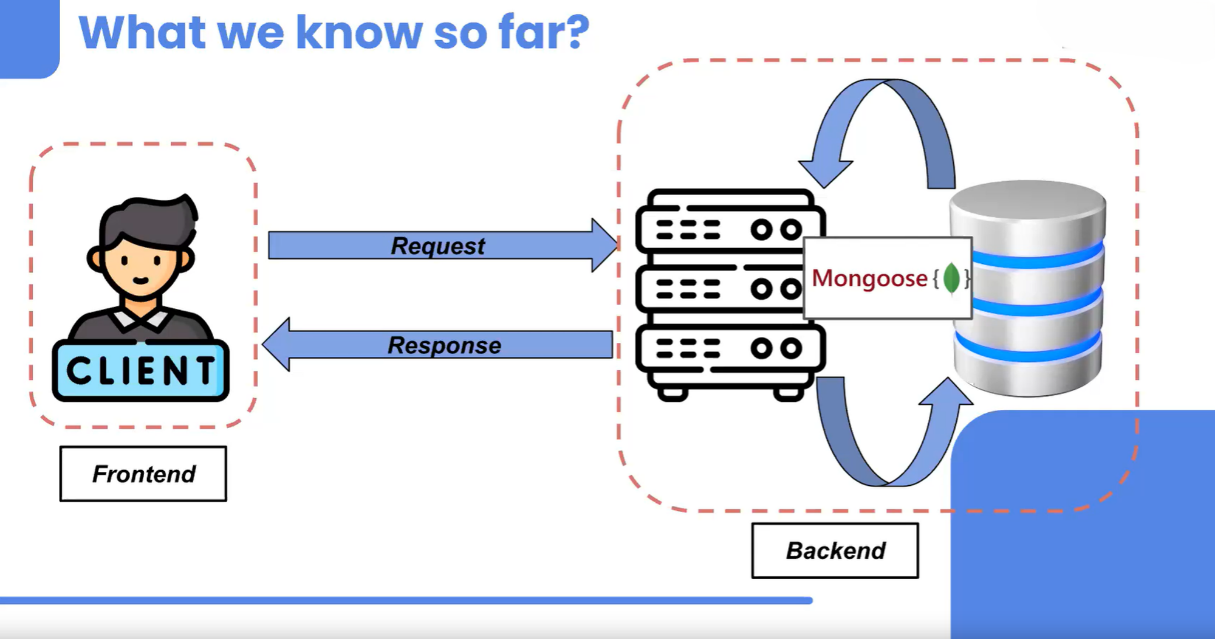
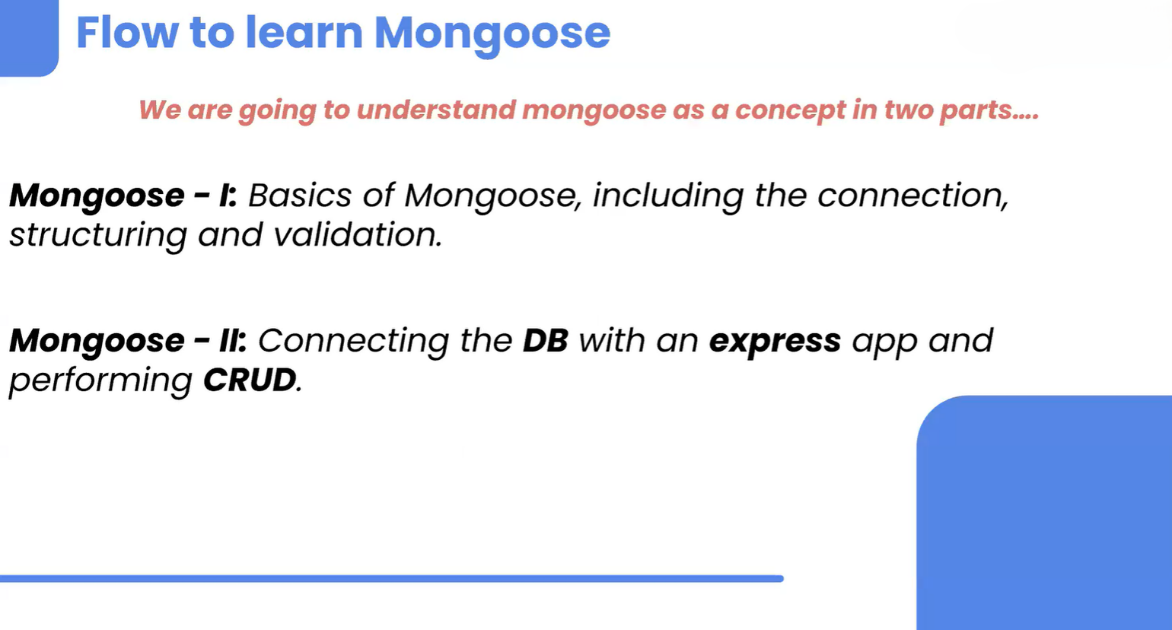
## Page:1

