VNU-HCM International University

TRAIN TICKET MANAGEMENT

Final Project Report

Subject: Web Application Development

Instructor: Nguyen Trung Nghia

Group Members:

1. Le Nhat Duy - ITITWE22143
2. 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 to Station B by selected date.

- Manage remaining seats based on seat class and route.

- Generate a unique PNR for each reservation.

- Integrate basic payment flow:

    + QR payment is shown using the internal project image qr.png (not an external URL).

    + Save payment records on 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

+ Register user: Search → Booking → Passenger Details → Payment → My Tickets → Cancel Ticket

+ Admin: Manage train/station/schedule/class seats data

4. Technologies Used

Frontend: HTML/CSS, Bootstrap

Backend: PHP

Database: MySQL

Local environment: XAMPP and 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)

Allow user to search the train without registered but not allow booking if user isn't register
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 users 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 seats left 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.

UC06 – My Tickets

Functional Requirements:

User logs in using mobile and password.

Display bookings from resv.

Display payment status to payments.

UC07 – Cancel Ticket

Functional Requirements:

User enters PNR to cancel (must belong to the logged-in user).

Update resv.status: showing ‘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.

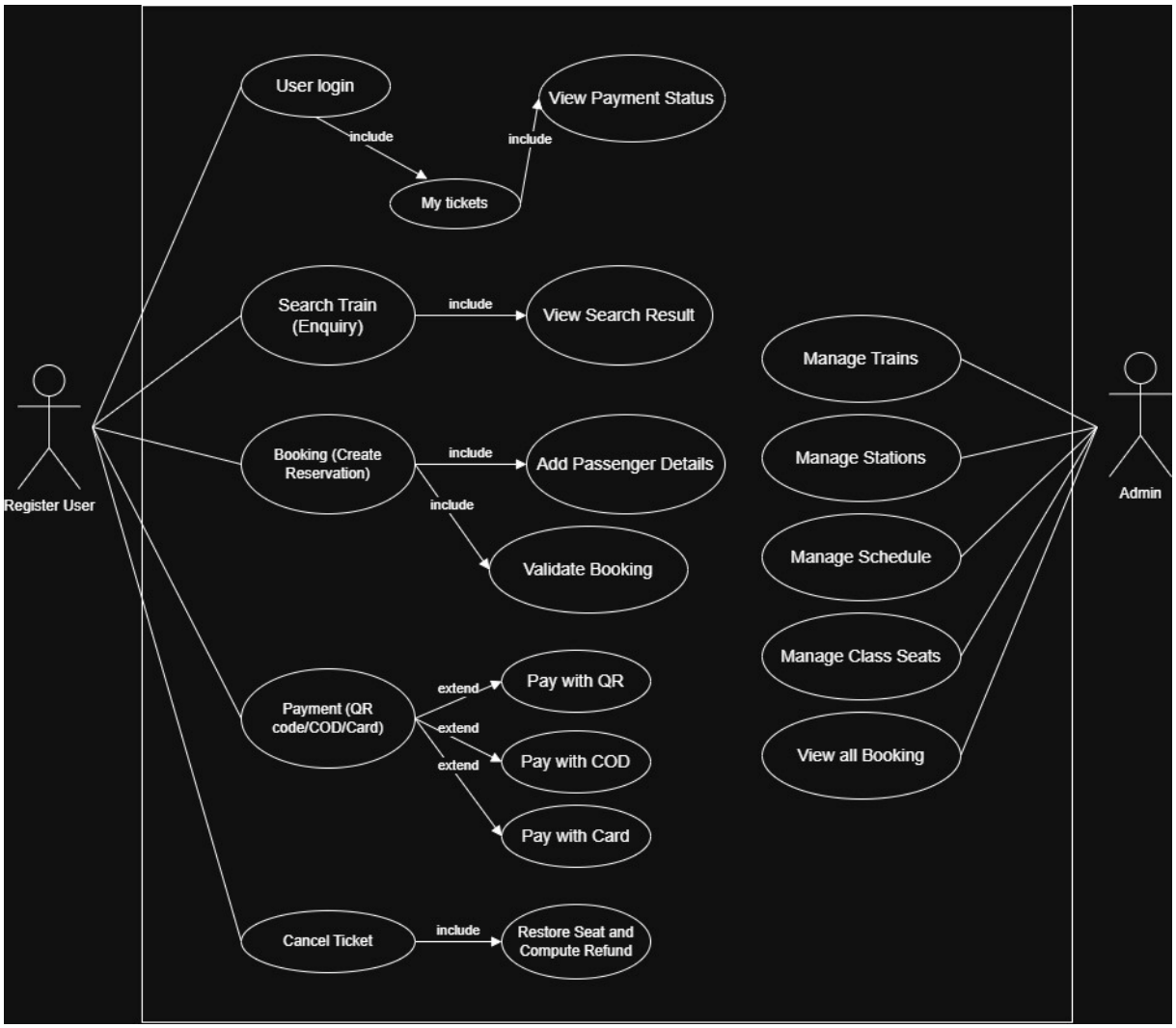
Reliability: Reservation and payment data must remain consistent; triggers enforce cancellation rules automatically.

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. Use Case Diagram



2. Database Design (ERD Description)

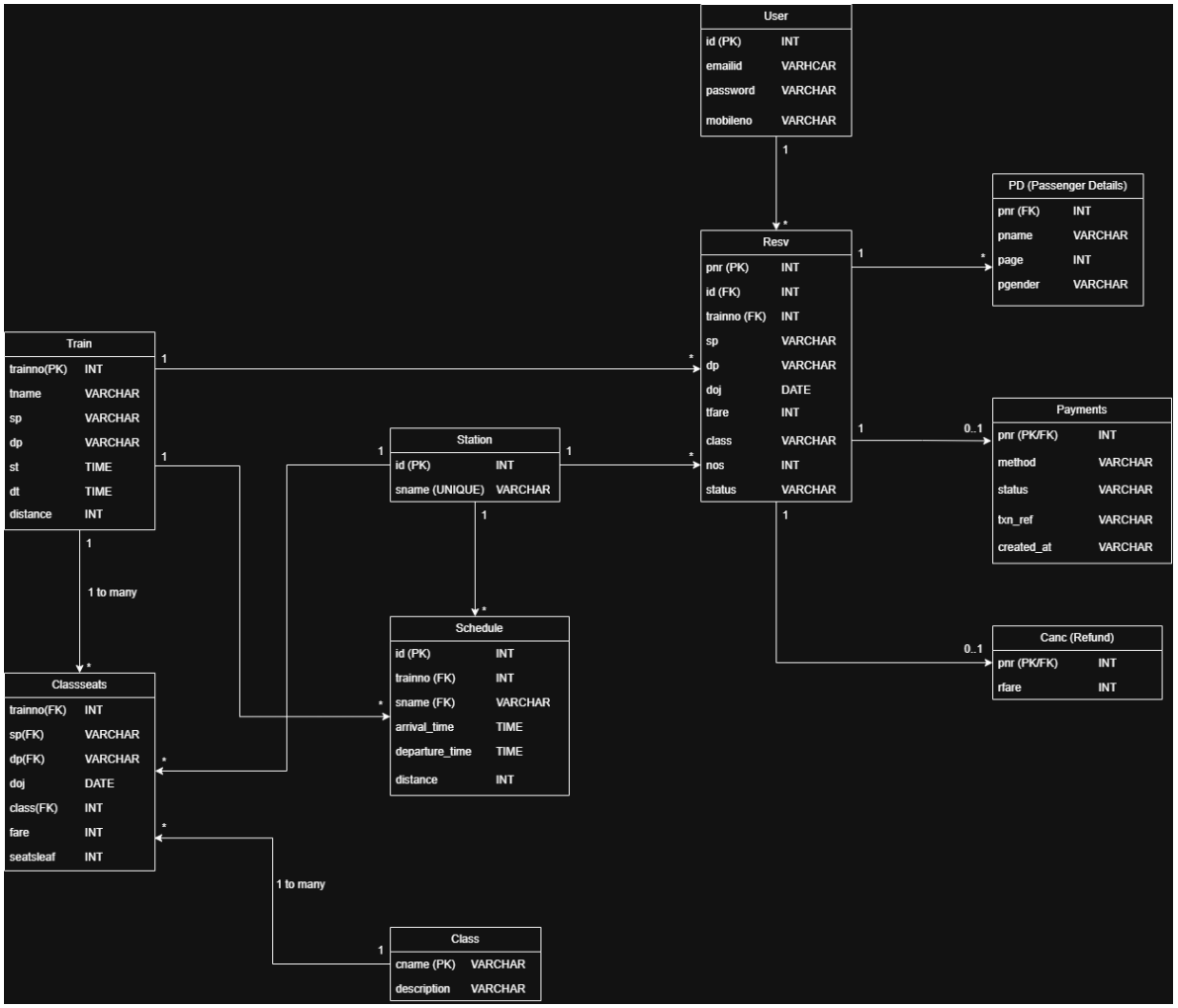
Main tables:

