A
PROJECT REPORT
ON

Online City Information

By

Machchhar Palna (CE-68)
Mahyavanshi Jenil (CE-69)
Shah Diya (CE-138)

B.Tech CE Semester-V
Subject: Advanced Technology

Guided by:
Prof.Prashant M. Jadav
Prof.Siddharth P. Shah

Faculty of Technology
Department of Computer Engineering
Dharmsinh Desai University

Faculty of Technology
Department of Computer Engineering
Dharmsinh Desai University

CERTIFICATE

This is to certify that the practical / term work carried out in the subject of **Advanced Technology** and recorded in this journal is the bonafide work of

Machchhar Palna (CE-68)
Mahyavanshi Jenil (CE-69)
Shah Diya (CE-138)

of B.Tech semester **V** in the branch of **Computer Engineering**

during the academic year **2020-2021**.

Contents:

1. Abstract

City Information System Provides a simple interface to see the details of a particular city. This can be used by travel agents, students or any particular user who is interested to search about city. The main purpose of this project is to help tourists and other visitors of an unknown city by providing information about famous places and food. This Project serves the visitors as a reliable and user-friendly guide of the city. It also contains map which shows exact location of famous places of a city.

2. Introduction

2.1 Brief Introduction

Whenever a person visiting a new city wants to get some information about the city, then they need to ask the locals or take help from the guide in the city. This requires lots of time and the person might need to ask multiple times. The design and implementation of City Information System and user interface is to replace this tedious job. The system utilizes user and admin authentication.

Admin can add the cities along with location and brief details about that city. Also he can add famous places and food details for particular city along with image and location. Admin can also update or delete details of city, place and food. User can see the all the cities name by logging in to the system. He/She can view his/her desirable city details by clicking on that particular city. User can also see exact location of all famous places by clicking on the map.

2.2 Tools/Technologies Used

Technologies:

- React Library
- Express
- NodeJS
- MongoDB
- Bootstrap
- JavaScript
- jQuery
- HTML
- CSS

Tools

- GitHub
- Visual Studio Code

Platform

- Local Host

3. Software Requirement Specifications

Admin

R.1 Login

Description: Admin can login by entering email and password.

Input: Enter email and password

Output: Show Admin Panel.

R.2 Add City

Description: Admin can add new city by entering city details like image, location etc.

Input: Details of city

Output: “confirmation message” (City added Successfully)

R.3 Add Place

Description: Admin can add new place by entering place details like image, location etc. and also select appropriate city.

Input: Details of place

Output: “confirmation message” (Place added Successfully)

R.4 Add Food

Description: Admin can add famous food by entering food details like image, location etc.

Input: Details of food

Output: “confirmation message” (Food added Successfully)

R.2 Update City Details

Description: Admin can update all city details.

Input: change value of input field.

Output: “confirmation message” (City detail updated Successfully)

R.2 Update Place Details

Description: Admin can update all place details along with changing its city.

Input: change value of input field.

Output: “confirmation message” (Place detail updated Successfully)

R.2 Update Food Details

Description: Admin can update all Food details like location, image etc .

Input: change value of input field

Output: “confirmation message” (Food detail updated Successfully)

R.2 Delete City

Description: Admin can Delete particular city

Input: By clicking delete button.

Output: “confirmation message” (City deleted Successfully)

R.2 Delete Place

Description: Admin can Delete particular place.

Input: By clicking delete button.

Output: “confirmation message” (Place deleted Successfully)

R.2 Delete Food

Description: Admin can Delete particular food

Input: By clicking delete button.

Output: “confirmation message” (Food deleted Successfully)

User

R.1 Register

Description: Users can register by giving their details.

Input: User details

Output: Shows Login Page

R.1 Login

Description: User can login by entering email and password.

Input: Enter Email and password

Output: Shows User Panel.

R.2 View City

Description: User can view city details by clicking button. It will display all places and food along with their locations on the map.

Input: Click on View button.

Output: Shows all the locations on the map.

R.3 Logout

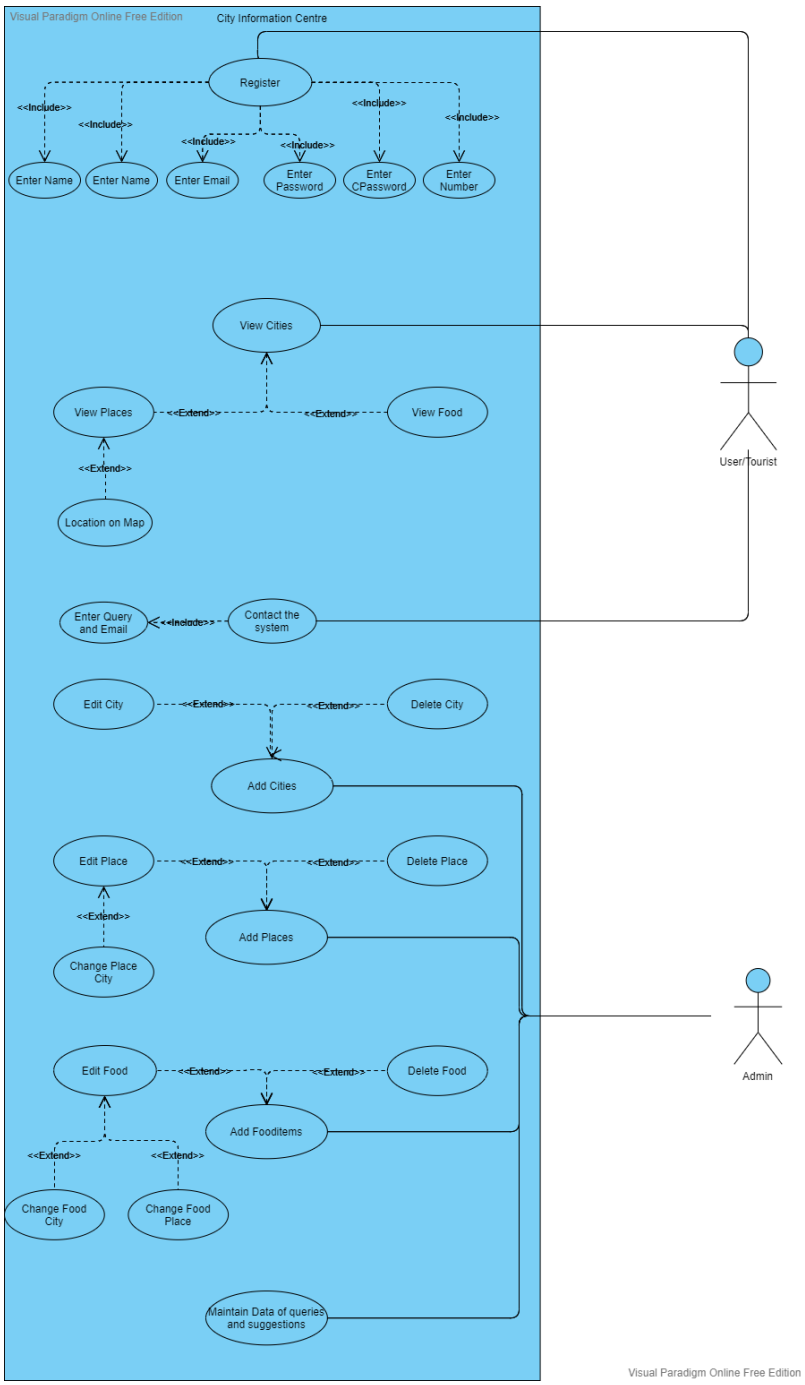
Description: User can logout by clicking on the logout button.

