PROJECT REPORT

WEATHER APP

SUBMITTED BY,

NAFLA SHAJAHAN

FOUSIYA. C. I

BLESSON M ALEX

CONTENTS

CHAPTER I

INTRODUCTION AND OBJECTIVE

1.1. Introduction:

In some or other way our daily lives are dependant on the weather conditions. It has been always essential to know the regular updates of weather, as it continuously varies with every passing day. Certain changes in weather are always wonderful and cherishable. No one can imagine how the weather is going to be on a subsequent day. Definitely, the Weather forecast is a big thing that enabled many of us to stay notified about the changes in climatic conditions beforehand. It can be said that it is one of the greatest advancements of all time, mothered by innovative technologies and creative thoughts. The furtherance of the weather forecasting is the weather app development.

Having an up to date information about the weather helps us to take well-read decisions. This app constantly update the forecasts for a day or hour or sometimes for even a minute.

1.2. Objective:

The objective of this project is to develop a web based application through which the user will be able to get all the necessary weather forecasting details of any desired locations. The details include

- weather details of a particular city
- air quality details of all the cities of a state and
- allows the user to mark his favourite details.

CHAPTER II
DESCRIPTION OF EXISTING SYSTEM

Weather Report project application is a web based application through which you will able to get all the reports related to weather forecasting of any locations.

Previously built Weather Report project web based application was compatible with system and every time users start this application, they have to set their default location to get weather reports on it. Due to complex coding, system responding time was high and require more memory to get start up. The concept of graphics for geographical region was not implemented in older version. Dynamic concept was not implemented under the existing system, thus theme and colour of web page was not changing.

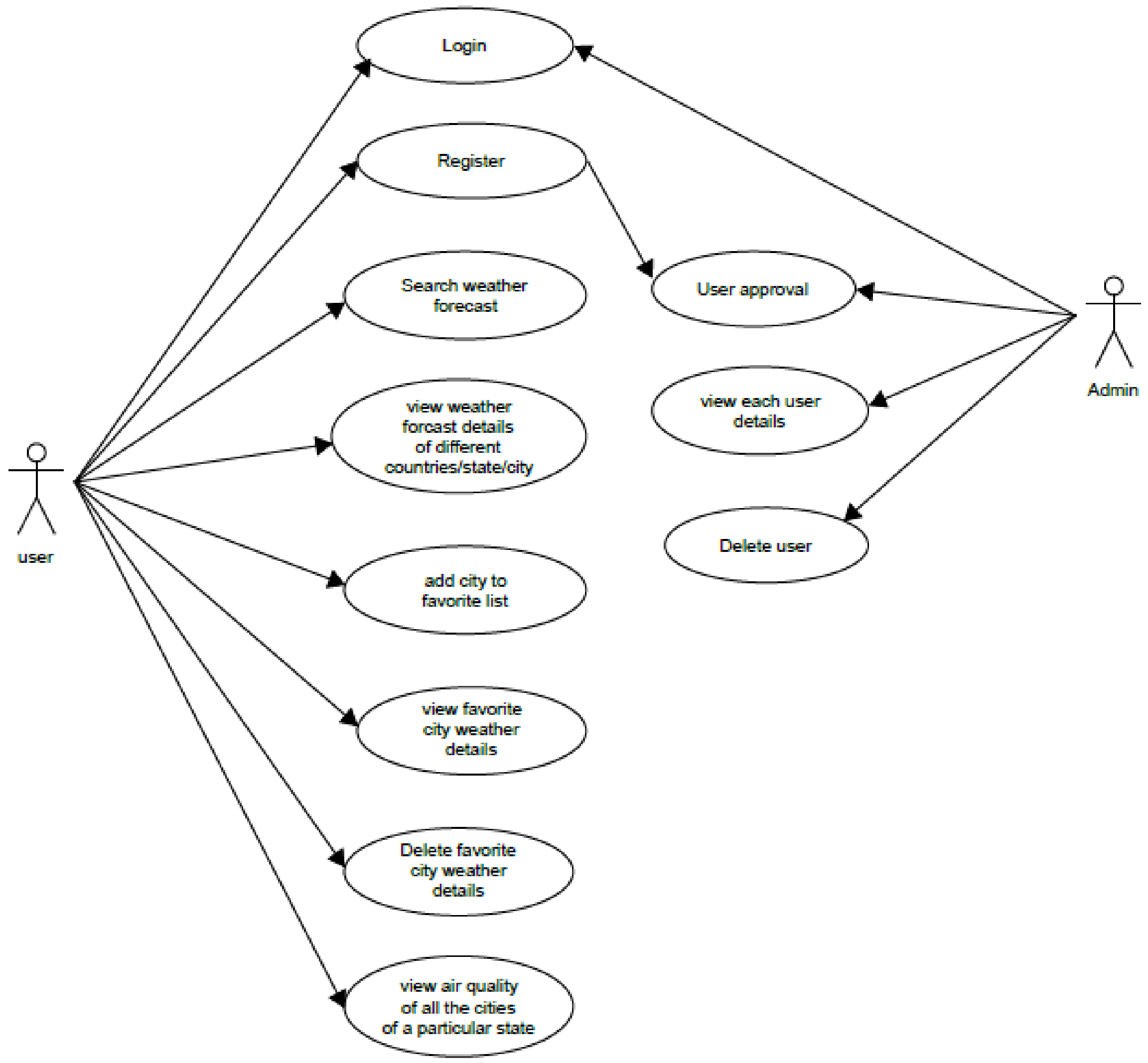
Most weather apps on computers and smartphones can only predict the weather 10 days into the future, which is still pretty impressive compared to what meteorologists could do in the past. Today, a forecast for the next five to seven days is just as accurate as a forecast for the next day was 50 years ago.

CHAPTER III

OBJECT-ORIENTED ANALYSIS AND DESIGN

3.1. Use Case Diagram

A use case is a sequence of action that provides a measurable value to an actor another way to look at it is that a use case describes a way to which a real world to interacts with the system. An essential use case sometimes called a business the case is simplified, abstract, generalized use case that captures the intention of the user in a technology and implementation independent manner.


Login
Register
Search weather forecast
view weather forcast details of different countries/state/city
add city to favorite list
view favorite city weather details
Delete favorite city weather details
view air quality of all the cities of a particular state
User approval
view each user details
Delete user
user
Admin

Relationship between actors and use cases exists whenever an actor is involved with an interaction described by a use case and modelled as a line connecting use cases and actors.

3.1.1 Actor description

Administrator: performs activities login, view and delete all users, and also can view the weather forecast details, air watcher details.

User: performs login and add favourite cities, search updated weather details and air watcher details.

3.1.2 Use Case Textual Description

Use case name: Login use case

Identification: UC01

Description: Use case to ensure security in system usage

Actor: User (Administrator, All internet users)

Precondition: The user must have username and password.

Post condition: User get access to the system according to their predefined system privilege and finally he/she logout or turn off the page.

Basic course of action:

1. User activates the system.

2. System response by displaying the login interfaces and prompts the user for the user ID and password.

3. User fills his or her user ID and password and presses enter.

4. System verifies user ID and Password.

5. User authenticated and gets access to the system.

6. System displays its main window.

7. Use case ends.

Alternative course of action (if user enters wrong user ID and / or password)

A1.User is not authenticated and is denied access to the system.

A2.System displays an incorrect user ID and password message.

A3.System enables user to try again.

A4.Use case returns to step 2 of main use case.

Use case name: Apply to register

Identification: UCO2

Description: Use case to register new applicant.

Actor: User (All internet user)

Precondition: The applicant must be able to fill all the criteria perfectly and the applicant has the ability to access the internet.

Post condition: An applicant or new customer is registered.

Basic course of action :

1. Not include login use case.

2. Applicant selects applicant’s link.

3. Applicant select form.

4. System display applicant form.

5. Applicant fills necessary data.

6. Submit.

7. Use case ends.

Alternative course of action (if applicant enters wrong information)

A1. Applicant is not authenticated and is denied access to the system.

A2. System displays an incorrect message.

A3. Use case returns to step 5 of main use case.

Use case name: Search weather forecast details

Identifier: UC03

Description: Use case to retrieve a weather forecast details.

Actor: User (Administrator, all internet user)

Precondition: The user must be a registered member.

Postcondition: System display information about the weather details .

Basic course of action:

1. Include login use case.

2. User activates the weather interface.

3. User selects country, state, city from the displayed list.

4. System consults the database and displays the collection matching the search data.

5. Use case ends.

Alternative course of action (user entered a search data that doesn't exist in the system)

A1. System responds stating there is no resource matching the search in its result display.

A2.use case returned to step 3.

Use case name: Add favourite service

Description: UC04

Actor: User (Administrator , All internet user)

Precondition: User wants to add a favourite cities to the data base.

Post condition: system adds the a favourite cities to the data base.

Basic course of action:

1. Include login use case.

2. User selects record link.

3. User clicks the add button from displayed record link.

4. User   over views the add entry.

5. User clicks the add button.

6. End use case.

Use case name: Search air watcher details

Identifier: UC05

Description: Use case to retrieve air watcher details.

Actor: User (Administrator , All internet user)

Precondition: The user must be a member.

Postcondition: System display information about the air watcher details .

Basic course of action:

1. Include login use case.

2. User selects country, state from the displayed list.

3. System consults the database and displays the air quality details of particular state of matching the search data.

4. Use case ends.

Alternative course of action (user entered a search data that doesn’t exist in the system)

A1. System responds stating there is no resource matching the search in its result display.

