

UNIVERSITY OF HERTFORDSHIRE
Department of Engineering and Computer Science

BSc Honours in Computer Science (Networks)
6COM1053 - Computer Science Project

Final Report
April 2022

The Peer-to-Peer Encrypted File Sharing Service

Subhan Rehman 18024473
Supervised by: Dr Tazeen Syed

Abstract

This project aims to create a solution to the ever-increasing difficulties surrounding the transfer of exceptionally large files from host to recipient without using the services of a third-party company. Files we deal with get larger every day with higher resolution videos, and incredibly high bitrates now available to people in their own homes, transferring these large files around can prove to be cumbersome and expensive, often incurring costs from cloud storage solutions to transfer the data.

In this report I explore the options available to users to transfer their files, and I develop a solution of my own which uses peer-to-peer networks to expedite transfer speeds while avoiding the costs associated with utilising the storage space of a server and implements encryption to keep the data secure in transit. The report includes the research I conducted, as well as the methods used to implement my solution and create my artefact.

The report further includes testing and evaluation of my solution, the project as a whole and the learning achievements made throughout it all.

Acknowledgements

I would like to thank my project supervisor Dr Tazeen Syed for her continued support and help with my project. Her assistance has been invaluable, and I am incredibly grateful for the time she has sacrificed for not only this project, but as a referee so that I may continue my studies in the field of computer science.

Contents

Abstract.....	1
Acknowledgements.....	3
1.0 Introduction	6
1.1 Purpose of Project.....	6
1.2 Motivation for Project.....	6
1.3 Success Criteria and Objectives	6
1.3.1 Learning Objectives.....	7
1.3.2 Solution Success Criteria	8
1.4 Software Development Methodology	10
1.5 Report Structure	10
1.6 Gantt Chart.....	10
2.0 Background Research.....	11
2.1 Researching C#.....	11
2.2 Researching Peer to Peer Networks	11
2.3 Researching File Transfer Over Networks.....	12
2.4 Researching Network Protocols.....	12
2.5 Researching End to End Encryption	13
2.6 NAT Routers	14
2.7 Port Forwarding	15
3.0 Software Design and Implementation Plan	16
3.1 Proposed UI.....	16
3.2 Implementation Plan.....	17
3.3 Risk Assessment	17
3.3.1 Risk – During Implementation	18
3.3.2 Risk – Due to Artefact	20
3.4 Legal Ethical Social and Professional Implications.....	22
4.0 Implementation	23
4.1 Implementation Outline and Approach.....	23
4.2 C# Visual Studio and Windows Forms.....	23
4.3 Preliminary Application Layout and Navigation.....	23
4.4 Sending Files to Another System	26
4.4.1 FTP Connection Test.....	26

4.4.2 SFTP Connection Test.....	32
4.5 Receiving Files from Another System	34
4.5.1 Receiving Files from Within the Local Area Network	34
4.5.2 Receiving Files from Outside the Local Area Network.....	42
4.6 Fixing Bugs and Improving Code	45
4.6.1 Improving Code and UI	45
4.6.2 Fixing Bugs and Addressing Security Concern	46
4.7 Application Test of Core Functionality.....	48
4.8 Implementation of Additional Features.....	52
4.8.1 Implementation of Speed Control	52
4.8.2 Implementation of Progress Bar	52
4.8.3 Send a File Final Revision	54
4.8.4 Receive a File Final Revision.....	54
5.0 Testing.....	55
5.1 Testing Outline	55
5.2 Testing Against Success Criteria.....	55
5.3 Testing Against Nielsen's Heuristics.....	59
6.0 Evaluation	61
6.1 Evaluation of my Performance and Approach	61
6.2 Evaluation of the Success of my Learning Objectives	61
6.3 Future Work and Further Development of my Project.....	63
7.0 Conclusion.....	65
8.0 References	66
9.0 Appendix	68
9.1 Push Notification Chart.....	68
9.2 Router Configuration	68
9.3 Gannt Chart.....	69

1.0 Introduction

1.1 Purpose of Project

The purpose of this project is to develop a file sharing application that utilises peer-to-peer (P2P) networking and encryption to ensure a secure, fast and reliable transfer of data from the host to the recipient. The use of P2P allows the application not to require dedicated servers in order to run and transfer files to another system running the application connected to the internet. This also allows files to be sent of any size or file type allowing for more freedom when transferring. The use of encryption ensures that the data remains unreadable until it has reached the recipient, this ensures that in the event of data interception the data is not compromised.

1.2 Motivation for Project

My motivation for this project is to solve a problem that I have encountered in my daily life, the transfer of large and / or sensitive files from one computer to another while circumventing the costs, size and file type restriction associated with using third party servers and services such as Google Drive or Email. This is a problem that came about when attempting to transfer large media files between me and my friend; without paying for cloud storage there was little way for me and my friend to transfer the files to one another. To solve this issue, I believe there are three tasks I must achieve to solve the problem. To establish a peer-to-peer connection between the host and recipient systems, achieve a reliable file transfer over the connection and ensure that the connection is encrypted making it unreadable along its journey. With these three, the core problem will be addressed and solved.

I believe that this project will not only solve the problem that I have encountered, but also help me in my development in computer science. The research into the technologies to be used and the hurdles I may encounter will greatly aid my knowledge in networking as will my work into establishing a secure connection. I expect my experience and knowledge in cyber security to increase through the research and implementation of encryption and data security practices throughout the duration of the project.

1.3 Success Criteria and Objectives

I have created a list of success criteria and objectives in order to measure the success of the solution and of the project as a whole. These contain a list of requirements that must be met to solve the problem I encountered as well as learning objectives for my personal development.

1.3.1 Learning Objectives

Objective	Description	Justification	Method
Expand knowledge on establishing network connections	To increase my knowledge on networks and file transfer over networks as well as peer-to-peer network connections	This will aid me with the implementation of my artefact as I will be able to better understand the peer-to-peer connection I intend to establish as well as the stability of that connection.	I will conduct research into network connections and document my findings to aid me with better understanding how to implement them.
Expand knowledge on network protocols	To increase my knowledge on network protocols that may be useful within the implementation	This will aid me with the implementation of my artefact as I will be able to better understand the several types of network protocols that are used and the ones I intend to use so that I may utilise them in my solution effectively and correctly. This will facilitate file transfer over the established connection	I will conduct research into network protocols and document my findings, comparing different protocols and rationalising my reason for the ones I will choose in my solution.
Expand knowledge on encryption	To increase my knowledge on encryption in how it works and the ways I can implement it within my artefact	To understand and learn about different types of encryptions and the ways I can implement it in order to ensure that all communications are encrypted in my artefact ensuring security.	I will conduct research into different types and methods of encryption as well as the correct usage and implementation of the technology to ensure my use of the technology is safe and secure.
Expand knowledge on C#	To increase my knowledge on the programming language	As my solution is to be coded in C# it is imperative that I have an exceptional understanding of the language as this will allow me to code the solution as efficiently as possible.	I will practise basic programming with the language as well as conducting research into the language to familiarise myself to a point of near fluency to avoid needless problems when coding my artefact.

1.3.2 Solution Success Criteria

Objective	Description	Justification	Method	Importance
Transfer a file from the host system to the recipient system securely and without utilising a central server	The success of this will mean the artefact will be a suitable solution to the problem the project aims to solve.	The success of this criterion fulfils the three main requirements outlined in section 1.2 therefore signalling complete success of the artefact.	This will be achieved through the successful completion of the other success criteria outlined	This is especially important to achieve as without it there will be an incomplete solution to the problem.
Establish an initial peer-to-peer connection between the two systems	This will require a peer-to-peer connection to be established between the host system and the recipient system without a central server	The problem revolves around the issues associated with the use of central servers therefore their use would be counterproductive to the problem the project is addressing	This will be achieved within the implementation stage aided by the research conducted in the next stage of the project	This is critical to the success of the artefact as it is a piece of core functionality that will allow the artefact to work
Establish a stable connection capable of file transfer between the two systems	This will require an existing peer-to-peer connection to remain stable when used for file transfer	As the project focuses heavily on the success of file transfer and not a video delivery system for example. It is imperative that the data arrive intact and complete as missing bits can cause corruption or data loss.	Aided by my research on peer-to-peer networks and on network protocols I hope to implement this feature once a stable connection has been established	This is very important as a failure to achieve this will create a solution that underdelivers at best and causes issues for parties involved at worse due to data loss.
Create a stable application	This will require a solid foundational knowledge in C# and software development.	This is important as a stable bug free application will increase the user experience as well as ensuring that the application is able to complete a file transfer successfully without the risk of failure during a transfer.	I expect this to be achieved after many iterations of stress testing the application and exploring all possible ways a normal user or an inquisitive one may try to cause application instability.	This is important as an unstable application may lead to unsuccessful transfers nullifying my other efforts and not delivering a working solution

Objective	Description	Justification	Method	Importance
Establish an encrypted connection between the two systems	This will require an existing connection between the two systems and a solid foundational knowledge of encryption to implement.	Encryption will ensure that all data being transferred between the two systems is unreadable to anyone who may intercept the data in transmission	This should be implemented as part of the protocols being used to send the file from one system to another. If for whatever reason those secure protocols cannot be used, then a method to encrypt and decrypt files will have to be designed and implemented.	This is important as un-encrypted transmissions are a security risk and anyone who intercepts the communication will be able to see the data that is being sent between the two systems.
Implement additional features to increase the quality of the user experience.	This will be implemented after the rest of the core functionality has been successfully deployed	User experience and ease of application use will serve to increase the speed that the application can be used as well as increase its utility	This will be implemented after the artefact has solved the project problem as allocating development time to non-core parts of the software should only be done once the core has been developed to ensure that the most important aspects are completed on time	This is of least importance as the success or failure of this will have little bearing on whether or not the artefact developed was sufficient to be considered a suitable solution to the problem.

1.4 Software Development Methodology

My chosen software development methodology is a type of Agile methodology focused on the delivery of features through an incremental and iterative process. This “Feature-Driven Development” (FDD) (Lynn, No Date) methodology is geared towards delivering results on a frequent basis as well as encouraging documentation. I have chosen this particular methodology as my aims and goals have already been clearly defined and using FDD will provide a “simple but comprehensive methodology.” (Lynn, No Date) I have a clear hierarchy of features that I believe must be created in order to achieve project success (successful connection established, successful file transfer and successful encryption implemented) that I believe this methodology will allow me to develop along these lines iterating my code and releasing new features to my software on each cycle.

FDD does have its downsides, it is intended for use in “long-term, complex projects” (Lynn, No Date) and is often cited as not being efficient for “smaller projects” (Project Practical Editorial Team, No Date) or where there is “one developer” (Project Practical Editorial Team, No Date) due to the difficulty of the developer needing to perform multiple duties at once. However, due to small nature of the project, all goals and aims being already defined, a clear success criteria being outlined and no client urgently waiting for my next software release; I believe I can make FDD work to my strengths and take advantage of the benefits it provides. This will be done by delivering the core features incrementally and slowly working outwards to less critical features after the core has been successfully developed.

1.5 Report Structure

This report will now analyse research on key topics outlined within the learning objectives section 1.3.1. The justification for this is to gain more knowledge and a deeper understanding into these topics as they will help with not only the coding of the solution but with the implementation of network protocols and encryption that will need to be used in order to ensure a successful artefact is created.

After the research section has been concluded and conclusions have been drawn the development of the artefact will begin using the FDD methodology. The artefact will then be tested against the success criteria and the solution created as well as the project itself will be evaluated. Lastly, a final conclusion will be drawn which will provide a definitive statement judging the success of the artefact, future work that may be done and what I have learnt and taken away from this project.

1.6 Gantt Chart

To assist me with my planning and time management I have created a Gantt Chart (See Appendix 9.1). Within the chart will be the timeline of the tasks I wish to do when I would like them completed and what will run concurrent to them. It is my hope that the use of chart will improve the quality of my work as I will be taking several opportunities to evaluate my work, test throughout my development and use that information to plan my next step all the while ensuring I stay within my deadlines.

2.0 Background Research

2.1 Researching C#

C# is a programming language developed by Microsoft and is an object-oriented language. It is the “major technology for creating desktop applications on Windows” (Altexsoft, 2021) and is one of the most popular programming languages in the world. I feel it would be suitable to code my artefact in C# as it is a language developed by Microsoft specifically for software app development.

I currently have some limited experience working with C# from my 2nd year C Family module where I gained some coding practise. However, I am familiar with the C family of languages and object-oriented programming so understanding the concepts and syntax should not be an issue that will be encountered in the development stage. Finally, I feel that the integration with Windows that C# has due to its developers being Microsoft as well as the included IDE makes it a compelling choice for use in developing my lightweight application.

2.2 Researching Peer to Peer Networks

In order to send a file from a host system to a recipient system without paying for third party servers or hosting on free services, a direct connection must be established from the host to the recipient. This is the only way to directly send files of an unlimited size or of any nature across the internet as any need to worry about file restrictions associated with using services such as Google Drive or Dropbox are eliminated, nor is there any need to incur a fee to host a file on a server.

This type of a connection is called a peer to peer (p2p) network and is the type of connection I intend to utilise for my file sharing application for the following reasons.

Cost: The benefits of a p2p network are that they are “relatively inexpensive” (Roomi, 2020) and that due to no requirement to have a dedicated server it can lead to “saving more overhead costs” (Roomi, 2020). This is appealing to myself as it allows me to code my artefact and use it as efficiently as possible.

Reliability: As a p2p network does not require a central server, issues such as server outages and maintenance will not affect a host’s ability to communicate with the recipient system. This is a positive side-effect of not having to rely on the uptime of another system which is beneficial to me as it will greatly improve the success rate of transfers.

Ease of Implementation and Administration: Due to the lack of a central server to configure and setup a p2p network is much easier to initially deploy and much faster allowing for my artefact to be installed and used faster. Furthermore, due to the lack of a server there is no need to maintain the server or administrate it meaning that the artefact will not require future intervention or effort.

There are however some drawbacks to p2p networks which I will now discuss. A server is typically more likely to be online than a PC. A dedicated server in a data centre is far more likely to be online than a normal home users PC. This combined with the average UK upload speed of approximately 10 Mbps (Ofcom, 2021) being over 5 times slower than the average download speed of approximately 50 Mbps (Ofcom, 2021) means that a serious bottleneck will be present on average for download speed when compared to data centres where the total throughput can be several terabits. Despite this bottleneck issue I believe that the time taken for the host to upload the file to the central server in a p2p system will

be exactly equal to the time taken to download the file as it is being done simultaneously (provided the download speed is as fast as the upload). This means there is no need to wait for the host to upload before the recipient download begins. A second reason why I feel this bottleneck can be overlooked is due to the fact that the data is being sent directly from one system to another; this increases security as the encrypted data is less likely to be intercepted and stored on another system which may prove to serve as a benefit that is worth losing bandwidth over.

An unavoidable characteristic of a decentralised p2p system which has encryption baked into it means that the ability to monitor communications between users is near impossible. While this can be considered a good thing and in my view is indeed a good thing for reasons such as security, speed and ease of use. The issue associated is that this can lead to malicious or illegal communications being sent, such as “copyrighted contents like movies and music” (Roomi, 2020) using the artefact. Torrenting services are infamous for this and is a problem that has not been solved since their inception.⁶ In my view, users are responsible for the content they distribute, and the design of my artefact means that there will need to be explicit consent between the host and recipient to send the file to one another. My artefact also does not have a public nature to it therefore any possible unsavoury communications will require the consent and cooperation of both parties and as a result is the decision and responsibility of those parties. Therefore, it is not my responsibility to compromise the security of my system to police the communications through my artefact.

2.3 Researching File Transfer Over Networks

File transfer over networks is a ground-breaking achievement and has nullified the need to send data in a physical form via drives at an increasingly faster rate as technology progresses daily. File transfer is incredibly important to the point where “more than 50 percent of all systems integration is done through file transfer” (IBM, no date)

The ability to communicate files over wider networks and not just local ones is imperative to my solution and there exists several ways to achieve this with the best being the Secure File Transfer Protocol (SFTP) which I will discuss in more detail in 2.4. As files become increasingly larger and increasingly more sensitive data is being sent online as opposed to more traditional methods such as post, a secure file delivery system is needed to ensure that data reaches its intended destination with IBM stressing the importance to a point where when file transfer fails “your organization is grounded” (IBM, no date) My artefact will provide a solution to aid in file transfer for any two parties that need to communicate data rapidly and securely and it is my hope that my project will enhance my data in networks and cyber security throughout this process.

2.4 Researching Network Protocols

In order to facilitate the data transfer over the established connection between the host and recipient there will be a requirement to use network protocols in order to achieve this. A network protocol that can handle file transfer will either use the Transmission Control Protocol (TCP) or the User Datagram Protocol (UDP) to send the files from host to recipient. These are the two most popular types of protocols used today with each one serving a different purpose.

UDP is considered to be “simpler and faster” (Bischoff, 2019) than TCP which are attributes that are incredibly attractive for use in my artefact as it will help with the speed of data transfer and the ease of use to set up the connection. However, UDP has a significant drawback, it “doesn’t require the recipient

to acknowledge that each packet has been received” (Bischoff, 2019) nor does it require that “any packets that get lost in transit are not resent.” (Bischoff, 2019) This is a significant issue with UDP as if used during a file transfer in a network system that cannot guarantee a 100% reliability rate when it comes to the success of packets reaching the intended location it will lead to data loss and corruption as the data will in all likelihood not have reached the intended recipient without failure.

This is where the speed and simplicity of UDP is sacrificed for the reliability offered from the TCP protocol and is the reason it is so widely used across the internet. Where UDP does not require acknowledgment “TCP guarantees the recipient will receive packets” (Bischoff, 2019) through the recipient sending a message back with “each packet, acknowledging that they’ve been received.” (Bischoff, 2019) As a result of this TCP will be the choice of my artefact and I will be using protocols that rely upon it.

“In 1985, the first communications protocol, FTP, was established” (IBM, no date) and it is still supported and used to this day. FTP is an incredibly powerful protocol whereby only needing an IP address and a port, data can be sent and retrieved from another system. This protocol utilises TCP to send files across the internet providing a reliable data transfer. The issue with FTP is that it is completely unencrypted, this will allow any actor intercepting the data to be able to view the data without having to decrypt it. This makes FTP’s use much less compelling as the data is insecure.

FTPS was proposed as a solution to this adding a layer of encryption over the protocol securing the data transfer and proved a success. However, it is now prohibited to be used by internet standards due the encryption used ‘SSLv3’ (IETF, 2015) being not “sufficiently secure” (IETF, 2015). This protocol was going to be my initial choice before my research uncovered this issue.

The protocol I intend to use is SFTP which utilises the Secure Shell protocol (SSH) which works by using public and private keys in order to encrypt the data in a way that cannot be read even if every step of the communication is intercepted. The SFTP protocol is different to the first two protocols mentioned because it is not built on the foundation of FTP but rather it is its own protocol built from the ground up by the Internet Engineering Task Force (Secure Shell Working Group, 2006) in an attempt to create a secure file transfer protocol.

2.5 Researching End to End Encryption

As discussed in the previous section the protocol I intend to use utilises end to end encryption as a basis for its operation and use. I will discuss the merits and demerits of using encryption in this section to properly rationalise and explore the topic and why it is suitable and necessary for my artefact to have it included.

Encryption has many obvious benefits, and it is small wonder why it is used for almost all communications conducted today. The biggest and most obvious benefit is privacy, the data being transferred is visible only to you and the system on the other end, this provides users with a peace of mind knowing that “data is protected from outside viewers” (Hiter, 2021). This is something that has a benefit for my solution as when transferring data from one system to another it may be of a sensitive nature and as such it may not be intended for anyone apart from the recipient.

Encryption also has use within security of the systems involved as it can prevent man in the middle attacks due to an attacker being unable to modify data if the data they are trying to modify is unreadable therefore maintaining the “integrity of data” (Hiter, 2021). This is especially important with p2p

connections as there is not central server to verify and check the validity of the data and is another reason why I will be using it within my solution. There are however some drawbacks to encryption which I will now discuss.

If encrypted data is sent to the proper recipient but for whatever reason the recipient no longer possesses the correct decryption keys for the data, it is unreadable despite the fact the user may be able to validate they are the intended recipient. This can cause data loss in situations where decryption keys are lost to data rendering it useless despite the data itself being safe in storage. While this is a risk with storing encrypted files and documents my artefact will decrypt files upon receipt and will not require any keys to be remembered by the user eliminating human error.

