Curd of courses and lectures backend and front end with login and logout:-

1□Backend (Node + Express + MongoDB)

Make sure you are in your backend folder, e.g., `F:\mern-stack\project5\backend`.

Install Node dependencies

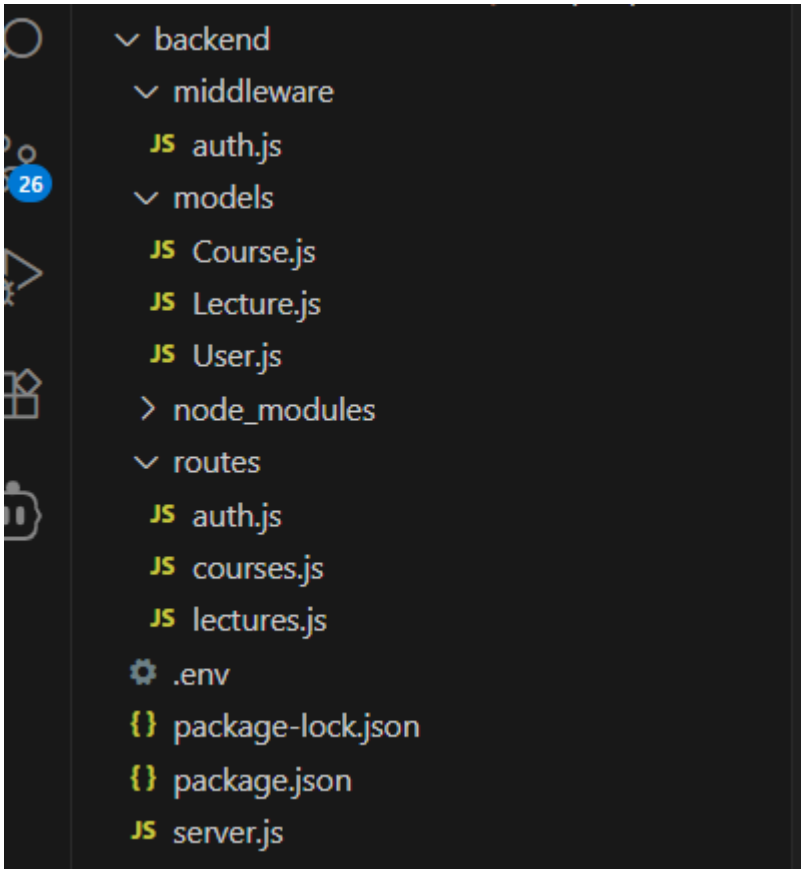
mkdir backend

cd backend

```
npm init -y
npm install express mongoose dotenv bcryptjs jsonwebtoken cors
```

Your backend folder structure should look like this:

```
backend/
├─ routes/
│   ├─ auth.js
│   ├─ courses.js
│   └─ lectures.js
├─ models/
│   ├─ User.js
│   └─ Course.js
├─ middleware/
│   └─ auth.js
├─ server.js
└─ .env
```


backend
middleware
auth.js
models
Course.js
Lecture.js
User.js
node_modules
routes
auth.js
courses.js
lectures.js
.env
package-lock.json
package.json
server.js
26

models/Course.js:-

```
// backend/models/course.js
const mongoose = require('mongoose');

const courseSchema = new mongoose.Schema({
  title: { type: String, required: true },
  description: String,
  image: String,
  createdBy: { type: mongoose.Schema.Types.ObjectId, ref: 'User' },
}, { timestamps: true });

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

models/Lecture.js file:-

```
const mongoose = require('mongoose');

const lectureSchema = new mongoose.Schema({
```

```
  title: { type: String, required: true },
  content: { type: String, required: true },
  videoUrl: String,
  course: { type: mongoose.Schema.Types.ObjectId, ref: 'Course', required: true }
}, { timestamps: true });

module.exports = mongoose.model('Lecture', lectureSchema);
```

models/User.js file :-

```
const mongoose = require('mongoose');
const userSchema = new mongoose.Schema({
  name: { type: String, required: true },
  email: { type: String, required: true, unique: true },
  password: { type: String, required: true },
  role: { type: String, enum: ['admin','student'], default: 'student' },
  enrolledCourses: [{ type: mongoose.Schema.Types.ObjectId, ref: 'Course' }]
}, { timestamps: true });

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

middleware/auth.js file code:-

```
// middleware/auth.js
```

```
const jwt = require('jsonwebtoken');
require('dotenv').config();

/**
 * Middleware to authenticate requests by verifying JWT token.
 * Expects Authorization header in the format: 'Bearer <token>'.
 * On success, attaches decoded token payload to req.user.
 */
const auth = (req, res, next) => {
  const header = req.headers.authorization;
  console.log('Authorization header:', header);

  if (!header) {
    return res.status(401).json({ message: 'No token provided in Authorization header' });
  }

  const token = header.split(' ')[1];
  if (!token) {
    return res.status(401).json({ message: 'Malformed token in Authorization header' });
  }

  try {
    const decoded = jwt.verify(token, process.env.JWT_SECRET);
    console.log('Decoded token:', decoded);
    req.user = decoded; // decoded token contains user id and role
    next();
```

```
  } catch (err) {
    console.error('Token verification failed:', err);
    return res.status(401).json({ message: 'Invalid or expired token' });
  }
};

/**
 * Middleware to authorize admin users only.
 * Checks if req.user.role === 'admin'.
 */
const adminOnly = (req, res, next) => {
  if (req.user && req.user.role === 'admin') {
    return next();
  }
  return res.status(403).json({ message: 'Admin only access' });
};

module.exports = { auth, adminOnly };
```

