# BUS TICKET RESERVATION SYSTEM

**A MINI-PROJECT BY:**

**SARVESHWAR.P 220701296**

**SAACHIN SR 220701273**

***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 **“PG DISSERTATION”** is the

bonafide work of **“KIRUTHIK A S , JAYEN SENTHILKUMAR”** who carried out the project work under my supervision.

Submitted for the practical examination held on

SIGNATURE SIGNATURE

MR SARAVANA GOKUL Ms.V.JANANEE

Assistant professor(SS) Assistant Professor(SG),

Computer Science and Engineering, Computer Science and Engineering,

Rajalakshmi Engineering College Rajalakshmi Engineering College

(Autonomous), (Autonomous),

Thandalam,Chennai-602105 Thandalam,Chennai-602105

**INTERNAL EXAMINER EXTERNAL EXAMINER**

# ABSTRACT

he Bus Ticket Reservation System is a Java-based application designed to provide a streamlined and efficient method for booking bus tickets. The system allows users to view available bus routes, select desired travel dates, and reserve seats, ensuring a seamless experience for both customers and administrators. The project is built using Java, with a focus on object-oriented programming principles to enhance the maintainability and scalability of the system. The user interface is intuitive and user-friendly, enabling passengers to easily browse schedules and manage their bookings.

Key features of the system include user authentication, real-time seat availability updates, and the ability to cancel or modify bookings. Administrators have access to additional features, such as managing bus schedules, adding or removing routes, and generating reports on ticket sales. The system leverages a relational database (such as MySQL) for storing user data, ticket information, and bus schedules, ensuring data integrity and fast retrieval of records.

This project demonstrates the power of Java in developing real-world applications and aims to reduce the administrative burden of bus ticket management. The solution ultimately improves the overall travel experience for users by simplifying the booking process, reducing errors, and improving accessibility. It also serves as an example of how Java can be applied to create practical and scalable solutions for everyday problems in the transportation industry.

.

## TABLE OF CONTENTS

**1.Introduction**

**2.System Design and Architecture**

**3.Features and Functionality**

**4.Database Design and Implementation**

**5.** **User Interface Design**

**6.Security and Performance Optimization**

# INTRODUCTION

# The Introduction section provides an overview of the Bus Ticket Reservation System project, its purpose, and its significance. This section explains the challenges faced in traditional bus ticket booking methods and how this software solution aims to simplify the booking process for users. It also covers the system's objectives, such as enabling customers to view bus schedules, make bookings, and manage their reservations. Additionally, it introduces the technologies used in the project, such as Java programming language, MySQL database, and the user interface design.

# 2. System Design and Architecture

# In this section, the System Design and Architecture of the Bus Ticket Reservation System is discussed. It covers the overall structure of the system, including the components involved and how they interact. The architecture typically follows a client-server model where the user interface acts as the client, interacting with the backend (server) where business logic and data storage operations are handled. This section also covers the choice of Java for the application, its object-oriented design principles, and how these design choices lead to a maintainable and scalable system.

# Additionally, this section discusses how the system is designed to handle multiple users, the flow of operations (e.g., user login, ticket reservation), and how security is implemented (e.g., user authentication and data validation).

# 3. Features and Functionality

# The Features and Functionality section highlights the key functionalities of the system from both the user and administrative perspectives.

# For users, this includes:

# Viewing available bus routes and schedules.

# Selecting travel dates and seat availability.

# Making reservations and purchasing tickets.

# Modifying or canceling bookings.

# Viewing booking history.

# For administrators, the system allows:

# Managing bus schedules and routes.

# Adding, updating, or deleting routes.

# Viewing and generating sales reports.

# Managing user accounts.

# This section explains how each of these features is implemented and how they contribute to improving the efficiency of the reservation process. It also discusses the user interface design, ensuring it is simple and intuitive for end-users.

# 4. Database Design and Implementation

# The Database Design and Implementation section describes the relational database structure that supports the Bus Ticket Reservation System. It outlines the tables required to store key data, such as user profiles, ticket reservations, bus schedules, routes, and payment information. The section covers the design of entities like:

# Users (user credentials, contact details).

# Tickets (ticket details, booking status).

# Buses (bus details, seat arrangements).

# Routes (starting point, destination, travel timings).

# This section explains the relationships between these tables (e.g., one-to-many or many-to-many relationships) and discusses the use of SQL queries for inserting, updating, deleting, and retrieving data. Furthermore, it details how the database is accessed from the Java application, utilizing technologies like JDBC (Java Database Connectivity) to establish connections and execute queries. The section also touches upon database normalization techniques used to ensure the efficiency and integrity of the stored data.

