A Micro project on

**BUS TICKET SYSTEM**

Submitted in partial fulfillment of the

**Java Programming**

**GRIET Lab On Board (G-LOB)**

By

**Abbas Hussain(23241A0501)**

**Sai Teja (23241a0502)**

Under the esteemed guidance of

**Ramya Manasvi**

**Associate Professor**

****

**Department of Computer Science and Engineering**

**GOKARAJU RANGARAJU**

**INSTITUTE OF ENGINEERING AND TECHNOLOGY**

**(Approved by AICTE, Autonomous under JNTUH)**

**Bachupally, Kukatpally, Hyderabad-500090.**

2024-25



GOKARAJU RANGARAJU

INSTITUTE OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the micro project titled “BUS TICKET SYSTEM” is a bonafide work done Abbas Hussain (23241a0501), Sai Teja (23241a0502) under Java Programming Lab- GRIET Lab On Board (G-LOB) practice of our institute and that this work has not been submitted for the award of any other Degree/Diploma of any Institution/University.

Project Guide

Ramya Manansvi

Associate Professor

Table of Contents

|  |  |  |
| --- | --- | --- |
| Ch. No. | Chapter Name |  |
| 1 | Introduction |  |
|  | 1.1. Need for the Project |  |
|  | 1.2. Project Description  1.3 Components of the Projects |  |
| 2 | Requirement Analysis |  |
| 3 | System Design |  |
|  | 3.1. Flow Chart |  |
|  |
|  |
| 4 | Implementation |  |
| 5 | Results and Discussion |  |
| 6 | Conclusion |  |
| 7 | References |  |

[[1]](#footnote-1)

## 1. Introduction

### 1.1. Need for the Project

### In today's fast-paced world, efficient public transportation systems are crucial for daily commuting. Managing and accessing bus schedules, routes, and ticketing systems can be challenging without proper tools. A digital Bus Ticket Management System simplifies these tasks by offering users a convenient way to view buses, generate tickets, and manage bus details. This system is particularly beneficial in reducing manual processes, enhancing accessibility, and ensuring a seamless user experience for both passengers and administrators.

### 1.2. Project Description

### The Bus Ticket Management System is a graphical desktop application developed in Java using the Swing framework. It is designed to facilitate:

### Users: By enabling them to view available buses, generate tickets for their journey, and ensure smooth operations with minimal user input.

### Administrators: By providing tools to manage bus details like adding new buses and updating their status.

### The application features a user-friendly interface with interactive components such as dropdown menus, text areas, and buttons, making it accessible to users with varying technical expertise. Error-handling mechanisms ensure input validation, enhancing the reliability of the system. Additionally, it is visually appealing with distinct color schemes and intuitive navigation.

### 1.3. Components of the Project

### The project comprises the following key components:

### Graphical User Interface (GUI):

### Built using Swing, the GUI provides a clean, interactive layout for both users and administrators. Key elements include:

### Buttons: For actions like viewing buses, generating tickets, and accessing the admin panel.

### Dropdown Menus (JComboBox): For selecting start and destination stops.

### Text Area (JTextArea): For displaying bus details and tickets.

### Scroll Pane: For enhanced readability of the output.

### Bus Class:

### Represents the blueprint of a bus with attributes such as:

### Name

### Route

### Type (AC/Non-AC)

### Seat capacity

### Availability status

### Provides a formatted representation for display purposes.

### Functionality:

### View Buses: Displays a list of all available buses with detailed information.

### Generate Ticket: Allows users to generate a ticket by selecting valid start and destination stops. Includes error handling for invalid selections.

### Admin Panel: Provides administrators with a panel to add new buses, ensuring system scalability and adaptability.

### Event Handling:

### Uses action listeners to capture and respond to user inputs dynamically.

### Styling and User Experience:

### Applies a thoughtful color scheme and layout design for visual appeal and user-friendliness.

### This system serves as a stepping stone towards a more comprehensive and scalable public transportation management solution. It offers the flexibility to integrate advanced features, such as database connectivity, fare calculation, and user authentication, in future iterations.

