Python Chat Room Application

Adam Varden

B.Sc.(Hons) in Software Development

May 10, 2021

Final Year Project

Advised by: Andrew Beatty
Department of Computer Science and Applied Physics
Galway-Mayo Institute of Technology (GMIT)

GMIT
INSTITIÚID TEICNEOLAÍOCHTA NA GAILLIMHE-MAIGH EO
GALWAY-MAYO INSTITUTE OF TECHNOLOGY

Contents

List of Figures

About this project

Abstract This Dissertation is about the development of building a chat room application with the use of a set of technologies such as the coding language Python, socket programming and NoSQL databases. Each chapter details the progress of the development of the application and the research that went into it along with my full evaluation of the finished system. The aim of this project was to be able to pick a set of technologies, research them and using that research be able to apply the knowledge gathered from them to create the chat room application.

Authors Adam Varden is currently in his final year of the B.Sc.(Hons) in Software Development in Galway-Mayo Institute of Technology currently already possessing an ordinary degree in B.Sc. in Software Development.

Chapter 1

Introduction

1.1 Context

My project incorporates the programming language Python to create a server client relationship using socket programming to communicate with one another along with a NoSQL database communicating with the server and a graphical user interface using the python package tkinter to display and search visited users. The concept and purpose of this application is a chat room allowing users to join the chat room and communicate with one another with the added feature of being able to share files over sockets with ease. Here is a brief explanation of Socket programming it is a process of sending data over a network they provide a form of inter process communications (IPC) which is a mechanism an operating system provides in the management of shared data. IPC is mainly to do with clients and servers where a client requests data from the server and the server sends back a response.[3] The network that sockets work with can be logical, local networks or wired networks which are physical networks. Further research and detail is covered within my Technology Review chapter. These technologies are still now a strong area of interest in computing with Python being one of the top three most used languages, sockets being a form of network communication which is what allows everyone in the world to be able to communicate from all sides of the globe and finally databases being the source of storing data and knowing more effective ways to store data is always going to be an area of interest for developers.

1.2 Objectives

My objectives for this project is to expand my knowledge on the popular coding language Python which is vastly used within the computer industry and further my knowledge of an area of interest which is networking that is tied into socket programming which is a network communication programming style and to familiarize myself with the NoSQL database Mongo as it is a competitor and solution to the relational database further comparison found within the Technology Review chapter.

My objectives for my programming project are to create a server client relationship using socket programming, to have the server listening on a socket for incoming connections from multiple clients and have messages be easily distributed among those clients, have a user interface keep messages up to date among all users, allow for users to be able to effectively send files over the sockets to one another, have the server record the users that connect to the chat room in a database, have a server side graphical user interface and using the server side user interface to get the contents of the database and allow for the server administration to search the database for a specific records. The metrics in which failure and success is measured is based on the objectives and how many of them I meet by the end of the project period.

1.3 Chapter Description

The chapters in this dissertation are Methodology, Technology Review, System Design, System Evaluation and Conclusion. The Methodology chapter I will be discussing how I went about my project. This chapter elaborates on the life cycle of my project and what went into its development such as describing each component of the project and further breaking them down into small sub components that were needed to be implemented, how I went about testing these functionalities to make sure that they worked successfully. The chapter of the Technology Review discusses in detail each piece of technology I used for my project this being, the coding language python, NoSQL databases and socket programming at a conceptual level. I discuss comparisons of coding languages against Python, comparing relational databases versus NoSQL databases and on the current state of sockets and protocols implemented for the protection of data. The System Design chapter discusses the system itself, and how I used the knowledge I learned from the technology review to create my project and how the components interact with one another. This chapter will display the different user interface components and how they look to the user. The chapter also provides detailed descriptions with sequence diagrams of how the application operates and how the server handles creating new threads for new clients. The System Evaluation chapter details my testing on my application and limitations I found during the progress of my development of this application My conclusion summarises the objectives and context of the project what outcomes came from this project such as what I learned from doing this project, did I learn anything that I did not think I would learn from taking on this project and opportunities that may have been discovered for the future of other projects. My GitHub link can be found using this reference [4] in the References section at the end of this paper. Within that repository you will find a server python script and a client python script and a markdown README file. Follow along the instruction provided on the GitHub and view the video of how the application works.

Chapter 2

Methodology

2.1 Planning

When beginning my application which is a chat room application, I needed to decide how the users were going to connect to one another. I had looked into packages in python such as flask and networking packages such as sockets. I decided to go with sockets as it incorporated the topic of networking which is an area of interest so to expand my knowledge to a practical level was a desirable aspect in my choice. When researching the python language as my choice of a coding language. I was looking at its popularity in the industry and its desirable traits as a coding language. All of my code was documented on GitHub with detailed commits and information on how to run my application.[4] When dealing with issues along the way in my development process I took an approach of putting a hold on the particular component and working on another section of the project this proved beneficial when later returning with a clearer mind to tackle the issues at hand.

2.2 Concept and Requirements

The chat room app would be a meeting place using the IP address and port number to connect to the server. I then began my project by listing out the smallest parts of the project and grouping them based on what component they belonged to such component headings were client-side functionality, server-side functionality, client graphical user interface, server graphical user interface, server database relationship and server client relationship.

2.3 Development

When developing I chose to use an incremental Agile development approach to my development of this project. Having broken my requirements down into smaller components allowed for it to be easier to build up my application.

The first step in my project was the creation of the server to bind it to an IP address and port so it can listen to incoming connections. It currently is set up running on the local host using the IP address of 127.0.0.1 and the port number of 33000. When having my code run successfully to test that my server was up and running, I used a curl command with the servers details to see would it give me a response and it did so my server was up and running.

Once I had established that the server was running successfully, I then went on to making a server client relationship by having my client-side program connect to the server using the IP address and port through the command line and receiving output from both client and server that a connection has been established.

Currently I have one server client relationship, I needed to allow for multiple connections by creating a thread for accepting new connections from there, a new thread is added to handle a new individual client. After further investigation into threads and how they worked I was then able to create a new thread for each client. I needed to create a while loop that listened to new connections which created a new thread for that specific user to send messages to. Once I had multiple clients being able to connect to the server the next step was to allow messages sent by a single client to be passed to all of them, I needed to figure out a way of broadcasting these messages. This I did by looping over the connecting addresses of each on of the clients and sending out the messages for them all to see. I stored the addresses and client names in lists that had matching indexes.

Having established a basis of a multiple client and server relationship I then went on to looking into building a graphical user interface. Firstly, I went to the python docs in order to find what graphical user interfaces were available to me and took a liking to the tkinter user interface library. Before beginning to integrate the tkinter graphical user interface I took to google scholar to find books on python and tkinter programming and found research about the module before integration. Once gathering a knowledge of the library I started off by creating a basic message list for the incoming messages, an input bar for typing in messages and a send button linked to a method called send that when called sends the message that was typed into it through the socket. I created a receive method which loops and listens for incoming messages from the server and immediately adds them to the message list in the user interface that possess its own thread by appending the message list. Currently at this stage the input from the IP address and port were being inputted through the command line to remove this I started by adding tabs using the notebook feature on tkinter to create a tab for inputting the IP address, port and the name and a tab for the chat. I decided to use the name in this tab because the name should be the first message sent to the server to give a prefix for messages and a notification to other users within the chat that a new user has joined e.g., “Adam has joined the chat” and “Adam: Hi”.

Once having a basic user interface for the client side, I began testing by using print lines in the command prompt to track each part of the application. This stage in development has proven successful so far. When dealing with a client leaving the chat room an on closing method was made to send a unique protocol message to the server and alert the other clients that “Adam has left the chat”, this unique code is simply “quit”. I created an if statement in the server to check every message for this quit protocol. At this stage of the development progress has been made in the following components client-side functionality, server functionality, client graphical user interface and server client relationship. When presenting my progress on my project during Christmas presentations I was offered criticism on adding in some form of file sharing system to send to all users this feature I then added to the server side functionality component and client side functionality when adjusting my project break down.

