*Mini project report on*

# Bus Reservation Management System

*Submitted in partial fulfilment of the requirements for the award of degree of*

**Bachelor of Technology**

**in**

**Computer Science & Engineering**

**UE22CS351A – DBMS Project**

***Submitted by:***

| | |
|---|---|
| **Dhanush M** | **PES2UG22CS174** |
| **DVidyasagar** | **PES2UG22CS154** |

Under the guidance of

Dr. Mannar Mannan

Assistant Professor

PES University

**AUG - DEC 2024**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**
**PES UNIVERSITY**

(Established under Karnataka Act No. 16 of 2013)

Electronic City, Hosur Road, Bengaluru – 560 100, Karnataka, India

**PES UNIVERSITY**

(Established under Karnataka Act No. 16 of 2013)
Electronic City, Hosur Road, Bengaluru – 560 100, Karnataka, India

# CERTIFICATE

*This is to certify that the mini project entitled*

***Bus Reservation Management System***

*is a bonafide work carried out by*

| | |
|---|---|
| **Dhanush M** | **PES2UG22CS174** |
| **DVidyaSagar** | **PES2UG22CS154** |

In partial fulfilment for the completion of fifth semester DBMS Project (UE22CS351A) in the Program of Study - Bachelor of Technology in Computer Science and Engineering under rules and regulations of PES University, Bengaluru during the period AUG. 2024 – DEC. 2024. It is certified that all corrections / suggestions indicated for internal assessment have been incorporated in the report. The project has been approved as it satisfies the 5th semester academic requirements in respect of project work.

Signature
Prof. Shilpa S
Assistant Professor

## DECLARATION

We hereby declare that the DBMS Project entitled ***Bus Reservation Management System*** has been carried out by us under the guidance of **Dr Mannar Manna, Assistant Professor** and submitted inpartial fulfilment of the course requirements for the award of degree of **Bachelor of Technology** in **Computer Science and Engineering** of **PES University, Bengaluru** during the academic semester AUG – DEC 2024.

## ABSTRACT

The Bus Reservation Management System is a comprehensive platform that streamlines bus transportation management through digital integration and real-time coordination. It features a dual-interface system where administrators oversee operations while bus operators manage daily activities. The system enables dynamic route management with real-time tracking of seat availability and automated fare calculations. Operators can efficiently process bookings, manage passenger data, and handle ticket modifications.

Key Features:
- Secure authentication system with role-based access control
- Integrated notification system for seamless communication
- Real-time booking management with automated seat allocation
- Complaint tracking system with resolution workflows
- Dynamic route management with fare optimization
- Passenger information management and booking history
- Resource tracking with seat availability monitoring

The platform facilitates transparent communication between administrators and operators through an internal messaging system, while maintaining a structured complaint resolution process. It incorporates automated notifications for booking confirmations and updates, enhancing user experience. The system's robust database integration ensures data integrity and real-time synchronization of bookings, making it a reliable solution for modern bus transportation management.

# TABLE OF CONTENTS

# INTRODUCTION

Bus Reservation Management System: Modernizing Public Transportation Management

Transportation plays a vital role in connecting communities, facilitating commerce, and enabling mobility across regions. Efficient management of bus transportation systems is crucial for ensuring reliable service delivery and passenger satisfaction. The Bus Reservation Management System (BRMS) is a comprehensive digital platform that revolutionizes how bus operations are managed, tickets are booked, and services are monitored.

At its core, the BRMS recognizes that modern bus transportation requires a sophisticated approach that integrates various stakeholders - administrators, operators, and passengers - into a seamless digital ecosystem. By leveraging advanced database management and real-time tracking capabilities, the system enables efficient resource allocation, transparent operations, and enhanced service delivery.

One of the key features of the Bus Reservation Management System is its robust route management functionality. Administrators can create and manage bus routes with detailed information on sources, destinations, distances, and fares. This data is centralized, allowing for real-time tracking of seat availability and dynamic fare calculations, ensuring optimal resource utilization and revenue management.

The BRMS empowers bus operators with tools for efficient ticket management and passenger service. Operators can process bookings, manage passenger information, and handle modifications through an intuitive interface. The system's notification framework ensures seamless communication between administrators and operators, while a structured complaint management system enables quick resolution of service-related issues.

The platform implements a sophisticated security architecture with role-based access control, ensuring that sensitive operations and data are protected. Through SHA-256 encryption and secure database connections, the system maintains the integrity and confidentiality of all transactions and user information.

By integrating route management, ticket booking, complaint handling, and administrative oversight, the Bus Reservation Management System creates a comprehensive ecosystem that fosters efficient operations and enhanced service delivery. This approach aims to modernize bus transportation management, improve operator efficiency, and enhance passenger experience.

Through the BRMS, administrators and operators can collaborate effectively to provide reliable transportation services while maintaining transparency and accountability in operations.

**The Bus Reservation Management System's key features include:**

- Dual-interface system for administrators and operators

- Real-time seat availability tracking and dynamic fare calculation

- Secure user authentication and role-based access control

- Integrated notification system for seamless communication

- Structured complaint management with resolution tracking

- Comprehensive booking management with passenger data handling

- Transparent resource monitoring and route management

By leveraging modern database management and secure transaction processing, the Bus Reservation Management System aims to enhance operational efficiency, strengthen service reliability, and provide a seamless experience for all stakeholders in the bus transportation ecosystem.

# PROBLEM DEFINITION WITH USER REQUIREMENT SPECIFICATIONS

# Bus Reservation Management System
## Problem Definition

In the current transportation landscape, bus operators and passengers face several challenges in managing and accessing bus services efficiently. The key issues include:

1. *Decentralized Booking Management:*
   - Bus operators struggle to manage bookings, routes, and seat availability across different locations
   - Manual booking systems lead to errors and inefficient resource allocation
   - Lack of real-time updates on seat availability causes overbooking issues

2. *Limited Communication Channels:*
   - Passengers lack direct channels to communicate with bus operators
   - Operators struggle to effectively notify passengers about schedule changes or important updates
   - No structured system for handling passenger complaints and feedback

3. *Inefficient Route Management:*
   - Difficulty in managing multiple bus routes, schedules, and fares
   - No centralized system to track route performance and occupancy rates
   - Manual coordination between different operators and routes leads to scheduling conflicts

4. *Inadequate Administrative Oversight:*
   - Limited ability to monitor and manage multiple operators
   - Lack of standardized reporting and management tools
   - Difficulty in maintaining consistent service quality across different operators

## User Requirements Specification

### 1. User Authentication and Management
- *Administrator Requirements:*
  - Secure login system with encrypted password storage
  - Ability to manage bus operators and their accounts
  - Access to system-wide notifications and complaints
  - Authority to create and manage bus routes

- *Operator Requirements:*
  - Separate login portal with role-specific access
  - Profile management with personal and professional details
  - Ability to view assigned routes and manage bookings

### 2. Route Management System
- *Route Creation and Monitoring:*

--Create new bus routes with details (source, destination, distance, duration)
   - Set and update fare prices
   - Track available seats and route performance
   - Monitor route utilization and passenger statistics

