VNU-HCM International University

TRAIN TICKET MANAGEMENT

Final Project Report

Subject:	Web Application Development
Instructor:	Nguyen Trung Nghia

Group Members:

Le Nhat Duy - ITITWE22143

Nguyen Dinh Minh Quan – ITITWE22178

December 2025

Contents

I. Introduction

1. Project Overview

The Railway Reservation System is a web application that allows users to search for trains, book tickets, enter passenger information, choose a payment method (QR/COD/Card), and view booked tickets. The system uses a MySQL database to store information about stations, train schedules, seat classes, available seats, reservations (PNR), passenger details, and payment status.

2. Objectives

- Provide a complete booking flow from Station A → Station B by selected date.
- Manage remaining seats based on seat class and route.
- Generate a unique PNR for each reservation.
- Integrate a basic payment flow:
  - QR payment is shown using the internal project image qr.png (not an external URL).
  - Save payment records into the payments table.

    - Allow users to view tickets (My Tickets) and display Payment Status.

- Support ticket cancellation and automatically restore seats / compute refunds using database triggers.

3. Scope

- User side: Search → Booking → Passenger Details → Payment → My Tickets → Cancel Ticket
- Admin side (if implemented): Manage train/station/schedule/classseats data

4. Technologies Used

- Frontend: HTML/CSS, Bootstrap 5, Font Awesome, lightweight JavaScript
- Backend: PHP (session-based), MySQLi
- Database: MySQL (Triggers for cancellation logic)
- Local environment: XAMPP / phpMyAdmin

II. Requirement Analysis

1. Actors

- Registered User: Book tickets, view tickets, cancel tickets, and track payment status.
- Admin: Manage master data (train, station, schedule, class seats).

2. Use Cases & Functional Requirements

UC01 – Search Train (Enquiry)

User story: As a user, I want to select departure station, destination station, and journey date to search for available trains.

Functional Requirements:

- Display station list (from station table).
- User selects sp, dp, doj.
- Return matching train results with departure/arrival time, class, fare, and remaining seats.

UC02 – View Search Result

Functional Requirements:

- Show result table: Train No, Name, Departure, Arrival, Class, Fare, Seats Left.
- Allow user to choose Train + Class and number of seats to book.

UC03 – Booking (Create Reservation)

Functional Requirements:

- Validate booking rules (e.g., at least one adult ≥ 18).
- Create a reservation record in resv (PNR auto-generated).
- Decrease seatsleft in classseats.

UC04 – Add Passenger Details

Functional Requirements:

- Input passenger list based on nos.
- Save to pd(pnr, pname, page, pgender).

UC05 – Payment (QR / COD / Card)

Functional Requirements:

- User selects payment method.
- If QR, display internal image qr.png from the project folder.
- Save payment record into payments (pnr, method, status=PENDING/PAID, txn_ref, …).

UC06 – My Tickets

Functional Requirements:

- User logs in using mobile and password.
- Display bookings from resv.
- Display payment status using a JOIN to payments.

UC07 – Cancel Ticket

Functional Requirements:

- User enters PNR to cancel (must belong to the logged-in user).
- Update resv.status = 'CANCELLED'.
- Trigger automatically handles:
    - Block cancellation if journey date has passed.
    - Restore seats to classseats.
    - Insert refund record to canc(pnr, rfare) based on refund rules.

3. Non-Functional Requirements

- Security: Use prepared statements for sensitive operations (e.g., login) to prevent SQL injection.
- Reliability: Reservation and payment data must remain consistent; triggers

