Name : ujwal sahu

Roll no. :

10 Div : B

Branch : B.Voc AIDS

Experiment 12 :

Subject : Python Mini Project

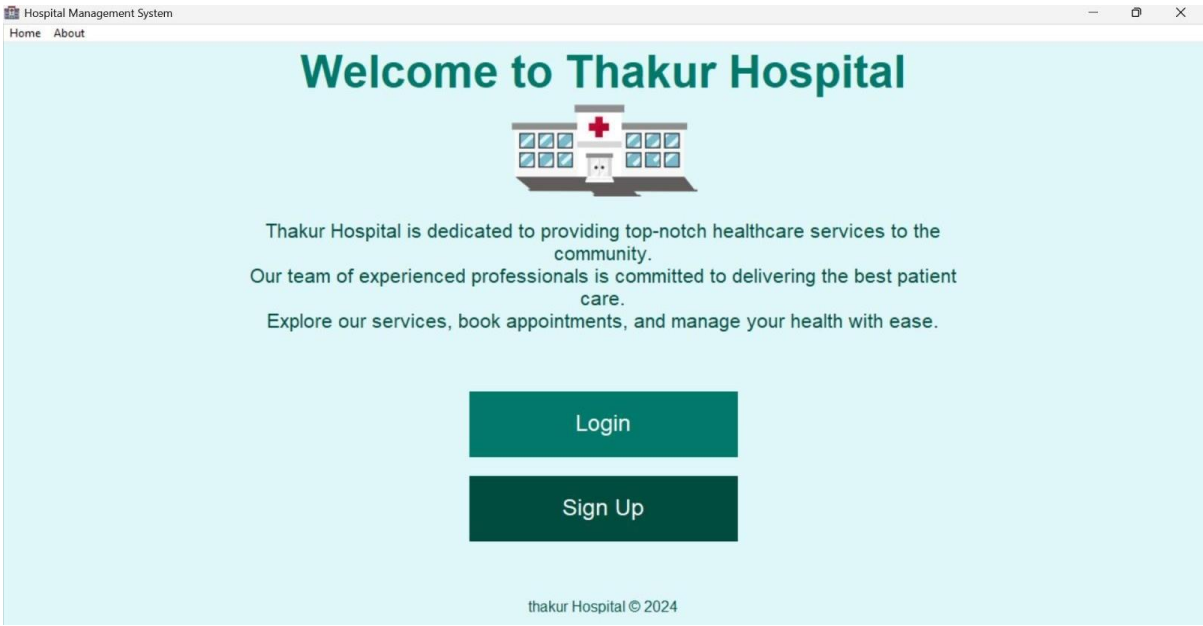
Topic : Hospital Management system

Introduction : The Hospital Management System is a desktop application built using Python's tkinter library for the GUI and SQLite for the database. It allows patients to sign up, log in, and submit their personal and medical details, while doctors can log in to view and manage appointments. The application is designed with two primary user roles: patients and doctors.

Workflow of the Project:

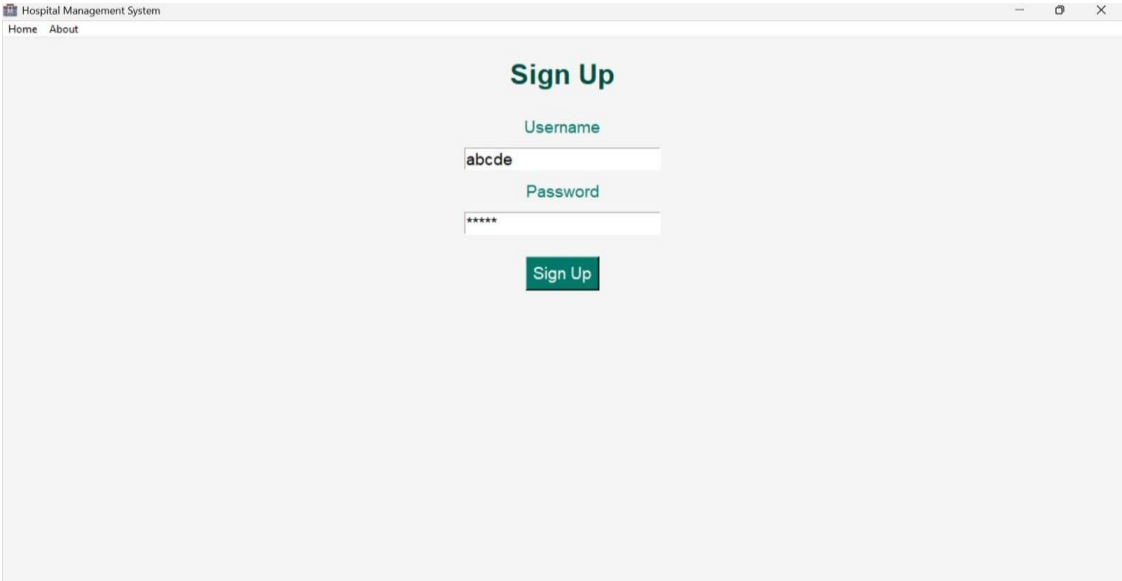
1. **Application Launch:**

   - When the application starts, it opens the **Intro Page** where users can choose to sign up, log in, or learn more about the hospital through the About Page.



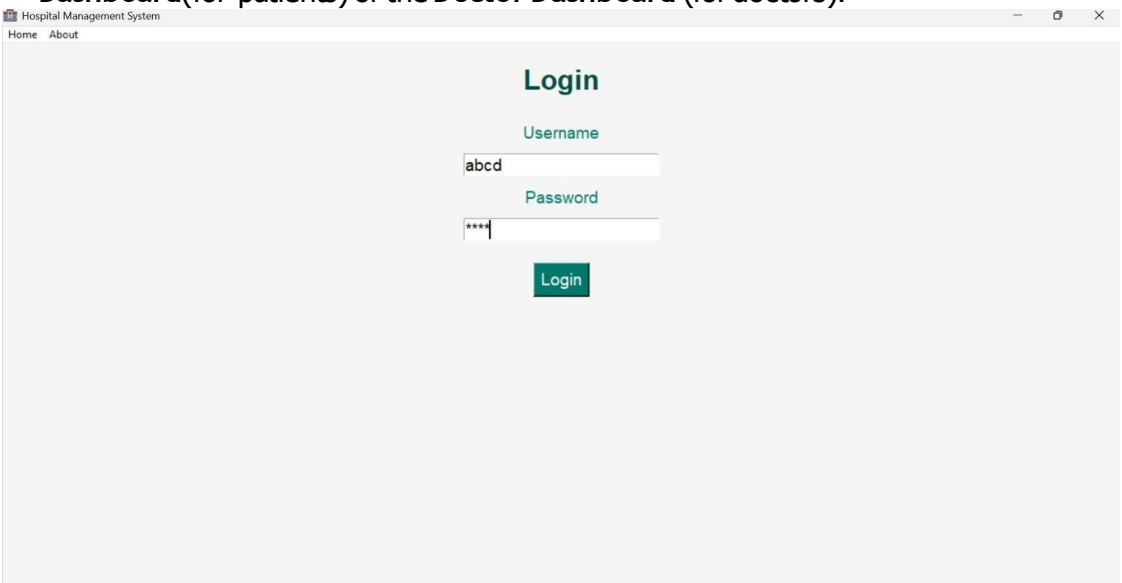
2. User Sign-Up:

- New users (patients) sign up through the **SignUp Page** by providing a username and password.
- The credentials are stored in the `users` table in the database, with the `user_type` set to "patient".
- After signing up, users are redirected to the login page.



