Express js Crud for adding courses and lectures with video upload and pdf notes with backend mongodb and front end in ejs

Here's a full **Express.js CRUD application** with:

- **MongoDB (Mongoose)** backend
- **Video & PDF upload**
- **Course and Lecture model**
- **EJS frontend templates**

open your command prompt in windows :-

and create project folder as "course-app "

mkdir course-app

cd course-app

npm init -y

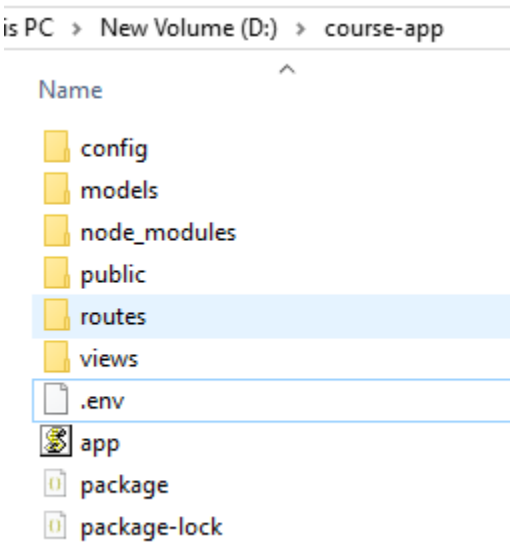
Then Install dependencies

```
npm install express mongoose multer ejs dotenv method-override
```

Project Structure :-

```
course-app/
│
├── public/
│   └── uploads/            # Stores videos & PDFs
│
├── views/
│   ├── courses/
│   │   ├── index.ejs
│   │   ├── new.ejs
│   │   └── show.ejs
```

```
│   ├── lectures/
│   │   └── new.ejs
│   └── layout.ejs
│
├── models/
│   ├── Course.js
│   └── Lecture.js
│
├── routes/
│   ├── courseRoutes.js
│   └── lectureRoutes.js
│
├── .env
├── app.js
└── package.json
```



.env file :-

MONGO_URI=mongodb://localhost:27017/courseApp

1. □ Models

`models/Course.js:-`

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

```
const CourseSchema = new mongoose.Schema({
  title: String,
  description: String,
});

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

models/Lecture.js:-

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



const LectureSchema = new mongoose.Schema({

  courseId: { type: mongoose.Schema.Types.ObjectId, ref: 'Course'
},

  title: String,

  videoUrl: String,

  notesUrl: String,

});



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

app.js file code:-

```
// app.js

const express = require('express');

const mongoose = require('mongoose');

const methodOverride = require('method-override');

const path = require('path');

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

```
const expressLayouts = require('express-ejs-layouts');

const app = express();


// Load .env variables

dotenv.config();


// Connect to MongoDB

mongoose.connect(process.env.MONGO_URI ||
'mongodb://localhost:27017/courseApp', {

  useNewUrlParser: true,

  useUnifiedTopology: true,

})

.then(() => console.log('□ MongoDB Connected'))

.catch((err) => console.error('MongoDB Connection Error:', err));


// Set view engine

app.set('view engine', 'ejs');




// Middlewares

app.use(express.urlencoded({ extended: true })); // Parse form data

app.use(methodOverride('_method'));              // Support
PUT/DELETE in forms

app.use(express.static(path.join(__dirname, 'public'))); // Serve
static files (like uploads, CSS)
```

```
// Routes

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

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



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

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



// Home route (redirect to courses)

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

  res.redirect('/courses');

});



// Start server

const PORT = process.env.PORT || 3000;

app.listen(PORT, () => {

  console.log(`□ Server running on http://localhost:${PORT}`);

});
```

3. □ File Upload Setup

config/multer.js :-

```
// config/multer.js
const multer = require('multer');
const path = require('path');

const storage = multer.diskStorage({
  destination: function (req, file, cb) {
    cb(null, 'public/uploads/');
  },
  filename: function (req, file, cb) {
```

```
    cb(null, Date.now() + path.extname(file.originalname));
  }
});

module.exports = multer({ storage: storage });
```

