user doesn't exist")
                return render(request, "login.html",{'error':"Wrong credentials
or user doesn't exist"})
        else:
            return render(request, "login.html",{'error':"Wrong credentials or
user doesn't exist"})

```



```

        entered_details['phno'],
        entered_details['email'],
        entered_details['pname'],
        entered_details['p_phno'])

    reg_login_query =
reg_login_query.format(entered_details['rollno'],entered_details['pass1'])
    try:
        cursor.execute(reg_login_query)
        cursor.execute(insert_query)
    except IntegrityError:
        returning_values['error'] = 'Invalid institute email.'
        return render(request, "signup.html",returning_values)
    return render(request,"signup_1.html")
return render(request,"signup.html")

'''*****
*****'''

def admin_module(request):
    if request.session.has_key('user'):
        user_details = request.session['user']
        if user_details[1] == 1:
            return render(request,"admin_module.html")
        else:
            return redirect("../login")
    else:
        return redirect("../login")

'''*****
*****'''

def student_module(request):
    if request.session.has_key('user'):
        user_details = request.session['user']
        if user_details[1] == 2:
            display_name = {'name':'Profile'}
            cursor = connection.cursor()
            query = "select name from student_details where roll_no = '{}';"
            query = query.format(user_details[0])
            cursor.execute(query)
            y = cursor.fetchall()
            if (len(y)!=0):
                display_name['name'] = y[0][0]
                return render(request,"student_module.html",display_name)
            else:
                return redirect("../login")
        else:
            return redirect("../login")

```

```
def application_h(request):
    if request.session.has_key('user'):
        user_details = request.session['user']
        if user_details[1] == 2:
            display_name = {'name': 'Profile'}
            cursor = connection.cursor()
            query = "select name from student_details where roll_no = '{}';"
            query = query.format(user_details[0])
            cursor.execute(query)
            y = cursor.fetchall()
            if (len(y)!=0):
                display_name['name'] = y[0][0]
                if request.method=="POST":
                    entered_floor = request.POST
                    if len(entered_floor.keys())==1:
                        display_name['error'] = 'Kindly choose a floor.'
                        return render(request,
"application_h.html",display_name)
                    pfloor = entered_floor['hostel']
                    query = "select * from student_allocation_h where roll_no =
'{}';"
                    query = query.format(user_details[0])
                    cursor.execute(query)
                    y = cursor.fetchall()
                    if(len(y)==0):
                        return fuc(request, pfloor)
                    else:
                        display_name['error'] = 'You have already been allotted a
room.'
                        return render(request,
"application_h.html",display_name)
                    return render(request,"application_h.html",display_name)
                else:
                    return redirect("../../login")
            else:
                return redirect("../../login")

def application_m(request):
    if request.session.has_key('user'):
        user_details = request.session['user']
        if user_details[1] == 2:
            cursor = connection.cursor()
            query = "select name from student_details where roll_no = '{}';"
            query = query.format(user_details[0])
            cursor.execute(query)
```

```

display_name = cursor.fetchall()[0][0]
if request.method == "POST":
    if 'mess' in request.POST:
        pmess = request.POST['mess']
        roll_no = request.session['user'][0]

        query = "select gender from student_details where roll_no =
'{}';"

        query = query.format(roll_no)
        cursor.execute(query)
        gender = cursor.fetchall()[0][0]
        if gender == 'M':
            gender = 1
        else:
            gender = 2

        query = "select * from student_allocation_m where roll_no =
'{}';"

        query = query.format(roll_no)
        cursor.execute(query)
        present = len(cursor.fetchall())

        today = datetime.now().strftime("%d")
        if len(today) == 1:
            today = '0' + today

        if present and today < '25':
            return render(request,"application_m.html",{ 'error':"A
mess is already allocated. Please wait till 25th.", "name":display_name})

        query = "select capacity, allocated, accepted_gender, m_id
from mess_details where m_name = '{}';"
        query = query.format(pmess)
        cursor.execute(query)
        mess_details = cursor.fetchall()

        if len(mess_details) == 0:
            return
        render(request,"application_m.html",{ 'error':"Kindly select a valid
mess.", "name":display_name})

        if gender != mess_details[0][2] and int(mess_details[0][2])
!= 3:
            return render(request, "application_m.html",
{'error':"Kindly select a mess with your gender", "name":display_name})
            elif mess_details[0][0] > mess_details[0][1]:
                query = "update mess_details set allocated = '{}' where
m_name = '{}';"

```

