**Chapter 1  
EXPRESS API**

**-: Content :-**

* INTRODUCTION OF EXPRESS API
* PACKAGES WITH INSTALL COMMAND
* CREATE JS FILES AND MAKE CONNECTION
* USE POSTMAN FOR SEND REQUEST OR FILE
* USE INSOMNIA FOR SEND REQUEST OR FILE
* DIBUGGING

**1.INTRODUCTION OF EXPRESS API**

**EXPRESS API** is a web API that uses the **RESTful approach.** **REST API** is an API that is use **resource-based interfaces**. In EXPRESS API that use web resources that’s is formatted in JSON or XML file that accessed via HTTP represented by URL (Path).

**2. Install package and create application**

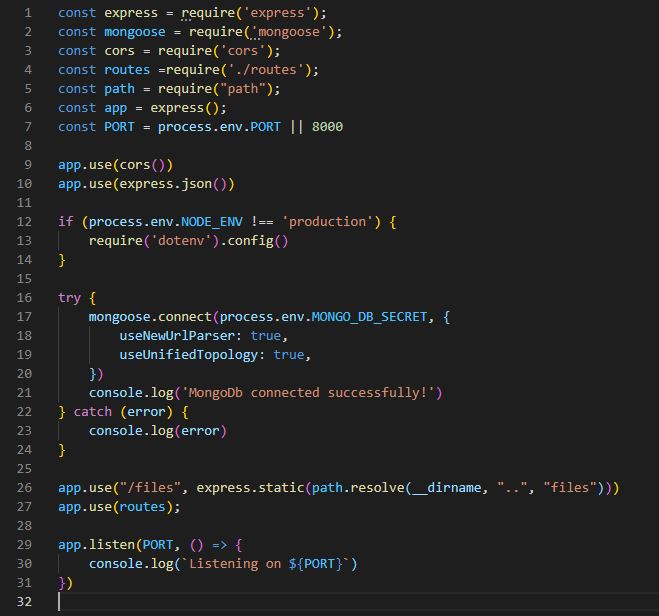
* Create a folder as a name like registration
* Open the folder using VS Code editors
* Open terminal of VS CODE Editors using this steps: - **VS Code Manu > Terminal > New Terminal**.
* Go to Terminal and use npm command to create the application.
* Use npm command to initialize the application.

**Command: - npm init**

* After that follow the step and give information like Author, Name, Description, Script etc.
* After pressing **ENTER** key,it will ask you follow things and at last it will ask you **“Is this OK? Write yes”.**
* install node module that are use to create application using **npm install or npm i** command
* **Express: -** Express is a framework in express js .its provided various futures on web application. For install express use **npm i express.**
* **Mongoose –** mongoose provided database access for the local web site. It’s provided a schema that’s connecting to mongo DB database. For install mongoose use **npm i mongoose**
* **Multer-** it is use in node JS middleware multipart or form data which is primary used. Its function that received the request and send response object when user create any type of request. For install multer package we use the command **npm i multer.**
* **Dotenv: -** Dotenv is a file that’s used for environment variable from .env file. For install we use the command **npm i dotenv**.
* **Bcrypt**: - Bcrypt is use for hash user password in the database .Its store the plain text with encrypted hash password method. For install we use **npm i bcrypt**.

**3. CREATE JS FILES AND MAKE CONNECTION**

**Create Server.js fle**



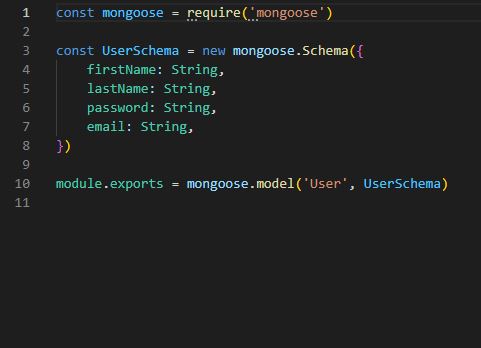
* **Explanation: -** In the server.js file we are use as a server file where we are make our connection. In this file we are deploy our application in PORT 8000 .In this file we are connect to our mongodb server with local. When we give URL **localhost:/8000** that time it’s gave some result.

