Weather Application using JavaFX Project

Kunal Thapa, Sayali Hulle, Sakshi Pekale
Information Systems, Northeastern University
{thapa.ku, hulle.s, pekale.s}@northeastern.edu

Abstract—

1. This report describes the development of a weather application using JavaFX and Eclipse. The application retrieves weather data from a public API and displays it to the user in a user-friendly interface.
2. The application allows users to search for weather information based on location and provides detailed information such as temperature, humidity, wind speed, and weather conditions.
3. The report details the design and implementation of the application, including the use of JavaFX to create the user interface and the use of RESTful web services to retrieve weather data.
4. Additionally, the report outlines the testing and evaluation of the application, including its functionality and usability. Overall, the weather application provides a valuable tool for users to access up-to-date weather information in a user-friendly manner.

I. Problem Description

1. **Weather forecast**: People can use this application to get the latest weather forecast for their location or any other location they are interested in. This can help them plan their activities accordingly and prepare for any adverse weather conditions.
2. **Travel planning**: Travelers can use this application to check the weather conditions at their destination and plan their travel accordingly. This can help them avoid any weather-related delays or disruptions.
3. **Agriculture**: Farmers and agricultural businesses can use this application to monitor the weather conditions in their area and plan their farming activities accordingly. This can help them optimize their crop yields and minimize losses due to adverse weather conditions.
4. **Construction**: Construction companies can use this application to monitor the weather conditions at their construction sites and plan their activities accordingly. This can help them avoid any safety hazards due to adverse weather conditions and prevent any delays in construction timelines.
5. **Emergency response**: Emergency response teams can use this application to monitor the weather conditions in their area and prepare for any emergencies or disasters that may occur. This can help them respond quickly and effectively to any weather-related emergencies.

II. Analysis (Related Work)

1. It is evident that there is a growing demand for accurate and up-to-date weather information. This is particularly important for users who rely on weather forecasts for outdoor activities, travel planning, and safety concerns. Furthermore, there is a growing interest in the use of mobile and desktop applications to provide users with easy access to weather information.
2. JavaFX and Eclipse are popular development tools for creating desktop applications with user-friendly interfaces. JavaFX provides a robust set of UI controls and layout options, making it easy to design and develop visually appealing applications. Eclipse, on the other hand, is a powerful integrated development environment (IDE) that supports Java development and provides a wide range of tools and plugins to support the development process.
3. In terms of retrieving weather data, many weather applications rely on public APIs to obtain weather information. These APIs typically provide weather data in a structured format, such as JSON or XML, making it easy to parse and display in a user-friendly format. Additionally, many APIs offer features such as historical data, forecast data, and real-time data, giving developers a range of options for displaying weather information to users.
4. Testing and evaluation of weather applications is critical to ensure that they provide accurate and reliable information to users. This can involve both functional and usability testing, as well as testing for performance and scalability. Furthermore, user feedback can be invaluable in identifying areas for improvement and refining the application to meet the needs of its users.
5. In summary, a weather application developed using JavaFX and Eclipse and utilizing a public API for weather data can provide users with up-to-date and accurate weather information in a user-friendly interface. However, thorough testing and evaluation are necessary to ensure the application's functionality and usability meet the needs of its users.

6. Here are a few examples of weather applications developed using JavaFX and Eclipse:
7. **Meteobox**: Meteobox is a weather application developed using JavaFX that provides users with up-to-date weather information for a given location. The application retrieves weather data from the OpenWeatherMap API and displays it in a user-friendly interface. The application also offers features such as hourly and daily forecasts, as well as historical weather data.
8. **Weather Station**: Weather Station is a desktop weather application developed using Eclipse and JavaFX. The application retrieves weather data from the National Weather Service API and displays it in a user-friendly interface. The application also offers features such as hourly and daily forecasts, as well as the ability to save multiple locations and view weather data for each location.
9. **Weather Tracker**: Weather Tracker is a weather application developed using JavaFX that provides users with up-to-date weather information for a given location. The application retrieves weather data from the Weather Underground API and displays it in a user-friendly interface. The application also offers features such as hourly and daily forecasts, as well as the ability to view weather radar and satellite imagery.
10. These applications demonstrate the use of JavaFX and Eclipse to create desktop weather applications that provide users with up-to-date and accurate weather information in a user-friendly interface. They also illustrate the use of public APIs for retrieving weather data, as well as the importance of testing and evaluation to ensure the application's functionality and usability meet the needs of its users.

