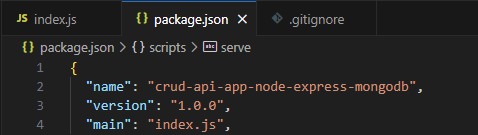
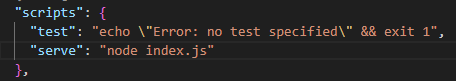
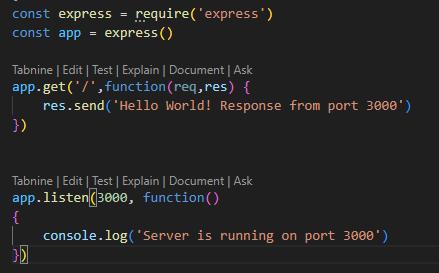
1. In main folder write **npm init –y**
2. Package.json file will be created with name of app as the parent application



1. Write the serve: “node index.js” in the scripts portion of package.json

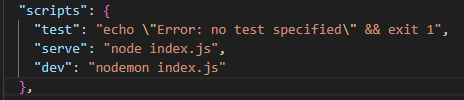


1. Install express server to serve the application as a web application. Go to npm site, search for express and copy the relevant code:



App.listen launches server on port 3000, app.get launches a response for an API route / (Home) and gives a response back when localhost:3000 is accessed which Is a **get** request.

1. Install nodemon. Nodemon is to automatically monitor for changes in the file, so user has don’t have to re-run the code. Add nodemon config in package.json under scripts as:



1. Go to mongodb, create account, create new project, assign it a name choose the free server class
2. Assign 0.0.0.0 to accesss the database from anywhere
3. Create username and password
4. Install mongodb and mongoose using npm
5. Import mongoose in index.js as:

const mongoose = require('mongoose')

1. Connect to mongodb using connection string and using mongoose in between:

mongoose.connect('

mongodb+srv://admin:<db\_password>@cluster0.lstf4.mongodb.net/?retryWrites=true&w=majority&appName=Cluster0')

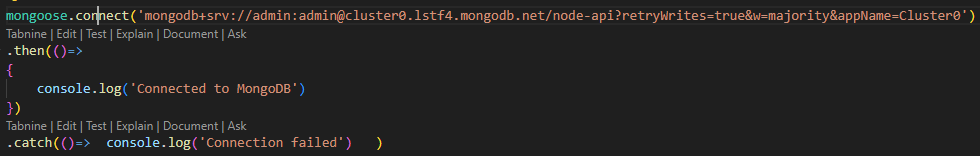
here replace the password with the password we have setup, also before “?retry” enter a custom name for the user to access the endpoint. We will use node-api

1. The connection string is:

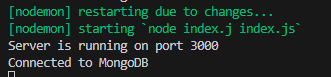
mongoose.connect(

mongodb+srv://admin:admin@cluster0.lstf4.mongodb.net/node-api?retryWrites=true&w=majority&appName=Cluster0')

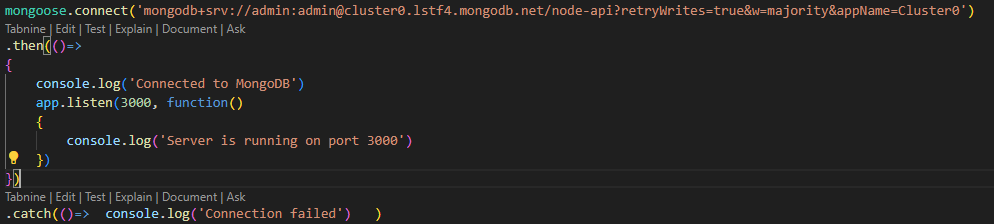
1. Next insert then and catch with it:



1. We can see the server responses as:



1. Ideal scenario is to first check for database and then run the web application, this can be done by running express after database is connected i.e in the then block:

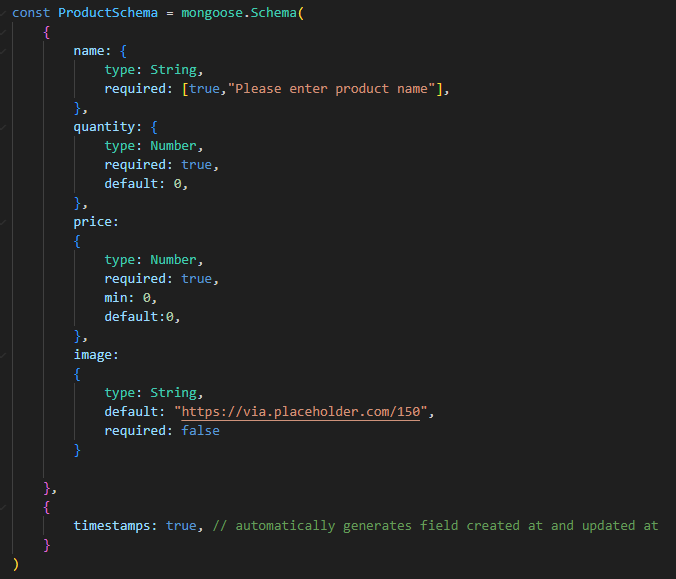


1. Create a folder name models and create a java file named as products.model.js. Here we will define our model:



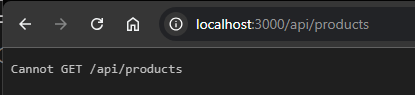
1. Within the file import mongoose, define the Product schema and then export it:





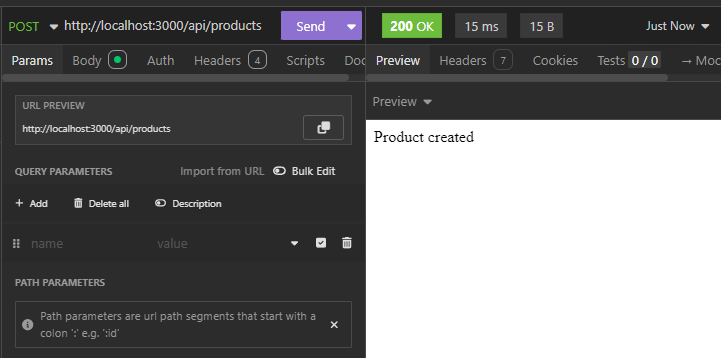


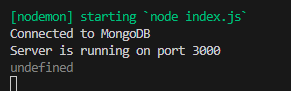
1. Define a post request as api/products in index.js:



We get an error, because the requested url is supposed to be for post request not for get request.

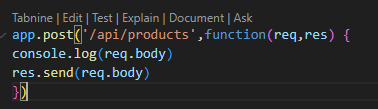
1. Define a new post reqest in insomnia and paste the link and send the request:

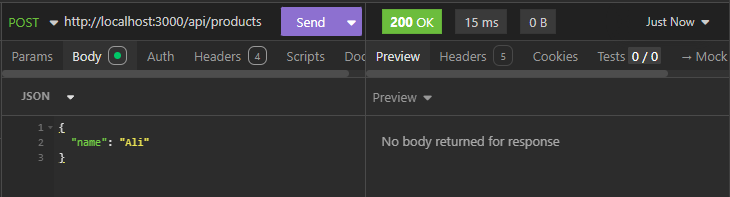




The server displays that undefined data is displayed.

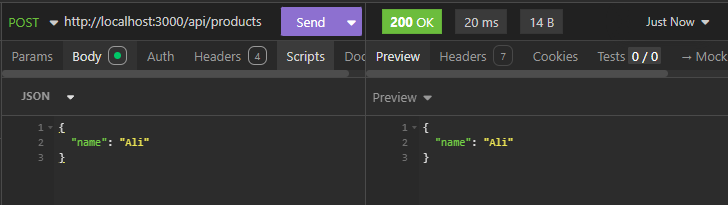
Let’s change the code to also display the req.body on the client side, that is on the browser:





No body received. Issue is we can’t use JSON directly, we have to use a middleware of express server as shown:





**Server log:  
  
**

1. Import Product as:

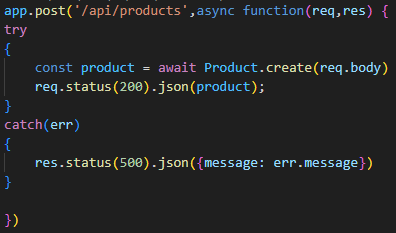
****

1. Let us actually save the fields in the model instead of just creating it:

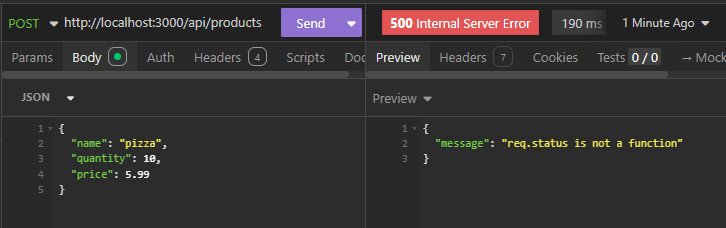
Code: **await Product.create(req.body)**

