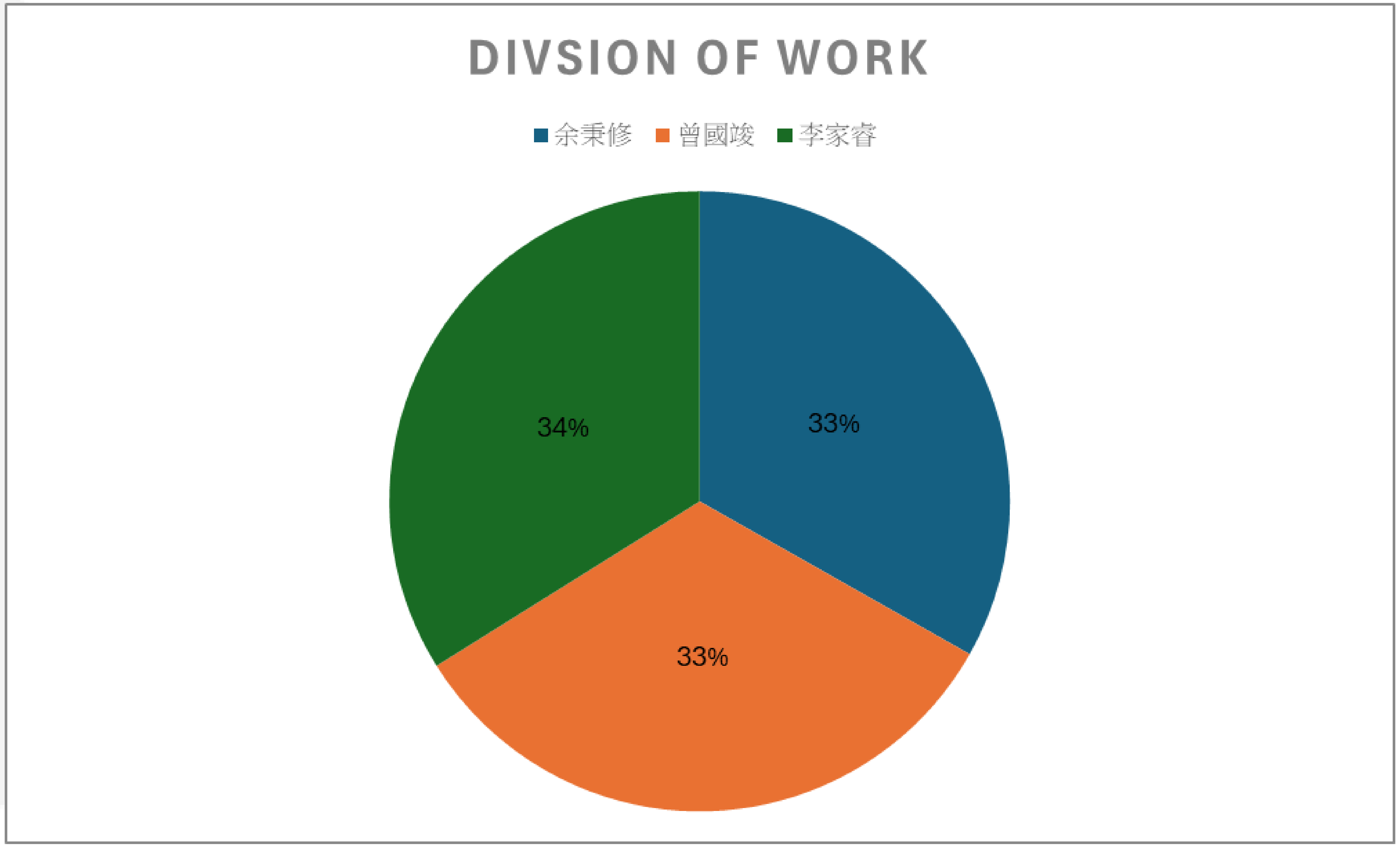
Final Report

Class 3A Electronics	Bingxiu Yu
Class 3B Electronics	Chiajui Lee
Class 3C Electronics	Guojun Zeng

Table of contents

Division of work



Back to Agenda

System Architecture

.HTML

HTML is a standard markup language used for creating web pages. As a foundational technology, it is commonly combined with **CSS** (Cascading Style Sheets) and **JavaScript** by websites, web applications, and mobile applications to design user interfaces.

.CSS

CSS is a computer language used to add styling (e.g., fonts, spacing, colors) to structured documents such as **HTML** or **XML**-based applications. It is defined and maintained by the **W3C** (World Wide Web Consortium).

.JS

JavaScript is a programming language used to manipulate the **content** and **structure** of HTML documents, enabling dynamic page updates without requiring reloads. It responds to user interactions (e.g., button clicks, text input) to modify a page's appearance and behavior.

Back to Agenda

System Architecture

Node.js

Node.js is a JavaScript-based backend runtime environment that executes code using Chrome's V8 JavaScript engine. It specializes in providing a **non-blocking** (asynchronous) and **event-driven** architecture, making it ideal for building high-performance, scalable network applications. As a pivotal technology in modern web development, Node.js is widely used for real-time applications and API development.

Express

Express.js is a minimalist web framework for Node.js, designed for building **web applications** and **APIs**. It simplifies handling HTTP **requests**, **responses**, and **middleware** in server-side development.

Back to Agenda

System Architecture

MySQL

MySQL is an open-source (free) **database management system** widely used in small to medium-sized websites. It integrates with server-side languages like PHP, ASP, or ASP.NET to store large volumes of data. Websites with backend administrative systems (e.g., content management) typically require database functionality.

WebRTC

WebRTC is an **open-source project** that enables web browsers and mobile applications to perform real-time **audio/video communication** and **data transfer** through simple JavaScript APIs.

Back to Agenda

Azure

System Architecture

Back to Agenda

Azure

Azure Web App Services enables users to **quickly deploy and manage** website environments, supporting multiple development frameworks and tools. It is ideal for **testing**, **deployment**, and **scaling applications**. By integrating with Azure services, developers can effortlessly achieve **high-performance**, **secure**, and **scalable** web solutions.

Socket

WebSocket provides the foundation for **real-time communication** in web development, enabling persistent connections between servers and clients. It supports **full-duplex data transmission**, making it ideal for building:

•Chat applications

•Live notifications

•Multi-user interactive apps

By offering **high-efficiency, low-latency** communication, WebSocket delivers robust real-time capabilities for modern web applications.

Design Concept

Back to Agenda

Design Concept

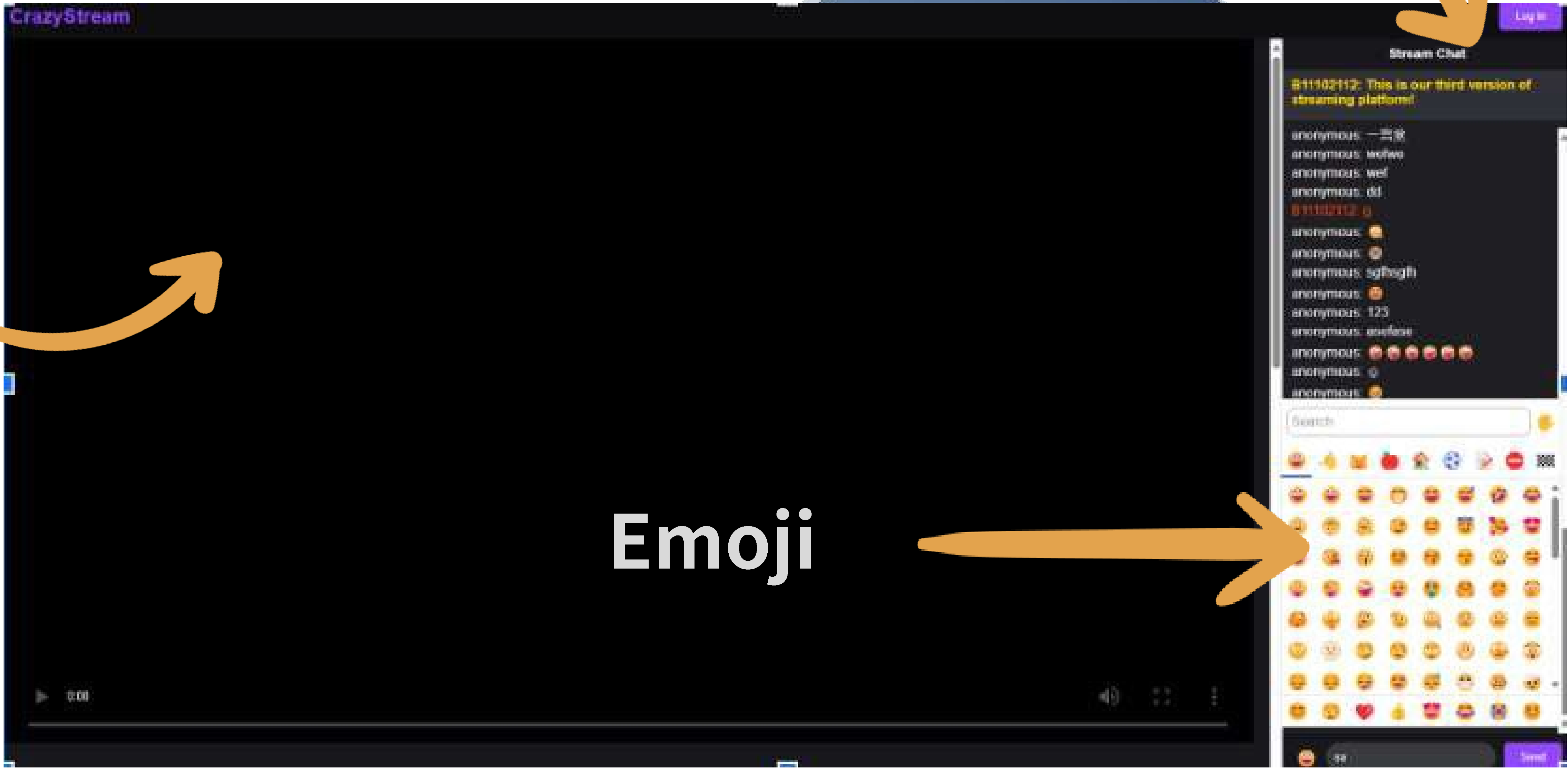
WE FOLLOW STREAMING PLATFORM---TWITCH

1	STREAMING PLATFORM MAIN PAGE	5	FORGET PASSWORD PAGE
2	STREAMER DASHBORAD	6	STREAMER CHAT BOX
3	LOGIN PAGE	7	EMOJI
4	REGISTER PAGE	8	STREAMING SCREEN

Main Streaming Page

Chat Box

Streaming Screen

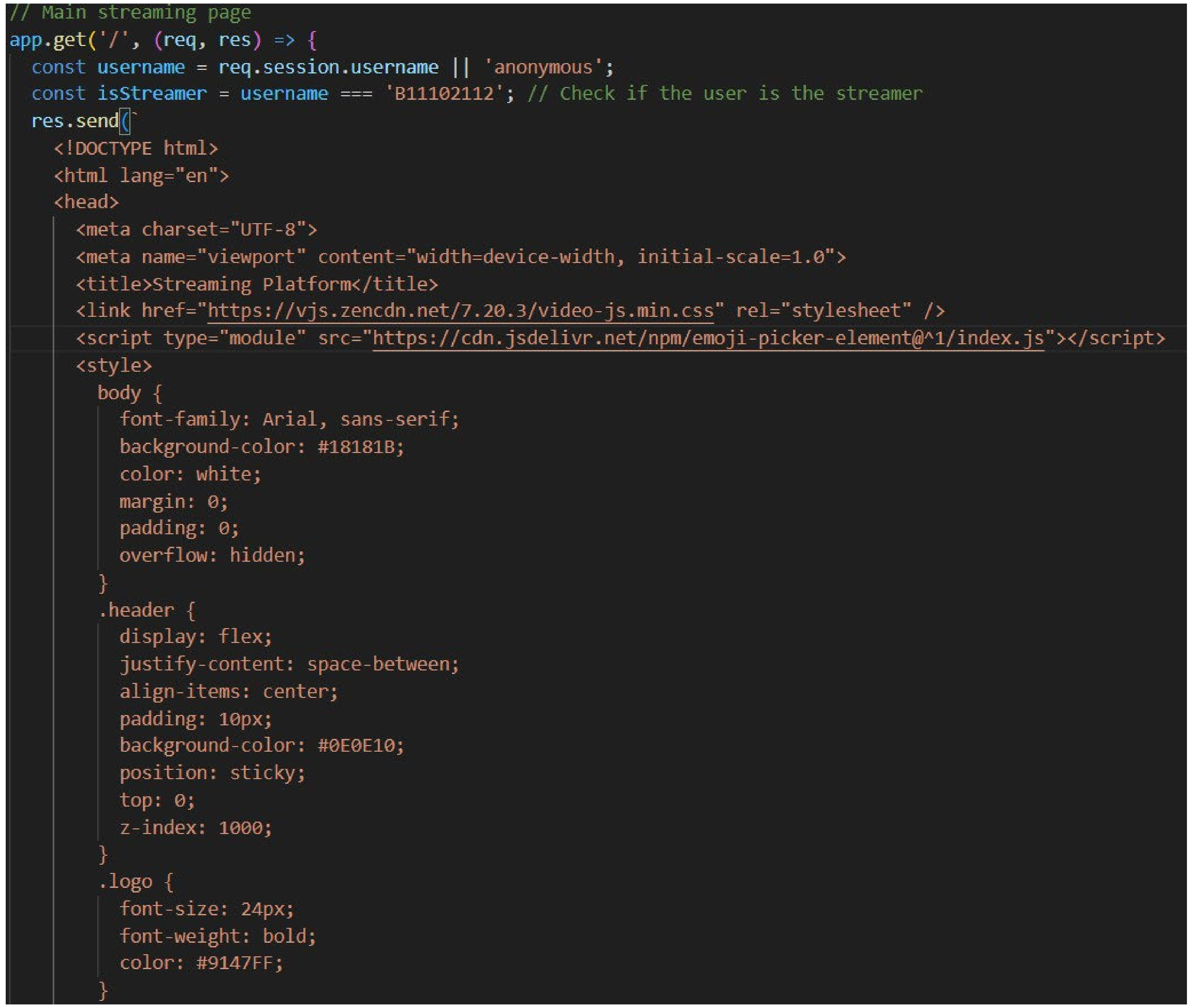


B11102112: This is our third version of streaming platform!

