

RAILWAY MANAGEMENT SYSTEM

A MINI PROJECT REPORT

Submitted by

VAKAMALA SRI RAM CHARAN REDDY(RA2011003010900)

ANURAG RISWADKAR (RA2011003010930)

VAIDEHI JADHAO(RA2011003010958)

Under the guidance of

Ms. Viji.D

(Assistant professor, Department
of Computing Technologies)

In partial satisfaction of the requirements for the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING



SCHOOL OF COMPUTING

COLLEGE OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR - 603203

APRIL 2023

BONAFIDE CERTIFICATE

Certified that this project report “**RAILWAY MANAGEMENT SYSTEM**” is the bonafide work of “**VAKAMALA SRI RAM CHARAN REDDY(RA2011003010900), ANURAG RISWADKAR(RA2011003010930), VAIDEHI JADHAO(RA2011003010958)**” of III Year/VI Sem B.tech(CSE) who carried out the mini project work under my supervision for the course 18CSC303J- Database Management systems in SRM Institute of Science and Technology during the academic year 2022-2023(Even sem).

SIGNATURE

Dr. Pushpalatha M
HOD School of Computing

SIGNATURE

Ms. Viji D
Assistant Professor
Department of Computing

ABSTRACT

The main purpose of maintaining database for railway reservation system is to reduce the manual errors involved in the booking and cancellation of tickets and make it convenient for the customers and providers to maintain the data about their customers and also the seats available at them. The purpose of Railway Reservation System is to automate the existing manual system by the help of computerized equipment and full-fledged computer software, fulfilling their requirements, so that their valuable data/information can be stored for a longer period with easy accessing and manipulation of the same. The required software and hardware are easily available and easy to work with. The railway system is going to incorporate the new feature of manual selection of favorable seats which is not currently offered by the booking facilities of the Central Government. This system is inspired from the airline industry which regularly uses this facility to choose seats. Currently the people have to go to the TTE to change seats. Our reservation system would allow people to modify their seat in-transit by using the option of a passenger to change to an empty seat. The empty seats will be modified at each station. Railway Reservation System, as described above, can lead to error free, secure, reliable and fast management systems. It can assist the user to concentrate on their other activities rather than concentrating on the record keeping. Thus it will help organizations in better utilization of resources. The organization can maintain computerized records without redundant entries. That means that one need not be distracted by information that is not relevant, while being able to reach the information. The aim is to automate its existing manual system by the help of computerized equipment and full-fledged computer software, fulfilling their requirements, so that their valuable data/information can be stored for a longer period with easy accessing and manipulation of the same. Basically the project describes how to manage for good performance and better services for the clients.

Table of content

Chapter No	Title	Page No
	Abstract	iii
	Table of content	iv
	List of Figures	v
	List of Tables	vi
1	Introduction	
1.1	Introduction	1
1.2	Problem Statement	2
1.3	Objectives	3
1.4	Scope and applications	3
1.5	General and Unique Services in the database application	3
1.6	Software Requirements Specification	4
1.7	Advantages of Using DBMS	5
2	Literature Survey	
2.1	Existing system	6
3	System Architecture and Design	
3.1	Architecture Diagram	10
3.2	Entity Relationship Diagram	11
4	Modules and Functionalities	
4.1	Front End Design	14
4.2	Back-end Design	15
4.3	List of modules in Railway Reservation System	15
4.4	DataBase Connectivity	16
4.5	MySQL Database	16
5	Coding and Testing	18
6	Result and Discussion	
6.1	Creating New User	40

6.2	Enquiry	41
6.3	Cancellation	43
7	Conclusion and Future Enhancement	
7.1	Conclusion	44
7.2	Future Enhancement	44
8	References	45

List of figures

Figure.No	Figure Name	Page.No
2.1.2	Working of the UID-based Reservation System in Indian Railways	7
3.1	System Architecture diagram	10
3.2	ER diagram for Railway management system	11
4.1	Front End Design	14
4.2	Back-end and Database	15

List of Tables

Table.no	Table name	Page.no
2.1	Train Database after adding Aadhar card number	6
3.2	Relationship between entities	13

CHAPTER 1

Introduction:

A railway management system is a software platform used to manage various aspects of railway operations. It is designed to improve efficiency, safety, and overall performance of railway systems by providing real-time information and data analysis.

The railway management system includes modules that track train schedules, manage ticketing and reservations, monitor train movements and control signals, and ensure safety protocols are followed. It also manages maintenance schedules, repairs, and inspections of the railway infrastructure, such as tracks, signals, and locomotives.

The system uses advanced technologies, such as GPS and RFID, to monitor train positions and speed, track maintenance activities, and provide real-time information to passengers about train schedules, delays, and cancellations. Additionally, it can generate reports and analytics to help railway management make informed decisions about resource allocation, capacity planning, and performance improvement.

Overall, a railway management system helps railway companies optimize their operations and improve customer satisfaction by providing timely and accurate information to all stakeholders, including passengers, employees, and management.

1.2 PROBLEM STATEMENT:

The proposed railway management system aims to provide passengers with a more comfortable and hassle-free travel experience. The system will allow passengers to quickly reserve and manage their seats, making it easy for them to plan their journey and avoid last-minute rush. This will be accomplished through the integration of flexible seat reservation and UID verification system in-train.

The flexible seat reservation system will allow passengers to choose their preferred seats and travel class, and also make changes to their reservations as per their convenience. This will not only reduce the waiting time at the ticket counter but also provide a more personalized experience to the passengers.

Moreover, the UID verification system will be used to ensure the security of the passengers while traveling. This system will require passengers to verify their identity using a unique identification number, which will be generated at the time of booking. This will help to prevent any fraudulent activity and enhance the safety of the passengers.

In addition, the system will provide real-time information about train arrival and departure times, route schedules, and fare information. This will enable passengers to plan their journey better and avoid any inconvenience due to delays or changes in the schedule. The information will be accessible through a mobile application, website, or at the railway stations, making it convenient for passengers to access it from anywhere.

Overall, the development of an electronic rail ticket reservation system will significantly improve the travel experience for passengers. It will reduce the waiting time at ticket counters, provide more personalized services, enhance the security of the passengers, and offer real-time information about the journey. The implementation of such a system is a step towards modernizing the railway infrastructure and meeting the changing needs of the passengers.

1.3 OBJECTIVES

The project will be built in AWS platform and will make use of the lambda function and S3 buckets to store the details.

- Creation of a railway management system which will allow passengers to easily book and manage their seats thus making their travel more comfortable by showing available seats during transit.
- It manages all the information about train, seat, payment, Train details and uses a postgresql database.
- The project is totally built at administrative end and thus only the administrator is granted the access.
- The purpose of the project is to build an application program to enhance the current available system and make the system more streamlined.
- It tracks all the details about the booking, customer, payment etc.

1.4 SCOPE OF THE PROJECT:

Our project aims at stream-lining the ticket reservation system that is currently available and to add additional features such as seat selection and upgradation.

- In the computer system, we fill all of the details manually, the irctc system has been able to relieve some of the problems but is still a complicated system to book tickets. Our system is easier to use and more flexible. IRCTC assigns seats on its own but in the proposed system we can choose our own seat.
- It will reduce workload of the staff
- Expandible
- Easy to use
- Good interface
- Satisfies user requirement

1.5 LIST OF GENERAL AND UNIQUE SERVICES

- In the database, information about the booked and unbooked seats will be updated every time the new seat is booked, which will allow users to book a seat according to their seat number preference.
- Booking of tickets
- Display of schedule
- Cancellation of Tickets
- Enquiry of Train Status

1.6 HARDWARE AND SOFTWARE REQUIREMENTS

Hardware requirement:

1. A desktop or laptop with a proper internet connection
2. 2 500GB or 60GB of the hard disk
3. 3.4GB 2GB of the RAM
4. 4 Windows 7 or 8 or 10 Operating system

Software Requirements:

Server side

1. Programming language: PHP 5.6.31
2. Web Server: Apache 2.4.27
3. Database: SQL 5.7.19

Client side

1. Programming language: JAVASCRIPT, HTML, CS
2. OS: windows7/8/10
3. MYSQL serve

1.7 ADVANTAGES OF USING DBMS

- **Efficient Data Access:** A DBMS utilizes a variety of sophisticated techniques to store and retrieve data efficiently. This feature is especially important if the data is to be stored on an external device.
- **Data integrity and security:** if data is always accessed through DBMS, the DBMS can enforce integrity constraints. For example, before inserting salary information for an employee, the DBMS can check that the department budget is not exceeded. Also, it can enforce access controls that govern what data is visible to different classes of users.
- **Data Administration:** when several users share data, centralizing the administration of data can offer significant improvements. Experienced professionals who understand the nature of the data being managed, and how different groups of users use it, can be responsible for organizing the data representation to minimize redundancy and for fine-tuning the storage of the data to make retrieval efficient.
- **Concurrent Access and Crash Recovery:** A DBMS schedules concurrent accesses to the data in such a manner that users can think of the data as being accessed by only one user at a time. Further, the DBMS protects users from the effects of system failures.
- **Reduced Application Development Time:** clearly, the DBMS supports important functions that are common to many applications accessing data in the DBMS. This, in conjunction with the high-level interface to data, facilitates quick application development. DBMS applications are also likely to be more robust than a similar stand-alone application because many important tasks are handled by the DBMS.

CHAPTER 2

LITERATURE SURVEY:

2.1 Existing system:

Ref.No:1->Railway Reservation Verification by Aadhar Card

(by Rajneesh Tanwar,Ahmad Khalid Nazari,Vikas Deep,Naveen Garg)

In today's world, as we know, the railway is one of the biggest networks which is used for connecting different areas. But this network is not secure enough as there is no verification of passengers. Anyone can register a ticket and travel by train. Security issues are also there as many passengers having tickets enter the platform without going through security checks.

For making this network secure and verified, using Aadhar Card Number is one of the best solutions. By using this, security and verification of all passengers can be done. Ticket verification will be done at the railway station by going through a biometric check which will confirm the details filled by the passenger and after verification from the Aadhar database, the passenger will get the confirmation message. This biometric check will be done at a security checkpoint so that all passengers have to go through a security check.

Implementation of Verification of passenger through aadhar card number includes following stage: Addition of Aadhar card number in database for railway reservation and biometric test for verification. Addition of Aadhar Card Number in Database At the time of reservation of ticket for verification only aadhar card number is necessary to add if passenger nationality is India otherwise enter passport number. Aadhar card number will also going to store in database for future reference and this will also help in verifying the passenger travelling. Database will now look like

Train No.	Name	Age	Aadhar Card Number	Mobile Number	Set No.	Time For verification	Gender
1024	Deep Singh	33	123456783456	9998988954	B2	10:00 am	Male
1024	Komal Rana	28	987654321567	9976543217	C23	5:30 am	Female

Table 2.1 Train Database after adding aadhar card number

Now verification of passengers traveling in trains will be done at the station only. Biometric test will verify all passengers by using aadhar database and matching their finger print with aadhaar number. The biometric Machine will be available at security check only so that every passenger for verification of ticket will go through security check. This will increase the security in railway stations.

REF.NO2->Smart Computing Applications in Railway Systems - A case study in Indian Railways Passenger Reservation System

(By Parag Chatterjee, Asoke Nath)-2014

The demand for safe, fast, and reliable rail services continues to be the reason for concern in all the countries across the globe. Lack of operational efficiency and reliability, safety and security issues, and aging railway systems and practices are haunting various countries to bring about a change in their existing rail infrastructure. The global rail industry struggles to meet the increasing demand for freight and passenger transportation due to lack of optimized use of the rail network and inefficient use of rail assets. This is expected to induce rail executives to build rail systems that are smarter and more efficient. The passenger reservation system of Indian Railways is one of the world's largest reservation models. Daily about one million passengers travel in reserved accommodation with Indian Railways. Another sixteen million travel with unreserved tickets in Indian Railways. In this vast system, it is a herculean task to efficiently handle the passenger data, which is a key point of consideration now-a-days. In this paper, the authors have explored different issues of implementing smart computing in railway systems pertaining to reservation models

Since the existing PRS model is a wide-spread existing structure, it is practically impossible to usher in a new model, completely flushing away the old one. But this smart PRS model can be implemented just by introducing some modifications into the existing system. Just as in case, presently the data entry in the system during the time of reservation is done manually. When the new model introduces the UID-based entry and verification, an overlying interface is to be designed, which will handle this portion of data entry and will feed the same data to the system, but the data will be supplied based on the UID.