- user(id, mobileno, password, emailid, …)
- station(id, sname)
- train(trainno, tname, dd, …)
- schedule(trainno, sname, departure_time/arrival_time, …)
- classseats(trainno, sp, dp, doj, class, fare, seatsleft)
- resv(pnr, id, trainno, sp, dp, doj, tfare, class, nos, status)

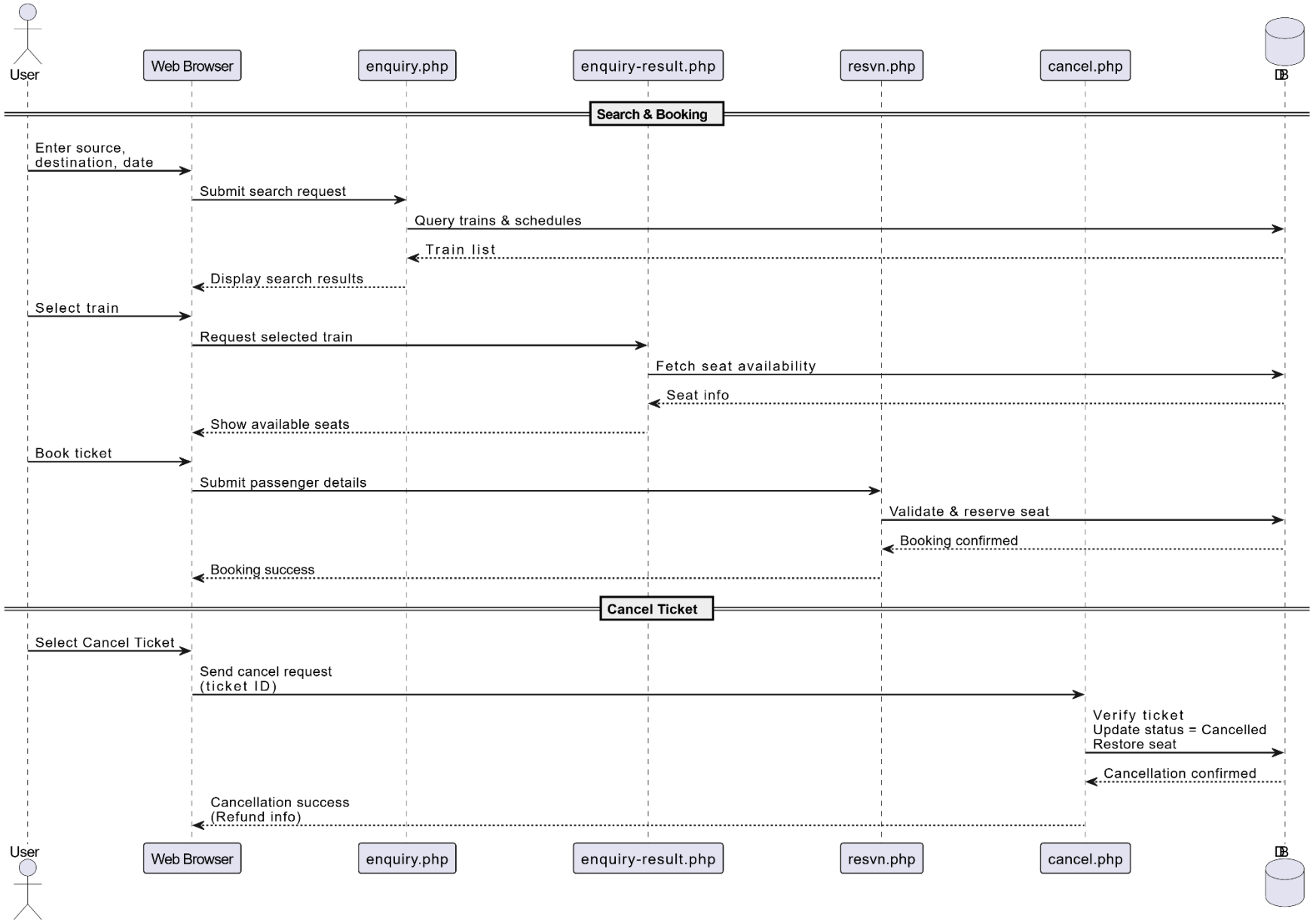
- pd(pnr, pname, page, pgender)
- payments(pnr, method, status, txn_ref, created_at, …)
- canc(pnr, rfare)

Key relationships:

- user (1) → (n) resv
- resv (1) → (n) pd
- resv (1) → (0..1) payments (a ticket may not be paid yet)
- resv (1) → (0..1) canc (one cancellation may create one refund record)