When beginning how I would incorporate the file sharing feature I first off went to break down what was necessary for file sharing and the sort of process I would need to do for file sharing to be successful such processes were to allow the user to locate the file and save the path, read the file and convert it into bytes to share over the socket. I started off by creating a new tab in the user interface for file sharing adding buttons to select a file and send a file, once the button to select a file had been selected a file explorer window appears allowing the user to navigate to the desired file they wish to send. Once the file has been selected the path to that file is stored. Once the button to send a file is selected the file is read into bytes and sent over the socket. To prevent issues with sending messages a file share mode variable was created having a true or false value depending on if the user is going to be sharing a file. When sending a file, a protocol value is sent to the server before the file is sent to alert the server and other users that a file will be sent this values purpose is to change the other clients file share mode to true along with the server. Moving to the server an if statement needed to be created to check for the code of “fileshare” and to follow a necessary order when dealing with this file sharing scenario. Now that the sending client and server are aware of what needs to be done when sending a file, the most difficult part was receiving the file. When dealing with receiving the file I needed to tell the receiver client to enter its own file share mode when first developing the code which would create a file I was having it replace the created file every time for testing purposes but for the first while the messages sent after a file would be shared was being added to the file or the file would not add the content until the client and socket connection had been terminated. After several weeks wondering if it was the sending method that was causing this issue I traced the entire process of sending the file using print methods monitoring which values were being sent and stored until I eventually realized it was the receiving method which was causing the issues the issue being that the receiving method provided by the socket module was remaining open in the writing to the file method and would keep it in a loop until the connection was terminated. Once I added the bytes that would have been received into a local variable instead of receiving it directly in the writing method and creating it from the stored data I was then receiving the correct functionality that I was looking for. Once having the correct values being created I then added that the name of the sending file this would be sent before the file so that when the users would receive the file it would have its original file name created in the directory where the application is being ran.

I then moved onto working on my database component of my application. Before I may begin working on this component, I needed to research the sort of NoSQL databases I wished to use and the document style database MongoDB proved to be most popular in my research. I then went onto downloading a community version of Mongo onto my machine so my python script would be able to interact with it. Using the package pymongo [5]from python I am able to interact with the Mongo database that I will use for recording users that come into the chat room noting their IP address, port, name used for entry and time visited. I downloaded the Mongo community program to run the “mongod” command to give me access to create the Mongo database. When first launching the server, it checks if that particular database exists and if it does not exist it will create it for us. Once creating the database, I then went onto inserting data into it as soon as the client connects to the server this proven to be successful. Having a database running and inserting records to the database I went onto working on the graphical user interface for the server. I planned out the design based off the client-side user interface creating two tabs one tab for viewing all the database contents and a second for searching the database. When creating the user interface I ran into difficulties where the user interface clashed with the method for accepting clients after a bit of time looking into this issue and referring back to my research on threads I realized that I needed to make a new thread for the user interface this proved to resolve my issue. The first tab I began working on was the tab that displayed all contents of the database by creating a method for retrieving the contents and adding them to the user interface labels along with this I added a refresh method to link to a button which would on click refresh the contents of the server for the user. The second tab works on searching for the database for a specific record this works with a input bar and a drop down menu which specifies what category of record you are looking for this being the name or date, once entered the desired input and selecting the attribute then pressing search the user interface displays the contents of the with the specified value.

2.4 Version Control

In order to maintain my versions of code I used GitHub to backup and update my code during the course of the project development. Providing detailed commits describing the current state of the project, issues if present and other functionality yet to be implemented. GitHub was a very useful and essential tool because it provided a back up of my code in the situation of a mass alteration that would cause a cascading effect of errors.

Chapter 3

Technology Review

The technology review chapter provides my research on the technologies I incorporated into my project and further research on the state of each technology. The research gathered here was pivotal for my project success as it gave me a clearer insight into these technologies such as their advantages, disadvantages, and concepts. Within this chapter I will discuss the use of sockets, the main protocol used when dealing with sockets, the programming language Python, threads, python with its tkinter library, python versus the java programming language, NoSQL databases and the document oriented database model. The research gathered assisted in the program development previously mentioned in my methodology chapter.

3.1 The Use of Sockets

The first technology up for review will be the use socket programming. Socket programming began in the 1980's when development began upon the United States governments giving funding to the University of Berkeley to integrate the Transmission Control Protocol (TCP) and the Internet Protocol(IP) into a UNIX operating system. During the development of this project a group of researchers were able to develop an Application Program Interface(API) for TCP and IP networking communication called the socket interface. This API was then implemented into the UNIX operating system but for other operating systems like Microsoft it was implemented as a library. Socket programming can be developed in any language that has support for network communications. A network compromises of several devices such as computers and other devices that extend the physical networks like switches, access points and bridges. The server's job in this network is an important resource which manages several tasks such as file sharing systems and sharing processing power with other devices on the network it provides a service for clients on the network. The client's purpose is to request the serves resource this may include data or processing power. The client is a simple worker in the network and not as powerful as the server. Both servers and clients use sockets to read and write data over sockets in byte streams. See Figure 3.1 Sockets are a programming interface that is provided by protocols

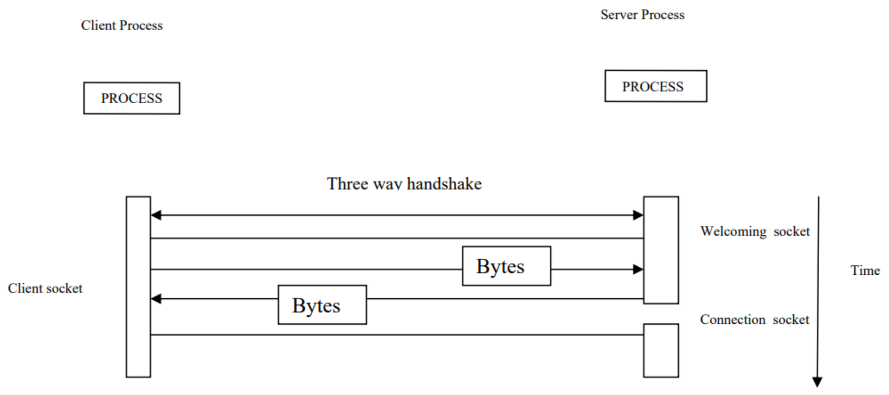


Figure 3.1: Three-way Handshake [1]

such as the Transmission Control Protocol(TCP) and User Datagram Protocol(UDP) for streaming communication and datagram communication of the transport layer belonging to the TCP/IP stack. When beginning development using socket programming the programmer is responsible for writing the code for both server and client because they are reliant on one another. A socket is an endpoint of an inter process communication flow across networks. The operations that sockets perform are as follows: connect to remote machines using IP addresses and their port that application is running on, send data in the form of bytes over to the remote device, receive the bytes data and close the connection between the two machines.