# [\*\*Mongoose](https://mongoosejs.com/docs/)-1\*\*

## Clinet-server-Model(Revision)



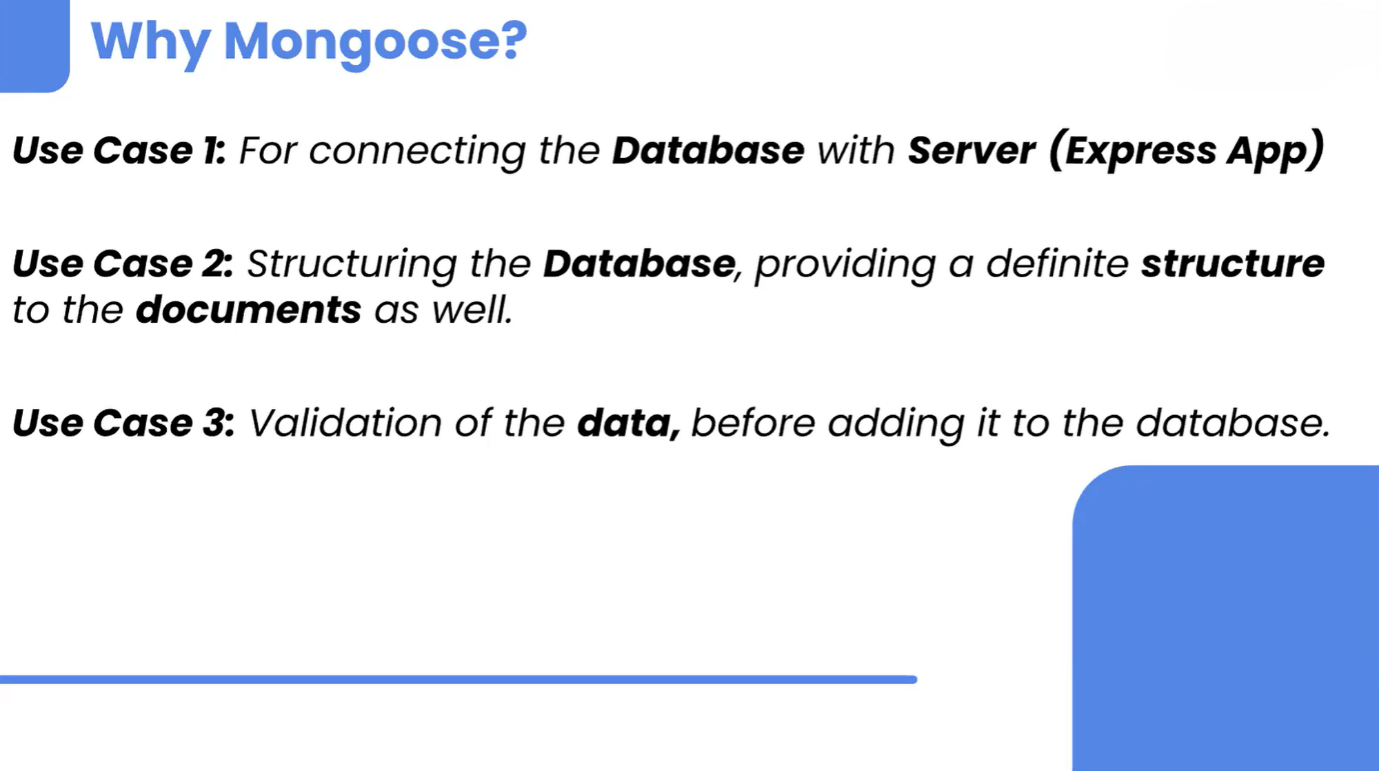
* **Client**: The client is the device or software (such as a web browser or mobile app) that makes requests to a server. Clients initiate communication with servers to request resources or services.
* **Server**: The server is a more powerful machine or software application that provides services or resources requested by clients. Servers wait for incoming requests from clients, process them, and then respond with the requested data or services.