Input: click logout button.

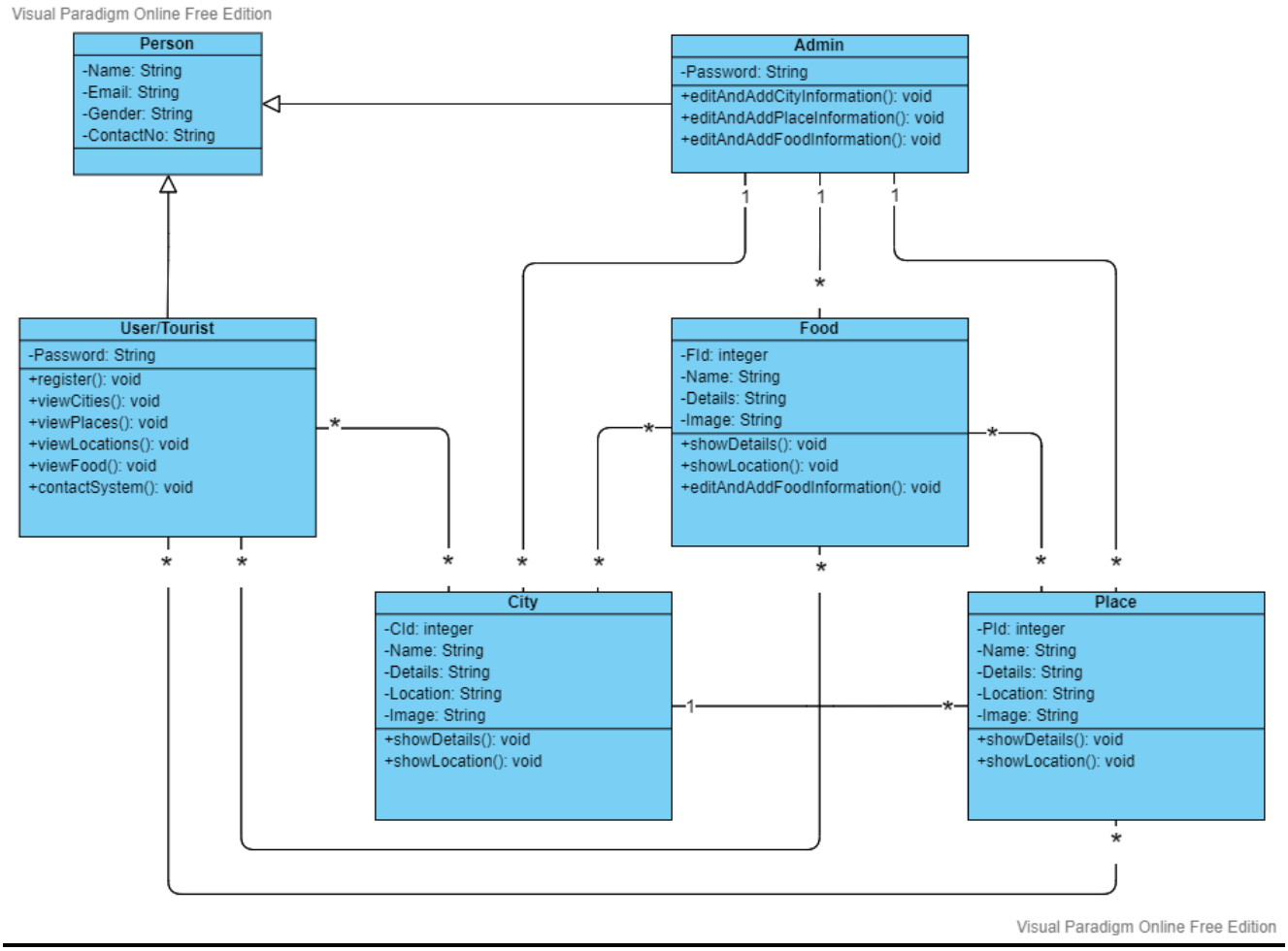
Output: Shows Login Panel.

