Computer Science Project On

MEDICS: MEDICAL EXAMINATION AND DIAGNOSIS INFORMATION CARE SYSTEM

Project Link: https://github.com/codingwithnsh/MEDICS

Submitted by:

Class: 12

Section: B

Registration Number:

Submitted To: **Tr.P. Sathyavathi Mallya**

INDEX

ACKNOWLEDGMENT

I would like to express my gratitude to our Principal, **Mrs. Swathi Kulkarni**, for providing us with the lab and necessary resources that were crucial for its completion.

I would also like to thank my computer science subject teacher,**Tr.P. Sathyavathi Mallya**, for her invaluable guidance throughout the coding process helping us to understand complex concepts and fixing our errors with patience.

Additionally, I extend my thanks to **Tr. Ramya Shetty**, for her technical assistance and ensuring that everything ran smoothly in the lab.

Lastly, I sincerely appreciate the dedication and teamwork of my teammates **T.Sharan Pai** and **Sweekrath K** whose collaboration made this project not only possible but also successful.

Moreover, I would like to thank **God** for providing the strength and wisdom to complete this project.

INTRODUCTION

The rapid growth of digital technology has changed the healthcare industry, making medical services faster, smarter, and more accessible. As hospitals and clinics increasingly depend on automated systems to manage appointments, patient data, and diagnosis support, there is a strong need for efficient and user-friendly software that can assist both patients and medical professionals. With these needs in mind, this project, **MEDICS: Medical Examination and Diagnosis Information Care System**, has been developed .

The main goal of this program is to speed up the early stages of medical consultation. Many patients struggle to find the right medical department for their symptoms, which can delay proper care. The project is used by both patients and doctors, each with their own dashboards. In the patient dashboard, individuals can enter their symptoms. The system assigns a department based on these symptoms. It solves the issue by allowing patients to input their symptoms. The system then matches these inputs with the right medical specialization. Once it identifies the correct department, it lists available doctors in the patient's city, along with appointment dates and time slots. There is also a doctor dashboard. Where doctors can see their scheduled patients for the day, track the progress of appointments, and update patient status. This leads to smoother workflow management and helps with efficient patient care.

Overall, this project serves as a solid example of how software can support healthcare operations, improve the patient experience, and contribute to the advancement of smart medical systems. It reflects the growing role of automation in today’s world and highlights the importance of technology in enhancing everyday life.

HARDWARE AND SOFTWARE REQUIREMENTS

Minimum Hardware Requirements:

1. Processor: 1 GHz or faster
2. RAM: 2 GB or more
3. Storage: 500 MB of free disk space
4. Display: 1024 x 768 resolution or higher

Minimum Software Requirements:

1. Operating System: Windows 7 or later
2. Python: Version 3.7 or later
3. Libraries:
    a. tkinter
    b. pandas
    c. openpyxl
    d. pillow
    e. textwrap
    f. datetime

WHY PYTHON?

Python was chosen for developing this project MEDICS: Medical Examination and Diagnosis Information Care System, because of its simplicity, versatility, and powerful library support. Python is a high level, user friendly programming language that allows developers to build applications quickly without worrying about complex syntax. It emphasizes readability, which makes the code easier to write, understand, and debug. Python provides a rich collection of built-in libraries and frameworks that make application development efficient.

Another reason for selecting Python is its cross-platform support the same code can run on Windows, Linux, or macOS without modification. This allows the project to be easily portable and usable on different systems.

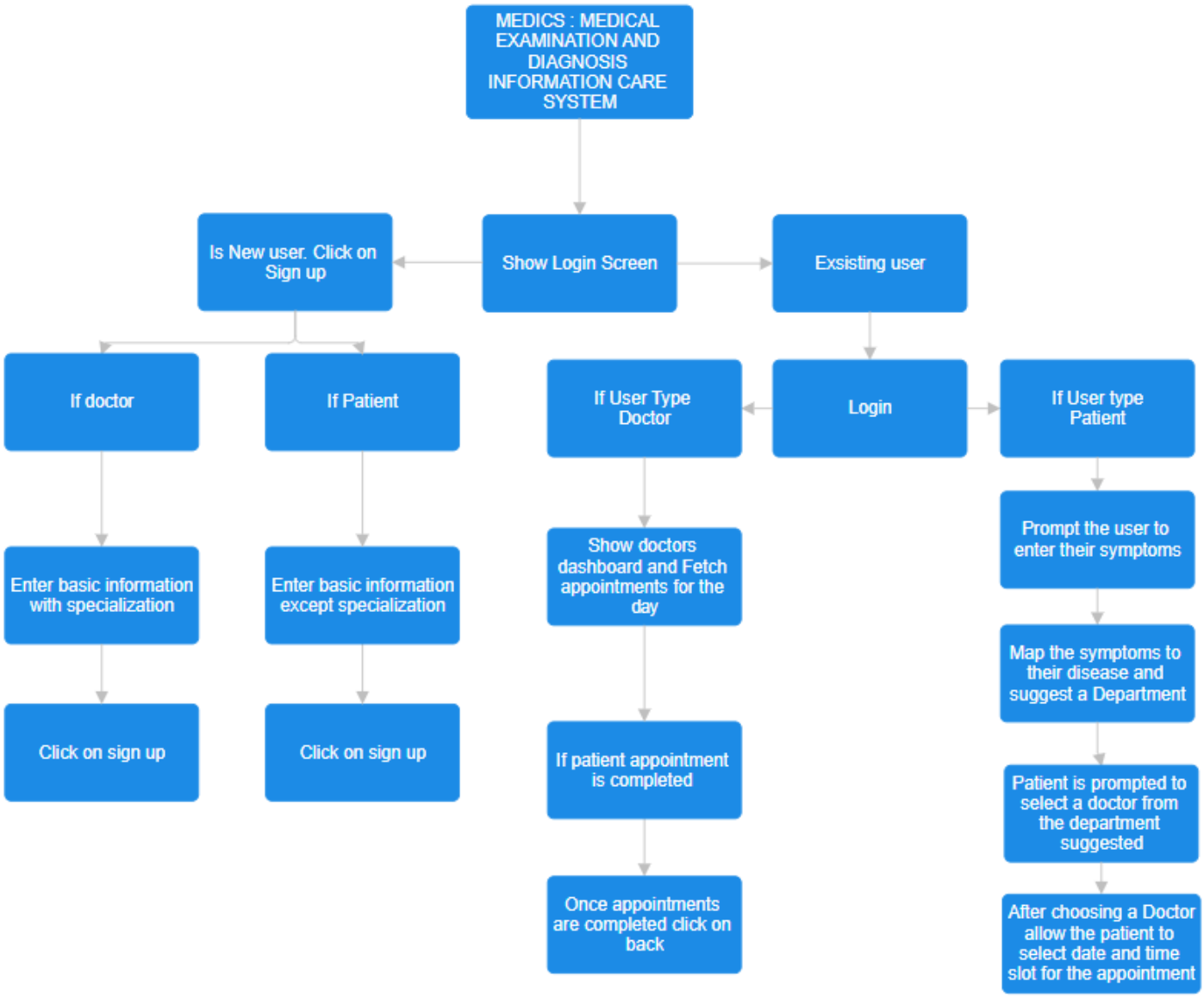
Overall, Python is the best fit for this project because it is beginner friendly, powerful, flexible, and highly suitable for GUI-based applications that require data storage and real time interaction.

