SHRI VILEPARLE KELAVANI MANDAL'S

SHRI BHAGUBHAI MAFATLAL POLYTECHNIC

Name: Yash Avsarmal & Shubham Parmar

Course Name: Programming in Python

Course code: PRP198918

Program: Information Technology

Semester: Fourth/ IV

Roll no.: T020 & T004

ACKNOWLEDGEMENT

I would like to express my deepest gratitude and appreciation to all individuals who have made invaluable contributions to the successful completion of this mini python project.

First and foremost, I want to extend my sincere thanks to our esteemed faculty members. Their unwavering guidance, support, and expertise have been instrumental in shaping our understanding of Python programming and its practical applications. Their continuous encouragement and valuable feedback have played a pivotal role in our project's development.

I am immensely grateful to my project supervisor, whose exceptional guidance, patience, and unwavering support have been invaluable. Their insightful suggestions and constructive criticism have significantly improved the quality of our project. I am truly indebted to them for their generous time and effort spent in reviewing and refining our work.

Additionally, I would like to express my heartfelt appreciation to my fellow classmates and project team members. Their dedication, collaboration, and collective effort have made this project an enriching experience. The vibrant brainstorming sessions, constant exchange of ideas, and collaborative problem-solving have all contributed significantly to the project's success.

Together, with the guidance of our faculty, the support of our supervisor, and the collaborative efforts of my team members, we have achieved a successful outcome. I am immensely grateful for the opportunity to work with such talented individuals, whose contributions have made this project truly memorable and impactful.

PROBLEM STATEMENT

The objective of this project is to develop a hospital management system that will facilitate effective management of various aspects of the hospital's operations. Using Tkinter, the system will provide a user-friendly graphical user interface (GUI) and integrate with a MySQL database to store and retrieve relevant information.

The system should address the following requirements:

User Authentication: Implement a secure login mechanism to authenticate users, including doctors, nurses, administrative staff, and other personnel. Different levels of access should be granted based on user roles to ensure appropriate data privacy and security.

Patient Management: Enable users to manage patient records, including registration, appointment scheduling, medical history, and treatment details. The system should support functionalities such as adding new patients, updating their information, and retrieving patient data when required.

Ward Management: Facilitate the allocation and management of hospital wards and beds. Users should be able to assign patients to specific wards, track bed availability, and manage patient transfers between wards.

Staff Management: Assist in managing hospital personnel, including doctors, nurses, and support staff. The system should enable users to track employee information, such as work schedules, shifts, leave management, and performance evaluations.

Billing and Payment: Provide functionality for generating and managing patient bills, tracking payments, and generating financial reports. The system should be capable of integrating with billing systems and insurance providers to streamline the billing process.

Database Connectivity: Establish a connection to a MySQL database to store and retrieve critical information, including patient records, inventory data, staff details, billing information, and other relevant data necessary for efficient hospital management.

By addressing these needs, the hospital management system will streamline administrative tasks, improve patient care, and enhance overall operational efficiency within the hospital environment.

FEATURES

LITERATURE SURVEY (NEW TECHNOLOGIES/TOOLS LEARNED)

In the development of a Hospital Management System, exploring various technologies and tools can significantly enhance the system's functionality. Here are some technologies and tools that can be beneficial:

1) GUI Frameworks: Consider using popular GUI frameworks such as Tkinter to create a user-friendly interface for the hospital management system. These frameworks provide a wide range of graphical components and functionality for building interactive interfaces.

2) Database Management System: Utilize a reliable database management system like MySQL, PostgreSQL, or MongoDB to store and manage the hospital's data efficiently. These systems offer robust features for data storage, retrieval, and management, ensuring data integrity and security.

3) Python Modules:

a) File Module: The file module enables file manipulation operations, including reading, writing, and managing files. It can be used for tasks such as storing patient records, generating reports, and managing document files.

b) MySQL Module: The MySQL module allows seamless integration with a MySQL database. It provides functionalities to establish a connection to the database, execute SQL queries, fetch results, and manage transactions. By using the MySQL module in Python, you can easily store and retrieve data from the MySQL database, making it an essential tool for implementing the hospital management system's database connectivity and data management features.

4) Functions: Utilize functions in Python to modularize code and improve code reusability. Functions can be created for specific tasks such as calculating bills, validating input, or generating reports.

Modularizing code using functions enhances code organization and readability.

5) Object-Oriented Programming (OOP): Implement OOP principles in Python, utilizing classes, objects, and interfaces. OOP allows for better organization of code, encapsulation of data and behavior, and facilitates the creation of reusable and maintainable code.

By incorporating these technologies, tools, and programming concepts into the development of the Hospital Management System, you can enhance its functionality, maintainability, and scalability.

SRS FOR HOSPITAL MANAGEMENT SYSTEM

TABLE OF CONTENT

1. INTRODUCTION

1.1 PURPOSE

The purpose of this document is to go the hospital management system and management of the hospitals.

1.2 DOCUMENT CONVENTIONS

This document uses the following conventions:

DDB Distributed Database

ER Entity Relation

1.3 INTENDED AUDIENCE AND READING SUGGESTIONS

This project is a prototype for the Hospital Management System. It has been developed under the guidance of a college professor to showcase the potential benefits and functionalities of an efficient hospital management system. This project is designed to cater to the needs of hospital owners and administrators who are responsible for managing hospitals and healthcare facilities.

1.4 PROJECT SCOPE

The scope of this project is very high as it reduces the paperwork by a huge amount. As well as it is a user-friendly system. The system is based on a relational database. We will have a database server supporting hundreds of users around the world as well as their activity on this system. Above all, we hope to provide a comfortable user experience along with the best pricing available.

2. OVERALL DESCRIPTION

2.1 PRODUCT PERSPECTIVE

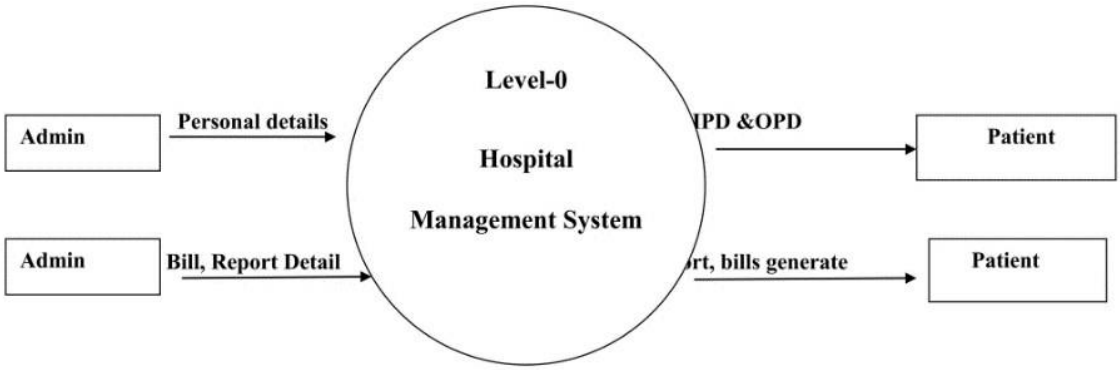
A distributed Hospital database stores the following information:

1. Patient Details
2. Medical Records
3. Appointment Details
4. Staff Details
5. Billing and Payment
6. Ward and Bed Management
7. Employee Payroll 8. Reports and Analytics

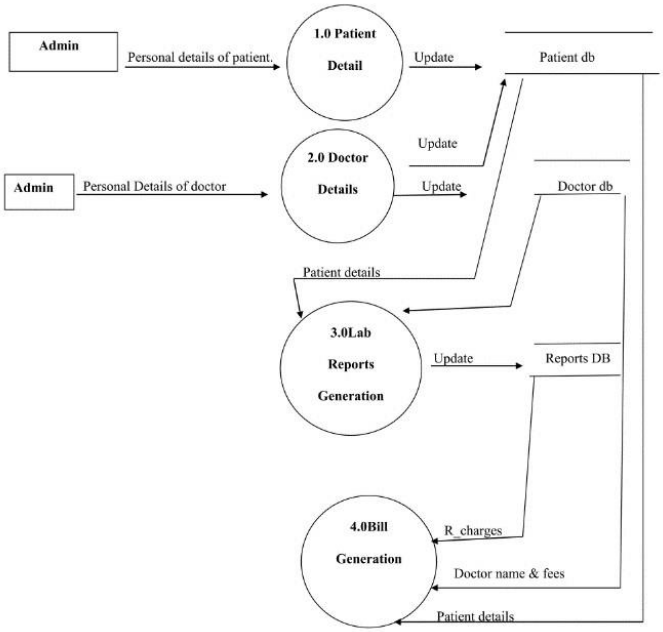
2.2 SYSTEM DESIGN

Data Flow Diagrams (DFD):

Level 0:

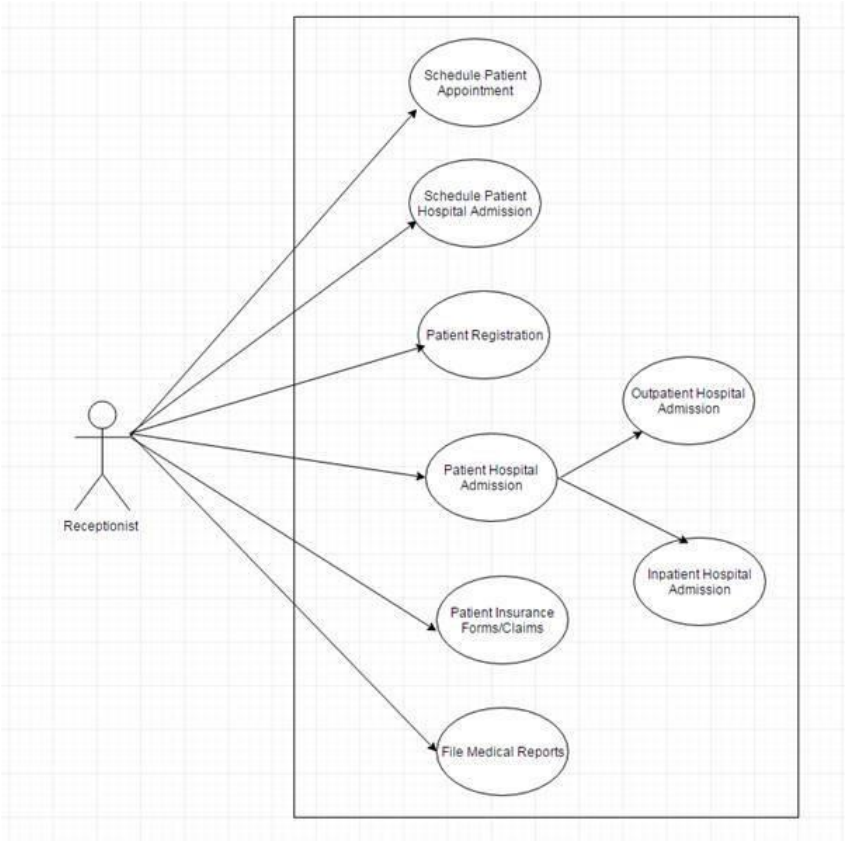


Level 1:

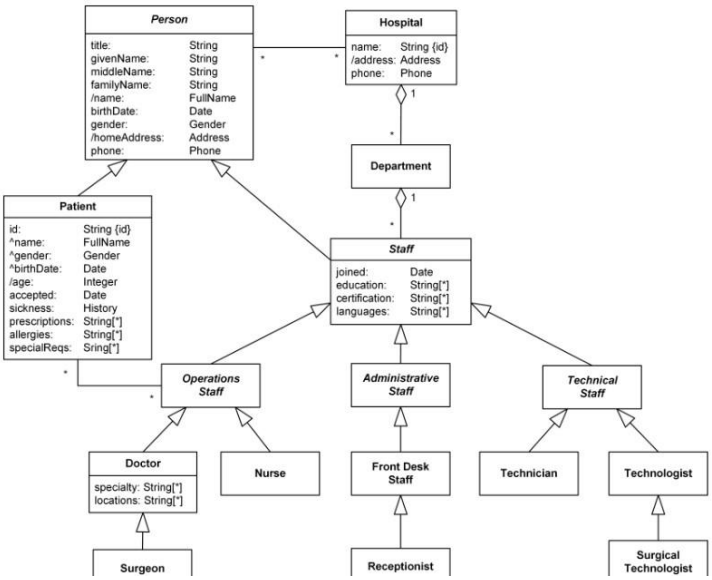


UML Diagrams:

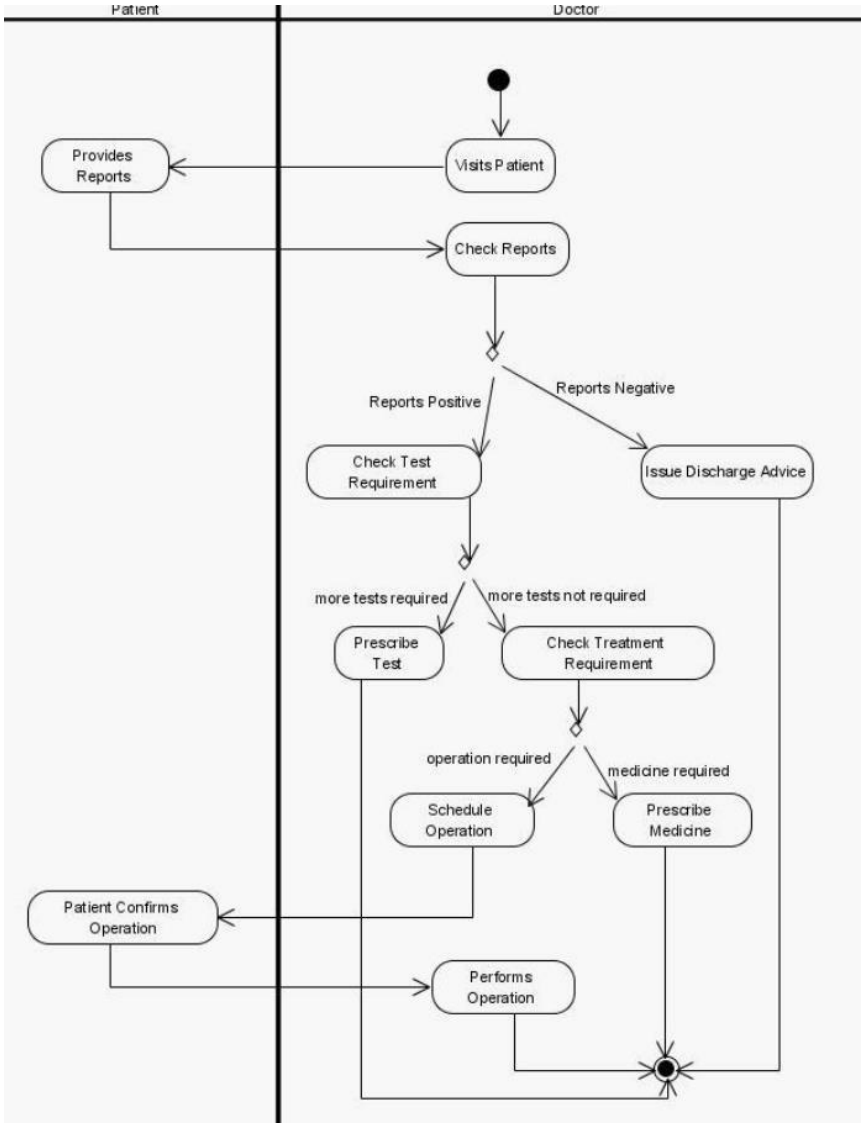
Use Case:



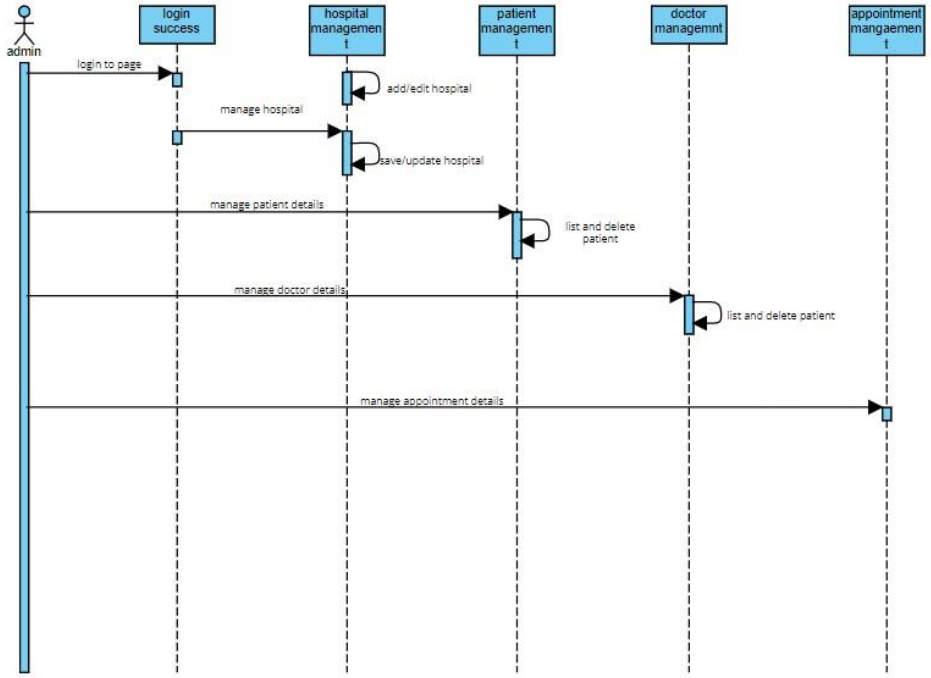
Class Diagram:



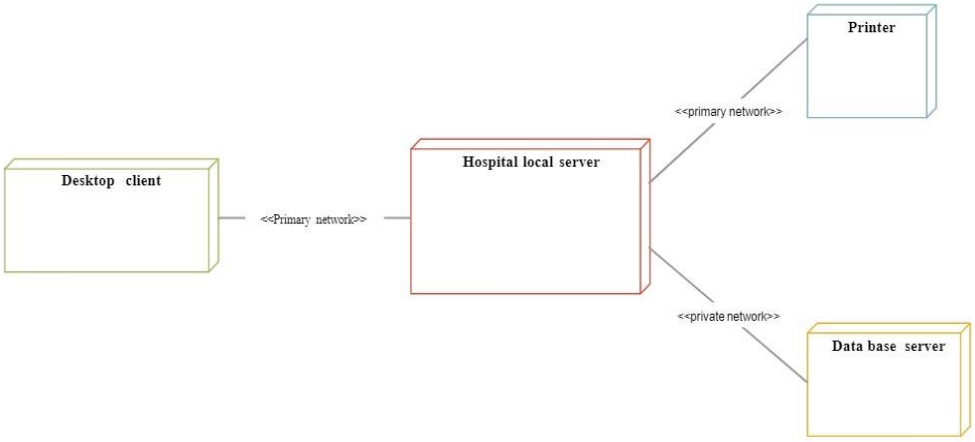
Activity Diagram:



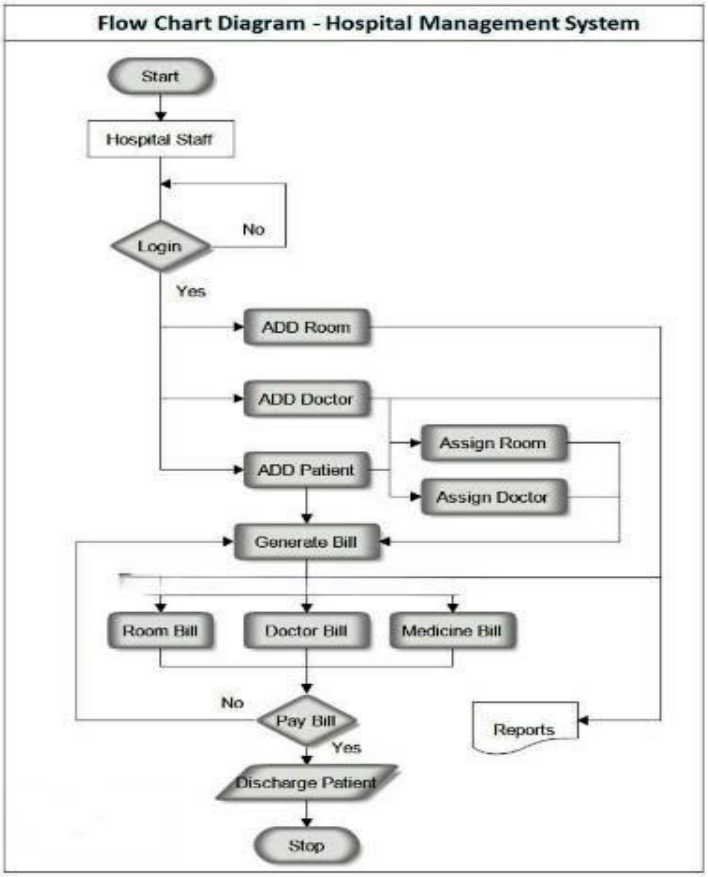
Sequence Diagram:



Deployment Diagram:



Flow Chart:



2.3 USER CLASS AND CHARACTERISTICS

As the project is desktop-based so the main role is of admin instead of a customer.

Here the user is admin.

The admin has the following features:

- Managing User Accounts
- Monitoring the system
- Performing Maintenance Task
- Responding to Security Issues

2.4 OPERATING ENVIRONMENT

The operating environment of Hospital Management is listed below:

- MYSQL Database
- OS: Windows 11

- Platform: Sublime Text and Xampp

2.5 DESIGN AND IMPLEMENTATION CONSTRAINTS

How the response for applications 1 and 2 will be generated. Assuming these are global required

2.6 ASSUMPTION DEPENDENCIES

Let us assume that this is a Hospital, here the admin part will remain the same, but the customer can also buy a food at any time using this project.

3.SYSTEM FEATURES

3.1 DESCRIPTION AND PRIORITY

The Hospital Management System plays a crucial role in streamlining and enhancing the efficiency of healthcare facilities. This project holds an even higher priority as it significantly reduces paperwork and eliminates the fear of losing critical information or data.

The Hospital Management System automates various administrative and operational tasks within a hospital, such as patient registration, appointment scheduling, billing and invoicing, inventory management, staff management, and medical record-keeping. By digitizing and centralizing these processes, the system minimizes the reliance on manual paperwork, ensuring a more streamlined and accurate workflow.

One of the primary advantages of the Hospital Management System is the reduction of paperwork. By eliminating the need for physical forms, registers, and documents, healthcare providers can save time, resources, and storage space. Additionally, the system enables easy retrieval and access to patient information, medical records, and diagnostic reports, facilitating faster decision-making and improving patient care.

3.2 STIMULUS/RESPONSE SEQUENCES

- Enter User details
- Shows report of User
- Bill Printed

4. EXTERNAL INTERFACE REQUIREMENTS

4.1 USER INTERFACE

- Front-end software: Sublime Text

4.2 BACKEND INTERFACE

- Back-end software: Xampp

4.3 HARDWARE INTERFACE

- Windows with python 3.11 already installed
- Internet connection

4.4 SOFTWARE INTERFACE

Following are the software used for the Hospital Management System

Software Used	Description
Operating system	We have chosen Windows operating system for its best support and user-friendliness.
Python	To implement the project we have chosen Python language for its more interactive support.

4.5 COMMUNICATION INTERFACES

This project is supported in windows machine with the Sublime Text module. We are using simple electronic forms for Hospital Management System.

5. NONFUNCTIONAL REQUIREMENTS

5.1 PERFORMANCE REQUIREMENTS

The steps involved to perform the implementation of the Hospital Management system database are as follows:

5.1.1 ER-DIAGRAM

The E-R Diagram constitutes a technique for representing the logical structure of a database pictorially. This analysis is then used to organize data as a relation, normalizing relation and finally obtaining a relation database.

- ENTITIES: Specifies distinct real-world items in an application.
- PROPERTIES/ATTRIBUTES: Specifies properties of an entity and relationships.
- RELATIONSHIPS: Connects entities and represents meaningful dependencies between them.

5.1.2 NORMALIZATION

The basic objective of normalization is to reduce redundancy which means that information is to be stored only once. Storing information several times leads to wastage of storage space and an increase in the total size of the data stored.

If a database is not properly designed it can give rise to modification anomalies. Modification anomalies arise when data is added to, changed, or deleted from a database table. Similarly, in traditional databases as well as improperly designed relational databases, data redundancy can be a problem. These can be eliminated by normalizing a database.

Normalization is the process of breaking down a table into smaller tables. So that each table deals with a single theme. There are three different kinds of modifications of anomalies and formulated the first, second, and third normal forms (3NF) is considered sufficient for most practical purposes. It should be considered only after a thorough analysis and a complete understanding of its implications.

5.2 SAFETY REQUIREMENTS

If there is extensive damage to a wide portion of the database due to catastrophic failure, such as a disk crash, the recovery method restores a past copy of the database that was backed up to archival storage (typically tape) and reconstructs a more current state by reapplying or redoing the operations of committed transactions from the backed-up log, up to the time of failure.

5.3 SECURITY REQUIREMENTS

Security systems need database storage just like many other applications. However, the special requirements of the security market mean that vendors must choose their database partner carefully.

5.4 SOFTWARE QUALITY ATTRIBUTES

- CORRECTNESS: All the expense and the income amount and date should be registered correctly into the database.
- MAINTAINABILITY: The database should be regularly backup through several RAID levels for recovery from failure.
- USABILITY: The database should be large enough to store the data of all of its customers who want to use the app

GANTT CHART (TIME CHART)

Hospital Management System:
Task 1: Feasibility Study
Start Date: [1/24/2023]
End Date: [2/11/2023]
Duration: [18 Days]

Task 2: Requirement Analysis
Start Date: [2/12/2023]
End Date: [2/21/2023]
Duration: [10 Days]

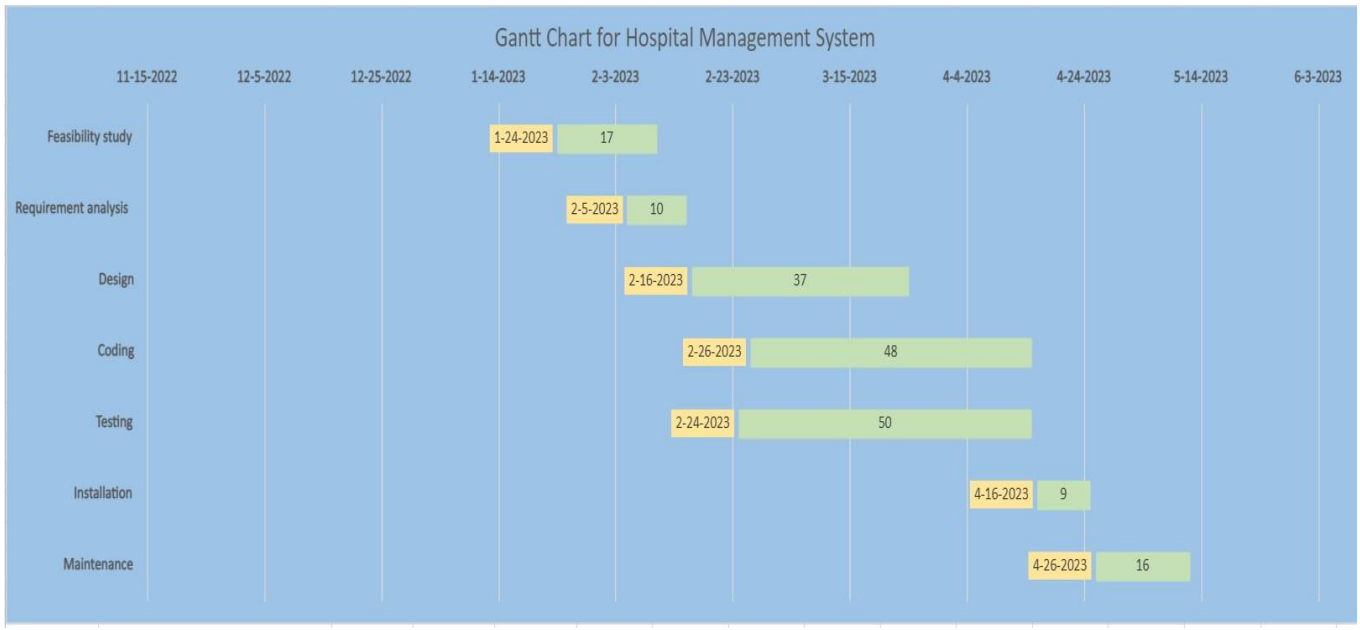
Task 3: GUI Design
Start Date: [2/22/2023]
End Date: [3/2/2023]
Duration: [10 Days]

Task 4: Coding (Functionality Implementation)
Start Date: [3/3/2023]
End Date: [3/18/2023]
Duration: [15 Days]

Task 5: Integration and Testing
Start Date: [3/18/2023]
End Date: [4/25/2023]
Duration: [38 Days]

Task 6: Installations
Start Date: [4/26/2023]
End Date: [5/4/2023]
Duration: [9 Days]

Task 7: Maintainace
Start Date: [5/4/2023]
End Date: [5/12/2023] Duration:
[8 Days]


Gantt Chart for Hospital Management System
11-15-2022
12-5-2022
12-25-2022
1-14-2023
2-3-2023
2-23-2023
3-15-2023
4-4-2023
4-24-2023
5-14-2023
6-3-2023
Feasibility study
1-24-2023
17
Requirement analysis
2-5-2023
10
Design
2-16-2023
37
Coding
2-26-2023
48
Testing
2-24-2023
50
Installation
4-16-2023
9
Maintenance
4-26-2023
16

MODULE DESCRIPTION WITH IMPLEMENTATION DATE (DELIVERABLES SCHEDULE)

1. Module: User Authentication and Access Control
- Description: Implement user login and access control features to secure the system.
- Implementation Date: [Date]

2. Module: Menu
- Description: Develop features for adding, editing, and deleting menu items. Include options for categorization, pricing, and displaying menu items.
- Implementation Date: [Date]

3. Module: Order Management
- Description: Implement order placement, tracking, and management functionalities. Include features like adding items to the order, modifying quantities, and calculating the total price.
- Implementation Date: [Date]

4. Module: Table Management
- Description: Develop features for managing Hospital tables, including table reservation, availability status, and table allocation to customers.
- Implementation Date: [Date]

5. Module: Inventory Management
- Description: Implement functionalities to track inventory items, their quantities, and automatic stock update upon order placement or item consumption.
- Implementation Date: [Date]

6. Module: Database Connectivity
- Description: Establish database connectivity to store and retrieve data related to menu items, orders, tables, and inventory.
- Implementation Date: [Date]

SYSTEM TESTING

login.py

```
from tkinter import * import
tkinter.messagebox
from menu import Menu

class MainWindow:     def
__init__(self, master):
        self.master = master
        self.master.title("CARE & CURE NURSING")
self.master.geometry("1600x800+0+0")
self.master.config(bg="cadet blue")        self.frame =
Frame(self.master,bg="cadet blue")
self.frame.pack()

        self.Username = StringVar()
self.Password = StringVar()

        self.create_widgets()

    def create_widgets(self):
        lbl_title = Label(self.frame, text="CARE & CURE NURSING", font="Helvetica 20 bold", bg="powder
blue", fg="black")
        lbl_title.grid(row=0, column=0, columnspan=2, pady=40)

        login_frame1 = Frame(self.frame, width=400, height=80, relief="ridge", bg="cadet blue", bd=20)
login_frame1.grid(row=1, column=0)
        login_frame2 = Frame(self.frame, width=400, height=80, relief="ridge", bg="cadet blue", bd=20)
login_frame2.grid(row=2, column=0)

        lbl_username = Label(login_frame1, text="Username", font="Helvetica 14 bold", bg="cadet blue",
bd=22)
        lbl_username.grid(row=0, column=0)
        entry_username = Entry(login_frame1, font="Helvetica 14 bold", textvariable=self.Username, bd=2)
entry_username.grid(row=0, column=1)

        lbl_password = Label(login_frame1, text="Password", font="Helvetica 14 bold", bg="cadet blue",
bd=22)
        lbl_password.grid(row=1, column=0)
        entry_password = Entry(login_frame1, font="Helvetica 14 bold", show="*", textvariable=self.Password,
bd=2)
        entry_password.grid(row=1, column=1)

        btn_login = Button(login_frame2, text="Login", font="Helvetica 10 bold", width=10, bg="powder
blue", command=self.login_system)        btn_login.grid(row=3, column=0)
        btn_exit = Button(login_frame2, text="Exit", font="Helvetica 10 bold", width=10, bg="powder blue",
command=self.exit)
        btn_exit.grid(row=3, column=1)

    def login_system(self):        username =
self.Username.get()        password =
```

```
self.Password.get()        if username == 'admin' and
password == '1234':
        self.open_menu()        elif username ==
'root' and password == '4321':
        self.open_menu()
else:
        tkinter.messagebox.askretrycancel("CARE & CURE NURSING", "PLEASE ENTER VALID
USERNAME AND PASSWORD")

    def open_menu(self):
        self.new_window = Toplevel(self.master)
self.app = Menu(self.new_window)
        #self.withdraw()

    def exit(self):
self.master.destroy()

def main():
root = Tk()
    app = MainWindow(root)
    root.mainloop()

if __name__ == "__main__":
    main()
```

