CDI College

Programmer Analyst, 2023

Report On

Web Programming
Chat Room Application

Author: Saman Khoshroo
Course Instructor: Ms. Kawther Aarizou

Contents

0.1 Introduction

The subject of this internship project is the design and implementation of a real-time chat application; developed to be a tool for students of a class to share notes, discuss ideas and review each others work.

The application has 2 types of users, one person may become the admin and the rest may become regular users who don't have access to admins'tools.

The following technologies were used in the development of this application:

- HTML
- CSS
- JavaScript
- PHP
- AJAX
- MySQL
- Socket.io

0.2 project Description

VCR (Virtual Chat Room) is a real-time chat application designed to help students share notes and discuss ideas in an environment, away from social media.

First, upon launching the app, the user is provided with a login form. The admin and users fill out the same form while selecting the right role. In the case of a new user they may click on the sign up link to get redirected to a different page in order to full out a different form, upon completion of which the user is automatically logged in.

Next, the user arrives at the main part of the application, the home page. It is here where users may see the rooms they have access to and can join them; all the discussions and note sharing happen here. In the case of a new user, they might need to wait for the admin to grant them access to the appropriate rooms first.

Upon logging in, a session is opened and user's information, most notably their username and role are saved as session variables; these information are crucial and are vital components in order for many parts of the application to function; needless to say the ability to logout is easily accessible to users at all times through the nav bar, which automatically destroys the session and redirects the user to index page, where they may fill out the sign in form again.

A simple user, may choose the room they would like to to join; upon joining a room the user can see every other user who is online in that room, all the messages and may sent their own messages as well. If it is the first a user joins a room, they are greeted by a simple greeting message from the system which is not saved in the database.

All the messages are sent to every online user within a room in real-time, and in case of an offline user, the message will be sent as soon as they reconnect; this chat system is the same for the admin, except they have access to extra administrator tools.

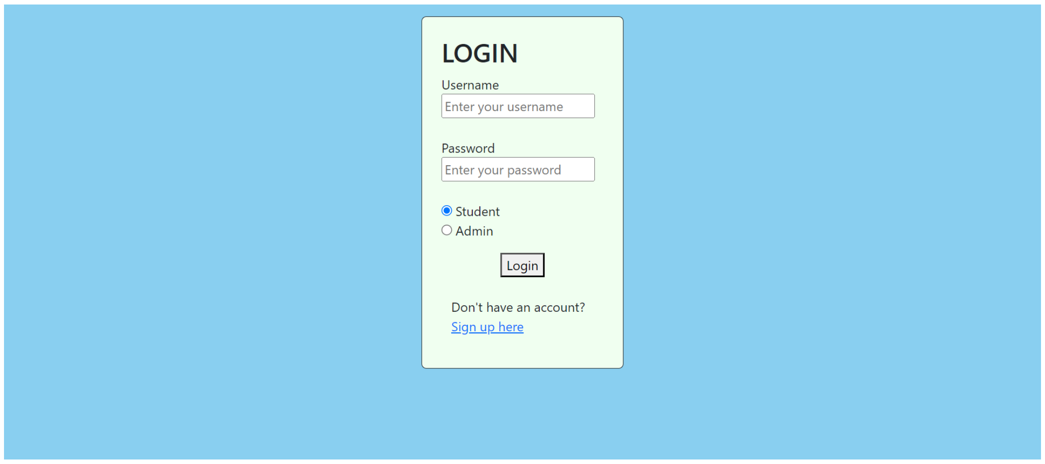


Figure 1: Login Form

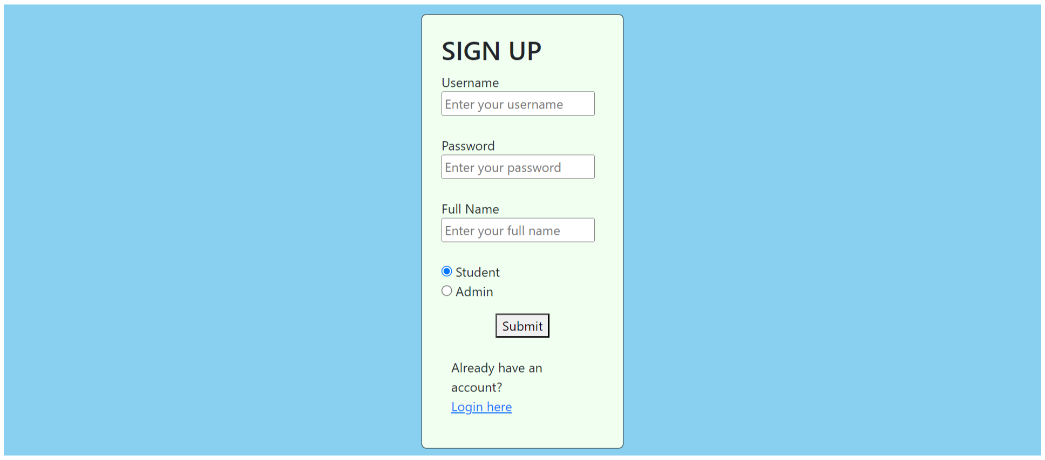


Figure 2: Sign up Form

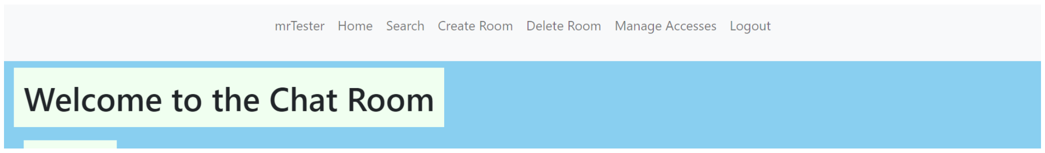


Figure 3: Admin navbar

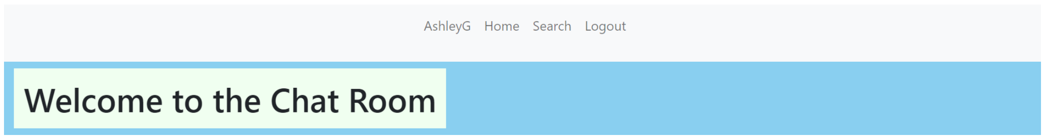


Figure 4: User navbar

While the nav bar for a simple user consists of a link to the search function, which will be explained soon, a logout button and the home button, the admin has access to the following features additionally.

