HOSPITAL CHATBOT

A Real-Time / Field-Based Research Project (22AM284) report submitted to the Jawaharlal Nehru Technological University, Hyderabad

Submitted by

DARMAYAGARI AKSHAYA **23B81A6667**
CHUNDURU NIVEDITHA **23B81A6695**
DOMALA POOJITHA **23B81A6698**

Under the guidance of
Dr. H. N. Lakshmi
Professor & Associate Dean

DEPARTMENT OF CSE (ARTIFICIAL INTELLIGENCE & MACHINE LEARNING)

CVR COLLEGE OF ENGINEERING

(***An Autonomous Institution, NAAC Accredited and Affiliated to JNTUH, Hyderabad***)
Vastunagar, Mangalpalli(V),Ibrahimpatnam(M),
Rangareddy (D), Telangana- 501 510

APRIL 2025

CVR COLLEGE OF ENGINEERING

(*An Autonomous institution , NAAC Accredited and Affiliated to JNTUH, Hyderabad*)

Vastunagar,Mangalpalli(V),Ibrahimpatnam(M), Rangareddy (D), Telangana- 501 510

DEPARTMENT OF CSE (ARTIFICIAL INTELLIGENCE & MACHINE LEARNING)

CERTIFICATE

This is to certify that the Real time/ Field-Based research project(22AM284) report entitled "**HOSPITAL CHATBOT"** is a record of work carried out by **Darmayagari Akshaya, Chunduru Niveditha, Domala Poojitha** submitted to Department of **CSE (ARTIFICIAL INTELLIGENCE & MACHINE LEARNING),** CVR College of Engineering, affiliated to Jawaharlal Nehru Technological University, Hyderabad during the year 2024-2025.

Project Guide	**Project Coordinator**	**Head of the Department**
Dr. H. N. Lakshmi	**Dr. M. Surya Bhupal Rao,**	**Dr. R. Usha Rani**
Professor & Associate Dean	**Ms. K. Navaneetha**	Professor
ET		CSE(AI&ML)

DECLARATION

We hereby declare that the Real time/ Field-Based research project (22AM284) report entitled "**HOSPITAL CHATBOT"** is an original work done and submitted to **CSE (ARTIFICIAL INTELLIGENCE & MACHINE LEARNING)** Department, CVR College of Engineering, affiliated to Jawaharlal Nehru Technological University, Hyderabad and it is a record of bonafide project work carried out by us under the guidance of **Dr. H. N. Lakshmi, Professor & Associate Dean, ET.**

We further declare that the work reported in this project has not been submitted, either in part or in full, for the award of any other degree or diploma in this Institute or any other Institute or University.

Signature of the Student

Signature of the Student

Signature of the Student

Date:

Place:

ACKNOWLEDGEMENT

We are thankful and fortunate enough to get constant encouragement, support, and guidance from all **Teaching staff of CSE (ARTIFICIAL INTELLIGENCE & MACHINE LEARNING)** Department who helped us in successfully completing this project work.

We thank **Dr. M. Surya Bhupal Rao, Ms. K. Navaneetha,** Project Coordinators**,** for their valuable guidance and support which helped us to complete the project work successfully.

We respect and thank our internal guide **Dr. H. N. Lakshmi**, Professor & Associate Dean, ET, for giving us all the support and guidance, which made us complete the project duly.

We would like to express heartful thanks to **Dr. R. Usha Rani**, Professor & Head of the Department, for providing us an opportunity to do this project and extending support and guidance

We would like to express heartful thanks to **Dr. H. N. Lakshmi,** Professor & Associate Dean, ET, for providing us an opportunity to do this project and extending support and guidance.

We thank our Vice-Principal **Prof. L. C. Siva Reddy** for providing excellent computing facilities and a disciplined atmosphere for doing our work.

We wish a deep sense of gratitude and heartfelt thanks to **Dr. Rama Mohan Reddy**, Principal and the **Management** for providing excellent lab facilities and tools. Finally, we thank all those guidance helpful to us in this regard.

List Of Figures

ABSTRACT

The Hospital Chatbot is an advanced, interactive desktop application developed using Java Swing, designed to streamline and simplify the hospital appointment process through a user-friendly chatbot interface. The system enables patients to seamlessly interact with the chatbot to perform vital functions such as viewing available doctors, booking new appointments, and cancelling or modifying existing ones. Supporting multiple hospitals, each with their unique set of specialized doctors, the chatbot aims to enhance healthcare accessibility and reduce administrative overhead by automating appointment management.

The application provides real-time updates to prevent double bookings by dynamically checking doctor-wise availability for each time slot. Patients can select a preferred hospital, browse through the list of doctors, and book an appropriate 20-minute time slot, excluding a predefined lunch break. The system alerts users immediately if a chosen slot is already occupied. Additionally, patients receive feedback notifications and a digital confirmation letter upon successful booking, which can be displayed or saved for verification. Cancellation options are available, except for hospitals like *Sunrise Hospital*, where cancellation is restricted by design.

Beyond appointment handling, the chatbot includes several helpful features such as a medicine information system, a symptom checker that suggests the relevant doctor specialization, emergency contact access, and a patient medical history tracker. It also provides daily health tips and reminders to promote well-being.

Visually, the interface incorporates intuitive design elements such as color-coded sections and neatly organized layouts, offering smooth and engaging user experience. Data is managed efficiently through a map-based storage mechanism, ensuring quick access and accurate tracking of appointments at the hospital and doctor. The Hospital Chatbot effectively bridges the gap between patients and healthcare facilities, promoting smarter, faster, and more reliable hospital management.

KEYWORDS

Hospital Chatbot, Appointment Booking System, Chatbot Automation, Hospital Scheduler

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

The Hospital Chatbot is an innovative and interactive desktop-based application developed using Java Swing, designed to simplify and automate the hospital appointment management process. It serves as a digital assistant for patients, enabling them to interact with a chatbot to perform essential appointment-related tasks such as viewing available doctors, booking appointments, modifying existing bookings, or canceling them with ease.

The system supports multiple hospitals, each having a distinct list of specialized doctors. Patients can choose their preferred hospital and browse through the list of available doctors along with their schedules. The chatbot ensures real-time updates to prevent double bookings by dynamically checking and updating the availability of time slots for each doctor. If a selected time slot is already occupied, the chatbot immediately notifies the user, maintaining the integrity and accuracy of the booking system.

In case of cancellations, the system automatically adjusts the available time slots, allowing other patients to book them. After a successful appointment booking, the system generates a confirmation letter, which is displayed on-screen for verification and future reference.

The user interface of the Hospital Chatbot is designed to be visually appealing, featuring color- coded themes, clean layouts, and an intuitive structure to enhance the user experience. The application utilizes a map-based data structure to efficiently manage and store appointment records for each hospital and doctor, ensuring smooth operation and quick data retrieval.

Overall, the Hospital Chatbot aims to improve the hospital appointment process by offering a convenient, automated, and reliable solution for both patients and hospital administrators. It enhances user satisfaction through a seamless interface and real-time functionalities while promoting efficient management of healthcare services.

1.1 PROBLEM STATEMENT

In today's fast-paced world, managing hospital appointments efficiently has become a critical need for both patients and healthcare providers. Traditional appointment booking systems are often plagued by issues such as double bookings, poor time slot management, lack of real-time updates, and user inconvenience. These limitations frequently result in patient dissatisfaction and operational inefficiencies within hospitals.

Additionally, patients often face difficulties in selecting the appropriate doctor, remembering scheduled appointments, or cancelling them when needed. These challenges are further compounded when dealing with hospital-specific rules and restrictions.

The **Hospital Chatbot** is developed to address these issues through a streamlined and interactive chatbot-based platform. The system simplifies the tasks of booking, viewing, and cancelling appointments by guiding users through an intuitive interface and providing real-time updates.
The primary challenges that the system aims to resolve include:

1. **Preventing Double Bookings** – Ensuring accurate time slot allocation to avoid scheduling conflicts.
2. **Efficient Time Slot Management** – Utilizing fixed 20-minute intervals with a designated one- hour lunch break to optimize doctor availability.
3. **Simplified Appointment Handling** – Allowing users to easily book, view, and cancel appointments with minimal technical knowledge.
4. **Hospital-Specific Logic** – Implementing custom restrictions like disabling cancellations at certain hospitals (e.g., Sunrise Hospital).
5. **User-Friendly Accessibility** – Designing an intuitive chatbot interface that caters to users of all ages and technical backgrounds.

By tackling these challenges, the Hospital Chatbot System aims to improve the overall patient experience, reduce hospital workload, and contribute to more effective healthcare appointment management.