The port which combines with the IP to create a socket is an application specific software construct endpoint in a computers operating system meaning it is a port that a specific application uses on your machine it is an identifier for other applications that may wish to connect to it. Transport protocols such as TCP and UDP use ports to map any incoming data to the correct process running on a device using ports for direction. The Transmission Control Protocol connection is a direct virtual line between the client's socket and the server, the TCP protocol gives a guarantee that the server receives data from the client as it establishes a direct connection with the server and client. Socket programming that incorporates the User Datagram Protocol is considered unreliable compared to TCP because TCP establishes a connection while UDP uses an input address and sends the data this can cause issues such as data corruption without a fully established connection between client and server.[1] See Figure 3.2

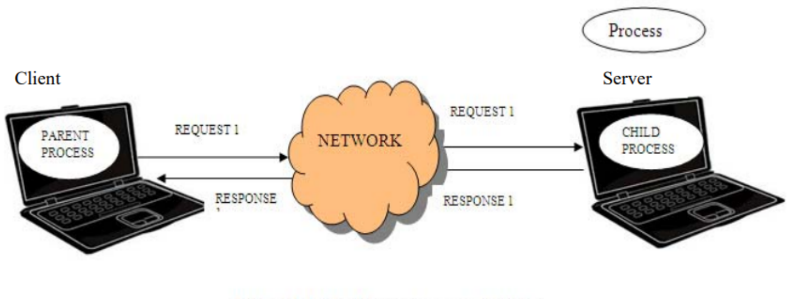


Figure 3.2: Socket Communication [1]

From researching about how sockets were developed and how they work internally on a logical level I was able to gain a better prospective when developing my server and client. I was able to grasp the relationship they have with one another and learned that I would be sending my data in the form of bytes of the socket connection and how the the sending and receiving of data process worked.

3.2 The Secure Socket Layer Protocol

After talking about how sockets work, I will briefly discuss what goes into protecting data when transmitting messages using sockets. One of the main protocols to protect user's data is the Secure Socket Layer (SSL) protocol. Why would we need this protocol well they ensure that sensitive data is protected, confirms the integrity of the data, can be used for authentication among users and security against unauthorized users and attacks that come with them. Such services like Virtual Private Network (VPN) are used to provide a stricter access control and protection of user data. SSL is a widely implemented protocol used in many E-Business organizations. When the first web browser was developed by the company Netscape they realized the connection between the client and server was essential and wanted to ensure its protection. The most effective way was to encrypt and decrypt the data at the end points of the connection. SSL was developed on top of the TCP layer providing TCP like interfaces to the upper-level applications. The advantages that came with SSL was that developers could make call using SSL rather than TCP calls. Further development came from Microsoft with their Internet Engineering Task Force (IEFT) to define a standard for an encryption layer protocol. Multiple vendors participated in working together to further this standard and protocol this resulted in the Transport Layer Standard (TLS) being formed. When dealing with the encryption and decryption process of a user's data key were made for this process these can be public or private keys. The first key type secret key encryption. Both sender and receiver possess the same private key used for encryption and decryption. Algorithms for encryption are DES (Data Encryption Standard), AES (Advanced Encryption Standard) and RC4. Problem attached to using these secret keys is regarding the distribution of these keys they need to be distributed over a secure network and if that network is not secure outsiders can learn the decryption algorithm and decrypt successful messages. The mathematical protocols attached to these keys are relatively straightforward DES using basic arithmetic, bit rearrangement and table look ups. The second type of key is public key encryption this key uses a public corresponding to a private key and vice versa. Draw backs with public keys are they are CPU intensive so taking more CPU power. Using this type of key adds in an authentication feature using the private key to send an encrypted message to the public key and if they can successfully decrypt the message it is verified that it is the correct receiver. [6]

3.3 Threads

When having to use threads in my application I took to this useful guide on how the threads work on a system while having a socket network for an example. I learned that threads are used everywhere including in all web servers and most Java graphical user interfaces. Operating systems have threads and monitor and manage them by the use of time sharing all these processes. When choosing the type of relationship of thread there is two types asynchronous and synchronous. Asynchronous works when there is no guaranteed order of how the messages will be received so it can handle messages in any order by creating a new thread for each client as previously stated on how I managed my multiple clients in my methodology. Synchronous works by handling a message and having to wait for the next message the following tasks cannot be executed until all processes has been completed. Threads are similar to a process but depending on the type of thread system it can be a process. Threads are a lighter form of process because threads use less memory than processes. What makes threads have an advantage over processes is the parent thread's ability to share its global variable with their child threads serving as a main function of communication. Learning about threads proved to be quite useful especially when learning that giving a new client a new thread can promote the asynchronous relationship.[7]

3.4 Python as a Programming Language

The next piece of technology up for review is the coding language python a popular coding language that has ranked in the top ten most popular coding languages consistently since 2003 in 2021 python has ranked third in the top ten with Java and C being the top two. Python is used by a variety of large organisations including Google, Facebook, Amazon, Instagram, and Spotify. Python first began as a high-level general-purpose programming language with its primary focuses on for developers to be able to write clear and logical code for both small projects and much larger projects. The development of python began in the 1980's and was then released in 1991. The sort of programming python supports is functional programming where focus is made on computation and list processing applications such as working with mathematical functions, aspect-oriented programming that aims to adaptable to changing situations , Meta programming which involves self-modifying code, and logic programming which involves logic circuits to controls facts and rules to control a domain. Python offers dynamic typing which belongs to the type system of the logic system of programming where usually with other languages you need to define a value with a type before run time but not the case with python, it allows for the developer to not have to declare a type and the type is only defined at run time by an interrupter. Python also provides a garbage collector allowing for memory management. Designed to be a highly extensible programming language on top of python's large standard library adding new feature can be added by the use of modules. Modularity has brought a variety of programming capabilities. How its syntax is formatted is that it does not require curly brackets and optional semi colons it uses indentation to format its code the increase in indentation means a code block while a decrease in indentation means the closing of a block the recommended indentation is four space. It uses dynamic name resolution also know as late binding this binds methods and other variables during program execution to further clarify dynamic name resolution we will compare dynamic to static. Static name resolution can prevent programming errors during compile time by catching the use of variables that are not in the scope while dynamic set and get variables in the same scope run time it can be seen with dynamic that it is not recommended by the python community because you would be trading safety for more flexibility.[8]

3.5 Python with Tkinter

When deciding on a graphical user interface I went with the standard library module Tkinter in python. Tkinter is seen as an easy user interface for developers to create quick user interfaces for prototypes. Tkinter is a python interface for the Tk package making tkinter the GUI toolkit for the service TCL/Tk a graphical facility developed by John Ousterhout. When first looking at the user interface module some say they expected it to fail but with modern computers now it proved to be quite beneficial and helpful in production. The module used to be a imported external resource but since the release of Python 1.5.2 tkinter became part of the standard library packages. Tkinter incorporates object oriented interfaces to the Tk package and Tk/TCL uses a command oriented scripting language. When working with the tkinter widgets they are referenced as objects making reading and understanding the code that much easier. Selecting such things as fonts and other system architecture designs there is no worry about when migrating to another operating system the design choices remain unchanged. The tkinter hierarchy is quite simple to the point that there is not a hierarchy the classes WM, Misc, Pack, Place and Grid classes are mixins to each of the widget classes. A mixin is in object oriented programming it is a class that contains methods that can be used by other classes accessing this classes varies on programming language.[9] The proposed hierarchy most find themselves focusing on the lower level of it and find it easy to forget about the higher level.[2] See Figure 3.3.

When researching python with tkinter programming this book proved to be a major asset in my development process at it gave me details not just about how to use tkinter but what is happening in the background of the library which allowed me to be able to manipulate it into doing what I want it to do. Tkinter proved itself in my development cycle how intuitive it was in creating a user interface. This provided me with a basis to be able to experiment using the different methods and options available by tkinter.

