**HOSPITAL MANAGEMENT SYSTEM**

**AIM:**

To develop a Java application titled “Hospital Management System”.

**ALGORITHM:**

**Step 1:** Start the Program

1. Open the Java IDE (Eclipse).
2. Ensure that the JavaFX library is properly configured in the project.
3. Create a new Java class file for the program, e.g., Hospital Management System.java.

iv. Write the provided code into the class file.

**Step 2:** Configure the Project

1. Set up the project to use JavaFX by configuring the module-info.java file.
2. Add the VM options to the Run/Debug configuration to include JavaFX modules.

**Step 3:**

1. Define Entities:Identify main entities such as Patient, Doctor, Appointment, etc.
2. Database Design:Create a simple database schema to store relevant information.
3. Database Connectivity:Establish a connection to the database (SQLite, H2, etc.) using JDBC.
4. Entity Classes: Implement Java classes representing entities with attributes corresponding to the database schema.
5. User Interface:Design a basic user interface using Java Swing or JavaFX.Include screens for patient registration, appointment scheduling, and basic information display.
6. Implement CRUD Operations:Create methods for basic CRUD operations (Create, Read, Update, Delete) for entities.
7. User Input Validation:Implement simple validation for user inputs on the UI.
8. Search Functionality:Add basic search functionality to retrieve information efficiently.
9. Appointment Reminders: Integrate a basic appointment reminder feature for upcoming appointments.

**Step 4:** End the Program

1. Close the application window.
2. Stop the program execution in the IDE.

**APPLICATION ENVIRONMENT SUPPORT:**

**HARDWARE:** PC with sufficient RAM to run JavaFX applications.

**SOFTWARE:** JDK 9.0 or later, Eclipse/NetBeans IDE, JavaFX Libraries

**PROGRAM:**

package application;

import javafx.application.Application;

import javafx.collections.FXCollections;

import javafx.collections.ObservableList;

import javafx.geometry.Insets;

import javafx.scene.Scene;

import javafx.scene.control.\*;

import javafx.scene.layout.HBox;

import javafx.scene.layout.VBox;

import javafx.stage.Stage;

import java.time.LocalDateTime;

import java.time.format.DateTimeFormatter;

import java.util.Optional;

public class Main extends Application {

private ObservableList<Patient> patients;

private ObservableList<Doctor> doctors;

private ComboBox<Patient> patientComboBox;

private ComboBox<Doctor> doctorComboBox;

private TextArea medicalHistoryTextArea;

private TextField medicalHistoryEntryField;

private ListView<Appointment> appointmentListView;

public static void main(String[] args) {

launch(args);

}

@Override

public void start(Stage primaryStage) {

primaryStage.setTitle("Hospital Management System");

patients = FXCollections.observableArrayList();

doctors = FXCollections.observableArrayList();

Label nameLabel = new Label("Patient Name:");

TextField nameTextField = new TextField();

Button addPatientButton = new Button("Add Patient");

Label doctorLabel = new Label("Select Doctor:");

doctorComboBox = new ComboBox<>(doctors);

Button addDoctorButton = new Button("Add Doctor");

Label scheduleLabel = new Label("Schedule Appointment:");

patientComboBox = new ComboBox<>(patients);

Button scheduleAppointmentButton = new Button("Schedule Appointment");

Label medicalHistoryLabel = new Label("Medical History:");

medicalHistoryTextArea = new TextArea();

medicalHistoryTextArea.setEditable(false);

medicalHistoryEntryField = new TextField();

Button addMedicalHistoryButton = new Button("Add to Medical History");

Label appointmentLabel = new Label("Appointments:");

appointmentListView = new ListView<>();

addPatientButton.setOnAction(e -> addPatient(nameTextField.getText()));

addDoctorButton.setOnAction(e -> addDoctor());

scheduleAppointmentButton.setOnAction(e -> scheduleAppointment(patientComboBox.getValue(), doctorComboBox.getValue()));

addMedicalHistoryButton.setOnAction(e -> addToMedicalHistory(patientComboBox.getValue(), medicalHistoryEntryField.getText()));

patientComboBox.setOnAction(e -> updateMedicalHistoryTextArea());

doctorComboBox.setOnAction(e -> updateAppointments());