rotues/auth.js file code:-

```
// routes/auth.js
const express = require('express');
const router = express.Router();
const bcrypt = require('bcryptjs');
```

```
const jwt = require('jsonwebtoken');
const User = require('../models/User');
const { auth, adminOnly } = require('../middleware/auth');
require('dotenv').config();
// Register
router.post('/register', async (req, res) => {
const { name, email, password, role } = req.body;
console.log("Incoming registration data:", req.body);

try {
const existing = await User.findOne({ email });
if (existing) return res.status(400).json({ message: 'User already exists'
});
const hashedPassword = await bcrypt.hash(password, 10);
const user = new User({ name, email, password: hashedPassword, role });
await user.save();
res.status(201).json({ message: 'User registered' });
} catch (err) {
  console.error("Register error:", err); // ðŸ”  Add this
  res.status(500).json({ message: 'Server error', error: err.message });
}
});
// Login
router.post('/login', async (req, res) => {
const { email, password } = req.body;
try {
const user = await User.findOne({ email });
```

```
if (!user) return res.status(400).json({ message: 'Invalid credentials' });
const isMatch = await bcrypt.compare(password, user.password);
if (!isMatch) return res.status(400).json({ message: 'Invalid credentials'
});
const token = jwt.sign(
{ id: user._id, role: user.role },
process.env.JWT_SECRET,
{ expiresIn: '1d' }
);
res.json({ token, user: { id: user._id, name: user.name, role: user.role }
});
} catch (err) {
res.status(500).json({ message: 'Server error' });
}
});



// Get all students - admin only
router.get('/students', auth, adminOnly, async (req, res) => {
  try {
    const students = await User.find({ role: 'student' }).select('-password');
    res.json(students);
  } catch (err) {
    console.error(err);
    res.status(500).json({ message: 'Server error' });
  }
});
```

```
module.exports = router;
```

rotues/courses.js file code:-

```
// routes/courses.js
const express = require('express');
const router = express.Router();
const Course = require('../models/Course');
const User = require('../models/User');
const { auth, adminOnly } = require('../middleware/auth');

console.log('auth middleware:', auth, 'adminOnly middleware:', adminOnly); // debug

// ------------------------------
// CREATE COURSE (Admin Only)
// ------------------------------
router.post('/', auth, adminOnly, async (req, res) => {
  const { title, description, image } = req.body;
  try {
    const course = new Course({
      title,
      description,
      image,
      createdBy: req.user.id,
    });
    await course.save();
    res.status(201).json(course);
  } catch (err) {
```

```
    console.error('Create course error:', err);
    res.status(500).json({ message: 'Server error', error: err.message });
  }
});

// ----------------------------
// GET ALL COURSES (Public)
// ----------------------------
router.get('/', async (req, res) => {
  try {
    const courses = await Course.find().populate('createdBy', 'name');
    res.json(courses);
  } catch (err) {
    console.error('Get courses error:', err);
    res.status(500).json({ message: 'Server error', error: err.message });
  }
});

// ----------------------------
// GET SINGLE COURSE
// ----------------------------
router.get('/:id', async (req, res) => {
  try {
    const course = await Course.findById(req.params.id).populate('createdBy', 'name');
    if (!course) return res.status(404).json({ message: 'Course not found' });
    res.json(course);
  } catch (err) {
```

```
    console.error('Get course error:', err);
    res.status(500).json({ message: 'Server error', error: err.message });
  }
});

// ------------------------------
// UPDATE COURSE (Admin Only)
// ------------------------------
router.put('/:id', auth, adminOnly, async (req, res) => {
  try {
    const course = await Course.findByIdAndUpdate(req.params.id, req.body, { new: true });
    if (!course) return res.status(404).json({ message: 'Course not found' });
    res.json(course);
  } catch (err) {
    console.error('Update course error:', err);
    res.status(500).json({ message: 'Server error', error: err.message });
  }
});

// ------------------------------
// DELETE COURSE (Admin Only)
// ------------------------------
router.delete('/:id', auth, adminOnly, async (req, res) => {
  try {
    const course = await Course.findByIdAndDelete(req.params.id);
    if (!course) return res.status(404).json({ message: 'Course not found' });
    res.json({ message: 'Course deleted' });
```

```
  } catch (err) {
    console.error('Delete course error:', err);
    res.status(500).json({ message: 'Server error', error: err.message });
  }
});

// ------------------------------
// ASSIGN COURSE TO STUDENT (Admin Only)
// ------------------------------
router.post('/:courseId/assign/:studentId', auth, adminOnly, async (req, res) => {
  const { courseId, studentId } = req.params;
  try {
    const student = await User.findById(studentId);
    if (!student) return res.status(404).json({ message: 'Student not found' });

    if (!Array.isArray(student.enrolledCourses)) student.enrolledCourses = [];

    if (!student.enrolledCourses.includes(courseId)) {
      student.enrolledCourses.push(courseId);
      await student.save();
    }

    res.json({ message: 'Course assigned to student successfully' });
  } catch (err) {
    console.error('Assign course error:', err);
    res.status(500).json({ message: 'Server error', error: err.message });
  }
```

```
});


// ------------------------------
// GET ENROLLED COURSES FOR LOGGED-IN USER
// ------------------------------
router.get('/my/enrolled', auth, async (req, res) => {
  try {
    const user = await User.findById(req.user.id).populate('enrolledCourses');
    if (!user) return res.status(404).json({ message: 'User not found' });


    res.json(user.enrolledCourses);
  } catch (err) {
    console.error('Get enrolled courses error:', err);
    res.status(500).json({ message: 'Server error', error: err.message });
  }
});


module.exports = router;
```