 - *Schedule Management:*
   - Set and update departure/arrival times
   - Manage route availability on different days
   - Handle route modifications and cancellations

 ### 3. Booking Management
 - *Ticket Booking:*
   - Real-time seat availability checking
   - Multiple passenger booking capability
   - Fare calculation based on seats and route
   - Collection and verification of passenger details

 - *Booking Modifications:*
   - Ticket cancellation functionality
   - Refund processing system
   - Booking status tracking
   - Seat allocation management

 ### 4. Communication System
 - *Notification Management:*
   - Send targeted notifications to specific users
   - Broadcast system-wide announcements
   - Track notification delivery and reading status
   - Custom message creation for different user groups

 - *Complaint Management:*
   - Structured complaint submission system
   - Complaint tracking and status updates
   - Resolution management workflow
   - Performance tracking of complaint handling

 ### 5. Reporting and Analytics
 - *Administrative Reports:*
   - Route performance analysis
   - Booking statistics and trends
   - Revenue reports and projections
   - Operator performance metrics

*Operator Reports:*
   - Daily booking summaries
   - Route-wise passenger lists
   - Revenue collection reports

  - Complaint resolution statistics

### 6. Security Requirements
- Encrypted password storage using SHA-256
- Secure session management
- Role-based access control
- Data backup and recovery systems

### 7. Technical Requirements
- *Database Management:*
  - MySQL database for data storage
  - Stored procedures for complex operations
  - Data integrity and consistency checks
  - Regular backup procedures

- *User Interface:*
  - Streamlit-based web interface
  - Responsive design for different devices
  - Intuitive navigation and operation
  - Real-time updates and notifications

### 8. Performance Requirements
- *System Performance:*
  - Quick response time for booking operations
  - Real-time seat availability updates
  - Concurrent user support
  - Minimal system downtime

- *Scalability:*
  - Support for multiple routes and operators
  - Expandable database structure
  - Modular system design for future enhancements
  - API support for potential integrations

# **LIST OF SOFTWARES/TOOLS/PROGRAMMING LANGUAGES USED**

# List of Software/Tools/Programming Languages Used

## Front-End Development
- *Streamlit*
  - Python-based web framework for creating interactive user interfaces
  - Used for building the entire front-end interface
  - Provides built-in components for forms, tables, and data visualization
  - Handles session management and user interactions

## Back-End Development
- *Python*
  - Primary programming language for the entire application
  - Handles business logic and data processing
  - Manages database connections and queries
  - Implements security features and authentication

## Database Management
- *MySQL*
  - Relational database management system
  - Stores all application data including:
    - User accounts (administrators and operators)
    - Bus routes and schedules
    - Booking information
    - Complaints and notifications
  - Implements stored procedures for complex operations
  - Ensures data integrity and security

## Development Tools
- *Visual Studio Code*
  - Primary Integrated Development Environment (IDE)
  - Features used:
    - Python code editing and debugging
    - SQL query editing
    - Integrated terminal
    - Source control integration

## Version Control
- *Git*
  - Version control system for tracking code changes
  - Enables collaborative development
  - Maintains project history and documentation

## Key Libraries and Dependencies
- *mysql-connector-python*
  - Python library for MySQL database connectivity
  - Handles database operations and stored procedure calls

- *hashlib*
  - Python standard library for password encryption
  - Implements SHA-256 hashing for secure password storage

- *datetime*
  - Python standard library for date and time operations
  - Handles booking dates and timestamp management

- *typing*
  - Python standard library for type hints
  - Improves code readability and maintainability

## Programming Languages
1. *Python*
   - Version: 3.x
   - Primary language for application logic
   - Used for both frontend (Streamlit) and backend development

2. *SQL*
   - Used for database operations
   - Implements stored procedures and complex queries
   - Handles data manipulation and retrieval

## Development Requirements
- Python 3.x installation
- MySQL Server installation
- Required Python packages:

```
  streamlit
  mysql-connector-python
```

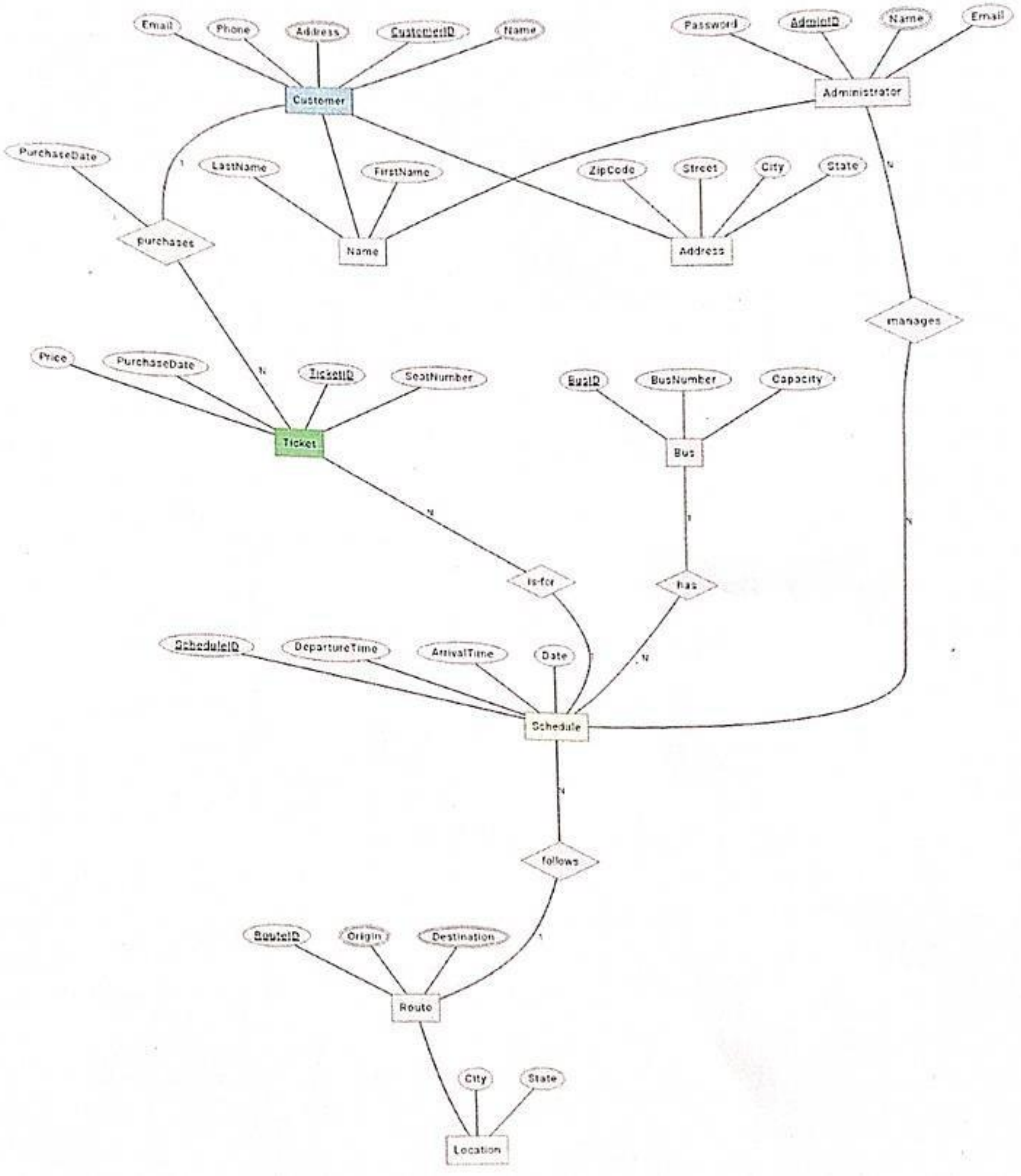
- MySQL Workbench (recommended for database management)