Encryption is not impenetrable and can be broken or circumvented. If encrypted data is intercepted it may be unreadable to the attacker at that moment but because they now have the data in their possession it means that over time, it is feasible that they may gain access to it. Modern encryption algorithms are strong enough that charitable estimates claim that testing every possibility for a basic encryption would take 10.79 quintillion years to exhaust the list. (Wood, 2011). However, technology progresses fast, and brute force is not the only tool an attacker has, if an attacker gets access to a transmission be it encrypted or not, there is always a risk of it being compromised. It is my belief that current encryption systems are strong enough that the risk of any data being read is incredibly remote and there is always an accepted risk when using any network connected to the wider internet and in the current day, that risk is insignificant compared to the advantages encryption provides.

2.6 NAT Routers

When connecting to FTP and SFTP servers, set addresses and set port numbers are entered into the application to establish the connection. The ports 21 and 22 are used respectively for these as they are “well-known, industry-standard ports” (Apple, 2021) and are the same ports every single time a connection attempt is typically made to the server. By using the given credentials plus the port number, a server is able to correctly verify the authenticity of the connection and it is able to be easily established. Unfortunately, this is not the case elsewhere and means that a similar connection to my application may not be possible.

The issue lies with how routers connect the devices connected to them, to the internet. Routers typically have one public IP address that all devices on the network share. This is the public IP address and is the address used by all systems external to the network to connect to the network. The system that the routers use to connect the local IP addresses of the individual devices to one single external IP address is known as NAT. “NAT was originally developed as an interim solution to combat IPv4 address depletion” (Phifer, 2000) and is still in use today among all modern routers. NAT is able to take a request from an internal system out of the network and to then correctly identify that the response externally is for the device that sent the initial request. From an outsider’s, or more typically a server’s, point of view, they are simply sending packets to the external IP that requested them, it is the routers job to send those packets to the correct device.

The problem is that this does not work in reverse for NAT systems, the devices behind the router must make the first request as the external devices have no idea what lies behind the NAT system or the devices local IP. This is how all instant messaging and notifications work, our devices are constantly asking the servers for an update for a new notification, if successful then the server will reply. The image showing

how the mobile push notification service works illustrates this (See Appendix 9.1), it is the device that must make the request.

This means that in the case of my application, I will only have knowledge of the external IP address of the recipient as local ones cannot be used outside of the network. I could have the recipient device send a request to the host device, unfortunately as I am not using any servers to facilitate the transfer due to it being P2P this will not be possible as the host device will also be behind an NAT router.

2.7 Port Forwarding

Another method to establish a connection involves connecting directly to a port, this allows an external device to connect to a router's open ports which refer to a specific device's running service for that port in order to establish the connection. On a router that has not been configured, as is the case for the majority of residential routers, the IP and port numbers assigned to devices is not set in stone. An automated service known as the Dynamic Host Configuration Protocol (DHCP) is responsible for assigning IP addresses to all devices connected to a router and has the benefit of convenience by bringing "plug-and-play operation to the Internet." (Droms, 1999) This is an incredibly useful service allowing devices to get up and online very quickly while removing the need for users to configure their devices or their routers. The problem however is that IP addresses are assigned randomly and can be unassigned automatically meaning that the same local IP address or port number may be a different device every time a connection is attempted.

Therefore, to connect directly to a port, a mechanism known as port forwarding must be used to allow external connections to go directly to the device behind the NAT router. This requires the IP allocation of the device to be static and any existing DHCP service must either be disabled or leave the IP address reserved for the device that must be left static. However, this step is not enough as an external user still has no way of identifying a particular device running a particular service. This is where port forwarding is used by informing the router "that we are trying to access it through a program, and it needs to send the program to a device connected at a particular port." (Verma, Kashyap and Jha, 2018)

As you can see above my router has been manually configured in the topmost entry to accept a specific range of ports externally to then reserve for the device at 192.168.0.31. Any request to the router between 52000 – 53000 will go directly to the device between 50000 – 51000. If the device is listening on those ports, it will accept the request and reply.

The issue with port forwarding however, is that it requires access to a router's settings, these are almost always password protected and cannot be changed by anyone apart from the network administrator. This means that my application will unfortunately be limited to situations where someone has access to the router and has the technical knowledge to modify the settings to allow port forwarding or has pre-configured the router to allow port forwarding to my application. While this is unfortunate it is the only approach that ensures that the host and recipient have a direct connection without a central server and therefore maintains the security albeit at the cost of a more complex setup.

Therefore, the upside is that as port forwarding does exist, and I have access to my own router which I pay for, I should be able to successfully continue the implementation and develop my artefact to the stage where it can receive files.

3.0 Software Design and Implementation Plan

3.1 Proposed UI

I have decided that the UI for my artefact should be in line with industry standards and provide the user with a sense of familiarity, therefore I have opted to follow Nielsen's 10 heuristics for UI design (Nielsen, 2020) as they will ensure that the UI I develop will be as accessible and easy to use as possible providing an excellent user experience.

In particular, I will be focusing on #5 “error prevention” (Nielsen, 2020) and #9 “help users recognize, diagnose, and recover from errors” (Nielsen, 2020) as without these two the artefact will be unpleasant and unfriendly to use. Once all core features have been developed, I intend to create a help function within the program and to include features within the program that ensure the users of the program are not forced to re-enter information they have already entered to increase the speed and ease of use.

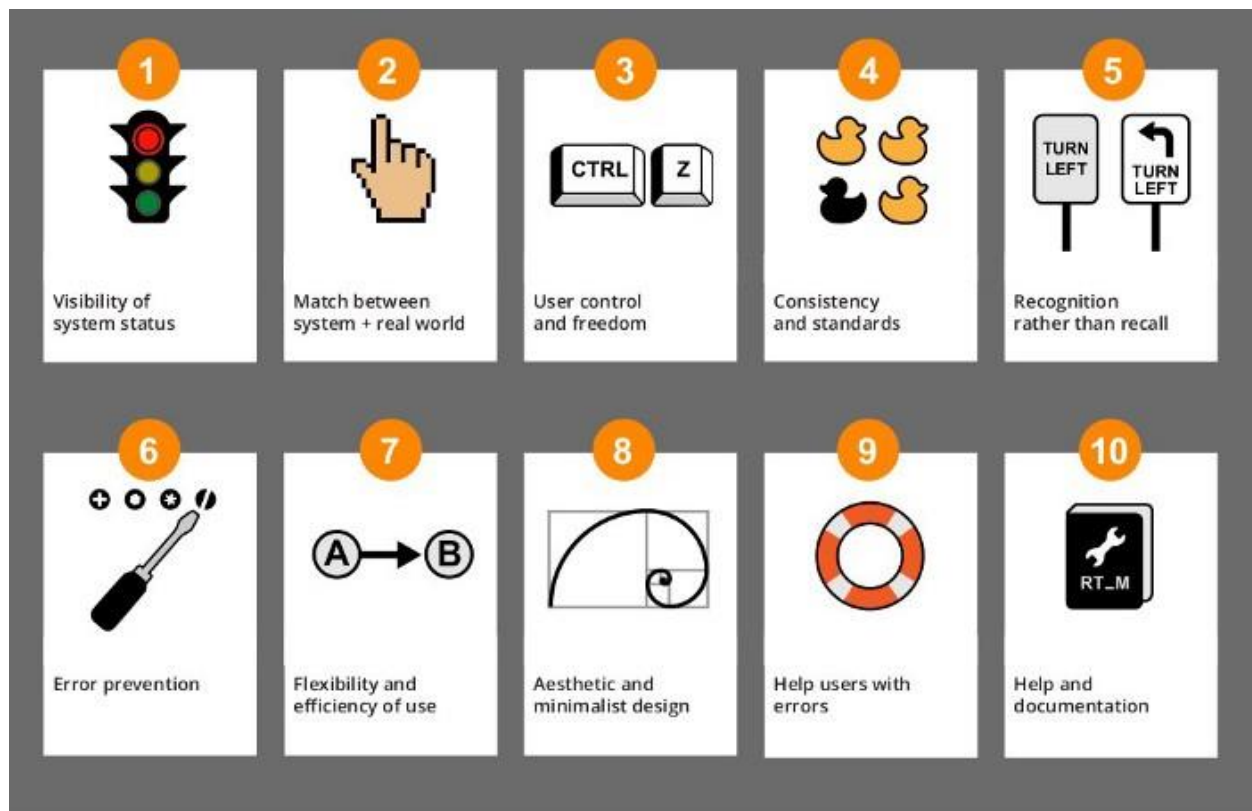


Figure 1 - Nielsen's 10 Usability Heuristics for UI Design (Putra, 2020)

3.2 Implementation Plan

I will be using feature driven development as my software development methodology of choice throughout the creation of my artefact. This will allow prioritisation of my success criteria which has been developed in order to solve the problem this project aims to address. Therefore, the development of my solution will consist of building a barebones application and adding features to it one at a time, testing and iterating as I do so. Prioritisation of aspects such as Nielsen's heuristics and extra features I would like to add being left until after the core features have been successfully implemented.

I will be using Visual Studio for my IDE as it was developed by Microsoft and has support for coding in C# and for creating Windows Forms applications which is what I will be using for the development of my artefact. I have chosen to use Windows Forms over the newer and more feature rich Windows Presentation Foundation (WPF) UI framework for several reasons.

Firstly, my application is intended to be a lightweight tool to facilitate the fast transfer of files and as such it is my intention to prioritise function over form and I simply have no use for the "rich, interactive, animated, hardware accelerated, vector 2D and 3D capabilities" (Hammad, 2021) that WPF has to offer opposed to Windows Forms. WPF uses more RAM than the older Windows Forms which has a "smaller memory footprint" (Hammad, 2021) which is desirable as it will mean the application is not resource intensive. Lastly, Windows Forms is also "less time-consuming and less tricky" (Hammad, 2021) to develop with compared to WPF allowing my feature driven development approach to be more efficient and successful.

I will begin the implementation by installing all prerequisite software I will require; this will include Visual Studio and C# libraries. Any additional libraries will be added later if relevant. From here I will develop a barebones UI so that the feature driven development has a foundation on which to develop from. After creating a basic UI and navigation features, I will then focus my efforts into establishing a connection with another system, I will firstly aim to implement the ability to send files unencrypted, and then send files encrypted. After this development has completed, I will then continue my development to configure my artefact to accept files from within the local area network. Once the core functionality has been established as well as the capability to handle files will I then focus my efforts on enabling file receipt from a system outside of the local area network. The next step will be to test the two main features of my application together to see if one instance of the application can send files to another, on another network in an encrypted manner. The success of this will be the milestone to declare all functionality present and the core problem addressed. I will then conduct several tests to ensure the success criteria I have created has been met and that all tests ran against the artefact have provided satisfactory results. Finally, I will go about refining the application and adding additional features. These features will mainly focus on usability making the solution as easy and accessible as possible, this stage will also be used to iron out bugs and to develop a more polished solution.

3.3 Risk Assessment

There are two main events within the project lifecycle where risks can occur, that is during implementation, and after implementation due to the artefact. Therefore, I have created two comprehensive risk assessments which detail all possible risks that may be encountered alongside mitigation and effects of the risks. With a better understanding of the risks associated with the undertaking of the project implementation, it will allow me to factor in every possibility when making critical decisions such as data backups or when allowing for problems outside of my control.

3.3.1 Risk – During Implementation

Risk	Likelihood	Impact Severity	Impact	Mitigation
Internet Outages	Very Low	High	This will lead to an inability to test the purpose for which my application was developed.	The internet I pay for is reliable. In the event of an outage, I have other ways to get online. Such as the university, relatives, and mobile networks.
Power Outages	Very Low	Severe	This will prevent all development and progress of my project and will be devastating until resolved.	I will create backups on multiple devices including battery powered ones so that development can continue without power.
Equipment Failure	Low	Medium	This may lead to my devices not functioning as intended leaving me unable to complete the task on that device.	I own multiple devices and can access several more. All my devices are reliable and from reputable manufacturers. Equipment failure should not be too damaging.
Data Loss	Very Low	Severe	Loss of data will be catastrophic and can range from an annoyance to a complete failure of my project.	I have 4 drives on my pc, a server in the Netherlands, my phone, and cloud storage solutions. I intend to back up my work daily to all these to keep the impact of possible data loss to a bare minimum.
Inability to Establish Connection	Low	High	This will lead to the artefact failing and the transfer of files not being possible due to a lack of a recipient / host.	I intend to research more into peer-to-peer connections and practice establishing connections before development to keep the risk of this to an absolute minimum.
Inability to Transfer Files	Medium	High	This will lead to the artefact failing and the transfer of files not being possible due to the established connection refusing the transfer of files.	I intend to research more into the SFTP protocol to ensure that any established connection is configured properly to allow the transfer of files.

Risk	Likelihood	Impact Severity	Impact	Mitigation
Inability to Implement End-to-End encryption	Medium	Medium	This will not lead to the failure of the artefact but will severely damage its usability in the real world.	As I have little experience with end-to-end encryption, I plan to dedicate time to ensure that I am able to deploy such a system robustly and effectively.
Unable to Complete Project by Deadline	Low	High	This will lead to an incomplete project and possibly an incomplete artefact. This will cost me marks and the problem will not be solved.	Using a Gantt chart, I will methodically plan out the timeline of this project in order to track my progress and finish the project on time. This will protect me from poor time management and should lead to project completion in a timely manner.
Inability to Successfully Develop a Stable Application	Low	Medium	This may lead to the application partially working but being unable to be relied upon.	I intend to make the application as simple as possible to begin with using good coding practises to ensure initial stability is in place. From there I shall expand further, and I shall plan this out in the Gantt chart.
Ill Health Preventing Application Completion	Low	Medium	This may lead to my inability to work effectively or at all on the project for a period of time.	In my Gantt chart I shall be sure to allow myself extra headroom in the event of any unexpected events. If the illness is too severe, I shall apply for a SAC or deferral with the university.
COVID-19 Related Restrictions	Medium	Low	This may lead to my ability to access the universities facilities being restricted as well as potentially changing the nature of interactions with university staff.	My project can comfortably be done from within the confines of my own room. This means that it is not compulsory for me to be at university for project completion.
Catastrophic Family / Close Friend Incident	Unknown	High	This may require me to take a break from studies. Affecting the progress and completion of my project.	Unfortunately, nothing can be done to mitigate this.

3.3.2 Risk – Due to Artefact

Risk	Affected Stakeholder	Likelihood	Impact Severity	Impact	Mitigation
Data Loss During Transfer	Me, My Friend, Developer, Video Editor, Scientist	Low	High	Data Loss during transfer can lead to failed or corrupted transfers. This would mean data would need to be resent by the stakeholders.	Using the TCP protocol, I hope to avoid this as the protocol is able to resend undelivered packets.
Application Not Able to Transfer Files	Me, My Friend, Developer, Video Editor, Scientist	Low	High	Inability to transfer files would make the application unable to fulfil its purpose meaning that the stakeholders cannot use it.	I shall do research into the SFTP protocol to ensure I am confidently able to setup a working file transfer.
Application Unable to Establish Connection	Me, My Friend, Developer, Video Editor, Scientist	Low	High	Inability to establish a connection would make the application unable to fulfil its purpose meaning that the stakeholders cannot use it.	I shall do research into the peer-to-peer connections to ensure I am confidently able to setup a stable connection.
End-to-End Encryption Does Not Work / Has not Been Implemented	Developer, Video Editor, Scientist	Medium	Medium	Stakeholders will be unable to send their data securely, this will be particularly impacting for those who may want to send private data.	I will do extensive research into end-to-end encryption systems to ensure I am able to implement it correctly.

Risk	Affected Stakeholder	Likelihood	Impact Severity	Impact	Mitigation
File Transfer is Slow	Developer, Video Editor, Scientist	Medium	Medium	Stakeholders who are prone to regularly transferring massive files may run into a speed bottleneck, this will affect their workflow.	I shall take extra steps to ensure that the correct protocols are being used and the code is written as efficiently as possible.
Project is Incomplete	Me, My Friend, Project Supervisor, The University of Hertfordshire	Low	Medium	Stakeholders may be unable to use the application or may use an application with a limited feature set. This will hurt user experience. Furthermore, this will reflect poorly on the marking performed by the supervisor and the university.	Using a Gantt chart, I will methodically plan out the timeline of this project in order to track my progress and finish the project on time. This will protect me from poor time management and should lead to project completion in a timely manner.
Application is Buggy / Unstable	Me, My Friend, Project Supervisor, The University of Hertfordshire, Developer, Video Editor, Scientist	Low	High	Stakeholders may suffer from an inconsistent experience that may lead to errors and incomplete transfers and crashes. This will damage the user experience and potentially affect the usability of the app. Furthermore, this will reflect poorly on the marking performed by the supervisor and the university.	I intend to make the application as simple as possible to begin with using good coding practises to ensure initial stability is in place. From there I shall expand further, and I shall plan this out in the Gantt chart.

3.4 Legal Ethical Social and Professional Implications

There is a final subject to discuss before the implementation of my artefact and that is the potential implications the creation of my artefact can have both during its development and after.

Legal Implications – I believe my artefact will have no legal implications due to it not being software that is a virus or malware. Nor will the artefact be intended to facilitate law breaking in any capacity. It will be created using libraries and software that are licensed for my use for the purposes of this project. All testing and use of the solution will be done over networks and on systems I have ownership over / permission to use.

Ethical Implications – I believe my artefact will have no ethical implications. It is a file sharing application designed to circumvent restrictions placed by freely using the services of other companies. I believe my peer-to-peer decentralised approach will allow for greater security and privacy for the users and will mean they will not need to incur costs using solutions offered from companies which host third party servers. The application has been designed in no way that exploits or takes unfair advantage of any person, system or party.

Social Implications – I believe my artefact may have minor social implications. Individuals who find themselves facing the same problem I have encountered, may want to use my application to send files securely and rapidly to other users free of charge. This may lead to more people having access to secure peer-to-peer encrypted file transfer.

Professional Implications – If my artefact were created as a research and development project outside of university with grander project aims it could take business from companies that offer cloud storage solutions for individuals and companies who simply desire the file sharing functionality.

4.0 Implementation

4.1 Implementation Outline and Approach

This chapter covers the creation and implementation of my artefact by detailing the steps taken and how I utilise the research, the technologies, the methodologies and the design previously discussed to create a suitable solution to the project problem while meeting all success criteria. As I will be using the FDD methodology throughout the implementation I will be primarily focusing on addressing the success criteria in each section and expanding the feature list of the artefact before diverting my efforts to Nielsen's Heuristics and other quality of life aspects of the artefact. I will make continual revisions to the artefact, documenting my changes and improving on features each iteration.

I aim to begin by establishing a connection to a server I own to prove that my application can establish connections.; from there I will develop the artefact further as detailed in section 3.2.

4.2 C# Visual Studio and Windows Forms

I began by installing the latest public release of Visual Studio version 17.1 onto my system which is free to use for academic purposes which is the purpose I intend to use the software for.

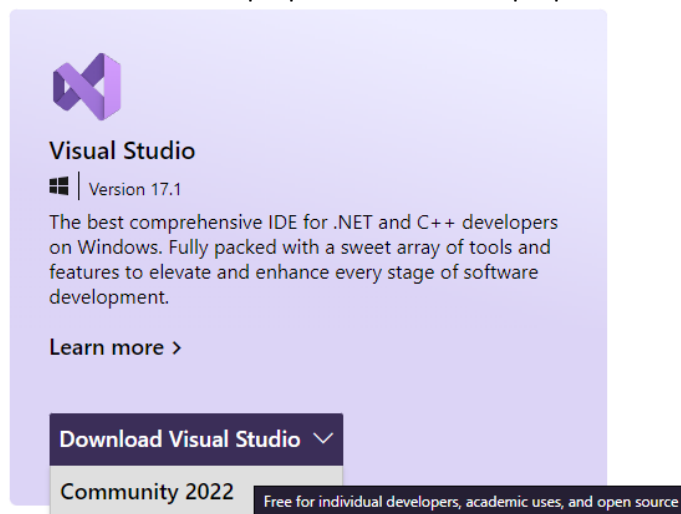


Figure 2 – Screenshot of Visual Studio download page (Microsoft, 2022)

After successfully installing the product I was able to quickly create a new project thanks to Microsoft providing templates for Windows Forms Applications as standard with Visual Studio, one of the many reasons I have chosen it for my IDE. Once a blank Windows Forms project was created, I was then easily able to begin the design and create the basic UI for my artefact which I will go into more detail in the next section.

4.3 Preliminary Application Layout and Navigation

I began by creating 3 separate forms within the project file, I have done this to divide up the functionality the application offers to make it more straightforward to use as I will create a separate form for sending and receiving files. At this stage of the development, I wanted to focus my efforts into making a usable and functional user interface and developing a more usable and aesthetically pleasing one. The first form I created would contain the main menu.

