Weather Forecast

MINOR PROJECT REPORT

By

CH.S.K .GOWTHAM (RA2211027010149)
S.SAI CHARANI (RA2211027010186)

Under the guidance of

Mrs. B. Yasotha

In partial fulfilment for the Course

Of

21CSC206P – ADVANCED OBJECT ORIENTED AND PROGRAMMING

in **DATA SCIENCE AND BUSINESS SYSTEMS**

FACULTY OF ENGINEERING AND TECHNOLOGY

SCHOOL OF COMPUTING

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR

NOVEMBER 2023

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that this minor project report for the course **22CSC206P ADVANCED OBJECT ORIENTED AND PROGRAMMING** entitled in **“WEATHER FORECAST”** is the bonafide work of **CH.S.K.GOWTHAM (RA2211027010149)** and **S.SAI CHARANI(RA2211027010186)** who carried out the work under my supervision

SIGNATURE

Mrs. B. Yasotha

Assistant Professor

Department of Data Science and Business System

SRM Institute of Science and Technology

Kattankulathur

ABSTRACT

Weather Forecast application that allows users to input a city name and retrieve weather data via the OpenWeatherMap API. The application features components for searching cities, selecting search results, and displaying current weather conditions. The core of the code is the WeatherData class, responsible for fetching both current weather and 5-day forecast data for specified cities, processed as JSON objects. The application collects and displays current weather information and forecasts for the next 5 days at 3-hour intervals. Developers can customize and extend this code for building weather forecast applications according to their requirements, making it a versatile tool for weather-related projects.

ACKNOWLEDGEMENT

We express our heartfelt thanks to our honorable **Vice Chancellor Dr. C. Muthamizhchelvan**, for being the beacon in all our endeavors.

We would like to express my warmth of gratitude to our **Registrar Dr. S. Ponnusamy,** for his encouragement.

We express our profound gratitude to our **Dean (College of Engineering and Technology) Dr. T. V. Gopal,** for bringing out novelty in all executions.

We would like to express my heartfelt thanks to Chairperson, School of Computing **Dr. Revathi Venkataraman,** for imparting confidence to complete my course project

We wish to express my sincere thanks to **Course Audit Professors Dr. Vadivu. G, Professor, Department of Data Science and Business System Course Coordinator** for their constant encouragement and support.

We are highly thankful to my Course project Faculty **Mrs.B.Yasotha , Assistant Professor**,**Department of Data Science and Business System** for his/her assistance, timely suggestion and guidance throughout the duration of this course project.

We extend my gratitude to our **HoD Dr. M Lakshmi, Professor, Department of Data Science and Business Systems** and my Departmental colleagues for their Support.

Finally, we thank our parents and friends near and dear ones who directly and indirectly contributed to the successful completion of our project. Above all, I thank the almighty for showering his blessings on me to complete my Course project.

TABLE OF CONTENTS

1.INTRODUCTION

This project is a sophisticated Weather Forecast application designed to provide users with up-to-the-minute weather information based on their city preferences. Leveraging the OpenWeatherMap API, this application offers a user-friendly experience, allowing individuals to effortlessly input the name of a city and receive detailed weather data. Its intuitive interface encompasses key components for city searches, result selection, and the display of current weather conditions, ensuring a seamless and accessible user experience

At the core of this application lies the WeatherData class, a robust and efficient module responsible for fetching both current weather data and five-day forecasts for user-specified cities. It processes this data as JSON objects, enabling the collection and presentation of comprehensive weather information.

1.1.MOTIVATION

The motivation behind this project is to create a user-friendly and versatile Weather Forecast application that empowers users with real-time weather data. Weather information is essential for daily life, from planning outdoor activities to making informed travel decisions. By developing this application, we aim to provide individuals with a straightforward tool to access accurate and timely weather forecasts based on their city of interest. This project seeks to bridge the gap between users and meteorological data, making it more accessible and comprehensible, ultimately enhancing their overall experience and enabling them to plan their activities with greater confidence.

1.2. OBJECTIVE

The main objective of this project is to develop a comprehensive Weather Forecast application that offers users the ability to retrieve detailed weather data for their desired cities quickly and effectively. This application's core functionality includes city search, result selection, and the presentation of current weather conditions and 5-day forecasts. By providing an intuitive and accessible interface, our goal is to give users easy access to weather information, enabling them to plan their activities and make informed decisions. Furthermore, this project serves as a flexible foundation that developers can expand upon and customize to meet their specific weather-related application requirements, promoting adaptability and extensibility in the development of weather forecast applications.

1.3. PROBLEM STATEMENT

The problem statement addressed by this code is the need for a weather forecast application that enables users to obtain current weather information for specific cities through the OpenWeatherMap API. This application aims to simplify the process of accessing weather data by providing a user-friendly interface for city search and result selection. The code's core challenge is to efficiently collect and display accurate weather data, including temperature, humidity, and forecasts at 3-hour intervals, enhancing user convenience and decision-making based on weather conditions. The project also tackles the broader issue of code adaptability, encouraging developers to create customized weather forecast applications to suit various weather-related project requirements.

1.4. CHALLENEGES

The primary challenges of this project lie in effectively connecting with the OpenWeatherMap API, retrieving and processing weather data, and presenting it coherently. Ensuring a reliable connection to the API while handling potential errors and data inconsistencies is crucial. Additionally, efficiently searching for cities based on user input and matching them to unique IDs poses a challenge.

The code must also organize and display the retrieved information accurately. This involves parsing and structuring data for both current weather conditions and 5-day forecasts in a user-friendly format. Making the code adaptable and extensible for developers looking to customize it adds the challenge of writing clean, well-documented, and modular code. In summary, this project's challenges encompass data acquisition, processing, presentation, and code design to create a functional and flexible weather forecast application.

2.LITERATURE SURVEY

1. **"Meteorology Today"** by C. Donald Ahrens and Robert Henson - This book provides a comprehensive introduction to meteorology, which is essential for understanding weather forecasting principles.

2. **"Swing (2nd Edition)"** by Matthew Robinson and Pavel Vorobiev - Swing is used for creating the application's GUI, and this book provides in-depth knowledge of Swing components and design.

3. "**Weather Analysis and Forecasting:** Applying Satellite Water Vapor Imagery and Potential Vorticity Analysis" by Christo Georgiev - Focuses on weather analysis and forecasting, which aligns with the project's objectives.

4. **"OpenWeatherMap Cookbook"** by Matthias Einbrodt - While not a book per se, this online resource provides guidance on using the OpenWeatherMap API effectively.

3.REQUIREMENTS ANALYSIS

3.1. HARDWARE REQUIREMENTS

Processor:

• Minimum: Intel Core i3 or equivalent

• Recommended: Intel Core i5 or equivalent

RAM (Random Access Memory):

• Minimum: 4GB

• Recommended: 8GB or higher

Storage:

• Minimum: 128GB SSD or HDD

• Recommended: 256GB SSD or higher for better performance

Operating System:

• Windows 10 or later

• macOS 10.14 Mojave or later

• Ubuntu 18.04 LTS or late

3.2. SOFTWARE REQUIREMENTS

1.Integrated Development Environment (IDE):

A Java-compatible IDE such as Eclipse, IntelliJ IDEA, or NetBeans is needed for coding and building the application.

2.Java Development Kit (JDK):

The latest version of JDK should be installed to compile and run the Java code. This project appears to be written in Java, so a JDK is essential.

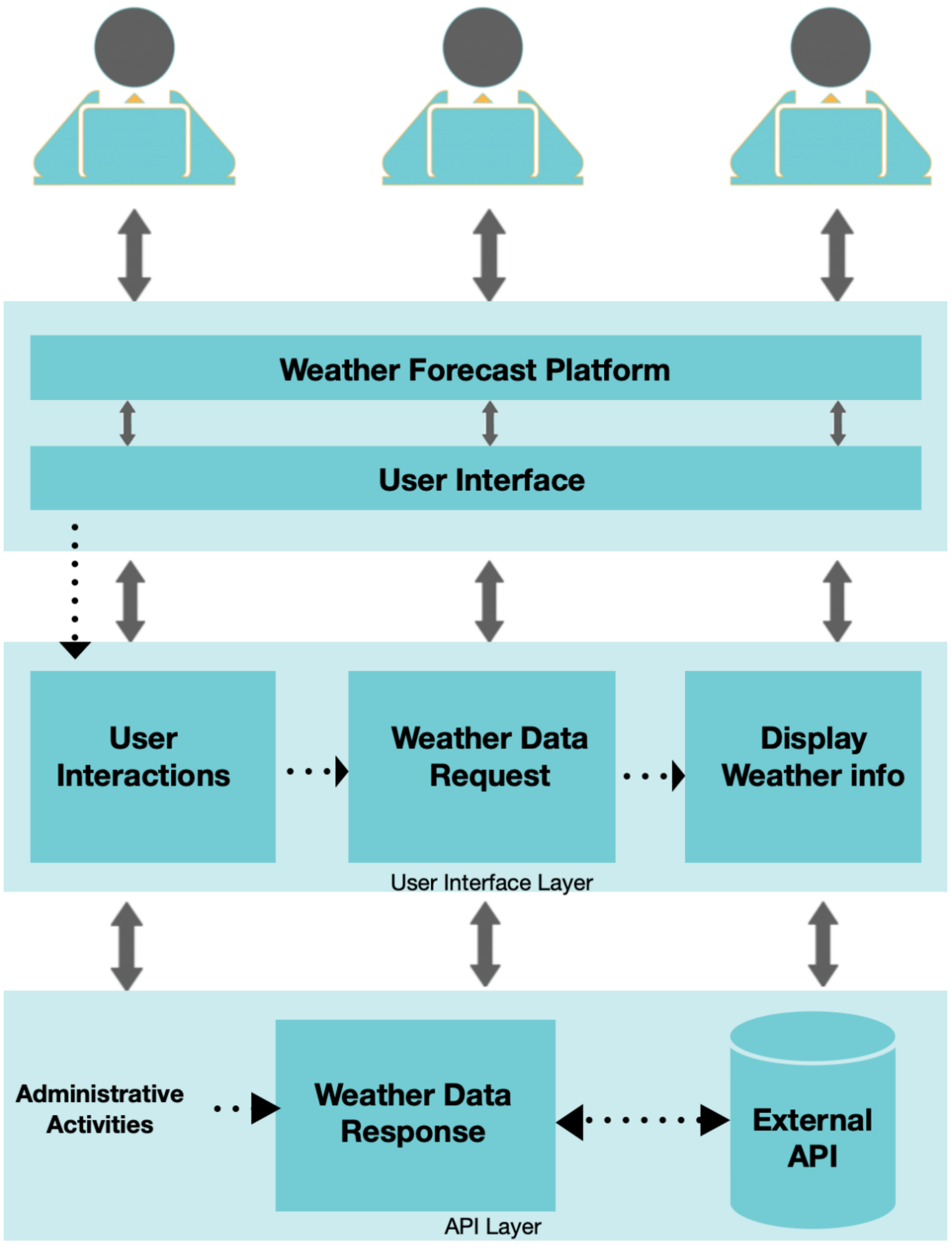
3.Java Libraries:

Libraries for handling JSON data, such as the Java API for JSON Processing (JSON-P), are required for processing weather data.

4.Version Control: A version control system like Git for tracking changes and collaborating on the project.

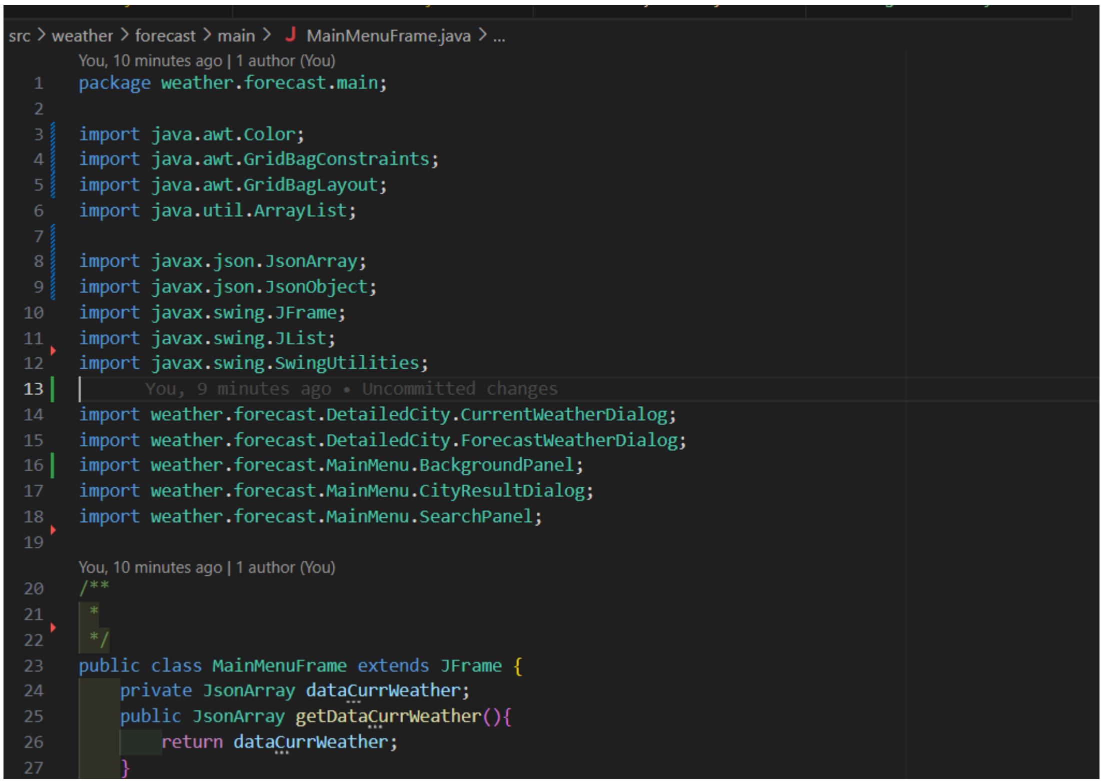
5.API Key: Developers need an API key from OpenWeatherMap to access weather data. This key should be stored securely and not hard-coded in the project**.**

4.ARCHITECTURE AND DESIGN



5.IMPLEMENTATION

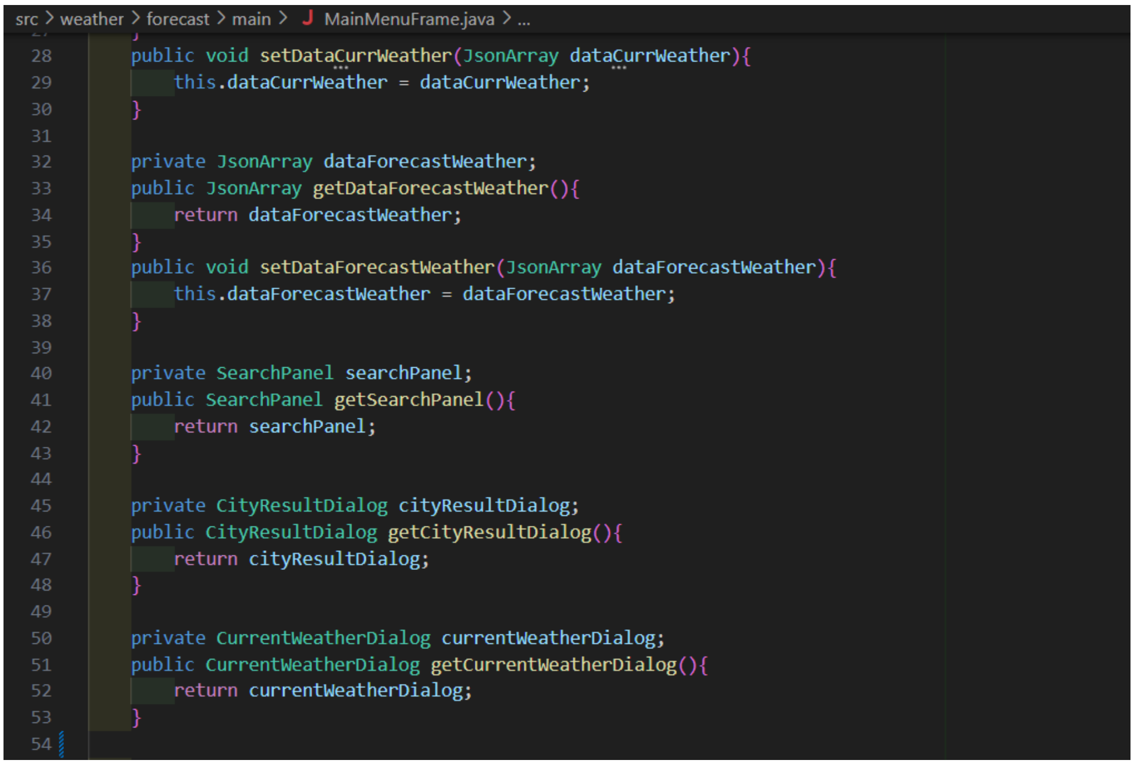
CODE SNIPPETS:



src > weather > forecast > main > MainMenuFrame.java > ...

