5/30/2021

DBMS PROJECT REPORT

PROJECT TITLE: RAILWAY RESERVATION

SUBMITTED BY:

SHRUTHI G	ADITYA RAMACHANDRAN	VARSHA SSKM
2019103061	2019103502	2019103603

BATCH -P

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

COLLEGE OF ENGINEERING GUINDY

CONTENT

SNO	TOPIC	PAGE NUMBER
1	INTRODUCTION	2
2	FUNCTIONAL AND NON-FUNCTIONAL REQUIREMENTS	3
3	ER DIAGRAM	4
4	RELATIONAL SCHEMA	5
5	CONNECTING TO THE DATABASE	6
6	USER INTERFACE DESIGN	7
7	SETTING UP RELATIONS	13
8	SETTING UP VIEWS	17
9	SETTING UP TRIGGERS	18
10	SETTING UP PROCEDURES	18
11	IMPLEMENTATION	19
12	RESULTS SNAPSHOTS	31
13	CONCLUSION	53
14	REFERENCES	53

INTRODUCTION

The project “Railway Reservation” is developed to act as an interface between the railways and the user. This application provides two modes, the Admin mode, which is used to add train details to the train database and User mode where the user can sign up to create new user to the user database, book tickets or cancel them if necessary. They will be generated fare from the system.

This application is built using:

The Front-End: HTML, CSS, JavaScript, Embedded JavaScript

The Back-End: Node JS

Database Server: MySQL Server 8.0

Database Tools: MySQL Shell 8.0.19

Functionalities of the admin:

- The admin can add trains to the train database after logging in using his username and password.
- He can also add expresses to the express relation.

Functionalities of the user:

- The user can sign up to create a new user or log in using the existing credentials in the user database.
- He can book tickets in the trains available by paying the fare generated after logging in.
- He can also cancel the existing tickets if necessary.

REQUIREMENTS:

FUNCTIONAL REQUIREMENTS:

The system maintains the data of:

- Train details and cost of seats
- Express details and available number of seats
- User details
- Tickets currently booked by the user
- Previous tickets booked by the user
- Billing details of each user while booking a ticket
- Cancellation of ticket by the user

NON-FUNCTIONAL REQUIREMENTS:

The system provides the following:

- Security:
  Hashing of Passwords: passwords are hashed in the user_details relation to promise the user of safety.

- User Interface:
  A friendly and intuitive user interface has been developed for better user experience.

- Reliability:
  Reliable costs are generated as per the required number of seats accurately.

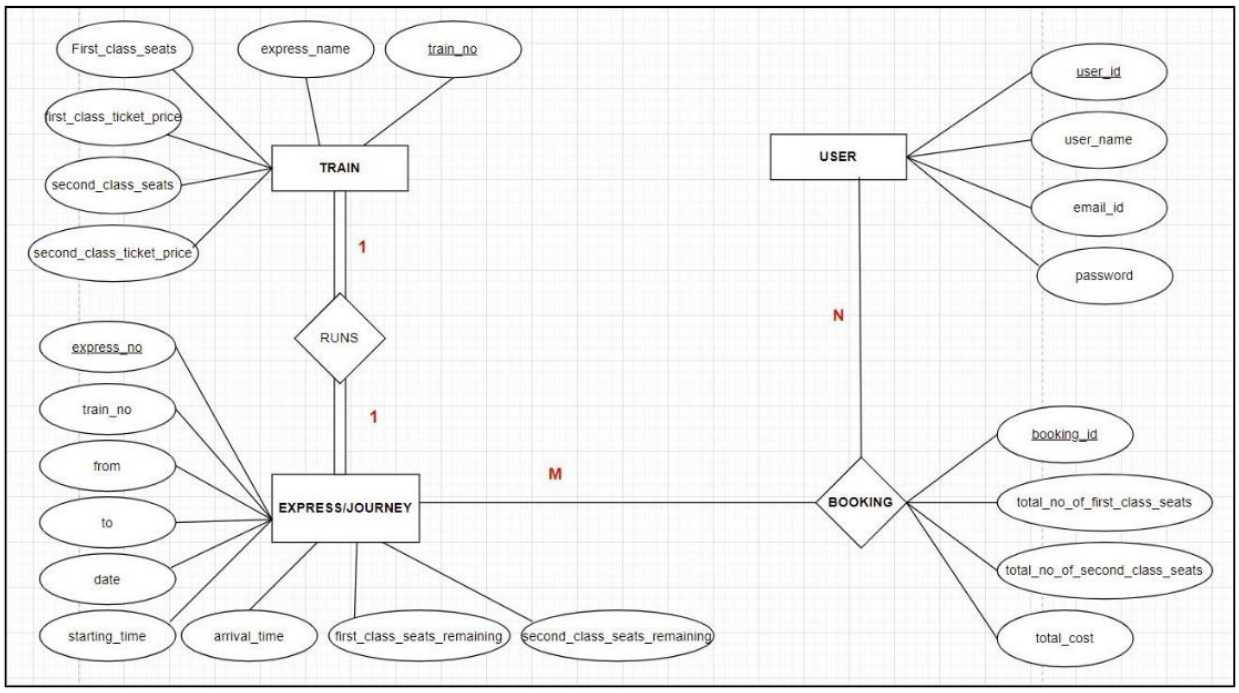
ER DIAGRAM OF THE DATABASE

This ER Model of database consists of 3 entities: **train, express and user.**

Train has attributes pertaining to a train which don't change like train_no, express_no, first_class_seats, first_class_ticket_price, second_class_seats, second_class_ticket_price.

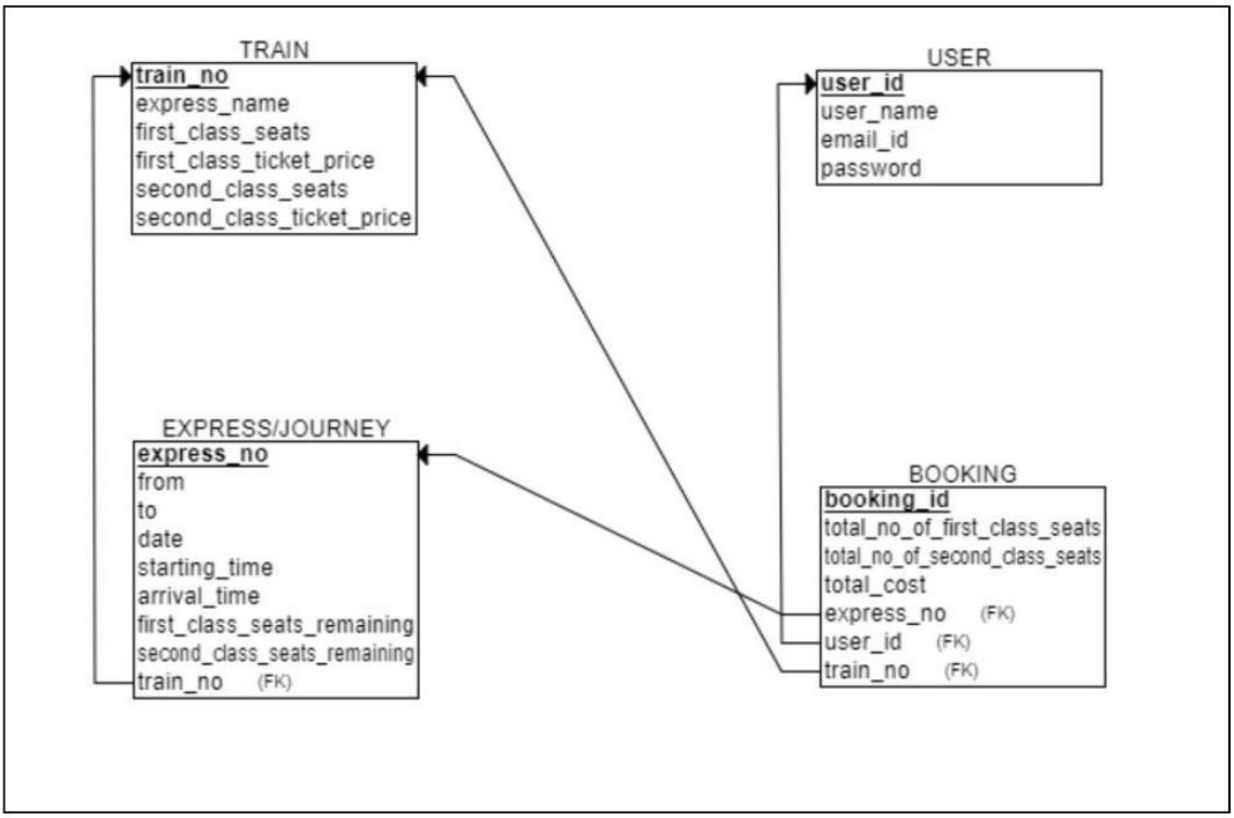
Express has attributes pertaining to a particular express like express_no, train_no, from, to, date, starting_time, arrival_time and some attributes which can be changed when tickets are booked and cancelled by the user like first_class_seats_remaining and second_class_seats_remaining.

User has attributes pertaining to a user like id, name, etc.



RELATIONAL SCHEMA OF THE DATABASE

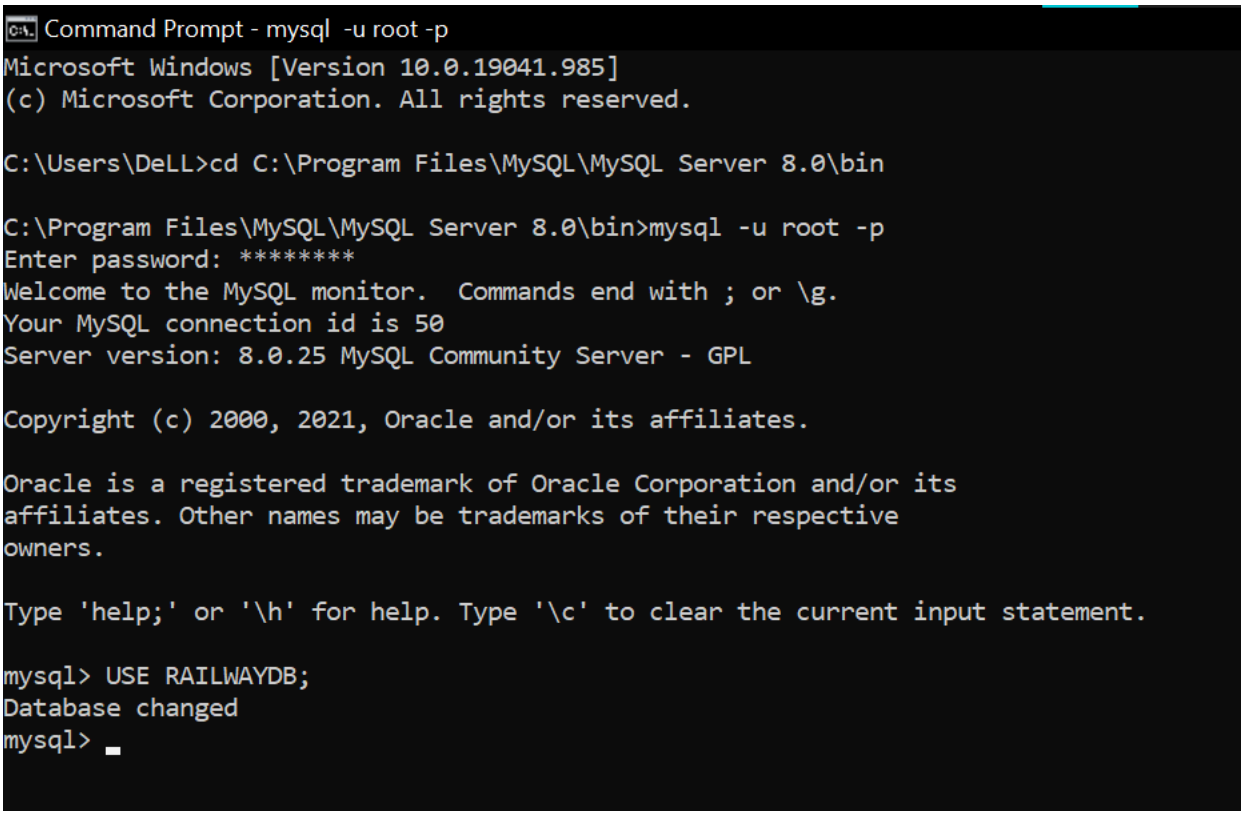
The relationship between train and express is one-one since a particular train no corresponds to an express and vice versa. The relationship between express and user is many-many since an express can be booked by many users and a single user can book tickets in many expresses. The relationship set(booking) between user and express consists of attributes like booking_id, total_no_of_first_class_seats, total_no_of_second_class_seats and total_cost



CONNECTING TO THE DATABASE

MySQL has been used for the implementation of the application.

A railwaydb database has been created using the MySQL shell which is accessed through windows command prompt.

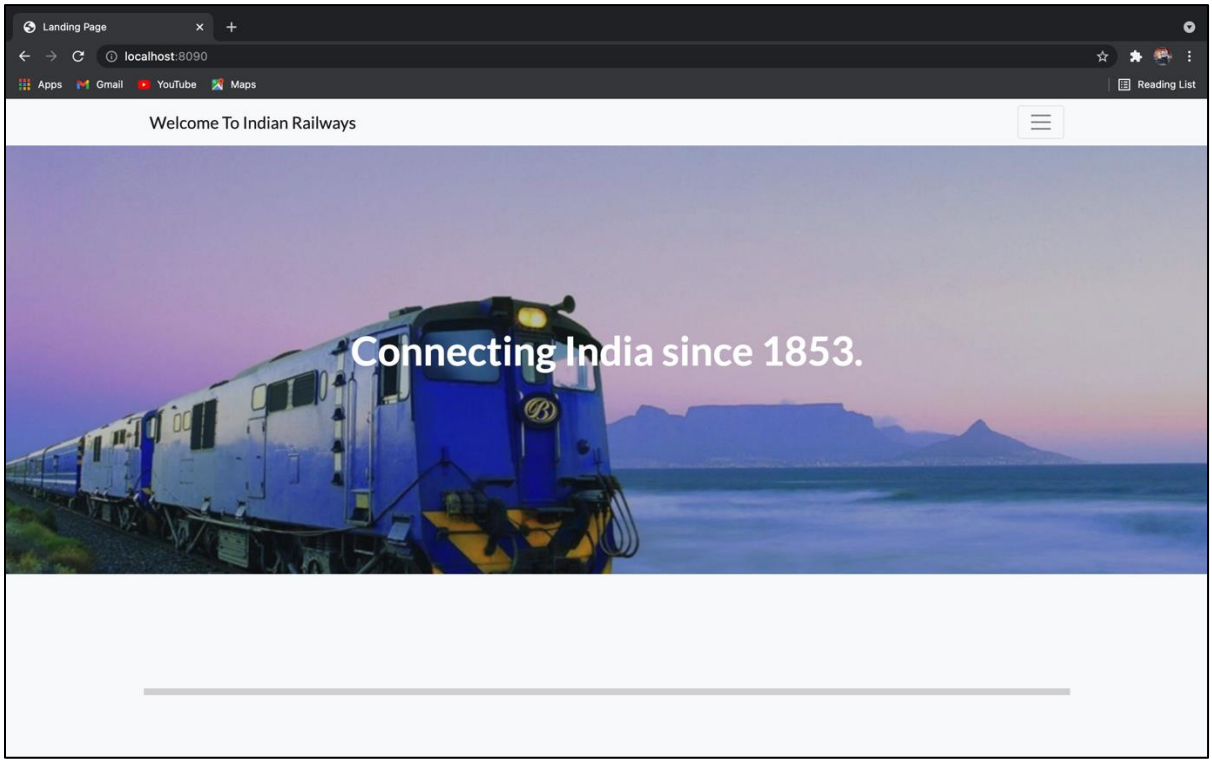


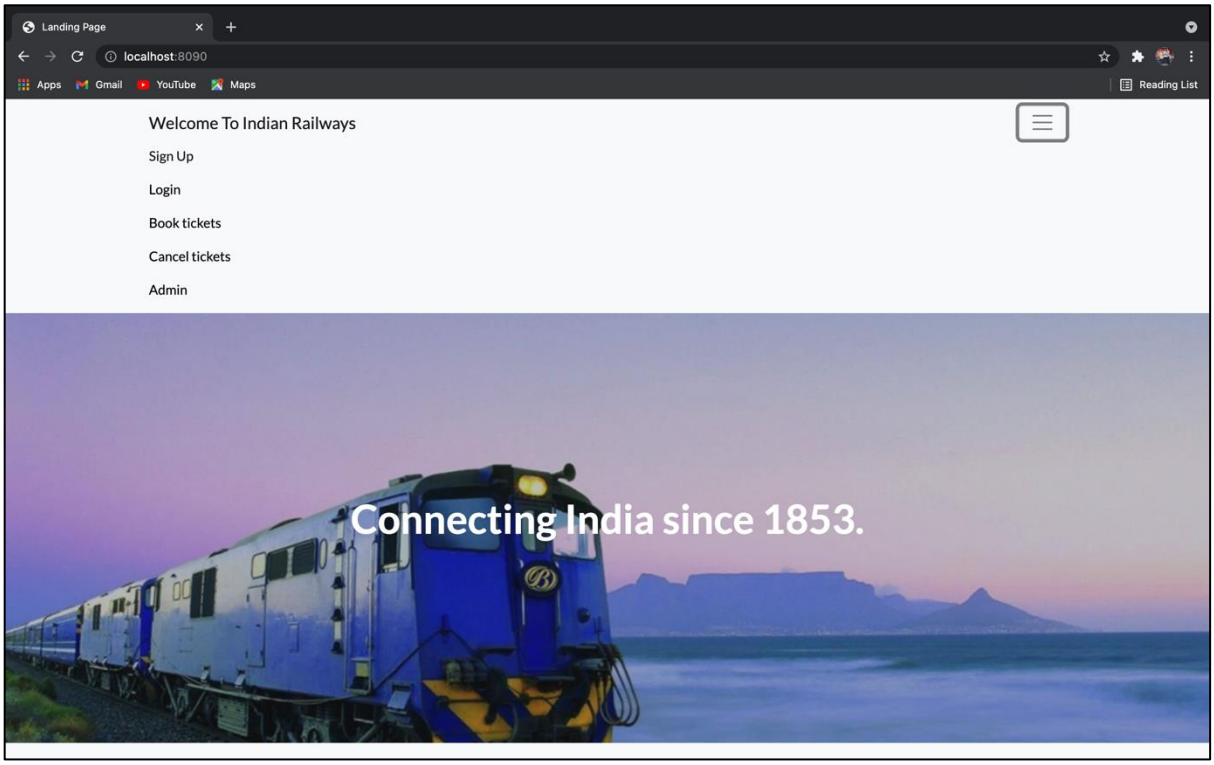
In NodeJS :

```
26  var host1 = 'localhost';
27  var user1 = 'root';
28  var pwd1 = 'gdrs9701';
29  var port1 = 3306;
30  const con = mysql.createConnection({
31      host: host1,
32      user: user1,
33      password: pwd1,
34      port: port1,
35      database: 'railwaydb',
36  });
```

USER INTERFACE DESIGN

Landing page

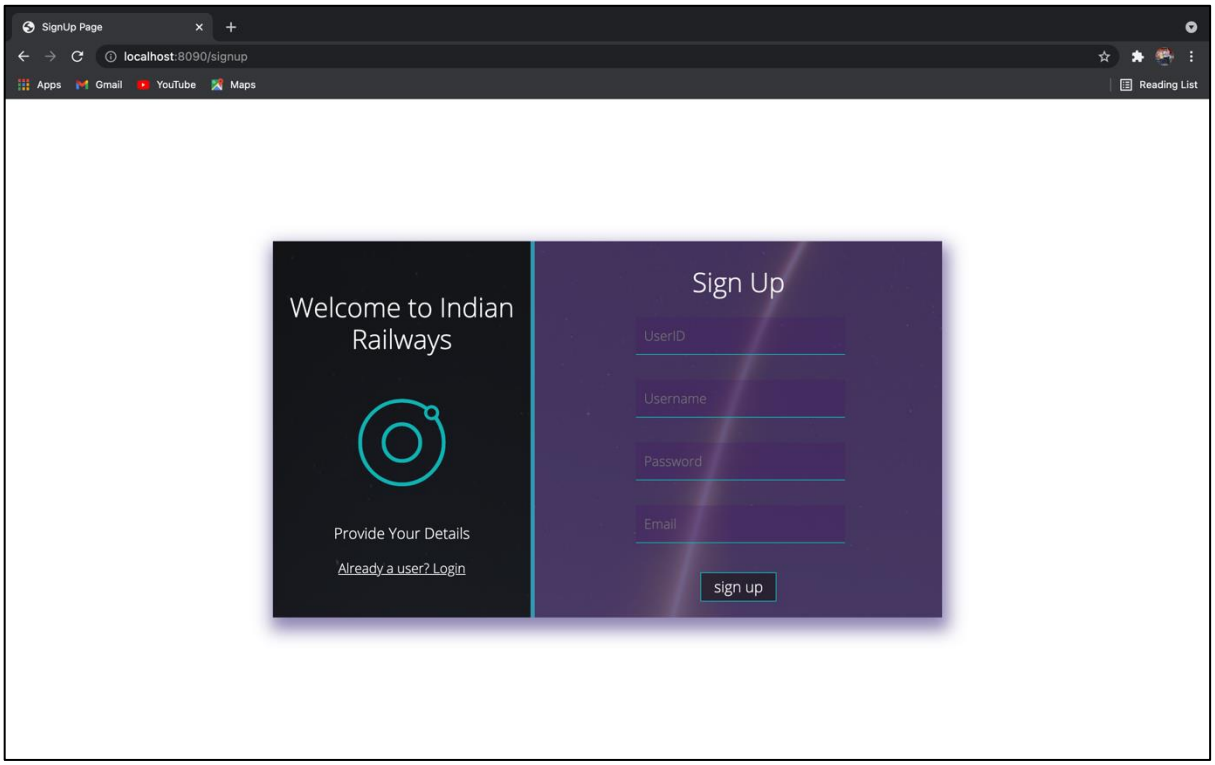




The following sections are available for the users of the application:

1. Sign Up:

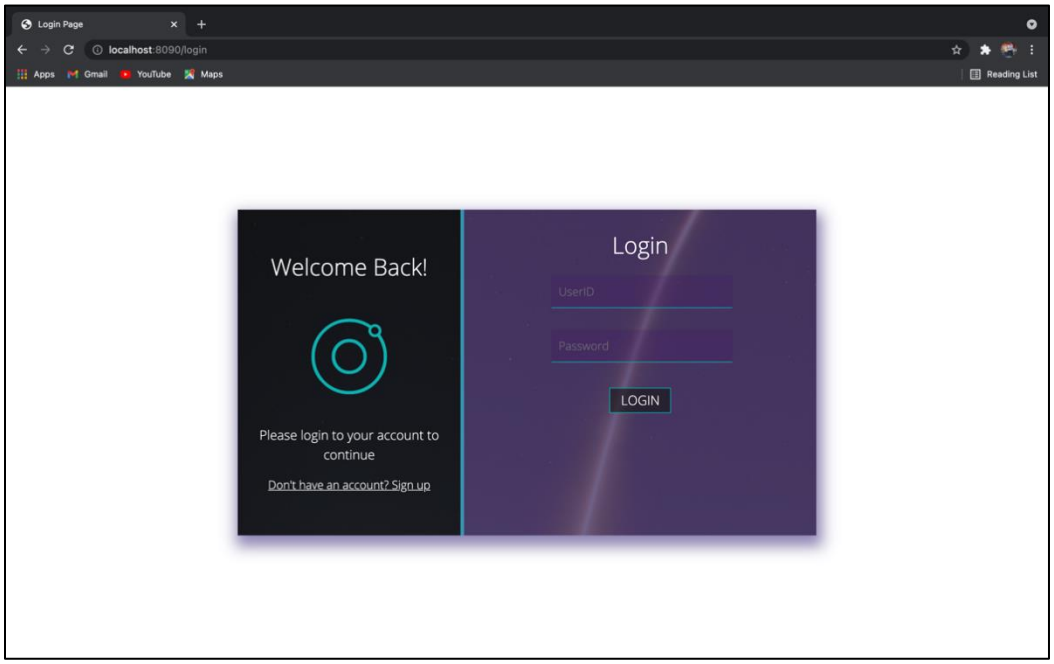
The area where the user can create a new user account to enjoy all the features of the application.



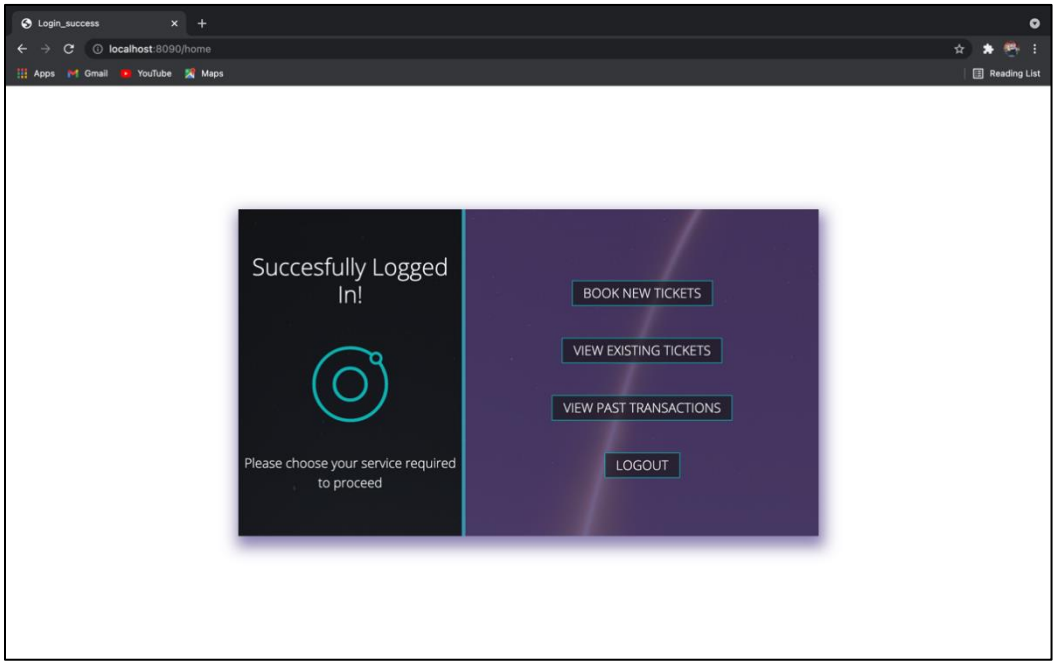
Here the user must enter UserID, Username, Password, and the Email address where the UserID must be unique. From here, he is redirected to the login page.

