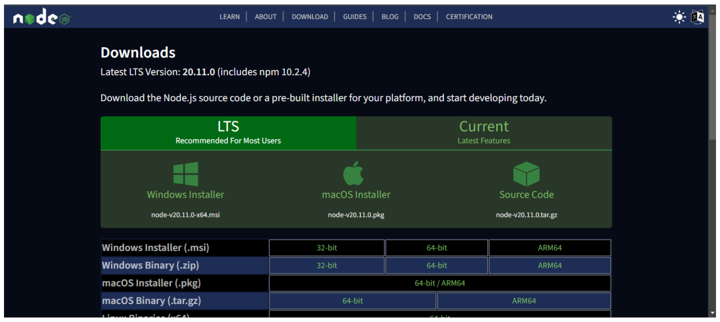
Mathew James M. Dagle IT - INTPROG32 BSIT - 3

Build a REST API with Typescript, NodeJS, ExpressJS and a file-based storage system.

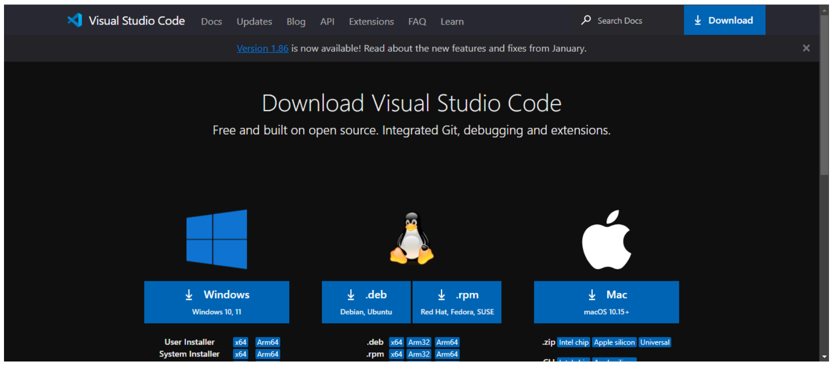
Step 1: Install the necessary software & runtime environment for building the API.


LEARN | ABOUT | DOWNLOAD | GUIDES | BLOG | DOCS | CERTIFICATION
Downloads
Latest LTS Version: 20.11.0 (includes npm 10.2.4)
Download the Node.js source code or a pre-built installer for your platform, and start developing today.
LTS
Recommended For Most Users
Current
Latest Features
Windows Installer
node-v20.11.0-x64.msi
macOS Installer
node-v20.11.0.pkg
Source Code
node-v20.11.0.tar.gz
Windows Installer (.msi) 32-bit 64-bit ARM64
Windows Binary (.zip) 32-bit 64-bit ARM64
macOS Installer (.pkg) 64-bit / ARM64
macOS Binary (.tar.gz) 64-bit ARM64

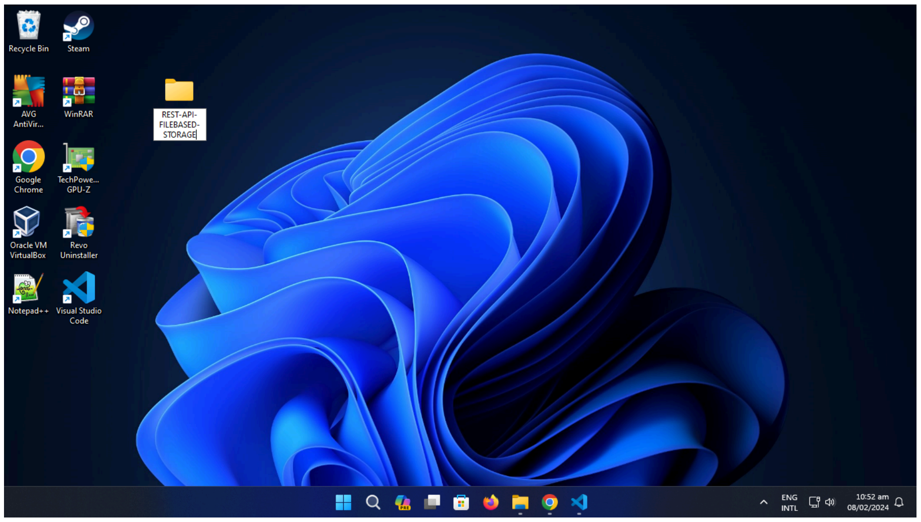
NodeJS
Download link/portal:
https://nodejs.org/en/download

Visual Studio Code
Download link/portal:
https://code.visualstudio.com/download

(or you can download any IDE of your choice).

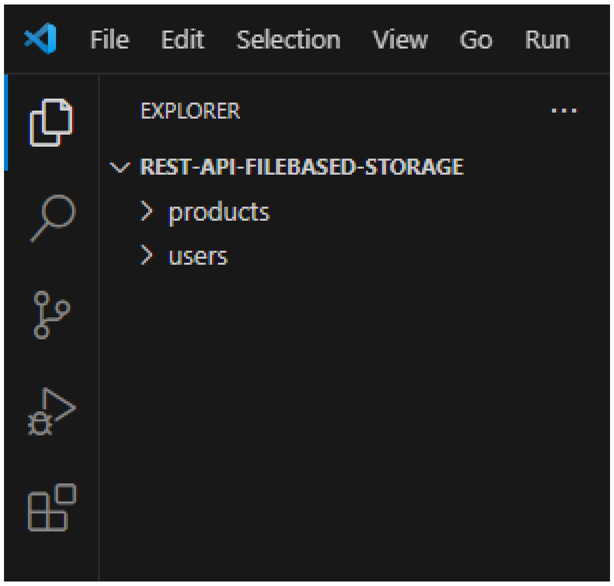

Visual Studio Code Docs Updates Blog API Extensions FAQ Learn
Search Docs
Download
Version 1.86 is now available! Read about the new features and fixes from January.
Download Visual Studio Code
Free and built on open source. Integrated Git, debugging and extensions.
Windows
Windows 10, 11
.deb
Debian, Ubuntu
.rpm
Red Hat, Fedora, SUSE
Mac
macOS 10.15+
User Installer x64 Arm64
System Installer x64 Arm64
.deb x64 Arm32 Arm64
.rpm x64 Arm32 Arm64
.zip Intel chip Apple silicon Universal

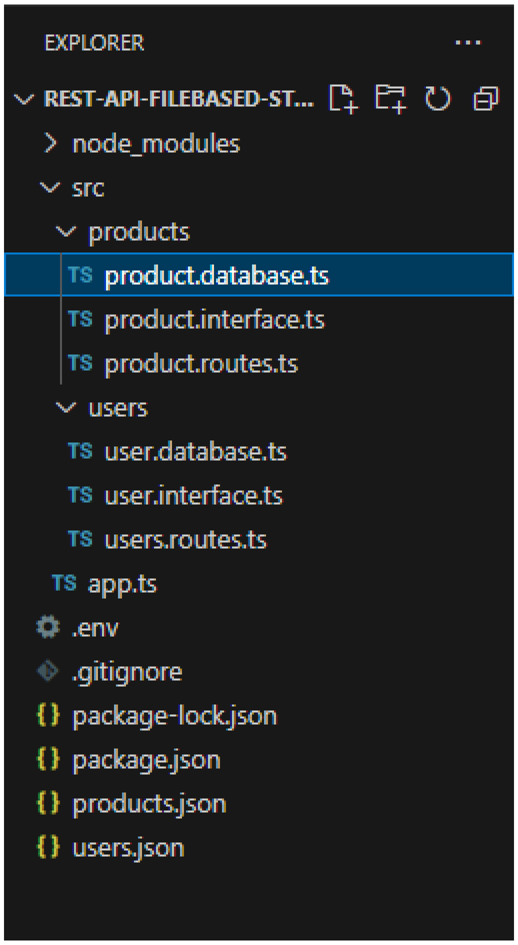
Step 2: Getting Started with TypeScript in Node.js



Create a New Folder for your project on your local machine.

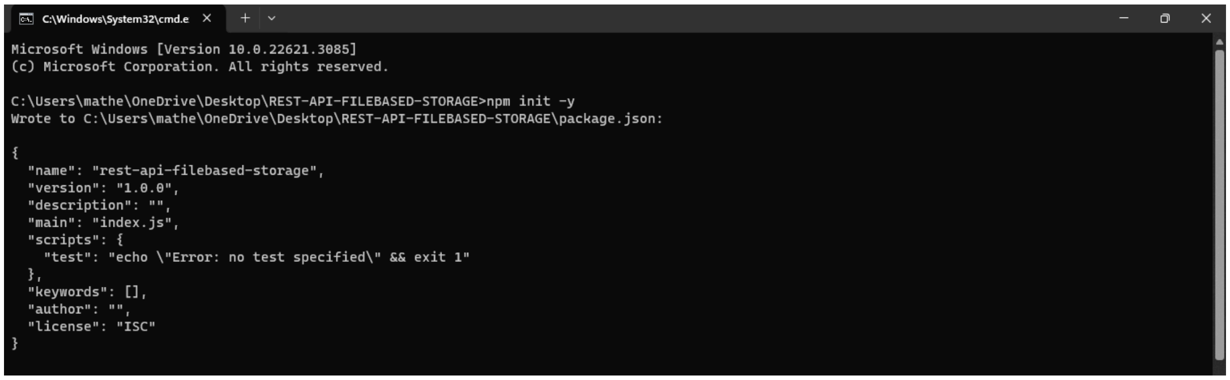
Create a project directory that looks like this:





- All the **files** and **directories** created on the right side are user-defined except for the **node_modules** folder which is generated after initializing the node package module (npm).

Initializing a NodeJS project

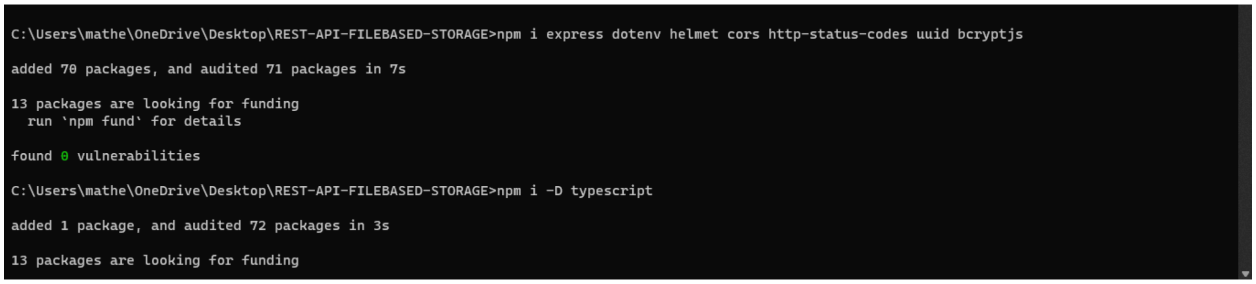


```
Microsoft Windows [Version 10.0.22621.3085]
(c) Microsoft Corporation. All rights reserved.

C:\Users\mathe\OneDrive\Desktop\REST-API-FILEBASED-STORAGE>npm init -y
Wrote to C:\Users\mathe\OneDrive\Desktop\REST-API-FILEBASED-STORAGE\package.json:

{
  "name": "rest-api-filebased-storage",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}
```

Next, initialize a Node.js project within the project directory by creating a package.json file with default settings, using this command:

- npm init -y

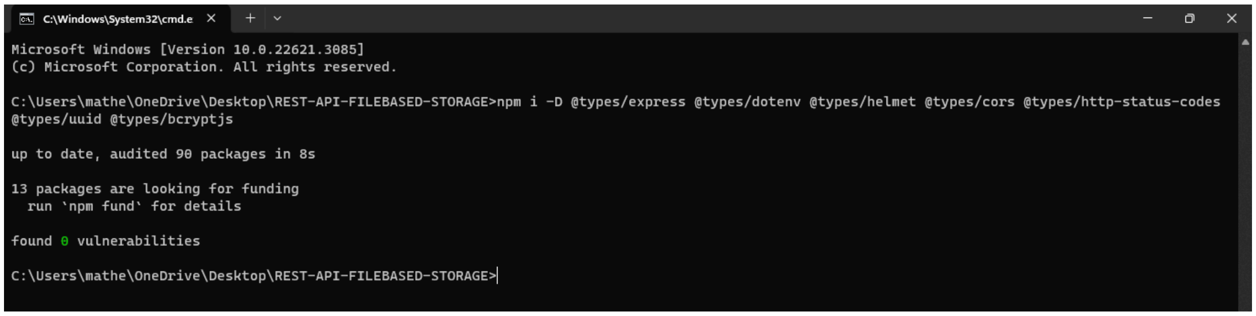
Installing Project Dependencies



```
C:\Users\mathe\OneDrive\Desktop\REST-API-FILEBASED-STORAGE>npm i express dotenv helmet cors http-status-codes uuid bcryptjs

added 70 packages, and audited 71 packages in 7s

13 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities

C:\Users\mathe\OneDrive\Desktop\REST-API-FILEBASED-STORAGE>npm i -D typescript

added 1 package, and audited 72 packages in 3s

13 packages are looking for funding
```

Your Node.js project requires a couple of dependencies to create a secure Express server with TypeScript. You also need the TypeScript for enhanced code development Install them like so:

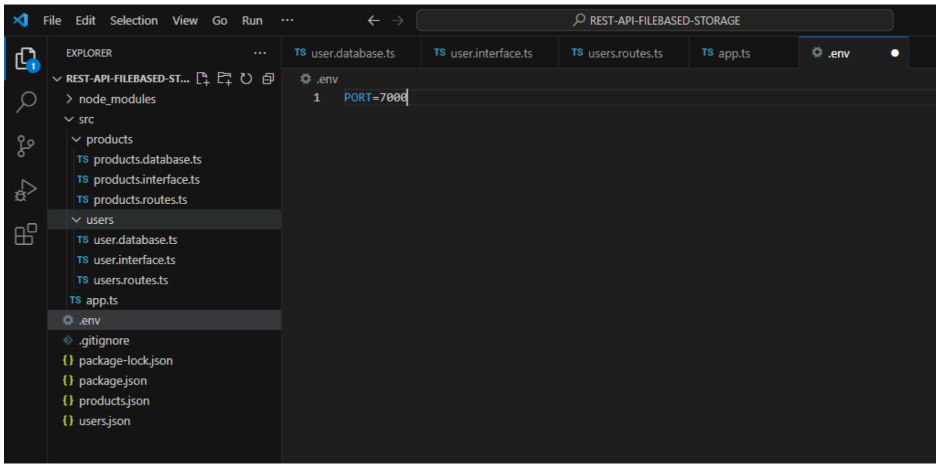
- npm i express dotenv helmet cors http-status-codes uuid bcryptjs
- npm i -D typescript



```
Microsoft Windows [Version 10.0.22621.3085]
(c) Microsoft Corporation. All rights reserved.

C:\Users\mathe\OneDrive\Desktop\REST-API-FILEBASED-STORAGE>npm i -D @types/express @types/dotenv @types/helmet @types/cors @types/http-status-codes
@types/uuid @types/bcryptjs

up to date, audited 90 packages in 8s

13 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities

C:\Users\mathe\OneDrive\Desktop\REST-API-FILEBASED-STORAGE>
```

To use TypeScript effectively, you need to install type definitions for the packages you installed previously:

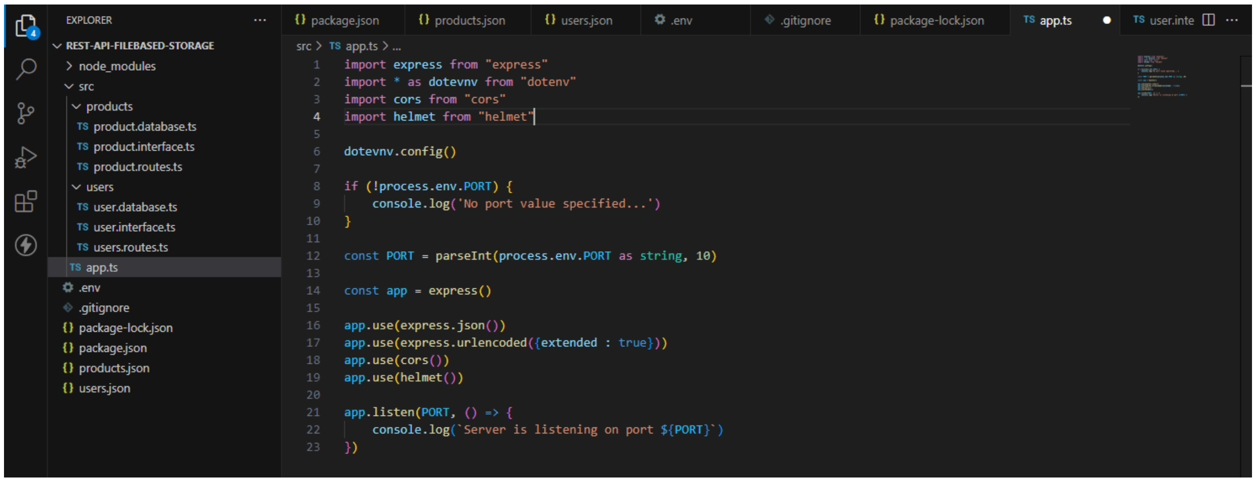
- npm i -D @types/express @types/dotenv @types/helmet @types/cors @types/http-status-codes @types/uuid @types/bcryptjs



```
PORT=7000
```

Populate the .env file with a variable called **PORT** with a value of 7000 for which the server can use to listen for requests

Import the project dependencies installed earlier on the app.ts file and load any environmental variables from the local .env file using the dotenv.config() method.