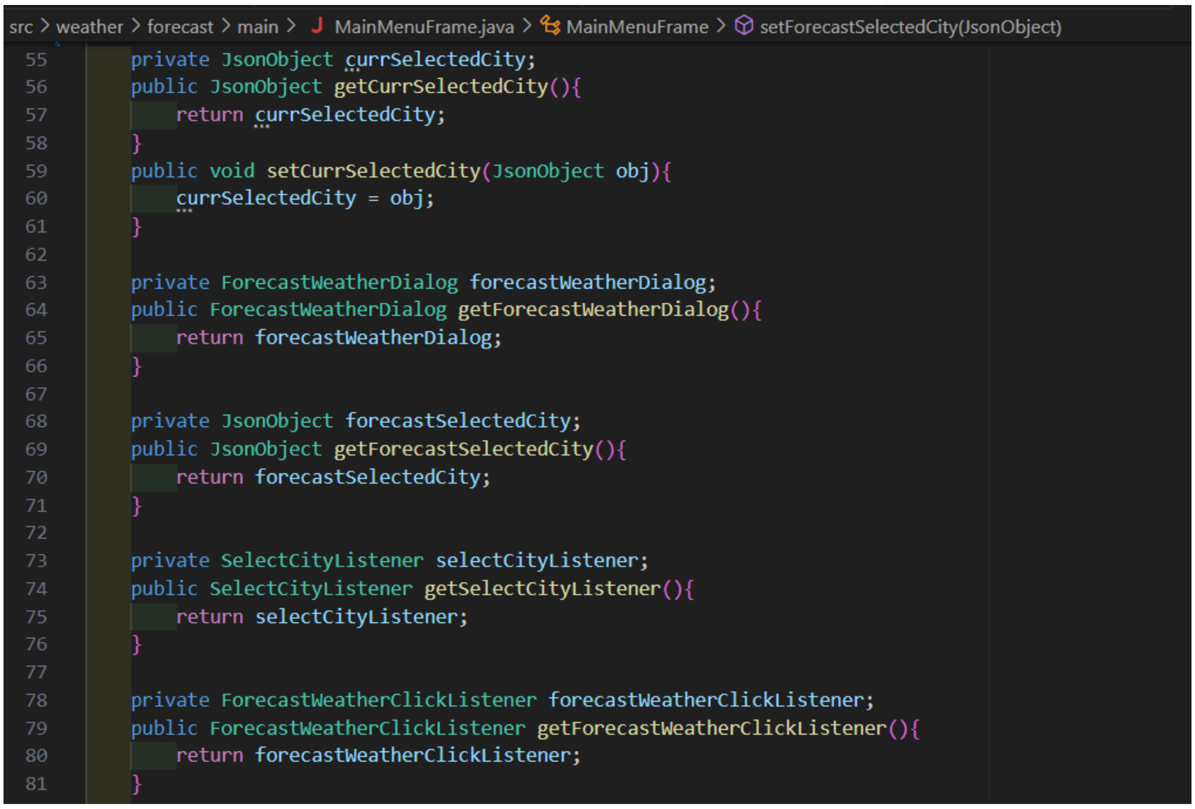
You, 10 minutes ago | 1 author (You)

```java
package weather.forecast.main;

import java.awt.Color;
import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;
import java.util.ArrayList;

import javax.json.JsonArray;
import javax.json.JsonObject;
import javax.swing.JFrame;
import javax.swing.JList;
import javax.swing.SwingUtilities;
|       You, 9 minutes ago • Uncommitted changes
import weather.forecast.DetailedCity.CurrentWeatherDialog;
import weather.forecast.DetailedCity.ForecastWeatherDialog;
import weather.forecast.MainMenu.BackgroundPanel;
import weather.forecast.MainMenu.CityResultDialog;
import weather.forecast.MainMenu.SearchPanel;

You, 10 minutes ago | 1 author (You)
/**
 *
 */
public class MainMenuFrame extends JFrame {
    private JsonArray dataCurrWeather;
    public JsonArray getDataCurrWeather(){
        return dataCurrWeather;
    }
```

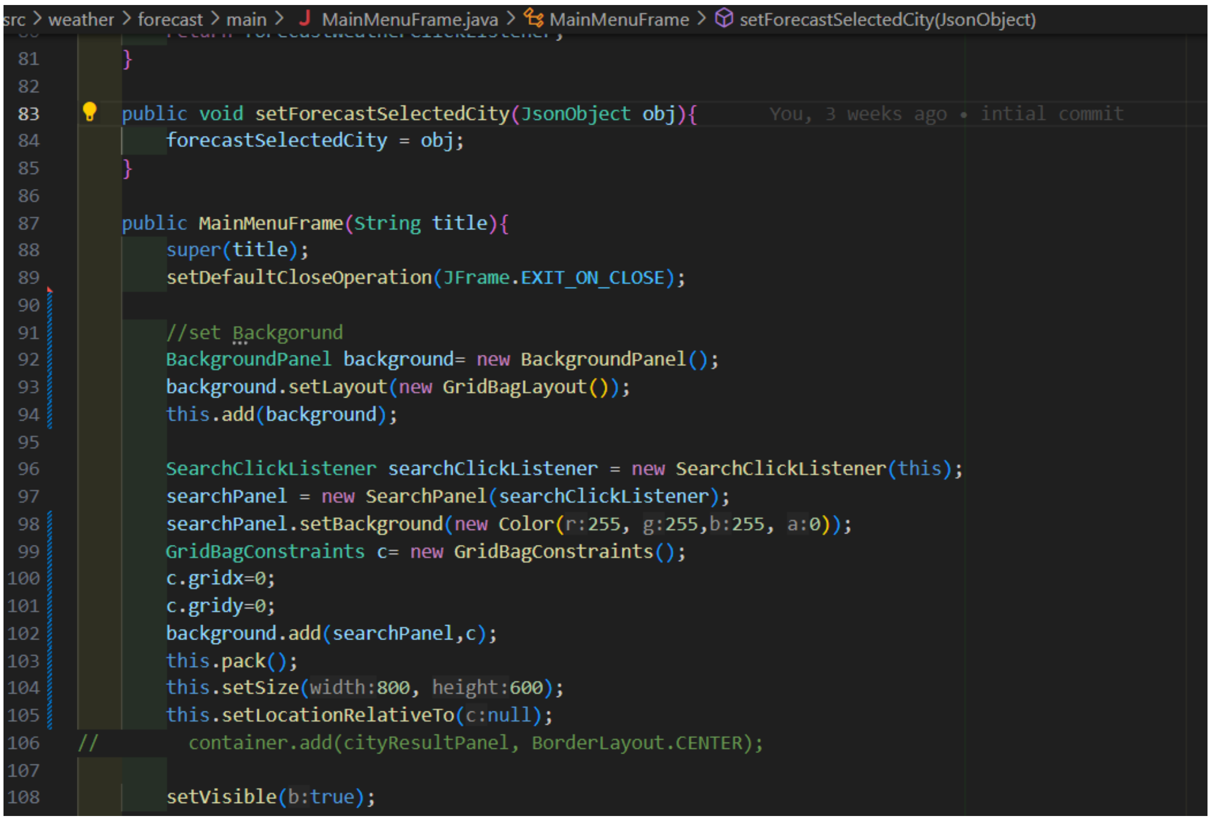


src > weather > forecast > main > MainMenuFrame.java > ...

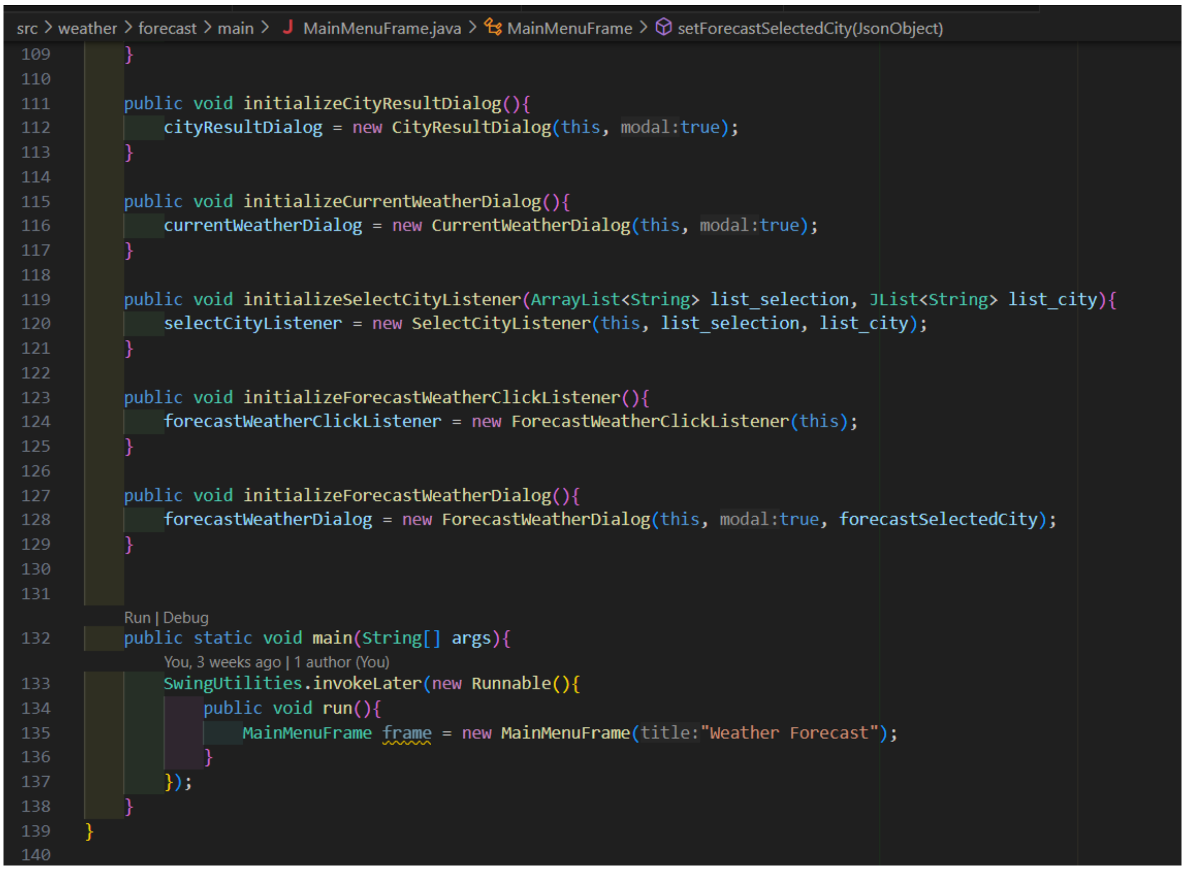
```java
    public void setDataCurrWeather(JsonArray dataCurrWeather){
        this.dataCurrWeather = dataCurrWeather;
    }

    private JsonArray dataForecastWeather;
    public JsonArray getDataForecastWeather(){
        return dataForecastWeather;
    }
    public void setDataForecastWeather(JsonArray dataForecastWeather){
        this.dataForecastWeather = dataForecastWeather;
    }

    private SearchPanel searchPanel;
    public SearchPanel getSearchPanel(){
        return searchPanel;
    }

    private CityResultDialog cityResultDialog;
    public CityResultDialog getCityResultDialog(){
        return cityResultDialog;
    }

    private CurrentWeatherDialog currentWeatherDialog;
    public CurrentWeatherDialog getCurrentWeatherDialog(){
        return currentWeatherDialog;
    }

```


src > weather > forecast > main > MainMenuFrame.java > MainMenuFrame > setForecastSelectedCity(JsonObject)

```
    private JsonObject currSelectedCity;
    public JsonObject getCurrSelectedCity(){
        return currSelectedCity;
    }
    public void setCurrSelectedCity(JsonObject obj){
        currSelectedCity = obj;
    }

    private ForecastWeatherDialog forecastWeatherDialog;
    public ForecastWeatherDialog getForecastWeatherDialog(){
        return forecastWeatherDialog;
    }

    private JsonObject forecastSelectedCity;
    public JsonObject getForecastSelectedCity(){
        return forecastSelectedCity;
    }

    private SelectCityListener selectCityListener;
    public SelectCityListener getSelectCityListener(){
        return selectCityListener;
    }

    private ForecastWeatherClickListener forecastWeatherClickListener;
    public ForecastWeatherClickListener getForecastWeatherClickListener(){
        return forecastWeatherClickListener;
    }
```

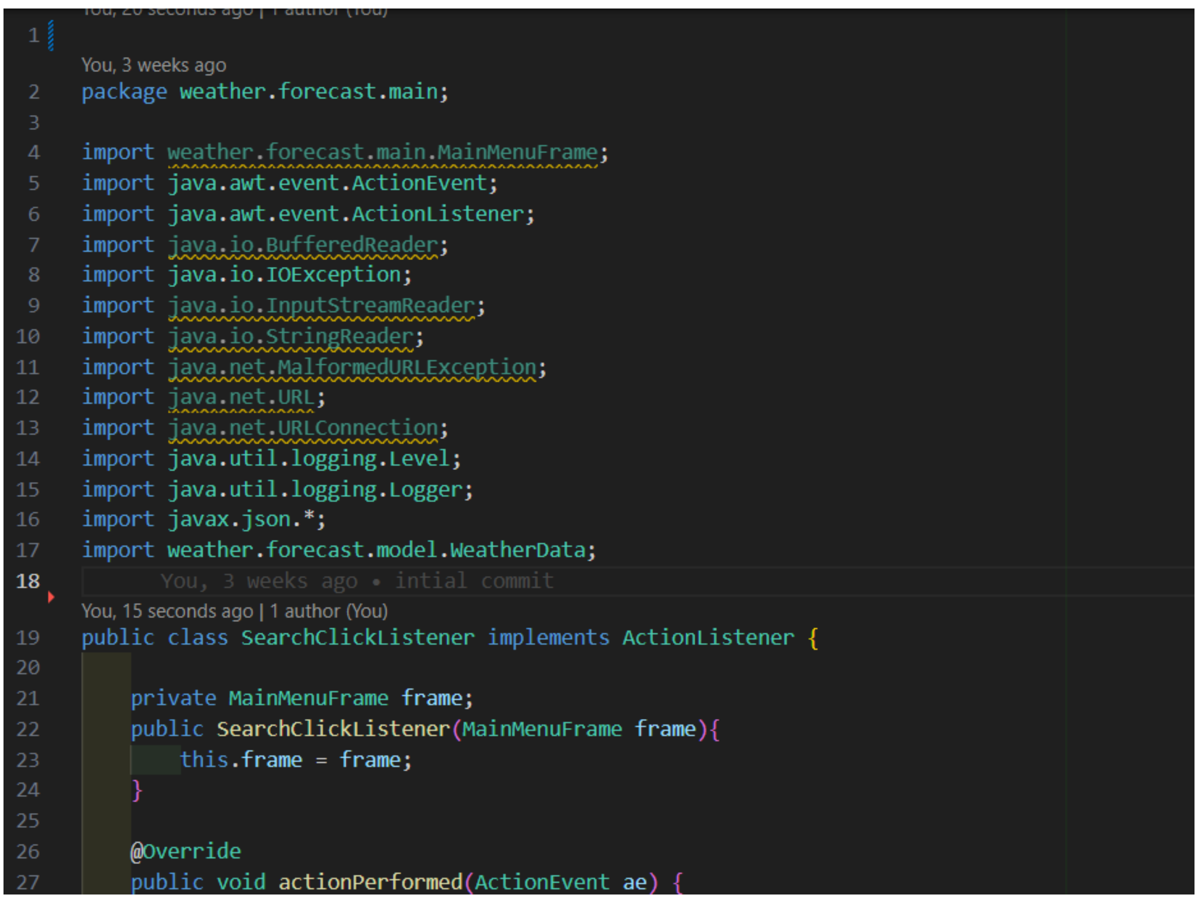

src > weather > forecast > main > MainMenuFrame.java > MainMenuFrame > setForecastSelectedCity(JsonObject)

```
    }

    public void setForecastSelectedCity(JsonObject obj){        You, 3 weeks ago • intial commit
        forecastSelectedCity = obj;
    }

    public MainMenuFrame(String title){
        super(title);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        //set Backgorund
        BackgroundPanel background= new BackgroundPanel();
        background.setLayout(new GridBagLayout());
        this.add(background);

        SearchClickListener searchClickListener = new SearchClickListener(this);
        searchPanel = new SearchPanel(searchClickListener);
        searchPanel.setBackground(new Color(r:255, g:255,b:255, a:0));
        GridBagConstraints c= new GridBagConstraints();
        c.gridx=0;
        c.gridy=0;
        background.add(searchPanel,c);
        this.pack();
        this.setSize(width:800, height:600);
        this.setLocationRelativeTo(c:null);
//          container.add(cityResultPanel, BorderLayout.CENTER);

        setVisible(b:true);
```