4. □ Routes

routes/courseRoutes.js

```
const express = require('express');
const router = express.Router();
const Course = require('../models/Course');
const Lecture = require('../models/Lecture');

// List all courses
router.get('/', async (req, res) => {
  const courses = await Course.find();
  res.render('courses/index', { courses });
});

// New Course Form
router.get('/new', (req, res) => {
  res.render('courses/new');
});
```

```
// Create a new course
router.post('/', async (req, res) => {
  await Course.create(req.body);
  res.redirect('/courses');
});

// Edit Course Form (must come BEFORE '/:id')
router.get('/:id/edit', async (req, res) => {
  try {
    const course = await Course.findById(req.params.id);
    if (!course) return res.status(404).send('Course not found');
    res.render('courses/edit', { course });
  } catch (err) {
    console.error(err);
    res.status(500).send('Server Error');
  }
});

// Update Course
router.put('/:id', async (req, res) => {
  try {
    await Course.findByIdAndUpdate(req.params.id, req.body);
    res.redirect(`/courses/${req.params.id}`);
```

```
  } catch (err) {
    console.error(err);
    res.status(500).send('Update failed');
  }
});

// Delete Course (and its lectures)
router.delete('/:id', async (req, res) => {
  try {
    const courseId = req.params.id;
    await Lecture.deleteMany({ courseId }); // delete associated
lectures
    await Course.findByIdAndDelete(courseId);
    res.redirect('/courses');
  } catch (err) {
    console.error(err);
    res.status(500).send('Delete failed');
  }
});

// Show course and its lectures
router.get('/:id', async (req, res) => {
  try {
    const course = await Course.findById(req.params.id);
    if (!course) return res.status(404).send('Course not found');
```

```
    const lectures = await Lecture.find({ courseId: course._id });
    res.render('courses/show', { course, lectures });
  } catch (err) {
    console.error(err);
    res.status(500).send('Error loading course');
  }
});

module.exports = router;
```

routes/lectureRoutes.js file:-

```
const express = require('express');
const router = express.Router();
const upload = require('../config/multer');
const Lecture = require('../models/Lecture');
const fs = require('fs');
const path = require('path');

// Show New Lecture Form
router.get('/new/:courseId', (req, res) => {
  res.render('lectures/new', { courseId: req.params.courseId });
});
```

```
// Create Lecture
router.post('/', upload.fields([{ name: 'video' }, { name: 'notes' }]), async (req, res) => {
  const { title, courseId } = req.body;
  const video = req.files['video']?.[0]?.filename || '';
  const notes = req.files['notes']?.[0]?.filename || '';

  await Lecture.create({
    title,
    courseId,
    videoUrl: '/uploads/' + video,
    notesUrl: '/uploads/' + notes,
  });

  res.redirect(`/courses/${courseId}`);
});

// Edit Lecture Form
router.get('/:id/edit', async (req, res) => {
  const lecture = await Lecture.findById(req.params.id);
  if (!lecture) return res.status(404).send('Lecture not found');
  res.render('lectures/edit', { lecture });
});

// Update Lecture
```

```
router.put('/:id', upload.fields([{ name: 'video' }, { name:
'notes' }]), async (req, res) => {

  const lecture = await Lecture.findById(req.params.id);

  if (!lecture) return res.status(404).send('Lecture not found');



  const { title } = req.body;



  if (req.files['video']) {

    if (lecture.videoUrl) fs.unlinkSync('public' +
lecture.videoUrl);

    lecture.videoUrl = '/uploads/' +
req.files['video'][0].filename;

  }



  if (req.files['notes']) {

    if (lecture.notesUrl) fs.unlinkSync('public' +
lecture.notesUrl);

    lecture.notesUrl = '/uploads/' +
req.files['notes'][0].filename;

  }



  lecture.title = title;

  await lecture.save();



  res.redirect(`/courses/${lecture.courseId}`);

});
```

```
// Delete Lecture
router.delete('/:id', async (req, res) => {
  const lecture = await Lecture.findById(req.params.id);
  if (!lecture) return res.status(404).send('Lecture not found');

  if (lecture.videoUrl) fs.unlinkSync('public' + lecture.videoUrl);
  if (lecture.notesUrl) fs.unlinkSync('public' + lecture.notesUrl);

  const courseId = lecture.courseId;
  await Lecture.findByIdAndDelete(req.params.id);

  res.redirect(`/courses/${courseId}`);
});

module.exports = router;
```

