2021

System Software Project Report

COMP20081

Group Members:

- Hannah Jacob 2019 (N0865554)
- Jamie Voce 2019 (N0868446)
- Nick McCaig 2019 (N0787115)
- Jonathan Archer (N0856331)
- Ryan McGouran 2018 (N0799130)

Contents

Table of figures:

Introduction

A farmer wishes to use technology to improve their business. This will be achieved using a server-client system in which the server will gather necessary information from connected weather stations and then present this information to user clients to improve the efficiency of their fields. This data being temperature, relative humidity, and the GPS coordinates of the weather stations. This central server will be responsible for extracting and storing the data from the weather stations when the system is powered up. Users will be required to create password-protected accounts that are stored securely in an encrypted database to log in to the system.

This software will aid the farmer in visualising the data that is presented to them. This will be implemented using a graph that plots the change in temperature and humidity over time. Visualisation tools like this will allow the farmer to develop a greater understanding of their fields.

Aims and Objectives.

The main aim of this group is to create the three main components of the application:

- A weather station client that collects and sends out accurate readings on temperature, barometric pressure, relative humidity, wind force etc.
- A user client which is password protected and is used to display data sent from the weather station client using a graphical user interface (GUI).
- A server that facilitates communication between the two clients mentioned above and includes a GUI of its own.

Implementation:

Implemented features:

- Secure user account validation and creation: The security of user's details is always ensured. Passwords are validated and encrypted before being written to the database.
- Unique weather station identification: Weather stations are given a randomly generated ID, if the ID matches an ID already known to the server, a new ID will be generated to ensure that all IDs are unique.
- The weather station sends data to the server.
- The server sends the data from a specific weather station to the client upon request.
- Server GUI shows all connected clients: The server uses the lists of user and weather station clients to display an up-to-date record of all connected clients. The GUI is implemented using the java swing library.
- User GUI shows the data from the weather stations.
- Graphs that plot humidity and temperature against time: The server keeps an updated log of all the data the weather stations produce so that over time a graph of humidity and temperature can be plotted.

Protocol Choice: Research

What is TCP?

The Transmission Control Protocol (TCP) is a connection-oriented communications protocol that handles the interaction between devices in a network (sdxcentral, 2021).

TCP for the most part handles the breaking down of application data into packets, packet ordering and error checking for the IP protocol. It is responsible for assembling the packets of data that have been sent over the network.

When application data is broken down into packets, the packets are numbered and are sent off to the destination address. These packets will travel in a multitude of ways and so will be received in random order. And so, the TCP protocol correctly orders the packets before they are handed off to the application, and any packets that have not arrived will be requested to be sent again (allows for error-free data transmission).

The TCP protocol maintains the network connection with the sender before the first packet is sent until after the final packet has been sent. Only once all packets have been transferred will the connection terminate (Cloudflare, 2021) (Lutkevich, 2020)**.**

What is UDP?

UDP takes application data and divides It into packets called datagrams. These are then sent out across the network. UDP does not reassemble or number the datagrams but the datagrams do include a header that includes the port numbers to help distinguish user requests. There is also an optional checksum that can be attached to the header of these datagram headers that can help verify the integrity of the data transferred (sdxcentral, 2021).

The User Datagram Protocol (Cloudflare, 2021) is used across the internet for time-sensitive events. This includes events like Video playback and DNS lookups.

The UDP accomplishes this as it is a connectionless-oriented communications protocol. It does not establish a connection before the data packets are sent out. This increases the transfer speed for the data. Essentially for two computers in a network, one can simply start sending packets straight away to the other computer with no connection being established first.

But the packets can very easily be lost in transit which will not be sent out again. Applications made to use this communication protocol must be able to tolerate errors, loss, and duplication. The unreliability in this protocol creates opportunities for exploitation (DDOS attacks).

Protocol Choice: Decision

For our application, a lot of factors have to be taken into account. For instance, weather data is constantly changing and it's always a struggle to get accurate readings. For this reason, a speedy connection could be beneficial. However, you have to consider that UDP although it has a speedy data transfer (connection speed), it is not reliable. It is prone to data loss and in some cases data corruption. For the farmers to more efficiently run their business, this data needs to be as accurate as possible without the data losses providing inaccurate results. This also applies to the login services as well, as user data would be vulnerable to these data losses and corruption. This weakness could be exploited. Not to mention the loss of personal user data goes against the data protection act. For these reasons, the best connection type to use is TCP. It is a slower connection but provides an error-free data transmission. This overall would be more beneficial than a speedy connection.

