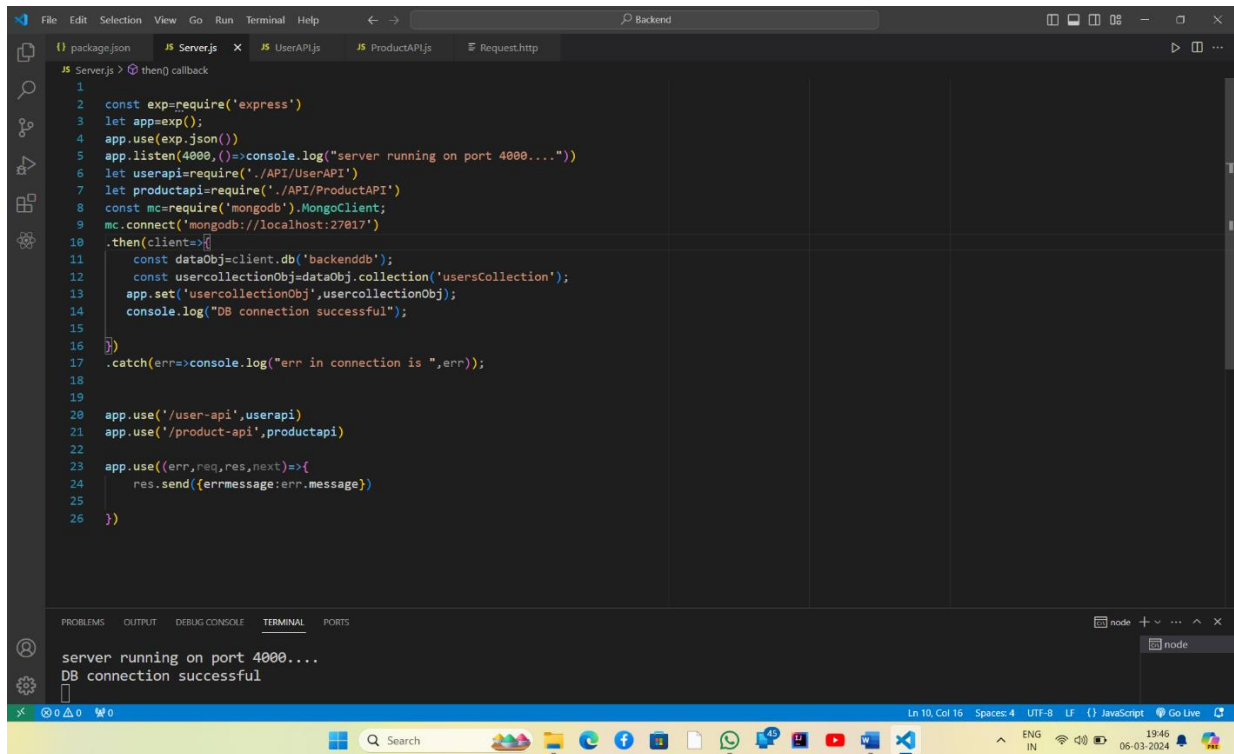


BACKEND ASSIGNMENT-2

Server.js



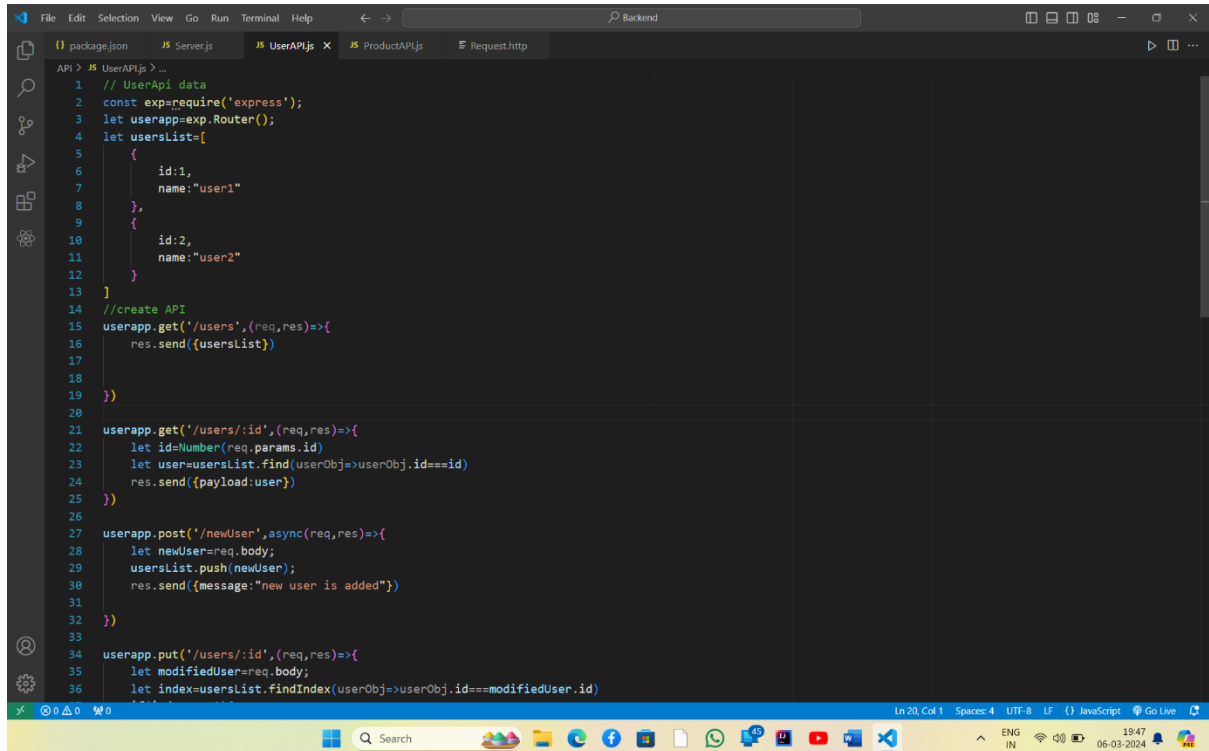
The screenshot shows the VS Code editor with the `Server.js` file open. The code defines an Express application, connects to a MongoDB database, and sets up error handling. The terminal at the bottom shows the output of running the application.

```
1 const exp=require('express')
2 let app=exp();
3 app.use(exp.json())
4 app.listen(4000,()=>console.log("server running on port 4000...."))
5 let userapi=require('./API/UserAPI')
6 let productapi=require('./API/ProductAPI')
7 const mc=require('mongodb').MongoClient;
8 mc.connect('mongodb://localhost:27017')
9 .then(client=>{
10     const dataObj=client.db('backenddb');
11     const usercollectionObj=dataObj.collection('usersCollection');
12     app.set('usercollectionObj',usercollectionObj);
13     console.log("DB connection successful");
14 })
15 .catch(err=>console.log("err in connection is ",err));
16
17 app.use('/user-api',userapi)
18 app.use('/product-api',productapi)
19
20 app.use((err,req,res,next)=>{
21     res.send({errmessage:err.message});
22 })
```

Terminal Output:

```
server running on port 4000....
DB connection successful
```

