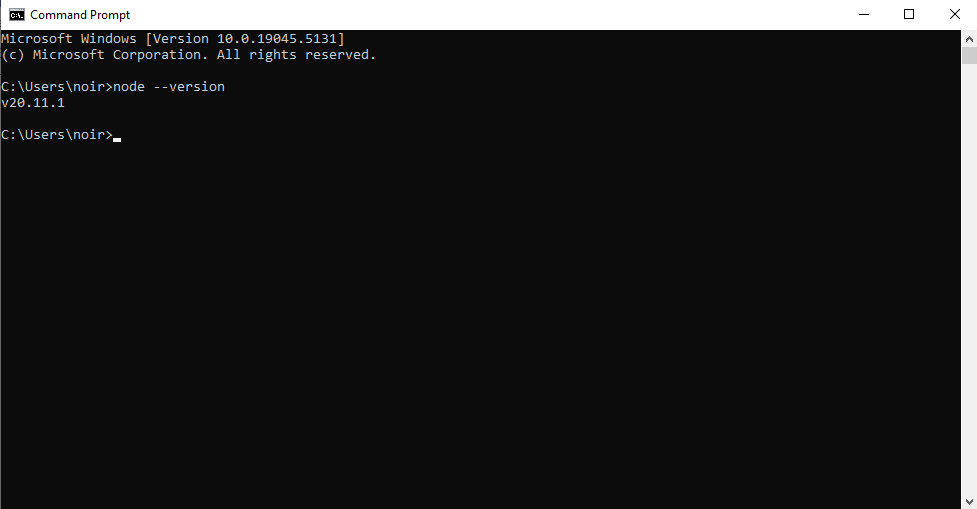
**BACKEND**

**STEP 1.**

Download and Install Node.js, prebuilt installer: <https://nodejs.org/en/download/prebuilt-installer>

**STEP 2.**

Check if Node.js is installed, by typing “node –version” in cmd.



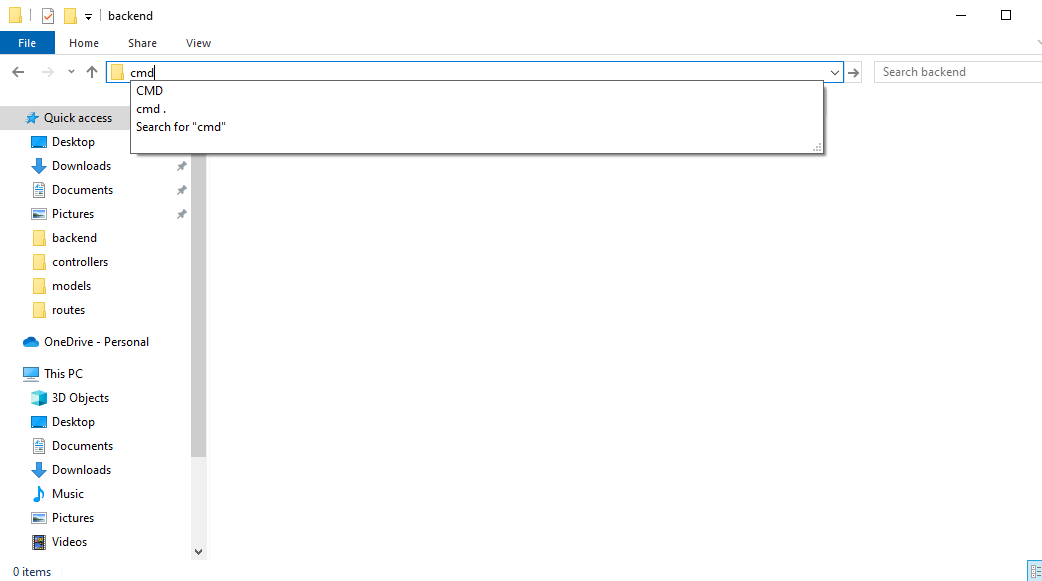
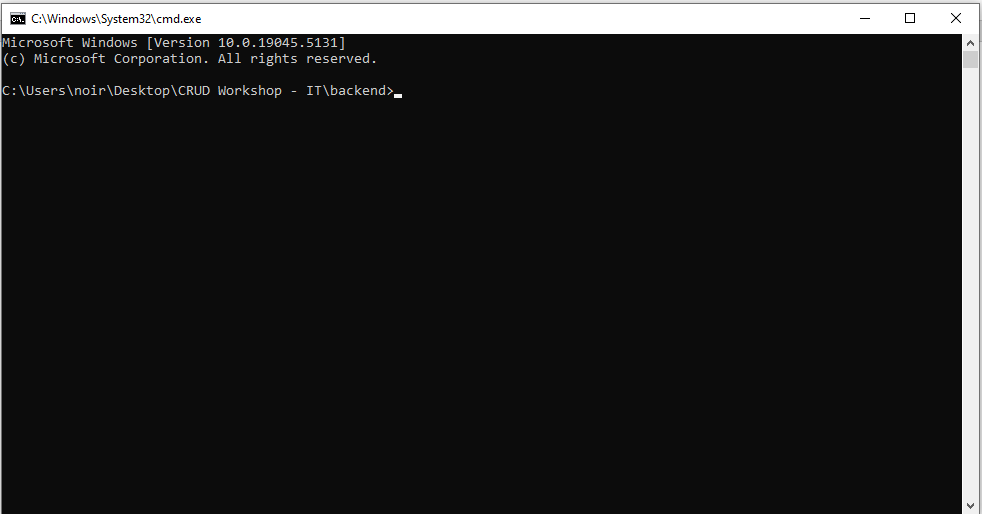
**STEP 3.**

Create a folder name “BASIC CRUD APP”. Inside the root folder, create another folder named “backend”.

