| | | |
|---|---|---|
| **Course Code** | **:** | **MCS-014** |
| **Course Title** | **:** | **Systems Analysis and Design** |
| **Assignment Number** | **:** | **BCA(III)/014/Assignment/2024-25** |
| **Maximum Marks** | **:** | **100** |
| **Weightage** | **:** | **25%** |
| **Last Dates for Submission** | **:** | **31st October,2024(For July Session)** |
| | | **30th April,2025(For January Session)** |

**This assignment has three questions of 80 marks. Rest 20 marks are for viva voce. Answer all questions. You may use illustrations and diagrams to enhance the explanations. Please go through the guidelines regarding assignments given in the Programme Guide for the format of presentation.**

**Q1.** Develop SRS for **Railway Reservation System .** SRS should be as per IEEE standard SRS template. Make necessary assumptions. **(30 Marks)**

**Q2.** Draw the DFDs upto 3rd level for **Railway ReservationSystem.** **(30 Marks)**

**Q3.** Draw ERD for an **Railway Reservation System**. Make necessary assumptions. **(20 Marks)**

**Q1**. Develop SRS for Railway Reservation System . SRS should be as per IEEE standard SRS template. Make necessary assumptions.

**Ans Software Requirements Specification (SRS)**

## 1. Introduction

### 1.1 Purpose

The purpose of this document is to specify the requirements for a Railway Reservation System (RRS). The RRS will provide a platform for passengers to reserve train tickets, view schedules, and manage bookings. This document outlines the system's functionalities, interfaces, performance, and design constraints.

### 1.2 Scope

The RRS will allow users to:

- Search for trains between specific source and destination stations.
- Reserve tickets for available trains.
- Cancel or modify reservations.
- View train schedules and availability.
- Generate reports for administrative purposes.

### 1.3 Definitions, Acronyms, and Abbreviations

- **RRS:** Railway Reservation System
- **UI:** User Interface
- **API:** Application Programming Interface
- **DBMS:** Database Management System

### 1.4 References

- IEEE Std 830-1998, "IEEE Recommended Practice for Software Requirements Specifications."

### 1.5 Overview

This document is organized into several sections: an introduction to the project, a description of the overall system, specific requirements, and any additional appendices that might be relevant.

## 2. Overall Description

### 2.1 Product Perspective

The RRS is a standalone application that interacts with external systems like payment gateways and a central train schedule database. The system will support multiple user roles, including passengers, administrators, and ticketing agents.

### 2.2 Product Functions

- **User Registration and Authentication:** Users can register, log in, and manage their profiles.
- **Search Trains:** Users can search for trains based on source, destination, and travel date.
- **Booking Management:** Users can book, view, cancel, and modify reservations.
- **Payment Processing:** Secure payment gateway integration for ticket purchases.
- **Schedule Viewing:** Users can view train schedules and seat availability.
- **Reporting:** Administrators can generate reports on bookings, cancellations, and revenue.

### 2.3 User Characteristics

- **Passengers:** General users looking to book or manage reservations.
- **Administrators:** Manage train schedules, monitor system performance, and generate reports.
- **Ticketing Agents:** Assist passengers with bookings and cancellations.

### 2.4 Constraints

- The system must comply with GDPR for data protection.
- The system must support high traffic, especially during peak hours.
- Payment processing must adhere to industry security standards.

### 2.5 Assumptions and Dependencies

- Users have access to the internet and basic knowledge of using a web application.
- The system relies on external train schedule databases and payment gateways.

## 3. Specific Requirements

### 3.1 Functional Requirements

#### 3.1.1 User Registration and Authentication

- **Description:** Users must register with a valid email address and password. The system will provide authentication using these credentials.
- **Input:** User details (name, email, password, etc.).

- **Output:** Confirmation of registration or login.

### 3.1.2 Train Search

- **Description:** Users can search for trains by selecting source, destination, and date of travel.
- **Input:** Source station, destination station, date.
- **Output:** List of available trains with details like timing and seat availability.

### 3.1.3 Booking Management

- **Description:** Users can book tickets, view current bookings, cancel, or modify them.
- **Input:** Train details, passenger details, payment information.
- **Output:** Booking confirmation or cancellation.

