PROJECT REPORT

ON

**Hospital Management System**

Submitted in partial fulfillment of the

Academic requirements of the KL (Deemed to be) University

for the award of degree

BTech - Computer Science and Engineering

IN

Honors

SUBMITTED BY

**2310030412 N.Aravind**

**2310030120 D.Ganesh**

**2310030121 K.RaviKiran**

**2310030193 Karthik Reddy**

**Under the guidence of**

**Mr. Sasidhar.K**

**Computer Science and Engineering Department**

**KL (Deemed to be) UNIVERSITY,**

**Off Campus, Hyderabad**

**2023**

**CERTIFICATE**

This is to certify that the project report entitled “**Hospital Management System**” being submitted by

**2310030412 N.Aravind**

**2310030120 D.Ganesh**

**2310030121 K.RaviKiran**

**2310030193 Karthik Reddy**

towards partial fulfillment for the award of BTech Computer Science and Engineering in Honors is a record of bonafide work carried out by them. The results embodied in this report have not been submitted to any other University or Institution for the award of any degree or diploma.

**Mr. Sasidhar.K Dr. Arpita Gupta** Internal guide HOD, CSE

.

**ACKNOWLEDGEMENT**

We express our heartfelt gratitude and sincere thanks to Dr. Srinivas Rao, Principal, Dr. Arpita Gupta, HOD, CSE Department and Mr. Sasidhar.K internal guide for their constant encouragement and support in the fulfillment of our project.

We also express our sincere thanks for their valuable suggestions and advices without which this project would not have seen the light of the day.

We give in our gratitude and indebtedness to the Dean Academics and Academic Coordinator for their support in completion of our project.

Finally, we would also like to thank the people who directly or indirectly helped us and parents and friends for their cooperation in completion of the project work.

**2310030412 N.Aravind**

**2310030120 D.Ganesh**

**2310030121 K.RaviKiran**

**2310030193 Karthik Reddy**

# ABSTRACT

The Hospital Management System is a comprehensive Java-based application designed to streamline the operations of a hospital by automating essential tasks such as patient admission, employee information management, room and department handling, ambulance service, and image management. It ensures efficient record-keeping, faster patient service, and effective resource utilization through modular implementation and real-time database interactions.

The application is structured to handle patient admissions and discharges, manage room availability, track ambulance services, allow the uploading and viewing of patient-related images, and enable advanced searching and filtering of data. With a layered MVC architecture and database integration, this project highlights the practical application of software engineering principles in the healthcare domain.

**CONTENTS**

1. **INTRODUCTION**
2. **LITERATURE SURVEY**
3. **PROPOSED SYSTEM**
4. **DESIGN**
5. **SAMPLE CODE**
6. **TESTING**
7. **SCREENSHOTS**
8. **CONCLUSION AND FUTURE SCOPE**

**REFERENCES AND BIBLIOGRAPHY**

**LIST OF TABLES**

|  |  |  |
| --- | --- | --- |
| **Table No** | **Table Title** | **Page No** |
| 1.1 | WEB BASED APPLICATION | 1 |
| 1.2 | SOFTWARE DEVELOPMENT METHODOLOGY | 1 |
| 2.1 | EXISTING SYSTEM | 2 |
| 3.1 | PROJECT ENVIRONMENT SPECIFICATION | 4 |
| 3.2 | FEASIBILITY STUDY | 5 |
| 3.3 | DEVELOPMENT MODEL | 6 |
| 4.1 | MODEL ARCHITECTURE | 7 |
| 4.2 | MODULES | 8 |
| 4.3 | FLOW OF THE SCENARIO | 9 |
| 4.4 | UML DIAGRAMS | 9 |
| 5.1 | SAMPLE CODE | 10 |
| 6.1 | SYSTEM TESTING | 11 |
| 6.2 | TESTING LEVELS | 12 |
| 6.3 | TESTING APPROACH | 13 |
| 7 | SCRENSHOTS | 14 - 18 |
| 8.1 | CONCLUSION | 19 |
| 8.2 | FUTURE SCOPE | 20 |
|  | REFERENCES AND BIBILOGRAPHY | 21 |

# INTRODUCTION

The **Hospital Management System** is a software application designed to manage the operations of a hospital in an efficient, automated, and organized manner. This system helps with functions such as patient registration, doctor scheduling, treatment records, billing, and staff management. Built using Advanced Object-Oriented Programming concepts, it provides a modular, scalable, and user-friendly solution to streamline hospital operations.

# WEB BASED APPLICATION:

The system is designed as a web-based application to ensure easy accessibility, centralized data management, and platform independence.

**1.1.1 Browser Based:**

The application can be accessed through standard web browsers (like Chrome, Firefox, or Edge) without the need to install additional software. This makes it convenient for hospital staff to access the system from any device connected to the hospital network.

# 1.2 SOFTWARE DEVELOPMENT METHODOLOGY (SDLC CYCLE):

The development of the Hospital Management System follows the **Software Development Life Cycle (SDLC)** to ensure a systematic, disciplined, and quantifiable approach.

* **Requirement Analysis:** Understanding and documenting the needs of users including doctors, patients, and staff.
* **System Design:** Defining system architecture, class structures, and user interfaces using OOP principles.
* **Implementation (Coding):** Writing modular and reusable code using classes, inheritance, and polymorphism.
* **Testing:** Verifying the system for errors, bugs, and inconsistencies using unit testing and integration testing.
* **Maintenance:** Regular updates and improvements based on feedback and new requirements.

# 2. LITERATURE SURVEY

**2.1 EXISTING SYSTEM:**

In many hospitals, the management of records and day-to-day operations is still done manually or with basic software tools like spreadsheets and paper-based files. This existing system presents several challenges:

* **Time-Consuming Processes:** Manual entries for patient registration, billing, and appointment scheduling take up a significant amount of time and effort.
* **Data Inconsistency:** Records may be misplaced, duplicated, or inconsistently updated, leading to errors in patient information or medical history.
* **Lack of Centralization:** Information is often scattered across different departments, making it hard to access complete records quickly.
* **Limited Accessibility:** Paper files or local software can only be accessed from specific locations, limiting flexibility and delaying processes during emergencies.
* **Security Risks:** Physical documents are more prone to damage, loss, or unauthorized access.

These limitations clearly show the need for a **centralized, web-based, object-oriented hospital management system** that enhances reliability, efficiency, and security of hospital operations.

**3. PROPOSED SYSTEM**

The proposed Hospital Management System is a web-based, object-oriented application designed to overcome the limitations of the existing manual and semi-automated systems. It provides a centralized, secure, and efficient platform for managing hospital operations including patient care, doctor scheduling, billing, staff details, and medical records.

