

Final Report

電子三甲 余秉修

電子三乙 李家睿

電子三丙 曾國竣

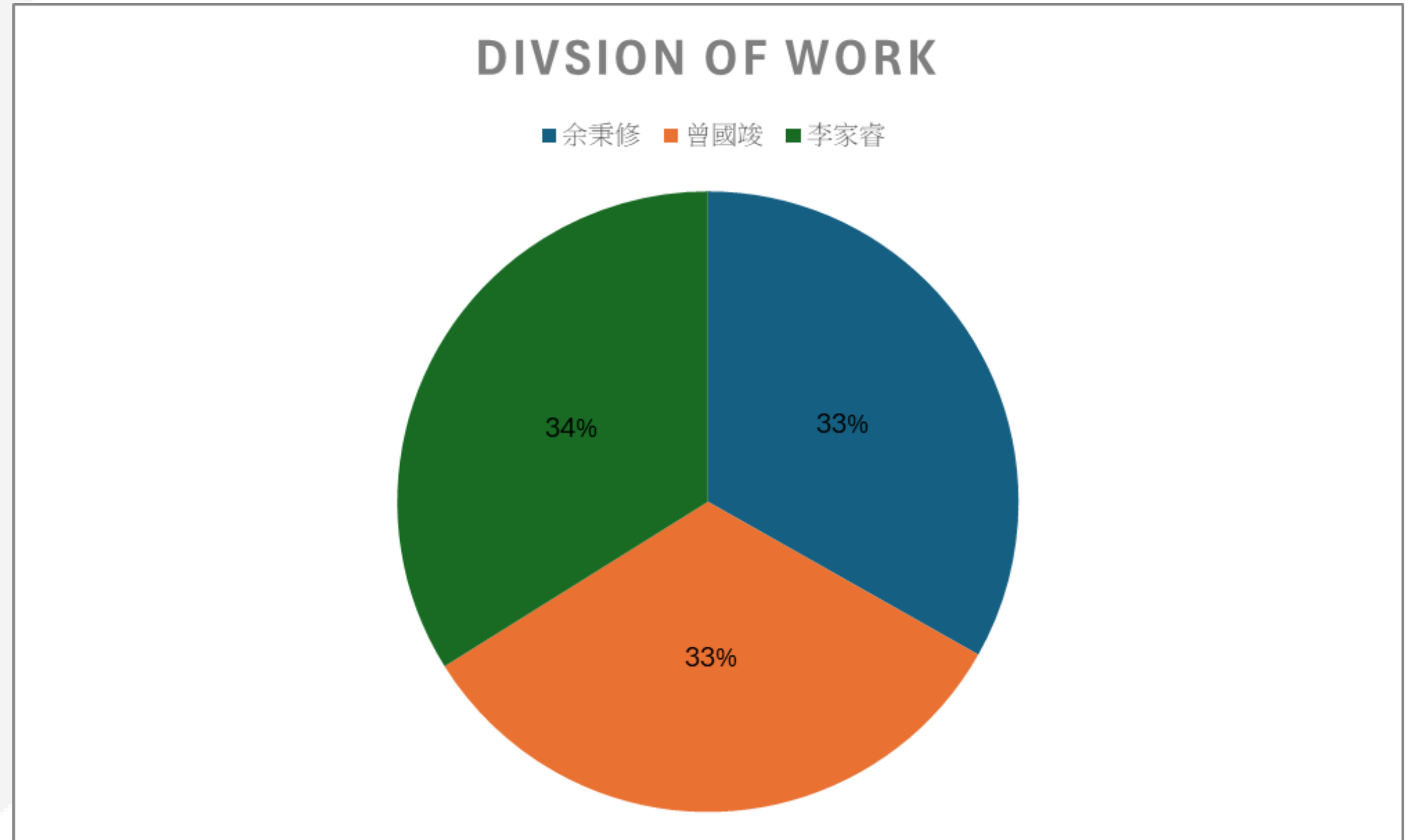
Table of contents

Division of work	3
System Architecture	4
Design Concept	7
What have we learned	20
What problem did we meet	27