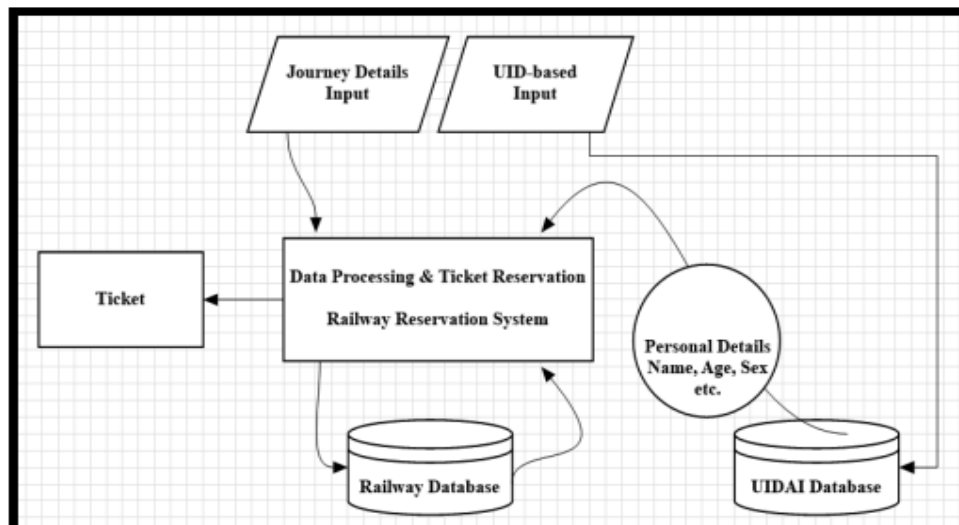


Figure 2.1.2. Working of the UID-based Reservation System in Indian Railways

Hence the modification is to be implemented in a superficial layer, lying over the existing system of manually entering the personal details of the passenger. And for the time being, due to the introduction of the new system (Figure 2.1.2), the two systems can run parallel.

Discussing the advantages brings firstly the no breach in verification of the passengers traveling in the reserved coaches of the train. The new system involves a comprehensive check of identity in case of every passenger traveling onboard. Also the huge overhead of typing the details of all the passengers is bypassed by a more intelligent and efficient method of data entry using UID. This identification is also comprehensively verified in a second phase at the running train, when the Travelling Ticket Examiner (TTE) verifies the passengers physically traveling. The passengers produce their Aadhaar cards to the TTE, who verifies the UID on his reservation chart with that of the card. Alternatively, the passengers can also show the auto-generated SMS from their mobile phone, bearing all journey details, as sent by the railway reservation portal. Also another point of checking can be the photograph of the passenger because this time new reservation charts can be prepared with photographs of the passengers. The corruption regarding impersonation of passengers can thus easily be avoided. Also the fraudulent booking of tours/travel agents can also be avoided. Presently these persons purchase a bulk amount of tickets taking arbitrary names and ages. Later these tickets are sold to groups of persons matching the age groups and one person per ticket travels correctly, the other persons travel in false identity

Ref.No->3: Automatic Processing of Structured Handwritten Documents:
An Application for Indian Railway Reservation System
(by Sandip Rakshit,Soumya Sona Das,Kalyan S Sengupta,Subhadip Basu)

An effective document processing system must be able to recognize structured and semi structured forms that are written by different persons' handwriting. In this work we have developed a method and system that can process structured form document layout and recognize its contents. Our approach has been applied here in the context of the Indian railway reservation/cancellation requisition system with encouraging results. In reality, handwritten data usually touch or cross the preprinted form frames and texts, creating complex problems for the recognition routines. In this paper, we address these issues and attempt to solve the problem for the Indian Railway Reservation system using our custom built form processing software and Tesseract open source character recognition engine.

In our current work we have first designed the structured form layout and then fill up the forms manually from random users. We then scan the filled-up forms using a flatbed scanner and analyze its contents using the custom-build form-recognition system. This section presents the main steps of our system. The key issues are discussed below:

Design Issues: We have used structured blocks/boxes/grids for input of every character in the newly designed IRRS form. There are two defined entities in this form, viz., characters and data fields. Multiple characters of one data part are grouped together to constitute one data field (like, six digits of any date field or a signature field). In this way, we have 2510 boxed regions that constitute 35 data fields. Out of these data fields, 27 box fields are in machine readable forms, where each such field contains isolated boxformatted characters. The remaining 8 non box fields contain 2 signature fields and 6 concession fields. These parts are the forms are written in a continuous way and entered into the system manually (or as an image).

Another important design issue in form document processing is image registration. In this work, we have put square block markers in four corners of the form for registration (skew correction and alignment) of the input images. Input images are scanned by a flatbed document scanner at a resolution of 300 dpi. Once the registration blocks/markers on the input/scanned images are identified by the designed system/program, different morphological filtering is done to extract the embedded data.

CHAPTER 3

SYSTEM ARCHITECTURE AND DESIGN

3.1 ARCHITECTURE DIAGRAM:

A typical three-layer structure is used in the system: the database layer, the application service layer, the user interface layer. System architecture as shown in *Figure 1*.

The database layer The database is used to hold data, including user registration information, ticket ordering information, ticket information and all of the other information. **The application service layer** The application service layer is the core of this three-layer structure, the system functions and business logic are handled in this layer. In this layer, the system's business logic is encapsulated, the application service interface is provided for the user interface layer and the system modules between the function calls. The application service layer also updates data in the database, according to the service request of the top layer. **The user interface layer** The user interface layer is a program that runs on a remote user computer. It displays the provided services by the server to the user. When the user selects a service, this program sends a request to the server. When the server returns the processed result, this program shows it to the user.

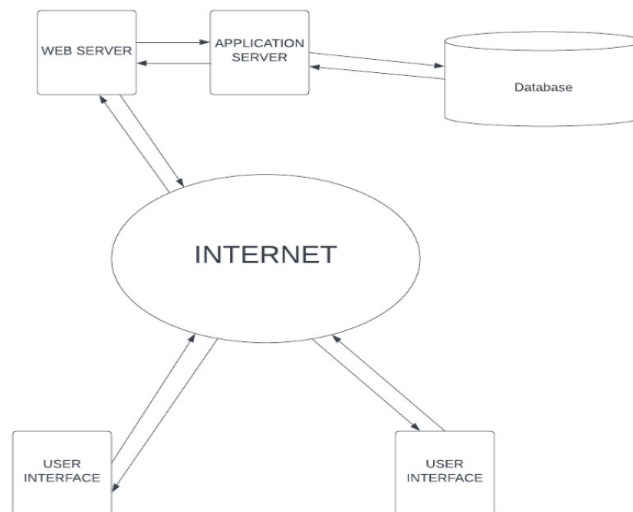


Fig 3.1 System Architecture diagram

It provides the application logic layer in three-tier architecture, enabling client components to interact with data resources and legacy applications. The first tier represents the company and customer with production unit access. The second tier is the application server which receives the queries. The third tier is the database server from which information is fetched and served for requests. Collectively, three tier architectures are programming models that enable the distribution of application functionality across three independent systems, typically:

- i. Client components running on local workstations (tier one)
- ii. Processes running on remote servers (tier two)
- iii. A discrete database, resource managers, and mainframe applications (tier three) rational

A Relational Database Management System was used to design the database. This design system is an excellent tool for organizing large amounts of data and defining the relationship between the datasets in a consistent and understandable way. A RDBMS provides a structure which is flexible enough to accommodate almost any kind of data. Relationships between the tables were defined by creating special columns (keys), which contained the same set of values in each table. The tables can then be joined in different combinations to extract the needed data. A RDBMS also offers flexibility that enables it to redesign and regenerate reports from the database without need to re-enter the data. Data dictionaries were used to provide definitions of the data to be used; these included the final data structures for the various tables and their corresponding data fields, description and sizes. The user application programs and interface is developed using PHP, CSS, HTML, and Java Script with support of structured query language (SQL) and MYSQL.

3.2 Entity Relationship Diagram

An entity–relationship model describes inter-related things of interest in a specific domain of knowledge. An ER model is composed of entity types and specifies relationships that can exist between instances of those entity types.

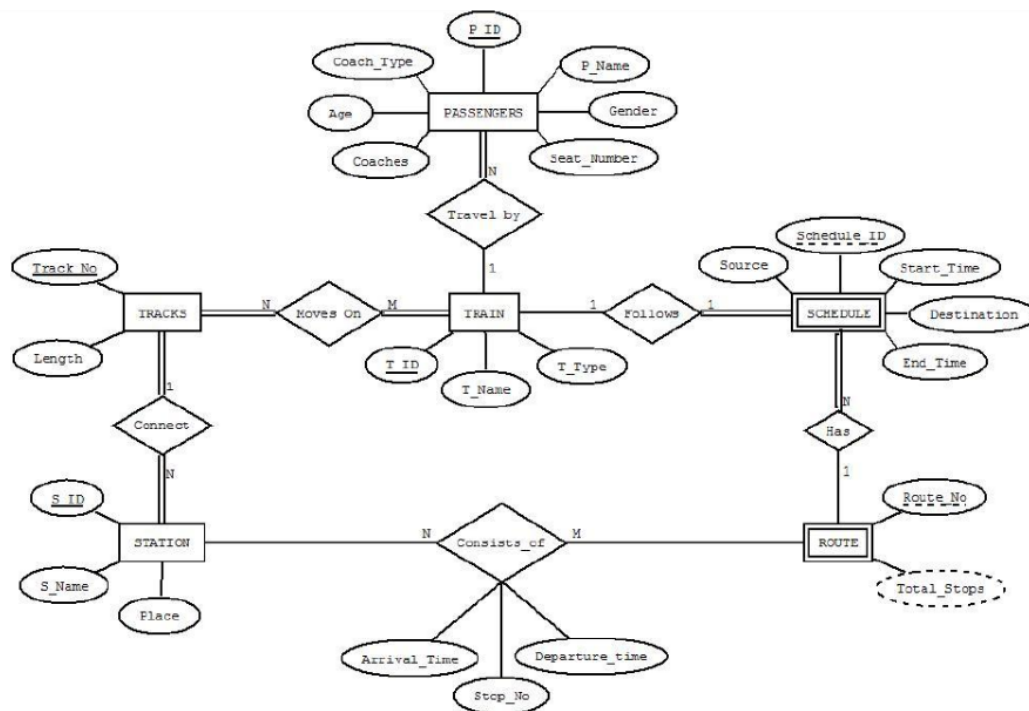


Fig. 3.2 ER diagram for Railway Management system

This ER Diagram gives a brief idea about the relations existing between the tables and tells about the primary and the foreign keys being used in this Database.

List of Entities and its attributes:

- **Passengers**
P_id- Passenger ID
P_name- Passenger name
Gender- Gender of passenger
Seat_No
Age
Coaches
Coach Type
- **Schedule**
Source
Schedule_id
Start_time
Destination
End_Time
- **Route**
Route_no
Total Stops
- **Stations**
Place
S_name- Station Name
S_id- Station id
- **Tracks**
Track_no
Length- Like narrow gauge, broad gauge etc
- **Train**
T_id- Unique train Id eg 12164
T_name- Name of train eg Chennai Express
T_type- Train types like Mail, Superfast etc;

Relationships:

- Travel By
- Follows
- Has
- Consists of
- Connect
- Moves on

Entity 1	Relationship	Entity 2
Passengers	Travel by	Train
Train	Follows	Schedule
Schedule	Has	Route
Route	Consists of	Stations

Station	connects	tracks
Train	Moves on	tracks

Table 3.2 Relationship between entities

NORMALIZATION

- First Normal Form (1NF)

As the domain of all attributes of all relations in the database has atomic value and no tuples can have a set of these values, all relations are in 1NF.