Key Features of the Proposed System:

* Centralized Data Management: All patient, doctor, staff, and treatment data are stored in a central database that can be accessed from anywhere within the hospital network.
* User-Friendly Interface: Intuitive UI for different user roles such as admin, doctor, nurse, and receptionist to perform their specific tasks easily.
* Role-Based Access Control: Secure login system ensures that users can only access the modules relevant to their roles.
* Automated Scheduling: Efficient appointment booking system to manage doctor availability and patient appointments.
* Electronic Medical Records (EMR): Digital storage of patient diagnosis, treatment history, and prescriptions.
* Report Generation: Admins and doctors can generate detailed reports for patients, finances, and hospital statistics.

Advantages over Existing System:

* Faster access to information
* Reduced human errors
* Improved patient care and hospital service quality
* Easy to maintain and scale using OOP principles

# 3.1. PROJECT ENVIRONMENT SPECIFICATION:

**3.1.1 Hardware**

|  |  |
| --- | --- |
| **Processor** | Pentium IV or higher |
| **Hard Disk** | 40 GB or above |
| **RAM** | 512 MB or above |

**3.1.2 Software**

**3.1.2 Software Requirements**

**a) Front-End Tools**

* HTML – Used to structure the user interface forms and layouts.
* JavaScript – Used for basic form validations and dynamic behavior.
* CSS (Cascading Style Sheets) – Used for styling and layout enhancements.

Note: Although your main GUI is developed using Java Swing (JFrames), this section can reflect a conceptual web alternative or previous prototype.

**b) Software Interface**

* Operating System: Windows XP or higher (preferably Windows 10)
* Development Platform: IntelliJ IDEA
* Database: MySQL
* Connectivity: JDBC (Java Database Connectivity)

**c) Design Tools**

* UML (Unified Modeling Language) – Used for designing use-case diagrams, class diagrams, sequence diagrams, etc.

# 3.2 FEASIBILITY STUDY:

he feasibility study ensures that the project is viable from technical, economic, and operational perspectives.

**1. Technical Feasibility**

* The project uses proven technologies such as Java, MySQL, and Swing, which are well-supported and reliable.
* The development environment is set up using commonly available tools like IntelliJ IDEA and JDBC connectors.

**2. Operational Feasibility**

* The application simplifies and automates hospital activities, improving efficiency.
* User-friendly GUI makes it easy for non-technical users to interact with the system.

**3. Economic Feasibility**

* The project is cost-effective as it uses open-source tools.
* No high-end hardware is required; the system can run on basic configurations.

# 3.3 DEVELOPMENT MODEL:

**Waterfall Model**

The **Waterfall model** was followed for developing the Hospital Management System. It is a linear and sequential approach where each phase must be completed before moving to the next.

**Phases Followed:**

1. **Requirement Analysis:**  
   Collected and documented all functional and non-functional system requirements.
2. **Design:**  
   Created UML diagrams, database schemas, and UI mockups for system structure.
3. **Implementation:**  
   Developed application modules using Java Swing for the GUI and MySQL for backend storage.
4. **Testing:**  
   Verified functionality through unit, integration, system, and UAT testing.
5. **Deployment:**  
   Released the fully tested system for actual use in a live environment.
6. **Maintenance:**  
   Performed regular updates, bug fixes, and feature enhancements based on user feedback.

This model ensured a structured and disciplined development cycle with clear deliverables at each stage.

# 4. DESIGN

# 4.1 MODEL ARCHITECTURE:

# The proposed Hospital Management System is built on a three-tier architecture, promoting modularity, scalability, and ease of maintenance.

# • Presentation Layer (Frontend):

# Provides user interfaces for various roles like admin, doctor, nurse, and receptionist.

# Developed using HTML, CSS, and JavaScript to ensure responsiveness and a smooth user experience.

# • Application Layer (Backend Logic):

# Implements business logic using Object-Oriented Programming (OOP) principles.

# Manages request handling, data validation, and session control.

# Acts as the bridge between the UI and the database.

# • Database Layer (Storage):

# Handles storage and retrieval of persistent data such as patient records, appointments, bills, and prescriptions.

# Typically uses a relational database (e.g., MySQL) but can be adapted for NoSQL if needed.

# 4.2 MODULES:

# User Management Module: Handles registration, login, and role-based access (admin, doctor, nurse, receptionist).

# Patient Management Module: Stores patient details, history, and medical reports. Allows new patient entry and updates.

# Doctor Management Module: Maintains doctor profiles, specialties, schedules, and appointment slots.

# Appointment Scheduling Module: Enables patients or staff to book, update, and cancel appointments based on doctor availability.

