Mini Project Report on

Railway Booking System

Submitted by:

Atharva Ghodmare

Under the Guidance of

Prof. Snehal Umare

At



School of Computer Engineering and Technology

# Abstract:

The Railway Booking System facilitates the passengers to enquire about the trains available on the basis of source and destination, Booking and Cancellation of tickets, enquire about the status of the booked ticket, etc. The aim of case study is to design and develop a database maintaining the records of different trains, train status, and passengers. This project contains Introduction to the Railways reservation system. It is the computerized system of reserving the seats of train seats in advanced. On-line reservation has made the process for the reservation of seats very much easier than ever before. In our country India, there are number of counters for the reservation of the seats and one can easily make reservations and get tickets. This project contains ERD diagram which completely explains how entities are related to each other for easy understanding about our project. Along with MySQL for handling the backend we have also used Java for setting up the GUI and making it more users friendly. This project explains well about MySQL Java connectivity and how frontends and backend work in any system [3].

Table of Contents

[**I.** **Abstract:** 2](#_Toc51336928)

[**II.** **List of Abbreviations** 4](#_Toc51336929)

[**III.** **List of Figures** 4](#_Toc51336930)

[**IV.** **List of Tables** 4](#_Toc51336931)

[1. Introduction: 5](#_Toc51336932)

[1.1 Motivation: 6](#_Toc51336933)

[1.2 Objectives: 6](#_Toc51336934)

[1.3 Assumptions: 7](#_Toc51336935)

[2. Problem Definition: 7](#_Toc51336936)

[3. Tools and Technologies Used 8](#_Toc51336937)

[4. Database Design 9](#_Toc51336938)

[5. Database Schema 10](#_Toc51336939)

[5.1 Relational Database Design using schema diagram 12](#_Toc51336940)

[6. Database Normalization 13](#_Toc51336941)

[6.1 First Normal Form 13](#_Toc51336942)

[6.2 Second Normal Form 13](#_Toc51336943)

[6.3 Third Normal Form 14](#_Toc51336944)

[7.0 DDL 15](#_Toc51336945)

[8.0 DML along with the results of the queries 16](#_Toc51336946)

[9.0 DCL 16](#_Toc51336947)

[10.0 Triggers 17](#_Toc51336948)

[11.0 PLSQL procedure/function 19](#_Toc51336949)

[12.0 Frontend GUI screenshots 21](#_Toc51336950)

[13.0 Conclusion: 25](#_Toc51336951)

[14.0 References: 26](#_Toc51336952)

# List of Abbreviations

* DBMS: Database Management System
* GUI: Graphical User Interface
* RDBMS: Relational Database Management System
* JDK: Java Development Kit

# List of Figures

* Entity Relationship diagram
* Schema Diagram
* Relational Schema Diagram

# List of Tables

* Location
* Passenger
* Admin
* Train
* Class
* Seat
* Journey
* Booking

## 1. Introduction:

Database is an organized collection of data. The data is typically organized to model aspects of reality in a way that supports processes requiring information. A DBMS makes it possible for end users to create, read, update and delete data in a database. The DBMS essentially serves as an interface between the database and end users or application programs, ensuring that data is consistently organized and remains easily accessible.

The DBMS manages three important things: the data, the database engine that allows data to be accessed, locked and modified and the database schema, which defines the database’s logical structure. These three foundational elements help provide concurrency, security, data integrity and uniform administration procedures. The DBMS can offer both logical and physical data independence. That means it can protect users and applications from needing to know where data is stored or having to be concerned about changes to the physical structure of data. The main purpose of maintaining database for Railway Reservation System is to reduce the manual errors involved in the booking and cancelling of tickets and make it convenient for the customers and providers to maintain the data about their customers and also about the seats available at them.

Along with DBMS, the use of Java NetBeans [8] makes it much easier to handle databases. An IDE is much more than a text editor. The NetBeans Editor indents lines, matches words and brackets, and highlights source code syntactically and semantically. It lets you easily refactor code, with a range of handy and powerful tools, while it also provides code templates, coding tips, and code generators.

It also helps programmers to make a user-friendly GUI. The NetBeans GUI Builder automatically takes care of correct spacing and alignment, while supporting in-place editing, as well. The GUI builder is so easy to use and intuitive that it has been used to prototype GUIs live at customer presentations.

### Motivation:

India is a densely populated country with a complex railway system used by millions on daily basis. Our system can make the ticket booking and cancelling process more efficient. Railway booking System can be implemented using general purpose Programming Languages but tend to have limited functionality. The most significant drawback is that of data preservation and multiple access.

Although files handling can be used to solve the issue for storage, but it has got disadvantages of its own. All these issues can be resolved by the introduction of DBMS. The future versions of this system can be applied on large scale to solve real life problems faced in today's scenario.

### 1.2 Objectives:

Due to automation many loopholes that exist in the manual maintenance of the records can be removed. The speed of obtaining and processing the data will be fast. For future expansion the proposed system can be web enabled so that clients can make various enquiries about trains between stations.

Due to this, sometimes a lot of problems occur and they are facing many disputes with customers. To solve the above problem, we design a data base which includes customer details, availability of seats in trains, no of trains and their details.

Also, many websites and application are often very confusing and difficult to use. Hence in our project we have tried to make the GUI as appealing as possible. The GUI often decides the value of the product; hence we have made it as user friendly as possible.

