# Client-Server Secure Messaging Platform

Computer Science A Level Non-Exam Assessment

Sam Poirier

18 October 2018

Contents

[1 Analysis 2](#_Toc527637172)

[1.1 Analysis Plan 2](#_Toc527637173)

[1.1.1 Interview 2](#_Toc527637174)

[1.1.2 Questionnaire 2](#_Toc527637175)

[1.1.3 Project Background 2](#_Toc527637176)

[1.1.4 Objectives 2](#_Toc527637177)

[1.1.5 Solutions 2](#_Toc527637178)

[1.2 Interview 3](#_Toc527637179)

[Questions 3](#_Toc527637180)

[Responses 3](#_Toc527637181)

[Analysis 3](#_Toc527637182)

[1.3 Questionnaire 4](#_Toc527637183)

[Questions/Responses 4](#_Toc527637184)

[Analysis 4](#_Toc527637185)

[1.4 Project Background 7](#_Toc527637186)

[1.4.1 Specific Client Needs 7](#_Toc527637187)

[1.4.2 Existing Solutions 8](#_Toc527637188)

[1.5 System Objectives 12](#_Toc527637189)

[1.6 Solutions 13](#_Toc527637190)

[1.6.1 Framework 13](#_Toc527637191)

[1.6.2 GUI Framework 13](#_Toc527637192)

[1.6.3 Integrated Development Environments 14](#_Toc527637193)

[1.6.4 Database Formats 14](#_Toc527637194)

[1.6.5 Solution Decisions 14](#_Toc527637195)

# Analysis

## Analysis Plan

### Interview

It is important to interview the client, Mr Ovayolu; a secondary school teacher, in order to establish the key features that they require for the program to include, as well as what features are less important not needed. I will work from the responses the client gives in order to accurately tailor the software to their requirements.

### Questionnaire

I will develop a questionnaire to determine the demand from the general public in order to tailor the software to be desirable to users other than the initial client, as long as the general demand does not contradict the client’s needs. I will use google forms to create the questionnaire.

### Project Background

Within the Project Background I will detail the specific Client needs, in order to fully flesh out the purpose of the software. I will also analyse a number of existing pieces of software that fulfil similar solutions to determine crucial aspects that should be included in my project, but also determine flaws which I should make sure not to include in the software

### Objectives

Under the objectives section, I will detail the exact objectives the project must meet, along with the specific criteria of each objective. This will allow me to ensure that the final product meets with all the client expectations.

### Solutions

Here I will detail the technical aspects I plan to use in order to develop the software, including programming languages, Integrated Development Environments, general database rules (if applicable) and other technical Solutions.

## Interview

|  |  |  |
| --- | --- | --- |
| Questions | Responses | Analysis |
| What features of an instant messaging platform are most important for you? | • Fast and secure communication between students by 1:1 or group chat  • Easy to understand, clean but functional user interface  • Be able to run in all Windows versions  • View chat history  • Group chat and direct messaging  • Reply to a particular message, both in a group chat and in a private chat.  • Forward the messages to a particular group chat or an individual.  • Get alert when new messages arrived.  • View list of connected users in the school.  • Report button for inappropriate, bullying, threatening messages… | The software is being designed specifically for this user. It is important that as many of these are implemented as possible, and ideally all of them are. |
| How often will the users use the platform? | Break and lunch time | The software will mostly be used during the day, so there will be time available for potential server maintenance; there isn’t a need for 100% uptime. |
| How many users should the platform support? | Concurrent around 100, total around 2000 | The program must support a large quantity of users – some stress/load testing will be needed to ensure the load can be managed successfully. |
| How much message history should users be able to see? | 3 Months or 100 messages. Ideally, user should be able to set this in settings. | The client wants a configurable number of messages to be readable, meaning the server should likely store all messages then send the client the amount they have requested. |