Database.py

```
import mysql.connector

conn = mysql.connector.connect(
    host="localhost",
user="root",     password="",
    database="hospital"
)

print("DATABASE CONNECTION SUCCESSFUL")

c = conn.cursor()

c.execute("DROP TABLE IF EXISTS PATIENT")
c.execute("""
    CREATE TABLE PATIENT (
        PATIENT_ID INT(10) PRIMARY KEY,
        NAME VARCHAR(20) NOT NULL,
        SEX VARCHAR(10) NOT NULL,
        BLOOD_GROUP VARCHAR(5) NOT NULL,
        DOB DATE NOT NULL,
        ADDRESS VARCHAR(100) NOT NULL,
        CONSULT_TEAM VARCHAR(50) NOT NULL,
```

```
        EMAIL VARCHAR(20) NOT NULL
    )
""")
print("PATIENT TABLE CREATED SUCCESSFULLY")

c.execute("DROP TABLE IF EXISTS CONTACT_NO")
c.execute("""
    CREATE TABLE CONTACT_NO (
        PATIENT_ID INT(10) PRIMARY KEY,
        CONTACTNO INT(15) NOT NULL,
        ALT_CONTACT INT(15),
        FOREIGN KEY (PATIENT_ID) REFERENCES PATIENT (PATIENT_ID) ON DELETE CASCADE
    )
""")
print("CONTACT_NO TABLE CREATED SUCCESSFULLY")

c.execute("DROP TABLE IF EXISTS EMPLOYEE")
c.execute("""
    CREATE TABLE EMPLOYEE (
        EMP_ID VARCHAR(10) PRIMARY KEY,
        EMP_NAME VARCHAR(20) NOT NULL,
        SEX VARCHAR(10) NOT NULL,
        AGE INT(5) NOT NULL,
        DESIG VARCHAR(20) NOT NULL,
        SAL INT(10) NOT NULL,
        EXP VARCHAR(100) NOT NULL,
        EMAIL VARCHAR(20) NOT NULL,
        PHONE INT(12)
    )
""")
print("EMPLOYEE TABLE CREATED SUCCESSFULLY")

c.execute("DROP TABLE IF EXISTS TREATMENT")
c.execute("""
    CREATE TABLE TREATMENT (
        PATIENT_ID INT(10) PRIMARY KEY,
        TREATMENT VARCHAR(100) NOT NULL,
        TREATMENT_CODE VARCHAR(30) NOT NULL,
        T_COST INT(20) NOT NULL,
        FOREIGN KEY (PATIENT_ID) REFERENCES PATIENT (PATIENT_ID) ON DELETE CASCADE
    )
""")
print("TREATMENT TABLE CREATED SUCCESSFULLY")

c.execute("DROP TABLE IF EXISTS MEDICINE")
c.execute("""
    CREATE TABLE MEDICINE (
        PATIENT_ID INT(10) PRIMARY KEY,
        MEDICINE_NAME VARCHAR(100) NOT NULL,
        M_COST INT(20) NOT NULL,
        M_QTY INT(10) NOT NULL,
        FOREIGN KEY (PATIENT_ID) REFERENCES PATIENT (PATIENT_ID) ON DELETE CASCADE
    )
```

```
""")
print("MEDICINE TABLE CREATED SUCCESSFULLY")

c.execute("DROP TABLE IF EXISTS ROOM")
c.execute("""
   CREATE TABLE ROOM (
      PATIENT_ID INT(10) NOT NULL,
      ROOM_NO VARCHAR(20) PRIMARY KEY,
      ROOM_TYPE VARCHAR(10) NOT NULL,
      RATE INT(10) NOT NULL,
      DATE_ADMITTED DATE,
      DATE_DISCHARGED DATE NULL,
      FOREIGN KEY (PATIENT_ID) REFERENCES PATIENT (PATIENT_ID) ON DELETE CASCADE
   )
""")
print("ROOM TABLE CREATED SUCCESSFULLY")

c.execute("DROP TABLE IF EXISTS APPOINTMENT")
c.execute("""
   CREATE TABLE APPOINTMENT (
      PATIENT_ID INT(20) NOT NULL,
      EMP_ID VARCHAR(10) NOT NULL,
      AP_NO VARCHAR(10) PRIMARY KEY,
      AP_TIME TIME,
      AP_DATE DATE,
      DESCRIPTION VARCHAR(100),
      FOREIGN KEY (PATIENT_ID) REFERENCES PATIENT (PATIENT_ID) ON DELETE CASCADE,
FOREIGN KEY (EMP_ID) REFERENCES EMPLOYEE (EMP_ID)
   )
""")
print("APPOINTMENT TABLE CREATED SUCCESSFULLY")

conn.commit()
conn.close()
```

Menu.py

```
from tkinter import * import
tkinter.messagebox from tkinter import ttk
from tkinter import font from patient_form
import Patient from room_form import
Room from employee_form import
Employee from appointment_form import
Appointment from billing_form import
Billing
import mysql.connector

conn = mysql.connector.connect(
host="localhost",     user="root",
password="",
   database="hospital"
)

print("DATABASE CONNECTION SUCCESSFUL")

#root=Tk()


class Menu:
    def __init__(self,master):
      self.master = master
       self.master.title("HOSPITAL MANAGEMENT SYSTEM")
       self.master.geometry("1600x800+0+0")
self.master.config(bg="cadet blue")        self.frame =
Frame(self.master,bg="cadet blue")
self.frame.pack()

       self.lblTitle = Label(self.frame,text = "MAIN MENU", font="Helvetica 20 bold",bg="cadet blue")
self.lblTitle.grid(row =0 ,column = 0,columnspan=2,pady=50)

       self.LoginFrame = Frame(self.frame,width=400,height=80,relief="ridge",bg="cadet blue",bd=20)
self.LoginFrame.grid(row=1,column=0)

       self.button1 = Button(self.LoginFrame,text = "1.PATIENT REGISTRATION", width =30,font="Helvetica 14
bold",bg="cadet blue",command=self.Patient_Reg)              self.button1.grid(row=1,column=0,pady=10)

       self.button2 = Button(self.LoginFrame, text="2.ROOM ALLOCATION",width =30,font="Helvetica 14
bold",bg="cadet blue",command=self.Room_Allocation)        self.button2.grid(row=3,column=0,pady=10)
```

```
        self.button3 = Button(self.LoginFrame, text="3.EMPLOYEE REGISTRATION",width =30,font="Helvetica 14
bold",bg="cadet blue",command=self.Employee_Reg)        self.button3.grid(row=5,column=0,pady=10)

        self.button4 = Button(self.LoginFrame, text="4.BOOK APPOINTMENT",width =30,font="Helvetica 14
bold",bg="cadet blue",command=self.Appointment_Form)        self.button4.grid(row=7,column=0,pady=10)

        self.button5 = Button(self.LoginFrame, text="5.PATIENT BILL",width =30,font="Helvetica 14
bold",bg="cadet blue",command=self.Billing_Form)
self.button5.grid(row=9,column=0,pady=10)

        self.button6 = Button(self.LoginFrame, text="6.EXIT",width =30,font="Helvetica 14 bold",bg="cadet
blue",command = self.Exit)
        self.button6.grid(row=11,column=0,pady=10)

        #EXIT FOR MENU
def                 Exit(self):
self.master.destroy()

    def Patient_Reg(self):
        self.newWindow = Toplevel(self.master)
self.app = Patient(self.newWindow)

    def Room_Allocation(self):
self.newWindow = Toplevel(self.master)
        self.app = Room(self.newWindow)

    def Employee_Reg(self):
self.newWindow = Toplevel(self.master)
        self.app = Employee(self.newWindow)

    def                         Appointment_Form(self):
self.newWindow = Toplevel(self.master) self.app
= Appointment(self.newWindow)

    def Billing_Form(self):
self.newWindow = Toplevel(self.master)
        self.app = Billing(self.newWindow)


#root.mainloop()
```

Patient_form.py

```
from tkinter import *
import tkinter.messagebox
from tkinter import ttk from
tkinter import font import
mysql.connector

conn = mysql.connector.connect(
host="localhost",     user="root",
password="",
    database="hospital"
)

#root = Tk()
cursor = conn.cursor()
print("DATABASE CONNECTION SUCCESSFUL")


# PATIENT FORM     class
Patient:     def __init__(self,
master):        self.master =
master
        self.master.title("HOSPITAL MANAGEMENT SYSTEM")
        self.master.geometry("1600x800+0+0")
self.master.config(bg="cadet blue")        self.frame =
Frame(self.master, bg="cadet blue")
        self.frame.pack()

        # =============ATTRIBUTES===========

        self.pat_ID = IntVar()
self.pat_name = StringVar()
self.pat_dob = StringVar()
self.pat_address = StringVar()
self.pat_sex = StringVar() self.pat_BG =
StringVar()        self.pat_email =
StringVar()        self.pat_contact =
IntVar()        self.pat_contactalt =
IntVar()
        self.pat_CT = StringVar()


        # ===============TITLE==========
        self.lblTitle = Label(self.frame, text="PATIENT REGISTRATION FORM", font="Helvetica 20 bold",
bg="cadet blue")
        self.lblTitle.grid(row=0, column=0, columnspan=2, pady=50)
        # ==============FRAME==========
        self.LoginFrame = Frame(self.frame, width=400, height=80, relief="ridge", bg="cadet blue", bd=20)
self.LoginFrame.grid(row=1, column=0)

        self.LoginFrame2 = Frame(self.frame, width=400, height=80, relief="ridge", bg="cadet blue", bd=20)
self.LoginFrame2.grid(row=2, column=0)        # ===========LABELS=============
```

```
        self.lblpatid = Label(self.LoginFrame, text="PATIENT ID", font="Helvetica 14 bold", bg="cadet blue", bd=22)
        self.lblpatid.grid(row=0, column=0)
        self.lblpatid  = Entry(self.LoginFrame, font="Helvetica 14 bold", bd=2, textvariable=self.pat_ID)
self.lblpatid.grid(row=0, column=1)

        self.lblPatname = Label(self.LoginFrame, text="PATIENT NAME", font="Helvetica 14 bold", bg="cadet blue",
bd=22)
        self.lblPatname.grid(row=1, column=0)
        self.lblPatname  = Entry(self.LoginFrame, font="Helvetica 14 bold", bd=2, textvariable=self.pat_name)
self.lblPatname.grid(row=1, column=1)

        self.lblsex = Label(self.LoginFrame, text="SEX", font="Helvetica 14 bold", bg="cadet blue", bd=22)
self.lblsex.grid(row=2, column=0)
        self.lblsex  = Entry(self.LoginFrame, font="Helvetica 14 bold", bd=2, textvariable=self.pat_sex)
self.lblsex.grid(row=2, column=1)

        self.lblDOB = Label(self.LoginFrame, text="DOB (YYYY-MM-DD)", font="Helvetica 14 bold", bg="cadet
blue", bd=22)
        self.lblDOB.grid(row=3, column=0)
        self.lblDOB  = Entry(self.LoginFrame, font="Helvetica 14 bold", bd=2, textvariable=self.pat_dob)
self.lblDOB.grid(row=3, column=1)

        self.lblbgrp = Label(self.LoginFrame, text="BLOOD GROUP", font="Helvetica 14 bold", bg="cadet blue",
bd=22)
        self.lblbgrp.grid(row=4, column=0)
        self.lblbgrp  = Entry(self.LoginFrame, font="Helvetica 14 bold", bd=2, textvariable=self.pat_BG)
self.lblbgrp.grid(row=4, column=1)

        self.lblCon = Label(self.LoginFrame, text="CONTACT NUMBER", font="Helvetica 14 bold", bg="cadet
blue", bd=22)
        self.lblCon.grid(row=0, column=2)
        self.lblCon  = Entry(self.LoginFrame, font="Helvetica 14 bold", bd=2, textvariable=self.pat_contact)
self.lblCon.grid(row=0, column=3)

        self.lblAlt = Label(self.LoginFrame, text="ALTERNATE CONTACT", font="Helvetica 14 bold", bg="cadet
blue", bd=22) self.lblAlt.grid(row=1, column=2)
        self.lblAlt  = Entry(self.LoginFrame, font="Helvetica 14 bold", bd=2, textvariable=self.pat_contactalt)
        self.lblAlt.grid(row=1, column=3)

        self.lbleid = Label(self.LoginFrame, text="EMAIL", font="Helvetica 14 bold", bg="cadet blue", bd=22)
self.lbleid.grid(row=2, column=2)
        self.lbleid  = Entry(self.LoginFrame, font="Helvetica 14 bold", bd=2, textvariable=self.pat_email)
self.lbleid.grid(row=2, column=3)

        self.lbldoc = Label(self.LoginFrame, text="CONSULTING TEAM / DOCTOR", font="Helvetica 14 bold",
bg="cadet blue", bd=22)
        self.lbldoc.grid(row=3, column=2)
        self.lbldoc  = Entry(self.LoginFrame, font="Helvetica 14 bold", bd=2, textvariable=self.pat_CT)
self.lbldoc.grid(row=3, column=3)

        self.lbladd = Label(self.LoginFrame, text="ADDRESS", font="Helvetica 14 bold", bg="cadet blue", bd=22)
self.lbladd.grid(row=4, column=2)
```

```
        self.lbladd  = Entry(self.LoginFrame, font="Helvetica 14 bold", bd=2, textvariable=self.pat_address)
self.lbladd.grid(row=4, column=3)

        self.button2 = Button(self.LoginFrame2, text="SUBMIT", width=10, font="Helvetica 14 bold", bg="cadet
blue", command=self.INSERT_PAT)
        self.button2.grid(row=3, column=1)

        self.button3 = Button(self.LoginFrame2, text="UPDATE", width=10, font="Helvetica 14 bold", bg="cadet
blue", command=self.UPDATE_PAT)
        self.button3.grid(row=3, column=2)

        self.button4 = Button(self.LoginFrame2, text="DELETE", width=10, font="Helvetica 14 bold", bg="cadet
blue", command=self.D_DISPLAY)
        self.button4.grid(row=3, column=3)

        self.button5 = Button(self.LoginFrame2, text="SEARCH", width=10, font="Helvetica 14 bold", bg="cadet
blue", command=self.S_DISPLAY)
        self.button5.grid(row=3, column=4)

        self.button6 = Button(self.LoginFrame2, text="EXIT", width=10, font="Helvetica 14 bold", bg="cadet blue",
command=self.Exit)
        self.button6.grid(row=3, column=5)



    def clear(self):
self.lblpatid.delete(0, 'end')
self.lblPatname.delete(0, 'end')
self.lblsex.delete(0, 'end')
self.lblDOB.delete(0, 'end')
self.lblbgrp.delete(0, 'end')
self.lblCon.delete(0, 'end')
self.lblAlt.delete(0, 'end')
self.lbleid.delete(0, 'end')
self.lbldoc.delete(0, 'end')
self.lbladd.delete(0, 'end')

        # Clear the corresponding fields in the database table
        #query = "DELETE FROM your_table_name"
        #cursor.execute(query) conn.commit()
        #INSERT DATA IN PATIENT FORM
        def INSERT_PAT(self):
        global pp1, pp2, pp3, pp4, pp5, pp6, pp7, pp8, pp9, pp10, ce1, con
conn = mysql.connector.connect(
            host="localhost",
user="root",            password="",
            database="hospital"
        )
        p1 = self.pat_ID.get()
p2 = self.pat_name.get()        p3
= self.pat_sex.get()        p4 =
self.pat_BG.get()        p5 =
self.pat_dob.get()        p6 =
```

```
self.pat_contact.get()        p7 =
self.pat_contactalt.get()        p8 =
self.pat_address.get()        p9 =
self.pat_CT.get()        p10 =
self.pat_email.get()        cursor =
conn.cursor()
        cursor.execute("SELECT * FROM PATIENT WHERE PATIENT_ID = %s", (p1,))
p = cursor.fetchall()        x = len(p)        if x != 0:
            tkinter.messagebox.showerror("HOSPITAL  DATABASE  SYSTEM",  "PATIENT_ID ALREADY
EXISTS")        else:
            cursor.execute('INSERT INTO PATIENT (PATIENT_ID, NAME, SEX, BLOOD_GROUP, DOB,
ADDRESS, CONSULT_TEAM, EMAIL) VALUES (%s, %s, %s, %s, %s, %s, %s, %s)',
                        (p1, p2, p3, p4, p5, p8, p9, p10))
            cursor.execute('INSERT INTO CONTACT_NO (PATIENT_ID, CONTACTNO, ALT_CONTACT) VALUES
(%s, %s, %s)', (p1, p6, p7))
            tkinter.messagebox.showinfo("HOSPITAL DATABASE SYSTEM", "DETAILS INSERTED INTO
DATABASE")
        self.clear()
conn.commit()
        conn.close()
    #UPDATE DATA IN PATIENT FORM

    def UPDATE_PAT(self):
        global u1, u2, u3, u4, u5, u6, u7, u8, u9, u10
u1 = self.pat_ID.get()        u2 =
self.pat_name.get()        u3 = self.pat_sex.get()
u4 = self.pat_dob.get()        u5 =
self.pat_BG.get()        u6 = self.pat_contact.get()
u7 = self.pat_contactalt.get() u8 =
self.pat_email.get() u9 = self.pat_CT.get()
        u10 = self.pat_address.get()

        # Perform the update operation in the MySQL database
        query = "UPDATE PATIENT SET NAME=%s, SEX=%s, DOB=%s, BLOOD_GROUP=%s, ADDRESS=%s,
CONSULT_TEAM=%s, EMAIL=%s WHERE PATIENT_ID=%s"
        values = (u2, u3, u4, u5, u10, u9, u8, u1)
cursor.execute(query, values)

        query = "UPDATE CONTACT_NO SET CONTACTNO=%s, ALT_CONTACT=%s WHERE
PATIENT_ID=%s"
        values = (u6, u7, u1)
cursor.execute(query, values)

        conn.commit()
        tkinter.messagebox.showinfo("HOSPITAL DATABASE SYSTEM", "DETAILS UPDATED INTO
DATABASE")
        self.clear()

    def Exit(self):
self.master.destroy()

    def D_DISPLAY(self):
self.newWindow = Toplevel(self.master)
```

```
        self.app = DMenu(self.newWindow)

    def S_DISPLAY(self):
self.newWindow = Toplevel(self.master)
        self.app = SMenu(self.newWindow)

class DMenu:     def __init__(self,
master):        global inp_d, entry1,
DeleteB        self.master = master
        self.master.title("HOSPITAL MANAGEMENT SYSTEM")
        self.master.geometry("1600x800+0+0")
self.master.config(bg="cadet blue")        self.frame =
Frame(self.master, bg="cadet blue")
self.frame.pack()        self.del_pid = IntVar()
        self.lblTitle = Label(self.frame, text="DELETE WINDOW", font="Helvetica 20 bold", bg="cadet blue")
self.lblTitle.grid(row=0, column=0, columnspan=2, pady=50)
        #==============FRAME==========
        self.LoginFrame = Frame(self.frame, width=400, height=80, relief="ridge", bg="cadet blue", bd=20)
self.LoginFrame.grid(row=1, column=0)
        self.LoginFrame2 = Frame(self.frame, width=400, height=80, relief="ridge", bg="cadet blue", bd=20)
self.LoginFrame2.grid(row=2, column=0)        #===========LABELS=============
        self.lblpatid = Label(self.LoginFrame, text="ENTER PATIENT ID TO DELETE", font="Helvetica 14 bold",
bg="cadet blue", bd=22)        self.lblpatid.grid(row=0, column=0)
        self.lblpatid = Entry(self.LoginFrame, font="Helvetica 14 bold", bd=2, textvariable=self.del_pid)
self.lblpatid.grid(row=0, column=1)

        self.DeleteB = Button(self.LoginFrame2, text="DELETE", width=10, font="Helvetica 14 bold", bg="cadet
blue",
                              command=self.DELETE_PAT)
        self.DeleteB.grid(row=3, column=1)

    def DELETE_PAT(self):
global inp_d, del_pid        cursor
= conn.cursor()        inp_d =
self.del_pid.get()
        cursor.execute("SELECT * FROM PATIENT WHERE PATIENT_ID = %s", (inp_d,))
p = cursor.fetchall()        if len(p) == 0:
            tkinter.messagebox.showerror("HOSPITAL DATABASE SYSTEM", "PATIENT RECORD NOT
FOUND")        else:
            cursor.execute("DELETE FROM PATIENT WHERE PATIENT_ID = %s", (inp_d,))
conn.commit()
            tkinter.messagebox.showinfo("HOSPITAL DATABASE SYSTEM", "DETAILS DELETED FROM
DATABASE")


class SMenu:     def __init__(self,
master):        global inp_s, s_pid,
SearchB        self.master =
master
        self.master.title("HOSPITAL MANAGEMENT SYSTEM")
        self.master.geometry("1600x800+0+0")
self.master.config(bg="cadet blue")        self.frame =
```

```
Frame(self.master, bg="cadet blue")
self.frame.pack()        self.s_pid = IntVar()
        self.lblTitle = Label(self.frame, text="SEARCH WINDOW", font="Helvetica 20 bold", bg="cadet blue")
self.lblTitle.grid(row=0, column=0, columnspan=2, pady=25)
        # ==============FRAME==========
        self.LoginFrame = Frame(self.frame, width=400, height=80, relief="ridge", bg="cadet blue", bd=20)
self.LoginFrame.grid(row=1, column=0)
        self.LoginFrame2 = Frame(self.frame, width=400, height=80, relief="ridge", bg="cadet blue", bd=20)
self.LoginFrame2.grid(row=2, column=0)

        # ===========LABELS=============
        self.lblpatid = Label(self.LoginFrame, text="ENTER PATIENT ID TO SEARCH", font="Helvetica 14 bold",
bg="cadet blue", bd=22)
        self.lblpatid.grid(row=0, column=0)
        self.lblpatid = Entry(self.LoginFrame, font="Helvetica 14 bold", bd=2, textvariable=self.s_pid)
self.lblpatid.grid(row=0, column=1)

        self.SearchB = Button(self.LoginFrame2, text="SEARCH", width=10, font="Helvetica 14 bold", bg="cadet
blue", command=self.SEARCH_PAT)
        self.SearchB.grid(row=0, column=1)


    def SEARCH_PAT(self):
        global inp_s, s_pid, errorS, t, i, dis1, dis2, dis3, dis4, dis5, dis6, dis7, dis8, dis9, dis10, l1, l2, l3, l4, l5, l6, l7, l8,
l9, l10        c1 = conn.cursor()        inp_s = self.s_pid.get()
        c1.execute('SELECT * FROM PATIENT NATURAL JOIN CONTACT_NO WHERE PATIENT_ID = %s',
(inp_s,))
        p = c1.fetchall()
        if len(p) == 0:
            tkinter.messagebox.showerror("HOSPITAL DATABASE SYSTEM", "PATIENT RECORD NOT
FOUND")        else:            for i in p:
                self.l1 = Label(self.LoginFrame, text="PATIENT ID", font="Helvetica 14 bold", bg="cadet blue", bd=22)
self.l1.grid(row=1, column=0)
                self.dis1 = Label(self.LoginFrame, font="Helvetica 14 bold", bd=2, bg="cadet blue", text=i[0])
self.dis1.grid(row=1, column=1)

                self.l2 = Label(self.LoginFrame, text="PATIENT NAME", font="Helvetica 14 bold", bg="cadet blue",
bd=22)
                self.l2.grid(row=2, column=0)
                self.dis2 = Label(self.LoginFrame, font="Helvetica 14 bold", bd=2, bg="cadet blue", text=i[1])
self.dis2.grid(row=2, column=1)

                self.l3 = Label(self.LoginFrame, text="SEX", font="Helvetica 14 bold", bg="cadet blue", bd=22)
self.l3.grid(row=3, column=0)
                self.dis3 = Label(self.LoginFrame, font="Helvetica 14 bold", bg="cadet blue", bd=2, text=i[2])
self.dis3.grid(row=3, column=1)

                self.l4 = Label(self.LoginFrame, text="DOB (YYYY-MM-DD)", font="Helvetica 14 bold", bg="cadet
blue", bd=22)
                self.l4.grid(row=4, column=0)
                self.dis4 = Label(self.LoginFrame, font="Helvetica 14 bold", bg="cadet blue", bd=2, text=i[3])
self.dis4.grid(row=4, column=1)
```

```
        self.l5 = Label(self.LoginFrame, text="BLOOD GROUP", font="Helvetica 14 bold", bg="cadet blue",
bd=22)
        self.l5.grid(row=5, column=0)
        self.dis5 = Label(self.LoginFrame, font="Helvetica 14 bold", bg="cadet blue", bd=2, text=i[4])
self.dis5.grid(row=5, column=1)

        self.l6 = Label(self.LoginFrame, text="ADDRESS", font="Helvetica 14 bold", bg="cadet blue", bd=22)
self.l6.grid(row=1, column=2)
        self.dis6 = Label(self.LoginFrame, font="Helvetica 14 bold", bg="cadet blue", bd=2, text=i[5])
self.dis6.grid(row=1, column=3)

        self.l7 = Label(self.LoginFrame, text="CONSULTING TEAM / DOCTOR", font="Helvetica 14 bold",
bg="cadet blue",
                    bd=22)
        self.l7.grid(row=2, column=2)
        self.dis7 = Label(self.LoginFrame, font="Helvetica 14 bold", bd=2, bg="cadet blue", text=i[6])
self.dis7.grid(row=2, column=3)
        self.l8 = Label(self.LoginFrame, text="EMAIL", font="Helvetica 14 bold", bg="cadet blue", bd=22)
self.l8.grid(row=3, column=2)
        self.dis8 = Label(self.LoginFrame, font="Helvetica 14 bold", bd=2, bg="cadet blue", text=i[7])
self.dis8.grid(row=3, column=3)
        self.l9 = Label(self.LoginFrame, text="CONTACT NUMBER", font="Helvetica 14 bold", bg="cadet
blue", bd=22)
        self.l9.grid(row=4, column=2)
        self.dis9 = Label(self.LoginFrame, font="Helvetica 14 bold", bd=2, bg="cadet blue", text=i[8])
self.dis9.grid(row=4, column=3)

        self.l10 = Label(self.LoginFrame, text="ALTERNATE CONTACT", font="Helvetica 14 bold", bg="cadet
blue", bd=22)
        self.l10.grid(row=5, column=2)
        self.dis10 = Label(self.LoginFrame, font="Helvetica 14 bold", bd=2, bg="cadet blue", text=i[9])
self.dis10.grid(row=5, column=3)

#root.mainloop()
```