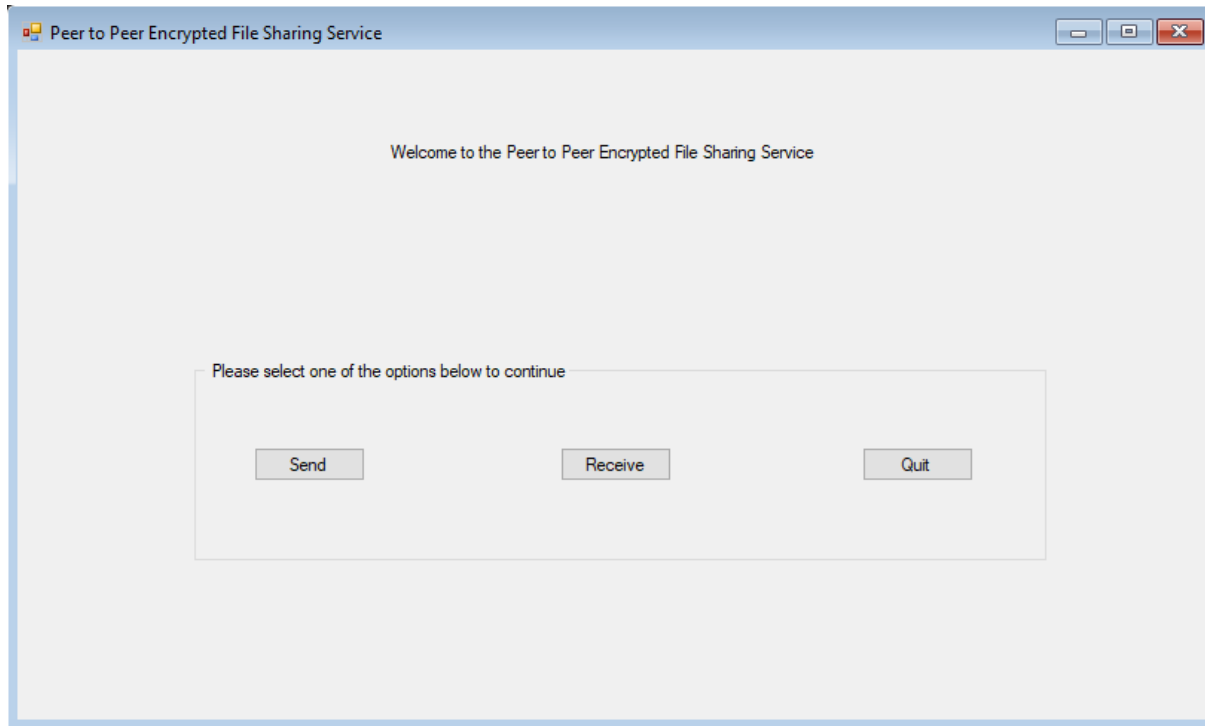


Figure 3 – Main Menu Revision 1

As can be seen above a basic main menu was created providing the user with options to send or receive files as well as the ability to quit. Everything that the user can interact with at this stage is suitably labelled and the user will be able to easily understand the buttons and what they do. After creating the main menu, I then created the form where the user would be directed to when they clicked on the send button. For this form I decided to include the text boxes where the user would input the details of the recipient as well as the ability to select the intended file to send as this would be information that would be needed for someone sending a file. I drew inspiration from the popular FTP client app FileZilla in order to make sure what I have on my form is relevant and useful to the user.

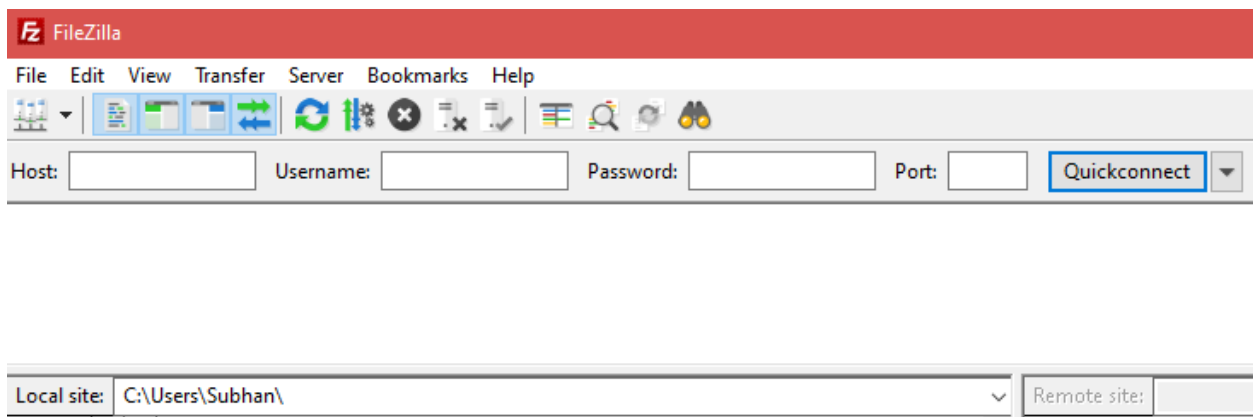


Figure 4 – Screenshot of FileZilla UI 3.59.0

Send a File

Recipient Details

Name

IP Address

Port

Password

Selected File

Browse

Preferences

Set Upload Speed kbps

Apply

Send Return to Main Menu

Figure 5 – Send a File Revision 1

I have also included a preferences section which I intend to flesh out more at a later stage, currently it contains an option for limiting the download speed which can be a desired feature on a shared network to prevent the application from utilising too much bandwidth.

Lastly, I created the 'Receive a File' form where I listed basic options to control the username and password for the recipient system as a security measure and an option to cap the download speed of the incoming files to prevent the application from utilising too much bandwidth.

Receive a File

Preferences

Set Username

Set Password

Set Download Speed (kbps)

Set

Return to Main Menu

Figure 6 - Receive a File Revision 1

4.4 Sending Files to Another System

While I do intend to use SFTP for my application, I firstly needed to explore whether or not connections could be established with a remote system using my application. This will allow me to re-think my strategy if the connection did not work, however if it does work it will mean I have successfully met one of my success criteria.

To establish my connection, I will be connecting to a server using the File Transfer Protocol (FTP) I am renting in the Netherlands. I believe this will be a suitable test as it is a remote system not on my local network allowing me to test a connection with a system outside of my local network. As the server is also incredibly reliable and well programmed, any issues encountered will likely be the fault of my code.

4.4.1 FTP Connection Test

To begin the FTP test, I firstly needed to ensure I had all pre-requisites needed to establish SFTP / FTP connection. For this, I decided to use the WinSCP .NET Assembly (Přikryl, 2022a) which is a library of packages that will allow my artefact to establish SFTP and FTP connections and utilise them. The license is free to use to “Run WinSCP for any purpose for free” and to “Copy WinSCP anywhere you want” (Přikryl, 2022b). After importing the library into Visual Studio, I then included the packages within the C# code through the code below.


```
10  using WinSCP;
11
```

Figure 7 – WinSCP inclusion

Now that I have installed the required packages I am now able to continue my development. I believe the next logical step will be to revise my UI for the ‘Send a File’ form as well as code the functionality into the buttons allowing the form to fulfil its function.

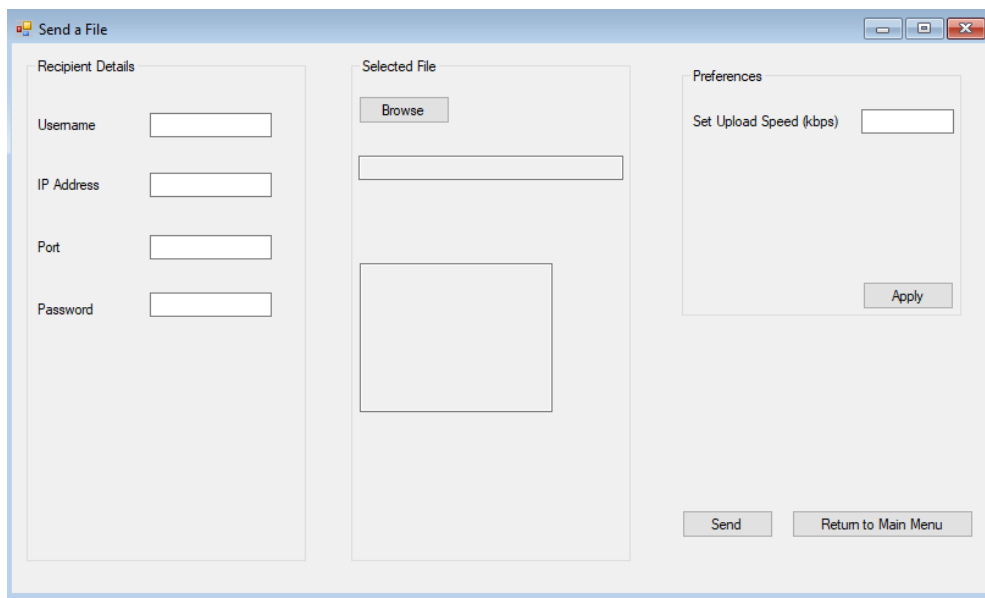


Figure 8 – Send a File Revision 2

As seen above, a couple of subtle changes were made to the UI, The Recipient Details section is now using more appropriate naming for the details of the system, and the selected file section will now display the result of the file transfer in the created text box below.

Moving onto the code, I have added functionality to the Browse button using the openFileDialog class, allowing the user to select any file on the host computer that their account has access to. This file name as well as its complete path will then be shown in the text box below the browse button.

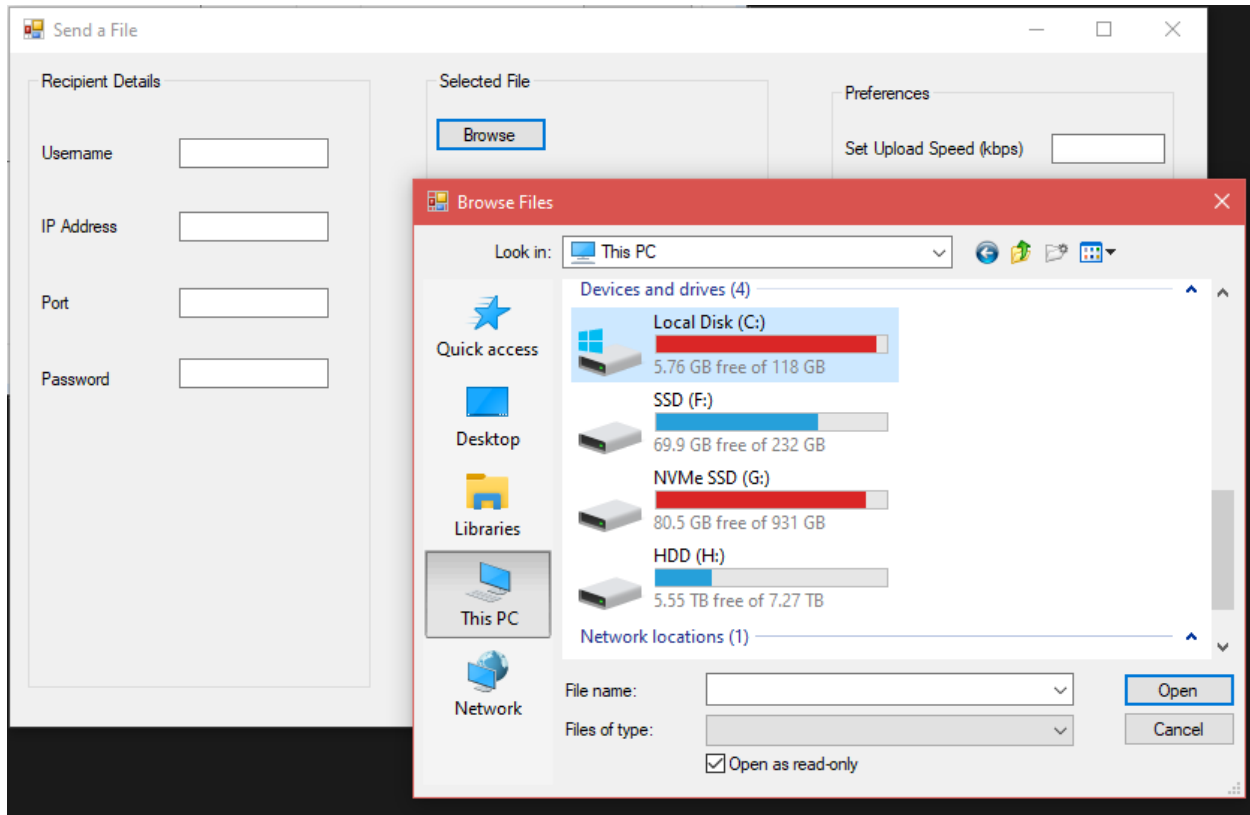


Figure 9 – Browse Functionality i

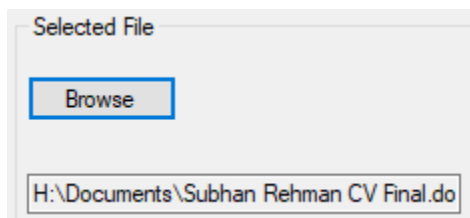


Figure 10 – Browse Functionality ii

In order to use the address taken from the use of the Browse button, the variable it would be stored in must be accessible in other classes. Therefore, I created a public class called Variables and I stored the variable within it as an empty string.

```
2 references
public class Variables
{
    public static string address = " ";
}
```

Figure 11 – Public Variables class

Now I am ready to implement the FTP functionality into my code. Using the private class of the Send button shown in figure 8, I wrote the required code to establish an FTP connection. As you can see below, 4 variables were created to store the inputs of the 4 text fields in figure 8. The input at the time of pressing send will be written to the variables and stored for later use. For the variable 'port' I used the Parse() function translate the textbox input which is considered to be a string, into an integer. While at this stage there is no error checking for the user inputs, this is a feature driven development and I aim to implement such features later on.

```
private void button2_Click(object sender, EventArgs e)
{
    string username = textBoxName.Text;
    string pswd = textBoxPSWD.Text;
    int port = int.Parse(textBoxPort.Text);
    string host = textBoxIP.Text;
```

Figure 12 – Assigning User input to Variables.

After the user input had been taken I created a try-catch statement to run the FTP code within, this will allow me to display an error in the event that something goes wrong and the transfer is unsuccessful. Within the statement, I used the SessionOptions class to create a new session and input in the required information to establish the session. Below, is the class with the pre-requisite data given to the relevant variables taken from the user input. The protocol has been set by me to FTP.

```
try
{
    SessionOptions sessionOptions = new SessionOptions
    {
        Protocol = Protocol.Ftp,
        HostName = host,
        UserName = username,
        Password = pswd,
        PortNumber = port,
    };
}
```

Figure 13 – SessionOptions Information

This next segment of code creates a new session and attempts to establish a connection with the FTP server. I have specified within transferOptions to set the transfer mode to binary. This is incredibly important as without this specification the transfer may default to ASCII mode, where only a “single text file between two PCs” (Broadcom, 2018) is supported. Binary mode allows the transfer of any file. Within transferResult the files are transferred from the user specified path from earlier, into a specific directory which I later also intend to make user modifiable. The ‘false’ is referring to an option to delete the file on the host system, this is something I have chosen to keep disabled as in the event of an unsuccessful file transfer the data might be lost without a record.

```
using (Session session = new Session())
{
    session.Open(sessionOptions);

    TransferOptions transferOptions = new TransferOptions();
    transferOptions.TransferMode = TransferMode.Binary;

    TransferOperationResult transferResult;
    transferResult = session.PutFiles(Variables.address, "/misc/", false, transferOptions);
    transferResult.Check();
}
```

Figure 14 – FTP Transfer Code

Finally, I need to show the result to the user of the application so that they are aware whether or not the file has been successfully sent. To achieve this a simple foreach loop was used that appended the text within the large textbox created in figure 8 with a message declaring whether or not the file transfer was successful or if an error was encountered.

```
foreach (TransferEventArgs transfer in transferResult.Transfers)
{
    textBox2.AppendText("Uploading ");
    textBox2.AppendText(transfer.FileName);
    textBox2.AppendText(" has been successfully completed.");
}

catch (Exception)
{
    textBox2.AppendText("The transfer ran into an unexpected error");
}
```

Figure 15 – Results of FTP

Now that the FTP code had been written it was time to test it. For this, I will use the aforementioned Netherlands server which I rent and have full rights to access and use for any lawful purpose. I have the server connected to windows as an sshfs network drive, so I will be able to easily see the success or failure of my file transfer to verify whether or not my artefact has been successful.

Compiling the code and running the program I navigated to the 'Send a File' page of my solution. On this page I input the information of my server, as well as the intended file which would be an image of the national flag of Azerbaijan. The intended directory at this stage has been emptied for the purposes of the test.

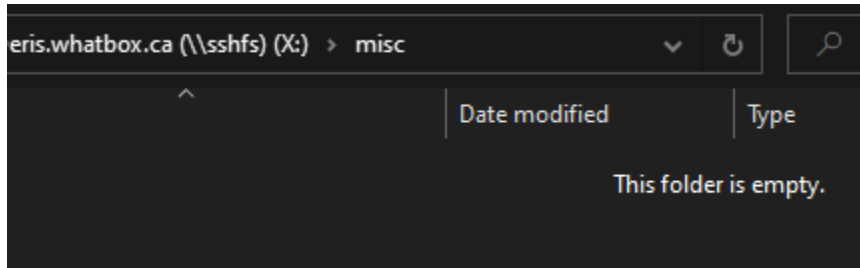


Figure 16 – Before FTP Transfer

Below is what the user would see before clicking on send, all the information would be correctly entered for the intended recipient and the file would be chosen. I am using port 21 as that is the default port for the file transfer protocol and is the configured port for my server.

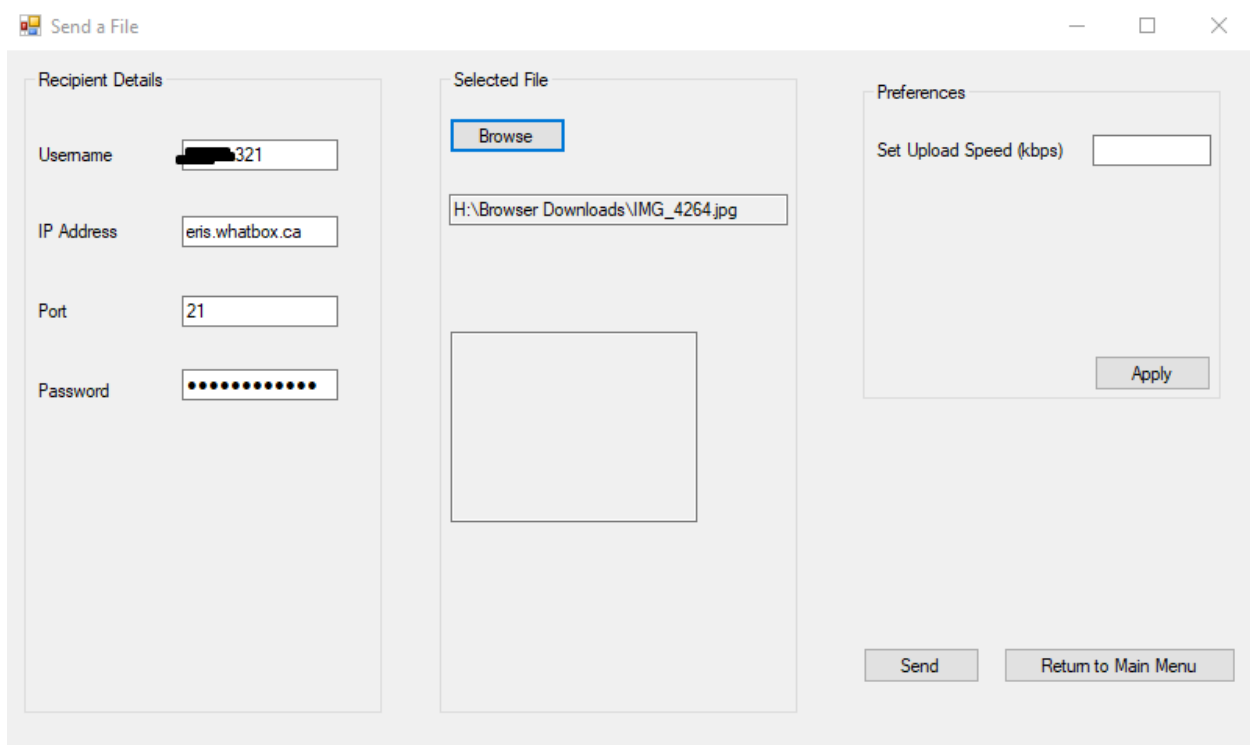


Figure 17 – Send a File for FTP

After I clicked on send, the application paused briefly before updating with the message below. This shows that the transfer was successful, and that the recipient has received the file.