To make secure MongoDB connection, we create new file **“.env”** . In this file we give the URL path of connection.

MONGO\_DB\_SECRET =mongodb+srv://{username}:{password} @cluster0.1v2oe.mongodb.net/DbName?retryWrites=true&w=majority

Make one folder in src named **"MODELS"**. The data model is a JS module that connects to the database and exports some functions that let us operate on the data. The model is defined in a file called **models/user.js** , inside our project folder. One of the functions exported by the data model is the find function.

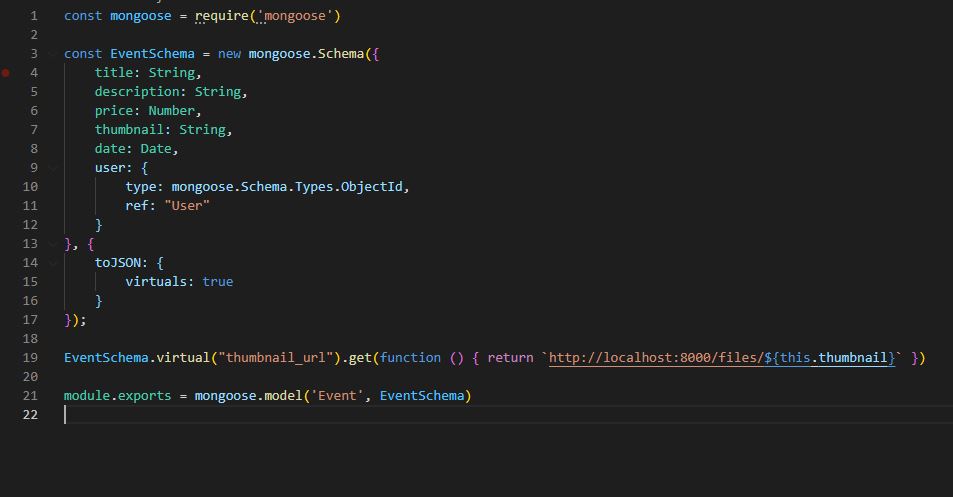
**Create user.js File**

****

* **Explanation**: - In the models folder we create User.js file. In this file we are deploy mongoose that is use for create schema in mongodb server. After that in this file we create our user schema like email firstName lastName. After that we are export that other side.

Create another file in **“models/event.js”.**

**Create event.js**

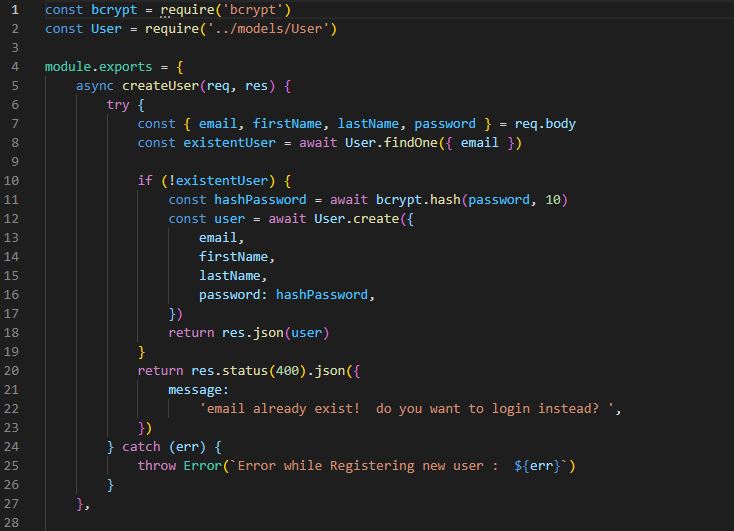
****

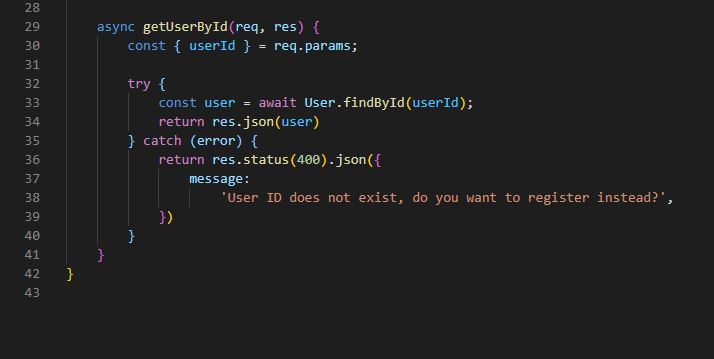
* **Explanation**: - In the Models folder another file name event.js. In this file we are deploy mongoose for create event schema in the local we create local schema attributes like title ,date, thumbnail etc and all thous are take in strings. After that we are export that makes for event schema.

