LAU
School of Arts and Sciences
Department of Computer Science & Mathematics

Color Chat

Color your conversations, color your life!

Public web chat application

by

Abdallah Tourbah

Email: abdallah.tourbah@lau.edu

Tel: +961 3 900895

A REPORT

submitted to Dr. Ramzi R. Haraty in partial fulfillment of the requirements for the course “CSC599: Capstone Project” in Computer Science

Report– 12,458 words – 91 pages

April 30 2024

This page is intentionally left blank

I. Instructor's Evaluation:

Dear Dr. Haraty,

After Reading the full and understanding my capstone project and the report, i would like to thank you for taking the time to fill my evaluation feedback form since i always want to collect your very own feedback on my work since you are the judge obviously but also because ur advices also help me to improve and your guidance every 2 weeks was extremely helpful and well appreciated.

- Grade: 
- Document Report:
- Likes:
  -
  -
- Dislikes:
  -
  -
- General Comments:
  -
  -
- Future Recommendations:
  -
  -

Your dear student,

Abdallah Tourbah

II. Acknowledgment:

In this section I would like to point out all the people who adviced me and encouraged me and guided me throughout this project. First of all, I would like to express my utmost thanks to Dr. Haraty who is the judge of this project and my capstone advisor whom I always consult every 2 weeks regarding my progress in this project and seek out his critical advices.

In addition, I owe huge thanks for my web doctor who tried his best to help me when i faced issues or stopping points always helped me for example advicing me instead of using shell command during login with SSH keys advicing me to use SSH built-in functions in php and was also the one who tried his best to help me solve the email sending to the user the otp code to reset the password.I also owe some of my thanks to my LAU IT network team who fixed the database listening port and allowed me to continue testing and special shoutout mentioned to Mr. Saad Kalash who tested my website and was also the one who gave me the capstone idea. I also owe huge respect to Dr. Tajeddine who taught me about network and security without him I maybe wouldn't have chosen that capstone topic.

Nobody has been more important in this pursuit of this project rather than my own dedication and perseverance with the encouragement from Dr. Haraty. I would like to sincerely thank my uncle who already did a chat application but ended up unsuccessful that was one major factor that drove me to choose chat application with the support of my uncle towards me.

III. Preface:

This report was presented to the department of Computer Science at the Lebanese American University, Beirut Campus aims at precisely describing the making a chat application with the use of security ssh key generation during login and public group chat app for all comunicating through ip and port number like a discord server single channel for all which i developed. The main elements of this report include general features, overview of structure and source code explanation. It is intended to affirm the specific functionalities that were implemented in the game, which were discussed with my advisor Dr. Ramzi Haraty. The report also includes glossary, table of contents and other trivial material.

IV. Declarations:

I hereby confirm, affirm and ascertain that:

- This project is my own professional work developed during 4 months time and i have received little to no assistance from doing it basically more than 99% of the project was my work.

- No university resources in whole or in part have been used when doing this project such as library resources (books, computers, journals, printers etc…) apart from accessing the wifi for my web application to test it and trying to use lau host name for the smtp mail server for password reset.

- Not a single part of money was put for this project because I didn't have to deploy it online but if i had to deploy it online i may have paid a total of 5 dollars because uploading a remote sql database is for premium members in 000webhost. However so many hours of pure and hard work was put for this project.
- Due to time constraints, i was neither able to finish the forgot password functionality reset by mail with otp code work nor to add a search function for the ip and port in my main public chat page.
- After the project has been graded, the chat app won't be shared or showed from me to anyone. In fact, it shall remain a secret and the only connection i will be having with LAU as it will probably be the main proof left i would have of my university when I look back to my career achievements.

V. Table of Contents:

VI. Report Guidelines:

A REPORT FORMAT SPECIFICATION document has been provided for students in order to follow report guidelines. Not all report guidelines have been followed since some are unreasonable and absurd.

Followed guidelines:

1. Report Structure Pages

- ✓ Title page (unnumbered)
- ✓ Signature page (numbered as "ii")
- ✓ Acknowledgment page
- ✓ Table of contents (all material which follows this page should be listed)
- ✓ Abstract (one page)
- ✓ Introduction
- ✓ Main body
- ✓ Conclusion
- ✓ References (Bibliography)
- ✓ Appendices.

2. Report Guidelines

- ✓ Photographs, charts, graphs, Symbols and Other Illustrations.
- ✓ Selecting a Typeface (consitent in the use of typeface).
- ✓ Type Size (Limit of 12 characters and spaces per inch of basic text).
- ✓ Font Selection (simple and standard font: Times New Roman, 11 size).
- ✓ Margins: 1-1/2" from the left edge of the page
- ✓ 1" from the top, right, and bottom edges of the page.
- ✓ Pagination (The title page is counted in determining the total number of pages in the report).
- ✓ Body of the Manuscript
- ✓ On Figures and Tables (Table numbers and titles should be placed at the top of the table, Figure numbers and legends should appear at the bottom of figures)
- ✓ Style and Format (*The Elements of Style,* by William Strunk, Jr., and E.B. White).
- ✓ Captions on illustrative material.
- ✓ Typeface Quality(Laser Printer used).
- ✓ Limit of 12 characters and spaces per inch of basic text.

- ✓ Punctuation (Leave one space after a comma).
- ✓ Footnotes and Endnotes
- ✓ Report Presentation (Report must be professionally binded).

VII. Glossary:

In this glossary, we list major technical and non-technical terms and/or words found with explanations and a brief dictionary.

A. Non-Technical Glossary

1. Color Chat

Color Chat is very simple example of public web chat application written in Go and JavaScript, just by using standard package to handle WebSocket. Each user having an ID colored differently than the other (Source: https://github.com/otiai10/colorchat)

2. Public Chatting for all

A public channel chat app for everyone like discord is a chat apps meaning that everyone can search and join as well as decide to participate in, or at least see, relevant conversations, while crucially also allowing people to ignore messages that aren't relevant to them. (Source: https://zapier.com/blog/best-team-chat-app/#google)

3. Servername/ Hostname

In the Internet, a hostname is a domain name assigned to a host computer. This is usually a combination of the host's local name with its parent domain's name. (Source: https://en.wikipedia.org/wiki/Hostname)

4. IP address

An Internet Protocol address (IP address) is a numerical label that is assigned to a device connected to a computer network that uses the Internet Protocol for communication. (Source: https://simple.wikipedia.org/wiki/IP_address)

5. Port number

A port or port number is a number assigned to uniquely identify a connection endpoint and to direct data to a specific service (Source: https://en.wikipedia.org/wiki/Port_(computer_networking))

6. RSA

RSA (Rivest–Shamir–Adleman) is a public-key cryptosystem, one of the oldest widely used for secure data transmission. In a public-key cryptosystem, the encryption key is public and distinct from the decryption key, which is kept secret (private). An RSA user creates and publishes a public key based on two large prime numbers, along with an auxiliary value. The prime numbers are kept secret. Messages

can be encrypted by anyone, via the public key, but can only be decrypted by someone who knows the private key. (Source: https://en.wikipedia.org/wiki/RSA_(cryptosystem))

7. SSH

The Secure Shell Protocol (SSH) is a cryptographicnetwork protocol for operating network services securely over an unsecured network. Its most notable applications are remote login and command-lineexecution. (Source: https://en.wikipedia.org/wiki/Secure_Shell)

8. Public key

A cryptographic key that can be obtained and used by anyone to encrypt messages intended for a particular recipient, such that the encrypted messages can be deciphered only by using a second key that is known only to the recipient (the private key). (Source: Oxford Languages Dictionnary)

9. Private Key

A private key, also known as a secret key, is a mathematical key (kept secret by the holder) used to create digital signatures and, depending on the algorithm, to decrypt messages or files encrypted (for confidentiality) with the corresponding public key. (Source: https://utimaco.com/service/knowledge-base/keys-secrets-management/private-key)

B. Technical Glossary

1. TablePlus

TablePlus is a Modern, native, and friendly GUI tool for relational databases: MySQL, PostgreSQL, SQLite & more. (Source: https://tableplus.com)

2. MySQL

MySQL is the world's most popular open source database. According to DB-Engines, MySQL ranks as the second-most-popular database, behind Oracle Database. MySQL powers many of the most accessed applications, including Facebook, Twitter, Netflix, Uber, Airbnb, Shopify, and Booking.com. (Source: https://www.oracle.com/mysql/what-is-mysql/)

3. PHP

PHP is a popular general-purpose scripting language that is especially suited to web development. Fast, flexible and pragmatic, PHP powers everything from your blog to the most popular websites in the world. (Source: https://www.php.net)

4. JavaScript

JavaScript , often abbreviated as "JS", is a high-level, dynamic, untyped, interpreted runtime language. (Source: https://en.wikipedia.org/wiki/JavaScript).

5. jQuery

jQuery is a cross-platform JavaScript library designed to simplify the client-side scripting of HTML. (Source: https://en.wikipedia.org/wiki/JQuery).