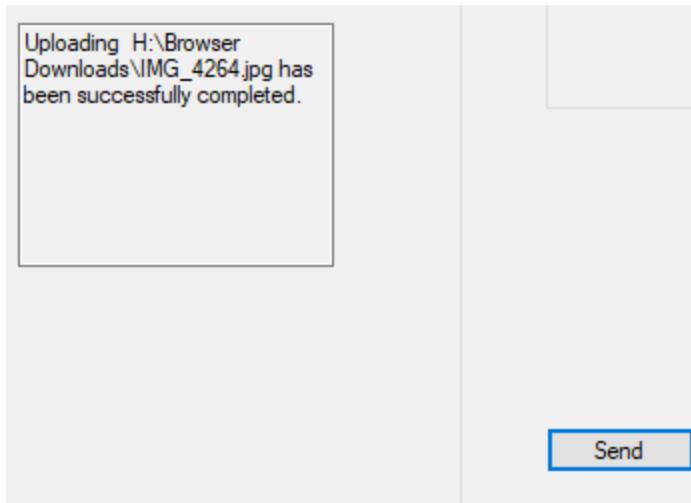


Figure 18 – Successful FTP Transfer

I then checked on Windows file explorer to go to the directory my application claimed the file had been sent to. The file had transferred perfectly with no corruption or loss in image quality.

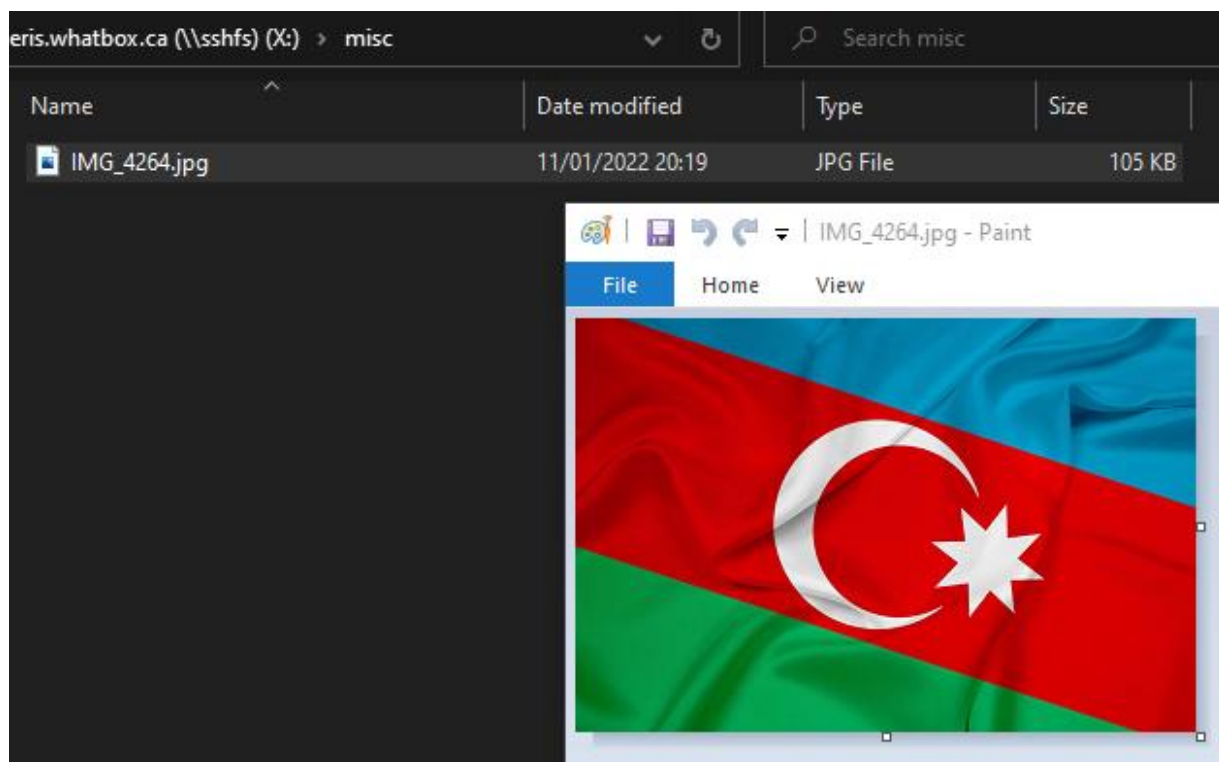


Figure 19 – Proof of Successful FTP Transfer

To conclude, the FTP connection test has been a complete success highlighting my artefact's ability to establish a connection, send files using peer to peer networking and to do so in a stable manner. As I now have this knowledge, my next step is to implement encryption, namely the SFTP protocol to see if I can replicate my results with a secure connection. If successful, I will begin developing functionality to receive files.

4.4.2 SFTP Connection Test

After establishing that a connection is possible with another device outside of my network using peer to peer networking, the next step is to use an encrypted protocol to ensure that the data transfer is encrypted and therefore unreadable to anyone who may intercept it. The protocol I will be using is SFTP as mentioned earlier for its similar usage to FTP but with added security.

To begin, I edited the protocol field within sessionOptions to change the protocol to Sftp from Ftp, this will mean that the imported package will treat the connection as an SFTP connection and attempt to establish an encrypted connection when the session is created.

```
SessionOptions sessionOptions = new SessionOptions
{
    Protocol = Protocol.Sftp,
```

Figure 20 – Changing protocol to sftp

After specifying this I then began my connection test leaving all other parameters unchanged and the code unaltered. I performed the exact same test using the same flag file and transferring the file to the same location in order to minimise any variance between the two tests. The exact same credentials were used with the exception of the port number. I had previously used port 21 as that is the standard for FTP connection, however as I am trying to establish an SFTP connection I require the use of port 22 which is the standard for that protocol.

The screenshot shows a window titled "Send a File" with three main sections: Recipient Details, Selected File, and Preferences. In the Recipient Details section, the Username is "321", IP Address is "eris.whatbox.ca", Port is "22", and Password is masked with dots. The Selected File section shows a "Browse" button and a text box containing "H:\Browser Downloads\IMG_4264.jpg". Below this, a message box states "The transfer ran into an unexpected error". The Preferences section has a "Set Upload Speed (kbps)" input field and an "Apply" button. At the bottom right, there are "Send" and "Return to Main Menu" buttons.

Figure 21 – Unsuccessful SFTP transfer

As can be seen above the transfer failed to transfer any files so clearly something I had done had been incorrect. To solve this issue, I referred to the documentation on the WinSCP website as I was using their library to help facilitate file transfer. I discovered that I was missing the ssh host key fingerprint, this fingerprint allows for authentication allowing the user “to verify the expected server host key” (Přikryl, 2022c) and to check whether or not the connection they are establishing is with a trusted party or an attacker. As I had failed to implement this, the code refused to send any files to the recipient or establish the connection.

```
using (Session session = new Session())
{
    string serverfingerprint = session.ScanFingerprint(sessionOptions, "SHA-256");
    sessionOptions.SshHostKeyFingerprint = serverfingerprint;
}
```

Figure 22 – Retrieving Fingerprint

I created a string which would hold the server’s fingerprint for use when conducting the file transfer, this string would then take the fingerprint that the server provided when a connection was made. From here I would add the fingerprint into the sessionOptions for the SFTP session allowing the program to establish the connection and send the file.

I ran my SFTP test again in the exact same manner as before to see the result of my changes.

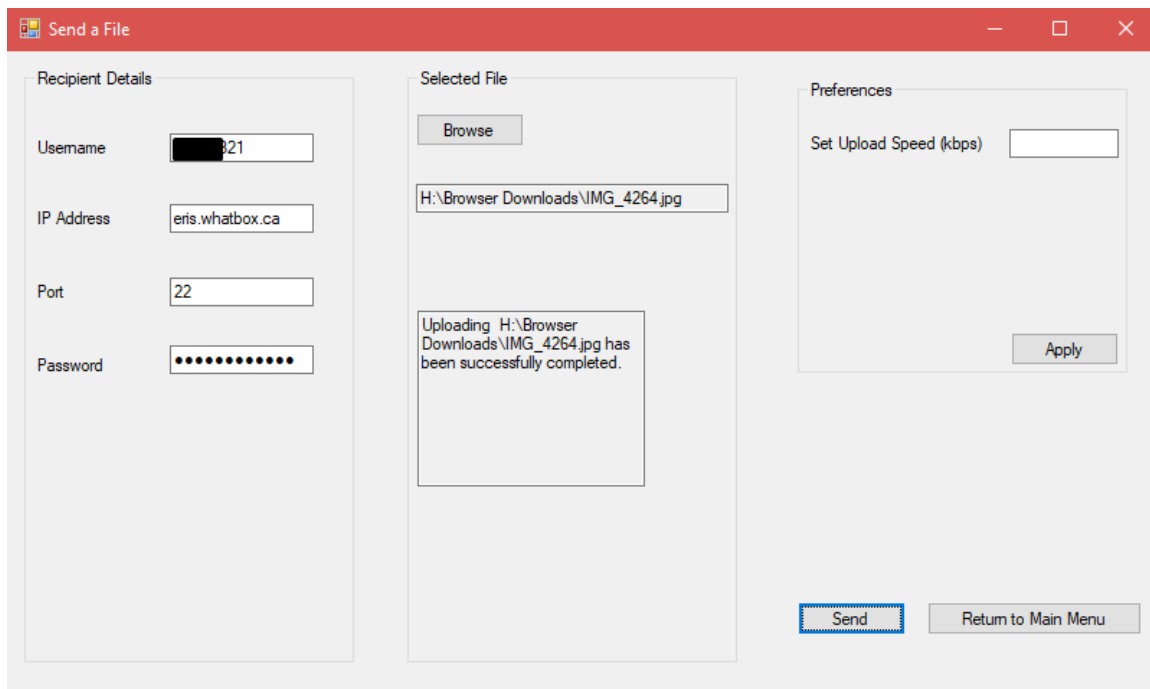


Figure 23 – Successful SFTP Transfer

To conclude, the SFTP transfer was a success thanks to the fingerprint verification now implemented into the program. My next step from here is to now implement features within my program that allow it to receive files. Once this has been achieved, I will test the two features together and conduct a file transfer.

4.5 Receiving Files from Another System

Now that I have established and proven my application's capability of sending files to remote systems, it is now time to develop and implement the functionality to receive files. However, for this I will need to use port forwarding which will require the manual configuration of my router as both devices are behind NAT routers.

4.5.1 Receiving Files from Within the Local Area Network

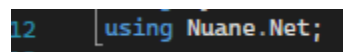
To receive files from within a local area network is a lot easier than receiving files from outside one. This is due to lack of port forwarding required to connect the two local devices together only requiring the knowledge of the other systems local IP address.

I decided that as my functionality for sending files used the secure file transfer protocol, it would be best that if the receiving functionality of my artefact created a temporary SFTP server on the recipient device for the sending functionality to connect to. This would allow for fast encrypted file transfer with the recipient able to terminate the server whenever they pleased.

Therefore, I needed a .NET library that would be able to host an SFTP server in C#. This proved more difficult than the previous section and I could only find two vendors. The first was a company called /n software and their Secure Blackbox® software (/n software, 2022) provided a suite of features and libraries that could assist with what I required. Unfortunately, this software required a license that started at \$1199 and the free version was only a 30-day free trial. The second company was called Rebex (Rebex, 2022) and their software was much more focussed around SFTP, unfortunately this company had similar issues to /n software and I was unable to use the software without a temporary trial or a fee from \$349.

After doing some more research I discovered that Rebex had acquired an older piece of software known as SFTP Server Lite (SFTP Server Lite, 2011) and had based their current software of the improvements they had done to this one. However, this software was still available for "free for both commercial and non-commercial use" (SFTP Server Lite, 2011) meaning that I am able to use it. Unfortunately, the library is small and underdeveloped however the core functionality was still present meaning that my implementation will be able to continue as planned.

I imported the package into the receive form's C# code file and I began to inspect the package and the features it offered.



```
12 using Nuane.Net;
```

Figure 26 – Importing Nuane.net

I knew from the previous section that I would require a username, password, IP address and a port number as essential information for an SFTP session, therefore I decided to revise my 'Receive a File' form to meet these requirements. As you can see below, I have added fields for the IP address of this machine and the port number that will need to be forwarded. I have decided to use the NumericUpDown item for the port selection as the entry will always be an integer between 0 – 65535, this should also reduce errors when assigning the value to an int variable. I have removed the download speed functionality as it is not supported by the library I will be using.

Figure 27 – Receive a File Revision 2

As I will be selecting a download directory and not a file through the browse button in this form, I have decided to use the `FolderBrowserDialog` feature which will allow the user to select a desired download directory for the files. The folder that the user selects will be displayed in a read only text box below the browse button (Fig 27) and the address will also be assigned to the public variable `DownloadPath` (Fig 28) so that it may be accessed by the other classes.

```
1 reference
private void button3_Click(object sender, EventArgs e)
{
    FolderBrowserDialog FolderBrowserDialog1 = new FolderBrowserDialog();

    if (FolderBrowserDialog1.ShowDialog() == DialogResult.OK)
    {
        Variables.DownloadPath = FolderBrowserDialog1.SelectedPath;
        textBox6.Text = Variables.DownloadPath;
    }
}
```

Figure 28 – Folder Selection Code

My next step is to ensure that my code is able to correctly understand the 4 textbox inputs, under the class for the click of the 'Receive' button I listed the 4 inputs being assigned to 4 variables

```
string Username = textBox1.Text;
string Password = textBox2.Text;
string IP_Address = textBox3.Text;
int Port = Convert.ToInt32(Math.Round(numericUpDownPort.Value, 0));
```

Figure 29 – Variable Assignment

This is a similar method I used in the 'Send a File' form however I am now converting the value within the NumericUpDown box to a 32bit integer, more than enough for what I will require.

After analysing the packages from within visual studio and from reading the website the library was hosted on, it became evident that the package is designed to be run on a command line interface. As a result of this all outputs would be sent to the console, something that would be inaccessible to the users of my application. Therefore, I would require a way to display the output within the program.

```
2 references
public class TextBoxConsole : TextWriter
{
    TextBox _output = null;
    1 reference
    public TextBoxConsole(TextBox output)
    {
        _output = output;
    }
    0 references
    public override void Write(char value)
    {
        base.Write(value);
        _output.AppendText(value.ToString());
    }

    0 references
    public override Encoding Encoding
    {
        get { return System.Text.Encoding.UTF8; }
    }
}
```

To do this I imported the System.Text library as this has a writer within it that can take console output and place it within my form. Within the code below the console will constantly append its output onto wherever I decide to use the 'TextWriter', the encoding was set to UTF8 as it is widely used and has the widest compatibility for file types. The class was set as public so that I could use 'TextWriter' later when I coded the SFTP server.

Figure 30 – Console to Form Writer

My next and final step before coding the receiving functionality was to revise the 'Receive a File' form so that it may be used with the TextWriter I created. I created a Console Output textbox that would show the output of the server, I made it exceptionally large as from experience working with servers they produce a lot of output very quickly, for this reason I also added a scroll bar. I decided to also create my own output log to show the user any non-console related messages such as errors.

Figure 30 – Receive a File Form Revision 3

I now began coding the server using the imported library, I decided the most sensible option would be to use a try-catch statement as the server could have many possible points of failure and an exception would prevent the program from crashing. For the catch statement I created a public variable 'i' that would append every time the catch was triggered, this would ensure that the user is able to identify that they are reading a new failure message and not an older one. This is an improvement over the method I used in the 'Send a File' form and allows for better readability. This catch state will write to the Output Log.

```
catch (Exception)
{
    Variables.i = Variables.i + 1;
    textBox4.AppendText(Variables.i + ": The session has enocuntered an error" + Environment.NewLine);
}
```

Figure 31 – Improved Catch Statement

I began my try statement by creating the writer that would be later assigned to the server.Log function allowing the output to be directed to the form. I disabled checking for illegal cross thread calls as this lead to catch exceptions later on in the implementation.

```
try
{
    TextWriter writer = new TextBoxConsole(textBox5);
    TextBox.CheckForIllegalCrossThreadCalls = false;
```

Figure 32 – Implementing writer into try statement.

My next step was to ensure that before the server was created that it would be able to communicate securely with the host, therefore I used the built-in functionality to generate an RSA key, this would allow the application to communicate securely.

```
SshKey rsaKey = SshKey.Generate(SshKeyAlgorithm.RSA, 1024);
SftpServer server = new SftpServer();
```

Figure 33 – RSA Generation

After this I created a new SftpServer named server. With this variable I used 4 functions to assign different parameters to it before starting the server.

```
server.Log = writer;
server.Keys.Add(rsaKey);
server.Bindings.Add(IPAddress.Parse(IP_Address), Port);
server.Users.Add(new SshUser(Username, Password, Variables.DownloadPath));
```

Figure 34 – Server Configuration

server.Log – This is where I am instructing the server to send its log output, as can be seen here, I have assigned it to the writer I have created so that it can send the output to the form.

server.Keys – This is where I am adding the previously generated RSA key to the variable.

server.Bindings – This is where I assign the IP address and port number of the system so that the server is able to correctly communicate.

server.Users – Finally, this is where the server creates a temporary user account with a username, password and the directory that the user has been given access to write to.

After the server has been configured with the information, I write the command to start the server as well as informing the user that the service is operational and ready to receive files within the output log.

```
server.Start();
textBox4.AppendText("The service is operational and ready to recieve files" + Environment.NewLine);
```

Figure 35 – Server Start Code and Information

With the program coded it was time to compile and test if the functionality worked as intended. For this part of the test, I decided to use an SFTP client on my phone before using the 'Send a File' form as the app is much more stable and feature rich than my code and therefore any issues would be the fault of my programming, this would make identifying issues much easier as the list of possible variables would be reduced. I also wanted to avoid any unintended consequences of attempting to connect my computer to itself potentially causing a conflict and preventing me from testing my software.