Whenever await is used the **async** keyword is attached with the function

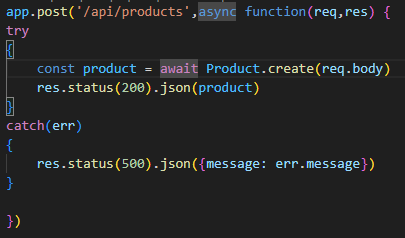
1. The part **await Product.create(req.body)** is assigned to a const product to store it’s result. In the next line the status 200 is sent and the product just created is displayed on the browser:



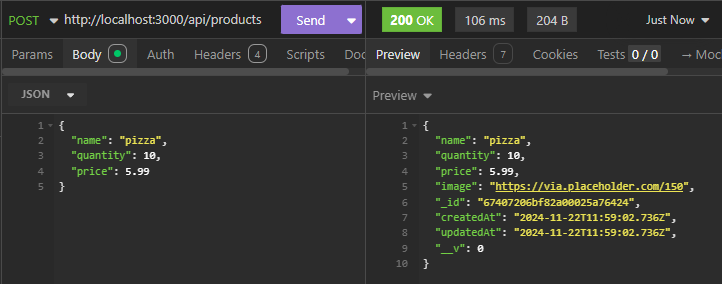
**Error:**

****

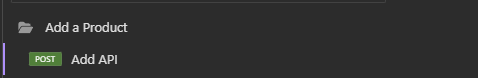
This is because instead of res we used req.status in above code. Let’s fix this and modified code is:

****

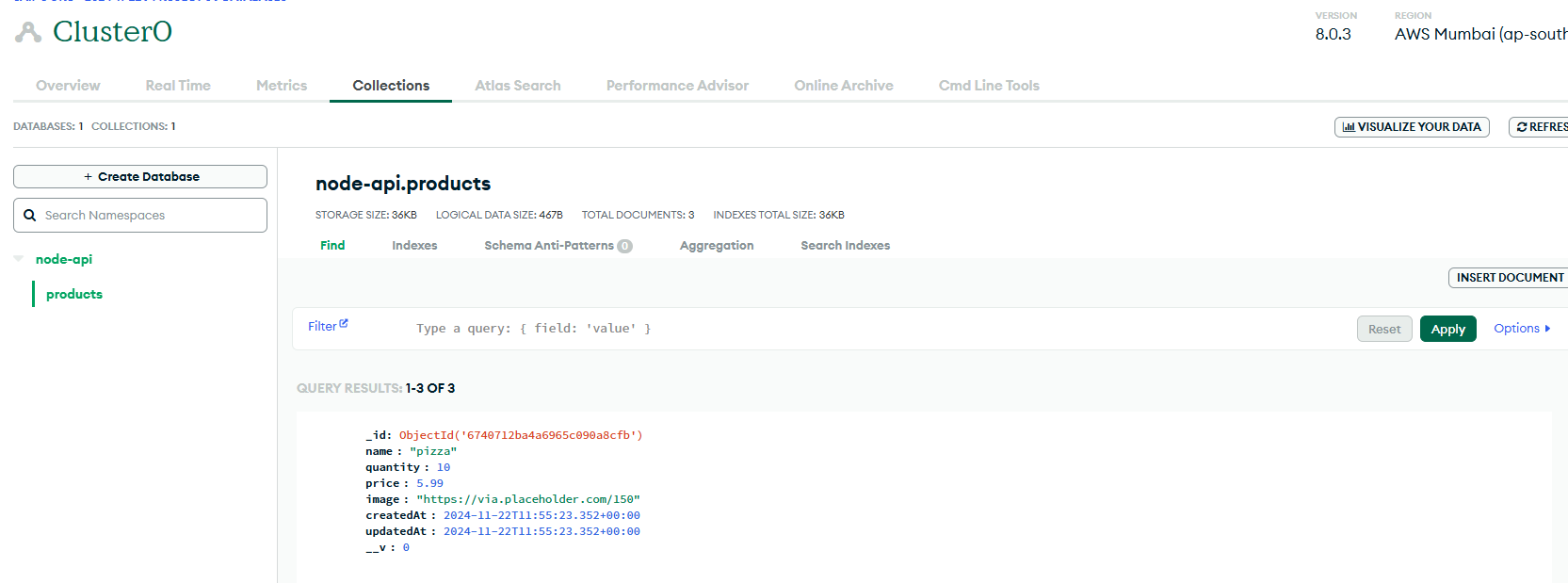
**Result:**

****

We saved this API route in another folder in insomnia as:



We only added three fields in JSON, because these were the required ones, now let’s verify if it exsists in the MONGODB.



1. Lets crete for read:

Copy same code, just replace product.create with product.find with empty parenthesis. You got the code for listing the producsts