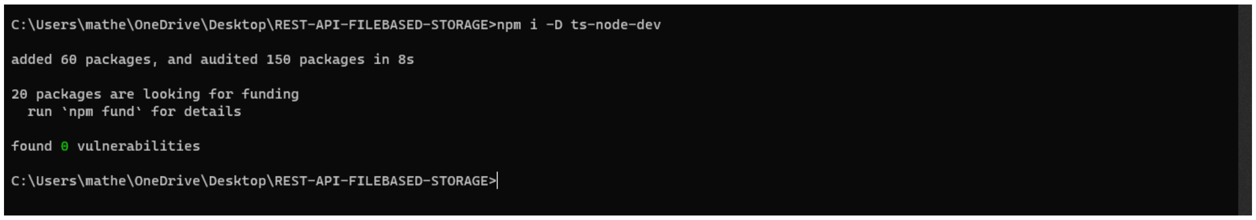


```ts
import express from "express"
import * as dotenv from "dotenv"
import cors from "cors"
import helmet from "helmet"

dotenv.config()

if (!process.env.PORT) {
    console.log('No port value specified...')
}

const PORT = parseInt(process.env.PORT as string, 10)

const app = express()

app.use(express.json())
app.use(express.urlencoded({extended : true}))
app.use(cors())
app.use(helmet())

app.listen(PORT, () => {
    console.log(`Server is listening on port ${PORT}`)
})
```

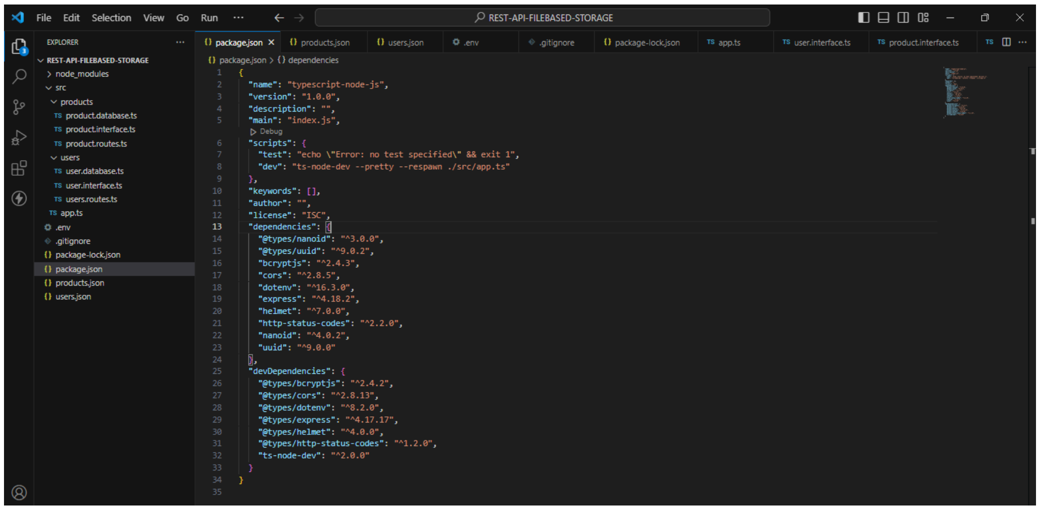
Step 3: Improve the TypeScript Development Workflow

Start by installing this package to power up your development workflow:

- npm i -D ts-node-dev



```
C:\Users\mathe\OneDrive\Desktop\REST-API-FILEBASED-STORAGE>npm i -D ts-node-dev

added 60 packages, and audited 150 packages in 8s

20 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities

C:\Users\mathe\OneDrive\Desktop\REST-API-FILEBASED-STORAGE>
```

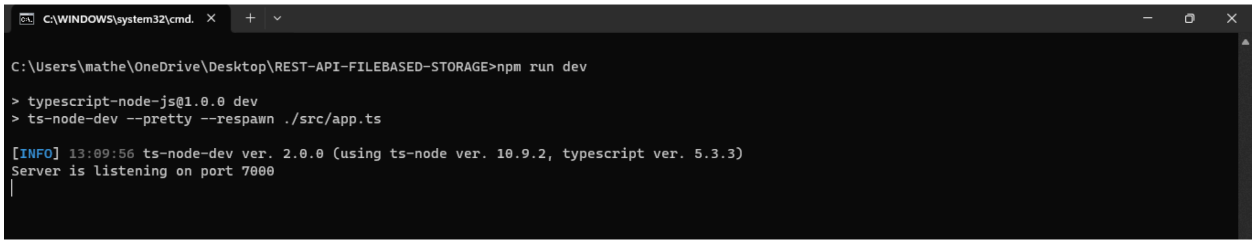


```json
{
  "name": "typescript-node-js",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "dev": "ts-node-dev --pretty --respawn ./src/app.ts"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "@types/nanoid": "^3.0.0",
    "@types/uuid": "^9.0.2",
    "bcryptjs": "^2.4.3",
    "cors": "^2.8.5",
    "dotenv": "^16.3.0",
    "express": "^4.18.2",
    "helmet": "^7.0.0",
    "http-status-codes": "^2.2.0",
    "nanoid": "^4.0.2",
    "uuid": "^9.0.0"
  },
  "devDependencies": {
    "@types/bcryptjs": "^2.4.2",
    "@types/cors": "^2.8.13",
    "@types/dotenv": "^8.2.0",
    "@types/express": "^4.17.17",
    "@types/helmet": "^4.0.0",
    "@types/http-status-codes": "^1.2.0",
    "ts-node-dev": "^2.0.0"
  }
}
```

You can create a dev npm script in package.json to run your server. Update your package.json file like this.

Now, simply run the dev script to launch your project:

- npm run dev

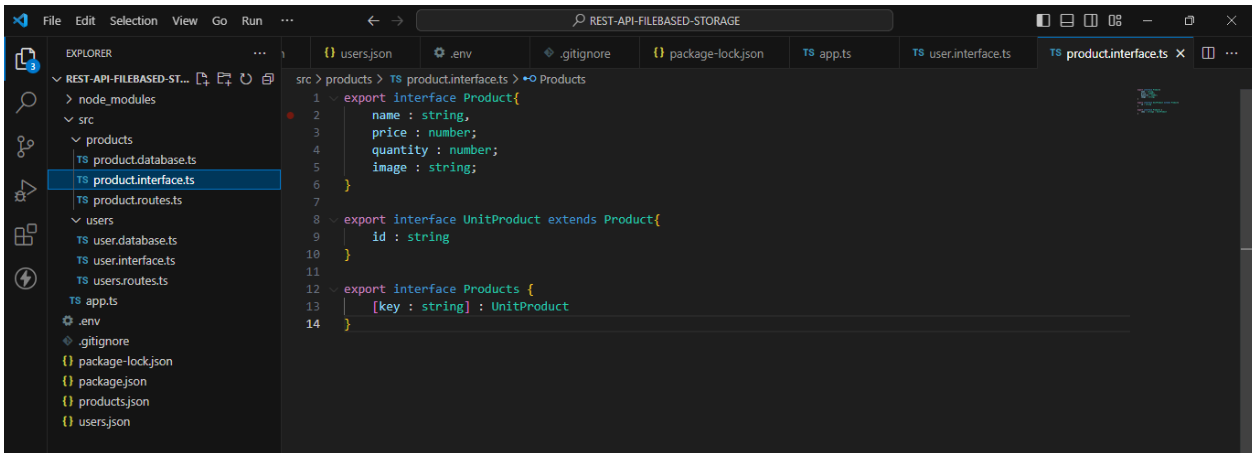


```
C:\Users\mathe\OneDrive\Desktop\REST-API-FILEBASED-STORAGE>npm run dev

> typescript-node-js@1.0.0 dev
> ts-node-dev --pretty --respawn ./src/app.ts

[INFO] 13:09:56 ts-node-dev ver. 2.0.0 (using ts-node ver. 10.9.2, typescript ver. 5.3.3)
Server is listening on port 7000
```

If everything is working correctly, you'll see a message indicating that the server is listening for requests on port 7000.

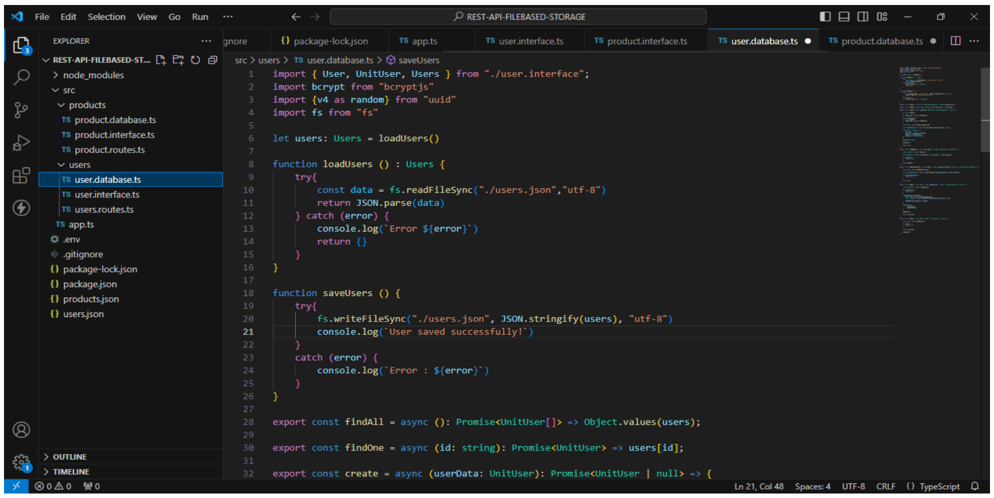
Step 4: Model Data with TypeScript Interfaces /Users

Populate src/users/user.interface.ts with the following definition:

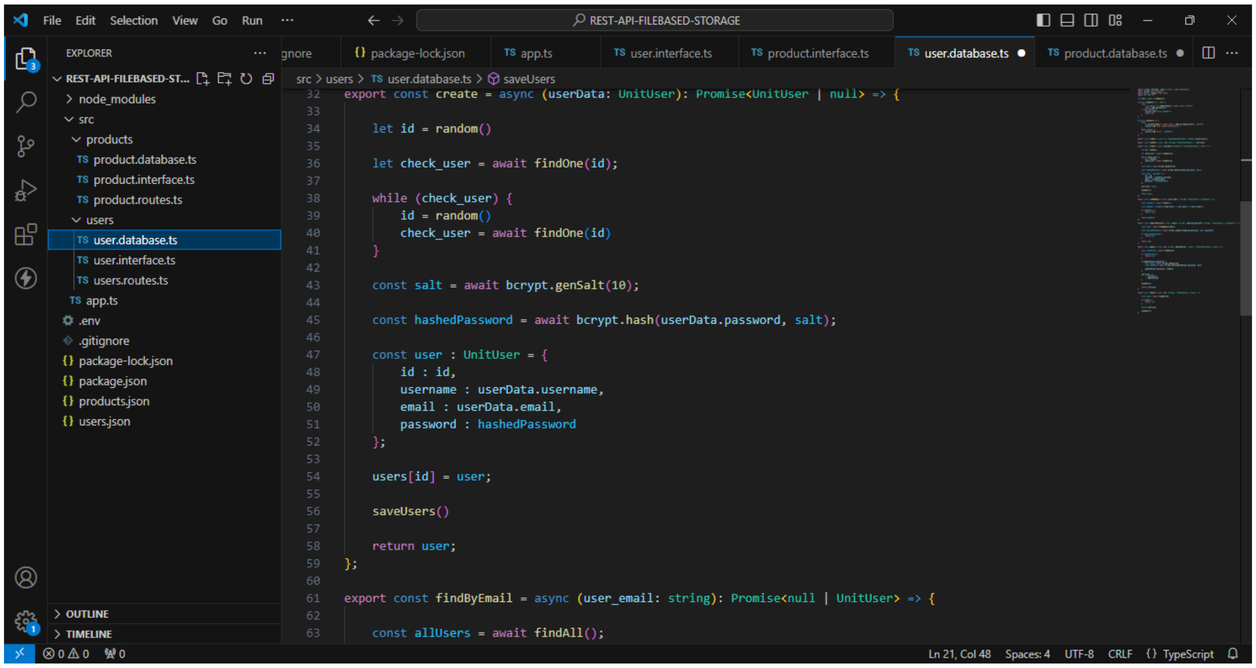


```typescript
export interface Product{
    name : string,
    price : number;
    quantity : number;
    image : string;
}

export interface UnitProduct extends Product{
    id : string
}

export interface Products {
    [key : string] : UnitProduct
}
```

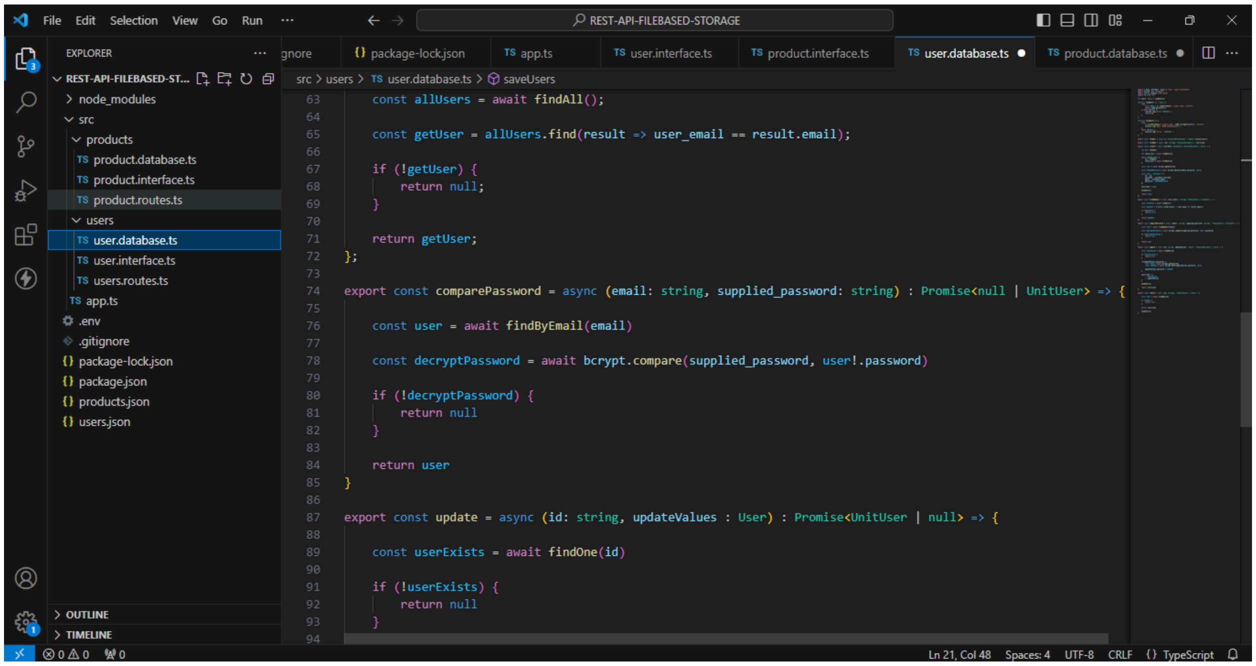
Next, we will create the logic for our data storage. You can call it a database if you like. Populate src/users/user.database.ts with the following code:



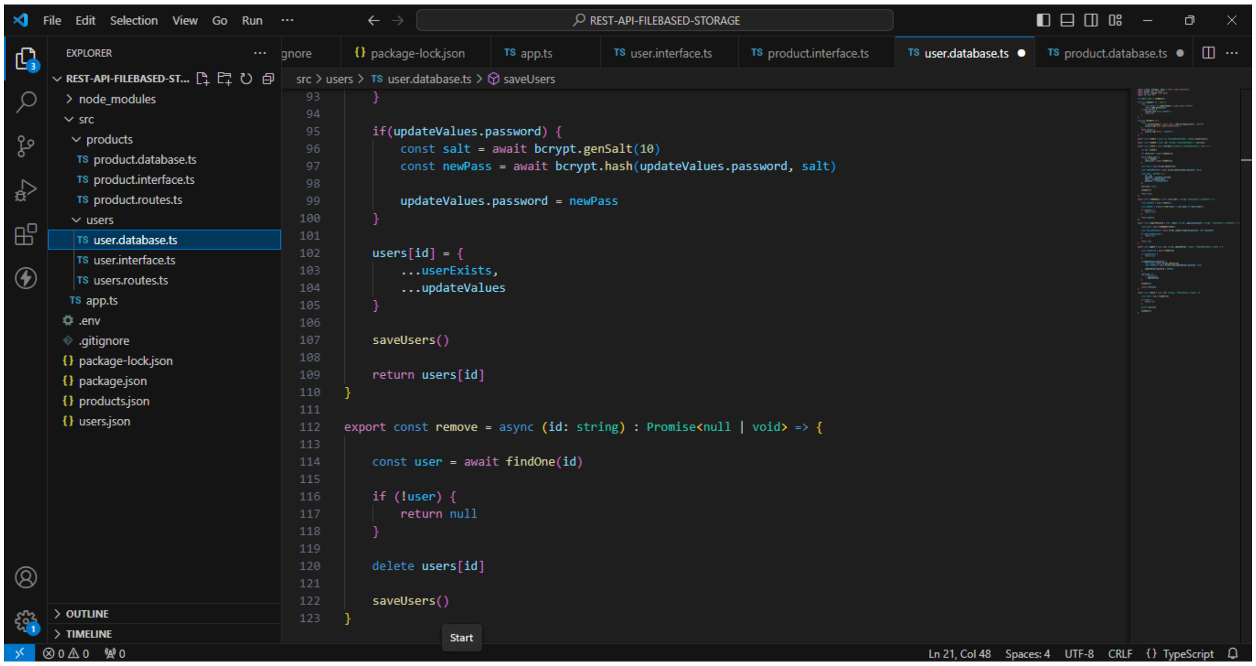
```typescript
import { User, UnitUser, Users } from "./user.interface";
import bcrypt from "bcryptjs"
import {v4 as random} from "uuid"
import fs from "fs"

let users: Users = loadUsers()

function loadUsers () : Users {
    try{
        const data = fs.readFileSync("./users.json","utf-8")
        return JSON.parse(data)
    } catch (error) {
        console.log(`Error ${error}`)
        return {}
    }
}

function saveUsers () {
    try{
        fs.writeFileSync("./users.json", JSON.stringify(users), "utf-8")
        console.log(`User saved successfully!`)
    }
    catch (error) {
        console.log(`Error : ${error}`)
    }
}

export const findAll = async (): Promise<UnitUser[]> => Object.values(users);

export const findOne = async (id: string): Promise<UnitUser> => users[id];

export const create = async (userData: UnitUser): Promise<UnitUser | null> => {
```



```typescript
export const create = async (userData: UnitUser): Promise<UnitUser | null> => {

    let id = random()

    let check_user = await findOne(id);

    while (check_user) {
        id = random()
        check_user = await findOne(id)
    }

    const salt = await bcrypt.genSalt(10);

    const hashedPassword = await bcrypt.hash(userData.password, salt);

    const user : UnitUser = {
        id : id,
        username : userData.username,
        email : userData.email,
        password : hashedPassword
    };

    users[id] = user;

    saveUsers()

    return user;
};

export const findByEmail = async (user_email: string): Promise<null | UnitUser> => {

    const allUsers = await findAll();
```

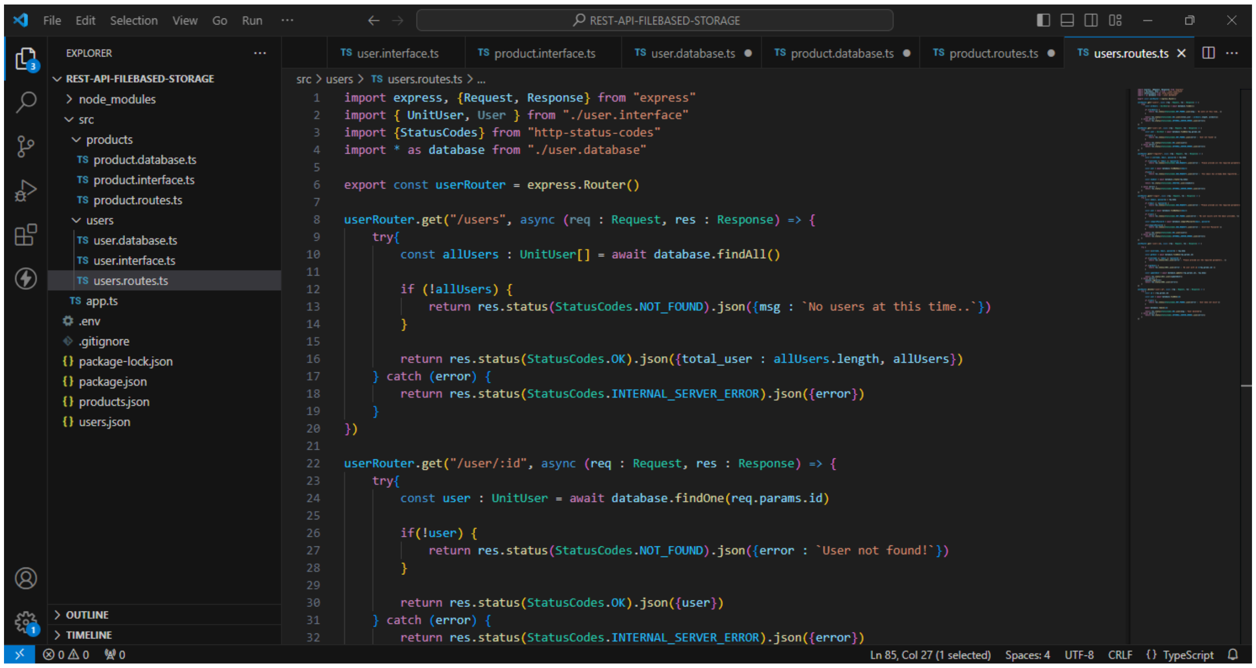


```typescript
    const allUsers = await findAll();

    const getUser = allUsers.find(result => user_email == result.email);

    if (!getUser) {
        return null;
    }

    return getUser;
};

export const comparePassword = async (email: string, supplied_password: string) : Promise<null | UnitUser> => {

    const user = await findByEmail(email)

    const decryptPassword = await bcrypt.compare(supplied_password, user!.password)

    if (!decryptPassword) {
        return null
    }

    return user
}

export const update = async (id: string, updateValues : User) : Promise<UnitUser | null> => {

    const userExists = await findOne(id)

    if (!userExists) {
        return null
    }
```