Make new folder **“controller”** in src .This will house all the controllers needed for the application. These controller methods get the request from the routes and convert them to HTTP responses with the use of any middleware as necessary.

In controller folder, add another file **“UserController”.**

**Create Src > Controller > UserController.js**

****

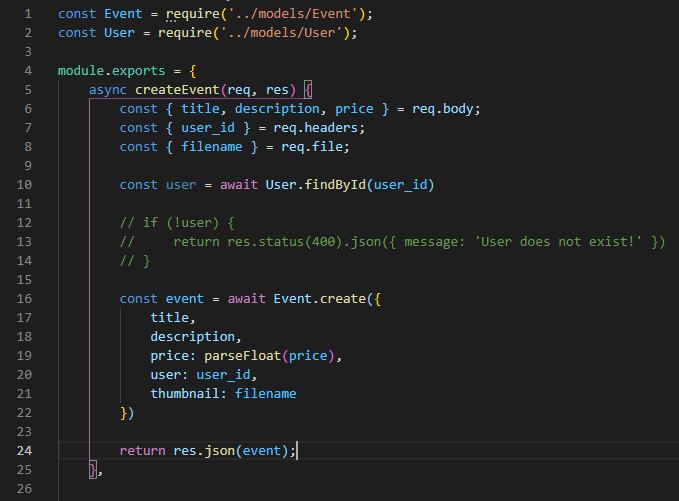


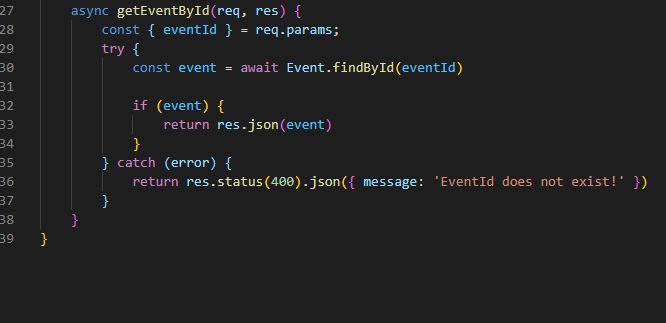
* **Explanation**: - In the UserController.js file we are control or call the user sub file. First we export the module and fetch that user is exist on the database or not if exist then show a massage user is exist and if not exist then create a new user using user schema.

After that we use **getUserById () method** and use the User\_id and fetch the user is exist on database or not if exist it show the result and not then 400 error massage show in console log.

In controller folder, add another file **“EventController”.**

**Create Src > Controller >EventController.js**

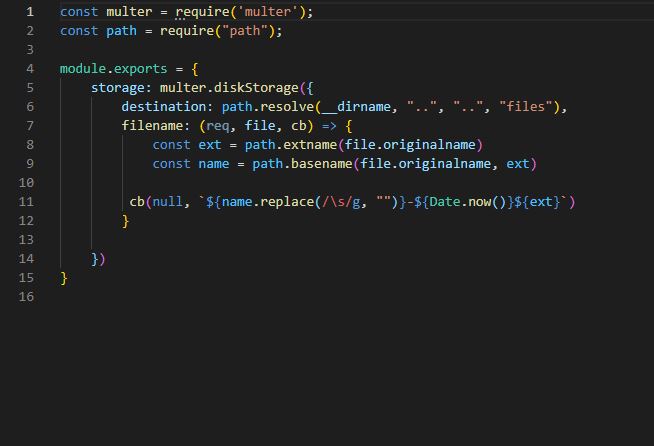




* **Explanation**: - In the Eventcontroler.js file we are control or call event sub file. First we create event to receive all the information from the request. Then we apply **findById() method** to create user and check it using User\_id.If the user is exist then it create an event or if not exist then it show a massage user is not exist.

