NEA Project

By Sarwar Rashid

Introduction

Liane Silva is a foreign exchange student who has recently moved to the city of Manchester to study at its respective university. She has moved from her hometown Iloilo, Philippines and would like to maintain contact with her relatives.

Specifically, she only wants to exchange messages back and forth between her relatives to update them on her current status to which they can do the same without having to use an application or physical means of contacting overseas as there is the risk of complication. This complication can be due to her relative's level of ICT skills as they are not be frequent users of technology or have a good understanding of it.

Therefore, she has suggested to her relatives to maintain contact via a chatroom that can be accessed by a web browser where they can all gather and exchange messages in real time with identifiable accounts. It will be by browser as this will help to eliminate some complexity for the relatives when attempting to access the method of contact as they will simply have to load the website and assign themselves a name for them to be identified by the end-users.

As a result, I have devised a chatroom orientated website that will be easy to access and navigate for the user, given their level of IT expertise. From there, they can access the chatroom so long as they have provided information to identify them for them to be considered a 'user'. These users can then exchange text messages to each other by simply inputting the message they desire and sending it to the receiving end who will be able to view the message in real time, keeping the chatroom dynamic. The system will have a simple interface as this will allow both sides to immediately grasp the function of the site and to be able to use it with ease. As the website is intended for a small group of users, the chatroom will lack common features such as a friends list as it would be unnecessary **(See pg. 5 1.2.2. Figures 2 - 3).** Despite this, I would ideally have other functions implemented into my chatroom to enhance the user experience.

End User

I have identified an end user, Ruiz Nagagor, who will assist me by questioning the purpose and features of the website alongside ensuring that the website works by testing various features that I plan on implementing such as whether a valid account can be created and logged in and if messages sent from one end can be seen by the receiving end within a reasonable time.

Index

1. Analysis

1.1. Investigation

To understand how to articulate my chatroom, I will have to utilise different investigation methods that delve into the structure and features of a chatroom. The investigation will therefore help me to create and develop my chatroom because it will highlight the requirements and possible implementations I could include.

And so, my investigation methods will include a questionnaire, research of different chatting platforms and an interview with my end user. I found that a questionnaire would be extremely beneficial as this will aid me in getting a diverse range of answers from various users who have experienced at least one popular chatting platform, helping me to accordingly identify more features that I could include and assistance in understanding what my chatroom must require. Research has also been chosen for a similar reason as by identifying and examining an existing chatting platform, I will be able to structure and include features in my chatroom that will be essential. As for an interview, it helps to include my end-user asking me specific questions about the chatroom so that I myself can identify its necessities and the process of going through it from a user's perspective.

1.2. Questionnaire

For the first process of my research, I conducted a questionnaire on the application Google Forms as this will allow me to gain an understanding of what users know, enjoy and would like to see in my chatroom website. Below displays the statistics I have obtained and what each statistic will mean for my chatroom.

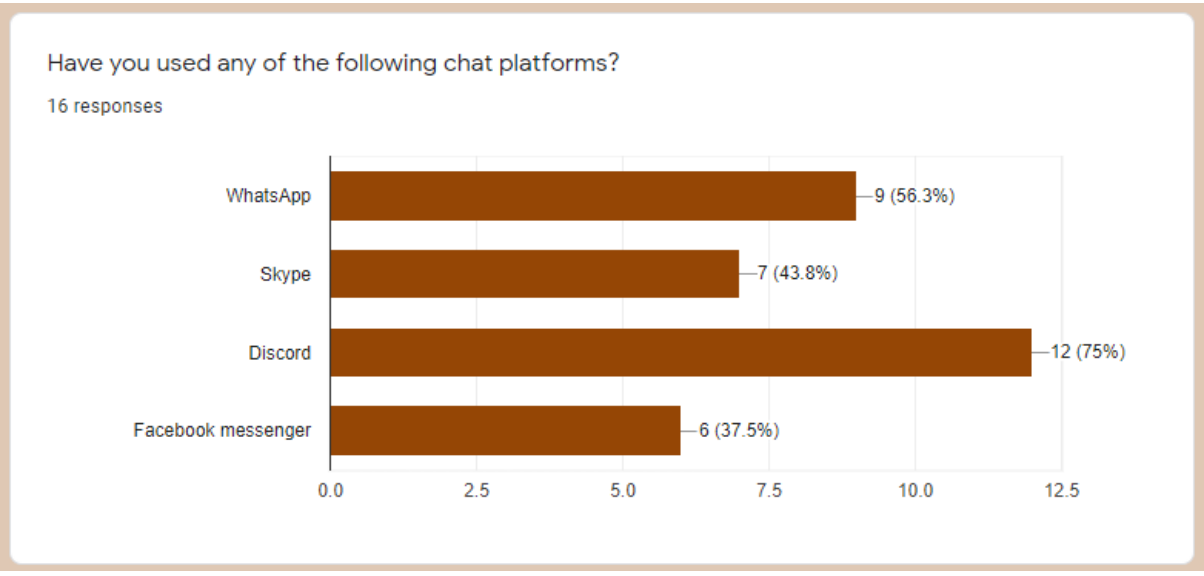


Figure 1 - Popular chatting applications that people have used before

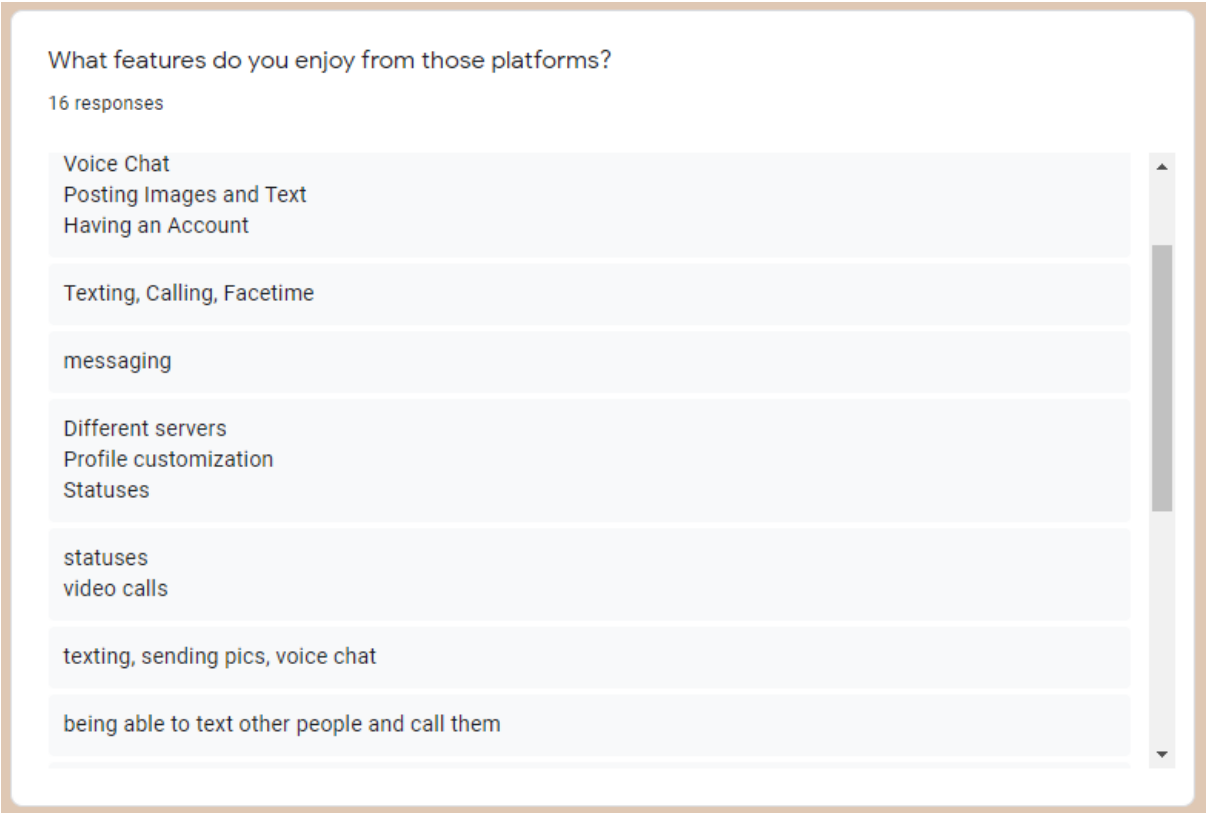


Figure 2 - Features that the person enjoy from these applications

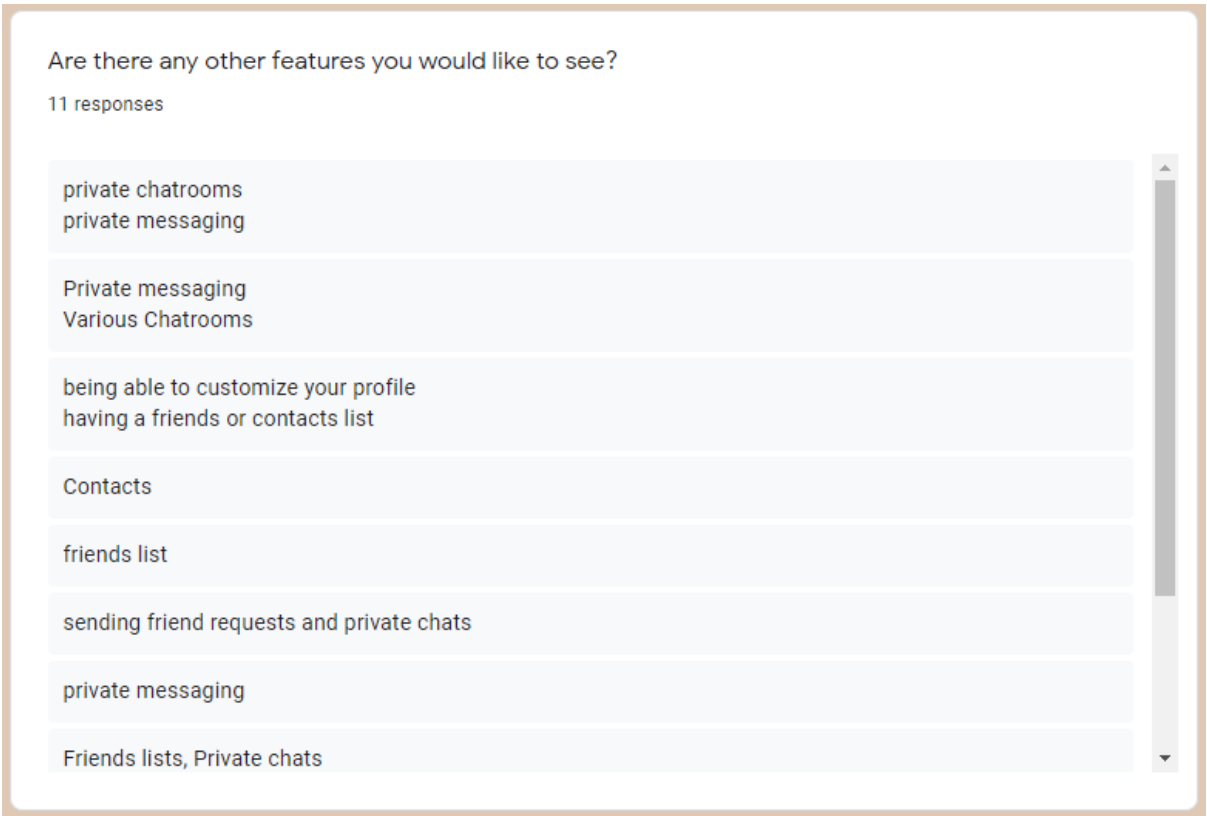


Figure 3 - Other features that they may like

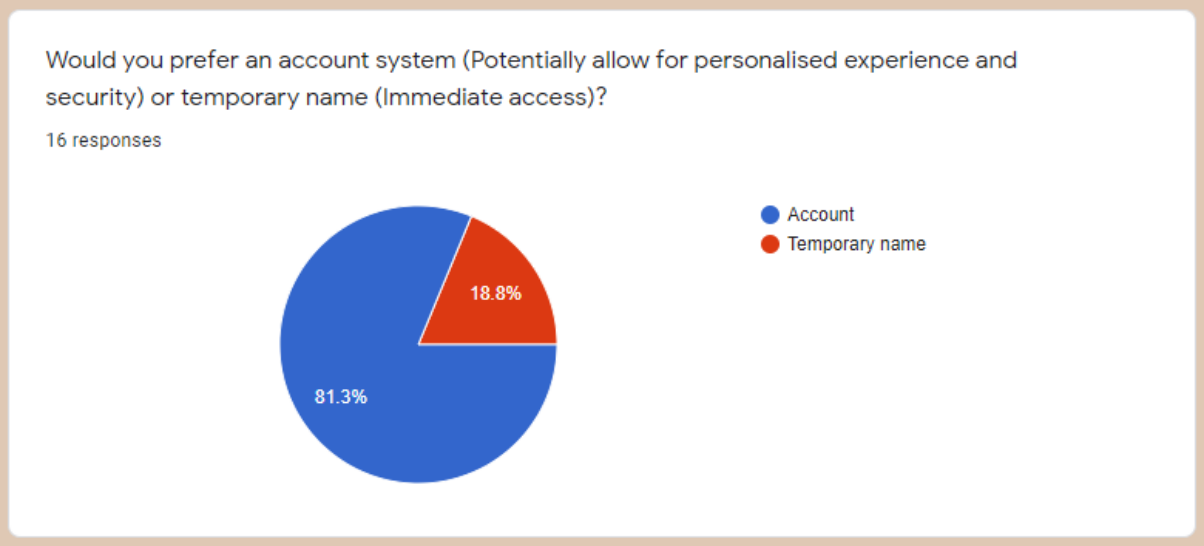


Figure 4 - Whether they would like an account system or temporary name

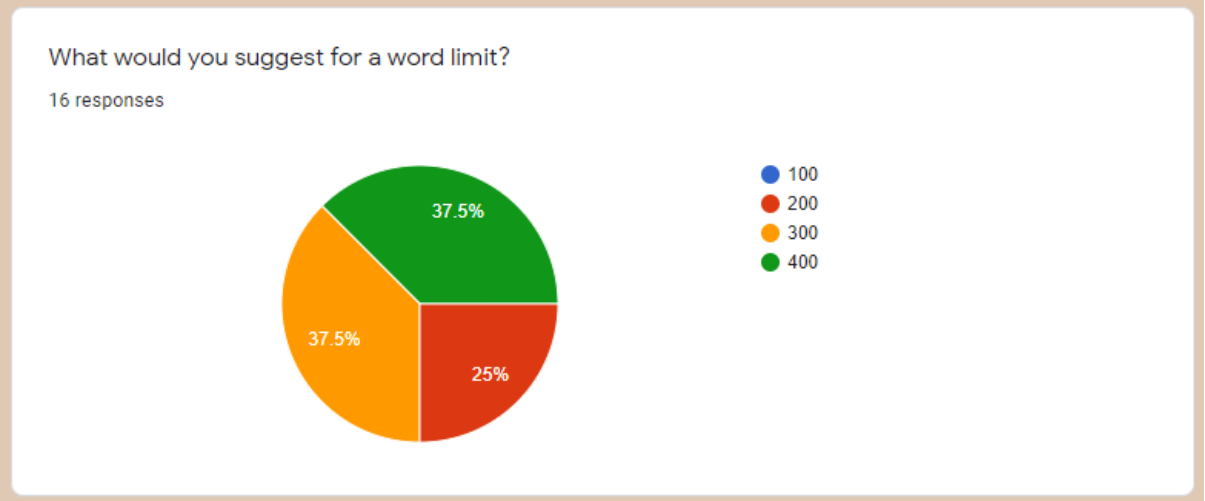


Figure 5 - What they would prefer as a word limit (Prevent text walls)

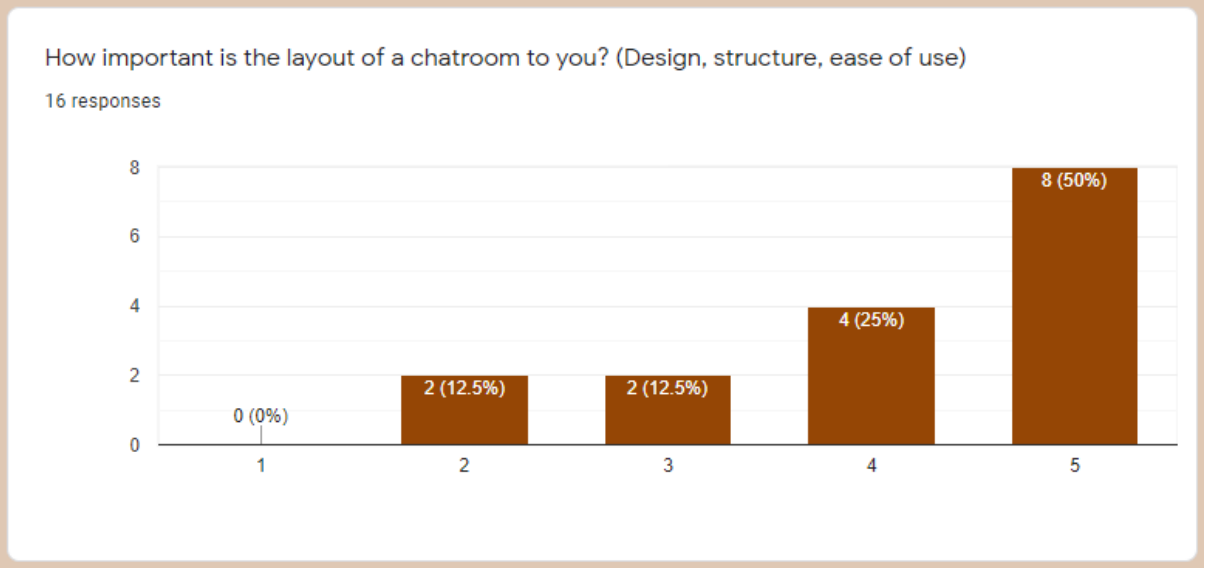


Figure 6 - The importance of the chatroom's design

1.2.1. Figure 1

I asked people if they have used any of the applications listed, all of which are popular chatting programs. From what I have gathered, most users have used the application Discord which is a modern chatting and voice calling application. Below Discord by a three-person difference is WhatsApp which is a commonly known application that is used to send direct messages and form group chats on mobile devices. Skype and Facebook appear to be the least used applications as together they make up only 38% of the total amount. With this information, I have decided to focus my research on Discord to formulate my chatroom.

1.2.2. Figures 2 - 3

I asked them what features they had enjoyed from these applications and what other features they may want as this will help me to establish the features that I will or may add onto my chatroom, given that I have the time. I could only display a select few responses; however, the consensus was that most users enjoyed the communication aspect of these whether that be text or voice. Users have also referred to the ability to customize their profile such as with statuses and profile images to which I will consider. Some users also mentioned private messaging and the ability to create a chatroom. While these would be an ideal feature to have, it may not be possible due to the limitations in set **(See pg. 18 1.12. Limitations and constraints).** This would also mean a friends/contacts list would be unnecessary to include, but despite this barrier, I can still try to comply with most of the features mentioned.

1.2.3. Figure 4

I questioned whether they would like to have a temporary name system where users can immediately connect to the chatroom or an account system for users to create a permanent profile with security. The results show that most users would prefer an account system to which I can comply with. With that in mind, account security will include a valid password system and no duplicate usernames.

1.2.4. Figure 5

The results here were split as some users wanted a two hundred limit and there was a split between a three hundred and four hundred limit. To settle on a limit, I have the three hundred limits as it falls comfortably between a split of users wanting three hundred and four hundred and a few users wanting two hundred. This would be the ideal amount to limit large amounts of text being sent.

1.2.5. Figure 6

The final question I asked regarded the design of the chatroom which would consequently fall in line with the rest of the website is important or not on a one to five scale. Most users have answered with a four and five, insinuating to me that users would enjoy a chatroom with little clutter and is easily accessible. This works in conjunction with my goal as I intended for an easy to use interface for my chatroom to improve upon its accessibility.

1.2.6. Conclusion of questionnaire

Taken together, I have identified the core features that I will need in my chatroom, this being able to create a secure account and having the ability to text with a limiter. I will also be required to focus on the design to comply with the user as the design and layout appears to be held in a high regard. Other features I have also taken into account such as a profile picture and my research topic has been declared which will be Discord.

1.3. Research

From a surface level analysis, a chatroom includes multiple users accessing a room where they can exchange messages with each other. Its simplicity is not enough for me to start creating a structure and design for my chatroom. With that in mind, to deepen my understanding of a chatroom, I have chosen the application Discord as my main focal point that I will study.

I have chosen Discord due to its modern innovation of chatrooms as the program has continuously grown and developed since its inception in 2015, successfully becoming one of the most popular platforms for users to socialise with each other to the same extent as programs like Skype and WhatsApp. In contrast to WhatsApp and Skype, Discord has become a phenomenon in recent culture due to its minimalistic approach to communication and its relative ease of use and diverse range of functions, making it both functional and enjoyable as Discord includes features such as voice calls that can handle up to 100+ users simultaneously and bots that can bring various personal features themselves like being able to play text based games, proving Discord's extensive capabilities.

1.3.1. Background

Discord was initially released on May 13th, 2015 by Discord Inc., built on the programming languages JavaScript, React, Elixir and Rust for PC and mobile operating systems such as Windows and Android. The program's intent was to allows users to instant message both text and media, voice call and video call one another within what they call a 'Server' which contains both a chatroom and voice/video channels in the most minimalistic way possible both visually and performance wise. The platform has since gained a substantial amount of popularity and as of 2019, it had reached over 250 million users.

The user demographic mainly consisted of people who play videogames as the initial intention of Discord was to create communities and groups where players could easily communicate with one another rather than using a loose in-game communication system. However, as the program has developed overtime, it started to branch out towards non-videogame users such as students, fanbases and programmers., making Discord an extremely diverse and social platform.

Figure 7 - Image of Discord logo

1.3.2. Breakdown

Access

Accessing Discord is relatively easy as an individual can use the platform from either the website or by downloading an application variant. From there, they will be required to create an account. Creating the account is simple as the user will only be required to enter a username, password, date of birth and an email. However, there are no mandatory validation rules as the only optional validation rule in place is confirming the email to verify the account. On access, the user can then begin to join servers, create servers, private chat or personalise their settings.

Process of entering communication

To communicate, the user can as mentioned join a server, create one or privately chat to a contact or group. Joining a server will require an invite. How the user accesses that invite is dependent on the server rules as a server can either publicise its invite for any user to enter whether that be on Discord's "Join server" option or by an external source or a server can private its invites where a user will require a direct message to be invited or the invite has an expiry date attached. A user can also create a server with the "Create server" option of which they can personally customize and attach privileges to its users. Additionally, the user can privately chat to another user or group so long as they have identified the end-user(s).

Process of communication

To communicate, users can either enter text into the chat bar at the bottom where the message will then be displayed or speak into their mic so long as they have the privilege to do so in the respective voice/text channel. Users are also able send emotes and attach a variety of files such as .txt, .jpg, .gif, .wav etc., although there are limitations on this as there is a max file size. Users also have the capabilities to alter another user's communication to them as for example, in a voice call a user can adjust the sound of that user on a client-side basis or block messages from that user.

Other notable features

- Users can also send friend requests to other users by clicking on their profile and clicking the "Send friend request" button which will add them to a friend's list.
- A role can be assigned to a user which will change their username colour as well as the privileges they have e.g. an Admin role can be assigned which can allow for a group of users to handle messages.
- All users in a server can be seen along the side in a list format where they are split into subsections. Each subsection's title is based on the role they have been given

Issues

Discord lacks any notably glaring issues as the platform has been designed with minimal errors. There is one issue however that it can suffer from which is a standard issue with any hosting service where if its servers were to be impacted heavily, then the service itself cannot continue running meaning that all communications will halt. Despite this issue, Discord rarely encounters issues with server maintenance or outages.

A potential issue that could be noted is the lack of end-to-end encryption when communicating, meaning attacks on Discord can lead to data from even private messages being leaked which can severely affect the platform's reputation and security concerns. However, Discord has not been a victim to these attacks as of recent.

Users on the platform can spread malware to other uses via files or websites, meaning that files and links themselves can be an issue if users do not practice safe etiquette when handling these issues. To prevent this, Discord has a message pop-up which can alert users of the problems they may run into, although this does not prevent it completely.

1.3.3. Conclusion of research

From my research, I have been provided some insight as to how my chatroom will go. First, I will present the user with the ability to sign-up or login. If the user is new, then they should choose to sign-up. I will only have the user enter a username and valid password as I will not require two-factor authentication nor a date of birth from the user. This will help to make the process of entering the chatroom easier as part of my objective.

For me to create multiple servers and the ability to private message another use would be too taxing on my device and the time I have, therefore I have chosen to neglect that feature that Discord has including the ability to communicate through audio as my chatroom will for now be strictly text based as part of my objective is to ensure that messages can be exchanged. Therefore, I have identified the process of texting where a user will have a text area for them to enter their message and a send button for it to then be received by the end-users.

Taking a note from other notable features, I could include a profile image and coloured usernames because it will allow for a customized experience for the user. As mentioned in the questionnaire, a friend request system would be unnecessary as users will not have the ability to private message or take advantage of a friendship system. I may also be able to implement a list feature to display which users are online.

Not encrypting messages will not appear as an issue because the chatroom will be public anyway and so there is no need to encrypt data other than the user's password. I will however suffer from an issue similar to the one in Discord which is server downtime as I cannot maintain the server constantly. Sending links will also be possible, however they will not be hyperlinked nor can files be attached as I do not intend the end-users being able to receive files from other users.

1.4. Interview

Interview with Ruiz Nagagor (End user) – Taken over a phone call - 23/12/20

Q1. What is the purpose of your website?

My aim is to simply allow for one individual to converse with another over a long distance. This is my original goal, although I want to improve upon this by adding more personal features to the website such as letting a user customize their own experience with name colours and profile pictures or by creating their own private chatroom. However, these aren't my main priorities and so the purpose of the website remains as an easy to use and accessible chatroom.

Q2. What is the process of entering the chatroom?

After receiving feedback from my questionnaire, it was clear to me that users want more security when accessing a social site.

So, when it comes to the process of entering the chatroom, the user will first be required to create an account. To ensure its security, a valid password will be required, and hashing will be used where their account can then be stored in a database I've created. After that, a user can then log in and access the chatroom immediately.

Q3. What else can you do in the chatroom?

For now, the user can only exchange and receive texts with other users. Taking in the user feedback has enlightened me to work on other features that I could possibly implement like being able to have a profile picture as mentioned or exchanging more than just text like gifs and images. There's a wide range of features I could implement, although this will be highly dependent on how much time I have left and the ability of my device which will act as the server.

1.5. Objectives

1.5.1. Objective overview

Taking my investigation together, I have formulated the process of entering and interacting with my chatroom which will begin with an account signup, login, access, and interaction with the features within the chatroom. Before setting these in place, I will need a design suitable for the chatroom. The design preferably will be kept simple and minimalistic while aligning with the fundamental human-computer interaction; perspective, to ensure that the user can view the chatroom with maximum visual clarity such as by having a contrasting background and foreground to highlight areas, similarly to the design of Discord as their minimalistic approach helps to navigate and interact with the server.

I will have the account creation prior to the chatroom itself similarly to Discord as this will ensure that valid users are entering the chatroom. The signup process will require a valid password from the user which will then be hashed as identified from my interview where I stated that “users want more security”. A unique ID should then be attached to the newly created account to easily identify them.

