CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

UNIVERSITY INSTITUTE OF COMPUTING

Bachelors of Computer Applications

DBMS

23CAT-251

CASE STUDY REPORT
ON
‘RAILWAY RESERVATION SYSTEM’

SUBMITTED TO : Mr.Arvinder sir

SUBMITTED BY : Tanu verma

UID: 23BCA10070

Class : 23BCA4-B

CHANDIGARH UNIVERSITY
Discover. Learn. Empower.
NAAC GRADE A+
Accredited University

ABSTRACT:

- *Introduction:*
- *Technique:*
- *System Configuration:*
- *INPUT:*
- *ER DIAGRAM:*
  - *TABLE REALTION:*
- *TABULAR FORMAT:*
- *TABLE CREATION:*
- *SQL QUERIES WITH OUTPUT:*
- *SUMMARY:*
- *CONCLUSION:*

INTRODUCTION

The Railway Reservation System is a computerized platform designed to simplify and automate the process of booking train tickets, managing passenger information, and organizing train schedules. It replaces manual booking processes, reduces human error, and enhances the efficiency and reliability of railway operations. The increasing demand for transportation, particularly in populous countries, makes such systems essential for smooth and effective railway management.

At the heart of this system lies a Database Management System (DBMS), which ensures that all data related to train services — including train numbers, routes, schedules, seat availability, and passenger details — are accurately stored, retrieved, and managed. A well-structured DBMS enables various stakeholders (passengers, ticketing agents, administrators) to access up-to-date information and perform tasks such as booking, cancellation, seat selection, and payment processing with ease.

The DBMS plays a critical role in ensuring data integrity, consistency, security, and concurrency control. It also supports transaction management, ensuring that operations such as seat reservations and cancellations are executed reliably, even when multiple users are accessing the system simultaneously. By using relational databases, data is organized into well-defined tables and relationships, making it easier to query and maintain.

In addition to its technical advantages, a railway reservation system built on DBMS principles contributes to better customer satisfaction, reduced wait times, and streamlined administrative operations. It also allows for the generation of detailed reports and analytics, which help in improving services, managing demand, and planning future infrastructure.

With advancements in technology, modern railway reservation systems are increasingly integrated with online portals, mobile apps, and real-time data processing, further increasing their accessibility and efficiency. The DBMS continues to be the core component that ensures the stability and scalability of these systems in handling millions of daily transactions across vast railway networks.

Let me know if you want to follow this with sections like Objectives, System Architecture, Entity-Relationship Model, or Advantages of DBMS in Railway Systems.

OBJECTIVES

The primary objective of the Railway Reservation System using a Database Management System (DBMS) is to develop an efficient, reliable, and user-friendly platform that streamlines the process of train ticket booking and management. The system aims to address the challenges of manual reservation systems by leveraging the capabilities of modern database technologies. The key objectives include:

Automation of Reservation Process

To eliminate manual errors and delays by automating ticket booking, cancellation, seat allocation, and schedule management.

Efficient Data Management

To store, organize, and manage large volumes of data related to passengers, trains, routes, and bookings using relational database principles.

Real-Time Availability and Updates

To ensure that seat availability, train schedules, and booking statuses are updated in real-time for accurate and timely access by users.

Improved User Experience

To provide a seamless interface for users to search for trains, check seat availability, book tickets, and make payments with minimal effort.

Data Security and Integrity

To maintain the confidentiality, accuracy, and consistency of data through proper access control and database constraints.

Support for Concurrent Access

To allow multiple users to access and perform transactions on the system simultaneously without data conflicts or corruption.

TECHNIQUES USED

In the development of a Railway Reservation System, various technical techniques and tools are employed to ensure the system is efficient, reliable, and scalable. Below are the primary techniques used in the project:

Relational Database Management System (RDBMS)

The system uses an RDBMS (such as MySQL, PostgreSQL, or Oracle) to store and manage data in structured tables with well-defined relationships. This ensures data integrity, easy retrieval, and efficient storage.

Entity-Relationship (ER) Modeling

ER diagrams are used to design the database structure. Entities such as Passenger, Train, Ticket, Reservation, and Route are defined along with their relationships. This forms the blueprint for creating the actual database schema.

