

5/30/2021

DBMS PROJECT REPORT

PROJECT TITLE: RAILWAY RESERVATION

SUBMITTED BY:

SHRUTHI G

2019103061

ADITYA RAMACHANDRAN

2019103502

VARSHA SSKM

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	8
3	ER DIAGRAM	9
4	RELATIONAL SCHEMA	10
5	USER INTERFACE DESIGN	11
6	IMPLEMENTATION	17
7	RESULTS SNAPSHOTS	29
8	CONCLUSION	51
9	REFERENCES	51

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.

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.

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

SQL FILE TO CREATE RELATIONS AND SET UP VIEWS:

```
SQL > # Create_tables.sql
1  CREATE DATABASE railwaydb;
2  USE railwaydb;
3
4  CREATE TABLE TRAIN
5  (TRAIN_NO VARCHAR(15) PRIMARY KEY,
6  EXPRESS_NAME VARCHAR(20),
7  FC_SEATS INT,
8  SC_SEATS INT,
9  FC_PRICE INT,
10 SC_PRICE INT);
11
12 CREATE TABLE EXPRESS
13 (EXPRESS_NO INT NOT NULL AUTO_INCREMENT PRIMARY KEY ,
14 TRAIN_NO VARCHAR(15),
15 FROMM VARCHAR(20),
16 TOO VARCHAR(20),
17 STARTING_TIME TIMESTAMP,
18 ARRIVAL_TIME TIMESTAMP,
19 FC_SEATS_REMAINING INT,
20 SC_SEATS_REMAINING INT,
21 FOREIGN KEY (TRAIN_NO) REFERENCES TRAIN(TRAIN_NO)
22 );
23 ALTER TABLE EXPRESS AUTO_INCREMENT = 10001;
24
25 CREATE TABLE USER_DETAILS
26 (USER_ID VARCHAR(15) PRIMARY KEY,
27 USERNAME VARCHAR(20),
28 EMAIL_ID VARCHAR(25),
29 PASSWORD VARCHAR(200)
30 );
```

```
SQL > # existingView.sql
1  CREATE OR REPLACE VIEW PRESENTVIEW AS SELECT BOOKING.BOOKING_ID AS BOOKING_ID, BOOKING.EXPRESS_NO AS EXPRESS_NO,
2  BOOKING.USER_ID AS USER_ID, BOOKING.TRAIN_NO AS TRAIN_NO, DATE(EXPRESS.STARTING_TIME) AS DEPARTURE_DATE,
3  TIME(EXPRESS.STARTING_TIME) AS DEPARTURE_TIME, BOOKING.TOTAL_FC_SEATS AS TOTAL_FC_SEATS,
4  BOOKING.TOTAL_SC_SEATS AS TOTAL_SC_SEATS,
5  BOOKING.TOTAL_COST AS TOTAL_COST FROM BOOKING INNER JOIN EXPRESS ON (BOOKING.EXPRESS_NO=EXPRESS.EXPRESS_NO);
```

```
SQL > # express_view.sql
1  CREATE OR REPLACE VIEW EXPRESSVIEW AS SELECT EXPRESS_NO, TRAIN_NO, FROMM, TOO, DATE(STARTING_TIME) AS
2  DEPARTURE_DATE, TIME(STARTING_TIME) AS DEPARTURE_TIME FROM EXPRESS;
```