Progressing on is the login process where the user must then be able to sign into the account. This process will require the user to input an existing username with a matching password or else they will be presented with an error as part of account security.

Once logged in, the user can now access the chatroom. Identified from my investigation was the process of communication. I have stated that voice communications will not be allowed and so my focus is solely on the texting aspect of communicating. My main objective in the chatroom will be to allow for the user inputs and display recent messages for every user accessing the chatroom. From my questionnaire, I have concluded a message limit of three hundred characters which helps me as I will not require a large chatlog, keeping the chatroom’s layout as compact and minimalistic as possible as per the design. Messages will also include metadata such as the username of the user who sent the message and a timestamp to allow for unique identification of a message. The metadata will be highlighted in a colour to help ease the identification process, whether the user can adjust their metadata colour will be dependent on the time I have to allow for such a feature.

The chatroom will also include other features and processes inside to maximise user experience. Taking a note from Discord is the online feature which displays which users are online which I can include to aid users in identifying who else has access to the chatroom. I can also include a profile picture feature to help add further identity to the user when sending messages, although this could appear limiting as for example, the profile picture will be required to have a specific file type, size and dimensions. I will also include the option to logout for users exiting the chatroom.

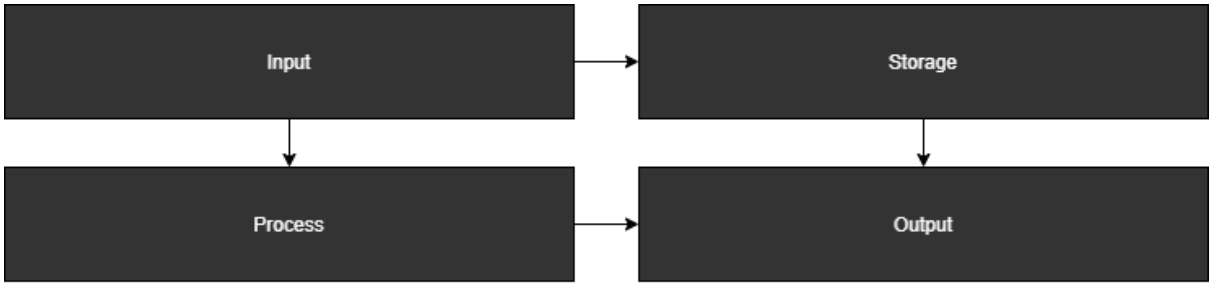
1.5.2. Objective table

Here displays the main specific objectives of my website and their performance criteria stating what will make them a success.

No.	Objective	Performance Criteria
1	Allow for a user to create a unique account securely	Sign-up process must include validation rules such as a password requiring a minimum of eight characters to ensure user's account is kept secure as possible. Additionally, a unique ID and username must be assigned for the user to be searched in the database efficiently and to allow for normalisation to occur. The process will ask the user for their: • Username • Password On successful completion, user must be directed to the login page or else an error message should appear stating what they have done incorrectly.
2	Create an area that can store user data which includes the user's username, password, messages, and profile pictures in an SQL database	User's account must only be accessible to the end-user and only they can define these values that are to be stored. The password must also undergo hashing as this will ensure further security of their account by making it extremely difficult to crack their password.
3	Allow for the user to log into their created account	End-user must be able to access their account through the login process without issue. The login process must contain a rule that verifies whether the username and password are matching to make sure that the user accessing the account is authenticated. This can be achieved by matching the inputs to existing data in the database. On successful completion, user must be directed to the chatroom or else an error message should appear stating what they have done incorrectly.
4	Have the user's account be associated with a message they will send by linking the two through a relation table	The message a user will send should be associated with the correct user as this will allow for metadata for other users to see who has sent the message.
5	Let users be able to send messages	When logged in, the user must be able to input messages. When inputting a message, the user should see what they are inputting and confirm the message they want to send.

6	Let the user be able to see all messages sent	User must be able to see who has sent messages alongside the time sent to provide metadata on the message. The messages must be also be displayed in order of time sent from most recent to least recent to keep it within a chronological order, making it easier to view recently added messages on the chatlog.
7	Add a feature that lets the user be able to see who is online	Logged in users must be able to see who else has logged in and has access to the chatroom which should be viewable to them through a section displayed somewhere on the user's screen.
8	Add a feature that will allow users to change their profile picture	Users must have the ability to upload and change a profile picture which can be displayed for others to view, giving the user a sense of identity and additional metadata. The image uploaded must be set to a certain size to prevent large images overtaking the chatroom. A button should be included to allow for this to be optional. A default profile picture should also be attached if the user lacks a personal image to prevent errors occurring.
9	Let users log out	Allow for users to log out of their logged in account which should remove them from the online list. A button will be displayed for the user somewhere on their screen.

1.6. IPSO Chart



IPSO	Information	Evidence
Input	**User account information** • Username • Password • Profile picture • Online information **Message information** • Message **Button clicks** • Sign-up • Login • Logout • Submit	Observation Interview
Process	**User account** Take user information and direct it to the appropriate table in the database on valid signup. Then have the information available to use in other processes which includes the login process, having their account attached to the message they send and the profile picture they upload. Also, set user online status to either 0 or 1. **Messages** Have messages directed to the appropriate table in the database on message send. Then have the message attached to the user who has sent the message and display it with the user's username and a timestamp of when it was sent. Other processes include appropriately responding to the user in response to an event which includes directing them to the appropriate page, having a section appear before them on button click such as uploading a profile picture and refreshing the chatlog and online users.	Observation
Storage	**Information to be stored** • User account information (User ID, Username, Password) • Profile picture (Image ID, User, Image) • Messages (Message ID, Message, Timestamp) • Users online (User ID, Username)	Observation
Output	**Information to be outputted** • User interface and information to guide and notify the user • User's username • User's profile picture • Message and its metadata • All users that are online	Observation

Figure 8 - IPSO Chart (Analysis)

1.7. Data Dictionary

Data Item	Data Type	Validation	Sample Data
User ID	Int	Not Null	0000001
Username	String	Not Null	Eren9
Password	String	Not Null	qwn4al94s
Message ID	Int	Not Null	0000001
Message	String	<=300	This is a sample message
Timestamp	Date/Time	Not Null	2020/07/20 - 17:38:00
Image ID	Int	Not Null	0000001
Image	String	Not Null	example.jpeg
isOnline	INT	Not Null	0 (False)

Figure 9 - Data Dictionary table (Analysis)

1.8. Context Free Diagram

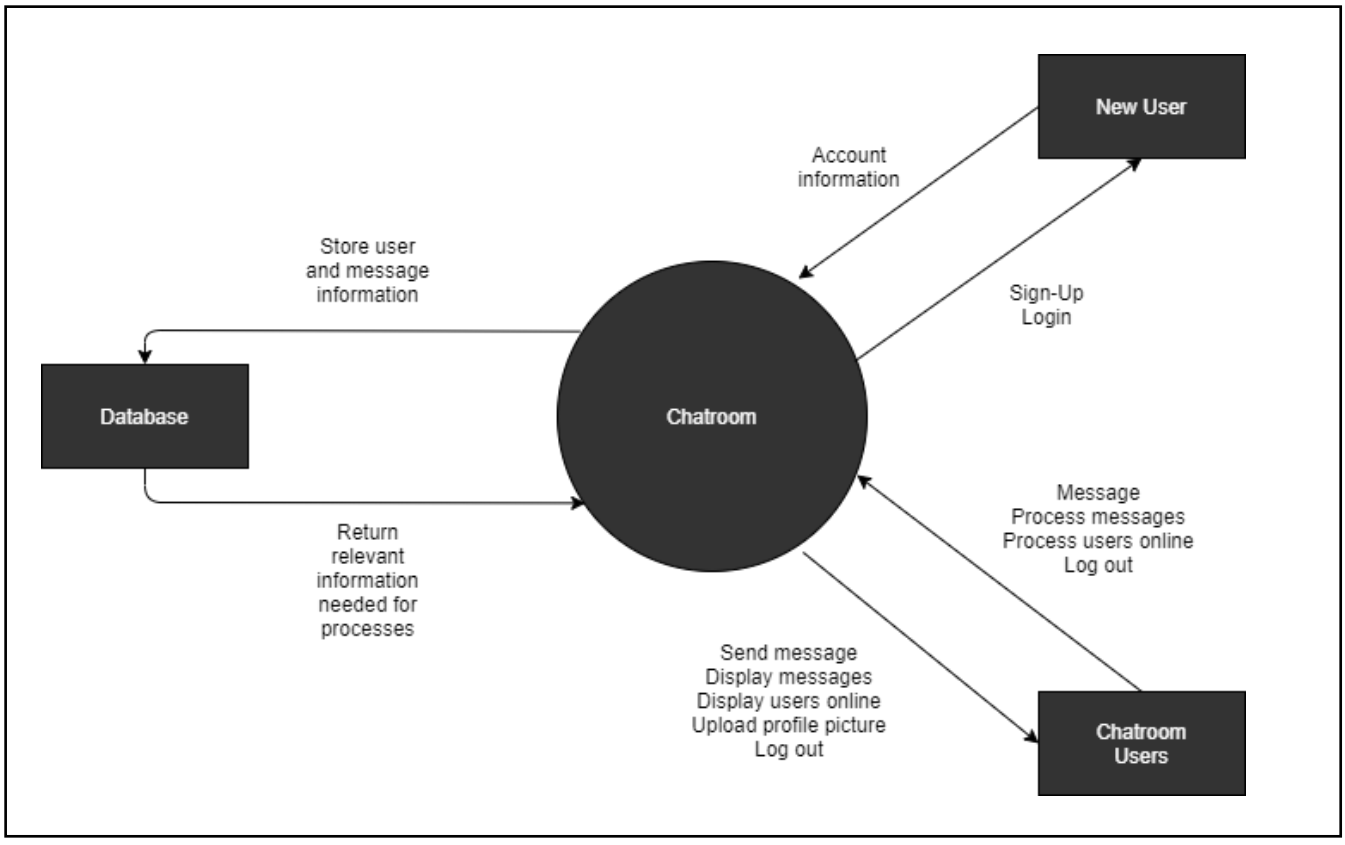


Figure 10 - Context Free Diagram

1.9. Data Flow Diagram

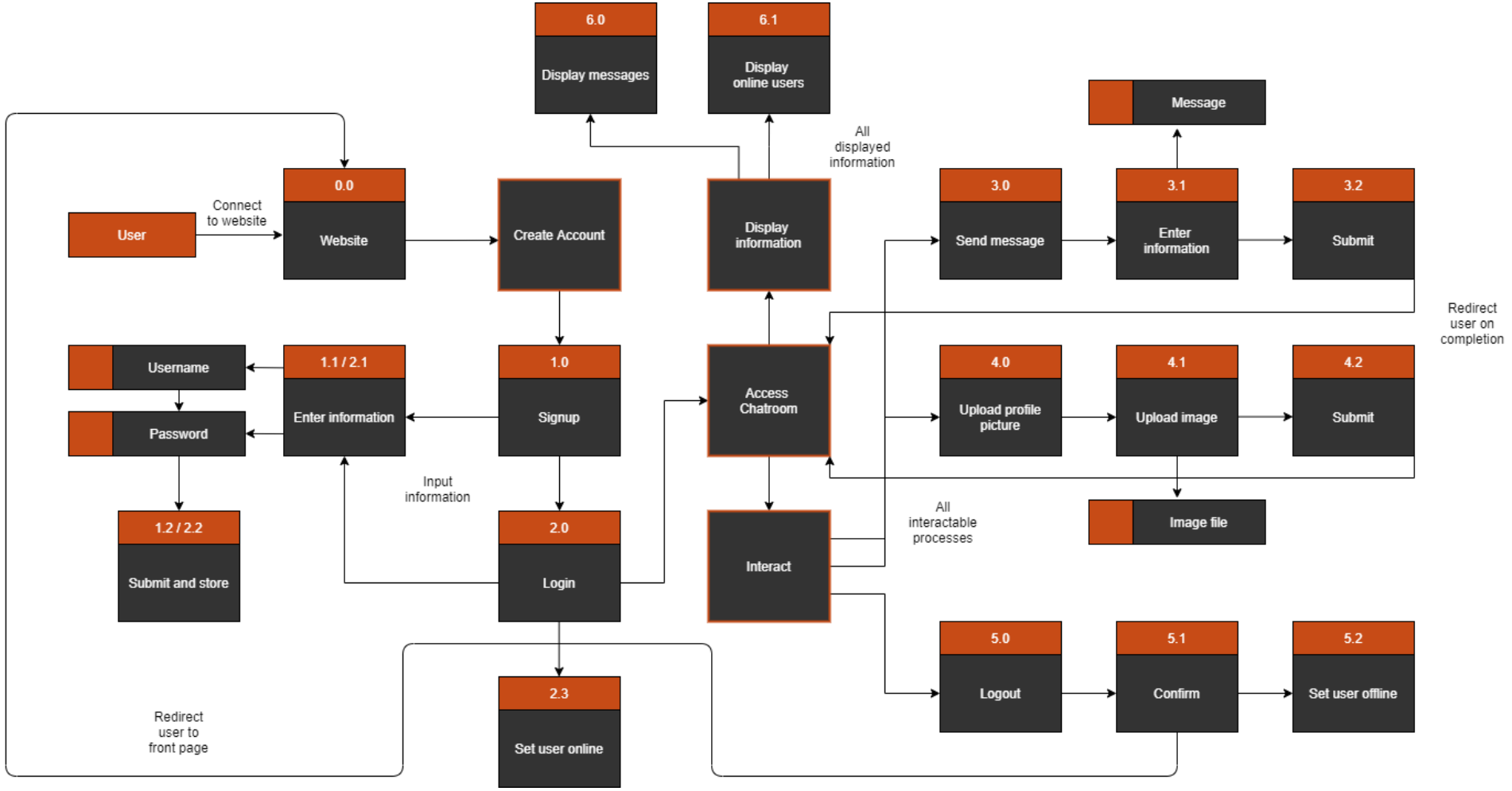


Figure 11 - Data Flow Diagram

1.10. Entity Relationship and Attribute Model

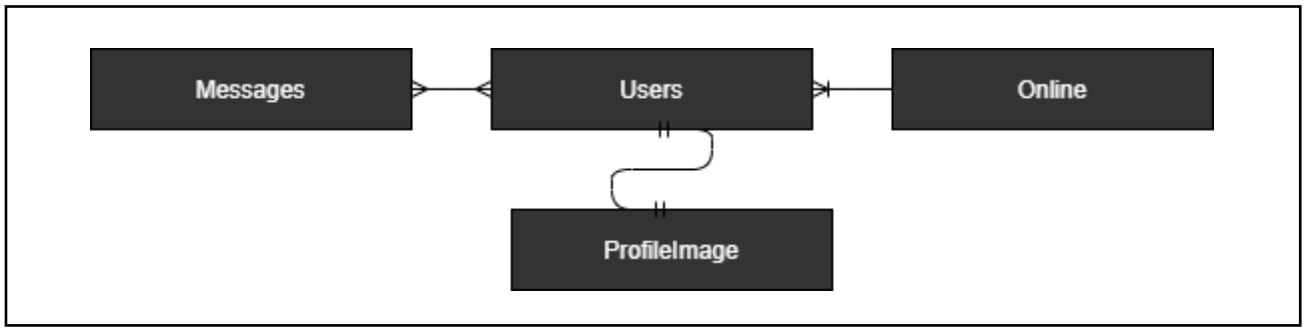


Figure 12 - Entity Relationship Model (Analysis)

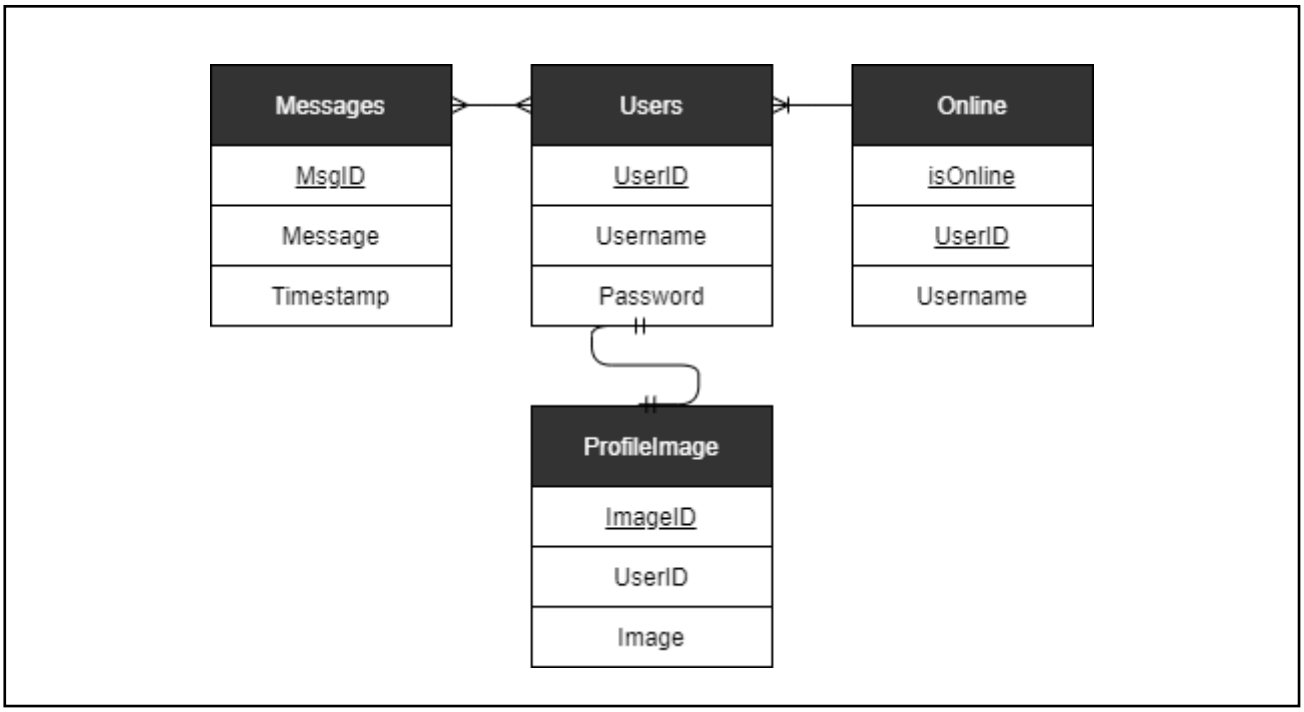


Figure 13 - Entity Attribute Model (Analysis)

1.10.1. Explanation of figures above

The entity models above represent the relationships and database notations that will be included for each table.

- Messages and Users will have a many-to-many relationship
- Users and Online will have a many-to-one relationship
- Users and ProfileImage will have a one-to-one relationship

1.11. Potential solutions

To formulate this chatroom, I will require a substantial amount of information regarding databases. Fortunately, the computer science course that I have been recently studying has given me an in depth understanding of how a technological database works, specifically the relational database management system SQL which can create, read, update, and delete data. SQL is ideal for this system as my system revolves around multiple types of data being handled ranging from assigning unique IDs to user accounts to the message information being sent. With SQL, I can store the account information and messages in a database which can then allow me to procure that stored information into my program. This means that I can manipulate the database data within my program for its intended use which is to be able to use the account information to login and identify the user, view and send messages, see which users are online and be able to upload a profile picture.

Further research into websites has led me to utilise four programming languages that will be the foundation upon which the website is built on. These languages are HTML, CSS, PHP, and AJAX. HTML and CSS are the standard programming languages used for all websites as they are what structure the design and layout of the website. These languages will henceforth be used to ensure that the website's design is as user friendly possible.

PHP on the other hand will be the language used to convert the website from a static website to a dynamic one. This is because PHP is a scripting language that will offer itself as the back end to my website whereas HTML and CSS act as the front end. By being the backend, it allows for my website to accept and process data that a user has inputted meaning that I am able to use this language to store the account information that is provided alongside the messages which can then be used to be displayed onto the website. Additionally, as PHP is a scripting language, it allows for algorithms and functions to be programmed meaning that I can provide my website with further security and validity of the information by evaluating the information that is sent, making the language ideal as it enables me to manipulate the data that has been processed. PHP can also be used to set my database notations, proving further utility.

As for AJAX which is a variant of the language JavaScript, it will allow me to make a specific section of my website dynamic which will be the chat log. This makes sure that the chat log is kept up-to-date and doesn't require the user to refresh the entire page themselves. This would also be used for my online users as I can use the language to keep the user updated on which users are online and have gone offline.

1.12. Limitations and constraints

1.12.1. Data protection

As the website handles user information, I will be required to comply to the Data Protection Act where I then must follow the Data Protection Principles, specifically where it states that data must be ***"handled in a way that ensures appropriate security, including protection against unlawful or unauthorised processing, access, loss, destruction or damage".***

To ensure that the principle is being met, my database will be kept secure by preventing any outsider access other than me when using the database which can be done by locking the database which can be accessed via an account with a randomly generated password. Additionally, passwords users submit will be secured to prevent unauthorised accessed to the account which can be done by using a hashing algorithm which will scramble the password in a one-way system, therefore keeping the user's password secure as it would be difficult for an unauthorised user to crack the user's password.

1.12.2. Time constraints

Due to the limited time I have on this project, I cannot guarantee that all systems will work as intended, especially systems that may be too ambitious or taxing due to the server resources of my website being allocated to my computer which could strain my device and lead to unwarranted effects on my website which can be caused by systems such as users having and displaying a profile picture or exchanging media besides text.

As I also require conceptualizing this project through a design and test, I will have to balance both this and the project itself until my due date and so some systems may not be finished to the extent I wanted to or I cannot implement any of the features I intended to include or implied to – Reason as to why I have decided to focus on a small set of objectives to prioritise as these will be the core features of the chatroom, therefore they will require more focus than other aspects I would like to incorporate due to their importance.

END OF ANALYSIS

2. Design

Below is a key explaining the format within the code sections further on:

```
Code – Consolas format indicates text as a line of code

Variable – Italic and Underlined text represents a variable

Statement – Emboldened text indicates a programming statement e.g. IF, AND, FOR

SQL – Orange text represents SQL statements
```

2.1. Database Notation

2.1.1 Original

```
Users (UserID, Username, Password)

Messages (MsgID, Message, Timestamp)

ProfileImage (ImageID, UserID, Image)

Online (isOnline, UserID, Username)
```

2.1.2. Normalised - Relational table created

```
Users (UserID, Username, Password)

Messages (MsgID, Message, Timestamp)

UserMsg (UserID, MsgID, ImageID)

ProfileImage (ImageID, UserID, Image)

Online (isOnline, UserID, Username)
```

2.2. Database Diagrams

2.2.1. Entity relationship diagram

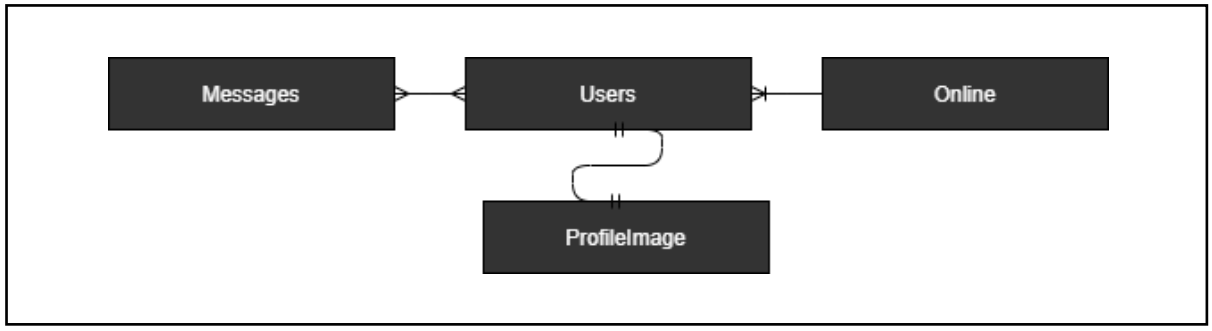


Figure 14 - Entity Relationship Diagram

2.2.2. Post-Normalisation Entity Relationship Diagram

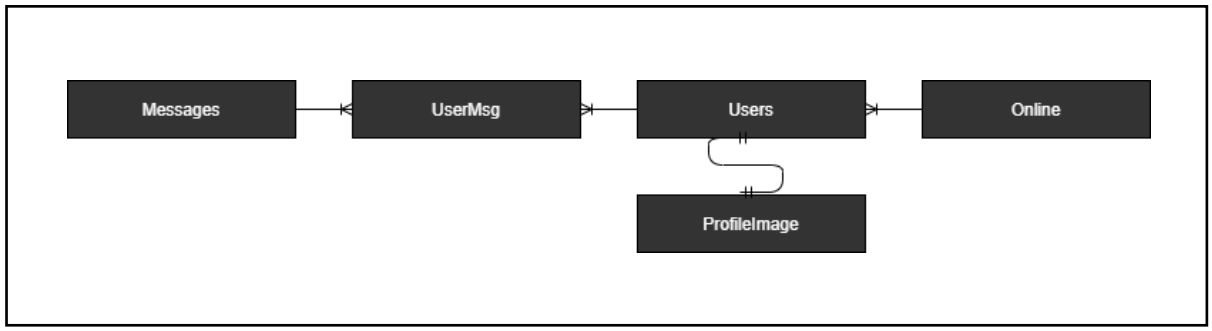


Figure 15 - Post-Normalisation Entity Relationship Diagram

2.2.3. Entity Attribute Diagram

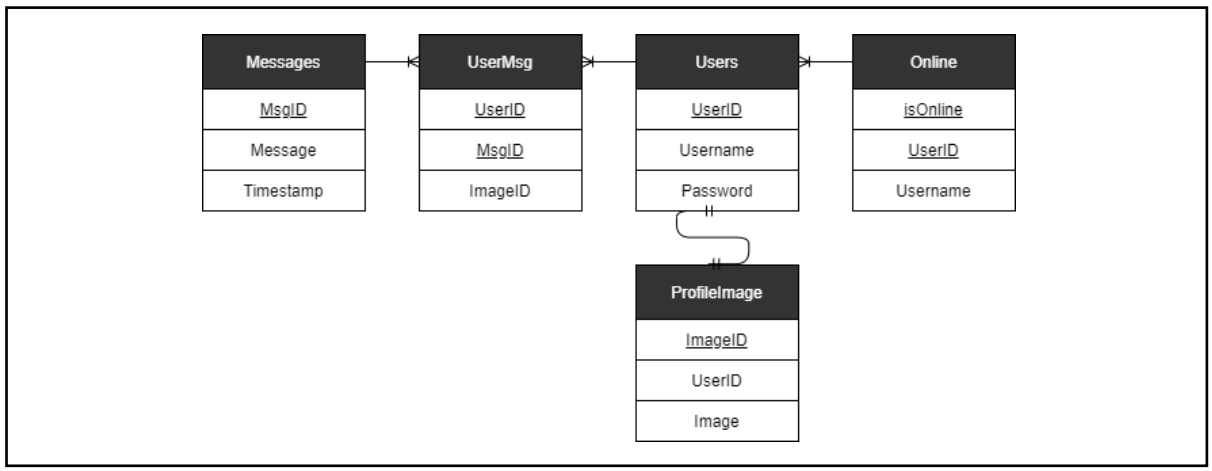


Figure 16 - Entity Attribute Diagram

2.2.4. Data Dictionary

Users			
Primary Key		UserID	
Foreign Key		N/A	
Data Item	Data Type	Validation	Sample Data
UserID	Int	NOT NULL	0000001
Username	String	NOT NULL	Eren9
Password	String	NOT NULL	qwn4al94s

ProfileImage			
Primary Key		ImageID	
Foreign Key		UserID	
Data Item	Data Type	Validation	Sample Data
ImageID	Int	NOT NULL	0000001
UserID	Int	NOT NULL	0000001
Image	String	NOT NULL	example.jpeg

Messages			
Primary Key		MsgID	
Foreign Key		N/A	
Data Item	Data Type	Validation	Sample Data
MsgID	Int	NOT NULL	0000001
Message	String	<=300	This is a sample message
Timestamp	Date/Time	NOT NULL	2020/07/20 - 17:38:00

Online			
Primary Key		UserID, isOnline	
Foreign Key		UserID	
Data Item	Data Type	Validation	Sample Data
isOnline	Int	NOT NULL	0 (False)
UserID	Int	NOT NULL	0000001
Username	String	NOT NULL	Eren9

UserMsg			
Primary Key		UserID, MsgID	
Foreign Key		UserID, MsgID, ImageID	
Data Item	Data Type	Validation	Sample Data
UserID	Int	NOT NULL	0000001
MsgID	Int	NOT NULL	0000001
ImageID	Int	NOT NULL	0000001

Figure 17 - Data Dictionary

2.2.5. Explanation of relationships

Many users accounts will be created, therefore a UserID attribute has been created to easily identify these unique user accounts where an automatic value will be provided on creation of their account.