6. Frontend

Front end, often referred to as the client side, is the part of a website or web application that users interact with directly. It includes everything that users see or use to interact with a device or software including layout design, text, images, buttons, and navigation menus. (Source: https://roadmap.sh/frontend)

7. Backend

The back end refers to parts of a computer application or a program's code that allow it to operate and that cannot be accessed by a user. Most data and operating syntax are stored and accessed in the back end of a computer system. Typically the code is comprised of one or more programming languages. (Source: https://www.techtarget.com/whatis/definition/front-end)

8. HTML

HyperText Markup Language or HTML is the standard markup language for documents designed to be displayed in web defining the structure of web content. (Source:https://en.wikipedia.org/wiki/HTML)

9. CSS

Cascading Style Sheets is a style sheet language used for specifying the presentation and styling of a document written in a markup language such as HTML or XML. CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript. (Source: https://en.wikipedia.org/wiki/CSS)

10. Bootstrap

Bootstrap is a free and open-source CSS framework directed at responsive, mobile-first front-end web development containing HTML, CSS and JavaScript design templates for forms, buttons, navigation and other interface components. (Source: https://en.wikipedia.org/wiki/Bootstrap_(front-end_framework))

11. Xampp

XAMPP is a completely free, easy to install Apache distribution containing MariaDB, PHP, and Perl. The XAMPP open source package has been set up to be incredibly easy to install and to use. (Source: https://www.apachefriends.org)

12. Apache

The Apache HTTP Server Project is an effort to develop and maintain an open-source HTTP server for modern operating systems including UNIX and Windows. The goal of this project is to provide a secure, efficient and extensible server that provides HTTP services in sync with the current HTTP standards. (Source: https://httpd.apache.org)

VIII. Why Color Chat is distinguished?

This web public chat application is built using network and security technologies unlike most chat app. Experimenting with network and security features in the chat app was a crucial learning experience allowing me to mix different course topics into one project, this case

web (frontend and backend) being the outer topic covering other course learning topics such as network and security. I also learned new ways of implementing chat application outside my university teachings. There are a lot of chat applications on the web but none of them used security login keys generation and network ip port use for communication in a public chat.
Here are the list of differences between my web app and other web apps:

	Color Chat	Other Chat applications
Backend environment	PHP, jQuery	Node.js, Django, Python, Java
Frontend environment	Html, CSS, JavaScript	React, Angular, Dart
Database	Relational	Non-relational
Communication	Servers used	Sockets.io
Authentication Credentials	Stored by sessions and databases	Stored in simple text files
Modern Frameworks	jQuery	AngularJS
Platform	Public	Private
Security features	Keys generations and RSA	None
Loading Time	Less than 5 seconds.	More than 20 seconds.
Real Time capabilities	Milliseconds	Seconds

IX. About Color Chat:

Color Chat is chat messaging application that allows you to communicate with colors instead of words resulting to a scene full of colors based on our choice of colors. This chat app also have theme to choose between light, cream and dark and this was simple and very unique for the original color chat application which u can fing in app store color chat app. In addition, people can join anytime however it is not public room chatting but private. (Source: https://www.google.com/url?sa=t&source=web&rct=j&opi=89978449&url=https://apps.apple.com/us/app/color-chat-chat-with-colors/id1057712795&ved=2ahUKEwj5uYDsmc-FAxXmdqQEHaudBUoQFnoECBMQAQ&usg=AOvVaw0xu-nuU8WsMbS3ajt6u3Gz).

As for the Play Store app which is a different color chat app, this is is public where anyone can join on any channel to chat on even choosing their own text background messaging by playing with RGBA which i also found rather unique and very interesting not only public channel for everyone but also decide your own text background color. (Source: https://play.google.com/store/apps/details?id=com.codingal.curriculum.colorchat&hl=en&gl=US).

My colorchat web app name originated from these two app inspiring me for my capstone project but my colorchat app I added some of my own personal touch making it public like the play store and adding choice between light and dark theme inspired from the app store app aside from those I added what I learned in the mix by adding network's ip and port number as for security adding public and private keys.

X. Web Chat App Features:

RSA Authentication:

The web chat application colorchat uses RSA bits algorithm for the private key generations it uses as type SSL RSA built in method in php as well as the number of bits for RSA that the user has chosen in this case 1024 or 2048 or 3072 or 4096. The difference is that the first one is 1024 bits long and doesn't have a great encryption strength where the 4096 is very strong for encryption but it comes at a cost of CPU use.

Keys Generation:

Color Chat also generates key pairs and saves them inside the Documents folder as pem file for the public key and p12 file for the private key. However, it generates them using openssl functions in php encrypting them using the public key and hashing them using SHA256 algorithm to sign them using the private key as well as saving both keys in the database.

Port Use:

This web chat application makes use of port checking. But first the user has to choose the port he or she wants to use. The application then verifies if the port is used or not by executing a shell command if it is used it will not allow the user to choose it

Public Channel Social Network Chat App:

The chat application also posesses chat message sending and display messages on public platform by simply not mentioning messages being publiv or private php takes it by default as public with the use of sessions and databases to store message and send messages and retrieve messages via ajax jquery function i implemented.

Dark/Light Theme:

I added a feature in the login process where the user gets to choose whether he or she wants to activate dark mode for the page by using the toggle dark mode method if picked since light mode is already implemented by default. Implementind dark mode checks for local storage using toggle switch event listener.

XI. Web Chat App Architecture:

In this brief section, I will describe the architecture of the chat application “Color Chat”.

Frontend:

1. Storage

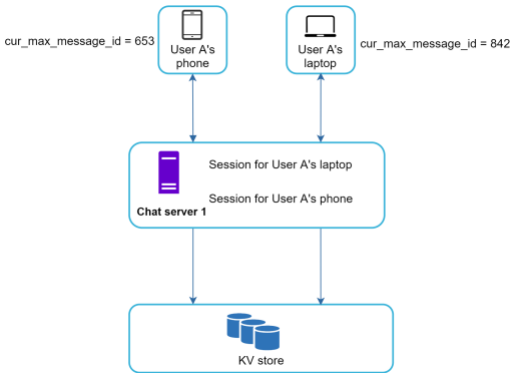
The client side sotrage consists of Javascript APIs which are used to store data then retrieve them from client side such as persisting previous site activities where the contents of login data is stored in sessions and remembering if the user was logged in before. Or personalizing site preferences such as showing user’s ligt or dark theme.

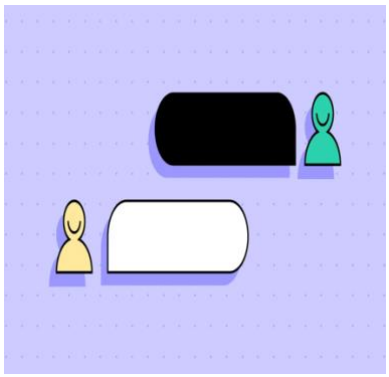
2. Robust Security features

Frontend security deals with protecting user interface and focuses on mitigating security risks such as using http with SSL or TLS to protect user credential login informations or using HTTPS also SSH

3. User Experience

The frontend user experience originates mainly from the CSS side where web design takes it all and gives the best or worse user experience depending ont the design of a web Chat App without forgetting the structure of the website which plays some part int the user experience with the use of Html. Giving the user the option of choice too such as the theme helps impove the user’s experience and bootstrap.





Backend:

1. Database

The database forming Color Chat is in fact MySQL which is a relational database loaded in Table Plus which is very similar to phpmyadmin but it offers the choice between all database type I chose to use MariaDB for my database. I used a total of 3 tables one for the user login, another for the configuration(IP, port...) and the last one for the chat messages with the use of foreign key too. MariaDB is a fork from MySQL since it is more scalabale and offers higher query speed than MySQL’s. it makes use of database using php files with php configuration file giving it permission to add new data entry.

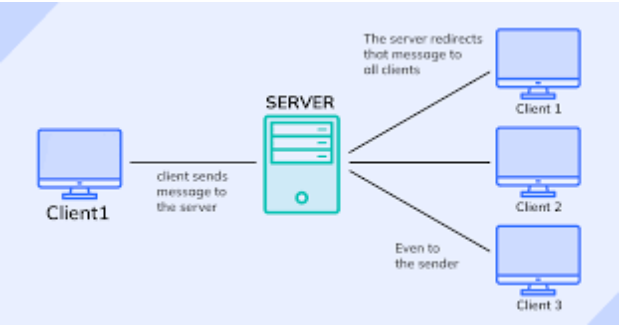
2. User Authentication

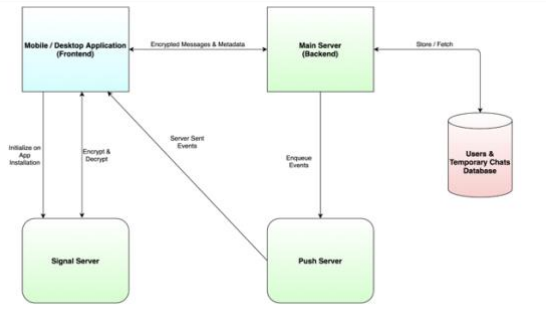
User Authentication in the backend happens through php verifying all the login data entered passes all the tests and is suited for use in any case it will send encode or decode messages in the form of JSON to

notify the user about the process and giving access or not. In addition the use of sessions also help the user get access immediately without having to login again by the use of user ids. Futhermore the use of keypairs, ssh functions and password hash verification help the authentication through php.

3. Fast network facilities

With the use of servers comes at the cost of network access without network, the web page may not load as for message sending it will likely not work offlin even to this day people are struggling to make chatting withou internet access. As for servers, Apache's web server and My SQL database servers are both needed for the website to function. As for chat messages displayed, i put it so that every 2 seconds at maximum it refreshes the chat for other users chatting online. As for performance the use of Maria DB helps in performance and fast responses from the server too.



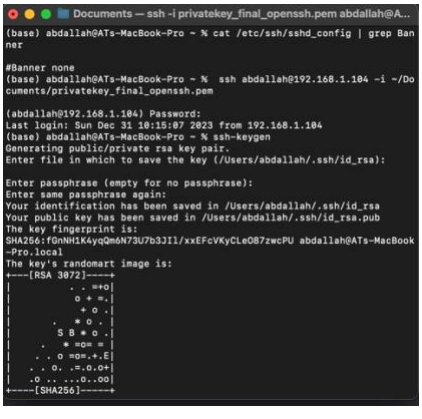


XII. Web Chat App Algorithm:

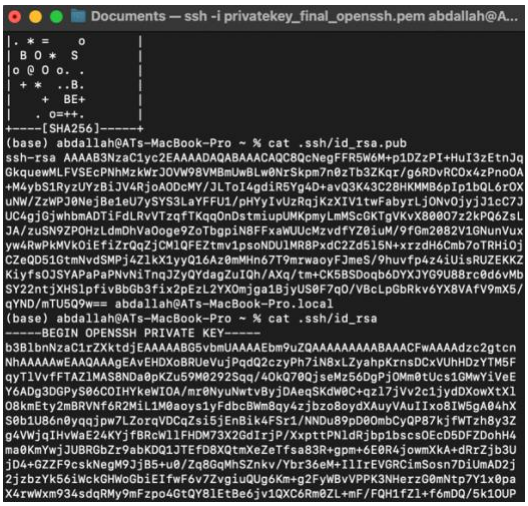
This was the hardest challenge I faced which is the method and the plan used to implement this web chat application. The web chat application algorithm takes into account the security keys generation and using them to encrpyt and decrypt for the login as well as public chat channel by implementing send and get functions (inspired by Faysal Imtiyaz: https://github.com/faysalimtyaz1991/chat_system_youtube)

But first some looks at the orginal login using terminal method:

The methods uses at first creates a private key file then usese the hostname and the ip to specify the file configuration then generates key pair after that choosing the rsa bit algorithm with the use of hashing method SHA256 and RSA 3072 bits for more security.