```

        query = query.format(mess_details[0][1] + 1, pmess)
        cursor.execute(query)

        if present:
            query = "update student_allocation_m set m_id = '{}'"
where roll_no = '{}';"
            query = query.format(mess_details[0][3], roll_no)
            cursor.execute(query)
        else:
            query = "insert into student_allocation_m values
('{}','{}');"
            query = query.format(mess_details[0][3], roll_no)
            cursor.execute(query)

        return render(request, "application_m_1.html")
    else:
        return render(request, "application_m.html",
{'error':"There are no vacancies in the selected mess. Try
another.", "name":display_name})
    else:
        return render(request, "application_m.html", {'error':"Kindly
select a mess.", "name":display_name})
    else:
        return render(request, "application_m.html",
{"name":display_name})
    else:
        return redirect("../../login")
else:
    return redirect("../../login")

'''*****
*****'''

def hostel_manager_module(request):
    if request.session.has_key('user'):
        user_details = request.session['user']
        if user_details[1] == 3:
            return render(request, "hostel_manager_module.html")
        else:
            return redirect("../login")
    else:
        return redirect("../login")

def mess_manager_module(request):
    if request.session.has_key('user'):
        user_details = request.session['user']
        if user_details[1] == 4:
            return render(request, "mess_manager_module.html")

```

```

        else:
            return redirect("../login")
    else:
        return redirect("../login")

'''*****
*****
*'''

#APPOINT HOSTEL MANAGER
def ahm(request):
    if request.session.has_key('user'):
        user_details = request.session['user']
        if user_details[1] == 1:
            if request.method == "POST":
                new_hm = request.POST
                # for i in new_hm.values():
                #     if i=='':
                #         return render(request, "ahm.html",{'error':'Kindly
enter all values.}')
                if len(new_hm.keys())<5:
                    return render(request, "ahm.html",{'error':'Kindly enter all
values.}')
                insert_query = "insert into hostel_manager values
('{}','{}','{}')"
                reg_login_query = "insert into login_cred values ('{}','{}',3)"
                insert_query = insert_query.format(new_hm['username'],
new_hm['name'], new_hm['hostel'])
                reg_login_query = reg_login_query.format(new_hm['username'],
new_hm['password'])
                cursor = connection.cursor()
                try:
                    cursor.execute(reg_login_query)
                    cursor.execute(insert_query)
                except IntegrityError :
                    return render(request, "ahm.html",{'error':'User already
exists.}')
                return render(request,"ahm_1.html")
                return render(request,"ahm.html")
            else:
                return redirect("../../login")
        else:
            return redirect("../../login")

'''*****
*****
*'''

```

```
#REMOVE HOSTEL MANAGER
def rhm(request):
    if request.session.has_key('user'):
        user_details = request.session['user']
        if user_details[1] == 1:
            if request.method == "POST":
                remove_hm = request.POST
                # print(remove_hm)
                # for i in remove_hm.values():
                #     if i=='':
                #         return render(request, "rhm.html",{'error':'Kindly
enter all values.'})
                if len(remove_hm.keys())<3:
                    return render(request, "rhm.html",{'error':'Kindly enter all
values.'})
                # get_hostel_query = "select hostel_id from hostel_manager where
hm_id = '{}'"
                # get_hostel_query =
get_hostel_query.format(remove_hm['username'])
                select_query = "select * from hostel_manager where hm_id = '{}'"
                select_query = select_query.format(remove_hm['username'])
                remove_query = "delete from hostel_manager where hm_id = '{}'"
                remove_login_query = "delete from login_cred where u_id = '{}'"
                remove_query = remove_query.format(remove_hm['username'])
                remove_login_query =
remove_login_query.format(remove_hm['username'])
                cursor = connection.cursor()
                cursor.execute(select_query)
                y = cursor.fetchall()
                if len(y)==0:
                    return render(request, "rhm.html",{'error':"User doesn't
exist."})
                # cursor.execute(get_hostel_query)
                # y = cursor.fetchall()
                if y[0][2] != int(remove_hm['hostel']):
                    return render(request, "rhm.html",{'error':"Username and
hostel doesn't match."})
                cursor.execute(remove_query)
                cursor.execute(remove_login_query)
                return render(request, "rhm_1.html")
            return render(request, "rhm.html")
        else:
            return redirect("../../login")
    else:
        return redirect("../../login")
```

```

'''*****
*****
*'''

#APPOINT MESS MANAGER
def amm(request):
    if request.session.has_key('user'):
        user_details = request.session['user']
        if user_details[1] == 1:
            if request.method=="POST":
                new_mm = request.POST
                # print(new_mm)
                # for i in new_mm.values():
                #     if i=='':
                #         return render(request, "amm.html",{'error':'Kindly
enter all values.})
                if len(new_mm.keys())<5:
                    return render(request, "amm.html",{'error':'Kindly enter all
values.})
                insert_query = "insert into mess_manager values
('{}','{}','{}')"
                reg_login_query = "insert into login_cred values ('{}','{}',4)"
                insert_query = insert_query.format(new_mm['username'],
new_mm['name'], new_mm['mess'])
                reg_login_query = reg_login_query.format(new_mm['username'],
new_mm['password'])
                cursor = connection.cursor()
                try:
                    cursor.execute(reg_login_query)
                    cursor.execute(insert_query)
                except IntegrityError :
                    return render(request, "amm.html",{'error':'User already
exists or mess already has a manager.})
                    return render(request,"amm_1.html")
                    return render(request,"amm.html")
                else:
                    return redirect("../../login")
            else:
                return redirect("../../login")

