Media Streaming with IBM Cloud Video Streaming

TEAM MEMBER

312621243005:Badri Gopikrishna

Phase-5 Document submission

Project Title: Cloud Media Streaming with IBM Cloud Video Streaming

Problem Statement

Objective: Develop a seamless and reliable cloud-based media streaming platform using IBM Cloud Video Streaming.

Problem identified:

In today's digital age, the demand for high-quality, uninterrupted media streaming experiences is higher than ever. Whether it's for entertainment, education, or business purposes, users expect smooth and reliable playback across various devices and platforms. However, setting up a robust and scalable cloud media streaming solution can be a complex task, involving considerations for video encoding, content delivery, and user experience optimization.

Introduction:

In response to this growing need, our project, "Cloud Media Streaming with IBM Cloud Video Streaming," aims to create a cutting-edge platform for delivering high-quality video content over the internet. Leveraging the capabilities of IBM Cloud Video Streaming, we will design a solution that ensures seamless playback, regardless of the viewer's location or device.

The project starts by addressing the core issue: providing a reliable and scalable infrastructure for media streaming. This involves tasks such as video transcoding, adaptive bitrate streaming, content delivery network (CDN) integration, and user interface design. We will utilize the powerful features of IBM Cloud Video Streaming to handle these tasks efficiently.

Our project focuses on a well-structured process that spans from content upload and encoding to user interface customization and integration with third-party services. We will also implement features like analytics and user authentication to enhance the overall streaming experience.

In the following sections, we will delve into the intricacies of our project, describing the methods, tools, and technologies we utilize to build a robust cloud media streaming platform. Our goal is to address the increasing demand for high-quality video content delivery, empowering content creators and businesses to reach their audience with a seamless streaming experience.

Data: The primary data for this project will consist of various video files in different formats and resolutions. Additionally, we will collect user engagement metrics to analyze viewer behavior and optimize the streaming experience. This data will be instrumental in fine-tuning our platform for optimal performance.

LITERATURE SURVEY

1.” Cost Minimization of Cloud Services for

On-Demand Video Streaming”, Mahmoud Darwich [2022]

This research explores the utilization of cloud technology for video stream processing, leveraging the benefits of virtual machines and cost-effective storage servers. To cater to diverse user devices, multiple video formats are typically prepared, posing a challenge in terms of storage costs. This study presents an approach to optimize cloud storage by strategically determining which videos, in what formats, should be stored, thereby minimizing overall cloud service expenses. Promising results were obtained, demonstrating effectiveness with a growing number of frequently accessed videos and increasing view counts. The proposed method resulted in a noteworthy reduction of up to 22% in cloud service costs.

2.” Cost-Efficient Storage for On-Demand Video

Streaming on Cloud ”, Ishihara [2021]

This paper addresses the challenge of video transcoding, where videos are converted into various formats to accommodate different user devices. This process demands significant storage and resources. With the rise of cloud technology, video streaming companies have shifted to processing videos in the cloud. Traditionally, multiple pre-transcoded video formats are stored and streamed, leading to high storage costs. This paper proposes a solution for cost-effective video storage in the hierarchical cloud storage. The method optimizes video pre-transcoding decisions based on suitable cloud storage to minimize costs. Experimental results demonstrate the effectiveness of the approach, particularly in scenarios with a high percentage of frequently accessed videos, leading to potential cost reduction of up to 40%.

3.” PERFORMANCE ANALYSIS OF VIDEO ON-DEMAND AND LIVE VIDEO STREAMING USING CLOUD BASED SERVICES - 2021”, UJASH PATEL [2022]

The paper explores Cyber-Physical Systems (CPS) and their integration with video streaming for applications like smart grids and health monitoring. It emphasizes the importance of Quality of Experience (QoE), cost, and bandwidth impact on cloud-based video analysis for Video-On-Demand Streaming (VoDS) and Live Video Streaming (LVS). Content Delivery Networks (CDNs) are discussed as crucial for achieving optimal user experience across various cloud providers

4.” Improving Hierarchy Storage for Video Streaming in Cloud ”, Yasser Ismail [2021]

To address the cost implications of storing various video stream formats, this research focuses on utilizing cloud technology for efficient storage. Traditionally, storing multiple formats incurred high expenses. By adopting cloud services, video stream companies aim to mitigate costs. However, storing all streams in a single cloud type escalates costs, worsened by changing access patterns. To optimize storage costs, the paper introduces a method that leverages hierarchical cloud storage. The algorithm identifies frequently accessed video segments and stores them in the appropriate cloud storage type. Experiments demonstrate a promising 18.75% reduction in cloud storage costs through this approach.

5.” Point Cloud Video Streaming: Challenges and Solutions ”, Weishan Zhang [2021]

This article addresses the emerging field of volumetric video, essential for VR/AR/MR experiences and well-suited for advanced wireless communication like 5G. It emphasizes the need for efficient volumetric video streaming and focuses on point cloud video as a popular way to represent volumetric media. The article introduces point cloud video technology and its applications, outlines challenges and solutions in point cloud video streaming, and discusses encoding, tiling, viewing angle prediction, decoding, quality assessment, and transmission optimization. A preliminary MPEG DASH-based point cloud video streaming prototype is explained with simulation results, and future research directions are highlighted for high-quality point cloud video streaming.

DESIGN THINKING

Design Thinking Approach for Media Streaming with IBM Cloud Video Streaming

Empathize:

Gain insights into users' needs and expectations, focusing on viewers and content creators.

Actions:

- Conduct user surveys or interviews to understand preferences and pain points.
- Analyze market trends and competitors to identify key success factors.

- Seek input and feedback from experts in content delivery, user experience design, and cloud infrastructure.

Define:

Set clear objectives and success criteria.

Objectives:

- Develop a robust, scalable cloud-based platform using IBM Cloud Video Streaming.
- Ensure high-quality video playback and provide a user-friendly interface.
- Achieve high user satisfaction metrics, including increased user engagement, minimal buffering, and positive feedback.

Ideate:

Brainstorm innovative approaches leveraging IBM Cloud Video Streaming.

Actions:

- Explore IBM Cloud Video Streaming capabilities and integration with HTML, CSS, PHP, and JS for hosting and streaming content.
- Consider various streaming protocols and codecs to deliver high-quality video content through the platform.
- Brainstorm features like content recommendation algorithms, user-generated content integration, and social sharing functionalities within the framework of IBM Cloud Video Streaming.

Prototype:

Visualize the platform using HTML, CSS, PHP, JS with IBM Cloud Video Streaming.

Actions:

- Set up an IBM Cloud environment to test content upload, storage, and streaming capabilities.

- Develop a user interface prototype using HTML, CSS, PHP, and JS, integrating with IBM Cloud Video Streaming.
- Conduct usability testing with a small group of users to validate the initial design concepts and user experience.

Test:

Evaluate technical and user experience performance with IBM Cloud Video Streaming.

Actions:

- Test the IBM Cloud Video Streaming infrastructure for scalability, security, and performance under various loads.
- Gather user feedback on the prototype's usability, navigation, and overall satisfaction, specifically in conjunction with IBM Cloud Video Streaming.
- Use metrics such as video buffering rates, load times, and user engagement to assess the prototype's effectiveness with IBM Cloud Video Streaming.

Implement:

Build and deploy the platform using HTML, CSS, PHP, and JS in conjunction with IBM Cloud Video Streaming.

Actions:

- Integrate content upload, storage, and streaming components with IBM Cloud Video Streaming capabilities.
- Implement user authentication and authorization mechanisms for secure access, leveraging IBM Cloud Video Streaming features.
- Conduct comprehensive testing to ensure seamless content delivery and user interactions with IBM Cloud Video Streaming.

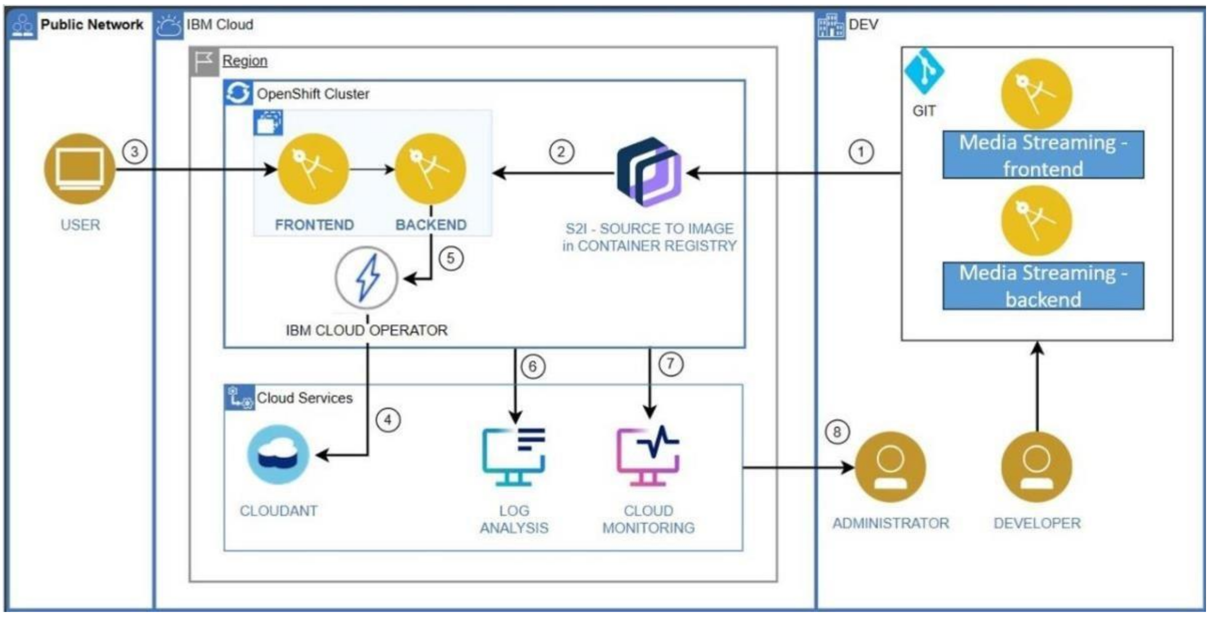
Iterate:

Continuously gather feedback and make improvements, in collaboration with IBM Cloud Video Streaming.

Actions:

- Monitor system performance, user engagement, and content popularity to identify areas for improvement, particularly leveraging IBM Cloud Video Streaming capabilities.
- Address user feedback and implement enhancements, such as refining recommendation algorithms or adding social sharing features, while integrating with IBM Cloud Video Streaming.
- Stay updated with emerging technologies and trends in media streaming to incorporate innovations into the platform, in collaboration with IBM Cloud Video Streaming.