VBox vbox = new VBox(10);

vbox.setPadding(new Insets(10, 10, 10, 10));

HBox patientBox = new HBox(10, nameLabel, nameTextField, addPatientButton);

HBox doctorBox = new HBox(10, doctorLabel, doctorComboBox, addDoctorButton);

HBox scheduleBox = new HBox(10, scheduleLabel, patientComboBox, scheduleAppointmentButton);

HBox medicalHistoryBox = new HBox(10, medicalHistoryLabel, medicalHistoryTextArea, medicalHistoryEntryField, addMedicalHistoryButton);

HBox appointmentBox = new HBox(10, appointmentLabel, appointmentListView);

vbox.getChildren().addAll(patientBox, doctorBox, scheduleBox, medicalHistoryBox, appointmentBox);

Scene scene = new Scene(vbox, 600, 500);

primaryStage.setScene(scene);

primaryStage.show();

}

private void addPatient(String patientName) {

if (!patientName.trim().isEmpty()) {

Patient patient = new Patient(patientName);

patients.add(patient);

updatePatientComboBox();

}

}

private void addDoctor() {

TextInputDialog dialog = new TextInputDialog();

dialog.setTitle("Add Doctor");

dialog.setHeaderText(null);

dialog.setContentText("Enter doctor's name:");

Optional<String> result = dialog.showAndWait();

result.ifPresent(name -> {

TextInputDialog specialtyDialog = new TextInputDialog();

specialtyDialog.setTitle("Add Doctor");

specialtyDialog.setHeaderText(null);

specialtyDialog.setContentText("Enter doctor's specialty:");

Optional<String> specialtyResult = specialtyDialog.showAndWait();

specialtyResult.ifPresent(specialty -> {

Doctor doctor = new Doctor(name, specialty);

doctors.add(doctor);

updateDoctorComboBox();

});

});

}

private void scheduleAppointment(Patient patient, Doctor doctor) {

if (patient != null && doctor != null) {

LocalDateTime dateTime = LocalDateTime.now(); // Replace with your logic to set the appointment date and time

doctor.addAppointment(patient, dateTime);

updateAppointments();

}

}

private void addToMedicalHistory(Patient patient, String entry) {

if (patient != null && !entry.trim().isEmpty()) {

patient.addToMedicalHistory(entry);

updateMedicalHistoryTextArea();

}

}

private void updatePatientComboBox() {

patientComboBox.setItems(patients);

patientComboBox.getSelectionModel().clearSelection();

updateMedicalHistoryTextArea();

}

private void updateDoctorComboBox() {

doctorComboBox.setItems(doctors);

doctorComboBox.getSelectionModel().clearSelection();

updateAppointments();

}

private void updateMedicalHistoryTextArea() {

Patient selectedPatient = patientComboBox.getValue();

if (selectedPatient != null) {

medicalHistoryTextArea.setText(selectedPatient.getMedicalHistoryWithNames());

} else {

medicalHistoryTextArea.clear();

}

}

private void updateAppointments() {

Doctor selectedDoctor = doctorComboBox.getValue();

if (selectedDoctor != null) {

appointmentListView.setItems(selectedDoctor.getAppointments());

} else {

appointmentListView.getItems().clear();

}

}

private static class Patient {

private String name;

private String medicalHistory = "";

public Patient(String name) {

this.name = name;

}

public String getName() {

return name;

}

public String getMedicalHistory() {

return medicalHistory;

}

public void addToMedicalHistory(String entry) {

medicalHistory += name + ": " + entry + "\n";

}

public String getMedicalHistoryWithNames() {

return medicalHistory;

