

# **HOSPITAL INVOICE MANAGEMENT SYSTEM**

**A MINI-PROJECT BY:**

**KUMARAN R      230701158**

**MANOHARAN K      230701177**

*in partial fulfillment of the award of the degree*

*OF*

***BACHELOR OF ENGINEERING***

**IN**

**COMPUTER SCIENCE AND ENGINEERING**



**RAJALAKSHMI ENGINEERING COLLEGE, CHENNAI**

**An Autonomous Institute**

**CHENNAI**

**NOVEMBER 2024**

## **BONAFIDE CERTIFICATE**

Certified that this project report “**Hospital Invoice Management System**” is a Bonafide work of “**KUMARAN R (230701158) & MANOHARAN K (230701177)**” .

Submitted for the Practical Examination held on \_\_\_\_\_

### **Signature**

Mrs. K. Maheshmeena

Assistant Professor,

Computer Science and Engineering,

Rajalakshmi Engineering College (Autonomous),

Thandalam, Chennai-602 105

**Internal Examiner**

**External Examiner**

## **ACKNOWLEDGEMENT**

I would like to extend my sincere gratitude to everyone who has contributed to the successful completion of this mini project.

First and foremost, I am deeply thankful to my Professor **Mrs. K. Maheshmeena** my project advisor, for their invaluable guidance, insightful feedback, and continuous support throughout the duration of this mini project. Their expertise and encouragement have been instrumental in shaping my research and bringing this mini project to completion.

I would also like to express my appreciation to the faculty and staff of the **Computer Science and Engineering Department** at Rajalakshmi Engineering College for providing the necessary resources and a conducive learning environment. We express our sincere thanks to **Dr. P. Kumar, M.E., Ph.D.**, Professor and Head of the Department Computer Science and Engineering for his guidance and encouragement throughout the project work.

My heartfelt thanks go to my peers and friends for their collaboration, constructive criticism, and moral support.

Thank you all for your contributions, both direct and indirect, to the success of this project.

## **Abstract of the Project :**

The **Hospital Invoice Management System** is a database-driven solution designed to streamline patient billing and invoice management processes in healthcare settings. The system employs a **relational database (MySQL)** to securely store and organize crucial data, including patient details, medical treatments, billing records, and payment history. By structuring this information within well-defined tables and relationships, the database ensures efficient data retrieval and updates while maintaining integrity and security.

The system's core functionalities are centered around database operations. **Patient information management** involves storing demographic and medical details in a dedicated table, allowing for quick access and updates. **Billing generation** is automated by leveraging treatment records stored in the database, ensuring accuracy in calculating charges. **Invoice creation** generates detailed, itemized summaries of patient charges, seamlessly linked to treatment and billing data. Additionally, **payment tracking** is facilitated through tables that record payment statuses, ensuring up-to-date financial records for each patient. By integrating the MySQL database with a **JavaFX frontend**, the system offers an intuitive interface for hospital staff to manage data efficiently. This integration enables seamless data entry, secure transactions, and accurate invoice generation, reducing administrative effort and enhancing billing transparency.

# **TABLE OF CONTENTS**

## **1. INTRODUCTION**

### **1.1 Introduction**

### **1.2 Objectives**

### **1.3 Modules**

## **2. SURVEY OF TECHNOLOGIES**

### **2.1 Software Description**

### **2.2 Languages**

### **2.3 SQL**

### **2.4 Java**

## **3. REQUIREMENTS AND ANALYSIS**

### **3.1 Requirement Specification**

### **3.2 Hardware and Software Requirements**

### **3.3 Architecture Diagram**

### **3.4 ER Diagram**

### **3.5 Normalization**

## **4. DESIGN AND IMPLEMENTATION**

### **4.1 Database Design**

## **5. SOURCE CODE**

## **6. SNAPSHOTS**

## **7. CONCLUSION**

## **8. REFERENCES**

# 1. INTRODUCTION

## 1.1 Introduction

The **Database Management System (DBMS)** project focuses on creating a structured and efficient way to manage data for various applications. It ensures data integrity, reduces redundancy, and provides an interface for performing CRUD (Create, Read, Update, Delete) operations. The project demonstrates the integration of a relational database with a programming language to develop a functional system that meets specific user requirements.

## 1.2 Objectives

- To design and implement a relational database for efficient data storage and retrieval.
- To ensure data integrity and consistency through normalization.
- To provide user-friendly interfaces for performing operations on the database.
- To integrate SQL with a programming language for seamless database interaction.

## 1.3 Modules

1. **User Authentication:** Secure login and access control.
2. **Data Management:** CRUD operations on the database.
3. **Reports:** Generate reports from the stored data.
4. **Validation:** Ensure data accuracy and security.
5. **Backup and Restore:** Facilitate database maintenance.

Supporting potential future enhancements, such as integrating insurance claims or online payment systems.

## 2. SURVEY OF TECHNOLOGIES

### 2.1 Software Description

The project utilizes **MySQL** for the database, renowned for its robustness, scalability, and wide usage in both small and enterprise-level applications. It provides reliable data storage and supports complex queries with ease.

**Java** is chosen as the programming language due to its flexibility, platform independence, and strong community support. Java facilitates both backend development and frontend design, making it an ideal choice for building a comprehensive system with a user-friendly interface and seamless integration with the MySQL database.

### 2.2 Languages

The system is built using the following languages:

- **SQL (Structured Query Language):** SQL is the core language used for managing and manipulating relational databases. It allows us to create, read, update, and delete records within the MySQL database. SQL is crucial for implementing the data operations and performing complex queries for reporting and data retrieval.
- **Java:** Java is the primary language used for backend logic and GUI development. It connects seamlessly with MySQL using JDBC (Java Database Connectivity) for querying and updating the database. The language's object-oriented nature also aids in organizing the system's functionality effectively. Java is also utilized for the creation of a rich, interactive user interface (UI), making the application intuitive and easy to use.

### 2.3 SQL

SQL is employed in this project for the efficient design, manipulation, and querying of the relational database. It provides a standardized way of interacting with the database to retrieve, modify, and manage the stored data. The SQL queries handle operations such as:

- **Data Retrieval:** SELECT queries are used to fetch specific patient, billing, and service-related information.

- **Data Insertion:** INSERT queries are used to add new records for patients, services, medications, and payments.
- **Data Modification:** UPDATE queries allow modification of patient details or billing information as required.
- **Data Deletion:** DELETE queries are used to remove records from the database when necessary.

The use of SQL ensures efficient management of the database, optimizing performance while maintaining data integrity and consistency.

## 2.4 Java

Java plays a significant role in the development of both the backend logic and the user interface of this project. As an object-oriented language, it allows for a modular and maintainable approach to coding, ensuring that each component of the system is self-contained and interacts with others in an organized manner.

- **Backend Logic:** Java handles the business logic, ensuring that data from the database is processed correctly and efficiently. It manages interactions with the MySQL database through JDBC to execute SQL queries, retrieve data, and update records.
- **Database Connectivity:** Using JDBC, Java establishes a connection to the MySQL database, allowing for seamless communication between the frontend and the database. This ensures that user actions in the GUI are reflected in the database in real-time.
- **User Interface (UI):** Java, using libraries like JavaFX or Swing, is used to build the graphical user interface. The UI is designed to be simple, interactive, and intuitive, providing an easy way for users to interact with the system, view patient details, generate bills, and manage payments.

Together, MySQL and Java offer a strong foundation for developing a reliable, scalable, and user-friendly hospital management system.



### **3. REQUIREMENTS AND ANALYSIS**

#### **3.1 Requirement Specification**

- **Functional Requirements:**
  - User authentication.
  - CRUD operations for data.
  - Report generation.
- **Non-Functional Requirements:**
  - Scalability to handle large datasets.
  - High performance and reliability.

#### **3.2 Hardware and Software Requirements**

- **Hardware Requirements:**
  - Processor: Intel i3 or above.
  - RAM: 4 GB or higher.
  - Storage: Minimum 500 MB.
- **Software Requirements:**
  - Operating System: Windows/Linux/macOS.
  - Java Development Kit (JDK).
  - MySQL Server.
  - IDE: Eclipse/IntelliJ IDEA/NetBeans.

