🔍 What is Mongoose & Why Do We Use It?

Mongoose is an **ODM (Object Data Modeling)** library for **MongoDB and Node.js**. It simplifies **interacting with MongoDB** by providing:

- **Schema-based models** (structure for our documents)
- **Built-in validation** (ensuring data consistency)
- **Query methods** (easy CRUD operations)

🗄️ What is MongoDB?

MongoDB is a **NoSQL database**, which means:

- It **stores data as documents** (JSON-like format called BSON).
- It is **schema-less** by default (you don't need to define table structures like in SQL databases).
- It is **flexible**, allowing for changes in data structure over time.

📝 SQL vs NoSQL (MongoDB)

Feature	SQL (MySQL, PostgreSQL)	NoSQL (MongoDB)
Data Storage	Tables & Rows (Structured)	Documents & Collections (Flexible)
Schema	Fixed schema (columns)	Dynamic schema (BSON)
Scaling	Vertical scaling (bigger servers)	Horizontal scaling (sharding)
Joins	Uses JOINs for relations	Uses embedded documents & references

---

🛠️ How Mongoose Works with MongoDB

Mongoose acts as a **bridge** between MongoDB and Node.js. It helps us:

1. **Define a schema** (structure for our documents)
2. **Create models** (representing collections)

3. **Perform CRUD operations** (using simple functions)

📌 Mongoose Workflow

1️⃣ **Define Schema** (Structure of your document)
2️⃣ **Create Model** (Based on schema)
3️⃣ **Use Model for CRUD Operations**

🔍 What is a Schema in Mongoose?

A **Schema** in Mongoose **defines the structure of documents** inside a MongoDB collection. It acts like a **blueprint** that tells MongoDB:

- **What fields** each document should have.
- **What data types** each field should be (String, Number, Boolean, etc.).
- **Optional rules** like default values, validation, or required fields.

📌 Breakdown of Schema Properties

Property	Description
`title: { type: String, required: true }`	Field **must** be a String and is **required**.
`yearPublished: { type: Number, default: 2000 }`	Field is a **Number** and defaults to **2000** if not provided.
`genre: { type: String, enum: [...] }`	Field **must** be one of the given values.
`available: { type: Boolean, default: true }`	Field is a Boolean, and the default value is **true**.

📌 Why Use a Schema?

✅ **Ensures Consistency** → Every document follows the same structure.
✅ **Prevents Errors** → Rejects invalid data types.
✅ **Adds Extra Features** → You can set **default values**, **validations**, and **constraints**.