N.Srividhya
185002103

Creating a Simple Web Application Using MEAN Stack

Exp No.12 **Date:1-05-21**

Objective:

- To Write a simple web application using MEAN stack
- To make a Sign Up and Login page using Mean stack which allows new users to sign up and inserts their details in the database and the existing user to login. It validates if the user is present in the database and if the entered userid and password is valid. If found valid it allows users to book tickets in my tourism website.

Requirements:

- **Front End Requirements:**
  1. Web Browser HTML support
  2. HTML
  3. CSS
  4. Javascript

- **Back End Requirements:**
  1. NodeJS
  2. Express
  3. MongoDB
  4. Mongoose
  5. Body-Parser

- **Text Editor/IDE**

Theory:

- **MongoDB:**
  - MongoDB is an open-source document database and leading NoSQL database. MongoDB is written in C++.
  - Rich and expressive query language that allows you to filter and sort by any field, no matter how nested it may be within a document.
  - Support for aggregations and other modern use-cases such as geo-based search, graph search, and text search.
  - Queries are themselves JSON, and thus easily composable. No more concatenating strings to dynamically generate SQL queries**.**
  - Mongoose is a MongoDB object Modelling tool which provides a straight-forward, schema-based solution to model your application data. It includes built-in type casting, validation, query building, business logic hooks and more, out of the box.

- **Node js:**
  - Node js is a free and an open source environment
  - It used javascript to run on server.
  - Nodejs is used to generate dynamic page content.
  - It executes javascript code outside browser.

- **Express:**
  - Express is a minimal and flexible Node.js web application framework that provides a robust set of features to develop web and mobile applications. It facilitates the rapid development of Node based Web applications. Following are some of the core features of Express framework –
  - Allows to set up middlewares to respond to HTTP Requests.
  - Defines a routing table which is used to perform different actions based on HTTP Method and URL.

Procedure:

1. Install Mongodb server.
2. In the windows C drive create a folder data and another folder db within data.
3. Create file app.js using npm init.
4. Install dependencies-express, mongoose, body-parser.
5. Create new database usersdb using mongoose.
6. Create login, signup UI for getting input from the user.
7. Insert the data into userdb for new users using body-parser.
8. For login check if the user and password exist in the database and if they are correct. Otherwise prompt the user to signup.
9. Start the server using node server.js in cmd.
10. In cmd go to the directory where mongodb is installed and open the bin folder and open cmd there and type mongod.exe.
11. In cmd go to the directory where mongodb is installed and open the bin folder and open cmd there and type mongo.exe.
12. Open login and signup page http://localhost:3000/login and http://localhost:3000 in the browser.
13. In mongo.exe cmd type use <database name> and db.users.find() to view the data entered.

Implementation:

app.js

```
const express=require("express");
const bodyParser=require("body-parser");
var app=express();

const mongoose=require('mongoose');

mongoose.connect("mongodb://localhost:27017/userDB",{ useNewUrlParser: true,
useUnifiedTopology: true });
app.use(bodyParser.urlencoded({extended:true}));

const userSchema=new mongoose.Schema({
    username:String,
    password:String,
    email:String,
    mobile:Number

   });

const User=mongoose.model("User",userSchema);

app.get("/",function(req,res){
  res.sendFile(__dirname+"/signup.html");
});

app.get("/fail",function(req,res){
    res.sendFile(__dirname+"/unsucess.html");
 });

app.get("/login",function(req,res){
    res.sendFile(__dirname+"/login.html");
 });

app.use("/", express.static(__dirname + "/"));
app.post("/",function(req,res){
    console.log("Welcome");
    const uname=req.body.username;
    const pw=req.body.pwd;
    const em=req.body.email;
    const mo=req.body.mobile;

    const newuser=new User({

    username:uname,
```

```
    password:pw,
    email:em,
    mobile:mo

    });



  User.insertMany([newuser],function(err){
     if(err)
     {
        console.log(err);
     }
     else
     {
      res.redirect('http://localhost:3000/login');
     }
  });



});

app.post("/login",function(req,res){
    console.log("Welcome to Login Page");
    const uname=req.body.username;
    const pw=req.body.pwd;

    User.find({$and:[{ username:uname, password:pw}]}, function (err,doc) {
        if (doc.length==0||err){
            /*res.send("Login Unsuccessfull!!Sign Up First!");*/
            res.redirect('http://localhost:3000/fail');
        }
        else{
           res.redirect("ticket.html");
        }
    });



});

app.post("/fail",function(req,res){
    console.log("Welcome");
    const uname=req.body.username;
    const pw=req.body.pwd;
    const em=req.body.email;
    const mo=req.body.mobile;
```

```
    const newuser=new User({

    username:uname,
    password:pw,
    email:em,
    mobile:mo

    });



   User.insertMany([newuser],function(err){
     if(err)
     {
        console.log(err);
     }
     else
     {
       res.send("You signed up Successfully");
     }
   });



 });



 app.listen(3000,function(req,res){
   console.log("Server is listening on port 3000");
});
```

signup.html

```
 <html>
    <head>
       <link rel="stylesheet" href="styles.css">
    </head>
    <title>Sign Up page</title>
    <body>
       <center><h1><i>Sign Up</i></h1></center>
       <div id="div">
       <form action="/" method="POST">
          <p id="name">Enter UserName:<input type="text" name="username"></p>
          <p id="pwd">Enter Password:<input type="password" name="pwd" ></p>
          <p id="mail">Enter Email:<input type="text" name="email"></p>
```

```
        <p id="num">Enter Mobile Number:<input type="number" name="mobile"></p>
        <p><input type="submit" value="Sign Up" id="btn"></p>
        <p id="already">Already have an account?<a href="http://localhost:3000/login"><span
 id="login">Login</span></p>
      </form>
    </div>
    </body>
</html>
```

