Skill Sequence: A Comprehensive Training Application

Table of Contents

Introduction

The application is designed to guide users through a series of training videos while tracking their progress and providing feedback on their completion status. The system is built using the React.js framework for the frontend and Node.js with Express.js for the backend, with data stored in a MySQL database.

Application Architecture

The application follows a client-server architecture with a React frontend, a Node.js backend, and a MySQL database.

- **Frontend**: Built using React.js, this layer handles user interaction, displaying video content, and updating the UI based on the user's progress.
- **Backend**: The backend is powered by Express.js. It serves as the API layer that handles requests from the frontend, manages database operations, and controls the flow of the application.
- **Database**: A MySQL database is used to store user information, video data, and progress details.

Frontend Implementation

Frameworks and Libraries

- **React.js**: The main library used for building the user interface.
- **React Router**: Used for routing between different pages in the application.
- **Axios**: Used for making HTTP requests to the backend API.
- **React Icons**: Provides icons for the UI.
- **React Player**: A video player component based on HTML5 used to display the videos.
- **Tailwind CSS**: For styling the components.
- **Chakra UI**: Used for displaying the circular progress bar.

Code Overview

The frontend is composed of several React components, each responsible for a specific part of the application. Below is an overview of the key components:

Components

1. **App.js**

The main entry point for the React application, which sets up the routing for the different pages.

```
import React from 'react';
import { BrowserRouter as Router, Routes, Route } from 'react
-router-dom';
import Dashboard from './components/dashboard';
import Player from './components/player';
import Home_component from './components/home';
import VideoPage from './components/videopage';

function App() {
    return (
        <Router>
            <Routes>
            <Route path="/" element={<><Home_component/></>}
/>
                <Route path="/dashboard" element={<Dashboard
/>} />
                <Route path="/dashboard/video/:id" element={<
><Dashboard/></>} />
                <Route path="/dashboard/training-completed" e
lement={<><Home_component/></>} />
            </Routes>
        </Router>
    );
}

export default App;
```

1. **home.js**

Handles the logic for displaying the home page, including the option to start training or revisit completed modules.

```
import React, { useEffect, useState } from 'react';
import { Link } from 'react-router-dom';
import axios from 'axios';
import { FaAward, FaPlayCircle } from 'react-icons/fa';

const HomeComponent = () => {
    const [trainingStatus, setTrainingStatus] = useState(fals
e);
    const [id, setId] = useState(1);

    useEffect(() => {
        const fetchStatus = async () => {
            try {
                const response = await axios.get('http://loca
lhost:5000/api/status');
                const statusData = response.data;

                if (statusData[2] && statusData[2].completed)
{
                    setTrainingStatus(true);
                }
            } catch (error) {
                console.error('Error fetching video progres
s:', error);
            }
        };

        fetchStatus();
    }, []);

    return (
        <div className="min-h-screen bg-gradient-to-r from-bl
ue-900 to-indigo-900 flex justify-center items-center p-8">
            {!trainingStatus ? (
                <div className="text-center animate-fade-in-u
```

```
p">
                    <div className="flex flex-col items-cente
r">
                        <h1 className="text-white text-7xl fo
nt-bold mb-10">Welcome to Skill Sequence !</h1>
                        <div className="flex items-center gap
-4">
                            <FaPlayCircle className="text-whi
te text-6xl mt-5 mr-5 -translate-y-7 animate-bounce" />
                            <Link to={`/dashboard/video/${id}
`} className="inline-block mt-4">
                                <button className="bg-green-5
00 hover:bg-blue-700 text-white font-bold py-4 px-8 rounded-f
ull shadow-lg transition-transform transform hover:scale-105
focus:outline-none focus:ring-2 focus:ring-blue-400">
                                    Start Training
                                </button>
                            </Link>
                        </div>
                    </div>
                </div>
            ) : (
                <div className="text-center animate-fade-in-u
p">
                    <div className="flex flex-col items-cente
r">
                        <h2 className="text-white text-7xl fo
nt-bold mb-4">Congratulations!</h2>
                        <div className="flex items-center gap
-4">
                            <FaAward className="text-yellow-4
00 text-6xl animate-pulse" />
                            <p className="text-blue-200 text-
2xl">Your training is completed.</p>
                        </div>
                        <Link to={`/dashboard/video/1`} class
```

```
Name="inline-block mt-8">
                            <button className="bg-green-500 h
over:bg-green-700 text-white font-bold py-4 px-8 rounded-full
shadow-lg transition-transform transform hover:scale-105 focu
s:outline-none focus:ring-2 focus:ring-green-400">
                                Revisit the Modules?
                            </button>
                        </Link>
                    </div>
                </div>
            )}
        </div>
    );
};

export default HomeComponent;
```

