**Systems Software Coursework**

**Group 9**

**T0269826 Yakoubi Sifal**

**N0894808 Allsop, Samuel**

**N0894873 Alves da Cruz Tocha, Joao**

**N0868224 Barrett, George**

**N0895830 Kotwal, Nabeel**

**Introduction:**

The aim of our project is, using java, to build a distributed system on a TCP/IP-based enable network infrastructure that will allow its users to connect, get information about the field and the network and be able to upload their own data. In this report, it will be discussed how the application was planned, how the application was designed, and the thought process behind the decisions and design choices accompanied by some helpful diagrams and design mockups.

**Features, design and implementation:**

Before establishing a connection, the client will be prompted for a username and a password, to insure that only one user can access uploaded information.

Each Weather client is given a different ID, which is stored by the server when a connection is established and used to launch the correct service provider handler. This ID is registered automatically and the client has the choice to upload some information to the server such as GPS positioning, temperature and humidity. This data is then stored in the server.

After a connection has been established, the client will get an automatic up-to-date description of the field, then he can request a new one by clicking a button.

A GUI has been implemented to facilitate usage of the application and display all of the data.

**Protocol choice explanation**:

* Tcp/ip is an industry–standard model that can be effectively deployed in practical networking problems.
* Used for cross platform communications for diverse networks.
* It is not owned by any institute, thus it can be used by any person or organization.
* It is a scalable, client-server architecture. This allows networks to be added without disrupting the current services.
* It makes it easy to identify devices on the network by assigning an ip address for each one, providing name and address resolution services.

**Multithreading handling:**

Multithreaded was handled by creating an executor that sets the max amount of threads available (newFixedThreadPool currently set to 5). After a newly born thread is started it becomes runnable by implementing the runnable interface, When a client leaves the server, that thread is open for another client to use after it has cycled through the rest of the available threads methods like sleep were also implemented to cease the execution of the current thread.

**Recording active users:**

In order to record connected users a Linked list has been used to store all Weather stations connected, these can be displayed with a combobox later in the GUI.

**Establishing connections:**

The accept()method waits until a client starts and requests a connection on the host and port of this server. When a connection is requested and successfully established, the accept()method returns a new Socket object. It's bound to the same local port, and its remote address and remote port are set to match the client's. The server can communicate with the client over this new object and listen for client connection requests.

