**Internship Project**

|  |  |  |
| --- | --- | --- |
| Company | : | Lucid Technologies & Solutions |
| Project | : | Weather Report based on User’s Preference |
| Dated by | : | Oct 2020 |
| Submitted by | : | 1. Subashree V 2. Sriram B |

**TABLE OF CONTENTS**

Table of Contents

[1. Introduction 3](#_Toc192761210)

[2. Prerequisites 3](#_Toc192761211)

[2.1 Software Requirements 3](#_Toc192761212)

[2.2 API Requirements 3](#_Toc192761213)

[2.3 Language Requirements 3](#_Toc192761214)

[2.4 User Requirements 3](#_Toc192761215)

[3. Functionality Description 4](#_Toc192761216)

[3.1 Sequence Diagram 4](#_Toc192761217)

[3.2 Modules 5](#_Toc192761218)

[4. Screenshots 6](#_Toc192761219)

[5. Code Implementations 9](#_Toc192761220)

[6. Test Cases 17](#_Toc192761221)

[7. Conclusion 18](#_Toc192761222)

[8. References 18](#_Toc192761223)

## 1. Introduction

As known, Climatic conditions play a vital role in our daily life. Keeping in mind the sudden changes in Climatic conditions, this project keeps an eye on the end users requirement and generates a weather report based on the user's choice. This project gets the detailed requirement of the user by allowing the user to enter the country, state and city from the set listed. Once getting the requirements, the user has given the option to view the climatic condition of the current day or for the past week. Once the particular option is chosen, a weather report for the user’s wis will be generated.

## 2. Prerequisites

### 2.1 Software Requirements

* Eclipse Java EE Edition
* Postman
* Server - Apache Tomcat

### 2.2 API Requirements

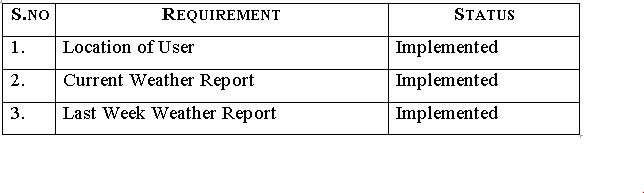
* Current Weather Report API
* Past Week Weather Report API
* Country, State and City API

### 2.3 Language Requirements

* Groovy
* Java

### 2.4 User Requirements

The following are the user requirements for this project,



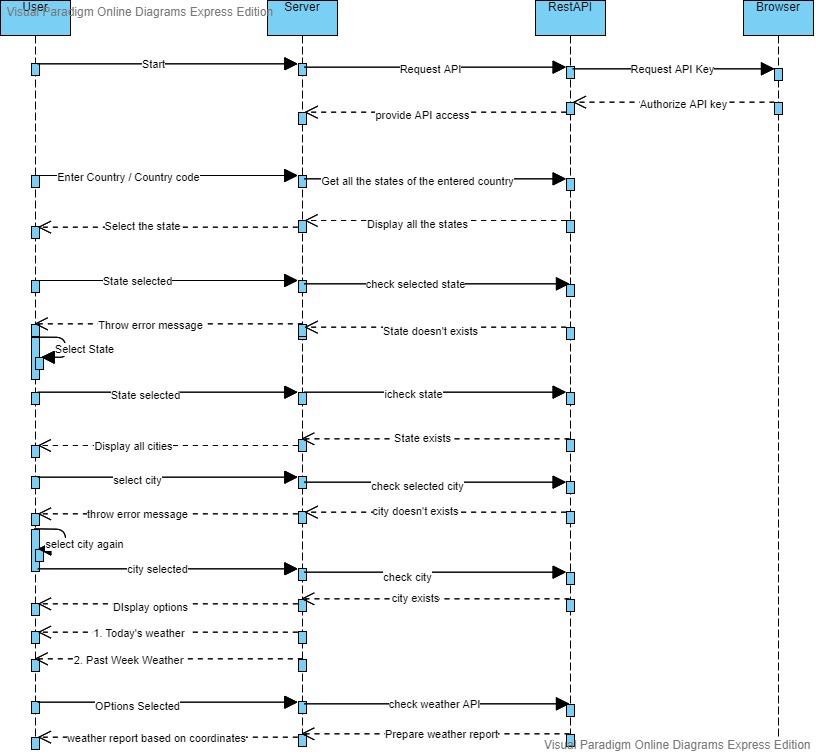
## 3. Functionality Description

The list of modules present in the weather report system is listed below

* Display Countries
* Display States
* Display Cities
* Display weather of particular location

The functionality for the above modules will be described below.