Billing_form.py

```
import mysql.connector
from tkinter import * from
tkinter import ttk import
tkinter.messagebox
from tkinter import font

conn = mysql.connector.connect(
   host="localhost",
user="root",     password="",
   database="hospital"
)

#root = Tk()
print("DATABASE CONNECTION SUCCESSFUL")
class   Billing:              def   __init__(self,   master):
self.master = master
      self.master.title("HOSPITAL MANAGEMENT SYSTEM")
```

```
        self.master.geometry("1600x800+0+0")
self.master.config(bg="cadet blue")         self.frame =
Frame(self.master, bg="cadet blue")
self.frame.pack()

        self.P_id = IntVar()
self.dd = StringVar()
self.treat_1 = StringVar()
self.treat_2 = StringVar()
self.cost_t = IntVar()
self.med = StringVar()
self.med_q = IntVar()
        self.price = IntVar()

        self.lblTitle = Label(self.frame, text="BILLING WINDOW", font="Helvetica 20 bold", bg="cadet blue")
self.lblTitle.grid(row=0, column=0, columnspan=2, pady=25)

        self.LoginFrame = Frame(self.frame, width=400, height=80, relief="ridge", bg="cadet blue", bd=20)
self.LoginFrame.grid(row=1, column=0)

        self.LoginFrame2 = Frame(self.frame, width=400, height=80, relief="ridge", bg="cadet blue", bd=20)
self.LoginFrame2.grid(row=2, column=0)

        self.lblpid = Label(self.LoginFrame, text="PATIENT ID", font="Helvetica 14 bold", bg="cadet blue", bd=22)
self.lblpid.grid(row=0, column=0)

        self.lblpid = Entry(self.LoginFrame, font="Helvetica 14 bold", bd=2, textvariable=self.P_id)
self.lblpid.grid(row=0, column=1)

         self.lbldid = Label(self.LoginFrame, text="DATE DISCHARGED(YYYY-MM-DD)", font="Helvetica 14
bold",bg="cadet blue", bd=22)
self.lbldid.grid(row=1, column=0)

        self.lbldid = Entry(self.LoginFrame, font="Helvetica 14 bold", bd=2, textvariable=self.dd)
self.lbldid.grid(row=1, column=1)

        self.button2 = Button(self.LoginFrame, text="UPDATE DISCHARGE DATE", width=25, font="Helvetica 14
bold",bg="cadet blue", command=self.UPDATE_DATE)         self.button2.grid(row=1, column=3)

        self.lbltreat = Label(self.LoginFrame, text="TREATMENT", font="Helvetica 14 bold", bg="cadet blue",
bd=22)
        self.lbltreat.grid(row=2, column=0)

        self.lbltreat = Entry(self.LoginFrame, font="Helvetica 14 bold", bd=2, textvariable=self.treat_1)
self.lbltreat.grid(row=2, column=1)

        self.lblcode_t1 = Label(self.LoginFrame, text="TREATMENT CODE", font="Helvetica 14 bold", bg="cadet
blue",bd=22)
        self.lblcode_t1.grid(row=3, column=0)

        self.lblcode_t1 = Entry(self.LoginFrame, font="Helvetica 14 bold", bd=2, textvariable=self.treat_2)
self.lblcode_t1.grid(row=3, column=1)
```

```
        self.lblap = Label(self.LoginFrame, text="TREATMENT COST ₹", font="Helvetica 14 bold", bg="cadet
blue",bd=22)
        self.lblap.grid(row=4, column=0)

        self.lblap = Entry(self.LoginFrame, font="Helvetica 14 bold", bd=2, textvariable=self.cost_t)
self.lblap.grid(row=4, column=1)

        self.lblmed = Label(self.LoginFrame, text="MEDICINE", font="Helvetica 14 bold", bg="cadet blue", bd=22)
self.lblmed.grid(row=2, column=2)

        self.lblmed = Entry(self.LoginFrame, font="Helvetica 14 bold", bd=2, textvariable=self.med)
self.lblmed.grid(row=2, column=3)

        self.med_t1 = Label(self.LoginFrame, text="MEDICINE QUANTITY", font="Helvetica 14 bold", bg="cadet
blue",bd=22)
        self.med_t1.grid(row=3, column=2)

        self.med_t1 = Entry(self.LoginFrame, font="Helvetica 14 bold", bd=2, textvariable=self.med_q)
self.med_t1.grid(row=3, column=3)

        self.lblapd = Label(self.LoginFrame, text="MEDICINE PRICE ₹", font="Helvetica 14 bold", bg="cadet
blue",bd=22)
        self.lblapd.grid(row=4, column=2)

        self.lblapd = Entry(self.LoginFrame, font="Helvetica 14 bold", bd=2, textvariable=self.price)
self.lblapd.grid(row=4, column=3)

        self.button3 = Button(self.LoginFrame2, text="UPDATE DATA", width=15, font="Helvetica 14
bold",bg="cadet blue", command=self.UPDATE_DATA) self.button3.grid(row=3, column=2)

        self.button3 = Button(self.LoginFrame2, text="GENERATE BILL", width=15, font="Helvetica 14
bold",bg="cadet blue", command=self.GEN_BILL)
        self.button3.grid(row=3, column=3)

        self.button6 = Button(self.LoginFrame2, text="EXIT", width=10, font="Helvetica 14 bold", bg="cadet
blue",command=self.Exit)
        self.button6.grid(row=3, column=4)

    def UPDATE_DATE(self):
        global b1, b2
        conn = mysql.connector.connect(
            host="localhost",
user="root",            password="",
            database="hospital"
        )
        cursor = conn.cursor()
b1 = self.P_id.get()        b2
= self.dd.get()
        cursor.execute("UPDATE ROOM SET DATE_DISCHARGED=%s WHERE ROOM.PATIENT_ID=%s", (b2,
b1))
        tkinter.messagebox.showinfo("HOSPITAL DATABASE SYSTEM", "DISCHARGE DATE UPDATED")
conn.commit()
```

```
    def UPDATE_DATA(self):
        global c1, b1, P_id, b3, b4, b5, b6, dd, treat_1, treat_2, cost_t, b7, b8, med, med_q, price, u
conn = mysql.connector.connect(
            host="localhost",
user="root",            password="",
            database="hospital"
        )
        c1 = conn.cursor()
b1 = self.P_id.get()
b3 = self.treat_1.get()
b4 = self.treat_2.get()
b5 = self.cost_t.get()
b6 = self.med.get()
b7 = self.med_q.get()
b8 = self.price.get()
        c1.execute("SELECT * FROM TREATMENT WHERE TREATMENT.PATIENT_ID=%s", (b1,))
p = c1.fetchall()        if len(p) != 0:
            tkinter.messagebox.showerror("HOSPITAL DATABASE SYSTEM", "PATIENT ID IS ALREADY
REGISTERED")        else:
            c1.execute("INSERT INTO TREATMENT VALUES(%s, %s, %s, %s)", (b1, b3, b4, b5))
c1.execute("INSERT INTO MEDICINE VALUES(%s, %s, %s, %s)", (b1, b6, b7, b8))            conn.commit()
            tkinter.messagebox.showinfo("HOSPITAL DATABASE SYSTEM", "BILLING DATA SAVED")


    def GEN_BILL(self):        global
b1 b1 = (self.P_id.get()) conn =
mysql.connector.connect(
host="localhost",            user="root",
password="",
            database="hospital"
        )
        cursor = conn.cursor()
        cursor.execute("SELECT SUM(T_COST + (M_COST * M_QTY) + (DATE_DISCHARGED -
DATE_ADMITTED) * RATE) FROM ROOM NATURAL JOIN TREATMENT NATURAL JOIN MEDICINE
WHERE PATIENT_ID=%s", (b1,))
        result = cursor.fetchone()[0]
        self.pp = Label(self.LoginFrame, text="TOTAL AMOUNT OUTSTANDING", font="Helvetica 14 bold",
bg="cadet blue", bd=22)        self.pp.grid(row=5, column=0)
        self.uu = Label(self.LoginFrame, font="Helvetica 14 bold", bg="cadet blue", bd=22, text=result)
self.uu.grid(row=5, column=1)

        cursor.close()
        conn.close()

    def Exit(self):
self.master.destroy()

#root.mainloop()
```