## Security Tools

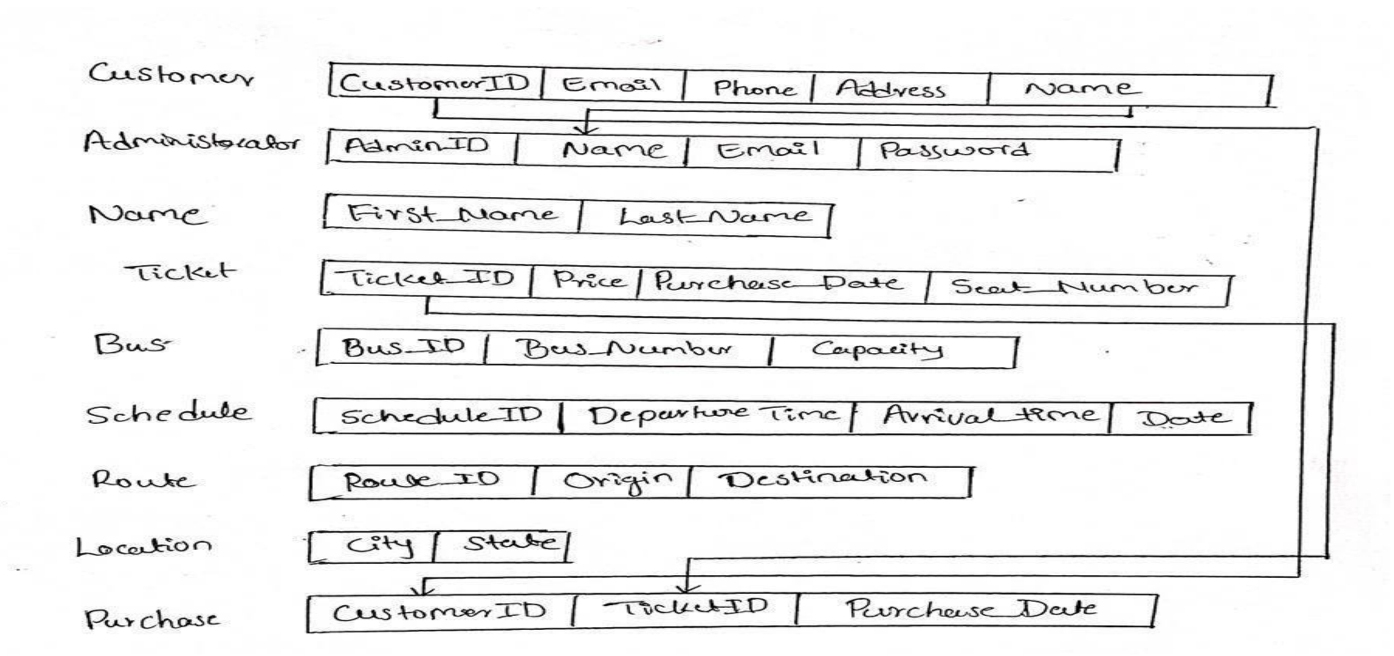
- SHA-256 encryption for password handling
- MySQL built-in security features
- Session management through Streamlit

This setup differs from your friend's project in several key ways:
1. Instead of React/Node.js, your system uses Python/Streamlit
2. Rather than separate frontend/backend frameworks, you have an integrated solution
3. While both use MySQL, your implementation leverages stored procedures extensively
4. Your system uses Python's built-in libraries for many features instead of external frameworks

## ER Diagram:


Email
Phone
Address
CustomerID
Name
Customer
Password
AdminID
Name
Email
Administrator
PurchaseDate
LastName
FirstName
ZipCode
Street
City
State
purchases
Name
Address
manages
Price
PurchaseDate
TicketID
SeatNumber
BusID
BusNumber
Capacity
Ticket
Bus
is-for
has
ScheduleID
DepartureTime
ArrivalTime
Date
Schedule
follows
RouteID
Origin
Destination
Route
City
State
Location
1
N
N
N
1
N
N
1
N
1

**ER TO RELATIONAL MAPPING:**



::

## DDL Statements

### Create Statements:

```
-- Users table
CREATE TABLE users (
    User_ID INT PRIMARY KEY AUTO_INCREMENT,
    Username VARCHAR(50) NOT NULL UNIQUE,
    email VARCHAR(100) NOT NULL UNIQUE,
    password VARCHAR(100) NOT NULL,
    Role ENUM('user', 'operator', 'conductor') NOT NULL,
    Created_At TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);


-- Bus Routes table
CREATE TABLE bus_routes (
    Route_ID INT PRIMARY KEY AUTO_INCREMENT,
    RouteName VARCHAR(100) NOT NULL,
    Source VARCHAR(100) NOT NULL,
    Destination VARCHAR(100) NOT NULL,
    Distance VARCHAR(50) NOT NULL,
    Duration VARCHAR(50) NOT NULL,
    Fare DECIMAL(10,2) NOT NULL,
    Available_Seats INT DEFAULT 30
);


-- Administrators table
CREATE TABLE Administrators (
    Admin_ID INT PRIMARY KEY AUTO_INCREMENT,
    Username VARCHAR(50) NOT NULL UNIQUE,
    Password VARCHAR(256) NOT NULL,
    Role VARCHAR(20) NOT NULL
);
```

```
-- Operators table
CREATE TABLE Operators (
    Operator_ID INT PRIMARY KEY AUTO_INCREMENT,
    Username VARCHAR(50) NOT NULL UNIQUE,
    Password VARCHAR(100) NOT NULL,
    First_Name VARCHAR(50) NOT NULL,
    Last_Name VARCHAR(50) NOT NULL,
    Email VARCHAR(100) NOT NULL UNIQUE,
    Status ENUM('Active', 'Inactive') NOT NULL
);


-- Notifications table (updated with is_read flag)
CREATE TABLE notifications (
    Notification_ID INT PRIMARY KEY AUTO_INCREMENT,
    User_ID INT NOT NULL,
    Message TEXT NOT NULL,
    is_read BOOLEAN DEFAULT FALSE,
    Created_At TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (User_ID) REFERENCES users(User_ID)
);


-- Complaints table
CREATE TABLE complaints (
    Complaint_ID INT PRIMARY KEY AUTO_INCREMENT,
    Operator_ID INT NOT NULL,
    Subject VARCHAR(200) NOT NULL,
    Message TEXT NOT NULL,
    Status ENUM('Pending', 'In Progress', 'Resolved') DEFAULT 'Pending',
    Created_At TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (Operator_ID) REFERENCES Operators(Operator_ID)
);


-- Tickets table
CREATE TABLE tickets (
    Ticket_ID INT PRIMARY KEY AUTO_INCREMENT,
    Route_ID INT NOT NULL,
    Passenger_Name VARCHAR(100) NOT NULL,
    Passenger_Email VARCHAR(100) NOT NULL,
    Passenger_Phone VARCHAR(20) NOT NULL,
    Booking_Date DATE NOT NULL,
    Number_Of_Seats INT NOT NULL,
    Total_Fare DECIMAL(10,2) NOT NULL,
    Status ENUM('Booked', 'Cancelled') DEFAULT 'Booked',
    Created_At TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (Route_ID) REFERENCES bus_routes(Route_ID)
);
```