### 3.1 Sequence Diagram



### 3.2 Modules

#### 3.2.1 Display Countries

* The user will be displayed with a list of country names along with the country code from the API server.
* The user should enter either country name or country code to get a weather report.
* If a user enters a country name, the country code associated with the country name will be found and the appropriate regions / states present in the country will be displayed to the user.
* If a user enters country code, the list of regions / states present in the country will be displayed to the user.
* If a user enters the wrong country name or code, they will be suggested to re-enter the details again.

#### 3.2.2 Display States

* The user gets the details of the States present in the particular country.
* The user needs to enter the region / state name specifically to get the weather report.
* If the user enters the state name correctly, then the list of cities present in the particular county’s state will be displayed to the user.
* If user enters wrong state name, they will be suggested to re – enter the state name again

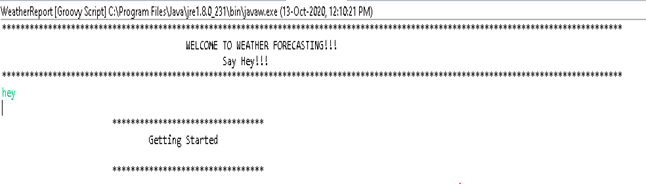
#### 3.2.3 Display City

* The user gets cities of a particular state from the API.
* The user needs to enter the specific city name in order to know the weather condition of that location.
* If the user enters the correct city name, then the location entered by the user will be displayed.
* If user enters wrong city name, they will be suggested to re – enter the city name again

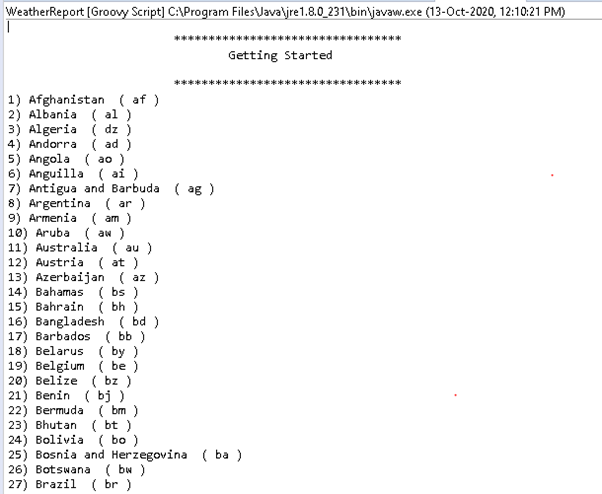
#### 3.2.4 Display Weather Report of Location

* The user will be given 2 options to know the weather report of the particular location.
* The first option is to know the current (today’s) weather report.
* The second option is to know last week’s weather report.
* If the user chooses option 1, then today’s weather condition will be displayed.
* If the user chooses option2, then last week’s weather condition will be displayed.