3.6 Python as an Object Oriented Language

When looking into a coding language an attractive feature is when they support the use of object oriented programming. Object Oriented Programming aids in solving complex issues by breaking down larger components into smaller sets of objects. Python supports the use of this style of programming because everything in python is an object as mentioned previously how every widget in tkinter is an object. Python's multi paradigm characteristic for

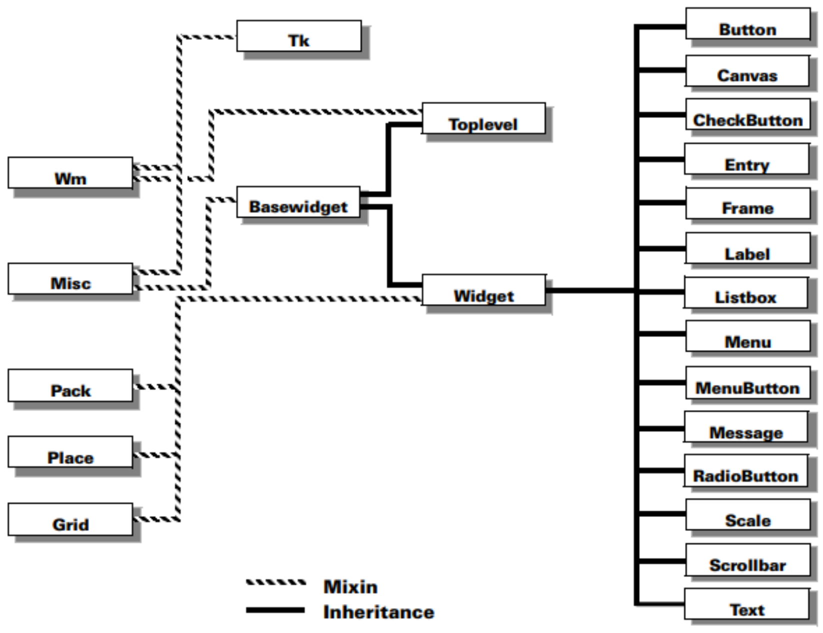


Figure 3.3: Tkinter Hierarchy [2]

support both object oriented programming and structure programming is a desirable trait in a programming language. Python possess great flexibility with its capabilities to be embedded into existing applications that require a programmable interface. Other supports python provides is web application development with MIME and HTTP, system programming being able to work with resources such as files, sockets, command line and shell programming along with a lot more. Further supports in other areas in IT fields such as game development, image processing, robotic programming, and artificial intelligence.[10]

3.7 Python Versus Java

3.7.1 Java

I have discussed python in a variety of aspects and shown how useful of language it is but it is not the top language used ahead of it would be java. Now I will give a brief comparison of the two programming languages naming their advantages and disadvantages. I will begin with Java, Java was first released in 1995 where it gained traction due to its trait for its platform independent properties write once, run anywhere (WORA), it is a compiled language that is statically type meaning the values are declared before assigning them a value. Java has been seen to run faster than Python but compared to C++ is much slower. The websites Airbnb, LinkedIn and Pinterest all use Java code. The sort of features that Java provides are that it is an Object Oriented language that allows for inheritance, polymorphism, encapsulation and abstraction. Platform independent with the use of a Java Virtual Machine (JVM) this makes compiling java code easy on other platforms simple. JVM allows for more security but lacks the use of pointers which encourage security. Java offers multi threading that is inbuilt into it and supports in the construction of complex and responsive applications that are required to perform multiple processes. Lastly its support of web based applications with use of servlets and Java Server Pages (JSP) allows for straightforward development and security of applications that require sensitivity e.g. social security and health applications. Such advantages that java brings to programming is that it is a strict typing and structure regarding code forcing good coding habits, it is easier to use, construct and debug java code than it would be compared to C languages. The characteristic of an Object Oriented programming language produces reusable code. As stated before Java's consideration to security provides safer applications. Java's extensive libraries are heavily tested for accuracy when used and Java comes with a strong community of developers providing solutions to nearly every issue possible. Java seems to have plenty of advantages to make it look like a very useful and intuitive programming language but its disadvantages would start with Oracle announcing in 2019 that the Javas standard edition 8 would be made only for business uses and not free to download. Java focuses heavily on storage and not on the backup of data making its memory management expensive with memory space. Compared to the C languages java is a much slower and memory consuming. Java's Graphical User Interface toolkit is seen to be a lot slower when ran on desktop applications and finally its extensive amount of code makes it harder to create and understand making simplicity and ease of understanding low.

3.7.2 Python

Since previously talking about python, I will now go on to talk about its advantage such as its beginner friendly standards in regards to writing code making it easier for people to start python compared to java that has more syntax rules. Python was developed with the objective to be clearer and easier to use and its extensive library makes creating application code much smaller. Python is open source and accompanied with a large community with solutions and diverse use of code to use. Python compared to other languages is great for the purpose of visualisation of data creating accurate and prestige charts and graphs. As state previously regarding pythons easier to write code create a smaller learning curve for users compared to other languages. Python runs its code line by line and in the situation of errors occurring only one error is shown even if multiple errors are present allowing programmers to handle errors one at a time and reduces a cascading effect of errors on other functioning code. Python as previously stated is a programming language with multiple paradigms offering object oriented, imperative, functional, structure and procedural programming techniques also offers multiple inheritance that java does not offer. Finally on to disadvantages compared to other languages python is slower in execution, it is not a favourable language for mobile development and an unlikely choice for comprehensive memory tasks. Python's dynamic typing can lead to run time errors causing restriction in the application design and further limitations in regards to database access layers compared to java with their library called JDBC (Java Database Connectivity). Finally with python being a relatively straightforward understanding language makes migrating to another language with stricter syntax much more difficult[11]

Each programming language has their own strengths and weaknesses while java has a more complex in terms of code structure compared to python and is more considerate of memory management compared to Java. Since I have already had a knowledge of Java from modules taken in this course Python was a strong choice as it was a language I had little to know experience knowledge in so to take on this language and learn it in greater detail was something I was up for.

3.8 NoSQL Databases

Next review is the use of NoSQL databases. First off let us talk about the limitations with the tradition relational databases. These relational databases such a SQL have a predefined structure such as a table with names and types with corresponding row and columns. These relational databases are limiting when it comes to scaling because it would usually require a stronger machine for storing the database in them this would cause having to make a distributed system and having the databases communicating with one another but these relational databases are not very effective in a distributed systems context, joining the tables from a different server is difficult and another draw back is relational databases are not designed to function with data partitioning technique. Data partitioning is a technique to distribute data over multiple disks, site, or tables to increase database manageability. [12] There is added difficulty and complexity when dealing with data in a highly structured databases especially when data does not fit into the table and thus would reduce its performance. Having a highly structured database can lead to large amounts of code in order to make all data fit the database and thus not making it work well with agile development. Now moving onto the competitor, the NoSQL databases and what they have to offer. Businesses have been seen to be trading in their relational databases for a NoSQL database because of the previous limitations mentioned above. Amazon was one of the first big named organization to switch to a NoSQL database when announcing their Dynamo distributed NoSQL system for their internal uses. When it comes to NoSQL databases there are three main types a key value type which focuses on using indexes and can be structure and unstructured, column oriented which uses one extendable column of closely related data and finally document based this type store data as collections in a document, within these collections the data can have any number of fields and have any length examples of this type of document based database are CouchDB by Apache, MongoDB and Basho. Most NoSQL databases are open sourced this is beneficial towards organisations because it would allow to proper testing and experimentation with it at a low cost. Advantages of using an NoSQL database includes faster at processing data compared to relational databases because relational databases usually subjects all data to A.C.I.D this stands for Atomicity which regards updating the data it is either done to all the data or none, Consistency when making a transaction with a relational database there can be no breaking of database rules with the transaction, Isolation meaning the transaction is running independently and finally Durability where the completed transition will persist. Having to perform this on all transactions and data can majorly reduce the performance. A.C.I.D is not usually applied to NoSQL databases as to increase its performance, but a drawback can be when working with an application that needs precision when working with database data. With NoSQL's data model it is much faster than the relational database table-based model and is much more flexible with its data storing.[13]

The choice between a relational SQL database and NoSQL database really does vary on application needs. Properties such as being reliant on data integrity and consistency are reasons to choose a relational database with its implementation of A.C.I.D as mentioned previously ensure protection of sensitive data. Such applications that handle the delicate data of banking and other important data usually steers towards the relation database direction. When objectives are to expand your database to the point that relational databases cannot support without limitations then the direction is pointed towards NoSQL databases. NoSQL databases were designed not as a replacement for the relational database but more as a solution for what they cannot do. NoSQL can handle larger amounts of data compared to relational databases. We can see possibilities of using a combination of both types of database for different aspects for an application such as relational handling more sensitive data and NoSQL handling large scale unstructured data.[14]

3.9 Document Oriented NoSQL Databases

The main choice for a NoSQL database would be the document oriented model. Document oriented model within these databases is a series of collections and within these collections are the documents. These collections have a flexible and no structural pattern compared to the relations database which requires a table and is more structural. They possess superior index searching features which made using them quite effective when retrieving data. When retrieving data, it in JSON format which is an object that contains a series of arrays and other objects inside of it. The document oriented model can have two forms of relationships one being referencing and the other being embedded documents. Firstly the referencing relationship links two documents together one documents name would be in the other for referencing and the embedded documents works with multiple arrays within the same document each with their own unique identification code. When it comes to dealing with these databases there is no designated standard when designing them, they are free from all structure for complete flexibility.[15].