I printed the public and private key with the rsa 3072 bits method, it worked but was an unnecessary step.



I then logged in using ssh, the hostname and IP address which logged me in and worked with password.

In addition these are the step I followed for writing rsa key generation for the new private key file.

```
(base) abdallah@ATs-MacBook-Pro Documents % openssl rsa -in privatekey.
pem -outform pem > privatekey_final_openssh.pem

writing RSA key
```

And this is how I transformed p12 file to pem file for private key which I didn't do for my algorithm

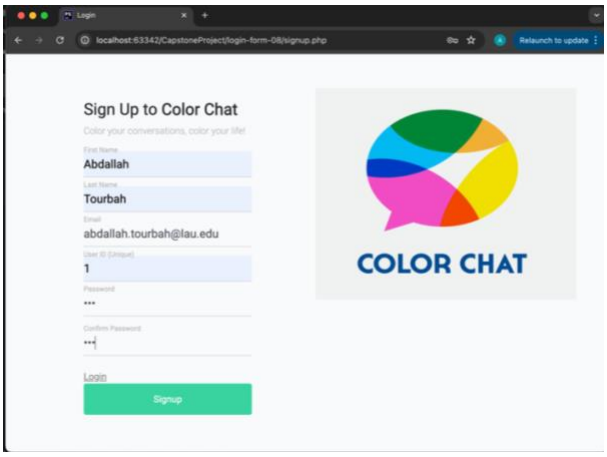
```
(base) abdallah@ATs-MacBook-Pro Documents % openssl pkcs12 -in privatek
ey.p12 -out privatekey.pem -nocerts -nodes

Enter Import Password:
(base) abdallah@ATs-MacBook-Pro Documents % puttygen privatekey.pem -O
private-openssh -o privatekey_final.pem
```

As for my web chat app algorithm that I used:

1- Users start on the login page if the users didn't login previously or are new to webchat application they should head to the sign up page for registering.
2- The users mainly start in the sign up page where they have to enter their firstname, lastname, email, user id, password and confirm password.
3- After successful signup the users then is redirected to login where they enter the user id they chose and the password they entered.
4- If they forgot their password there is the option to reset password by getting a code they have to enter sent by mail and the users then can modify their password successfully.
5- Then the users will reach followed up configuration page where they have to enter their own hostname which is the folder usually named under the pc name or "Users" folder or their hostname or can be seen by opening the terminal and see their hostname. Anyways, the user will also have to add the ip address which is the wifi's ip address they are using and they are asked as well to choose a port number which is available the users will get notified if the port number is busy then the user chooses the rsa algorithm bits security as well as the option between light and dark mode.
6- Then the users will be directed to the main chat app page where they can type and press enter, the chat history will show their message to everyone who logged in and reached this point by showing the ip, port number and the public key under details
7- After enjoying the talk with everyone else the users can logout and be redirected to login page.

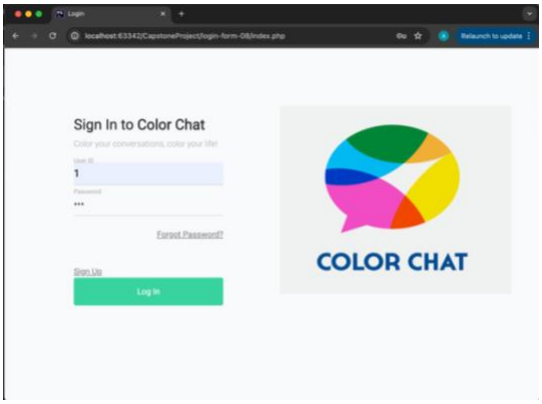
XIII. Web Chat App Design:

My chat application is divided into 4 main pages but 2 categories: the login and the main chat page. Starting off with the signup page the moment you want to join as a new user. The page looks like this:

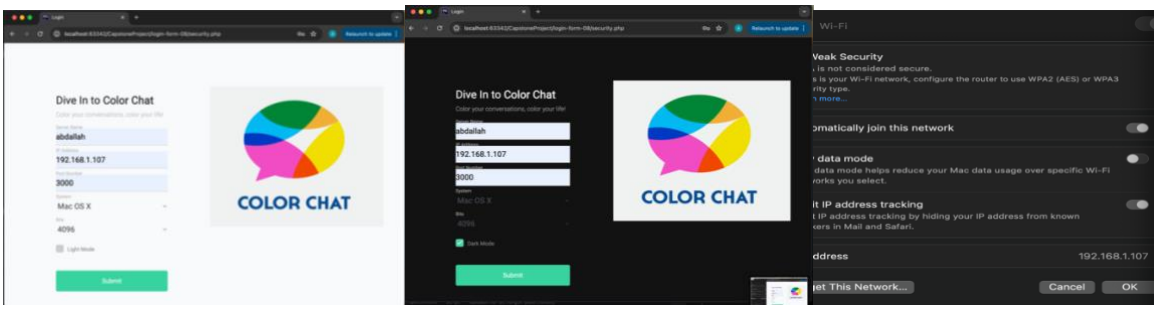


The signup page will show you the color chat logo on the right. The signup page also has a title body marked in bold indicating it is a signup and a slogan quote under the title to make the users days a happier one followed. Users will have to fill the information if the user id is chosen, the user will have to choose another one but users have to make sure the email they entered is theirs in case the users forget their password. As for both password fields users will have to make sure they are both the same and they have to make sure to remember it as well as the id they chose. In case the user headed to signup but

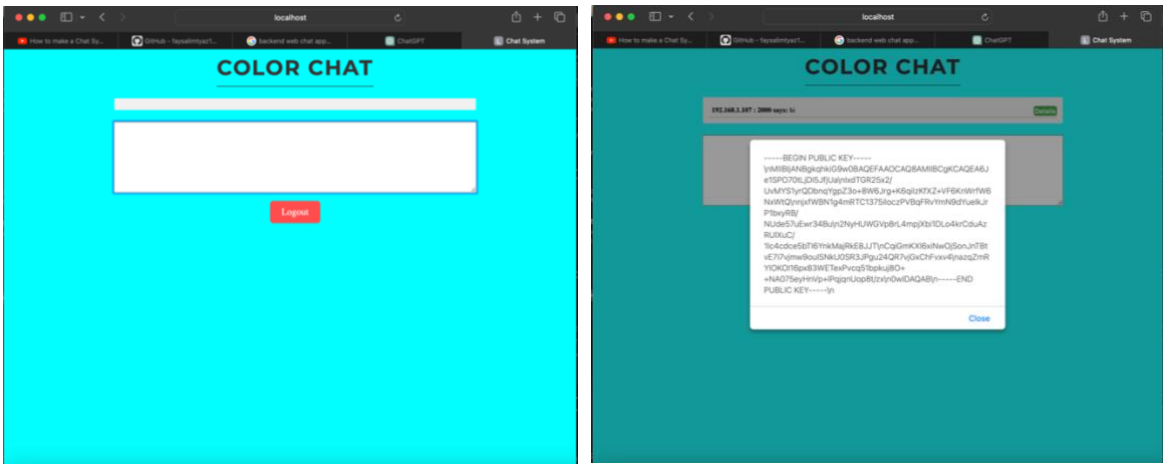
remembered they already signed up before they can head to “Login” link which will bring them to login page. After all, when the users click on “Signup” button, they will receive an alert showing successful signup or the errors the user made when signing up. Now for my login page:



The users will have to enter their user id and password to type. In case the user forgot password they can click on “Forgot Password?” which will lead them to a prompt to type their email then a random code will be sent to them by mail which they will use to put to the code verification prompt if successful they can reset or modify their password. Then in case the user hasn’t signed up the “Sign Up” link will lead them to the signup page. At the end after entering user id and password, the users will receive alert if login was successful or show an error message. In case they got successful they will proceed to the security or configuration page:

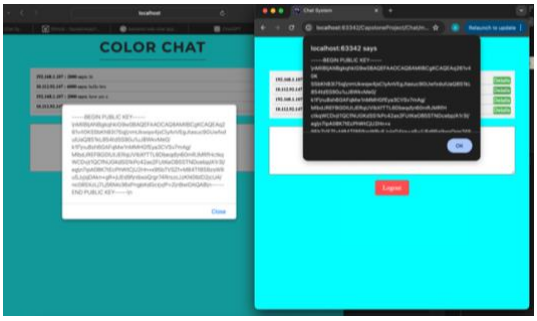


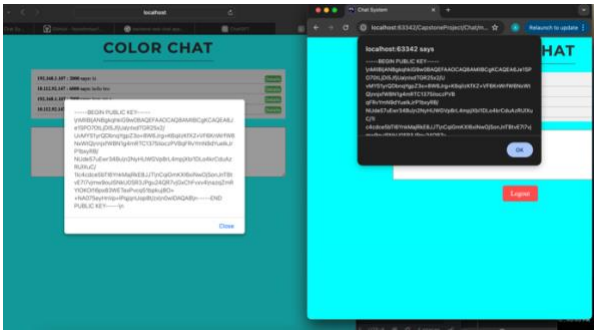
In the security page, the users will have to dive and search to input their hostname or servername as well as the IP of theirs and the port they want to communicate on if it is available. They can view their IP addresses by checking wifi settings details. As for the device operating system used needs to be specified as well as the number of bits for rsa algorithm for security purposes by the users themselves. They can also choose whether to activate dark mode or stay on lightmode. Now moving to Chat page:

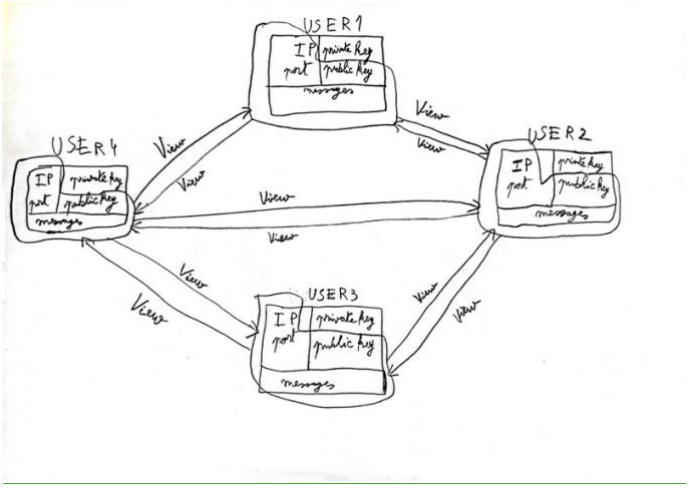


In this page there is the web title chat system meaning we finished the login process. You can see up the name of my application well designed as well as the cyan background giving a special sensation to the users. Anyways, there is 2 sections in this page the first one is the chat history where all previous messages show, in this section you will notice previous users including you with ip port and public key when you press on “Details” button using other chatters public key and ur public key u can encrypt and sign the message. As for the section under it is where you type and press enter to send message. After you enjoy comfortable group chat, you press on “Logout” button to close your session leading to login page.

XIV. Biased Color Chat User View:



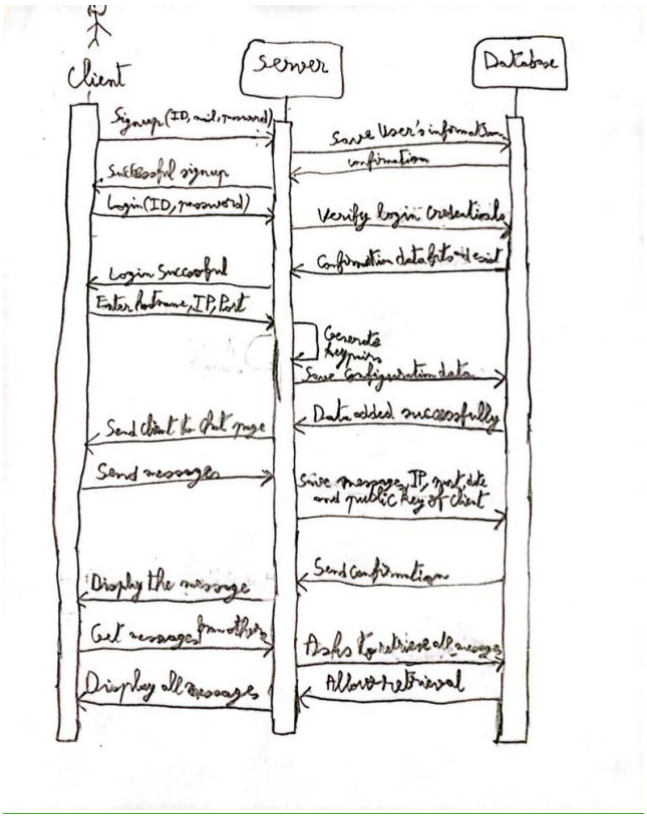




Based on the pictures above, you can clearly notice that each user can view their own IP, port, messages and public key as well as other users IP, port, messages and public key as shown in the figure drawn above. In addition each user if they can look clearly, the public key is not the same for everyone, it is different for every user same for private key which can't be viewed unless users head to the p12 file where it is saved. Consequently, with the same logic, all the following has to be changed to fit the view as well:

- Users Name (Which includes the IP and port number each user is using).
- Messages (user's messages alongside other users messages)
- Users public keys

N.B: I forgot to add arrow signs around each user with view telling users they can view their own.

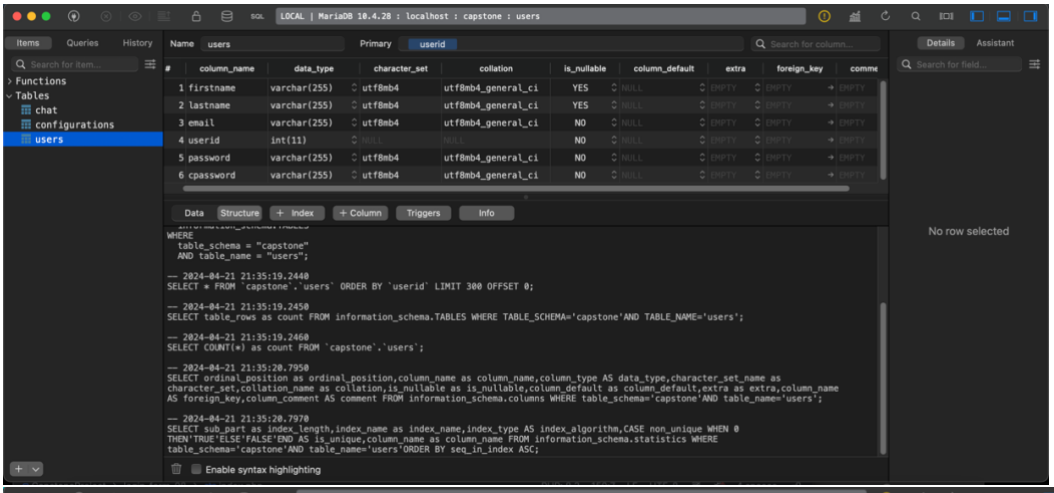


Then I showed a clear sequence diagram illustrating the users detailed way they will go through in my chat application (Honorable memories from software engineering course. The client requests from the server the signup, login, messages sent and receieved but the server always goes to access the database table to add and verify informations in the database tables which send acceptance or rejection request to the server delivering error messages as form of alerts from the server to client.

XV. Tables:

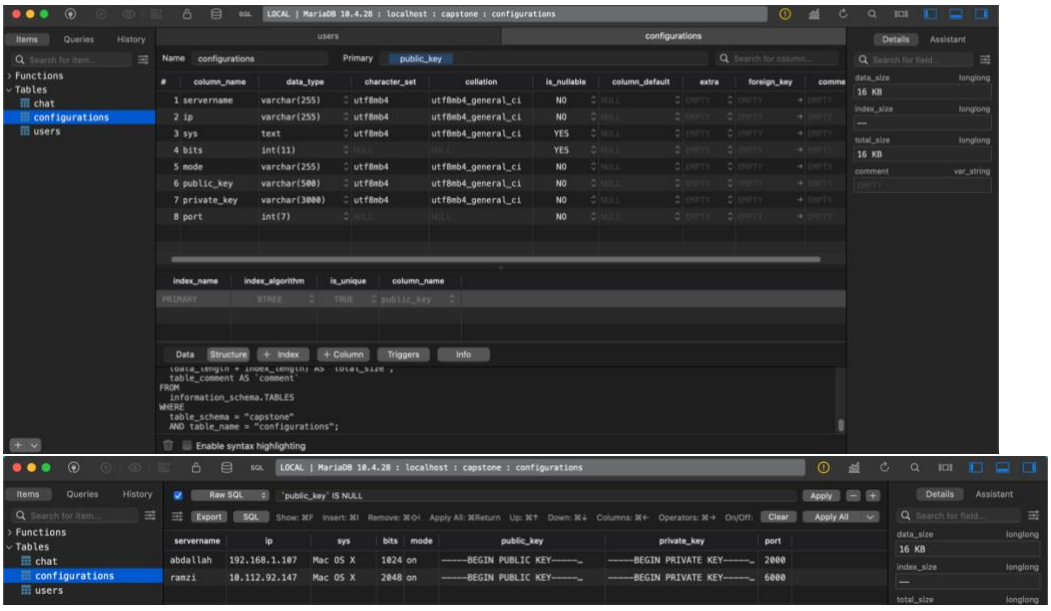
Database tables are the most important part in my chat application because without them the application will only have pages that are pretty but won't work. Every table is stored remote on a MariaDB remote sql database stored in Table Plus.

signup.sql:



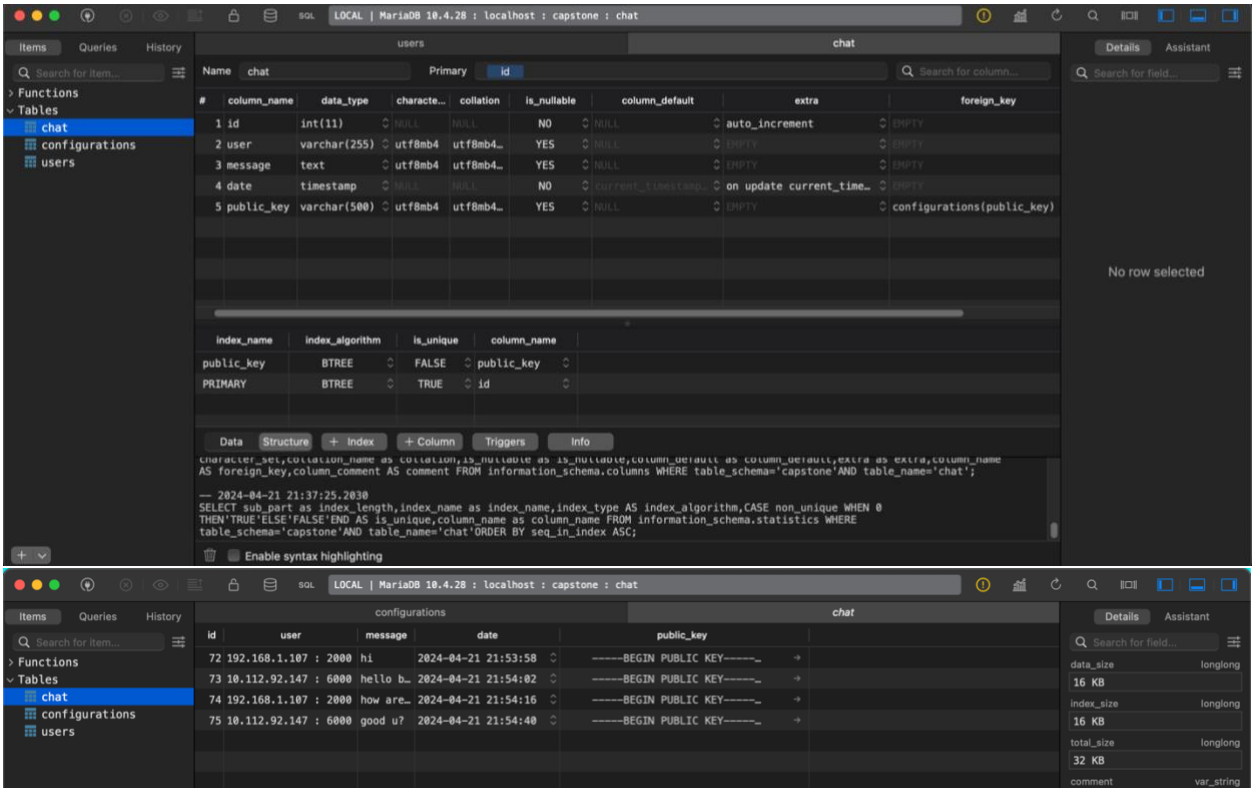
The first table represents the signup database table with userid being the primary key where only firstname and lastname are allowed to be NULL since I will work using IP and port instead. In the data of that table the password is clearly hashed using SHA256 method.

configurations.sql:



The second table relies on ip, port, servername, bits, mode, public key, private key and port number with the public key being the primary key since it is very unique for each individual. I also chose it to be the public key for reason I will explain in. the third table. Only bits and system are allowed to be null the rest can't be null although the mode is a checkbox only for light and dark theme this can be null but it is considered on or off since it is a checkbox. The user doesn't have to worry about public and private key since the system will generate for the user.

chat.sql:



In my last table for the chatting, this is how the database handles messages from users. The primary key this time being the message id which will auto increment everytime for example it started with 72 in this case because the previous one I erased them manually so they disappeared for testing purposes, after all testing is the most important step I got taught from software engineering course. Anyways, I said previously why I made the public key primary key that is the other reason, because I wanted to use it as foreign key in this table which worked by extracting it from the configurations table primary key I added a column for public key here and linked it via foreign key. Additionally other than message I chose the user to be as the form "ip : port" in the source code ajax file. As for the date I put it current timestamp showing the day and time I could have displayed it next to each message but to keep it simple I didn't and since the messages are sent one after the other there was no need to display the date in the chat application.

XVI. System Evolution:

The system empowering Color Chat felt efficient and reliable, as to the question what can make

This project even more complete are the following:

- Adding cloud system suchas AWS to the equation to rely less on servers and have the chat application serverless.
- The light and dark mode could have showed up in the main chat page where the chatting is.
- Add more colors or let the user chooses the rgb filter they want to apply on the background of the chat page.
- Fixing the forgot password which almost worked but I tried my best as suggested by Dr. Haraty.
- Adding a search bar for the ip or port or message search in case the message was so far up and the user couldn't find it.

These were the last modifications that could have been made to improve Color Chat.

XVII. Source Code:

- Login Folder:

dbconf.php: (configuration for the database)

```php
<?php
global $pdo;
$host = '127.0.0.1';
$port = '3306';
$dbname = 'capstone';
$username = 'root';
$password = '';

try {
    $pdo = new PDO("mysql:host=$host;port=$port;dbname=$dbname;charset=utf8mb4", $username,
$password);

    // Set PDO to throw exceptions on error
    $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

    // Optionally, set the default fetch mode to fetch associative arrays
    $pdo->setAttribute(PDO::ATTR_DEFAULT_FETCH_MODE, PDO::FETCH_ASSOC);
} catch (PDOException $e) {
    // If connection fails, display an error message
    die("Connection failed: " . $e->getMessage());
}
?>
```

signup.php:

```php
<!doctype html>
<html lang="en">
<head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <link href="https://fonts.googleapis.com/css?family=Roboto:300,400&display=swap"
```

```
rel="stylesheet">
    <link rel="stylesheet" href="fonts/icomoon/style.css">
    <link rel="stylesheet" href="css/owl.carousel.min.css">

    <!-- Bootstrap CSS -->
    <link rel="stylesheet" href="css/bootstrap.min.css">

    <!-- Style -->
    <link rel="stylesheet" href="css/style.css">

    <title>Login</title>
    <!-- Dark Mode CSS (Sample, customize as needed) -->
    <style id="dark-mode-style">
        body.theme-dark {
            background-color: #121212;
            color: #ffffff;
        }

        .navbar.theme-dark {
            background-color: #333333;
        }

    </style>
</head>
<body>

<div class="content">
    <div class="container">
        <div class="row">
            <div class="col-md-6 order-md-2">
                <img src="images/colorchat.jpeg" alt="Image" class="img-fluid">
            </div>
            <div class="col-md-6 contents">
                <div class="row justify-content-center">
                    <div class="col-md-8">
                        <div class="mb-4">
                            <h3>Sign Up to <strong>Color Chat</strong></h3>
                            <p class="mb-4"> Color your conversations, color your life!</p>
                        </div>
                        <form action="#" method="post">
                            <div class="form-group first">
                                <label for="firstname">First Name</label>
                                <input type="text" class="form-control" id="firstname" name="firstname">

                            </div>
                            <div class="form-group first">
                                <label for="lastname">Last Name</label>
                                <input type="text" class="form-control" id="lastname" name="lastname">

                            </div>
                            <div class="form-group first">
```

```
                        <label for="email">Email</label>
                        <input type="text" class="form-control" id="email" name="email">

                    </div>
                    <div class="form-group first">
                        <label for="userid">User ID (Unique)</label>
                        <input type="number" min="1" class="form-control" id="userid" name="userid">
                    </div>

                    <div class="form-group last mb-4">
                      <label for="password">Password</label>
                      <input type="password" class="form-control" id="password" name="password">

                    </div>
                    <div class="form-group last mb-4">
                        <label for="cpassword">Confirm Password</label>
                        <input type="password" class="form-control" id="cpassword" name="cpassword">

                    </div>


                    <span class="ml-auto"><a href="#" class="login">Login</a></span>

                    <input type="submit" value="Signup" class="btn text-white btn-block btn-primary"
id="Signup" name="Signup">
                  </form>
                </div>
              </div>

          </div>

        </div>
    </div>
</div>


<script src="js/jquery-3.3.1.min.js"></script>
<script src="js/popper.min.js"></script>
<script src="js/bootstrap.min.js"></script>
<script src="js/main.js"></script>
<!-- Include jQuery -->
<script src="https://code.jquery.com/jquery-3.3.1.min.js"></script>

<!-- Font Awesome Icons -->
<script src="https://kit.fontawesome.com/3a03770c9c.js" crossorigin="anonymous"></script>

<!-- Dark Mode Script -->
<script>

    $('#Signup').click(function(event) {
        event.preventDefault();
```

```
        var formData = $('#login-form').serialize()+ '&firstname=' + $('#firstname').val() + '&lastname=' +
$('#lastname').val() + '&email=' + $('#email').val()+ '&userid=' + $('#userid').val()+ '&password=' +
$('#password').val()+ '&cpassword=' + $('#cpassword').val();
        $.ajax({
            type: 'POST',
            url: 'dbsignup.php',
            data: formData,
            dataType: 'json',
            success: function(response) {

                if (response.success) {
                    alert('Sign Up Successful:\n You can login now');
                    window.location.href = 'index.php';
                } else {
                    alert('Error: ' + response.error);
                }
            },
            error: function(error) {
                console.error(error);
                // Handle errors
            }
        });
    });
</script>
<script>
    // Add a click event listener to the Sign Up link
    $('.login').click(function() {
        // Redirect the user to signup.php
        window.location.href = 'index.php';
    });
</script>

</body>
</html>
```