Create folder in src name it as **“config”.** In this folder create one new file **“upload.js”.**

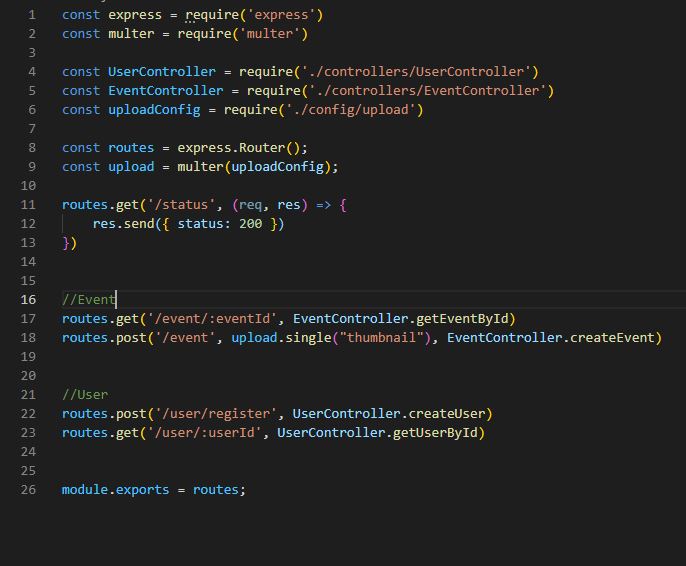
**Create Src > config > upload.js**

****

* **Explanation**:- In the upload.js file we construct the file path. First we construct the path and then storage in local and make and URL. After that it’s back to controller with file extension. If that file had any rejection then its show error massage using controller file.

Now make one new file in src and named it as**"routes.js"**. We use it as middleware to allow import the path of different files.

**Create Src > routes.js**



* **Explanation:-** In the routes.js file we are connected all the middleware that are routes of the main file . And also create the user and event routes that we call through an API.

**4.POSTMAN**

**Postman** is a scalable API testing tool that quickly integrates into CI/CD pipeline. API stands for Application Programming Interface which allows software applications to communicate with each other via API calls.

## How to Download and Install POSTMAN

**Step 1)** Download Postman.

Go to <https://www.postman.com/downloads/> and choose your desired platform among Mac, Windows or Linux. Click Download.

**Step 2)** Click on Run  
Your download is in progress message should now display on the Apps page. Once the Postman download is completed, click on Run.

**Step 3)**Postman Installation Start. Wait for some time to complete the Installation of Postman.

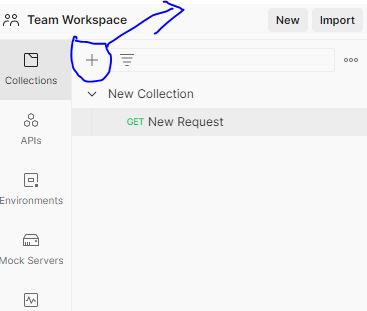
**Step 4)** Signup for Postman Account. In the next window, Signup for a Postman Account. There are two ways to sign up for a Postman account. One is to create an own Postman account, and the other is to use a Google account. Though Postman allows users to use the tool without logging in, signing up ensures that your collection is saved and can be accessed for later use.

**Step 5)**Click on Save My Preferences. Select the workspace tools you need and click Save My Preferences

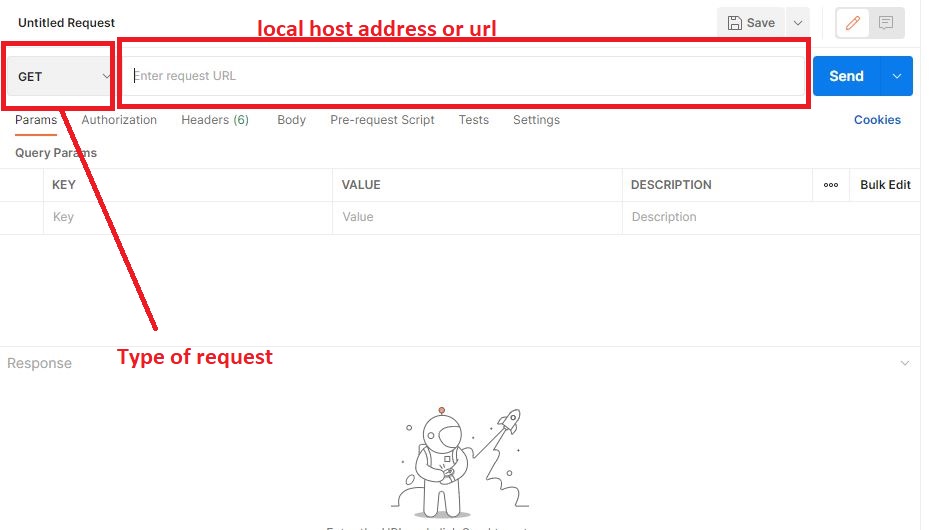
**Step 6)** You will see the Startup Screen