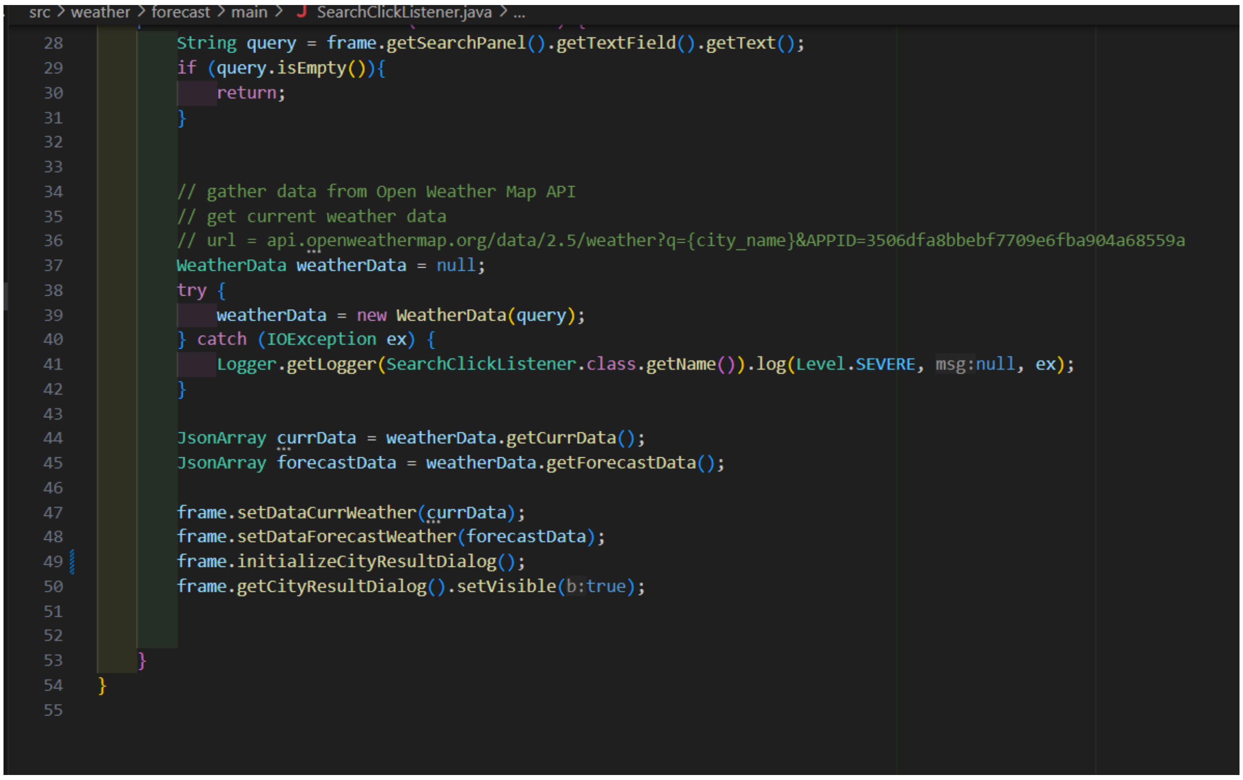


src > weather > forecast > main > MainMenuFrame.java > MainMenuFrame > setForecastSelectedCity(JsonObject)

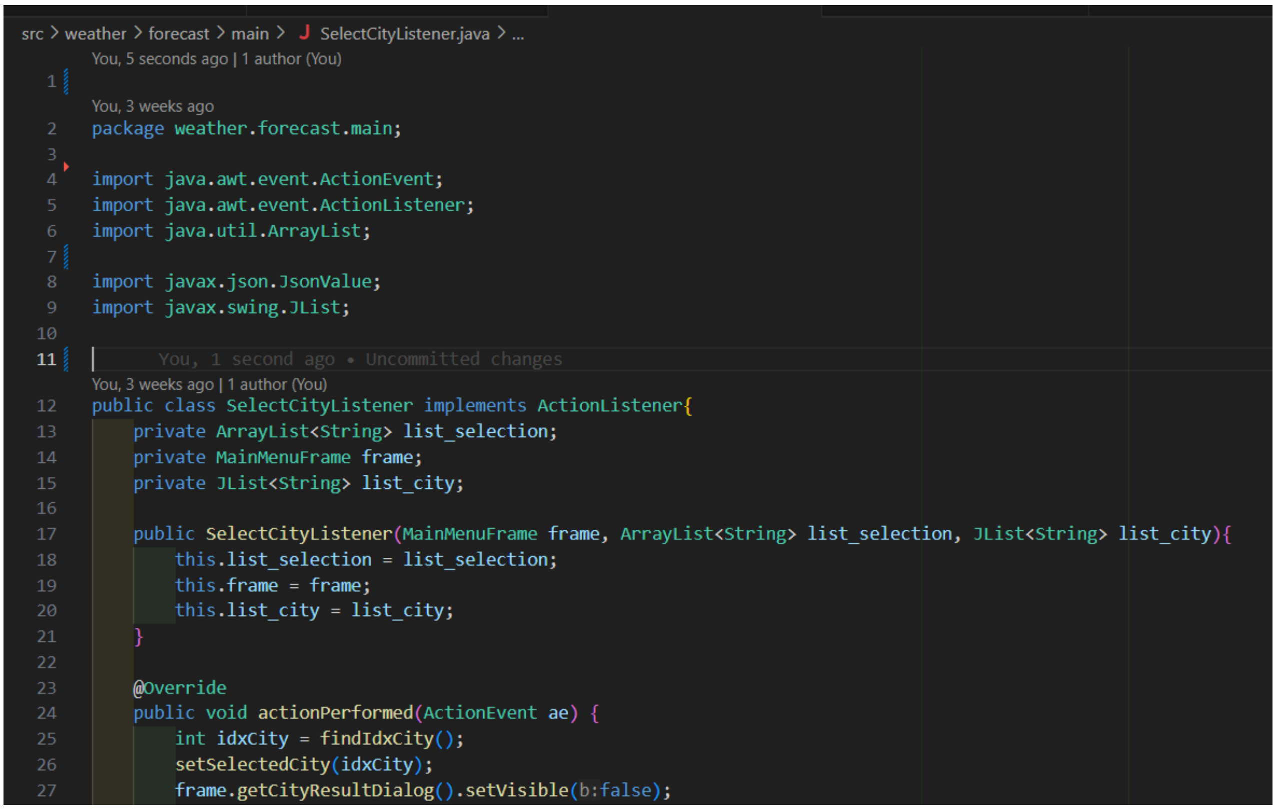
```java
    }

    public void initializeCityResultDialog(){
        cityResultDialog = new CityResultDialog(this, modal:true);
    }

    public void initializeCurrentWeatherDialog(){
        currentWeatherDialog = new CurrentWeatherDialog(this, modal:true);
    }

    public void initializeSelectCityListener(ArrayList<String> list_selection, JList<String> list_city){
        selectCityListener = new SelectCityListener(this, list_selection, list_city);
    }

    public void initializeForecastWeatherClickListener(){
        forecastWeatherClickListener = new ForecastWeatherClickListener(this);
    }

    public void initializeForecastWeatherDialog(){
        forecastWeatherDialog = new ForecastWeatherDialog(this, modal:true, forecastSelectedCity);
    }


    Run | Debug
    public static void main(String[] args){
        You, 3 weeks ago | 1 author (You)
        SwingUtilities.invokeLater(new Runnable(){
            public void run(){
                MainMenuFrame frame = new MainMenuFrame(title:"Weather Forecast");
            }
        });
    }
}
```



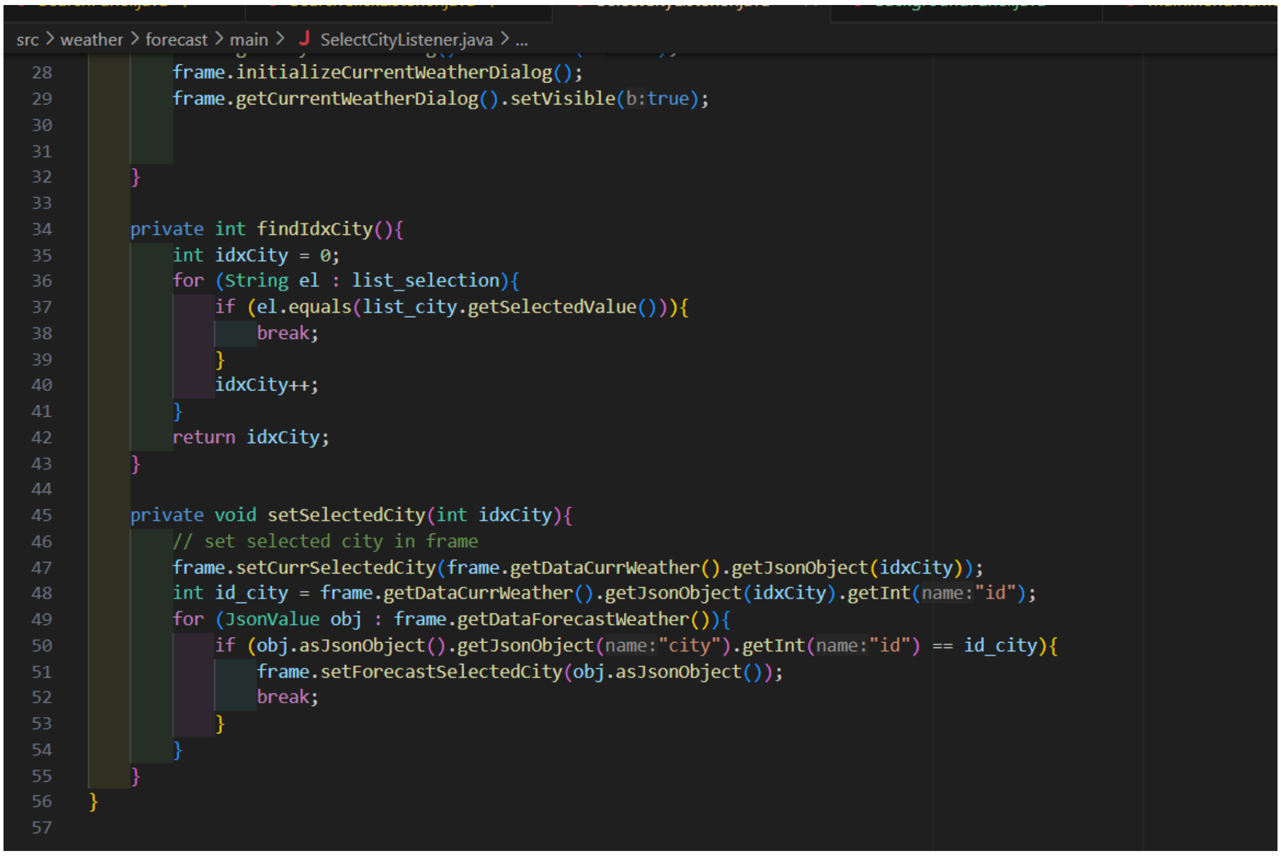
```java
You, 20 seconds ago | 1 author (You)

You, 3 weeks ago
package weather.forecast.main;

import weather.forecast.main.MainMenuFrame;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.StringReader;
import java.net.MalformedURLException;
import java.net.URL;
import java.net.URLConnection;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.json.*;
import weather.forecast.model.WeatherData;
        You, 3 weeks ago • intial commit
You, 15 seconds ago | 1 author (You)
public class SearchClickListener implements ActionListener {

    private MainMenuFrame frame;
    public SearchClickListener(MainMenuFrame frame){
        this.frame = frame;
    }

    @Override
    public void actionPerformed(ActionEvent ae) {
```


src > weather > forecast > main > J SearchClickListener.java > ...

```java
        String query = frame.getSearchPanel().getTextField().getText();
        if (query.isEmpty()){
            return;
        }


        // gather data from Open Weather Map API
        // get current weather data
        // url = api.openweathermap.org/data/2.5/weather?q={city_name}&APPID=3506dfa8bbebf7709e6fba904a68559a
        WeatherData weatherData = null;
        try {
            weatherData = new WeatherData(query);
        } catch (IOException ex) {
            Logger.getLogger(SearchClickListener.class.getName()).log(Level.SEVERE, msg:null, ex);
        }

        JsonArray currData = weatherData.getCurrData();
        JsonArray forecastData = weatherData.getForecastData();

        frame.setDataCurrWeather(currData);
        frame.setDataForecastWeather(forecastData);
        frame.initializeCityResultDialog();
        frame.getCityResultDialog().setVisible(b:true);


    }
}