A2.use case returned to step 2.

Use case name: Search user details

Identifier: UC06

Description: Use case to retrieve user details.

Actor: Administrator

Precondition: Administrator must be a member.

Postcondition: System displays information about the user details .

Basic course of action:

1. Include login use case.

2. User selects particular user from the displayed list.

3. System consults the database and displays the user details of matching the search data.

4. Use case ends.

Alternative course of action (user entered a search data that doesn’t exist in the system)

A1. System responds stating there is no resource matching the search in its result display.

A2.use case returned to step 2.

Use case name: Delete user details

Identifier: UC07

Description: Use case to delete user details.

Actor: Administrator

Precondition: Administrator must be a member.

Postcondition: System displays information about the user details.

Basic course of action:

1. Include login use case.

2. User selects particular user from the displayed list and click delete button.

3. System consults the database and displays the user details .

4. Use case ends.

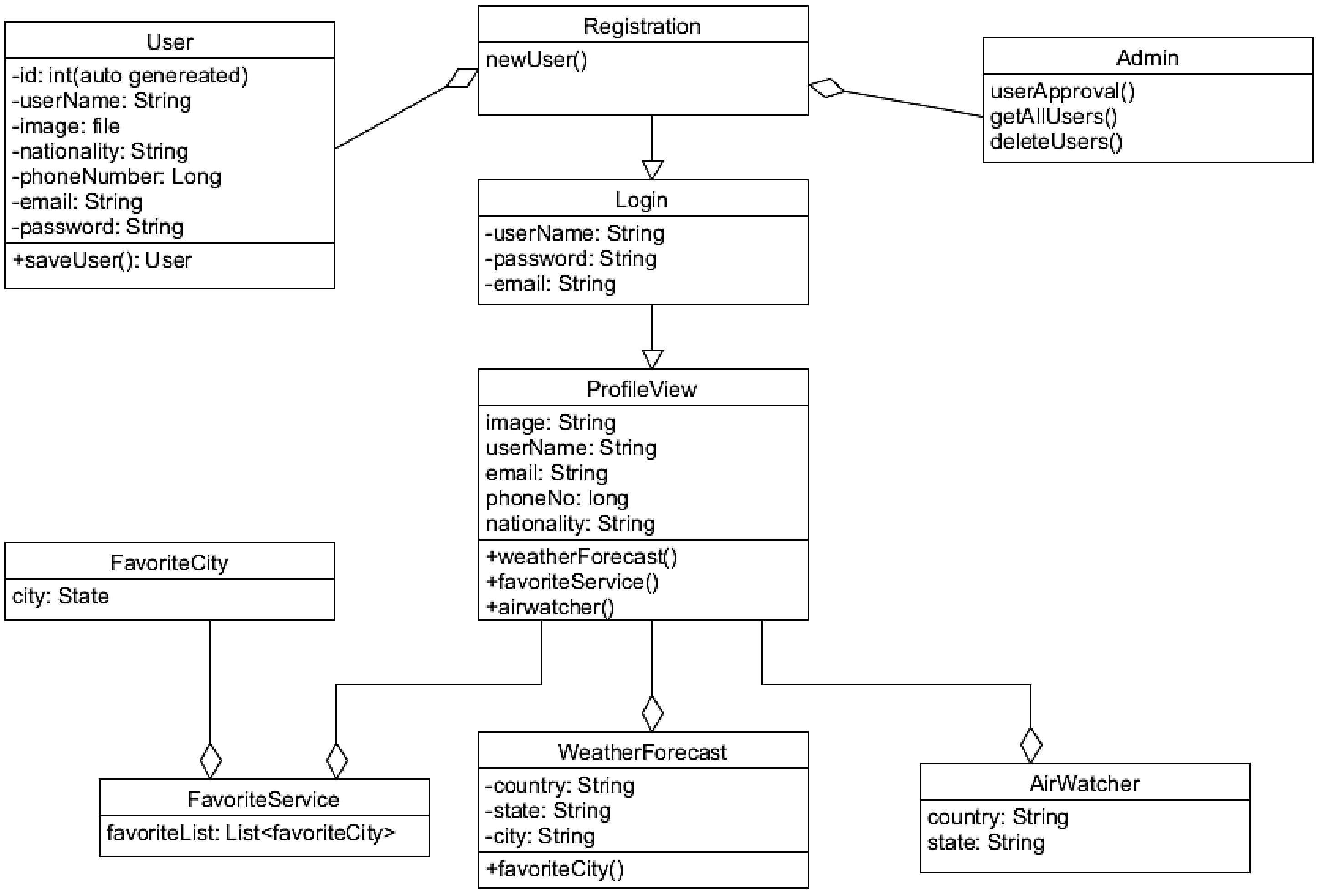
Alternative course of action (user entered a search data that doesn’t exist in the system)

A1. System responds stating there is no resource matching the search in its result display.

A2.use case returned to step 2.

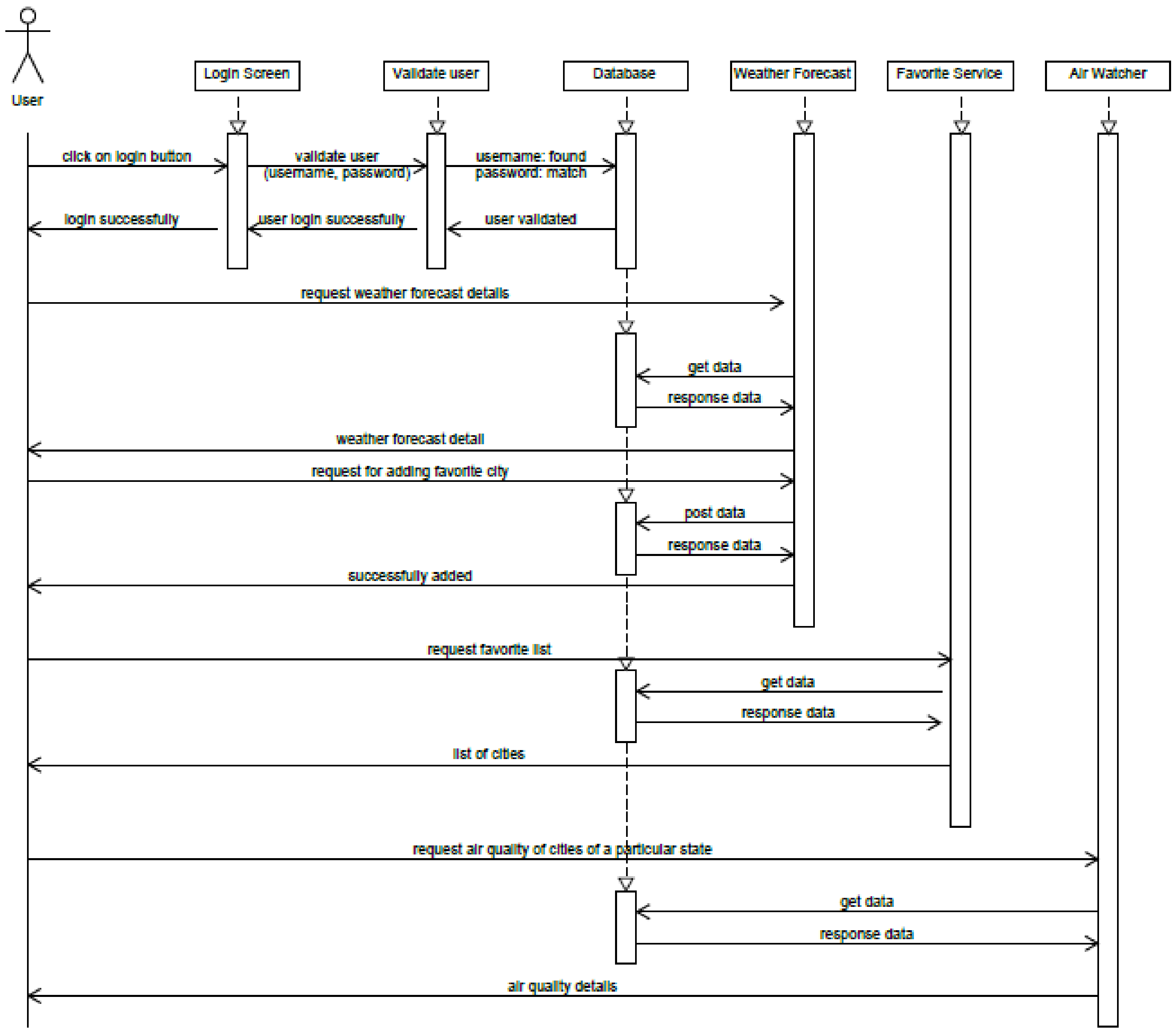
3.2. Class Diagram

Class diagrams are used to represent the structure of the system in terms of objects, their notes and nature of relationship between classes. It shows the static features of the objects and do not represent any particular processing.