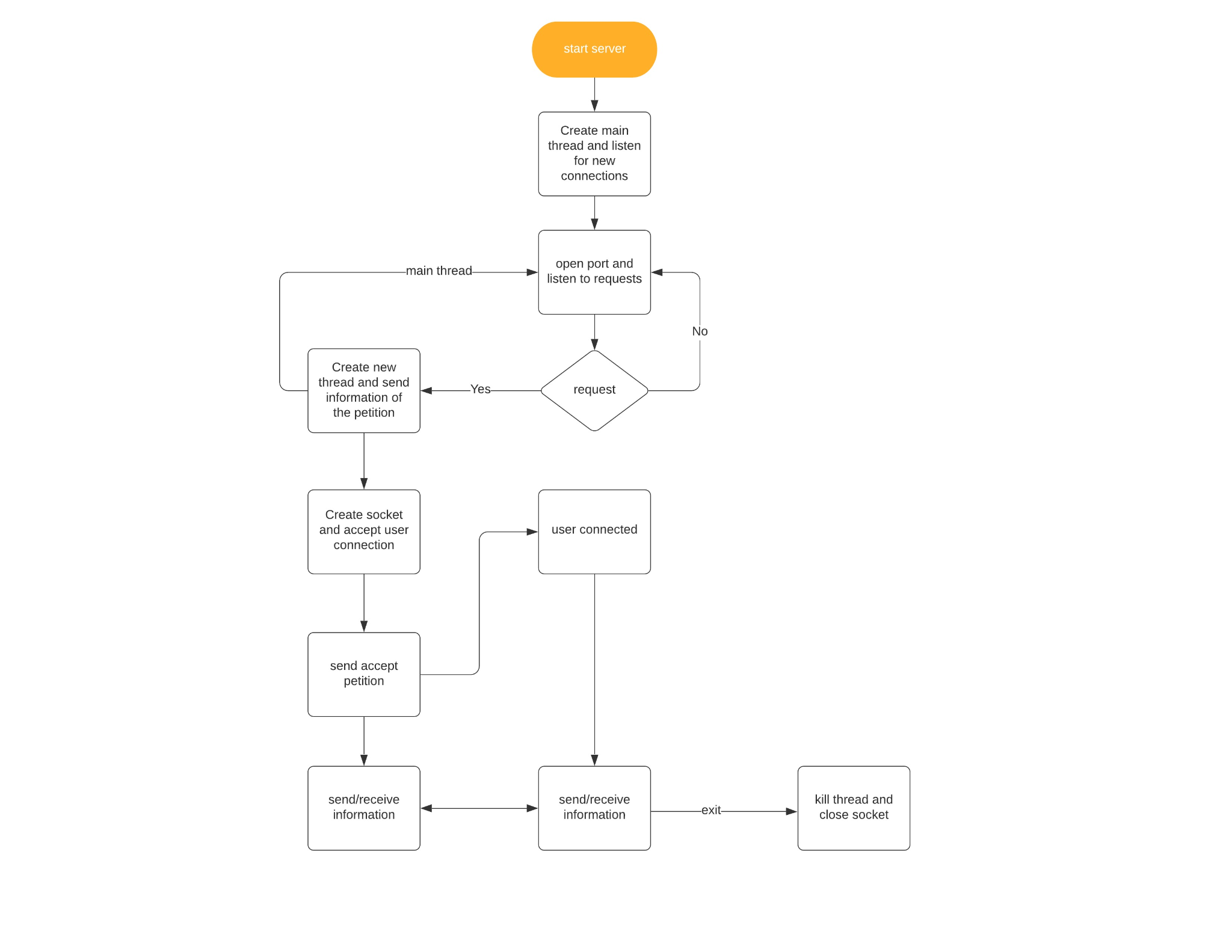
**Other used structures:**

Datagrams were used to store the weather information sent from the weather client.

**Additional features:**

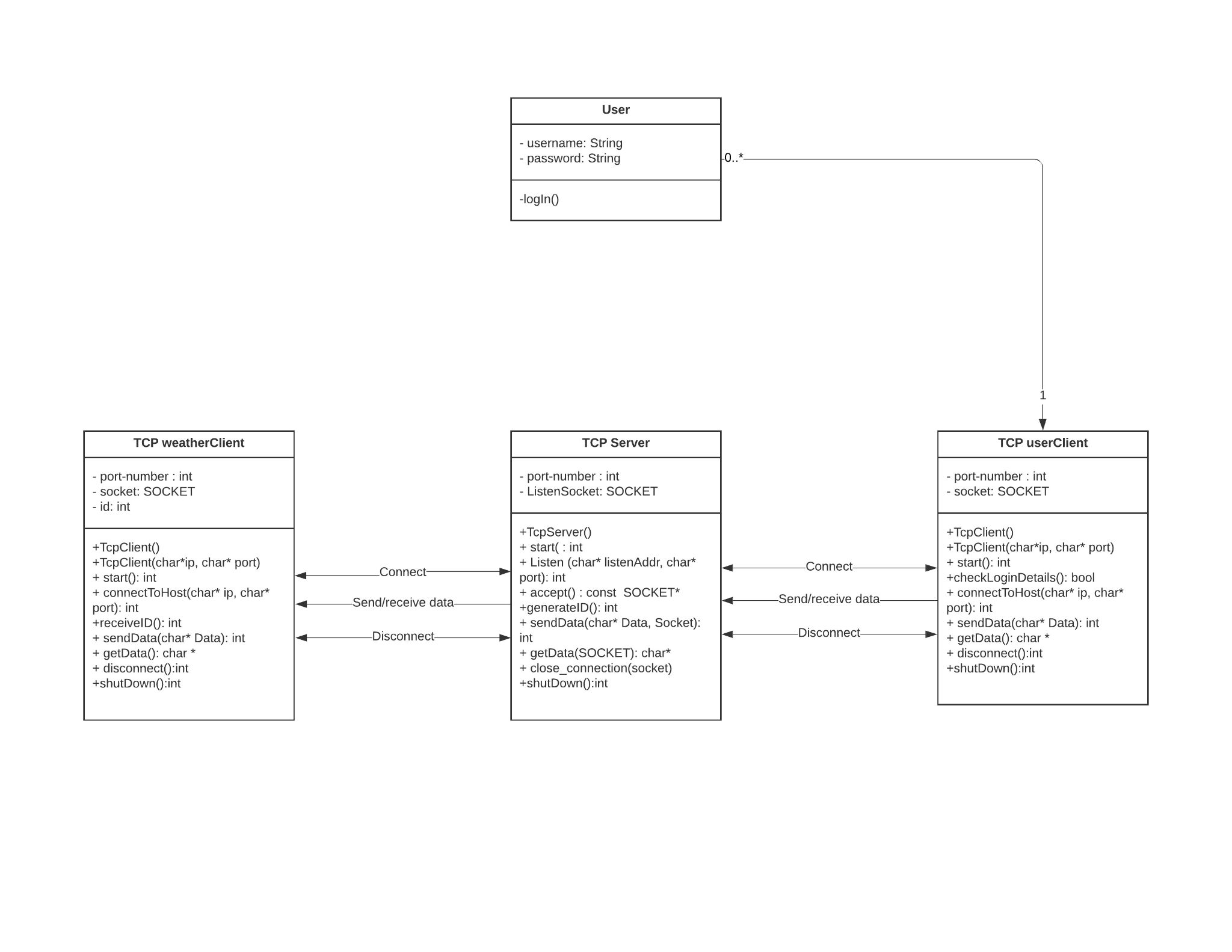
Connection to server status is represented with a small square, green = connected, red = not connected.

