Great setup! You've already implemented a full **user authentication** system, **course CRUD operations**, and **role-based access** in your MERN app. Now, let's extend your backend and frontend with the following:

□ New Features You Requested

1. **Lecture CRUD** (Admin-only):
   - Add lectures to a course (e.g., lessons or videos).
   - View, edit, delete lectures for each course.
2. **Admin Assign Courses to Students**:
   - Admin can assign students to specific courses.
   - Students can view only their assigned courses.

□ 1. BACKEND UPDATES

□ A. Update `User` model to track enrolled courses

□ backend/models/User.js

```
// Add this inside the userSchema
enrolledCourses: [{ type: mongoose.Schema.Types.ObjectId, ref: 'Course' }]
```

So final model looks like:

```
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 });
```

□ B. Add Lecture CRUD routes

□ backend/routes/lectures.js

```
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;
```

□ C. Add route to assign courses to students

□ Add to `backend/routes/courses.js`

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

// Admin assigns course to student
```

```
router.post('/:id/assign', auth, adminOnly, async (req, res) => {
  const { studentId } = req.body;
  const courseId = req.params.id;

  try {
    const student = await User.findById(studentId);
    if (!student || student.role !== 'student') {
      return res.status(400).json({ message: 'Invalid student' });
    }

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

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

□ D. Route: Get courses for the logged-in student

□ Add to `backend/routes/courses.js`

```
// Get logged-in student's courses
router.get('/my/enrolled', auth, async (req, res) => {
  try {
    const user = await
User.findById(req.user.id).populate('enrolledCourses');
    if (!user || user.role !== 'student') {
      return res.status(403).json({ message: 'Access denied' });
    }
    res.json(user.enrolledCourses);
  } catch (err) {
    res.status(500).json({ message: 'Server error' });
  }
});
```

□ E. Register the new `lectures.js` route

□ `backend/index.js`

```
const lectureRoutes = require('./routes/lectures');
app.use('/api/lectures', lectureRoutes);
```

□ 2. FRONTEND UPDATES

□ A. Create Lecture Management Components (Admin)

You can make:

- `/course/:id/lectures` page for listing/adding lectures.
- Only admins can see and modify.
- Reuse your AddCourse style.

□ B. Add Course Assignment Page (Admin)

New route idea: `/assign-course/:courseId`

You'll fetch a list of students and allow selecting one to assign:

```
// 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)); // You'll
create this route below
  }, []);

  const assign = async () => {
    try {
      await API.post(`/courses/${courseId}/assign`, { studentId: selected });
      alert("□ Assigned");
    } catch {
      alert("□ Error");
    }
  };

  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>
  );
}
```

□ C. API: Get all students

□ Add to `routes/auth.js`

```
const { auth, adminOnly } = require('../middleware/auth');

router.get('/students', auth, adminOnly, async (req, res) => {
  const students = await User.find({ role: 'student' }, 'name email');
  res.json(students);
});
```

□ D. Update `Courses.js` for students to view only enrolled ones

```
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));
    }
  }
}, []);
```

□ DONE! Summary of What You Have

Feature	Status
User Auth + Roles	□ Implemented
Course CRUD	□ Implemented
Lecture CRUD	□ Added
Assign Courses to Students (admin)	□ Added
Students see only their courses	□ Added
Secure API (auth + role checks)	□ Done

□ Next Steps

If you'd like, I can help you:

- Add **Lecture list** and **Add Lecture form** to frontend.
- Build a **student dashboard** showing enrolled courses and lectures.

- Add **video player** for lectures.
- Add **quizzes** or **progress tracking**.

Just let me know what's next!

You said:

yes give this section also A. Create Lecture Management Components (Admin) You can make: /course/:id/lectures page for listing/adding lectures. Only admins can see and modify. Reuse your AddCourse style.

