

**HOTEL MANAGEMENT SYSTEM**

**LOGIN and SIGNUP PAGE**

**A MINI PROJECT REPORT**

**Submitted by**

**DINESH KARTHIK K 231501042**

**In partial fulfilment for the award of the degree of**

**BACHELOR OF TECHNOLOGY IN**

**ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING**

**RAJALAKSHMI ENGINEERING COLLEGE**

**(AUTONOMOUS) THANDALAM**

**CHENNAI-602105**

**2024 - 2025**



**BONAFIDE CERTIFICATE**

Certified that this project report **“HOTEL MANAGEMENT SYSTEM**” is the Bonafide work of **“DINESH KARTHIK K (231501042).”** who carried out the project work under my supervision.

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

**SIGNATURE**

**Mrs. Manju S,**

**Assistant Professor (SS)**

**AIML,**

**Rajalakshmi Engineering**

**College (Autonomous),**

**Thandalam, Chennai - 602 105**

**INTERNAL EXAMINER EXTERNAL EXAMINER**

**ABSTRACT**

The **Hotel Management Login and Signup System** is a Java-based application designed to provide secure access to a centralized platform for managing resources within educational institutions. Developed using NetBeans and XAMPP, this system enables authenticated users to log in and access resource management functionalities, while ensuring that only authorized personnel can interact with sensitive features. By utilizing Java Database Connectivity (JDBC) for seamless interaction with a MySQL database, the system securely stores user credentials, including hashed passwords, to protect user data and maintain the integrity of the login process.

This project is an educational tool for developers, offering hands-on experience in core areas of software development, such as user authentication, data validation, GUI design, and database management. The NetBeans IDE facilitates intuitive GUI creation through Java Swing components, resulting in a user-friendly interface that enhances the user experience. Security practices are prioritized throughout the project, emphasizing encryption, error handling, and secure database interactions to safeguard sensitive information.

With practical applications in resource allocation and user management, the **Hotel Management Login and Signup System** highlights best practices in secure software development, while laying a foundation for more advanced features such as multi-factor authentication and role-based access control. This project not only equips developers with essential programming skills but also underscores the importance of security and data protection in today’s digital landscape.

**TABLE OF CONTENTS**

**1.INTRODUCTION ---------------------------------------------------------------------5**

1.1. INTRODUCTION TO PROJECT

1.2. OBJECTIVES

**2.STUDY OF TECHNOLOGIES -----------------------------------------------------7**

2.1. SOFTWARE DESCRIPTION AND FEATURES

2.2. LANGUAGES

**3.REQUIREMENTS AND ANALYSIS ---------------------------------------------10**

**3.1 REQUIREMENTS SPECIFICATION**

**3.2 Functional Requirements**

**3.3 Non-Functional Requirements**

1. **PROGRAM CODE ------------------------------------------------------------13**
2. **RESULTS AND DISCUSSION -----------------------------------------------35**
3. **CONCLUSION -------------------------------------------------------------------38**
4. **REFERENCES -------------------------------------------------------------------39**

**1.INTRODUCTION**

### 1.1 INTRODUCTION TO PROJECT

The **Hotel Management System** is a software application designed to automate and streamline various aspects of hotel operations. It facilitates the management of hotel rooms, customer reservations, billing, and payments. This project is built using Java, a powerful and versatile programming language known for its platform independence and robust performance. The system offers features such as room booking, customer check-in/check-out, generating invoices, room status management, and reports. The objective is to simplify hotel management tasks, improve efficiency, and enhance the customer experiences.

* 1. **OBJECTIVES:**
* To automate the process of room booking, reservation, and check-in/check-out.
* To manage and update room availability in real time.
* To track customer information and store booking history.
* To generate invoices and handle payments.
* To provide a simple and intuitive user interface for both hotel staff and customers.
* To store data securely in a database and ensure efficient data retrieval.
* To provide a reporting feature for hotel admins or managers to track business performance (e.g., booking frequency, revenue, etc.).

#### **1.3 Scope of the Project**

The scope of the **Hotel Management System** covers the following key functionalities:

* **Room Management**: Manage room categories, availability, and pricing.
* **Reservation System**: Customers can book rooms, view availability, and cancel bookings if necessary.
* **Customer Management**: Record and update customer details, including personal information and booking history.
* **Billing and Payments**: Generate invoices for customers based on room charges and services availed, and process payments.
* **Check-in/Check-out**: Handle customer check-in and check-out processes, including room allocation and status updates.
* **Admin Panel**: Admins can access reports, manage bookings, update room statuses, and oversee operations.

The project does not cover complex features like hotel management across multiple locations, mobile application integration, or advanced AI-based recommendations for customers. It focuses on the core operations of a single-location hotel.