* BASIC CRUD APP
  + backend

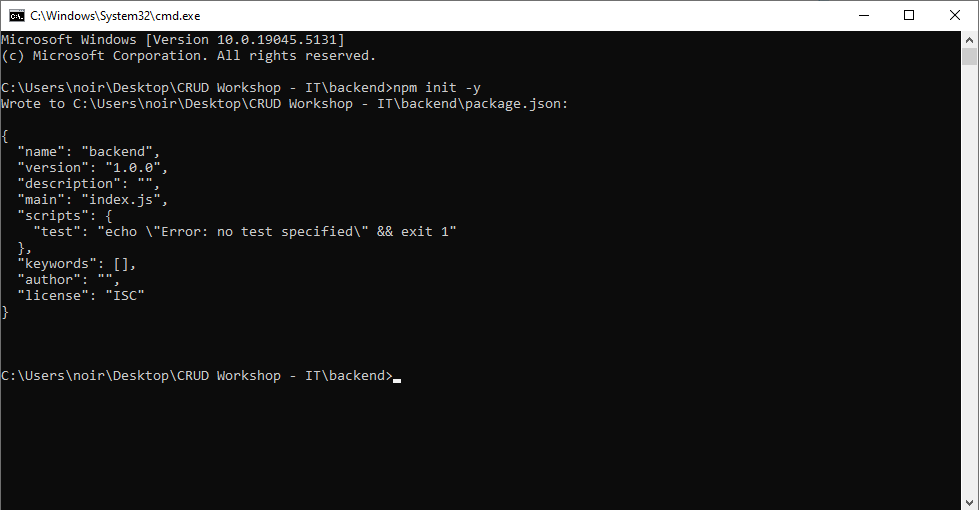
**STEP 4.**

Access the backend folder, in the path field type “cmd”, then hit enter

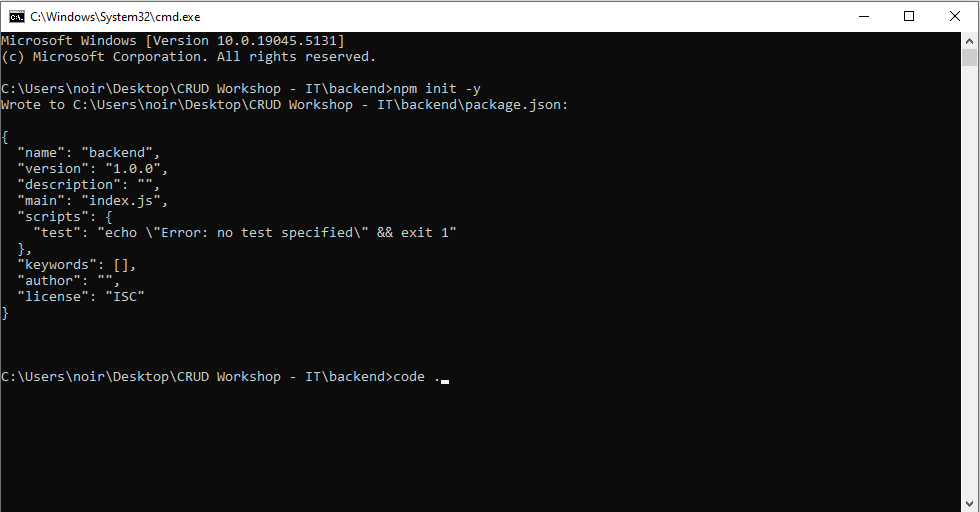
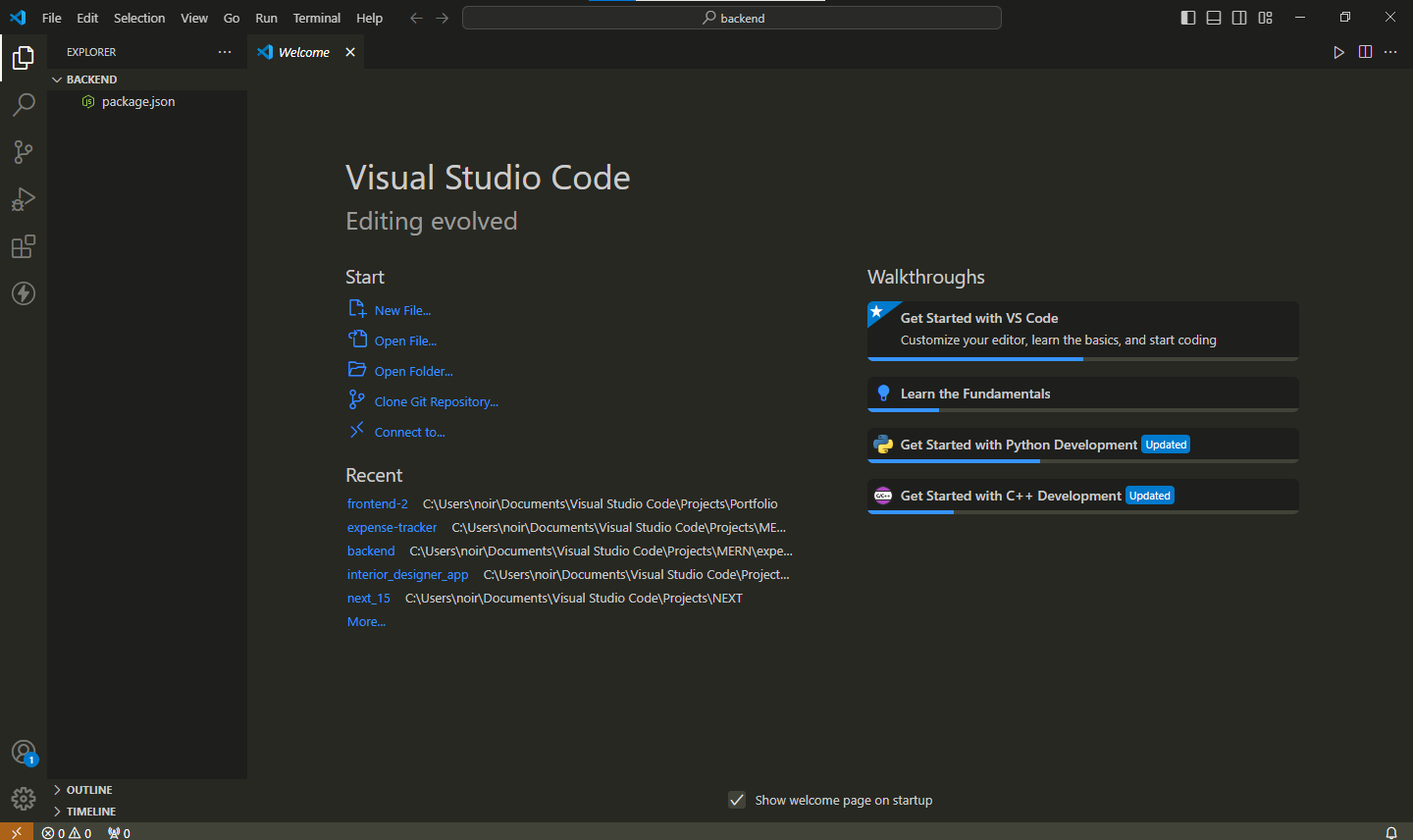
**STEP 5.**

In the cmd, type “npm init -y”, then hit enter.



**STEP 6**

In the cmd, type “code .”, then hit enter, it should the VS Code Editor.

STEP 7.

In the explorer section, click the “package.json”, then add this code block:

{

*"name"*: "backend",

*"version"*: "1.0.0",

*"description"*: "",

*"type"*: "module",

*"main"*: "server.js",

*"scripts"*: {

*"start"*: "nodemon server.js",

*"test"*: "echo \"Error: no test specified\" && exit 1"

  },

*"keywords"*: [],

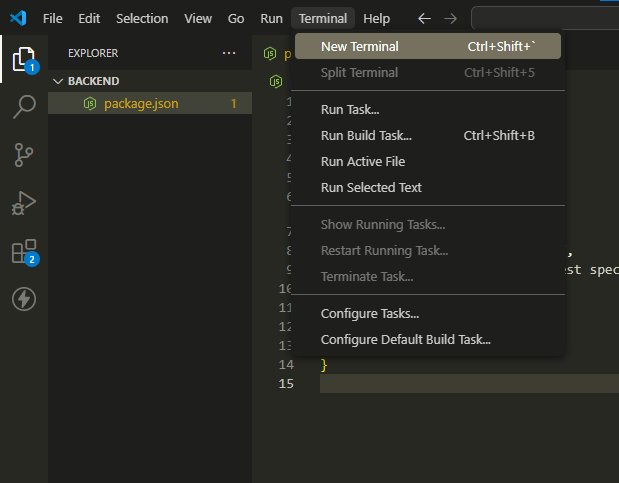
*"author"*: "",

*"license"*: "ISC"

}

STEP 8.

Go to terminal, then click new terminal.

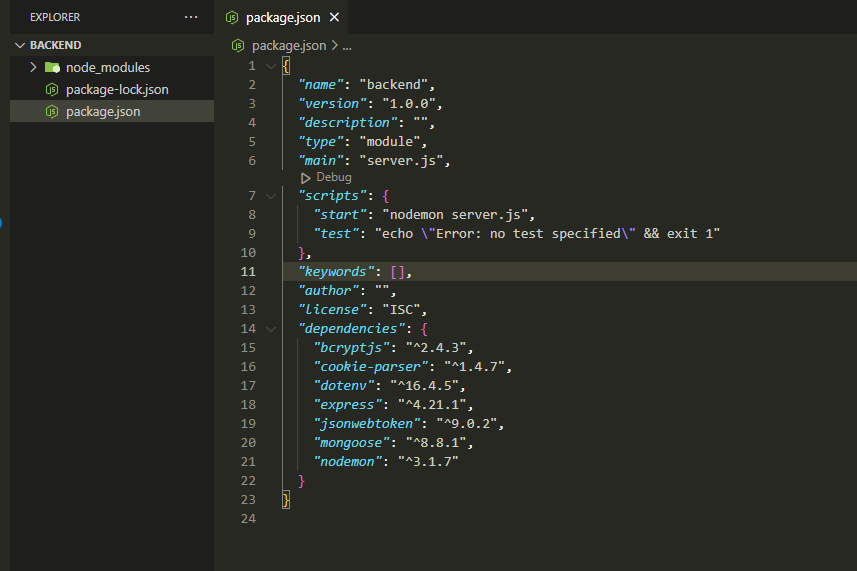


STEP 9.

