



**SYSTEM SOFTWARE**

**GROUP: B**

**TEAM: 4**

**CONTRIBUTORS:**

**Dovydas Skackovas (N0805114)**

**Justin Kwok (T0095695)**

**Man Li (T0084893)**

**Marco Choi (T0088203)**

## **1. Abstract:**

### **1.1 : Background**

The assessment scenario is based on creating a weather stations controlled by microcomputers for the farmer's worker to connect and login to the system server with a simple GUI to check the visualised informations on the field and allow download their respective data on request. The language Javascript and the application NetBeans IDE will be used for the whole project programming, the explanation documents will be presented inside NetBeans application.

### **1.2 : Aims**

For this project, the aims are mainly divided into four parts:

- 1) To create two GUI that can operate by the user include user login and user client. One GUI for the weather station as a server.
- 2) To design a login system that can validate username and password. Username and password must be matched with the database in order to access the client
- 3) Working user client after successful login, it allows user to download and show the weather data on the client screen and switch between weather stations to read different data. It also have an option to log-out / disconnect from the client.
- 4) A weather station server to check who is using the client. It can generate weather data and output the data to the user client if requested.

### **1.3 : Objectives**

The objectives are for creating a working client and server according to the aims.

- 1) The main program is linked to the login GUI and user client GUI along with the user account document which store default username and password.
- 2) Weather station class generates weather information and sends it to the user client if the user had requested it.
- 3) Server class stores every data that will be used during the performance.

## 2. Feature, design and implementaion

### Features

- Weather station

All of the information below is generated randomly and is sending the data out .

1. Lattitude
2. Longtitude
3. Elevation
4. Temperature
5. Humidity
6. Barometric pressure
7. Wind force

- Server

Server class is where the server is set up and where all of the data is stored in, includes weather station field information and data also clients account data .

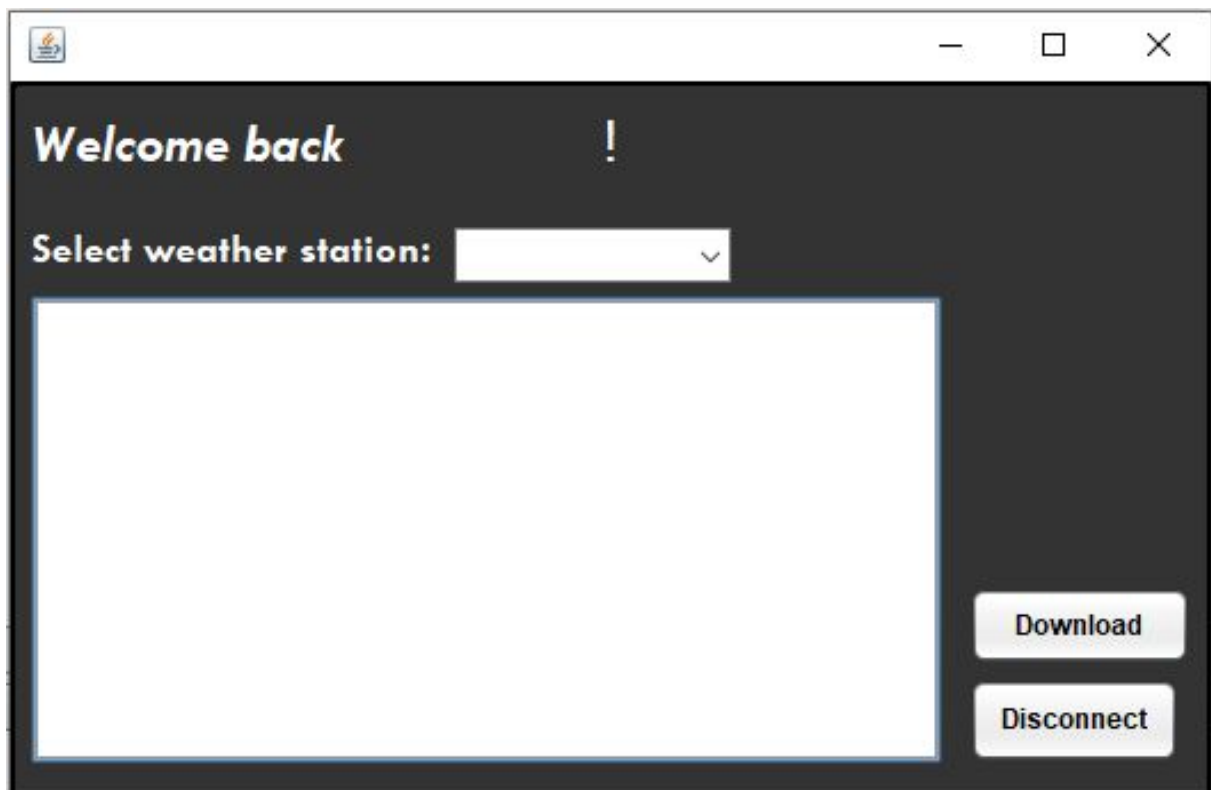
- Main

Main class mainly is do catching event in login page and main page if user or client clicked those button there.