Appointment_form.py

```
from tkinter import * import
tkinter.messagebox from tkinter
import ttk from tkinter import
font import mysql.connector
conn = mysql.connector.connect(
   host="localhost",
user="root",     password="",
   database = "hospital"
)
#root = Tk()
cursor=conn.cursor()
print("DATABASE CONNECTION SUCCESSFUL") class
Appointment:
    def __init__(self,master):
      self.master = master
       self.master.title("HOSPITAL MANAGEMENT SYSTEM")
       self.master.geometry("1600x800+0+0")
self.master.config(bg="cadet blue")        self.frame =
Frame(self.master,bg="cadet blue")
self.frame.pack()

       #=============ATTRIBUTES===========

       self.pat_ID=IntVar() self.emp_ID=StringVar()
       self.ap_no=StringVar()
       self.ap_time=StringVar()
       self.ap_date=StringVar()
       self.des=StringVar()

       #===============TITLE==========
       self.lblTitle = Label(self.frame,text = "APPOINTMENT FORM", font="Helvetica 20 bold",bg="cadet blue")
self.lblTitle.grid(row =0 ,column = 0,columnspan=2,pady=50)
       #==============FRAME==========
       self.LoginFrame = Frame(self.frame,width=400,height=80,relief="ridge",bg="cadet blue",bd=20)
self.LoginFrame.grid(row=1,column=0)

       self.LoginFrame2 = Frame(self.frame,width=400,height=80,relief="ridge",bg="cadet blue",bd=20)
self.LoginFrame2.grid(row=2,column=0)        #===========LABELS=============
       self.lblpid = Label(self.LoginFrame,text="PATIENT ID",font="Helvetica 14 bold",bg="cadet blue",bd=22)
self.lblpid.grid(row=0,column=0)
       self.lblpid  = Entry(self.LoginFrame,font="Helvetica 14 bold",bd=2,textvariable= self.pat_ID)
self.lblpid.grid(row=0,column=1)

       self.lbldid = Label(self.LoginFrame,text="DOCTOR ID",font="Helvetica 14 bold",bg="cadet blue",bd=22)
self.lbldid.grid(row=1,column=0)
       self.lbldid  = Entry(self.LoginFrame,font="Helvetica 14 bold",bd=2,textvariable=self.emp_ID )
self.lbldid.grid(row=1,column=1)


       self.lblap = Label(self.LoginFrame,text="APPOINTMENT NO",font="Helvetica 14 bold",bg="cadet
blue",bd=22)
       self.lblap.grid(row=2,column=0)
```

```
        self.lblap  = Entry(self.LoginFrame,font="Helvetica 14 bold",bd=2,textvariable=self.ap_no )
self.lblap.grid(row=2,column=1)

        self.lblapt = Label(self.LoginFrame,text="APPOINTMENT TIME(HH:MM:SS)",font="Helvetica 14
bold",bg="cadet blue",bd=22)         self.lblapt.grid(row=0,column=2)
        self.lblapt  = Entry(self.LoginFrame,font="Helvetica 14 bold",bd=2,textvariable=self.ap_time )
self.lblapt.grid(row=0,column=3)

        self.lblapd = Label(self.LoginFrame,text="APPOINTMENT DATE(YYYY-MM-DD)",font="Helvetica 14
bold",bg="cadet blue",bd=22)         self.lblapd.grid(row=1,column=2)
        self.lblapd  = Entry(self.LoginFrame,font="Helvetica 14 bold",bd=2,textvariable= self.ap_date)
self.lblapd.grid(row=1,column=3)

        self.lbldes = Label(self.LoginFrame,text="DESCRIPTION",font="Helvetica 14 bold",bg="cadet blue",bd=22)
self.lbldes.grid(row=2,column=2)
        self.lbldes  = Entry(self.LoginFrame,font="Helvetica 14 bold",bd=2,textvariable=self.des)
self.lbldes.grid(row=2,column=3)


        self.button2 = Button(self.LoginFrame2, text="SAVE",width =10,font="Helvetica 14 bold",bg="cadet
blue",command = self.INSERT_AP)
        self.button2.grid(row=3,column=1)

        self.button3 = Button(self.LoginFrame2, text="DELETE",width =10,font="Helvetica 14 bold",bg="cadet
blue",command= self.DE_AP_DISPLAY) self.button3.grid(row=3,column=2)


        self.button3 = Button(self.LoginFrame2, text="SEARCH APPOINTMENTS",width =20,font="Helvetica 14
bold",bg="cadet blue",command= self.S_AP_DISPLAY)
        self.button3.grid(row=3,column=3)

        self.button6 = Button(self.LoginFrame2, text="EXIT",width =10,font="Helvetica 14 bold",bg="cadet
blue",command = self.Exit)
        self.button6.grid(row=3,column=4)
      def Exit(self):
self.master.destroy()

    def INSERT_AP(self):
        global e1, e2, e3, e4, e5, e6
e1 = self.pat_ID.get()         e2 =
self.emp_ID.get()         e3 =
self.ap_no.get()         e4 =
self.ap_time.get()         e5 =
self.ap_date.get()
        e6 = self.des.get()

        conn = mysql.connector.connect(
            host="localhost",
user="root",            password="",
            database="hospital"
        )
        cursor = conn.cursor()
```

```
        cursor.execute("SELECT * FROM appointment WHERE AP_NO = %s", (e3,))
p = cursor.fetchall()         x = len(p)
         if x !=
0:
            tkinter.messagebox.showerror("HOSPITAL DATABASE SYSTEM", "APPOINTMENT ALREADY
EXISTS")         else:
            cursor.execute("INSERT INTO appointment VALUES (%s, %s, %s, %s, %s, %s)", (e1, e2, e3, e4, e5, e6))
tkinter.messagebox.showinfo("HOSPITAL DATABASE SYSTEM", "APPOINTMENT SET SUCCESSFULLY")

        conn.commit()

    def DE_AP_DISPLAY(self):
        self.newWindow = Toplevel(self.master)
        self.app = DEL_AP(self.newWindow)

    def S_AP_DISPLAY(self):
        self.newWindow = Toplevel(self.master)
        self.app = SEA_AP(self.newWindow)


class DEL_AP:     def
__init__(self,master):
        global de1_ap,de
        self.master = master
        self.master.title("HOSPITAL MANAGEMENT SYSTEM")
        self.master.geometry("1600x800+0+0")
        self.master.config(bg="cadet blue")
self.frame = Frame(self.master,bg="cadet blue")
self.frame.pack()         self.de1_ap=StringVar()
        self.lblTitle = Label(self.frame,text = "DELETE APPOINTMENT WINDOW", font="Helvetica 20
bold",bg="cadet blue")
        self.lblTitle.grid(row =0 ,column = 0,columnspan=2,pady=50)
        #==============FRAME==========
        self.LoginFrame = Frame(self.frame,width=400,height=80,relief="ridge",bg="cadet blue",bd=20)
self.LoginFrame.grid(row=1,column=0)
        self.LoginFrame2 = Frame(self.frame,width=400,height=80,relief="ridge",bg="cadet blue",bd=20)
self.LoginFrame2.grid(row=2,column=0)         #===========LABELS=============
        self.lblpatid = Label(self.LoginFrame,text="ENTER APPOINTMENT NO TO DELETE",font="Helvetica 14
bold",bg="cadet blue",bd=22)         self.lblpatid.grid(row=0,column=0)
        self.lblpatid= Entry(self.LoginFrame,font="Helvetica 14 bold",bd=2,textvariable= self.de1_ap)
self.lblpatid.grid(row=0,column=1)

        self.DeleteB = Button(self.LoginFrame2, text="DELETE",width =10,font="Helvetica 14 bold",bg="cadet
blue",command = self.DELETE_AP)
        self.DeleteB.grid(row=3,column=1)

    def DELETE_AP(self):         global
inp_d         inp_d =
str(self.de1_ap.get())         conn =
mysql.connector.connect(
            host="localhost",
user="root",             password="",
            database="hospital"
```

```
        )
        cursor = conn.cursor()

        cursor.execute("SELECT * FROM appointment WHERE AP_NO = %s", (inp_d,))
v = cursor.fetchall()

        if len(v) == 0:
            tkinter.messagebox.showerror("HOSPITAL DATABASE SYSTEM", "PATIENT APPOINTMENT NOT
FIXED")        else:
            cursor.execute('DELETE FROM APPOINTMENT WHERE AP_NO = %s', (inp_d,))
            tkinter.messagebox.showinfo("HOSPITAL DATABASE SYSTEM", "PATIENT APPOINTMENT
DELETED")

        conn.commit()


class    SEA_AP:                    def
__init__(self,master):
global inp_s,entry,SearchB
        self.master = master
        self.master.title("HOSPITAL MANAGEMENT SYSTEM")
        self.master.geometry("1600x800+0+0")
        self.master.config(bg="cadet blue") self.frame =
        Frame(self.master,bg="cadet blue")
        self.frame.pack()
self.entry=StringVar()
        self.lblTitle = Label(self.frame,text = "SEARCH APPOINTMENT WINDOW", font="Helvetica 20
bold",bg="cadet blue")
        self.lblTitle.grid(row =0 ,column = 0,columnspan=2,pady=25)
        #==============FRAME==========
        self.LoginFrame = Frame(self.frame,width=400,height=80,relief="ridge",bg="cadet blue",bd=20)
self.LoginFrame.grid(row=1,column=0)
        self.LoginFrame2 = Frame(self.frame,width=400,height=80,relief="ridge",bg="cadet blue",bd=20)
self.LoginFrame2.grid(row=2,column=0)

        #===========LABELS=============
        self.lblpatid = Label(self.LoginFrame,text="ENTER DATE TO VIEW APPOINTMENTS(YYYY-MM-
DD)",font="Helvetica 14 bold",bg="cadet blue",bd=22)
        self.lblpatid.grid(row=0,column=0)
        self.lblpatid= Entry(self.LoginFrame,font="Helvetica 14 bold",bd=2,textvariable= self.entry)
self.lblpatid.grid(row=0,column=1)

        self.SearchB = Button(self.LoginFrame2, text="SEARCH",width =10,font="Helvetica 14 bold",bg="cadet
blue",command = self.SEARCH_AP)
        self.SearchB.grid(row=0,column=1)

    def SEARCH_AP(self):
        global inp_s, entry, errorS, t, i, q, dis1, dis2, dis3, dis4, dis5, dis6, dis7, dis8, dis9, dis10, l1, l2, l3, l4, l5, l6, l7,
l8, l9, l10        c1 = conn.cursor()
        ap = (self.entry.get())

        c1.execute("SELECT * FROM appointment WHERE AP_DATE = %s", (ap,))
p = c1.fetchall()
```

```
        if len(p) == 0:
            tkinter.messagebox.showerror("HOSPITAL DATABASE SYSTEM", "NO APPOINTMENT FOR
TODAY")        else:
            c1.execute('SELECT PATIENT_ID, NAME, AP_NO, EMP_ID, AP_DATE, AP_TIME FROM PATIENT
NATURAL JOIN appointment WHERE AP_DATE = %s', (ap,))
t = c1.fetchall()

            self.l1 = Label(self.LoginFrame, text="PATIENT ID", font="Helvetica 14 bold", bg="cadet blue", bd=22)
self.l1.grid(row=1, column=0)

            self.dis1 = Label(self.LoginFrame, font="Helvetica 14 bold", bd=2, bg="cadet blue", text=t[0][0])
self.dis1.grid(row=1, column=1)

            self.l2 = Label(self.LoginFrame, text="PATIENT NAME", font="Helvetica 14 bold", bg="cadet blue",
bd=22)
            self.l2.grid(row=2, column=0)

            self.dis2 = Label(self.LoginFrame, font="Helvetica 14 bold", bd=2, bg="cadet blue", text=t[0][1])
self.dis2.grid(row=2, column=1)

            self.l3 = Label(self.LoginFrame, text="APPOINTMENT NO", font="Helvetica 14 bold", bg="cadet blue",
bd=22)
            self.l3.grid(row=3, column=0)

            self.dis3 = Label(self.LoginFrame, font="Helvetica 14 bold", bg="cadet blue", bd=2, text=t[0][2])
self.dis3.grid(row=3, column=1)

            self.l4 = Label(self.LoginFrame, text="DOCTOR ID", font="Helvetica 14 bold", bg="cadet blue", bd=22)
self.l4.grid(row=4, column=0)

            self.dis4 = Label(self.LoginFrame, font="Helvetica 14 bold", bg="cadet blue", bd=2, text=t[0][3])
self.dis4.grid(row=4, column=1
            self.l5 = Label(self.LoginFrame, text="APPOINTMENT TIME(HH:MM:SS)", font="Helvetica 14 bold",
bg="cadet blue", bd=22)
            self.l5.grid(row=5, column=0)
            self.dis5 = Label(self.LoginFrame, font="Helvetica 14 bold", bg="cadet blue", bd=2, text=t[0][5])
self.dis5.grid(row=5, column=1)

#root.mainloop()
```

