**Day 45**





**“Web Development + Security”**

**Response, Request and Routers in Express:**

**What Are Requests and Responses?**

Every time a client (like your browser or frontend app) interacts with your Express server:

CLIENT → [HTTP REQUEST] → SERVER (Express)

SERVER → [HTTP RESPONSE] → CLIENT

**Request (req)**

It represents data coming into the server (from the client).

It can include:

* URL → /home, /about, etc.
* Method → GET, POST, PUT, DELETE
* Query parameters → ?id=10&name=Aditya
* Body data → JSON or form data in a POST request
* Headers, cookies, etc.

**Response (res)**

This is what the server sends back to the client — can be text, JSON, HTML, or status codes.

Common methods:

* res.send() → Send text, JSON, or HTML
* res.json() → Send JSON data
* res.status(code) → Set HTTP status code
* res.redirect() → Redirect to another URL

**Routers in Express (From Basics)**

When your app grows, you’ll have many routes — /, /login, /products, etc. Putting them all in one file makes it messy.  
Routers help organize routes into smaller, modular files.

**Summary**

| **Concept** | **Description** | **Example** |
| --- | --- | --- |
| **req** | Request object → contains info sent by client | req.query, req.params, req.body |
| **res** | Response object → used to send data back | res.send("Hello"), res.json({...}) |
| **Router** | Helps organize routes into smaller files | app.use("/users", userRoutes) |

**What Are HTTP Request Methods?**

When a client (browser, frontend, or app) sends data to your Express.js server, it does so using different HTTP methods — each representing a different kind of operation.

Think of it like:

🔹 GET = Ask for info  
🔹 POST = Send new info  
🔹 PUT = Update existing info  
🔹 DELETE = Remove info

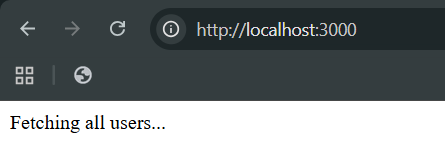
1. GET Request – can be checked by browser

**Purpose:** Used to retrieve data from the server. It does not modify anything — just reads data.

Example: of get request

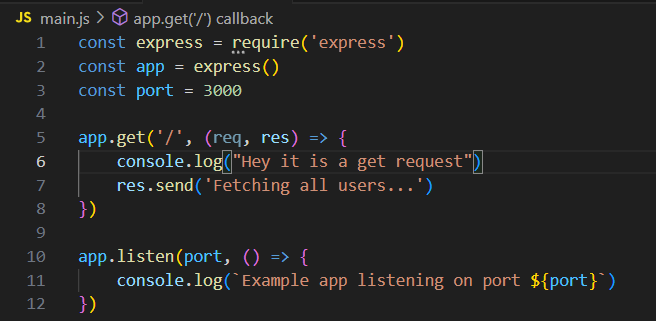


Output:

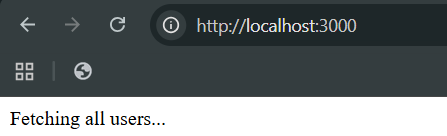


Now, printing something in the terminal we will use the console.log():

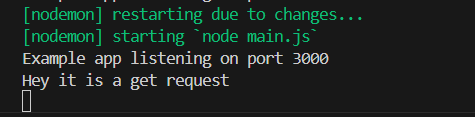
Code:



Output: Browser



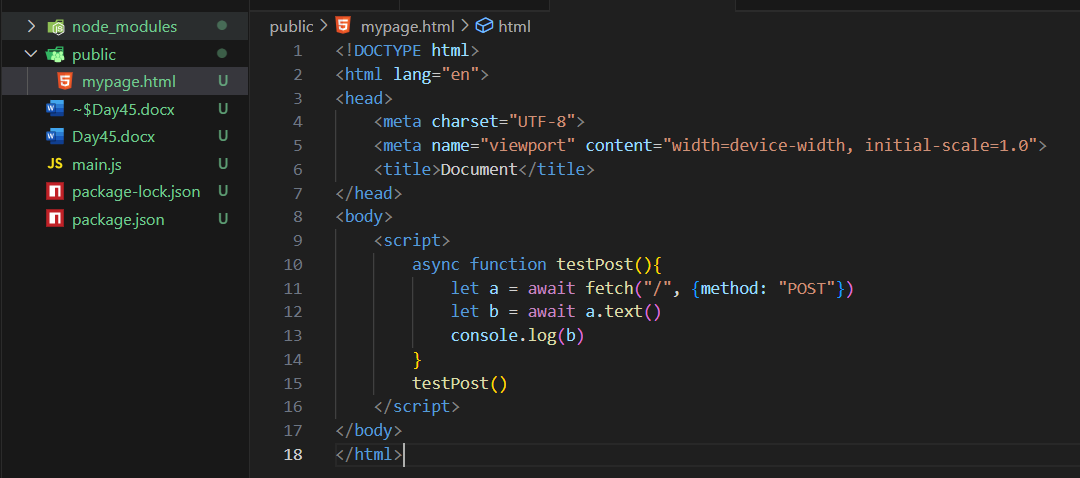
Output: terminal. Clearly, console.log prints on the terminal and res.send prints on the browser window.



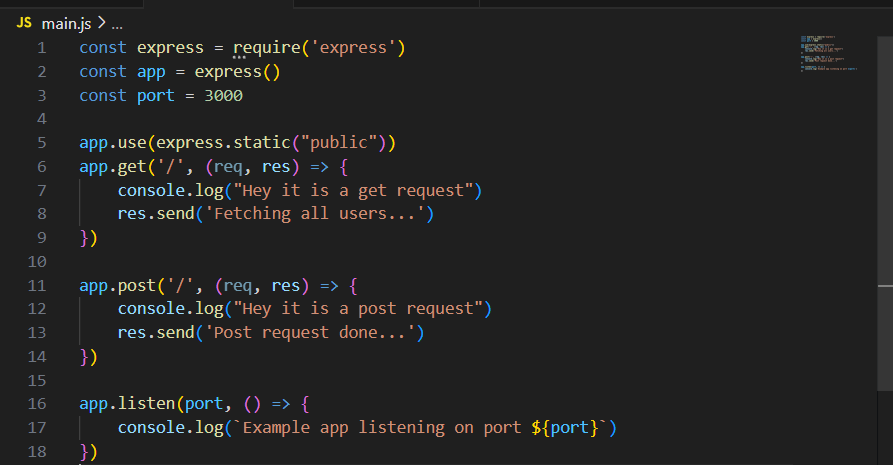
1. POST Request – can be checked when by creating a public folder