4. Design

4.1 Use Case Diagram

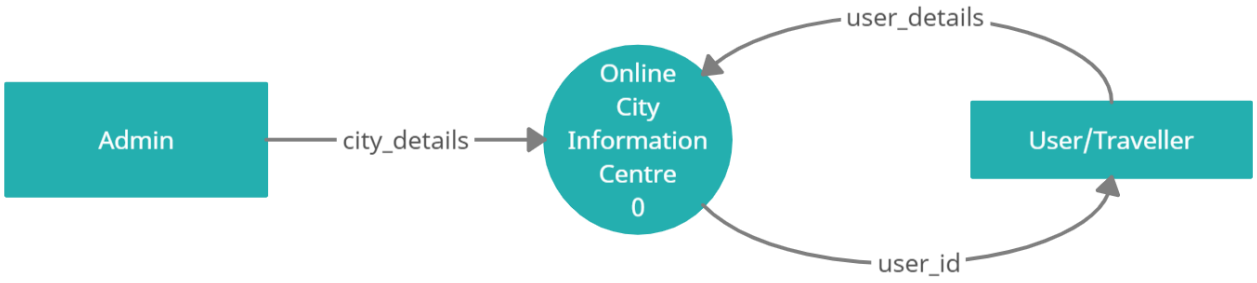


4.2 Class Diagram

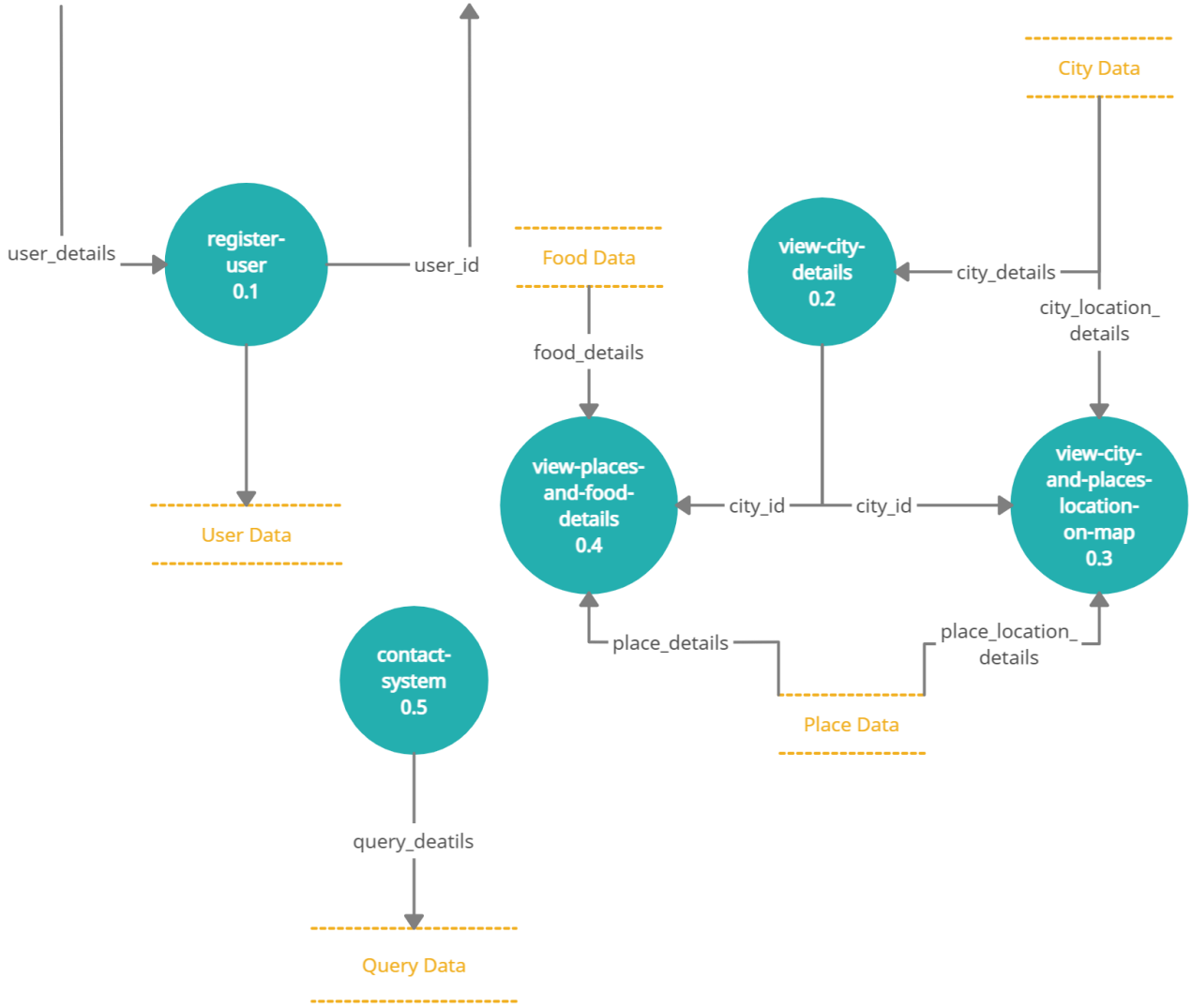


4.3 Data Flow Diagram

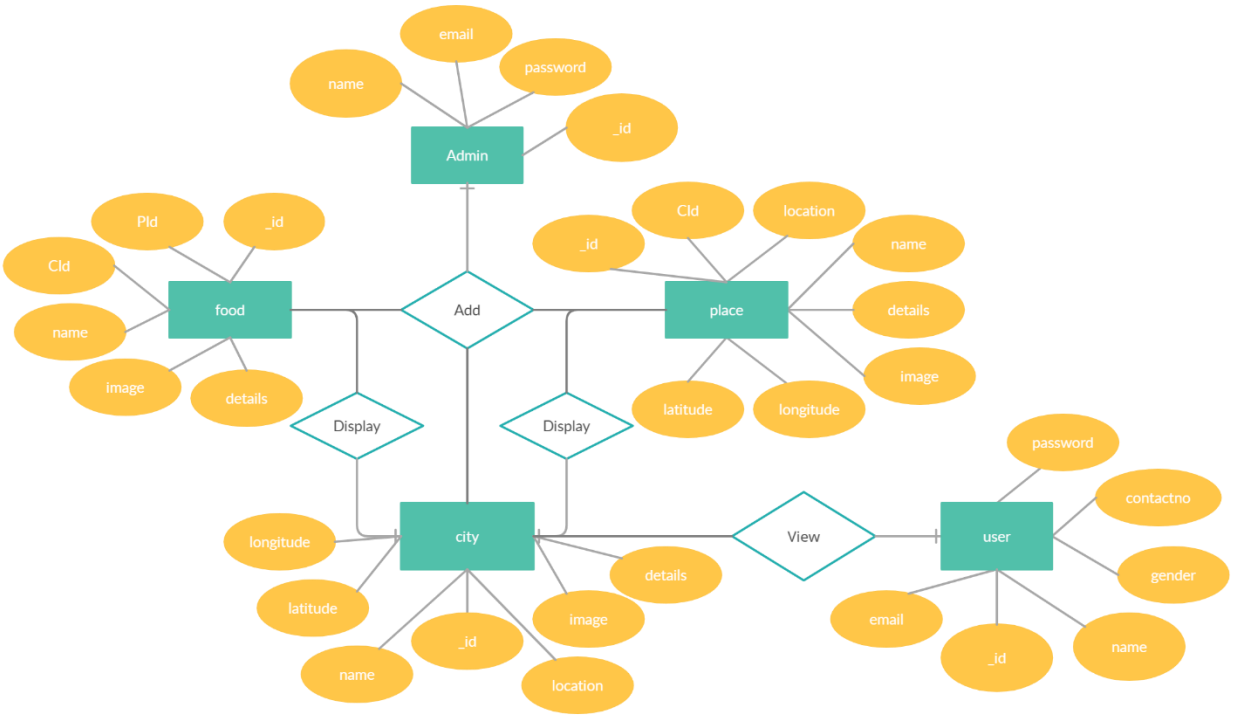
Level 0:



Level 1:


user_details
register-user 0.1
user_id
User Data
Food Data
food_details
view-city-details 0.2
city_details
City Data
city_location_details
view-places-and-food-details 0.4
city_id
city_id
view-city-and-places-location-on-map 0.3
place_details
place_location_details
Place Data
contact-system 0.5
query_deatils
Query Data

4.4 E-R Diagram



4.5 Data Dictionary

1. **User Schema**

```
const userSchema = new mongoose.Schema({
    name: {
        type : String,
        required : "Required"
    },
    email:{
        type : String,
        required : "Required"
    },
    password:{
        type : String,
        required : "Required"
    },
    cpassword:{
        type : String,
        required : "Required"
    },
    gender:{
        type : String,
        required : "Required"
    },
    contactNumber:{
        type : Number,
        required : "Required"
    }
});
```

2. **City Schema**

```
const citySchema = new mongoose.Schema({
    image: {
        type : String,
        required : "Required"
    },
    name: {
        type : String,
        required : "Required"
    },
    details:{
        type : String,
        required : "Required"
    },
    location:{
        type : String,
        required : "Required"
    },
    latitude:{
        type: String,
        required: "Required"
    },
    longitude:{
        type: String,
        required: "Required"
    }
});
```