ALGORITHM

Step 1: Start
Step 2: Initialize required libraries
Step 3: Load database.xlsx.
Step 4: Display Login screen.
Step 5: User enters **username** and **password**.
Step 6: Validate credentials from database.
Step 7: If login is successful go to **Step 9** else go to step
Step 8: Identify User Type (Patient or Doctor), If patient go to **Step 10** else go to Step
Step 9: Open Patient Dashboard.
Step 10: Input symptoms.
Step 11: Compare symptoms with predefined specialization mapping.
Step 12: If a matching specialization is found then go to **Step 14**. Else show error.
Step 13: Display list of available doctors of that specialization in same city.
Step 14: Input selected doctor, date, and time slot.
Step 15: Save appointment slot and doctor choose by the patient in database.
Step 16: Redirect patient to login page.
Step 17: Display error and return to **Step 11**.
Step 18: Open Doctor Dashboard.
Step 19: Display list of assigned patients whose appointment is scheduled for today.
Step 20: Input the selected patient and highlight that patient.
Step 21: Show Done and Next Patient Button.
Step 22: If done pressed go to **Step 23** or if next patient is pressed go to **Step 26**
Step 23: Remove patient from doctor's assigned list.
Step 24: Clear patient appointment details from database.
Step 25: Refresh dashboard

Step 26: Highlight next patient in list.
Step 27: If back pressed go to **Step 4**.
Step 28: Stop

DESIGN WORK


MEDICS : MEDICAL EXAMINATION AND DIAGNOSIS INFORMATION CARE SYSTEM
Show Login Screen
Is New user. Click on Sign up
Exsisting user
If doctor
If Patient
Enter basic information with specialization
Enter basic information except specialization
Click on sign up
Click on sign up
If User Type Doctor
Login
If User type Patient
Show doctors dashboard and Fetch appointments for the day
If patient appointment is completed
Once appointments are completed click on back
Prompt the user to enter their symptoms
Map the symptoms to their disease and suggest a Department
Patient is prompted to select a doctor from the department suggested
After choosing a Doctor allow the patient to select date and time slot for the appointment

SOURCE CODE

```
import os
import tkinter as tk
from tkinter import messagebox, ttk
from PIL import Image, ImageTk
import pandas as pd
import textwrap
import datetime

# Get the directory of the script
script_dir = os.path.dirname(os.path.abspath(__file__))

# Load Excel database
users_df = pd.read_excel(os.path.join(script_dir, 'database.xlsx'), sheet_name='users')
filtered_specializations = set()

# Ensure proper column types
users_df['Symptoms'] = users_df['Symptoms'].astype(str)
if 'Appointment Slot' not in users_df.columns:
    users_df['Appointment Slot'] = ''
if 'Preferred Doctor' not in users_df.columns:
    users_df['Preferred Doctor'] = ''
# Add Appointment Date column if missing and ensure type is string
if 'Appointment Date' not in users_df.columns:
    users_df['Appointment Date'] = ''
users_df['Appointment Date'] = users_df['Appointment Date'].astype(str)
```

```
# Fixed number of patients allowed per slot/date per doctor
SLOT_CAPACITY = 5
# Main root window
root = tk.Tk()
root.title('MEDICS: MEDICAL EXAMINATION AND DIAGNOSIS INFORMATION CARE SYSTEM')
root.geometry('800x600')
# Global variable for the currently logged-in user
current_user = None
# Global variable for results_frame
results_frame = None
# Global variable for confirm button
confirm_btn = None
def show_view(view, matched_specialization=None):
    for widget in root.winfo_children():
        widget.destroy()
    if view == 'login':
        show_login()
    elif view == 'signup':
        show_signup()
    elif view == 'patient':
        show_patient_dashboard()
    elif view == 'doctor':
        show_doctor_dashboard()
    elif view == 'select_doctor':
        show_select_doctor(matched_specialization)
```

```
def show_login():
    frame = ttk.Frame(root, padding="20")
    frame.pack(fill='both', expand=True)

    # Load and display image
    img = Image.open(os.path.join(script_dir, 'login_image.png'))
    img = img.resize((400, 399), Image.LANCZOS)
    img = ImageTk.PhotoImage(img)
    img_label = ttk.Label(frame, image=img)
    img_label.image = img  # Keep a reference to avoid garbage collection
    img_label.pack(pady=10)
    ttk.Label(frame, text='Username').pack(pady=5)
    username_entry = ttk.Entry(frame)
    username_entry.pack(pady=5)
    ttk.Label(frame, text='Password').pack(pady=5)
    password_entry = ttk.Entry(frame, show='*')
    password_entry.pack(pady=5)
    login_btn = ttk.Button(frame, text='Login', command=lambda:
authenticate_user(username_entry, password_entry))
    login_btn.pack(pady=10)
    signup_btn = ttk.Button(frame, text='Sign Up', command=lambda: show_view('signup'))
    signup_btn.pack(pady=10)
def show_signup():
    frame = ttk.Frame(root, padding="20")
    frame.pack(fill='both', expand=True)
    ttk.Label(frame, text='Name').pack(pady=5)
    name_entry = ttk.Entry(frame)
```

```
    name_entry.pack(pady=5)
    ttk.Label(frame, text='Username').pack(pady=5)
    username_entry = ttk.Entry(frame)
    username_entry.pack(pady=5)
    ttk.Label(frame, text='Password').pack(pady=5)
    password_entry = ttk.Entry(frame, show='*')
    password_entry.pack(pady=5)
    ttk.Label(frame, text='City').pack(pady=5)
    city_entry = ttk.Entry(frame)
    city_entry.pack(pady=5)
    ttk.Label(frame, text='Address').pack(pady=5)
    address_entry = ttk.Entry(frame)
    address_entry.pack(pady=5)
    ttk.Label(frame, text='Specialization (if Doctor)').pack(pady=5)
    specialization_entry = ttk.Entry(frame)
    specialization_entry.pack(pady=5)
    signup_btn = ttk.Button(frame, text='Sign Up', command=lambda:
register_user(name_entry, username_entry, password_entry, city_entry, address_entry,
specialization_entry))
    signup_btn.pack(pady=10)
    login_btn = ttk.Button(frame, text='Back to Login', command=lambda:
show_view('login'))
    login_btn.pack(pady=10)
def register_user(name_entry, username_entry, password_entry, city_entry, address_entry,
specialization_entry):
    global users_df
    name = name_entry.get()
    username = username_entry.get()
```

```
    password = password_entry.get()
    city = city_entry.get()
    address = address_entry.get()
    user_type = 'Doctor' if specialization_entry.get() else 'Patient'
    specialization = specialization_entry.get() if user_type == 'Doctor' else ''
 if not name or not username or not password or not city or not address or (user_type == 'Doctor' and not specialization):
        messagebox.showerror('Error', 'All fields are required.')
        return
    if username in users_df['Username'].values:
        messagebox.showerror('Error', 'Username already exists.')
        return
    new_user = pd.DataFrame([[name, username, password, address, city, '', '', '', user_type, specialization, '', '']],
                            columns=['Name', 'Username', 'Password', 'Address', 'City', 'Diagnosis', 'Appointment Slot', 'Preferred Doctor', 'User Type', 'Specialization', 'Assigned Patients', 'Doctor ID'])
    users_df = pd.concat([users_df, new_user], ignore_index=True)
    save_data()
    messagebox.showinfo('Success', 'User registered successfully.')
    show_view('login')
def authenticate_user(username_entry, password_entry):
    global current_user
    current_user = None  # Reset current_user on each login attempt
    username = username_entry.get()
    password = password_entry.get()
    user = users_df[(users_df['Username'] == username) & (users_df['Password'] == password)]
    if not user.empty:
```

```
        current_user = username
        user_type = user['User Type'].values[0]
        if user_type == 'Patient':
            show_view('patient')
        elif user_type == 'Doctor':
            show_view('doctor')
    else:
        messagebox.showerror('Error', 'Invalid username or password.')
def show_patient_dashboard():
    global results_frame
    frame = ttk.Frame(root, padding="20")
    frame.pack(fill='both', expand=True)
    # Load and display image
    img = Image.open(os.path.join(script_dir, 'patient_dashboard_image.png'))
    img = img.resize((200, 200), Image.LANCZOS)
    img = ImageTk.PhotoImage(img)
    img_label = ttk.Label(frame, image=img)
    img_label.image = img  # Keep a reference to avoid garbage collection
    img_label.pack(pady=10)
    ttk.Label(frame, text='Enter Symptoms (comma-separated):').pack(pady=5)
    diagnosis_entry = ttk.Entry(frame)
    diagnosis_entry.pack(pady=5)
    submit_btn = ttk.Button(frame, text='Submit Symptoms', command=lambda:
submit_symptoms(diagnosis_entry))
    submit_btn.pack(pady=10, anchor='s')  # Anchor the button to the bottom
    results_frame = ttk.Frame(frame)
    results_frame.pack(pady=10, fill='both', expand=True)
```

```
def submit_symptoms(diagnosis_entry):
    symptoms = diagnosis_entry.get().strip().lower()
    users_df.loc[users_df['Username'] == current_user, 'Symptoms'] = symptoms
    save_data()
    # Map symptoms to specializations
    specialization_map = {
        # General Medicine
        'fever': 'The Department of General Medicine',
        'cough': 'The Department of General Medicine',
        'cold': 'The Department of General Medicine',
        'nausea': 'The Department of General Medicine',
        'headache': 'The Department of General Medicine',
        'fatigue': 'The Department of General Medicine',
        'body ache': 'The Department of General Medicine',
        'vomiting': 'The Department of General Medicine',
        'sore throat': 'The Department of General Medicine',
        'dizziness': 'The Department of General Medicine',
        'loss of appetite': 'The Department of General Medicine',
        'weakness': 'The Department of General Medicine',
        'chills': 'The Department of General Medicine',
        'high fever': 'The Department of General Medicine',
        # Urology
        'urine': 'The Department of Urology',
        'urine problems': 'The Department of Urology',
        'kidney pain': 'The Department of Urology',
        'bladder issues': 'The Department of Urology',
        'frequent urination': 'The Department of Urology',
```