```

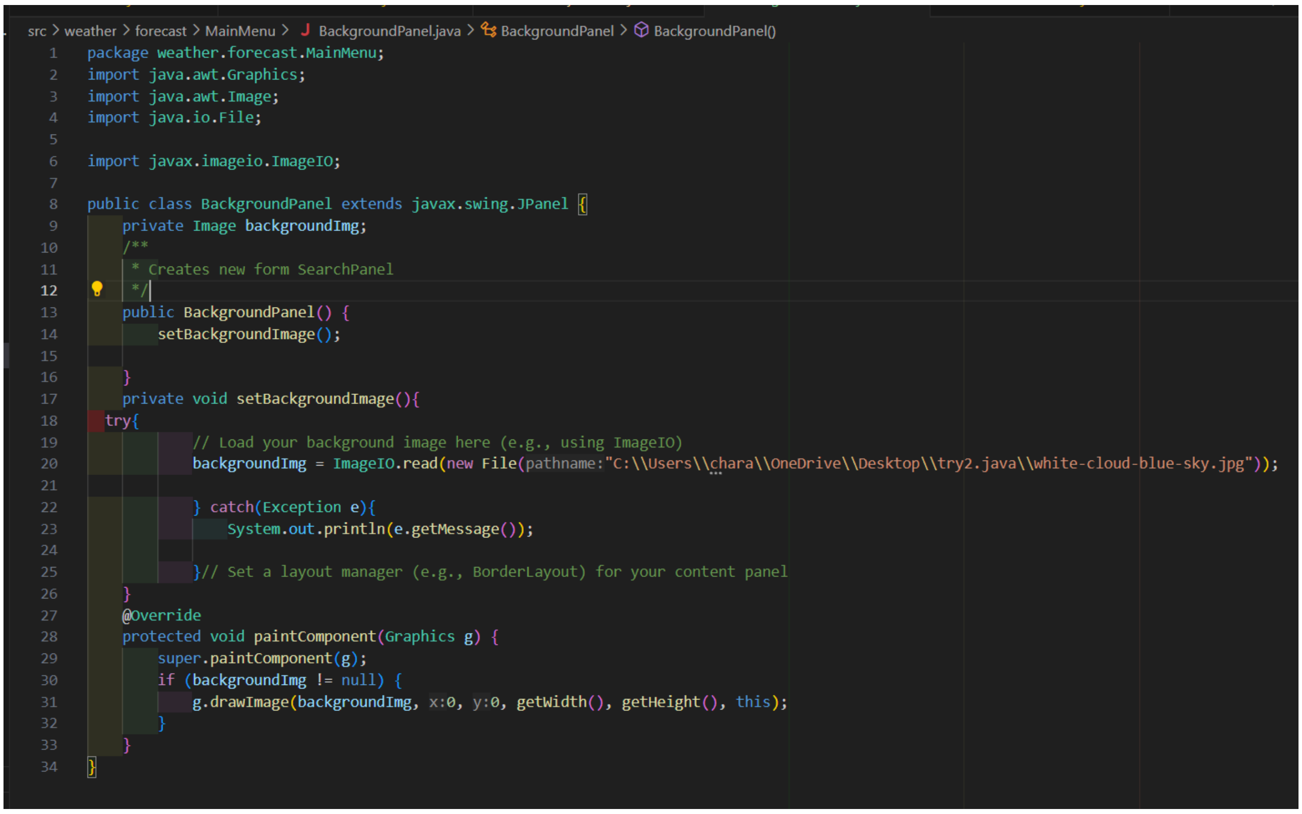

src > weather > forecast > main > J SelectCityListener.java > ...

```java
You, 5 seconds ago | 1 author (You)

You, 3 weeks ago
package weather.forecast.main;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.ArrayList;

import javax.json.JsonValue;
import javax.swing.JList;

|       You, 1 second ago • Uncommitted changes
You, 3 weeks ago | 1 author (You)
public class SelectCityListener implements ActionListener{
    private ArrayList<String> list_selection;
    private MainMenuFrame frame;
    private JList<String> list_city;

    public SelectCityListener(MainMenuFrame frame, ArrayList<String> list_selection, JList<String> list_city){
        this.list_selection = list_selection;
        this.frame = frame;
        this.list_city = list_city;
    }

    @Override
    public void actionPerformed(ActionEvent ae) {
        int idxCity = findIdxCity();
        setSelectedCity(idxCity);
        frame.getCityResultDialog().setVisible(b:false);
```



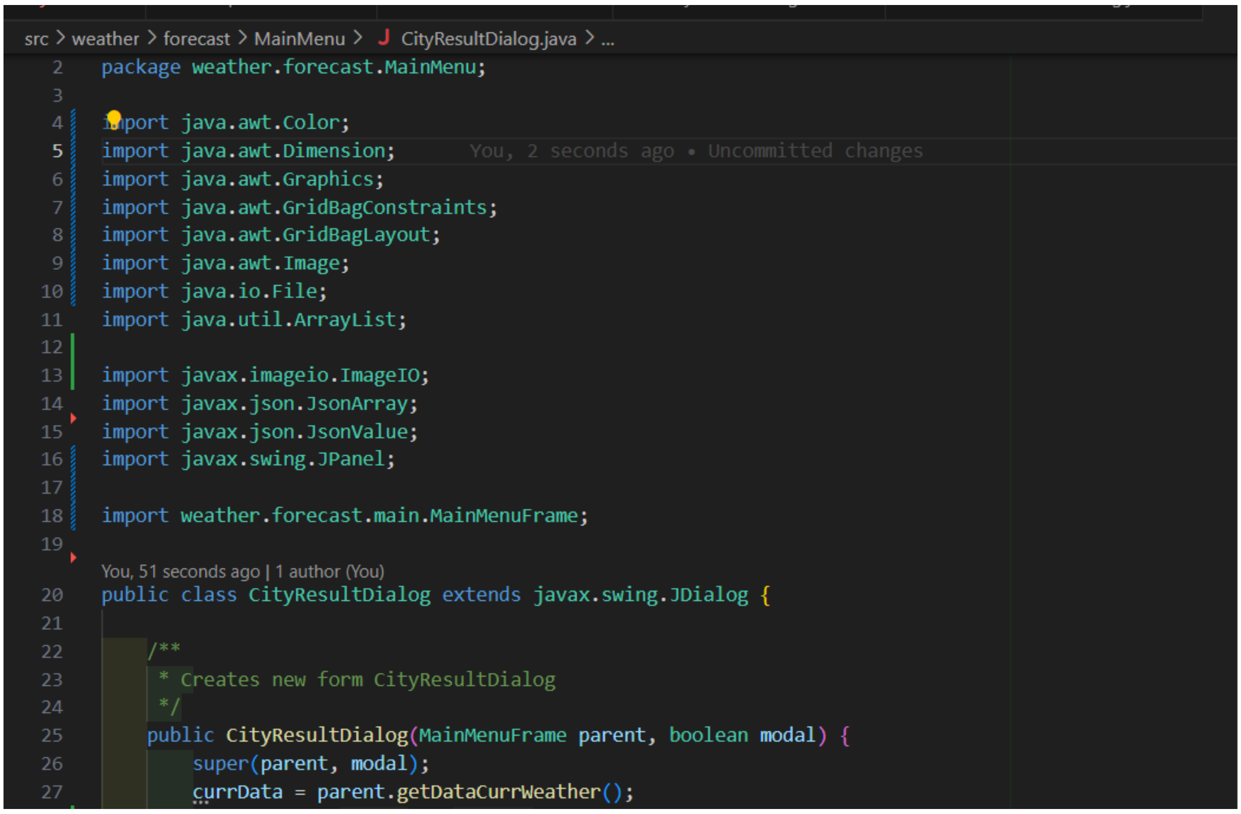
src > weather > forecast > main > SelectCityListener.java > ...

```java
        frame.initializeCurrentWeatherDialog();
        frame.getCurrentWeatherDialog().setVisible(b:true);


    }

    private int findIdxCity(){
        int idxCity = 0;
        for (String el : list_selection){
            if (el.equals(list_city.getSelectedValue())){
                break;
            }
            idxCity++;
        }
        return idxCity;
    }

    private void setSelectedCity(int idxCity){
        // set selected city in frame
        frame.setCurrSelectedCity(frame.getDataCurrWeather().getJsonObject(idxCity));
        int id_city = frame.getDataCurrWeather().getJsonObject(idxCity).getInt(name:"id");
        for (JsonValue obj : frame.getDataForecastWeather()){
            if (obj.asJsonObject().getJsonObject(name:"city").getInt(name:"id") == id_city){
                frame.setForecastSelectedCity(obj.asJsonObject());
                break;
            }
        }
    }
}
```

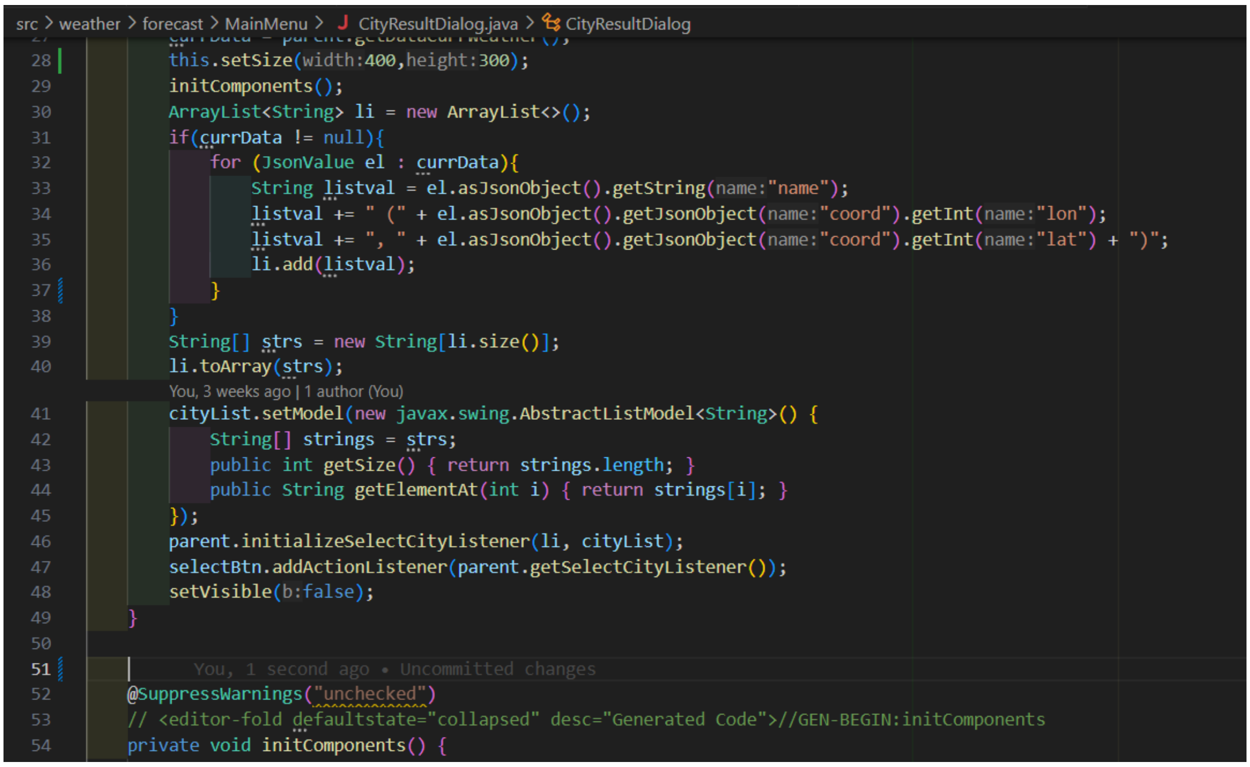


src > weather > forecast > MainMenu > BackgroundPanel.java > BackgroundPanel > BackgroundPanel()

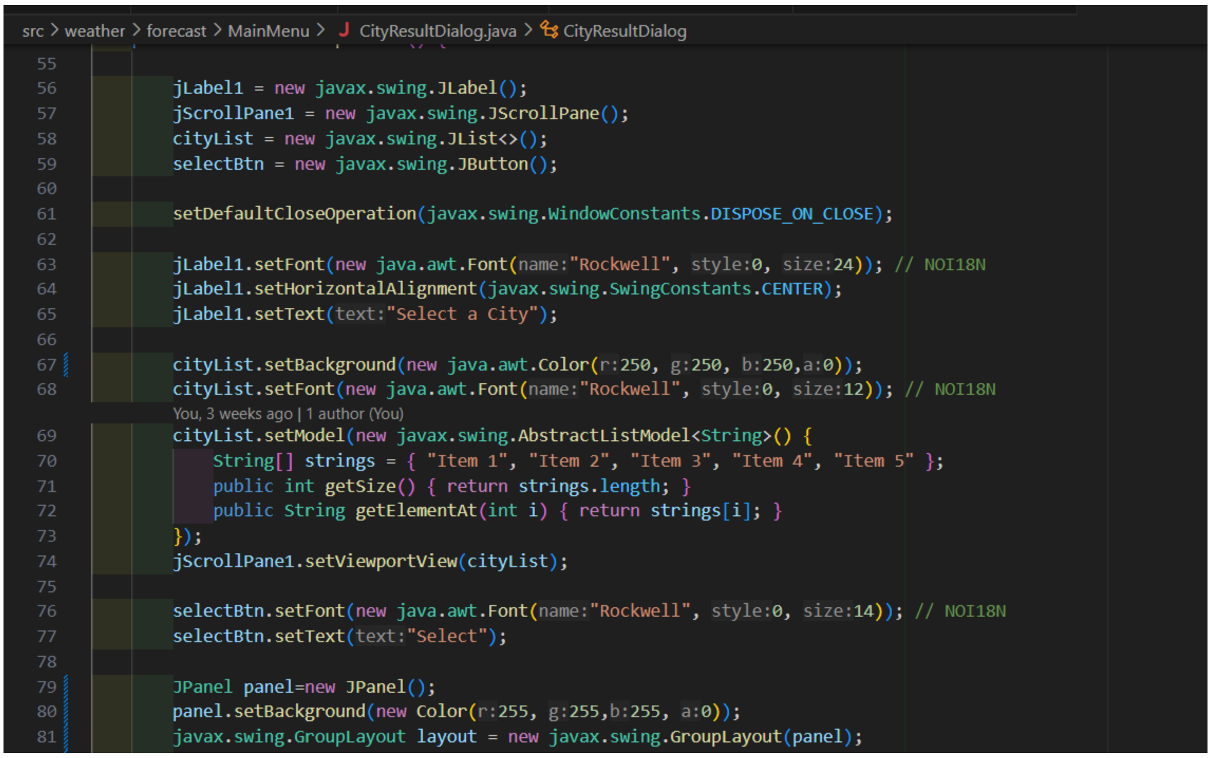
```java
package weather.forecast.MainMenu;
import java.awt.Graphics;
import java.awt.Image;
import java.io.File;

import javax.imageio.ImageIO;

public class BackgroundPanel extends javax.swing.JPanel {
    private Image backgroundImg;
    /**
     * Creates new form SearchPanel
     */
    public BackgroundPanel() {
        setBackgroundImage();

    }
    private void setBackgroundImage(){
  try{
            // Load your background image here (e.g., using ImageIO)
            backgroundImg = ImageIO.read(new File(pathname:"C:\\Users\\chara\\OneDrive\\Desktop\\try2.java\\white-cloud-blue-sky.jpg"));

            } catch(Exception e){
                System.out.println(e.getMessage());

            }// Set a layout manager (e.g., BorderLayout) for your content panel
    }
    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
        if (backgroundImg != null) {
            g.drawImage(backgroundImg, x:0, y:0, getWidth(), getHeight(), this);
        }
    }
}
```


src > weather > forecast > MainMenu > CityResultDialog.java > ...