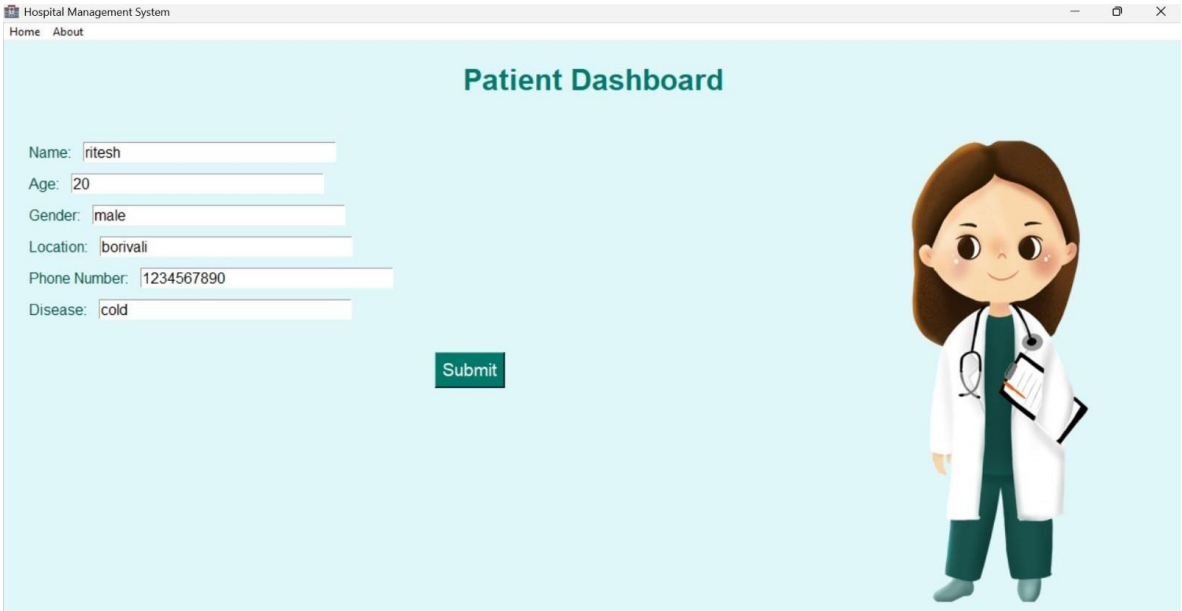
3. User Login:

- Users log in through the **Login Page**.
- The system checks the `users` table to verify the credentials.
- Based on the `user_type`, the user is either redirected to the **Patient Dashboard** (for patients) or the **Doctor Dashboard** (for doctors).



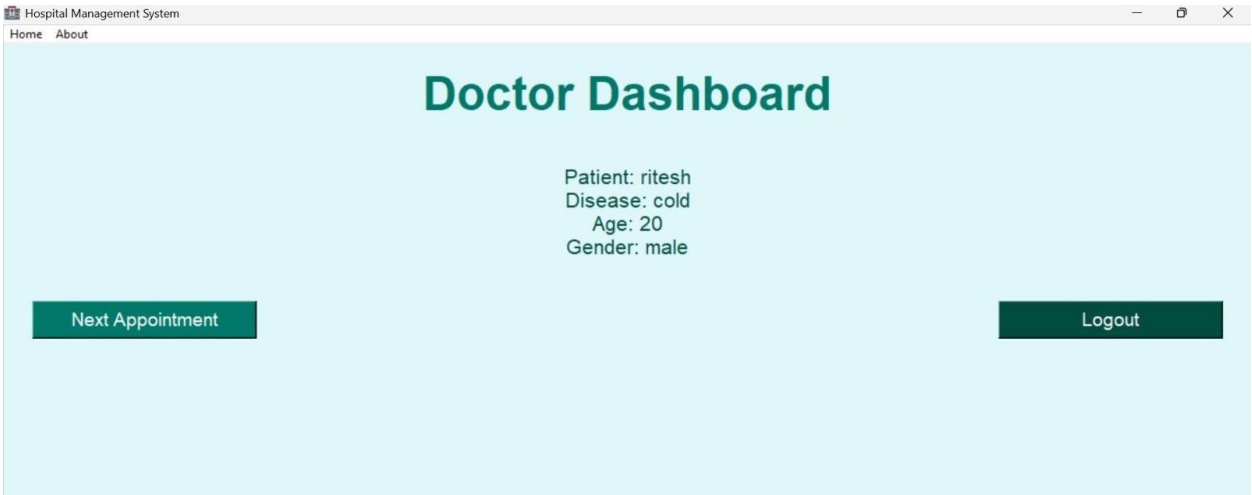
4. Patient Dashboard:

- Patients can enter their personal details like name, age, gender, location, phone number, and disease.
- These details are stored in the `patients` table in the database.
- Once the patient submits their details, they are saved and can be accessed later by the doctor.



5. Doctor Dashboard:

- The doctor logs in and is directed to the **Doctor Dashboard**.
- The doctor can view appointments (patient data) one by one.
- The system fetches patient details from the `patients` table, displaying information such as the patient's name, age, gender, and disease.
- The doctor can navigate through appointments using the "Next Appointment" button.



6. Menu and Navigation:

- The application features a menu bar with "Home" and "About" options on every page.
- The "Home" option returns the user to the **Intro Page**.