2. Login:

Here, the user must enter correct credentials to enter his account and use all the features of the website.

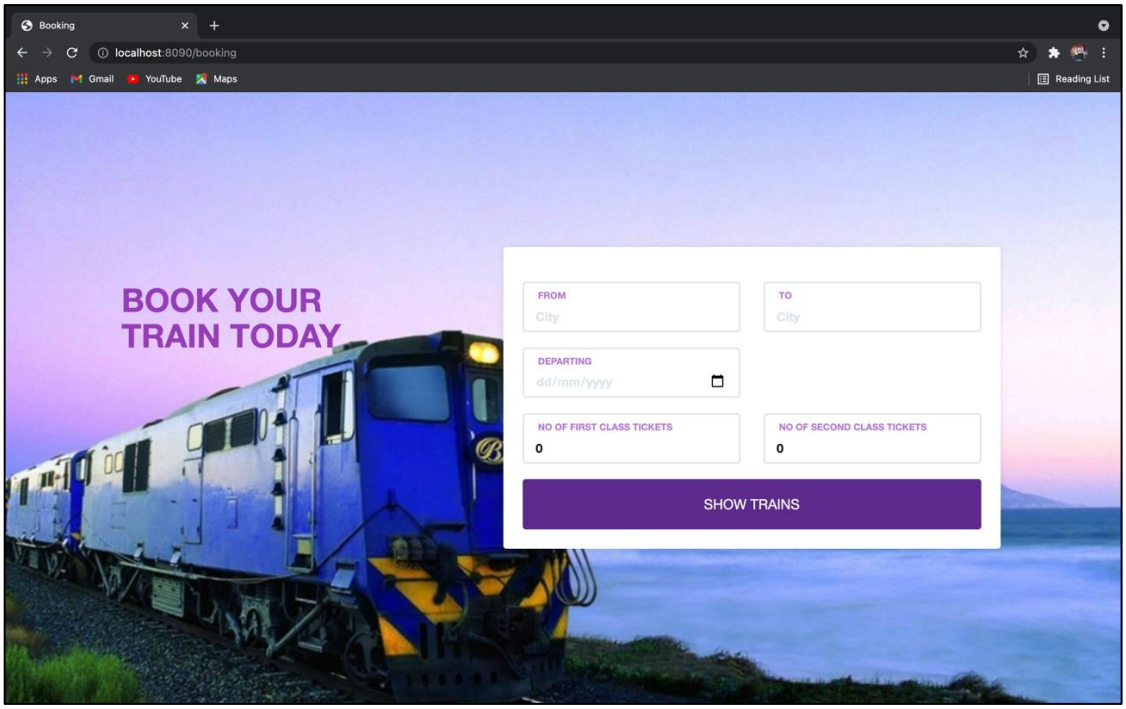


From here, the user will be redirected to the home page.



3. Book Tickets

Here the users can specify various details like From, To, Departing Date and number of tickets in each class and they will be shown trains based on availability of trains and seats on a particular day.



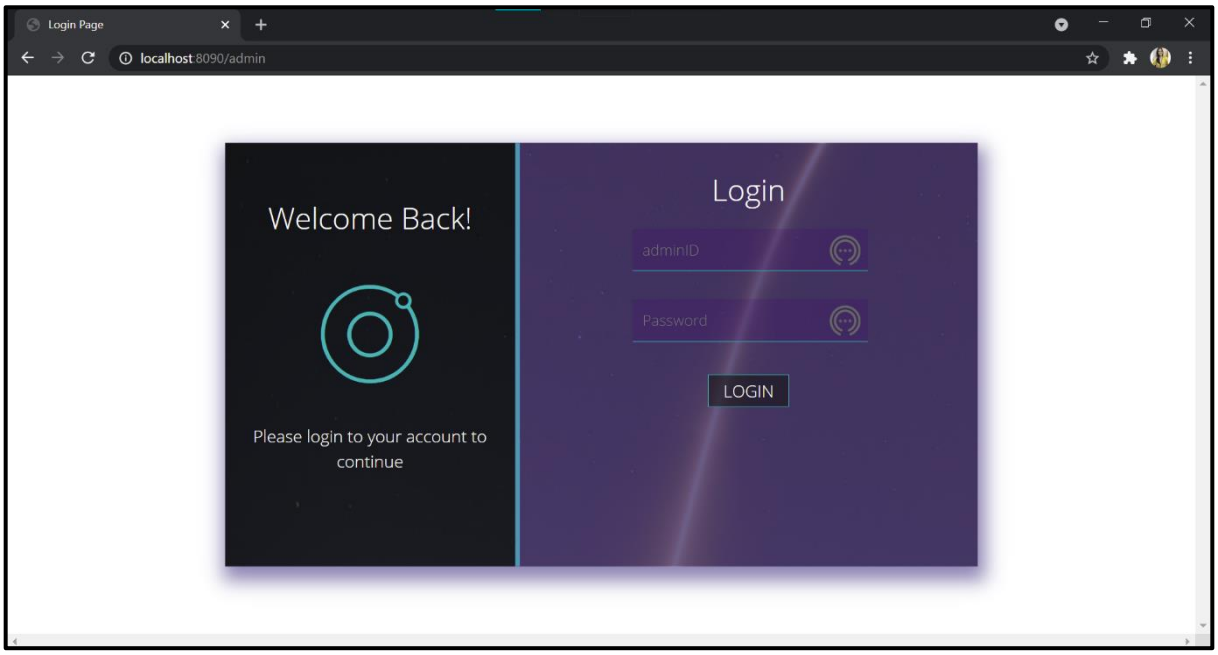
4. Cancel Tickets

Here, the user is redirected to “View Existing Tickets” section where the user can choose among the tickets already booked and choose to cancel them or return to the home page.(Shown in the implementation part)

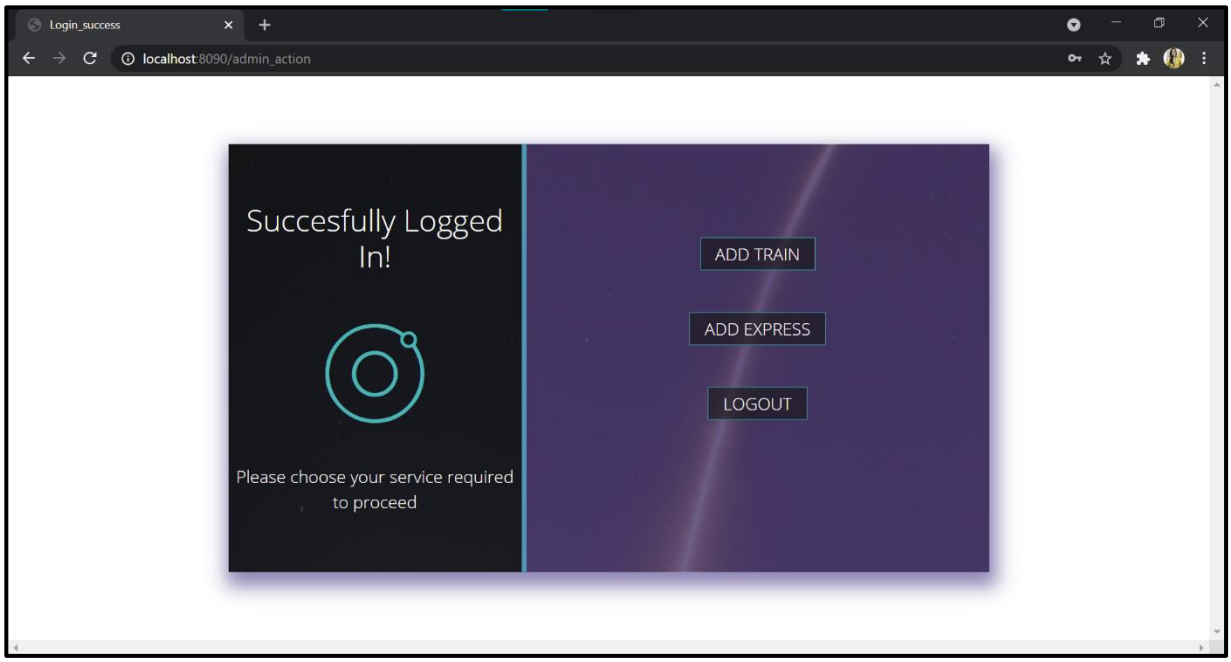
The following sections are available for the admin section of the application:

1. Login:

   Here the admin is provided with their adminID and password through which they login to add trains and expresses to the database.

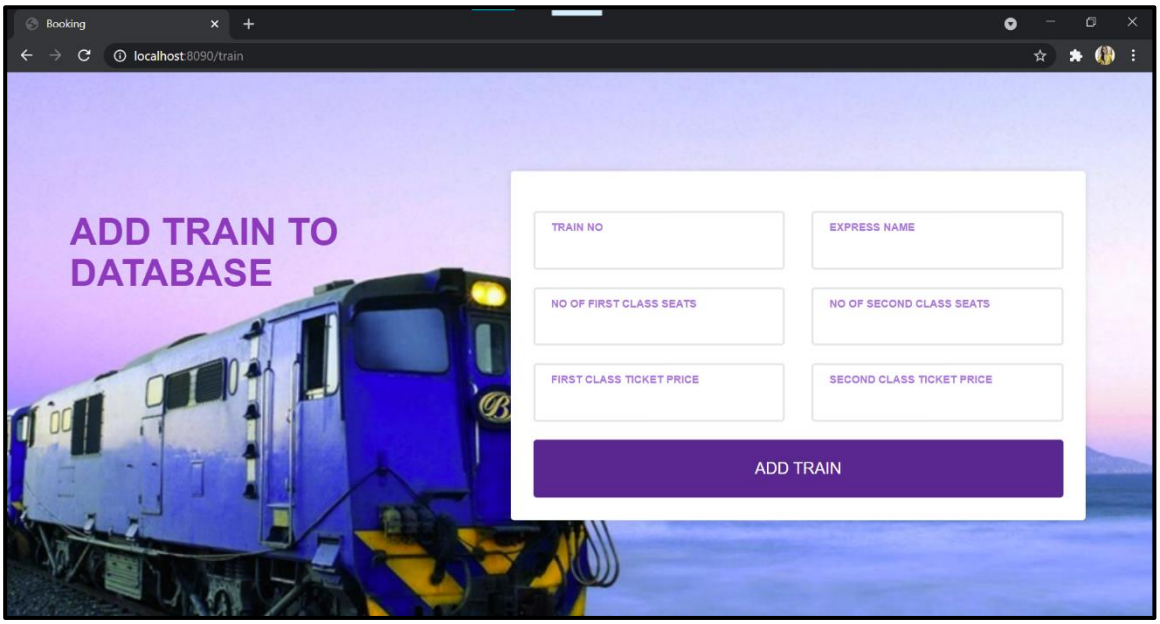


From here, the admin is directed to admin_actions page.



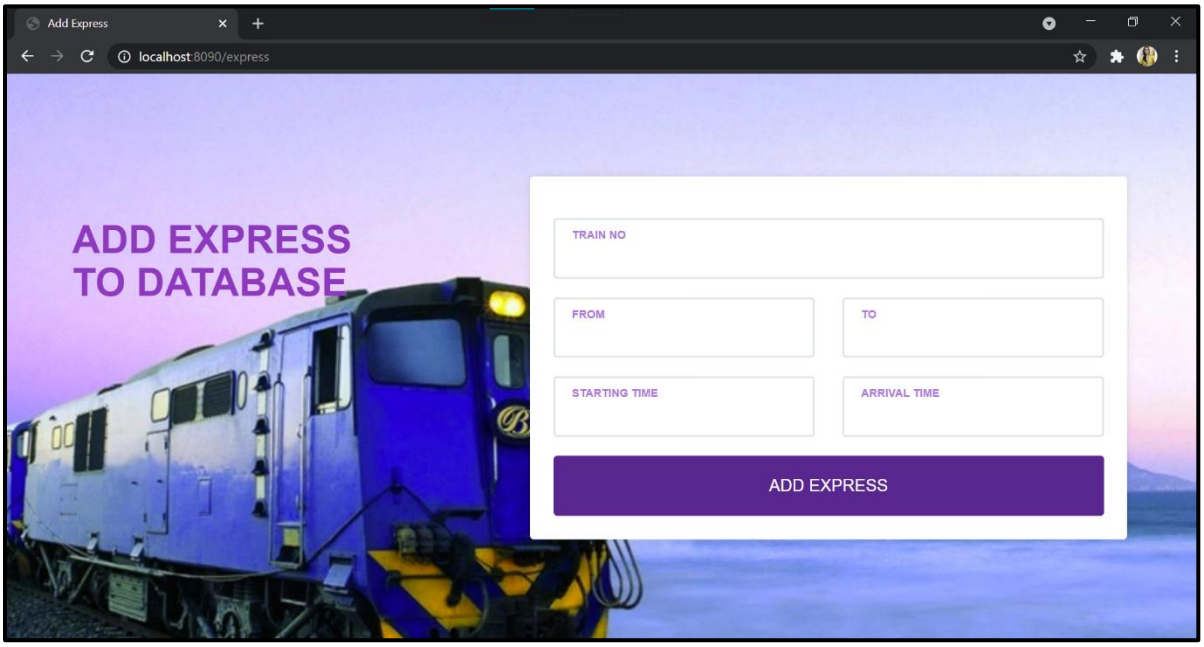
2. Add Trains:

Here the admin is allowed to add trains to the TRAIN relation in the database, hence facilitating the users to book their desired trains.



3. Add Express:

Here the admin is allowed to add expresses to the EXPRESS relation in the database, hence facilitating the users to book their desired express.



SETTING UP RELATIONS

Relations under the database “railwaydb”:

1) TRAIN:
   Attributes:
   - **Train_no** (Primary Key, Uniquely identifies a train)
   - Express_name
   - FC_seats
   - SC_seats
   - FC_price
   - SC_price

   TRAIN_NO -> EXPRESS_NAME, FC_SEATS, SC_SEATS, FC_PRICE, SC_PRICE

All attributes are non-transitively determined by the Primary Key(Super Key) TRAIN_NO. Hence, it is in Boyce-Codd Normal Form.

```
mysql> DESC TRAIN;
+--------------+-------------+------+-----+---------+-------+
| Field        | Type        | Null | Key | Default | Extra |
+--------------+-------------+------+-----+---------+-------+
| TRAIN_NO     | varchar(15) | NO   | PRI | NULL    |       |
| EXPRESS_NAME | varchar(20) | YES  |     | NULL    |       |
| FC_SEATS     | int         | YES  |     | NULL    |       |
| SC_SEATS     | int         | YES  |     | NULL    |       |
| FC_PRICE     | int         | YES  |     | NULL    |       |
| SC_PRICE     | int         | YES  |     | NULL    |       |
+--------------+-------------+------+-----+---------+-------+
6 rows in set (0.02 sec)
```

2) EXPRESS:
   Attributes:
   - **Express_no** (Primary Key, Uniquely identifies an express)
   - Train_no (Foreign Key, refers TRAIN.TRAIN_NO)
   - Fromm
   - Too
   - Starting_time
   - Arrival_time
   - FC_seats_remaining
   - SC_seats_remaining

EXPRESS_NO -> TRAIN_NO, FROMM, TOO, STARTING_TIME, ARRIVAL_TIME, FC_SEATS_REMAINING, SC_SEATS_REMAINING

All attributes are non-transitively determined by the Primary Key(Super Key) EXPRESS_NO. Hence, it is in Boyce-Codd Normal Form.

```
mysql> DESC EXPRESS;
+--------------------+-------------+------+-----+---------+----------------+
| Field              | Type        | Null | Key | Default | Extra          |
+--------------------+-------------+------+-----+---------+----------------+
| EXPRESS_NO         | int         | NO   | PRI | NULL    | auto_increment |
| TRAIN_NO           | varchar(15) | YES  | MUL | NULL    |                |
| FROMM              | varchar(20) | YES  |     | NULL    |                |
| TOO                | varchar(20) | YES  |     | NULL    |                |
| STARTING_TIME      | timestamp   | YES  |     | NULL    |                |
| ARRIVAL_TIME       | timestamp   | YES  |     | NULL    |                |
| FC_SEATS_REMAINING | int         | YES  |     | NULL    |                |
| SC_SEATS_REMAINING | int         | YES  |     | NULL    |                |
+--------------------+-------------+------+-----+---------+----------------+
8 rows in set (0.01 sec)
```

3) USER_DETAILS:

Attributes:

- **User_ID** (Primary Key, Uniquely identifies a user)
- Username
- Email_ID
- Password

USER_ID -> USERNAME, EMAIL_ID, PASSWORD

All attributes are non-transitively determined by the Primary Key(Super Key) USER_ID. Hence, it is in Boyce-Codd Normal Form.

```
mysql> DESC USER_DETAILS;
+----------+--------------+------+-----+---------+-------+
| Field    | Type         | Null | Key | Default | Extra |
+----------+--------------+------+-----+---------+-------+
| USER_ID  | varchar(15)  | NO   | PRI | NULL    |       |
| USERNAME | varchar(20)  | YES  |     | NULL    |       |
| EMAIL_ID | varchar(25)  | YES  |     | NULL    |       |
| PASSWORD | varchar(200) | YES  |     | NULL    |       |
+----------+--------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

4) BOOKING:

Attributes:

- **Booking_ID** (Primary Key, Uniquely identifies a booking)
- Express_no (Foreign Key, refers EXPRESS.EXPRESS_NO)
- User_ID (Foreign Key, refers USER_DETAILS.USER_ID)
- Train_no (Foreign Key, refers TRAIN.TRAIN_NO)
- Total_FC_seats
- Total_SC_seats
- Total_cost

BOOKING_ID -> EXPRESS_NO, USER_ID, TRAIN_NO, TOTAL_FC_SEATS, TOTAL_SC_SEATS, TOTAL_COST

All attributes are non-transitively determined by the Primary Key(Super Key) BOOKING_ID. Hence, it is in Boyce-Codd Normal Form.

```
mysql> DESC BOOKING;
+----------------+-------------+------+-----+---------+----------------+
| Field          | Type        | Null | Key | Default | Extra          |
+----------------+-------------+------+-----+---------+----------------+
| BOOKING_ID     | int         | NO   | PRI | NULL    | auto_increment |
| EXPRESS_NO     | int         | YES  | MUL | NULL    |                |
| USER_ID        | varchar(20) | YES  | MUL | NULL    |                |
| TRAIN_NO       | varchar(15) | YES  | MUL | NULL    |                |
| TOTAL_FC_SEATS | int         | YES  |     | NULL    |                |
| TOTAL_SC_SEATS | int         | YES  |     | NULL    |                |
| TOTAL_COST     | int         | YES  |     | NULL    |                |
+----------------+-------------+------+-----+---------+----------------+
7 rows in set (0.00 sec)
```

ALL THE RELATIONS UNDER THE DATABASE ARE BOYCE-CODD NORMALIZED.

SQL FILE TO CREATE RELATIONS

SQL > Create_tables.sql

```sql
CREATE DATABASE railwaydb;
USE railwaydb;

CREATE TABLE TRAIN
(TRAIN_NO VARCHAR(15) PRIMARY KEY,
EXPRESS_NAME VARCHAR(20),
FC_SEATS INT,
SC_SEATS INT,
FC_PRICE INT,
SC_PRICE INT);

CREATE TABLE EXPRESS
(EXPRESS_NO INT NOT NULL AUTO_INCREMENT PRIMARY KEY ,
TRAIN_NO VARCHAR(15),
FROMM VARCHAR(20),
TOO VARCHAR(20),
STARTING_TIME TIMESTAMP,
ARRIVAL_TIME TIMESTAMP,
FC_SEATS_REMAINING INT,
SC_SEATS_REMAINING INT,
FOREIGN KEY (TRAIN_NO) REFERENCES TRAIN(TRAIN_NO)
);
ALTER TABLE EXPRESS AUTO_INCREMENT = 10001;

CREATE TABLE USER_DETAILS
(USER_ID VARCHAR(15) PRIMARY KEY,
USERNAME VARCHAR(20),
EMAIL_ID VARCHAR(25),
PASSWORD VARCHAR(200)
);
```

Running the file on MySQL:

```
mysql> source C:\Users\DeLL\Desktop\DBMS PROJECT\SQL files\Create_tables.sql
Query OK, 1 row affected (0.01 sec)

Database changed
Query OK, 0 rows affected (0.02 sec)

Query OK, 0 rows affected (0.03 sec)

Query OK, 0 rows affected (0.01 sec)
Records: 0  Duplicates: 0  Warnings: 0

Query OK, 0 rows affected (0.02 sec)

Query OK, 0 rows affected (0.04 sec)
```

SETTING UP VIEWS

The database contains two view:

1. Expressview: a view created from Express relation to filter trains as per user requirements.
2. Presentview: a view created as the booking table gets populated, for the cancellation of tickets.

EXPRESSVIEW:

SQL > express_view.sql

```sql
CREATE OR REPLACE VIEW EXPRESSVIEW AS SELECT EXPRESS_NO, TRAIN_NO, FROMM, TOO, DATE(STARTING_TIME) AS
DEPARTURE_DATE, TIME(STARTING_TIME) AS DEPARTURE_TIME FROM EXPRESS;
```

PRESENTVIEW:

SQL > existingView.sql

```sql
CREATE OR REPLACE VIEW PRESENTVIEW AS SELECT BOOKING.BOOKING_ID AS BOOKING_ID, BOOKING.EXPRESS_NO AS EXPRESS_NO,
BOOKING.USER_ID AS USER_ID, BOOKING.TRAIN_NO AS TRAIN_NO,DATE(EXPRESS.STARTING_TIME) AS DEPARTURE_DATE,
TIME(EXPRESS.STARTING_TIME) AS DEPARTURE_TIME, BOOKING.TOTAL_FC_SEATS AS TOTAL_FC_SEATS,
BOOKING.TOTAL_SC_SEATS AS TOTAL_SC_SEATS,
BOOKING.TOTAL_COST AS TOTAL_COST FROM BOOKING INNER JOIN EXPRESS ON (BOOKING.EXPRESS_NO=EXPRESS.EXPRESS_NO);
```

RUNNING FILES ON SQL

```
mysql> source C:\Users\DeLL\Desktop\DBMS PROJECT\SQL files\express_view.sql
Query OK, 0 rows affected (0.01 sec)

mysql> source C:\Users\DeLL\Desktop\DBMS PROJECT\SQL files\existingView.sql
Query OK, 0 rows affected (0.01 sec)
```

```
mysql> use railwaydb;
Database changed
mysql> show tables;
+---------------------+
| Tables_in_railwaydb |
+---------------------+
| booking             |
| express             |
| expressview         |
| presentview         |
| train               |
| user_details        |
+---------------------+
6 rows in set (0.01 sec)
```

SETTING UP TRIGGERS

1. A bookingtrigger has been created to reduce the number of available seats upon booking of tickets by the user.
2. A canceltrigger has been used to reset the number of available seats upon cancellation of ticket by the user.

SQL > expresstrigger.sql

```sql
DELIMITER $$
CREATE TRIGGER bookingtrigger
AFTER INSERT ON BOOKING FOR EACH ROW
BEGIN
  UPDATE EXPRESS SET FC_SEATS_REMAINING = FC_SEATS_REMAINING - NEW.TOTAL_FC_SEATS WHERE EXPRESS_NO = NEW.EXPRESS_NO;
  UPDATE EXPRESS SET SC_SEATS_REMAINING = SC_SEATS_REMAINING - NEW.TOTAL_SC_SEATS WHERE EXPRESS_NO = NEW.EXPRESS_NO;
END$$
DELIMITER ;