1.2 PROBLEM OBJECTIVES

The Hospital Chatbot System is designed to streamline the appointment management process, making it more efficient and user-friendly. The main objectives are:

1. **Simplified Appointment Management**:

Allow patients to easily book, view, and cancel appointments through a chatbot interface, ensuring no double bookings.

2. **Real-Time Doctor and Slot Availability**:

Provide up-to-date information on doctor availability and time slots, offering 20-minute intervals with a 1-hour lunch break.

3. **Instant Appointment Updates**:

Send immediate notifications for booking confirmations, modifications, or cancellations, ensuring transparency.

4. **Hospital-Specific Rules**:

Implement restrictions, such as disabling appointment cancellations for "Sunrise Hospital," to meet hospital policies.

5. **User-Friendly Interface**:

Develop a simple, intuitive chatbot interface using Java Swing for easy navigation by users of all technical backgrounds.

5. **Appointment Confirmation**:

Generate confirmation letters containing appointment details for patients, enhancing clarity and reliability.

These objectives aim to improve the overall appointment scheduling process, ensuring seamlessness. experience for patients and hospital staff.

1.3 SOFTWARE & HARDWARE SPECIFICATIONS

1.3.1 SOFTWARE REQUIREMENTS

- Operating System: Compatible with Windows, macOS, and Linux for broad platform support.
- Programming Language: Built in Java (version 8 or higher) for cross-platform compatibility and strong object-oriented support.
- IDE: Java IDEs like IntelliJ IDEA, Eclipse, or NetBeans are used for development and debugging.
- Libraries: Java Swing and AWT for the GUI, with the Java Collections Framework for managing data.
- Database: No external database; uses in-memory structures like ArrayList and HashMap.
- Version Control (Optional): Git can be used for version control and collaboration.

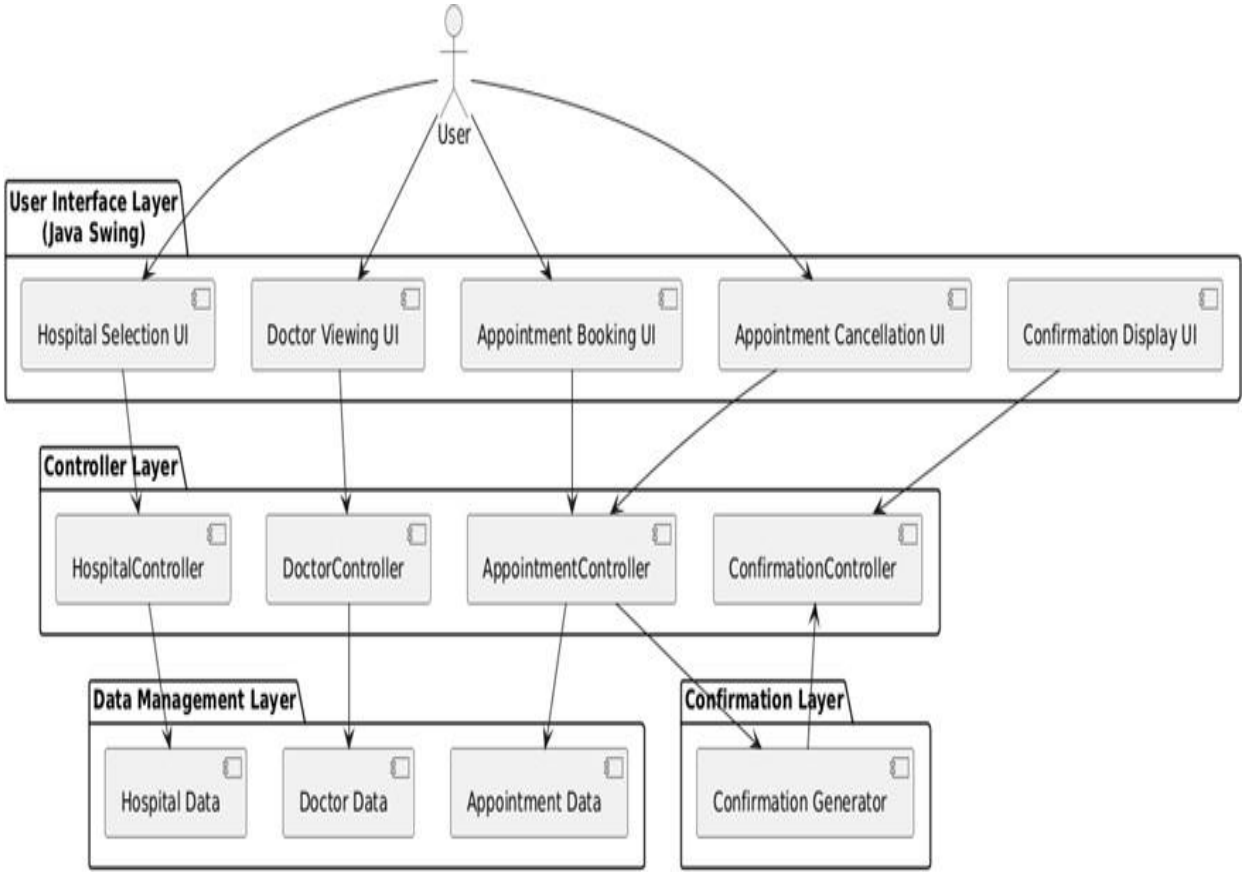
1.3.2 HARDWARE REQUIREMENTS

- Processor: Intel Core i3 or equivalent for smooth performance.
- RAM: 4 GB of RAM recommended for efficient operation.
- Storage: 500 MB of free space required for project files and resources.
- Display: Minimum screen resolution of 1024x768 for proper UI display.
- Peripherals: Keyboard and mouse for user interaction; optional printer for reports.