dbsignup.php:

```
<?php
global $pdo;
session_start();
require_once 'dbconf.php';
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // Check if all required form fields are set
    if (!isset($_POST["firstname"]) || !isset($_POST["lastname"]) || !isset($_POST["email"]) ||
!isset($_POST["userid"]) || !isset($_POST["password"]) || !isset($_POST["cpassword"])) {
        echo json_encode(array("error" => "One or more required fields are missing."));
        exit;
    }

    // Get user input
    $firstname = $_POST["firstname"];
    $lastname = $_POST["lastname"];
```

```php
    $email = $_POST["email"];
    $userid = $_POST["userid"];
    $password = $_POST["password"];
    $cpassword = $_POST["cpassword"];

    // Check if password matches confirm password
    if ($password != $cpassword) {
        echo json_encode(array("error" => "Password and confirm password do not match."));
        exit;
    }

    // Check if the user ID is unique
    $query = "SELECT * FROM users WHERE userid = :userid";
    // Use prepared statements to prevent SQL injection
    $stmt = $pdo->prepare($query);
    $stmt->bindParam(":userid", $userid);
    $stmt->execute();

    if ($stmt->rowCount() > 0) {
        // User ID already exists
        echo json_encode(array("error" => "User ID is already chosen by someone else. Please choose a
different User ID."));
    } else {
        // User ID is unique, proceed with registration
        // Hash the password before saving to the database for security
        if (!empty($password)) {
            $hashedPassword = password_hash($password, PASSWORD_DEFAULT);
        } else {
            echo json_encode(array("error" => "Password cannot be empty."));
            exit;
        }

        // Insert user information into the database
        $insertQuery = "INSERT INTO users (firstname, lastname, email, userid, password, cpassword)
VALUES (:firstname, :lastname, :email, :userid, :password, :cpassword)";
        $insertStmt = $pdo->prepare($insertQuery);
        $insertStmt->bindParam(":firstname", $firstname);
        $insertStmt->bindParam(":lastname", $lastname);
        $insertStmt->bindParam(":email", $email);
        $insertStmt->bindParam(":userid", $userid);
        $insertStmt->bindParam(":password", $hashedPassword);
        $insertStmt->bindParam(":cpassword", $hashedPassword);

        if ($insertStmt->execute()) {
            // Registration successful
            echo json_encode(array("success" => true));
        } else {
            // Registration failed
            echo json_encode(array("error" => "Failed to register user."));
        }
    }
```

```
} else {
    // Handle the case where the script is accessed directly without a POST request
    echo json_encode(array("error" => "Invalid request method."));
}
?>
```

index.php: (Login page)

```
<?php
session_start();
if(isset($_SESSION['userid'])){
    header("location: ../Chat/main.php");
}
?>
<!doctype html>
<html lang="en">
  <head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <link href="https://fonts.googleapis.com/css?family=Roboto:300,400&display=swap"
rel="stylesheet">
    <link rel="stylesheet" href="fonts/icomoon/style.css">
    <link rel="stylesheet" href="css/owl.carousel.min.css">

    <!-- Bootstrap CSS -->
    <link rel="stylesheet" href="css/bootstrap.min.css">

    <!-- Style -->
    <link rel="stylesheet" href="css/style.css">

    <title>Login</title>
    <!-- Dark Mode CSS (Sample, customize as needed) -->
    <style id="dark-mode-style">
      body.theme-dark {
        background-color: #121212;
        color: #ffffff;
      }

      .navbar.theme-dark {
        background-color: #333333;
      }

    </style>
  </head>
  <body>

  <div class="content">
    <div class="container">
      <div class="row">
        <div class="col-md-6 order-md-2">
          <img src="images/colorchat.jpeg" alt="Image" class="img-fluid">
```

```
      </div>
      <div class="col-md-6 contents">
        <div class="row justify-content-center">
          <div class="col-md-8">
            <div class="mb-4">
            <h3>Sign In to <strong>Color Chat</strong></h3>
            <p class="mb-4"> Color your conversations, color your life!</p>
          </div>
          <form action="#" method="post">
            <div class="form-group first">
              <label for="userid">User ID</label>
              <input type="text" class="form-control" id="userid" name="userid">

            </div>
              <div class="form-group last mb-4">
                 <label for="password">Password</label>
                 <input type="password" class="form-control" id="password">

              </div>

              <div class="d-flex mb-5 align-items-center">
                 <span class="ml-auto"><a href="#" class="forgot-pass">Forgot Password?</a></span>
              </div>

            <span class="ml-auto"><a href="#" class="sign-up">Sign Up</a></span>

            <input type="submit" value="Log In" class="btn text-white btn-block btn-primary" id="login"
name="login">
          </form>
        </div>

      </div>

    </div>
  </div>
</div>

  <script src="js/jquery-3.3.1.min.js"></script>
  <script src="js/popper.min.js"></script>
  <script src="js/bootstrap.min.js"></script>
  <script src="js/main.js"></script>
  <!-- Include jQuery -->
  <script src="https://code.jquery.com/jquery-3.3.1.min.js"></script>

<!-- Font Awesome Icons -->
<script src="https://kit.fontawesome.com/3a03770c9c.js" crossorigin="anonymous"></script>
<script>
  $('#login').click(function(event) {
     event.preventDefault();
     var formData = $('#login-form').serialize()+ '&userid=' + $('#userid').val() + '&password=' +
$('#password').val();
```

```
    $.ajax({
      type: 'POST',
      url: 'dblogin.php',
      data: formData,
      dataType: 'json',
      success: function(response) {
        if (response.success) {
          alert('Login Successful!\n Generate keys now ^_^');
          window.location.href = 'security.php';
        } else {
          alert('Error: ' + response.error);
        }
      },
      error: function(error) {
        console.error(error);
        // Handle errors
      }
    });
  });
</script>
<script>
    // Add a click event listener to the Sign Up link
    $('.sign-up').click(function() {
        // Redirect the user to signup.php
        window.location.href = 'signup.php';
    });

    $('.forgot-pass').click(function() {
        // Prompt the user to enter their email
        var userEmail = prompt('Enter your email:');

        // Check if the user entered an email
        if (userEmail) {
            // Now you do something with the user's email, such as sending a reset password link

            $.ajax({
                url: 'reset_password.php',
                method: 'POST',
                data: {email: userEmail},
                success: function (response) {
                    // Handle the success response, e.g., show a message to the user
                    alert('Password reset email sent successfully.');
                    // Prompt the user to enter the OTP code
                    var userCode = prompt('Enter the OTP code sent to your email:');

                    // Check if the user entered an OTP code
                    if (userCode) {
                        // Now you can verify the OTP code entered by the user
                        // Make another AJAX request to your server to verify the OTP code
                        // Replace the placeholder URL with the actual endpoint on your server
                        $.ajax({
```

```
                    url: 'verify_otp.php',
                    method: 'POST',
                    data: { email: userEmail, code: userCode },
                    success: function(verificationResponse) {
                        // Handle the verification response from the server
                        if (verificationResponse === 'success') {
                            // If the OTP code is correct, proceed to the next page
                            window.location.href = 'update_password.php';
                        } else {
                            // If the OTP code is incorrect, show an error message
                            alert('Incorrect OTP code. Please try again.');
                        }
                    },
                    error: function(error) {
                        // Handle the error response from the server
                        alert('Failed to verify OTP code. Please try again.');
                    }
                });
            }
        },
        error: function (error) {
            // Handle the error, show an error message
            alert('Failed to send password reset email. Please try again.');
        }
    });
  }
});
</script>
</body>
</html>
```

dblogin.php:

```
<?php
session_start();
require_once 'dbconf.php'; // Include database configuration file

$userid = isset($_POST['userid']) ? $_POST['userid'] : '';
$password = isset($_POST['password']) ? $_POST['password'] : '';

// Validate input
if (empty($userid) || empty($password)) {
    echo json_encode(array("error" => "Please provide both user ID and password."));
    exit;
}

// Connect to the database
try {
    $pdo = new PDO("mysql:host=127.0.0.1;port=3306;dbname=capstone;charset=utf8mb4", "root", "");
    $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
} catch (PDOException $e) {
    echo json_encode(array("error" => "Database connection error: " . $e->getMessage()));
```

```
    exit;
}

// Prepare and execute the query to retrieve user credentials
$query = "SELECT * FROM users WHERE userid = :userid";
$stmt = $pdo->prepare($query);
$stmt->bindParam(":userid", $userid);
$stmt->execute();

// Fetch the user record
$userRecord = $stmt->fetch(PDO::FETCH_ASSOC);

if ($userRecord) {
    // User found, compare the hashed password
    $hashedPasswordInDatabase = $userRecord['password'];

    // Use password_verify to check if the entered password matches the stored hash
    if (password_verify($password, $hashedPasswordInDatabase)) {
        $_SESSION['userid']=$userid;
        echo json_encode(array("success" => true));}
    } else {
    echo json_encode(array("error" => "User not found. Please Signup"));
}
?>
```

reset_password.php: (forgot password)

```
<?php
session_start();
global $pdo;
require_once 'dbconf.php'; // Include database configuration file
require_once 'PHPMailer/PHPMailer.php';
require_once 'PHPMailer/Exception.php';

use PHPMailer\PHPMailer\PHPMailer;
use PHPMailer\PHPMailer\Exception;

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // Get the user's email from the POST data
    $userEmail = $_POST["email"];
    // Perform server-side validation on the email address if needed

    // Check if the user email exists in the database
    $query = "SELECT * FROM users WHERE email = :email";
    $stmt = $pdo->prepare($query);
    $stmt->bindParam(":email", $userEmail);
    $stmt->execute();
    $user = $stmt->fetch(PDO::FETCH_ASSOC);

    if ($user) {
        // Generate a unique OTP code
        $otpCode = generateOTP();
```

```
        // Store the OTP code in the session
        $_SESSION['otp'][$userEmail] = $otpCode;

        // Send the OTP code via email
        $subject = "Password Reset OTP";
        $message = "Your OTP code for password reset is: $otpCode";

        try {
            sendEmail($userEmail, $subject, $message);
            echo json_encode(array("status" => "success"));
        } catch (Exception $e) {
            echo json_encode(array("status" => "error", "message" => "Failed to send the OTP code."));
        }
    } else {
        // User with the provided email does not exist
        echo json_encode(array("status" => "error", "message" => "User not found."));
    }
} else {
    // Handle the case where the script is accessed directly without a POST request
    echo json_encode(array("status" => "error", "message" => "Invalid request method."));
}

function generateOTP() {
    // Generate a random OTP code (6 digits)
    return mt_rand(100000, 999999);
}

function sendEmail($recipient, $subject, $message) {
    // Create a PHPMailer instance
    $mail = new PHPMailer(true);

    // SMTP configuration
    $mail->isSMTP();
    $mail->Host = 'pod51013.outlook.com';//'live.smtp.mailtrap.io'; // LAU SMTP server address
    $mail->SMTPAuth = true;
    $mail->Username = 'abdallah.tourbah@lau.edu';//'api'; // SMTP username
    $mail->Password = 'Wonderland123+';//'bd8b714ec9336eee8d9cf095a03d221e'; // SMTP password
    $mail->SMTPSecure = 'tls';
    $mail->Port = 587;

    // Sender and recipient settings
    $mail->setFrom('abdallah.tourbah@lau.edu', 'ColorChat');
    $mail->addAddress($recipient);

    // Email content
    $mail->isHTML(true);
    $mail->Subject = $subject;
    $mail->Body = $message;

    // Send the email
```

```
    $mail->send();
}
?>
```

