**Airport Management System**

**A MINI-PROJECT REPORT**

**Submitted by**

**Prasannakumar P 230701238**

**Rakesh David 230701256**

**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 “AIRPORT MANAGEMENT SYSTEM” is the bonafide work of “PRASANNAKUMAR P, RAKESH D” who carried out the project work under my supervision.

Submitted for the practical examination held on: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

SIGNATURE

Ms. ASWANA LAL

Asst.Professor

Computer Science and Engineering,

Rajalakshmi Engineering College (Autonomous), Thandalam,Chennai-602105

**INTERNAL EXAMINER SIGNATURE** **EXTERNAL EXAMINER SIGNATURE**

S.NO TITLE PAGE

Abstract

1. Introduction 05
2. Scope of Project 06
3. UML Diagrams 09
4. Code Implementation 22
5. Conclusion 31
6. Future Work 33
7. Reference 35

**Abstract**

The **Airport Management System** is a comprehensive, automated solution designed to streamline and simplify airport operations through the use of modern technologies. Developed using **Java** and **MySQL Workbench**, this system is aimed at optimizing core airport functions such as **flight scheduling**, **ticket booking**, **passenger management**, and **real-time data processing**. By integrating these functionalities into one cohesive platform, the system minimizes human errors, increases efficiency, and enhances the overall passenger experience.

Airports, which are vital hubs for travel, require seamless management to handle complex logistics. Traditionally, much of the work related to flight management, ticketing, and passenger information is done manually, resulting in inefficiencies, delays, and errors. The **Airport Management System** tackles these challenges by automating key airport functions, thus improving operational flow and ensuring that airport staff can focus on critical tasks while passengers benefit from faster, more reliable services.

This system allows airport staff and administrators to manage the following:

* **Flight Scheduling and Management:** Administrators can easily update and monitor the flight schedules, ensuring passengers are aware of real-time changes to flight times, delays, and cancellations. The system automatically tracks and updates flight details, making it easier for staff to manage large volumes of data.
* **Passenger Management:** The system allows the registration and storage of passenger data, including personal details, booking history, and preferences. This enhances personalized service and provides an accessible record of passenger information, which can be used to streamline check-in processes.
* **Ticket Booking System:** The platform allows passengers to book, modify, and cancel flight tickets online, providing a real-time view of available flights. This reduces waiting time at the counter, making the booking process faster and more convenient for travelers.
* **Real-time Notifications:** The system generates real-time notifications for passengers, informing them of flight status changes, ticket confirmations, and other important updates. This feature improves communication between the airport and passengers, reducing confusion and improving satisfaction.

In addition to providing functional benefits, the system also prioritizes **security** and **data integrity** by utilizing **MySQL** to securely store flight, passenger, and booking information. Through these robust data management techniques, the system ensures that all information is easily accessible, accurate, and up-to-date. The use of **Java** enhances the system’s scalability and flexibility, allowing for easy updates, integration with other airport management systems, and adaptation to evolving technological needs.

The **Airport Management System** not only contributes to smoother operational workflows but also fosters a better travel experience by reducing human errors, speeding up administrative processes, and providing real-time information to passengers and airport staff. With future enhancements, the system can be expanded to include additional functionalities such as **mobile integration** for passengers, predictive flight delay analysis, and **biometric authentication** for improved passenger security.

**Scope of the Project**

The **Airport Management System** (AMS) is designed to automate, manage, and optimize key airport operations, ensuring that airports run efficiently and deliver excellent services to passengers. This system integrates multiple functionalities that cover the entire lifecycle of airport operations, from flight scheduling to ticket booking, passenger management, and real-time communication. The project encompasses several critical components that streamline daily tasks, enhance data accuracy, and improve the overall user experience for both airport staff and passengers.