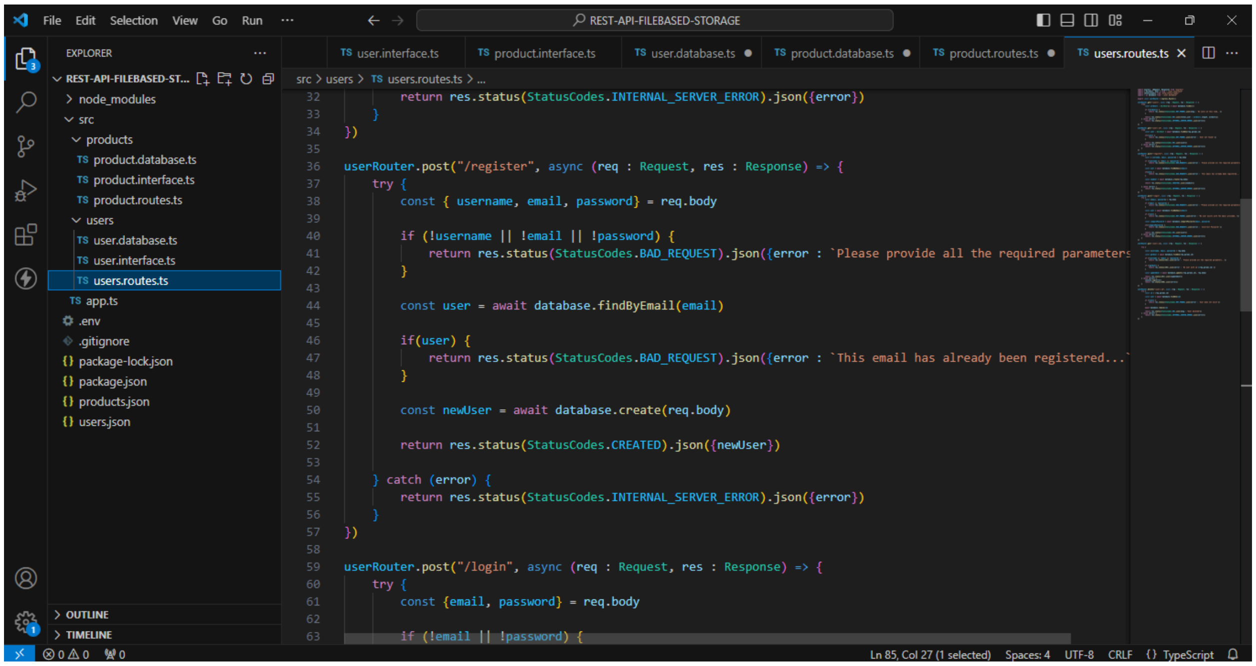


```
    }

    if(updateValues.password) {
        const salt = await bcrypt.genSalt(10)
        const newPass = await bcrypt.hash(updateValues.password, salt)

        updateValues.password = newPass
    }

    users[id] = {
        ...userExists,
        ...updateValues
    }

    saveUsers()

    return users[id]
}

export const remove = async (id: string) : Promise<null | void> => {

    const user = await findOne(id)

    if (!user) {
        return null
    }

    delete users[id]

    saveUsers()
}
```

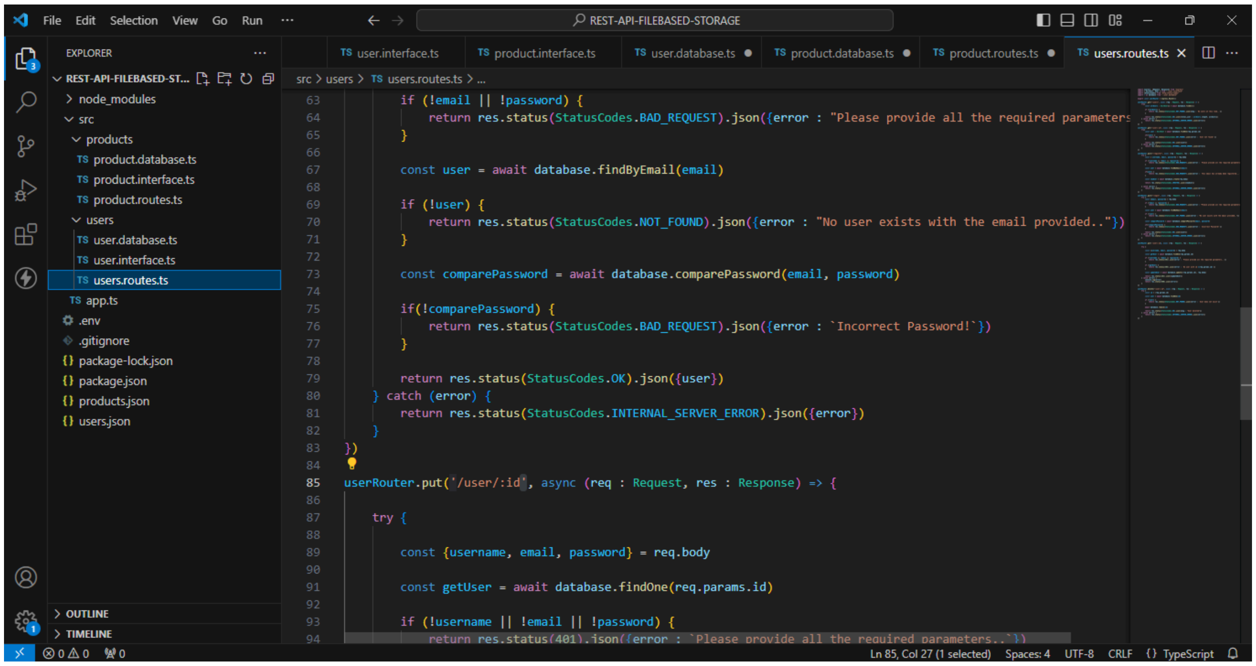
Next, let all import all the required functions and modules into the routes file ./src/users.routes.ts and populate as follows:



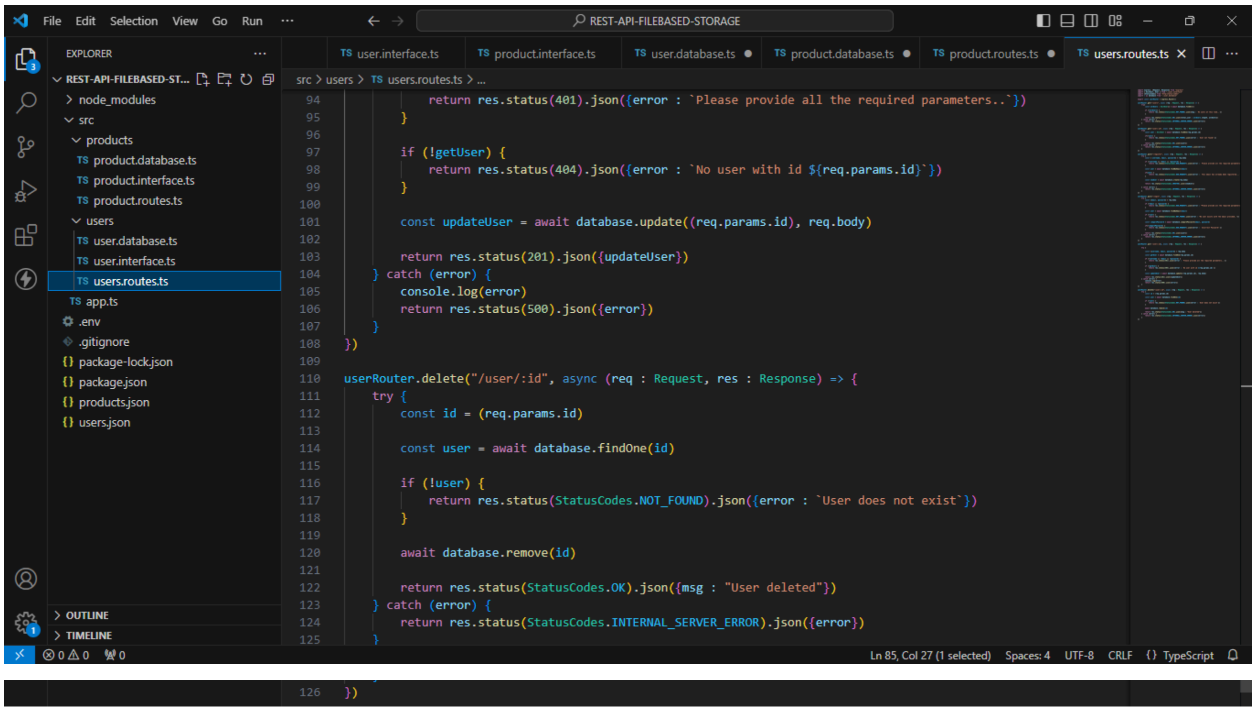
```
import express, {Request, Response} from "express"
import { UnitUser, User } from "./user.interface"
import {StatusCodes} from "http-status-codes"
import * as database from "./user.database"

export const userRouter = express.Router()

userRouter.get("/users", async (req : Request, res : Response) => {
    try{
        const allUsers : UnitUser[] = await database.findAll()

        if (!allUsers) {
            return res.status(StatusCodes.NOT_FOUND).json({msg : `No users at this time..`})
        }

        return res.status(StatusCodes.OK).json({total_user : allUsers.length, allUsers})
    } catch (error) {
        return res.status(StatusCodes.INTERNAL_SERVER_ERROR).json({error})
    }
})

userRouter.get("/user/:id", async (req : Request, res : Response) => {
    try{
        const user : UnitUser = await database.findOne(req.params.id)

        if(!user) {
            return res.status(StatusCodes.NOT_FOUND).json({error : `User not found!`})
        }

        return res.status(StatusCodes.OK).json({user})
    } catch (error) {
        return res.status(StatusCodes.INTERNAL_SERVER_ERROR).json({error})
```



```
            return res.status(StatusCodes.INTERNAL_SERVER_ERROR).json({error})
        }
})

userRouter.post("/register", async (req : Request, res : Response) => {
    try {
        const { username, email, password} = req.body

        if (!username || !email || !password) {
            return res.status(StatusCodes.BAD_REQUEST).json({error : `Please provide all the required parameters
        }

        const user = await database.findByEmail(email)

        if(user) {
            return res.status(StatusCodes.BAD_REQUEST).json({error : `This email has already been registered...`
        }

        const newUser = await database.create(req.body)

        return res.status(StatusCodes.CREATED).json({newUser})

    } catch (error) {
        return res.status(StatusCodes.INTERNAL_SERVER_ERROR).json({error})
    }
})

userRouter.post("/login", async (req : Request, res : Response) => {
    try {
        const {email, password} = req.body

        if (!email || !password) {
```



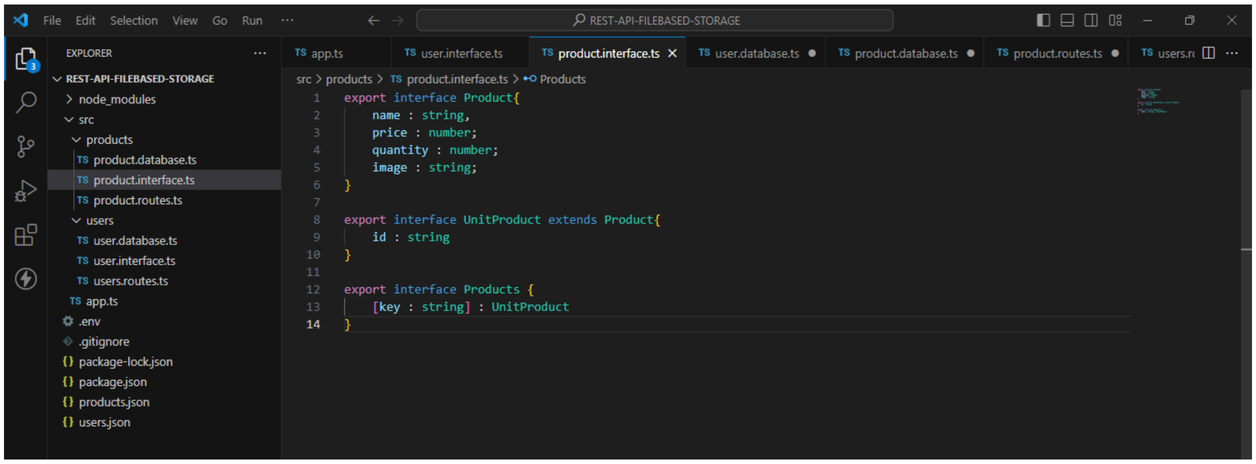
```
        if (!email || !password) {
            return res.status(StatusCodes.BAD_REQUEST).json({error : "Please provide all the required parameters
        }

        const user = await database.findByEmail(email)

        if (!user) {
            return res.status(StatusCodes.NOT_FOUND).json({error : "No user exists with the email provided.."})
        }

        const comparePassword = await database.comparePassword(email, password)

        if(!comparePassword) {
            return res.status(StatusCodes.BAD_REQUEST).json({error : `Incorrect Password!`})
        }

        return res.status(StatusCodes.OK).json({user})
    } catch (error) {
        return res.status(StatusCodes.INTERNAL_SERVER_ERROR).json({error})
    }
})

userRouter.put('/user/:id', async (req : Request, res : Response) => {

    try {

        const {username, email, password} = req.body

        const getUser = await database.findOne(req.params.id)

        if (!username || !email || !password) {
            return res.status(401).json({error : `Please provide all the required parameters..`})
```



```
            return res.status(401).json({error : `Please provide all the required parameters..`})
        }

        if (!getUser) {
            return res.status(404).json({error : `No user with id ${req.params.id}`})
        }

        const updateUser = await database.update((req.params.id), req.body)

        return res.status(201).json({updateUser})
    } catch (error) {
        console.log(error)
        return res.status(500).json({error})
    }
})

userRouter.delete("/user/:id", async (req : Request, res : Response) => {
    try {
        const id = (req.params.id)

        const user = await database.findOne(id)

        if (!user) {
            return res.status(StatusCodes.NOT_FOUND).json({error : `User does not exist`})
        }

        await database.remove(id)

        return res.status(StatusCodes.OK).json({msg : "User deleted"})
    } catch (error) {
        return res.status(StatusCodes.INTERNAL_SERVER_ERROR).json({error})
    }
})
```

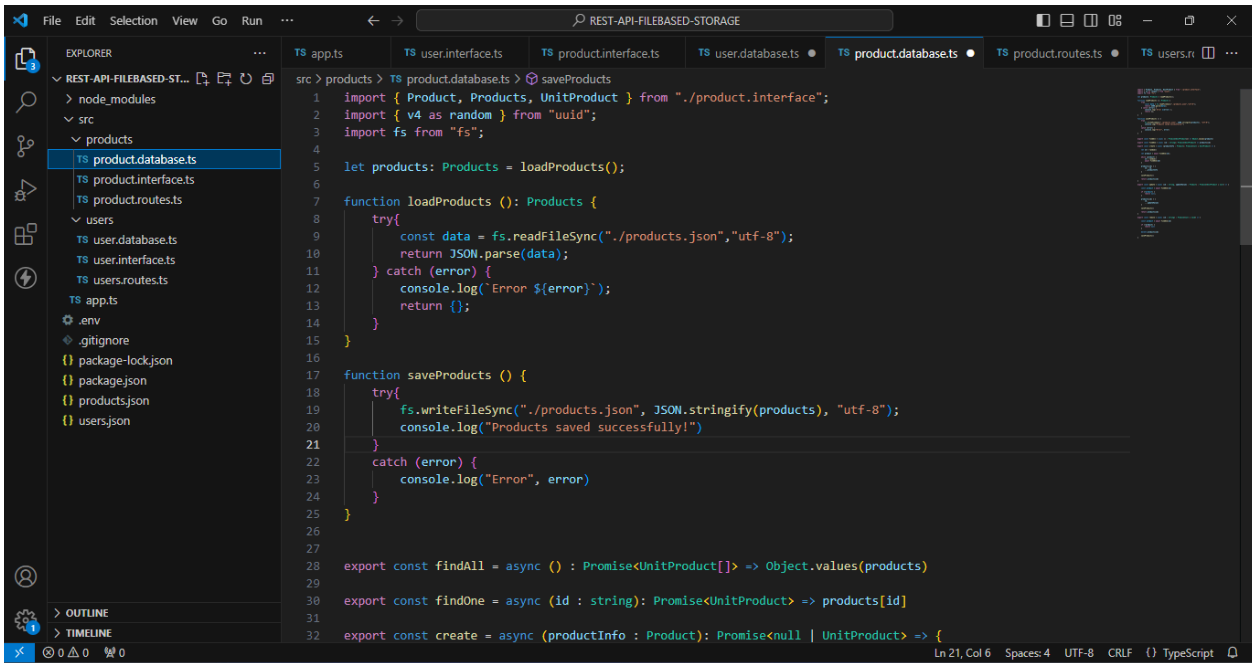
/Products

Create the login and routes for our products.
So let's duplicate the contents of our users interface with minor changes into the file ./src/product.interface.ts

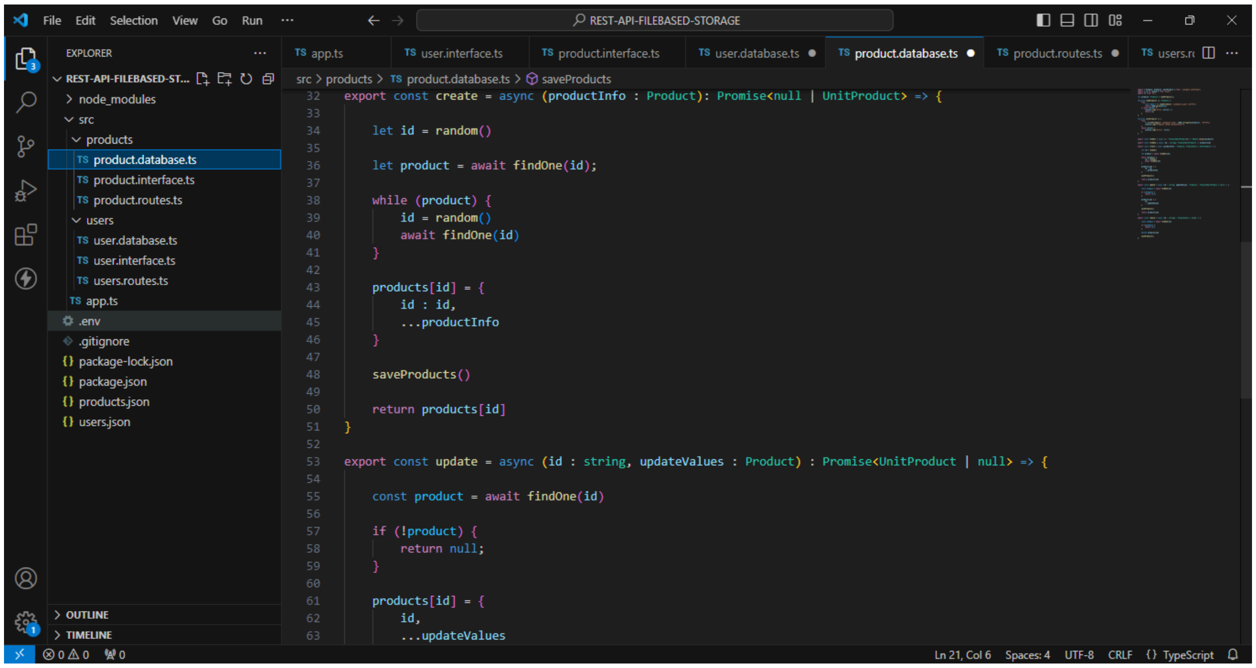


```
export interface Product{
    name : string,
    price : number;
    quantity : number;
    image : string;
}

export interface UnitProduct extends Product{
    id : string
}

export interface Products {
    [key : string] : UnitProduct
}
```

