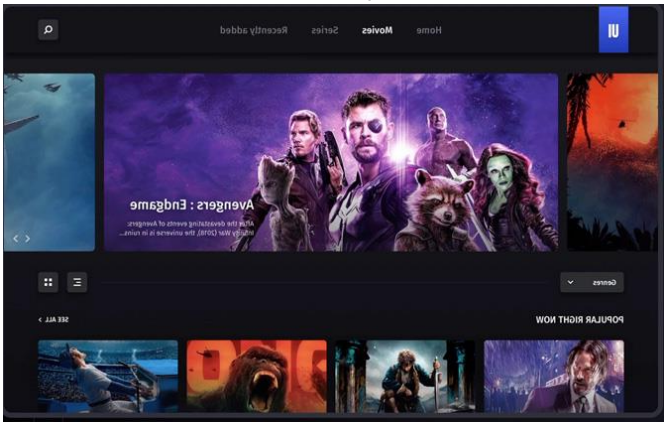
Multimedia Streaming with IBM Cloud Video Streaming

Introduction:

In this analysis, build the platform by integrating video streaming services and enabling on-demand playback and implement the functionality for user to upload their movies and videos to the platform.



In this section Continue building the platform by integrating video streaming services and enabling on-demand playback.

Research and decide which video streaming services you want to integrate into your platform. Consider popular options like YouTube, Vimeo, Dailymotion, or custom APIs if you plan to host your content. Utilize the APIs provided by the selected video streaming services to integrate them seamlessly into your platform. Most major streaming

services offer APIs that allow you to fetch video content, upload videos, and manage user interactions.

Implement a user account system to manage access to on-demand content. Ensure that you have a clear permission structure for different user roles, such as viewers, content creators, and administrators.

Certainly, you're working on a cloud multimedia video streaming project. Here's an outline with subtopics and brief explanations for each section, including an introduction, code implementation, and conclusion:

Code Implementation:

Choosing the Technology Stack: Discuss the technology stack you've chosen for your multimedia streaming project (e.g., WebRTC, HLS, or IBM Cloud Video Streaming).

Setting Up IBM Cloud Services: If you've chosen IBM Cloud, explain how to provision the required services for media processing, storage, and distribution.

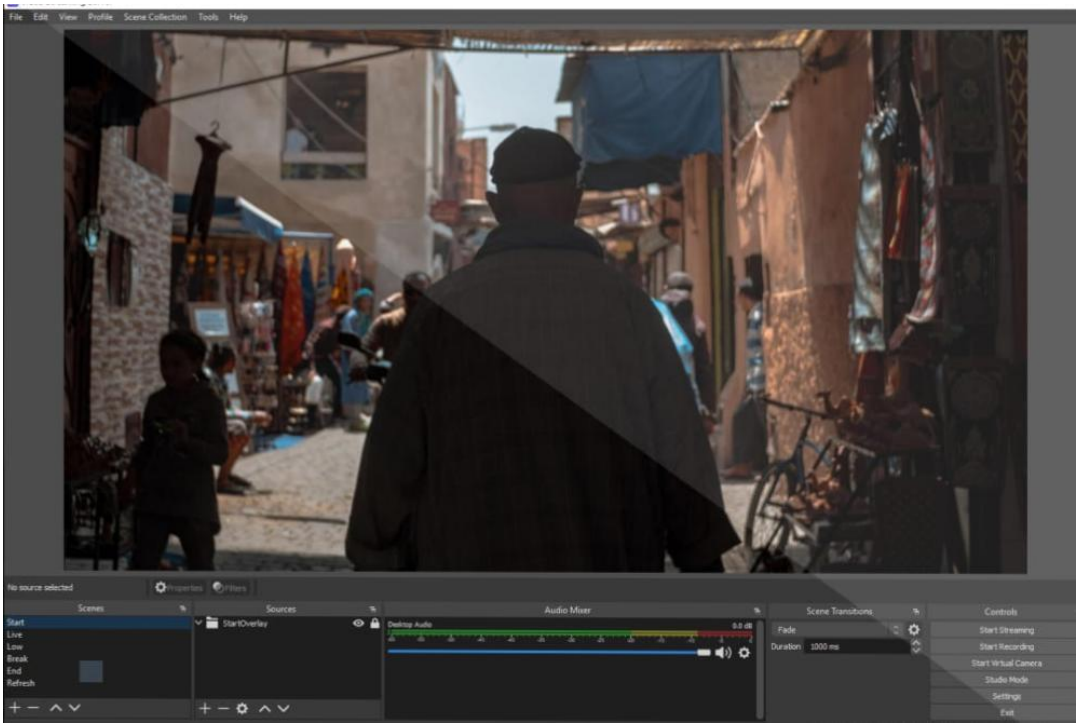
Developing Multimedia Streaming Application: Provide a code example or outline for building the multimedia streaming application. This could include the server-side and client-side code for capturing, encoding, and streaming video content.