'''*****
*****
*'''

#REMOVE MESS MANAGER
def rmm(request):
    if request.session.has_key('user'):
        user_details = request.session['user']

```

```

    if user_details[1] == 1:
        if request.method == "POST":
            remove_mm = request.POST
            if len(remove_mm.keys())<3:
                return render(request, "rmm.html",{'error':'Kindly enter all
values.'])

            select_query = "select * from mess_manager where mm_id = '{}'"
            select_query = select_query.format(remove_mm['username'])
            remove_query = "delete from mess_manager where mm_id = '{}'"
            remove_login_query = "delete from login_cred where u_id = '{}'"
            remove_query = remove_query.format(remove_mm['username'])
            remove_login_query =
remove_login_query.format(remove_mm['username'])
            # get_mess_query = "select mess_id from mess_manager where mm_id
= '{}'"

            # get_mess_query = get_mess_query.format(remove_mm['username'])
            cursor = connection.cursor()
            cursor.execute(select_query)
            y = cursor.fetchall()
            if len(y)==0:
                return render(request, "rmm.html",{'error':"User doesn't
exist."})

            # cursor.execute(get_mess_query)
            # y = cursor.fetchall()
            # print(remove_mm)
            # print(y[0][2])
            if y[0][2] != int(remove_mm['mess']):
                return render(request, "rmm.html",{'error':"Username and
mess doesn't match."})
            cursor.execute(remove_query)
            cursor.execute(remove_login_query)
            return render(request, "rmm_1.html")
            return render(request,"rmm.html")
        else:
            return redirect("../../login")
    else:
        return redirect("../../login")

```

```

'''*****
*****'''

```

#CHANGE PASSWORD

```

def change_password(request):
    if request.session.has_key('user'):
        user_details = request.session['user']
        if request.method=="POST":
            entered_details = request.POST
            for i in entered_details.values():

```



```

        if i=='':
            return render(request,
"change_password.html",{ 'error': 'Kindly enter all values.'})
        if (entered_details['n_pass'] != entered_details['re_n_pass']):
            return render(request,
"change_password.html",{ 'error': "Passwords doesn't match."})
        if (entered_details['c_pass'] == entered_details['re_n_pass'] ==
entered_details['n_pass']):
            return render(request, "change_password.html",{ 'error': "New
password cannot be same as old password."})
        get_password_query = "select password from login_cred where u_id =
'{}'"
        get_password_query = get_password_query.format(user_details[0])
        cursor = connection.cursor()
        cursor.execute(get_password_query)
        y = cursor.fetchall()
        if y[0][0] != entered_details['c_pass']:
            return render(request, "change_password.html",{ 'error': "Password
incorrect."})
        else:
            change_password_query = "update login_cred set password='{}'
where u_id='{}'"
            change_password_query =
change_password_query.format(entered_details['n_pass'],user_details[0])
            cursor.execute(change_password_query)
            return render(request, "change_password_1.html")
        return render(request, 'change_password.html')
    else:
        return redirect("../login")

'''*****
*****'''

def get_free_room(phostel,pfloor,rno): #integer ags
    cursor = connection.cursor()

    query = "SELECT * FROM Student_Allocation_H WHERE Roll_No='{}'"
    query = query.format(rno)
    cursor.execute(query)
    w = cursor.fetchall()

    if len(w) > 0:
        return (-1,-2)

    query = "select * from Room_Details where Floor_Id='{}' and H_Id='{}' and
(Status=0 or Status=1)"
    query = query.format(pfloor,phostel)
    cursor.execute(query)

```

