Here's a **complete Express.js CRUD example** using **MongoDB** for managing a `courses` collection with fields:

- `title`
- `description`
- `lectures`
- `fees`
- `notes` (PDF file)

We'll use:

- **Express.js** for server
- **MongoDB + Mongoose** for database
- **EJS** as view engine
- **Multer** for handling PDF uploads

✅ 1. Project Setup

```
mkdir mongo-crud
cd mongo-crud
npm init -y
npm install express mongoose multer ejs body-parser
```

✅ 2. Folder Structure

```
mongo-crud/
│
├── models/
│   └── Course.js
│
├── routes/
│   └── courseRoutes.js
│
├── views/
│   ├── index.ejs
│   ├── new.ejs
│   └── edit.ejs
│
├── uploads/
│   └── (pdf files will be stored here)
│
├── app.js
└── package.json
```

models/Course.js :-

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

const courseSchema = new mongoose.Schema({
   title: String,
   description: String,
   lectures: Number,
   fees: Number,
   notes: String // path to PDF file
});

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

✅ 4. Express App (`app.js`)

```
const express = require('express');
const mongoose = require('mongoose');
const bodyParser = require('body-parser');
const courseRoutes = require('./routes/courseRoutes');
const path = require('path');

const app = express();

// Connect to MongoDB
mongoose.connect('mongodb://localhost:27017/mongo-crud', {
   useNewUrlParser: true,
   useUnifiedTopology: true,
});

// Middleware
```

```
app.set('view engine', 'ejs');
app.use(bodyParser.urlencoded({ extended: false }));
app.use(express.static(path.join(__dirname, 'uploads')));
app.use('/courses', courseRoutes);

// Home
app.get('/', (req, res) => res.redirect('/courses'));

const PORT = 3000;
app.listen(PORT, () => console.log(`Server running on http://localhost:${PORT}`));
```

✅ 5. Routes (`routes/courseRoutes.js`) :-

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

// Multer setup for file uploads
const storage = multer.diskStorage({
  destination: (req, file, cb) => {
    cb(null, 'uploads/');
  },
  filename: (req, file, cb) => {
    cb(null, Date.now() + path.extname(file.originalname)); // Unique filename
  }
});
const upload = multer({ storage });

// Get all courses
```

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

// Show form to create new course
router.get('/new', (req, res) => {
   res.render('new');
});

// Create new course
router.post('/', upload.single('notes'), async (req, res) => {
   const { title, description, lectures, fees } = req.body;
   const notes = req.file ? req.file.filename : null;

   await Course.create({ title, description, lectures, fees, notes });
   res.redirect('/courses');
});

// Show edit form
router.get('/:id/edit', async (req, res) => {
   const course = await Course.findById(req.params.id);
   res.render('edit', { course });
});

// Update course
router.post('/:id', upload.single('notes'), async (req, res) => {
   const { title, description, lectures, fees } = req.body;
   const updateData = { title, description, lectures, fees };

   if (req.file) {
```

```
    updateData.notes = req.file.filename;
  }

  await Course.findByIdAndUpdate(req.params.id, updateData);
  res.redirect('/courses');
});

// Delete course
router.post('/:id/delete', async (req, res) => {
  await Course.findByIdAndDelete(req.params.id);
  res.redirect('/courses');
});

module.exports = router;
```

✅ 6. EJS Views

🔷 `views/index.ejs`

```
<!DOCTYPE html>
<html>
<head><title>Courses</title></head>
<body>
  <h1>All Courses</h1>
  <a href="/courses/new">Add New Course</a>
  <ul>
    <% courses.forEach(course => { %>
      <li>
        <h2><%= course.title %></h2>
        <p><%= course.description %></p>
        <p>Lectures: <%= course.lectures %></p>
```

```
            <p>Fees: $<%= course.fees %></p>
            <% if (course.notes) { %>
              <a href="/<%= course.notes %>" target="_blank">View Notes (PDF)</a>
            <% } %>
            <br>
            <a href="/courses/<%= course._id %>/edit">Edit</a>
            <form method="POST" action="/courses/<%= course._id %>/delete" style="display:inline;">
              <button type="submit">Delete</button>
            </form>
          </li>
        <% }) %>
    </ul>
</body>
</html>
```