The database design within my project is a Mongo document oriented model database from my research of the advantages and disadvantages of the two types of databases my choice was clear to use the document model because of its characteristics of being flexible and having strong index searching engines.

Chapter 4

System Design

Here I will discuss the overall system design of my application. Proceeding I will be discussing using sequence diagrams created using PlantUML how the database, server and client all interact with one another. PlantUML is a diagram tool that allows the creation of UML diagrams from plain text this proved to be a useful tool when creating my sequence diagrams for my overall system structure and server threads diagram.[16] The process of how the message inputs are converted and distributed and finally the components of the user interfaces in my server and client along with their functionality. My application was constructed using python to create a client server relationship. The relationship is made by the use of sockets using the IP address and port where the server is running on the machine my clients are able to connect to the server.[1] When first running the server, you are shown a graphical user interface that I will explain more in detail in the system design. The client also has a user interface. The functionality of this application is that is it a chat room application by having multiple clients connected to the server they are all able to talk to one another in the chat and have the option to send files over the network. The UI elements were constructed using the python library tkinter.[2]

4.1 Application Process Structure

Figure 4.1 is a sequence diagram to articulate the structure of how the processes are preformed when first dealing the application the separators initialization, connection, repetition and circumstantial are used to show the different stages in the program. First stage is the initialization we need to have the Mongo database running before we run the server otherwise the server user interface will not work correctly. Once the database is up and running its time to run the server. When the server is ran it is bound to the socket and waits for connection following this we fetch the database contents for the user interface to display. Now that both database and server are up and running its time for the client to make a connection using the server's IP and Port. Once there is a successful connection the server sends a message out to the client which is a welcome message. Now we enter out repetition stage where it is simply messages being sent to the server and returns to clients. The client in the diagram can be represented as many clients connected to the server. When coming to the circumstantial stage this only refers to the search function within the server user interface so search specific data or to refresh the display data for the admin user. See Figure 4.1

4.2 Server System Design

The internal structure of the server's processes begins with the first thread which in the diagram on figure 4.2 below is represented as the listener. The processes that begin is the socket binding process where we make our application available through the specific port and address, we set once we have our socket prepared, we then turn to our listening method that listens for incoming connections from clients. We then have an internal loop for accepting future connections. During the circumstance we have a connection a new processing thread is made for that specific client connecting this is how we are able to notify other users which user is sending what message by have a separate thread for each new user then within this new thread it listens for messages and broadcasts to the other connected clients. Using a new thread for each client promotes the use of an asynchronous relationship among the thread system [7] .Our final thread is our GUI thread which is where our tkinter user interface is running by first initialising all its widgets and finally looping over all the widgets to display them. See figure 4.2

4.3 Message Passing Process

Figure 4.3 is a simple diagram of how the message passing occurs from a client. We start off with the user input then once we have our input, we transform it into bytes for sending the message over the socket once the server has received the bytes it then just distributes that message to the other clients where they retrieve the messages from the set of bytes sent out from the server.[1]Learning to understand this process proved very easy once I had completed the research necessary previously mentioned in the technology review.

4.4 Server User Interface

For the server user interface design it has constructed using a single frame with two tabs one for getting the Database data and another for searching the database. The first tab if for listing all contents of the database below you see a list with the values for name, their address and port number along with the date they visited the chat room. The refresh button above the values is connected to a refresh method that updates the list for the admin users in case of new users entering the chat. A comparison of the before and after can be seen in Figure 4.4 of two new pieces of data being add. The next tab is the search tab here we have a search bar, a drop down which give the options of searching by name or by the visit date and a search button which is linked to a search function which uses the pymongo [5] to find the data based on what specific value they are looking for. Pymongo is python module that allows for python to interact with Mongo databases. See Figure 4.4

4.5 Client User Interface

Next, we move onto the client-side user interface. We start off with three different tabs as seen are connect, chat and file share. I will discuss each tab in detail now firstly we have the connect tab this tab is the opening tab which is where you will enter the details necessary to connect to the server. There is also an input for the user's name this is because the first message the server will take in is what the username is. Once connected they will move onto the chat tab to begin using the full functionality of the application. For Connect tab see Figure 4.5 Next, we move onto the client-side user interface. We start off with three different tabs as seen are connect, chat and file share. I will discuss each tab in detail now firstly we have the connect tab this tab is the opening tab which is where you will enter the details necessary to connect to the server. There is also an input for the user's name this is because the first message the server will take in is what the username is. Once connected they will move onto the chat tab to begin using the full functionality of the application. Next, we have the Chat tab for comparison I will show what the chat tab will look like before a connection has been made to the server and after the connection. Before as you can see the message list is empty but once the client has made a connection you will see now that we have a welcome message along with an alert that the user has joined that chat. Following you see that user can message in the chat and their name appears beside it to identify the user sending the message. See figure 4.6 Finally, we are onto the file share tab here is where the file sharing process begins, we start off by using the browse files button this then brings up a file explorer for the user to select the text file they wish to send to other users in the chat room. Once selected as seen the path of the file is displayed for the user for verification and then pressing send will send it to all users in the chat room. The sender does not receive the file in their directory only the other users receive the file. Following can be seen of a before and after the file sharing process the original name, file type and content is receive successfully. See Figure 4.7

4.6 Database Model

The chosen model for my NoSQL database was the document oriented model.[15]. My choice in this model became clear when researching this model as it possesses clear abstract structure allowing for full flexibility. The form of relationship I chose was an embedded relationship having multiple documents within the one document. This style made retrieving the data from the database easier for my server as all information are within the same document. My schema for my database is an object for name and date for when the client visited and an array object containing the clients port and IP addresses.The python package pymongo [5] provides the setting up of the database if it does not exist and the methods provided handle the inserting and finding of data.