**DML STATEMENTS (CRUD OPERATION SCREENSHOTS)**

## --1.Insert Operation

```
INSERT INTO users (Username, email, password, Role)
VALUES
    ('testuser1', 'test1@example.com', 'testpass123', 'user'),
    ('operator1', 'operator1@example.com', 'testpass123', 'operator'),
    ('conductor1', 'conductor1@example.com', 'testpass123', 'conductor');
```

**-- 2.SELECT**
to get tickets for a route

```
SELECT * FROM tickets
   WHERE Route_ID = p_Route_ID
   ORDER BY Created_At DESC;
```

***-- 3 UPDATE operation***
*. CancelTicket procedure*

```
UPDATE tickets
SET Status = 'Cancelled'
WHERE Ticket_ID = p_Ticket_ID;
```

**--4 DELETE**

```
cursor.execute("DELETE FROM bus_routes WHERE Route_ID = %s", (route_id,))
```

## -- 5 Nested Query:

```
cursor.execute("DELETE FROM bus_routes WHERE Route_ID = %s", (route_id,))
```

**QUERIES (JOIN QUERY, AGGREGATE FUNCTION QUERIES AND NESTED QUERY)**

**JOIN Queries:**

```
-- From GetComplaintList procedure
SELECT c.*, o.Username as Operator_Name
FROM complaints c
JOIN Operators o ON c.Operator_ID = o.Operator_ID;

-- From GetUserNotifications procedure
SELECT n.*, u.Username as Recipient
FROM notifications n
JOIN users u ON n.User_ID = u.User_ID
WHERE n.User_ID = p_User_ID
ORDER BY n.Created_At DESC;
```

**Aggregate Function:**

```
-- From the ticket booking logic (inside BookTicket procedure)
SELECT Available_Seats, Fare INTO v_available_seats, v_fare
FROM bus_routes WHERE Route_ID = p_Route_ID;

-- From operator registration check (inside RegisterOperator procedure)
SELECT COUNT(*) INTO user_exists
FROM Operators
WHERE Username = p_Username OR Email = p_Email;

-- From operator login check (inside CheckOperatorLogin procedure)
SELECT COUNT(*) INTO user_exists
FROM Operators
WHERE Username = p_Username
AND Password = p_Password
AND Status = 'Active';
```

## STORED PROCEDURE, FUNCTIONS AND TRIGGERS

**Functions/Procedures:**

```
-- Procedure to register new operator
DELIMITER //
CREATE PROCEDURE RegisterOperator(
IN p_Username VARCHAR(50),
IN  p_Password  VARCHAR(256),
IN p_First_Name VARCHAR(50),
IN p_Last_Name VARCHAR(50),
IN p_Email VARCHAR(100)
)
BEGIN
DECLARE user_exists INT;

SELECT COUNT(*) INTO user_exists
FROM Operators
WHERE Username = p_Username OR Email = p_Email;

IF user_exists = 0 THEN
INSERT INTO Operators (
Username, Password, First_Name, Last_Name, Email, Status
) VALUES (
p_Username, p_Password, p_First_Name, p_Last_Name, p_Email, 'Active'
);
SELECT 'SUCCESS' as result;
ELSE
SELECT 'DUPLICATE' as result;
END IF;
END //
DELIMITER ;
```

**Trigger:**

```
-- 1. Trigger to log ticket bookings and update notification
DELIMITER //
CREATE TRIGGER after_ticket_booking
AFTER INSERT ON tickets
FOR EACH ROW
BEGIN
    -- Log the ticket booking
    INSERT INTO audit_log (Table_Name, Action_Type, Record_ID, Action_Details, User_Info)
    VALUES ('tickets', 'INSERT', NEW.Ticket_ID,
            CONCAT('New ticket booked for route ', NEW.Route_ID,
                   ' with ', NEW.Number_Of_Seats, ' seats'),
            NEW.Passenger_Email);

    -- Create notification for the booking
    INSERT INTO notifications (User_ID, Message)
    SELECT User_ID, CONCAT('New ticket booking: ', NEW.Passenger_Name,
                          ' booked ', NEW.Number_Of_Seats, ' seats')
    FROM users
    WHERE Role = 'operator';
END //
DELIMITER ;
```

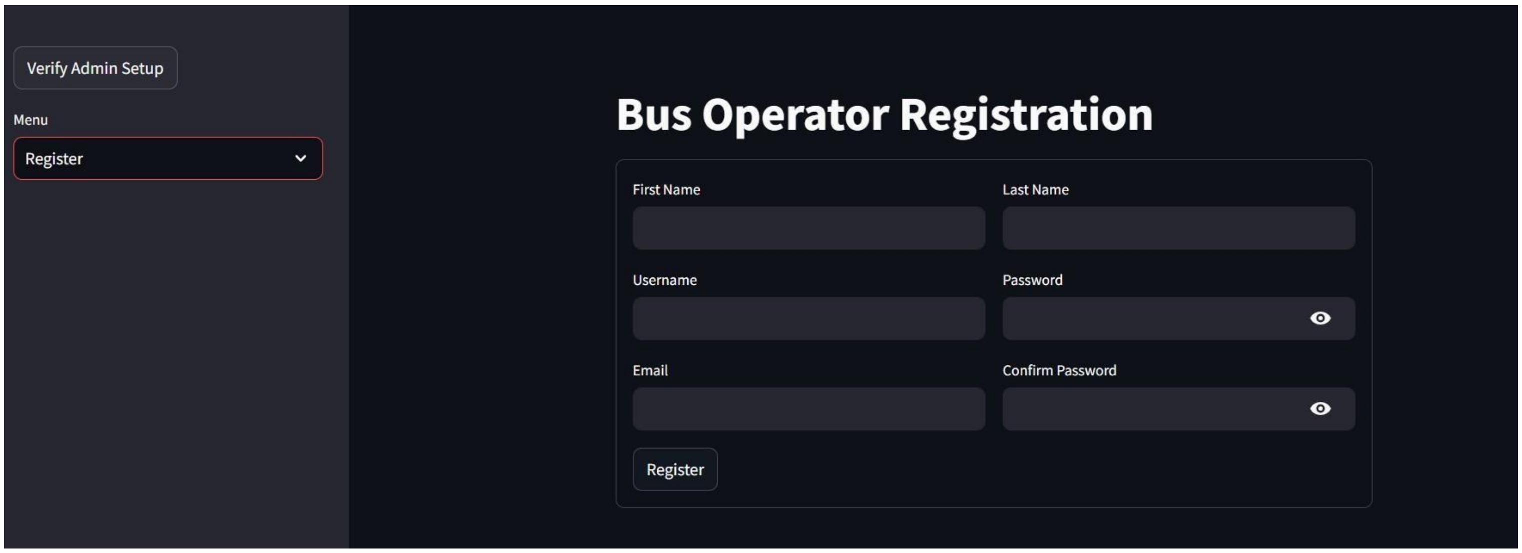
```
-- 2. Trigger to log and notify ticket cancellations
DELIMITER //
CREATE TRIGGER after_ticket_cancellation
AFTER UPDATE ON tickets
FOR EACH ROW
BEGIN
    IF NEW.Status = 'Cancelled' AND OLD.Status = 'Booked' THEN
        -- Log the cancellation
        INSERT INTO audit_log (Table_Name, Action_Type, Record_ID, Action_Details, User_Info)
        VALUES ('tickets', 'CANCEL', NEW.Ticket_ID,
                CONCAT('Ticket cancelled for route ', NEW.Route_ID,
                       ' with ', NEW.Number_Of_Seats, ' seats'),
                NEW.Passenger_Email);

        -- Create notification for the cancellation
INSERT INTO notifications (User_ID, Message)
        SELECT User_ID, CONCAT('Ticket cancellation: Ticket #', NEW.Ticket_ID,
                              ' has been cancelled')
        FROM users
                  WHERE Role =
    'operator';END IF;
END //
```