2. Sequence Diagram

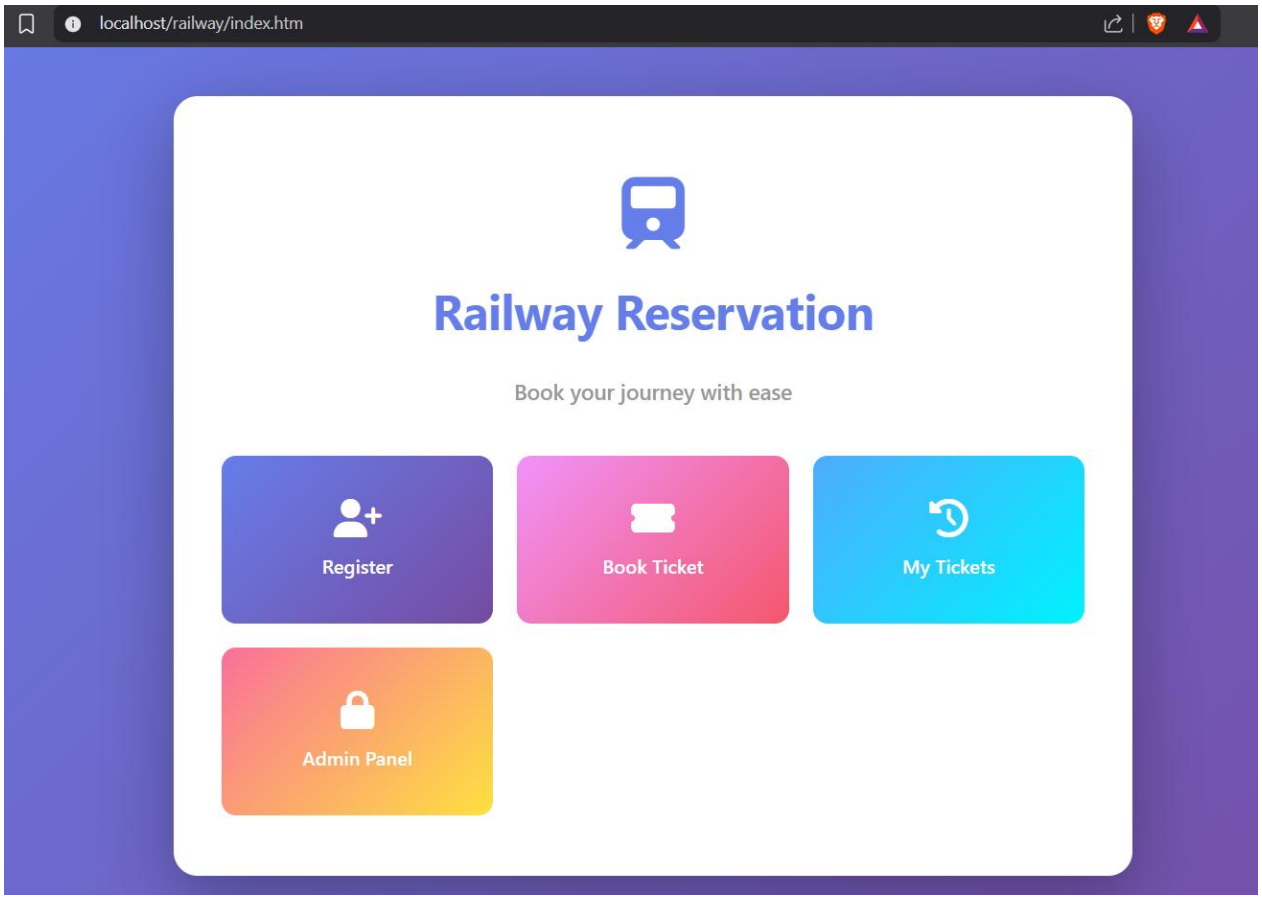

User
Web Browser
enquiry.php
enquiry-result.php
resvn.php
cancel.php
DB
Search & Booking
Enter source, destination, date
Submit search request
Query trains & schedules
Train list
Display search results
Select train
Request selected train
Fetch seat availability
Seat info
Show available seats
Book ticket
Submit passenger details
Validate & reserve seat
Booking confirmed
Booking success
Cancel Ticket
Select Cancel Ticket
Send cancel request (ticket ID)
Verify ticket
Update status = Cancelled
Restore seat
Cancellation confirmed
Cancellation success (Refund info)
User
Web Browser
enquiry.php
enquiry-result.php
resvn.php
cancel.php
DB

IV. Flow Code

1. Main Functional Flow

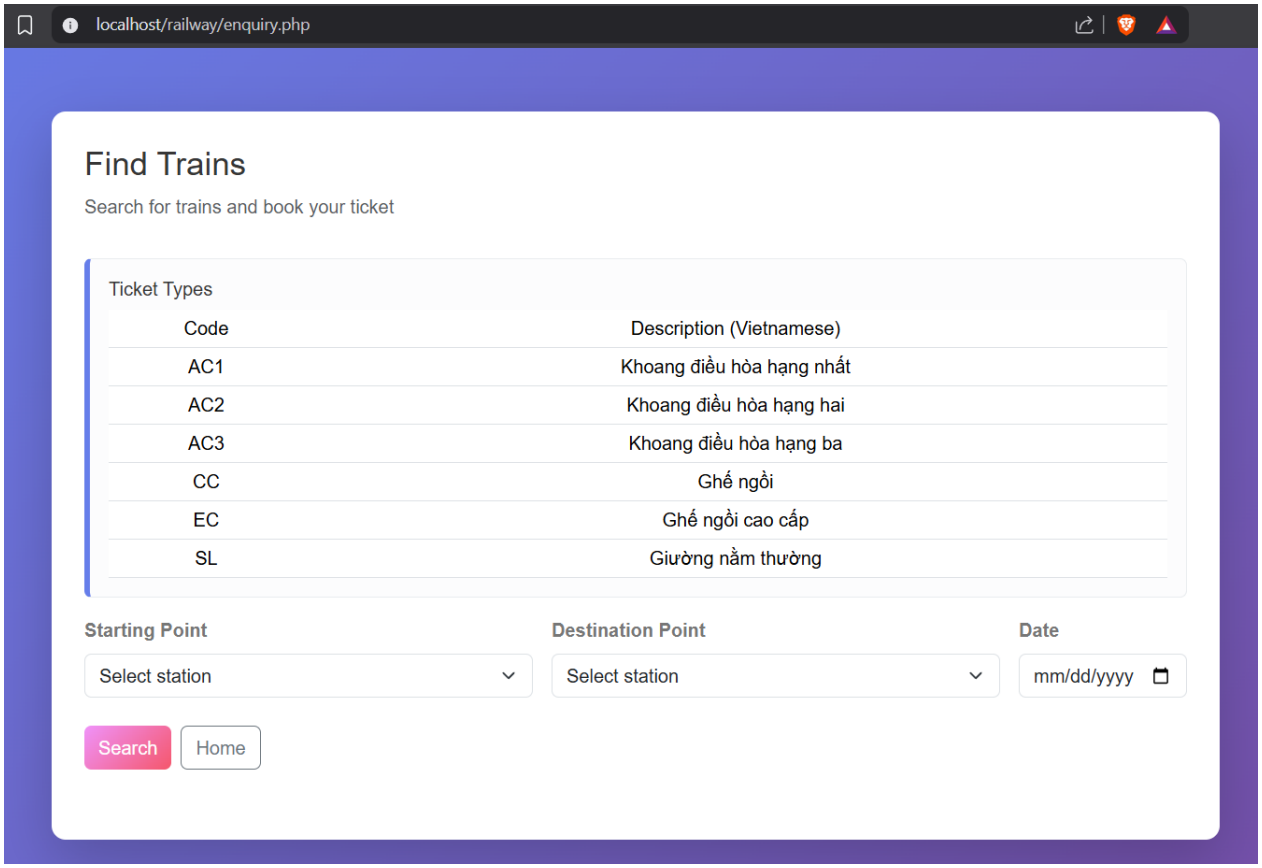
index.html (first start UI): including all main functions of the train ticket web. Entry point of all system. When users click a feature button:

- Register: redirects to the registration page to create a new user account.
- Book ticket: starts the booking process (search, choose train, enter passenger details, payment)
- My tickets: opens the login + ticket history page where users can view bookings and payment status.
- Admin: login as admin account, allows administrators to manage system data (stations, trains, schedules, class seats).



Book Ticket (enquire.php): allow people to choose the starting point, destination point and the date to search for available train that occurs on that date.

Starting with creating session by session.start() so the system can store the user's search data and reuse it in later pages. After that we need to use require "db.php" to loads the database connection so the page can query stations from MySQL. Then we call method from SQL: "SELECT sname FROM station" to fetches the list of available stations from the station table to populate the FROM and TO dropdowns. With id: sp (starting point), dp (destination point) and doj (date of journey) allow user to selected and the system will compare the user's selection to database to show the suitable outcome after user click Search. Search buttons will POST to enquire_result.php which can submit the search criteria to the results page to retrieve matching trains.



enquiry_result.php (after complete action at Book Ticket – enquire.php): show search result and booking form

Just like enquire.php, we will call session first using session_start(); continues the session so

the page can store the search values for later booking steps. After that, we ues require “db.php” to enables database queries to fetch train results and seat/fare data. Because the user choosed id: sp, dp and doj at the previous enquire.php, so this web pages will read the inputs to retrieves the user’s search criteria sent from enquiry.php. System will save the search inputs into session using code function from line 5 to line 10, this also keeps the journey info across multiple pages (so the user does not re-enter it).

enquiry_result.php

```
1  <?php
2  session_start();
3  require "db.php";
4
5  $doj = $_POST["doj"];
6  $_SESSION["doj"] = "$doj";
7  $sp = $_POST["sp"];
8  $_SESSION["sp"] = "$sp";
9  $dp = $_POST["dp"];
10 $_SESSION["dp"] = "$dp";
11
12 $query = mysqli_query(mysql: $conn, query: "SELECT t.trainno, t.tname, c.sp, s1.departure_ti
13
14 $trains = mysqli_fetch_all(result: $query, mode: MYSQLI_ASSOC);
15 ?>
```

Then we use the function at line 12 to return train options that match the route and date, including: train number/name, class, fare per seat, seats left and it will show all available trains for the selected route/date so the user can choose one. We create some options for customers like selecting trains and payment methods to collect booking preferences before moving to passenger details. At payment methods, if user choose QR code, it will show the QR scan for them. After all, user clicks the Proceed with booking, it will send booking preferences to the passenger details page (resvn.php).