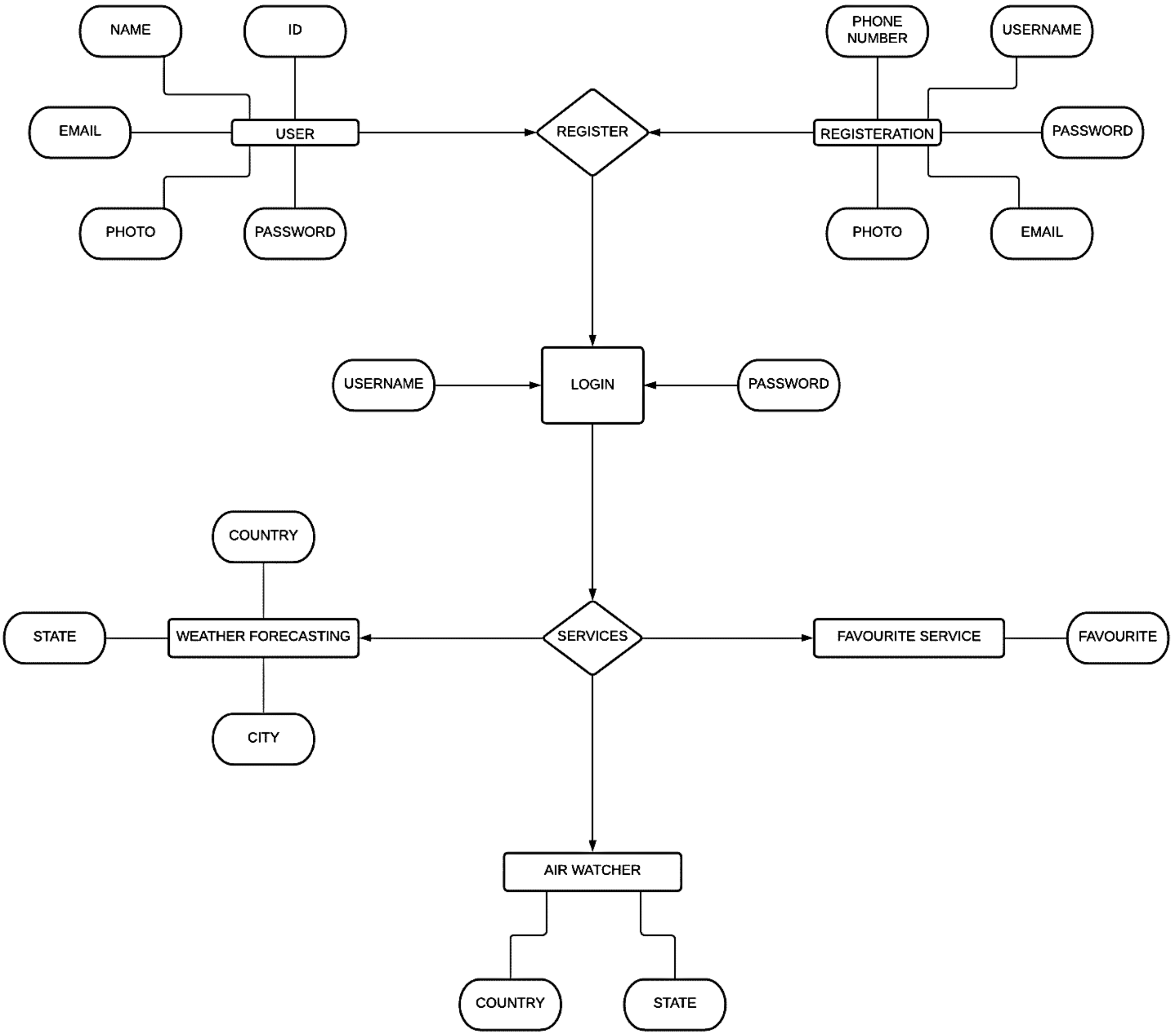
3.3. Sequence Diagram

A sequence diagram is a Unified Modelling Language (UML) diagram that illustrates the sequence of messages between objects in an interaction. A sequence diagram consists of a group of objects that are represented by lifelines, and the messages that they exchange over time during the interaction. It shows the sequence of messages passed between objects and the control structures between objects.



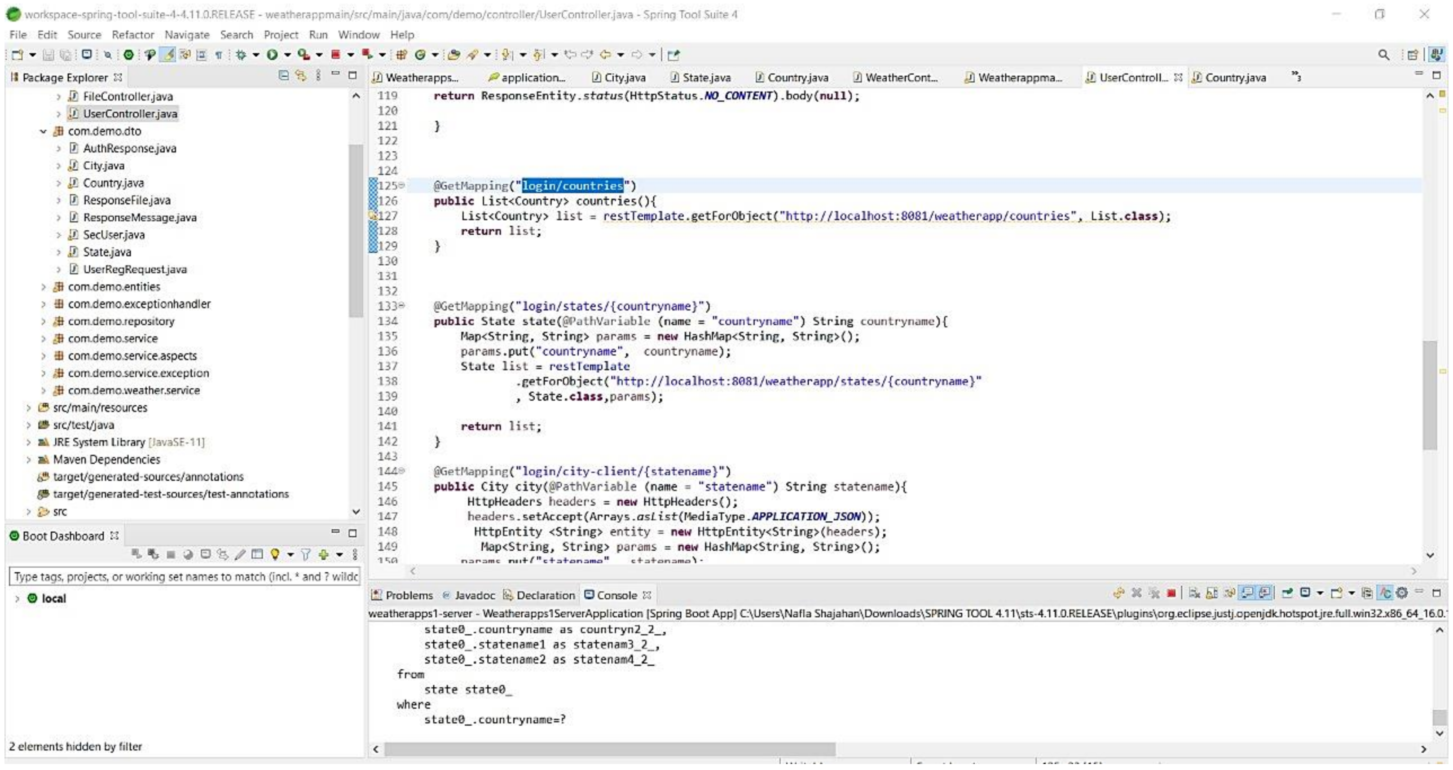
3.4. ER Diagram

Entity Relationship Diagram (ERD) is a diagram that displays the relationship of entity sets stored in a database. In other words, ER diagrams help to explain the logical structure of databases. ER diagrams are created based on three basic concepts: entities, attributes and relationships. ER Diagrams contain different symbols that use rectangles to represent entities, ovals to define attributes and diamond shapes to represent relationships.



CHAPTER IV

WORKING OF THE APPLICATION (IMAGES)