### 2. Requirement Analysis

## 2.1. Functional Requirements

## User Features:

## View the list of available buses with details (name, route, type, seats, availability).

## Select starting and destination stops to generate a bus ticket.

## Display error messages for invalid input (e.g., same start and destination stops).

## Admin Features:

## Add new buses with details like name, route, type, and seat availability.

## Dynamically update the bus list after adding new entries.

## Validation and Error Handling:

## Ensure input correctness for ticket generation and bus addition.

## Display error/success messages in the output area or popups.

## 2.2. Non-Functional Requirements

## Usability: Intuitive GUI with dropdowns, buttons, and color-coded messages.

## Performance: Fast response to user actions like viewing buses or generating tickets.

## Portability: Compatible with any Java-supported platform.

## 2.3. System Requirements

## Hardware: 2 GHz processor, 2 GB RAM, 50 MB storage.

## Software: JDK 8+, Swing framework, cross-platform compatibility.

## 2.4. Constraints

## Static bus data stored in memory (ArrayList) without persistence.

## Hardcoded fare structure.

## This concise analysis highlights key features, requirements, and limitations for the system.

## 3. System Design

### 3.1. Flow Chart

## 

## 4.Implementation

## import java.awt.\*;

## import java.util.ArrayList;

## import javax.swing.\*;

## public class BusTicketSystem {

## private static final ArrayList<Bus> buses = new ArrayList<>();

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

## buses.add(new Bus("Bus 101", "A-B-C", "AC", 40, true));

## buses.add(new Bus("Bus 102", "B-C-D-E", "Non-AC", 50, true));

## JFrame frame = new JFrame("Bus Ticket Management");

## frame.setSize(700, 500);

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

## frame.setLayout(new BorderLayout());

## frame.setLocationRelativeTo(null);

## JPanel topPanel = new JPanel(new FlowLayout());

## topPanel.setBackground(new Color(173, 216, 230));

## JTextArea outputArea = new JTextArea(15, 50);

## outputArea.setEditable(false);

## outputArea.setBackground(new Color(245, 245, 245));

## JScrollPane scrollPane = new JScrollPane(outputArea);

## JComboBox<String> startStop = new JComboBox<>(new String[]{"A", "B", "C", "D", "E"});

## JComboBox<String> destStop = new JComboBox<>(new String[]{"A", "B", "C", "D", "E"});

## JButton generateTicket = new JButton("Generate Ticket");

## JButton viewBuses = new JButton("View Buses");

## JButton adminPanel = new JButton("Admin Panel");

## generateTicket.setBackground(new Color(135, 206, 250));

## generateTicket.setForeground(Color.BLACK);

## viewBuses.setBackground(new Color(144, 238, 144));

## viewBuses.setForeground(Color.BLACK);

## adminPanel.setBackground(new Color(255, 182, 193));

## adminPanel.setForeground(Color.BLACK);

## topPanel.add(new JLabel("From:"));

## topPanel.add(startStop);

## topPanel.add(new JLabel("To:"));

## topPanel.add(destStop);

## topPanel.add(generateTicket);

## topPanel.add(viewBuses);

## topPanel.add(adminPanel);

## frame.add(topPanel, BorderLayout.NORTH);

## frame.add(scrollPane, BorderLayout.CENTER);

## viewBuses.addActionListener(e -> {

## outputArea.setForeground(Color.BLUE);

## outputArea.setText(getBusList());

## });

## generateTicket.addActionListener(e -> {

## String start = (String) startStop.getSelectedItem();

## String dest = (String) destStop.getSelectedItem();

## if (start.equals(dest)) {

## outputArea.setForeground(Color.RED);

## outputArea.setText("Error: Starting and destination stops cannot be the same.");

## } else {

## if (buses.isEmpty()) {

## outputArea.setForeground(Color.RED);

## outputArea.setText("No buses available to generate tickets.");

## } else {

## outputArea.setForeground(Color.BLACK);

## outputArea.setText(generateTicket(start, dest, buses.get(0)));

## }

## }

## });

## adminPanel.addActionListener(e -> showAdminPanel(outputArea));

## frame.setVisible(true);

## }

## private static String getBusList() {

## if (buses.isEmpty()) {

## return "No buses available.";

## }

## StringBuilder result = new StringBuilder("------ Available Buses ------\n");

## for (Bus bus : buses) {

## result.append(bus).append("\n");

## }

## result.append("-----------------------------");

## return result.toString();

## }

## private static String generateTicket(String start, String dest, Bus bus) {

## return String.format("""

## \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

## \* BUS TICKET \*

## \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

## \* Bus Name: %s

## \* Bus Type: %s

## \* From: %s

## \* To: %s

## \* Fare: ₹20

## \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

## """, bus.name, bus.type, start, dest);

## }

## private static void showAdminPanel(JTextArea outputArea) {

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