}

@Override

public String toString() {

return name;

}

}

private static class Doctor {

private String name;

private String specialty;

private ObservableList<Appointment> appointments;

public Doctor(String name, String specialty) {

this.name = name;

this.specialty = specialty;

this.appointments = FXCollections.observableArrayList();

}

public String getName() {

return name;

}

public String getSpecialty() {

return specialty;

}

public ObservableList<Appointment> getAppointments() {

return appointments;

}

public void addAppointment(Patient patient, LocalDateTime dateTime) {

appointments.add(new Appointment(patient, this, dateTime));

}

@Override

public String toString() {

return name + " (" + specialty + ")";

}

}

private static class Appointment {

private Patient patient;

private Doctor doctor;

private LocalDateTime dateTime;

public Appointment(Patient patient, Doctor doctor, LocalDateTime dateTime) {

this.patient = patient;

this.doctor = doctor;

this.dateTime = dateTime;

}

public Patient getPatient() {

return patient;

}

public Doctor getDoctor() {

return doctor;

}

public LocalDateTime getDateTime() {

return dateTime;

}

@Override

public String toString() {

return patient.getName() + " - " + doctor.getName() + " (" + doctor.getSpecialty() + ") - " +

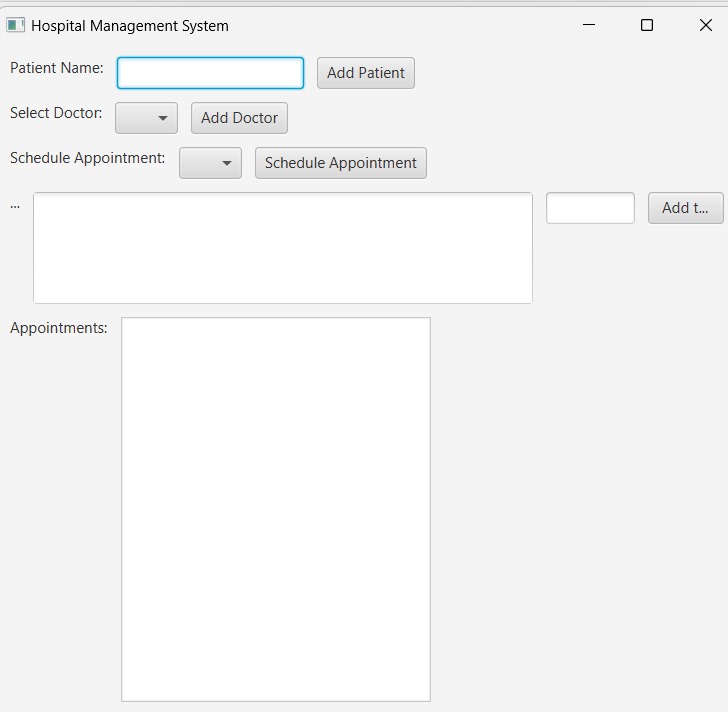
dateTime.format(DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm"));