## Questionnaire

|  |  |
| --- | --- |
| Questions/Responses | Analysis |
|  | Everyone who answered the survey has used an IM application at some point, which means that the feedback received is not necessarily representative of all the users, so should not be treated as an absolute representation of the userbase. |
|  | WhatsApp is the most widely used IM application, followed closely by Discord and Instagram. The least widely used applications are IRC and Slack. All of these extremes should be investigated in more detail in order to replicate their success and avoid their mistakes. |
|  | Interestingly enough, although WhatsApp was the most widely used, Discord was in fact the one that was used the most overall, and snapchat which was thoroughly in the middle of the widely used, was one of the only three chosen as most used. This shows that the most widely used applications are not necessarily the best ones. |
|  | Windows and Android Phone are by far the most widely used, and so for the best coverage of users, are the platforms that would be the best to target. |
|  | Again, Windows is by far the most used platform and is therefore the clear frontrunner in terms of OS to develop for. |
|  | 73.4% of those surveyed indicated that Data Security is important for them. This is clearly an important feature. |
|  | 93.3% of users indicated that secure logins where an important feature for them so this also is a crucial aspect of the project. |
|  | Again, most users agree that having a viewable message history is an important feature of an IM application. |
|  | Only just over half of the responses indicated that they cared about transparency regarding their personal data and what was done with it. |
|  | Most people agreed that an easy to understand/use user interface was a good feature. It could be worth developing a few different layouts to receive feedback about. |
|  | A large number of those surveyed indicated that a customisable UI would be wanted. More detail might be needed into what exact customisations are wanted. |
|  | Most users agreed that Direct Messaging would be an important feature. It was also requested by the Client so is a high priority. |
|  | Group chats were also indicated a feature wanted by the client as well as the survey candidates. This means that it should definitely be implemented in the product. |
|  | A fairly mixed response from the survey, but this was also requested by the client so should be included regardless. |
|  | Some of these feature requests are more in the scope of this project than others. These were individually requested and so are low priority. |

## Project Background

Instant Messaging platforms have existed pretty much as long as computers have, originally (in the 1960s) acting as a way of communicating between users on multi-user operating systems. They were also originally used for notification systems like printing. A good example of possibly the first instance of this software is from CTSS (Compatible Time-Sharing System), an operating system developed by MIT which was the first to demonstrate any sort of messaging, and was a precursor to email.

Hashing has also existed as a concept for a long time, with the term originating from its non-technical meaning (to "chop" or "make a mess" out of something), and was first coined in the 1960s. For this project, we will be specifically looking at Cryptographic Hashing which usually consists of a mathematical algorithm that maps data of arbitrary size to a bit string (a hash) and is designed to be a one-way function. The purpose of this function is to obscure a user’s personal data so that it is secure.

### Specific Client Needs

I will be creating a secure Client-Server messaging platform. My client is Mr Ovayolu, a Computer Science teacher who wishes to set up a secure platform for the teachers and students to discuss work with each other. The focus will be on a secure design, hosted by the client to ensure they know exactly where their messages/data is stored and that it is not being sold to companies by the owners of the software. This is important as a large amount of the data on the system will belong to minors, so the school has extra responsibility to protect them. The end users will be teachers or students who may not be very ‘tech-literate’; the application must be easy to set up and use.

### Existing Solutions

#### Internet Relay Chat

The first IM platform to run over the internet was Internet Relay Chat (IRC), which was made in 1988. By 1989 IRC had spread across the world with over 40 servers worldwide.

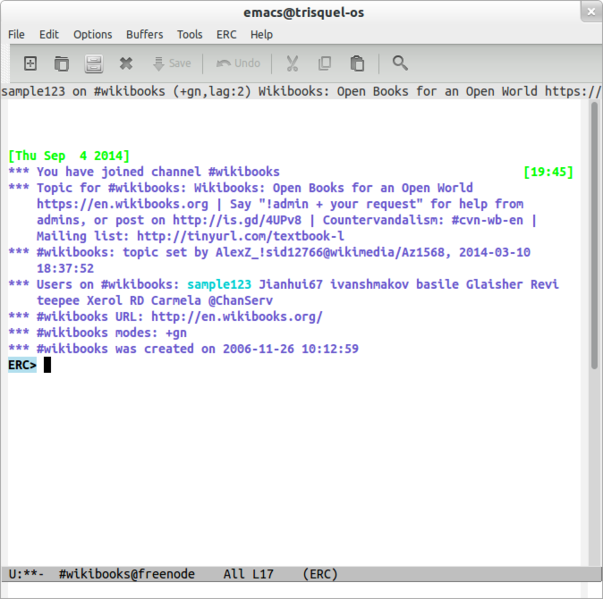


Figure 1.1: An IRC Client called ‘ERC’ which runs within the text editor Emacs.

|  |  |
| --- | --- |
| Pros | Cons |
| Client-server model – the client’s GUI or functionality can be upgraded without touching the server allowing for all sorts of clients to be compatible with each other. | Complicated to set up servers correctly and securely. |
| Can be secure if set up correctly. | Some clients are complicated to use and set up. |
|  | Little support for multimedia formats, reporting messages, file transfer etc. |

#### Discord

Released in May 2015, Discord is a multiplatform Instant Messaging and voice-over-internet Protocol (VoIP) application. It runs on pretty much every platform available and is free to use.