Running the files 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)
```

```
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)
```

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.

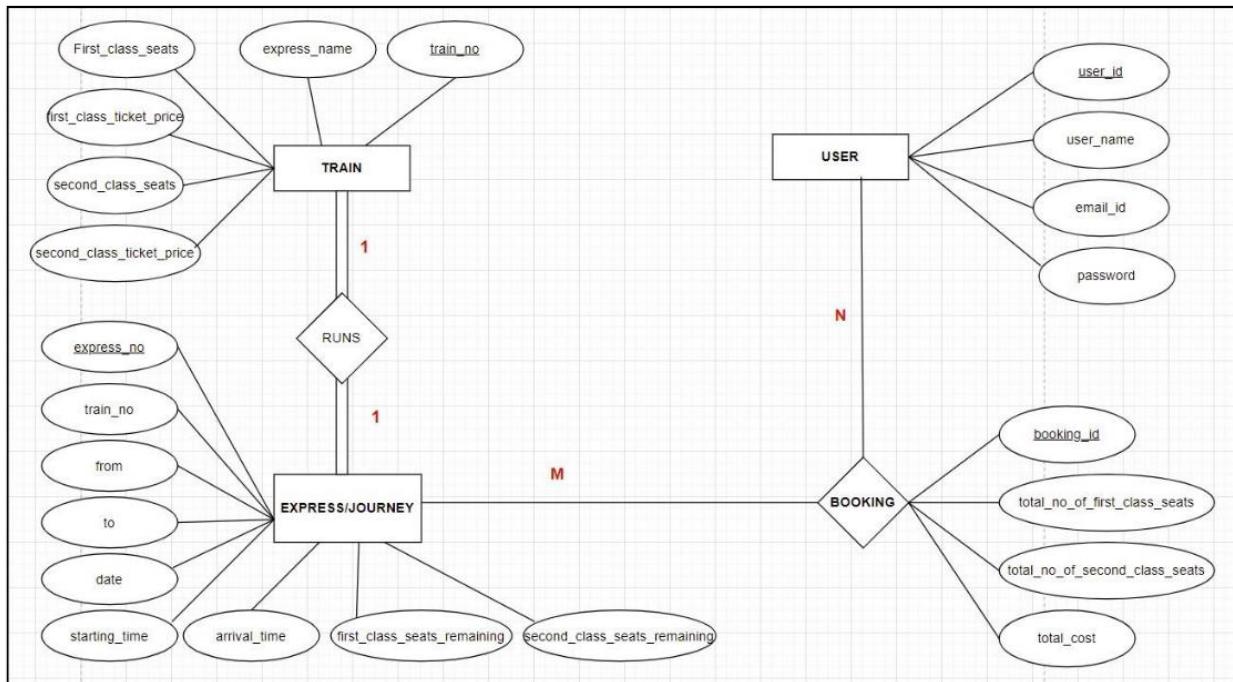
ER DIAGRAM OF THE DATABASE

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`.

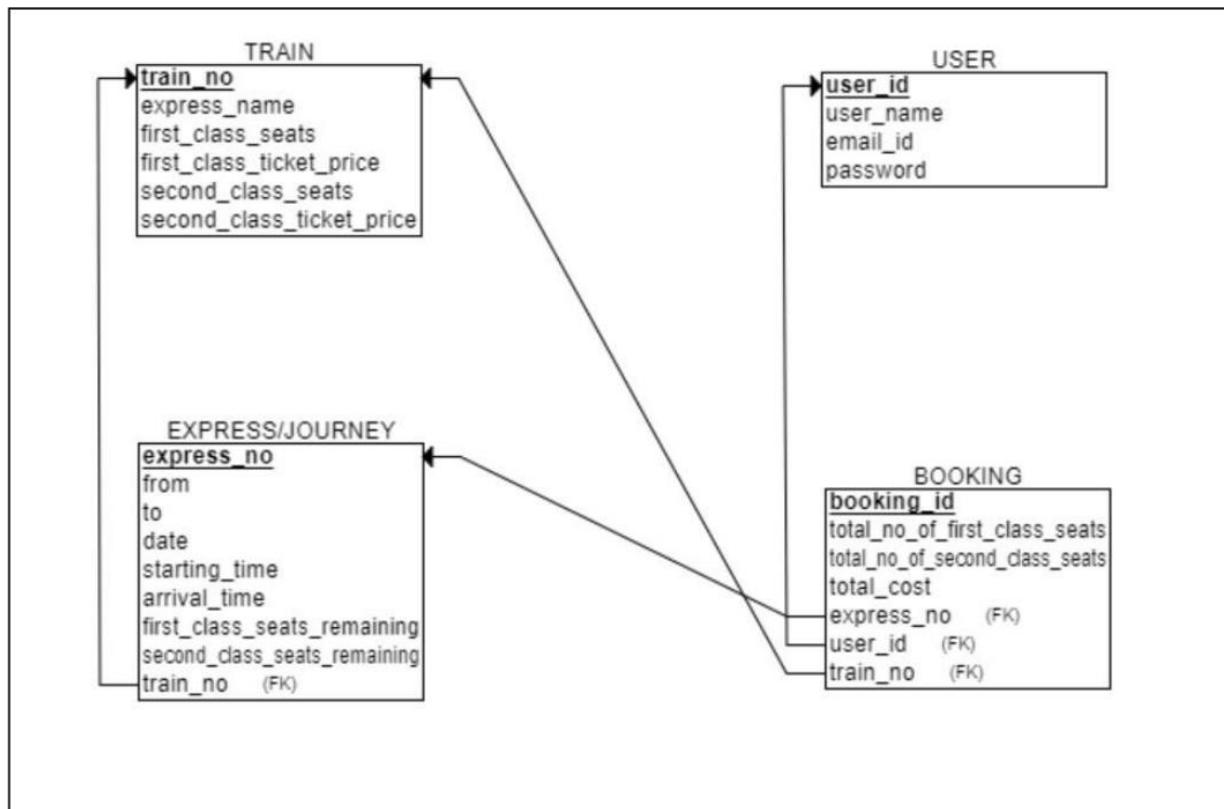
Express has attributes pertaining to a particular express like `express_no`, `train_no`, `from`, `to`, `date`, `starting_time`, `arrival_time` and some attributes which can be changed when tickets are booked and cancelled by the user like `first_class_seats_remaining` and `second_class_seats_remaining`.

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



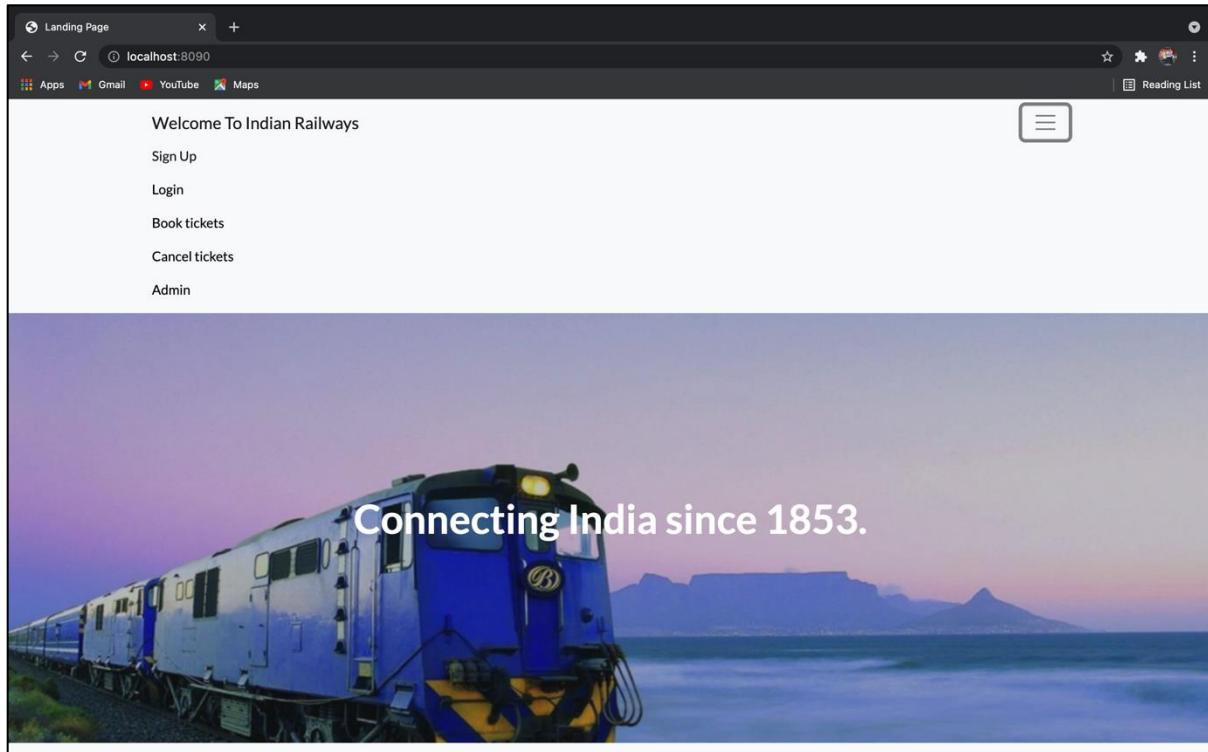
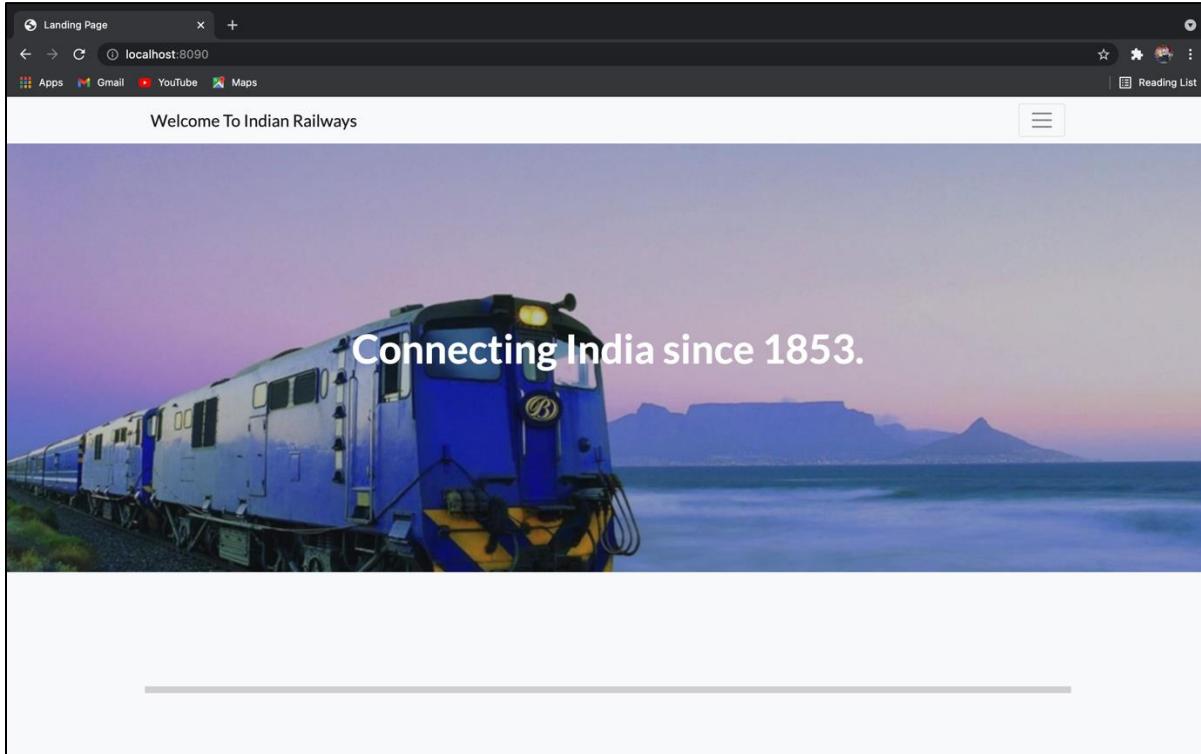
RELATIONAL SCHEMA OF THE DATABASE

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



USER INTERFACE DESIGN

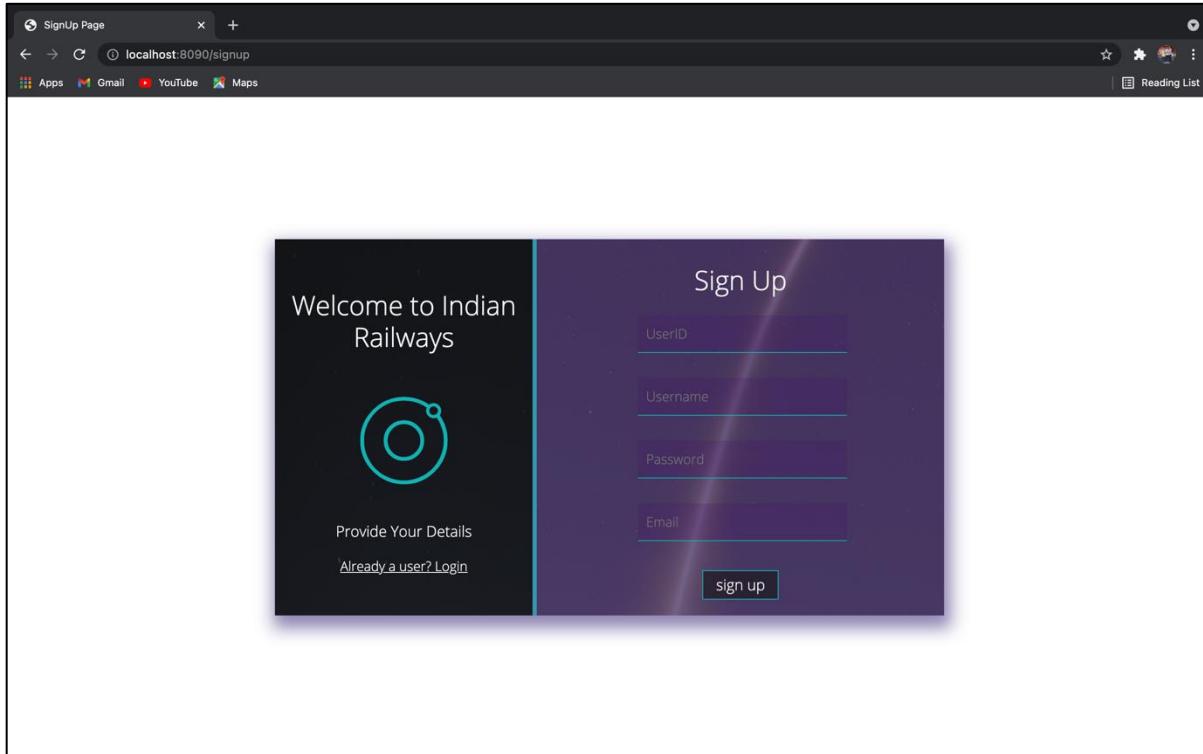
Landing page



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

1. Sign Up:

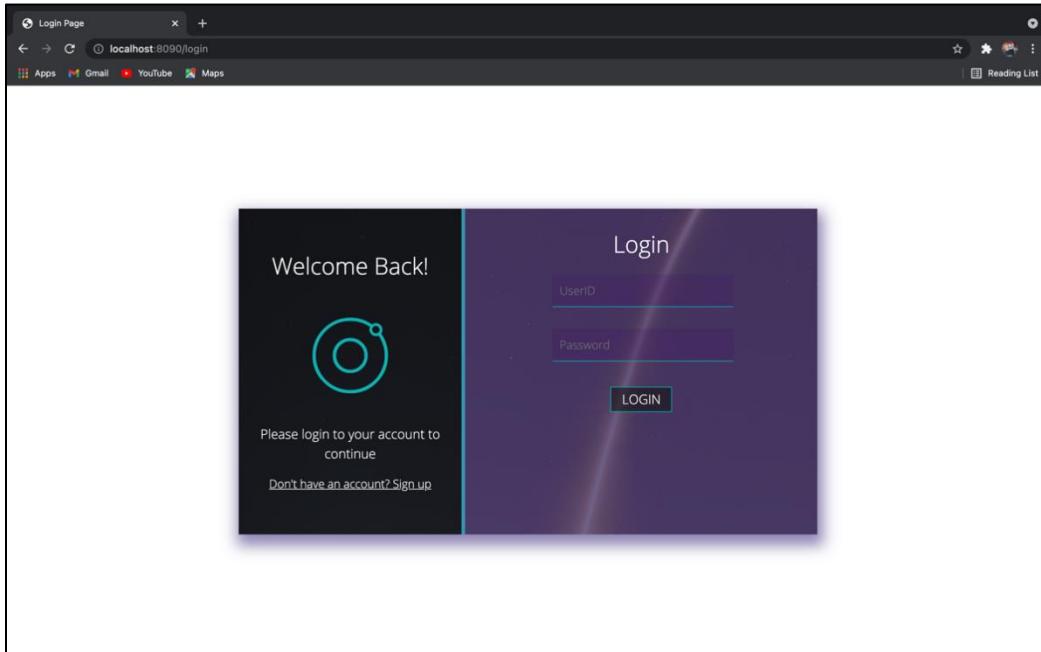
The area where the user can create a new user account to enjoy all the features of the application.



