Weather Forecast

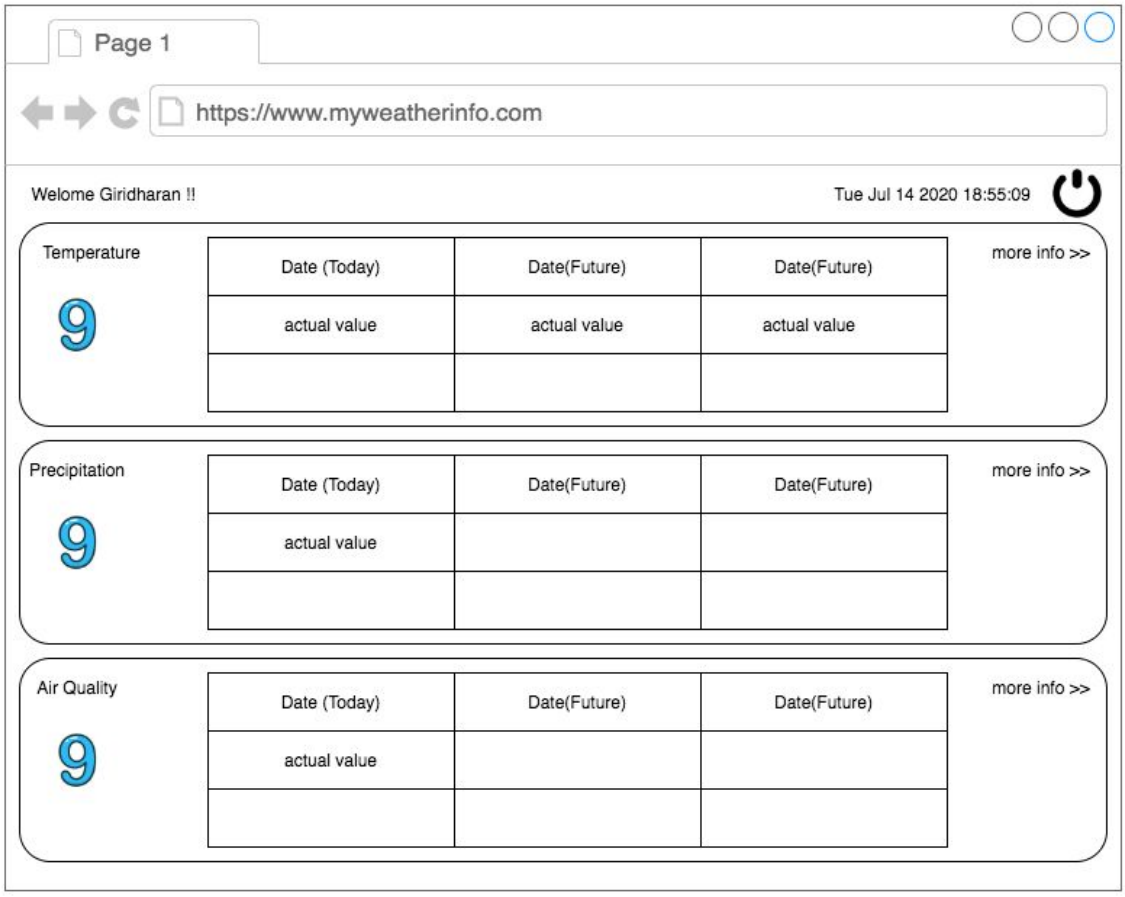
Objective

To develop a Weather forecasting application which streams air pollution and weather forecast data from an open weather API using Spring boot and Mysql.

Sketch

The below diagram represents the Landing page when an user logged in. The basic information that are displayed :
1.User’s name
2.Current date
3.Digital clock
4.Temperature
5.Precipitation
6.Air Quality
Temperature,Precipitation and Air quality are forecasted on a daily basis.