◈ views/new.ejs:-

```
<!DOCTYPE html>
<html>
<head><title>New Course</title></head>
<body>
    <h1>Add New Course</h1>
    <form method="POST" action="/courses" enctype="multipart/form-data">
        <input type="text" name="title" placeholder="Title" required><br>
        <textarea name="description" placeholder="Description" required></textarea><br>
        <input type="number" name="lectures" placeholder="Lectures" required><br>
        <input type="number" name="fees" placeholder="Fees" required><br>
        <input type="file" name="notes" accept=".pdf"><br>
        <button type="submit">Create</button>
    </form>
    <a href="/courses">Back</a>
```

```
</body>
</html>
```

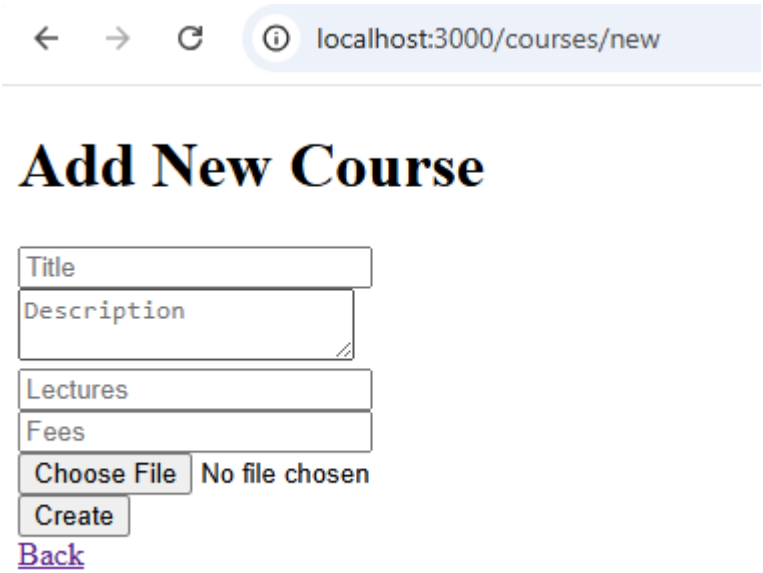
views/edit.ejs:-

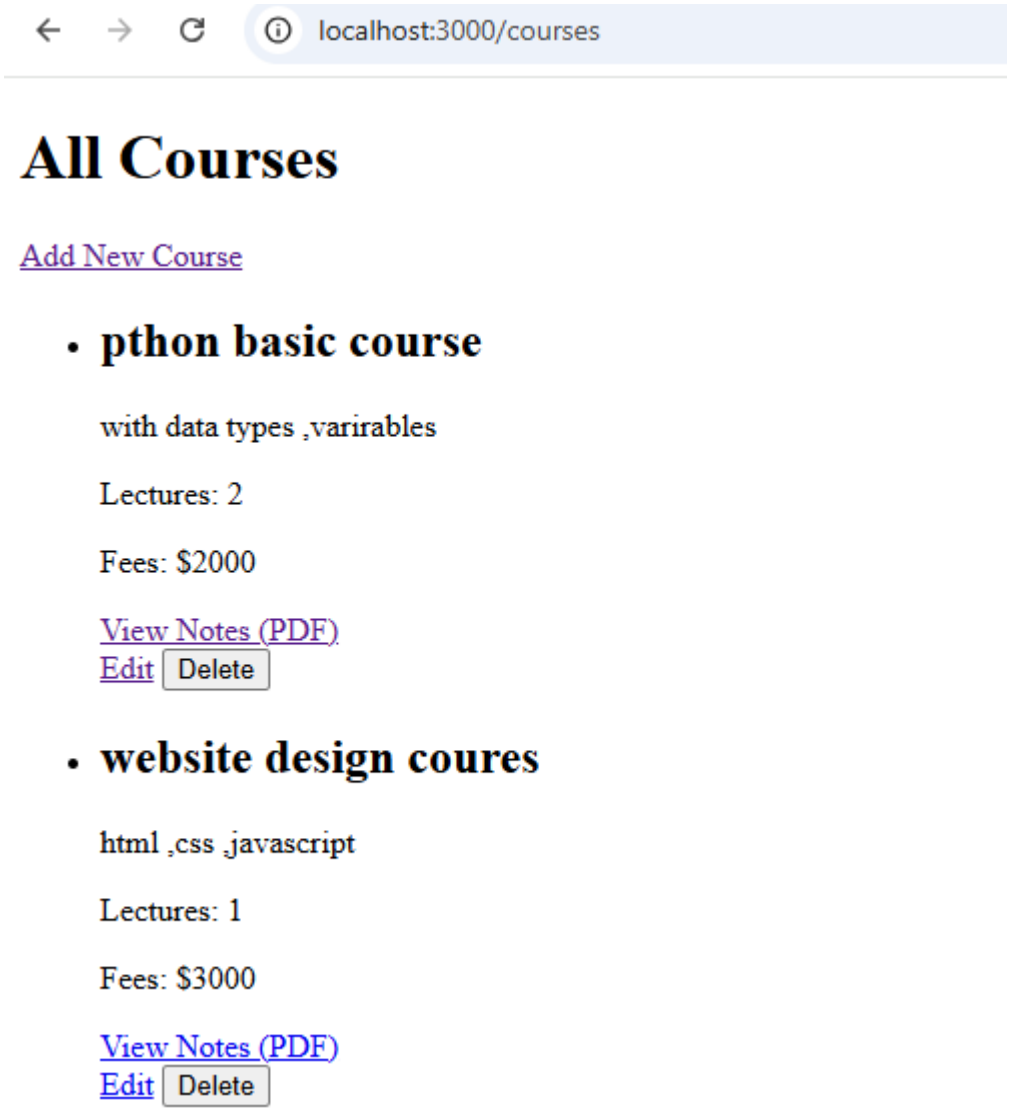
```
<!DOCTYPE html>
<html>
<head><title>Edit Course</title></head>
<body>
  <h1>Edit Course</h1>
  <form method="POST" action="/courses/<%= course._id %>" enctype="multipart/form-data">
    <input type="text" name="title" value="<%= course.title %>" required><br>
    <textarea name="description" required><%= course.description %></textarea><br>
    <input type="number" name="lectures" value="<%= course.lectures %>" required><br>
    <input type="number" name="fees" value="<%= course.fees %>" required><br>
    <p>Current Notes:
      <% if (course.notes) { %>