Install the required depencies by hitting enter.

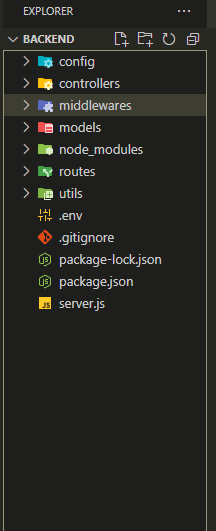
npm install nodemon express mongoose dotenv multer

Your “package.json” should have the dependencies installed.



STEP 10.

In the explorer section, create a new file name “server.js”, “.env”, and “.gitignore”. Also create folders, “config”, “controllers”, “routes”, “models”, and “middlewares”



STEP 11.

Click the “.env” and add this port:

PORT=5000

Crtl + s to save

STEP 12.

Click the “.gitignore” and add this statement:

.env

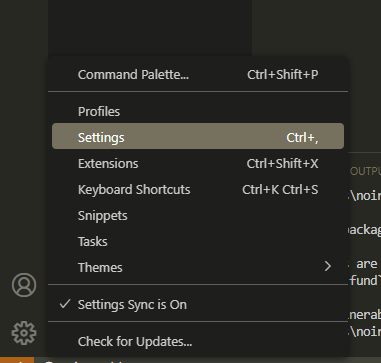
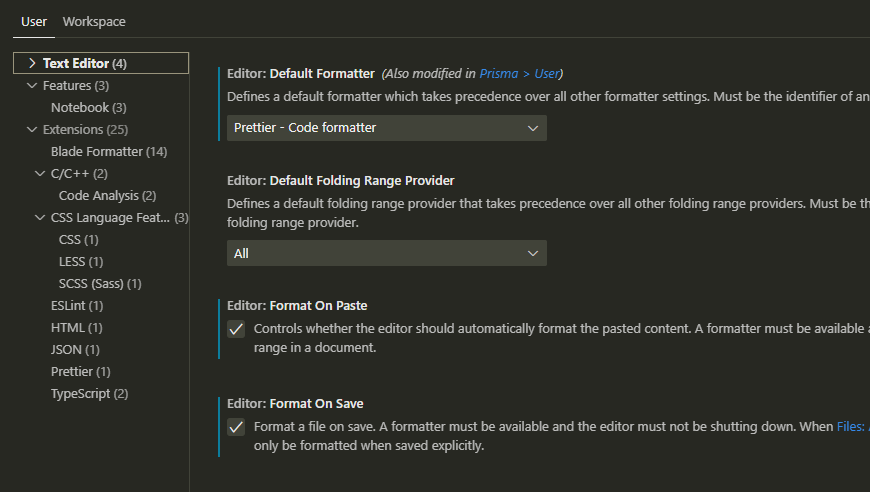
Crtl + s to save

STEP 13. (Optional)

Click the “Extensions tab”, then install “material icon” and “prettier code formatter”.

STEP 13.1. (Optional)

In the settings, type “formatter”. Click the Text Editor. Set the default formatter to “Prettier – Code Formatter”, and check “Format on Paste” and Format on Save”

STEP 14.

Click the “server.js”, then add this code block:

import express from "express";

import dotenv from "dotenv";

dotenv.config();

*const* app = express();

// Middlewares

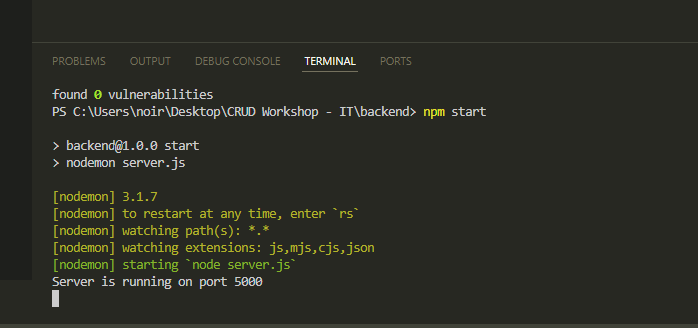
app.use(express.json());

app.use(express.urlencoded({ extended: true }));

// Listen

*const* port = process.env.PORT || 4000;

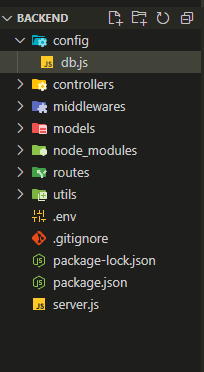
app.listen(port, () *=>* console.log(`Server is running on port ${port}`));

CTRL + s to save. In the terminal, type “npm start” then hit enter. 

Good job you have created a server.

STEP 15. Create a mongo db connection.

In the explorer, creater a file named “db.js” under the “config” folder.



Click the “db.js”, then add this code block:

import mongoose from "mongoose";

*const* connectDB = async () *=>* {

  try {

    await mongoose.connect(process.env.MONGO\_URI);

    console.log(`MongoDB is Connected: ${mongoose.connection.host}`);

  } catch (error) {

    console.log(`Error: ${error.message}`);

    process.exit(1);

  }

};

export default connectDB;

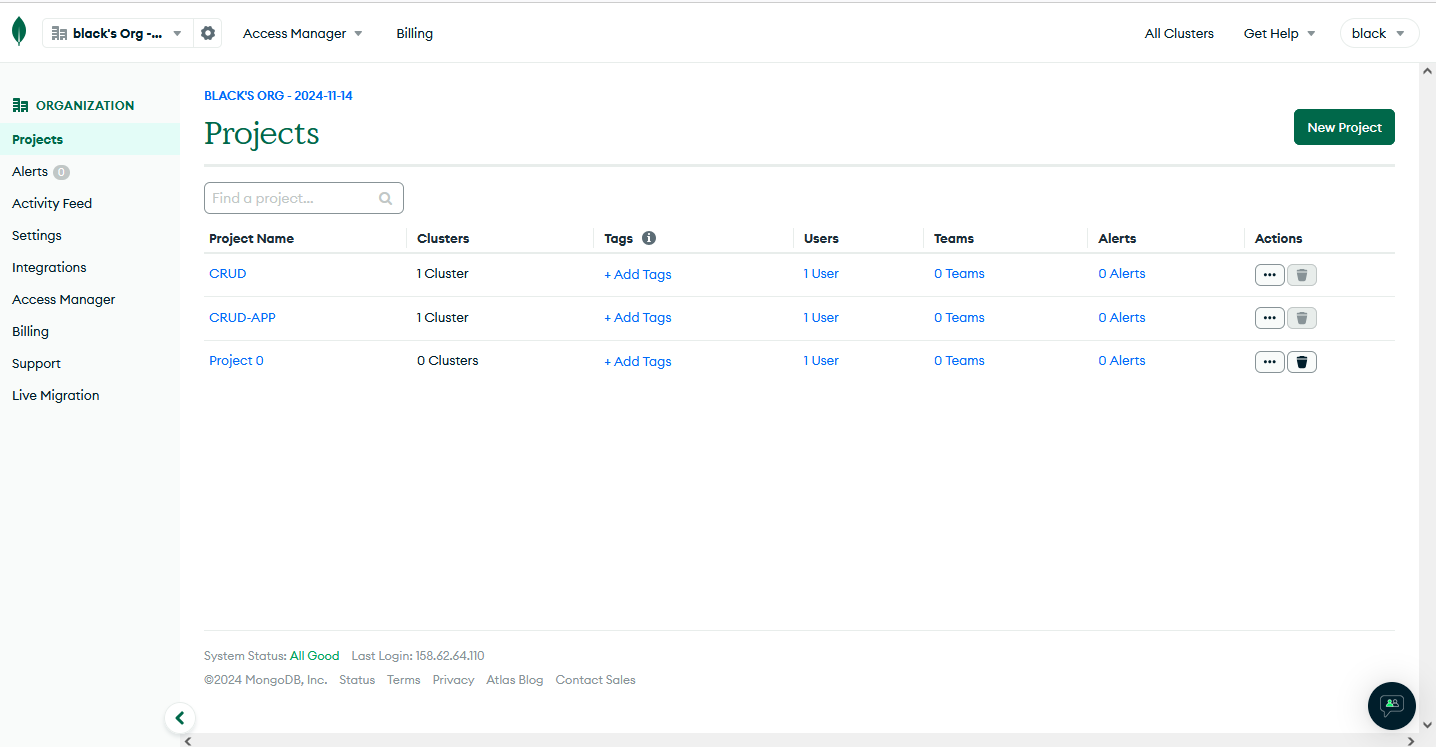
CRTL + S to save.