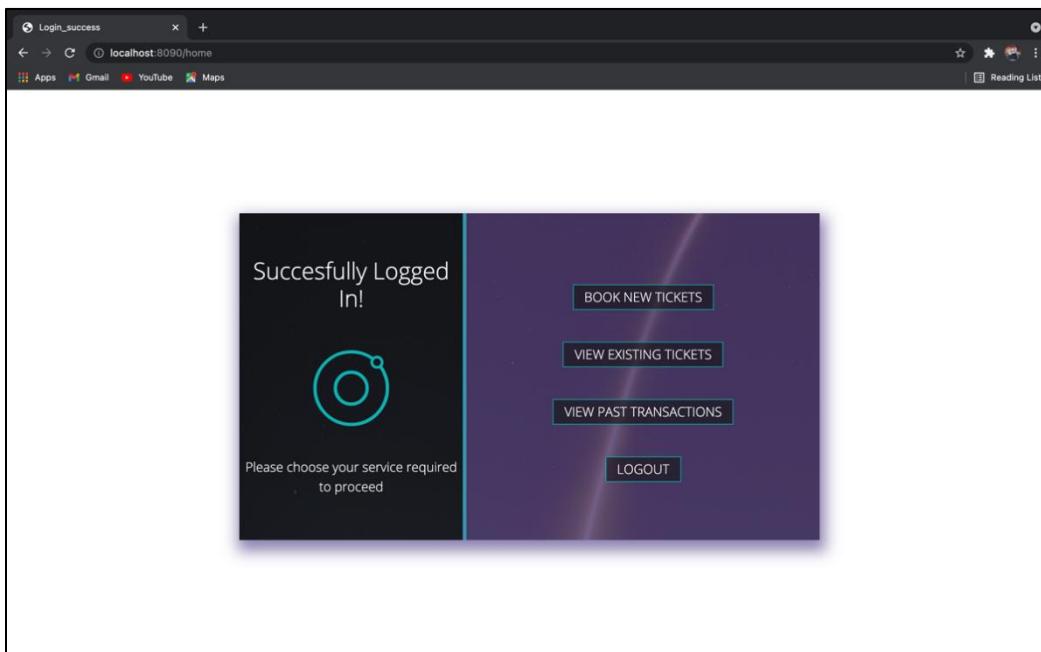
Here the user must enter UserID, Username, Password, and the Email address where the UserID must be unique. From here, he is redirected to the login page.

2. Login:

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

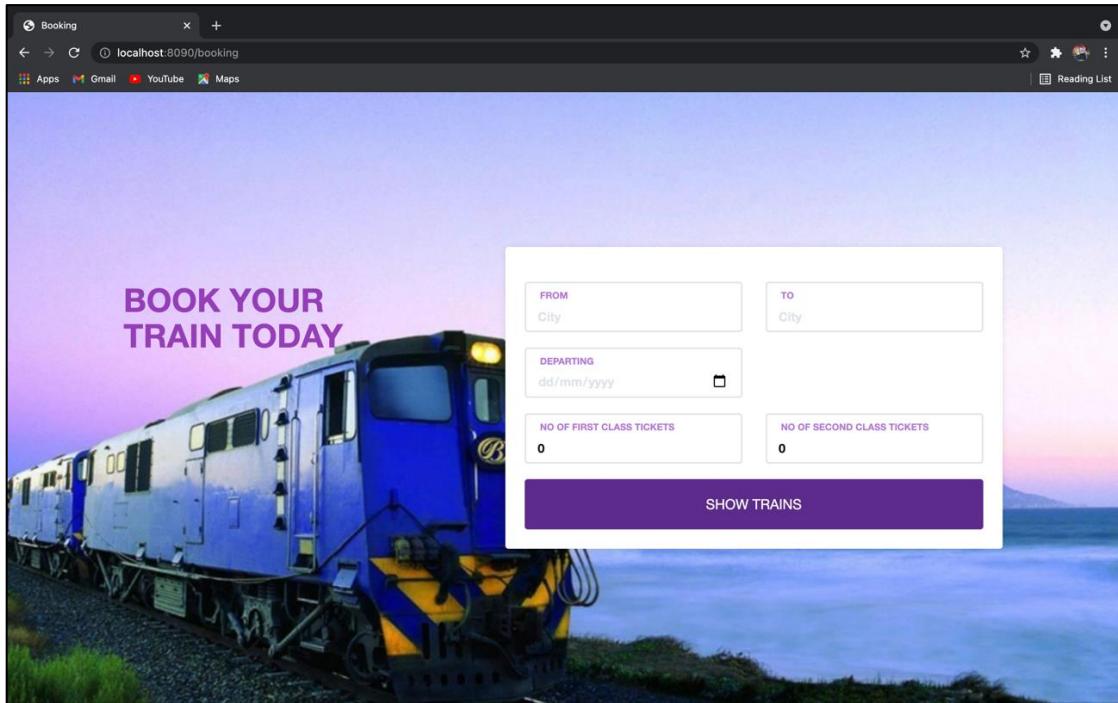


From here, the user will be redirected to the home page.



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.



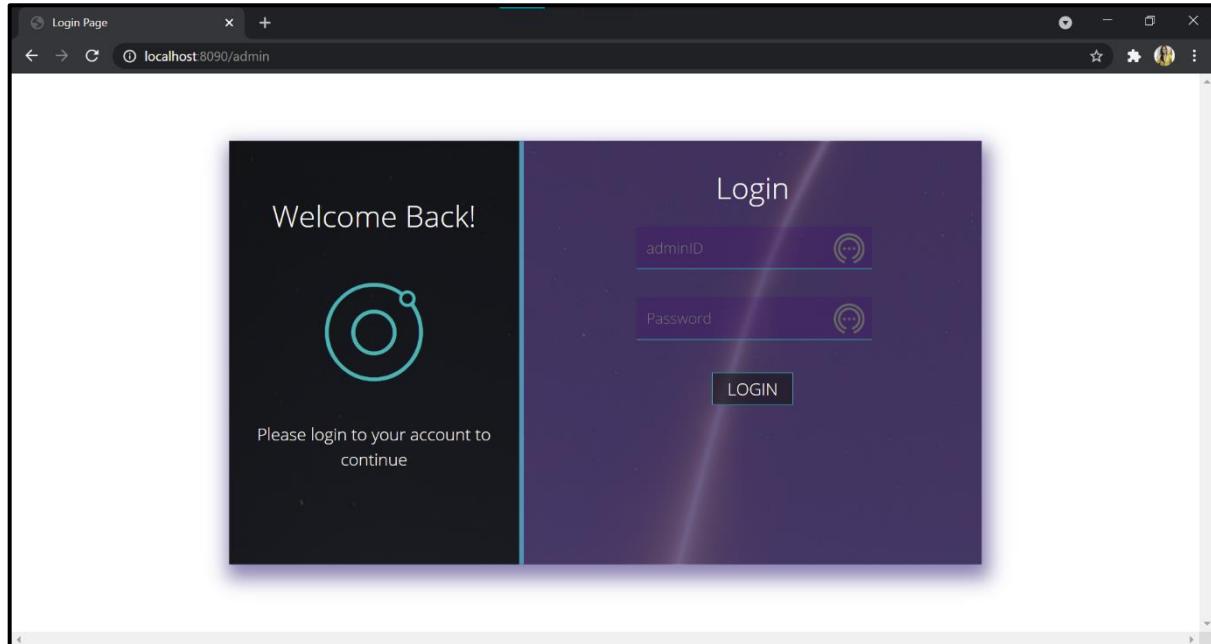
4. Cancel Tickets

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)

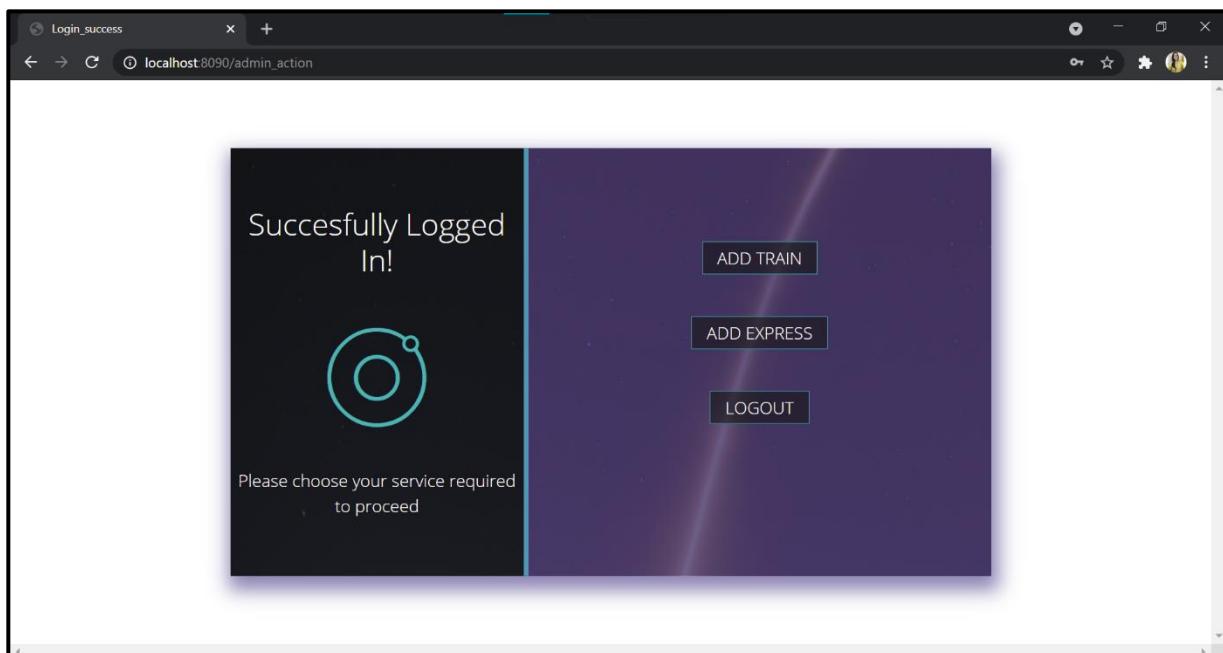
The following sections are available for the admin section of the application:

1. Login:

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

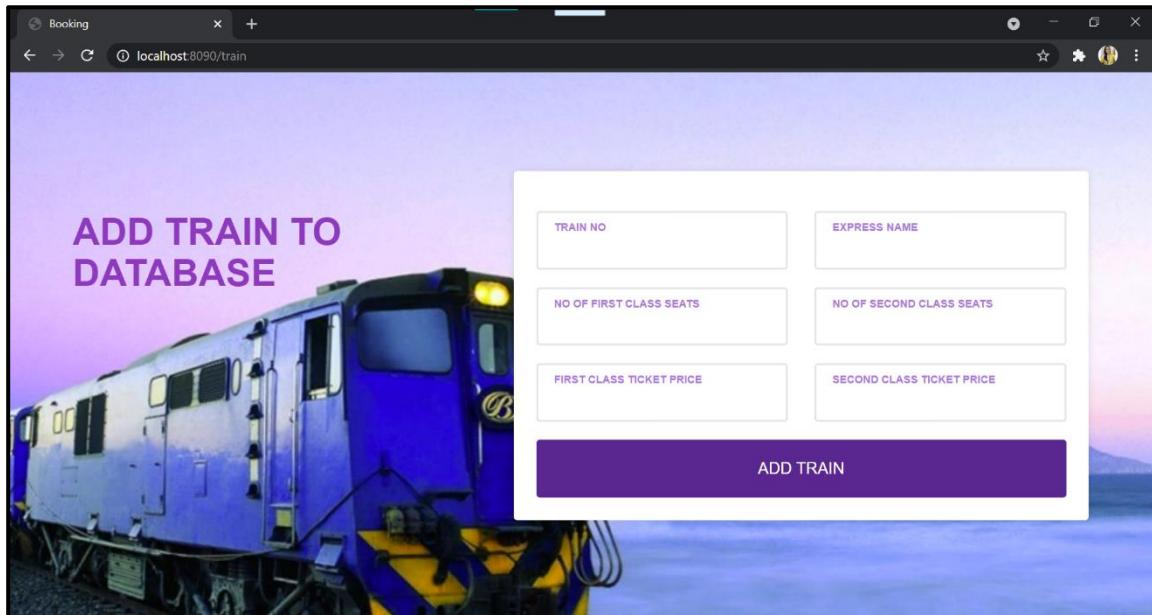


From here, the admin is directed to admin_actions page.



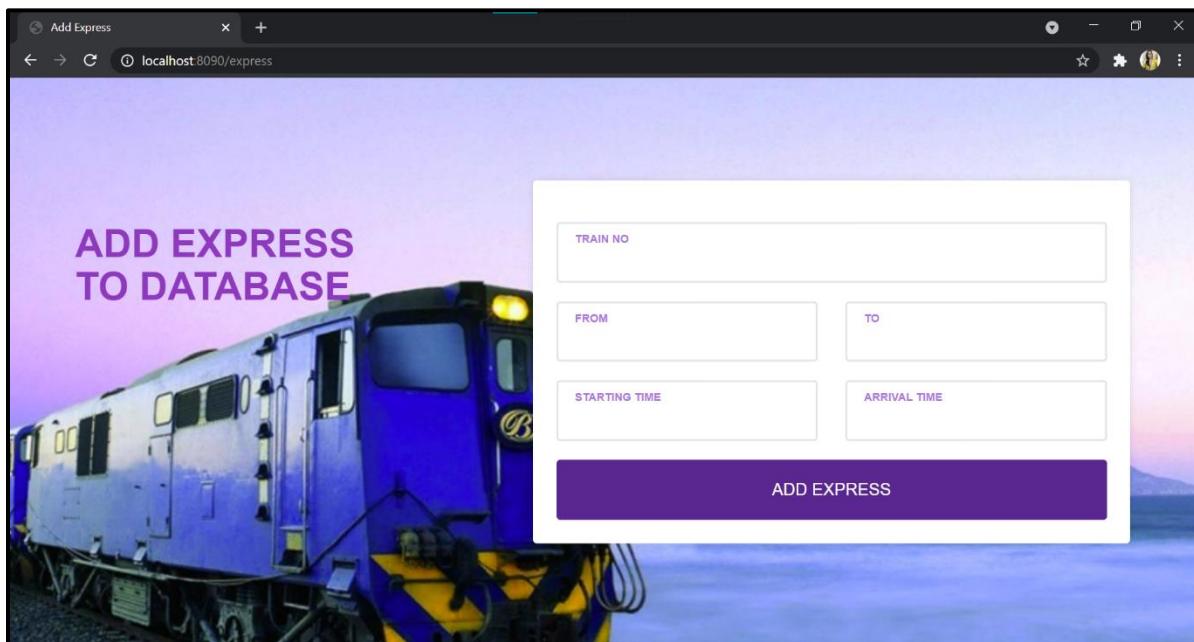
2. Add Trains:

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



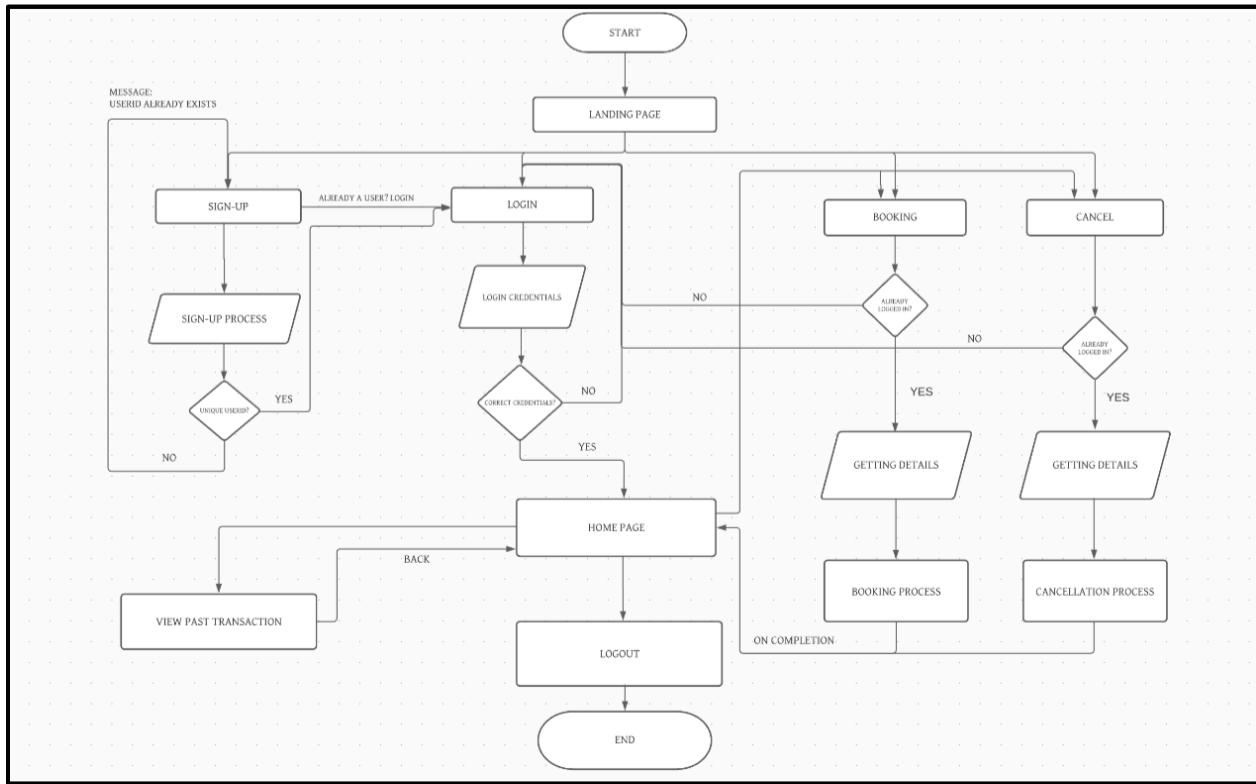
3. Add Express:

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



IMPLEMENTATION

General flowchart depicting the flow of control inside the program.



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:

```

81 app.post('/addtrain', function (req, res) { █
82   if (req.session.loggedin) {
83     con.connect(function (err) { █
84       var sql =
85         "CALL ADDTRAIN('" +
86         req.body.trainno +
87         "','" +
88         req.body.expressno +
89         "','" +
90         req.body.fcseats +
91         "','" +
92         req.body.scseats +
93         "','" +
94         req.body.fcprice +
95         "','" +
96         req.body.scprice +
97         "')";
98       con.query(sql, function (err, result) { █
99         if (err) {
100           console.error('Error inserting!\n');
101           throw err;
102           res.end();
103         } else {
104           res.sendFile(path.join(__dirname, 'train_success.html'));
105         }
106       });
107     });
108   } else
109     res.redirect('/admin');
110 });

```

The procedure ADDTRAIN is given as:

```

SQL > proc_addtrain.sql
1  DELIMITER $$
2  CREATE PROCEDURE ADDTRAIN (IN TNO VARCHAR(15), IN ENO VARCHAR(20), IN FCS INT, IN SCS INT, IN FCP INT, IN SCP INT)
3  BEGIN
4    INSERT INTO TRAIN(TRAIN_NO, EXPRESS_NAME, FC_SEATS, SC_SEATS, FC_PRICE, SC_PRICE)
5    VALUES(TNO, ENO, FCS, SCS, FCP, SCP);
6  END $$
```

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
 1  INSERT INTO EXPRESS(TRAIN_NO,FROMM,TOO,STARTING_TIME,ARRIVAL_TIME,FC_SEATS_Remaining,SC_SEATS_Remaining)
 2  SELECT
 3  'E15698','Kolkata','Hyderabad','2021-06-13 08:30:00','2021-06-14 05:29:00',
 4  FC_SEATS,SC_SEATS FROM TRAIN WHERE TRAIN_NO = 'E15698';
 5
 6  INSERT INTO EXPRESS(TRAIN_NO,FROMM,TOO,STARTING_TIME,ARRIVAL_TIME,FC_SEATS_Remaining,SC_SEATS_Remaining)
 7  SELECT
 8  'E15698','Kolkata','Hyderabad','2021-06-15 10:30:00','2021-06-16 07:29:00',
 9  FC_SEATS,SC_SEATS FROM TRAIN WHERE TRAIN_NO = 'E15698';
10
11 INSERT INTO EXPRESS(TRAIN_NO,FROMM,TOO,STARTING_TIME,ARRIVAL_TIME,FC_SEATS_Remaining,SC_SEATS_Remaining)
12 SELECT
13 'E15698','Kolkata','Hyderabad','2021-05-29 10:30:00','2021-05-29 07:29:00',
14 FC_SEATS,SC_SEATS FROM TRAIN WHERE TRAIN_NO = 'E15698';
15
16 INSERT INTO EXPRESS(TRAIN_NO,FROMM,TOO,STARTING_TIME,ARRIVAL_TIME,FC_SEATS_Remaining,SC_SEATS_Remaining)
17 SELECT
18 'E16742','Kolkata','Hyderabad','2021-06-13 20:30:00','2021-06-14 12:29:00',
19 FC_SEATS,SC_SEATS FROM TRAIN WHERE TRAIN_NO = 'E16742';
20
21 INSERT INTO EXPRESS(TRAIN_NO,FROMM,TOO,STARTING_TIME,ARRIVAL_TIME,FC_SEATS_Remaining,SC_SEATS_Remaining)
22 SELECT
23 'E16742','Kolkata','Hyderabad','2021-05-29 20:30:00','2021-05-30 12:29:00',
24 FC_SEATS,SC_SEATS FROM TRAIN WHERE TRAIN_NO = 'E16742';
25
26 INSERT INTO EXPRESS(TRAIN_NO,FROMM,TOO,STARTING_TIME,ARRIVAL_TIME,FC_SEATS_Remaining,SC_SEATS_Remaining)
27 SELECT
28 'E17431','Kolkata','Hyderabad','2021-06-13 10:50:00','2021-06-14 08:45:00',
29 FC_SEATS,SC_SEATS FROM TRAIN WHERE TRAIN_NO = 'E17431';

31 INSERT INTO EXPRESS(TRAIN_NO,FROMM,TOO,STARTING_TIME,ARRIVAL_TIME,FC_SEATS_Remaining,SC_SEATS_Remaining)
32 SELECT
33 'E17431','Kolkata','Hyderabad','2021-06-15 12:50:00','2021-06-16 10:45:00',
34 FC_SEATS,SC_SEATS FROM TRAIN WHERE TRAIN_NO = 'E17431';
35
36 INSERT INTO EXPRESS(TRAIN_NO,FROMM,TOO,STARTING_TIME,ARRIVAL_TIME,FC_SEATS_Remaining,SC_SEATS_Remaining)
37 SELECT
38 'E18156','Kolkata','Hyderabad','2021-06-13 02:30:00','2021-06-14 00:29:00',
39 FC_SEATS,SC_SEATS FROM TRAIN WHERE TRAIN_NO = 'E18156';
40
41 INSERT INTO EXPRESS(TRAIN_NO,FROMM,TOO,STARTING_TIME,ARRIVAL_TIME,FC_SEATS_Remaining,SC_SEATS_Remaining)
42 SELECT
43 'E18156','Kolkata','Hyderabad','2021-05-29 02:30:00','2021-05-30 00:29:00',
44 FC_SEATS,SC_SEATS FROM TRAIN WHERE TRAIN_NO = 'E18156';
45
46 INSERT INTO EXPRESS(TRAIN_NO,FROMM,TOO,STARTING_TIME,ARRIVAL_TIME,FC_SEATS_Remaining,SC_SEATS_Remaining)
47 SELECT
48 'E19473','Kolkata','Hyderabad','2021-06-13 13:00:01','2021-06-14 14:45:01',
49 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:

```
120 app.post('/addexpress', function (req, res) { █
121   if (req.session.loggedin) {
122     con.connect(function (err) { █
123       var sql =
124         "insert into EXPRESS(TRAIN_NO,FROMM,TOO,STARTING_TIME,ARRIVAL_TIME,FC_SEATS_Remaining,SC_SEATS_Remaining) SELECT '' +
125         req.body.trainno +
126         '' , '' +
127         req.body.from +
128         '' , '' +
129         req.body.to +
130         '' , '' +
131         req.body.startingtime +
132         '' , '' +
133         req.body.arrivaltime +
134         '' , '' +'FC_SEATS,SC_SEATS FROM TRAIN WHERE TRAIN_NO=' + "" +
135         req.body.trainno +
136         ''' ;
137       con.query(sql, function (err, result) { █
138         if (err) [
139           console.error('Error inserting!\n');
140           throw err;
141         ] else {
142           res.sendFile(path.join(__dirname, 'express_success.html'));
143         }
144       });
145     });
146   } else
147     res.redirect('/admin');
148 });

});
```

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

```

164 app.post('/insert', function (req, res) {
165   bcrypt.hash(req.body.pwd, saltRounds,(err,hash)=>{
166     con.connect(function (err) {
167       var sql = "CALL ADDUSER('"+ req.body.uid + "','" + req.body.uname +
168           "','" + req.body.emailid + "','" + hash + "')";
169       con.query(sql, function (err, result) {
170         if (err) {
171           if (err(errno == 1062) {
172             res.sendFile(path.join(__dirname, 'errorins.html'));
173           } else {
174             console.error('Error inserting!\n');
175             throw err;
176             res.end();
177           }
178         } else {
179           res.sendFile(path.join(__dirname, 'regsucces.html'));
180         }
181       });
182     });
183   });
184 });

```

```

1 DELIMITER $$
2 CREATE PROCEDURE ADDUSER (IN UID VARCHAR(15), IN UNAME VARCHAR(20), IN EID VARCHAR(25), IN PWD VARCHAR(200))
3 BEGIN
4   INSERT INTO USER_DETAILS(USER_ID, USERNAME, EMAIL_ID, PASSWORD) VALUES(UID, UNAME, EID, PWD);
5 END $$
6 DELIMITER;
7

```

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.

```

186 app.post('/auth', function (request, response) {
187   var username = request.body.uid;
188   var password = request.body.pwd;
189   con.query('SELECT * FROM USER_DETAILS WHERE USER_ID = ? ', [username], function (error, results, fields) {
190     if (results.length !== 0) {
191       bcrypt.compare(password, results[0]['PASSWORD'], (error, result1) =>{
192         if(result1 === true){
193           request.session.loggedin = true;
194           request.session.username = username;
195           response.redirect('/home');
196         }
197         else {
198           response.sendFile(path.join(__dirname, 'Login_error.html'));
199         }
200       });
201     }
202   }
203   else response.sendFile(path.join(__dirname, 'Login_error.html'));
204 });
205 });
206 });

```

3. Home Page:

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

```

27 <ul class="noBullet">
28   <br><br>
29   <li id="center-btn">
30     <input type="button" id="join-btn" name="join" alt="booking" onclick="location.href='/booking'" value="BOOK NEW TICKETS">
31   </li><br>
32   <li id="center-btn">
33     <input type="button" id="join-btn" name="join" alt="Login" onclick="location.href='/existing'" value="VIEW EXISTING TICKETS">
34   </li><br>
35   <li id="center-btn">
36     <input type="button" id="join-btn" name="join" alt="booking" onclick="location.href='/prev'" value="VIEW PAST TRANSACTIONS">
37   </li><br>
38   <li id="center-btn">
39     <input type="button" id="join-btn" name="join" alt="booking" onclick="location.href='/logout'" value="LOGOUT">
40   </li>
41 </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.

```

261 app.post('/showtable', function (request, response) {
262   if (request.session.loggedin) {
263     data = [request.body.from, request.body.to, request.body.date, request.body.fc, request.body.sc];
264     var sqlquery =
265       'select EXPRESSVIEW.EXPRESS_NO,EXPRESSVIEW.TRAIN_NO,TRAIN.EXPRESS_NAME,EXPRESSVIEW.FROMM,' +
266       '"EXPRESSVIEW.TOO, DATE_FORMAT(EXPRESSVIEW.DEPARTURE_DATE, "%d-%b-%Y") as DEPARTURE_DATE,"+' +
267       '| DEPARTURE_TIME from EXPRESSVIEW,TRAIN where "' +
268       'EXPRESSVIEW.TRAIN_NO=TRAIN.TRAIN_NO and FROMM=? and TOO=? ' +
269       'and DATE(?)=DATE(DEPARTURE_DATE)' + ' and current_date() <= DATE(DEPARTURE_DATE)' +
270       'and FC_SEATS > ?' +
271       'and SC_SEATS> ?';
272     con.query(sqlquery, data, function (error, results, fields) {
273       if (error) throw error;
274       if (results.length <= 0) response.sendFile(path.join(__dirname, 'showtrains_error.html'));
275       else if (data[3] == 0 && data[4] == 0) response.sendFile(path.join(__dirname, 'invalid_error.html'));
276       else response.render('showtrain.ejs', { result: results });
277     });
278   } else {
279     response.redirect('/login');
280   }
281 });

```

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.

```

6      <script type="text/javascript">
7      function select(x){
8        document.getElementById("selection").value = x ;
9      }
10     </script>

```

```

60      <div align="center">
61          <form class="bookingform" action="/confirm" method="post">
62              <input type="text" style="text-align: center;" name="selection" id="selection" value="" placeholder="Express_no"
63              required onkeydown="return false;">
64              <input type="submit" class="smallbut" name="" value="Book tickets">
65          </form>
66      </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

391 app.post('/topay', function (request, response) {
392     if (request.session.loggedin) {
393         var password = request.body.pwd;
394         con.query('SELECT * FROM USER_DETAILS WHERE USER_ID = ? ', [request.session.username], function (error, results, fields) {
395             if (results.length !== 0) {
396                 bcrypt.compare(password, results[0]['PASSWORD'], (error, result1) => {
397                     if (result1 === true) {
398                         var sql ='insert into BOOKING(EXPRESS_NO,USER_ID,TRAIN_NO,TOTAL_FC_SEATS,TOTAL_SC_SEATS,TOTAL_COST) values(' +
399                             "'" + final[0] + "','" + "'" + request.session.username + "','" + ',' +
400                             "'" + final[1] + "','" + final[4] + ',' + final[5] + ',' + final[6] + ')';
401                         con.query(sql, function (err, result) {
402                             if (err) {
403                                 console.error('Error inserting!\n');
404                                 throw err;
405                             }
406                             else {
407                                 var updateview = 'CREATE OR REPLACE VIEW EXPRESSVIEW AS SELECT EXPRESS_NO, TRAIN_NO, FROMM,' +
408                                     ' TOO, DATE(STARTING_TIME) AS DEPARTURE_DATE, TIME(STARTING_TIME) AS DEPARTURE_TIME FROM EXPRESS';
409                                 con.query(updateview);
410                                 response.sendFile(path.join(__dirname, 'paysuccess.html'));
411                             }
412                         });
413                     }
414                     else {
415                         response.sendFile(path.join(__dirname, 'payerror.html'));
416                     }
417                 });
418             } else response.sendFile(path.join(__dirname, 'payerror.html'));
419         });
420     } else {
421         response.redirect('/login');
422     }
423 });
424

```

```

1 DELIMITER $$ 
2 CREATE TRIGGER bookingtrigger
3 AFTER INSERT ON BOOKING FOR EACH ROW
4 BEGIN
5     UPDATE EXPRESS SET FC_SEATS_REMAINING = FC_SEATS_REMAINING - NEW.TOTAL_FC_SEATS WHERE EXPRESS_NO = NEW.EXPRESS_NO;
6     UPDATE EXPRESS SET SC_SEATS_REMAINING = SC_SEATS_REMAINING - NEW.TOTAL_SC_SEATS WHERE EXPRESS_NO = NEW.EXPRESS_NO;
7 END$$
8 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.