Division of work

分工

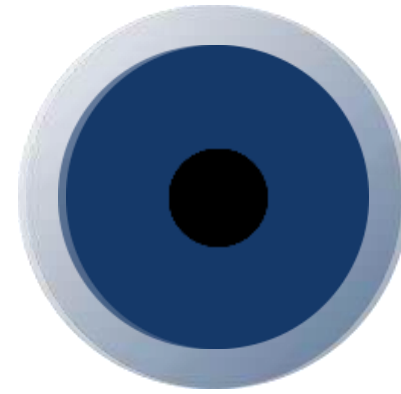
[Back to Agenda](#)



System Architecture

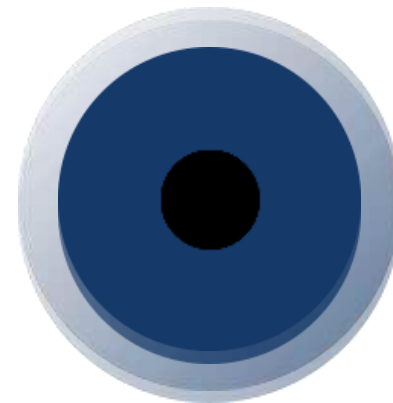
網頁程式

[Back to Agenda](#)



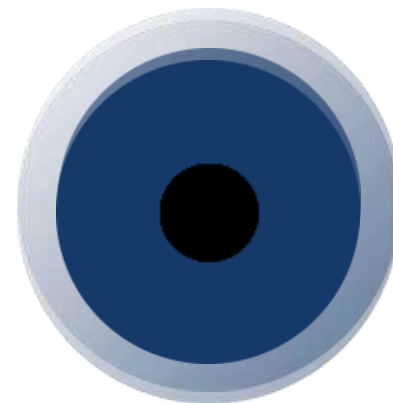
.HTML

是一種用於建立網頁的標準標記語言。HTML 是一種基礎技術，常與CSS、JavaScript一起被眾多網站用於設計網頁、網頁應用程式以及行動應用程式的使用者介面



.CSS

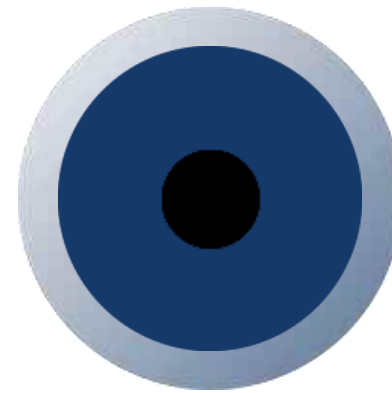
是一種用來為結構化文件（如HTML文件或XML應用）添加樣式（字型、間距和顏色等）的電腦語言，由W3C定義和維護。



.JS

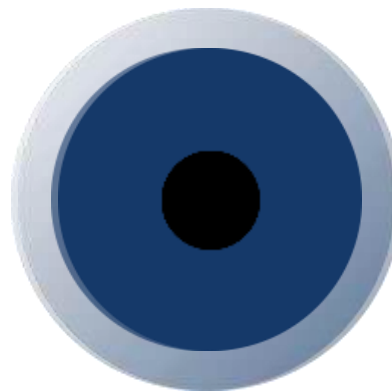
可以用來操作HTML文件的內容和結構，動態地更新頁面而無需重新載入。它可以回應使用者的操作，例如點擊按鈕、輸入文字等，從而改變頁面的外觀和行為。

System Architecture



Node.js

Node.js 是一個基於 JavaScript 的後端執行環境，使用 Chrome V8 JavaScript 引擎運行代碼。專注於提供非阻塞（non-blocking）和事件驅動（event-driven）的架構，適合構建高效能、可擴展的網路應用程式。Node.js 是現代 Web 開發的重要技術之一，尤其在實時應用和 API 開發中廣泛使用。