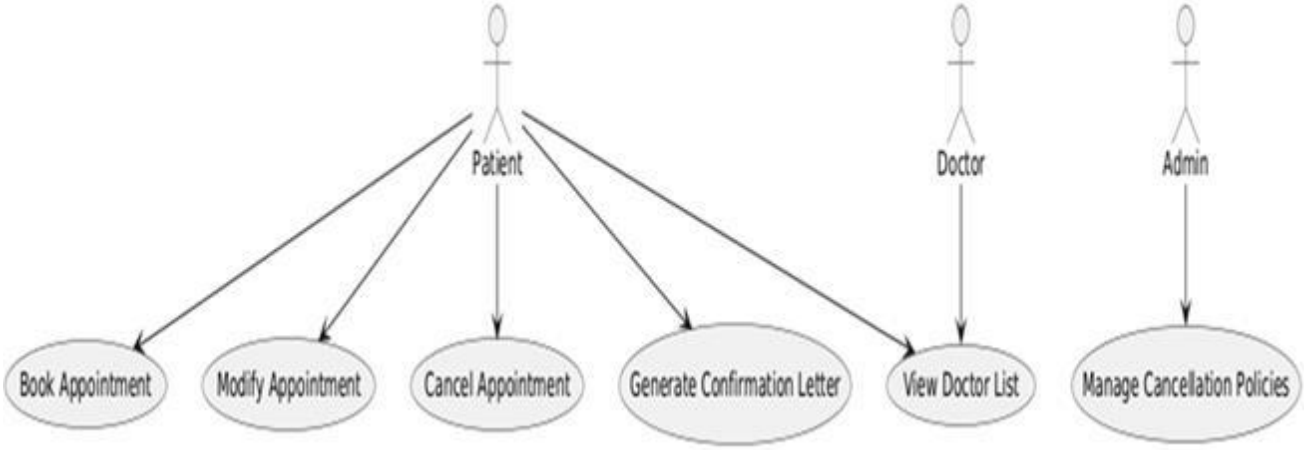
CHAPTER 2

DESIGN & METHODOLGY

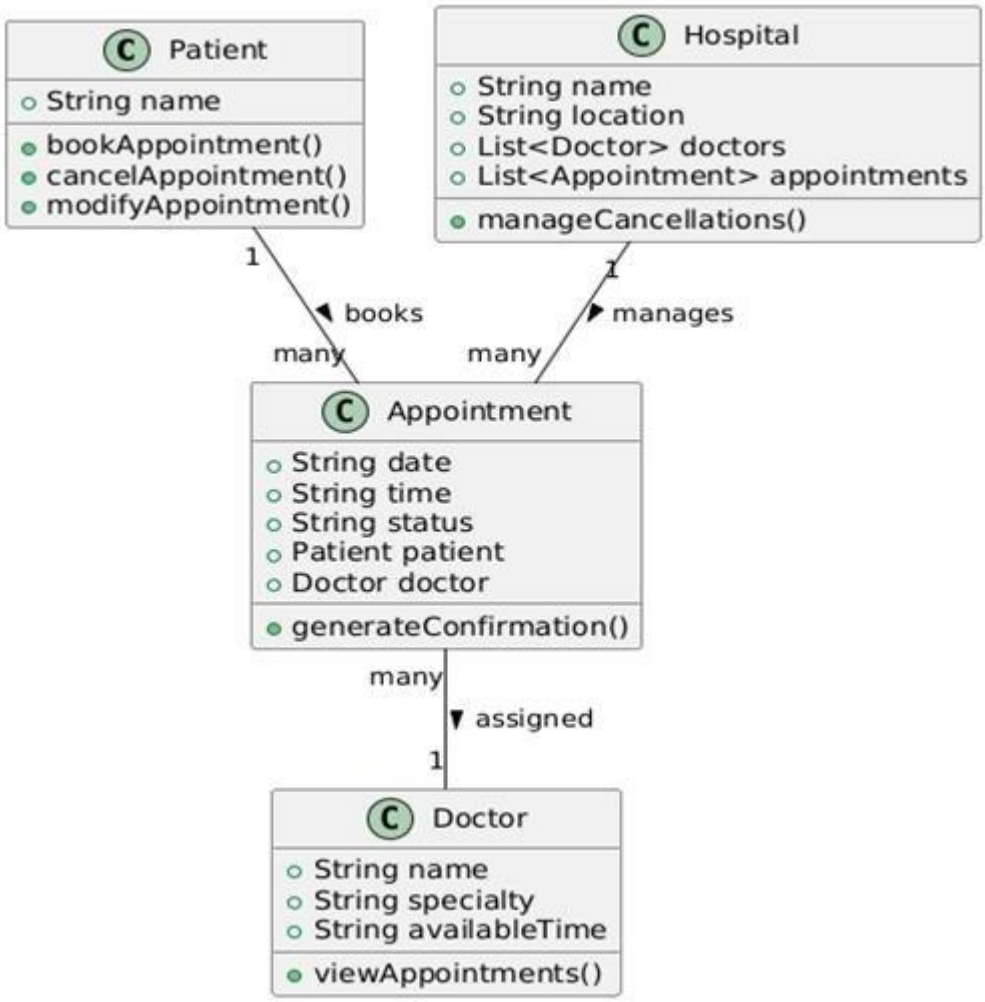
2.1 SYSTEM ARCHITECTURE



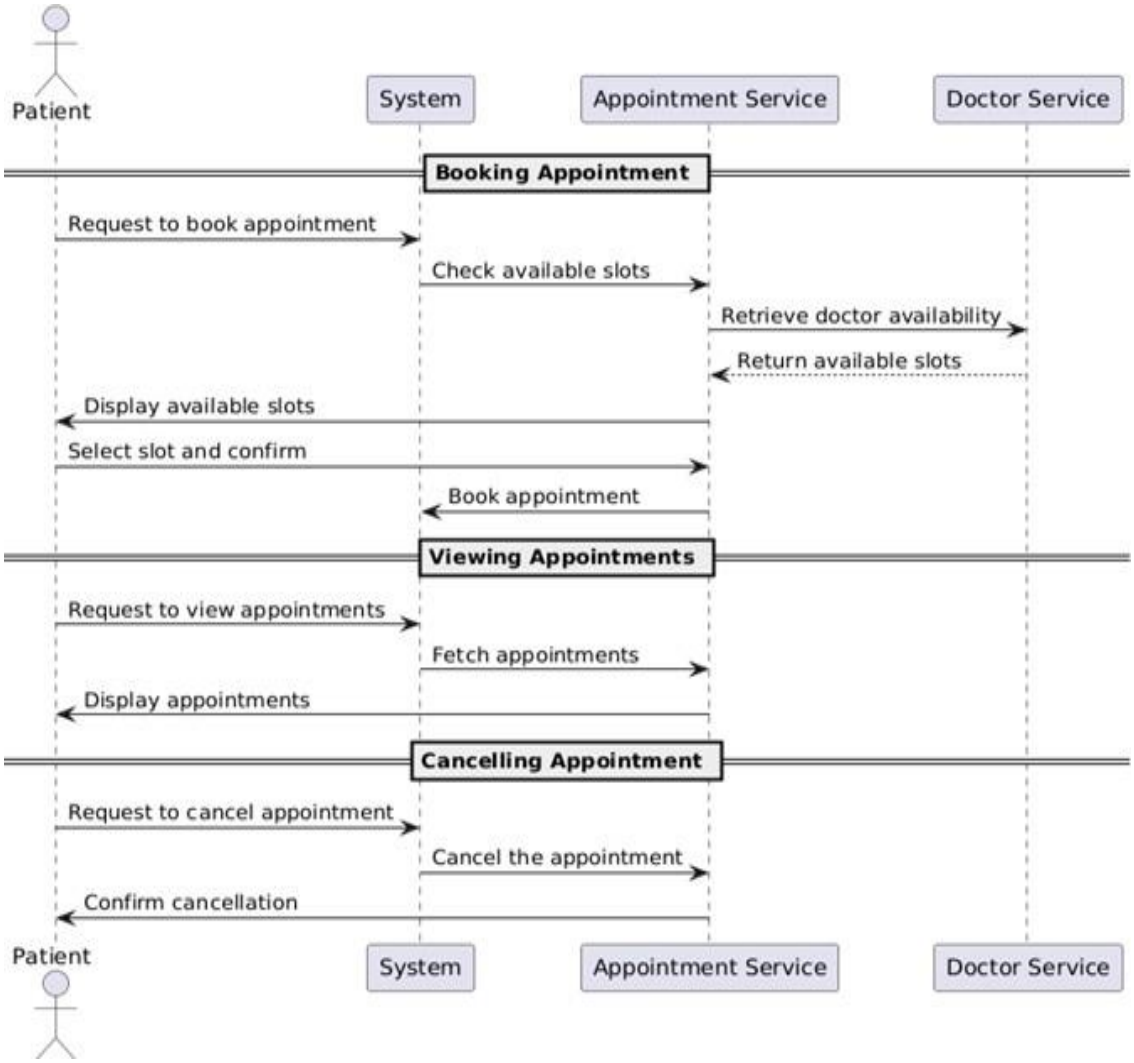
2.2.1 USE CASE DIAGRAM



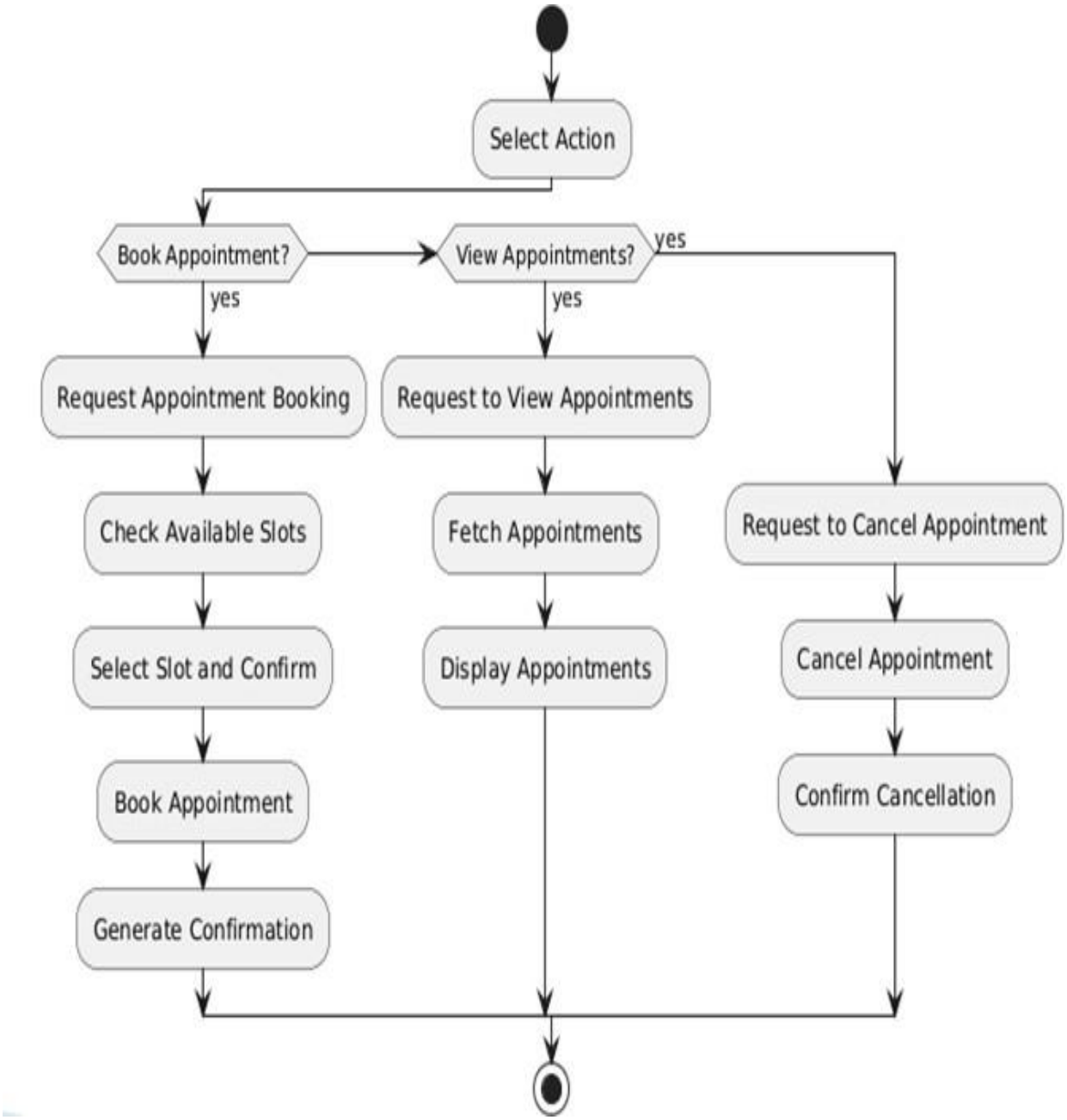
2.2.2 CLASS DIAGRAM



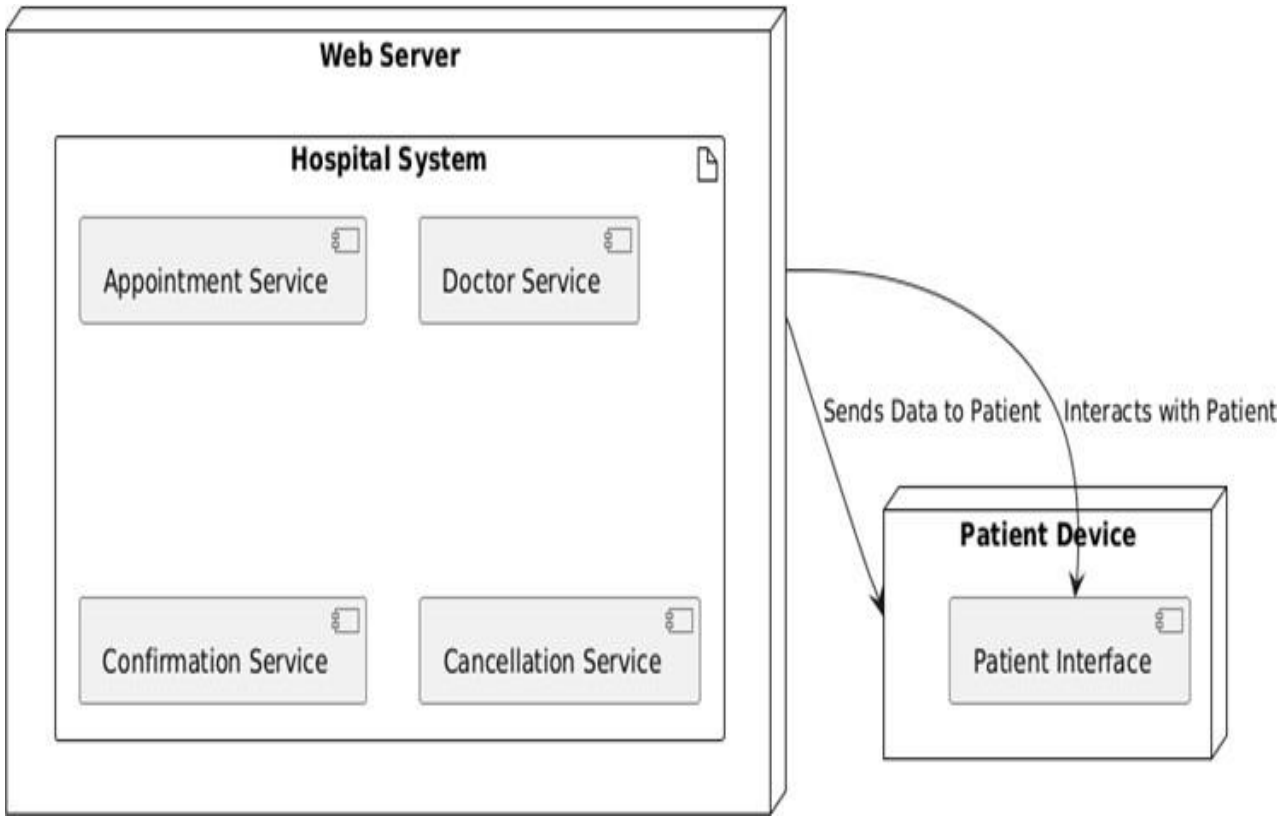
2.2.3 SEQUENCE DIAGRAM



2.2.4 ACTIVITY DIAGRAM



2.2.5 DEPLOYMENT DIAGRAM



2.3 TECHNOLOGY DESCRIPTION

The ***Hospital Chatbot*** is a Java-based desktop application developed using the Java Swing framework for its graphical user interface (GUI). It aims to simplify hospital appointment management through an intuitive chatbot-like interface that interacts with users in real-time. The system is designed to handle tasks such as booking, cancelling, and modifying appointments, while also providing additional features like a medicine information system, symptom checker, and health tips.

Core Technologies Used:

1. **Java (JDK 8 or above):**

Java serves as the core programming language for implementing the system's logic. It offers platform independence, object-oriented features, and robust exception handling which are ideal for application-level development.

2. **Java Swing:**

Swing is used for designing the GUI components such as buttons, text fields, tables, dialogs, and frames. It allows building a responsive, interactive, and visually friendly interface for the chatbot and other modules.

3. **Event-Driven Programming:**

The application heavily uses event listeners and handlers to respond to user actions, such as clicking buttons or selecting dropdown options. This ensures smooth and interactive user experience.

4. **Data Structures (ArrayList, HashMap):**

These collections are used for managing doctor schedules, storing appointment details, filtering booked slots, and maintaining user records like medical history.

5. **File Handling (optional or as an enhancement):**

To store persistent data such as past appointments or user profiles, file handling techniques may be integrated using File Reader, File Writer, or serialization in Java.