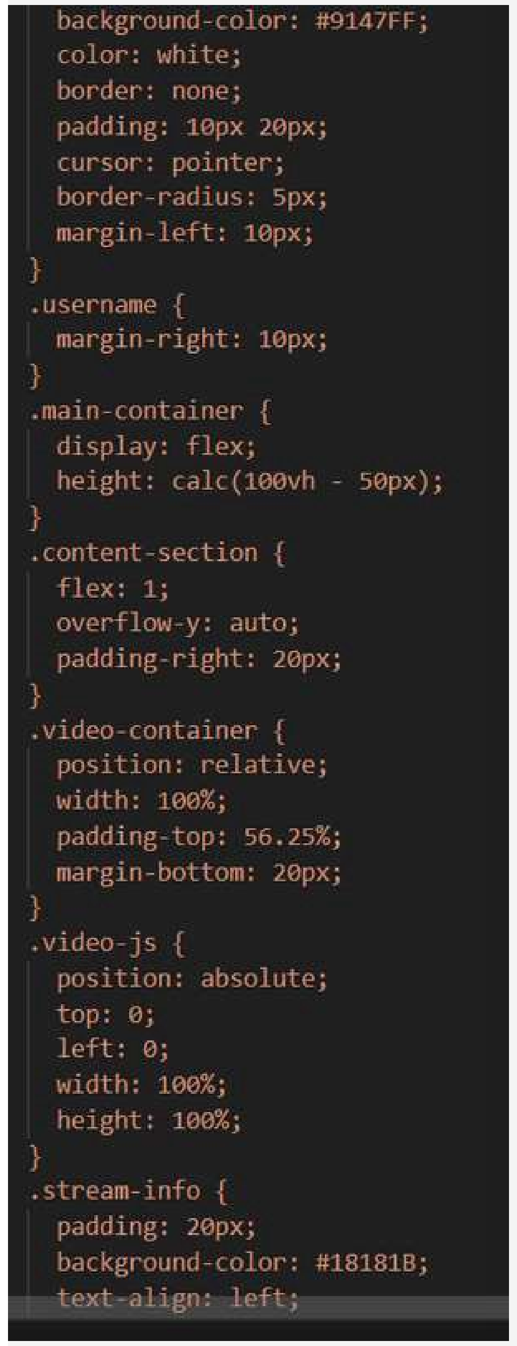
anonymous: 一言堂
anonymous: wefwe
anonymous: wef
anonymous: dd
B11102112: g
anonymous: 😀
anonymous: 😀
anonymous: sgfhsgfh
anonymous: 😀
anonymous: 123
anonymous: asefase
anonymous: 😀😀😀😀😀😀
anonymous: 😀
anonymous: 😀

Back to Agenda

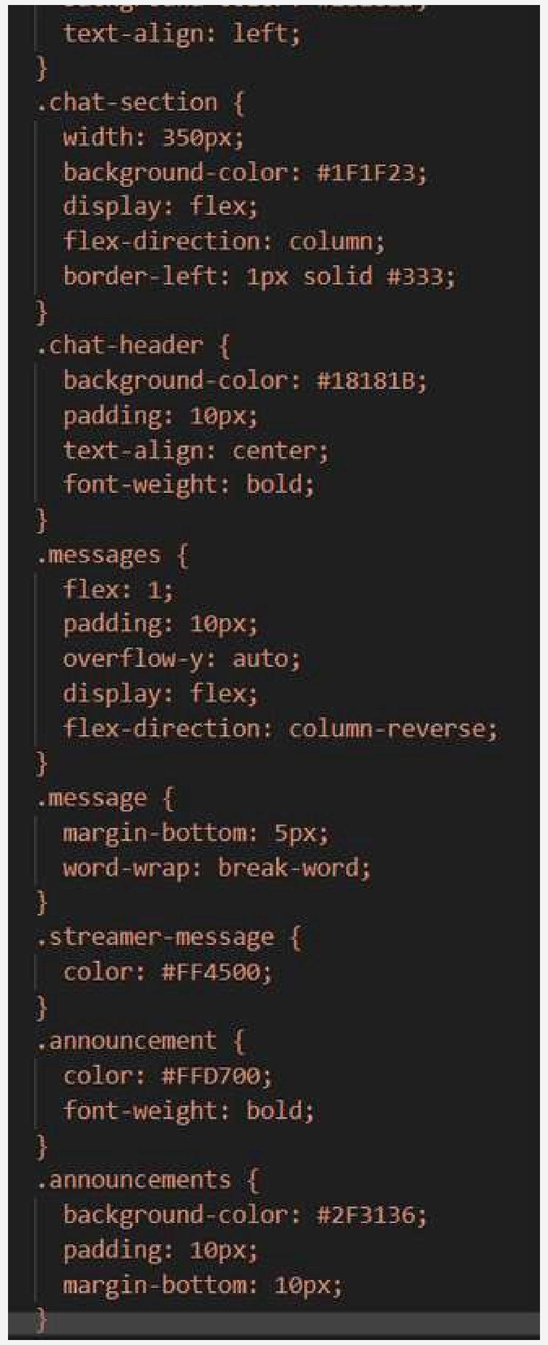
Main Streaming Page code



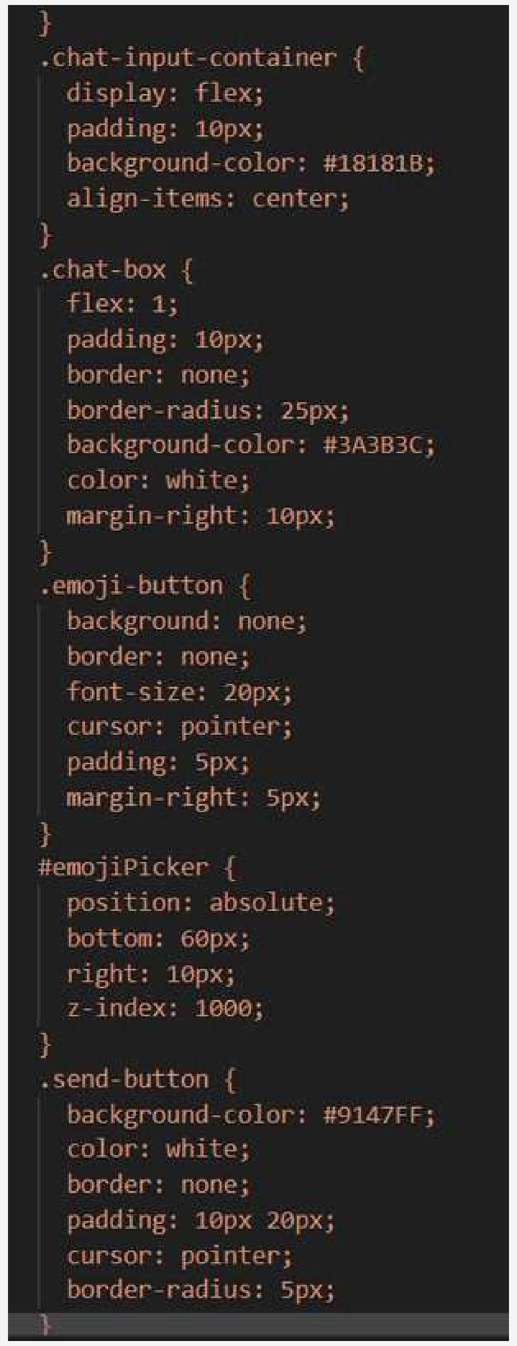
```
// Main streaming page
app.get('/', (req, res) => {
  const username = req.session.username || 'anonymous';
  const isStreamer = username === 'B11102112'; // Check if the user is the streamer
  res.send(`
    <!DOCTYPE html>
    <html lang="en">
    <head>
      <meta charset="UTF-8">
      <meta name="viewport" content="width=device-width, initial-scale=1.0">
      <title>Streaming Platform</title>
      <link href="https://vjs.zencdn.net/7.20.3/video-js.min.css" rel="stylesheet" />
      <script type="module" src="https://cdn.jsdelivr.net/npm/emoji-picker-element@^1/index.js"></script>
      <style>
        body {
          font-family: Arial, sans-serif;
          background-color: #18181B;
          color: white;
          margin: 0;
          padding: 0;
          overflow: hidden;
        }
        .header {
          display: flex;
          justify-content: space-between;
          align-items: center;
          padding: 10px;
          background-color: #0E0E10;
          position: sticky;
          top: 0;
          z-index: 1000;
        }
        .logo {
          font-size: 24px;
          font-weight: bold;
          color: #9147FF;
        }
```



```
  background-color: #9147FF;
  color: white;
  border: none;
  padding: 10px 20px;
  cursor: pointer;
  border-radius: 5px;
  margin-left: 10px;
}
.username {
  margin-right: 10px;
}
.main-container {
  display: flex;
  height: calc(100vh - 50px);
}
.content-section {
  flex: 1;
  overflow-y: auto;
  padding-right: 20px;
}
.video-container {
  position: relative;
  width: 100%;
  padding-top: 56.25%;
  margin-bottom: 20px;
}
.video-js {
  position: absolute;
  top: 0;
  left: 0;
  width: 100%;
  height: 100%;
}
.stream-info {
  padding: 20px;
  background-color: #18181B;
  text-align: left;
```



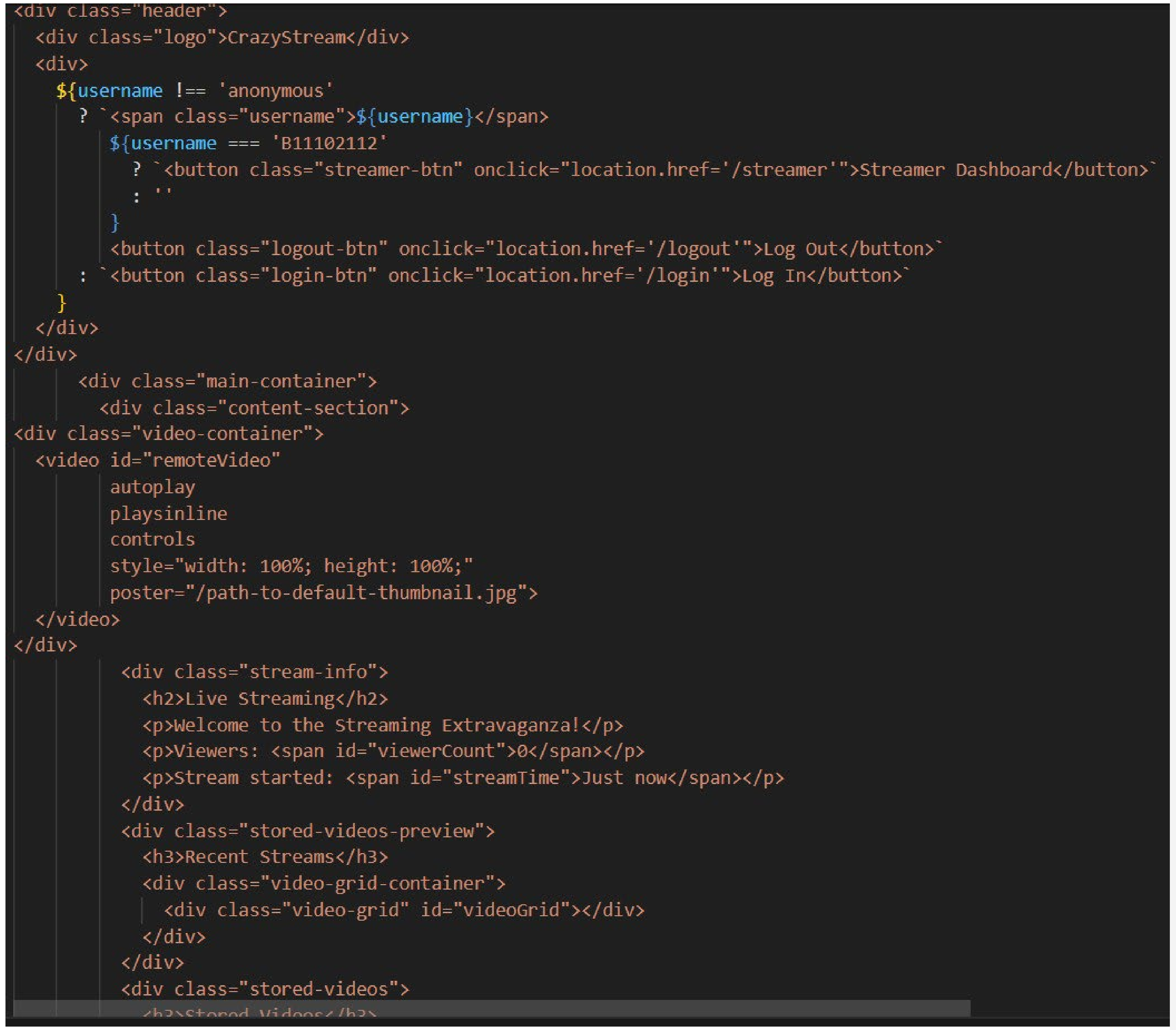
```
  text-align: left;
}
.chat-section {
  width: 350px;
  background-color: #1F1F23;
  display: flex;
  flex-direction: column;
  border-left: 1px solid #333;
}
.chat-header {
  background-color: #18181B;
  padding: 10px;
  text-align: center;
  font-weight: bold;
}
.messages {
  flex: 1;
  padding: 10px;
  overflow-y: auto;
  display: flex;
  flex-direction: column-reverse;
}
.message {
  margin-bottom: 5px;
  word-wrap: break-word;
}
.streamer-message {
  color: #FF4500;
}
.announcement {
  color: #FFD700;
  font-weight: bold;
}
.announcements {
  background-color: #2F3136;
  padding: 10px;
  margin-bottom: 10px;
}
```



