Day 19 – [19th July 2025]

TOPICS COVERED

Mongoose (with Async/Await + Try/Catch):

Mongoose is a powerful ODM library that helps us connect and interact with MongoDB in an organized way.

Today, I learned how to connect to MongoDB Atlas using Mongoose and handle asynchronous operations using async/await wrapped in try/catch blocks to avoid application crashes and handle errors properly.

Example Connection Code:

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

async function connectDB() {
  try {
    await mongoose.connect("your_mongodb_connection_string");
    console.log(" MongoDB connected successfully");
  } catch (error) {
    console.error(" MongoDB connection failed:", error);
  }
}

connectDB();
```

Schema and Model:

- Schemas define the structure of documents in MongoDB.
- A model is a wrapper for that schema to perform operations.

```
const userSchema = new mongoose.Schema({
  name: String,
  email: String,
  age: Number
});

const User = mongoose.model("User", userSchema);
```

Middleware:

Middleware functions are functions that have access to req, res, and next().

I used:

express.json() middleware to parse incoming JSON data:

app.use(express.json());

Params, Query, and Body:

- req.params: Read route parameters (like /users/:id)
- req.query: Read optional filters (like /users?age=20)
- req.body: Read submitted data from forms or frontend.

```
app.get("/users/:id", (req, res) => {
  console.log(req.params.id); // URL param
});
app.get("/search", (req, res) => {
  console.log(req.query); // Query string
});
app.post("/users", (req, res) => {
  console.log(req.body); // Form or JSON data
});
```

TOOLS USED:

VS Code: Code writing

MongoDB Atlas: Database in the cloud

Mongoose: ODM for structuring MongoDB data

Express.js: Backend server and routing

Hoppscotch: API testing tool

Nodemon: Auto-reload backend on changes

TASK:

Read about bcrypt