- Second Normal Form (2NF)

As there is no partial dependency in the database, i.e. all non prime attributes of a relation are fully functionally dependent on the primary key of the relation schema, all relations are in 2NF.

- Third Normal Form (3NF)

As all relations are in 2NF and no non-prime attribute of a relation schema is transitively dependent on the primary key, all relations are in 3NF.

CHAPTER 4

MODULES AND FUNCTIONALITIES

4.1 Front End Design:

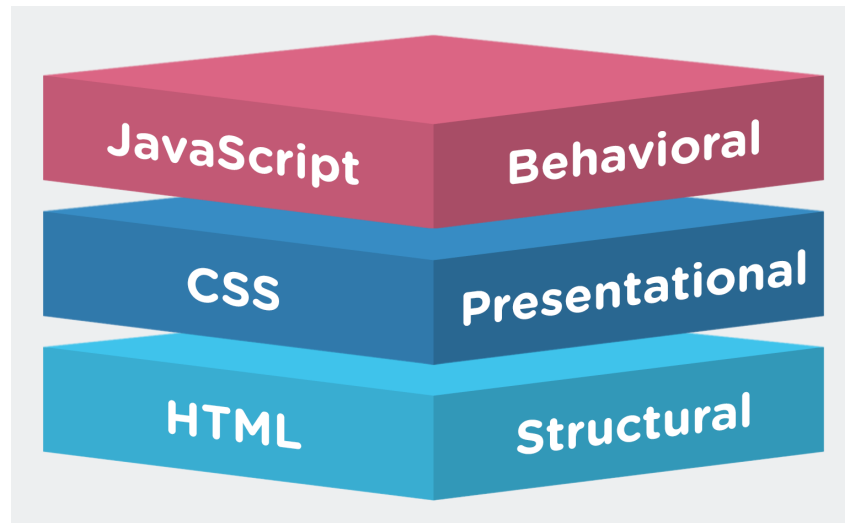


Fig.4.1 Front End Design

HTML:

HTML is an acronym that stands for HyperText Markup Language.

HyperText: HyperText simply means "Text within Text". A text has a link within it, is a hypertext. Every time you click on a word that brings you to a new webpage, you have clicked on a hypertext.

Markup language: A markup language is a programming language that is used to make text more interactive and dynamic. It can turn a text into images, tables, links, etc. An HTML document is made of many HTML tags and each HTML tag contains different content

CSS:

Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language such as HTML or XML (including XML dialects such as SVG, MathML or XHTML). CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript.

Javascript:

Javascript is a dynamic computer programming language. It is lightweight and most commonly used as a part of web pages, whose implementations allow client-side script to interact with the user and make dynamic pages.

4.2 Back-end Design:

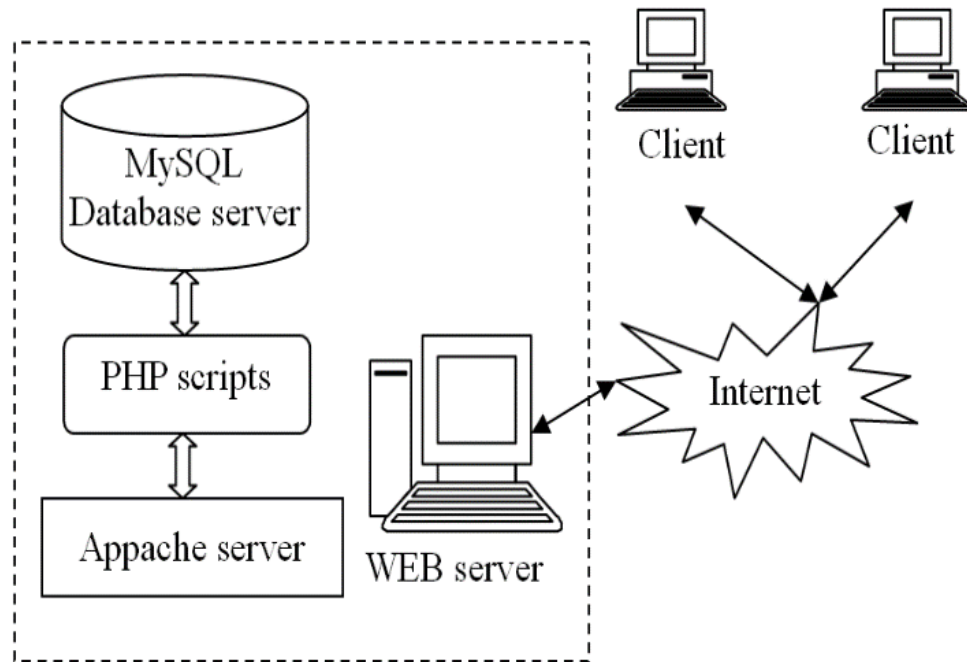


Fig.4.2 Back-end and Database

PHP:

PHP is a server-side scripting language designed primarily for web development but also used as a general programming language. PHP code may be embedded into HTML or HTML5 markup or it can be used in combination with various web template systems, web content management systems, and web frameworks. PHP code is usually processed by a PHP interpreter implemented as a module in the web server. The web server software combines the results of the interpreted and executed PHP code, which may be any type of data, including images, with the generated webpage.

Web Apache Server:

Apache is the most widely used web server software. Developed and maintained by Apache Software Foundation, Apache is open-source software available for free. It runs on 67% of all web servers in the world. It is fast, reliable, and secure. It can be highly customized to meet the needs of many different environments by using extensions and modules. Most WordPress hosting providers use Apache as their web server software. However, WordPress can run on other web server software as well.

4.3 List of modules in railway reservation system:

1. New user
2. Enquiry and Ticket Booking
3. Ticket History or Cancellation
4. Admin Login

1. New User:

The user module allows users to register, log in, and log out. Users benefit from being able to sign on because this associates content they create with their account and allows various permissions to be set for their roles. This feature is for the candidates who are visiting the site for the very first time. They need to register themselves for utilizing further useful features. And for maintaining the contact between both ends.

2. Enquiry and Ticket booking:

It shows the total enquiries, applications, and admissions for the academic year. This data is fetched from multiple modules of Fedena which shows the comprehensive top-level data for the institution. Also, you can see that this is equal to a sales funnel. This feature is for all the users. Enquiry section consists of the FAQs and also a section where the enquiry of a particular doubt can be done.

Ticket booking consists of a list of starting stations and destination stations with their approximated timings.

3. Ticket History or cancellation:

This module shows all the past transactions and past booking done from the particular account. Cancellation allows you to cancel the trip bookings and keep a track of it.

4. Admin Login:

A module for administration purposes. The Administration module allows you to control the operation of Business Process Server, install and uninstall applications, and manage all users and groups. Administrators can change security settings, install software and hardware, access all files on the computer, and make changes to other user accounts.

4.4 DATABASE CONNECTIVITY:

Relational Database:

A relational database is a collection of data items with pre-defined relationships between them. These items are organized as a set of tables with columns and rows. Tables are used to hold information about the objects to be represented in the database. Each column in a table holds a certain kind of data and a field stores the actual value of an attribute. The rows in the table represent a collection of related values of one object or entity. Each row in a table could be marked with a unique identifier called a primary key, and rows among multiple tables can be made related using foreign keys. This data can be accessed in many different ways without reorganizing the database tables themselves.

4.5 MySQL Database:

The MySQL Database Software is a client/server system that consists of a multithreaded SQL server that supports different back ends, several different client programs and libraries, administrative tools, and a wide range of application programming interfaces (APIs). A relational database stores data in separate tables rather than putting all the data in one big storeroom. The database structures are organized into physical files optimized for speed. The logical model, with objects such as databases, tables, views, rows, and columns, offers a flexible programming environment. You set up rules governing the relationships between different data fields, such as one-to-one, one-to-many, unique, required or optional, and “pointers” between different tables. The database enforces these rules, so that with a well-designed database, your application never sees inconsistent, duplicate, orphan, out-of-date, or missing data.

The SQL part of “MySQL” stands for “Structured Query Language”. SQL is the most common standardized language used to access databases. Depending on your programming environment, you might enter SQL directly (for example, to generate reports), embed SQL statements into code written in another language, or use a language-specific API that hides the SQL syntax.

A database is a structured collection of data. It may be anything from a simple shopping list to a picture gallery or the vast amounts of information in a corporate network. To add, access, and process data stored in a computer database, you need a database management system such as MySQL Server. Since computers are very good at handling large amounts of data, database management systems play a central role in computing, as standalone utilities, or as parts of other applications.

CHAPTER 5

CODING AND TESTING

SQL QUERIES:

```
SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
```

```
SET time_zone = "+00:00";
```

```
--
```

```
-- Table structure for table `canc`
```

```
--
```

```
CREATE TABLE IF NOT EXISTS `canc` (  
  `pnr` int(11) NOT NULL,  
  `rfare` int(11) DEFAULT '0',  
  PRIMARY KEY (`pnr`)  
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```
--
```

```
-- Dumping data for table `canc`
```

```
--
```

```
INSERT INTO `canc` (`pnr`, `rfare`) VALUES  
(57, 1100),  
(58, 5600);
```

```
-- -----
```

```
--
```

```
-- Table structure for table `class`
```

```
--
```

```
CREATE TABLE IF NOT EXISTS `class` (  
  `cname` varchar(10) NOT NULL,  
  PRIMARY KEY (`cname`)  
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```
--
```

```
-- Dumping data for table `class`
```

```
--
```

```
INSERT INTO `class` (`cname`) VALUES  
('AC1'),  
('AC2'),  
('AC3'),  
('CC');
```

```
('EC'),  
('SL');
```

```
-- -----
```

```
--  
-- Table structure for table `classeats`  
--
```

```
CREATE TABLE IF NOT EXISTS `classeats` (  
  `trainno` int(11) NOT NULL,  
  `sp` varchar(50) NOT NULL COMMENT 'Starting_Point',  
  `dp` varchar(50) NOT NULL COMMENT 'Destination_Point',  
  `doj` date NOT NULL,  
  `class` varchar(10) NOT NULL,  
  `fare` int(11) NOT NULL,  
  `seatsleft` int(11) NOT NULL,  
  PRIMARY KEY (`trainno`,`sp`,`dp`,`doj`,`class`),  
  KEY `class` (`class`),  
  KEY `sp` (`sp`,`dp`),  
  KEY `dp` (`dp`)  
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```
--  
-- Dumping data for table `classeats`  
--
```

```
INSERT INTO `classeats` (`trainno`, `sp`, `dp`, `doj`, `class`, `fare`, `seatsleft`) VALUES  
(12, 'Chandigarh', 'Jaipur', '2023-05-01', 'AC1', 2200, 107),  
(12, 'Chandigarh', 'Jaipur', '2023-05-02', 'AC1', 3200, 20),  
(12, 'Chandigarh', 'Jaipur', '2023-05-03', 'AC3', 2400, 60),  
(12, 'Chandigarh', 'Jaipur', '2023-05-04', 'EC', 1200, 100),  
(12, 'Chandigarh', 'Jaipur', '2023-05-05', 'SL', 500, 200),  
(12, 'Jaipur', 'Kolkata', '2023-05-06', 'AC1', 1434, 243),  
(12, 'Jaipur', 'Kolkata', '2023-05-07', 'AC1', 2900, 15),  
(12, 'Jaipur', 'Kolkata', '2023-05-08', 'AC3', 2100, 40),  
(12, 'Jaipur', 'Kolkata', '2023-05-09', 'EC', 1500, 120),  
(12, 'Jaipur', 'Kolkata', '2023-05-10', 'SL', 800, 250),  
(12, 'Kolkata', 'Lucknow', '2023-05-11', 'AC1', 934, 322),  
(12, 'Kolkata', 'Lucknow', '2023-05-12', 'AC1', 3100, 30),  
(12, 'Kolkata', 'Lucknow', '2023-05-13', 'AC3', 1900, 30),  
(12, 'Kolkata', 'Lucknow', '2023-05-14', 'EC', 1700, 150),  
(12, 'Kolkata', 'Lucknow', '2023-05-15', 'SL', 700, 220),  
(12, 'Lucknow', 'Delhi', '2023-05-16', 'AC1', 344, 326),  
(12, 'Lucknow', 'Delhi', '2023-05-17', 'AC1', 2750, 20),  
(12, 'Lucknow', 'Delhi', '2023-05-17', 'AC3', 2350, 60),
```

```

(12, 'Lucknow', 'Delhi', '2023-05-17', 'EC', 1100, 118),
(12, 'Lucknow', 'Delhi', '2023-05-17', 'SL', 900, 180),
(18, 'Chandigarh', 'Jaipur', '2023-05-12', 'AC1', 2420, 50),
(18, 'Chandigarh', 'Jaipur', '2023-05-12', 'AC3', 1700, 20),
(18, 'Chandigarh', 'Jaipur', '2023-05-12', 'CC', 750, 120),
(18, 'Jaipur', 'Delhi', '2023-05-12', 'AC1', 2750, 20),
(18, 'Jaipur', 'Delhi', '2023-05-12', 'AC3', 1200, 20),
(18, 'Jaipur', 'Delhi', '2023-05-12', 'CC', 900, 150),
(20, 'Delhi', 'Jaipur', '2023-05-09', 'AC1', 4500, 20),
(20, 'Delhi', 'Jaipur', '2023-05-09', 'AC2', 3200, 50),
(20, 'Delhi', 'Jaipur', '2023-05-09', 'AC3', 2700, 50),
(20, 'Delhi', 'Jaipur', '2023-05-09', 'SL', 900, 300);