TECHNOLOGY ARCHITECTURE



Design Thinking Approach for Media Streaming with IBM Cloud Video Streaming

1. Content Acquisition and Storage:

- Content Sources: Utilize various sources to acquire video content, including user uploads, licensed content, and streaming services.

- Cloud Storage: Store video files in IBM Cloud Object Storage for efficient retrieval and distribution.

2. Content Preprocessing:

- Quality Assurance: Ensure video content meets specified quality standards, addressing issues like resolution, aspect ratio, and compression artifacts.

- Transcoding: Convert videos into multiple formats and resolutions to accommodate various user devices and network conditions.

3. Platform Development and Integration:

- IBM Cloud Video Streaming: Leverage IBM Cloud Video Streaming for hosting and managing video content.

- API Integration: Integrate IBM Cloud Video Streaming APIs for seamless content upload, retrieval, and streaming.

4. User Interface (UI) Design:

- Create an intuitive UI for users to browse and interact with the media streaming platform.

- Implement features for content search, recommendations, and user-generated playlists.

5. Video Player Customization:

- Customize the video player using HTML, CSS, and JavaScript to provide a branded and user-friendly viewing experience.

- Implement features like adaptive bitrate streaming for optimal playback quality.

6. User Authentication and Authorization:

- Implement secure user authentication mechanisms to control access to premium content and user-specific features.
- Utilize IBM Cloud Identity and Access Management (IAM) for authentication.

7. Monetization (Optional):

- Implement monetization strategies such as subscription models, pay-per-view, or ad-based revenue streams using IBM Monetize.

8. Analytics and Insights:

- Integrate analytics tools to gather user engagement data, including views, watch time, and popular content.
- Utilize IBM Cloud Analytics for in-depth insights into user behavior and content performance.

9. Content Recommendation Engine (Optional):

- Develop a recommendation engine using machine learning algorithms to suggest personalized content to users.

10. Social Integration (Optional):

- Implement social sharing features to allow users to share content on social media platforms.

11. Content Moderation (Optional):

- Integrate content moderation tools to ensure compliance with community guidelines and prevent inappropriate or offensive content from being published.

12. Continuous Improvement and Innovation:

- Stay updated with emerging technologies in media streaming and leverage IBM Cloud services for ongoing enhancements.

- Gather user feedback and conduct usability testing to refine features and optimize the user experience.

MODULES DESCRIPTION

1. Content Acquisition and Storage Module:

- Objective: Acquire and store video content efficiently for the media streaming platform.

- Key Tasks:

- Gather video content from various sources, including user uploads and licensed content.

- Store video files in IBM Cloud Object Storage for easy retrieval.

2. Content Preprocessing Module:

- Objective: Prepare video content for optimal streaming quality and compatibility.

- Key Tasks:

- Ensure video quality meets specified standards, addressing resolution, aspect ratio, and compression.

- Transcode videos into multiple formats and resolutions for diverse user devices and network conditions.

3. Platform Development and Integration Module:

- Objective: Utilize IBM Cloud Video Streaming for hosting and managing video content.

- Key Tasks:

- Integrate IBM Cloud Video Streaming APIs for seamless content upload, retrieval, and streaming.

4. User Interface (UI) Design Module:

- Objective: Design an intuitive user interface for seamless interaction with the media streaming platform.

- Key Tasks:

- Create a user-friendly UI with features for content search, recommendations, and user-generated playlists.

5. Video Player Customization Module:

- Objective: Customize the video player to provide an enhanced viewing experience.

- Key Tasks:

- Utilize HTML, CSS, and JavaScript to customize the video player for branding and optimal playback.

6. User Authentication and Authorization Module:

- Objective: Implement secure access control mechanisms for user authentication and authorization.

- Key Tasks:

- Integrate IBM Cloud Identity and Access Management (IAM) for authentication and authorization.

7. Monetization Strategies Module (Optional):

- Objective: Implement revenue-generating models for the media streaming platform.

- Key Tasks:

- Introduce monetization strategies like subscription models, pay-per-view, or ad-based revenue streams.

8. Analytics and Insights Module:

- Objective: Gather and analyze user engagement data to gain valuable insights.

- Key Tasks:
- Integrate analytics tools to track views, watch time, and popular content.
- Leverage IBM Cloud Analytics for comprehensive user behavior analysis.

9. Continuous Improvement and Innovation Module:

- Objective: Stay up-to-date with emerging technologies and continuously enhance the platform's features and performance.
- Key Tasks:
- Gather user feedback and conduct usability testing to refine features and optimize the user experience.
- Explore and incorporate new technologies and trends in media streaming.

ALGORITHM AND TECHNOLOGY USED

1. Content Acquisition and Storage:

- Technology: IBM Cloud Object Storage
- Description: Utilize IBM Cloud Object Storage for efficient storage and retrieval of video content. This technology allows seamless handling of large video files.

2. Content Preprocessing:

- Technology: HTML, CSS, JavaScript (for frontend processing)
- Description: Use HTML, CSS, and JavaScript to ensure video content meets specified quality standards, addressing resolution, aspect ratio, and compression. This step ensures optimal viewing experience.

3. Platform Development and Integration:

- Technology: IBM Cloud Video Streaming APIs, HTML, CSS, PHP, JS
- Description: Leverage IBM Cloud Video Streaming APIs for hosting and managing video content. Additionally, utilize HTML, CSS, PHP, and JS for seamless integration and user interaction within the platform.

4. User Interface (UI) Design:

- Technology: HTML, CSS, JavaScript
- Description: Create an intuitive user interface using HTML, CSS, and JavaScript. This UI allows users to browse, search, and interact with the media streaming platform effortlessly.

5. Video Player Customization:

- Technology: HTML, CSS, JavaScript
- Description: Customize the video player using HTML, CSS, and JavaScript to provide a branded and user-friendly viewing experience. This customization ensures optimal playback quality.

6. User Authentication and Authorization:

- Technology: HTML, CSS, PHP, JS
- Description: Implement secure user authentication and authorization mechanisms using HTML, CSS, PHP, and JS. This step ensures controlled access to premium content and user-specific features.

7. Monetization Strategies (Optional):

- Technology: HTML, CSS, PHP, JS
- Description: If required, implement revenue-generating models using HTML, CSS, PHP, and JS. This may include subscription models, pay-per-view, or ad- based revenue streams.

8. Analytics and Insights:

- Technology: IBM Cloud Analytics, HTML, CSS, PHP, JS
- Description: Integrate analytics tools, like IBM Cloud Analytics, to gather user engagement data. Use HTML, CSS, PHP, and JS for comprehensive user behavior analysis, tracking views, watch time, and popular content.

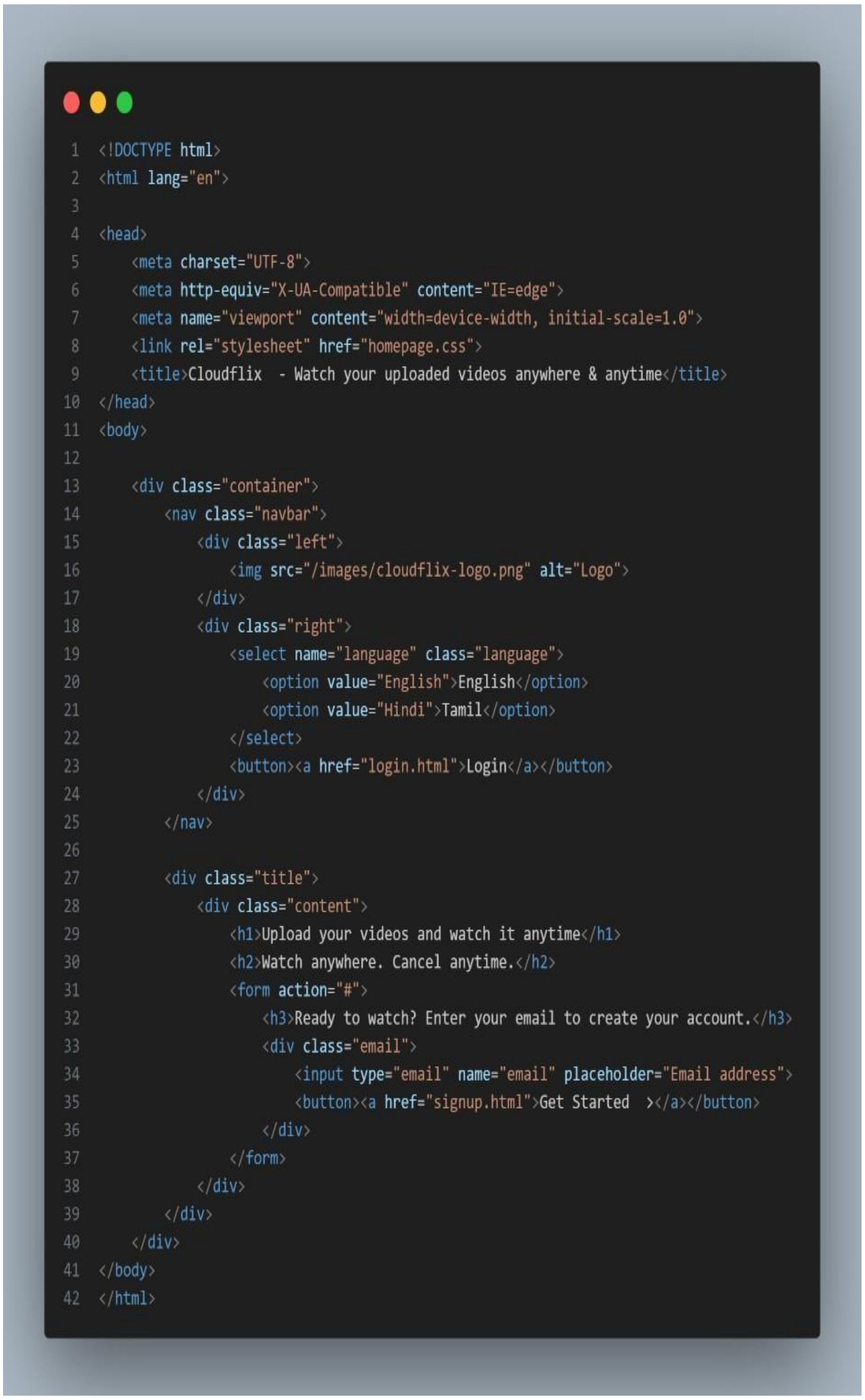
9. Continuous Improvement and Innovation:

- Technology: HTML, CSS, PHP, JS
- Description: Stay updated with emerging technologies and trends in media streaming. Continuously enhance the platform's features and performance using HTML, CSS, PHP, and JS. Gather user feedback and conduct usability testing for refinement.