CODE : import tkinter as tk
from tkinter import messagebox
import sqlite3
from PIL import Image, ImageTk

```
#Connect to the SQLite database
conn =sqlite3.connect("hospital.db")
cursor =conn.cursor()

#Create tables if they don't exist
cursor.execute('''
  CREATE TABLE IF NOT EXISTS users (
    username TEXT PRIMARY KEY,
    password TEXT,
    user_type TEXT
  )
''')

cursor.execute('''
    CREATE TABLE IF NOT EXISTS patients (
      id INTEGER PRIMARY KEY AUTOINCREMENT,
      name TEXT,
      age INTEGER,
      gender TEXT,
      location TEXT,
      phone_number TEXT,
      disease TEXT -- New field for disease
    )
    ''')



#Insert default doctor credentials (if not already exists)
cursor.execute('''
  INSERT OR IGNORE INTO users (username, password, user_type) VALUES (?, ?, ?)
''', ("doc", "doc", "doctor"))

conn.commit()



class HospitalManagementSystem(tk.Tk):
  def__init__(self):
    super().__init__()

    self.title("Hospital Management System")
    self.geometry("400x300")

    #Set the application icon (favicon)
    self.iconbitmap('hospital.ico')  #Adjust the path to your favicon file
```

```
        container =tk.Frame(self)
        container.pack(side="top", fill="both", expand=True)
        container.grid_rowconfigure(0, weight=1)
        container.grid_columnconfigure(0, weight=1)

        self.frames ={}
        for F in (IntroPage, SignUpPage, LoginPage, PatientDashboard, DoctorDashboard,
      AboutPage):
            page_name =F.__name
            frame =F(parent=container, controller=self)
            self.frames[page_name] =frame
            frame.grid(row=0, column=0, sticky="nsew")

        self.show_frame("IntroPage")

    def show_frame(self, page_name):
        frame =self.frames[page_name]
        frame.tkraise()

    def create_menu(self):
        menu =tk.Menu(self)
        self.config(menu=menu)

        home_menu =tk.Menu(menu)
      menu.add_cascade(label="Home", menu=home_menu)
      home_menu.add_command(label="Home", command=lambda:
      self.show_frame("IntroPage"))

        about_menu =tk.Menu(menu)
      menu.add_cascade(label="About", menu=about_menu)
      about_menu.add_command(label="About", command=lambda:
      self.show_frame("AboutPage"))


class IntroPage(tk.Frame):
    def __init__(self, parent, controller):
        super()._init_(parent)
        self.controller =controller

        self.controller.create_menu()

        #Background color
        self.configure(bg="#e0f7fa")

        #Title Label
        title_label =tk.Label(self, text="Welcome to Thakur Hospital", font=('Helvetica', 36,
      'bold'), bg="#e0f7fa", fg="#00796b")
        title_label.pack(pady=0)

        #Hospital Image
        img =Image.open('hospital.png')  #Adjust the path to your image file
        resized =img.resize((200, 100))  #Resize to appropriate dimensions
```

```
        self.hospital_image =ImageTk.PhotoImage(resized) #Convert to PhotoImage
        self.hospital_image_label =tk.Label(self, image=self.hospital_image, bg="#e0f7fa")
        self.hospital_image_label.pack(pady=0)

        #Hospital Description
        description =(
            "Thakur Hospital is dedicated to providing top-notch healthcare services to the
    community.\n"
            "Our team of experienced professionals is committed to delivering the best patient
    care.\n"
            "Explore our services, book appointments, and manage your health with ease."
        )
        description_label =tk.Label(self, text=description, font=('Helvetica', 16), bg="#e0f7fa",
    fg="#004d40", wraplength=800, justify="center")
        description_label.pack(pady=20, padx=20)

        #Buttons
        button_frame =tk.Frame(self, bg="#e0f7fa")
        button_frame.pack(pady=30)

        tk.Button(button_frame, text="Login", font=('Helvetica', 18), width=20, height=2,
    bg="#00796b", fg="white", borderwidth=0, relief="flat", command=lambda:
    controller.show_frame("LoginPage")).pack(pady=10)
        tk.Button(button_frame, text="Sign Up", font=('Helvetica', 18), width=20, height=2,
    bg="#004d40", fg="white", borderwidth=0, relief="flat", command=lambda:
    controller.show_frame("SignUpPage")).pack(pady=10)

        #Footer
        footer_frame =tk.Frame(self, bg="#e0f7fa")
        footer_frame.pack(side="bottom", pady=20)

        tk.Label(footer_frame, text="thakur Hospital ©2024", font=('Helvetica', 12),
    bg="#e0f7fa", fg="#004d40").pack()

        #Ensure the layout resizes correctly
        self.pack(fill="both", expand=True)


class SignUpPage(tk.Frame):
    def __init__(self, parent, controller):
        super()._init_(parent)
        self.controller =controller

        self.controller.create_menu()

        #Background color
        self.configure(bg="#f5f5f5")

        #Title
        tk.Label(self, text="Sign Up", font=('Helvetica', 24, 'bold'), bg="#f5f5f5",
    fg="#004d40").pack(pady=20)
```

```
        #Username Entry
        tk.Label(self, text="Username", font=('Helvetica', 14), bg="#f5f5f5",
      fg="#00796b").pack(pady=5)
        self.username_entry =tk.Entry(self, font=('Helvetica', 14))
        self.username_entry.pack(pady=5)

        #Password Entry
