

Title: Build a REST API with TS, Node, Express & File based Storage System

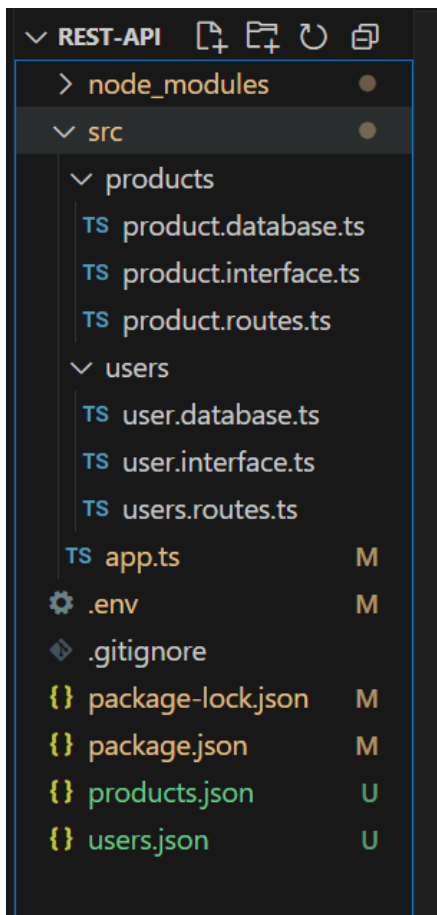
Author: Rey Mark Rivas

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:



Problem initializing:

```
C:\Windows\System32\cmd.exe X + v
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_510Z-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/papskil/REST-API.git"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "bugs": {
    "url": "https://github.com/papskil/REST-API/issues"
  },
  "homepage": "https://github.com/papskil/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://github.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 if you want a good and tested way to coalesce async requests by a key value, which is much more comprehensive and powerful.
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:

```
1  import express from "express"
2  import * as dotenv from "dotenv"
3  import cors from "cors"
4  import helmet from "helmet"
5
6  dotenv.config()
7
8  if (!process.env.PORT){
9    |   console.log(`No port value specified...`)
10  }
11
12  const PORT = parseInt(process.env.PORT as string, 10)
13
14  const app = express()
15
16  app.use(express.json())
17  app.use(express.urlencoded({extended : true}))
18  app.use(cors())
19  app.use(helmet())
20
21  app.listen(PORT, () => {
22    |   console.log(`Server is listening on port ${PORT}`)
23  })
```

Updated package.json

```
1  {
2    "name": "typescript-nodejs",
3    "version": "1.0.0",
4    "description": "",
5    "main": "index.js",
6    "scripts": {
7      "test": "echo \"Error: no test specified\" && exit 1",
8      "dev": "ts-node-dev --pretty --respawn ./src/app.ts"
9    },
10   "keywords": [],
11   "author": "",
12   "license": "ISC",
13   "dependencies": {
14     "@types/nanoid": "^3.0.0",
15     "@types/uuid": "^9.0.2",
16     "bcryptjs": "^2.4.3",
17     "cors": "^2.8.5",
18     "dotenv": "^16.3.0",
19     "express": "^4.18.2",
20     "helmet": "^7.0.0",
21     "http-status-codes": "^2.2.0",
22     "nanoid": "^4.0.2",
23     "uuid": "^9.0.0"
24   },
25   "devDependencies": {
26     "@types/bcryptjs": "^2.4.2",
27     "@types/cors": "^2.8.13",
28     "@types/dotenv": "^8.2.0",
29     "@types/express": "^4.17.17",
30     "@types/helmet": "^4.0.0",
31     "@types/http-status-codes": "^1.2.0",
32     "ts-node-dev": "^2.0.0"
33   }
34 }
```

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
1  export interface User {
2      username : string,
3      email : string,
4      password : string
5  }
6
7  export interface UnitUser extends User {
8      id : string
9  }
10
11 export interface Users {
12     [key : string] : UnitUser
13 }
```