Normalization

The database is normalized (up to 3NF or higher) to eliminate data redundancy and maintain consistency. This helps in optimizing storage and improving query performance.

Structured Query Language (SQL)

SQL is used for creating and manipulating the database. It handles operations like inserting data, retrieving information, updating records, and managing transactions (booking, cancellation, etc.).

CRUD Operations

Basic database operations — Create, Read, Update, and Delete — are implemented for managing records such as passenger profiles, train schedules, and ticket bookings.

Transaction Management

Transactions ensure that each booking or cancellation is completed fully or not at all, maintaining data consistency even in the case of failures or simultaneous access.

Concurrency Control

Techniques such as locking or timestamp ordering are used to handle multiple users accessing the system at the same time, preventing conflicts and ensuring smooth operation.

Frontend and Backend Integration

Frontend: Technologies like HTML/CSS, JavaScript, or frontend frameworks (e.g., React, Angular) are used to design the user interface.

Backend: Server-side scripting languages such as PHP, Python, Java, or Node.js handle the application logic and communicate with the database.

Authentication and Access Control

Basic user roles (e.g., Admin, Passenger) are implemented with secure login systems to restrict unauthorized access and manage permissions effectively.

Data Validation

Input validation techniques are used to prevent incorrect or malicious data from being entered into the system.

SYSTEM CONFIGURATION

1. Hardware Requirements

Component	Minimum Requirement
Processor	Intel Core i3 or equivalent
RAM	4 GB (8 GB or more recommended)
Hard Disk	500 GB (SSD preferred for faster access)
Monitor	1024x768 resolution or higher
Keyboard & Mouse	Standard input devices
Network	Internet connectivity for online features

2. Software Requirements

Software Component	Specification
Operating System	Windows 10 / Linux Ubuntu / macOS
Database	MySQL / PostgreSQL / Oracle DB
Backend Language	Python / Java / PHP / Node.js
Frontend Tools	HTML, CSS, JavaScript (optional: Bootstrap)
Web Server	Apache / Nginx / XAMPP / WAMP
IDE/Editor	VS Code / PyCharm / NetBeans / Sublime Text
Browser	Google Chrome / Mozilla Firefox / Edge

3. Optional Tools (For Advanced Features)

Tool/Framework	Use Case
React / Angular	For building a dynamic, modern frontend interface
Postman	For API testing and backend validation
Git/GitHub	Version control and collaboration

INPUTS:

1. Login

- login_id (Primary Key)
- login_username
- user_password
- login_role_id

2. User

- user_id (Primary Key)
- user_name
- user_mobile
- user_email
- user_address

3. Roles

- role_id (Primary Key)
- role_name
- role_desc

4. Permission

- per_id (Primary Key)
- per_name
- per_module
- per_role_id

5. Trains

- trn_id (Primary Key)
- trn_name
- trn_num
- trn_tckt
- trn_type
- trn_desc

6. Booking

- book_id (Primary Key)
- book_desc
- book_type
- pay_id (Foreign Key from Payment)