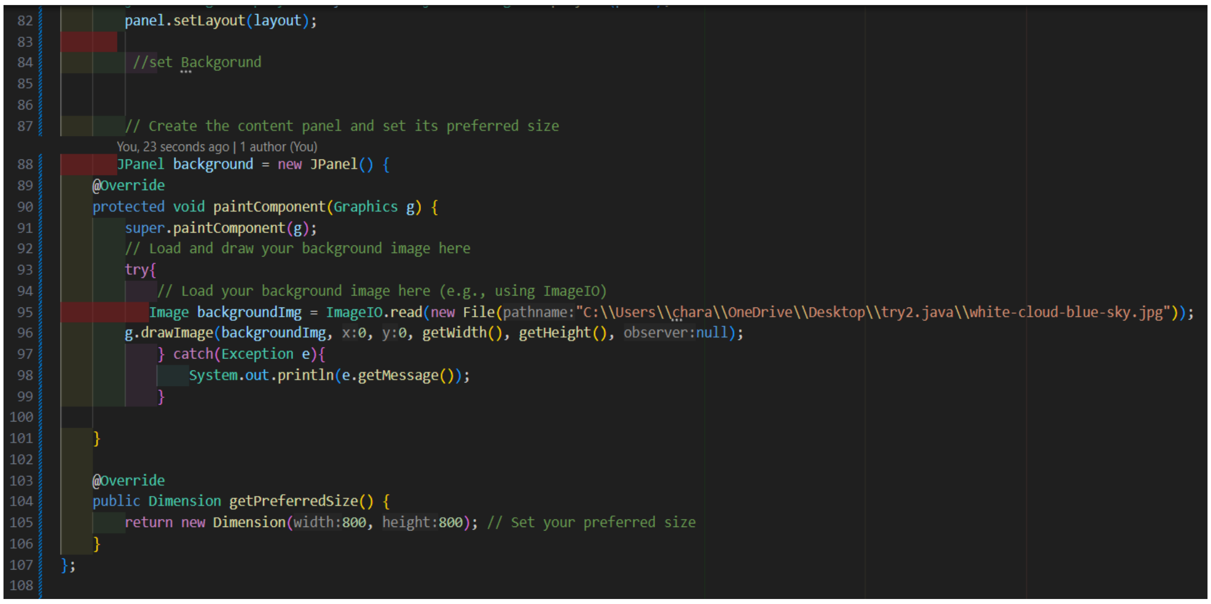
```java
package weather.forecast.MainMenu;

import java.awt.Color;
import java.awt.Dimension;          You, 2 seconds ago • Uncommitted changes
import java.awt.Graphics;
import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;
import java.awt.Image;
import java.io.File;
import java.util.ArrayList;

import javax.imageio.ImageIO;
import javax.json.JsonArray;
import javax.json.JsonValue;
import javax.swing.JPanel;

import weather.forecast.main.MainMenuFrame;

You, 51 seconds ago | 1 author (You)
public class CityResultDialog extends javax.swing.JDialog {

    /**
     * Creates new form CityResultDialog
     */
    public CityResultDialog(MainMenuFrame parent, boolean modal) {
        super(parent, modal);
        currData = parent.getDataCurrWeather();
```


src > weather > forecast > MainMenu > CityResultDialog.java > CityResultDialog

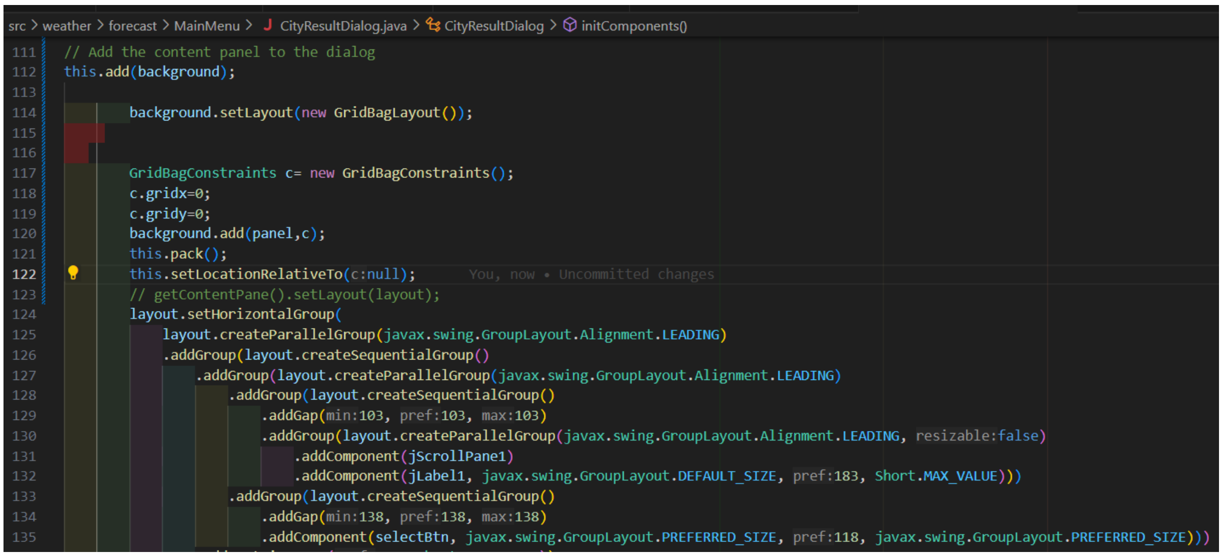
```java
        this.setSize(width:400,height:300);
        initComponents();
        ArrayList<String> li = new ArrayList<>();
        if(currData != null){
            for (JsonValue el : currData){
                String listval = el.asJsonObject().getString(name:"name");
                listval += " (" + el.asJsonObject().getJsonObject(name:"coord").getInt(name:"lon");
                listval += ", " + el.asJsonObject().getJsonObject(name:"coord").getInt(name:"lat") + ")";
                li.add(listval);
            }
        }
        String[] strs = new String[li.size()];
        li.toArray(strs);
        You, 3 weeks ago | 1 author (You)
        cityList.setModel(new javax.swing.AbstractListModel<String>() {
            String[] strings = strs;
            public int getSize() { return strings.length; }
            public String getElementAt(int i) { return strings[i]; }
        });
        parent.initializeSelectCityListener(li, cityList);
        selectBtn.addActionListener(parent.getSelectCityListener());
        setVisible(b:false);
    }

    |       You, 1 second ago • Uncommitted changes
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-BEGIN:initComponents
    private void initComponents() {
```


src > weather > forecast > MainMenu > CityResultDialog.java > CityResultDialog

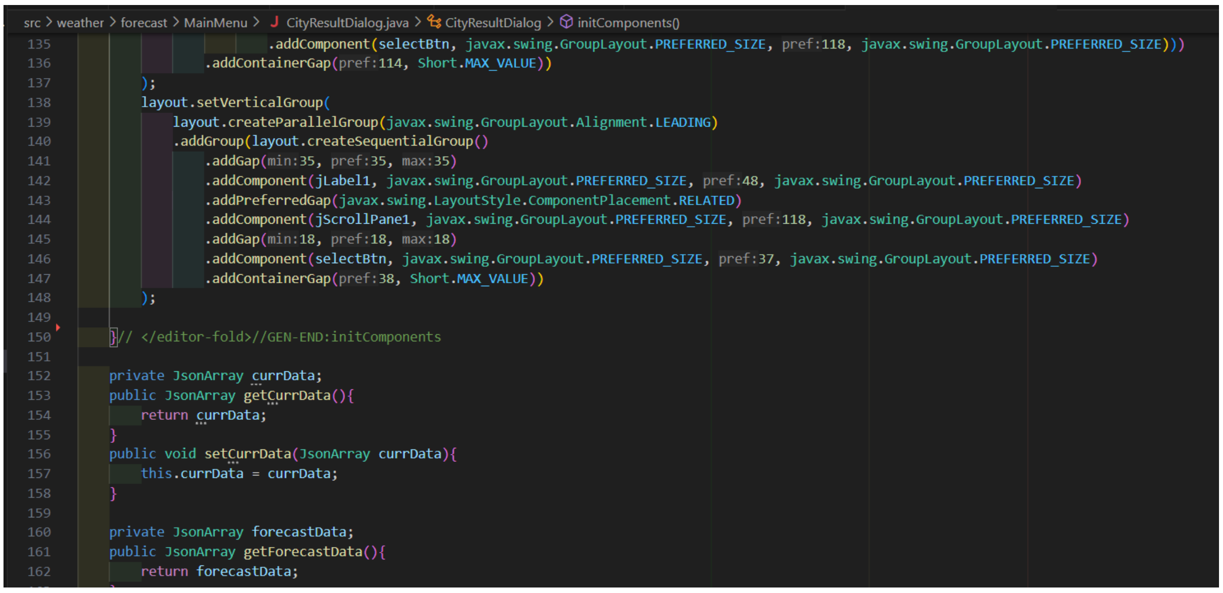
```java

        jLabel1 = new javax.swing.JLabel();
        jScrollPane1 = new javax.swing.JScrollPane();
        cityList = new javax.swing.JList<>();
        selectBtn = new javax.swing.JButton();

        setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);

        jLabel1.setFont(new java.awt.Font(name:"Rockwell", style:0, size:24)); // NOI18N
        jLabel1.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
        jLabel1.setText(text:"Select a City");

        cityList.setBackground(new java.awt.Color(r:250, g:250, b:250,a:0));
        cityList.setFont(new java.awt.Font(name:"Rockwell", style:0, size:12)); // NOI18N
        You, 3 weeks ago | 1 author (You)
        cityList.setModel(new javax.swing.AbstractListModel<String>() {
            String[] strings = { "Item 1", "Item 2", "Item 3", "Item 4", "Item 5" };
            public int getSize() { return strings.length; }
            public String getElementAt(int i) { return strings[i]; }
        });
        jScrollPane1.setViewportView(cityList);

        selectBtn.setFont(new java.awt.Font(name:"Rockwell", style:0, size:14)); // NOI18N
        selectBtn.setText(text:"Select");

        JPanel panel=new JPanel();
        panel.setBackground(new Color(r:255, g:255,b:255, a:0));
        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(panel);
```



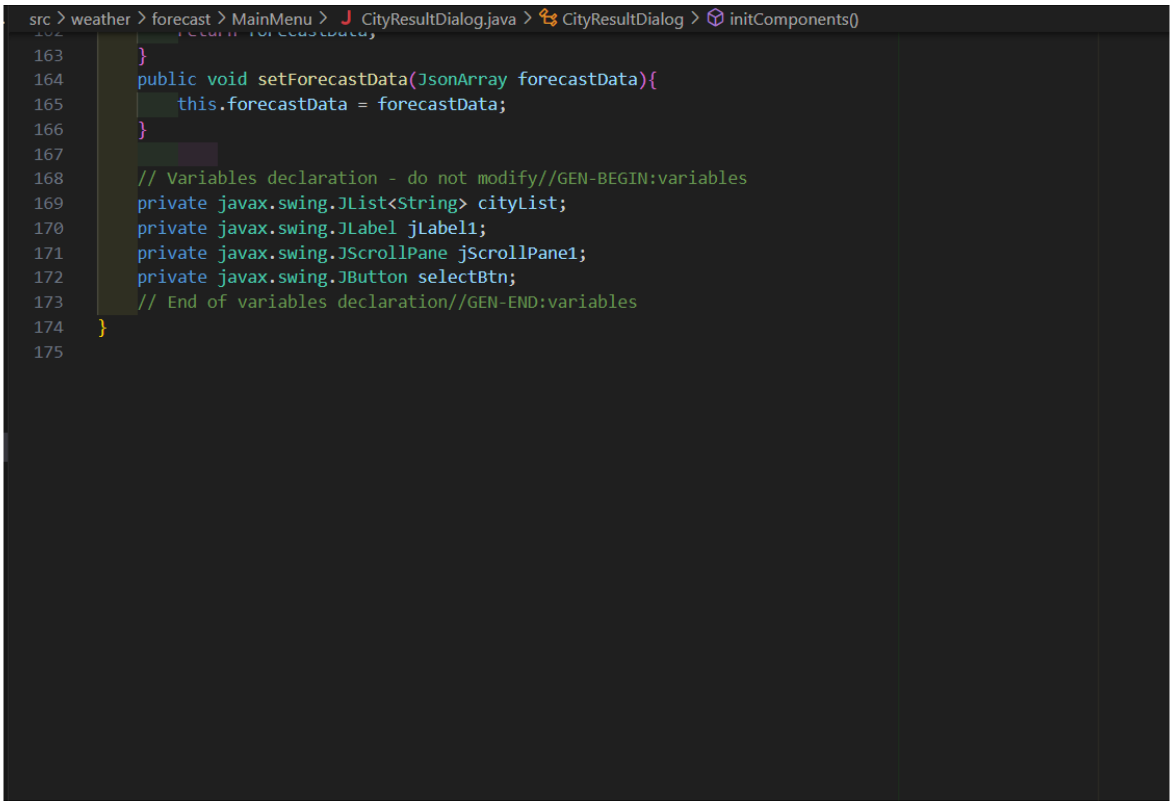
```java
        panel.setLayout(layout);

         //set Backgorund


        // Create the content panel and set its preferred size
      You, 23 seconds ago | 1 author (You)
      JPanel background = new JPanel() {
    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
        // Load and draw your background image here
        try{
            // Load your background image here (e.g., using ImageIO)
          Image backgroundImg = ImageIO.read(new File(pathname:"C:\\Users\\chara\\OneDrive\\Desktop\\try2.java\\white-cloud-blue-sky.jpg"));
        g.drawImage(backgroundImg, x:0, y:0, getWidth(), getHeight(), observer:null);
            } catch(Exception e){
                System.out.println(e.getMessage());
            }

    }

    @Override
    public Dimension getPreferredSize() {
        return new Dimension(width:800, height:800); // Set your preferred size
    }
};

```


src > weather > forecast > MainMenu > CityResultDialog.java > CityResultDialog > initComponents()

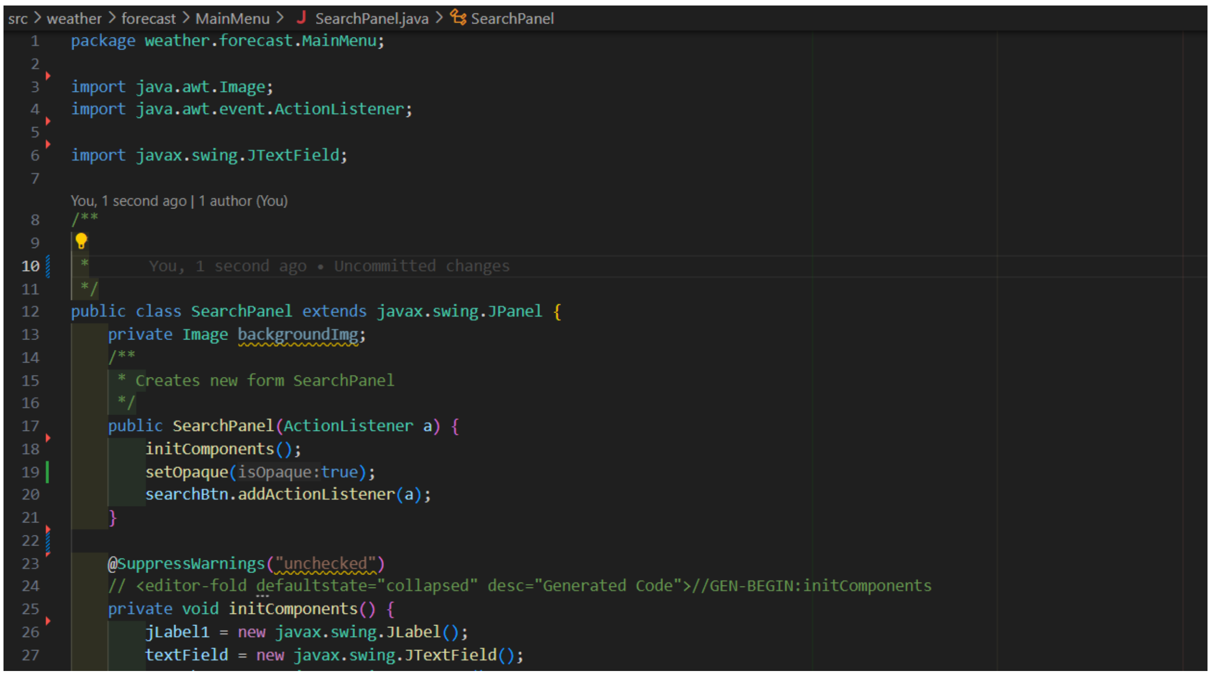
```java
// Add the content panel to the dialog
this.add(background);

        background.setLayout(new GridBagLayout());


        GridBagConstraints c= new GridBagConstraints();
        c.gridx=0;
        c.gridy=0;
        background.add(panel,c);
        this.pack();
        this.setLocationRelativeTo(c:null);        You, now • Uncommitted changes
        // getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(layout.createSequentialGroup()
                        .addGap(min:103, pref:103, max:103)
                        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, resizable:false)
                            .addComponent(jScrollPane1)
                            .addComponent(jLabel1, javax.swing.GroupLayout.DEFAULT_SIZE, pref:183, Short.MAX_VALUE)))
                    .addGroup(layout.createSequentialGroup()
                        .addGap(min:138, pref:138, max:138)
                        .addComponent(selectBtn, javax.swing.GroupLayout.PREFERRED_SIZE, pref:118, javax.swing.GroupLayout.PREFERRED_SIZE)))
```


src > weather > forecast > MainMenu > CityResultDialog.java > CityResultDialog > initComponents()

```java
                        .addComponent(selectBtn, javax.swing.GroupLayout.PREFERRED_SIZE, pref:118, javax.swing.GroupLayout.PREFERRED_SIZE)))
                .addContainerGap(pref:114, Short.MAX_VALUE))
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(min:35, pref:35, max:35)
                .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, pref:48, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, pref:118, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(min:18, pref:18, max:18)
                .addComponent(selectBtn, javax.swing.GroupLayout.PREFERRED_SIZE, pref:37, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap(pref:38, Short.MAX_VALUE))
        );

    }// </editor-fold>//GEN-END:initComponents

    private JsonArray currData;
    public JsonArray getCurrData(){
        return currData;
    }
    public void setCurrData(JsonArray currData){
        this.currData = currData;
    }

    private JsonArray forecastData;
    public JsonArray getForecastData(){
        return forecastData;
```