1. **Dashboard.js**

Displays the list of videos, the user's progress, and the currently on going module as a part of training.

```
import React, { useEffect, useState } from 'react';
import { Link, useParams } from 'react-router-dom';
import axios from 'axios';
import Player from './player';
import { CircularProgress, CircularProgressLabel } from '@cha
kra-ui/react';

const Dashboard = () => {
    const [videos, setVideos] = useState([]);
    const [status, setStatus] = useState([]);
    const { id } = useParams();
    const [videoDetails, setVideoDetails] = useState([]);
    const [backstate, setBackstate] = useState(false);
    const [nextstate, setNextstate] = useState(false);
```

```
    useEffect(() => {
        const fetchData = async () => {
            try {
                const videoListResponse = await axios.get('ht
tp://localhost:5000/api/video-list');
                setVideos(videoListResponse.data);

                const statusResponse = await axios.get('htt
p://localhost:5000/api/status');
                setStatus(statusResponse.data);

                const videoDetailsResponse = await axios.get
('http://localhost:5000/api/video-details');
                setVideoDetails(videoDetailsResponse.data);

                const currentId = parseInt(id);

                setBackstate(currentId > 1);
                setNextstate(currentId < videoListResponse.da
ta.length);
            } catch (error) {
                console.error('Error fetching data:', error);
            }
        };

        fetchData();
    }, [id]);

    const completedVideos = status.filter(videoStatus => vide
oStatus.completed).length;
    const totalVideos = videos.length;
    const percentage = (completedVideos / totalVideos) * 100;

    const isCurrentVideoCompleted = status.some(videoStatus =
> videoStatus.video_id === parseInt(id) && videoStatus.comple
```

```
ted);
    const isLastVideoCompleted = status.some(videoStatus => v
ideoStatus.video_id === 3 && videoStatus.completed);

    return (
        <div className="min-h-screen bg-black text-white font
-sans">
            <div className="container mx-auto py-10 px-4">
                <div className="text-center mb-8">
                    <h1 className="text-5xl font-extrabold te
xt-blue-600 drop-shadow-lg tracking-wide">
                        Training Dashboard
                    </h1>
                </div>

                <div className="flex flex-wrap items-center j
ustify-between bg-gray-800 p-6 rounded-lg shadow-lg mb-6">
                    <div className="flex flex-col w-full lg:w
-2/3 ">
                        {videos.map((video) => (
                            <div key={video.id} className="mb
-4">
                                {status.some(videoStatus => v
ideoStatus.video_id === video.id && videoStatus.completed) ?
(
                                    <Link to={`/dashboard/vid
eo/${video.id}`} className="block text-lg font-semibold text-
green-400 bg-gray-700 p-4 rounded-lg hover:text-white hover:b
g-blue-600 transition-all duration-300">
                                        Module: {videoDetail
s.find(v => v.videoId === video.id)?.title || `Video ${video.
id}`}
                                    </Link>
                                ) : (
                                    <div to={`/dashboard/vide
o/${video.id}`} className="block text-lg font-semibold text-g
```

```
ray-400 bg-gray-700 p-4 rounded-lg hover:text-white hover:bg-
blue-600 cursor-not-allowed transition-all duration-300">