'painful urination': 'The Department of Urology',

'blood in urine': 'The Department of Urology',

'urinary incontinence': 'The Department of Urology',

'prostate issues': 'The Department of Urology',

'pelvic pain': 'The Department of Urology',

'reduced urine output': 'The Department of Urology'

Cardiology

'chest pain': 'The Department of Cardiology',

'shortness of breath': 'The Department of Cardiology',

'irregular heartbeat': 'The Department of Cardiology',

'high blood pressure': 'The Department of Cardiology',

'low blood pressure': 'The Department of Cardiology',

'heart palpitations': 'The Department of Cardiology',

'swelling in legs': 'The Department of Cardiology',

'dizziness with chest pain': 'The Department of Cardiology',

'cyanosis': 'The Department of Cardiology',

Endocrinology

'diabetes': 'The Department of Endocrinology',

'thyroid issues': 'The Department of Endocrinology',

'obesity': 'The Department of Endocrinology',

'hormonal imbalance': 'The Department of Endocrinology',

'growth problems': 'The Department of Endocrinology',

'adrenal gland issues': 'The Department of Endocrinology',

'hyperthyroidism': 'The Department of Endocrinology',

'hypothyroidism': 'The Department of Endocrinology',

Dermatology

'skin rash': 'The Department of Dermatology',

'acne': 'The Department of Dermatology',

'eczema': 'The Department of Dermatology',

'psoriasis': 'The Department of Dermatology',

'hair loss': 'The Department of Dermatology',

'skin infection': 'The Department of Dermatology',

'allergic rash': 'The Department of Dermatology',

'dry skin': 'The Department of Dermatology',

'dark spots': 'The Department of Dermatology',

Hematology

'anemia': 'The Department of Hematology',

'blood disorders': 'The Department of Hematology',

'leukemia': 'The Department of Hematology',

'clotting problems': 'The Department of Hematology',

'low platelet count': 'The Department of Hematology',

'iron deficiency': 'The Department of Hematology',

'abnormal bleeding': 'The Department of Hematology',

'excessive bruising': 'The Department of Hematology',

Orthopedics

'back pain': 'The Department of Orthopedics',

'joint pain': 'The Department of Orthopedics',

'bone fracture': 'The Department of Orthopedics',

'arthritis': 'The Department of Orthopedics',

'muscle strain': 'The Department of Orthopedics',

'neck pain': 'The Department of Orthopedics',

'knee pain': 'The Department of Orthopedics',

'spinal problems': 'The Department of Orthopedics',

'frozen shoulder': 'The Department of Orthopedics',

'osteoporosis': 'The Department of Orthopedics',

Ophthalmology

'vision loss': 'The Department of Ophthalmology',

'red eyes': 'The Department of Ophthalmology',

'blurred vision': 'The Department of Ophthalmology',

'cataract': 'The Department of Ophthalmology',

'glaucoma': 'The Department of Ophthalmology',

'watery eyes': 'The Department of Ophthalmology',

'eye pain': 'The Department of Ophthalmology',

'itchy eyes': 'The Department of Ophthalmology',

'light sensitivity': 'The Department of Ophthalmology',

ENT

'earache': 'The Department of ENT',

'hearing loss': 'The Department of ENT',

'nosebleed': 'The Department of ENT',

'sinusitis': 'The Department of ENT',

'tonsillitis': 'The Department of ENT',

'blocked ears': 'The Department of ENT',

'throat pain': 'The Department of ENT',

'ear discharge': 'The Department of ENT',

Gastroenterology

'stomach pain': 'The Department of Gastroenterology',

'acid reflux': 'The Department of Gastroenterology',

'constipation': 'The Department of Gastroenterology',

'diarrhea': 'The Department of Gastroenterology',

'liver disease': 'The Department of Gastroenterology',

'abdominal bloating': 'The Department of Gastroenterology',

```
    'vomiting blood': 'The Department of Gastroenterology',
    'jaundice': 'The Department of Gastroenterology',
    'indigestion': 'The Department of Gastroenterology',
    'intestinal blockage': 'The Department of Gastroenterology',
}
# Extended diseases list for better filtering (100+ entries)
extra_map = {
    # General Medicine
    'flu': 'The Department of General Medicine',
    'influenza': 'The Department of General Medicine',
    'viral fever': 'The Department of General Medicine',
    'chills and sweats': 'The Department of General Medicine',
    'sweating': 'The Department of General Medicine',
    'weight loss': 'The Department of General Medicine',
    'unintentional weight loss': 'The Department of General Medicine',
    'weight gain': 'The Department of General Medicine',
    'dehydration': 'The Department of General Medicine',
    'insomnia': 'The Department of General Medicine',
    'sleep problems': 'The Department of General Medicine',
    'body weakness': 'The Department of General Medicine',
    'generalized weakness': 'The Department of General Medicine',
    'food poisoning': 'The Department of General Medicine',
    'poisoning': 'The Department of General Medicine',
    'allergy': 'The Department of General Medicine',
    'seasonal allergies': 'The Department of General Medicine',
    'malaria symptoms': 'The Department of General Medicine',
    'dengue symptoms': 'The Department of General Medicine',
```