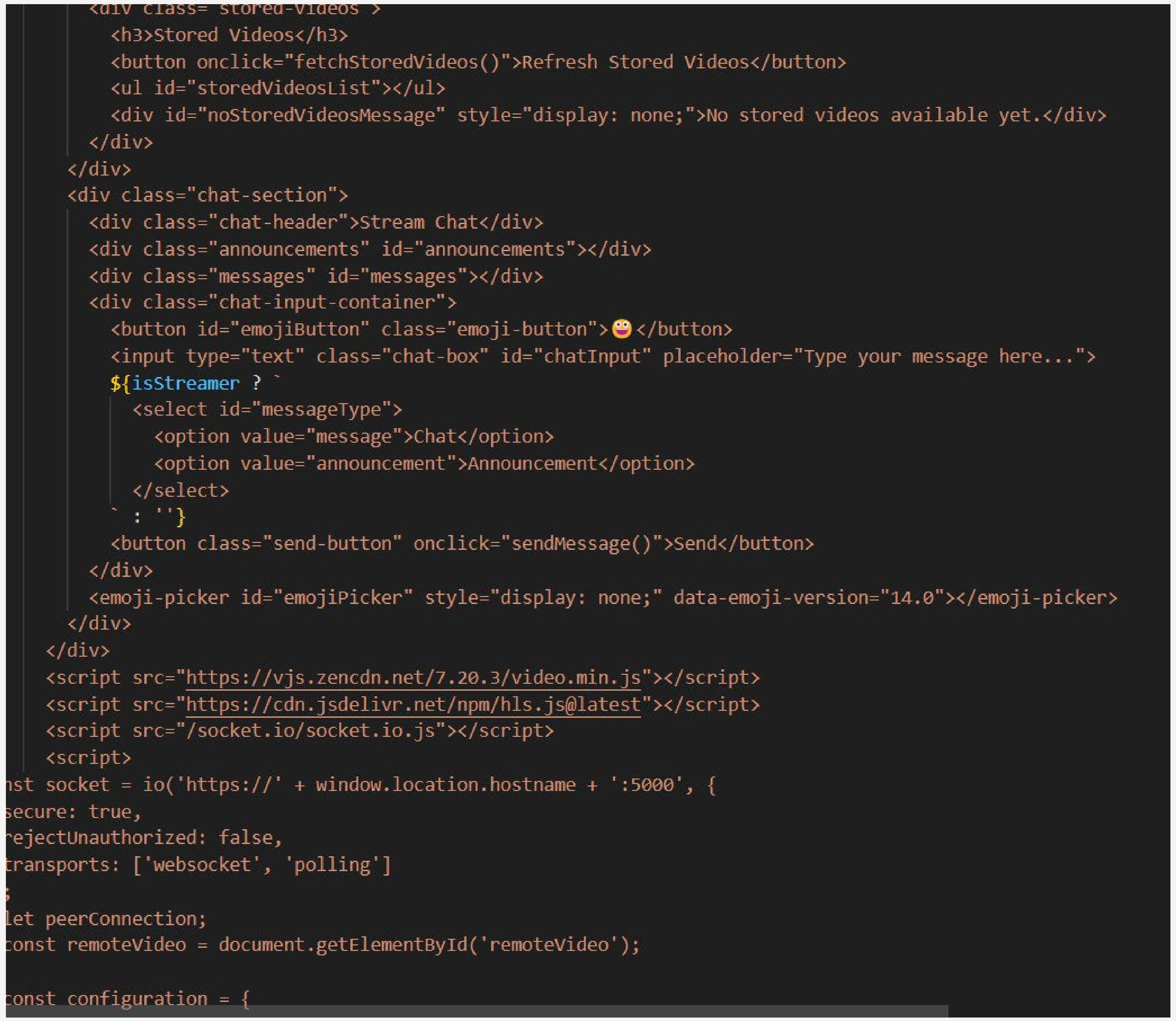
```
}
.chat-input-container {
  display: flex;
  padding: 10px;
  background-color: #18181B;
  align-items: center;
}
.chat-box {
  flex: 1;
  padding: 10px;
  border: none;
  border-radius: 25px;
  background-color: #3A3B3C;
  color: white;
  margin-right: 10px;
}
.emoji-button {
  background: none;
  border: none;
  font-size: 20px;
  cursor: pointer;
  padding: 5px;
  margin-right: 5px;
}
#emojiPicker {
  position: absolute;
  bottom: 60px;
  right: 10px;
  z-index: 1000;
}
.send-button {
  background-color: #9147FF;
  color: white;
  border: none;
  padding: 10px 20px;
  cursor: pointer;
  border-radius: 5px;
}
```

Back to Agenda

Main Streaming Page code



```html
<div class="header">
  <div class="logo">CrazyStream</div>
  <div>
    ${username !== 'anonymous'
      ? `<span class="username">${username}</span>
        ${username === 'B11102112'
          ? `<button class="streamer-btn" onclick="location.href='/streamer'">Streamer Dashboard</button>`
          : ''
        }
        <button class="logout-btn" onclick="location.href='/logout'">Log Out</button>`
      : `<button class="login-btn" onclick="location.href='/login'">Log In</button>`
    }
  </div>
</div>
      <div class="main-container">
        <div class="content-section">
<div class="video-container">
  <video id="remoteVideo"
        autoplay
        playsinline
        controls
        style="width: 100%; height: 100%;"
        poster="/path-to-default-thumbnail.jpg">
  </video>
</div>
          <div class="stream-info">
            <h2>Live Streaming</h2>
            <p>Welcome to the Streaming Extravaganza!</p>
            <p>Viewers: <span id="viewerCount">0</span></p>
            <p>Stream started: <span id="streamTime">Just now</span></p>
          </div>
          <div class="stored-videos-preview">
            <h3>Recent Streams</h3>
            <div class="video-grid-container">
              <div class="video-grid" id="videoGrid"></div>
            </div>
          </div>
          <div class="stored-videos">
            <h3>Stored Videos</h3>
```



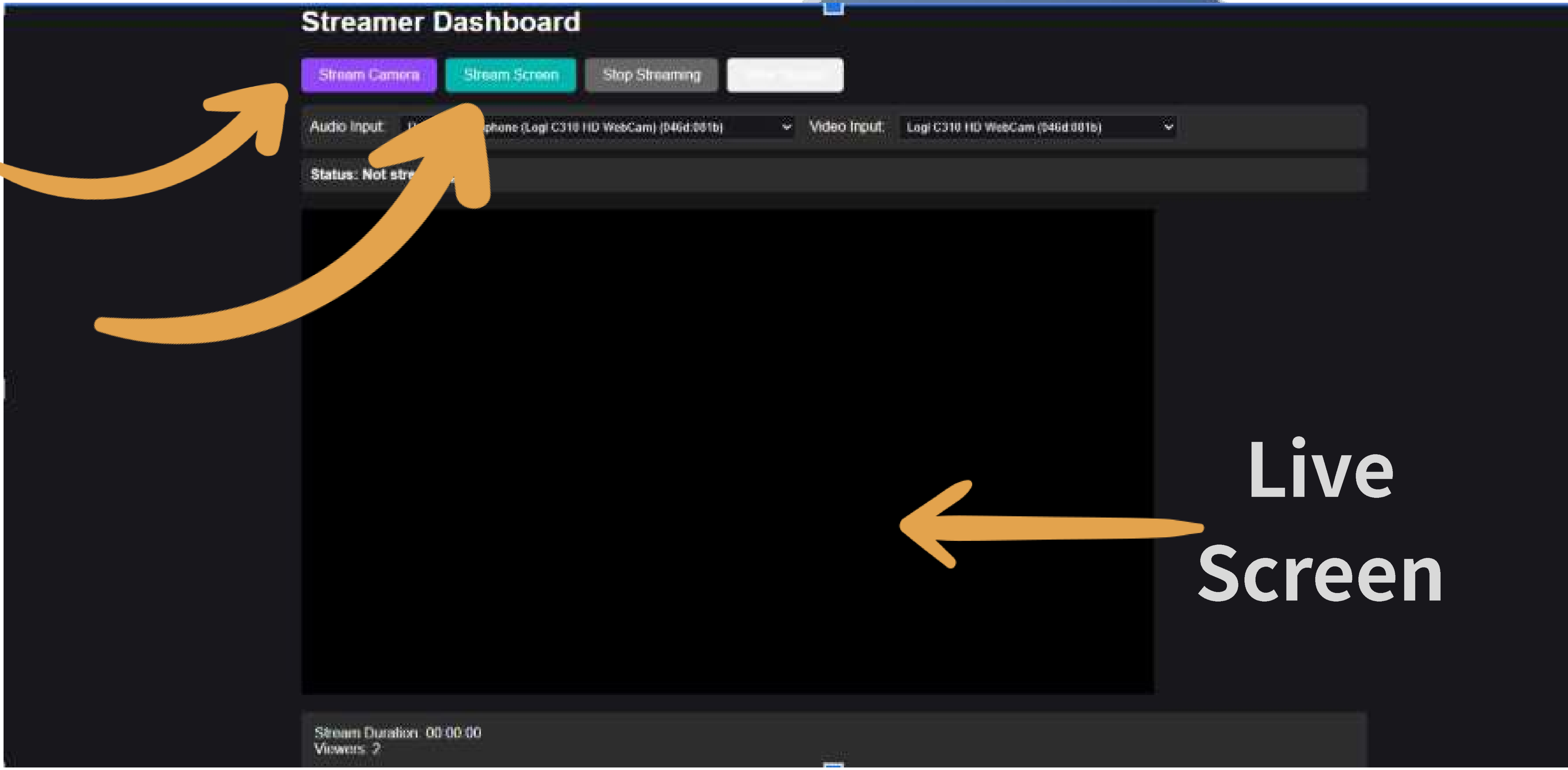
```html
          <div class="stored-videos">
            <h3>Stored Videos</h3>
            <button onclick="fetchStoredVideos()">Refresh Stored Videos</button>
            <ul id="storedVideosList"></ul>
            <div id="noStoredVideosMessage" style="display: none;">No stored videos available yet.</div>
          </div>
        </div>
        <div class="chat-section">
          <div class="chat-header">Stream Chat</div>
          <div class="announcements" id="announcements"></div>
          <div class="messages" id="messages"></div>
          <div class="chat-input-container">
            <button id="emojiButton" class="emoji-button">😛</button>
            <input type="text" class="chat-box" id="chatInput" placeholder="Type your message here...">
            ${isStreamer ? `
              <select id="messageType">
                <option value="message">Chat</option>
                <option value="announcement">Announcement</option>
              </select>
            ` : ''}
            <button class="send-button" onclick="sendMessage()">Send</button>
          </div>
          <emoji-picker id="emojiPicker" style="display: none;" data-emoji-version="14.0"></emoji-picker>
        </div>
      </div>
      <script src="https://vjs.zencdn.net/7.20.3/video.min.js"></script>
      <script src="https://cdn.jsdelivr.net/npm/hls.js@latest"></script>
      <script src="/socket.io/socket.io.js"></script>
      <script>
nst socket = io('https://' + window.location.hostname + ':5000', {
ecure: true,
ejectUnauthorized: false,
ransports: ['websocket', 'polling']
;
let peerConnection;
onst remoteVideo = document.getElementById('remoteVideo');

onst configuration = {
```