routes/lectures.js file code:-

```
const express = require('express');
const router = express.Router();
const Lecture = require('../models/Lecture');
const Course = require('../models/Course');
const { auth, adminOnly } = require('../middleware/auth');

// Create a lecture
router.post('/', auth, adminOnly, async (req, res) => {
  const { title, content, videoUrl, courseId } = req.body;
  try {
    const lecture = new Lecture({ title, content, videoUrl, course: courseId });
    await lecture.save();
    res.status(201).json(lecture);
  } catch (err) {
    res.status(500).json({ message: 'Server error' });
  }
});

// Get lectures for a course
router.get('/:courseId', auth, async (req, res) => {
  try {
    const lectures = await Lecture.find({ course: req.params.courseId });
    res.json(lectures);
  } catch (err) {
```

```
    res.status(500).json({ message: 'Server error' });
  }
});

// Update a lecture
router.put('/:id', auth, adminOnly, async (req, res) => {
  try {
    const lecture = await Lecture.findByIdAndUpdate(req.params.id, req.body, { new: true });
    res.json(lecture);
  } catch (err) {
    res.status(500).json({ message: 'Server error' });
  }
});

// Delete a lecture
router.delete('/:id', auth, adminOnly, async (req, res) => {
  try {
    await Lecture.findByIdAndDelete(req.params.id);
    res.json({ message: 'Lecture deleted' });
  } catch (err) {
    res.status(500).json({ message: 'Server error' });
  }
});

module.exports = router;
```

.env file :-

```
PORT=5000
JWT_SECRET=Yx9$kD@7v!wLqP3z#NcRtUe2FgHbJmX1
MONGO_URI=mongodb://localhost:27017/mydbcourse
```

server.js file code :-

```
// server.js
const express = require('express');
const mongoose = require('mongoose');
const cors = require('cors');
require('dotenv').config();

const authRoutes = require('./routes/auth');
const courseRoutes = require('./routes/courses');
const lectureRoutes = require('./routes/lectures');

const app = express();

// ----------------------------
// MIDDLEWARE
```

```
// ------------------------------

app.use(express.json());

app.use(cors());


// ------------------------------

// ROUTES

// ------------------------------

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

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

app.use('/api/lectures', lectureRoutes);


// ------------------------------

// HEALTH CHECK

// ------------------------------

app.get('/', (req, res) => {

  res.send('API is running');

});


// ------------------------------

// MONGODB CONNECTION & SERVER START

// ------------------------------

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

const MONGO_URI = process.env.MONGO_URI;


if (!MONGO_URI) {

  console.error('⚠ MONGO_URI is not defined in .env');

  process.exit(1);
```

```
}


mongoose
  .connect(MONGO_URI, {
    useNewUrlParser: true,
    useUnifiedTopology: true,
  })
  .then(() => {
    console.log('✅ MongoDB connected');
    app.listen(PORT, () => console.log(`🚀 Server started on port ${PORT}`));
  })
  .catch((err) => {
    console.error('❌ MongoDB connection error:', err);
    process.exit(1);
  });


// ------------------------------
// GLOBAL ERROR HANDLER
// ------------------------------
app.use((err, req, res, next) => {
  console.error('Global error:', err);
  res.status(500).json({ message: 'Internal server error', error: err.message });
});



============================
```

React Js Front End Code :-

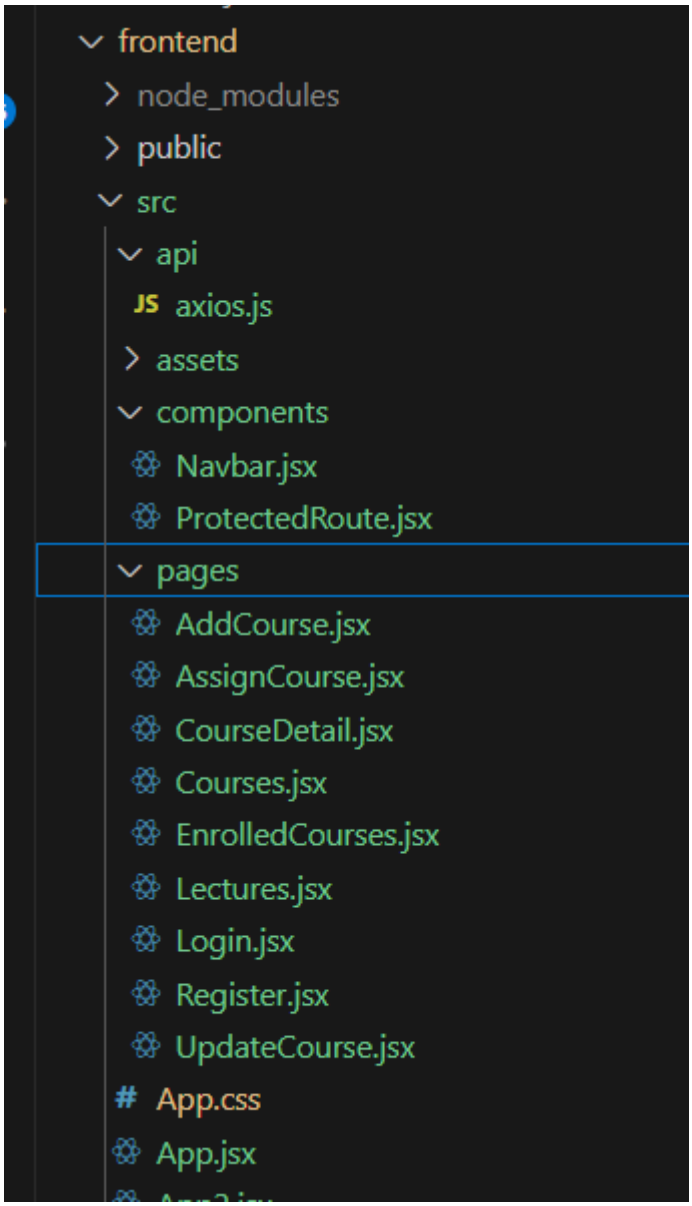
2□Frontend (React)