login.html

```
 <html>
    <head><link rel="stylesheet" href="styles.css"></head>
    <title>Login Page</title>
    <body>
      <center><h1><i>Login to Book Tickets</i></h1></center>
      <div id="div">
      <form action="/login" method="POST">
        <p id="name">Enter UserName:<input type="text" name="username"></p>
        <p id="pwd">Enter Password:<input type="password" name="pwd"></p>
        <p><input type="submit" value="Login" id="btn"></p>
        <p id="already">Do not have an account?<a href="http://localhost:3000"><span
 id="signup">Sign Up</span></p>
      </form>
    </div>
    </body>
</html>
```

unsuccess.html

```
 <html>
    <head><link rel="stylesheet" href="styles.css"></head>
    <title>Sign Up page</title>
    <body>
      <center><h1><i>Sign Up</i></h1></center>
      <div id="div">
      <p id="uns">Login Unsucessfull !! Sign Up first</p>
      <form action="/fail" method="POST">
        <p id="name">Enter UserName:<input type="text" name="username"></p>
        <p id="pwd">Enter Password:<input type="password" name="pwd"></p>
        <p id="mail">Enter Email:<input type="text" name="email"></p>
        <p id="num">Enter Mobile Number:<input type="number" name="mobile"></p>
        <p><input type="submit" value="Sign Up" id="btn"></p>
      </form>
    </div>
    </body>
</html>
```