        tk.Label(self, text="Password", font=('Helvetica', 14), bg="#f5f5f5",
      fg="#00796b").pack(pady=5)
        self.password_entry =tk.Entry(self, show="*", font=('Helvetica', 14))
        self.password_entry.pack(pady=5)

        #Sign Up Button
        tk.Button(self, text="Sign Up", font=('Helvetica', 14), bg="#00796b", fg="white",
      command=self.sign_up).pack(pady=20)

    def sign_up(self):
        username =self.username_entry.get()
        password =self.password_entry.get()

        try:
            cursor.execute("INSERT INTO users (username, password, user_type) VALUES
      (?, ?, ?)", (username, password, "patient"))
            conn.commit()
            messagebox.showinfo("Success", "Sign Up Successful! Redirecting to Home Page.")
            self.controller.show_frame("IntroPage")
        except sqlite3.IntegrityError:
            messagebox.showerror("Error", "Username already exists. Please choose a different
      username.")


class LoginPage(tk.Frame):
    def __init__(self, parent, controller):
        super()._init_(parent)
        self.controller =controller

        self.controller.create_menu()

        #Background color
        self.configure(bg="#f5f5f5")

        #Title
        tk.Label(self, text="Login", font=('Helvetica', 24, 'bold'), bg="#f5f5f5",
      fg="#004d40").pack(pady=20)

        #Username Entry
        tk.Label(self, text="Username", font=('Helvetica', 14), bg="#f5f5f5",
      fg="#00796b").pack(pady=5)
        self.username_entry =tk.Entry(self, font=('Helvetica', 14))
        self.username_entry.pack(pady=5)

        #Password Entry
```

```
        tk.Label(self, text="Password", font=('Helvetica', 14), bg="#f5f5f5",
      fg="#00796b").pack(pady=5)
        self.password_entry =tk.Entry(self, show="*", font=('Helvetica', 14))
        self.password_entry.pack(pady=5)

        #Login Button
        tk.Button(self, text="Login", font=('Helvetica', 14), bg="#00796b", fg="white",
      command=self.check_login).pack(pady=20)

    def check_login(self):
        username =self.username_entry.get()
        password =self.password_entry.get()

        cursor.execute("SELECT * FROM users WHERE username=? AND password=?",
      (username, password))
        result =cursor.fetchone()

        if result:
            user_type =result[2] #"patient" or "doctor"
            if user_type =="patient":
                self.controller.show_frame("PatientDashboard")
            else:
                self.controller.show_frame("DoctorDashboard")
        else:
            messagebox.showerror("Error", "Invalid Credentials")


class PatientDashboard(tk.Frame):
    def __init__(self, parent, controller):
        super()._init_(parent)
        self.controller =controller

        self.controller.create_menu()

        #Background color
        self.configure(bg="#e0f7fa")

        #Title
        tk.Label(self, text="Patient Dashboard", font=('Helvetica', 24, 'bold'), bg="#e0f7fa",
      fg="#00796b").pack(pady=20)

        #Frame for patient information
        info_frame =tk.Frame(self, bg="#e0f7fa")
        info_frame.pack(side="left", padx=20, pady=20, fill='y")

        #Patient Information Entries
        self.name_var =tk.StringVar()
        self.age_var =tk.StringVar()
        self.gender_var =tk.StringVar()
        self.location_var =tk.StringVar()
        self.phone_var =tk.StringVar()
        self.disease_var =tk.StringVar()  #New field for disease
```

```
  self.create_label_with_entry(info_frame, "Name:", self.name_var, 30).pack(pady=5,
anchor="w")
  self.create_label_with_entry(info_frame, "Age:", self.age_var, 30).pack(pady=5,
anchor="w")
  self.create_label_with_entry(info_frame, "Gender:", self.gender_var, 30).pack(pady=5,
anchor="w")
  self.create_label_with_entry(info_frame, "Location:", self.location_var,
30).pack(pady=5, anchor='w")
  self.create_label_with_entry(info_frame, "Phone Number:", self.phone_var,
30).pack(pady=5, anchor='w")
self.create_label_with_entry(info_frame, "Disease:", self.disease_var,
30).pack(pady=5, anchor='w") #New field for disease

  #Submit Button
  submit_frame =tk.Frame(self, bg="#e0f7fa")
  submit_frame.pack(side='left", padx=20, pady=20)
  tk.Button(submit_frame, text="Submit", font=('Helvetica', 14), bg="#00796b",
fg="white", command=self.submit).pack(pady=20)

  #Add image on the right side
  self.image =Image.open('doctor.png') #Adjust path to your image file
  self.image =self.image.resize((350, 500)) #Resize to fit the layout
  self.photo =ImageTk.PhotoImage(self.image)

  self.image_label =tk.Label(self, image=self.photo, bg="#e0f7fa")
  self.image_label.pack(side='right", padx=20)

def create_label_with_entry(self, parent, label_text, text_variable, entry_width):
  frame =tk.Frame(parent, bg="#e0f7fa")
  label =tk.Label(frame, text=label_text, font=('Helvetica', 12), bg="#e0f7fa",
fg="#004d40")
  entry =tk.Entry(frame, textvariable=text_variable, width=entry_width, font=('Helvetica',
12))

  label.pack(side="left", padx=5)
  entry.pack(side='left", padx=5)

  return frame

def submit(self):
  name =self.name_var.get()
  age =self.age_var.get()
  gender =self.gender_var.get()
  location =self.location_var.get()
  phone_number =self.phone_var.get()