3. **Place Schema**

```
const placeSchema = new mongoose.Schema({
    CId: {
        type : String , ref: 'city'
    },
    name:{
        type : String,
        required : "Required"
    },
    image:{
        type : String,
        required : "Required"
    },
    details:{
        type : String,
        required : "Required"
    },
   location:{
        type : String,
        required : "Required"
    },
    latitude:{
        type: String,
        required: "Required"
    },
    longitude:{
        type: String,
        required: "Required"
    }
});
```

4. **Food Schema**

```
const foodSchema = new mongoose.Schema({
    CId: {
        type : String , ref: 'city'
    },
    PId: {
        type : String , ref: 'place'
    },
    name:{
        type : String,
        required : "Required"
    },
    image:{
        type : String,
        required : "Required"
    },
    details:{
        type : String,
        required : "Required"
    },
});
```

5. Implementation Details

The system consists of 2 basic modules namely

1. User Module
2. Admin Module

Each module consists of several methods to implement the required functionality. Implementation is done using Express and NodeJs. Database used in these modules is MongoDB.

1. User Module

- This module is the base for user authentication to ensure the security aspect. It includes user login and registration details.
- It also displays all cities names along with their famous places and food.

2. Admin Module

- This module firstly authenticates admin credentials and then admin can see all the details of cities, places and food.
- In this module admin can add new city, place and food and also update or delete all the details of existing city, place or food whenever required.

5.2 Function prototypes

```
router.post('/signup', async (req, res) => {
    const { name, email, password, cpassword, gender, contactNumber } = req.body;
    if (!name || !email || !password || !cpassword || !gender || !contactNumber) {
        return res.status(422).json({ error: "All fields are required" });
    }
    try {
        const userExist = await User.findOne({ email: email });

        if (userExist) {
            return res.status(422).json({ error: "Email already in use" });

        } else if (password != cpassword) {
            return res.status(422).json({ error: "Passwords do not match" });
        }
        else {
            const user = new User({ name, email, password, cpassword, gender, contactNumber });
            await user.save();
            res.status(201).json({ message: "User Registered Successfully!" });
        }
    } catch (err) {
        console.log(err);
    }
});
```

User Signup

```
router.post('/signin', async (req, res) => {

    const { email, password } = req.body;

    if (!email || !password) {
        return res.status(400).json({ error: "All fields are required" });
    }
    else if (email == "admin@gmail.com" && password == "admin") {
        return res.status(500).json({ error: "Admin login" });
    }

    try {
        const user = await User.findOne({ email: email });

        const valid = await bcrypt.compare(password, user.password);

        if (user && valid) {
            return res.status(422).json({ error: "Signin Success" });
        }
        else if (us  (parameter) res: Response<any, Record<string, any>, number>
            return res.status(400).json({ error: "Incorrect Password" });
        }
        else if (valid) {
            return res.status(400).json({ error: "User Not Found" });
        }
    } catch (err) {
        console.log(err);
    }
```

Authentication module for admin and user

```
router.post('/addcity', async (req, res) => {
    const { image, name, details, location,locationCoordinates } = req.body;
    if (!image || !name || !details || !location || !locationCoordinates || image == " ") {
        return res.status(500).json({ error: "All fields are required" });
    }
    else {
        const city = new City({ image, name, details, location,locationCoordinates });
        city.save();
        res.status(422).json({ message: "City Information Added Successfully" });
    }
});
```

Admin module for add city

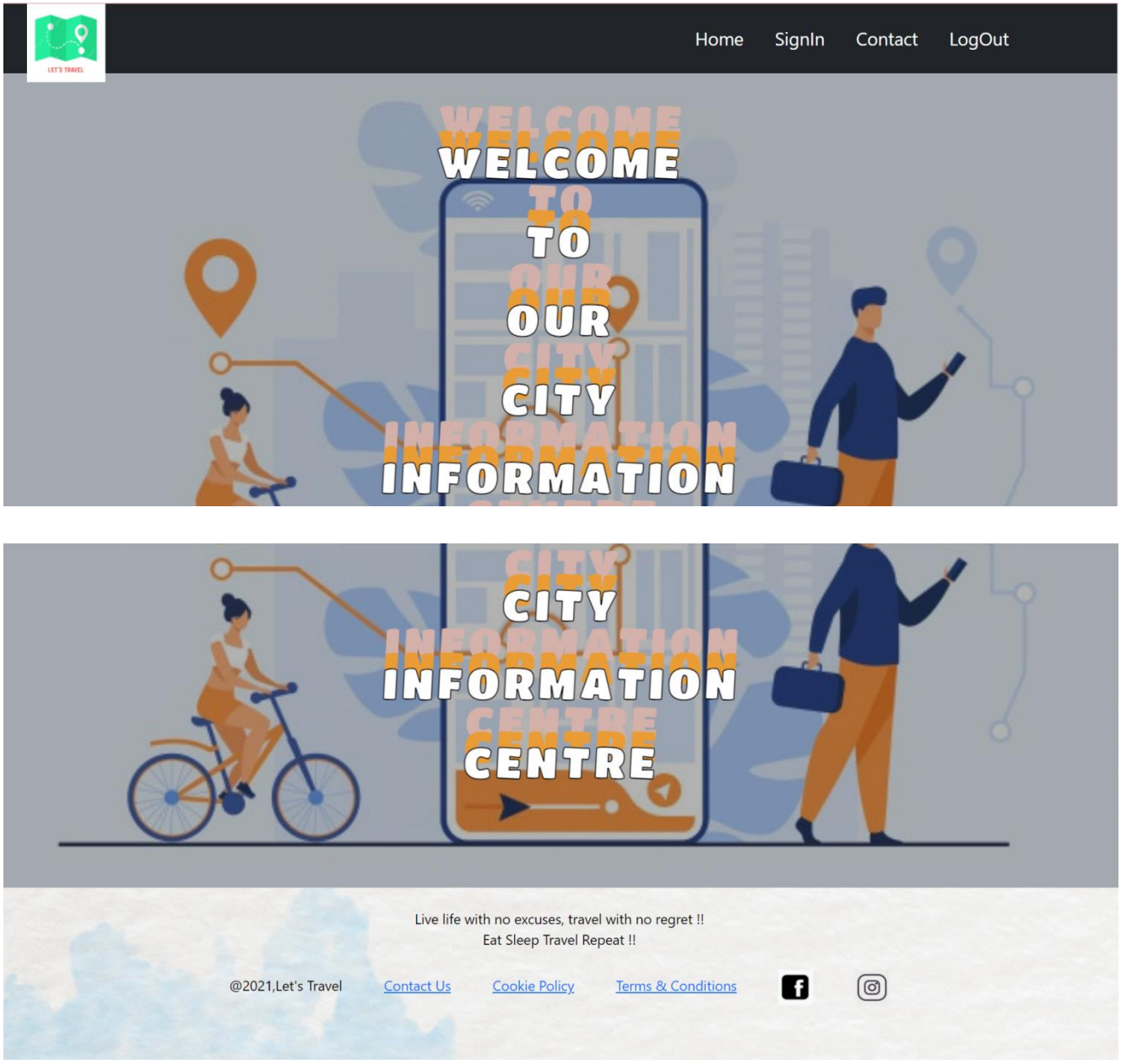
```
router.post('/editcity', async (req, res) => {
    const id = req.body.id;
    const image = req.body.image;
    const name = req.body.name;
    const details = req.body.details;
    const location = req.body.location;
    const locationCoordinates = req.body.locationCoordinates;
    City.updateOne({ _id: id },
        {
            $set: {
                "image": image,
                "name": name,
                "details": details,
                "location": location,
                "locationCoordinates":locationCoordinates
            }
        }).then(
            () => {
                res.status(422).json({ message: "Done" });
            }
        ).catch(
            (error) => {
                res.status(400).json({ error: error });
            }
        )
});
```