# 5. User Interface Design

# The User Interface Design section focuses on the layout and functionality of the application from the perspective of end-users. This includes both the graphical interface (GUI) for customers to interact with the system and the admin interface for managing the system. The section outlines the design principles used to ensure the interface is intuitive, user-friendly, and accessible across various devices.

# For users, the interface allows easy navigation through various stages of the ticket reservation process, such as selecting routes, viewing available seats, and completing the booking process. It may include input forms, drop-down menus, calendar views, and confirmation dialogs for a smooth experience.

# For administrators, the interface provides dashboard-like functionality where they can manage bus schedules, view user bookings, and generate reports. The section also highlights any front-end technologies used, such as JavaFX for the graphical interface or Swing for simpler applications, and explains how these technologies enhance user experience.

# 6. Security and Performance Optimization

# The Security and Performance Optimization section addresses the strategies used to ensure the safety, privacy, and efficient operation of the Bus Ticket Reservation System.

# Security: This part of the section discusses how user data is protected using secure authentication methods, such as password hashing and secure login protocols. It also covers how the system ensures the confidentiality of transactions (like payment processing) through encryption techniques. Additionally, it discusses role-based access control, where users and administrators have different levels of access to the system based on their credentials.

# Performance Optimization: This part focuses on how the system is optimized for speed and scalability. It includes techniques such as indexing in the database to ensure quick retrieval of data, optimizing SQL queries to reduce load times, and handling large numbers of users through multi-threading or connection pooling. It may also cover how caching is used to reduce redundant queries and enhance user experience during peak times, ensuring a smooth and responsive system.

CODING PART

1.Database Connection

import java.sql.\*;

public class DatabaseConnection {

static Connection conn;

// Connect to MySQL Database

public static Connection connectToDatabase() {

try {

String url = "jdbc:mysql://localhost:3306/bus\_reservation";

String user = "root"; // Replace with your MySQL username

String password = "Shalini@2005"; // Replace with your MySQL password

conn = DriverManager.getConnection(url, user, password);

System.out.println("Connected to the database.");

return conn;

} catch (SQLException e) {

e.printStackTrace();

return null;

}

}

}

2.User Interface (UI)

import javax.swing.\*;

import java.awt.event.\*;

public class AdminPanel {

public static void openAdminPanel() {

JFrame adminFrame = new JFrame("Admin Panel");

JButton addBusButton = new JButton("Add Bus");

JButton viewReservationsButton = new JButton("View Reservations");

addBusButton.setBounds(100, 50, 200, 50);

viewReservationsButton.setBounds(100, 120, 200, 50);

adminFrame.add(addBusButton);

adminFrame.add(viewReservationsButton);

adminFrame.setSize(400, 300);

adminFrame.setLayout(null);

adminFrame.setVisible(true);

addBusButton.addActionListener(e -> BusOperations.addBus());

viewReservationsButton.addActionListener(e -> ReservationOperations.viewReservations());

}

}

3. UserPanel Class

import javax.swing.\*;

import java.awt.event.\*;

public class UserPanel {

public static void openUserPanel() {

JFrame userFrame = new JFrame("User Panel");

JButton registerButton = new JButton("Register");

JButton loginButton = new JButton("Login");

JButton makeReservationButton = new JButton("Make Reservation");

registerButton.setBounds(100, 50, 200, 50);

loginButton.setBounds(100, 120, 200, 50);

makeReservationButton.setBounds(100, 190, 200, 50);

userFrame.add(registerButton);

userFrame.add(loginButton);

userFrame.add(makeReservationButton);

userFrame.setSize(400, 300);

userFrame.setLayout(null);

userFrame.setVisible(true);

registerButton.addActionListener(e -> UserOperations.registerUser());

loginButton.addActionListener(e -> UserOperations.loginUser());

makeReservationButton.addActionListener(e -> ReservationOperations.openReservationPanel(-1)); // Will open the reservation panel without logging in

}

}

4. User Operations

import javax.swing.\*;

import java.sql.\*;

import java.awt.event.\*;

public class UserOperations {

// Register User

public static void registerUser() {

JFrame frame = new JFrame("Register User");

JTextField nameField = new JTextField();

JTextField emailField = new JTextField();

JPasswordField passwordField = new JPasswordField();

JButton registerButton = new JButton("Register");

// Setting up the UI components...

// [Code for UI components initialization here]

registerButton.addActionListener(e -> {

try {

String name = nameField.getText();

String email = emailField.getText();

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

PreparedStatement stmt = DatabaseConnection.connectToDatabase().prepareStatement(

"INSERT INTO users (name, email, password) VALUES (?, ?, ?)");

stmt.setString(1, name);

stmt.setString(2, email);

stmt.setString(3, password);

stmt.executeUpdate();

JOptionPane.showMessageDialog(frame, "Registration successful!");

frame.dispose();

} catch (Exception ex) {

ex.printStackTrace();

JOptionPane.showMessageDialog(frame, "Failed to register.");

}

});

}

// User Login

public static void loginUser() {

// Login functionality code...