- Create room
- Delete room
- Manage access

The search function, which all users have access to, is a feature that allows the search of sentences, words, or even partial words on the chat history of the rooms which the user currently has access to. Inside the form, user chooses a room from the option select element and types in the search word, then app redirects them to a result page, where they can see all the result messages in a table in great detail (sender, timestamp, message content).

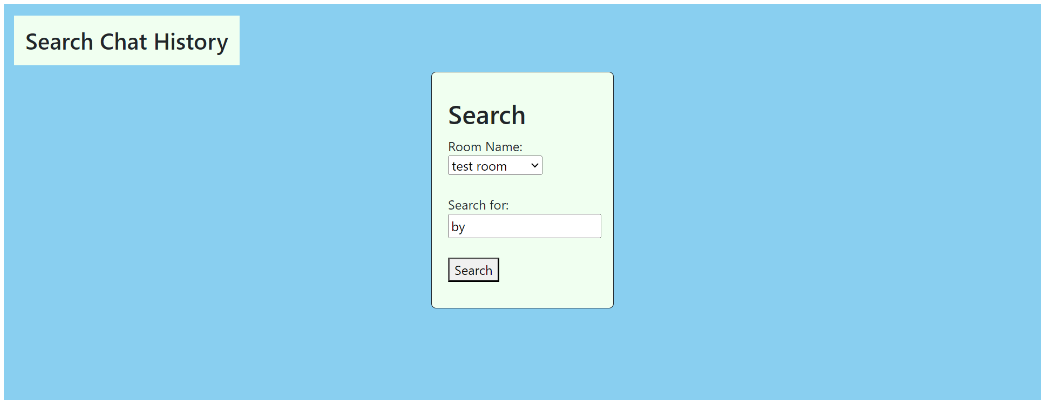
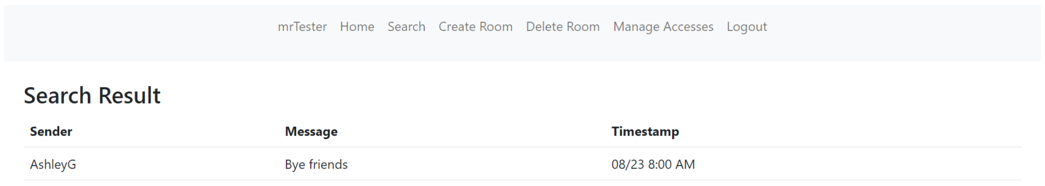


Figure 5: Search form (partial word)


mrTester Home Search Create Room Delete Room Manage Accesses Logout

Search Result

Sender	Message	Timestamp
AshleyG	Bye friends	08/23 8:00 AM

Figure 6: Search Result

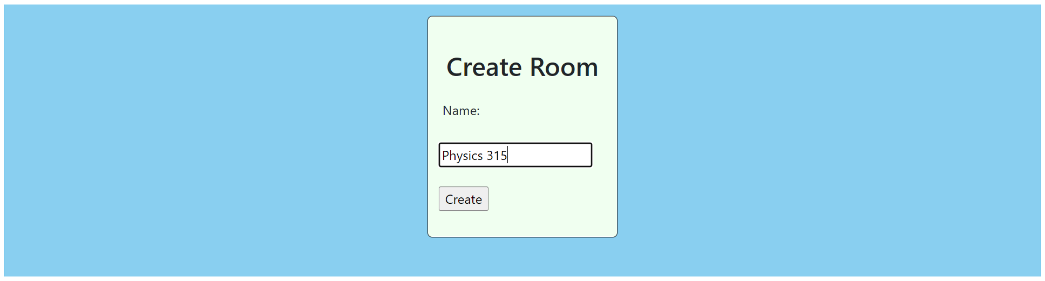
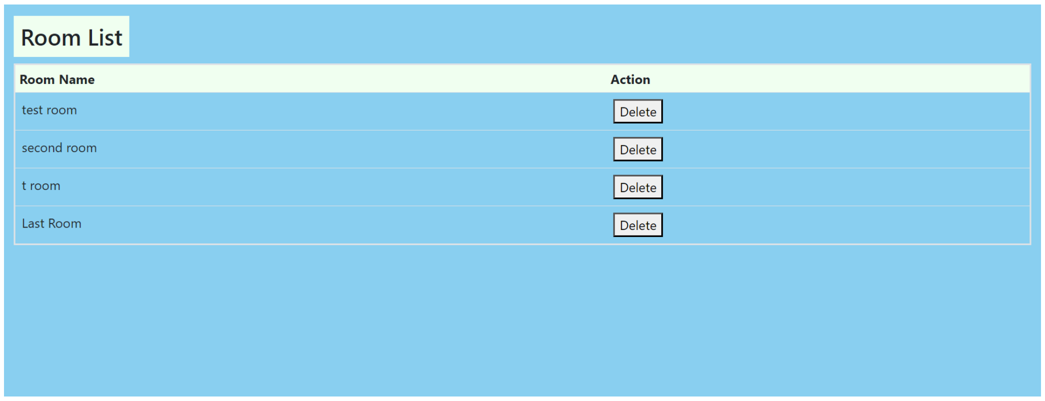