**Key Areas of Scope**

1. **User Authentication & Role Management:**
   * The system supports different user roles with distinct access privileges:
     + **Admin:** Full access to all operations including flight schedule management, ticket bookings, and system configuration.
     + **Passenger:** Limited access to personal details, ticket booking, and flight status.
   * **User authentication** ensures that only authorized individuals can access certain features, maintaining data integrity and security.
2. **Flight Management System:**
   * The system allows **airport administrators** to manage **flight schedules** efficiently. Key functions include:
     + **Adding new flights** with details such as flight number, departure/arrival times, destination, and status (on-time, delayed, etc.).
     + **Modifying and updating flight information**, including changes to flight schedules, cancellations, or rerouting.
     + **Tracking real-time flight status**, providing administrators with updated details on the current status of flights.
   * The **flight management system** ensures that all flight-related data is accurate and up-to-date, reducing the chances of errors or miscommunications.
3. **Ticket Booking System:**
   * The ticket booking component allows passengers to **search for available flights**, **book tickets**, and **view their booking details**. Features include:
     + **Real-time flight availability:** Passengers can view available flights based on their preferred departure/arrival times and destination.
     + **Ticket reservations and cancellations:** Passengers can book tickets, cancel bookings, and receive immediate confirmations.
     + **Ticket modifications:** Allows passengers to modify bookings, such as changing flight dates or passenger details, based on availability and fare rules.
   * The system ensures **real-time updates** and **data synchronization**, guaranteeing accurate availability and seat reservations.
4. **Passenger Information Management:**
   * The system stores and manages **personal details** of passengers, such as:
     + **Passenger profiles:** Including name, contact details, and travel history.
     + **Booking history:** The system tracks past flight bookings and cancellations, enabling administrators to manage frequent travelers or offer personalized services.
     + **Passenger preferences:** The system can store specific preferences (e.g., seat selection, meal preferences) to offer a customized service during check-in or on flights.
   * This feature ensures that all passenger information is **securely stored** and **easily accessible**, helping airport staff provide faster and more personalized services.
5. **Real-time Notifications:**
   * The system integrates **real-time notifications** to keep passengers and airport staff informed:
     + **Flight status updates:** Passengers receive notifications for flight delays, cancellations, or boarding updates.
     + **Booking confirmations:** Passengers are notified immediately upon successful ticket bookings, cancellations, or modifications.
     + **Payment reminders and alerts:** For pending payments or ticket changes.
   * This ensures that both passengers and staff are **constantly updated**, improving communication and minimizing misunderstandings.
6. **Database Integration:**
   * The system utilizes **MySQL** as the backend database to manage and store critical data, including:
     + **Flight schedules**
     + **Ticket reservations**
     + **Passenger information**
   * The database ensures **data consistency** and **real-time updates**, which are essential for the smooth functioning of the airport. By using a relational database system, the system can handle large datasets efficiently and support multiple concurrent users.
7. **Reporting and Data Analytics:**
   * The system provides **reporting features** for airport management to monitor key metrics, such as:
     + **Flight performance**: Monitoring on-time performance, cancellations, and delays.
     + **Passenger trends**: Analyzing booking trends, peak travel periods, and frequent destinations.
     + **Financial reports**: Generating revenue reports from ticket sales, cancellations, and refunds.
   * These reports help airport authorities make **data-driven decisions**, optimize resources, and improve the overall management of the airport.
8. **System Scalability:**
   * The system is designed to be **scalable**, allowing it to handle growing data and an increasing number of users (both staff and passengers).
   * Future enhancements can include additional features, such as **mobile integration**, **biometric authentication**, and integration with **airport security systems** or **airlines’ reservation systems**.
   * Scalability is also supported through the use of **Java** and **MySQL**, which are both highly adaptable for future system expansions and integrations.

