**E-learning platform development**

**Front-End:** React.js

**Back-End:** Node.js with Express.js

**Database:** MongoDB

**Authentication**: JSON Web Tokens (JWT)

**1. Project Structure**

e-learning-platform/

│

├── backend/

│ ├── config/

│ │ └── db.js

│ ├── controllers/

│ │ ├── authController.js

│ │ └── courseController.js

│ ├── models/

│ │ ├── User.js

│ │ └── Course.js

│ ├── routes/

│ │ ├── authRoutes.js

│ │ └── courseRoutes.js

│ ├── middleware/

│ │ └── authMiddleware.js

│ ├── server.js

│

├── frontend/

│ ├── public/

│ └── src/

│ ├── components/

│ ├── pages/

│ ├── services/

│ └── App.js

│

└── package.json

**2. Backend (Node.js + Express)**

backend/server.js

javascript

const express = require('express');

const cors = require('cors');

const mongoose = require('mongoose');

const authRoutes = require('./routes/authRoutes');

const courseRoutes = require('./routes/courseRoutes');

const { requireAuth } = require('./middleware/authMiddleware');

require('dotenv').config();

const app = express();

const PORT = process.env.PORT || 5000;

app.use(express.json());

app.use(cors());

**// Connect to MongoDB**

mongoose.connect(process.env.MONGO\_URI, { useNewUrlParser: true, useUnifiedTopology: true })

.then(() => console.log('MongoDB connected'))

.catch(err => console.log(err));

// Routes

app.use('/api/auth', authRoutes);

app.use('/api/courses', requireAuth, courseRoutes);

app.listen(PORT, () => {

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

});

***backend/models/User.js***

javascript

const mongoose = require('mongoose');

const bcrypt = require('bcrypt');

const userSchema = new mongoose.Schema({

username: { type: String, required: true, unique: true },

email: { type: String, required: true, unique: true },

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

role: { type: String, default: 'student' }, // 'student' or 'instructor'

});

// Hash the password before saving

userSchema.pre('save', async function (next) {

if (!this.isModified('password')) return next();

this.password = await bcrypt.hash(this.password, 10);

next();

});

**// Password validation**

userSchema.methods.comparePassword = async function (password) {

return await bcrypt.compare(password, this.password);

};

module.exports = mongoose.model('User', userSchema);

backend/models/Course.js

javascript

const mongoose = require('mongoose');

const courseSchema = new mongoose.Schema({

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

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

instructor: { type: mongoose.Schema.Types.ObjectId, ref: 'User', required: true },

students: [{ type: mongoose.Schema.Types.ObjectId, ref: 'User' }],

});

module.exports = mongoose.model('Course', courseSchema);

***backend/controllers/authController.js***

javascript

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

const jwt = require('jsonwebtoken');

**// Generate JWT token**

const createToken = (user) => {

return jwt.sign({ id: user.\_id, role: user.role }, process.env.JWT\_SECRET, { expiresIn: '1d' });

};

**// User Registration**

exports.register = async (req, res) => {

const { username, email, password } = req.body;

try {

const user = await User.create({ username, email, password });

const token = createToken(user);

res.status(201).json({ user, token });

} catch (err) {

res.status(400).json({ error: 'Registration failed' });

}

};

**// User Login**

exports.login = async (req, res) => {

const { email, password } = req.body;

try {

const user = await User.findOne({ email });

if (!user || !(await user.comparePassword(password))) {

return res.status(400).json({ error: 'Invalid credentials' });

}

const token = createToken(user);

res.status(200).json({ user, token });

} catch (err) {

res.status(400).json({ error: 'Login failed' });

}

};

***backend/routes/authRoutes.js***

javascript

const express = require('express');

const { register, login } = require('../controllers/authController');

const router = express.Router();

router.post('/register', register);

router.post('/login', login);

module.exports = router;

***backend/routes/courseRoutes.js***

javascript

const express = require('express');

const { getCourses, createCourse } = require('../controllers/courseController');

const router = express.Router();

router.get('/', getCourses);

router.post('/', createCourse);

module.exports = router; ***backend/controllers/courseController.js***

javascript

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

// Get all courses

exports.getCourses = async (req, res) => {

try {

const courses = await Course.find().populate('instructor');

res.status(200).json(courses);

} catch (err) {

res.status(400).json({ error: 'Unable to fetch courses' });

}

};

// Create a course (Instructor only)

exports.createCourse = async (req, res) => {

const { title, description } = req.body;

try {

const course = await Course.create({

title,

description,

instructor: req.user.id,

});

res.status(201).json(course);

} catch (err) {

res.status(400).json({ error: 'Unable to create course' });

}

};

***backend/middleware/authMiddleware.js***

javascript

const jwt = require('jsonwebtoken');

exports.requireAuth = (req, res, next) => {

const token = req.headers.authorization.split(' ')[1];

if (!token) return res.status(401).json({ error: 'Unauthorized' });

jwt.verify(token, process.env.JWT\_SECRET, (err, decodedToken) => {

if (err) return res.status(401).json({ error: 'Unauthorized' });

req.user = decodedToken;

next();

});

};

**3. Frontend (React.js)**

***frontend/src/App.js***

javascript

import React from 'react';

import { BrowserRouter as Router, Route, Switch } from 'react-router-dom';

import Login from './pages/Login';

import Register from './pages/Register';

import Courses from './pages/Courses';

import CreateCourse from './pages/CreateCourse';

function App() {

return (

<Router>

<Switch>

<Route path="/login" component={Login} />

<Route path="/register" component={Register} />

<Route path="/courses" component={Courses} />

<Route path="/create-course" component={CreateCourse} />

</Switch>

</Router>

);

}

**export default App**;

***frontend/src/services/authService.js***

javascript

import axios from 'axios';

const API\_URL = 'http://localhost:5000/api/auth/';

export const register = async (userData) => {

const response = await axios.post(API\_URL + 'register', userData);

return response.data;

};

export const login = async (userData) => {

const response = await axios.post(API\_URL + 'login', userData);

return response.data;

};

***frontend/src/pages/Login.js***

javascript

import React, { useState } from 'react';

import { login } from '../services/authService';

const Login = () => {

const [email, setEmail] = useState('');

const [password, setPassword] = useState('');

const handleSubmit = async (e) => {

e.preventDefault();

const user = { email, password };

try {

const data = await login(user);

console.log('User logged in', data);

} catch (error) {

console.error('Error logging in', error);

}

};

return (

<form onSubmit={handleSubmit}>

<input type="email" placeholder="Email" value={email} onChange={(e) => setEmail(e.target.value)} />

<input type="password" placeholder="Password" value={password} onChange={(e) => setPassword(e.target.value)} />

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

</form>

);

};

export default Login;

**4. Conclusion**

This is a basic structure of an e-learning platform with user registration, login, course creation, and listing functionalities.