Figure 7: Create room)

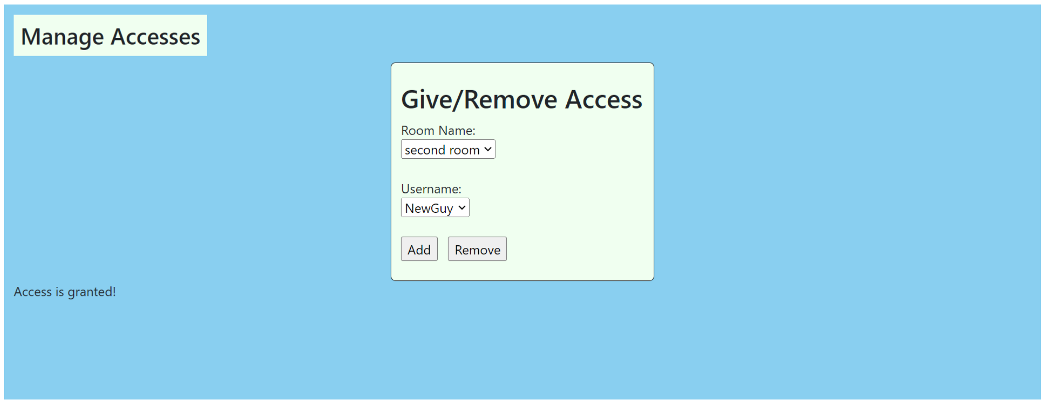


Room Name	Action
test room	Delete
second room	Delete
t room	Delete
Last Room	Delete

Figure 8: Delete room

That concludes the features that are available to all users; the first admin-only feature of VCR is the ability to create or delete rooms. They each have their own respective page and form.

The create room feature is the simpler of the two, admin may create a new room by only typing in the name of their new room, needless to say the app throws an error if a room with the same name currently exists; giving access to the users for a created room happens in a different section.



Access is granted!

Figure 9: Access form

Finally, there's the admin only feature of giving or taking away access from rooms. Similar to most features discussed above, this feature has its own form and page, where the admin chooses a room, and a user from two option select elements and then presses the add (grant access) or remove (take away access) button. A message will confirm the success of the operation.

0.3 My Role and Responsibilities

This is not a group project and all the design and development (front-end and back-end was carried out by a single person. The following is a list that demonstrates most of my responsibilities as a full stack developer.

- Planning
- Creation of algorithm
- Design and implementation of the database
- Development of all the forms
- Development of the interface
- Design and implementation of the socket.io server
- Implementation of proper error handling
- Testing

0.4 Technical Skills Acquired

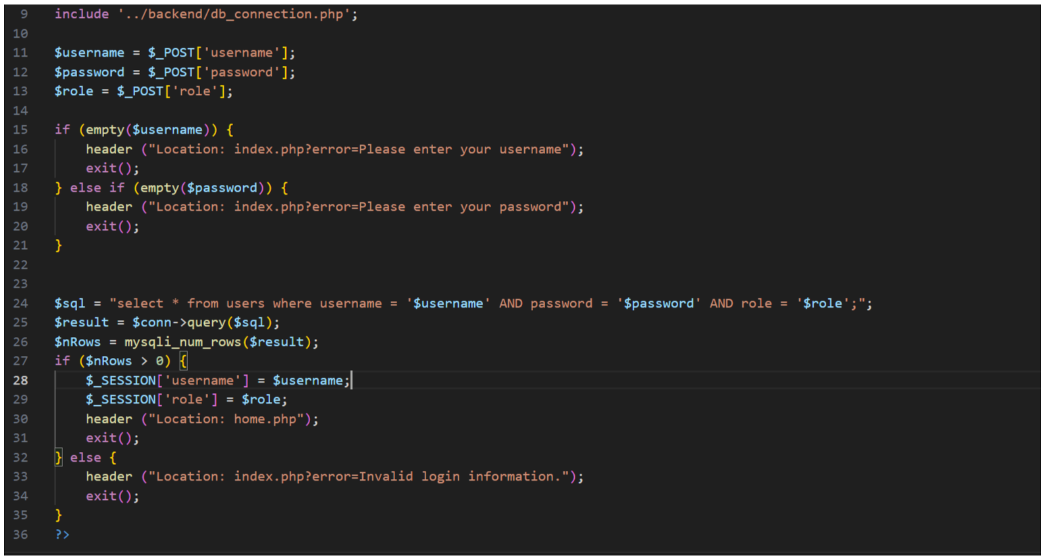
Many technologies and tools were used during the development of this project, most notably HTML, CSS, PHP, JavaScript, MySQL, Ajax and socket.io. Most of these tools I was familiar with, though certainly in many cases, I needed to do extra research and or come up with solutions which I had not seen before during the studies.

First, the HTML; even though, most of the documents have a .php extension, it's impossible to deny the fundamental role of HTML. During the development of VCR, HTML is used to hold the skeleton of the page while PHP and JavaScript gave it life and functionality. In addition, a considerable amount of the libraries, scripts (both JS and PHP), as well as CSS styles are embedded among the HTML tags.

Next, the CSS; this tool is used both directly and indirectly throughout the app, but always serves only one purpose, making every element more graceful and presentable to the client.

The indirect use of CSS is through bootstrap 5; although the syntax is somewhat different, this library still incorporates CSS, but facilitates and fastens the use of CSS. Bootstrap 5 was an excellent choice for the development of VCR mainly because it made much quicker to style the pages on the fly, though for the final version of the app, and more detailed styling, regular inline CSS styles were used.