//User-API.js



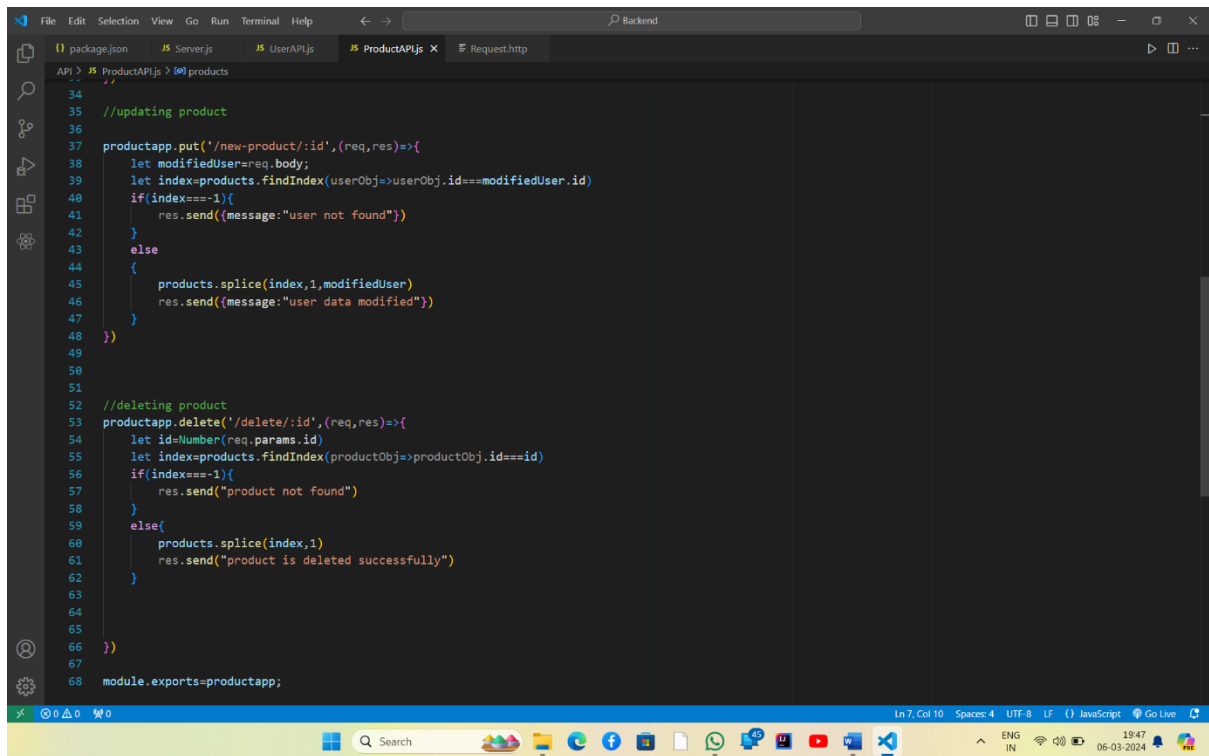
The screenshot shows the VS Code editor with the `//User-API.js` file open. The code defines a User API with endpoints for getting all users, getting a user by ID, creating a new user, and updating a user by ID.

```
1 // UserApi data
2 const exp=require('express');
3 let userapp=exp.Router();
4 let usersList=[
5     {
6         id:1,
7         name:"user1"
8     },
9     {
10        id:2,
11        name:"user2"
12    }
13 ]
14 //create API
15 userapp.get('/users',(req,res)=>{
16     res.send({usersList});
17 })
18
19
20
21 userapp.get('/users/:id',(req,res)=>{
22     let id=Number(req.params.id)
23     let user=usersList.find(userObj=>userObj.id===id)
24     res.send({payload:user})
25 })
26
27 userapp.post('/newUser',async(req,res)=>{
28     let newUser=req.body;
29     usersList.push(newUser);
30     res.send({message:"new user is added"});
31 })
32
33
34 userapp.put('/users/:id',(req,res)=>{
35     let modifiedUser=req.body;
36     let index=usersList.findIndex(userObj=>userObj.id===modifiedUser.id)
```

```
File Edit Selection View Go Run Terminal Help Backend
package.json JS Server.js JS UserAPI.js JS ProductAPI.js Request.http
API > JS UserAPI.js > ...
27 userapp.post('/newUser', async(req, res) => {
28   // new user = req.body
29   usersList.push(newUser);
30   res.send({message: "new user is added"});
31 }
32 }
33
34 userapp.put('/users/:id', (req, res) => {
35   let modifiedUser = req.body;
36   let index = usersList.findIndex(userObj => userObj.id === modifiedUser.id)
37   if(index === -1) {
38     res.send({message: "user not found"});
39   }
40   else {
41     {
42       usersList.splice(index, 1, modifiedUser)
43       res.send({message: "user data modified"});
44     }
45   }
46 }
47
48 userapp.delete('/users/:id', (req, res) => {
49   let id = Number(req.params.id)
50   let index = usersList.findIndex(userObj => userObj.id === id)
51   if(index === -1) {
52     res.send({message: "user not found"});
53   }
54   else {
55     usersList.splice(index, 1)
56     res.send({message: "data is deleted"});
57   }
58 }
59 module.exports = userapp;
```