```
y = cursor.fetchall()
if len(y)==0:
    query = "select * from Room_Details where H_Id='{}' and (Status=0 or
Status=1)"
    query = query.format(phostel)
    cursor.execute(query)
    y = cursor.fetchall()
if len(y)==0:
    return (-1,-1)
else:
    query = "UPDATE Room_Details SET Status='{}' WHERE H_Id='{}' and
Room_Id='{}' and Floor_Id='{}'"
    query = query.format(y[0][3]+1,y[0][0],y[0][2],y[0][1])
    cursor.execute(query)

    query = "select * from Hostel_Details where H_Id='{}'"
    query = query.format(pfloor,phostel)
    cursor.execute(query)
    z=cursor.fetchall()

    query = "UPDATE Hostel_Details SET Free_Room='{}' WHERE H_Id='{}'"
    query = query.format(z[0][2]-1,y[0][0])
    cursor.execute(query)

    query = "INSERT INTO Student_Allocation_H VALUES('{}','{}','{}','{}')"
    query = query.format(y[0][0],y[0][1],y[0][2],rno)
    cursor.execute(query)
    return (y[0][1], y[0][2])

def check(pfloor,jaba,gender):
    if (gender=="M"):
        if (jaba!=2 and jaba!=3 and pfloor>3):
            return -1
        else:
            return 0
    elif (gender=="F"):
        if (jaba==1 or jaba==4 and pfloor>3):
            return -1
        else:
            return 0

def fuc(request,pfloor):
    # stuedtn module
    rollno=request.session["user"][0]
    cursor = connection.cursor()
    query = "select * from Student_Details where Roll_No='{}'"
    query = query.format(rollno)
    y = cursor.execute(query)
```

```
y = cursor.fetchall()

gender=y[0][2]
jaba=y[0][3]
if (check(pfloor,jaba,gender)==-1):
    return render(request, "application_h.html',{'ferror':"Choose floor
between 1 to 3"})
else:
    if (gender=='M'):
        if jaba==1:
            query = "select * from Hostel_Details where H_Id=1"
            cursor.execute(query)
            z=cursor.fetchall()

            if z[0][2]==0:
                query = "select * from Hostel_Details where H_Id=2"
                cursor.execute(query)
                z=cursor.fetchall()
            if z[0][2]==0:
                print("error room not available in a and b")
            else:
                a,b=get_free_room(z[0][0],pfloor,rollno)

        elif jaba==2 or jaba==3:
            query = "select * from Hostel_Details where H_Id=10"
            cursor.execute(query)
            z=cursor.fetchall()
            if z[0][2]==0:
                print("error room not available in mbh")
            else:
                a,b=get_free_room(z[0][0],pfloor,rollno)
        elif jaba==4:
            query = "select * from Hostel_Details where H_Id=4"
            cursor.execute(query)
            z=cursor.fetchall()

            if z[0][2]==0:
                query = "select * from Hostel_Details where H_Id=5"
                cursor.execute(query)
                z=cursor.fetchall()
            if z[0][2]==0:
                query = "select * from Hostel_Details where H_Id=6"
                cursor.execute(query)
                z=cursor.fetchall()
            if z[0][2]==0:
                print("error room not available in d,e,f")
            else:
                a,b=get_free_room(z[0][0],pfloor,rollno)
```

```

elif jaba==5:
    query = "select * from Hostel_Details where H_Id=8"
    cursor.execute(query)
    z=cursor.fetchall()
    if z[0][2]==0:
        print("error room not available in pg1")
    else:
        a,b=get_free_room(z[0][0],pfloor,rollno)
elif jaba==6:
    query = "select * from Hostel_Details where H_Id=9"
    cursor.execute(query)
    z=cursor.fetchall()
    if z[0][2]==0:
        print("error room not available in pg2")
    else:
        a,b=get_free_room(z[0][0],pfloor,rollno)
elif (gender=="F"):
    if jaba==1 or jaba==4:
        query = "select * from Hostel_Details where H_Id=11"
        cursor.execute(query)
        z=cursor.fetchall()
        if z[0][2]==0:
            print("error room not available in lh")
        else:
            a,b=get_free_room(z[0][0],pfloor,rollno)
    else:
        query = "select * from Hostel_Details where H_Id=12"
        cursor.execute(query)
        z=cursor.fetchall()
        if z[0][2]==0:
            print("error room not available in mlh")
        else:
            a,b=get_free_room(z[0][0],pfloor,rollno)
return render(request,"application_h_1.html")

'''*****
*****
*'''

def check_if_in_mess(rno,mess):
    cursor = connection.cursor()

    query = "select * from student_allocation_m where roll_no='{ }' and m_id={}"
    query = query.format(rno,mess)
    cursor.execute(query)
    y = cursor.fetchall()
    if(len(y)==0):
        return -1

```

```

    else:
        return 1

def view_enroll_m(request):
    if request.session.has_key('user'):
        user_details = request.session['user']
        if user_details[1] == 4:
            if request.method=="POST":
                entered_detail=request.POST
                rollno = entered_detail['roll_no']
                if (rollno == ''):
                    return render(request, 'view_enroll_m.html',{'error':"Kindly
enter a roll number."})
                if (rollno_validation(rollno)==-1):
                    return render(request,
'view_enroll_m.html',{'error':"Invalid roll number."})
                mm_id = user_details[0]
                cursor = connection.cursor()
                query = "select * from mess_manager where mm_id = '{}'"
                query = query.format(mm_id)
                cursor.execute(query)
                y=cursor.fetchall()
                mess=y[0][2]
                a = check_if_in_mess(rollno,mess)
                if a==-1:
                    return render(request,
'view_enroll_m.html',{'error':"Student not enrolled in the mess."})
                elif a==1:
                    return render(request,
'view_enroll_m.html',{'message':"Student is enrolled in the mess."})
                return render(request,"view_enroll_m.html")
            else:
                return redirect("../../login")
        else:
            return redirect("../../login")

```