### 3.3 Architecture Diagram :



### 3.4 ER Diagram :



### 3.5 Normalization :

Table 1: Patient Information (Before Normalization) :

PatientID	Name	Age	PhoneNumbers	Address	MedicalHistory
001	John Doe	35	12345, 67890	123 Main Street	Hypertension
002	Jane Smith	28	54321	456 Oak Avenue	Asthma

Table 1: Patient Information (After Normalization - 1NF)

PatientID	Name	Age	Phone	Address	MedicalHistory
001	John Doe	35	12345	123 Main Street	Hypertension
001	John Doe	35	67890	123 Main Street	Hypertension
002	Jane Smith	28	54321	456 Oak Avenue	Asthma

Table 2: Patient Information (After Normalization - 2NF)

- Separate the dependent attributes into different tables:

#### Patient Table

PatientID	Name	Age	Address
001	John Doe	35	123 Main Street
002	Jane Smith	28	456 Oak Avenue

### Phone Numbers Table :

PatientID	Phone
001	12345
001	67890
002	54321

### Medical History Table

PatientID	MedicalHistory
001	Hypertension
002	Asthma

### Table 3: Patient Information (After Normalization - 3NF)

- Ensure no transitive dependencies exist.

### Patient Table

PatientID	Name	Age
001	John Doe	35
002	Jane Smith	28

## Address Table

PatientID	Address
001	123 Main Street
002	456 Oak Avenue

## Phone Numbers Table

PatientID	Phone
001	12345
001	67890
002	54321

## Medical History Table

PatientID	MedicalHistory
001	Hypertension
002	Asthma

## 4. Design and Implementation

### 4.1 Database Design

Entity-Relationship (ER) Model:

- Entities: Patients, Treatments, Invoices, InvoiceDetails, Payments, Services.
- Relationships:
  - One-to-Many between Patients and Treatments.
  - One-to-Many between Invoices and InvoiceDetails.
  - One-to-Many between Invoices and Payments.

Schema (Tables):

#### Patients Table

Column	Type	Constraints
PatientID	INT	Primary Key
Name	VARCHAR(100)	NOT NULL
DOB	DATE	NOT NULL
Address	VARCHAR(200)	NOT NULL
ContactNumber	VARCHAR(15)	NOT NULL

#### Treatments Table

Column	Type	Constraints
TreatmentID	INT	Primary Key
PatientID	INT	Foreign Key
TreatmentDate	DATE	NOT NULL
Description	VARCHAR(200)	NOT NULL
Cost	DECIMAL(10,2)	NOT NULL

## Invoices Table

Column	Type	Constraints
InvoiceID	INT	Primary Key
PatientID	INT	Foreign Key
IssueDate	DATE	NOT NULL
TotalAmount	DECIMAL(10,2)	NOT NULL

## InvoiceDetails Table

Column	Type	Constraints
InvoiceID	INT	Foreign Key
ServiceID	INT	Foreign Key
Description	VARCHAR(200)	NOT NULL
Quantity	INT	NOT NULL
Cost	DECIMAL(10,2)	NOT NULL

## Services Table

Column	Type	Constraints
ServiceID	INT	Primary Key
Description	VARCHAR(200)	NOT NULL
Cost	DECIMAL(10,2)	NOT NULL

## Payments Table

Column	Type	Constraints
PaymentID	INT	Primary Key
InvoiceID	INT	Foreign Key
PaymentDate	DATE	NOT NULL
AmountPaid	DECIMAL(10,2)	NOT NULL
PaymentMethod	VARCHAR(50)	NOT NULL

## 5.SOURCE CODE:

### MAIN APPLICATION :

```
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import javafx.application.Application;
import javafx.geometry.Insets;
import javafx.geometry.Pos;
import javafx.scene.Scene;
import javafx.scene.control.*;
import javafx.scene.layout.*;
import javafx.scene.paint.Color;
import javafx.scene.paint.CycleMethod;
import javafx.scene.paint.LinearGradient;
import javafx.scene.paint.Stop;
import javafx.stage.Stage;

public class MainApp extends Application {

    public Stage primaryStage;

    @Override
    public void start(Stage primaryStage) {
        this.primaryStage = primaryStage;
        showLoginPage(primaryStage);
    }

    private void showLoginPage(Stage primaryStage) {
        primaryStage.setTitle("Login Page");

        Label userLabel = new Label("Username:");
        TextField userField = new TextField();
        userField.setPromptText("Enter your username");

        Label passLabel = new Label("Password:");
        PasswordField passField = new PasswordField();
        passField.setPromptText("Enter your password");

        Button loginButton = new Button("Login");
        Label messageLabel = new Label("");

        userLabel.setStyle("-fx-font-family: 'Roboto'; -fx-font-weight: bold; -fx-text-fill: black;");
        passLabel.setStyle("-fx-font-family: 'Roboto'; -fx-font-weight: bold; -fx-text-fill: black;");
```



```

loginButton.setStyle("-fx-font-family: 'Roboto'; -fx-font-weight: bold; -fx-text-fill:
white; -fx-background-color: black;");
userField.setStyle("-fx-font-family: 'Roboto'; -fx-font-weight: bold; -fx-text-fill:
black;");
passField.setStyle("-fx-font-family: 'Roboto'; -fx-font-weight: bold; -fx-text-fill:
black;");

```

```

loginButton.setOnAction(e -> {
    String username = userField.getText();
    String password = passField.getText();
    if (authenticateUser(username, password)) {
        messageLabel.setText("Login Successful!");
        messageLabel.setTextFill(Color.GREEN);
        showMainWindow(primaryStage);
    } else {
        messageLabel.setText("Invalid username or password.");
        messageLabel.setTextFill(Color.RED);
    }
});

```

```

VBox vbox = new VBox(10, userLabel, userField, passLabel, passField, loginButton,
messageLabel);
vbox.setAlignment(Pos.CENTER);
vbox.setPadding(new Insets(20));

```

```

Stop[] stops = new Stop[] {
    new Stop(0, Color.CORNFLOWERBLUE),
    new Stop(1, Color.DARKSLATEBLUE)
};
LinearGradient gradient = new LinearGradient(0, 0, 1, 1, true,
CycleMethod.NO_CYCLE, stops);

```

```

BorderPane root = new BorderPane();
root.setCenter(vbox);
root.setBackground(new javafx.scene.layout.Background(new
javafx.scene.layout.BackgroundFill(gradient, javafx.scene.layout.CornerRadii.EMPTY,
Insets.EMPTY)));

```

```

root.setStyle("-fx-background-image: url('file:/D:/image.jpg');"
+ "-fx-background-size: cover;");

```

```

Scene scene = new Scene(root, 400, 300);
primaryStage.setScene(scene);
primaryStage.show();
}

```

```

private boolean authenticateUser(String username, String password) {
    String query = "SELECT * FROM users WHERE username = ? AND password = ?";
    try (Connection connection = DatabaseConnection.connect();
        PreparedStatement stmt = connection.prepareStatement(query)) {

```

```

        stmt.setString(1, username);
        stmt.setString(2, password);
        ResultSet rs = stmt.executeQuery();
        return rs.next();
    } catch (SQLException e) {
        System.out.println("Error authenticating user: " + e.getMessage());
        return false;
    }
}

private void showMainWindow(Stage primaryStage) {
    Button backButton = new Button("Back");
    backButton.setStyle("-fx-font-family: 'Arial'; -fx-font-size: 14px; -fx-font-weight: bold;
-fx-text-fill: white; -fx-background-color: #333;");
    backButton.setOnAction(e -> showLoginPage(primaryStage));

    Button patientRegistrationButton = new Button("Patient Registration");
    Button billingButton = new Button("Billing");
    Button invoiceButton = new Button("Invoices");
    Button medicationButton = new Button("Medications");
    Button servicesButton = new Button("Services");
    Button paymentsButton = new Button("Payments");

    patientRegistrationButton.setStyle("-fx-font-family: 'Arial'; -fx-font-size: 14px;");
    billingButton.setStyle("-fx-font-family: 'Arial'; -fx-font-size: 14px;");
    invoiceButton.setStyle("-fx-font-family: 'Arial'; -fx-font-size: 14px;");
    medicationButton.setStyle("-fx-font-family: 'Arial'; -fx-font-size: 14px;");
    servicesButton.setStyle("-fx-font-family: 'Arial'; -fx-font-size: 14px;");
    paymentsButton.setStyle("-fx-font-family: 'Arial'; -fx-font-size: 14px;");

    patientRegistrationButton.setOnAction(e -> openPatientRegistration());
    billingButton.setOnAction(e -> openBilling());
    invoiceButton.setOnAction(e -> openInvoices());
    medicationButton.setOnAction(e -> openMedications());
    servicesButton.setOnAction(e -> openServices());
    paymentsButton.setOnAction(e -> openPayments());

    VBox layout = new VBox(20);
    layout.getChildren().addAll(
        backButton,
        patientRegistrationButton,
        billingButton,
        invoiceButton,
        medicationButton,
        servicesButton,
        paymentsButton
    );
    layout.setAlignment(Pos.CENTER);

    StackPane root = new StackPane();