Populate src/users/user.database.ts:

```
1  import { User, UnitUser, Users } from "../user.interface";
2  import bcrypt from "bcryptjs"
3  import {NIL, v4 as random} from "uuid"
4  import fs from "fs"
5
6  let users: Users = loadUsers()
7
8  function loadUsers () : Users {
9      try {
10         const data = fs.readFileSync("../users.json", "utf-8")
11         return JSON.parse(data)
12     } catch (error) {
13         console.log(`Error ${error}`)
14         return {}
15     }
16 }
17
18 function saveUsers () {
19     try {
```

```

20     fs.writeFileSync("./users.json", JSON.stringify(users), "utf-8")
21     console.log(`User saved successfully!`)
22   } catch (error) {
23     console.log(`Error : ${error}`)
24   }
25 }
26
27 export const findAll = async (): Promise<UnitUser[]> => Object.values(users);
28
29 export const findOne = async (id: string): Promise<UnitUser> => users[id];
30
31 export const create = async (userData: UnitUser): Promise<UnitUser | null> => {
32
33   let id = random()
34
35   let check_user = await findOne(id);

```

```

37   while (check_user) {
38     id = random()
39     check_user = await findOne(id)
40   }
41
42   const salt = await bcrypt.genSalt(10);
43
44   const hashedPassword = await bcrypt.hash(userData.password, salt);
45
46   const user : UnitUser = {
47     id : id,
48     username : userData.username,
49     email : userData.email,
50     password : hashedPassword
51   };

```

```

53   users[id] = user;
54
55   saveUsers()
56
57   return user;
58 };
59
60 export const findByEmail = async (user_email: string): Promise<null | UnitUser> => {
61
62   const allUsers = await findAll();
63
64   const getUser = allUsers.find(result => user_email === result.email);
65
66   if (!getUser) {
67     return null;
68   }

```

```

70     return getUser;
71 };
72
73 export const comparePassword = async (email : string, supplied_password : string) : Promise<null | UnitUser> => {
74
75     const user = await findByEmail(email)
76
77     const decryptPassword = await bcrypt.compare(supplied_password, user!.password)
78
79     if (!decryptPassword) {
80         return null
81     }
82
83     return user
84 }

```

```

86 export const update = async (id : string, updateValues : User) : Promise<UnitUser | null> => {
87
88     const userExists = await findOne(id)
89
90     if (!userExists) {
91         return null
92     }
93
94     if(updateValues.password) {
95         const salt = await bcrypt.genSalt(10)
96         const newPass = await bcrypt.hash(updateValues.password, salt)
97
98         updateValues.password = newPass
99     }
100
101     users[id] = {
102         ...userExists,
103         ...updateValues

```

```

104     }
105
106     saveUsers()
107
108     return users[id]
109 }
110
111 export const remove = async (id : string) : Promise<null | void> => {
112
113     const user = await findOne(id)
114
115     if (!user) {
116         return null
117     }
118
119     delete users[id]
120