To begin I needed to know the local IP of my system. So, I used the ipconfig command in command prompt to gather this information. This is the IP that every single device on the local network will recognise my device as. As for the port number, for now I will be using the standard for SFTP which is 22, however as I am not currently venturing outside of my network, any non-reserved or in use IP address can be freely used.

```

Windows IP Configuration

Ethernet adapter Ethernet 3:

    Connection-specific DNS Suffix  . : 
    IPv4 Address. . . . . : 192.168.0.31
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . : 192.168.0.1

```

Figure 36 – ipconfig Screenshot

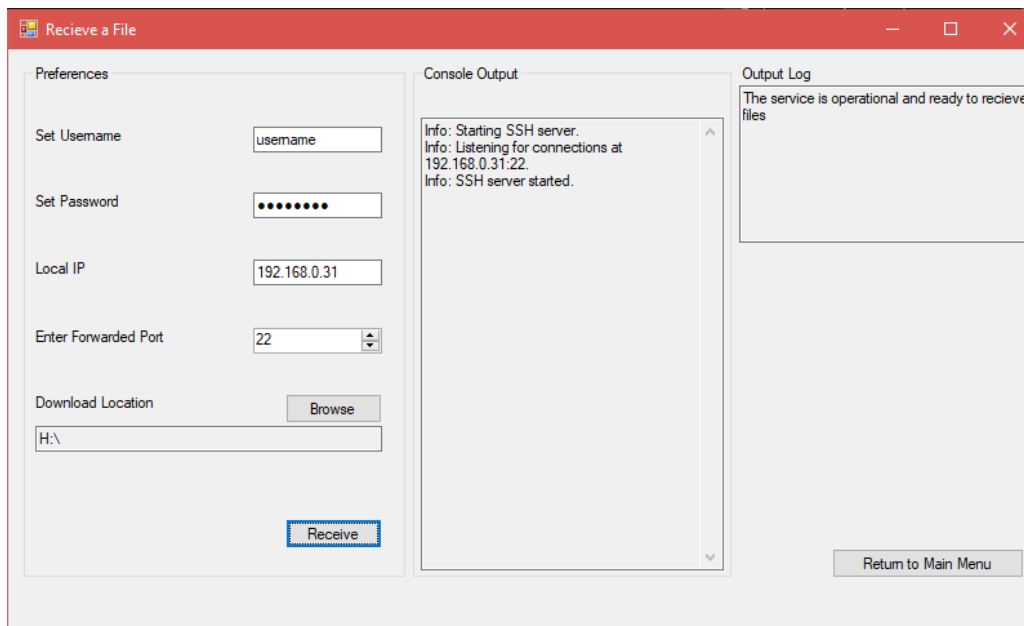
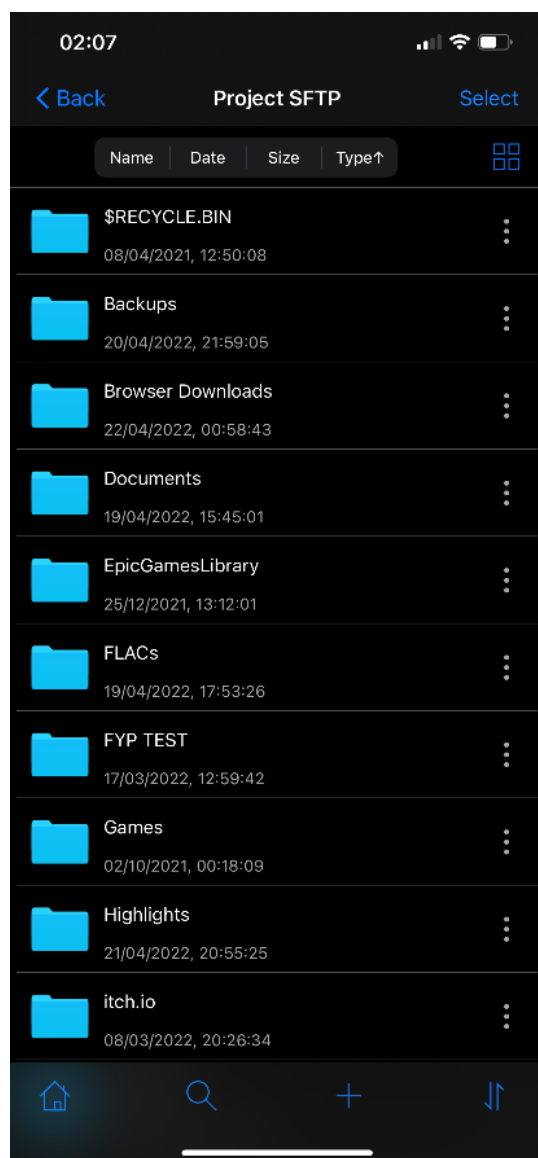


Figure 37 – Successful Server Deployment

With all the information entered into the program, I clicked on receive and began the server. As can be seen above username and password were set to that respectively and the download location was set to the top-level directory. As can be seen in the console output the text writer functionality had worked correctly and the server output is showing the IP address and port number that the server is currently listening for. Finally, the output log has confirmed that the server is operational and is ready to receive files. I will now attempt to connect with my phone, if successful, I expect to see a directory listing of all folders and files present within H:/.



As can be seen on the left, I was able to successfully connect to the server created within my form using an application on my phone within my local network. I am able to navigate around the directory, but I cannot access other drives as the server did not allow me access to those drives within the download location settings. As can be seen on the below are the settings used to connect to the server (the password has been automatically redacted due to the apps security settings).

Figure 38 – SFTP Phone

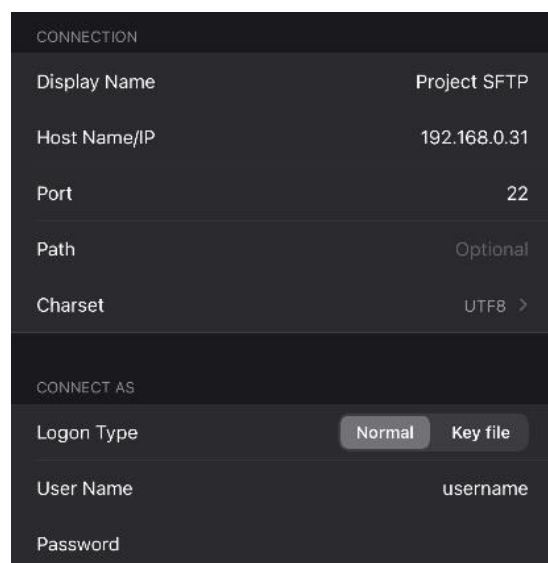


Figure 39 – SFTP Phone Settings

On the lower left we have the console output generated by the server when the connection was established, as we can see from the screenshot, the RSA keys were successfully exchanged before the connection was established and the user was successfully authenticated as the login details provided matched those that had been set.

Now that it has been shown that the artefact is capable of establishing a connection capable of receiving files, the next step is to send some files to the application to test whether or not the application is capable of saving them. For this I will be sending an image of a duck to my computer from my phone using the Receive a File form. I will also be sending the file to a specific sub directory which will be set within the settings presented to the user before they begin the server. I have chosen a directory which is several folders nested to see if the server is capable of writing to that location and successfully transferring the file. See bottom right.

Figure 40 – Console Output

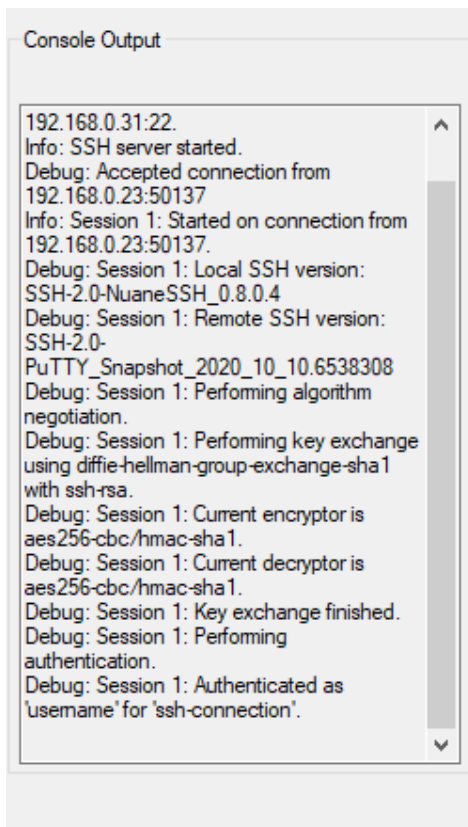
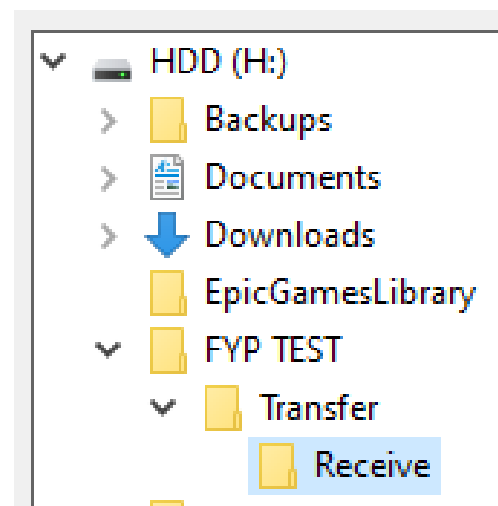


Figure 41 – Receive Test



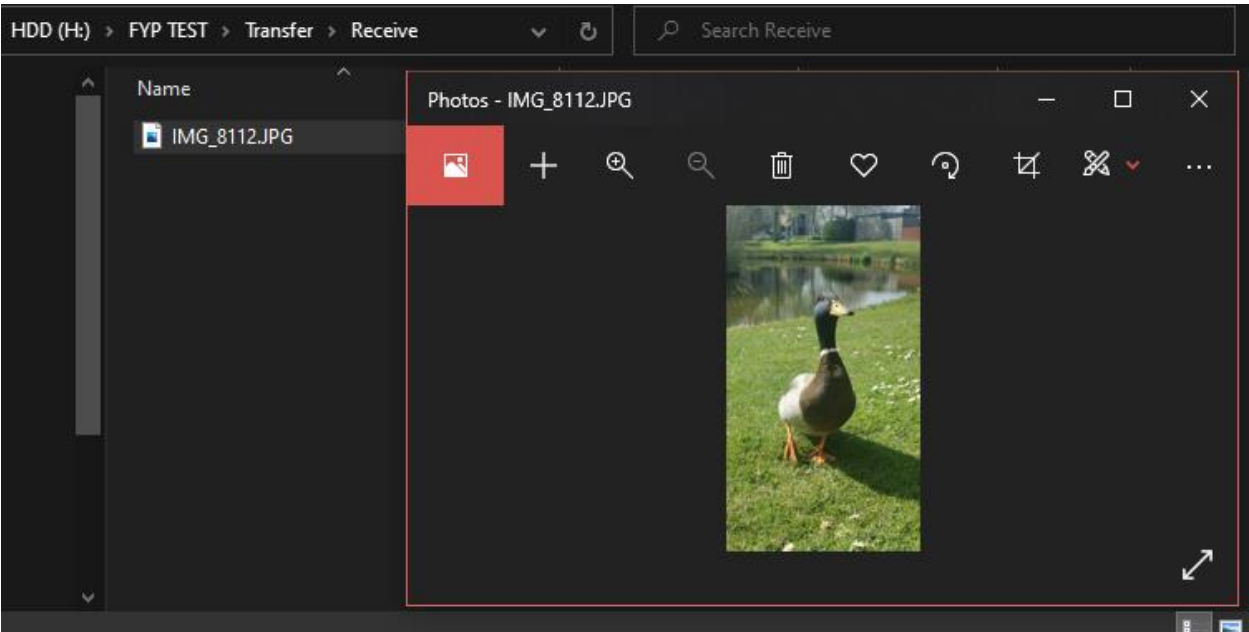


Figure 42 – Successful Receipt

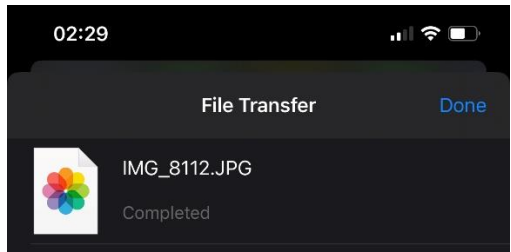


Figure 43 – Successful Receipt Notification

As shown in figure 42 and 43, the transfer was a success. This shows that my application is able to send and receive files on a local network, and send files externally, the next step is to allow the receiving of files from outside the local area network. This will require router configuration and additional setup. If this is successful, I will run the application on two different devices, on two different networks and test whether or not the applications can send and receive files from one another.

4.5.2 Receiving Files from Outside the Local Area Network

After establishing that receiving files is indeed possible with my application the next step is to enable this functionality for systems external to my local area network, this must be achieved with port forwarding which is done through the router. I logged into the administrator settings for my router, and I enabled port forwarding for a small number of ports, the port I will be using for the purposes test will be 1604.

Local		External		Enabled	Delete
IP address	Port range	Port range	Protocol		
192.168.0.31	1600-1604	1600-1604	TCP	<input checked="" type="checkbox"/>	<input type="checkbox"/>

Figure 44 – Port Forwarded

If a device is external to a network, it is not useful for them to know the local IP of the machine, they must know the public IP address. This can be easily found with a simple Google search 'my ip'. Now that I have the public IP address and a port number has been successfully forwarded I am ready to attempt communicating with the application from outside the application.

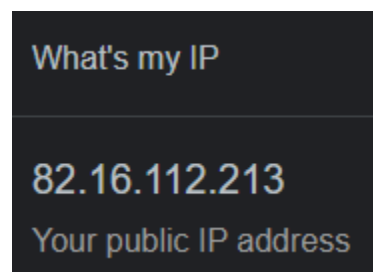


Figure 45 – Public IP address. **Note to UH Staff:** *If this work is ever released outside of marking purposes, ensure this address is redacted as well as any other instances of it within this document. Do not attempt to connect to the address.*

For this test I will be using a more secure and harder to guess username and password as I am exposing my computer to the internet. As shown below the only major change required is the port number, the IP address remains the same as it is the router that handles the request and sends it to my application via the port number which the application is currently listening through.

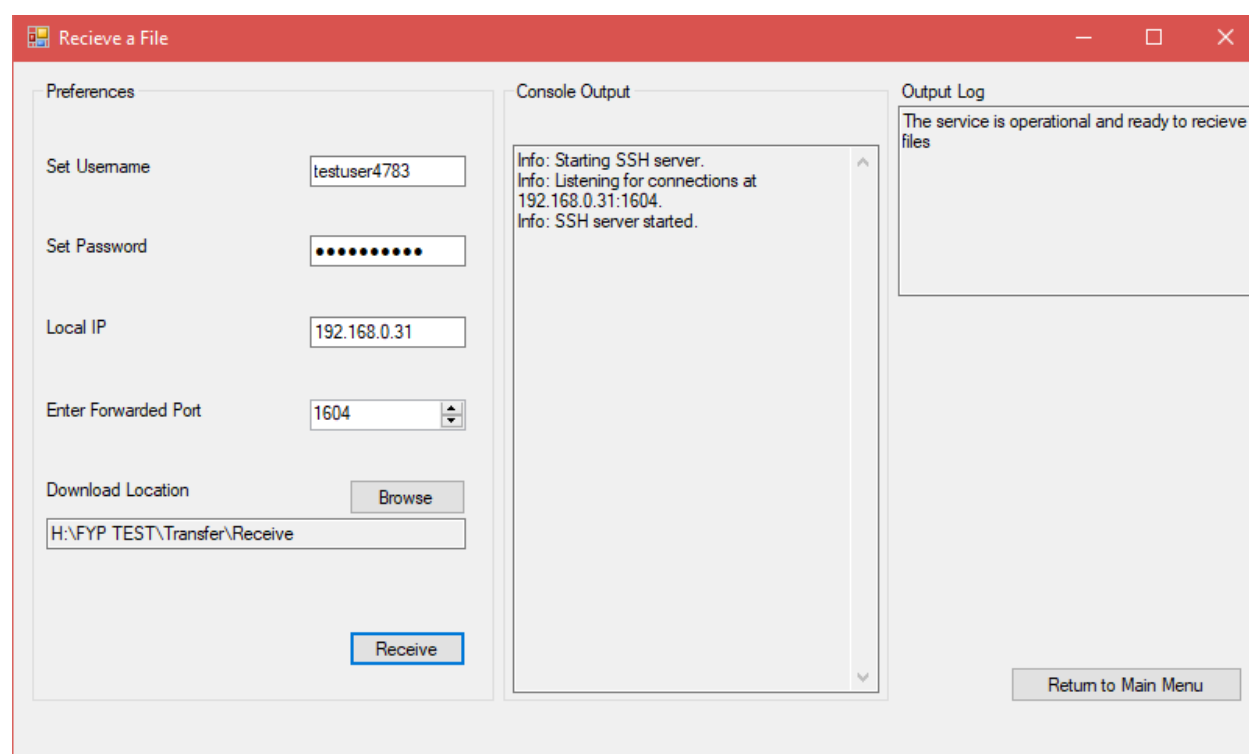


Figure 46 – External Receive Configuration

I turned off the Wi-Fi feature on my phone and used the mobile data to establish the connection, the reason for this is that I can use a network that is external to my local area network to make the connection. As can be seen on the left, the application received a connection from a public IP address completely different to mine proving that an external connection was established.

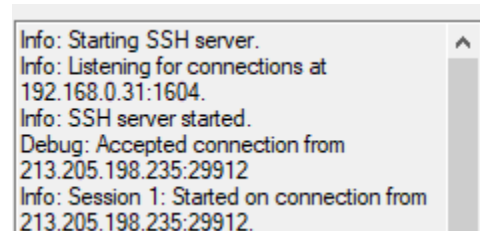


Figure 47 – External Receive Connection Proof

I will now send another image of a duck to my application this time from outside my local area network

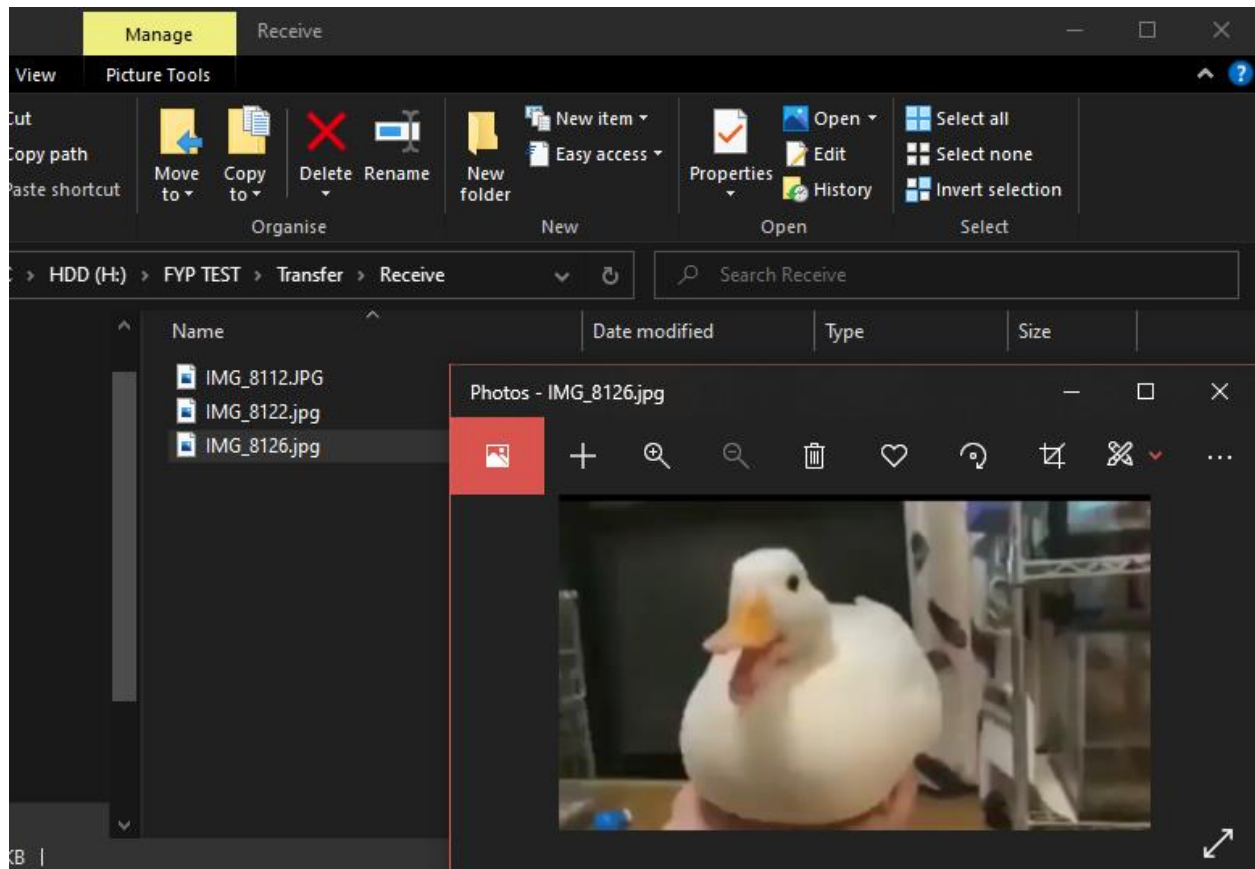


Figure 48 – Successful External Receipt

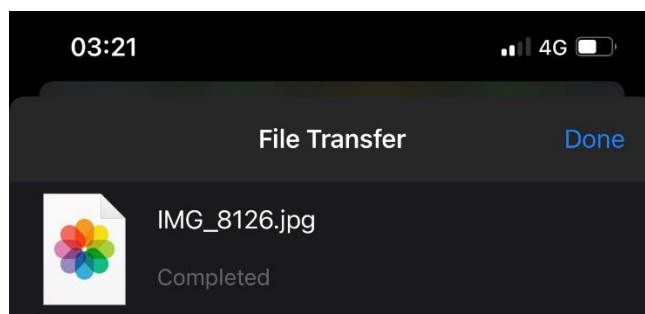


Figure 49 – Successful External Receipt Notification

As can be seen above the external file receiving test was a success, my application is able to send and receive to devices both within the local area network and external to it. My next step before running two instances of the application and making them communicate to one another, will be to fix bugs I have noticed and to improve the code's reliability and stability. I will also make minor improvements to the UI.

4.6 Fixing Bugs and Improving Code

Instead of beginning with fixing bugs, then implementing changes which will inevitably create brand new bugs, I decided to implement my enhancements to the UI and code first, before addressing bugs. I decided to begin with 'Send a File' first, before moving to 'Receive a File'.

