CLOUD APPLICATION DEVELOPMENT(CAD)

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

**Phase 4:** Development Part 2

**Step 1: Setting Up Your Project:**

Create a new directory for your project and initialize it with npm.

mkdir video-platform

cd video-platform

npm init -y

**Step 2: Install Dependencies:**

Install necessary Node.js packages: Express.js, Mongoose, and any other packages you might need.

npm install express mongoose body-parser multer

**Step 3: Create Server and Database Connection:**

Create an index.js file in the project directory.

**Program:**

// index.js

const express = require('express');

const mongoose = require('mongoose');

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

const multer = require('multer');

const app = express();

const port = process.env.PORT || 3000;

// MongoDB connection

mongoose.connect('mongodb://localhost:27017/video-platform', {

useNewUrlParser: true,

useUnifiedTopology: true

});

// Middleware

app.use(bodyParser.json());

app.use(bodyParser.urlencoded({ extended: true }));

// Routes

// Define your routes for handling video uploads, retrieval, etc. here.

app.listen(port, () => {

console.log(Server is running on port ${port});

});

**Step 4: Define Video Model:**

Create a models directory and inside it, create a Video.js file to define the video schema.

**Program:**

// models/Video.js

const mongoose = require('mongoose');

const videoSchema = new mongoose.Schema({

title: { type: String, required: true },

description: { type: String },

videoUrl: { type: String, required: true } // Store the video file URL here

});

const Video = mongoose.model('Video', videoSchema);

module.exports = Video;

**Step 5: Implement Routes for Video Management:**

Create a routes directory and inside it, create a videos.js file to handle video-related routes.

**Program:**

// routes/videos.js

const express = require('express');

const router = express.Router();

const Video = require('../models/Video');

// Upload a video

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

// Handle video upload here and save video data to the database

});

// Get all videos

router.get('/videos', async (req, res) => {

try {

const videos = await Video.find();

res.json(videos);

} catch (err) {

res.status(500).json({ message: err.message });

}

});

// Get a specific video by ID

router.get('/videos/:id', async (req, res) => {

// Retrieve and send the video with the specified ID

});

module.exports = router;

**Step 6: Integrate Routes in the Main Server File:**

Modify your index.js file to use the video routes.

**Program:**

// index.js

const express = require('express');

const mongoose = require('mongoose');

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

const multer = require('multer');

const videoRoutes = require('./routes/videos');

const app = express();

const port = process.env.PORT || 3000;

// MongoDB connection

mongoose.connect('mongodb://localhost:27017/video-platform', {

useNewUrlParser: true,

useUnifiedTopology: true

});

// Middleware

app.use(bodyParser.json());

app.use(bodyParser.urlencoded({ extended: true }));

// Routes

app.use('/api', videoRoutes);

app.listen(port, () => {

console.log(Server is running on port ${port});

});

**Step 7: Implement Video Upload Logic:**

In the routes/videos.js file, implement the logic for video upload using multer and save the video data to the database.

**Program:**

// routes/videos.js

const express = require('express');

const router = express.Router();

const multer = require('multer');

const Video = require('../models/Video');

// Multer configuration for video upload

const storage = multer.diskStorage({

destination: function(req, file, cb) {

cb(null, 'uploads/'); // Save uploaded files to the 'uploads' directory

},

filename: function(req, file, cb) {

cb(null, Date.now() + '-' + file.originalname);

}

});

const upload = multer({ storage: storage });

// Upload a video

router.post('/upload', upload.single('video'), async (req, res) => {

try {

const { title, description } = req.body;

const videoUrl = req.file.path; // Video file path in the server

const video = new Video({

title: title,

description: description,

videoUrl: videoUrl

});

const savedVideo = await video.save();

res.json(savedVideo);

} catch (err) {

res.status(400).json({ message: err.message });

}

});

// Get all videos

router.get('/videos', async (req, res) => {

// Retrieve and send all videos from the database

});

// Get a specific video by ID

router.get('/videos/:id', async (req, res) => {

User to upload their movies and videos to platform

**Step 1: Create Server and Database Connection:**

Create an index.js file in your project directory.

**Program:**

// index.js

const express = require('express');

const mongoose = require('mongoose');

const multer = require('multer');

const path = require('path');

const app = express();

const port = process.env.PORT || 3000;

// MongoDB connection

mongoose.connect('mongodb://localhost:27017/video-upload-platform', {

useNewUrlParser: true,

useUnifiedTopology: true

});

// Multer configuration for video upload

const storage = multer.diskStorage({

destination: function (req, file, cb) {

cb(null, 'uploads/'); // Save uploaded files to the 'uploads' directory

},

filename: function (req, file, cb) {

cb(null, Date.now() + path.extname(file.originalname));

}

});

const upload = multer({ storage: storage });

// Define video schema and model (if you need to store video metadata)

const Video = mongoose.model('Video', {

filename: String,

originalname: String,

uploadDate: Date

});

// Route for handling video uploads

app.post('/upload', upload.single('video'), async (req, res) => {

try {

const video = new Video({

filename: req.file.filename,

originalname: req.file.originalname,

uploadDate: new Date()

});

await video.save();

res.status(201).send('Video uploaded successfully!');

} catch (error) {

console.error(error);

res.status(500).send('Internal Server Error');

}

});

app.listen(port, () => {

console.log(Server is running on port ${port});

});

**Step 2: Create Upload Form in Frontend:**

**Program:**

html

Copy code

<!-- index.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 Platform</title>

</head>

<body>

<h1>Upload Your Video</h1>

<form action="/upload" method="POST" enctype="multipart/form-data">

<input type="file" name="video" accept="video/\*" required>

<button type="submit">Upload</button>

</form>

</body>

</html>