Many messages will be sent; therefore, they have a unique MsgID to create a relation with the user which will be incremented as each new message is sent – Preferably messages would remain to allow for a chat history.

A many-to-many relationship can be identified here, and so a relational table has been created to amend this issue as it will convert the database into first normal form because it would prevent attributes being repeated.

A user can also have a profile picture and so a one-to-one relation will exist between the Users table ProfileImage table. ImageID is also stored as a foreign key in UserMsg to allow for their picture to be displayed alongside the message data they send.

For the online table, many users can be online at once, therefore a many-to-one relationship can be used to connect the two tables so that the logged user can see who is online on the chatroom.

2.3. SQL Queries

2.3.1. Creation of database and tables

```
CREATE DATABASE ChatroomDB
```

```
CREATE TABLE Users(
  UserID int NOT NULL PRIMARY KEY AUTO_INCREMENT,
  Username varchar(30) NOT NULL,
  Password varchar(255) NOT NULL
);
```

```
CREATE TABLE Messages(
  MsgID int NOT NULL PRIMARY KEY AUTO_INCREMENT,
  Message varchar(300),
  Timestamp timestamp NOT NULL
);
```

```
CREATE TABLE UserMsg(
  UserID INT,
  MsgID INT,
  ImageID INT,
  PRIMARY KEY (UserID, MsgID),
  FOREIGN KEY (UserID) REFERENCES users(UserID),
  FOREIGN KEY (MsgID) REFERENCES messages(MsgID),
  FOREIGN KEY (ImageID) REFERENCES profileimage(ImageID)
);
```

```
CREATE TABLE ProfileImage(
  ImageID INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
  UserID INT,
  Image varchar(),
  FOREIGN KEY (UserID) REFERENCES users(UserID)
);
```

```
CREATE TABLE Online (
  isOnline INT NOT NULL,
  UserID INT NOT NULL,
  Username VARCHAR(300) NOT NULL,
  FOREIGN KEY (UserID) REFERENCES users(UserID),
  PRIMARY KEY (isOnline, UserID)
);
```

2.3.2. Check if username exists

```
SELECT Username
FROM Users
WHERE Username = Username
```

2.3.3. Store user details into Users table is valid information

```
INSERT INTO Users(Username, Password)
VALUES (Username, Password)
```

2.3.4. Check if password matches username

```
SELECT Password
FROM Users
WHERE Username = Username
```

2.3.5. Get the user's UserID

```
SELECT UserID
FROM Users
WHERE Username = Username
```

2.3.6 Get the user's ImageID

```
SELECT ImageID
FROM profileimage
WHERE UserID = userID
```

2.3.7. Insert new message into Messages table

```
INSERT INTO Messages(Message)
VALUES (Message)
```

2.3.8. Store UserID, MsgID and ImageID on message sent

```
INSERT INTO UserMsg(UserID, MsgID, ImageID)
VALUES (userID, msgID, imageID)
```

2.3.9. Get the messages, username, and timestamp to be displayed in log in order of timestamp

```
SELECT Users.Username, ProfileImage.Image, Messages.Message,
Messages.Timestamp
FROM Users
INNER JOIN UserMsg ON Users.UserID = UserMsg.UserID
INNER JOIN Messages ON Messages.MsgID = UserMsg.MsgID
INNER JOIN ProfileImage ON ProfileImage.ImageID = UserMsg.ImageID
ORDER BY Messages.Timestamp
```

2.3.10. Insert user into ProfileImage table

```
INSERT INTO ProfileImage(UserID)
SELECT UserID
FROM Users
WHERE Users.Username = Username
```

2.3.11. Update Image to default profile picture

```
UPDATE ProfileImage
SET Image = "Default.png"
WHERE UserID = userID
```

2.3.12. Update Image to uploaded profile picture

```
UPDATE ProfileImage
SET Image = Image
WHERE UserID = userID
```

2.3.13. Get profile picture

```
SELECT Image
FROM ProfileImage
WHERE ProfileImage.UserID = userID
```

2.3.14. Insert users into Online table and set isOnline to false (0) on signup

```
INSERT INTO Online(UserID, Username)
SELECT UserID, Username
FROM Users
WHERE Users.Username = Username
```

2.3.15. Set isOnline value of user to true (1) on login

```
UPDATE Online
SET isOnline = 1
WHERE Username = Username
```

2.3.16. Set isOnline value of user to false (0) on logout

```
UPDATE Online
SET isOnline = 0
WHERE Username = Username
```

2.4. Flowchart Diagram

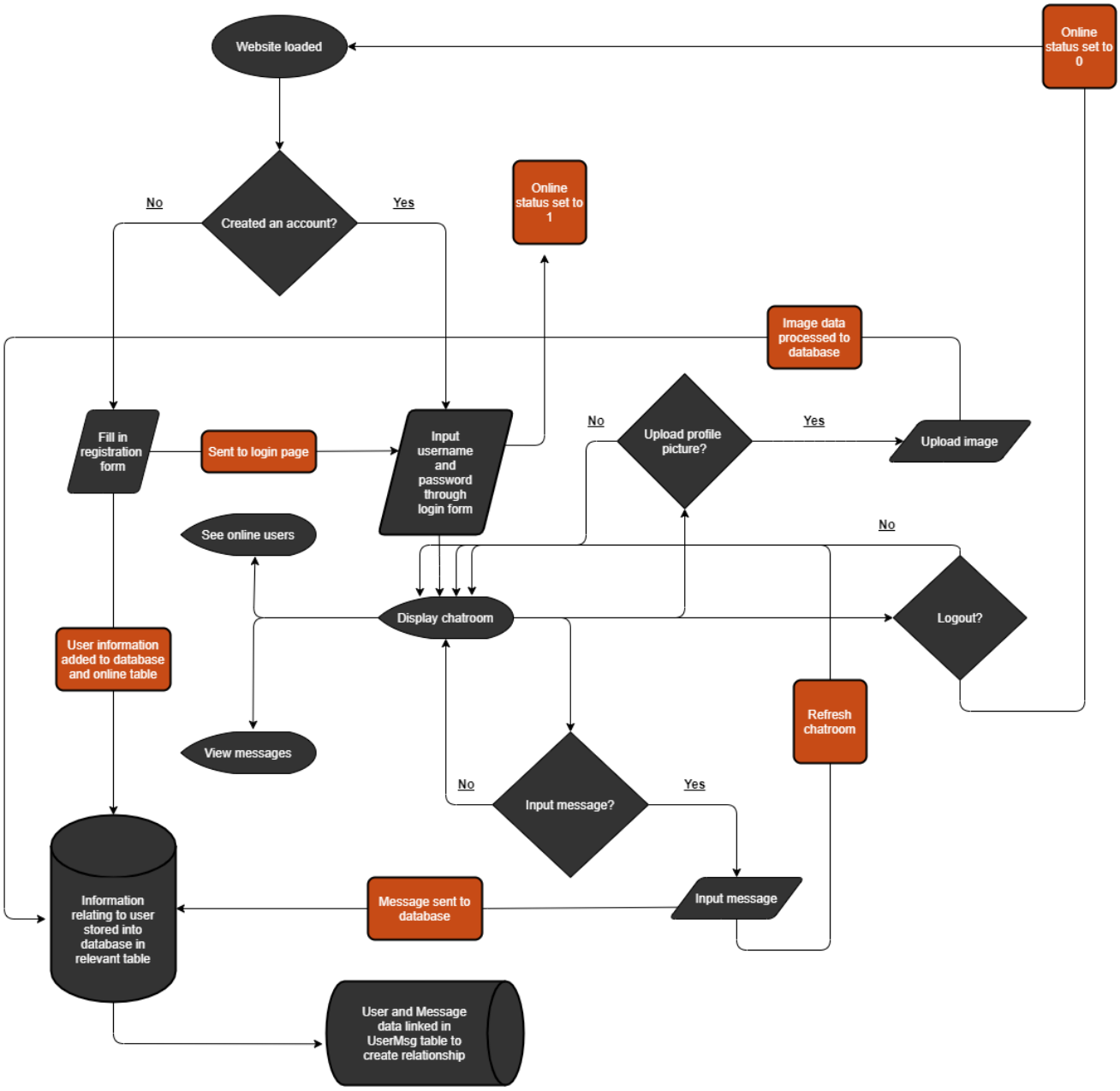


Figure 18 - Flowchart Diagram

2.5. Top-Down Diagram

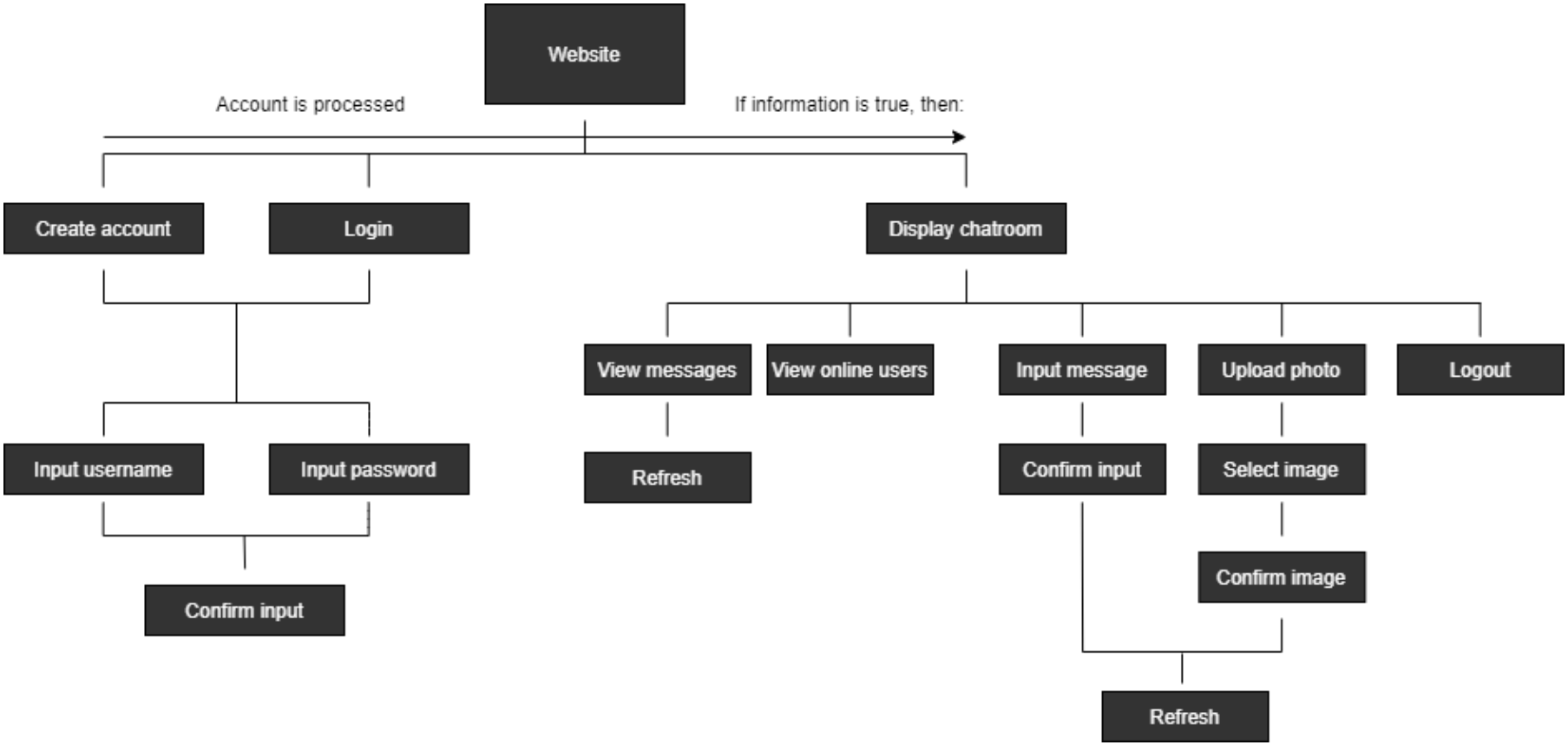


Figure 19 - Top-Down Diagram

2.6. IPSO Chart

IPSO	Program section	Item
Input	Get user login and sign-up information	Username User password
	Accessing the chatroom	Login button Get valid username Get matching password
	Get text	Text message Username Timestamp
	Get profile picture	Image file
	Exit chatroom	Logout button
Processing	Verifying user information (sign-up)	Did the user fill in the forms? Did the user enter a valid password?
	Securing user information	Has the user's password been encrypted?
	Validating user information (login)	Does the user information match existing data?
	Add user to online table	Has the user signed up?
	Change user online status	Has the user logged in or logged out?
	Add user to profile image table	Has the user signed up?
	Update user profile picture	Did the user upload a new profile picture?
	Check for message	Did the user send a message?
	Check if exited	Did the user logout?
Storage	Save user information	Username and password saved to database
	Save user login details	User's account is kept logged in by saving information to temporary session
	Save text messages	Sent text messages are saved to database
	Save profile picture	Profile picture saved to database
Output	Redirect user	User is redirected to the appropriate page after input
	Display username	Username displayed in the top left when logged in
	Display text messages	All text messages that have been sent are displayed in the chat log
	Display user's profile picture	User's profile picture displayed along the side and alongside the message
	Display online users	All currently logged in users added to online section

Figure 20 - IPSO Chart

2.7. Interface Design

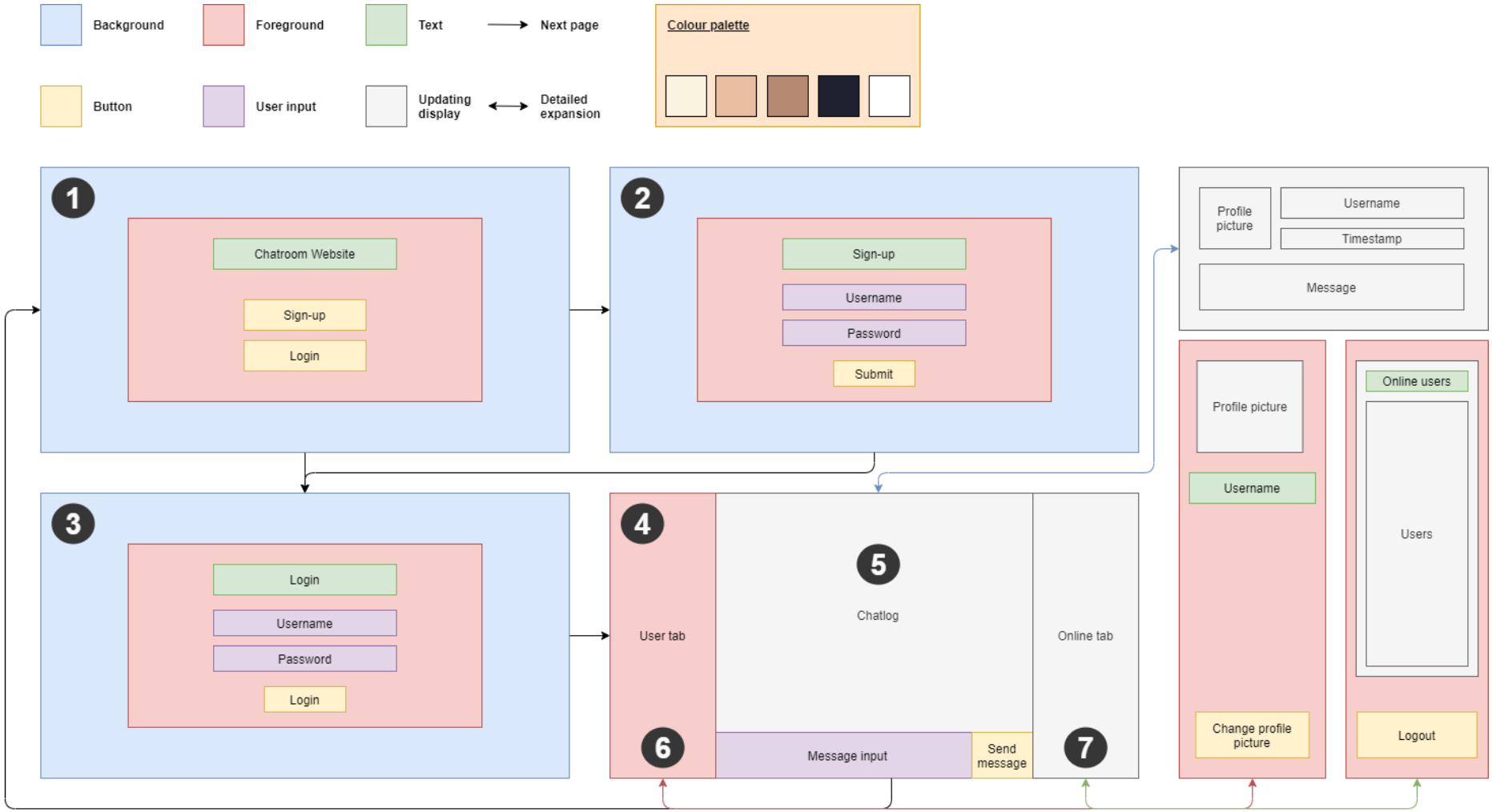


Figure 21 - Interface Design

2.7.1. Interface explanation

1. **Index** – This will be the first page presented to the user containing the title of the website alongside two buttons: 'sign-up' and 'login'. Each button will direct the user to the page appropriate to the button.
2. **Sign-up** – This page will be a result of selecting the 'sign-up' button on the Index page. Here, the user can input a username and password where they can then submit the information which will be checked for its validity. If successful, they will be directed to the Login page.
3. **Login** – This page will be a result of selecting the 'login' button on the Index page or successful submission during the sign-up process. Here, the user can input an existing username and password which they can submit to assess whether they are valid. If successful, they will be redirected to the Chatroom page.
4. **Chatroom** – The chatroom, the main function of the site, presents multiple interfaces for the user to digest. The user can view the chatlog which will include newly added data, enter messages, and submit them, view online users, and interact with the sidebar.
5. **Chatlog** – The chatlog section will contain all the messages that have been submitted by users which will display the user's username, profile picture, message and the time it was sent in a list format with the recent messages at the bottom.
6. **User tab** – The sidebar will allow for further user interaction by allowing the user to change their profile picture and preview it alongside being able to logout, resetting their isOnline status to 0. It is not limited however to these two interactions.
7. **Online tab** – The online section will continuously be updated to display the users who have logged in.

I have aimed for a minimalistic design similar to Discord as this will assist user navigation and enhance the ease of accessibility when using the site as different parts will clearly be sectioned and have a clean aesthetic to them to keep the design neatly formatted.

The colour palette I have chosen consists of varying levels of cream alongside black and white. This is done with contrast in mind as part of my objective is to ensure user accessibility, given the audience. Therefore, this can be achieved by providing colours that clash with one another. They are also soft, pastel colours which can help to prevent user eyestrain.

2.8. Algorithms and classes

2.8.1. Sign-up validation

As I have an account system in place, I will require a process that will allow for a user to create an account. To do this, I will create a form that will query the user for a Username and Password. The username will be validated to see if it is available (To prevent duplicates), if so; an error will be returned. The password will then be validated through multiple validation rules e.g. regex pattern to ensure that their password is strong. If either field is empty, an error will be returned.

If no errors have been returned, their account details will be stored in three tables: Users, ProfileImage and Online which will be manipulated elsewhere in the program. The user will then be redirected to the login page.

2.8.1.1. Sign-up pseudocode

```
Username = USER INPUT
Password = USER INPUT
SignUp(Username, Password)
  IF Username IS NULL OR Password IS NULL
    RETURN "Must fill in all boxes"
  ELSE IF Username EXISTS
    RETURN "Username already taken"
  ELSE IF Password.LENGTH < 8
    RETURN "Password must be greater than 8 characters"
  ELSE IF Password DOES NOT CONTAIN Regex([A-Z]+[a-z]+0-9]+)
    RETURN "Password must contain at least one uppercase and lowercase
letter and one digit"
  ELSE
    INSERT INTO users(Username, Password) VALUES (Username, Password)
    INSERT INTO online(UserID, Username) SELECT UserID, Username FROM
Users WHERE users.Username = Username
    INSERT INTO profileimage(UserID) SELECT UserID FROM Users WHERE
users.Username = Username
LOCATION -> "login.php"
```

2.8.2 Login validation

The user can then proceed to enter their account details onto the login form where they will be queried again for their username and password with different validation rules. The username and password inputted will be evaluated for whether the username appropriately matches the password associated with it in the Users table. If not, an error will be returned. If either field is empty, an error will be returned.

If the login was successful, the user's isOnline status in the Online table will be updated to 1, their UserID will be retrieved for later use and they will have access to the chatroom.

2.8.2.1. Login pseudocode

```
Username = USER INPUT
Password = USER INPUT
Login(Username, Password)
IF Username IS NULL OR Password IS NULL
  RETURN "Must fill in all boxes"
ELSE
  userPassword = SELECT users.Password WHERE users.Username = Username
  IF userPassword == Password
    UPDATE online SET isOnline = 1 WHERE Username = Username
    START SESSION
    LOCATION -> "chatroom.php"
  ELSE
    RETURN "Invalid username or password"
```

2.8.2.2. UserID and ImageID Pseudocode

```
userID = SELECT users.UserID FROM users WHERE users.Username = Username
```

```
imageID = SELECT ProfileImage.ImageID FROM ProfileImage WHERE UserID =
userID
```

2.8.3. Entering messages

A text area will be displayed on-screen for the user to enter a message. When a message has been typed out and submitted via a submit button, the message will be stored in the Messages table in the database where it will be assigned a MsgID so that it can be retrieved to associate with the user later in the UserMsg table. If an empty box is submitted, nothing will happen to prevent blanks being stored in the database.

2.8.3.1. Entering messages pseudocode

```
Message = USER INPUT
sendMessage(Message)
IF Message IS NULL
  RETURN "Cannot send an empty message"
ELSE
  newMessage = INSERT INTO messages.Message VALUES (Message)
  msgID = GET LastID OF newMessage
  Result = INSERT INTO usermsg(UserID, MsgID, ImageID) VALUES (userID,
msgID, imageID)
IF Result
   REFRESH
 ELSE
   RETURN "Error sending message"
```

2.8.4. Displaying messages

A chatlog will be displayed on the screen which will be used to display the messages that users have sent with appropriate metadata. The user from the Users table, message and timestamp from the Messages table and image from the ProfileImage table will be retrieved for this, however the UserMsg table will be needed as it is a relational table between the three. By using this table, I can associate the user with their message and profile picture despite being from two different tables.

On successful relation, I can store the message as a variable and output it onto a new line in the chatlog whilst refreshing the log every set interval or on detection of a new message if possible.

2.8.4.1. Displaying messages pseudocode

```
Chatlog = NULL
newMessage = SELECT Users.Username, ProfileImage.Image, Messages.Message, Messages.Timestamp
FROM Users
INNER JOIN UserMsg ON Users.UserID = UserMsg.UserID
INNER JOIN Messages ON Messages.MsgID = UserMsg.MsgID
INNER JOIN ProfileImage ON ProfileImage.ImageID = UserMsg.ImageID
ORDER BY Messages.Timestamp
FOR NEXT AVAILABLE Line IN Chatlog
  WRITE Username, Timestamp, Message
END
```

2.8.4.2. Refreshing chatlog pseudocode

```
IF NEW Line IN Chatlog
  REFRESH
    SCROLL TO Bottom OF Chatlog
OR
EVERY 7000 MILISECONDS
  REFRESH
    SCROLL TO Bottom OF Chatlog
```

2.8.5. Online users and logging out

Users can see other users that are online through an online log displayed along the sidebar. To display the online users, users who have a value of 1 in the isOnline table will be selected as 1 refers to them being online as true. The online log can also be refreshed every set interval or on detection of a change made in the Online table as the UPDATE query is being used to adjust their isOnline value.

The users can also logout which will remove them from the chatroom, set their isOnline value to 0 and redirect them to the index page on click.

2.8.5.1. Online users pseudocode

```
onlineLog = NULL
onlineUsers = SELECT Username FROM online WHERE isOnline = 1
IF NEW UPDATE IN TABLE Online
    FOR NEXT AVAILABLE Line IN onlineLog
    WRITE onlineUsers
  END
OR
EVERY 7000 MILISECONDS
  FOR NEXT AVAILABLE Line IN onlineLog
    WRITE onlineUsers
  END
```

2.8.5.2. Logout pseudocode

```
Logout = BUTTON
IF Logout
  UPDATE isOnline SET isOnline = 0 WHERE Username = Username
  DELETE SESSION
  LOCATION -> “index.php”
```

2.8.6. Profile picture

Users will have the ability to upload a profile picture that would ideally be seen by other users. As the user has been added to the ProfileImage table on sign-up, their Image value will be null. If it is null, the user will receive a default profile picture to ensure everyone has one. If the user wants a custom profile picture, they can select Upload where they will be prompted to do so. Validation rules will then be used to assess the user's uploaded file and if no error is returned, an SQL update query will be used to replace the default profile picture.

The image can then be displayed alone along the side of the user's screen by selecting the user and their profile image with set dimensions.