```

242 app.get('/existing', function (request, response) {
243   if (request.session.loggedin) {
244     var quer =
245       'select PRESENTVIEW.BOOKING_ID,PRESENTVIEW.EXPRESS_NO,PRESENTVIEW.TRAIN_NO,' +
246       'DATE_FORMAT(PRESENTVIEW.DEPARTURE_DATE,"%d-%b-%Y") as DEPARTURE_DATE,PRESENTVIEW.DEPARTURE_TIME,' +
247       'PRESENTVIEW.TOTAL_FC_SEATS,PRESENTVIEW.TOTAL_SC_SEATS,PRESENTVIEW.TOTAL_COST ' +
248       'from EXPRESS,PRESENTVIEW where EXPRESS.EXPRESS_NO=PRESENTVIEW.EXPRESS_NO and ' +
249       'PRESENTVIEW.USER_ID = ' +
250       '""' +
251       request.session.username +
252       '""' +
253       ' and current_timestamp() < EXPRESS.STARTING_TIME';
254   con.query(quer, function (error, results, fields) {
255     if (error) throw error;
256     if (results.length <= 0) response.sendFile(path.join(__dirname, 'no_existing.html'));
257     else response.render('viewpresent.ejs', { result: results });
258   });
259 } else {
260   response.redirect('/login');
261 }
262 });

```

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.

```

6   <script type="text/javascript">
7     function select(x){
8       document.getElementById("selection").value = x ;
9     }
10    </script>

```

```

63      <div align="center">
64        <form class="bookingform" action="/cancel" method="post">
65          <input type="text" style="text-align: center;" name="selection" id="selection" value="" placeholder="Ticket no."
66          required onkeydown="return false;">
67          <input type="submit" class="smallbut" name="" value="Cancel ticket">
68        </form>
69      </div>
70      <div align="center">
71        <form class="bookingform" action="/home" method="get">
72          <input type="submit" class="smallbut" name="" value="Go back">
73        </form>
74      </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.

```

329 app.post('/tocancel', function (request, response) {
330   if (request.session.loggedin) {
331     var password = request.body.pwd;
332     con.query('SELECT * FROM USER_DETAILS WHERE USER_ID = ? ', [request.session.username], function (error, results, fields) {
333       if (results.length !== 0) {
334         bcrypt.compare(password, results[0]['PASSWORD'], (error, result1) => {
335           if (result1 === true) {
336             var sql = 'delete from BOOKING where BOOKING_ID = ' + "" + ticketno + "";
337             con.query(sql, function (err, result) {
338               if (err) {
339                 console.error('Error inserting!\n');
340                 throw err;
341               } else {
342                 response.sendFile(path.join(__dirname, 'cancelsuccess.html'));
343               }
344             });
345           } else {
346             response.sendFile(path.join(__dirname, 'cancelerror.html'));
347           }
348         });
349       } else response.sendFile(path.join(__dirname, 'cancelerror.html'));
350     });
351   } else {
352     response.redirect('/login');
353   }
354 });
355

```

```

10 DELIMITER $$ 
11 CREATE TRIGGER canceltrigger
12 AFTER DELETE ON BOOKING FOR EACH ROW
13 BEGIN
14   UPDATE EXPRESS SET FC_SEATS_REMAINING = FC_SEATS_REMAINING + OLD.TOTAL_FC_SEATS WHERE EXPRESS_NO = OLD.EXPRESS_NO;
15   UPDATE EXPRESS SET SC_SEATS_REMAINING = SC_SEATS_REMAINING + OLD.TOTAL_SC_SEATS WHERE EXPRESS_NO = OLD.EXPRESS_NO;
16 END$$
17 DELIMITER ;
18

```

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.

```

223 app.get('/prev', function (request, response) {
224   if (request.session.loggedin) {
225     var quer =
226       'select BOOKING.* from BOOKING,EXPRESS where BOOKING.EXPRESS_NO=EXPRESS.EXPRESS_NO and ' +
227       'USER_ID = ' +
228       '"" +
229       request.session.username +
230       '"" +
231       ' and current_timestamp() > EXPRESS.STARTING_TIME';
232     con.query(quer, function (error, results, fields) {
233       if (error) throw error;
234       if (results.length <= 0) response.sendfile(path.join(__dirname, 'no_previous.html'));
235       else response.render('viewpast.ejs', { result: results });
236     });
237   } else {
238     response.redirect('/login');
239   }
240 });

```

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.

```

441 app.get('/logout', function (request, response) {
442   request.session.loggedin = false;
443   request.session.username = ' ';
444   response.redirect('/');
445 });
446

```

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)

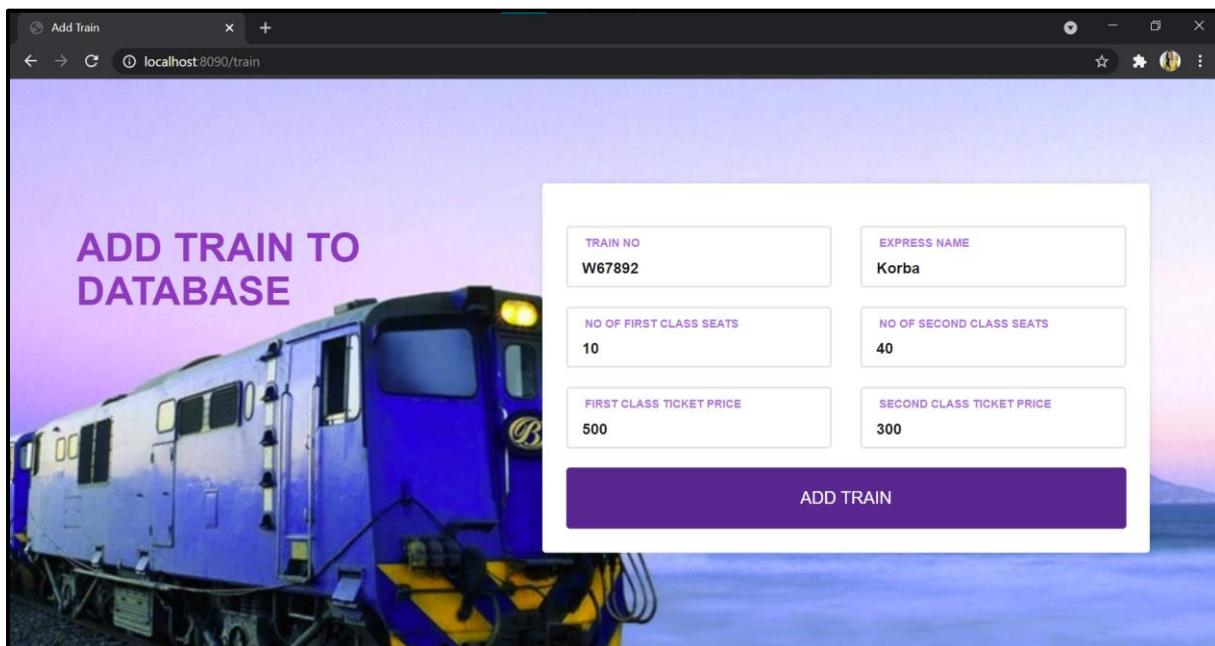
Query OK, 1 row affected (0.00 sec)
```

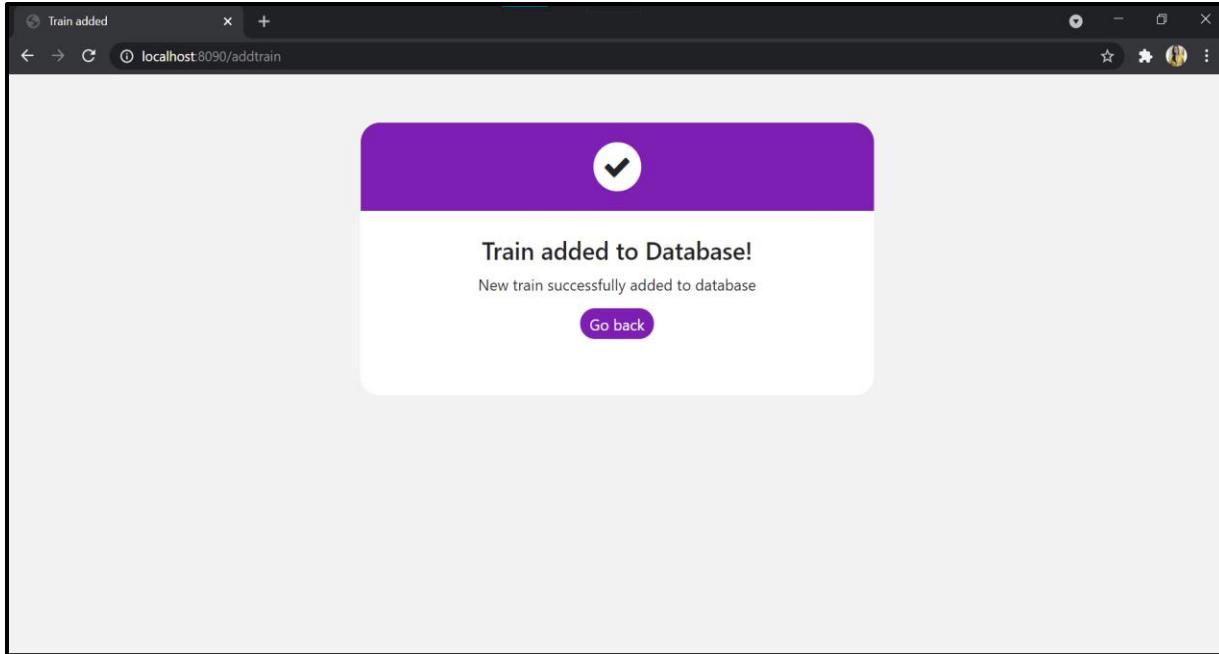
```
Query OK, 1 row affected (0.00 sec)
```

```
+-----+
| 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
```

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	N21823	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	S55813	Chennai	Mysore	2021-06-25 10:00:00	2021-06-25 18:10:00	20	35
10035	S55813	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.

ADD EXPRESS TO DATABASE

TRAIN NO
W65681

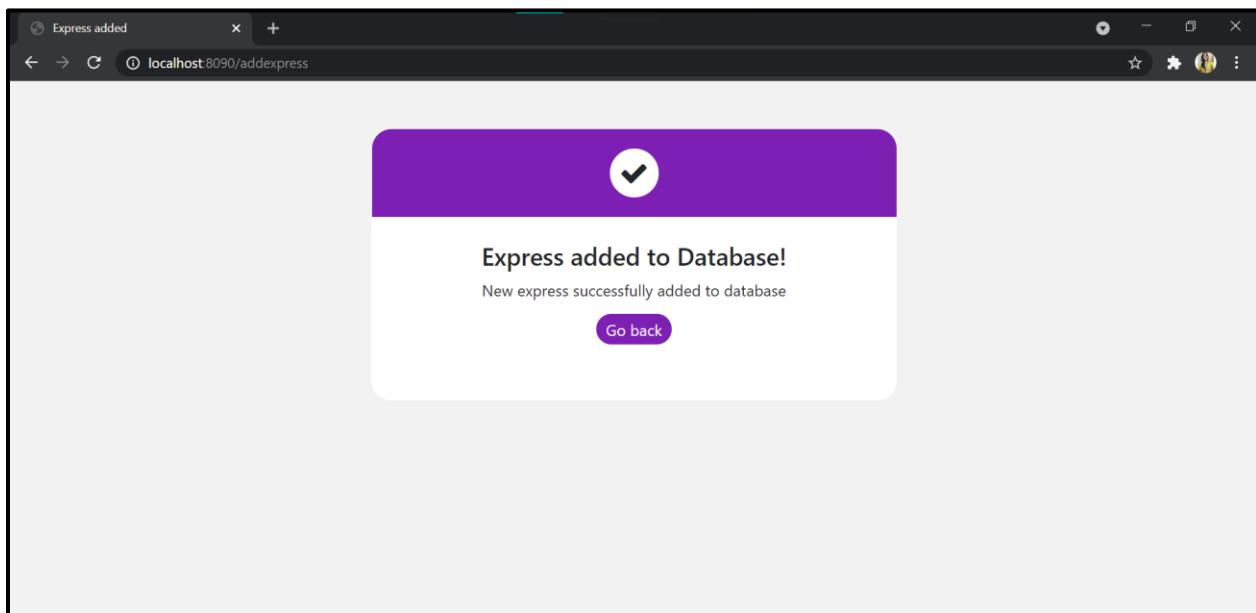
FROM
Nagpur

TO
Bhopal

STARTING TIME
2021-07-08 14:30:00

ARRIVAL TIME
2021-07-08 22:45:00

ADD EXPRESS



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)

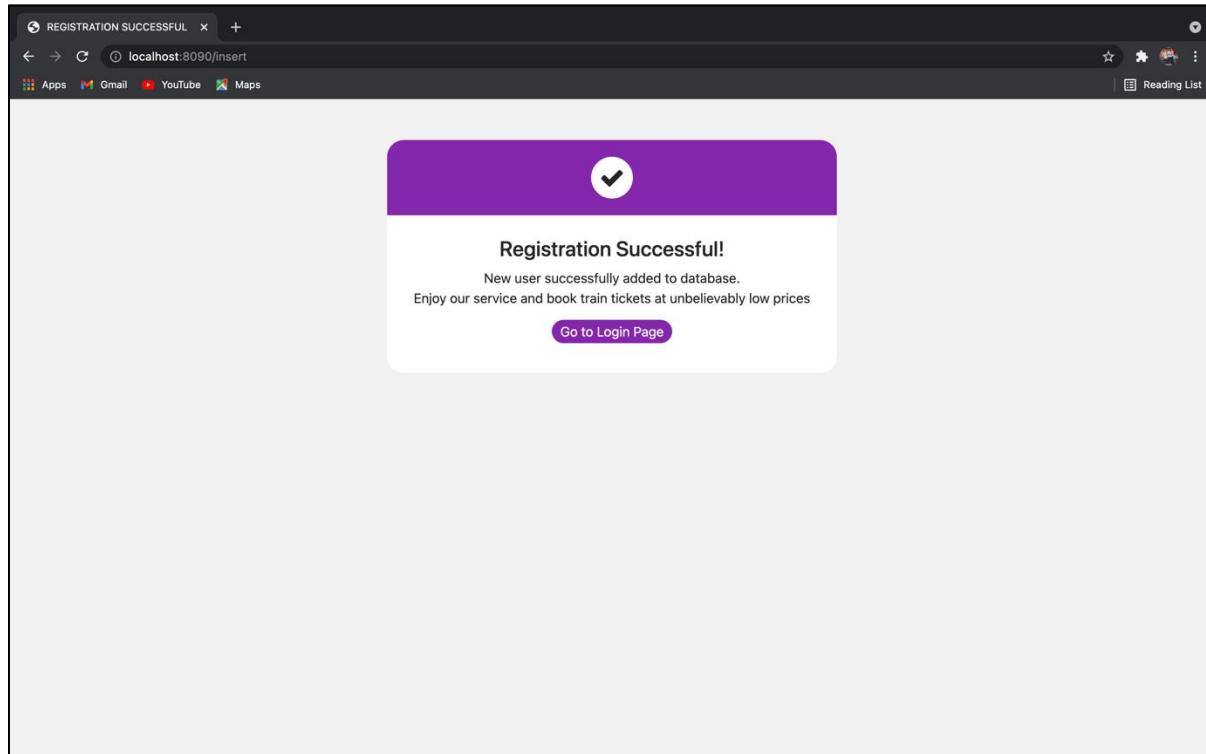
mysql> source C:\Users\DeLL\Desktop\DBMS PROJECT\SQL files\proc_adduser.sql
Query OK, 0 rows affected (0.01 sec)
```