Simple CSS is used through every page that a user can access; it mainly modifies the background color, margin, padding and position of elements.

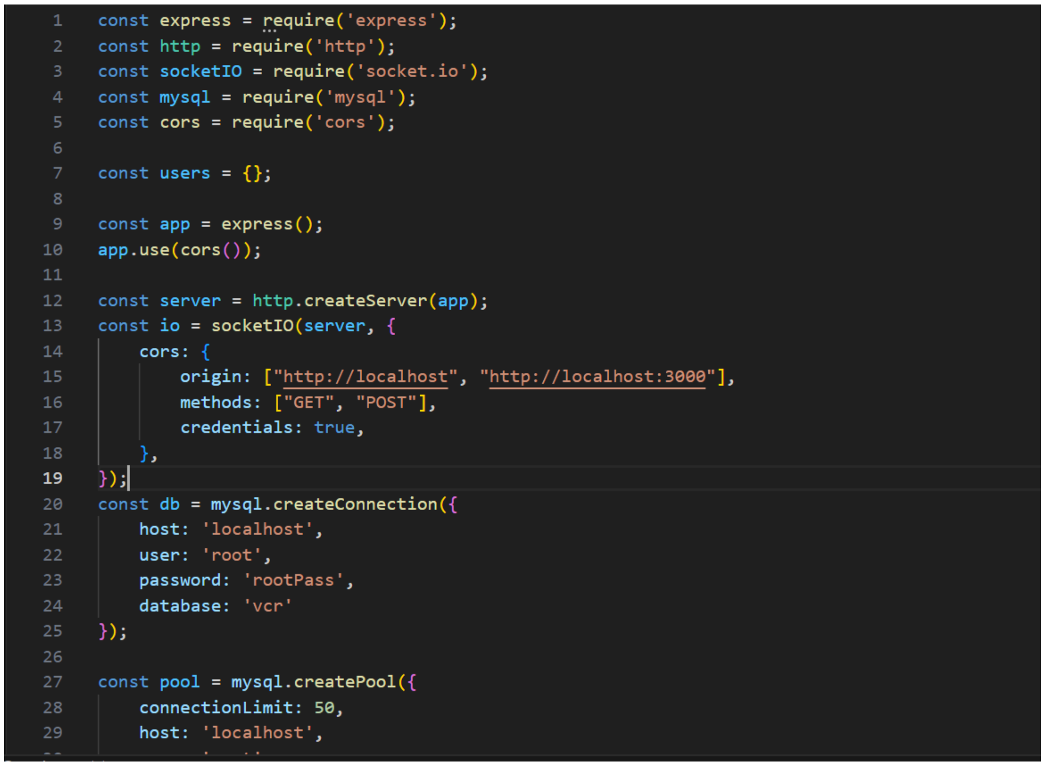


```php
include '../backend/db_connection.php';

$username = $_POST['username'];
$password = $_POST['password'];
$role = $_POST['role'];

if (empty($username)) {
    header ("Location: index.php?error=Please enter your username");
    exit();
} else if (empty($password)) {
    header ("Location: index.php?error=Please enter your password");
    exit();
}


$sql = "select * from users where username = '$username' AND password = '$password' AND role = '$role';";
$result = $conn->query($sql);
$nRows = mysqli_num_rows($result);
if ($nRows > 0) {
    $_SESSION['username'] = $username;
    $_SESSION['role'] = $role;
    header ("Location: home.php");
    exit();
} else {
    header ("Location: index.php?error=Invalid login information.");
    exit();
}
?>
```

Figure 10: Checking user credentials and storing session variables

Third, the PHP. This language is perhaps the biggest part of VCR, not surprisingly considering the amount of files with the .php extension at their tail. The following list demonstrates what PHP was used for during the development of this app.

- Navigation
- Storing user's information and session
- Interacting with the database
- Error handling
- Storing server information
- Connecting the front-end and back-end parts of the app
- Security and checking user authority
- Vital functions



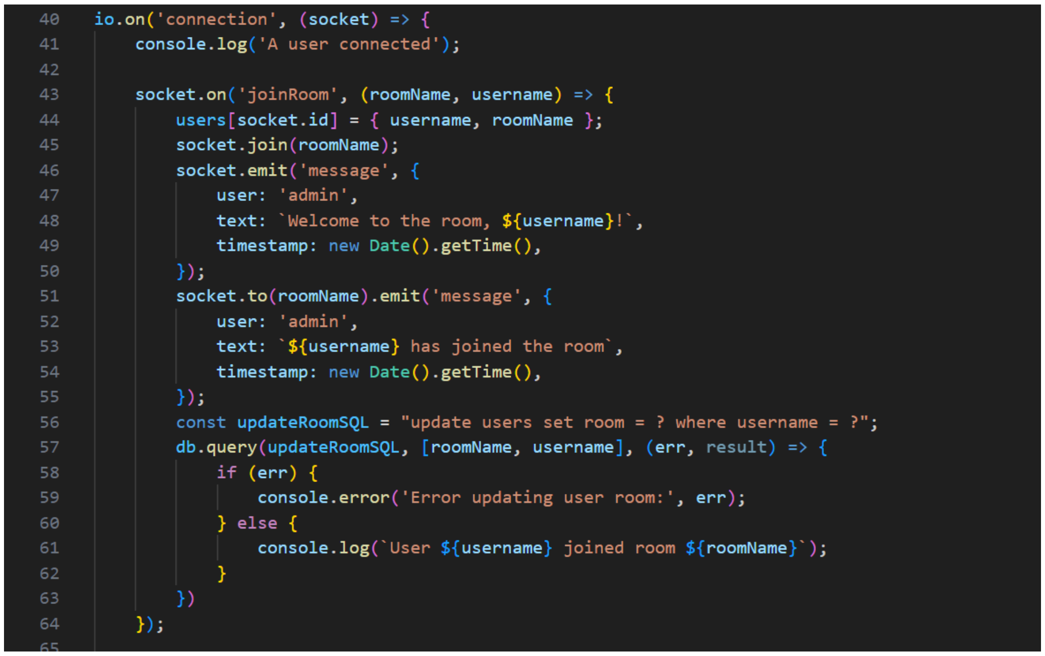
```
const express = require('express');
const http = require('http');
const socketIO = require('socket.io');
const mysql = require('mysql');
const cors = require('cors');

const users = {};

const app = express();
app.use(cors());

const server = http.createServer(app);
const io = socketIO(server, {
    cors: {
        origin: ["http://localhost", "http://localhost:3000"],
        methods: ["GET", "POST"],
        credentials: true,
    },
});
const db = mysql.createConnection({
    host: 'localhost',
    user: 'root',
    password: 'rootPass',
    database: 'vcr'
});

const pool = mysql.createPool({
    connectionLimit: 50,
    host: 'localhost',
```