STEP 16. Create a mongo db account

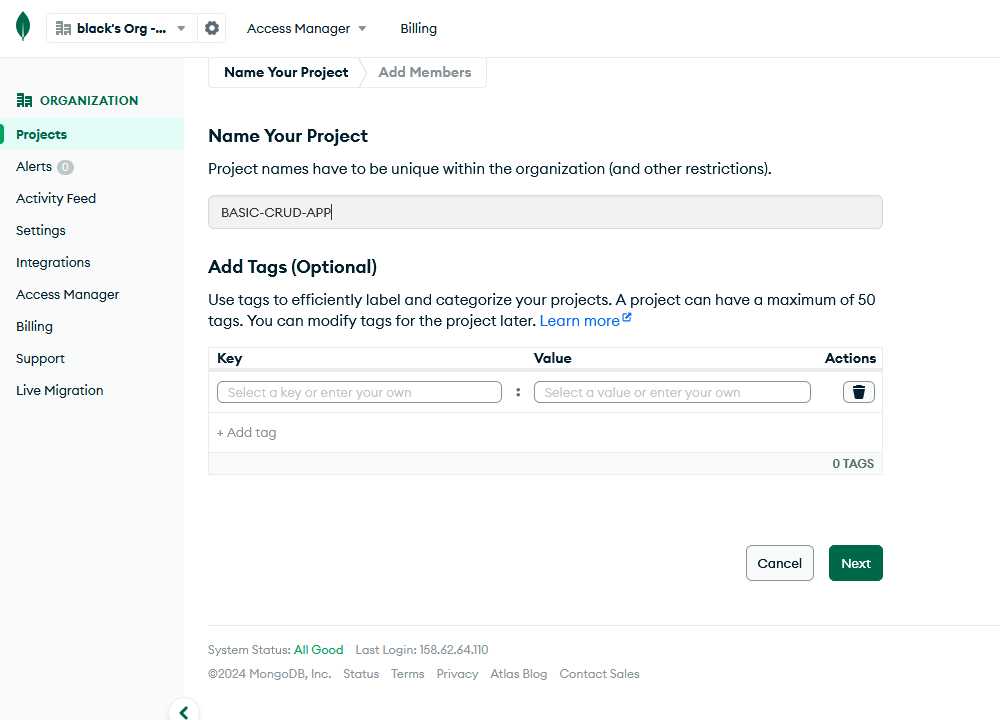
Sign in to MongoDB Atlas: <https://account.mongodb.com/account/login?n=https%3A%2F%2Fcloud.mongodb.com%2Fv2%2F65f6b51c49a07b25297e2fd4&nextHash=%23clusters&signedOut=true>

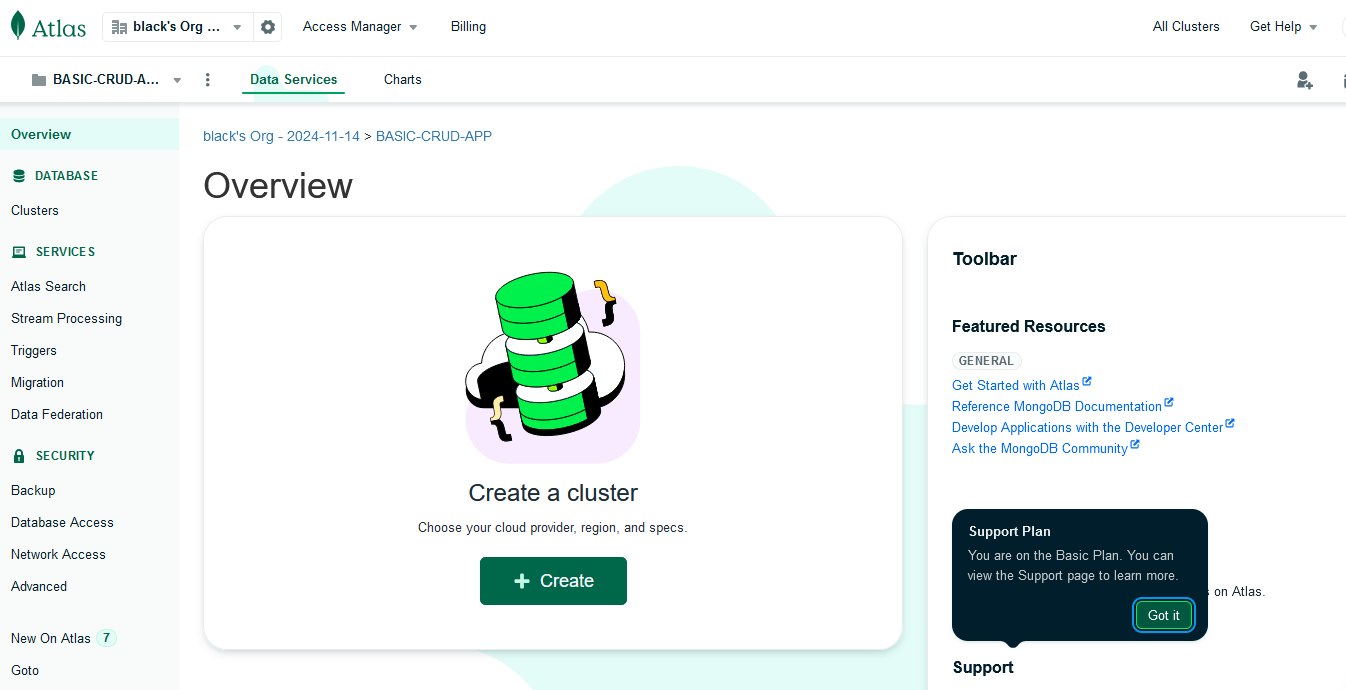
Use google account to sign in.

After Signing in. Create a new project.