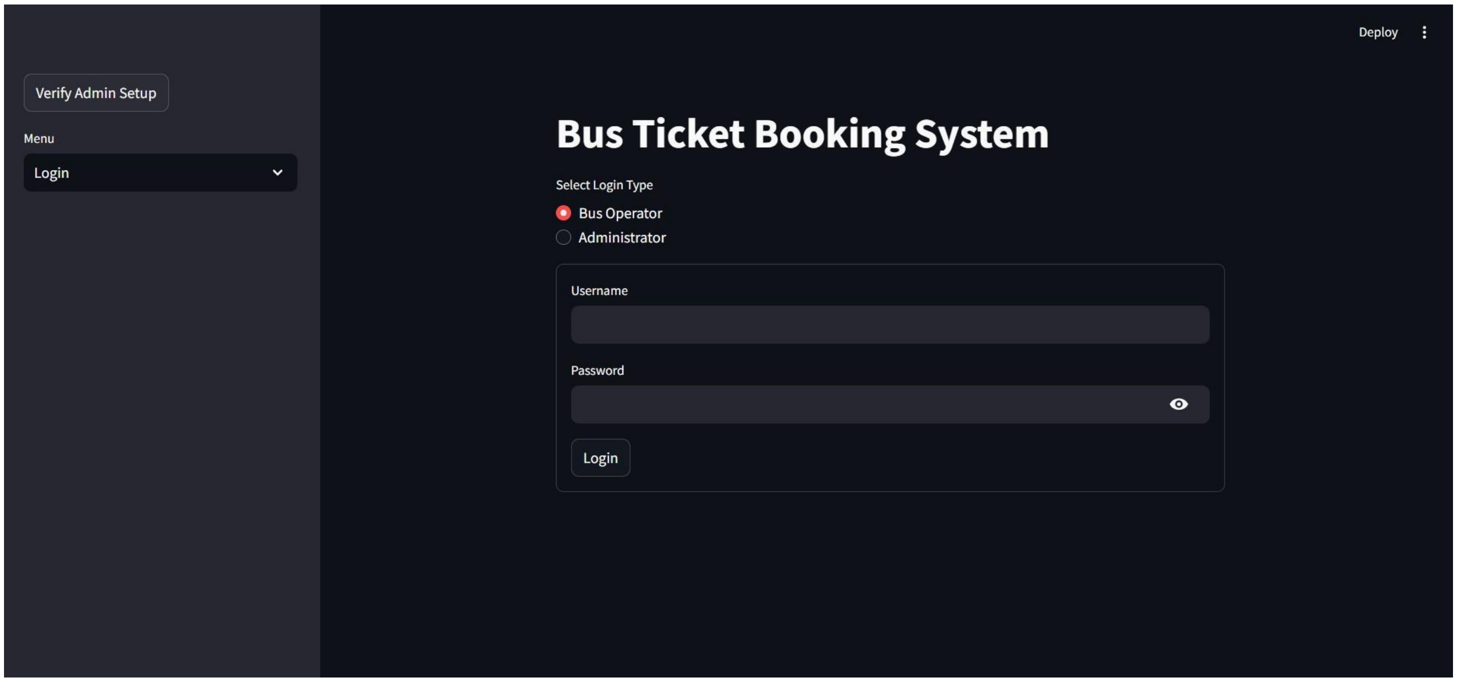
## RONT END DEVELOPMENT (FUNCTIONALITIES/FEATURES OF THE APPLICATION)