#### **1.4 Technologies Used**

This project leverages the following technologies:

* **Java**: The core programming language used to develop the application. Java’s object-oriented nature and strong integration capabilities make it ideal for building scalable applications.
* **Database (MySQL)**: MySQL is used for managing and storing data such as customer information, room bookings, payments, and invoices. A relational database ensures that the data is structured and can be easily queried.
* **IDE (Eclipse or IntelliJ)**: Integrated Development Environments (IDEs) such as Eclipse or IntelliJ IDEA are used to write, debug, and compile the Java code.
* **JDBC (Java Database Connectivity)**: JDBC is used to connect the Java application to the MySQL database, allowing for CRUD (Create, Read, Update, Delete) operations on data.
* **Swing (for GUI)**: Java Swing provides a graphical user interface (GUI) framework, which is used to design the front-end interface of the hotel management system. It includes elements like buttons, text fields, and tables to interact with the user.
* **SQL**: SQL is used to create and manage the database structure and perform queries to interact with the stored data.

#### **1.5 Problem Statement**

The hotel industry involves numerous repetitive tasks such as managing room bookings, customer details, payment processing, and generating invoices. These tasks, if done manually, can lead to inefficiency, errors, and a poor customer experience. The goal of this project is to build a system that automates these processes, reduces human error, and ensures smoother operations, ultimately leading to improved customer service and business management.

#### **1.6 Benefits**

* **Efficiency**: Automates time-consuming manual tasks such as booking reservations, checking-in customers, and generating invoices.
* **Accuracy**: Reduces human errors in room allocation, billing, and customer management.
* **Convenience**: Provides an easy-to-use interface for both hotel staff and customers, improving the overall experience.
* **Scalability**: The system can be extended to support more features, such as multiple hotel locations or more complex reporting tools.

### 

### 

**2. STUDY OF TECHNOLOGIES**

**2.1 SOFTWARE DESCRIPTION AND FEATURES:** The **Hotel Management System (HMS)** is a comprehensive software solution designed to handle and automate various operations of a hotel. It allows hotel staff and management to efficiently manage customer reservations, room bookings, billing, payments, and reporting. The system leverages a user-friendly interface that simplifies hotel operations, improving both administrative efficiency and customer satisfaction.

Built using **Java**, the system utilizes a **Graphical User Interface (GUI)** with **Swing** for user interactions, and **MySQL** as the backend database to securely store all hotel-related data. The system can be used by both front-end hotel staff (e.g., receptionists) and administrative users (e.g., managers), with different levels of access and functionalities depending on the user role.

The primary aim of the software is to replace traditional manual systems and reduce human error, allowing hotel operations to run smoothly and efficiently. It ensures real-time updates of room availability, smooth check-in and check-out processes, accurate billing, and instant reports for decision-making.

**Features**

1. **User Authentication and Role-Based Access**
   * **Login System**: Different types of users (e.g., staff, admin) have separate logins.
   * **Role-Based Permissions**: Admins have access to advanced features like generating reports, managing room details, and viewing all bookings, while staff can only handle basic operations like booking and check-in.
2. **Room Management**
   * **Room Availability**: Displays the availability of rooms in real-time (booked or available).
   * **Room Categories**: Different categories like standard, deluxe, suite, etc., each with specific pricing.
   * **Room Status Updates**: Admin can update room statuses (e.g., clean, under maintenance, booked, available).
3. **Reservation System**
   * **Room Booking**: Users can book rooms by selecting available rooms, entering customer details, and specifying the check-in/check-out dates.
   * **Real-Time Availability**: The system automatically checks room availability based on the selected dates.
   * **Booking Cancellation**: Users can cancel bookings if needed and update room availability accordingly.
4. **Customer Management**
   * **Customer Registration**: Staff can register customer details, including personal information, contact details, and payment information.
   * **Customer History**: The system tracks booking history for each customer, allowing for easy retrieval of past data.
5. **Check-In/Check-Out**
   * **Customer Check-In**: Upon arrival, customers can be checked into a room, and the system updates the room’s status to "Occupied."
   * **Check-Out Process**: On departure, the customer is checked out, the room status is updated, and a final invoice is generated.
   * **Room Reallocation**: In case of early check-out or cancellations, rooms are updated and made available for new bookings.
6. **Billing and Invoice Generation**
   * **Invoice Creation**: Automatically generates invoices based on room charges, additional services (like meals, spa), taxes, and discounts.
   * **Payment Processing**: Facilitates different payment methods (e.g., cash, card, online) and tracks the payment status for each booking.
   * **Payment History**: Users can view and track payment history for customers.
7. **Reporting and Analytics**
   * **Booking Reports**: Admin can view reports showing booking trends, occupancy rates, and the total number of reservations.
   * **Revenue Reports**: Detailed revenue generation reports, broken down by room category, services, and periods.
   * **Customer Reports**: Insights into customer booking patterns, frequent stays, and payment history.
   * **Daily/Monthly Statistics**: Shows the number of check-ins, check-outs, and current room occupancy.
8. **Database Integration and Data Management**
   * **Data Storage**: All customer, room, and payment details are securely stored in a MySQL database.
   * **Data Retrieval**: The system performs fast and efficient queries to fetch and update data as needed, using JDBC for database connectivity.
   * **Backup and Restore**: Admins can back up the database to ensure data security and restore it in case of failures.
9. **User-Friendly Graphical Interface**
   * **Graphical Interface (GUI)**: Built using Java Swing, providing an intuitive and simple interface for hotel staff and admins.
   * **Navigation**: Easy-to-use buttons, dropdowns, and forms for smooth navigation through the application.
   * **Real-Time Updates**: The system immediately reflects changes made by users, such as booking a room or updating a status.
10. **Security and Data Protection**
    * **Data Encryption**: Sensitive customer and payment data is securely stored, and basic encryption methods are applied to protect personal details.
    * **Session Management**: Ensures that user sessions are managed securely, and users must log in again after a timeout period.