'typhoid': 'The Department of General Medicine',

'heat stroke': 'The Department of General Medicine',

'heat exhaustion': 'The Department of General Medicine',

'anxiety': 'The Department of General Medicine',

'panic attack': 'The Department of General Medicine',

'depression': 'The Department of General Medicine',

'body stiffness': 'The Department of General Medicine',

'drug reaction': 'The Department of General Medicine',

'sinus headache': 'The Department of General Medicine',

'migraine headache': 'The Department of General Medicine',

'ear pain with fever': 'The Department of General Medicine',

Urology

'kidney stone': 'The Department of Urology',

'kidney stones': 'The Department of Urology',

'renal colic': 'The Department of Urology',

'urinary retention': 'The Department of Urology',

'urinary tract infection': 'The Department of Urology',

'uti': 'The Department of Urology',

'burning urination': 'The Department of Urology',

'nocturia': 'The Department of Urology',

'bedwetting': 'The Department of Urology',

'enlarged prostate': 'The Department of Urology',

'bph': 'The Department of Urology',

'hematuria': 'The Department of Urology',

'urinary urgency': 'The Department of Urology',

'urinary frequency': 'The Department of Urology',

'incomplete emptying': 'The Department of Urology',

Cardiology

'angina': 'The Department of Cardiology',

'heart failure': 'The Department of Cardiology',

'tachycardia': 'The Department of Cardiology',

'bradycardia': 'The Department of Cardiology',

'edema': 'The Department of Cardiology',

'syncope': 'The Department of Cardiology',

'fainting': 'The Department of Cardiology',

'orthopnea': 'The Department of Cardiology',

'paroxysmal nocturnal dyspnea': 'The Department of Cardiology',

'pnd': 'The Department of Cardiology',

'hypertension': 'The Department of Cardiology',

'hypotension': 'The Department of Cardiology',

'bp high': 'The Department of Cardiology',

'bp low': 'The Department of Cardiology',

'leg swelling': 'The Department of Cardiology',

'chest tightness': 'The Department of Cardiology',

Endocrinology

'pcos': 'The Department of Endocrinology',

'pcod': 'The Department of Endocrinology',

'menstrual irregularities': 'The Department of Endocrinology',

'infertility': 'The Department of Endocrinology',

'metabolic syndrome': 'The Department of Endocrinology',

"cushing's syndrome": 'The Department of Endocrinology',

"addison's disease": 'The Department of Endocrinology',

'pituitary disorder': 'The Department of Endocrinology',

'goiter': 'The Department of Endocrinology',

'thyroid swelling': 'The Department of Endocrinology',

'hyperglycemia': 'The Department of Endocrinology',

'hypoglycemia': 'The Department of Endocrinology',

'vitamin d deficiency': 'The Department of Endocrinology',

'vitamin b12 deficiency': 'The Department of Endocrinology',

'osteomalacia': 'The Department of Endocrinology',

Dermatology

'urticaria': 'The Department of Dermatology',

'hives': 'The Department of Dermatology',

'dermatitis': 'The Department of Dermatology',

'contact dermatitis': 'The Department of Dermatology',

'ringworm': 'The Department of Dermatology',

'fungal infection': 'The Department of Dermatology',

'tinea': 'The Department of Dermatology',

'scabies': 'The Department of Dermatology',

'alopecia': 'The Department of Dermatology',

'dandruff': 'The Department of Dermatology',

'nail fungus': 'The Department of Dermatology',

'warts': 'The Department of Dermatology',

'vitiligo': 'The Department of Dermatology',

'sunburn': 'The Department of Dermatology',

'melasma': 'The Department of Dermatology',

'boils': 'The Department of Dermatology',

'cellulitis': 'The Department of Dermatology',

'impetigo': 'The Department of Dermatology',

'itching': 'The Department of Dermatology',

'pruritus': 'The Department of Dermatology',

'folliculitis': 'The Department of Dermatology',

'athlete foot': 'The Department of Dermatology',

Hematology

'thrombocytopenia': 'The Department of Hematology',

'hemophilia': 'The Department of Hematology',

'sickle cell disease': 'The Department of Hematology',

'thalassemia': 'The Department of Hematology',

'polycythemia': 'The Department of Hematology',

'pancytopenia': 'The Department of Hematology',

'neutropenia': 'The Department of Hematology',

'hemolysis': 'The Department of Hematology',

'aplastic anemia': 'The Department of Hematology',

'deep vein thrombosis': 'The Department of Hematology',

'dvt': 'The Department of Hematology',

Orthopedics

'sciatica': 'The Department of Orthopedics',

'osteoarthritis': 'The Department of Orthopedics',

'rheumatoid arthritis': 'The Department of Orthopedics',

'gout': 'The Department of Orthopedics',

'low back pain': 'The Department of Orthopedics',

'shoulder pain': 'The Department of Orthopedics',

'elbow pain': 'The Department of Orthopedics',

'wrist pain': 'The Department of Orthopedics',

'ankle pain': 'The Department of Orthopedics',

'sprain': 'The Department of Orthopedics',

'strain': 'The Department of Orthopedics',

'tendonitis': 'The Department of Orthopedics',

'carpal tunnel': 'The Department of Orthopedics',

'slipped disc': 'The Department of Orthopedics',

'herniated disc': 'The Department of Orthopedics',

'meniscus injury': 'The Department of Orthopedics',

'ligament tear': 'The Department of Orthopedics',

'acl tear': 'The Department of Orthopedics',

'mcl injury': 'The Department of Orthopedics',

'sports injury': 'The Department of Orthopedics',

'plantar fasciitis': 'The Department of Orthopedics',

'heel pain': 'The Department of Orthopedics',

'tennis elbow': 'The Department of Orthopedics',

'frozen elbow': 'The Department of Orthopedics',

Ophthalmology

'conjunctivitis': 'The Department of Ophthalmology',

'pink eye': 'The Department of Ophthalmology',

'corneal ulcer': 'The Department of Ophthalmology',

'dry eyes': 'The Department of Ophthalmology',

'eye redness': 'The Department of Ophthalmology',

'eye swelling': 'The Department of Ophthalmology',

'floaters': 'The Department of Ophthalmology',

'double vision': 'The Department of Ophthalmology',

'eye trauma': 'The Department of Ophthalmology',

'eyelid infection': 'The Department of Ophthalmology',

'sty': 'The Department of Ophthalmology',

'stye': 'The Department of Ophthalmology',

'blepharitis': 'The Department of Ophthalmology',

'contact lens irritation': 'The Department of Ophthalmology',

'vision changes': 'The Department of Ophthalmology',

ENT

'otitis media': 'The Department of ENT',

'otitis externa': 'The Department of ENT',

'ear ringing': 'The Department of ENT',

'tinnitus': 'The Department of ENT',

'vertigo': 'The Department of ENT',

'nasal congestion': 'The Department of ENT',

'runny nose': 'The Department of ENT',

'allergic rhinitis': 'The Department of ENT',

'deviated septum': 'The Department of ENT',

'snoring': 'The Department of ENT',

'sleep apnea': 'The Department of ENT',

'loss of smell': 'The Department of ENT',

'anosmia': 'The Department of ENT',

'hoarseness': 'The Department of ENT',

'laryngitis': 'The Department of ENT',

'ear fullness': 'The Department of ENT',

'post nasal drip': 'The Department of ENT',

Gastroenterology

'gastritis': 'The Department of Gastroenterology',

'stomach ulcer': 'The Department of Gastroenterology',

'peptic ulcer': 'The Department of Gastroenterology',

'hemorrhoids': 'The Department of Gastroenterology',

'piles': 'The Department of Gastroenterology',

'anal fissure': 'The Department of Gastroenterology',

'anal pain': 'The Department of Gastroenterology',

```
    'blood in stool': 'The Department of Gastroenterology',
    'black stool': 'The Department of Gastroenterology',
    'diarrhoea': 'The Department of Gastroenterology',
    'appendicitis': 'The Department of Gastroenterology',
    'pancreatitis': 'The Department of Gastroenterology',
    'gallstones': 'The Department of Gastroenterology',
    'gall bladder pain': 'The Department of Gastroenterology',
    'hepatitis': 'The Department of Gastroenterology',
    'fatty liver': 'The Department of Gastroenterology',
    'colitis': 'The Department of Gastroenterology',
    'irritable bowel syndrome': 'The Department of Gastroenterology',
    'ibs': 'The Department of Gastroenterology',
    'crohn disease': 'The Department of Gastroenterology',
    'crohn\'s': 'The Department of Gastroenterology',
    'ulcerative colitis': 'The Department of Gastroenterology',
    'celiac disease': 'The Department of Gastroenterology',
    'flatulence': 'The Department of Gastroenterology',
    'gas': 'The Department of Gastroenterology',