6. **Swing Timers and Date Utilities:**

The system uses time validation features (20-minute slots, 1-hour lunch break) by leveraging Java's LocalDate, LocalTime, and DateTimeFormatter classes to ensure accurate time-bound appointment scheduling.

CHAPTER 3

IMPLEMETATION & TESTING

3.1 CODE SNIPPETS

Main frame

This part sets up the main window and adds buttons for each hospital.

```
public static void main (String[] args) {
    hospitalDoctors.put("City Hospital", new String[]{"Dr. A. Guru Prakash (Cardiologist)", "Dr.
    Krishnaveni (Dermatologist)", "Dr. Rama Shankar (Neurology)"});
    hospitalDoctors.put("Green Valley Hospital", new String[]{"Dr. Vikram Kalra (Orthopedic)", "Dr. Shilpa
    (Pediatrician)"});
    hospitalDoctors.put("Sunrise Hospital", new String[]{"Dr. Surendar (Neurologist)", "Dr. Rakesh (ENT
    Specialist)"});

    SwingUtilities.invokeLater(() -> {
        JFrame mainFrame = new JFrame("Hospital Chatbot");
        mainFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        mainFrame.setSize(600, 400);
        mainFrame.setLayout(new FlowLayout());
        mainFrame.getContentPane().setBackground(new Color(240, 255, 255));

        JButton cityHospitalButton = new JButton("City Hospital");
        JButton greenValleyHospitalButton = new JButton("Green Valley Hospital");
        JButton sunriseHospitalButton = new JButton("Sunrise Hospital");
        JButton exitButton = new JButton("Exit");

        mainFrame.add(cityHospitalButton);
        mainFrame.add(greenValleyHospitalButton);
        mainFrame.add(sunriseHospitalButton);
        mainFrame.add(exitButton);

        cityHospitalButton.addActionListener(e -> showHospitalOptions("City Hospital"));
        greenValleyHospitalButton.addActionListener(e -> showHospitalOptions("Green Valley Hospital"));
        sunriseHospitalButton.addActionListener(e -> showHospitalOptions("Sunrise Hospital"));
        exitButton.addActionListener(e -> System.exit(0));
```

```
        mainFrame.setVisible(true);
    });
}
```