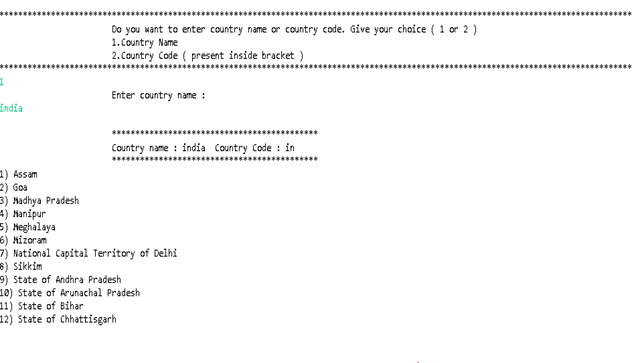
## 4. Screenshots



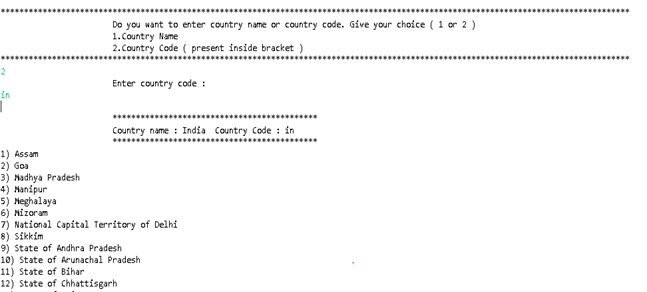
*Fig 4.1 Welcome Page*



*Fig 4.2 Displaying Country List*



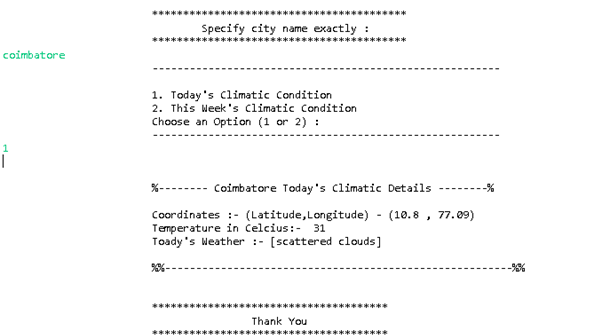
*Fig 4.3 Choosing Country Name*



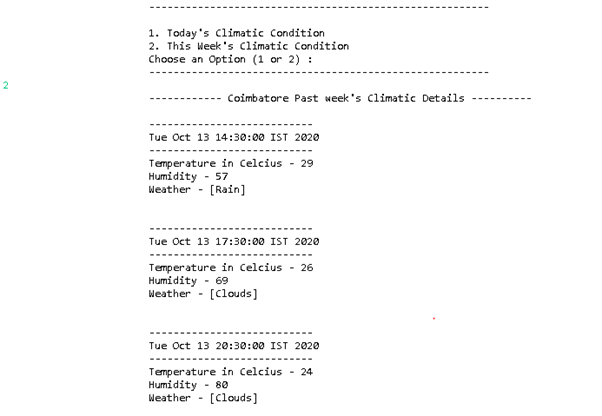
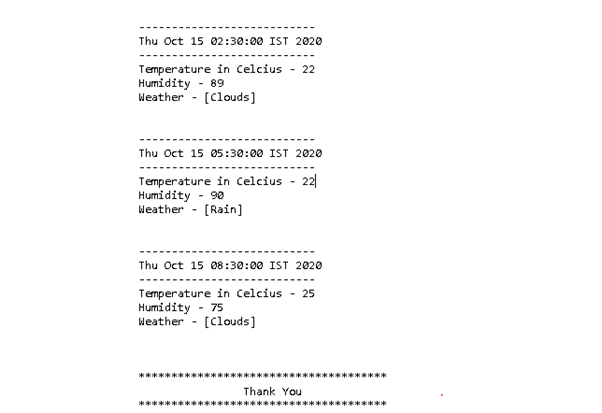
*4.4 Choosing Country Code*



*Fig 4.5 Choosing Region Name*



*Fig 4.6 Choosing city to know current weather*

*Fig 4.7 Last Week’s Weather report*

## 5. Code Implementations

import groovy.json.JsonSlurper

import groovy.json.JsonOutputt tharen

import java.util.Scanner;

import java.util.List

import java.time.Instant

import groovy.json.JsonBuilder

//Creating JSON Slurper object

JsonSlurper js= new JsonSlurper()