Back to Agenda

Streamer DashBoard

Streaming Camera

Streaming Screen



Streamer Dashboard

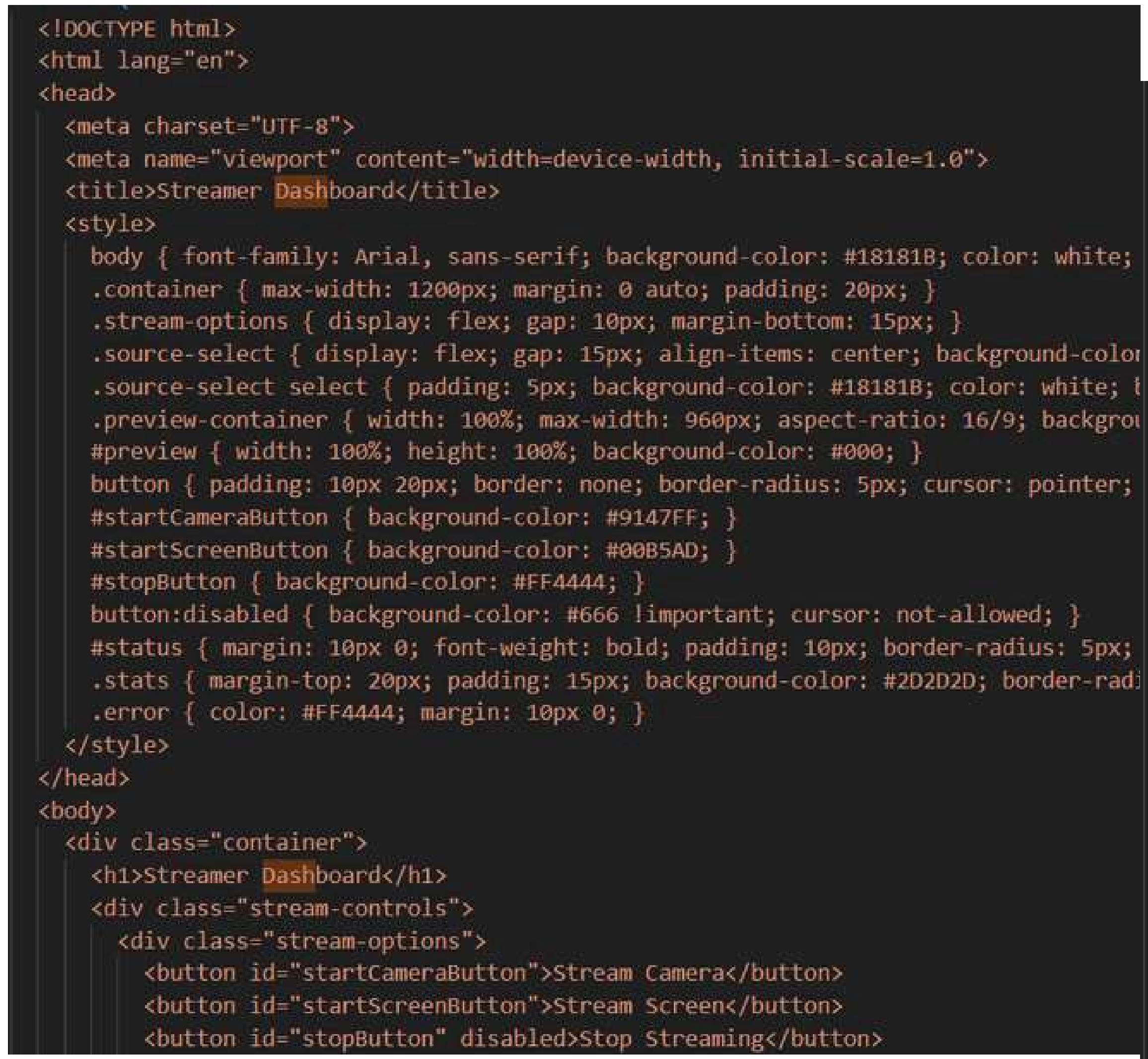
Audio Input: phone (Logi C310 HD WebCam) (046d:081b)   Video Input: Logi C310 HD WebCam (046d:081b)

Status: Not stre

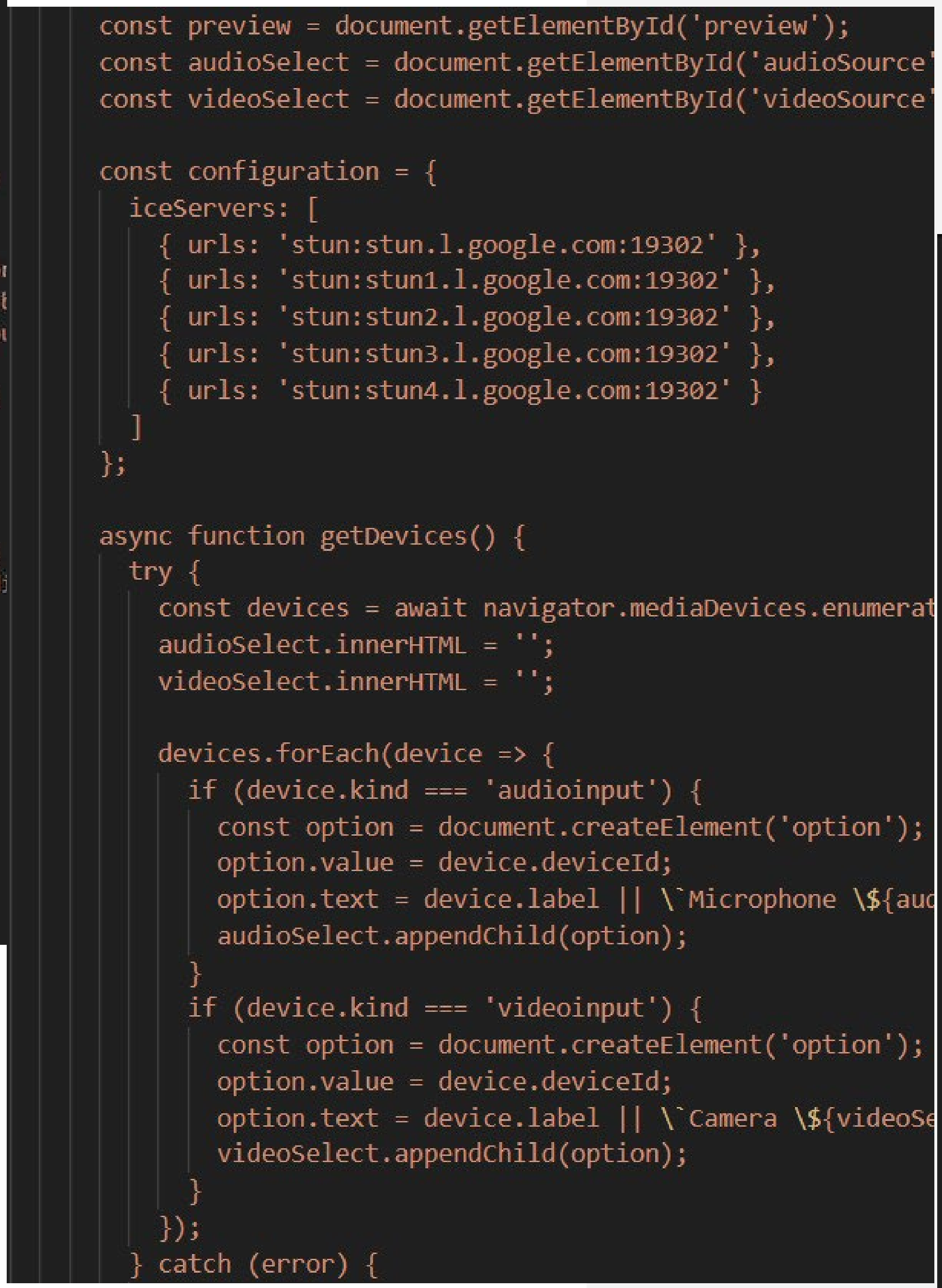
Stream Duration: 00:00:00
Viewers: 2

Back to Agenda

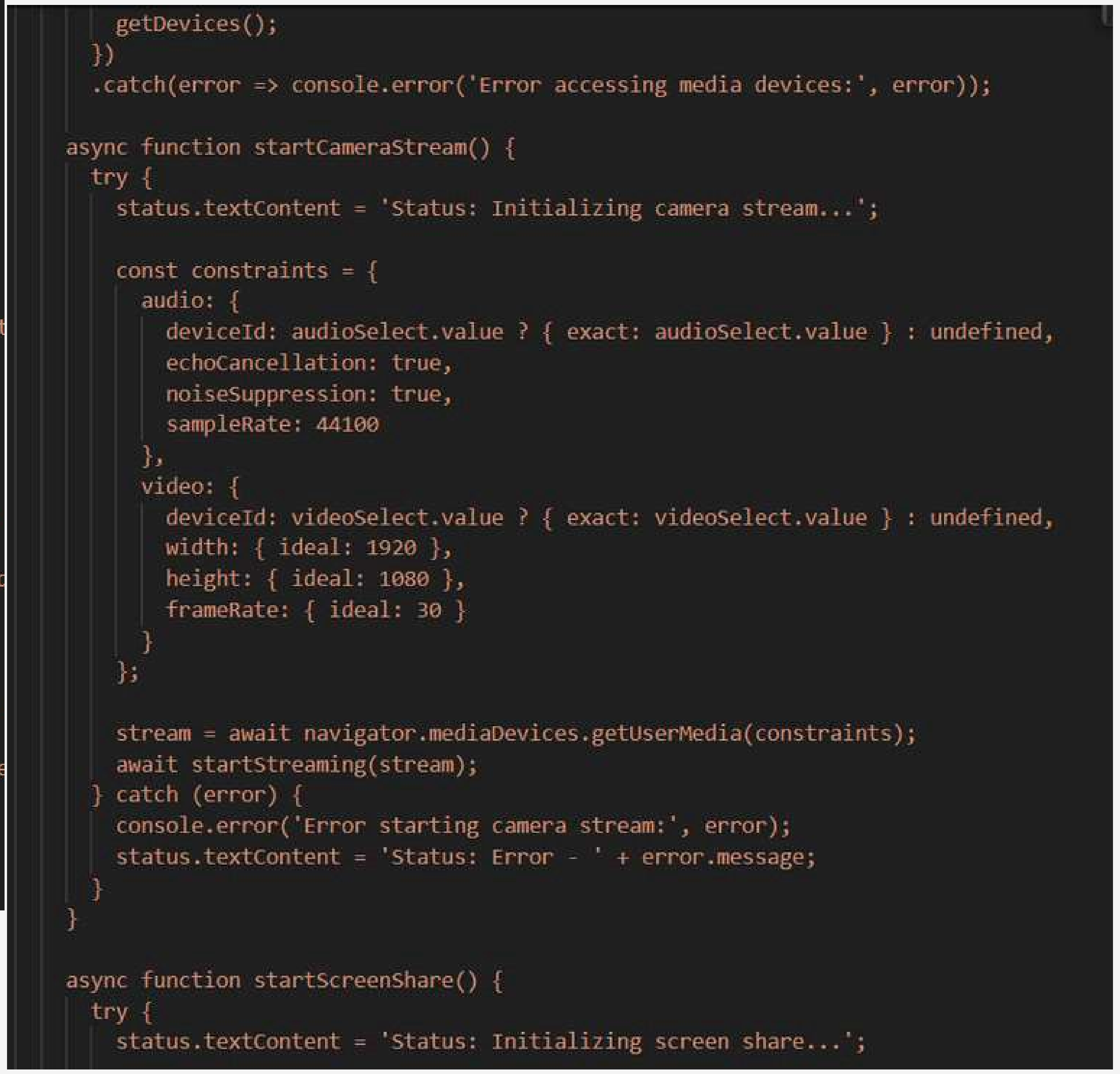
Code



```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Streamer Dashboard</title>
  <style>
    body { font-family: Arial, sans-serif; background-color: #18181B; color: white;
    .container { max-width: 1200px; margin: 0 auto; padding: 20px; }
    .stream-options { display: flex; gap: 10px; margin-bottom: 15px; }
    .source-select { display: flex; gap: 15px; align-items: center; background-colo
    .source-select select { padding: 5px; background-color: #18181B; color: white; 
    .preview-container { width: 100%; max-width: 960px; aspect-ratio: 16/9; backgrou
    #preview { width: 100%; height: 100%; background-color: #000; }
    button { padding: 10px 20px; border: none; border-radius: 5px; cursor: pointer;
    #startCameraButton { background-color: #9147FF; }
    #startScreenButton { background-color: #00B5AD; }
    #stopButton { background-color: #FF4444; }
    button:disabled { background-color: #666 !important; cursor: not-allowed; }
    #status { margin: 10px 0; font-weight: bold; padding: 10px; border-radius: 5px;
    .stats { margin-top: 20px; padding: 15px; background-color: #2D2D2D; border-radi
    .error { color: #FF4444; margin: 10px 0; }
  </style>
</head>
<body>
  <div class="container">
    <h1>Streamer Dashboard</h1>
    <div class="stream-controls">
      <div class="stream-options">
        <button id="startCameraButton">Stream Camera</button>
        <button id="startScreenButton">Stream Screen</button>
        <button id="stopButton" disabled>Stop Streaming</button>
```