2.8.6.1. Upload profile picture pseudocode

```
Upload = BUTTON
File = NULL
Image = SELECT Image FROM profileimage WHERE profilemage.UserID = userID
IF Image IS NULL
  UPDATE profileimage SET Image = "Default.png" WHERE UserID = userID
IF USER CLICKS Upload
  DISPLAY File Browser
    IF File SELECTED AND Confirmed
      IF File ENDS WITH ".jpg" OR ".png"
        File = SELECTED File
        STORE File IN SERVER DIRECTORY
        Result = UPDATE profileimage SET Image = File.name WHERE UserID =
userID
        IF Result
          REFRESH
        ELSE
          RETURN "Error uploading image"
      ELSE
        RETURN "Invalid format"
```

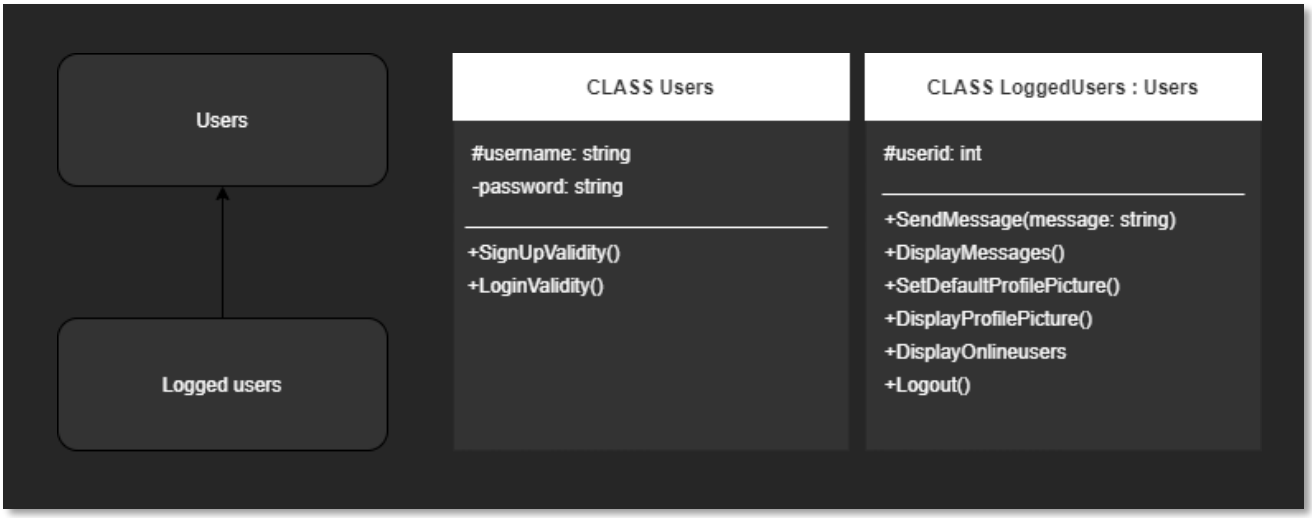
2.8.6.2. Displaying current profile picture pseudocode

```
Image = SELECT Image FROM profileimage WHERE profileimage.UserID = userID
IF Image IS NOT NULL
  SET IMAGE SIZE
  DISPLAY Image
```

2.8.7. Class diagram

Most of the pseudocode above will be turned into methods of a class, these being the Users and LoggedUsers. The Users class will consist of methods and properties that will be appropriate for a user who has not yet logged into the chatroom.

The LoggedUsers class will be a derivative of class Users to ensure the same properties can be used within the LoggedUsers constructer. LoggedUsers will contain the methods appropriate for a user who has logged into the chatroom page. Below depicts a diagram of the classes.



END OF DESIGN

3. Technical solution

Note

This section of the document consists of the code used to achieve the solution. The languages used are PHP, JavaScript, HTML and CSS.

The languages have been used in various files within a directory on my computer labelled "chatserver" which will also store the user's profile pictures.

I have activated the modules Apache and MySQL using the open-source web server solution program XAMPP because this enables my computer to act as a local web server. Below details what the modules allow for.

- **Apache** – Apache opens ports 80 and 443 on my computer. With port 80, I can allow for internet communications to take place via hyper-text-transfer-protocols, suitable for my chatroom as data across the internet will be exchanged. With port 443, it provides security for web browser communications which is a standard to ensure hyper-text-transfer-protocol-secure.
- **MySQL** – MySQL opens port 3306 on my computer which is the port for the MySQL protocol. This protocol allows my computer to store the database alongside its tables and respective data that I have conceptualised.

To provide a visual representation of these tables, phpMyAdmin will be used which is an open source tool that allows for administration of MySQL which will be useful as this assists me in understanding how the data is being handled and it will allow for me to provide images as evidence during the white-box testing phase as this will assess my program's internal functionality.

3.1. Evidence of setting up tables and relationships

```
$serverName = "localhost";
$Username = "root";
$Password = "";
$dbName = "chatroomdb";
$connection = mysqli_connect($serverName, $Username, $Password, $dbName);
if(!$connection){
  echo "Cannot connect to database - " . mysqli_connect_error();
}

$usersTable = "CREATE TABLE Users(
  UserID int NOT NULL PRIMARY KEY AUTO_INCREMENT,
  Username varchar(30) NOT NULL,
  Password varchar(255) NOT NULL
);";

$messagesTable = "CREATE TABLE Messages(
  MsgID int NOT NULL PRIMARY KEY AUTO_INCREMENT,
  Message varchar(300),
  Timestamp timestamp NOT NULL
);";

$usermsgTable = "CREATE TABLE UserMsg(
  UserID INT,
  MsgID INT,
  ImageID INT,
  PRIMARY KEY (UserID, MsgID, ImageID),
  FOREIGN KEY (UserID) REFERENCES users(UserID),
  FOREIGN KEY (MsgID) REFERENCES messages(MsgID),
  FOREIGN KEY (ImageID) REFERENCES profileimage(ImageID)
);";

$profileimageTable = "CREATE TABLE ProfileImage(
  ImageID INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
  UserID INT,
  Image varchar(),
  FOREIGN KEY (UserID) REFERENCES users(UserID)
);";

$onlineTable = "CREATE TABLE online (
      isOnline INT NOT NULL,
      UserID INT NOT NULL,
      Username VARCHAR(300) NOT NULL,
      FOREIGN KEY (UserID) REFERENCES users(UserID),
      PRIMARY KEY (isOnline, UserID)
);";

try{
  mysqli_query($connection, $usersTable);
  try{
    mysqli_query($connection, $messagesTable);
    try{
      mysqli_query($connection, $usermsgTable);
      try{
        mysqli_query($connection, $profileimageTable);
        try{
          mysqli_query($connection, $onlineTable);
        }
        catch(Exception $e){
          echo "Could not create Online table";
        }
      }
      catch(Exception $e){
        echo "Could not create ProfileImage table";
      }
    }
    catch(Exception $e){
      echo "Could not create UserMsg table";
    }
  }
  catch(Exception $e){
    echo "Could not create Messages table";
  }
}
catch(Exception $e){
  echo "Could not create Users table";
}
```

3.2. System overview

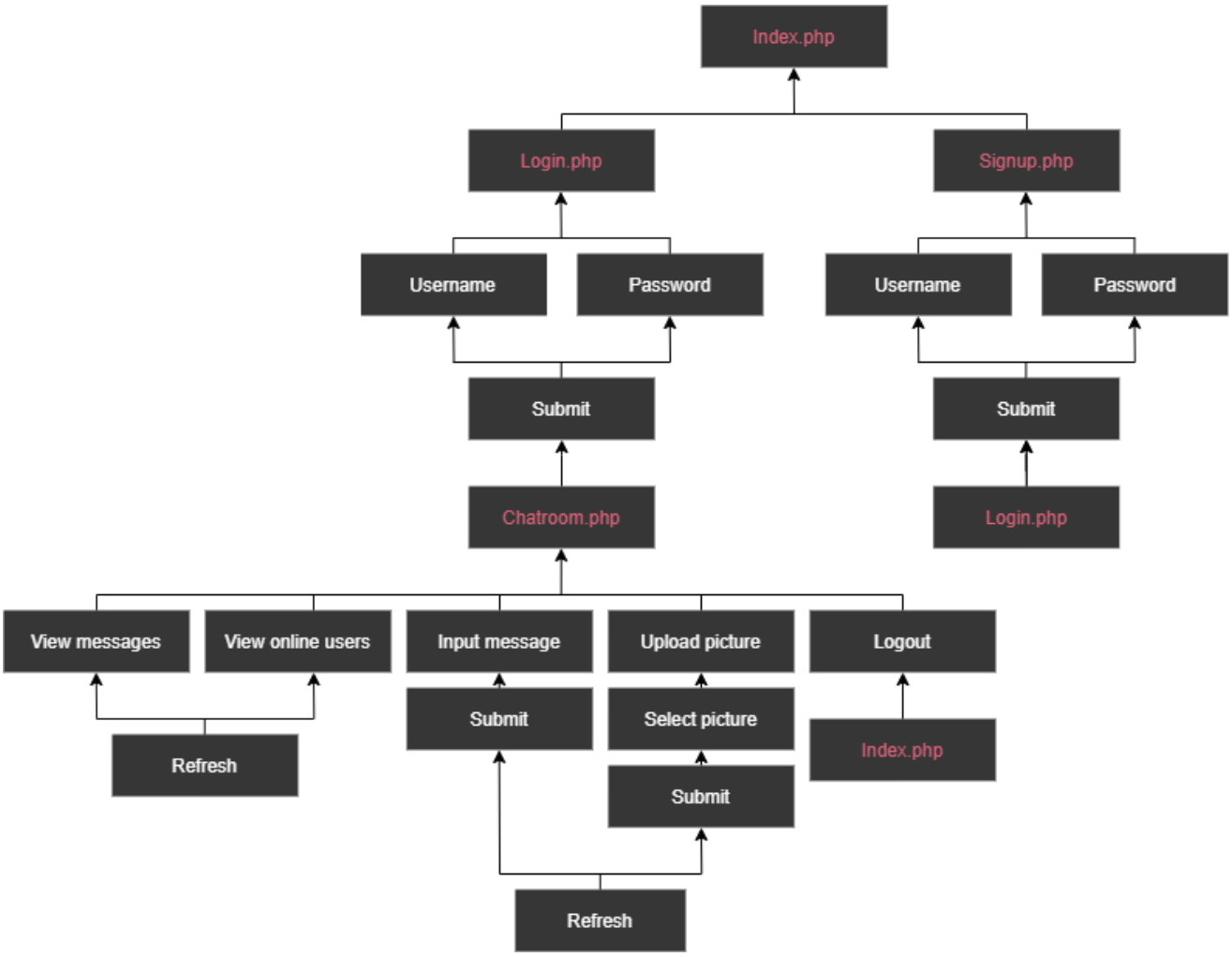


Figure 22 - System overview diagram

3.3. Evidence of built classes

The classes below are stored within the “classes.php” file which is to be included in other files so that instances can be created.

```php
<?php
/*=====================================================================
* classes.php *
File holds the classes that are to be used.
=====================================================================*/

class Users{
  // [PROPERTIES]
  public $Username;
  private $Password;

  // [CONSTRUCTOR]
  public function __construct($Username, $Password){
    $this->Username = $Username;
    $this->Password = $Password;
  }

  // [METHODS]
  function LoginValidity(){
    include ('dbh.php');

    // SQL
    $SQL_SelectPassword = mysqli_query($connection, "SELECT Password FROM users WHERE Username = '$this->Username'"); // Get
the inputted user's password (If real)
    $SQL_PasswordRow = mysqli_fetch_array($SQL_SelectPassword); // Row password
    $SQL_UpdateOnline = mysqli_query($connection, "UPDATE online SET isOnline = 1 WHERE Username = '$this->Username'"); //
Set online status to 1 to indicate they are online

    // If user passes all validation rules, log the user in and redirect them to the chatroom
    if(empty($this->Username) || empty($this->Password)){
      header('location: ../chatserver/login.php?error=emptyinput'); // ERROR - EMPTY FORM
      exit();
    }
    elseif(mysqli_num_rows($SQL_SelectPassword) > 0){
      if(password_verify($this->Password, $SQL_PasswordRow['Password'])){ // VERIFY PASSWORD
        try{
          $SQL_UpdateOnline;
          $_SESSION['Username'] = $this->Username;
          header("location: ../chatserver/chatroom.php"); // DIRECT USER TO CHATROOM
          $connection -> close();
          exit();
        }
        catch(Exception $e) {
          die("Error connecting with database - Online Status"); // ERROR - UPDATING ISONLINE
        }
      }
      else{
        header('location: ../chatserver/login.php?error=invaliddetails'); // ERROR - INCORRECT ACCOUNT INFO.
        exit();
      }
    }
    else{
      header('location: ../chatserver/login.php?error=inexistent'); // ERROR - ACCOUNT DOESN'T EXIST
      exit();
    }
  }
  function SignUpValidity(){
    include ("dbh.php");

    // VARIABLES
```

```
    $Regex = '/^\S*(?=\S*[a-z])(?=\S*[A-Z])(?=\S*[\d])\S*$/'; // At least one or more lowercase, uppercase and numeric
characters
    $passHash = password_hash($this->Password, PASSWORD_DEFAULT); // Hash the password inputted

    // SQL
    $SQL_CheckIfUsernameExists = mysqli_query($connection, "SELECT Username FROM users WHERE Username = '$this->Username'");
// Check if username is already taken
    $SQL_AddToUsers = mysqli_query($connection, "INSERT INTO users(Username, Password) VALUES ('$this->Username',
'$passHash')"); // Add the new user to the Users table
    $SQL_AddToOnline = mysqli_query($connection, "INSERT INTO online(UserID, Username) SELECT UserID, Username FROM Users
WHERE users.Username = ('$this->Username')"); // Add the new user to the Online table
    $SQL_AddToProfile = mysqli_query($connection, "INSERT INTO profileimage(UserID) SELECT UserID FROM Users WHERE
users.Username = ('$this->Username')"); // Add the new user to the profileimage table

    // If user passes all validation rules, sign up the user and redirect them to the login page
    if(empty($this->Username) || empty($this->Password)){
      return header("location: ../chatserver/signup.php?error=emptyinput"); // ERROR - EMPTY FORM
      exit();
    }
    elseif(mysqli_num_rows($SQL_CheckIfUsernameExists) > 0){
      return header("location: ../chatserver/signup.php?error=usernametaken"); // ERROR - USERNAME IS TAKEN
      exit();
    }
    elseif(strlen($this->Password) < 8 || !preg_match($Regex, $this->Password)){
      return header("location: ../chatserver/signup.php?error=invalidpassword"); // ERROR - PASSWORD ISN'T VALID
      exit();
    }
    else{
      try{
        $SQL_AddToUsers;
        try{
          $SQL_AddToOnline;
          try{
            $SQL_AddToProfile;
            header("location: login.php"); // DIRECT USER TO LOGIN PAGE
            $connection -> close();
            exit();
          }
          catch(Exception $e){
            die("Error connecting to database - Adding to Profile"); // ERROR - PROFILEIMAGE TABLE
          }
        }
        catch(Exception $e){
          die("Error connecting to database - Adding to Online"); // ERROR - ONLINE TABLE
        }
      }
      catch(Exception $e){
        die("Error connecting to database - Adding to Users"); // ERROR - USERS TABLE
      }
    }
  }
}

class LoggedUsers extends Users{
  // [PROPERTIES]
  public $UserID;

  // [CONSTRUCTOR]
  public function __construct($Username, $UserID){
    $this->Username = $Username;
    $this->UserID = $UserID;
  }

  // [METHODS]
  public function SendMessage($Message){
    include ('dbh.php');

    // SQL
    $SQL_ImageID = mysqli_query($newConnection, "SELECT ImageID FROM profileimage WHERE UserID = '$this->UserID'"); // Get
all users' image ID
    while($row = mysqli_fetch_array($SQL_ImageID)){
      $ImageID[] = $row['ImageID'];
    }
    $SQL_InsertMessage = mysqli_query($newConnection, "INSERT INTO messages(Message) VALUES ('$Message')"); // Insert the
message into the Messages table
    $SQL_MessageID = mysqli_insert_id($newConnection); // Fetch the recent message ID by that user
    $SQL_InsertUserMsg = mysqli_query($newConnection, "INSERT INTO usermsg(UserID, MsgID, ImageID) VALUES ('$this->UserID',
'$SQL_MessageID', '$ImageID[0]')"); // Insert values into usermsg table to establish relation

    // If user sends a message, pass the message through the appropriate tables to match the message with the correct user
      try{
        $SQL_InsertMessage;
        try{
          $SQL_MessageID;
          try{
            $SQL_InsertUserMsg;
            header("location: ../chatserver/chatroom.php"); // REFRESH CHATROOM
            exit();
          }
          catch(Exception $e){
            die("Error with database - Relation table"); // ERROR - RELATION TABLE NOT INSERTING
          }
        }
        catch(Exception $e){
           die("Error with database - Message ID"); // ERROR - CANNOT RETRIEVE MESSAGE ID
        }
      }
      catch(Exception $e){
        die("Error with database - Sending Message"); // ERROR - MESSAGES TABLE NOT INSERTING
      }
    }
```

```
  public function DisplayMessages(){
    include ('dbh.php');

    // SQL - Select all the messages alongside their respective user and their profile image for each new line in the chatlog
    try{
      return mysqli_query($newConnection, "SELECT users.Username, profileimage.Image, messages.Message, messages.Timestamp
      FROM users
      INNER JOIN usermsg ON users.UserID = usermsg.UserID
      INNER JOIN messages ON usermsg.MsgID = messages.MsgID
      INNER JOIN profileimage ON usermsg.ImageID = profileimage.ImageID
      ORDER BY messages.Timestamp");
    }
    catch(Exception $e){
      die("Error with database - Relation not executing"); // ERROR - RELATION TABLE ISN'T RELATING DATA
    }
  }

  public function SetDefaultPFP(){
    include ('dbh.php');

    // SQL
    $SQL_Default = mysqli_query($newConnection, "SELECT Image FROM profileimage WHERE profileimage.UserID = '$this-
>UserID'"); // Set a default profile picture if Image == null

    // Set a default profile picture when the user logs in and has a null value in the Image column
    while($Row = mysqli_fetch_array($SQL_Default)){
      try{
        if($Row['Image'] == null){
          mysqli_query($newConnection, "UPDATE profileimage SET Image = 'Default.png' WHERE UserID = '$this->UserID'");
        }
      }
      catch(Exception $e){
        die("Error setting default profile picture"); // ERROR - COULDN'T SET DEFAULT PFP
      }
    }
  }

  public function DisplayPFP(){
    include ('dbh.php');

    // Get user's profile image to be displayed
    return mysqli_query($newConnection, "SELECT Image FROM profileimage WHERE profileimage.UserID = '$this->UserID'");

  }

  public function UploadPFP(){
    include ('dbh.php');

    // VARIABLES
    $Location = "Images/".basename($_FILES['image']['name']); // File directory
    $Image = $_FILES['image']['name']; // Establish image name

    // SQL
    $SQL_UpdatePFP = mysqli_query($newConnection, "UPDATE profileimage SET Image = '$Image' WHERE UserID = '$this->UserID'");
// Update user's Image to the one they have uploaded

    // On a confirmed image file, update the user's profile picture by updating the Image column and storing the file into my
directory
    try{
      $SQL_UpdatePFP;
      try{
        move_uploaded_file($_FILES['image']['tmp_name'], $Location); // Try for whether the file has been directed
successfuly to my directory
        header("location: ../chatserver/chatroom.php"); // REFRESH CHATROOM
        exit();
      }
      catch(Exception $e){
        die("Error with directory - Moving file"); // ERROR - MOVING FILE TO MY DIRECTORY
      }
    }
    catch(Exception $e){
      die("Error with database - Updating Profile"); // ERROR - PROFILEIMAGE TABLE NOT UPDATING
    }
  }

  public function DisplayOnlineUsers(){
    include ('dbh.php');

    // Get all users who have an online status of 1
    return mysqli_query($newConnection, "SELECT Username FROM online WHERE isOnline = 1");
  }

  public function Logout(){
    include ('dbh.php');
    // End the users session and update them to appear offline on logout
    unset($_SESSION['Username']);
    session_destroy(); // Delete the user's session
    header("location: /chatserver/index.php");
    return mysqli_query($newConnection, "UPDATE online SET isOnline = 0 WHERE Username = '$this->Username'"); // Set online
status to 0 and redirect them to the login page
    exit();
  }
}
```

3.4. Evidence of complete code listing

3.4.1. Index page

3.4.1.1. index.php

```
<!--=====================================================================
* index.php *
Page only serves as an introduction, main function belongs to other files.
=====================================================================-->

<!DOCTYPE html>
<html lang="en" dir="ltr">
   <head>
      <meta charset="utf-8">
      <link rel="stylesheet" href="CSS/indexstyle.css" type="text/css">
      <script src="https://code.jquery.com/jquery-3.5.1.min.js"></script> <!-- Ajax script -->
      <title>chatroom</title>
   </head>
   <body>
      <div id="box">
         <h1>CHATROOM WEBSITE</h1>
         <!-- PAGE BUTTONS -->
         <p>
            <a href="signup.php" id="sign">Sign Up</a> <br>
            <a href="login.php" id="login">Login</a>
         </p>
      </div>
      <!-- Box animation -->
      <script type="text/javascript">
         $("#box").animate({top:'25%'}, 1000);
      </script>
   </body>
</html>
```

3.4.1.2. indexstyle.css

```
body{
  height: 100%;
  text-align: center;
  font-family: monospace;
  background-color: #b68973;
  margin: 0;
}

#box{
  position: fixed;
  background-color: #1e212d;
  box-shadow: 0px 0px 10px;
  opacity: 100%;
  width: 35%;
  top: 100%;
  height: 50%;
  left: 50%;
  transform: translateX(-50%);
  margin: 0;
}

h1{
  font-size: 50px;
  text-shadow: 0px, 0px 10px;
  line-height: 100px;
  color: #faf3e0;
}

#sign{
  position: absolute;
  top: 190px;
  left: 50%;
  transform: translateX(-50%);
  font-size: 30px;
  margin-bottom: 10px;
  background-color: #faf3e0;
  box-shadow: 0px 0px 5px black;
  width: 200px;
  height: 50px;
  line-height: 50px;
  border-radius: 50px;
  opacity: 100%;
}

#login{
  position: absolute;
  top: 290px;
  left: 50%;
  transform: translateX(-50%);
  font-size: 30px;
  margin-bottom: 10px;
  background-color: #faf3e0;
  box-shadow: 0px 0px 5px black;
```

```
  width: 200px;
  height: 50px;
  line-height: 50px;
  border-radius: 50px;
  opacity: 100%;
}

#sign:hover{
  background-color: #eabf9f;
}

#login:hover{
  background-color: #eabf9f;
}

a:link{
  color: #1e212d;
  text-decoration: none;
  font-weight: normal;
}

a:visited{
  color: #1e212d;
  text-decoration: none;
  font-weight: normal;
}
```

3.4.2. Form pages

3.4.2.1. signup.php

```
<?php
/*================================================================================
* signup.php *
On click of the signup button, the user will be directed here.
This file contains all the code relating to the account creation process.
User will be an instance of the class User, calling upon methods appropriate to that class and event.
The HTML error messages are in response to the validation rules that have been set
================================================================================*/

include 'classes.php';

// USER INSTANCE
$User = new Users(@trim($_POST['username']), @$_POST['password']);

// [METHOD CALL]
if(isset($_POST['submit'])){
  $User->SignUpValidity();
}
?>

<!-- HTML -->
<!DOCTYPE html>
<html lang="en" dir="ltr">
   <head>
      <meta charset="utf-8">
      <title>Sign up</title>
      <link rel="stylesheet" href="CSS/formstyle.css" type="text/css">
      <script src="https://code.jquery.com/jquery-3.5.1.min.js"></script>
   </head>
   <body>
      <div id="box">
         <h1>Sign Up</h1>
         <!-- SIGNUP FORM -->
         <form class="signup-form" action="<?=$_SERVER['PHP_SELF']?>" method="post" autocomplete="off">
            <p1>Enter a username:</p1>
            <br><input type="text" name="username">
            <br></br>
            <p1>Enter a password:</p1>
            <br><input type="password" name="password">
            <br></br>
            <button type="submit" name="submit">Sign Up</button>
            <br>
            <p2>Password: <br> * 8 <u>characters</u> or more<br> * One or more <u>uppercase</u> and <u>lowercase
character(s)</u><br> * One or more <u>number(s)</u></p2>
            <br>
            <br>
            <?php
               // ERROR MESSAGES
               if(isset($_GET["error"])){
                 if($_GET["error"] == "emptyinput"){
                   echo "<p3>Fields must not be left empty</p3>";
                 }
                 else if ($_GET["error"] == "usernametaken") {
                   echo "<p3>Username is taken</p3>";
                 }
                 else if ($_GET["error"] == "invalidpassword") {
                   echo "<p3>Password must be at least 8 characters and have one uppercase and lowercase letter and a
number</p3>";
                 }
               }
               ?>
         </form>
      </div>
      <!-- Box animation -->
      <script type="text/javascript">
         $("#box").animate({top:'25%'}, 500);
      </script>
```

```
   </body>
</html>
```

3.4.2.2. login.php

```
<?php
/*================================================================================
* login.php *
On click of the login button or after a successful signup, the user will be directed here.
This file contains all the code relating to the account creation process.
User will be an instance of the class User, calling upon methods appropriate to that class and event.
The HTML error messages are in response to the validation rules that have been set
Session Start is used to temporarily hold the user's account as a session
================================================================================*/

include 'classes.php';
session_start();

// USER INSTANCE
$User = new Users(@$_POST['username'], @$_POST['password']);

// [METHOD CALL]
if(isset($_POST['submit'])){
  $User->LoginValidity();
}
?>

<!-- HTML -->
<!DOCTYPE html>
<html lang="en" dir="ltr">
   <head>
      <meta charset="utf-8">
      <title>Login</title>
      <link rel="stylesheet" href="CSS/formstyle.css" type="text/css">
      <script src="https://code.jquery.com/jquery-3.5.1.min.js"></script>
   </head>
   <body>
      <div id="box">
         <h1>LOGIN</h1>
         <!-- LOGIN FORM -->
         <form class="login-form" action="<?=$_SERVER['PHP_SELF']?>" method="post" autocomplete="off">
            <p1>Enter username:</p1>
            <br><input type="text" name="username" value="">
            <br></br>
            <p1>Enter password:</p1>
            <br><input type="password" name="password" value="">
            <br></br>
            <button type="submit" name="submit">Login</button>
            <br></br>
            <br>
            <p2>Enter existing account details</p2>
            <br>
            <br>
            <?php
               // ERROR MESSAGES
               if(isset($_GET["error"])){
                 if($_GET["error"] == "emptyinput"){
                   echo "<p3>Fields must not be left empty</p3>";
                 }
                 else if ($_GET["error"] == "invaliddetails") {
                   echo "<p3>Incorrect username or password</p3>";
                 }
               }
             ?>
         </form>
      </div>
      <!-- Box animation -->
      <script type="text/javascript">
         $("#box").animate({top:'25%'}, 500);
      </script>
   </body>
</html>
```

3.4.2.3. formstyle.css

```
body{
  height: 100%;
  text-align: center;
  font-family: monospace;
  background-color: #b68973;
  margin: 0;
}

#box{
  position: fixed;
  background-color: #1e212d;
  box-shadow: 0px 0px 10px;
  opacity: 100%;
  width: 35%;
  top: 100%;
  height: 60%;
  left: 50%;
  transform: translateX(-50%);
  margin: 0;
}
```

```
h1{
  font-size: 50px;
  text-shadow: 0px, 0px 10px;
  line-height: 50px;
  color: #faf3e0;
}

p1{
  transform: translateX(-50%);
  font-size: 17px;
  margin-bottom: 10px;
  line-height: 50px;
  border-radius: 50px;
  opacity: 100%;
  color: #faf3e0;
  font-weight: bold;
}

button{
  margin-top: 30px;
  margin-bottom: 30px;
  padding: 8px;
  width: 100px;
  background-color: #faf3e0;
  color: #1e212d;
  border: 0px;
  border-radius: 50px;
  font-weight: bold;
}

button:hover{
  background-color: #eabf9f;
}
p2{
  color: white;
  font-style: oblique;
}

p3{
  color: red;
}
```

