

# HOSPITAL MANAGEMENT SYSTEM (HMS)

(Presented by M.Ahmad (SP25-BSE-019) , Rabia Tehreem(SP25-BSE-048))

December 16, 2025

## INTRODUCTION

The **Hospital Management System (HMS)** is a desktop application developed using Java and JavaFX to automate hospital operations. The system allows hospital administrators to manage patients, doctors, appointments, and billing efficiently.

The application uses **CSV files** for persistent storage, making it lightweight and easy to manage without a database.

**Target Users:** Hospital administrators and staff responsible for managing hospital records.

### **Purpose of the Project:**

- Reduce manual errors in record-keeping.
- Streamline hospital operations including patient and doctor management.
- Enable quick billing and reporting of hospital activities.

## OBJECTIVES OF THE SYSTEM

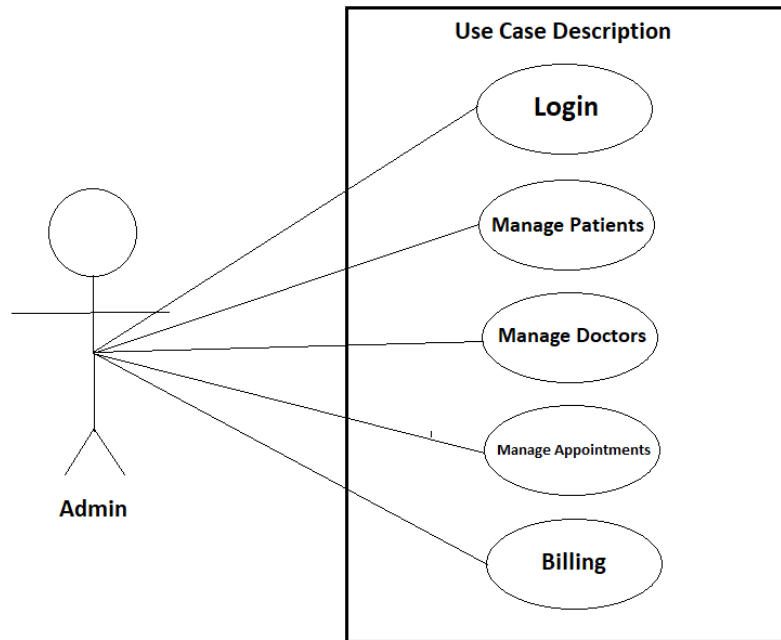
The main objectives of the Hospital Management System are:

- Manage Patients: Add, update, delete, and search patient records.
- Manage Doctors: Add, update, delete, and search doctor records.
- Manage Appointments: Schedule and search appointments.
- Billing: Calculate service and medicine charges, maintain total bill.
- Provide an intuitive GUI for easy interaction.
- Persist data reliably in CSV files for future retrieval.

## SCOPE OF THE PROJECT

- Admin login with authentication.
- Full CRUD operations for Patients, Doctors, and Appointments.
- Bill generation and management for each patient.
- Dashboard displaying all tables with search functionality.
- Limitations: Currently uses CSV files.

# MAIN USE CASES



## Use Case Description:

Use Case	Description	Steps
Login	Admin authenticates to access the system	Enter username and password → Validate credentials → Access dashboard
Manage Patients	Add, update, delete, or search patient records	Fill patient form → Submit → Refresh table → Search by Bill ID
Manage Doctors	Add, update, delete, or search doctor records	Fill doctor form → Submit → Refresh table → Search by Row#
Manage Appointments	Schedule or search appointments	Select patient & doctor → Select date → Submit → Refresh table → Search by ID
Billing	Generate and clear bills for patients	Select patient → Enter charges → Generate total

## SYSTEM ARCHITECTURE

The HMS uses a layered architecture:

- Model Layer: Represents entities – Patient, Doctor, Appointment, Bill.
- Repository Layer: PatientRepository , DoctorRepository , AppointmentRepository – responsible for CRUD and file operations.
- Controller/GUI Layer: App class manages JavaFX GUI including login page, dashboard, and panels.

The system follows OOP principles for modularity, encapsulation, and reusability.