```

```

121     saveUsers()
122 }

```


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
1  import express, {Request, Response} from "express"
2  import { UnitUser, User } from "../user.interface"
3  import {StatusCodes} from "http-status-codes"
4  import * as database from "../user.database"
5
6  export const userRouter = express.Router()
7
8  userRouter.get("/users", async (req : Request, res : Response) => {
9      try {
10         const allUsers : UnitUser[] = await database.findAll()
11
12         if (!allUsers) {
13             return res.status(StatusCodes.NOT_FOUND).json({msg : `No users at this time..`})
14         }
15
16         return res.status(StatusCodes.OK).json({total_user : allUsers.length, allUsers})
17     } catch (error) {
18         return res.status(StatusCodes.INTERNAL_SERVER_ERROR).json({error})
19     }
20 })
21
22 userRouter.get("/user/:id", async (req : Request, res : Response) =>{
23     try {
24         const user : UnitUser = await database.findOne(req.params.id)
25
26         if (!user) {
27             return res.status(StatusCodes.NOT_FOUND).json({error : `User not found!`})
28         }
29
30         return res.status(StatusCodes.OK).json({user})
31     } catch (error) {
32         return res.status(StatusCodes.INTERNAL_SERVER_ERROR).json({error})
33     }
34 })
35
36 userRouter.post("/register", async (req : Request, res : Response) => {
37     try {
38         const { username, email, password } = req.body
39
40         if (!username || !email || !password) {
41             return res.status(StatusCodes.BAD_REQUEST).json({error : `Please provide all the required parameters..`})
42         }
43
44         const user = await database.findByEmail(email)
45
46         if (user) {
47             return res.status(StatusCodes.BAD_REQUEST).json({error : `This email has already been registerd..`})
48         }
49
50         const newUser = await database.create(req.body)
51
52         return res.status(StatusCodes.CREATED).json({newUser})
53     } catch (error) {
54         return res.status(StatusCodes.INTERNAL_SERVER_ERROR).json({error})
55     }
56 }
57 })
```

```

59 userRouter.post("/login", async (req : Request, res : Response) => {
60   try {
61     const {email, password} = req.body
62
63     if (!email || !password) {
64       return res.status(StatusCodes.BAD_REQUEST).json({error : "Please provide all the required parameters.."})
65     }
66
67     const user = await database.findByEmail(email)
68
69     if (!user) {
70       return res.status(StatusCodes.NOT_FOUND).json({error : "No user exists with the email provided.."})
71     }
72
73     const comparePassword = await database.comparePassword(email, password)
74
75     if (!comparePassword) {
76       return res.status(StatusCodes.BAD_REQUEST).json({error : `Incorrect Password!`})
77     }
78
79     return res.status(StatusCodes.OK).json({user})
80
81   } catch (error) {
82     return res.status(StatusCodes.INTERNAL_SERVER_ERROR).json({error})
83   }
84 })
85

```

```

87 userRouter.put('/user/:id', async (req : Request, res : Response) => {
88
89   try {
90
91     const {username, email, password} = req.body
92
93     const getUser = await database.findOne(req.params.id)
94
95     if(!username || !email || !password) {
96       return res.status(401).json({error : `Please provide all the required parameters..`})
97     }
98
99     if (!getUser) {
100       return res.status(404).json({error : `No user with id ${req.params.id}`})
101     }
102
103     const updateUser = await database.update((req.params.id), req.body)
104
105     return res.status(201).json({updateUser})
106   } catch (error) {
107     console.log(error)
108     return res.status(500).json({error})
109   }
110 })
111

```

```

112 userRouter.delete("/user/:id", async (req : Request, res : Response) => {
113   try {
114     const id = (req.params.id)
115
116     const user = await database.findOne(id)
117
118     if (!user) {
119       return res.status(StatusCodes.NOT_FOUND).json({error : `User does not exist`})
120     }
121
122     await database.remove(id)
123
124     return res.status(StatusCodes.OK).json({msg : "User deleted"})
125   } catch (error) {
126     return res.status(StatusCodes.INTERNAL_SERVER_ERROR).json({error})
127   }
128 })

```

Updated app.ts:

```
1  import express from "express"
2  import * as dotenv from "dotenv"
3  import cors from "cors"
4  import helmet from "helmet"
5  import { userRouter } from "./users/users.routes"
6  import { productRouter } from "./products/product.routes"
7
8  dotenv.config()
9
10 if (!process.env.PORT) {
11   console.log(`No port value specified...`)
12 }
13
14 const PORT = parseInt(process.env.PORT as string, 10)
15
16 const app = express()
17
18 app.use(express.json())
19 app.use(express.urlencoded({extended : true}))
20 app.use(cors())
21 app.use(helmet())
22
23 app.use('/', userRouter)
24 app.use('/', productRouter)
25
26 app.listen(PORT, () => {
27   console.log(`Server is listening on port ${PORT}`)
28 })
29
```

Register users:

POST http://localhost:7000/register Send

Params Authorization Headers (8) Body Scripts Settings Cookies Beautify

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL JSON

