**ABSTRACT**

|  |
| --- |
| The main objective is to develop an application for the end user to access weather of any location of their choice of that particular day. Also, to provide accurate information to the user who’s  seeking it irrespective of location. The user shall be provided with a vague description of the current status of the atmosphere along with few other parameters such as pressure, humidity, maximum and minimum temperature. We will be using  Openweather map API which is a free API that has restrictions on number of API calls per minute in the free package. We will be building a fairly simple and user-friendly application that will help the user’s view the live weather data for a desired city. We will not be focusing on historical or weather-forecast and rather will keep the application simple and only concentrate on the live data. |

**TABLE OF CONTENT**

1. Introduction 3
2. Design/ Implementation 4
3. Result/Analysis 7
4. Conclusion 9
5. Future Updates 10
6. References 11

|  |
| --- |
|  |

**INTRODUCTION**

Weather forecasting is the application of science and technology to predict the conditions of the atmosphere for a given location and time. Java is a general-purpose computer-programming language that is concurrent, class-based, object-oriented and specifically designed to have as few implementation dependencies as possible. We will use API, an application programming interface (API) is a set of subroutine definitions, communication protocols, and tools for building software.. An API may be for a web-based system, operating system, database system, computer hardware, or library. Just as a graphical user interface (GUI) makes it easier for people to use programs, application programming interfaces make it easier for developers to use certain technologies in building applications. The algorithm a computer uses to solve equations is a big part of why different weather models give different predictions. Weather depends on things at small and large scales.  A thermometer may measure the temperature to the nearest tenth of a degree.  A computer usually goes to 16 decimal places.But a chaotic system needs far more precision. Example, the position of every leaf on every tree on Earth, which is practically impossible to locate at any instant time. It is only in recent years that the development of chaos theory has given some insight into this. In addition, the topographic and oceanic complexity of the earth’s surface makes any long range forecasting beyond a few weeks near impossible and any specific forecasting for local effects like showers limited to. Basically, every application nowadays on the internet shows different weather.

This website provides an accurate and precise weather forecast of any location that we desire.We are designing a desktop application which requires the user to put the location of their desire to get the report of it.

|  |
| --- |
|  |

**DESIGN/IMPLEMENTATION**

Take Input from the user regarding which city he/she would like to know about

Wrong city (city doesn’t belong in the database). Display the appropriate message.

Correct city, take the weather from the API and add it to the GUI

Display appropriate graphics with appropriate message.

STOP

STOP

We have used Java Swing to build our GUI. Unlike AWT, Java Swing provides platform-independent and lightweight components.The javax.swing package provides classes for java swing API such as JButton, JTextField, JFrame, JLabel, etc.

Correct city, take the weather from the API and add it to the GUI

Correct city, take the weather from the API and add it to the GUI

On launching the application, the window frame is displayed asking the user’s input to which city’s weather forecast he/she wants. On pressing the submit button, this high-level event is generated by a button. The event is passed to every ActionListener object that registered to receive such events using the component's addActionListener method. The object that implements the ActionListener interface gets this ActionEvent when the event occurs. A class that implements this interface contains an actionPerformed() method, where the ActionEvent parameter is a Event Object containing the information of the event.

The JOptionPane class is used to provide dialog boxes such as message dialog box, confirm dialog box and input dialog box. These dialog boxes are used to display information or get input from the user. Depending on what the user has entered, the appropriate message is displayed.

If the entered input is the existing city in the API, a different class is called to perform the event. This class contains the required information. We are using JSON Parsing for displaying the data from the server to the user.

PSUEDOCODE:

JSONParserparse = **new**JSONParser();

JSONObjectobj = (JSONObject)parse.parse(inline);

JSONArrayjson\_array = (JSONArray) obj.get("weather");

**for**(**int**i = 0 ; i<json\_array.size() ; i++) {

JSONObjectjson\_obj = (JSONObject)json\_array.get(i);

desc.setText((String) json\_obj.get("description"));

}

JSONObjectmain = (JSONObject) obj.get("main");

temp.setText(String.*valueOf*(main.get("temp")));

pressure.setText(String.*valueOf*(main.get("pressure")));