Make sure you are in your frontend folder, e.g., `F:\mern-stack\project5\frontend`.

Install React dependencies

```
npx create-react-app .   # if you haven't already initialized React
npm install axios react-router-dom
```

project folder structure:-



api/axios.js file code:-

```
import axios from "axios";

// Create Axios instance
const API = axios.create({
  baseURL: "http://localhost:5000/api", // Backend base URL
});

// Attach token to each request if exists
API.interceptors.request.use((req) => {
  const token = localStorage.getItem("token");
  if (token) {
    req.headers = req.headers || {}; // ensure headers object exists
    req.headers.Authorization = `Bearer ${token}`;
  }
  return req;
});

export default API;
```

components/ProtectedRoute.jsx:-

```
import { Navigate } from "react-router-dom";

export default function ProtectedRoute({ children, role }) {

const token = localStorage.getItem("token");

const user = JSON.parse(localStorage.getItem("user") || "{}");

// If no token, redirect to login

if (!token) return <Navigate to="/login" />;

// If role is required (e.g., "admin") and user doesn’t match → block

if (role && user.role !== role) return <Navigate to="/courses" />;

// Otherwise, render children

return children;

}
```

components/Navbar.jsx file:-

```
import { Link, useNavigate } from "react-router-dom";
import { useEffect, useState } from "react";
export default function Navbar() {
const [user, setUser] = useState(null);
const navigate = useNavigate();
useEffect(() => {
const storedUser = localStorage.getItem("user");
if (storedUser) setUser(JSON.parse(storedUser));
}, []);
const handleLogout = () => {
localStorage.removeItem("token");
localStorage.removeItem("user");
setUser(null);
navigate("/login");
};
return (
<nav style={styles.nav}>
<div>
<Link to="/" style={styles.link}> Home</Link>
<Link to="/courses" style={styles.link}> Courses</Link>
{user?.role === "admin" && (
<Link to="/add-course" style={styles.link}> Add Course</Link>



)}
```

```
</div>
<div>
{!user ? (
<>
<Link to="/login" style={styles.link}>Login</Link>
<Link to="/register" style={styles.link}>Register</Link>
</>
) : (
<>
<span style={styles.user}>👤 Hi, {user.name}</span>
<button onClick={handleLogout} style={styles.btn}>Logout</button>
</>
)}
</div>
</nav>
);
}
const styles = {
nav: {
display: "flex",
justifyContent: "space-between",
alignItems: "center",
padding: "10px 20px",
background: "#333",
color: "white",
},
```

```
link: {
color: "white",
margin: "0 10px",
textDecoration: "none",
},
user: {
marginRight: "15px",
},
btn: {
background: "red",
color: "white",
border: "none",
padding: "5px 10px",
cursor: "pointer",
borderRadius: "5px",
},
};
```

pages/AddCourse.jsx:-

```
import { useState } from "react";
import { useNavigate } from "react-router-dom";
import API from "../api/axios";
export default function AddCourse() {
const [form, setForm] = useState({ title: "", description: "", image: "" });
const navigate = useNavigate();
const handleChange = (e) => setForm({ ...form, [e.target.name]: e.target.value
});
const handleSubmit = async (e) => {
e.preventDefault();
try {
await API.post("/courses", form);
navigate("/courses");
} catch (err) {
alert("⚠ Only admin can create courses");
}
};
return (
<div>
<h2>Add Course</h2>
<form onSubmit={handleSubmit}>
<input name="title" placeholder="Title" onChange={handleChange} />
<textarea name="description" placeholder="Description"
onChange={handleChange}></textarea>
<input name="image" placeholder="Image URL" onChange={handleChange} />
<button type="submit">Add</button>
</form>
```

```
</div>
);
}
```

pages/AssignCourse.jsx:-

```
// Example idea - AssignCourse.js
import { useEffect, useState } from "react";
import { useParams } from "react-router-dom";

import API from "../api/axios";

export default function AssignCourse() {
  const { courseId } = useParams();
  const [students, setStudents] = useState([]);
  const [selected, setSelected] = useState("");

  useEffect(() => {
  API.get('/auth/students')
    .then((res) => setStudents(res.data))
    .catch((err) => {
      console.error('Failed to fetch students:', err);
      alert('Failed to load students');
    });
}, []);
```

```
 const assign = async () => {
  try {
    if (!selected) {
      alert('Please select a student!');
      return;
    }
    await API.post(`/courses/${courseId}/assign/${selected}`);
    alert('✅ Assigned');
  } catch (error) {
    console.error(error);
    alert('❌ Error assigning course');
  }
 };



  return (
    <div>
      <h3>Assign Course to Student</h3>
      <select onChange={(e) => setSelected(e.target.value)}>
        <option>Select student</option>
        {students.map((s) => (
          <option key={s._id} value={s._id}>{s.name}</option>
        ))}
      </select>
      <button onClick={assign}>Assign</button>
    </div>
```

```
  );
}
```

