UNIVERSITY APP

A MINI-PROJECT REPORT

18CSC207J - ADVANCED PROGRAMMING PRACTICE

Submitted by

GAURANG ASHAVA (RA2111027010007)
SYED ADNAN HUSSAINY (RA2111027010008)

Under the guidance of

Dr. Sasikumar

Assistant Professor, Department of Data Science and Business Systems

in partial fulfillment of the award of the degree

of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

of

FACULTY OF ENGINEERING AND TECHNOLOGY

SRM
INSTITUTE OF SCIENCE & TECHNOLOGY
Deemed to be University u/s 3 of UGC Act, 1956

S.R.M. Nagar, Kattankulathur, Chengalpattu District

MAY 2023

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that the Mini project report titled **"COLLEGE HOSTEL MANAGEMENT SYSTEM"** is the bonafide work of **GAURANG ASHAVA (RA2111027010007) & SYED ADNAN HUSSAINY (RA2111027010008)** who carried out the minor project under my supervision. Certified further, that to the best of my knowledge, the work reported herein does not form any other project report or dissertation based on which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

Dr. Sasikumar
SUPERVISOR
Assistant Professor
Department of Data Science and Business Systems

SIGNATURE

Dr. M Lakshmi
HEAD OF THE DEPARTMENT
Professor and Head
Department of Data Science and Business Systems

ABSTRACT

This project presents a University Hostel Management System implemented using Python, Tkinter, and a database. The system is designed to simplify the management of hostel accommodation for university students.

The system has two main user interfaces: one for administrators and another for students. The administrator interface allows for adding new hostels, allocating rooms to students, and managing student records. The student interface allows students to view their allocated rooms, make room transfer requests, and view hostel information.

The system is implemented using Pythons Tkinter library for the graphical user interface, and a database is used to store all data related to the hostels and students. The database is designed to store and retrieve data while maintaining integrity. The system offers various features such as generating reports and managing billing.

Overall, the University Hostel Management System provides a simple, yet effective way of managing hostel accommodation for university students. It eliminates manual processes and enhances accuracy and efficiency.

It is an effective tool for hostel administrators and students, making the entire process streamlined and user-friendly.

TABLE OF CONTENTS

ABBREVIATIONS

- **CHMS:-** College Hostel Management System
- **DOB :-** Date of Birth
- **BRD:-** Bedroom with Bed No.
- **DTH:-** Direct to Home Television
- **HOD :-** Hostel Office Desk
- **MT :-** Maintenance
- **GUI :-** Graphical User Interface
- **IDE :-** Integrated Development Environment
- **CRUD:-** Create, Read, Upload, Delete
- **DBMS :-** Database Management System
- **SQL :-** Structured Query Language
- **API :-** Application Programming Interface

INTRODUCTION

The University Hostel Management System is an application designed to provide efficient and reliable management of university hostel accommodation for both administrators and students.

The system is implemented using Python, Tkinter, and a database, which provide a user-friendly interface, powerful functionality, and efficient data management capabilities.

The system is designed to automate the entire process of managing hostel accommodation for university students. It simplifies the process of room allocation, billing, and management of student records.

The system provides two user interfaces: one for administrators and another for students. The administrator interface enables hostel administrators to manage hostel facilities, allocate rooms to students, and manage student records.

The student interface, on the other hand, enables students to view their allocated rooms, make room transfer requests, and access information about the hostels.

Python is a widely used programming language known for its simplicity, flexibility, and ease of use. The Tkinter library, which is a standard GUI toolkit for Python, provides an intuitive graphical interface for the system. The database used in the system is designed to store and retrieve data efficiently while maintaining data integrity.

The University Hostel Management System aims to eliminate manual processes and enhance accuracy and efficiency in managing hostel accommodation for university students. The system is an effective tool for hostel administrators and students, providing a streamlined and user-friendly process.

The system's features, including generating reports and managing bills, make it an essential tool for efficient and reliable management of hostel accommodation in universities.

LITERATURE SURVEY

The management of university hostels is an essential aspect of university administration, and over the years, various systems have been developed to simplify and streamline the process. In recent times, technological advancements have led to the development of computer-based systems for the management of hostel accommodation. In this literature survey, we review some related works that have been developed for the management of hostel accommodation in universities.

One study by Rajesh et al. (2018) presents a web-based system for the management of hostel accommodation in a university. The system is designed to automate the entire process of managing hostel accommodation for university students, including room allocation, billing, and managing student records. The system uses PHP and MySQL for the back end and HTML, CSS, and JavaScript for the front end. The system provides a user-friendly interface, and it has been tested and proven to be effective in managing hostel accommodation.

Another study by Ali et al. (2019) presents a hostel management system that uses a desktop-based application to manage hostel accommodation in a university. The system is designed to automate the process of managing student records, room allocation, and billing. The system is implemented using Java and MySQL, and it provides a user-friendly interface for hostel administrators and students.

In a different approach, Adeniyi et al. (2020) present a mobile-based system for the management of hostel accommodation in a university. The system uses a mobile application to provide students with access to hostel information, including room allocation, hostel facilities, and payment information. The system is implemented using Android and MySQL, and it provides a simple and efficient way for students to manage their hostel accommodation.

In this work, we present a University Hostel Management System implemented using Python, Tkinter, and a database. The system is designed to automate the entire process of managing hostel accommodation for university students, and it provides a user-friendly interface for hostel administrators and students. The system is expected to be effective in managing hostel accommodation, eliminating manual processes, and enhancing accuracy and efficiency.

SYSTEM ARCHITECTURE

The University Hostel Management System is implemented using a client-server architecture with a desktop-based client application and a database server. The system is designed to automate the process of managing hostel accommodation for university students, including room allocation, billing, and management of student records. The system is implemented using Python, Tkinter, and a database.

The client application is implemented using Python and Tkinter, which provide a user-friendly graphical interface for hostel administrators and students. The client application communicates with the database server using Structured Query Language (SQL) to retrieve and store data. The database server is implemented using a Relational Database Management System (RDBMS), which stores and manages data efficiently while ensuring data integrity.

The system is designed using a Model-View-Controller (MVC) architecture, which separates the system into three components: the model, view, and controller. The model represents the data and business logic of the system, the view represents the user interface, and the controller represents the application logic that mediates between the model and view.

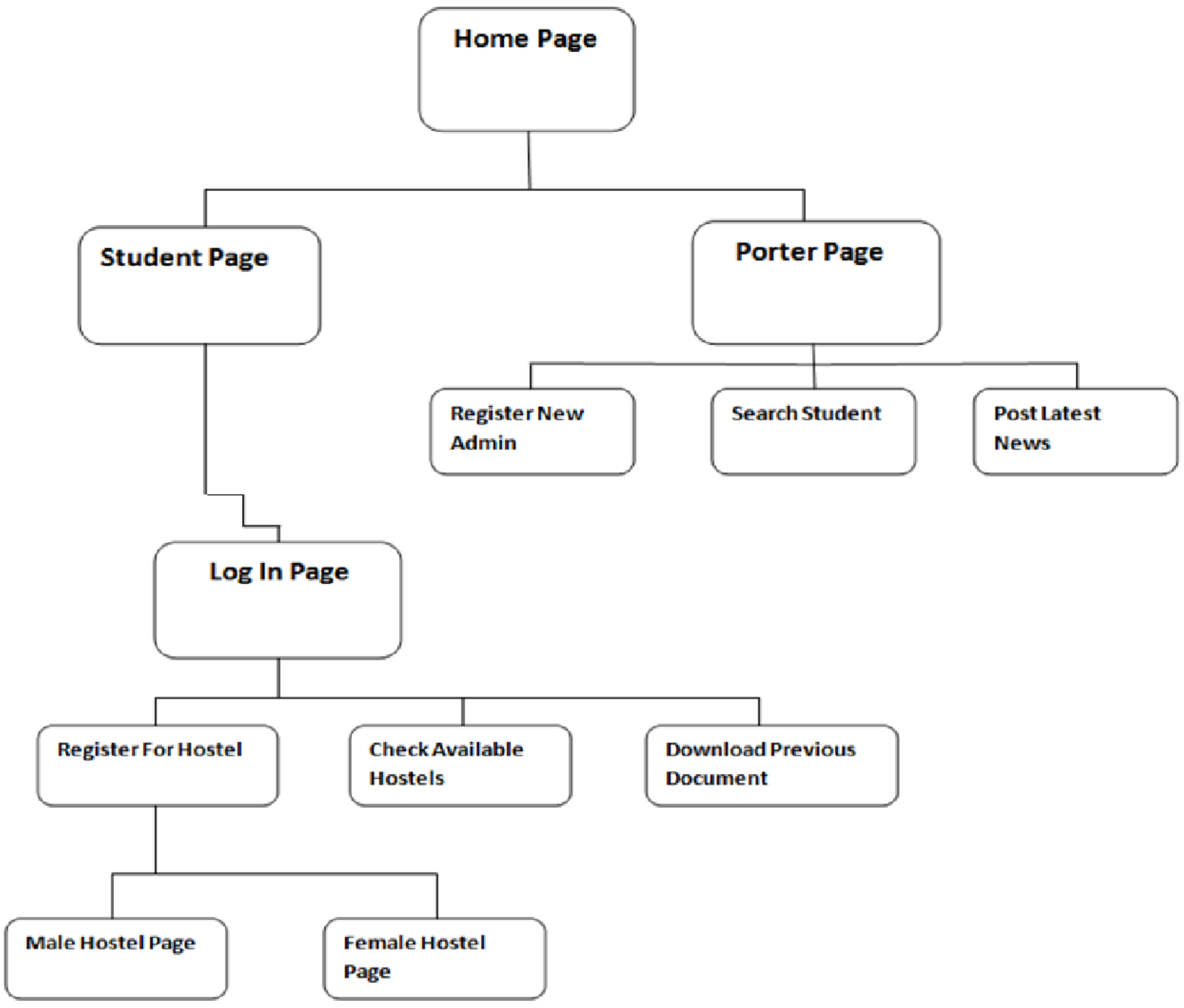
The model component of the system consists of the database schema and the Python classes that represent the data entities in the database. The database schema defines the structure of the database and the relationships between data entities, while the Python classes provide an object-oriented interface for accessing and manipulating data.