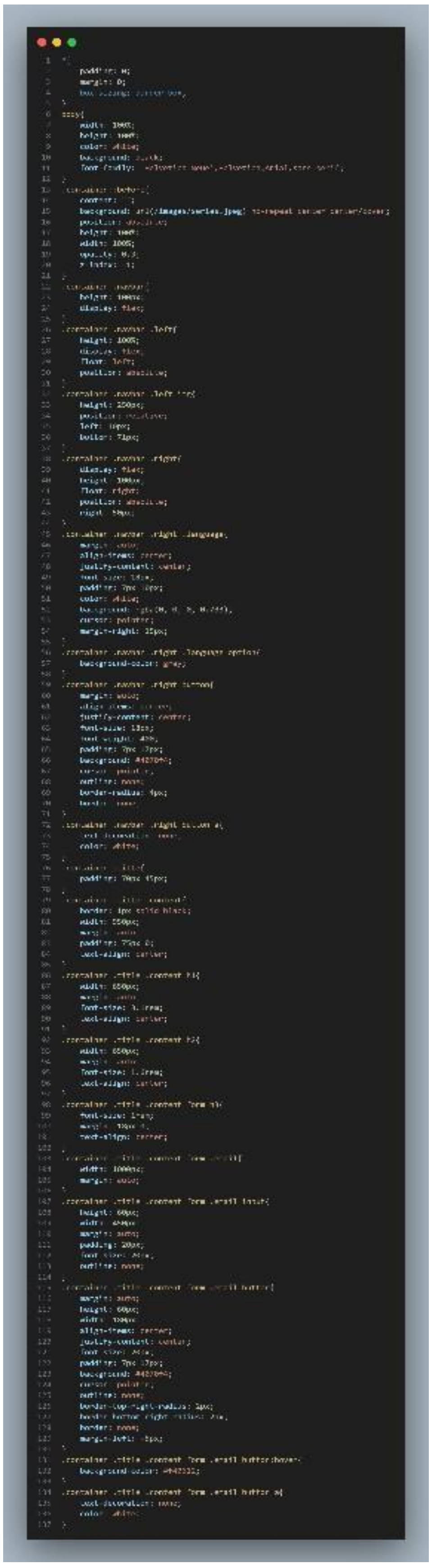
Utilizing a combination of HTML, CSS, PHP, JS, and IBM Cloud Video Streaming, this project ensures an interactive and user-friendly media streaming platform with features for content customization, monetization (optional), and comprehensive analytics for user engagement insights. Continuous improvement and innovation are emphasized to adapt to evolving trends in media streaming.

PROJECT DEVELOPMENT STEPS AND SCREENSHOT

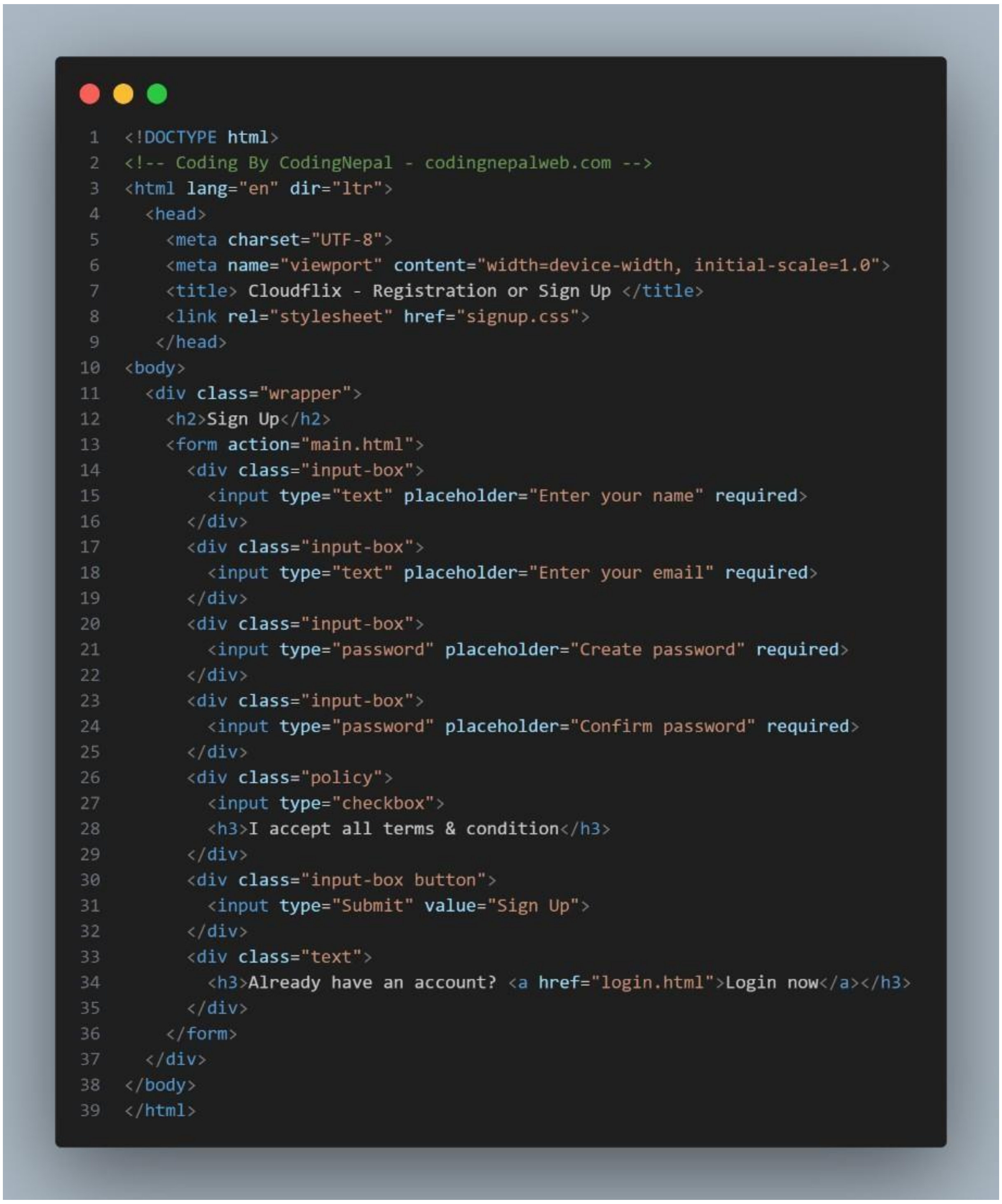
Front-End Development



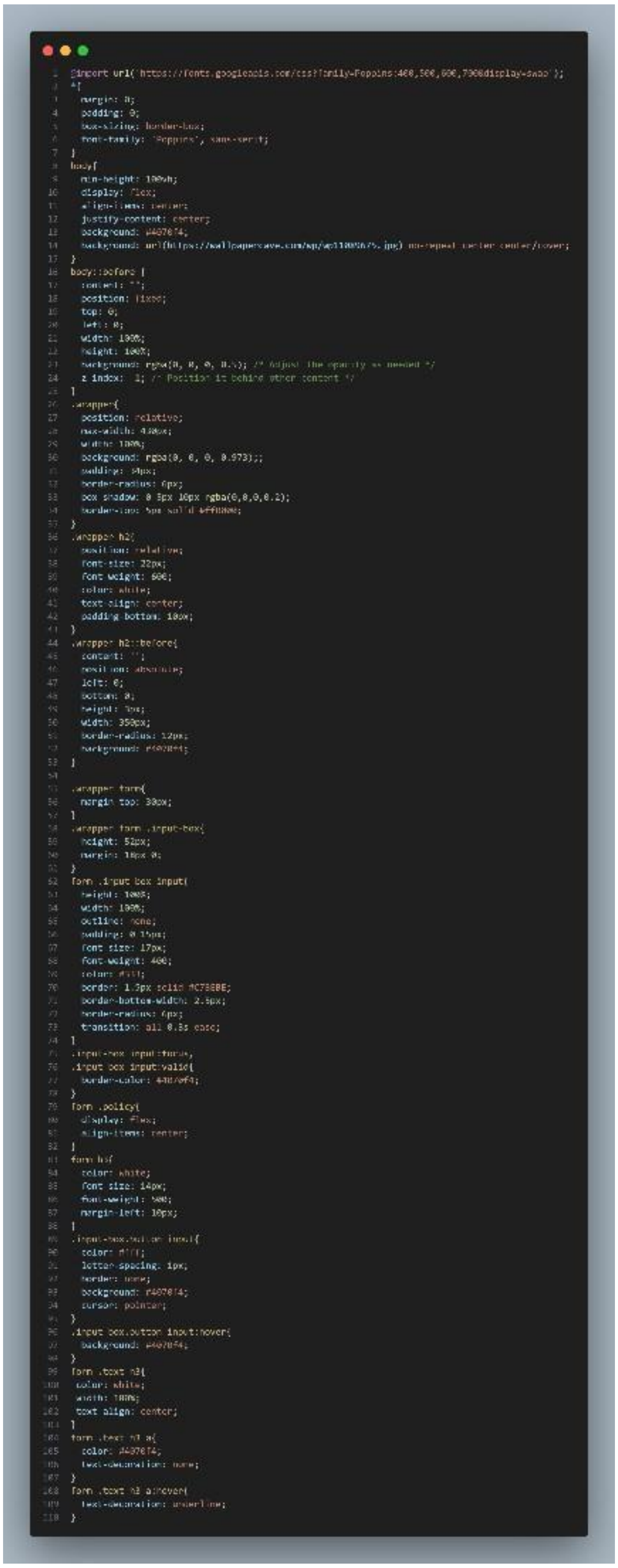
```html
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="homepage.css">
    <title>Cloudflix  - Watch your uploaded videos anywhere & anytime</title>
</head>
<body>

    <div class="container">
        <nav class="navbar">
            <div class="left">
                <img src="/images/cloudflix-logo.png" alt="Logo">
            </div>
            <div class="right">
                <select name="language" class="language">
                    <option value="English">English</option>
                    <option value="Hindi">Tamil</option>
                </select>
                <button><a href="login.html">Login</a></button>
            </div>
        </nav>

        <div class="title">
            <div class="content">
                <h1>Upload your videos and watch it anytime</h1>
                <h2>Watch anywhere. Cancel anytime.</h2>
                <form action="#">
                    <h3>Ready to watch? Enter your email to create your account.</h3>
                    <div class="email">
                        <input type="email" name="email" placeholder="Email address">
                        <button><a href="signup.html">Get Started  ></a></button>
                    </div>
                </form>
            </div>
        </div>
    </div>
</body>
</html>
```