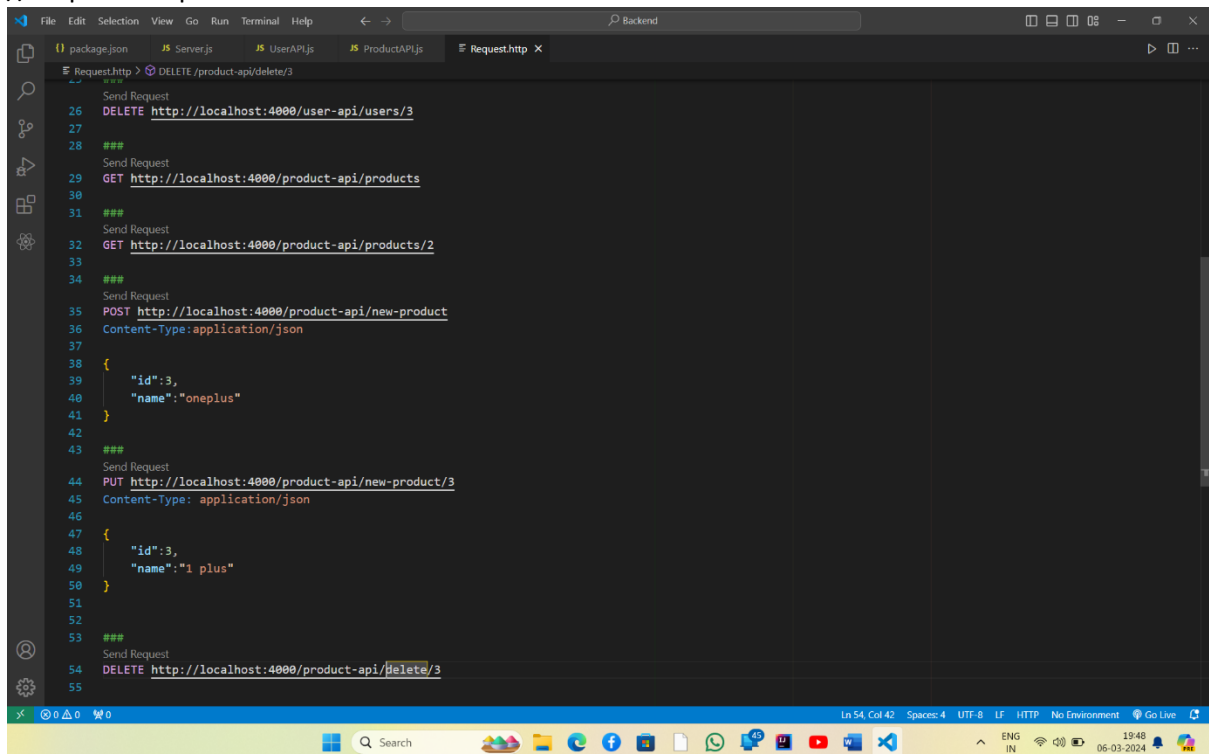
//Product-API

```
File Edit Selection View Go Run Terminal Help Backend
package.json JS Server.js JS UserAPI.js JS ProductAPI.js Request.http
API > JS ProductAPI.js > products
1
2 // productApi data
3
4 const exp = require('express');
5 let productapp = exp.Router();
6 let products = [
7   {
8     id: 1,
9     name: "samsung"
10  },
11  {
12    id: 2,
13    name: "realme"
14  }
15 ]
16
17 productapp.get('/products', (req, res) => {
18   res.send({products})
19 }
20 )
21
22 //getting product by id
23
24 productapp.get('/products/:id', (req, res) => {
25   let id = Number(req.params.id);
26   let product = products.find(productObj => productObj.id === id)
27   res.send({payload: product})
28 }
29 )
30
31 //adding new product
32
33 productapp.post('/new-product', (req, res) => {
34   let newProduct = req.body
35   products.push(newProduct)
36   res.send("product is added")
37 }
38 )
39
40 //updating product
41
```



```
34
35 //updating product
36
37 productapp.put('/new-product/:id',(req,res)=>{
38   let modifiedUser=req.body;
39   let index=products.findIndex(userObj=>userObj.id===modifiedUser.id)
40   if(index===-1){
41     res.send({message:"user not found"})
42   }
43   else
44   {
45     products.splice(index,1,modifiedUser)
46     res.send({message:"user data modified"})
47   }
48 })
49
50
51
52 //deleting product
53 productapp.delete('/delete/:id',(req,res)=>{
54   let id=Number(req.params.id)
55   let index=products.findIndex(productObj=>productObj.id===id)
56   if(index===-1){
57     res.send("product not found")
58   }
59   else{
60     products.splice(index,1)
61     res.send("product is deleted successfully")
62   }
63
64
65
66 })
67
68 module.exports=productapp;
```

