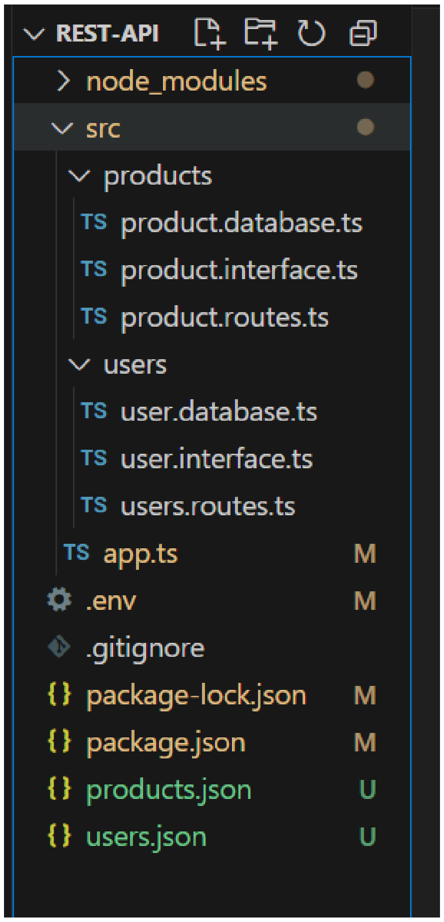
Title: Build a REST API with TS, Node, Express & FIle based Storage System
Author: Rey Mark Rivas
Date: 02/28/25

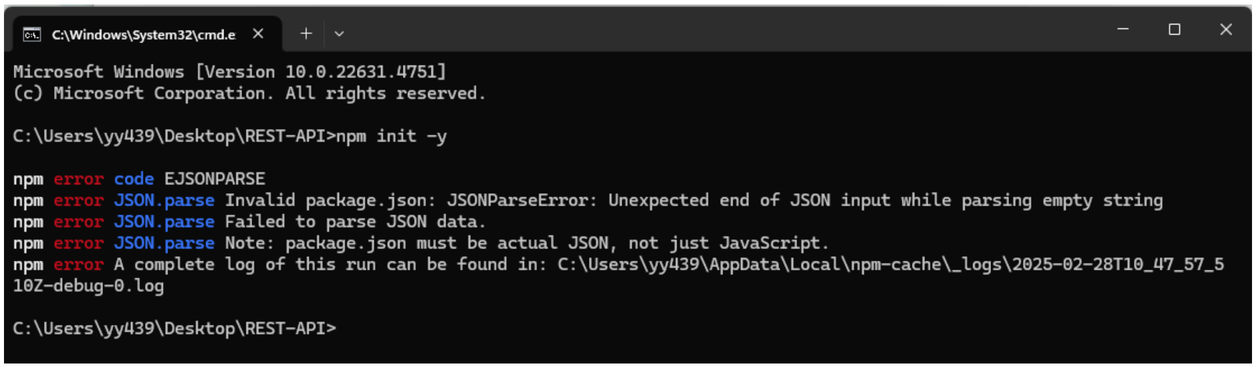
Introduction:
This document outlines the steps taken during Build a REST API with TS, Node, Express & FIle based Storage, including screenshots and descriptions of key processes.

creating a project directory:



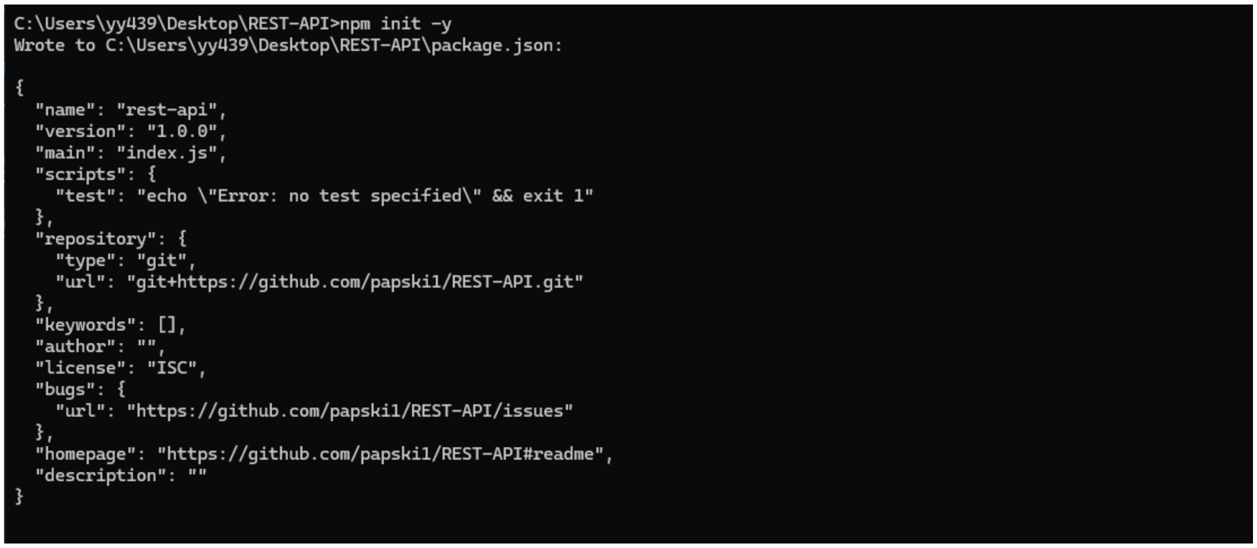
```
REST-API
  node_modules
  src
    products
      product.database.ts
      product.interface.ts
      product.routes.ts
    users
      user.database.ts
      user.interface.ts
      users.routes.ts
    app.ts              M
  .env                  M
  .gitignore
  package-lock.json     M
  package.json          M
  products.json         U
  users.json            U
```

Problem initializing:



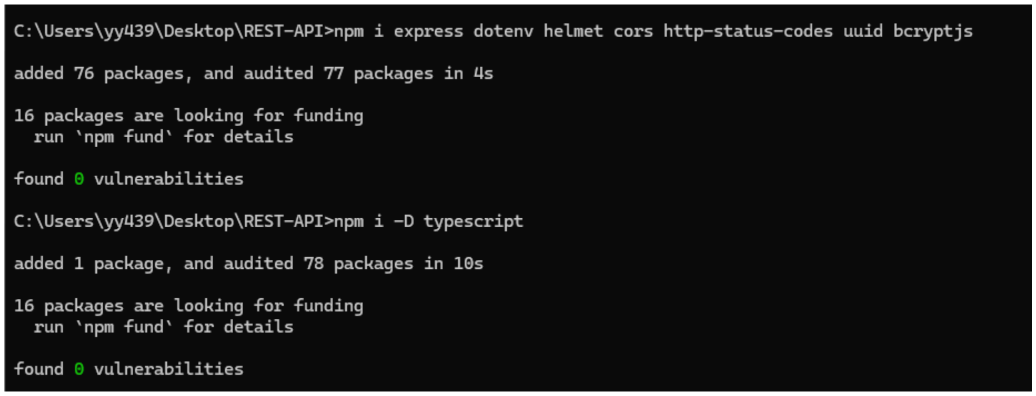
```
Microsoft Windows [Version 10.0.22631.4751]
(c) Microsoft Corporation. All rights reserved.

C:\Users\yy439\Desktop\REST-API>npm init -y

npm error code EJSONPARSE
npm error JSON.parse Invalid package.json: JSONParseError: Unexpected end of JSON input while parsing empty string
npm error JSON.parse Failed to parse JSON data.
npm error JSON.parse Note: package.json must be actual JSON, not just JavaScript.
npm error A complete log of this run can be found in: C:\Users\yy439\AppData\Local\npm-cache\_logs\2025-02-28T10_47_57_5
10Z-debug-0.log

C:\Users\yy439\Desktop\REST-API>
```

Solved:



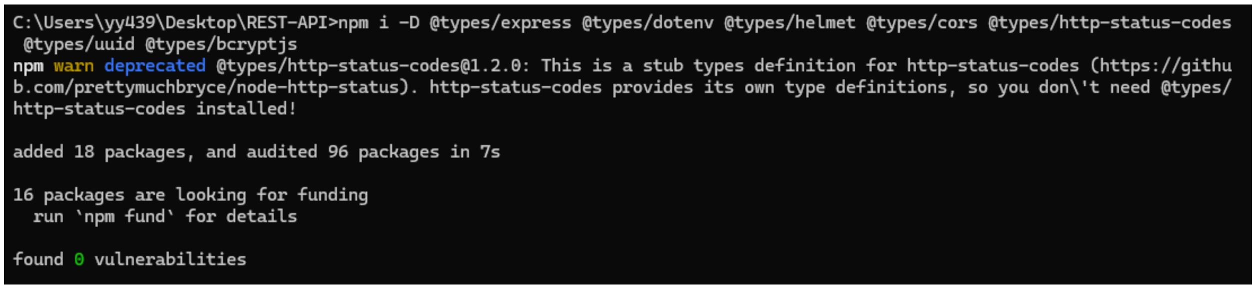
```
C:\Users\yy439\Desktop\REST-API>npm init -y
Wrote to C:\Users\yy439\Desktop\REST-API\package.json:

{
  "name": "rest-api",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "repository": {
    "type": "git",
    "url": "git+https://github.com/papski1/REST-API.git"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "bugs": {
    "url": "https://github.com/papski1/REST-API/issues"
  },
  "homepage": "https://github.com/papski1/REST-API#readme",
  "description": ""
}
```

Express server with TypeScript:



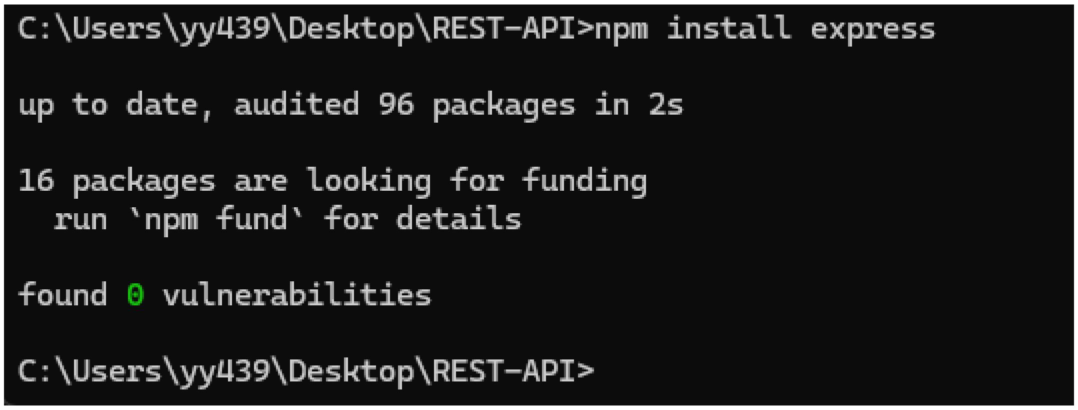
```
C:\Users\yy439\Desktop\REST-API>npm i express dotenv helmet cors http-status-codes uuid bcryptjs

added 76 packages, and audited 77 packages in 4s

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

found 0 vulnerabilities

C:\Users\yy439\Desktop\REST-API>npm i -D typescript

added 1 package, and audited 78 packages in 10s

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

found 0 vulnerabilities
```

Installing type definitions:



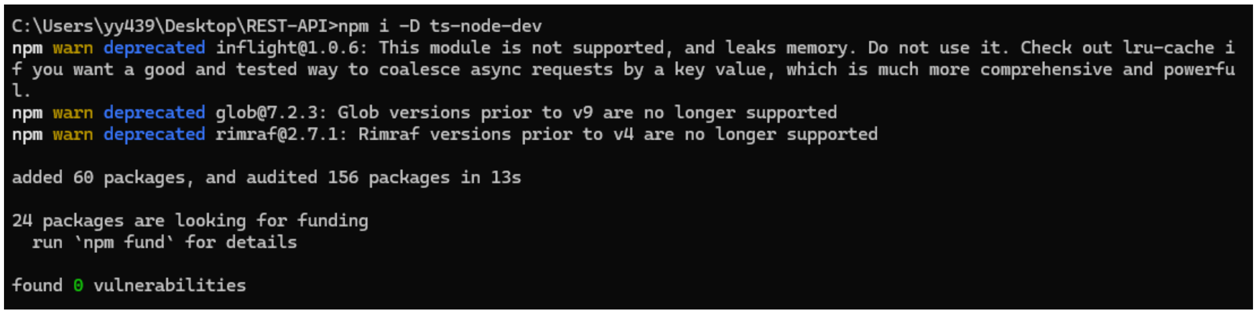
```
C:\Users\yy439\Desktop\REST-API>npm i -D @types/express @types/dotenv @types/helmet @types/cors @types/http-status-codes
 @types/uuid @types/bcryptjs
npm warn deprecated @types/http-status-codes@1.2.0: This is a stub types definition for http-status-codes (https://githu
b.com/prettymuchbryce/node-http-status). http-status-codes provides its own type definitions, so you don\'t need @types/
http-status-codes installed!

added 18 packages, and audited 96 packages in 7s

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

found 0 vulnerabilities
```

Installing express.js:



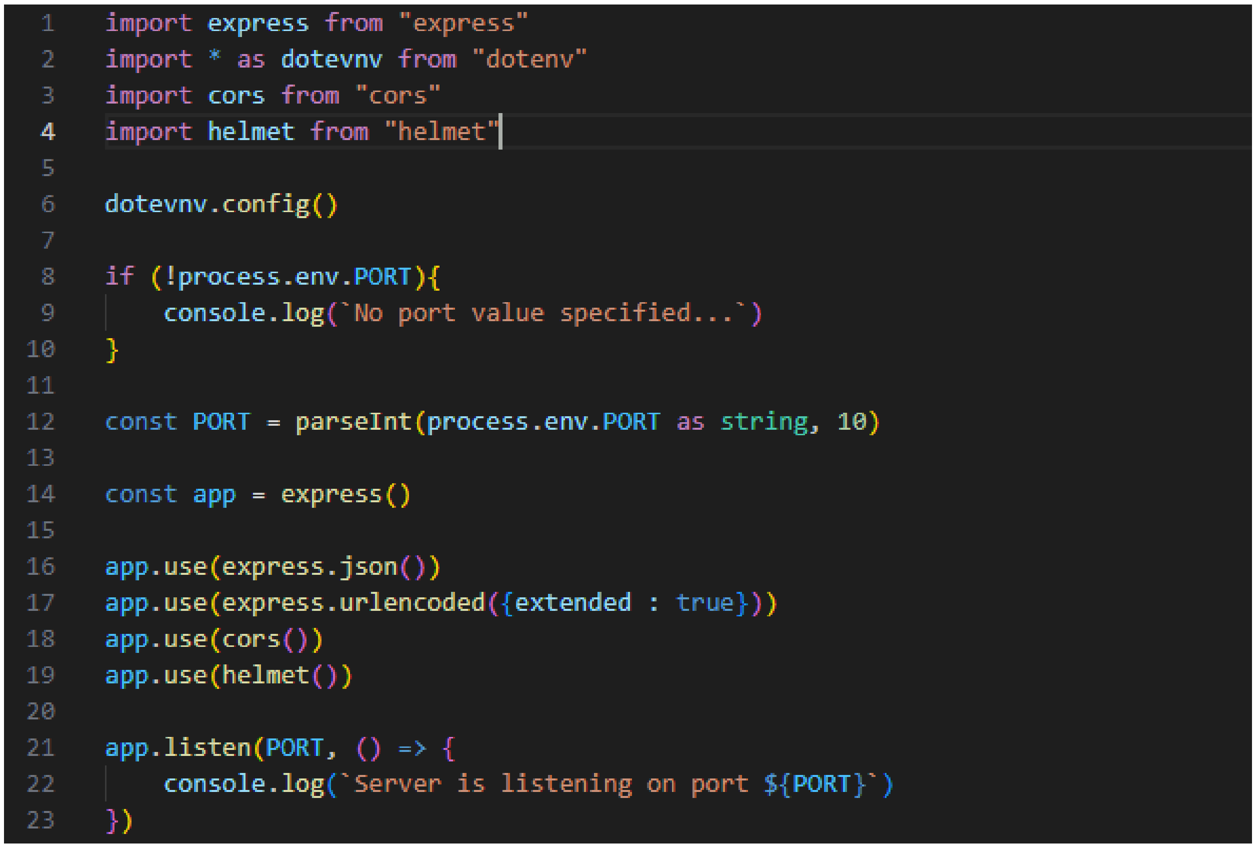
```
C:\Users\yy439\Desktop\REST-API>npm install express

up to date, audited 96 packages in 2s

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

found 0 vulnerabilities

C:\Users\yy439\Desktop\REST-API>
```

installing this package to power up your development workflow:



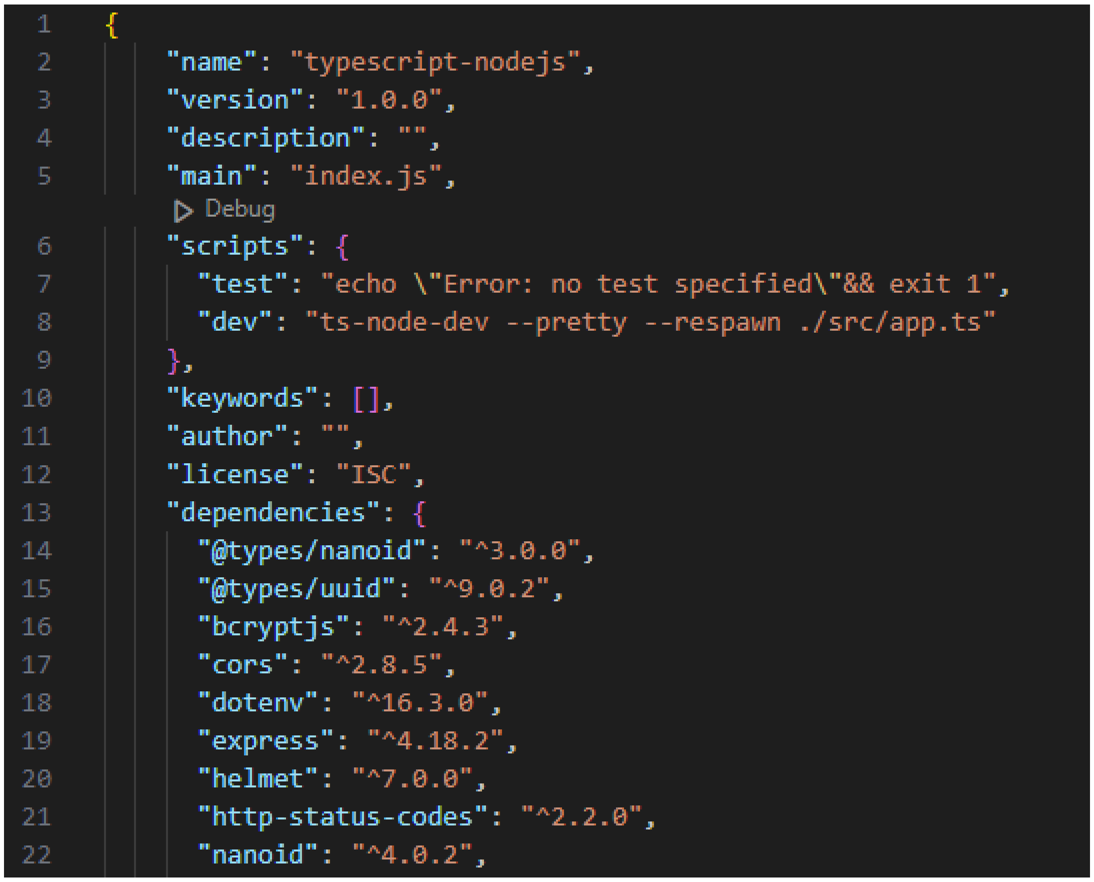
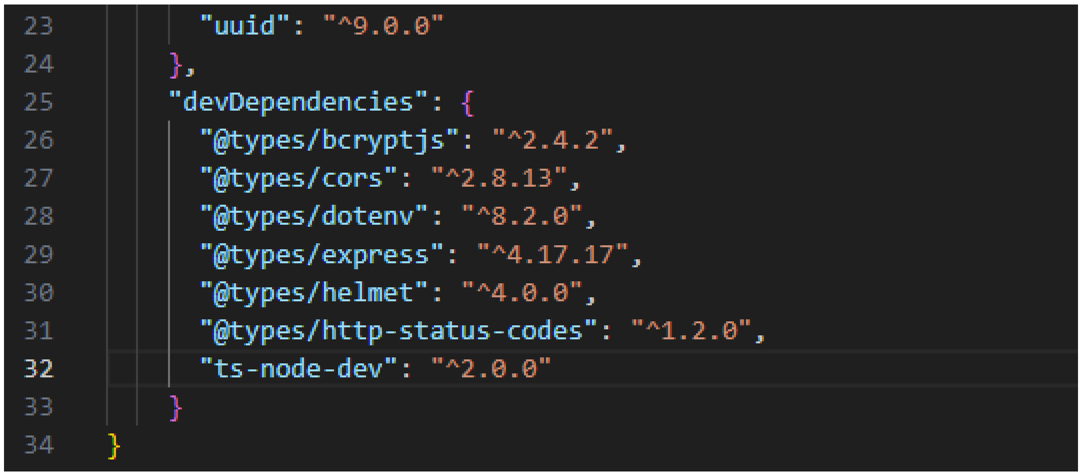
```
C:\Users\yy439\Desktop\REST-API>npm i -D ts-node-dev
npm warn deprecated inflight@1.0.6: This module is not supported, and leaks memory. Do not use it. Check out lru-cache i
f you want a good and tested way to coalesce async requests by a key value, which is much more comprehensive and powerfu
l.
npm warn deprecated glob@7.2.3: Glob versions prior to v9 are no longer supported
npm warn deprecated rimraf@2.7.1: Rimraf versions prior to v4 are no longer supported

added 60 packages, and audited 156 packages in 13s

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

found 0 vulnerabilities
```

Import project dependencies:

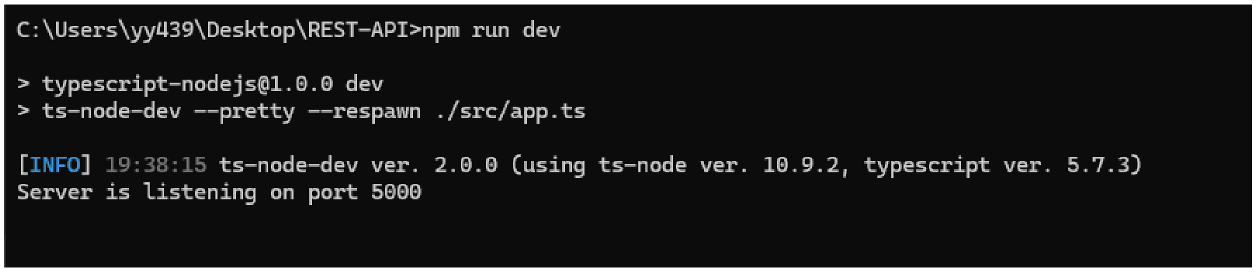


```
import express from "express"
import * as dotevnv from "dotenv"
import cors from "cors"
import helmet from "helmet"

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

Updated package.json

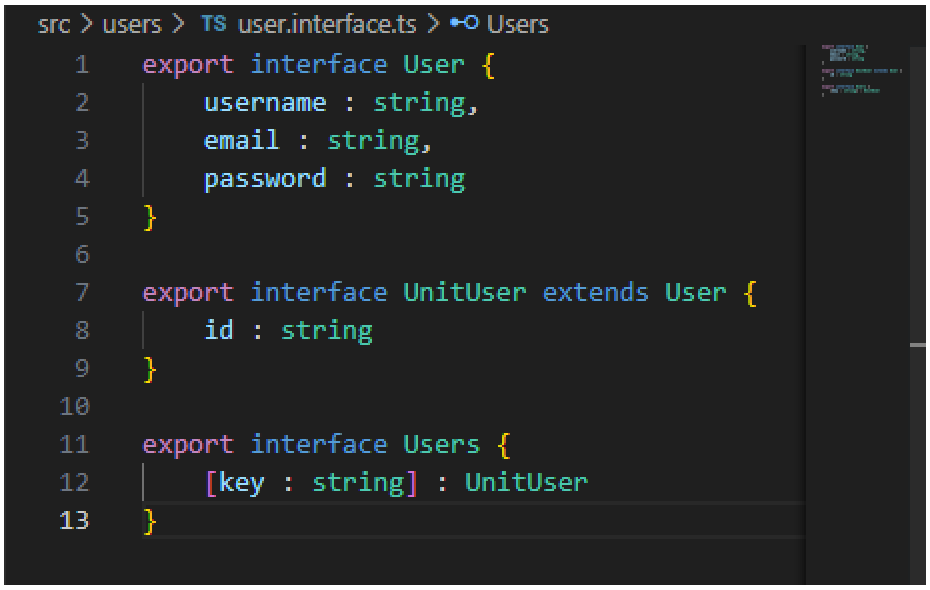



```
{
    "name": "typescript-nodejs",
    "version": "1.0.0",
    "description": "",
    "main": "index.js",
    ▷ Debug
    "scripts": {
      "test": "echo \"Error: no test specified\"&& exit 1",
      "dev": "ts-node-dev --pretty --respawn ./src/app.ts"
    },
    "keywords": [],
    "author": "",
    "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"
    }
}
```



```
C:\Users\yy439\Desktop\REST-API>npm run dev

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

[INFO] 19:38:15 ts-node-dev ver. 2.0.0 (using ts-node ver. 10.9.2, typescript ver. 5.7.3)
Server is listening on port 5000
```

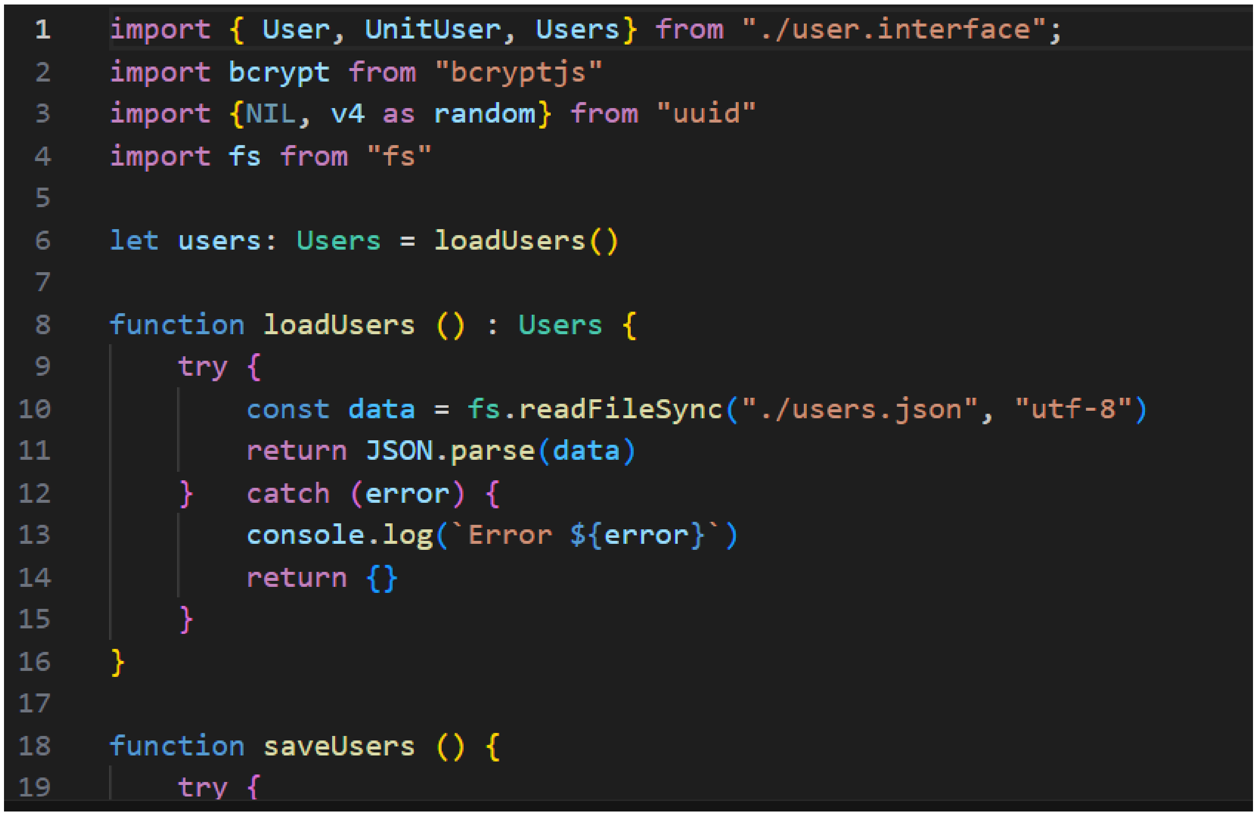
Populate src/users/user.interface.ts:



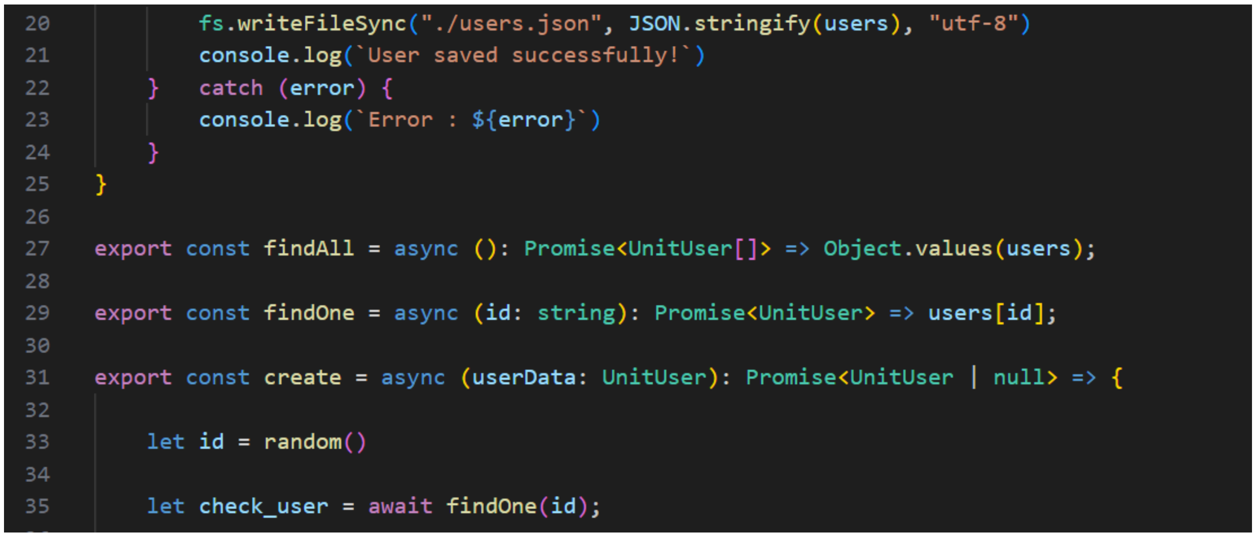
src > users > TS user.interface.ts > Users