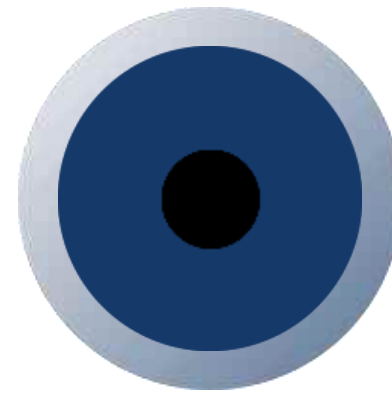


Express

Express.js 是一個基於 Node.js 的快速、靈活且極簡的 Web 應用框架，主要用於構建 Web 應用和 API，提供了強大的工具集來處理路由、請求、回應和中間件。

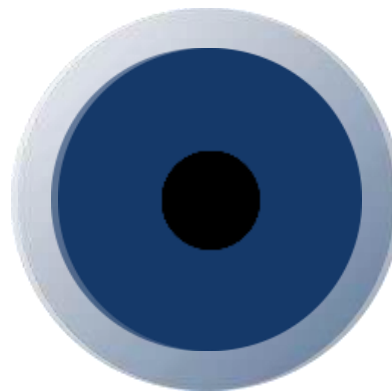
[Back to Agenda](#)

System Architecture



MySQL

是一種開源(免費)的 數據庫、資料庫管理系統，廣泛應用在中小型的網站中，用來配合如 PHP、ASP或ASP.NET等網頁程式語言，儲存大量數據，若網站擁有後端管理程式系統(網站後台)，多須配合資料庫功能。資料庫是用來放置大量資料與檔案的一個倉庫，SQL是跟網站倉庫溝通的管員，而MySQL是用來管理倉庫的系統。

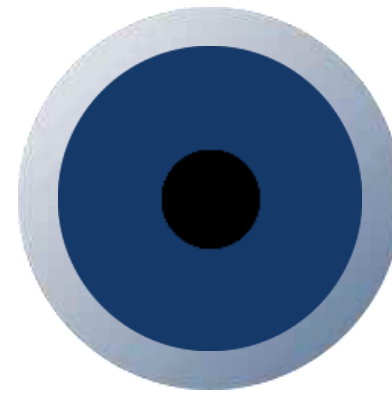


WebRTC

WebRTC 是一個 Open source project，主要在讓網頁瀏覽器和手機應用程式能夠透過簡單的 JavaScript API 進行即時的聲音影像通訊和數據傳輸。

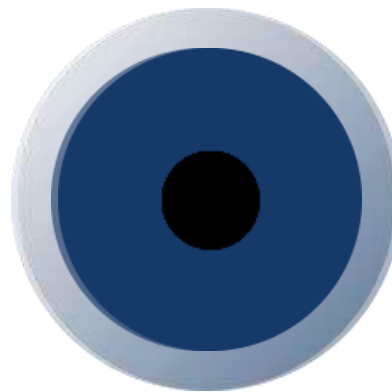
[Back to Agenda](#)

System Architecture



Azure

用戶可快速部署並管理網站環境，支援多種開發框架和工具，適用於測試、部署及擴充 Web 應用程式。透過整合 Azure 服務，開發者能輕鬆實現高效能、安全性及可擴展性的網站解決方案。



Socket

Socket 為網站開發提供即時通訊的基礎，允許伺服器與客戶端之間建立持久的連接。它支援雙向數據傳輸，適用於聊天室、即時通知及多人互動應用程式的開發。透過高效、低延遲的通訊方式，Socket 為現代 Web 應用程式實現即時功能提供強大支援。

[Back to Agenda](#)