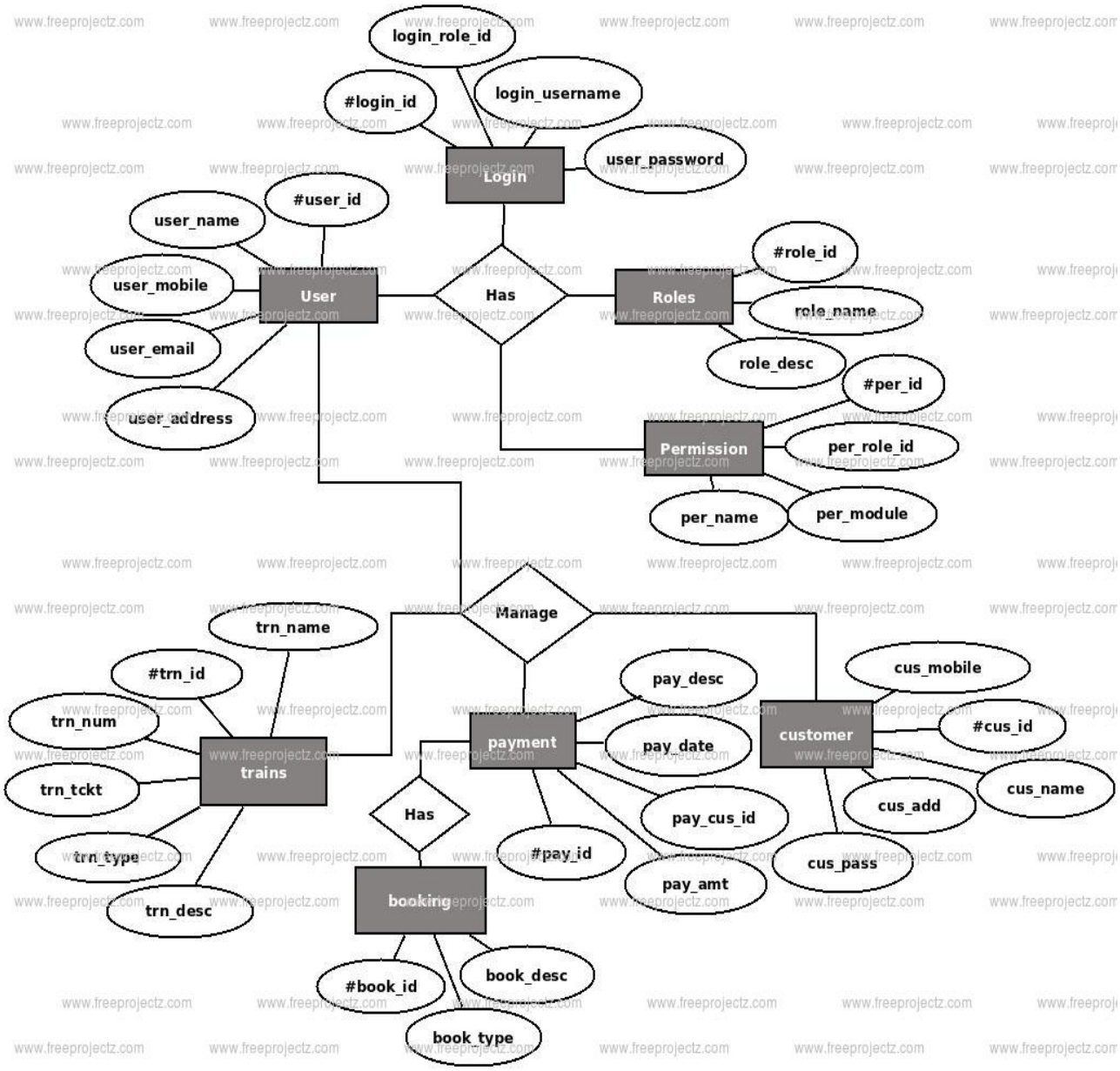
7. Payment

- pay_id (Primary Key)
- pay_desc
- pay_date
- pay_cus_id (Foreign Key from Customer)
- pay_amt

8. Customer

- cus_id (Primary Key)
- cus_name
- cus_pass
- cus_add
- cus_mobile

ER DIAGRAM:



ER Diagram For Railway Reservation System

TABLE RELATION:

- **Roles (1) → Login (Many):**

Explanation:

Each Role (e.g., Admin, User, Manager) can be assigned to many Login accounts.

So, one role can apply to multiple users logging into the system.

Foreign Key: login.login_role_id → roles.role_id

- **Roles (1) → Permission (Many)**

Explanation:

Each Role has access to multiple Permissions (like add, update, delete).

So, one role can have many permissions attached.

Foreign Key: permission.per_role_id → roles.role_id

- **User (1) → Login (1) (Implied)**

Explanation:

Each User corresponds to one Login. This is not explicitly connected in the diagram with a key, but logically, each user would need a login to access the system.

Implied Link: Likely via matching user_name = login_username or through a user_id reference in Login.

- **User (1) → Trains (Many)**

Explanation:

A User (likely an admin or railway staff) can manage multiple Trains.

So, one user enters or manages data for many trains.

Label on diagram: "Manages"

- **User (1) → Payment (Many)**

Explanation:

A User also manages Payments — probably in an administrative capacity.