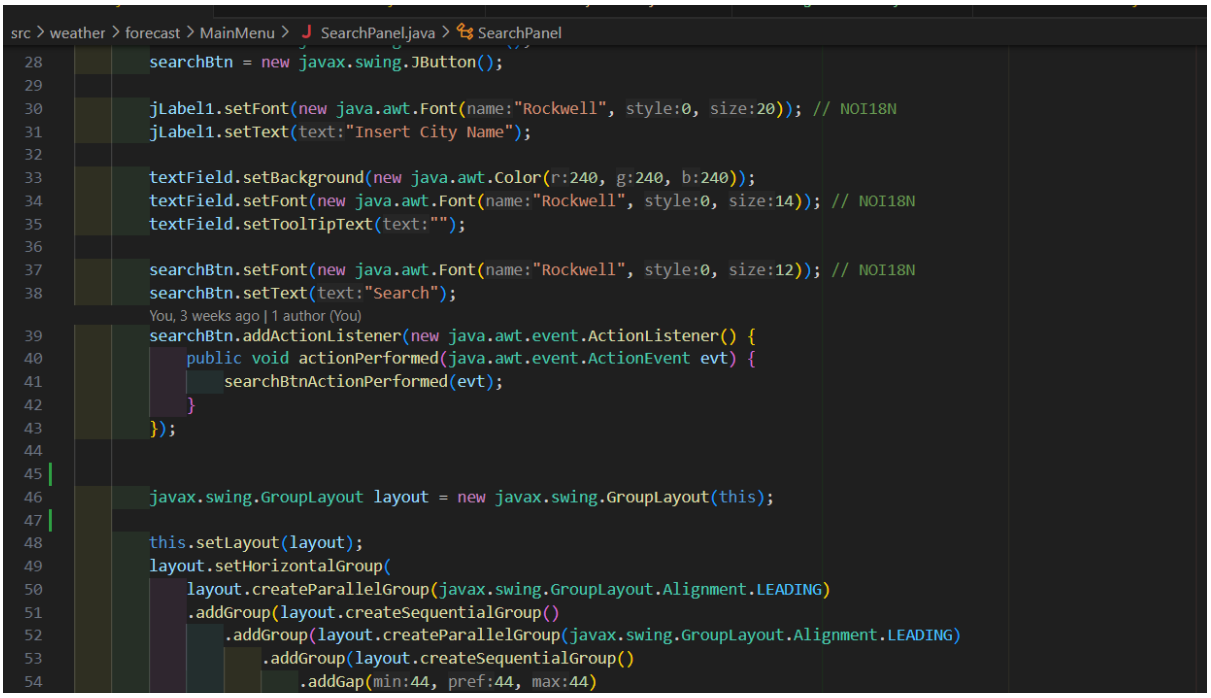

src > weather > forecast > MainMenu > CityResultDialog.java > CityResultDialog > initComponents()

```java
    }
    public void setForecastData(JsonArray forecastData){
        this.forecastData = forecastData;
    }

    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JList<String> cityList;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JButton selectBtn;
    // End of variables declaration//GEN-END:variables
}
```


src > weather > forecast > MainMenu > SearchPanel.java > SearchPanel

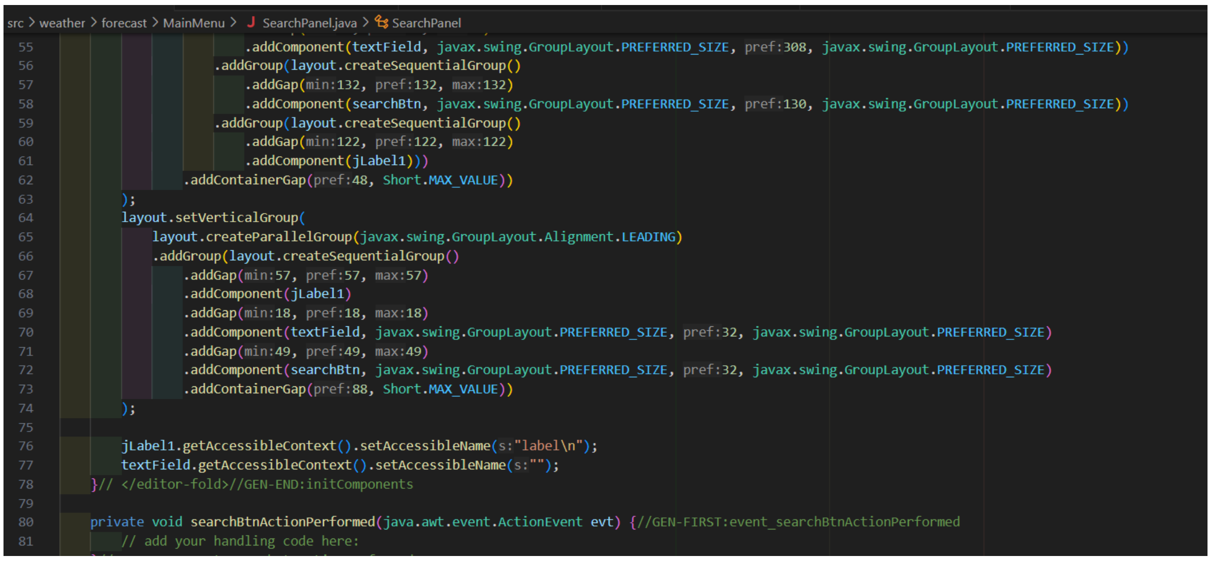
```java
package weather.forecast.MainMenu;

import java.awt.Image;
import java.awt.event.ActionListener;

import javax.swing.JTextField;

You, 1 second ago | 1 author (You)
/**
 *
 *       You, 1 second ago • Uncommitted changes
 */
public class SearchPanel extends javax.swing.JPanel {
    private Image backgroundImg;
    /**
     * Creates new form SearchPanel
     */
    public SearchPanel(ActionListener a) {
        initComponents();
        setOpaque(isOpaque:true);
        searchBtn.addActionListener(a);
    }

    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-BEGIN:initComponents
    private void initComponents() {
        jLabel1 = new javax.swing.JLabel();
        textField = new javax.swing.JTextField();
```

16



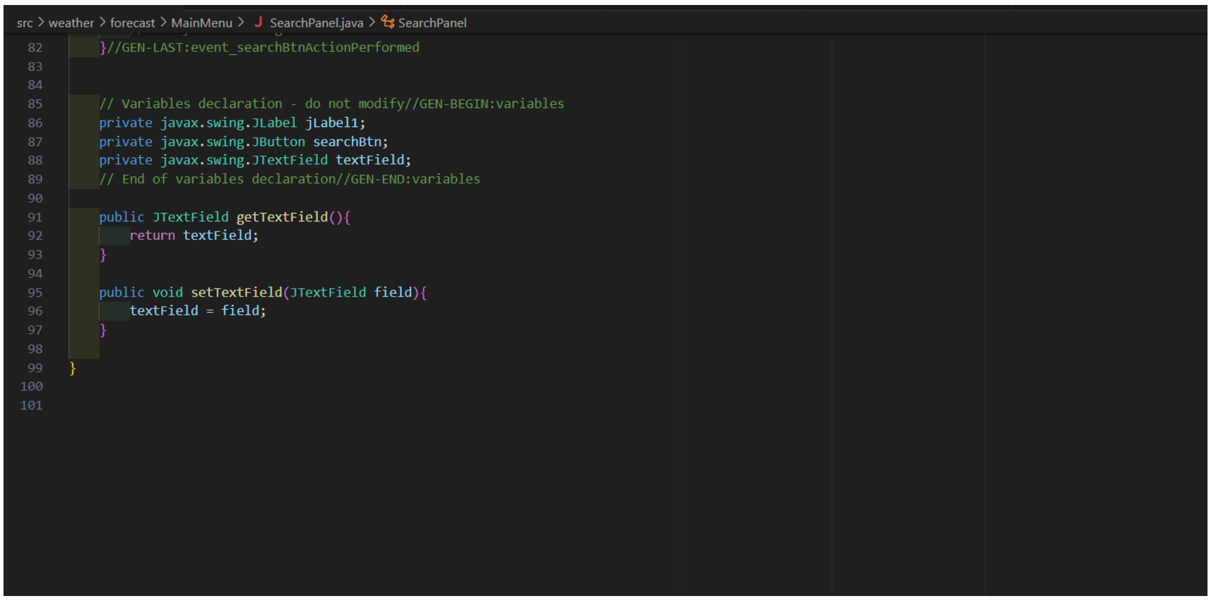
src > weather > forecast > MainMenu > SearchPanel.java > SearchPanel

```java
        searchBtn = new javax.swing.JButton();

        jLabel1.setFont(new java.awt.Font(name:"Rockwell", style:0, size:20)); // NOI18N
        jLabel1.setText(text:"Insert City Name");

        textField.setBackground(new java.awt.Color(r:240, g:240, b:240));
        textField.setFont(new java.awt.Font(name:"Rockwell", style:0, size:14)); // NOI18N
        textField.setToolTipText(text:"");

        searchBtn.setFont(new java.awt.Font(name:"Rockwell", style:0, size:12)); // NOI18N
        searchBtn.setText(text:"Search");
        You, 3 weeks ago | 1 author (You)
        searchBtn.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                searchBtnActionPerformed(evt);
            }
        });


        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(this);

        this.setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(layout.createSequentialGroup()
                        .addGap(min:44, pref:44, max:44)
```



src > weather > forecast > MainMenu > SearchPanel.java > SearchPanel

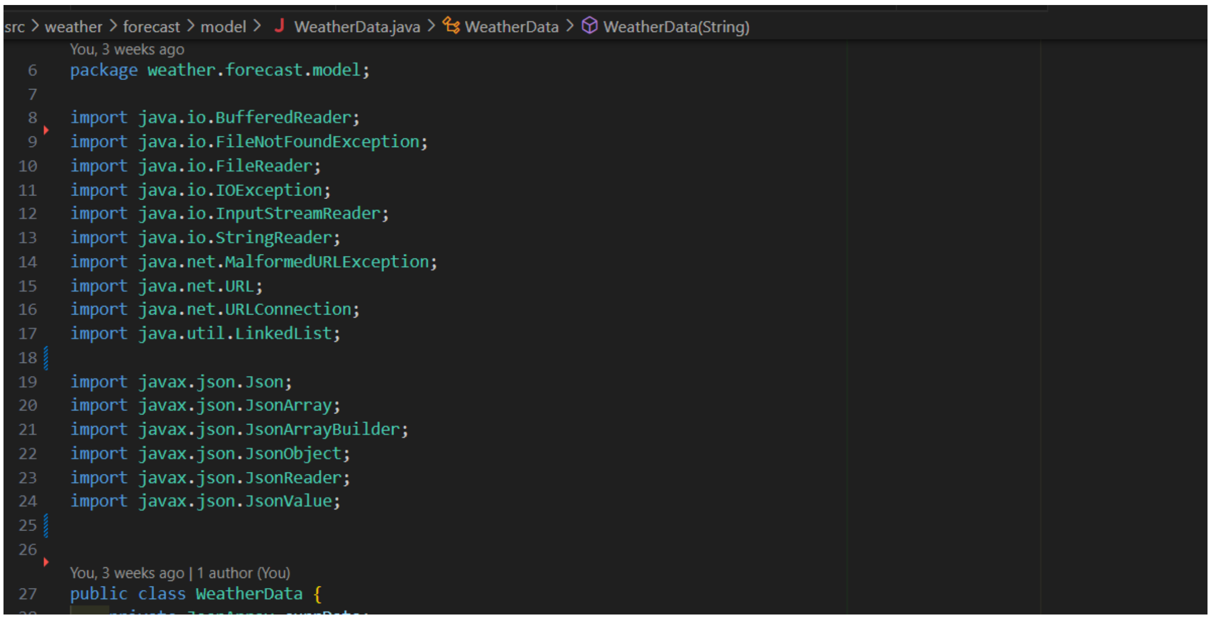
```java
                        .addComponent(textField, javax.swing.GroupLayout.PREFERRED_SIZE, pref:308, javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addGroup(layout.createSequentialGroup()
                        .addGap(min:132, pref:132, max:132)
                        .addComponent(searchBtn, javax.swing.GroupLayout.PREFERRED_SIZE, pref:130, javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addGroup(layout.createSequentialGroup()
                        .addGap(min:122, pref:122, max:122)
                        .addComponent(jLabel1)))
                .addContainerGap(pref:48, Short.MAX_VALUE))
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(min:57, pref:57, max:57)
                .addComponent(jLabel1)
                .addGap(min:18, pref:18, max:18)
                .addComponent(textField, javax.swing.GroupLayout.PREFERRED_SIZE, pref:32, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(min:49, pref:49, max:49)
                .addComponent(searchBtn, javax.swing.GroupLayout.PREFERRED_SIZE, pref:32, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap(pref:88, Short.MAX_VALUE))
        );

        jLabel1.getAccessibleContext().setAccessibleName(s:"label\n");
        textField.getAccessibleContext().setAccessibleName(s:"");
    }// </editor-fold>//GEN-END:initComponents

    private void searchBtnActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_searchBtnActionPerformed
        // add your handling code here:
```



src > weather > forecast > MainMenu > SearchPanel.java > SearchPanel

```java
    }//GEN-LAST:event_searchBtnActionPerformed


    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JLabel jLabel1;
    private javax.swing.JButton searchBtn;
    private javax.swing.JTextField textField;
    // End of variables declaration//GEN-END:variables

    public JTextField getTextField(){
        return textField;
    }

    public void setTextField(JTextField field){
        textField = field;
    }

}
```

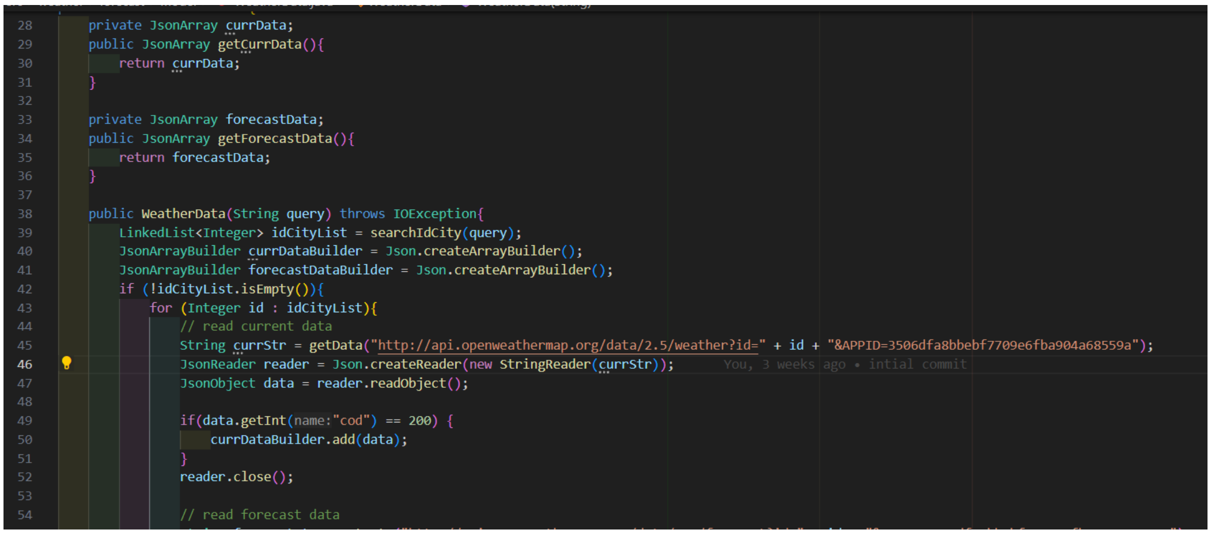
EXTERNAL API:



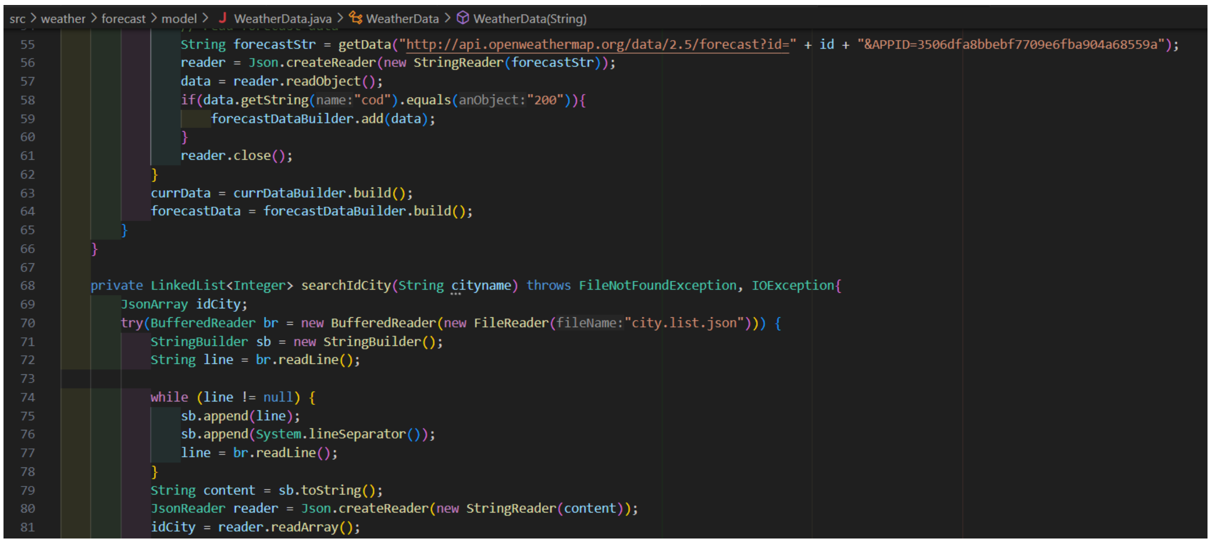
src > weather > forecast > model > WeatherData.java > WeatherData > WeatherData(String)

You, 3 weeks ago

```java
package weather.forecast.model;

import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.StringReader;
import java.net.MalformedURLException;
import java.net.URL;
import java.net.URLConnection;
import java.util.LinkedList;

import javax.json.Json;
import javax.json.JsonArray;
import javax.json.JsonArrayBuilder;
import javax.json.JsonObject;
import javax.json.JsonReader;
import javax.json.JsonValue;


You, 3 weeks ago | 1 author (You)
public class WeatherData {
```

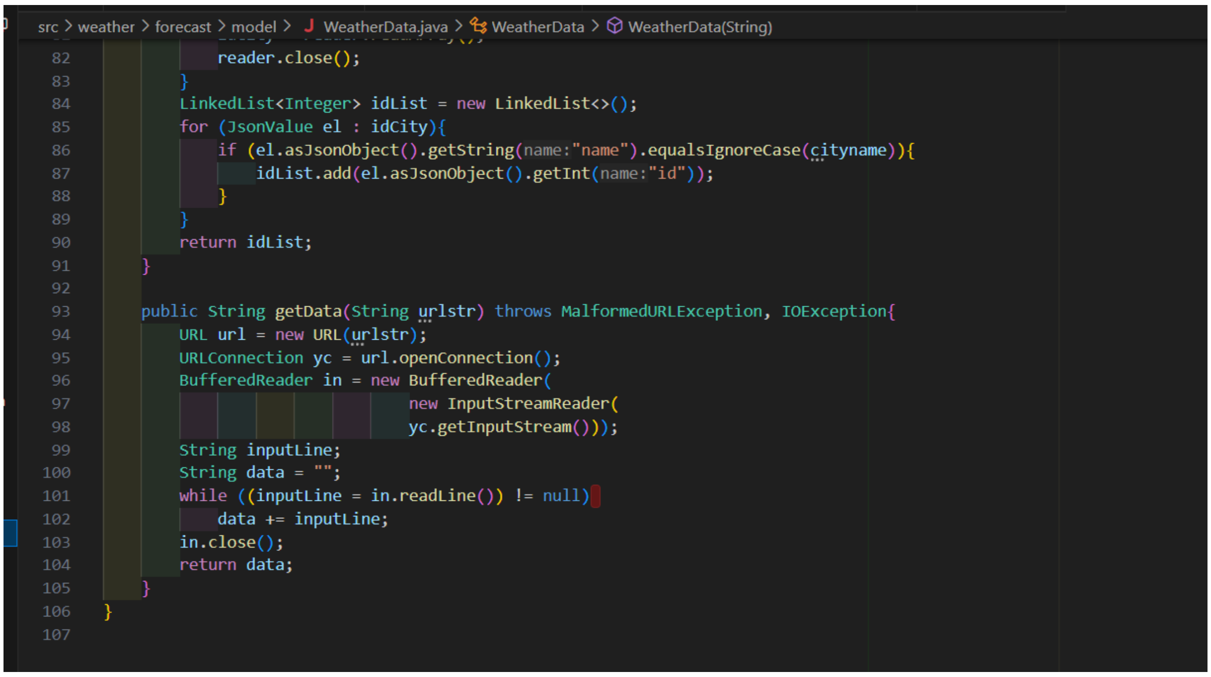


```java
    private JsonArray currData;
    public JsonArray getCurrData(){
        return currData;
    }

    private JsonArray forecastData;
    public JsonArray getForecastData(){
        return forecastData;
    }

    public WeatherData(String query) throws IOException{
        LinkedList<Integer> idCityList = searchIdCity(query);
        JsonArrayBuilder currDataBuilder = Json.createArrayBuilder();
        JsonArrayBuilder forecastDataBuilder = Json.createArrayBuilder();
        if (!idCityList.isEmpty()){
            for (Integer id : idCityList){
                // read current data
                String currStr = getData("http://api.openweathermap.org/data/2.5/weather?id=" + id + "&APPID=3506dfa8bbebf7709e6fba904a68559a");
                JsonReader reader = Json.createReader(new StringReader(currStr));      You, 3 weeks ago • intial commit
                JsonObject data = reader.readObject();

                if(data.getInt(name:"cod") == 200) {
                    currDataBuilder.add(data);
                }
                reader.close();

                // read forecast data
```



src > weather > forecast > model > WeatherData.java > WeatherData > WeatherData(String)

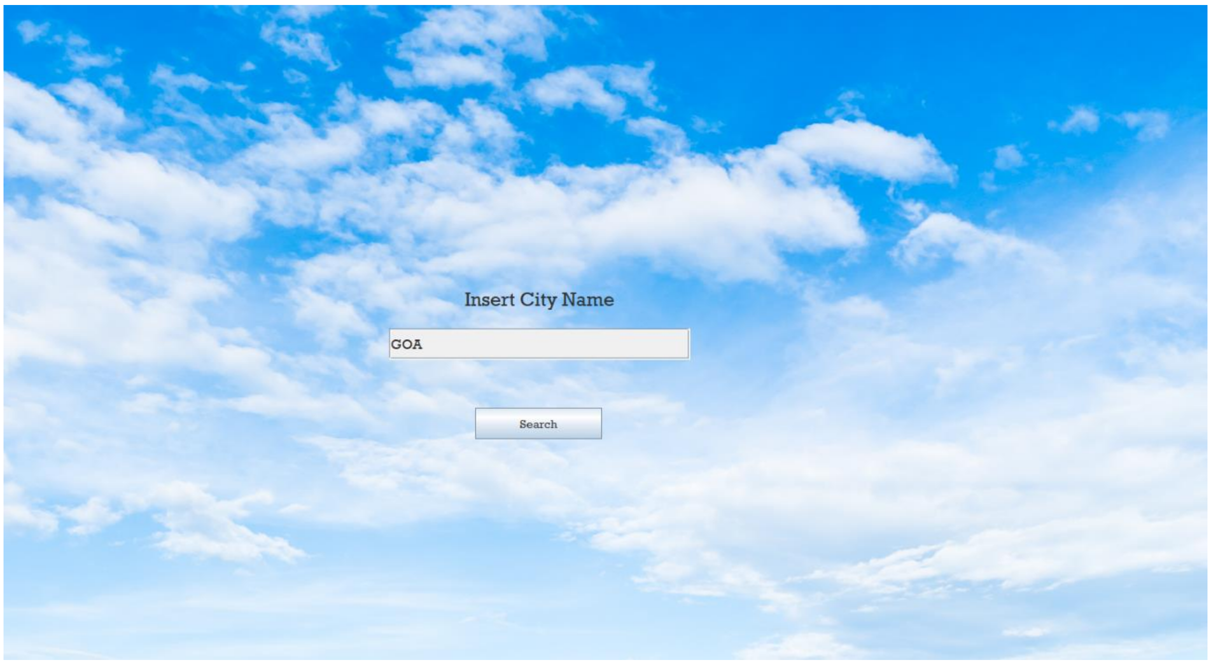
```java
                String forecastStr = getData("http://api.openweathermap.org/data/2.5/forecast?id=" + id + "&APPID=3506dfa8bbebf7709e6fba904a68559a");
                reader = Json.createReader(new StringReader(forecastStr));
                data = reader.readObject();
                if(data.getString(name:"cod").equals(anObject:"200")){
                    forecastDataBuilder.add(data);
                }
                reader.close();
            }
            currData = currDataBuilder.build();
            forecastData = forecastDataBuilder.build();
        }
    }

    private LinkedList<Integer> searchIdCity(String cityname) throws FileNotFoundException, IOException{
        JsonArray idCity;
        try(BufferedReader br = new BufferedReader(new FileReader(fileName:"city.list.json"))) {
            StringBuilder sb = new StringBuilder();
            String line = br.readLine();

            while (line != null) {
                sb.append(line);
                sb.append(System.lineSeparator());
                line = br.readLine();
            }
            String content = sb.toString();
            JsonReader reader = Json.createReader(new StringReader(content));
            idCity = reader.readArray();
```

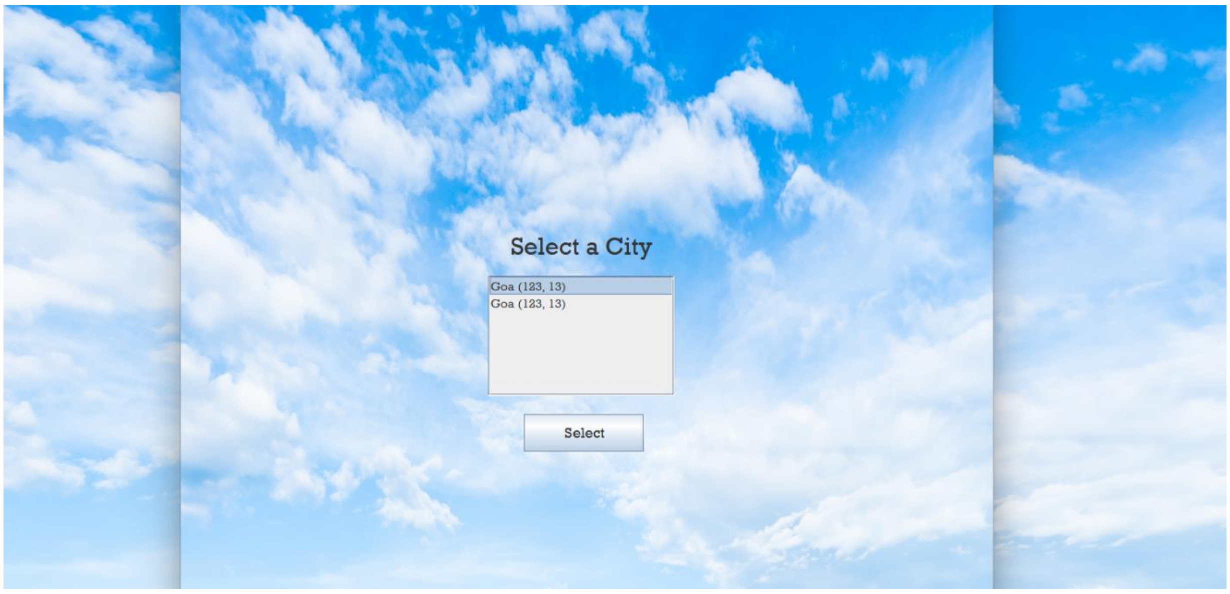


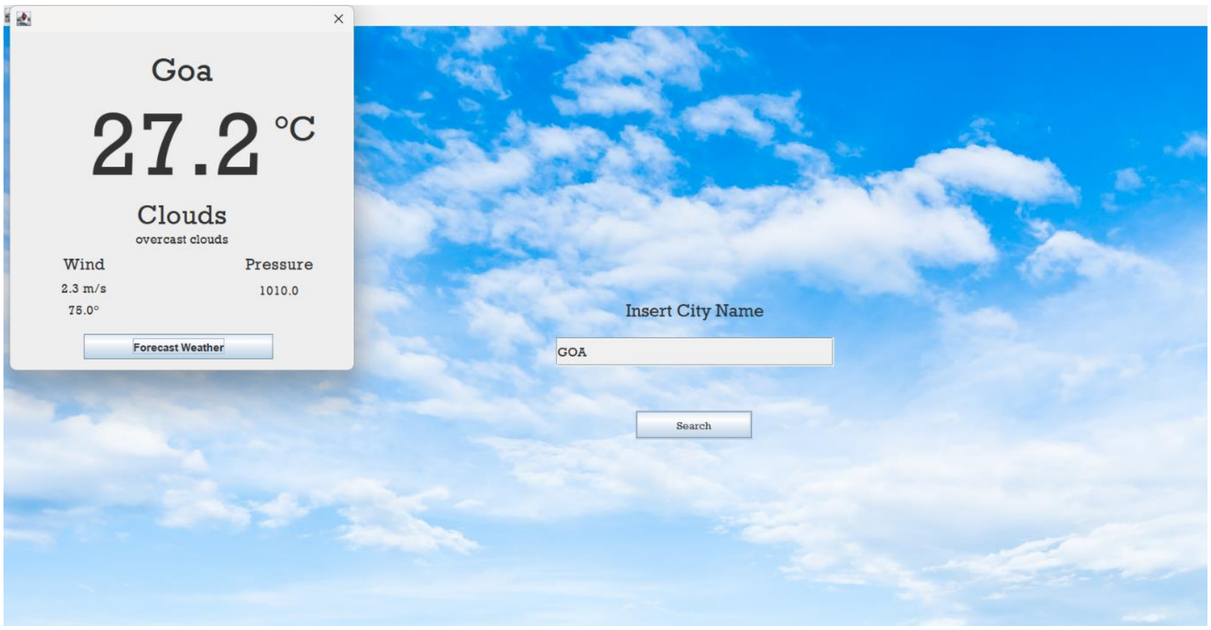
src > weather > forecast > model > WeatherData.java > WeatherData > WeatherData(String)