Landing Page Version 1



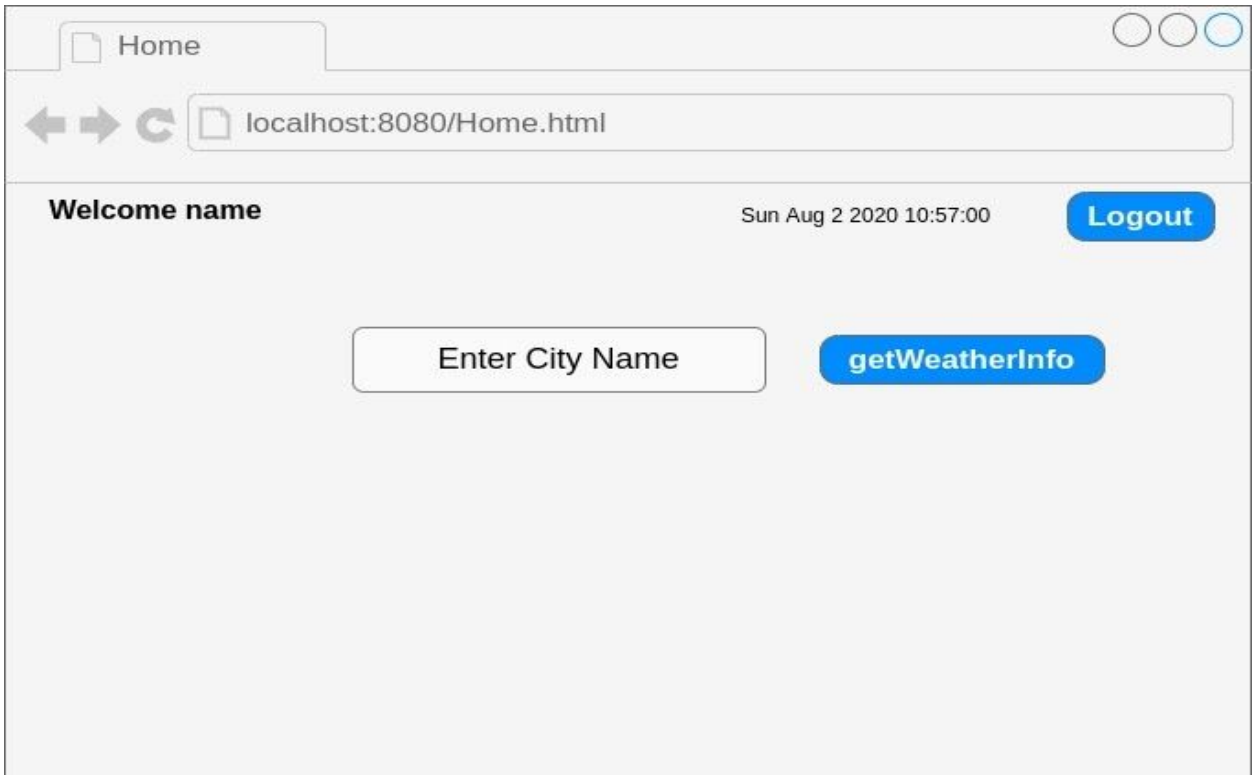
In version 2 - replace tables with appropriate charts.

In version 3 - we will implement what will happen on click of more info.