Admin module for update city

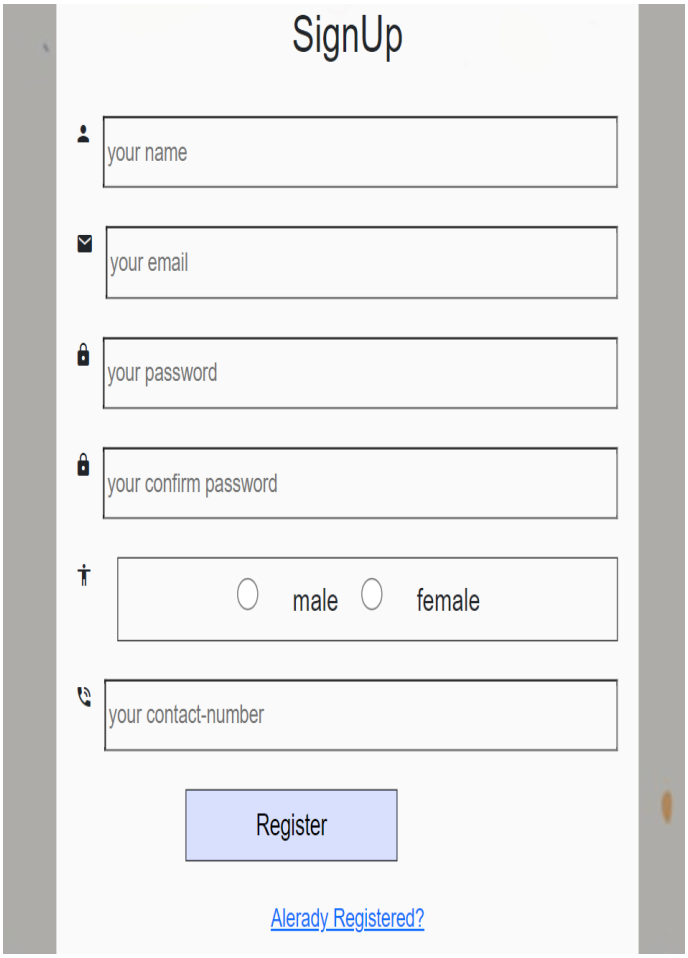
```
router.post('/deletecity', async (req, res) => {
    const { id } = req.body;
    City.findByIdAndDelete(id).
        then(
            () => {
                res.status(422).json({ message: "Deleted" });
            })
        .catch(
            (error) => {
                res.status(400).json({ error: error });
            }
        )
});
```