// Global Variables

def global // global flag variable to encounter the flow of process

def counCode // Country Code variable

def counName // Country Name variable

def regionName // Region / State Name variable

def result // Stores resultant location details in the form of list

def cityAPI // API for location tracking

// Displaying Country with code details

def dispCountry(){

println("\n\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

println("\t\t\tCountry name : " + counName + " Country Code : " + counCode)

println("\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

}

// Used for user interface

def design(){

println("\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

}

// Sample for ending code. Mainly for user interface

def endFun(){

println("\n\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

println("\t\t\t\t\tThank You")

println("\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

}

// Displaying All Country List present in world

def dispCounList(){

// URL for displaying Country List

def counCodeAPI = "http://battuta.medunes.net/api/country/all/?key=f6aa01d280721e65f81f6a9f1bf2a1fa"

// Converting URL into String

def counList = new JsonSlurper().parseText(new URL(counCodeAPI).getText())

//Getting Size or Length of List

int counListSize = counList.size()

// Iterating through the list ( String )

for(int i = 0; i < counListSize; i++){

println(i+1 + ") " + counList.get(i).name + " ( " + counList.get(i).code + " )")

}

// Loop variable used as flag signal for looping

boolean loop = true

while(loop){

println("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

println("\t\t\tDo you want to enter country name or country code. Give your choice ( 1 or 2 )")

println("\t\t\t1.Country Name \n \t\t\t2.Country Code ( present inside bracket ) ")

println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

// Getting choice from user

def choice = System.in.newReader().readLine()

int flag = 0 // Setting flag as 0

global = "0"

if(choice == "1"){ // User wishes to enter country name

while(loop){

println("\t\t\tEnter country name : ")

// Getting country name from user

counName = System.in.newReader().readLine()

// Generating Country's Code

for(int i = 0; i < counListSize; i++){

if(counList.get(i).name.toLowerCase() == counName.toLowerCase()){ //Checking input is present in the API list

counCode = counList.get(i).code // Getting appropriate country's code

flag = 1 // To indicate Country is present

global = "1"

break

}

}

if(flag == 0){ // Flag not set so country is not available

design()

println("\t\t\t Invalid Country Name@")

println("\t\t\t Press 1 to re-enter 0 to stop")

design()

flag = System.in.newReader().readLine() // Option for user to re-enter or exit

if(flag == "1") // User needs to re-enter Country name

loop = true

else // User needs to exit

loop = false

}

else{

dispCountry() // Displaying country name with country code to user function call

loop = false // Country is choosed. So end of loop

}

}

loop = false

}

else if(choice == "2"){ // User wishes to enter country code

while(loop){

println("\t\t\tEnter country code : ")

counCode = System.in.newReader().readLine() // Getting country code

// Generating country name from country code

for(int i = 0; i < counListSize; i++){

if(counList.get(i).code.toLowerCase() == counCode.toLowerCase()){ //Checking input is present in the API list

counName = counList.get(i).name// Getting appropriate country's code

flag = 1 // To indicate country is present

global = "1"

break

}

}

if(flag == 0){ // Flag not set so country is not available

design()

println("\t\t\t @Invalid Country Code@")

println("\t\t\tPress 1 to re-enter 0 to stop")

design()

flag = System.in.newReader().readLine() // Option for user to re-enter or exit

if(flag == "1") // User needs to re-enter Country name

loop = true

else // User needs to exit

loop = false

}

else{

dispCountry() // Displaying country name with country code to user function call

loop = false // Country is chosen. So end of loop

}

}

loop = false

}

else{

design()

println("\t\t\t @Invalid choice. Press 1 or 2 .Try Again@")

println("\t\t\tPress 1 to re-enter 0 to stop")

design()

flag = System.in.newReader().readLine()

if(flag == "1"){ // User wishes to re-enter country name

dispCounList() // Allowing User to re-enter country name

loop = false

}

else{

endFun()

loop = false

}

}

}

return global // to Check whether is choosed or not

}

// Displaying Regions / States List of particular Country

def dispRegionList(){

// URL for displaying Region List of selected Country

def regionAPI = "http://battuta.medunes.net/api/region/" + counCode + "/all/?key=f6aa01d280721e65f81f6a9f1bf2a1fa"