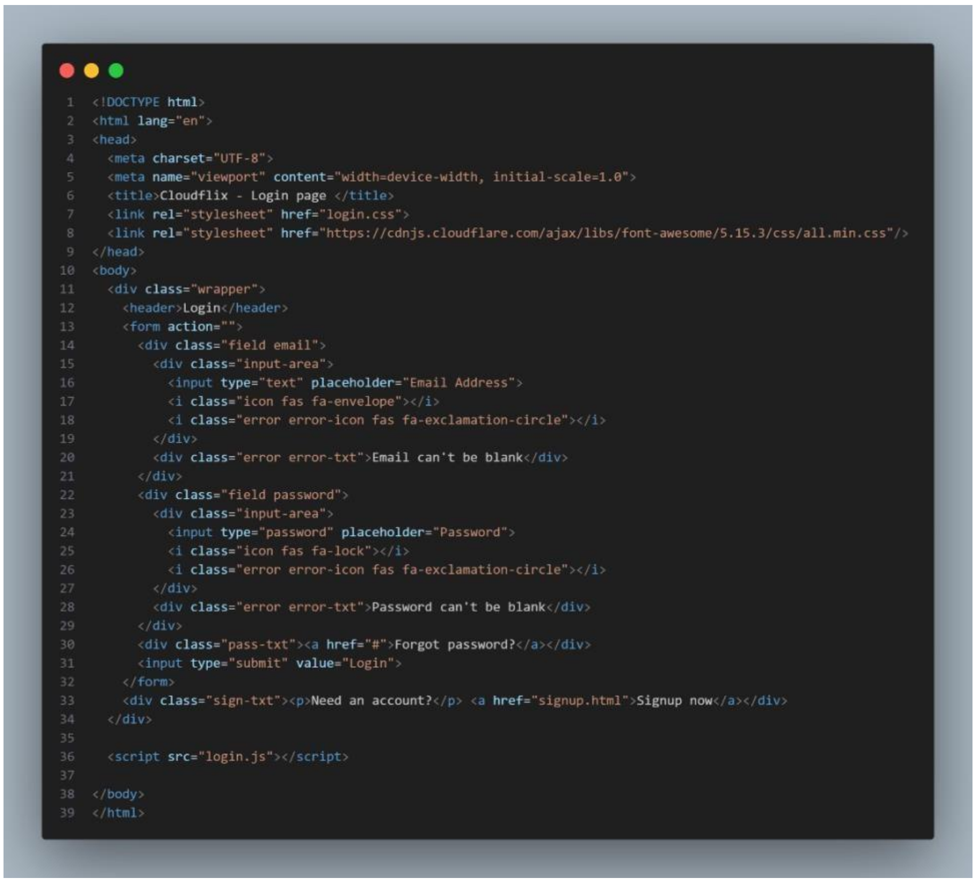
```css
*{
    padding: 0;
    margin: 0;
    box-sizing: border-box;
}
body{
    width: 100%;
    height: 100vh;
    color: white;
    background: black;
    font-family: 'Helvetica Neue',Helvetica,Arial,sans-serif;
}
.container::before{
    content: "";
    background: url(/images/series.jpeg) no-repeat center center/cover;
    position: absolute;
    height: 100%;
    width: 100%;
    opacity: 0.5;
    z-index: -1;
}
.container .navbar{
    height: 100px;
    display: flex;
}
.container .navbar .left{
    height: 100%;
    display: flex;
    float: left;
    position: absolute;
}
.container .navbar .left img{
    height: 150px;
    position: relative;
    left: 10px;
    bottom: 71px;
}
.container .navbar .right{
    display: flex;
    height: 100px;
    float: right;
    position: absolute;
    right: 50px;
}
.container .navbar .right .language{
    margin: auto;
    align-items: center;
    justify-content: center;
    font-size: 16px;
    padding: 7px 10px;
    color: white;
    background: rgba(0, 0, 0, 0.444);
    cursor: pointer;
    margin-right: 15px;
}
.container .navbar .right .language option{
    background-color: grey;
}
.container .navbar .right button{
    margin: auto;
    align-items: center;
    justify-content: center;
    font-size: 13px;
    font-weight: 400;
    padding: 7px 17px;
    background: #427044;
    cursor: pointer;
    outline: none;
    border-radius: 4px;
    border: none;
}
.container .navbar .right button a{
    text-decoration: none;
    color: white;
}
.container .title{
    padding: 70px 45px;
}
.container .title .content{
    border: 1px solid black;
    width: 550px;
    margin: auto;
    padding: 75px 0;
    text-align: center;
}
.container .title .content h1{
    width: 650px;
    margin: auto;
    font-size: 3.1rem;
    text-align: center;
}
.container .title .content h2{
    width: 650px;
    margin: auto;
    font-size: 1.2rem;
    text-align: center;
}
.container .title .content form h3{
    font-size: 1rem;
    margin: 10px 0;
    text-align: center;
}
.container .title .content form .email{
    width: 1000px;
    margin: auto;
}
.container .title .content form .email input{
    height: 60px;
    width: 450px;
    margin: auto;
    padding: 20px;
    font-size: 16px;
    outline: none;
}
.container .title .content form .email button{
    margin: auto;
    height: 60px;
    width: 150px;
    align-items: center;
    justify-content: center;
    font-size: 24px;
    padding: 7px 17px;
    background: #427044;
    cursor: pointer;
    outline: none;
    border-top-right-radius: 2px;
    border-bottom-right-radius: 2px;
    border: none;
    margin-left: -5px;
}
.container .title .content form .email button:hover{
    background-color: #842121;
}
.container .title .content form .email button a{
    text-decoration: none;
    color: white;
}
```



```html
<!DOCTYPE html>
<!-- Coding By CodingNepal - codingnepalweb.com -->
<html lang="en" dir="ltr">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title> Cloudflix - Registration or Sign Up </title>
    <link rel="stylesheet" href="signup.css">
   </head>
<body>
  <div class="wrapper">
    <h2>Sign Up</h2>
    <form action="main.html">
      <div class="input-box">
        <input type="text" placeholder="Enter your name" required>
      </div>
      <div class="input-box">
        <input type="text" placeholder="Enter your email" required>
      </div>
      <div class="input-box">
        <input type="password" placeholder="Create password" required>
      </div>
      <div class="input-box">
        <input type="password" placeholder="Confirm password" required>
      </div>
      <div class="policy">
        <input type="checkbox">
        <h3>I accept all terms & condition</h3>
      </div>
      <div class="input-box button">
        <input type="Submit" value="Sign Up">
      </div>
      <div class="text">
        <h3>Already have an account? <a href="login.html">Login now</a></h3>
      </div>
    </form>
  </div>
</body>
</html>
```

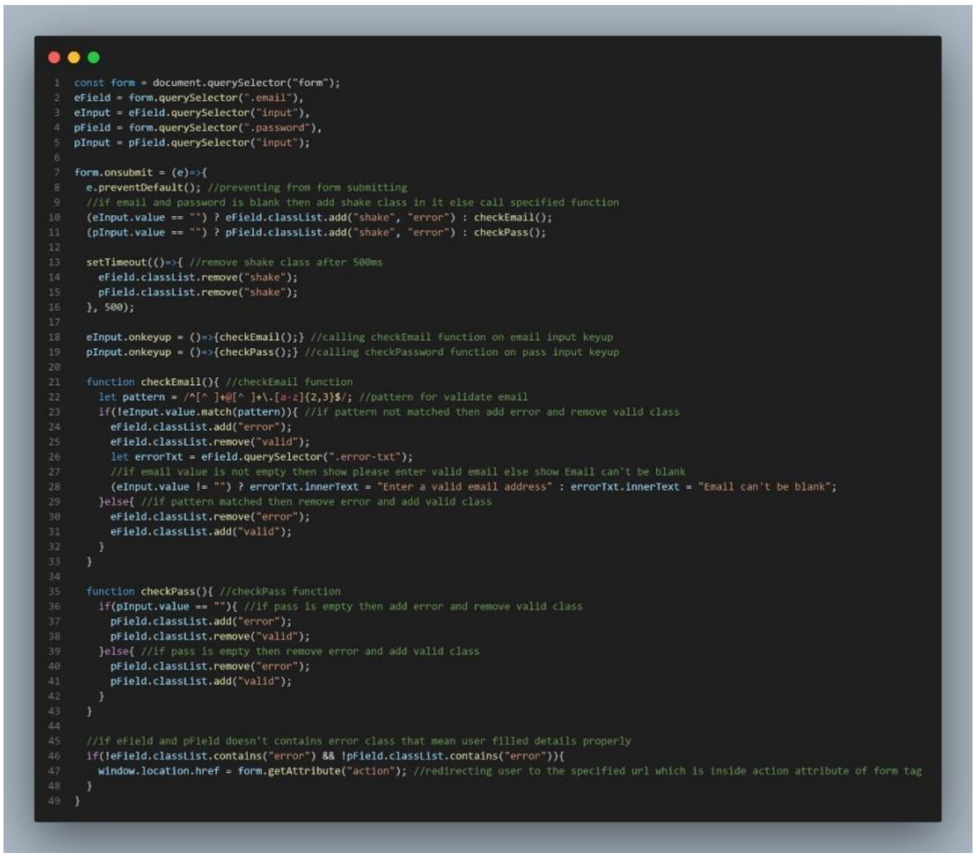


```css
@import url('https://fonts.googleapis.com/css?family=Poppins:400,500,600,700&display=swap');
*{
  margin: 0;
  padding: 0;
  box-sizing: border-box;
  font-family: 'Poppins', sans-serif;
}
body{
  min-height: 100vh;
  display: flex;
  align-items: center;
  justify-content: center;
  background: #4070f4;
  background: url(https://wallpapercave.com/wp/wp11005675.jpg) no-repeat center center/cover;
}
body::before {
  content: "";
  position: fixed;
  top: 0;
  left: 0;
  width: 100%;
  height: 100%;
  background: rgba(0, 0, 0, 0.5); /* Adjust the opacity as needed */
  z index: 1; /* Position it behind other content */
}
.wrapper{
  position: relative;
  max-width: 430px;
  width: 100%;
  background: rgba(0, 0, 0, 0.973);;
  padding: 34px;
  border-radius: 6px;
  box shadow: 0 5px 10px rgba(0,0,0,0.2);
  border-top: 5px solid #ff0000;
}
.wrapper h2{
  position: relative;
  font-size: 22px;
  font weight: 600;
  color: white;
  text-align: center;
  padding-bottom: 10px;
}
.wrapper h2::before{
  content: '';
  position: absolute;
  left: 0;
  bottom: 0;
  height: 3px;
  width: 350px;
  border-radius: 12px;
  background: #4070f4;
}

.wrapper form{
  margin top: 30px;
}
.wrapper form .input-box{
  height: 52px;
  margin: 18px 0;
}
form .input box input{
  height: 100%;
  width: 100%;
  outline: none;
  padding: 0 15px;
  font size: 17px;
  font-weight: 400;
  color: #333;
  border: 1.5px solid #C7BEBE;
  border-bottom-width: 2.5px;
  border-radius: 6px;
  transition: all 0.3s ease;
}
.input-box input:focus,
.input box input:valid{
  border-color: #4070f4;
}
form .policy{
  display: flex;
  align-items: center;
}
form h3{
  color: white;
  font size: 14px;
  font-weight: 500;
  margin-left: 10px;
}
.input-box.button input{
  color: #fff;
  letter-spacing: 1px;
  border: none;
  background: #4070f4;
  cursor: pointer;
}
.input box.button input:hover{
  background: #4070f4;
}
form .text h3{
 color: white;
 width: 100%;
 text align: center;
}
form .text h3 a{
  color: #4070f4;
  text-decoration: none;
}
form .text h3 a:hover{
  text-decoration: underline;
}
```