**Purpose:** Used to send or create new data on the server. Usually used with forms or JSON bodies.

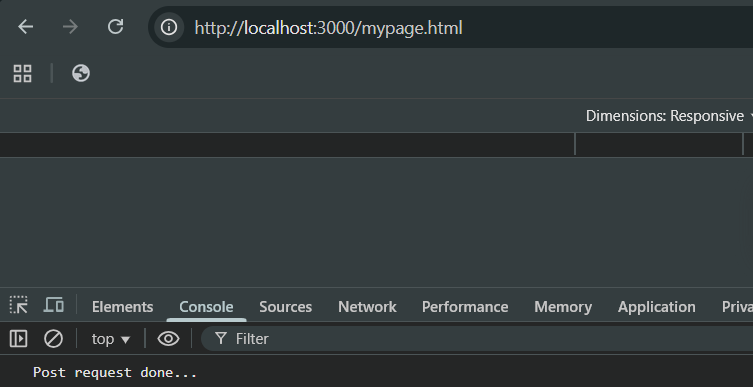
Example: of post request. For which we will be creating a public folder and creating mypage.html inside it.



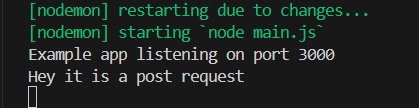
Also, we will be writing the post request code in the main.js:



Output: in console when we visit the mypage.html in the website via local host



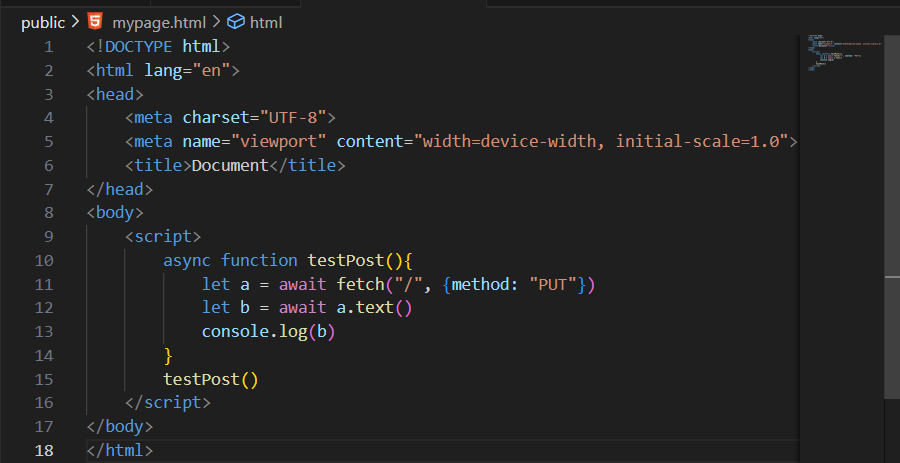
Terminal output:



1. **PUT Request – try to change the value we had given by post**

**Purpose:** Used to update or replace an existing resource completely.

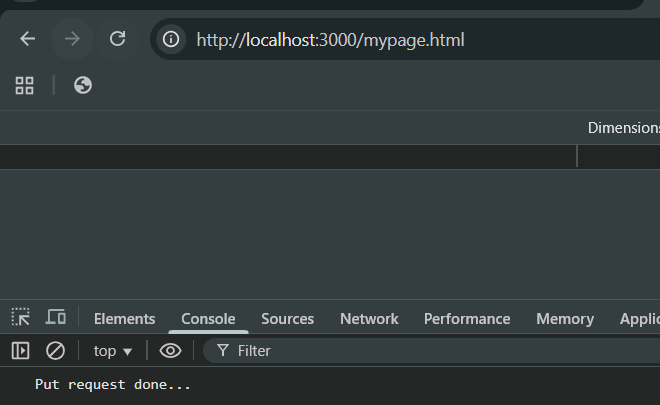
Mypage.html: we wrote “PUT” over the method name.



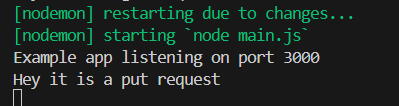
Main.js: we added put request



Output: console

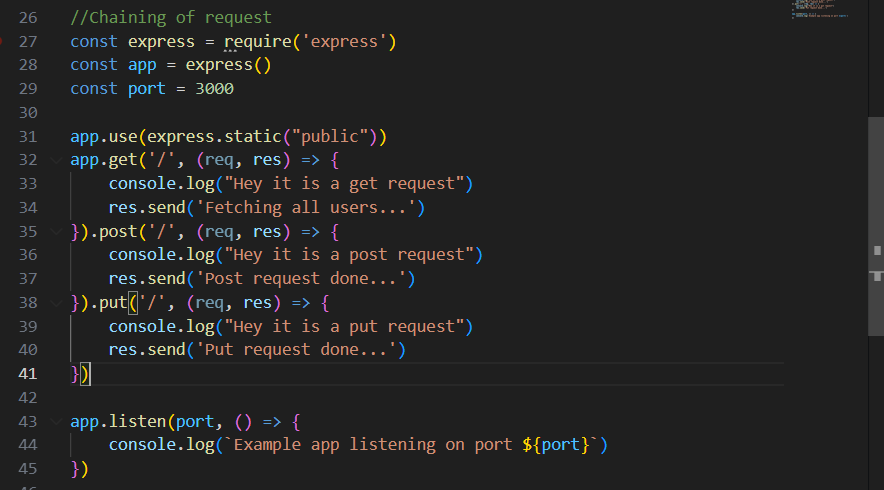


Output: terminal



Now, we can also make the chain of the request. It is basically the other way of writing the code we wrote till now:

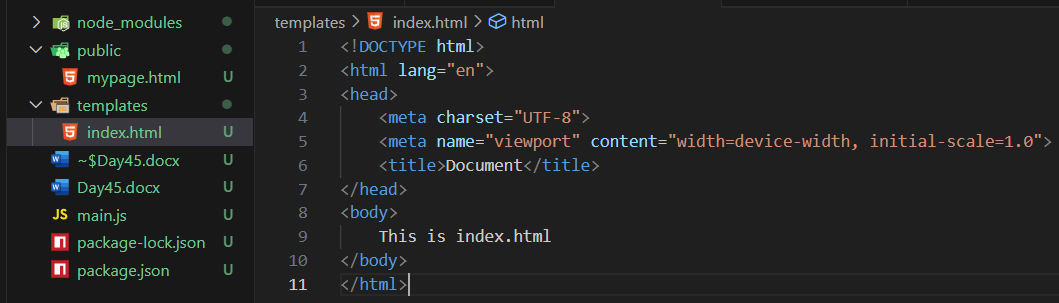
Main.js: we just remove “app” from each .post, .put and made a chain.