```
1 {
2   "username": "ReyMark",
3   "email": "reymarkrivas61@gmail.com",
4   "password": "lakai2021."
5 }
```

Body Cookies Headers (19) Test Results 201 Created 380 ms 1.07 KB

{ JSON Preview Visualize

```
1 {
2   "newUser": {
3     "id": "5e9a06ee-eefa-4567-aa91-851b59aaba86",
4     "username": "ReyMark",
5     "email": "reymarkrivas61@gmail.com",
6     "password": "$2b$10$Q76ivheblWtg6zrontsvXSedg8e0/iWZbZvwQBZT0uEZp2awUBVRGw"
7   }
8 }
```

Login users:

http://localhost:7000/login Save Share

POST http://localhost:7000/login Send

Params Authorization Headers (8) Body Scripts Settings Cookies Beautify

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL JSON

```
1 {
2   "email": "reymarkrivas61@gmail.com",
3   "password": "lakai2021."
4 }
```

Body Cookies Headers (19) Test Results 200 OK 342 ms 1.06 KB

{ JSON Preview Visualize

```
1 {
2   "id": "5e9a06ee-eefa-4567-aa91-851b59aaba86",
3   "username": "ReyMark",
4   "email": "reymarkrivas61@gmail.com",
5   "password": "$2b$10$Q76ivheblWtg6zrontsvXSedg8e0/iWZbZvwQBZT0uEZp2awUBVRGw"
6 }
```

Get all users:

http://localhost:7000/users Save Share

GET Send

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

Query Params

Key	Value	Description	Bulk Edit
Key	Value	Description	

Body Cookies Headers (19) Test Results 200 OK 21 ms 1.09 KB ...

JSON Preview Visualize

```
1 {
2   "total_user": 1,
3   "allUsers": [
4     {
5       "id": "5e9a06ee-eefa-4567-aa91-851b59aaba86",
6       "username": "ReyMark",
7       "email": "reymarkrivas61@gmail.com",
```

Get a single user:

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

GET Send

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

Query Params


Key	Value	Description	Bulk Edit
Key	Value	Description	

Body Cookies Headers (19) Test Results 200 OK 12 ms 1.07 KB ...

JSON Preview Visualize

```
1 {
2   "user": {
3     "id": "5e9a06ee-eefa-4567-aa91-851b59aaba86",
4     "username": "ReyMark",
5     "email": "reymarkrivas61@gmail.com",
6     "password": "$2b$10$Q76ivheblwtg6rontsvXSedg8e0/iWZbZvwQBZT0uEZp2awUBVRGW"
7   }
}
```

Update user:

 <http://localhost:7000/user/5e9a06ee-eefa-4567-aa91-851b59aaba86> Save Share

GET


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


Send

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

Query Params

	Key	Value	Description	...	Bulk Edit
	Key	Value	Description		


Body Cookies Headers (19) Test Results 

200 OK • 8 ms • 1.07 KB •  ...

{}

JSON

Preview

 Visualize

1 {

2 "user": {

3 "id": "5e9a06ee-eefa-4567-aa91-851b59aaba86",

4 "username": "ReyMark",

5 "email": "reymarkrivas61@gmail.com",

6 "password": "\$2b\$16\$Q76ivheblwtg6rontsvXSedg8e0/iWZbZvwQBZT0uEZp2awUBVRGM"

7 }

Delete user:

DELETE


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


Send

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

Query Params

	Key	Value	Description	...	Bulk Edit
	Key	Value	Description		


Body Cookies Headers (19) Test Results 

200 OK • 20 ms • 930 B •  ...

{}

JSON

Preview

 Visualize

1 {

2 "msg": "User Deleted!"

3 }

Users-data-storage-file :