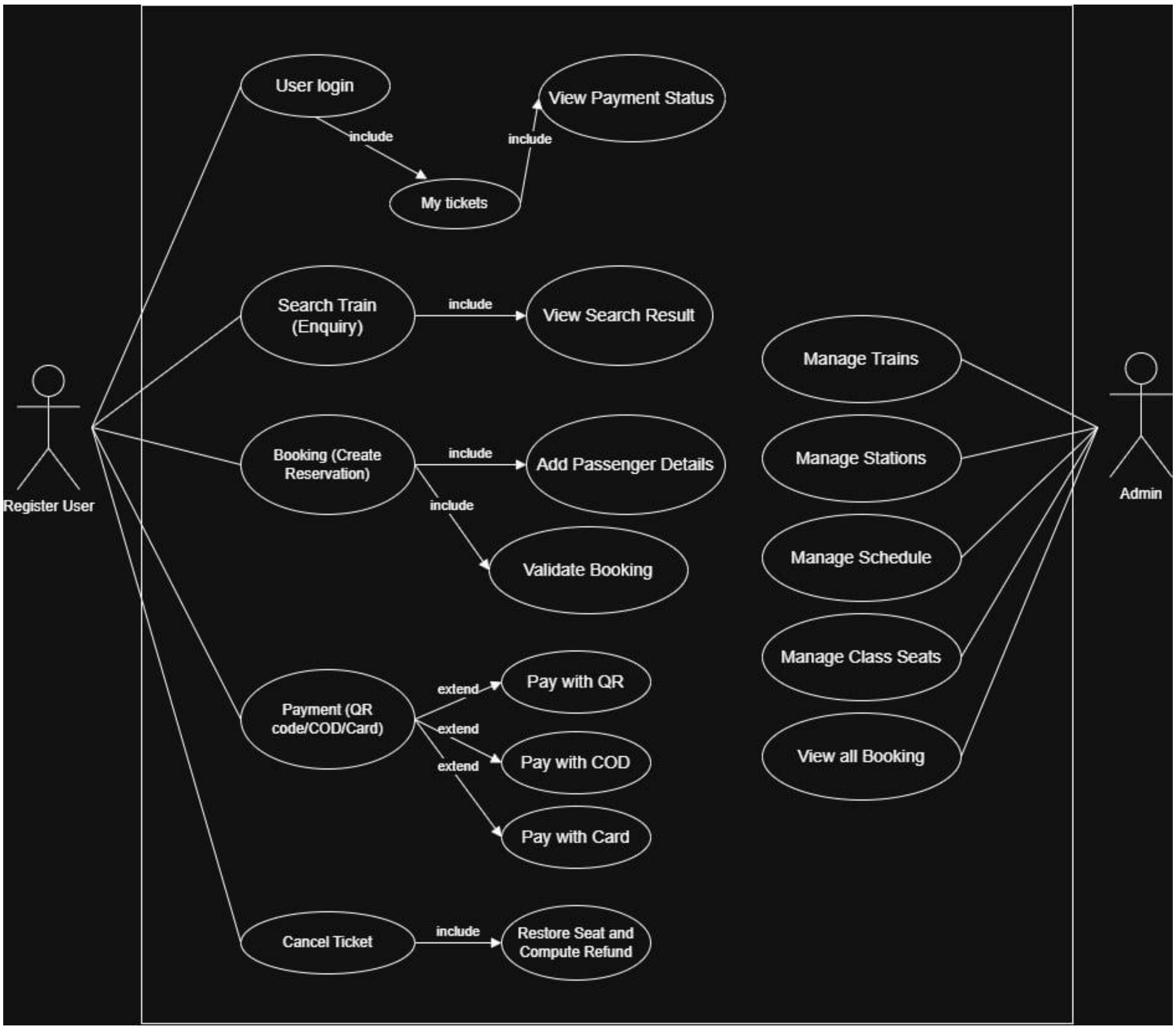
enforce cancellation rules automatically.

- Usability: Clear, responsive UI using Bootstrap and user-friendly error messages.
- Maintainability: Separate PHP files by feature (enquiry, result, booking, payment, tickets, cancel).