DESIGN

The view component of the system consists of the user interface implemented using Tkinter. The user interface provides a graphical interface for hostel administrators and students to interact with the system. The user interface includes forms for inputting data, tables for displaying data, and buttons for triggering system events.

The controller component of the system consists of Python scripts that mediate between the model and view components. The controller scripts receive user input from the view, process the input, and update the model and view accordingly. The controller scripts also communicate with the database server to retrieve and store data.

Overall, the University Hostel Management System is designed to be a reliable and efficient system for managing hostel accommodation in universities. The system architecture and design provide a scalable and extensible framework that can be customized to meet the specific needs of different universities.



METHODOLOGY

The development of the University Hostel Management System using Python, Tkinter, and a database follows a structured methodology that includes the following steps:

1) **Requirements gathering:** The first step in the development process is to gather and analyze the requirements for the system. This involves identifying the needs of hostel administrators and students and the features and functionalities required in the system. The requirements are documented in a software requirements specification (SRS) document.

2) **System analysis and design:** In this step, the system is analysed and designed based on the requirements gathered in the previous step. The system is broken down into components, and the architecture and design of each component are determined. This involves creating a detailed design specification (DDS) document, which includes system requirements, use cases, data flow diagrams, class diagrams, and other system design details.

3) **Implementation:** The implementation phase involves the actual development of the system components. This step includes writing code for the client application using Python and Tkinter and setting up the database server using an RDBMS. The code is tested at various stages of development to ensure that it is functioning as expected.

4) **Integration and testing:** In this step, the different system components are integrated and tested to ensure that they work together seamlessly. This involves testing the system for bugs, errors, and other issues and fixing any issues that are found.

5) **Deployment and maintenance:** The final step in the development process is to deploy the system and maintain it. The system is deployed in the university hostel environment, and users are trained to use the system. The system is monitored for performance and security issues, and maintenance and support are provided as required.

Overall, the development of the University Hostel Management System follows a structured methodology that ensures that the system is developed in a systematic and organized manner. This methodology ensures that the system meets the requirements of hostel administrators and students and is reliable, efficient, and user-friendly.