```

```
--
```

```
-- Triggers `classseats`
```

```
--
```

```

DROP TRIGGER IF EXISTS `before_insert_on_classseats`;
DELIMITER //
CREATE TRIGGER `before_insert_on_classseats` BEFORE INSERT ON `classseats`
FOR EACH ROW begin
if datediff(curdate(),new.doj)>0 then
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'Check date!!!';
end if;
if new.fare<=0 then
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'Check fare!!!';
end if;
if new.seatsleft<=0 then
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'Check seats!!!';
end if;
end
//
DELIMITER ;
DROP TRIGGER IF EXISTS `before_update_on_classseats`;
DELIMITER //
CREATE TRIGGER `before_update_on_classseats` BEFORE UPDATE ON `classseats`
FOR EACH ROW begin
if datediff(curdate(),new.doj)>0 then
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'check date!!!';
end if;
if new.fare<=0 then
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'Check fare!!!';

```

```

end if;
if new.seatsleft<=0 then
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'Check seats!!!';
end if;
end
//
DELIMITER ;

-----

--
-- Table structure for table `pd` passenger details
--

CREATE TABLE IF NOT EXISTS `pd` (
  `pnr` int(11) NOT NULL,
  `pname` varchar(50) NOT NULL,
  `page` int(11) NOT NULL,
  `pgender` varchar(10) NOT NULL,
  PRIMARY KEY (`pnr`,`pname`,`page`,`pgender`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
-- Dumping data for table `pd`
--

INSERT INTO `pd` (`pnr`, `pname`, `page`, `pgender`) VALUES
(58, 'akhil', 20, 'M'),
(58, 'deepak', 21, 'M'),
(58, 'rahul', 12, 'M'),
(58, 'shyam', 50, 'M'),
(59, 'abhinav', 20, 'M'),
(59, 'vikas', 40, 'M'),
(60, 'mohan', 20, 'M');

--
-- Triggers `pd`
--
DROP TRIGGER IF EXISTS `before_insert_on_pd`;
DELIMITER //
CREATE TRIGGER `before_insert_on_pd` BEFORE INSERT ON `pd`
FOR EACH ROW begin
if new.pgender NOT IN ('M','F') then
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'Enter M:Male F:Female.';

```

```

end if;
end
//
DELIMITER ;
DROP TRIGGER IF EXISTS `before_update_on_pd`;
DELIMITER //
CREATE TRIGGER `before_update_on_pd` BEFORE UPDATE ON `pd`
FOR EACH ROW begin
if new.pgender NOT IN ('M','F') then
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'Enter M:Male F:Female.';
end if;
end
//
DELIMITER ;

-- -----

--
-- Table structure for table `resv`
--

CREATE TABLE IF NOT EXISTS `resv` (
  `pnr` int(11) NOT NULL AUTO_INCREMENT,
  `id` int(11) NOT NULL,
  `trainno` int(11) NOT NULL,
  `sp` varchar(50) NOT NULL,
  `dp` varchar(50) NOT NULL,
  `doj` date NOT NULL,
  `tfare` int(11) NOT NULL,
  `class` varchar(50) NOT NULL,
  `nos` int(11) NOT NULL,
  `status` varchar(50) NOT NULL,
  PRIMARY KEY (`pnr`),
  UNIQUE KEY `UNIQUE` (`id`,`trainno`,`doj`,`status`),
  UNIQUE KEY `pnr` (`pnr`,`id`,`trainno`,`doj`,`class`,`status`),
  UNIQUE KEY `pnr_2` (`pnr`,`id`,`trainno`,`sp`,`dp`,`doj`,`tfare`,`class`,`nos`,`status`),
  KEY `FK_ID` (`id`),
  KEY `FK_TN_DOJ_C` (`trainno`,`doj`,`class`),
  KEY `class` (`class`),
  KEY `sp` (`sp`,`dp`),
  KEY `dp` (`dp`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=61 ;

--
-- Dumping data for table `resv`

```

--

```
INSERT INTO `resv` (`pnr`, `id`, `trainno`, `sp`, `dp`, `doj`, `tfare`, `class`, `nos`, `status`)
VALUES
(51, 4, 12, 'Chandigarh', 'Jaipur', '2015-05-07', 3300, 'AC1', 2, 'BOOKED'),
(57, 5, 12, 'Chandigarh', 'Jaipur', '2015-05-07', 2200, 'AC1', 1, 'CANCELLED'),
(58, 6, 20, 'Delhi', 'Jaipur', '2015-05-09', 11200, 'AC2', 4, 'CANCELLED'),
(59, 10, 12, 'Lucknow', 'Delhi', '2015-05-17', 2200, 'EC', 2, 'BOOKED');
```

--

-- Triggers `resv`

--

```
DROP TRIGGER IF EXISTS `after_insert_on_resv`;
DELIMITER //
CREATE TRIGGER `after_insert_on_resv` AFTER INSERT ON `resv`
FOR EACH ROW begin
UPDATE classseats SET seatsleft=seatsleft-new.nos where trainno=new.trainno AND
class=new.class AND doj=new.doj AND sp=new.sp AND dp=new.dp;
end
//
DELIMITER ;
DROP TRIGGER IF EXISTS `after_update_on_resv`;
DELIMITER //
CREATE TRIGGER `after_update_on_resv` AFTER UPDATE ON `resv`
FOR EACH ROW begin
if (new.status='CANCELLED' AND datediff(new.doj,curdate())<0 ) then
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'Cancellation Not Possible!!!!';
end if;

if (new.status='CANCELLED' AND datediff(new.doj,curdate())>0 )then
UPDATE classseats SET seatsleft=seatsleft+new.nos where trainno=new.trainno AND
class=new.class AND doj=new.doj AND sp=new.sp AND dp=new.dp;
if datediff(new.doj,curdate())>=30 then
INSERT INTO canc values (new.pnr,new.tfare);
end if;
if datediff(new.doj,curdate())<30 then
INSERT INTO canc values (new.pnr,0.5*new.tfare);
end if;
end if;
end
//
DELIMITER ;
DROP TRIGGER IF EXISTS `before_insert_on_resv`;
DELIMITER //
CREATE TRIGGER `before_insert_on_resv` BEFORE INSERT ON `resv`
```

```

FOR EACH ROW begin
if new.tfare<0 then
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'Negative balance NOT possible';
end if;
if new.nos<=0 then
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'Negative OR 0 seats NOT possible';
end if;
if (select seatsleft from classseats where trainno=new.trainno AND class=new.class AND
doj=new.doj AND sp=new.sp AND dp=new.dp) < new.nos then
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'Not enough seats available!!!';
end if;
if datediff(new.doj,curdate())<0 then
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'Booking Not Possible!!!!';
end if;
SET new.status='BOOKED';
end
//
DELIMITER ;
DROP TRIGGER IF EXISTS `before_update_on_resv`;
DELIMITER //
CREATE TRIGGER `before_update_on_resv` BEFORE UPDATE ON `resv`
FOR EACH ROW begin
if new.tfare<0 then
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'Negative balance NOT possible';
end if;
if new.nos<=0 then
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'Negative OR 0 seats NOT possible';
end if;
if (select seatsleft from classseats where trainno=new.trainno AND class=new.class AND
doj=new.doj AND sp=new.sp AND dp=new.dp) < new.nos then
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'Not enough seats available!!!';
end if;
end
//
DELIMITER ;

```

-- -----

--

-- Table structure for table `schedule`

--

```
CREATE TABLE IF NOT EXISTS `schedule` (  
  `id` int(11) NOT NULL AUTO_INCREMENT,  
  `trainno` int(11) NOT NULL,  
  `sname` varchar(50) NOT NULL,  
  `arrival_time` time NOT NULL,  
  `departure_time` time NOT NULL DEFAULT '00:00:00',  
  `distance` int(11) NOT NULL,  
  PRIMARY KEY (`id`),  
  KEY `trainno` (`trainno`),  
  KEY `sname` (`sname`),  
  KEY `id` (`id`),  
  KEY `distance` (`distance`),  
  KEY `id_2` (`id`)  
) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=42 ;
```