**USE POSTMAN TO CHECK REQUEST STEPS**

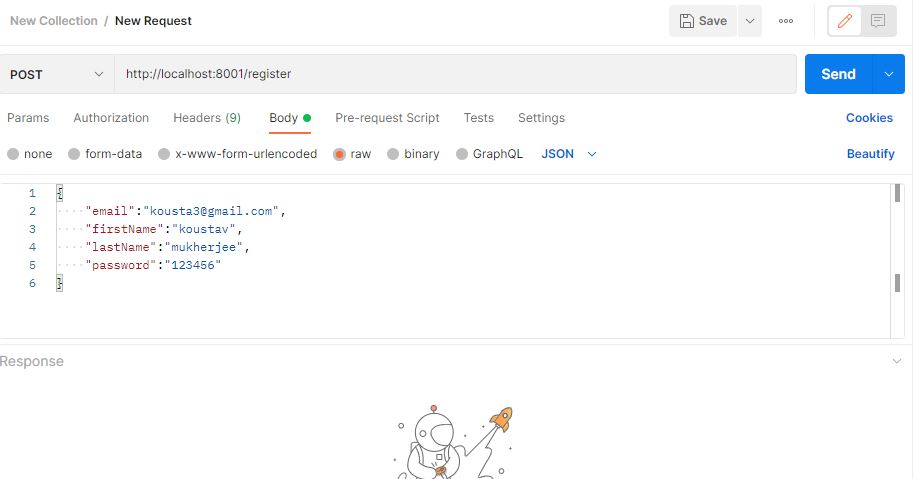
* **Step 1 -> go to create a new collection(+) and create a request**

****

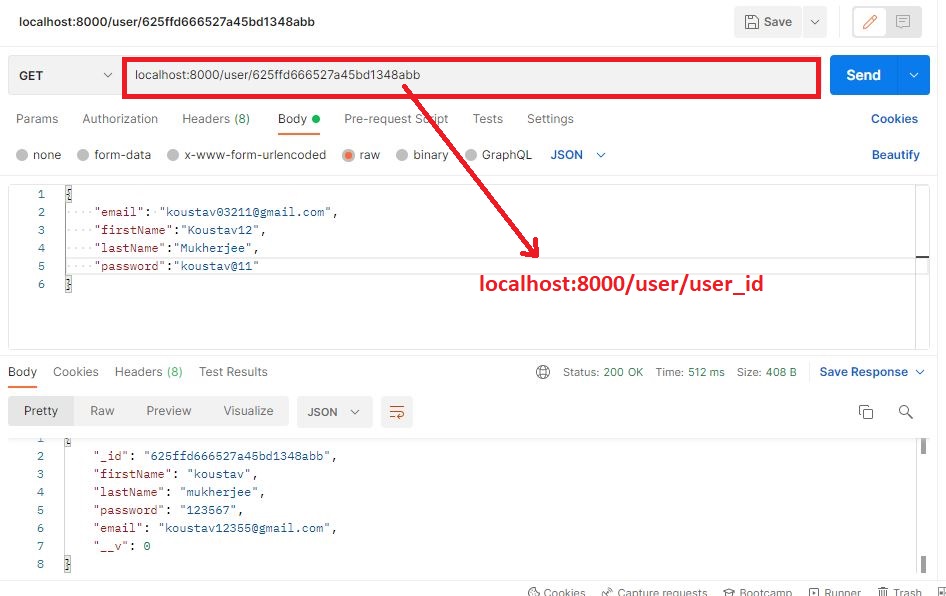
* **Step 2 -> setup ulr and method (post,get etc)**