N.Srividhya
185002103

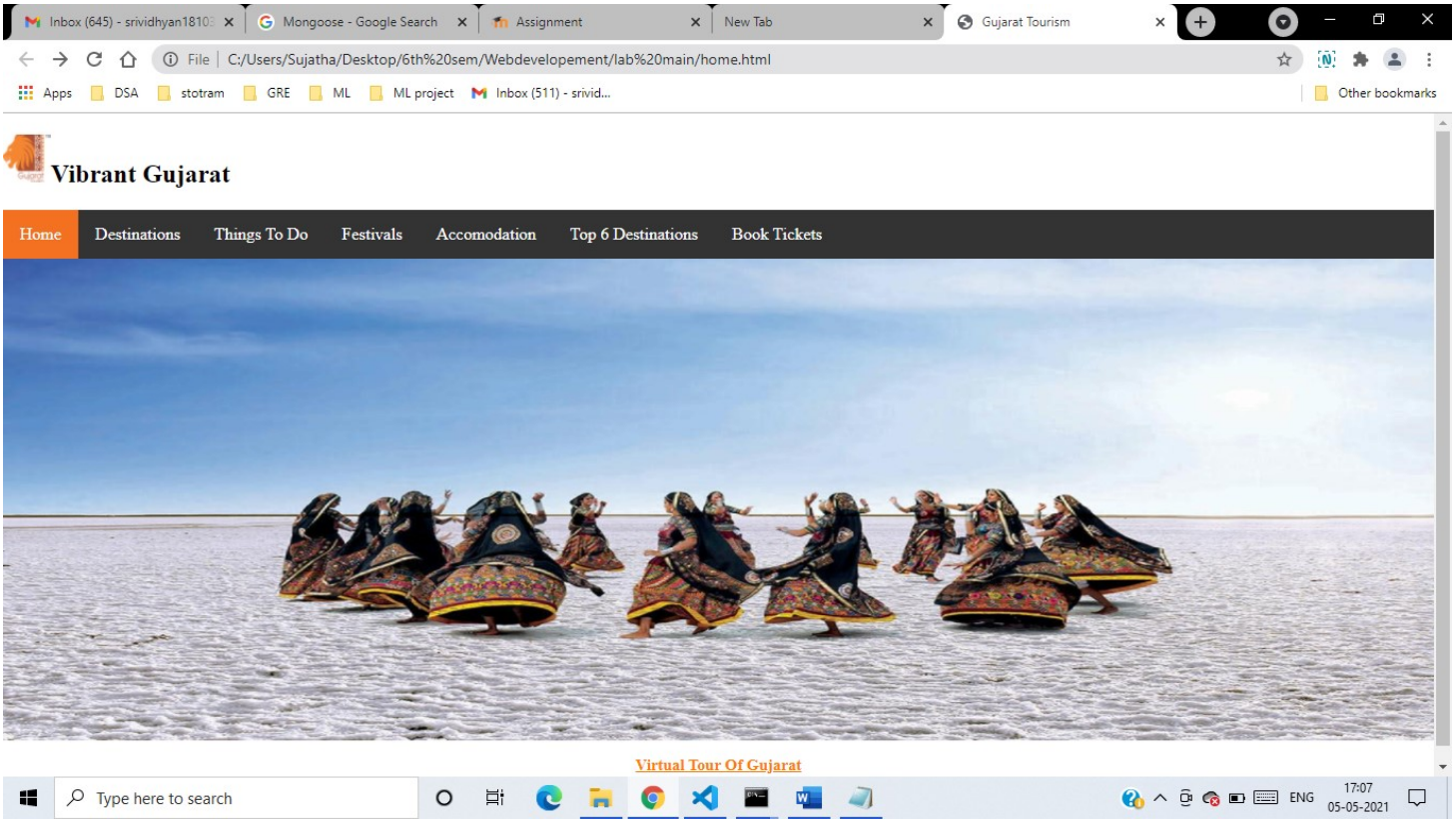
Styles.css

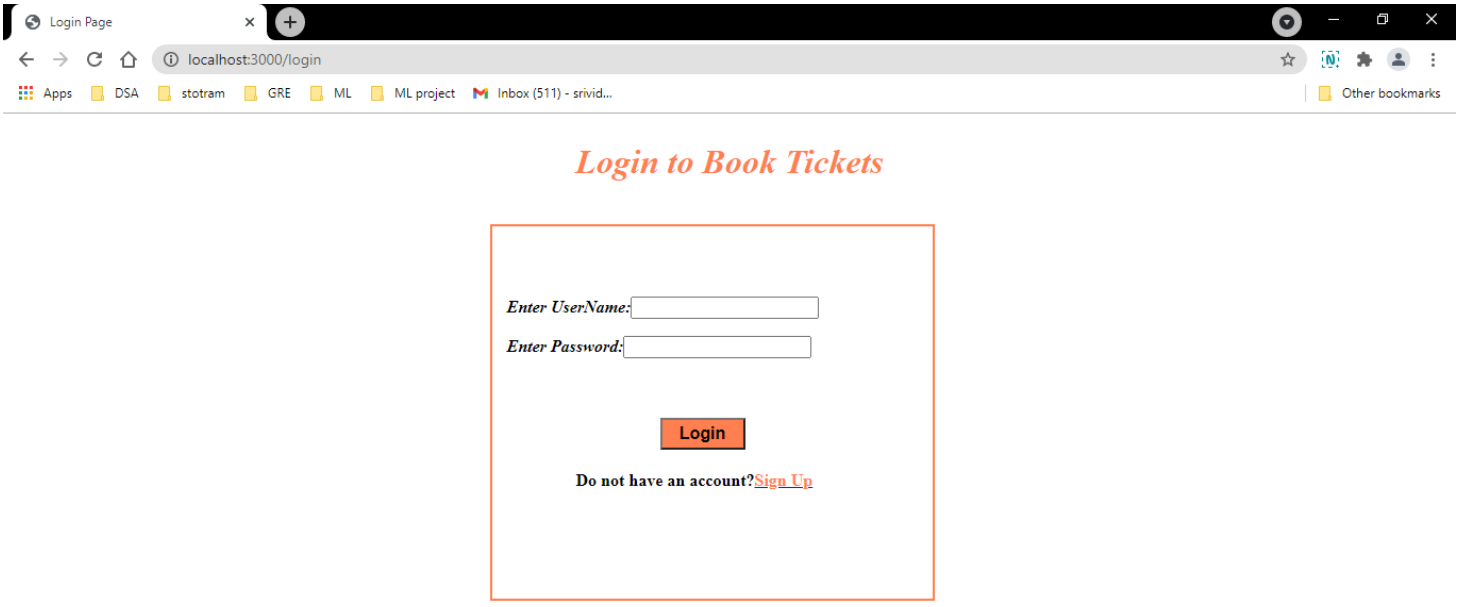
```
#div{
    border:2px solid coral;
    height:300px;
    width:400px;
    padding-left:1%;
    padding-top:50px;
    margin-top: 40px;
    margin-left:450px;
}
h1{
    color:coral;
    margin-top:2%;

}

#login,#signup{
    color: coral;
    font-weight: bold;
}

#already{
    margin-left: 65px;
    margin-top: 20px;
    font-weight: bold;

}

#name,#mail,#num,#pwd{
    font-weight:bold;
    font-size: medium;;
    font-style: italic;

}

#btn{
    margin-left:36%;
    margin-top:10%;
    font-size: medium;
    font-weight: bold;
    background-color: coral;
    width:80px;
    height:30px;
}
#uns{
    font-weight:bold;
    font-style: italic;
    color: crimson;
```

```
    font-size: 18px;
}
```

Output:

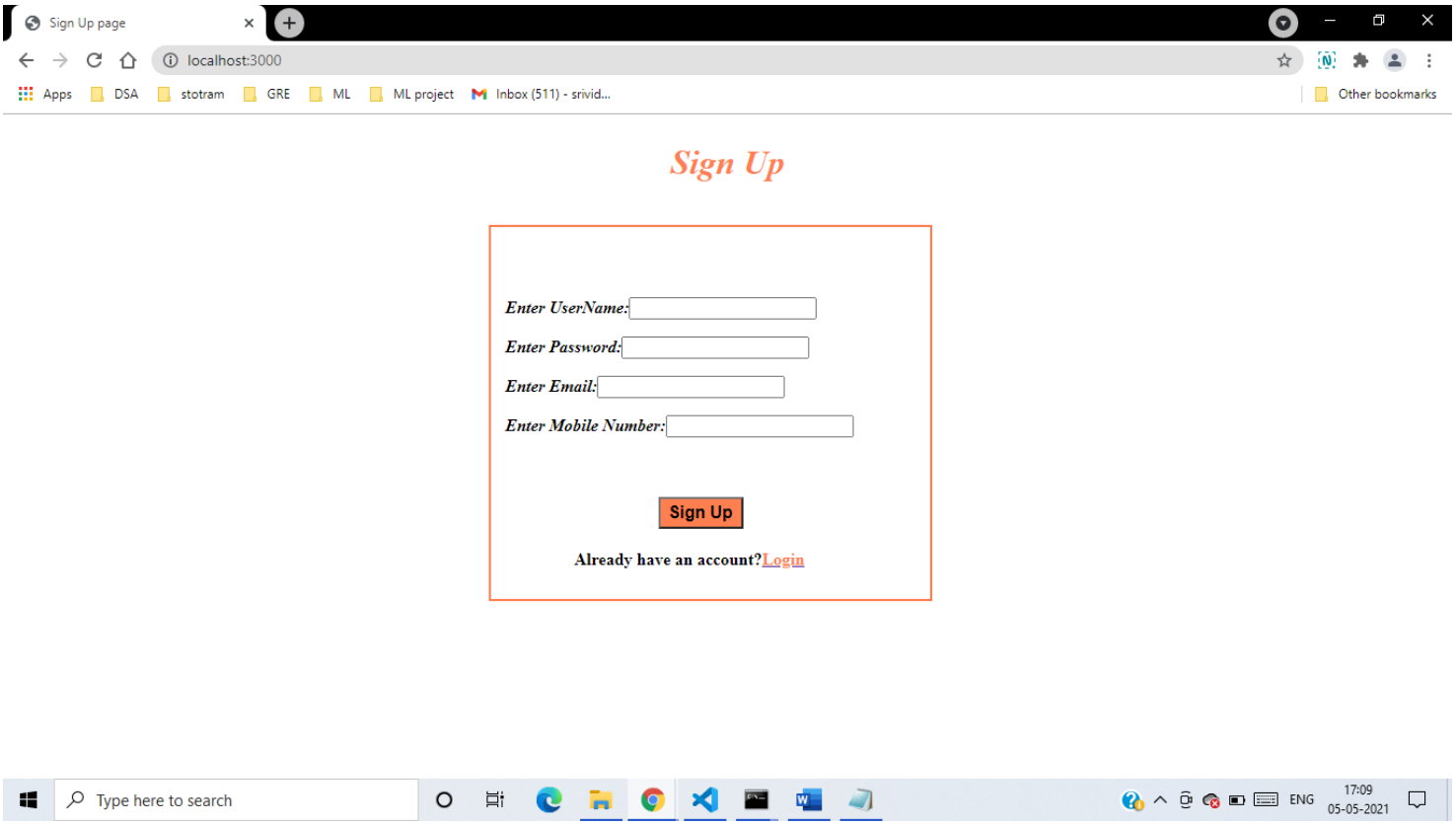
On clicking book tickets it will take user to the login page.

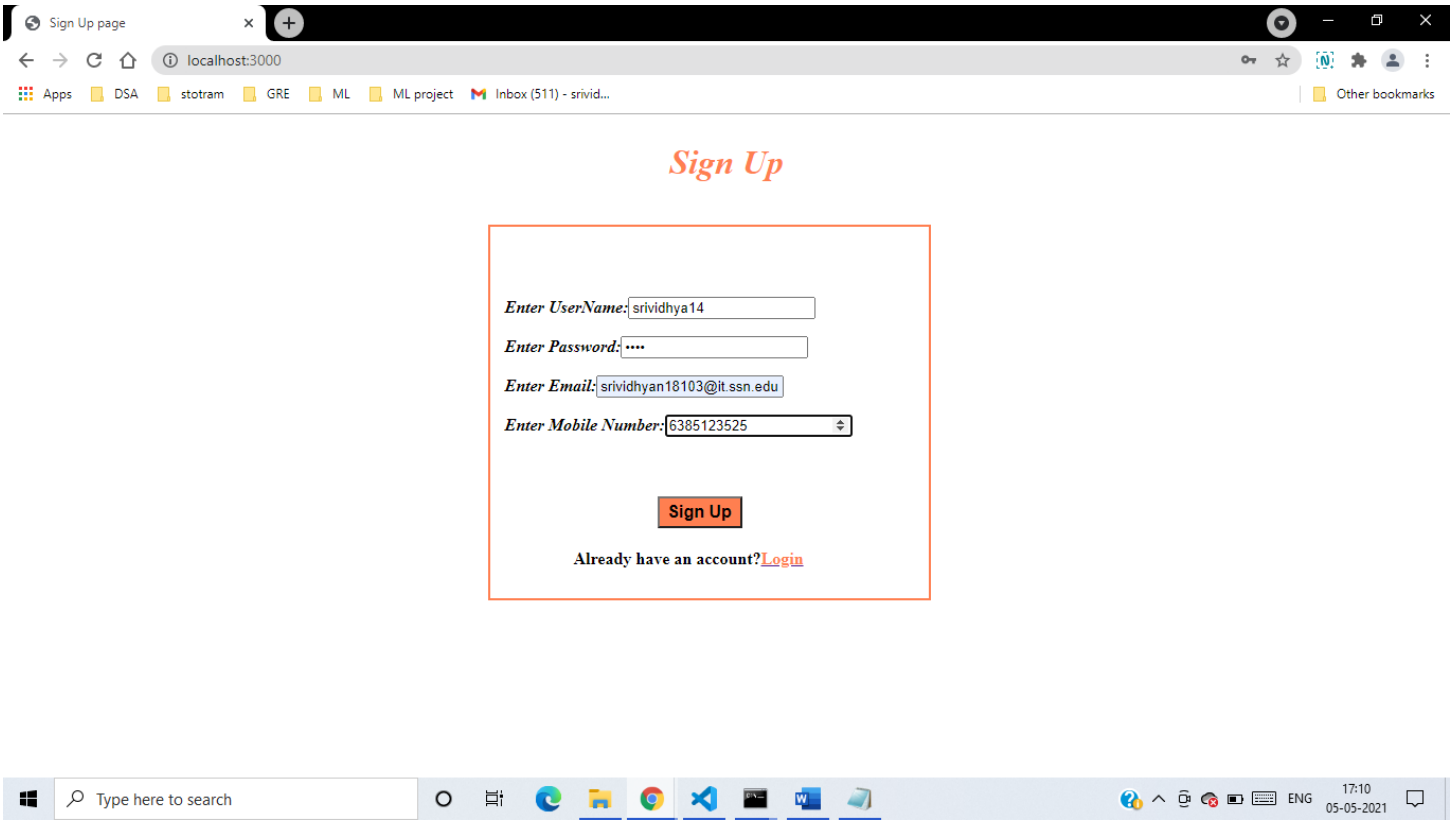




N.Srividhya
185002103

If it's a new user can sign up.