CODING

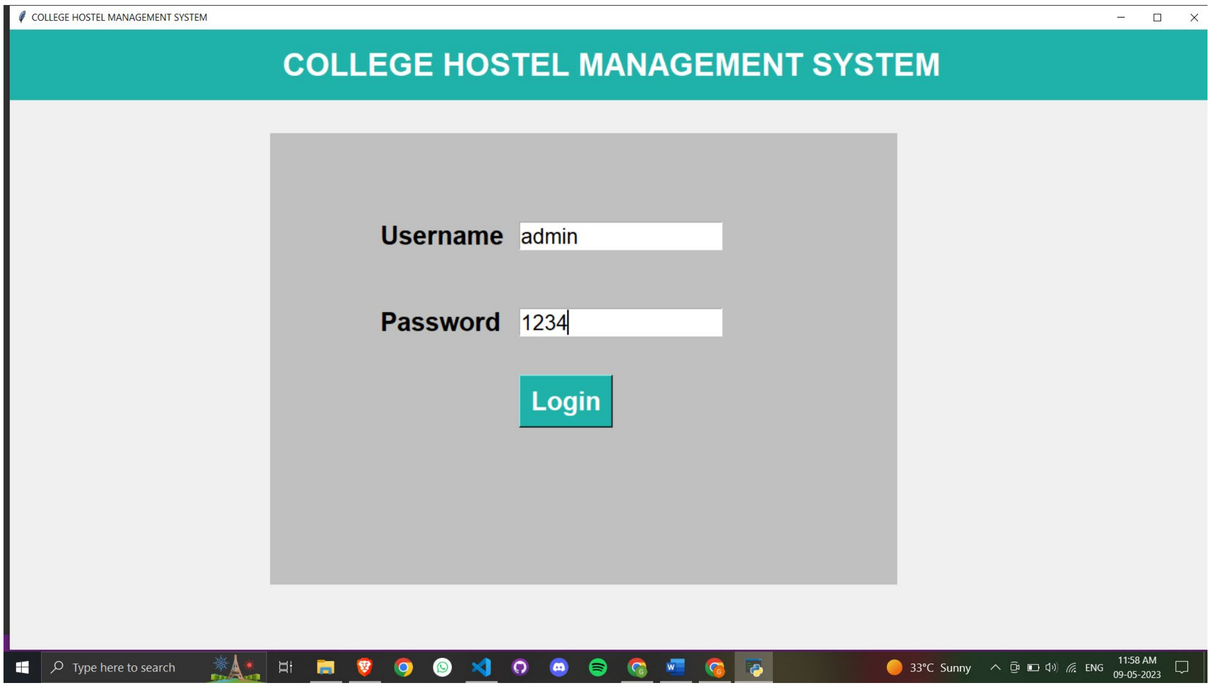
```
from tkinter.ttk import *
from Tkinter import *
from PIL import ImageTk, Image
def date():
    from datetime import datetime
    # current date time
    now = datetime.now()
    t = now.strftime("%H:%M")
    s1 = now.strftime("%H:%M,%Y-%m-%d")
    s1 = str(s1)
    return s1 #returns Current Date And Time In String
base = Tk()
base.title("COLLEGE HOSTEL MANAGEMENT SYSTEM")
base.geometry(f'{1535}x{790}+{0}+{0}')
heading = Label(base, text="COLLEGE HOSTEL MANAGEMENT SYSTEM", font=("Arial 30
bold"), bg="lightseagreen", fg="white", padx=490, pady=20)
heading.pack()
canvas = Canvas(base, bg='silver', height=575, width=800)
canvas.place(x=330, y=130)
G = 1
def main():
    global G
    canvas = Canvas(base, bg='lightseagreen', height=675, width=310)
    canvas.place(x=-1, y=100)
    can = Canvas(base, bg='silver', height=675, width=1500)
    can.place(x=320, y=105)
    def add_stud():
        canvas = Canvas(base, bg='silver', height=675, width=1210)
        canvas.place(x=320, y=105)
        first_name = Label(base, text="First Name", font=("Arial 15 bold"),
bg='silver', fg="black")
        first_name.place(x=400, y=150)
        fir_name_entry = Entry(base, width=15, font=("Arial 15"))
        fir_name_entry.place(x=400, y=180)
        fir_name_entry.focus()
        last_name = Label(base, text="Last Name", font=("Arial 15 bold"),
bg="silver", fg="black")
        last_name.place(x=610, y=150)
        last_name_entry = Entry(base, width=15, font=("Arial 15"))
        last_name_entry.place(x=610, y=180)
        father_name = Label(base, text="Father Name", font=("Arial 15 bold"),
bg="silver", fg="black")
        father_name.place(x=400, y=220)
        fathr_name_entry = Entry(base, width=15, font=("Arial 15"))
        fathr_name_entry.place(x=400, y=250)
        mother_name = Label(base, text="Mother Name", font=("Arial 15 bold"),
bg="silver", fg="black")
        mother_name.place(x=610, y=220)
        mther_name_entry = Entry(base, width=15, font=("Arial 15"))
        mther_name_entry.place(x=610, y=250)
        dob = Label(base, text="DOB", font=("Arial 15 bold"), bg="silver",
fg="black")
        dob.place(x=400, y=300)
        dob_entry = Entry(base, width=15, font=("Arial 15 bold"))
        dob_entry.place(x=400, y=330)
        m1 = Label(base, text="(Y-M-D)", font=("Arial 12 bold"), bg="silver",
fg="black")
        m1.place(x=450, y=300)
        contact = Label(base, text="Roll No", font=("Arial 15 bold"),
bg="silver", fg="black")
        contact.place(x=610, y=300)
        cont_entry = Entry(base, width=15, font=("Arial 15 bold"))
        cont_entry.place(x=610, y=330)
        message = Label(base, text="(Roll No. will be your Hostel ID)",
font=("Arial 10 bold"), bg="silver",fg="black")
        message.place(x=690, y=300)
        email = Label(base, text="Email", font=("Arial 15 bold"), bg="silver",
fg="black")
        email.place(x=400, y=370)
        email_entry = Entry(base, width=15, font=("Arial 15 bold"))
        email_entry.place(x=400, y=410)
        address = Label(base, text="Address", font=("Arial 15 bold"),
bg="silver", fg="black")
        address.place(x=610, y=370)
        addrs_entry = Entry(base, width=15, font=("Arial 15 bold"))
        addrs_entry.place(x=610, y=410)
        vehicle = Label(base, text="Vehicle No.", font=("Arial 15 bold"),
bg="silver", fg="black")
        vehicle.place(x=400, y=450)
        vehicle_entry = Entry(base, width="15", font=("Arial 15 bold"))
        vehicle_entry.place(x=400, y=490)
        m2 = Label(base, text="(MH20EU9295)", font=("Arial 11 bold"),
bg="silver", fg="black")
        m2.place(x=470, y=450)
        work_place = Label(base, text="Work Place/College", font=("Arial 15
bold"), bg="silver", fg="black")
        work_place.place(x=610, y=450)
        place_entry = Entry(base, width="15", font=("Arial 15 bold"))
        place_entry.place(x=610, y=490)
        gender = Label(base, text="Gender", font=("Arial 15 bold"),
bg="silver", fg="black")
        gender.place(x=400, y=580)
        G = 1
        def selected():
            global G
            if c1.get() == 1:
                G = 1
            elif c1.get() == 2:
                G = 2
            elif c1.get() == 0:
                G = 0
            else:
                G = 1
        c1 = IntVar()
        a = Radiobutton(base, text="Male", bg="silver", fg="black",
font=("Arial 15 bold"), variable=c1, value=1, command=selected)
        b = Radiobutton(base, text="Female", bg="silver", fg="black",
font=("Arial 15 bold"), variable=c1, value=2, command=selected)
        c = Radiobutton(base, text="Other", bg="silver", fg="black",
font=("Arial 15 bold"), variable=c1, value=0, command=selected)
        a.place(x=490, y=580)
        b.place(x=580, y=580)
        c.place(x=680, y=580)
        def available_roome():
            rooms_availble = Label(base, text="Rooms Available", font=("Arial
20 bold"), bg="lightseagreen", fg="white", padx=140)
            rooms_availble.place(x=1005, y=110)
            x = " Room No. | Beds"
            room = Label(base, text=x, font=("Arial 15 bold"), bg="dark slate
grey", fg="white")
            room.place(x=1005, y=150)
            global G
            bed1 = None
            bed2 = None
            bed3 = None
            if G == 1:
                f1 = open("room_info_boys.txt","r")
                bed1 = "B1"
                bed2 = "B2"
                bed3 = "B3"
            elif G==2:
                f1 = open("room_info_girls.txt","r")
                bed1 = "G1"
                bed2 = "G2"
                bed3 = "G3"
            elif G==0:
                f1 = open("room_info_others.txt","r")
                bed1 = "O1"
                bed2 = "O2"
                bed3 = "O3"
            else:
                file_name = "room_info_boys.txt"
                bed1 = "B1"
                bed2 = "B2"
                bed3 = "B3"
            all_lines = f1.readlines()
            count = 1
            rooms = []
            for i in all_lines:
                temp1 = str(i)
                one_line = temp1.split(',')
                if one_line[1] == bed1 and one_line[2] == bed2 and one_line[3]
== bed3:
                    temp = []
                    count = 2
                    temp.append(one_line[0])
                    temp.append(bed1)
                    temp.append(bed2)
                    temp.append(bed3)
                    rooms.append(temp)
                if one_line[1] != bed1 and one_line[2] == bed2 and one_line[3]
== bed3:
                    count = 2
                    temp = []
                    temp.append(one_line[0])
                    temp.append("NA")
                    temp.append(bed2)
                    temp.append(bed3)
                    rooms.append(temp)
                if one_line[1] == bed1 and one_line[2] != bed2 and one_line[3]
== bed3:
                    temp = []
                    count = 2
                    temp.append(one_line[0])
                    temp.append(bed1)
                    temp.append("NA")
                    temp.append(bed3)
                    rooms.append(temp)
                if one_line[1] == bed1 and one_line[2] == bed2 and one_line[3]
!= bed3:
                    temp = []
                    count = 2
                    temp.append(one_line[0])
                    temp.append(bed1)
                    temp.append(bed2)
                    temp.append("NA")
                    rooms.append(temp)
                if one_line[1] == bed1 and one_line[2] != bed2 and one_line[3]
!= bed3:
                    temp = []
                    count = 2
                    temp.append(one_line[0])
                    temp.append(bed1)
                    temp.append("NA")
                    temp.append("NA")
                    rooms.append(temp)
                if one_line[1] != bed1 and one_line[2] == bed2 and one_line[3]
!= bed3:
                    temp = []
                    count = 2
                    temp.append(one_line[0])
                    temp.append("NA")
                    temp.append(bed2)
                    temp.append("NA")
                    rooms.append(temp)
                if one_line[1] != bed1 and one_line[2] != bed2 and one_line[3]
== bed3:
                    temp = []
                    count = 2
                    temp.append(one_line[0])
                    temp.append("NA")
                    temp.append("NA")
                    temp.append(bed3)
                    rooms.append(temp)
                if one_line[1] != bed1 and one_line[2] != bed2 and one_line[3]
!= bed3:
                    temp = []
                    count = 2
                    temp.append("--")
                    temp.append("NA")
                    temp.append("NA")
                    temp.append("NA")
                    rooms.append(temp)
            if count == 1:
                y = "No Rooms Available"
                x = Label(base, text=y, font=("Arial", 40), bg="dark slate
grey", fg="white")
                x.place(x=1020, y=300)
            else:
                x1co = 1150
                y1co = 200
                x2co = 1350
                y2co = 200
                flag = 1
                for i in rooms:
                    r_no = Label(base, text=i[0], font=("Arial", 15),
bg="silver", fg="black")
                    r_no.place(x=x1co, y=y1co)
                    y = i[1] + " " + i[2] + " " + i[3]
                    r_bk_no = Label(base, text=y, font=("Arial", 15),
bg="silver", fg="black")
                    r_bk_no.place(x=x2co, y=y2co)
                    y1co = y1co + 40
                    y2co = y2co + 40
                    count = 2
                    if flag >= 8:
                        break
                    flag = flag + 1
            f1.close()
            c1 = Canvas(base, height=2, width=600)
            c1.place(x=1000, y=550)
            r1 = Label(base, text="Room No.", font=("Arial 15 bold"),
bg="silver", fg="black")
            r1.place(x=1040, y=570)
            r2 = Entry(base, width=15, font=("Arial 15 bold"))
            r2.place(x=1150, y=570)
            r3 = Label(base, text="Bed No.", font=("Arial 15 bold"),
bg="silver", fg="black")
            r3.place(x=1040, y=610)
            r4 = Entry(base, width=15, font=("Arial 15 bold"))
            r4.place(x=1150, y=610)
            r5 = Label(base, text="(B1 B2 B3)", font=("Arial 10 bold"),
bg="silver", fg="black")
            r5.place(x=1040, y=640)
            r6 = Label(base, text="( Girl = G1 )", font=("Arial 15 bold"),
bg="silver", fg="black")
            r6.place(x=1350,y=570)
            r7 = Label(base, text="( Other = O1)",font=("Arial 15 bold"),
bg="silver", fg="black")
            r7.place(x=1350,y=610)
            def student():
                global G
                bed1 = None
                bed2 = None
                bed3 = None
                file_name = None
                if G == 1:
                    file_name = "room_info_boys.txt"
                    bed1 = "B1"
                    bed2 = "B2"
                    bed3 = "B3"
                elif G == 2:
                    file_name = "room_info_girls.txt"
                    bed1 = "G1"
                    bed2 = "G2"
                    bed3 = "G3"
                elif G == 0:
                    file_name = "room_info_others.txt"
                    bed1 = "O1"
                    bed2 = "O2"
                    bed3 = "O3"
                else:
                    file_name = "room_info_boys.txt"
                    bed1 = "B1"
                    bed2 = "B2"
                    bed3 = "B3"
                f1 = open("student_info.txt", "a")
                f2 = open(file_name,"a")
                n = str(fir_name_entry.get()) + " "
                b = str(r4.get()).upper()
                ln = str(last_name_entry.get()).lower()
                f = str(fathr_name_entry.get()).lower()
                m = str(mther_name_entry.get()).lower()
                d = str(dob_entry.get()).lower()
                add = str(addrs_entry.get()).lower()
                c = str(cont_entry.get()).lower()
                e = str(email_entry.get())
                w = str(place_entry.get()).lower()
                v = str(vehicle_entry.get()).lower()
                da = date().lower()
                rom = str(r2.get()).lower()
                f1.write(
                c + "," + n + ln + "," + f + "," + m + "," + d + "," + e + ","
+ w + "," + v + "," + da + "," +
                b + "," + rom + "," + "\n")
                f1.close()
                fobj = open(file_name, "r")
                fdata_ls = fobj.readlines()
                fobj.close()
                rdate = date()