The project is being developed with the following objectives in mind:

* A User-friendly GUI for the passengers to access the system
* Implementing variety of MySQL queries for
* The internal functioning of the system should be automated
* Storing data by implementing the concept of RDBMS
* Mimic real-life scenario of a Railway Booking System
* Possible scope for further development and innovation

### 1.3 Assumptions:

We are limiting ourselves with the following assumptions since the system is very substantial and thus is not very feasible to develop and document at that level. Therefore, a small working portion of the actual system will be developed as a part of project.

* The Number of trains has been restricted to 15
* Tickets can be booked by the passengers at the most a week prior to the journey
* Total number of tickets for each class is limited to 20
* Available classes are restricted to General and AC
* Each city has just one station, so the source and destination will just have the city name.

## 2. Problem Definition:

A software has to be developed for automating the manual reservation system of railway. The system should be standalone in nature. The user of this system is an administrator or a passenger. The user can login to the system via email and a password. If a new passenger wants to create an account, he/she can sign-up by filling up mandatory details - Name, Contact, Email, Date of Birth, Aadhar Number, Address and Password. When logged in, the passenger can search for train, book tickets, view booked tickets and cancel booked tickets. Searching takes place on the basis of source, destination, date of travel, number of seats. Each passenger may book 1 or more tickets if seats are available. Passenger can cancel his/her booking at the most two days prior to the schedule. There will be reduction in refund as the scheduled date nears. User with the admin privileges can view and modify all tables. He/she can enter train details- train Id, name, source, destination, arrival time, departure time. Train is uniquely identified by a train id. Each train has multiple classes with limited seats per class. A booking is identified uniquely by a seat, journey and the user associated with it.

## 3. Tools and Technologies Used

* MySQL database 8.0 [10]
* MySQL Connector Java 8.021 [2]
* Java Development Kit (JDK) 14.0.2 GA [11]
* Java Toast Library [12]
* JDBC 4.3 API
* Apache NetBeans IDE 8.2 RC [9]
* Windows 10 64-bit OS

## 4. Database Design

Entity Relationship Diagram with EER features

# 

## 5. Database Schema

* Passenger

(

aadhar\_number DECIMAL (12),

first\_name VARCHAR (30),

last\_name VARCHAR (30),

email VARCHAR (30)),

date\_of\_birth DATE,

street VARCHAR (20),

pin\_code DECIMAL (6),

phone DECIMAL (10),

passenger\_password VARCHAR (15)

)

* Admin

(

adminID INT,

email varchar (30),

admin\_password VARCHAR (15)

)

* Train

(

train\_id INT,

train\_name VARCHAR (20)

)

* Class

(

train\_id INT,

class\_name VARCHAR (20),

fare DECIMAL (10,2)

)

* Journey

(

journey\_id INT,

train\_id INT,

source\_name VARCHAR (20),

destination\_name VARCHAR (20),

departure DATETIME,

arrival DATETIME

)

* Seat

(

seat\_no INT NOT NULL,

journey\_id INT NOT NULL,

class\_name VARCHAR (20) NOT NULL,

booking\_status INT

)

* Booking

(

booking\_id INT,

aadhar\_number DECIMAL (12),

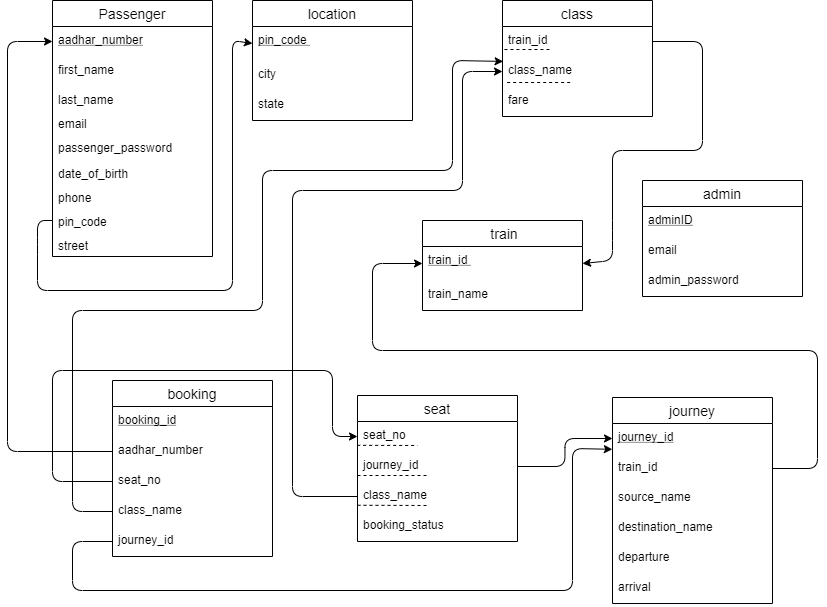
seat\_no INT,

class\_name VARCHAR (20),

journey\_id INT

)

### 5.1 Relational Database Design using schema diagram



## 6. Database Normalization

[5]

### 6.1 First Normal Form

First normal form (1NF) states that the domain of an attribute must include only atomic (simple, indivisible) values and that the value of any attribute in a tuple must be a single value from the domain of that attribute. Consider a table with all possible attributes in a single table. Since all fields taken are single valued, we can say that the tables are in first normal form.

### 6.2 Second Normal Form

To be in second normal form, a relation must be in first normal form (1NF) and it must not contain any partial dependencies. So, a relation is in 2NF as long as it has no partial dependencies, i.e., no non-prime attributes (attributes which are not part of any candidate key) is dependent on any proper subset of a composite primary key of the table. The tables are already in 1NF. Partial dependencies are found, which we eliminate in the following way to attain 2NF.