```

```

        BackgroundImage backgroundImage = new BackgroundImage(
            new javafx.scene.image.Image("file:/D:/image.jpg"),
            BackgroundRepeat.NO_REPEAT,
            BackgroundRepeat.NO_REPEAT,
            BackgroundPosition.CENTER,
            BackgroundSize.DEFAULT
        );
        root.setBackground(new Background(backgroundImage));

        root.getChildren().add(layout);

        Scene scene = new Scene(root, 600, 400);
        primaryStage.setTitle("Hospital Management System");
        primaryStage.setScene(scene);
        primaryStage.show();
    }

    private void openPatientRegistration() {
        PatientRegistration registrationWindow = new PatientRegistration();
        Stage registrationStage = new Stage();
        registrationWindow.start(registrationStage);
    }

    private void openBilling() {
        Billing billingWindow = new Billing();
        billingWindow.show();
    }

    private void openInvoices() {
        Invoices invoicesWindow = new Invoices();
        invoicesWindow.show();
    }

    private void openMedications() {
        Medications medicationsWindow = new Medications();
        medicationsWindow.show();
    }

    private void openServices() {
        Services servicesWindow = new Services();
        servicesWindow.show();
    }

    private void openPayments() {
        Payments paymentsWindow = new Payments();
        paymentsWindow.show();
    }

    public static void main(String[] args) {
        launch(args);
    }

```

```
}  
}
```

## LOGIN WINDOW DESIGN :

```
import javafx.application.Application;  
import javafx.geometry.Insets;  
import javafx.geometry.Pos;  
import javafx.scene.Scene;  
import javafx.scene.control.Button;  
import javafx.scene.control.Label;  
import javafx.scene.control.PasswordField;  
import javafx.scene.control.TextField;  
import javafx.scene.layout.BorderPane;  
import javafx.scene.layout.VBox;  
import javafx.scene.paint.Color;  
import javafx.scene.paint.LinearGradient;  
import javafx.scene.paint.CycleMethod;  
import javafx.scene.paint.Stop;  
import javafx.stage.Stage;  
  
import java.sql.Connection;  
import java.sql.PreparedStatement;  
import java.sql.ResultSet;  
import java.sql.SQLException;  
  
public class LoginWindow extends Application {  
  
    @Override  
    public void start(Stage primaryStage) {  
        primaryStage.setTitle("Login Page");  
  
        // Create login form components  
        Label userLabel = new Label("Username:");  
        TextField userField = new TextField();  
        userField.setPromptText("Enter your username");  
  
        Label passLabel = new Label("Password:");  
        PasswordField passField = new PasswordField();  
        passField.setPromptText("Enter your password");  
  
        Button loginButton = new Button("Login");  
        Label messageLabel = new Label(""); // To display login status  
  
        // Styling components  
        userLabel.setStyle("-fx-font-family: 'Roboto'; -fx-font-weight: bold; -fx-text-fill:  
black;");  
        passLabel.setStyle("-fx-font-family: 'Roboto'; -fx-font-weight: bold; -fx-text-fill:  
black;");
```

```

loginButton.setStyle("-fx-font-family: 'Roboto'; -fx-font-weight: bold; -fx-text-fill:
white; -fx-background-color: black;");
userField.setStyle("-fx-font-family: 'Roboto'; -fx-font-weight: bold; -fx-text-fill:
black;");
passField.setStyle("-fx-font-family: 'Roboto'; -fx-font-weight: bold; -fx-text-fill:
black;");

// Action for login button
loginButton.setOnAction(e -> {
    String username = userField.getText();
    String password = passField.getText();
    if (authenticateUser(username, password)) {
        messageLabel.setText("Login Successful!");
        messageLabel.setTextFill(Color.GREEN);
        openMainWindow(primaryStage); // Open the main window after successful login
    } else {
        messageLabel.setText("Invalid username or password.");
        messageLabel.setTextFill(Color.RED);
    }
});

// Arrange nodes in a VBox layout
VBox vbox = new VBox(10, userLabel, userField, passLabel, passField, loginButton,
messageLabel);
vbox.setAlignment(Pos.CENTER);
vbox.setPadding(new Insets(20));

// Create a gradient background
Stop[] stops = new Stop[] {
    new Stop(0, Color.CORNFLOWERBLUE),
    new Stop(1, Color.DARKSLATEBLUE)
};
LinearGradient gradient = new LinearGradient(0, 0, 1, 1, true,
CycleMethod.NO_CYCLE, stops);

// Root pane setup
BorderPane root = new BorderPane();
root.setCenter(vbox);
root.setBackground(new javafx.scene.layout.Background(new
javafx.scene.layout.BackgroundFill(gradient, javafx.scene.layout.CornerRadii.EMPTY,
Insets.EMPTY)));

// Add background image using CSS style in root node
root.setStyle("-fx-background-image: url('file:/D:/image.jpg');"
+ "-fx-background-size: cover;");

Scene scene = new Scene(root, 400, 300);
primaryStage.setScene(scene);
primaryStage.show();
}

```

```

// Authenticate user using the database
private boolean authenticateUser(String username, String password) {
    String query = "SELECT * FROM users WHERE username = ? AND password = ?";
    try (Connection connection = DatabaseConnection.connect();
        PreparedStatement stmt = connection.prepareStatement(query)) {
        stmt.setString(1, username);
        stmt.setString(2, password);
        ResultSet rs = stmt.executeQuery();
        return rs.next();
    } catch (SQLException e) {
        System.out.println("Error authenticating user: " + e.getMessage());
        return false;
    }
}

// Open the main window after successful login
private void openMainWindow(Stage primaryStage) {
    // Create buttons for different sections (Patient Registration, Billing, etc.)
    Button patientRegistrationButton = new Button("Patient Registration");
    Button billingButton = new Button("Billing");
    Button invoiceButton = new Button("Invoices");
    Button medicationButton = new Button("Medications");
    Button servicesButton = new Button("Services");
    Button paymentsButton = new Button("Payments");

    // Event handlers for buttons
    patientRegistrationButton.setOnAction(e -> openPatientRegistration());
    billingButton.setOnAction(e -> openBilling());
    invoiceButton.setOnAction(e -> openInvoices());
    medicationButton.setOnAction(e -> openMedications());
    servicesButton.setOnAction(e -> openServices());
    paymentsButton.setOnAction(e -> openPayments());

    // Layout for the main window
    VBox layout = new VBox(10);
    layout.getChildren().addAll(
        patientRegistrationButton,
        billingButton,
        invoiceButton,
        medicationButton,
        servicesButton,
        paymentsButton
    );

    // Set the scene for the main window
    Scene scene = new Scene(layout, 300, 250);
    primaryStage.setScene(scene);
    primaryStage.show();
}

```

```

// Methods to handle button clicks for each section
private void openPatientRegistration() {
    PatientRegistration registration = new PatientRegistration();
    registration.show();
}

private void openBilling() {
    Billing billing = new Billing();
    billing.show();
}

private void openInvoices() {
    Invoices invoices = new Invoices();
    invoices.show();
}

private void openMedications() {
    Medications medications = new Medications();
    medications.show();
}

private void openServices() {
    Services services = new Services();
    services.show();
}

private void openPayments() {
    Payments payments = new Payments();
    payments.show();
}

public static void main(String[] args) {
    launch(args);
}
}

```

## HOME PAGE DESIGN :

```

import javafx.application.Application;
import javafx.geometry.Pos;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.image.Image;
import javafx.scene.layout.Background;
import javafx.scene.layout.BackgroundImage;
import javafx.scene.layout.BackgroundPosition;
import javafx.scene.layout.BackgroundRepeat;
import javafx.scene.layout.BackgroundImageSize;
import javafx.scene.layout.StackPane;