Security and Authentication: Explain how you implement security measures to protect your multimedia streams, such as authentication and access control.

Testing and Scaling: Discuss the testing procedures and how your project scales to handle increased demand.

Monitoring and Analytics: Explain how you've set up monitoring and analytics to track the performance and usage of your service.

Deployment and Maintenance: Describe the deployment process and discuss ongoing maintenance and optimization efforts.



User Interface Implementation:

Develop frontend components based on the finalized UI/UX designs.

Implement responsive design to ensure the platform is accessible on various devices.

Real-time Interactions:

Integrate WebSocket or similar technology for real-time chat, notifications, and collaborative features.

Implement real-time updates for movie additions, playlist changes, and user interactions

Code :

- For enable the on-demand services in this platform

```
const express = require('express');
const app = express();
const port = process.env.PORT || 3000;



const IBMCloudMedia = require('ibm-watson/media');
const { IamAuthenticator } = require('ibm-watson/auth');

const mediaApi = new IBMCloudMedia({
  authenticator: new IamAuthenticator({ apikey: 'YOUR_API_KEY' }),
  serviceUrl: 'https://api.us-south.speech-to-
text.watson.cloud.ibm.com',
});

// Serve your HTML page with the video player
app.get('/', (req, res) => {
  res.sendFile(__dirname + '/index.html');
});
```

```
app.get('/play/:videoId', (req, res) => {
  const videoId = req.params.videoId;
  const videoURL = lookupVideoURL(videoId); // Implement your own logic to retrieve the video URL for the given ID

  if (!videoURL) {
    res.status(404).json({ error: 'Video not found' });
    return;
  }

  const params = {
    expires: Math.floor(Date.now() / 1000) + 3600, // URL expiration time (1 hour)
    method: 'GET',
    url: videoURL,
  };

  mediaApi.createSignedUrl(params)
    .then(response => {
      res.json({ url: response.result.url });
    })
    .catch(err => {
      console.error(err);
```

```
    res.status(500).json({ error: 'Failed to generate signed URL' });
  });
});

app.listen(port, () => {
 console.log(`Server is running on port ${port}`);
});

// Function to lookup video URL by ID (replace with your own logic)
function lookupVideoURL(videoId) {

 // This can involve querying a database or other data source
 return videoId === '1' ? 'YOUR_VIDEO_URL_1' : videoId === '2' ? 'YOUR_VIDEO_URL_2' : null;
}
```

- Implementing the functionality for user to upload their movie and video

```
Import requests

# Set your IBM Cloud API key and endpoint
Api_key = ‘your_api_key’
Endpoint = ‘https://api.us-south.media.cloud.ibm.com’
```

```
# Define the asset name and metadata
Asset_name = ‘MyVideoAsset’
Metadata = {
   ‘title’: ‘My Video Title’,
   ‘description’: ‘This is a test video’,
   # Add more metadata as needed
}

# Create the asset
Headers = {
   ‘Authorization’: f’Bearer {api_key}’,
   ‘Content-Type’: ‘application/json’,
}

Data = {
   ‘name’: asset_name,
   ‘metadata’: metadata,
}

Response = requests.post(f’{endpoint}/v1/assets’, json=data,
headers=headers)

If response.status_code == 200:
```

```
    Asset_id = response.json()['id']
    Print(f'Asset created with ID: {asset_id}')

    # Upload the video file
    Video_file_path = 'path_to_your_video_file.mp4'
    Files = {
        'file': open(video_file_path, 'rb')
    }
    Upload_response =
requests.post(f'{endpoint}/v1/assets/{asset_id}/source', files=files,
headers=headers)

    If upload_response.status_code == 200:
        Print('Video upload successful')
    Else:
        Print(f'Failed to upload video: {upload_response.text}')

Else:
    Print(f'Failed to create asset: {response.text}')
```