DELIMITER $$
CREATE TRIGGER canceltrigger
AFTER DELETE ON BOOKING FOR EACH ROW
BEGIN
  UPDATE EXPRESS SET FC_SEATS_REMAINING = FC_SEATS_REMAINING + OLD.TOTAL_FC_SEATS WHERE EXPRESS_NO = OLD.EXPRESS_NO;
  UPDATE EXPRESS SET SC_SEATS_REMAINING = SC_SEATS_REMAINING + OLD.TOTAL_SC_SEATS WHERE EXPRESS_NO = OLD.EXPRESS_NO;
END$$
DELIMITER ;
```

SETTING UP PROCEDURES

1. ADDTRAIN procedure is written to insert values into the EXPRESS relation from the admin side of the application.
2. ADDUSER procedure is written to insert values into the USER_DETAILS relation upon user sign up.

SQL > proc_addtrain.sql

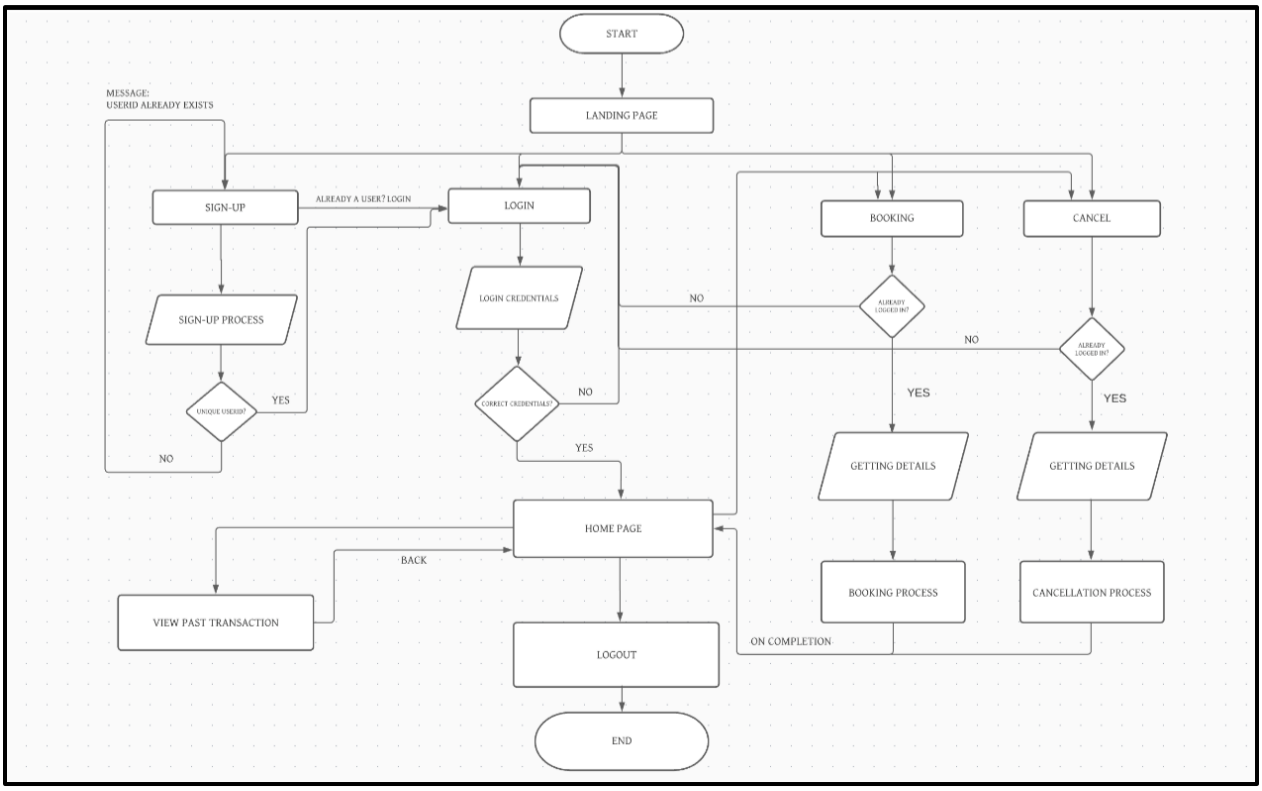
```sql
DELIMITER $$
CREATE PROCEDURE ADDTRAIN (IN TNO VARCHAR(15), IN ENO VARCHAR(20), IN FCS INT, IN SCS INT, IN FCP INT, IN SCP INT)
BEGIN
  INSERT INTO TRAIN(TRAIN_NO, EXPRESS_NAME, FC_SEATS, SC_SEATS, FC_PRICE, SC_PRICE)
  VALUES(TNO, ENO, FCS, SCS, FCP, SCP);
END $$
DELIMITER;
```

SQL > proc_adduser.sql

```sql
DELIMITER $$
CREATE PROCEDURE ADDUSER (IN UID VARCHAR(15), IN UNAME VARCHAR(20), IN EID VARCHAR(25), IN PWD VARCHAR(200))
BEGIN
  INSERT INTO USER_DETAILS(USER_ID, USERNAME, EMAIL_ID, PASSWORD) VALUES(UID, UNAME, EID, PWD);
END $$
DELIMITER;
```

IMPLEMENTATION

General flowchart depicting the flow of control inside the program.



The server is started in Visual Studio Code as:

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE
C:\Program Files\nodejs\node.exe .\server.js
Server started at http://localhost:8090
```

Now the application is running on the web browser as localhost:8090

ADMIN SIDE:

1. Adding trains to the database:

   The trains are initially added to the TRAIN relation using SQL queries and further addition of trains are done by the admin.

SQL > insert_train.sql

```
1   INSERT INTO TRAIN VALUES('E15698','Howrah',10,20,500,300);
2   INSERT INTO TRAIN VALUES('E16742','Falaknuma',15,25,450,300);
3   INSERT INTO TRAIN VALUES('E17431','Guwahati',20,30,400,200);
4   INSERT INTO TRAIN VALUES('E18156','Shalimar',10,20,550,350);
5   INSERT INTO TRAIN VALUES('E19473','East Coast',10,15,550,350);
6
7   INSERT INTO TRAIN VALUES('N21023','Rajdhani',11,20,500,200);
8   INSERT INTO TRAIN VALUES('N24563','Kongu',15,30,450,300);
9   INSERT INTO TRAIN VALUES('N25931','Nizamuddin',10,30,400,300);
10  INSERT INTO TRAIN VALUES('N26781','Karnataka',10,25,400,250);
11
12  INSERT INTO TRAIN VALUES('N36913','Ganga-Kaveri',15,30,400,250);
13
14  INSERT INTO TRAIN VALUES('S44592','Pandian',15,20,450,200);
15  INSERT INTO TRAIN VALUES('S45678','Vaigai',10,25,400,250);
16  INSERT INTO TRAIN VALUES('S46423','Tejas',10,30,400,300);
17  INSERT INTO TRAIN VALUES('S47365','Guruvayur',15,25,350,200);
18
19  INSERT INTO TRAIN VALUES('S53813','Mysuru',20,35,400,250);
20  INSERT INTO TRAIN VALUES('S54671','Shatabdi',10,15,450,250);
21  INSERT INTO TRAIN VALUES('S56892','Bagmati',15,35,400,200);
22  INSERT INTO TRAIN VALUES('S57893','Kaveri',20,30,500,300);
23
24  INSERT INTO TRAIN VALUES('W61962','Gorakhpur',15,25,400,200);
25  INSERT INTO TRAIN VALUES('W63014','Konkan',15,30,300,200);
26  INSERT INTO TRAIN VALUES('W64567','Ernakulam',10,25,400,250);
27  INSERT INTO TRAIN VALUES('W65681','Korba',10,20,450,300);
28
29  SELECT CONCAT (EXPRESS_NAME,' Express') FROM TRAIN;
```

Further addition of trains to the relation as per the needs of the user is done using the admin side of the application. The data entered in the webpage is extracted in the NodeJS file using html name tag. A procedure has been used to insert the

entered fields into the relation TRAINS. The code snippet for this action is as follows:

```
app.post('/addtrain', function (req, res) {
    if (req.session.loggedin) {
        con.connect(function (err) {
            var sql =
                "CALL ADDTRAIN('" +
                req.body.trainno +
                "','" +
                req.body.expressno +
                "','" +
                req.body.fcseats +
                "','" +
                req.body.scseats +
                "','" +
                req.body.fcprice +
                "','" +
                req.body.scprice +
                "')";
            con.query(sql, function (err, result) {
                if (err) {
                    console.error('Error inserting!\n');
                    throw err;
                    res.end();
                } else {
                    res.sendFile(path.join(__dirname, 'train_success.html'));
                }
            });
        });
    } else
        res.redirect('/admin');
});
```

The procedure ADDTRAIN is given as:

SQL > proc_addtrain.sql

```
DELIMITER $$
CREATE PROCEDURE ADDTRAIN (IN TNO VARCHAR(15), IN ENO VARCHAR(20), IN FCS INT, IN SCS INT, IN FCP INT, IN SCP INT)
BEGIN
  INSERT INTO TRAIN(TRAIN_NO, EXPRESS_NAME, FC_SEATS, SC_SEATS, FC_PRICE, SC_PRICE)
  VALUES(TNO, ENO, FCS, SCS, FCP, SCP);
END $$
```

2. Adding expresses to the database:

The EXPRESS relation is initially populated using SQL queries and further addition of expresses are done by the admin.

SQL > insert_express.sql

```sql
INSERT INTO EXPRESS(TRAIN_NO,FROMM,TOO,STARTING_TIME,ARRIVAL_TIME,FC_SEATS_REMAINING,SC_SEATS_REMAINING)
SELECT
'E15698','Kolkata','Hyderabad','2021-06-13 08:30:00','2021-06-14 05:29:00',
FC_SEATS,SC_SEATS FROM TRAIN WHERE TRAIN_NO = 'E15698';

INSERT INTO EXPRESS(TRAIN_NO,FROMM,TOO,STARTING_TIME,ARRIVAL_TIME,FC_SEATS_REMAINING,SC_SEATS_REMAINING)
SELECT
'E15698','Kolkata','Hyderabad','2021-06-15 10:30:00','2021-06-16 07:29:00',
FC_SEATS,SC_SEATS FROM TRAIN WHERE TRAIN_NO = 'E15698';

INSERT INTO EXPRESS(TRAIN_NO,FROMM,TOO,STARTING_TIME,ARRIVAL_TIME,FC_SEATS_REMAINING,SC_SEATS_REMAINING)
SELECT
'E15698','Kolkata','Hyderabad','2021-05-29 10:30:00','2021-05-29 07:29:00',
FC_SEATS,SC_SEATS FROM TRAIN WHERE TRAIN_NO = 'E15698';

INSERT INTO EXPRESS(TRAIN_NO,FROMM,TOO,STARTING_TIME,ARRIVAL_TIME,FC_SEATS_REMAINING,SC_SEATS_REMAINING)
SELECT
'E16742','Kolkata','Hyderabad','2021-06-13 20:30:00','2021-06-14 12:29:00',
FC_SEATS,SC_SEATS FROM TRAIN WHERE TRAIN_NO = 'E16742';

INSERT INTO EXPRESS(TRAIN_NO,FROMM,TOO,STARTING_TIME,ARRIVAL_TIME,FC_SEATS_REMAINING,SC_SEATS_REMAINING)
SELECT
'E16742','Kolkata','Hyderabad','2021-05-29 20:30:00','2021-05-30 12:29:00',
FC_SEATS,SC_SEATS FROM TRAIN WHERE TRAIN_NO = 'E16742';

INSERT INTO EXPRESS(TRAIN_NO,FROMM,TOO,STARTING_TIME,ARRIVAL_TIME,FC_SEATS_REMAINING,SC_SEATS_REMAINING)
SELECT
'E17431','Kolkata','Hyderabad','2021-06-13 10:50:00','2021-06-14 08:45:00',
FC_SEATS,SC_SEATS FROM TRAIN WHERE TRAIN_NO = 'E17431';
```

```sql
INSERT INTO EXPRESS(TRAIN_NO,FROMM,TOO,STARTING_TIME,ARRIVAL_TIME,FC_SEATS_REMAINING,SC_SEATS_REMAINING)
SELECT
'E17431','Kolkata','Hyderabad','2021-06-15 12:50:00','2021-06-16 10:45:00',
FC_SEATS,SC_SEATS FROM TRAIN WHERE TRAIN_NO = 'E17431';

INSERT INTO EXPRESS(TRAIN_NO,FROMM,TOO,STARTING_TIME,ARRIVAL_TIME,FC_SEATS_REMAINING,SC_SEATS_REMAINING)
SELECT
'E18156','Kolkata','Hyderabad','2021-06-13 02:30:00','2021-06-14 00:29:00',
FC_SEATS,SC_SEATS FROM TRAIN WHERE TRAIN_NO = 'E18156';

INSERT INTO EXPRESS(TRAIN_NO,FROMM,TOO,STARTING_TIME,ARRIVAL_TIME,FC_SEATS_REMAINING,SC_SEATS_REMAINING)
SELECT
'E18156','Kolkata','Hyderabad','2021-05-29 02:30:00','2021-05-30 00:29:00',
FC_SEATS,SC_SEATS FROM TRAIN WHERE TRAIN_NO = 'E18156';

INSERT INTO EXPRESS(TRAIN_NO,FROMM,TOO,STARTING_TIME,ARRIVAL_TIME,FC_SEATS_REMAINING,SC_SEATS_REMAINING)
SELECT
'E19473','Kolkata','Hyderabad','2021-06-13 13:00:01','2021-06-14 14:45:01',
FC_SEATS,SC_SEATS FROM TRAIN WHERE TRAIN_NO = 'E19473';
```

Further addition of expresses to the relation as per the needs of the user is done using the admin side of the application. The admin enters data in the HTML form in the webpage. The data entered in the webpage is extracted in the NodeJS file using html name tag. The field entered by the admin are added to the database with SQL queries. The code snippet for this action is as follows:

```
app.post('/addexpress', function (req, res) {
    if (req.session.loggedin) {
        con.connect(function (err) {
            var sql =
                "insert into EXPRESS(TRAIN_NO,FROMM,TOO,STARTING_TIME,ARRIVAL_TIME,FC_SEATS_REMAINING,SC_SEATS_REMAINING) SELECT '" +
                req.body.trainno +
                "','" +
                req.body.from +
                "','" +
                req.body.to +
                "','" +
                req.body.startingtime +
                "','" +
                req.body.arrivaltime +
                "'," +'FC_SEATS,SC_SEATS FROM TRAIN WHERE TRAIN_NO=' + "'" +
                req.body.trainno +
                "'";
            con.query(sql, function (err, result) {
                if (err) {
                    console.error('Error inserting!\n');
                    throw err;
                } else {
                    res.sendFile(path.join(__dirname, 'express_success.html'));
                }
            });
        });
    } else
        res.redirect('/admin');
});
```

The views, EXPRESSVIEW is created to filter the required trains by the user to display and PRESENTVIEW is created to facilitate cancellation of tickets.

USER SIDE:

1. Sign Up Page:

   In the sign-up page, user enters the account details through a HTML form where the data entered in the form is collected and sent through POST method to /insert where the data is checked for primary key violations. The data obtained is extracted in the server-side (Node JS file) using the name attribute of the html element. If no such violations are found, the data is added to the USER_DETAILS relation in the RAILWAYDB database using stored procedure ADDUSER after required encryptions. Else, an error message is displayed using errorins.html

```
app.post('/insert', function (req, res) {
    bcrypt.hash(req.body.pwd, saltRounds,(err,hash)=>{
        con.connect(function (err) {
            var sql = "CALL ADDUSER('" + req.body.uid + "','" + req.body.uname +
                "','" + req.body.emailid + "','" + hash +  "')";
            con.query(sql, function (err, result) {
                if (err) {
                    if (err.errno == 1062) {
                        res.sendFile(path.join(__dirname, 'errorins.html'));
                    } else {
                        console.error('Error inserting!\n');
                        throw err;
                        res.end();
                    }
                } else {
                    res.sendFile(path.join(__dirname, 'regsuccess.html'));
                }
            });
        });
    });
});
```

```
DELIMITER $$
CREATE PROCEDURE ADDUSER (IN UID VARCHAR(15), IN UNAME VARCHAR(20), IN EID VARCHAR(25), IN PWD VARCHAR(200))
BEGIN
  INSERT INTO USER_DETAILS(USER_ID, USERNAME, EMAIL_ID, PASSWORD) VALUES(UID, UNAME, EID, PWD);
END $$
DELIMITER;
```

2. Login Page:

In the login page, the user enters appropriate UserID and Password in HTML form where the data entered is collected using the POST method and is authenticated. If the entered data is found in the database, the user is redirected to the home page of the Application (/home) where the user can book tickets, cancel tickets and view past successful transactions. As soon as the data is verified true, a session is initiated, and various session variables are initialized. If data is not found in the database, an error page is loaded.

```
app.post('/auth', function (request, response) {
    var username = request.body.uid;
    var password = request.body.pwd;
    con.query('SELECT * FROM USER_DETAILS WHERE USER_ID = ? ', [username], function (error, results, fields) {
        if (results.length !== 0) {
            bcrypt.compare(password,results[0]['PASSWORD'], (error,result1) =>{
                if(result1 === true){
                    request.session.loggedin = true;
                    request.session.username = username;
                    response.redirect('/home');
                }
                else {

                    response.sendFile(path.join(__dirname, 'Login_error.html'));
                }
            });
        }
        else response.sendFile(path.join(__dirname, 'Login_error.html'));

    });
});
```

3. Home Page:

The home page has various links to various sections like book tickets, view existing tickets, view past transactions and logout.

```
    <ul class="noBullet">
    <br><br>
    <li id="center-btn">
        <input type="button" id="join-btn" name="join" alt="booking"  onclick="location.href='/booking'" value="BOOK NEW TICKETS">
      </li><br>
      <li id="center-btn">
        <input type="button" id="join-btn" name="join" alt="Login" onclick="location.href='/existing'"  value="VIEW EXISTING TICKETS">
      </li><br>
      <li id="center-btn">
        <input type="button" id="join-btn" name="join" alt="booking"  onclick="location.href='/prev'" value="VIEW PAST TRANSACTIONS">
      </li><br>
      <li id="center-btn">
        <input type="button" id="join-btn" name="join" alt="booking"  onclick="location.href='/logout'" value="LOGOUT">
      </li>
    </ul>
```

The server redirects to various location URLs as per the link chosen by the user.

4. Booking:

In the booking page, user enters the booking details like from, to, date of departure, number of seats through a HTML form where the data entered in the form is collected and sent through POST method to /showtable where trains are searched based on requirements in mysql RAILWAYDB database. The result-set is passed to an EJS file. If no such trains are found, message is displayed using showtrains_error.html. It is also checked whether at least one ticket is purchased totally.

```
app.post('/showtable', function (request, response) {
    if (request.session.loggedin) {
        data = [request.body.from, request.body.to, request.body.date, request.body.fc, request.body.sc];
        var sqlquery =
            'select EXPRESSVIEW.EXPRESS_NO,EXPRESSVIEW.TRAIN_NO,TRAIN.EXPRESS_NAME,EXPRESSVIEW.FROMM,' +
            "EXPRESSVIEW.TOO, DATE_FORMAT(EXPRESSVIEW.DEPARTURE_DATE,'%d-%b-%Y') as DEPARTURE_DATE,"+
            ' DEPARTURE_TIME from EXPRESSVIEW,TRAIN where " +
            'EXPRESSVIEW.TRAIN_NO=TRAIN.TRAIN_NO and FROMM=? and TOO=? ' +
            'and DATE(?)=DATE(DEPARTURE_DATE)' + ' and current_date() <= DATE(DEPARTURE_DATE)' +
            'and FC_SEATS > ?' +
            'and SC_SEATS> ?';
        con.query(sqlquery, data, function (error, results, fields) {
            if (error) throw error;
            if (results.length <= 0) response.sendFile(path.join(__dirname, 'showtrains_error.html'));
            else if (data[3] == 0 && data[4] == 0) response.sendFile(path.join(__dirname, 'invalid_error.html'));
            else response.render('showtrain.ejs', { result: results });
        });
    } else {
        response.redirect('/login');
    }
});
```

In showtrain.ejs, the result is displayed in the form of a table where the express no. is displayed as a button inside a table cell. As soon as any button is pressed, the value of textbox is filled using the below JavaScript code.

```
<script type="text/javascript">
function select(x){
  document.getElementById("selection").value = x ;
}
</script>
```

```
<div align="center">
  <form class="bookingform" action="/confirm" method="post">
    <input type="text" style="text-align: center;" name="selection" id="selection" value="" placeholder="Express_no"
    required onkeydown="return false;">
    <input type="submit" class="smallbut" name="" value="Book tickets">
  </form>
</div>
```

As soon as form gets submitted, the booking process takes place. As soon as payment is completed, the booking details are added to BOOKING relation in the RAILWAYDB and changes are introduced in the EXPRESS relation by the deployment of triggers.

server.js

```
app.post('/topay', function (request, response) {
    if (request.session.loggedin) {
        var password = request.body.pwd;
        con.query('SELECT * FROM USER_DETAILS WHERE USER_ID = ? ', [request.session.username], function (error, results, fields) {
            if (results.length !== 0) {
                bcrypt.compare(password, results[0]['PASSWORD'], (error, result1) => {
                    if (result1 === true) {
                        var sql ='insert into BOOKING(EXPRESS_NO,USER_ID,TRAIN_NO,TOTAL_FC_SEATS,TOTAL_SC_SEATS,TOTAL_COST) values(' +
                            "'" + final[0] + "'" + ',' + "'" + request.session.username + "'" + ',' +
                            "'" + final[1] + "'" + ',' + final[4] + ',' + final[5] + ',' + final[6] + ')';
                        con.query(sql, function (err, result) {
                            if (err) {
                                console.error('Error inserting!\n');
                                throw err;
                            }
                            else {
                                var updateview = 'CREATE OR REPLACE VIEW EXPRESSVIEW AS SELECT EXPRESS_NO, TRAIN_NO, FROMM,' +
                                    ' TOO, DATE(STARTING_TIME) AS DEPARTURE_DATE, TIME(STARTING_TIME) AS DEPARTURE_TIME FROM EXPRESS';
                                con.query(updateview);
                                response.sendFile(path.join(__dirname, 'paysuccess.html'));
                            }
                        });
                    }
                    else {
                        response.sendFile(path.join(__dirname, 'payerror.html'));
                    }
                });
            } else response.sendFile(path.join(__dirname, 'payerror.html'));
        });
    } else {
        response.redirect('/login');
    }
});
```

```
DELIMITER $$
CREATE TRIGGER bookingtrigger
AFTER INSERT ON BOOKING FOR EACH ROW
BEGIN
  UPDATE EXPRESS SET FC_SEATS_REMAINING = FC_SEATS_REMAINING - NEW.TOTAL_FC_SEATS WHERE EXPRESS_NO = NEW.EXPRESS_NO;
  UPDATE EXPRESS SET SC_SEATS_REMAINING = SC_SEATS_REMAINING - NEW.TOTAL_SC_SEATS WHERE EXPRESS_NO = NEW.EXPRESS_NO;
END$$
DELIMITER ;
```

5. Cancel Tickets:

When the user clicks “View Existing Tickets”, all current tickets corresponding to the given User-ID are searched in the mysql database. The result-set is passed to an EJS file. If no such trains are found, message is displayed using no_existing.html.

```
app.get('/existing', function (request, response) {
    if (request.session.loggedin) {
        var quer =
            'select PRESENTVIEW.BOOKING_ID,PRESENTVIEW.EXPRESS_NO,PRESENTVIEW.TRAIN_NO,'+
            'DATE_FORMAT(PRESENTVIEW.DEPARTURE_DATE,"%d-%b-%Y") as DEPARTURE_DATE,PRESENTVIEW.DEPARTURE_TIME,'+
            'PRESENTVIEW.TOTAL_FC_SEATS,PRESENTVIEW.TOTAL_SC_SEATS,PRESENTVIEW.TOTAL_COST '+
            'from EXPRESS,PRESENTVIEW where EXPRESS.EXPRESS_NO=PRESENTVIEW.EXPRESS_NO and ' +
            'PRESENTVIEW.USER_ID = ' +
            "'" +
            request.session.username +
            "'" +
            ' and current_timestamp() < EXPRESS.STARTING_TIME';
        con.query(quer, function (error, results, fields) {
            if (error) throw error;
            if (results.length <= 0) response.sendFile(path.join(__dirname, 'no_existing.html'));
            else response.render('viewpresent.ejs', { result: results });
        });
    } else {
        response.redirect('/login');
    }
});
```

In viewpresent.ejs, the result is displayed in the form of a table where the booking ID is displayed as a button inside a table cell. As soon as any button is pressed, the value of textbox is filled using the below JavaScript code.

```
<script type="text/javascript">
function select(x){
  document.getElementById("selection").value = x ;
}
</script>
```

```
    <div align="center">
      <form class="bookingform" action="/cancel" method="post">
        <input type="text" style="text-align: center;" name="selection" id="selection" value="" placeholder="Ticket no."
        required onkeydown="return false;">
        <input type="submit" class="smallbut" name="" value="Cancel ticket">
      </form>
    </div>
    <div align="center">
        <form class="bookingform" action="/home" method="get">
          <input type="submit" class="smallbut" name="" value="Go back">
        </form>
    </div>
```

As soon as form gets submitted, the cancelling process takes place. As soon as cancelling is confirmed, the booking details are deleted from the BOOKING relation in the RAILWAYDB and changes are introduced in the EXPRESS relation by the deployment of triggers.

```
app.post('/tocancel', function (request, response) {
    if (request.session.loggedin) {
        var password = request.body.pwd;
        con.query('SELECT * FROM USER_DETAILS WHERE USER_ID = ? ', [request.session.username], function (error, results, fields) {
            if (results.length !== 0) {
                bcrypt.compare(password, results[0]['PASSWORD'], (error, result1) => {
                    if (result1 === true) {
                        var sql = 'delete from BOOKING where BOOKING_ID = ' + "'" + ticketno + "'";
                        con.query(sql, function (err, result) {
                            if (err) {
                                console.error('Error inserting!\n');
                                throw err;
                            } else {
                                response.sendFile(path.join(__dirname, 'cancelsuccess.html'));
                            }
                        });
                    } else {
                        response.sendFile(path.join(__dirname, 'cancelerror.html'));
                    }
                });
            } else response.sendFile(path.join(__dirname, 'cancelerror.html'));
        });
    } else {
        response.redirect('/login');
    }
});

```

```
DELIMITER $$
CREATE TRIGGER canceltrigger
AFTER DELETE ON BOOKING FOR EACH ROW
BEGIN
  UPDATE EXPRESS SET FC_SEATS_REMAINING = FC_SEATS_REMAINING + OLD.TOTAL_FC_SEATS WHERE EXPRESS_NO = OLD.EXPRESS_NO;
  UPDATE EXPRESS SET SC_SEATS_REMAINING = SC_SEATS_REMAINING + OLD.TOTAL_SC_SEATS WHERE EXPRESS_NO = OLD.EXPRESS_NO;
END$$
DELIMITER ;

```

6. View Past transactions:

When the user clicks “View Past Transactions”, all past transactions corresponding to the given User-ID are searched in the mysql RAILWAYDB database. The result-set is passed to an EJS file. If no such trains are found, message is displayed using no_previous.html.

```
app.get('/prev', function (request, response) {
    if (request.session.loggedin) {
        var quer =
            'select BOOKING.* from BOOKING,EXPRESS where BOOKING.EXPRESS_NO=EXPRESS.EXPRESS_NO and ' +
            'USER_ID = ' +
            "'" +
            request.session.username +
            "'" +
            ' and current_timestamp() > EXPRESS.STARTING_TIME';
        con.query(quer, function (error, results, fields) {
            if (error) throw error;
            if (results.length <= 0) response.sendfile(path.join(__dirname, 'no_previous.html'));
            else response.render('viewpast.ejs', { result: results });
        });
    } else {
        response.redirect('/login');
    }
});
```

7. Logout:

When the user clicks the logout button, the session is ended, and all the session variables are de initialised and the page is redirected to the landing page.

```
app.get('/logout', function (request, response) {
    request.session.loggedin = false;
    request.session.username = ' ';
    response.redirect('/');
});
```

RESULTS SNAPSHOTS

ADMIN SIDE:

1. Adding Trains to TRAIN relation:

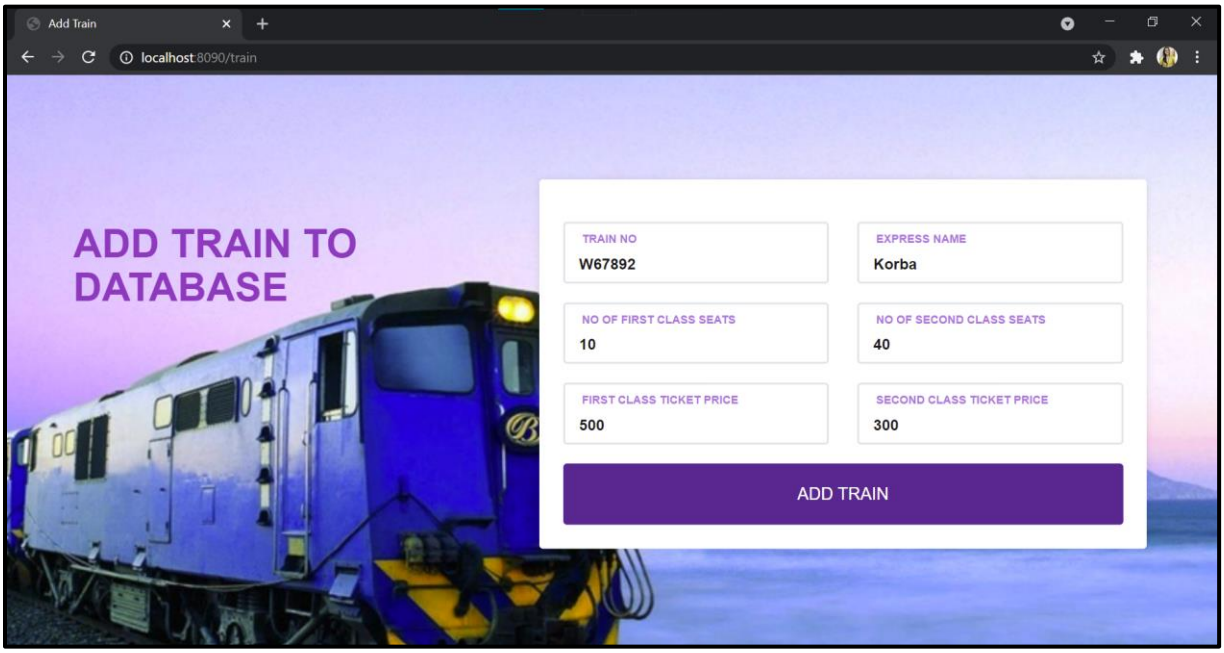
   Running the insert_train file with SQL queries is shown below.