```

```

import javafx.scene.layout.VBox;
import javafx.scene.text.Font;
import javafx.stage.Stage;

public class Home extends Application {

    @Override
    public void start(Stage primaryStage) {
        // Call method to open the main window after successful login
        openMainWindow(primaryStage);
    }

    // Method to open the main window with buttons for different sections
    private void openMainWindow(Stage primaryStage) {
        // Create buttons for different sections (Patient Registration, Billing, etc.)
        Button patientRegistrationButton = new Button("Patient Registration");
        Button billingButton = new Button("Billing");
        Button invoiceButton = new Button("Invoices");
        Button medicationButton = new Button("Medications");
        Button servicesButton = new Button("Services");
        Button paymentsButton = new Button("Payments");

        // Set button styles (fonts, size, etc.)
        patientRegistrationButton.setFont(new Font("Arial", 14));
        billingButton.setFont(new Font("Arial", 14));
        invoiceButton.setFont(new Font("Arial", 14));
        medicationButton.setFont(new Font("Arial", 14));
        servicesButton.setFont(new Font("Arial", 14));
        paymentsButton.setFont(new Font("Arial", 14));

        // Event handlers for buttons to open the corresponding sections
        patientRegistrationButton.setOnAction(e -> openPatientRegistration());
        billingButton.setOnAction(e -> openBilling());
        invoiceButton.setOnAction(e -> openInvoices());
        medicationButton.setOnAction(e -> openMedications());
        servicesButton.setOnAction(e -> openServices());
        paymentsButton.setOnAction(e -> openPayments());

        // Layout for the main window with vertical arrangement of buttons
        VBox layout = new VBox(20); // Adjust spacing between buttons
        layout.getChildren().addAll(
            patientRegistrationButton,
            billingButton,
            invoiceButton,
            medicationButton,
            servicesButton,
            paymentsButton
        );
        layout.setAlignment(Pos.CENTER); // Center-align buttons
    }
}

```



```

// Set background image
StackPane root = new StackPane();
BackgroundImage backgroundImage = new BackgroundImage(
    new Image("file:/D:/image.jpg"), // Replace with your image path
    BackgroundRepeat.NO_REPEAT,
    BackgroundRepeat.NO_REPEAT,
    BackgroundPosition.CENTER,
    BackgroundSize.DEFAULT
);
root.setBackground(new Background(backgroundImage));
// Add layout to the root pane
root.getChildren().add(layout);

// Set the scene for the main window
Scene scene = new Scene(root, 600, 400); // Increase window size for better visibility
primaryStage.setTitle("Hospital Management System");
primaryStage.setScene(scene);
primaryStage.show();
}
// Methods to handle button clicks for each section
private void openPatientRegistration() {
    // Logic to open the Patient Registration page
    System.out.println("Patient Registration Page Opened");
}

private void openBilling() {
    // Logic to open the Billing page
    System.out.println("Billing Page Opened");
}

private void openInvoices() {
    // Logic to open the Invoices page
    System.out.println("Invoices Page Opened");
}

private void openMedications() {
    // Logic to open the Medications page
    System.out.println("Medications Page Opened");
}

private void openServices() {
    // Logic to open the Services page
    System.out.println("Services Page Opened");
}

private void openPayments() {
    // Logic to open the Payments page
    System.out.println("Payments Page Opened");
}

```

```

    public static void main(String[] args) {
        launch(args);
    }
}

```

## PATIENT REGISTRATION PAGE DESIGN

```

import javafx.application.Application;
import javafx.geometry.Insets;
import javafx.geometry.Pos;
import javafx.scene.Scene;
import javafx.scene.control.*;
import javafx.scene.layout.GridPane;
import javafx.stage.Stage;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.SQLException;

public class PatientRegistration extends Application {

    @Override
    public void start(Stage primaryStage) {
        primaryStage.setTitle("Patient Registration");

        // Create the root layout (GridPane)
        GridPane grid = new GridPane();
        grid.setAlignment(Pos.CENTER);
        grid.setHgap(10);
        grid.setVgap(10);
        grid.setPadding(new Insets(25, 25, 25, 25));

        // Set the background image
        grid.setStyle("-fx-background-image: url(file:/D:/image.jpg);"
            + "-fx-background-size: cover; -fx-background-position: center;");

        // Create and add form fields for patient registration
        Label nameLabel = new Label("Patient Name:");
        nameLabel.setStyle("-fx-font-family: 'Roboto'; -fx-font-weight: bold; -fx-text-fill:
black;");
        grid.add(nameLabel, 0, 0);

        TextField nameField = new TextField();
        nameField.setStyle("-fx-font-family: 'Roboto'; -fx-font-weight: bold; -fx-text-fill:
black;");
        grid.add(nameField, 1, 0);

        Label ageLabel = new Label("Age:");
        ageLabel.setStyle("-fx-font-family: 'Roboto'; -fx-font-weight: bold; -fx-text-fill:
black;");
    }
}

```

```

grid.add(ageLabel, 0, 1);

TextField ageField = new TextField();
ageField.setStyle("-fx-font-family: 'Roboto'; -fx-font-weight: bold; -fx-text-fill: black;");
grid.add(ageField, 1, 1);

Label genderLabel = new Label("Gender:");
genderLabel.setStyle("-fx-font-family: 'Roboto'; -fx-font-weight: bold; -fx-text-fill:
black;");
grid.add(genderLabel, 0, 2);

ComboBox<String> genderComboBox = new ComboBox<>();
genderComboBox.getItems().addAll("Male", "Female", "Other");
genderComboBox.setValue("Male"); // Default value
genderComboBox.setStyle("-fx-font-family: 'Roboto'; -fx-font-weight: bold; -fx-text-
fill: black;");
grid.add(genderComboBox, 1, 2);

Label contactLabel = new Label("Contact Number:");
contactLabel.setStyle("-fx-font-family: 'Roboto'; -fx-font-weight: bold; -fx-text-fill:
black;");
grid.add(contactLabel, 0, 3);

TextField contactField = new TextField();
contactField.setStyle("-fx-font-family: 'Roboto'; -fx-font-weight: bold; -fx-text-fill:
black;");
grid.add(contactField, 1, 3);

// Create an address field
Label addressLabel = new Label("Address:");
addressLabel.setStyle("-fx-font-family: 'Roboto'; -fx-font-weight: bold; -fx-text-fill:
black;");
grid.add(addressLabel, 0, 4);

TextField addressField = new TextField();
addressField.setStyle("-fx-font-family: 'Roboto'; -fx-font-weight: bold; -fx-text-fill:
black;");
grid.add(addressField, 1, 4);

// Create a medical history field
Label medicalHistoryLabel = new Label("Medical History:");
medicalHistoryLabel.setStyle("-fx-font-family: 'Roboto'; -fx-font-weight: bold; -fx-text-
fill: black;");
grid.add(medicalHistoryLabel, 0, 5);

TextField medicalHistoryField = new TextField();
medicalHistoryField.setStyle("-fx-font-family: 'Roboto'; -fx-font-weight: bold; -fx-text-
fill: black;");
grid.add(medicalHistoryField, 1, 5);

```

```

// Register button setup
Button registerButton = new Button("Register");
registerButton.setStyle("-fx-font-family: 'Roboto'; -fx-font-weight: bold; -fx-text-fill:
white; -fx-background-color: black;");
grid.add(registerButton, 1, 6);

// Set action for Register button
registerButton.setOnAction(e -> {
    String name = nameField.getText();
    int age = Integer.parseInt(ageField.getText());
    String gender = genderComboBox.getValue();
    String contactInfo = contactField.getText();
    String address = addressField.getText();
    String medicalHistory = medicalHistoryField.getText();

    // Insert the patient into the database
    String query = "INSERT INTO patients (name, age, gender, contact_info, address,
medical_history) VALUES (?, ?, ?, ?, ?, ?)";
    try (Connection connection = DatabaseConnection.connect();
        PreparedStatement stmt = connection.prepareStatement(query)) {

        stmt.setString(1, name);
        stmt.setInt(2, age);
        stmt.setString(3, gender);
        stmt.setString(4, contactInfo);
        stmt.setString(5, address);
        stmt.setString(6, medicalHistory);

        int rowsAffected = stmt.executeUpdate();
        System.out.println(rowsAffected + " patient(s) added.");

        Alert alert = new Alert(Alert.AlertType.INFORMATION);
        alert.setTitle("Registration Successful");
        alert.setHeaderText("Patient Registered Successfully");
        alert.setContentText("The patient has been added to the system.");
        alert.showAndWait();

        // Clear the fields after registration
        nameField.clear();
        ageField.clear();
        contactField.clear();
        addressField.clear();
        medicalHistoryField.clear();

    } catch (SQLException ex) {
        System.out.println("Error inserting patient: " + ex.getMessage());
        Alert alert = new Alert(Alert.AlertType.ERROR);
        alert.setTitle("Error");
        alert.setHeaderText("Patient Registration Failed");
        alert.setContentText("There was an error while registering the patient.");
    }
}

```

```

        alert.showAndWait();
    }
});

// Create the scene and display it
Scene scene = new Scene(grid, 400, 400);
primaryStage.setScene(scene);
primaryStage.show();
}

public static void main(String[] args) {
    launch(args);
}

public void show() {
    throw new UnsupportedOperationException("Unimplemented method 'show'");
}
}

