Lecture 10. Connecting MongoDB with Nodejs

12 April 2025 14:36

🌐 Node.js + MongoDB Connection – Revision Summary

☑ Why MongoDB with Node.js?

- MongoDB is a **NoSQL database** (document-based).
- Node.js + MongoDB = **Full-stack JavaScript**.
- Ideal for **flexible**, **scalable**, and **fast** applications.

🔧 Requirements

1. **MongoDB installed** (or use MongoDB Atlas for cloud DB)
2. npm install mongodb or mongoose (ODM)
3. Node.js runtime

METHOD : Using mongoose (Preferred ODM)

☑ Benefits:

- Schema-based
- Simplifies CRUD operations
- Validation, hooks, relationships, etc.

☑ Install:

```
bash
npm install mongoose
```

☑ Example:

```
// mongoose-connect.js
const mongoose = require('mongoose');

mongoose.connect('mongodb://127.0.0.1:27017/schoolDB')
  .then(() => console.log('☑ Mongoose Connected'))
  .catch((err) => console.error(err));

// Define schema
const studentSchema = new mongoose.Schema({
  name: String,
  age: Number,
});

// Create model
const Student = mongoose.model('Student', studentSchema);

// Create & save document
const student = new Student({ name: 'Krishan', age: 20 });
student.save()
  .then(() => console.log('Student Saved'))
  .catch(err => console.error(err));
```

📌 Key Methods in Mongoose:

Method	Use
Model.find()	Get all documents
Model.findOne()	Get a single document
Model.create()	Create a document
Model.updateOne()	Update a document
Model.deleteOne()	Delete a document

✅ Best Practice Tip:

- Use **environment variables (.env)** for DB connection string (especially in production).
- Always handle **connection errors** and use **async/await** with try-catch.

🧠 Quick Recap:

- mongodb: Low-level control
- mongoose: Easy to use with schemas
- MongoDB stores data in **JSON-like documents (BSON)**
- Connect using connect(), define Schema, create Model, and use CRUD methods