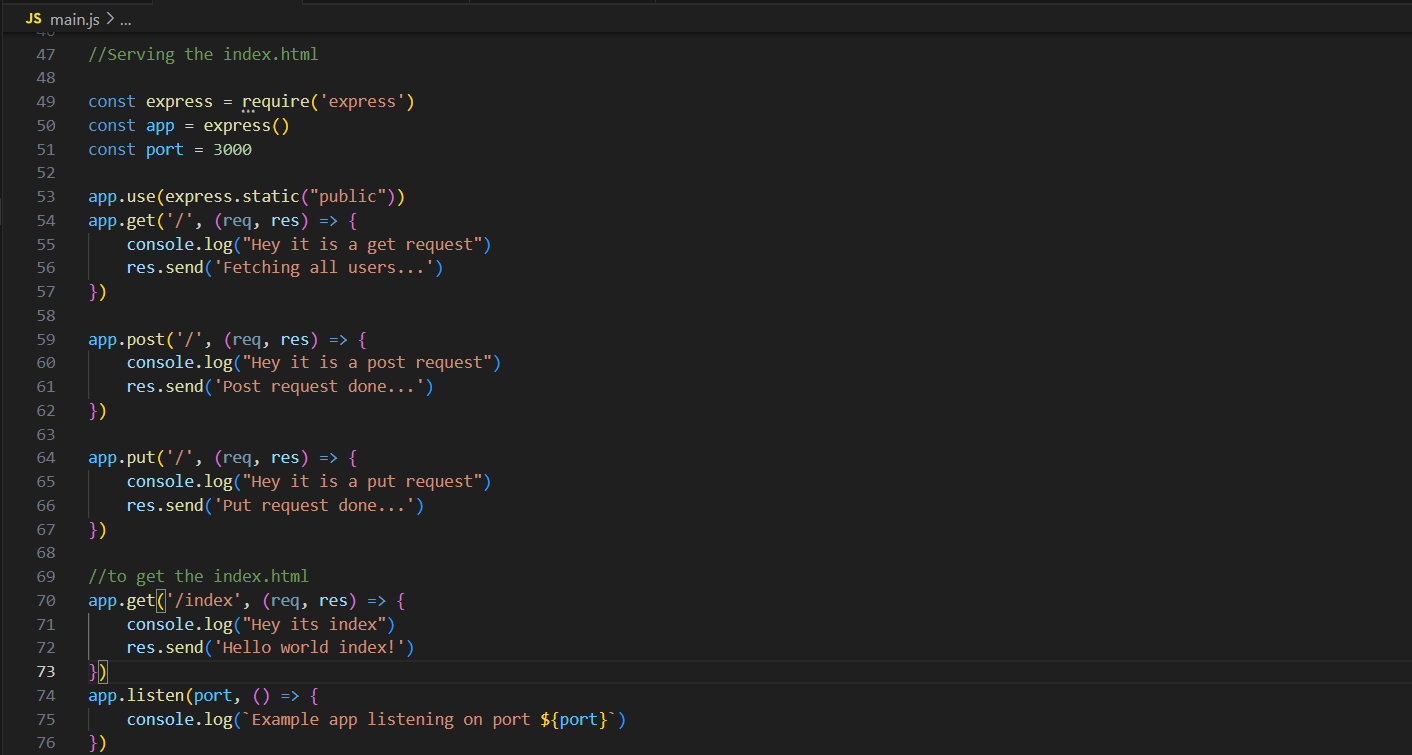
Now, suppose **we want to serve the HTML**:

Then to do so, we should first create a template folder, and inside it we will create the index.html.

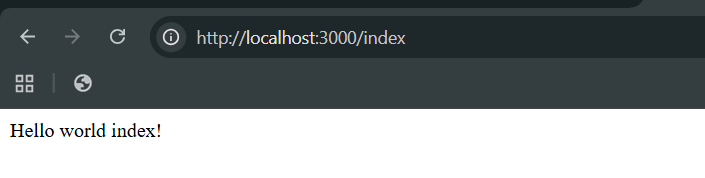
Index.html:



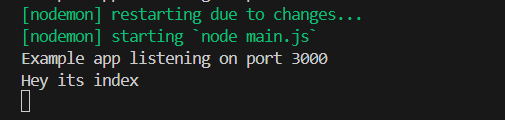
Main.js:



Output: at browser when we visited index

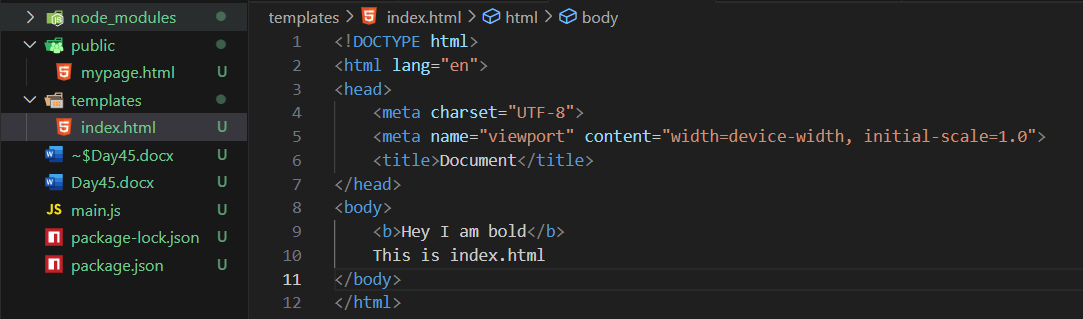


Output: console



Now, suppose we want to render the HTML properly, then we will use .sendFile as shown below:

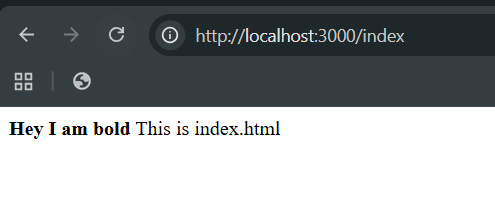
Index.html:



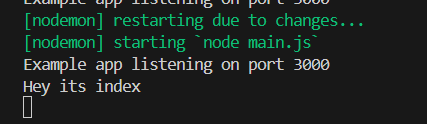
Main.js:



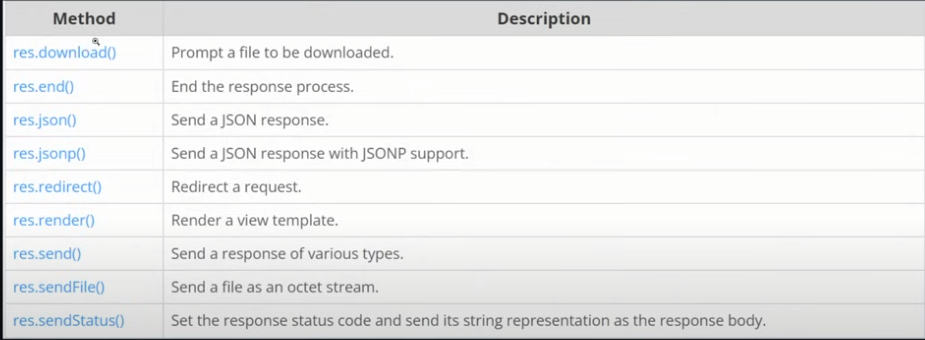
Output: browser

****

Output: terminal

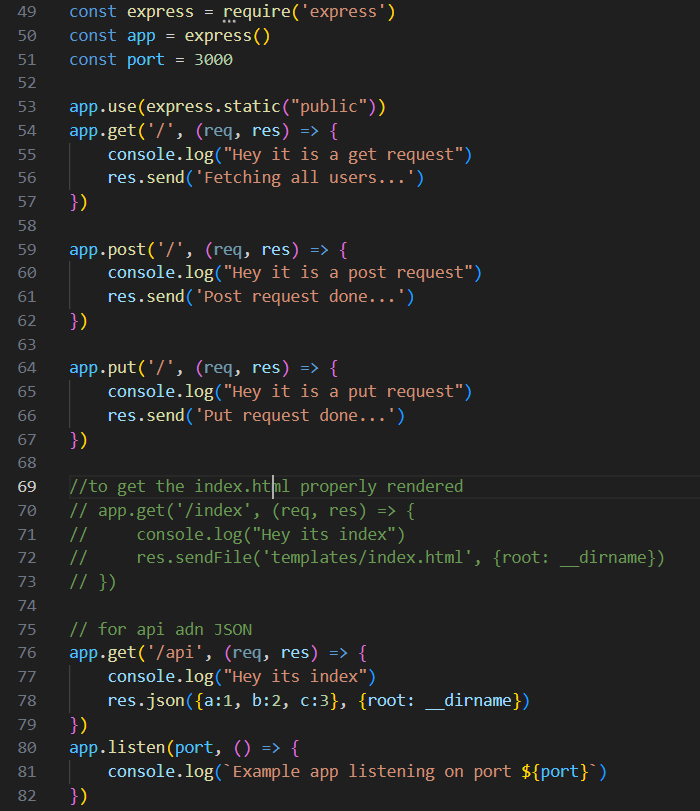


Also,

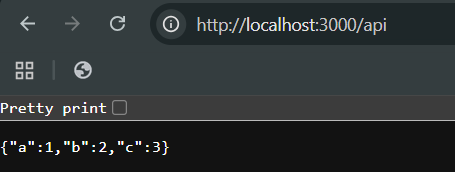


Now, suppose we want that “/api” should return the JSON.

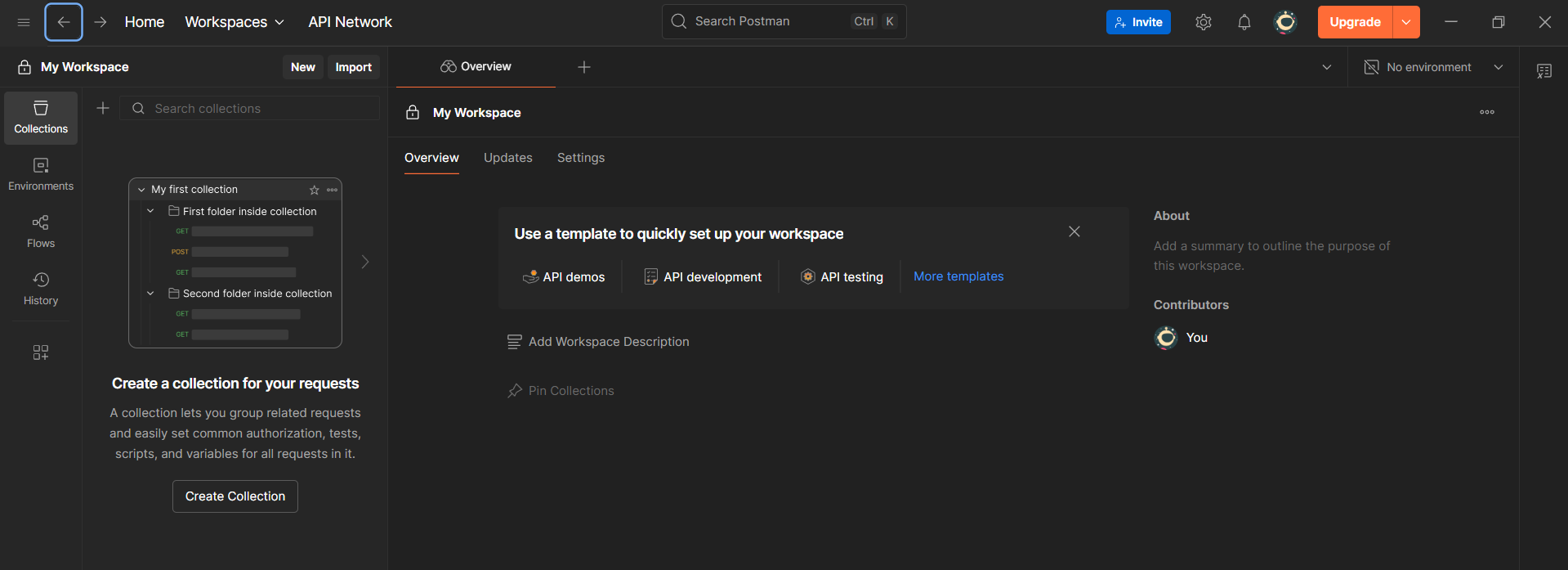
Main.js:



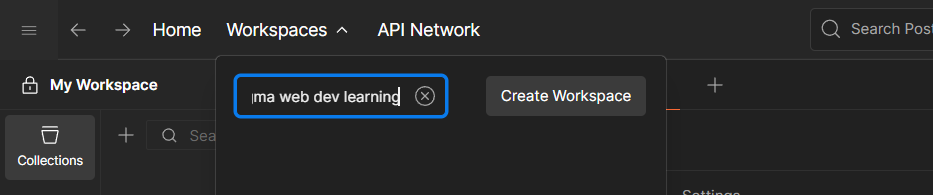
Output: browser



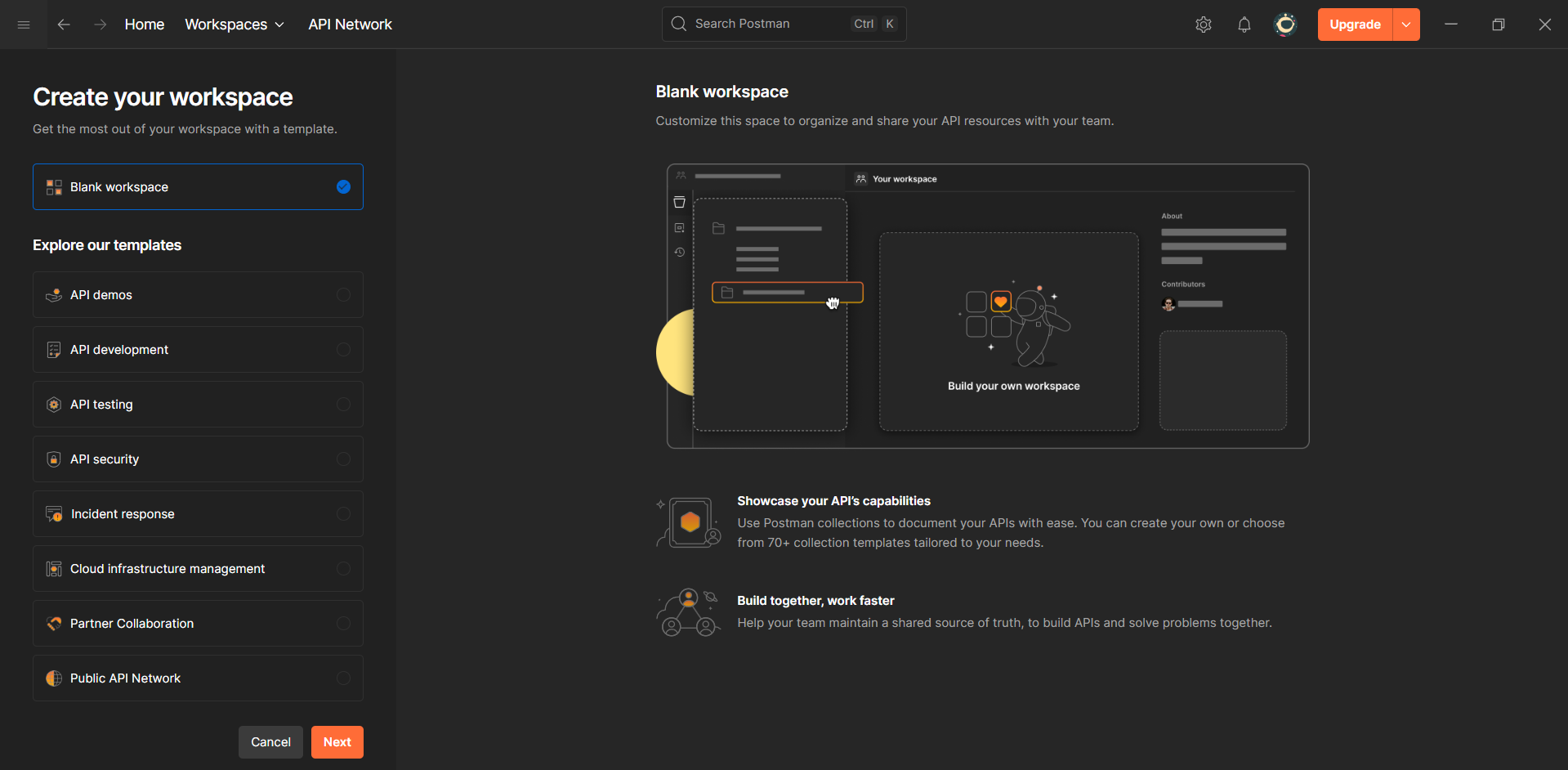
Now, postman:

Creating a new workspace in the Postman:  


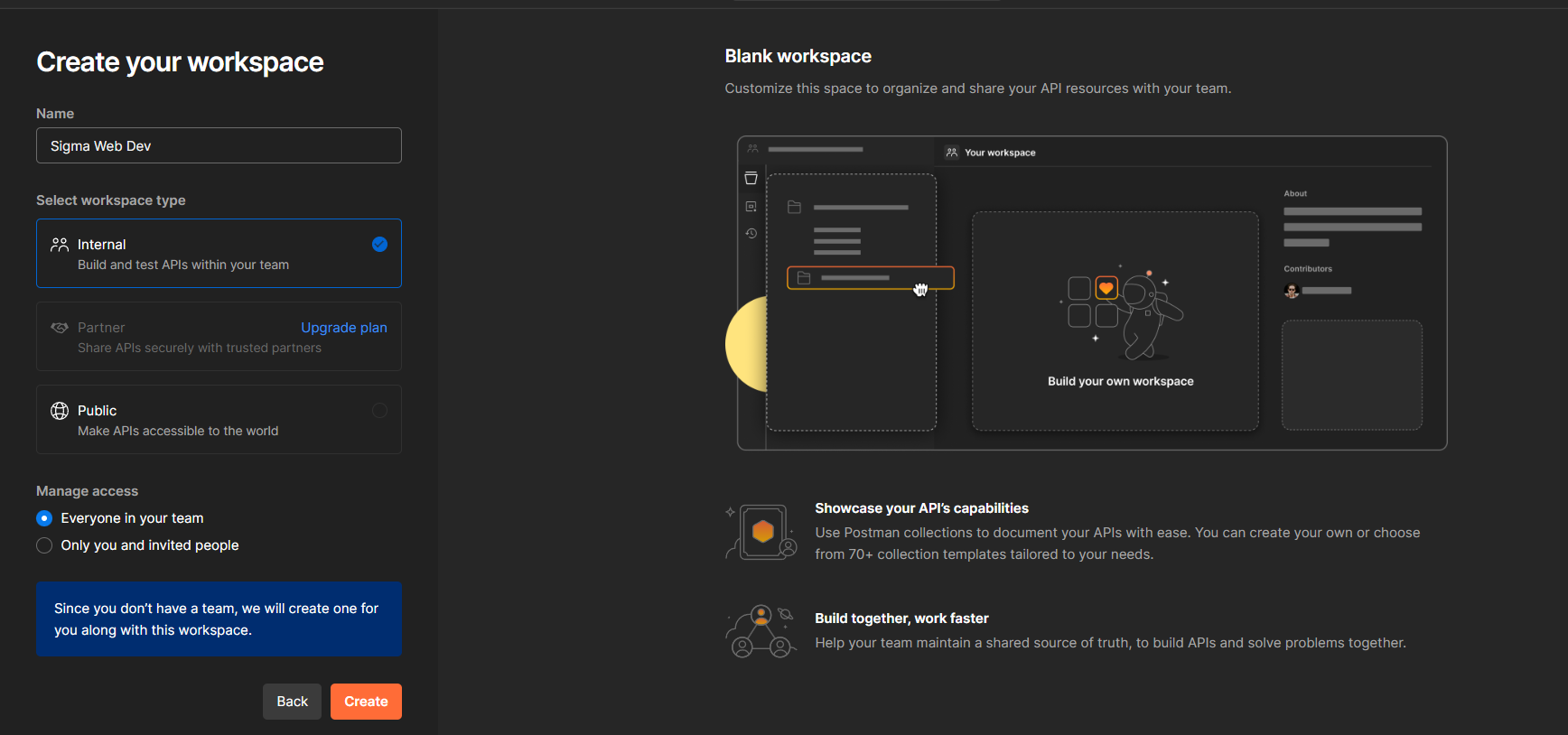
Type the name:



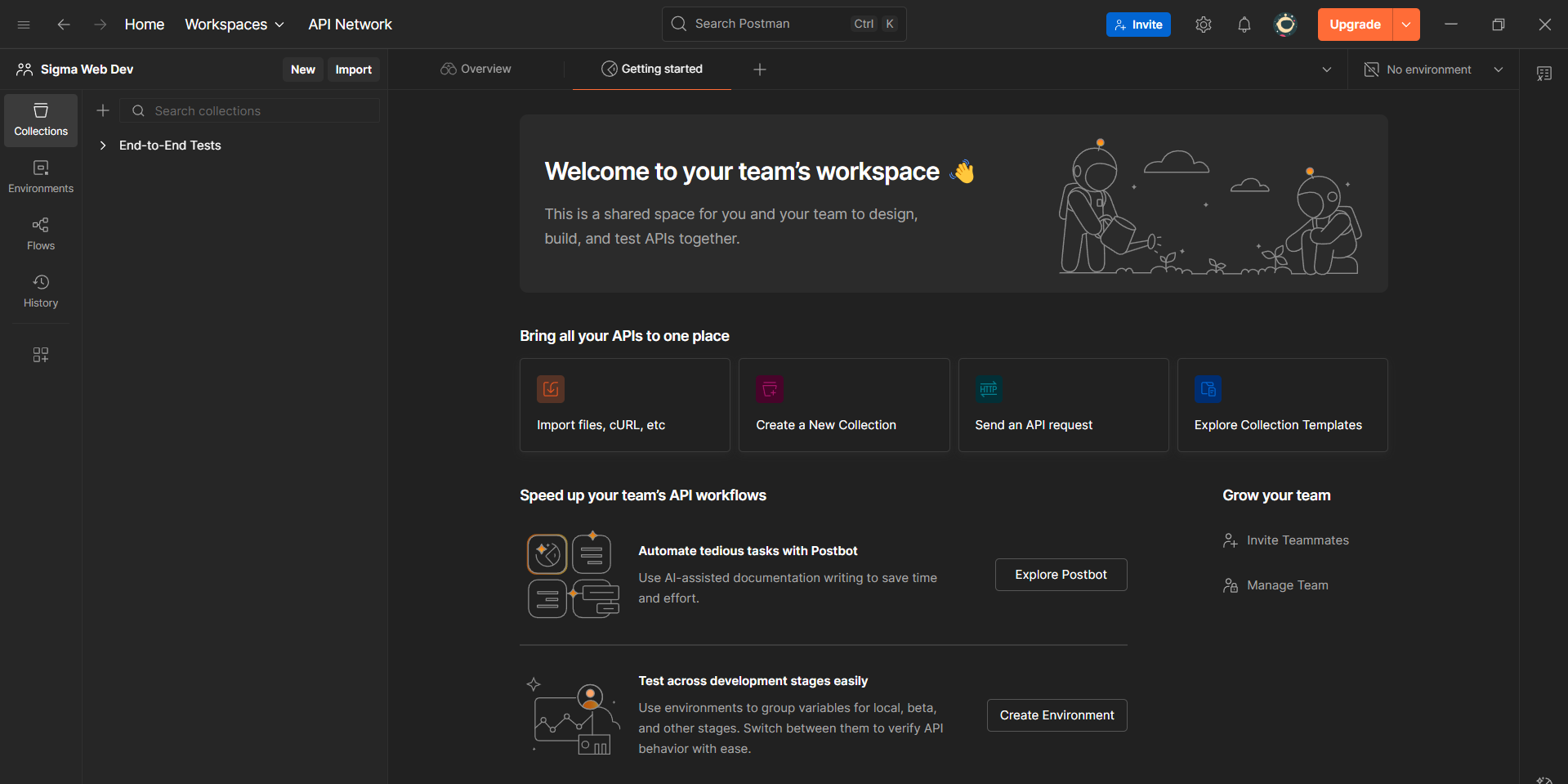
Following window will appear: click on the “blank workspace” option.