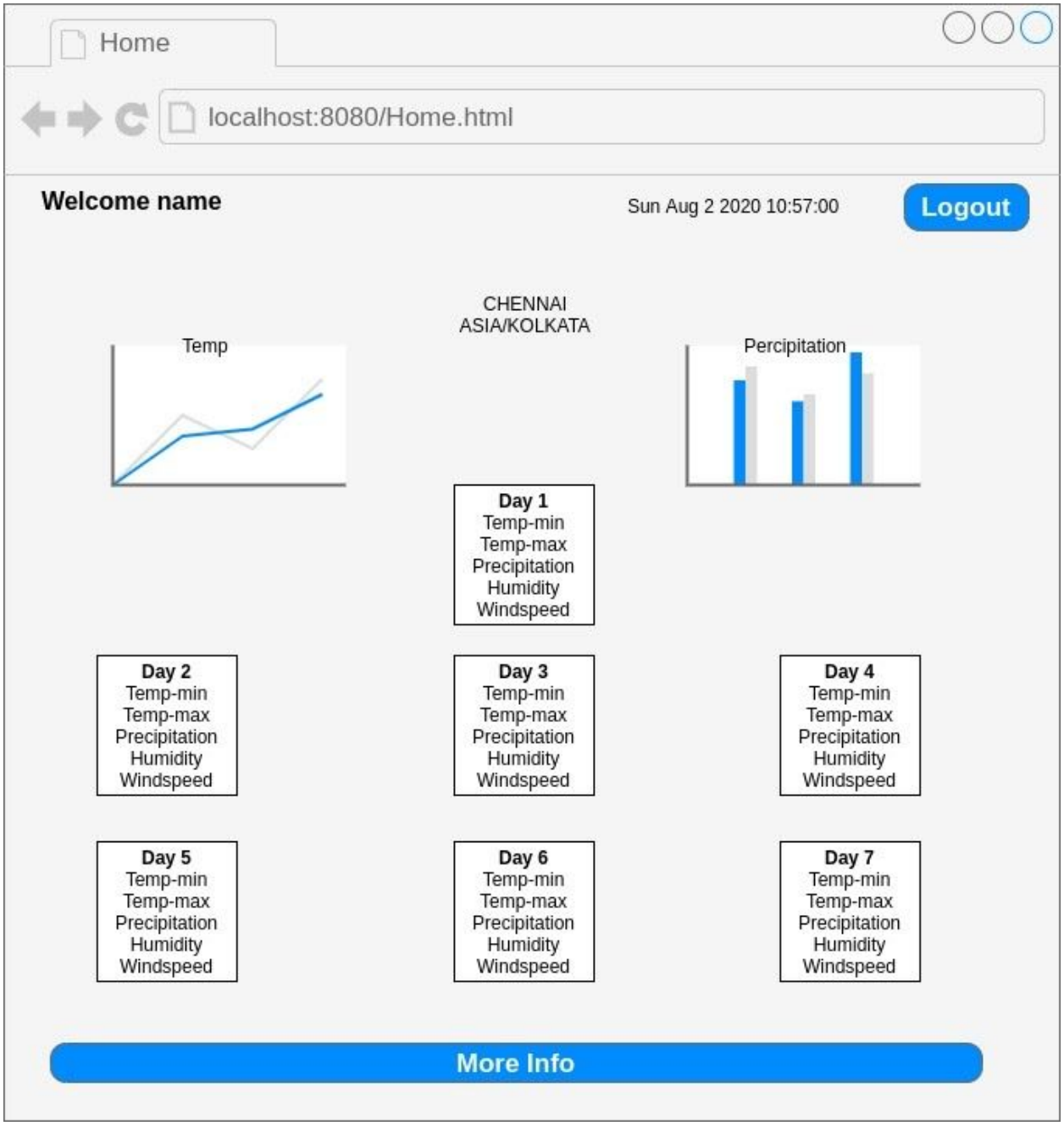
Design

From the sketch, some components are reformed. Instead of presenting temperature, precipitation and air quality as separate components, they are reformed and replaced as *Information blocks* where these components are included. The basic information to be presented in the Information block are Temperature,Precipitation,Humidity and Wind speed. Temperature and Precipitation are graphically represented.