USER SIDE:

1. Creating account (Sign-Up)

Successful Signup:

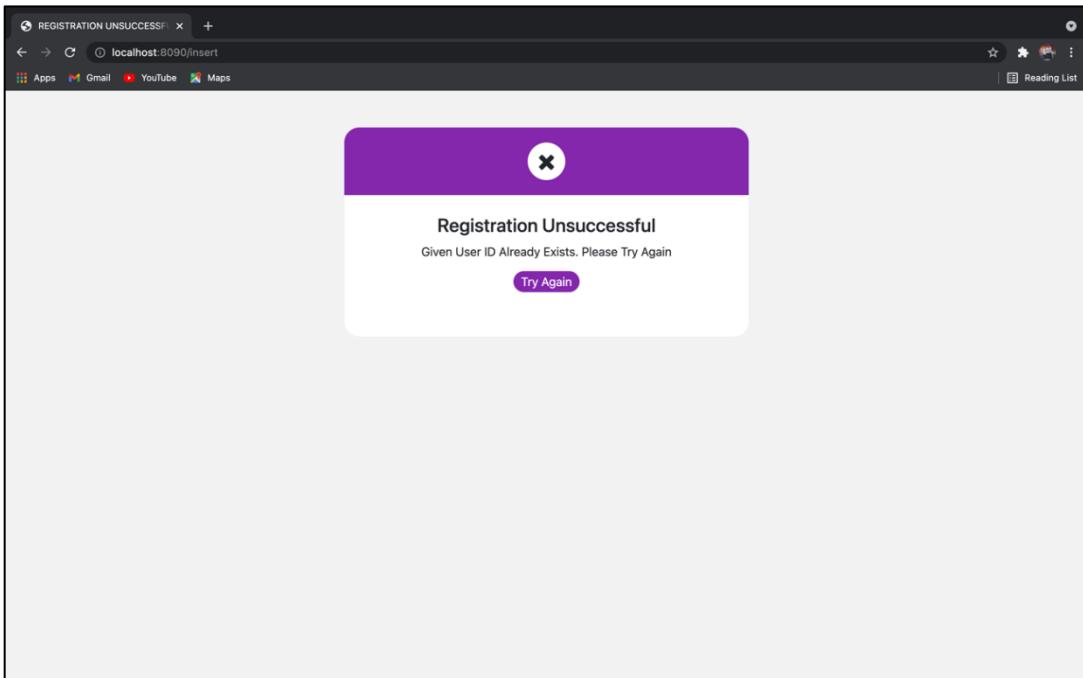
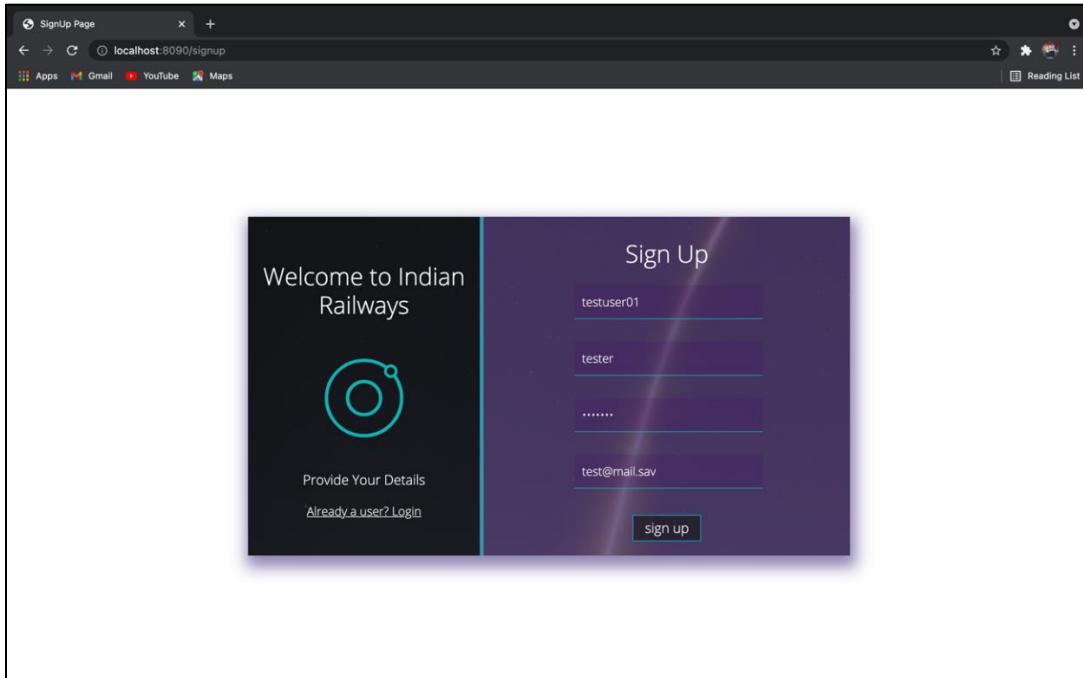
The screenshot shows a web browser window titled "SignUp Page" with the URL "localhost:8090/signup". The page is divided into two sections: a dark blue header and a white body. The header contains the text "Welcome to Indian Railways" and a teal circular logo. The body has a teal header "Sign Up" and four input fields: "testuser01" (username), "Test user" (first name), "....." (last name), and "test01@admin.sav" (email). A teal "sign up" button is at the bottom.



Account details added to database after encryption:

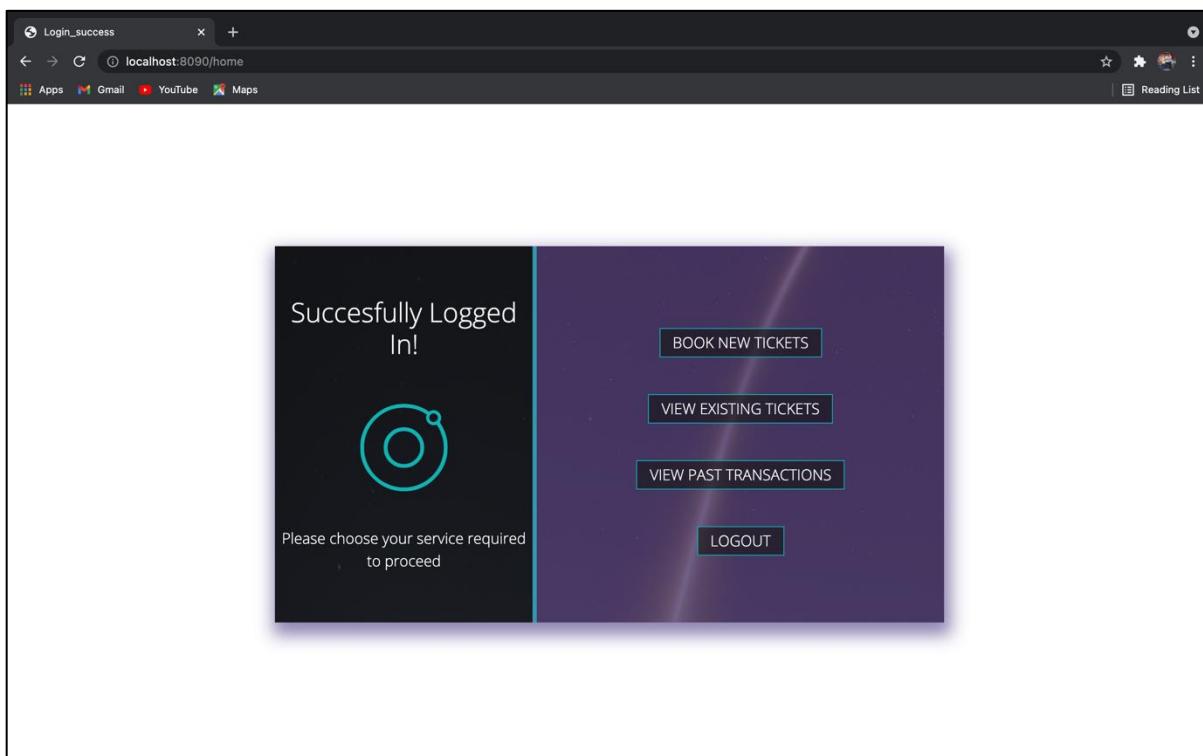
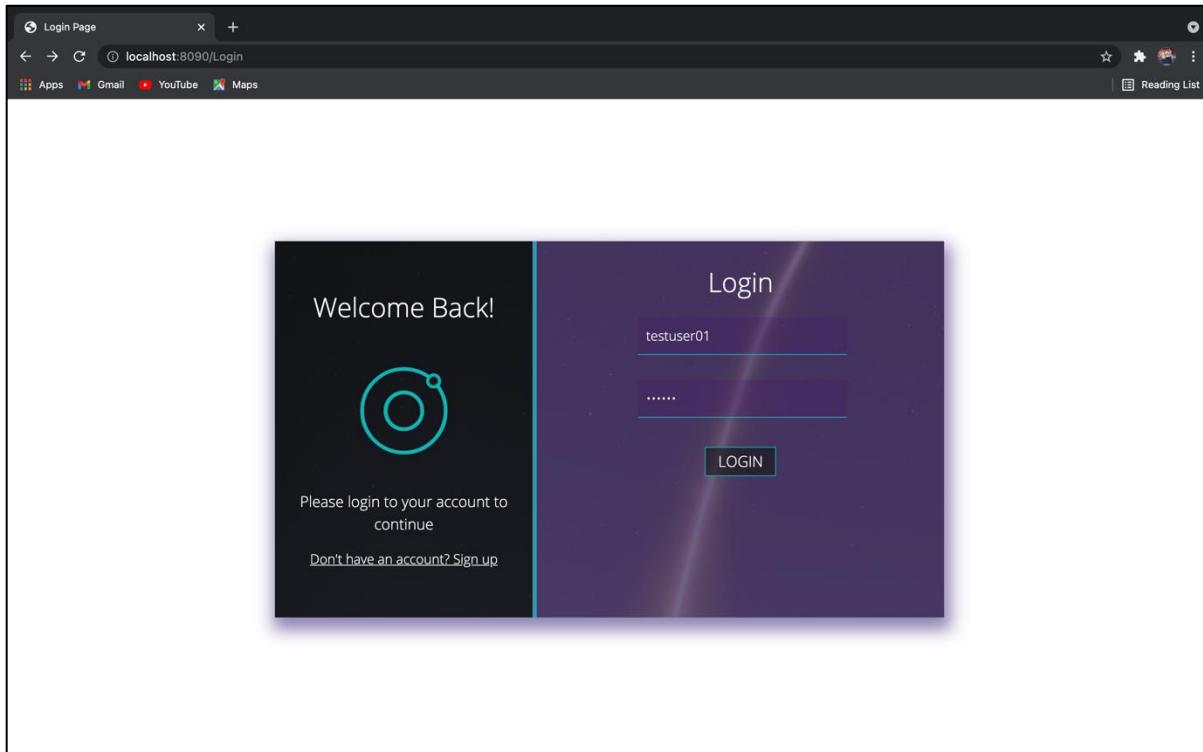
```
|mysql> select * from user_details where USER_ID = 'testuser01';
+-----+-----+-----+
| USER_ID | USERNAME | EMAIL_ID           | PASSWORD
+-----+-----+-----+
| testuser01 | Test user | test01@admin.sav | $2b$10$s6UIshlc6b1FCqDq2Ri1ROHG03KGUL5Ccnz7MAjh.5LHTBbLf.h3e |
+-----+-----+-----+
1 row in set (0.01 sec)
```

Unsuccessful Signup:

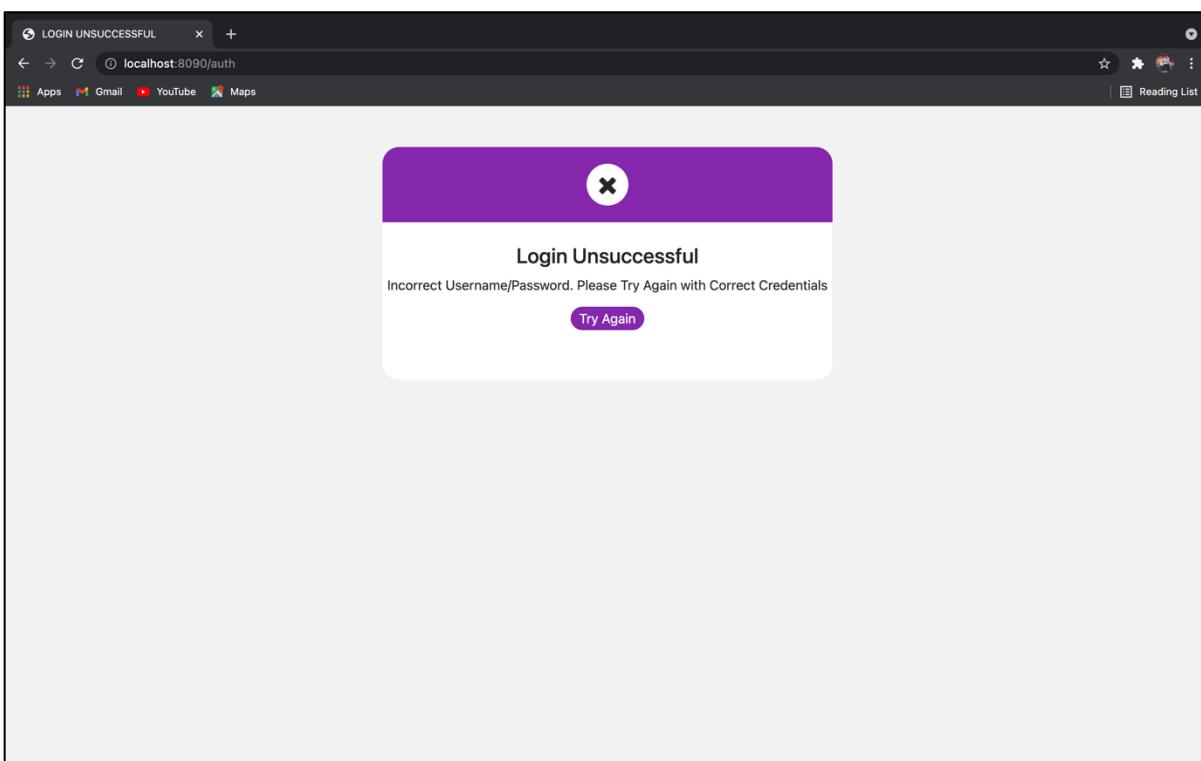
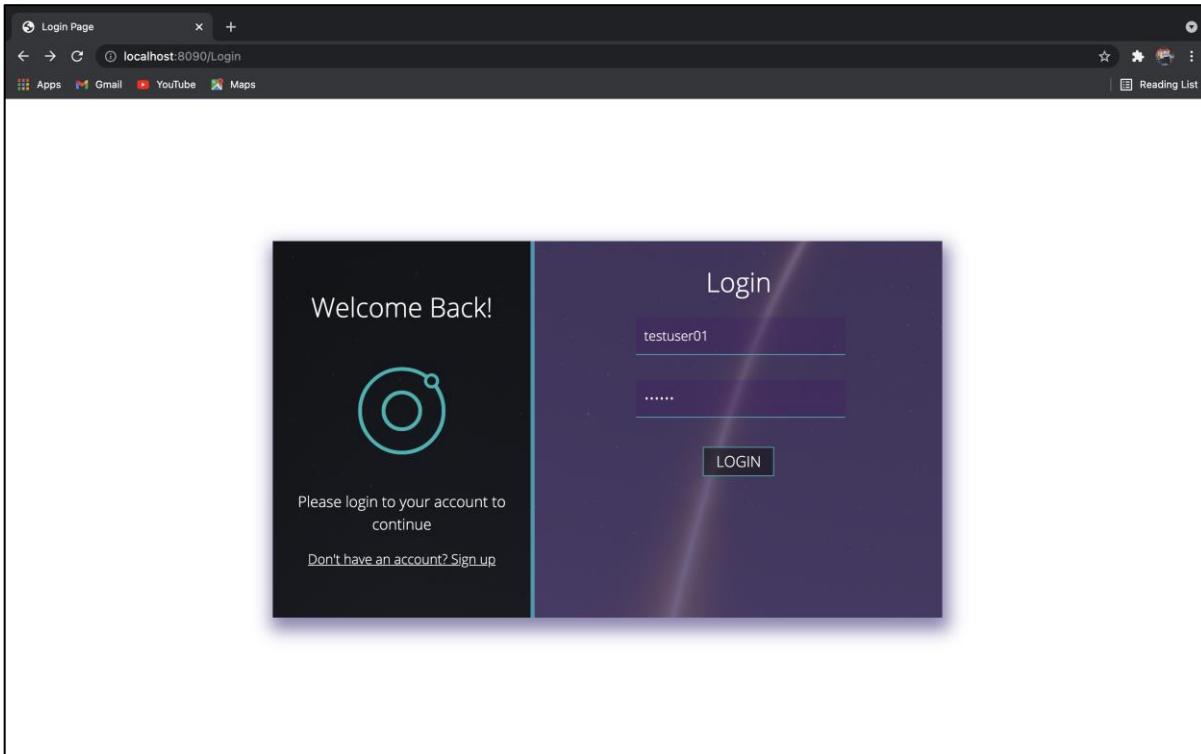