**UML Diagram**

A diagram of a flight

Description automatically generated

**Code Implementation**

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.\*;

// Flight Management class (stub)

// Ticket Booking class with form fields

class TicketBooking extends JFrame {

    public TicketBooking() {

        setTitle("Ticket Booking");

        setSize(400, 300);

        setDefaultCloseOperation(JFrame.DISPOSE\_ON\_CLOSE);

        setLocationRelativeTo(null);

        // Create the form panel

        JPanel panel = new JPanel();

        panel.setLayout(new GridLayout(5, 2, 10, 10)); // 5 rows, 2 columns layout

        // Create form fields

        JLabel nameLabel = new JLabel("Name:");

        JTextField nameField = new JTextField();

        JLabel flightLabel = new JLabel("Flight Number:");

        JTextField flightField = new JTextField();

        JLabel dateLabel = new JLabel("Date (DD/MM/YYYY):");

        JTextField dateField = new JTextField();

        JLabel seatLabel = new JLabel("Seat Preference:");

        JComboBox<String> seatComboBox = new JComboBox<>(new String[]{"Window", "Aisle", "Middle"});

        JButton submitButton = new JButton("Submit");

        // Add components to the panel

        panel.add(nameLabel);

        panel.add(nameField);

        panel.add(flightLabel);

        panel.add(flightField);

        panel.add(dateLabel);

        panel.add(dateField);

        panel.add(seatLabel);

        panel.add(seatComboBox);

        panel.add(new JLabel()); // Empty cell for alignment

        panel.add(submitButton);

        // Action listener for the submit button

        submitButton.addActionListener(new ActionListener() {

            public void actionPerformed(ActionEvent e) {

                // Fetch input values

                String name = nameField.getText();

                String flightNumber = flightField.getText();

                String date = dateField.getText();

                String seatPreference = (String) seatComboBox.getSelectedItem();

                // Display confirmation dialog

                JOptionPane.showMessageDialog(null,

                    "Ticket booked successfully!\n" +

                    "Name: " + name + "\n" +

                    "Flight Number: " + flightNumber + "\n" +

                    "Date: " + date + "\n" +

                    "Seat Preference: " + seatPreference,

                    "Confirmation",

                    JOptionPane.INFORMATION\_MESSAGE);

                // Close the booking window

                dispose();

            }

        });

        // Add panel to the frame

        add(panel);

    }

}

// Check-In System class with form fields

class CheckInSystem extends JFrame {

    public CheckInSystem() {

        setTitle("Check-In System");

        setSize(400, 300);

        setDefaultCloseOperation(JFrame.DISPOSE\_ON\_CLOSE);

        setLocationRelativeTo(null);

        // Create the form panel

        JPanel panel = new JPanel();

        panel.setLayout(new GridLayout(4, 2, 10, 10)); // 4 rows, 2 columns layout

        // Create form fields

        JLabel nameLabel = new JLabel("Name:");

        JTextField nameField = new JTextField();

        JLabel flightLabel = new JLabel("Flight Number:");

        JTextField flightField = new JTextField();

        JButton checkInButton = new JButton("Check-In");

        // Add components to the panel

        panel.add(nameLabel);

        panel.add(nameField);

        panel.add(flightLabel);

        panel.add(flightField);

        panel.add(new JLabel()); // Empty cell for alignment

        panel.add(checkInButton);

        // Action listener for the check-in button

        checkInButton.addActionListener(new ActionListener() {

            public void actionPerformed(ActionEvent e) {

                // Fetch input values

                String name = nameField.getText();

                String flightNumber = flightField.getText();

                // Display confirmation dialog

                JOptionPane.showMessageDialog(null,

                    "Check-In successful!\n" +

                    "Name: " + name + "\n" +

                    "Flight Number: " + flightNumber,

                    "Confirmation",

                    JOptionPane.INFORMATION\_MESSAGE);

                // Close the check-in window

                dispose();

            }

        });

        // Add panel to the frame

        add(panel);

    }

}

// Main class for the application

public class Main extends JFrame {

    private String userRole;

    // Constructor

    public Main(String userRole) {

        this.userRole = userRole;

        setTitle("Main - Airport Management System");

        setSize(500, 400);

        setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

        setLocationRelativeTo(null);

        JPanel panel = new JPanel(new GridLayout(3, 1)); // Layout for buttons

        JButton flightButton = new JButton("Manage Flights");

        JButton ticketButton = new JButton("Book Ticket");

        JButton checkInButton = new JButton("Check-In");