# Billing Module: Automates invoice generation based on treatments, tests, and medications.

# Report & Prescription Module: Allows doctors to upload prescriptions and medical reports for each consultation.

# Staff Management Module: Maintains details of nurses, ward boys, lab technicians, etc.

# Feedback/Helpdesk Module (Optional): Enables patients or users to raise queries or feedback to the admin.

# 4.3 FLOW OF THE SCENARIO:

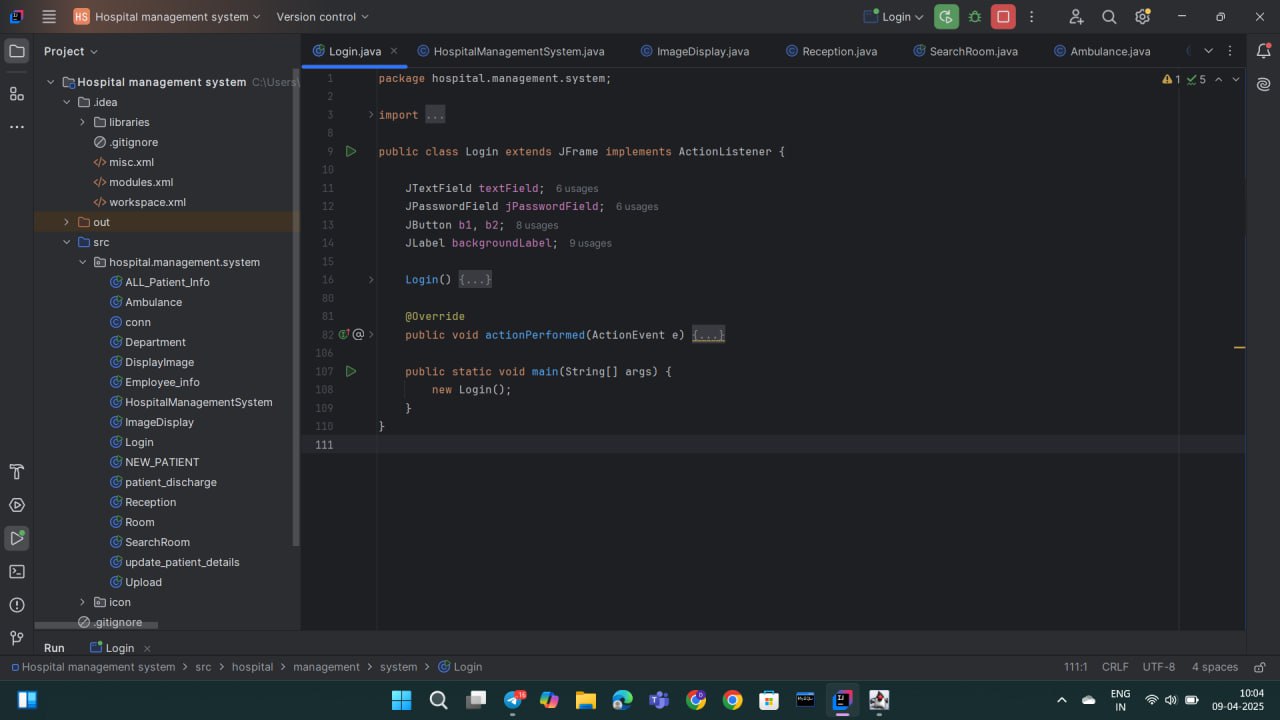
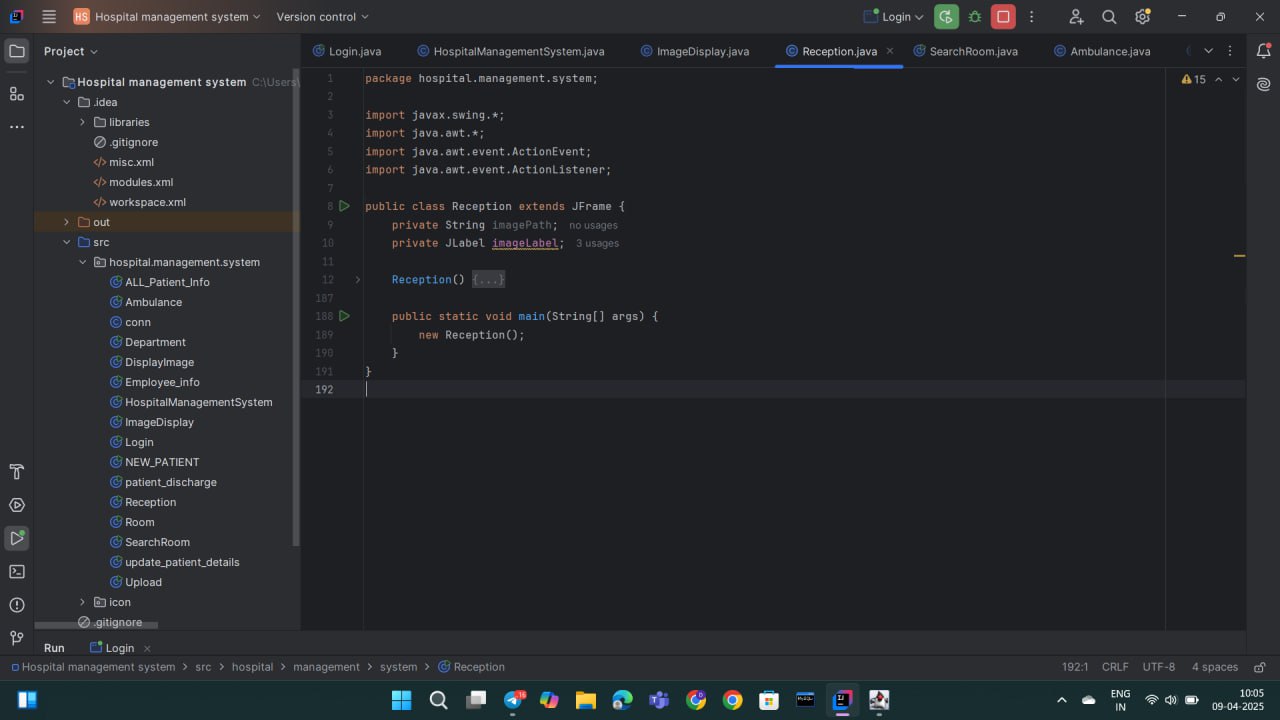
1. User logs in based on their role.
2. Admin manages staff, doctors, and patients.
3. Receptionist registers new patients and schedules appointments.
4. Doctors view appointments, treat patients, and upload prescriptions.
5. Patients receive treatment, and billing is auto-generated.
6. All actions are logged and stored in the database for future access or reporting.

**4.4 UML DIAGRAMS:**

1. **Use Case Diagram:**  
   Shows interactions between users (actors) and the system (e.g., Admin, Doctor, Patient) and their use cases like Register Patient, Generate Bill, View Appointments.
2. **Class Diagram:**  
   Demonstrates the object-oriented structure with classes like User, Patient, Doctor, Appointment, Bill, showing attributes and methods with inheritance and relationships.
3. **Sequence Diagram:**  
   Visualizes the sequence of operations for booking an appointment, including interactions between the user interface, backend logic, and database.
4. **Activity Diagram:**  
   Describes the workflow of actions for modules like appointment scheduling or patient treatment process.
5. **ER Diagram:**  
   Represents the database structure with entities such as Patients, Doctors, Appointments, Bills, and their relationships

**5. DESIGN**

**5.1. Sample Code**

****

# 6. TESTING

**6.1 SYSTEM TESTING:**

System testing was carried out after integrating all modules of the Hospital Management System to ensure the system functions correctly as a whole. It focused on validating complete workflows, data integrity, and UI responsiveness.

**Objectives:**

* **End-to-End Workflow Validation:**
  + Tested full processes such as patient registration → appointment → consultation → billing → report generation.
* **Database Integration (MySQL via JDBC):**
  + Verified CRUD operations, foreign key constraints, and rollback mechanisms in case of failure.
* **GUI Testing (Java Swing):**
  + Checked input validation, responsive buttons, proper form transitions, and correct rendering of UI elements.
* **Inter-Module Communication:**
  + Ensured smooth data flow between modules (e.g., diagnosis results reflecting in billing and discharge).
* **Security & Access Control:**
  + Verified restricted access for sensitive modules based on user roles (admin, doctor, receptionist).
* **Performance & Load:**
  + Simulated concurrent users and tested behavior under load to check system responsiveness.
* **Error Handling:**
  + Checked that appropriate messages appear for invalid inputs and system exceptions.
* **Backup and Recovery:**
  + Validated data backup functionality and tested recovery in case of system failure.

Testing was conducted manually and with tools like JUnit and SQL queries. Results confirmed the system is stable, secure, and ready for deployment.

**6.2 TESTING LEVELS:**

To ensure comprehensive quality, the Hospital Management System was tested at multiple levels:

1. **Unit Testing:**
   * Individual classes and methods, such as Patient.java, Doctor.java, and Appointment.java, were tested independently.
   * Focus was on verifying business logic, object creation, and method outputs using JUnit.
2. **Integration Testing:**
   * Modules were combined (e.g., Java Swing GUI with MySQL backend via JDBC) and tested for smooth communication and data exchange.
   * Checked how components like registration, scheduling, and billing work together.
3. **System Testing:**
   * The fully integrated system was tested as a complete unit.
   * Validated end-to-end scenarios, module interactions, data consistency, and UI responsiveness.
4. **User Acceptance Testing (UAT):**
   * Conducted from the perspective of different users—Admin, Doctor, and Patient.
   * Ensured that the system is intuitive, meets functional requirements, and provides a seamless user experience.
   * Feedback from test users was incorporated to fix minor usability issues.