        <a href="/<%= course.notes %>" target="_blank">View PDF</a>
      <% } else { %>
        None
      <% } %>
    </p>
    <input type="file" name="notes" accept=".pdf"><br>
    <button type="submit">Update</button>
  </form>
  <a href="/courses">Back</a>
</body>
</html>
```

✅ 7. Run the Project

```
node app.js
```

Go to: **http://localhost:3000/courses**





You can go and check your inserted data to your mongo db using shell:-

```
> use mongo-crud
< switched to db mongo-crud
> db.courses.find().pretty()
< {
    _id: ObjectId('6891f1a358afd374da9c1cfc'),
    title: 'pthon basic course',
    description: 'with data types ,varirables',
    lectures: 2,
    fees: 2000,
    notes: '1754395043153.pdf',
    __v: 0
  }
  {
    _id: ObjectId('6891f1e558afd374da9c1d02'),
    title: 'website design coures',
    description: 'html ,css ,javascript',
    lectures: 1,
    fees: 3000,
    notes: '1754395109267.pdf',
    __v: 0
  }
```

Note we can also put Course.js and courseRouter.js file code together in app.js :-

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

const mongoose = require('mongoose');

const bodyParser = require('body-parser');

const multer = require('multer');

const path = require('path');


const app = express();
```

```
// 📄 Connect to MongoDB
mongoose.connect('mongodb://localhost:27017/mongo-crud', {
  useNewUrlParser: true,
  useUnifiedTopology: true,
});