N.Srividhya
185002103

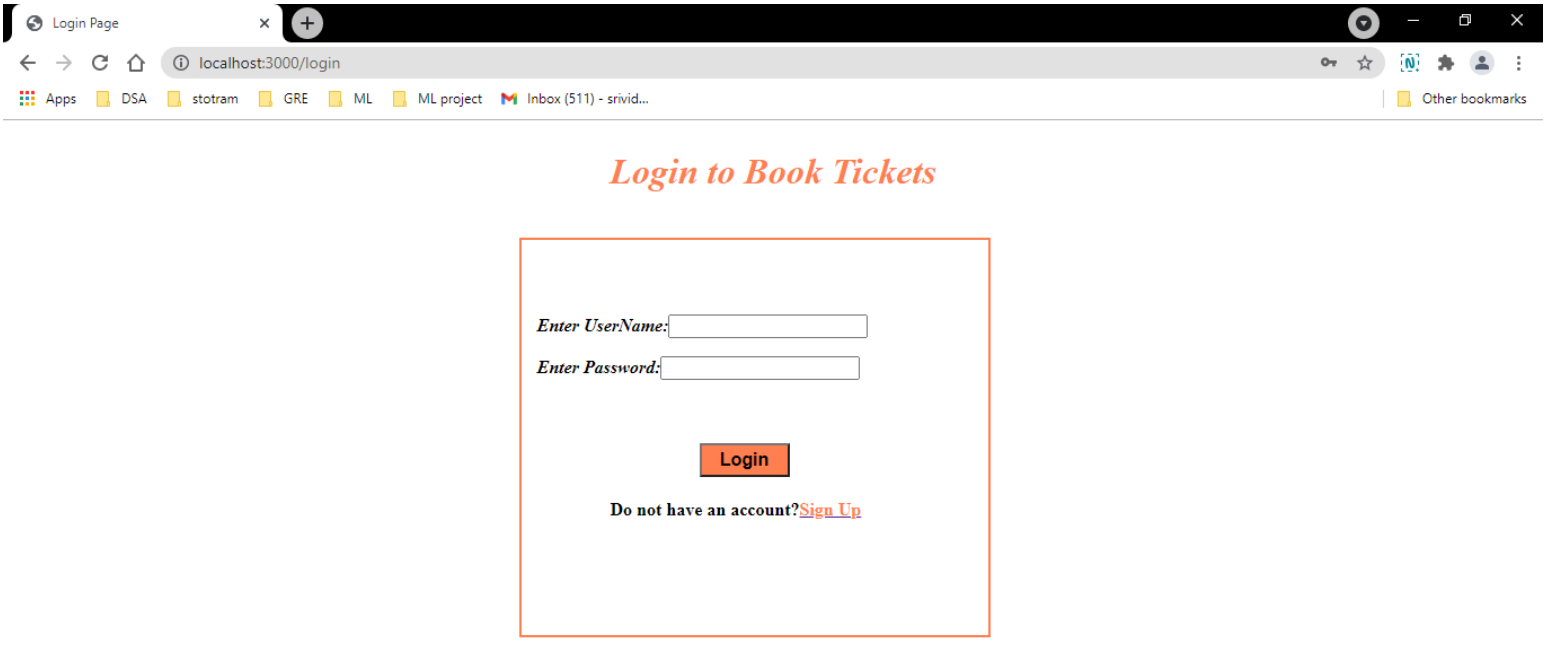
Userdb is users database.

```
> show dbs
admin     0.000GB
config    0.000GB
fruitsDB  0.000GB
local     0.000GB
shopDB    0.000GB
userDB    0.000GB
> use userDB
switched to db userDB
```