* City and state fields can be uniquely identified using just the pin code. Hence, we create a table Location like location (pin\_code, city, state)
* Train Name can be uniquely identified using just the Train Id. Hence, we create a table train (train\_id, train\_name).
* Fare can be uniquely identified using Train Id and the Class Name. Hence, we create a table class (train\_id, class\_name, fare).
* Booking Status of the seat can be identified using the seat number, class name and the journey ID. Hence, we create a table seat (seat\_no, journey\_id, class\_name, booking\_status).
* Train\_id, source\_name, destination\_name, departure and arrival can be identified using journey\_id. Hence, we create a table journey (journey\_id, train\_id, source\_name, destination\_name, departure, arrival)

### 6.3 Third Normal Form

For a table to be in the third normal form, it should be in the second normal form and also it should not have transitive dependency. The tables are already in the 2NF and no Transitive dependencies are observed. After completely normalizing we obtain following tables:

Location:

|  |  |  |
| --- | --- | --- |
| pin\_code | city | state |

Passenger:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| aadhar\_number | first\_name | last\_name | email | passenger\_password | date\_of\_birth | phone | pin\_code | street |

Train:

|  |  |
| --- | --- |
| train\_id | train\_name |

Class:

|  |  |  |
| --- | --- | --- |
| train\_id | class\_name | fare |

Journey:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| journey\_id | train\_id | source\_name | destination\_name | departure | arrival |

Seat:

|  |  |  |  |
| --- | --- | --- | --- |
| seat\_no | journey\_id | class\_name | booking\_status |

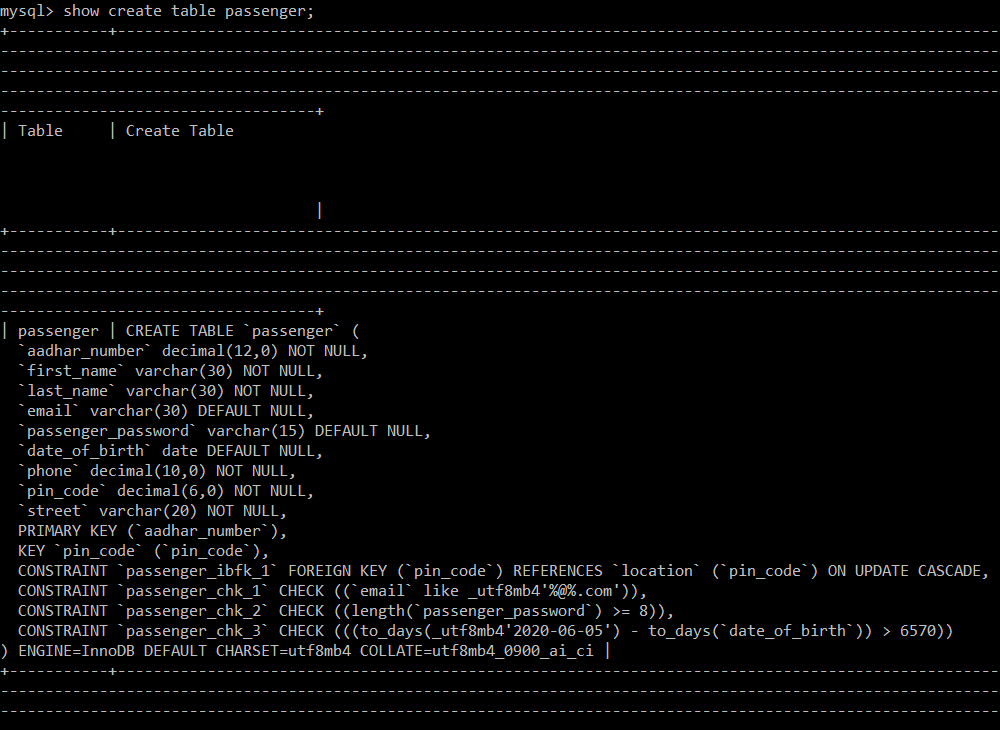
Booking:

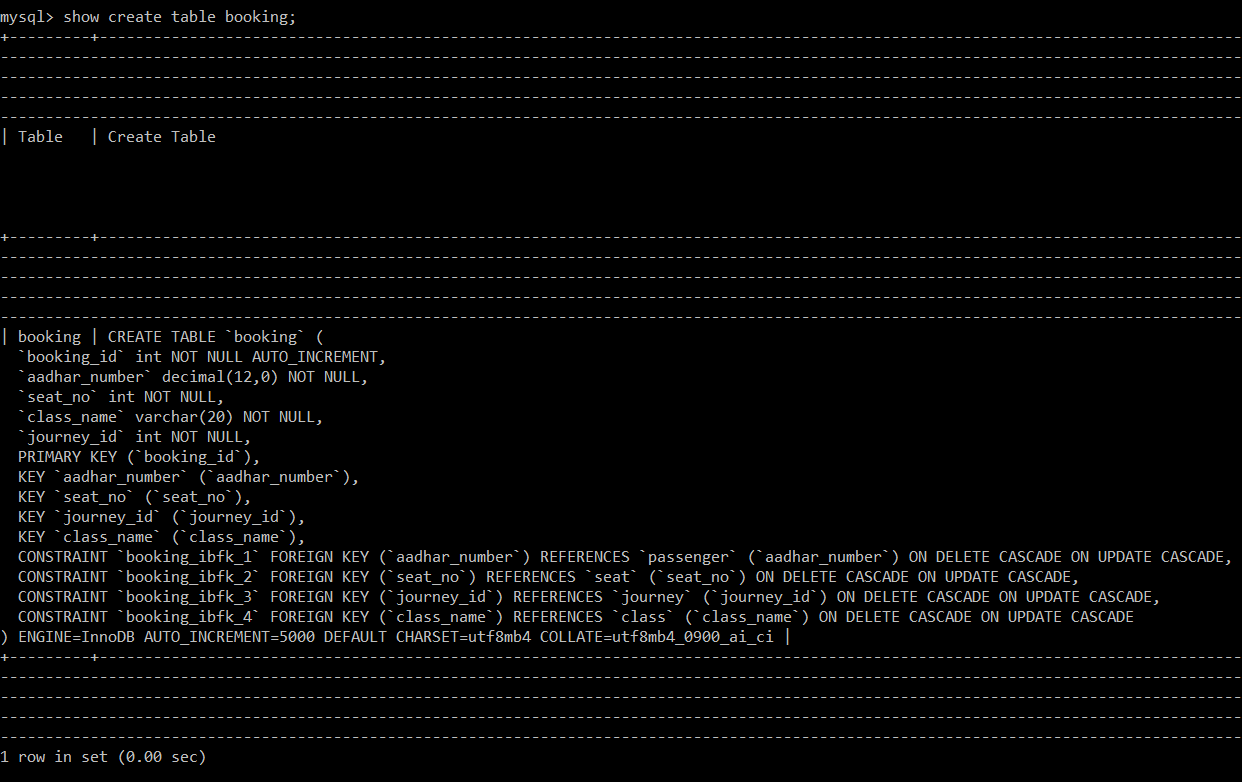
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| booking\_id | aadhar\_number | seat\_no | class\_name | journey\_id |

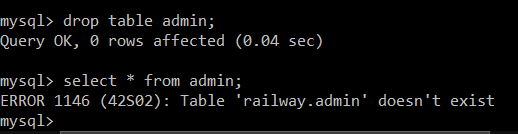
Admin:

|  |  |  |
| --- | --- | --- |
| adminID | email | admin\_password |

## 7.0 DDL Commands

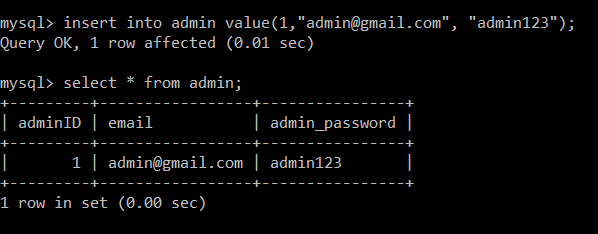




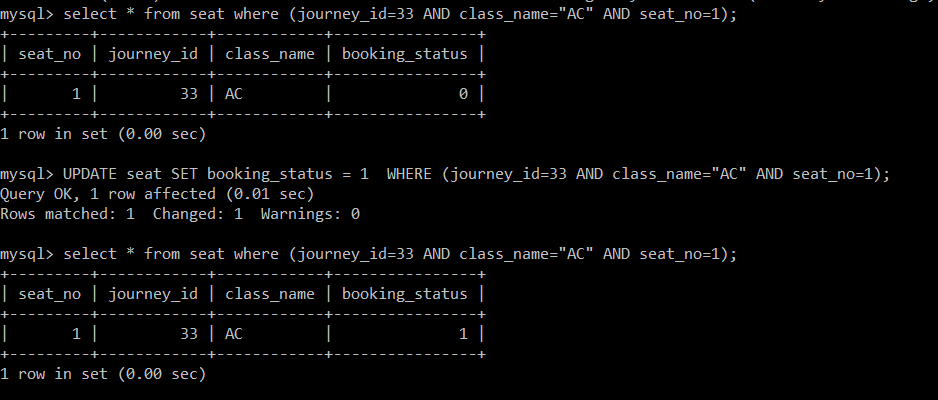


## 8.0 DML Commands

Inserting a value into the table admin



Extracting the record with the following journeyid, classname and seat\_no.