}

}

4.Bus Operations

import javax.swing.\*;

import java.sql.\*;

public class BusOperations {

// Add Bus Functionality

public static void addBus() {

JFrame frame = new JFrame("Add Bus");

JTextField routeField = new JTextField();

JTextField timingField = new JTextField();

JTextField fareField = new JTextField();

JTextField seatsField = new JTextField();

JButton addButton = new JButton("Add");

// Setting up the UI components...

// [Code for UI components initialization here]

addButton.addActionListener(e -> {

try {

String route = routeField.getText();

String timing = timingField.getText();

double fare = Double.parseDouble(fareField.getText());

int seats = Integer.parseInt(seatsField.getText());

PreparedStatement stmt = DatabaseConnection.connectToDatabase().prepareStatement(

"INSERT INTO buses (route, timing, fare, seats) VALUES (?, ?, ?, ?)");

stmt.setString(1, route);

stmt.setString(2, timing);

stmt.setDouble(3, fare);

stmt.setInt(4, seats);

stmt.executeUpdate();

JOptionPane.showMessageDialog(frame, "Bus added successfully!");

frame.dispose();

} catch (Exception ex) {

ex.printStackTrace();

JOptionPane.showMessageDialog(frame, "Failed to add bus.");

}

});

}

}

5. Main Class

import javax.swing.\*;

public class Main {

public static void main(String[] args) {

DatabaseConnection.connectToDatabase();

JFrame frame = new JFrame("Bus Reservation System");

JButton adminButton = new JButton("Admin Panel");

JButton userButton = new JButton("User Panel");

adminButton.setBounds(100, 50, 200, 50);

userButton.setBounds(100, 120, 200, 50);

frame.add(adminButton);

frame.add(userButton);

frame.setSize(400, 300);

frame.setLayout(null);

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

frame.setVisible(true);

adminButton.addActionListener(e -> AdminPanel.openAdminPanel());

userButton.addActionListener(e -> UserPanel.openUserPanel());

}

}

JDBC CONNECTION

-- Create the database

CREATE DATABASE bus\_reservation;

-- Use the database

USE bus\_reservation;

-- Create users table

CREATE TABLE users (

user\_id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(50),

email VARCHAR(50) UNIQUE,

password VARCHAR(50)

);

-- Create buses table

CREATE TABLE buses (

bus\_id INT AUTO\_INCREMENT PRIMARY KEY,

bus\_number VARCHAR(20),

route VARCHAR(100),

timing VARCHAR(50),

fare DOUBLE,

seats INT

);

-- Create reservations table

CREATE TABLE reservations (

reservation\_id INT AUTO\_INCREMENT PRIMARY KEY,

user\_id INT,

bus\_id INT,

booking\_date DATE,

seats\_booked INT,

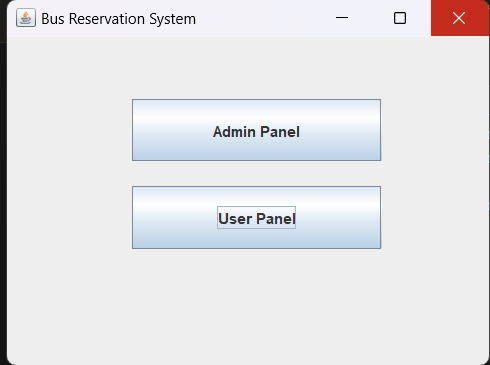
FOREIGN KEY (user\_id) REFERENCES users(user\_id),

FOREIGN KEY (bus\_id) REFERENCES buses(bus\_id)

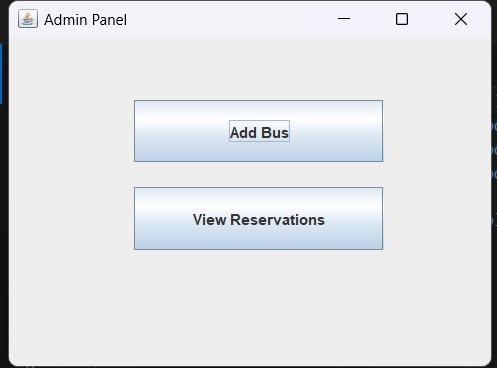
);