Design Concept

設計理念、想法

[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

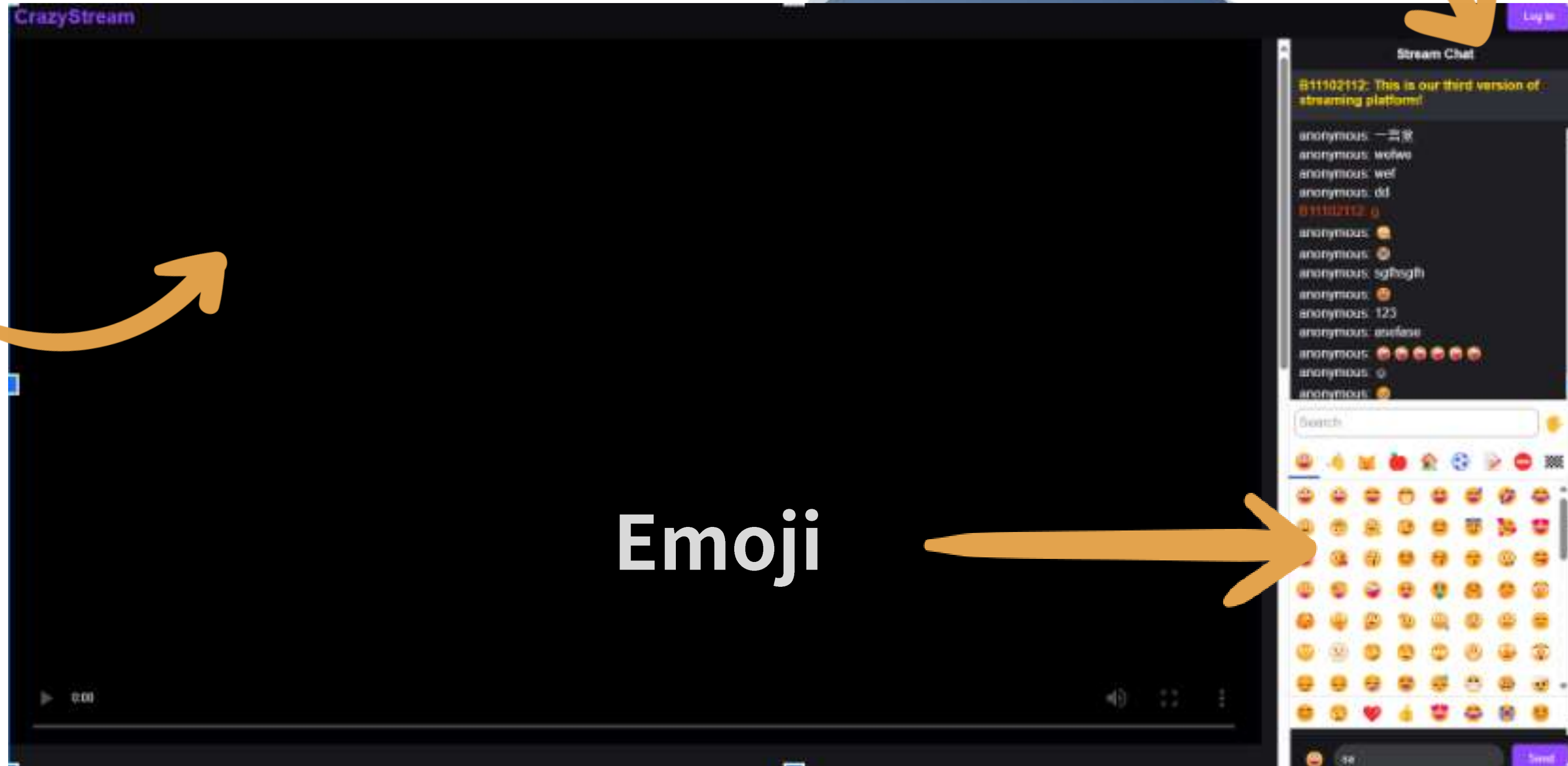
Main Streaming
Page

Chat Box

Streaming
Screen

Emoji

[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;
        }
      </style>
    </html>
  `);
});
```