2.2 Programming Languages

1. **Java**
   * **Core Programming Language**: Java is used for building the application's logic, handling user input, managing database interactions, and implementing secure authentication processes.
   * **Java Swing for GUI**: Java Swing is utilized to create the graphical user interface (GUI), enabling an interactive and user-friendly experience.
2. **SQL (Structured Query Language)**
   * **Database Management**: SQL is used for managing the MySQL database, including creating tables, inserting user data, retrieving records for login authentication, and performing other database operations.
3. **Java Database Connectivity (JDBC)**
   * **Database Connectivity**: JDBC enables seamless communication between the Java application and the MySQL database, allowing the system to perform SQL queries and updates within the Java environment.
4. **XML (Extensible Markup Language)**
   * **Configuration Files**: XML may be used in certain configurations, such as setting up JDBC connections or configuring database properties, to define parameters that the Java application can reference.

**3. REQUIREMENTS AND ANALYSIS**

**3.1 REQUIREMENTS SPECIFICATION**

The **Requirement Specification** document provides a detailed description of the functional and non-functional requirements of the **Hotel Management System (HMS)**. It is a blueprint for the system that defines the expected behavior of the software. The requirements specification serves as a communication tool between stakeholders, such as developers, users, and project managers, and guides the development and testing of the system.

**3.2 Functional Requirements**

### ****1. Functional Requirements****

Functional requirements describe the specific behavior of the system and the tasks it must perform. These include interactions with users, data processing, and system responses to various inputs. For a **Hotel Management System (HMS)**, the following functional requirements must be considered.

#### **1.1 User Authentication and Authorization**

* **Login and Logout**: Users (admin, receptionist, staff) must be able to log in and out using a secure username and password.
* **Role-Based Access Control**:
  + **Admin** has full access to all system features, including managing bookings, customers, reports, and system settings.
  + **Staff/Receptionists** have limited access, focusing on daily operations like booking management, check-ins, check-outs, and customer interactions.

#### **1.2 Room Management**

* **Room Categories**: The system should allow for multiple room types (e.g., standard, deluxe, suite) with different prices and features.
* **Room Availability**: The system should check and update room availability based on customer bookings and show the available rooms.
* **Room Status Management**: Admin and staff should be able to update room status (e.g., available, booked, occupied, under maintenance).
* **Room Assignment**: Receptionists should be able to assign rooms to customers when booking or checking in.

#### **1.3 Reservation and Booking Management**

* **Create New Bookings**: The system should allow receptionists or customers (through an online interface) to create new reservations.
  + Enter booking details, such as room type, customer details, check-in/check-out dates, and special requests.
* **Modify Bookings**: Users should be able to modify booking details, including extending stays, changing room types, or modifying check-in/check-out dates.
* **Cancel Bookings**: Receptionists or customers should be able to cancel bookings. Upon cancellation, room availability must be updated.

#### **1.4 Customer Management**

* **Customer Registration**: Receptionists should be able to add new customer details (e.g., name, address, phone, email, payment information).
* **Customer History**: The system should store and display a customer’s past bookings and preferences.
* **Customer Search**: The system should allow staff to search for customer details using filters such as name, phone number, or booking ID.

#### **1.5 Check-In and Check-Out**

* **Check-In Process**: Receptionists should be able to check in a customer by verifying their booking, assigning a room, and providing key information. The room status should be updated to "Occupied."
* **Check-Out Process**: Upon check-out, the system should calculate the final amount (including room charges and additional services) and generate an invoice. The room status should be updated to "Available."
* **Early Check-Out**: Receptionists should be able to process early check-outs and update the system accordingly.

#### **1.6 Billing and Payment Management**

* **Generate Invoices**: The system should automatically generate invoices for customers at check-out, showing room charges, additional services, taxes, and total amount.
* **Track Payments**: The system should allow the tracking of payments (partial or full) and show the balance remaining on each booking.
* **Payment Methods**: The system should support multiple payment methods, such as cash, credit card, and online payments.
* **Refunds and Adjustments**: In case of cancellations or early check-out, the system should process refunds or adjustments to the invoice.

#### **1.7 Reporting**

* **Booking Reports**: Admin should be able to generate reports on bookings, such as total bookings per day, week, or month, and booking history.
* **Revenue Reports**: Admin can generate reports showing revenue from room bookings, services (e.g., spa, meals), and payment methods.
* **Occupancy Reports**: The system should generate reports showing the occupancy rate of rooms over different periods.
* **Customer Reports**: Admin can generate reports detailing customer visits, payment history, and customer satisfaction.

#### **1.8 Database Management**

* **Data Storage**: The system must store essential data (room availability, customer details, booking history) in a secure, relational database (e.g., MySQL).
* **Data Integrity**: The system should ensure that data is accurate, and there is no data duplication, especially with bookings and customer records.
* **Backup and Restore**: Admins should have the ability to back up the database and restore data in case of a system failure or data corruption.