**Project Design:**

****

**Flow diagram:**

This diagram represents the flow of this software, it shows the underlying elements and their interaction.



**Class diagram:**

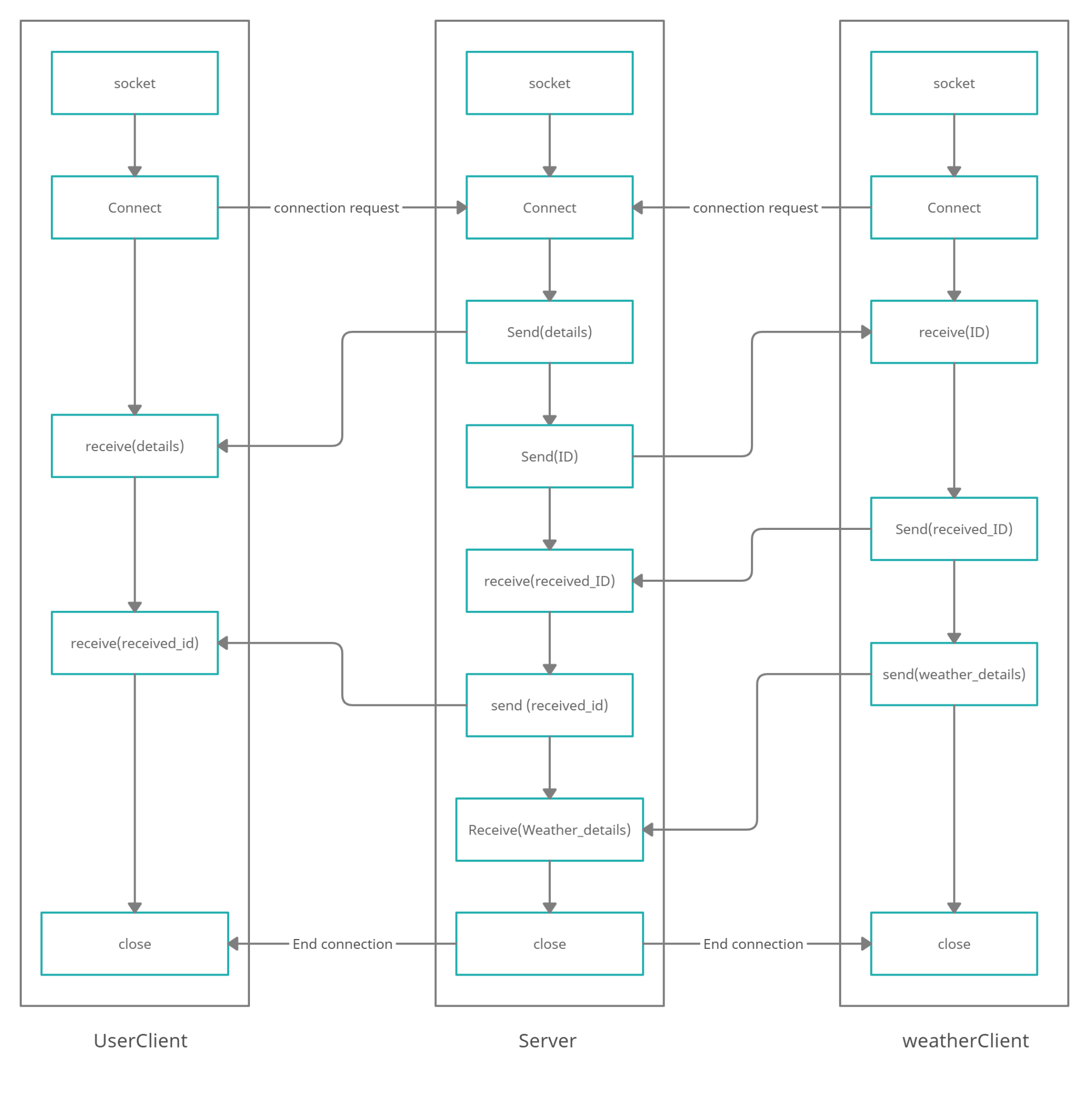
Classes:

User: the class manages the user access to the application.

TCP Server: The class launches a server and awaits for connections, once a connection is established it sends and receives data.

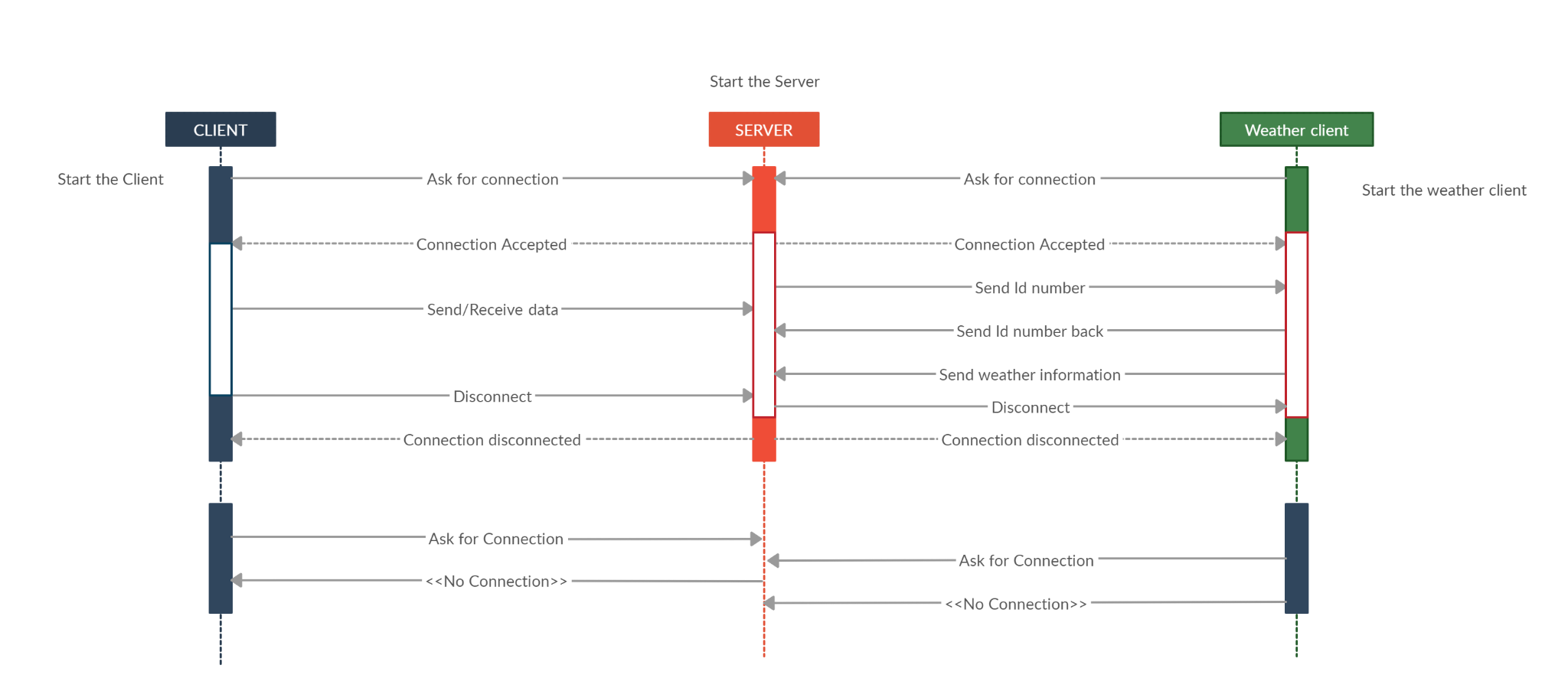
TCP userClient: The class request to connect to the server, once it is allowed it receives data from it.