--

-- Dumping data for table `schedule`

--

```
INSERT INTO `schedule` (`id`, `trainno`, `sname`, `arrival_time`, `departure_time`, `distance`)  
VALUES  
(1, 12, 'Chandigarh', '01:00:12', '01:00:00', 0),  
(2, 12, 'Jaipur', '03:45:15', '03:50:00', 100),  
(3, 12, 'Kolkata', '05:00:00', '05:15:00', 300),  
(4, 12, 'Lucknow', '11:50:10', '12:00:00', 450),  
(5, 12, 'Delhi', '16:30:00', '16:30:00', 600),  
(6, 13, 'Jammu Kashmir', '22:00:00', '22:00:00', 0),  
(7, 13, 'Delhi', '04:00:00', '04:05:00', 700),  
(8, 13, 'Rajasthan', '07:30:50', '07:33:00', 900),  
(9, 13, 'Allahbad', '09:00:00', '09:10:00', 1700),  
(10, 13, 'Patna', '11:45:00', '11:47:00', 2500),  
(11, 13, 'Madhya Pradesh', '13:00:00', '13:00:00', 3600),  
(12, 14, 'Jammu Kashmir', '01:00:12', '01:00:12', 0),  
(13, 14, 'Madras', '22:00:00', '22:00:00', 2500),  
(14, 15, 'Delhi', '16:00:00', '16:00:00', 0),  
(15, 15, 'Jaipur', '22:45:00', '22:45:00', 800),  
(16, 16, 'Jaipur', '03:30:00', '03:30:00', 0),  
(17, 16, 'Delhi', '09:30:00', '09:30:00', 800),  
(18, 17, 'Delhi', '00:00:14', '00:00:14', 0),  
(19, 17, 'Jaipur', '16:00:00', '16:10:00', 500),  
(20, 17, 'Chandigarh', '20:30:00', '20:30:00', 1200),  
(21, 18, 'Chandigarh', '08:05:00', '08:05:00', 0),  
(22, 18, 'Jaipur', '10:15:00', '10:20:00', 700),
```



```
(23, 18, 'Delhi', '14:00:00', '14:00:00', 1200),
(24, 6, 'Jaipur', '03:30:00', '03:30:00', 0),
(25, 6, 'Allahbad', '08:00:00', '08:15:00', 200),
(26, 6, 'Lucknow', '15:15:00', '15:15:00', 700),
(27, 19, 'Lucknow', '13:30:00', '13:30:00', 0),
(28, 19, 'Allahbad', '20:00:00', '20:10:00', 300),
(29, 19, 'Jaipur', '05:15:00', '05:15:00', 700),
(30, 20, 'Delhi', '10:04:00', '10:04:00', 0),
(31, 20, 'Jaipur', '16:00:00', '16:00:00', 800),
(32, 21, 'Jaipur', '20:00:00', '20:00:00', 0),
(33, 21, 'Delhi', '10:00:00', '10:00:00', 800),
(34, 22, 'Delhi', '16:35:00', '16:35:00', 0),
(35, 22, 'Rajasthan', '20:00:00', '20:10:00', 1100),
(36, 22, 'Madhya Pradesh', '03:30:00', '03:33:00', 1500),
(37, 22, 'Mumbai', '09:00:00', '09:00:00', 2300),
(38, 23, 'Mumbai', '01:00:00', '01:00:00', 0),
(39, 23, 'Madhya Pradesh', '05:30:00', '05:40:00', 1500),
(40, 23, 'Rajasthan', '15:45:00', '15:50:00', 2000),
(41, 23, 'Delhi', '20:30:00', '20:30:00', 2300);
```

-- -----

```
--
-- Table structure for table `station`
--
```

```
CREATE TABLE IF NOT EXISTS `station` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `sname` varchar(50) NOT NULL,
  PRIMARY KEY (`sname`),
  KEY `id` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=14 ;
```

```
--
-- Dumping data for table `station`
--
```

```
INSERT INTO `station` (`id`, `sname`) VALUES
(1, 'Chandigarh'),
(2, 'Delhi'),
(3, 'Jaipur'),
(4, 'Lucknow'),
(5, 'Mumbai'),
(6, 'Allahbad'),
(7, 'Kolkata'),
(8, 'Patna'),
```

```
(9, 'Madras'),  
(10, 'Jammu Kashmir'),  
(11, 'Rajasthan'),  
(12, 'Madhya Pradesh');
```

```
-- -----
```

```
--  
-- Table structure for table `train`  
--
```

```
CREATE TABLE IF NOT EXISTS `train` (  
  `trainno` int(11) NOT NULL AUTO_INCREMENT COMMENT 'Train_no',  
  `tname` varchar(50) NOT NULL COMMENT 'Train_name',  
  `sp` varchar(50) NOT NULL COMMENT 'Starting_Point',  
  `st` time NOT NULL COMMENT 'Arrival_Time',  
  `dp` varchar(50) NOT NULL COMMENT 'Destination_Point',  
  `dt` time NOT NULL,  
  `dd` varchar(10) DEFAULT NULL COMMENT 'Day',  
  `distance` int(11) NOT NULL COMMENT 'Distance',  
  PRIMARY KEY (`trainno`),  
  KEY `sp` (`sp`),  
  KEY `dp` (`dp`)  
) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=24 ;
```

```
--  
-- Dumping data for table `train`  
--
```

```
INSERT INTO `train` (`trainno`, `tname`, `sp`, `st`, `dp`, `dt`, `dd`, `distance`) VALUES  
(6, 'Ashram Express', 'Jaipur', '10:00:00', 'Lucknow', '21:30:00', 'Day 1', 700),  
(12, 'Shatabdi Express', 'Chandigarh', '01:00:12', 'Delhi', '16:30:00', 'Day 1', 600),  
(13, 'Harijan Express', 'Jammu Kashmir', '22:00:00', 'Madhya Pradesh', '13:00:00', 'Day2', 3600),  
(14, 'Jammu Mail Express', 'Jammu Kashmir', '01:00:12', 'Madras', '22:00:00', 'Day 1', 2500),  
(15, 'Delhi Jaipur Double Decker', 'Delhi', '16:00:00', 'Jaipur', '22:45:00', 'Day 1', 800),  
(16, 'Jaipur Delhi Double Decker', 'Jaipur', '03:30:00', 'Delhi', '09:30:00', 'Day 1', 800),  
(17, 'Delhi Chandigarh Shatabdi', 'Delhi', '00:00:14', 'Chandigarh', '20:30:00', 'Day 1', 1200),  
(18, 'Chandigarh Delhi Shatabdi', 'Chandigarh', '08:05:00', 'Delhi', '14:00:00', 'Day 2', 1200),  
(19, 'Ashram Express', 'Lucknow', '13:30:00', 'Jaipur', '05:15:00', 'Day 2', 700),  
(20, 'Frontier Express', 'Delhi', '10:04:00', 'Jaipur', '16:00:00', 'Day 1', 800),  
(21, 'Frontier Express', 'Jaipur', '20:00:00', 'Delhi', '10:00:00', 'Day 2', 800),  
(22, 'Rajdhani Express', 'Delhi', '16:35:00', 'Mumbai', '09:00:00', 'Day 2', 2300),  
(23, 'Rajdhani Express', 'Mumbai', '01:00:00', 'Delhi', '20:30:00', 'Day 1', 2300);
```

```
--  
-- Triggers `train`
```

```

--
DROP TRIGGER IF EXISTS `before_insert_on_train`;
DELIMITER //
CREATE TRIGGER `before_insert_on_train` BEFORE INSERT ON `train`
  FOR EACH ROW begin
if (new.dt<new.st AND new.dd='Day 1') then
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'Improper Timings';
end if;
if (new.dp=new.sp) then
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'Same Starting & Destination Points not allowed';
end if;
end
//
DELIMITER ;
DROP TRIGGER IF EXISTS `before_update_on_train`;
DELIMITER //
CREATE TRIGGER `before_update_on_train` BEFORE UPDATE ON `train`
  FOR EACH ROW begin
if (new.dt<new.st AND new.dd='Day 1') then
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'Improper Timings';
end if;
end
//
DELIMITER ;

-- -----

--
-- Table structure for table `user`
--

CREATE TABLE IF NOT EXISTS `user` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `emailid` varchar(50) NOT NULL,
  `password` varchar(50) NOT NULL,
  `mobilen0` varchar(10) NOT NULL,
  `dob` date NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `UNIQUEMN` (`mobilen0`),
  on` (`sname`),
  ADD CONSTRAINT `resv_ibfk_3` FOREIGN KEY (`dp`) REFERENCES `station` (`sname`);

/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;

```

```
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_C UNIQUE KEY
`UNIQUEEI` (`emailid`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=21 ;
```

```
--
-- Dumping data for table `user`
--
```

```
INSERT INTO `user` (`id`, `emailid`, `password`, `mobilen`, `dob`) VALUES
(4, 'garvitjain@gmail.com', 'garvit', '9312201852', '1994-01-01'),
(5, 'deepakgoel@hotmail.com', 'deepak', '9312201853', '1994-02-22'),
(6, 'akhilkumar@yahoo.co.in', 'akhil', '9872231234', '1994-03-04'),
(7, 'ayushjain@outlook.com', 'ayush', '9810150525', '1995-01-03'),
(8, 'aakashbharadwaj@yahoo.com', 'aakash', '9013452635', '1993-12-30'),
(10, 'abhinavsingh@gmail.com', 'abhinav', '9876675567', '1991-01-01'),
(12, 'amanmalik@hotmail.com', 'aman', '9878876654', '1997-09-08'),
(19, 'dhruvgosian@gmail.com', 'dhruv', '9807890453', '1965-04-01'),
(20, 'chiragbansal@nsitononline.com', 'chirag', '9123456789', '1960-06-02');
```

```
--
-- Triggers `user`
--
DROP TRIGGER IF EXISTS `before_insert_on_user`;
DELIMITER //
CREATE TRIGGER `before_insert_on_user` BEFORE INSERT ON `user`
FOR EACH ROW begin
if (year(curdate())-year(new.dob))<18 then
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'Minimum age bar of 18 years.';
end if;
end
//
DELIMITER ;
DROP TRIGGER IF EXISTS `before_update_on_user`;
DELIMITER //
CREATE TRIGGER `before_update_on_user` BEFORE UPDATE ON `user`
FOR EACH ROW begin
if (year(curdate())-year(new.dob))<18 then
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'Minimum age bar of 18 years.';
end if;
end
//
DELIMITER ;
```

```

--
-- Constraints for dumped tables
--

--
-- Constraints for table `classseats`
--
ALTER TABLE `classseats`
  ADD CONSTRAINT `classseats_ibfk_1` FOREIGN KEY (`trainno`) REFERENCES `train`
(`trainno`),
  ADD CONSTRAINT `classseats_ibfk_3` FOREIGN KEY (`sp`) REFERENCES `station`
(`sname`),
  ADD CONSTRAINT `classseats_ibfk_4` FOREIGN KEY (`dp`) REFERENCES `station`
(`sname`),
  ADD CONSTRAINT `classseats_ibfk_5` FOREIGN KEY (`class`) REFERENCES `class`
(`cname`);

--
-- Constraints for table `resv`
--
ALTER TABLE `resv`
  ADD CONSTRAINT `resv_ibfk_1` FOREIGN KEY (`trainno`) REFERENCES `train` (`trainno`),
  ADD CONSTRAINT `resv_ibfk_2` FOREIGN KEY (`sp`) REFERENCES `station` ONNECTION */;

-- phpMyAdmin SQL Dump
-- version 4.0.10deb1
-- http://www.phpmyadmin.net
--
-- Host: localhost
-- Generation Time: May 04, 2015 at 01:13 PM
-- Server version: 5.5.43-0ubuntu0.14.04.1
-- PHP Version: 5.5.9-1ubuntu4.9

SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
SET time_zone = "+00:00";

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8 */;

--
-- Database: `railway`
--

```

```

-----

--
-- Table structure for table `canc`
--

CREATE TABLE IF NOT EXISTS `canc` (
  `pnr` int(11) NOT NULL,
  `rfare` int(11) DEFAULT '0',
  PRIMARY KEY (`pnr`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
-- Dumping data for table `canc`
--

INSERT INTO `canc` (`pnr`, `rfare`) VALUES
(57, 1100),
(58, 5600);

-----

--
-- Table structure for table `class`
--

CREATE TABLE IF NOT EXISTS `class` (
  `cname` varchar(10) NOT NULL,
  PRIMARY KEY (`cname`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
-- Dumping data for table `class`
--

INSERT INTO `class` (`cname`) VALUES
('AC1'),
('AC2'),
('AC3'),
('CC'),
('EC'),
('SL');

-----

--

```

-- Table structure for table `classseats`

--

```
CREATE TABLE IF NOT EXISTS `classseats` (  
  `trainno` int(11) NOT NULL,  
  `sp` varchar(50) NOT NULL COMMENT 'Starting_Point',  
  `dp` varchar(50) NOT NULL COMMENT 'Destination_Point',  
  `doj` date NOT NULL,  
  `class` varchar(10) NOT NULL,  
  `fare` int(11) NOT NULL,  
  `seatsleft` int(11) NOT NULL,  
  PRIMARY KEY (`trainno`,`sp`,`dp`,`doj`,`class`),  
  KEY `class` (`class`),  
  KEY `sp` (`sp`,`dp`),  
  KEY `dp` (`dp`)  
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

--

-- Dumping data for table `classseats`

--