                fobj = open(file_name, "w")
                if b == bed1:
                    for oneline in fdata_ls:
                        if oneline.startswith(rom + ",") and
oneline.__contains__(","+bed1):  # write date
                            new_oneline = oneline.replace(","+bed1+",", "," +
c + ",")

                            new_oneline2 = new_oneline.replace(",NOT,", "," +
rdate + ",")

                            fobj.write(new_oneline)
                        else:
                            fobj.write(oneline)
                else:
                    if b == bed2:
                        for oneline in fdata_ls:
                            if oneline.startswith(rom + ",") and
oneline.__contains__(","+bed2): # write date
                                new_oneline = oneline.replace(","+bed2+"," ,
"," + c + ",")

                                new_oneline2 = new_oneline.replace(",NOT," ,
"," + rdate + ",")

                                fobj.write(new_oneline)
                            else:
                                fobj.write(oneline)
                    else:
                        for oneline in fdata_ls:
                            if oneline.startswith(rom + ",") and
oneline.__contains__(","+bed3): # write date
                                new_oneline = oneline.replace(","+bed3+"," ,
"," + c + ",")

                                new_oneline2 = new_oneline.replace(",NOT," ,
"," + rdate + ",")

                                fobj.write(new_oneline)
                            else:
                                fobj.write(oneline)
                fobj.close()
                l = Label(base, text="Student Added Successfully....!",
font=("Arial 15 bold"), bg='silver', fg="black")
                l.place(x=1150, y=650)
            add_student = Button(base, text="Save", font=("Arial 20 bold"),
bg="white", fg="black" ,command=student)
            add_student.place(x=1050, y=700)
        c1 = IntVar()
        a = Radiobutton(base, text="Male", bg="silver", fg="black",
font=("Arial 15 bold"), variable=c1, value=1, command=selected)
        b = Radiobutton(base, text="Female", bg="silver", fg="black",
font=("Arial 15 bold"), variable=c1, value=2, command=selected)
        c = Radiobutton(base, text="Other", bg="silver", fg="black",
font=("Arial 15 bold"), variable=c1, value=0, command=selected)
        a.place(x=490,y=580)
        b.place(x=580,y=580)
        c.place(x=680,y=580)
        continu = Button(base, text="Continue", font=("Arial 15 bold"),
bg="white", fg="black", command=available_roome)
        continu.place(x=490, y=670)
        line = Canvas(base, height=670, width=2)
        line.place(x=1000, y=105)
    def add_room():
            canvas = Canvas(base, bg='silver', height=675, width=1210)
            canvas.place(x=320, y=105)
            h1 = Label(base, text="Add New Room", bg="dark slate grey",
font=("Arial 20 bold"), fg="white", padx=500, pady=10)
            h1.place(x=320, y=105)
            n_rm_n = Label(base, text="New Room No.", font=("Arial 20 bold"),
bg="silver", fg="black")
            n_rm_n.place(x=500, y=300)
            rm_n_entry = Entry(base, width=20, font=("Arial 15 bold"))
            rm_n_entry.place(x=750, y=303)
            gender = Label(base,text="Gender", font=("Arial 20 bold"),
bg="silver", fg="black")
            gender.place(x=500,y=350)
            global G
            G = 1
            def selected():
                global G
                if c1.get() == 1:
                    G = 1
                if c1.get() == 2:
                    G = 2
                if c1.get() == 0:
                    G = 0
            c1 = IntVar()
            a = Radiobutton(base, text="Male", bg="silver", fg="black",
font=("Arial 15 bold"), variable=c1, value=1, command=selected)
            b = Radiobutton(base, text="Female", bg="silver", fg="black",
font=("Arial 15 bold"), variable=c1, value=2, command=selected)
            c = Radiobutton(base, text="Other", bg="silver", fg="black",
font=("Arial 15 bold"), variable=c1, value=0, command=selected)
            a.place(x=750, y=350)
            b.place(x=840, y=350)
            c.place(x=950, y=350)
            def add():
                file_name = ""
                bed1 = bed2 = bed3 = None
                r = (rm_n_entry.get())
                if G==1:
                    bed1 = "B1"
                    bed2 = "B2"
                    bed3 = "B3"
                    file_name = "room_info_boys.txt"
                    f2 = open(file_name, "r")
                if G==2:
                    bed1 = "G1"
                    bed2 = "G2"
                    bed3 = "G3"
                    file_name = "room_info_girls.txt"
                    f2 = open(file_name, "r")
                if G==0:
                    bed1 = "01"
                    bed2 = "02"
                    bed3 = "03"
                    file_name = "room_info_others.txt"
                    f2 = open(file_name, "r")
                all_lines = f2.readlines()
                count = 0
                for i in all_lines:
                    temp = str(i)
                    one_line = temp.split(',')
                    if one_line[0] == r:
                        l = Label(base, text="Room Is Already Addded....!",
font=("Arial 25 bold"), bg='silver', fg="black")
                        l.place(x=650, y=600)
                        f2.close()
                        count = 2
                f2.close()
                if count != 2:
                    q = "NOT"
                    f1 = open(file_name, "a")
                    f1.write(r + "," + bed1 + "," + bed2 + "," + bed3 + "," +
q + "," + "\n")
                    f1.close()
                    l = Label(base, text="ROOM Successfully Added....!",
font=("Arial 25 bold"), bg='silver', fg="black")
                    l.place(x=650, y=600)
            ad_rm_btn = Button(base, text="Add Room", font=("Arial 20 bold"),
bg="white", fg="black", command=add)
            ad_rm_btn.place(x=550, y=500)
    def in_out_time():
            canvas = Canvas(base, bg='silver', height=675, width=1210)
            canvas.place(x=320, y=105)
            a = "---------------Outtime---------------"
            out_time_h1 = Label(base, text=a, font=("Arial 20 bold"), bg="dark
slate grey", fg="white", padx=20, pady=10)
            out_time_h1.place(x=320, y=115)
            out_time = Label(base, text="Enter ID", font=("Arial 20 bold"),
bg="silver", fg="black")
            out_time.place(x=470, y=200)
            out_time_entry = Entry(base, width=20, font=("Arial 20 bold"))
            out_time_entry.place(x=620, y=200)
            out_time_entry.focus()
            purpose = Label(base, text="Purpose", font=("Arial 20 bold"),
bg="silver", fg="black")
            purpose.place(x=470, y=270)
            purpose_entry = Entry(base, width=20, font=("Arial 20 bold"))
            purpose_entry.place(x=620, y=270)
            def save1():
                Id = str(out_time_entry.get())
                pur1 = str(purpose_entry.get())
                t = date()
                f1 = open("inouttime.txt", "a")
                f1.write(Id + "," + pur1 + "," + t + "," + "OUTTIME" + "," +
"REMARK" + "\n")
                f1.close()
                done = Label(base, text="Successfull...!", font=("Arial 20
bold"), bg="silver", fg="black")
                done.place(x=1200, y=220)
            save1 = Button(base, text="Save", font=("Arial 20 bold"),
bg="white", fg="black", command=save1)
            save1.place(x=1050, y=220)
            a1 = "---------------Inttime---------------"
            in_time_h1 = Label(base, text=a1, font=("Arial 20 bold"), bg="dark
slate grey", fg="white", padx=20, pady=10)
            in_time_h1.place(x=320, y=400)
            in_time = Label(base, text="Enter ID", font=("Arial 20 bold"),
bg="silver", fg="black")
            in_time.place(x=470, y=500)
            in_time_entry = Entry(base, width=20, font=("Arial 20 bold"))
            in_time_entry.place(x=680, y=500)
            in_time_entry.focus()
            ot_entry = Label(base, text="Outtime Entry", font=("Arial 20
bold"), bg="silver", fg="black")
            ot_entry.place(x=470, y=550)
            ot_entry_entry = Entry(base, width=20, font=("Arial 20 bold"))
            ot_entry_entry.place(x=680, y=550)
            def search_outtime():
                s_id = str(in_time_entry.get())
                fobj = open("inouttime.txt", "r")
                fdata_ls = fobj.readlines()
                count = 1
                for oneline in fdata_ls:
                    if oneline.startswith(s_id + ",") and
oneline.__contains__(",OUTTIME,"): # write date
                        count = 2
                        y = oneline.split(",")
                        tv = StringVar()
                        tv.set(y[2] + "  " + y[3])
                        ot = Entry(base, width=20, textvariable=tv,
font=("Arial 20 bold"))
                        ot.place(x=680, y=550)
                        break
                fobj.close()
                if count == 1:
                    tv = StringVar()
                    tv.set("Invalid Id")
                    ot = Entry(base, width=20, textvariable=tv, font=("Arial
20 bold"))
                    ot.place(x=680, y=550)
            def save2():
                sel_opt = str(r_sel.get())
                s_id = str(in_time_entry.get())
                fobj = open("inouttime.txt", "r")
                fdata_ls = fobj.readlines()
                fobj.close()
                rdate = date()
                count = 1
                fobj = open("inouttime.txt", "w")
                for oneline in fdata_ls:
                    if oneline.startswith(s_id + ",") and
oneline.__contains__(",OUTTIME,"):  # write date
                        new_oneline = oneline.replace(",OUTTIME,", "," + rdate
+ ",")
                        new_oneline2 = new_oneline.replace(",REMARK", "," +
sel_opt + ",")
                        fobj.write(new_oneline2)
                        count = 2
                    else:
                        fobj.write(oneline)
                if count == 2:
                    done = Label(base, text="Successful...!", font=("Arial 20
bold"), bg="silver", fg="black")
                    done.place(x=700, y=720)
                if count == 1:
                    done = Label(base, text="Please Give Correct Information",
font=("Arial 20 bold"), bg="silver", fg="black")
                    done.place(x=600, y=720)
                fobj.close()
            sear_ot = Button(base, text="Search Outtime", font=("Arial 18
bold"), bg="white", command=search_outtime)
            sear_ot.place(x=1050, y=520)
            save2 = Button(base, text="Save", font=("Arial 20 bold"),
bg="white", fg="black", command=save2)
            save2.place(x=1050, y=700)
            r_sel = StringVar(base)
            ls = ['Before Time', 'On Time', 'Late']
            r_sel.set(ls[0])
            remark = OptionMenu(base, r_sel, *ls)
            remark.config(width=20, font=("Arial 15 bold"), fg="black")
            remark.place(x=680, y=610)
    def visitor():
            canvas = Canvas(base, bg='silver', height=675, width=1210)
            canvas.place(x=320, y=105)
            h1 = Label(base, text="Visitor's Information", bg="dark slate 
grey", font=("Arial 20 bold"), fg="white", padx=480, pady=10)
            h1.place(x=320, y=105)
            v_name = Label(base, text="Name", font=("Arial 20 bold"), 
bg="silver", fg="black")
            v_name.place(x=500, y=200)
            v_name_entry = Entry(base, width=20, font=("Arial 20 bold"))
            v_name_entry.place(x=650, y=200)
            v_contact = Label(base, text="Contact", font=("Arial 20 bold"), 
bg="silver", fg="black")
            v_contact.place(x=500, y=300)
            v_contact_entry = Entry(base, width=20, font=("Arial 20 bold"))
            v_contact_entry.place(x=650, y=300)
            v_reason = Label(base, text="Reason", font=("Arial 20 bold"), 
bg="silver", fg="black")
            v_reason.place(x=500, y=400)
            v_reason_entry = Entry(base, width=20, font=("Arial 20 bold"))
            v_reason_entry.place(x=650, y=400)
            v_address = Label(base, text="Address", font=("Arial 20 bold"), 
bg="silver", fg="black")
            v_address.place(x=500, y=500)
            v_address_entry = Entry(base, width=20, font=("Arial 20 bold"))
            v_address_entry.place(x=650, y=500)
            st_name = Label(base, text="Student Name", font=("Arial 20 bold"), 
bg="silver", fg="black")
            st_name.place(x=1020, y=200)
            st_name_entry = Entry(base, width=15, font=("Arial 20 "))
            st_name_entry.place(x=1250, y=200)