        // Action listener for Manage Flights button

        flightButton.addActionListener(new ActionListener() {

            public void actionPerformed(ActionEvent e) {

                // Only allow Admin to access flight management

                if ("Admin".equals(userRole)) {

                    new FlightManagement().setVisible(true); // Open Flight Management for Admin

                } else {

                    JOptionPane.showMessageDialog(null, "Access denied: Admin only.");

                }

            }

        });

        // Action listener for Book Ticket button

        ticketButton.addActionListener(new ActionListener() {

            public void actionPerformed(ActionEvent e) {

                TicketBooking ticketBooking = new TicketBooking(); // Create instance of TicketBooking

                ticketBooking.setVisible(true); // Make TicketBooking window visible

            }

        });

        // Action listener for Check-In button

        checkInButton.addActionListener(new ActionListener() {

            public void actionPerformed(ActionEvent e) {

                CheckInSystem checkInSystem = new CheckInSystem(); // Create instance of CheckInSystem

                checkInSystem.setVisible(true); // Make CheckInSystem window visible

            }

        });

        panel.add(flightButton); // Add buttons to panel

        panel.add(ticketButton);

        panel.add(checkInButton);

        add(panel); // Add panel to frame

    }

    // Main method

    public static void main(String[] args) {

        String role = "Admin"; // Can be changed to "User" or any other role for testing

        new Main(role).setVisible(true); // Pass the role to the constructor

    }

}

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.\*;

import java.sql.\*;

public class SignUpScreen extends JFrame {

    private JTextField usernameField;

    private JPasswordField passwordField;

    private JButton signUpButton;

    private JTextField roleField; // Admin or User

    public SignUpScreen() {

        setTitle("Sign Up - Airport Management System");

        setSize(400, 300);

        setLocationRelativeTo(null);

        setDefaultCloseOperation(JFrame.DISPOSE\_ON\_CLOSE);

        JPanel panel = new JPanel(new GridLayout(4, 2));

        JLabel usernameLabel = new JLabel("Username: ");

        usernameField = new JTextField();

        JLabel passwordLabel = new JLabel("Password: ");

        passwordField = new JPasswordField();

        JLabel roleLabel = new JLabel("Role (Admin/User): ");

        roleField = new JTextField();

        signUpButton = new JButton("Sign Up");

        panel.add(usernameLabel);

        panel.add(usernameField);

        panel.add(passwordLabel);

        panel.add(passwordField);

        panel.add(roleLabel);

        panel.add(roleField);

        panel.add(new JLabel()); // Empty space

        panel.add(signUpButton);

        add(panel);

        signUpButton.addActionListener(new ActionListener() {

            public void actionPerformed(ActionEvent e) {

                String username = usernameField.getText();

                String password = new String(passwordField.getPassword());

                String role = roleField.getText();

                try {

                    if (role.equals("Admin") || role.equals("User")) {

                        Connection conn = DatabaseConnection.getConnection();

                        String sql = "INSERT INTO Users (Username, Password, Role) VALUES (?, ?, ?)";

                        PreparedStatement stmt = conn.prepareStatement(sql);

                        stmt.setString(1, username);

                        stmt.setString(2, password);

                        stmt.setString(3, role);

                        stmt.executeUpdate();

                        JOptionPane.showMessageDialog(null, "Sign up successful.");

                        dispose(); // Close sign-up window

                    } else {

                        JOptionPane.showMessageDialog(null, "Invalid role. Please choose Admin or User.");

                    }

                } catch (SQLException ex) {

                    ex.printStackTrace();

                    JOptionPane.showMessageDialog(null, "Error during sign-up.");

                }

            }

        });

    }

    public static void main(String[] args) {

        new SignUpScreen().setVisible(true);

    }

}

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.\*;

public class LoginWindow extends JFrame {

    private JTextField usernameField;

    private JPasswordField passwordField;

    private JComboBox<String> roleComboBox;

    public LoginWindow() {

        setTitle("Login - Airport Management System");

        setSize(400, 300);

        setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

        setLocationRelativeTo(null);

