The University of Western Ontario
Department of Computer Science
CS2212b 2015

Team Project Specification

Your team will develop a weather-viewing program that allows a user to view the current weather in their current location.

Base Functional Specifications

All teams should complete the specifications in this section. They are worth 60% of the marks allocated to the programming portion of the team project.

Start up and Exit

1. The application name must have your team number as a prefix (Ex. Team 4 could use the name `4_TheWeather.jar`)
2. It must be possible to run the application by executing the command: `java -jar` *`teamJARname.jar`*, where `teamJARname.jar` is the name of your team's jar file (Ex. `4_TheWeather.jar`)
    2.1. All dependencies must be contained within your JAR file.
3. The user must be able to quit the application at any time
4. On initial startup, the user must be able to set their current location
5. User information (including configuration preferences) must persist between runs of the program.

Local weather view

6. The user must be able to change their current location
    6.1. Changing the current location must update all weather views to show the weather for the new location
7. The user must be able to view the current location
8. The user must be able to view the current weather for their current location including
    8.1. The temperature
    8.2. The wind speed and direction

    8.3. The air pressure
    8.4. The humidity
    8.5. The condition of the sky (clear, partial cloud, etc.)
    8.6. The expected minimum and maximum temperature
    8.7. The time of sunrise and sunset
9. The user must be shown an icon indicating sky state (cloudy, sunny, etc.)
10. The user must be able to select the units in which they want their weather displayed (Celsius, Fahrenheit, etc.)
11. The user must be shown the time at which the weather information was last updated
12. The user must be able to refresh the weather data

Additional Components Specifications

The team should add the following components based on their interests. Each component listed is given a grade value which indicates how much the implementation is worth at maximum in the marks allocated to the programming portion of the team project.

Combining additional components will make the team eligible for 100% of the marks allocated to the programming portion on the team project. Upon submission, teams will indicate which components they would like to be graded. The components added to the base specifications to be graded cannot total more than 100%, but do not have to total 100% (i.e. your group could decide to aim for a 90% completion instead of a 100%).

Teams can also propose additional components of their own creation. These will be evaluated for difficulty and assigned a grade value by the teaching staff. All additional component proposals must be made by the due date of the first stage.

Custom Weather View (5%)

13. The user must be able to hide or remove information in the local weather view
14. The user must be able to reveal information in the local weather view that had been hidden/removed
15. The user's preferences must persist between program runs

My Locations (5%)

16. The user must be able to add a new location to a location list
17. The user must be able to view their location list

18. The user must be able to change their current location by selecting a location from their location list
    18.1. The weather data of all views must be updated to the new city

---

Multiple Locations (10%)

19. The user must be able to open a view that shows the weather of multiple locations as once, including the following information for each location
    19.1. The temperature
    19.2. The wind speed and direction
    19.3. The air pressure
    19.4. The humidity
    19.5. The condition of the sky (clear, partial cloud, etc.)
    19.6. An icon indicating the condition of the sky
20. The user must be able to add locations to the multiple location view
21. The user must be able to remove locations from the multiple location view

---

Short-term Forecast (15%)

22. The user must be able to view a short term forecast of their current city in 3 hour increments for at least the next 24 hours which includes at least
    22.1. the temperature,
    22.2. the condition of the sky
    22.3. an icon representing the sky condition

Long-term Forecast (15%)

23. The user must be able to view a daily forecast for their current location for at least the following 5 days including
    23.1. the temperature,
    23.2. the condition of the sky
    23.3. an icon representing the sky condition
    23.4. The expected minimum and maximum temperature
    ~~23.5. The time of sunrise and sunset~~

Precipitation (5%)

24. The user must be able to see precipitation levels in the short and long term forecasts (if implemented)

25. The user must be able to see an expected precipitation amount for the next 24 hours on the local weather view.

Weather on Mars (10%)

26. The user must be able to select Mars as a location
27. The current weather data must be gathered from the web service at http://marsweather.ingenology.com/ (Note: their JSON format is different from that of OpenWeatherMap)
28. The weather data displayed must include
    - 28.1. The temperature
    - 28.2. The wind speed and direction
    - 28.3. The air pressure
    - 28.4. The humidity
    - 28.5. The condition of the sky (clear, partial cloud, etc.)
    - 28.6. An icon indicating the sky condition

Technical Specifications

1. The application must run on the Linux systems in MC 10.
    - 1.1. A virtual machine identical to the MC 10 systems will be provided to you for convenience.
    - 1.2. You can, however, develop outside of the virtual machine on any system, but you must ensure that the final product will run within the virtual machine (test early and test often).
2. The application must be developed in Java.
3. The application must have a Swing graphical user interface.
4. It must be possible to both compile and package the application into an executable JAR file using the Maven command mvn package .
5. Data persistence between runs must be stored using object serialization.
6. The weather data must be gathered from the web service OpenWeatherMap.org in JSON format
7. The system must be easy to use. For example, a user must be able to navigate easily between screens, and must have the option of quitting or going back to a previous screen at all times.
    - 7.1. Messages, as well as errors, must be reported correctly and consistently.
    - 7.2. Keep in mind that popup messages are jarring to a user and interrupt his/her workflow. They should typically be used only in

exceptional circumstances. When considering using a popup message, ask yourself whether or not there is another way to convey the same information.

8. You must make sure that your code is commented with javadoc-style comments so that it can be modified and reused by others.
9. The code implementing your project must be written by members of your team.