```

'''*****
*****
*'''

```

```

# DOWNLOAD MESS STUDENTS CSV
def mm_get_students(request):
    if request.session.has_key('user'):
        user_details = request.session['user']
        if user_details[1] == 4:
            manager_name = user_details[0]
            cursor = connection.cursor()

```

```

        query = "select m_id from mess_manager where mm_id = '{}'"
        query = query.format(manager_name)
        cursor.execute(query)
        m_id = cursor.fetchall()[0][0]

        query = "select a.roll_no, b.name, a.m_id, b.ph_no, b.email from
student_allocation_m as a, student_details as b where a.m_id = '{}'" and
a.roll_no = b.roll_no;"
        query = query.format(m_id)
        cursor.execute(query)
        res = cursor.fetchall()
        filename =
os.path.dirname(os.path.dirname(os.path.abspath(__file__))) + '/statics/Mess
manager module/assets/mess_students.csv'
        with open(filename, 'w', newline = '') as csvfile:
            x = csv.writer(csvfile)
            x.writerow(('Roll number', 'Name', 'Mess', 'Phone
number', 'EmailID'))
            x.writerows(res)
            csvfile.close()
        file = open(filename, 'rb')
        response = HttpResponse(file, content_type='application/csv')
        response['Content-Length'] = os.path.getsize(filename)
        response['Content-Disposition'] = 'attachment; filename=%s' % 'Mess
Students.csv'
        return response
    else:
        return redirect("../login")

'''*****
*****
*'''

#HM SEARCH STUDENT BY ROLLNO
def view_enroll_h_rollno(request):
    if request.session.has_key('user'):
        user_details = request.session['user']
        if user_details[1] == 3:
            if request.method == "POST":
                rno = request.POST['roll_no']
                if rno == '':
                    return
            render(request, "view_enroll_h_rollno.html", {'error': 'Kindly enter all values.'})
        else:
            details = {}
            cursor = connection.cursor()

```

```
query = "select * from student_details where roll_no =
'{}';"

query = query.format(rno)
cursor.execute(query)
res1 = cursor.fetchall()

if len(res1) == 0:
    return
render(request, "view_enroll_h_rollno.html", {'error': 'Student not registered.'})

res1 = res1[0]

query = "select h_id, floor_id, room_id from
student_allocation_h where roll_no = '{}';"
query = query.format(rno)
cursor.execute(query)
res2 = cursor.fetchall()

query = "select a.h_id, b.h_name from hostel_manager as a,
hostel_details as b where a.hm_id = '{} and a.h_id=b.h_id;"
query = query.format(request.session['user'][0])
cursor.execute(query)
hos = cursor.fetchall()[0]
hid = hos[0]
hname = hos[1]

if len(res2) == 0 or int(hid) != int(res2[0][0]):
    return
render(request, "view_enroll_h_rollno.html", {'error': 'Student is not allotted to
this hostel.'})

res2 = res2[0]

details['name'] = res1[0]
details['roll_no'] = res1[1]
details['gender'] = res1[2]
details['course'] = res1[3]
details['ph_no'] = res1[4]
details['email'] = res1[5]
details['p_name'] = res1[6]
details['p_ph_no'] = res1[7]

details['hostel'] = hname
details['floor'] = res2[1]

if(res2[2] < 10):
    details['room'] = '00' + str(res2[2])
elif res2[2] < 100:
```

```

        details['room'] = '0' + str(res2[2])
    else:
        details['room'] = str(res2[2])

    if details['course'] == 1:
        details['course'] = "UG First Year"
    elif details['course'] == 2:
        details['course'] = "UG Second Year"
    elif details['course'] == 3:
        details['course'] = "UG Third Year"
    elif details['course'] == 4:
        details['course'] = "UG Fourth Year"
    elif details['course'] == 5:
        details['course'] = "PG First Year"
    else:
        details['course'] = "PG Second Year"

    return
render(request,"display_enroll_h_rollno.html",details)
    else:
        return render(request,"view_enroll_h_rollno.html")
    else:
        return redirect("../../login")
else:
    return redirect("../../login")

'''*****
*****
*'''