Employee_form.py

```
from tkinter import * import
tkinter.messagebox from tkinter
import ttk from tkinter import
font import mysql.connector
conn = mysql.connector.connect(
    host="localhost",
user="root",     password="",
    database = "hospital"
)
#root = Tk()
cursor=conn.cursor()
```

```
print("DATABASE CONNECTION SUCCESSFUL")


#PATIENT FORM     class
Employee:     def
__init__(self,master):
self.master = master
        self.master.title("HOSPITAL MANAGEMENT SYSTEM")
        self.master.geometry("1600x800+0+0")
self.master.config(bg="cadet blue")         self.frame =
Frame(self.master,bg="cadet blue")
self.frame.pack()

        #=============ATTRIBUTES===========

        self.emp_ID=StringVar()
self.emp_name=StringVar()
self.emp_sex=StringVar()         self.emp_age=IntVar()
self.emp_type=StringVar()
self.emp_salary=IntVar()
self.emp_exp=StringVar()
self.emp_email=StringVar()
        self.emp_phno=IntVar()


        #===============TITLE==========
        self.lblTitle = Label(self.frame,text = "EMPLOYEE REGISTRATION FORM", font="Helvetica 20
bold",bg="cadet blue")
        self.lblTitle.grid(row =0 ,column = 0,columnspan=2,pady=50)
        #==============FRAME==========
        self.LoginFrame = Frame(self.frame,width=400,height=80,relief="ridge",bg="cadet blue",bd=20)
self.LoginFrame.grid(row=1,column=0)

        self.LoginFrame2 = Frame(self.frame,width=400,height=80,relief="ridge",bg="cadet blue",bd=20)
self.LoginFrame2.grid(row=2,column=0)         #===========LABELS=============
        self.lblempid = Label(self.LoginFrame,text="EMPLOYEE ID",font="Helvetica 14 bold",bg="cadet
blue",bd=22)
        self.lblempid.grid(row=0,column=0)
        self.lblempid  = Entry(self.LoginFrame,font="Helvetica 14 bold",bd=2,textvariable= self.emp_ID)
self.lblempid.grid(row=0,column=1)

        self.lblempname = Label(self.LoginFrame,text="EMPLOYEE NAME",font="Helvetica 14 bold",bg="cadet
blue",bd=22)
        self.lblempname.grid(row=1,column=0)
        self.lblempname  = Entry(self.LoginFrame,font="Helvetica 14 bold",bd=2,textvariable= self.emp_name)
self.lblempname.grid(row=1,column=1)

        self.lblsex = Label(self.LoginFrame,text="SEX",font="Helvetica 14 bold",bg="cadet blue",bd=22)
self.lblsex.grid(row=2,column=0)
        self.etype1 =Entry(self.LoginFrame,font="Helvetica 14 bold",bd=2,textvariable= self.emp_sex)
self.etype1.grid(row=2,column=1)
```

```
        self.lblage = Label(self.LoginFrame,text="AGE",font="Helvetica 14 bold",bg="cadet blue",bd=22)
self.lblage.grid(row=3,column=0)
        self.lblage  = Entry(self.LoginFrame,font="Helvetica 14 bold",bd=2,textvariable= self.emp_age)
self.lblage.grid(row=3,column=1)

        self.etype1=Label(self.LoginFrame,text="EMPLOYEE DESIGNATION
[DOCTOR,NURSE,RECEPTIONIST] ",font="Helvetica 14 bold",bg="cadet blue",bd=22)
self.etype1.grid(row=4,column=0)
        self.etype1 =Entry(self.LoginFrame,font="Helvetica 14 bold",bd=2,textvariable= self.emp_type)
self.etype1.grid(row=4,column=1)

        self.lblCon = Label(self.LoginFrame,text="SALARY",font="Helvetica 14 bold",bg="cadet blue",bd=22)
self.lblCon.grid(row=0,column=2)
        self.lblCon  = Entry(self.LoginFrame,font="Helvetica 14 bold",bd=2,textvariable= self.emp_salary)
self.lblCon.grid(row=0,column=3)

        self.lblAlt = Label(self.LoginFrame,text="EXPERIENCE",font="Helvetica 14 bold",bg="cadet blue",bd=22)
self.lblAlt.grid(row=1,column=2)
        self.lblAlt  = Entry(self.LoginFrame,font="Helvetica 14 bold",bd=2,textvariable= self.emp_exp)
self.lblAlt.grid(row=1,column=3)

        self.lbleid = Label(self.LoginFrame,text="CONTACT NUMBER",font="Helvetica 14 bold",bg="cadet
blue",bd=22)
        self.lbleid.grid(row=2,column=2)
        self.lbleid  = Entry(self.LoginFrame,font="Helvetica 14 bold",bd=2,textvariable= self.emp_phno)
self.lbleid.grid(row=2,column=3)

        self.lbleid = Label(self.LoginFrame,text="EMAIL",font="Helvetica 14 bold",bg="cadet blue",bd=22)
self.lbleid.grid(row=3,column=2)
        self.lbleid  = Entry(self.LoginFrame,font="Helvetica 14 bold",bd=2,textvariable= self.emp_email)
self.lbleid.grid(row=3,column=3)

        self.button2 = Button(self.LoginFrame2, text="SAVE",width =10,font="Helvetica 14 bold",bg="cadet
blue",command = self.INSERT_EMP)
        self.button2.grid(row=3,column=1)

        self.button3 = Button(self.LoginFrame2, text="DELETE",width =10,font="Helvetica 14 bold",bg="cadet
blue",command= self.DE_DISPLAY)
        self.button3.grid(row=3,column=2)

        self.button6 = Button(self.LoginFrame2, text="EXIT",width =10,font="Helvetica 14 bold",bg="cadet
blue",command = self.Exit)
        self.button6.grid(row=3,column=3)
      def Exit(self):
self.master.destroy()

    def INSERT_EMP(self):
        global e1, e2, e3, e4, e5, e6, e7, e8, e9

        e1 = self.emp_ID.get()
e2 = self.emp_name.get()
e3 = self.emp_sex.get()
e4 = self.emp_age.get()
```

```
e5 = self.emp_type.get()
e6 = self.emp_salary.get()
e7 = self.emp_exp.get()
e8 = self.emp_email.get()
        e9 = self.emp_phno.get()

        conn = mysql.connector.connect(
            host="localhost",
user="root",            password="",
            database="hospital"
        )
        cursor = conn.cursor()

        cursor.execute("SELECT * FROM employee WHERE EMP_ID = %s", (e1,))
p = cursor.fetchall()         x = len(p)
          if x !=
0:
            tkinter.messagebox.showerror("HOSPITAL DATABASE SYSTEM", "EMPLOYEE ID ALREADY
EXISTS")         else:
            cursor.execute("INSERT INTO employee VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s)",(e1, e2, e3, e4,
e5, e6, e7, e8, e9))
            tkinter.messagebox.showinfo("HOSPITAL DATABASE SYSTEM", "EMPLOYEE DATA ADDED")
conn.commit()

    def DE_DISPLAY(self):
self.newWindow = Toplevel(self.master)
        self.app = D_EMP(self.newWindow)


class D_EMP:     def
__init__(self,master):
        global de1_emp,de
        self.master = master
        self.master.title("HOSPITAL MANAGEMENT SYSTEM")
        self.master.geometry("1600x800+0+0")
self.master.config(bg="cadet blue")         self.frame =
Frame(self.master,bg="cadet blue")
self.frame.pack()         self.de1_emp=StringVar()
        self.lblTitle = Label(self.frame,text = "DELETE EMPLOYEE WINDOW", font="Helvetica 20 bold",bg="cadet
blue")
        self.lblTitle.grid(row =0 ,column = 0,columnspan=2,pady=50)
        #==============FRAME==========
        self.LoginFrame = Frame(self.frame,width=400,height=80,relief="ridge",bg="cadet blue",bd=20)
self.LoginFrame.grid(row=1,column=0)
        self.LoginFrame2 = Frame(self.frame,width=400,height=80,relief="ridge",bg="cadet blue",bd=20)
self.LoginFrame2.grid(row=2,column=0)         #===========LABELS=============
        self.lblpatid = Label(self.LoginFrame,text="ENTER EMPLOYEE ID TO DELETE",font="Helvetica 14
bold",bg="cadet blue",bd=22)         self.lblpatid.grid(row=0,column=0)
        self.lblpatid= Entry(self.LoginFrame,font="Helvetica 14 bold",bd=2,textvariable= self.de1_emp)
self.lblpatid.grid(row=0,column=1)

        self.DeleteB = Button(self.LoginFrame2, text="DELETE",width =10,font="Helvetica 14 bold",bg="cadet
blue",command = self.DELETE_EMP)
```

```
        self.DeleteB.grid(row=3,column=1)

    def DELETE_EMP(self):
        global inp_d
        inp_d = str(self.de1_emp.get())

        conn = mysql.connector.connect(
            host="localhost",
user="root",            password="",
            database="hospital"
        )
        cursor = conn.cursor()

        cursor.execute("SELECT * FROM employee WHERE EMP_ID = %s", (inp_d,))
p = cursor.fetchall()

        if len(p) != 0:
            cursor.execute("DELETE FROM employee WHERE EMP_ID = %s", (inp_d,))
            tkinter.messagebox.showinfo("HOSPITAL DATABASE SYSTEM", "EMPLOYEE DATA DELETED") else:
            tkinter.messagebox.showerror("HOSPITAL DATABASE SYSTEM", "EMPLOYEE DATA DOESN'T
EXIST")

        conn.commit()

#root.mainloop()
```

Room_form.py

```
from tkinter import * import
tkinter.messagebox from tkinter
import ttk from tkinter import
font import mysql.connector
conn = mysql.connector.connect(
    host="localhost",
user="root",     password="",
    database = "hospital"
)
#root        =        Tk()
cursor=conn.cursor()
class Room:

    def __init__(self,master):

        self.master = master
        self.master.title("HOSPITAL MANAGEMENT SYSTEM")
        self.master.geometry("1600x800+0+0")
self.master.config(bg="cadet blue")         self.frame =
```

```
Frame(self.master,bg="cadet blue")
self.frame.pack()

    #=============ATTRIBUTES===========
self.P_id=IntVar()        self.room_t=StringVar()
self.room_no=StringVar()        self.rate=IntVar()
self.da=StringVar()
    self.dd=StringVar()

    #===============TITLE==========
    self.lblTitle = Label(self.frame,text = "ROOM ALLOCATION FORM", font="Helvetica 20 bold",bg="cadet
blue")
    self.lblTitle.grid(row =0 ,column = 0,columnspan=2,pady=50)
    #==============FRAME========== self.LoginFrame =
    Frame(self.frame,width=400,height=80,relief="ridge",bg="cadet blue",bd=20)
    self.LoginFrame.grid(row=1,column=0)

    self.LoginFrame2 = Frame(self.frame,width=400,height=80,relief="ridge",bg="cadet blue",bd=20)
    self.LoginFrame2.grid(row=2,column=0)
    #===========LABELS=============                    self.lblpatid =
Label(self.LoginFrame,text="PATIENT ID",font="Helvetica 14 bold",bg="cadet blue",bd=22)
self.lblpatid.grid(row=0,column=0)
    self.lblpatid  = Entry(self.LoginFrame,font="Helvetica 14 bold",bd=2,textvariable= self.P_id)
self.lblpatid.grid(row=0,column=1)        self.room_t1= Label(self.LoginFrame,text="ROOM TYPE\nSINGLE
ROOM: Rs 4500\nTWIN SHARING : Rs2500\nTRIPLE SHARING: Rs2000\n",font="Helvetica 14
bold",bg="cadet blue",bd=22)        self.room_t1.grid(row=1,column=0)
    self.room_t1 = Entry(self.LoginFrame,font="Helvetica 14 bold",bd=2,textvariable= self.room_t)
self.room_t1.grid(row=1,column=1)

    self.room_no1=Label(self.LoginFrame,text="ROOM NUMBER ",font="Helvetica 14 bold",bg="cadet
blue",bd=22)
    self.room_no1.grid(row=2,column=0)

    self.room_no1 = Entry(self.LoginFrame,font="Helvetica 14 bold",bd=2,textvariable= self.room_no)
self.room_no1.grid(row=2,column=1)
    self.lblrate=Label(self.LoginFrame,text="ROOM CHARGES",font="Helvetica 14 bold",bg="cadet
blue",bd=22)
    self.lblrate.grid(row=0,column=2)
    self.lblrate=Entry(self.LoginFrame,font="Helvetica 14 bold",bd=2,textvariable= self.rate)
self.lblrate.grid(row=0,column=3)
    self.lblda=Label(self.LoginFrame,text="DATE ADMITTED",font="Helvetica 14 bold",bg="cadet blue",bd=22)
    self.lblda.grid(row=1,column=2)
    self.lblda=Entry(self.LoginFrame,font="Helvetica 14 bold",bd=2,textvariable= self.da)
self.lblda.grid(row=1,column=3)
    self.lbldd=Label(self.LoginFrame,text="DATE DISCHARGED",font="Helvetica 14 bold",bg="cadet
blue",bd=22)
    self.lbldd.grid(row=2,column=2)
    self.lbldd=Entry(self.LoginFrame,font="Helvetica 14 bold",bd=2,textvariable= self.dd)
self.lbldd.grid(row=2,column=3)

    self.button2 = Button(self.LoginFrame2, text="SUBMIT",width =10,font="Helvetica 14 bold",bg="cadet
blue",command=self.INSERT_ROOM)
    self.button2.grid(row=3,column=1)
```

```
        self.button3 = Button(self.LoginFrame2, text="UPDATE",width =10,font="Helvetica 14 bold",bg="cadet
blue",command=self.UPDATE_ROOM)
        self.button3.grid(row=3,column=2)

        self.button5 = Button(self.LoginFrame2, text="ROOM DETAILS",width =15,font="Helvetica 14
bold",bg="cadet blue",command=self.SEARCH_ROOM)         self.button5.grid(row=3,column=4)

        self.button6 = Button(self.LoginFrame2, text="EXIT",width =10,font="Helvetica 14 bold",bg="cadet
blue",command = self.Exit)
        self.button6.grid(row=3,column=5)


    def clear(self):
        self.lblpatid.delete(0,'end')
        self.room_t1.delete(0,'end')
        self.room_no1.delete(0,'end')
        self.lblrate.delete(0,'end') self.lblda.delete(0,'end')
        self.lbldd.delete(0,'end')


    def INSERT_ROOM(self):
        global r1, r2, r3, r4, r5, r6, conn, p
conn = mysql.connector.connect(
            host="localhost",
user="root",            password="",
            database="hospital"
        )
        cursor = conn.cursor()
r1 = (self.P_id.get())         r2
= (self.room_t.get())         r3
= (self.room_no.get())
r4 = (self.rate.get())         r5 =
(self.da.get())
        r6 = (self.dd.get())

        cursor.execute("SELECT * FROM ROOM WHERE ROOM_NO = %s", (r3,))
p = cursor.fetchall()         x = len(p)
          if x !=
0:
            tkinter.messagebox.showerror("HOSPITAL DATABASE SYSTEM", "ROOM_NO IS CURRENTLY
OCCUPIED")         else:
            cursor.execute('INSERT INTO ROOM VALUES (%s, %s, %s, %s, %s, %s)', (r1, r3, r2, r4, r5, r6,))
tkinter.messagebox.showerror("HOSPITAL DATABASE SYSTEM", "ROOM ALLOCATED")            self.clear()
            conn.commit()

    def SEARCH_ROOM(self):
        self.newWindow= Toplevel(self.master)
        self.app = S_Room(self.newWindow)
      def Exit(self):
self.master.destroy()

    def UPDATE_ROOM(self):
```

```
        global P_id, r1, r2, room_t, da, dd, rate, room_no, r3, r4, r5, r6, conn
conn = mysql.connector.connect(
            host="localhost",
user="root",            password="",
            database="hospital"
        )
        cursor = conn.cursor()
r1 = (self.P_id.get())         r2
= (self.room_t.get())         r3
= (self.room_no.get()) r4 =
(self.rate.get()) r5 =
(self.da.get())
        r6 = (self.dd.get())

        cursor.execute("SELECT * FROM ROOM WHERE PATIENT_ID = %s AND ROOM_NO = %s", (r1, r3,)) p
        = cursor.fetchall()

        if len(p) != 0:
            tkinter.messagebox.showerror("HOSPITAL DATABASE SYSTEM", "PATIENT IS NOT ALLOCATED A
ROOM")         else:
            cursor.execute('UPDATE ROOM SET ROOM_NO=%s, ROOM_TYPE=%s, RATE=%s,
DATE_ADMITTED=%s, DATE_DISCHARGED=%s WHERE PATIENT_ID=%s',(r3, r2, r4, r5, r6, r1,))
tkinter.messagebox.showinfo("HOSPITAL DATABASE SYSTEM", "ROOM DETAILS UPDATED")
self.clear()
            conn.commit()

class    S_Room:                    def
__init__(self,master):
global inp_s,entry,SearchB
        self.master = master
        self.master.title("HOSPITAL MANAGEMENT SYSTEM")
        self.master.geometry("1600x800+0+0")
self.master.config(bg="cadet blue")         self.frame =
Frame(self.master,bg="cadet blue")
self.frame.pack()         self.Pr_id=IntVar()
        self.lblTitle = Label(self.frame,text = "SEARCH PATIENT DETAILS", font="Helvetica 20 bold",bg="cadet
blue")
        self.lblTitle.grid(row =0 ,column = 0,columnspan=2,pady=25)
        #==============FRAME==========
        self.LoginFrame = Frame(self.frame,width=400,height=80,relief="ridge",bg="cadet blue",bd=20)
self.LoginFrame.grid(row=1,column=0)
        self.LoginFrame2 = Frame(self.frame,width=400,height=80,relief="ridge",bg="cadet blue",bd=20)
self.LoginFrame2.grid(row=2,column=0)

        #===========LABELS=============
        self.lblpatid = Label(self.LoginFrame,text="ENTER PATIENT ID TO SEARCH",font="Helvetica 14
bold",bg="cadet blue",bd=22)         self.lblpatid.grid(row=0,column=0)
        self.lblpatid= Entry(self.LoginFrame,font="Helvetica 14 bold",bd=2,textvariable= self.Pr_id)
self.lblpatid.grid(row=0,column=1)

        self.SearchB = Button(self.LoginFrame2, text="SEARCH",width =10,font="Helvetica 14 bold",bg="cadet
blue",command = self.ROOM_DISPLAY)
        self.SearchB.grid(row=0,column=1)
```

```
    #FUNCTION FOR ROOM DISPLAY BUTTON

    def ROOM_DISPLAY(self):
        global pat_rm, lr1, dis1, lr2, dis2, c1, i, conn, c1, Pr_id
        conn = mysql.connector.connect(
            host="localhost",
user="root",            password="",
            database="hospital"
        )
        c1 = conn.cursor()
pat_rm = (self.Pr_id.get())
c1.execute('SELECT * FROM
ROOM WHERE
PATIENT_ID = %s',
(pat_rm,)) p = c1.fetchall()
        if len(p) == 0:
            tkinter.messagebox.showerror("HOSPITAL DATABASE SYSTEM", "PATIENT NOT ALLOCATED
ROOM")        else:
            c1.execute('SELECT PATIENT_ID, NAME, ROOM_NO, ROOM_TYPE FROM ROOM NATURAL JOIN
PATIENT WHERE PATIENT_ID = %s',(pat_rm,))
t = c1.fetchone()
            self.l1 = Label(self.LoginFrame, text="PATIENT ID", font="Helvetica 14 bold", bg="cadet blue", bd=22)
self.l1.grid(row=1, column=0)
            self.dis1 = Label(self.LoginFrame, font="Helvetica 14 bold", bd=2, bg="cadet blue", text=t[0])
self.dis1.grid(row=1, column=1)

            self.l2 = Label(self.LoginFrame, text="PATIENT NAME", font="Helvetica 14 bold", bg="cadet blue",
bd=22)
            self.l2.grid(row=2, column=0)
            self.dis2 = Label(self.LoginFrame, font="Helvetica 14 bold", bd=2, bg="cadet blue", text=t[1])
self.dis2.grid(row=2, column=1)

            self.l3 = Label(self.LoginFrame, text="ROOM NO", font="Helvetica 14 bold", bg="cadet blue", bd=22)
self.l3.grid(row=1, column=2)
            self.dis3 = Label(self.LoginFrame, font="Helvetica 14 bold", bg="cadet blue", bd=2, text=t[2])
self.dis3.grid(row=1, column=3)

            self.l4 = Label(self.LoginFrame, text="ROOM TYPE", font="Helvetica 14 bold", bg="cadet blue", bd=22)
self.l4.grid(row=2, column=2)
            self.dis4 = Label(self.LoginFrame, font="Helvetica 14 bold", bg="cadet blue", bd=2, text=t[3])
self.dis4.grid(row=2, column=3)


#root.mainloop()
```

SYSTEM TESTING

The following are some essential components of a Python software testing project for Hospital management:

1. Unit Testing: It includes evaluating distinct parts or units of code, such as methods or functions, to make sure they execute properly when used alone. For instance, you may test the procedures in charge of adding or removing menu items, figuring out costs, or changing bookings.