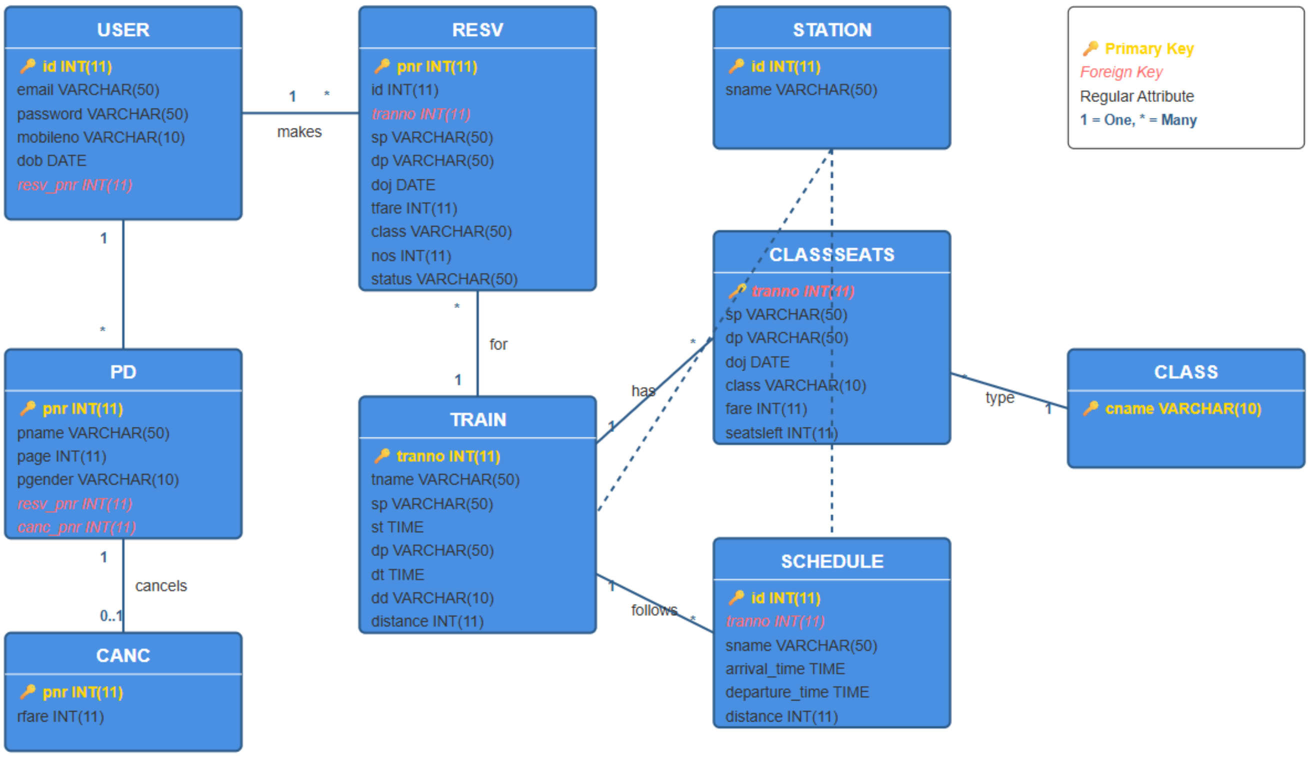
III. System Design

1. System Architecture (Simple 3-tier)

- Presentation layer: HTML/CSS/Bootstrap pages (index.htm, enquiry.php, enquiry_result.php, …)
- Application layer: PHP logic (session handling + booking/payment/cancel flows)
- Data layer: MySQL database (tables + triggers)



2. Database Design (ERD Description)



Main Tables:

- **user**(id, emailid, password, mobileno, dob, resv_pnr)
- **station**(id, sname)
- **train**(tranno, tname, sp, dp, st, dt, dd, distance)
- **schedule**(id, tranno, sname, arrival_time, departure_time, distance, tran_tranno)
- **classseats**(tranno, sp, dp, doj, class, fare, seatsleft)
- **class**(cname)
- **resv**(pnr, id, tranno, sp, dp, doj, tfare, class, nos, status)
- **pd**(pnr, pname, page, pgender, resv_pnr, canc_pnr, resv_pnr1)
- **canc**(pnr, rfare)