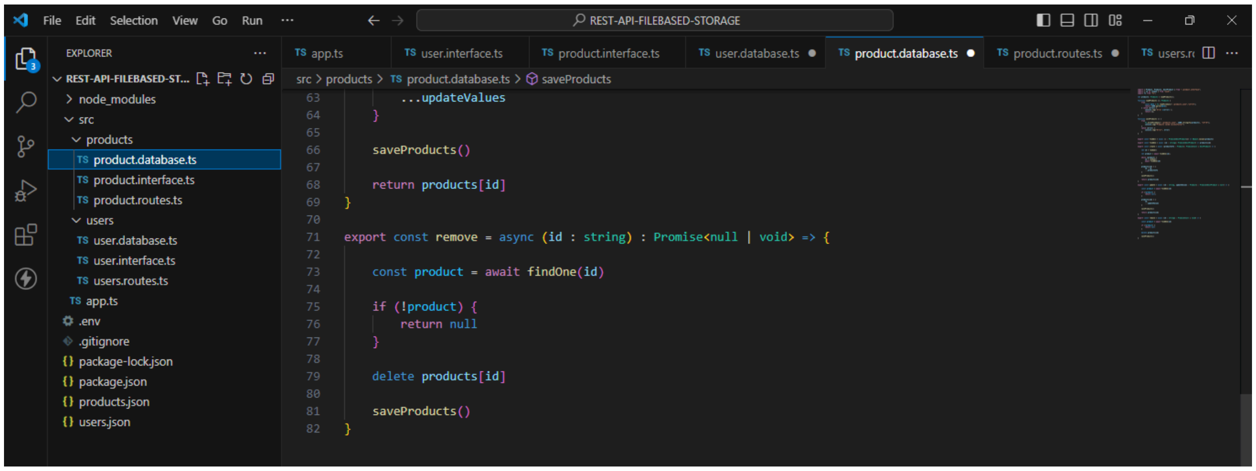
Next, just like in the ./src/users.database.ts file, let us populate the ./src/products.database.ts with a similar logic.



```typescript
import { Product, Products, UnitProduct } from "./product.interface";
import { v4 as random } from "uuid";
import fs from "fs";

let products: Products = loadProducts();

function loadProducts (): Products {
    try{
        const data = fs.readFileSync("./products.json","utf-8");
        return JSON.parse(data);
    } catch (error) {
        console.log(`Error ${error}`);
        return {};
    }
}

function saveProducts () {
    try{
        fs.writeFileSync("./products.json", JSON.stringify(products), "utf-8");
        console.log("Products saved successfully!")
    }
    catch (error) {
        console.log("Error", error)
    }
}


export const findAll = async () : Promise<UnitProduct[]> => Object.values(products)

export const findOne = async (id : string): Promise<UnitProduct> => products[id]

export const create = async (productInfo : Product): Promise<null | UnitProduct> => {
```

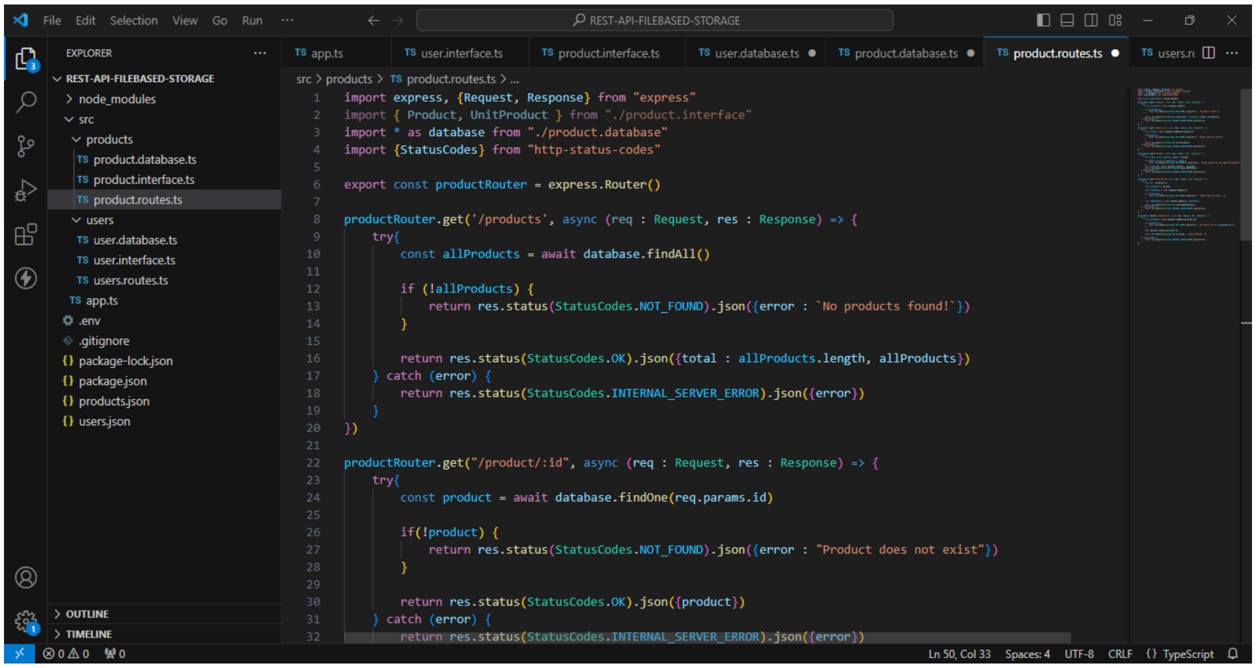


```typescript
export const create = async (productInfo : Product): Promise<null | UnitProduct> => {

    let id = random()

    let product = await findOne(id);

    while (product) {
        id = random()
        await findOne(id)
    }

    products[id] = {
        id : id,
        ...productInfo
    }

    saveProducts()

    return products[id]
}

export const update = async (id : string, updateValues : Product) : Promise<UnitProduct | null> => {

    const product = await findOne(id)

    if (!product) {
        return null;
    }

    products[id] = {
        id,
        ...updateValues
```

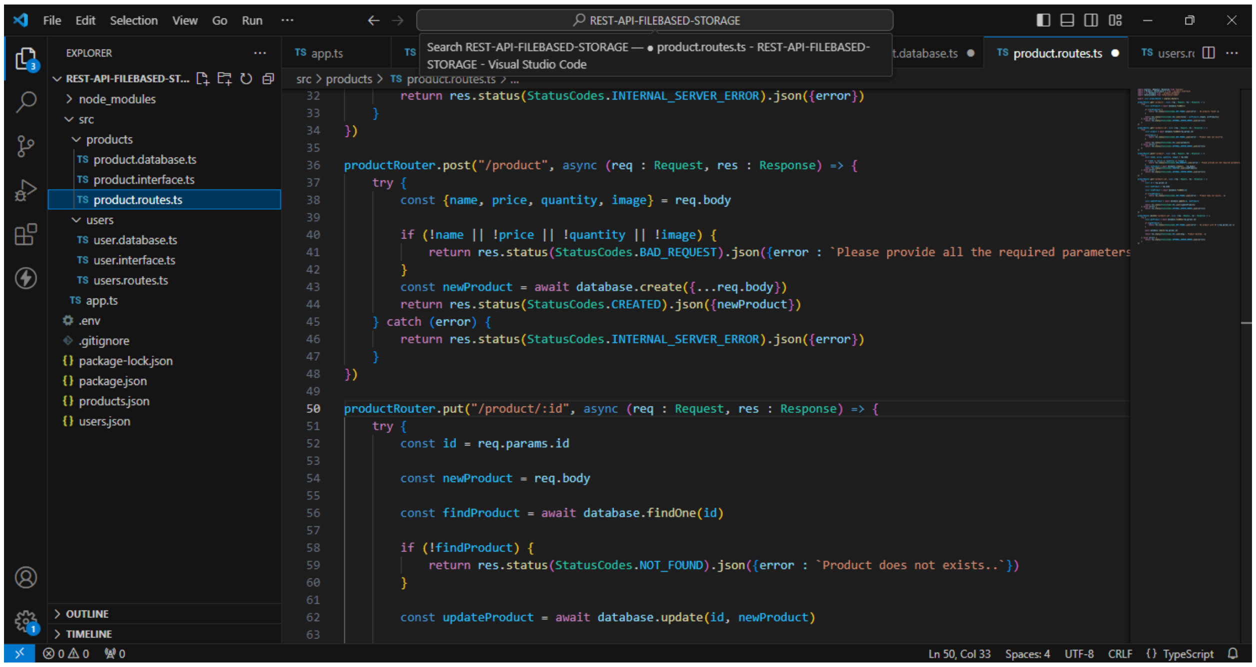


```
        ...updateValues
    }

    saveProducts()

    return products[id]
}

export const remove = async (id : string) : Promise<null | void> => {

    const product = await findOne(id)

    if (!product) {
        return null
    }

    delete products[id]

    saveProducts()
}
```

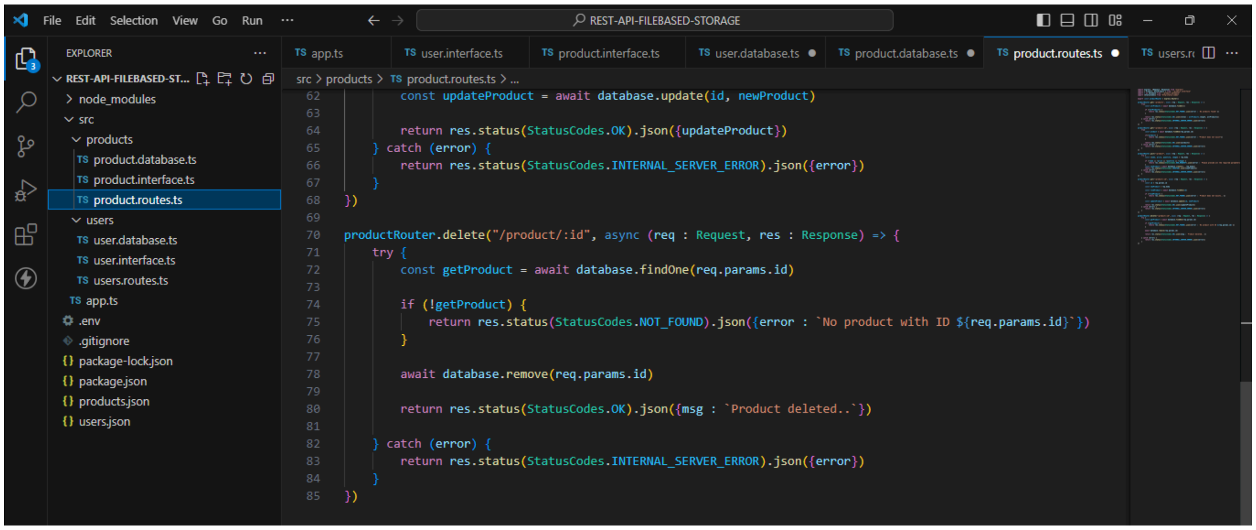
Once our logic checks out, it's time to implement the routes for our products. Populate the ./src/products.routes.ts file with the following code :



```
import express, {Request, Response} from "express"
import { Product, UnitProduct } from "./product.interface"
import * as database from "./product.database"
import {StatusCodes} from "http-status-codes"

export const productRouter = express.Router()

productRouter.get('/products', async (req : Request, res : Response) => {
    try{
        const allProducts = await database.findAll()

        if (!allProducts) {
            return res.status(StatusCodes.NOT_FOUND).json({error : `No products found!`})
        }

        return res.status(StatusCodes.OK).json({total : allProducts.length, allProducts})
    } catch (error) {
        return res.status(StatusCodes.INTERNAL_SERVER_ERROR).json({error})
    }
})

productRouter.get("/product/:id", async (req : Request, res : Response) => {
    try{
        const product = await database.findOne(req.params.id)

        if(!product) {
            return res.status(StatusCodes.NOT_FOUND).json({error : "Product does not exist"})
        }

        return res.status(StatusCodes.OK).json({product})
    } catch (error) {
        return res.status(StatusCodes.INTERNAL_SERVER_ERROR).json({error})
```

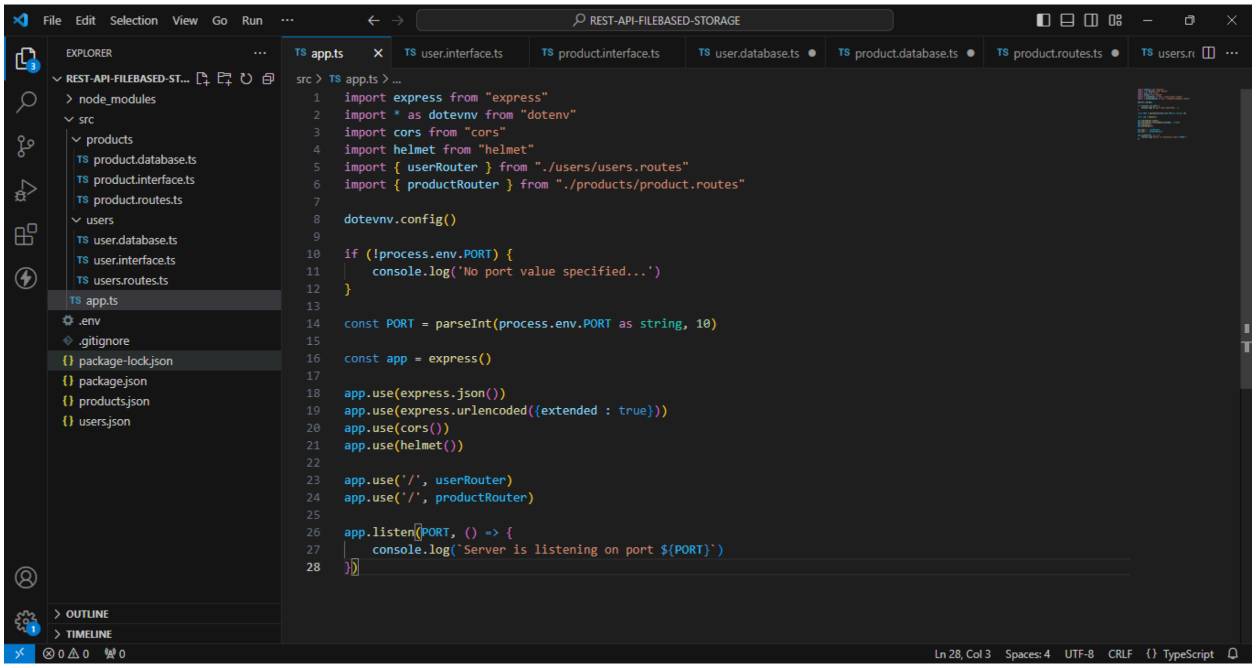


```
        return res.status(StatusCodes.INTERNAL_SERVER_ERROR).json({error})
    }
})

productRouter.post("/product", async (req : Request, res : Response) => {
    try {
        const {name, price, quantity, image} = req.body

        if (!name || !price || !quantity || !image) {
            return res.status(StatusCodes.BAD_REQUEST).json({error : `Please provide all the required parameters
        }
        const newProduct = await database.create({...req.body})
        return res.status(StatusCodes.CREATED).json({newProduct})
    } catch (error) {
        return res.status(StatusCodes.INTERNAL_SERVER_ERROR).json({error})
    }
})

productRouter.put("/product/:id", async (req : Request, res : Response) => {
    try {
        const id = req.params.id

        const newProduct = req.body

        const findProduct = await database.findOne(id)

        if (!findProduct) {
            return res.status(StatusCodes.NOT_FOUND).json({error : `Product does not exists..`})
        }

        const updateProduct = await database.update(id, newProduct)
```



```
        const updateProduct = await database.update(id, newProduct)

        return res.status(StatusCodes.OK).json({updateProduct})
    } catch (error) {
        return res.status(StatusCodes.INTERNAL_SERVER_ERROR).json({error})
    }
})

productRouter.delete("/product/:id", async (req : Request, res : Response) => {
    try {
        const getProduct = await database.findOne(req.params.id)

        if (!getProduct) {
            return res.status(StatusCodes.NOT_FOUND).json({error : `No product with ID ${req.params.id}`})
        }

        await database.remove(req.params.id)

        return res.status(StatusCodes.OK).json({msg : `Product deleted..`})

    } catch (error) {
        return res.status(StatusCodes.INTERNAL_SERVER_ERROR).json({error})
    }
})
```

Finally, to make API calls to these routes we need to import them into our app.ts file and update our code like this :

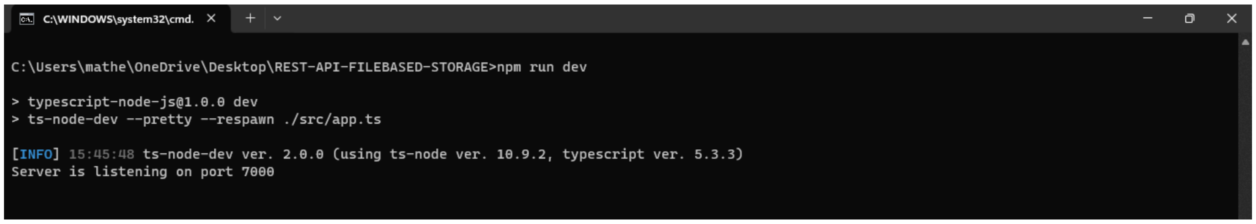


```typescript
import express from "express"
import * as dotevnv from "dotenv"
import cors from "cors"
import helmet from "helmet"
import { userRouter } from "./users/users.routes"
import { productRouter } from "./products/product.routes"

dotevnv.config()

if (!process.env.PORT) {
    console.log('No port value specified...')
}

const PORT = parseInt(process.env.PORT as string, 10)

const app = express()

app.use(express.json())
app.use(express.urlencoded({extended : true}))
app.use(cors())
app.use(helmet())

app.use('/', userRouter)
app.use('/', productRouter)

app.listen(PORT, () => {
    console.log(`Server is listening on port ${PORT}`)
})
```

Step 5: Testing the API

Start the server and test our API using Thunder Client (VS Code Extension).
Note: You can use any other app to test the API if you had used a different IDE.