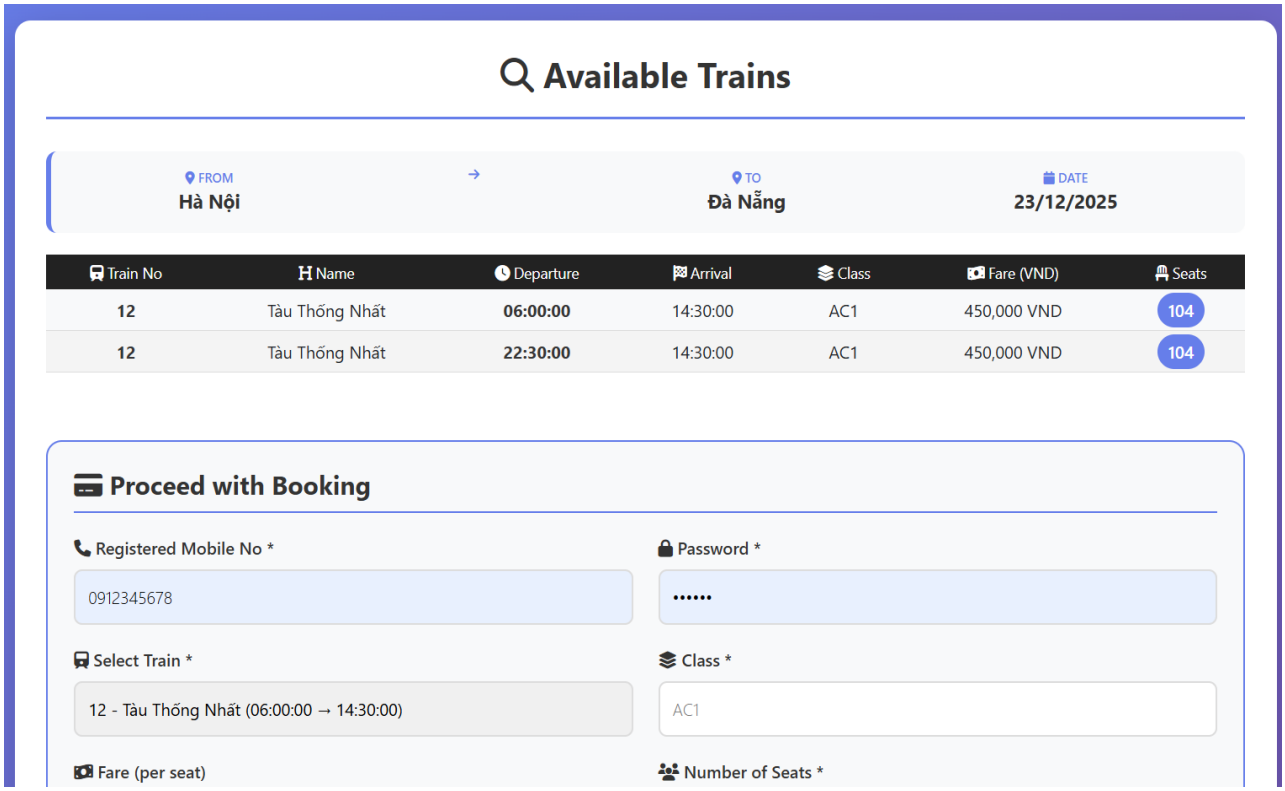
resvn.php (after user click Proceed with booking – enquire_result.php):

First, we call session to read existing session values from the previous pages. This page will read data booking from enquire_result.php through variables id, tno, class, nos and get the user’s chosen train/class/seats to show up in html. Then we save booking data to session (line 23 to 35) for the next page (new_png.php) and next page will insert the booking into the database using these session values. We create a place for users to insert their data like name age gender and collected it for each seat booked. After user done all of that, they click

Confirm Booking, it will submit all passenger arrays + booking data for database insertion and move to new_png.php

new_png.php

```php
$success = false;
$error = "";
$tempfare = 0;
$adultCount = 0;
$rpnr = null;

$pname = $_POST["pname"];
$page = $_POST["page"];
$pgender = $_POST["pgender"];

$tno = $_SESSION["tno"];
$doj = $_SESSION["doj"];
$sp = $_SESSION["sp"];
$dp = $_SESSION["dp"];
$class = $_SESSION["class"];

// Get fare
$query = "SELECT fare FROM classseats WHERE trainno='" . $conn->real_escape_string(string: $
$result = mysqli_query(mysql: $conn, query: $query);

if($result && mysqli_num_rows(result: $result) > 0) {
  $row = mysqli_fetch_array(result: $result);
  $fare = $row[0];
```



12 - Tàu Thống Nhất (06:00:00 → 14:30:00)

AC1

Fare (per seat)

450,000 VND

Number of Seats *

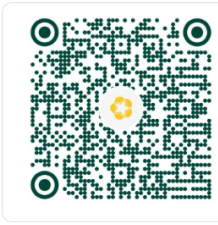
2

Seats left: 104

Payment Method *

QR (Scan)

Scan QR to Pay



After scanning and paying, enter transaction reference (optional):

Transaction reference (optional)

Proceed with Booking

USER ID 2 TRAIN 12 CLASS AC1 SEATS 2

Payment Method

QR (Scan QR / Banking)

Note (optional)

E.g. pay later / special request

Passenger Information

1 Passenger 1

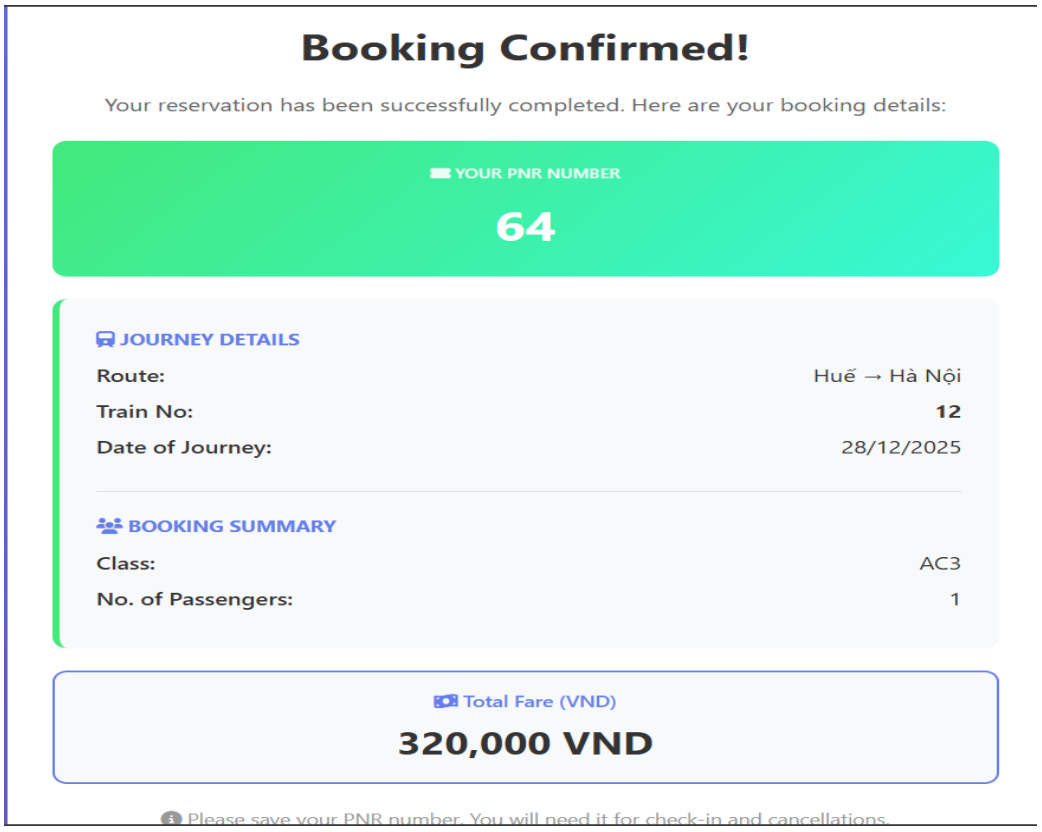
Name	Age	Gender
Le	22	Male

2 Passenger 2

Name	Age	Gender
Nhat	22	Male

← Back to Enquiry

✓ Confirm Booking



Booking Creation Flow (new_png.php):