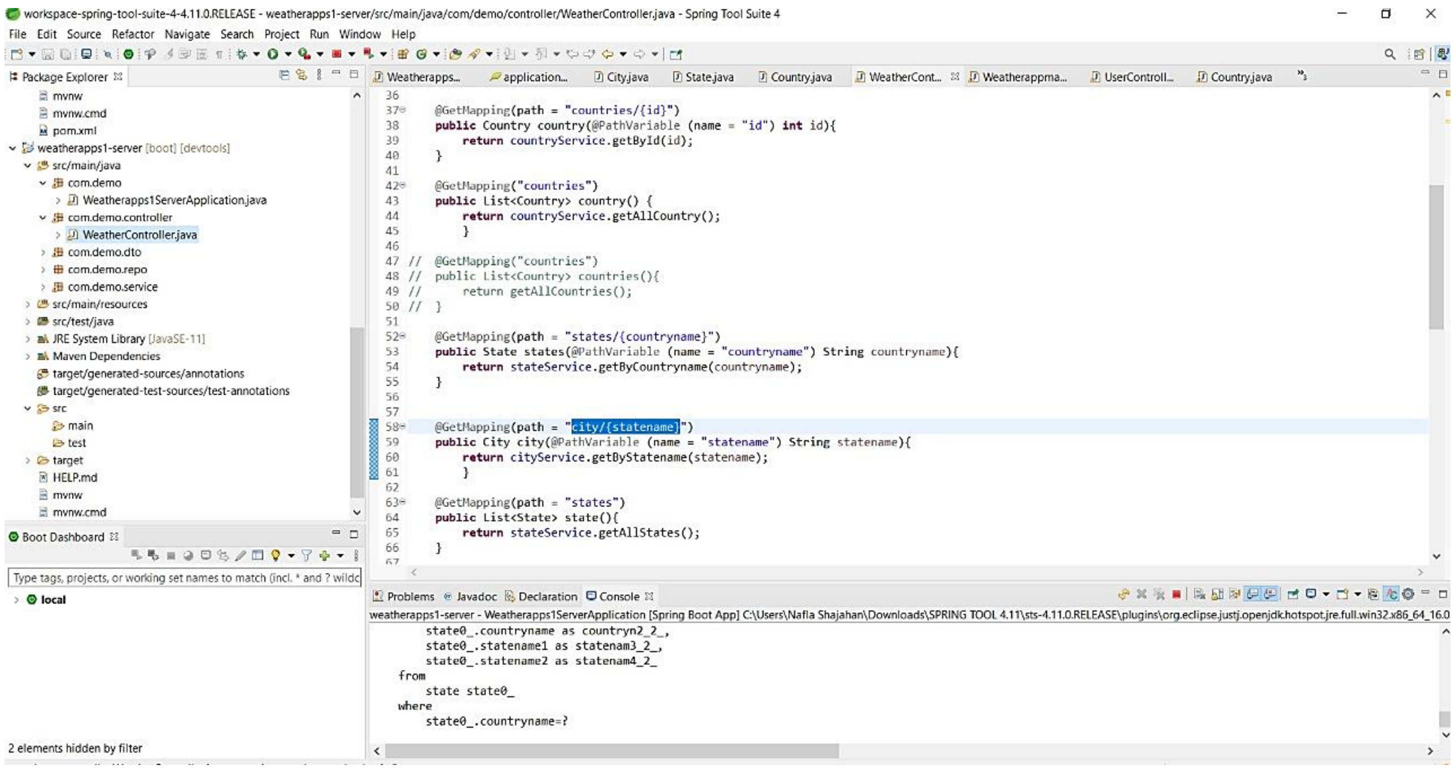


workspace-spring-tool-suite-4-4.11.0.RELEASE - weatherappmain/src/main/java/com/demo/controller/UserController.java - Spring Tool Suite 4

```java
        return ResponseEntity.status(HttpStatus.NO_CONTENT).body(null);

    }



    @GetMapping("login/countries")
    public List<Country> countries(){
        List<Country> list = restTemplate.getForObject("http://localhost:8081/weatherapp/countries", List.class);
        return list;
    }



    @GetMapping("login/states/{countryname}")
    public State state(@PathVariable (name = "countryname") String countryname){
        Map<String, String> params = new HashMap<String, String>();
        params.put("countryname",  countryname);
        State list = restTemplate
                .getForObject("http://localhost:8081/weatherapp/states/{countryname}"
                , State.class,params);

        return list;
    }

    @GetMapping("login/city-client/{statename}")
    public City city(@PathVariable (name = "statename") String statename){
         HttpHeaders headers = new HttpHeaders();
         headers.setAccept(Arrays.asList(MediaType.APPLICATION_JSON));
          HttpEntity <String> entity = new HttpEntity<String>(headers);
           Map<String, String> params = new HashMap<String, String>();
```

weatherapps1-server - Weatherapps1ServerApplication [Spring Boot App] C:\Users\Nafla Shajahan\Downloads\SPRING TOOL 4.11\sts-4.11.0.RELEASE\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_16.0.

```
        state0_.countryname as countryn2_2_,
        state0_.statename1 as statenam3_2_,
        state0_.statename2 as statenam4_2_
    from
        state state0_
    where
        state0_.countryname=?
```

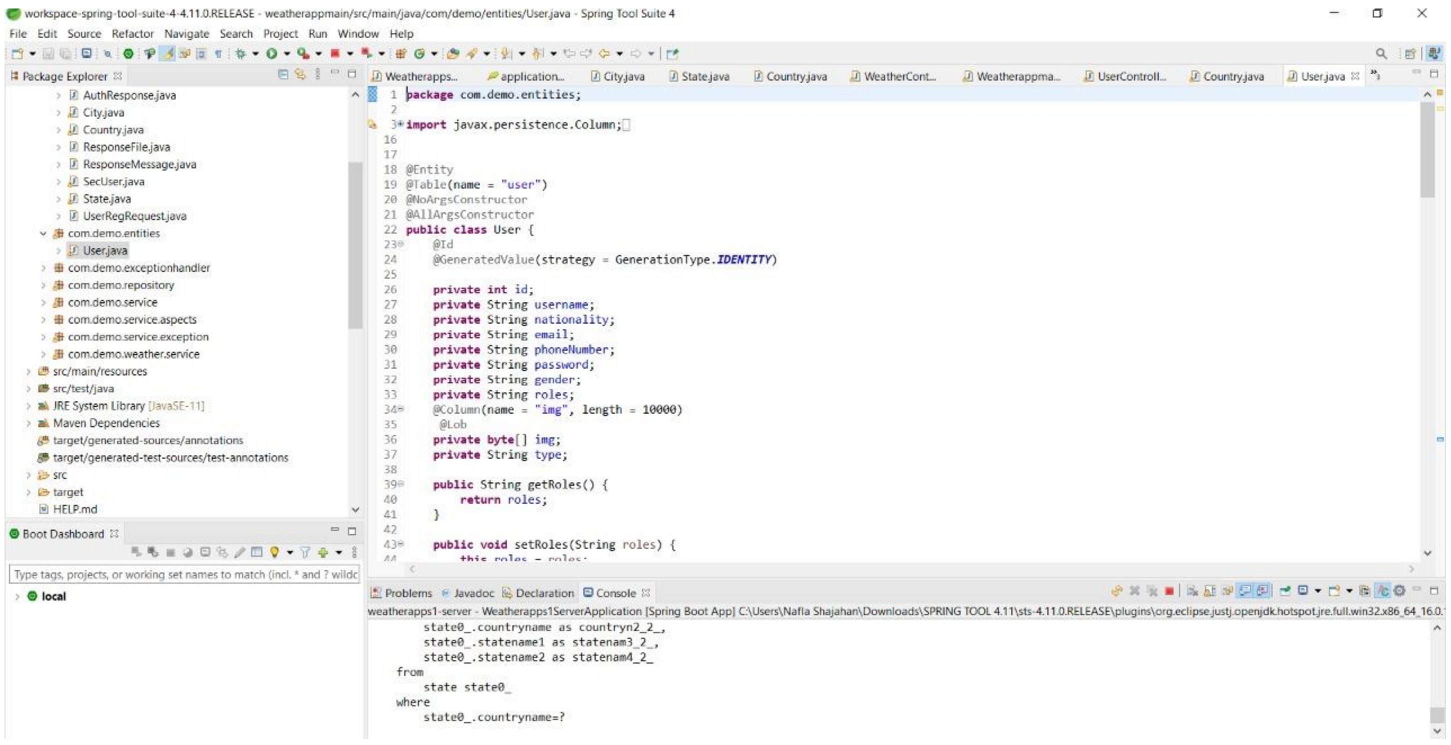


workspace-spring-tool-suite-4-4.11.0.RELEASE - weatherapps1-server/src/main/java/com/demo/controller/WeatherController.java - Spring Tool Suite 4

```java

    @GetMapping(path = "countries/{id}")
    public Country country(@PathVariable (name = "id") int id){
        return countryService.getById(id);
    }

    @GetMapping("countries")
    public List<Country> country() {
        return countryService.getAllCountry();
        }

//  @GetMapping("countries")
//  public List<Country> countries(){
//      return getAllCountries();
//  }

    @GetMapping(path = "states/{countryname}")
    public State states(@PathVariable (name = "countryname") String countryname){
        return stateService.getByCountryname(countryname);
    }


    @GetMapping(path = "city/{statename}")
    public City city(@PathVariable (name = "statename") String statename){
        return cityService.getByStatename(statename);
        }

    @GetMapping(path = "states")
    public List<State> state(){
        return stateService.getAllStates();
    }
```