4.6.1 Improving Code and UI

I decided to improve the UI for 'Send a File' taking inspiration from the changes I made to 'Receive a File'. I firstly began with resizing and relocating the output log to be more aesthetically pleasing and easier to read. I changed to a NumericUpDown box for the port number like the one I implemented in 'Receive a File' as it will only accept integer inputs. Finally, I have added the functionality for the user to write to the directory of their choice, this will allow for a user to send a file to wherever they desire within the directory they have access to.

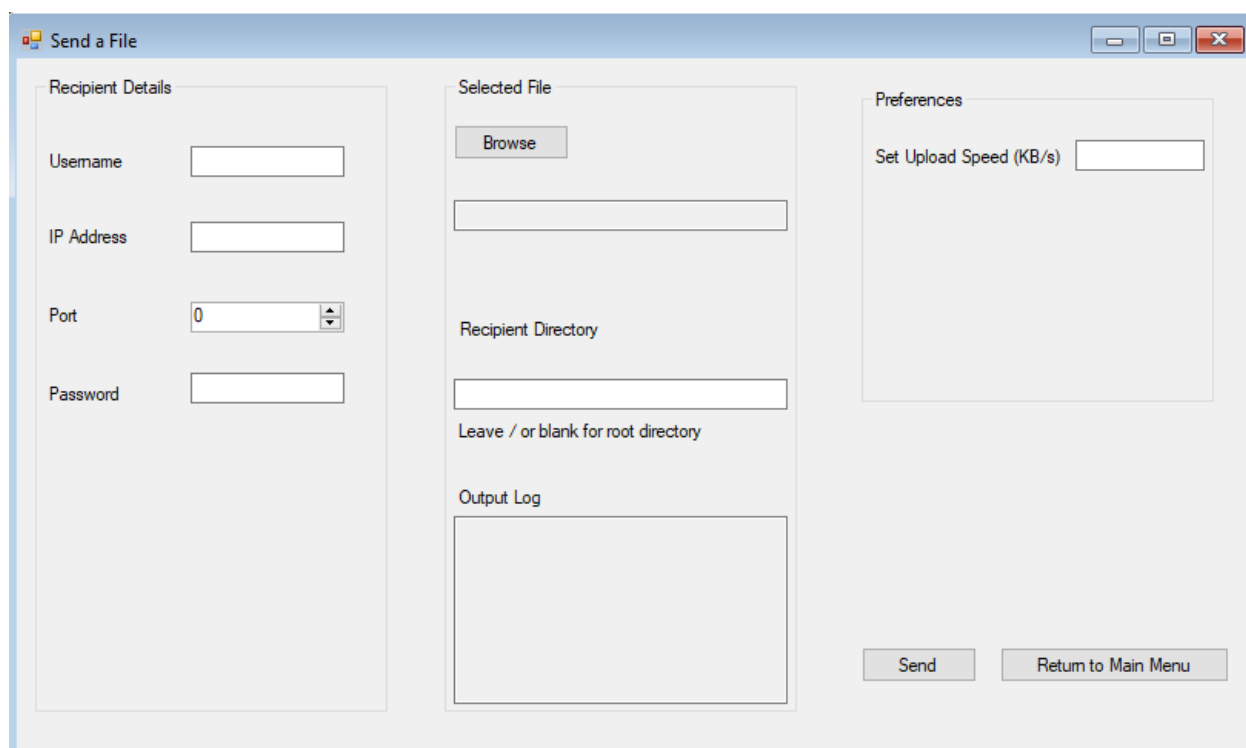


Figure 50 – Send a File Revision 3

Just like in 'Receive a File' I decided to implement an improved catch statement which iterates on each error and is more readable than before, see Figure 31. This ensures that regardless of input, the program will not crash, this makes the program much more resilient to unexpected situations. I have also added a `session.Close()` function after the user is notified of a successful transfer to prevent unnecessary bandwidth and communication.

For 'Receive a File' I have implemented several changes to the UI for a better user experience. Firstly, as the IP address entered is always going to be a local IP address, and local IP addresses follow the convention of 192.168.x.x. I decided to have the first two parts of the IP address filled out for the user so they would not have to re-enter the information every time. I have added a warning notification below the download location text box informing the user that anyone connecting will have access to all subdirectories by the

nature of how SFTP servers work. Therefore, a user who does not desire this can set the download directory to one with no subdirectories or contents, preventing the connecting party from accessing anything else. I have further implemented a close session button which is designed to terminate the session immediately preventing all new connections and severing all existing ones. Lastly, I have added a preferences section where I hope to implement additional features to the application at a later stage.

Figure 51 – Receive a File Form Revision 4

For the code I implemented an if statement which prevents an empty input for the download location which I found breaks the code, instead it now throws an exception safely preventing a crash.

```
if (Variables.DownloadPath == " ")
{
    throw new Exception();
}
```

Figure 52 – Download Path Error Check

4.6.2 Fixing Bugs and Addressing Security Concern

I had two major bugs, one on each form, which would crash the application for seemingly no discernible reason, and it was with a great deal of testing I was able to figure out the two issues.

The first issue is with the 'Send a File' form, when entering my own recipient directory, I noticed that a user who would either leave that directory blank or forget to put a forward slash after the final directory, would cause the application to crash or for a file to be sent without any type. It was the latter issue that I felt foul to and it prompted me to seek a fix. I originally struggled with how I could force the user to enter syntax correctly to prevent this issue, but I decided against that approach and instead designed a solution that would not break regardless of user input.

```

char last_char;
if (Directory.Length >= 1)
{
    last_char = Directory[Directory.Length - 1];
    if (last_char != '/')
    {
        last_char = '/';
        Directory = Directory + last_char.ToString();
    }
}

try
{
    if (Directory == "")
    {
        Directory = "/";
    }
}

```

Figure 53 – Directory Input Validation

Within the code to the left I would take any string that the user provided as a directory provided it was at least 1 character long and was not referring to the root directory, I would append a forward slash to the end of the directory to prevent issues. In the event that the user had left the field blank, I would simply change their input (or lack of) to a forward slash. Therefore, the code would no longer break due to faulty input.

The second issue was a little more serious and involved the failure for the `server.Stop()` function to properly stop the server. This led to issues where a new instance of the server could not be ran if the existing session had any incorrect information, such as the wrong password, attempting to call the function to stop the server seemingly had no effect and would cause a crash. Leaving the form would keep the application running in the background meaning that the session was not terminated, so creating another session on the same port would cause an error. Furthermore, accidentally pressing receive more than once would also crash the code as it would try to create two sessions on the same socket. To get around this I decided to disable the buttons for receive and returning to the menu, with the latter button being renamed to inform the user as to why the button is no longer functional. This would prevent leaving the server running in the background and it would prevent the execution of two concurrent servers.

```

button2.Enabled = false;
button1.Text = "Session Must Be Closed";
button1.Enabled = false;

```

Figure 54 – Disabling Buttons

I created the close session button as can be seen in Figure 51 with a warning message informing the user that this action will restart the application. Restarting the application leads to the destruction of the server and a new one can be created. I am not exceptionally pleased with the solution; however, it is the only one that I could get to work given the limited library available to me from the outdated software.

Finally, I would like to briefly discuss my security concerns. My application gives an external user access to the files and directories of wherever the recipient allows. This can lead to the potential for abuse if a user unwittingly provides too much access. However, I believe with my warning message, the fact that a potential attacker would need to know all the system network details, the username, password, and knowledge that port forwarding is enabled and the port to connect to. The risk is extremely low especially when one considers that the recipient has complete control over the username and passwords, and the availability of the server, being able to shut the server down if there any connections they do not recognise. It is also important to remember there are no accounts and no central server, it is impossible to know if and when someone would be using the application, therefore making it much harder for a bad actor to even be aware that a transfer is occurring. Therefore, I am satisfied with the security of the system.

4.7 Application Test of Core Functionality

It is now time to test both the sending and receiving functionality together, the success of this test will show that my artefact satisfies all success criteria with the exception of additional features which will be expanded upon in the next section. To conduct the test, I will be using two separate computers both running Windows 10. To simulate being on different networks, I will connect one of the systems to a VPN, this will ensure all requests go through the VPN server to my routers public IP address. This will be sufficient to make the connection seem like an external connection.

I began with compiling my current code into a release and sending it to my other system, along with all libraries to ensure that the two programs are identical and that the only variable would be the network connection. For my testing I will conduct 4 tests, the first test will involve sending a small file from my laptop to my PC approximately under 1MB, the file will then be inspected for corruption, I will use an image for this test. My second test will involve sending a much larger file approximately a half a gigabyte in size, this file will be a high bitrate video. The remaining two tests will involve swapping which PC is sending and which one is receiving to see if the test works both ways.

My program is designed to address the file size constraints posed by cloud services and I am aware that half a gigabyte is not a large enough of a transfer to demonstrate this capability. However, I will conduct a transfer of a much larger size in the testing section over my local network, this is because my upload speed from my ISP is only 4 MB/s and it would simply take too long to test efficiently.

I firstly began by connecting my laptop to a VPN and running both programs on my systems. For the first and second tests my laptop will be sending the files to my PC and so I filled out the relevant details for the file transfer on both systems. The image I will be sending will be a small toy duck from my laptop's desktop to a hard drive subfolder on my PC, the image is approximately 800kb and I will be testing for speed and quality of the transfer.

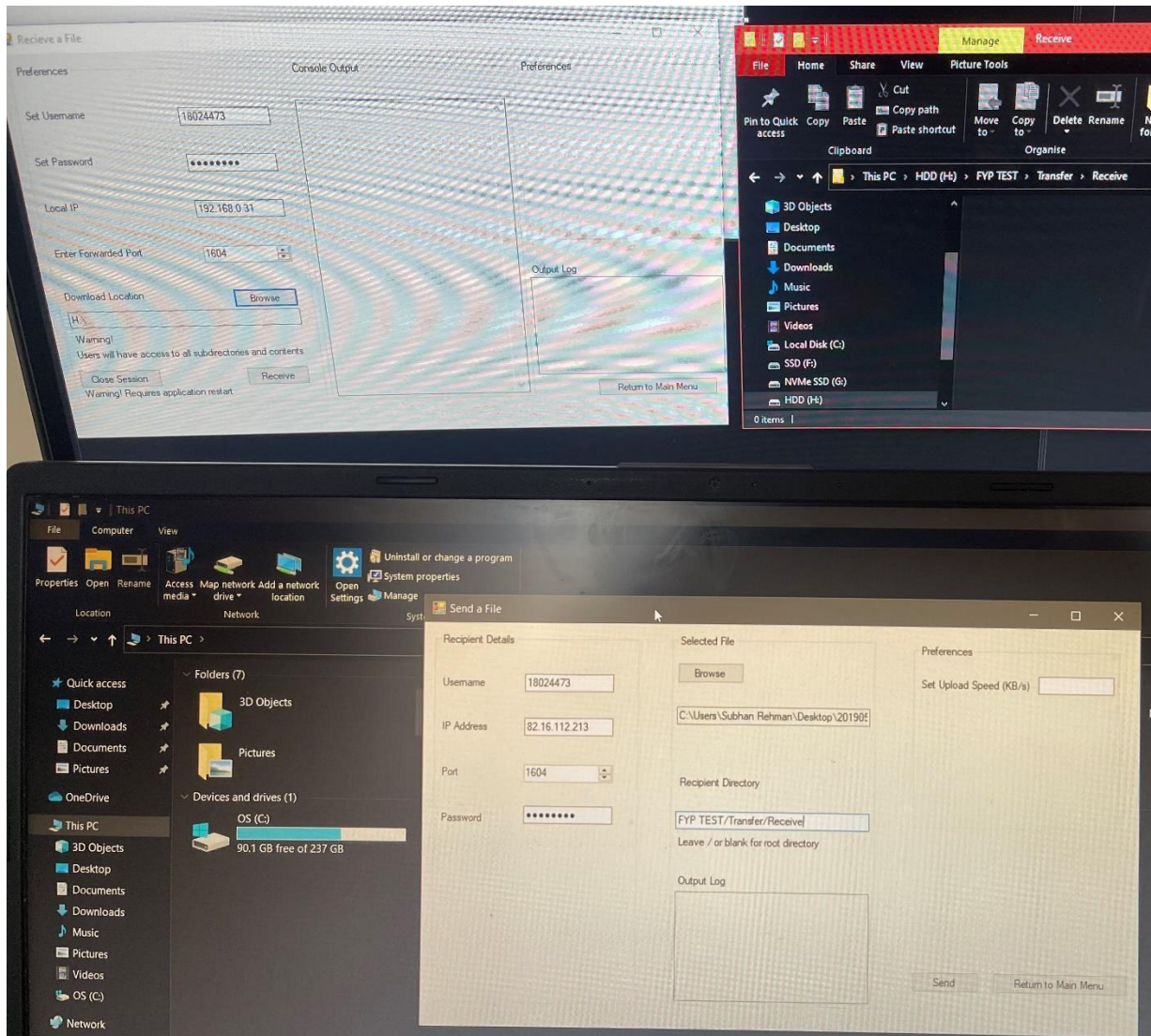


Figure 55 – File Transfer Before

As can be seen above the first test is ready to be conducted, I will be leaving the recipient directory field deliberately wrong to see if my if statements can detect and correct the error. After starting the server and pressing send I am pleased to report that the file sent successfully and instantly while retaining its full quality. My syntax error was also detected and rectified by the program. It can also be seen that the session was closed by my laptop after the transfer has been successfully completed as intended, this can help notify the recipient that it is safe to close the session as there is nothing being transferred. I then decided to begin the second test this time using some old dashcam footage as the file. The footage was in an MP4 format and was approximately 438MB in size.

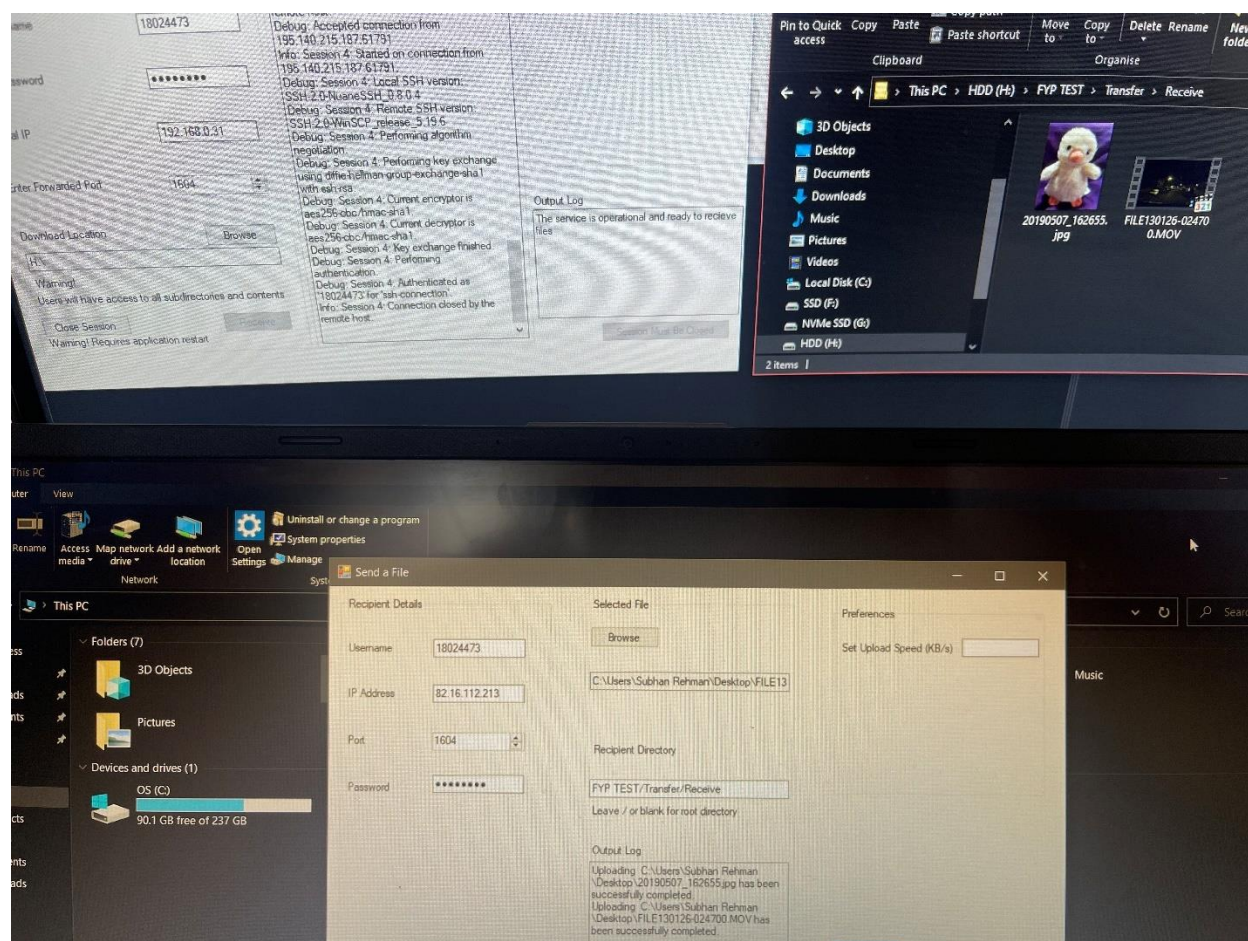


Figure 56 – Proof of Transfers 1

After pressing send for the second test there was a very long pause of around 2 minutes as the file was transferred, I am pleased to report that although the file was successfully transferred, due to this pause without any information or feedback presented to the user, it may seem as though the program had crashed. I will be implementing a way for the user to track the progress of the transfer in the next section and control the speed of the transfer to suit their needs and use case.

It is now time to swap the roles of the two systems, I will be connecting my PC to a VPN to send files and my laptop will be receiving, connected normally to the router and the internet. I will be sending the same files back but in this instance to a different directory on the laptop. If this test is successful, then my artefact is functionally complete as it satisfies the success criteria. I enabled port forwarding on my laptop through my router for this test as the receiving device must be able to be accessed from outside of its own network.

I am pleased to report that the test was a success and all files transferred properly and the quality of them remained intact. My next step will be to implement additional features into the software to make it more user friendly and feature rich. After this step I will test my code against the success criteria and go on to evaluate my project.

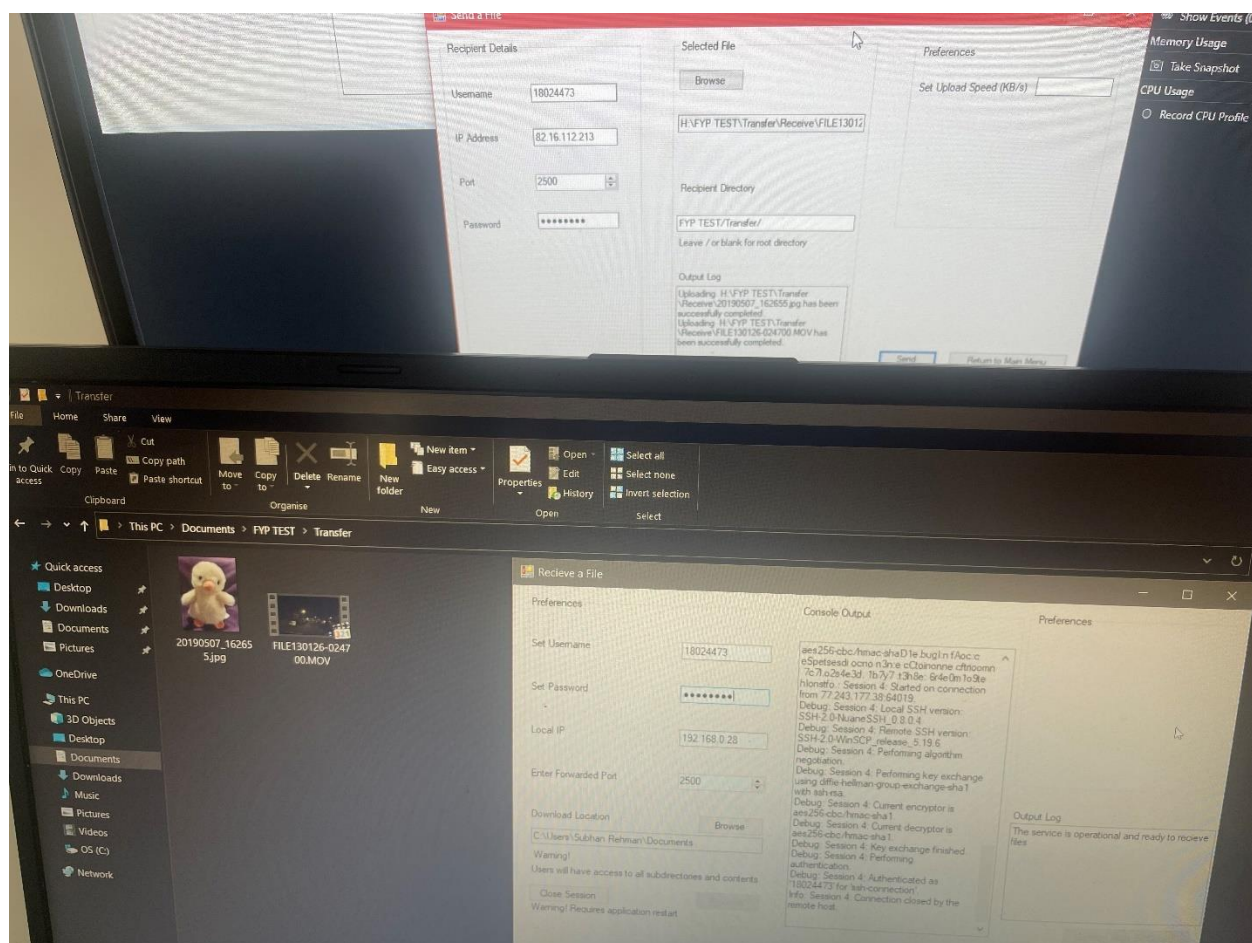


Figure 57 – Proof of Transfers 2

4.8 Implementation of Additional Features

In this section I will be implementing additional features to my program for making it easier to use and to provide a more responsive and pleasant user experience.