```

## **MEDICATIONS PAGE DESIGN**

```

import javafx.scene.Scene;
import javafx.scene.control.TextField;
import javafx.scene.control.Button;
import javafx.scene.layout.VBox;
import javafx.stage.Stage;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import javafx.scene.control.Alert;
import javafx.scene.control.Alert.AlertType;

public class Medications extends Stage {

    public Medications() {
        setTitle("Medications");

        // Create TextFields with CSS styles
        TextField patientIDField = new TextField();
        patientIDField.setPromptText("Enter Patient ID");
        patientIDField.setStyle("-fx-font-family: 'Arial'; -fx-font-size: 14px; -fx-font-weight:
bold; -fx-text-fill: black;");

        TextField medicationField = new TextField();
        medicationField.setPromptText("Enter Medication");
        medicationField.setStyle("-fx-font-family: 'Arial'; -fx-font-size: 14px; -fx-font-weight:
bold; -fx-text-fill: black;");

        TextField dosageField = new TextField();

```

```

dosageField.setPromptText("Enter Dosage");
dosageField.setStyle("-fx-font-family: 'Arial'; -fx-font-size: 14px; -fx-font-weight: bold; -fx-text-fill: black;");

// Create Save Button with style
Button saveButton = new Button("Save Medication");
saveButton.setStyle("-fx-font-family: 'Arial'; -fx-font-size: 16px; -fx-font-weight: bold; -fx-text-fill: white; -fx-background-color: black;");

// Set the save action
saveButton.setOnAction(e -> {
    String patientID = patientIDField.getText();
    String medication = medicationField.getText();
    String dosage = dosageField.getText();

    // Input validation
    if (patientID.isEmpty() || medication.isEmpty() || dosage.isEmpty()) {
        showAlert(AlertType.ERROR, "Validation Error", "Please fill out all fields.");
        return;
    }

    // Save medication to the database
    saveMedicationToDatabase(patientID, medication, dosage);
    showAlert(AlertType.INFORMATION, "Success", "Medication saved successfully.");

    // Clear fields after saving
    patientIDField.clear();
    medicationField.clear();
    dosageField.clear();
});

// Layout for the medications form
VBox layout = new VBox(10);
layout.getChildren().addAll(patientIDField, medicationField, dosageField, saveButton);

// Set background image with CSS style
layout.setStyle("-fx-background-image: url('file:/D:/image.jpg');"
    + "-fx-background-size: cover;");

// Set the scene and window properties
Scene scene = new Scene(layout, 400, 300);
setScene(scene);
}

// Database connection details
private Connection connectToDatabase() throws SQLException {
    String url = "jdbc:mysql://localhost:3306/hospital_management";
    String user = "root"; // replace with your MySQL username
    String password = "kumaran"; // replace with your MySQL password
    return DriverManager.getConnection(url, user, password);
}

```

```

    }

    // Save medication information to the database
    private void saveMedicationToDatabase(String patientID, String medication, String
dosage) {
        String sql = "INSERT INTO medications (patient_id, medication, dosage) VALUES (?,
?, ?)";

        try (Connection conn = connectToDatabase();
            PreparedStatement pstmt = conn.prepareStatement(sql)) {

            pstmt.setString(1, patientID);
            pstmt.setString(2, medication);
            pstmt.setString(3, dosage);
            pstmt.executeUpdate();
        } catch (SQLException e) {
            showAlert(AlertType.ERROR, "Database Error", "Failed to save medication: " +
e.getMessage());
        }
    }

    // Show alert helper method
    private void showAlert(AlertType alertType, String title, String message) {
        Alert alert = new Alert(alertType);
        alert.setTitle(title);
        alert.setHeaderText(null);
        alert.setContentText(message);
        alert.showAndWait();
    }
}

```

## SERVICES PAGE DESIGN

```

import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.control.TextField;
import javafx.scene.control.Button;
import javafx.scene.layout.VBox;
import javafx.stage.Stage;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import javafx.scene.control.Alert;

public class Services extends Application {

    @Override
    public void start(Stage primaryStage) {
        primaryStage.setTitle("Services");
    }
}

```

```

// Create TextFields with styling
TextField serviceNameField = new TextField();
serviceNameField.setPromptText("Enter Service Name");
serviceNameField.setStyle("-fx-font-family: 'Arial'; -fx-font-size: 14px; -fx-font-weight:
bold; -fx-text-fill: black;");

TextField descriptionField = new TextField();
descriptionField.setPromptText("Enter Description");
descriptionField.setStyle("-fx-font-family: 'Arial'; -fx-font-size: 14px; -fx-font-weight:
bold; -fx-text-fill: black;");

// Create Save Button with styling
Button saveButton = new Button("Save Service");
saveButton.setStyle("-fx-font-family: 'Arial'; -fx-font-size: 16px; -fx-font-weight: bold; -
fx-text-fill: white; -fx-background-color: black;");

// Layout for the Services form
VBox layout = new VBox(10);
layout.getChildren().addAll(serviceNameField, descriptionField, saveButton);

// Set background image and additional styles
layout.setStyle("-fx-background-image: url('file:/D:/image.jpg');"
+ "-fx-background-size: cover;");

// Set the scene and window properties
Scene scene = new Scene(layout, 400, 300);
primaryStage.setScene(scene);
primaryStage.show();

// Set action for Save Service button
saveButton.setOnAction(e -> {
    String serviceName = serviceNameField.getText();
    String description = descriptionField.getText();

    // Insert the service into the database
    String query = "INSERT INTO services (service_name, description) VALUES (?, ?)";
    try (Connection connection = DatabaseConnection.connect();
        PreparedStatement stmt = connection.prepareStatement(query)) {

        stmt.setString(1, serviceName);
        stmt.setString(2, description);

        int rowsAffected = stmt.executeUpdate();
        System.out.println(rowsAffected + " service(s) added.");

        // Display success message
        Alert alert = new Alert(Alert.AlertType.INFORMATION);
        alert.setTitle("Service Registration");
        alert.setHeaderText("Service Saved Successfully");
        alert.setContentText("The service has been added to the system.");
    }
});

```



```

        alert.showAndWait();

        // Clear fields after saving
        serviceNameField.clear();
        descriptionField.clear();

    } catch (SQLException ex) {
        System.out.println("Error inserting service: " + ex.getMessage());
        Alert alert = new Alert(Alert.AlertType.ERROR);
        alert.setTitle("Error");
        alert.setHeaderText("Service Registration Failed");
        alert.setContentText("There was an error while saving the service.");
        alert.showAndWait();
    }
});
}

public static void main(String[] args) {
    launch(args);
}

// Database connection class
public static class DatabaseConnection {
    public static Connection connect() throws SQLException {
        // Replace with your MySQL connection details
        String url = "jdbc:mysql://localhost:3306/your_database";
        String username = "root";
        String password = "kumaran";

        return java.sql.DriverManager.getConnection(url, username, password);
    }
}

public void show() {
    throw new UnsupportedOperationException("Unimplemented method 'show'");
}
}

```

## **BILLING PAGE DESIGN**

```

import javafx.scene.Scene;
import javafx.scene.control.TextField;
import javafx.scene.control.Button;
import javafx.scene.layout.VBox;
import javafx.stage.Stage;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import javafx.scene.control.Alert;

```

```

import javafx.scene.control.Alert.AlertType;

public class Billing {

    public Billing() {
        Stage billingStage = new Stage(); // Create a new Stage instance for the Billing window
        billingStage.setTitle("Billing");

        // Create TextFields
        TextField patientIDField = new TextField();
        patientIDField.setPromptText("Enter Patient ID");
        patientIDField.setStyle("-fx-font-family: 'Arial'; -fx-font-size: 14px; -fx-font-weight:
bold; -fx-text-fill: black;");

        TextField serviceField = new TextField();
        serviceField.setPromptText("Enter Service");
        serviceField.setStyle("-fx-font-family: 'Arial'; -fx-font-size: 14px; -fx-font-weight: bold;
-fx-text-fill: black;");

        TextField amountField = new TextField();
        amountField.setPromptText("Enter Amount");
        amountField.setStyle("-fx-font-family: 'Arial'; -fx-font-size: 14px; -fx-font-weight: bold;
-fx-text-fill: black;");

        // Create a Save Button
        Button saveButton = new Button("Save Billing");
        saveButton.setStyle("-fx-font-family: 'Arial'; -fx-font-size: 16px; -fx-font-weight: bold; -
fx-text-fill: white; -fx-background-color: black;");

        // Set the save action
        saveButton.setOnAction(e -> {
            String patientID = patientIDField.getText();
            String service = serviceField.getText();
            String amountText = amountField.getText();

            // Input validation
            if (patientID.isEmpty() || service.isEmpty() || amountText.isEmpty()) {
                showAlert(AlertType.ERROR, "Validation Error", "Please fill out all fields.");
                return;
            }

            try {
                double amount = Double.parseDouble(amountText);
                saveBillingToDatabase(patientID, service, amount);
                showAlert(AlertType.INFORMATION, "Success", "Billing information saved
successfully.");

                // Clear fields after saving
                patientIDField.clear();
                serviceField.clear();
            }
        });
    }
}