How multithreaded client handling was implemented:

To achieve multithreaded client handling, the server has two handlers, one for the user client and another for the weather client. When either client is run, a connection request will be sent to the server along with a port number that identifies the type of client that is sending the request. The server will then spawn a new thread that is tasked with handling that specific client.

User clients will not be able to attempt to connect to the server until they have first been granted authorised access to the system. This is achieved by the implementation of a registration system that utilises validation and encryption to ensure user security. The encrypted valid account details will be stored in a text file. Once the user has logged in using valid credentials, the client will automatically be granted access to the server.

Weather station clients will automatically connect to the server once their connection request is accepted and established after they are run, they will generate a unique ID which will be stored in a text file and used to identify a weather station so the user client can access the relevant data from the weather station corresponding with that ID value. For security purposes, ID’s will be randomly generated, if an ID is already taken the server will send a request to the weather station client for a new ID.
Weather stations will send their data to the server every thirty seconds, this arbitrary value was chosen because temperature and humidity were unlikely to change significantly in less than thirty seconds. The primary reason we decided to not have a wait time longer than thirty seconds was usability, farmers will want to see visual feedback that the weather stations are periodically updating otherwise they might be led to believe that part of the system is not working as expected.

Design:

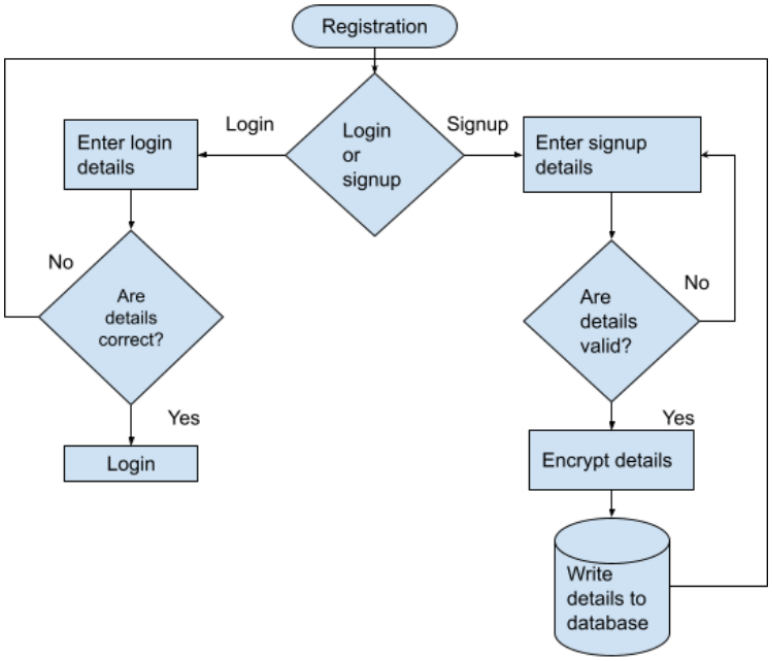

Figure 1: Flowchart describing the login process

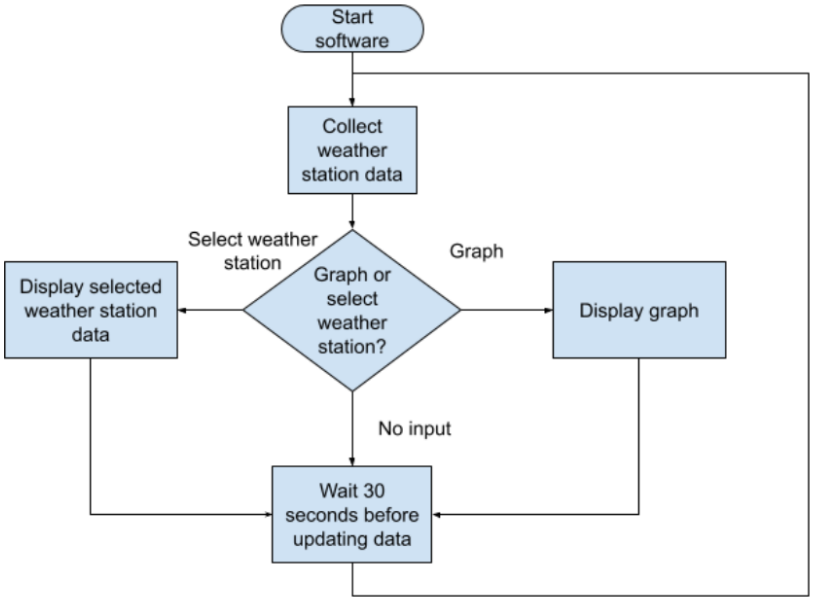

Figure 2: Flowchart describing the graph functionality

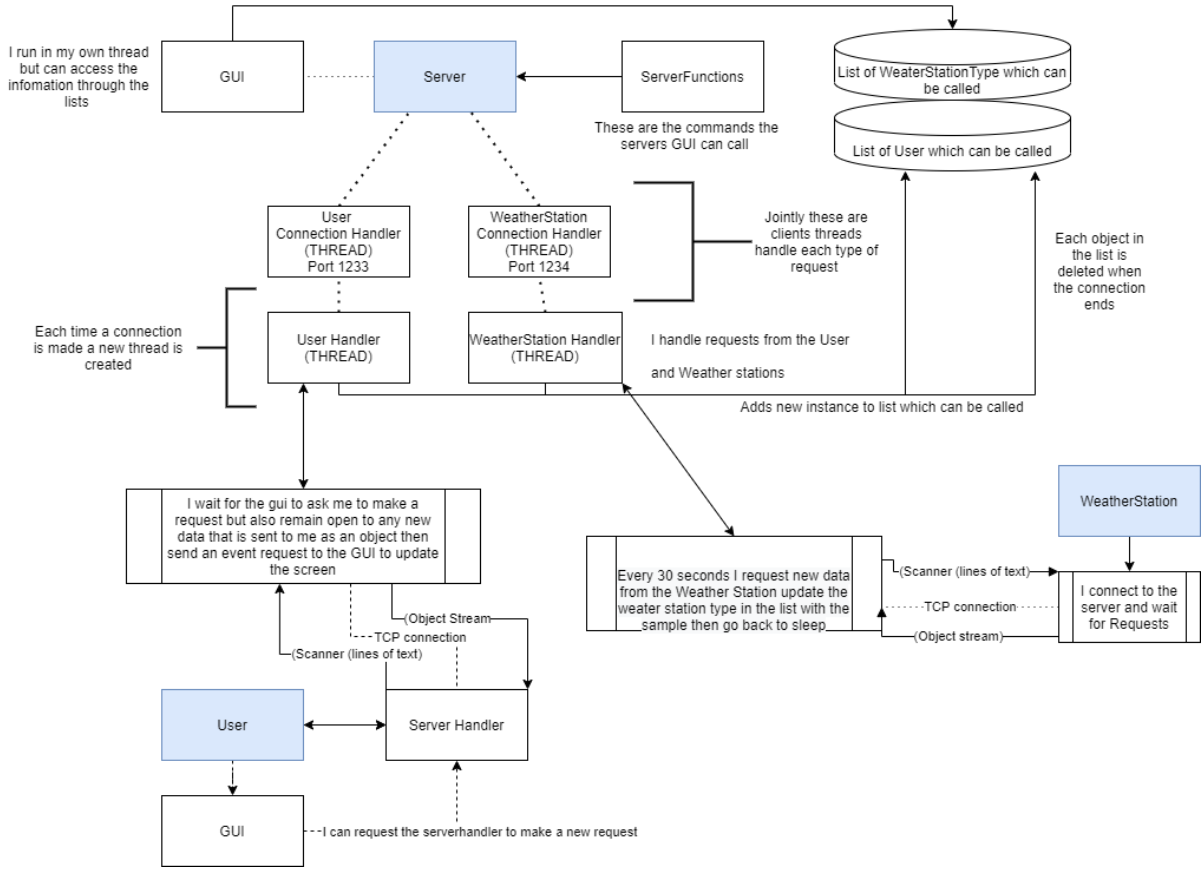

Figure 3: A flow-chart to visualise the implementation of the server-client system

Our program uses runnable multithreading, the server runs three threads One for the GUI and two threads for handling each type of client (WS/User). When a client connects to the server the connection is accepted and a new thread is created to handle connection requests. Connections have an object stream and a scanner stream. We use the scanner stream for sending requests and the object stream for receiving responses.