// Converting URL to String

def regionList = new JsonSlurper().parseText(new URL(regionAPI).getText())

// Getting size of regions displayed list

int regionSize = regionList.size()

// Iterating Regions List

for(int i = 0; i < regionSize; i++){

println(i+1 + ") " + regionList.get(i).region)

}

println()

loop = true

global = "0"

while(loop){

design()

println("\t\t\tRegions of country " + counName + " are listed above")

println("\t\t\tEnter the region / state name : ")

design()

// Getting region name from user

regionName = System.in.newReader().readLine()

while(regionName.size() < 2){ // If region name is less than 2 letters

println("Please enter region / state name with atleast 2 characters : ")

regionName = System.in.newReader().readLine()

}

// Generating City list for selected region

def cityAPI = "http://battuta.medunes.net/api/city/" + counCode + "/search/?region=" + regionName + "&key=f6aa01d280721e65f81f6a9f1bf2a1fa"

// Converting URL to String

def cityList = new JsonSlurper().parseText(new URL(cityAPI).getText())

if(cityList.size() == 0){ // No such region found

design()

println("\t\t\t @Invalid region / state name.\n\t\t\t Try Again@")

println("\t\t\tPress 1 to re-enter 0 to stop")

design()

flag = System.in.newReader().readLine() // Getting user opinion

if(flag == "1"){ // User wishes to re-enter region name

loop = true

}

else{ // User wishes to exit

endFun()

loop = false

}

}

else{

design()

global = "1"

println("\t\t\t\t\tRegion exists")

design()

loop = false

}

}

return global

}

// Displaying City List of particular Country's State

def dispCityList(){

// Displaying City of particular region

def partCity = "http://battuta.medunes.net/api/city/" + counCode + "/search/?region=" + regionName + "&key=f6aa01d280721e65f81f6a9f1bf2a1fa"

// Converting URL to String

def cityList = new JsonSlurper().parseText(new URL(partCity).getText())

// Iterating City list

for(int i = 0; i < cityList.size(); i++){

println(i+1 + ") " + cityList.get(i).city)

}

println()

loop = true

while(loop){

design()

println("\t\t\t\tSpecify city name exactly : ")

design()

// Getting city name from user

cityName = System.in.newReader().readLine()

while(cityName.size() < 3){ // If city name is less than 3 letters

println("Please enter city name with atleast 3 characters : ")

cityName = System.in.newReader().readLine()

}

cityAPI = "http://battuta.medunes.net/api/city/" + counCode + "/search/?region=" + regionName + "&city=" + cityName + "&key=f6aa01d280721e65f81f6a9f1bf2a1fa"

result = new JsonSlurper().parseText(new URL(cityAPI).getText())

if(result.size() == 0){ // No such city found

design()

println("\t\t\t @Invalid city name.\n\t\t\t Try Again@")

println("\t\t\tPress 1 to re-enter 0 to stop")

design()

flag = System.in.newReader().readLine() // Getting user's opinion

if(flag == "1"){ // user wishes to re-enter city name

loop = true

}

else{ // user wishes to exit

endFun()

loop = false

}

}

else if(result.size() >= 2){ // If two or more cities found with specified name

design()

println("\t\t\t\tMetion city name exactly. Try again")

design()

loop = true

}

else if(result.size() == 1){

Loop = true

while(loop){

println("\t\t\t--------------------------------------------------------")

println("\n\t\t\t1. Today's Climatic Condition")

println("\t\t\t2. This Week's Climatic Condition")

println("\t\t\tChoose an Option (1 or 2) : ")

println("\t\t\t--------------------------------------------------------")

Scanner scan = new Scanner(System.in); // creating Scanner class object

int num = scan.nextInt() // Getting user opinion

JsonSlurper js= new JsonSlurper()