```
INSERT INTO `classseats` (`trainno`,`sp`,`dp`,`doj`,`class`,`fare`,`seatsleft`) VALUES  
(12, 'Chandigarh', 'Jaipur', '2023-05-01', 'AC1', 2200, 107),  
(12, 'Chandigarh', 'Jaipur', '2023-05-02', 'AC1', 3200, 20),  
(12, 'Chandigarh', 'Jaipur', '2023-05-03', 'AC3', 2400, 60),  
(12, 'Chandigarh', 'Jaipur', '2023-05-04', 'EC', 1200, 100),  
(12, 'Chandigarh', 'Jaipur', '2023-05-05', 'SL', 500, 200),  
(12, 'Jaipur', 'Kolkata', '2023-05-06', 'AC1', 1434, 243),  
(12, 'Jaipur', 'Kolkata', '2023-05-07', 'AC1', 2900, 15),  
(12, 'Jaipur', 'Kolkata', '2023-05-08', 'AC3', 2100, 40),  
(12, 'Jaipur', 'Kolkata', '2023-05-09', 'EC', 1500, 120),  
(12, 'Jaipur', 'Kolkata', '2023-05-10', 'SL', 800, 250),  
(12, 'Kolkata', 'Lucknow', '2023-05-11', 'AC1', 934, 322),  
(12, 'Kolkata', 'Lucknow', '2023-05-12', 'AC1', 3100, 30),  
(12, 'Kolkata', 'Lucknow', '2023-05-13', 'AC3', 1900, 30),  
(12, 'Kolkata', 'Lucknow', '2023-05-14', 'EC', 1700, 150),  
(12, 'Kolkata', 'Lucknow', '2023-05-15', 'SL', 700, 220),  
(12, 'Lucknow', 'Delhi', '2023-05-16', 'AC1', 344, 326),  
(12, 'Lucknow', 'Delhi', '2023-05-17', 'AC1', 2750, 20),  
(12, 'Lucknow', 'Delhi', '2023-05-17', 'AC3', 2350, 60),  
(12, 'Lucknow', 'Delhi', '2023-05-17', 'EC', 1100, 118),  
(12, 'Lucknow', 'Delhi', '2023-05-17', 'SL', 900, 180),  
(18, 'Chandigarh', 'Jaipur', '2023-05-12', 'AC1', 2420, 50),  
(18, 'Chandigarh', 'Jaipur', '2023-05-12', 'AC3', 1700, 20),  
(18, 'Chandigarh', 'Jaipur', '2023-05-12', 'CC', 750, 120),  
(18, 'Jaipur', 'Delhi', '2023-05-12', 'AC1', 2750, 20),
```

```
(18, 'Jaipur', 'Delhi', '2023-05-12', 'AC3', 1200, 20),
(18, 'Jaipur', 'Delhi', '2023-05-12', 'CC', 900, 150),
(20, 'Delhi', 'Jaipur', '2023-05-09', 'AC1', 4500, 20),
(20, 'Delhi', 'Jaipur', '2023-05-09', 'AC2', 3200, 50),
(20, 'Delhi', 'Jaipur', '2023-05-09', 'AC3', 2700, 50),
(20, 'Delhi', 'Jaipur', '2023-05-09', 'SL', 900, 300);
```

```
--
```

```
-- Triggers `classeats`
```

```
--
```

```
DROP TRIGGER IF EXISTS `before_insert_on_classeats`;
```

```
DELIMITER //
```

```
CREATE TRIGGER `before_insert_on_classeats` BEFORE INSERT ON `classeats`
```

```
FOR EACH ROW begin
```

```
if datediff(curdate(),new.doj)>0 then
```

```
SIGNAL SQLSTATE '45000'
```

```
SET MESSAGE_TEXT = 'Check date!!!';
```

```
end if;
```

```
if new.fare<=0 then
```

```
SIGNAL SQLSTATE '45000'
```

```
SET MESSAGE_TEXT = 'Check fare!!!';
```

```
end if;
```

```
if new.seatsleft<=0 then
```

```
SIGNAL SQLSTATE '45000'
```

```
SET MESSAGE_TEXT = 'Check seats!!!';
```

```
end if;
```

```
end
```

```
//
```

```
DELIMITER ;
```

```
DROP TRIGGER IF EXISTS `before_update_on_classeats`;
```

```
DELIMITER //
```

```
CREATE TRIGGER `before_update_on_classeats` BEFORE UPDATE ON `classeats`
```

```
FOR EACH ROW begin
```

```
if datediff(curdate(),new.doj)>0 then
```

```
SIGNAL SQLSTATE '45000'
```

```
SET MESSAGE_TEXT = 'check date!!!';
```

```
end if;
```

```
if new.fare<=0 then
```

```
SIGNAL SQLSTATE '45000'
```

```
SET MESSAGE_TEXT = 'Check fare!!!';
```

```
end if;
```

```
if new.seatsleft<=0 then
```

```
SIGNAL SQLSTATE '45000'
```

```
SET MESSAGE_TEXT = 'Check seats!!!';
```

```
end if;
```

```
end
```



```
//
DELIMITER ;

-----

--
-- Table structure for table `pd` passenger details
--

CREATE TABLE IF NOT EXISTS `pd` (
  `pnr` int(11) NOT NULL,
  `pname` varchar(50) NOT NULL,
  `page` int(11) NOT NULL,
  `pgender` varchar(10) NOT NULL,
  PRIMARY KEY (`pnr`,`pname`,`page`,`pgender`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
-- Dumping data for table `pd`
--

INSERT INTO `pd` (`pnr`, `pname`, `page`, `pgender`) VALUES
(58, 'akhil', 20, 'M'),
(58, 'deepak', 21, 'M'),
(58, 'rahul', 12, 'M'),
(58, 'shyam', 50, 'M'),
(59, 'abhinav', 20, 'M'),
(59, 'vikas', 40, 'M'),
(60, 'mohan', 20, 'M');

--
-- Triggers `pd`
--

DROP TRIGGER IF EXISTS `before_insert_on_pd`;
DELIMITER //
CREATE TRIGGER `before_insert_on_pd` BEFORE INSERT ON `pd`
FOR EACH ROW begin
if new.pgender NOT IN ('M','F') then
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'Enter M:Male F:Female.';
end if;
end
//
DELIMITER ;
DROP TRIGGER IF EXISTS `before_update_on_pd`;
DELIMITER //
```

```

CREATE TRIGGER `before_update_on_pd` BEFORE UPDATE ON `pd`
FOR EACH ROW begin
if new.pggender NOT IN ('M','F') then
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'Enter M:Male F:Female.';
end if;
end
//
DELIMITER ;

-- -----

--
-- Table structure for table `resv`
--

CREATE TABLE IF NOT EXISTS `resv` (
  `pnr` int(11) NOT NULL AUTO_INCREMENT,
  `id` int(11) NOT NULL,
  `trainno` int(11) NOT NULL,
  `sp` varchar(50) NOT NULL,
  `dp` varchar(50) NOT NULL,
  `doj` date NOT NULL,
  `tfare` int(11) NOT NULL,
  `class` varchar(50) NOT NULL,
  `nos` int(11) NOT NULL,
  `status` varchar(50) NOT NULL,
  PRIMARY KEY (`pnr`),
  UNIQUE KEY `UNIQUE` (`id`,`trainno`,`doj`,`status`),
  UNIQUE KEY `pnr` (`pnr`,`id`,`trainno`,`doj`,`class`,`status`),
  UNIQUE KEY `pnr_2` (`pnr`,`id`,`trainno`,`sp`,`dp`,`doj`,`tfare`,`class`,`nos`,`status`),
  KEY `FK_ID` (`id`),
  KEY `FK_TN_DOJ_C` (`trainno`,`doj`,`class`),
  KEY `class` (`class`),
  KEY `sp` (`sp`,`dp`),
  KEY `dp` (`dp`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=61 ;

--
-- Dumping data for table `resv`
--

INSERT INTO `resv` (`pnr`,`id`,`trainno`,`sp`,`dp`,`doj`,`tfare`,`class`,`nos`,`status`)
VALUES
(51, 4, 12, 'Chandigarh', 'Jaipur', '2015-05-07', 3300, 'AC1', 2, 'BOOKED'),
(57, 5, 12, 'Chandigarh', 'Jaipur', '2015-05-07', 2200, 'AC1', 1, 'CANCELLED'),

```

```
(58, 6, 20, 'Delhi', 'Jaipur', '2015-05-09', 11200, 'AC2', 4, 'CANCELLED'),  
(59, 10, 12, 'Lucknow', 'Delhi', '2015-05-17', 2200, 'EC', 2, 'BOOKED');
```

```
--  
-- Triggers `resv`  
--  
DROP TRIGGER IF EXISTS `after_insert_on_resv`;  
DELIMITER //  
CREATE TRIGGER `after_insert_on_resv` AFTER INSERT ON `resv`  
  FOR EACH ROW begin  
  UPDATE classseats SET seatsleft=seatsleft-new.nos where trainno=new.trainno AND  
  class=new.class AND doj=new.doj AND sp=new.sp AND dp=new.dp;  
end  
//  
DELIMITER ;  
DROP TRIGGER IF EXISTS `after_update_on_resv`;  
DELIMITER //  
CREATE TRIGGER `after_update_on_resv` AFTER UPDATE ON `resv`  
  FOR EACH ROW begin  
  if (new.status='CANCELLED' AND datediff(new.doj,curdate())<0 ) then  
  SIGNAL SQLSTATE '45000'  
  SET MESSAGE_TEXT = 'Cancellation Not Possible!!!!';  
  end if;  
  
  if (new.status='CANCELLED' AND datediff(new.doj,curdate())>0 )then  
  UPDATE classseats SET seatsleft=seatsleft+new.nos where trainno=new.trainno AND  
  class=new.class AND doj=new.doj AND sp=new.sp AND dp=new.dp;  
  if datediff(new.doj,curdate())>=30 then  
  INSERT INTO canc values (new.pnr,new.tfare);  
  end if;  
  if datediff(new.doj,curdate())<30 then  
  INSERT INTO canc values (new.pnr,0.5*new.tfare);  
  end if;  
  end if;  
end  
//  
DELIMITER ;  
DROP TRIGGER IF EXISTS `before_insert_on_resv`;  
DELIMITER //  
CREATE TRIGGER `before_insert_on_resv` BEFORE INSERT ON `resv`  
  FOR EACH ROW begin  
  if new.tfare<0 then  
  SIGNAL SQLSTATE '45000'  
  SET MESSAGE_TEXT = 'Negative balance NOT possible';  
  end if;  
  if new.nos<=0 then
```

```

SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'Negative OR 0 seats NOT possible';
end if;
if (select seatsleft from classseats where trainno=new.trainno AND class=new.class AND
doj=new.doj AND sp=new.sp AND dp=new.dp) < new.nos then
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'Not enough seats available!!!';
end if;
if datediff(new.doj,curdate())<0 then
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'Booking Not Possible!!!!';
end if;
SET new.status='BOOKED';
end
//
DELIMITER ;
DROP TRIGGER IF EXISTS `before_update_on_resv`;
DELIMITER //
CREATE TRIGGER `before_update_on_resv` BEFORE UPDATE ON `resv`
FOR EACH ROW begin
if new.tfare<0 then
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'Negative balance NOT possible';
end if;
if new.nos<=0 then
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'Negative OR 0 seats NOT possible';
end if;
if (select seatsleft from classseats where trainno=new.trainno AND class=new.class AND
doj=new.doj AND sp=new.sp AND dp=new.dp) < new.nos then
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'Not enough seats available!!!';
end if;
end
//
DELIMITER ;

-- -----
--
-- Table structure for table `schedule`
--

CREATE TABLE IF NOT EXISTS `schedule` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `trainno` int(11) NOT NULL,

```