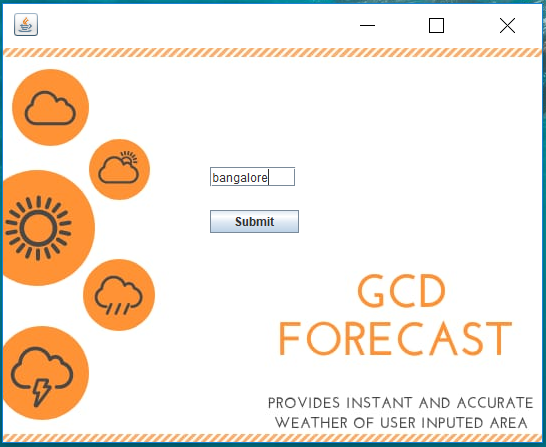
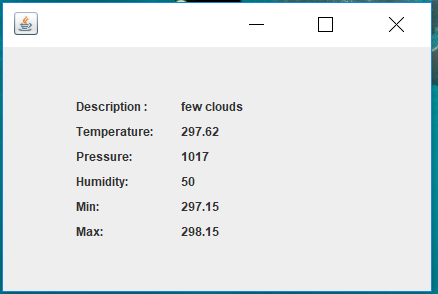
humidity.setText(String.*valueOf*(main.get("humidity")));

min.setText(String.*valueOf*(main.get("temp\_min")));

max.setText(String.*valueOf*(main.get("temp\_max")));

conn.disconnect();

**RESULT/ANALYSIS**

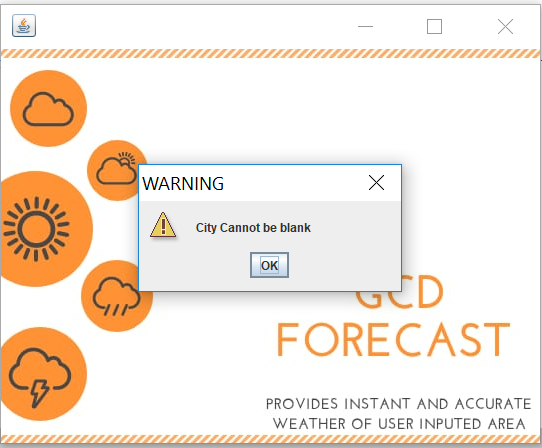
***Figure 1***

Here, the user’s input was bangalore. Once the input was submitted, the following results were displayed based on the current atmospheric conditions during that time.

***Figure 2***

In the figure 2, the user’s input was completely invalid. Hence, it displayed the appropriate error message to the user.



***Figure 3***

Figure 3 shows you a warning message issued by the application when the city field is submitted without the appropriate city name.

**CONCLUSION**

This desktop application provides the user’s the accurate weather forecast. GCD Forecast for java enabled desktop is expected to work successfully on all types of operating systems. The application is small and free which makes usage easy. User can selectively and interactively access the desired data, no matter where the person is.

**FUTURE UPDATES**

In the future updates there will be auto detecting location, showing weather forecast for the coming week, showing past weather trends and alert for weather changes.

**REFERENCES**

1. <https://stackoverflow.com/questions/1466240/how-to-set-an-image-as-a-background-for-frame-in-swing-gui-of-java>
2. <http://java-demos.blogspot.com/2012/09/setting-background-image-in-jframe.html?m=1>
3. <https://dzone.com/articles/how-to-parse-json-data-from-a-rest-api-using-simpl>
4. <https://www.youtube.com/watch?v=KdTsY3G_To0>
5. Calkins, J., M. R. Peroutka, R. K. Meiggs, and J. P. Lang, 1998: Creating NOAA

Weather Radio broadcast scrips using interactive forecast preparation. Preprints,

14th Int. Conf. on Interactive Information and Processing Systems for

Meteorology, Oceanography, and Hydrology, Phoenix, AZ, Amer. Meteor.

Soc., 355-358.

1. <https://www.javatpoint.com/java-swing>
2. Dickman, I. R., 2002: NWS Eastern Region Interactive Forecast Preparation

System (IFPS) Implementation.  Preprints, Interactive Symptom on the Advanced

Weather Interactive Processing System (AWIPS), Or-

lando, FL, Amer. Meteor. Soc., 28-32.

1. api.openweathermap.org/data/2.5/weather?q={city name}