2. Integration Testing: This kind of testing is concerned with confirming that various software components interact and communicate properly. You would test how different modules, including menu management, order processing, and customer management, operate together in the Hospital management project.

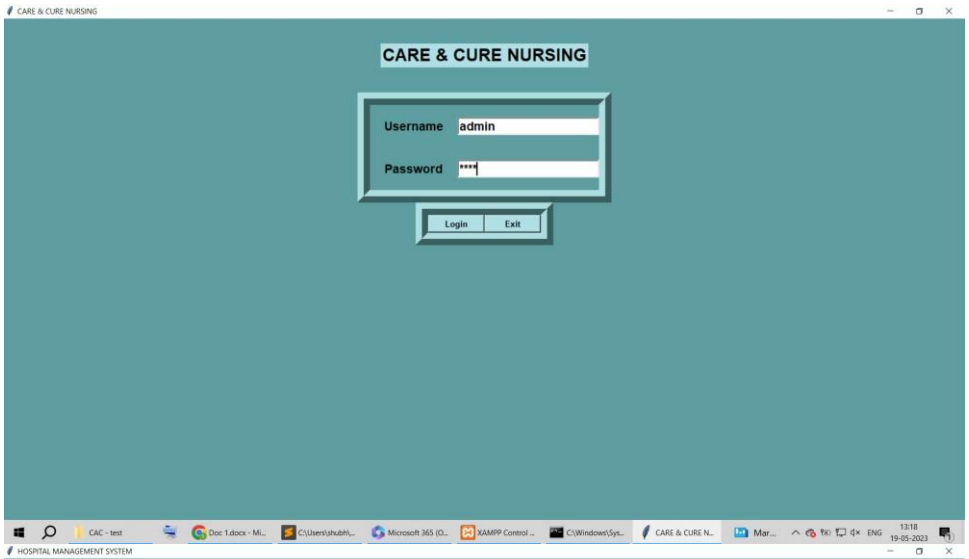
3. Functional testing: This entails evaluating the programme in light of its functionality. You would evaluate the software's ability to carry out all the anticipated tasks, including adding new menu items, taking orders, creating invoices, and handling reservations, in the context of the Hospital management mini project.

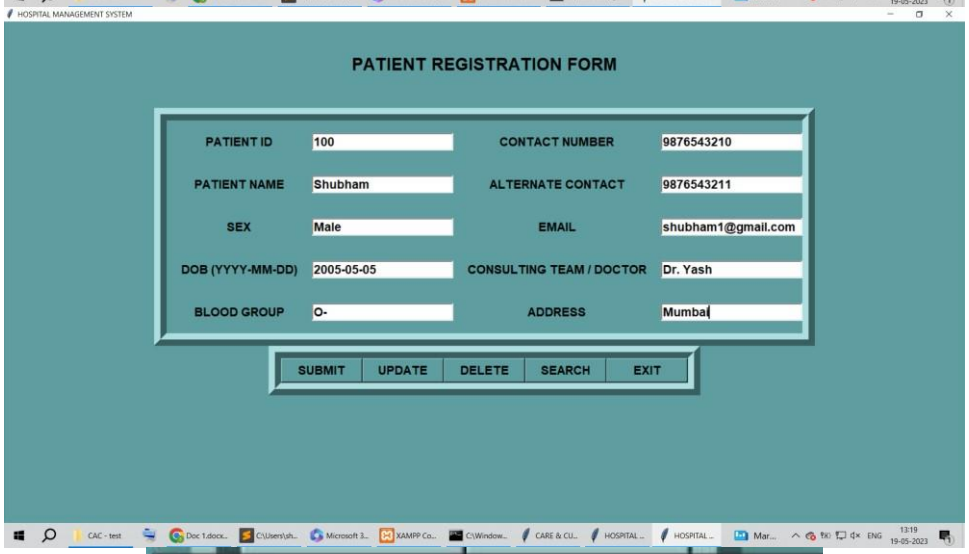
4. User Interface (UI) Testing: This sort of testing looks to see if the software's user interface is simple to use, intuitive, and reliable. You would examine if buttons, menus, forms, and other UI components behave as expected and meet your needs.
5. Performance testing: It involves determining how well the software performs and responds under various circumstances, such as managing a high volume of orders or concurrent users. You may test how the software manages a high number of concurrent orders or reservations for the Hospital management project.

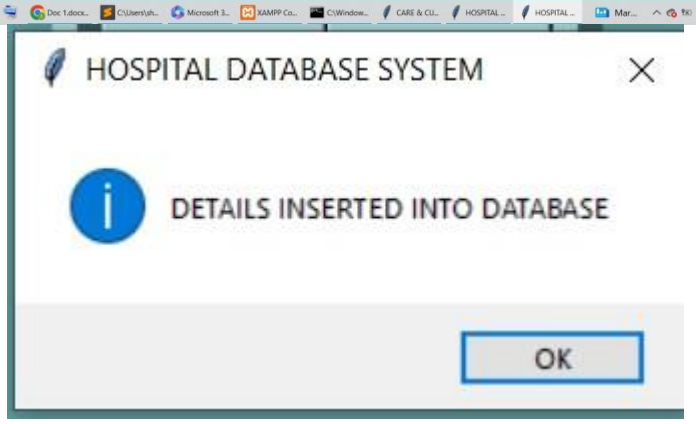
6. Security Testing: This category of testing is concerned with finding and fixing software security flaws. You may check, for instance, if the programme properly manages user authentication and authorisation, safeguards sensitive data, and restricts unauthorised access.

7. Regression testing: It is the process of retesting previously tested software components to make sure that subsequent additions or modifications haven't resulted in any new problems or broken functionality. This is essential each time you update.

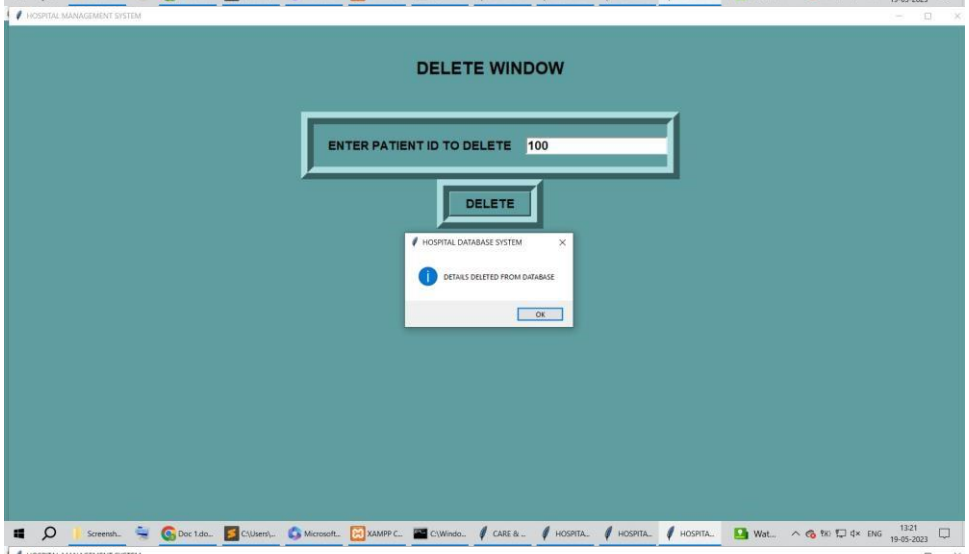
RESULTS


CARE & CURE NURSING
CARE & CURE NURSING
Username
admin
Password
Login
Exit


HOSPITAL MANAGEMENT SYSTEM
PATIENT REGISTRATION FORM
PATIENT ID
100
CONTACT NUMBER
9876543210
PATIENT NAME
Shubham
ALTERNATE CONTACT
9876543211
SEX
Male
EMAIL
shubham1@gmail.com
DOB (YYYY-MM-DD)
2005-05-05
CONSULTING TEAM / DOCTOR
Dr. Yash
BLOOD GROUP
O-
ADDRESS
Mumbai
SUBMIT
UPDATE
DELETE
SEARCH
EXIT


HOSPITAL DATABASE SYSTEM
DETAILS INSERTED INTO DATABASE
OK


HOSPITAL MANAGEMENT SYSTEM
DELETE WINDOW
ENTER PATIENT ID TO DELETE
100
DELETE


HOSPITAL MANAGEMENT SYSTEM
DELETE WINDOW
ENTER PATIENT ID TO DELETE
100
DELETE
HOSPITAL DATABASE SYSTEM
DETAILS DELETED FROM DATABASE
OK


HOSPITAL MANAGEMENT SYSTEM
SEARCH WINDOW
ENTER PATIENT ID TO SEARCH
10
SEARCH

HOSPITAL MANAGEMENT SYSTEM

SEARCH WINDOW

ENTER PATIENT ID TO SEARCH	10		
PATIENT ID	10	ADDRESS	jb
PATIENT NAME	shubham	CONSULTING TEAM / DOCTOR	dr. s
SEX	male	EMAIL	abc1@gmail.com
DOB (YYYY-MM-DD)	O+	CONTACT NUMBER	2147483647
BLOOD GROUP	2005-08-20	ALTERNATE CONTACT	2147483647

SEARCH

HOSPITAL MANAGEMENT SYSTEM

ROOM ALLOCATION FORM

PATIENT ID	10	ROOM CHARGES	2000
ROOM TYPE SINGLE ROOM: Rs 4500 TWIN SHARING : Rs2500 TRIPLE SHARING: Rs2000	triple	DATE ADMITTED	2005-05-05
ROOM NUMBER	150	DATE DISCHARGED	2006-05-05

SUBMIT | UPDATE | ROOM DETAILS | EXIT

HOSPITAL MANAGEMENT SYSTEM

SEARCH PATIENT DETAILS

ENTER PATIENT ID TO SEARCH 10

SEARCH

HOSPITAL MANAGEMENT SYSTEM

SEARCH PATIENT DETAILS

ENTER PATIENT ID TO SEARCH	10		
PATIENT ID	10	ROOM NO	150
PATIENT NAME	shubham	ROOM TYPE	triple

SEARCH

HOSPITAL MANAGEMENT SYSTEM

EMPLOYEE REGISTRATION FORM

EMPLOYEE ID	100	SALARY	50000
EMPLOYEE NAME	Manav	EXPERIENCE	2
SEX	male	CONTACT NUMBER	9876548914
AGE	18	EMAIL	manav@gmail.com
EMPLOYEE DESIGNATION [DOCTOR,NURSE,RECEPTIONIST]	Dr		

SAVE DELETE EXIT

HOSPITAL MANAGEMENT SYSTEM

DELETE EMPLOYEE WINDOW

ENTER EMPLOYEE ID TO DELETE

DELETE

HOSPITAL MANAGEMENT SYSTEM

APPOINTMENT FORM

PATIENT ID	10	APPOINTMENT TIME(HH:MM:SS)	10:10:10
DOCTOR ID	100	APPOINTMENT DATE(YYYY-MM-DD)	2005-05-05
APPOINTMENT NO	1000	DESCRIPTION	abc-xyz

SAVE | DELETE | SEARCH APPOINTMENTS | EXIT

HOSPITAL MANAGEMENT SYSTEM

BILLING WINDOW

PATIENT ID	10		
DATE DISCHARGED(YYYY-MM-DD)	2006-06-06		UPDATE DISCHARGE DATE
TREATMENT	abc	MEDICINE	xyz
TREATMENT CODE	102	MEDICINE QUANTITY	2
TREATMENT COST ₹	10000	MEDICINE PRICE ₹	1000

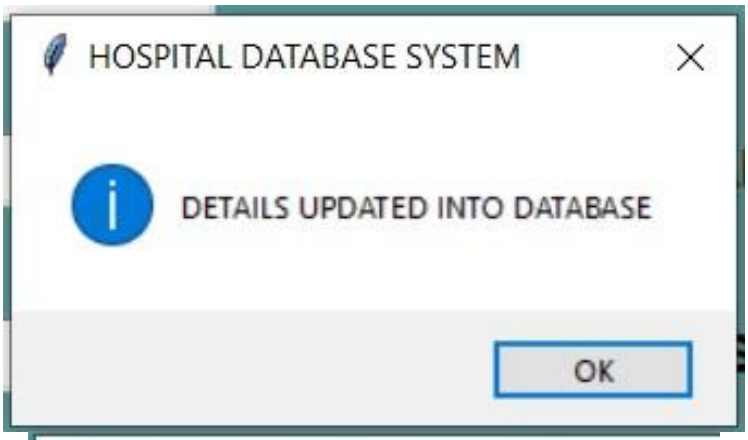
UPDATE DATA | GENERATE BILL | EXIT

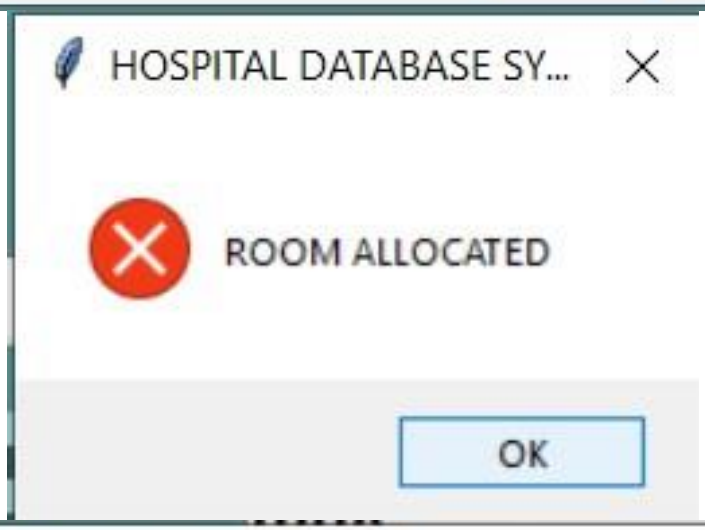
HOSPITAL MANAGEMENT SYSTEM

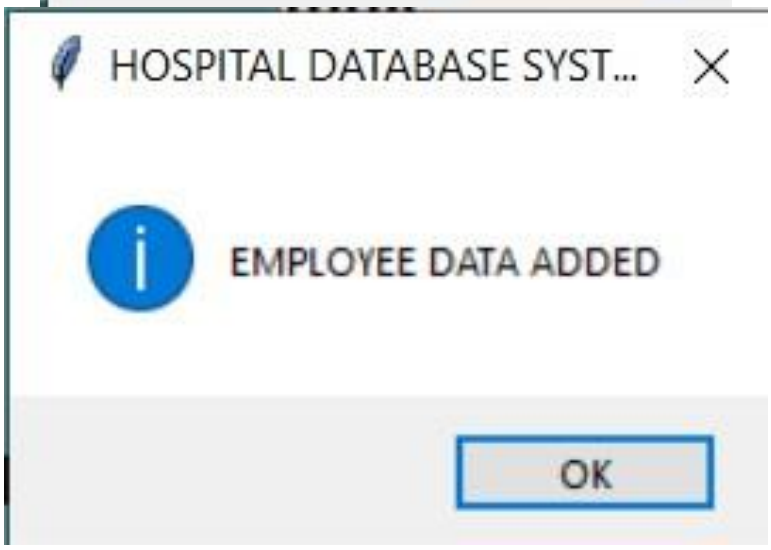
BILLING WINDOW

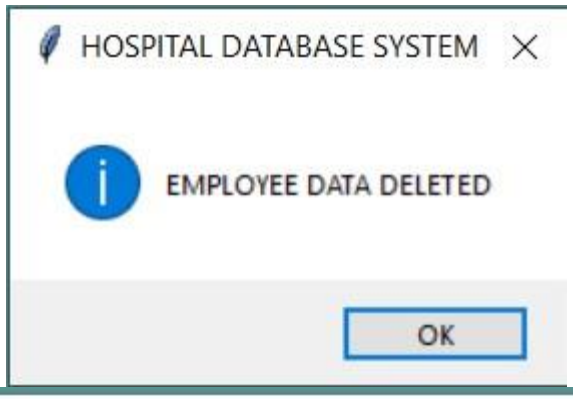
PATIENT ID	10		
DATE DISCHARGED(YYYY-MM-DD)	2006-06-06		UPDATE DISCHARGE DATE
TREATMENT	abc	MEDICINE	xyz
TREATMENT CODE	102	MEDICINE QUANTITY	2
TREATMENT COST ₹	10000	MEDICINE PRICE ₹	1000
TOTAL AMOUNT OUTSTANDING	-320988000		