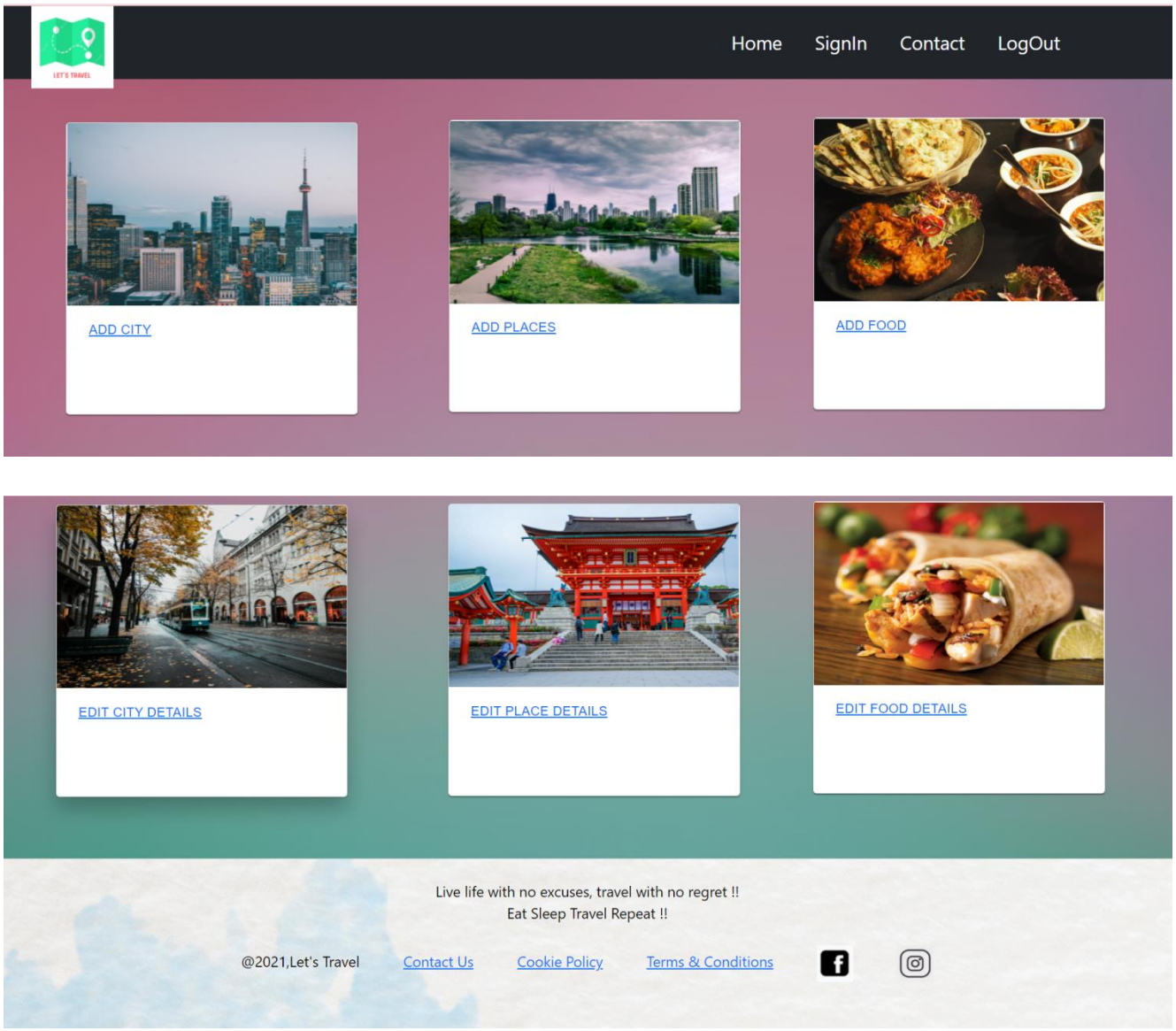
Admin module for Delete city

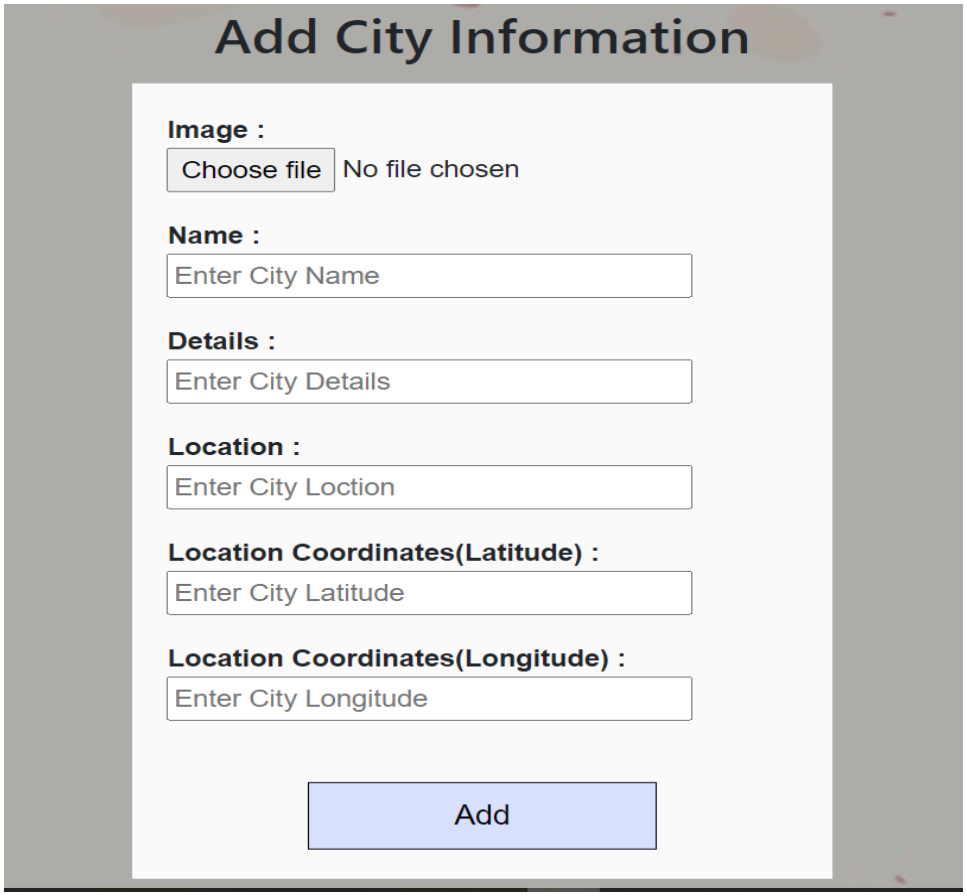
6. Screenshots

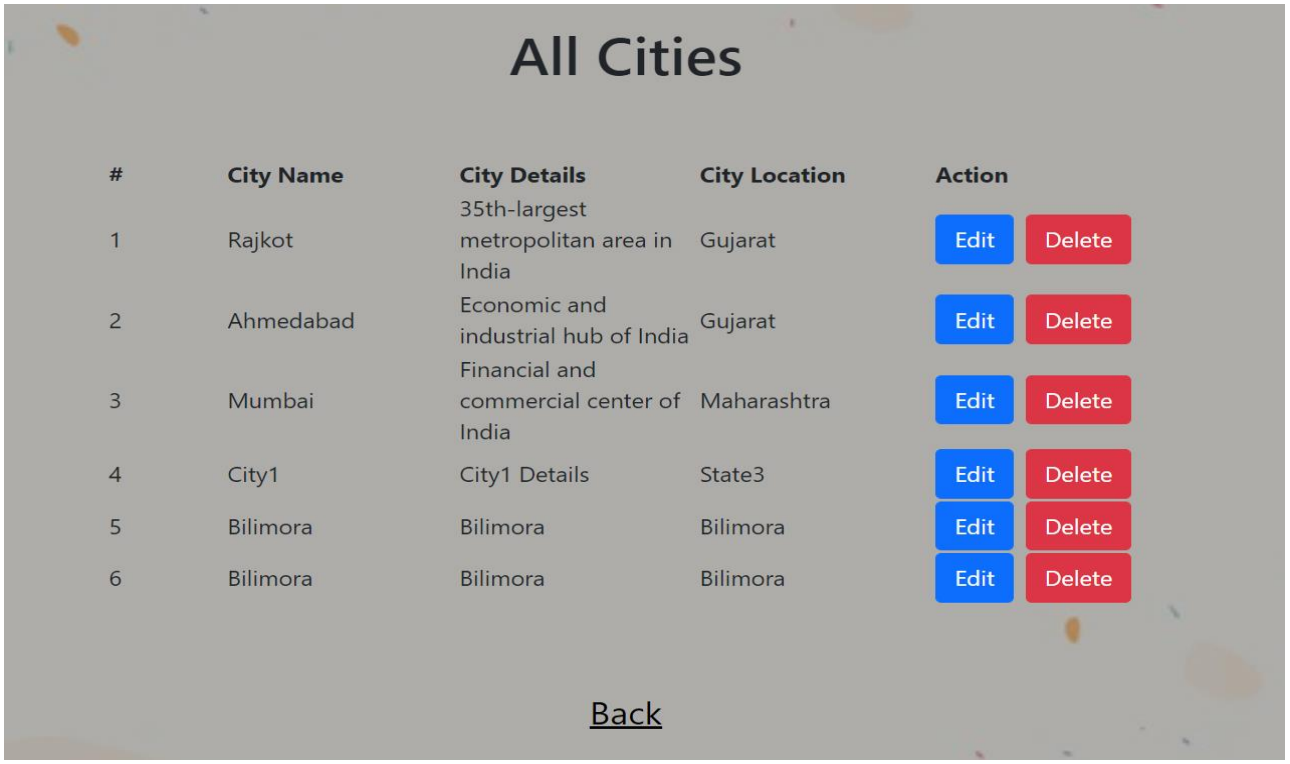


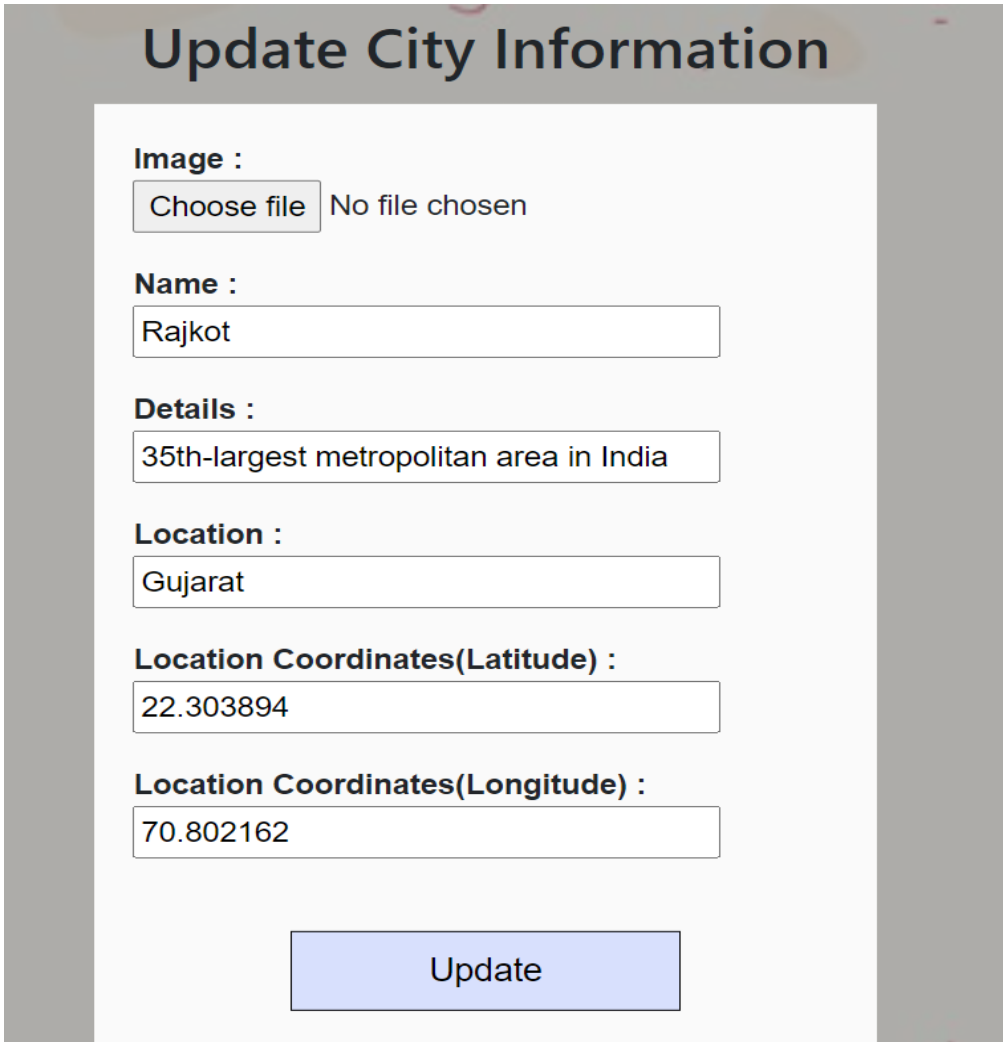

SignIn