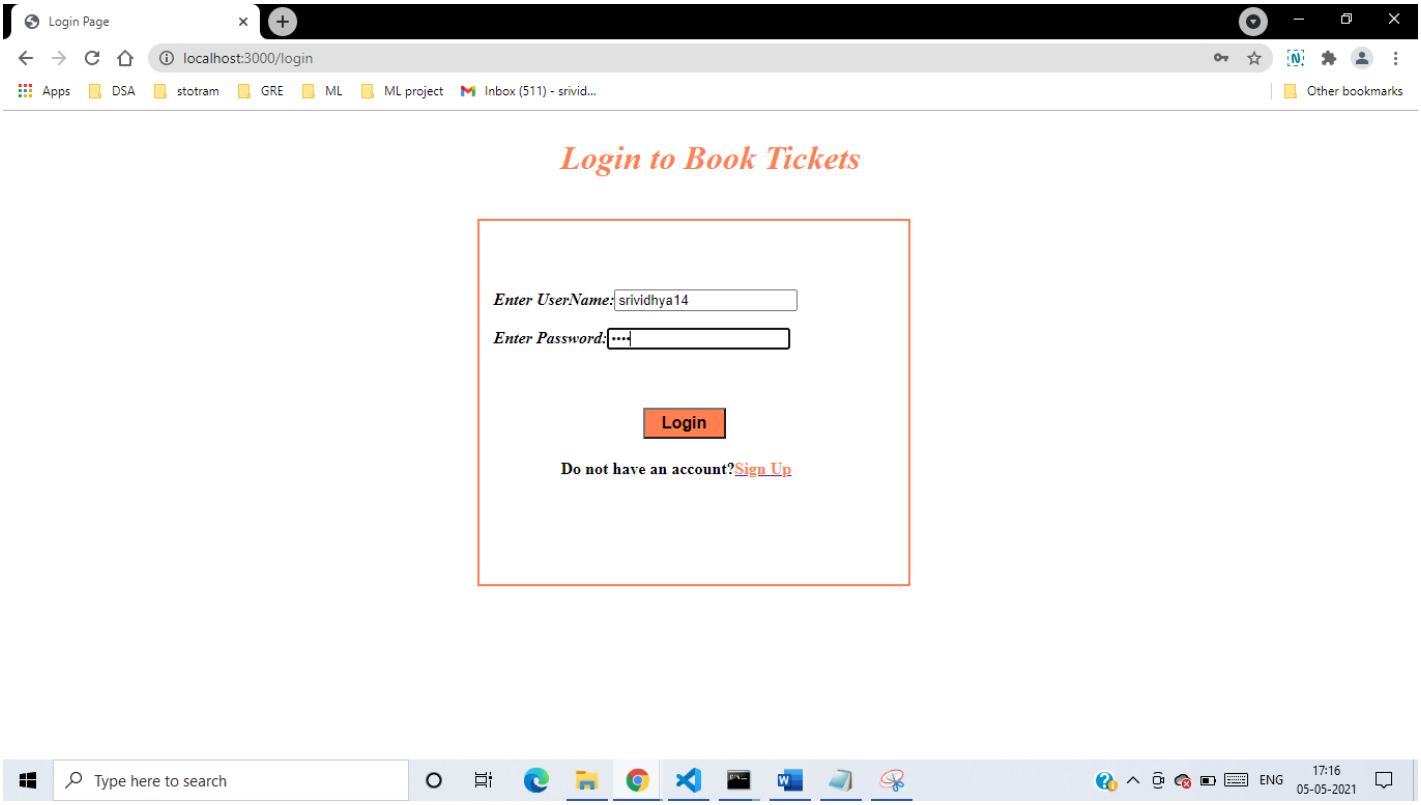
The new user data has been entered into users collection inside userDB.

```
> show collections
users
> db.users.find()
{ "_id" : ObjectId("60923470e30b5923c00047b9"), "username" : "Divya", "password" : "divi", "email" : "divya.narayanan216@gmail.com", "mobile" : 123456789, "__v" : 0 }
{ "_id" : ObjectId("6092402ed8fb801d3c747cca"), "username" : "srividhya_14", "password" : "Vaishu", "email" : "srividhya.narayanan990@gmail.com", "mobile" : 6385123525,
 "__v" : 0 }
{ "_id" : ObjectId("60924968b1324c3c144dce88"), "username" : "Sujatha", "password" : "suja", "email" : "sujatha@gmail.com", "mobile" : 9998981845, "__v" : 0 }
{ "_id" : ObjectId("60925b3aadcd8050708f08dc"), "username" : "kesha7", "password" : "kesha", "email" : "123@gmail.com", "mobile" : 1234, "__v" : 0 }
{ "_id" : ObjectId("609268e025835813944e523c"), "username" : "suresh26", "password" : "suresh", "email" : "suresh26@gmail.com", "mobile" : 9998215586, "__v" : 0 }
{ "_id" : ObjectId("6092693564c9dc3160b472f6"), "username" : "laasya8", "password" : "laasya", "email" : "vlaasya@gmail.com", "mobile" : 99854526, "__v" : 0 }
{ "_id" : ObjectId("6092696764c9dc3160b472f7"), "username" : "sujatha22", "password" : "sujatha", "email" : "srividhya.narayanan990@gmail.com", "mobile" : 12345, "__v"
: 0 }
{ "_id" : ObjectId("6092844220b7f821b4e50c63"), "username" : "srividhya14", "password" : "1234", "email" : "srividhyan18103@it.ssn.edu.in", "mobile" : 6385123525, "__v"
 : 0 }
>
```