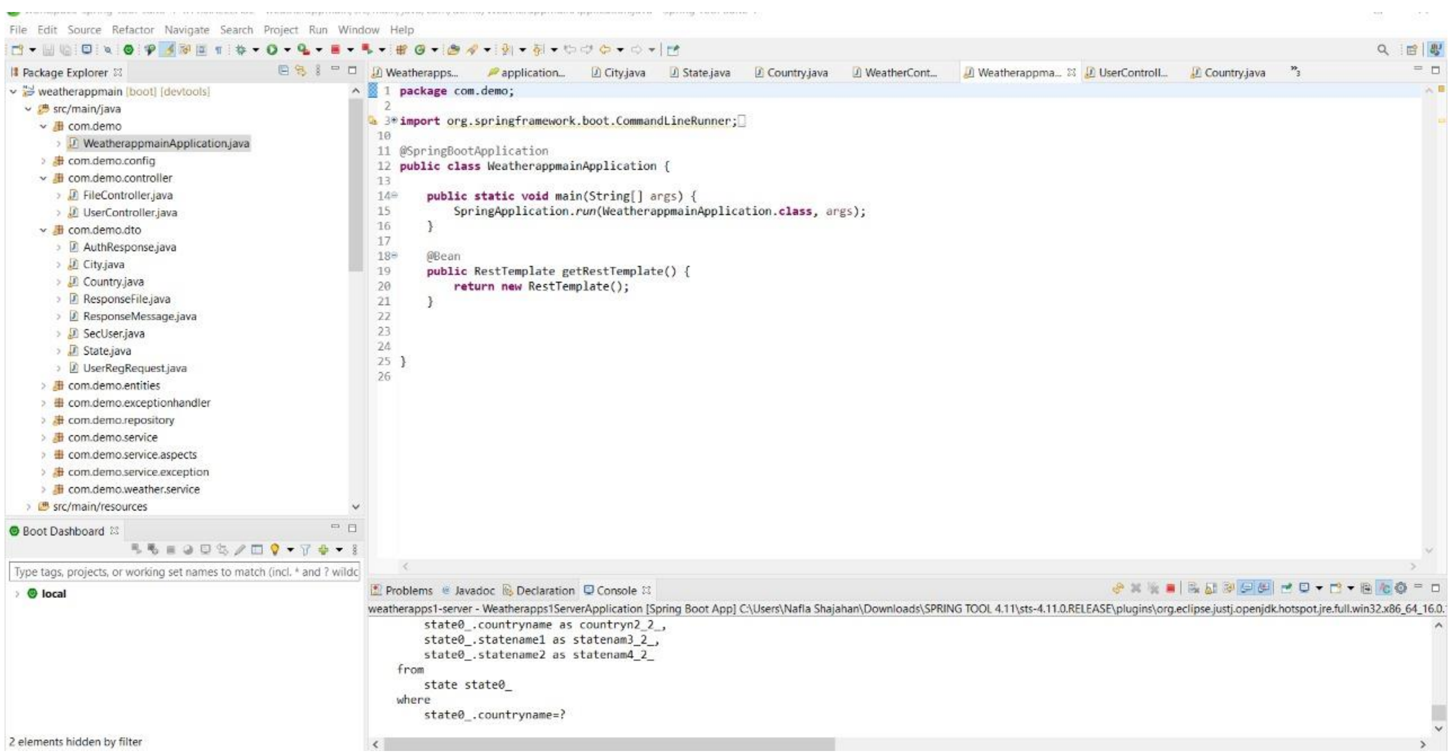
weatherapps1-server - Weatherapps1ServerApplication [Spring Boot App] C:\Users\Nafla Shajahan\Downloads\SPRING TOOL 4.11\sts-4.11.0.RELEASE\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_16.0

```
        state0_.countryname as countryn2_2_,
        state0_.statename1 as statenam3_2_,
        state0_.statename2 as statenam4_2_
    from
        state state0_
    where
        state0_.countryname=?
```


workspace-spring-tool-suite-4-4.11.0.RELEASE - weatherappmain/src/main/java/com/demo/entities/User.java - Spring Tool Suite 4
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
AuthResponse.java
City.java
Country.java
ResponseFile.java
ResponseMessage.java
SecUser.java
State.java
UserRegRequest.java
com.demo.entities
User.java
com.demo.exceptionhandler
com.demo.repository
com.demo.service
com.demo.service.aspects
com.demo.service.exception
com.demo.weather.service
src/main/resources
src/test/java
JRE System Library [JavaSE-11]
Maven Dependencies
target/generated-sources/annotations
target/generated-test-sources/test-annotations
src
target
HELP.md
Boot Dashboard
Type tags, projects, or working set names to match (incl. * and ? wildc
local
Weatherapps... application... City.java State.java Country.java WeatherCont... Weatherappma... UserControll... Country.java User.java
Problems Javadoc Declaration Console

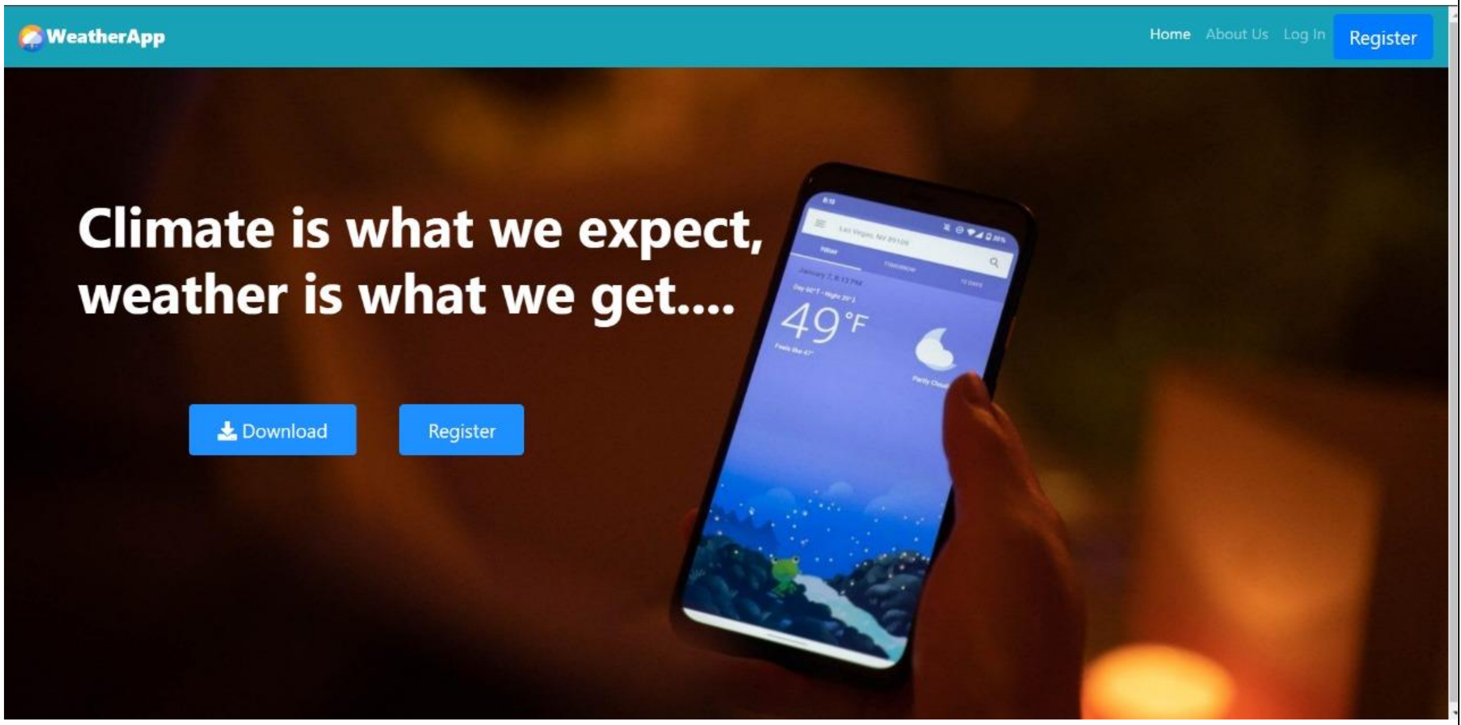
```java
package com.demo.entities;

import javax.persistence.Column;


@Entity
@Table(name = "user")
@NoArgsConstructor
@AllArgsConstructor
public class User {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)

    private int id;
    private String username;
    private String nationality;
    private String email;
    private String phoneNumber;
    private String password;
    private String gender;
    private String roles;
    @Column(name = "img", length = 10000)
     @Lob
    private byte[] img;
    private String type;

    public String getRoles() {
        return roles;
    }

    public void setRoles(String roles) {
        this.roles = roles;
```

```
weatherapps1-server - Weatherapps1ServerApplication [Spring Boot App] C:\Users\Nafla Shajahan\Downloads\SPRING TOOL 4.11\sts-4.11.0.RELEASE\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_16.0.
        state0_.countryname as countryn2_2_,
        state0_.statename1 as statenam3_2_,
        state0_.statename2 as statenam4_2_
    from
        state state0_
    where
        state0_.countryname=?
```