****

* **Step 3 -> go to body and select and formdata for form or row like JSON format and fill data**

****

* **STEP 4 -> After complite data fill send .**

****

**5.INSOMNIA**

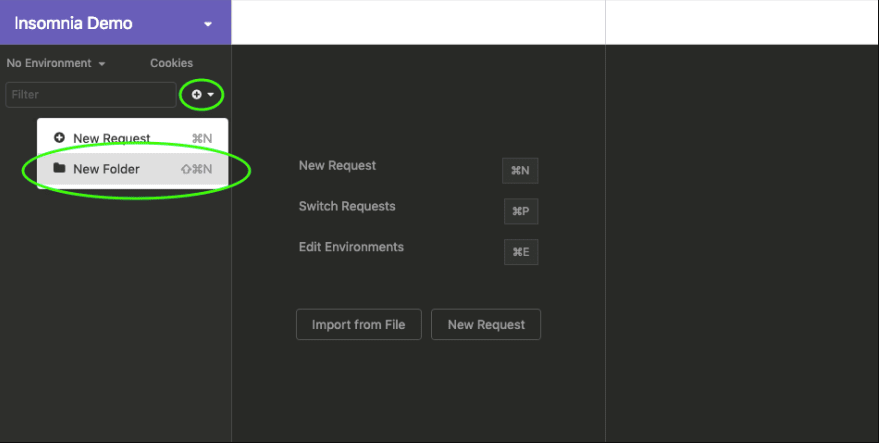
It's a free cross-platform desktop framework that incorporates a user-friendly user interface and sophisticated features, such as security helpers, code creation, and environment variables.

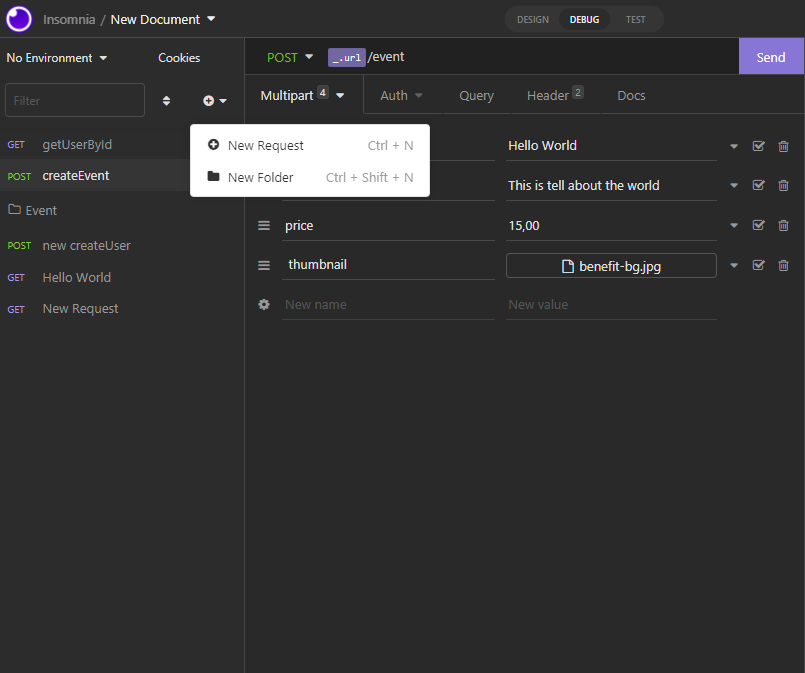
**Steps to use INSOMINA**

**Step 1: Install Insomnia -** Head over to the [official Insomnia website](https://insomnia.rest/) to install it on your machine. There are directions to prompt you once you're there.

**Step 2:** Check your API Docs - Once you're more familiar with the functionality of your API, go ahead and open up Insomnia.

**Step 3:** Make your 1st HTTP GET Request w/ Insomnia - Once the app is open, you'll see a mostly "blank" workspace. To begin making requests click the plus icon, which will trigger a dropdown giving you the option to either make a New Request or create a New Folder. Let's make a folder, specific to our project.