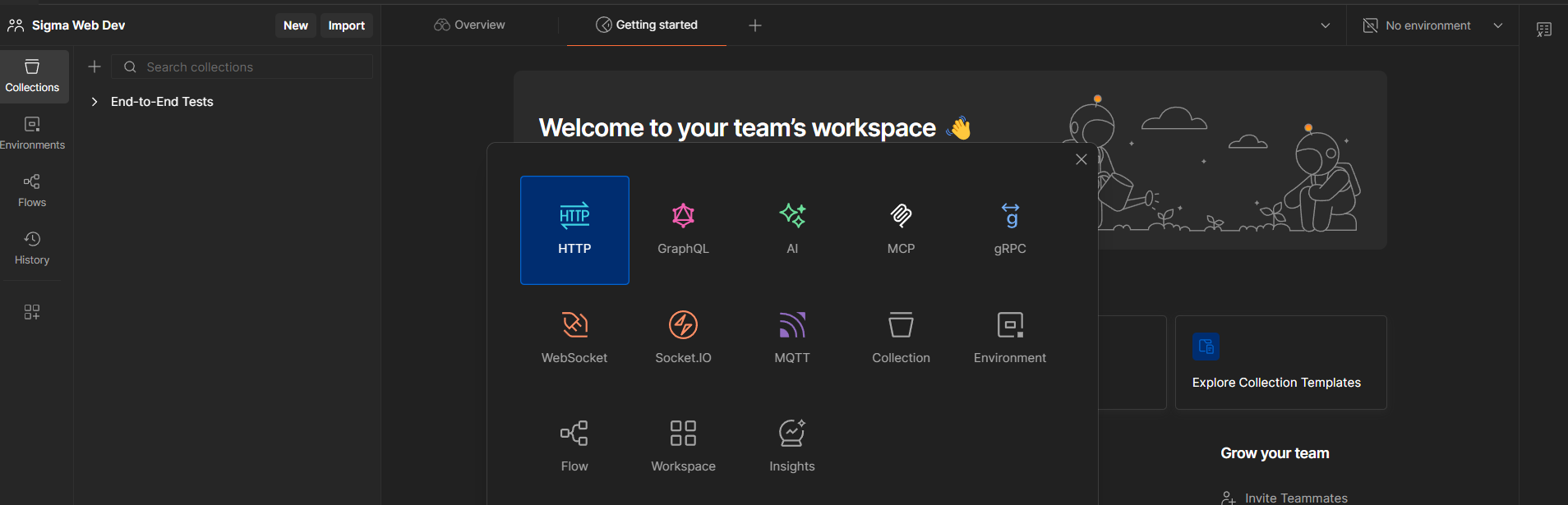


Specify the name:

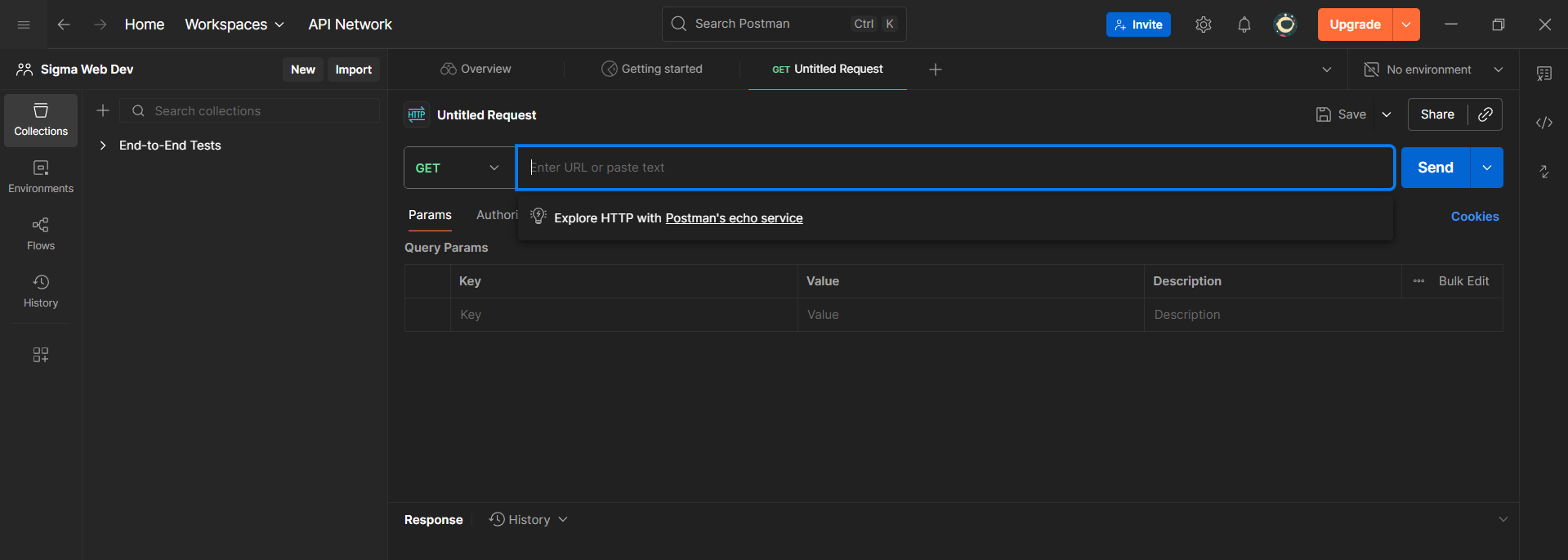


Following window confirms that workspace is created.

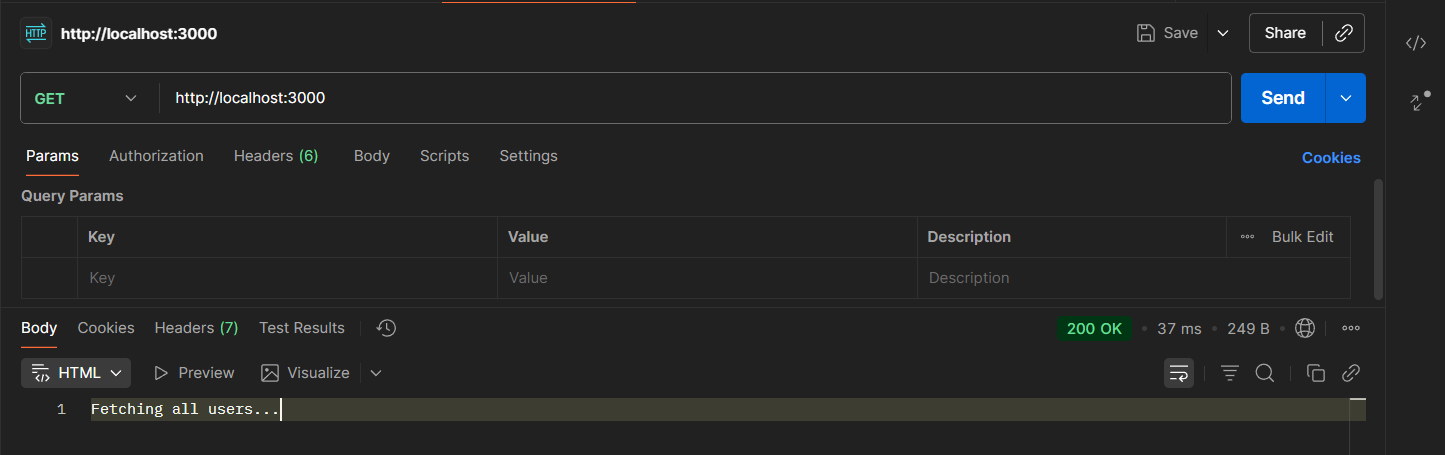


Now, click on the “New”, following options will occur:  