Viewing Doctors in a Hospital

This code displays a list of doctors available at the selected hospital.

```
private static void viewDoctors(String hospital) {
    JFrame doctorsFrame = new JFrame("Available Doctors at " + hospital);
    doctorsFrame.setSize(400, 300);
    doctorsFrame.setLayout(new BorderLayout());

    JTextArea doctorsList = new JTextArea();
    doctorsList.setEditable(false);
    StringBuilder sb = new StringBuilder("Doctors at " + hospital + ":\n\n");
    for (String doctor : hospitalDoctors.get(hospital)) {
        sb.append("- ").append(doctor).append("\n");
    }
    doctorsList.setText(sb.toString());
    doctorsList.setFont(new Font("Courier New", Font.PLAIN, 14));

    doctorsFrame.add(new JScrollPane(doctorsList), BorderLayout.CENTER);
    doctorsFrame.setVisible(true);
}
```

Booking an Appointment

This code sets up a booking form where users can select the doctor, time slot, and date.

```
private static void bookAppointment(String hospital) {
    JFrame bookingFrame = new JFrame("Book Appointment at " + hospital);
    bookingFrame.setSize(400, 400);
    bookingFrame.setLayout(new GridLayout(8, 1));
```

```
JComboBox<String> doctorDropdown = new JComboBox<>(hospitalDoctors.get(hospital));
JComboBox<String> timeSlotDropdown = new JComboBox<>(timeSlots);
JSpinner dateSpinner = new JSpinner(new SpinnerDateModel());
JSpinner.DateEditor dateEditor = new JSpinner.DateEditor(dateSpinner, "yyyy-MM-dd");
dateSpinner.setEditor(dateEditor);
JTextField nameField = new JTextField();
JButton bookButton = new JButton("Book");
JLabel statusLabel = new JLabel();

bookButton.addActionListener(e -> {
    String doctor = (String) doctorDropdown.getSelectedItem();
    String timeSlot = (String) timeSlotDropdown.getSelectedItem();
    String name = nameField.getText().trim();

    if (name.isEmpty()) {
        statusLabel.setText("Invalid name! Please enter your name.");
        return;
    }

    Date selectedDate = (Date) dateSpinner.getValue();
    SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd");
    String dateString = dateFormat.format(selectedDate);

    Map<String, Map<String, Map<String, String>>> hospitalAppointments =
    appointments.get(hospital);
    Map<String, Map<String, String>> doctorAppointments =
    hospitalAppointments.getOrDefault(doctor, new HashMap<>());
    Map<String, String> dateAppointments = doctorAppointments.getOrDefault(dateString, new
    HashMap<>());

    if (dateAppointments.containsKey(timeSlot)) {
        statusLabel.setText("Time slot already booked on this date!");
    } else {
        dateAppointments.put(timeSlot, name);
        doctorAppointments.put(dateString, dateAppointments);
```

```
            hospitalAppointments.put(doctor, doctorAppointments);
            appointments.put(hospital, hospitalAppointments);
            statusLabel.setText("Appointment booked successfully!");
            showAppointmentLetter(name, hospital, doctor, timeSlot, dateString);
    }
    });


    bookingFrame.add(doctorDropdown);
    bookingFrame.add(dateSpinner);
    bookingFrame.add(timeSlotDropdown);
    bookingFrame.add(nameField);
    bookingFrame.add(bookButton);
    bookingFrame.add(statusLabel);

    bookingFrame.setVisible(true);
 }
```

Cancelling an Appointment

This code allows users to cancel their appointment by selecting a doctor, time slot, and date.

```
 private static void cancelAppointment(String hospital) {
    JFrame cancelFrame = new JFrame("Cancel Appointment at " + hospital);
    cancelFrame.setSize(400, 300);
    cancelFrame.setLayout(new GridLayout(7, 1));

    JComboBox<String> doctorDropdown = new JComboBox<>(hospitalDoctors.get(hospital));
    JComboBox<String> timeSlotDropdown = new JComboBox<>(timeSlots);
    JSpinner dateSpinner = new JSpinner(new SpinnerDateModel());
    JSpinner.DateEditor dateEditor = new JSpinner.DateEditor(dateSpinner, "yyyy-MM-dd");
    dateSpinner.setEditor(dateEditor);
    JButton cancelButton = new JButton("Cancel Appointment");
    JLabel statusLabel = new JLabel();
    cancelButton.addActionListener(e -> {
        String doctor = (String) doctorDropdown.getSelectedItem();
        String timeSlot = (String) timeSlotDropdown.getSelectedItem;
```

```
        Date selectedDate = (Date) dateSpinner.getValue();
        SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd");
        String dateString = dateFormat.format(selectedDate);

        Map<String, Map<String, Map<String, String>>> hospitalAppointments =
        appointments.get(hospital);
        Map<String, Map<String, String>> doctorAppointments =
        hospitalAppointments.getOrDefault(doctor, new HashMap<>());
        Map<String, String> dateAppointments = doctorAppointments.getOrDefault(dateString, new
        HashMap<>());

        if (dateAppointments.containsKey(timeSlot)) {
            dateAppointments.remove(timeSlot);
            if (dateAppointments.isEmpty()) {
                doctorAppointments.remove(dateString);
            }
            statusLabel.setText("Appointment canceled successfully!");
        } else {
            statusLabel.setText("No appointment found at the selected time slot on this date!");
}
});

    cancelFrame.add(doctorDropdown);
    cancelFrame.add(dateSpinner);
    cancelFrame.add(timeSlotDropdown);
    cancelFrame.add(cancelButton);
    cancelFrame.add(statusLabel);

    cancelFrame.setVisible(true);
 }
```

Viewing Booked Appointments

This code allows users to view appointments based on doctor or date filters.

```
private static void viewBookedAppointments(String hospital) {
```

```
    JFrame filterFrame = new JFrame("Filter Appointments at " + hospital);
    filterFrame.setSize(400, 300);
    filterFrame.setLayout(new GridLayout(4, 1));

    JComboBox<String> doctorDropdown = new JComboBox<>(hospitalDoctors.get(hospital));
    doctorDropdown.insertItemAt("All Doctors", 0);
    doctorDropdown.setSelectedIndex(0);

    JSpinner dateSpinner = new JSpinner(new SpinnerDateModel());
    JSpinner.DateEditor dateEditor = new JSpinner.DateEditor(dateSpinner, "yyyy-MM-dd");
    dateSpinner.setEditor(dateEditor);
    JButton viewButton = new JButton("View Appointments");

    viewButton.addActionListener(e -> {
        String selectedDoctor = (String) doctorDropdown.getSelectedItem();
        Date selectedDate = (Date) dateSpinner.getValue();
        SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd");
        String dateString = dateFormat.format(selectedDate);

        viewAppointments(hospital, selectedDoctor, dateString);
    });

    filterFrame.add(doctorDropdown);
    filterFrame.add(dateSpinner);
    filterFrame.add(viewButton);
    filterFrame.setVisible(true);
}
```

8. Displaying the Appointment Letter

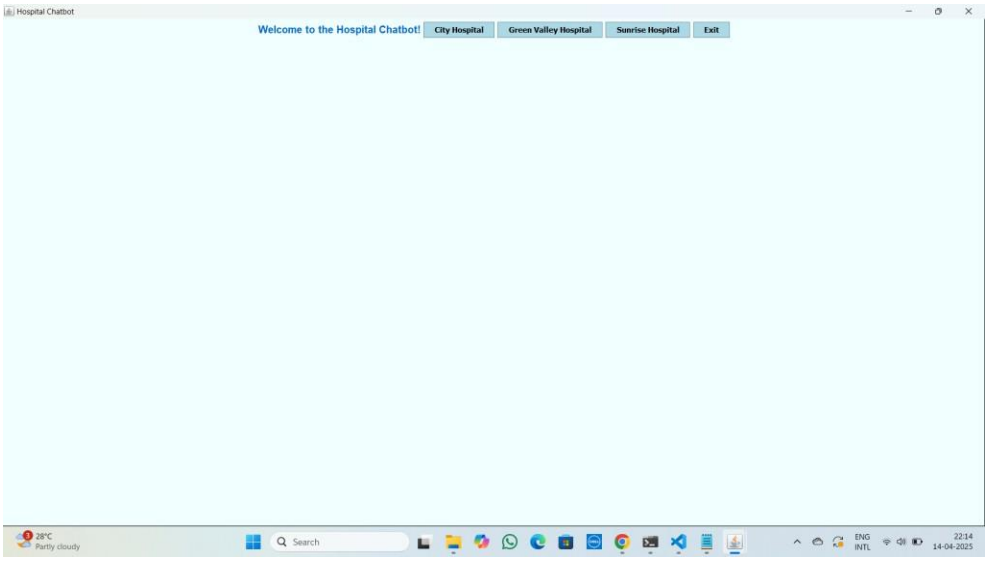
This method creates an appointment letter confirming the appointment details.

```
private static void showAppointmentLetter(String name, String hospital, String doctor, String
timeSlot, String dateString) {
    JFrame letterFrame = new JFrame("Appointment Confirmation");
    letterFrame.setSize(400, 300);
```