String cityname=result.get(0).city // Generating city name from list

String latitude=result.get(0).latitude // Generating latitude

String longitude=result.get(0).longitude // Generating longitude

String api="http://api.openweathermap.org/data/2.5/" // API for weather

String appid="&appid=4c19fc141ec73cc06973aca5a9a8dc42" // Access key for weather API

if (num ==1) {

String today="weather?lat="+latitude+"&lon="+longitude

// Converting URL to Map

Map map = js.parseText(new URL (api+today+appid).getText())

// Getting required detail from map

String lat = map.coord.lat

String lon = map.coord.lon

double temp = map.main.temp -273.15

String weather = map.weather.description

// Displaying today's weather condition

println("\n")

println("\t\t\t%-------- "+ cityname +" Today's Climatic Details --------%\n")

println("\t\t\tCoordinates :- (Latitude,Longitude) - ("+lat+" , "+lon+")")

println("\t\t\tTemperature in Celcius:- "+ Math.round(temp))

println("\t\t\tToady's Weather :- "+weather)

println("\n\t\t\t%%--------------------------------------------------------%%")

println()

Loop = false

}

else {

String thisweek="forecast?lat="+latitude+"&lon="+longitude

// Displaying last week's weather condition

String count="&cnt=15"

Map map = js.parseText(new URL (api+thisweek+count+appid).getText())

int c=1

println("\t\t\t------------ "+ cityname +" Past week's Climatic Details ----------\n")

for(i in map.list) {

Instant instant = Instant.ofEpochSecond( i.dt );

Date date = Date.from( instant )

println("\t\t\t---------------------------")

println("\t\t\t" + date)

println("\t\t\t---------------------------")

println("\t\t\tTemperature in Celcius - "+Math.round(i.main.temp -273.15))

println("\t\t\tHumidity - "+ i.main.humidity)

println("\t\t\tWeather - "+ i.weather.main)

println("\n")

Loop = false

}

}

Else{

design()

println(“Invalid Input”)

design()

Loop = true

}

}

loop = false

}

}

}

// Displaying API for weather reports

def dispAPI(){

design()

println("\n\t\t\t\t\tAPI")

println("\n" + cityAPI)

design()

}

// Main Part where program Starts

println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

println("\t\t\t\t\tWELCOME TO WEATHER FORECASTING!!!")

println("\t\t\t\t\t\tSay Hey!!!")

println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

def input = System.in.newReader().readLine()

if(input.toLowerCase() == "hey"){

println("\n\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

println("\t\t\t\tGetting Started")

println("\n\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

boolean flag = true

global = "0"

while(flag){

global = dispCounList()

if(global == "1"){ // Country is selected

global = "0"

global = dispRegionList()

if(global == "1"){ // Region is selected

global = "0"

dispCityList()

endFun()

}

else{ // Region is not selected

println()

design()

println("\t\t\tState not mentioned!")

design()

}

}

else{ // Country is not selected

println()

design()

println("\t\t\t Country not metioned!")

design()

}

println("Press 1 to continue or 0 to stop")

input = System.in.newReader().readLine() // input for another location's weather condition

if(input == "1")