2. Login

Successful login:

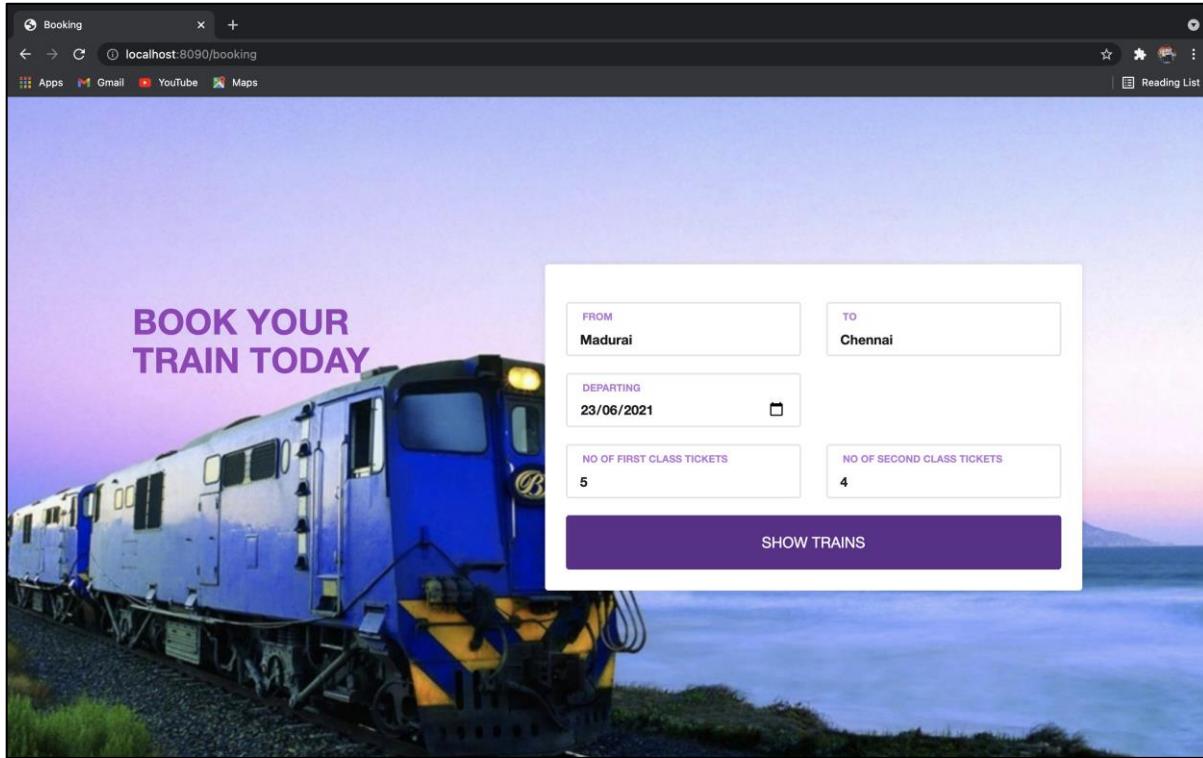


Unsuccessful login:



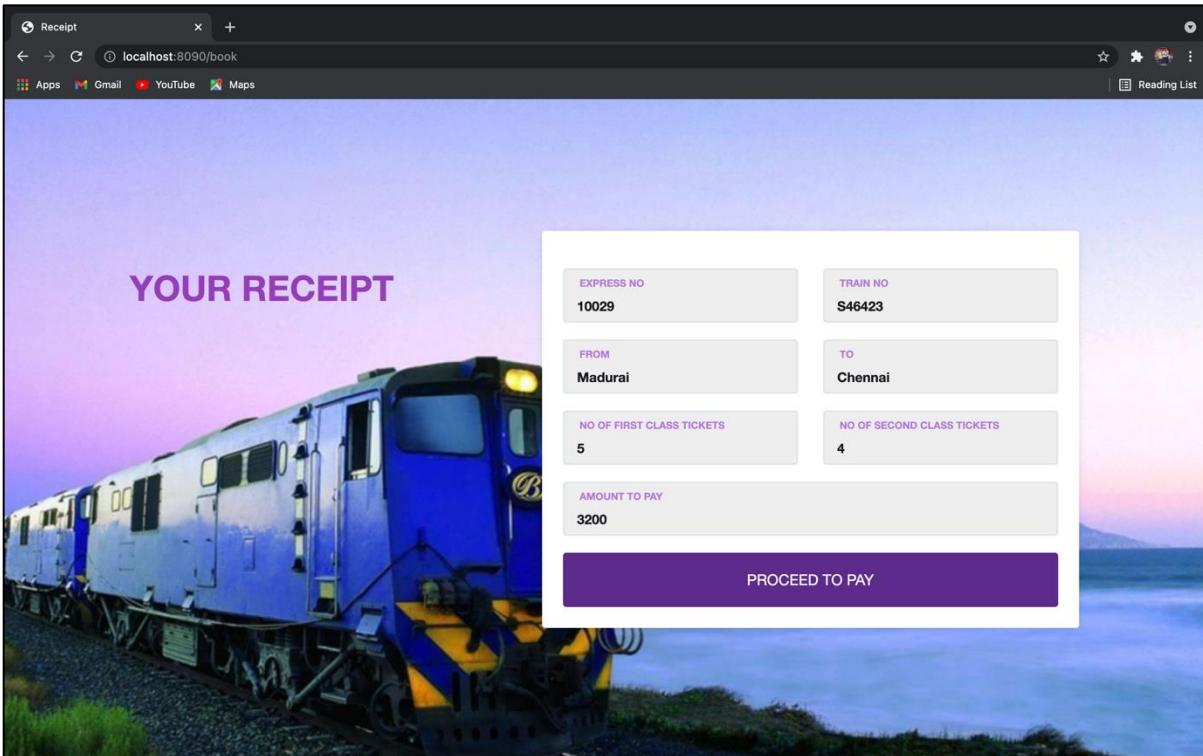
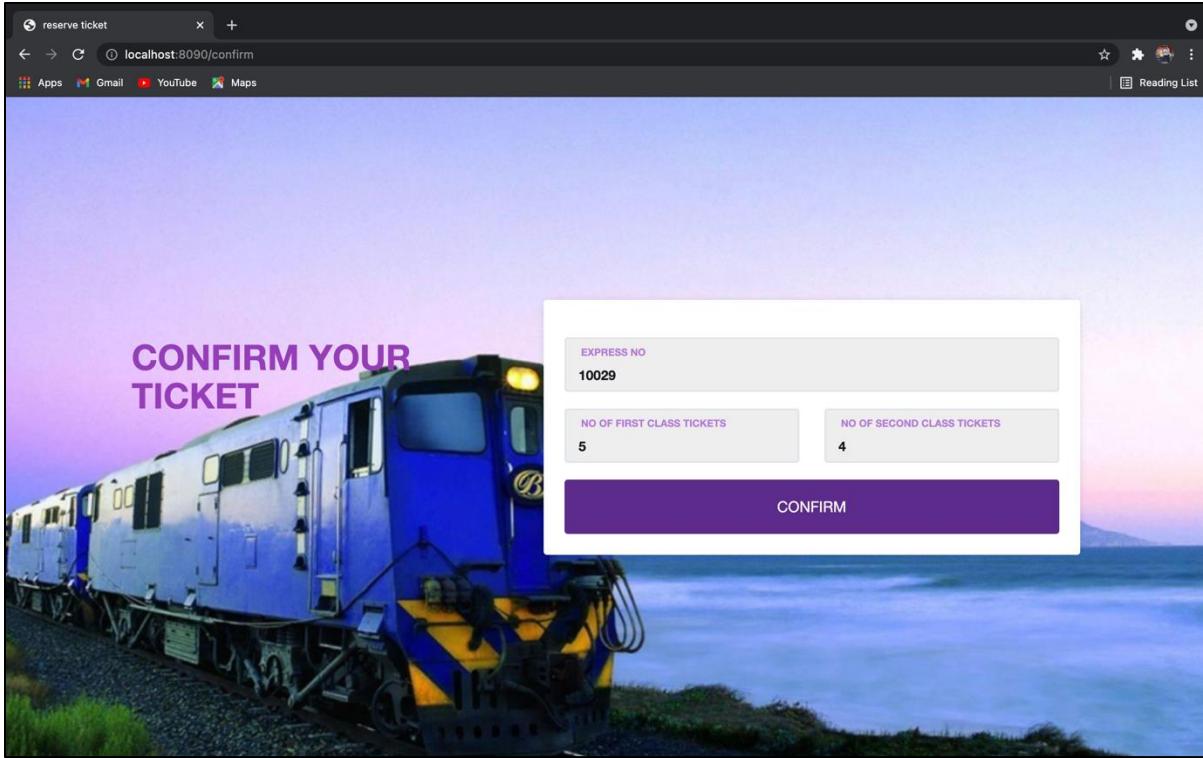
3. Booking

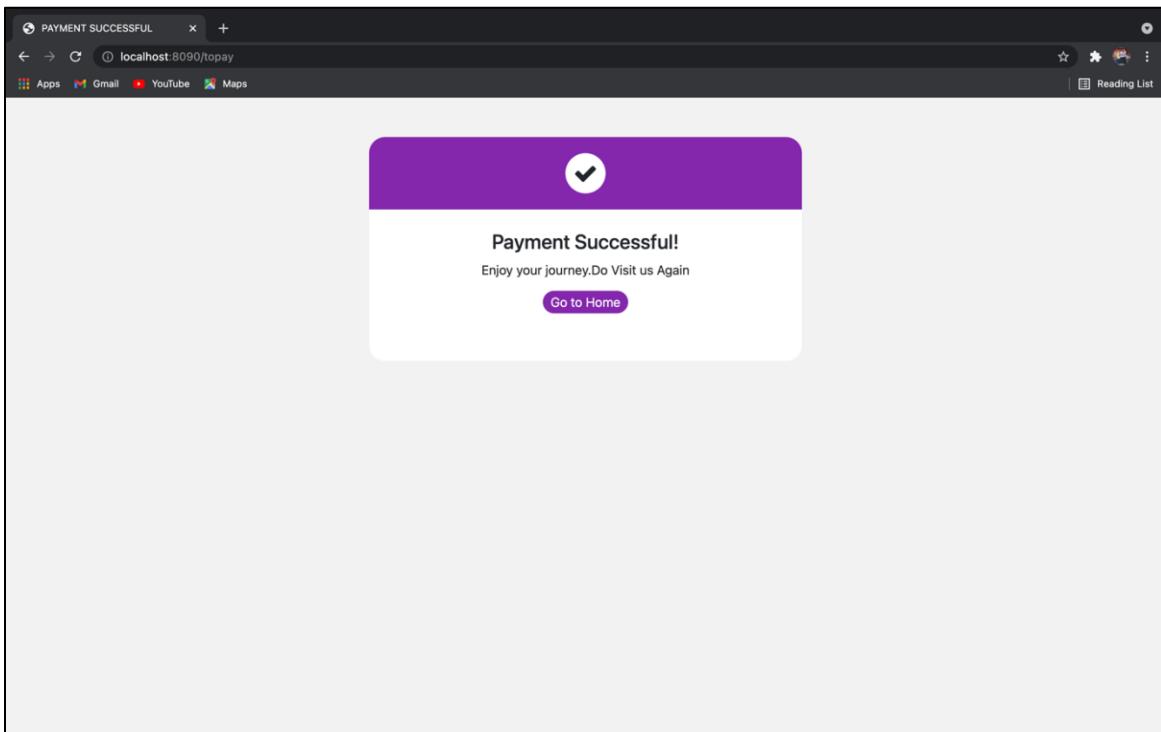
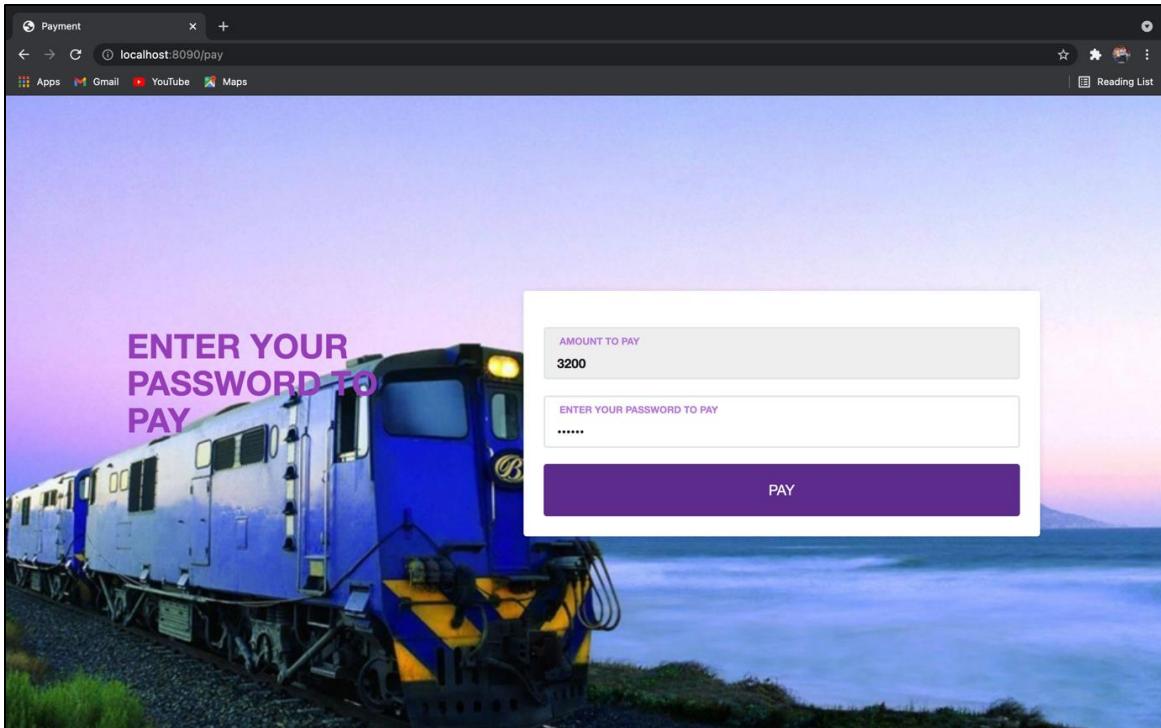
Successful Booking:



Express Number	Train Number	Express Name	From	Destination	Departure Date	Departure Time
10023	S44592	Pandian Express	Madurai	Chennai	23-Jun-2021	23:30:00
10025	S45678	Vaigai Express	Madurai	Chennai	23-Jun-2021	10:00:00
10029	S46423	Tejas Express	Madurai	Chennai	23-Jun-2021	13:30:00
10031	S47365	Guruvayur Express	Madurai	Chennai	23-Jun-2021	23:30:00