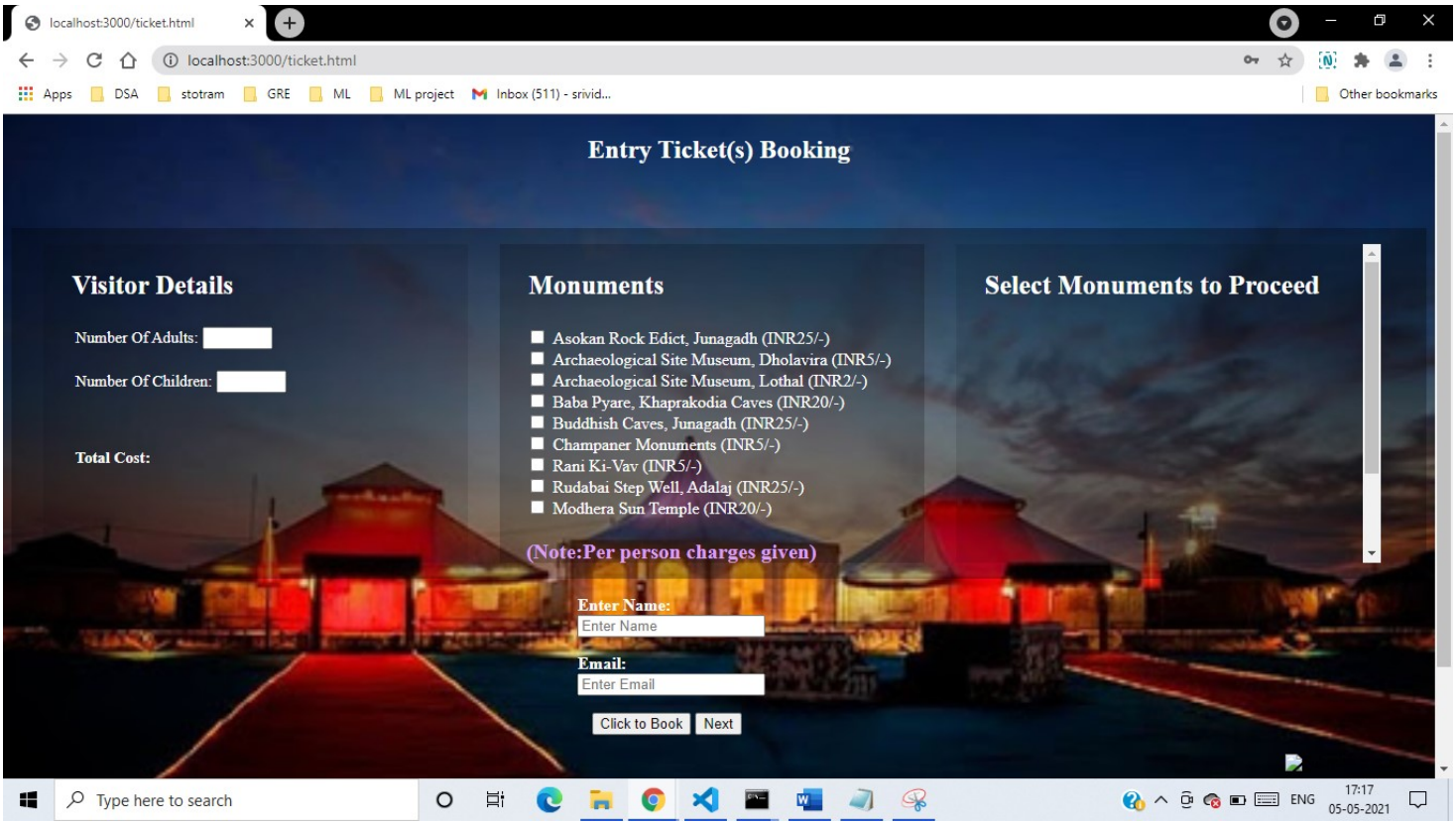
After successful sign up user is navigated to login page