```

`sname` varchar(50) NOT NULL,
`arrival_time` time NOT NULL,
`departure_time` time NOT NULL DEFAULT '00:00:00',
`distance` int(11) NOT NULL,
PRIMARY KEY (`id`),
KEY `trainno` (`trainno`),
KEY `sname` (`sname`),
KEY `id` (`id`),
KEY `distance` (`distance`),
KEY `id_2` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=42 ;

```

```

--
-- Dumping data for table `schedule`
--

```

```

INSERT INTO `schedule` (`id`, `trainno`, `sname`, `arrival_time`, `departure_time`, `distance`)
VALUES
(1, 12, 'Chandigarh', '01:00:12', '01:00:00', 0),
(2, 12, 'Jaipur', '03:45:15', '03:50:00', 100),
(3, 12, 'Kolkata', '05:00:00', '05:15:00', 300),
(4, 12, 'Lucknow', '11:50:10', '12:00:00', 450),
(5, 12, 'Delhi', '16:30:00', '16:30:00', 600),
(6, 13, 'Jammu Kashmir', '22:00:00', '22:00:00', 0),
(7, 13, 'Delhi', '04:00:00', '04:05:00', 700),
(8, 13, 'Rajasthan', '07:30:50', '07:33:00', 900),
(9, 13, 'Allahbad', '09:00:00', '09:10:00', 1700),
(10, 13, 'Patna', '11:45:00', '11:47:00', 2500),
(11, 13, 'Madhya Pradesh', '13:00:00', '13:00:00', 3600),
(12, 14, 'Jammu Kashmir', '01:00:12', '01:00:12', 0),
(13, 14, 'Madras', '22:00:00', '22:00:00', 2500),
(14, 15, 'Delhi', '16:00:00', '16:00:00', 0),
(15, 15, 'Jaipur', '22:45:00', '22:45:00', 800),
(16, 16, 'Jaipur', '03:30:00', '03:30:00', 0),
(17, 16, 'Delhi', '09:30:00', '09:30:00', 800),
(18, 17, 'Delhi', '00:00:14', '00:00:14', 0),
(19, 17, 'Jaipur', '16:00:00', '16:10:00', 500),
(20, 17, 'Chandigarh', '20:30:00', '20:30:00', 1200),
(21, 18, 'Chandigarh', '08:05:00', '08:05:00', 0),
(22, 18, 'Jaipur', '10:15:00', '10:20:00', 700),
(23, 18, 'Delhi', '14:00:00', '14:00:00', 1200),
(24, 6, 'Jaipur', '03:30:00', '03:30:00', 0),
(25, 6, 'Allahbad', '08:00:00', '08:15:00', 200),
(26, 6, 'Lucknow', '15:15:00', '15:15:00', 700),
(27, 19, 'Lucknow', '13:30:00', '13:30:00', 0),
(28, 19, 'Allahbad', '20:00:00', '20:10:00', 300),

```

```
(29, 19, 'Jaipur', '05:15:00', '05:15:00', 700),
(30, 20, 'Delhi', '10:04:00', '10:04:00', 0),
(31, 20, 'Jaipur', '16:00:00', '16:00:00', 800),
(32, 21, 'Jaipur', '20:00:00', '20:00:00', 0),
(33, 21, 'Delhi', '10:00:00', '10:00:00', 800),
(34, 22, 'Delhi', '16:35:00', '16:35:00', 0),
(35, 22, 'Rajasthan', '20:00:00', '20:10:00', 1100),
(36, 22, 'Madhya Pradesh', '03:30:00', '03:33:00', 1500),
(37, 22, 'Mumbai', '09:00:00', '09:00:00', 2300),
(38, 23, 'Mumbai', '01:00:00', '01:00:00', 0),
(39, 23, 'Madhya Pradesh', '05:30:00', '05:40:00', 1500),
(40, 23, 'Rajasthan', '15:45:00', '15:50:00', 2000),
(41, 23, 'Delhi', '20:30:00', '20:30:00', 2300);
```

```
-- -----
```

```
--
-- Table structure for table `station`
--
```

```
CREATE TABLE IF NOT EXISTS `station` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `sname` varchar(50) NOT NULL,
  PRIMARY KEY (`sname`),
  KEY `id` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=14 ;
```

```
--
-- Dumping data for table `station`
--
```

```
INSERT INTO `station` (`id`, `sname`) VALUES
(1, 'Chandigarh'),
(2, 'Delhi'),
(3, 'Jaipur'),
(4, 'Lucknow'),
(5, 'Mumbai'),
(6, 'Allahbad'),
(7, 'Kolkata'),
(8, 'Patna'),
(9, 'Madras'),
(10, 'Jammu Kashmir'),
(11, 'Rajasthan'),
(12, 'Madhya Pradesh');
```

```
-- -----
```

```
--  
-- Table structure for table `train`  
--
```

```
CREATE TABLE IF NOT EXISTS `train` (  
  `trainno` int(11) NOT NULL AUTO_INCREMENT COMMENT 'Train_no',  
  `tname` varchar(50) NOT NULL COMMENT 'Train_name',  
  `sp` varchar(50) NOT NULL COMMENT 'Starting_Point',  
  `st` time NOT NULL COMMENT 'Arrival_Time',  
  `dp` varchar(50) NOT NULL COMMENT 'Destination_Point',  
  `dt` time NOT NULL,  
  `dd` varchar(10) DEFAULT NULL COMMENT 'Day',  
  `distance` int(11) NOT NULL COMMENT 'Distance',  
  PRIMARY KEY (`trainno`),  
  KEY `sp` (`sp`),  
  KEY `dp` (`dp`)  
) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=24 ;
```

```
--  
-- Dumping data for table `train`  
--
```

```
INSERT INTO `train` (`trainno`, `tname`, `sp`, `st`, `dp`, `dt`, `dd`, `distance`) VALUES  
(6, 'Ashram Express', 'Jaipur', '10:00:00', 'Lucknow', '21:30:00', 'Day 1', 700),  
(12, 'Shatabdi Express', 'Chandigarh', '01:00:12', 'Delhi', '16:30:00', 'Day 1', 600),  
(13, 'Harijan Express', 'Jammu Kashmir', '22:00:00', 'Madhya Pradesh', '13:00:00', 'Day2', 3600),  
(14, 'Jammu Mail Express', 'Jammu Kashmir', '01:00:12', 'Madras', '22:00:00', 'Day 1', 2500),  
(15, 'Delhi Jaipur Double Decker', 'Delhi', '16:00:00', 'Jaipur', '22:45:00', 'Day 1', 800),  
(16, 'Jaipur Delhi Double Decker', 'Jaipur', '03:30:00', 'Delhi', '09:30:00', 'Day 1', 800),  
(17, 'Delhi Chandigarh Shatabdi', 'Delhi', '00:00:14', 'Chandigarh', '20:30:00', 'Day 1', 1200),  
(18, 'Chandigarh Delhi Shatabdi', 'Chandigarh', '08:05:00', 'Delhi', '14:00:00', 'Day 2', 1200),  
(19, 'Ashram Express', 'Lucknow', '13:30:00', 'Jaipur', '05:15:00', 'Day 2', 700),  
(20, 'Frontier Express', 'Delhi', '10:04:00', 'Jaipur', '16:00:00', 'Day 1', 800),  
(21, 'Frontier Express', 'Jaipur', '20:00:00', 'Delhi', '10:00:00', 'Day 2', 800),  
(22, 'Rajdhani Express', 'Delhi', '16:35:00', 'Mumbai', '09:00:00', 'Day 2', 2300),  
(23, 'Rajdhani Express', 'Mumbai', '01:00:00', 'Delhi', '20:30:00', 'Day 1', 2300);
```

```
--  
-- Triggers `train`  
--
```

```
DROP TRIGGER IF EXISTS `before_insert_on_train`;  
DELIMITER //  
CREATE TRIGGER `before_insert_on_train` BEFORE INSERT ON `train`  
FOR EACH ROW begin  
if (new.dt<new.st AND new.dd='Day 1') then
```

```

SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'Improper Timings';
end if;
if (new.dp=new.sp) then
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'Same Starting & Destination Points not allowed';
end if;
end
//
DELIMITER ;
DROP TRIGGER IF EXISTS `before_update_on_train`;
DELIMITER //
CREATE TRIGGER `before_update_on_train` BEFORE UPDATE ON `train`
FOR EACH ROW begin
if (new.dt<new.st AND new.dd='Day 1') then
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'Improper Timings';
end if;
end
//
DELIMITER ;

-- -----
--
-- Table structure for table `user`
--

CREATE TABLE IF NOT EXISTS `user` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `emailid` varchar(50) NOT NULL,
  `password` varchar(50) NOT NULL,
  `mobilenno` varchar(10) NOT NULL,
  `dob` date NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `UNIQUEMN` (`mobilenno`),
  UNIQUE KEY `UNIQUEEI` (`emailid`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=21 ;

--
-- Dumping data for table `user`
--

INSERT INTO `user` (`id`, `emailid`, `password`, `mobilenno`, `dob`) VALUES
(4, 'garvitjain@gmail.com', 'garvit', '9312201852', '1994-01-01'),
(5, 'deepakgoel@hotmail.com', 'deepak', '9312201853', '1994-02-22'),

```



```
(6, 'akhilkumar@yahoo.co.in', 'akhil', '9872231234', '1994-03-04'),
(7, 'ayushjain@outlook.com', 'ayush', '9810150525', '1995-01-03'),
(8, 'aakashbharadwaj@yahoo.com', 'aakash', '9013452635', '1993-12-30'),
(10, 'abhinavsingh@gmail.com', 'abhinav', '9876675567', '1991-01-01'),
(12, 'amanmalik@hotmail.com', 'aman', '9878876654', '1997-09-08'),
(19, 'dhruvgosian@gmail.com', 'dhruv', '9807890453', '1965-04-01'),
(20, 'chiragbansal@nsitonline.com', 'chirag', '9123456789', '1960-06-02');
```

```
--
```

```
-- Triggers `user`
```

```
--
```

```
DROP TRIGGER IF EXISTS `before_insert_on_user`;
```

```
DELIMITER //
```

```
CREATE TRIGGER `before_insert_on_user` BEFORE INSERT ON `user`
```

```
FOR EACH ROW begin
```

```
if (year(curdate())-year(new.dob))<18 then
```

```
SIGNAL SQLSTATE '45000'
```

```
SET MESSAGE_TEXT = 'Minimum age bar of 18 years.';
```

```
end if;
```

```
end
```

```
//
```

```
DELIMITER ;
```

```
DROP TRIGGER IF EXISTS `before_update_on_user`;
```

```
DELIMITER //
```

```
CREATE TRIGGER `before_update_on_user` BEFORE UPDATE ON `user`
```

```
FOR EACH ROW begin
```

```
if (year(curdate())-year(new.dob))<18 then
```

```
SIGNAL SQLSTATE '45000'
```

```
SET MESSAGE_TEXT = 'Minimum age bar of 18 years.';
```

```
end if;
```

```
end
```

```
//
```

```
DELIMITER ;
```

```
--
```

```
-- Constraints for dumped tables
```

```
--
```

```
--
```

```
-- Constraints for table `classseats`
```

```
--
```

```
ALTER TABLE `classseats`
```

```
ADD CONSTRAINT `classseats_ibfk_1` FOREIGN KEY (`trainno`) REFERENCES `train`
(`trainno`),
```

```
ADD CONSTRAINT `classseats_ibfk_3` FOREIGN KEY (`sp`) REFERENCES `station`
(`sname`),
```

```

    ADD CONSTRAINT `classseats_ibfk_4` FOREIGN KEY (`dp`) REFERENCES `station`
(`sname`),
    ADD CONSTRAINT `classseats_ibfk_5` FOREIGN KEY (`class`) REFERENCES `class`
(`cname`);