## adminFrame.setSize(400, 300);

## adminFrame.setLayout(new FlowLayout());

## adminFrame.getContentPane().setBackground(new Color(255, 239, 213));

## adminFrame.setLocationRelativeTo(null);

## JTextField busName = new JTextField(10);

## JTextField route = new JTextField(10);

## JTextField type = new JTextField(10);

## JTextField seats = new JTextField(5);

## JTextField available = new JTextField(5);

## JButton addBus = new JButton("Add Bus");

## addBus.setBackground(new Color(173, 255, 47));

## addBus.setForeground(Color.BLACK);

## adminFrame.add(new JLabel("Name:"));

## adminFrame.add(busName);

## adminFrame.add(new JLabel("Route:"));

## adminFrame.add(route);

## adminFrame.add(new JLabel("Type:"));

## adminFrame.add(type);

## adminFrame.add(new JLabel("Seats:"));

## adminFrame.add(seats);

## adminFrame.add(new JLabel("Available:"));

## adminFrame.add(available);

## adminFrame.add(addBus);

## addBus.addActionListener(e -> {

## try {

## String name = busName.getText().trim();

## String routeStr = route.getText().trim();

## String typeStr = type.getText().trim();

## int seatCount = Integer.parseInt(seats.getText().trim());

## boolean isAvailable = Boolean.parseBoolean(available.getText().trim());

## if (name.isEmpty() || routeStr.isEmpty() || typeStr.isEmpty()) {

## throw new IllegalArgumentException("Fields cannot be empty.");

## }

## buses.add(new Bus(name, routeStr, typeStr, seatCount, isAvailable));

## outputArea.setForeground(new Color(34, 139, 34));

## outputArea.setText("Bus added successfully!\n" + name + " -> " + routeStr);

## adminFrame.dispose();

## } catch (Exception ex) {

## JOptionPane.showMessageDialog(adminFrame, "Invalid input. Please check your entries.");

## }

## });

## adminFrame.setVisible(true);

## }

## static class Bus {

## String name, route, type;

## int seats;

## boolean available;

## Bus(String name, String route, String type, int seats, boolean available) {

## this.name = name;

## this.route = route;

## this.type = type;

## this.seats = seats;

## this.available = available;

## }

## @Override

## public String toString() {

## return String.format("%s (%s) - Route: %s, Seats: %d, Available: %s",

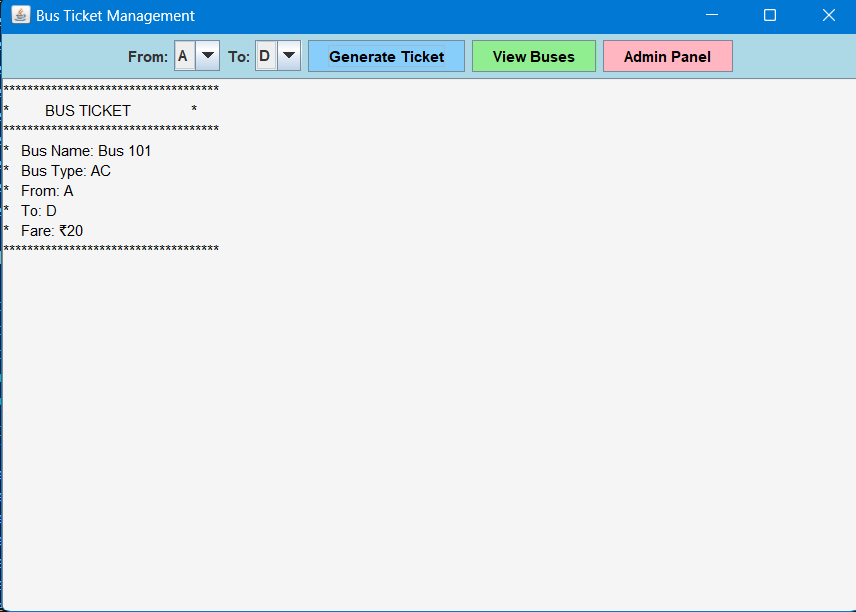
## name, type, route, seats, available ? "Yes" : "No");

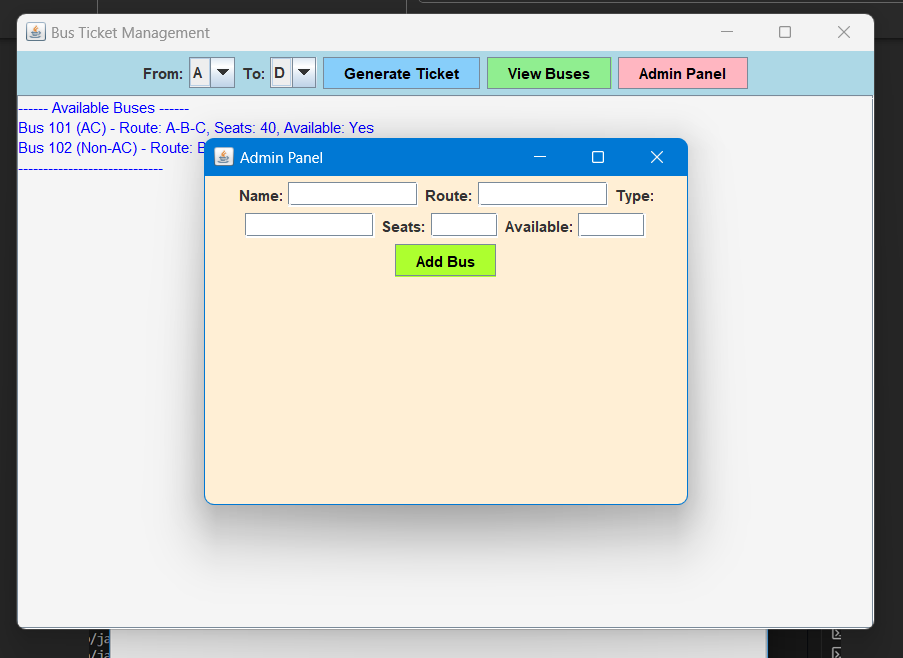
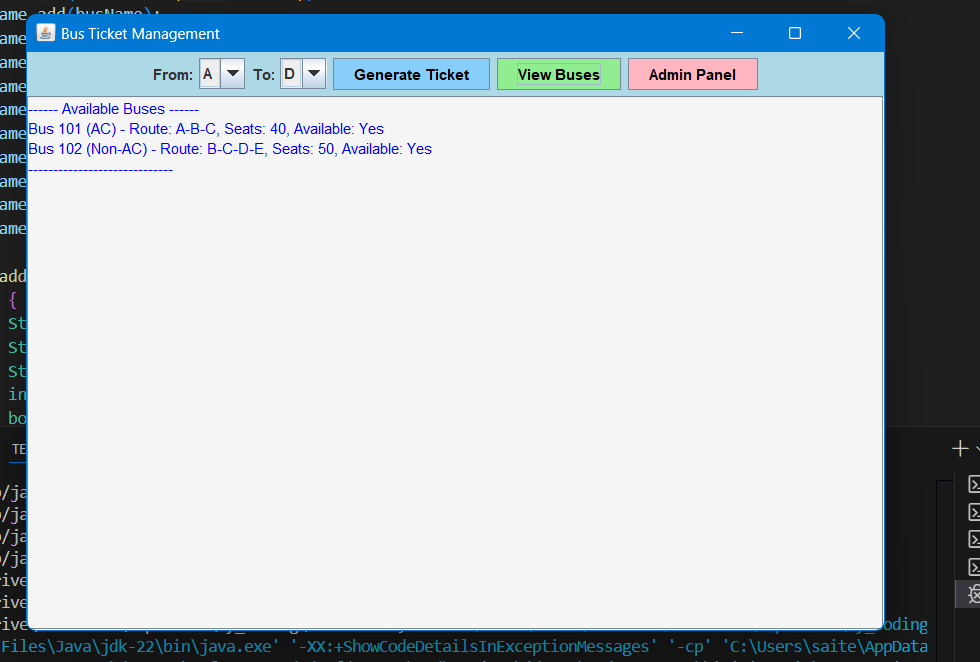
## }