TCP weatherClient: The class request to connect to the server, once it is allowed it uploads its data to it



**Data flow diagram:**

This diagram shows the flow of the data in the software, explaining how the server and clients interact to send/receive data.



**Sequence diagram:**

The diagram above shows the interaction between 2 clients and the server. The clients (user) will establish a connection through the server, which will then allow them to send/receive data.

**GUI Implementation:**

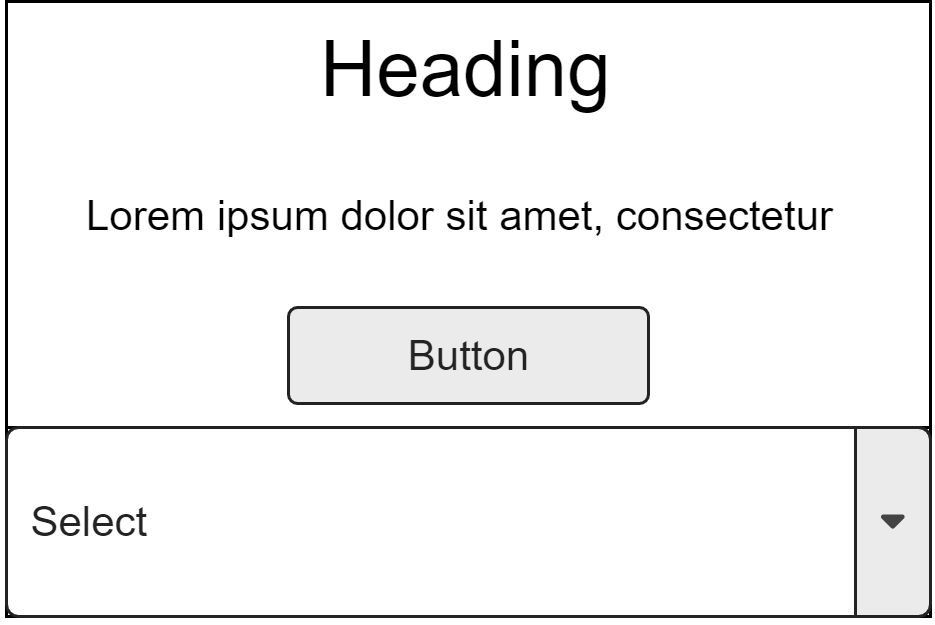
JFrame: used to display the main frame.

JPanel: used as containers.