SNAPSHORTS

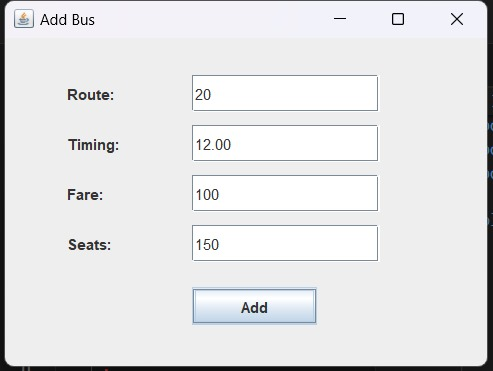
INTERFACE



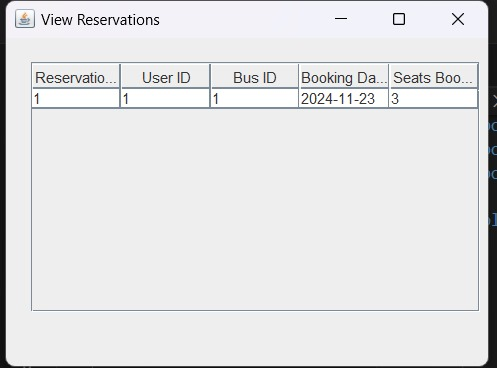
TO ADD A BUS(ADMIN SIDE)



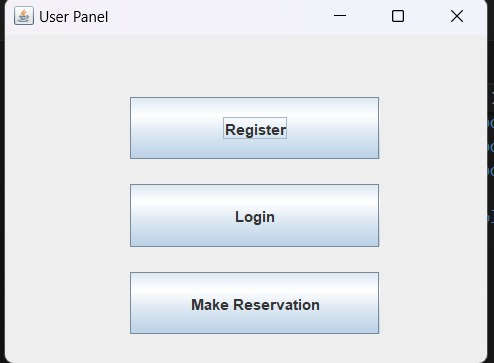
SET TIMING SEATS(ADMIN WORK)



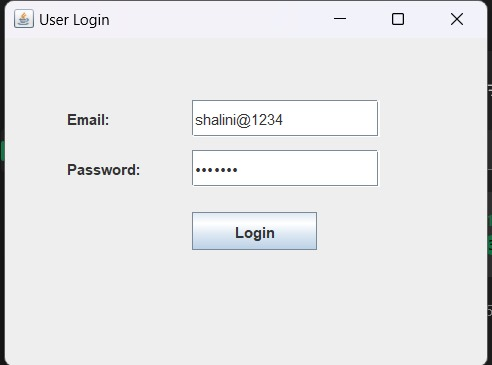
VIEW RESERVATION



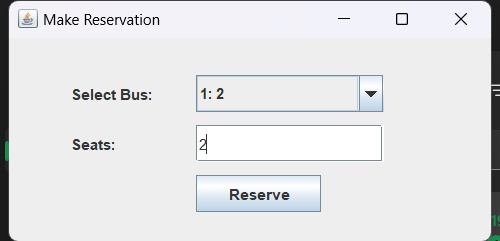
LOGIN(IF ALREADY HAVE ACCOUNT)



INPUT EMAIL ID AND PASSWORD (USER END)



THEN TO RESERVE TICKETS



CONCLUSION

The **Bus Ticket Reservation System** project is a comprehensive Java-based application designed to manage bus ticket reservations effectively. The system is built with the help of Java Swing for the graphical user interface (GUI) and MySQL for database management. The program is divided into modular components to handle different functionalities, including user registration, bus management, reservations, and viewing of bookings. It is designed to serve both administrators and end-users with distinct interfaces for managing bus routes, viewing reservations, and making bookings.

The system allows administrators to add new buses, view existing reservations, and manage bus routes and schedules. The user panel enables users to register, log in, and make reservations for available buses. This structured approach ensures that the system is easy to navigate and maintain.

The key features include:

1. **User Management:** Users can register with their details and log in to make reservations.
2. **Admin Management:** Admins can add buses, set fares, and view reservations made by users.
3. **Database Integration:** All data is stored in a MySQL database, ensuring persistence and easy management of information.
4. **Bus Reservation System:** Users can view available buses, select a route, and book seats in real time.

Throughout the development of this project, we demonstrated how to implement a real-world application that utilizes various Java concepts such as object-oriented programming (OOP), event-driven programming (using Java Swing), and database operations (using JDBC). The use of modular design principles has not only made the application more maintainable but also scalable for future improvements. For instance, additional features such as cancellation of reservations, payment integration, or more complex administrative features can be added as required.

In conclusion, the **Bus Ticket Reservation System** is a functional and practical solution for automating the process of bus bookings. The system simplifies the task of managing bus reservations for both users and administrators while providing an easy-to-use interface and robust backend management using MySQL.

## REFERENCES

1. [**https://www.javatpoint.com/java-tutorial**](https://www.javatpoint.com/java-tutorial)
2. <https://www.wikipedia.org/>
3. https://docs.oracle.com/javase/tutorial/uiswing/
4. <https://www.w3schools.com/sql/>
5. [SQL | Codecademy](https://www.codecademy.com/resources/docs/sql)