CLOUD APPLICATION DEVELOPMENT(CAD)

**Project:** Media Streaming with IBM Cloud video Streaming

**Phase 3:** Development Part1

**Step 1: Set Up IBM Cloud Account**

**Sign Up:** If you haven't already, sign up for IBM Cloud at IBM Cloud website.

**Step 2: Create a Video Streaming Service**

Create Video Streaming Service: In the IBM Cloud dashboard, create a new instance of the IBM Video Streaming service.

**Step 3: Prepare Your Videos**

Upload Videos: Upload your movies or videos to the IBM Cloud Video Streaming service. Ensure they are properly formatted and encoded for streaming.

**Step 4: Build the Virtual Cinema Platform**

Choose a Tech Stack: Decide on the technology stack for your platform. For example, you can use Node.js for the backend and React for the frontend.

**Set Up Backend (Node.js Example):**

Install required packages:

npm install express ibm-cloud-sdk

**Create a Node.js server (app.js) to handle requests:**

const express = require('express');

const app = express();

const bodyParser = require('body-parser');

app.use(bodyParser.json());

// Movie upload endpoint

app.post('/api/upload', (req, res) => {

// Implement logic to handle movie uploads using IBM Cloud Video Streaming API

// Save movie details to the database

res.json({ success: true, message: 'Movie uploaded successfully.' });

});

// Premiere scheduling endpoint

app.post('/api/schedule', (req, res) => {

res.json({ success: true, message: 'Premiere scheduled successfully.' });

});

app.listen(3000, () => { console.log('Server started on port 3000');});

**Set Up Frontend (React Example):**

Install required packages:

npx create-react-app virtual-cinema

cd virtual-cinema

**Modify the src/App.js file to display video links:**

import React from 'react';

import './App.css';

function App() {

const [movieDetails, setMovieDetails] = useState({ title: '', description: '', trailerURL: '', file: null });

const handleMovieUpload = () => {

// Implement logic to send movie details to the backend for upload

fetch('/api/upload', {

method: 'POST',

body: JSON.stringify(movieDetails),

headers: { 'Content-Type': 'application/json' }

})

.then(response => response.json())

.then(data => {

console.log(data);

// Handle response from the server

});

};

return (

<div className="App">

{/\* Movie upload form \*/}

<input type="text" placeholder="Movie Title" onChange={(e) => setMovieDetails({ ...movieDetails, title: e.target.value })} />

{/\* More input fields for movie description, trailer URL, and file upload \*/}

<button onClick={handleMovieUpload}>Upload Movie</button>

</div>

);

}

**Step 5: Integrate IBM Cloud Video Streaming API**

Use IBM Cloud SDK: Utilize the IBM Cloud SDK for Node.js to interact with the IBM Cloud Video Streaming service. Refer to IBM Cloud documentation for specific API endpoints and methods.

**Example (inside your Node.js backend):**

const IBMCloud = require('ibm-cloud-sdk');

const ibmCloud = new IBMCloud({

apiKey: 'YOUR\_API\_KEY',

serviceInstanceId: 'YOUR\_SERVICE\_INSTANCE\_ID',

});

const videoService = ibmCloud.createVideoStreamingService();

**Step 6: Implement User Authentication and Authorization**

User Management: Implement user authentication and authorization mechanisms to secure your platform. You can use services like IBM Cloud App ID for user authentication.

**Example Code (App ID Setup):**

const AppID = require('ibmcloud-appid');

const appID = new AppID({

clientId: 'YOUR\_CLIENT\_ID',

tenantId: 'YOUR\_TENANT\_ID',

oauthServerUrl: AppID.Region.US\_SOUTH // Adjust region as needed

});

**Step 7: Implement Payment Integration (Optional)**

Payment Gateway: If you plan to charge users for accessing premium content, integrate a payment gateway such as Stripe or PayPal.

**Step 8: Testing and Deployment**

Testing: Test your virtual cinema platform thoroughly, including video streaming, user authentication, and payment processes.

Deployment: Deploy your backend (Node.js) to a server or a cloud service like IBM Cloud Foundry, and deploy your React frontend to a hosting service like Netlify or Vercel.

**Step 9: Continuous Improvement**

Gather Feedback: Collect feedback from users and make necessary improvements to enhance the user experience.

Update Content: Regularly update your video catalog with new releases or curated content to keep users engaged.