**GitHub link:**

**https://github.com/HegdeDhanush/Bus-Reservation-Management-System**

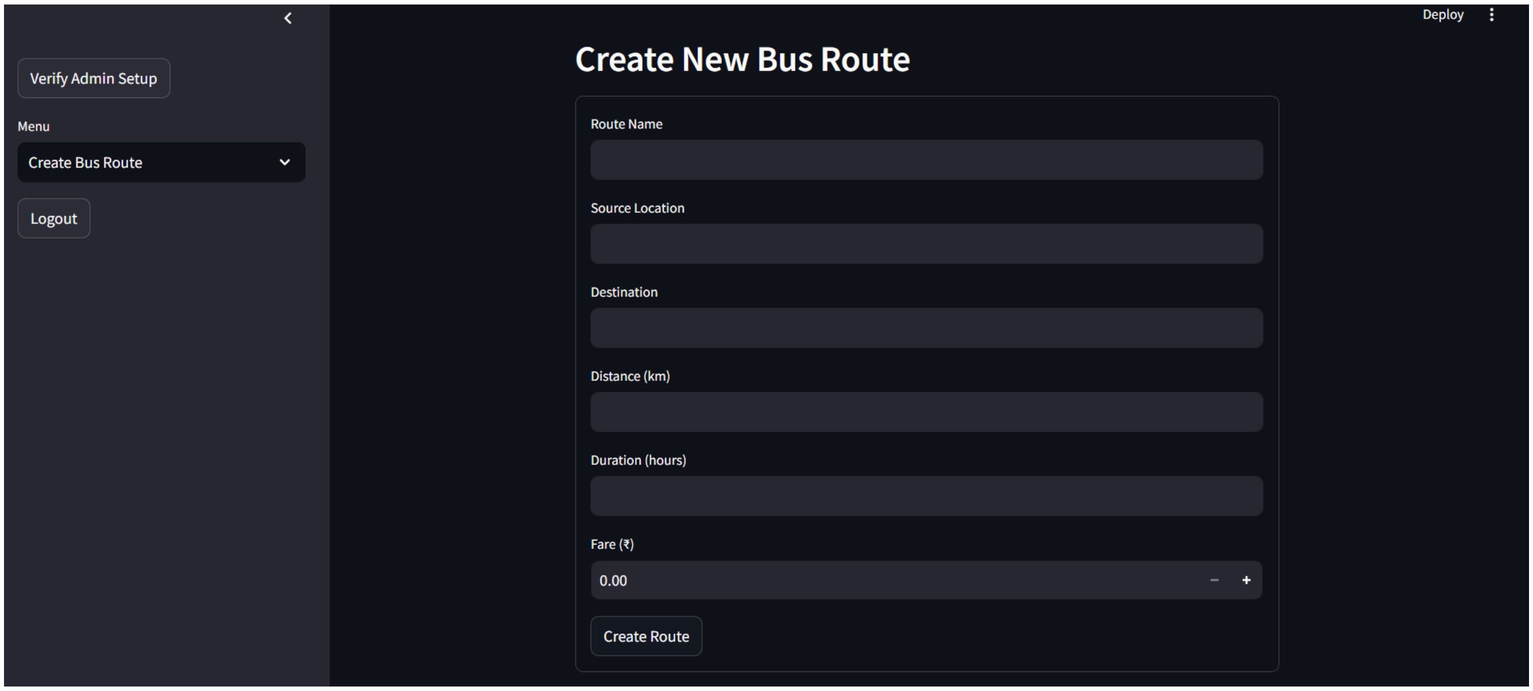
**Login and signup: (Insert)**





**Functionalities of Admin:**

**Create Bus Route:**



Create New Bus Route

Route Name

Source Location

Destination

Distance (km)

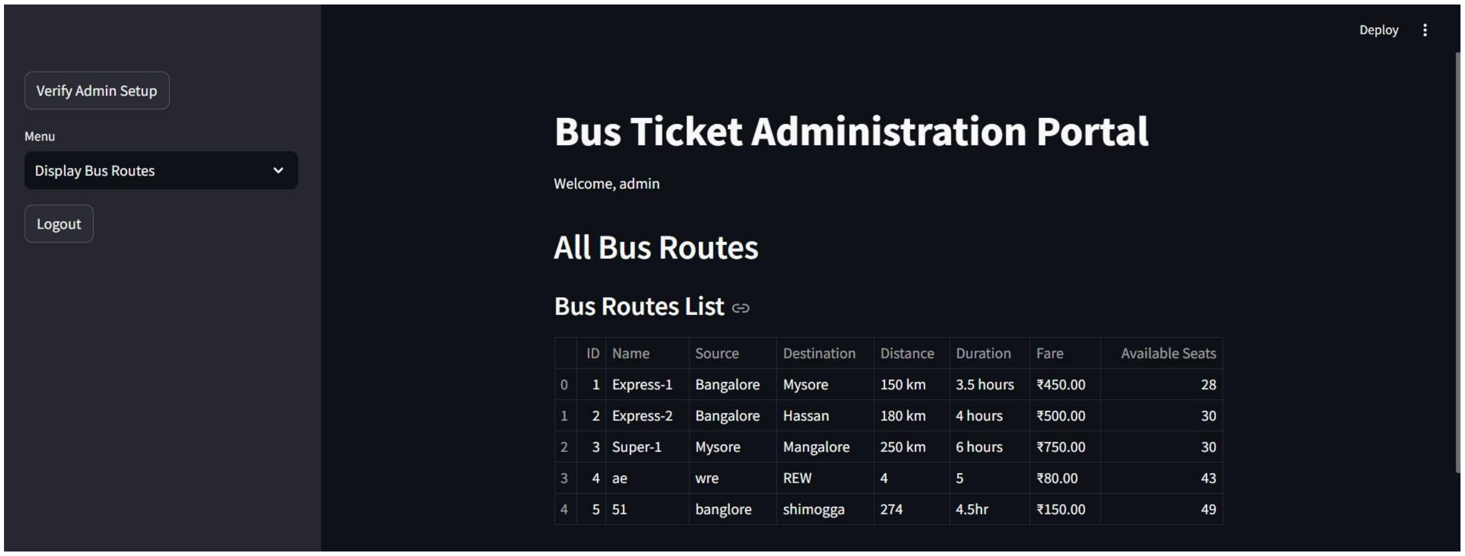
Duration (hours)

Fare (₹)

0.00

Create Route

**Display Bus Routes:**



Bus Ticket Administration Portal

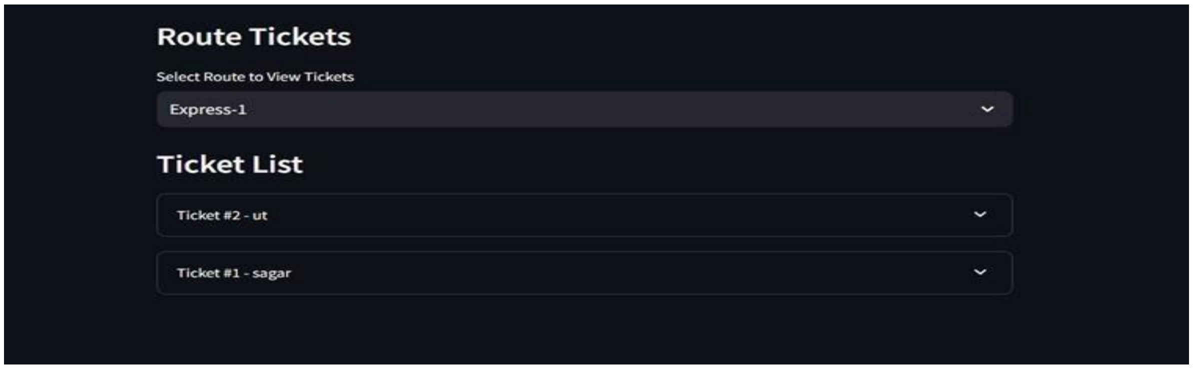
Welcome, admin

All Bus Routes

Bus Routes List

| | ID | Name | Source | Destination | Distance | Duration | Fare | Available Seats |
|---|---|---|---|---|---|---|---|---|
| 0 | 1 | Express-1 | Bangalore | Mysore | 150 km | 3.5 hours | ₹450.00 | 28 |
| 1 | 2 | Express-2 | Bangalore | Hassan | 180 km | 4 hours | ₹500.00 | 30 |
| 2 | 3 | Super-1 | Mysore | Mangalore | 250 km | 6 hours | ₹750.00 | 30 |
| 3 | 4 | ae | wre | REW | 4 | 5 | ₹80.00 | 43 |
| 4 | 5 | 51 | banglore | shimogga | 274 | 4.5hr | ₹150.00 | 49 |