## 9.0 DCL Commands

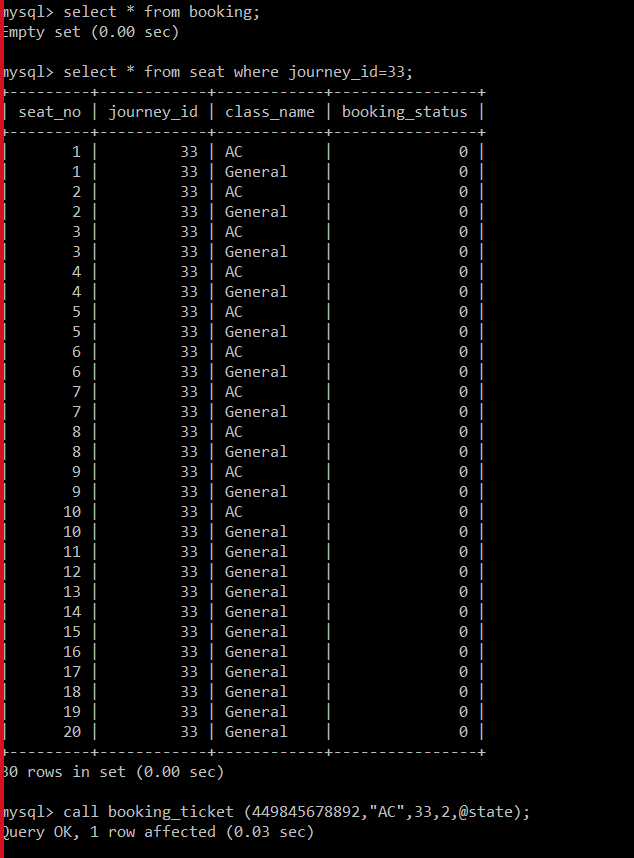
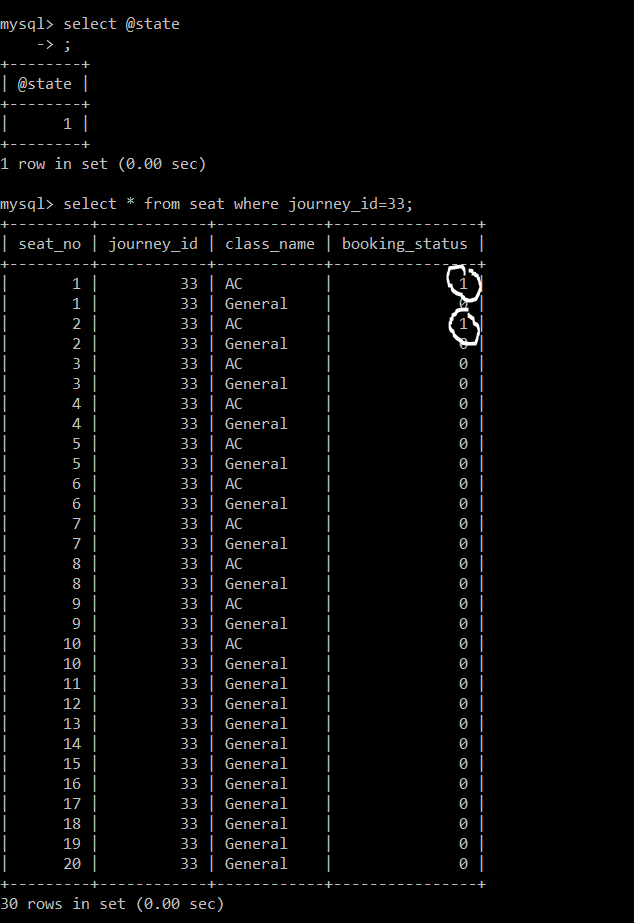
NA

## 

## 10.0 Triggers

after\_booking\_modify\_seat Trigger is implemented to update the seat status from available to booked by implicitly calling a procedure update\_seat\_status.

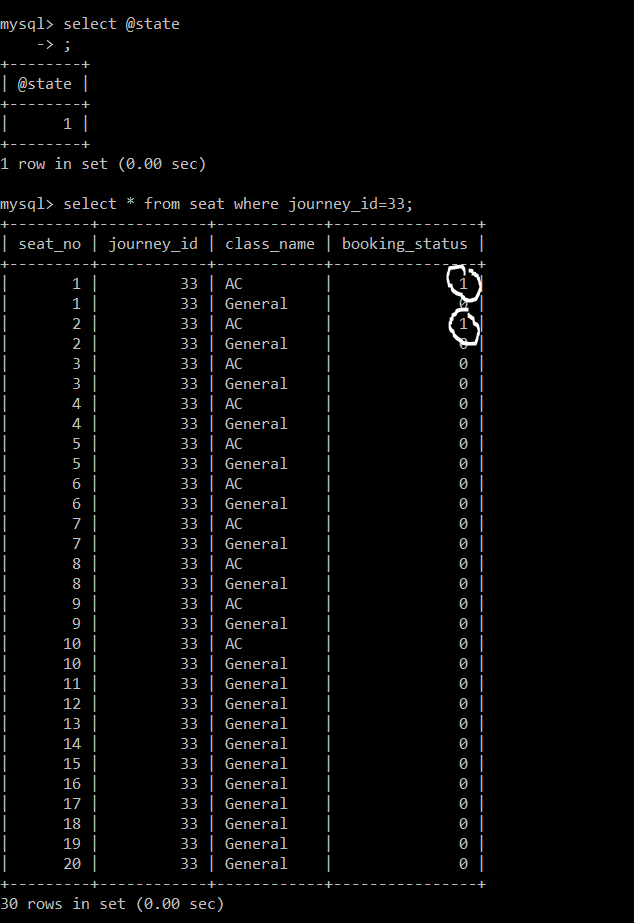
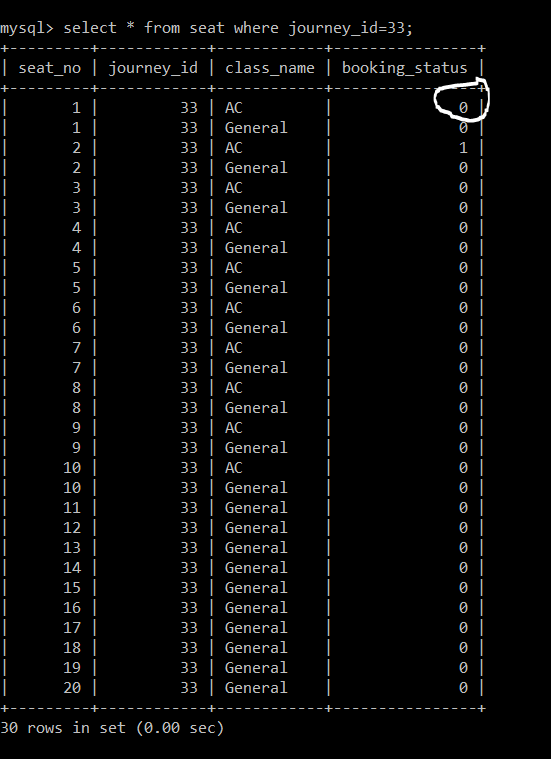
* CREATE TRIGGER after\_booking\_modify\_seat AFTER INSERT ON booking FOR EACH ROW
* BEGIN
* CALL update\_seat\_status (NEW.seat\_no,NEW.class\_name,NEW.journey\_id);
* END &&