//Requests.http



```
26 Send Request
27 DELETE http://localhost:4000/user-api/users/3
28 ###
29 Send Request
30 GET http://localhost:4000/product-api/products
31 ###
32 Send Request
33 GET http://localhost:4000/product-api/products/2
34 ###
35 Send Request
36 POST http://localhost:4000/product-api/new-product
37 Content-Type:application/json
38 {
39   "id":3,
40   "name":"oneplus"
41 }
42 ###
43 Send Request
44 PUT http://localhost:4000/product-api/new-product/3
45 Content-type: application/json
46 {
47   "id":3,
48   "name":"1 plus"
49 }
50 ###
51 Send Request
52 DELETE http://localhost:4000/product-api/delete/3
53
```

The screenshot shows the VS Code interface with a REST client file named 'Request.http'. The file contains several HTTP requests, with the last one being a DELETE request to 'http://localhost:4000/user-api/delete/3'. The response pane on the right shows the result of this request: a 200 OK status with headers including 'X-Powered-By: Express', 'Content-Type: application/json; charset=utf-8', and 'Content-Length: 63'. The response body is a JSON array of two user objects: [{id: 1, name: 'user1'}, {id: 2, name: 'user2'}].

```
1 DELETE http://localhost:4000/user-api/delete/3
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
```