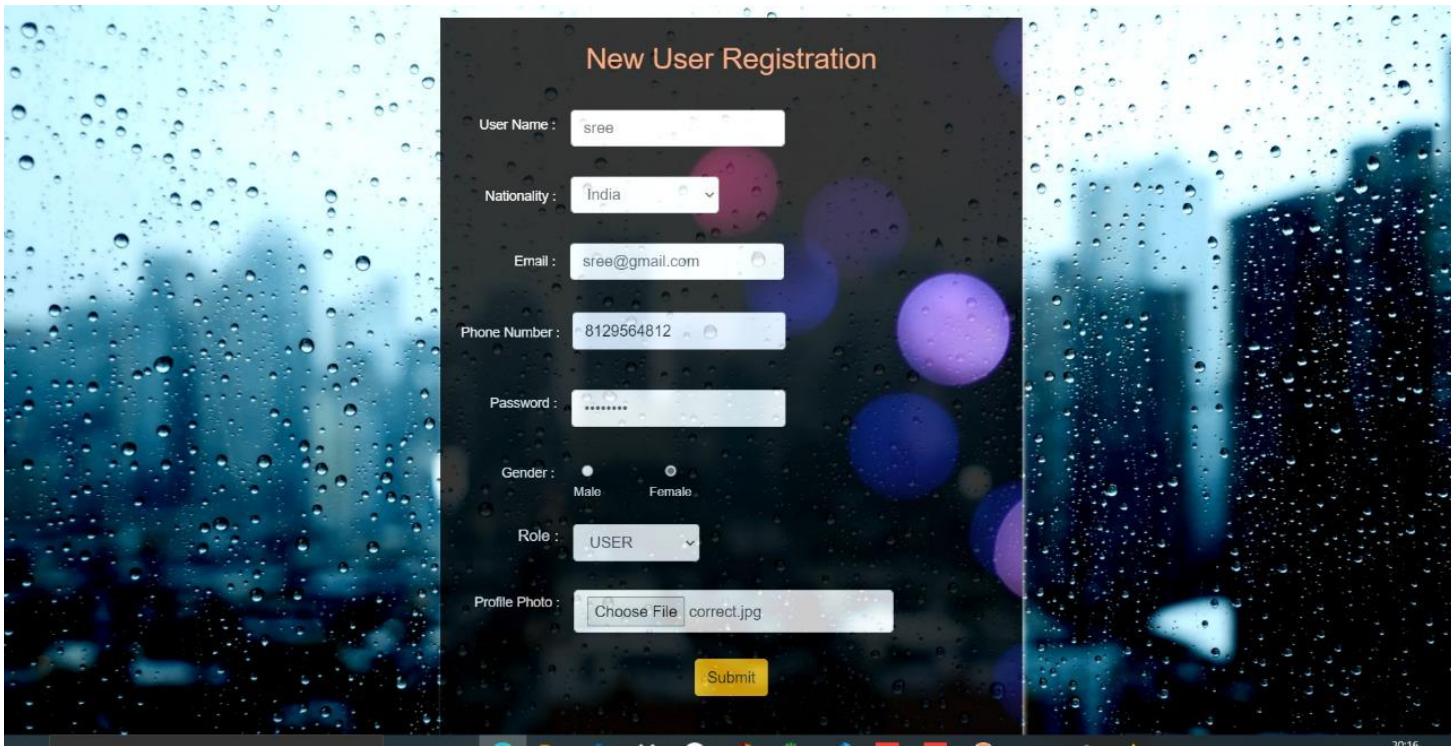

File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
weatherappmain [boot] [devtools]
src/main/java
com.demo
WeatherappmainApplication.java
com.demo.config
com.demo.controller
FileController.java
UserController.java
com.demo.dto
AuthResponse.java
City.java
Country.java
ResponseFile.java
ResponseMessage.java
SecUser.java
State.java
UserRegRequest.java
com.demo.entities
com.demo.exceptionhandler
com.demo.repository
com.demo.service
com.demo.service.aspects
com.demo.service.exception
com.demo.weather.service
src/main/resources
Boot Dashboard
Type tags, projects, or working set names to match (incl. * and ? wildc
local
2 elements hidden by filter
Weatherapps... application... City.java State.java Country.java WeatherCont... Weatherappma... UserControll... Country.java
Problems Javadoc Declaration Console

```java
package com.demo;

import org.springframework.boot.CommandLineRunner;

@SpringBootApplication
public class WeatherappmainApplication {

    public static void main(String[] args) {
        SpringApplication.run(WeatherappmainApplication.class, args);
    }

    @Bean
    public RestTemplate getRestTemplate() {
        return new RestTemplate();
    }




}
```

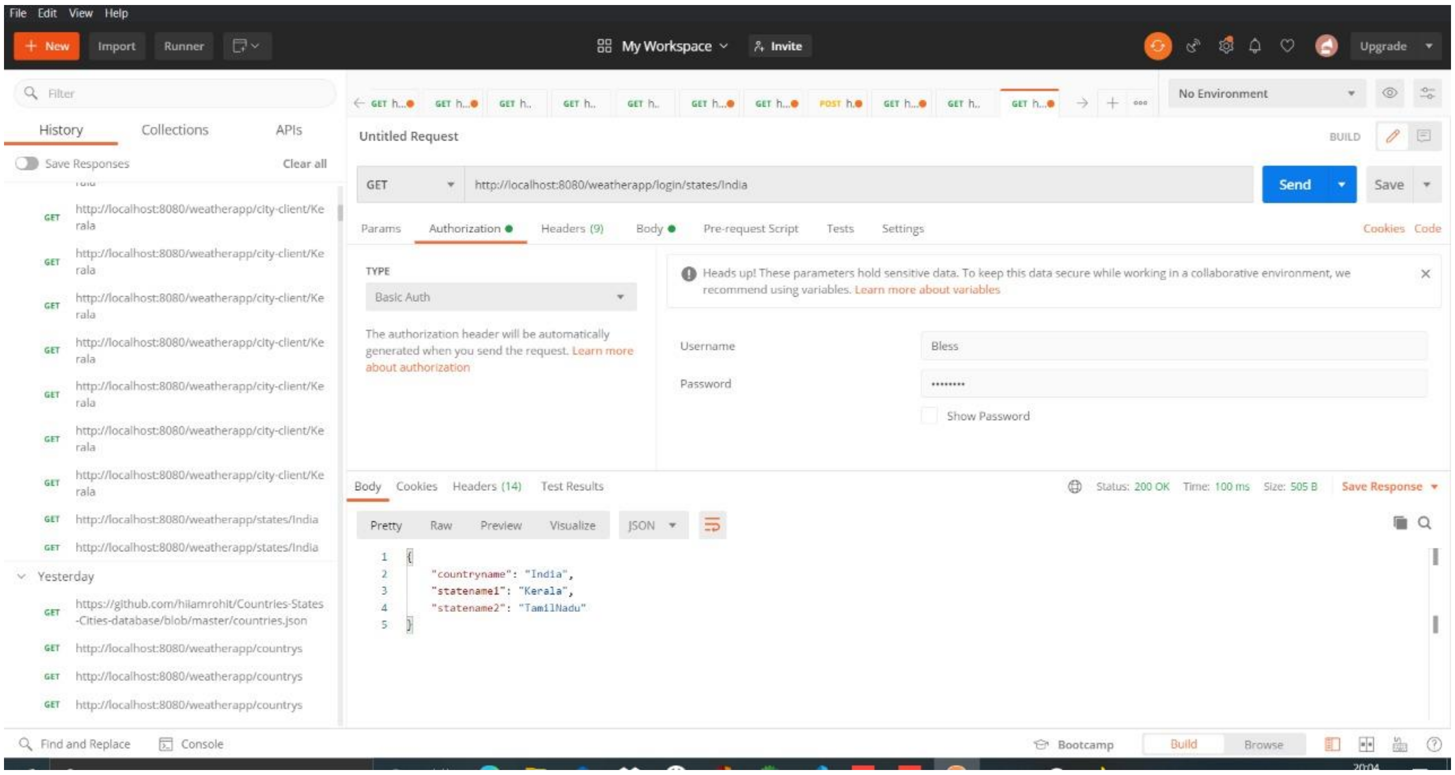
```
weatherapps1-server - Weatherapps1ServerApplication [Spring Boot App] C:\Users\Nafla Shajahan\Downloads\SPRING TOOL 4.11\sts-4.11.0.RELEASE\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_16.0.
        state0_.countryname as countryn2_2_,
        state0_.statename1 as statenam3_2_,
        state0_.statename2 as statenam4_2_
    from
        state state0_
    where
        state0_.countryname=?
```



Climate is what we expect, weather is what we get....


New User Registration

User Name :	sree
Nationality :	India
Email :	sree@gmail.com
Phone Number :	8129564812
Password :	••••••••
Gender :	○ Male ○ Female
Role :	USER
Profile Photo :	Choose File correct.jpg



GET http://localhost:8080/weatherapp/login/states/India

TYPE: Basic Auth

The authorization header will be automatically generated when you send the request. Learn more about authorization

Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. Learn more about variables

Username: Bless

Password: ••••••••

Show Password

History:

GET http://localhost:8080/weatherapp/city-client/Kerala

GET http://localhost:8080/weatherapp/city-client/Kerala

GET http://localhost:8080/weatherapp/city-client/Kerala

GET http://localhost:8080/weatherapp/city-client/Kerala

GET http://localhost:8080/weatherapp/city-client/Kerala

GET http://localhost:8080/weatherapp/city-client/Kerala

GET http://localhost:8080/weatherapp/city-client/Kerala

GET http://localhost:8080/weatherapp/states/India

GET http://localhost:8080/weatherapp/states/India

Yesterday

GET https://github.com/hilamrohit/Countries-States-Cities-database/blob/master/countries.json