### 3.1.4 Payment Processing

- **Description:** The system will process payments via a secure payment gateway.
- **Input:** Payment details (card number, expiry date, etc.).
- **Output:** Payment confirmation or error message.

### 3.1.5 Schedule Viewing

- **Description:** Users can view detailed train schedules and seat availability.
- **Input:** Train number or station details.
- **Output:** Schedule and seat availability.

### 3.1.6 Reporting

- **Description:** Administrators can generate reports on bookings, cancellations, and system usage.
- **Input:** Date range or specific criteria.
- **Output:** Report in PDF or Excel format.

## 3.2 Performance Requirements

- **Response Time:** The system should respond to user actions within 2 seconds.
- **Scalability:** The system should support up to 10,000 concurrent users.
- **Availability:** The system should be available 99.9% of the time.

## 3.3 Design Constraints

- The system must be developed using the MERN stack (MongoDB, Express.js, React.js, Node.js).
- The system should be compatible with major web browsers.

**3.4 Software System Attributes**

**3.4.1 Security**

- All user data must be encrypted, both in transit and at rest.
- Multi-factor authentication should be implemented for administrators.

**3.4.2 Maintainability**

- The codebase should be modular to facilitate easy updates and debugging.
- Documentation must be provided for each module.

**3.4.3 Usability**

- The UI should be intuitive and accessible, following the WCAG 2.1 guidelines.
- The system should support multiple languages.

**4. Appendices**

**4.1 Appendix A: Use Case Diagrams**

[Include any use case diagrams here]

**4.2 Appendix B: Glossary**

- **Booking:** The process of reserving a seat on a train.
- **Cancellation:** The process of voiding a previously made booking.
- **Schedule:** The timetable of trains.

**Q2. Draw the DFDs upto 3 rd level for Railway ReservationSystem.**

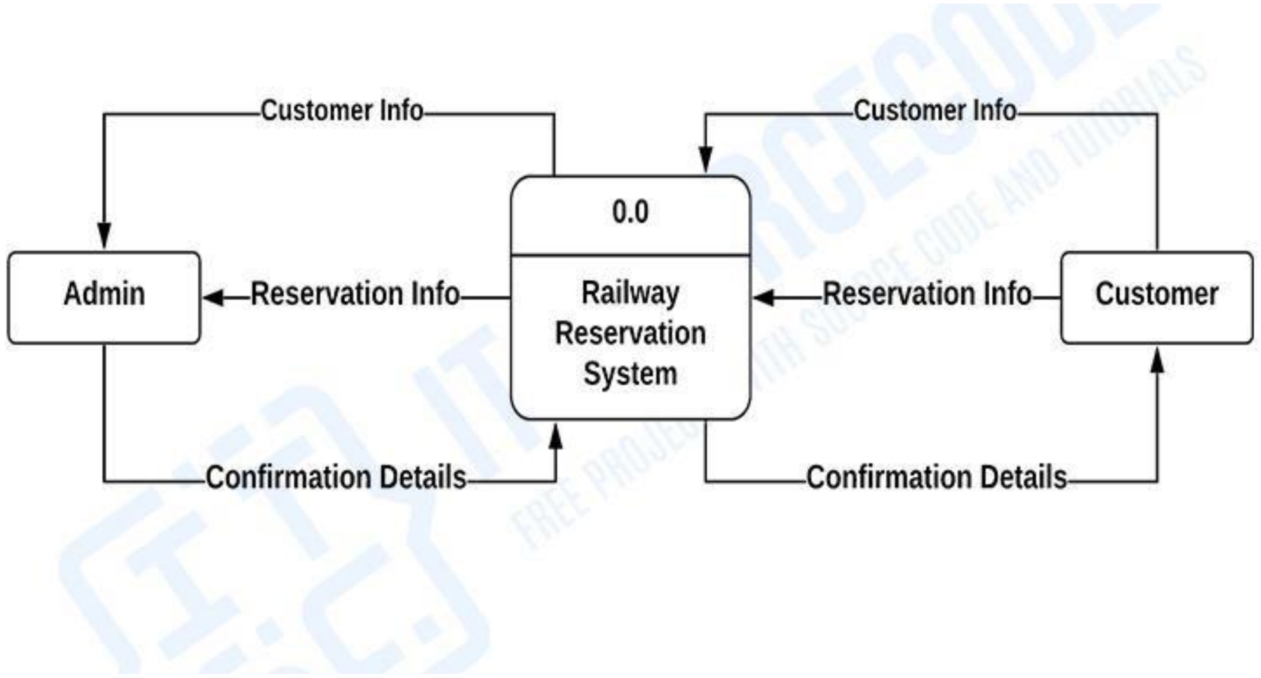
**Ans Railway Reservation System Data Flow Diagram (DFD)**