4.8.1 Implementation of Speed Control

Controlling the speed of the file transfer is something that is incredibly useful as often times it may be inappropriate to utilise all available bandwidth, there may be other services of more importance or other users to consider. Therefore, I will be implementing control over upload speed for my application. On the UI side of things, I will be using a NumericUpDown box much like the one found to prevent any non-integer input for the port number.

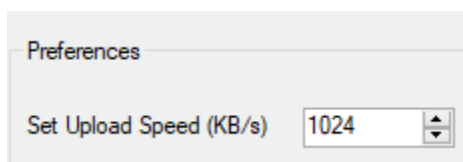


Figure 58 – Speed Control UI

I have limited the acceptable range of values from 1KB/s all the way to 1,250,000 KB/s which translates to 10 Gigabit per second. The reason I have chosen this value is because it is the upper limit of consumer network infrastructure. If the user decides to not enter a value, it will default to 1024 KB's which is 1MB. On most modern connections this is enough data to facilitate a respectable speed of file transfer without interfering with other services.

For the code I will be using the WinSCP provided 'SpeedLimit' parameter for the transfer options. The UI input will firstly be converted into a 32-bit integer before being applied to the options for the transfer while configuring the session.

```
int speed = Convert.ToInt32(Math.Round(numericBoxSpeed.Value, 0));
```

Figure 58 – Converting Speed Value

```
transferOptions.SpeedLimit = speed;
```

Figure 59 – Applying speed value

To test, I will send a file to a remote system while capping the transfer speed to 625 KB/s which is 5 Mbps. This result should be the upper limit of the network activity used by the application.



Figure 60 – Speed Control Proof

As can be seen above in task manage the reported upload speed was exactly the limit specified in the UI, this important feature will allow users to better manage their overall bandwidth and give them more control over their network.

4.8.2 Implementation of Progress Bar

As I discovered when transferring a large file between two of my systems was that it was impossible to tell, without closely monitoring network activity, how far along the data transfer was. This makes the application appear unresponsive for the duration of the file transfer. As one of Nielsen's Heuristics are

“Visibility of System Status” (Nielsen, 2020) and my program was not informing the user “about what is going on” (Nielsen, 2020) I decided to implement a progress bar. This would keep the user up to date on not only how far along the transfer was, but also the fact a transfer was in progress. This coupled with messages informing the user on the success or failure of transfers would provide adequate system feedback.

For the UI I used the progress bar built into visual studio from the toolbox and placed it near the output log. When the file had not transferred or just begun the bar would be empty and slowly fill up until the transfer had been completed. Within the code I used WinSCP’s built in library to use the FileTransferProgress() function which would take the current progress as a percentage of 100 and display that onto the progress bar which too had a range from 0 – 100.

```
session.FileTransferProgress += (send, x) =>
{
    progressBar1.Invoke((MethodInvoker))(() =>
    {
        progressBar1.Value = (int)(x.OverallProgress * 100);
    });
};
```

Figure 61 – Progress Bar Code

Below you can see the progress bar midway through a file transfer and after a file transfer. It clearly shows how far along the transfer is proving the user with real time information.

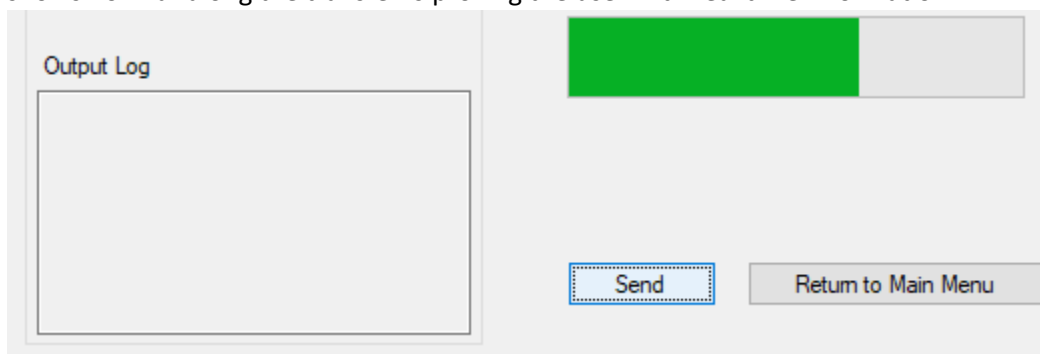


Figure 62 – Progress Bar in Progress

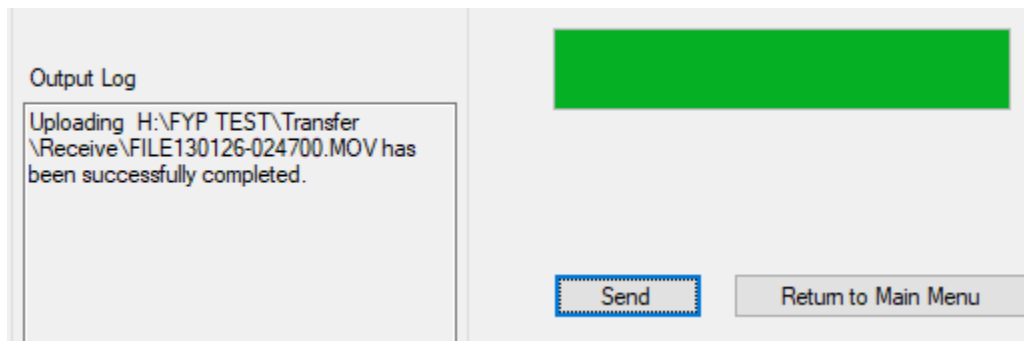


Figure 63 – Progress Bar Complete.

4.8.3 Send a File Final Revision

After implementing the additional features to my Send a File form I believe that it is in a state where I can declare it to be completed. I am satisfied with layout of the UI and from a user experience view I believe it is perfectly adequate.

The 'Send a File' window is divided into three main sections:

- Recipient Details:** Contains input fields for Username, IP Address, Port (set to 0), and Password.
- Selected File:** Includes a 'Browse' button, a text field for the file path, a 'Recipient Directory' field, and an 'Output Log' area. A note states: 'Leave / or blank for root directory'.
- Preferences:** Features a 'Set Upload Speed (KB/s)' dropdown menu currently set to 1024.

At the bottom right, there are 'Send' and 'Return to Main Menu' buttons.

Figure 64 – Send a File Complete

4.8.4 Receive a File Final Revision

I do not believe that I can currently add any customisable preferences to my Receive a File form as the function it serves does not allow for much customisability and the library in use is extremely limited with its functionality. However, I do believe that the form is satisfactory in showing the user all relevant output and keeping them informed about the state of the temporary SFTP server.

The 'Receive a File' window is divided into three main sections:

- Preferences:** Contains input fields for Set Username, Set Password, Local IP (set to 192.168.), and Enter Forwarded Port (set to 0). It also includes a 'Download Location' field with a 'Browse' button, a 'Warning!' message, and 'Close Session' and 'Receive' buttons.
- Console Output:** A large text area for displaying real-time output.
- Output Log:** A large text area for displaying a log of operations.

At the bottom right, there is a 'Return to Main Menu' button.

Figure 65 – Receive a File Complete

5.0 Testing

5.1 Testing Outline

Now that my implementation has concluded and I have a finished artefact, it is now time to test the success of said artefact against the success criteria originally created at the start of this project. For each success criteria I will discuss the degree of success achieved, how my work could be improved upon for that test, and the value that each test has for my project. I will also be testing my project against Nielsen's Heuristics for user interface design.

5.2 Testing Against Success Criteria

I will firstly test against the success criteria as this is the main metric of success for my artefact.

Create a stable application:

This success criteria was created as any application should run smoothly and without issue or error as the presence of these can severely damage user experience. This success criteria was fulfilled and my application is stable due to several exception clauses present ensuring that regardless of the error the application will not crash. The proof for this can be found in section 4.6. However, the caveat is that the application will reply with an error within the output textboxes I have created and while this is much more preferable to an application crash; a generic error message is not helpful to a user and they may find themselves struggling with diagnosing the issue.

I believe that preventing the user from executing a command without all fields filled out correctly is one step that be taken to prevent errors within my code. I believe that although this success criteria is a technical success, the presence of more relevant error messages are much more informative to a user as they inform the user better where the problem lies and how they could solve it.

Establish an initial peer-to-peer connection between the two systems:

This success criteria was created as the artefact is intended to operate without the need for a third-party server, therefore this success criteria reflects this by requiring that an initial connection is made between the two systems. This success criteria was fulfilled in multiple stages. I started by working on the ability to connect to systems using a peer-to-peer connection in section 4.4, later on in section 4.5 I worked on the ability to receive an incoming connection request from another system. The initial peer-to-peer connection was established in 4.7 between my PC and a laptop connecting to a VPN to simulate an external connection.

A peer-to-peer connection was established between the two systems that did not require the need of any third-party servers to help establish or maintain the connection in anyway. I would consider this success criteria to be completely met and fulfilled. I believe my approach to this success criteria cannot be improved upon as there is little room to improve on.

Establish an encrypted connection between the two systems:

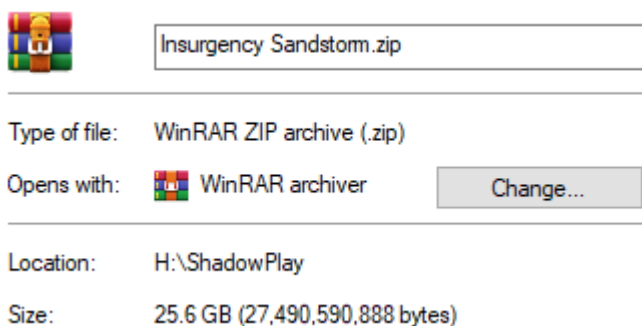
This success criteria was created as the artefact will be communicating over external networks to reach the recipient; therefore the data will be vulnerable to interception. If data is intercepted it may be read and this will cause a serious problem for the security of the solution. This success criteria was fulfilled in

multiple stages. Firstly, the ability to make an SFTP connection to a server was established in 4.4.2 where I was able to successfully connect to a remote SFTP server. I then developed my artefact to host a temporary SFTP server in section 4.5.1 so that it could be remotely connected to, to receive files. Finally in section 4.7 the initial connection was made between the two systems and this session was a peer-to-peer end-to-end encryption.

I believe that for this success criteria my solution demonstrates a perfect success. An end-to-end encrypted connection is established by the two systems which is further supported by a username and password to connect to the recipient system. I believe however that there is room to improve. Firstly, I could implement stronger username and password requirements to increase the security of the session. Secondly, I could do more research on encryption algorithms to ensure the strongest one is in use.

Establish a stable connection capable of file transfer between the two systems:

This success criteria was created as the artefact must be capable of sending the entire file without error otherwise it will serve no purpose and may lead to data corruption. In section 4.7 I send files back and forth across external networks to demonstrate the robustness of my solution and the stability of my connection. I was able to send files without any data loss or corruption using my application.



I previously mentioned that due to my ISP limited 4 MB/s upload I was not able to conduct a large file transfer over an external network and I would reserve a large file transfer over an internal one for testing. Below, I transfer 25.6 GB of recorded gameplay footage from my PC to my laptop over my internal network. This test is designed to see if my application is capable of handling large file transfers without running into problems.

Figure 66 – File to be Transferred

Below, you can see the state of both applications before I begin the transfer, all information is written in and the applications are ready for transfer. I have set the upload speed cap to 1 gigabit per second as that is the max speed of the CAT 5e cable my PC is connected to the router by.

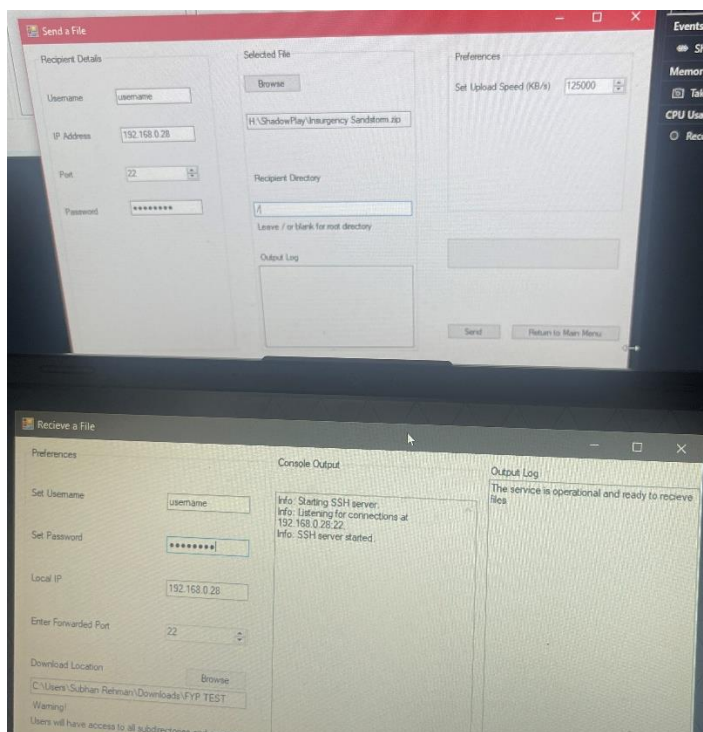


Figure 67 – Both Devices Before Transfer

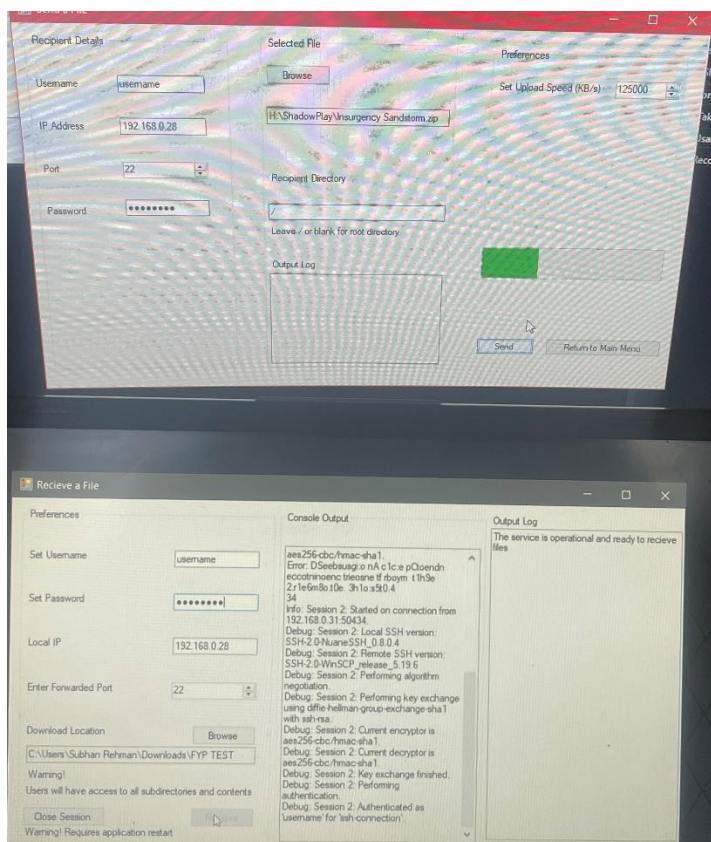


Figure 68 – Both Devices During Transfer

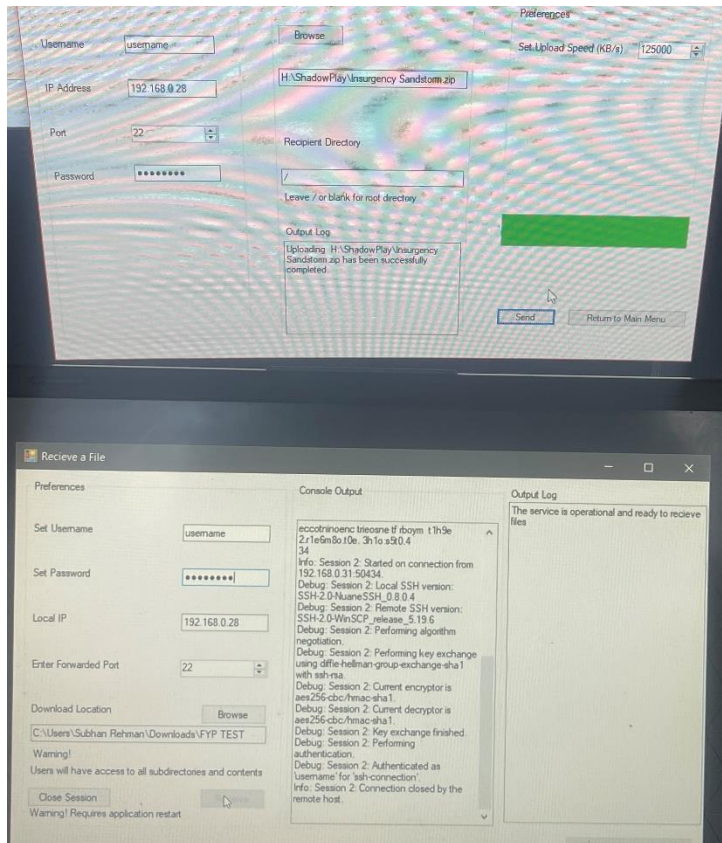


Figure 69 – Both Devices After Transfer

As can be seen above, the progress bar implementation worked successfully, and the application is capable of handling files of this magnitude. I have no reason to suspect that the application would not be able to handle a file transfer that is even larger provided both devices has the required free space available.

I believe for this success criteria I have mostly succeeded, while the artefact is perfectly capable of sending files between the two systems, it is unable to facilitate the transfer of multiple files at the same time nor is it able to pause a transfer in progress. I believe that the lack of these features, while not negating the success against the criteria, take away from the usability of the file transfer.

Implement additional features to increase the quality of the user experience:

This success criteria was created to place a focus on improving the UI of the system and to improve user experience. I consider this success criteria partially met. At several points throughout the program, I have taken steps to implement additional features and functionality into the program (see 4.6 and 4.8), however I do believe that more steps could be taken for the user interface. I believe my failure to implement more additional features stemmed from my focus on delivering a more suitable and fleshed out core solution, giving less priority to the additional features or the quality of life of the solution.

I believe my artefact can be improved upon in the following ways. Resizable UI and text for accessibility, additional user preferences for more control over transfers, possibly a contacts list so that users do not need to manually enter the details of the recipient on every instance. I believe that as the fields for

receiving a file typically would not change there should be a way to implement some continuity between application instances and save that information for next time.

Transfer a file from the host system to the recipient system securely and without utilising a central server:

This success criteria was created to consolidate the other success criteria to one singular metric which would determine the overall success of the project. After considering the artefact, the way connections are established, the nature of those connections and the security of the system I consider my project to have met the success criteria satisfactorily.

This is not to say that there is no room for improvement, I believe there is plenty and within my evaluation and conclusion I will go into greater detail when it comes to this. However, I do believe that as I have achieved what I set out to do, and that an encrypted peer-to-peer transfer is indeed now possible through the use of my application that the project can be considered a success.