```
TS product.routes.ts TS users.routes.ts {} users.json M X {} products.json
{} users.json > ...
1  {
2    "16d062bd-c166-4b05-a744-1f38f62acb5c": {
3      "id": "16d062bd-c166-4b05-a744-1f38f62acb5c",
4      "username": "papskikoy",
5      "email": "kikoypaps1@gmail.com",
6      "password": "$2b$10$j5Qkc7ARUHMsjNjJ..L0nOgqOpF0BZn2Wxs8gL7ErA2y7XkSxh2zq"
7    },
8    "640c8fa4-16b9-4509-b461-0829bcd61058": {
9      "id": "640c8fa4-16b9-4509-b461-0829bcd61058",
10     "username": "papsleroi",
11     "email": "papsleroi1@gmail.com",
12     "password": "$2b$10$YucGGiKRr6oQuXp4PZDPkODBFrgPBv12FV9lQxmzbSApsBcZdgfXe"
13   }
14 }
```

create the login and routes for our products:

```
TS product.routes.ts TS users.routes.ts {} users.json M TS product.interface.ts X
src > products > TS product.interface.ts > Product
1  export interface Product {
2    name : string,
3    price: number;
4    quantity : number;
5    image : string;
6  }
7
8  export interface UnitProduct extends Product {
9    id : string
10 }
11
12 export interface Products {
13   [key: string] : UnitProduct;
14 }
```

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

```
TS product.routes.ts TS users.routes.ts {} users.json M TS product.database.ts X
src > products > TS product.database.ts > ...
1 import { Product, Products, UnitProduct } from "../product.interface"
2 import { v4 as random } from "uuid"
3 import fs from "fs"
4
5 let products : Products = loadProducts ();
6
7 function loadProducts () : Products {
8   try {
9     const data = fs.readFileSync("./products.json", "utf-8")
10    return JSON.parse(data)
11  } catch (err) {
12    console.log(`Error ${err}`)
13    return {}
14  }
15 }
16
17 function saveProducts() {
18   try{
19     fs.writeFileSync("./products.json", JSON.stringify(products), "utf8")
20     console.log("Products saved successfully!")
21   }catch (err) {
22     console.log("Error", err)
23   }
24 }
25
26 export const findAll = async (): Promise <UnitProduct[]> => Object.values(products)
27
28 export const findOne = async(id:string) : Promise<UnitProduct> => products[id]
29
30 export const create = async (productInfo:Product) : Promise <null | UnitProduct > => {
```

```
TS product.routes.ts TS users.routes.ts {} users.json M TS product.database.ts X
src > products > TS product.database.ts > ...
30 export const create = async (productInfo:Product) : Promise <null | UnitProduct > => {
31   let id = random()
32
33   let product = await findOne(id)
34
35   while (product) {
36     id = random ()
37     await findOne(id)
38   }
39
40   products[id] = {
41     id :id,
42     ...productInfo
43   }
44
45   return products[id]
46 }
47
48 export const update = async (id:string, updateValues : Product) : Promise<UnitProduct | null> => {
49   const product = await findOne(id)
50
51   if (!product) {
52     return null
53   }
54
55   products[id] = {
56     id,
57     ...updateValues
58   }
59 }
```



```
TS product.routes.ts TS users.routes.ts {} users.json M TS product.database.ts X
src > products > TS product.database.ts > ...
48 export const update = async (id:string, updateValues : Product) : Promise<UnitProduct | null> => {
59     saveProducts()
60
61     return products[id]
62 }
63
64
65 export const remove = async (id:string) : Promise<null |void> => {
66     const product = await findOne(id)
67
68     if (!product){
69         return null
70     }
71
72     delete products[id]
73     saveProducts()
74 }
```

implement the routes for our products:

```
TS product.routes.ts X TS users.routes.ts {} users.json M
src > products > TS product.routes.ts > ...
1 import express, {Request, Response} from "express"
2 import { Product, UnitProduct} from "../product.interface"
3 import * as database from "../product.database"
4 import {StatusCodes} from "http-status-codes"
5
6 export const productRouter = express.Router()
7
8 productRouter.get("/products", async (req: Request, res: Response) => {
9     try{
10         const allProducts = await database.findAll()
11
12         if (!allProducts){
13             return res.status(StatusCodes.NOT_FOUND).json({error:`No products found!`})
14         }
15
16         return res.status(StatusCodes.OK).json({total: allProducts.length, allProducts: allProducts})
17     } catch (err) {
18         return res.status(StatusCodes.INTERNAL_SERVER_ERROR).json({err})
19     }
20 })
```

```
TS product.routes.ts X TS users.routes.ts {} users.json M
src > products > TS product.routes.ts > ...
21
22 productRouter.get("/product/:id", async (req: Request, res: Response) => {
23     try {
24         const product = await database.findOne(req.params.id)
25
26         if (!product){
27             return res.status(StatusCodes.NOT_FOUND).json({error: `Product does not exist!`})
28         }
29
30         return res.status(StatusCodes.OK).json(product)
31     } catch(err) {
32         return res.status(StatusCodes.INTERNAL_SERVER_ERROR).json({err})
33     }
34 })
35
36 productRouter.post("/product", async (req: Request, res: Response) => {
37     try {
38         const {name, price, quantity, image} = req.body
39
40         if(!name || !price || !quantity || !image) {
41             return res.status(StatusCodes.BAD_REQUEST).json({error: "Please provide all the required parameters..."})
42         }
43         const newProduct = await database.create({...req.body})
44         return res.status(StatusCodes.CREATED).json({newProduct})
45     } catch (err) {
46         return res.status(StatusCodes.INTERNAL_SERVER_ERROR).json({err})
47     }
48 })
49
```

```
TS product.routes.ts X TS users.routes.ts {} users.json M
src > products > TS product.routes.ts > ...
50 productRouter.put("/product/:id", async (req: Request, res: Response) => {
51
52     const id = req.params.id
53     const newProduct = req.body
54     const findProduct = await database.findOne(id)
55
56     if (!findProduct){
57         return res.status(StatusCodes.NOT_FOUND).json({error: `Product does not exist...`})
58     }
59
60     const updateProduct = await database.update(id, newProduct)
61     return res.status(StatusCodes.OK).json({updateProduct})
62 } catch (err) {
63     return res.status(StatusCodes.INTERNAL_SERVER_ERROR).json({err})
64 }
65 })
66
67 productRouter.delete("/product/:id", async (req: Request, res: Response) => {
68     try {
69         const getProduct = await database.findOne(req.params.id)
70
71         if(!getProduct){
72             return res.status(StatusCodes.NOT_FOUND).json({error: `Product does not exist...`})
73         }
74
75         await database.remove(req.params.id)
76         return res.status(StatusCodes.OK).json({msg: "Product deleted successfully!"})
77     } catch (err) {
78         return res.status(StatusCodes.INTERNAL_SERVER_ERROR).json({err})
79     }
80 })
```

import and call the product's route:

```
TS product.routes.ts TS app.ts M X TS users.routes.ts {} users.json M
src > TS app.ts > ...
1  import express from "express"
2  import * as dotenv from "dotenv"
3  import cors from "cors"
4  import helmet from "helmet"
5  import { userRouter } from "../users/users.routes"
6  import { productRouter } from "../products/product.routes"
7
8  dotenv.config()
9
10 if (!process.env.PORT) {
11   console.log("No port value specified...")
12 }
13
14 const PORT = parseInt(process.env.PORT as string, 10)
15
16 const app = express()
17
18 app.use(express.json())
19 app.use(express.urlencoded({extended: true}))
20 app.use(cors())
21 app.use(helmet())
22
23 app.use("/", userRouter)
24 app.use("/", productRouter)
25
26 app.listen(PORT, () => {
27   console.log(`Server is running on port ${PORT}`)
28 })
```

Create product:

POST http://localhost:7000/product Send

Params Authorization Headers (8) Body Scripts Settings Cookies

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL JSON Beautify