#HM SEARCH STUDENTS IN ROOM NO
def view_enroll_h_rno(request):
    if request.session.has_key('user'):
        user_details = request.session['user']
        if user_details[1] == 3:
            if request.method == "POST":
                details = {'name1':"-","roll_no1": "-","gender1": "-","course1": "-","ph_no1": "-","email1": "-","p_name1": "-","p_ph_no1": "-","hostel": "-","floor": "-","room": "-","name2": "-","roll_no2": "-","gender2": "-","course2": "-","ph_no2": "-","email2": "-","p_name2": "-","p_ph_no2": "-"},
                fid=request.POST['floor_id']
                rid=request.POST['room_id']
                if fid == '' or rid=='':
                    return
            render(request,"view_enroll_h_rno.html",{ 'error': 'Kindly enter all values.'})
        else:
            cursor = connection.cursor()

```



```
        query = "select a.h_id, b.h_name from hostel_manager as a,
hostel_details as b where a.hm_id = '{} ' and a.h_id=b.h_id;"
        query = query.format(request.session['user'][0])
        cursor.execute(query)
        hos = cursor.fetchall()[0]
        hid = hos[0]
        hname = hos[1]

        if (hname != 'MHB 2' and hname != 'MLH' and int(fid) > 3) or
((hname == 'MHB 2' or hname == 'MLH') and int(fid) > 9):
            return
        render(request, "view_enroll_h_rno.html", {'error': 'Floor value exceeds number of
hostel floors.'})

        if (hname != 'MHB 2' and hname != 'MLH' and int(rid) > 50)
or ((hname == 'MHB 2' or hname == 'MLH') and int(rid) > 100):
            return
        render(request, "view_enroll_h_rno.html", {'error': 'Room value exceeds number of
hostel rooms on a floor.'})

        details['hostel'] = hname
        details['floor'] = fid
        details['room'] = '0'*(3-len(str(rid))) + str(rid)

        query = "select * from student_details, student_allocation_h
where student_details.roll_no=student_allocation_h.roll_no and h_id='{} ' and
floor_id='{} ' and room_id='{} '"
        query = query.format(hid, fid, rid)
        cursor.execute(query)
        res = cursor.fetchall()
        if len(res) >= 1:
            res1=res[0]
            details['name1'] = res1[0]
            details['roll_no1'] = res1[1]
            details['gender1'] = res1[2]
            details['course1'] = res1[3]
            details['ph_no1'] = res1[4]
            details['email1'] = res1[5]
            details['p_name1'] = res1[6]
            details['p_ph_no1'] = res1[7]

            if details['course1'] == 1:
                details['course1'] = "UG First Year"
            elif details['course1'] == 2:
                details['course1'] = "UG Second Year"
            elif details['course1'] == 3:
                details['course1'] = "UG Third Year"
            elif details['course1'] == 4:
```

```

        details['course1'] = "UG Fourth Year"
    elif details['course1'] == 5:
        details['course1'] = "PG First Year"
    else:
        details['course1'] = "PG Second Year"

    if len(res) == 2:
        res2=res[1]
        details['name2'] = res2[0]
        details['roll_no2'] = res2[1]
        details['gender2'] = res2[2]
        details['course2'] = res2[3]
        details['ph_no2'] = res2[4]
        details['email2'] = res2[5]
        details['p_name2'] = res2[6]
        details['p_ph_no2'] = res2[7]

        if details['course2'] == 1:
            details['course2'] = "UG First Year"
        elif details['course2'] == 2:
            details['course2'] = "UG Second Year"
        elif details['course2'] == 3:
            details['course2'] = "UG Third Year"
        elif details['course2'] == 4:
            details['course2'] = "UG Fourth Year"
        elif details['course2'] == 5:
            details['course2'] = "PG First Year"
        else:
            details['course2'] = "PG Second Year"

    print(details)
    return render(request,"view_enroll_h_rno_1.html",details)

else:
    return render(request,"view_enroll_h_rno.html")
else:
    return redirect("../../login")
else:
    return redirect("../../login")

'''*****
*****'''

#ADMIN SEARCH BY ROLLNO
def view_enroll_admin_rollno(request):
    if request.session.has_key('user'):
        user_details = request.session['user']
        if user_details[1] == 1:

```