Before After

after\_cancellation\_modify\_seat Trigger is implemented to update the seat booking status from booked to available by implicitly calling a procedure update\_seat\_status.

* CREATE TRIGGER after\_cancellation\_modify\_seat AFTER DELETE ON booking FOR EACH ROW
* BEGIN
* CALL update\_seat\_status (OLD.seat\_no,OLD.class\_name,OLD.journey\_id);
* END &&

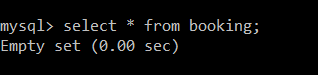
 

Before After

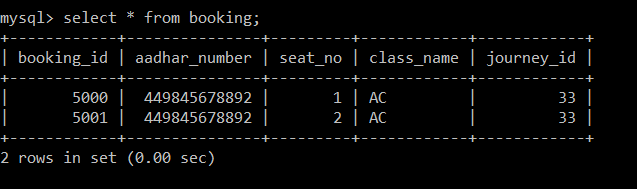
## 11.0 PLSQL procedure/function

Booking\_ticket Procedure is implemented to insert an entry in the booking table and setting off the after\_booking\_modify\_seat trigger. If the booking is successful then the out parameter is set to 1 else it is set to 0.

* CREATE PROCEDURE booking\_ticket
* (IN a\_no DECIMAL(12), IN c\_name VARCHAR(20), IN j\_id INT, IN qty INT, OUT state INT)
* BEGIN
  + DECLARE i INTEGER DEFAULT 1;
* DECLARE s\_no INTEGER;
* DECLARE available INTEGER DEFAULT check\_available\_seats(j\_id,c\_name);
  + IF qty > available THEN
    - SET state=0;
* ELSE
  + - WHILE i<=qty DO
      * SET s\_no=get\_seat\_no(j\_id,c\_name);
      * INSERT INTO booking (aadhar\_number,seat\_no, class\_name,journey\_id) VALUES (a\_no,s\_no,c\_name,j\_id);
* SET i=i+1;
  + - END WHILE;
    - SET state=1;
* END IF;
* END &&



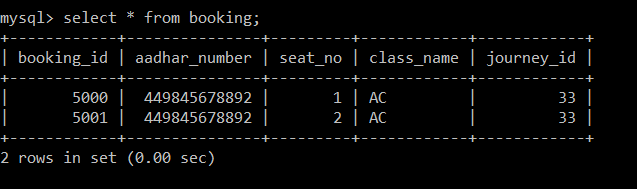
Before Calling booking\_ticket



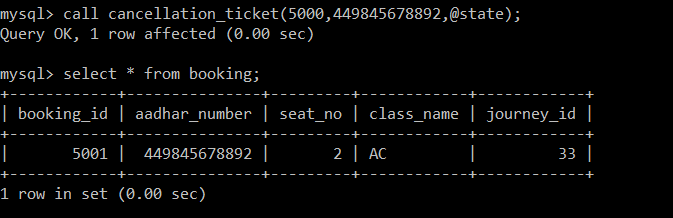
After calling booking\_ticket

cancellation\_ticket Procedure is implemented to delete an entry from the booking table and setting off the after\_cancellation\_modify\_seat trigger. If the cancellation is successful then the out parameter is set to 1 else it is set to 0.

* CREATE PROCEDURE cancellation\_ticket(IN b\_id INT,IN a\_no DECIMAL(12),OUT state INT)
* BEGIN
* DECLARE counter INTEGER DEFAULT 0;
* SELECT count(booking\_id) INTO counter FROM booking WHERE (booking\_id=b\_id AND aadhar\_number=a\_no);
* IF (counter>0) THEN
* DELETE FROM booking WHERE (booking\_id=b\_id AND aadhar\_number=a\_no);
* SET state=1;
* ELSE
* SET state=0;
* END IF;
* END &&