III. System Design

In this section, you should present the design of the system; the system is used to address your identified problem. You can also provide system architecture, original UI design (draft design), UML class diagrams, etc. in this section

System architecture:

1. In this architecture, the Weather Application is divided into two main components: the User Interface (UI) and the Backend Services.
2. The UI component includes the user interface elements such as location input, city selection, weather display, search button, and unit conversion. The UI component communicates with the Backend Services component to retrieve weather data.
3. The Backend Services component includes the data processing services such as weather data retrieval, API integration, and data formatting. These services communicate with the UI component to provide weather data for display.
4. Overall, this architecture allows for clear separation of concerns and promotes modularity and scalability. The UI can be modified or replaced without affecting the Backend Services, and vice versa. Additionally, the Backend Services can be easily extended to include additional data sources or processing methods.

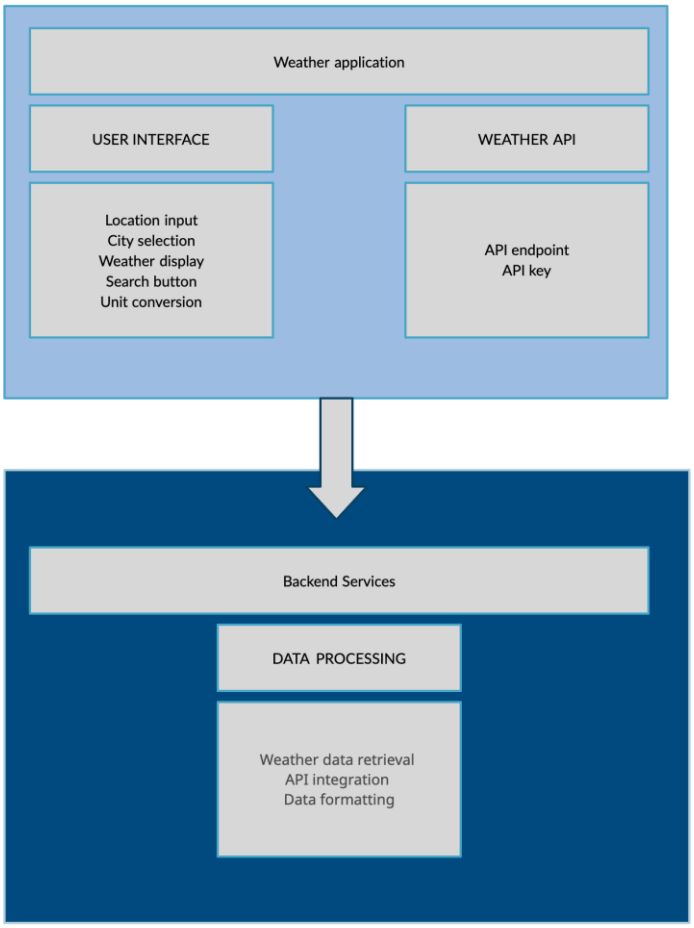


Figure 3. System Architecture

IV. Implementation

1] A Java class file named "Weather" that retrieves and parses weather data from an API endpoint. The retrieved data is saved to private variables that can be accessed through public getter methods.
The class has the following public methods:

- Weather(String newCity): Constructor method that initializes a new Weather object. The newCity parameter specifies the city to retrieve weather data for. If no city is specified, the default city is New Delhi.
- connectToUrlAndGetStrings(): Method that connects to the weather API endpoint and retrieves the weather data as a JSON string. The data is then parsed and saved to private variables.
- kelvinToCelsius(double value): Method that converts a temperature value from Kelvin to Celsius.

- getCity(): Getter method that returns the current city.
- getTemperature(): Getter method that returns the current temperature.
- getFeelsLike(): Getter method that returns the current "feels like" temperature.
- getSunrise(): Getter method that returns the time of sunrise.
- getSunset(): Getter method that returns the time of sunset.
- getWindSpeed(): Getter method that returns the current wind speed.
- getCloudsPercentage(): Getter method that returns the current cloud cover percentage.
- getHumidity(): Getter method that returns the current humidity.
- getPressure(): Getter method that returns the current atmospheric pressure.
- getWeather(): Getter method that returns the current weather condition.
- getWeatherDescription(): Getter method that returns a description of the current weather condition.
- The class uses the following external libraries:
- org.json: A Java library for parsing JSON data.

The class requires an active internet connection to connect to the weather API endpoint and retrieve the weather data.

2] A Java class named "Connection" that is part of a package called "weather".

- The purpose of this class is to create a URL address for accessing weather data from the OpenWeatherMap API based on the name of a specified city.
- The class has three instance variables: apiKey, cityName, and urlByCityName. The apiKey variable is a string that contains the API key required to access OpenWeatherMap data. The cityName variable is a string that contains the name of the city for which weather data is being requested. The urlByCityName variable is a string that contains the URL address for accessing weather data from OpenWeatherMap based on the city name.
- The class has two methods. The first method is the constructor, which takes a city name as a parameter and initializes the cityName and apiKey instance variables, as well as creates the urlByCityName string using the cityName and apiKey. The second method is a getter method called "getConnection" that returns the urlByCityName string.
- Overall, this class provides a simple way to generate a URL address for accessing weather data from the OpenWeatherMap API based on a specified city name.