Key Relationships:

- **user** (1) → (n) **resv** - One user can make many reservations
- **resv** (n) → (1) **train** - Many reservations for one train
- **resv** (1) → (n) **pd** - One reservation has many passenger details
- **resv** (1) → (0..1) **canc** - One reservation may have one cancellation record
- **train** (1) → (n) **classseats** - One train has multiple class seat configurations
- **train** (1) → (n) **schedule** - One train follows multiple scheduled stops
- **classseats** (n) → (1) **class** - Many seat records belong to one class type
- **station** is referenced by **train** (sp, dp), **schedule** (sname), and **classseats** (sp, dp)

Important fix (to avoid Duplicate Entry error):
If canc.pnr is a PRIMARY KEY, repeated insert will cause error. Replace insert with UPSERT:

- INSERT ... ON DUPLICATE KEY UPDATE ...
  (or delete old record before inserting, but UPSERT is cleaner).

IV. Implementation

1. Main Functional Flow

- index.htm → Book Ticket
- enquiry.php → choose From/To/Date → submit
- enquiry_result.php → show trains + booking form
- resvn.php → passenger details + payment method

- new_png.php → insert reservation + passengers + payment record
- user_login.php → My Tickets + payment status
- cancel.php → set status CANCELLED → trigger updates seats + refund

2. Payment Module (QR)

When user selects QR:

- UI displays <img src="qr.png"> using the internal image in the project directory.
  When booking is created:
- Insert into payments: method='QR', status='PENDING', optional txn_ref, created_at.

A separate payment_confirm.php (optional) can later update PENDING → PAID.

3. My Tickets – Display Payment Status

Example query:

```
SELECT r.*, p.status AS pay_status, p.method AS pay_method

FROM resv r

LEFT JOIN payments p ON p.pnr = r.pnr

WHERE r.id = ?

ORDER BY r.pnr DESC;
```

UI suggestion:

- Ticket status badge: BOOKED / CANCELLED
- Payment badge: PAID (green), PENDING/UNPAID (yellow/red), NULL →

UNPAID

V. Testing

1. Sample Test Cases

- TC01 Search valid trains: sp, dp, doj → returns train list.
- TC02 Booking all children: all ages < 18 → reject with “must have 1 adult”.
- TC03 Booking with QR: show qr.png; create records in resv, pd, payments(PENDING).
- TC04 My Tickets: login → list bookings + show pay_status.
- TC05 Cancel before journey date: restore seats + create refund record in canc.
- TC06 Cancel after journey date: trigger blocks cancellation.

2. Issues Encountered & Fixes

- Duplicate entry (PRIMARY/UNIQUE) when trigger inserts into canc more than once or when unique constraints conflict.
  → Fixed by using ON DUPLICATE KEY UPDATE or adjusting constraints based on business rules.

VI. Discussion & Conclusion

1. Achievements

- Implemented an end-to-end booking flow: enquiry → booking → passenger → payment → tickets → cancel.
- Improved UI with Bootstrap and clean layout.

- QR payment meets requirement: uses internal qr.png (not external URL).
- My Tickets shows payment status per PNR.
- Trigger ensures consistent seat restoration and refund rules.

2. Limitations

- Payment is simulated (no real payment gateway integration).
- Session-based flow depends on correct navigation order.
- Admin and user roles may not be fully separated if not implemented.

3. Future Enhancements

- Add “Confirm Payment” to update PENDING → PAID and store paid timestamp.
- Restrict viewing ticket details if payment is not PAID (optional).
- Improve security: hash passwords, stricter validation, CSRF protection.

VII. References

- PHP Manual – MySQLi Prepared Statements
- MySQL Documentation – Triggers and Constraints
- Bootstrap 5 Documentation
- Font Awesome Documentation