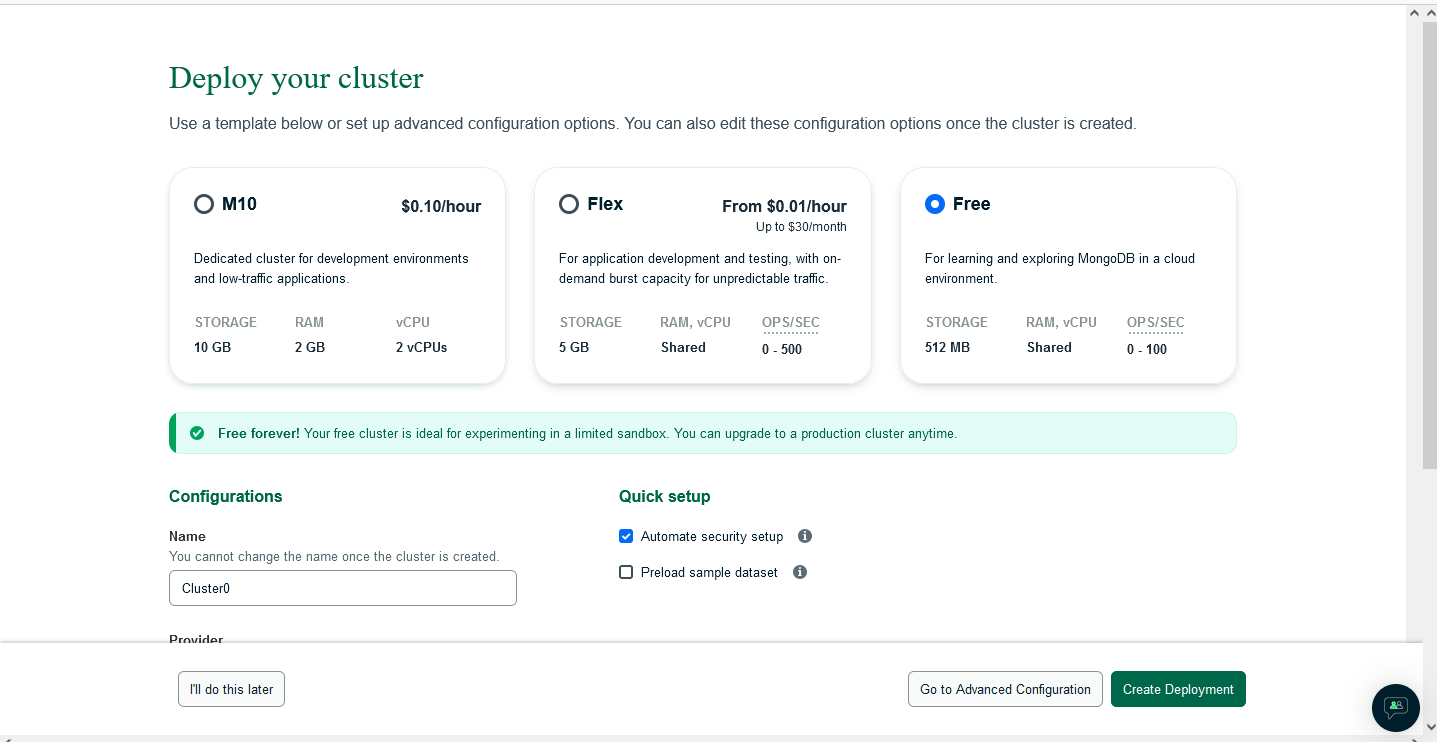


Name the project. Then next and create project.

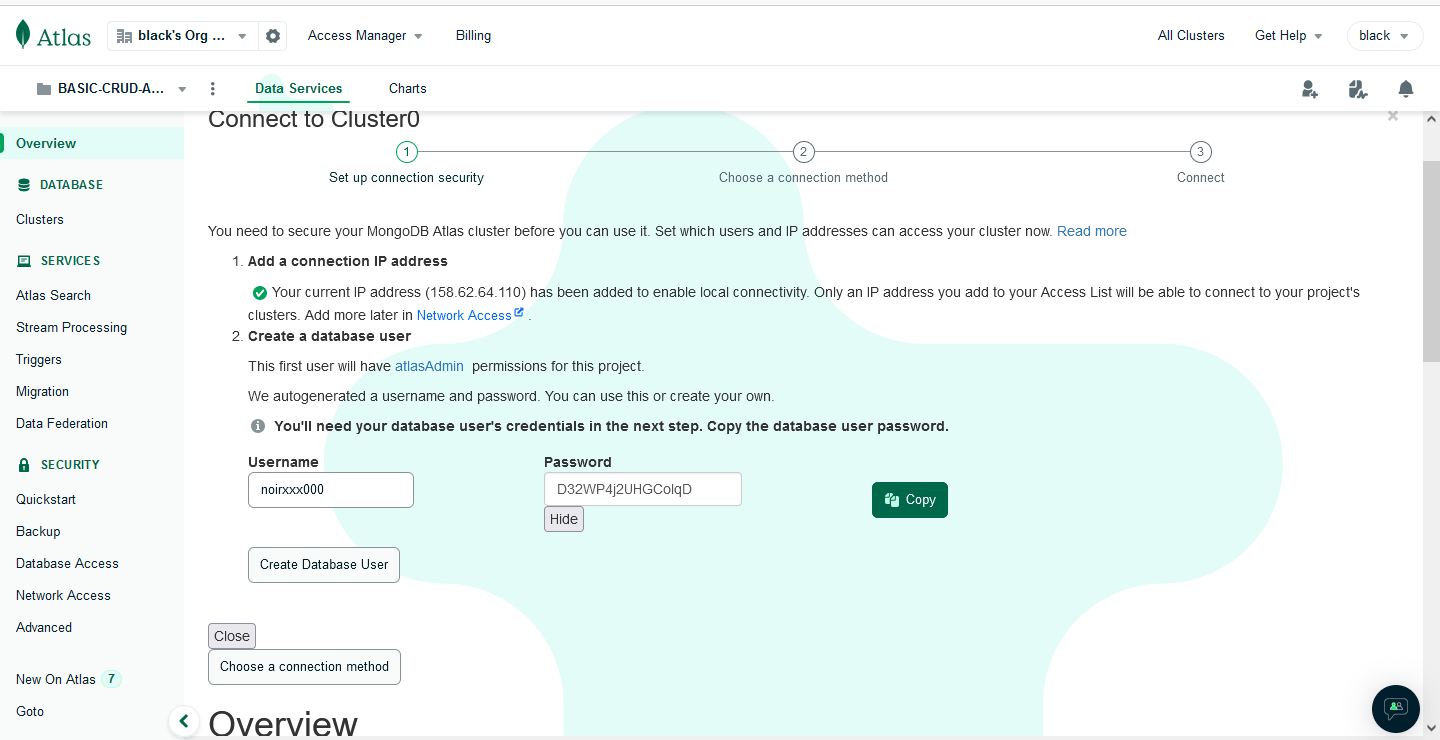


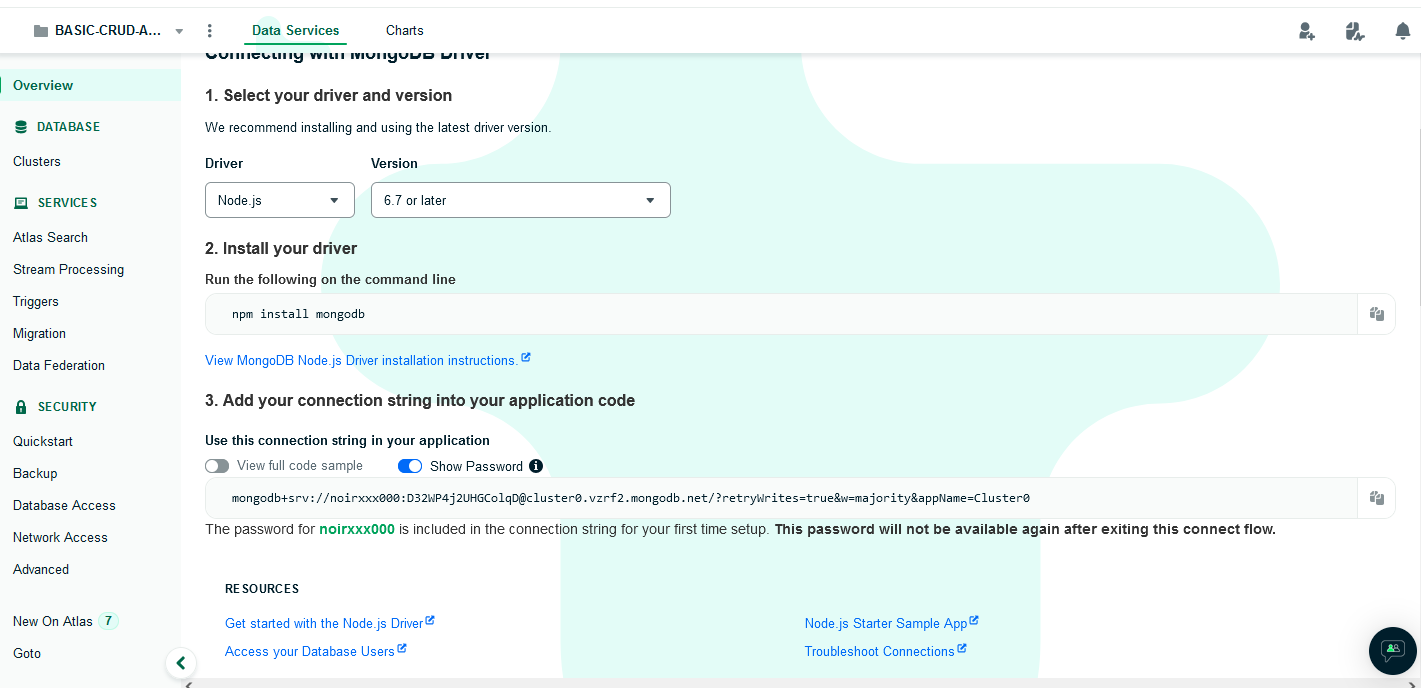
After creating, create a cluster. 