A thorough explanation is provided for the example **data flow diagram for railway reservation system**. This example emphasizes the three **DFD levels (DFD Levels 0, 1, and 2)**.

**0 Level DFD for Railway Reservation System**

The context diagram is an alternative name for the **Level 0 DFD Diagram for Railway Reservation System**. Users, the main process, and data flow make up its parts. Also, the project concept is demonstrated using the single process visualization.

DFD Level 0 shows the entities that interact with a system and defines the border between the system and its environment. This diagram also depicts the railway reservation system at a high level.



*0 Level DFD for Railway Reservation System*

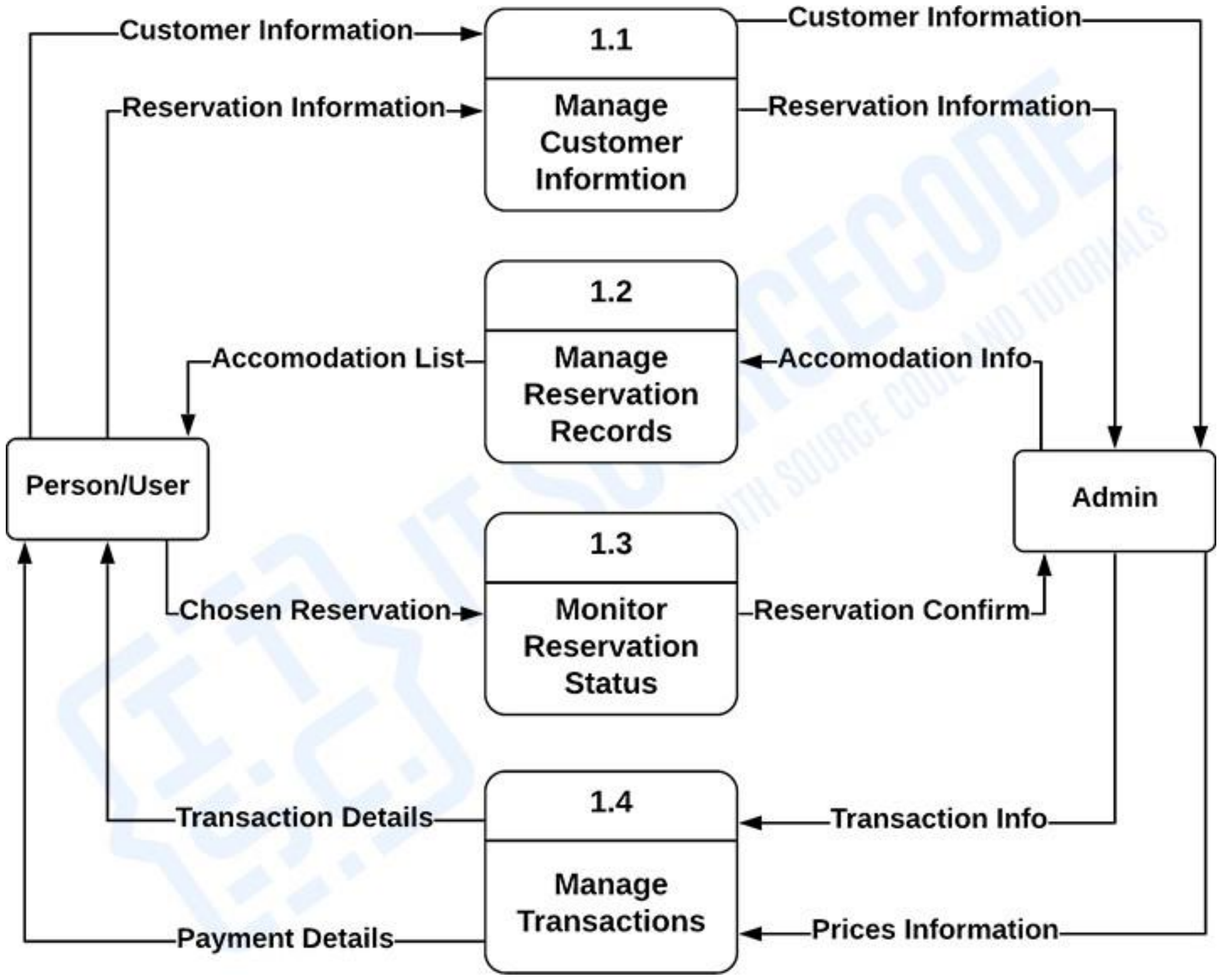
The illustration presents the main process in a single node to introduce the project context. This context explains how the project works in just one look. The user feeds data into the system and then receives the output from it.