            def search():
                def reset():
                    ot.destroy()
                    st_name_entry.delete(0, END)
                    st_name_entry.place(x=1250, y=200)
                def add_visitor():
                    vn = str(v_name_entry.get())
                    vcon = str(v_contact_entry.get())
                    vr = str(v_reason_entry.get())
                    vadd = str(v_address_entry.get())
                    sn = str(st_name_entry.get())
                    d = str(date())
                    fp = open("visitor_info.txt", "a")
                    fp.write(sn + "," + vn + "," + vcon + "," + vr + "," + 
vadd + "," + d + "," + "\n")
                    fp.close()
                    s = Label(base, text="Successfull...!", font=("Arial 20 
bold"), bg="dark slate grey", fg="white", padx=100)
                    s.place(x=1100, y=700)
                reset_btn = Button(base, text="Reset", font=("Arial 20 bold"),
command=reset)
                reset_btn.place(x=1340, y=280)
                s_n = str(st_name_entry.get())
                fobj = open("student_info.txt", "r")
                fdata_ls = fobj.readlines()
                count = 1
                for oneline in fdata_ls:
                    if oneline.__contains__("," + s_n + ","):
                        # write date
                        count = 2
                        y = oneline.split(",")
                        tv = StringVar()
                        tv.set(y[11])
                        ot_name = Label(base, text="Room No.", font=("Arial 20
bold"), bg="silver")
                        ot_name.place(x=1020, y=400)
                        ot = Entry(base, width=15, textvariable=tv,
font=("Arial 20 bold"), justify=CENTER)
                        ot.place(x=1250, y=400)
                        add_btn = Button(base, text="Add Visitor",
font=("Arial 20 bold"), command=add_visitor)
                        add_btn.place(x=1100, y=500)
                        break
                fobj.close()
                if count == 1:
                    tv = StringVar()
                    tv.set("Invalid Name")
                    ot = Entry(base, width=15, textvariable=tv, font=("Arial
20 bold"), fg="red")
                    ot.place(x=1250, y=200)
            line = Canvas(base, height=650, width=5)
            line.place(x=1000, y=162)
            ser_btn = Button(base, text="Search", font=("Arial 20 bold"),
command=search)
            ser_btn.place(x=1180, y=280)
    def view_info():
            canvas = Canvas(base, bg='silver', height=675, width=1215)
            canvas.place(x=320, y=105)
            def all_girl_info():
                canvas = Canvas(base, bg='silver', height=675, width=810)
                canvas.place(x=715, y=105)
                h1 = Label(base, text="Information Of Students", font=("Arial
20 bold"), bg="dark slate grey", fg="white",padx=270, pady=5)
                h1.place(x=720, y=105)
                h = "Room NO.     Name     Contact     Workplace"
                h2 = Label(base, text=h, font=("Arial 20 bold"), bg="dark
orange", fg="white")
```

```
                h2.place(x=723, y=150)
                students = []
                f1 = open("student_info.txt", "r")
                all_lines = f1.readlines()
                for i in all_lines:
                    temp = str(i)
                    one_line = temp.split(',')
                    t = []
                    t.append(one_line[11])
                    t.append(one_line[1])
                    t.append(one_line[0])
                    t.append(one_line[6])
                    students.append(t)
                x1co = 800
                y1co = 200
                x2co = 900
                x3co = 1160
                x4co = 1380
                flag = 1
                for i in students:
                    y = i
                    c = 0
                    while c <= 2:
                        label = Label(base, text=y[0], font=("Arial 15 bold"),
bg="silver", fg="black")
                        label.place(x=x1co, y=y1co)
                        label2 = Label(base, text=y[1], font=("Arial 15
bold"), bg="silver", fg='black')
                        label2.place(x=x2co, y=y1co)
                        label3 = Label(base, text=y[2], font=('Arial 15
bold'), bg="silver", fg='black')
                        label3.place(x=x3co, y=y1co)
                        label4 = Label(base, text=y[3], font=("Arial 15
bold"), bg="silver", fg="black")
                        label4.place(x=x4co, y=y1co)
                        c = c + 1
                    y1co = y1co + 50
                    if flag > 8:
                        break
            def room_wise():
                canvas = Canvas(base, bg='silver', height=675, width=810)
                canvas.place(x=715, y=105)
                l1 = Label(base, text="Enter Room No.", font=("Arial 20
bold"), bg="silver", fg="black")
                l1.place(x=750, y=150)
                l1_entry = Entry(base, width=10, font=("Arial 20 bold"))
                l1_entry.place(x=1000, y=150)
                l1_entry.focus()
                def search():
                    r_no = str(l1_entry.get())
                    f1 = open("student_info.txt", "r")
                    all_lines = f1.readlines()
                    r_info = []
                    for i in all_lines:
                        temp = str(i)
                        one_line = temp.split(',')
                        if one_line[11] == r_no:
                            temp = []
                            temp.append(one_line[10])
                            temp.append(one_line[1])
                            temp.append(one_line[0])
                            temp.append(one_line[6])
                            r_info.append(temp)
                    h1 = Label(base, text="Information Of Students",
font=("Arial 20 bold"), bg="dark slate grey",fg="white", padx=270, pady=5)
                    h1.place(x=720, y=250)
                    h = "Bed No. Name Contact Workplace"
                    h2 = Label(base, text=h, font=("Arial 20 bold"), bg="dark
orange", fg="white")
                    h2.place(x=723, y=300)
                    x1co = 780
                    y1co = 400
                    x2co = 900
                    x3co = 1160
                    x4co = 1380
                    flag = 1
                    for i in r_info:
                        y = i
                        c = 0
                        while c <= 2:
                            label = Label(base, text=y[0], font=("Arial 15
bold"), bg="silver", fg="black")
                            label.place(x=x1co, y=y1co)
                            label2 = Label(base, text=y[1], font=("Arial 15
bold"), bg="silver", fg='black')
                            label2.place(x=x2co, y=y1co)
                            label3 = Label(base, text=y[2], font=('Arial 15
bold'), bg="silver", fg='black')
                            label3.place(x=x3co, y=y1co)
                            label4 = Label(base, text=y[3], font=("Arial 15
bold"), bg="silver", fg="black")
                            label4.place(x=x4co, y=y1co)
                            c = c + 1
                        y1co = y1co + 100
                        if flag > 2:
                            break
                search_btn = Button(base, text="Search", font=("Arial 20
bold"), command=search)
                search_btn.place(x=1200, y=140)
            stud_info = Button(base, text="Information Of All Students",
wraplength=180, justify=CENTER, font=("Arial 20 bold"), padx=50, pady=1,
command=all_girl_info)
            stud_info.place(x=400, y=200)
            stud_room_wise = Button(base, text="Room Info", font=("Arial 20
bold"), padx=60, pady=5, command=room_wise)
            stud_room_wise.place(x=400, y=400)
            line = Canvas(base, height=680, width=5)
            line.place(x=710, y=105)
    def leave_application():
        canvas = Canvas(base, bg='silver', height=675, width=1210)
        canvas.place(x=320, y=105)
        h1 = Label(base, text="Leave Application", bg="dark slate grey",
font=("Arial 20 bold"), fg="white", padx=485, pady=10)
        h1.place(x=320, y=105)
        h_id = Label(base, text="Hostel ID", bg="silver", font=("Arial 20
bold"), fg="black")
        h_id.place(x=400, y=200)
        id_entry = Entry(base, width=15, font=("Arial 20"))
        id_entry.place(x=550, y=200)
        n_label = Label(base, text="Name", bg="silver", font=("Arial 20
bold"), fg="black")
        n_label.place(x=800, y=200)
        n_entry = Entry(base, width=20, font=("Arial 20"))
        n_entry.place(x=900, y=200)
        r_label = Label(base, text="Room No.", bg="silver", font=("Arial 20
bold"), fg="black")
        r_label.place(x=400, y=300)
        r_entry = Entry(base, width=15, font=("Arial 20"))
        r_entry.place(x=550, y=300)
        m_no = Label(base, text="Mobile No.", bg="silver", font=("Arial 20
bold"), fg="black")
        m_no.place(x=800, y=300)
        m_entry = Entry(base, width=15, font=("Arial 20"))
        m_entry.place(x=950, y=300)
        reason = Label(base, text="Reason For Leave", bg="silver",
font=("Arial 20 bold"), fg="black")
        reason.place(x=400, y=400)
        reason_entry = Entry(base, width=20, font=("Arial 20"))
        reason_entry.place(x=700, y=400)
        return_date = Label(base, text="Return Date", bg="silver",
font=("Arial 20 bold"), fg="black")
        return_date.place(x=400, y=500)
        x = Label(base, text="(YEAR-DD-MM)", font=("Arial 15 bold"),
bg="silver", fg="black")
```

```
        x.place(x=1000, y=500)
        date_entry = Entry(base, width=18, font=("Arial 20"))
        date_entry.place(x=700, y=500)
        def leave_info():
            i = str(id_entry.get())
            n = str(n_entry.get())
            r = str(r_entry.get())
            m = str(m_entry.get())
            reas = str(reason_entry.get())
            rdate = str(date_entry.get())
            current_date = date()
            fopen = open("leave_applications.txt", "a")
            fopen.write(i + "," + n + "," + r + "," + m + "," + reas + "," +
current_date + "," + rdate + "," + "\n")
            fopen.close()
            success = Label(base, text="Submit Successfully..!", font=("Arial
20 bold"), fg="green", bg="white")
            success.place(x=700, y=700)
        submit = Button(base, text="Submit", font=("Arial 20 bold"),
fg="black", command=leave_info)
        submit.place(x=700, y=600)

    add_std_btn = Button(base, text="Add Student", font=("Arial 20 bold"),
padx=15, bg="white", command=add_stud)
    add_std_btn.place(x=50, y=150)
    add_new_room = Button(base, text="Add New Room", font=("Arial 20 bold"),
bg="white", command=add_room)
    add_new_room.place(x=50, y=250)
    in_ot_time = Button(base, text="In And Outtime", font=("Arial 20 bold"),
bg="white", command=in_out_time)
    in_ot_time.place(x=50, y=350)
    visitor = Button(base, text="Visitor", font=("Arial 20 bold"), bg="white",
padx=55, command=visitor)
    visitor.place(x=50, y=450)
    view_info = Button(base, text="View Information", font=("Arial 18 bold"),
bg="white", command=view_info)
    view_info.place(x=50, y=535)
    leave_application = Button(base, text="Leave Application", font=("Arial 18
bold"), bg="white", command=leave_application)
    leave_application.place(x=50, y=620)
    exit = Button(base, text="EXIT", font=("Arial 18 bold"), bg="white",
padx=70, command=quit)
    exit.place(x=50, y=700)
username = Label(base, text="Username", font=("Arial 25 bold"), bg="silver",
fg="black")
username.place(x=470, y=240)
user_entry = Entry(base, width=17, font=("Arial 20"))
user_entry.place(x=650, y=245)
user_entry.focus()
password = Label(base, text="Password", font=("Arial 25 bold"), bg="silver",
fg="black")
password.place(x=470, y=350)
pass_entry = Entry(base, width=17, font="Arial 20")
pass_entry.place(x=650, y=355)
def login():
    id = str(user_entry.get())
    key = str(pass_entry.get())
    if id == "admin" and key == "1234":
        main()
    else:
        user_entry.focus()
        user_entry.delete(0, END)
        pass_entry.delete(0, END)
        fail = Label(base, text="Wrong Username Or Password...!", font=("Arial
30 bold"),bg="silver", fg="red")
        fail.place(x=450, y=540)
login_btn = Button(base, text="Login", font=("Arial 25 bold"),
bg="lightseagreen", fg="white",command=login)
login_btn.place(x=650, y=440)
base.mainloop()


file_name = ["inouttime.txt", "leave_applications.txt", "room_info_boys.txt",
"room_info_girls.txt",
             "student_info.txt", "room_info_others.txt", "visitor_info.txt"]

for x in file_name:
    fp = open(x,"x")
    fp.close()
```