Name the folder however you'd like. I usually use the file name. Name the folder however you'd like. I usually use the project name.



### You can see in the top bar that there's a place to select the type of request you want to make, an input for the URL/endpoint, and a SEND button.

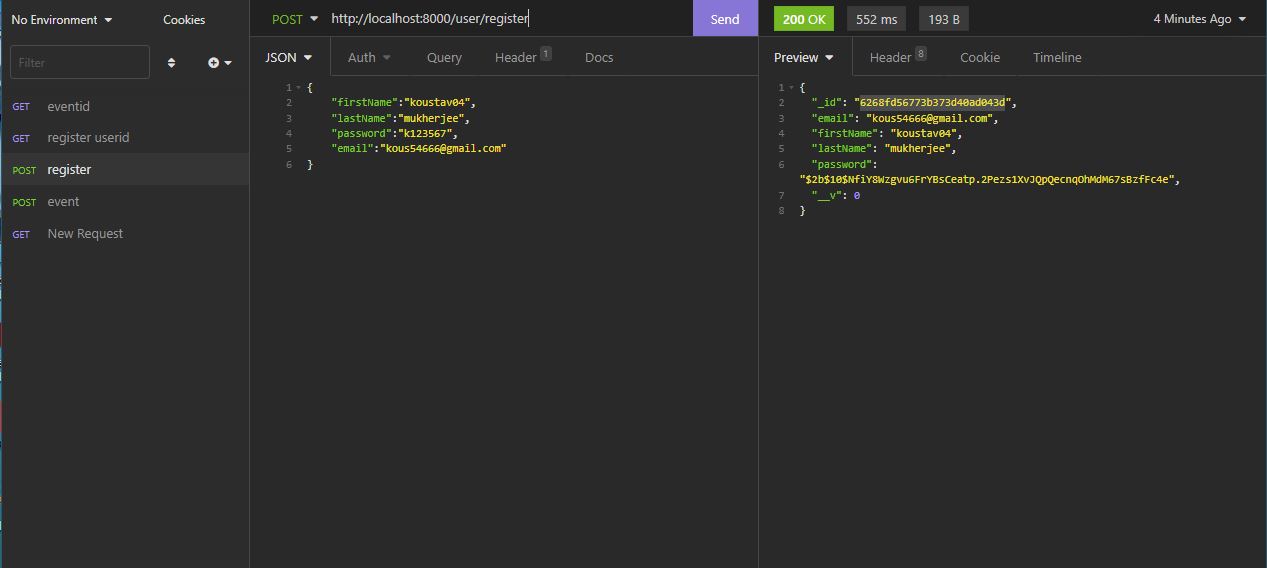
### 

### I want to make a GET request, so that's what I'll select. Then I'll put the base URL followed by the endpoint in the input .This request won't have a body, header, or anything else, so I can go ahead and hit SEND.

### 

You can see the status of your request (here it was 200, which means it was successful), the time it took to complete, when it was placed (which has a dropdown to view/select from all instances of the request being made), and the response (which has a WHOLE LOTTA options, itself.

### **Step 4:** Make a POST Request - I need to add the URL/endpoint and the body. Once I complete all required fields for my request, I can hit SEND.]

****

**6. DEBUGGING**

**The Visual Studio Code editor has all the tools to debug Node.js applications effectively. Its built-in debugger can debug any application that targets the Node.js runtime, even if the source code for the application is a language that transpires to JavaScript, such as TypeScript.**

**The easiest way to start a debugging session in Visual Studio Code is to open a file in the editor, click the Run View icon in the Activity Bar (or press Ctrl+Shift+D on your keyboard), followed by the Run and Debug button at the top left corner of the application.**

**The Visual Studio Code debugger will try to auto-detect the debug environment for your project, but if this fails, you will be prompted to select the appropriate environment; in this case, select Node.js.**

**After selecting an environment, the project launches and the debugger attaches to the process. You can see the output of your project in the DEBUG CONSOLE, and the debug toolbar appears at the top of the screen to step through the code, pause the script, or end the session. On the left-hand side of the editor, there are five panes titled VARIABLES, WATCH, CALL STACK, LOADED SCRIPTS, and BREAKPOINTS.**