### ****2. Non-Functional Requirements****

Non-functional requirements describe the system's performance characteristics, quality attributes, and operational constraints. These include aspects such as reliability, security, scalability, and usability.

#### **2.1 Performance**

* **Response Time**: The system should respond to user requests (e.g., booking creation, invoice generation) within 3-5 seconds, ensuring a smooth user experience.
* **Throughput**: The system should be able to handle multiple concurrent users without performance degradation (e.g., during peak times, such as holidays or weekends).
* **Scalability**: The system should be able to scale to accommodate additional users and data (e.g., when the hotel expands or during high-demand periods).

#### **2.2 Usability**

* **User Interface (UI)**: The user interface must be intuitive, with a clean design and simple navigation to ensure ease of use for hotel staff.
  + **Desktop or Web Interface**: For desktop applications, Java Swing or JavaFX will be used, and for web interfaces, HTML, CSS, and JavaScript will be used.
* **Accessibility**: The system should be easy to use by people with various levels of computer literacy, with clear labels, instructions, and error messages.
* **Multi-Language Support** (Optional): The system could support multiple languages for international users (e.g., English, Spanish, French).

#### **2.3 Security**

* **Authentication and Authorization**: The system must ensure that users are authenticated using a secure login process, and that role-based access control is implemented.
* **Data Encryption**: Sensitive customer information, such as payment data, should be encrypted both at rest (in the database) and in transit (when sent over the network, using SSL/TLS).
* **Session Management**: The system should automatically log out users after a set period of inactivity (e.g., 15 minutes) to prevent unauthorized access.
* **Data Protection**: Personal data (e.g., name, address, payment details) must be securely stored and comply with data privacy laws, such as GDPR.

#### **2.4 Availability and Reliability**

* **Uptime**: The system should be available 99.9% of the time, excluding scheduled maintenance periods. This means no more than 8 hours of downtime per year.
* **Fault Tolerance**: The system should be designed to gracefully handle errors and failures, such as a sudden loss of database connectivity, without crashing or losing data.
* **Backup and Recovery**: Regular backups should be performed, and recovery procedures should be in place to restore data in case of system failure.

#### **2.5 Maintainability**

* **Code Modularity**: The codebase should be modular, meaning that features (e.g., room management, booking, payment) are separated into distinct components that can be independently updated or replaced.
* **Documentation**: Comprehensive user documentation (for hotel staff and admins) and technical documentation (for developers) should be provided.
* **Error Logging**: The system should maintain detailed logs of errors and events for debugging purposes and system monitoring.

#### **2.6 Compatibility**

* **Cross-Platform Compatibility**: The system should be compatible with multiple operating systems, including Windows, Linux, and macOS.
* **Browser Compatibility** (for Web Systems): The system should be compatible with modern web browsers (e.g., Chrome, Firefox, Safari, Edge) for web-based access.
* **Mobile Compatibility** (Optional): If a mobile version is developed, the system should be responsive and usable on smartphones and tablets (for reservations, check-in/check-out).

#### **2.7 Compliance**

* **Legal Compliance**: The system must comply with relevant local and international laws and regulations, including:
  + Data protection laws (e.g., GDPR).
  + Payment gateway security standards (e.g., PCI-DSS compliance).
  + Tax and invoicing regulations.