## Page:2

**Understanding Mongoose:**

Mongoose is a popular library that allows you to establish a connection between your MongoDB database and your server. While you can connect to MongoDB using the Mongo Driver, Mongoose offers a more user-friendly interface, making it easier to understand and implement.

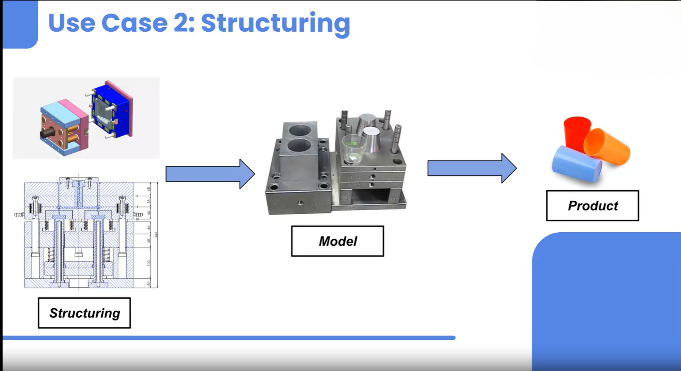


## Why Learn Mongoose? 🤔

Mongoose serves several purposes in backend development:

1. **Database Connection**: Mongoose simplifies connecting your server to a MongoDB database.
2. **Schema Definition**: It helps in defining the structure (or schema) of your documents, ensuring consistency across the database.
3. **Validation**: Mongoose provides built-in validation to ensure the data stored in your database adheres to defined rules.

**Flexibility in Structure**: You might wonder, "If we're defining a structure, how flexible is Mongoose?" The answer is simple—Mongoose allows you to modify the structure (or schema) anytime you want, giving you flexibility when you need to make changes.



## Page:3

**Getting Started with Mongoose 🚀​**

Let's create a backend project and connect it to a MongoDB database using Mongoose. First, we need to install Mongoose:

Shell

Copy

bashCopy code

npm i mongoose

**Connecting to MongoDB**

Since connecting to the database is an asynchronous operation, it might take some time, and we need to handle any potential errors. Here’s a basic example to get you started:

javascriptCopy code

// connect.js

const mongoose = require("mongoose");

const main = async () => {

try {

const connection = await mongoose.connect("mongodb://127.0.0.1:27017");

console.log("Connected to Database");

// Disconnect example (optional)

connection.disconnect();

console.log("Disconnected");

} catch (err) {

console.log("Error connecting to DB");

console.log(err);

}

}

main();

## Page:4

**Defining the Structure: Schemas and Models 🏗️**

**Schema**: Think of a schema as a blueprint for your documents. It defines the structure and types of data that will be stored.

**Model**: A model is essentially a constructor that creates and manipulates documents within the database, based on the defined schema.

Here’s how you can define a simple schema and model:

javascriptCopy code

const mongoose = require("mongoose");

// 1. Connect to MongoDB

const main = async () => {

try {

const connection = await mongoose.connect("mongodb://127.0.0.1:27017/newdb");

console.log("Connected to database");

await StudentModel.insertMany([{ name: "Rahul", age: 22, city: "Pune" }]);

console.log("Data added");

connection.disconnect();

console.log("Disconnected");

} catch (err) {

console.log("Error connecting to DB");

console.log(err);

}

}

main();

## Page:5

// 2. Define the structure of the document

const studentSchema = new mongoose.Schema({

name: String,

age: Number,

city: String,

}, { versionKey: false });

const StudentModel = mongoose.model("Student", studentSchema);

**Understanding Mongoose Features**

1. **\_id Field**: MongoDB automatically adds an \_id field to uniquely identify each document.
2. **\_\_v Field**: Mongoose adds a \_\_v field to track the document version, but this can be disabled by setting { versionKey: false } in the schema options.

**Creating and Manipulating Documents 📄**

Using Mongoose models, you can create and manipulate documents. For example, to insert a single document:

javascriptCopy code

const student = new StudentModel({

name: "Pulkit",

age: 27,

city: "Delhi"

});

await student.save(); // Saves the document to the database

**Reading Data: Retrieving Documents from MongoDB 📚**

To read all documents from the Student collection:

javascriptCopy code

const students = await StudentModel.find();

console.log(students); // Outputs all student documents in the terminal

​

With Mongoose, you can leverage all the CRUD operations you've learned in MongoDB, making it a powerful tool for managing your database. Happy coding! 🎉​