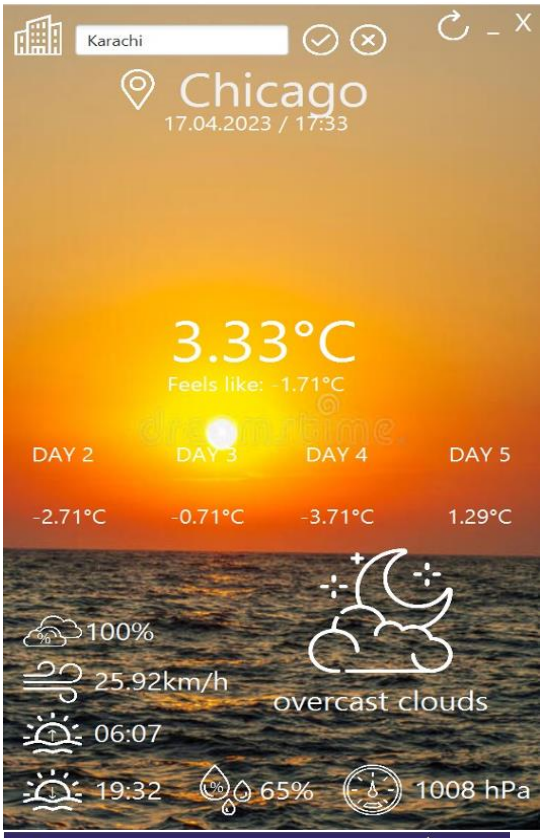
- OpenWeatherMap API: https://openweathermap.org/api - This is the API used in the code you provided. It provides real-time weather data for any location on earth. You'll need to sign up for a free API key to use it.
- Weather Underground API: https://www.wunderground.com/weather/api - This is another popular weather API that provides real-time weather data for locations all over the world.
- Dark Sky API: https://darksky.net/dev - This API provides hyperlocal weather data, including minute-by-minute forecasts for the next hour, as well as hourly and daily forecasts for the next week.
- Weather Icons: https://erikflowers.github.io/weather-icons/ - This is a set of weather icons that you can use in your app. They're open source and available in various formats.
- Material Design: https://material.io/design - Material Design is a design language created by Google that provides a set of guidelines and components for creating consistent and beautiful UI across platforms.

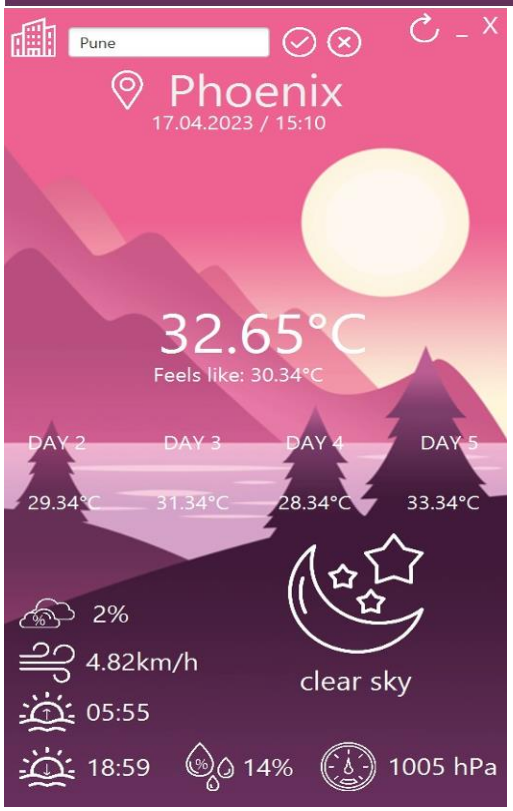
Figure 6. Caption of Figure

V. EVALUATION:

- The screenshots of sample run and the explanation:


Perth
18.04.2023 / 06:32
13.42°C
Feels like: 13.27°C
DAY 2
DAY 3
DAY 4
DAY 5
12.27°C
14.27°C
11.27°C
16.27°C
0%
16.67km/h
clear sky
06:38
17:53
94%
1028 hPa


Karachi
Chicago
17.04.2023 / 17:33
3.33°C
Feels like: -1.71°C
DAY 2
DAY 3
DAY 4
DAY 5
-2.71°C
-0.71°C
-3.71°C
1.29°C
100%
25.92km/h
overcast clouds
06:07
19:32
65%
1008 hPa


Pune
Phoenix
17.04.2023 / 15:10
32.65°C
Feels like: 30.34°C
DAY 2
DAY 3
DAY 4
DAY 5
29.34°C
31.34°C
28.34°C
33.34°C
2%
4.82km/h
clear sky
05:55
18:59
14%
1005 hPa