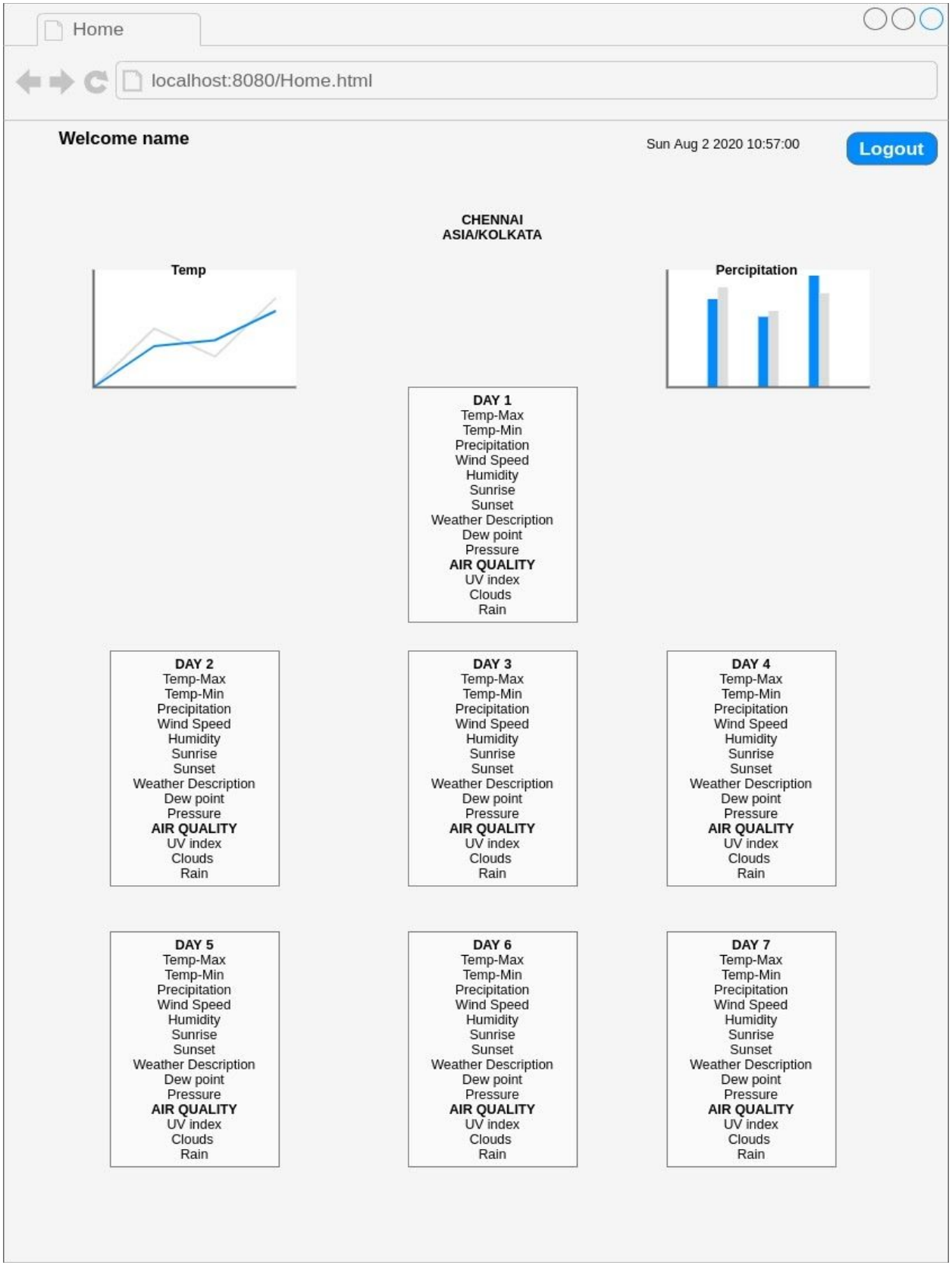
Home Page


Home
localhost:8080/Home.html
Welcome name
Sun Aug 2 2020 10:57:00
Logout
Enter City Name
getWeatherInfo

After Entering the City name



After clicking more info button



GITHUB

The below repository has all the source code.
https://github.com/giridharan-01/WeatherApp

Schema

In this project we have two tables

1.projuser

This table contains all the users registered in this application. The Structure will be

ID	NAME	USERNAME	PASSWORD
Datatype: Int Primary key	Datatype: Varchar Not null	Datatype: Varchar Unique key	Datatype: Varchar Not null

The *Username* field contains the user's email-id and *Password* field is encrypted and stored.

2.citydetails

This table contains all the information regarding the main cities.There are about 2,09,579 records dumped in this table. The Structure will be

COUNTRY	STATE	ID	COORD	NAME
Datatype: Text Not null	Datatype: Text	Datatype: Int Not null	Datatype: Text Not null	Datatype: Text Not null

The *Coord* field contains both latitude and longitude information. The *Id* field will be unique. The *Name* field can be the same for many entities but they can be distinct by means of *State* and *country*. Example, for city *Delhi,* it exist in five distinct locations such as

Country	State	Name
IN		Delhi
US	LA	Delhi
US	NY	Delhi
US	CA	Delhi
CA		Delhi

Implementation

When the user starts typing, an autocomplete function calls the *search* api which will provide a list of city suggestions to the user. The user can either choose from the list or can enter the city name directly. When the user selects from the suggestion list and clicks the *getWeatherInfo* button it will call a function to check whether the city exists or not through the *test* api at the backend. If the api call returns null, then it displays "City not found".Else it will return a series of seven day forecasts which will be fed into the information blocks.
There are three main functions in the program :

- reinitialize
- getWeatherData
- getWeatherInfoOfTheDay.

The reinitialize function is used when any error is encountered. The errors can be of either an invalid city name or an empty input.
The getWeatherData function fetches the basic information for the given city using *test* api and stored in variable *output* .The getWeatherInfoOfTheDay function fetches all the information for the given city using the *output* variable and mapped.

Tech stacks used

- Spring boot
- Spring data
- REST API
- Mysql
- Bootstrap
- Jquery and Javascript

- CSS and HTML

Execution

In the execution part when the user hits the *getWeatherInfo* button it first calls the getId function. In the *getId* function, the input gets split into an array. If the array is of length more than 1 then it calls *getId* api to fetch the city id and calls the *getWeatherData* function.
There are three main apis in the backend:

- search - to suggest similar city names.
- test - sends the weather information of that city.
- getId - sends city id.

And there are two more apis:

- login - to check whether the user is registered in this application or not.
- reg - to register a new user to this application.

The application can be run using the following command :
java -jar MainProj-0.0.1-SNAPSHOT.jar

The application can be built using the following command :
mvn clean install

Advantages

This application can forecast the following seven days starting from the current day.
This application suggests some city names while the user is typing in the input field (autocompletes are implemented).
This application is responsive in nature and it is compatible with all browsers(using Bootstrap and Chart.js).

Limitations

This application will display only standard metrics such as Temperature in celsius etc.
This application cant forecast on an hourly basis of a day.

Future Scope

Conversion of units will be added later. For example, for temperatures in celsius to kelvin etc.
Hourly forecast will be added later in this application.
Historical analysis will be added so that users can get all information for a particular period of time.
Spring security related features will also be added.