Enter Your Email
Enter Your Password
Login
Sign up?

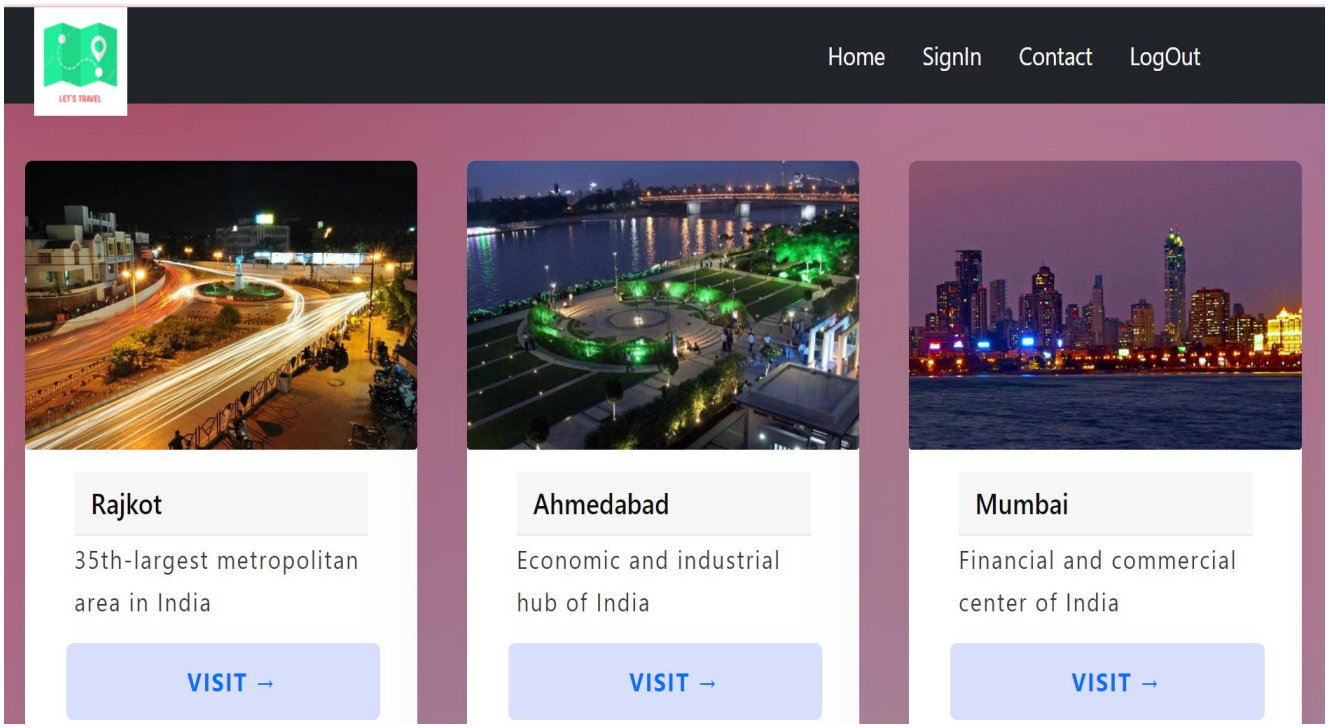

SignUp
your name
your email
your password
your confirm password
male
female
your contact-number
Register
Alerady Registered?