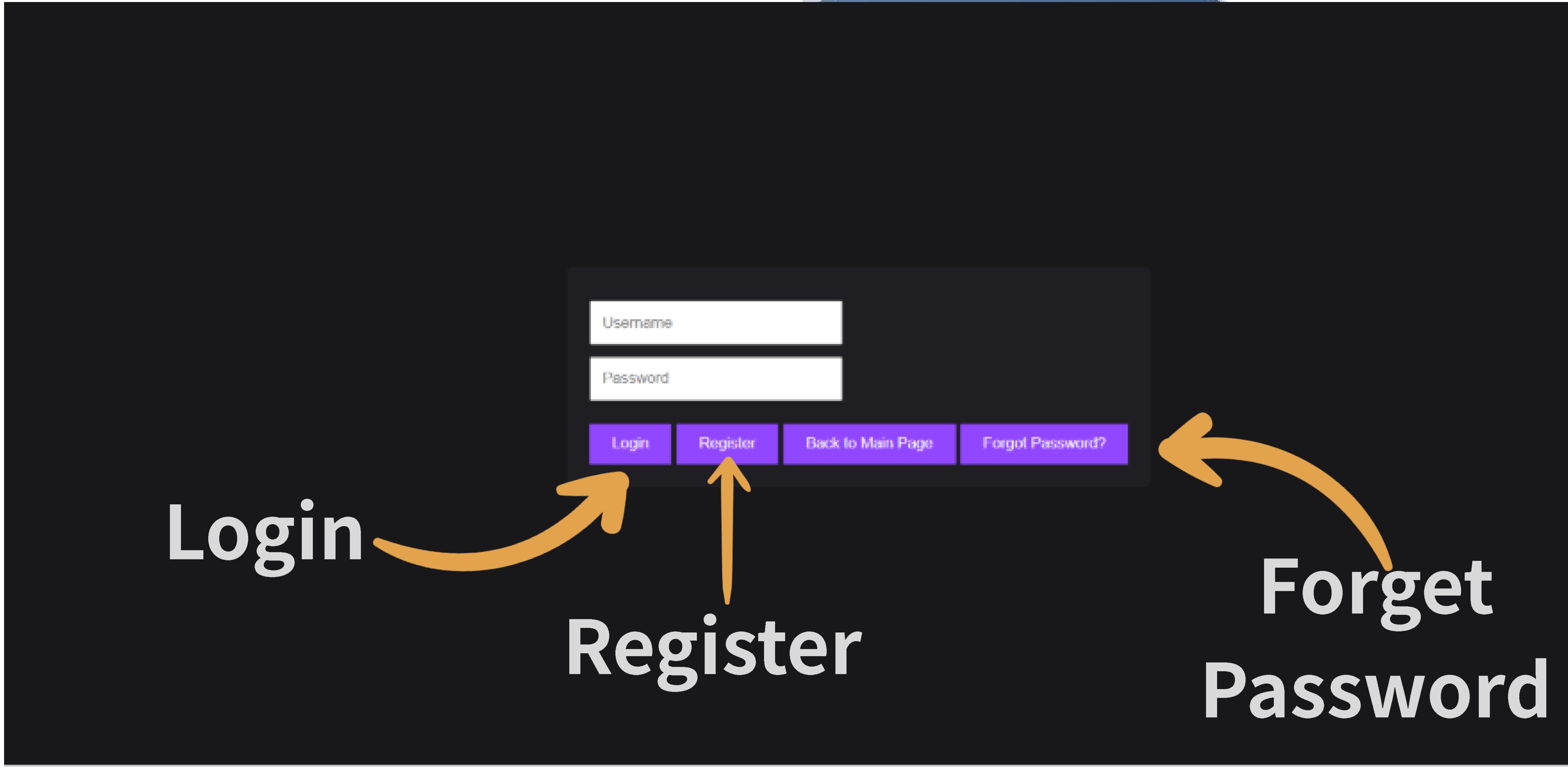


```
    const preview = document.getElementById('preview');
    const audioSelect = document.getElementById('audioSource'
    const videoSelect = document.getElementById('videoSource'

    const configuration = {
      iceServers: [
        { urls: 'stun:stun.l.google.com:19302' },
        { urls: 'stun:stun1.l.google.com:19302' },
        { urls: 'stun:stun2.l.google.com:19302' },
        { urls: 'stun:stun3.l.google.com:19302' },
        { urls: 'stun:stun4.l.google.com:19302' }
      ]
    };

    async function getDevices() {
      try {
        const devices = await navigator.mediaDevices.enumerat
        audioSelect.innerHTML = '';
        videoSelect.innerHTML = '';

        devices.forEach(device => {
          if (device.kind === 'audioinput') {
            const option = document.createElement('option');
            option.value = device.deviceId;
            option.text = device.label || \`Microphone \${aud
            audioSelect.appendChild(option);
          }
          if (device.kind === 'videoinput') {
            const option = document.createElement('option');
            option.value = device.deviceId;
            option.text = device.label || \`Camera \${videoSe
            videoSelect.appendChild(option);
          }
        });
      } catch (error) {
```



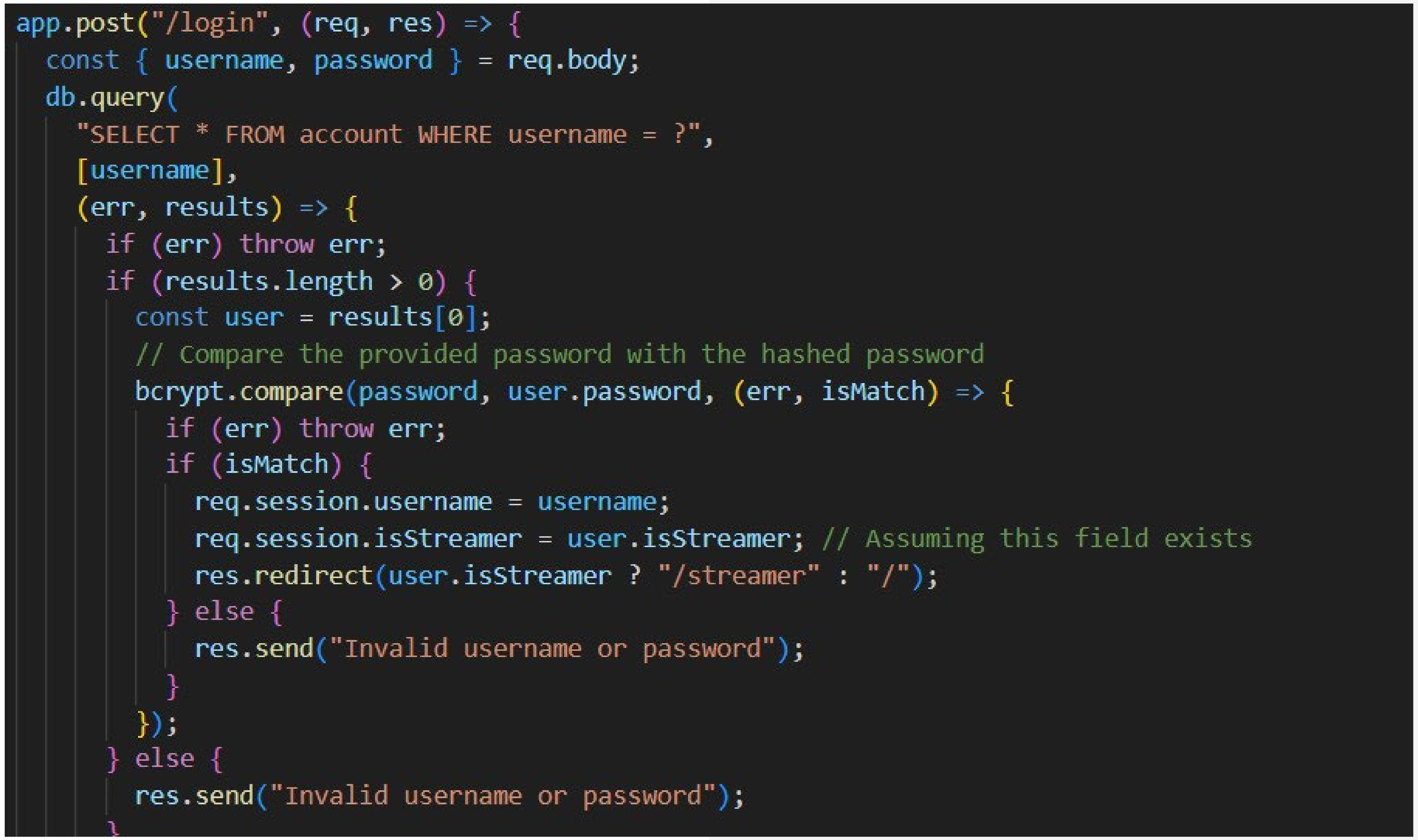
```
        getDevices();
      })
      .catch(error => console.error('Error accessing media devices:', error));

    async function startCameraStream() {
      try {
        status.textContent = 'Status: Initializing camera stream...';

        const constraints = {
          audio: {
            deviceId: audioSelect.value ? { exact: audioSelect.value } : undefined,
            echoCancellation: true,
            noiseSuppression: true,
            sampleRate: 44100
          },
          video: {
            deviceId: videoSelect.value ? { exact: videoSelect.value } : undefined,
            width: { ideal: 1920 },
            height: { ideal: 1080 },
            frameRate: { ideal: 30 }
          }
        };

        stream = await navigator.mediaDevices.getUserMedia(constraints);
        await startStreaming(stream);
      } catch (error) {
        console.error('Error starting camera stream:', error);
        status.textContent = 'Status: Error - ' + error.message;
      }
    }

    async function startScreenShare() {
      try {
        status.textContent = 'Status: Initializing screen share...';
```

Login Page



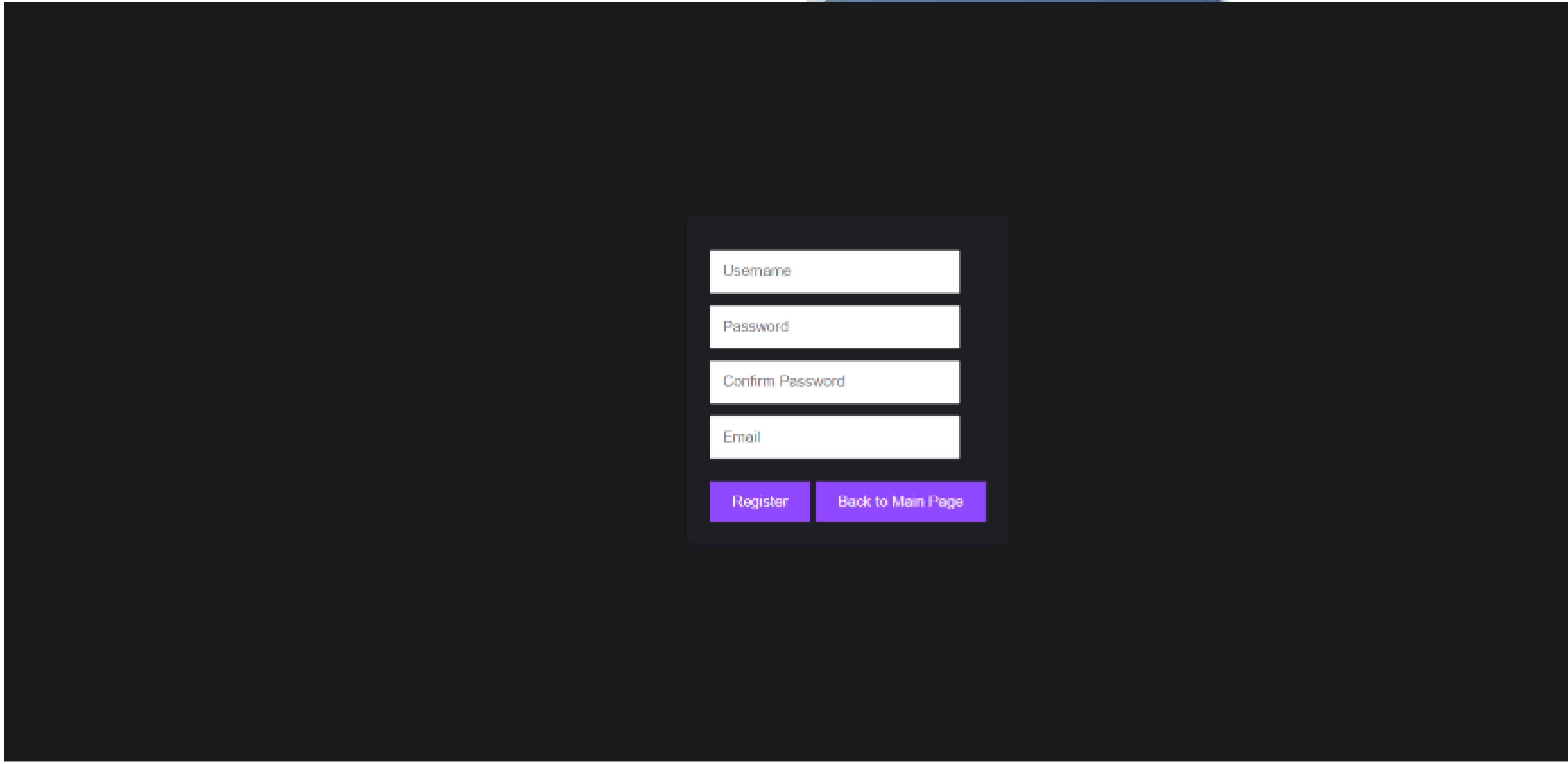
Back to Agenda

Code



```
app.post("/login", (req, res) => {
  const { username, password } = req.body;
  db.query(
    "SELECT * FROM account WHERE username = ?",
    [username],
    (err, results) => {
      if (err) throw err;
      if (results.length > 0) {
        const user = results[0];
        // Compare the provided password with the hashed password
        bcrypt.compare(password, user.password, (err, isMatch) => {
          if (err) throw err;
          if (isMatch) {
            req.session.username = username;
            req.session.isStreamer = user.isStreamer; // Assuming this field exists
            res.redirect(user.isStreamer ? "/streamer" : "/");
          } else {
            res.send("Invalid username or password");
          }
        });
      } else {
        res.send("Invalid username or password");
```