                                        Module: {videoDetail
s.find(v => v.videoId === video.id)?.title || `Video ${video.
id}`}
                                    </div>
                                )}
                            </div>
                        ))}
                    </div>

                    <div className="flex items-center justify
-center w-full lg:w-1/3 mt-5 lg:mt-0">
                        <div className="text-center">
                            <h2 className="text-3xl font-bold
mb-4 text-white">Your Progress</h2>
                            <CircularProgress value={Math.rou
nd(percentage)} size="120px" color="blue" trackColor="black">
                                <CircularProgressLabel classN
ame="text-white font-bold">{Math.round(percentage)}%</Circula
rProgressLabel>
                            </CircularProgress>
                            <div className="text-md mt-2 text
-gray-300">{completedVideos}/{totalVideos} completed</div>
                        </div>
                    </div>
                </div>

                <div className="flex justify-between mb-4">
                    {backstate ? (
                        <Link to={`/dashboard/video/${parseIn
t(id) - 1}`}>
                            <button className="bg-blue-500 te
xt-white py-2 px-5 rounded-lg hover:bg-blue-700 transition du
ration-300">Back</button>
                        </Link>
```

```
                    ) : (
                        <button className="bg-gray-600 text-w
hite py-2 px-5 rounded-lg cursor-not-allowed">Back</button>
                    )}
                    {isLastVideoCompleted ? (
                        <button className="bg-gray-600 text-w
hite py-2 px-5 rounded-lg cursor-not-allowed">Completed Train
ing</button>
                    ) : (
                        nextstate && isCurrentVideoCompleted
? (
                            <Link to={`/dashboard/video/${par
seInt(id) + 1}`}>
                                <button className="bg-blue-50
0 text-white py-2 px-5 rounded-lg hover:bg-blue-700 transitio
n duration-300">Next</button>
                            </Link>
                        ) : (
                            <button className="bg-gray-600 te
xt-white py-2 px-5 rounded-lg cursor-not-allowed">Not Complet
ed</button>
                        )
                    )}
                </div>

                <div className="bg-gray-800 p-6 rounded-lg sh
adow-lg flex flex-col lg:flex-row">
                    <div className="lg:w-2/3">
                        {videoDetails.map((videoDetail) => (
                            videoDetail.videoId === parseInt
(id) ? (
                                <div key={videoDetail.videoI
d}>
                                    <h2 className="text-4xl f
ont-semibold mb-4 text-blue-400 text-center pb-2">{videoDetai
l.title}</h2>
```

```
                                        {Object.entries(videoDeta
il.content).map(([point_no, desc]) => (
                                            <p key={point_no} cla
ssName="mb-3 text-lg text-gray-300 p-3">-- {desc}</p>
                                        ))}
                                    </div>
                                ) : null
                            ))}
                        </div>
                        <div className="-mt-12">
                            <Player />
                        </div>
                    </div>
                </div>
            </div>
        );
};

export default Dashboard;
```