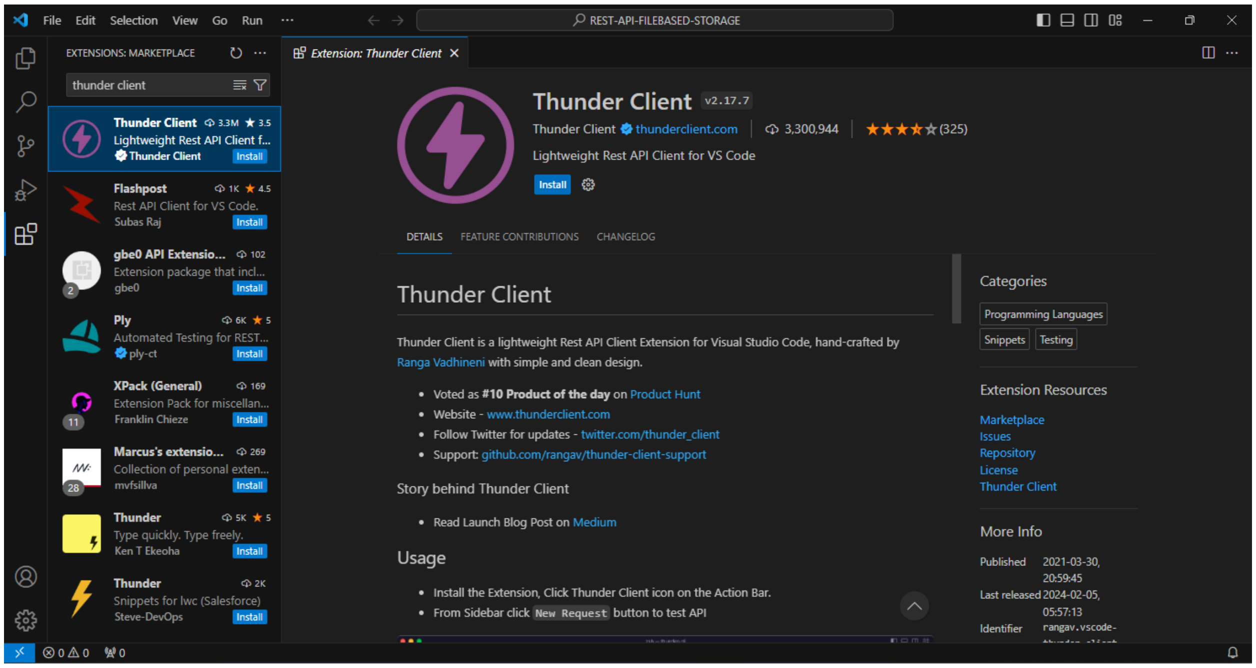
- run the npm run dev command in your terminal
- If there are no errors, the server should be listening to port 7000



```
C:\Users\mathe\OneDrive\Desktop\REST-API-FILEBASED-STORAGE>npm run dev

> typescript-node-js@1.0.0 dev
> ts-node-dev --pretty --respawn ./src/app.ts

[INFO] 15:45:48 ts-node-dev ver. 2.0.0 (using ts-node ver. 10.9.2, typescript ver. 5.3.3)
Server is listening on port 7000
```

Install Thunder Client

- Click the Extensions tab on the left sidebar
- Search the thunder client extension
- And click the install button.



Thunder Client 3.3M ★ 3.5
Lightweight Rest API Client f...
Thunder Client Install

Flashpost 1K ★ 4.5
Rest API Client for VS Code.
Subas Raj Install

gbe0 API Extensio... 102
Extension package that incl...
gbe0 Install

Ply 6K ★ 5
Automated Testing for REST...
ply-ct Install

XPack (General) 169
Extension Pack for miscellan...
Franklin Chieze Install

Marcus's extensio... 269
Collection of personal exten...
mvfsillva Install

Thunder 5K ★ 5
Type quickly. Type freely.
Ken T Ekeoha Install

Thunder 2K
Snippets for lwc (Salesforce)
Steve-DevOps Install

Thunder Client v2.17.7
Thunder Client thunderclient.com | 3,300,944 | ★★★★☆ (325)
Lightweight Rest API Client for VS Code

Thunder Client

Thunder Client is a lightweight Rest API Client Extension for Visual Studio Code, hand-crafted by Ranga Vadhineni with simple and clean design.

- Voted as **#10 Product of the day** on Product Hunt
- Website - www.thunderclient.com
- Follow Twitter for updates - twitter.com/thunder_client
- Support: github.com/rangav/thunder-client-support

Story behind Thunder Client

- Read Launch Blog Post on Medium

Usage

- Install the Extension, Click Thunder Client icon on the Action Bar.
- From Sidebar click `New Request` button to test API

Categories
Programming Languages
Snippets
Testing

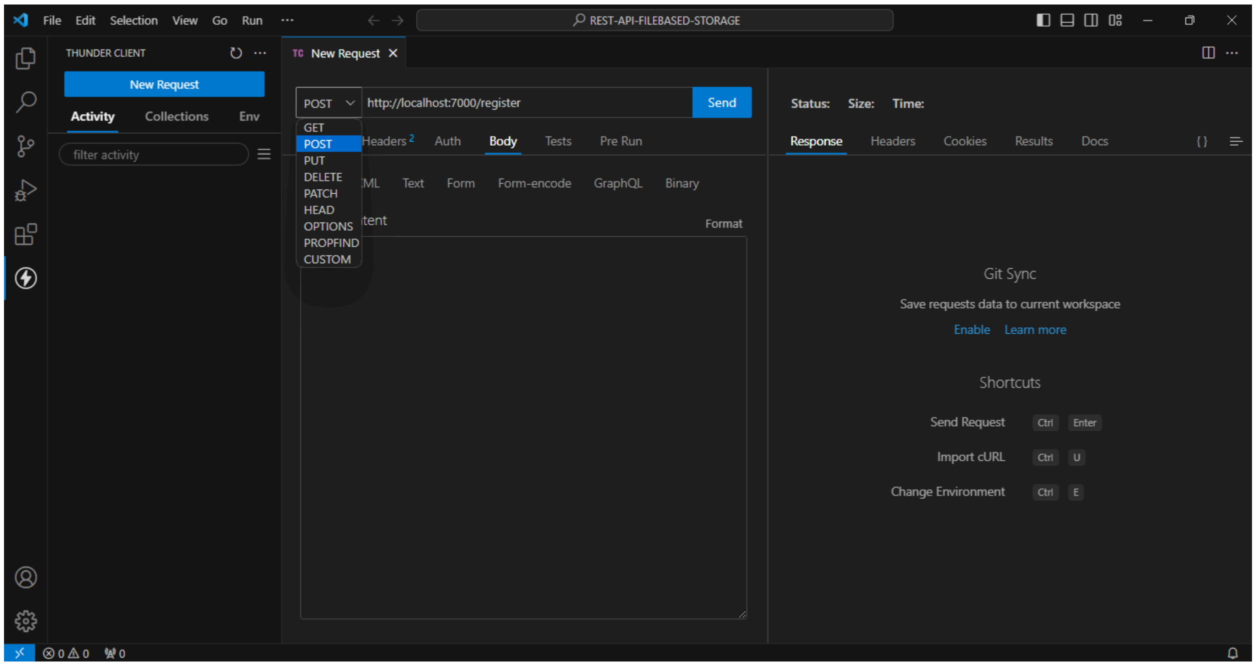
Extension Resources
Marketplace
Issues
Repository
License
Thunder Client

More Info

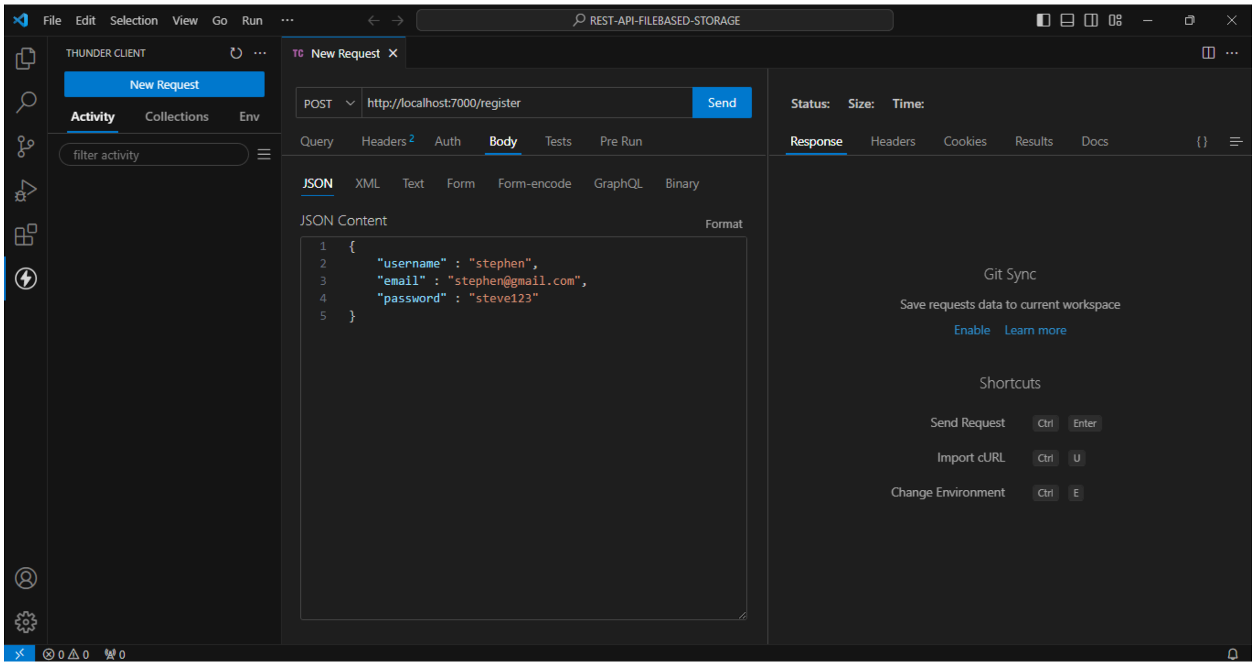
Published	2021-03-30, 20:59:45
Last released	2024-02-05, 05:57:13
Identifier	rangav.vscode-

Making Requests (Thunder Client)

- Click on the new request button
- You can configure the request url and what type of request you want to send on the page next to the activity/collections/env tab.
- If you are done with the configurations then click the send button.

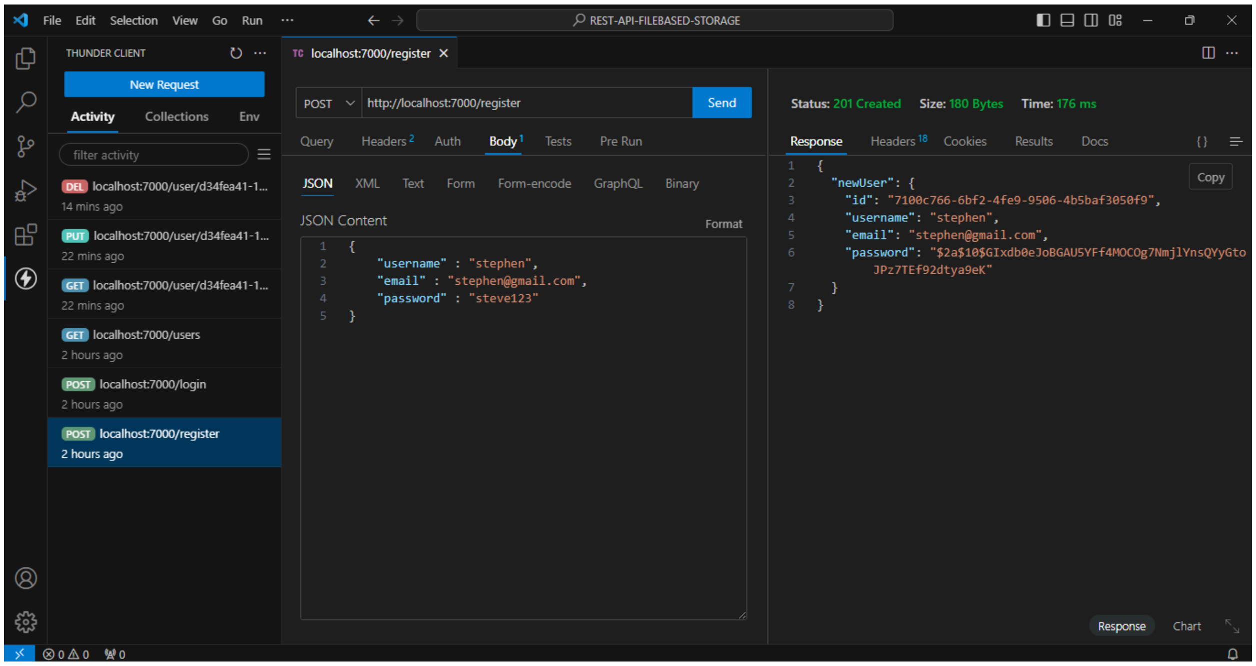

File Edit Selection View Go Run ···
REST-API-FILEBASED-STORAGE
THUNDER CLIENT
New Request
Activity Collections Env
filter activity
TC New Request
POST
http://localhost:7000/register
Send
GET
POST
PUT
DELETE
PATCH
HEAD
OPTIONS
PROPFIND
CUSTOM
Headers 2 Auth Body Tests Pre Run
XML Text Form Form-encode GraphQL Binary
Format
Status: Size: Time:
Response Headers Cookies Results Docs
Git Sync
Save requests data to current workspace
Enable Learn more
Shortcuts
Send Request Ctrl Enter
Import cURL Ctrl U
Change Environment Ctrl E

Register a user


File Edit Selection View Go Run ···
REST-API-FILEBASED-STORAGE
THUNDER CLIENT
New Request
Activity Collections Env
filter activity
TC New Request
POST
http://localhost:7000/register
Send
Query Headers 2 Auth Body Tests Pre Run
JSON XML Text Form Form-encode GraphQL Binary
JSON Content
Format
Status: Size: Time:
Response Headers Cookies Results Docs
Git Sync
Save requests data to current workspace
Enable Learn more
Shortcuts
Send Request Ctrl Enter
Import cURL Ctrl U
Change Environment Ctrl E

```json
{
    "username" : "stephen",
    "email" : "stephen@gmail.com",
    "password" : "steve123"
}
```

Register a user (Response after sending the request)



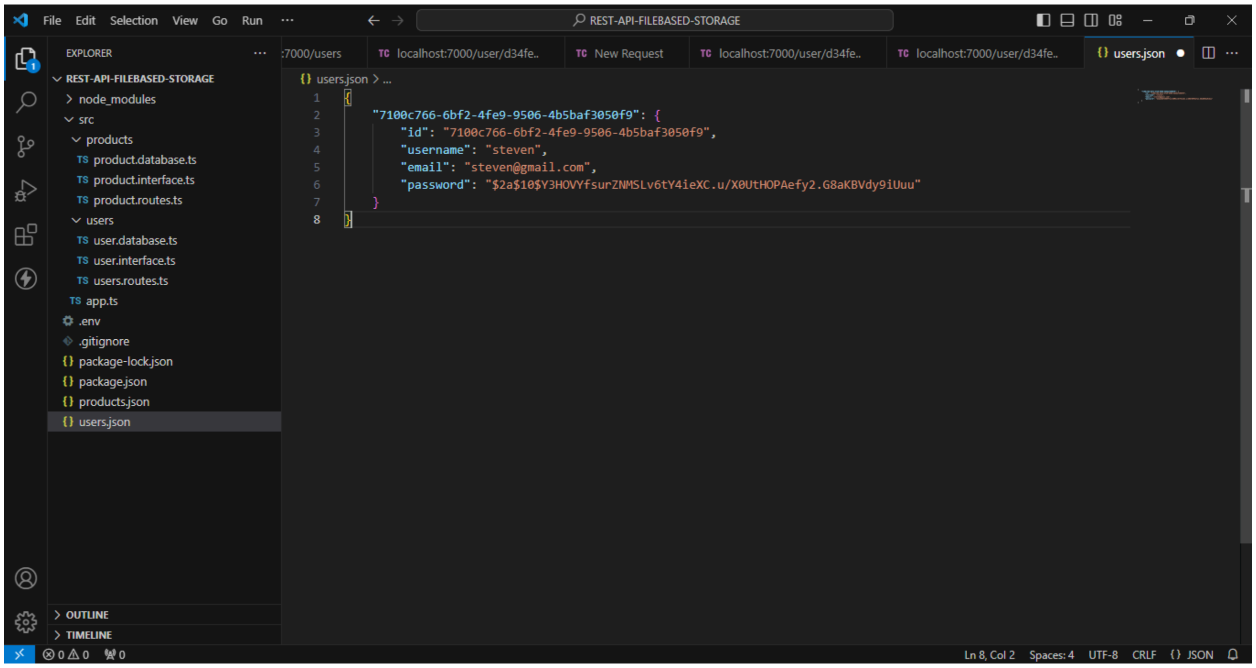
POST http://localhost:7000/register

```
{
    "username" : "stephen",
    "email" : "stephen@gmail.com",
    "password" : "steve123"
}
```

Status: 201 Created Size: 180 Bytes Time: 176 ms

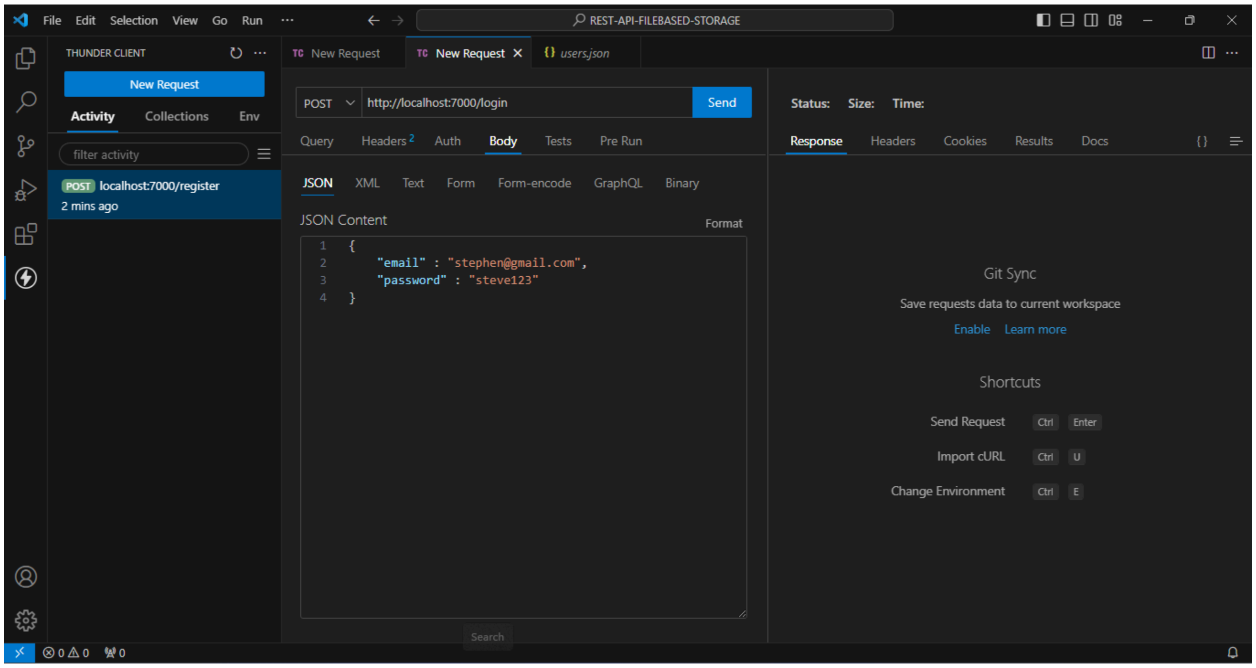
```
{
  "newUser": {
    "id": "7100c766-6bf2-4fe9-9506-4b5baf3050f9",
    "username": "stephen",
    "email": "stephen@gmail.com",
    "password": "$2a$10$GIxdb0eJoBGAU5YFf4MOCOg7NmjlYnsQYyGto
      JPz7TEf92dtya9eK"
  }
}
```