SCREENSHOTS AND RESULTS

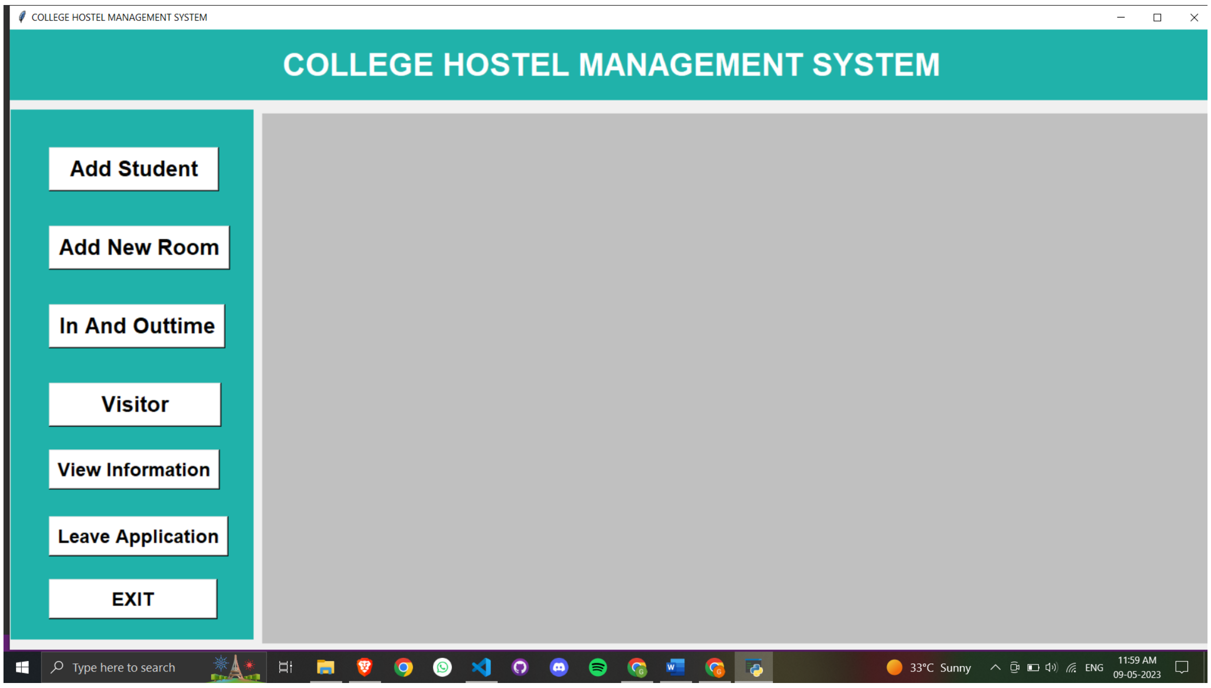
Login Page


COLLEGE HOSTEL MANAGEMENT SYSTEM
COLLEGE HOSTEL MANAGEMENT SYSTEM
Login
Type here to search
33°C Sunny
ENG
11:58 AM
09-05-2023

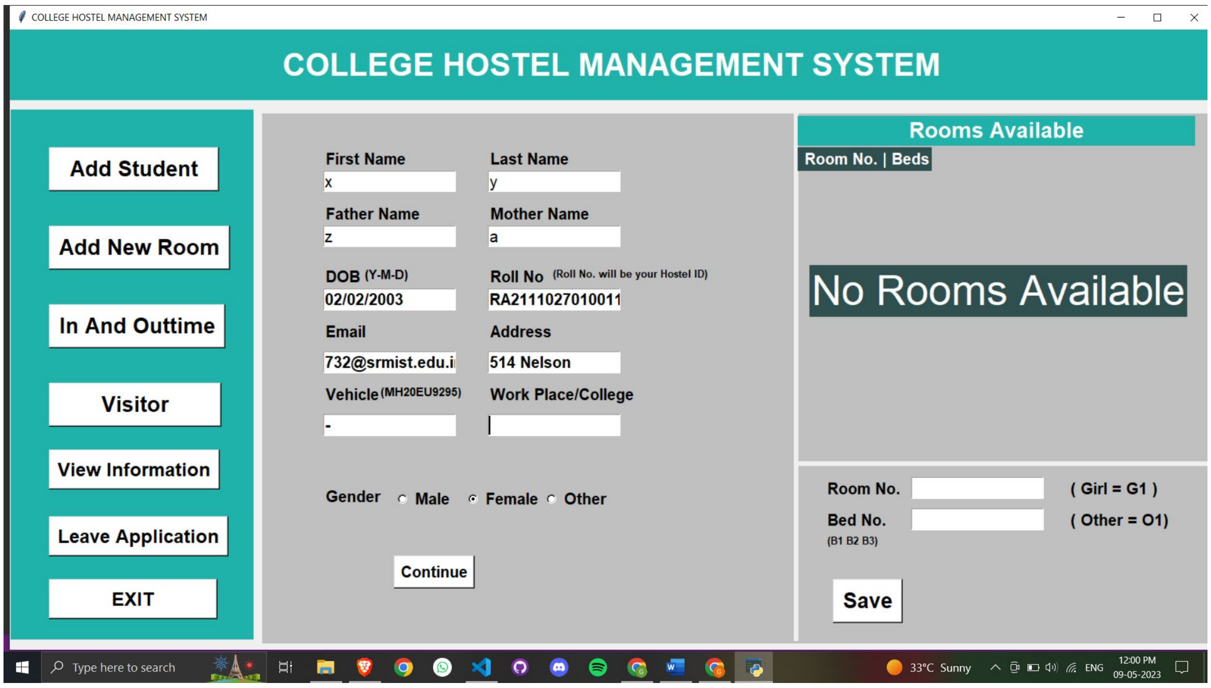
Username admin

Password 1234

Home Page



Student Registration Page


COLLEGE HOSTEL MANAGEMENT SYSTEM

COLLEGE HOSTEL MANAGEMENT SYSTEM

First Name: x	Last Name: y
Father Name: z	Mother Name: a
DOB (Y-M-D): 02/02/2003	Roll No (Roll No. will be your Hostel ID): RA2111027010011
Email: 732@srmist.edu.i	Address: 514 Nelson
Vehicle (MH20EU9295): -	Work Place/College:

Gender Male [✓ Female] Other

Continue

Rooms Available

Room No. | Beds

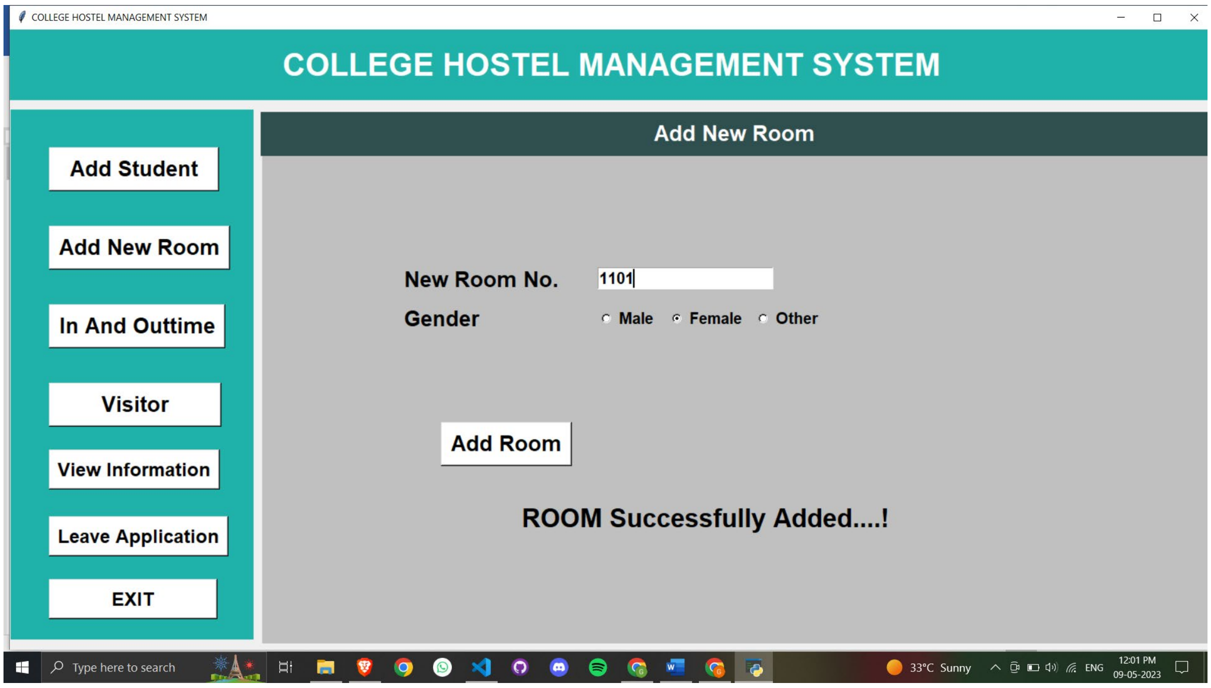
No Rooms Available

Room No. (Girl = G1)

Bed No. (B1 B2 B3) (Other = O1)

Save

Add New Room Page


COLLEGE HOSTEL MANAGEMENT SYSTEM

COLLEGE HOSTEL MANAGEMENT SYSTEM

Add New Room

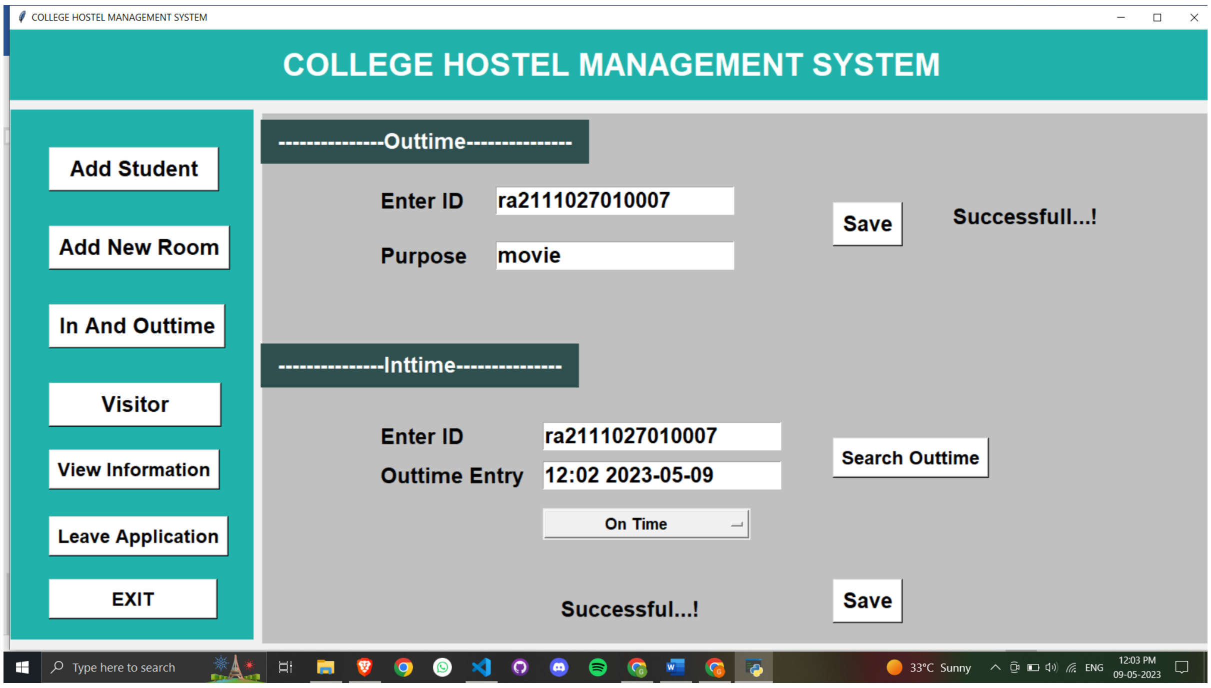
New Room No. 1101

Gender Male [✓ Female] Other

Add Room

ROOM Successfully Added....!

Intime – Out time Page


COLLEGE HOSTEL MANAGEMENT SYSTEM

---------------Outtime---------------

Enter ID: ra2111027010007

Purpose: movie

Save — Successfull...!