Label on diagram: "Manages"

- **User (1) → Customer (Many)**

Explanation:

One User can manage multiple Customers. Again, this represents the admin role where users handle customer records.

Label on diagram: "Manages"

- **Customer (1) → Payment (Many)**

Explanation:

Each Customer can make multiple payments for ticket bookings.

Foreign Key: payment.pay_cus_id → customer.cus_id

- **Payment (1) → Booking (Many)**

Explanation:

A single Payment can be linked to many Bookings — for example, if a customer books multiple tickets in a single transaction.

Foreign Key: booking.pay_id → payment.pay_id

- **(Implied) Trains (1) → Booking (Many)**

Explanation:

Although not directly connected in the diagram, it's logical that one Train will have many Bookings. This is a natural part of a reservation system.

SUMMARY TABLE:

From Entity	To Entity	Cardinality	Meaning
Roles	Login	1 to Many	One role for many logins
Roles	Permission	1 to Many	One role has many permissions
User	Login	1 to 1	One user has one login (assumed)
User	Trains	1 to Many	One user manages many trains
User	Payment	1 to Many	One user manages many payments
User	Customer	1 to Many	One user manages many customers
Customer	Payment	1 to Many	One customer makes

TABULAR FORMAT:

1. Roles:

Column Name	Data Type	Key
role_id	INT	PK
role_name	VARCHAR(50)	
role_desc	TEXT	

2. Login:

Column Name	Data Type	Key
login_id	INT	PK
login_username	VARCHAR(50)	
user_password	VARCHAR(50)	
login_role_id	INT	FK → Roles(role_id)

3. Permission:

Column Name	Data Type	Key
per_id	INT	PK
per_name	VARCHAR(50)	
per_module	VARCHAR(50)	
per_role_id	INT	FK → Roles(role_id)

4. User:

Column Name	Data Type	Key

user_id	INT	PK
user_name	VARCHAR(100)	
user_mobile	VARCHAR(15)	
user_email	VARCHAR(100)	
user_address	TEXT	

5. Trains:

Column Name	Data Type	Key
trn_id	INT	PK
trn_name	VARCHAR(100)	
trn_num	VARCHAR(20)	
trn_tckt	INT	
trn_type	VARCHAR(50)	
trn_desc	TEXT	
user_id	INT	FK → User(user_id)

6. Customer:

Column Name	Data Type	Key
cus_id	INT	PK
cus_name	VARCHAR(100)	
cus_pass	VARCHAR(50)	
cus_add	TEXT	
cus_mobile	VARCHAR(15)	
user_id	INT	FK → User(user_id)

7. Payment:

Column Name	Data Type	Key
pay_id	INT	PK
pay_desc	TEXT	
pay_date	DATE	
pay_amt	DECIMAL(10,2)	
pay_cus_id	INT	FK → Customer(cus_id)
user_id	INT	FK → User(user_id)

8. Booking:

Column Name	Data Type	Key
book_id	INT	PK
book_desc	TEXT	
book_type	VARCHAR(50)	
pay_id	INT	FK → Payment(pay_id)
trn_id	INT	FK → Trains(trn_id)

TABLE CREATION

1. Roles Table:

```
CREATE TABLE Roles (
role_id INT PRIMARY KEY,
role_name VARCHAR(50),
role_desc TEXT
);
```

2. Login Table:

```
CREATE TABLE Login (
login_id INT PRIMARY KEY,
login_username VARCHAR(50),
user_password VARCHAR(50),
login_role_id INT,
FOREIGN KEY (login_role_id) REFERENCES Roles(role_id)
);
```

3. Permission Table:

```
CREATE TABLE Permission (
per_id INT PRIMARY KEY,
per_name VARCHAR(50),
per_module VARCHAR(50),
per_role_id INT,
FOREIGN KEY (per_role_id) REFERENCES Roles(role_id)
);
```

4. User Table:

```
CREATE TABLE User (
user_id INT PRIMARY KEY,
user_name VARCHAR(100),
user_mobile VARCHAR(15),
user_email VARCHAR(100),
user_address TEXT
);
```