In addition to this, you will perceive through the diagram that there is already the presence of data flow. Though the process is very general, the flow of data is clear. Nevertheless, just modify this diagram to meet the other requirements and include other matters regarding railway reservations.

**Level 1 DFD for Railway Reservation System**

The "detonated view" of the context diagram is **Railway Reservation System DFD Level 1**. Its function is to deepen the concept derive from the context diagram.

Specifically, level 1 shows the broader details of Railway Reservation System DFD Level 0. This is to clarify the paths (flow) of data and its transformation from input to output.



*Level 1 DFD for Railway Reservation System*

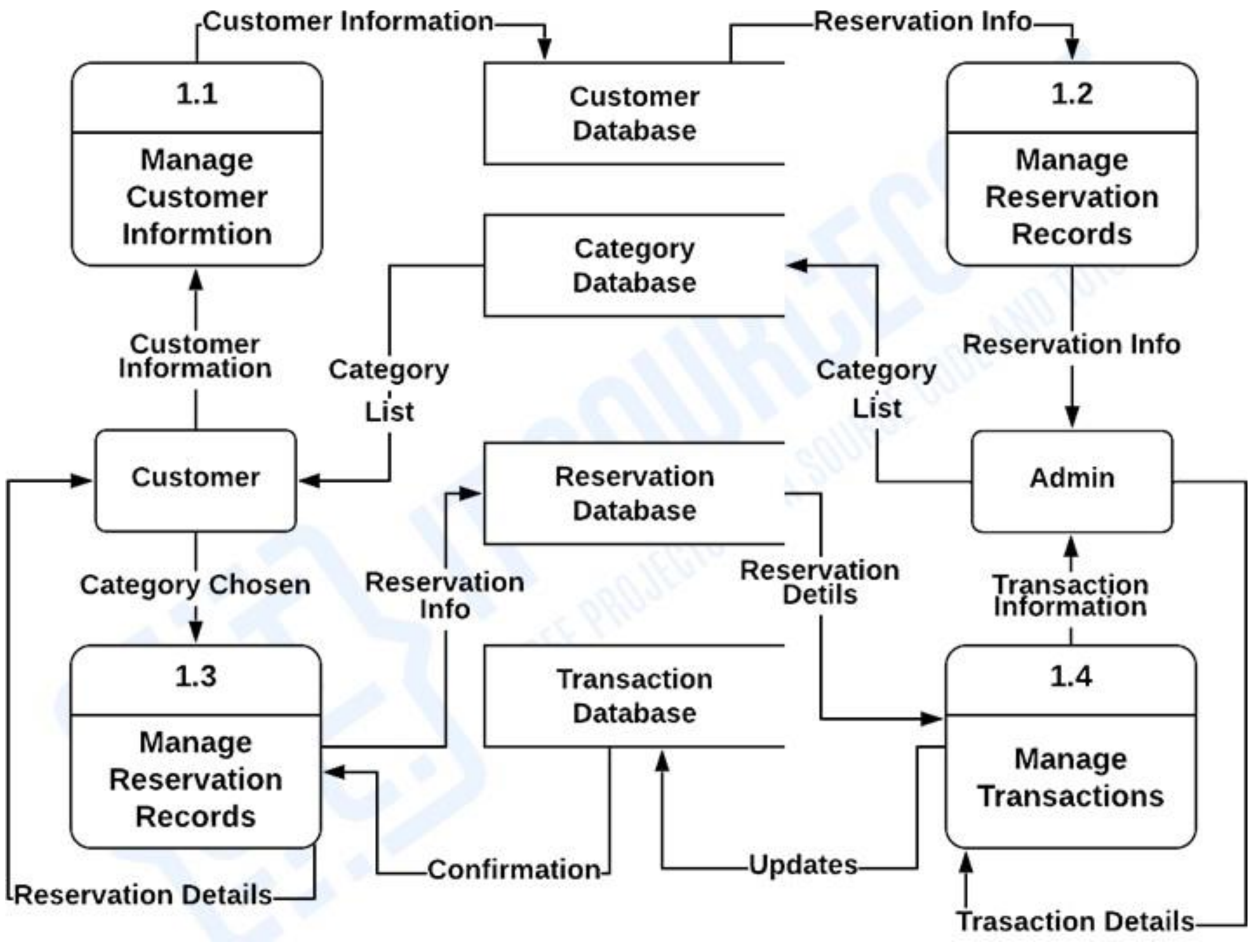
The designed diagram portrays four different scenarios: customer information management, reservation records management, reservation status monitoring, and transaction management