1. **player.js**

Handles the video playback and interaction. Tracks video completion and reports progress to the backend.

```
import React, { useEffect, useState, useRef, memo } from 'rea
ct';
import ReactPlayer from 'react-player';
import axios from 'axios';
import { useParams, useNavigate } from 'react-router-dom';

const Player = memo(() => {
    const { id } = useParams();
    const navigate = useNavigate();
```

```
    const [videoUrl, setVideoUrl] = useState('');
    const [progress, setProgress] = useState(0);
    const playerRef = useRef(null);
    const [readyToPlay, setReadyToPlay] = useState(false);
    const [loading, setLoading] = useState(true);

    useEffect(() => {
        let isMounted = true;

        const fetchVideoData = async () => {
            try {
                setLoading(true);

                const response = await axios.get(`http://loca
lhost:5000/api/videos/${id}`, {
                    responseType: 'arraybuffer',
                });

                if (isMounted) {
                    const videoBlob = new Blob([response.dat
a], { type: 'video/mp4' });
                    const videoUrl = URL.createObjectURL(vide
oBlob);
                    setVideoUrl(videoUrl);

                    const progressResponse = await axios.get
(`http://localhost:5000/api/progress/${id}`);
                    const savedProgress = progressResponse.da
ta.progress || 0;

                    setProgress(savedProgress);
                    setReadyToPlay(true);
                }
            } catch (error) {
                if (isMounted) {
                    console.error('Error fetching video dat
```

```
a:', error);
                }
            } finally {
                if (isMounted) {
                    setLoading(false);
                }
            }
        };

        fetchVideoData();

        return () => {
            isMounted = false;
            URL.revokeObjectURL(videoUrl);
        };
    }, [id]);

    const handleProgress = (state) => {
        if (state.playedSeconds <= progress) return;

        setProgress(state.playedSeconds);
        axios.post('http://localhost:5000/api/save-progress',
{
            userId: 1,
            videoId: parseInt(id),
            progress: state.playedSeconds,
        }).catch((error) => console.error('Error saving progr
ess:', error));
    };

    const handleReady = () => {
        if (readyToPlay && playerRef.current) {
            playerRef.current.seekTo(progress, 'seconds');
            setReadyToPlay(false);
        }
    };
```

```
    const handleEnded = async () => {
        try {
            await axios.post('http://localhost:5000/api/save-
status', {
                userId: 1,
                videoId: parseInt(id),
                completed: true,
            });

            const nextVideoId = parseInt(id) + 1;

            if (nextVideoId <= 3) {
                await axios.post('http://localhost:5000/api/s
ave-progress', {
                    userId: 1,
                    videoId: nextVideoId,
                    progress: 0,
                });

                navigate(`/dashboard/video/${nextVideoId}`);
            } else {
                navigate(`/dashboard/training-completed`);
            }
        } catch (error) {
            console.error('Error handling video end:', erro
r);
        }
    };

    const handleSeek = (seconds) => {
        if (seconds > progress) {
            playerRef.current.seekTo(progress);
        }
    };
```

```
    return (
        <div className="p-5">
            {loading ? (
                <div>Loading video...</div>
            ) : (
                <ReactPlayer
                    ref={playerRef}
                    url={videoUrl}
                    controls
                    playing={false}
                    playbackRate={1}
                    onReady={handleReady}
                    onProgress={handleProgress}
                    onEnded={handleEnded}
                    onSeek={handleSeek}
                    pip={false}
                    progressInterval={1000}
                    width="100%"
                    config={{ file: { attributes: { controlsL
ist: 'nodownload noremoteplayback noplaybackrate' } } }}
                />
            )}
        </div>
    );
});

export default Player;
```

Backend Implementation

Frameworks and Libraries

- **Express.js**: The web framework used to build the REST API.
- **MySQL**: The relational database used to store user data and video progress.

- **CORS**: Middleware to enable Cross-Origin Resource Sharing.
- **Body-Parser**: Middleware to parse incoming request bodies.

Setting up the Connection

The backend server is set up using Express.js and connects to a MySQL database. Here's the code for establishing the connection:

```
const express = require('express');
const mysql = require('mysql2');
const cors = require('cors');
const app = express();
app.use(express.json());


app.use(cors());

const db = mysql.createConnection({
    'user': 'avnadmin',
    'password': 'AVNS_h4eCBRzKkM_Bl1FWHqq',
    'host': 'skill-skillsequence.h.aivencloud.com',
    'database': 'skillsequence',
    'port': '28152',
});
db.connect(err => {
    if (err) {
        console.error('Database connection failed: ' + err.stacl
        return;
    }
    console.log('Connected to database.');
});
```

API Endpoints

1. **GET /api/video-list**

Retrieves the list of all videos in the sequence.

```
app.get('/api/video-list', (req, res) => {
    const query = 'SELECT id, title FROM videos';
    db.query(query, (err, results) => {
        if (err) {
            return res.status(500).send(err);
        }
        res.json(results);
    });
});
```