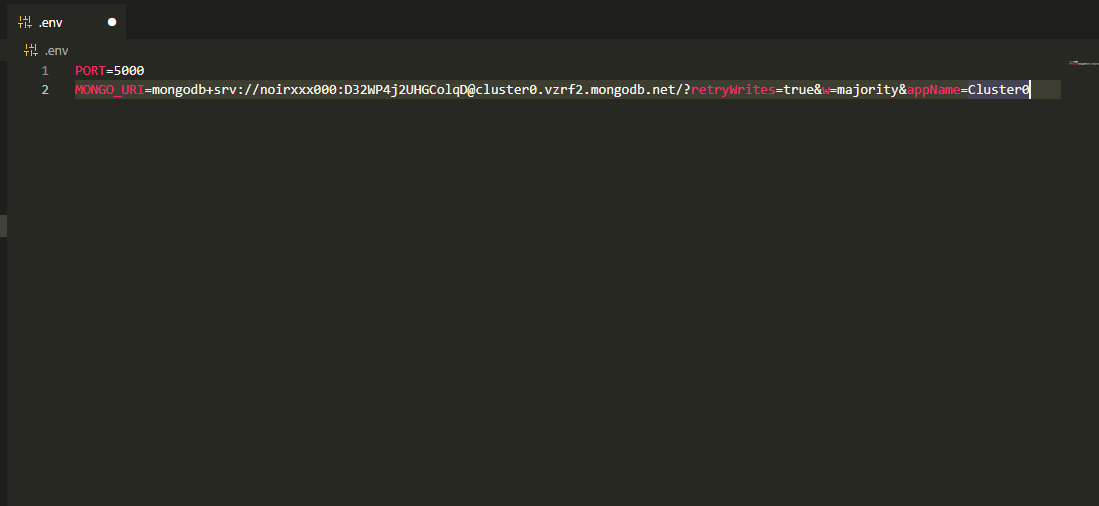
Choose the free tier, then create deployment



Copy the password and paste on a notepad. After that “create database user”

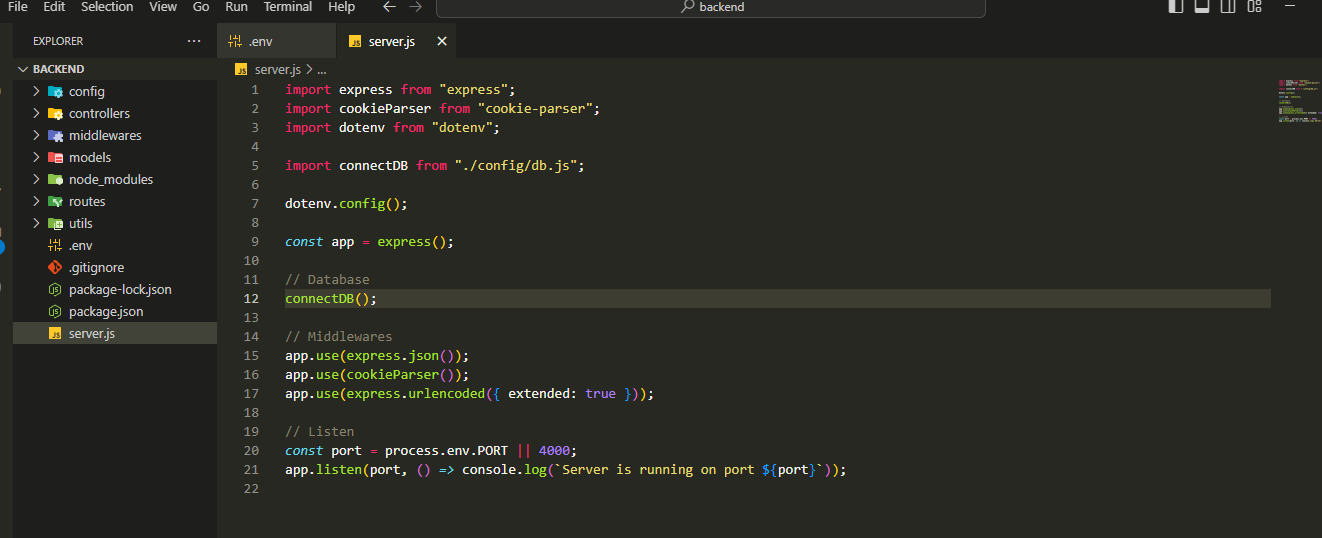


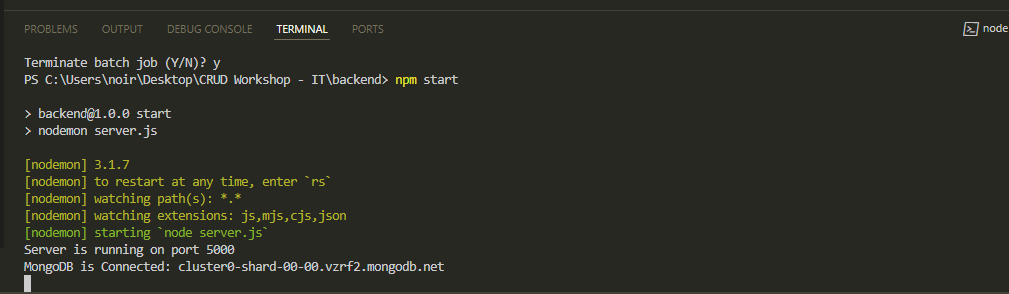
After creating, choose connection, then select “driver”. Copy the connection string then paste it on “.env” file. 



CRTL + S to save

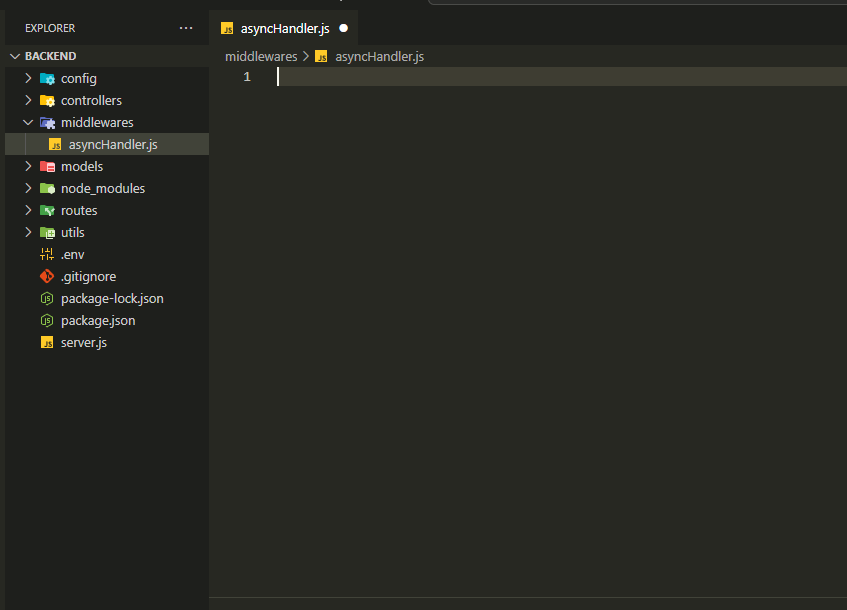
STEP 17.

Click the “server.js”, import the connectDB from db.js, and add the connectDB under “app” 

CRTL + S to save. Then restart the server. It should display like this 

STEP 19.