5. □ Views

views/courses/index.ejs

```
<h1>Courses</h1>
```

```
<a href="/courses/new" class="btn btn-primary mb-3">Add
Course</a>



<ul class="list-group">

  <% courses.forEach(course => { %>

    <li class="list-group-item d-flex justify-content-between
align-items-center">

      <div>

        <a href="/courses/<%= course._id %>" class="fw-bold
text-decoration-none">

          <%= course.title %>

        </a>

      </div>

      <div>

        <a href="/courses/<%= course._id %>/edit" class="btn
btn-sm btn-warning me-2">Edit</a>



        <form action="/courses/<%= course._id %>?_method=DELETE"
method="POST" style="display:inline;">

          <button class="btn btn-sm btn-danger" onclick="return
confirm('Delete this course?')">Delete</button>

        </form>

      </div>

    </li>

  <% }) %>

</ul>
```

views/courses/new.ejs:-

```
<h1>New Course</h1>
<form action="/courses" method="POST">
  <input class="form-control" name="title" placeholder="Title"><br>
  <textarea class="form-control" name="description"
placeholder="Description"></textarea><br>
  <button class="btn btn-success">Create</button>
</form>
```

views/courses/show.ejs:-

```
<h1>Lectures for: <%= course.title %></h1>

<ul class="list-group">
  <% lectures.forEach(lec => { %>
    <li class="list-group-item">
      <div class="d-flex justify-content-between align-items-
start">
        <div>
          <strong><%= lec.title %></strong><br>

          <% if (lec.videoUrl) { %>
            <video width="320" controls class="mt-2">
              <source src="<%= lec.videoUrl %>" type="video/mp4">
            </video><br>
          <% } %>

          <% if (lec.notesUrl) { %>
            <a href="<%= lec.notesUrl %>" target="_blank">Download
Notes</a><br>
          <% } %>
        </div>

        <div>
          <a href="/lectures/<%= lec._id %>/edit" class="btn btn-sm
btn-warning me-2">Edit</a>

          <form action="/lectures/<%= lec._id %>?_method=DELETE"
method="POST" style="display:inline;">
            <button class="btn btn-sm btn-danger" onclick="return
confirm('Delete this lecture?')">Delete</button>
          </form>
        </div>
      </div>
    </li>
  <% }) %>
</ul>
```

Views/courses/edit.ejs :-

```
<h1>Edit Course</h1>

<form action="/courses/<%= course._id %>?_method=PUT"
method="POST">
  <div class="mb-3">
    <label for="title" class="form-label">Course Title</label>
    <input type="text" class="form-control" name="title" id="title"
value="<%= course.title %>" required>
  </div>

  <div class="mb-3">
    <label for="description" class="form-label">Description</label>
    <textarea class="form-control" name="description"
id="description" rows="4"><%= course.description %></textarea>
  </div>

  <button class="btn btn-success">Update Course</button>
  <a href="/courses/<%= course._id %>" class="btn btn-
secondary">Cancel</a>
</form>
```

views/lectures/new.ejs:-

```
<h1>New Lecture</h1>
<form action="/lectures" method="POST" enctype="multipart/form-
data">
  <input type="hidden" name="courseId" value="<%= courseId %>">
  <input class="form-control" name="title" placeholder="Lecture
Title"><br>
  <label>Video:</label>
  <input class="form-control" type="file" name="video"><br>
  <label>PDF Notes:</label>
  <input class="form-control" type="file" name="notes"><br>
  <button class="btn btn-success">Upload</button>
</form>
```