Each level of testing played a critical role in delivering a reliable and user-friendly Hospital Management System.

# 6.3 TESTING APPROACH:

# The Black Box Testing approach was used to verify system functionality based on user requirements, without checking internal code. Test cases focused on input/output behavior and user interactions.

# Sample Test Cases:

# Login Validation: Checked role-based access (admin, doctor, etc.).

# Appointment Booking: Verified time slot availability and database update.

# Patient Registration: Ensured correct saving and retrieval of patient details.

# Billing Module: Validated billing calculations and receipt generation.

# Error Handling:

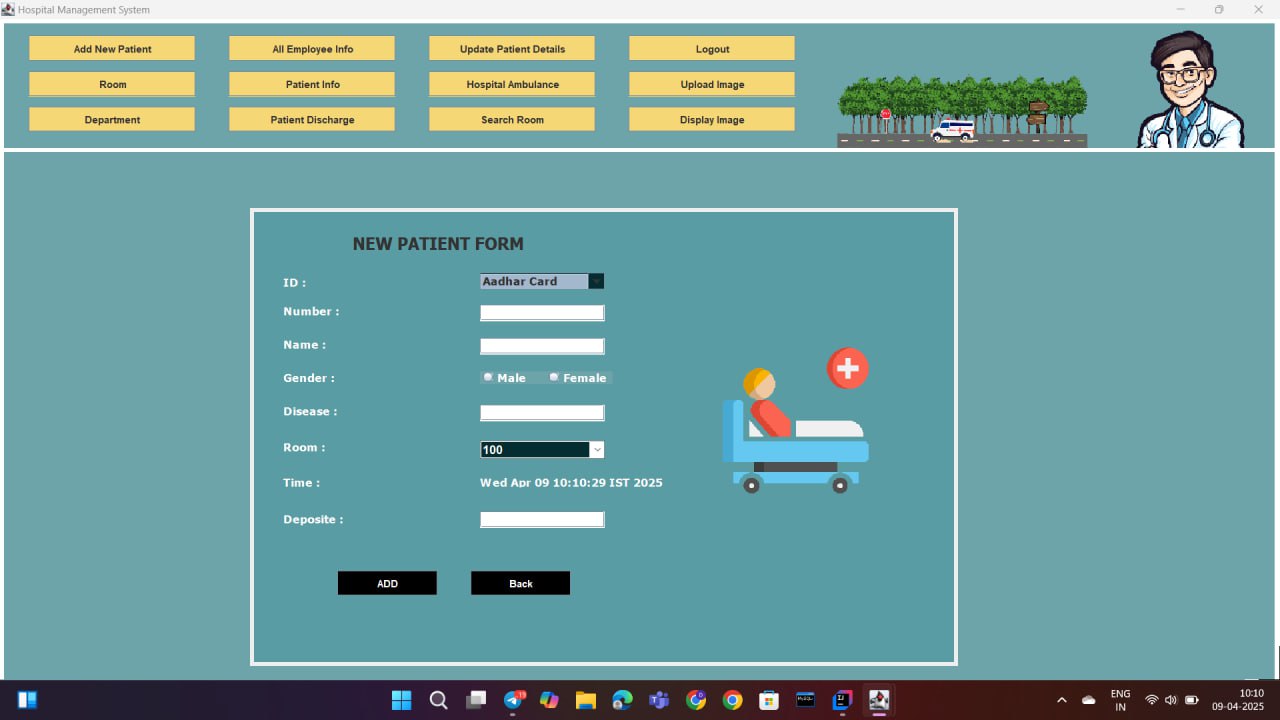
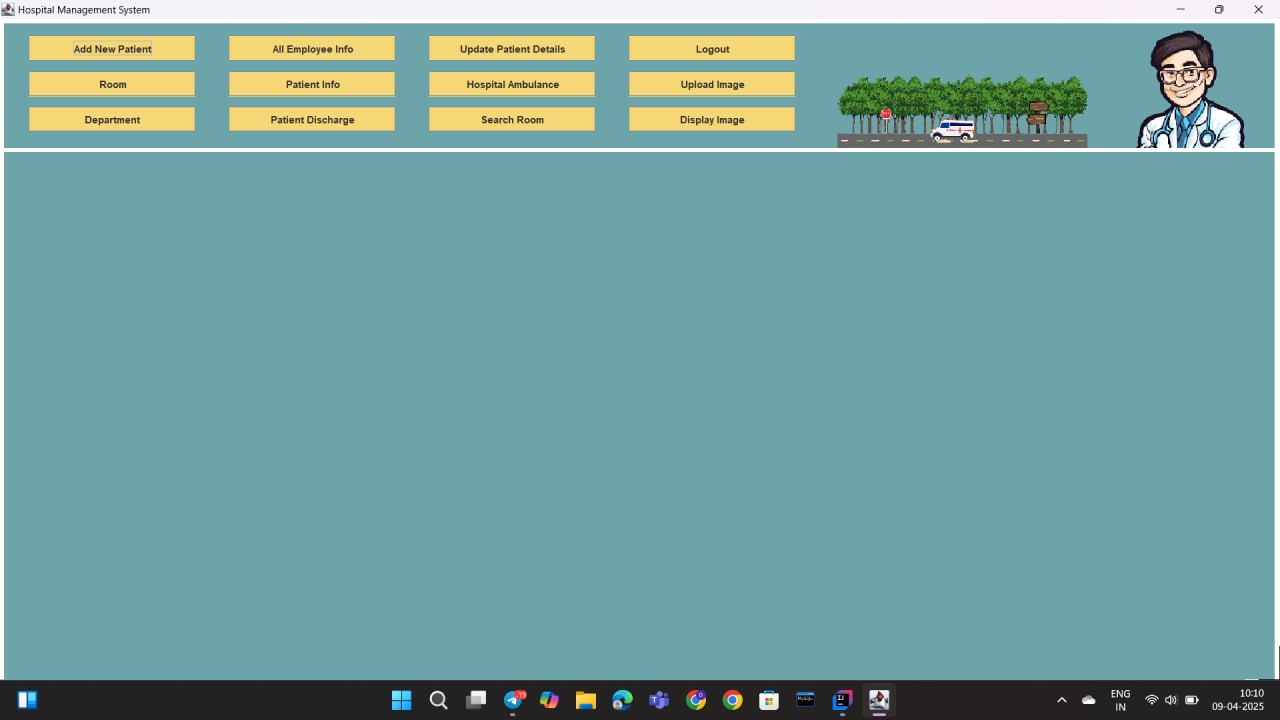
# Invalid input formats (e.g., date, phone).