Before calling cancellation\_ticket



After calling cancellation\_ticket

## 12.0 Frontend GUI screenshots

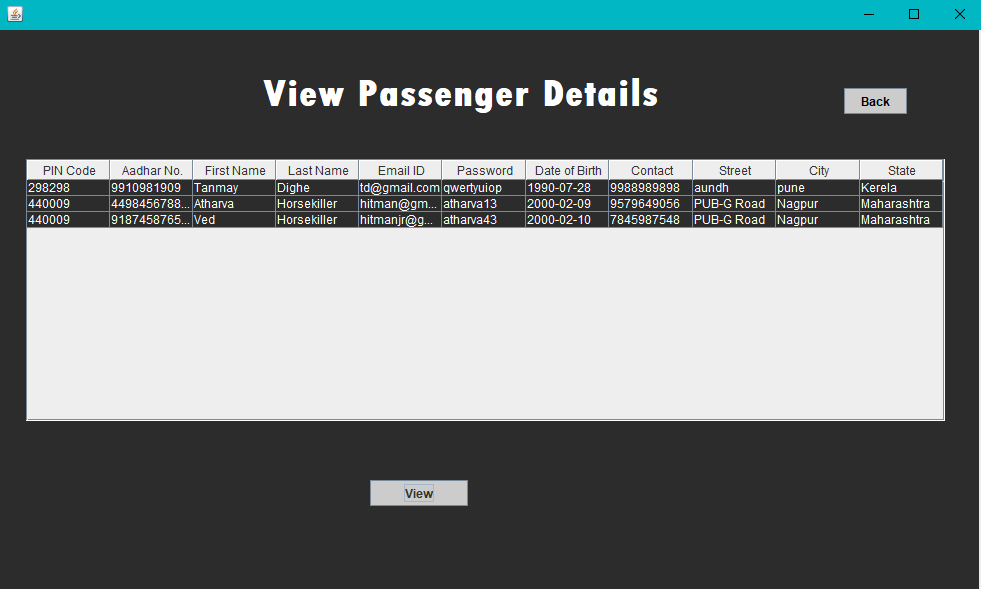
[1] [4]