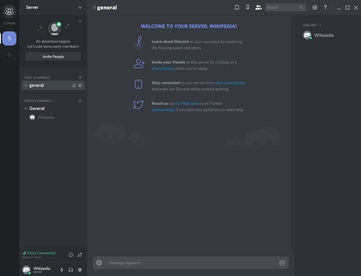


Figure 1.2: The Discord desktop application after creating a new server.

|  |  |
| --- | --- |
| Pros | Cons |
| Extremely easy to create new accounts and servers – all implementation is abstracted away. | No personally hosted servers – all data is controlled and stored by Discord Inc. meaning that a user is not in control of it’s data’s security or status. |
| Servers are secure in the sense that only invited users can join. | Written using Electron – a framework with a famously high memory footprint on the PCs running it, so may not work well on older machines. |
| Lots of support of extra features, for example VOIP, image embeds, direct messages, file transfer. |  |

#### WhatsApp

WhatsApp is a free Instant Messaging, Voice and Video Chat client. It was founded in 2009 and is owned by Facebook. It is supported on many platforms including most smart phones.

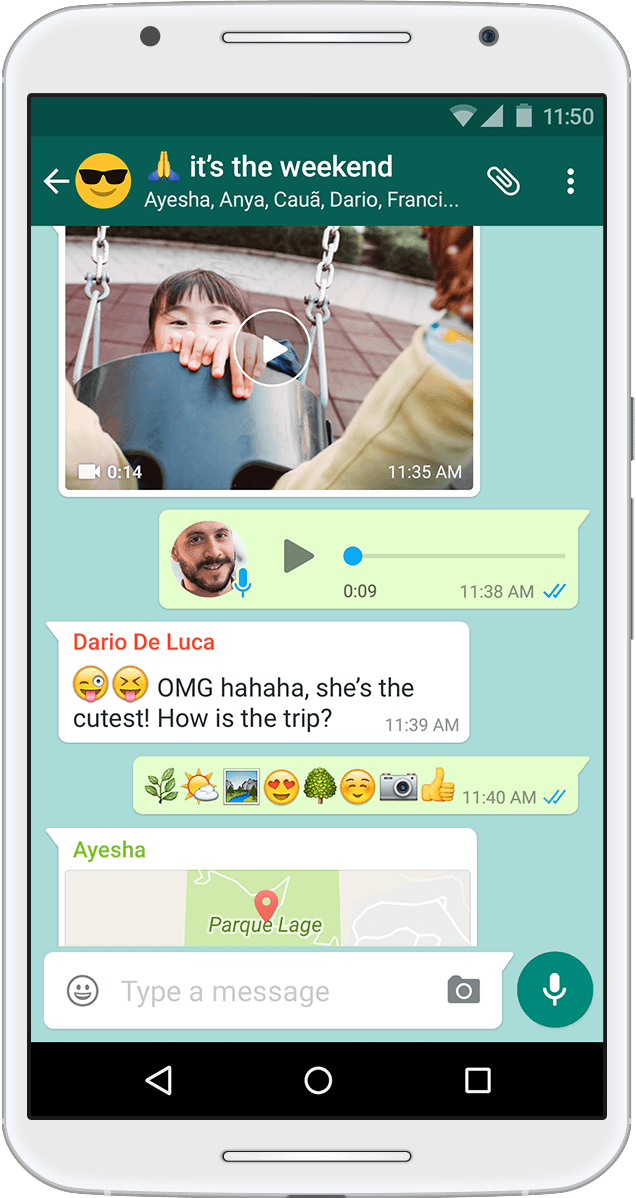


Figure 1.3: WhatsApp running on an Android Device

|  |  |
| --- | --- |
| Pros | Cons |
| Easy to use and set up. | Very little privacy or control over data – it is owned by Facebook, a company which is notorious for abuse of its users’ personal data. |
| Easy to connect to new ‘servers’ or group chats. | User interface is not designed for large groups of people or high rate of messages being sent. |
| Lots of features including voice chat, file sharing, direct messages etc. |  |

#### MD5 Hashing Algorithm

MD5 is one in a series of algorithms designed by Professor Ronald Rivest of MIT in 1991. It was designed as a cryptographic hashing function for use in encryption.

|  |  |
| --- | --- |
| Pros | Cons |
| Implementations exist already in many different programming languages already, in addition to pseudocode for the algorithm being widely available. | Suffers from extensive vulnerabilities, including a high chance of collisions (when two or more different input string produce the same output hash string) rendering it obsolete for encryption usage. |

#### 1.4.2.5 SHA-2 Hashing Algorithm

SHA-2 (Secure Hash Algorithm 2) is a set of cryptographic hash functions designed by the United States National Security Agency (NSA) in 2001. It utilises a Merkle–Damgård structure.