## }

## }

## 5. Results and Discussion





## 

## 6. Conclusion

The **Bus Ticket Management System** developed in Java provides a fundamental structure for managing bus details, ticket generation, and administrative operations in a simple GUI-based application. Key features of the system include:

* **Bus Information Management**: It tracks buses' details, such as their name, route, type (AC or Non-AC), number of seats, and availability.
* **Ticket Generation**: It allows users to generate tickets by selecting a start and destination stop, while validating inputs and ensuring availability.
* **Admin Panel**: A feature that enables an administrator to add new buses with specific details through an intuitive interface.
* **GUI Interface**: The system leverages Swing components for a user-friendly, interactive graphical interface, making it easy for users to interact with the system.

While the system serves as a basic model, it could be extended to incorporate additional features such as advanced fare calculations, dynamic seat booking, and persistence using databases for data storage. The user interface and functionality can also be improved for better usability, especially in handling errors and input validations.

In terms of real-world application, this system serves as a good starting point for creating a bus ticketing application that could scale up to handle more complex functionalities and handle concurrent users effectively.

## 7. References

* **Java Swing Documentation:**
* **The core of the GUI in this system relies on Java Swing, which is part of the Java Foundation Classes (JFC). For detailed documentation on components like JFrame, JButton, JTextArea, and JComboBox, refer to the official Java documentation.**
* [**Java Swing API Documentation**](https://docs.oracle.com/en/java/javase/17/docs/api/java.desktop/javax/swing/package-summary.html)
* **Java Programming Basics:**
* **Understanding of basic Java programming concepts such as classes, objects, event handling, and collections (like ArrayList) is crucial for creating this kind of application.**
* [**Java Programming Basics**](https://www.oracle.com/java/technologies/javase/javase-jdk8-doc-downloads.html)
* **Event-Driven Programming in Java:**
* **The event-driven approach used in the application (with event listeners for buttons and interactions) is a fundamental concept in GUI programming.**
* [**Java Event Handling Guide**](https://docs.oracle.com/javase/tutorial/uiswing/events/)
* **Java Exception Handling:**
* **The code uses exception handling to deal with invalid inputs in the admin panel (e.g., incorrect seat numbers or empty fields). Proper exception handling is essential for building robust applications.**
* [**Java Exception Handling**](https://docs.oracle.com/javase/tutorial/essential/exceptions/)
* **Books & Tutorials:**
* **Head First Java by Kathy Sierra and Bert Bates – A great resource for learning Java programming and object-oriented concepts.**
* **Java: A Beginner's Guide by Herbert Schildt – A comprehensive book for understanding Java programming and Swing GUI development.**
* **By reviewing these references, you can gain a deeper understanding of the individual components that make up the system and explore advanced topics in Java development for building more sophisticated applications.**

1. [↑](#footnote-ref-1)