3.4.4. Chatroom page

3.4.4.1. chatroom.php

```
<?php
/*================================================================================
* chatroom.php *
On successful login the user will be directed here.
This file contains all the code relating to the bulk of the project, the chatroom itself.
User will now be an instance of the class LoggedUser, calling upon methods appropriate to that class and event.
!! For the actual output of Messages and Online Users, see the HTML below !!
================================================================================*/

session_start();
include 'dbh.php';
include 'classes.php';

// VARIABLES
$LoggedUser = $_SESSION['Username'];

// SQL
$SQL_UserID = mysqli_query($newConnection, "SELECT UserID FROM users WHERE Username = '$LoggedUser'");
while ($Row = mysqli_fetch_array($SQL_UserID)) {
  $userID[] = $Row['UserID'];
}

// USER INSTANCE
$User = new LoggedUsers($LoggedUser, $userID[0]);

// [METHOD CALL]
// SEND MESSAGE
$Message = @$_POST["chatbox"];
if(isset($_POST['submit'])){
  if(empty($Message)){
    header("location: /chatserver/chatroom.php"); // Empty message was sent, do nothing
    exit();
  }
  else{
    $User->SendMessage($Message);
  }
}

// DISPLAY MESSAGES
$Messages = $User->DisplayMessages();

// DEFAULT PFP
$User->SetDefaultPFP();

// DISPLAY PFP
$SidePFP = $User->DisplayPFP();

// UPLOAD PFP
if(isset($_POST['profilepic'])){
  header("location: /chatserver/chatroom.php?action=changeprofilepicture");
```

```
  exit();
}
if(isset($_POST['confirm'])){
  $User->UploadPFP();
}

// ONLINE USERS
$List = $User->DisplayOnlineUsers();

// LOGOUT
if(isset($_POST['logout'])){
  $User->Logout();
}
?>

<!-- HTML -->
<!DOCTYPE html>
<html lang="en" dir="ltr">
   <head>
      <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
      <link rel="stylesheet" href="CSS/chatroomstyle.css" type="text/css">
      <link rel="shortcut icon" href="#">
      <title>Chatroom</title>
   </head>
   <body>
      <!-- CHATLOG -->
      <div id="log">
         <p id="messages">
            <?php
            // Fetch messages and output them for each line
               while($MessageRow = mysqli_fetch_array($Messages)){
                 echo "<img id='iconPFP' src='Images/".$MessageRow['Image']."'>";
                 echo "<p id='time'>[{$MessageRow['Timestamp']}]";
                 echo "<p id='name'>{$MessageRow['Username']}<p>";
                 echo "<p id='message'>{$MessageRow['Message']}</p>";
               }
            ?>
         </p>
      </div>
      <!-- REFRESH-->
      <script type="text/javascript">


         // Set refresh timer (Chatlog)
         var Refresh = setInterval('window.location.reload()', 7000);
         $('#log').on('refresh', function () {
             if ($(this).val().length == '') {
                         clearInterval(Refresh);
                         Refresh = setInterval('window.location.reload()', 7000);
             } else {
                         clearInterval(Refresh);
             }
         });
         // Scroll on refresh
         document.addEventListener('DOMContentLoaded', function () {
             var messages = document.querySelector('#log');
             messages.scrollTop = messages.scrollHeight;
         });
         // Set refresh timer (Online)
         RefreshList();
         function RefreshList() {
           setInterval(function() {
             $('#onlinelist).load('chatroom.php #onlinelist);
           }, 7000);
         }
      </script>
      <!-- USER TAB -->
      <div id="usertab">
         <p id="usertitle">User tab</p>
         <div>
            <form action="<?=$_SERVER['PHP_SELF']?>" method="post">
               <button id="profilepic" type="submit" name="profilepic">Change profile picture</button>
            </form>
            <form id="change" action="<?=$_SERVER['PHP_SELF']?>" method="post" enctype="multipart/form-data">
<?php
                  // Execute method if user submitted an image
                  if(isset($_GET['action'])){
                    if($_GET['action'] == "changeprofilepicture"){
                      echo "<input type='file' name='image' accept='image/*'> <button id='confirm' type='submit'
name='confirm'>Confirm</button>";
                    }
                  }
                ?>
</form>
            <?php
               // Get the user's image file to be displayed as a preview
               while($imageRow = mysqli_fetch_array($SidePFP)){
                 echo "<div>";
                 echo "<img id='sidePFP' src='Images/".$imageRow['Image']."'";
                 echo "</div>";
               }
               if(isset($_GET["error"])){
                 if($_GET["error"] == "invalidpfp"){
                   echo "<p id='pfperror'>Image must be either a .png or .jpg</p>";
                 }
             }

                ?>
            <p id="sidename"><?=$User->Username?></p>
         </div>
      </div>
      <!-- ONLINE TAB -->
```

```
      <div id="onlinelist">
         <p id="onlinetitle">Online users</p>
         <?php
            // Output a list of the online users
            while($list = mysqli_fetch_array($List)){
              echo "<p id='onlineuser'>›› {$list['Username']}</p>";
            }
             ?>
         <form action="<?=$_SERVER['PHP_SELF']?>" method="post">
            <button id="logout" type="submit" name="logout">Logout</button>
         </form>
      </div>
      <!-- MESSAGE INPUT BOX -->
      <div id="chatboxoutline">
         <form action="<?=$_SERVER['PHP_SELF']?>" method="post">
            <textarea id="text" rows="8" cols="80" method="post" name="chatbox"></textarea>
            <p id="notice">Make sure to click "Logout" when leaving!!</p> <!-- !! Inform user on logging out to set online
status to 0 -->
            <button id="submit" type="submit" name="submit">Send ››</button>
         </form>
      </div>
   </body>
</html>
```

3.4.4.2. chatroom.css

```
body{
  height: 100%;
  font-family: monospace;
  background-color: #eabf9f;
  margin: 0;
}

#log{
  position: fixed;
  left: 15%;
  background-color: #eabf9f;
  padding: 1px 15px;
  width: 68%;
  height: 90%;
  overflow-y: scroll;
  overflow-wrap: break-word;
}

#onlinelist{
  position: fixed;
  height: 100%;
  left: 85%;
  width: 15%;
  background-color: #1e212d;
  overflow-wrap: break-word;
  padding-bottom: 83%;
}

#onlinetitle{
  text-decoration: underline;
  position: fixed;
  background-color: #010205;
  width: 15%;
  height: 5%;
  font-size: 25px;
  color: white;
  text-align: center;
  line-height: 50px;
  margin-top: 0;
}

#onlineuser{
  position: relative;
  top: 7%;
  padding-left: 10%;
  line-height: 50%;
  color: #faf3e0;
  font-size: 20px;
}

/* MESSAGES */

#iconPFP{
  width: 50px;
  height: 50px;
  border-radius: 50px;
}

#name{
  position: relative;
  font-size: 20px;
  color: #1e212d;
  top: -90px;
  left: 65px;
  margin-right: 50px;
}

#time{
  position: relative;
  font-size: 10px;
```

```
  color: #070912;
  top: -35px;
  left: 65px;
  opacity: 50%;
  margin-right: 50px;
}

#message{
  position: relative;
  font-size: 17px;
  color: #faf3e0;
  top: -75px;
  left: 5px;
  background-color: #b68973;
  border-radius: 10px;
  padding: 10px;
  margin-right: 50px;
  margin-bottom: -50px;
}

#chatboxoutline{
  position: fixed;
  border-radius: 50px;
  resize: none;
  width: 69%;
  height: 5%;
  top: 93.5%;
  left: 15.5%;
  background-color: #1e212d;
}

textarea{
  position: fixed;
  resize: none;
  width: 62%;
  height: 3.5%;
  top: 94%;
  left: 16%;
  background-color: #1e212d;
  color: white;
  border: 0px;
  vertical-align: text-top;
  text-align: left;
  font-size: 25px;
}

input:focus, textarea:focus, select:focus{
  outline: none;
}

#notice{
  position: fixed;
  top: 90%;
  left: 15.6%;
  color: darkred;
}

::-webkit-scrollbar{
  width: 10px;
}

::-webkit-scrollbar-track {
  opacity: 0%;
}

::-webkit-scrollbar-thumb {
  background-color: #1e212d;
  box-shadow: 0px;
  border-radius: 50px;
  border: 0px;
}

#usertab{
  position: fixed;
  height: 100%;
  width: 15%;
  background-color: #1e212d;
  overflow-wrap: break-word;
  margin: 0;
}

#usertitle{
  text-decoration: underline;
  position: fixed;
  background-color: #010205;
  width: 15%;
  height: 5%;
  font-size: 25px;
  color: white;
  text-align: center;
  line-height: 50px;
  margin-top: 0;
}

#sidePFP{
  position: fixed;
  top: 10%;
  left: 2.5%;
  width: 190px;
  height: 190px;
  border-radius: 100px;
}
```

```css
#pfperror{
  position: fixed;
  color: red;
  top: 50%;
  left: 1%;
}

#sidename{
  position: fixed;
  width: 15%;
  font-size: 50px;
  text-align: center;
  top: 30%;
  color: #faf3e0;
}

/* BUTTONS */

#submit{
  position: fixed;
  left: 79%;
  top: 94.1%;
  width: 5%;
  height: 3.8%;
  background-color: #faf3e0;
  border-radius: 50px;
  border: none;
  color: #b68973;
}

#submit:hover{
  background-color: #b68973;
  color: #faf3e0;
}

#logout{
  position: fixed;
  left: 87.5%;
  top: 90%;
  width: 10%;
  height: 5%;
  color: #4a000e;
  background-color: #e8002b;
  font-size: 20px;
  border: 0px;
  border-radius: 50px;
}

#logout:hover{
  color: #e8002b;
  background-color: #4a000e;
}

#profilepic{
  position: fixed;
  left: 2.5%;
  top: 90%;
  width: 10%;
  height: 5%;
  color: #1e212d;
  background-color: #faf3e0;
  font-size: 15px;
  border: 0px;
  border-radius: 50px;
}

#profilepic:hover{
  color: #1e212d;
  background-color: #eabf9f;
}

#change{
  position: fixed;
  left: 2.5%;
  top: 69%;
  width: 20px;
}
```

3.4.5. Backend files

3.4.5.1. dbh.php

```php
<?php
/*=============================================================================
* dbh.php *
This file contains the main variables Connection and NewConnection which are connect queries that will be used in other files
for query execution.
Database queries are also included here which will create the tables.
=============================================================================*/

// [CONNECTIONS]
// Initial user access
$serverName = "localhost";
$Username = "root";
$Password = "";
$dbName = "chatroomdb";
```

```
$connection = mysqli_connect($serverName, $Username, $Password, $dbName); // Allows for execution of queries pre-chatroom
if(!$connection){
  echo "Cannot connect to database - " . mysqli_connect_error();
}

// Logged user access (Read/Write)
$loggedUser = "siteUser";
$newPassword = "iFDqje7e9St69L02"; // Randomly generated password to access the SQL account
$newConnection = mysqli_connect($serverName, $loggedUser, $newPassword, $dbName); // Allows for execution of queries post-
chatroom
if(!$newConnection){
  echo "Cannot connect to database - " . mysqli_connect_error();
}

// [TABLES]
// Creation
$usersTable = "CREATE TABLE Users(
  UserID int NOT NULL PRIMARY KEY AUTO_INCREMENT,
  Username varchar(30) NOT NULL,
  Password varchar(255) NOT NULL
);";

$messagesTable = "CREATE TABLE Messages(
  MsgID int NOT NULL PRIMARY KEY AUTO_INCREMENT,
  Message varchar(300),
  Timestamp timestamp NOT NULL
);";

$usermsgTable = "CREATE TABLE UserMsg(
  UserID INT,
  MsgID INT,
  ImageID INT,
  PRIMARY KEY (UserID, MsgID, ImageID),
  FOREIGN KEY (UserID) REFERENCES users(UserID),
  FOREIGN KEY (MsgID) REFERENCES messages(MsgID),
  FOREIGN KEY (ImageID) REFERENCES profileimage(ImageID)
);";

$profileimageTable = "CREATE TABLE ProfileImage(
  ImageID INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
  UserID INT,
  Image varchar(),
  FOREIGN KEY (UserID) REFERENCES users(UserID)
);";

$onlineTable = "CREATE TABLE online (
      isOnline INT NOT NULL,
      UserID INT NOT NULL,
      Username VARCHAR(300) NOT NULL,
      FOREIGN KEY (UserID) REFERENCES users(UserID),
      PRIMARY KEY (isOnline, UserID)
);";

// Execution
try{
  mysqli_query($connection, $usersTable);
  try{
    mysqli_query($connection, $messagesTable);
    try{
      mysqli_query($connection, $usermsgTable);
      try{
        mysqli_query($connection, $profileimageTable);
        try{
          mysqli_query($connection, $onlineTable);
        }
        catch(Exception $e){
          echo "Could not create Online table";
        }
      }
      catch(Exception $e){
        echo "Could not create ProfileImage table";
      }
    }
    catch(Exception $e){
      echo "Could not create UserMsg table";
    }
  }
  catch(Exception $e){
    echo "Could not create Messages table";
  }
}
catch(Exception $e){
  echo "Could not create Users table";
}
?>
```

3.4.5.2. classes.php

```
<?php
/*================================================================================
* classes.php *
File holds the classes that are to be used.
================================================================================*/

class Users{
  // [PROPERTIES]
  public $Username;
  private $Password;
```

```
  // [CONSTRUCTOR]
  public function __construct($Username, $Password){
    $this->Username = $Username;
    $this->Password = $Password;
  }

  // [METHODS]
  function LoginValidity(){
    include ('dbh.php');

    // SQL
    $SQL_SelectPassword = mysqli_query($connection, "SELECT Password FROM users WHERE Username = '$this->Username'"); // Get
the inputted user's password (If real)
    $SQL_PasswordRow = mysqli_fetch_array($SQL_SelectPassword); // Row password
    $SQL_UpdateOnline = mysqli_query($connection, "UPDATE online SET isOnline = 1 WHERE Username = '$this->Username'"); //
Set online status to 1 to indicate they are online

    // If user passes all validation rules, log the user in and redirect them to the chatroom
    if(empty($this->Username) || empty($this->Password)){
      header('location: ../chatserver/login.php?error=emptyinput'); // ERROR - EMPTY FORM
      exit();
    }
    elseif(mysqli_num_rows($SQL_SelectPassword) > 0){
      if(password_verify($this->Password, $SQL_PasswordRow['Password'])){ // VERIFY PASSWORD
        try{
          $SQL_UpdateOnline;
          $_SESSION['Username'] = $this->Username;
          header("location: ../chatserver/chatroom.php"); // DIRECT USER TO CHATROOM
          $connection -> close();
          exit();
        }
        catch(Exception $e) {
          die("Error connecting with database - Online Status"); // ERROR - UPDATING ISONLINE
        }
      }
      else{
        header('location: ../chatserver/login.php?error=invaliddetails'); // ERROR - INCORRECT ACCOUNT INFO
        exit();
      }
else{
        header('location: ../chatserver/login.php?error=invaliddetails'); // ERROR - IF USERNAME DOES NOT EXIST (SAME ERROR)
        exit();
      }
    }
  }
  function SignUpValidity(){
    include ("dbh.php");

    // VARIABLES
    $Regex = '/^\S*(?=\S*[a-z])(?=\S*[A-Z])(?=\S*[\d])\S*$/'; // At one or more lowercase, uppercase and numeric character
    $passHash = password_hash($this->Password, PASSWORD_DEFAULT); // Hash the password inputted

    // SQL
    $SQL_CheckIfUsernameExists = mysqli_query($connection, "SELECT Username FROM users WHERE Username = '$this->Username'");
// Check if username is already taken
    $SQL_AddToUsers = mysqli_query($connection, "INSERT INTO users(Username, Password) VALUES ('$this->Username',
'$passHash')"); // Add the new user to the Users table
    $SQL_AddToOnline = mysqli_query($connection, "INSERT INTO online(UserID, Username) SELECT UserID, Username FROM Users
WHERE users.Username = ('$this->Username')"); // Add the new user to the Online table
    $SQL_AddToProfile = mysqli_query($connection, "INSERT INTO profileimage(UserID) SELECT UserID FROM Users WHERE
users.Username = ('$this->Username')"); // Add the new user to the profileimage table

    // If user passes all validation rules, sign up the user and redirect them to the login page
    if(empty($this->Username) || empty($this->Password)){
      header("location: ../chatserver/signup.php?error=emptyinput"); // ERROR - EMPTY FORM
      exit();
    }
    elseif(mysqli_num_rows($SQL_CheckIfUsernameExists) > 0){
      return header("location: ../chatserver/signup.php?error=usernametaken"); // ERROR - USERNAME IS TAKEN
      exit();
    }
    elseif(strlen($this->Password) < 8 || !preg_match($Regex, $this->Password)){
      return header("location: ../chatserver/signup.php?error=invalidpassword"); // ERROR - PASSWORD ISN'T VALID
      exit();
    }
    else{
      try{
        $SQL_AddToUsers;
        try{
          $SQL_AddToOnline;
          try{
            $SQL_AddToProfile;
            header("location: login.php"); // DIRECT USER TO LOGIN PAGE
            $connection -> close();
            exit();
          }
          catch(Exception $e){
            die("Error connecting to database - Adding to Profile"); // ERROR - PROFILEIMAGE TABLE
          }
        }
        catch(Exception $e){
          die("Error connecting to database - Adding to Online"); // ERROR - ONLINE TABLE
        }
      }
      catch(Exception $e){
        die("Error connecting to database - Adding to Users"); // ERROR - USERS TABLE
      }
    }
  }
}

class LoggedUsers extends Users{
```

```
  // [PROPERTIES]
  public $UserID;

  // [CONSTRUCTOR]
  public function __construct($Username, $UserID){
    $this->Username = $Username;
    $this->UserID = $UserID;
  }

  // [METHODS]
  public function SendMessage($Message){
    include ('dbh.php');

    // SQL
    $SQL_ImageID = mysqli_query($newConnection, "SELECT ImageID FROM profileimage WHERE UserID = '$this->UserID'"); // Get
all users' image ID
    while($row = mysqli_fetch_array($SQL_ImageID)){
      $ImageID[] = $row['ImageID'];
    }
    $SQL_InsertMessage = mysqli_query($newConnection, "INSERT INTO messages(Message) VALUES ('$Message')"); // Insert the
message into the Messages table
    $SQL_MessageID = mysqli_insert_id($newConnection); // Fetch the recent message ID by that user
    $SQL_InsertUserMsg = mysqli_query($newConnection, "INSERT INTO usermsg(UserID, MsgID, ImageID) VALUES ('$this->UserID',
'$SQL_MessageID', '$ImageID[0]')"); // Insert values into usermsg table to establish relation

    // If user sends a message, pass the message through the appropriate tables to match the message with the correct user
    if(empty($Message)){
      header("location: /chatserver/chatroom.php"); // Empty message was sent, do nothing
      exit();
    }
    else{
      try{
        $SQL_InsertMessage;
        try{
          $SQL_MessageID;
          try{
            $SQL_InsertUserMsg;
            header("location: ../chatserver/chatroom.php"); // REFRESH CHATROOM
            exit();
          }
          catch(Exception $e){
            die("Error with database - Relation table"); // ERROR - RELATION TABLE NOT INSERTING
          }
        }
        catch(Exception $e){
           die("Error with database - Message ID"); // ERROR - CANNOT RETRIEVE MESSAGE ID
        }
      }
      catch(Exception $e){
        die("Error with database - Sending Message"); // ERROR - MESSAGES TABLE NOT INSERTING
      }
    }
  }

  public function DisplayMessages(){
    include ('dbh.php');

    // SQL - Select all the messages alongside their respective user and their profile image for each new line in the chatlog
    try{
      return mysqli_query($newConnection, "SELECT users.Username, profileimage.Image, messages.Message, messages.Timestamp
      FROM users
      INNER JOIN usermsg ON users.UserID = usermsg.UserID
      INNER JOIN messages ON usermsg.MsgID = messages.MsgID
      INNER JOIN profileimage ON usermsg.ImageID = profileimage.ImageID
      ORDER BY messages.Timestamp");
    }
    catch(Exception $e){
      die("Error with database - Relation not executing"); // ERROR - RELATION TABLE ISN'T RELATING DATA
    }
  }

  public function SetDefaultPFP(){
    include ('dbh.php');

    // SQL
    $SQL_Default = mysqli_query($newConnection, "SELECT Image FROM profileimage WHERE profileimage.UserID = '$this-
>UserID'"); // Set a default profile picture if Image == null

    // Set a default profile picture when the user logs in and has a null value in the Image column
    while($Row = mysqli_fetch_array($SQL_Default)){
      try{
        if($Row['Image'] == null){
          mysqli_query($newConnection, "UPDATE profileimage SET Image = 'Default.png' WHERE UserID = '$this->UserID'");
        }
      }
      catch(Exception $e){
        die("Error setting default profile picture"); // ERROR - COULDN'T SET DEFAULT PFP
      }
    }
  }

  public function DisplayPFP(){
    include ('dbh.php');

    // Get user's profile image to be displayed
    return mysqli_query($newConnection, "SELECT Image FROM profileimage WHERE profileimage.UserID = '$this->UserID'");

  }

  public function UploadPFP(){
    include ('dbh.php');
```

```
    // VARIABLES
    $Location = "Images/".basename($_FILES['image']['name']); // File directory
    $Image = $_FILES['image']['name']; // Establish image name

    // SQL
    $SQL_UpdatePFP = mysqli_query($newConnection, "UPDATE profileimage SET Image = '$Image' WHERE UserID = '$this->UserID'");
// Update user's Image to the one they have uploaded

    // On a confirmed image file, update the user's profile picture by updating the Image column and storing the file into my
directory
    if($_FILES['image']['type'] == "image/jpeg" || $_FILES['image']['type'] == "image/png"){
      try{
        $SQL_UpdatePFP;
        try{
          move_uploaded_file($_FILES['image']['tmp_name'], $Location); // Try for whether the file has been directed
successfuly to my directory
          header("location: ../chatserver/chatroom.php"); // REFRESH CHATROOM
          exit();
        }
        catch(Exception $e){
          die("Error with directory - Moving file"); // ERROR - MOVING FILE TO MY DIRECTORY
        }
      }
      catch(Exception $e){
        die("Error with database - Updating Profile"); // ERROR - PROFILEIMAGE TABLE NOT UPDATING
      }
    }
    else{
      header("location: ../chatserver/chatroom.php?error=invalidpfp");
      mysqli_query($newConnection, "UPDATE profileimage SET Image = 'Default.png' WHERE UserID = '$this->UserID'");
      exit();
    }
  }

  public function DisplayOnlineUsers(){
    include ('dbh.php');

    // Get all users who have an online status of 1
    return mysqli_query($newConnection, "SELECT Username FROM online WHERE isOnline = 1");
  }

  public function Logout(){
    include ('dbh.php');

    // End the users session and update them to appear offline on logout
    unset($_SESSION['Username']);
    session_destroy(); // Delete the user's session
    header("location: /chatserver/index.php");
    return mysqli_query($newConnection, "UPDATE online SET isOnline = 0 WHERE Username = '$this->Username'"); // Set online
status to 0 and redirect them to the login page
    exit();
  }
}
```

3.5. Evidence of interface

3.5.1. Index page

Figure 23 - index.php

3.5.2. Sign up page

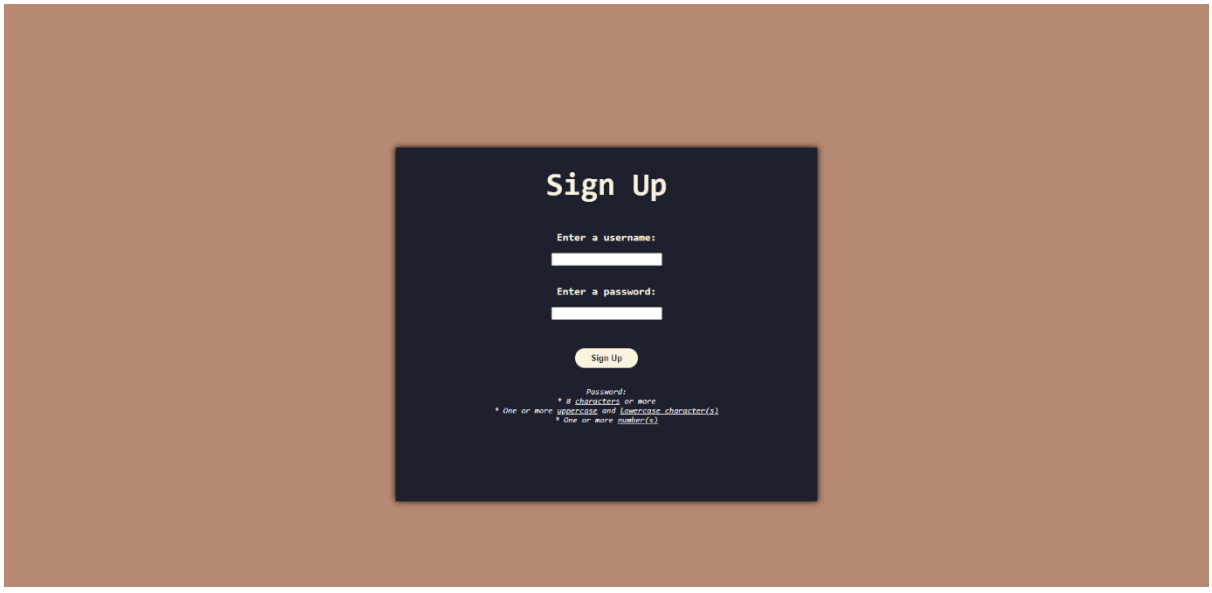


Figure 24 - signup.php

3.5.3. Login page

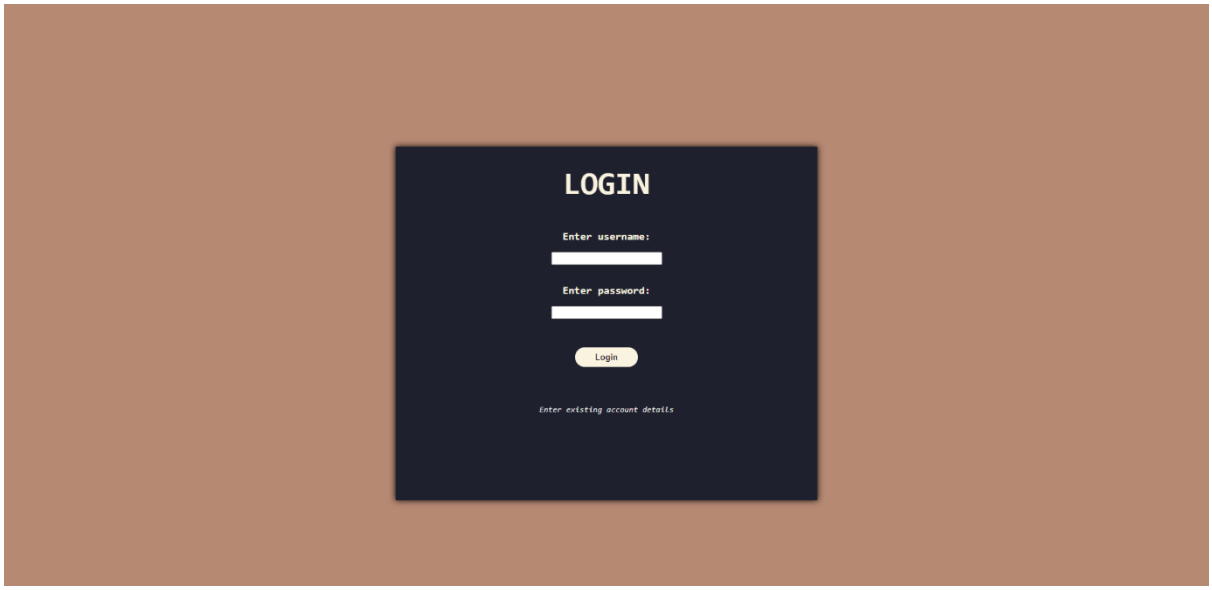


Figure 25 - login.php

3.5.4. Chatroom page

Figure 26 - chatroom.php

3.6. Evidence of procedures and variables

3.6.1. General procedures and variables table

Procedures	Explanation	Variables
echo()	Outputs string	**String** (Parameter)
trim()	Removes unnecessary whitespace in a string	**Username** (Parameter)
isset()	Checks if user presses a button in HTML form	**Submit** (Parameter)
session_start()	Used to transfer user's data into the chatroom and create a temporary session	
unset()	Removes user's data from session	**Username** (Parameter)
session_destroy()	Deletes user's session	
mysqli_query()	Allows for SQL queries to be executed by passing database and query as parameters	**Connection** (Parameter) **newConnection** (Parameter) **SQL_[QueryName]** (Parameter)
mysqli_fetch_array()	Outputs a row which is stored as an array	**SQL_[QueryName]** (Parameter) **Row** **UserID[]** **messageRow[]** **imageRow[]** **List[]**
mysqli_connect()	Establishes connection between database and program for queries to be executed	**serverName** (Parameter) **Username** (Parameter) **Password** (Parameter) **dbName** (Parameter) **loggedUser** (Parameter) **newPassword** (Parameter) **Connection** **newConnection**
header()	Redirects user to specific URL which is the parameter	**String** (Parameter)
setInterval()	Sets a specific interval of time that is to be iterated once met	**Time** (Parameter)
clearInterval()	Clears timer set by setInterval()	**Time** (Parameter)
addEventListener()	Clears timer set by setInterval()	**Time** (Parameter)
load()	Loads an element	**Element** (Parameter)
animate()	Animates an element	**Element** (Parameter) **Time** (Parameter) **TweenType** (Parameter)

Figure 27 - General procedures and variables table

3.6.2. Class procedures and variables table

Procedures	Explanation	Variables
SignUpValidity()	Validates the information entered by the user during sign-up process	Username Password Regex passHash SQL_CheckIfUsernameExists SQL_AddToUsers SQL_AddToOnline SQL_AddToProfile
LoginValidity()	Validates the information entered by the user during login process	Username Password SQL_SelectPassword SQL_PasswordRow SQL_UpdateOnline
SendMessage()	Allows user to send message	Message (Parameter) SQL_ImageID UserID row ImageID[] SQL_InsertMessage SQL_MessageID SQL_InsertUserMsg
DisplayMessages()	Allows user to view messages	Messages
SetDefaultPFP()	Sets a default profile picture if user does not have one	SQL_Default UserID Row
DisplayPFP()	Allows user to see a preview of their profile picture	sidePFP UserID
UploadPFP()	Allows user to upload a custom profile picture	Location Image SQL_UpdatePFP Image UserID
DisplayOnlineUsers()	Allows user to view all online users	List
Logout()	Allows user to log out and remove their session	

Figure 28 - Class procedures and variables table

END OF TECHNICAL SOLUTION

4. Testing

In this section of the document, I will be conducting various testing methods on my program to assess its functionality and whether it has met the original objectives that I have laid out.