```typescript
export interface User {
    username : string,
    email : string,
    password : string
}

export interface UnitUser extends User {
    id : string
}

export interface Users {
    [key : string] : UnitUser
}
```

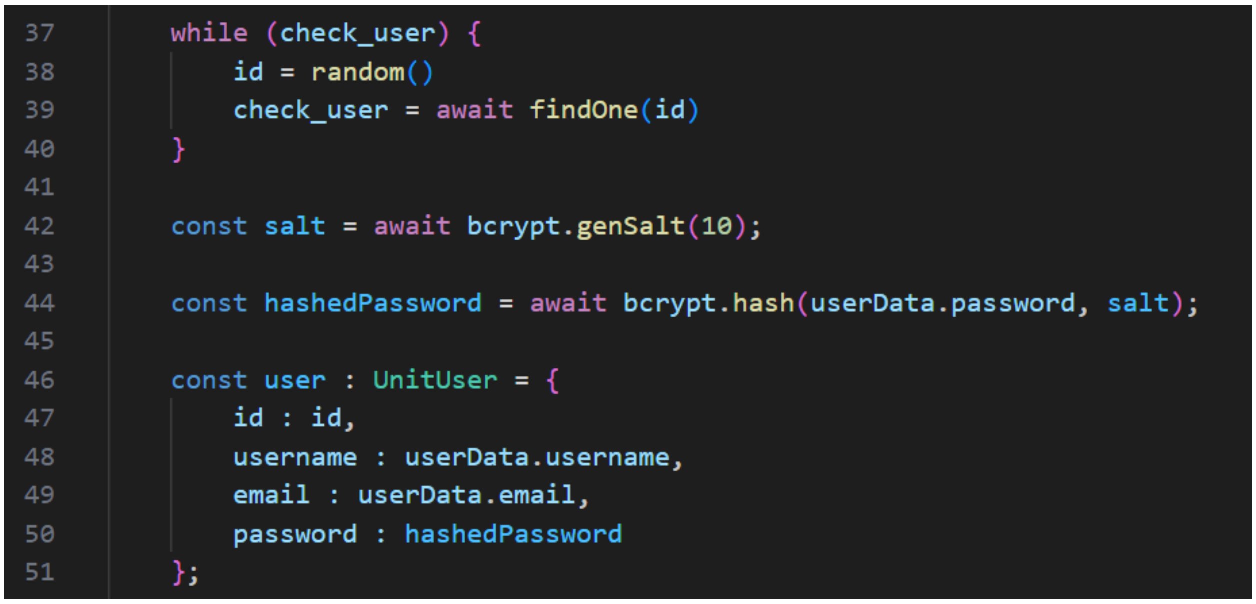
Populate src/users/user.database.ts:



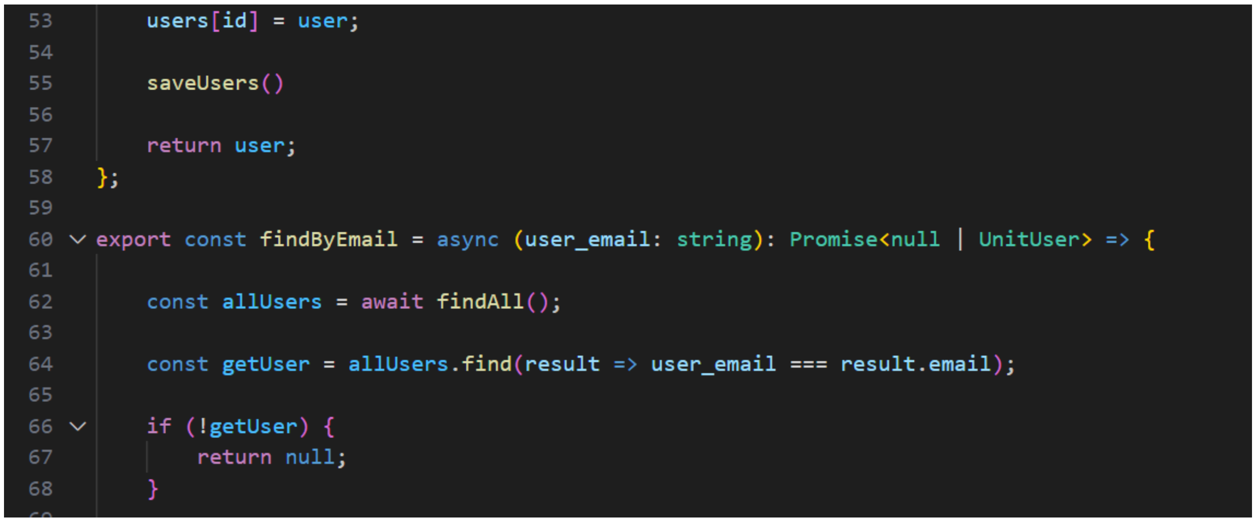
```typescript
import { User, UnitUser, Users} from "./user.interface";
import bcrypt from "bcryptjs"
import {NIL, v4 as random} from "uuid"
import fs from "fs"

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 {
```



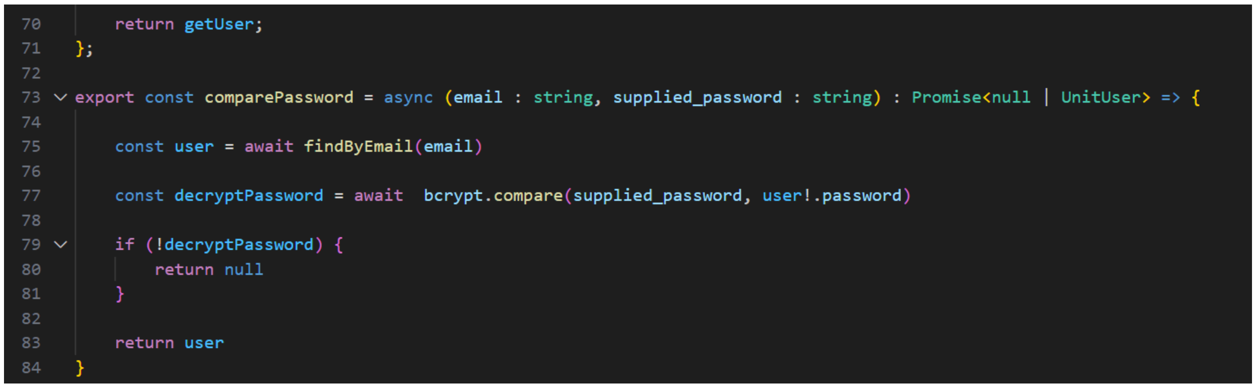
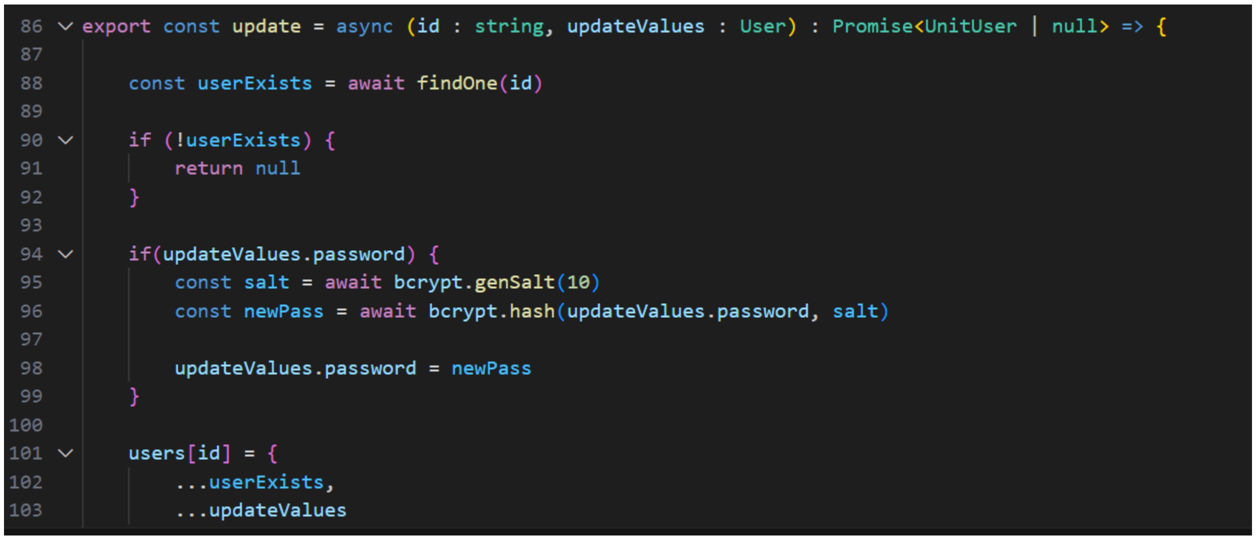
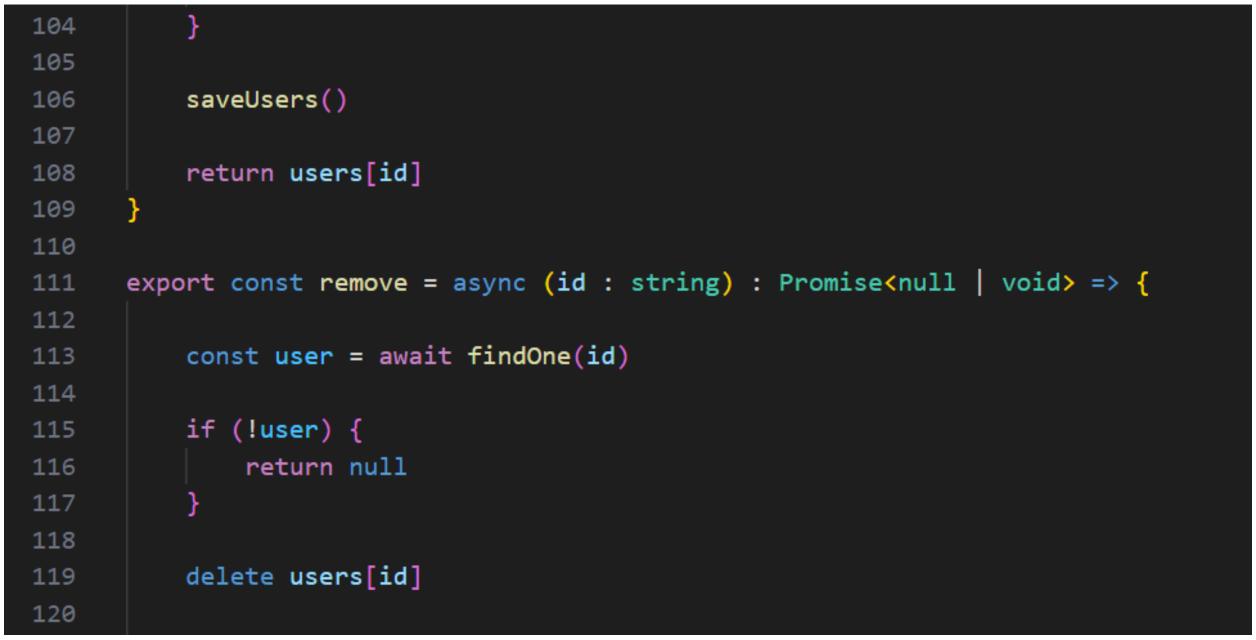
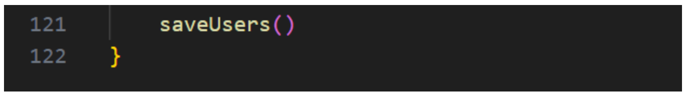
```
        fs.writeFileSync("./users.json", JSON.stringify(users), "utf-8")
        console.log(`User saved successfully!`)
    } catch (error) {
        console.log(`Error : ${error}`)
    }
}

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> => {

    let id = random()

    let check_user = await findOne(id);
```



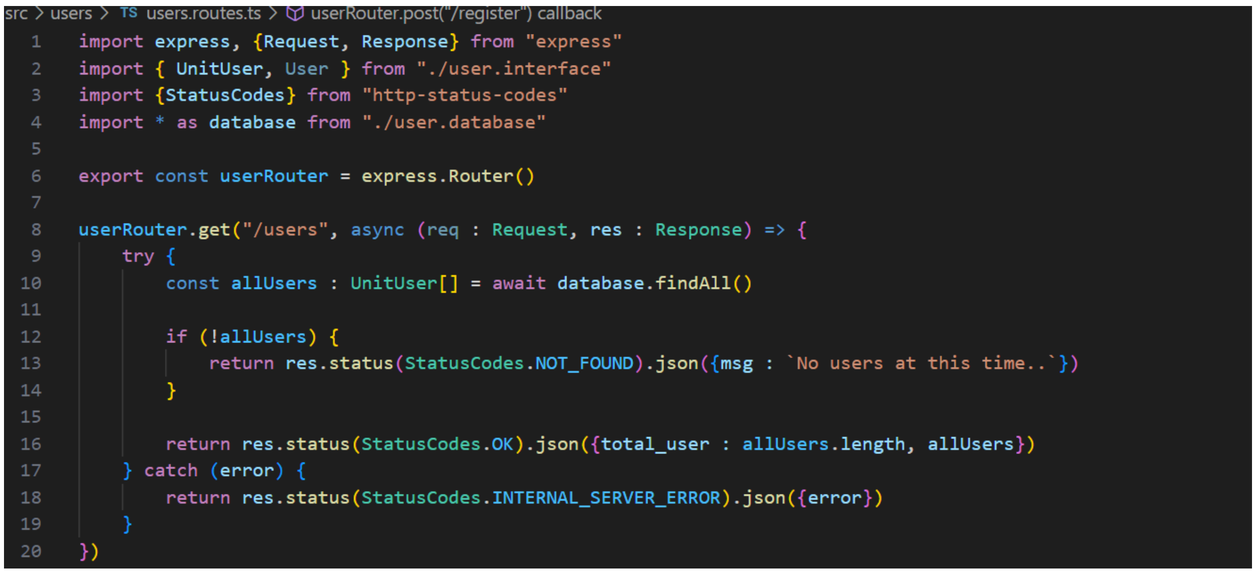
```
    while (check_user) {
        id = random()
        check_user = await findOne(id)
    }