--
-- Constraints for table `resv`
--
ALTER TABLE `resv`
    ADD CONSTRAINT `resv_ibfk_1` FOREIGN KEY (`trainno`) REFERENCES `train` (`trainno`),
    ADD CONSTRAINT `resv_ibfk_2` FOREIGN KEY (`sp`) REFERENCES `station` (`sname`),
    ADD CONSTRAINT `resv_ibfk_3` FOREIGN KEY (`dp`) REFERENCES `station` (`sname`);

/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;

```

CHAPTER 6

RESULTS AND DISCUSSION

6.1 CREATING NEW USER

Kindly enter your credentials!!!

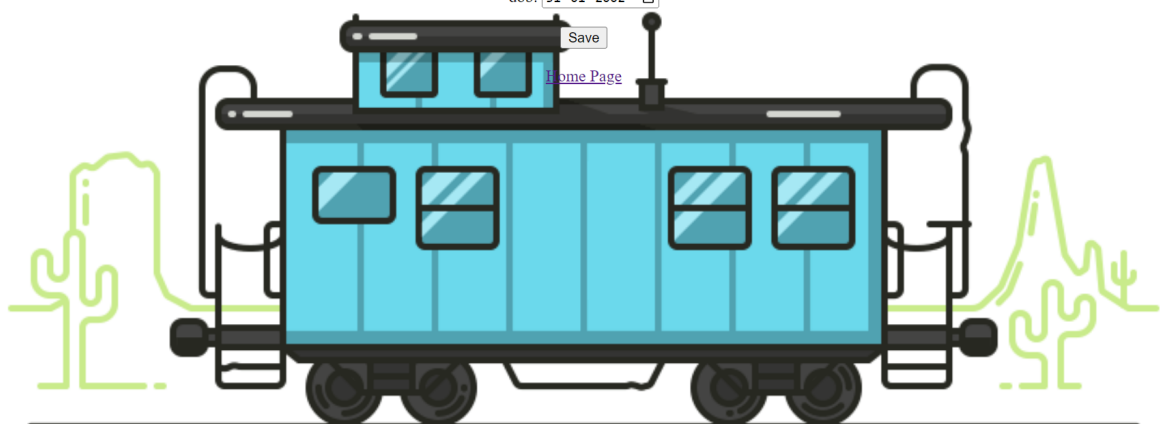
Password:

E-mail ID:

Mobile No.:

dob:

[Save](#) [Home Page](#)



phpMyAdmin Server: 127.0.0.1 » Database: railway » Table: user

Showing rows 0 - 0 (1 total, Query took 0.0002 seconds.)

```
SELECT * FROM `user`
```

☐ Profiling [\[Edit inline \]](#) [\[Edit \]](#) [\[Explain SQL \]](#) [\[Create PHP code \]](#) [\[Refresh \]](#)

☐ Show all | Number of rows: 25 | Filter rows:

Extra options

	id	emailid	password	mobilen	dob
<input type="checkbox"/>	21	ar5362@srmist.edu.in	aNURAG31	9503085801	2002-01-31

☐ Check all | With selected: [Edit](#) [Copy](#) [Delete](#) [Export](#)

☐ Show all | Number of rows: 25 | Filter rows:

Query results operations

[Print](#) [Copy to clipboard](#) [Export](#) [Display chart](#) [Create view](#)

Console

Creating new user

In the above screenshots we can see how when information is entered into the New User Portal, the INSET QUERRY gets executed and the data is entered into the database and now the user can log into their account using this information.

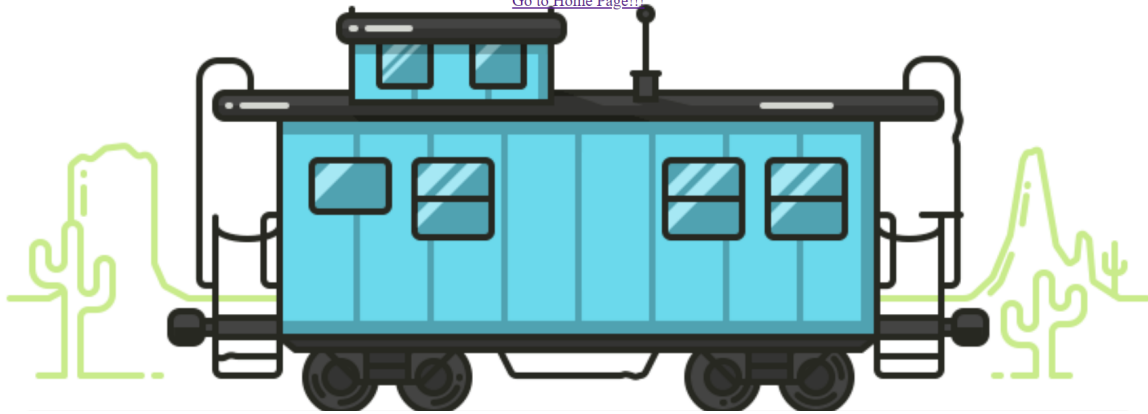
6.2 ENQUIRY

Starting Point:

Destination Point:

Date of Journey: ☐

[Go to Home Page!!!](#)



Train No	Train_Name	Starting_Point	Arrival_Time	Destination_Point	Departure_Time	Day	Train_Class	Fare	Seats_Left
12	Shatabdi Express	Chandigarh	01:00:00	Jaipur	03:45:15	Day 1	AC1	3200	20

If you wish to proceed with booking fill in the following details:

Registered Mobile No:

Password:

Enter Train No:

Enter Class:

No. of Seats:

[More Enquiry](#)

[Go to Home Page!!!](#)

Total fare is Rs.3200/-

Reservation Successful

Passenger details added!!!

[Go Back!!!](#)



phpMyAdmin

Recent Favorites

- New
- dbms_p1
- information_schema
- mysql
- performance_schema
- phpmyadmin
- railway
 - New
 - canc
 - class
 - classseats
 - pd
 - resv
 - schedule
 - station
 - train
 - user
- test

Server: 127.0.0.1 » Database: railway » Table: resv

Browse Structure SQL Search Insert Export Import Privileges Operations Triggers

Showing rows 0 - 0 (1 total, Query took 0.0002 seconds.)

SELECT * FROM `resv`

☐ Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

☐ Show all | Number of rows: 25 | Filter rows: Search this table

Extra options

		pnr	id	trainno	sp	dp	doj	tfare	class	nos	status		
<input type="checkbox"/>	Edit	Copy	Delete	61	21	12	Chandigarh	Jaipur	2023-05-02	3200	AC1	1	BOOKED

☐ Check all | With selected: Edit Copy Delete Export

☐ Show all | Number of rows: 25 | Filter rows: Search this table

Query results operations

Print Copy to clipboard Export Display chart Create view

In the above images we can see how the ticket has been booked for 2nd May and the reservation has been added to the database.

6.3 CANCELLATION

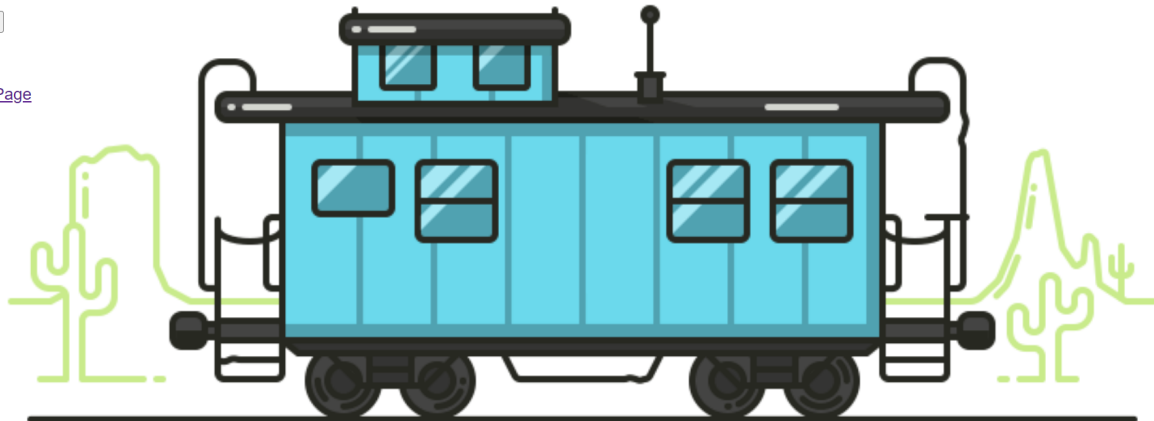
Welcome ar5362@srmist.edu.in

PNR	Train_no	Date_Of_Journey	Total_Fare	Train_Class	Seats_Reserved	Status
61	12	2023-05-02	3200	AC1	1	BOOKED

Enter PNR for Cancellation: 61

Cancel

[Home Page](#)



We have entered the details of the cancellation and canceled the booking.

phpMyAdmin

Server: 127.0.0.1 » Database: railway » Table: resv

Showing rows 0 - 0 (1 total, Query took 0.0009 seconds.)

SELECT * FROM `resv`

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 | Filter rows: Search this table

Extra options

	pnr	id	trainno	sp	dp	doj	tfare	class	nos	status
<input type="checkbox"/>	61	21	12	Chandigarh	Jaipur	2023-05-02	3200	AC1	1	CANCELLED

Check all | With selected: Edit Copy Delete Export

Show all | Number of rows: 25 | Filter rows: Search this table

Query results operations

Print Copy to clipboard Export Display chart Create view

Console

The status of the ticket has been changed to CANCELED.

CHAPTER 7

CONCLUSION AND FUTURE ENHANCEMENT

7.1 CONCLUSION:

Our system can successfully give information on any train and the available seats that can be booked right at the moment, find trains running between two stations, book tickets and cancel tickets. This system could be used for official train bookings. The proposed system gives transparency about the tickets to the customers. However several other features could be added like booking meals on trains at the next station etc. Also payment gateways have to be implemented to make sure the transactions happen securely.

7.2 FUTURE ENHANCEMENT

- We can even further make it private and secured by implementing Log-in IDs and encrypting them with passwords.
- We can give away this software for more number of people and organizations to conduct a Beta Testing and based upon the results we can just make those changes and be assured of the application developed.
- We can make it more space and resource efficient so that this application consumes lesser RAM and ROM and battery power (if available).

REFERENCES:

- [1]Profillidis, Vassilios. Railway management and engineering. Routledge, 2016
- [2]Tanwar, Rajneesh, et al. "Railway reservation verification by aadhar card." *Procedia Computer Science* 85 (2016): 970-975.
- [3]Chatterjee, Parag, and Asoke Nath. "Smart Computing Applications in Railway Systems-A case study in Indian Railways Passenger Reservation System." *International Journal* 3.4 (2014): 61-66.
- [4]Rakshit, Sandip, et al. "Automatic processing of structured handwritten documents: An application for the Indian railway reservation system." *International Journal of Computer Applications* 6.11 (2010): 26-30.
- [5]Srivastava, Shirish C., Sharat S. Mathur, and Thompson SH Teo. "Modernization of passenger reservation system: Indian Railways' dilemma." *Journal of Information Technology* 22.4 (2007): 432-439.
- [6]Agarwal, S. (2013). Computerized Passenger Reservation System for Indian Railways–Its Development and System Architecture. *Journal of Engineering, Computers & Applied Sciences (JEC&AS)*, 2(6).
- [7]Shandilya, S., and P. Rajmohan. "Cyber Threat: Indian Railways Perspective." *The Indian Police Journal*: 26.
- [8]Gupta, Saumya, et al. "Aadhar Card Based Double Identity Verification System for Railway." *International Journal of Engineering & Technology* 7.2.31 (2018): 259-261.
- [9]Gupta, Saumya, et al. "Aadhar Card Based Double Identity Verification System for Railway." *International Journal of Engineering & Technology* 7.2.31 (2018): 259-261.
- [10]Nunna, Swamy. "Spread of IT in Indian Railways." *The Management Accountant Journal* 56.8 (2021): 46-48.
- [11]Mohapatra, Sanjay, and Sanjay Mohapatra. *E-commerce Strategy*. Springer US, 2013.