Create a file named “asycnHandler.js” under the “middlewares” folder



Add this code block:

*const* asyncHandler = (*fn*) *=>* (*req*, *res*, *next*) *=>* {

*Promise*.resolve(fn(*req*, *res*, *next*)).catch((*err*) *=>* {

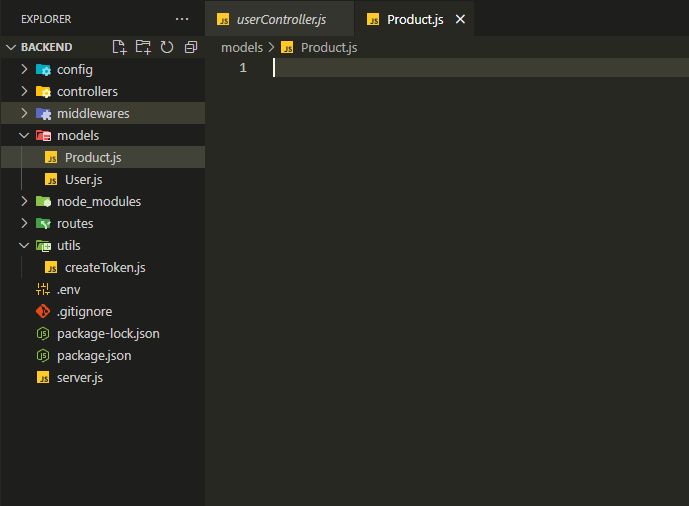
*res*.status(500).json({ message: *err*.message });

  });

};

export default asyncHandler;

STEP 26:

In the editor, create a file named “Product.js” under the models forler. 

Add this code block:

import mongoose from "mongoose";

*const* productSchema = new mongoose.Schema(

  {

    productName: {

      type: *String*,

      required: true,

    },

    price: {

      type: *Number*,

      required: true,

      min: 0,

    },

    description: {

      type: *String*,

      required: true,

      trim: true,

    },

    image: {

      type: *String*,

      required: true,

    },

  },

  {

    timestamps: true,

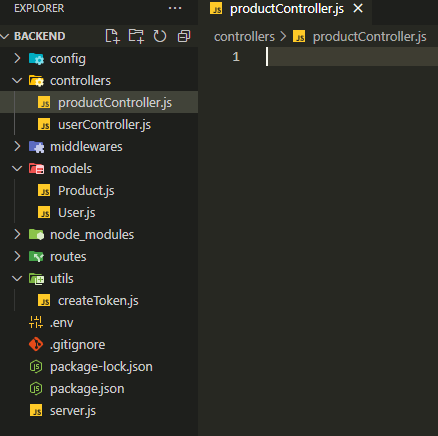
  }

);

*const* Product = mongoose.model("Product", productSchema);

export default Product;

STEP 27:

Create a file named “productController.js” under the “controllers” folder. 

Add this code block:

import Product from "../models/Product.js";

import asyncHandler from "../middlewares/asyncHandler.js";

export *const* createProduct = asyncHandler(async (*req*, *res*) *=>* {

  try {

*const* { productName, price, description } = *req*.body;

*const* image = *req*.file ? *req*.file.path : null; // Get the image path from multer

*const* newProduct = new Product({

      productName,

      price,

      description,

      image, // Save the image path to the product model

    });

*const* savedProduct = await newProduct.save();

*res*.status(201).json({

      message: "Product created successfully",

      savedProduct,

    });

  } catch (error) {

    console.log(error);

*res*.status(400).json({ message: "Error creating a product", error });

  }

});

export *const* updateProduct = asyncHandler(async (*req*, *res*) *=>* {

  try {

*const* { productId } = *req*.params;

*const* updates = *req*.body;

    if (*req*.file) {

      updates.image = *req*.file.path; // Add the new image path if an image is uploaded

    }

*const* product = await Product.findByIdAndUpdate(productId, updates, {

      new: true,

    });

    if (!product) {

      return *res*.status(400).json({ message: "Product not found" });

    }

*res*.status(200).json(product);

  } catch (error) {

*res*.status(400).json({ message: "Error updating product", error });

  }

});

export *const* getAllProducts = asyncHandler(async (*req*, *res*) *=>* {

  try {

*const* products = await Product.find();

*res*.status(200).json(products);

  } catch (error) {

*res*.status(400).json({ message: "Error fetching products", error });

  }

});

export *const* getProductById = asyncHandler(async (*req*, *res*) *=>* {

  try {

*const* { productId } = *req*.params;

*const* product = await Product.findById(productId);

    if (!product) {

      return *res*.status(400).json({ message: "Product not found" });

    }

*res*.status(200).json(product);

  } catch (error) {

*res*.status(400).json({ message: "Error fetching product", error });

  }

});

export *const* deleteProduct = asyncHandler(async (*req*, *res*) *=>* {

  try {

*const* { productId } = *req*.params;

*const* result = await Product.deleteOne({ \_id: productId });

    if (result.deletedCount == 0) {

      return *res*.status(400).json({ message: "Product not found" });

    }

*res*.status(200).json({ message: "Product deleted successfully" });

  } catch (error) {

*res*.status(400).json({ message: "Error deleting product", error });

  }

});

STEP 28: create a file name “upload.js” under the middlewares forlder

Add this code block:

import multer from "multer";

import path from "path";

// Define where the images will be stored and their file naming convention

*const* storage = multer.diskStorage({

  destination: (*req*, *file*, cb) *=>* {

    cb(null, "uploads/"); // Specify the 'uploads' folder for storing images

  },

  filename: (*req*, *file*, cb) *=>* {

    cb(null, `${*Date*.now()}\_${*file*.originalname}`); // Use timestamp to prevent filename collisions

  },

});

// File filter for image files only

*const* fileFilter = (*req*, *file*, *cb*) *=>* {

*const* allowedTypes = /jpeg|jpg|png|gif/;

*const* extname = allowedTypes.test(

    path.extname(*file*.originalname).toLowerCase()

  );

*const* mimetype = allowedTypes.test(*file*.mimetype);

  if (extname && mimetype) {

    return cb(null, true);

  } else {

    cb(new *Error*("Only image files are allowed"), false);

  }

};

*const* upload = multer({

  storage: storage,

  fileFilter: fileFilter,

});

export default upload;

STEP 29:

Create a file named “productRoute.js” under the “routes” folder.



Add this code block:

import express from "express";

import {

  createProduct,

  getAllProducts,

  getProductById,

  updateProduct,

  deleteProduct,

} from "../controllers/productController.js";

import upload from "../middlewares/upload.js"; // Import multer configuration

*const* router = express.Router();

// Route to create a new product with image upload

router.route("/").post(upload.single("image"), createProduct);

// Route to fetch all products

router.route("/").get(getAllProducts);

// Route to fetch, update or delete a single product by ID

router

  .route("/:productId")

  .get(getProductById)

  .put(upload.single("image"), updateProduct) // Handle image upload during update

  .delete(deleteProduct);

export default router;

STEP 30:

Click the “server.js”, and then add this statement

import path from "path";

import { fileURLToPath } from "url";

import productRoutes from "./routes/productRoute.js"

app.use("/api/products", productRoutes);

// Serve images from the "uploads" folder

*const* \_\_filename = fileURLToPath(import.meta.url); // Get the current file URL

*const* \_\_dirname = path.dirname(\_\_filename); // Get the directory name

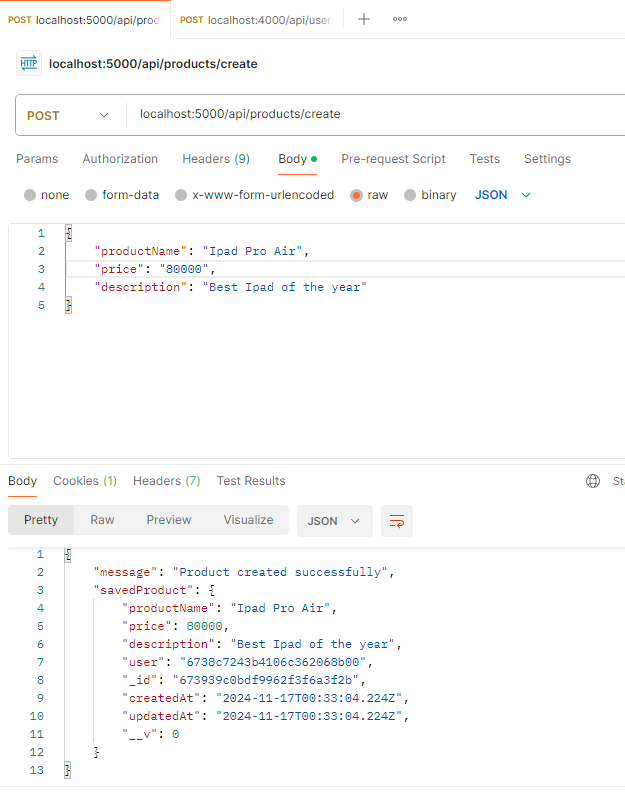
app.use("/uploads", express.static(path.join(\_\_dirname, "uploads")));