```
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

```
<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>
```

```
<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>
const socket = io('https://' + window.location.hostname + ':5000', {
  secure: true,
  rejectUnauthorized: false,
  transports: ['websocket', 'polling']
});
let peerConnection;
const remoteVideo = document.getElementById('remoteVideo');

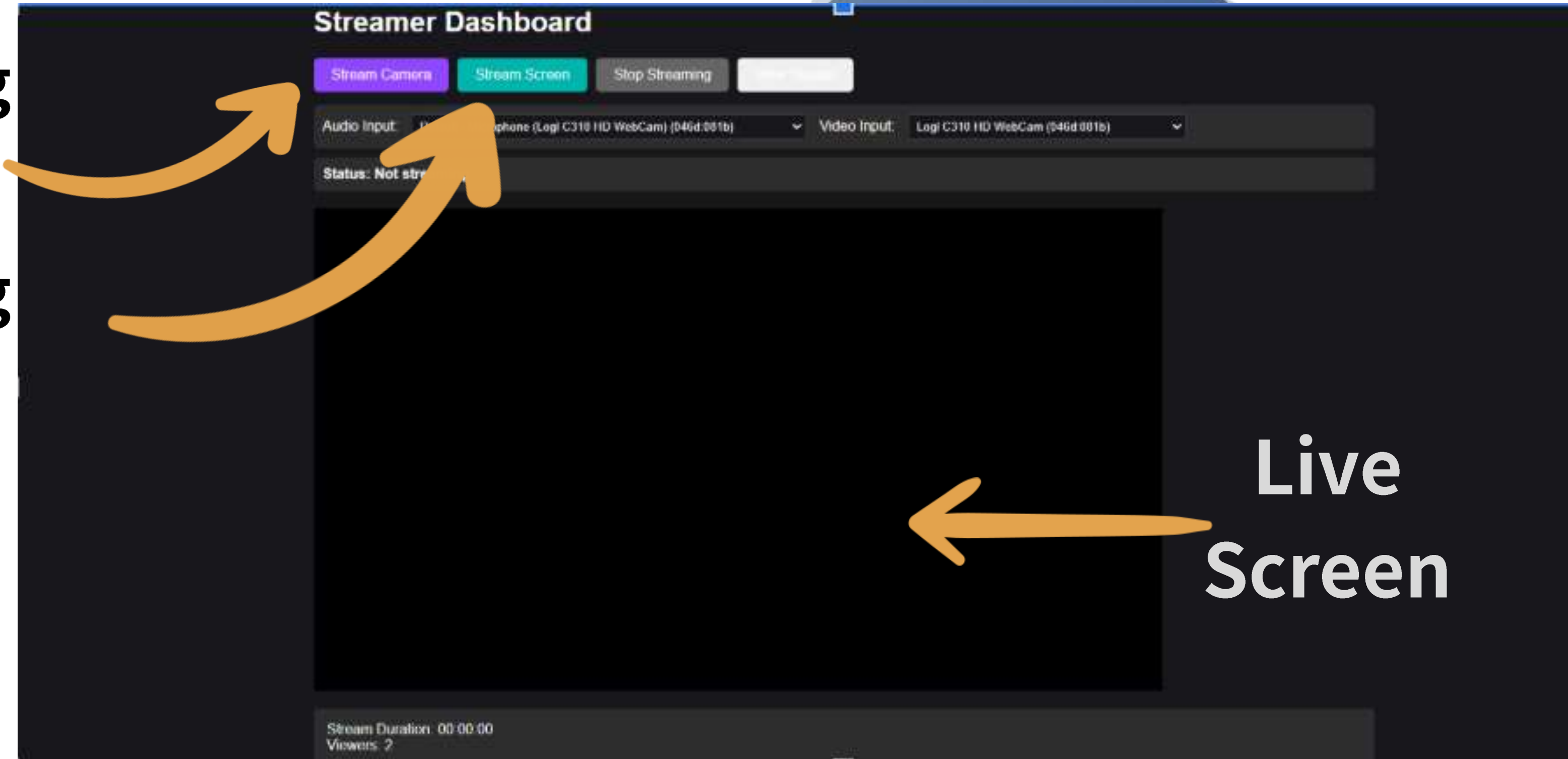
const configuration = {
```

[Back to Agenda](#)

