Group Members:

1. Hannah Jacob 2019 (N0865554)
2. Jamie Voce 2019 (N0868446)
3. Nick McCaig 2019 (N0787115)
4. Jonathan Archer (N0856331)
5. Ryan McGouran 2018 (N0799130)

System Software Project Report

COMP20081

2021

**Contents:**

[Introduction 2](#_Toc70020597)

[Aims and Objectives. 2](#_Toc70020598)

[Research 4](#_Toc70020599)

[What is TCP? 4](#_Toc70020600)

[What is UDP? 4](#_Toc70020601)

[Our Decision 5](#_Toc70020602)

[Design 6](#_Toc70020603)

[6](#_Toc70020604)

[Implementation: 7](#_Toc70020605)

[Implemented features: 7](#_Toc70020606)

[How multithreaded client handling was implemented. 7](#_Toc70020607)

**Figures:**

[Figure 1: A flowchart to show the implemented server-client system. 6](file:///C:\Users\Student\Documents\GitHub\Systems-Software\Report.docx#_Toc70020608)

# Introduction

A farmer wishes to use technology in order to improve their business. This will be achieved through the use of a server that will gather necessary information from connected weather stations and then present this information to workstations (managed/ used by the farmers) to improve the efficiency of the field. 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 login 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 which 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. The weather stations that collect have accurate readings on temperature, barometric pressure, relative humidity, wind force etc. The user database where login credentials are saved (most likely using textfiles). And the server side of the application that combines all these features and allows the access of these features via a GUI.

# Research

## What is TCP?

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

<https://www.sdxcentral.com/resources/glossary/transmission-control-protocol-tcp/>

TCP for the most part handles the breaking down of application data into packets, packet ordering and error checking for the IP protocol. Its 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 number and are sent off to the destination address. These packets will travel in a multitude of ways and so will be received 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.

<https://www.cloudflare.com/en-gb/learning/ddos/glossary/tcp-ip/>

<https://searchnetworking.techtarget.com/definition/TCP>

## 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.

<https://www.sdxcentral.com/resources/glossary/user-datagram-protocol-udp/>

The User Datagram Protocol 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 speeds up the transfer speed for the data. Essentially for two computers in a network, one can simply start sending packets straight away to other the computer with no connection being established first.

But the packets can very easily be lost in transit which will not be resent out. 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).

<https://www.cloudflare.com/en-gb/learning/ddos/glossary/user-datagram-protocol-udp/>

## Our Decision

For our application, a lot of factors have to be taken into account. For instance, weather data is constantly changing and its 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’s a slower connection but provides an error free data transmission. This overall would be more beneficial than a speedy connection.

# Design

## 

Figure : A flowchart to show the implemented server-client system.

Figure 1 shows the planned flow of our system. The server creates two threads, one for the user connection handler, another for the weather station connection handler. Following this, each of these threads will be able to spawn a new thread whenever the appropriate connection is made. I.e, a weather station connecting to the server will cause a weather station handler thread to spawn and a client connecting to server will cause a client handler thread to spawn.

When a new client or weather station is created a unique ID will be stored in a file by the server. This is so the server can keep an up-to-date log of all ongoing connections and display them on the server’s GUI which will run on its own thread.

Every thirty seconds, the server will request new data from the weather stations, this data will be passed to the server using a TCP connection and will utilise an object stream. When the user client requests information from the server, the server will simply send the most up to date information it has stored. Thirty seconds was chosen as the interval between new data being sent to the server as temperate and humidity was unlikely to change drastically in a short period of time.

# Implementation:

## Implemented features:

Secure user account validation and creation: The security of user’s details are ensured at all times. 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 ID’s are unique.

Weather station sends data to the server.

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.

## 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 ran, a connection request will be sent to the server along with a with a port number which 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. One 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 when they are ran, they will generate a unique ID which will be stored in a list on the server and used to identify a weather station so the user client can access the relevant data from the weather station corresponding with that ID. 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, asking for a new ID to be generated and sent via an object stream.

# 