Hospital Management System			
Person >		App	
String name	<b>Admin</b>	Admin admin	<b>Application</b>
int age	String username	Hospital hospital1	/* Java
String gender	String password	TableView patientTable	default class
String CNIC	void displayInfo()	TableView doctorTable	
T metaData	getter/setters	TableView appointmentTable	
getters/setters		Scene loginPage(Stage stage)	
	<b>Bill</b>	Scene dashBoard(Stage stage)	
<b>Patient</b>	static int counter	VBox patientPanel()	
<PatientDetails>	int billId	VBox doctorPanel()	
getInfoString()	int serviceCharge	VBox appointmentPanel()	
	int medicineCharge	VBox billingPanel()	<b>Doctor</b>
<b>PatientDetails</b>	int total	void start(Stage stage)	<DoctorDetails>
String Disease	calculateTotal()	void main(String[] args)	getInfoString()
String phone	display BillString()		
Bill bill		<b>Repository &gt;</b>	<b>DoctorDetails</b>
	<b>Appointment</b>	void add(R r1)	String specialization
	int appointmentID	void update(R r1)	int salary
	static int counter	void delete(R r1)	Boolean available
<b>PatientRepository</b>	Patient patient	ObservableList getAll()	
ObservableList <Patient>	Doctor doctor	R getById(int i)	<b>AppointmentRepository</b>
<b>DoctorRepository</b>	String date	void saveToFile()	ObservableList <Appointment>
ObservableList <Doctor>	generateSlipString()		
<b>Hospital</b>			

Class	Attributes	Methods	Responsibility
Person	name, age, gender, CNIC	get/set methods	Base class for Patient and Doctor
Patient	metaData (PatientDetails)	displayInfo String()	Manages patient records
Patient Details	disease, phone, bill (Bill)	get/set methods	Composed inside Patient to store patient-specific info
Doctor	metaData (DoctorDetails)	displayInfo String()	Manages doctor records
Doctor Details	specialization, salary, isAvailable	get/set methods	Composed inside Doctor for doctor-specific info
Appointment	appointmentID, patient, doctor, date	generate SlipString()	Manages appointment records
Bill	serviceCharges, medicineCharges, total	Calculate Total()	Calculates billing totals

Repository <T>	add, update, delete, getAll, getById, saveToFile	Generic CRUD operations	Abstract repository interface
Patient Repository / Doctor Repository / Appointment Repository	ObservableList <T>	CRUD & saveToFile	Implements repository for each entity
Hospital	patientRepo, doctorRepo, appointmentRepo	Construct or	Aggregates all repositories
Admin	username, password	get/set	Manages admin login
App	GUI components	start(), loginPage(), dashBoard() , panel methods	Main JavaFX application

(UML Class Diagram – reflecting exact Java classes and relationships)



## FUNCTIONAL MODULES

- Admin login module
- Patient management module
- Doctor management module
- Appointment management module
- Billing module