Firstly, the flow of data starts from the railway admin, crews, and customers. Then the system caters to the transaction. This idea was based on railway reservation processes.

You can also see the data store used or the database. The database is also used in storing users' data inputs. Then it serves as the source of outputs.

**DFD Level 2 for Railway Reservation System**

**Level 2 DFD for Railwy Reservation System** is also the highest abstraction of the data flow diagram. This level also broadens the idea from the DFD level 1. It includes the sub-processes from level 1 as well as the data that flows.



*DFD Level 2 for Railway Reservation System*

However, not all of the processes in the project must have sub-processes. Only provide this diagram if needed. As long as your previous diagrams were clear and precise, this level is not required.

You can add more to this and it is up to you how will you create your data flow diagram.

Also, consider the data flow included and be precise with your information.

**Q3** Draw ERD for an Railway Reservation System. Make necessary assumptions.

**Entities:**

1. **Passenger**
   - Passenger_ID (Primary Key)
   - Name
   - Age
   - Gender
   - Contact_Number
   - Email
2. **Train**
   - Train_ID (Primary Key)
   - Train_Name
   - Source
   - Destination
   - Number_of_Seats
   - Train_Type
3. **Reservation**
   - Reservation_ID (Primary Key)
   - Reservation_Date
   - Travel_Date
   - Seat_Number
   - Class (e.g., Sleeper, AC, General)
   - Fare
   - Status (Confirmed/Waiting/Canceled)
   - Passenger_ID (Foreign Key referencing Passenger)
   - Train_ID (Foreign Key referencing Train)
4. **Payment**
   - Payment_ID (Primary Key)
   - Payment_Date
   - Amount
   - Payment_Method (e.g., Credit Card, Debit Card, Net Banking)

    - Reservation_ID (Foreign Key referencing Reservation)

