Project Report

Lahore Orange Line Metro Train Management System

## Prepared by: Zaneeha Afzal, Mahnoor Aslam

## Roll Numbers: Bsdsf23a011, bsdsf23a037

## Date: June 2025

## Submitted to: Dr. Khurram Shahzad

## Course: Database

## Project Title: Orange Line Metro Train Management System

# Project Type

Web-based Metro Train Ticketing and Administration System

Built using:

- PHP (Backend)  
- SQLite (Database)  
- HTML/CSS + JavaScript (Frontend)  
- Font Awesome (Icons)

# Objective

To provide an intuitive, responsive, and role-based interface for:  
- Users to search trains, view available schedules, calculate fare, and book tickets.  
- Admins to monitor system-wide statistics, manage trains, schedules, users, and inquiries with live reporting.

# Modules Overview

## 1. User ****Functionalities****

#### ****Purpose:****

Allows registered users to view and manage their metro travel activities.

#### ****Key Functionalities:****

* Displays logged-in user’s name using PHP sessions.
* Statistics:
  + Total Booked Tickets
  + Used Tickets
  + Cancelled Tickets
* Train Search & Filter Options by From, To, and Travel Date.
* Real-time Train Schedule Table from joined database tables.
* Distance & Fare Calculation using Haversine formula and fare\_per\_km.
* Booking Feature with a popup form and booking logic.
* Security via session checks and SQLite PDO usage.

## 2.A****dmin Functionalities****

The admin panel is designed to help system administrators monitor, manage, and maintain all aspects of the metro train system.

### ****1. Dashboard Overview****

* View real-time system statistics:
  + **Total Trains** in the system.
  + **Daily Passengers** (tickets booked today).
  + **Pending Inquiries** from users.
  + **Maintenance Train Count** to monitor issues.
* **System Status Indicator**:
  + If trains are under maintenance, status shows **“Some issues”** (orange).
  + Otherwise, status shows **“Operational”** (green).

### ****2. Recent Activity Logs****

* Displays the latest activities using complex SQL UNION ALL queries from multiple tables:
  + **Train Updated** – Shows who updated a train and when.
  + **Schedule Modified** – Logs schedule edits with admin name and time.
  + **Ticket Booked** – Shows ticket booking activity of users.
  + **New Inquiry** – Displays latest open inquiries submitted by users.

### ****3. Train Management****

* Add new trains with relevant data (train name, status).
* Edit existing train details.
* Mark trains as under maintenance.

### ****4. Schedule Management****

* View and update train schedules.
* Modify arrival/departure times and assign stations to trains.
* Ensures proper timing between consecutive trains.

### ****5. Inquiry Management****

* View all **user-submitted inquiries** from the database.
* Filter by inquiry status (e.g., open, closed).
* **Reply to inquiries** and update their status to “closed”.

### ****6. User Management****

* View all registered users in the system.
* **Delete suspicious or inactive users** from the database.
* Optionally extend to edit user roles in future.

### ****7. Secure Admin Access****

* Static login with predefined “Admin” session (extendable to role-based login).
* All database operations use secure **SQLite with PDO**.

### ****8. Navigation & Interface****

* Intuitive **sidebar navigation** with links to:
  + Dashboard (Overview)
  + Train Management
  + Schedule Management
  + Inquiries
  + User Management
  + Logout
* Modern UI using HTML, CSS, FontAwesome, and custom styles.

# Database Structure

### 1. ****Users****

* user\_id: Unique ID for each user (Primary Key)
* username: User's unique name
* password: Encrypted user password
* email: Unique email address
* role: 'admin' or 'user'
* created\_at: Timestamp of account creation
* last\_login: Last login datetime

Relationships:

* Can create Trains
* Can update Schedules
* Can book Tickets
* Can submit/respond to Inquiries

### 2. ****Stations****

* station\_id: Unique ID for each station (Primary Key)
* station\_name: Name of the station
* latitude: Geographical latitude
* longitude: Geographical longitude
* address: Optional physical address
* status: 'active', 'under construction', or 'closed'

Relationships:

* Linked as origin and destination in Schedules

### 3. ****Trains****

* train\_id: Unique train identifier (Primary Key)
* train\_name: Name of the train
* capacity: Number of passengers the train can carry
* status: 'active', 'maintenance', or 'retired'
* created\_by: Admin user who added the train
* created\_at: Timestamp of creation
* updated\_at: Timestamp of last update

Relationships:

* Has many Schedules
* M:N with Schedules via train\_schedules

### 4. ****Schedules****

* schedule\_id: Unique ID for each schedule (Primary Key)
* train\_id: Associated train
* from\_station: Starting station ID
* to\_station: Ending station ID
* departure\_time: Departure time (text)
* arrival\_time: Arrival time (text)
* frequency: 'daily', 'weekdays', or 'weekends'
* status: 'active' or 'cancelled'
* updated\_by: User who last updated the schedule
* updated\_at: Timestamp of last update

Relationships:

* Linked to Trains, Stations, Users
* Referenced by Tickets

### 5. ****Tickets****

* ticket\_id: Unique ticket ID (Primary Key)
* user\_id: User who booked the ticket
* schedule\_id: Schedule on which ticket is booked
* seat\_number: Allocated seat
* booking\_time: Booking timestamp
* travel\_date: Date of travel
* status: 'confirmed', 'cancelled', or 'used'
* price: Fare calculated for the trip

Relationships:

* Linked to Users and Schedules

### 6. ****Inquiries****

* inquiry\_id: Unique inquiry ID (Primary Key)
* user\_id: User who submitted the inquiry
* subject: Inquiry subject
* message: Full message content
* status: 'open', 'answered', or 'closed'
* created\_at: Timestamp of inquiry
* answered\_by: Admin who responded
* answer: Admin’s reply
* answered\_at: Reply timestamp

Relationships:

* Linked to both user and admin accounts

### 7. ****Train\_Schedules (Join Table)****

* train\_id: Train involved (FK)
* schedule\_id: Schedule involved (FK)

Purpose:

* To support many-to-many relationship between Trains and Schedules

### 8. ****MetroFares****

* id: Unique ID (Primary Key)
* min\_distance: Minimum distance of the slab
* max\_distance: Maximum distance of the slab
* fare: Fare charged for that distance range

Purpose:

* Used for calculating ticket price using the **Haversine formula** based on station coordinates.

## Tools and Technologies

|  |  |
| --- | --- |
| Technology | Description |
| PHP | Backend scripting language |
| SQLite | Lightweight relational database |
| HTML/CSS | Frontend structure and styling |
| JavaScript | Dynamic frontend behavior |
| Bootstrap | Responsive design (optional) |
| Session Handling | User login and dashboard control |

# Special Features

- Real-time metrics for admin panel  
- Distance-based fare using GPS formula  
- Responsive UI for both user/admin  
- Secure session management  
- Live SQL updates for activity feed  
- Color-coded status indicators

# Future Improvements

- Implement login session role checks for admin  
- Prevent duplicate seat bookings  
- Paginate train schedule results  
- Add user profile view/edit  
- Add CRUD for trains/schedules/inquiries

# Github Link:

( <https://github.com/Neeha2005/LahoreOrangeLineManagementSystem> )

# Conclusion

The Lahore Orange Line Metro Train Management System successfully provides essential train booking and monitoring functionality through a well-structured web interface. With separate dashboards for users and administrators, it ensures a seamless user experience and powerful backend control. The use of SQLite makes it lightweight and portable, ideal for academic or small-scale production deployments.