Figure 11: Starting a Socket.io server with JavaScript

Next, JavaScript. Just like PHP the use of JavaScript is apparent throughout this application and it would have been nearly impossible to complete this project without it.

First and foremost, the server.js and app.js files, which are perhaps the most fundamental parts of VCR are written in JavaScript. These two files are the reason VCR is a real-time chat application, meaning the messages show up automatically and without much of a delay to every user connected, but that will be discussed further in the Socket.io section soon.

Moreover, JavaScript's Ajax, document element manipulation and event handling were are invaluable during the development of this project. The ability to put inline JS scripts furthermore enhanced the readability and cleanness of the code and project folder.

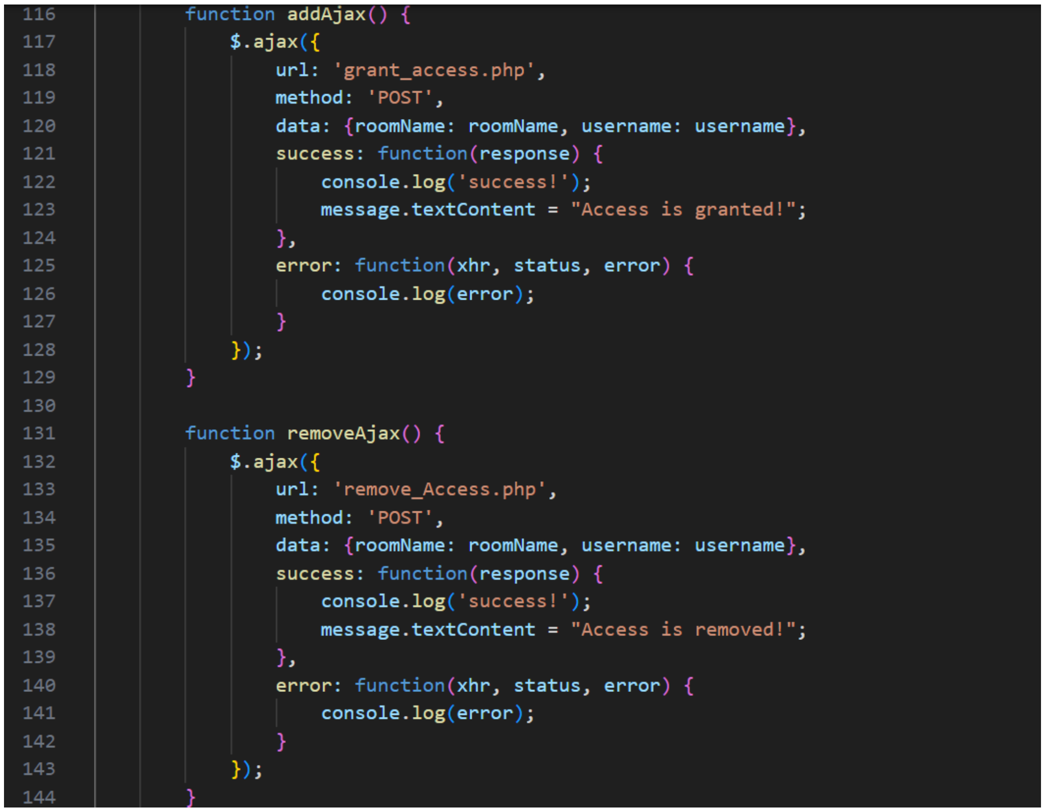


```
io.on('connection', (socket) => {
    console.log('A user connected');

    socket.on('joinRoom', (roomName, username) => {
        users[socket.id] = { username, roomName };
        socket.join(roomName);
        socket.emit('message', {
            user: 'admin',
            text: `Welcome to the room, ${username}!`,
            timestamp: new Date().getTime(),
        });
        socket.to(roomName).emit('message', {
            user: 'admin',
            text: `${username} has joined the room`,
            timestamp: new Date().getTime(),
        });
        const updateRoomSQL = "update users set room = ? where username = ?";
        db.query(updateRoomSQL, [roomName, username], (err, result) => {
            if (err) {
                console.error('Error updating user room:', err);
            } else {
                console.log(`User ${username} joined room ${roomName}`);
            }
        })
    });
```

Figure 12: Part of server.js, use of Socket.io

After that, Socket.io. The newest technology for me personally and the most challenging part of the app to implement. Socket.io allows the clients to establish and hold a live connection with the server, as oppose to the traditional method of attempting to make a new connection with each response. This, makes Socket.io the ideal technology for subject of this project since we don't want the user to have to reload their page constantly.