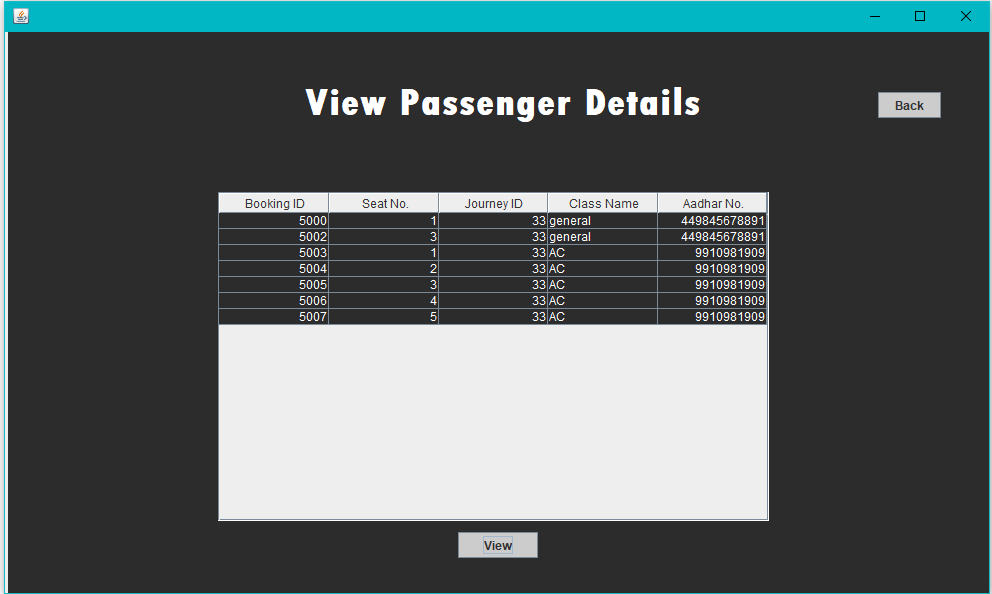
A subway train at a train station

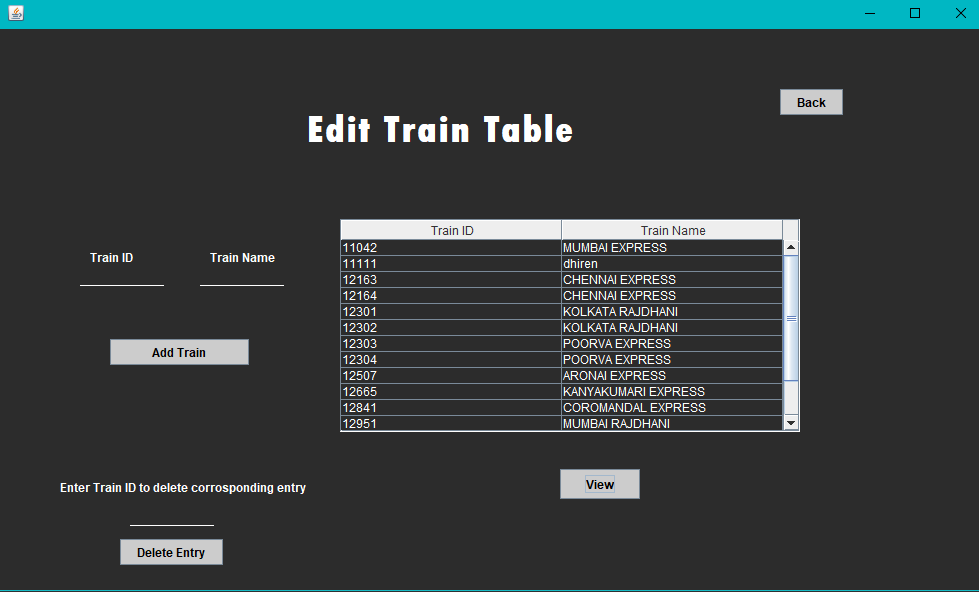
Description automatically generated

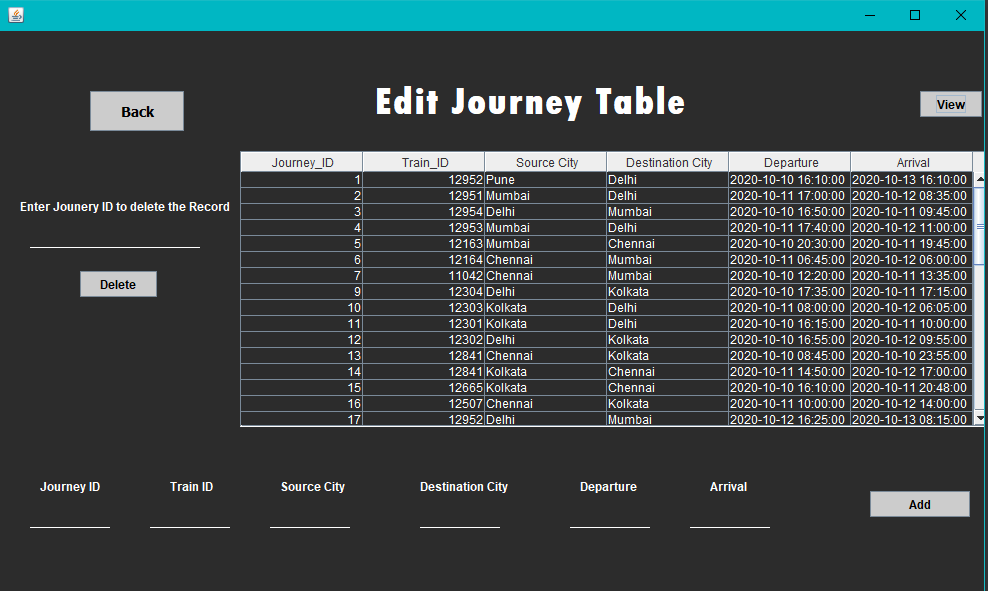
A screenshot of a cell phone

Description automatically generated



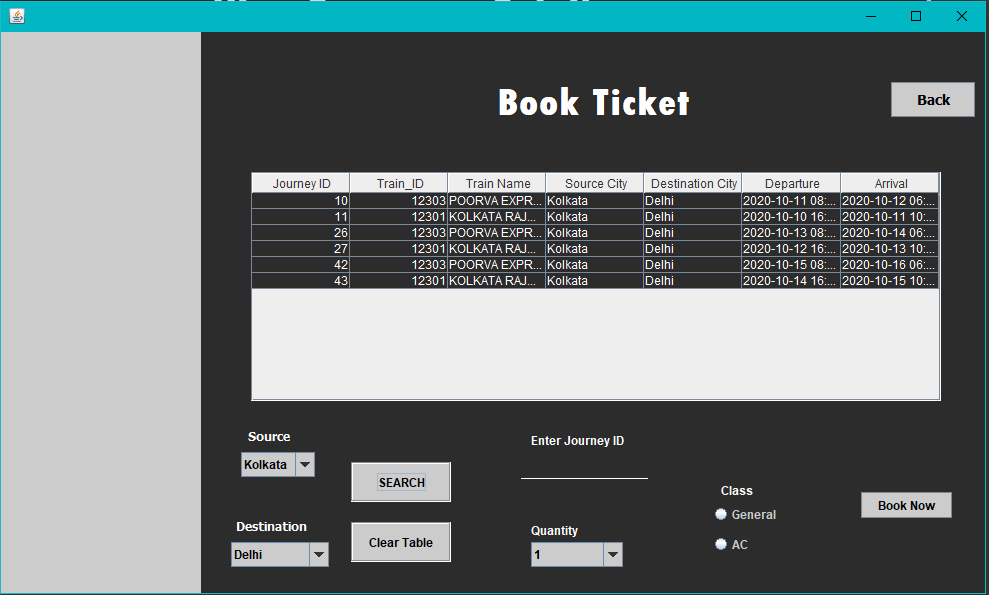






A screenshot of a cell phone

Description automatically generated



A screenshot of a cell phone

Description automatically generated

## 

## A screenshot of a cell phone Description automatically generated

## 13.0 Conclusion:

In our project, Railway reservation system we have stored all the information about the Trains scheduled and the users booking tickets and even status of trains, seats etc. The project implementation facilitates passengers to book the train tickets and check the details of trains and their status from their place itself. It avoids inconveniences of going to railway station for each and every query they get. Although we have considered only the most important requirements, many more features and details which are mentioned in the scope can be added to our project in order to obtain even more functionalities. These applications are already in progress and in future they can be upgraded and may become part of amazing technology.

## 14.0 References:

References:

[1] <https://netbeans.org/kb/docs/ide/mysql.html>

[2] <https://dev.mysql.com/downloads/connector/j/>

[3] <http://www.muengineers.in/computer-project-list/java-projects-list/railway-reservation-system>

[4] <https://stackoverflow.com/questions/17164014/java-lang-classcastexception-java-lang-long-cannot-be-cast-to-java-lang-integer>

[5] <https://beginnersbook.com/2015/05/normalization-in-dbms/>

[6] <https://www.geeksforgeeks.org/sql-ddl-dql-dml-dcl-tcl-commands/>

[7] <https://www.slideshare.net/shashankkarnati/railway-management-system-database-mini-project>

[8] <https://www.w3schools.com/sql/func_mysql_connection_id.asp>

[9] <https://netbeans.org/downloads/8.2/rc/>

[10] <https://dev.mysql.com/downloads/installer/>

[11] <https://www.oracle.com/java/technologies/javase/jdk14-archive-downloads.html>

[12] <https://github.com/topics/toast-message>