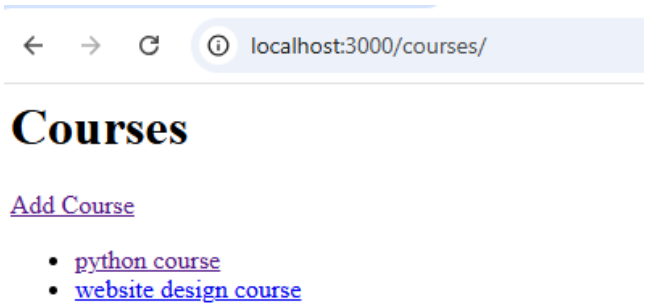
Views/lectures/edit.ejs:-

```
<h1>Edit Lecture</h1>
<form action="/lectures/<%= lecture._id %>?_method=PUT"
method="POST" enctype="multipart/form-data">
  <input class="form-control" name="title" value="<%= lecture.title
%>"><br>

  <label>Replace Video (optional):</label>
```

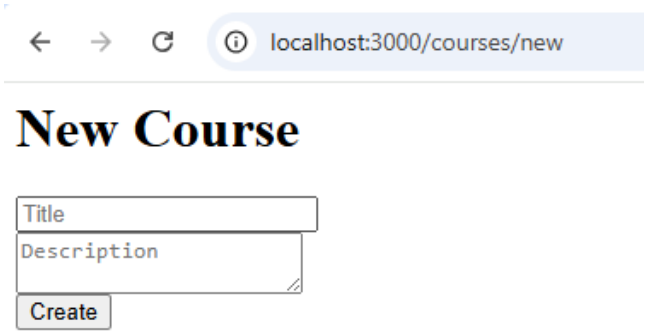
```
  <input class="form-control" type="file" name="video"><br>

  <label>Replace Notes (optional):</label>
  <input class="form-control" type="file" name="notes"><br>

  <button class="btn btn-success">Update Lecture</button>
</form>
```

finally run projects:-

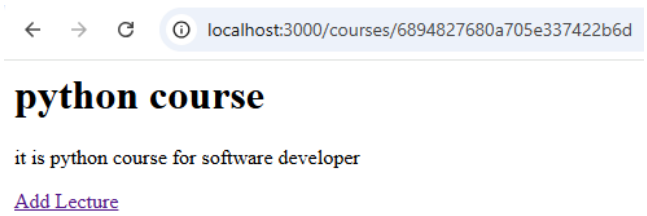
node app.js



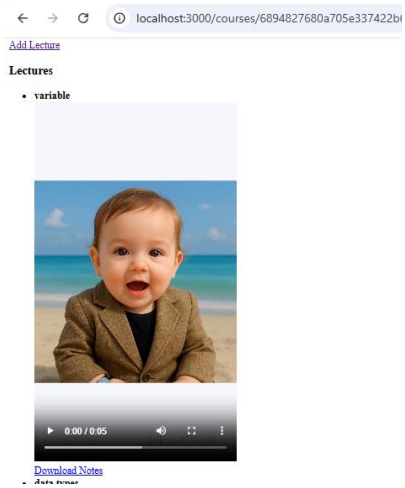
click on add course and add course title and description