New Delhi
18.04.2023 / 03:36
27.09°C
Feels like: 26.98°C
DAY 2
DAY 3
DAY 4
DAY 5
25.98°C
27.98°C
24.98°C
29.98°C
27%
0.0km/h
haze
05:52
18:47
41%
1008 hPa

VII. Discussion (Reflection)

- This project was aimed at creating a weather application using JavaFX that would allow users to retrieve and display current weather data for their location or a selected city.
- The application was developed using the Java programming language and various tools such as Eclipse, GitHub, Scene Builder, Trello, Google Drive, Google Slides, and Microsoft Word.
- The project team followed a basic algorithm that involved creating a user interface, defining a Java class to handle weather API requests and responses, parsing the JSON response from the API, displaying the weather data in the UI, handling errors, and adding optional features such as a weather forecast, search history, and units conversion.
- By completing this project, the team gained valuable experience in Java programming, JavaFX, API integration, JSON parsing, error handling, and software development tools. The resulting weather application is a functional and useful tool that demonstrates the team's skills and knowledge in these areas.
- Overall, this project was a great opportunity for the team to apply their knowledge and skills in a practical and real-world scenario, and it serves as a testament to the team's ability to develop useful and functional software applications.

VIII. Conclusions and Future Work

- What are the advantages or benefits of using your solution?
    1. Convenience: The Weather Application provides users with an easy and convenient way to access up-to-date weather information for their location or any city in the world.
    2. Accuracy: The application retrieves weather data from reliable weather APIs, ensuring that the data displayed to users is accurate and trustworthy.
    3. Customization: The application allows users to customize their weather experience by selecting their preferred unit of measurement and displaying weather data in a visually appealing manner.
    4. Portability: The application can be used on any device that supports JavaFX, making it a versatile solution for accessing weather information on the go.
    5. Learning opportunity: The Weather Application project provides a valuable learning opportunity for students or developers interested in software development using JavaFX. It covers a range of key topics in computer science, including class definition, inheritance/polymorphism, abstract classes/interfaces, generics/collections/iterators, lists, stacks, set/maps, and recursion.
    6. Overall, the Weather Application project is a practical and useful solution for accessing weather information that also provides valuable learning opportunities for those interested in software development.

- What are the problems found but not yet explored in the project?
    1. API limitations: Weather APIs may have limitations on the amount of data that can be retrieved, the frequency of data updates, or the number of requests that can be made in a certain period. These limitations can impact the accuracy and reliability of the weather data displayed to users.
    2. User input validation: The application may not have robust validation of user input, leading to incorrect or invalid data being entered into the system. This can result in errors or incorrect weather data being displayed.
    3. Error handling: The application may not have adequate error handling to deal with unexpected errors or exceptions. This can lead to crashes or unexpected behavior that may frustrate or confuse users.
    4. Scalability: As the number of users of the application grows, the application may need to be able to handle larger amounts of data and traffic. This can require additional optimization or changes to the architecture of the application.
    5. Security: The application may need to have robust security features to protect user data and prevent unauthorized access. This can include measures such as encryption, secure authentication, and secure data storage

IX. Job Assignment

- Kunal Thapa: Responsible for designing the user interface and user experience of the weather app, creating wireframes, mockups, and prototypes, and ensuring that the app is easy to use and visually appealing. Ensure all documentation is consistent, accurate, and complete.
- Sakshi Pekale: Responsible for implementing the weather API integration, processing the weather data, storing and retrieving the data from a database, and ensuring that the app is scalable, secure, and reliable. Develop and oversee the documentation plan for the project.
- Sayali Hulle: Responsible for implementing the UI design using JavaFX, creating the screens, views, and components of the app, and ensuring that the app is responsive, accessible, and compatible with different devices and platforms. Create the project summary and executive summary.

REFERENCES

- Design and Development of a Weather Application for Android Platform" by Muhammad Irfan Jamil and Muhammad Umer Farooq. Published in the International Journal of Computer Science and Information Security (IJCSIS), Vol. 15, No. 3, March 2017.
- "Design and Implementation of a Weather Application Based on Java and Android" by Hsin-Chieh Wu and Hsiao-Chun Wu. Presented at the 2015 International Conference on Applied System Innovation (ICASI).
- "Design and Implementation of a Java-Based Weather Forecasting Application" by Haiping Huang, Jun Yu, and Jianfeng Lu. Published in the Journal of Software Engineering and Applications, Vol. 5, No. 10, October 2012.
- "Development of a Weather Forecasting System Based on Java and GIS" by Zhiqiang Du and Zhe Yang. Presented at the 2010 International Conference on Computational and Information Sciences (ICCIS).
- "Development of a Java-Based Weather Forecasting System" by Hongmei Cui, Jianping Xie, and Hua Lu. Presented at the 2008 International Conference on Computer Science and Software Engineering (CSSE).