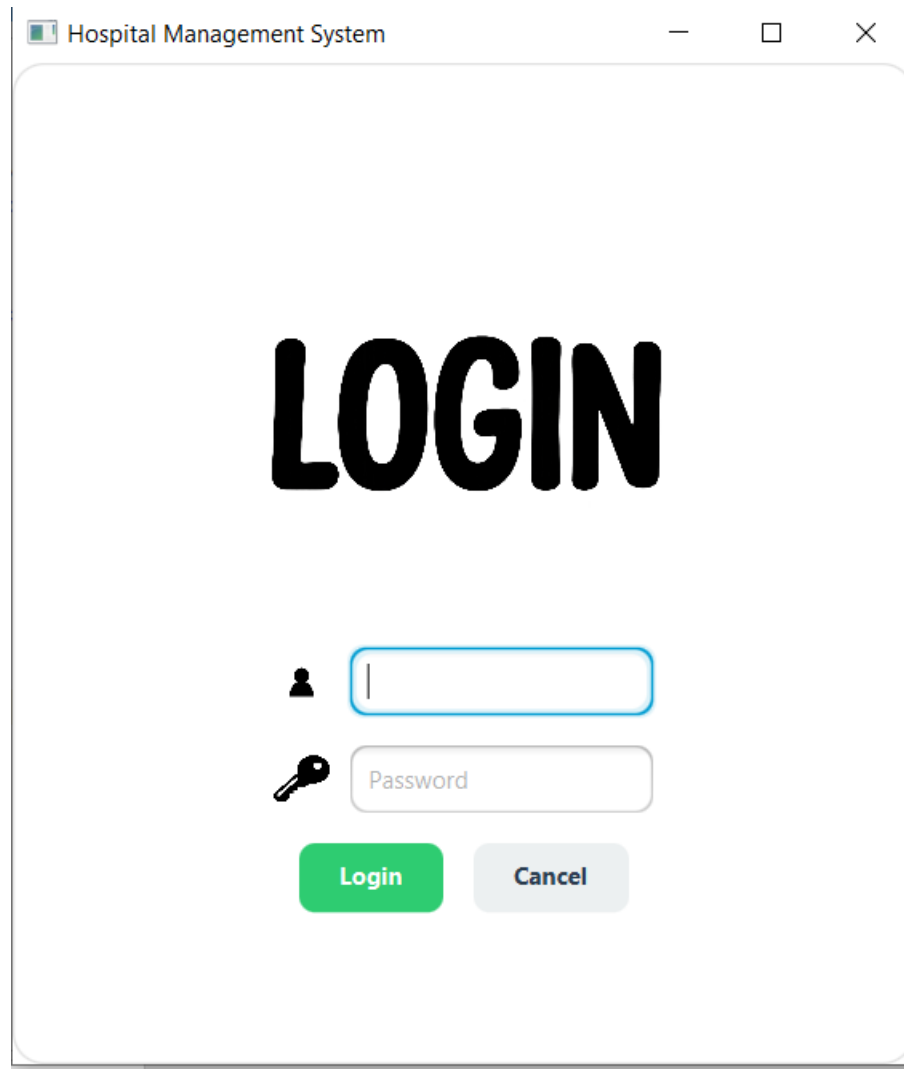
## DATA PERSISTENCE

Data is stored in CSV files:

- PatientData.csv
  - DoctorData.csv
  - AppointmentData.csv
- 
- Each repository handles reading and writing data for persistence.

# USER INTERFACE DESIGN


## 1.Login page:




The image shows a login page for a "Hospital Management System". The page is displayed within a window with a title bar containing the text "Hospital Management System" and standard window control buttons (minimize, maximize, close). The main content area is white and features the word "LOGIN" in large, bold, black capital letters. Below the title, there are two input fields. The first field is preceded by a person icon and has a blue border. The second field is preceded by a key icon and has the placeholder text "Password". At the bottom of the form, there are two buttons: a green "Login" button and a light gray "Cancel" button.


Hospital Management System

# LOGIN





## 2. Dashboard with tables:




Welcome to Hospital Management System!

Home  
Patient  
Doctor  
Appointment  
Billing  
Logout

**Patient's Data:**

Name	Age	Gender	CNIC	Disease	Phone	BillID	Service Charges	Medicine Charges	Total
No content in table									



Welcome to Hospital Management System!

Home  
Patient  
Doctor  
Appointment  
Billing  
Logout

**Doctor's Data:**

Name	Age	Gender	CNIC	Specialization	Salary	Available
No content in table						



Welcome to Hospital Management System!

Home

Patient

Doctor

Appointment

Billing

Logout

### Appointments:

Appointment ID	Patient Name	Doctor Name	Date
No content in table			

## 3. Patient management panel:



Welcome to Hospital Management System!

Home

Patient

Doctor

Appointment

Billing

Logout

### Patient Management

Name

Age

Gender

CNIC

Phone

Disease

billId(Required only for Update/Delete)

Add

Update


Delete

Search Patient by Bill ID

Enter Bill ID

Search

## 4. Doctor management panel:



Welcome to Hospital Management System!

[Home](#)  
[Patient](#)  
[Doctor](#)  
[Appointment](#)  
[Billing](#)  
[Logout](#)


### Doctor Management

[Add](#) [Update](#) [Delete](#)

Search Doctor by Row#

[Search](#)

## 5. Appointment management panel:



Welcome to Hospital Management System!