As significant as Socket.io is to VCR, it's implementation is presence is mostly summarized in two files; the app.js and server.js. The server.js, as the name suggests, takes care of the back-end side of the app while app.js (in this case VirtualChatRoom.js) is mainly for front-end operations and is in direct contact with other front-end files such as Home.php.



```
        function addAjax() {
            $.ajax({
                url: 'grant_access.php',
                method: 'POST',
                data: {roomName: roomName, username: username},
                success: function(response) {
                    console.log('success!');
                    message.textContent = "Access is granted!";
                },
                error: function(xhr, status, error) {
                    console.log(error);
                }
            });
        }

        function removeAjax() {
            $.ajax({
                url: 'remove_Access.php',
                method: 'POST',
                data: {roomName: roomName, username: username},
                success: function(response) {
                    console.log('success!');
                    message.textContent = "Access is removed!";
                },
                error: function(xhr, status, error) {
                    console.log(error);
                }
            });
        }
```

Figure 13: Access system's Ajax

Lastly, Ajax. Being able to call a php script from a JS file simplified many features of VCR and in some cases even, made the impossible, doable.

While the HTML and PHP are loading the correct information according to user, JavaScript takes care of event handling, and when the event is fired, the custom data travels from the PHP script to JS, an Ajax is sent to a different PHP script and event gets handled gracefully. The most important example of which can be seen in how VCR handles grant / remove access feature.

0.5 Project Development Process

At the beginning, there was already a clear image of what VCR is going to be, what wasn't so apparent, was the question how? The decision was made to break this humongous task into many small tasks, and from there start with things that are the most fundamental and develop outwards (or upwards).

The first stage of development, after setting up the required software and repositories, was creating the database. Fortunately, this was a straightforward task, a simple relational database. A database was created named VCR with three tables; one for users, one for rooms, and one for messages. The database structure and tables did change a few times naturally during the development of the application, but it was still the same idea and three tables.

After the creation of the database, it was time to start coding; so, next implementation were the sign in and sign up forms. Again, pretty straight forward, the app checks the form inputs and compares them to the information in the database, in the case of sign up, a user is simply added to the database; of course, both forms needed proper error handling and input checks.

After logging in, we arrive at the home page, where the main event is happening. This was by far the longest and most challenging part of the app to develop. Socket.io was completely brand new to me and I faced many errors. Unfortunately, it was mostly impossible to see the result of code without handling the errors first, and at the same time, there was certain amount of time that could be spent on error handling in each session.

Fortunately, after about two weeks, the home page of the application, when the chatting happens was all ready, no errors in the browser or command prompt. With the successful implementation of socket.io, and creation of app.js and server.js, the front-end and back-end parts of the application could interact with each other and connect to the database successfully, and it was time to add functionalities to elements.

Moving on, the room selector was implemented, followed by the chat window, which showed the messages based on the room selected; There were already messages hard coded inside the database for the purpose of testing. Finally, the submit button for sending users message was activated and the main part of the app was all done and ready.

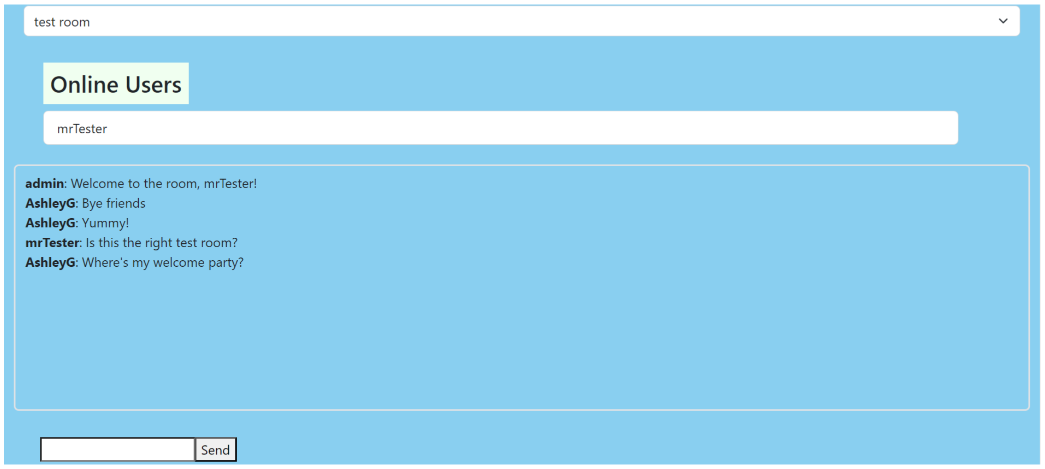
With the fundamental part of the app out of the way, it was time for some styling. At this point the app looked extremely generic, colorless and boring, so through the use of bootstrap 5 that changed, even though it still had a very basic look, the new version of the app was much more presentable.

Next, the extra features needed to be implemented. The admin got a new button in their nav bar redirecting them to a new page where they could make new rooms. A simple form with one text input for room name, the room name needed to be new and a character length limit was set.

After that, the admin needed to be able delete the rooms which were no longer needed, so an other button was added to their nav bar, taking them to a new page where they can see a list of all rooms inside a table. There was put a delete button in each row corresponding to the room name, and with a simple click of that button, the admin was then able to delete any room.

Moving on, it was time for implementation of an other extremely significant feature; the access system. It was required that VCR gives the admin the ability to give or remove access to any room from any user. Somewhat more challenging than the two former features, but with the use of Ajax, very doable. An other button was added to admin's nav bar, taking them to a page where they would find a form with two select elements, one for user and one for student, and two buttons at the bottom, one for granting access and one for taking it away.