In the case that users are required to input data of their own accord e.g. textual data, this information will be tested on a typical, erroneous and extreme scale. Below is a key representing each of the three:

Ty - Typical (Valid data) **Er** – Erroneous (Invalid data) **Ex** – Extreme (Boundary data)

4.1. Black-box testing – Assessing external functionality

4.1.1. Index page buttons

Test no.	1
Test purpose	This test will assess whether the buttons on the index page redirect the user accordingly
Test description	Both the sign-up and login buttons will be clicked on
Test data	Left click via mouse input on the login and sign-up buttons
Expected result	The user should be redirected to the appropriate page depending on the button that was pressed – /signup.php for sign-up button and /login.php for login button
Actual result	See figure 29

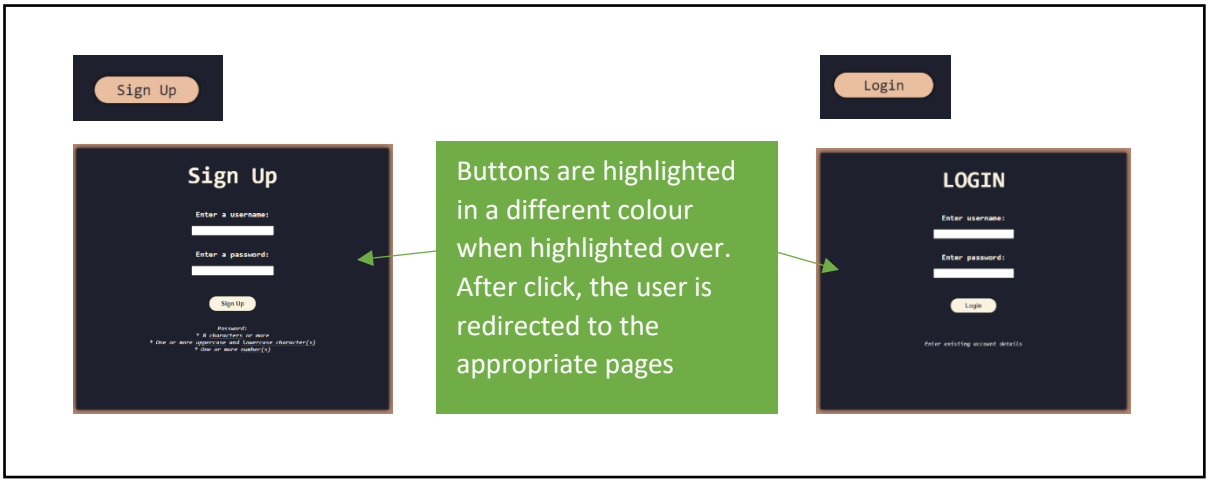


Figure 29

4.1.2. Sign-up validation

Test no.	2
Test purpose	For the user to create an account that is secure and unique to them
Test description	When the sign-up page is presented, different information will be entered to test the validation rules of the sign-up process
Test data	**Ty** Username: Test1, Password: ValidPa55word **Er** Username: Test1, Password: valid Username: Test2, Password: valid Username: Test2, Password: Valid Username: Test2, Password: Valid2 Username: , Password: **Ex** Username: Test3, Password: ValidPa Username: Test3, Password: ValidPa5 Username: Test3, Password: ValidPa55
Expected result	**Ty** The information entered should be valid and so the user should be redirected to login page on completion **Er** In the first case, an error message should appear stating that the “Username is taken” For the remaining cases except the last, an error message should appear stating that the “Password must be at least characters and have

	one uppercase and lowercase letter and a number” For the last case, an error message should appear stating that “Fields should not be empty” **Ex** In the first case, the password should be invalid as it is just outside the character limit The remaining two cases should allow the user to sign-up as the username is unique and the password is just within the character boundary count
Actual result	**Ty** See figure 30 **Er** See figure 31 See figure 32 See figure 33 **Ex** See figure 34 See figure 35

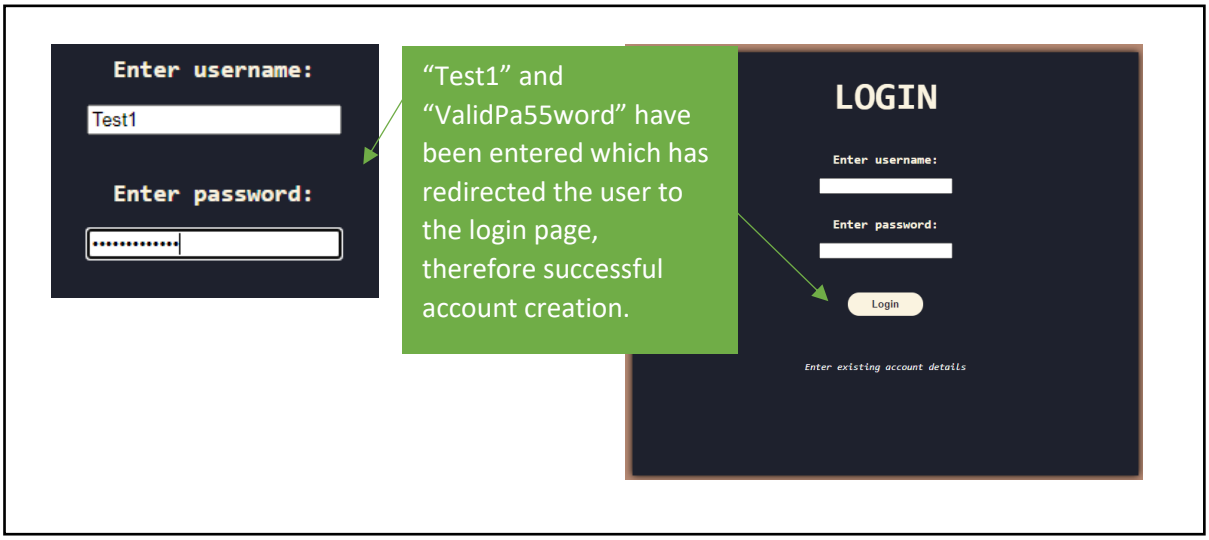


Figure 30

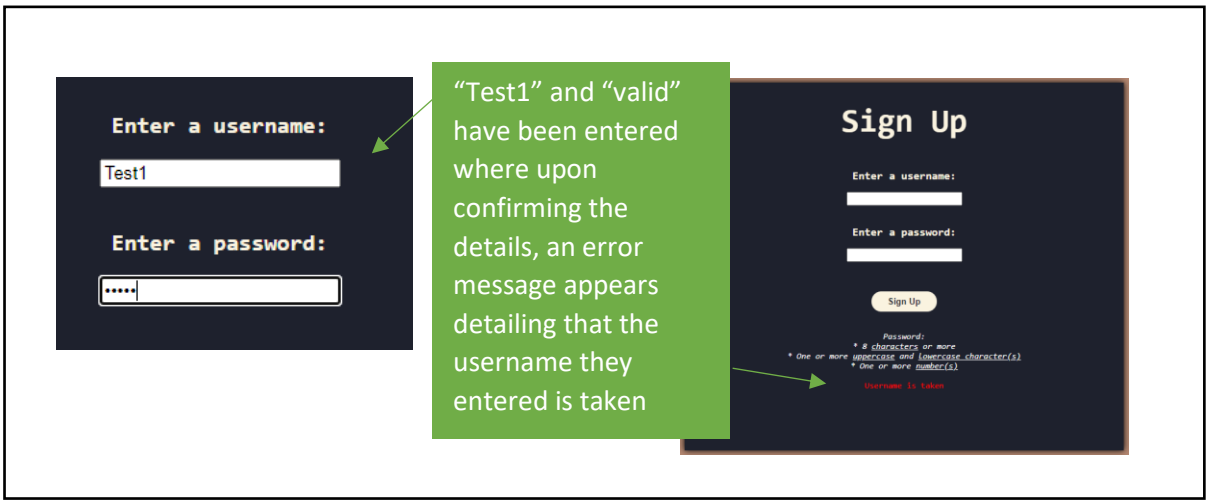


Figure 31

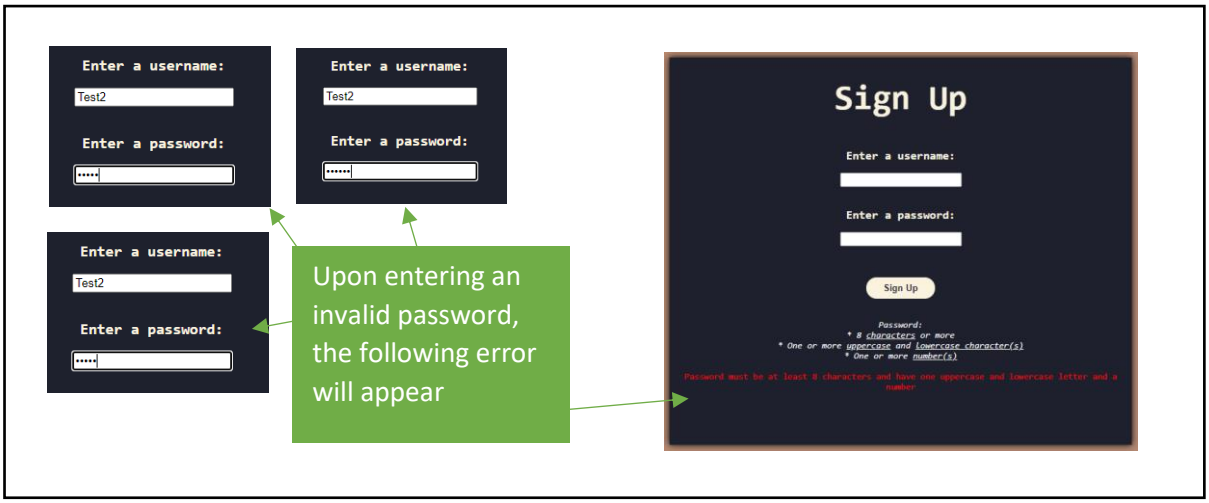


Figure 32

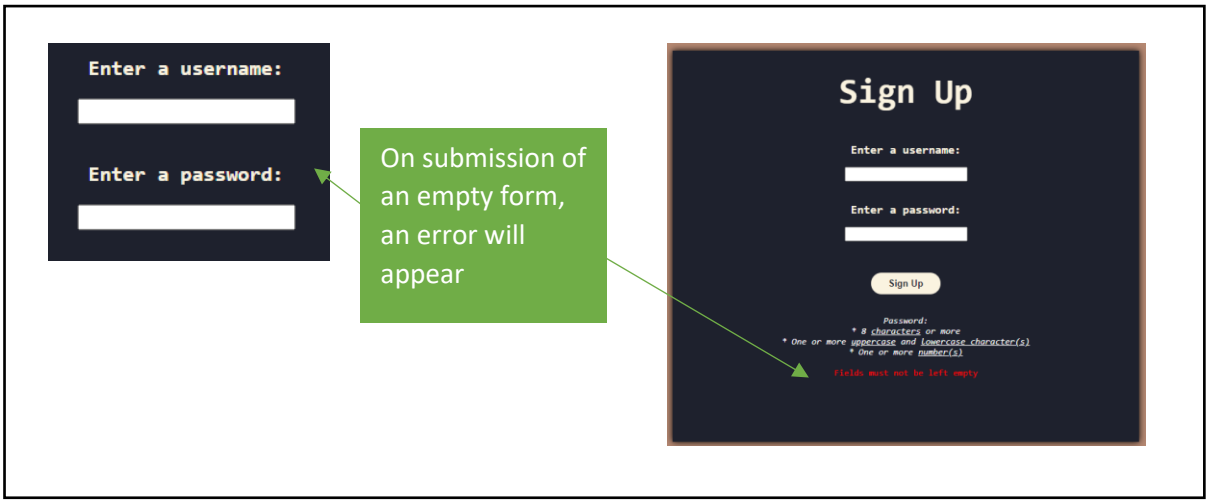


Figure 33

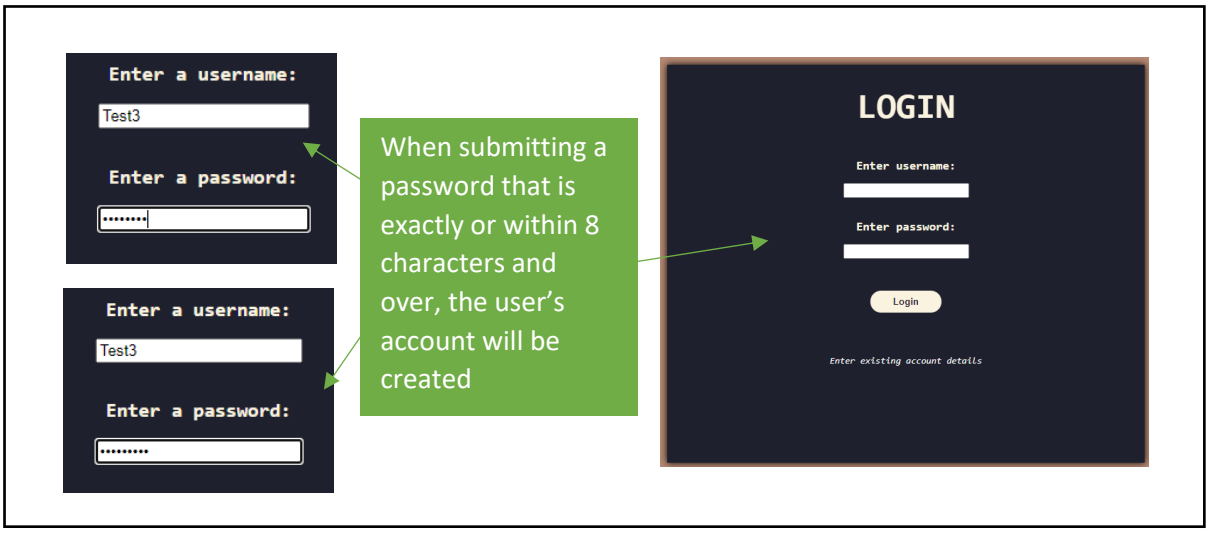


Figure 34

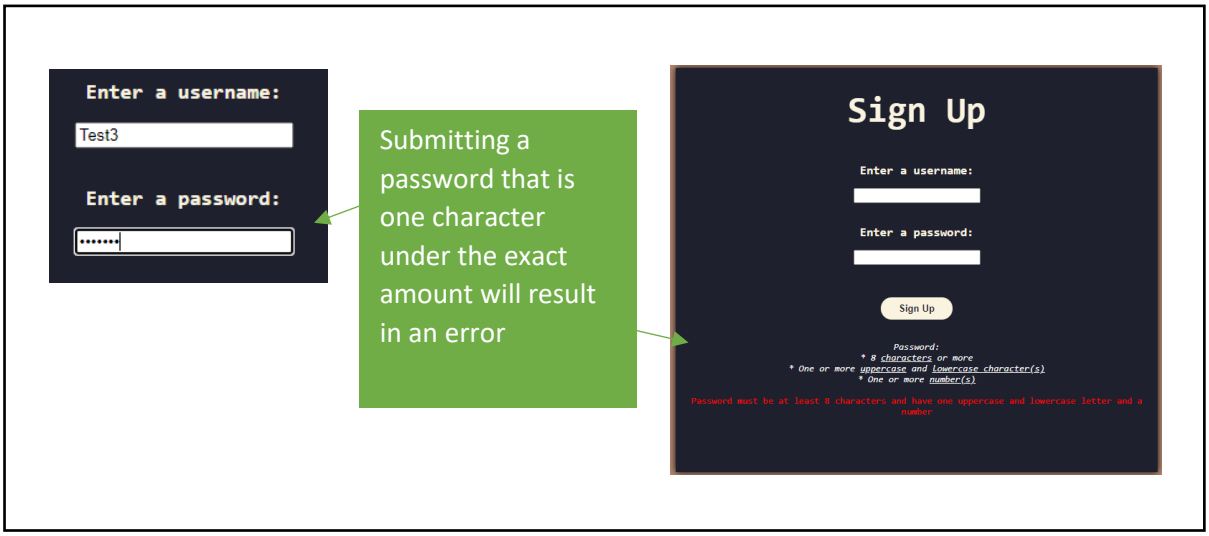


Figure 35

4.1.3. Login validation

Test no.	3
Test purpose	For the user to log into their account with appropriate data input
Test description	When the login page is presented, different information will be entered to test the validation rules of the login process
Test data	**Ty** Username: Test1, Password: ValidPa55word **Er** Username: Test1, Password: valid Username: Test5, Password: ValidPa55word Username: , Password:

	Ex Username: Test1, Password: ValidPa55wors
Expected result	**Ty** The information entered should allow the account Test1 to log in as we have set that information previously **Er** In the first two cases, an error should appear stating that either the “Username or password is incorrect” As for the last case, it should state that “Fields should not be empty” **Ex** The error “Username or password is incorrect” should still appear as it takes only one incorrect character from either form to return an error
Actual result	**Ty** See figure 36 **Er** See figure 37 See figure 38 **Ex** See figure 39

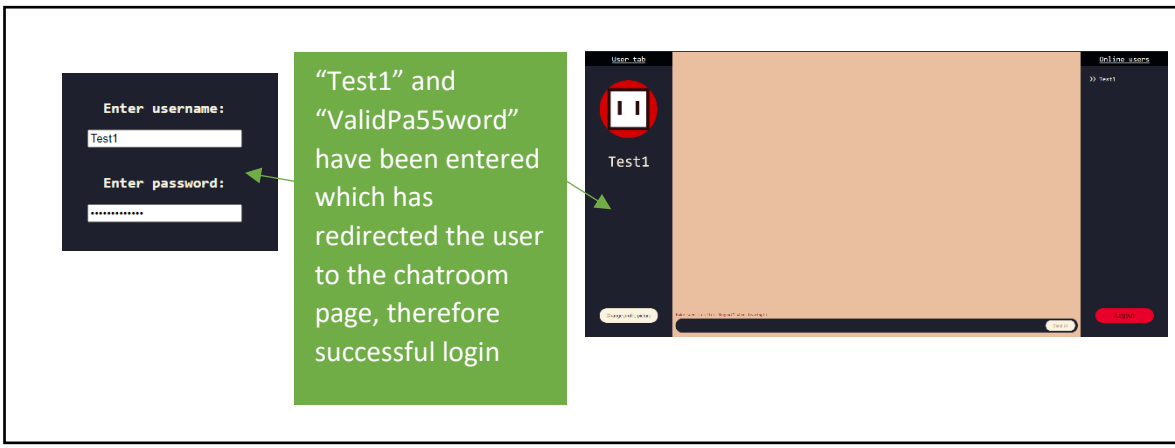


Figure 36

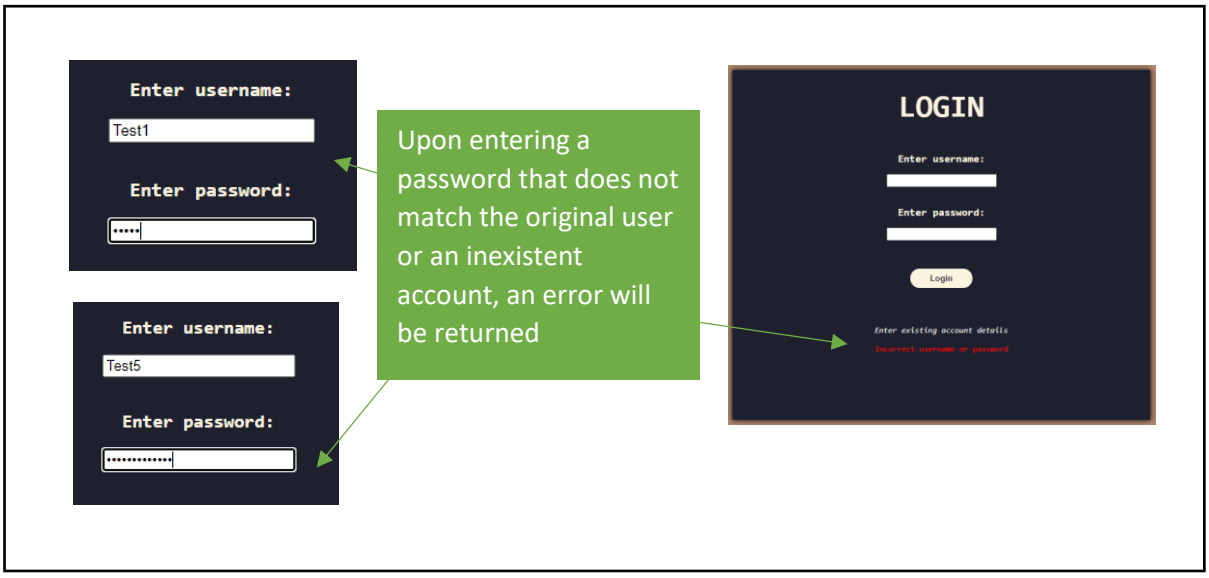


Figure 37

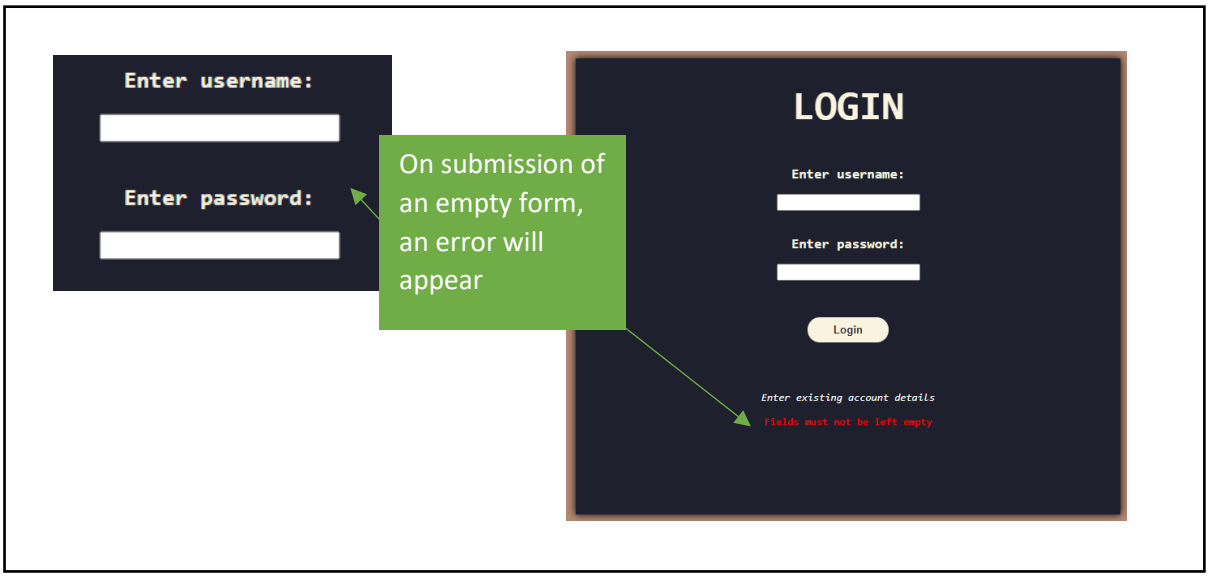


Figure 38

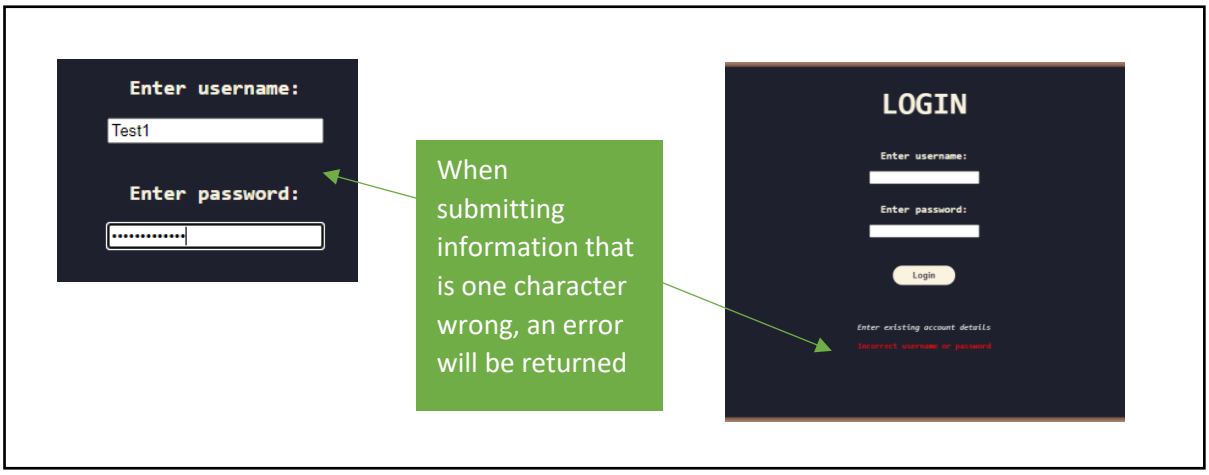


Figure 39

4.1.4. Sending and displaying messages

Test no.	4
Test purpose	To assess whether or not the user can effectively communicate via text to other users
Test description	In the chat box, a range of different messages will be sent where they will then be proved functional if they appear as intended within the chatlog
Test data	**Ty** "*My name is Test1! What is your name?*" – Simple message **Er** " " – Empty message "*Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Donec quam felis, ultricies nec, pellentesque eu, pretium quis, sem. Nulla consequat massa quis enim. Donecap pede justo, fringilla vel, aliquet nec, vulputate eget, arcu. In enim justo, rhoncus ut, imperdies*" – Message of 400 characters **Ex** "*Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Donec quam felis, ultricies nec, pellentesque eu, pretium quis, sem. Nulla consequat massa quis enim. Doneca*" – Message of 300 characters "*Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Donec quam felis, ultricies nec, pellentesque eu, pretium quis, sem. Nulla consequat massa quis enim. Donecap*" – Message of 301 characters
Expected result	**Ty** The message entered should be processed and displayed as is **Er** In the first case, no message should be sent as an empty message is invalid In the second case, the message should be sent but should be cut off after the 300th character **Ex** For the first case, the message should be sent as normal In the second case, the message should be sent but the 301st character should not exist
Actual result	**Ty** See figure 40 **Er** See figure 41 See figure 42 **Ex** See figure 43 See figure 44