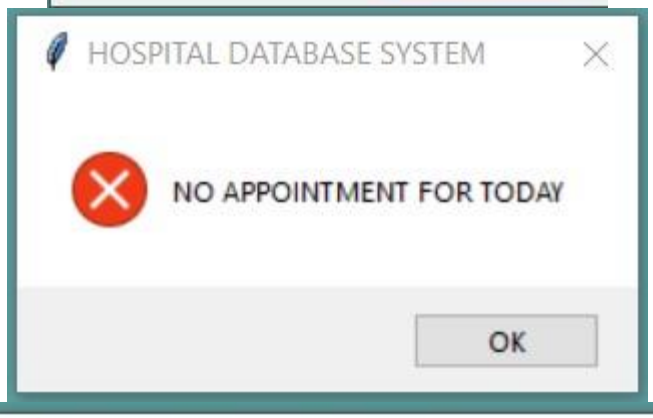
UPDATE DATA | GENERATE BILL | EXIT

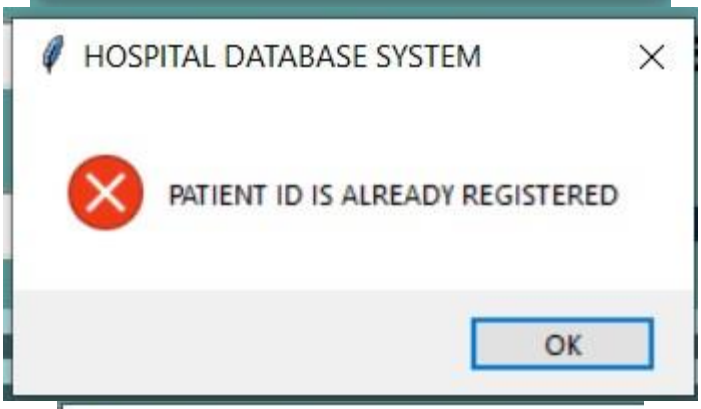

HOSPITAL DATABASE SYSTEM
DETAILS UPDATED INTO DATABASE
OK

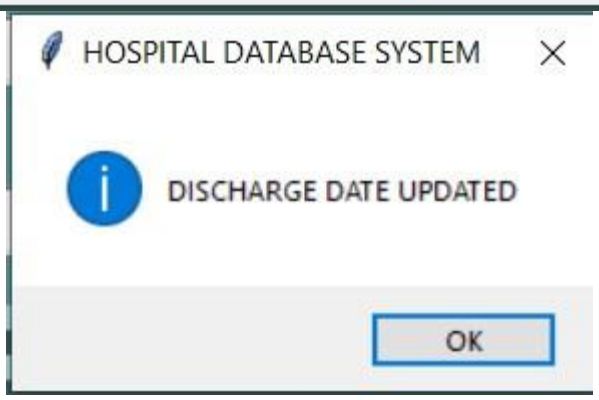

HOSPITAL DATABASE SY...
ROOM ALLOCATED
OK

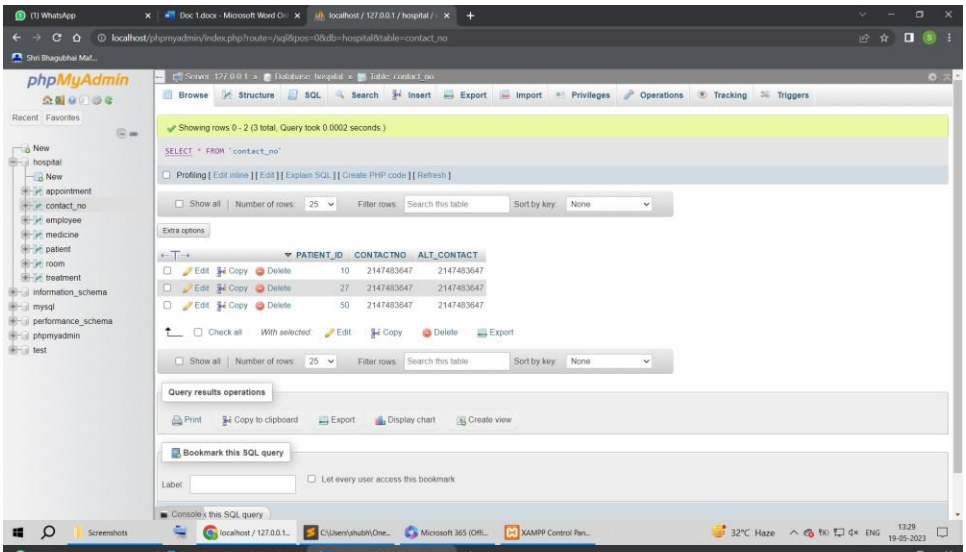

HOSPITAL DATABASE SYST...
EMPLOYEE DATA ADDED
OK

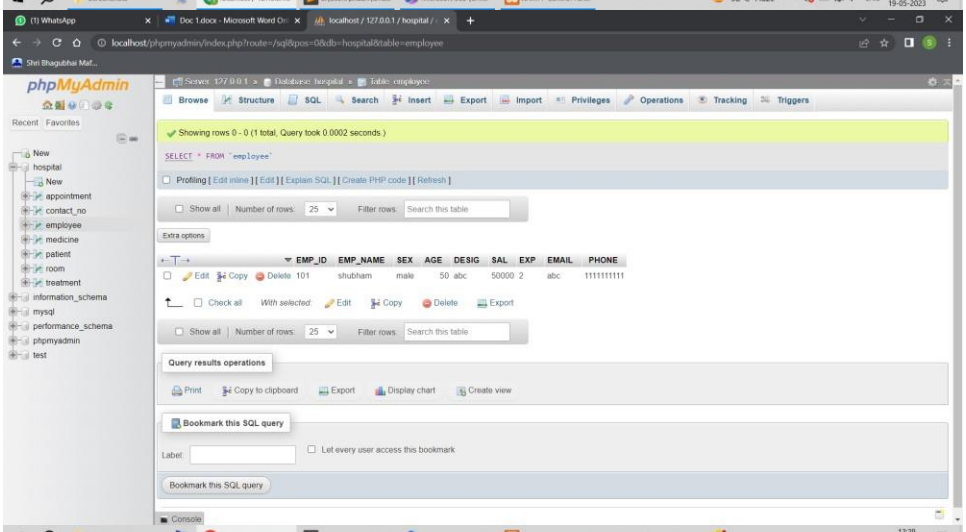

HOSPITAL DATABASE SYSTEM
EMPLOYEE DATA DELETED
OK

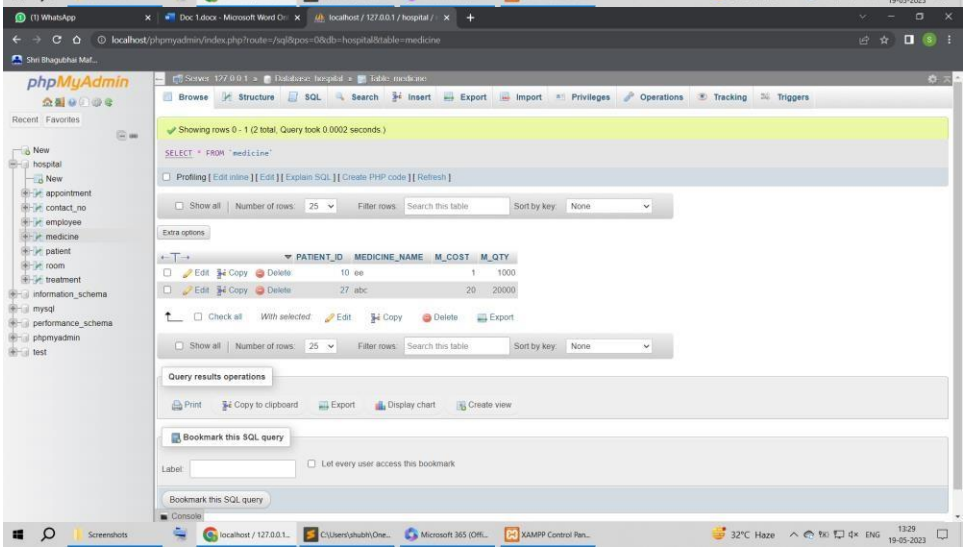

HOSPITAL DATABASE SYSTEM
NO APPOINTMENT FOR TODAY
OK

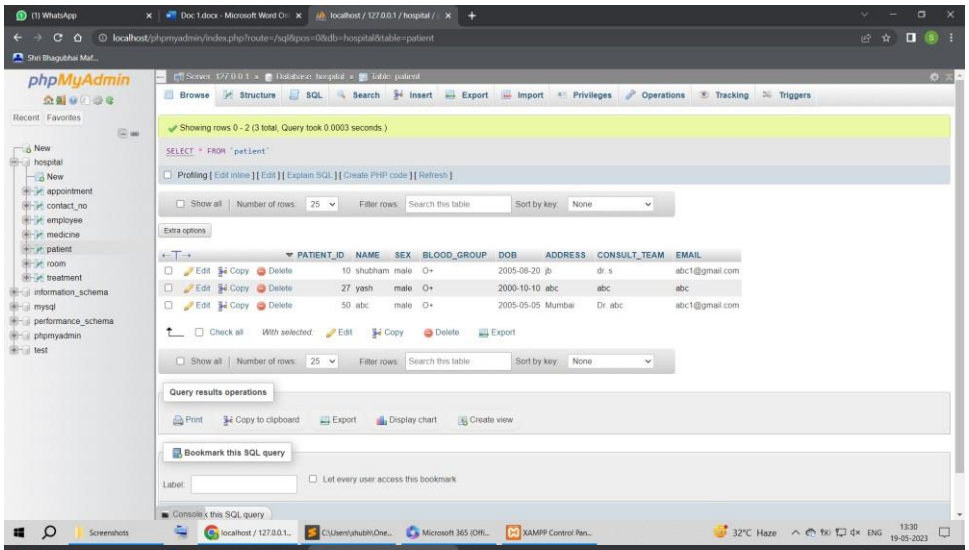

HOSPITAL DATABASE SYSTEM
PATIENT ID IS ALREADY REGISTERED
OK

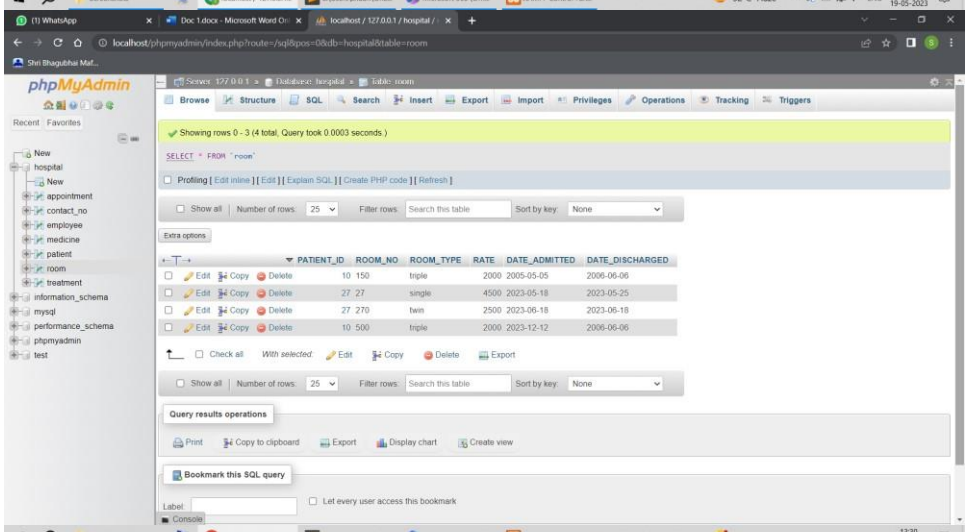

HOSPITAL DATABASE SYSTEM
DISCHARGE DATE UPDATED
OK

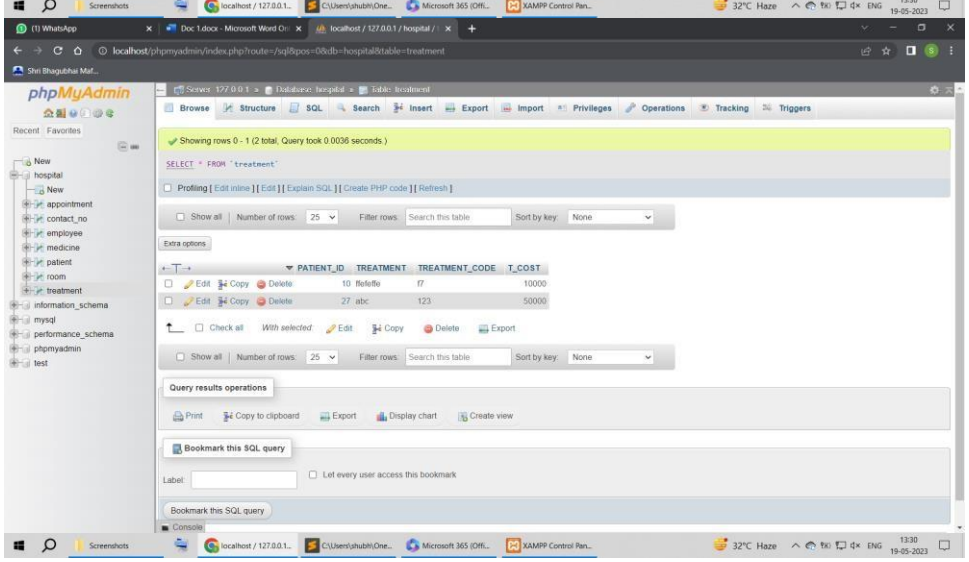

localhost/phpmyadmin/index.php?route=/sql&pos=0&db=hospital&table=contact_no
Server: 127.0.0.1 » Database: hospital » Table: contact_no
Showing rows 0 - 2 (3 total, Query took 0.0002 seconds.)
SELECT * FROM `contact_no`
PATIENT_ID CONTACTNO ALT_CONTACT
10 2147483647 2147483647
27 2147483647 2147483647
50 2147483647 2147483647


localhost/phpmyadmin/index.php?route=/sql&pos=0&db=hospital&table=employee
Server: 127.0.0.1 » Database: hospital » Table: employee
Showing rows 0 - 0 (1 total, Query took 0.0002 seconds.)
SELECT * FROM `employee`
EMP_ID EMP_NAME SEX AGE DESIG SAL EXP EMAIL PHONE
101 shubham male 50 abc 50000 2 abc 1111111111


localhost/phpmyadmin/index.php?route=/sql&pos=0&db=hospital&table=medicine
Server: 127.0.0.1 » Database: hospital » Table: medicine
Showing rows 0 - 1 (2 total, Query took 0.0002 seconds.)
SELECT * FROM `medicine`
PATIENT_ID MEDICINE_NAME M_COST M_QTY
10 ee 1 1000
27 abc 20 20000


localhost/phpmyadmin/index.php?route=/sql&pos=0&db=hospital&table=patient
Server: 127.0.0.1 » Database: hospital » Table: patient
Showing rows 0 - 2 (3 total, Query took 0.0003 seconds.)
SELECT * FROM `patient`
PATIENT_ID NAME SEX BLOOD_GROUP DOB ADDRESS CONSULT_TEAM EMAIL
10 shubham male O+ 2005-08-20 jb dr. s abc1@gmail.com
27 yash male O+ 2000-10-10 abc abc abc
50 abc male O+ 2005-05-05 Mumbai Dr. abc abc1@gmail.com


localhost/phpmyadmin/index.php?route=/sql&pos=0&db=hospital&table=room
Server: 127.0.0.1 » Database: hospital » Table: room
Showing rows 0 - 3 (4 total, Query took 0.0003 seconds.)
SELECT * FROM `room`
PATIENT_ID ROOM_NO ROOM_TYPE RATE DATE_ADMITTED DATE_DISCHARGED
10 150 triple 2000 2005-05-05 2006-06-06
27 27 single 4500 2023-05-18 2023-05-25
27 270 twin 2500 2023-06-18 2023-06-18
10 500 triple 2000 2023-12-12 2006-06-06


localhost/phpmyadmin/index.php?route=/sql&pos=0&db=hospital&table=treatment
Server: 127.0.0.1 » Database: hospital » Table: treatment
Showing rows 0 - 1 (2 total, Query took 0.0036 seconds.)
SELECT * FROM `treatment`
PATIENT_ID TREATMENT TREATMENT_CODE T_COST
10 fefefefe f7 10000
27 abc 123 50000

FUTURE SCOPE

The features that could not be added are array of objects and Multi-Threading. In the near future these concepts will be added in an appropriate way.

CONCLUSION

From this Mini Project I learned the various Python concept like Inheritance, Exception Handling, Packages, basic Class and objects, Basic Object Oriented Programming concepts, IO handling, File handling, GUI (Tkinter), Database Connectivity (MySQL) and String Functions. This helped me to strengthen the core Python concepts.

REFERENCES

I. https://www.javatpoint.com
II. https://www.geeksforgeeks.org
III. https://stackoverflow.com/questions/22492118/payroll-program-for-pythonusing-multiple-functions-and-return-function
IV. https://chat.openai.com/
V. https://github.com
VI. Python Essential Reference.