Register a user (Changes reflected on the users.json file after request has been done)



```
{
    "7100c766-6bf2-4fe9-9506-4b5baf3050f9": {
        "id": "7100c766-6bf2-4fe9-9506-4b5baf3050f9",
        "username": "steven",
        "email": "steven@gmail.com",
        "password": "$2a$10$Y3HOVYfsurZNMSLv6tY4ieXC.u/X0UtHOPAefy2.G8aKBVdy9iUuu"
    }
}
```

Login user



POST localhost:7000/register
2 mins ago

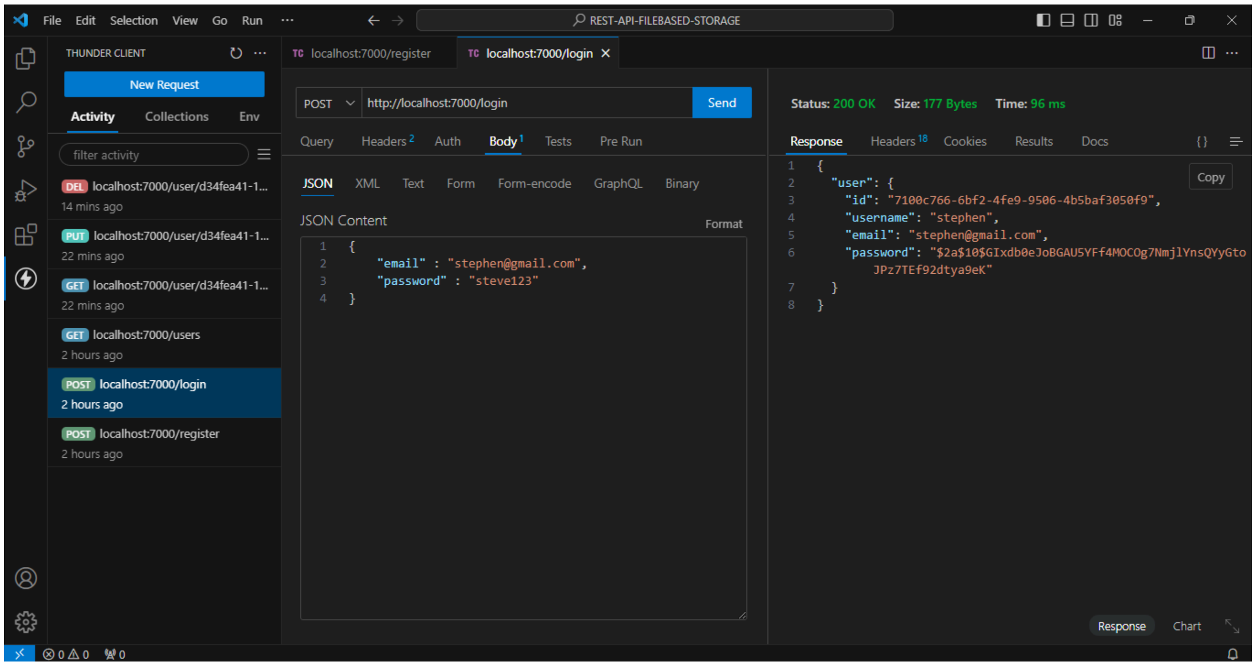
```
POST  http://localhost:7000/login  Send
```

JSON Content

```
{
    "email" : "stephen@gmail.com",
    "password" : "steve123"
}
```

Status: Size: Time:

Login user (Response after sending the request)



DEL localhost:7000/user/d34fea41-1...
14 mins ago
PUT localhost:7000/user/d34fea41-1...
22 mins ago
GET localhost:7000/user/d34fea41-1...
22 mins ago
GET localhost:7000/users
2 hours ago
POST localhost:7000/login
2 hours ago
POST localhost:7000/register
2 hours ago

```
POST  http://localhost:7000/login  Send
```

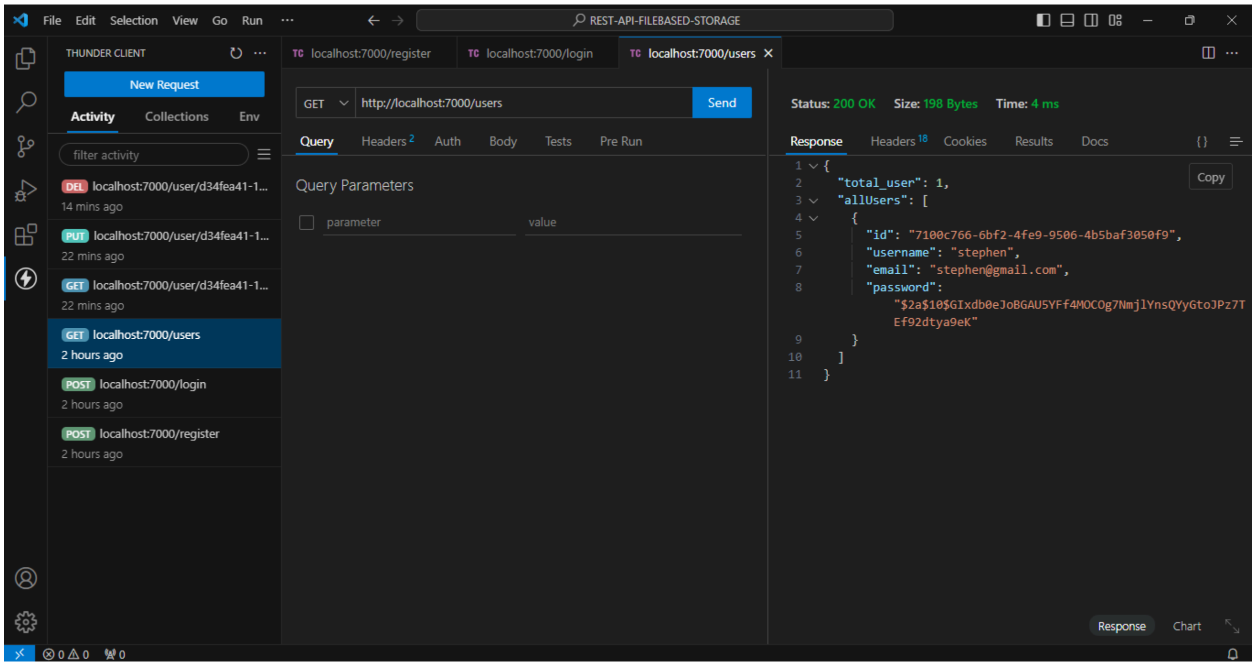
JSON Content

```
{
    "email" : "stephen@gmail.com",
    "password" : "steve123"
}
```

Status: 200 OK Size: 177 Bytes Time: 96 ms

```
{
  "user": {
    "id": "7100c766-6bf2-4fe9-9506-4b5baf3050f9",
    "username": "stephen",
    "email": "stephen@gmail.com",
    "password": "$2a$10$GIxdb0eJoBGAU5YFf4MOCOg7NmjlYnsQYyGto
        JPz7TEf92dtya9eK"
  }
}
```

Get all users (List all the registered users)

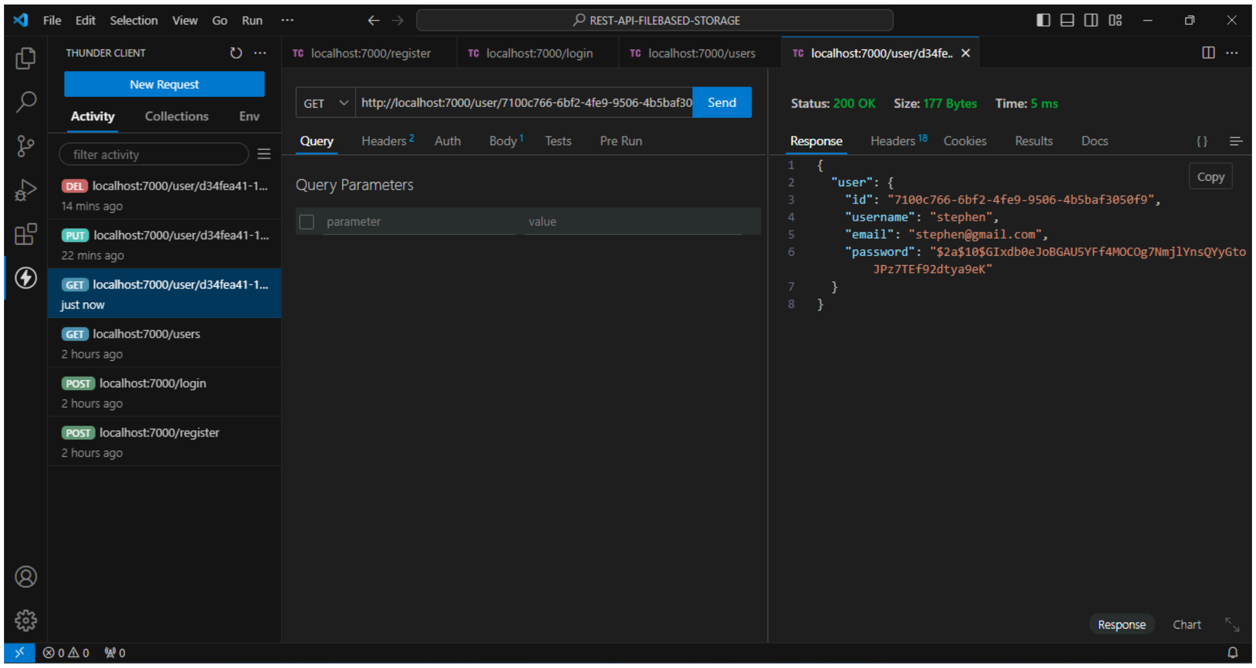


GET http://localhost:7000/users

Status: 200 OK Size: 198 Bytes Time: 4 ms

```
{
  "total_user": 1,
  "allUsers": [
    {
      "id": "7100c766-6bf2-4fe9-9506-4b5baf3050f9",
      "username": "stephen",
      "email": "stephen@gmail.com",
      "password":
          "$2a$10$GIxdb0eJoBGAU5YFf4MOCOg7NmjlYnsQYyGtoJPz7T
          Ef92dtya9eK"
    }
  ]
}
```

Get a single user (By ID)

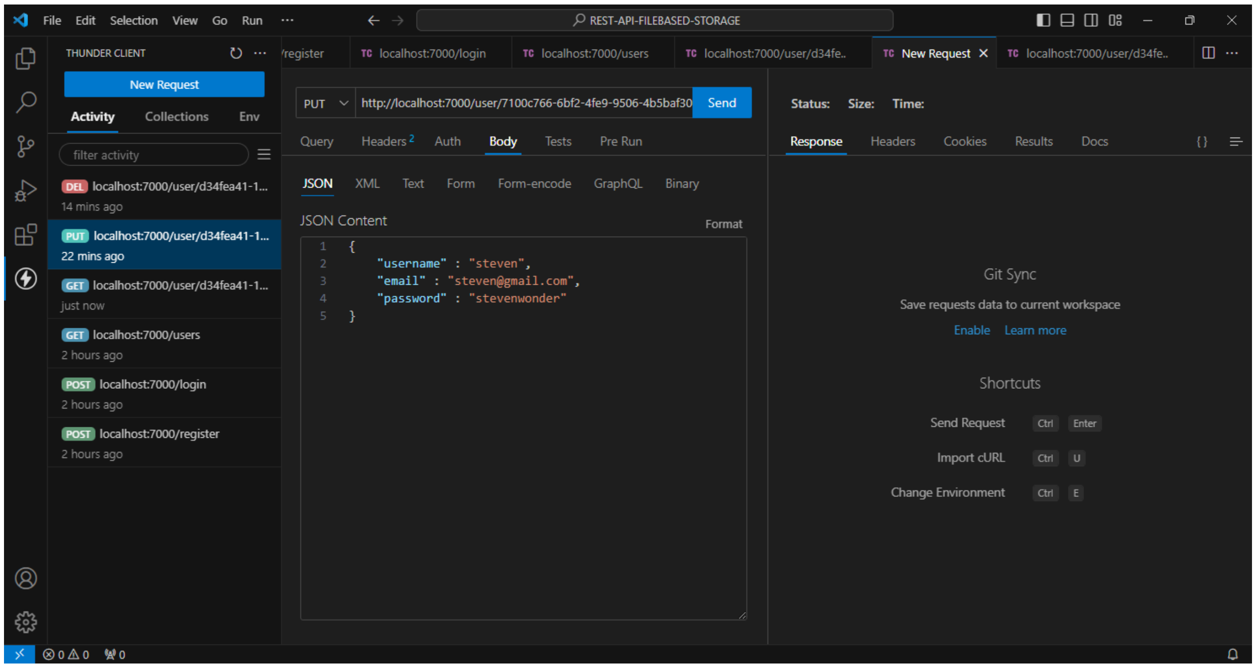


GET http://localhost:7000/user/7100c766-6bf2-4fe9-9506-4b5baf30

Status: 200 OK Size: 177 Bytes Time: 5 ms

```
{
  "user": {
    "id": "7100c766-6bf2-4fe9-9506-4b5baf3050f9",
    "username": "stephen",
    "email": "stephen@gmail.com",
    "password": "$2a$10$GIxdb0eJoBGAU5YFf4MOCOg7NmjlYnsQYyGto
        JPz7TEf92dtya9eK"
  }
}
```

Update a User (By ID)


File Edit Selection View Go Run ···
REST-API-FILEBASED-STORAGE
THUNDER CLIENT
New Request
Activity Collections Env
filter activity
Query Headers 2 Auth Body Tests Pre Run
JSON XML Text Form Form-encode GraphQL Binary
Response Headers Cookies Results Docs
Git Sync
Save requests data to current workspace
Enable Learn more
Shortcuts
Send Request Ctrl Enter
Import cURL Ctrl U
Change Environment Ctrl E

DEL localhost:7000/user/d34fea41-1...
14 mins ago
PUT localhost:7000/user/d34fea41-1...
22 mins ago
GET localhost:7000/user/d34fea41-1...
just now
GET localhost:7000/users
2 hours ago
POST localhost:7000/login
2 hours ago
POST localhost:7000/register
2 hours ago

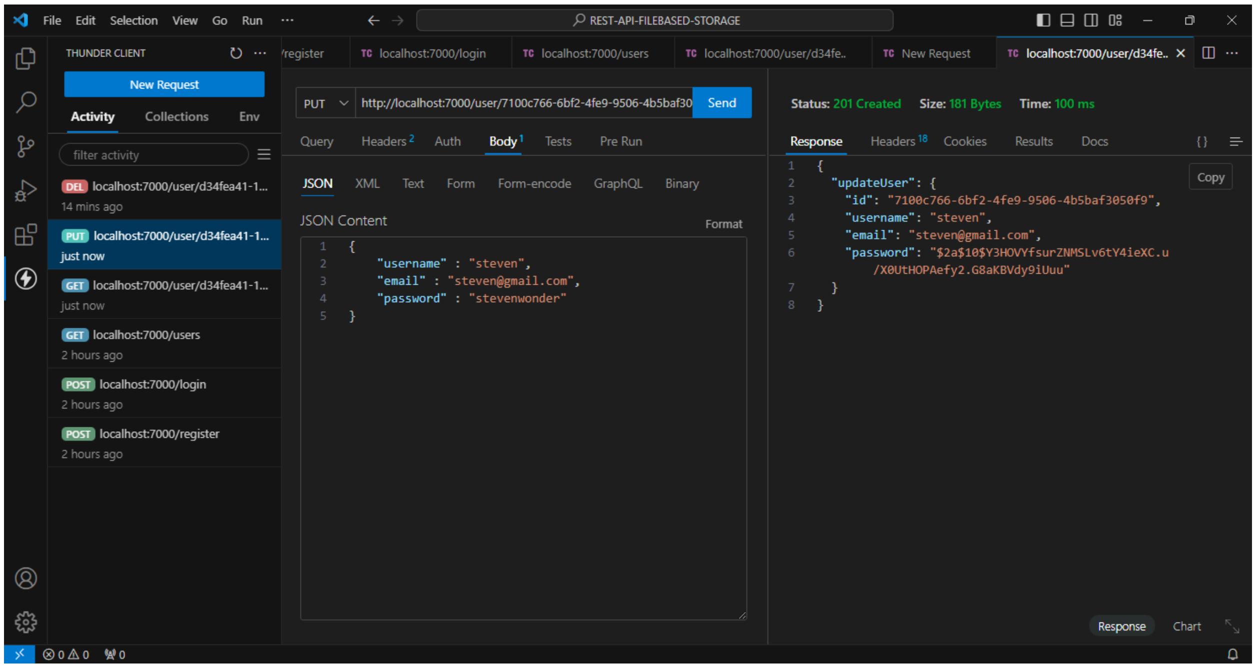
PUT http://localhost:7000/user/7100c766-6bf2-4fe9-9506-4b5baf30 Send

JSON Content

```
{
    "username" : "steven",
    "email" : "steven@gmail.com",
    "password" : "stevenwonder"
}
```

Status: Size: Time:

Update a User (Response after sending the request)


File Edit Selection View Go Run ···
REST-API-FILEBASED-STORAGE
THUNDER CLIENT
New Request
Activity Collections Env
filter activity
Query Headers 2 Auth Body 1 Tests Pre Run
JSON XML Text Form Form-encode GraphQL Binary
Response Headers 18 Cookies Results Docs
Copy
Response Chart

DEL localhost:7000/user/d34fea41-1...
14 mins ago
PUT localhost:7000/user/d34fea41-1...
just now
GET localhost:7000/user/d34fea41-1...
just now
GET localhost:7000/users
2 hours ago
POST localhost:7000/login
2 hours ago
POST localhost:7000/register
2 hours ago

PUT http://localhost:7000/user/7100c766-6bf2-4fe9-9506-4b5baf30 Send

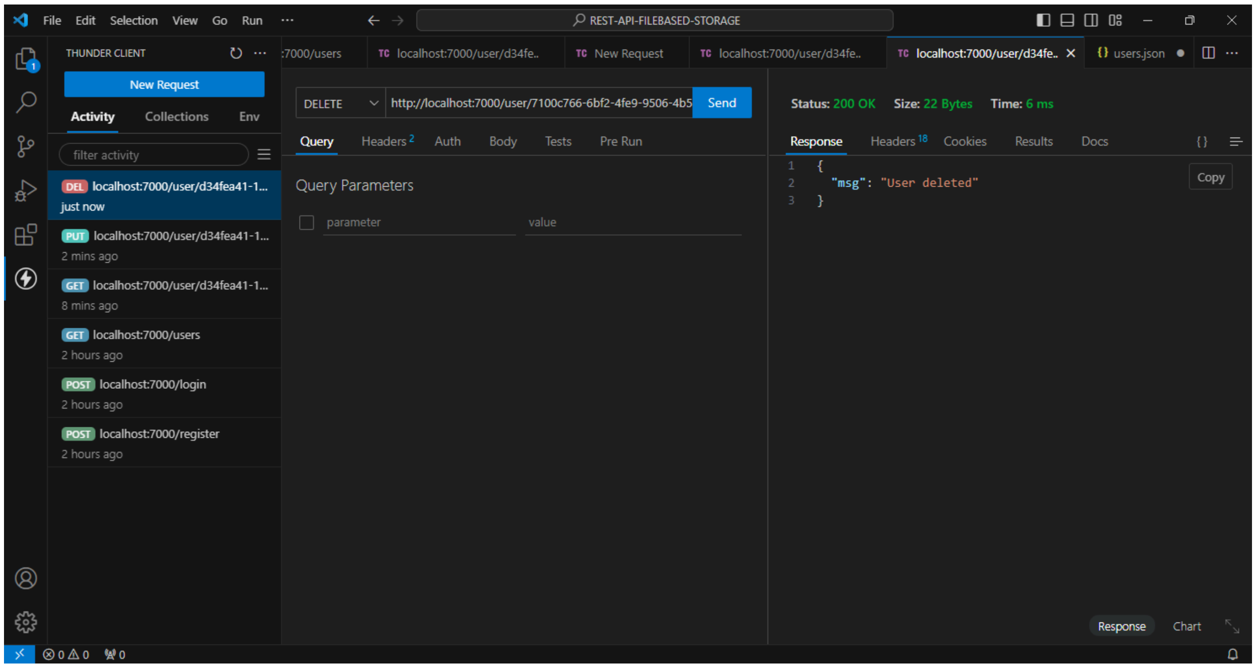
JSON Content

```
{
    "username" : "steven",
    "email" : "steven@gmail.com",
    "password" : "stevenwonder"
}
```