    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
    };
```



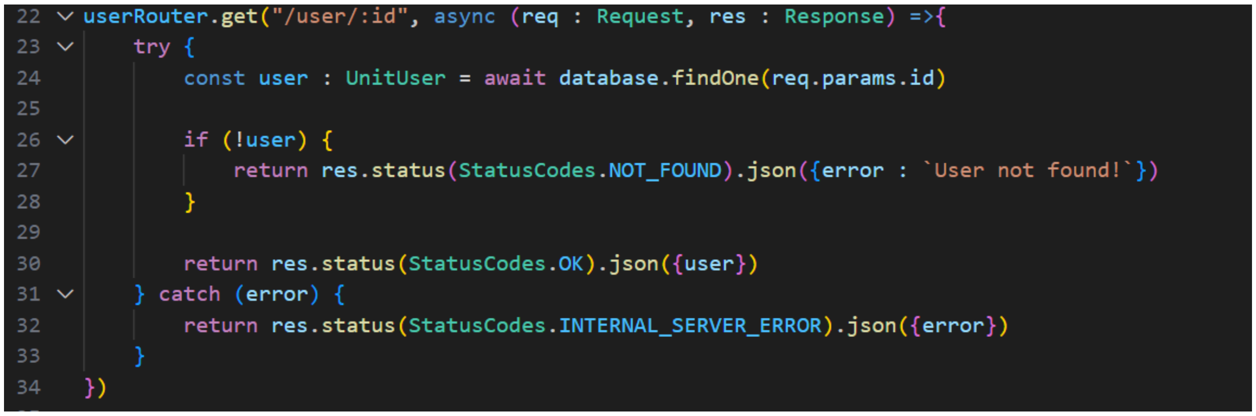
```
    users[id] = user;

    saveUsers()

    return user;
};

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

    const allUsers = await findAll();

    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()
}
```

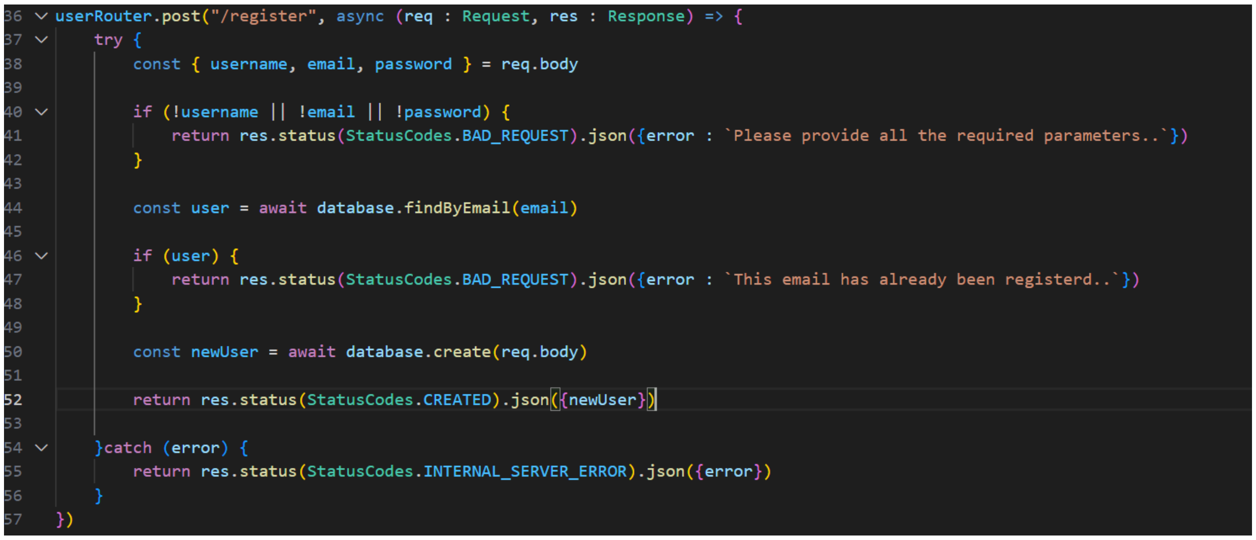
Import all the required functions and modules into the routes file ./src/users.routes.ts:


src > users > TS users.routes.ts > userRouter.post("/register") callback

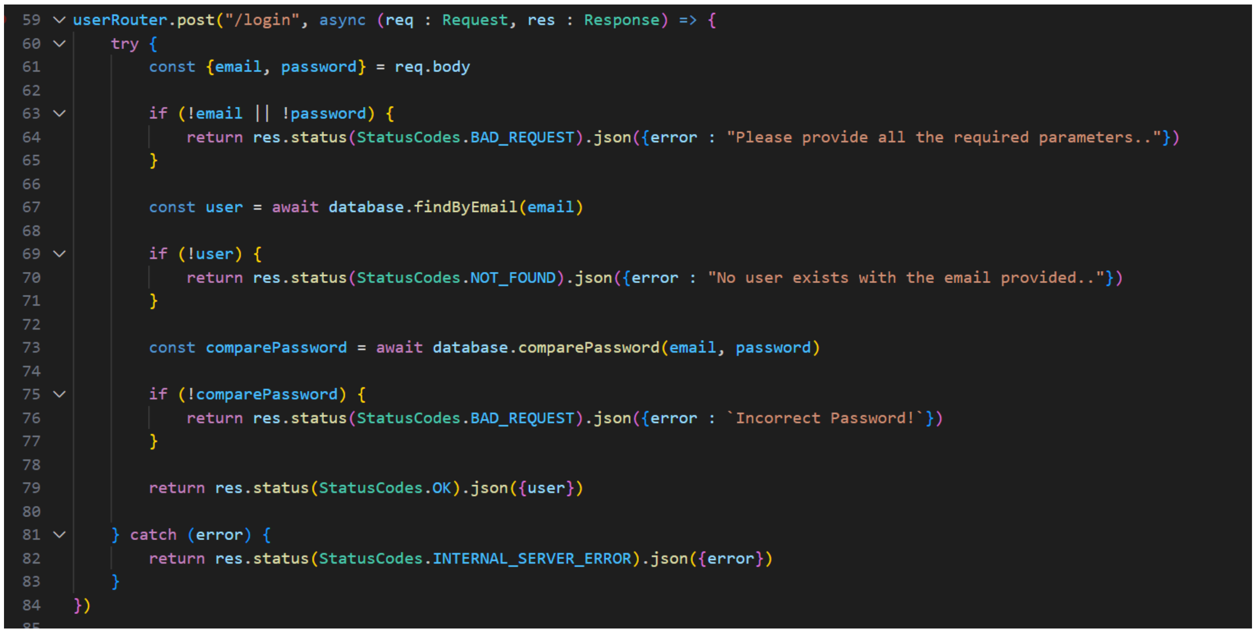
```
import express, {Request, Response} from "express"
import { UnitUser, User } from "./user.interface"
import {StatusCodes} from "http-status-codes"
import * as database from "./user.database"

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})
    }
})
```



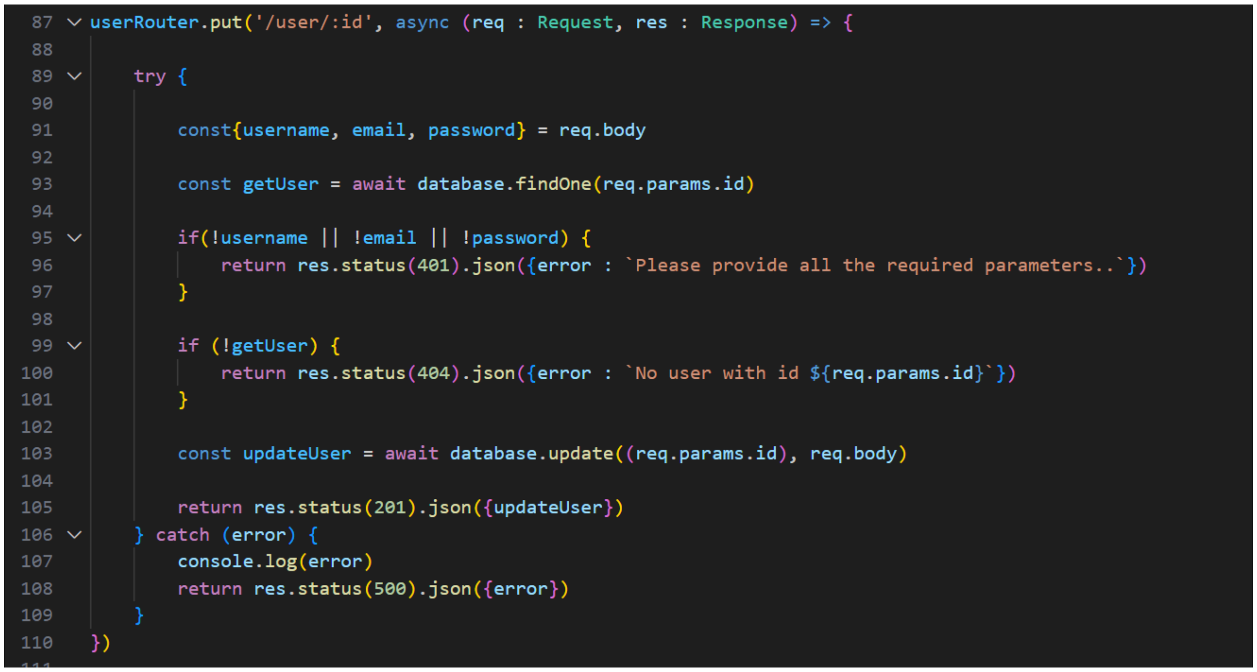
```
userRouter.get("/user/:id", async (req : Request, res : Response) =>{
    try {
        const user : UnitUser = await database.findOne(req.params.id)

        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})
    }
})
```



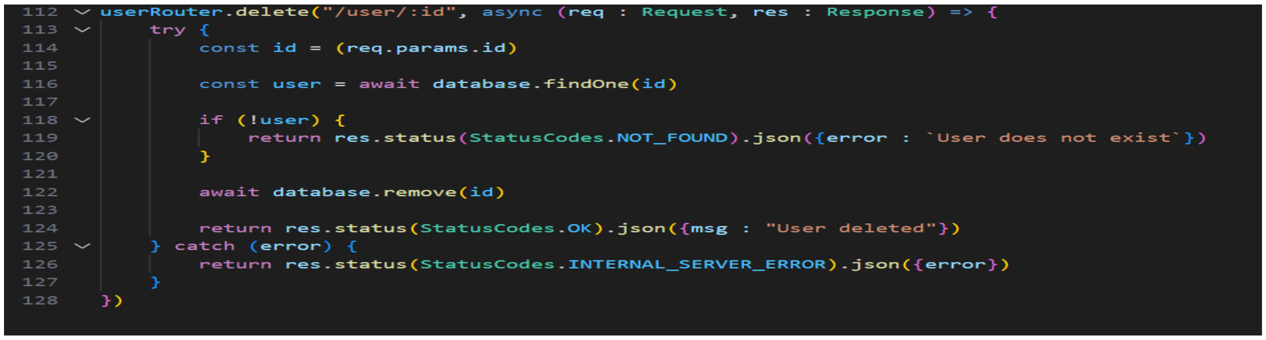
```
userRouter.post("/register", async (req : Request, res : Response) => {
    try {
        const { username, email, password } = req.body

        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 registerd..`})
        }

        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) {
            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})
    }
})
```

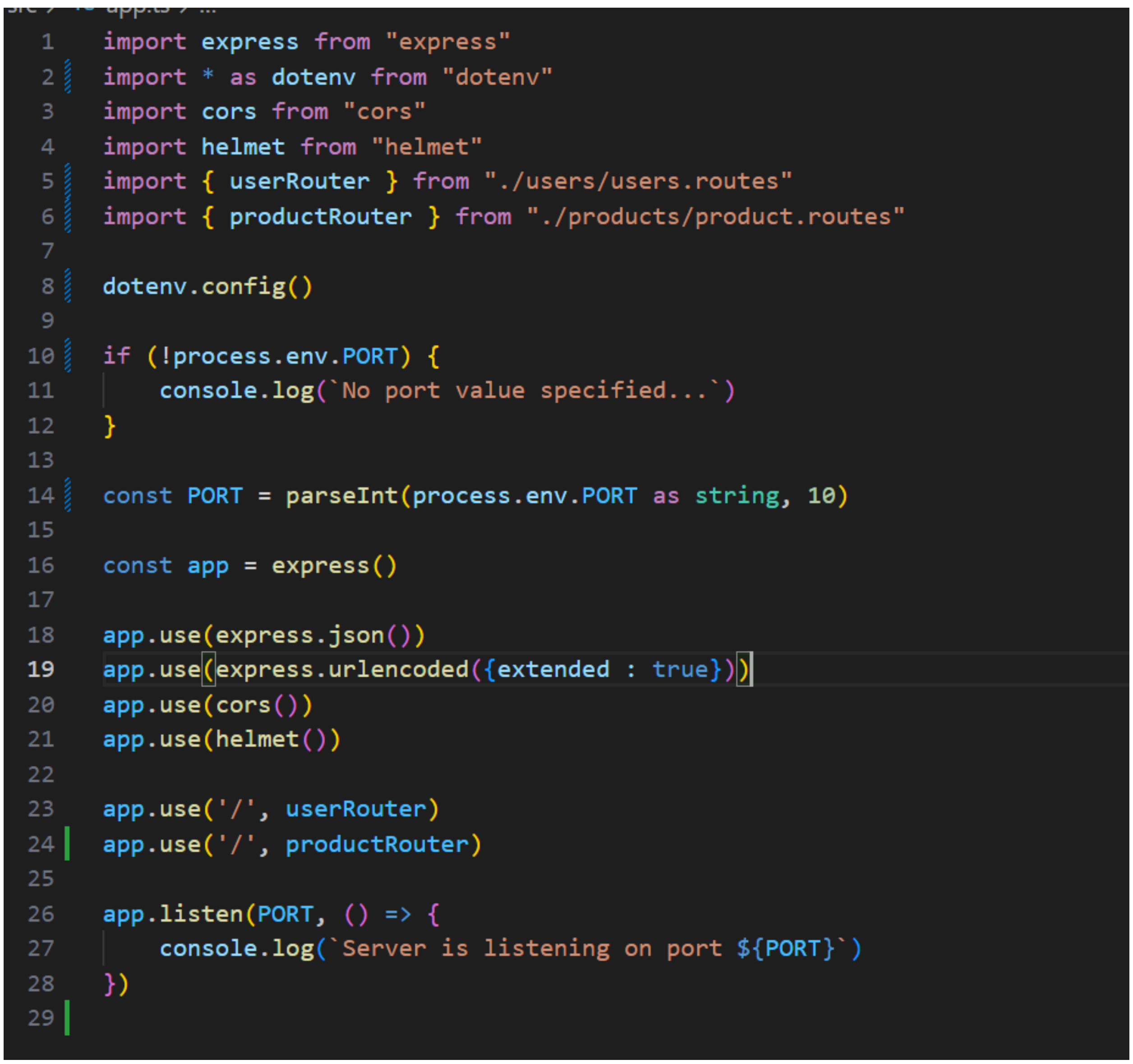