**4. PROGRAM CODE**

**LOGIN CODE:**

**import javax.swing.\*;**

**import java.awt.\*;**

**import java.awt.event.ActionEvent;**

**import java.awt.event.ActionListener;**

**public class LoginPage extends JFrame {**

**// Create components**

**private JLabel lblUsername, lblPassword, lblMessage;**

**private JTextField txtUsername;**

**private JPasswordField txtPassword;**

**private JButton btnLogin, btnSignUp;**

**public LoginPage() {**

**// Set up the frame**

**setTitle("Hotel Management System - Login");**

**setSize(400, 350); // Initial size**

**setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);**

**setLocationRelativeTo(null); // Center the window on the screen**

**// Set a maximum size (limits resizing to a certain size)**

**setMaximumSize(new Dimension(400, 350)); // Limit the size if resized**

**setResizable(true); // Allow resizing, but within limits**

**// Initialize components**

**lblUsername = new JLabel("Username:");**

**lblPassword = new JLabel("Password:");**

**txtUsername = new JTextField(20);**

**txtPassword = new JPasswordField(20);**

**btnLogin = new JButton("Login");**

**btnSignUp = new JButton("Sign Up");**

**lblMessage = new JLabel("");**

**// Customize fonts and colors**

**Font labelFont = new Font("Arial", Font.PLAIN, 14);**

**Font buttonFont = new Font("Arial", Font.BOLD, 14);**

**lblUsername.setFont(labelFont);**

**lblPassword.setFont(labelFont);**

**btnLogin.setFont(buttonFont);**

**btnSignUp.setFont(buttonFont);**

**btnLogin.setBackground(new Color(34, 139, 34)); // Green button**

**btnLogin.setForeground(Color.WHITE);**

**btnSignUp.setBackground(new Color(70, 130, 180)); // Blue button**

**btnSignUp.setForeground(Color.WHITE);**

**// Set up the layout manager**

**setLayout(new BorderLayout(10, 10));**

**// Create a panel for username and password fields**

**JPanel centerPanel = new JPanel();**

**centerPanel.setLayout(new GridLayout(3, 2, 10, 10));**

**centerPanel.setBorder(BorderFactory.createEmptyBorder(10, 10, 10, 10));**

**// Add components to the center panel**

**centerPanel.add(lblUsername);**

**centerPanel.add(txtUsername);**

**centerPanel.add(lblPassword);**

**centerPanel.add(txtPassword);**

**centerPanel.add(new JLabel()); // Empty label for spacing**

**centerPanel.add(btnLogin);**

**// Add the message label**

**lblMessage.setFont(new Font("Arial", Font.ITALIC, 12));**

**lblMessage.setHorizontalAlignment(SwingConstants.CENTER);**

**// Add the "Sign Up" button below login button**

**JPanel bottomPanel = new JPanel();**

**bottomPanel.add(btnSignUp);**

**// Add all panels to the frame**

**add(centerPanel, BorderLayout.CENTER);**

**add(lblMessage, BorderLayout.SOUTH);**

**add(bottomPanel, BorderLayout.PAGE\_END);**

**// Set up button action listeners**

**btnLogin.addActionListener(new ActionListener() {**

**@Override**

**public void actionPerformed(ActionEvent e) {**

**handleLogin();**

**}**

**});**

**// "Sign Up" button action listener**

**btnSignUp.addActionListener(new ActionListener() {**

**@Override**

**public void actionPerformed(ActionEvent e) {**

**openSignUpPage();**

**}**

**});**

**}**