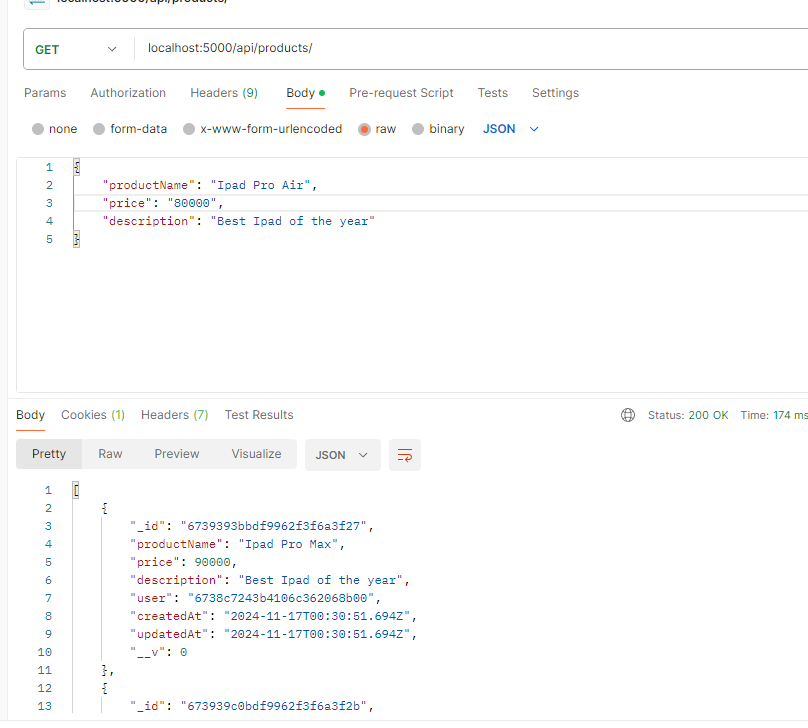
STEP 31:

Open Postman, and TEST the APIs

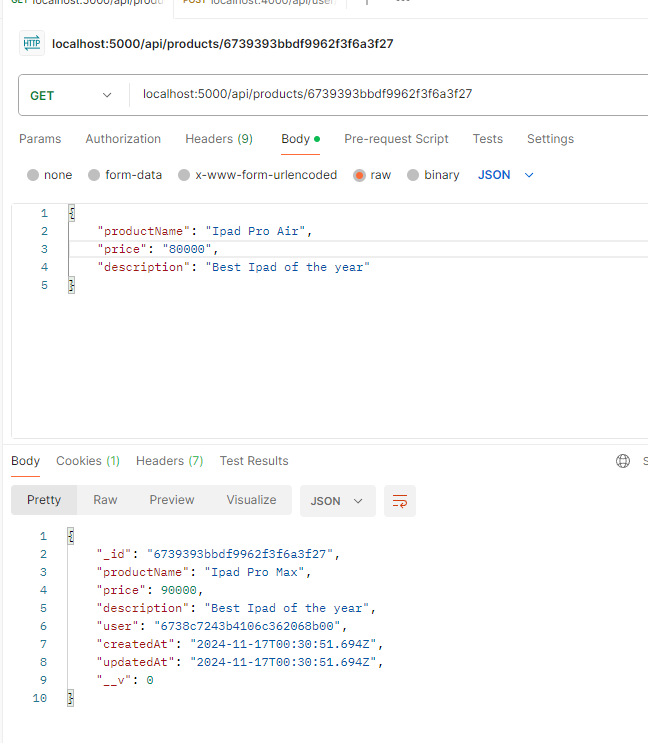
“/create”



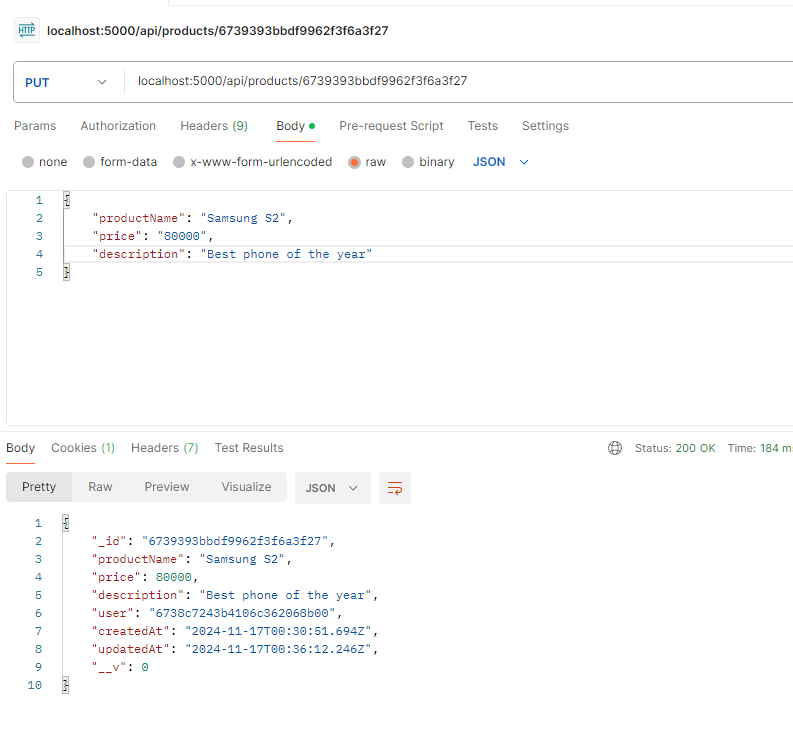
Get all products - “/”



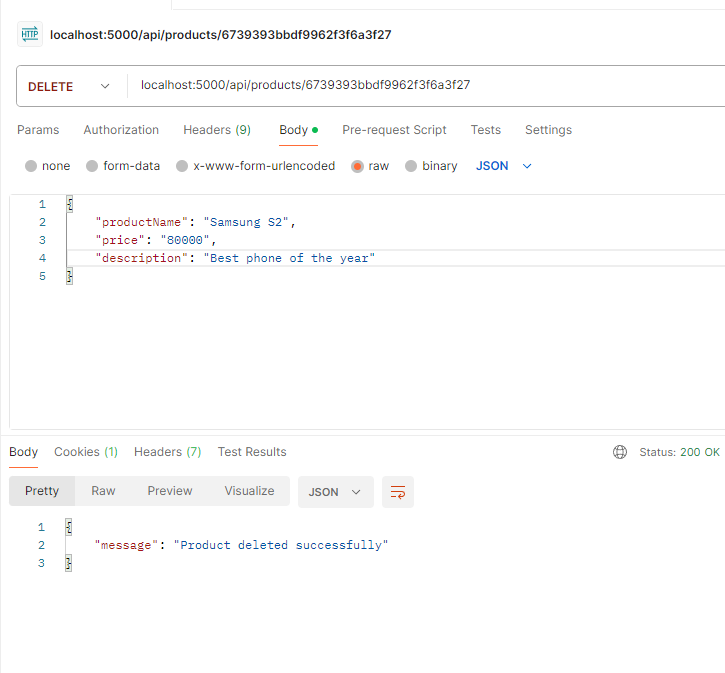
Get product by ID – “/:productId”



Update product by Id – “/productId"



Delete product by Id – “/productId"



CONGRATULATIONS YOU HAVE FINISHED THE BACKEND PART. !!! JAOY PA FRONTEND