5. Trains Table:

```
CREATE TABLE Trains (
trn_id INT PRIMARY KEY,
trn_name VARCHAR(100),
trn_num VARCHAR(20),
trn_tckt INT,
trn_type VARCHAR(50),
trn_desc TEXT,
user_id INT,
FOREIGN KEY (user_id) REFERENCES User(user_id)
);
```

6. Customer Table:

```
CREATE TABLE Customer (
cus_id INT PRIMARY KEY,
cus_name VARCHAR(100),
cus_pass VARCHAR(50),
cus_add TEXT,
cus_mobile VARCHAR(15),
```

```
user_id INT,

FOREIGN KEY (user_id) REFERENCES User(user_id)

);
```

7. Payment Table:

```
CREATE TABLE Payment (

pay_id INT PRIMARY KEY,

pay_desc TEXT,

pay_date DATE,

pay_amt DECIMAL(10, 2),

pay_cus_id INT,

user_id INT,

FOREIGN KEY (pay_cus_id) REFERENCES Customer(cus_id),

FOREIGN KEY (user_id) REFERENCES User(user_id)

);
```

8. Booking Table:

```
CREATE TABLE Booking (

book_id INT PRIMARY KEY,

book_desc TEXT,

book_type VARCHAR(50),

pay_id INT,

trn_id INT,

FOREIGN KEY (pay_id) REFERENCES Payment(pay_id),

FOREIGN KEY (trn_id) REFERENCES Trains(trn_id)


);
```

1.List all users:

```
SELECT * FROM User;
```

user_id	user_name	user_mobile	user_email	user_address
1	Admin1	9876543210	admin1@mail.com	Delhi
2	Manager1	9123456789	mgr1@mail.com	Mumbai

2. Show all roles and their descriptions:

SELECT * FROM Roles;

role_id	role_name	role_desc
1	Admin	System Admin
2	Manager	Train Manager

3. Find all login usernames and their role:

SELECT l.login_username, r.role_name

FROM Login l

JOIN Roles r ON l.login_role_id = r.role_id;

login_username	role_name
admin1	Admin
mgr1	Manager

4. List all customers managed by User 2:

SELECT cus_name, cus_mobile

FROM Customer

WHERE user_id = 2;

cus_name	cus_mobile
Alice	9876500000

5. Show all trains managed by Admin1:

SELECT t.trn_name

```
FROM Trains t

JOIN User u ON t.user_id = u.user_id

WHERE u.user_name = 'Admin1';
```

trn_name
Rajdhani Express

6. Show bookings and related train names:

```
SELECT b.book_id, t.trn_name, b.book_type

FROM Booking b

JOIN Trains t ON b.trn_id = t.trn_id;
```

book_id	trn_name	book_type
101	Rajdhani Express	AC
102	Shatabdi	Sleeper

7. Total amount paid by customer John:

```
SELECT c.cus_name, SUM(p.pay_amt) AS total_paid

FROM Payment p

JOIN Customer c ON p.pay_cus_id = c.cus_id

WHERE c.cus_name = 'John'

GROUP BY c.cus_name;
```

cus_name	total_paid
John	1500.00

8. Get permissions of the Manager role:

```
SELECT p.per_name, p.per_module

FROM Permission p

JOIN Roles r ON p.per_role_id = r.role_id
```

```
WHERE r.role_name = 'Manager';
```

per_name	per_module
View	Trains
Add	Booking

9. List all bookings with their payment date:

```
SELECT b.book_id, b.book_type, p.pay_date

FROM Booking b

JOIN Payment p ON b.pay_id = p.pay_id;
```

book_id	book_type	pay_date
101	AC	2025-04-01
102	Sleeper	2025-04-10

10. Customers who paid more than ₹800:

```
SELECT DISTINCT c.cus_name, p.pay_amt

FROM Customer c

JOIN Payment p ON p.pay_cus_id = c.cus_id

WHERE p.pay_amt > 800;
```

cus_name	pay_amt
Alice	1000.00

11. Number of trains managed by each user:

```
SELECT u.user_name, COUNT(t.trn_id) AS train_count

FROM User u

LEFT JOIN Trains t ON u.user_id = t.user_id

GROUP BY u.user_name;
```