Streamer DashBoard

**Streaming
Camera**

**Streaming
Screen**



[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: #181818; 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-color: #2D2D2D; padding: 5px; }
    .source-select select { padding: 5px; background-color: #181818; color: white; }
    .preview-container { width: 100%; max-width: 960px; aspect-ratio: 16/9; background-color: #000; }
    #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; }
    #status { margin: 10px 0; font-weight: bold; padding: 10px; border-radius: 5px; }
    .stats { margin-top: 20px; padding: 15px; background-color: #2D2D2D; border-radius: 5px; }
    .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>
      </div>
    </div>
  </div>
</body>
</html>
```

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

const configuration = {
  iceServers: [
    { 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.enumerateDevices();
    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 \${device.deviceId}`;
        audioSelect.appendChild(option);
      }
      if (device.kind === 'videoinput') {
        const option = document.createElement('option');
        option.value = device.deviceId;
        option.text = device.label || `Camera \${device.deviceId}`;
        videoSelect.appendChild(option);
      }
    });
  } catch (error) {
    console.error('Error accessing media devices:', 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;
  }
}

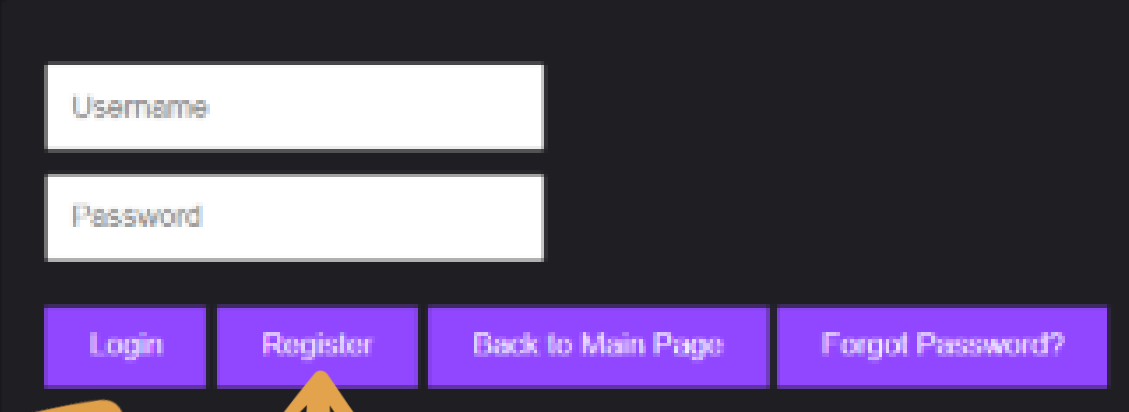
async function startScreenShare() {
  try {
    status.textContent = 'Status: Initializing screen share...';
```

Login Page

Login

Register

Forget
Password



A login form with two input fields: 'Username' and 'Password'. Below the fields are four buttons: 'Login', 'Register', 'Back to Main Page', and 'Forgot Password?'. The 'Login' and 'Register' buttons are highlighted with orange arrows pointing to them from the labels 'Login' and 'Register' respectively. The 'Forgot Password?' button is highlighted with an orange arrow pointing to it from the label 'Forget Password'.

[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\$HeFBfZTqmKthE24eCw0lK0EyshuXmb99GEbR34YDoI207IvLQbotC	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

Submit

[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)}`); // Reset Password POST route
    } else {
      res.send("Email not found");
    }
  });
});
```

```
app.post("/reset-password", (req, res) => {
  const { newPassword, confirmPassword, email } = req.body; // Get email from the request
  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: 123
B11102112: sdg
B11102112: sdrgs
anonymous: sdrsgd
anonymous: sdrsg
anonymous: sdrsgsdr
anonymous: sdr
anonymous: dsrgs
anonymous: gsdg
anonymous: g
anonymous: dsrgs
B11102112: Hello
B11102112: 123
B11102112: 123
anonymous: sd;nflgnk
anonymous: sfdgs
anonymous: sdfg
anonymous: sdr
anonymous: dsrgsdr
anonymous: gsdrg
anonymous: sdsd

Function

Sending Message:

與實況主聊天互動，即時對話

Not logged in user message:

如果是以為註冊者帳號，名稱會顯示
為ANONYMOUS

Logged in user message:

會顯示你註冊時所填寫的名稱

Chat BOX

[Back to Agenda](#)

USING ADMINISTRATIVE ACCOUNT



表情符號

Function

Inappropriate Comments can be removed:

實況主可以使用管理權限刪除
不當言論

Noticeable messages:

實況主的訊息以紅字表示

Pinned message settings

置頂留言設置功能

What have we learned ?