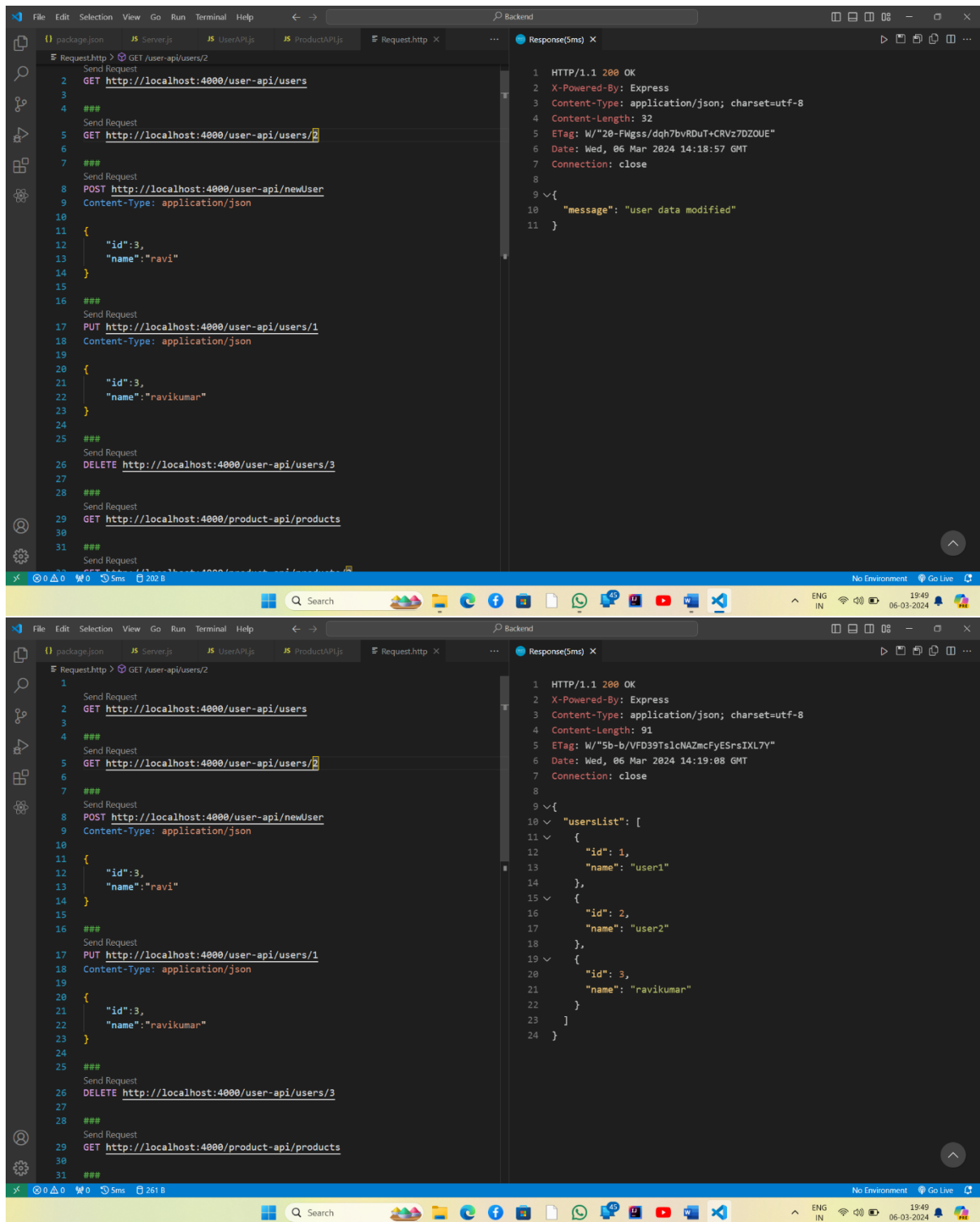
```
1 HTTP/1.1 200 OK
2 X-Powered-By: Express
3 Content-Type: application/json; charset=utf-8
4 Content-Length: 63
5 ETag: W/"3f-Gi20QQ+ieKgbzgyTFBn3I+GRpI"
6 Date: Wed, 06 Mar 2024 14:18:18 GMT
7 Connection: close
8
9 {
10   "usersList": [
11     {
12       "id": 1,
13       "name": "user1"
14     },
15     {
16       "id": 2,
17       "name": "user2"
18     }
19   ]
20 }
```

//CRUD operations using User-API

The screenshot shows the VS Code interface with a REST client file named 'Request.http'. The file contains several HTTP requests, with the last one being a POST request to 'http://localhost:4000/user-api/newUser'. The response pane on the right shows the result of this request: a 200 OK status with headers including 'X-Powered-By: Express', 'Content-Type: application/json; charset=utf-8', and 'Content-Length: 31'. The response body is a JSON object with a 'message' property: {message: 'new user is added'}.

```
1 GET http://localhost:4000/user-api/users/2
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
```

```
1 HTTP/1.1 200 OK
2 X-Powered-By: Express
3 Content-Type: application/json; charset=utf-8
4 Content-Length: 31
5 ETag: W/"1f-Ovy+drTW/m2vnjxiCaFYkCJC1Ys"
6 Date: Wed, 06 Mar 2024 14:18:37 GMT
7 Connection: close
8
9 {
10   "message": "new user is added"
11 }
```