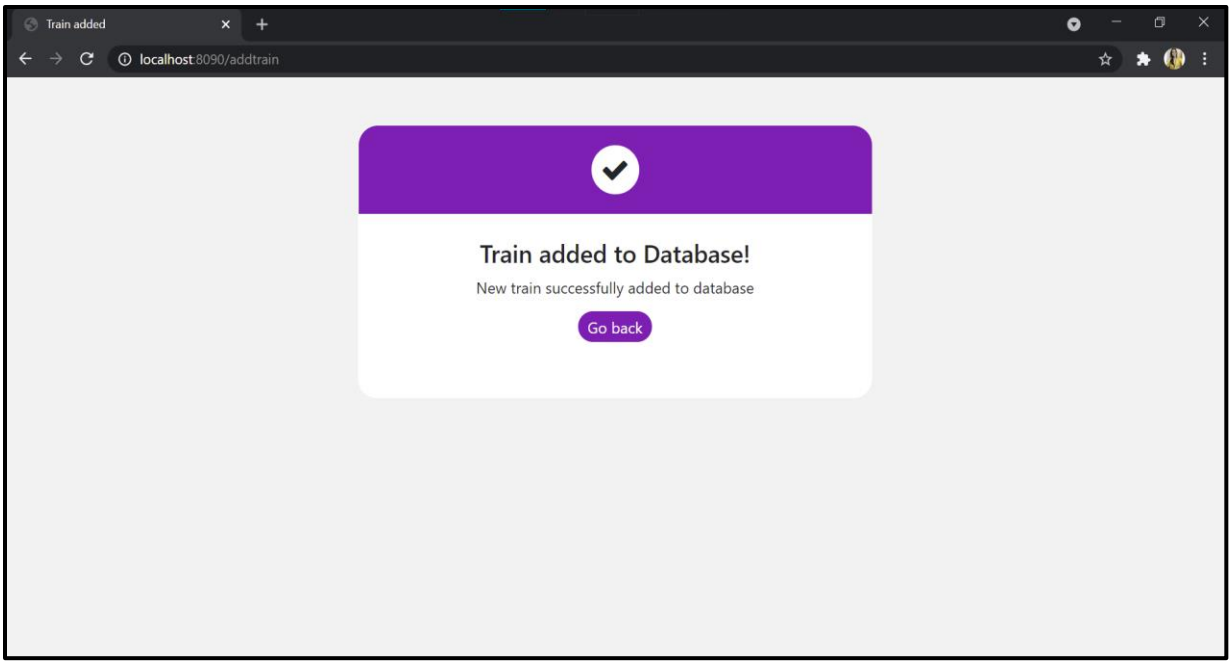
```
mysql> source C:\Users\DeLL\Desktop\DBMS PROJECT\SQL files\insert_train.sql
Query OK, 1 row affected (0.01 sec)

Query OK, 1 row affected (0.01 sec)

Query OK, 1 row affected (0.00 sec)

Query OK, 1 row affected (0.00 sec)

Query OK, 1 row affected (0.00 sec)

Query OK, 1 row affected (0.00 sec)

Query OK, 1 row affected (0.00 sec)

Query OK, 1 row affected (0.00 sec)
```

```
Query OK, 1 row affected (0.00 sec)

+---------------------------------+
| CONCAT (EXPRESS_NAME,' Express') |
+---------------------------------+
| Howrah Express                  |
| Falaknuma Express               |
| Guwahati Express                |
| Shalimar Express                |
| East Coast Express              |
| Rajdhani Express                |
| Kongu Express                   |
| Nizamuddin Express              |
| Karnataka Express               |
| Ganga-Kaveri Express            |
| Pandian Express                 |
| Vaigai Express                  |
| Tejas Express                   |
| Guruvayur Express               |
| Mysuru Express                  |
| Shatabdi Express                |
| Bagmati Express                 |
| Kaveri Express                  |
| Gorakhpur Express               |
| Konkan Express                  |
| Ernakulam Express               |
| Korba Express                   |
+---------------------------------+
22 rows in set (0.00 sec)
```

```
mysql> SELECT * FROM TRAIN;
+----------+--------------+----------+----------+----------+----------+
| TRAIN_NO | EXPRESS_NAME | FC_SEATS | SC_SEATS | FC_PRICE | SC_PRICE |
+----------+--------------+----------+----------+----------+----------+
| E15698   | Howrah       |       10 |       20 |      500 |      300 |
| E16742   | Falaknuma    |       15 |       25 |      450 |      300 |
| E17431   | Guwahati     |       20 |       30 |      400 |      200 |
| E18156   | Shalimar     |       10 |       20 |      550 |      350 |
| E19473   | East Coast   |       10 |       15 |      550 |      350 |
| N21023   | Rajdhani     |       11 |       20 |      500 |      200 |
| N24563   | Kongu        |       15 |       30 |      450 |      300 |
| N25931   | Nizamuddin   |       10 |       30 |      400 |      300 |
| N26781   | Karnataka    |       10 |       25 |      400 |      250 |
| N36913   | Ganga-Kaveri |       15 |       30 |      400 |      250 |
| S44592   | Pandian      |       15 |       20 |      450 |      200 |
| S45678   | Vaigai       |       10 |       25 |      400 |      250 |
| S46423   | Tejas        |       10 |       30 |      400 |      300 |
| S47365   | Guruvayur    |       15 |       25 |      350 |      200 |
| S53813   | Mysuru       |       20 |       35 |      400 |      250 |
| S54671   | Shatabdi     |       10 |       15 |      450 |      250 |
| S56892   | Bagmati      |       15 |       35 |      400 |      200 |
| S57893   | Kaveri       |       20 |       30 |      500 |      300 |
| W61962   | Gorakhpur    |       15 |       25 |      400 |      200 |
| W63014   | Konkan       |       15 |       30 |      300 |      200 |
| W64567   | Ernakulam    |       10 |       25 |      400 |      250 |
| W65681   | Korba        |       10 |       20 |      450 |      300 |
+----------+--------------+----------+----------+----------+----------+
22 rows in set (0.01 sec)
```

Adding a train using the admin side of the webpage is demonstrated below.






```
mysql> SELECT * FROM TRAIN;
+----------+--------------+----------+----------+----------+----------+
| TRAIN_NO | EXPRESS_NAME | FC_SEATS | SC_SEATS | FC_PRICE | SC_PRICE |
+----------+--------------+----------+----------+----------+----------+
| E15698   | Howrah       |       10 |       20 |      500 |      300 |
| E16742   | Falaknuma    |       15 |       25 |      450 |      300 |
| E17431   | Guwahati     |       20 |       30 |      400 |      200 |
| E18156   | Shalimar     |       10 |       20 |      550 |      350 |
| E19473   | East Coast   |       10 |       15 |      550 |      350 |
| N21023   | Rajdhani     |       11 |       20 |      500 |      200 |
| N24563   | Kongu        |       15 |       30 |      450 |      300 |
| N25931   | Nizamuddin   |       10 |       30 |      400 |      300 |
| N26781   | Karnataka    |       10 |       25 |      400 |      250 |
| N36913   | Ganga-Kaveri |       15 |       30 |      400 |      250 |
| S44592   | Pandian      |       15 |       20 |      450 |      200 |
| S45678   | Vaigai       |       10 |       25 |      400 |      250 |
| S46423   | Tejas        |       10 |       30 |      400 |      300 |
| S47365   | Guruvayur    |       15 |       25 |      350 |      200 |
| S53813   | Mysuru       |       20 |       35 |      400 |      250 |
| S54671   | Shatabdi     |       10 |       15 |      450 |      250 |
| S56892   | Bagmati      |       15 |       35 |      400 |      200 |
| S57893   | Kaveri       |       20 |       30 |      500 |      300 |
| W61962   | Gorakhpur    |       15 |       25 |      400 |      200 |
| W63014   | Konkan       |       15 |       30 |      300 |      200 |
| W64567   | Ernakulam    |       10 |       25 |      400 |      250 |
| W65681   | Korba        |       10 |       20 |      450 |      300 |
| W67892   | Korba        |       10 |       40 |      500 |      300 |
+----------+--------------+----------+----------+----------+----------+
23 rows in set (0.00 sec)
```

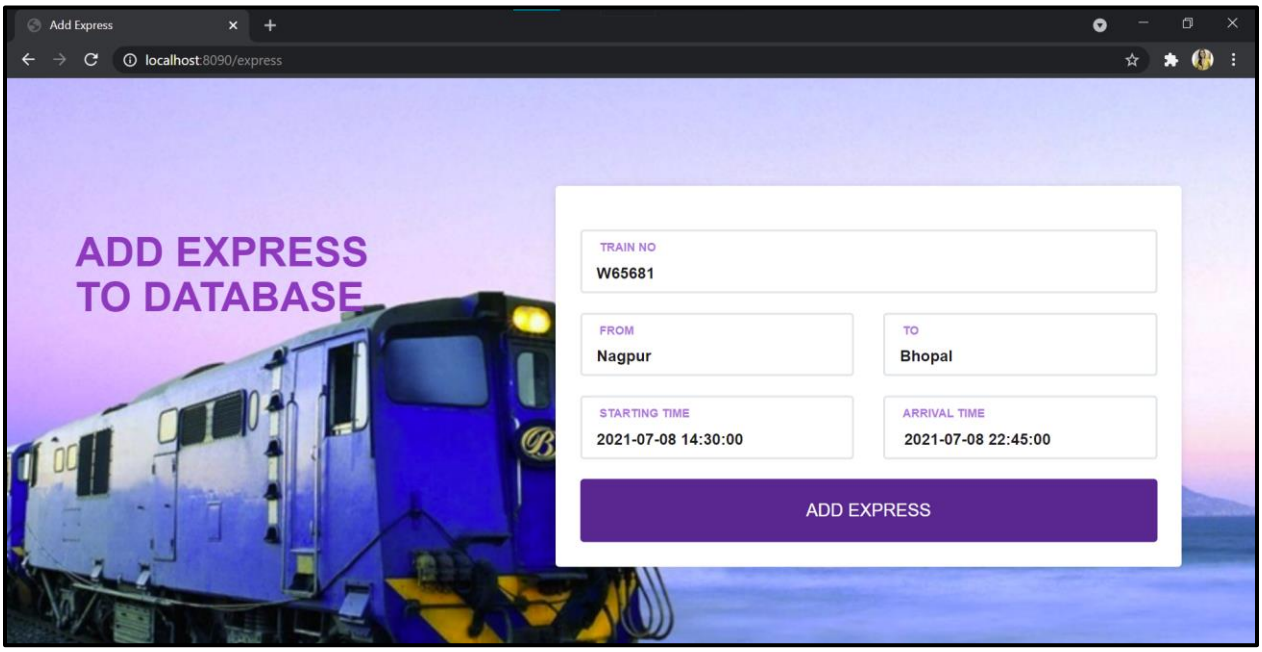
2. Adding Expresses to the EXPRESS relation

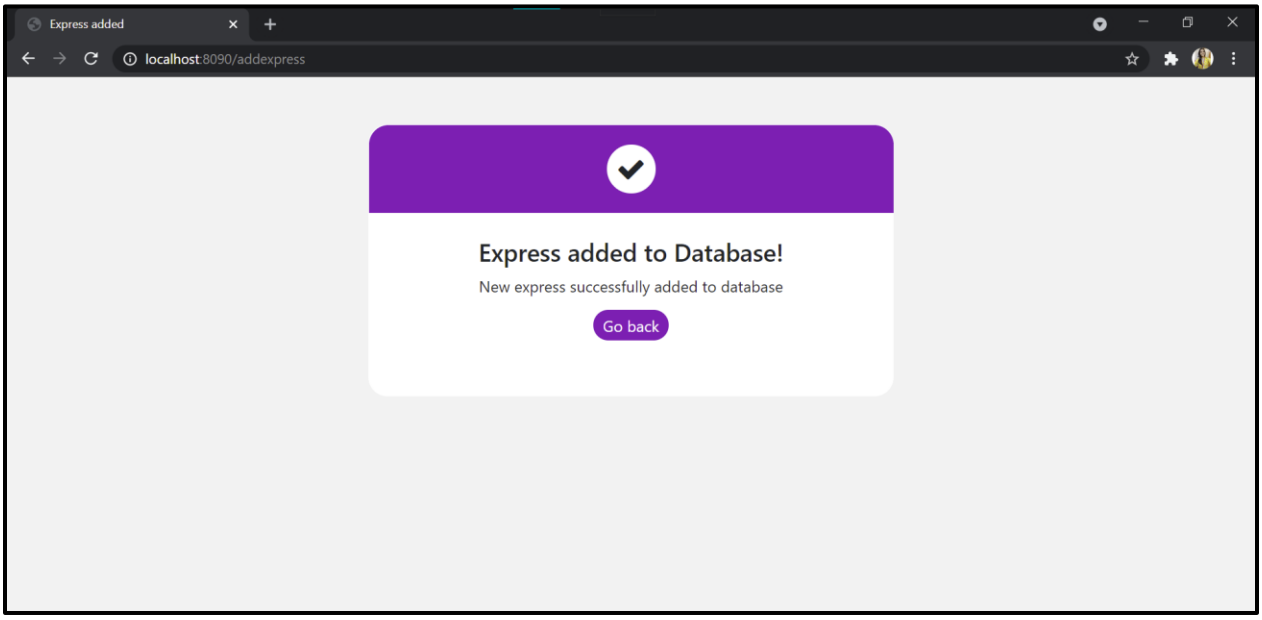
Running the insert_express file with SQL queries is shown below.

```
mysql> source C:\Users\DeLL\Desktop\DBMS PROJECT\SQL files\insert_express.sql
Query OK, 1 row affected (0.01 sec)
Records: 1  Duplicates: 0  Warnings: 0

Query OK, 1 row affected (0.01 sec)
Records: 1  Duplicates: 0  Warnings: 0

Query OK, 1 row affected (0.00 sec)
Records: 1  Duplicates: 0  Warnings: 0

Query OK, 1 row affected (0.00 sec)
Records: 1  Duplicates: 0  Warnings: 0

Query OK, 1 row affected (0.00 sec)
Records: 1  Duplicates: 0  Warnings: 0

Query OK, 1 row affected (0.00 sec)
Records: 1  Duplicates: 0  Warnings: 0

Query OK, 1 row affected (0.00 sec)
Records: 1  Duplicates: 0  Warnings: 0
```

```
mysql> SELECT * FROM EXPRESS;
+------------+----------+------------+----------+---------------------+---------------------+--------------------+--------------------+
| EXPRESS_NO | TRAIN_NO | FROMM      | TOO      | STARTING_TIME       | ARRIVAL_TIME        | FC_SEATS_REMAINING | SC_SEATS_REMAINING |
+------------+----------+------------+----------+---------------------+---------------------+--------------------+--------------------+
|      10001 | E15698   | Kolkata    | Hyderabad | 2021-06-13 08:30:00 | 2021-06-14 05:29:00 |                 10 |                 20 |
|      10002 | E15698   | Kolkata    | Hyderabad | 2021-06-15 10:30:00 | 2021-06-16 07:29:00 |                 10 |                 20 |
|      10003 | E15698   | Kolkata    | Hyderabad | 2021-05-29 10:30:00 | 2021-05-29 07:29:00 |                 10 |                 20 |
|      10004 | E16742   | Kolkata    | Hyderabad | 2021-06-13 20:30:00 | 2021-06-14 12:29:00 |                 15 |                 25 |
|      10005 | E16742   | Kolkata    | Hyderabad | 2021-05-29 20:30:00 | 2021-05-30 12:29:00 |                 13 |                 21 |
|      10006 | E17431   | Kolkata    | Hyderabad | 2021-06-13 10:50:00 | 2021-06-14 08:45:00 |                 20 |                 30 |
|      10007 | E17431   | Kolkata    | Hyderabad | 2021-06-15 12:50:00 | 2021-06-16 10:45:00 |                 20 |                 30 |
|      10008 | E18156   | Kolkata    | Hyderabad | 2021-06-13 02:30:00 | 2021-06-14 00:29:00 |                 10 |                 20 |
|      10009 | E18156   | Kolkata    | Hyderabad | 2021-05-29 02:30:00 | 2021-05-30 00:29:00 |                 10 |                 20 |
|      10010 | E19473   | Kolkata    | Hyderabad | 2021-06-13 13:00:01 | 2021-06-14 14:45:01 |                 10 |                 15 |
|      10011 | E19473   | Kolkata    | Hyderabad | 2021-06-15 13:00:01 | 2021-06-16 14:45:01 |                 10 |                 15 |
|      10012 | N21023   | Delhi      | Bangalore | 2021-06-15 18:30:00 | 2021-06-17 13:00:00 |                 11 |                 20 |
|      10013 | N21023   | Delhi      | Bangalore | 2021-06-17 15:30:00 | 2021-06-19 12:00:00 |                 11 |                 20 |
|      10014 | N24563   | Delhi      | Bangalore | 2021-06-15 08:30:00 | 2021-06-17 02:00:00 |                 15 |                 30 |
|      10015 | N24563   | Delhi      | Bangalore | 2021-06-17 08:30:00 | 2021-06-19 02:00:00 |                 15 |                 30 |
|      10016 | N25931   | Delhi      | Bangalore | 2021-06-15 00:30:00 | 2021-06-16 23:30:00 |                 10 |                 30 |
|      10017 | N26781   | Delhi      | Bangalore | 2021-06-15 13:30:00 | 2021-06-17 11:25:00 |                 10 |                 25 |
|      10018 | N26781   | Delhi      | Bangalore | 2021-06-17 10:30:00 | 2021-06-19 09:25:00 |                 10 |                 25 |
|      10019 | N36913   | Rameshwaram | Varanasi | 2021-06-20 12:30:00 | 2021-06-23 03:30:00 |                 15 |                 30 |
|      10020 | N36913   | Rameshwaram | Varanasi | 2021-06-21 12:30:00 | 2021-06-24 03:30:00 |                 15 |                 30 |
|      10021 | N36913   | Rameshwaram | Varanasi | 2021-05-29 12:30:00 | 2021-06-01 03:30:00 |                 13 |                 30 |
|      10022 | N36913   | Rameshwaram | Varanasi | 2021-06-23 08:30:00 | 2021-06-25 00:30:00 |                 15 |                 30 |
|      10023 | S44592   | Madurai    | Chennai  | 2021-06-23 23:30:00 | 2021-06-24 05:29:00 |                 15 |                 20 |
|      10024 | S44592   | Madurai    | Chennai  | 2021-06-24 13:30:00 | 2021-06-24 21:29:00 |                 15 |                 20 |
|      10025 | S45678   | Madurai    | Chennai  | 2021-06-23 10:00:00 | 2021-06-23 17:10:00 |                 10 |                 25 |
|      10026 | S45678   | Madurai    | Chennai  | 2021-05-30 10:00:00 | 2021-05-30 17:10:00 |                 10 |                 25 |
|      10027 | S45678   | Madurai    | Chennai  | 2021-06-24 10:00:00 | 2021-06-24 17:10:00 |                 10 |                 25 |
|      10028 | S45678   | Madurai    | Chennai  | 2021-05-30 10:00:00 | 2021-05-30 17:10:00 |                 10 |                 25 |
|      10029 | S46423   | Madurai    | Chennai  | 2021-06-23 13:30:00 | 2021-06-23 20:05:00 |                 10 |                 30 |
|      10030 | S46423   | Madurai    | Chennai  | 2021-05-30 13:30:00 | 2021-05-30 20:05:00 |                  9 |                 30 |
|      10031 | S47365   | Madurai    | Chennai  | 2021-06-23 23:30:00 | 2021-06-23 05:29:00 |                 15 |                 25 |
|      10032 | S47365   | Madurai    | Chennai  | 2021-06-24 10:30:00 | 2021-06-24 15:10:00 |                 15 |                 25 |
|      10033 | S47365   | Madurai    | Chennai  | 2021-05-30 10:30:00 | 2021-05-30 15:10:00 |                 15 |                 25 |
|      10034 | S53813   | Chennai    | Mysore   | 2021-06-25 10:00:00 | 2021-06-25 18:10:00 |                 20 |                 35 |
|      10035 | S53813   | Chennai    | Mysore   | 2021-06-27 08:00:00 | 2021-06-27 16:10:00 |                 20 |                 35 |
|      10036 | S54671   | Chennai    | Mysore   | 2021-06-25 09:00:00 | 2021-06-25 17:50:00 |                 10 |                 15 |
|      10037 | S56892   | Chennai    | Mysore   | 2021-06-25 14:00:00 | 2021-06-25 23:50:00 |                 15 |                 35 |
|      10038 | S56892   | Chennai    | Mysore   | 2021-06-27 14:00:00 | 2021-06-27 23:50:00 |                 15 |                 35 |
|      10039 | S57893   | Chennai    | Mysore   | 2021-06-25 17:00:00 | 2021-06-26 02:00:00 |                 20 |                 30 |
|      10040 | S57893   | Chennai    | Mysore   | 2021-06-27 11:00:00 | 2021-06-27 18:00:00 |                 20 |                 30 |
|      10041 | W61962   | Nagpur     | Bhopal   | 2021-06-28 15:30:00 | 2021-06-28 23:45:00 |                 15 |                 25 |
|      10042 | W63014   | Nagpur     | Bhopal   | 2021-06-28 10:30:00 | 2021-06-28 18:45:00 |                 15 |                 30 |
|      10043 | W63014   | Nagpur     | Bhopal   | 2021-06-30 07:30:00 | 2021-06-30 15:45:00 |                 15 |                 30 |
|      10044 | W64567   | Nagpur     | Bhopal   | 2021-06-28 05:50:00 | 2021-06-28 13:45:00 |                 10 |                 25 |
|      10045 | W64567   | Nagpur     | Bhopal   | 2021-06-30 05:50:00 | 2021-06-30 13:45:00 |                 10 |                 25 |
|      10046 | W65681   | Nagpur     | Bhopal   | 2021-06-28 14:30:00 | 2021-06-29 00:45:00 |                 10 |                 20 |
|      10047 | W65681   | Nagpur     | Bhopal   | 2021-06-30 14:30:00 | 2021-06-30 22:45:00 |                 10 |                 20 |
+------------+----------+------------+----------+---------------------+---------------------+--------------------+--------------------+
47 rows in set (0.01 sec)
```

Adding an express using the admin side of the webpage is demonstrated below.





```
mysql> SELECT * FROM EXPRESS;
```