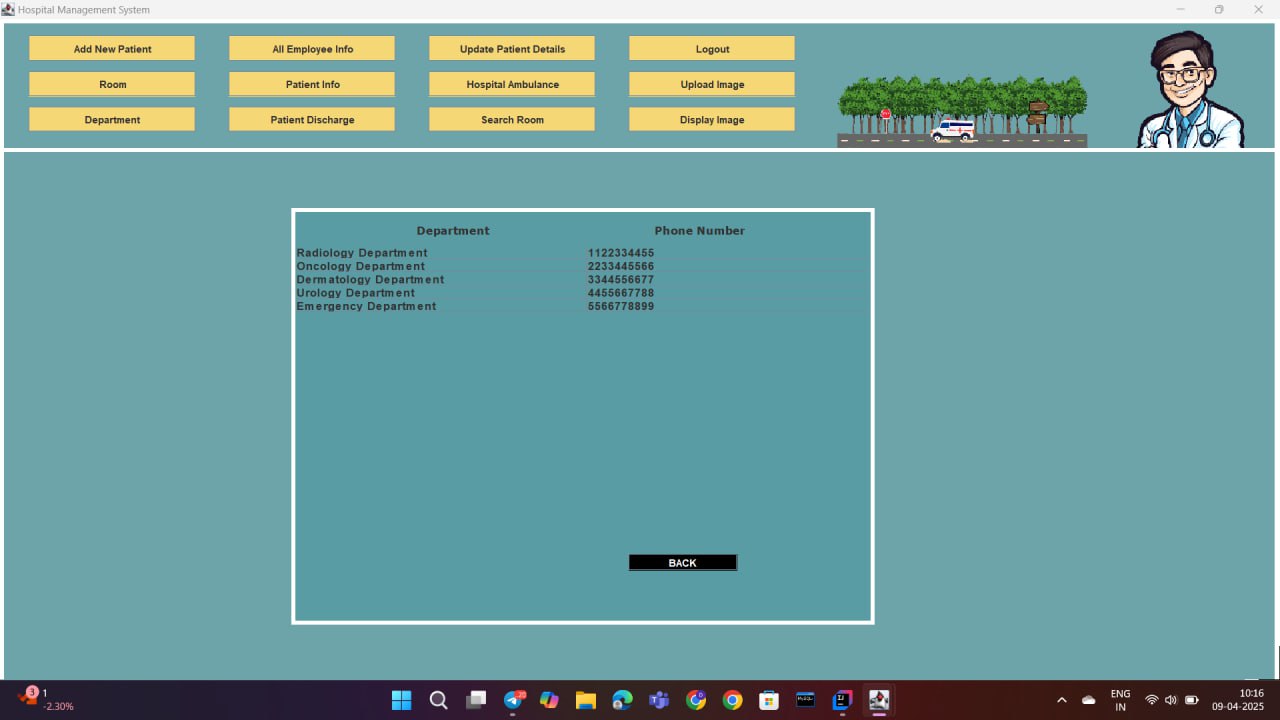
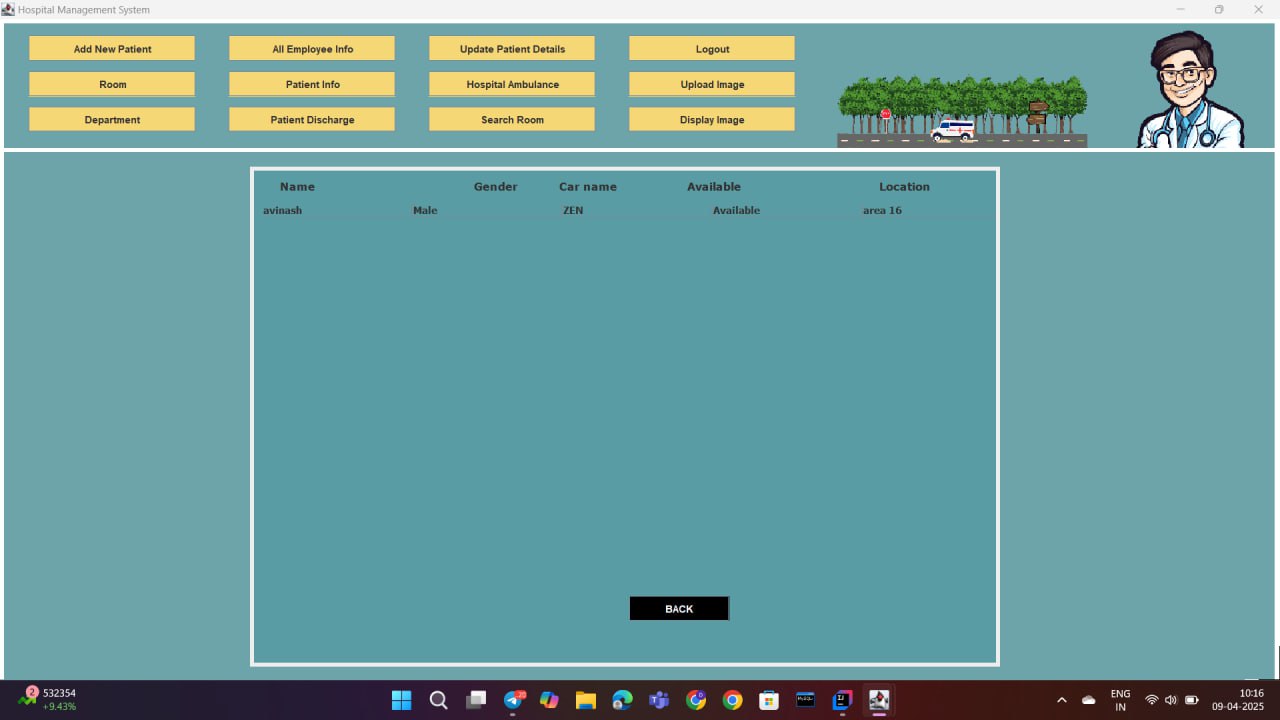
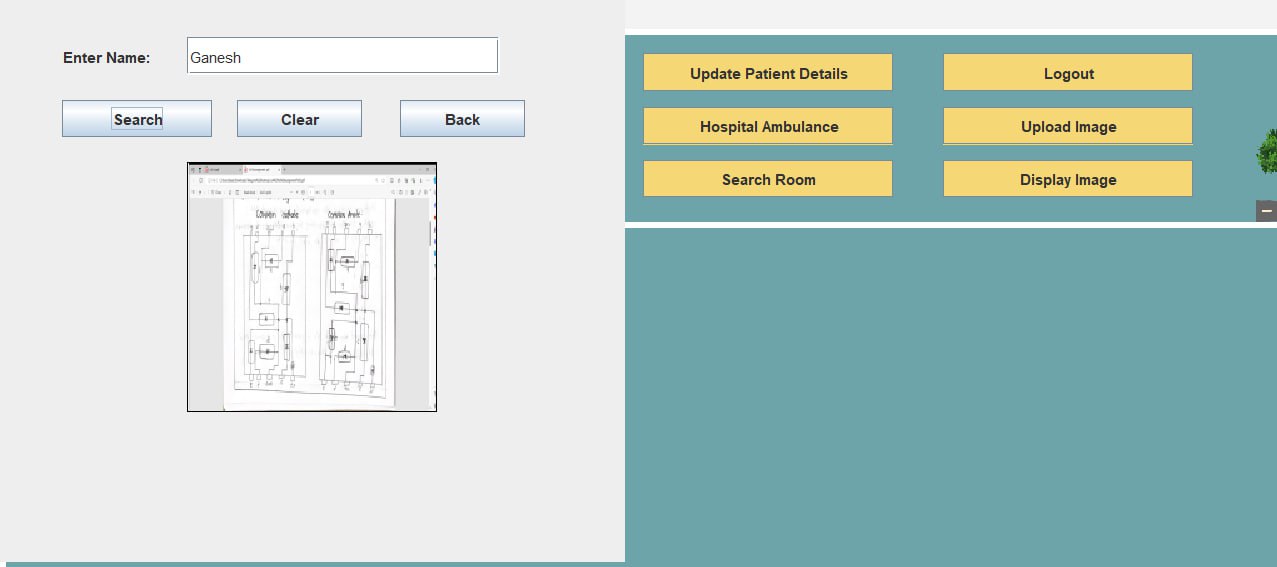