---------------Inttime---------------

Enter ID: ra2111027010007

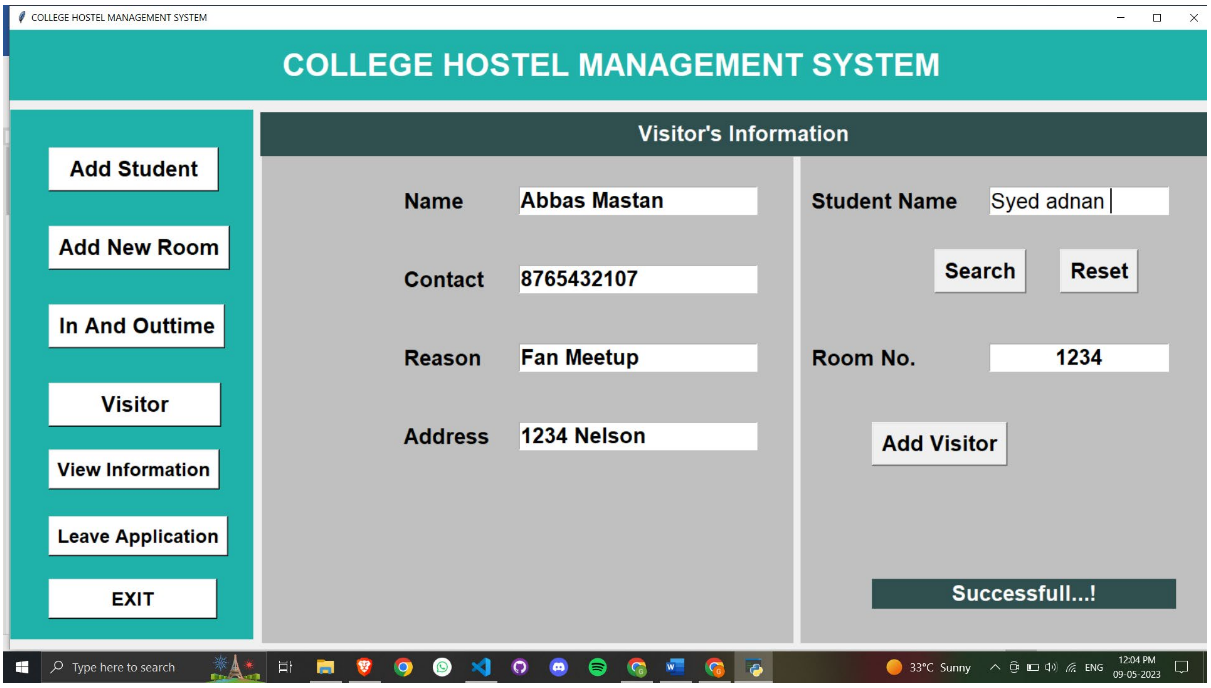
Outtime Entry: 12:02 2023-05-09

On Time

Search Outtime

Successful...! — Save

Visitor Page


COLLEGE HOSTEL MANAGEMENT SYSTEM

Visitor's Information

Name: Abbas Mastan

Contact: 8765432107

Reason: Fan Meetup

Address: 1234 Nelson

Student Name: Syed adnan

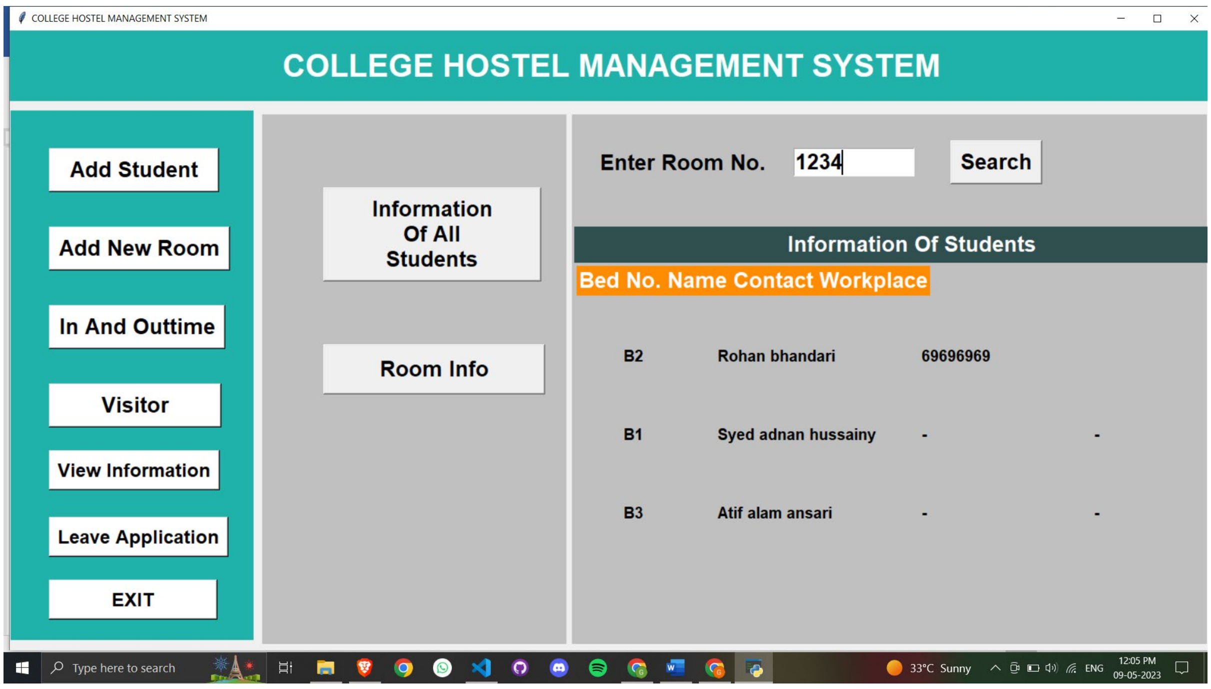
Search — Reset

Room No.: 1234

Add Visitor

Successfull...!

Information of Students and Room

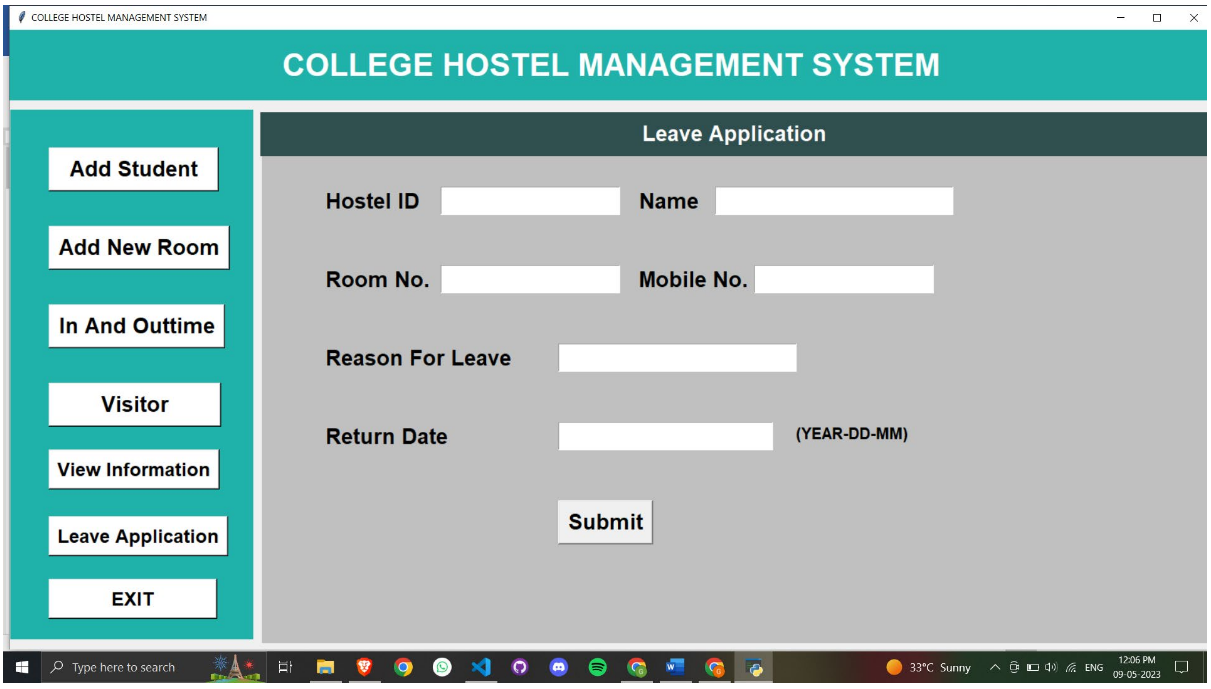

COLLEGE HOSTEL MANAGEMENT SYSTEM

Enter Room No. 1234

Information Of Students

Bed No.	Name	Contact	Workplace
B2	Rohan bhandari	69696969	
B1	Syed adnan hussainy	-	-
B3	Atif alam ansari	-	-

Leave Application Page


COLLEGE HOSTEL MANAGEMENT SYSTEM

Leave Application

Hostel ID
Name
Room No.
Mobile No.
Reason For Leave
Return Date (YEAR-DD-MM)

CONCLUSION

In conclusion, the University Hostel Management System implemented using Python, Tkinter, and a database provides an efficient and reliable solution for managing hostel accommodation in universities. The system automates the process of room allocation, billing, and management of student records, saving time and reducing errors associated with manual management.

The system is designed using a client-server architecture with a desktop-based client application and a database server. The system is implemented using a Model-View-Controller (MVC) architecture, which separates the system into three components: the model, view, and controller.

The development of the system follows a structured methodology that includes requirements gathering, system analysis and design, implementation, integration and testing, and deployment and maintenance.

The University Hostel Management System is scalable and extensible, making it customizable to meet the specific needs of different universities. The system architecture and design provide a framework for future system enhancements and improvements.

Overall, the University Hostel Management System is a valuable tool for managing hostel accommodation in universities, improving the efficiency and accuracy of the management process while providing a user-friendly interface for hostel administrators and students.

FUTURE ENHANCEMENTS

The University Hostel Management System implemented using Python, Tkinter, and a database can be enhanced further to meet the evolving needs of hostel management in universities. Some potential areas for future enhancements include:

1) **Mobile Application:** The system can be extended to include a mobile application that will allow students to check their room allocations, view their bills, and submit requests for maintenance or other services. This will provide greater convenience for students and reduce the workload on hostel administrators.

2) **Gateway Integration:** The system can be integrated with popular payment gateways to enable students to pay their hostel bills online. This will improve the efficiency of the billing process and reduce the workload on hostel administrators.

3) **Integration with Access Control Systems:** The system can be integrated with access control systems to enable hostel administrators to manage access to different parts of the hostel. This will improve security and enable better monitoring of student movements.

4) **Reporting and Analytics:** The system can be enhanced to include reporting and analytics capabilities, allowing hostel administrators to generate reports on occupancy rates, revenue, and other metrics. This will enable better decision-making and planning.

5) **Integration with Smart Devices:** The system can be integrated with smart devices such as sensors and cameras to enable better monitoring of hostel infrastructure and student behavior. This will provide greater insight into hostel operations and enable more efficient management.

REFERENCES

- "Tkinter GUI Application Development Projects" by Bhaskar Chaudhary
- "Python for Everybody: Exploring Data in Python 3" by Charles Severance
- "Database Systems: Design, Implementation, and Management" by Carlos Coronel and Steven Morris
- "Software Engineering: A Practitioner's Approach" by Roger Pressman and Bruce Maxim
- "Modern Systems Analysis and Design" by Jeffrey A. Hoffer, Joey F. George, and Joseph S. Valacich
- "Managing Hostel Accommodation in Universities Using an Integrated Information System: A Case Study of Covenant University, Ota, Nigeria" by Ayodeji A. Oluwatobi and Jonathan A. Oke
- "Development of a Hostel Management System using PHP and MySQL" by V. P. Saravanan and R. Saravanakumar
- "Online Hostel Management System" by T. Ch. Pradeep Kumar and K. Naveen Kumar