First, use session_start() to retrieves booking context from previous steps (user ID, train/class, journey, seats). Then use require “db.php” to enables all INSERT/UPDATE SQL queries. Then it will read passenger input arrays through pname[], page[], pgender[] to prepare details to insert into table pd. Then it will read booking data from session tno, sp, dp, doj, class, nos, id to prepares data for inserting the reservation into table resv (line 68 to 76).

```
for($i = 0; $i < $_SESSION["nos"]; $i++) {
  if($page[$i] >= 18) {
    $adultCount++;
    $tempfare += $fare;
  } else if($page[$i] < 18) {
    $tempfare += 0.5 * $fare;
  }
}

// Check for at least one adult
```

```
  if($adultCount > 0) {
    // Insert reservation
    $sql = "INSERT INTO resv(id, trainno, sp, dp, doj, tfare,
class, nos) VALUES ('" . $_SESSION["id"] . "', '" . $conn-
>real_escape_string($tno) . "', '" . $conn-
>real_escape_string($sp) . "', '" . $conn-
>real_escape_string($dp) . "', '" . $conn-
>real_escape_string($doj) . "', '" . $tempfare . "', '" . $conn-
>real_escape_string($class) . "', '" . $_SESSION["nos"] . "')";

    if ($conn->query($sql) === TRUE) {
      // Get the PNR
      $query = "SELECT pnr FROM resv WHERE id='" .
$_SESSION["id"] . "' AND trainno='" . $conn-
>real_escape_string($tno) . "' AND doj='" . $conn-
>real_escape_string($doj) . "' ORDER BY pnr DESC LIMIT 1";
      $result = mysqli_query($conn, $query);
      $row = mysqli_fetch_array($result);
      $rpnr = $row['pnr'];

      // Insert passenger details
      $passengerInsertSuccess = true;
      for($i = 0; $i < $_SESSION["nos"]; $i++) {
        $sql = "INSERT INTO pd(pnr, pname, page, pgender) VALUES
('" . $rpnr . "', '" . $conn->real_escape_string($pname[$i]) .
"', '" . $page[$i] . "', '" . $conn-
>real_escape_string($pgender[$i]) . "')";
        if ($conn->query($sql) !== TRUE) {
          $passengerInsertSuccess = false;
          $error = $conn->error;
          break;
        }
      }

      if($passengerInsertSuccess) {
```

```
        $success = true;
      }
    } else {
      $error = $conn->error;
    }
  } else {
    $error = "At least one adult (age 18+) must accompany the
group!";
  }
} else {
  $error = "Could not retrieve fare information. Please try
again.";
}
```

Then it gets fare from classeats to obtain the correct fare per seat based on train/class/route/data (line 37 to 42). Then we continue to calculate fare through aged that user send (line 51 to 59), loop through passengers: age >= 18 is full fare, otherwise is 50% fare and we have to make sure that the passengers include adult by checking adultcount > 0 (line 62) to ensure that there are at least one adult must be included in the group. Then we will insert data into resv table (line 68 to 76) to create main booking record (the system will generate random and unique pnr at the same time). PNR generation (line 81 to 88) - get the new pnr so it can be used to link passengers and payments to this reservation. Then it will insert passengers' rows into pd, it creates the statement $passengerInsertSuccess = true first, then insert passengers's details (line 97 to 101). After that, it will use the statements to check if it still true or false, if it true: it will insert payment row to pd(line 115) in order to stores passenger information linked to the reservation PNR. Then insert payment record into **payments** (line 129 – 131) so that can save payment stat per ticket so My Tickets can display payment status later. After all, show booking confirmation UI and give the user PNR.

Login + My Tickets Flow (user_login.php):

Case 1: user not logged yet.

Show login form and call session for preparation usage (to store user ID after login)

user_login.php

```php
session_start();
<?php
session_start();
require "db.php";

if ($conn->connect_error) {
  die("Connection failed: " . $conn->connect_error);
}

$isValidUser = false;
$invalidCreds = false;
$bookings = array();
$temp1 = "";
$temp2 = "";

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
  $mobile = isset($_POST['mno']) ? trim(string: $_POST['mno']) : '';
  $pwd = isset($_POST['password']) ? $_POST['password'] : '';

  $stmt = $conn->prepare(query: "SELECT id, emailid FROM user WHERE mobileno = ? AND passwor
  $stmt->bind_param(types: 'ss', var: &$mobile, vars: &$pwd);
  $stmt->execute();
  $res = $stmt->get_result();
```

Case 2: user is login.

Read input (line 16, 17) to receive user credentials from the login form. Then prepare statement user validation to check the credentials securely and prevents SQL injection (line 20). Check if credentials are correct, it will marks login success and stores the user ID to identify the user in future action (line 25 - 33). Then load ticket history with payment status (line 45 – 54) fetched all reservation for the logged-in user and attaches payment details per ticket. After successful login, user can see the system had already render the ticket with PNR, train, date, etc…

Cancel Ticket Flow (cancel.php):

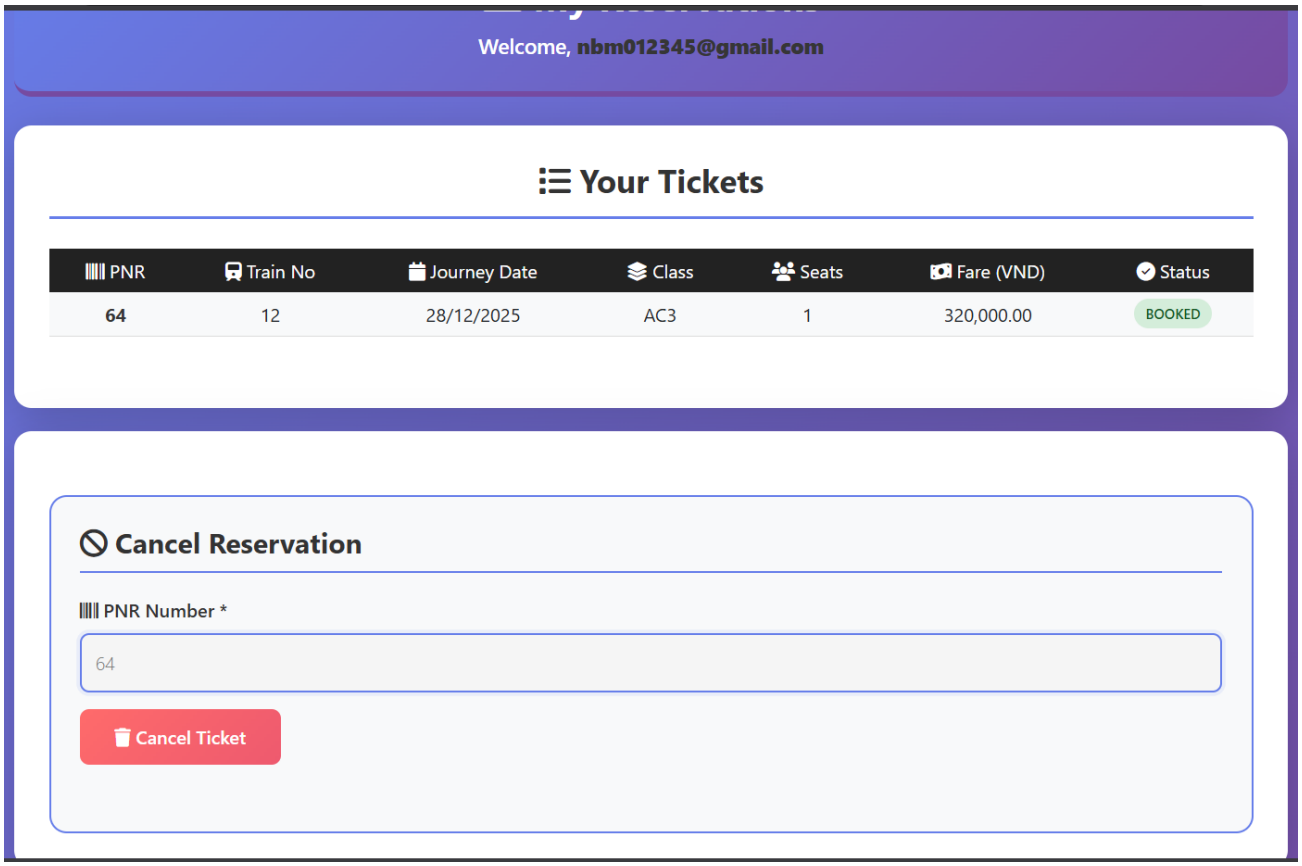
First, session_start() for reading logged-in user ID from session. Then require "db.php" to enable database update queries. Then system will read PNR from Post (line 5) to identify which booking the user wants to cancel and read user ID from session (line 6) to ensure the user can only cancel their own reservation. System will initialize result flags to prepare variables to show success/failure message on the result page (line 8, 9). Then system will validate some require value like user ID or PNR, if user id is missing or PNR is empty it will activate line 12, it helps prevent running SQL when input/session is invalid. System will check current booking status after that, this is important step with statement line 16, the system will confirm that PNR exists, the ticket belongs to the right user, read the current