```
userRouter.put('/user/:id', async (req : Request, res : Response) => {

    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..`})
        }

        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})
    }
})
```



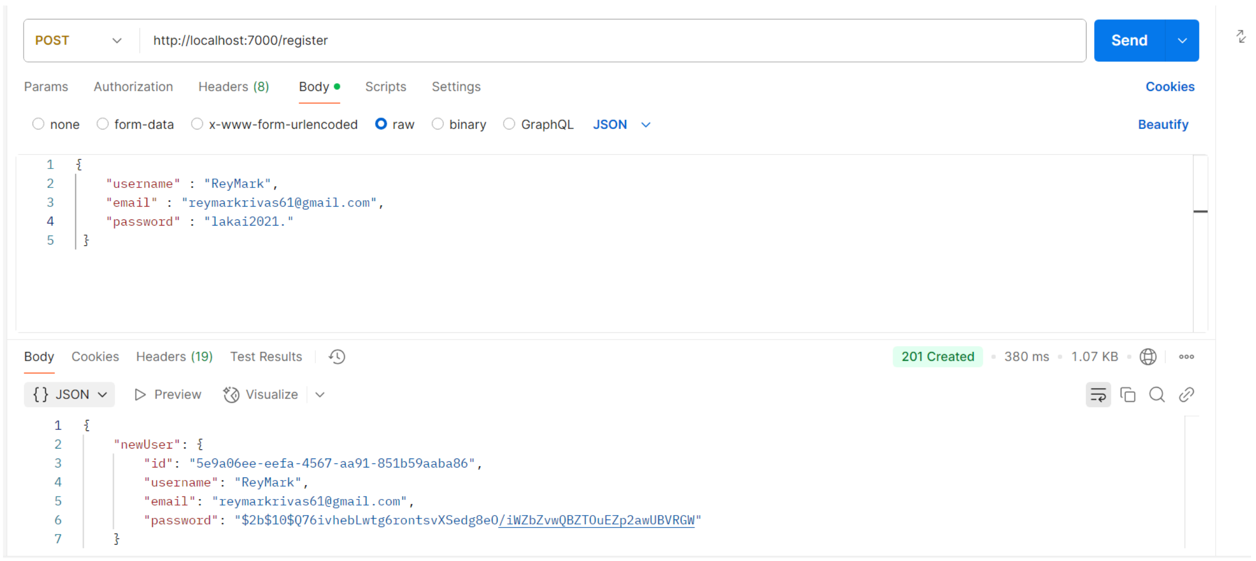
```
userRouter.delete("/user/:id", async (req : Request, res : Response) => {
    try {
        const id = (req.params.id)

        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})
    }
})
```

Updated app.ts:



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

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

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

Register users:



POST http://localhost:7000/register

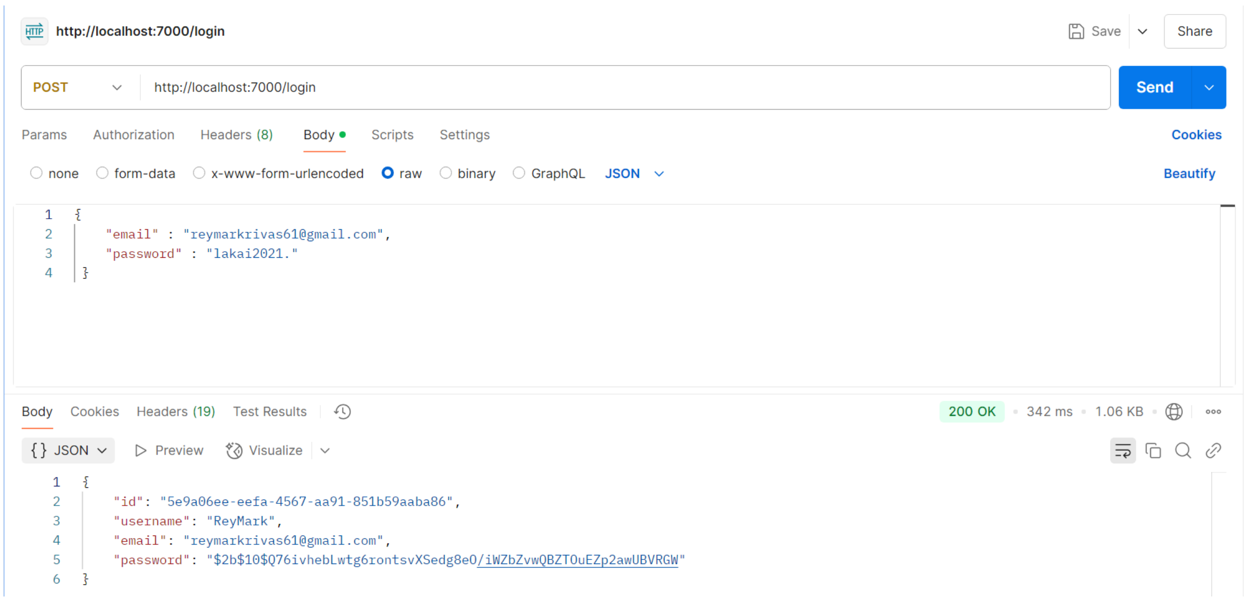
Body: [✓ raw] JSON

```
{
    "username" : "ReyMark",
    "email" : "reymarkrivas61@gmail.com",
    "password" : "lakai2021."
}
```

201 Created · 380 ms · 1.07 KB

```
{
    "newUser": {
        "id": "5e9a06ee-eefa-4567-aa91-851b59aaba86",
        "username": "ReyMark",
        "email": "reymarkrivas61@gmail.com",
        "password": "$2b$10$Q76ivhebLwtg6rontsvXSedg8e0/iWZbZvwQBZTOuEZp2awUBVRGW"
    }
```

Login users:



http://localhost:7000/login

POST http://localhost:7000/login

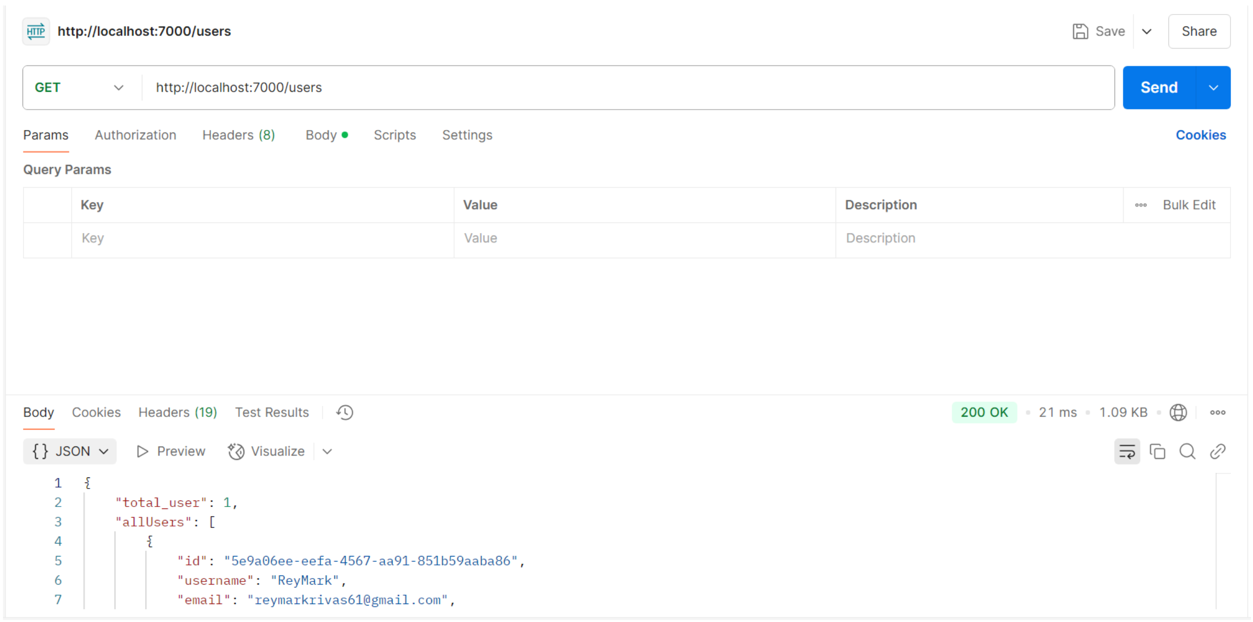
Body: [✓ raw] JSON

```
{
    "email" : "reymarkrivas61@gmail.com",
    "password" : "lakai2021."
}
```

200 OK · 342 ms · 1.06 KB

```
{
    "id": "5e9a06ee-eefa-4567-aa91-851b59aaba86",
    "username": "ReyMark",
    "email": "reymarkrivas61@gmail.com",
    "password": "$2b$10$Q76ivhebLwtg6rontsvXSedg8e0/iWZbZvwQBZTOuEZp2awUBVRGW"
}
```

Get all users:

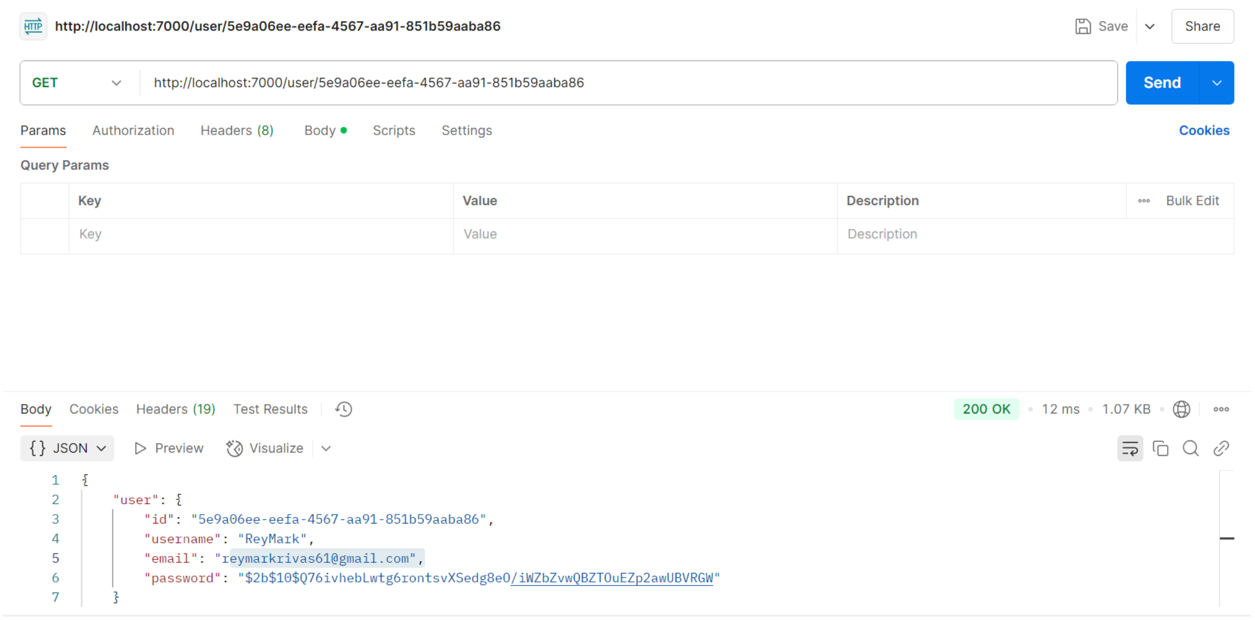


GET http://localhost:7000/users

Key	Value	Description
Key	Value	Description

```
{
    "total_user": 1,
    "allUsers": [
        {
            "id": "5e9a06ee-eefa-4567-aa91-851b59aaba86",
            "username": "ReyMark",
            "email": "reymarkrivas61@gmail.com",
```

Get a single user:

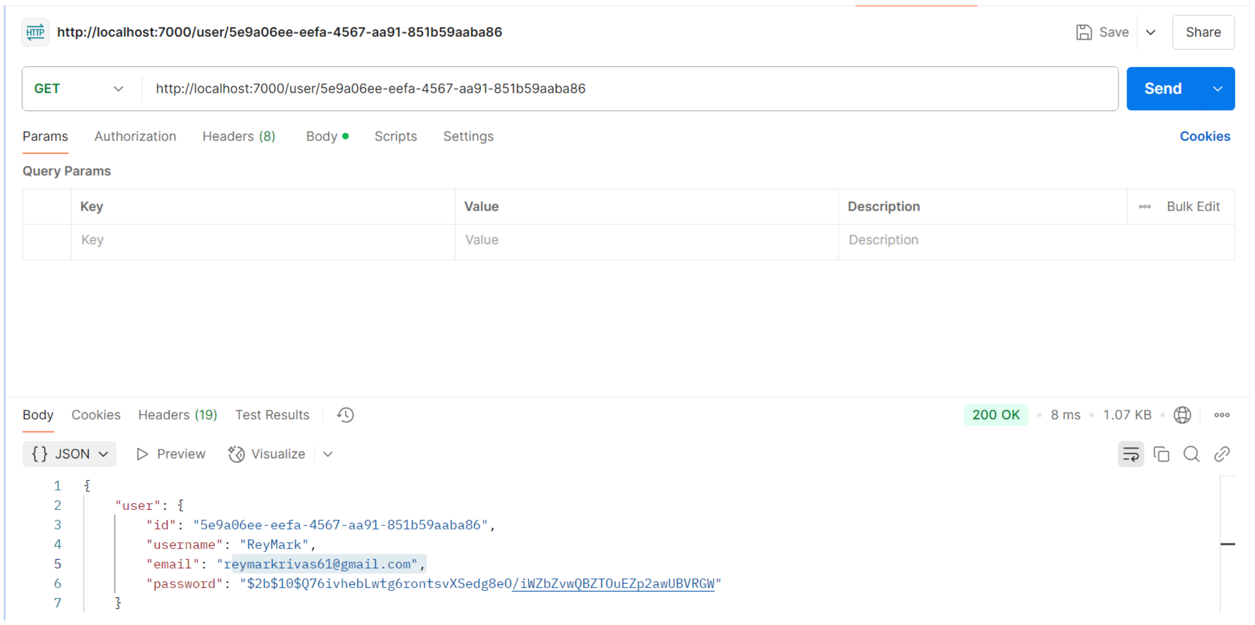


GET http://localhost:7000/user/5e9a06ee-eefa-4567-aa91-851b59aaba86

Key	Value	Description
Key	Value	Description