**Route Tickets:**


Route Tickets

Select Route to View Tickets

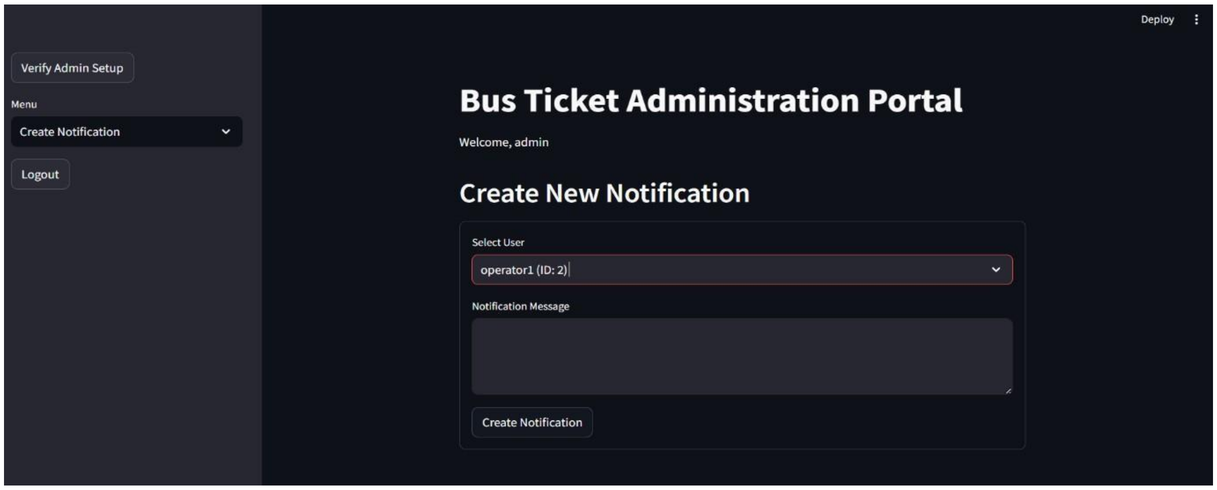
Express-1

Ticket List

Ticket #2 - ut

Ticket #1 - sagar

**Create Notifications:**



Bus Ticket Administration Portal

Welcome, admin

Create New Notification

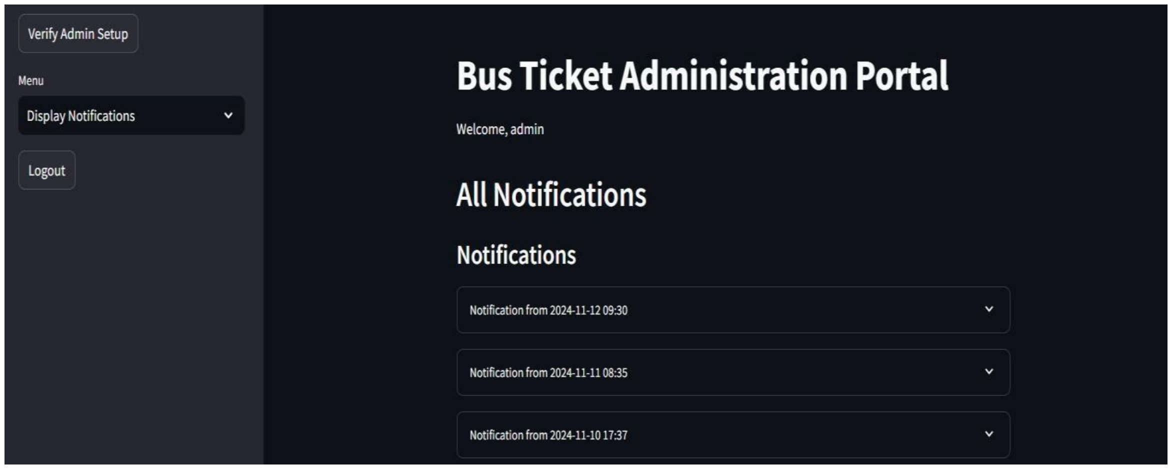
Select User

operator1 (ID: 2)