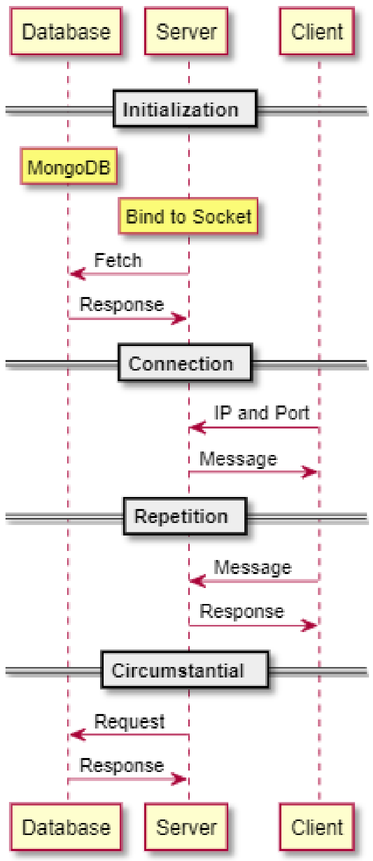


Figure 4.1: System Sequence Diagram

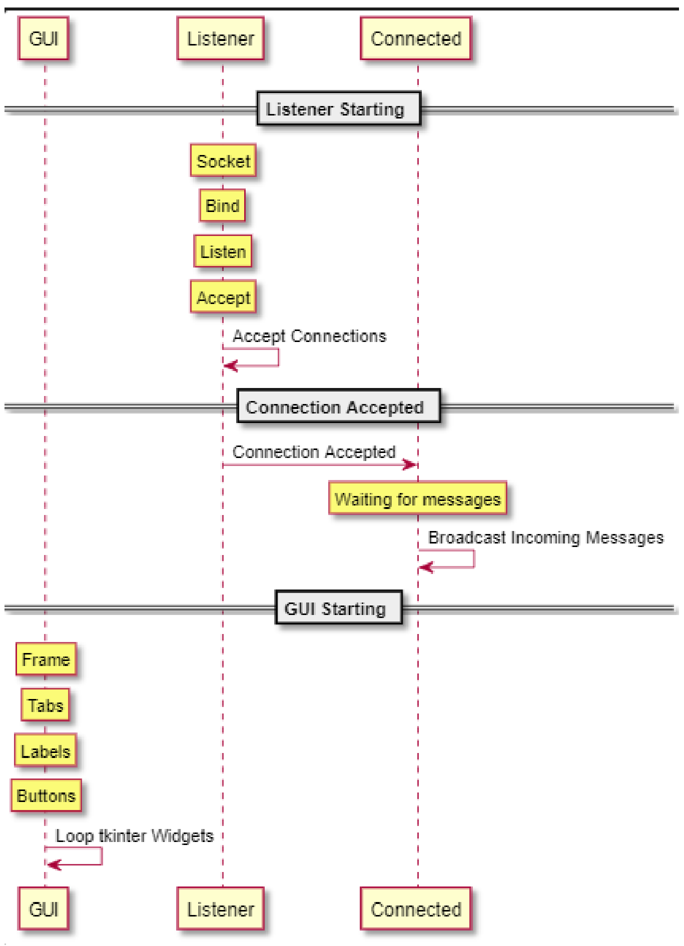


Figure 4.2: Server Sequence Diagram

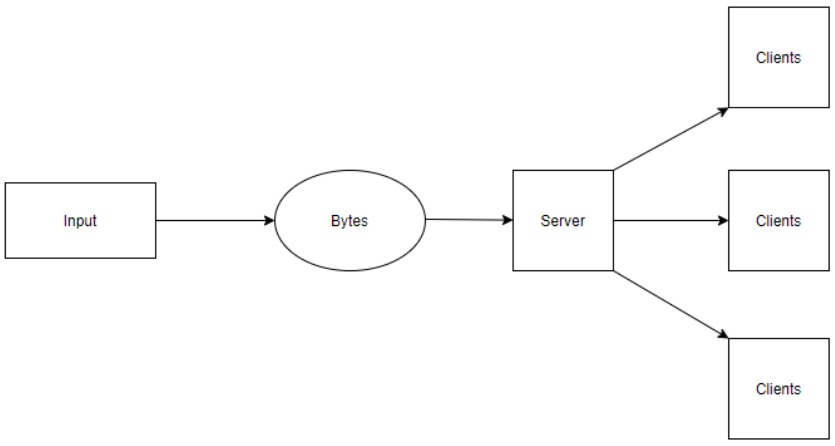
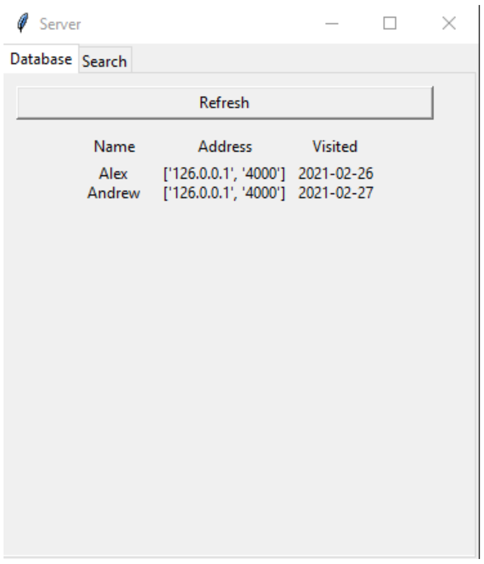
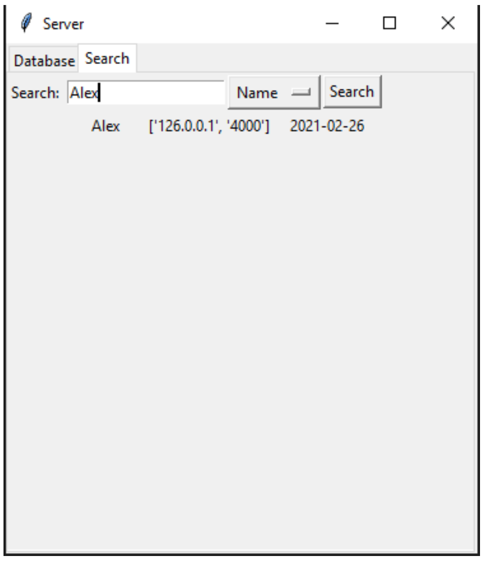


Figure 4.3: Converting input into Bytes



```
Name      Address               Visited
Alex      ['126.0.0.1', '4000']  2021-02-26
Andrew    ['126.0.0.1', '4000']  2021-02-27
```