Status: 201 Created Size: 181 Bytes Time: 100 ms

```
{
  "updateUser": {
    "id": "7100c766-6bf2-4fe9-9506-4b5baf3050f9",
    "username": "steven",
    "email": "steven@gmail.com",
    "password": "$2a$10$Y3HOVYfsurZNMSLv6tY4ieXC.u/X0UtHOPAefy2.G8aKBVdy9iUuu"
  }
}
```

Delete a User (By ID)


File Edit Selection View Go Run ...
REST-API-FILEBASED-STORAGE
THUNDER CLIENT
New Request
Activity Collections Env
filter activity
Query Headers 2 Auth Body Tests Pre Run
Response Headers 18 Cookies Results Docs
Copy
Response Chart

```
DEL  localhost:7000/user/d34fea41-1...   just now
PUT  localhost:7000/user/d34fea41-1...   2 mins ago
GET  localhost:7000/user/d34fea41-1...   8 mins ago
GET  localhost:7000/users                2 hours ago
POST localhost:7000/login                2 hours ago
POST localhost:7000/register             2 hours ago
```

```
DELETE  http://localhost:7000/user/7100c766-6bf2-4fe9-9506-4b5   Send
```

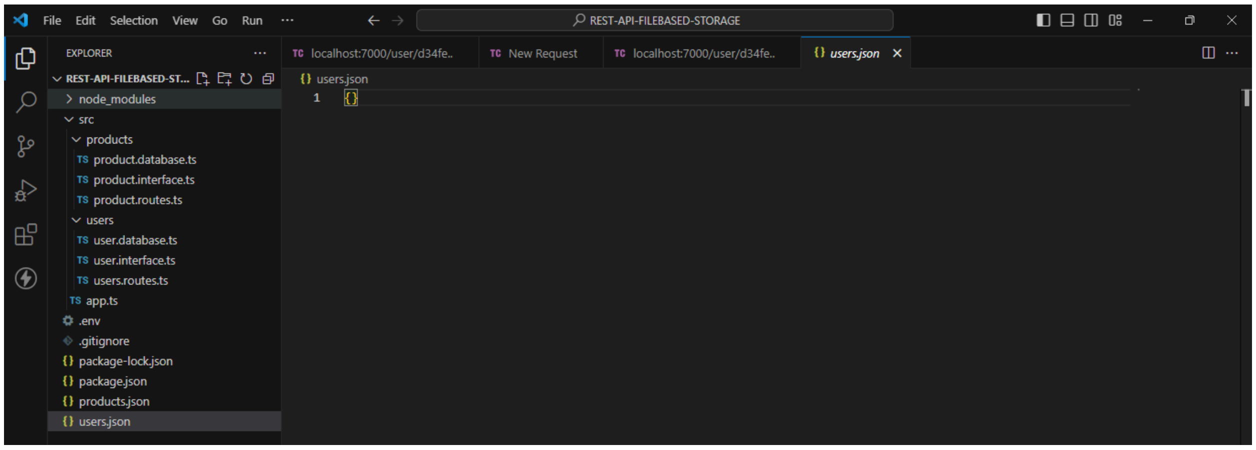
Query Parameters

parameter	value

```
Status: 200 OK   Size: 22 Bytes   Time: 6 ms
```

```
{
  "msg": "User deleted"
}
```

Delete a User (Changes reflected on the users.json file after request has been done)

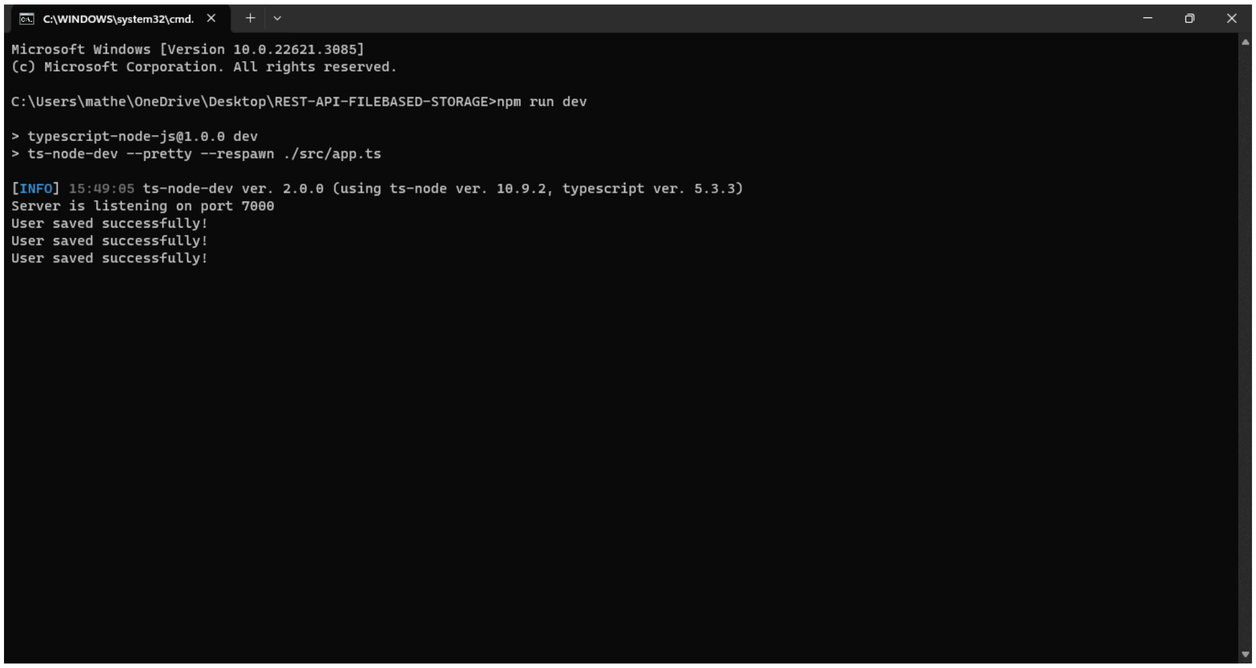

File Edit Selection View Go Run ...
REST-API-FILEBASED-STORAGE
EXPLORER
localhost:7000/user/d34fe..
New Request
localhost:7000/user/d34fe..
users.json

```
REST-API-FILEBASED-ST...
> node_modules
v src
  v products
    product.database.ts
    product.interface.ts
    product.routes.ts
  v users
    user.database.ts
    user.interface.ts
    users.routes.ts
  app.ts
.env
.gitignore
package-lock.json
package.json
products.json
users.json
```

users.json

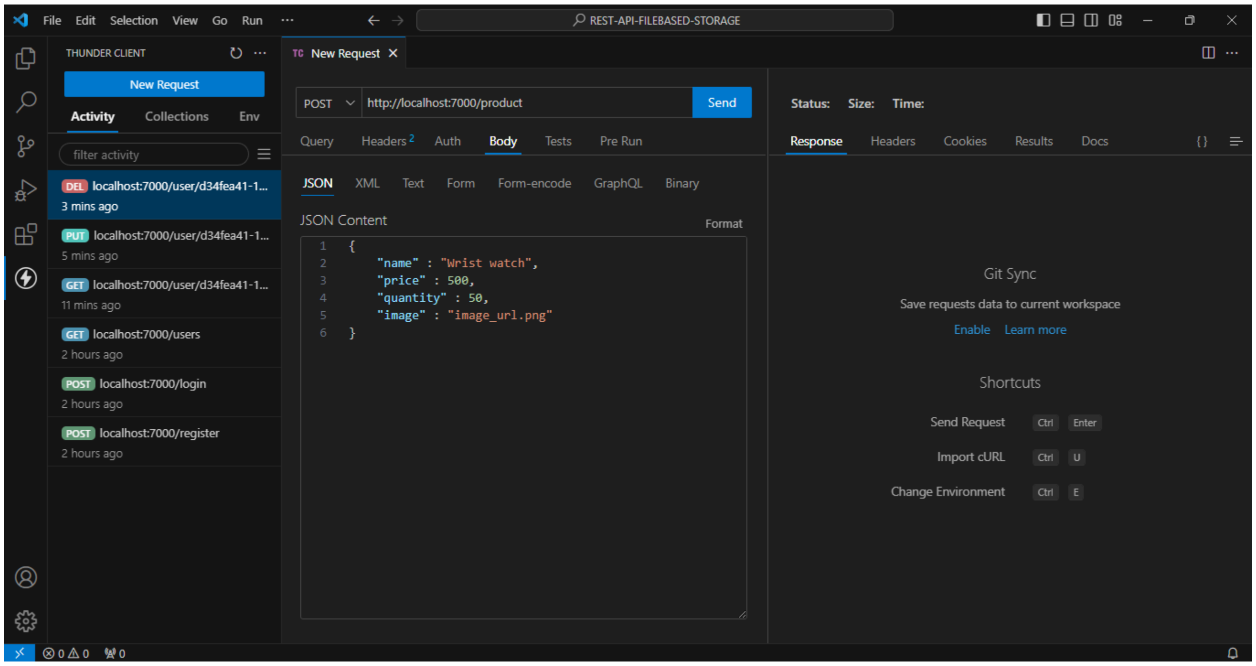
```
{}
```

Terminal Logs (Create, Update, Delete changes on users.json)



```
Microsoft Windows [Version 10.0.22621.3085]
(c) Microsoft Corporation. All rights reserved.

C:\Users\mathe\OneDrive\Desktop\REST-API-FILEBASED-STORAGE>npm run dev

> typescript-node-js@1.0.0 dev
> ts-node-dev --pretty --respawn ./src/app.ts

[INFO] 15:49:05 ts-node-dev ver. 2.0.0 (using ts-node ver. 10.9.2, typescript ver. 5.3.3)
Server is listening on port 7000
User saved successfully!
User saved successfully!
User saved successfully!
```

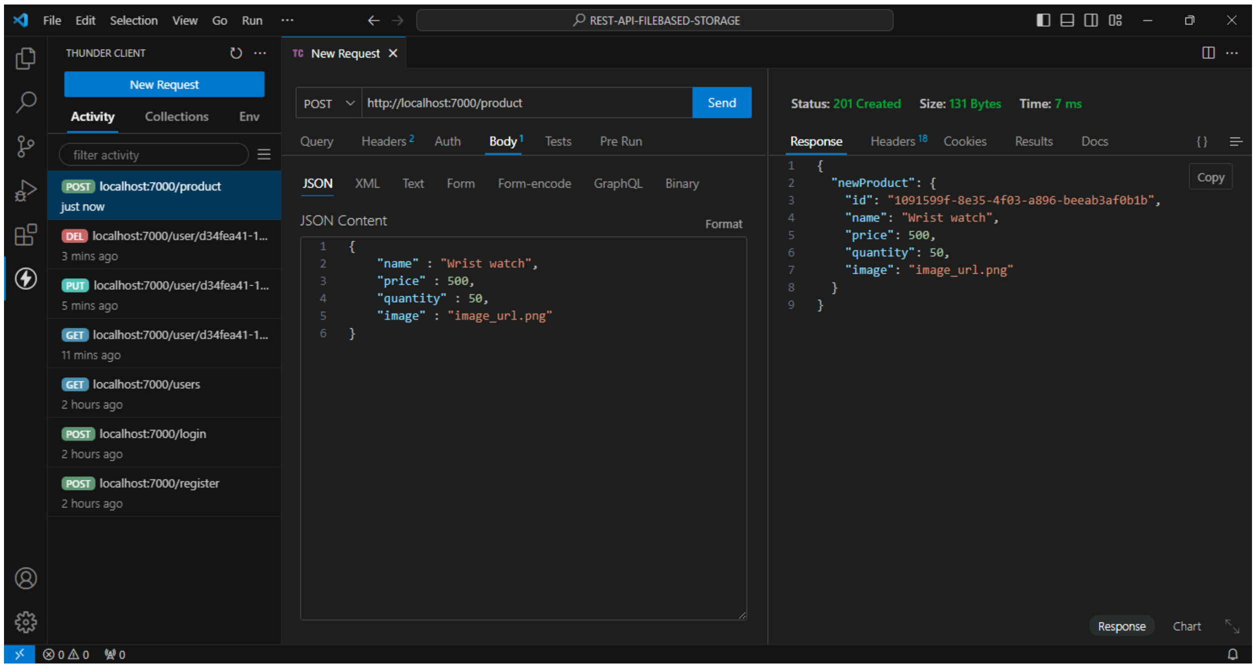
Create a product


File Edit Selection View Go Run
REST-API-FILEBASED-STORAGE
THUNDER CLIENT
New Request
Activity Collections Env
filter activity
DEL localhost:7000/user/d34fea41-1... 3 mins ago
PUT localhost:7000/user/d34fea41-1... 5 mins ago
GET localhost:7000/user/d34fea41-1... 11 mins ago
GET localhost:7000/users 2 hours ago
POST localhost:7000/login 2 hours ago
POST localhost:7000/register 2 hours ago
New Request
Query Headers 2 Auth Body Tests Pre Run
JSON XML Text Form Form-encode GraphQL Binary
JSON Content
Format
Status: Size: Time:
Response Headers Cookies Results Docs
Git Sync
Save requests data to current workspace
Enable Learn more
Shortcuts
Send Request Ctrl Enter
Import cURL Ctrl U
Change Environment Ctrl E

POST http://localhost:7000/product Send

```
{
    "name" : "Wrist watch",
    "price" : 500,
    "quantity" : 50,
    "image" : "image_url.png"
}
```

Create a product (Response after sending the request)



POST localhost:7000/product just now
DEL localhost:7000/user/d34fea41-1... 3 mins ago
PUT localhost:7000/user/d34fea41-1... 5 mins ago
GET localhost:7000/user/d34fea41-1... 11 mins ago
GET localhost:7000/users 2 hours ago
POST localhost:7000/login 2 hours ago
POST localhost:7000/register 2 hours ago

POST http://localhost:7000/product Send

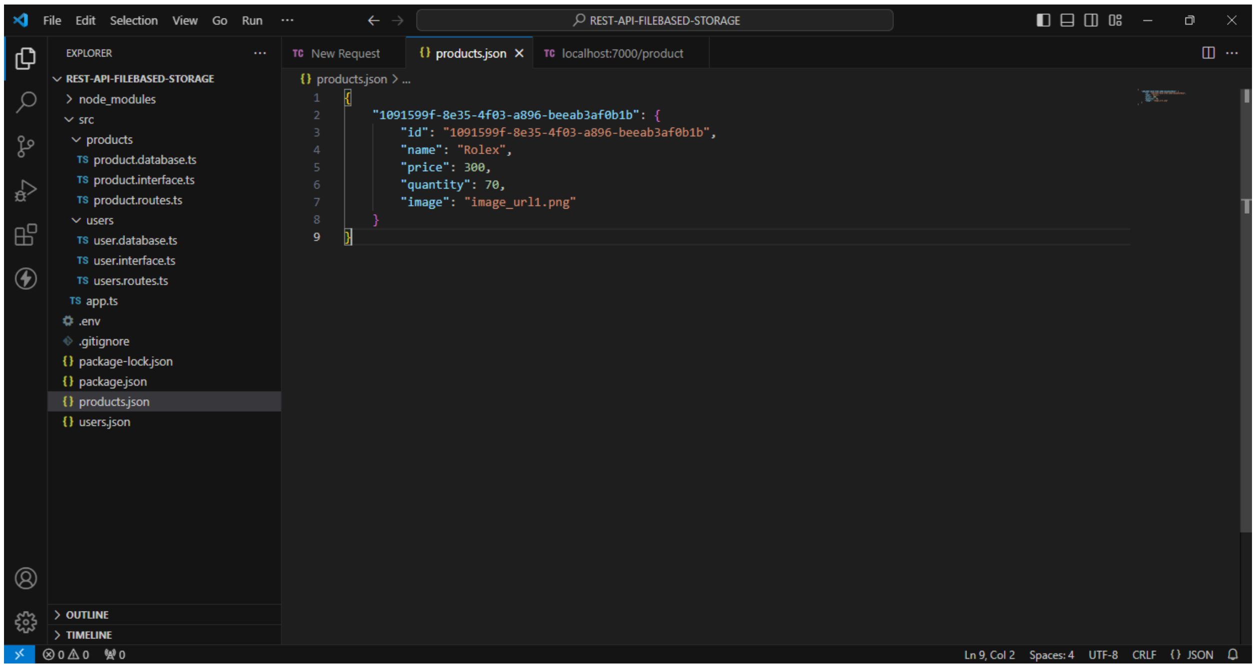
JSON Content

```
{
    "name" : "Wrist watch",
    "price" : 500,
    "quantity" : 50,
    "image" : "image_url.png"
}
```

Status: 201 Created Size: 131 Bytes Time: 7 ms

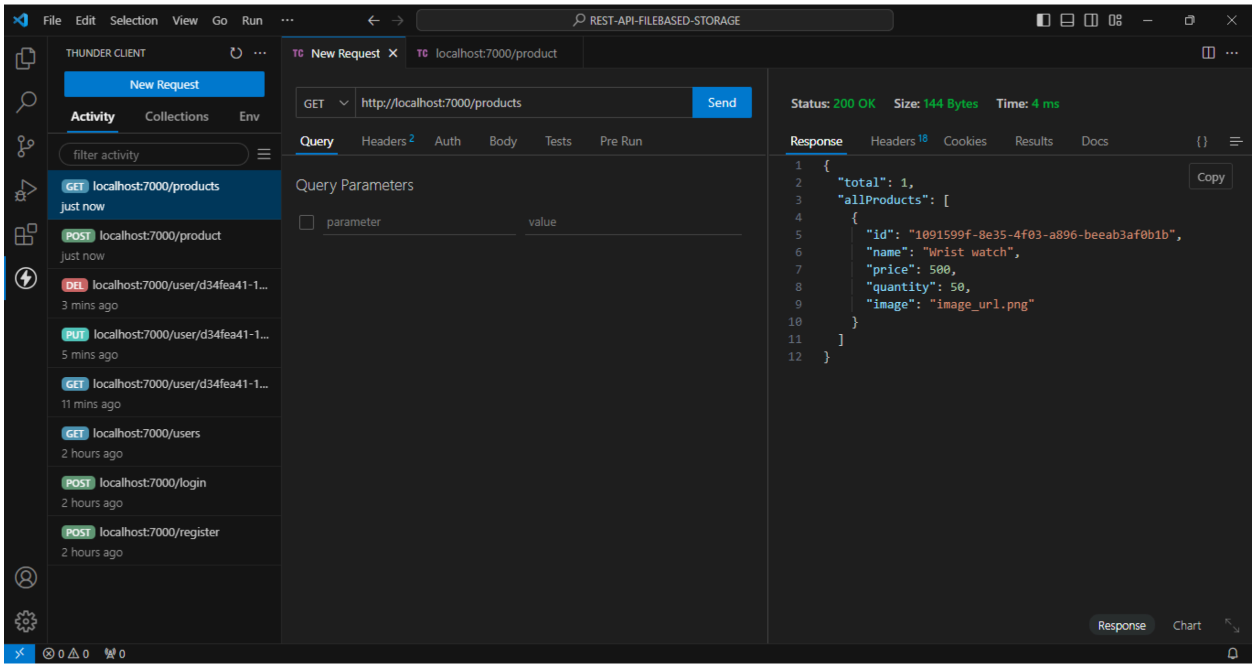
```
{
  "newProduct": {
    "id": "1091599f-8e35-4f03-a896-beeab3af0b1b",
    "name": "Wrist watch",
    "price": 500,
    "quantity": 50,
    "image": "image_url.png"
  }
}
```