10029 Book tickets





```
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:

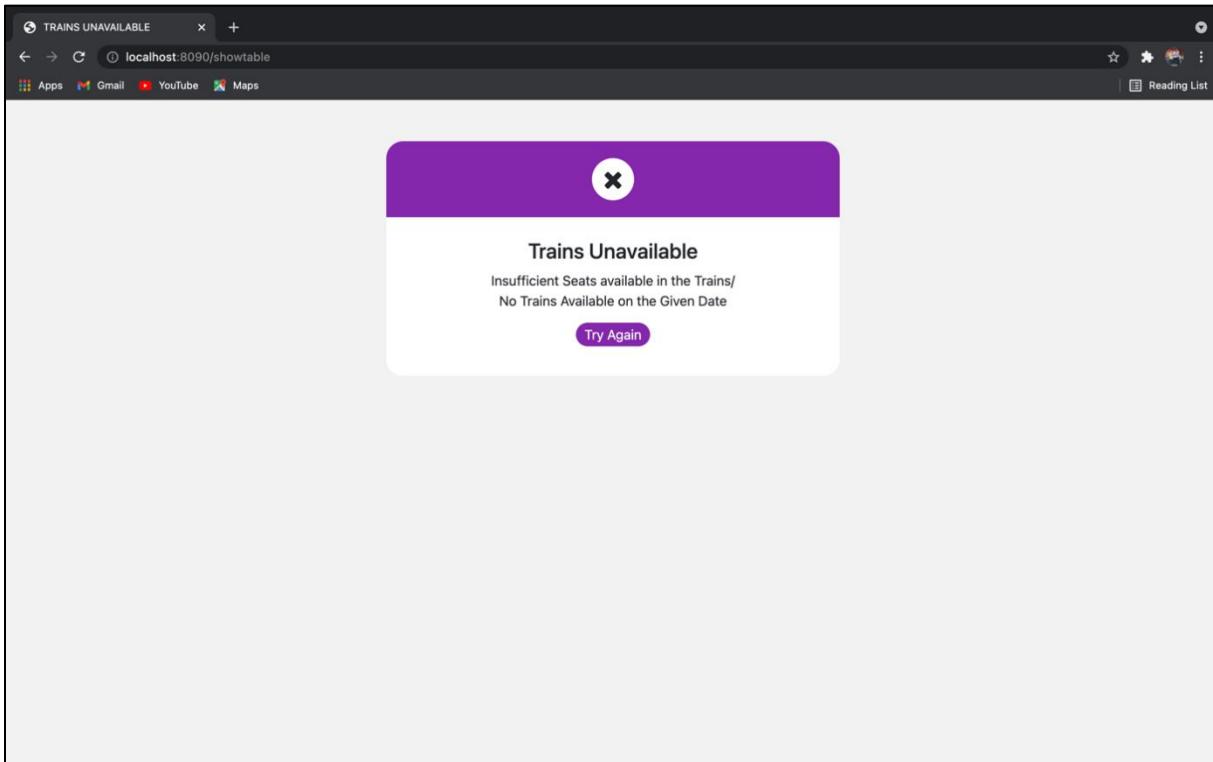
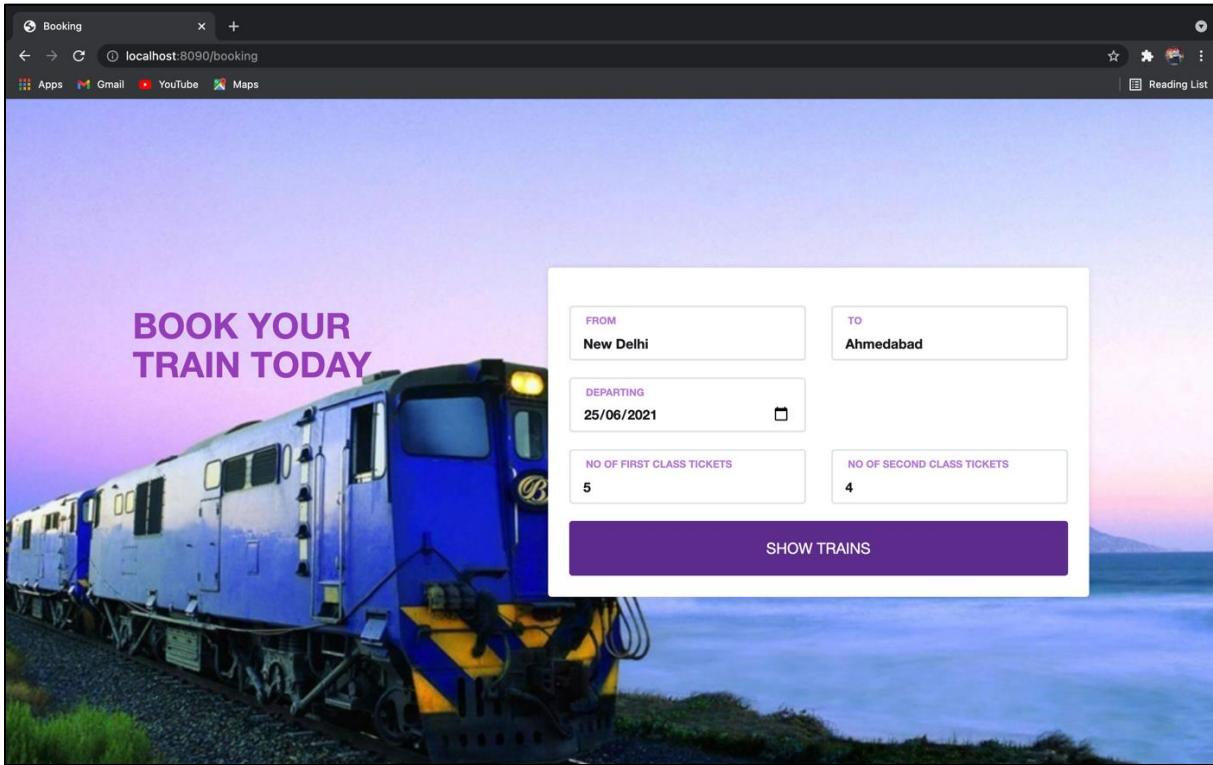
10027	S45678	Madurai	Chennai	2021-06-24 10:00:00	2021-06-24 17:10:00	10	25
10028	S45678	Madurai	Chennai	2021-05-30 10:00:00	2021-05-30 17:10:00	10	25
10029	S46423	Madurai	Chennai	2021-06-23 13:30:00	2021-06-23 20:05:00	10	30
10030	S46423	Madurai	Chennai	2021-05-30 13:30:00	2021-05-30 20:05:00	9	30

After booking the ticket:

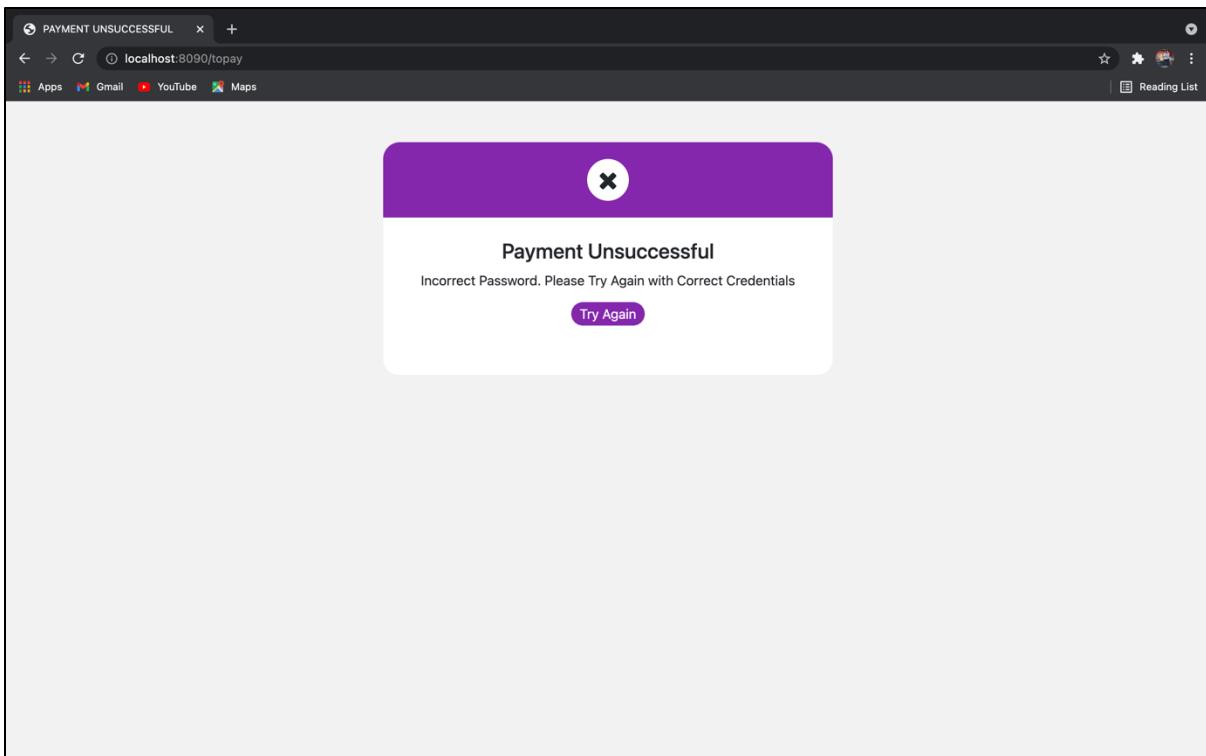
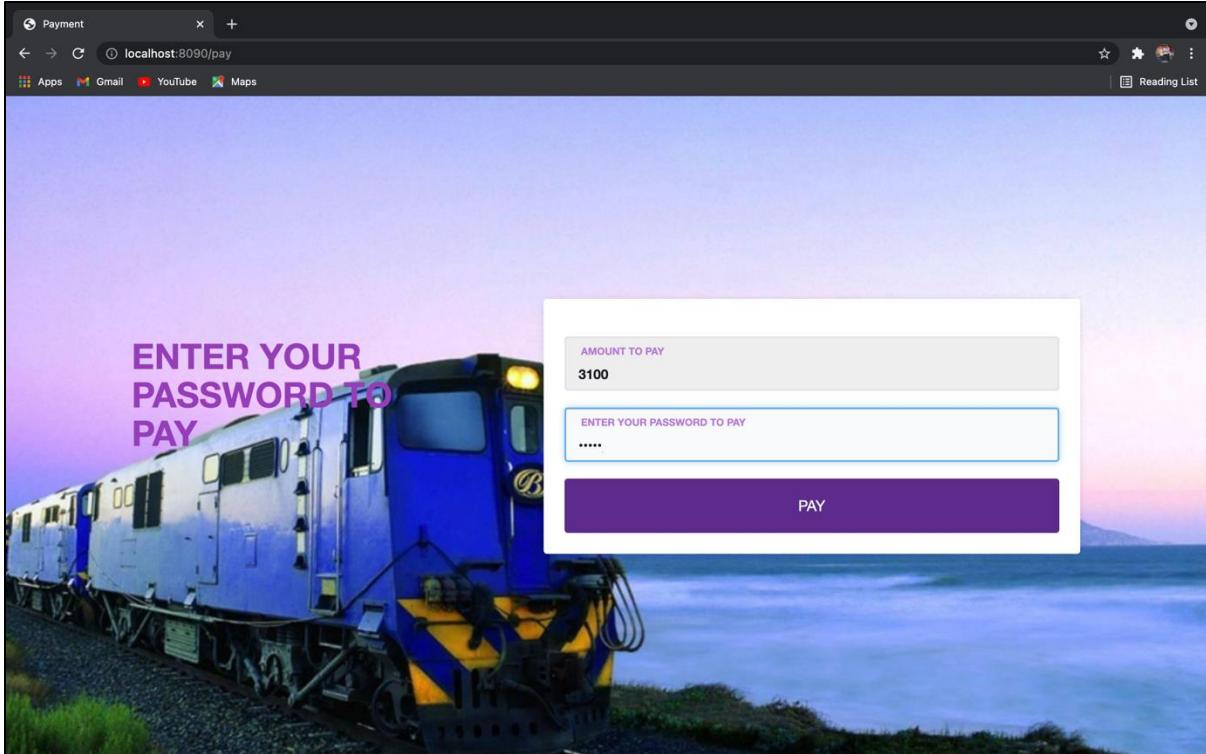
EXPRESS_NO	TRAIN_NO	FROMM	TOO	STARTING_TIME	ARRIVAL_TIME	FC_SEATS_Remaining	SC_SEATS_Remaining
10001	E15698	Kolkata	Hyderabad	2021-06-13 08:30:00	2021-06-14 05:29:00	10	20
10002	E15698	Kolkata	Hyderabad	2021-06-15 10:30:00	2021-06-16 07:29:00	10	20
10003	E15698	Kolkata	Hyderabad	2021-05-29 10:30:00	2021-05-29 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:



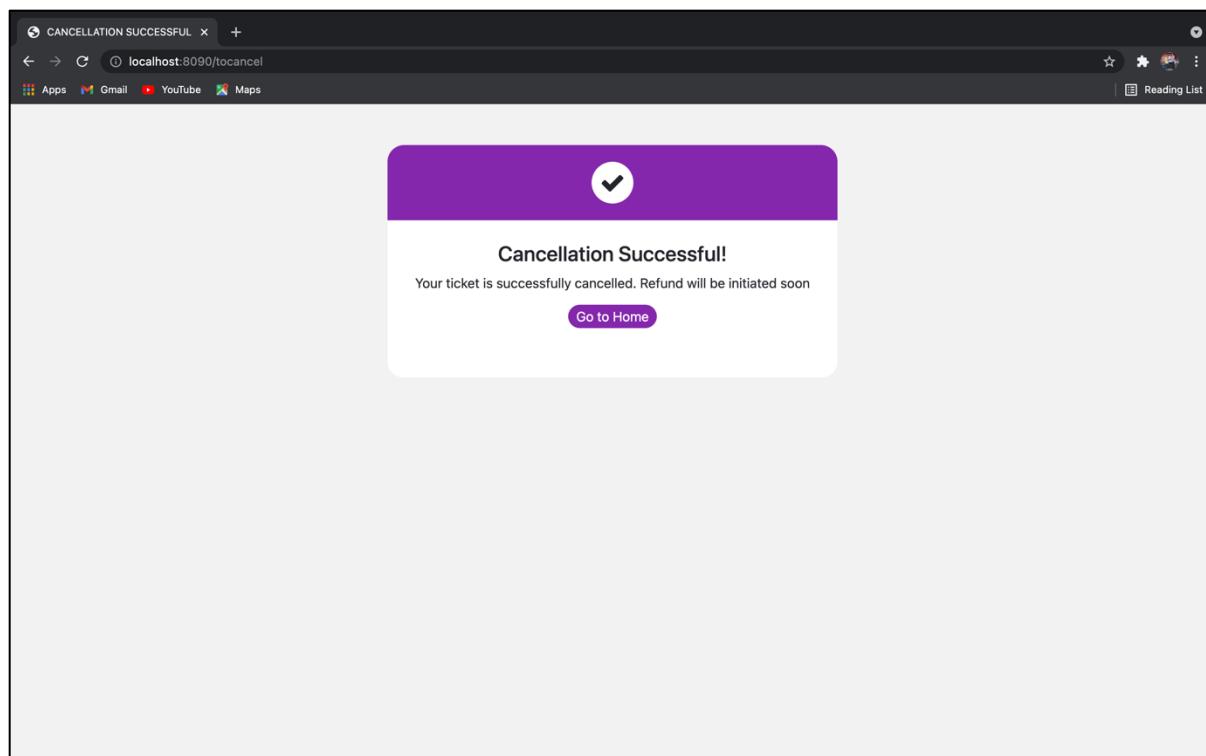
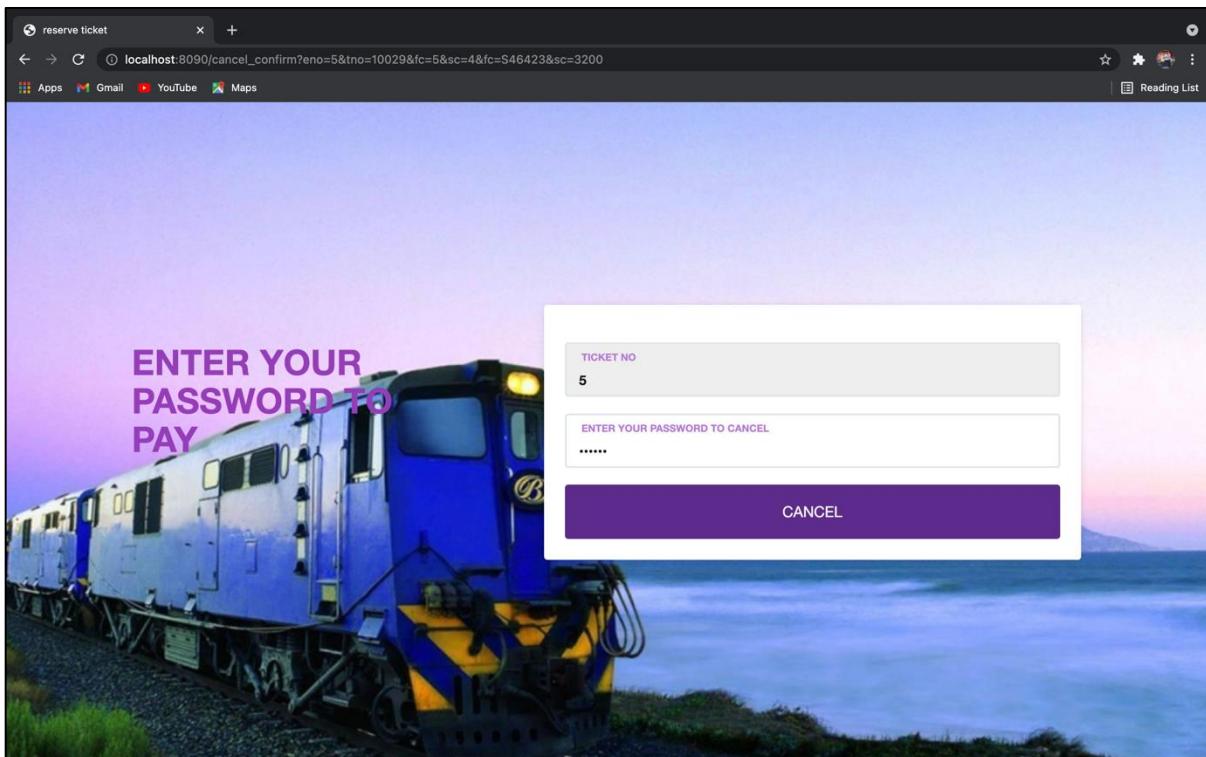
4. View existing tickets and cancellation

Screenshot of a web browser showing the "EXISTING TICKETS" page. The URL is `localhost:8090/existing`. The page displays a table of existing tickets with columns: Ticket number, Express no, Train no, Departure date, Departure time, No. of FC seats, No. of SC seats, and Total Cost. One row is selected, highlighted with a blue border.

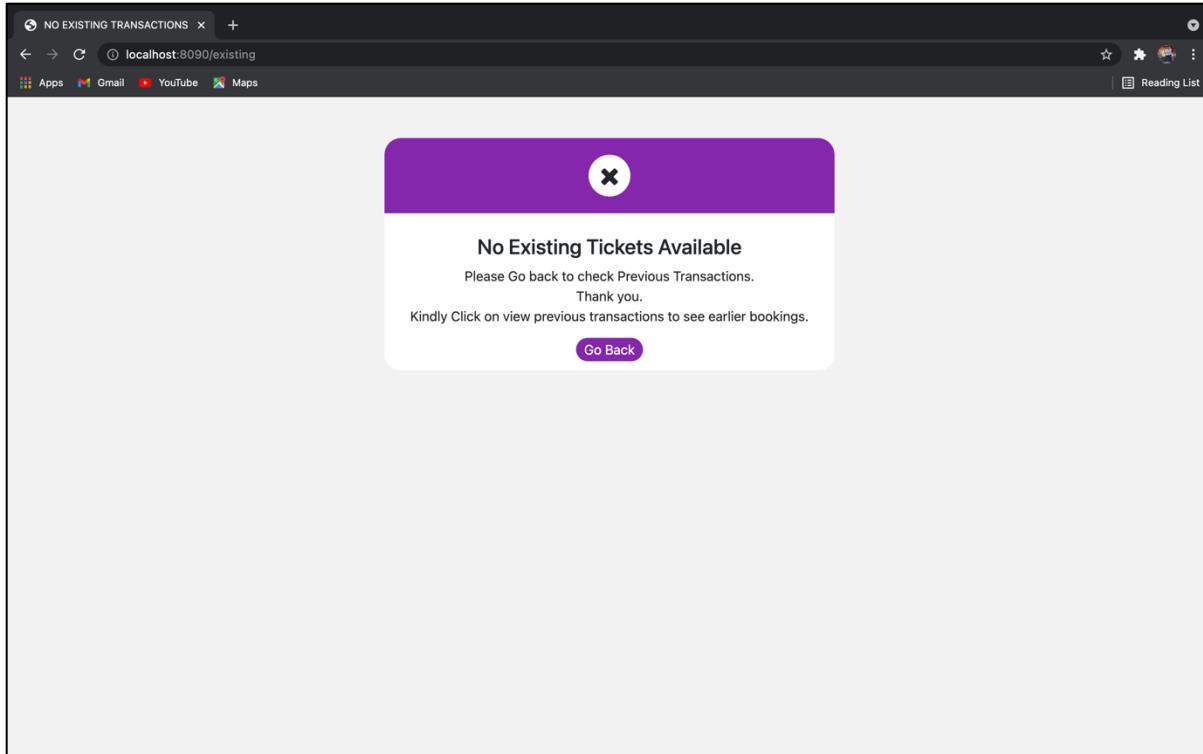
Ticket number	Express no	Train no	Departure date	Departure time	No. of FC seats	No. of SC seats	Total Cost
5	10029	S46423	23-Jun-2021	13:30:00	5	4	3200

Buttons at the bottom include "5" (highlighted), "Cancel ticket" (in a teal button), and "Go back" (in a blue button).

Screenshot of a web browser showing the "YOUR RECEIPT" page. The URL is `localhost:8090/cancel`. The page features a background image of a blue and yellow train. A modal dialog box displays the ticket details: Ticket No (5), Express No (10029), No of First Class Tickets (5), No of Second Class Tickets (4), Train No (S46423), and Total Cost (3200). A large purple "PROCEED TO CANCEL" button is at the bottom of the dialog.



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

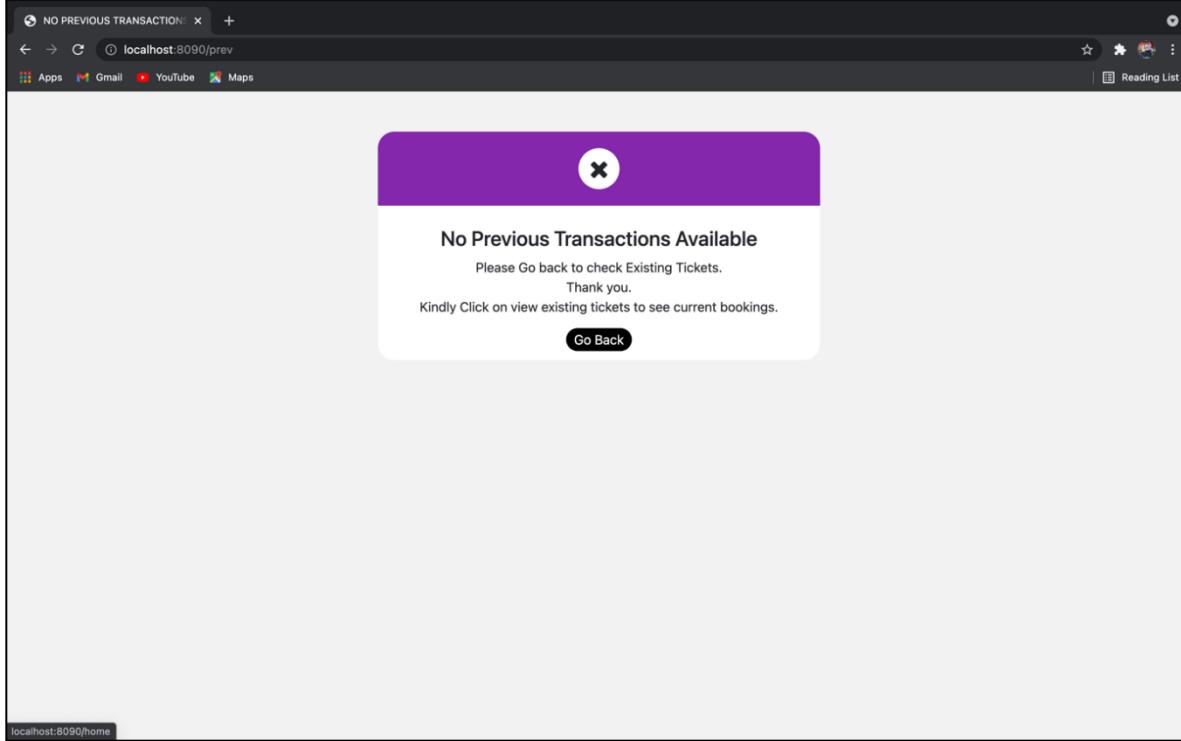


The trigger acts on the EXPRESS relation upon ticket cancellation as show below:

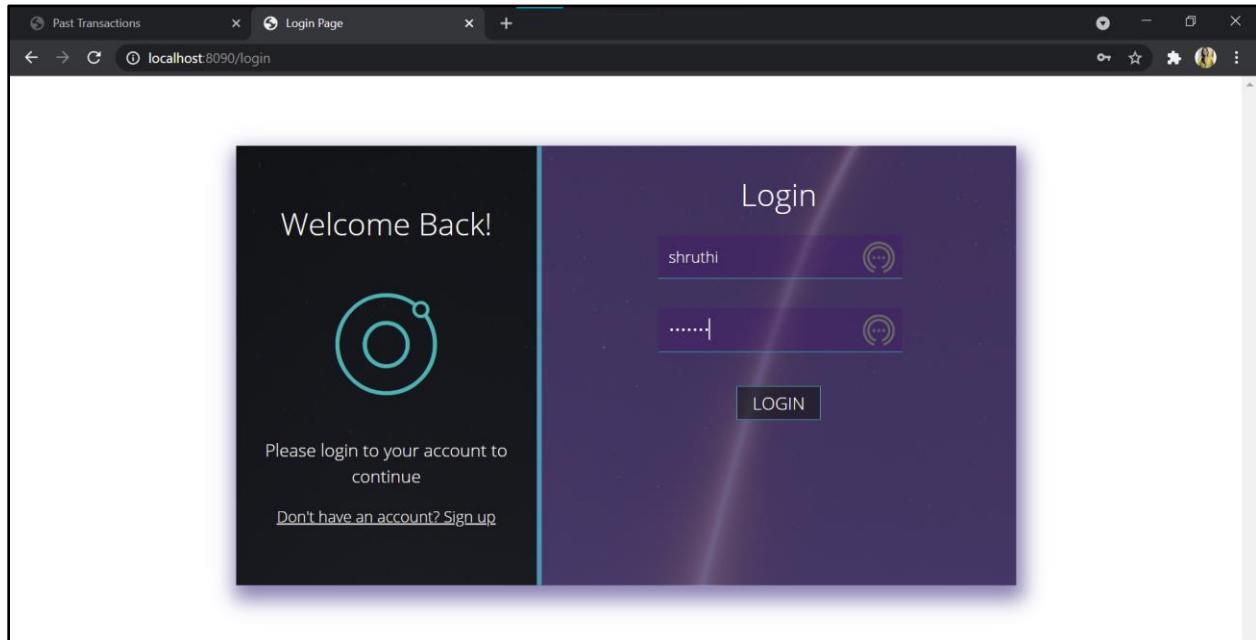
mysql> SELECT * FROM EXPRESS;								
EXPRESS_NO	TRAIN_NO	FROM	TO	STARTING_TIME	ARRIVAL_TIME	FC_SEATS_REMAINING	SC_SEATS_REMAINING	AC_SEATS_REMAINING
10001	E15698	Kolkata	Hyderabad	2021-06-13 08:30:00	2021-06-14 05:29:00	10	20	
10002	E15698	Kolkata	Hyderabad	2021-06-15 10:30:00	2021-06-16 07:29:00	10	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)



Past Transactions

Ticket number	Express no	Train no	Departure date	Departure time	No. of FC seats	No. of SC seats	Total Cost
2	10005	E16742	29-May-2021	20:30:00	2	4	2100
3	10030	S46423	30-May-2021	13:30:00	1	0	400
4	10021	N36913	29-May-2021	12:30:00	2	0	800

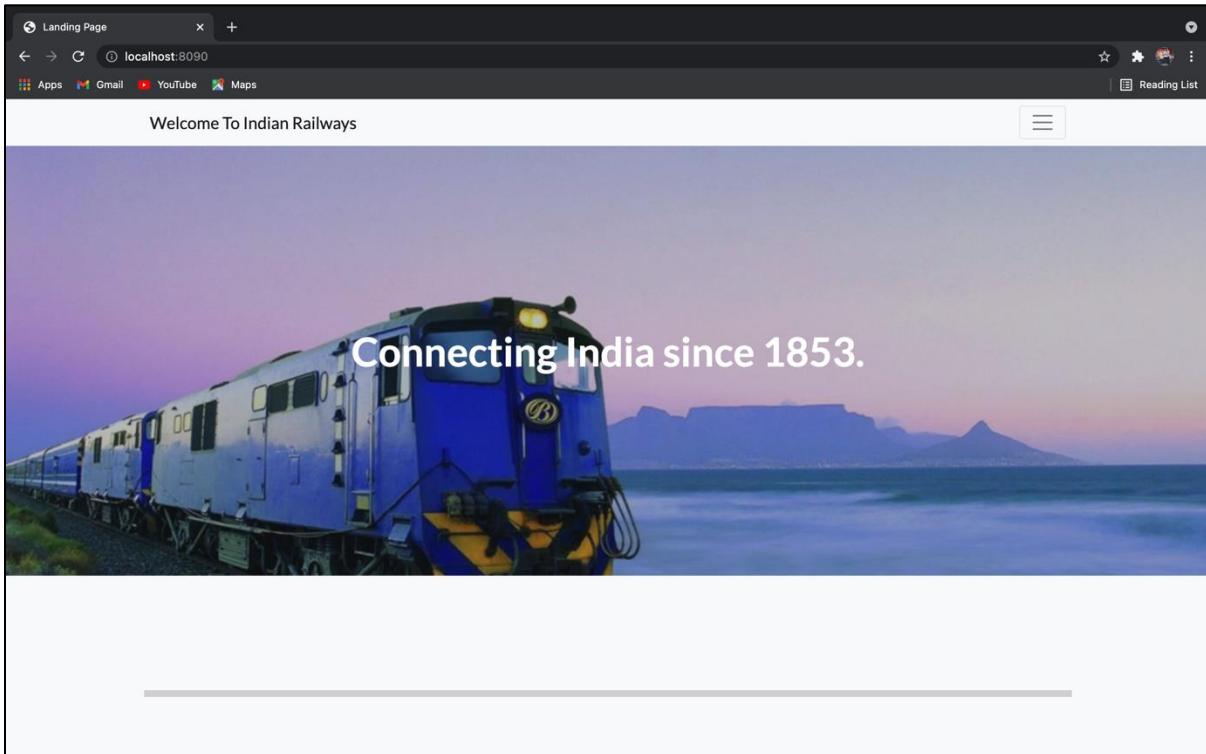
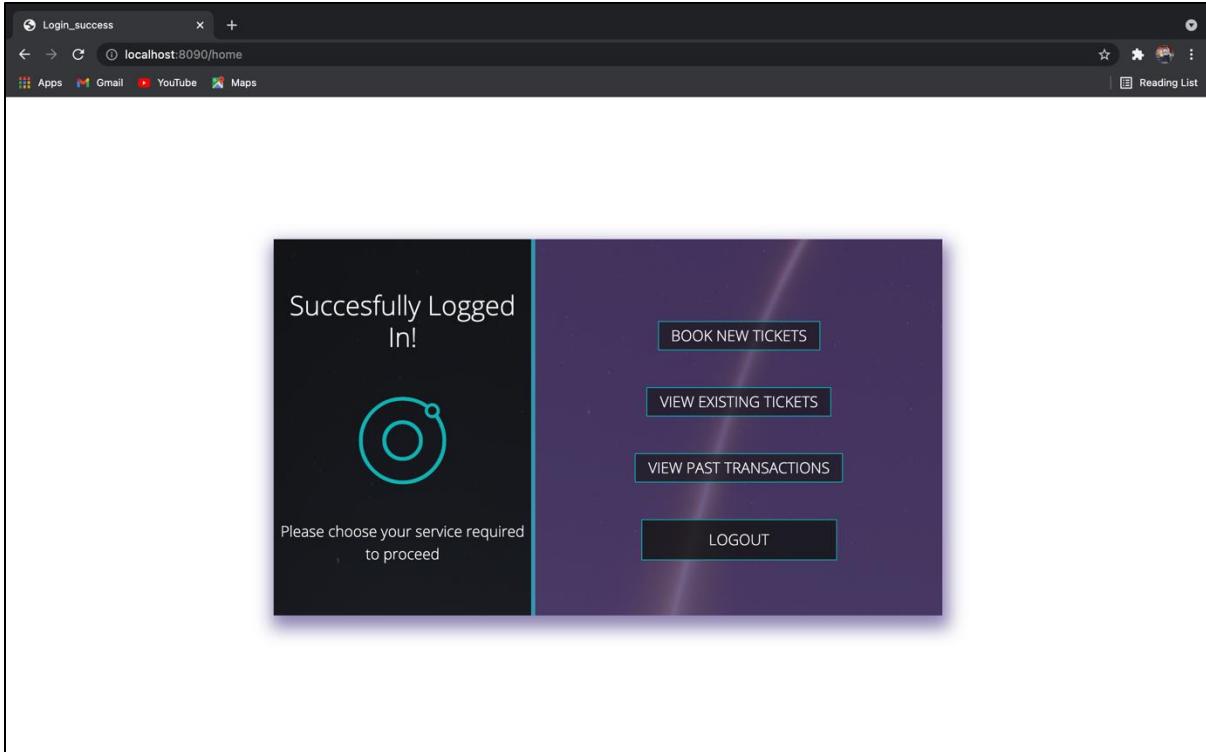
[Back to Home page](#)

```
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



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/>

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/>

