**CURD Operation inside MongoDB’s Database**

1. **Express Server Setup:**
   * 1. Make a folder for the project and name is as you wish and go inside it, I have named my-folder as Session-5.



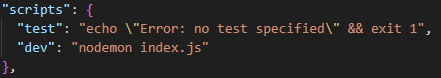
* + 1. In Terminal, CD to Session-5 and run **npm init** command, it will ask some question just hit enter to make all details as default.



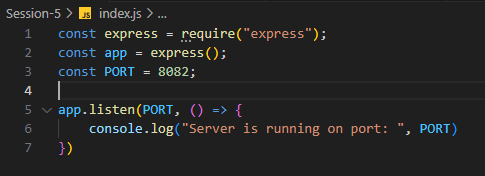
* + 1. Run **npm i express** to import express package for building the server, then run   
       **npm i nodemon** **-D** to import nodemon dev-dependencies package to avoid manual restart of our server when we do any changes in our server, then go the **package.json** file and inside **“scripts”** add **“dev”: “nodemon index.js”.**

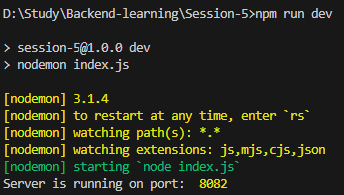






* + 1. Make **index.js** file inside Session-5 (your-project-folder) and inside it import express, make a **app** variable and add express instance to it, define the **PORT** and run the server on the PORT that you have defined.  
       Then go to the terminal and run **npm run dev** command to execute index.js and express-server will start and you will get a message that you have given in console.log()

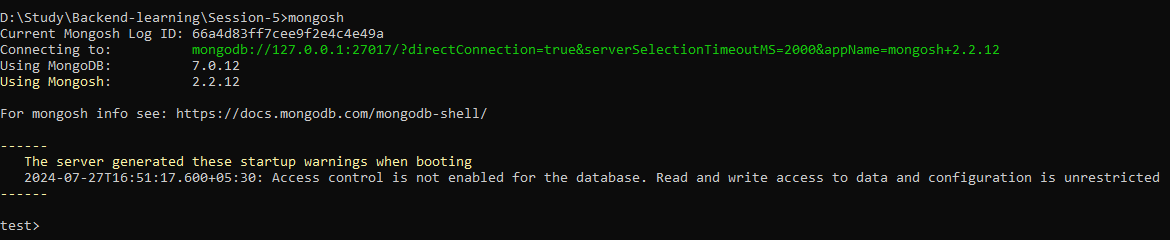




1. **MongoDB and Mongoose Setup:**
2. I have installed MongoDB in my system and its service is running in the system.  
   And in backend MongoDB Server will run on **mongodb://localhost:27017**

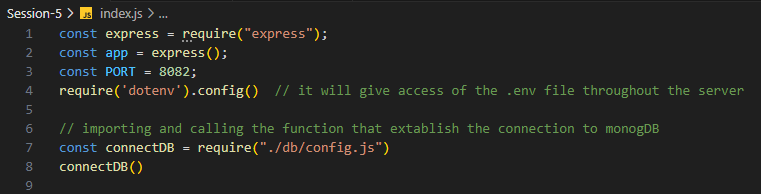
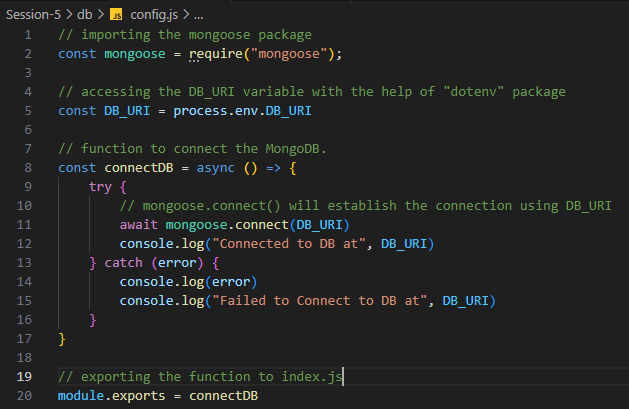


1. Run the client tool of MongoDB called **mongosh** (MongoDB Shell) in terminal, this command will **connect** to the MongoDB-Server which is running on **mongodb://localhost:27017** and we get the access of the MongoDB-Server and its databases.



1. Now we can access the MongoDB-server through terminal, but we have to access it in our express-server’s code.  
   So we have to import mongoose package (**npm i mongoose**) in project-folder, mongoose is the one of the popular **ODM** (Object Data Model) of the MongoDB.  
   With the help of the mongoose we can **connect and access** the MongoDB-server in our Express-server.



1. Now we have to configure MongoDB (Database-server) in our Express (Backend-server).  
   We should never **expose** the **Database-server location** in our code for security purpose, therefore we will make **.env** file (environment-file) inside project-folder, inside the **.env** file assign the MongoDB location to variable lets named it **DB\_URI**,  
   Make sure to add .env file inside **.gitignore** file.  
   To access variables inside .env file in our code during runtime we have to install a package called **dotenv** (**npm i dotenv**) and then we have to import the config() function inside the dotenv package (**require(“dotenv”).config()**)  
   Make sure that you import dotenv’s config() at starting of index.js because it will give environment-variables access through-out the backend-server.  
   When we run our Express-server then in backend process start and create a global-object called process, so during run-time dotenvwill create a object called env and inside the env it will add the all the variable and its value as key-value pair and then env object will get added inside process-object.  
   In our project-folder make a folder called **db** and inside the db folder create **config.js** file in this file we will connect our MongoDB server using mongoose and then import config.js inside the index.js.  
     
   
2. **POST Method Using Model.create()**https://mongoosejs.com/docs/api/model.html#Model.create()  
   * 1. In blogs.controllers.js, i made a async-function which will create the hard-coded Document based on model and Document will get added inside websiteBlogs collections which is inside website database. when client send POST method to the **/blogs** endpoint.  
        you will see that website-database will get created, websiteBlogs collection will get added to the website-database and the hard-coded document which we have send from backend-server it will get added to websiteBlogs-collection. And you will also see the response send to the client.