ChatGPT said:

Perfect! Let's implement the **Lecture Management** for Admins:

You'll create a **`CourseLectures.js`** page that allows Admins to:

- □ View all lectures in a course
- □ Add a new lecture
- □ Delete existing lectures

□ Lecture Management Page (`/course/:id/lectures`)

□ File: `frontend/src/pages/CourseLectures.js`

```
import { useEffect, useState } from "react";
import { useParams } from "react-router-dom";
import API from "../api/axios";

export default function CourseLectures() {
  const { id: courseId } = useParams();
  const [lectures, setLectures] = useState([]);
  const [form, setForm] = useState({ title: "", content: "", videoUrl: "" });

  // Fetch existing lectures
  useEffect(() => {
    API.get(`/lectures/${courseId}`).then((res) => setLectures(res.data));
  }, [courseId]);

  // Handle input change
  const handleChange = (e) => {
    setForm({ ...form, [e.target.name]: e.target.value });
  };

  // Add a new lecture
  const handleSubmit = async (e) => {
    e.preventDefault();
    try {
```

```
    const data = { ...form, courseId };
    const res = await API.post("/lectures", data);
    setLectures([...lectures, res.data]);
    setForm({ title: "", content: "", videoUrl: "" });
  } catch (err) {
    alert("□ Failed to add lecture");
  }
};

// Delete lecture
const handleDelete = async (lectureId) => {
  if (!window.confirm("Are you sure?")) return;
  try {
    await API.delete(`/lectures/${lectureId}`);
    setLectures(lectures.filter((lec) => lec._id !== lectureId));
  } catch {
    alert("□ Failed to delete");
  }
};

return (
  <div>
    <h2>□ Manage Lectures</h2>

    {/* Add Lecture Form */}
    <form onSubmit={handleSubmit} style={styles.form}>
      <input
        name="title"
        placeholder="Lecture Title"
        value={form.title}
        onChange={handleChange}
      />
      <textarea
        name="content"
        placeholder="Lecture Content"
        value={form.content}
        onChange={handleChange}
      ></textarea>
      <input
        name="videoUrl"
        placeholder="Video URL (optional)"
        value={form.videoUrl}
        onChange={handleChange}
      />
      <button type="submit">□ Add Lecture</button>
    </form>

    {/* Lecture List */}
    <h3>□ All Lectures</h3>
    <ul>
      {lectures.map((lec) => (
        <li key={lec._id}>
          <strong>{lec.title}</strong>
          <p>{lec.content}</p>
          {lec.videoUrl && (
            <p>
```

```
                 <a href={lec.videoUrl} target="_blank"
rel="noreferrer">Watch Video</a>
               </p>
             )}
             <button onClick={() => handleDelete(lec._id)}> Delete</button>
           </li>
         ))}
       </ul>
     </div>
  );
}

const styles = {
  form: {
    display: "flex",
    flexDirection: "column",
    maxWidth: "400px",
    gap: "10px",
    marginBottom: "20px",
  },
};
```

Route Setup

In `App.js`

Import and protect the route for admins:

```
import CourseLectures from "./pages/CourseLectures";

// Inside <Routes>
<Route
  path="/course/:id/lectures"
  element={
    <ProtectedRoute role="admin">
      <CourseLectures />
    </ProtectedRoute>
  }
/>
```

Navigation (Optional)

You can add a link in `CourseDetail.js` if the user is an admin:

```
{user?.role === "admin" && (
  <div>
    <Link to={`/update-course/${course._id}`}>✏ Edit</Link>
    <Link to={`/course/${course._id}/lectures`}> Manage Lectures</Link>
    <button onClick={handleDelete}> Delete</button>
  </div>
)}
```

□ Testing

1. □ Login as admin.
2. □ Go to a course (e.g., `/courses/123`).
3. □ Click **"□ Manage Lectures"**.
4. □ Add a few lectures.
5. □ Confirm they appear in the list.
6. □ Try deleting one.