// 🧾 Define Mongoose Model
const courseSchema = new mongoose.Schema({
  title: String,
  description: String,
  lectures: Number,
  fees: Number,
  notes: String // path to PDF file
});
const Course = mongoose.model('Course', courseSchema);

// 📂 Multer setup for PDF file uploads
const storage = multer.diskStorage({
  destination: (req, file, cb) => {
    cb(null, 'uploads/');
  },
  filename: (req, file, cb) => {
    cb(null, Date.now() + path.extname(file.originalname));
  }
});
const upload = multer({ storage });

// 🛠️ Middleware
app.set('view engine', 'ejs');
app.use(bodyParser.urlencoded({ extended: false }));
app.use(express.static(path.join(__dirname, 'uploads')));
```

```
// 📄 Routes

// Home redirect
app.get('/', (req, res) => res.redirect('/courses'));

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

// Show form to create new course
app.get('/courses/new', (req, res) => {
  res.render('new');
});

// Create new course
app.post('/courses', upload.single('notes'), async (req, res) => {
  const { title, description, lectures, fees } = req.body;
  const notes = req.file ? req.file.filename : null;

  await Course.create({ title, description, lectures, fees, notes });
  res.redirect('/courses');
});

// Show edit form
app.get('/courses/:id/edit', async (req, res) => {
  const course = await Course.findById(req.params.id);
  res.render('edit', { course });
});
```

```
// Update course
app.post('/courses/:id', upload.single('notes'), async (req, res) => {
   const { title, description, lectures, fees } = req.body;
   const updateData = { title, description, lectures, fees };

   if (req.file) {
      updateData.notes = req.file.filename;
   }

   await Course.findByIdAndUpdate(req.params.id, updateData);
   res.redirect('/courses');
});

// Delete course
app.post('/courses/:id/delete', async (req, res) => {
   await Course.findByIdAndDelete(req.params.id);
   res.redirect('/courses');
});

// 🔊 Start server
const PORT = 3000;
app.listen(PORT, () => {
   console.log(`Server running at http://localhost:${PORT}`);
});
```

☑ Also Ensure:

1. You have an **`uploads/`** folder created in your root:

   ```
   mkdir uploads
   ```

2. You have a **`views/`** folder with these 3 EJS files:
   - `index.ejs`
   - `new.ejs`
   - `edit.ejs`

🚗 Reminder:

This is a good way to **learn or prototype**, but in real-world projects:

- Keep models in `models/`
- Routes in `routes/`
- Views in `views/`
- App logic in `app.js`

It helps you scale and maintain your code more easily.

Very important meaning:-

const Course = mongoose.model('Course', courseSchema);

🧠 Meaning:

This line creates a Mongoose model called Course based on the courseSchema.

Here's what each part does:

1. mongoose.model('Course', courseSchema)

'Course' → This is the model name. It tells Mongoose:

Create a model named 'Course'

Use the schema courseSchema

Automatically link it to a MongoDB collection called courses (Mongoose auto-pluralizes the name)

courseSchema → This is the structure definition for a course document (i.e., fields like title, fees, notes, etc.)

2. const Course = ...

This stores the created model in a variable called Course, so you can use it in your app to:

Create new courses

Find courses in the DB

Update existing ones

Delete them

🔗 Example Usage:

```
// Create a new course
const newCourse = new Course({
    title: "Node.js Basics",
    description: "Learn Node.js from scratch",
    lectures: 10,
    fees: 299,
    notes: "nodejs.pdf"
});

await newCourse.save(); // saves it in MongoDB
```

Or find all courses:

```
const allCourses = await Course.find();
```

✅ Summary:

Part	Meaning
`'Course'`	Name of the model (also used for collection name: `courses`)
`courseSchema`	Schema that defines document structure
`mongoose.model(...)`	Creates a model class for CRUD operations
`const Course = ...`	Variable to use that model throughout your app