(a) Main Server Tab

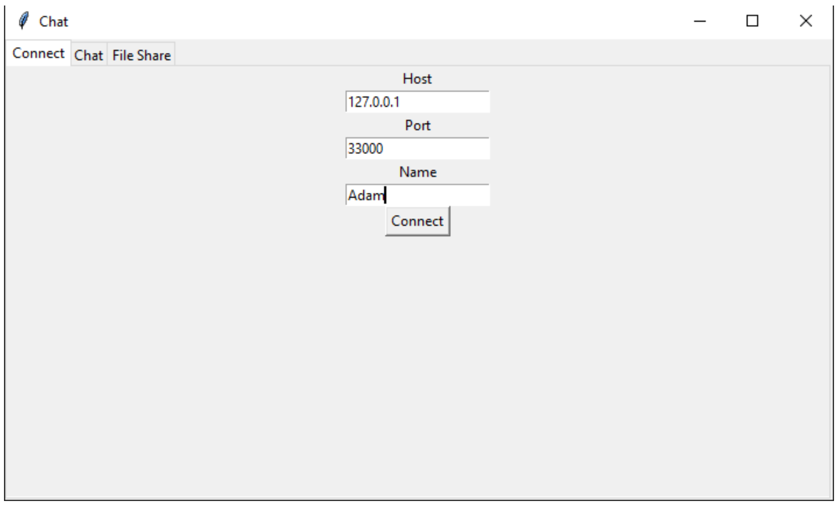


Search: Alex

```
Alex   ['126.0.0.1', '4000']   2021-02-26
```

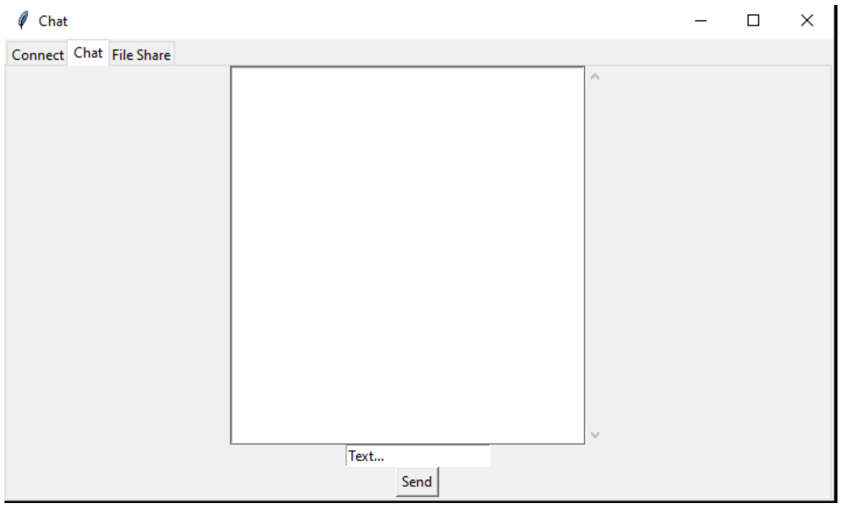
(b) Search Tab

Figure 4.4: Server UI

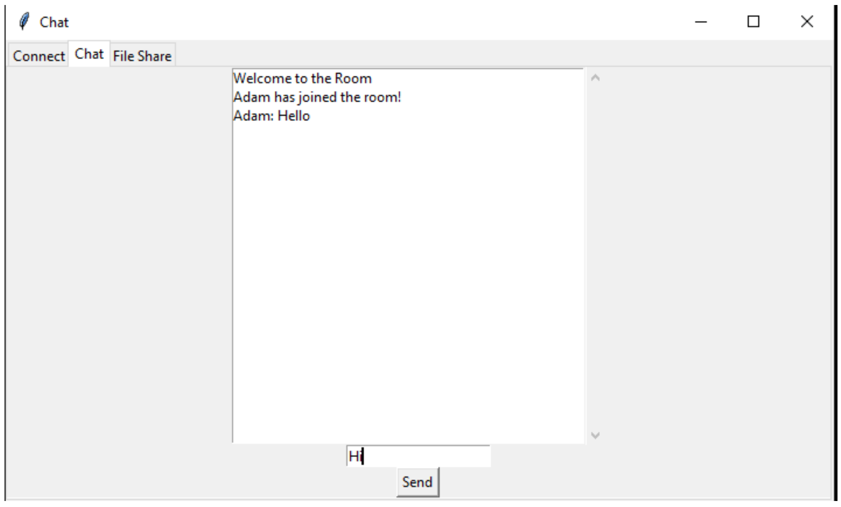


Host
127.0.0.1
Port
33000
Name
Adam

Figure 4.5: Connect Tab



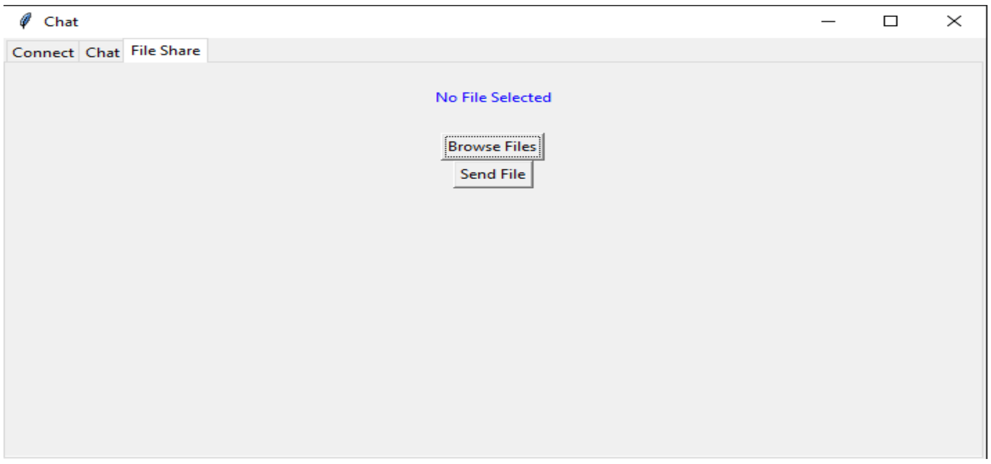
(a) Before Connection Chat Tab



Welcome to the Room
Adam has joined the room!
Adam: Hello

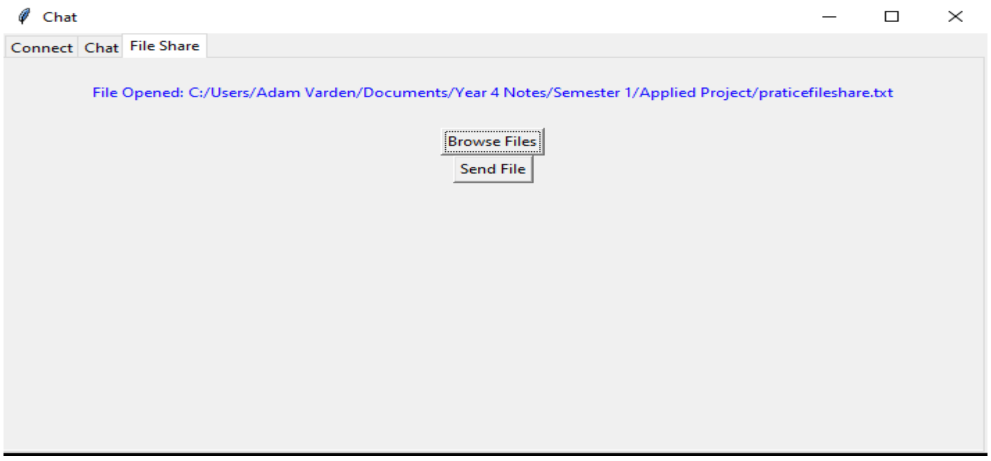
(b) After Connection Chat Tab

Figure 4.6: Client UI



No File Selected

(a) Before File



File Opened: C:/Users/Adam Varden/Documents/Year 4 Notes/Semester 1/Applied Project/praticefileshare.txt

(b) After Selecting File

Figure 4.7: File Share Tab

Chapter 5

System Evaluation

When beginning my system evaluation, I began by doing a series of acceptance tests once I believe my application was ready to be used by others. I first began by comparing the objectives of my application against the final application. I will break down my application into separate main components such as client and server and within these sections will be further subsections on the specific attributes I aimed to acceptance test against my objectives when beginning this project.

5.1 Server Tests

When first running the server after doing the necessary preparation which is simply having the Mongo database running. I then proceeded to do my acceptance tests.

5.1.1 Socket Opens for connections

I needed to make sure that the server allows for connections from clients to test this I had a client enter the servers IP address and port and tried connecting to it we know that the client was able to connect to the server by looking at the servers print statements, and it should print an IP address and port of the client who connected this means that connection was successful.

5.1.2 Server GUI

When first running the server the graphical user interface using tkinter should appear without any manual intervention provided the prerequisite steps had been done before hand. The server GUI was able to load without any manual intervention and worked successfully with displaying the necessary widgets coded for the GUI.

5.1.3 Server with Mongo Database

Continuing from the previous section when the GUI has loaded successfully you should expect to see a list of the database contents containing names, addresses and visited dates. In order to test my refresh button, I added a set of values into the Mongo database manually and clicked refresh in order to test that my refresh method linked to my refresh button was working correctly this also worked successfully. We then move on to the search tab here we input a name or a date we wish to search the database for by selecting which attribute using the drop-down menu. I test this using a random name from the previous tab that I knew would appear in the search and then a name that was not and both did their intended purposes proving to be successful. The name present appeared in the search and the name not present did not retrieve anything.

5.2 Client Tests

When first running the client, we do not need any prerequisites like the server needing the Mongo database all that is needed is the server running when trying to make a connection from the client.

