BACKEND :-

mkdir Project4

cd Project4

mkdir backend

npm init -y

npm install express mongoose cors

npm install jsonwebtoken

npm install dotenv

npm install bcryptjs

backend/index.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 app = express();
app.use(express.json());
app.use(cors());
app.use('/api/auth', authRoutes);
app.use('/api/courses', courseRoutes);
const PORT = process.env.PORT || 5000;
const MONGO = process.env.MONGO_URI;
mongoose.connect(MONGO, {
 useNewUrlParser: true,
 useUnifiedTopology: true,
}).then(() => {
 console.log('Mongo connected');
 app.listen(PORT, () => console.log('Server started on', PORT));
}).catch(err => console.error(err));
```

backend/.env

```
PORT=5000

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

backend/middleware/auth.js

```
const jwt = require('jsonwebtoken');
require('dotenv').config();
const auth = (req, res, next) => {
 const header = req.headers.authorization;
 if (!header) return res.status(401).json({ message: 'No token' });
 const token = header.split(' ')[1];
 if (!token) return res.status(401).json({ message: 'No token' });
 try {
 const decoded = jwt.verify(token, process.env.JWT_SECRET);
 req.user = decoded; // contains id and role
 next();
 } catch (err) {
 return res.status(401).json({ message: 'Invalid token' });
 }
};
const adminOnly = (req, res, next) => {
 if (req.user && req.user.role === 'admin') return next();
 return res.status(403).json({ message: 'Admin only' });
};
module.exports = { auth, adminOnly };
```

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);
```

backend/models/Lecture.js

```
const mongoose = require('mongoose');
const lectureSchema = new mongoose.Schema({
 course: { type: mongoose.Schema.Types.ObjectId, ref: 'Course', required: true },
 title: { type: String, required: true },
 content: String,
 videoUrl: String
}, { timestamps: true });
module.exports = mongoose.model('Lecture', lectureSchema);
```

backend/models/User.js

```
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' }
},
 { timestamps: true });
module.exports = mongoose.model('User', userSchema);
```

backend/routes/auth.js

```
// routes/auth.js
const express = require('express');
const router = express.Router();
const bcrypt = require('bcryptjs');
const jwt = require('jsonwebtoken');
const User = require('../models/User');
require('dotenv').config();
```

```
// Register
router.post('/register', async (req, res) => {
  const { name, email, password, role } = 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) {
    res.status(500).json({ message: 'Server error' });
  }
});

// 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' });
  }
});

module.exports = router;
```

backend/routes/courses.js

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

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

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

// 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) {
```

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

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

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

module.exports = router;
```

Frontend Notes

npx create-react-app frontend

cd frontend

npm install axios

npm install react-router-dom

frontend/src/api/axios.js

```
import axios from "axios";

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.Authorization = `Bearer ${token}`;
  }
  return req;
});

export default API;
```

frontend/components/Navbar.js

```
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",
  },
};
```

frontend/components/ProtectedRoute.js

```
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;
}
```

frontend/src/pages/AddCourse.js

```
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>
  );
}
```

frontend/src/pages/CourseDetail.js

```
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={`/update-course/${course._id}`}>✏️ Edit</Link>
          <button onClick={handleDelete}>🗑️ Delete</button>
        </div>
      )}
    </div>
  );
}
```

frontend/src/pages/Courses.js

```
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(() => {
    // Fetch courses
    API.get("/courses").then((res) => setCourses(res.data));

    // Get logged in user from localStorage
    const storedUser = localStorage.getItem("user");
    if (storedUser) setUser(JSON.parse(storedUser));
  }, []);

  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>
  );
}
```

frontend/src/pages/Login.js

```
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>
  );
}
```

frontend/src/pages/Register.js

```
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>
  );
}
```

frontend/src/pages/UpdateCourse.js

```
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>
  );
}
```

frontend/app.js

```
import { BrowserRouter as Router, Routes, Route } from "react-router-dom";
import Register from "./pages/Register";
import Login from "./pages/Login";
import Courses from "./pages/Courses";
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";

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="/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;
```