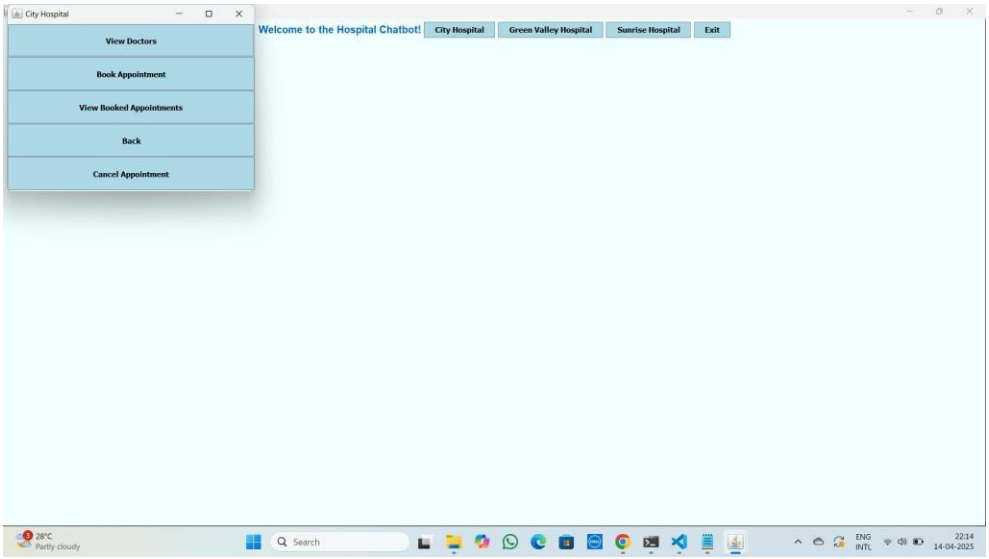
```
    letterFrame.setLayout(new BorderLayout());

    JTextArea letterArea = new JTextArea();
    letterArea.setEditable(false);
    String appointmentDetails = "Appointment Confirmation\n\n" +
            "Hospital: " + hospital + "\n" +
            "Doctor: " + doctor + "\n" +
            "Date: " + dateString + "\n" +
            "Time Slot: " + timeSlot + "\n\n" +
            "Patient: " + name + "\n\n" +
            "Thank you for choosing our hospital.";

    letterArea.setText(appointmentDetails);
    letterArea.setFont(new Font("Courier New", Font.PLAIN, 14));

    letterFrame.add(new JScrollPane(letterArea), BorderLayout.CENTER);
    letterFrame.setVisible(true);
}
```

3.2 SCREENSHOTS

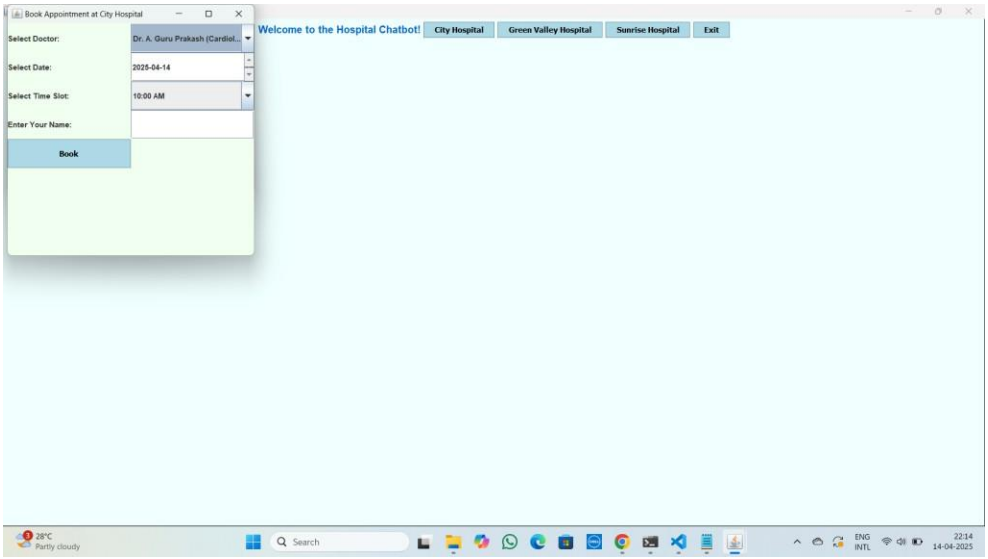
Main frame that gets opened as soon as you run the application:



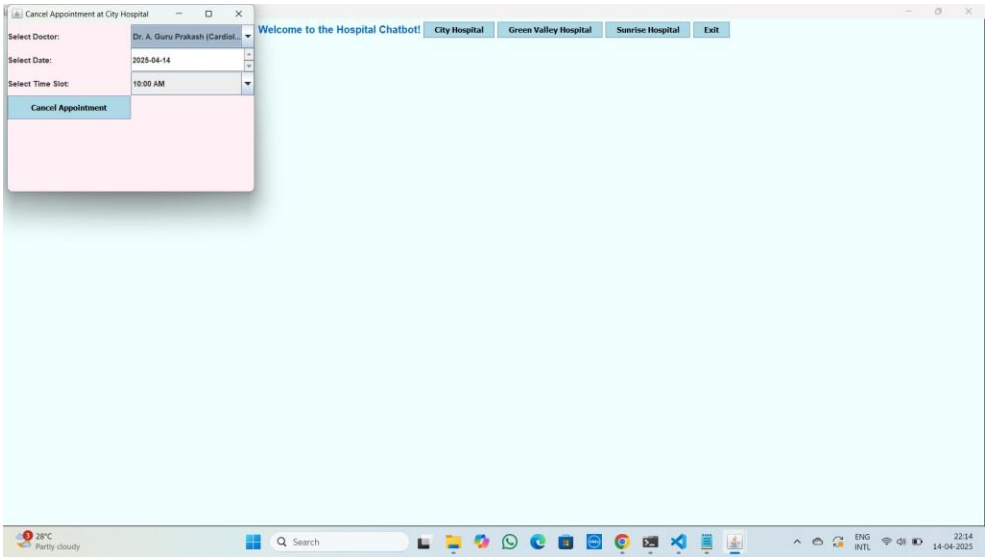
After selecting a hospital, you can see



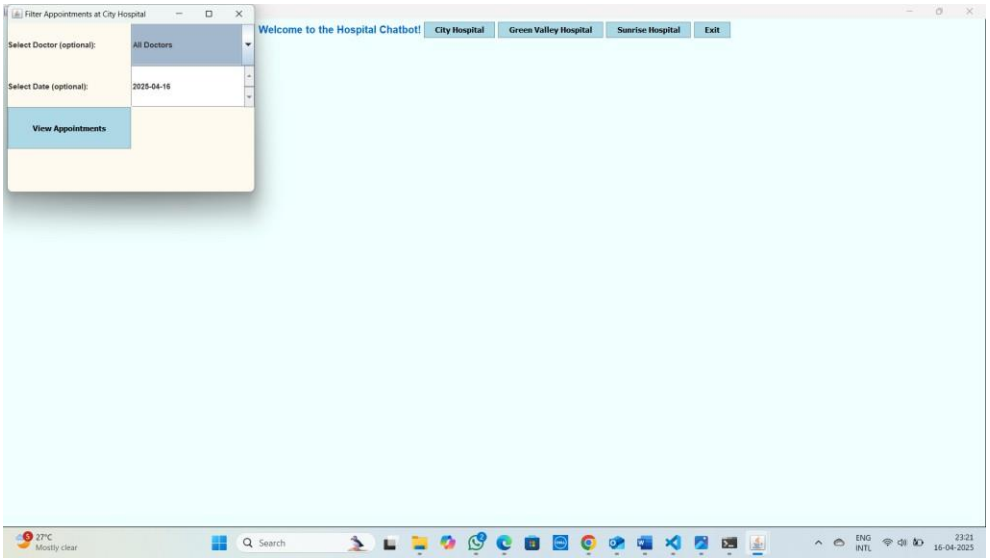
To book an appointment the frame will look like