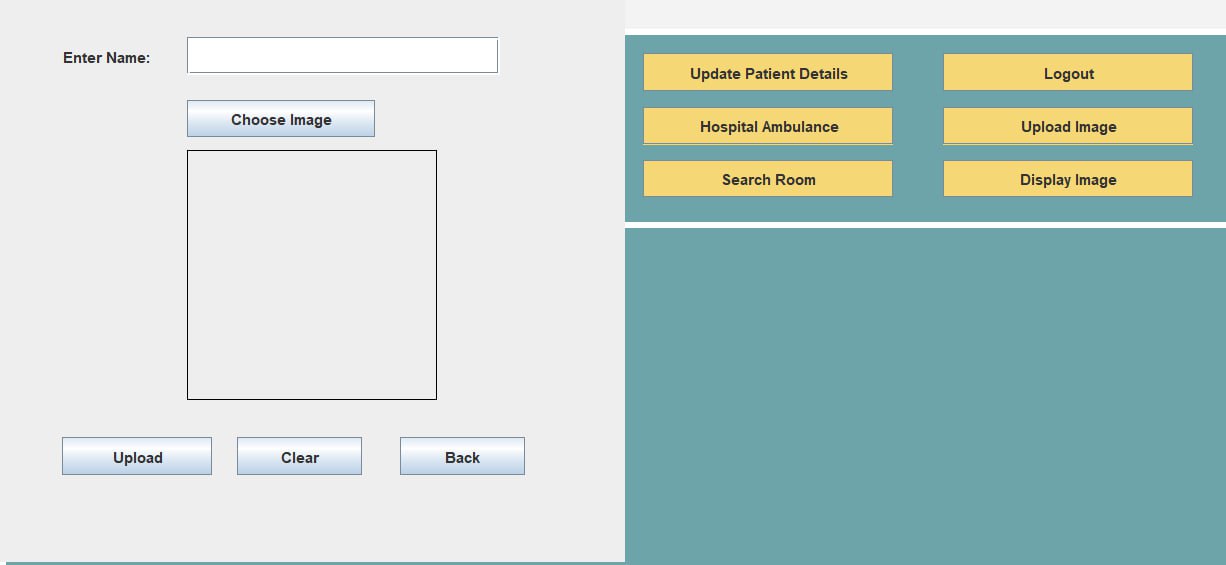
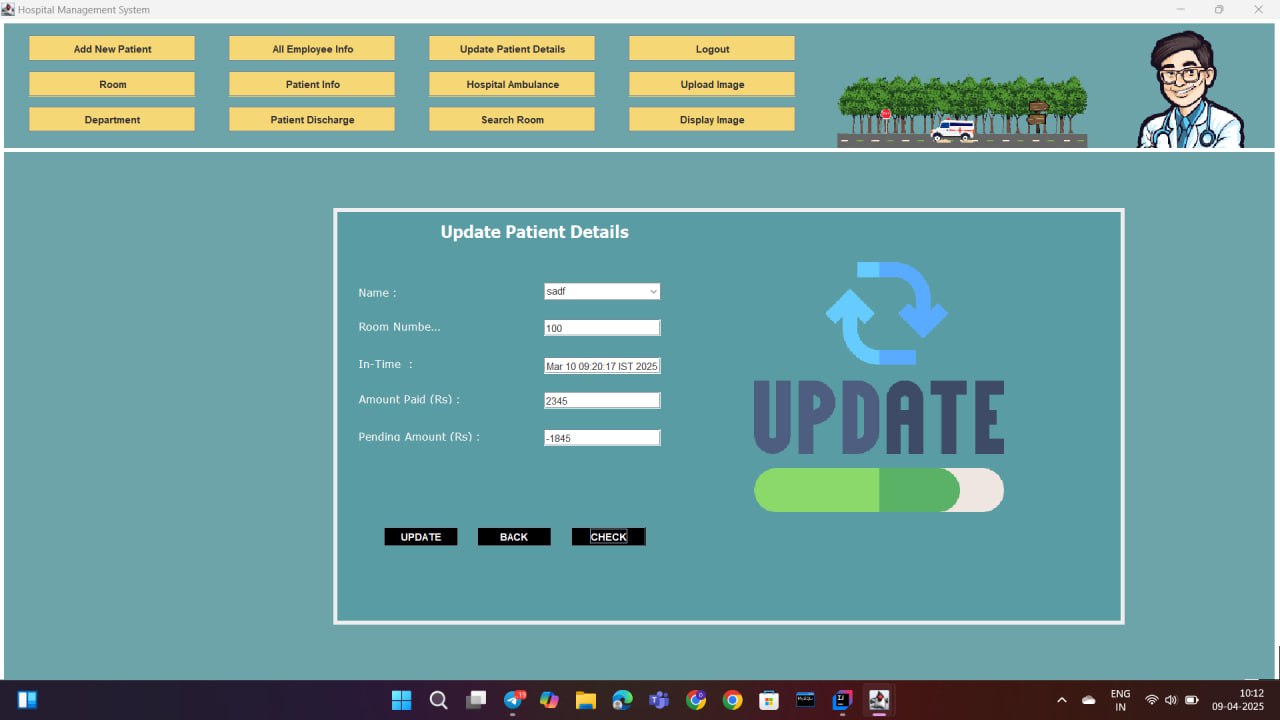
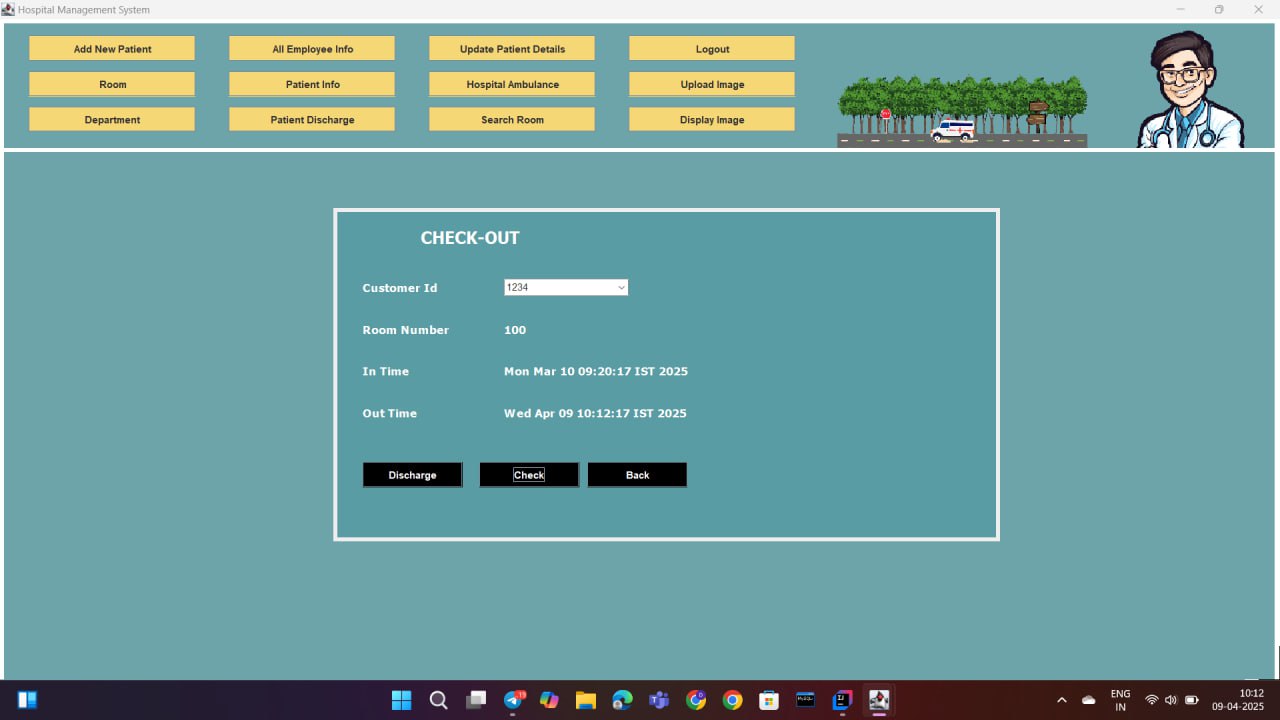
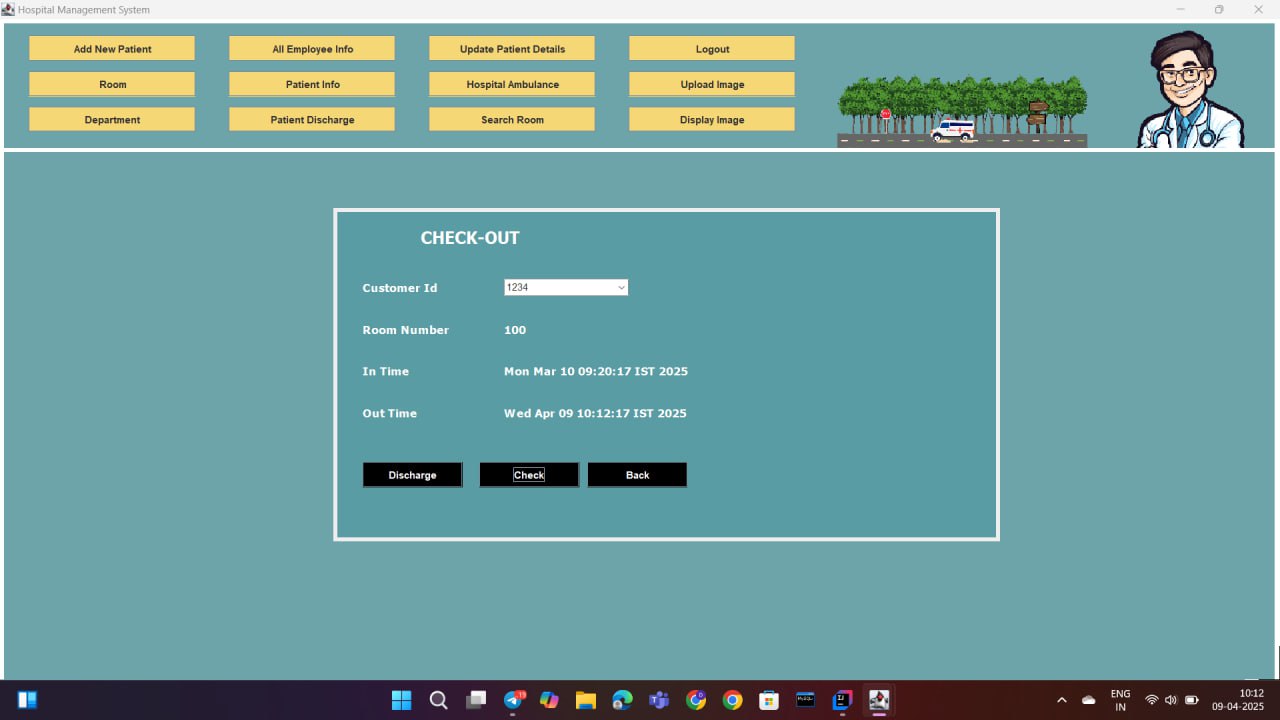