pages/CourseDetails.jsx:-

```
import { useEffect, useState } from "react";
import { useParams, useNavigate, Link } from "react-router-dom";
import API from "../api/axios";
export default function CourseDetail() {
const { id } = useParams();
const [course, setCourse] = useState(null);
const [user, setUser] = useState(null);
const navigate = useNavigate();
useEffect(() => {
API.get(`/courses/${id}`).then((res) => setCourse(res.data));
const storedUser = localStorage.getItem("user");
if (storedUser) setUser(JSON.parse(storedUser));
}, [id]);
const handleDelete = async () => {
try {
await API.delete(`/courses/${id}`);
navigate("/courses");
} catch (err) {
alert("⚠ Not authorized or error occurred");
```

```
}
};
if (!course) return <p>Loading...</p>;
return (
<div>
<h2>{course.title}</h2>
<p>{course.description}</p>
<img src={course.image} alt={course.title} width="200" />
{/* Admin actions */}
{user?.role === "admin" && (
 <div>
 <Link to={`/assign-course/${course._id}`}> Assign to Student</Link>

   <Link to={`/update-course/${course._id}`}>✏ Edit</Link>
   <Link to={`/course/${course._id}/lectures`}> Manage Lectures</Link>

   <button onClick={handleDelete}> Delete</button>
 </div>
)}

</div>
);
}
```

pages/Courses.jsx file:-

```
import { useEffect, useState } from "react";
```

```
import { Link } from "react-router-dom";
import API from "../api/axios";
export default function Courses() {
const [courses, setCourses] = useState([]);
const [user, setUser] = useState(null);
useEffect(() => {
  const storedUser = localStorage.getItem("user");
  if (storedUser) {
    const parsed = JSON.parse(storedUser);
    setUser(parsed);
    if (parsed.role === "student") {
      API.get("/courses/my/enrolled").then((res) => setCourses(res.data));
    } else {
      API.get("/courses").then((res) => setCourses(res.data));
    }
  }
}, []);
return (
<div>
<h2>All Courses</h2>
{/* Show Add Course link only if user is admin */}
{user?.role === "admin" && (
<Link to="/add-course">➕ Add Course</Link>
)}
<ul>
{courses.map((c) => (
<li key={c._id}>
```

```
<Link to={`/courses/${c._id}`}>{c.title}</Link> – {c.createdBy?.name}
</li>
))}
</ul>
</div>
);
}
```

pages/EnrolledCoures.jsx :-

```
import React, { useEffect, useState } from 'react';

function EnrolledCourses() {
  const [courses, setCourses] = useState([]);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState(null);

  const token = localStorage.getItem('token');

  useEffect(() => {
    const fetchEnrolledCourses = async () => {
      try {
        const res = await fetch('http://localhost:5000/api/courses/my/enrolled', {
          headers: {
```

```
        Authorization: `Bearer ${token}`,
      },
    });

    if (!res.ok) {
      const errData = await res.json();
      throw new Error(errData.message || 'Failed to fetch enrolled courses');
    }

    const data = await res.json();
    setCourses(data);
  } catch (err) {
    console.error('Error loading enrolled courses:', err);
    setError(err.message);
  } finally {
    setLoading(false);
  }
};

fetchEnrolledCourses();
}, [token]);

return (
  <div style={{ padding: '2rem' }}>
    <h2>My Enrolled Courses</h2>

    {loading ? (
```

```
      <p>Loading...</p>
    ) : error ? (
      <p style={{ color: 'red' }}>{error}</p>
    ) : courses.length === 0 ? (
      <p>You have not enrolled in any courses yet.</p>
    ) : (
      <ul>
        {courses.map((course) => (
          <li key={course._id} style={{ marginBottom: '1rem' }}>
            <h3>{course.title}</h3>
            <p>{course.description}</p>
          </li>
        ))}
      </ul>
    )}
  </div>
 );
}

export default EnrolledCourses;
```