Create a product (Changes reflected on the products.json file after request has been done)



```
{
    "1091599f-8e35-4f03-a896-beeab3af0b1b": {
        "id": "1091599f-8e35-4f03-a896-beeab3af0b1b",
        "name": "Rolex",
        "price": 300,
        "quantity": 70,
        "image": "image_url1.png"
    }
}
```

Get all products (List all the created products)

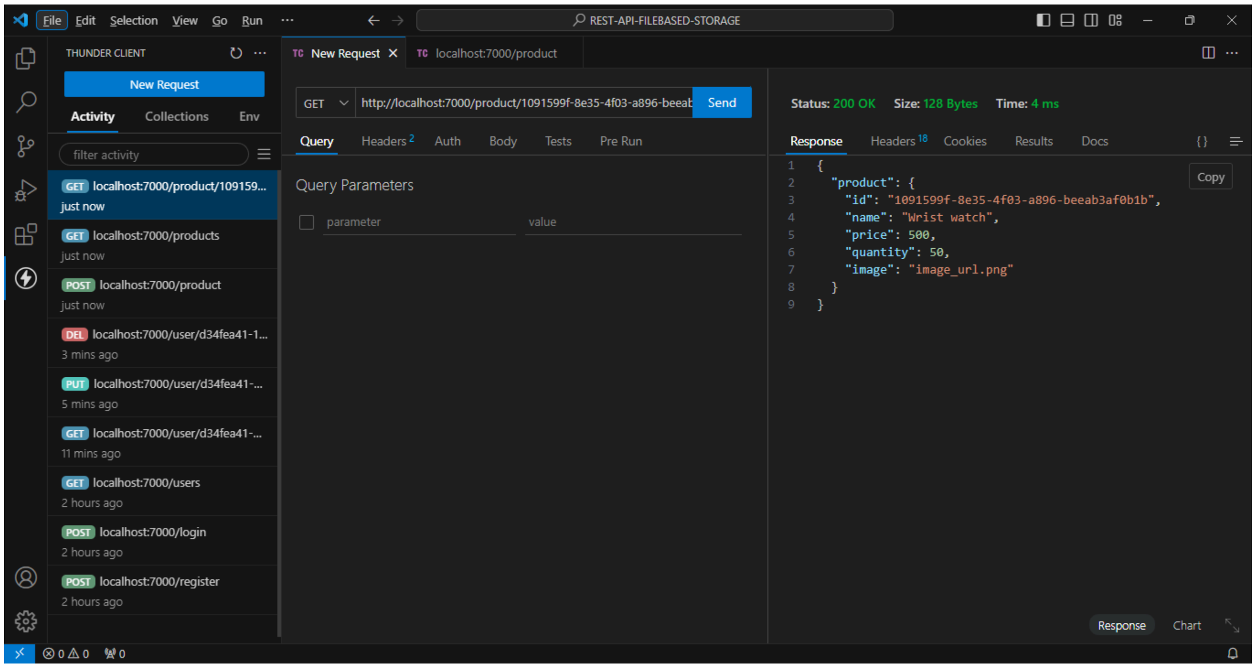

File Edit Selection View Go Run ···
REST-API-FILEBASED-STORAGE
THUNDER CLIENT
New Request
Activity Collections Env
filter activity
GET localhost:7000/products just now
POST localhost:7000/product just now
DEL localhost:7000/user/d34fea41-1... 3 mins ago
PUT localhost:7000/user/d34fea41-1... 5 mins ago
GET localhost:7000/user/d34fea41-1... 11 mins ago
GET localhost:7000/users 2 hours ago
POST localhost:7000/login 2 hours ago
POST localhost:7000/register 2 hours ago
New Request
localhost:7000/product
Send
Query Headers 2 Auth Body Tests Pre Run
Query Parameters
parameter
value
Response Headers 18 Cookies Results Docs
Copy
Response Chart

GET http://localhost:7000/products

Status: 200 OK Size: 144 Bytes Time: 4 ms

```
{
  "total": 1,
  "allProducts": [
    {
      "id": "1091599f-8e35-4f03-a896-beeab3af0b1b",
      "name": "Wrist watch",
      "price": 500,
      "quantity": 50,
      "image": "image_url.png"
    }
  ]
}
```

Get a single product (By ID)

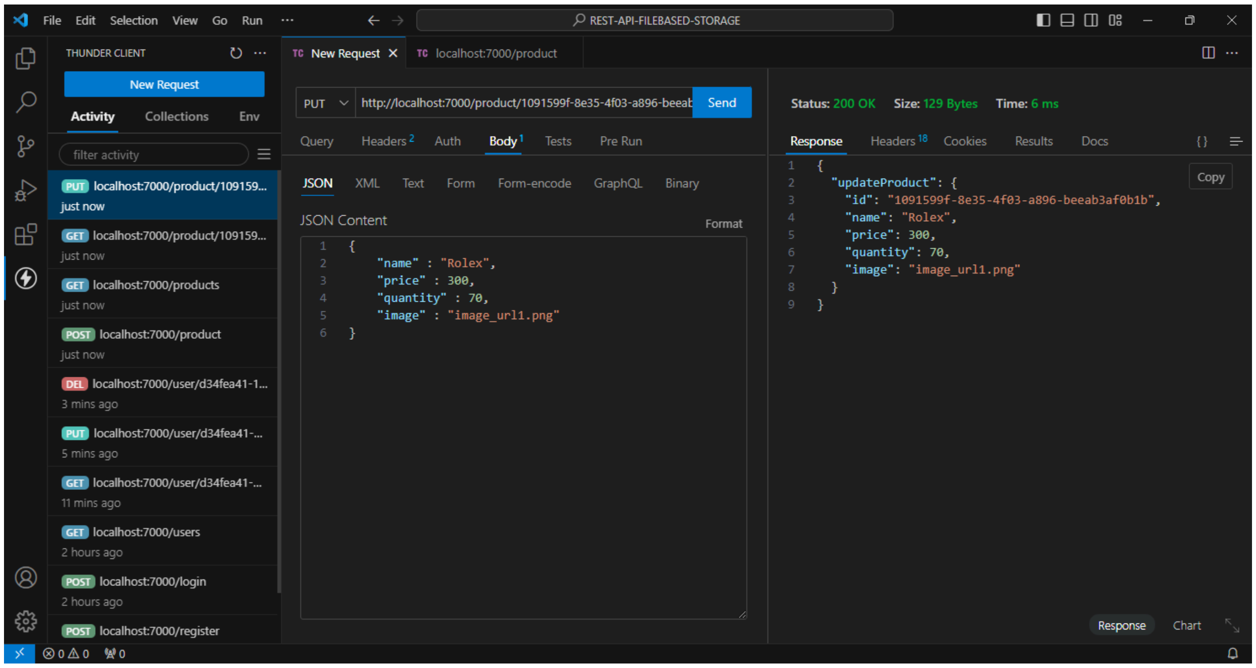

File Edit Selection View Go Run ···
REST-API-FILEBASED-STORAGE
THUNDER CLIENT
New Request
Activity Collections Env
filter activity
GET localhost:7000/product/109159... just now
GET localhost:7000/products just now
POST localhost:7000/product just now
DEL localhost:7000/user/d34fea41-1... 3 mins ago
PUT localhost:7000/user/d34fea41-... 5 mins ago
GET localhost:7000/user/d34fea41-... 11 mins ago
GET localhost:7000/users 2 hours ago
POST localhost:7000/login 2 hours ago
POST localhost:7000/register 2 hours ago
New Request
localhost:7000/product
Send
Query Headers 2 Auth Body Tests Pre Run
Query Parameters
parameter
value
Response Headers 18 Cookies Results Docs
Copy
Response Chart

GET http://localhost:7000/product/1091599f-8e35-4f03-a896-beeab

Status: 200 OK Size: 128 Bytes Time: 4 ms

```
{
  "product": {
    "id": "1091599f-8e35-4f03-a896-beeab3af0b1b",
    "name": "Wrist watch",
    "price": 500,
    "quantity": 50,
    "image": "image_url.png"
  }
}
```

Update a product (By ID)


File Edit Selection View Go Run ···
REST-API-FILEBASED-STORAGE
THUNDER CLIENT
New Request
Activity Collections Env
filter activity
PUT localhost:7000/product/109159... just now
GET localhost:7000/product/109159... just now
GET localhost:7000/products just now
POST localhost:7000/product just now
DEL localhost:7000/user/d34fea41-1... 3 mins ago
PUT localhost:7000/user/d34fea41-... 5 mins ago
GET localhost:7000/user/d34fea41-... 11 mins ago
GET localhost:7000/users 2 hours ago
POST localhost:7000/login 2 hours ago
POST localhost:7000/register
New Request
localhost:7000/product
Query Headers 2 Auth Body 1 Tests Pre Run
JSON XML Text Form Form-encode GraphQL Binary
JSON Content Format
Response Headers 18 Cookies Results Docs
Copy
Response Chart

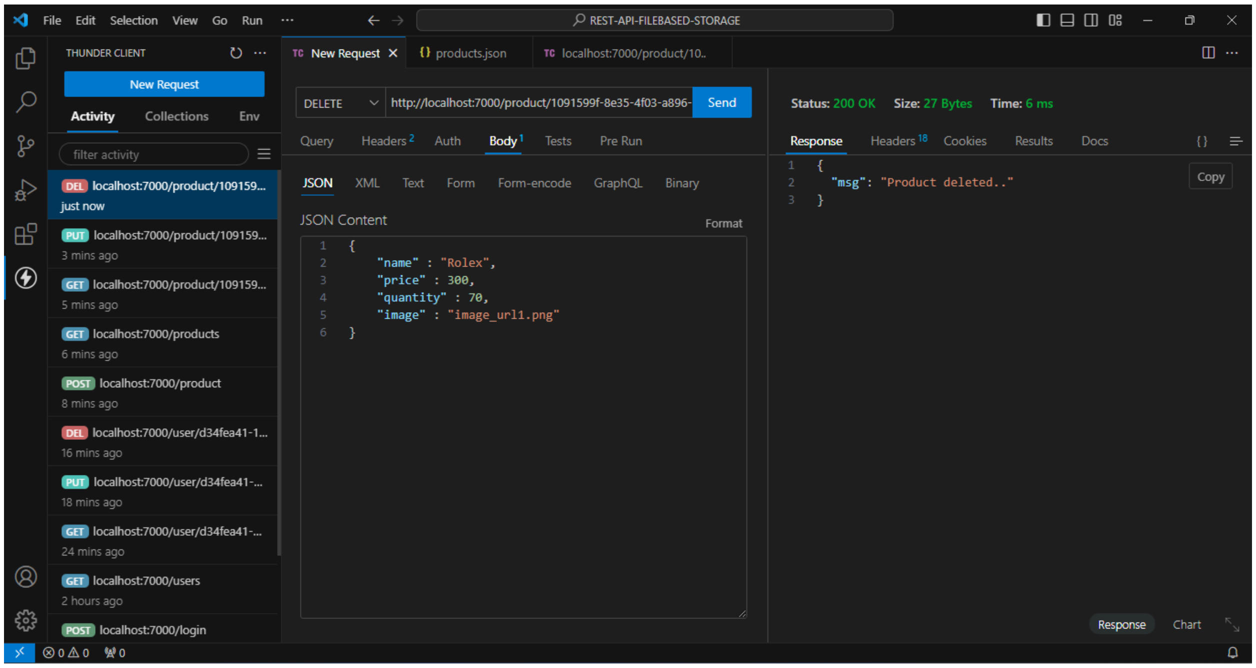
PUT http://localhost:7000/product/1091599f-8e35-4f03-a896-beeab Send

```
{
    "name" : "Rolex",
    "price" : 300,
    "quantity" : 70,
    "image" : "image_url1.png"
}
```

Status: 200 OK Size: 129 Bytes Time: 6 ms

```
{
  "updateProduct": {
    "id": "1091599f-8e35-4f03-a896-beeab3af0b1b",
    "name": "Rolex",
    "price": 300,
    "quantity": 70,
    "image": "image_url1.png"
  }
}
```

Delete a product (By ID)


File Edit Selection View Go Run ···
REST-API-FILEBASED-STORAGE
THUNDER CLIENT
New Request
Activity Collections Env
filter activity
DEL localhost:7000/product/109159... just now
PUT localhost:7000/product/109159... 3 mins ago
GET localhost:7000/product/109159... 5 mins ago
GET localhost:7000/products 6 mins ago
POST localhost:7000/product 8 mins ago
DEL localhost:7000/user/d34fea41-1... 16 mins ago
PUT localhost:7000/user/d34fea41-... 18 mins ago
GET localhost:7000/user/d34fea41-... 24 mins ago
GET localhost:7000/users 2 hours ago
POST localhost:7000/login
New Request
products.json
localhost:7000/product/10..
Query Headers 2 Auth Body 1 Tests Pre Run
JSON XML Text Form Form-encode GraphQL Binary
JSON Content Format
Response Headers 18 Cookies Results Docs
Copy
Response Chart

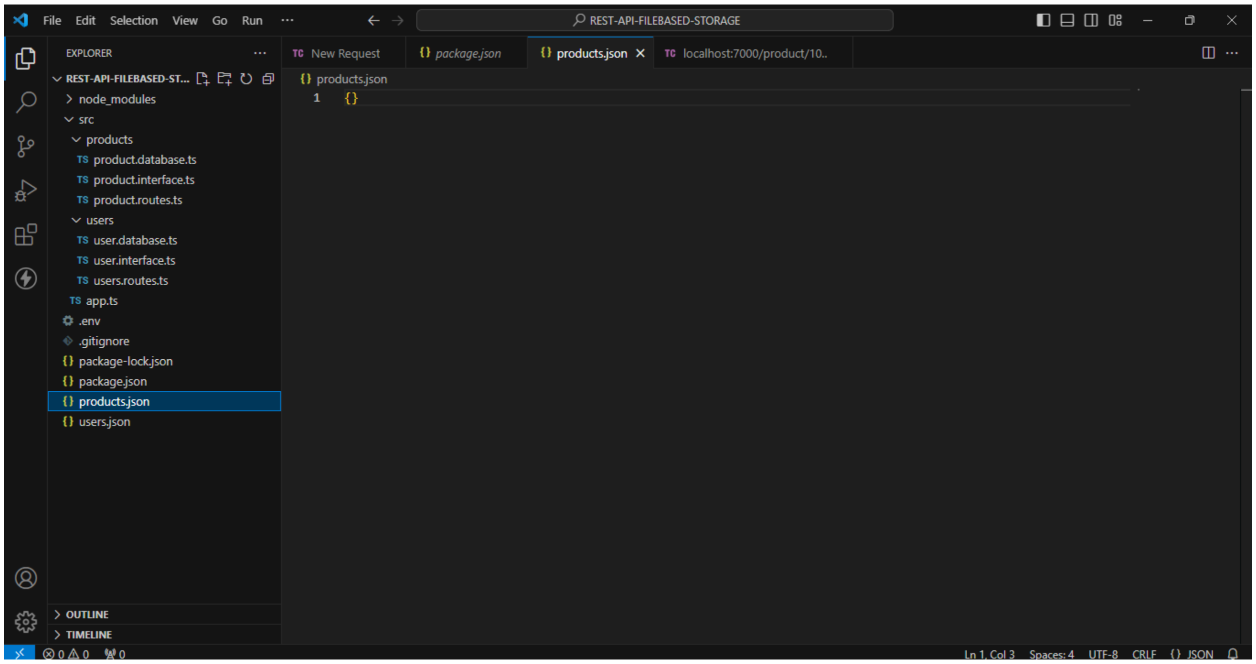
DELETE http://localhost:7000/product/1091599f-8e35-4f03-a896- Send

```
{
    "name" : "Rolex",
    "price" : 300,
    "quantity" : 70,
    "image" : "image_url1.png"
}
```

Status: 200 OK Size: 27 Bytes Time: 6 ms

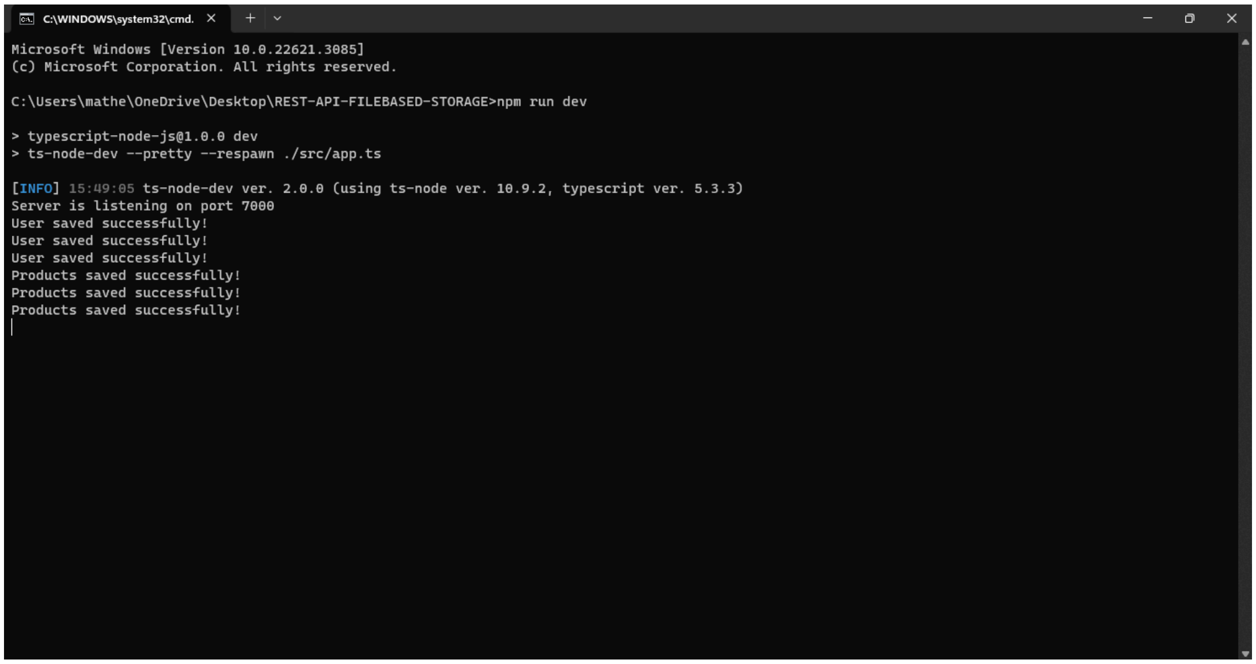
```
{
  "msg": "Product deleted.."
}
```

Delete a product (Changes reflected on the products.json file after request has been done)



```
{}
```

Terminal Logs (Create, Update, Delete changes on products.json)



```
Microsoft Windows [Version 10.0.22621.3085]
(c) Microsoft Corporation. All rights reserved.

C:\Users\mathe\OneDrive\Desktop\REST-API-FILEBASED-STORAGE>npm run dev

> typescript-node-js@1.0.0 dev
> ts-node-dev --pretty --respawn ./src/app.ts

[INFO] 15:49:05 ts-node-dev ver. 2.0.0 (using ts-node ver. 10.9.2, typescript ver. 5.3.3)
Server is listening on port 7000
User saved successfully!
User saved successfully!
User saved successfully!
Products saved successfully!
Products saved successfully!
Products saved successfully!
```