We follow the principle of MVC (Model, view, and controller) data is stored within distinct datatypes which are also used for communication. Our GUI (view) is always running on its own thread. It can request data and make requests to change data but following the principle of MVC, this never happens in the GUI but instead happens within the main thread of the client or the server's client handlers. The GUI can call this thread. The data stored by the server about the clients are stored in two lists one for each client type. We use a synchronized list to store data.

Connection Handling:

In our application, an open connection is used to both send and receive data between the clients and the server. This data is in the form of Object streams and printer writers. Scanners to request data via strings and objects to send the data across. The user receives the weather station data from the server via object streams and sends out requests and the username to the server via the print writers. The weather stations send out data via object streams. The data is the ID and the sample (weather) data but does not receive any. The weather stations receive requests from the server for data via the scanner. All connections are created on the fly as this is more efficient than pooling connections due

to it ensuring we only use computational resources that the application needs at any given moment instead of assuming how much the application will need from the beginning.

Data Structures:

When storing data in our program, it uses the data types defined in the Data Structures package. To store our User and Weather Station client data we use synchronized linked lists. Functions are also included such as adding a sample to a weather station and a count. The data structures are sent via the object stream in response to a request.

Weather Instruments:

The weather instruments package contains all the value generators for the weather station client. It generates a longitude/latitude value, a temperature value, a humidity value, and an altitude value. These values are generated randomly between certain ranges. The only exception to this is the Longitude and Latitude values, one of which is generated randomly (Latitude) and then the next value (Longitude) is based on the previously generated value so that they are within a radius of each other and do not exist on different parts of the planet giving the illusion of real data.

The remaining data is generated in various ranges to give the illusion of real data again but without complete randomness that can create large spikes in temperature/humidity for example. The changes are supposed to be gradual with the occasional spike/drop in the value as would occur in the real world. This is shown in the screenshot below of our application.

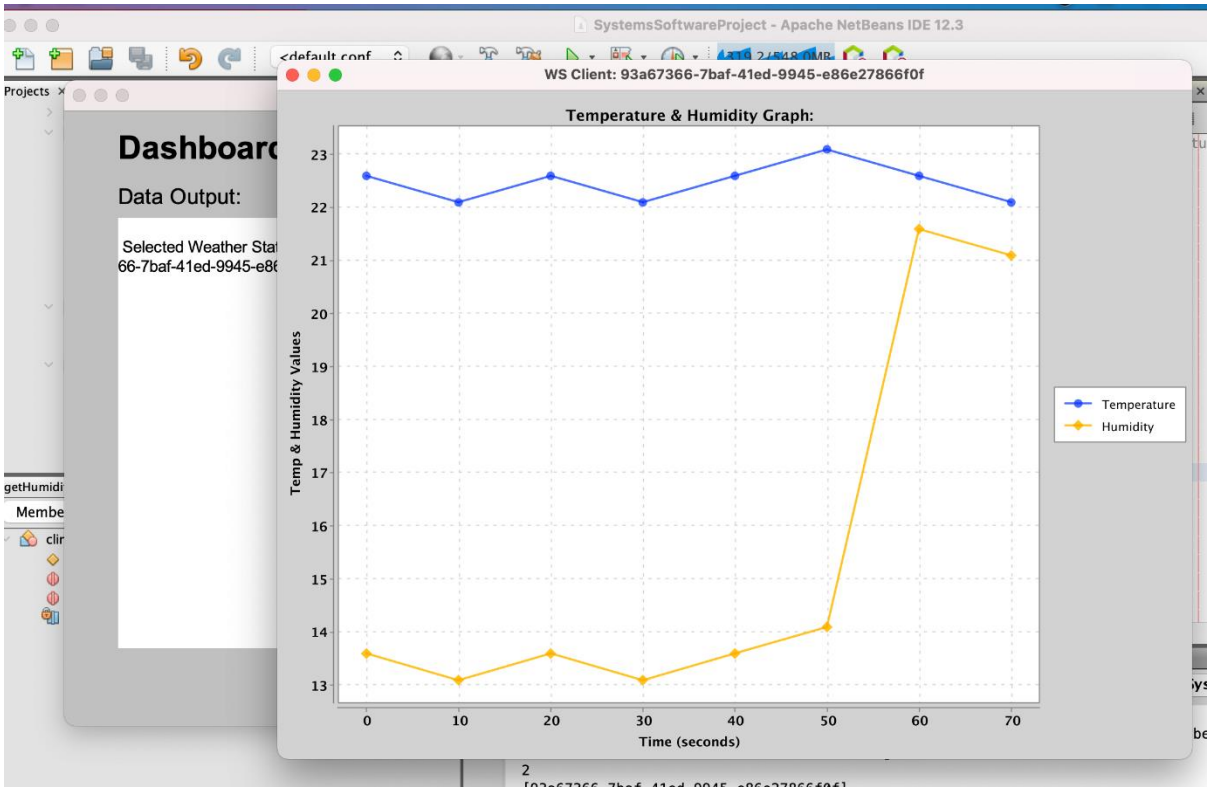

Figure 4: A screenshot of the graph generated

Enhancements:

During the initial stages of planning our team knew that we wanted to be able to display parts of a weather station's data using graphs/charts. Hence, we conducted some background research into potential libraries and found that xchart (Knowm Inc, 2021) was the ideal choice for this project. The application now charts a new graph based on the selected weather station using the sample data stored up until that moment. When the graph button is clicked again a new graph is plotted to include updates to the stored sample data. When a user switches to another weather station client and draws a graph, the graph updates to reflect the stored data of that specific weather station client instead.

Additional enhancements included account creation and credential encryption. Account creation allowed for the implemented system to be scalable. Our software could be distributed to farmers and they would each be able to create a unique account. One benefit of this feature is that farmers do not have to remember pre-defined account details and are much more likely to remember usernames and passwords that they create themselves. This reduces the likelihood that a farmer will lose access to our software.

Credential encryption ensures that accounts are always kept secure. If our software were to be distributed to farmers, it would be important that our software offers a reliable and secure service that users can trust. One such way that this achieved is by using encryption to offer farmers peace of mind that their data is kept safe.

Conclusion & Future Work:

Overall, the team believes that we have been able to successfully implement a concurrent client-server model using a TCP based network infrastructure to fulfil the project scenario's requirements and aims.

The project includes two types of clients: the user and weather station client that can communicate by sending and receiving data in the form of objects with the help of the server. In addition to this basic level of functionality, the team was able to integrate a simple graphing system to plot a weather station's temperature and humidity data on a chart for the user to be able to visualise the changes in the data over time. For this given scenario, this is especially useful as a Farmer's may wish to present this data to investors to convince them that their land is of the ideal type to suit certain crops. Hence, a future update to this application would implement the ability to download and save the graphs as image files. This can be done through the graph's java class and the user client's GUI class so that with a click of a button it triggers a function to save the current graph on the user's device.

Moving on, another feature the team would like to implement in the future is to swap out the random generation of sample data with actual sample data. To accomplish this, the team would need to make use of web scraping technology or external libraries that would allow for the asynchronous communication between our application and an actual weather station's sensor. This would in turn improve upon the functionality of the code and convert the application into one that can be applied as an actual real-world solution.

Future improvements could also be made to the credentials system. Farmers may lose their credentials at any time and if they were relying on the information that our system provides, it could be the case that the loss of this software could prove to be detrimental to the farmers business. For this reason, the team believes that an account retrieval system would be of use to the farmers. One potential implementation of this would be to request an email to be supplied along with the username and password during signup and send new passwords to the farmer's given email address if they lose their password.

In conclusion, the team is extremely proud of the project's outcome and that we have developed an application that fulfils its primary requirements in addition to including bonus features that enhance the overall user experience with the application.

References

Cloudflare, 2021. *What is TCP/IP.* [Online]
Available at: https://www.cloudflare.com/en-gb/learning/ddos/glossary/tcp-ip/
[Accessed 18 April 2021].

Cloudflare, 2021. *What is UDP.* [Online]
Available at: https://www.cloudflare.com/en-gb/learning/ddos/glossary/user-datagram-protocol-udp/
[Accessed 18 April 2021].

Knowm Inc, 2021. *XChart - Knowm.org.* [Online]
Available at: https://knowm.org/open-source/XChart/
[Accessed 23 April 2021].

Lutkevich, B., 2020. *What is TCP.* [Online]
Available at: https://searchnetworking.techtarget.com/definition/TCP
[Accessed 18 April 2021].

sdxcentral, 2021. *What is Transmission Control Protocol (TCP).* [Online]
Available at: https://www.sdxcentral.com/resources/glossary/transmission-control-protocol-tcp/
[Accessed 18 April 2021].

sdxcentral, 2021. *What is User Datagram Protocol (UDP).* [Online]
Available at: https://www.sdxcentral.com/resources/glossary/user-datagram-protocol-udp/
[Accessed 18 April 2021].