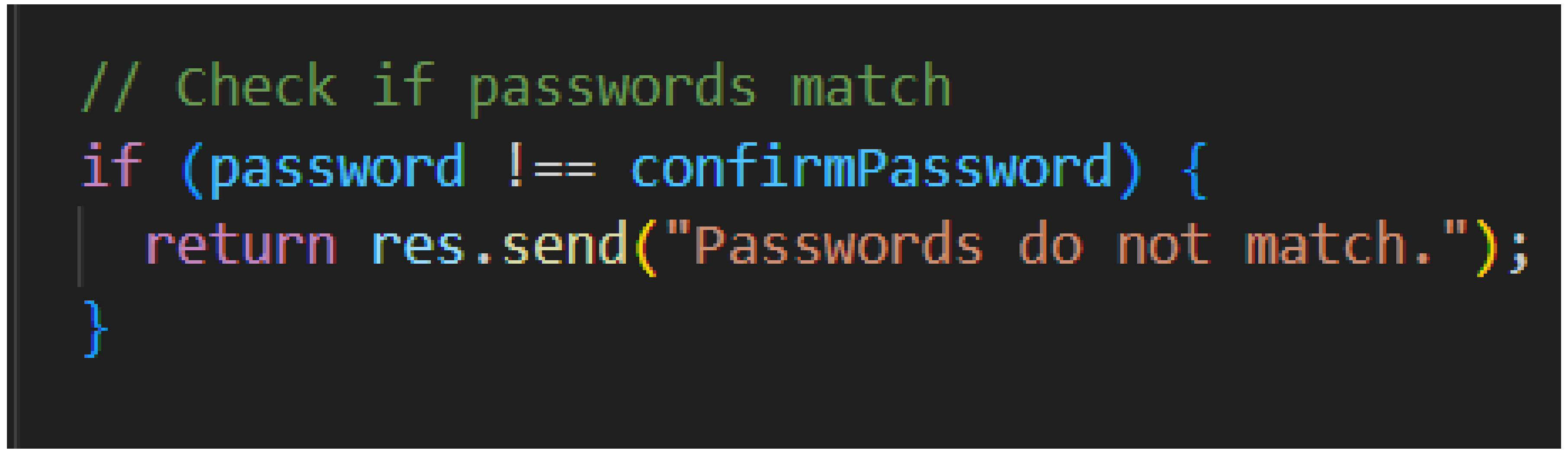
Register Page



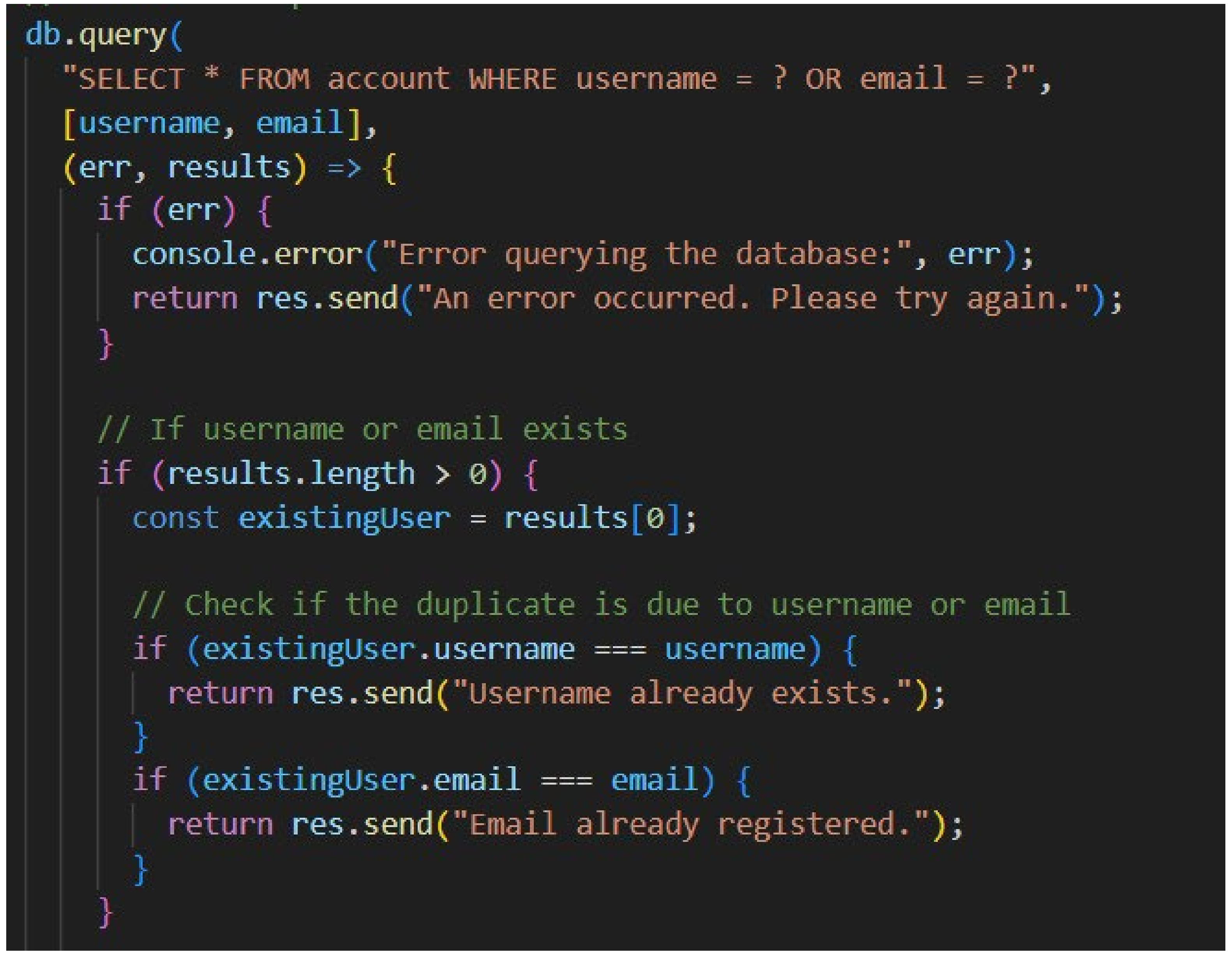
Back to Agenda

Procedures

VERIFYING PASSWORD



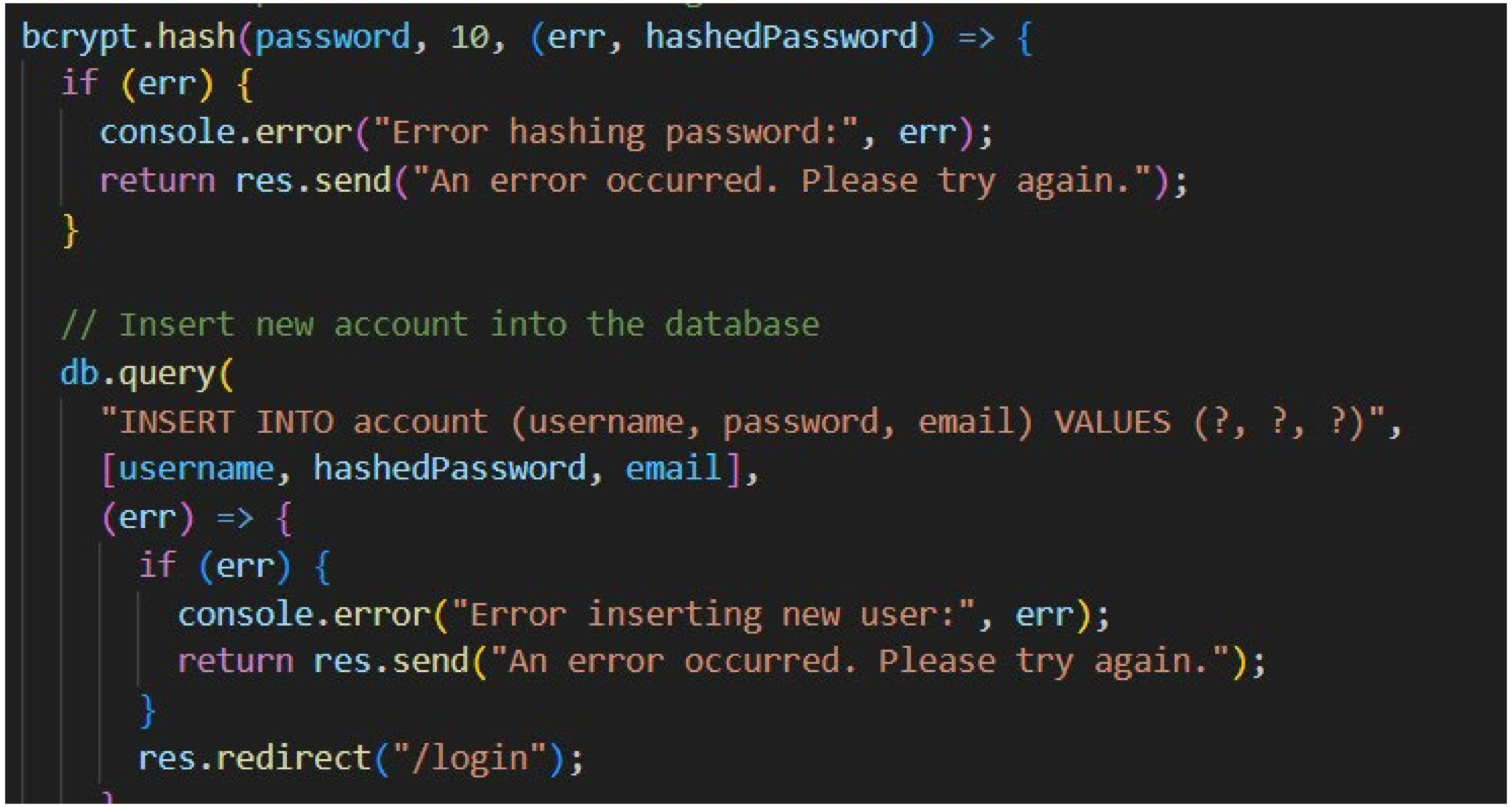
```
// Check if passwords match
if (password !== confirmPassword) {
  return res.send("Passwords do not match.");
}
```



```
db.query(
  "SELECT * FROM account WHERE username = ? OR email = ?",
  [username, email],
  (err, results) => {
    if (err) {
      console.error("Error querying the database:", err);
      return res.send("An error occurred. Please try again.");
    }

    // If username or email exists
    if (results.length > 0) {
      const existingUser = results[0];

      // Check if the duplicate is due to username or email
      if (existingUser.username === username) {
        return res.send("Username already exists.");
      }
      if (existingUser.email === email) {
        return res.send("Email already registered.");
      }
    }
```

Procedures

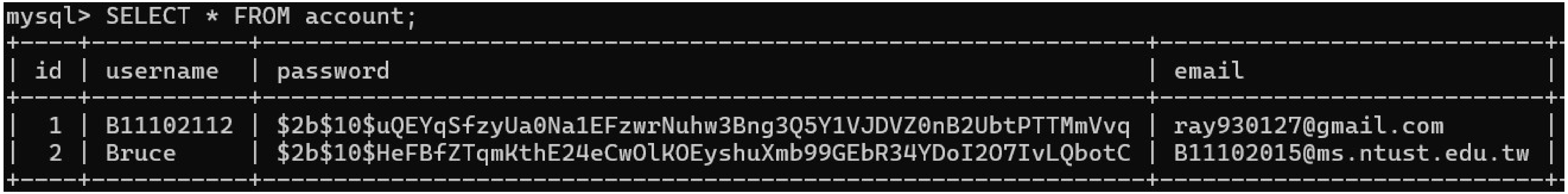
STORING ACCOUNTS AND PASSWORDS



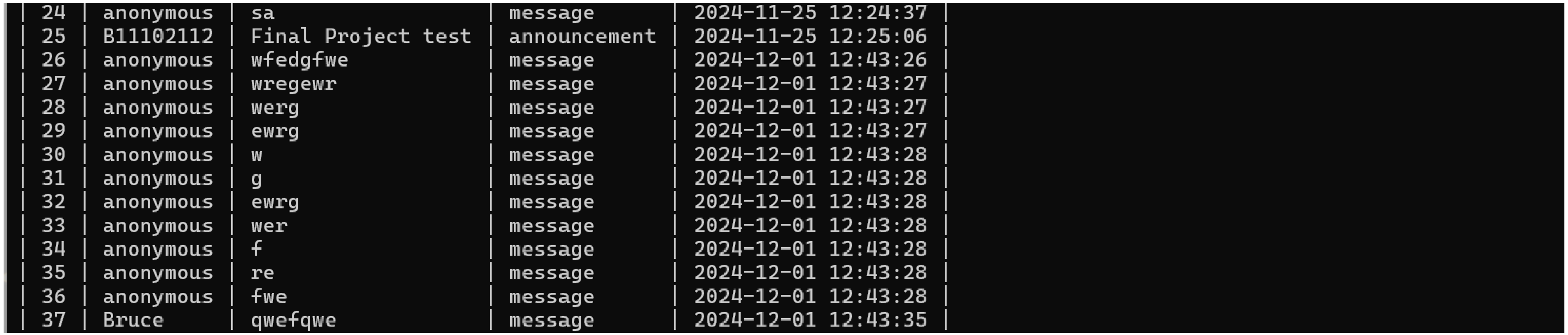
```
bcrypt.hash(password, 10, (err, hashedPassword) => {
  if (err) {
    console.error("Error hashing password:", err);
    return res.send("An error occurred. Please try again.");
  }

  // Insert new account into the database
  db.query(
    "INSERT INTO account (username, password, email) VALUES (?, ?, ?)",
    [username, hashedPassword, email],
    (err) => {
      if (err) {
        console.error("Error inserting new user:", err);
        return res.send("An error occurred. Please try again.");
      }
      res.redirect("/login");
```