EXPRESS_NO	TRAIN_NO	FROMM	TOO	STARTING_TIME	ARRIVAL_TIME	FC_SEATS_REMAINING	SC_SEATS_REMAINING
10001	E15698	Kolkata	Hyderabad	2021-06-13 08:30:00	2021-06-14 05:29:00	10	20
10002	E15698	Kolkata	Hyderabad	2021-06-15 10:30:00	2021-06-16 07:29:00	10	20
10003	E15698	Kolkata	Hyderabad	2021-05-29 10:30:00	2021-05-29 07:29:00	10	20
10004	E16742	Kolkata	Hyderabad	2021-06-13 20:30:00	2021-06-14 12:29:00	15	25
10005	E16742	Kolkata	Hyderabad	2021-05-29 20:30:00	2021-05-30 12:29:00	13	21
10006	E17431	Kolkata	Hyderabad	2021-06-13 10:50:00	2021-06-14 08:45:00	20	30
10007	E17431	Kolkata	Hyderabad	2021-06-15 12:50:00	2021-06-16 10:45:00	20	30
10008	E18156	Kolkata	Hyderabad	2021-06-13 02:30:00	2021-06-14 00:29:00	10	20
10009	E18156	Kolkata	Hyderabad	2021-05-29 02:30:00	2021-05-30 00:29:00	10	20
10010	E19473	Kolkata	Hyderabad	2021-06-13 13:00:01	2021-06-14 14:45:01	10	15
10011	E19473	Kolkata	Hyderabad	2021-06-15 13:00:01	2021-06-16 14:45:01	10	15
10012	N21023	Delhi	Bangalore	2021-06-15 18:30:00	2021-06-17 13:00:00	11	20
10013	N21023	Delhi	Bangalore	2021-06-17 15:30:00	2021-06-19 12:00:00	11	20
10014	N24563	Delhi	Bangalore	2021-06-15 08:30:00	2021-06-17 02:00:00	15	30
10015	N24563	Delhi	Bangalore	2021-06-17 08:30:00	2021-06-19 02:00:00	15	30
10016	N25931	Delhi	Bangalore	2021-06-15 00:30:00	2021-06-16 23:30:00	10	30
10017	N26781	Delhi	Bangalore	2021-06-15 13:30:00	2021-06-17 11:25:00	10	25
10018	N26781	Delhi	Bangalore	2021-06-17 10:30:00	2021-06-19 09:25:00	10	25
10019	N36913	Rameshwaram	Varanasi	2021-06-20 12:30:00	2021-06-23 03:30:00	15	30
10020	N36913	Rameshwaram	Varanasi	2021-06-21 12:30:00	2021-06-24 03:30:00	15	30
10021	N36913	Rameshwaram	Varanasi	2021-05-29 12:30:00	2021-06-01 03:30:00	13	30
10022	N36913	Rameshwaram	Varanasi	2021-06-23 08:30:00	2021-06-25 00:30:00	15	30
10023	S44592	Madurai	Chennai	2021-06-23 23:30:00	2021-06-24 05:29:00	15	20
10024	S44592	Madurai	Chennai	2021-06-24 13:30:00	2021-06-24 21:29:00	15	20
10025	S45678	Madurai	Chennai	2021-06-23 10:00:00	2021-06-23 17:10:00	10	25
10026	S45678	Madurai	Chennai	2021-05-30 10:00:00	2021-05-30 17:10:00	10	25
10027	S45678	Madurai	Chennai	2021-06-24 10:00:00	2021-06-24 17:10:00	10	25
10028	S45678	Madurai	Chennai	2021-05-30 10:00:00	2021-05-30 17:10:00	10	25
10029	S46423	Madurai	Chennai	2021-06-23 13:30:00	2021-06-23 20:05:00	10	30
10030	S46423	Madurai	Chennai	2021-05-30 13:30:00	2021-05-30 20:05:00	9	30
10031	S47365	Madurai	Chennai	2021-06-23 23:30:00	2021-06-23 05:29:00	15	25
10032	S47365	Madurai	Chennai	2021-06-24 10:30:00	2021-06-24 15:10:00	15	25
10033	S47365	Madurai	Chennai	2021-05-30 10:30:00	2021-05-30 15:10:00	15	25
10034	S53813	Chennai	Mysore	2021-06-25 10:00:00	2021-06-25 18:10:00	20	35
10035	S53813	Chennai	Mysore	2021-06-27 08:00:00	2021-06-27 16:10:00	20	35
10036	S54671	Chennai	Mysore	2021-06-25 09:00:00	2021-06-25 17:50:00	10	15
10037	S56892	Chennai	Mysore	2021-06-25 14:00:00	2021-06-25 23:50:00	15	35
10038	S56892	Chennai	Mysore	2021-06-27 14:00:00	2021-06-27 23:50:00	15	35
10039	S57893	Chennai	Mysore	2021-06-25 17:00:00	2021-06-26 02:00:00	20	30
10040	S57893	Chennai	Mysore	2021-06-27 11:00:00	2021-06-27 18:00:00	20	30
10041	W61962	Nagpur	Bhopal	2021-06-28 15:30:00	2021-06-28 23:45:00	15	25
10042	W63014	Nagpur	Bhopal	2021-06-28 10:30:00	2021-06-28 18:45:00	15	30
10043	W63014	Nagpur	Bhopal	2021-06-30 07:30:00	2021-06-30 15:45:00	15	30
10044	W64567	Nagpur	Bhopal	2021-06-28 05:50:00	2021-06-28 13:45:00	10	25
10045	W64567	Nagpur	Bhopal	2021-06-30 05:50:00	2021-06-30 13:45:00	10	25
10046	W65681	Nagpur	Bhopal	2021-06-28 14:30:00	2021-06-29 00:45:00	10	20
10047	W65681	Nagpur	Bhopal	2021-06-30 14:30:00	2021-06-30 22:45:00	10	20
10048	W65681	Nagpur	Bhopal	2021-07-08 14:30:00	2021-07-08 22:45:00	10	20

```
48 rows in set (0.00 sec)
```

We can see that now there are 48 rows in the relation. (earlier there was 47)

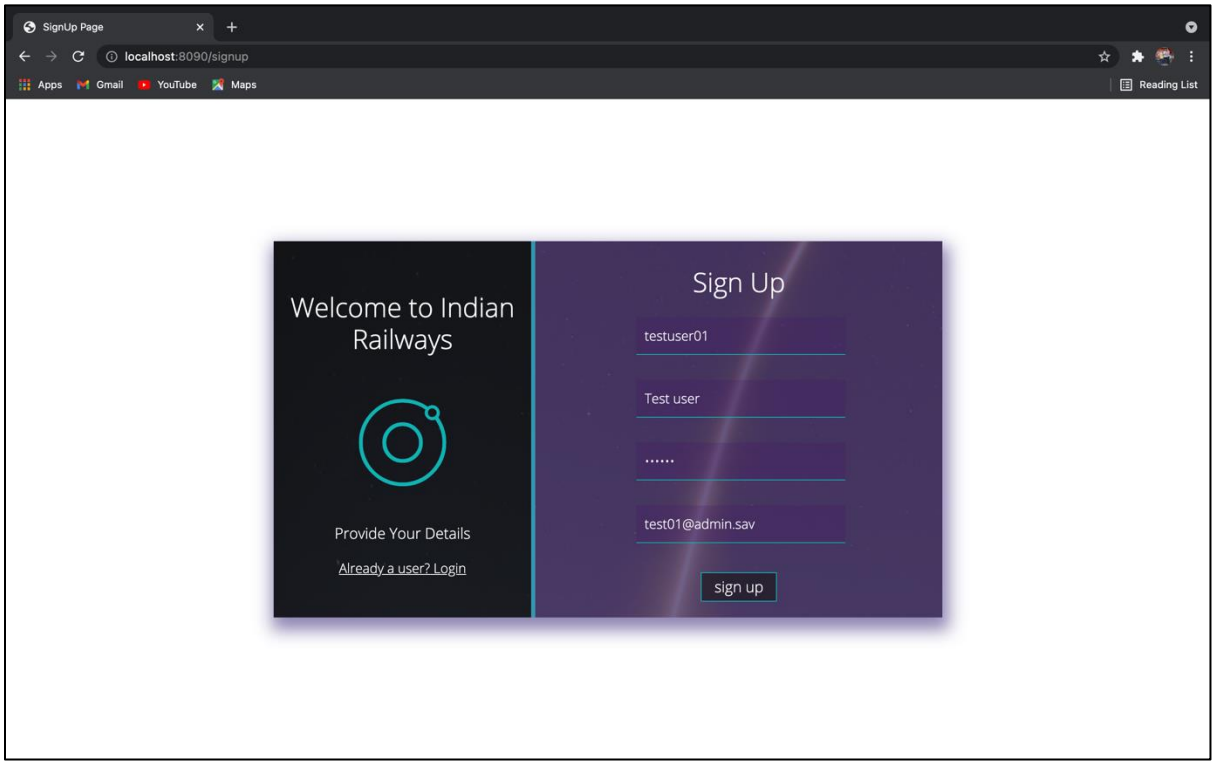
RUNNING THE TRIGGERS, VIEW AND PROCEDURE ON MYSQL TO FACILITATE USER SIDE:

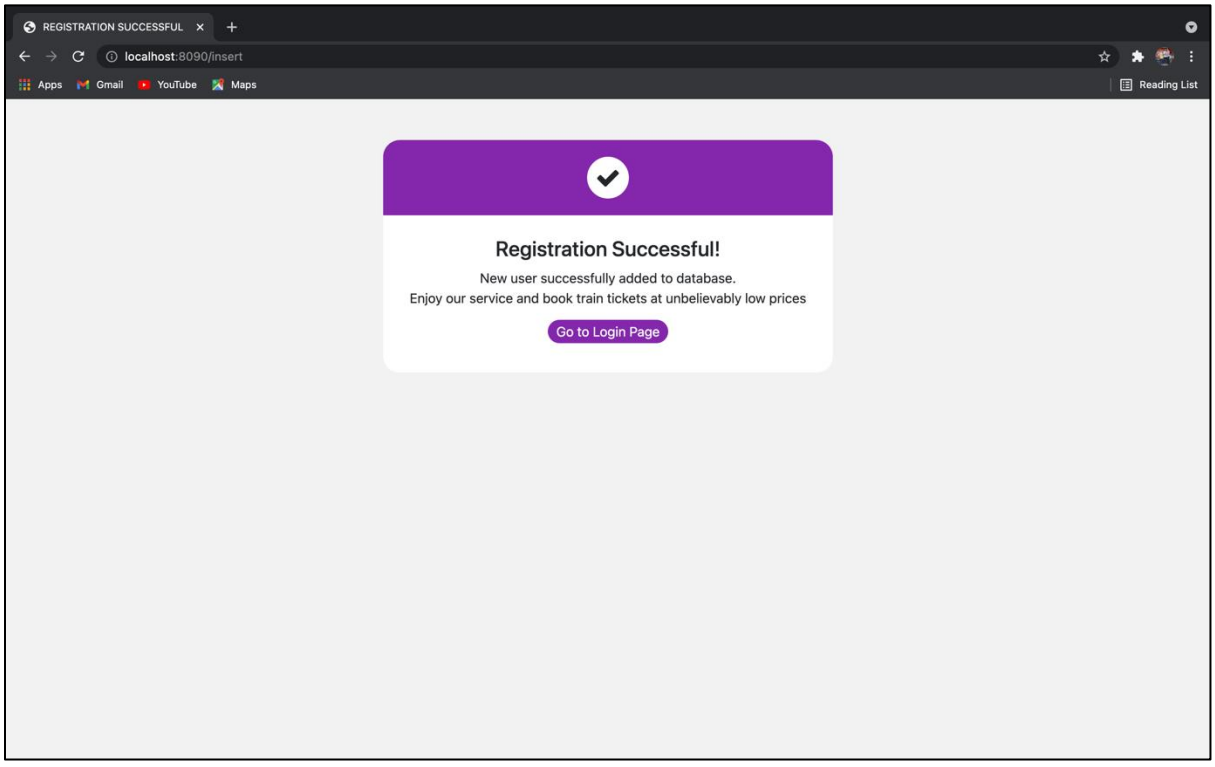
```
mysql> source C:\Users\DeLL\Desktop\DBMS PROJECT\SQL files\expresstrigger.sql
Query OK, 0 rows affected (0.02 sec)

Query OK, 0 rows affected (0.01 sec)

mysql> source C:\Users\DeLL\Desktop\DBMS PROJECT\SQL files\express_view.sql
Query OK, 0 rows affected (0.01 sec)

mysql> source C:\Users\DeLL\Desktop\DBMS PROJECT\SQL files\existingView.sql
Query OK, 0 rows affected (0.01 sec)

mysql> source C:\Users\DeLL\Desktop\DBMS PROJECT\SQL files\proc_addtrain.sql
Query OK, 0 rows affected (0.01 sec)

mysql> source C:\Users\DeLL\Desktop\DBMS PROJECT\SQL files\proc_adduser.sql
Query OK, 0 rows affected (0.01 sec)
```

USER SIDE:

1. Creating account (Sign-Up)

   Successful Signup:

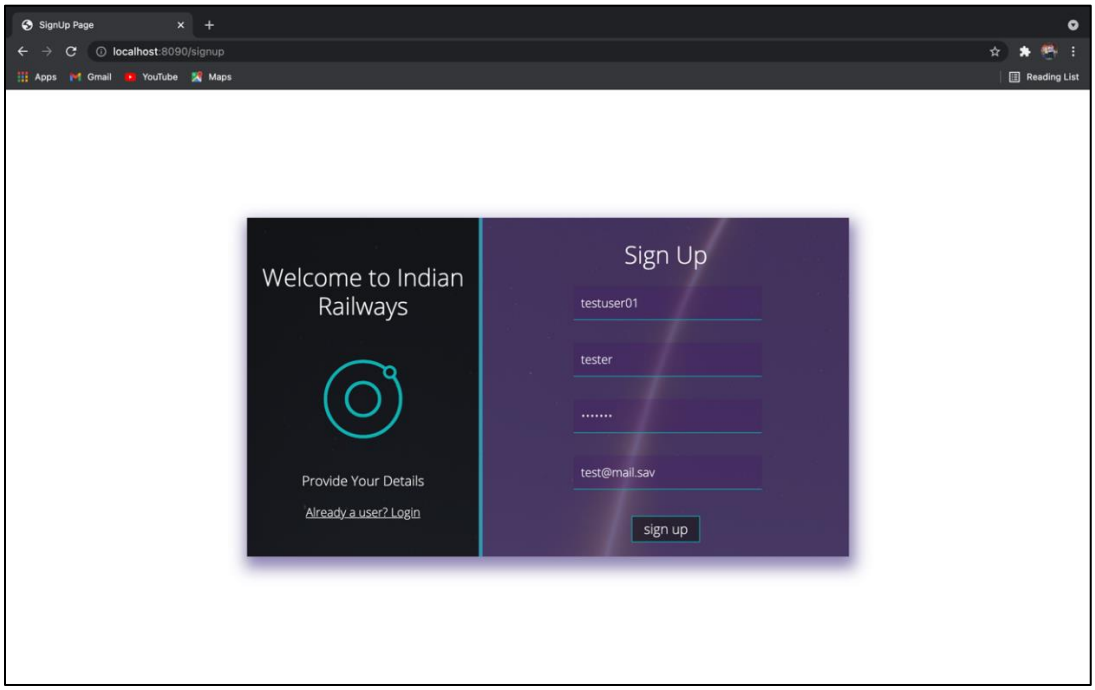


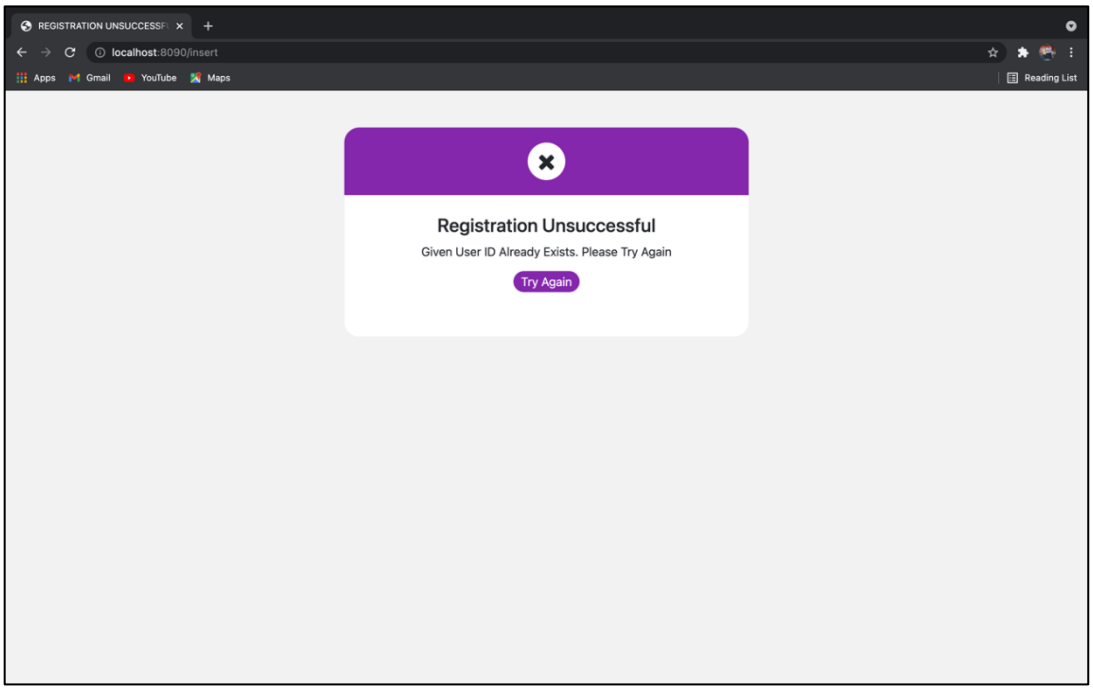


Account details added to database after encryption:

```
mysql> select * from user_details where USER_ID = 'testuser01';
+------------+-----------+-----------------+--------------------------------------------------------------+
| USER_ID    | USERNAME  | EMAIL_ID        | PASSWORD                                                     |
+------------+-----------+-----------------+--------------------------------------------------------------+
| testuser01 | Test user | test01@admin.sav | $2b$10$s6UIshlc6blFCqDq2Ri1ROHG03KGuL5Ccnz7MAjh.5LHTBbLf.h3e |
+------------+-----------+-----------------+--------------------------------------------------------------+
1 row in set (0.01 sec)
```

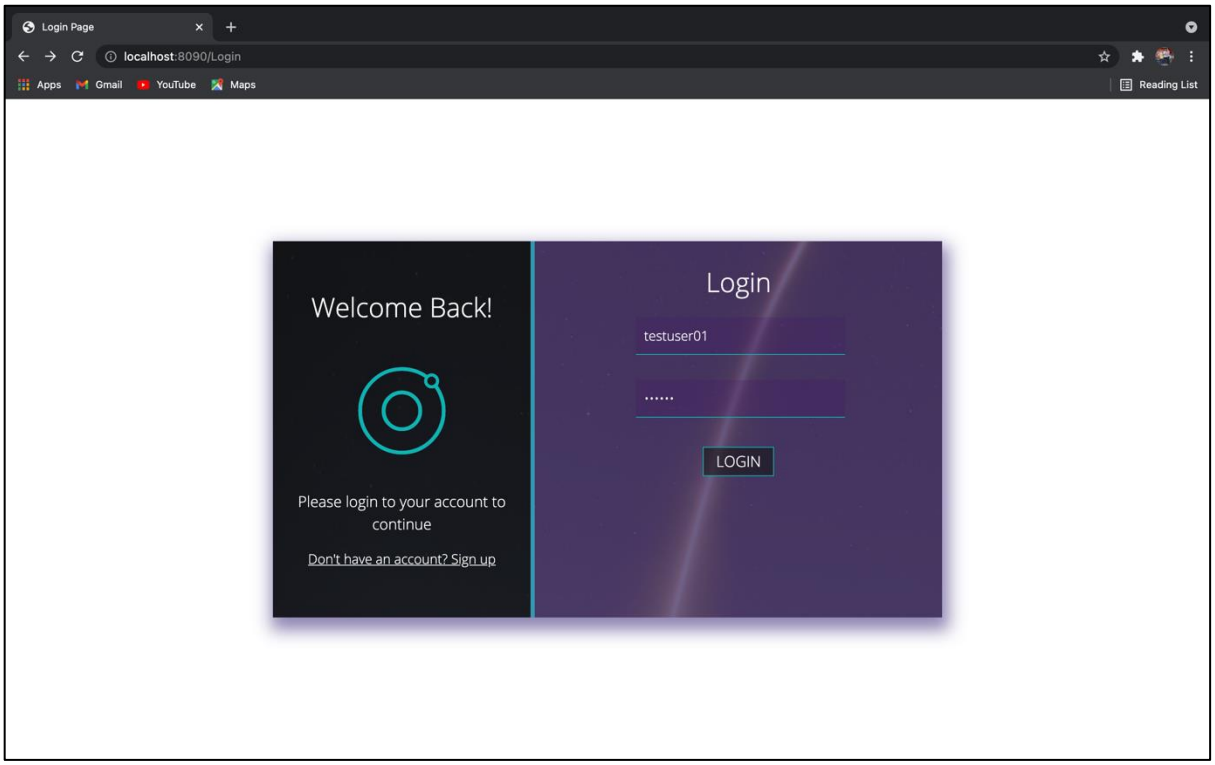
Unsuccessful Signup:

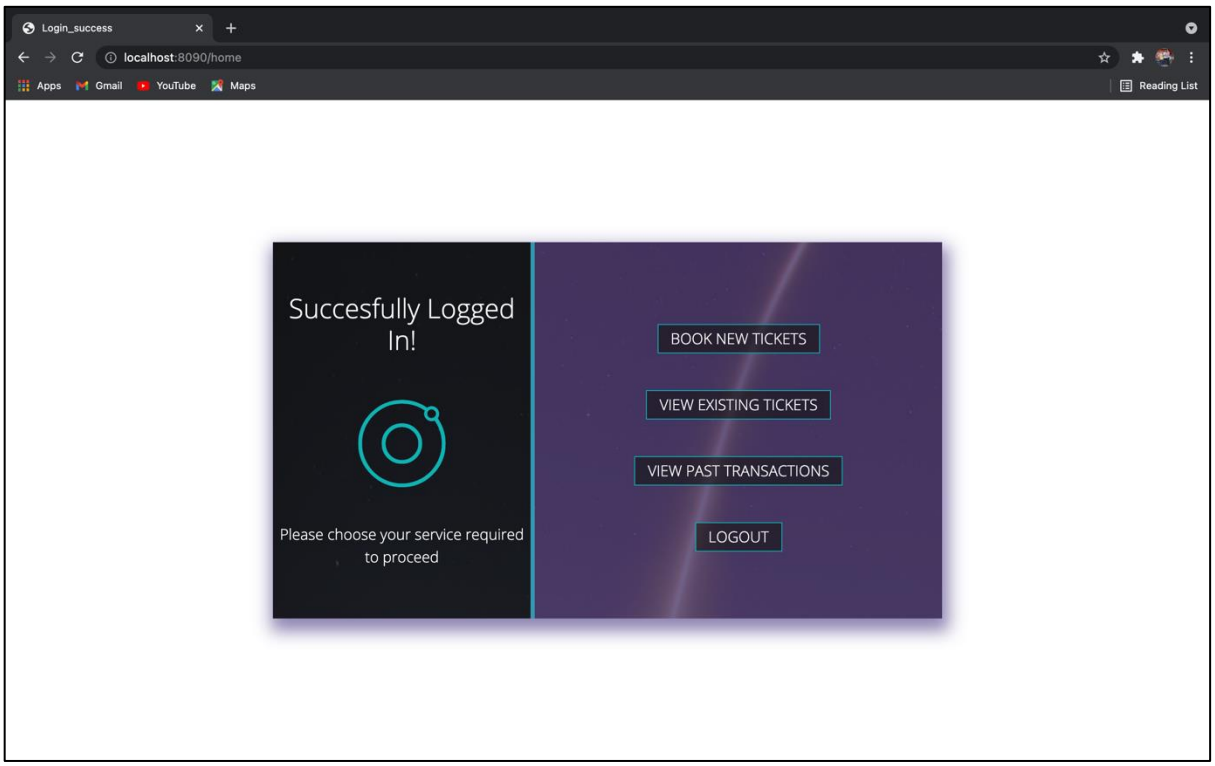




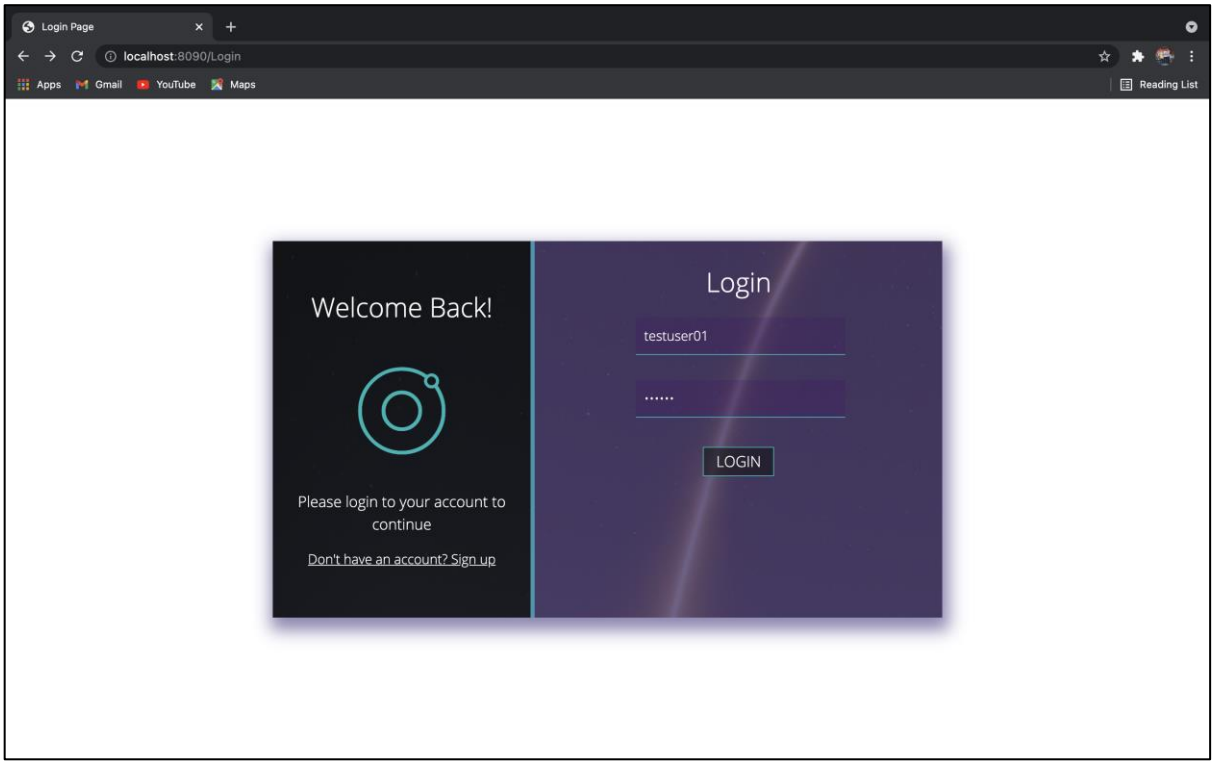
2. Login

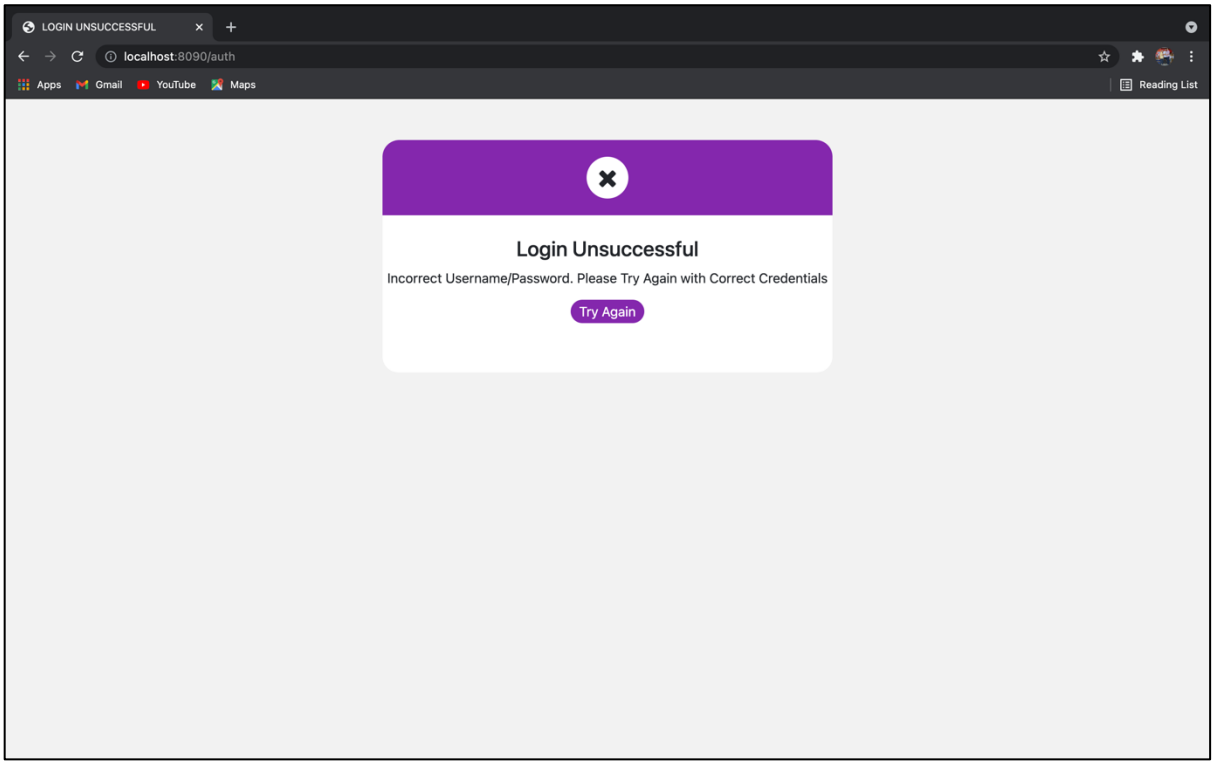
Successful login:





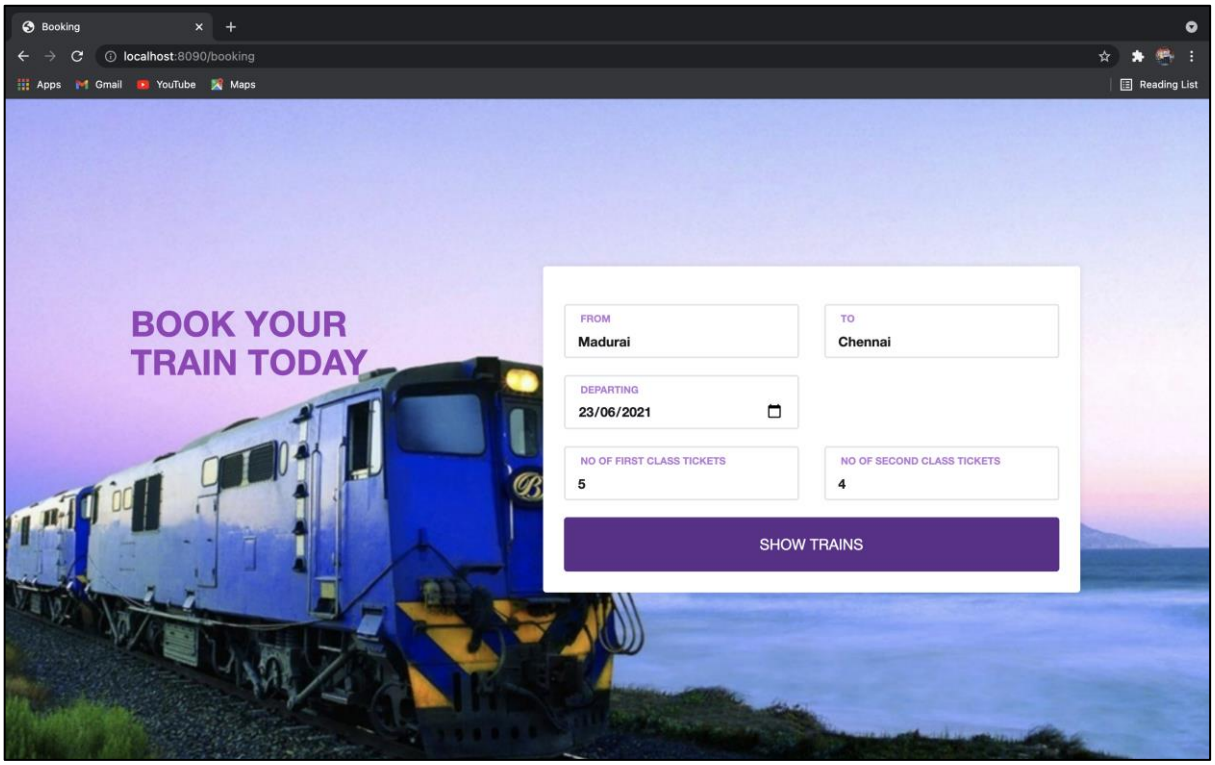
Unsuccessful login:

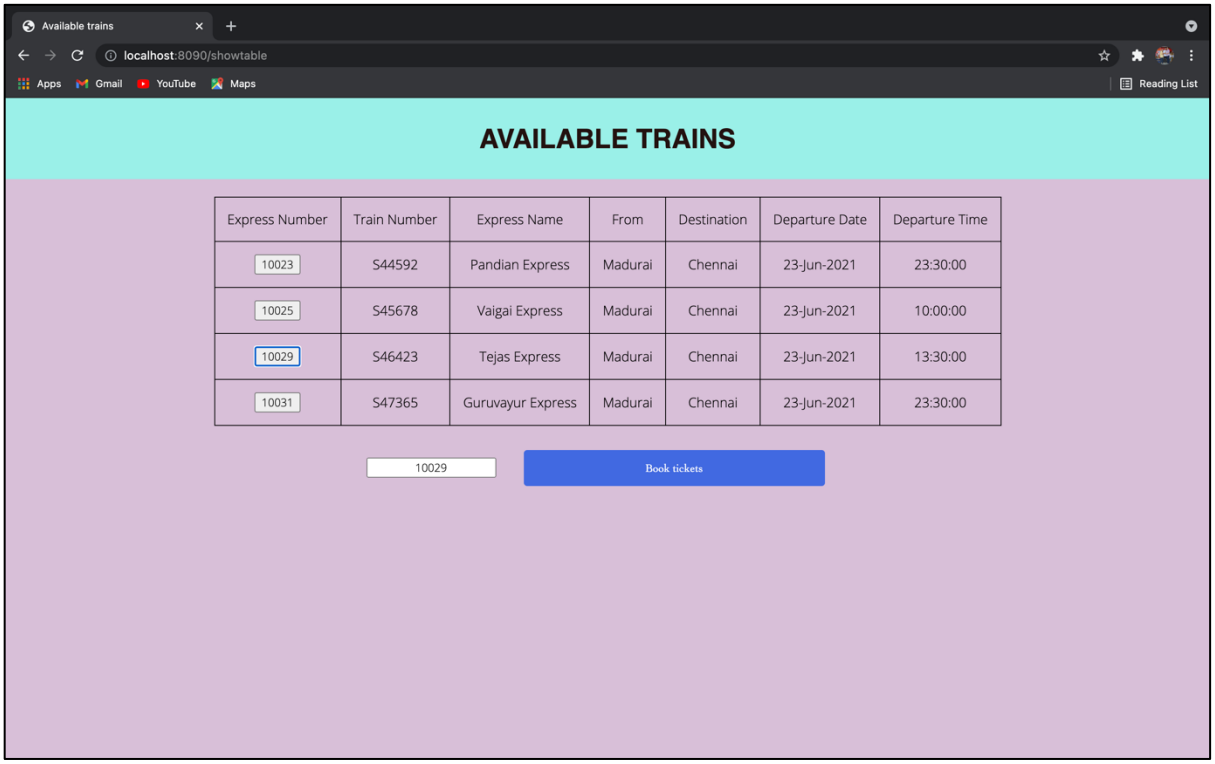


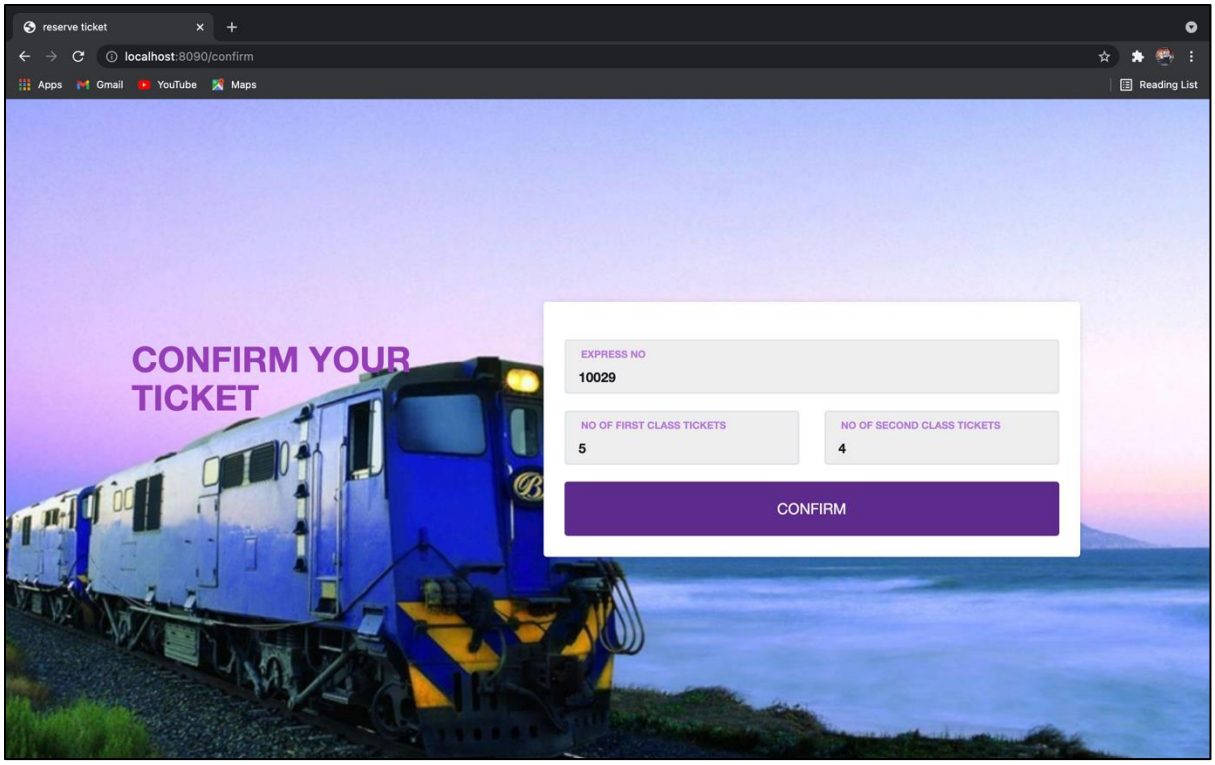


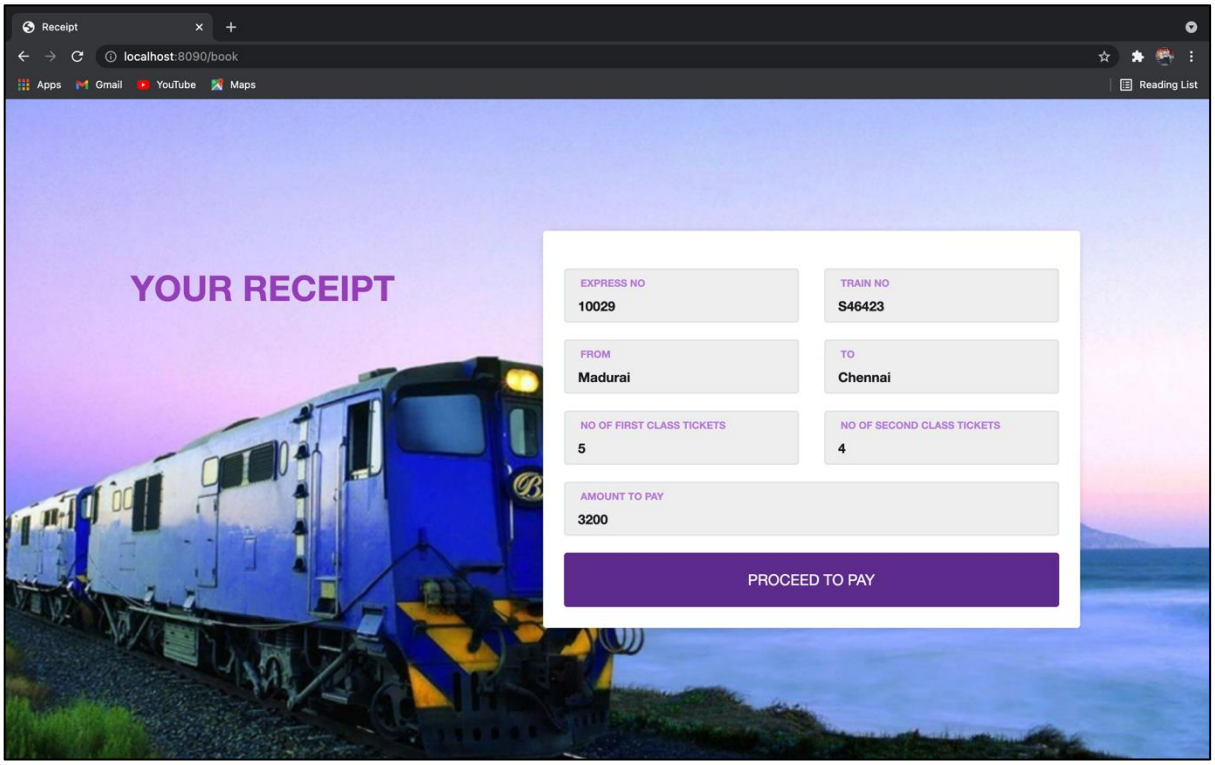
3. Booking

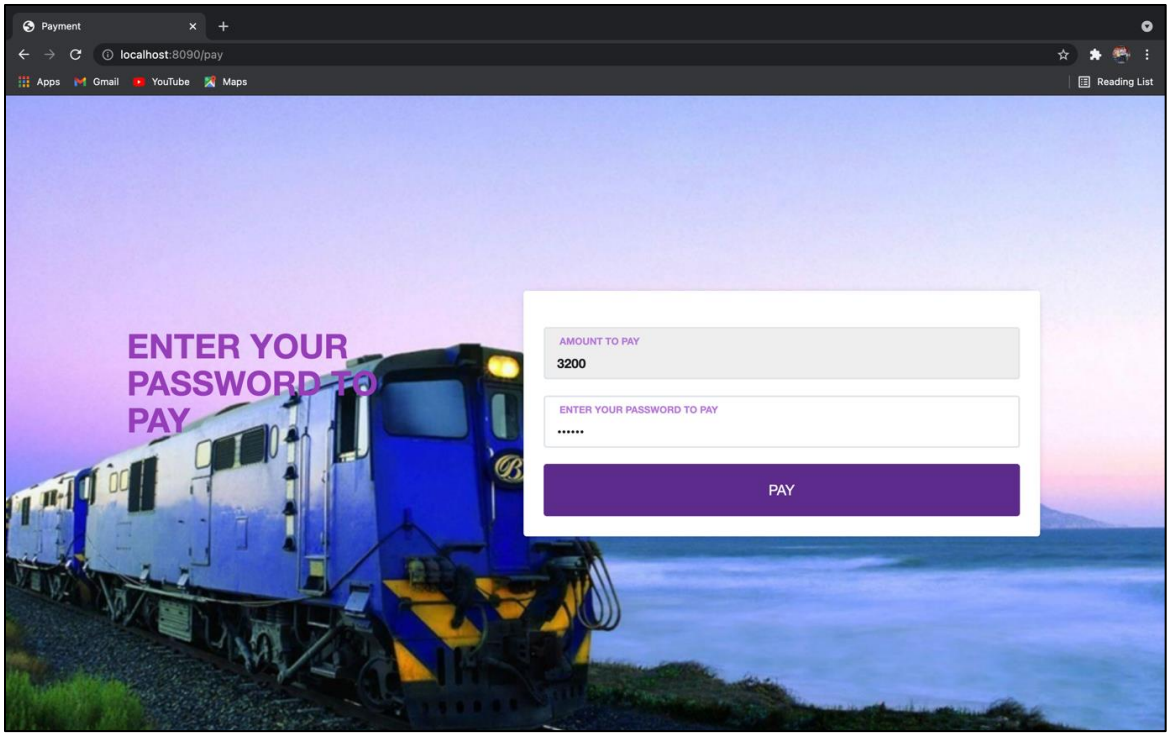
Successful Booking:

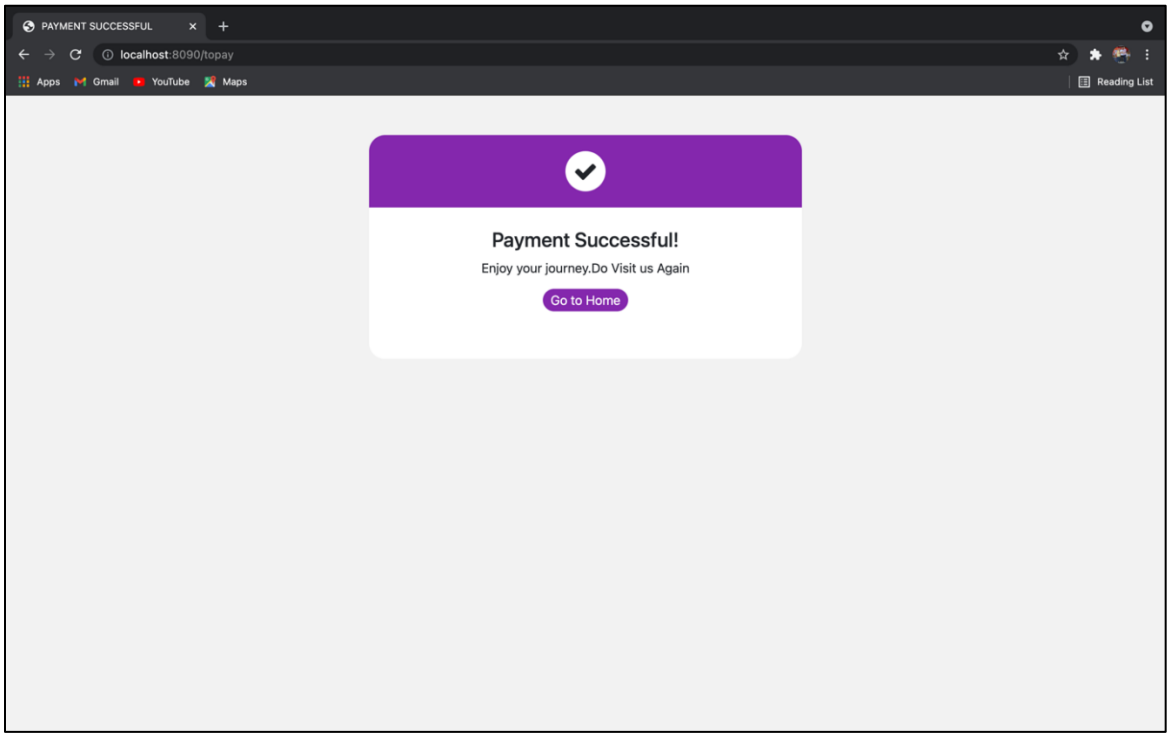





reserve ticket
localhost:8090/confirm
Apps
Gmail
YouTube
Maps
Reading List
CONFIRM YOUR TICKET
EXPRESS NO
10029
NO OF FIRST CLASS TICKETS
5
NO OF SECOND CLASS TICKETS
4
CONFIRM


Receipt
localhost:8090/book
Apps
Gmail
YouTube
Maps
Reading List
YOUR RECEIPT
EXPRESS NO
10029
TRAIN NO
S46423
FROM
Madurai
TO
Chennai
NO OF FIRST CLASS TICKETS
5
NO OF SECOND CLASS TICKETS
4
AMOUNT TO PAY
3200
PROCEED TO PAY






```
mysql> select * from booking where user_id = 'testuser01';
+------------+------------+------------+----------+----------------+----------------+------------+
| BOOKING_ID | EXPRESS_NO | USER_ID    | TRAIN_NO | TOTAL_FC_SEATS | TOTAL_SC_SEATS | TOTAL_COST |
+------------+------------+------------+----------+----------------+----------------+------------+
|          5 |      10029 | testuser01 | S46423   |              5 |              4 |       3200 |
+------------+------------+------------+----------+----------------+----------------+------------+
1 row in set (0.00 sec)
```

The trigger has reduced the number of seats in the express relation as follows:

Earlier:

10027	S45678	Madurai	Chennai	2021-06-24 10:00:00	2021-06-24 17:10:00	10	25
10028	S45678	Madurai	Chennai	2021-05-30 10:00:00	2021-05-30 17:10:00	10	25
10029	S46423	Madurai	Chennai	2021-06-23 13:30:00	2021-06-23 20:05:00	10	30
10030	S46423	Madurai	Chennai	2021-05-30 13:30:00	2021-05-30 20:05:00	9	30

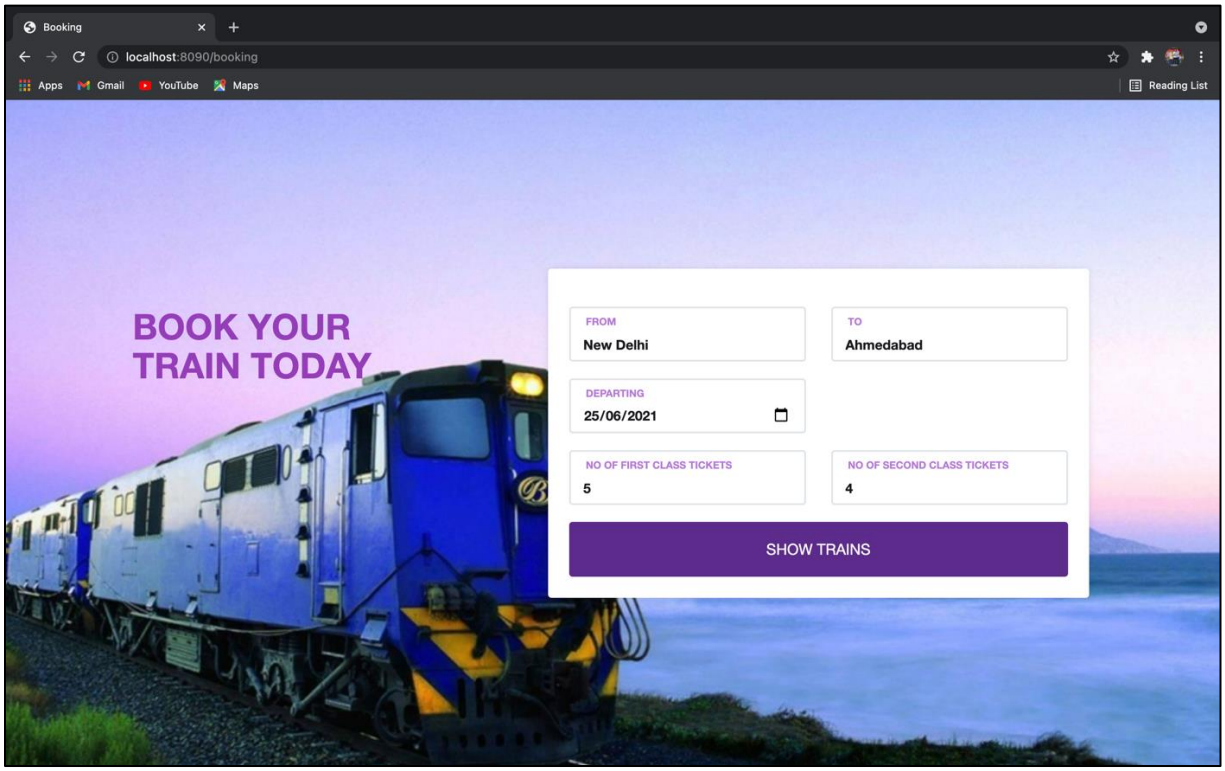
After booking the ticket:

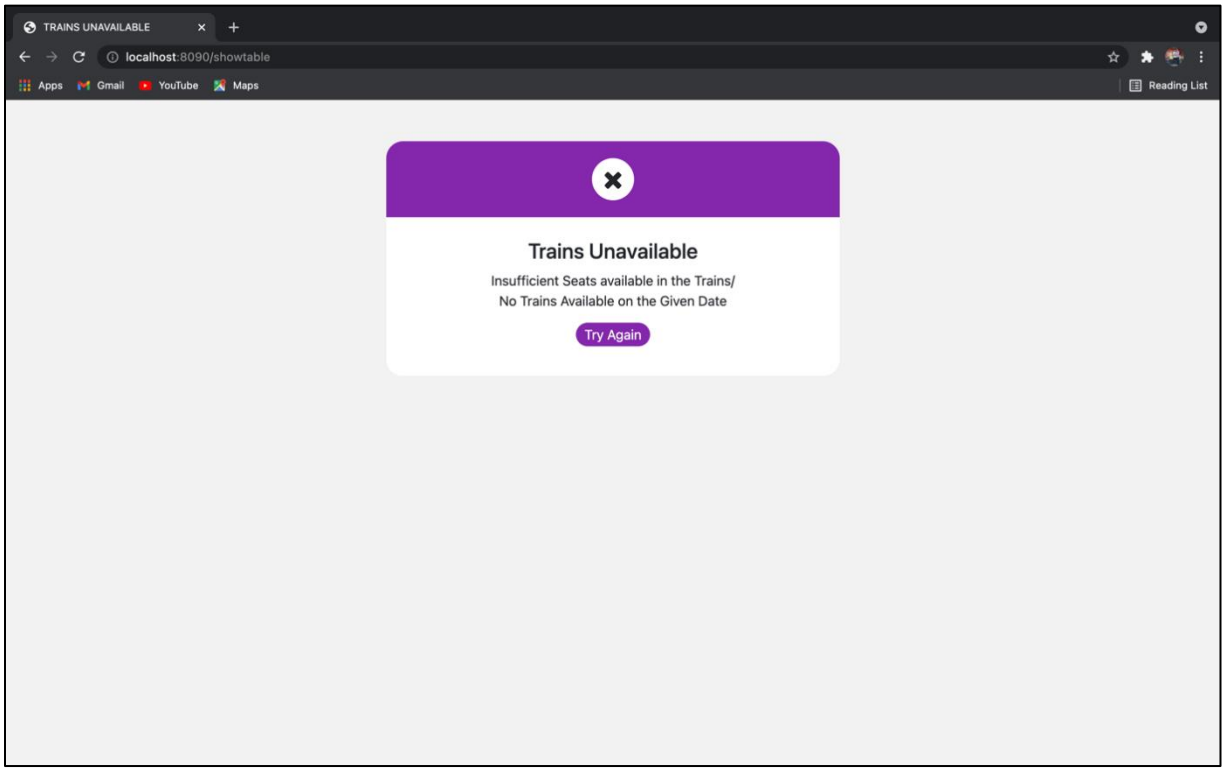
```
mysql> SELECT * FROM EXPRESS;
```

EXPRESS_NO	TRAIN_NO	FROMM	TOO	STARTING_TIME	ARRIVAL_TIME	FC_SEATS_REMAINING	SC_SEATS_REMAINING
10001	E15698	Kolkata	Hyderabad	2021-06-13 08:30:00	2021-06-14 05:29:00	10	20
10002	E15698	Kolkata	Hyderabad	2021-06-15 10:30:00	2021-06-16 07:29:00	10	20
10003	E15698	Kolkata	Hyderabad	2021-05-29 10:30:00	2021-05-29 07:29:00	10	20
10004	E16742	Kolkata	Hyderabad	2021-06-13 20:30:00	2021-06-14 12:29:00	15	25
10005	E16742	Kolkata	Hyderabad	2021-05-29 20:30:00	2021-05-30 12:29:00	13	21
10006	E17431	Kolkata	Hyderabad	2021-06-13 10:50:00	2021-06-14 08:45:00	20	30
10007	E17431	Kolkata	Hyderabad	2021-06-15 12:50:00	2021-06-16 10:45:00	20	30
10008	E18156	Kolkata	Hyderabad	2021-06-13 02:30:00	2021-06-14 00:29:00	10	20
10009	E18156	Kolkata	Hyderabad	2021-05-29 02:30:00	2021-05-30 00:29:00	10	20
10010	E19473	Kolkata	Hyderabad	2021-06-13 13:00:01	2021-06-14 14:45:01	10	15
10011	E19473	Kolkata	Hyderabad	2021-06-15 13:00:01	2021-06-16 14:45:01	10	15
10012	N21023	Delhi	Bangalore	2021-06-15 18:30:00	2021-06-17 13:00:00	11	20
10013	N21023	Delhi	Bangalore	2021-06-17 15:30:00	2021-06-19 12:00:00	11	20
10014	N24563	Delhi	Bangalore	2021-06-15 08:30:00	2021-06-17 02:00:00	15	30
10015	N24563	Delhi	Bangalore	2021-06-17 08:30:00	2021-06-19 02:00:00	15	30
10016	N25931	Delhi	Bangalore	2021-06-15 00:30:00	2021-06-16 23:30:00	10	30
10017	N26781	Delhi	Bangalore	2021-06-15 13:30:00	2021-06-17 11:25:00	10	25
10018	N26781	Delhi	Bangalore	2021-06-17 10:30:00	2021-06-19 09:25:00	10	25
10019	N36913	Rameshwaram	Varanasi	2021-06-20 12:30:00	2021-06-23 03:30:00	15	30
10020	N36913	Rameshwaram	Varanasi	2021-06-21 12:30:00	2021-06-24 03:30:00	15	30
10021	N36913	Rameshwaram	Varanasi	2021-05-29 12:30:00	2021-06-01 03:30:00	13	30
10022	N36913	Rameshwaram	Varanasi	2021-06-23 08:30:00	2021-06-25 00:30:00	15	30
10023	S44592	Madurai	Chennai	2021-06-23 23:30:00	2021-06-24 05:29:00	15	20
10024	S44592	Madurai	Chennai	2021-06-24 13:30:00	2021-06-24 21:29:00	15	20
10025	S45678	Madurai	Chennai	2021-06-23 10:00:00	2021-06-23 17:10:00	10	25
10026	S45678	Madurai	Chennai	2021-05-30 10:00:00	2021-05-30 17:10:00	10	25
10027	S45678	Madurai	Chennai	2021-06-24 10:00:00	2021-06-24 17:10:00	10	25
10028	S45678	Madurai	Chennai	2021-05-30 10:00:00	2021-05-30 17:10:00	10	25
10029	S46423	Madurai	Chennai	2021-06-23 13:30:00	2021-06-23 20:05:00	5	26
10030	S46423	Madurai	Chennai	2021-05-30 13:30:00	2021-05-30 20:05:00	9	30
10031	S47365	Madurai	Chennai	2021-06-23 23:30:00	2021-06-23 05:29:00	15	25

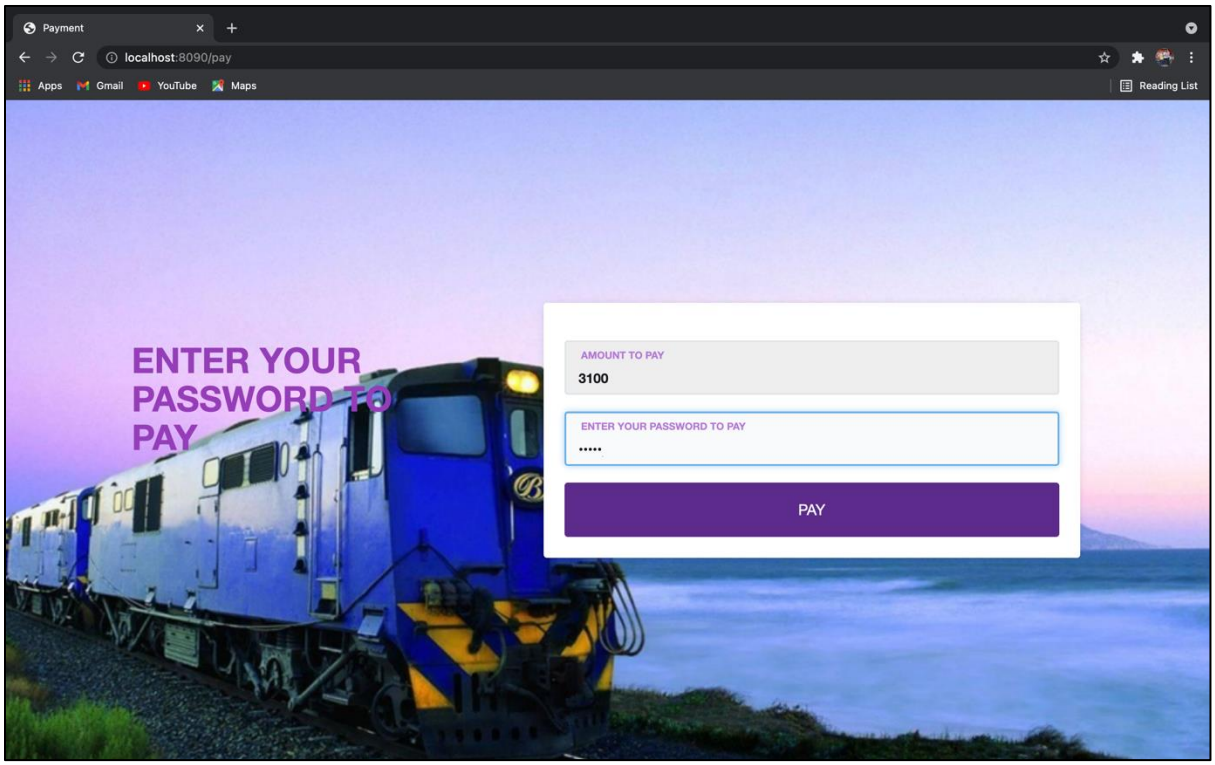
Unsuccessful Bookings

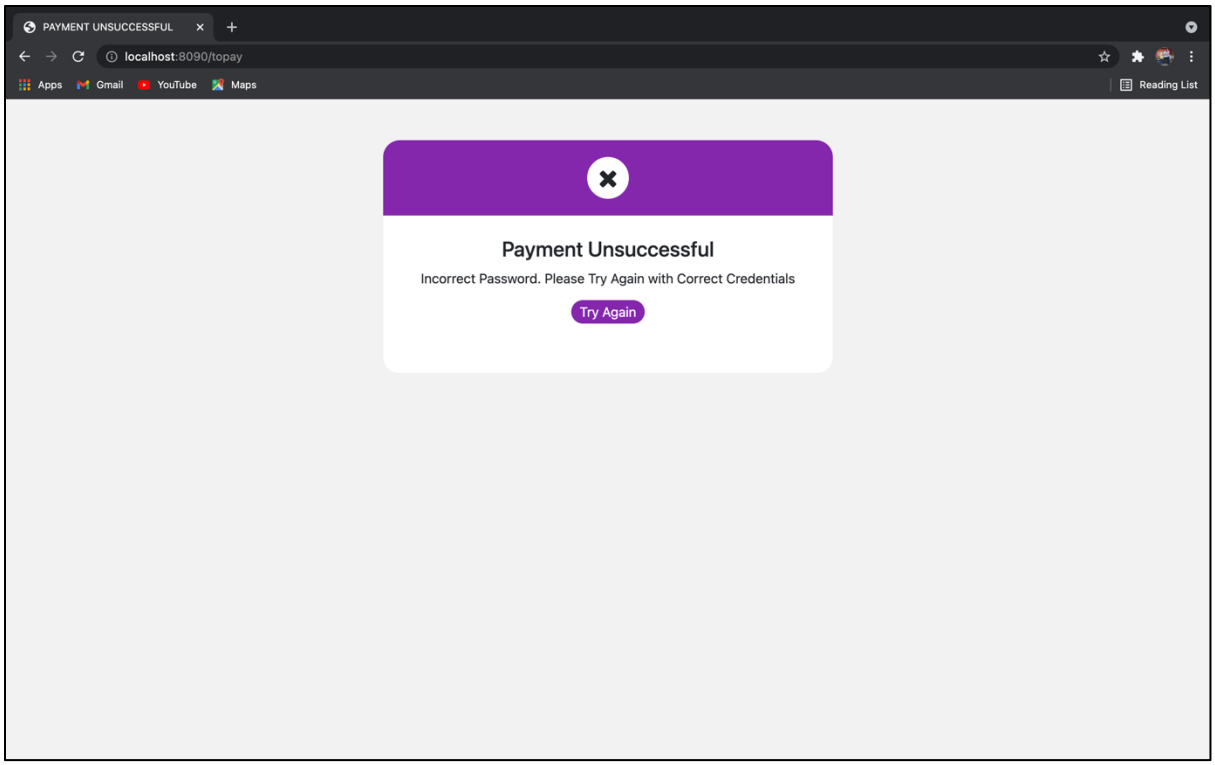
a) Train data unavailable:



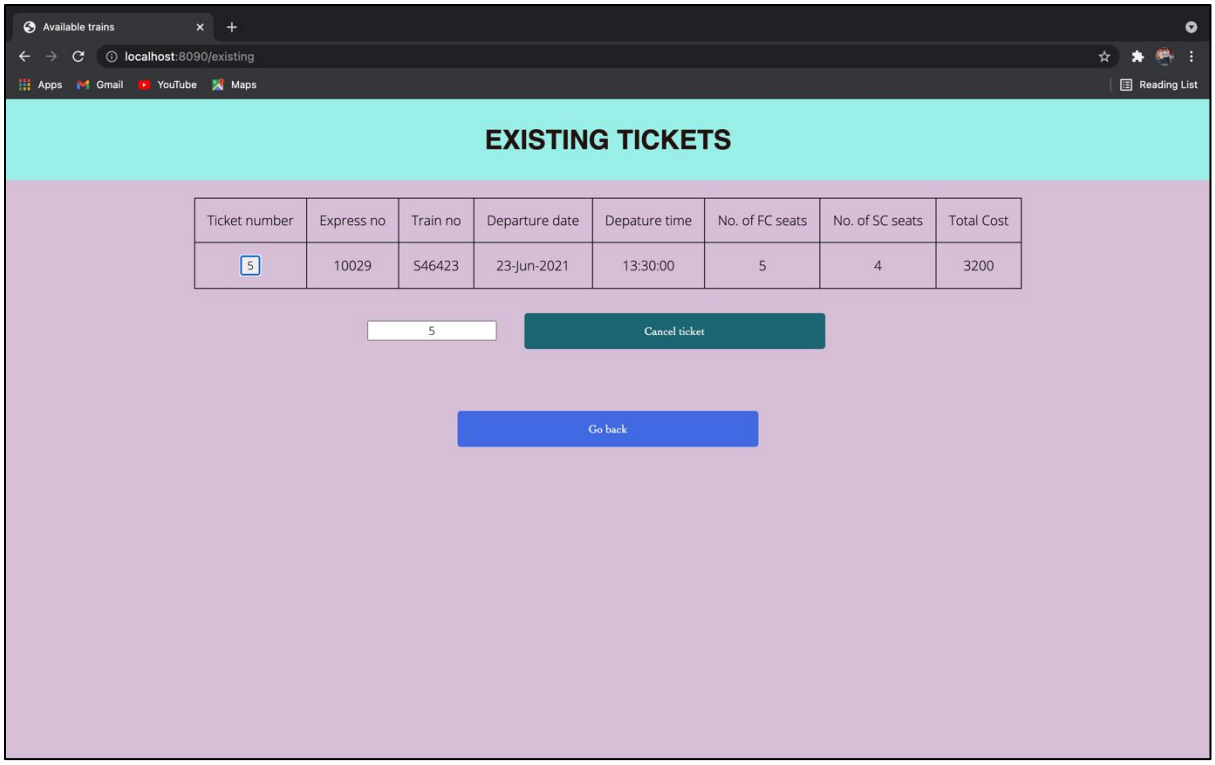


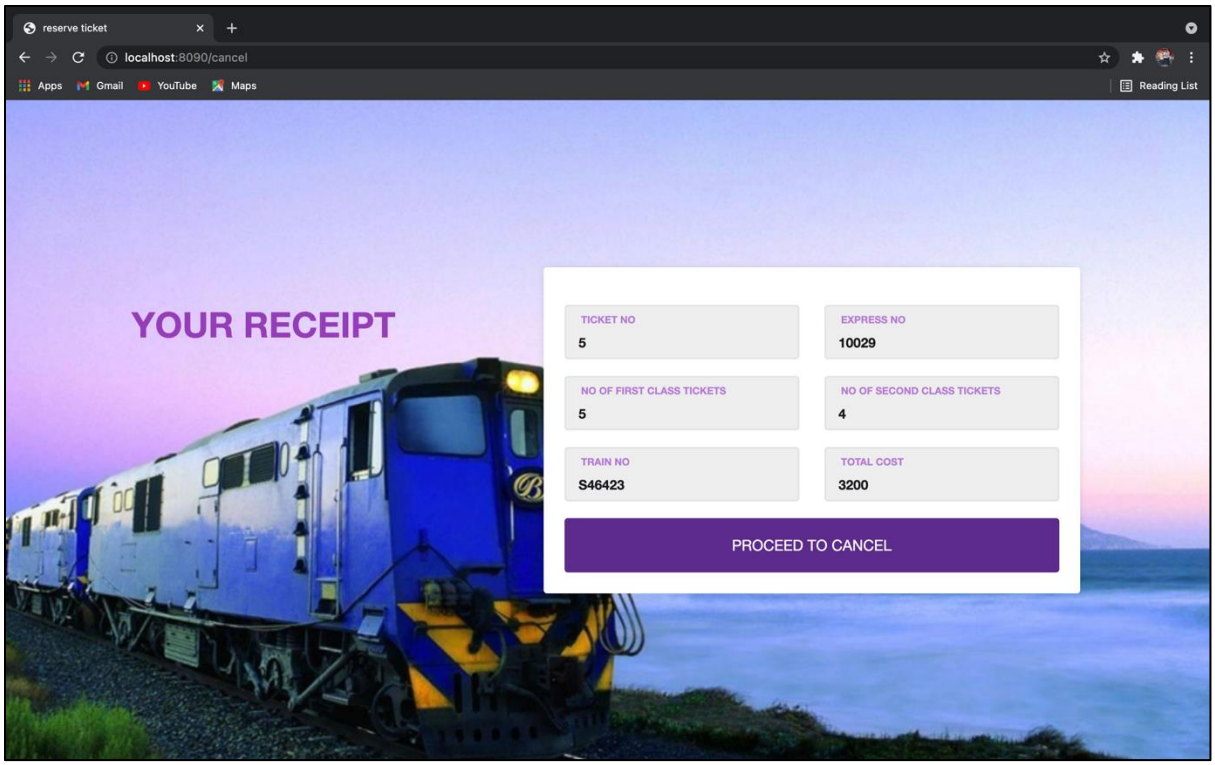
b) Incorrect password during payment process:

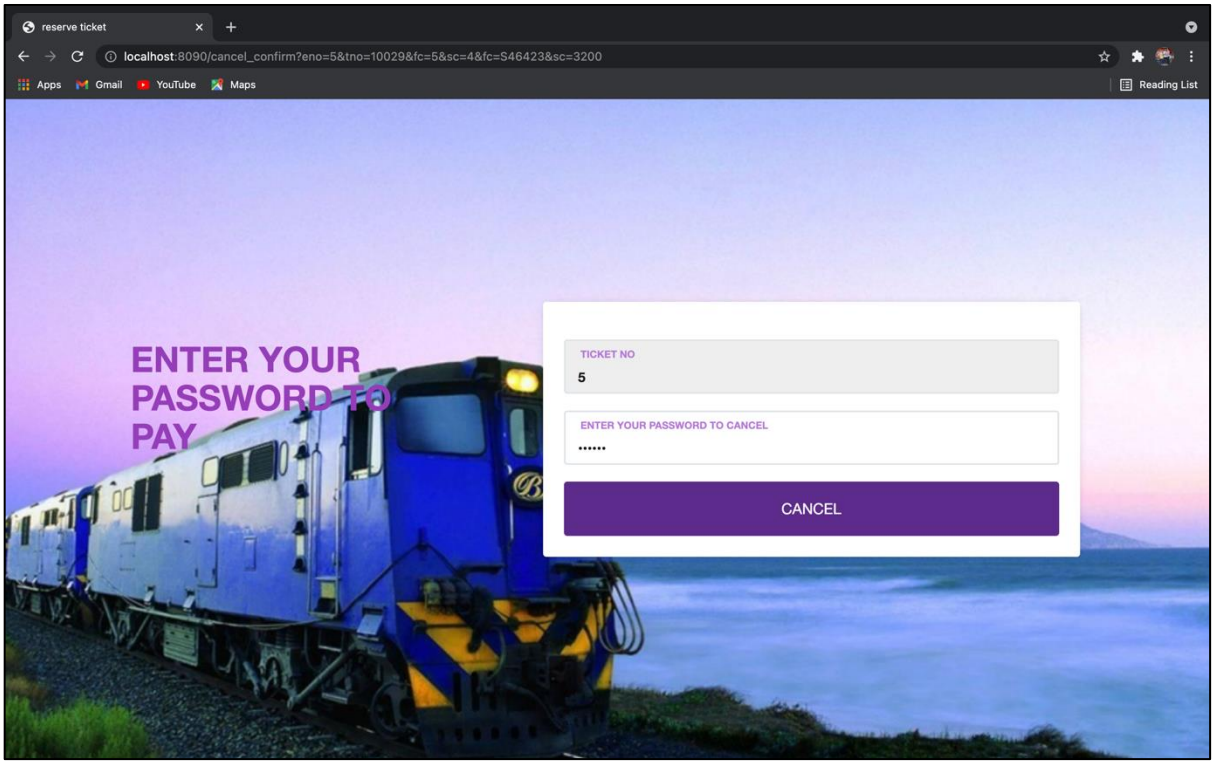


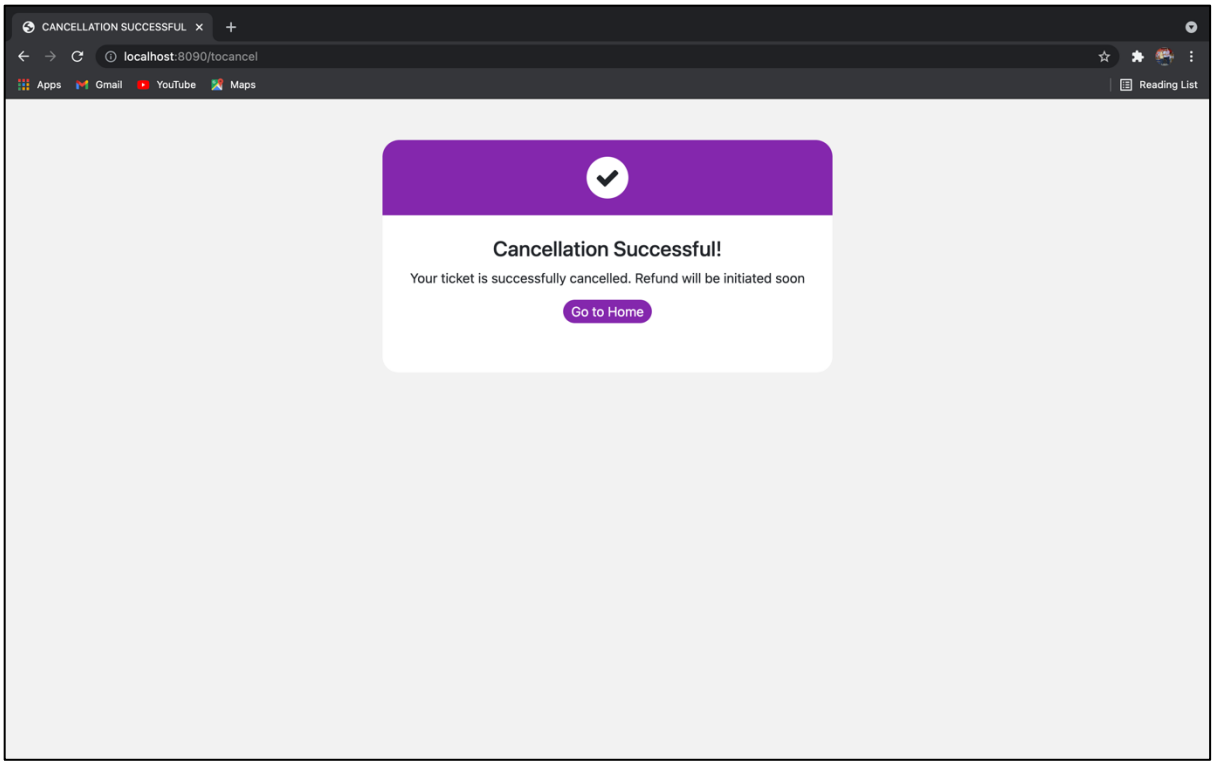


4. View existing tickets and cancellation

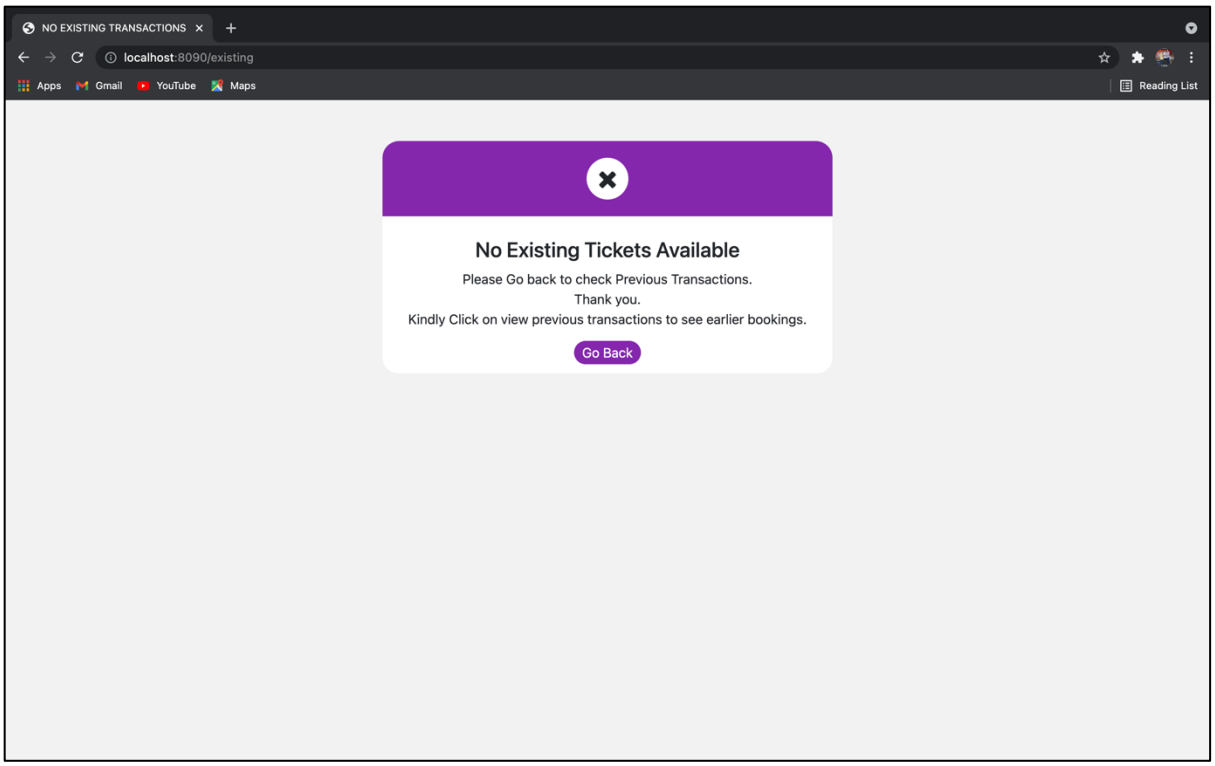









```
mysql> select * from booking where user_id = 'testuser01';
Empty set (0.00 sec)
```