GET http://localhost:8080/weatherapp/countrys

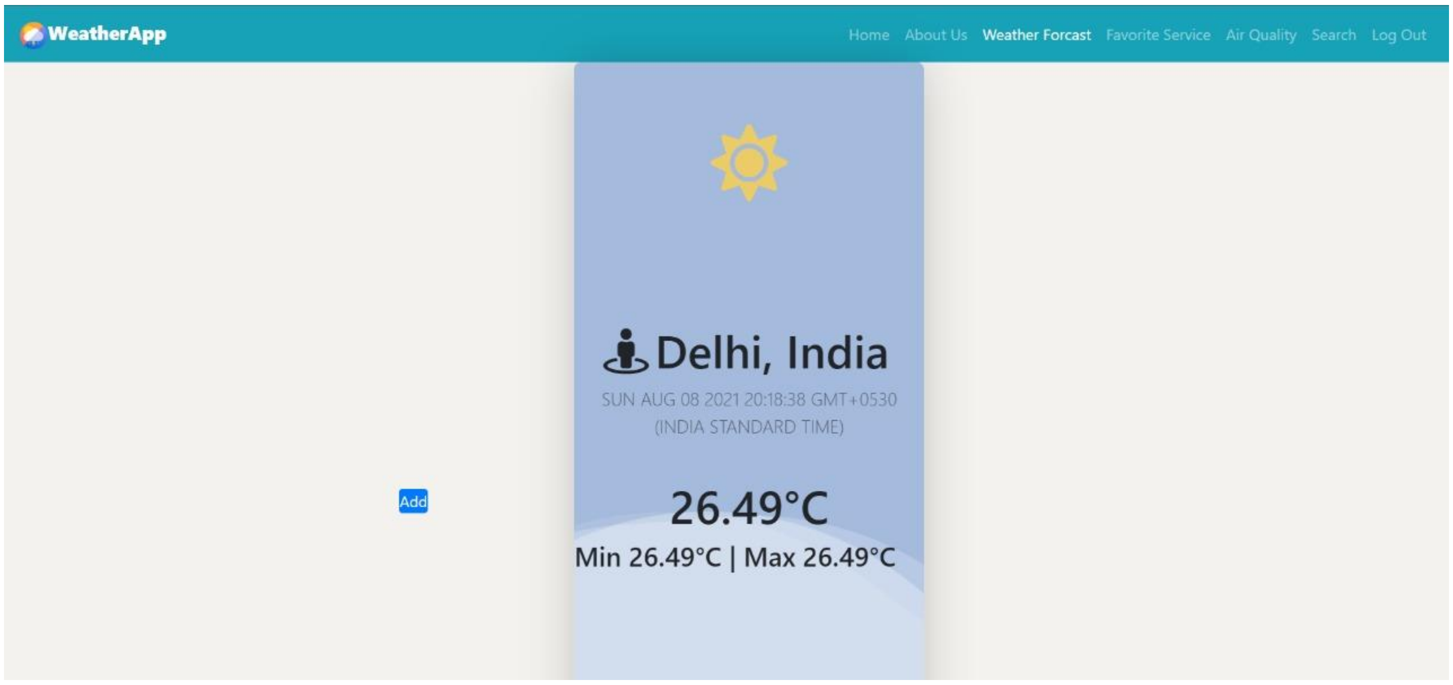
GET http://localhost:8080/weatherapp/countrys

GET http://localhost:8080/weatherapp/countrys

Body Cookies Headers (14) Test Results — Status: 200 OK Time: 100 ms Size: 505 B — Save Response

Pretty Raw Preview Visualize JSON

```
{
    "countryname": "India",
    "statename1": "Kerala",
    "statename2": "TamilNadu"
}
```

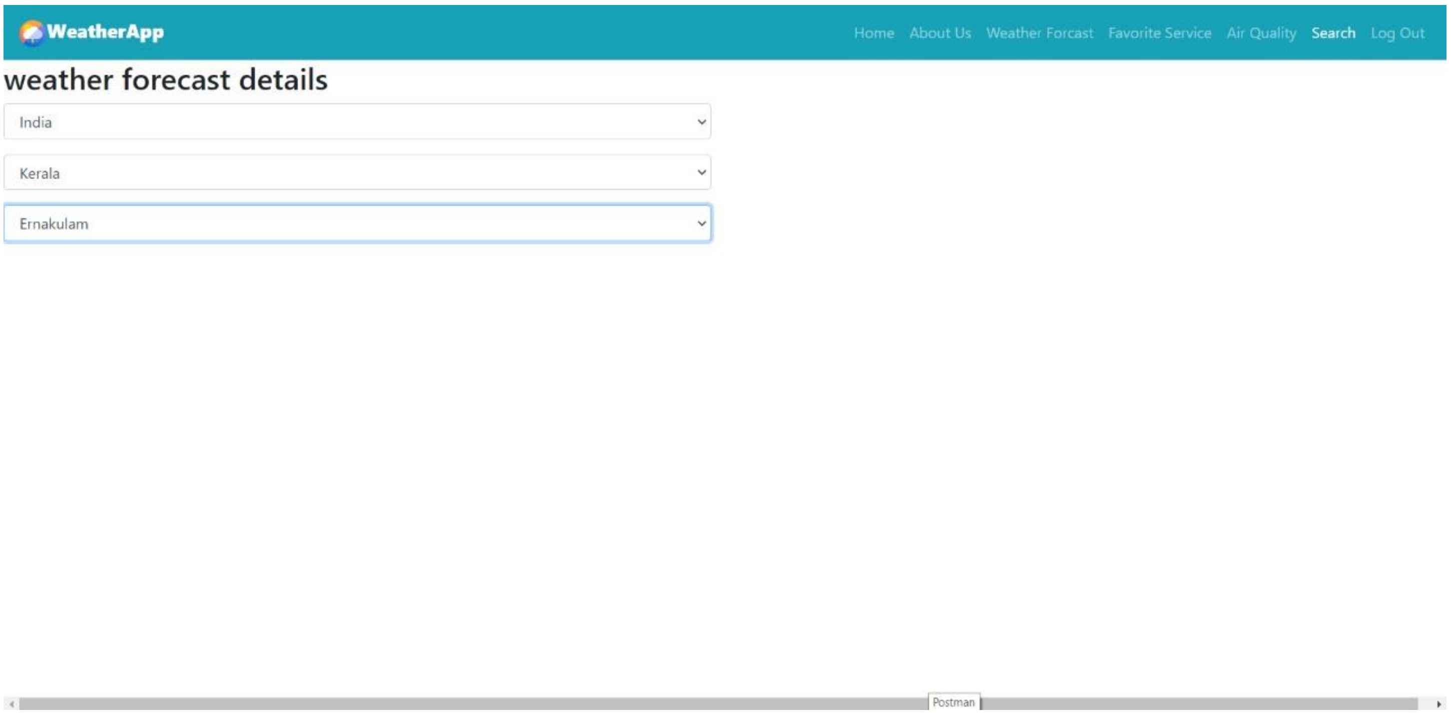


Delhi, India

SUN AUG 08 2021 20:18:38 GMT+0530 (INDIA STANDARD TIME)

26.49°C

Min 26.49°C | Max 26.49°C

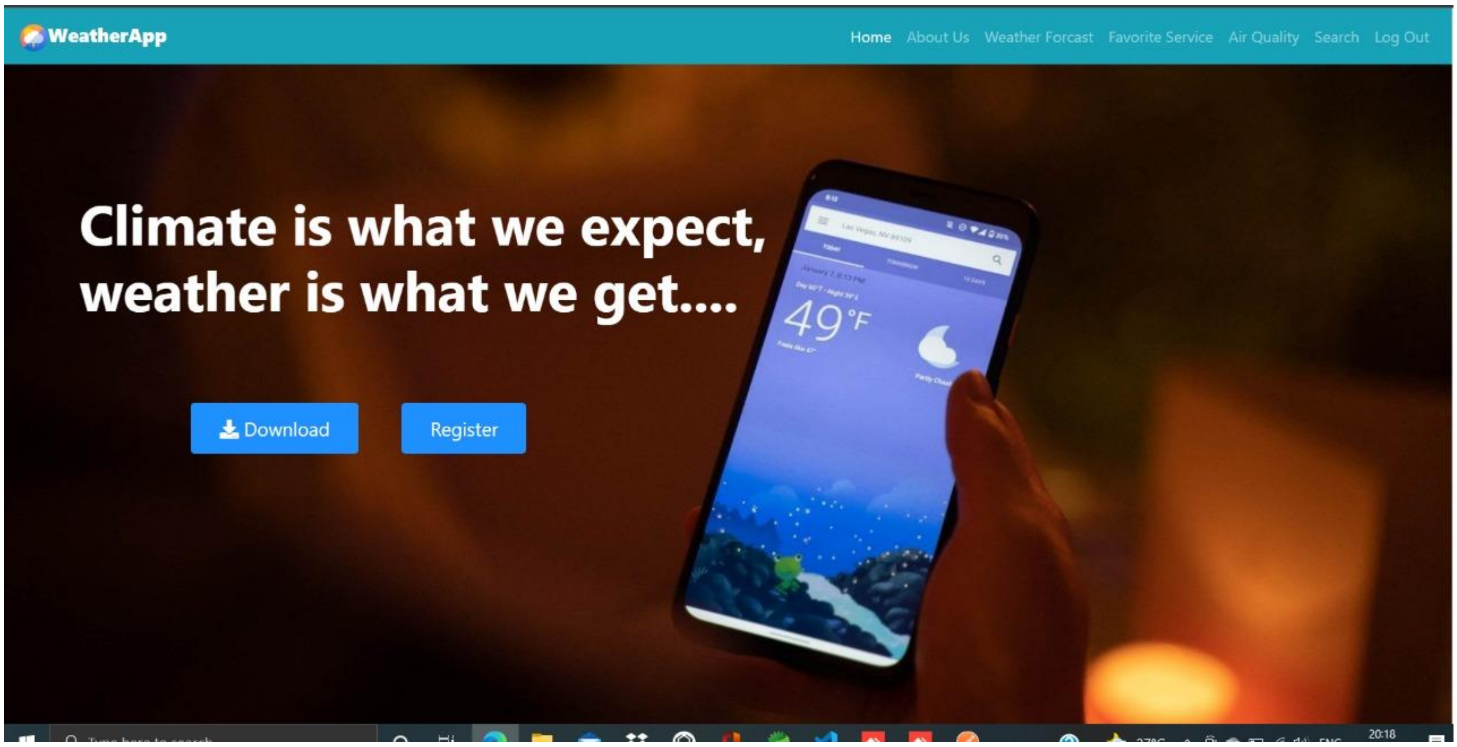

WeatherApp
Home About Us Weather Forcast Favorite Service Air Quality Search Log Out