    'acid burps': 'The Department of Gastroenterology',
    'heartburn': 'The Department of Gastroenterology',
    'belching': 'The Department of Gastroenterology',
    'loss of appetite with nausea': 'The Department of Gastroenterology',
}
specialization_map.update(extra_map)

# Find matching specialization based on symptoms
matched_specialization = None
```

```
    for symptom, specialization in specialization_map.items():
        if symptom in symptoms:
            matched_specialization = specialization
            break
    if matched_specialization:
        messagebox.showinfo('Specialization Matched', f"Please select a doctor from the {matched_specialization} department.")
        show_view('select_doctor', matched_specialization)
    else:
        messagebox.showerror('Error', 'No matching specialization found. Please check your symptoms.')
def save_data():
    try:
        with pd.ExcelWriter(os.path.join(script_dir, 'database.xlsx'), engine='openpyxl', mode='a', if_sheet_exists='replace') as writer:
            users_df.to_excel(writer, sheet_name='users', index=False)
    except FileNotFoundError:
        messagebox.showerror('Error', 'Database file not found. Please check the file path.')
    except Exception as e:
        messagebox.showerror('Error', f'An error occurred: {e}')
def save_patient_data():
    try:
        with pd.ExcelWriter(os.path.join(script_dir, 'database.xlsx'), engine='openpyxl', mode='a', if_sheet_exists='replace') as writer:
            users_df.to_excel(writer, sheet_name='users', index=False)
    except FileNotFoundError:
        messagebox.showerror('Error', 'Database file not found. Please check the file path.')
    except Exception as e:
```

```
        messagebox.showerror('Error', f'An error occurred: {e}')
def show_select_doctor(matched_specialization=None):
    frame = ttk.Frame(root, padding="20")
    frame.pack(fill='both', expand=True)
    ttk.Label(frame, text='Select Preferred Doctor:').pack(pady=10)
    # Get the city of the current patient
    patient_city = users_df.loc[users_df['Username'] == current_user, 'City'].values[0]
    columns = ('Name', 'Specialization', 'Address')
    tree = ttk.Treeview(frame, columns=columns, show='headings')
    for col in columns:
        tree.heading(col, text=col, anchor='w')
        tree.column(col, width=300, anchor='w')
    # Filter doctors based on the patient's city (case-insensitive) and specialization
    doctor_filter = users_df[(users_df['City'].str.lower().fillna('') == str(patient_city).lower())
& (users_df['User Type'] == 'Doctor')]
    if matched_specialization:
        doctor_filter = doctor_filter[doctor_filter['Specialization'] == matched_specialization]
    for _, row in doctor_filter.iterrows():
        name = row['Name']
        specialization = textwrap.fill(str(row['Specialization']), width=90)
        address = textwrap.fill(str(row['Address']), width=50)
        tree.insert('', 'end', values=(name, specialization, address))
    tree.pack(side='left', fill='both', expand=True)
    scrollbar = ttk.Scrollbar(frame, orient='vertical', command=tree.yview)
    scrollbar.pack(side='right', fill='y')
    tree.configure(yscrollcommand=scrollbar.set)
```

```
    # Adjust row height based on content
    style = ttk.Style()
    style.configure('Treeview', rowheight=40)  # Default row height
    for item in tree.get_children():
        values = tree.item(item, 'values')
        max_lines = max(len(values[1].split('\n')), len(values[2].split('\n')))
        style.configure('Treeview', rowheight=max_lines * 20)  # Adjust the multiplier as needed
    # Date selection (next 14 days)
    ttk.Label(frame, text='Select Preferred Date:').pack(pady=5)
    dates = [(datetime.date.today() + datetime.timedelta(days=i)).strftime('%Y-%m-%d') for i in range(0, 14)]
    date_var = tk.StringVar(value=dates[0])
    date_combo = ttk.Combobox(frame, values=dates, textvariable=date_var, state='readonly')
    date_combo.pack(pady=5)
    ttk.Label(frame, text='Select Preferred Time Slot:').pack(pady=10)
    slots = ['10:00 AM - 11:00 AM', '11:00 AM - 12:00 PM', '01:00 PM - 02:00 PM', '02:00 PM - 03:00 PM']
    slot_var = tk.StringVar(value=slots[0])
    for slot in slots:
        ttk.Radiobutton(frame, text=slot, variable=slot_var, value=slot).pack(pady=5)
    schedule_btn = ttk.Button(frame, text='Book Appointment', command=lambda: schedule_appointment(tree, slot_var, date_var))
    schedule_btn.pack(pady=10)
def schedule_appointment(tree, slot_var, date_var):
    selected_item = tree.selection()
    if not selected_item:
        messagebox.showerror('Error', 'Please select a doctor.')
```

```
        return
    selected_doctor = tree.item(selected_item[0], 'values')[0]
    selected_slot = slot_var.get()
    selected_date = date_var.get()
    # Check if the selected doctor exists in the DataFrame
    doctor_row = users_df[(users_df['Name'] == selected_doctor) & (users_df['User Type'] == 'Doctor')]
    if doctor_row.empty:
        messagebox.showerror('Error', 'Selected doctor not found.')
        return
    doctor_id = doctor_row['Username'].values[0]
    # Enforce slot capacity for the selected doctor, slot and date
    existing_count = users_df[(users_df['Preferred Doctor'] == doctor_id) & (users_df['Appointment Slot'] == selected_slot) & (users_df['Appointment Date'] == selected_date)].shape[0]
    if existing_count >= SLOT_CAPACITY:
        messagebox.showerror('Error', f'This slot on {selected_date} is full. Please choose another slot or date.')
        return
    users_df.loc[users_df['Username'] == current_user, 'Appointment Slot'] = str(selected_slot)
    users_df.loc[users_df['Username'] == current_user, 'Appointment Date'] = str(selected_date)
    users_df.loc[users_df['Username'] == current_user, 'Preferred Doctor'] = doctor_id


    assigned_patients = doctor_row['Assigned Patients'].values[0]
    if pd.isnull(assigned_patients) or not str(assigned_patients).strip():
        assigned_patients_list = []
    else:
```

```
        assigned_patients_list = [p.strip() for p in str(assigned_patients).split(',') if p.strip()]
    if current_user not in assigned_patients_list:
        assigned_patients_list.append(current_user)
    users_df.loc[users_df['Username'] == doctor_id, 'Assigned Patients'] = ',
'.join(assigned_patients_list)
    save_data()
    messagebox.showinfo('Appointment', f'Appointment scheduled with {selected_doctor} on
{selected_date} at {selected_slot}')
    # Reset the form elements
    for widget in root.winfo_children():
        widget.destroy(
    # Redirect to login view
    show_view('login')
def show_doctor_dashboard():
    global action_frame
    frame = ttk.Frame(root, padding="20")
    frame.pack(fill='both', expand=True)
    ttk.Label(frame, text='Doctor Dashboard').pack(pady=10)
    load_patient_data(current_user, frame)
    # Create a frame for action buttons
    action_frame = ttk.Frame(frame, padding="20")
    action_frame.pack(pady=10, fill='both', expand=True)
    back_btn = ttk.Button(frame, text='Back', command=lambda: show_view('login'))
    back_btn.pack(pady=10)
def load_patient_data(username, frame):
    doc_info = users_df[(users_df['Username'] == username) & (users_df['User Type'] ==
'Doctor')]
    if doc_info.empty:
```

```
        messagebox.showerror('Error', 'No doctor data found for the username.')
        return
    doc_info = doc_info.iloc[0]
    assigned_patients = doc_info['Assigned Patients']
    if isinstance(assigned_patients, str):
        assigned_patients = assigned_patients.split(', ')
    else:
        assigned_patients = []
    if not assigned_patients:
        ttk.Label(frame, text='No assigned patients found.').pack(pady=10)
        return
    columns = ('Name', 'Symptoms', 'Appointment Slot', 'Preferred Doctor ID')
    tree = ttk.Treeview(frame, columns=columns, show='headings')
    for col in columns:
        tree.heading(col, text=col)
        tree.column(col, width=150)
    today_str = datetime.date.today().strftime('%Y-%m-%d')
    rows_inserted = 0
    for patient in assigned_patients:
        patient_data = users_df[users_df['Username'] == patient]
        if not patient_data.empty:
            appt_date = str(patient_data['Appointment Date'].values[0]) if 'Appointment Date' in
patient_data.columns else ''
            if appt_date == today_str:
                tree.insert('', 'end', values=(patient_data['Name'].values[0],
patient_data['Symptoms'].values[0], patient_data['Appointment Slot'].values[0],
patient_data['Preferred Doctor'].values[0]))
                rows_inserted += 1
```