user_name	train_count
Admin1	1
Manager1	1

12. List all train names and total tickets:

```
SELECT trn_name, trn_tckt

FROM Trains;
```

trn_name	trn_tckt
Rajdhani Express	120
Shatabdi	100

13. Bookings done for 'Rajdhani Express':

```
SELECT b.book_id, b.book_type

FROM Booking b

JOIN Trains t ON b.trn_id = t.trn_id

WHERE t.trn_name = 'Rajdhani Express';
```

book_id	book_type
101	AC

14. Total number of bookings per train:

```
SELECT t.trn_name, COUNT(b.book_id) AS bookings

FROM Trains t

LEFT JOIN Booking b ON t.trn_id = b.trn_id

GROUP BY t.trn_name;
```

trn_name	bookings
Rajdhani Express	1

Shatabdi	1

15. List payments made on or after April 1, 2025:

SELECT pay_id, pay_amt, pay_date

FROM Payment

WHERE pay_date >= '2025-04-01';

pay_id	pay_amt	pay_date
1	500.00	2025-04-01
2	1000.00	2025-04-10

SUMMARY

The Railway Reservation System is a robust and efficient software solution developed to handle the complexities of train ticket booking, customer management, and administrative control within a digital environment. The system is designed to replace the traditional, manual method of railway reservations with an automated platform that ensures accuracy, security, and scalability. It caters to various stakeholders such as administrators, managers, and passengers (customers), each having a distinct role with specific access and permissions in the system.

At its core, the system provides functionalities for user login, role and permission management, train information management, customer registration, booking operations, and payment processing. Administrators and managers (referred to as Users in the database schema) have access to modules for creating and updating train schedules, handling customer queries, overseeing booking processes, and monitoring payments. The Login and Roles modules ensure secure access control, where each user is authenticated and provided with the appropriate level of access using Permissions defined for each role.

Customers can interact with the system by registering their details, logging in securely, browsing available trains, and booking seats based on real-time availability. Once a booking is made, the customer proceeds to the Payment module to complete the transaction. Each payment is tied to a booking and a customer, ensuring traceability and accountability. The Booking table connects the train, the payment, and the customer together in a seamless flow of transactional data.

The underlying ER diagram and database schema are designed using principles of normalization, ensuring minimal redundancy and maintaining data integrity through proper primary and foreign key relationships. The system supports a variety of SQL operations for reporting and analytics, such as tracking the number of bookings per train, calculating revenue generated, listing users by roles, identifying top customers, and monitoring payment histories.

This project serves not only as a practical solution for railway ticketing but also as a strong portfolio piece showcasing skills in database design, SQL querying, role-based access control, and application development.

CONCLUSION

The Railway Reservation System project marks a significant step toward the digital transformation of railway services. It demonstrates how technology can be leveraged to optimize and automate critical operations such as train scheduling, customer management, seat booking, and payment processing. The project not only simplifies the workflow for both passengers and railway administrators but also ensures the security and integrity of data through a well-designed relational database structure.

By implementing role-based access control, the system guarantees that only authorized personnel can manage sensitive modules, such as booking records, payment details, and train configurations. This layered access structure enhances security, ensures accountability, and minimizes the chances of data misuse. Moreover, the inclusion of permissions and login modules adds a professional and scalable dimension to the system, allowing for easy extension in larger enterprise settings.

The use of SQL and normalized database design principles ensures that data is organized, efficient, and free from redundancy. With multiple query possibilities, administrators and analysts can extract meaningful insights from the system—such as identifying the most booked trains, monitoring customer activity, or analyzing revenue patterns. These analytics can directly support strategic decisions for railway authorities.

From a learning and development perspective, this project reflects a strong understanding of database management systems, data modeling, SQL queries, and system design principles. It also demonstrates how multiple interconnected modules can function in harmony to deliver a complete, end-to-end solution.

Overall, the Railway Reservation System is not just an academic project but a real-world applicable solution that brings convenience, accuracy, and efficiency to the railway ticketing domain. With minor enhancements like live tracking, refund modules, and online payment gateway integration, this system can be made fully production-ready. It serves as a valuable asset in promoting digital infrastructure and improving public transportation systems through smart technology.