verify_otp.php: (forgot password)

```
<?php
session_start();
global $pdo;
require_once 'dbconf.php';

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // Get the user's email and OTP code from the POST data
    $userEmail = $_POST["email"];
    $userCode = $_POST["code"];

    // Validate the email and OTP code

    // Retrieve the OTP code stored in the session
    $storedCode = isset($_SESSION['otp'][$userEmail]) ? $_SESSION['otp'][$userEmail] : '';

    // Compare the user-entered OTP code with the stored OTP code
    if ($userCode === $storedCode) {
        // OTP code is correct
        echo json_encode(array("status" => "success"));
    } else {
        // OTP code is incorrect
        echo json_encode(array("status" => "error", "message" => "Code entered is not the same."));
    }
} else {
    // Handle the case where the script is accessed directly without a POST request
    echo json_encode(array("status" => "error"));
}
?>
```

update_password.php: (forgot password)

```
<?php
session_start();
global $pdo;
if(isset($_POST['change-pass'])) {
    // Check if the password field is not empty
    if(!empty($_POST['Pass'])) {
        // Sanitize and validate the new password
        $newPassword = $_POST['Pass'];
        // Connect to the database
        require_once 'dbconf.php';

        // Retrieve the user ID from the session
        $userId = isset($_SESSION['userid']) ? $_SESSION['userid'] : '';

        // Update the password in the database
```

```
        $query = "UPDATE users SET password = :password WHERE userid = :userid";
        $stmt = $pdo->prepare($query);
        $hashedPassword = password_hash($newPassword, PASSWORD_DEFAULT);
        $stmt->bindParam(':password', $newPassword);
        $stmt->bindParam(':userid', $userId);

        if($stmt->execute()) {
            // Password updated successfully, redirect the user to index.php
            header("Location: index.php");
            exit;
        } else {
            // Error occurred while updating the password
            echo "Error: Failed to update the password.";
        }
    } else {
        // Password field is empty
        echo "Error: Password field cannot be empty.";
    }
}
?>
<!doctype html>
<html lang="en">
  <head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <link href="https://fonts.googleapis.com/css?family=Roboto:300,400&display=swap"
rel="stylesheet">
    <link rel="stylesheet" href="fonts/icomoon/style.css">
    <link rel="stylesheet" href="css/owl.carousel.min.css">

    <!-- Bootstrap CSS -->
    <link rel="stylesheet" href="css/bootstrap.min.css">

    <!-- Style -->
    <link rel="stylesheet" href="css/style.css">

    <title>Login</title>
    <!-- Dark Mode CSS (Sample, customize as needed) -->
    <style id="dark-mode-style">
    body.theme-dark {
    background-color: #121212;
        color: #ffffff;
      }

      .navbar.theme-dark {
    background-color: #333333;
      }

    </style>
  </head>
```

```
  <body>

  <div class="content">
    <div class="container">
      <div class="row">
        <div class="col-md-6 order-md-2">
          <img src="images/colorchat.jpeg" alt="Image" class="img-fluid">
        </div>
        <div class="col-md-6 contents">
          <div class="row justify-content-center">
            <div class="col-md-8">
              <div class="mb-4">
              <h3>Modify Password In to <strong>Color Chat</strong></h3>
              <p class="mb-4"> Color your conversations, color your life!</p>
            </div>
            <form action="#" method="post">
                <div class="form-group last mb-4">
                    <label for="Pass">Password</label>
                    <input type="password" class="form-control" id="Pass" name="Pass">

                </div>

              <input type="submit" value="Change Password" class="btn text-white btn-block btn-primary"
id="change-pass" name="change-pass">
            </form>
          </div>

        </div>

      </div>
    </div>
  </div>



    <script src="js/jquery-3.3.1.min.js"></script>
    <script src="js/popper.min.js"></script>
    <script src="js/bootstrap.min.js"></script>
    <script src="js/main.js"></script>
    <!-- Include jQuery -->
    <script src="https://code.jquery.com/jquery-3.3.1.min.js"></script>

  <!-- Font Awesome Icons -->
  <script src="https://kit.fontawesome.com/3a03770c9c.js" crossorigin="anonymous"></script>
  </div>
  </body>
</html>
```

Security.php: (configurations page)

```
<!doctype html>
<html lang="en">
<head>
```

```
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <link href="https://fonts.googleapis.com/css?family=Roboto:300,400&display=swap"
rel="stylesheet">
    <link rel="stylesheet" href="fonts/icomoon/style.css">
    <link rel="stylesheet" href="css/owl.carousel.min.css">

    <!-- Bootstrap CSS -->
    <link rel="stylesheet" href="css/bootstrap.min.css">

    <!-- Style -->
    <link rel="stylesheet" href="css/style.css">

    <title>Login</title>
    <!-- Dark Mode CSS (Sample, customize as needed) -->
    <style id="dark-mode-style">
        body.theme-dark {
            background-color: #121212;
            color: #ffffff;
        }

        .navbar.theme-dark {
            background-color: #333333;
        }

    </style>
</head>
<body>

<div class="content">
    <div class="container">
        <div class="row">
            <div class="col-md-6 order-md-2">
                <img src="images/colorchat.jpeg" alt="Image" class="img-fluid">
            </div>
            <div class="col-md-6 contents">
                <div class="row justify-content-center">
                    <div class="col-md-8">
                        <div class="mb-4">
                            <h3>Dive In to <strong>Color Chat</strong></h3>
                            <p class="mb-4"> Color your conversations, color your life!</p>
                        </div>
                        <form action="#" method="post">
                            <div class="form-group first">
                                <label for="servername">Server Name</label>
                                <input type="text" class="form-control" id="servername" name="servername">

                            </div>
                            <div class="form-group first">
                                <label for="ip">IP Address</label>
```

```
                    <input type="text" class="form-control" id="ip" name="ip">
                </div>
                <div class="form-group first">
                    <label for="port">Port Number</label>
                    <input type="text" class="form-control" id="port" name="port">
                </div>

                <div class="form-group first">
                    <label for="sys">System</label>
                    <select class="form-control" id="sys" name="sys">
                        <option value=""></option>
                        <option value="Windows">Windows</option>
                        <option value="Mac OS X">Mac OS X</option>
                        <option value="Linux">Linux</option>
                    </select>

                </div>
                <div class="form-group first">
                    <label for="bits">Bits</label>
                    <select class="form-control" id="bits" name="bits">
                        <option value=""></option>
                        <option value="1024">1024</option>
                        <option value="2048">2048</option>
                        <option value="3072">3072</option>
                        <option value="4096">4096</option>
                    </select>
                </div>

                <div class="d-flex mb-5 align-items-center">
                    <label class="control control--checkbox mb-0">
                        <span class="caption" id="toggle-icon">
                            <span class="toggle-text">
                                Light Mode
                            </span>
                        </span>
                        <input type="checkbox" id="mode" name="mode"/>
                        <div class="control__indicator"></div>
                    </label>
                </div>



                <input type="submit" value="Submit" class="btn text-white btn-block btn-primary"
id="Submit" name="Submit">

            </form>
        </div>
    </div>

</div>
```

```
      </div>
    </div>
</div>


<script src="js/jquery-3.3.1.min.js"></script>
<script src="js/popper.min.js"></script>
<script src="js/bootstrap.min.js"></script>
<script src="js/main.js"></script>
<!-- Include jQuery -->
<script src="https://code.jquery.com/jquery-3.3.1.min.js"></script>

<!-- Font Awesome Icons -->
<script src="https://kit.fontawesome.com/3a03770c9c.js" crossorigin="anonymous"></script>

<!-- Dark Mode Script -->
<script>
    //const toggleSwitch = document.querySelector('input[type="checkbox"]');
    const toggleSwitch = document.getElementById('mode');
    const toggleIcon = document.getElementById('toggle-icon');
    const darkModeStyle = document.getElementById('dark-mode-style');

    // Function to toggle dark mode
    function toggleDarkMode() {
        document.body.classList.toggle('theme-dark');
        darkModeStyle.innerHTML = document.body.classList.contains('theme-dark')
            ? `
            body.theme-dark {
              background-color: #121212;
              color: #ffffff;
            }

            .navbar.theme-dark {
              background-color: #333333;
            }

          `
            : '';
        // Change text and icon
        toggleIcon.children[0].textContent = document.body.classList.contains('theme-dark') ? 'Dark Mode' :
'Light Mode';
        if (toggleIcon.children[1]) {
            toggleIcon.children[1].classList.toggle('fa-sun');
            toggleIcon.children[1].classList.toggle('fa-moon');
        }
    }

    // Event Listener for the switch
    toggleSwitch.addEventListener('change', toggleDarkMode);

    // Check Local Storage For Theme
```

```
    const currentTheme = localStorage.getItem('theme');
    if (currentTheme === 'dark') {
        toggleSwitch.checked = true;
        toggleDarkMode();
    }


    $('#Submit').click(function(event) {
        event.preventDefault();
        var formData = $('#login-form').serialize()+ '&servername=' + $('#servername').val() + '&ip=' +
$('#ip').val() + '&port=' + $('#port').val() + '&bits=' + $('#bits').val()+ '&mode=' + $('#mode').val()+
'&sys=' + $('#sys').val();
        $.ajax({
            type: 'POST',
            url: 'generate_key.php',
            data: formData,
            dataType: 'json',
            success: function(response) {

                if (response.success) {
                    alert('Keys Generated Successfully:\n You can chat now ^_^');
                    window.location.href = '../Chat/main.php';
                } else {
                    alert('Error: ' + response.error);
                }
            },
            error: function(error) {
                console.error(error);

            }
        });
    });
</script>
</body>
</html>
```