**// Handle login action**

**private void handleLogin() {**

**String username = txtUsername.getText();**

**char[] password = txtPassword.getPassword();**

**String passwordStr = new String(password); // Convert char array to string**

**// Simple validation (you can extend this to connect to a database)**

**if (username.equals("Dinesh") && passwordStr.equals("dinesh123")) {**

**lblMessage.setText("Login successful!");**

**lblMessage.setForeground(Color.GREEN);**

**JOptionPane.showMessageDialog(this, "Hi Dinesh Welcome to the Hotel Management System");**

**dispose(); // Close login window**

**} else {**

**lblMessage.setText("Invalid username or password.");**

**lblMessage.setForeground(Color.RED);**

**}**

**}**

**// Open SignUp page when Sign Up button is clicked**

**private void openSignUpPage() {**

**dispose(); // Close the login page**

**SignUpPage signUpPage = new SignUpPage();**

**signUpPage.setVisible(true); // Show the signup page**

**}**

**public static void main(String[] args) {**

**SwingUtilities.invokeLater(new Runnable() {**

**@Override**

**public void run() {**

**LoginPage loginPage = new LoginPage();**

**loginPage.setVisible(true);**

**}**

**});**

**}**

**}**

**SIGN UP CODE:**

**import javax.swing.\*;**

**import java.awt.\*;**

**import java.awt.event.ActionEvent;**

**import java.awt.event.ActionListener;**

**public class SignUpPage extends JFrame {**

**// Create components**

**private JLabel lblUsername, lblPassword, lblConfirmPassword, lblMessage;**

**private JTextField txtUsername;**

**private JPasswordField txtPassword, txtConfirmPassword;**

**private JButton btnSignUp;**

**public SignUpPage() {**

**// Set up the frame**

**setTitle("Hotel Management System - Sign Up");**

**setSize(400, 350);**

**setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);**

**setLocationRelativeTo(null);**

**// Initialize components**

**lblUsername = new JLabel("Username:");**

**lblPassword = new JLabel("Password:");**

**lblConfirmPassword = new JLabel("Confirm Password:");**

**txtUsername = new JTextField(20);**

**txtPassword = new JPasswordField(20);**

**txtConfirmPassword = new JPasswordField(20);**

**btnSignUp = new JButton("Sign Up");**

**lblMessage = new JLabel("");**

**// Customize fonts and colors**

**Font labelFont = new Font("Arial", Font.PLAIN, 14);**

**Font buttonFont = new Font("Arial", Font.BOLD, 14);**

**lblUsername.setFont(labelFont);**

**lblPassword.setFont(labelFont);**

**lblConfirmPassword.setFont(labelFont);**

**btnSignUp.setFont(buttonFont);**

**btnSignUp.setBackground(new Color(34, 139, 34)); // Green button**

**btnSignUp.setForeground(Color.WHITE);**

**// Set up the layout manager**

**setLayout(new BorderLayout(10, 10));**

**// Create a panel for username, password, and confirm password fields**

**JPanel centerPanel = new JPanel();**

**centerPanel.setLayout(new GridLayout(4, 2, 10, 10));**

**centerPanel.setBorder(BorderFactory.createEmptyBorder(10, 10, 10, 10));**

**// Add components to the center panel**

**centerPanel.add(lblUsername);**

**centerPanel.add(txtUsername);**

**centerPanel.add(lblPassword);**