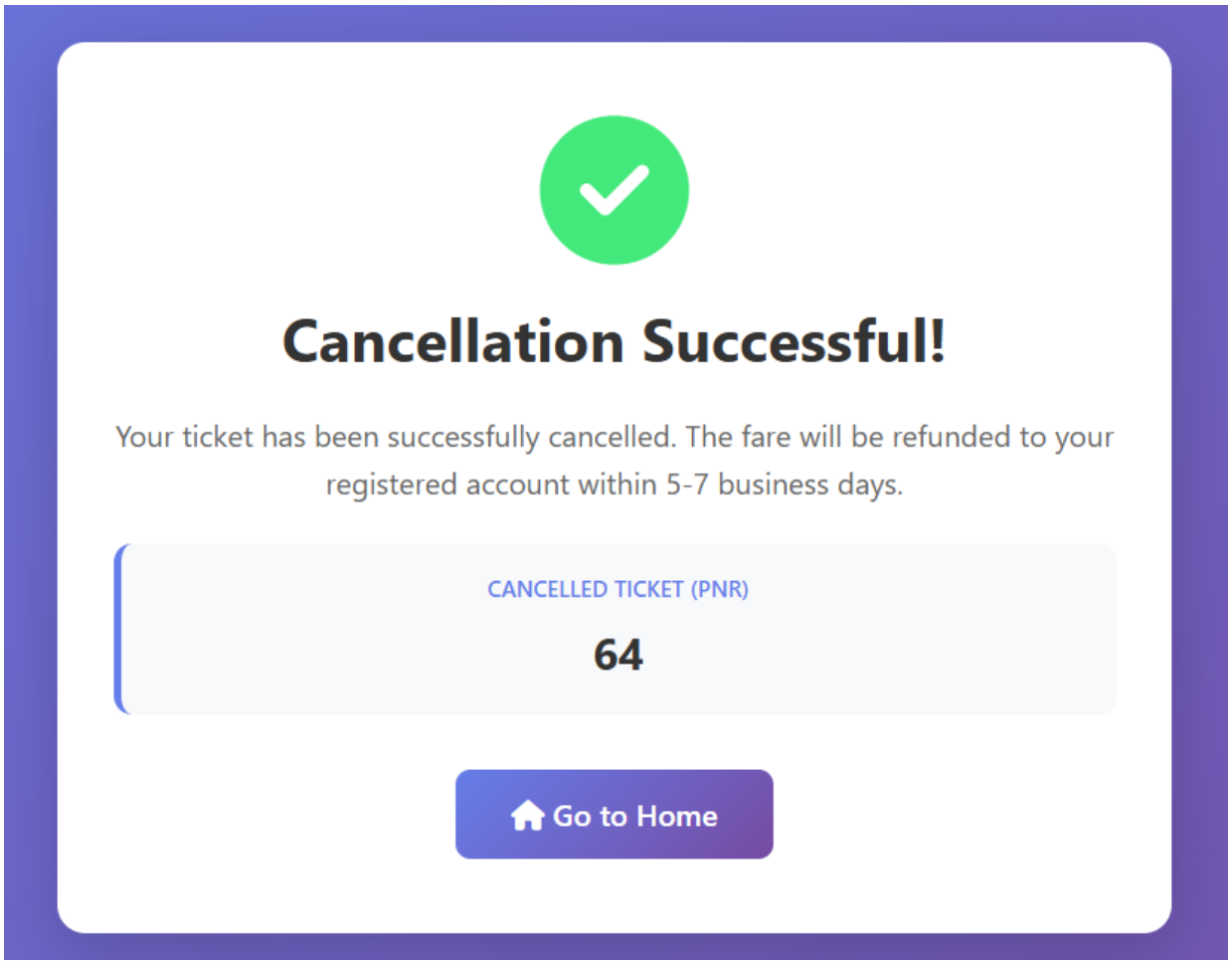
status (BOOKED or CANCELLED). If the PNR does not exist or the ticket belongs to the wrong user, it will alert (line 21-22), user cannot cancel the ticket that does not belong to them. And if the ticket has already CANCELLED (line 25) it will print line 26 to avoid cancelling the same ticket and triggering database logic (trigger can cause duplicate refund). If it does not meet any condition above, the system will update status that ticket has been cancelled (line 29). After all, show result page (HTML), if $success == true: show "Cancellation Successful" (line 147-151), else show "Cancellation Failed" and display error message.

user_login.php

```
session_start();
<?php
session_start();
require "db.php";

if ($conn->connect_error) {
  die("Connection failed: " . $conn->connect_error);
}

$isValidUser = false;
$invalidCreds = false;
$bookings = array();
$temp1 = "";
$temp2 = "";

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
  $mobile = isset($_POST['mno']) ? trim(string: $_POST['mno']) : '';
  $pwd = isset($_POST['password']) ? $_POST['password'] : '';

  $stmt = $conn->prepare(query: "SELECT id, emailid FROM user WHERE mobileno = ? AND passwor
  $stmt->bind_param(types: 'ss', var: &$mobile, vars: &$pwd);
  $stmt->execute();
  $res = $stmt->get_result();
```

```
  <h2>Cancellation Successful!</h2>
  <p class="result-message">
    Your ticket has been successfully cancelled. The fare will be refunded to your regis
  </p>
  <div class="pnr-display">
    <div class="pnr-label">Cancelled Ticket (PNR)</div>
    <div class="pnr-value"><?php echo htmlspecialchars(string: $pnr); ?></div>
  </div>
<?php } else { ?>
  <div class="error-icon">
    <i class="fas fa-exclamation-circle"></i>
  </div>
  <h2>Cancellation Failed</h2>
  <p class="result-message">
    We encountered an error while processing your cancellation request.
  </p>
  <div class="error-details">
    <strong>Error Details:</strong><br>
    <?php echo htmlspecialchars(string: $error); ?>
  </div>
  <p class="result-message">
```


Welcome, nbm012345@gmail.com
Your Tickets
PNR
Train No
Journey Date
Class
Seats
Fare (VND)
Status
64
12
28/12/2025
AC3
1
320,000.00
BOOKED
Cancel Reservation
PNR Number *
64
Cancel Ticket


Cancellation Successful!
Your ticket has been successfully cancelled. The fare will be refunded to your registered account within 5-7 business days.
CANCELLED TICKET (PNR)
64
Go to Home

4.4 Admin function

Session Initialization

Starting by calling session_start() to enable session management. This allows the system to store and retrieve the admin login state across multiple pages.

```
<body>
    <div class="admin-card">
        <?php
        session_start();
        if(isset($_POST['uid']) && $_POST['uid']==='admin' && isset($_POST['password']) && $
            $_SESSION['admin_login'] = true;
        }

        if(!empty($_SESSION['admin_login'])){
            echo '<h4>Admin Dashboard</h4>';
            echo '<div class="admin-links">';
            echo '<a href="insert_into_stations.php" class="btn btn-outline-primary btn-sm">
            echo '<a href="show_trains.php" class="btn btn-outline-primary btn-sm">Show All
            echo '<a href="show_users.php" class="btn btn-outline-primary btn-sm">Show All U
            echo '<a href="insert_into_train_3.php" class="btn btn-outline-primary btn-sm">I
            echo '<a href="insert_into_classseats_3.php" class="btn btn-outline-primary btn-
            echo '<a href="booked.php" class="btn btn-outline-primary btn-sm">View all booke
            echo '<a href="cancelled.php" class="btn btn-outline-primary btn-sm">View all ca
            echo '</div>';
        } else {
            echo '<h4 class="mb-3">Admin Login</h4>';
            echo '<form action="admin_login.php" method="post">';
            echo '<div class="mb-2"><label class="form-label">User ID</label><input class="f
            echo '<div class="mb-2"><label class="form-label">Password</label><input class='
            echo '<div><button class="btn btn-primary">Login</button></div>';
```

```
            echo '<a href="insert_into_classseats_3.php" class="btn btn-outline-primary btn-
            echo '<a href="booked.php" class="btn btn-outline-primary btn-sm">View all booke
            echo '<a href="cancelled.php" class="btn btn-outline-primary btn-sm">View all ca
            echo '</div>';
        } else {
            echo '<h4 class="mb-3">Admin Login</h4>';
            echo '<form action="admin_login.php" method="post">';
            echo '<div class="mb-2"><label class="form-label">User ID</label><input class="f
            echo '<div class="mb-2"><label class="form-label">Password</label><input class='
            echo '<div><button class="btn btn-primary">Login</button></div>';
            echo '</form>';
        }
        ?>

        <div style="margin-top:14px;"><a href="index.htm" class="btn btn-link">Go to Home Pa
    </div>

    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.j
</body>
</html>
```