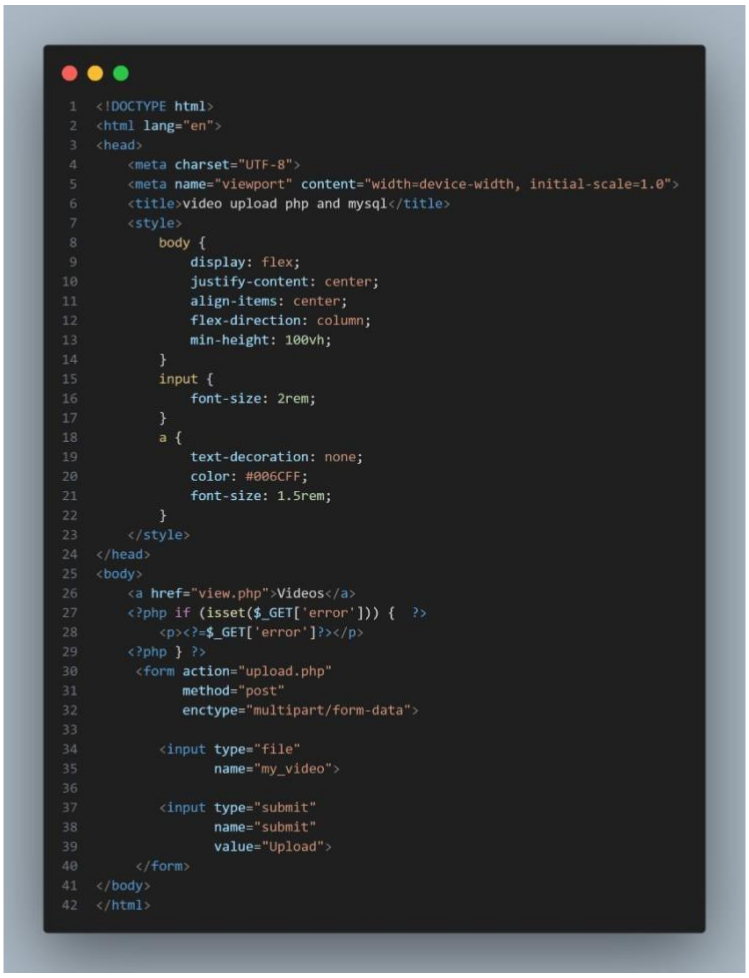
```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Cloudflix - Login page </title>
  <link rel="stylesheet" href="login.css">
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.15.3/css/all.min.css"/>
</head>
<body>
  <div class="wrapper">
    <header>Login</header>
    <form action="">
      <div class="field email">
        <div class="input-area">
          <input type="text" placeholder="Email Address">
          <i class="icon fas fa-envelope"></i>
          <i class="error error-icon fas fa-exclamation-circle"></i>
        </div>
        <div class="error error-txt">Email can't be blank</div>
      </div>
      <div class="field password">
        <div class="input-area">
          <input type="password" placeholder="Password">
          <i class="icon fas fa-lock"></i>
          <i class="error error-icon fas fa-exclamation-circle"></i>
        </div>
        <div class="error error-txt">Password can't be blank</div>
      </div>
      <div class="pass-txt"><a href="#">Forgot password?</a></div>
      <input type="submit" value="Login">
    </form>
    <div class="sign-txt"><p>Need an account?</p> <a href="signup.html">Signup now</a></div>
  </div>

  <script src="login.js"></script>

</body>
</html>
```

[illegible]

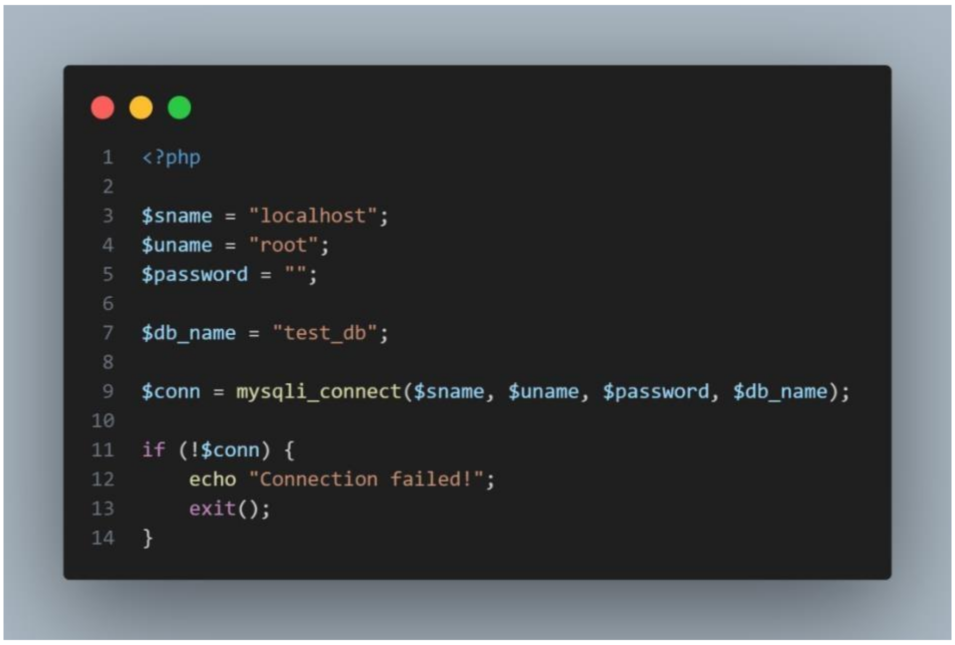


```javascript
const form = document.querySelector("form");
eField = form.querySelector(".email"),
eInput = eField.querySelector("input"),
pField = form.querySelector(".password"),
pInput = pField.querySelector("input");

form.onsubmit = (e)=>{
  e.preventDefault(); //preventing from form submitting
  //if email and password is blank then add shake class in it else call specified function
  (eInput.value == "") ? eField.classList.add("shake", "error") : checkEmail();
  (pInput.value == "") ? pField.classList.add("shake", "error") : checkPass();

  setTimeout(()=>{ //remove shake class after 500ms
    eField.classList.remove("shake");
    pField.classList.remove("shake");
  }, 500);

  eInput.onkeyup = ()=>{checkEmail();} //calling checkEmail function on email input keyup
  pInput.onkeyup = ()=>{checkPass();} //calling checkPassword function on pass input keyup

  function checkEmail(){ //checkEmail function
    let pattern = /^[^ ]+@[^ ]+\.[a-z]{2,3}$/; //pattern for validate email
    if(!eInput.value.match(pattern)){ //if pattern not matched then add error and remove valid class
      eField.classList.add("error");
      eField.classList.remove("valid");
      let errorTxt = eField.querySelector(".error-txt");
      //if email value is not empty then show please enter valid email else show Email can't be blank
      (eInput.value != "") ? errorTxt.innerText = "Enter a valid email address" : errorTxt.innerText = "Email can't be blank";
    }else{ //if pattern matched then remove error and add valid class
      eField.classList.remove("error");
      eField.classList.add("valid");
    }
  }

  function checkPass(){ //checkPass function
    if(pInput.value == ""){ //if pass is empty then add error and remove valid class
      pField.classList.add("error");
      pField.classList.remove("valid");
    }else{ //if pass is empty then remove error and add valid class
      pField.classList.remove("error");
      pField.classList.add("valid");
    }
  }

  //if eField and pField doesn't contains error class that mean user filled details properly
  if(!eField.classList.contains("error") && !pField.classList.contains("error")){
    window.location.href = form.getAttribute("action"); //redirecting user to the specified url which is inside action attribute of form tag
  }
}
```