```

```

        amountField.clear();
    } catch (NumberFormatException ex) {
        showAlert(AlertType.ERROR, "Invalid Input", "Amount must be a valid
number.");
    }
});

// Layout for the billing form
VBox layout = new VBox(10);
layout.getChildren().addAll(patientIDField, serviceField, amountField, saveButton);

// Set background image with CSS
layout.setStyle("-fx-background-image: url('file:/D:/image.jpg');"
    + "-fx-background-size: cover;");

// Set the scene and window properties
Scene scene = new Scene(layout, 400, 300);
billingStage.setScene(scene);
billingStage.show(); // Show the Billing window
}

// Database connection details
private Connection connectToDatabase() throws SQLException {
    String url = "jdbc:mysql://localhost:3306/hospital_management";
    String user = "root"; // replace with your MySQL username
    String password = "kumaran"; // replace with your MySQL password
    return DriverManager.getConnection(url, user, password);
}

// Save billing information to database
private void saveBillingToDatabase(String patientID, String service, double amount) {
    String sql = "INSERT INTO billing (patient_id, service, amount) VALUES (?, ?, ?)";

    try (Connection conn = connectToDatabase();
        PreparedStatement pstmt = conn.prepareStatement(sql)) {

        pstmt.setString(1, patientID);
        pstmt.setString(2, service);
        pstmt.setDouble(3, amount);
        pstmt.executeUpdate();
    } catch (SQLException e) {
        showAlert(AlertType.ERROR, "Database Error", "Failed to save billing information:
" + e.getMessage());
    }
}

// Show alert helper method
private void showAlert(AlertType alertType, String title, String message) {
    Alert alert = new Alert(alertType);
    alert.setTitle(title);

```

```

        alert.setHeaderText(null);
        alert.setContentText(message);
        alert.showAndWait();
    }

    public void show() {
        // Assuming this method is supposed to show the billing window
    }
}

```

## PAYMENTS PAGE DESIGN

```

import javafx.scene.Scene;
import javafx.scene.control.Alert;
import javafx.scene.control.Button;
import javafx.scene.control.TextField;
import javafx.scene.layout.VBox;
import javafx.stage.Stage;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.SQLException;

public class Payments extends Stage {

    public Payments() {
        setTitle("Payments");

        // Create TextFields with styling
        TextField billingIDField = new TextField();
        billingIDField.setPromptText("Enter Billing ID");
        billingIDField.setStyle("-fx-font-family: 'Arial'; -fx-font-size: 14px; -fx-font-weight: bold; -fx-text-fill: black;");

        TextField amountField = new TextField();
        amountField.setPromptText("Enter Payment Amount");
        amountField.setStyle("-fx-font-family: 'Arial'; -fx-font-size: 14px; -fx-font-weight: bold; -fx-text-fill: black;");

        TextField dateField = new TextField();
        dateField.setPromptText("Enter Payment Date");
        dateField.setStyle("-fx-font-family: 'Arial'; -fx-font-size: 14px; -fx-font-weight: bold; -fx-text-fill: black;");

        // Create Save Button with styling
        Button saveButton = new Button("Save Payment");
        saveButton.setStyle("-fx-font-family: 'Arial'; -fx-font-size: 16px; -fx-font-weight: bold; -fx-text-fill: white; -fx-background-color: black;");

        // Action for Save Button
    }
}

```

```

        saveButton.setOnAction(e -> savePayment(billingIDField.getText(),
amountField.getText(), dateField.getText()));

// Layout for the payment form
VBox layout = new VBox(10);
layout.getChildren().addAll(billingIDField, amountField, dateField, saveButton);

// Set background image and styling
layout.setStyle("-fx-background-image: url('file:/D:/image.jpg');"
    + "-fx-background-size: cover;");

// Set the scene and window properties
Scene scene = new Scene(layout, 400, 300);
setScene(scene);
}

// Method to save payment information to the database
private void savePayment(String billingID, String amount, String date) {
    // SQL query to insert payment details
    String query = "INSERT INTO payments (billing_id, amount, payment_date) VALUES
(?, ?, ?)";

    // Connect to the database and execute the query
    try (Connection connection = DatabaseConnection.connect();
        PreparedStatement statement = connection.prepareStatement(query)) {

        statement.setString(1, billingID);
        statement.setString(2, amount);
        statement.setString(3, date);

        int rowsAffected = statement.executeUpdate();
        if (rowsAffected > 0) {
            showAlert(Alert.AlertType.INFORMATION, "Success", "Payment saved
successfully.");
        }

    } catch (SQLException e) {
        showAlert(Alert.AlertType.ERROR, "Error", "Failed to save payment: " +
e.getMessage());
    }
}

// Helper method to display alerts
private void showAlert(Alert.AlertType alertType, String title, String message) {
    Alert alert = new Alert(alertType);
    alert.setTitle(title);
    alert.setContentText(message);
    alert.showAndWait();
}
}

```

## INVOICE PAGE DESIGN

```
import javafx.scene.Scene;
import javafx.scene.control.TextField;
import javafx.scene.control.Button;
import javafx.scene.layout.VBox;
import javafx.stage.Stage;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import javafx.scene.control.Alert;
import javafx.scene.control.Alert.AlertType;

public class Invoices extends Stage {

    public Invoices() {
        setTitle("Invoices");

        // Create TextFields with CSS styles
        TextField billingIDField = new TextField();
        billingIDField.setPromptText("Enter Billing ID");
        billingIDField.setStyle("-fx-font-family: 'Arial'; -fx-font-size: 14px; -fx-font-weight: bold; -fx-text-fill: black;");

        TextField dateField = new TextField();
        dateField.setPromptText("Enter Date (YYYY-MM-DD)");
        dateField.setStyle("-fx-font-family: 'Arial'; -fx-font-size: 14px; -fx-font-weight: bold; -fx-text-fill: black;");

        TextField totalField = new TextField();
        totalField.setPromptText("Enter Total");
        totalField.setStyle("-fx-font-family: 'Arial'; -fx-font-size: 14px; -fx-font-weight: bold; -fx-text-fill: black;");

        // Create Save Button with style
        Button saveButton = new Button("Save Invoice");
        saveButton.setStyle("-fx-font-family: 'Arial'; -fx-font-size: 16px; -fx-font-weight: bold; -fx-text-fill: white; -fx-background-color: black;");

        // Set the save action
        saveButton.setOnAction(e -> {
            String billingID = billingIDField.getText();
            String date = dateField.getText();
            String totalText = totalField.getText();

            // Input validation
            if (billingID.isEmpty() || date.isEmpty() || totalText.isEmpty()) {
```

```

        showAlert(AlertType.ERROR, "Validation Error", "Please fill out all fields.");
        return;
    }

    try {
        double total = Double.parseDouble(totalText);
        saveInvoiceToDatabase(billingID, date, total);
        showAlert(AlertType.INFORMATION, "Success", "Invoice saved successfully.");

        // Clear fields after saving
        billingIDField.clear();
        dateField.clear();
        totalField.clear();
    } catch (NumberFormatException ex) {
        showAlert(AlertType.ERROR, "Invalid Input", "Total must be a valid number.");
    }
});

// Layout for the invoice form
VBox layout = new VBox(10);
layout.getChildren().addAll(billingIDField, dateField, totalField, saveButton);

// Set background image with CSS style
layout.setStyle("-fx-background-image: url('file:/D:/image.jpg');"
    + "-fx-background-size: cover;");

// Set the scene and window properties
Scene scene = new Scene(layout, 400, 300);
setScene(scene);
}

// Database connection details
private Connection connectToDatabase() throws SQLException {
    String url = "jdbc:mysql://localhost:3306/hospital_management";
    String user = "root"; // replace with your MySQL username
    String password = "kumaran"; // replace with your MySQL password
    return DriverManager.getConnection(url, user, password);
}

// Save invoice information to the database
private void saveInvoiceToDatabase(String billingID, String date, double total) {
    String sql = "INSERT INTO invoices (billing_id, date, total) VALUES (?, ?, ?)";

    try (Connection conn = connectToDatabase();
        PreparedStatement pstmt = conn.prepareStatement(sql)) {

        pstmt.setString(1, billingID);
        pstmt.setString(2, date);
        pstmt.setDouble(3, total);
        pstmt.executeUpdate();
    }
}