Home
SignIn
Contact
LogOut
ADD CITY
ADD PLACES
ADD FOOD
EDIT CITY DETAILS
EDIT PLACE DETAILS
EDIT FOOD DETAILS
Live life with no excuses, travel with no regret !!
Eat Sleep Travel Repeat !!
@2021,Let's Travel
Contact Us
Cookie Policy
Terms & Conditions

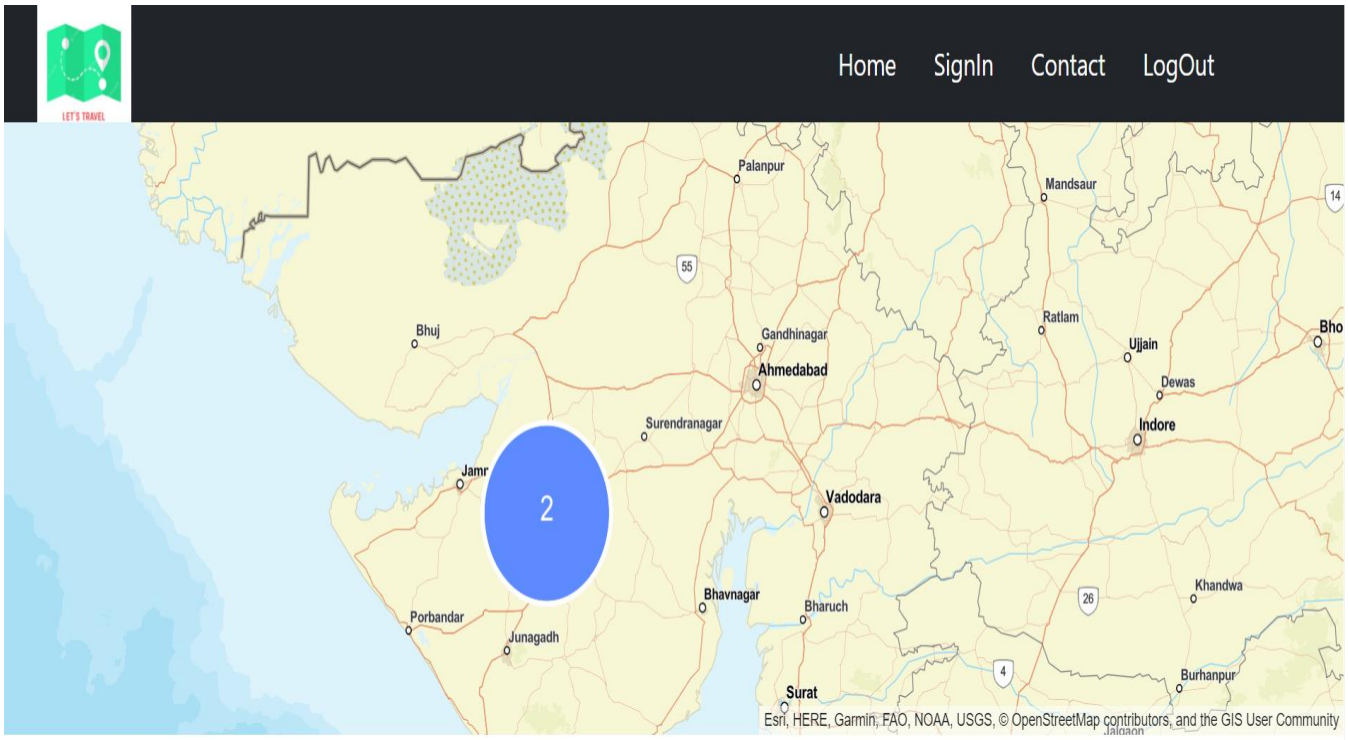

Add City Information
Image :
Choose file
No file chosen
Name :
Enter City Name
Details :
Enter City Details
Location :
Enter City Loction
Location Coordinates(Latitude) :
Enter City Latitude
Location Coordinates(Longitude) :
Enter City Longitude
Add


All Cities
City Name City Details City Location Action
1 Rajkot 35th-largest metropolitan area in India Gujarat Edit Delete
2 Ahmedabad Economic and industrial hub of India Gujarat Edit Delete
3 Mumbai Financial and commercial center of India Maharashtra Edit Delete
4 City1 City1 Details State3 Edit Delete
5 Bilimora Bilimora Bilimora Edit Delete
6 Bilimora Bilimora Bilimora Edit Delete
Back


Update City Information
Image :
Choose file
No file chosen
Name :
Rajkot
Details :
35th-largest metropolitan area in India
Location :
Gujarat
Location Coordinates(Latitude) :
22.303894
Location Coordinates(Longitude) :
70.802162
Update


LET'S TRAVEL
Home
SignIn
Contact
LogOut
Rajkot
35th-largest metropolitan area in India
VISIT →
Ahmedabad
Economic and industrial hub of India
VISIT →
Mumbai
Financial and commercial center of India
VISIT →


3
Rajkot
Airport Road
Raiya Road
Shree Guru Golwalkar Marg
Kuvadva Road
Pedak Road
Bhavnagar Road
Old Jail
Dudh Sagar Ro
121
23
23
25
Esri, HERE, Garmin, FAO, NOAA, USGS, © OpenStreetMap contributors, and the GIS User



Places To Visit

Places To Visit

Race Course

Place of activities and enjoyment

Location:
Race Course Ring Road, Race Course, Sadar, Rajkot, Gujarat 360001

FunWorld

Place full of energy and great fun activities

Location:
Ground, Race Course, Sadar, Rajkot, Gujarat 360001

Famous Foods

Channa

Yummm

7. Testing

Manual testing was performed in order to find and fix the bugs in development process.

Testing Method: Manual Testing

Sr. No.	Test Scenario	Expected Result	Actual Result	Status
1	Login with incorrect credentials	User should not able to log in.	User is given a message. And redirected to login page.	Success
2	Login with correct credentials	User should be able to log in.	User is logged in and redirected to home page.	Success
3	Validations on registration	User should not be allowed to enter incorrect email, password, re-enter password.	User is shown a message for any incorrect input data.	Success
4	Registering with an existing username.	The system should show a message.	The system shows the message 'User already registered.'	Success

5	Add city without filling details	Admin should not able to add city in database.	The system shows the message ‘Invalid data/input’.	Success

6	Add Place without filling details	Admin should not able to add place in database.	The system shows the message ‘Invalid data/input’.	Success
7	Add Food without filling details	Admin should not able to add food in database.	The system shows the message ‘Invalid data/input’ .	Success
8	Remove any city, place, food	Admin should be able to remove the details from the database.	Message will be shown as ‘Successfully Deleted’	Success
9	Log Out	User should be logged out	User is successfully logged out and not able to see the city details.	Success

8. Conclusion

Hence-forth in this project we have successfully implemented the Admin-side & User-side functionality.

Admin can add city, place and food details. Also admin can edit and delete city, place and food details.

The user can view details of the city and its places and food as per their choice. They can even check the locations of the city and its places on map.

9. Limitations and Future Enhancements

Limitations:

1.) There is no provision for user to change his/her email or password.

2.) Email id entered by user cannot be verified.

Future Enhancements:

1.) We can add hotels section which will display some if not all hotels of a city.

10. Reference / Bibliography

Stackoverflow: https://stackoverflow.com/

React official documentation: https://reactjs.org/docs/getting-started.html

W3Schools: https://www.w3schools.com/

GeekForGeeks: https://www.geeksforgeeks.org/