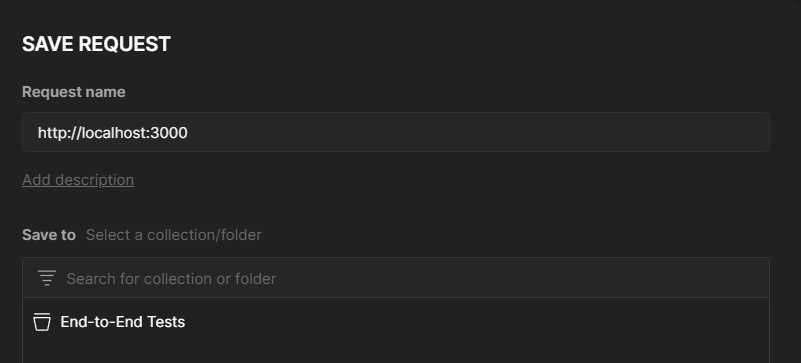
Click on the “HTTP”: following window will appear



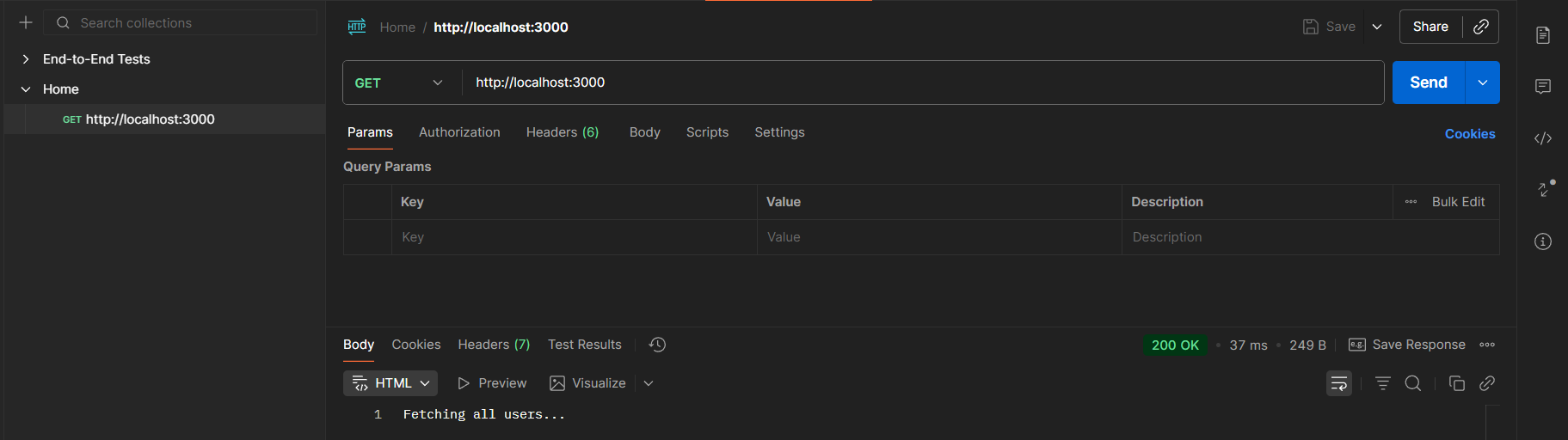
Now, just type the local host address in the search box: clearly, it worked



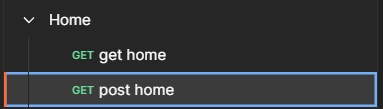
Now, we can also collect and arrange the requests in the postman, just click on “save” button: following screen will appear

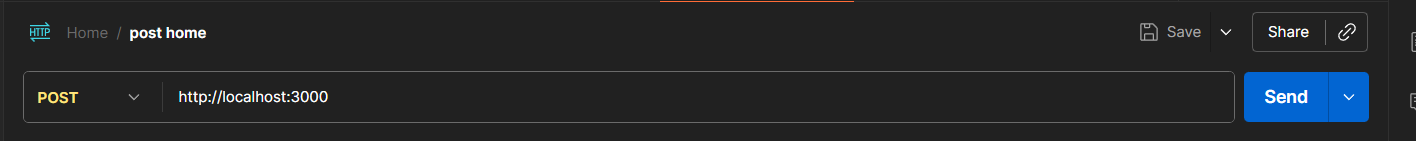


Suppose we create “home” named workspace and saved the content.

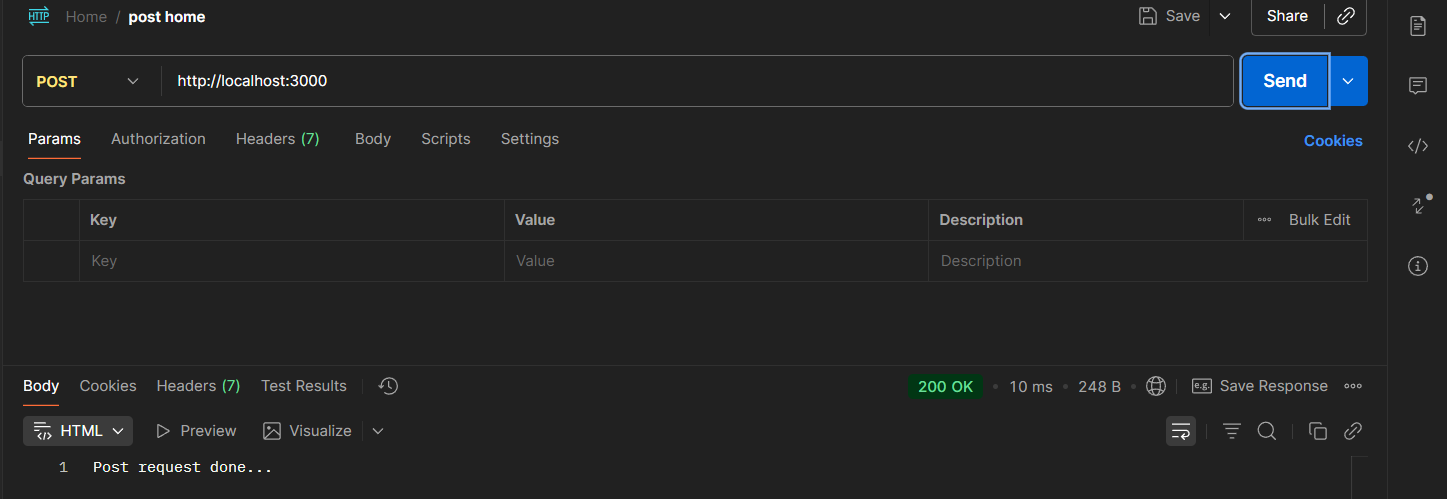


We can also rename and duplicate:



Now, we are doing the post request on the Home, using the 

Just click on the “Send” button to test: clearly, it worked



--The End--