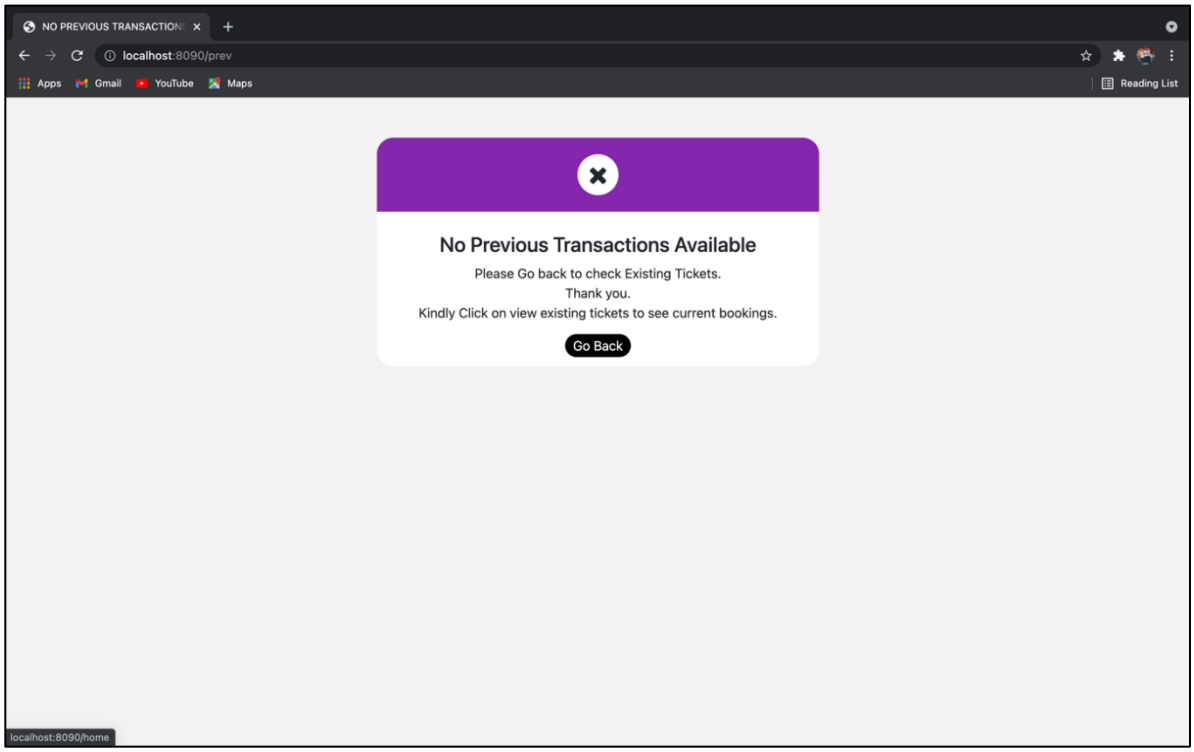


The trigger acts on the EXPRESS relation upon ticket cancellation as show below:

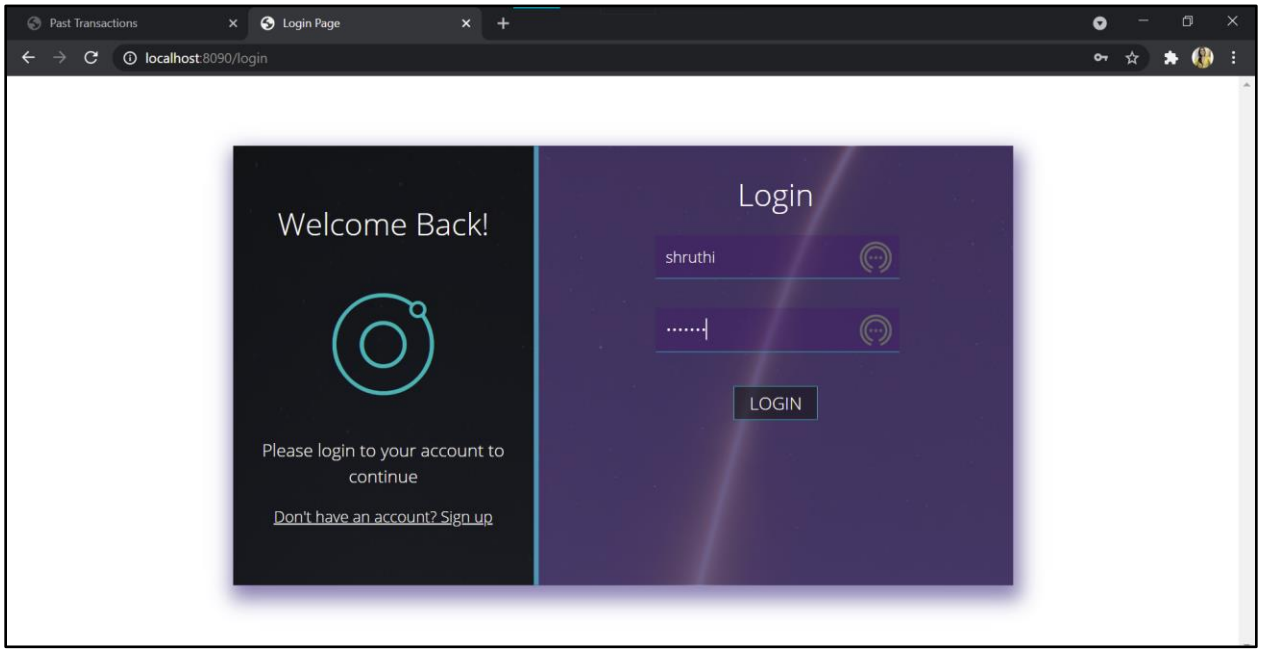
```
mysql> SELECT * FROM EXPRESS;
+------------+----------+-------------+-----------+---------------------+---------------------+--------------------+--------------------+
| EXPRESS_NO | TRAIN_NO | FROMM       | TOO       | STARTING_TIME       | ARRIVAL_TIME        | FC_SEATS_REMAINING | SC_SEATS_REMAINING |
+------------+----------+-------------+-----------+---------------------+---------------------+--------------------+--------------------+
|      10001 | E15698   | Kolkata     | Hyderabad | 2021-06-13 08:30:00 | 2021-06-14 05:29:00 |                 10 |                 20 |
|      10002 | E15698   | Kolkata     | Hyderabad | 2021-06-15 10:30:00 | 2021-06-16 07:29:00 |                 10 |                 20 |
|      10003 | E15698   | Kolkata     | Hyderabad | 2021-05-29 10:30:00 | 2021-05-29 07:29:00 |                 10 |                 20 |
|      10004 | E16742   | Kolkata     | Hyderabad | 2021-06-13 20:30:00 | 2021-06-14 12:29:00 |                 15 |                 25 |
|      10005 | E16742   | Kolkata     | Hyderabad | 2021-05-29 20:30:00 | 2021-05-30 12:29:00 |                 13 |                 21 |
|      10006 | E17431   | Kolkata     | Hyderabad | 2021-06-13 10:50:00 | 2021-06-14 08:45:00 |                 20 |                 30 |
|      10007 | E17431   | Kolkata     | Hyderabad | 2021-06-15 12:50:00 | 2021-06-16 10:45:00 |                 20 |                 30 |
|      10008 | E18156   | Kolkata     | Hyderabad | 2021-06-13 02:30:00 | 2021-06-14 00:29:00 |                 10 |                 20 |
|      10009 | E18156   | Kolkata     | Hyderabad | 2021-05-29 02:30:00 | 2021-05-30 00:29:00 |                 10 |                 20 |
|      10010 | E19473   | Kolkata     | Hyderabad | 2021-06-13 13:00:01 | 2021-06-14 14:45:01 |                 10 |                 15 |
|      10011 | E19473   | Kolkata     | Hyderabad | 2021-06-15 13:00:01 | 2021-06-16 14:45:01 |                 10 |                 15 |
|      10012 | N21023   | Delhi       | Bangalore | 2021-06-15 18:30:00 | 2021-06-17 13:00:00 |                 11 |                 20 |
|      10013 | N21023   | Delhi       | Bangalore | 2021-06-17 15:30:00 | 2021-06-19 12:00:00 |                 11 |                 20 |
|      10014 | N24563   | Delhi       | Bangalore | 2021-06-15 08:30:00 | 2021-06-17 02:00:00 |                 15 |                 30 |
|      10015 | N24563   | Delhi       | Bangalore | 2021-06-17 08:30:00 | 2021-06-19 02:00:00 |                 15 |                 30 |
|      10016 | N25931   | Delhi       | Bangalore | 2021-06-15 00:30:00 | 2021-06-16 23:30:00 |                 10 |                 30 |
|      10017 | N26781   | Delhi       | Bangalore | 2021-06-15 13:30:00 | 2021-06-17 11:25:00 |                 10 |                 25 |
|      10018 | N26781   | Delhi       | Bangalore | 2021-06-17 10:30:00 | 2021-06-19 09:25:00 |                 10 |                 25 |
|      10019 | N36913   | Rameshwaram | Varanasi  | 2021-06-20 12:30:00 | 2021-06-23 03:30:00 |                 15 |                 30 |
|      10020 | N36913   | Rameshwaram | Varanasi  | 2021-06-21 12:30:00 | 2021-06-24 03:30:00 |                 15 |                 30 |
|      10021 | N36913   | Rameshwaram | Varanasi  | 2021-05-29 12:30:00 | 2021-06-01 03:30:00 |                 13 |                 30 |
|      10022 | N36913   | Rameshwaram | Varanasi  | 2021-06-23 08:30:00 | 2021-06-25 00:30:00 |                 15 |                 30 |
|      10023 | S44592   | Madurai     | Chennai   | 2021-06-23 23:30:00 | 2021-06-24 05:29:00 |                 15 |                 20 |
|      10024 | S44592   | Madurai     | Chennai   | 2021-06-24 13:30:00 | 2021-06-24 21:29:00 |                 15 |                 20 |
|      10025 | S45678   | Madurai     | Chennai   | 2021-06-23 10:00:00 | 2021-06-23 17:10:00 |                 10 |                 25 |
|      10026 | S45678   | Madurai     | Chennai   | 2021-05-30 10:00:00 | 2021-05-30 17:10:00 |                 10 |                 25 |
|      10027 | S45678   | Madurai     | Chennai   | 2021-06-24 10:00:00 | 2021-06-24 17:10:00 |                 10 |                 25 |
|      10028 | S45678   | Madurai     | Chennai   | 2021-05-30 10:00:00 | 2021-05-30 17:10:00 |                 10 |                 25 |
|      10029 | S46423   | Madurai     | Chennai   | 2021-06-23 13:30:00 | 2021-06-23 20:05:00 |                 10 |                 30 |
|      10030 | S46423   | Madurai     | Chennai   | 2021-05-30 13:30:00 | 2021-05-30 20:05:00 |                  9 |                 30 |
```

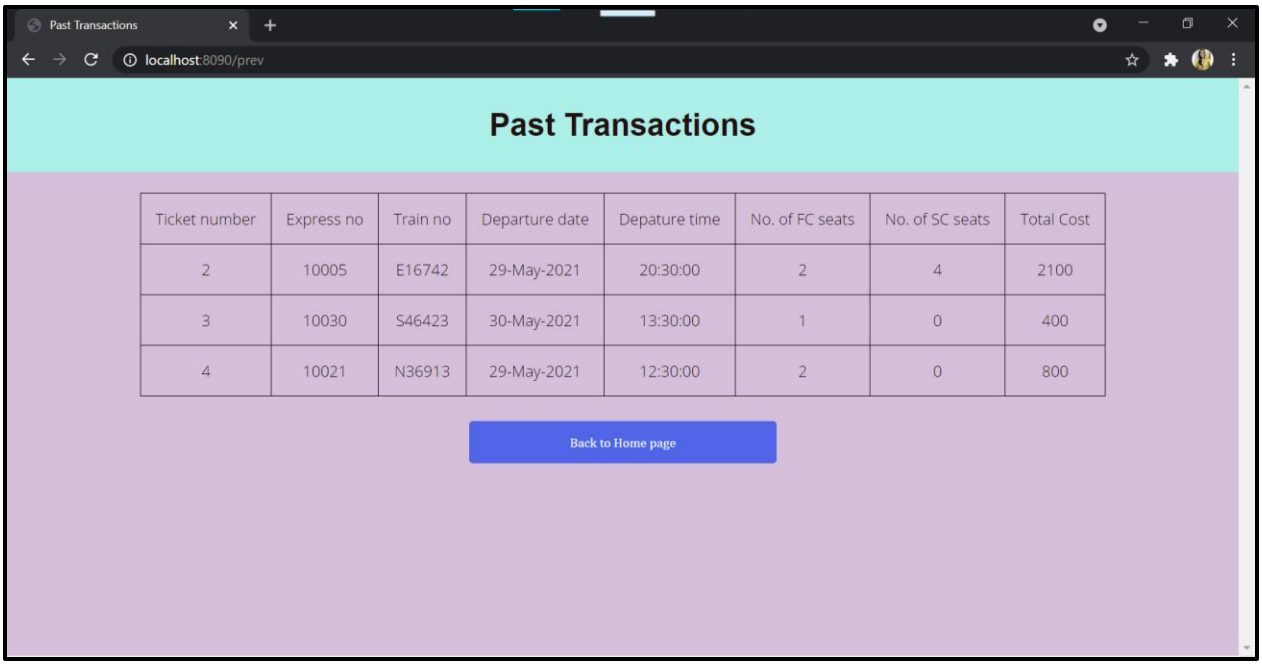
5. Previous Transactions

a) Empty set (because no previous successful transactions were made)



b) Previous transactions available (using another user login)



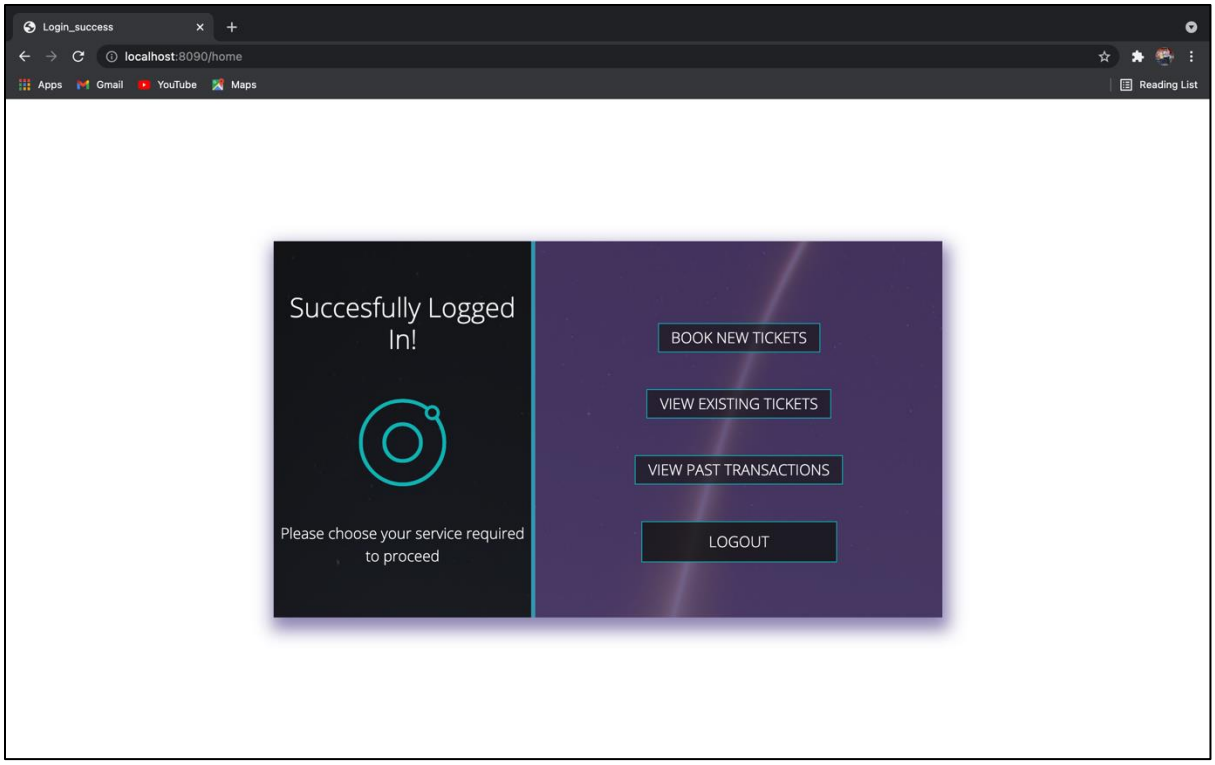


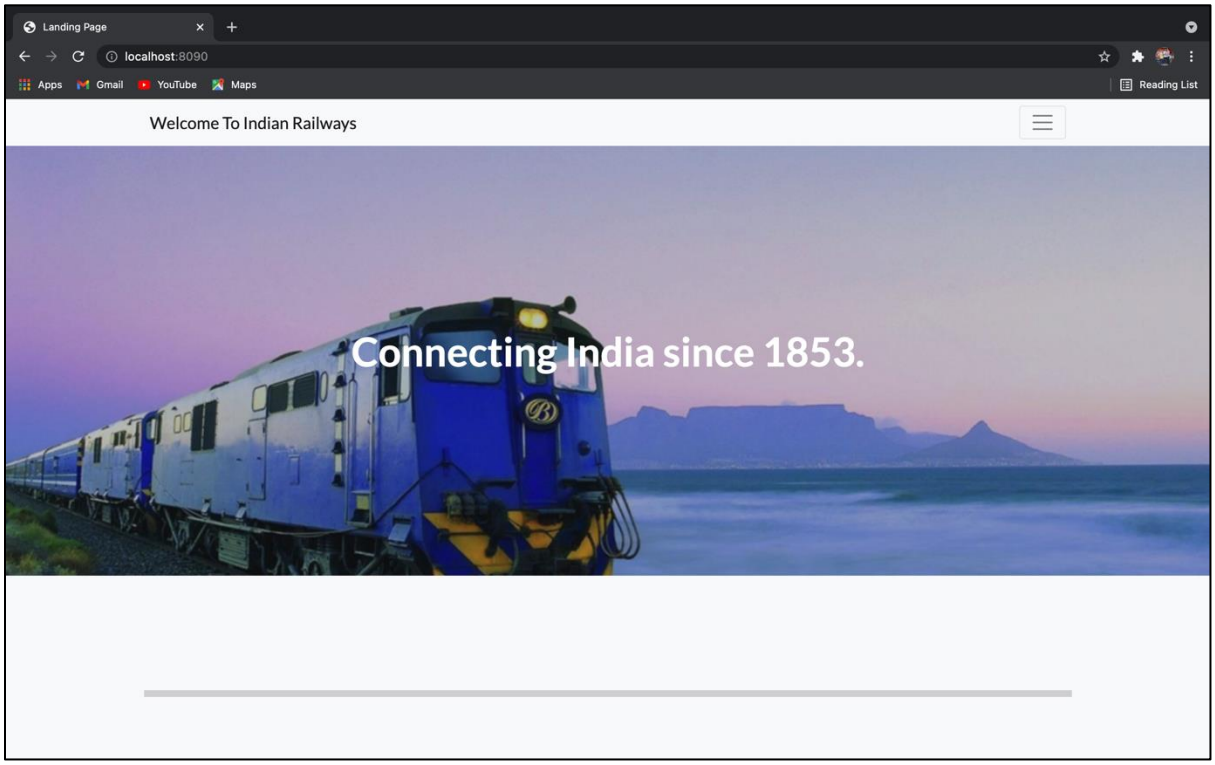

```
mysql> SELECT * FROM BOOKING;
+------------+------------+---------+----------+----------------+----------------+------------+
| BOOKING_ID | EXPRESS_NO | USER_ID | TRAIN_NO | TOTAL_FC_SEATS | TOTAL_SC_SEATS | TOTAL_COST |
+------------+------------+---------+----------+----------------+----------------+------------+
|          2 |      10005 | shruthi | E16742   |              2 |              4 |       2100 |
|          3 |      10030 | shruthi | S46423   |              1 |              0 |        400 |
|          4 |      10021 | shruthi | N36913   |              2 |              0 |        800 |
+------------+------------+---------+----------+----------------+----------------+------------+
3 rows in set (0.00 sec)
```

The presentview is shown below. This View is created and replaced from the insertion and deletion of tuples in the booking relation.

```
mysql> SELECT * FROM PRESENTVIEW;
+------------+------------+---------+----------+----------------+----------------+----------------+----------------+------------+
| BOOKING_ID | EXPRESS_NO | USER_ID | TRAIN_NO | DEPARTURE_DATE | DEPARTURE_TIME | TOTAL_FC_SEATS | TOTAL_SC_SEATS | TOTAL_COST |
+------------+------------+---------+----------+----------------+----------------+----------------+----------------+------------+
|          2 |      10005 | shruthi | E16742   | 2021-05-29     | 20:30:00       |              2 |              4 |       2100 |
|          3 |      10030 | shruthi | S46423   | 2021-05-30     | 13:30:00       |              1 |              0 |        400 |
|          4 |      10021 | shruthi | N36913   | 2021-05-29     | 12:30:00       |              2 |              0 |        800 |
+------------+------------+---------+----------+----------------+----------------+----------------+----------------+------------+
3 rows in set (0.00 sec)
```

6. Logout


Login_success
localhost:8090/home
Apps
Gmail
YouTube
Maps
Reading List
Succesfully Logged In!
Please choose your service required to proceed
BOOK NEW TICKETS
VIEW EXISTING TICKETS
VIEW PAST TRANSACTIONS
LOGOUT


Landing Page
localhost:8090
Apps
Gmail
YouTube
Maps
Reading List
Welcome To Indian Railways
Connecting India since 1853.

CONCLUSION

Thus, the user interface is connected to the database through the server. Data is entered into the tables and retrieved from the tables of the database based on requirements of the user with the help of the server and user interface.

Hence this application has been successfully developed to facilitate booking and cancellation of tickets using a simple and intuitive user interface.

For reference, please check our website at https://railway.synorita.com/

Source code for the same can be found at:
https://github.com/Shruthi-Gopal/RAILWAY-RESERVATION--DBMS-PROJECT.git

REFERENCES

https://www.w3schools.com/nodejs/

https://www.w3schools.com/nodejs/nodejs_mysql.asp

https://www.w3schools.com/html/

https://www.w3schools.com/css/

https://heynode.com/blog/2020-04/salt-and-hash-passwords-bcrypt

https://nodejs.org/en/docs/

https://www.npmjs.com/package/bcrypt

https://stackoverflow.com/questions/

https://www.geeksforgeeks.org/nodejs-tutorials/