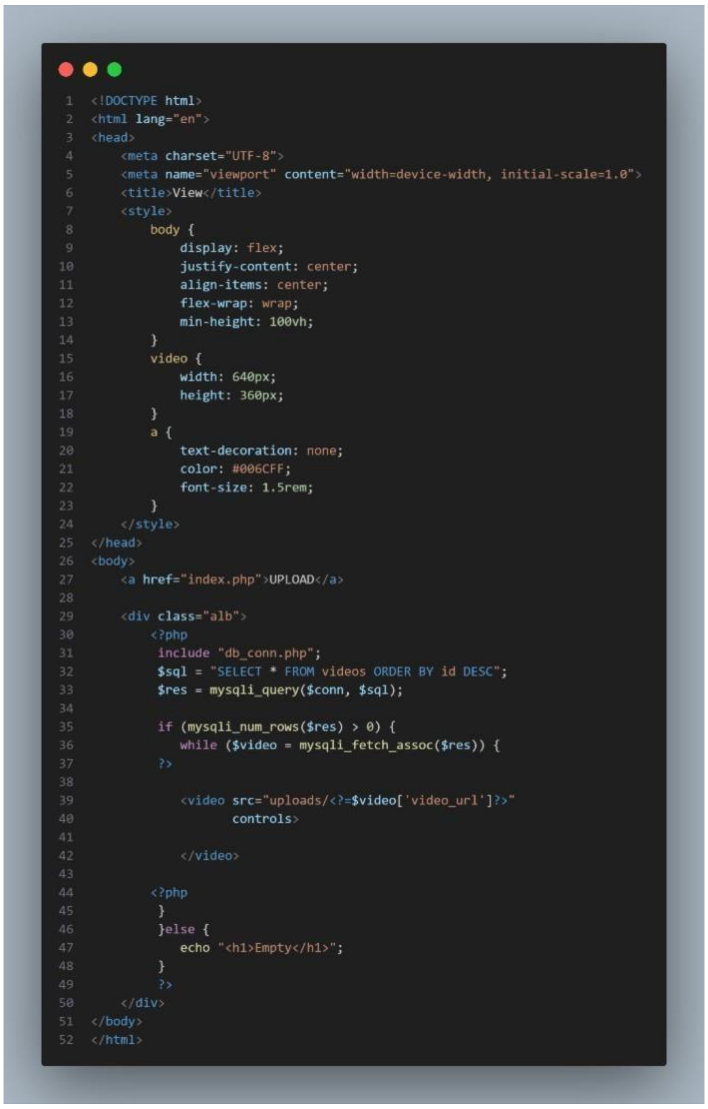
Back-end Development



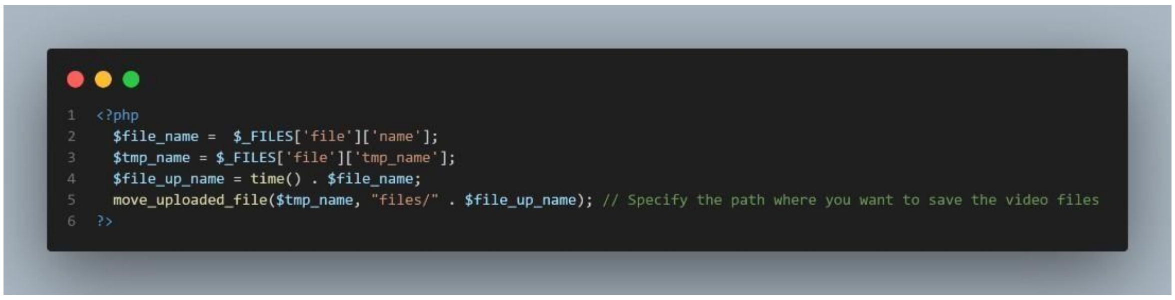
```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>video upload php and mysql</title>
    <style>
        body {
            display: flex;
            justify-content: center;
            align-items: center;
            flex-direction: column;
            min-height: 100vh;
        }
        input {
            font-size: 2rem;
        }
        a {
            text-decoration: none;
            color: #006CFF;
            font-size: 1.5rem;
        }
    </style>
</head>
<body>
    <a href="view.php">Videos</a>
    <?php if (isset($_GET['error'])) {  ?>
        <p><?=$_GET['error']?></p>
    <?php } ?>
     <form action="upload.php"
           method="post"
           enctype="multipart/form-data">

        <input type="file"
               name="my_video">

        <input type="submit"
               name="submit"
               value="Upload">
     </form>
</body>
</html>
```



```php
<?php

$sname = "localhost";
$uname = "root";
$password = "";

$db_name = "test_db";

$conn = mysqli_connect($sname, $uname, $password, $db_name);

if (!$conn) {
    echo "Connection failed!";
    exit();
}
```

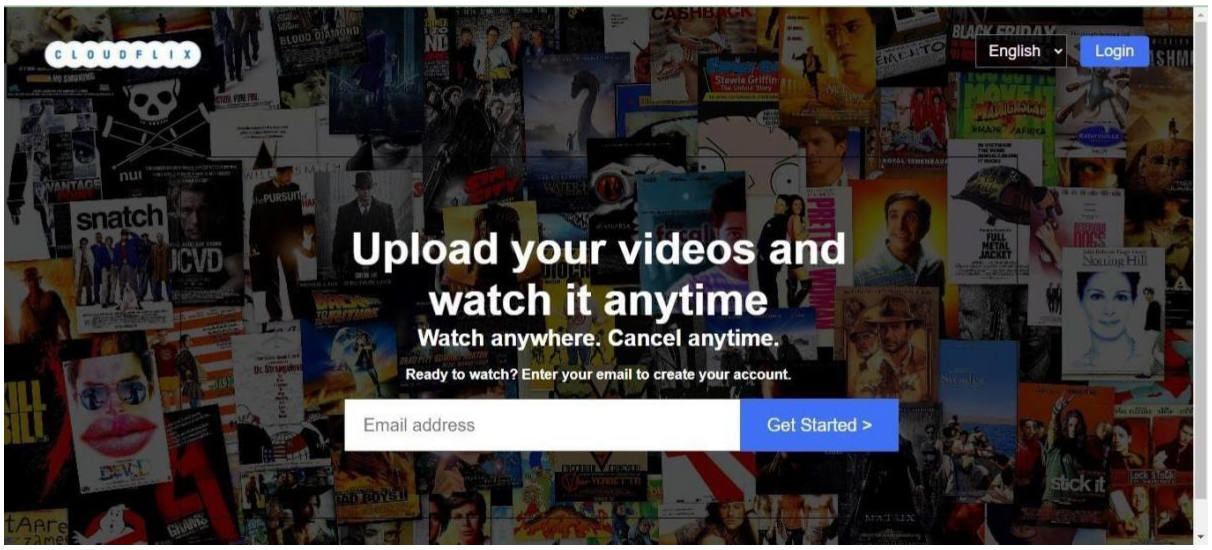


```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>View</title>
    <style>
        body {
            display: flex;
            justify-content: center;
            align-items: center;
            flex-wrap: wrap;
            min-height: 100vh;
        }
        video {
            width: 640px;
            height: 360px;
        }
        a {
            text-decoration: none;
            color: #006CFF;
            font-size: 1.5rem;
        }
    </style>
</head>
<body>
    <a href="index.php">UPLOAD</a>

    <div class="alb">
        <?php
         include "db_conn.php";
         $sql = "SELECT * FROM videos ORDER BY id DESC";
         $res = mysqli_query($conn, $sql);

         if (mysqli_num_rows($res) > 0) {
            while ($video = mysqli_fetch_assoc($res)) {
         ?>

            <video src="uploads/<?=$video['video_url']?>"
                   controls>

            </video>

        <?php
         }
         }else {
            echo "<h1>Empty</h1>";
         }
         ?>
    </div>
</body>
</html>
```

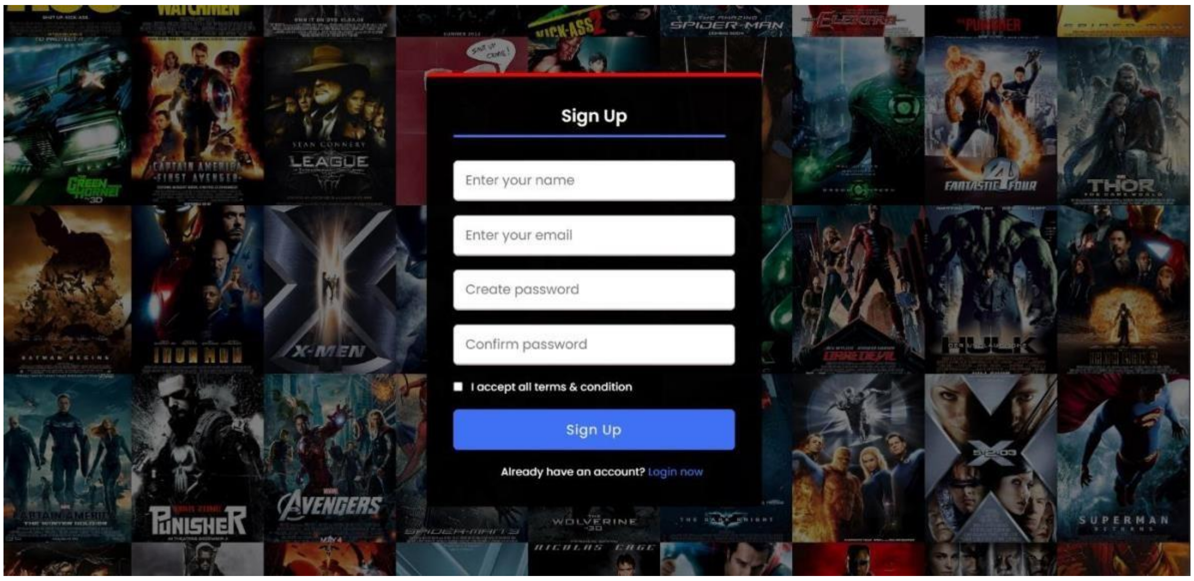


```
<?php
  $file_name =  $_FILES['file']['name'];
  $tmp_name = $_FILES['file']['tmp_name'];
  $file_up_name = time() . $file_name;
  move_uploaded_file($tmp_name, "files/" . $file_up_name); // Specify the path where you want to save the video files
?>
```

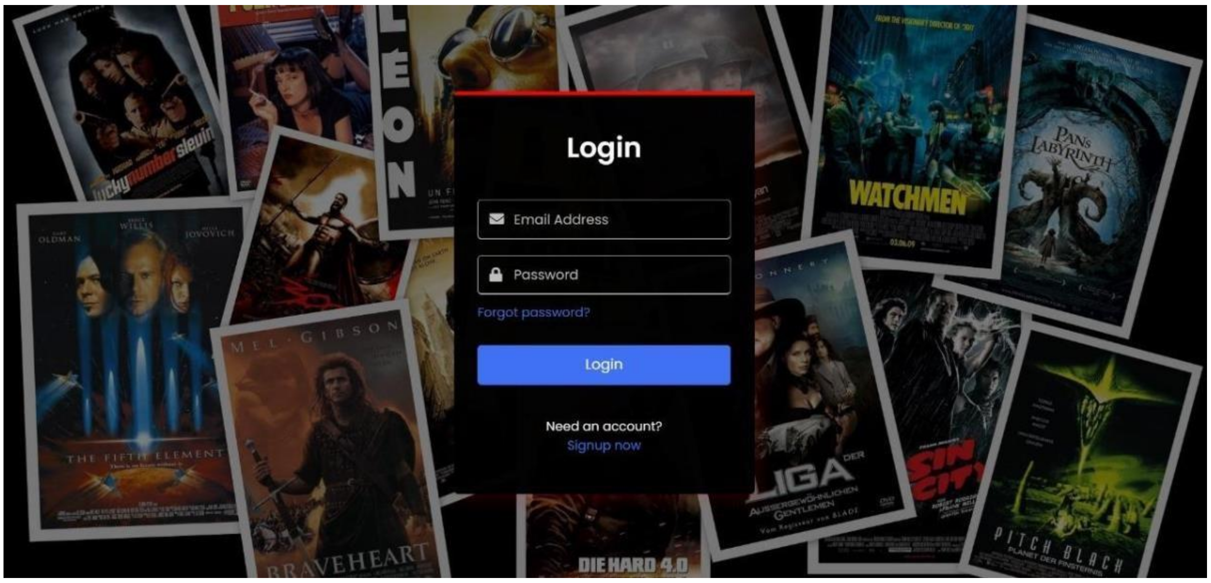
Step 1: Enter your e-mail id to create your account in cloudflix.