Notification Message

Create Notification

**Display notification:**



Bus Ticket Administration Portal

Welcome, admin

All Notifications

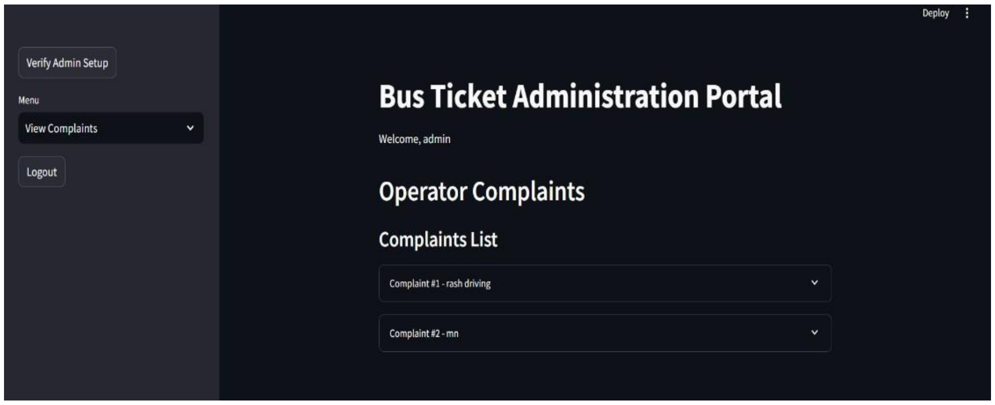
Notifications

Notification from 2024-11-12 09:30

Notification from 2024-11-11 08:35

Notification from 2024-11-10 17:37

## View Complaints:



Bus Ticket Administration Portal

Welcome, admin

Operator Complaints

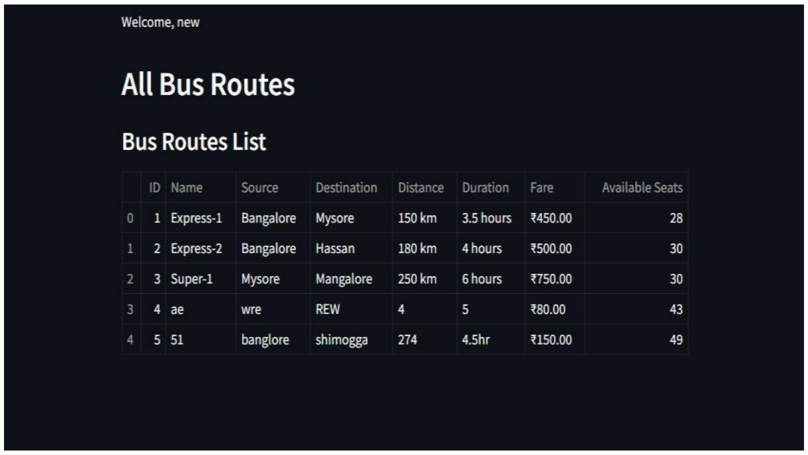
Complaints List

Complaint #1 - rash driving

Complaint #2 - mn

## Functionalities of Operator:

### View Routes:



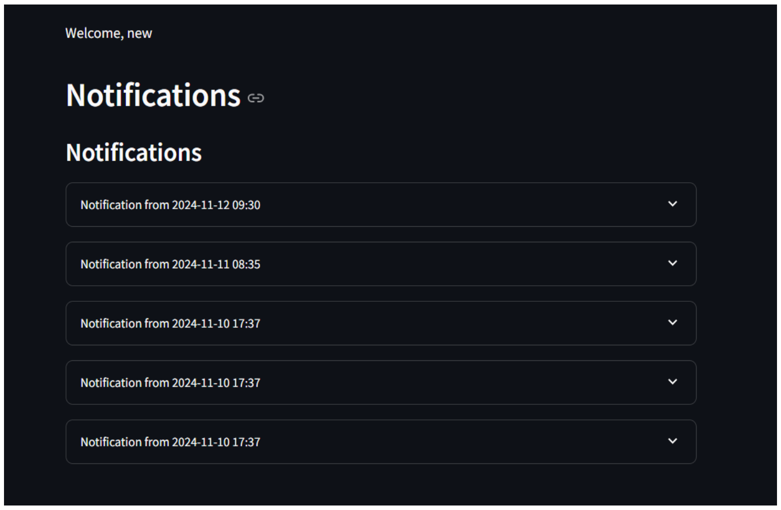
Welcome, new

All Bus Routes

Bus Routes List

| | ID | Name | Source | Destination | Distance | Duration | Fare | Available Seats |
|---|---|---|---|---|---|---|---|---|
| 0 | 1 | Express-1 | Bangalore | Mysore | 150 km | 3.5 hours | ₹450.00 | 28 |
| 1 | 2 | Express-2 | Bangalore | Hassan | 180 km | 4 hours | ₹500.00 | 30 |
| 2 | 3 | Super-1 | Mysore | Mangalore | 250 km | 6 hours | ₹750.00 | 30 |
| 3 | 4 | ae | wre | REW | 4 | 5 | ₹80.00 | 43 |
| 4 | 5 | 51 | banglore | shimogga | 274 | 4.5hr | ₹150.00 | 49 |

### View Notifications



Welcome, new

Notifications

Notifications

Notification from 2024-11-12 09:30

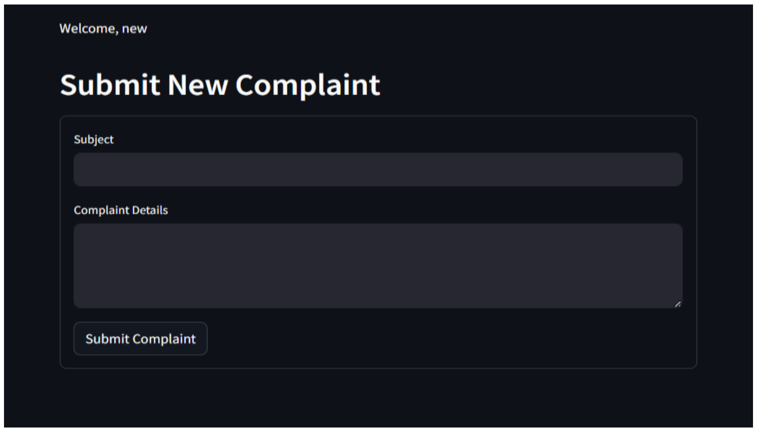
Notification from 2024-11-11 08:35

Notification from 2024-11-10 17:37

Notification from 2024-11-10 17:37

Notification from 2024-11-10 17:37

**Submit Complaint:**

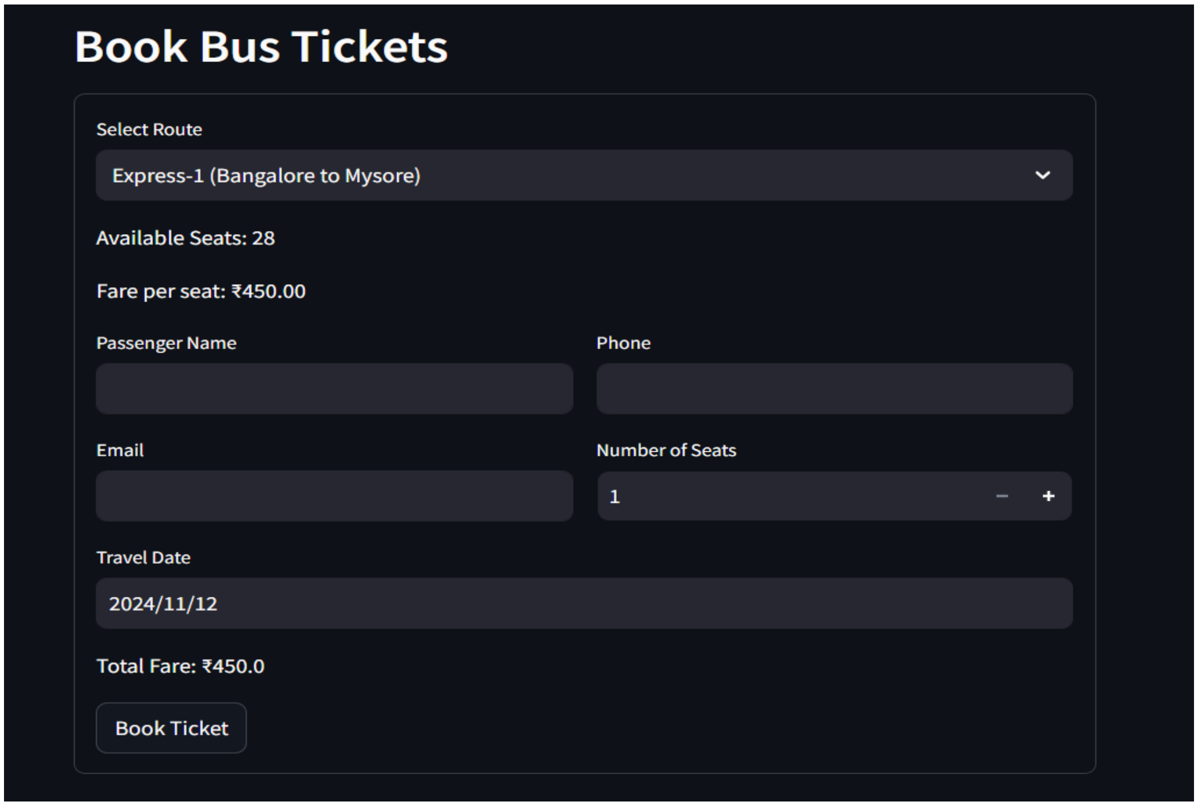

Welcome, new

Submit New Complaint

Subject

Complaint Details

## Book Tickets:


Book Bus Tickets

Select Route

Express-1 (Bangalore to Mysore)

Available Seats: 28

Fare per seat: ₹450.00

Passenger Name

Phone

Email

Number of Seats

1

Travel Date

2024/11/12

Total Fare: ₹450.0

## REFERENCES/BIBLIOGRAPHY

1. www.google.com
2. www.youtube.com
3. https://dev.mysql.com/doc/
4. https://www.geeksforgeeks.org/

**GitHub link:**

**https://github.com/HegdeDhanush/Bus-Reservation-Management-System**