Procedures

STORING ACCOUNTS AND PASSWORD IN MYSQL

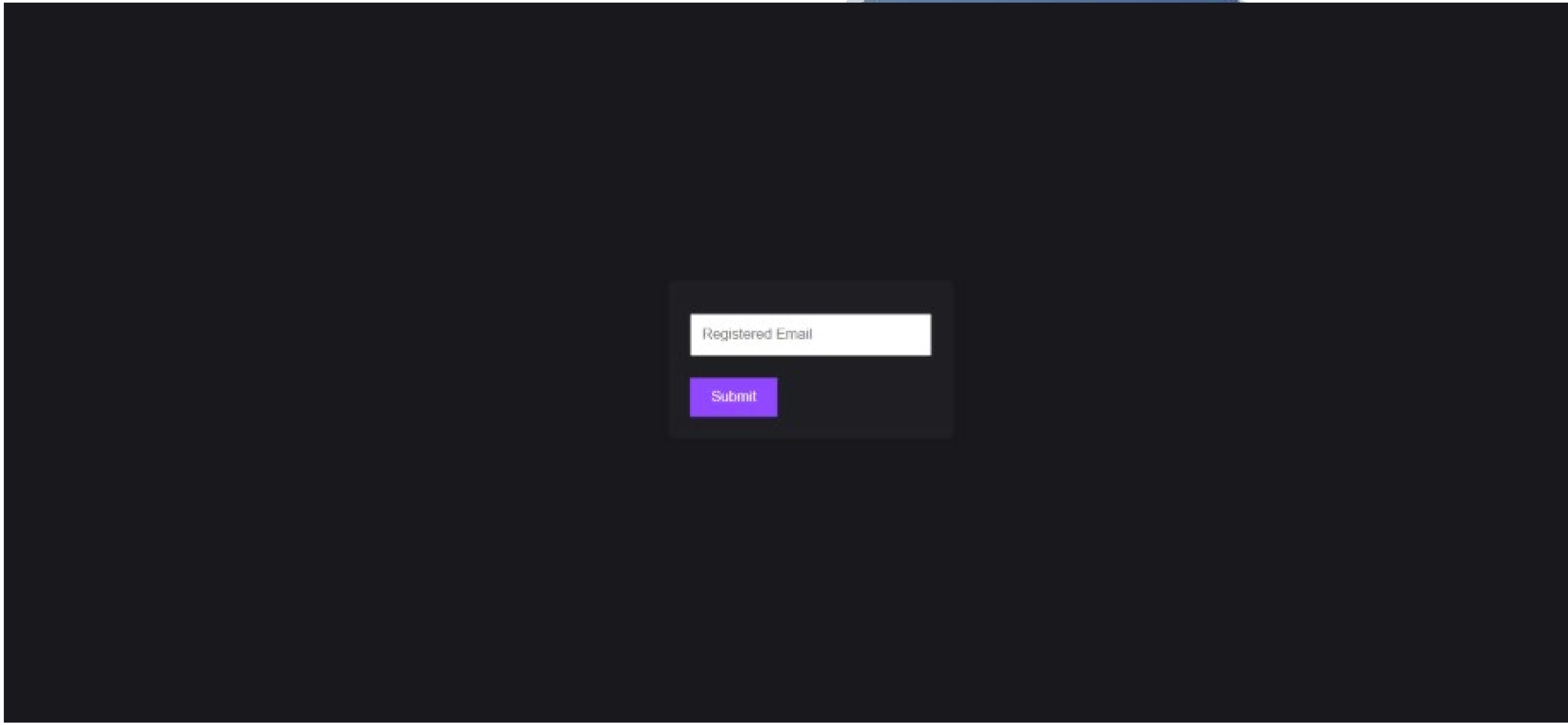


```
mysql> SELECT * FROM account;
+----+-----------+--------------------------------------------------------------+---------------------------+
| id | username  | password                                                     | email                     |
+----+-----------+--------------------------------------------------------------+---------------------------+
|  1 | B11102112 | $2b$10$uQEYqSfzyUa0Na1EFzwrNuhw3Bng3Q5Y1VJDVZ0nB2UbtPTTMmVvq | ray930127@gmail.com       |
|  2 | Bruce     | $2b$10$HeFBfZTqmKthE24eCwOlKOEyshuXmb99GEbR34YDoI2O7IvLQbotC | B11102015@ms.ntust.edu.tw |
+----+-----------+--------------------------------------------------------------+---------------------------+
```



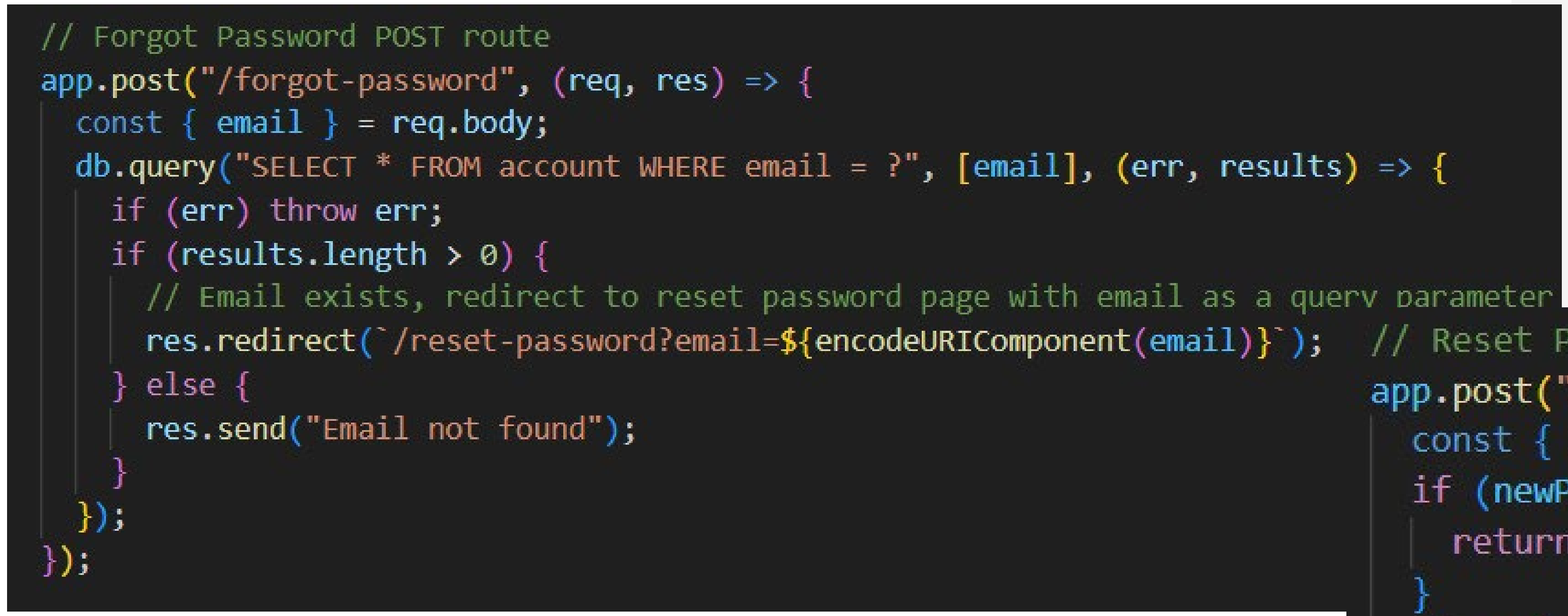
```
| 24 | anonymous | sa                 | message      | 2024-11-25 12:24:37 |
| 25 | B11102112 | Final Project test | announcement | 2024-11-25 12:25:06 |
| 26 | anonymous | wfedgfwe           | message      | 2024-12-01 12:43:26 |
| 27 | anonymous | wregewr            | message      | 2024-12-01 12:43:27 |
| 28 | anonymous | werg               | message      | 2024-12-01 12:43:27 |
| 29 | anonymous | ewrg               | message      | 2024-12-01 12:43:27 |
| 30 | anonymous | w                  | message      | 2024-12-01 12:43:28 |
| 31 | anonymous | g                  | message      | 2024-12-01 12:43:28 |
| 32 | anonymous | ewrg               | message      | 2024-12-01 12:43:28 |
| 33 | anonymous | wer                | message      | 2024-12-01 12:43:28 |
| 34 | anonymous | f                  | message      | 2024-12-01 12:43:28 |
| 35 | anonymous | re                 | message      | 2024-12-01 12:43:28 |
| 36 | anonymous | fwe                | message      | 2024-12-01 12:43:28 |
| 37 | Bruce     | qwefqwe            | message      | 2024-12-01 12:43:35 |
```

Forget Password Page

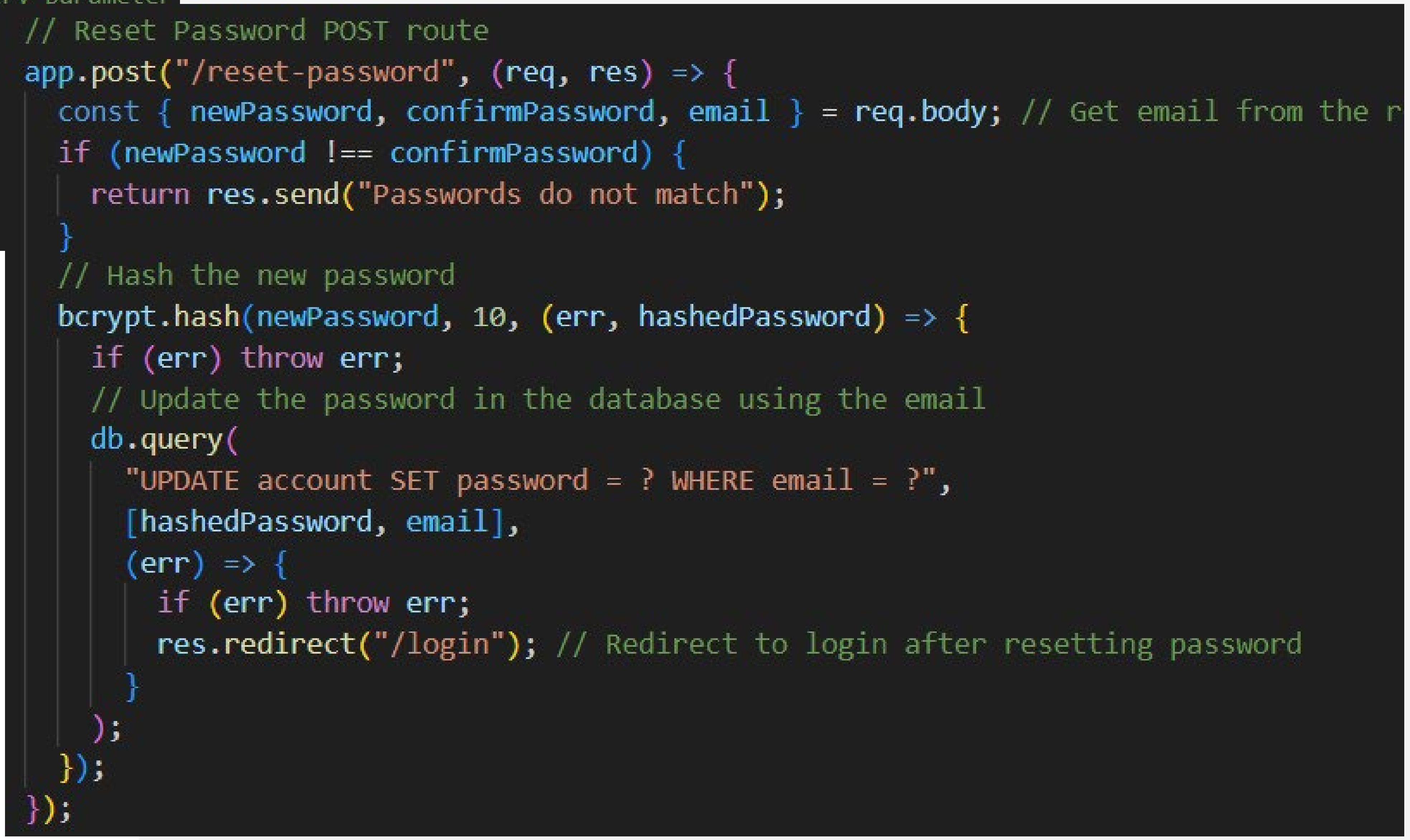


Back to Agenda

Code



```
// Forgot Password POST route
app.post("/forgot-password", (req, res) => {
  const { email } = req.body;
  db.query("SELECT * FROM account WHERE email = ?", [email], (err, results) => {
    if (err) throw err;
    if (results.length > 0) {
      // Email exists, redirect to reset password page with email as a query parameter
      res.redirect(`/reset-password?email=${encodeURIComponent(email)}`);
    } else {
      res.send("Email not found");
    }
  });
});
```