With that out of the way, the implementation of the last feature which was to be accessible to both the users and admin began; the search feature. This time, everybody received a new button on their nav bar, taking them to a form where they could select from the list of rooms which they currently have access to and a text input, so they can search their sentence, word, or even partial word. Upon clicking the search button, the user is taken to a new page where they can all the messages that matched their search inside a table with details such as the sender of the message, the date and time it was sent, and of course the full message itself.



admin: Welcome to the room, mrTester!
AshleyG: Bye friends
AshleyG: Yummy!
mrTester: Is this the right test room?
AshleyG: Where's my welcome party?

Figure 14: VCR's new and final look

Finally, as a last touch, the decision was made to make the app more beautiful and presentable. That was the week were most of the CSS was implemented and not so much bootstrap. The app got a light blue back ground color, the reason behind using this color is not just because it is easy on the eyes, but also because studies show that the color blue helps people with tasks like thinking, studying, brainstorming, etc, which are exactly what students using this app would want to do. A light green was also incorporated into some elements background color to further enhance the looks of the app.

In addition to adding new colors, the margin and padding of elements were adjust, and the borders were rounded were it seemed fit. Overall, by the end of the last week, the app looked much better with more life and character.

0.6 Challenges Faced

Developing VCR was certainly a challenging process, even though a good portion of the app was similar to the previous projects, the requirements and specifications of this project made it so I constantly had to be creative and come up with new solutions outside of what I had seen and was familiar with.

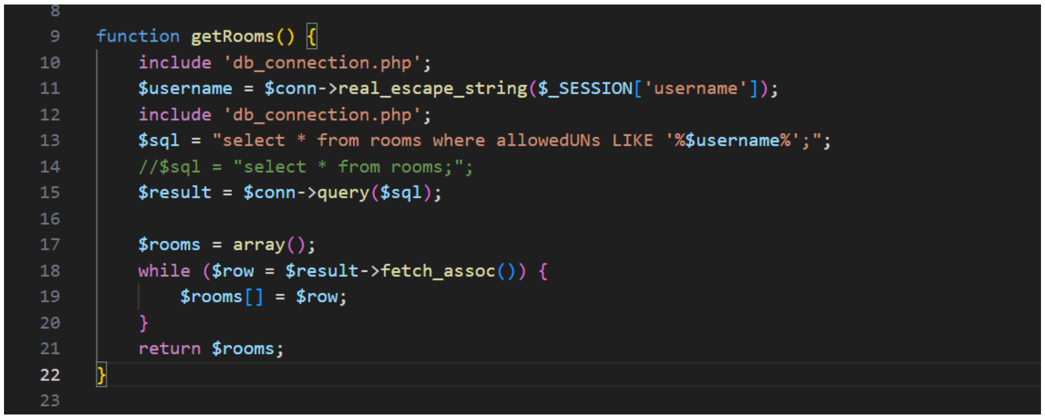
From the most challenging parts, the CORS issue definitely took the heaviest toll; I encountered this issue pretty early in the development and it slowed everything down significantly, to the point where it felt hopeless to finish the project on time, judging by the slow rate the project was moving forward.

After few weeks it finally got resolved by re-coding the server.js, re-structuring the files into two different folders, and adding CORS headers to php scripts. It is after resolving the CORS issue when the development started gaining pace and moving at a reasonable rate.

Aside from the CORS issue, the development f app.js and server.js were also pretty challenging, as mentioned before it was a brand new topic to me and I made so many mistakes before finally getting it right.

One other notable challenge during the development of VCR, was the implementation of the access system. At first, it seemed like not implementing it at the beginning may be a fatal mistake and would cost many hours of re-coding the already established and functional parts of the app. Fortunately, that was not the case, after some thinking, and weighing the pros and cons of each option, the decision was made to not add any new table to the database, instead, just add an extra column to the rooms table; it is here where the app keeps the usernames of clients who have access to the room in that specific row.

The usernames would be separated by two semicolons (;;), and that's how the VCR will later separate and re-arrange them into an array. Additionally, a form was designed and implemented that would allow the admin the pair a username with a room and then grant or remove the access to that room from user, technically adding or removing that username plus two semi-colons from that room's row in the database.



```php

function getRooms() {
    include 'db_connection.php';
    $username = $conn->real_escape_string($_SESSION['username']);
    include 'db_connection.php';
    $sql = "select * from rooms where allowedUNs LIKE '%$username%';";
    //$sql = "select * from rooms;";
    $result = $conn->query($sql);

    $rooms = array();
    while ($row = $result->fetch_assoc()) {
        $rooms[] = $row;
    }
    return $rooms;
}