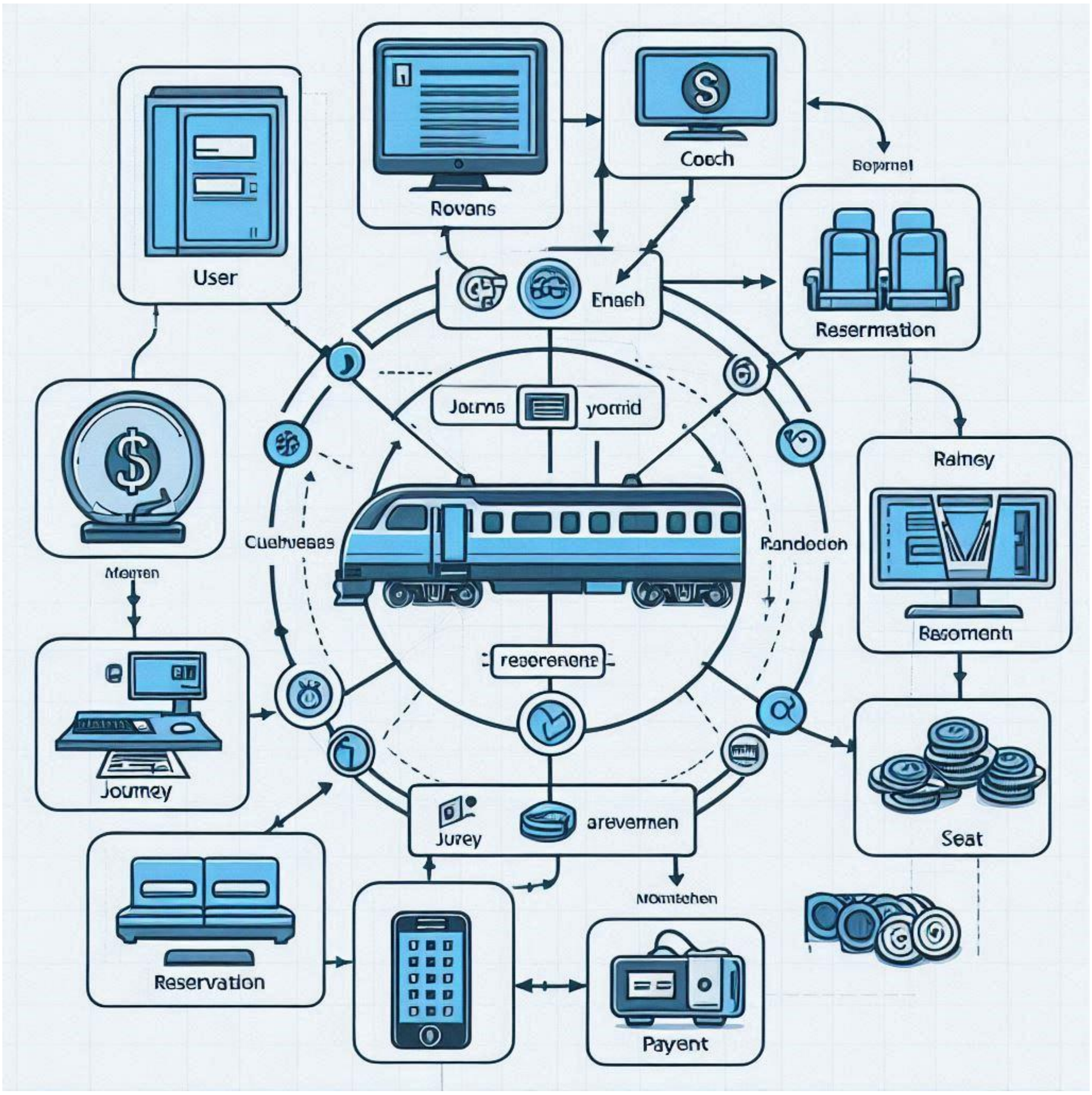
5. **Route**
    - Route_ID (Primary Key)
    - Train_ID (Foreign Key referencing Train)
    - Station_Name
    - Arrival_Time
    - Departure_Time
6. **Station**
    - Station_ID (Primary Key)
    - Station_Name
    - Location

**Relationships:**

1. **Passenger to Reservation**: One-to-Many (One passenger can make multiple reservations)
2. **Train to Reservation**: One-to-Many (One train can have multiple reservations)
3. **Reservation to Payment**: One-to-One (Each reservation has one payment)
4. **Train to Route**: One-to-Many (One train can have multiple routes)
5. **Route to Station**: Many-to-One (Multiple routes can involve the same station)

**ERD Explanation:**

- **Passenger**: This entity stores information about the passengers who make reservations.
- **Train**: This entity holds details of the trains available for booking.
- **Reservation**: This entity records the details of each booking made by passengers.
- **Payment**: This entity captures payment details associated with reservations.
- **Route**: This entity maps the routes that trains take, including station stops.
- **Station**: This entity keeps track of the stations where trains can stop.



Rovens
Coach
Boprnel
User
Enash
Reservation
Joums
yomid
Raimey
Cuelweses
Randodoh
Moeten
Basoment
reseorenens
Journey
Jurey
arevemen
Seat
Reservation
Momtchen
Payent