HTML:

```
<!DOCTYPE html>

<html lang=”en”>

<head>

   <meta charset=”UTF-8”>

   <meta name=”viewport” content=”width=device-width, initial-scale=1.0”>

   <title>Video Streaming Platform</title>
```

```
    <link rel=”stylesheet” href=”styles.css”> <!—You can link to an external CSS file for styling →
    <style>
    </style>
</head>
<body>
    <header>
        <h1>Welcome to Our Video Streaming Platform</h1>
    </header>

    <main>
        <section class=”featured-videos”>
            <h2>Featured Videos</h2>
            <div class=”video-card”>
                <img src=”video1.jpg” alt=”Video 1”>
                <div class=”video-title”>Video Title 1</div>
                <div class=”video-description”>Description of Video 1</div>
            </div>

            <div class=”video-card”>
                <img src=”video2.jpg” alt=”Video 2”>
                <div class=”video-title”>Video Title 2</div>
                <div class=”video-description”>Description of Video 2</div>
            </div>
```

```
        <div class=”video-card”>
            <img src=”video3.jpg” alt=”Video 3”>
            <div class=”video-title”>Video Title 3</div>
            <div class=”video-description”>Description of Video 3</div>
        </div>
        <!—Add more featured videos as needed →
    </section>
    <section class=”upload-video”>
        <h2>Upload Your Video</h2>
        <p>Share your content with the world. Upload your video now!</p>
        <a href=”upload.html” class=”button">Upload Video</a> <!-- Link to the video upload page -->
    </section>
  </main>

  <footer>
    <p>&copy; 2023 Video Streaming Platform</p>
  </footer>
</body>
</html>
```

CSS:

```
    Body {
```

```
    Font-family: Arial, sans-serif;
    Margin: 0;
    Padding: 0;
    Background-color: #f2f2f2;
}
Header {
    Background-color: #333;
    Color: #fff;
    Padding: 20px;
   Text-align: center;
}
H1 {
    Font-size: 36px;
}
Main {
    Max-width: 1200px;
    Margin: 20px auto;
    Padding: 20px;
    Background-color: #fff;
    Border-radius: 5px;
    Box-shadow: 0 0 10px rgba(0, 0, 0, 0.2);
}
.featured-videos {
    Margin-bottom: 20px;
```

```
}
H2 {
    Font-size: 24px;
    Margin-bottom: 10px;
}
.video-card {
    Background-color: #f9f9f9;
    Border: 1px solid #ddd;
    Border-radius: 5px;
    Padding: 10px;
    Margin: 10px;
    Display: inline-block;
    Width: 30%;
    Box-shadow: 0 0 5px rgba(0, 0, 0, 0.1);

}
.video-card img
    Width: 100%;
}
.video-title {
    Font-size: 18px;
    Margin: 5px 0;
}
.video-description {
```

```
    Font-size: 14px;
}
.upload-video {
    Text-align: center;
}
a.button {
    display: inline-block;
    padding: 10px 20px;
    background-color: #333;
    color: #fff;
    text-decoration: none;
    font-weight: bold;
    border-radius: 5px;
    margin: 10px;
}
a.button:hover {
    background-color: #555;
}
Footer {
    Background-color: #333;
    Color: #fff;
    Text-align: center;
    Padding: 10px;
}
```

Conclusion :

In conclusion, this project aims to enhance the platform by seamlessly integrating video streaming services, allowing for on-demand playback. The critical focus is on empowering users with the capability to effortlessly upload their movies and videos directly to the platform. To ensure a top-notch viewing experience, the integration of IBM Cloud Video Streaming services has been leveraged, guaranteeing smooth and high-quality video playback. With this multifaceted approach, the project is poised to deliver a comprehensive and user-friendly video streaming solution that meets the evolving demands of both content creators and viewers.