N.Srividhya
185002103

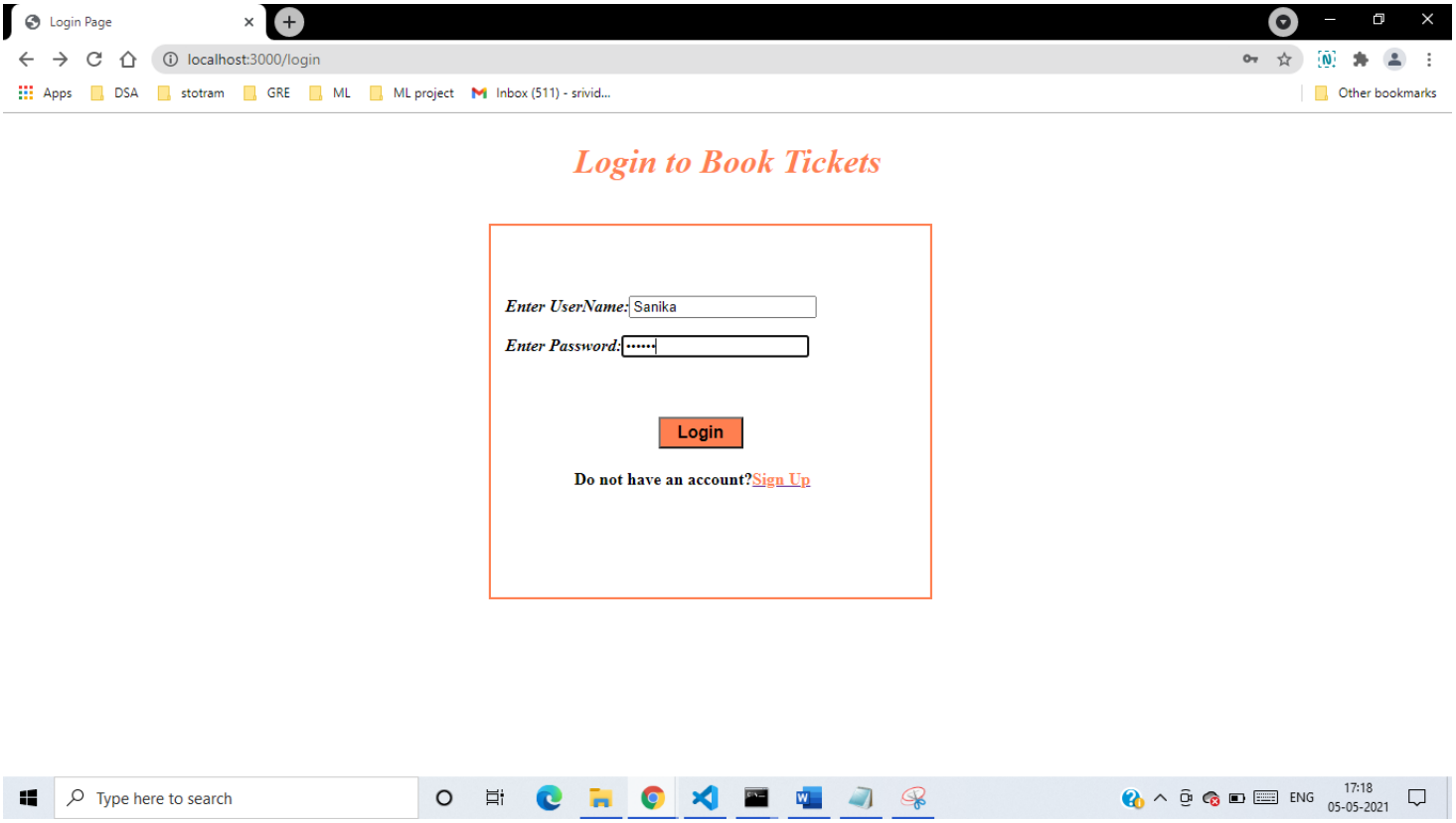


On successful login user can book tickets in the ticket booking page.

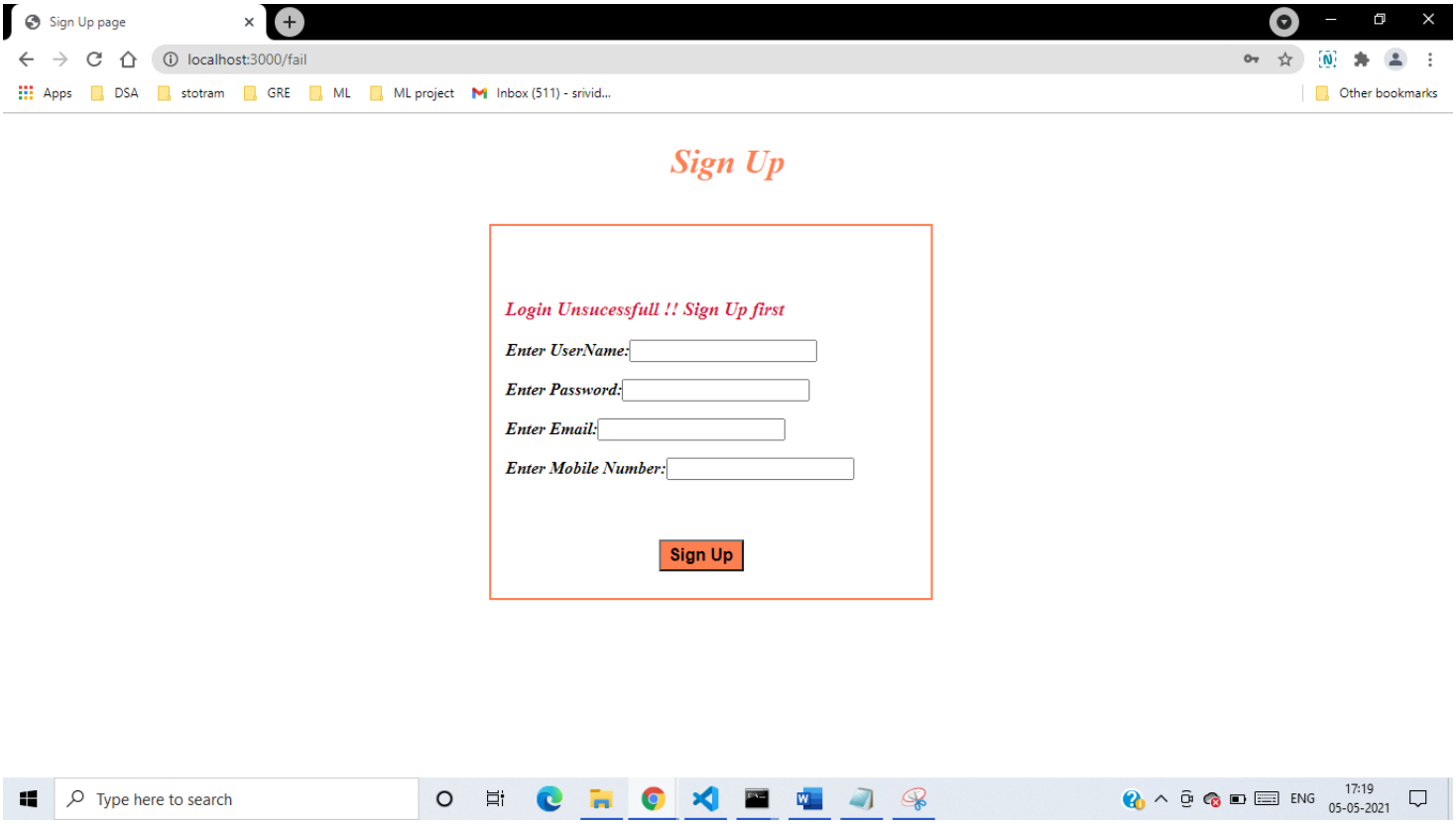


N.Srividhya
185002103

This user name doesn't exist in database.



Since the user does not exist Sign Up page asks user to register.



N.Srividhya
185002103

Result:

The simple web application using MEAN stack has been successfully written and executed.