        JPanel panel = new JPanel();

        panel.setLayout(new GridLayout(4, 2));

        JLabel usernameLabel = new JLabel("Username:");

        usernameField = new JTextField();

        JLabel passwordLabel = new JLabel("Password:");

        passwordField = new JPasswordField();

        JLabel roleLabel = new JLabel("Role:");

        roleComboBox = new JComboBox<>(new String[] {"Admin", "User"});

        JButton loginButton = new JButton("Login");

        loginButton.addActionListener(new ActionListener() {

            public void actionPerformed(ActionEvent e) {

                // Check for valid username/password

                String username = usernameField.getText();

                String password = new String(passwordField.getPassword());

                String role = (String) roleComboBox.getSelectedItem();

                if (username.equals("admin") && password.equals("admin123") || username.equals("user") && password.equals("user123")) {

                    // Login successful

                    JOptionPane.showMessageDialog(null, "Login Successful!");

                    dispose(); // Close the login window

                    // Pass user role to the main window and show it

                    new Main(role).setVisible(true);

                } else {

                    // Login failed

                    JOptionPane.showMessageDialog(null, "Invalid credentials.");

                }

            }

        });

        panel.add(usernameLabel);

        panel.add(usernameField);

        panel.add(passwordLabel);

        panel.add(passwordField);

        panel.add(roleLabel);

        panel.add(roleComboBox);

        panel.add(new JLabel()); // Placeholder

        panel.add(loginButton);

        add(panel);

    }

    public static void main(String[] args) {

        new LoginWindow().setVisible(true); // Show login window initially

    }

}

import java.sql.\*;

public class LoginService {

    public String authenticateUser(String username, String password) throws SQLException {

        String sql = "SELECT Role FROM Users WHERE Username = ? AND Password = ?";

        try (Connection conn = DatabaseConnection.getConnection();

             PreparedStatement stmt = conn.prepareStatement(sql)) {

            stmt.setString(1, username);

            stmt.setString(2, password);

            ResultSet rs = stmt.executeQuery();

            if (rs.next()) {

                return rs.getString("Role");

            } else {

                return null; // Invalid credentials

            }

        }

    }

}

import java.awt.\*;

import java.awt.event.\*;

import java.sql.\*;

import javax.swing.\*;

public class LoginScreen extends JFrame {

    private JTextField usernameField;

    private JPasswordField passwordField;

    private JButton loginButton, signUpButton;

    private LoginService loginService = new LoginService();

    public LoginScreen() {

        setTitle("Login - Airport Management System");

        setSize(400, 300);

        setLocationRelativeTo(null);

        setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

        // Create panel with background image

        JPanel panel = new JPanel() {

            @Override

            protected void paintComponent(Graphics g) {

                super.paintComponent(g);

                ImageIcon backgroundImage = new ImageIcon("airportmanagementsystem.jpeg"); // Assuming image.jpeg is in the project directory

                g.drawImage(backgroundImage.getImage(), 0, 0, null);

            }

        };

        panel.setLayout(new GridLayout(4, 2));

        // Username and password fields

        JLabel usernameLabel = new JLabel("Username: ");

        usernameField = new JTextField();

        JLabel passwordLabel = new JLabel("Password: ");

        passwordField = new JPasswordField();

        // Buttons for login and sign-up

        loginButton = new JButton("Login");

        signUpButton = new JButton("Sign Up");

        panel.add(usernameLabel);

        panel.add(usernameField);

        panel.add(passwordLabel);

        panel.add(passwordField);

        panel.add(new JLabel()); // Empty space

        panel.add(loginButton);

        panel.add(new JLabel()); // Empty space

        panel.add(signUpButton);

        add(panel);

        // Action listeners for login and sign-up

        loginButton.addActionListener(new ActionListener() {

            public void actionPerformed(ActionEvent e) {

                String username = usernameField.getText();

                String password = new String(passwordField.getPassword());

                try {

                    String role = loginService.authenticateUser(username, password);

                    if (role != null) {

                        JOptionPane.showMessageDialog(null, "Login successful! Role: " + role);