```
        if request.method == "POST":
            rno = request.POST['roll_no']
            if rno == '':
                return
render(request, "view_enroll_admin_rollno.html", {'error': 'Kindly enter a roll
number.'})
        else:
            details = {}
            cursor = connection.cursor()

            query = "select * from student_details where roll_no =
'{}';"

            query = query.format(rno)
            cursor.execute(query)
            res1 = cursor.fetchall()

            if len(res1) == 0:
                return
render(request, "view_enroll_admin_rollno.html", {'error': 'Student not
registered'})

            res1 = res1[0]

            query = "select b.h_name, a.floor_id, a.room_id from
student_allocation_h as a, hostel_details as b where a.roll_no = '{}' and a.h_id
= b.h_id;"

            query = query.format(rno)
            cursor.execute(query)
            res2 = cursor.fetchall()

            query = "select b.m_name from student_allocation_m as a,
mess_details as b where a.roll_no = '{}' and a.m_id = b.m_id;"
            query = query.format(rno)
            cursor.execute(query)
            res3 = cursor.fetchall()

            details['name'] = res1[0]
            details['roll_no'] = res1[1]
            details['gender'] = res1[2]
            details['course'] = res1[3]
            details['ph_no'] = res1[4]
            details['email'] = res1[5]
            details['p_name'] = res1[6]
            details['p_ph_no'] = res1[7]

            if len(res2):
                res2 = res2[0]
                details['hostel'] = res2[0]
```

```

        details['floor'] = res2[1]
        details['room'] = '0'*(3-len(str(res2[2]))) +
str(res2[2])
    else:
        details['hostel'] = "N/A"
        details['floor'] = "N/A"
        details['room'] = "N/A"

    if len(res3):
        details['mess'] = res3[0][0]
    else:
        details['mess'] = "N/A"

    if details['course'] == 1:
        details['course'] = "UG First Year"
    elif details['course'] == 2:
        details['course'] = "UG Second Year"
    elif details['course'] == 3:
        details['course'] = "UG Third Year"
    elif details['course'] == 4:
        details['course'] = "UG Fourth Year"
    elif details['course'] == 5:
        details['course'] = "PG First Year"
    else:
        details['course'] = "PG Second Year"

    return
render(request,"display_enroll_admin_rollno.html",details)
    else:
        return render(request,"view_enroll_admin_rollno.html")
    else:
        return redirect("../../login")
else:
    return redirect("../../login")

'''*****
*****'''

# STUDENT VIEW AND UPDATE PROFILE
def student_profile(request):
    if request.session.has_key('user'):
        user_details = request.session['user']
        if user_details[1] == 2:
            details = {'error': ''}
            if request.method == "POST":
                roll_no = request.session['user'][0]
                new_details = request.POST

```

```
        for i in new_details.values():
            if i == '':
                details['error'] = 'Kindly enter all values.'
                break
            if not(new_details['ph_no'].isnumeric and
len(new_details['ph_no']) == 10) or not(new_details['p_ph_no'].isnumeric and
len(new_details['p_ph_no']) == 10):
                details['error'] = 'Phone numbers should be ten-digit Indian
phone numbers.'
            if details['error'] == '':
                cursor = connection.cursor()
                print(new_details)
                query = "update student_details set name = '{}', course_id =
'{}', ph_no = '{}', p_name = '{}', p_ph_no = '{}' where roll_no = '{}';"
                query =
query.format(new_details['name'],new_details['course'],new_details['ph_no'],new_
details['p_name'],new_details['p_ph_no'],roll_no)
                cursor.execute(query)
                return render(request,"student_profile_1.html")

roll_no = request.session['user'][0]
cursor = connection.cursor()

query = "select * from student_details where roll_no = '{}';"
query = query.format(roll_no)
cursor.execute(query)
res1 = cursor.fetchall()[0]

query = "select b.h_name, a.floor_id, a.room_id from
student_allocation_h as a, hostel_details as b where a.roll_no = '{}'' and a.h_id
= b.h_id;"
query = query.format(roll_no)
cursor.execute(query)
res2 = cursor.fetchall()

query = "select b.m_name from student_allocation_m as a,
mess_details as b where a.roll_no = '{}'' and a.m_id = b.m_id;"
query = query.format(roll_no)
cursor.execute(query)
res3 = cursor.fetchall()

details['name'] = res1[0]
details['roll_no'] = res1[1]
details['gender'] = res1[2]
details['course'] = res1[3]
details['ph_no'] = res1[4]
details['email'] = res1[5]
details['p_name'] = res1[6]
```

```

        details['p_ph_no'] = res1[7]

    if len(res2):
        res2 = res2[0]
        details['hostel'] = res2[0]
        details['floor'] = res2[1]
        details['room'] = '0'*(3-len(str(res2[2]))) + str(res2[2])
    else:
        details['hostel'] = "N/A"
        details['floor'] = "N/A"
        details['room'] = "N/A"

    if len(res3):
        details['mess'] = res3[0][0]
    else:
        details['mess'] = "N/A"

    if details['course'] == 1:
        details['course'] = "UG 1st Year"
        details['course_id'] = 1
    elif details['course'] == 2:
        details['course'] = "UG 2nd Year"
        details['course_id'] = 2
    elif details['course'] == 3:
        details['course'] = "UG 3rd Year"
        details['course_id'] = 3
    elif details['course'] == 4:
        details['course'] = "UG 4th Year"
        details['course_id'] = 4
    elif details['course'] == 5:
        details['course'] = "PG 1st Year"
        details['course_id'] = 5
    else:
        details['course'] = "PG 2nd Year"
        details['course_id'] = 6

    return render(request,"student_profile.html",details)
else:
    return redirect_modules("../../login")
else:
    return redirect("../../login")