[Home](#)  
[Patient](#)  
[Doctor](#)  
[Appointment](#)  
[Billing](#)  
[Logout](#)

### Appointment Management

Select Patient

Select Doctor

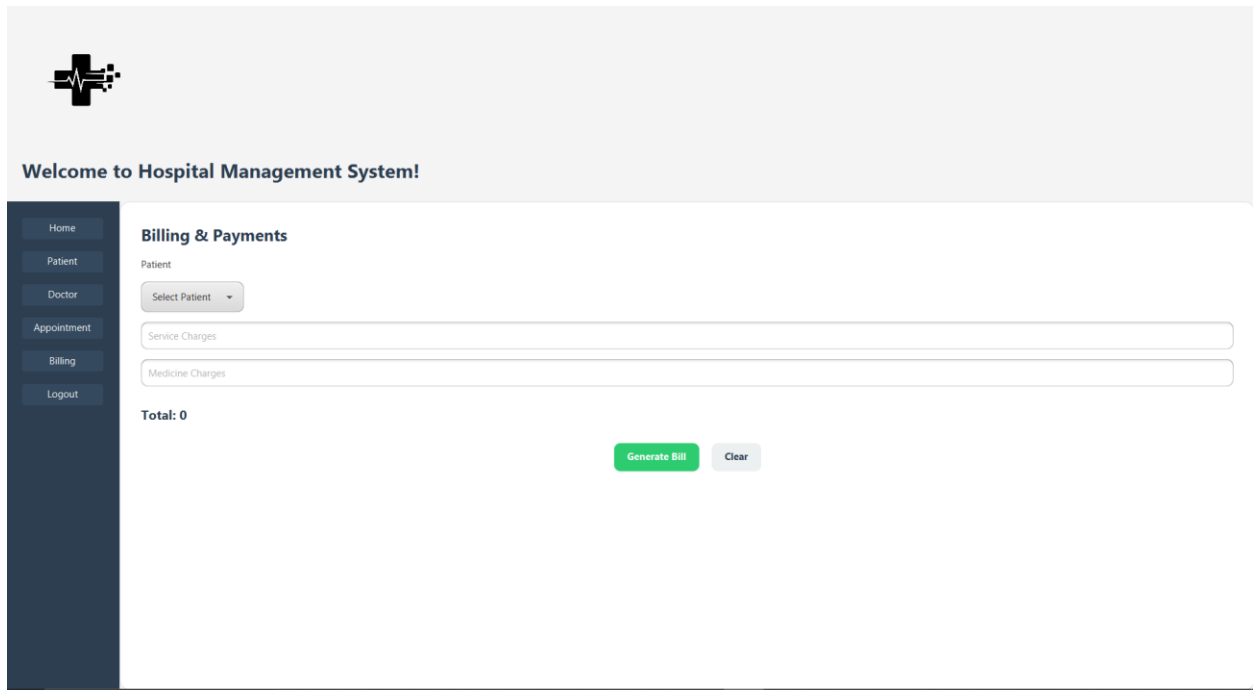
Select Date

[Add Appointment](#)

Search Appointment by ID

[Search](#)

## 6. Billing panel:



The screenshot shows a web application interface for a Hospital Management System. At the top, there is a logo of a cross with a heart rate line. Below the logo, a welcome message reads "Welcome to Hospital Management System!". The main content area is titled "Billing & Payments" and includes a sidebar with navigation links: Home, Patient, Doctor, Appointment, Billing, and Logout. The "Billing & Payments" section contains a "Patient" dropdown menu labeled "Select Patient", two input fields for "Service Charges" and "Medicine Charges", and a "Total: 0" label. At the bottom of the section, there are two buttons: "Generate Bill" (green) and "Clear" (grey).

## OOP CONCEPTS USED

**Inheritance:** Doctor and Patient extend Person.

**Encapsulation:** Private fields with getters/setters.

**Polymorphism:** Doctor being up-casted and down-casted simultaneously.

**Abstraction:** Repository interface abstracts common CRUD operations.

**Composition:** Patient → PatientDetails, Doctor → DoctorDetails.

## **LIMITATIONS**

- CSV-based persistence only.
- Only one user role (Admin).
- Appointment conflicts are not auto-checked.
- No real database integration.

These limitations can be addressed in future enhancements.

## **FUTURE ENHANCEMENTS**

- Integrate with SQL or NoSQL database.
- Add multiple user roles (Admin, Doctor, Receptionist).
- Automatic appointment conflict detection.
- Generate PDF bills and appointment slips.
- Add notifications for appointments.

## **CONCLUSION**

The HMS effectively simplifies hospital operations by providing a fully object-oriented, modular, and user-friendly application. Its GUI allows seamless management of patients, doctors, appointments, and billing, ensuring better hospital workflow efficiency. Future enhancements can make it even more robust and feature-rich.