CS-207

APPLIED DATABASES PRACTICUM

Made By-

- Bhumesh Gaur

PROBLEM STATEMENT

Railway Booking System

TABLE OF CONTENTS

FRONTEND

- HTML
- CSS
- JavaScript

LOGIN AND REGISTER PAGE

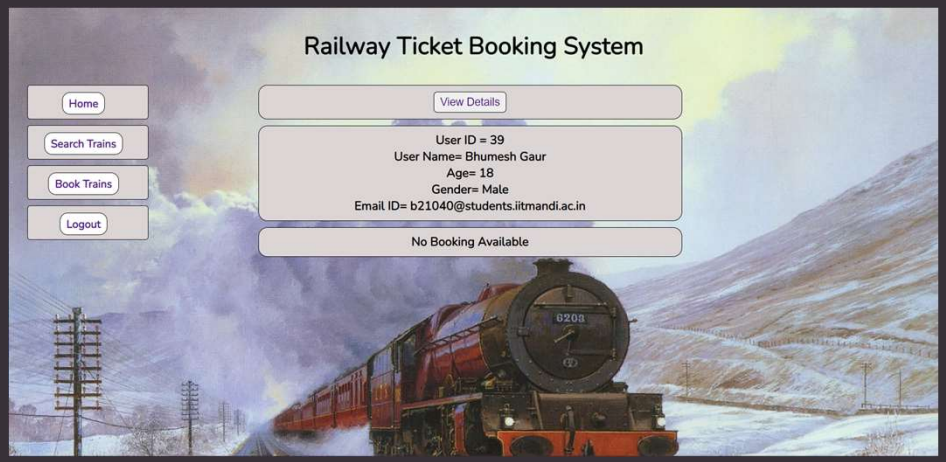
REGISTER PAGE

The user enters the credentials for which an account has to be created, and secures it with a password

LOGIN PAGE

The user logins with his username and password, and gets redirected to the home page of their account.

HOME PAGE



The logged in user gets an option to view his/her booked seats, and a side panel to navigate to various sections; and to logout.

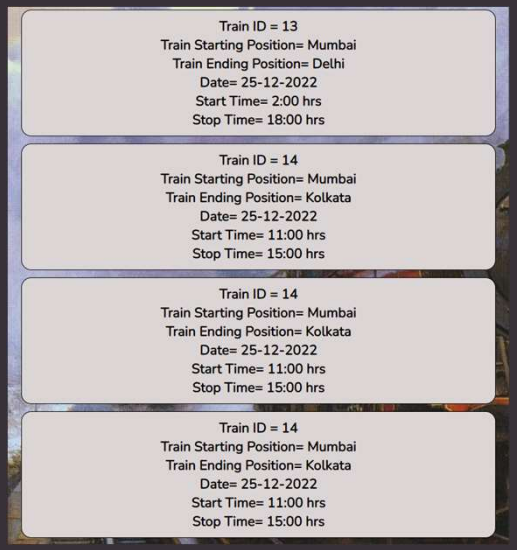
BOOKINGS

TRAIN SEARCH

The user can enter the cities and date to check trains and seats remaining in those trains.

BOOKING

Using train ID, the user can request to book certain number of seats in the train.



VIEW BOOKINGS

The user can view the details of his/her booked seats on the home page

BACKEND

- MySQL
- ExpressJS

BACKEND Demonstration

- We are using APIs in members.js file so that we can access the data from database to our frontend.
- Using Router we can send response to our desired data from the database and using javascript we can access the desired data on our front end page

```
const express=require('express');
const path=require('path');
const cors = require('cors');
const app=express();

app.use(cors());
const PORT=process.env.port || 5000;
const { ensureAsync } = require('async');
var mysql = require('mysql');
const db = mysql.createConnection({
        host: 'localhost',
        user: 'root', password: '',
        database: 'adp',
    });
    db.connect((err) => {
        if(err){throw err;}
        console.log('Mysql: Connected');
    });
    db.promise = (sql) => {
        return new Promise((resolve, reject) => {
            db.query(sql, (err, result) => {
              if(err){reject(new Error());}
              else{resolve(result);}
            });
        });
    };
connection= async ()=>{
  var result = await db.promise("SELECT * FROM users");
  return result;
}
connection().then((result)=>{
  var members=(result);
  console.log(members);
  module.exports = members;
  app.use(express.json());
  app.use('/api/members',require('./members'));
  app.listen(PORT,()=>console.log(`Server started on port: ${PORT}`));
  console.log(members);
})
```

APIs :