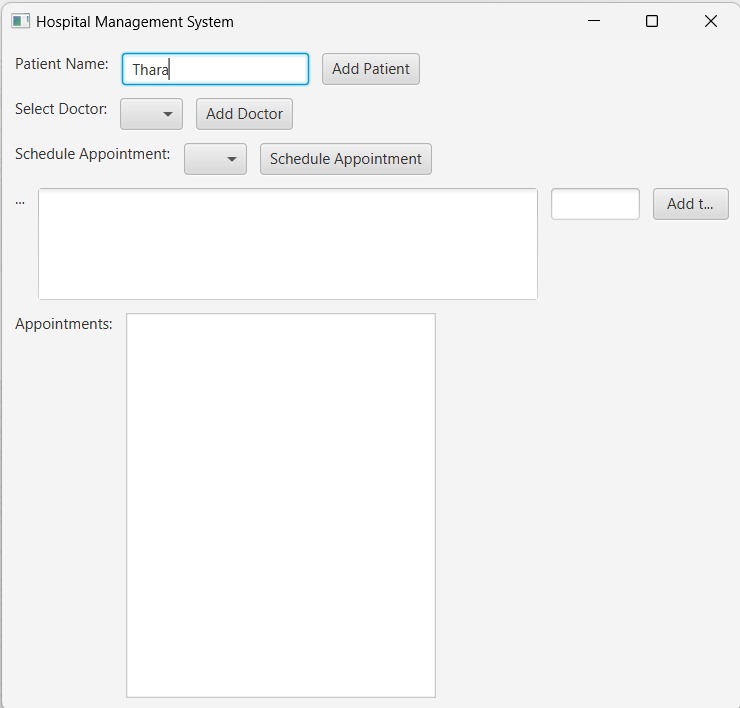
}

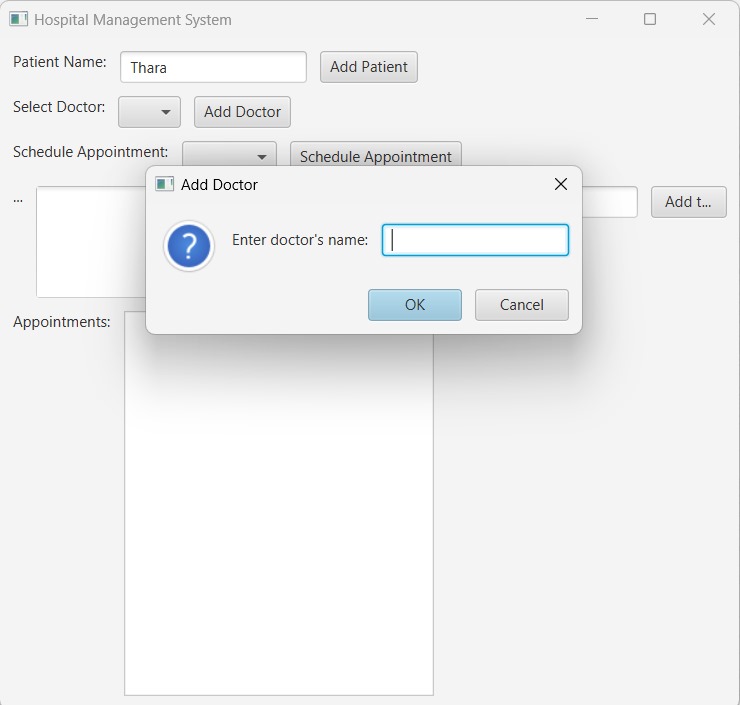
}

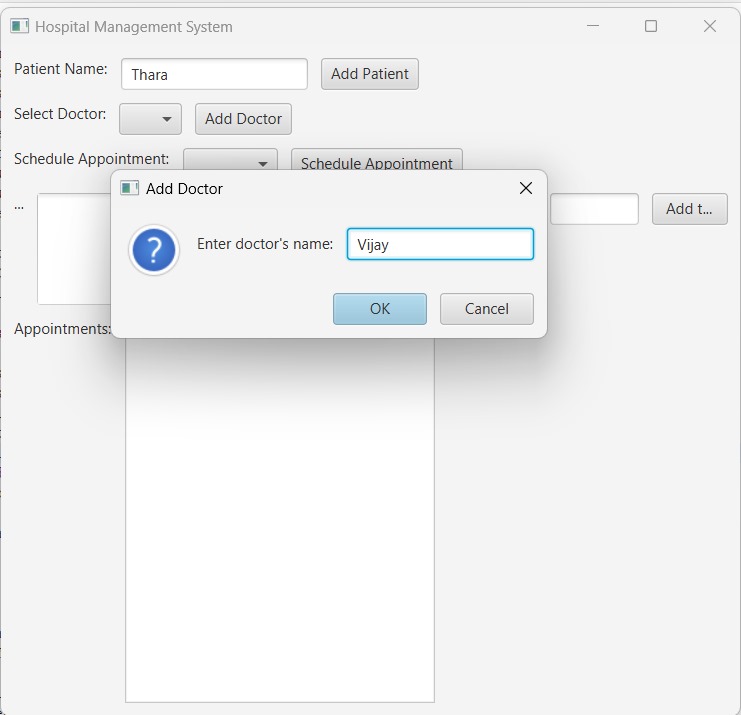
}

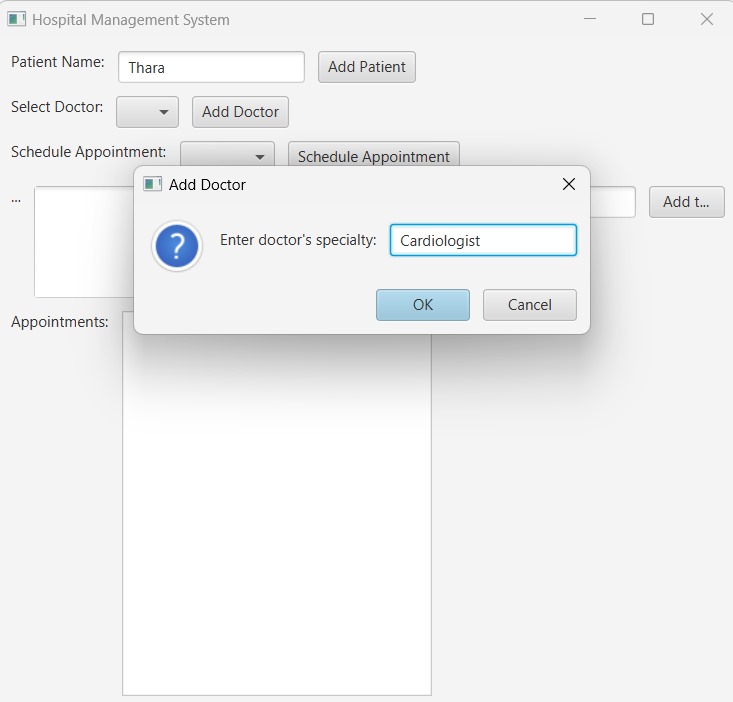