                        new Main(role).setVisible(true); // Pass role to main app

                        dispose(); // Close login window

                    } else {

                        JOptionPane.showMessageDialog(null, "Invalid credentials.");

                    }

                } catch (SQLException ex) {

                    ex.printStackTrace();

                    JOptionPane.showMessageDialog(null, "Database error during login.");

                }

            }

        });

        signUpButton.addActionListener(new ActionListener() {

            public void actionPerformed(ActionEvent e) {

                // Open the sign-up form

                new SignUpScreen().setVisible(true);

            }

        });

    }

    public static void main(String[] args) {

        new LoginScreen().setVisible(true);

    }

}

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.SQLException;

// Method to create a connection to the database

public class DatabaseConnection {

    // Ensure this method is public so it can be accessed from other classes

    public static Connection getConnection() throws SQLException {

        try {

            // Return a connection to the database using the JDBC URL, username, and password

            return DriverManager.getConnection("jdbc:mysql://localhost:3306/airportmanagement", "root", "Prasannakumar20");

        } catch (SQLException e) {

            // Throw the exception if a connection cannot be established

            throw new SQLException("Failed to connect to the database.", e);

        }

    }

}

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.\*;

import java.sql.\*;

public class FlightManagement extends JFrame {

    public FlightManagement() {

        setTitle("Flight Management");

        setSize(500, 300);

        setLocationRelativeTo(null);

        JPanel panel = new JPanel(new GridLayout(5, 2));

        JLabel flightLabel = new JLabel("Flight Number:");

        JTextField flightField = new JTextField();

        JLabel departureCityLabel = new JLabel("Departure City:");

        JTextField departureCityField = new JTextField();

        JLabel arrivalCityLabel = new JLabel("Arrival City:");

        JTextField arrivalCityField = new JTextField();

        JLabel capacityLabel = new JLabel("Capacity:");

        JTextField capacityField = new JTextField();

        JButton addFlightButton = new JButton("Add Flight");

        panel.add(flightLabel); panel.add(flightField);

        panel.add(departureCityLabel); panel.add(departureCityField);

        panel.add(arrivalCityLabel); panel.add(arrivalCityField);

        panel.add(capacityLabel); panel.add(capacityField);

        panel.add(addFlightButton);

        add(panel);

        addFlightButton.addActionListener(new ActionListener() {

            public void actionPerformed(ActionEvent e) {

                // Check if the input fields are not empty

                if (flightField.getText().isEmpty() || departureCityField.getText().isEmpty() ||

                    arrivalCityField.getText().isEmpty() || capacityField.getText().isEmpty()) {

                    JOptionPane.showMessageDialog(null, "All fields must be filled.");

                    return;

                }

                try {

                    // Database connection

                    Connection conn = DatabaseConnection.getConnection();

                    String sql = "INSERT INTO Flights (FlightNumber, DepartureCity, ArrivalCity, Capacity) VALUES (?, ?, ?, ?)";

                    PreparedStatement stmt = conn.prepareStatement(sql);

                    stmt.setString(1, flightField.getText());

                    stmt.setString(2, departureCityField.getText());

                    stmt.setString(3, arrivalCityField.getText());

                    // Check if Capacity is a valid integer before inserting

                    try {

                        stmt.setInt(4, Integer.parseInt(capacityField.getText()));

                    } catch (NumberFormatException ex) {

                        JOptionPane.showMessageDialog(null, "Capacity must be a valid integer.");

                        return;

                    }

                    // Execute the query

                    int rowsAffected = stmt.executeUpdate();

                    if (rowsAffected > 0) {

                        JOptionPane.showMessageDialog(null, "Flight added successfully.");

                    } else {

                        JOptionPane.showMessageDialog(null, "Failed to add flight.");

                    }

                    conn.close(); // Close the connection

                } catch (SQLException ex) {

                    ex.printStackTrace(); // Print detailed error to the console

                    JOptionPane.showMessageDialog(null, "Error adding flight: " + ex.getMessage());

                }

            }

        });