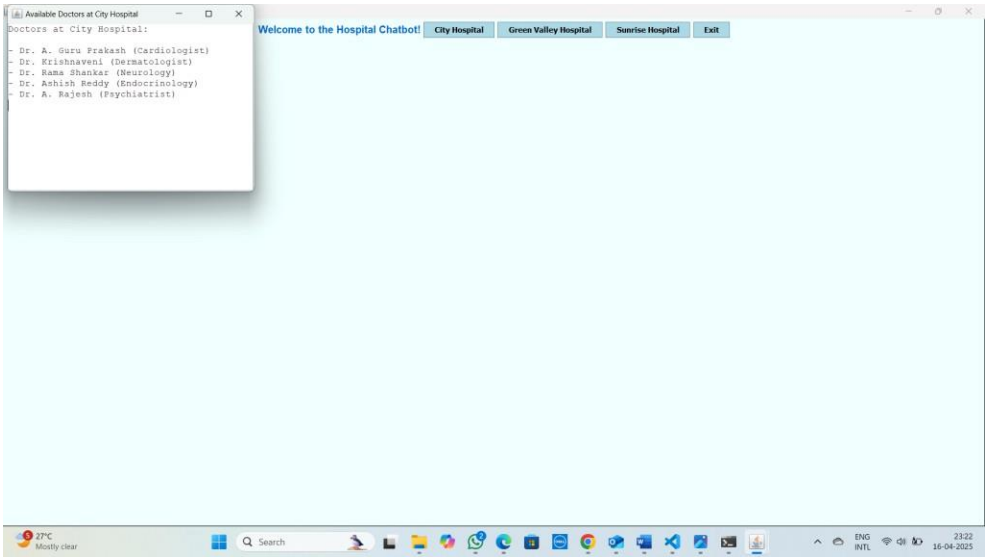
To cancel an appointment the frame will look like



To view booked appointments the frame will look like



To view available doctors in a hospital, the screen will look like



3.3 TESTCASES

Test Cases for Viewing Doctors

Test Case ID	Test Scenario	Input	Expected Output	Result
TC01	View doctors for valid hospital	Hospital: *Sunrise Hospital*	List of doctors available at *Sunrise Hospital*	Pass

Test Cases for Booking Appointments

Test Case ID	Test Scenario	Input	Expected Output	Result
TC04	Book appointment with valid data	Hospital: *Sunrise Hospital*, Doctor: *Dr. Rakesh*, Date: *2025-04-18*, Time: *10:00 AM*	"Appointment booked successfully!" message shown	Pass
TC05	Book appointment with invalid date	Hospital: *Sunrise Hospital*, Doctor: *Dr. Rakesh*, Date: *2025-04-15*, Time: *10:00 AM*	"Invalid date! Please choose a future date." message displayed	Pass
TC06	Book appointment with invalid doctor	Hospital: *Sunrise Hospital*, Doctor: *Dr. Unknown*, Date: *2025-04-18*, Time: *10:00 AM*	"Doctor not found!" message displayed	

Test Cases for Cancelling Appointments

Test Case ID	Test Scenario	Input	Expected Output	Result
TC07	Cancel existing appointment	Correct doctor, date, and time of booked appointment	"Appointment canceled successfully!" message shown	Pass
TC08	Cancel non-existent appointment	Random doctor, date, and time	"No appointment found at the selected time slot on this date!"	Pass
TC09	Cancel appointment for Sunrise Hospital	Hospital: *Sunrise Hospital*, Doctor: *Dr. Shilpa*, Date: *2025-*	"Cancellation not allowed for Sunrise Hospital"	

Test Case ID	Test Scenario	Input	Expected Output	Result
	(restricted)	*04-18*, Time: *10:00 AM*	message shown	

Test Cases for Viewing Booked Appointments

Test Case ID	Test Scenario	Input	Expected Output	Result
TC10	View all booked appointments for selected doctor	Doctor: *Dr. Rakesh*, Date: *2025-04-18*	List of booked appointments for selected doctor and date	Pass
TC11	View booked appointments for specific date	Date: *2025-04-18*	List of all booked appointments for the selected date	Pass
TC12	View booked appointments when no appointments are made	Date: *2025-04-20*	"No appointments found for this date" message shown	Pass

CHAPTER 4

CONCLUSION

The Hospital Chatbot project successfully demonstrates the integration of a chatbot-based user interface with a hospital appointment management system using Java Swing. This system simplifies the process of booking, cancelling, and managing doctor appointments by providing an intuitive, interactive GUI that guides users through each step.

Key functionalities such as:

1. Real-time appointment booking with 20-minute time slots,
2. Doctor and date-based appointment filtering,
3. Restrictions like preventing cancellations at Sunrise Hospital,
4. And user-friendly components for viewing doctors and confirming appointments

make the system highly efficient and practical for hospital usage.

Additionally, the chatbot interface adds a conversational element that enhances the user experience, reducing the complexity often associated with traditional forms or websites. The modular design, robust validations, and seamless interactions between components ensure reliability and maintainability.

This project not only fulfils its intended purpose but also opens doors to future enhancements like online consultation integration, medicine reminders, and more AI-based responses — making it a scalable and innovative solution for modern healthcare facilities.

BIBLIOGRAPHY

1. Oracle Java Swing Documentation

   Official documentation for Java Swing components and GUI programming.

   https://docs.oracle.com/javase/8/docs/technotes/guides/swing/

2. Oracle Java Platform SE 8 API Specification

   Reference for Java classes, interfaces, and methods used in the project.

   https://docs.oracle.com/javase/8/docs/api/

3. GeeksforGeeks – Java Swing Tutorial

   Easy-to-understand tutorials for implementing Swing components and event handling.

   https://www.geeksforgeeks.org/java-swing/

4. GeeksforGeeks – Event Handling in Java

   Guide to ActionListeners and event-driven programming in Java.

   https://www.geeksforgeeks.org/event-handling-in-java/

5. TutorialsPoint – Java Swing

   Detailed lessons and examples for building Java GUI applications.

   https://www.tutorialspoint.com/swing/index.htm

6. TutorialsPoint – Java Date and Time API

   Useful for working with date validations in the appointment system.

   https://www.tutorialspoint.com/java8/java8_datetime_api.htm

7. Oracle – Handling Events in Swing

   Official guide for implementing user interaction in Swing applications.