'''*****
*****'''

#EXPEL STUDENT
def expel_student(request):
    if request.session.has_key('user'):

```

```

        user_details = request.session['user']
        if user_details[1] == 1:
            if request.method=="POST":
                entered_details = request.POST
                rollno = entered_details['rollno']
                if (rollno == ''):
                    return render(request, 'expel_student.html',{'error':"Kindly
enter all values."})
                if (rollno_validation(rollno)==-1):
                    return render(request,
'expel_student.html',{'error':"Invalid roll number."})
                vacate_hostel(rollno)
                vacate_mess(rollno)
                cursor = connection.cursor()
                expel_password = "ulsjy@lt$bj#nk$adf^nkiB307SGBD"
                update_query = "update login_cred set password = '{ }' where u_id
= '{ }'"

                update_query = update_query.format(expel_password,rollno)
                cursor.execute(update_query)
                return render(request, 'expel_student_1.html')
            else:
                return render(request, 'expel_student.html')
        else:
            return redirect_modules("../../login")
    else:
        return redirect("../../login")

```

```

'''*****
*****'''

```

#VACATE STUDENT

```

def vacate_student(request):
    if request.session.has_key('user'):
        user_details = request.session['user']
        if user_details[1] == 3:
            if request.method=="POST":
                entered_details = request.POST
                rollno = entered_details['rollno']
                if (rollno == ''):
                    return render(request,
'vacate_student.html',{'error':"Kindly enter all values."})
                if (rollno_validation(rollno)==-1):
                    return render(request,
'vacate_student.html',{'error':"Invalid roll number."})
                hmid = request.session['user'][0]
                cursor = connection.cursor()
                query = "select * from hostel_manager where hm_id = '{ }'"
                query = query.format(hmid)

```

```

        cursor.execute(query)
        y = cursor.fetchall()

        hid=y[0][2]

        query = "select * from student_allocation_h where roll_no = '{}'"
and h_id={}
        query = query.format(rollno,hid)
        cursor.execute(query)
        y = cursor.fetchall()

        if(len(y)==0):
            return render(request,
'vacate_student.html',{'error':"Student not allotted to your hostel"})
            vacate_hostel(rollno)
            return render(request, 'vacate_student_1.html')
        else:
            return render(request, 'vacate_student.html')
    else:
        return redirect_modules("../../login")
else:
    return redirect("../../login")

'''*****
*****'''
def vacate_hostel(rno):
    cursor = connection.cursor()
    query = "select * from student_allocation_h where roll_no = '{}'"
    query = query.format(rno)
    cursor.execute(query)
    y = cursor.fetchall()

    if len(y)==1:
        hid = y[0][0]
        fid = y[0][1]
        rid = y[0][2]

        query = "update room_details set status = status-1 where h_id = {} and
floor_id = {} and room_id = {}"
        query = query.format(hid,fid,rid)
        cursor.execute(query)

        query = "update hostel_details set free_room = free_room+1 where h_id =
{}"
        query = query.format(hid)
        cursor.execute(query)

        query = "delete from student_allocation_h where roll_no = '{}'"

```



```
        query = query.format(rno)
        cursor.execute(query)
    return 0

def vacate_mess(rno):
    cursor = connection.cursor()
    query = "select * from student_allocation_m where roll_no = '{}'"
    query = query.format(rno)
    cursor.execute(query)
    y = cursor.fetchall()

    if len(y)==1:
        mid = y[0][0]

        query = "update mess_details set allocated=allocated-1 where m_id = {}"
        query = query.format(mid)
        cursor.execute(query)

        query = "delete from student_allocation_m where roll_no = '{}'"
        query = query.format(rno)
        cursor.execute(query)
    return 0

'''*****
*****'''
```

12 References

12.1 BIBLIOGRAPHY

- Arora, S. (2021). *Computer Science With Python*. New Delhi: DHANPAT RAI & CO.
- Elmasri, R., & Navathe, S. B. (2003). *Fundamentals of Database Systems*. New York: Pearson Education.

12.2 WEBLIOGRAPHY

- materializecss.com
- stackoverflow.com
- [geeksforgeeks.org](https://www.geeksforgeeks.org)
- [w3schools.com](https://www.w3schools.com)
- [javatpoint.com](https://www.javatpoint.com)
- [tutorialspoint.com](https://www.tutorialspoint.com)