```
{
    "user": {
        "id": "5e9a06ee-eefa-4567-aa91-851b59aaba86",
        "username": "ReyMark",
        "email": "reymarkrivas61@gmail.com",
        "password": "$2b$10$Q76ivhebLwtg6rontsvXSedg8e0/iWZbZvwQBZTOuEZp2awUBVRGW"
    }
```

Update user:



http://localhost:7000/user/5e9a06ee-eefa-4567-aa91-851b59aaba86

GET http://localhost:7000/user/5e9a06ee-eefa-4567-aa91-851b59aaba86

Params | Authorization | Headers (8) | Body | Scripts | Settings

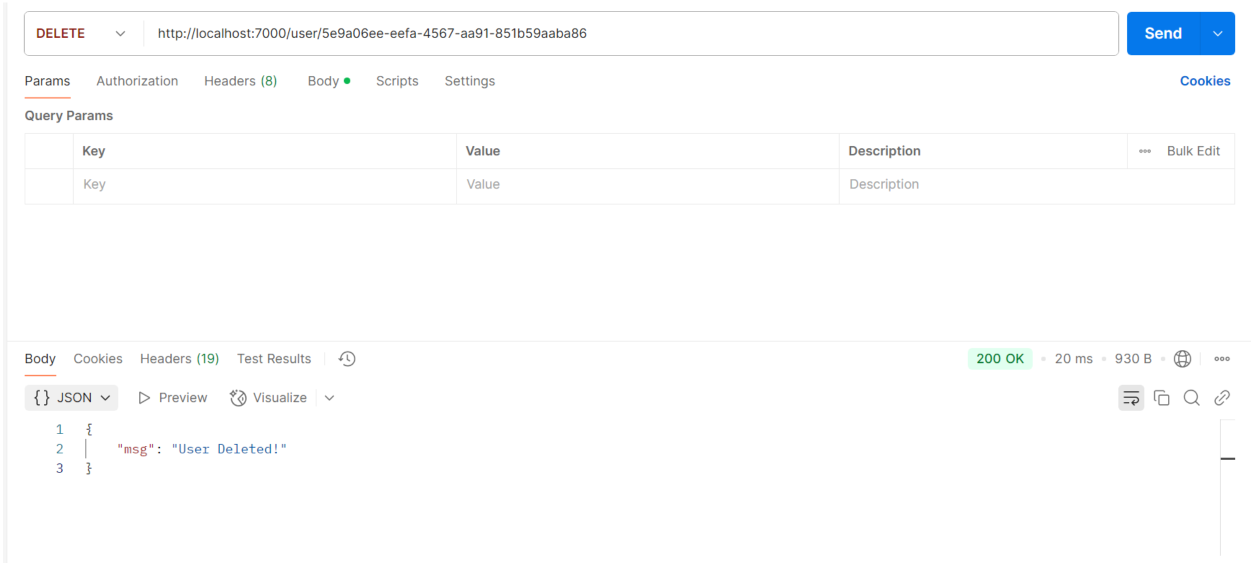
Query Params

Key	Value	Description
Key	Value	Description

Body | Cookies | Headers (19) | Test Results — 200 OK · 8 ms · 1.07 KB

```
{
    "user": {
        "id": "5e9a06ee-eefa-4567-aa91-851b59aaba86",
        "username": "ReyMark",
        "email": "reymarkrivas61@gmail.com",
        "password": "$2b$10$Q76ivhebLwtg6rontsvXSedg8e0/iWZbZvwQBZTOuEZp2awUBVRGW"
    }
```

Delete user:



DELETE http://localhost:7000/user/5e9a06ee-eefa-4567-aa91-851b59aaba86

Params | Authorization | Headers (8) | Body | Scripts | Settings

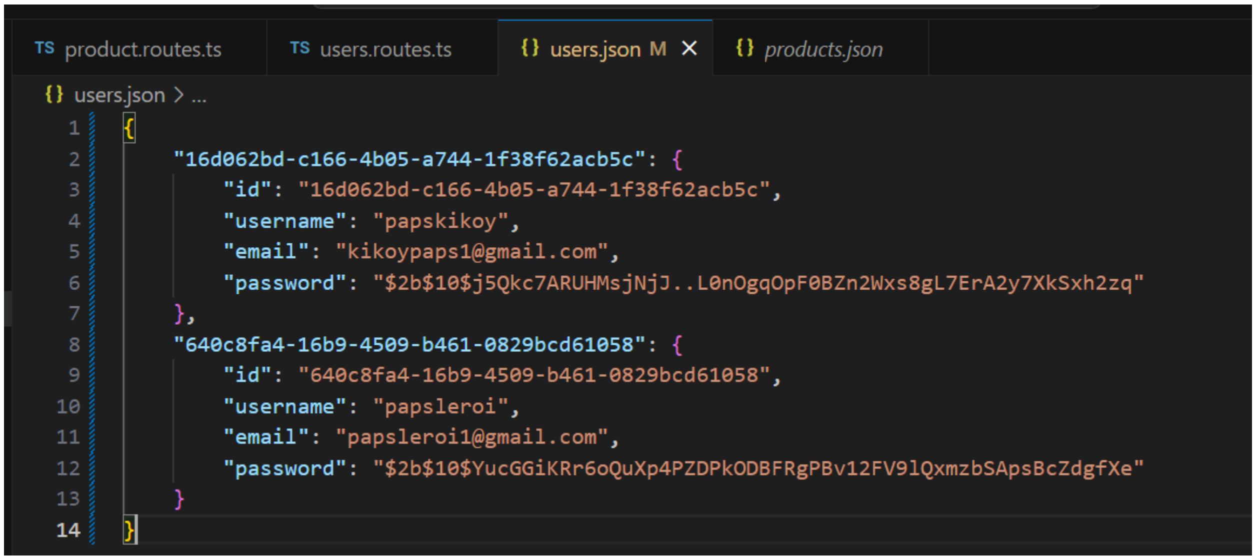
Query Params

Key	Value	Description
Key	Value	Description

Body | Cookies | Headers (19) | Test Results — 200 OK · 20 ms · 930 B

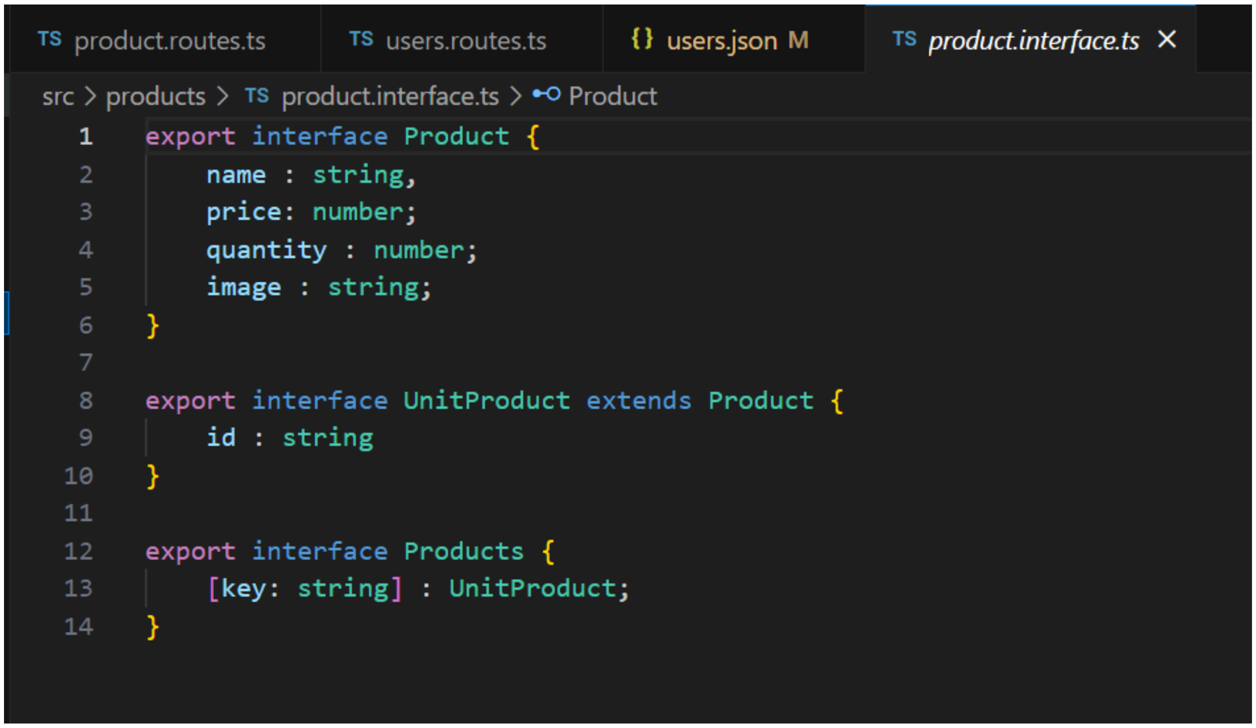
```
{
    "msg": "User Deleted!"
}
```

Users-data-storage-file :



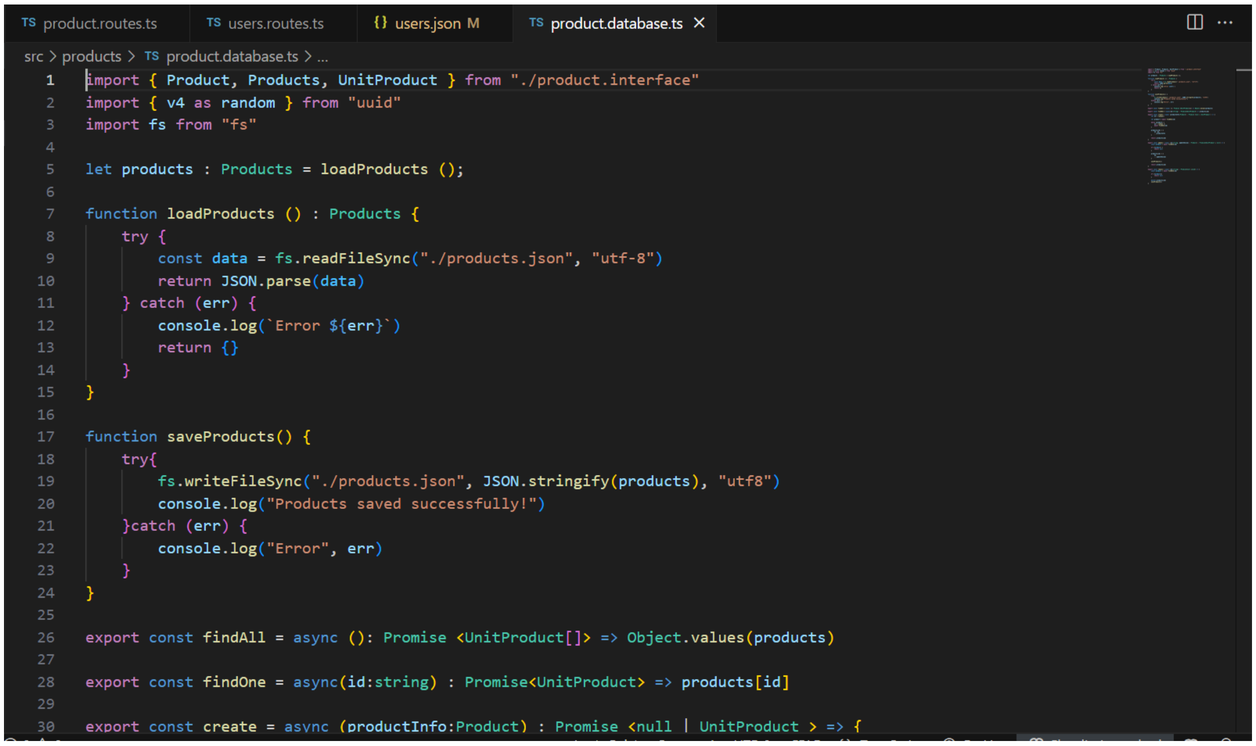
```json
{
    "16d062bd-c166-4b05-a744-1f38f62acb5c": {
        "id": "16d062bd-c166-4b05-a744-1f38f62acb5c",
        "username": "papskikoy",
        "email": "kikoypaps1@gmail.com",
        "password": "$2b$10$j5Qkc7ARUHMsjNjJ..L0nOgqOpF0BZn2Wxs8gL7ErA2y7XkSxh2zq"
    },
    "640c8fa4-16b9-4509-b461-0829bcd61058": {
        "id": "640c8fa4-16b9-4509-b461-0829bcd61058",
        "username": "papsleroi",
        "email": "papsleroi1@gmail.com",
        "password": "$2b$10$YucGGiKRr6oQuXp4PZDPkODBFRgPBv12FV91QxmzbSApsBcZdgfXe"
    }
}
```

create the login and routes for our products:



```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;
}
```

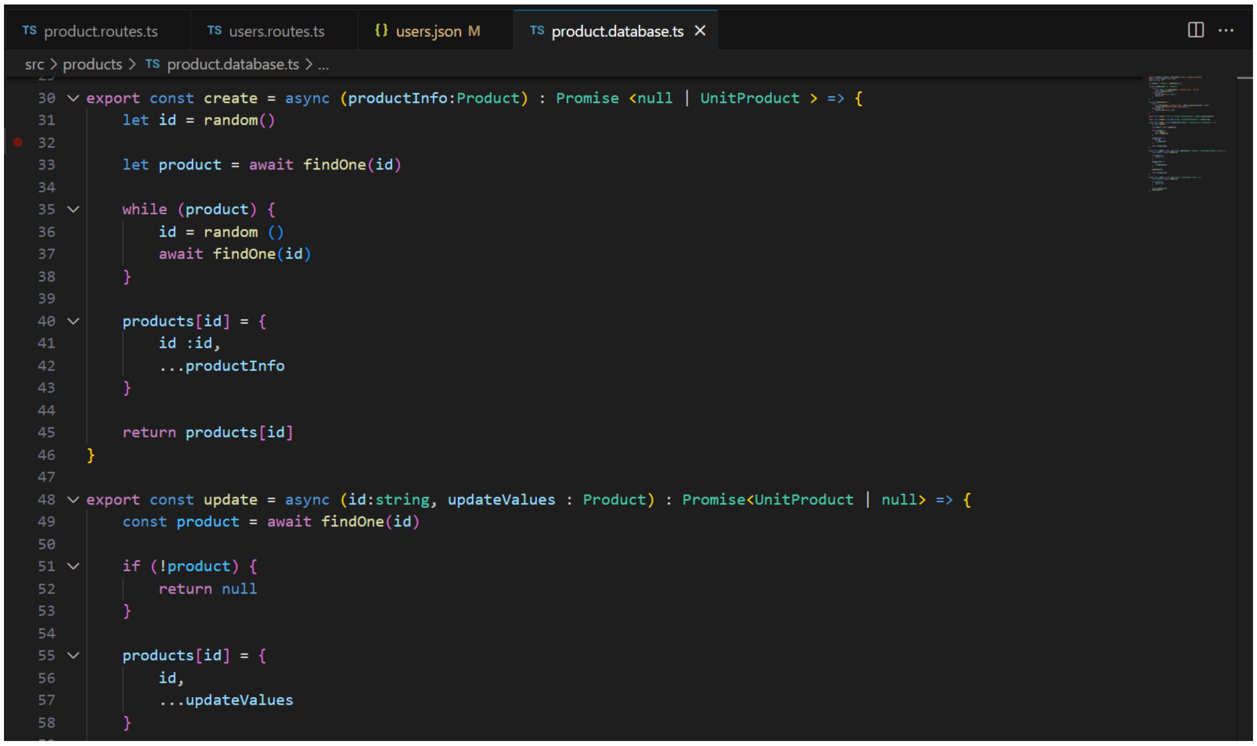
populate the `./src/products.database.ts` with a similar logic:



product.routes.ts | users.routes.ts | users.json M | product.database.ts

src > products > product.database.ts > ...

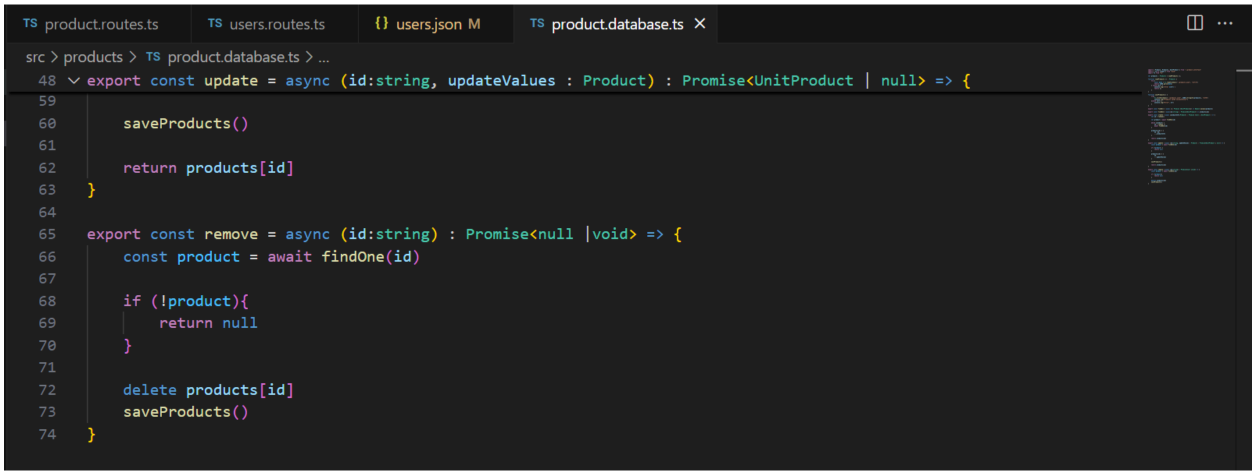
```typescript
import { Product, Products, UnitProduct } from "./product.interface"
import { v4 as random } from "uuid"
import fs from "fs"

let products : Products = loadProducts ();