**centerPanel.add(txtPassword);**

**centerPanel.add(lblConfirmPassword);**

**centerPanel.add(txtConfirmPassword);**

**centerPanel.add(new JLabel()); // Empty label for spacing**

**centerPanel.add(btnSignUp);**

**// Add message label**

**lblMessage.setFont(new Font("Arial", Font.ITALIC, 12));**

**lblMessage.setHorizontalAlignment(SwingConstants.CENTER);**

**// Add the panels to the frame**

**add(centerPanel, BorderLayout.CENTER);**

**add(lblMessage, BorderLayout.SOUTH);**

**// Set up button action listener**

**btnSignUp.addActionListener(new ActionListener() {**

**@Override**

**public void actionPerformed(ActionEvent e) {**

**handleSignUp();**

**}**

**});**

**}**

**// Handle sign up action**

**private void handleSignUp() {**

**String username = txtUsername.getText();**

**char[] password = txtPassword.getPassword();**

**char[] confirmPassword = txtConfirmPassword.getPassword();**

**String passwordStr = new String(password); // Convert char array to string**

**String confirmPasswordStr = new String(confirmPassword);**

**// Simple validation (you can extend this to check for existing users)**

**if (username.isEmpty() || passwordStr.isEmpty() || confirmPasswordStr.isEmpty()) {**

**lblMessage.setText("Please fill in all fields.");**

**lblMessage.setForeground(Color.RED);**

**} else if (!passwordStr.equals(confirmPasswordStr)) {**

**lblMessage.setText("Passwords do not match.");**

**lblMessage.setForeground(Color.RED);**

**} else {**

**lblMessage.setText("Account created successfully!");**

**lblMessage.setForeground(Color.GREEN);**

**JOptionPane.showMessageDialog(this, "Account created for " + username);**

**dispose(); // Close signup window**

**new LoginPage().setVisible(true); // Return to login page**

**}**

**}**

**public static void main(String[] args) {**

**SwingUtilities.invokeLater(new Runnable() {**

**@Override**

**public void run() {**

**SignUpPage signUpPage = new SignUpPage();**

**signUpPage.setVisible(true);**

**}**

**});**

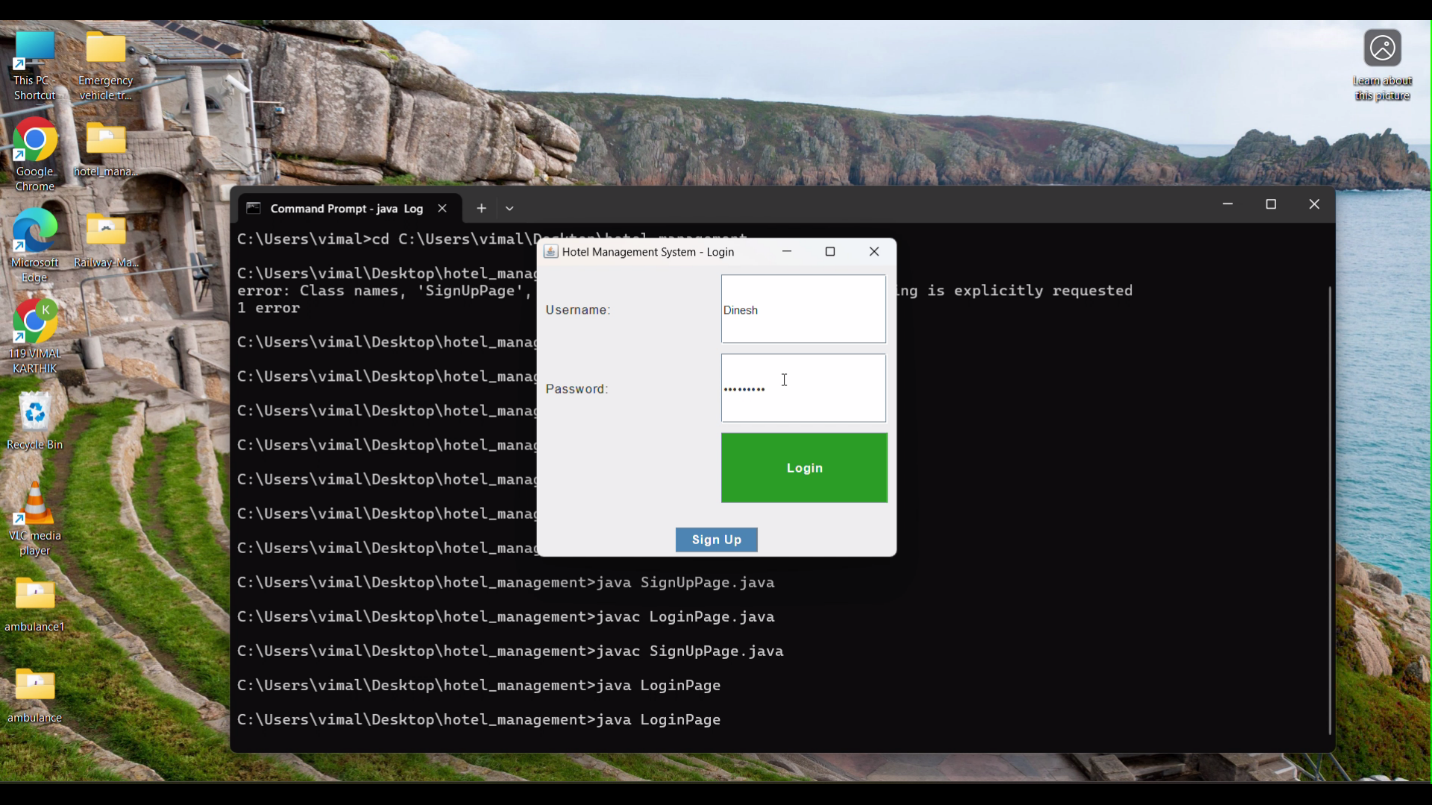
**}**

**}**

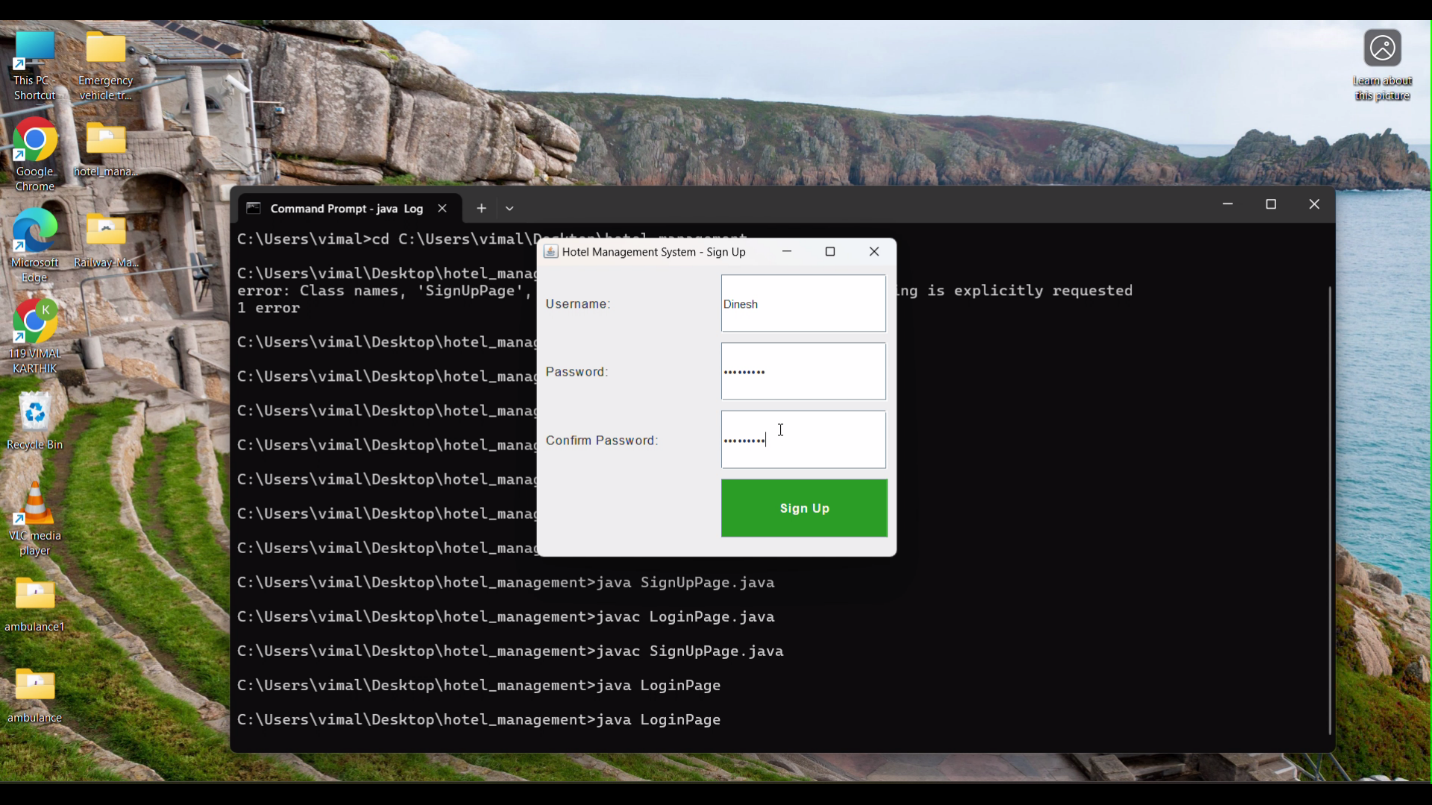
**5. RESULTS AND DISCUSSIONS**

RESULTS:

**Login Page of the Software**

****

**Sign up Page of the Software**

****

**Dashboard Page, with Item Request Feature**

****

**DISCUSSIONS**:

A Hotel Management System (HMS) streamlines hotel operations by automating tasks such as reservations, guest check-ins, billing, and room management. It helps improve efficiency, reduce human errors, and enhance the overall guest experience. Key features include room availability tracking, guest profiles, payment processing, and reporting. The system typically includes user roles such as administrators, front desk staff, and housekeeping, each with different access levels. It can be built using technologies like web development frameworks for the frontend and databases such as MySQL for the backend. Real-time data updates and security are crucial for ensuring smooth operations. Challenges include scalability, user experience design, and integration with third-party services. Future enhancements could involve AI for personalized services and IoT integration for smart rooms.