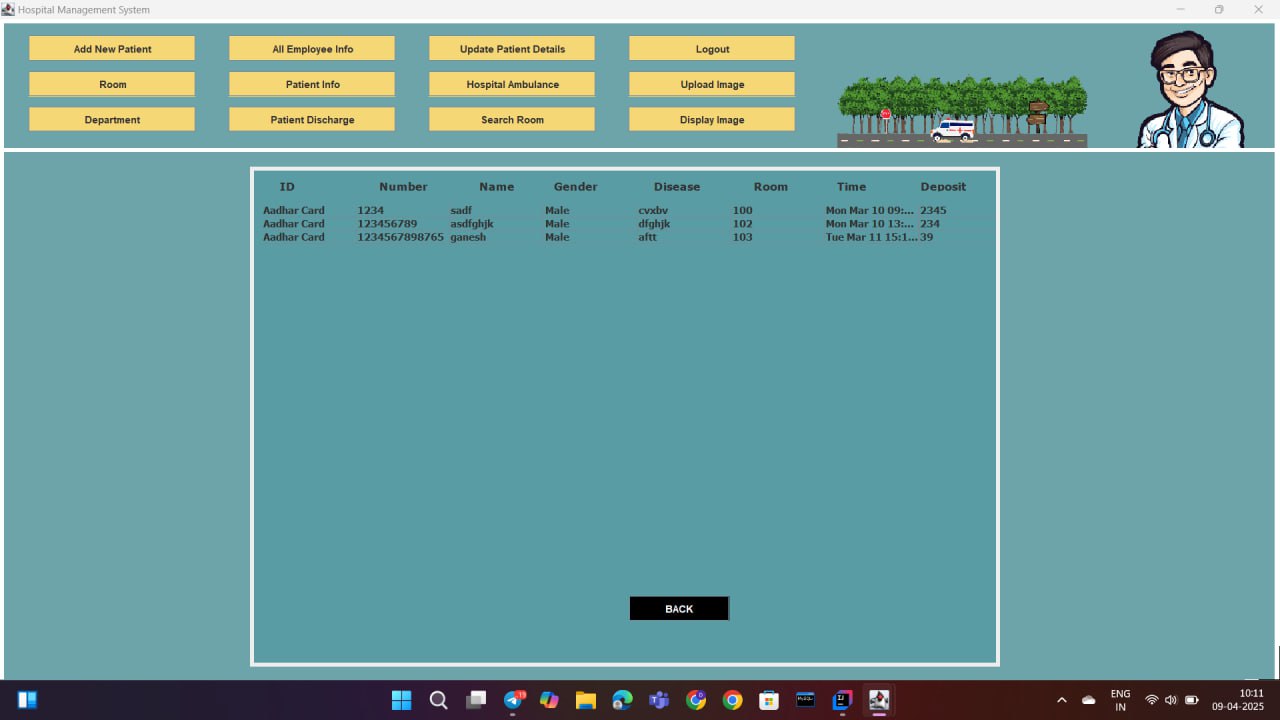
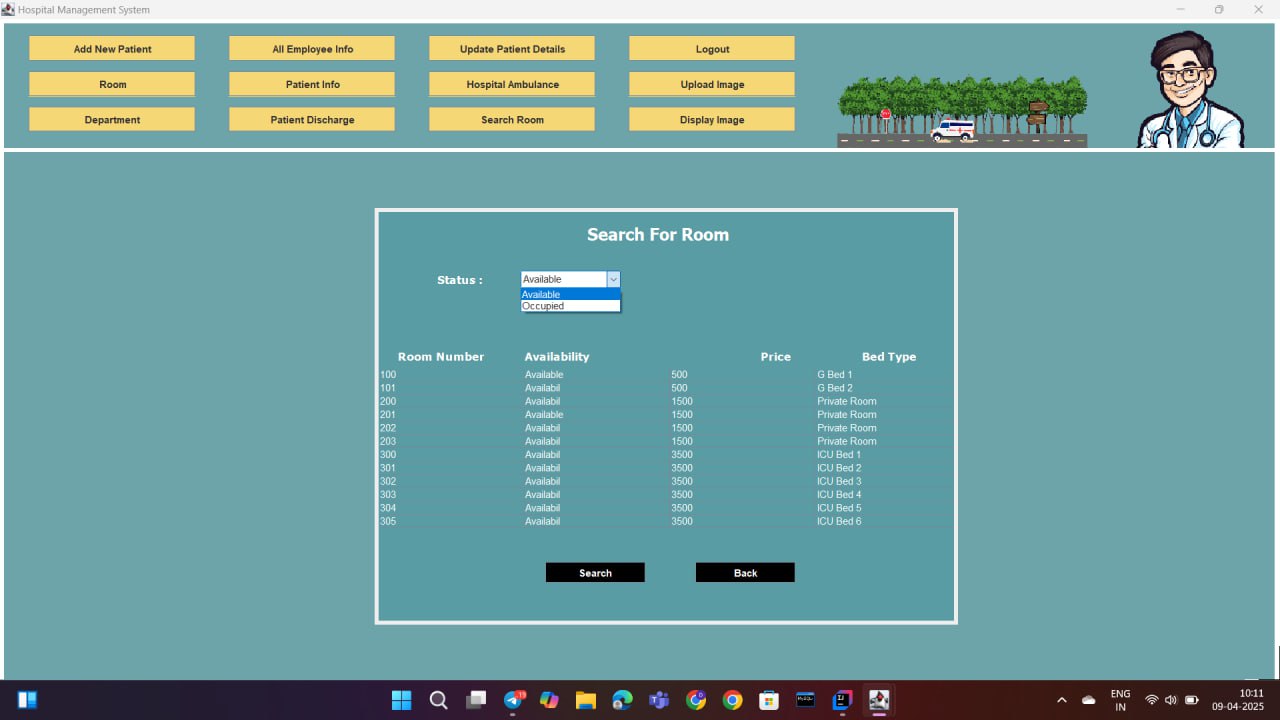
# Duplicate entries (e.g., patient ID).