我們從課堂、實作過程中學到什麼

What is cloud?

Azure

- ◆ 提供隨需的虛擬伺服器、儲存和網路服務
- ◆ 提供開發和部署應用程式的工具，無需管理基礎設施
- ◆ 透過網路提供即用型軟體

[Back to Agenda](#)

Create a virtual machine

Azure

- ◆ 進入 Azure 服務，點選「建立資源」
- ◆ 選擇「虛擬機器」，設定名稱、區域和規格
- ◆ 確認設定，點選「檢閱並建立」，然後「建立」

[Back to Agenda](#)

HTML, CSS, JS

Azure

[Back to Agenda](#)



- ◆ Controls the layout of the content
- ◆ Provides structure for the web page design
- ◆ The fundamental building block of any web page



- ◆ Applies style to the web page elements
- ◆ Targets various screen sizes to make web pages responsive
- ◆ Primarily handles the visual look of a web page



- ◆ Adds interactivity to a web page
- ◆ handles complex functions and features
- ◆ Programmatic code which enhances functionality

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



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.

[Back to Agenda](#)

WebRTC

Azure



What is WebRTC:

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



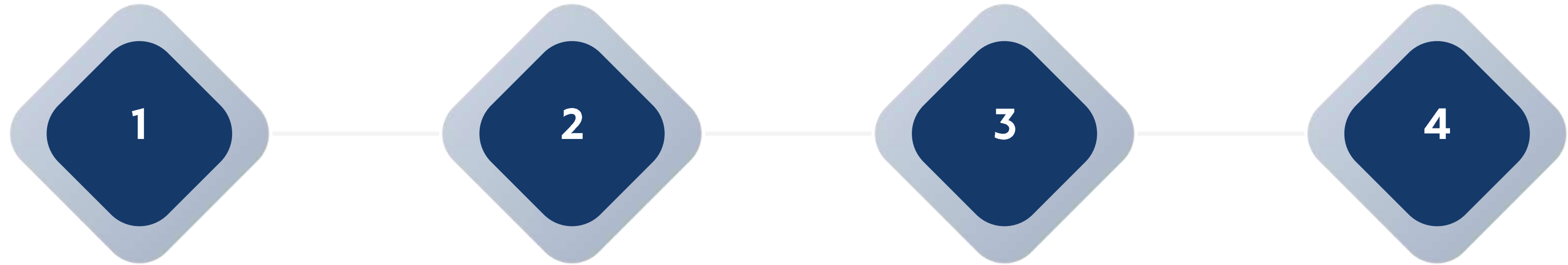
WebRTC facilitates peer-to-peer communication, allowing features like video calls and live chats on websites.

[Back to Agenda](#)

What problem did we meet?

我們遇到什麼困難？

Problem and its solutions



- 如何創建一個伺服器？
解決方案：JavaScript 有內置的http模塊，而 express框架是基於 http模塊封裝出來的，使用更方便，功能更強大。

- 為了保證密碼的安全性，需要對密碼進行加密存取。
- 解決方案：使用 bcrypt.hash可以對密碼進行加密，加密過後密碼不會被逆向破解。

對用戶進行身份驗證。
解決方案：使用Session來進行身份驗證，伺服器會在使用者登入時產生一個唯一的 Session ID，並把它傳送到客戶端。

RTMP會產生至少10秒以上的時間延遲。
解決方案:使用後來課堂上所教的WebRTC來進行直播串流。

RTMP AND WEBRTC

DIFFERENCE BETWEEN RTMP AND WEBRTC

RTMP

1	設計為低延遲，但通常約 2–5 秒
2	常用於直播至 YouTube 或 Twitch 等平台
3	需第三方外掛或軟體來支援瀏覽器播放

WebRTC

1	超低延遲，通常低於 500 毫秒，適合即時通訊
2	專注於點對點通訊，如視訊通話和即時聊天
3	內建於現代瀏覽器，無需