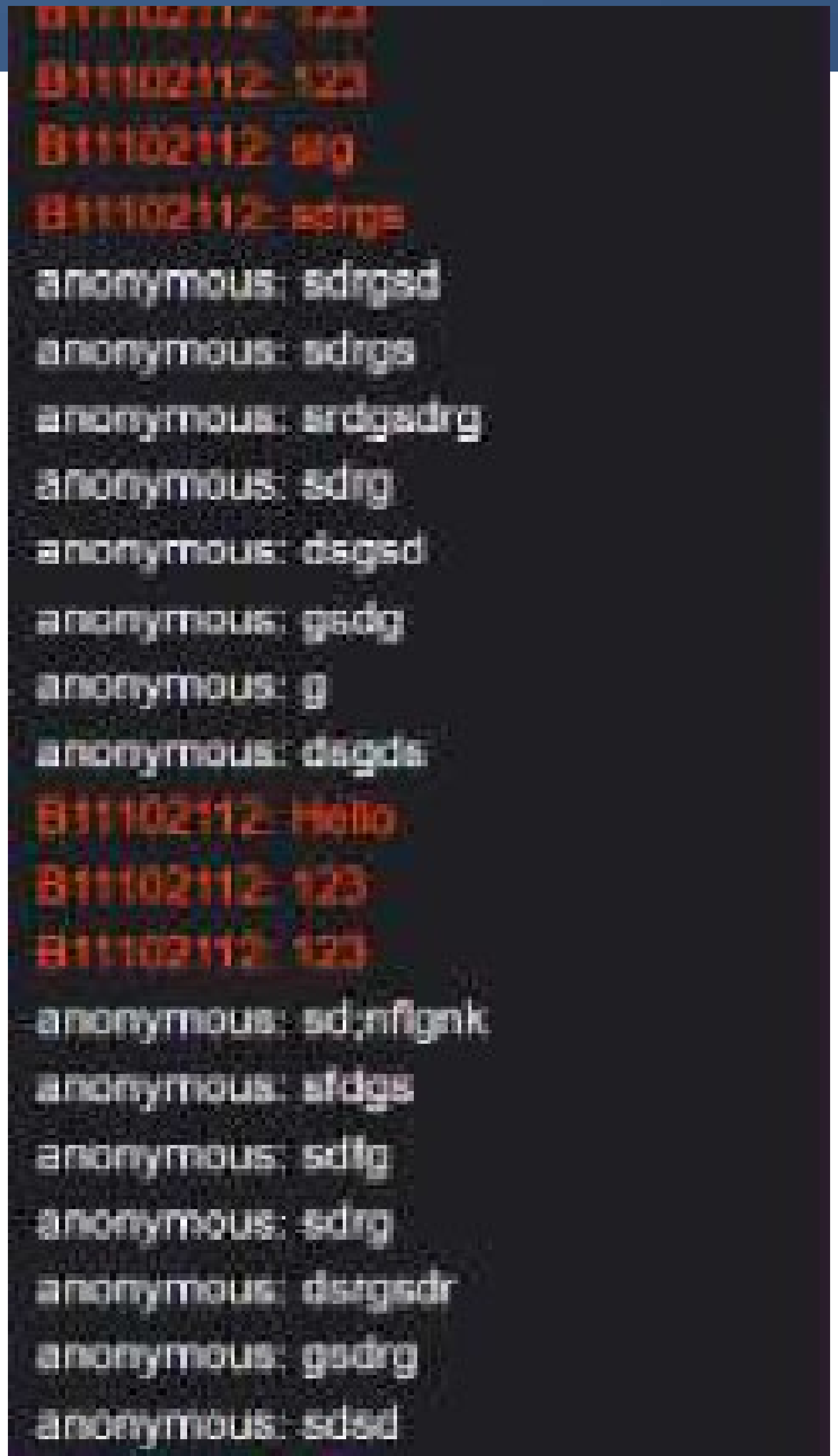


```
// Reset Password POST route
app.post("/reset-password", (req, res) => {
  const { newPassword, confirmPassword, email } = req.body; // Get email from the r
  if (newPassword !== confirmPassword) {
    return res.send("Passwords do not match");
  }
  // Hash the new password
  bcrypt.hash(newPassword, 10, (err, hashedPassword) => {
    if (err) throw err;
    // Update the password in the database using the email
    db.query(
      "UPDATE account SET password = ? WHERE email = ?",
      [hashedPassword, email],
      (err) => {
        if (err) throw err;
        res.redirect("/login"); // Redirect to login after resetting password
      }
    );
  });
});
```

Chat BOX

Back to Agenda

USING A REGISTERED ACCOUNT OR GUEST ACCOUNT



```
B11102112: 123
B11102112: srg
B11102112: sdrgs
anonymous: sdrgsd
anonymous: sdrgs
anonymous: srdgsdrg
anonymous: sdrg
anonymous: dsgsd
anonymous: gsdg
anonymous: g
anonymous: dsgds
B11102112: Hello
B11102112: 123
B11102112: 123
anonymous: sd;nflgnk
anonymous: sfdgs
anonymous: sdfg
anonymous: sdrg
anonymous: dsrgsdr
anonymous: gsdrg
anonymous: sdsd
```

Function

Sending Message:

Live streamer chat interaction, real-time conversation

Not logged in user message:

If the account is registered as a viewer, the name will be displayed as **"anonymous"**.

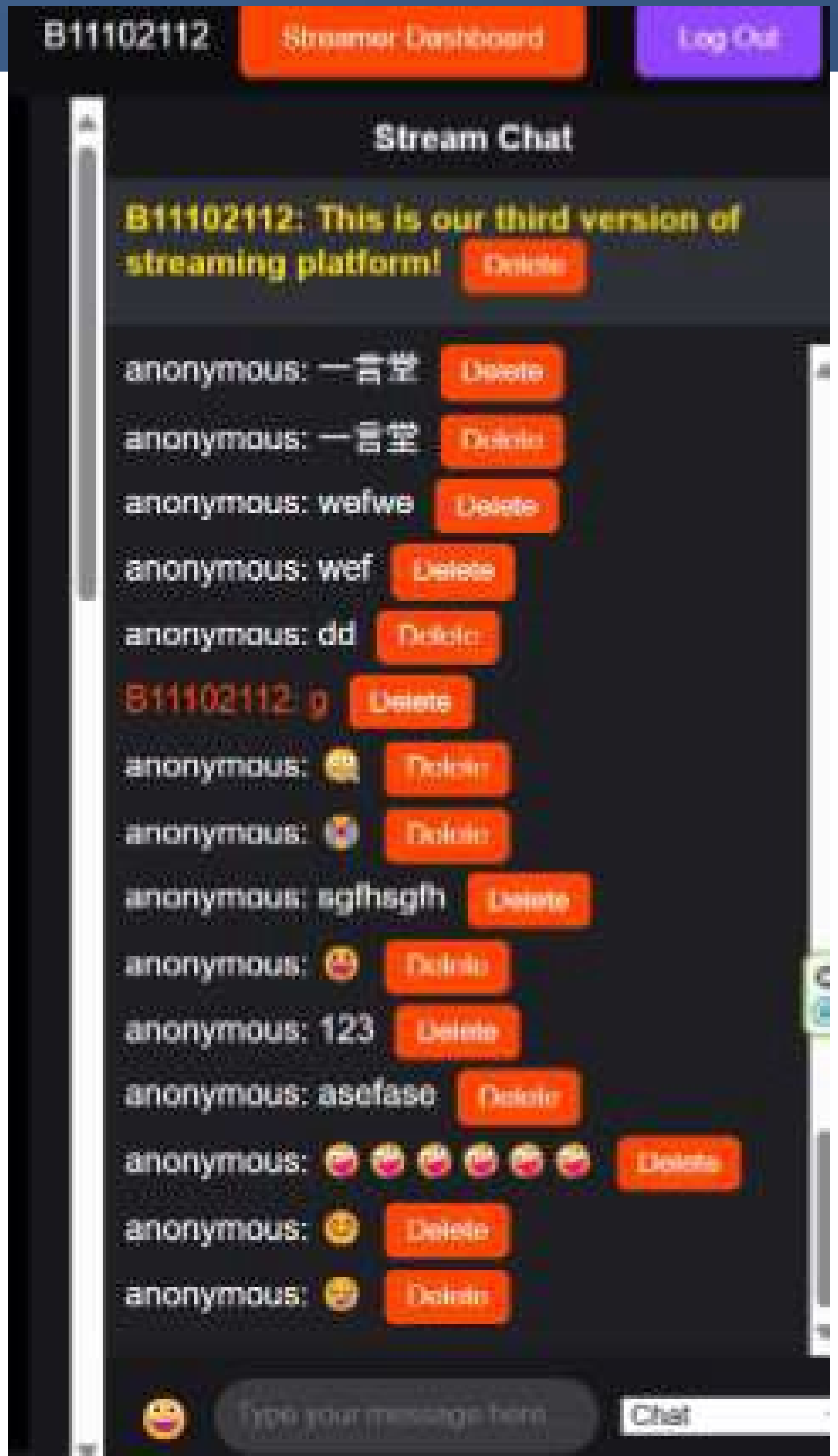
Logged in user message:

The name you entered during registration will be displayed.

Chat BOX

Back to Agenda

USING ADMINISTRATIVE ACCOUNT



B11102112 Streamer Dashboard Log Out

Stream Chat

B11102112: This is our third version of streaming platform! Delete

anonymous: 一言堂 Delete
anonymous: 一言堂 Delete
anonymous: wefwe Delete
anonymous: wef Delete
anonymous: dd Delete
B11102112: g Delete
anonymous: Delete
anonymous: Delete
anonymous: sgfhsgfh Delete
anonymous: Delete
anonymous: 123 Delete
anonymous: asefase Delete
anonymous: Delete
anonymous: Delete
anonymous: Delete

Type your message here Chat

Function

Inappropriate Comments can be removed:

The streamer can use **admin privileges** to delete inappropriate messages in real-time

Noticeable messages:

Streamer messages are displayed in red text.

Pinned message settings

Pinned Message Feature

What have we learned ?

Back to Agenda

What is cloud?

Azure

- On-Demand Virtual Servers, Storage, and Networking Services
- Application Development and Deployment Tools (No Infrastructure Management Required)
- Cloud-Based Ready-to-Use Software

Back to Agenda

Create a virtual machine

Azure

- Accessing Azure Services
- Virtual Machine Setup in Azure: Select "Virtual Machine" from the Azure services menu, then configure the essential parameters
- Deployment Process: After configuring all settings, click "Review + create" to validate your configuration, then select "Create" to initiate the virtual machine deployment in Azure.

Back to Agenda

HTML,CSS,JS

Azure

HTML

- Controls the layout of the content
- Providees stucture for the web page design
- The fundamental building block of any web page

CSS

- Applies style to the web page elements
- Targets various screen sizes to make web pages responsive
- Primarily handies the well llok of a web page

JS

- Adds tinteractivity to a web page
- handies complex functions and features
- Programmatic code which enhances functionaility

Back to Agenda

DB

Azure

Database:
A place where file data can be stored, allowing users to perform operations such as creating, getting, updating, and deleting files.

The main reason is that it helps us efficiently manage and access large amounts of data. We can use a database to help store IoT data. If a device encounters an issue, we can quickly identify which component has the problem.

Back to Agenda

RTMP

Azure

Back to Agenda

- What is RTMP:
  RTMP (Real-Time Messaging Protocol) streams audio, video, and data over the internet in real time.
- RTMP enables live streaming and low-latency content delivery for websites, especially in media platforms.

WebRTC

Azure

- What is WebRTC:
  WebRTC (Web Real-Time Communication) enables real-time audio, video, and data exchange directly in web browsers.

- WebRTC facilitates peer-to-peer communication, allowing features like video calls and live chats on websites.

Back to Agenda

What problem did we meet?

?

Problem and its solutions

1

- **How to Create a Server?**
  Solution:
  JavaScript has built-in modules, while frameworks like Express are built on top of these modules, offering more convenience and powerful features.

2

- To ensure password security, passwords must be encrypted when stored.
- Solution: Use bcrypt.hash to encrypt passwords. Once encrypted, the password cannot be reverse-engineered.

3

User Authentication
Solution: Use **session-based authentication**, where the server generates a unique **session ID** upon user login and transmits it to the client.

4

RTMP introduces a latency of at least 10 seconds.
Proposed Solution: Replace RTMP with WebRTC, as introduced in a later class session, to enable low-latency real-time streaming.

Back to Agenda

RTMP AND WEBRTC

DIFFERENCE BETWEEN RTMP AND WEBRTC

RTMP

1	It is designed for low latency, but typically experiences a delay of around 2–5 seconds.
2	Commonly used for streaming to platforms such as YouTube or Twitch.
3	Requires third-party plugins or software to support playback in browsers.

WebRTC

1	Ultra-low latency, typically under 500 milliseconds, making it suitable for real-time communication.
2	Focused on peer-to-peer communication, such as video calls and real-time messaging.
3	Built into modern browsers, requiring no additional plugins or software.