# Simulated database connection failures.

# This approach helped ensure functional reliability and user-friendly behavior.

# 7. SCREEN SHOTS





**8. CONCLUSION AND FUTURE SCOPE**

**8.1 CONCLUSION**

The **Hospital Management System** was successfully developed using **Java Swing (JFrames)** for the GUI, **MySQL** for backend data storage, and implemented with an **Object-Oriented Programming (OOP)** approach in **IntelliJ IDEA**. This system simplifies hospital operations such as patient registration, doctor scheduling, appointment booking, and billing. The modular structure ensures maintainability, and the interface is user-friendly for all types of users like admins, doctors, and receptionists.

The project showcases how core OOP concepts such as inheritance, encapsulation, abstraction, and polymorphism can be effectively used in building a real-world application. System testing confirmed that the application functions correctly and meets the user requirements.

# 8.2 FUTURE SCOPE:

# While the current system fulfills basic hospital management needs, it can be extended in the future in the following ways:

# Cloud Integration: Storing patient records and reports securely on the cloud for accessibility and backup.

# Mobile Application: Developing Android/iOS apps for real-time patient-doctor interaction.

# E-Prescription Module: Auto-generating prescriptions which can be sent digitally to pharmacies.

# SMS/Email Notifications: Sending appointment and billing notifications to patients.

# Data Analytics: Generating reports on patient trends, disease patterns, and doctor performance for better decision-making.

# 

**REFERENCES AND BIBILOGRAPHY**

1. Java Complete Reference – Herbert Schildt
2. MySQL Documentation – <https://dev.mysql.com/doc/>
3. Oracle Java Tutorials – <https://docs.oracle.com/javase/tutorial/>
4. IntelliJ IDEA Documentation – <https://www.jetbrains.com/idea/docs/>
5. Stack Overflow – <https://stackoverflow.com/>
6. W3Schools – <https://www.w3schools.com/>
7. GitHub Java Swing Projects for Reference and UI ideas