Receiving Login Credentials

- The system checks whether the HTTP request contains POST parameters:
    - uid (admin user ID)
    - password
- These values are submitted from the admin login form.

Admin Authentication

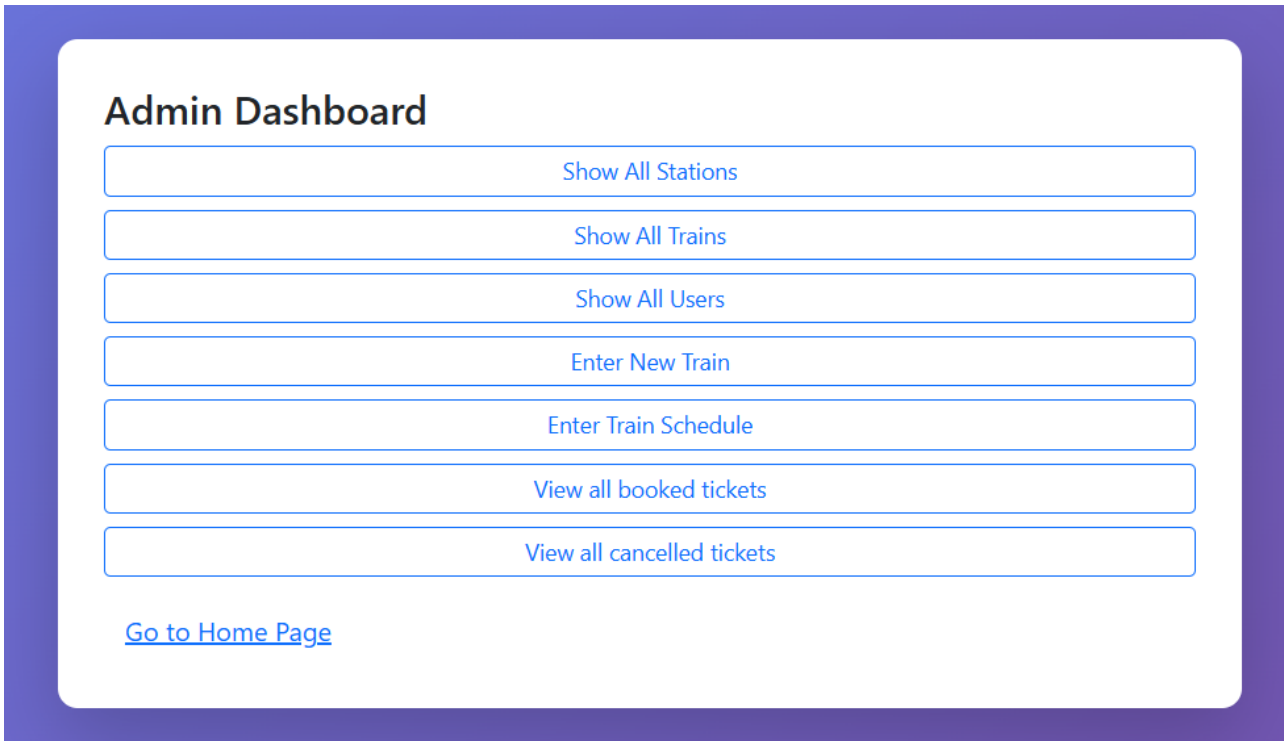
- The entered credentials are compared with predefined admin credentials:
    - User ID: admin
    - Password: admin
- If both values match:
    - A session variable $_SESSION['admin_login'] is set to true.
    - This indicates that the admin has been successfully authenticated.

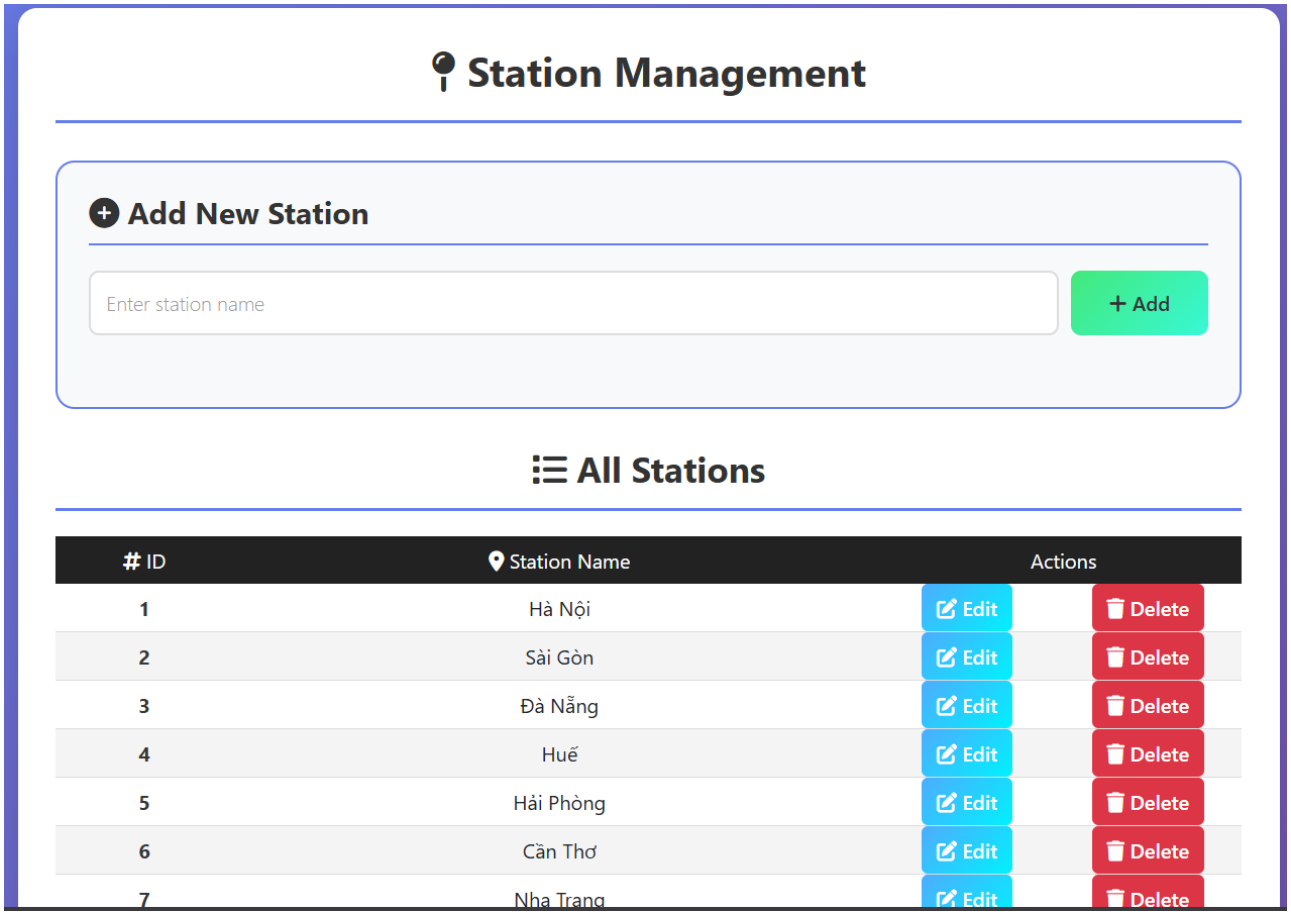
Session Validation

- The system checks whether the session variable admin_login exists and is not empty.
- If the session is valid:
    - The admin is considered logged in.
    - The system proceeds to display the admin dashboard.