pages/Lectures.jsx file:-

```
import React, { useEffect, useState } from 'react';
import { useParams } from 'react-router-dom';

function LecturesPage() {
  const { courseId } = useParams();
  const [lectures, setLectures] = useState([]);
  const [course, setCourse] = useState(null); // NEW: course info
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState(null);

  // Form state
  const [title, setTitle] = useState('');
  const [content, setContent] = useState('');
  const [videoUrl, setVideoUrl] = useState('');
  const [editingId, setEditingId] = useState(null);

  const token = localStorage.getItem('token');

  // Fetch course and lectures
  useEffect(() => {
    const fetchCourseAndLectures = async () => {
      try {
        // Fetch course
        const courseRes = await fetch(`http://localhost:5000/api/courses/${courseId}`);
        if (!courseRes.ok) throw new Error('Failed to load course');
        const courseData = await courseRes.json();
        setCourse(courseData);
```

```
      // Fetch lectures
      const lectureRes = await fetch(`http://localhost:5000/api/lectures/${courseId}`, {
        headers: { Authorization: `Bearer ${token}` },
      });
      if (!lectureRes.ok) throw new Error('Failed to load lectures');
      const lecturesData = await lectureRes.json();
      setLectures(lecturesData);
    } catch (err) {
      setError(err.message);
    } finally {
      setLoading(false);
    }
  };

  fetchCourseAndLectures();
}, [courseId, token]);

const handleSubmit = async (e) => {
  e.preventDefault();
  const endpoint = editingId
    ? `http://localhost:5000/api/lectures/${editingId}`
    : `http://localhost:5000/api/lectures`;

  const method = editingId ? 'PUT' : 'POST';

  try {
```

```
  const res = await fetch(endpoint, {
    method,
    headers: {
      'Content-Type': 'application/json',
      Authorization: `Bearer ${token}`,
    },
    body: JSON.stringify({ title, content, videoUrl, courseId }),
  });

  if (!res.ok) throw new Error('Failed to save lecture');
  const data = await res.json();

  if (editingId) {
    setLectures((prev) =>
      prev.map((lec) => (lec._id === editingId ? data : lec))
    );
  } else {
    setLectures((prev) => [...prev, data]);
  }

  setTitle('');
  setContent('');
  setVideoUrl('');
  setEditingId(null);
} catch (err) {
  alert(err.message);
}
```

```
};

const handleDelete = async (id) => {
  if (!window.confirm('Are you sure you want to delete this lecture?')) return;

  try {
    const res = await fetch(`http://localhost:5000/api/lectures/${id}`, {
      method: 'DELETE',
      headers: { Authorization: `Bearer ${token}` },
    });
    if (!res.ok) throw new Error('Failed to delete lecture');
    setLectures((prev) => prev.filter((l) => l._id !== id));
  } catch (err) {
    alert(err.message);
  }
};

const handleEdit = (lecture) => {
  setTitle(lecture.title);
  setContent(lecture.content);
  setVideoUrl(lecture.videoUrl || '');
  setEditingId(lecture._id);
};

return (
  <div style={{ padding: '2rem' }}>
    {/* COURSE NAME */}
```

```
{course ? <h2>Lectures for: {course.title}</h2> : <h2>Lectures</h2>}


{loading ? (
  <p>Loading...</p>
) : error ? (
  <p style={{ color: 'red' }}>{error}</p>
) : lectures.length === 0 ? (
  <p>No lectures found.</p>
) : (
  <ul>
    {lectures.map((lecture) => (
      <li key={lecture._id} style={{ marginBottom: '1rem' }}>
        <h3>{lecture.title}</h3>
        <p>{lecture.content}</p>
        {lecture.videoUrl && (
          <video
            src={lecture.videoUrl}
            controls
            style={{ width: '100%', maxWidth: '500px' }}
          />
        )}
        <br />
        <button onClick={() => handleEdit(lecture)}>Edit</button>
        <button
          onClick={() => handleDelete(lecture._id)}
          style={{ marginLeft: '0.5rem', color: 'red' }}
        >
```

```
          Delete
        </button>
      </li>
    ))}
  </ul>
)}

<hr />
<h3>{editingId ? 'Edit Lecture' : 'Add New Lecture'}</h3>
<form onSubmit={handleSubmit}>
  <input
    type="text"
    placeholder="Title"
    value={title}
    onChange={(e) => setTitle(e.target.value)}
    required
  />
  <br />
  <textarea
    placeholder="Content"
    value={content}
    onChange={(e) => setContent(e.target.value)}
    required
  />
  <br />
  <input
    type="text"
```

```
          placeholder="Video URL"
          value={videoUrl}
          onChange={(e) => setVideoUrl(e.target.value)}
        />
        <br />
        <button type="submit">{editingId ? 'Update' : 'Add'} Lecture</button>
        {editingId && (
          <button
            type="button"
            onClick={() => {
              setEditingId(null);
              setTitle('');
              setContent('');
              setVideoUrl('');
            }}
            style={{ marginLeft: '0.5rem' }}
          >
            Cancel
          </button>
        )}
      </form>
    </div>
  );
}

export default LecturesPage;
```

pages/login.jsx file code:-

```
import { useState } from "react";
import { useNavigate } from "react-router-dom";
import API from "../api/axios";
export default function Login() {
const [form, setForm] = useState({ email: "", password: "" });
const [message, setMessage] = useState("");
const navigate = useNavigate();
const handleChange = (e) => setForm({ ...form, [e.target.name]: e.target.value
});
const handleSubmit = async (e) => {
e.preventDefault();
try {
const res = await API.post("/auth/login", form);
localStorage.setItem("token", res.data.token);
localStorage.setItem("user", JSON.stringify(res.data.user));
navigate("/courses");
} catch (err) {
setMessage("⚠ " + err.response?.data?.message || "Error");
}
};
return (
<div>
<h2>Login</h2>
```

```
<form onSubmit={handleSubmit}>
<input name="email" placeholder="Email" onChange={handleChange} />
<input name="password" type="password" placeholder="Password"
onChange={handleChange} />
<button type="submit">Login</button>
</form>
<p>{message}</p>
</div>
);
}
```