5.2.1 Client GUI

I began by simply running the client application and the GUI loads up instantly for the user. The three tabs loads successfully with their required widgets in them. I moved to the file share tab first where I tested the browse files button to make sure that the file explorer appears for the user and when a text file is selected that files path is displayed at the top of the tab. Once making sure the tabs had their required widgets, I wrote off the client-side GUI as successful.

5.2.2 Client to Server Connectivity

When testing that my client can successfully connect to the server, I entered in the IP address and Port of the server and the name I wish to be displayed as and clicked the connect button. We then move to the chat tab and there should be a welcome message for the client with their name and saying that they joined the chat room. This all worked successfully. When testing that another client can join the chat room and the other clients already joined will be notified, I simply started up another client and repeated the previous steps above, but this time referred to the first client's chat tab within it there should be a message notification saying someone else has joined. Then for further tests I messaged using both clients to make sure the correct name appears beside their message and that proved to be successful in my testing. Then I moved onto my final feature to test which was the most difficult when developing which is the file share feature. We start off by using the browse button to find the text file we wish to send in the file share tab. Then simply pressing the send button will send that file over the socket to make sure it goes over the socket correctly, print statements were put into the client side and server side for verification and then looking into the other client's directory we see the file that was sent. This was all successful in my final tests.

5.3 Comparison against my Objectives

When setting out my objectives I was clear and simple with what I wanted to develop a chat room app which I managed to do using the technologies I chose which were python, socket programming and Mongo databases. My objectives in regards to system functionality were to create a server client relationship using socket programming, have the server listening on a socket for incoming connections from multiple clients and have messages be easily distributed among those clients, have a user interface keep messages updated among all users, allow for users to be able to effectively send files over the sockets to one another, have the server record the users that connect to the chat room in a database , have a server side graphical user interface, using the server side user interface can get the contents of the database and allow for the server admin to search the database for a specific record. When developing this project I wished I could of added for background autonomous checks over the network such as checking clients if they are still active and maybe to disconnect them due to inactivity, other functionality I had wished I could of implemented would have been the ability to display images and incorporate some form of web API for added features however finding one that would suit the theme of a chat room was difficult to find but adding these features would of made the scope of my project much bigger for one person to handle and working with sockets proved to be tricky in some instance when it came to the transferring of data over them. The socket byte stream can in some instances cause the cascading of message or a form of backlog when transmitting cause issues when receiving the messages When researching the SSL protocol it enlightened me that in industry security of a user's data is pivotal in development of a service that handles clients addresses and messages.

Chapter 6

Conclusion

When beginning this project, I began with the context of creating a chat room application using the technologies Python,a Mongo database and socket programming to develop this chat room application. I set out my objectives for this project in two contexts one being an extension of knowledge into the programming language python, learning more about networks at a practical level and the use of NoSQL databases the other context of objectives in setting out a vision for my project and that was to create a server client relationship with the use of sockets, develop both client side and server side graphical user interfaces with the use of the python package tkinter, having my server possess the functionality to distribute messages over multiple clients, have my server record users and search users with the use of a NoSQL database MongoDB as my database and offer the feature of sending files through the socket to the other connected clients. From my system evaluation what successfully worked for me were:

- Setting up a database for multiple clients to connect to.
- Client successfully connecting to the server.
- Having clients be able to send files over sockets.
- Have clients be able to send messages to one another.
- Database integration with the server.
- Graphical User Interface implemented in both client side and server side.

Regarding the metric of success and failure set out in my Introduction I did accomplish the objectives that I set out to do which would make this a success but I feel that there is more potential for expansion on this chat room application.

For future expansion it would be for the server to be running on a virtual machine and have multiple users from different places attempt to connect to it currently my application runs on the local IP address and ports. Incorporating better socket maintenance would have been an area that would of benefited this application as it may not know entirely if a client was disconnected not through the designed way of exiting through the graphical user interface such situations would include a computer crash having the server be able to send out a sort of heartbeat for active connected clients could of be a nice feature for the admin services.

I learned quite a lot from this project such as learning how to work on my own. Working on my own forced me to make better plans and try to manage my time better. When making plans I felt that I was better at evaluating my goals and objectives in regards to what goals are achievable in a given space of time and due to having only one person working on the application. I learned when running into errors that there has to be more than one way of doing the same thing this I learned greatly when dealing with the file sharing as it was the most difficult part of the project. When dealing with issues I learned its better to put a hold on that particular component and return to it later with a fresh eyes and this worked for me when dealing with issues. I quite enjoyed the development process of this project as it got me to try and document and manage my time better in the development of an application nearly similarly of how it would be done in industry. Areas that intrigued me were finding new methods within the python language that could be of use when partaking in another application development. Getting to expand my knowledge was fun as over time my code was looking neater and easier to read which is a fundamental of the python language and becoming second nature to identifying change in indentation. The project really pushed me to try my hardest even with the situation that this year has been and it is something I am proud of.

Chapter 7

Appendices

7.1 GitHub Link

https://github.com/AdamVarden/AppliedProject

7.2 Instructions

- Begin by downloading MongoDB Community edition from this link: https://www.mongodb.com/try/download/community
- Once downloaded navigate to the file directory of the bin folder in the mongo folder found in the Program Files folder of the C: drive
- Then run the following command: mongod
- For convenience it is easier to download the anaconda prompt that is provided by anaconda. See link here: https://docs.anaconda.com/anaconda/install/
- Once installed run these following commands: pip install sockets and pip install pymongo
- Further steps provided on GitHub README

Bibliography

[1] L. Kalita, “Socket programming,” *International Journal of Computer Science and Information Technologies*, vol. 5, no. 3, pp. 4802–4807, 2014.

[2] J. E. Grayson, *Python and Tkinter programming.* Manning Publications Co. Greenwich, 2000.

[3] Jennings, Nathan, “Socket programming in python (guide).” `https://realpython.com/python-sockets/`,.

[4] Adam Varden, “Github for applied project„” 2021. `https://github.com/AdamVarden/AppliedProject`,.

[5] “Pymongo link.” `https://docs.mongodb.com/drivers/pymongo/`,.

[6] M. A. Alnatheer, “Secure socket layer (ssl) impact on web server performance,” *Journal of Advances in Computer Networks*, vol. 2, no. 3, pp. 211–217, 2014.

[7] N. Matloff and F. Hsu, “Tutorial on threads programming with python,” *University of California*, 2007.

[8] G. Van Rossum *et al.*, “Python programming language.,” in *USENIX annual technical conference*, vol. 41, p. 36, 2007.

[9] “Mixins link.” `https://en.wikipedia.org/wiki/Mixin`,.

[10] K. Srinath, “Python–the fastest growing programming language,” *International Research Journal of Engineering and Technology*, vol. 4, no. 12, pp. 354–357, 2017.

[11] S. Khoirom, M. Sonia, B. Laikhuram, J. Laishram, and T. D. Singh, “Comparative analysis of python and java for beginners,” *Int. Res. J. Eng. Technol*, vol. 7, pp. 4384–4407, 2020.

[12] "Data partitioning link." https://link.springer.com/referenceworkentry/10.1007%2F978-0-387-39940-9_688#:~:text=Data%20Partitioning%20is%20the%20technique,in%20one%20of%20two%20ways.,.

[13] N. Leavitt, "Will nosql databases live up to their promise?," *Computer*, vol. 43, no. 2, pp. 12–14, 2010.

[14] K. Sahatqija, J. Ajdari, X. Zenuni, B. Raufi, and F. Ismaili, "Comparison between relational and nosql databases," in *2018 41st international convention on information and communication technology, electronics and microelectronics (MIPRO)*, pp. 0216–0221, IEEE, 2018.

[15] H. Vera, W. Boaventura, M. Holanda, V. Guimaraes, and F. Hondo, "Data modeling for nosql document-oriented databases," in *CEUR Workshop Proceedings*, vol. 1478, pp. 129–135, 2015.

[16] "Plantuml link." https://plantuml.com/,.