flag = true

else

flag = false

}

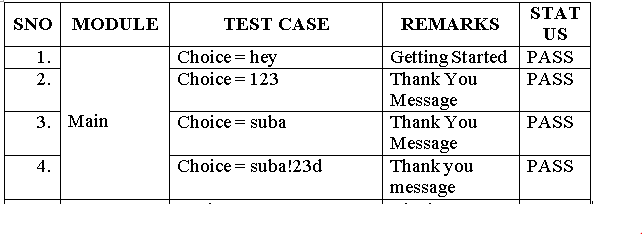
}

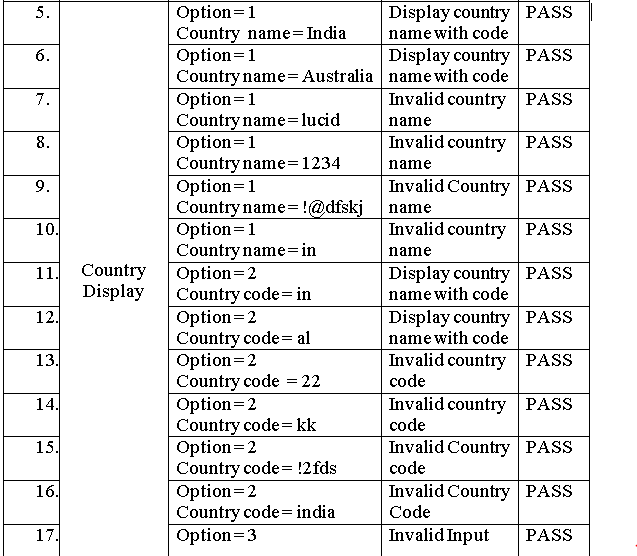
else{

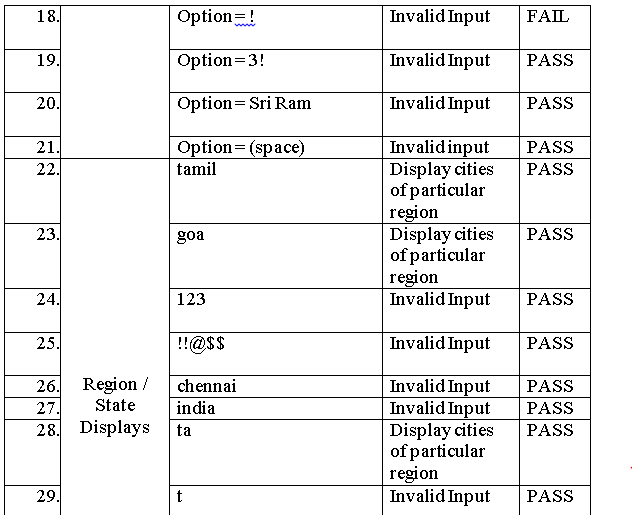
endFun()

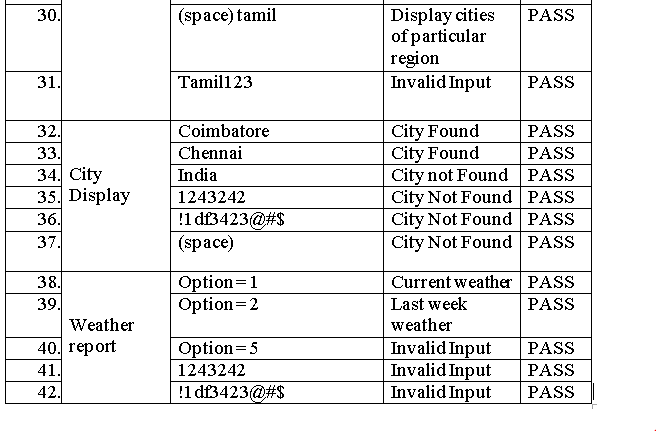
}

## 6. Test Cases

****

****

****

****

## 7. Conclusion

Thus, this project understands and gathers the user’s requirements and provides a detailed weather report based on the user's preference.

## 8. References

1. <https://www.eclipse.org/downloads/packages/release/helios/sr1/eclipse-ide-java-developers>
2. <https://www.postman.com/downloads/>
3. <https://rapidapi.com/teams?utm_source=google&utm_medium=cpc&utm_campaign=Teams_80056715881&utm_term=%2Bapi%20%2Btesting_b&gclid=EAIaIQobChMIkpKbobux7AIVmAVyCh2o7QiQEAAYASAAEgIXQfD_BwE>
4. <https://www.tutorialspoint.com/groovy/index.htm>
5. <https://www.guru99.com/groovy-tutorial.html>
6. <https://httpd.apache.org/download.cgi>
7. <https://openweathermap.org/>
8. <https://openweathermap.org/current>
9. <https://openweathermap.org/forecast16>
10. <https://tomcat.apache.org/download-90.cgi>
11. <http://battuta.medunes.net/api/country/all/?key=f6aa01d280721e65f81f6a9f1bf2a1fa>

***\*\*\* End of the document \*\*\****