API stands for Application Programming Interface. APIs are mechanisms that enable two software components to communicate with each other using a set of definitions and protocols.

```
//login check
router.post('/login',(req,res)=>{
  console.log(JSON.stringify(req.body));
  const newmember={
    user_email:req.body.user_email,
    user_password:req.body.user_password
}
console.log(newmember.user_password);
connection= async ()=>{
  var result = await db.promise("SELECT * FROM users");
  // console.log(result);
  return result;
}
connection().then((result)=>{
  var members=(result);
  const found1=members.some((members=>members.user_email===(newmember.user_email)));
  const found2=members.some((members=>members.user_password===(newmember.user_password)));
  if(found1 && found2){
      var s=(members.filter(members=>members.user_email===(newmember.user_email)));
      console.log(s[0].user_id);
      res.json(s[0]);
  }
  else{
      res.json("invalid login credentials");
  }});
});
```

DATABASE

A self created database using MySQL is used for the trains. The database consists of three tables

Users Table

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	user_id	int(11)			No	None		AUTO_INCREMENT	Change Drop More
2	user_name	varchar(255)	utf8mb4_general_ci		Yes	NULL			Change Drop More
3	user_password	varchar(255)	utf8mb4_general_ci		Yes	NULL			Change Drop More
4	age	varchar(255)	utf8mb4_general_ci		Yes	NULL			Change Drop More
5	gender	varchar(255)	utf8mb4_general_ci		No	None			Change Drop More
6	user_email	varchar(255)	utf8mb4_general_ci		Yes	NULL			Change Drop More

Bookings table

	booking_id	user_id	train_id	seats
Edit Copy Delete	21	40	13	1
Edit Copy Delete	22	40	14	1
Edit Copy Delete	23	40	14	1
Edit Copy Delete	24	40	14	1

Trains Table

Server: 127.0.0.1 » Database: irctc » Table: trains

Browse | Structure | SQL | Search | Insert | Export | Import | Privileges | Operations

	train_id	t_source	t_destination	t_date	t_start_time	t_stop_time	t_seats	t_price
Edit Copy Delete	12	Mumbai	Delhi	25-12-2022	10:00 hrs	16:00 hrs	0	2000
Edit Copy Delete	13	Mumbai	Delhi	25-12-2022	2:00 hrs	18:00 hrs	2	2000
Edit Copy Delete	14	Mumbai	Kolkata	25-12-2022	11:00 hrs	15:00 hrs	3	1000
Edit Copy Delete	15	Mumbai	Kolkata	25-12-2022	16:00 hrs	21:00 hrs	3	1000
Edit Copy Delete	16	Mumbai	Kolkata	26-12-2022	11:00 hrs	15:00 hrs	3	1000
Edit Copy Delete	17	Mumbai	Kolkata	26-12-2022	16:00 hrs	21:00 hrs	3	1000
Edit Copy Delete	18	Mumbai	Kanpur	25-12-2022	2:00 hrs	18:00 hrs	3	2000
Edit Copy Delete	19	Mumbai	Kanpur	25-12-2022	6:00 hrs	23:00 hrs	3	2000
Edit Copy Delete	20	Delhi	Kanpur	24-12-2022	9:00 hrs	15:00 hrs	3	1500
Edit Copy Delete	21	Delhi	Kanpur	24-12-2022	13:00 hrs	19:00 hrs	3	1500
Edit Copy Delete	22	Delhi	Kolkata	23-12-2022	10:00 hrs	23:00 hrs	3	1300
Edit Copy Delete	23	Delhi	Kolkata	23-12-2022	3:00 hrs	19:00 hrs	3	1300
Edit Copy Delete	24	Kolkata	Kanpur	26-12-2022	2:00 hrs	12:00 hrs	3	1000
Edit Copy Delete	25	Kolkata	Kanpur	26-12-2022	5:00 hrs	15:00 hrs	3	1000
Edit Copy Delete	26	Madras	Delhi	23-12-2022	11:00 hrs	16:00 hrs	3	3500
Edit Copy Delete	27	Madras	Delhi	23-12-2022	116:00 hrs	21:00 hrs	3	3500
Edit Copy Delete	28	Mumbai	Madras	25-12-2022	10:00 hrs	15:00 hrs	3	1000
Edit Copy Delete	29	Mumbai	Madras	25-12-2022	14:00 hrs	19:00 hrs	3	1000

PASSWORD ENCRYPTION

The password submitted by user for registering is encrypted using SHA-256 model to ensure that even if someone gets access to backend, he does not have any the passwords of the users.

THANK YOU!!