**OUTPUT:**

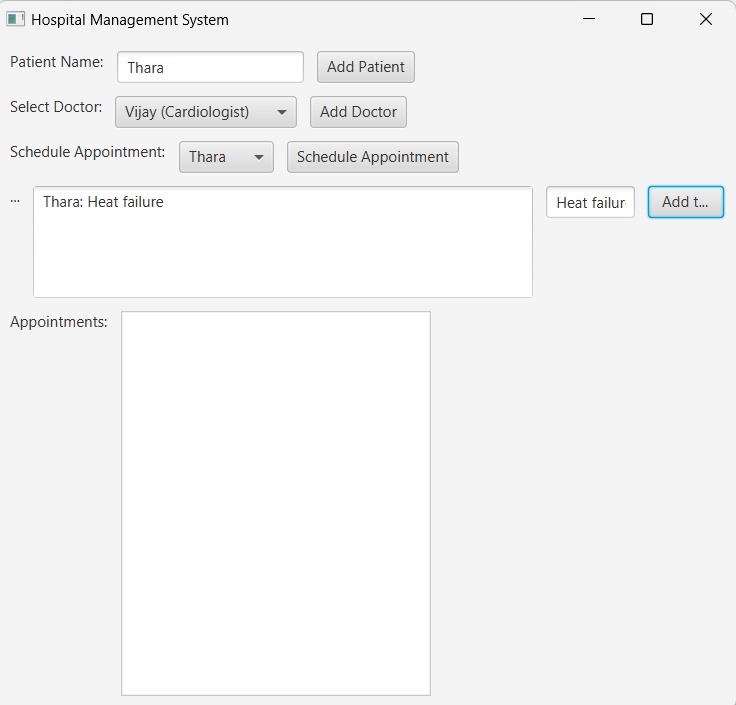


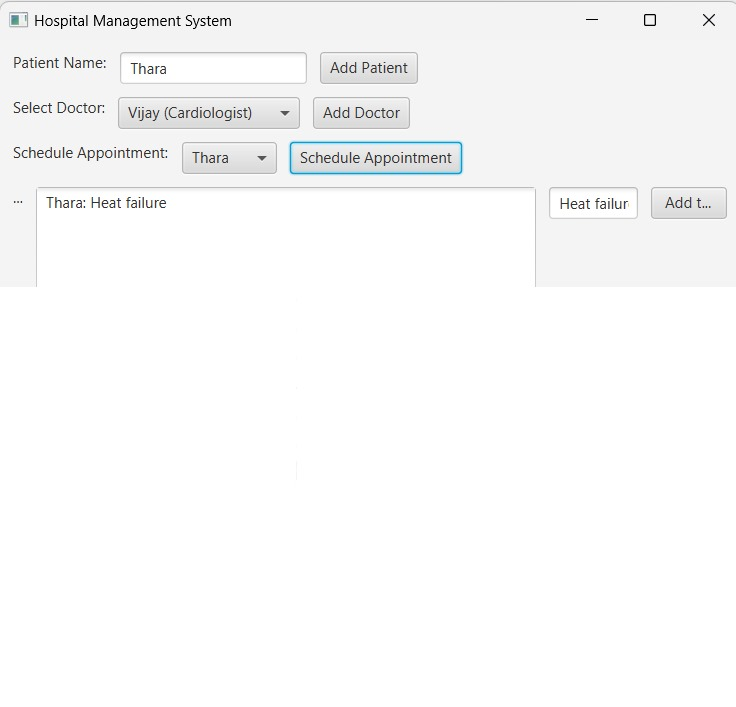


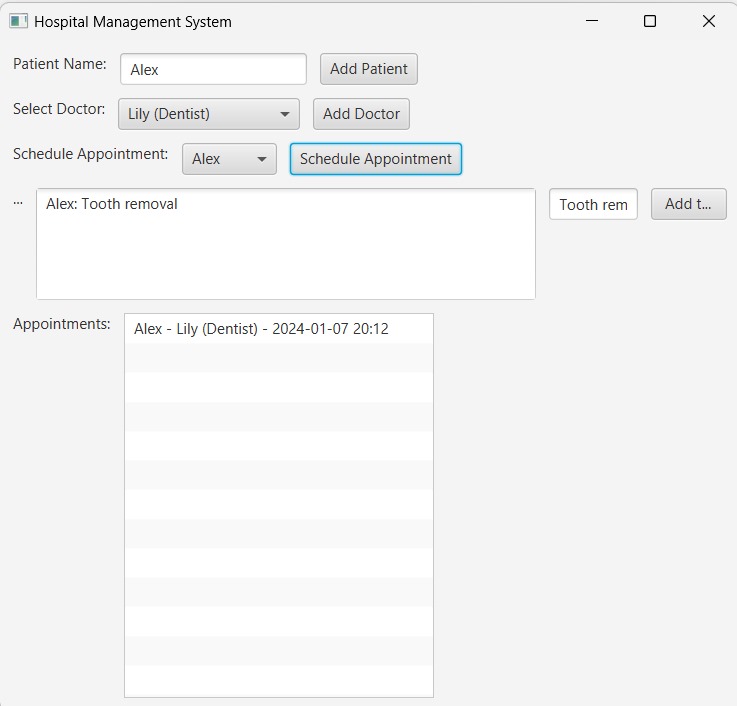


****

****

****

****

****

**RESULT:**

Thus, the Java application for developing hospital management system was successfully developed and implemented.