  disease =self.disease_var.get() #New field for disease

  if not (name and age and gender and location and phone_number and disease):
    messagebox.showerror("Input Error", "Please fill in all fields.")
    return
```

```
        #Insert data into the table
        cursor.execute('''
        INSERT INTO patients (name, age, gender, location, phone_number, disease)
        VALUES (?, ?, ?, ?, ?, ?)
        ''', (name, age, gender, location, phone_number, disease))

        conn.commit()
        conn.close()

        messagebox.showinfo("Success", "Patient information submitted successfully.")
        self.clear_fields()

    def clear_fields(self):
        #Clear all entry fields after submission
        self.name_var.set("")
        self.age_var.set("")
        self.gender_var.set("")
        self.location_var.set("")
        self.phone_var.set("")
        self.disease_var.set("")  #Clear the disease field


class DoctorDashboard(tk.Frame):
    def __init__(self, parent, controller):
        super()._init_(parent)
        self.controller =controller
        self.current_index =0

        self.controller.create_menu()

        #Background color
        self.configure(bg="#e0f7fa")

        #Title
        tk.Label(self, text="Doctor Dashboard", font=('Helvetica', 36, 'bold'), bg="#e0f7fa",
      fg="#00796b").pack(pady=20)

        #Appointment Details
        self.details_label =tk.Label(self, text="", font=('Helvetica', 16), bg="#e0f7fa",
      fg="#004d40", wraplength=800)
        self.details_label.pack(pady=20, padx=20)

        #Buttons
        button_frame =tk.Frame(self, bg="#e0f7fa")
        button_frame.pack(pady=20, padx=20, fill="x")

        tk.Button(button_frame, text="Next Appointment", font=('Helvetica', 14),
      bg="#00796b", fg="white", command=self.show_next_appointment,
      width=20).pack(side="left", padx=10)
```

```
        tk.Button(button_frame, text="Logout", font=('Helvetica', 14), bg="#004d40",
      fg="white", command=lambda: self.controller.show_frame("IntroPage"),
      width=20).pack(side="right", padx=10)

    def show_next_appointment(self):
        conn = sqlite3.connect('hospital.db')
        cursor = conn.cursor()

        cursor.execute("SELECT * FROM patients")
        patients = cursor.fetchall()

        if patients and self.current_index < len(patients):
            patient = patients[self.current_index]
            details = (
                f"Patient: {patient[1]}\n"
                f"Disease: {patient[6]}\n"  # Assuming disease is in the 7th column
                f"Age: {patient[2]}\n"
                f"Gender: {patient[3]}"
            )
            self.details_label.config(text=details)
            self.current_index += 1
        else:
            messagebox.showinfo("End", "No more patients.")

        conn.close()


import tkinter as tk
from tkinter import messagebox
import webbrowser
from PIL import Image, ImageTk

class AboutPage(tk.Frame):
    def __init__(self, parent, controller):
        super().__init__(parent)
        self.controller = controller

        self.controller.create_menu()

        # About Page Title
        tk.Label(self, text="About the Project", font=('Helvetica', 24, 'bold')).pack(pady=20)

        # Team Photo
        team_image = Image.open("hospital.png") # Replace with your team photo
        team_image = team_image.resize((200, 150))
        team_photo = ImageTk.PhotoImage(team_image)
        team_label = tk.Label(self, image=team_photo)
        team_label.image = team_photo  # Keep a reference to avoid garbage collection
        team_label.pack(pady=10)

        # Team Members Section
        tk.Label(self, text="Team Members:", font=('Helvetica', 16, 'underline')).pack(pady=10)
```

```
        #Team Members Names
        tk.Label(self, text="Ujwal", font=('Helvetica', 14)).pack()
        tk.Label(self, text="Ritesh", font=('Helvetica', 14)).pack()
        tk.Label(self, text="Yash", font=('Helvetica', 14)).pack()

        #GitHub Icon and Link Section
        github_frame =tk.Frame(self)
        github_frame.pack(pady=20)

        #Load the GitHub icon image
        github_image =Image.open("github-100.png") #Replace with your GitHub icon
        github_image =github_image.resize((32, 32))
        github_icon =ImageTk.PhotoImage(github_image)

        #Create a label with the image and make it clickable
        github_label =tk.Label(github_frame, image=github_icon, cursor="hand2")
        github_label.image =github_icon  #Keep a reference to avoid garbage collection
        github_label.pack(side="left", padx=10)

        #GitHub link text next to the image
        github_link =tk.Label(github_frame, text="View on GitHub", font=('Helvetica', 14),
      fg="blue", cursor="hand2")
        github_link.pack(side="left")
        github_link.bind("<Button-1>", lambda e:
      webbrowser.open_new("https://github.com/ujwalsahu123/Python-Hospital-
      Management-System"))

        #Description for the GitHub link
        tk.Label(self, text="This project is hosted on GitHub for open-source collaboration and
      version control.").pack(pady=10)



if __name__ == "__main__":
    app =HospitalManagementSystem()
    app.mainloop()
    conn.close()
```