pages/Register.jsx file code:-

```
import { useState } from "react";
import API from "../api/axios";
export default function Register() {
const [form, setForm] = useState({ name: "", email: "", password: "", role:
"student" });
const [message, setMessage] = useState("");
const handleChange = (e) => setForm({ ...form, [e.target.name]: e.target.value
});
const handleSubmit = async (e) => {
e.preventDefault();
try {
await API.post("/auth/register", form);
setMessage(" Registered successfully. Please login.");
```

```
} catch (err) {
setMessage("⚠ " + err.response?.data?.message || "Error");
}
};
return (
<div>
<h2>Register</h2>
<form onSubmit={handleSubmit}>
<input name="name" placeholder="Name" onChange={handleChange} />
<input name="email" placeholder="Email" onChange={handleChange} />
<input name="password" type="password" placeholder="Password"
onChange={handleChange} />
<select name="role" onChange={handleChange}>
<option value="student">Student</option>
<option value="admin">Admin</option>
</select>
<button type="submit">Register</button>
</form>
<p>{message}</p>
</div>
);
}
```

pages/UpdateCourse.jsx file code:-

```
import { useEffect, useState } from "react";
import { useParams, useNavigate } from "react-router-dom";
import API from "../api/axios";
export default function UpdateCourse() {
const { id } = useParams();
const [form, setForm] = useState({ title: "", description: "", image: "" });
const navigate = useNavigate();
useEffect(() => {
API.get(`/courses/${id}`).then((res) => {
setForm({
title: res.data.title,
description: res.data.description,
image: res.data.image,
});
});
}, [id]);
const handleChange = (e) => setForm({ ...form, [e.target.name]: e.target.value
});
const handleSubmit = async (e) => {
e.preventDefault();
try {
await API.put(`/courses/${id}`, form);
navigate(`/courses/${id}`);
} catch (err) {
alert("⚠ Only admin can update courses");
}
```

```
};
return (
<div>
<h2>Edit Course</h2>
<form onSubmit={handleSubmit}>
<input name="title" value={form.title} onChange={handleChange} />
<textarea name="description" value={form.description}
onChange={handleChange}></textarea>
<input name="image" value={form.image} onChange={handleChange} />
<button type="submit">Update</button>
</form>
</div>
);
}
```

App.jsx file code:-

```
import { BrowserRouter as Router, Routes, Route, Link } from "react-router-dom";

import Register from "./pages/Register";
import Login from "./pages/Login";
import Courses from "./pages/Courses";
import Lectures from './pages/Lectures';
 // or wherever your lectures page/component is
import CourseDetail from "./pages/CourseDetail";
import AddCourse from "./pages/AddCourse";
import UpdateCourse from "./pages/UpdateCourse";
import ProtectedRoute from "./components/ProtectedRoute";
import Navbar from "./components/Navbar";
import EnrolledCourses from './pages/EnrolledCourses';
import AssignCourse from './pages/AssignCourse';


function App() {
return (
<Router>
<Navbar /> {/*  Always visible */}
<Routes>
{/* Public */}
<Route path="/" element={<h2>Welcome to Course Platform </h2>} />
<Route path="/register" element={<Register />} />
<Route path="/login" element={<Login />} />
<Route path="/courses" element={<Courses />} />
<Route path="/course/:courseId/lectures" element={<Lectures />} />
```

```
<Route path="/my-courses" element={<EnrolledCourses />} />

<Route path="/assign-course/:courseId" element={<AssignCourse />} />



<Route path="/courses/:id" element={<CourseDetail />} />

{/* Admin-only */}

<Route

path="/add-course"

element={

<ProtectedRoute role="admin">

<AddCourse />



</ProtectedRoute>

}

/>

<Route

path="/update-course/:id"

element={

<ProtectedRoute role="admin">

<UpdateCourse />

</ProtectedRoute>

}

/>

</Routes>

</Router>

);
```

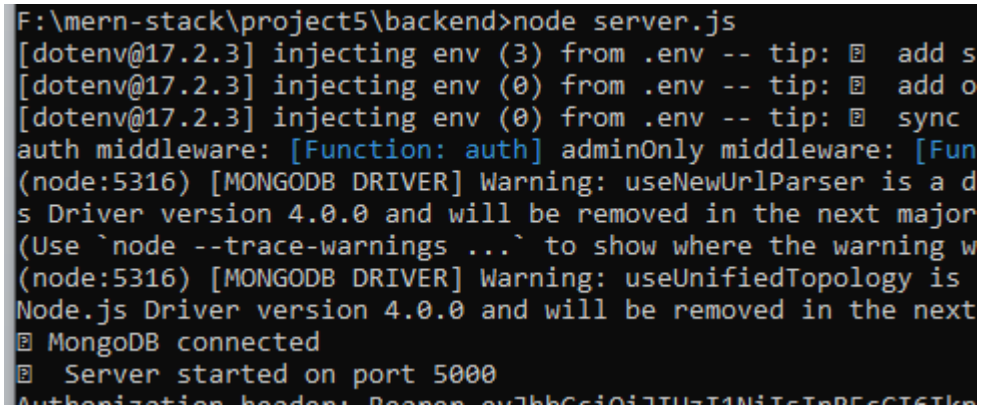
}

export default App;