2. **GET /api/video/:id**

   Retrieves a specific video by its ID.

```
app.get('/api/videos/:id', (req, res) => {
    const videoId = req.params.id;
    db.query('SELECT video FROM videos WHERE id = ?', [vid
eoId], (err, results) => {
        if (err) throw err;
        if (results.length > 0) {
            const videoData = results[0].video;
            res.writeHead(200, {
                'Content-Type': 'video/mp4',
                'Content-Length': videoData.length
            });

            res.end(videoData);
        } else {
            res.status(404).send('Video not found');
        }
    });
});
```

3. **GET /api/video-details**

Retrieves details about the module title and content for all videos.

```
app.get('/api/video-details', (req, res) => {

    db.query('SELECT * FROM video_details', (err, results)
=> {
        if (err) {
            return res.status(500).send(err);
        }
        res.json(results);
        console.log(results);

    });
});
```

4. **GET /api/status**

   Retrieves the completion status of all videos.

```
app.get('/api/status', (req, res) => {
    const userId = 1;
    db.query('SELECT video_id,completed FROM vid_progress
WHERE user_id = ?', [userId], (err, results) => {
        if (err) {
            return res.status(500).send(err);
        }
        res.json(results);
        console.log(results);

    });
});
```

5. **GET /api/**progress/:id

   Retrieves the completion status of all videos.

```
app.get('/api/progress/:id', (req, res) => {
    const videoId = req.params.id;
    const userId = 1;

    db.query('SELECT progress FROM vid_progress WHERE user
_id = ? AND video_id = ?', [userId, videoId], (err, result
s) => {
        if (err) {
            return res.status(500).send(err);
        }
        if (results.length > 0) {
            res.json({ progress: results[0].progress });
        } else {
            res.json({ progress: 0 });
        }
    });
});
```

6. **GET /api/**progress/:id

   Retrieves the video progress data of a specific video.

```
app.get('/api/progress/:id', (req, res) => {
    const videoId = req.params.id;
    const userId = 1;

    db.query('SELECT progress FROM vid_progress WHERE user
_id = ? AND video_id = ?', [userId, videoId], (err, result
s) => {
        if (err) {
            return res.status(500).send(err);
        }
        if (results.length > 0) {
            res.json({ progress: results[0].progress });
        } else {
```

```
            res.json({ progress: 0 });
        }
    });
});
```

7. **POST /api/**save-progress

   Saves the video progress data of a specific video.

```
app.post('/api/save-progress', (req, res) => {
    const { videoId, progress } = req.body;
    const userId = 1;
    db.query(
        'INSERT INTO vid_progress (user_id, video_id, prog
ress) VALUES (?, ?, ?) ON DUPLICATE KEY UPDATE progress =
?',
        [userId, videoId, progress, progress],
        (err) => {
            if (err) {
                return res.status(500).send(err);
            }
            res.sendStatus(200);
        }
    );
});
```

8. **POST /api/**save-status

   Saves the completion status of a specific video.

```
app.post('/api/save-status', (req, res) => {
    const { videoId,completed } = req.body;
    const progress = 0;
    const userId = 1;

    db.query(
        'UPDATE vid_progress SET progress = ?, completed =
```

```
? WHERE user_id = ? AND video_id = ? ',
        [progress, completed, userId,videoId],
        (err) => {
            if (err) {
                return res.status(500).send(err);
            }
            res.sendStatus(200);
        }
    );
});
```

Database Schema

The database schema consists of three main tables:

- **videos**: Stores information about each video, including its title and the video file itself.
- **vid_progress**: Tracks user progress for each video, including how much of the video has been watched and whether it's been completed.
- **video_details**: Contains additional details about each video, such as its title and content in JSON format.

```
CREATE TABLE videos (
    id INT AUTO_INCREMENT PRIMARY KEY,
    title VARCHAR(255) NOT NULL,
    video BLOB,
    upload_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE vid_progress (
    user_id INT,
    video_id INT,
    progress FLOAT,
```

```
    completed BOOLEAN,
    PRIMARY KEY (user_id,video_id)
);

CREATE TABLE video_details (
        videoId INT PRIMARY KEY,
    title VARCHAR(255),
    content JSON
);
```