```
1 {
2   "name": "RELOX",
3   "price": 15000,
4   "quantity": 15,
5   "image": "ohaha.png"
6 }
```

Body Cookies Headers (19) Test Results 201 Created 141 ms 1.01 KB

{ JSON Preview Visualize

```
2 "newProduct": {
3   "id": "0dd88bbd-eadf-4608-bd0d-3f89597c8a69",
4   "name": "RELOX",
5   "price": 15000,
6   "quantity": 15,
7   "image": "ohaha.png"
8 }
```

All products:

GET ▼ http://localhost:7000/products Send ▼

Params Authorization Headers (8) Body ● Scripts Settings Cookies

Query Params

Key	Value	Description	...	Bulk Edit
-----	-------	-------------	-----	-----------

Body Cookies Headers (19) Test Results 🔄 200 OK 17 ms 1.13 KB 🌐 ...

{} JSON ▼ ▶ Preview 🔗 Visualize ▼ 🔍 🔗

```
2  {"total": 2,
3    "allProducts": [
4      {
5        "id": "0dd88bbd-eadf-4608-bd0d-3f89597c8a69",
6        "name": "RELOX",
7        "price": 15000,
8        "quantity": 15,
9        "image": "ohaha.png"
10     },
11     {
12       "id": "482e252c-7fc6-45d4-8edd-0e58b1e19f61",
13       "name": "GSHACK",
14       "price": 1500,
15       "quantity": 40,
16       "image": "GCASH.png"
17     }
18   ]
19 }
```

Single product:

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

Params Authorization Headers (8) Body ● Scripts Settings Cookies

Query Params

Key	Value	Description	...	Bulk Edit
-----	-------	-------------	-----	-----------

Body Cookies Headers (19) Test Results 🔄 200 OK 10 ms 1016 B 🌐 ...

{} JSON ▼ ▶ Preview 🔗 Visualize ▼ 🔍 🔗

```
1  {
2    "id": "482e252c-7fc6-45d4-8edd-0e58b1e19f61",
3    "name": "GSHACK",
4    "price": 1500,
5    "quantity": 40,
6    "image": "GCASH.png"
7  }
```

Update product:

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

Params Authorization Headers (8) Body ● Scripts Settings Cookies

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL JSON ▼ Beautify

```
1  {
2    "name": "GSHACK",
3    "price": 15000,
4    "quantity": 20,
5    "image": "GCASH.png"
6  }
```

Body Cookies Headers (19) Test Results 🔄 200 OK 54 ms 1.01 KB 🌐 ...

{} JSON ▼ ▶ Preview 🔗 Visualize ▼ 🔍 🔗

```
1  {
2    "updateProduct": {
3      "id": "482e252c-7fc6-45d4-8edd-0e58b1e19f61",
4      "name": "GSHACK",
5      "price": 15000,
6      "quantity": 20,
7      "image": "GCASH.png"
8    }
9  }
```

Delete product:

The screenshot shows a REST client interface with a DELETE request to `http://localhost:7000/product/482e252c-7fc6-45d4-8edd-0e58b1e19f61`. The request body is a JSON object: `{ "name": "GSHACK", "price": 15000, "quantity": 20, "image": "GCASH.png" }`. The response is a 200 OK status with a response time of 11 ms and a body of 946 B. The response body is a JSON object: `{ "msg": "Product deleted successfully!" }`.

Products-data-storage-file :

The screenshot shows a code editor with a file named `products.json`. The file contains a JSON array with one object: `[{"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