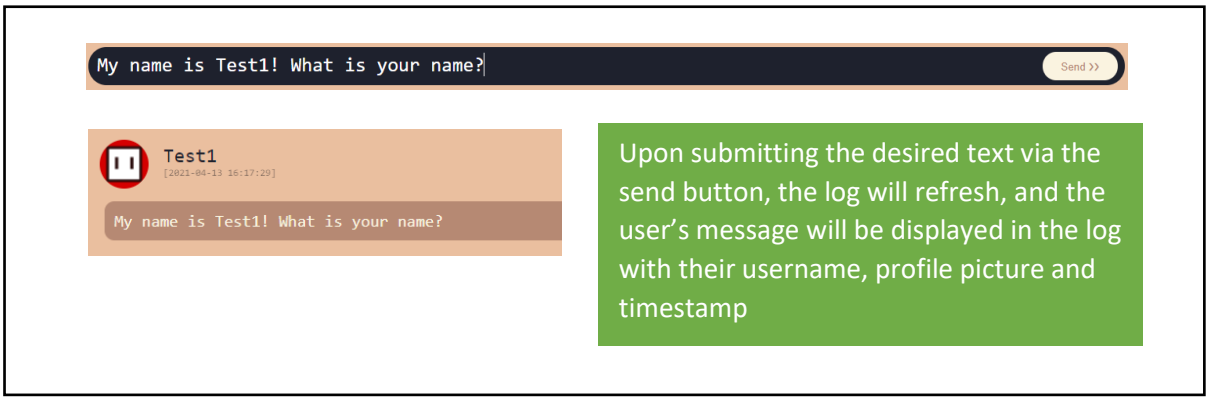


Figure 40

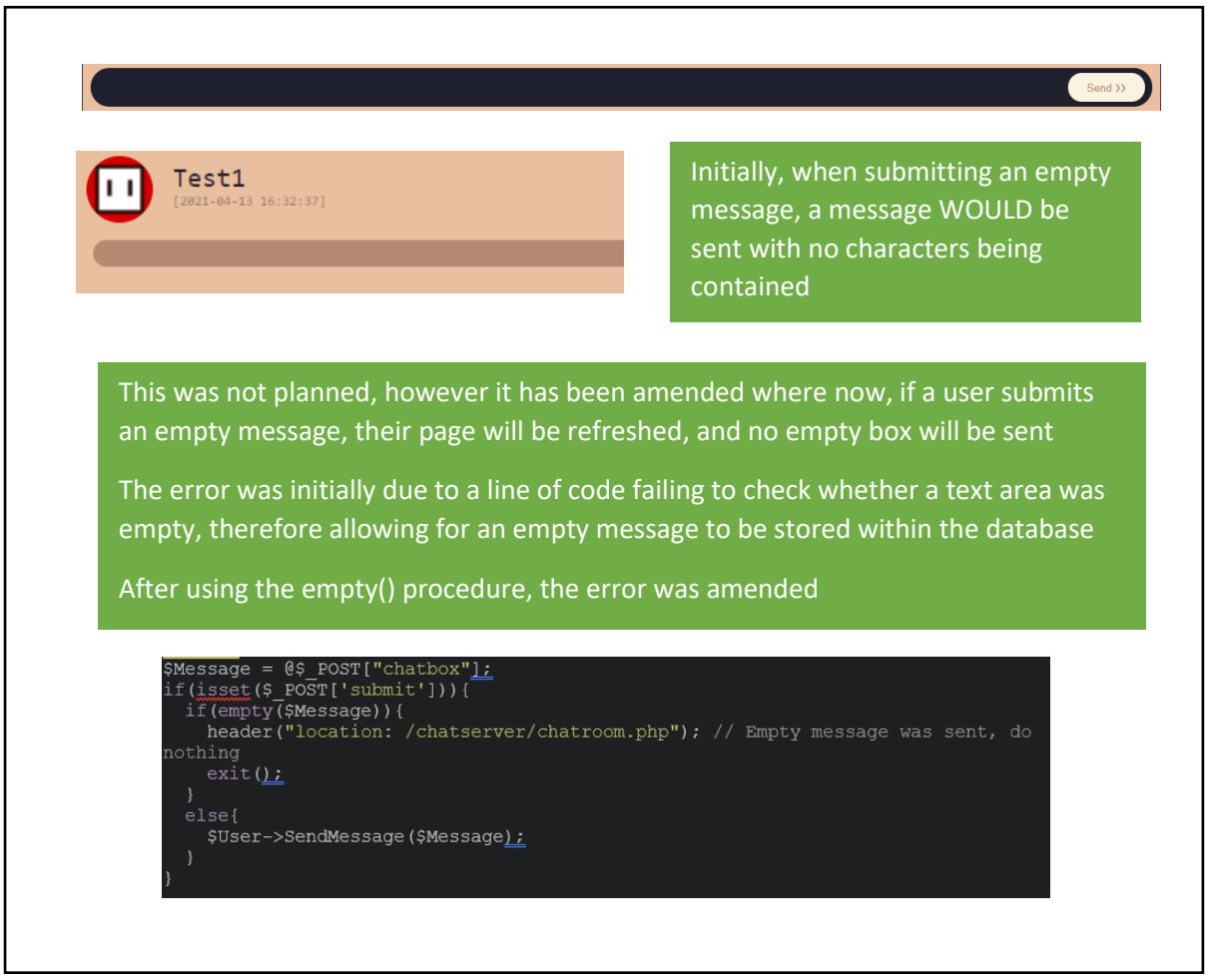


Figure 41

Figure 42

Figure 43

Figure 44

4.1.5. Uploading a profile picture

Test no.	5
Test purpose	To assess whether or not the user can upload a custom image successfully that can be displayed without error
Test description	In my chatroom exists the "Change profile picture" button which I will click and proceed to select a range of images (That I have designed) from my device to verify what can be uploaded
Test data	**Ty** typical1.png (110x110) – Typical profile image size in pixels **Er** erroneous1.txt - .txt file to test a non-image file empty – Upload no image **Ex** extreme1.png (1920x1080) – A very large profile image size extreme2.png (10x10) – A very small profile image size
Expected result	**Ty** The image should be accepted and scaled to the appropriate side for both the preview image and chat box image **Er** For a non-file image, an error should be produced stating that the file must be either a .png or .jpg Similarly, if nothing is selected, the same error should appear for the second case **Ex** For the first case, the image should be scaled down to better suit the dimensions of the preview and icon rather than using the centre of the image as the profile picture For the second case, the image should be scaled by an extreme amount to satisfy the preview's and icon's dimensions which could result in pixilation
Actual result	**Ty** See figure 45 **Er** See figure 46 **Ex** See figure 47

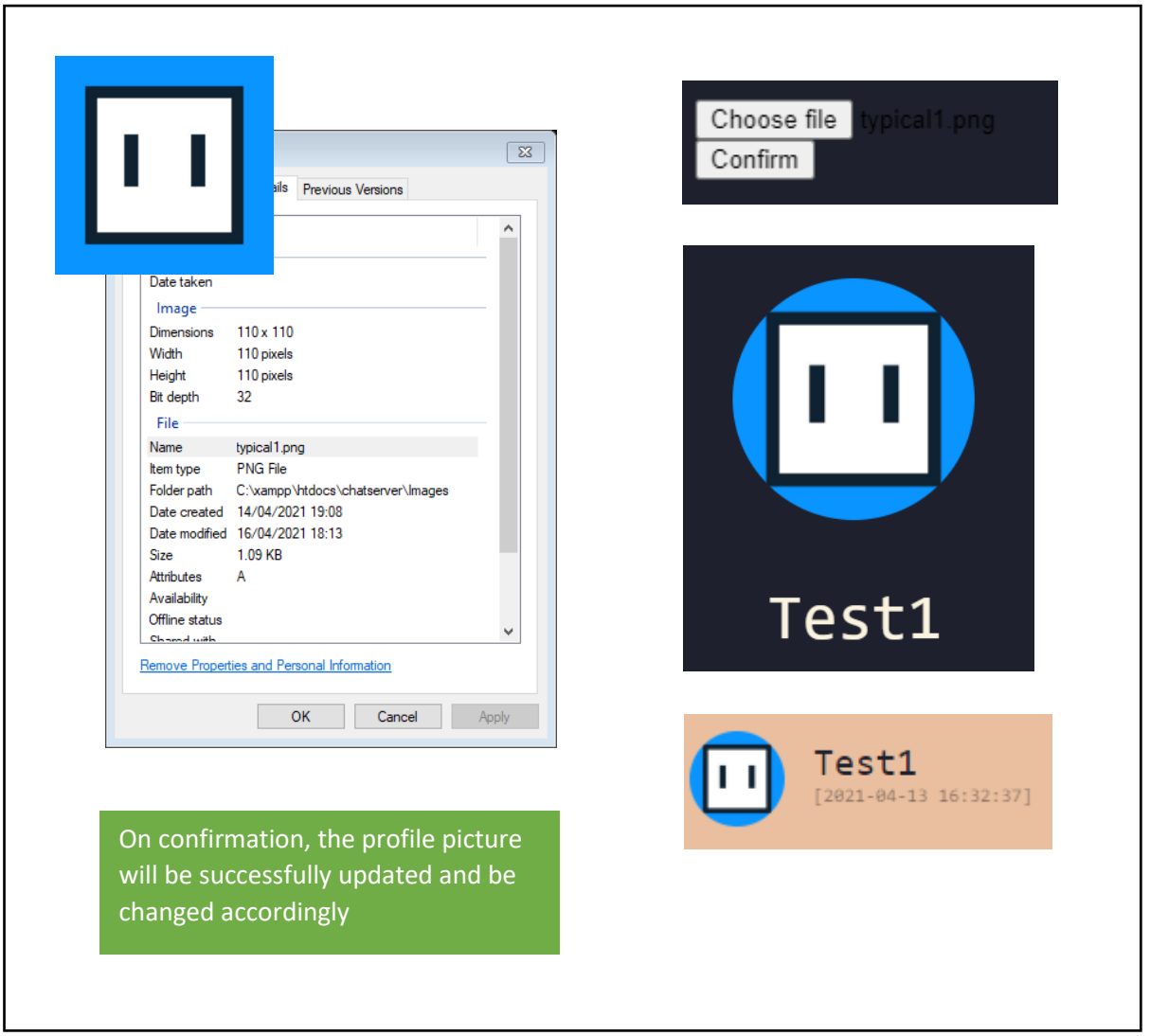
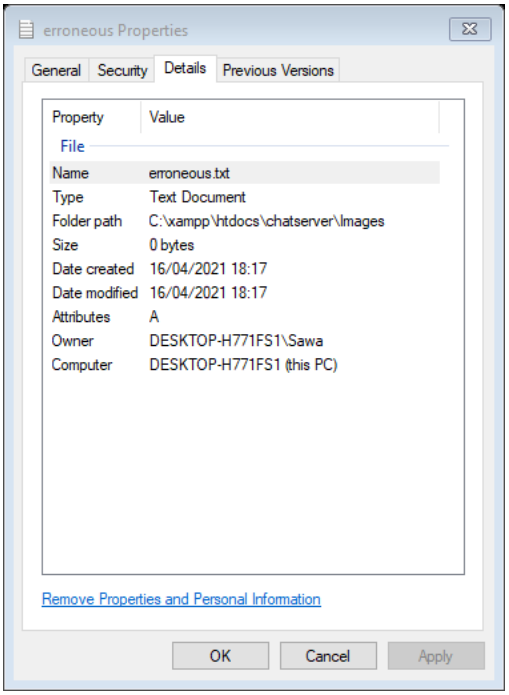
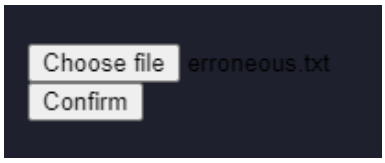
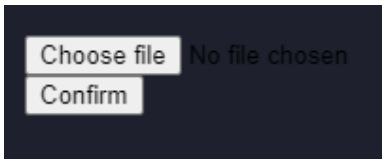

Previous Versions
Date taken
Image
Dimensions 110 x 110
Width 110 pixels
Height 110 pixels
Bit depth 32
File
Name typical1.png
Item type PNG File
Folder path C:\xampp\htdocs\chatserver\Images
Date created 14/04/2021 19:08
Date modified 16/04/2021 18:13
Size 1.09 KB
Attributes A
Availability
Offline status
Remove Properties and Personal Information
OK
Cancel
Apply
Choose file
typical1.png
Confirm
Test1
Test1
[2021-04-13 16:32:37]
On confirmation, the profile picture will be successfully updated and be changed accordingly

Figure 45







When confirming a non-jpg or png image or no image at all, an error will appear below the preview and the profile picture will be returned to the default profile picture

Initially, the user could submit any file beforehand which would result in the image icon below and no error would occur which was unfavourable as the icon represented a broken image error

This error was then resolved as I studied more about file type validation in PHP alongside further testing with my end-user as they helped identify this issue

Figure 46

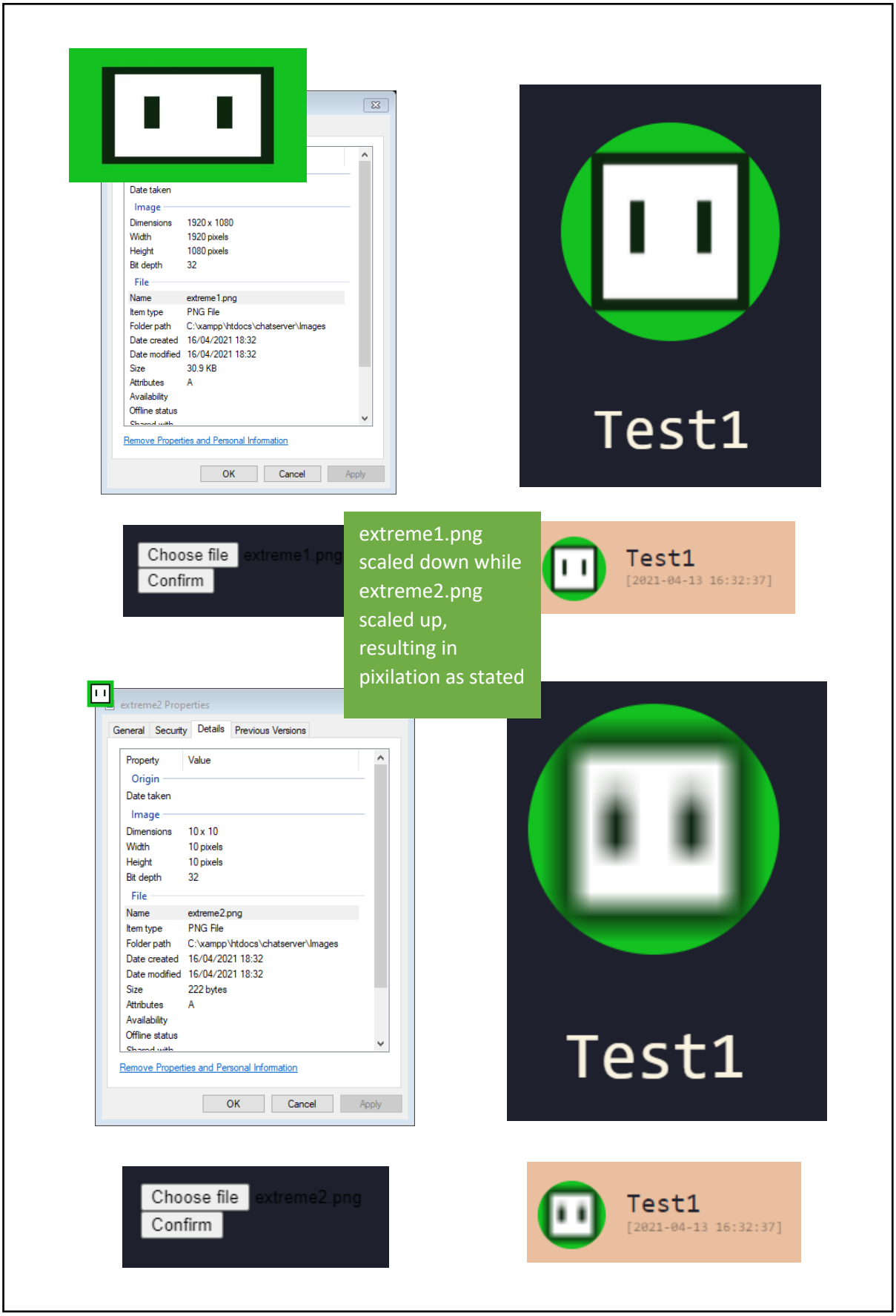
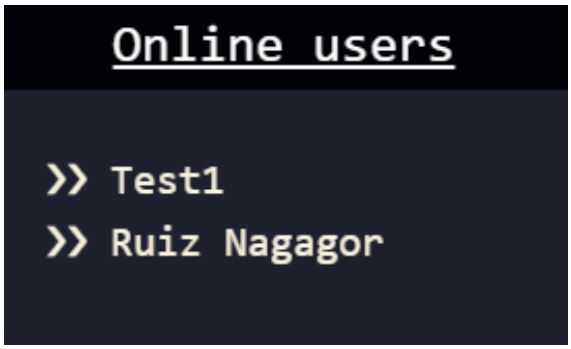

Date taken
Image
Dimensions 1920 x 1080
Width 1920 pixels
Height 1080 pixels
Bit depth 32
File
Name extreme1.png
Item type PNG File
Folder path C:\xampp\htdocs\chatserver\Images
Date created 16/04/2021 18:32
Date modified 16/04/2021 18:32
Size 30.9 KB
Attributes A
Availability
Offline status
Remove Properties and Personal Information
OK
Cancel
Apply
Test1
Choose file
extreme1.png
Confirm
extreme1.png scaled down while extreme2.png scaled up, resulting in pixilation as stated
Test1
[2021-04-13 16:32:37]
extreme2 Properties
General
Security
Details
Previous Versions
Property
Value
Origin
Date taken
Image
Dimensions 10 x 10
Width 10 pixels
Height 10 pixels
Bit depth 32
File
Name extreme2.png
Item type PNG File
Folder path C:\xampp\htdocs\chatserver\Images
Date created 16/04/2021 18:32
Date modified 16/04/2021 18:32
Size 222 bytes
Attributes A
Availability
Offline status
Remove Properties and Personal Information
OK
Cancel
Apply
Test1
Choose file
extreme2.png
Confirm
Test1
[2021-04-13 16:32:37]

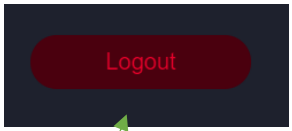
Figure 47

4.1.6. Logging out

Test no.	6
Test purpose	To assess whether or not the user can leave the chatroom and end their session
Test description	I will use my end-user to see whether or not I have left the chatroom as Test1 which can be verified by the Online log on the right-hand side of the chatroom
Test data	Left click via mouse input on the logout and close window buttons
Expected result	When clicking the logout button, Test1 should have their online status changed to 0 in the SQL online table, signifying that they are offline which should then remove them from the list When closing the window, there is no system in place to detect that the user has left, therefore the SQL table cannot change their online status, proving to be a fatal error when showcasing which users are online
Actual result	See figure 48

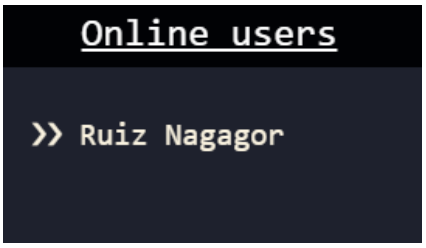


This image here showcases me, and my end user Ruiz Nagagor are present – They will assist me by relaying back an image of the list to indicate whether or not I have successfully logged out and been removed from the list



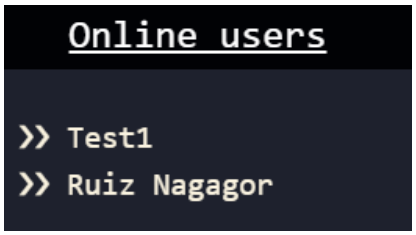
The logout button is clicked indicated by its highlighted colour where it then takes me to the index page

Logging out is successful as my end-user's account is the only account present within the list



Unfortunately, when a user exits the chatroom without clicking on the logout button, their status will remain 1 as PHP nor JavaScript can detect their activity ending and then change their status accordingly

I aim to amend this, however the only solution as of now to mitigate this issue is to provide a notice for the user that request them to click logout when leaving



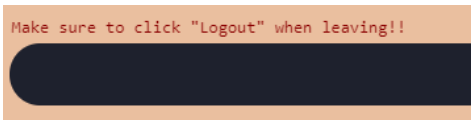


Figure 48

4.2. White-box testing – Assessing internal functionality

4.2.1. phpMyAdmin interface

This section contains the interface used to provide a visual representation of the database as this will assist me in understanding the flow of data throughout my program

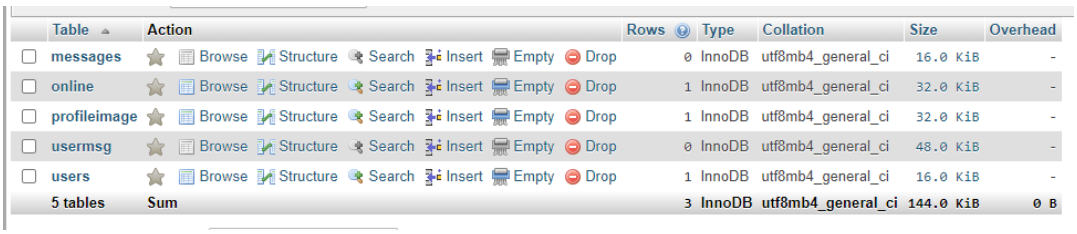
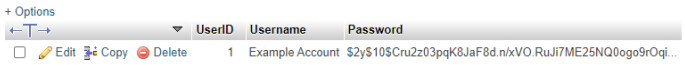


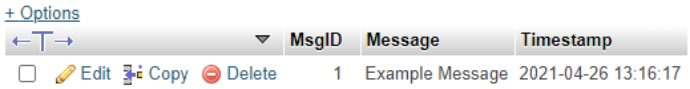
Table	Action	Rows	Type	Collation	Size	Overhead
messages	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_general_ci	16.0 KiB	-
online	Browse Structure Search Insert Empty Drop	1	InnoDB	utf8mb4_general_ci	32.0 KiB	-
profileimage	Browse Structure Search Insert Empty Drop	1	InnoDB	utf8mb4_general_ci	32.0 KiB	-
usermsg	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_general_ci	48.0 KiB	-
users	Browse Structure Search Insert Empty Drop	1	InnoDB	utf8mb4_general_ci	16.0 KiB	-
5 tables	Sum	3	InnoDB	utf8mb4_general_ci	144.0 KiB	0 B

ChatroomDB Tables



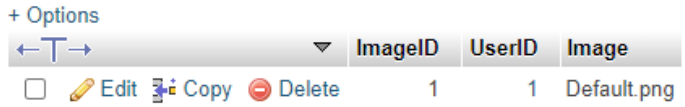
	UserID	Username	Password
Edit Copy Delete	1	Example Account	$2y$10$Cru2z03pqK8JaF8d.n/xVO.RuJi7ME25NQ0ogo9rOqi...

Users table



	MsgID	Message	Timestamp
Edit Copy Delete	1	Example Message	2021-04-26 13:16:17

Message table

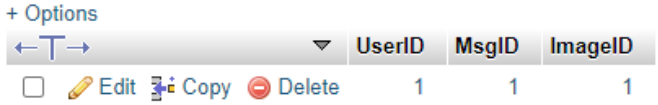


	ImageID	UserID	Image
Edit Copy Delete	1	1	Default.png

ProfileImage table

	isOnline	UserID	Username
Edit Copy Delete	1	1	Example Account

Online table



	UserID	MsgID	ImageID
Edit Copy Delete	1	1	1

UserMsg table

4.2.2. Creating an account

Test no.	7
Test purpose	This will peer into how the account the user created is stored
Test description	I will create a valid account and showcase the SQL database tables relating to the newly created account such as Users which stores the user's username and hashed password
Test data	Username: "Test1" Password: "ValidPa55word"
Expected result	The user should first be added to the Users table which will store a UserID, Username and Password which will be hashed They will also be added to the ProfileImage table where they will be assigned an ImageID and have a null Image value as this will be set to "Default.png" only when the user has logged in Finally, the user will be added to the Online table where their isOnline status will be set to 0 by default
Actual result	See figure 49

UserID	Username	Password
1	Test1	$2y$10$g0TVPdNt7neBNPqmDIlBP.S9z5pSTnMinXxeErq8dI....