5.3 Testing Against Nielsen's Heuristics

I will now test my artefact against the 10 heuristics for user interface design (Nielsen, 2020)

1: Visibility of system status

My application succeeds against this heuristic as seen in 4.5.1 and in 4.8.2 through the implementation of the console output and a progress bar to show the state of system status. This can be improved with a more detailed progress bar for both receiving and sending.

2: Match between system and the real world

My application partially succeeds against this heuristic as there is lots of jargon within the console output (as seen in 4.5.1), this is incomprehensible to someone who is not familiar with servers. However, I have used simple and clear wording and error / success messages throughout to make the code more approachable. This can be improved by editing log output to be simpler and more readable.

3: User control and freedom

My application succeeds against this heuristic throughout my program in all cases aside from one. When sending a file, the application will not be able to be interacted with, despite the progress bar working fine. This traps the user from being unable to stop the transfer and should be addressed by running the transfer on a separate thread and not freezing the UI.

4: Consistency and standards

My application succeeds against this heuristic as it follows naming conventions used across windows and the wider technological world, the names of text boxes as well as buttons are clearly labelled while being short and simple. My improvement for this would be the same as heuristic 2 where I would improve the console output.

5: Error prevention

My application fails against this heuristic as it is incredibly prone for users to enter situations where they enter into situation where they will encounter an error. While the application will not crash on the user

the lack of specific feedback for a failure harms user experience and an application designed around dealing with user input to prevent failure would be a recommended solution.

6: Recognition rather than recall

My application partially succeeds against this heuristic as while there is little memory required for the use of my application, field do not remember what was placed in them the next application restart and it may be tedious re-entering IP address' for the more casual user. To improve a contacts list will need to be implemented with user information.

7: Flexibility and efficiency of use

My application fails against this heuristic because it does not have accelerators, personalisation or customisation. While this is mostly due to the simplicity of the software, it can be greatly improved upon by implementing a greater selection of preferences.

8: Aesthetic and minimalist design

My application mostly succeeds against this heuristic as the design is incredibly functional, with forms and buttons located where the average user would expect them to be. (See 4.6) I would improve this by introducing better graphics and possibly rebuilding the application in something other than forms that can support that format/

9: Help users recognize, diagnose, and recover from errors

My application partially succeeds against this heuristic as error messages, while present, provide no information that would help the user track down the problem. This can be improved by adding text to the output next to each exception so that the user is aware where the failure occurred and why.

10: Help and documentation

My application partially succeeds against this heuristic as while there is no documentation. The application explains itself and what it does well enough that a user that has never interacted with it before could do so with ease. To improve this, I would create a user guide on the home page that users could consult.

6.0 Evaluation

In this section I will evaluate the success as the project as a whole; I will talk about the problems I faced and what my main takeaways are from this project. I finally talk about future work that can be done to my project and if done so, the effects it may have on the wider computer science community.

6.1 Evaluation of my Performance and Approach

I believe that my overall performance throughout the duration of the project has been satisfactory. I have successfully met all my success criteria in a timely manner and my application runs smoothly and is a suitable solution to the problem. While my application is flawed in some places, it could do with more polish and a much better error feedback system, the core functionality is present and the application does not crash or behave in an unexpected manner.

Throughout the project I have developed my skills and familiarity with networks and C# allowing me to slowly start with the basics of my application to work up to making my application communicate with another instance of itself on another device. At the start of this project my knowledge and experience with C# was limited and I feel that now I can competently develop and code with ease in C#. I previously had no experience working with Windows Forms and I am pleased to say that as a result of my project I am extremely confident working with the framework to develop applications. Throughout the implementation I conducted frequent tests to see if my application is capable of the functionality that I have coded into it.

My approach to the project and my implementation I believe was incredibly well executed. I began by defining clear aims and targets to achieve and over the course of the background research and implementation I was able to clearly explain the relevancy of what I was doing and how it related back to the tasks I had yet to complete. I presented what I was currently doing and how it would lead to my next steps within my implementation. My feature driven development allowed me to develop my code in a way that focussed on the success criteria; this made my implementation a logical progression where I would constantly iterate the code I already had, to add new features and conduct new tests to ensure that everything was still working.

I believe that centring my approach around the success criteria, which in itself was ultimately centred around my motivations, allowed my project to keep focus of its initial targets and allowed for the completion of all core project behaviour. This also helped with the implementation phase where I focussed on cleaning up my code and UI until after the core features had been established.

In conclusion I would consider my performance and approach to be very satisfactory and I am proud that the methodology used, and the allocation of my resources led to the successful completion of my artefact.

6.2 Evaluation of the Success of my Learning Objectives

At the start of this project, I identified 4 learning objectives that I wanted to develop upon in order to assist me with my personal development within networking and to aid me with the completion of my project. Below, I have made a table evaluating my degree of success in achieving each learning objective, as well as where I have demonstrated the expansion of my knowledge and how I can further improve in the future.

Objective	Degree of Success in Achieving my Learning Aims	Where I Have Demonstrated this Improvement	How Can I Further Improve
Expand knowledge on establishing network connections	I believe I have expanded my knowledge on establishing network connections significantly.	I have demonstrated my knowledge throughout my implementation firstly establishing connections to FTP servers, to establishing encrypted connections with external servers. This eventually led to a peer-to-peer encrypted connection between two devices on two different networks I was able to code and establish.	I can continue to create applications where connections can be established with multiple devices or where the connections established have needs outside of file transfer. I can also continue to flesh out my solution to make the connection establishment more painless and easier for the user.
Expand knowledge on network protocols	I believe I have expanded my knowledge on network protocols moderately. However, I have expanded my knowledge on how to use protocols I was already aware of and for those protocols I have gained massive experience working with them and configuring them for use.	At the start of my implementation, I utilise FTP and then SFTP the latter of which I go onto use for the duration of the implementation. I am able to comfortably work with SFTP and throughout my development I have shown many examples of using the SFTP protocol. Through my research I have also greatly expanded my knowledge on the use of TCP and the benefits it provides.	I can use different network protocols in future work and gain greater familiarity with the new protocols in that manner. Alternatively, I can expand upon the work I have done thus far and allow for multiple protocols to be used within my solution as well as incorporate my knowledge here into different applications in the future.
Expand knowledge on encryption	I believe I have expanded my knowledge on encryption to a lesser extent than the other objectives but still to a degree where I am able to more comfortably implement it	End-to-End encryption for my application was ultimately implemented however as the libraries had provided the algorithms it was a simple case of calling the functions required to establish the	I can tailor the parameters that popular encryption algorithms use in the future for more security and a better understanding of how they work. I can also implement multiple types of encryptions in

		encryption for me. I am pleased to say that my knowledge on fingerprints and key generation has significantly improved as I encountered it while attempting to establish connections.	the future into my application as well as conduct more research into different types of encryption algorithms and methods.
Expand knowledge on C#	I believe I have expanded my knowledge on C# very significantly and I have gone from incompetent and unfamiliar to very familiar and no longer daunted by complex C# code.	My working artefact is proof that I have greatly improved my knowledge and experience working with C#. I have followed good coding practice throughout my development, and I am able to confidently develop and code in C# and in the Windows Forms framework.	I can continue to develop upon my existing artefact which will inevitably expose me to C# increasing my experience with the language. As I develop my application to become more feature rich and advanced it will allow me to explore more into C# and how it can be used as a powerful language to do much more than file transfer.

6.3 Future Work and Further Development of my Project

While the initial problem has been addressed and the project has provided a suitable artefact as a solution to this problem, there is still plenty of work that can be done to increase the usability of the artefact as well as increase its scope and potential target audience.

Perfecting application polish and error messages. This can go a very long way into making the application look professional and extremely well made, careful attention to finer details within the program and UI can make the artefact run better as well as perform what is required by the user faster. The application is currently helpful in informing the user that a problem exists, however it lacks the ability to point the user to the exact issue and help the user with a solution. For example, if a user incorrectly enters their local IP address, a good solution will inform the user the problem lies there while a great solution may be able to find out the information itself and suggest to the user a potential fix with the correct data.

Increasing customisation and accessibility. At present the application is not customisable by the user and while the artefact does follow industry standards for its layout and naming conventions it is not exactly accessible with the UI being potentially confusing to a complete novice as well as not being resizable. This may cause issues for users' enjoyment of the application and the way they want to use it. Therefore, by increasing the level of customisation and accounting for greater accessibility the application will be of more value to those who may currently struggle with it.

Reducing the workload on the user. Implementation of features such as an address book of saved contacts or a persistent settings file which is able to save configurations and preferences between application instances will mean that the user will be able to use the application faster than ever before and more efficiently than before. These improvements can eventually stem to the application being able to retain more information such as a transfer list of files, the locations of previously transferred files and a more comprehensive list of preferences, perhaps on a per user basis.

Finally, I believe that improvements to security and encryption could be introduced by restricting the types of passwords that can be allowed, blocking IP addresses automatically after repeated failed attempts and by restricting access to other files and folders within the specified directory the application would provide users with more peace of mind when using the application to conduct file transfers.

With all these changes implemented in future work, I believe that my solution may have real value to the computer science community as a way to securely transfer files between any two systems in the entire world without worry of bandwidth caps, transfer limits, arbitrary speed limits and file restrictions. It would allow for uninterrupted flow of data in the scientific world and I believe this would be a welcome change from the world of file restrictions currently imposed by services such as Google Drive.

7.0 Conclusion

In conclusion all success criteria and learning objectives created at the start of the project were successfully achieved and the application fulfils not only its core functionality but, is able to provide some user preferences for the download location and transfer speed.

Throughout this project my skills in C#, a language I have seldom used outside of a single module in second year were tested, developed and improved to a level where I feel capable of handling the next problem with ease. Likewise, my skills in understanding networks have greatly improved through this project and the exposure it has provided to networks connections and file transfer.

My chosen methodology proved successful and the Gannt chart created was a valuable tool in keeping track of not only the task at hand, but what work was to come and the time that was left to do it in.

I believe my solution can be used in the real world and does remedy the initial problem with large file transfers that third party services require payment for, but it is not a solution I would recommend for novice users to networking as they may face some confusion with entering the exact correct syntax or configuring their router for port forwarding.

Overall, I consider the project a success as it succeeds in what it set out to do. However, there is certainly much work for improvement and refinement (as discussed in evaluation) and if completely fleshed out I believe my project may have real value in the scientific world.

8.0 References

- Altexsoft (2021) *The Good and the Bad of C# Programming*, Altexsoft. Available at: <https://www.altexsoft.com/blog/c-sharp-pros-and-cons/> (Accessed: April 2022).
- Apple (2021) *TCP and UDP ports used by Apple software products*, Apple. Available at: <https://support.apple.com/en-us/HT202944> (Accessed: April 2022).
- BISCHOFF (2019) *UDP vs TCP: What are they and how do they differ?*, comparitech. Available at: <https://www.comparitech.com/blog/vpn-privacy/udp-vs-tcp-ip/> (Accessed: April 2022).
- DROMS (ed.) (1999) *Automated configuration of TCP/IP with DHCP*. IEEE INTERNET COMPUTING. Available at: <https://ieeexplore.ieee.org/abstract/document/780960>.
- Hammad (2021) *Difference between WPF and WinForms*, Geeks for Geeks. Available at: <https://www.geeksforgeeks.org/difference-between-wpf-and-winforms/> (Accessed: April 2022).
- Hiter (2021) *End-to-End Encryption: Important Pros and Cons*, CIO Insight. Available at: <https://www.cioinsight.com/security/end-to-end-encryption/> (Accessed: April 2022).
- HUSPI (2020) *Everything You Need to Know About Push Notifications*, HUSPI. Available at: <https://huspi.com/blog-open/what-is-a-push-notification-and-how-does-it-work/> (Accessed: April 2022).
- IBM (no date) *What is a file transfer?*, IBM. Available at: <https://www.ibm.com/topics/file-transfer> (Accessed: April 2022)
- Internet Engineering Task Force (IETF) (2015) *Deprecating Secure Sockets Layer Version 3.0*. Available at: <https://datatracker.ietf.org/doc/html/rfc7568>.
- Lite, S. S. (2011) *SFTP Server Lite*, nuane. Available at: <https://nuane.com/sftp-lite/> (Accessed: April 2022).
- Lynn, R. (no date) *What is FDD in Agile?*, planview. Available at: <https://www.planview.com/resources/articles/fdd-agile/> (Accessed: April 2022).
- /n software (2022) *SecureBlackbox®*, /n software. Available at: <https://www.nsoftware.com/sbb/> (Accessed: April 2022).
- Nielsen (2020) *10 Usability Heuristics for User Interface Design*, Nielsen Norman Group. Available at: <https://www.nngroup.com/articles/ten-usability-heuristics/> (Accessed: April 2022).
- Ofcom (2021) *UK Home Broadband Performance*. Available at: https://www.ofcom.org.uk/__data/assets/pdf_file/0020/224192/uk-home-broadband-performance-technical-report-march-2021-data.pdf.
- Phifer (2000) *'The Trouble with NAT'*, The Internet Protocol Journal, 3(4), p. 2/44. Available at: https://www.netconf.co.uk/ipj/ipj_3-4.pdf.
- Přikryl (2022a) *Downloading and Installing WinSCP .NET Assembly*, WinSCP. Available at: https://winscp.net/eng/docs/library_install (Accessed: April 2022).

Přikryl (2022b) *License*, WinSCP. Available at: <https://winscp.net/eng/docs/license> (Accessed: April 2022).

Přikryl (2022c) *SessionOptions Class*, WinSCP. Available at: https://winscp.net/eng/docs/library_sessionoptions#sshhostkeyfingerprint (Accessed: April 2022).

Project Practical Editorial Team (2021) *Feature Driven Development Explained with Examples*, Project Practical. Available at: <https://www.projectpractical.com/feature-driven-development/> (Accessed: April 2022).

Putra (2020) *Visual Design : Nielsen's 10 Usability Heuristics*, Medium. Available at: <https://medium.com/@sef.gustingurah.yama/visual-design-the-face-of-your-apps-67a03427824f> (Accessed: April 2022)

Rebex (2022) *Rebex File Server*, Rebex. Available at: <https://www.rebex.net/file-server/> (Accessed: April 2022).

Roomi (2020) *7 Advantages and Disadvantages of Peer to Peer Network | Drawbacks & Benefits of Peer to Peer Network*, hitechwhizz. Available at: <https://www.hitechwhizz.com/2020/11/7-advantages-and-disadvantages-drawbacks-benefits-of-p2p-network.html> (Accessed: April 2022).

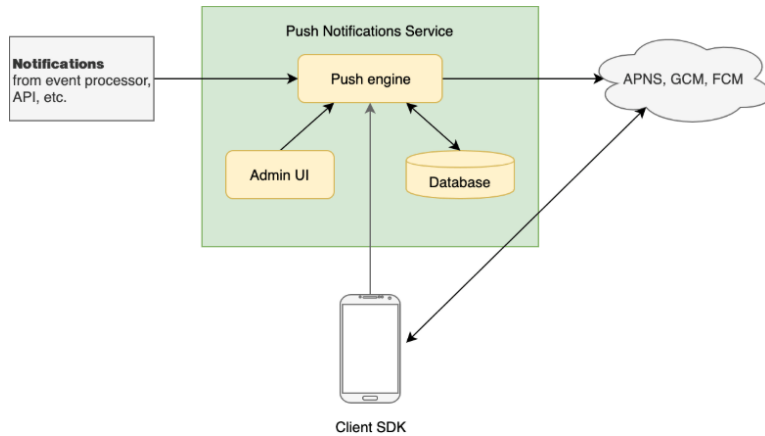
Secure Shell Working Group (2006) *SSH File Transfer Protocol*. Available at: <https://datatracker.ietf.org/doc/html/draft-ietf-secsh-filexfer-13>.

Verma, K. A. J. (2018) *Extending Port Forwarding Concept to IOT*. International Conference on Advances in Computing, Communication Control and Networking. Available at: <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8748430>.

Wood (2011) *The Clock Is Ticking for Encryption*, Computer World. Available at: <https://www.computerworld.com/article/2550008/the-clock-is-ticking-for-encryption.html> (Accessed: April 2022).

9.0 Appendix

9.1 Push Notification Chart



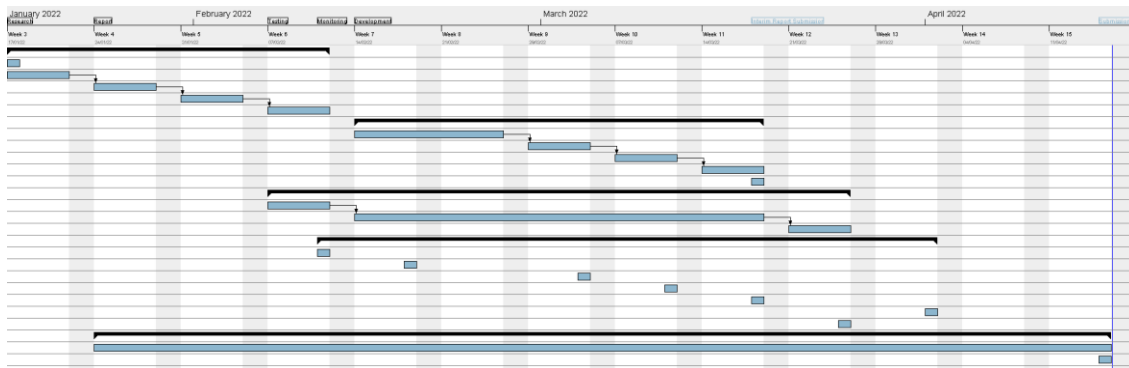
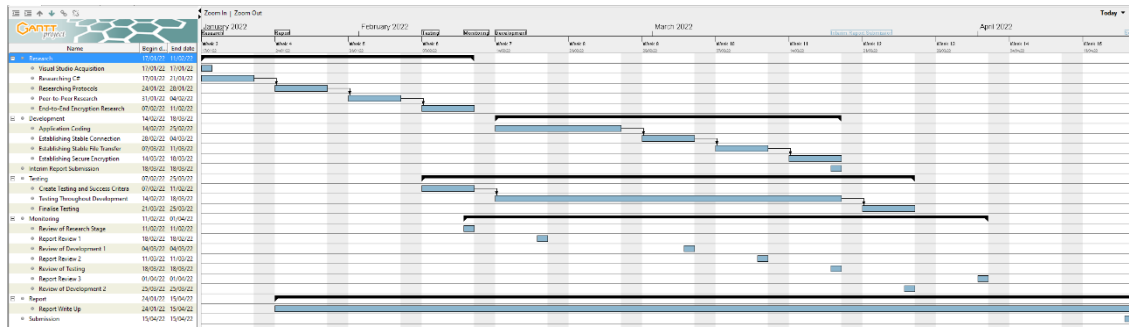
Push Notification Flowchart (Huspi, 2021)

9.2 Router Configuration

Local		External			
IP address	Port range	Port range	Protocol	Enabled	Delete
192.168.0.31	50000-51000	52000-53000	TCP	<input checked="" type="checkbox"/>	<input type="checkbox"/>
192.168.0.31	32400	29771	TCP	Automatically added by UPnP	
192.168.0.35	9308	9308	UDP	Automatically added by UPnP	
192.168.0.29	58989	58989	UDP	Automatically added by UPnP	
192.168.0.15	9306	9306	UDP	Automatically added by UPnP	

My Personal Router Configuration

9.3 Gantt Chart



Name		Begin d...	End date
Research		17/01/22	11/02/22
Visual Studio Acquisition		17/01/22	17/01/22
Researching C#		17/01/22	21/01/22
Researching Protocols		24/01/22	28/01/22
Peer-to-Peer Research		31/01/22	04/02/22
End-to-End Encryption Research		07/02/22	11/02/22
Development		14/02/22	18/03/22
Application Coding		14/02/22	25/02/22
Establishing Stable Connection		28/02/22	04/03/22
Establishing Stable File Transfer		07/03/22	11/03/22
Establishing Secure Encryption		14/03/22	18/03/22
Interim Report Submission		18/03/22	18/03/22
Testing		07/02/22	25/03/22
Create Testing and Success Criteria		07/02/22	11/02/22
Testing Throughout Development		14/02/22	18/03/22
Finalise Testing		21/03/22	25/03/22
Monitoring		11/02/22	01/04/22
Review of Research Stage		11/02/22	11/02/22
Report Review 1		18/02/22	18/02/22
Review of Development 1		04/03/22	04/03/22
Report Review 2		11/03/22	11/03/22
Review of Testing		18/03/22	18/03/22
Report Review 3		01/04/22	01/04/22
Review of Development 2		25/03/22	25/03/22
Report		24/01/22	15/04/22
Report Write Up		24/01/22	15/04/22
Submission		15/04/22	15/04/22