**Project Significance**

A Hotel Management System (HMS) project plays a crucial role in enhancing the operational efficiency of hotels by automating key processes like reservations, check-ins, billing, and room management. It reduces the manual workload and minimizes errors, leading to smoother daily operations. The system improves guest experience by offering faster services, personalized interactions, and easy online booking. By tracking room availability in real time, it helps optimize room occupancy and revenue through dynamic pricing.

Additionally, the HMS enhances data security by securely storing sensitive guest information and ensuring compliance with privacy regulations. With integrated reporting and analytics, hotel managers can make informed decisions based on performance data, improving profitability. The system is scalable, allowing hotels to expand without overhauling their operations. Moreover, it reduces operational costs by automating manual tasks, streamlining workflows, and optimizing

resource allocation.

Ultimately, an effective HMS gives hotels a competitive advantage by offering modern, efficient services that attract more guests, improve satisfaction, and increase profitability.

**6.CONCLUSION**

In conclusion, a Hotel Management System (HMS) plays a vital role in automating and streamlining hotel operations, leading to significant improvements in efficiency and accuracy. By handling essential tasks such as reservations, guest check-ins/check-outs, billing, and room assignments, the system reduces the reliance on manual work and minimizes errors. This automation enhances the overall guest experience by offering faster service, personalized options, and smoother interactions. Additionally, the system optimizes revenue management through real-time tracking of room availability and dynamic pricing based on demand, ensuring maximum occupancy and profitability.

The HMS provides comprehensive data and analytics that enable hotel management to make informed decisions, optimize operations, and identify areas for improvement. It is scalable, making it suitable for hotels of various sizes, from small boutique establishments to large hotel chains, without requiring major system changes as the business grows. With robust security measures in place to protect sensitive guest data, the system ensures compliance with privacy regulations and builds trust with customers.

By reducing operational costs through automation and improving resource management, the HMS helps hotels achieve greater cost efficiency. Moreover, it offers a competitive edge in the hospitality market by enhancing customer satisfaction, improving brand reputation, and boosting guest retention. Ultimately, the Hotel Management System is an essential tool for modern hotels aiming to thrive in an increasingly competitive and technology-driven industry.

**7.REFERENCES**

**Textbooks:**

1. "Hotel Management and Operations"by Michael J. O'Fallon and Denney G. Rutherford

2. "Introduction to Hospitality Management"\*\* by John R. Walker

3. "Managing Hospitality Organizations"\*\* by Robert A. Brymer

4. "Hospitality Management: A Brief Introduction"\*\* by Tim Hore

5. "Revenue Management for the Hospitality Industry"\*\* by David K. Hayes and Allisha A. Miller

6. "Hotel Front Office Management"\*\* by James A. Bardi

7. "The Theory and Practice of Revenue Management"\*\* by K. M. Cross

8. "The Cornell School of Hotel Administration on Hospitality: Cutting Edge Thinking and Practice"\*\* by Michael C. Sturman, Jack B. Corgel, and Rohit Verma.

**Websites:**

<https://www.geeksforgeeks.org/establishing-jdbc-connection-in-java/>

<https://www.javatpoint.com/example-to-connect-to-the-mysql-database>

Videos:

<https://www.youtube.com/watch?v=ZsbgiTR6osA&t=942s>

https://www.youtube.com/watch?v=8aTpDfsYTNQ

DBMS(Database Management Systems):  
https://www.youtube.com/watch?v=CCENlDeGvv8&t=378s

**Oracle. (2024).** *The Java™ Tutorials*. Oracle.  
Available at: <https://docs.oracle.com/javase/tutorial/>  
This resource provides comprehensive documentation on Java programming, including tutorials on topics such as Java Swing, JDBC, and other fundamental aspects of Java development.

**Conference Papers**

* **Keller, A., & Liu, S. (2020).** "Enhancing User Authentication in Resource Management Systems: A Case Study." *Proceedings of the 2020 International Conference on Computer Science and Software Engineering (ICCSSE)*.

This paper discusses approaches to improving security in resource management systems, including advanced authentication mechanisms and secure user access controls. The paper focuses on how to protect sensitive data while ensuring a seamless user experience.

* **Ramaswamy, A., & Kamat, R. (2019).** "Implementing Secure Login Systems for Web-Based Applications." *Proceedings of the 2019 International Conference on Secure Software Engineering (ICSSE)*.

This paper presents methodologies for building secure login systems, covering common vulnerabilities like SQL injection, session management, and encryption. It offers practical insights into designing secure login systems for web and desktop applications.

**Journal Articles**

Chaudhary, S., & Kumar, A. (2021). "A Survey on User Authentication Mechanisms in Web Applications." *International Journal of Computer Science and Security*, 15(3), 200-215.

* This article provides a comprehensive survey of various user authentication mechanisms commonly implemented in web applications. It reviews the strengths and weaknesses of different authentication methods, including password-based systems, multi-factor authentication (MFA), and biometric systems, offering a broad perspective on securing user login processes.

Github-Link:

https://github.com/DineshKarthik-231501042/java