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

function saveProducts() {
    try{
        fs.writeFileSync("./products.json", JSON.stringify(products), "utf8")
        console.log("Products saved successfully!")
    }catch (err) {
        console.log("Error", err)
    }
}

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 > => {
```



product.routes.ts | users.routes.ts | users.json M | product.database.ts

src > products > product.database.ts > ...

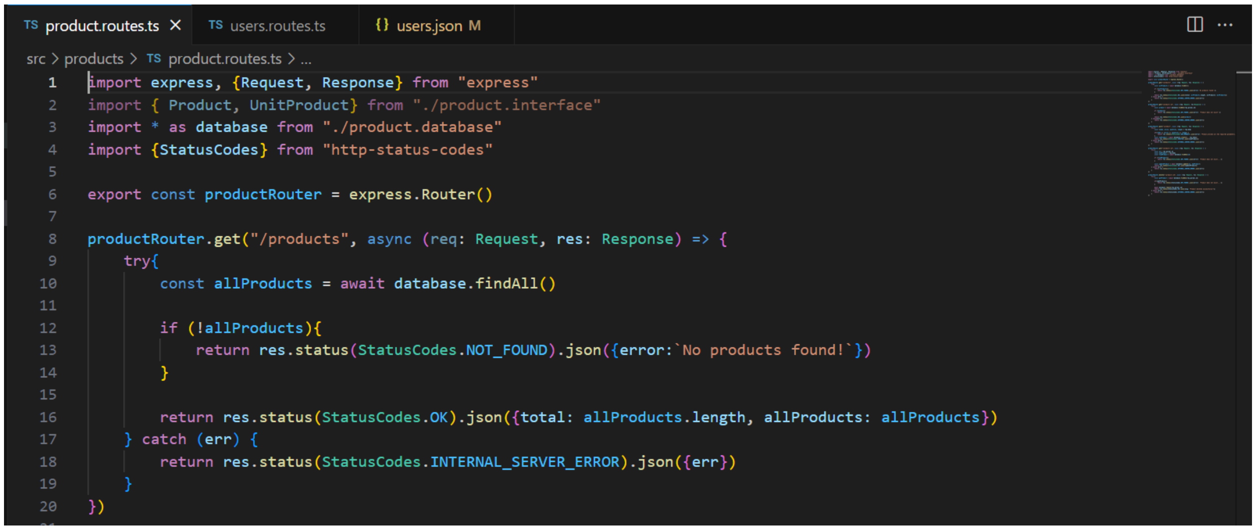
```typescript
export const create = async (productInfo:Product) : Promise <null | UnitProduct > => {
    let id = random()

    let product = await findOne(id)

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

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

    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
    }
```


TS product.routes.ts | TS users.routes.ts | {} users.json M | TS product.database.ts ×

src > products > TS product.database.ts > ...

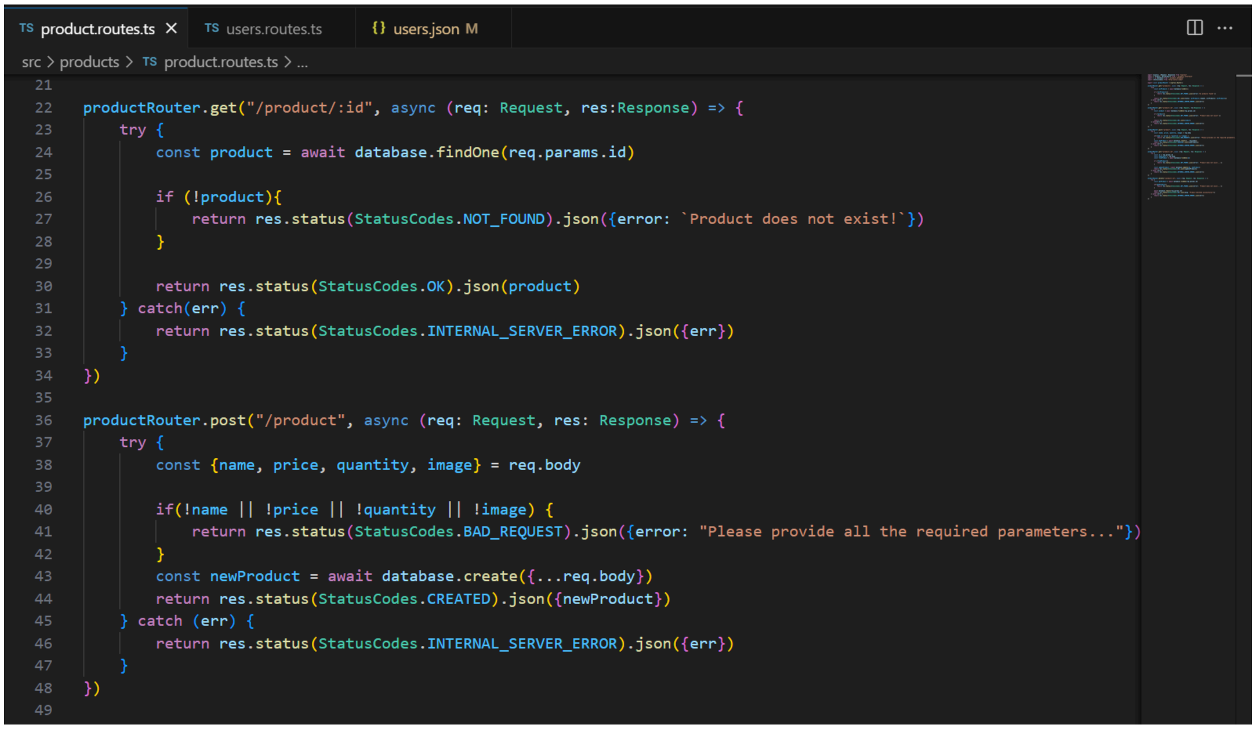
```
export const update = async (id:string, updateValues : Product) : Promise<UnitProduct | null> => {

    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()
}
```

implement the routes for our products:

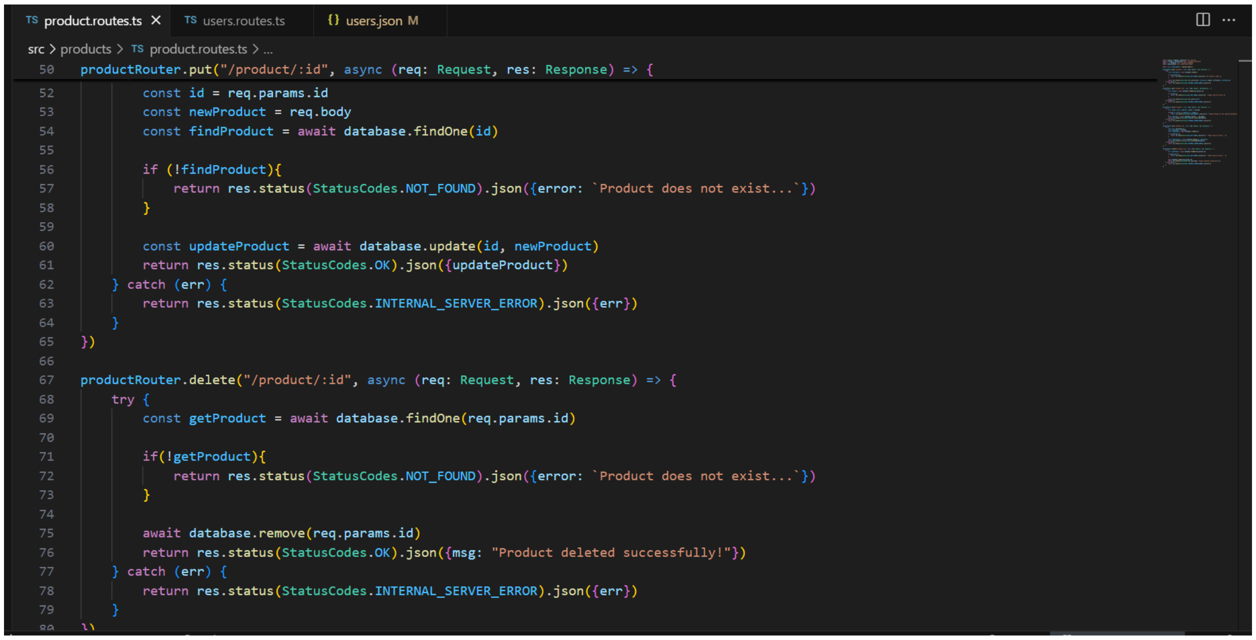

TS product.routes.ts × | TS users.routes.ts | {} users.json M

src > products > TS product.routes.ts > ...

```
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: allProducts})
    } catch (err) {
        return res.status(StatusCodes.INTERNAL_SERVER_ERROR).json({err})
    }
})
```

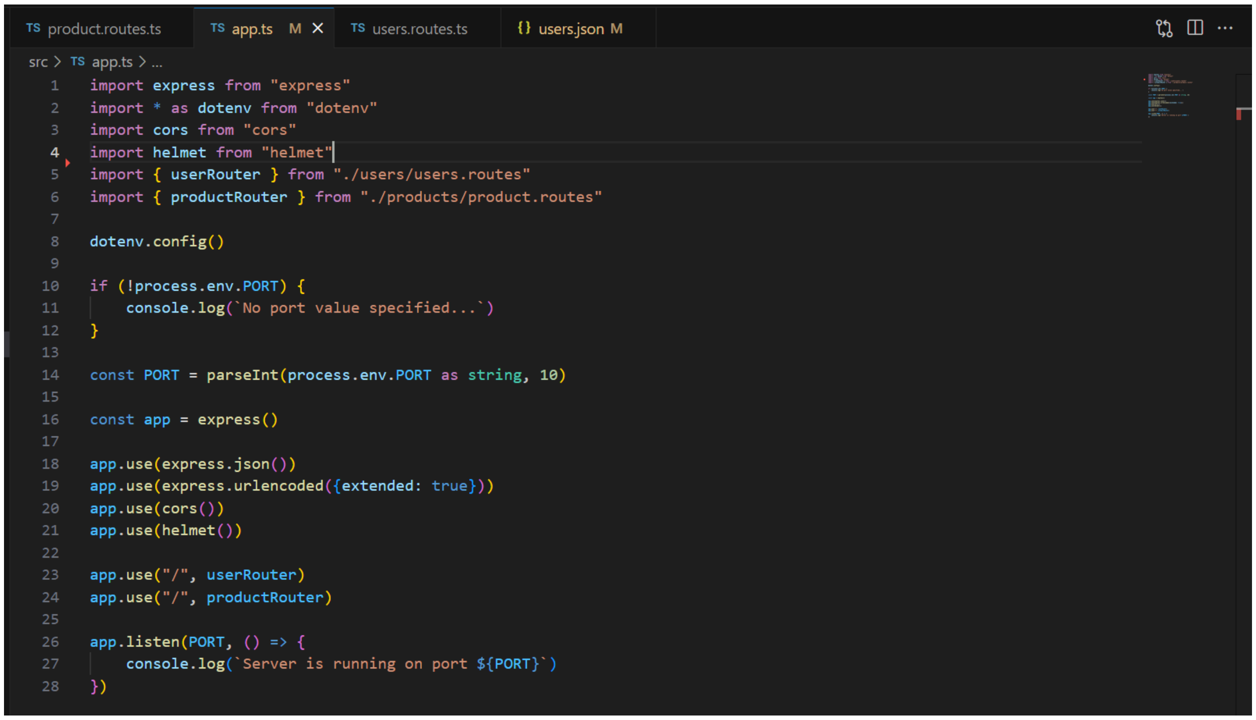


```
TS product.routes.ts ×    TS users.routes.ts    {} users.json M
src > products > TS product.routes.ts > ...