|  |  |
| --- | --- |
| Pros | Cons |
| When implemented correctly is virtually uncrackable. | Low support on older systems, for example Windows XP. This conflicts with the Client’s requests of running on all OSes |

## System Objectives

|  |  |  |
| --- | --- | --- |
| No | Objective | Performance Criteria |
| 1 | Login Screen User Interface must contain:   * Input for username and password * Login Button * Register Button * Error Label to display login issues | All of these UI elements must exist in an organised and easy to read layout. |
| 2 | Register User Screen UI must contain:   * Input for username, password and password confirmation * Register (submit) button * Error label to display register issues | All of these UI elements must exist in an organised and easy to read layout. |
| 3 | Main Window UI must contain:   * Box to contain message history * Input box for sending messages * Button to send message * General server information; who’s connected, who you’re logged in as etc | All of these UI elements must exist in an organised and easy to read layout. |
| 4 | Connecting to the server should be secure. | The initial handshake between client and server should be secure; ie no ‘fake’ clients attempting to connect should be accepted. |
| 5 | Client/Server Communication should be secure. | All packets to and from the server should be secure so that anyone intercepting them cannot read the information. |
| 6 | All messages and personal data should be stored securely. | The database should be encrypted; the whole thing in a two-way encryption unique to the server, and the passwords in a one-way encryption too. |
| 7 | User interface should be easy to use and read. | The UI should be intuitive to read, use and understand. |
| 8 | Client and server should be easy to set up and run. | The client and server should require minimal setup to get into full working/running order. |
| 9 | The server must be able to handle lots of users both consecutively and in general. | The server must be able to support 100 concurrently connected users and 2000 total registered users. |
| 10 | There must be a way for users to configure settings for the client. | The clients should be configurable via a settings menu which is saved between sessions. |
| 11 | The application must support most requested features by the client. | As many features requested by the client as possible should be implemented into the program. |

## Solutions

### Framework

|  |  |  |
| --- | --- | --- |
| Language | Pros | Cons |
| Python | Lightweight  Simple Syntax  Can be developed on school computers  Multiplatform  Decent documentation | Interpreted therefore slow and not memory efficient |
| C# | Powerful  Personally preferred language  Lots of good documentation  Very easy to debug | Low multiplatform support  Cannot be developed on school machines |
| JavaScript (via Electron) | Lightweight  Forgiving syntax  Multiplatform | Extremely loosely typed – easy to make bugs  Hard to debug |

### GUI Framework

|  |  |  |  |
| --- | --- | --- | --- |
| Language | Framework | Pros | Cons |
| Python | tKinter | Shipped stock with python, no extra installation | Extremely limited in terms of functionality  Hard to use |
| PyQt | Support for advanced GUI configuration  Easy to use | C++ Backend, some errors are hard to catch and result in silent crashes |
| C# | Winforms | Easy to use  Stock with C# | Lots of auto-generated code or lots of manual setup |
| JavaScript | Electron (with node.js, jQuery and npm) | Frontend developed in HTML/CSS so the GUI is very customisable. | Very memory inefficient  Uses lots of other people’s modules/code |

### Integrated Development Environments

|  |  |  |
| --- | --- | --- |
| IDE | Pros | Cons |
| Visual Studio | Advanced Debugging tools  Good at identifying programming errors before compiling/running  Easy to use | ‘Heavy’ piece of software – lots of overhead |
| Visual Studio Code | Advanced Debugging tools  Good at identifying programming errors before compiling/running  Easy to use  Lightweight | Less features than Visual Studio |
| IDLE | Comes stock with most python installations  Very lightweight  Simple Debugging tools | Feature lacking; little auto completion or pre-run bug finding |
| Notepad++ | Extremely lightweight | Very few features  Cannot compile many languages |

### Database Formats

### 

|  |  |  |
| --- | --- | --- |
| Format | Pros | Cons |
| SQL | Very powerful  Lots of supported datatypes  Can create databases with complex structures | Complicated to set up  ‘Heavy’ – not very lightweight at all |
| SQLite | Lightweight  Reasonable amount of datatypes  Support for reasonably complicated structures | Less features than SQLite |
| MongoDB | Unique Solution; structure is based upon a tree rather than a table | Feature lacking; little auto completion or pre-run bug finding |

### Solution Decisions

I plan to write my program in Python, as it runs on multiple platforms easily and can be debugged without too much trouble. I will use IDLE to develop the project as it has all the features I need, and is nice and lightweight without too much bloat. I will use PyQt for my GUI as tKinter is not advanced or powerful enough to meet the client needs. I will use SQLite for my database as it will likely be quite simple; SQLite is a good compromise between features and functionality.