Now run backend :-

cd backend

node server.js



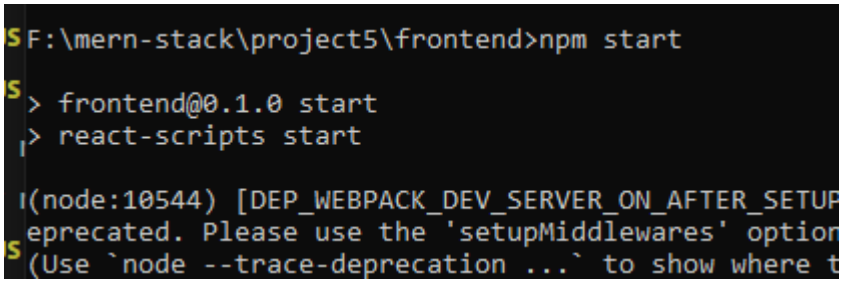
Open browser and visit :-

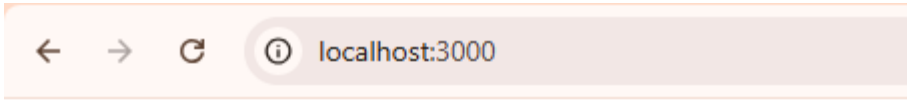
http://localhost:5000/api/courses/

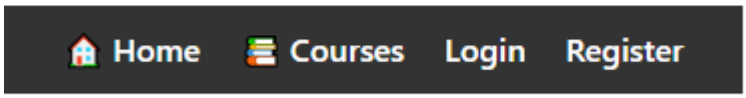
and then run frontend:-

cd frontend

npm start

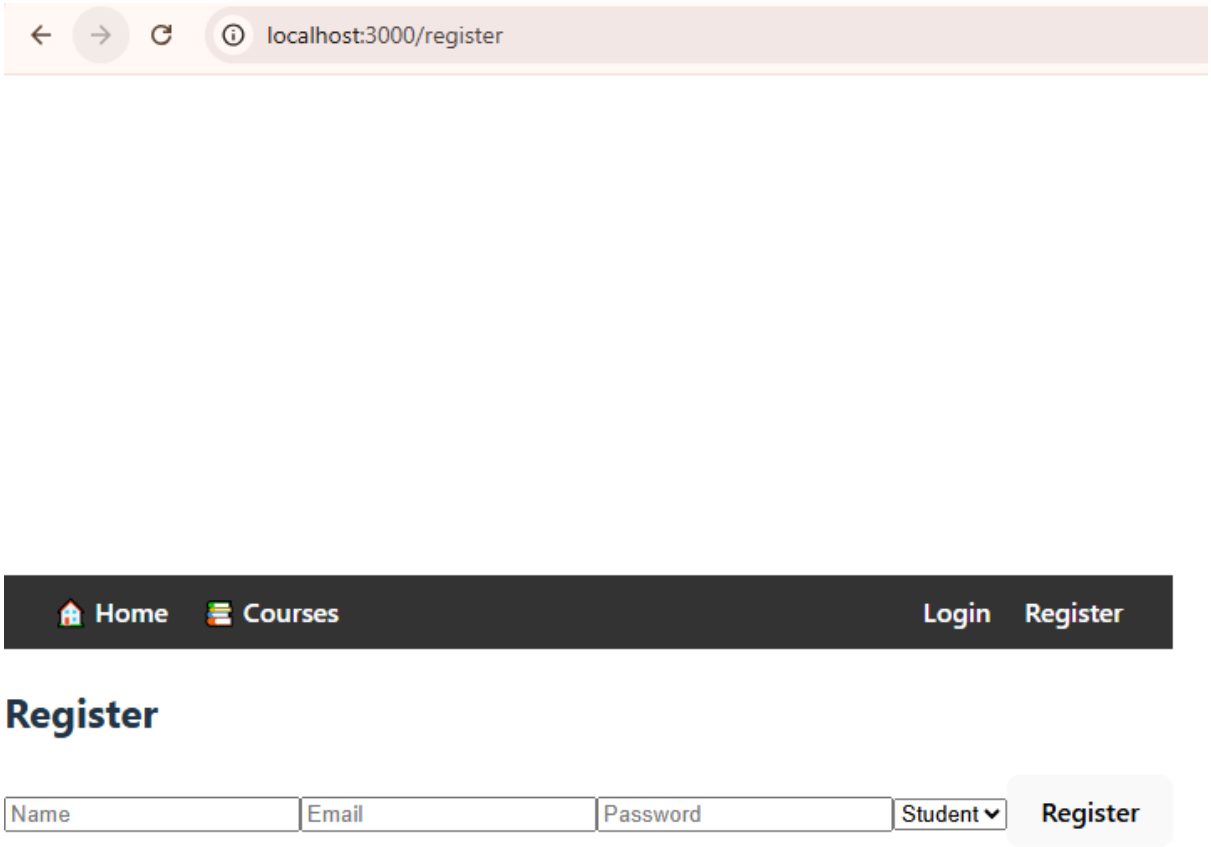


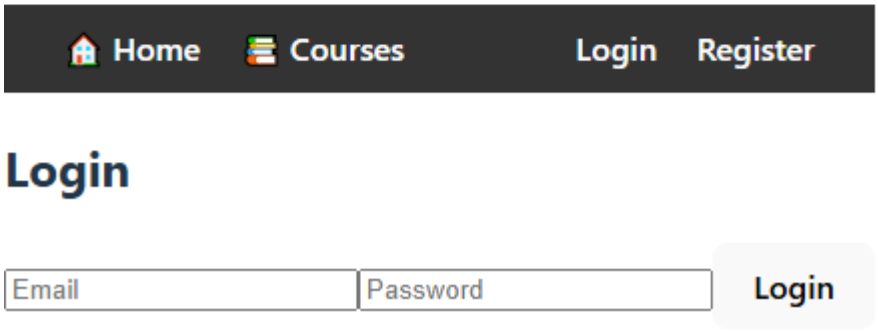




Welcome to Course Platform 🚀

Register user:-





After admin login:-

Home Courses Login Register

All Courses

Add Course

- pyhton – om
- react js – om