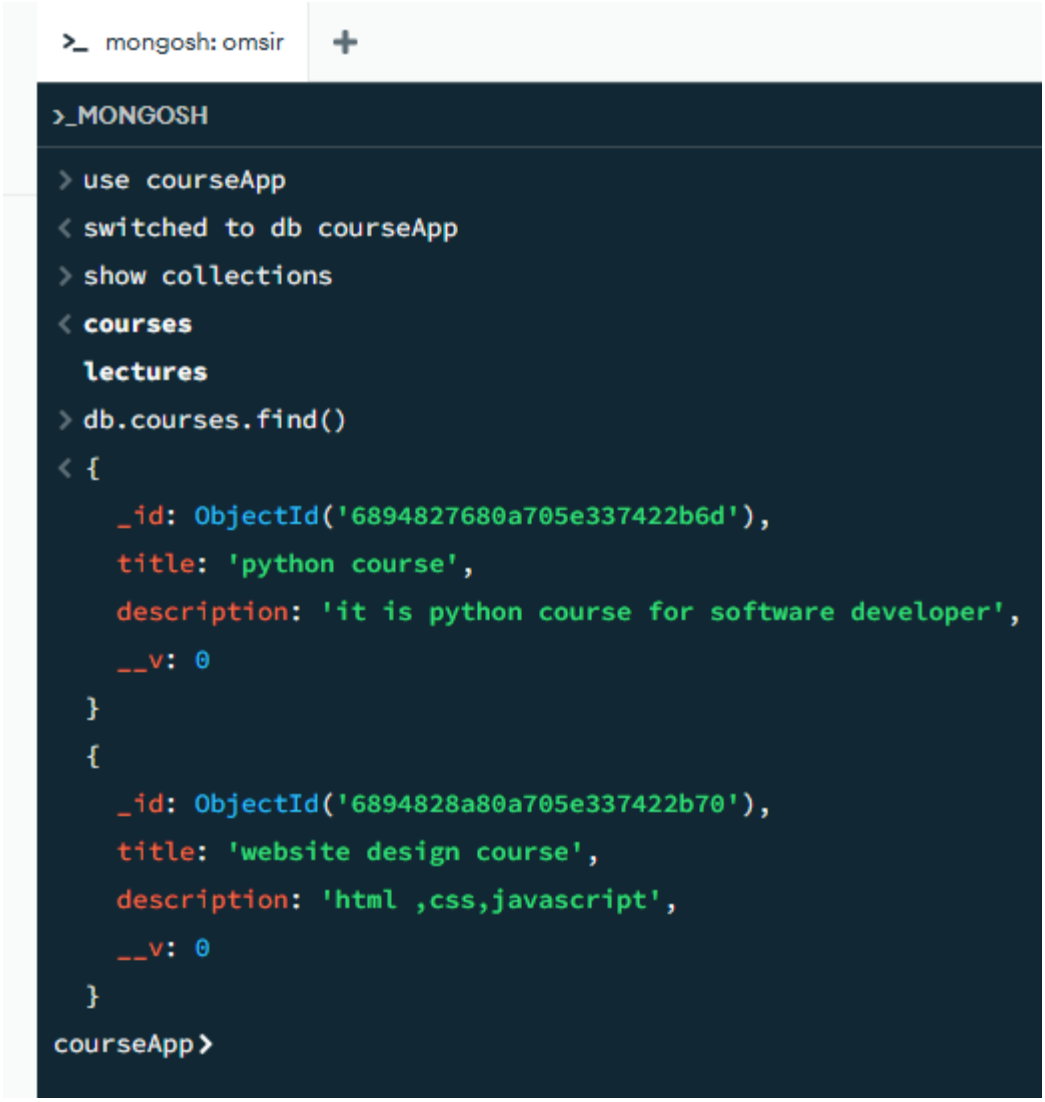
after it add lectures



you will see output after adding lectures :-



now you can check inserted data to your mongodb as shown below in shell query :-


mongosh: omsir
>_MONGOSH
> use courseApp
< switched to db courseApp
> show collections
< courses
lectures
> db.courses.find()
< {
_id: ObjectId('6894827680a705e337422b6d'),
title: 'python course',
description: 'it is python course for software developer',
__v: 0
}
{
_id: ObjectId('6894828a80a705e337422b70'),
title: 'website design course',
description: 'html ,css,javascript',
__v: 0
}
courseApp>