CLOUDFLIX
English
Login
Upload your videos and watch it anytime
Watch anywhere. Cancel anytime.
Ready to watch? Enter your email to create your account.
Email address
Get Started >

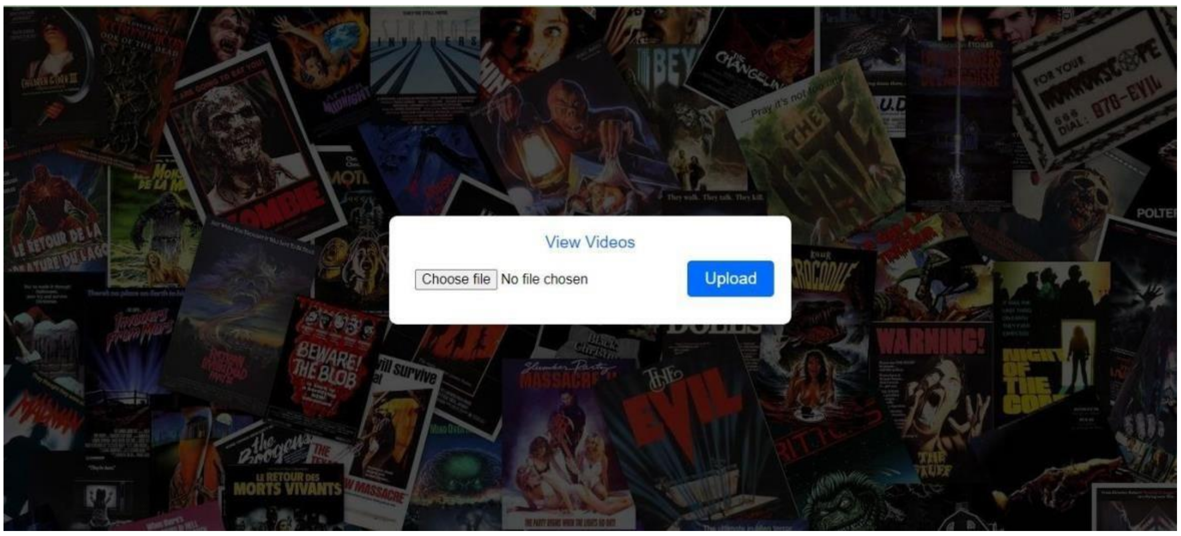
Step 2: Enter your name and create a strong password to complete the sign-up process.



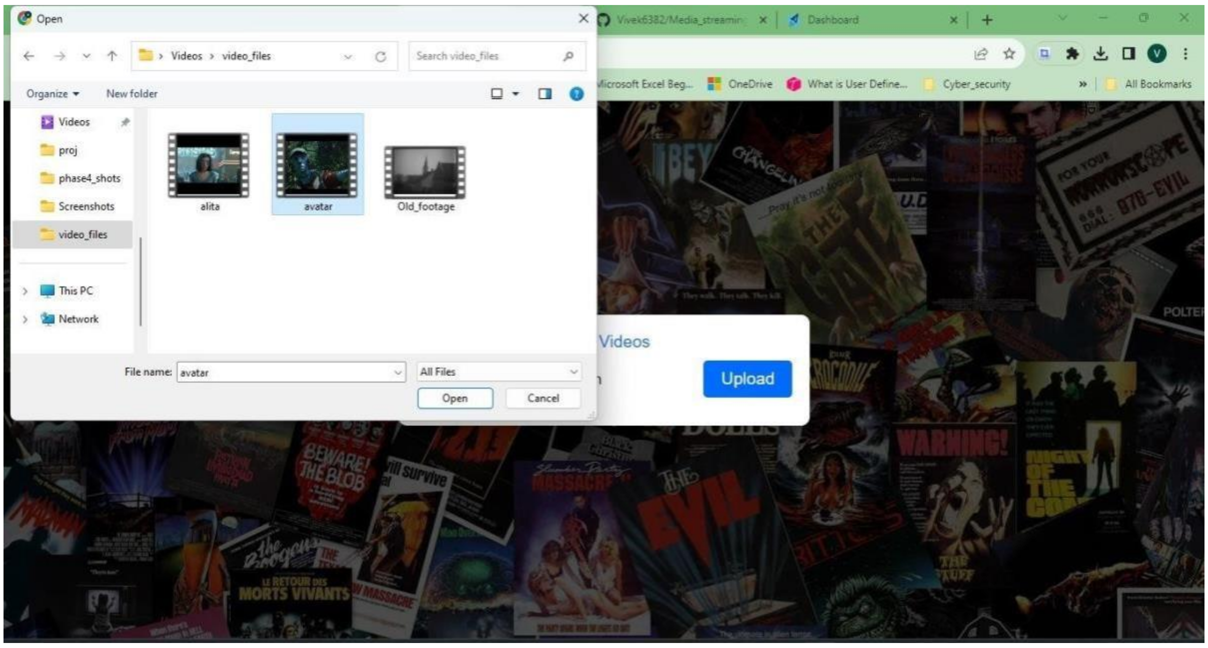
Step 3: Enter the registered mail id and password to log in.



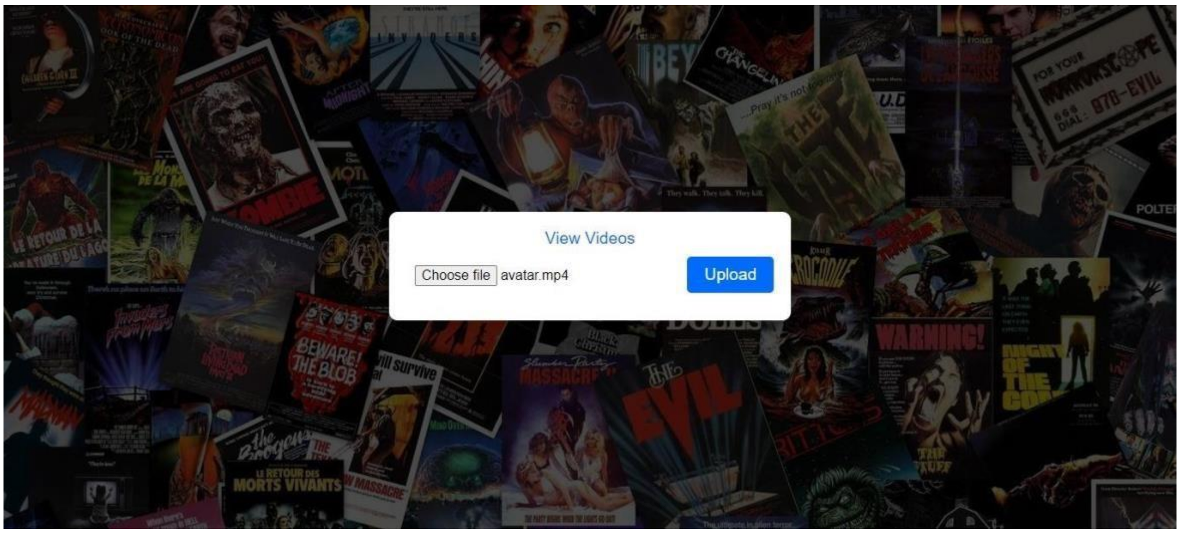
Step 4: Now click the choose file button to select the video to be uploaded.



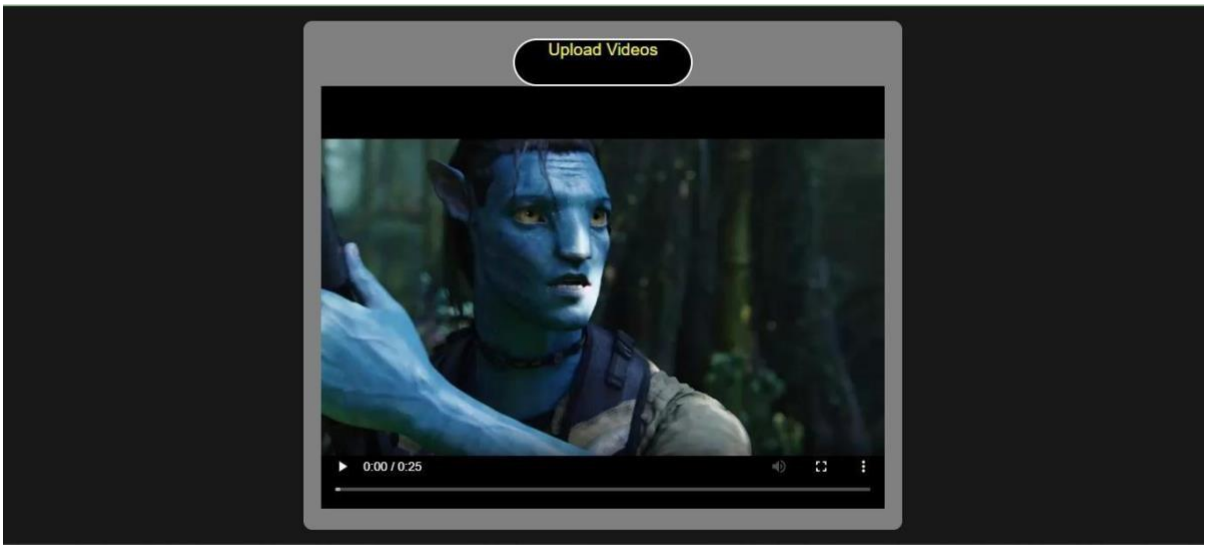
Step 5: Choose the video file which you want to ulpoad and click open.