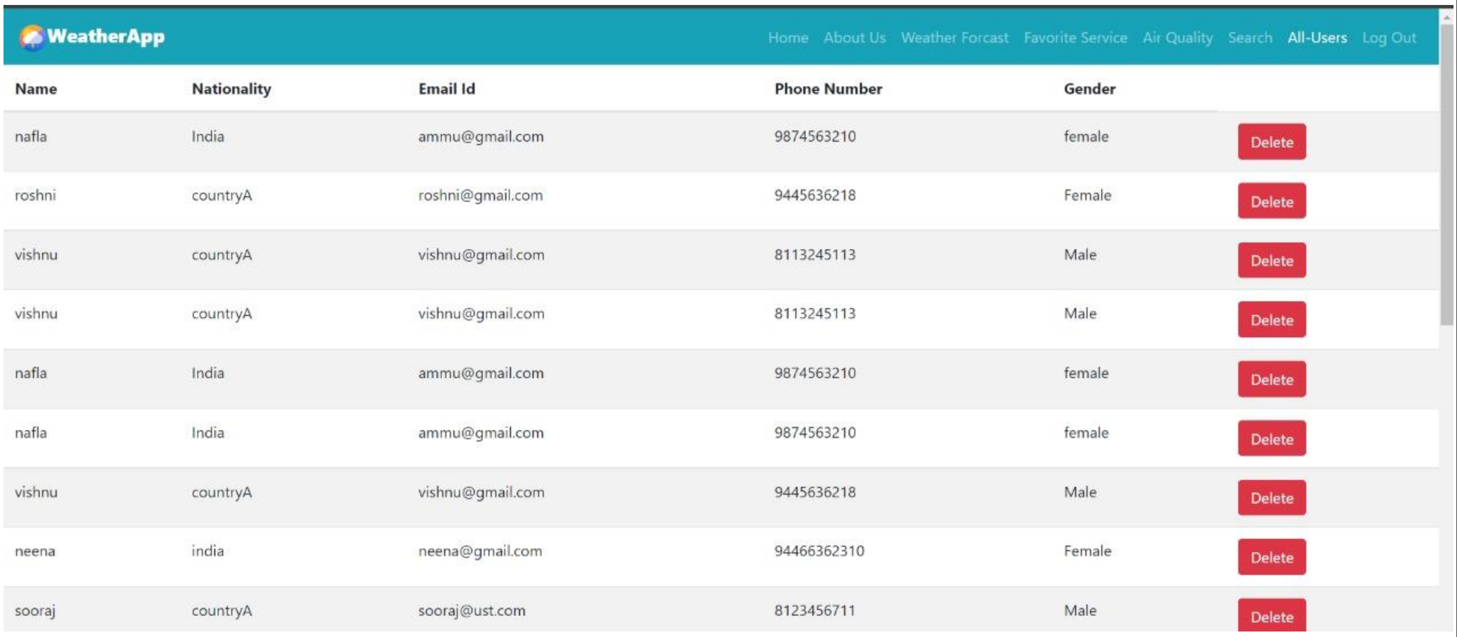
weather forecast details

India

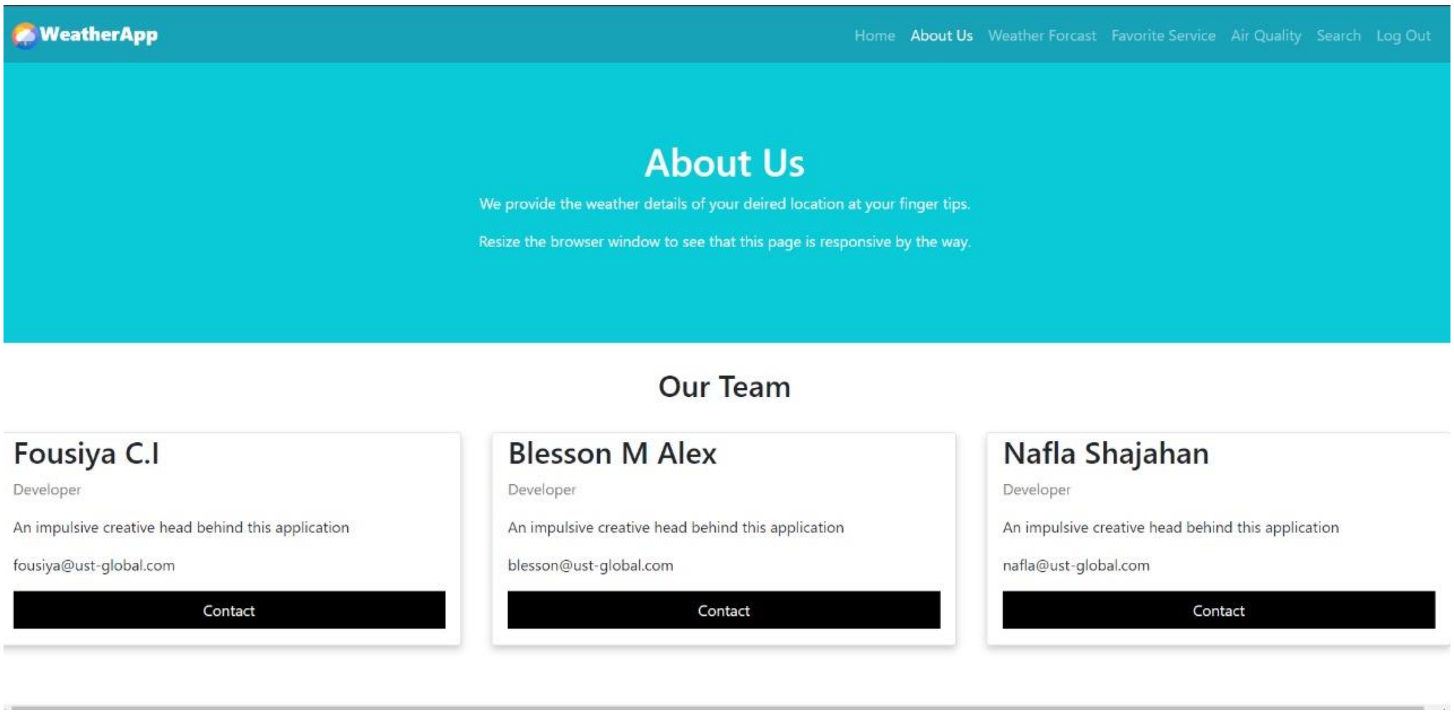
Kerala

Ernakulam


WeatherApp
Home About Us Weather Forcast Favorite Service Air Quality Search Log Out
Climate is what we expect, weather is what we get....
Download
Register


WeatherApp
Home About Us Weather Forcast Favorite Service Air Quality Search All-Users Log Out

Name	Nationality	Email Id	Phone Number	Gender	
nafla	India	ammu@gmail.com	9874563210	female	Delete
roshni	countryA	roshni@gmail.com	9445636218	Female	Delete
vishnu	countryA	vishnu@gmail.com	8113245113	Male	Delete
vishnu	countryA	vishnu@gmail.com	8113245113	Male	Delete
nafla	India	ammu@gmail.com	9874563210	female	Delete
nafla	India	ammu@gmail.com	9874563210	female	Delete
vishnu	countryA	vishnu@gmail.com	9445636218	Male	Delete
neena	india	neena@gmail.com	94466362310	Female	Delete
sooraj	countryA	sooraj@ust.com	8123456711	Male	Delete


WeatherApp
Home About Us Weather Forcast Favorite Service Air Quality Search Log Out

About Us

We provide the weather details of your deired location at your finger tips.

Resize the browser window to see that this page is responsive by the way.

Our Team

Fousiya C.I
Developer
An impulsive creative head behind this application
fousiya@ust-global.com

Blesson M Alex
Developer
An impulsive creative head behind this application
blesson@ust-global.com

Nafla Shajahan
Developer
An impulsive creative head behind this application
nafla@ust-global.com

CHAPTER V

TECHNOLOGIES USED IN PROJECT

➢ VCS	: Gitlab
➢ Middleware	: Spring
➢ Frond end	: Angular
➢ Data Store	: MongoDB/MySQL
➢ Testing	: Karma, Jasmine, Junit, Mockito
➢ Containerization	: Docker

CHAPTER VI

LIMITATIONS & SCOPE

6.1. Limitations

- The API using is an external API so if there is any error in their server it will affect our system too.
- The atmosphere is full of irregular flows of air that form clouds, power storms, and push around cold fronts--that build on each other and form layers. All that variation and uncertainty is why there's a limit to how far out we can meaningfully predict the weather.

6.2. Scope

This proposed application can be created along with a server that has the weather data so that dependency on other APIs could be avoided. This will improve the data accuracy and also the performance of the system. The system can also be upgraded with automatic location fetching abilities so that live weather details can be accessed for the user.

CHAPTER VII

REFERENCES

- https://openweathermap.org/api
- https://www.airvisual.com/air-pollution-data-api
- https://www.javatpoint.com
- https://www.w3schools.com/bootstrap
- https://developer.here.com/blog/collect-forecast-information-with-angular-and-the-here-weather-api
- https://www.digitalocean.com/community/tutorials/how-to-build-a-weather-app-with-angular-bootstrap-and-the-apixu-api