    }

    public static void main(String[] args) {

        // Initialize and show the JFrame

        FlightManagement frame = new FlightManagement();

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

        frame.setVisible(true);

    }

}

**DATABASE**

-- Create Users table

CREATE TABLE Users (

UserID INT PRIMARY KEY,

Username VARCHAR2(50) UNIQUE,

Password VARCHAR2(100),

Role VARCHAR2(20) CHECK (Role IN ('Admin', 'Staff'))

);

-- Create Flights table

CREATE TABLE Flights (

FlightID INT PRIMARY KEY,

FlightNumber VARCHAR2(20) NOT NULL,

DepartureCity VARCHAR2(50),

ArrivalCity VARCHAR2(50),

DepartureTime DATE,

ArrivalTime DATE,

Capacity INT

);

-- Create Passengers table

CREATE TABLE Passengers (

PassengerID INT PRIMARY KEY,

FirstName VARCHAR2(50),

LastName VARCHAR2(50),

Email VARCHAR2(100),

Phone VARCHAR2(15)

);

-- Create Tickets table

CREATE TABLE Tickets (

TicketID INT PRIMARY KEY,

FlightID INT,

PassengerID INT,

Status VARCHAR2(20) CHECK (Status IN ('Booked', 'Checked-In', 'Cancelled')),

PaymentStatus VARCHAR2(20) CHECK (PaymentStatus IN ('Paid', 'Pending')),

FOREIGN KEY (FlightID) REFERENCES Flights(FlightID),

FOREIGN KEY (PassengerID) REFERENCES Passengers(PassengerID)

);

-- Create CheckIn table

CREATE TABLE CheckIn (

CheckInID INT PRIMARY KEY,

TicketID INT,

CheckInTime DATE,

FOREIGN KEY (TicketID) REFERENCES Tickets(TicketID)

);

**Output**

**A screenshot of a computer

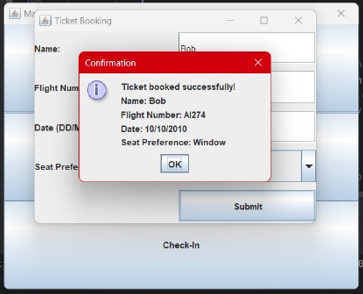
Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screen shot of a computer

Description automatically generated**

****

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**Conclusion**

Here’s a summarized version of the **Conclusion**:

The **Airport Management System** developed using **Java** and **MySQL Workbench** provides an efficient solution for managing airport operations like **flight scheduling**, **ticket booking**, and **passenger management**. It automates key tasks, reduces human errors, and improves operational efficiency.

**Key Achievements:**

* **Automation of Core Operations:** The system automates critical tasks, reducing administrative overhead and minimizing delays and errors.
* **Improved Customer Experience:** Passengers can easily book flights online, and real-time notifications keep them informed on flight status.
* **Data Integrity and Security:** Using **MySQL**, the system ensures secure and consistent data storage with role-based access.
* **Real-time Updates:** Passengers and staff are always informed of the latest flight and booking information.

**Challenges Encountered:**

* **Database Complexity:** Designing the database to handle large data volumes posed challenges.
* **Integration with Existing Systems:** Integrating the new system with legacy infrastructure and external systems needs improvement.

**Future Enhancements:**

* **Mobile App Integration:** Future versions could include a mobile app for greater accessibility.
* **Biometric Authentication:** Enhancing security with biometric features like fingerprints or facial recognition.
* **AI and Blockchain Integration:** Using AI for flight predictions and blockchain for secure ticketing.
* **Global Travel Network Integration:** Connecting with global networks for a unified ticket booking experience.

**Final Thoughts:**

The **Airport Management System** represents a major step in modernizing airport operations, improving efficiency, customer experience, and data security. While there are challenges with system integration and scalability, the project lays a solid foundation for future advancements in airport management.

**References**

* [www.javapoint.com/](http://www.javapoint.com/)
* [www.geeksforgeeks.com](http://www.geeksforgeeks.com)
* <https://chat.openai.com/>
* <https://chatuml.com/>