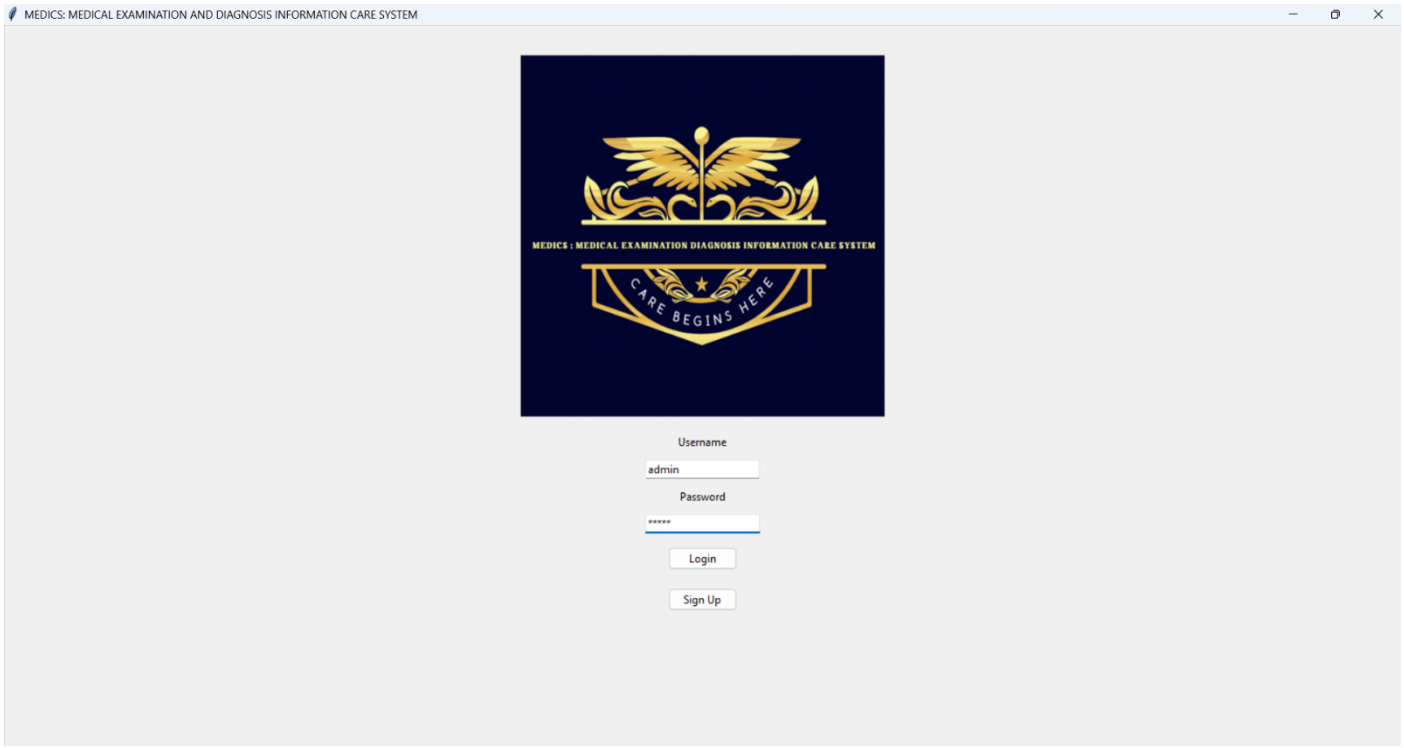
```
    if rows_inserted == 0:
        ttk.Label(frame, text='No patients scheduled for today.').pack(pady=10)
        return
    tree.pack(side='left', fill='both', expand=True)
    scrollbar = ttk.Scrollbar(frame, orient='vertical', command=tree.yview)
    scrollbar.pack(side='right', fill='y')
    tree.configure(yscrollcommand=scrollbar.set)
    tree.bind('<<TreeviewSelect>>', lambda event: show_action_buttons(event, tree,
username, assigned_patients))
def show_action_buttons(event, tree, username, assigned_patients):
    global action_frame
    selected_item = tree.selection()
    if selected_item:
        patient_name = tree.item(selected_item[0], 'values')[0]
        patient_username = users_df[users_df['Name'] == patient_name]['Username'].values[0]
        # Remove existing buttons if any
        for widget in action_frame.winfo_children():
            widget.destroy()
        btn_done = ttk.Button(action_frame, text='Done', command=lambda
p=patient_username: mark_done(p, username, assigned_patients))
        btn_done.pack(pady=5)
        btn_progress = ttk.Button(action_frame, text='Next Patient', command=lambda
p=patient_username: mark_in_progress(p, assigned_patients, tree))
        btn_progress.pack(pady=5)
def mark_done(patient_username, username, assigned_patients):
    if patient_username in assigned_patients:
        assigned_patients.remove(patient_username)
```

```
        users_df.loc[users_df['Username'] == username, 'Assigned Patients'] = ',
'.join(assigned_patients)
        # Clear appointment details for the patient
        users_df.loc[users_df['Username'] == patient_username, 'Preferred Doctor'] = ''
        users_df.loc[users_df['Username'] == patient_username, 'Appointment Slot'] = ''
        users_df.loc[users_df['Username'] == patient_username, 'Appointment Date'] = ''
        save_data()
        show_view('doctor')
    else:
        messagebox.showerror('Error', 'Patient not found in the assigned list.')
def mark_in_progress(patient_username, assigned_patients, tree):
    # Move the selection (blue highlight) to the next patient in the list.
    if not assigned_patients:
        messagebox.showinfo('Next Patient', 'No patients are currently assigned.')
        return
    # Ensure there is a selected item in the tree
    selected = tree.selection()
    if not selected:
        messagebox.showinfo('Next Patient', 'No patient is currently selected.')
        return
    children = tree.get_children()
    current_index = tree.index(selected[0])
    next_index = current_index + 1
    if next_index < len(children):
        next_item = children[next_index]
        tree.selection_set(next_item)
        tree.focus(next_item)
```