```

```

        } catch (SQLException e) {
            showAlert(AlertType.ERROR, "Database Error", "Failed to save invoice: " +
e.getMessage());
        }
    }

    // Show alert helper method
    private void showAlert(AlertType alertType, String title, String message) {
        Alert alert = new Alert(alertType);
        alert.setTitle(title);
        alert.setHeaderText(null);
        alert.setContentText(message);
        alert.showAndWait();
    }
}

```

## **DATABASE CONNECTIVITY**

```

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class DatabaseConnection {
    private static final String URL = "jdbc:mysql://localhost:3306/hospital_management";
    private static final String USER = "root";
    private static final String PASSWORD = "kumaran";

    public static Connection connect() {
        try {
            return DriverManager.getConnection(URL, USER, PASSWORD);
        } catch (SQLException e) {
            System.out.println("Database connection error: " + e.getMessage());
            return null;
        }
    }
}

```

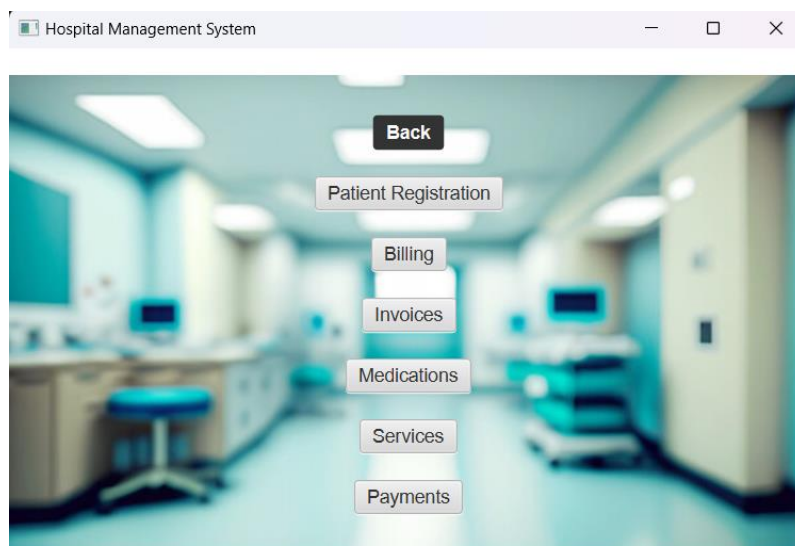


## 6.SNAPSHOTS :

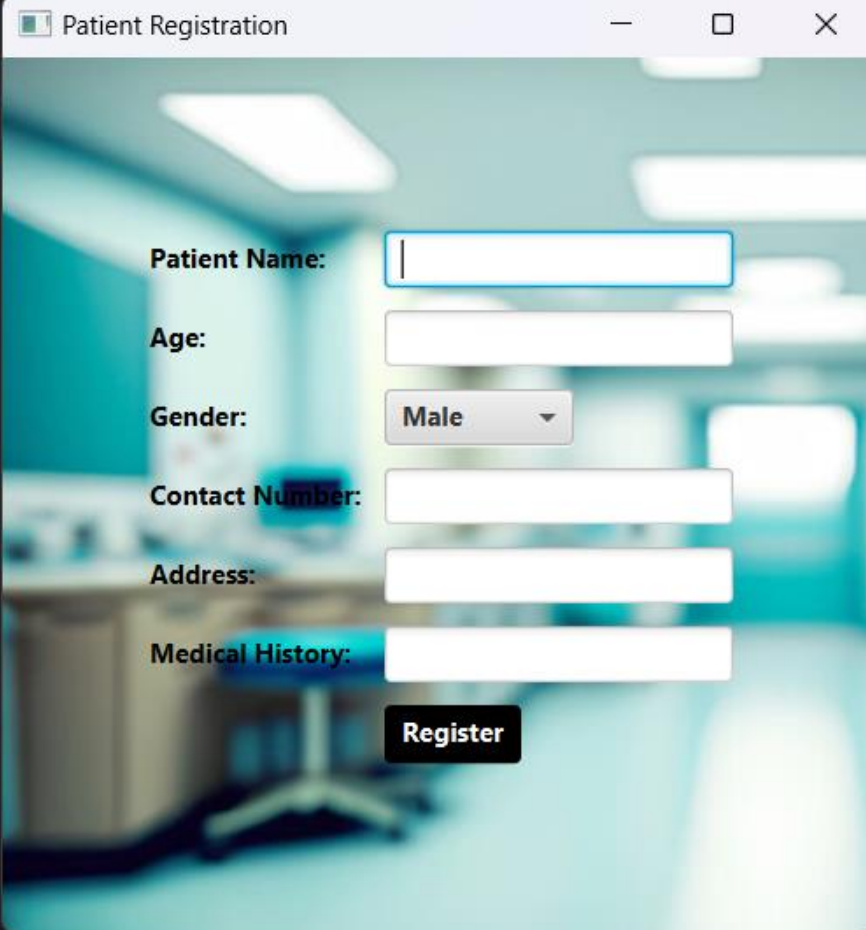
### 6.1 LOGIN PAGE:



### 6.2 HOME PAGE:



### 6.3 PATIENT REGISTRATION PAGE:



A screenshot of a web application window titled "Patient Registration". The window has a light blue header bar with the title and standard window controls (minimize, maximize, close). The background of the form is a blurred image of a hospital corridor. The form contains the following fields and controls:

- Patient Name:** A text input field with a blue border.
- Age:** A text input field.
- Gender:** A dropdown menu with "Male" selected.
- Contact Number:** A text input field.
- Address:** A text input field.
- Medical History:** A text input field.
- Register:** A black button with white text.

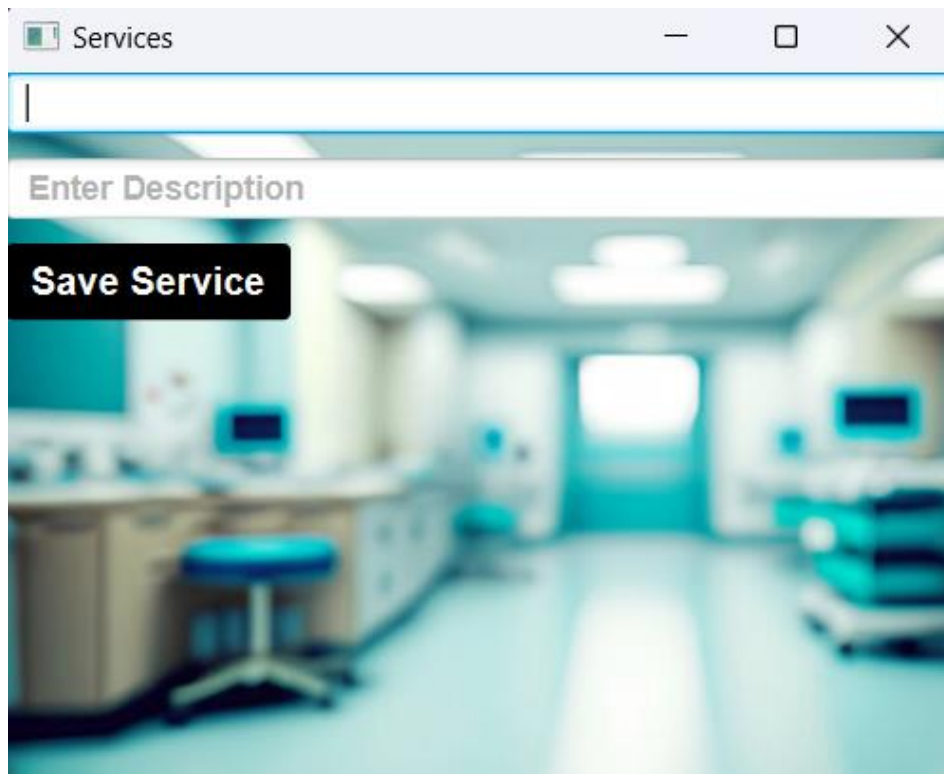
### 6.4 MEDICATIONS PAGE:



A screenshot of a web application window titled "Medications". The window has a light blue header bar with the title and standard window controls (minimize, maximize, close). The background of the form is a blurred image of a hospital corridor. The form contains the following fields and controls:

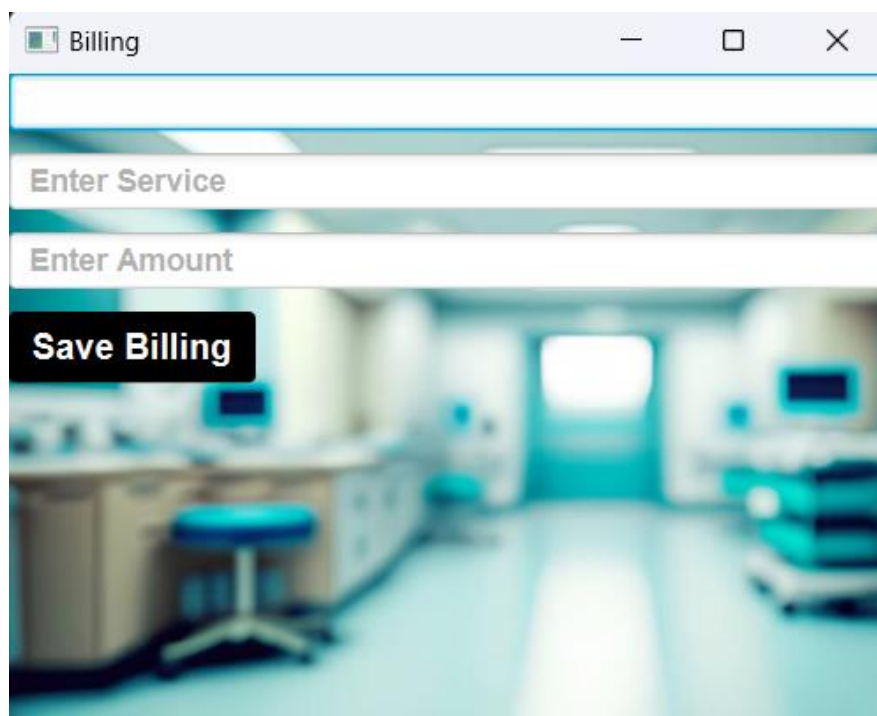
- Enter Medication:** A text input field.
- Enter Dosage:** A text input field.
- Save Medication:** A black button with white text.

## 6.5 SERVICES PAGE:



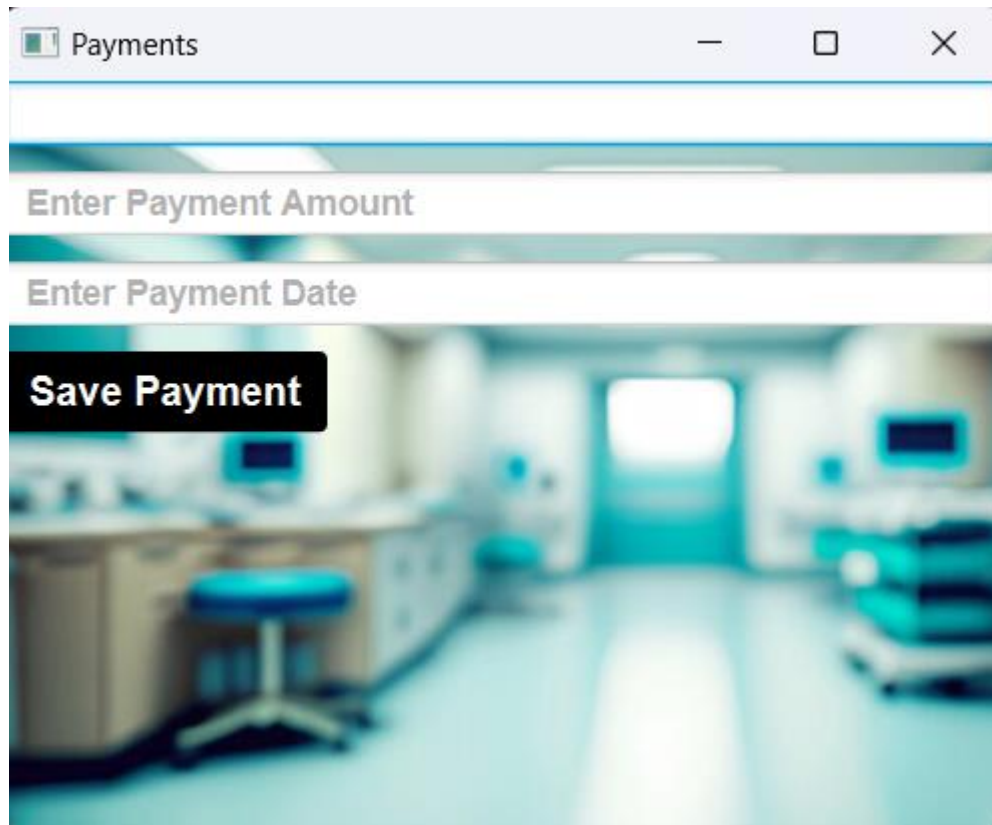
The screenshot shows a web browser window titled "Services". The page has a light blue header bar. Below the header, there is a text input field with a placeholder "Enter Description". To the left of the input field, there is a black button with white text that says "Save Service". The background of the page is a blurred image of a hospital corridor with blue walls and a white floor.

## 6.6 BILLING PAGE:



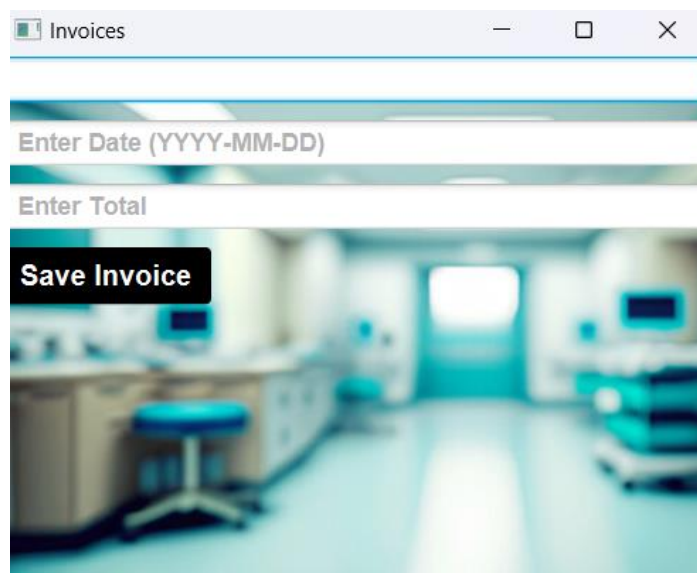
The screenshot shows a web browser window titled "Billing". The page has a light blue header bar. Below the header, there are two text input fields. The first input field has a placeholder "Enter Service" and the second input field has a placeholder "Enter Amount". To the left of the input fields, there is a black button with white text that says "Save Billing". The background of the page is a blurred image of a hospital corridor with blue walls and a white floor.

## 6.7 PAYMENTS PAGE:



The screenshot shows a web browser window titled "Payments". The page has a light blue header bar. Below the header, there is a white input field. Underneath the input field, there are two labels: "Enter Payment Amount" and "Enter Payment Date". At the bottom of the form, there is a black button with the text "Save Payment". The background of the page is a blurred image of a hospital corridor.

## 6.8 INVOICE PAGE:



The screenshot shows a web browser window titled "Invoices". The page has a light blue header bar. Below the header, there is a white input field. Underneath the input field, there are two labels: "Enter Date (YYYY-MM-DD)" and "Enter Total". At the bottom of the form, there is a black button with the text "Save Invoice". The background of the page is a blurred image of a hospital corridor.

## **7.CONCLUSION:**

The Hospital Invoice Management System project was successfully completed, integrating key functionalities like billing, payments, and service management. Developed using JavaFX for the user interface and MySQL for data management, the system ensures efficient, secure, and user-friendly operations.

The project effectively streamlines hospital processes, supports data integrity, and offers scalability for future enhancements such as appointment scheduling or role-based access. Overall, this system is a robust solution for improving operational efficiency in healthcare management.

## **8.REFERENCES:**

1. [<https://openjfx.io>](<https://openjfx.io>)
2. [<https://dev.mysql.com/doc/>](<https://dev.mysql.com/doc/>)
3. [<https://docs.oracle.com/javase/tutorial/jdbc/>](<https://docs.oracle.com/javase/tutorial/jdbc/>)
4. [<https://www.w3schools.com/sql/>](<https://www.w3schools.com/sql/>)
5. [<https://stackoverflow.com/>](<https://stackoverflow.com/>)