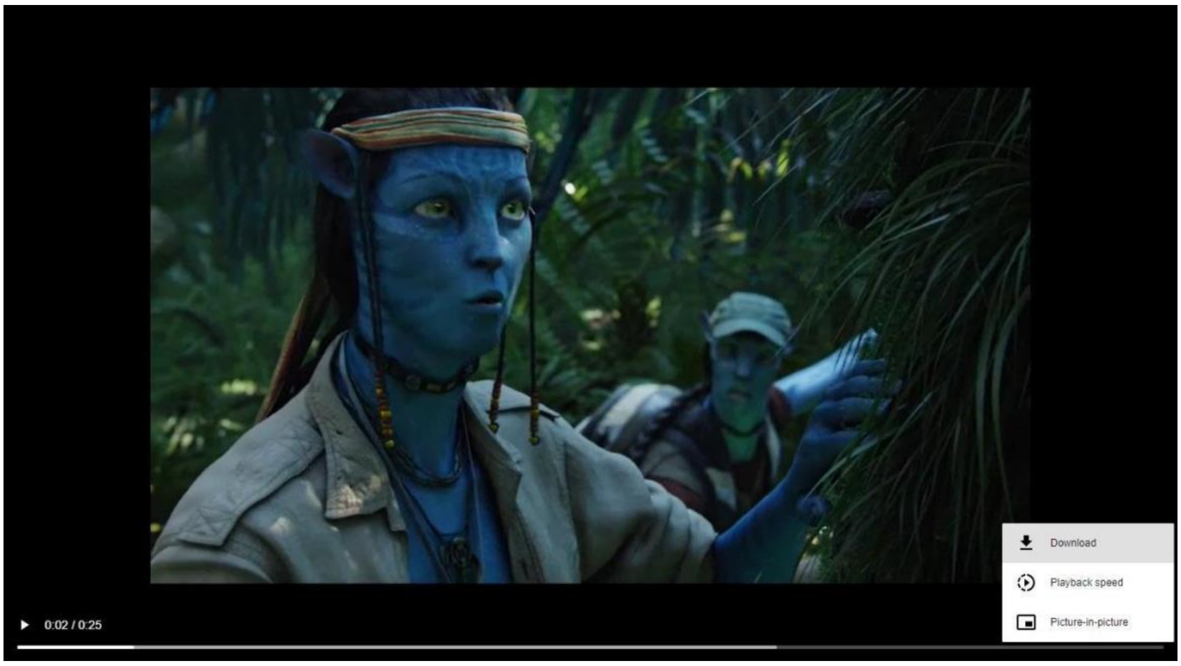


Step 6: Now the name of the chosen file will be displayed.

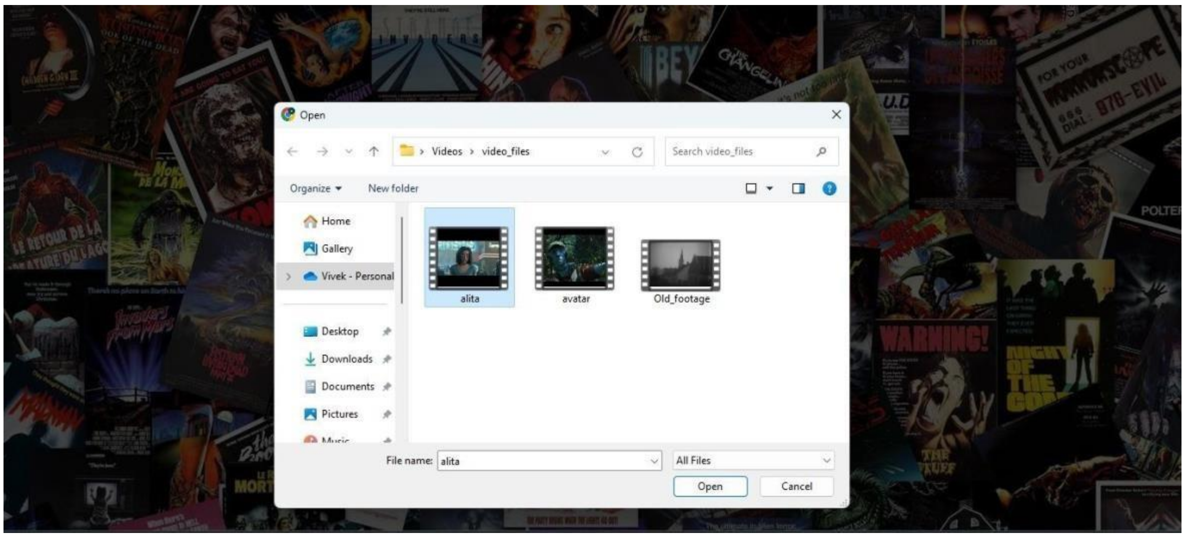


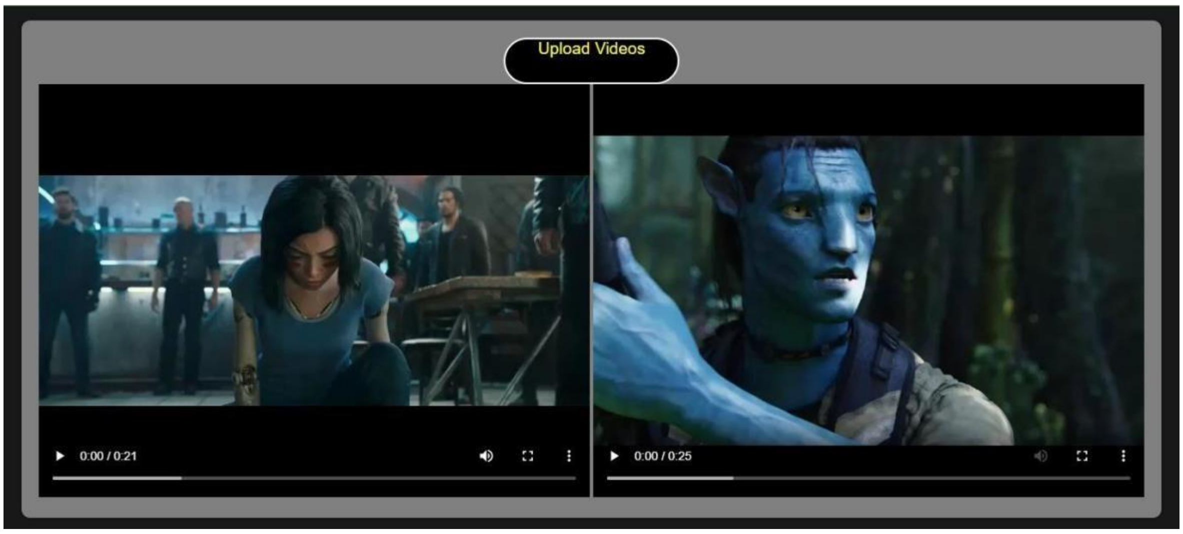
Step 7: Now click upload to upload the selected video file and now the video will be displayed .

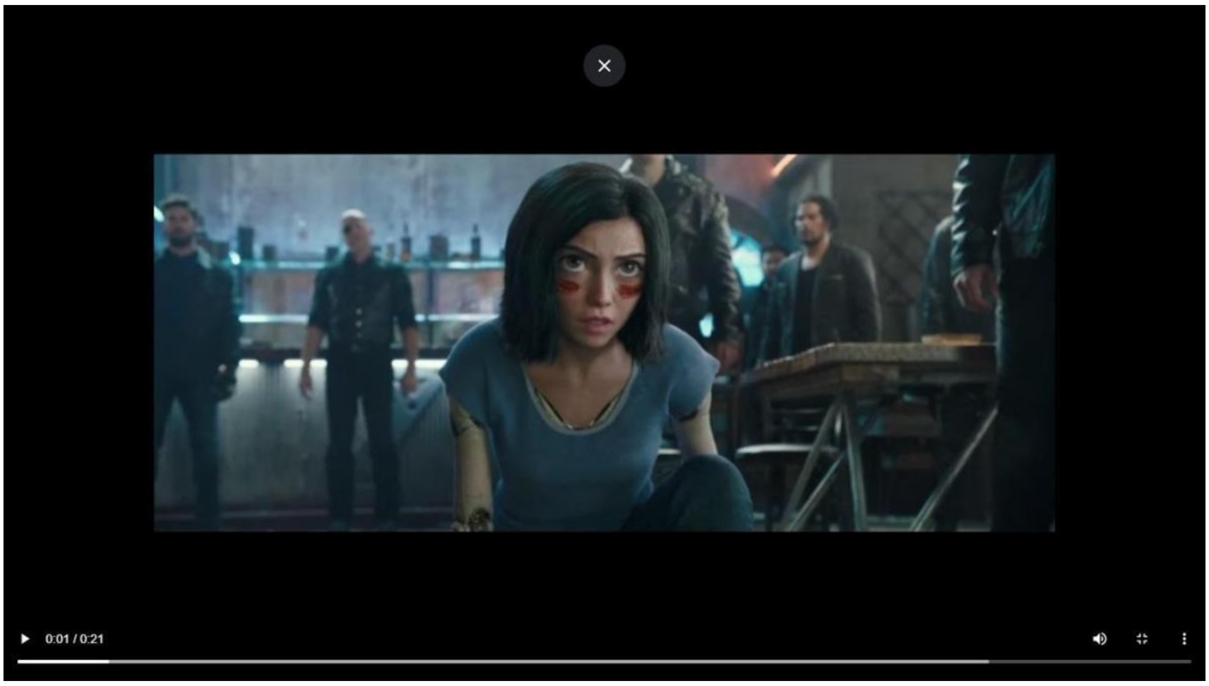




Step 8: Likewise you can upload another video and it will be displayed with the previously uploaded video.


Open
Videos > video_files
Search video_files
Organize
New folder
Home
Gallery
Vivek - Personal
Desktop
Downloads
Documents
Pictures
Music
alita
avatar
Old_footage
File name: alita
All Files
Open
Cancel


Upload Videos
0:00 / 0:21
0:00 / 0:25



CONCLUSION

In conclusion, the implementation of cloud media streaming exemplifies the boundless opportunities technology offers in the realm of content delivery and consumption. With the ability to seamlessly stream media from remote servers to various devices, it not only revolutionizes the way we access and enjoy content but also underscores the immense potential of cloud-based solutions. The scalability, convenience, and cost-efficiency of cloud media streaming make it a game-changer for both consumers and providers, shaping the future of entertainment and information dissemination. As we continue to witness rapid advancements in this field, it is clear that cloud media streaming will play a pivotal role in redefining the digital landscape and how we interact with media.