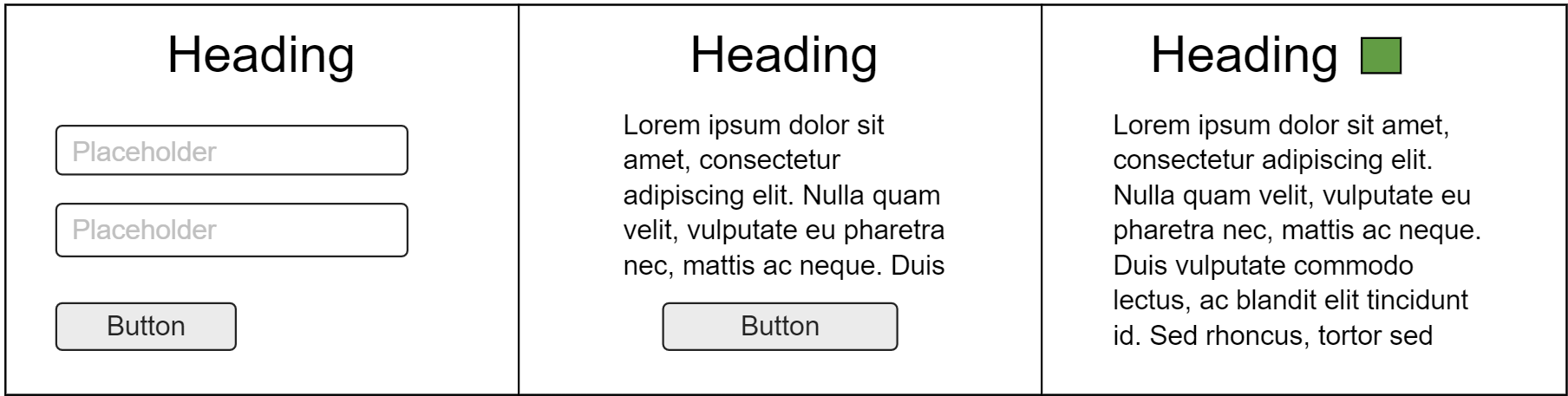
JLabel: used to display text.

JButton: used to get used choice.

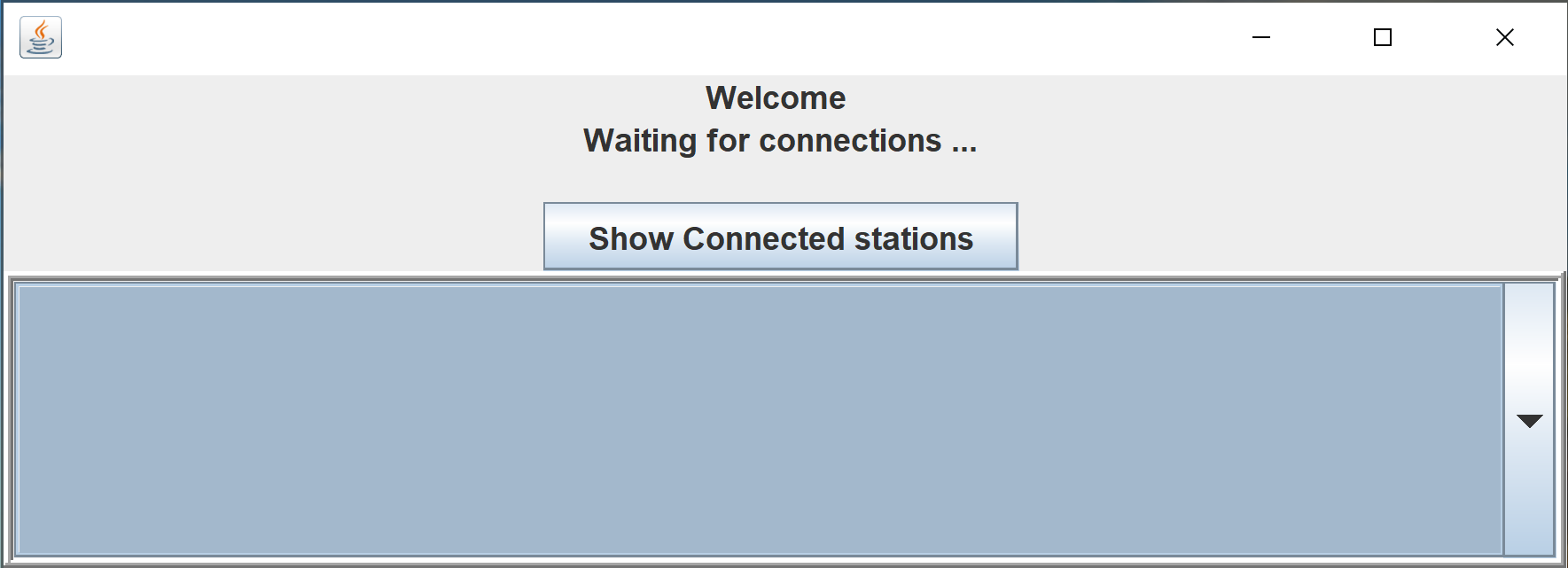
JCombobox: used to display the list of connected clients.