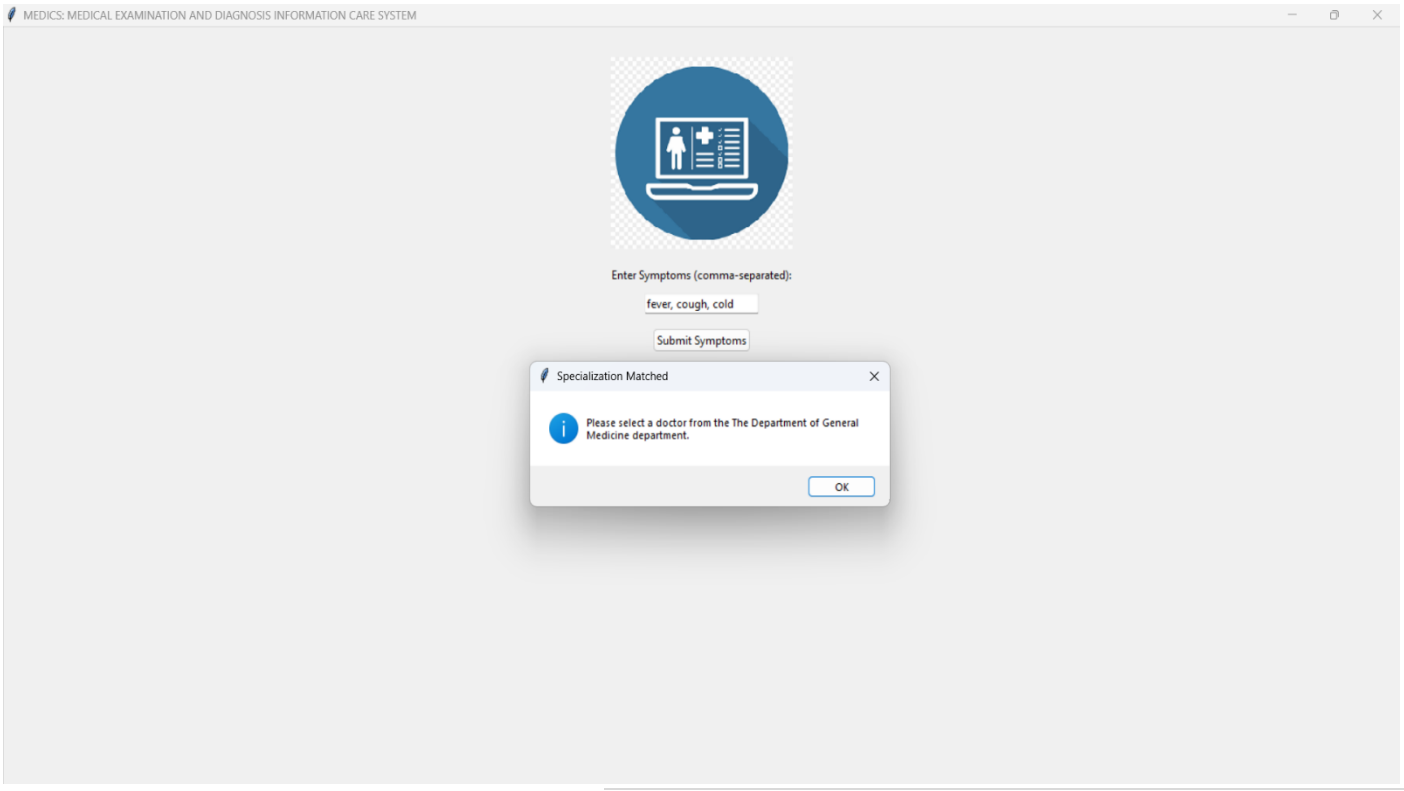
```
        tree.see(next_item)
        # Trigger the select event so buttons refresh for the new selection
        tree.event_generate('<<TreeviewSelect>>')
    else:
        # No next patient: clear the selection (remove blue highlight) and action buttons
        tree.selection_remove(selected[0])
        tree.focus('')
        for widget in action_frame.winfo_children():
            widget.destroy()
        messagebox.showinfo('Next Patient', 'No more patients in the schedule.')
# Show login view initially
show_view('login')
root.mainloop()
```

OUTPUT SCREEN

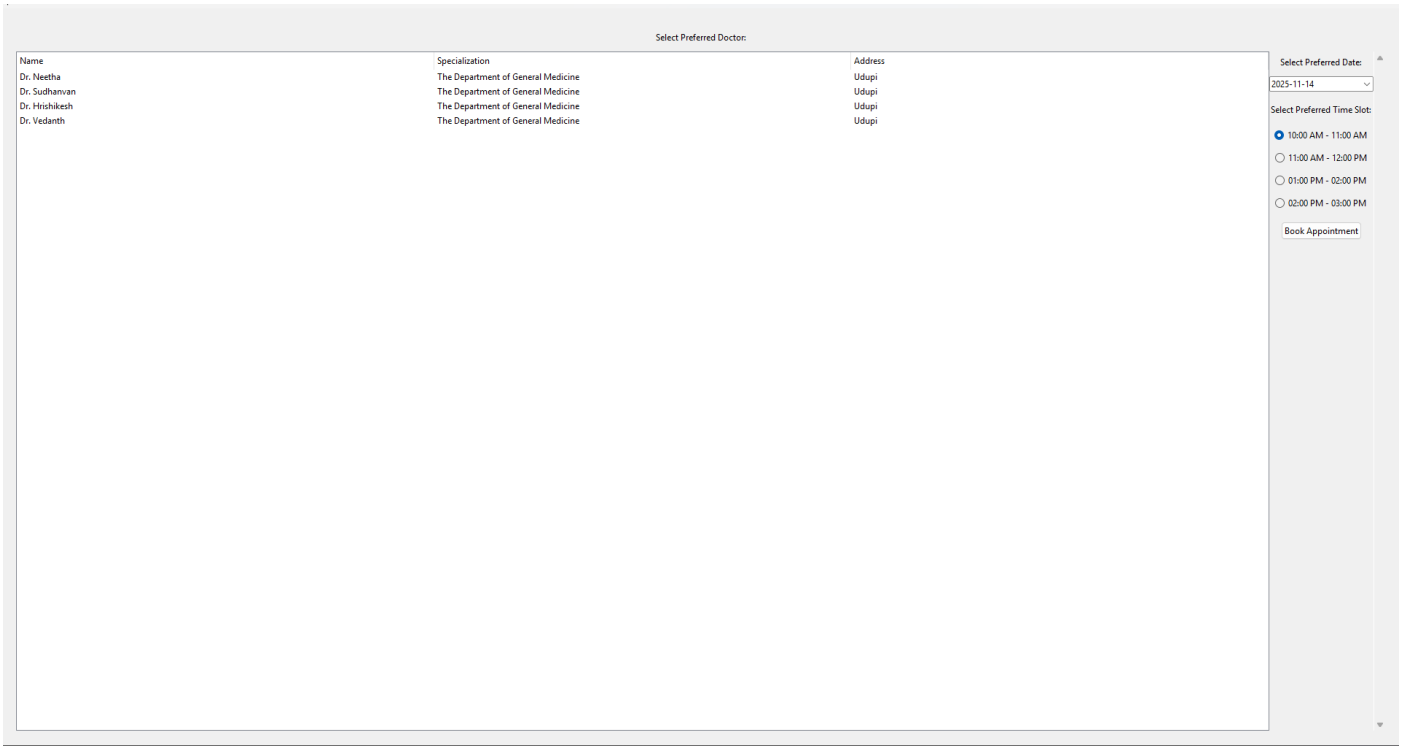
Login Screen:

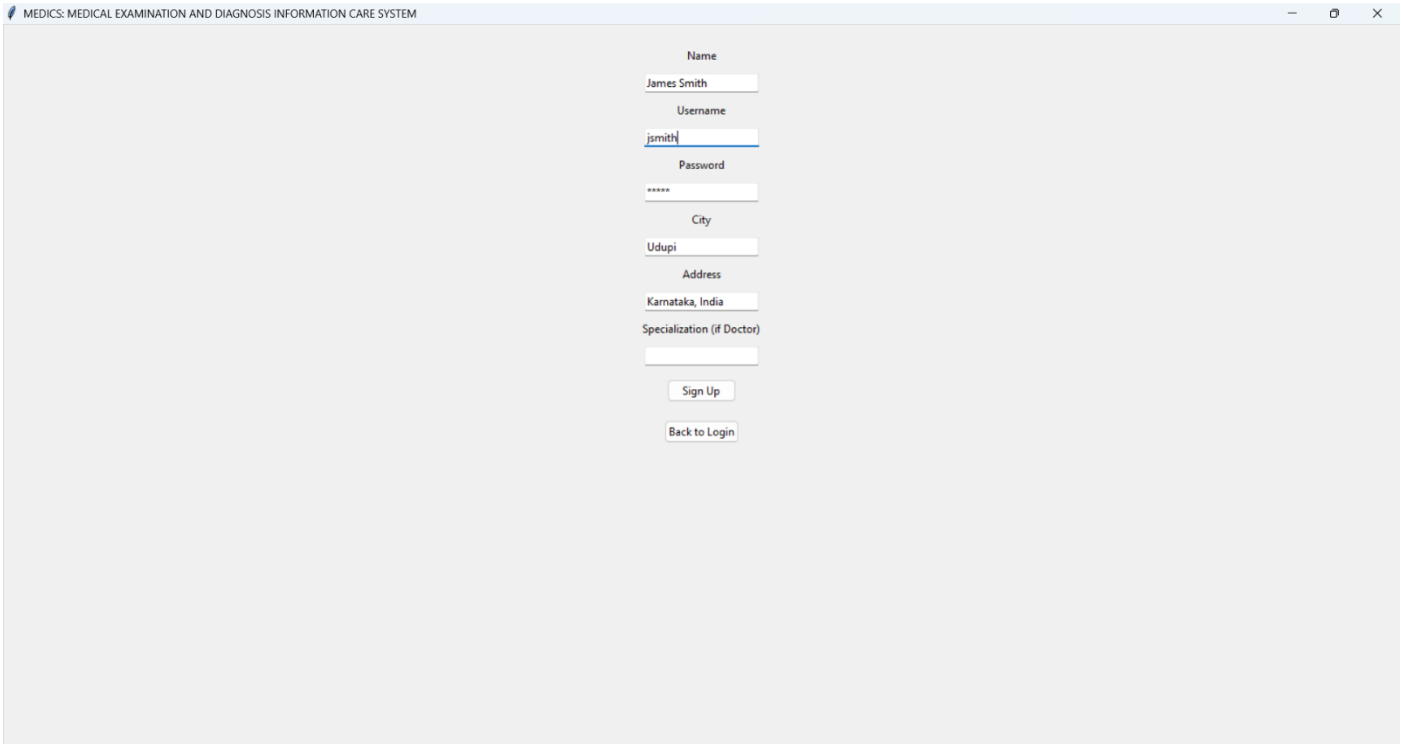


Signup Screen:

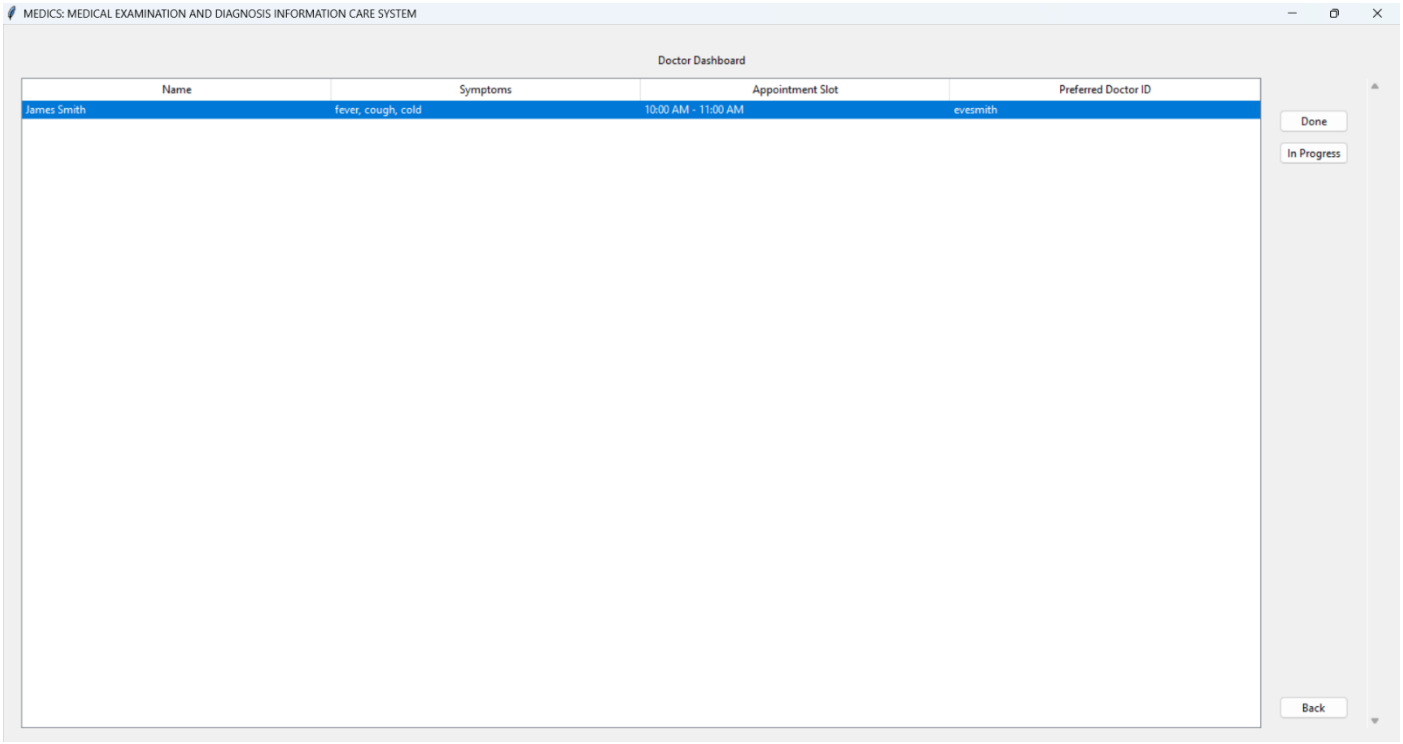


Patient Dashboard:





Doctor’s Dashboard:



ADVANTAGE

1. The system helps patients find the **right medical specialization** by analyzing the symptoms they enter. This prevents patients from booking with the wrong type of doctor and immediately directs them to the correct department, **streamlining the consultation process.**
2. To ensure all appointments are practical, both doctors and patients are **filtered by city**. This guarantees that a patient is only shown available doctors who are **geographically accessible** to them, making the selection process focused and efficient.
3. A doctor's dashboard displays **only the patients scheduled for that specific day**. This eliminates clutter and provides the doctor with a **clear, actionable queue**, allowing them to manage their time and focus on the immediate workflow.
4. Doctors can **mark a patient as "Done"** after the consultation. This action not only removes the patient from the day's list but also **clears the appointment slot** in the main database, keeping the system data current and the doctor's queue updated in real-time.

FUTURE ENHANCEMENTS

1. Defining more symptoms and integrating AI for determining to which department the patient should be routed.
2. Migration from Excel to a more robust database like SQL, Firebase because excel file are more prone to corruption and loss of data may also take place when large number of people use it simultaneously
3. Allowing users to reschedule or cancel appointments.
4. Implement a patient history feature to allow doctors to view past diagnoses and treatments.
5. Add multi language support to cater to a wider audience around the globe.
6. Implementation of a feature for doctors to prescribe medications and for patients to view their prescriptions.

REFERENCE

- Sumita Arora Computer Science Textbook Class XI
- Sumita Arora Computer Science Textbook Class XII
- Bosscoder academy : https://www.bosscoderacademy.com/
- Square Boat: https://www.squareboat.com/
- Smart Draw: https://www.smartdraw.com/
- Fotor: https://www.fotor.com/design/icon-maker/

Thank You