```

Figure 15: Updated function for retrieving rooms

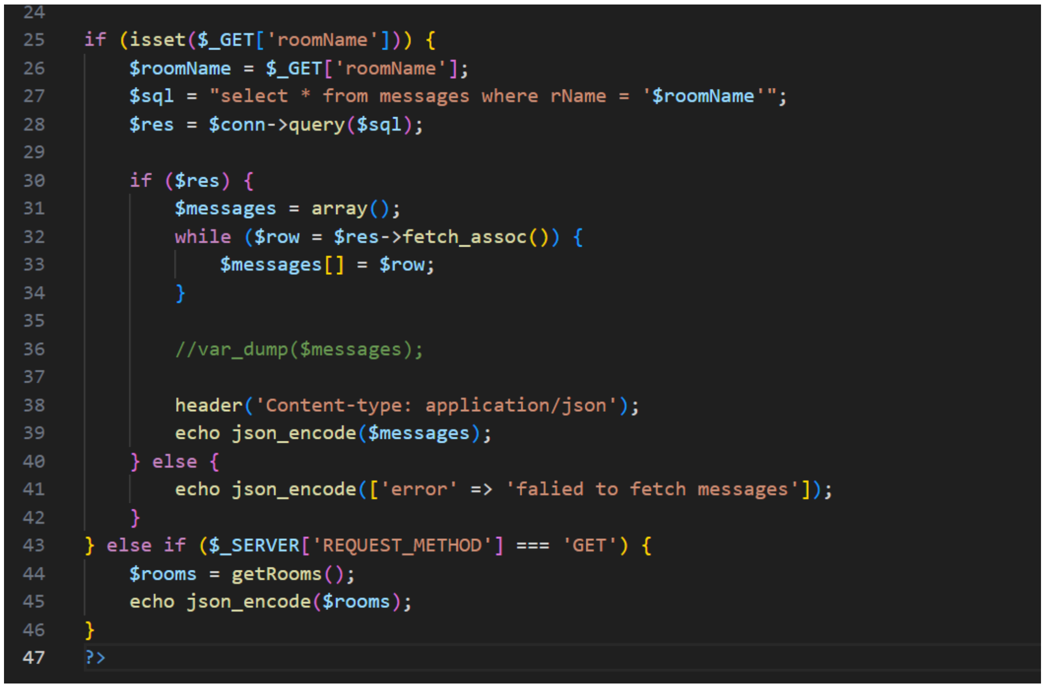
After that, the new query for getting the room list and populating the select element would take into account the access system and would only show the rooms where the user has access to , both for chatting and for searching the chat history.

0.7 Conclusion

Overall, this internship project was an extremely beneficial and exciting experience. Many thanks to my instructor ms. Kawther Aarizou for designing this project and guiding me through out the entire development; without which, VCR would still be in early stages of development by now.

Moreover, I'm thankful for learning many new things such as, socket programming, which was a completely new but very useful topic, even though it created the hardest challenges. Additionally, I certainly advanced in understanding and using JS, PHP, CSS and MySQL, topics which I had already studied, but had to learn more about in order to finish this project.

VCR is currently in a good position; certainly far from perfect, and it's clear how much room there is for improvements, but it's still a decent app that serves its purpose, looks somewhat charming, and to my knowledge and according to the last tests, free of bugs and errors.



```php

if (isset($_GET['roomName'])) {
    $roomName = $_GET['roomName'];
    $sql = "select * from messages where rName = '$roomName'";
    $res = $conn->query($sql);

    if ($res) {
        $messages = array();
        while ($row = $res->fetch_assoc()) {
            $messages[] = $row;
        }

        //var_dump($messages);

        header('Content-type: application/json');
        echo json_encode($messages);
    } else {
        echo json_encode(['error' => 'falied to fetch messages']);
    }
} else if ($_SERVER['REQUEST_METHOD'] === 'GET') {
    $rooms = getRooms();
    echo json_encode($rooms);
}
?>
```

Figure 16: Part of api.php

0.8 Appendix A: Code Snippets and Diagrams

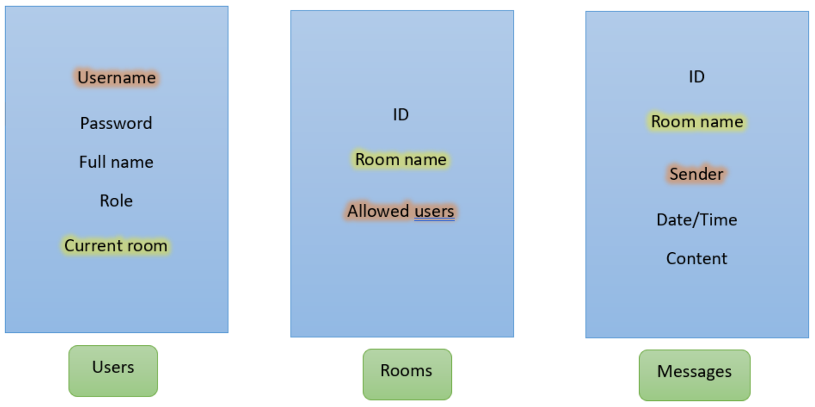


Figure 17: Database tables

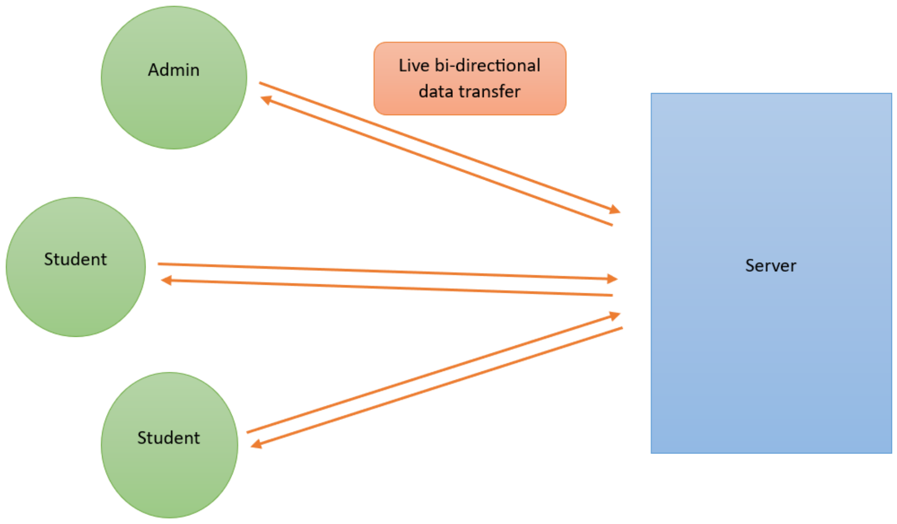


Figure 18: Socket.io