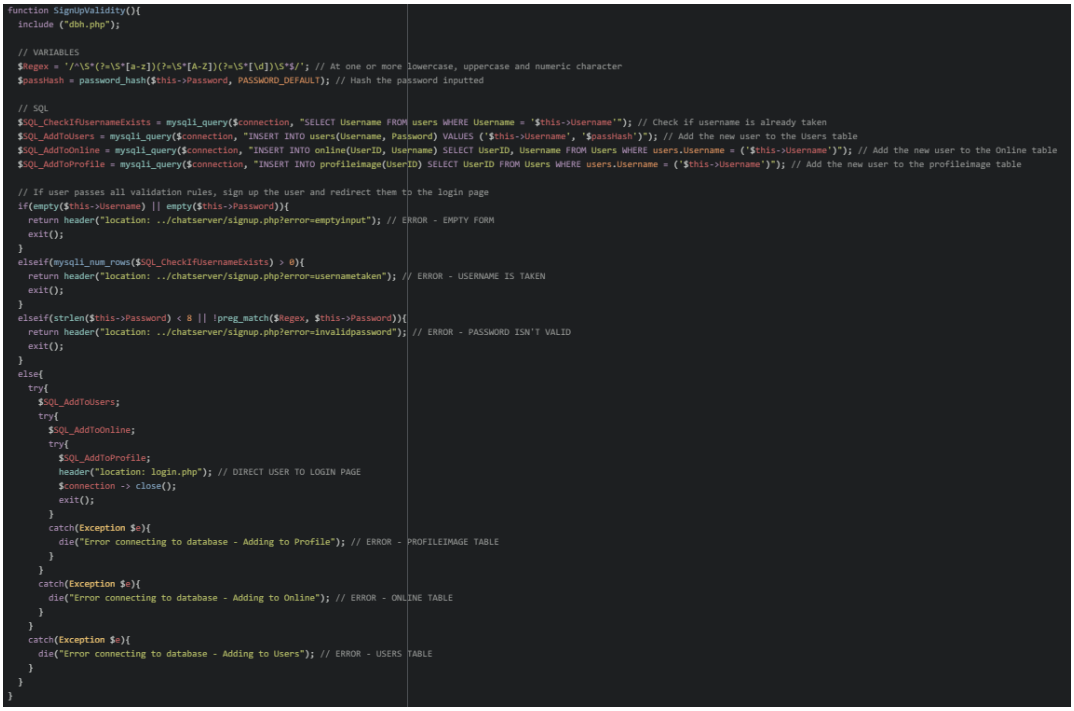
Within phpMyAdmin, the account information is stored within the respective columns with the password not resembling the original input as it has gone through a hash function within the SignUpValidity procedure

ImageID	UserID	Image
1	1	*NULL*

The user has been added to the ProfileImage table with the appropriate data assigned to their rows – In login, if it detects the Image value as null, "Default.png" will be automatically assigned – This is done to prevent mass amounts of "Default.png" files being stored to my directory just through sign-up alone

isOnline	UserID	Username
0	1	Test1

Within the Online table, the user has been added with an isOnline status set to 0 to signify that they are offline – This will change to 1 on login and reset to 0 on logout



```
function SignUpValidity(){
  include ("dbh.php");

  // VARIABLES
  $Regex = '/^\S*(?=\S*[a-z])(?=\S*[A-Z])(?=\S*[\d])\S*$/'; // At one or more lowercase, uppercase and numeric character
  $passHash = password_hash($this->Password, PASSWORD_DEFAULT); // Hash the password inputted

  // SQL
  $SQL_CheckIfUsernameExists = mysqli_query($connection, "SELECT Username FROM users WHERE Username = '$this->Username'"); // Check if username is already taken
  $SQL_AddToUsers = mysqli_query($connection, "INSERT INTO users(Username, Password) VALUES ('$this->Username', '$passHash')"); // Add the new user to the Users table
  $SQL_AddToOnline = mysqli_query($connection, "INSERT INTO online(UserID, Username) SELECT UserID, Username FROM Users WHERE users.Username = ('$this->Username')"); // Add the new user to the Online table
  $SQL_AddToProfile = mysqli_query($connection, "INSERT INTO profileimage(UserID) SELECT UserID FROM Users WHERE users.Username = ('$this->Username')"); // Add the new user to the profileimage table

  // If user passes all validation rules, sign up the user and redirect them to the login page
  if(empty($this->Username) || empty($this->Password)){
    return header("location: ../chatserver/signup.php?error=emptyinput"); // ERROR - EMPTY FORM
    exit();
  }
  elseif(mysqli_num_rows($SQL_CheckIfUsernameExists) > 0){
    return header("location: ../chatserver/signup.php?error=usernametaken"); // ERROR - USERNAME IS TAKEN
    exit();
  }
  elseif(strlen($this->Password) < 8 || !preg_match($Regex, $this->Password)){
    return header("location: ../chatserver/signup.php?error=invalidpassword"); // ERROR - PASSWORD ISN'T VALID
    exit();
  }
  else{
    try{
      $SQL_AddToUsers;
      try{
        $SQL_AddToOnline;
        try{
          $SQL_AddToProfile;
          header("location: login.php"); // DIRECT USER TO LOGIN PAGE
          $connection -> close();
          exit();
        }
        catch(Exception $e){
          die("Error connecting to database - Adding to Profile"); // ERROR - PROFILEIMAGE TABLE
        }
      }
      catch(Exception $e){
        die("Error connecting to database - Adding to Online"); // ERROR - ONLINE TABLE
      }
    }
    catch(Exception $e){
      die("Error connecting to database - Adding to Users"); // ERROR - USERS TABLE
    }
  }
}
```

Section of code responsible for account creation within classes.php, identified as SignUpValidity

Figure 49

4.2.3. Logging into an account

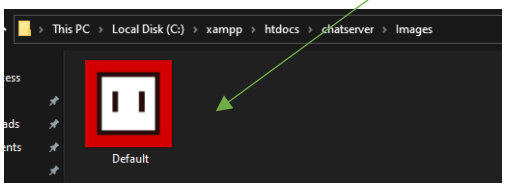
Test no.	8
Test purpose	This will assess the process involved when the user accesses their account
Test description	I will enter “Test1” and “ValidPa55word” and showcase the SQL database to see the values affected
Test data	Username: “Test1” Password: “ValidPa55word”
Expected result	After the data input has been compared to existing data within the table and is valid, the user will first have their isOnline status set to 1 to signify that they are online Then their Image column should be checked for whether or not it is null – If so, “Default.png” should be immediately assigned to provide them a profile picture
Actual result	See figure 50

isOnline	UserID	Username
1	1	Test1

isOnline is set to 1, indicating that the user has logged in successfully and are now online

ImageID	UserID	Image
1	1	Default.png

The user has been assigned “Default.png” – Interaction between PHP and SQL allows for the “Default.png” file within my directory to be brought up and used as the user’s profile picture as a temporary place holder



```php
function LoginValidity(){
  include ('dbh.php');

  // SQL
  $SQL_SelectPassword = mysqli_query($connection, "SELECT Password FROM users WHERE Username = '$this->Username'"); // Get the inputted user's password (If real)
  $SQL_PasswordRow = mysqli_fetch_array($SQL_SelectPassword); // Row password
  $SQL_UpdateOnline = mysqli_query($connection, "UPDATE online SET isOnline = 1 WHERE Username = '$this->Username'"); // Set online status to 1 to indicate they are online

  // If user passes all validation rules, log the user in and redirect them to the chatroom
  if(empty($this->Username) || empty($this->Password)){
    header('location: ../chatserver/login.php?error=emptyinput'); // ERROR - EMPTY FORM
    exit();
  }
  elseif(mysqli_num_rows($SQL_SelectPassword) > 0){
    if(password_verify($this->Password, $SQL_PasswordRow['Password'])){ // VERIFY PASSWORD
      try{
        $SQL_UpdateOnline;
        $_SESSION['Username'] = $this->Username;
        header("location: ../chatserver/chatroom.php"); // DIRECT USER TO CHATROOM
        $connection -> close();
        exit();
      }
      catch(Exception $e) {
        die("Error connecting with database - Online Status"); // ERROR - UPDATING ISONLINE
      }
    }
    else{
      header('location: ../chatserver/login.php?error=invaliddetails'); // ERROR - INCORRECT ACCOUNT INFO.
      exit();
    }
  }
  else{
    header('location: ../chatserver/login.php?error=invaliddetails'); // ERROR - ACCOUNT DOESN'T EXIST
    exit();
  }
}
```

Section of code responsible for logging into the user's account within classes.php, identified as LoginValidity

```php
public function SetDefaultPFP(){
  include ('dbh.php');

  // SQL
  $SQL_Default = mysqli_query($newConnection, "SELECT Image FROM profileimage WHERE profileimage.UserID = '$this->UserID'"); // Set a default profile picture if Image == null

  // Set a default profile picture when the user logs in and has a null value in the Image column
  while($Row = mysqli_fetch_array($SQL_Default)){
    try{
      if($Row['Image'] == null){
        mysqli_query($newConnection, "UPDATE profileimage SET Image = 'Default.png' WHERE UserID = '$this->UserID'");
      }
    }
    catch(Exception $e){
      die("Error setting default profile picture"); // ERROR - COULDN'T SET DEFAULT PFP
    }
  }
}
```

Section of code responsible for assigning the user a default profile picture if value is null within classes.php, identified as SetDefaultPFP

Figure 51

4.2.4. Sending a message

Test no.	9
Test purpose	The purpose of this test is to analyse how the Messages table interacts with the Users table to allow for a message to be sent which is linked to them
Test description	I will send a message by two different accounts: Test1 and Test2 – Then I will present the appropriate tables showcasing what has occurred when a message has been sent by at least two people
Test data	Message 1 by Test1: "Hello, I am Test1" Message 2 by Test2: "Hi! I am Test2" Message 2 by Test2: "Nice to meet you!"
Expected result	The messages should first be logged into the Messages table where the message itself will be stored alongside a timestamp and ID – This will allow for a relationship to be established Inside my code, a query has been issued which gets the most recent message and the user who has sent it – Then, within the UserMsg table, they are both related alongside the user's ImageID, allowing for a message to be sent and displayed
Actual result	See figure 52

MsgID	Message	Timestamp
1	Hello, I am Test1	2021-04-22 15:11:25
2	Hi! I am Test2	2021-04-22 15:11:37
3	Nice to meet you!	2021-04-22 15:11:42

As expected, the messages sent have been assigned a MsgID and Timestamp alongside itself being stored as Message for later use

UserID	MsgID	ImageID
1	1	7
2	2	8
2	3	8

Within the UserMsg table, UserID 1 (Test1) has been associated with MsgID 1 whereas UserID 2 (Test2) has been associated with MsgID 2 and 3 as both messages were sent by this account – The ImageID has also been stored here as this allows for the user's profile picture to be brought up and displayed alongside all other relevant information within the message from their appropriate tables e.g. Username which is accessed by UserID

```php
public function SendMessage($Message){
  include ('dbh.php');

  // SQL
  $SQL_ImageID = mysqli_query($newConnection, "SELECT ImageID FROM profileimage WHERE UserID = '$this->UserID'"); // Get all users' image ID
  while($row = mysqli_fetch_array($SQL_ImageID)){
    $ImageID[] = $row['ImageID'];
  }
  $SQL_InsertMessage = mysqli_query($newConnection, "INSERT INTO messages(Message) VALUES ('$Message')"); // Insert the message into the Messages table
  $SQL_MessageID = mysqli_insert_id($newConnection); // Fetch the recent message ID by that user
  $SQL_InsertUserMsg = mysqli_query($newConnection, "INSERT INTO usermsg(UserID, MsgID, ImageID) VALUES ('$this->UserID', '$SQL_MessageID', '$ImageID[0]')"); // Insert values into usermsg table to establish relation

  // If user sends a message, pass the message through the appropriate tables to match the message with the correct user
    try{
      $SQL_InsertMessage;
      try{
        $SQL_MessageID;
        try{
          $SQL_InsertUserMsg;
          header("location: ../chatserver/chatroom.php"); // REFRESH CHATROOM
          exit();
        }
        catch(Exception $e){
          die("Error with database - Relation table"); // ERROR - RELATION TABLE NOT INSERTING
        }
      }
      catch(Exception $e){
        die("Error with database - Message ID"); // ERROR - CANNOT RETRIEVE MESSAGE ID
      }
    }
    catch(Exception $e){
      die("Error with database - Sending Message"); // ERROR - MESSAGES TABLE NOT INSERTING
    }
  }
```

Section of code responsible for allowing the user to send a message within classes.php, identified as SendMessage

```php
<form action="<?=$_SERVER['PHP_SELF']?>" method="post">
    <textarea id="text" rows="8" cols="80" method="post" name="chatbox"></textarea>
    <p id="notice">Make sure to click "Logout" when leaving!!</p> <!-- !! Inform user on logging out to set online status to 0 -->
    <button id="submit" type="submit" name="submit">Send ))</button>
</form>
```

HTML code that forms the text area which is used for the user to send a message within chatroom.php

```php
// SEND MESSAGE
$Message = @$_POST["chatbox"];
if(isset($_POST['submit'])){
  if(empty($Message)){
    header("location: /chatserver/chatroom.php"); // Empty message was sent, do nothing
    exit();
  }
  else{
    $User->SendMessage($Message);
  }
}
```

Code that checks to see if the user is sending a valid message (Not empty) within chatroom.php

```php
public function DisplayMessages(){
  include ('dbh.php');

  // SQL - Select all the messages alongside their respective user and their profile image for each new line in the chatlog
  try{
    return mysqli_query($newConnection, "SELECT users.Username, profileimage.Image, messages.Message, messages.Timestamp
    FROM users
    INNER JOIN usermsg ON users.UserID = usermsg.UserID
    INNER JOIN messages ON usermsg.MsgID = messages.MsgID
    INNER JOIN profileimage ON usermsg.ImageID = profileimage.ImageID
    ORDER BY messages.Timestamp");
  }
  catch(Exception $e){
    die("Error with database - Relation not executing"); // ERROR - RELATION TABLE ISN'T RELATING DATA
  }
}
```

Section of code responsible for displaying messages within classes.php, identified as DisplayMessages

```php
<!-- CHATLOG -->
<div id="log">
    <p id="messages">
        <?php
        // Fetch messages and output them for each line
            while($MessageRow = mysqli_fetch_array($Messages)){
              echo "<img id='iconPFP' src='Images/".$MessageRow['Image']."'>";
              echo "<p id='time'>[{$MessageRow['Timestamp']}]";
              echo "<p id='name'>{$MessageRow['Username']}<p>";
              echo "<p id='message'>{$MessageRow['Message']}</p>";
            }
        ?>
    </p>
</div>
```

HTML code that allows for messages to be displayed alongside their metadata

Figure 52

4.2.5. Uploading a profile picture

Test no.	10
Test purpose	This test will be used to assess the process of uploading a profile picture internally
Test description	I will upload two different images that will act as Test1's profile picture and showcase what has occurred within the database and file directory
Test data	Image 1: "Image1.jpg" Image 2: "Image2.jpg"
Expected result	When uploading image 1, "Default.jpg" should be replaced with "Image1.jpg" within the database via the UPDATE query The image should then be transferred to my file directory where it will be stored for retrieval so that it can be displayed Image 2 should do the same except it will replace image 1
Actual result	See figure 53

Image1.jpg

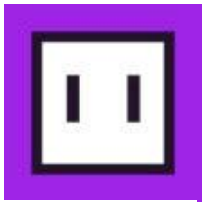
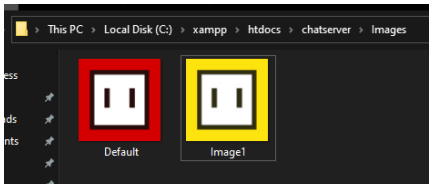
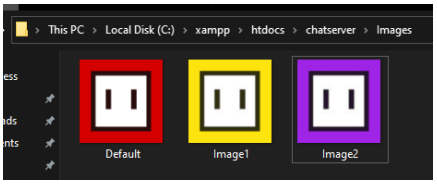


Image2.jpg

ImageID	UserID	Image
7	1	Image1.jpg



ImageID	UserID	Image
7	1	Image2.jpg



Upon confirmation, the value within Image of the respective user (Test1) will be changed to the name of the image

This image will then be stored within my directory so that it can then be accessed for it to be displayed

```php
public function UploadPFP(){
  include ('dbh.php');

  // VARIABLES
  $Location = "Images/".basename($_FILES['image']['name']); // File directory
  $Image = $_FILES['image']['name']; // Establish image name

  // SQL
  $SQL_UpdatePFP = mysqli_query($newConnection, "UPDATE profileimage SET Image = '$Image' WHERE UserID = '$this->UserID'"); // Update user's Image to the one they have uploaded

  // On a confirmed image file, update the user's profile picture by updating the Image column and storing the file into my directory
  if($_FILES['image']['type'] == "image/jpeg" || $_FILES['image']['type'] == "image/png"){
    try{
      $SQL_UpdatePFP;
      try{
        move_uploaded_file($_FILES['image']['tmp_name'], $Location); // Try for whether the file has been directed successfuly to my directory
        header("location: ../chatserver/chatroom.php"); // REFRESH CHATROOM
        exit();
      }
      catch(Exception $e){
        die("Error with directory - Moving file"); // ERROR - MOVING FILE TO MY DIRECTORY
      }
    }
    catch(Exception $e){
      die("Error with database - Updating Profile"); // ERROR - PROFILEIMAGE TABLE NOT UPDATING
    }
  }
  else{
    header("location: ../chatserver/chatroom.php?error=invalidpfp");
    mysqli_query($newConnection, "UPDATE profileimage SET Image = 'Default.png' WHERE UserID = '$this->UserID'");
    exit();
  }
}
```

Section of code responsible for uploading a profile picture within classes.php, identified as UploadPFP

```php
// UPLOAD PFP
if(isset($_POST['profilepic'])){
  header("location: /chatserver/chatroom.php?action=changeprofilepicture");
  exit();
}
if(isset($_POST['confirm'])){
  $User->UploadPFP();
}
```

Code that will adjust the page to display the file browser option for the user to change their profile picture and check whether they have submitted the image

```php
public function DisplayPFP(){
  include ('dbh.php');

  // Get user's profile image to be displayed
  return mysqli_query($newConnection, "SELECT Image FROM profileimage WHERE profileimage.UserID = '$this->UserID'");
}
```

Section of code responsible for getting the user's profile picture to be displayed within classes.php, identified as DisplayPFP

```php
<div>
    <form action="<?=$_SERVER['PHP_SELF']?>" method="post">
        <button id="profilepic" type="submit" name="profilepic">Change profile picture</button>
    </form>
    <form id="change" action="<?=$_SERVER['PHP_SELF']?>" method="post" enctype="multipart/form-data">
        <?php
            // Execute method if user submitted an image
            if(isset($_GET['action'])){
             if($_GET['action'] == "changeprofilepicture"){
                echo "<input type='file' name='image' accept='image/*'> <button id='confirm' type='submit' name='confirm'>Confirm</button>";
              }
            }
        ?>
    </form>
    <?php
        // Get the user's image file to be displayed as a preview
        while($imageRow = mysqli_fetch_array($SidePFP)){
          echo "<div>";
          echo "<img id='sidePFP' src='Images/".$imageRow['Image']."'";
          echo "</div>";
        }
        // ERRORS
        if(isset($_GET["error"])){
          if($_GET["error"] == "invalidpfp"){
            echo "<p id='usererror'>Image must be either a .png or .jpg</p>";
          }
        }
    ?>
    <p id="sidename"><?=$User->Username?></p>
</div>
```

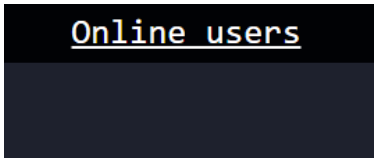
HTML and PHP code that will display the file browser button, validate the image the user has submitted and display their image as well as their username beneath it

Figure 53

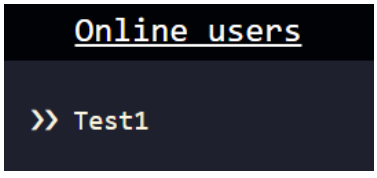
4.2.6. Updating online status

Test no.	11
Test purpose	This test will look into how the user's online status is changed when logging in and logging out
Test description	I will log in and log out as Test1 and provide evidence of the user's isOnline status
Test data	Left click via mouse input on the login and logout buttons
Expected result	When logging in, Test1 should have their isOnline status within the Online table changed to 1 to signify that they are online When logging out, Test1 should then have their isOnline status changed to 0, signifying that they are now offline with the list not representing their username as it only fetches accounts with an isOnline status of 1
Actual result	See figure 54

isOnline	UserID	Username
0	1	Test1



isOnline	UserID	Username
1	1	Test1



The images showcase what occurs in the Online table within the database as the user's isOnline status is changed depending on their action

```php
public function DisplayOnlineUsers(){
  include ('dbh.php');

  // Get all users who have an online status of 1
  return mysqli_query($newConnection, "SELECT Username FROM online WHERE isOnline = 1");
}

public function Logout(){
  include ('dbh.php');

  // End the users session and update them to appear offline on logout
  unset($_SESSION['Username']);
  session_destroy(); // Delete the user's session
  header("location: /chatserver/index.php");
  return mysqli_query($newConnection, "UPDATE online SET isOnline = 0 WHERE Username = '$this->Username'"); // Set online status to 0 and redirect them to the login page
  exit();
}
```

Section of code responsible for displaying online users and logging out the user within classes.php, identified as DisplayOnlineUsers and Logout respectively

```php
<div id="onlinelist">
   <p id="onlinetitle">Online users</p>
   <?php
      // Output a list of the online users
      while($list = mysqli_fetch_array($List)){
        echo "<p id='onlineuser'>>> {$list['Username']}</p>";
      }
       ?>
   <form action="<?=$_SERVER['PHP_SELF']?>" method="post">
      <button id="logout" type="submit" name="logout">Logout</button>
   </form>
</div>
```

HTML code that allows for the online users to be displayed and formulating the logout button

Figure 54

4.3. End-user testing – Assessing quality

This section will enquire my end-user Ruiz Nagagor over a phone call to provide me feedback on the qualitative features of my chatroom which includes its ease of use, responsiveness, features provided and their overall opinion of it.

Interview with Ruiz Nagagor (End user) – Taken over a phone call - 03/05/21

Q1. How easy was it to use the website?

The website was exceptionally easy to use throughout my experience because it was clear to me what I had to do as buttons were distinct and instructions were provided like the requirements for a strong password. When I entered the chatroom, I was drawn to each section through their contrast of colours and different features. The chatlog was easy to see as it occupied most of the page with each message having a clear presence because of the highlights and profile pictures which were easy to setup as well as messages being easy to type and send. It wasn't difficult seeing other users online and logging out was easy.

Q2. Was the website responsive to your liking?

Signing up and logging in was quick which I admired because someone who simply wanted to chat would begrudge an email process. Sending a message was not difficult and I found my messages appearing instantly on the chatlog. Admittedly, having to click the send message button instead of "Enter" that most messaging services offer does become tiring. Setting up a profile picture was not an issue and I was able to see other user's messages and hold conversations comfortably. Exiting the chatroom was also fast and simple, although I would prefer being able to just leave the website through alternative means to make myself offline.

Q3. What are your thoughts on the features provided by the website?

I liked being able to change my profile picture, it helped to provide me with a greater identity which a username would not do justice. Being able to see other users was also a nice touch and of course, being able to send a message and receive one back was great. But I do feel as if the website could use more features to make the experience better, perhaps changing the username after creating an account or sending messages besides text?

Q4. How was your overall experience?

Overall, it was a pleasant experience communicating with some of my friends through a chatroom through its easy to use and well-designed interface, but it definitely lacks features to make the aspect of communicating even better like sending images and gifs, therefore it could use additional work.

END OF TESTING

5. Evaluation

5.1. Evaluation of main objectives

No.	Objective	Evaluation
1	Allow for a user to create a unique account securely	***ACHIEVED*** I have been able to provide the user the opportunity to create an account which has been deemed secured as evident with password requirements and hashing alongside the account being unique through a provided account number unique to that user, allowing them to be identifiable.
2	Create an area that can store user data which includes the user's username, password, messages, and profile pictures in an SQL database	***ACHIEVED*** These areas have been created to accommodate the user's information with evidence of that being provided through the images depicting the relevant SQL tables inside of the phpMyAdmin tool.
3	Allow for the user to log into their created account	***ACHIEVED*** Users can log into their accounts immediately after their creation with evidence of this shown through my black box testing phase and my end-user's feedback as well as their presence on the site.
4	Have the user's account be associated with a message they will send by linking the two through a relation table	***ACHIEVED*** My database has been setup with a relation table which has allowed for messages and users to be related accordingly to output the message alongside its details and the sender's username alongside the additional ProfileImage table to display the user's profile image.
5	Let users be able to send messages	***ACHIEVED*** Messages can be sent of varying lengths and a limit has been set to prevent an abundance of text being sent as shown with my black-box testing.
6	Let the user be able to see all messages sent	***ACHIEVED*** User must be able to see who has sent messages alongside the time sent to provide metadata on the message. The messages must be also be displayed in order of time sent from most recent to least recent to keep it within a chronological

		order, making it easier to view recently added messages on the chatlog.
7	Add a feature that lets the user be able to see who is online	***ACHIEVED*** I have effectively created an area within the chatroom which displays all of the online users while ensuring that no offline users are present.
8	Add a feature that will allow users to change their profile picture	***ACHIEVED*** As proven by both my black-box and white-box testing alongside my end-user's feedback, I have formulated a method for the user to be provided and allow for them to update their profile picture with set errors to prevent abnormal files being sent.
9	Let users log out	***PARTIALLY ACHIEVED*** While I have made it so that the user can logout through the logout button which redirects them to the index page, they unfortunately cannot logout by exiting the website through others, or rather they can but this will cause for an error within the chatroom as the user will still be displayed as online.

5.2. Evaluation of end-user's feedback (4.3)

No.	End-user's feedback	Evaluation
1	*"Admittedly, having to click the send message button instead of "Enter" that most messaging services offer does become tiring"*	I did not consider using the keyboard to send text as it would then be difficult to provide separate lines of text or paragraphs as the Enter key allows for this, although this could be amended through JavaScript.
2	*"I would prefer being able to just leave the website through alternative means to make myself offline"*	This issue was difficult to mitigate as PHP did not offer any solution to allowing for the database to update on detection of the user logging out. This, however, could have been achieved with JavaScript but due to time constraints, this remained as an afterthought.
3	*"But I do feel as if the website could use more features to make the experience better, perhaps changing the username after creating an account or sending messages besides text?"*	I had initially conceptualised and designed a change username feature after my main objectives were complete but due to time constraints alongside a plethora of errors between the website and SQL tables, it had to be dropped. Unfortunately, other types of data such as images and gifs would be difficult to allow for as this could strain my device which enacts as the host, but it is doable as proven by being able to allow for

		profile pictures, however that operates on a smaller scale whereas sending images and gifs as messages could lead to a large amount being relayed to my device. I also intended for other features such as user's changing the colour of their metadata or having personal chatrooms but alas these were not apart of the main objectives and time constraints disabled me from doing such a thing.

5.3. Conclusion

To conclude, as my technical solution had adhered to all of my designs, the website I created has met all of my main objectives which were key to provide the user Liane Silva with an adequate means of communication across the internet by allowing for text to be relayed between two parties while maintaining an account unique to them which they can customise through a username and profile picture to identify one another and securely communicate.

While my main objectives were met, I unfortunately could not include additional substance to my website which could have improved their experience when communicating such as sending images to one another which is a prime feature of modern chatrooms or having the ability to edit their password or username upon submission of an account.

Nonetheless, the chatroom proves to be functional with my end-user ensuring both the functionality and ease of access when operating it, providing me with additional insight on what I could do to improve my website to enhance the experience. To achieve these improvements, it would require:

- A dedicated computer as this can act as the server to allow for heavier features such as private chatrooms and sending media – This can also improve the responsiveness of the chatroom as accessing an idle computer dedicated to this task will be quicker than accessing mine, allowing for an extremely active chatroom to be responsive to each user as it will prevent slow down
- Additional understanding of my chosen languages to improve upon existing features like being able to change the user's online status by detecting whether or not they are still on the chatroom within a window on their browser
- Additional time to provide myself with the ability to test and research different designs relating to a chatroom in hopes of creating a solution for different problems

END OF EVALUATION