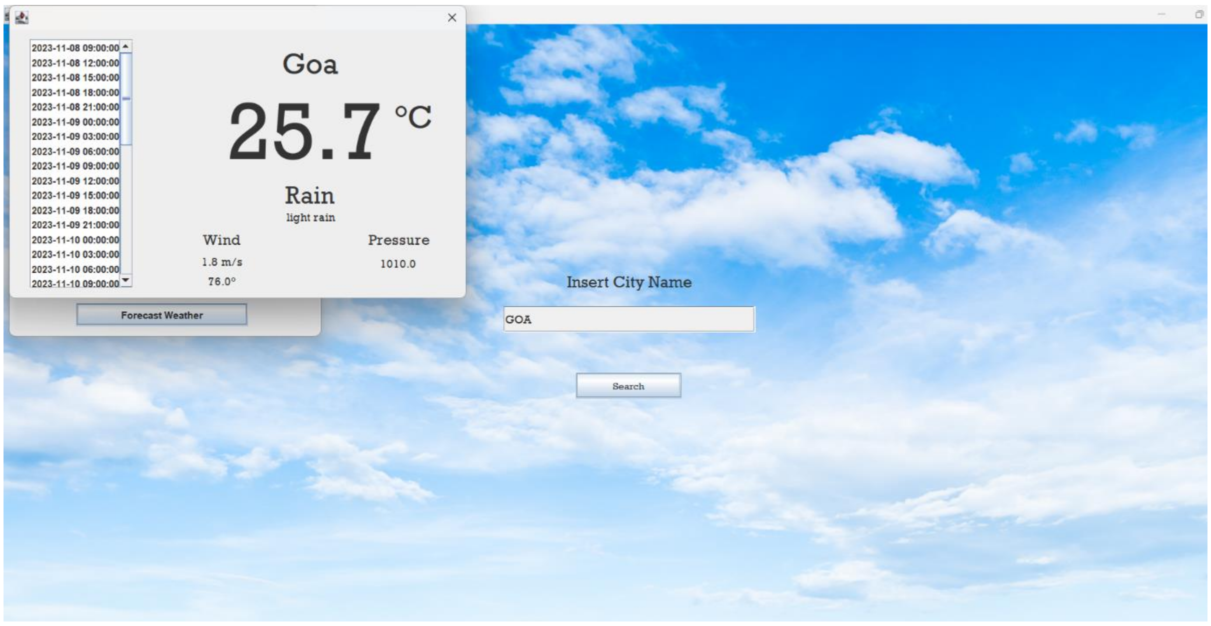
```java
            reader.close();
        }
        LinkedList<Integer> idList = new LinkedList<>();
        for (JsonValue el : idCity){
            if (el.asJsonObject().getString(name:"name").equalsIgnoreCase(cityname)){
                idList.add(el.asJsonObject().getInt(name:"id"));
            }
        }
        return idList;
    }

    public String getData(String urlstr) throws MalformedURLException, IOException{
        URL url = new URL(urlstr);
        URLConnection yc = url.openConnection();
        BufferedReader in = new BufferedReader(
                                new InputStreamReader(
                                yc.getInputStream()));
        String inputLine;
        String data = "";
        while ((inputLine = in.readLine()) != null)
            data += inputLine;
        in.close();
        return data;
    }
}
```

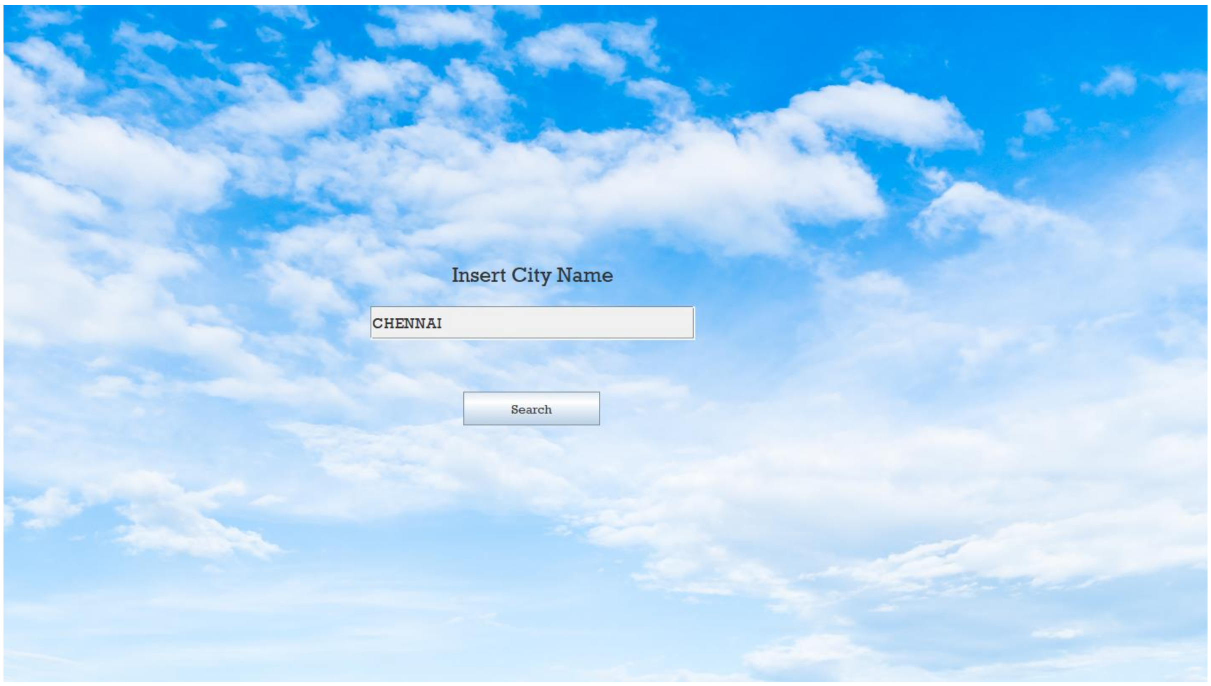
6.RESULTS AND DISCUSSION

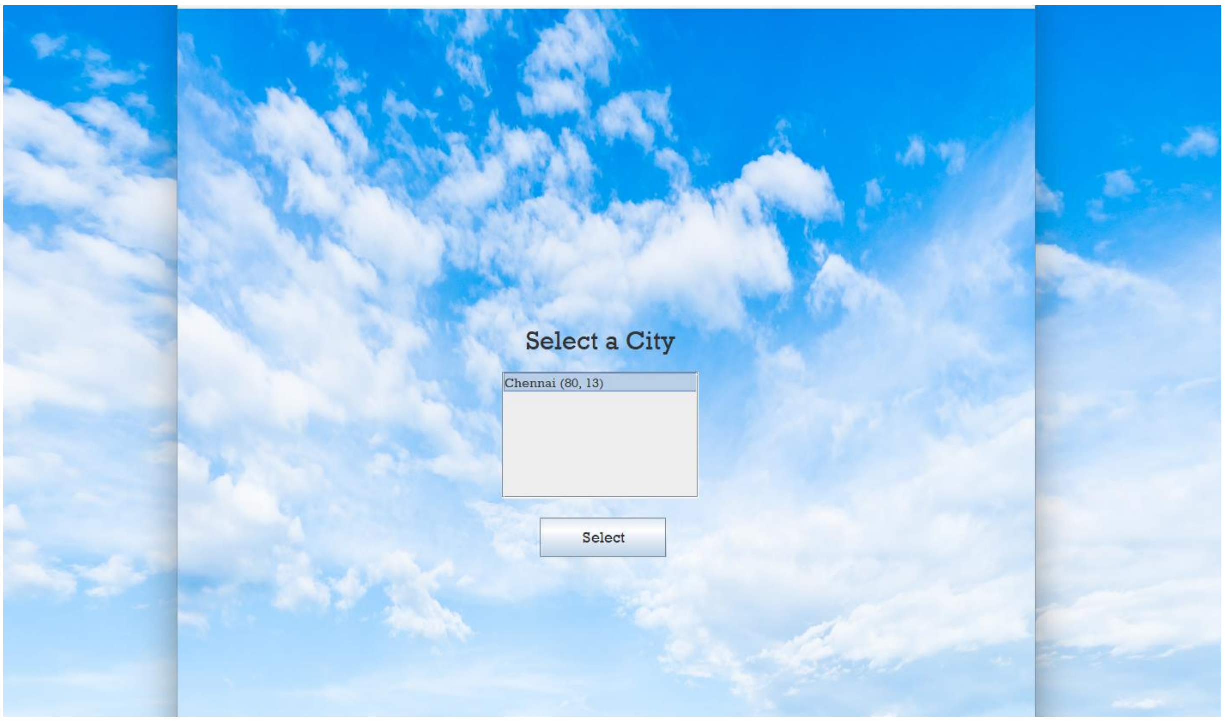


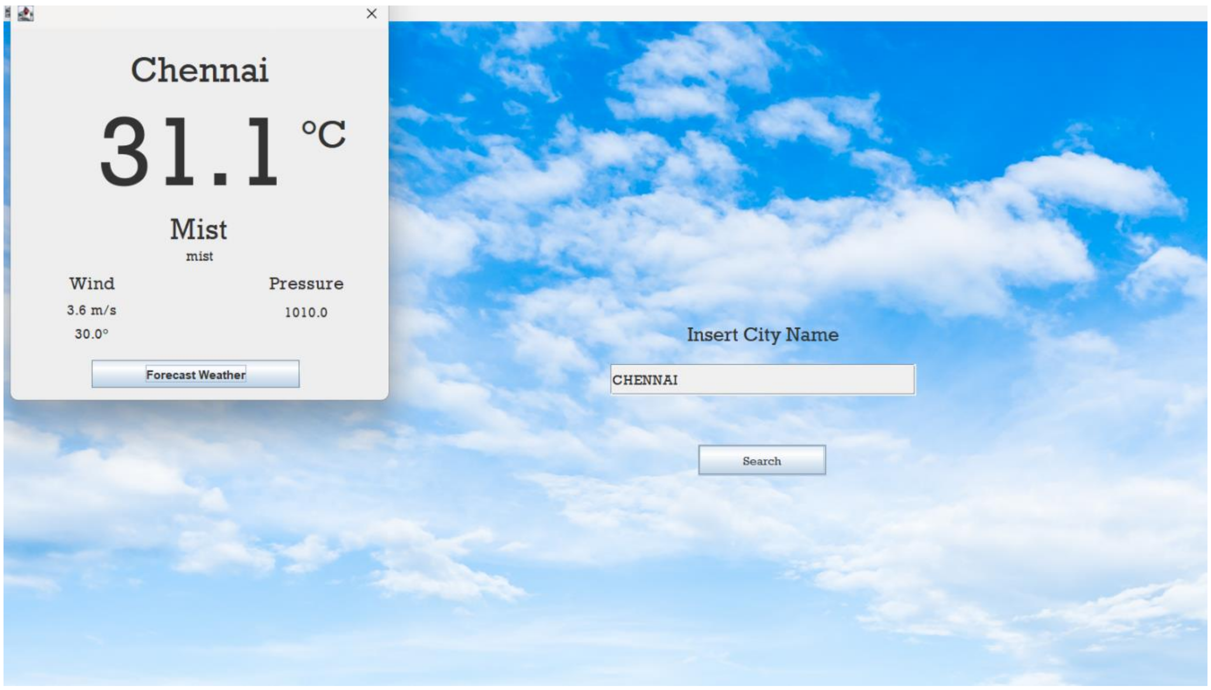



```
Goa
27.2 °C
Clouds
overcast clouds
Wind                Pressure
2.3 m/s             1010.0
75.0°
```

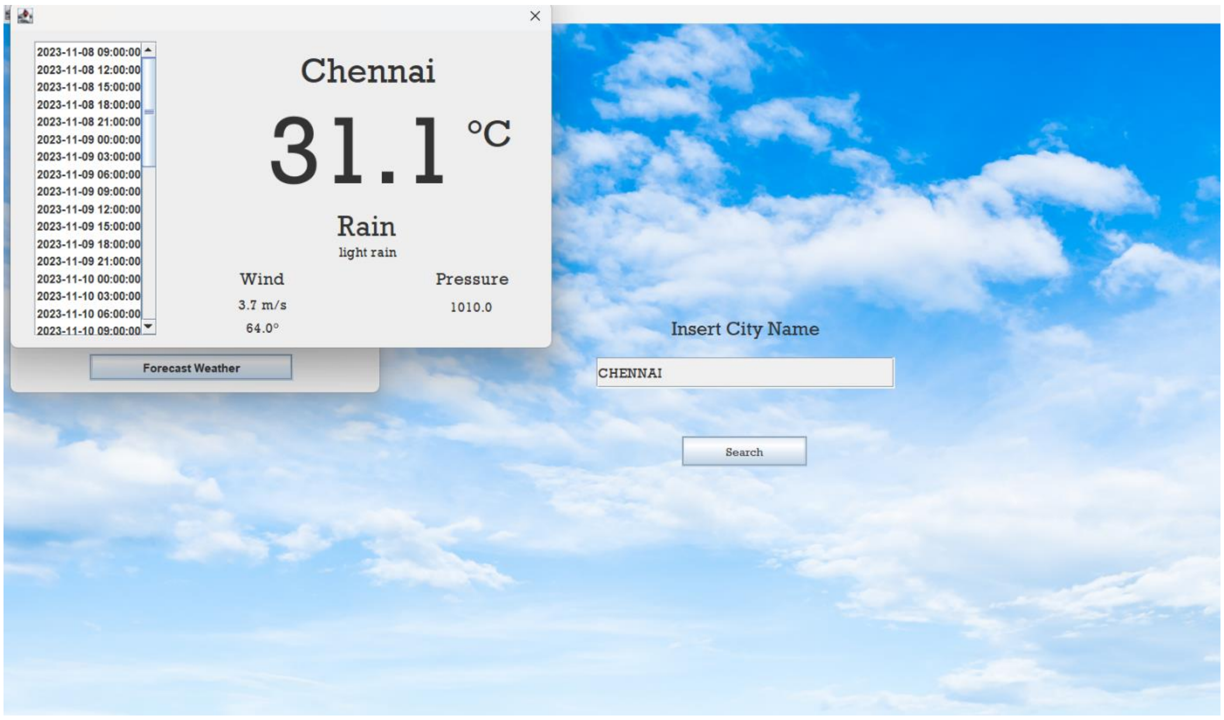

```
2023-11-08 09:00:00
2023-11-08 12:00:00
2023-11-08 15:00:00
2023-11-08 18:00:00
2023-11-08 21:00:00
2023-11-09 00:00:00
2023-11-09 03:00:00
2023-11-09 06:00:00
2023-11-09 09:00:00
2023-11-09 12:00:00
2023-11-09 15:00:00
2023-11-09 18:00:00
2023-11-09 21:00:00
2023-11-10 00:00:00
2023-11-10 03:00:00
2023-11-10 06:00:00
2023-11-10 09:00:00
```
```
Goa
25.7 °C
Rain
light rain
Wind                Pressure
1.8 m/s             1010.0
76.0°
```


Insert City Name
CHENNAI
Search


Select a City
Chennai (80, 13)
Select



```
Chennai
31.1 °C
Mist
mist
Wind            Pressure
3.6 m/s         1010.0
30.0°
```



```
2023-11-08 09:00:00
2023-11-08 12:00:00
2023-11-08 15:00:00
2023-11-08 18:00:00
2023-11-08 21:00:00
2023-11-09 00:00:00
2023-11-09 03:00:00
2023-11-09 06:00:00
2023-11-09 09:00:00
2023-11-09 12:00:00
2023-11-09 15:00:00
2023-11-09 18:00:00
2023-11-09 21:00:00
2023-11-10 00:00:00
2023-11-10 03:00:00
2023-11-10 06:00:00
2023-11-10 09:00:00
```

```
Chennai
31.1 °C
Rain
light rain
Wind            Pressure
3.7 m/s         1010.0
64.0°
```

7.CONCLUSION

7.1. CONCLUSION

The Weather Forecast application is a versatile and user-friendly tool that provides users with the ability to access up-to-date weather information for cities around the world. By integrating with the OpenWeatherMap API, it delivers real-time data on current weather conditions and 5-day forecasts, allowing users to make informed decisions and plan their activities accordingly. The application's flexibility is a key highlight, enabling developers to adapt and extend its functionality to meet specific project requirements. Its clean and intuitive user interface ensures a seamless user experience. While the application's success largely depends on the availability and accuracy of the API, it serves as a valuable resource for individuals and developers seeking reliable weather data. Whether used as a standalone weather tool or integrated into larger projects, this application is a valuable asset for staying updated on weather conditions.

7.2 FUTURE ENHANCEMENT

- **Location Services**: Implement geolocation services to automatically detect the user's current location, reducing the need for manual input.

- **Historical Weather Data**: Provide historical weather data for a location, allowing users to view past weather conditions and trends.

- **Data Visualization**: Enhance the app with interactive charts and graphs to display weather trends over time, such as temperature, rainfall, and wind speed.

- **Offline Access:** Implement the capability to store and retrieve weather data for offline access, useful when users have limited internet connectivity.

- **Feedback and Reporting**: Allow users to report inaccuracies in weather data and provide feedback to improve the app.

8.REFRENCE

- Open Weather Map API documentation: https://openweathermap.org/api
- Dark Sky API documentation: https://darksky.net/dev
- Java Swing tutorial: https://docs.oracle.com/javase/tutorial/uiswing/
- Stack Overflow question on how to display weather API data in JFrame: https://stackoverflow.com/questions/22332720/weather-api-display-in-jframe
- GitHub repository for a simple weather forecast app using Java Swing and OpenWeatherMap API: https://github.com/yonasadiel/weather-forecast