generate_key.php: (configurations backend file)

```
<?php
session_start();
global $pdo;
require_once 'dbconf.php';

if (isset($_SESSION['userid'])) {
    // Get form data
    $servername = isset($_POST['servername']) ? $_POST['servername'] : '';
    $ip = isset($_POST['ip']) ? $_POST['ip'] : '';
    $port = isset($_POST['port']) ? $_POST['port'] : '';
    $sys = isset($_POST['sys']) ? $_POST['sys'] : '';
    $bits = isset($_POST['bits']) ? intval($_POST['bits']) : 2048;
    $mode = isset($_POST['mode']) ? $_POST['mode'] : '';
```

```php
// Validate form data
if ($servername === '') {
    echo json_encode(['error' => 'Server Name is empty']);
    exit;
}

if ($ip === '') {
    echo json_encode(['error' => 'IP Address is empty']);
    exit;
}

if ($port === '') {
    echo json_encode(['error' => 'Port Number is empty']);
    exit;
}

if ($sys === '') {
    echo json_encode(['error' => 'System is empty']);
    exit;
}

// Validate bits
if ($bits !== 1024 && $bits !== 2048 && $bits !== 3072 && $bits !== 4096) {
    echo json_encode(['error' => 'Invalid bits value']);
    exit;
}

// Check if the port is available
$isPortAvailable = false;
$connection = @fsockopen($ip, $port, $errno, $errstr, 1);
if (!$connection) {
    // Port is available
    $isPortAvailable = true;
} else {
    // Port is not available
    fclose($connection);
}
if (!$isPortAvailable) {
    echo json_encode(['error' => 'Port is busy']);
    exit;
}

if ($sys === 'Mac OS X'||$sys=== 'Windows'||$sys==='Linux') {
    // Check if the port is already in use
    $portInUse = shell_exec("lsof -i :$port");

    if ($portInUse) {
        echo json_encode(['error' => 'Port is already in use']);
        exit;
    }
```

```
        // Check if public key already exists
        $publicKey = ''; // Assign the public key value here
        $query = "SELECT COUNT(*) AS count FROM configurations WHERE public_key =
:public_key";
        $stmt = $pdo->prepare($query);
        $stmt->execute(['public_key' => $publicKey]);
        $result = $stmt->fetch(PDO::FETCH_ASSOC);

        if ($result['count'] > 0) {
            // Public key already exists, do not insert
            echo json_encode(['error' => 'Public key already exists']);
            exit;
        }
        $config = [
            'private_key_bits' => $bits,
            'private_key_type' => OPENSSL_KEYTYPE_RSA,
        ];

        // Create a new private and public key pair
        $res = openssl_pkey_new($config);
        // Get the private key
        openssl_pkey_export($res, $privateKey);

        // Get the public key
        $publicKey = openssl_pkey_get_details($res)['key'];

        file_put_contents('/Users/abdallah/Documents/public_key.pem',$publicKey);
        file_put_contents('/Users/abdallah/Documents/private_key.p12',$privateKey);
        $data = [
            'servername' => $servername,
            'ip' => $ip,
            'sys' => $sys,
            'bits' => $bits,
            'mode' => $mode,
            'public_key' => $publicKey,
            'private_key' => $privateKey,
            'port' => $port
        ];
        $dataToSign = json_encode($data);
        // Encrypt data with the public key
        openssl_public_encrypt($dataToSign, $encryptedData, $publicKey);
        // Sign the data with the private key
        openssl_sign($dataToSign, $signature, $privateKey, OPENSSL_ALGO_SHA256);

        // Insert data into the database
        $query = "INSERT INTO configurations (servername, ip, sys, bits, mode, public_key, private_key,
port)
                VALUES (:servername, :ip, :sys, :bits, :mode, :public_key, :private_key, :port)";

        $stmt = $pdo->prepare($query);
        $stmt->execute($data);
```

```
        $response = [
            'success' => true,
            'public_key' => $publicKey,
            'private_key' => $privateKey,
            'data' => $data,
            'signature' => base64_encode($signature)
        ];
        $_SESSION['ip'] = $ip;
        $_SESSION['port'] = $port;
        $_SESSION['public_key']= $publicKey;
        $_SESSION['private_key']=$privateKey;

        echo json_encode($response);
        exit;

    }
}

echo json_encode(['error' => 'User not authenticated']);
?>
```

- Chat Folder:

main.php:

```
<?php session_start();?>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Chat System</title>


    <link rel="stylesheet" href="style.css" type="text/css" />


    <script src="https://code.jquery.com/jquery-3.3.1.js" crossorigin="anonymous"></script>
    <style>
        body {
            background-color: cyan;
        }

        /* Styles for title */
        .title-container {
            text-align: center;
            margin-bottom: 20px;
        }

        .title {
            font-size: 36px;
            font-weight: bold;
```

```
            color: #333; /* Change color as desired */
            text-transform: uppercase;
            letter-spacing: 2px;
            border-bottom: 1px solid #333; /* Underline effect */
            padding-bottom: 10px;
            display: inline-block;
            margin-bottom: 10px;
            font-family: 'Montserrat', sans-serif;
        }
        /* Style for logout button */
        .logout-btn {
            display: inline-block;
            padding: 10px 20px;
            background-color: #ff4d4d;
            color: #fff;
            text-decoration: none;
            border-radius: 5px;
            transition: background-color 0.3s ease;
        }

        .logout-btn:hover {
            background-color: #e60000;
        }


        /* Style for details button */
        .details-btn{
            position: absolute;
            right: 10px;
            padding: 2px 2px;
            display: inline-block;
            background-color: #4CAF50;
            color: white;
            border: none;
            border-radius: 4px;
            cursor: pointer;
            transition: background-color 0.3s;
        }

        .details-btn:hover {
            background-color: #45a049;
        }
        /* Style for chathistory div */
        .chathistory {
            position: relative;
            /* Add other styles as needed */
        }
    </style>
    <link href="https://fonts.googleapis.com/css2?family=Montserrat:wght@400;700&display=swap"
rel="stylesheet">
    <script>
```

```
        function showDetails(publicKey) {
            alert(publicKey);
        }
    </script>
    <!-- Dark Mode Script -->
    <script>
        $(document).ready(function(){
            // Check if dark mode is enabled
            var darkMode = '<?php echo isset($_SESSION["mode"]) && $_SESSION["mode"] === "off" ? "true" : "false"; ?>';

            // Function to toggle dark mode
            function toggleDarkMode() {
                var body = $('body');
                var navbar = $('.navbar');
                if (darkMode === 'true') {
                    body.addClass('theme-dark');
                    navbar.addClass('theme-dark');
                } else {
                    body.removeClass('theme-dark');
                    navbar.removeClass('theme-dark');
                }
            }

            // Toggle dark mode on page load
            toggleDarkMode();
        });
    </script>
</head>
<body>

<div class="centeralised">
    <div class="title">Color Chat</div>

    <div class="chathistory"></div>

    <div class="chatbox">

        <form action="" method="POST">

            <textarea class="txtarea" id="message" name="message"></textarea>

        </form>

    </div>
    <a href="logout.php" class="logout-btn">Logout</a>

</div>


<script>
```

```
    $(document).ready(function(){
        loadChat();

    });



    $('#message').keyup(function(e){


        var message = $(this).val();

        if( e.which == 13 ){

            $.post('handlers/ajax.php?action=SendMessage&message='+message, function(response){

                loadChat();
                $('#message').val('');

            });

        }

    });


    function loadChat()
    {
        $.post('handlers/ajax.php?action=getChat', function(response){

            $('.chathistory').html(response);

        });
    }


    setInterval(function(){
        loadChat();
    }, 2000);

</script>

</body>
</html>
```

style.css:

```
*{
    padding:0px;
    margin:0px;
}
```

```
.centeralised{
    margin:10px auto;
    width:1000px;
    text-align:center;
}

.chathistory{

    width:600px;
    margin:10px auto;
    padding:10px;
    background:#f1f1f1;
    text-align:left;
}


.txtarea{
    min-height:100px;
    width:600px;
    margin:10px auto;
    padding:10px;
}

.siglemessage{
    font-size:12px;
    padding:5px;
    border-bottom:1px solid #b3b3b3;
}
```

ajax.php: (inside a folder called handlers)

```
<?php
session_start();
include("../../login-form-08/dbconf.php");

if( isset($_REQUEST['action']) ){


    switch( $_REQUEST['action'] ){



        case "SendMessage":


            $query = $pdo->prepare("INSERT INTO chat SET user=?, message=?,public_key=?");

            $query->execute([$_SESSION['ip']. ' : ' . $_SESSION['port'], $_REQUEST['message'],
$_SESSION['public_key']]);

            echo 1;
```

```
        break;



    case "getChat":
        $query = $pdo->prepare("SELECT * from chat");
        $query->execute();

        $rs = $query->fetchAll(PDO::FETCH_OBJ);


        $chat = '';
        foreach( $rs as $r ){
            $publicKey = isset($r->public_key) ? $r->public_key : "N/A";
            // Remove line breaks and escape special characters
            $escapedPublicKey = str_replace(["\n", "\r", '\\', '"'], ['\\n', '\\r', '\\\\', '\\"'], $publicKey);
            $chat .=  '<div class="siglemessage"><strong>'.$r->user.' says:  </strong>'.$r-
>message.'<button class="details-btn" onclick="showDetails(\' . $escapedPublicKey .
'\')">Details</button></div>';

        }

        echo $chat;


        break;



  }


}


?>
```

logout.php:

```
<?php
// Start the session
session_start();

// Destroy the session
session_destroy();

// Redirect to the login page
```

```
header("Location: ../login-form-08/index.php");
exit;
?>
```

As per request of my instructor, the rest of the source code will be sent as a zip file to him. As for my final messages:

Enjoy the project ☺

Color Your Life!

Abdallah.