CRUD operations using Product-API

The image displays two screenshots of a Visual Studio Code editor with the REST client extension, showing HTTP requests and responses for a Product-API. The background is a light gray, and the editor has a dark theme.

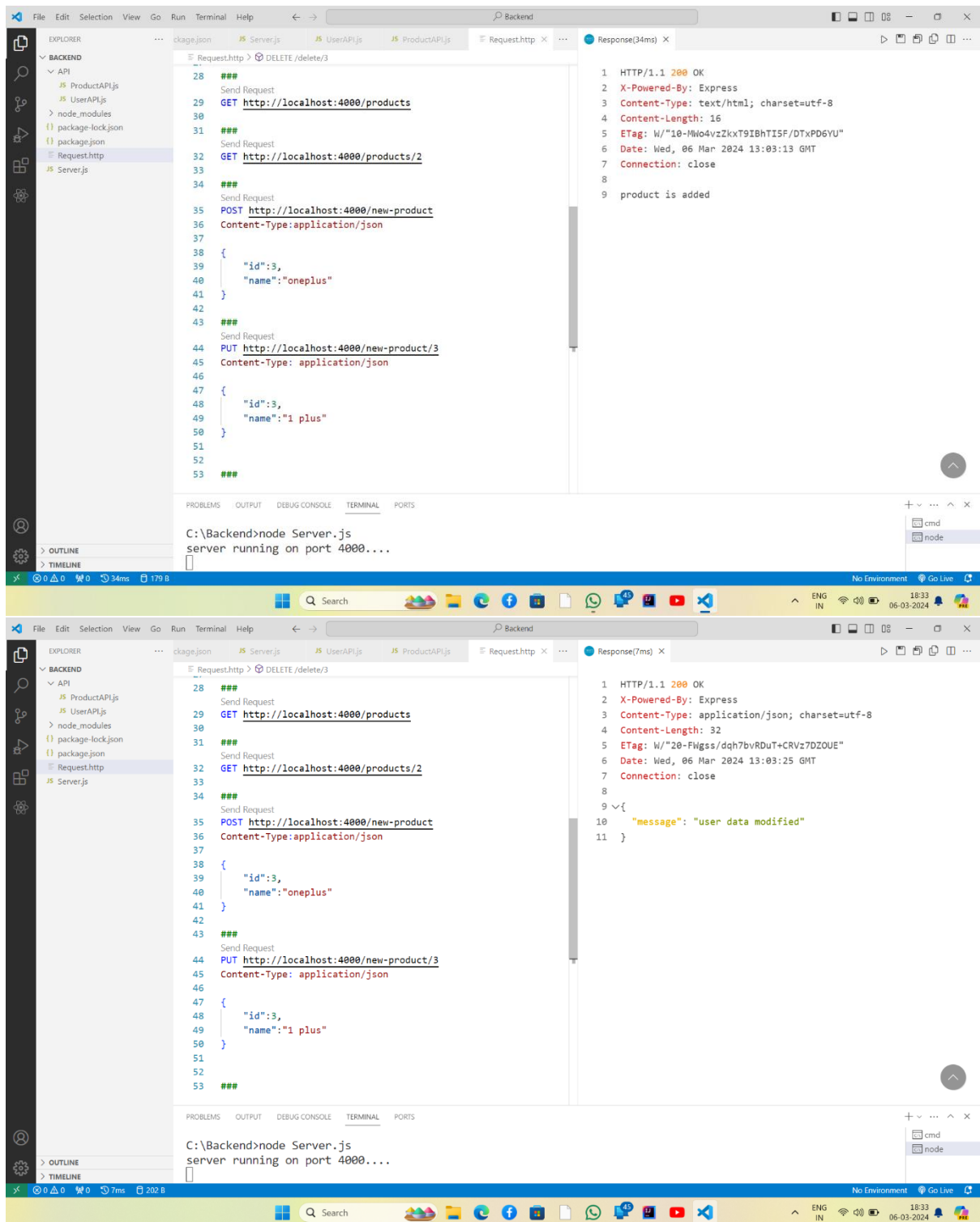
Top Screenshot:

- Request:** GET `http://localhost:4000/products/2`
- Response:** HTTP/1.1 200 OK, Content-Type: application/json; charset=utf-8, Content-Length: 65. The response body is a JSON array of two objects: `{ "id": 1, "name": "samsung" }, { "id": 2, "name": "realme" }`.

Bottom Screenshot:

- Request:** PUT `http://localhost:4000/new-product/3` with Content-Type: application/json. The request body is `{ "id": 3, "name": "1 plus" }`.
- Response:** HTTP/1.1 200 OK, Content-Type: application/json; charset=utf-8, Content-Length: 36. The response body is a JSON object: `{ "id": 2, "name": "realme" }`.

Both screenshots show the terminal output at the bottom: `C:\Backend>node Server.js` and `server running on port 4000....`



Visual Studio Code interface showing a REST client request and response for a DELETE operation.

Request:

```
29 GET http://localhost:4000/products
30
31 ###
32 Send Request
33 GET http://localhost:4000/products/2
34
35 ###
36 Send Request
37 POST http://localhost:4000/new-product
38 Content-Type: application/json
39 {
40   "id": 3,
41   "name": "oneplus"
42 }
43
44 ###
45 Send Request
46 PUT http://localhost:4000/new-product/3
47 Content-Type: application/json
48 {
49   "id": 3,
50   "name": "1 plus"
51 }
52
53 ###
54 Send Request
55 DELETE http://localhost:4000/delete/3
```

Response (6ms):

```
1 HTTP/1.1 200 OK
2 X-Powered-By: Express
3 Content-Type: text/html; charset=utf-8
4 Content-Length: 31
5 ETag: W/"1f-CvnFLk8D6tEAzp1QLijAMdXrRow"
6 Date: Wed, 06 Mar 2024 13:03:38 GMT
7 Connection: close
8
9 product is deleted successfully
```

Terminal:

```
C:\Backend>node Server.js
server running on port 4000....
```

Visual Studio Code Interface:

- Explorer: Backend, API, ProductAPI.js, UserAPI.js, node_modules, package-lock.json, package.json, Request.http, Server.js
- Request.http: DELETE /delete/3
- Response (6ms): HTTP/1.1 200 OK, X-Powered-By: Express, Content-Type: text/html; charset=utf-8, Content-Length: 31, ETag: W/"1f-CvnFLk8D6tEAzp1QLijAMdXrRow", Date: Wed, 06 Mar 2024 13:03:38 GMT, Connection: close, product is deleted successfully
- Terminal: C:\Backend>node Server.js, server running on port 4000....

Visual Studio Code Interface (Second Screenshot):

- Explorer: Backend, API, ProductAPI.js, UserAPI.js, node_modules, package-lock.json, package.json, Request.http, Server.js
- Request.http: DELETE /delete/3, GET /products, GET /products/2, POST /new-product, PUT /new-product/3
- Response (11ms): HTTP/1.1 200 OK, X-Powered-By: Express, Content-Type: application/json; charset=utf-8, Content-Length: 65, ETag: W/"41-NcTbqh/WY6vvOfcvLfVhJ2Vqe+k", Date: Wed, 06 Mar 2024 13:03:51 GMT, Connection: close, { "products": [{ "id": 1, "name": "samsung" }, { "id": 2, "name": "realme" }] }
- Terminal: C:\Backend>node Server.js, server running on port 4000....