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(err) {
        return res.status(StatusCodes.INTERNAL_SERVER_ERROR).json({err})
    }
})

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 (err) {
        return res.status(StatusCodes.INTERNAL_SERVER_ERROR).json({err})
    }
})

```



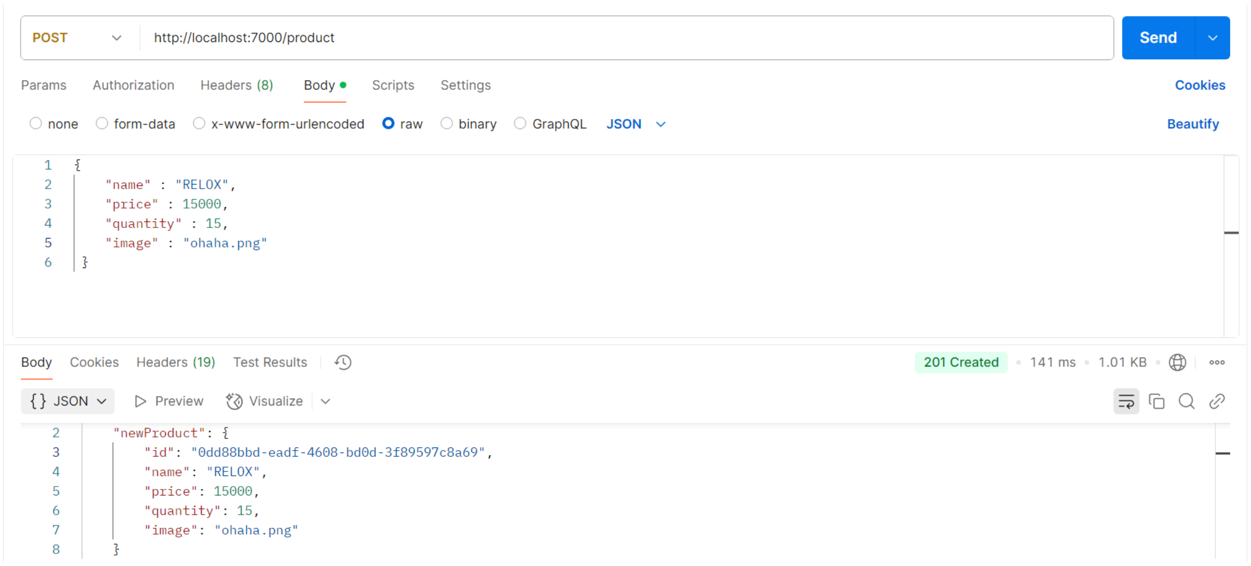
```
TS product.routes.ts ×    TS users.routes.ts    {} users.json M
src > products > TS product.routes.ts > ...
productRouter.put("/product/:id", async (req: Request, res: Response) => {
        const id = req.params.id
        const newProduct = req.body
        const findProduct = await database.findOne(id)

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

        const updateProduct = await database.update(id, newProduct)
        return res.status(StatusCodes.OK).json({updateProduct})
    } catch (err) {
        return res.status(StatusCodes.INTERNAL_SERVER_ERROR).json({err})
    }
})

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: `Product does not exist...`})
        }

        await database.remove(req.params.id)
        return res.status(StatusCodes.OK).json({msg: "Product deleted successfully!"})
    } catch (err) {
        return res.status(StatusCodes.INTERNAL_SERVER_ERROR).json({err})
    }
})
```

import and call the product's route:



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

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

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

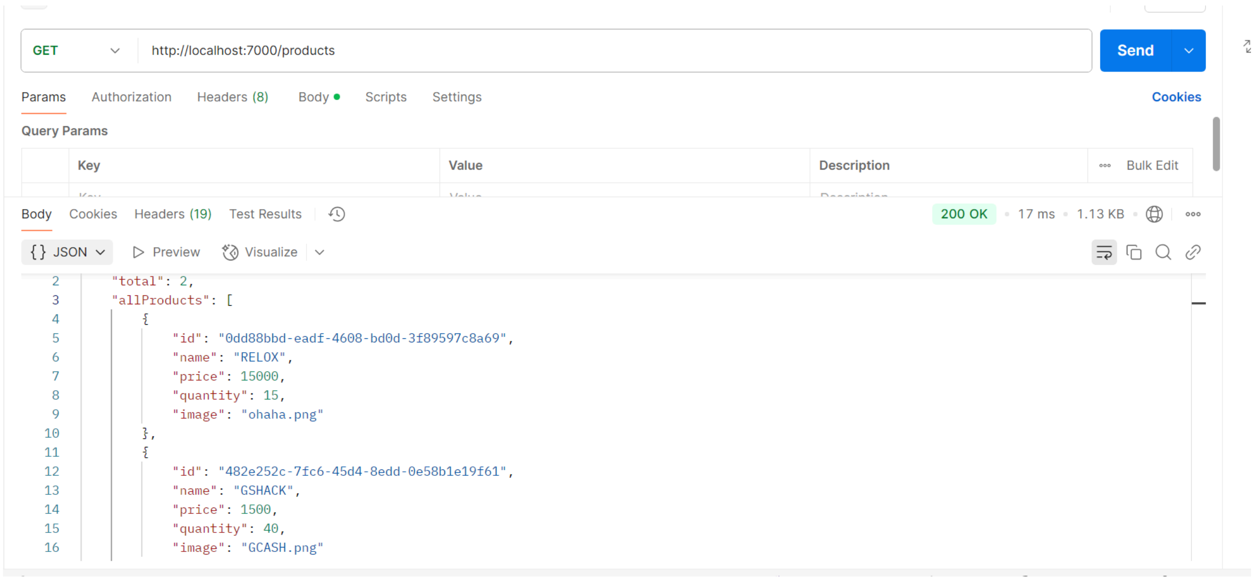
Create product:



```json
{
    "name" : "RELOX",
    "price" : 15000,
    "quantity" : 15,
    "image" : "ohaha.png"
}
```

```json
    "newProduct": {
        "id": "0dd88bbd-eadf-4608-bd0d-3f89597c8a69",
        "name": "RELOX",
        "price": 15000,
        "quantity": 15,
        "image": "ohaha.png"
    }
```

All products:


GET http://localhost:7000/products

Params | Authorization | Headers (8) | Body | Scripts | Settings | Cookies

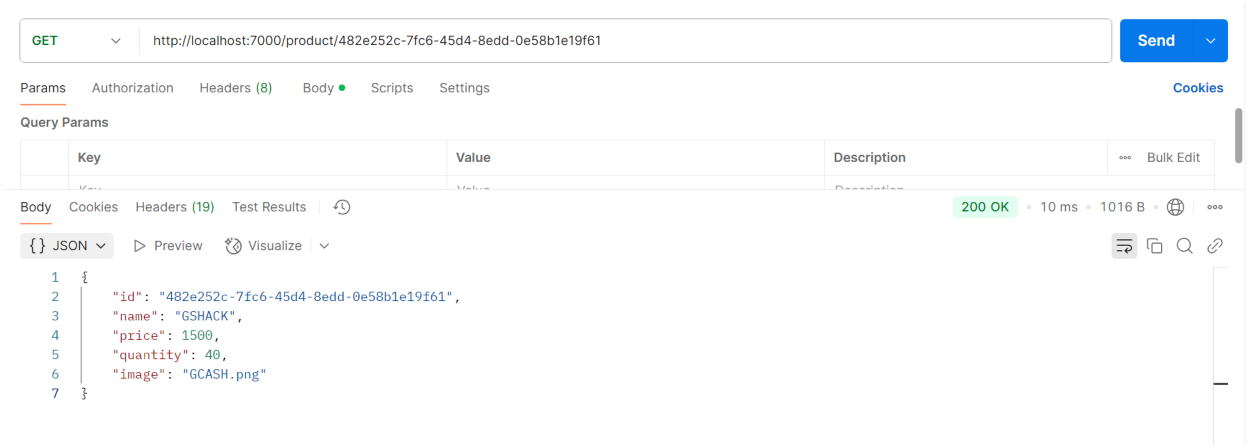
Query Params

Body | Cookies | Headers (19) | Test Results — 200 OK • 17 ms • 1.13 KB

{} JSON | Preview | Visualize

```
    "total": 2,
    "allProducts": [
        {
            "id": "0dd88bbd-eadf-4608-bd0d-3f89597c8a69",
            "name": "RELOX",
            "price": 15000,
            "quantity": 15,
            "image": "ohaha.png"
        },
        {
            "id": "482e252c-7fc6-45d4-8edd-0e58b1e19f61",
            "name": "GSHACK",
            "price": 1500,
            "quantity": 40,
            "image": "GCASH.png"
```

Single product:


GET http://localhost:7000/product/482e252c-7fc6-45d4-8edd-0e58b1e19f61

Params | Authorization | Headers (8) | Body | Scripts | Settings | Cookies

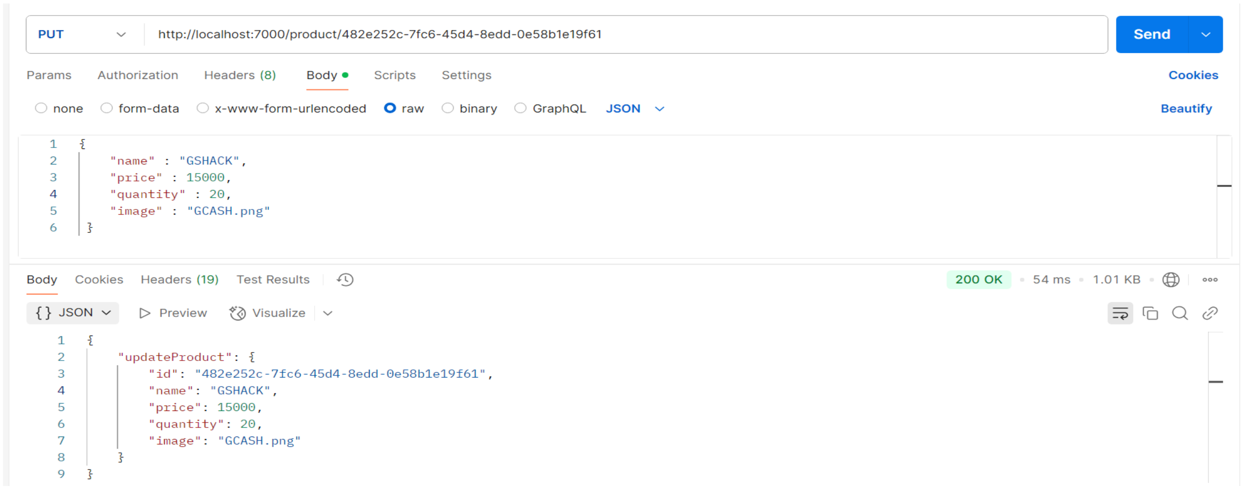
Query Params

Body | Cookies | Headers (19) | Test Results — 200 OK • 10 ms • 1016 B

{} JSON | Preview | Visualize

```
{
    "id": "482e252c-7fc6-45d4-8edd-0e58b1e19f61",
    "name": "GSHACK",
    "price": 1500,
    "quantity": 40,
    "image": "GCASH.png"
}
```

Update product:


PUT http://localhost:7000/product/482e252c-7fc6-45d4-8edd-0e58b1e19f61

Params | Authorization | Headers (8) | Body | Scripts | Settings | Cookies

none | form-data | x-www-form-urlencoded | [✓ raw] | binary | GraphQL | JSON | Beautify

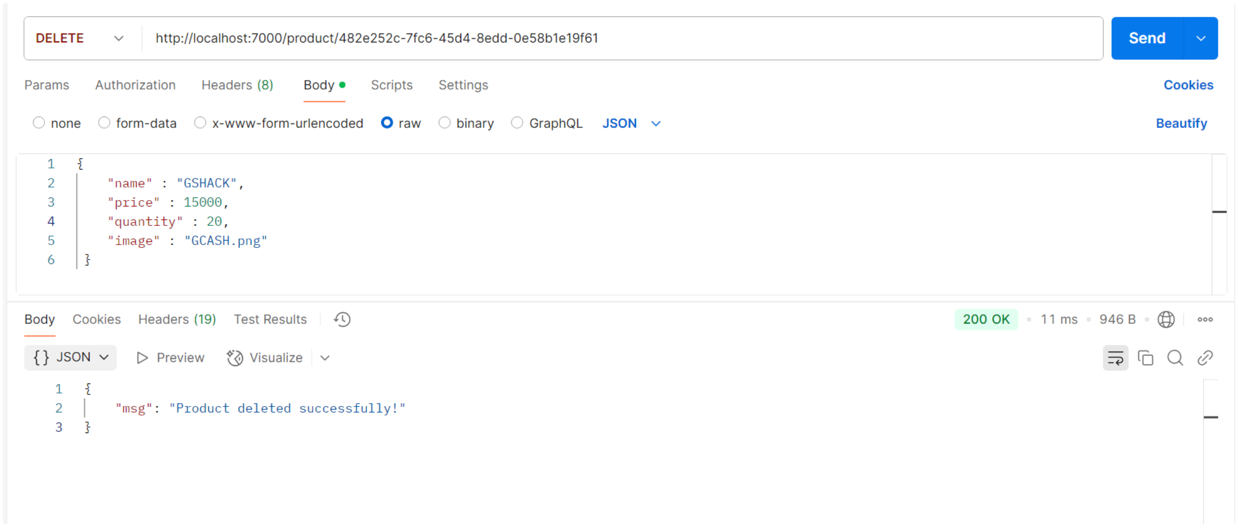
```
{
    "name" : "GSHACK",
    "price" : 15000,
    "quantity" : 20,
    "image" : "GCASH.png"
}
```

Body | Cookies | Headers (19) | Test Results — 200 OK • 54 ms • 1.01 KB

{} JSON | Preview | Visualize

```
{
    "updateProduct": {
        "id": "482e252c-7fc6-45d4-8edd-0e58b1e19f61",
        "name": "GSHACK",
        "price": 15000,
        "quantity": 20,
        "image": "GCASH.png"
    }
}
```

Delete product:



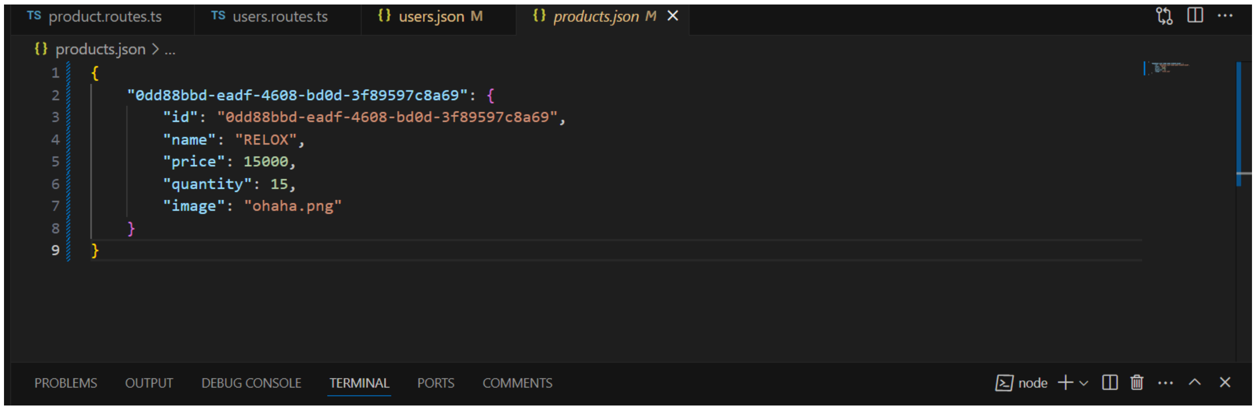
DELETE http://localhost:7000/product/482e252c-7fc6-45d4-8edd-0e58b1e19f61

```
{
    "name" : "GSHACK",
    "price" : 15000,
    "quantity" : 20,
    "image" : "GCASH.png"
}
```

200 OK • 11 ms • 946 B

```
{
    "msg": "Product deleted successfully!"
}
```

Products-data-storage-file :



products.json

```
{
    "0dd88bbd-eadf-4608-bd0d-3f89597c8a69": {
        "id": "0dd88bbd-eadf-4608-bd0d-3f89597c8a69",
        "name": "RELOX",
        "price": 15000,
        "quantity": 15,
        "image": "ohaha.png"
    }
}
```

Conclusion:

The activity was successfully completed, and all objectives were met. The GitHub repository for this project can be found here: https://github.com/papski1/act4_build_rest_api.git