Displaying Admin Dashboard

- Once logged in, the admin dashboard is shown.
- The dashboard provides navigation links to administrative functions, including:


Admin Dashboard
Show All Stations
Show All Trains
Show All Users
Enter New Train
Enter Train Schedule
View all booked tickets
View all cancelled tickets
Go to Home Page


Station Management
Add New Station
Enter station name
+ Add
All Stations
ID
Station Name
Actions
1
Hà Nội
Edit
Delete
2
Sài Gòn
Edit
Delete
3
Đà Nẵng
Edit
Delete
4
Huế
Edit
Delete
5
Hải Phòng
Edit
Delete
6
Cần Thơ
Edit
Delete
7
Nha Trang
Edit
Delete

All Trains

# No.	Name	From	Depart	To	Arrive	Day	Distance	Action
6	Tàu SE1	Đà Nẵng	06:30:00	Huế	11:00:00	Ngày 1	100 km	Details
12	Tàu Thống Nhất	Hà Nội	06:00:00	Sài Gòn	22:30:00	Ngày 2	1726 km	Details
13	Tàu SE3	Hải Phòng	07:00:00	Nha Trang	06:30:00	Ngày 2	1278 km	Details
14	Tàu SE5	Hải Phòng	08:00:00	Cần Thơ	22:00:00	Ngày 2	1800 km	Details
15	Tàu SE7	Hà Nội	16:00:00	Đà Nẵng	06:00:00	Ngày 2	764 km	Details
16	Tàu SE8	Đà Nẵng	07:30:00	Hà Nội	17:30:00	Ngày 1	764 km	Details
17	Tàu SE9	Hà Nội	05:00:00	Hải Phòng	18:30:00	Ngày 1	866 km	Details
18	Tàu SE10	Hải Phòng	08:00:00	Hà Nội	21:00:00	Ngày 1	968 km	Details
19	Tàu SE2	Huế	13:30:00	Đà Nẵng	18:30:00	Ngày 1	230 km	Details
20	Tàu SE11	Hà Nội	10:00:00	Đà Nẵng	20:00:00	Ngày 1	764 km	Details
21	Tàu SE12	Đà Nẵng	21:00:00	Hà Nội	07:00:00	Ngày 2	764 km	Details
22	Tàu SE13	Hà Nội	16:00:00	Sài Gòn	18:00:00	Ngày 2	1726 km	Details
23	Tàu SE14	Sài Gòn	06:00:00	Hà Nội	09:30:00	Ngày 2	1726 km	Details
24	SE123	Sài Gòn	09:00:00	Hà Nội	22:00:00	29/12/2025	500 km	Details

All Users

# ID	Email	Mobile	DOB	Actions	
2	tranvanb@gmail.com	0923456789	20/05/1995	Edit	Delete
3	lethic@gmail.com	0934567890	10/08/1997	Edit	Delete
4	phamvand@gmail.com	0945678901	25/12/1996	Edit	Delete
5	hoangvand@yahoo.com	0913452635	30/12/1993	Edit	Delete
6	dangthie@gmail.com	0987667556	01/01/1991	Edit	Delete
7	vuvang@hotmail.com	0987887665	08/09/1997	Edit	Delete
21	SERSETREWRT45@gmail.com	086969789	01/01/1111	Edit	Delete
30	nbm012345@gmail.com	0123456789	12/12/1999	Edit	Delete
33	minhquan2906.92@gmail.com	0986594972	02/01/2004	Edit	Delete
34	ericborder12@gmail.com	0123456798	12/12/1999	Edit	Delete

Add New User Admin Panel

Add Train Schedule

Step 1 of 2: Train Information

STEP 1 - TRAIN INFORMATION

Train Details

Train Name *

E.g., Express 2000

Starting Point *

-- Select Starting Station --

Departure Time *

--:-- --

Destination Point *

-- Select Destination Station --

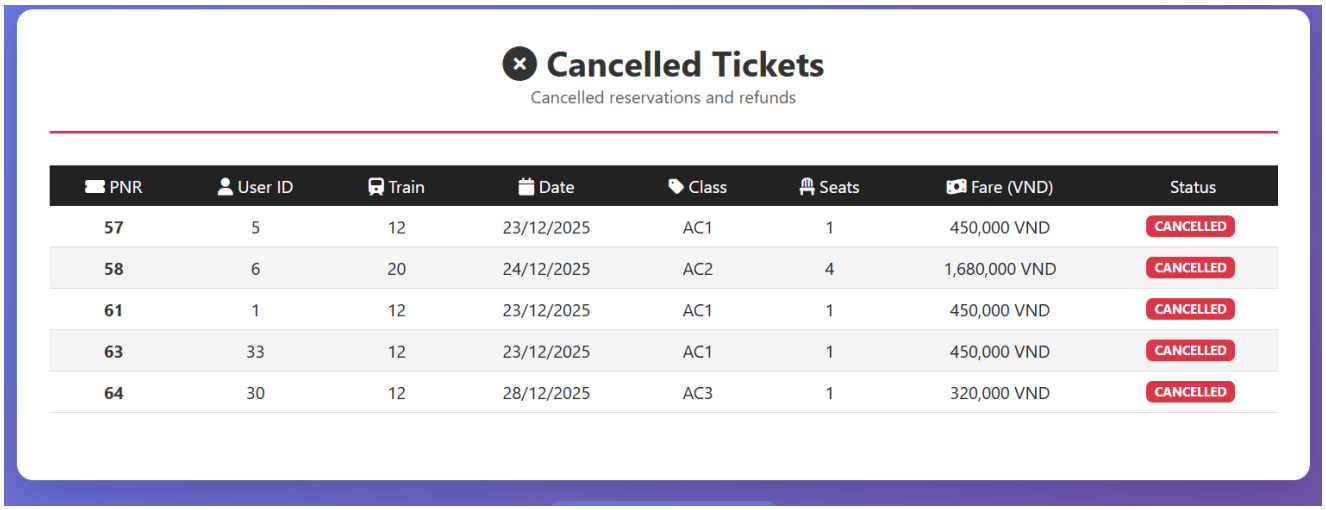
Arrival Time *

--:-- --

Booked Tickets

Active reservations

PNR	QR	User ID	Train	Date	Class	Seats	Fare (VND)	Status
51		4	12	23/12/2025	AC1	2	900,000 VND	BOOKED
59		10	12	28/12/2025	EC	2	440,000 VND	BOOKED
62		1	12	23/12/2025	AC1	1	450,000 VND	BOOKED



Cancelled Tickets

Cancelled reservations and refunds

PNR	User ID	Train	Date	Class	Seats	Fare (VND)	Status
57	5	12	23/12/2025	AC1	1	450,000 VND	CANCELLED
58	6	20	24/12/2025	AC2	4	1,680,000 VND	CANCELLED
61	1	12	23/12/2025	AC1	1	450,000 VND	CANCELLED
63	33	12	23/12/2025	AC1	1	450,000 VND	CANCELLED
64	30	12	28/12/2025	AC3	1	320,000 VND	CANCELLED

Handling Unauthenticated Access

- If the session variable admin_login is not set:
    - The admin login form is displayed.
    - The form requires the admin to enter a user ID and password.
    - The form submits data back to admin_login.php using the POST method.

Navigation Back to Home Page

- Regardless of login state, a link to the home page (index.htm) is displayed.
- This allows users or admins to return to the main application page easily.

V. Discussion & Conclusion

1. Achievements

Implemented an end-to-end booking flow: enquiry → booking → passenger → payment → tickets → cancel.

Improved UI and clean layout.

QR payment meets requirement uses internal qr.png (not external URL).

My Tickets show 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.

VI. References

PHP Manual – MySQLi Prepared Statements

MySQL Documentation – Triggers and Constraints

Bootstrap 5 Documentation

ChatGPT