   https://docs.oracle.com/javase/tutorial/uiswing/events/index.html

APPENDIX (SOURCE CODE)

```
// Import required libraries
import javax.swing.*;
import java.awt.*;
import java.text.SimpleDateFormat;
import java.util.*;

public class HospitalChatbotSwingc4 {
    // Map to store hospital names and their corresponding list of doctors
    private static final Map<String, String[]> hospitalDoctors = new HashMap<>();

    // Predefined time slots for appointment booking
    private static final String[] timeSlots = {
        "10:00 AM", "10:20 AM", "10:40 AM", "11:00 AM", "11:20 AM", "11:40 AM",
        "12:00 PM", "12:20 PM", "12:40 PM", "2:00 PM", "2:20 PM", "2:40 PM",
        "3:00 PM", "3:20 PM", "3:40 PM", "4:00 PM", "4:20 PM", "4:40 PM", "5:00 PM"
    };

    // Map to store appointment details
    private static final Map<String, Map<String, Map<String, Map<String, String>>>> appointments
    = new HashMap<>();
    private static JFrame mainFrame;

    // Main method to run the program
    public static void main(String[] args) {
        // Initialize the doctors for each hospital
        hospitalDoctors.put("City Hospital", new String[]{
            "Dr. A. Guru Prakash (Cardiologist)", "Dr. Krishnaveni (Dermatologist)", "Dr. Rama
        Shankar (Neurology)",
            "Dr. Ashish Reddy (Endocrinology)", "Dr. A. Rajesh (Psychiatrist)"
        });
            hospitalDoctors.put("Green Valley Hospital", new String[]{
            "Dr. Vikram Kalra (Orthopedic)", "Dr. Shilpa (Pediatrician)", "Dr. Naresh (Cardiologist)",
            "Dr. Kiranmayee (Pulmonologist)", "Dr. Ravinder (ENT Specialist)"
        });
```

```
hospitalDoctors.put("Sunrise Hospital", new String[]{
    "Dr. Surendar (Neurologist)", "Dr. Rakesh (ENT Specialist)", "Dr. Swapna (Gynecologist)",
    "Dr. Pallavi (Pediatrician)", "Dr. Vinay Kumar (Dentist)"
});

// Initialize appointments for each hospital
for (String hospital : hospitalDoctors.keySet()) {
    appointments.put(hospital, new HashMap<>());
}

// Set up the main frame for the chatbot interface
SwingUtilities.invokeLater(() -> {
    mainFrame = new JFrame("Hospital Chatbot");
    mainFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    mainFrame.setSize(600, 400);
    mainFrame.setLayout(new FlowLayout());
    mainFrame.getContentPane().setBackground(new Color(240, 255, 255));

    // Create and style welcome label
    JLabel welcomeLabel = new JLabel("Welcome to the Hospital Chatbot!");
    welcomeLabel.setFont(new Font("Arial", Font.BOLD, 16));
    welcomeLabel.setForeground(new Color(0, 102, 204));

    // Create buttons for hospital selection
    JButton cityHospitalButton = new JButton("City Hospital");
    JButton greenValleyHospitalButton = new JButton("Green Valley Hospital");
    JButton sunriseHospitalButton = new JButton("Sunrise Hospital");
    JButton exitButton = new JButton("Exit");

    // Style the buttons
    styleButton(cityHospitalButton);
    styleButton(greenValleyHospitalButton);
    styleButton(sunriseHospitalButton);
    styleButton(exitButton);
```

```
        // Add components to the main frame
        mainFrame.add(welcomeLabel);
        mainFrame.add(cityHospitalButton);
        mainFrame.add(greenValleyHospitalButton);
        mainFrame.add(sunriseHospitalButton);
        mainFrame.add(exitButton);

        // Button actions
        cityHospitalButton.addActionListener(e -> showHospitalOptions("City Hospital"));
        greenValleyHospitalButton.addActionListener(e -> showHospitalOptions("Green Valley
        Hospital"));
        sunriseHospitalButton.addActionListener(e -> showHospitalOptions("Sunrise Hospital"));
        exitButton.addActionListener(e -> System.exit(0));

        // Make the main frame visible
        mainFrame.setVisible(true);
    });
}

// Method to show hospital-specific options
private static void showHospitalOptions(String hospitalName) {
    JFrame hospitalFrame = new JFrame(hospitalName);
    hospitalFrame.setSize(400, 300);
    hospitalFrame.setLayout(new GridLayout(5, 1));
    hospitalFrame.getContentPane().setBackground(new Color(240, 255, 240));

    // Create buttons for hospital options
    JButton viewDoctorsButton = new JButton("View Doctors");
    JButton bookAppointmentButton = new JButton("Book Appointment");
    JButton viewBookedAppointmentsButton = new JButton("View Booked Appointments");
    JButton backButton = new JButton("Back");

    // Style the buttons
    styleButton(viewDoctorsButton);
    styleButton(bookAppointmentButton);
```

```
    styleButton(viewBookedAppointmentsButton);
    styleButton(backButton);

    // Button actions
    viewDoctorsButton.addActionListener(e -> viewDoctors(hospitalName));
    bookAppointmentButton.addActionListener(e -> bookAppointment(hospitalName));
    viewBookedAppointmentsButton.addActionListener(e ->
     viewBookedAppointments(hospitalName));
    backButton.addActionListener(e -> hospitalFrame.dispose());

    // Add buttons to the hospital frame
    hospitalFrame.add(viewDoctorsButton);
    hospitalFrame.add(bookAppointmentButton);
    hospitalFrame.add(viewBookedAppointmentsButton);
    hospitalFrame.add(backButton);

    // If the hospital is not Sunrise Hospital, show a cancel appointment option
    if (!hospitalName.equals("Sunrise Hospital")) {
        JButton cancelAppointmentButton = new JButton("Cancel Appointment");
        styleButton(cancelAppointmentButton);
        cancelAppointmentButton.addActionListener(e -> cancelAppointment(hospitalName));
        hospitalFrame.add(cancelAppointmentButton);
    }

    // Make the hospital frame visible
    hospitalFrame.setVisible(true);
}

// Method to view doctors at a particular hospital
private static void viewDoctors(String hospital) {
    JFrame doctorsFrame = new JFrame("Available Doctors at " + hospital);
    doctorsFrame.setSize(400, 300);
    doctorsFrame.setLayout(new BorderLayout());
    doctorsFrame.getContentPane().setBackground(new Color(255, 250, 240));
```

```
    JTextArea doctorsList = new JTextArea();
    doctorsList.setEditable(false);

    // List the doctors available at the hospital
    StringBuilder sb = new StringBuilder("Doctors at " + hospital + ":\n\n");
    for (String doctor : hospitalDoctors.get(hospital)) {
        sb.append("- ").append(doctor).append("\n");
    }

    doctorsList.setText(sb.toString());
    doctorsList.setFont(new Font("Courier New", Font.PLAIN, 14));

    // Add the doctor list to the doctors frame
    doctorsFrame.add(newJScrollPane(doctorsList),BorderLayout.CENTER);
    doctorsFrame.setVisible(true);
}

// Method to book an appointment at a hospital
private static void bookAppointment(String hospital) {
    JFrame bookingFrame = new JFrame("Book Appointment at " + hospital);
    bookingFrame.setSize(400, 400);
    bookingFrame.setLayout(new GridLayout(8, 1));
    bookingFrame.getContentPane().setBackground(new Color(240, 255, 240));

    JComboBox<String> doctorDropdown = new JComboBox<>(hospitalDoctors.get(hospital));
    JComboBox<String> timeSlotDropdown = new JComboBox<>(timeSlots);
    JSpinner dateSpinner = new JSpinner(new SpinnerDateModel());
    JSpinner.DateEditor dateEditor = new JSpinner.DateEditor(dateSpinner, "yyyy-MM-dd");
    dateSpinner.setEditor(dateEditor);
    JTextField nameField = new JTextField();
    JButton bookButton = new JButton("Book");
    JLabel statusLabel = new JLabel("");

    // Style the booking button
    styleButton(bookButton);
```

```
// Add components to the booking frame
bookingFrame.add(new JLabel("Select Doctor:"));
bookingFrame.add(doctorDropdown);
bookingFrame.add(new JLabel("Select Date:"));
bookingFrame.add(dateSpinner);
bookingFrame.add(new JLabel("Select Time Slot:"));
bookingFrame.add(timeSlotDropdown);
bookingFrame.add(new JLabel("Enter Your Name:"));
bookingFrame.add(nameField);
bookingFrame.add(bookButton);
bookingFrame.add(statusLabel);

// Action for booking the appointment
bookButton.addActionListener(e -> {
    String doctor = (String) doctorDropdown.getSelectedItem();
    String timeSlot = (String) timeSlotDropdown.getSelectedItem();
    String name = nameField.getText().trim();

    if (name.isEmpty()) {
        statusLabel.setText("Invalid name! Please enter your name.");
        return;
    }

    Date selectedDate = (Date) dateSpinner.getValue();
    Calendar calendar = Calendar.getInstance();
    calendar.setTime(selectedDate);
    int day = calendar.get(Calendar.DAY_OF_MONTH);
    int month = calendar.get(Calendar.MONTH); // 0-indexed (Feb = 1)
    int year = calendar.get(Calendar.YEAR);

    if (day == 30 && month == 1) {
        statusLabel.setText("30 February is not a valid date!");
        return;
    }
```

```
            if (!selectedDate.after(new Date())) {
                statusLabel.setText("Please select a future date!");
                return;
            }

            SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd");
            String dateString = dateFormat.format(selectedDate);

            // Check for the availability of the time slot
            Map<String, Map<String, Map<String, String>>> hospitalAppointments =
            appointments.get(hospital);
            Map<String, Map<String, String>> doctorAppointments =
            hospitalAppointments.getOrDefault(doctor, new HashMap<>());
            Map<String, String> dateAppointments = doctorAppointments.getOrDefault(dateString, new
            HashMap<>());

            if (dateAppointments.containsKey(timeSlot)) {
                statusLabel.setText("Time slot already booked on this date!");
            } else {
                // Book the appointment
                dateAppointments.put(timeSlot, name);
                doctorAppointments.put(dateString, dateAppointments);
                hospitalAppointments.put(doctor, doctorAppointments);
                appointments.put(hospital, hospitalAppointments);
                statusLabel.setText("Appointment booked successfully!");
                showAppointmentLetter(name, hospital, doctor, timeSlot, dateString);
    }
    });

            // Make the booking frame visible
            bookingFrame.setVisible(true);
        }

        // Method to display confirmation letter after booking an appointment
```

```
    private static void showAppointmentLetter(String name, String hospital, String doctor, String
    timeSlot, String date) {
        String letter = "Dear " + name + ",\n\n" +
                "Your appointment at " + hospital + " with Dr. " + doctor + " has been successfully
                booked.\n" +
                "Date: " + date + "\n" +
                "Time Slot: " + timeSlot + "\n\n" +
                "Thank you for using our service!\n\n" +
                "Regards,\nHospital Chatbot Team";

        // Show confirmation letter in a dialog box
        JOptionPane.showMessageDialog(null, letter, "Appointment Confirmation",
        JOptionPane.INFORMATION_MESSAGE);
    }

    // Method to cancel an appointment at a hospital
    private static void cancelAppointment(String hospital) {
        // Display cancel appointment options and functionality for hospitals other than Sunrise
        Hospital
        // Add your logic here for canceling the appointment
    }

    // Method to style the buttons
    private static void styleButton(JButton button) {
        button.setFont(new Font("Arial", Font.PLAIN, 14));
        button.setBackground(new Color(0, 102, 204));
        button.setForeground(Color.WHITE);
        button.setPreferredSize(new Dimension(200, 40));}
}
```