Tools Used : python 3 ,VS code , SQL lite .

Database Structure:

1 users Table:

- Stores login credentials and user types (patient or doctor).
- Fields:
  - id: Unique identifier.
  - name: The username for login.

- `password`: The user's password.
- `user_type`: Defines whether the user is a "patient" or "doctor".

id	name	age	gender	location	phone_number	disease
Fil...	Filter	Fil...	Filter	Filter	Filter	Filter
1	ritesh	20	male	borivali	123456789	cold
2	yash	21	male	powai	12345788	fever
3	ujwal	25	male	mira-road	123438594	cough

2 `patients` Table:

- Stores patient information.
- Fields:
  - `id`: Unique identifier.
  - `name`: Patient's name.
  - `age`: Patient's age.
  - `gender`: Patient's gender.
  - `location`: Patient's location.
  - `phone_number`: Patient's phone number.
  - `disease`: Patient's disease or medical issue.

	username	password	user_type
	Filter	Filter	Filter
1	doc	doc	doctor
2	ritesh	1234	patient
3	yash	1234	patient
4	ujwal	1234	patient

Conclusion

In conclusion, the development of the Hospital Management System (HMS) has demonstrated significant advancements in streamlining hospital operations and enhancing patient care. Through the implementation of Python and associated technologies, this project has effectively addressed key challenges faced by healthcare facilities, including efficient patient data management, seamless appointment scheduling, and improved inventory control.

The system's design incorporates a user-friendly interface, robust backend functionality, and secure data handling practices, ensuring that it meets the needs of both administrative staff and healthcare professionals. The modular approach to development

allows for future scalability and integration with other systems, making it a versatile tool for hospitals of various sizes.

Correction Parameter	Formative Assessment : 40%	Timely completions practical 40%	Attendance Learning 20%	
Marks Obtained				