Based on client action on frontend our database should get changed automatically.

**Send Request to server (node.js)**

**Server validate the client, and if request require the access of database, then server use ODM (Object Data Module).**

**We can access MongoDB through Mongoose**

We are going to use popular ODM for **MongoDB** called **Mongoose** and this is open source and free and maintain by community.

**Frontend**

**Changes done inside the Database**

**Client**

Basically, we can access of database (MongoDB) through our server (Node.js) via ODM (Mongoose).

In noSql (mongoDB) data is stored as collection of objects where document is object and collections is array of document. Each document is independent.

In frontend we display these data which we are getting from database.

Lets consider a senario

We have 2 documents inside the blogs collection, and we want to fetch all the author name at frontend, to fetch the array we will access author key and use map() method.

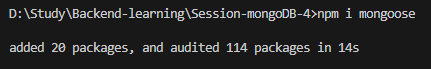


Did u notice that value of author in Document-1 is [] and Document-2 is “”, So data in database are inconsistency.

So, to avoid this inconsistency we have to add schema-validation (structure) to our data in MongoDB while storing the data.

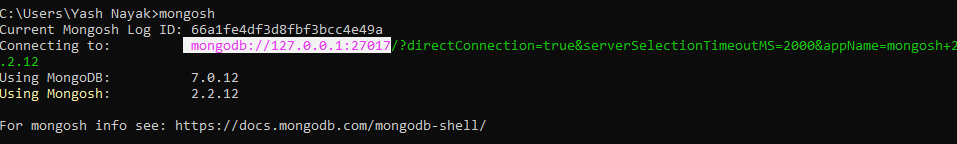
Since MongoDB is schema-less, we will use Mongoose to add schema-validation and mongoose also provide a interface to perfume CURD operation on MongoDB inside our express server.

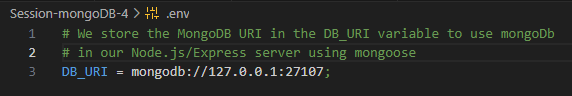
**Using Mongoose:**

* Mongoose is the npm package, it is ODM of MongoDB and Mongoose only support Node.js and Deno.
* **npm i mongoose** : To install mongoose:

**Connecting to MongoDB (DB):**

* To run the MongoDB, it requires an IP address (usually localhost or 127.0.0.1) and a port (default is 27017). When we run MongoDB, it typically runs on mongodb://127.0.0.1:27017, and we can access MongoDB using this URI.
* So, we store the IP address and port in the **DB\_URI** variable, which we use to access our database inside MongoDB through the Node.js/Express server.





* In production, backend-server will run on different server and database-server will run on different server.
* We can switch between **dbs** inside MongoDB through the code in server, but if we know that we will use only this specific **db** in our server then, we can mention db’s name during **DB\_URI** initialization inside **.env** file.
* Suppose we will use only **website db** from the MongoDB in our server then we can write 
* If we didn’t mention **db** while initializing the **DB\_URI** or didn’t switch **db** through code in side server, then MongoDB will use default **db** i.e. **test** to perform all the operations.
* See index.js and db/config.js file for more info and code.

**Make schema and wrap it with model:**

* See models/blog.model.js for more info.