**GUI mockups:**

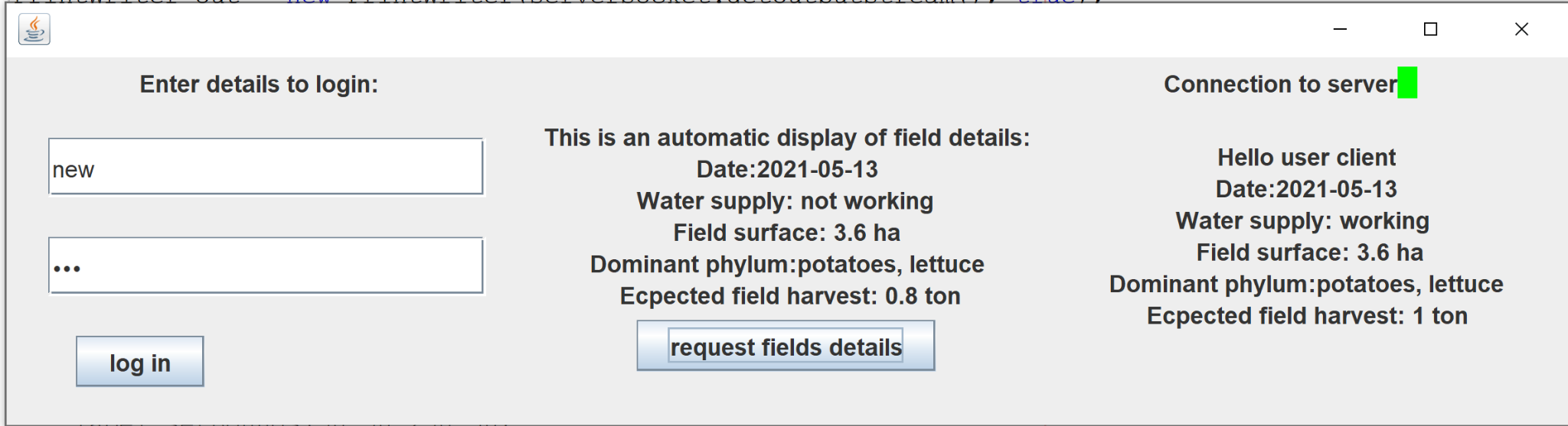
Server mock\_up.



User client mockup.

**Screenshots:**

Server GUI.



User client GUI.

**Conclusions and potential future work:**

In conclusion, the system we devised will provide a simple way of getting and storing information about the field and will make the farmer’s business easier to manage and run much more efficiently, we had for aim to make this software easy to use that’s why we have implemented a simple GUI that provides the user with all the information he needs, we have learned a lot from this project and in the list of things we would do differently is having a better risk assessment as some members of the team have been out and couldn't carry working, when it comes to the software the weather client could have been developed so it would automatically send the data to the server.

**References:**

https://docs.oracle.com/javase/tutorial/networking/datagrams/clientServer.html