### 3. Design

A screenshot of a login window with a dark background. At the top left is a small icon of a flame, and at the top right is a close button (X). The word "Login" is displayed in large white font. Below it, the labels "Username:" and "Password:" are in white, followed by white text input fields. To the right of the fields is a large, light gray button with the text "Connect" in bold black font.

This is GUI for Login page, users can login into the software with a registered account and password. If the account and password was correct, after clicking the connect button, the software will take user to the main page which can use the function in the software.

A screenshot of the main page window with a dark background. At the top left is a small icon of a flame, and at the top right are standard window controls (minimize, maximize, close). The text "Welcome back" is in white, followed by a large white exclamation mark. Below this, the label "Select weather station:" is in white, followed by a white dropdown menu. A large white rectangular area occupies the center of the window. In the bottom right corner, there are two stacked light gray buttons with the text "Download" and "Disconnect" in bold black font.

Here is the GUI for user's main page. If there is weather station online, then user can able to select that or them in the scrollbar button, then using the Download button, all the weather station information will be shows below.



This the GUI for server, it will shows all the user and weather station client if they are online.

## 4. Implementation

- **Protocol choices:**

A socket is one end-point of a two-way communication link between two programs running on the network. Socket classes are used to represent the connection between a client program and a server program. We used this for communication between client-server-weather stations with a handshake procedure.

- **Multithreading:**

Multithreading means you can have several executables within the program. It is used for maximum utilization of the CPU. Multithreaded program works like different CPU's working on different parts of the code at the same time.

The Multithreading in our project is that our project allow multiple file run and respond at the same time. For example, while Server.java was running, weatherstation.java and Main.java can be run at the same time. While running other .java when Server.java was running at the first place data and responds will be made and displayed.

## **5. Conclusion and potential future work**

### **Summary:**

The main idea behind this program was to make a network infrastructure which would connect the client and the weather stations together. The client can view information for every weather station. All of have been made through teamwork and good communication as everyone were assigned to do work in different fields and do research.

### **Improvements for future work:**

Overall, the team is satisfied with the work but the program could have many more features and be more polished. We thought of couple of ways we could improve our program in the future:

- With more time our team could have worked on getting real information from a weather website.
- The GUI could have been clearer and more pleasing for the user.
- Different types of data could have been added

## References:

1. Tutorialspoint (2020) *Java - Multithreading*, Available at: [https://www.tutorialspoint.com/java/java\\_multithreading.html](https://www.tutorialspoint.com/java/java_multithreading.html) (Accessed: 7th May, 2020).
2. Jakob Jenkov (2020-03-29) *Java Concurrency and Multithreading Tutorial*, Available at: <http://tutorials.jenkov.com/java-concurrency/index.html> (Accessed: 7th May,2020).
3. PrashantSingh27 (2020) *Multithreading in Java*, Available at: <https://www.geeksforgeeks.org/multithreading-in-java/> (Accessed: 7th May,2020).
4. baeldung (August 14, 2019) *Reading a CSV File into an Array*, Available at: <https://www.baeldung.com/java-csv-file-array> (Accessed: 7th May,2020).
5. baeldung (October 16, 2019) *Introduction to OpenCSV*, Available at: <https://www.baeldung.com/opencsv> (Accessed: 7th May,2020).
6. tutorialspoint (2020) *Java.io.DataInputStream.readUTF() Method*, Available at: [https://www.tutorialspoint.com/java/io/datainputstream\\_readutf.htm](https://www.tutorialspoint.com/java/io/datainputstream_readutf.htm) (Accessed: 7th May,2020).
7. Oracle (2020) *Class DataInputStream*, Available at: <https://docs.oracle.com/javase/7/docs/api/java/io/DataInputStream.html> (Accessed: 7th May,2020).
8. John Thompson (26th Dec, 2017) *Random Number Generation in Java*, Available at: <https://docs.oracle.com/javase/7/docs/api/java/io/DataInputStream.html> (Accessed: 7th May,2020).