INTERNSHALA

ASSIGNMENT-1

Build a RESTful API using Node.js and Express

Objective of the task:

Create a simple RESTful API for managing a list of users, testing concepts such as routing, middleware, HTTP methods, status codes, error handling, and interaction with a data source.

Initialize Node js project:

Install Node js in your system

↓

Ensure installed successfully using node -v

↓

Create new folder and cd… into project folder

↓

Initialize using npm init -y

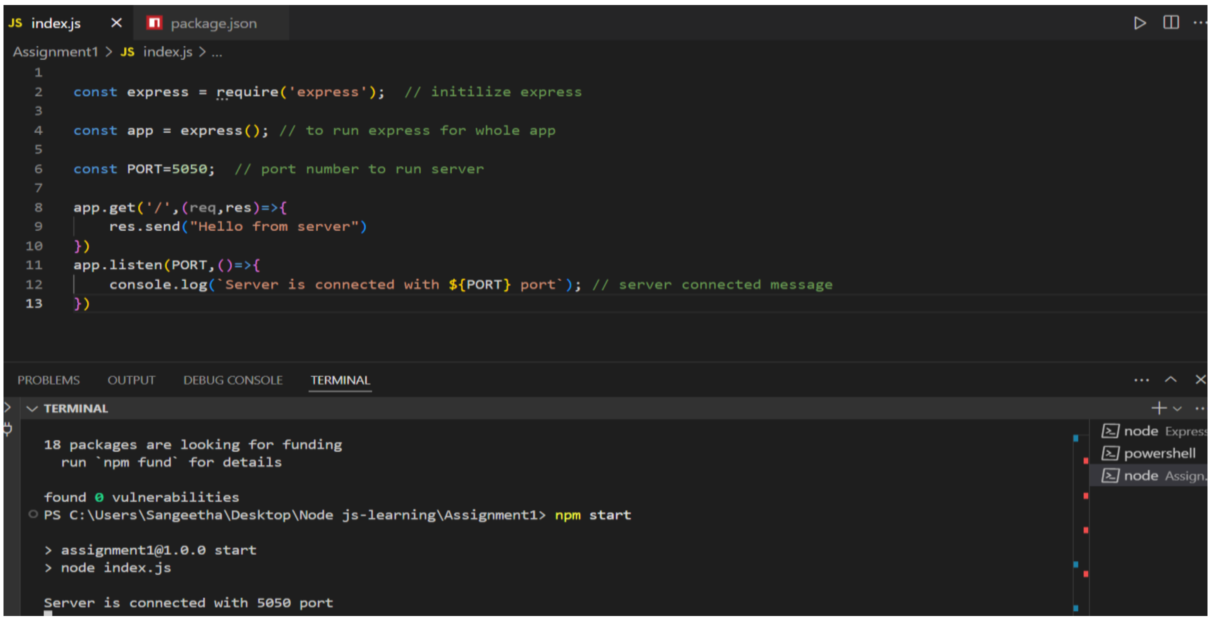
↓

Install express using npm install express

↓ Automatically save your changes

Npm install nodemon tool and start to code

Set-Up Node js Project



index.js | package.json

Assignment1 > index.js > ...

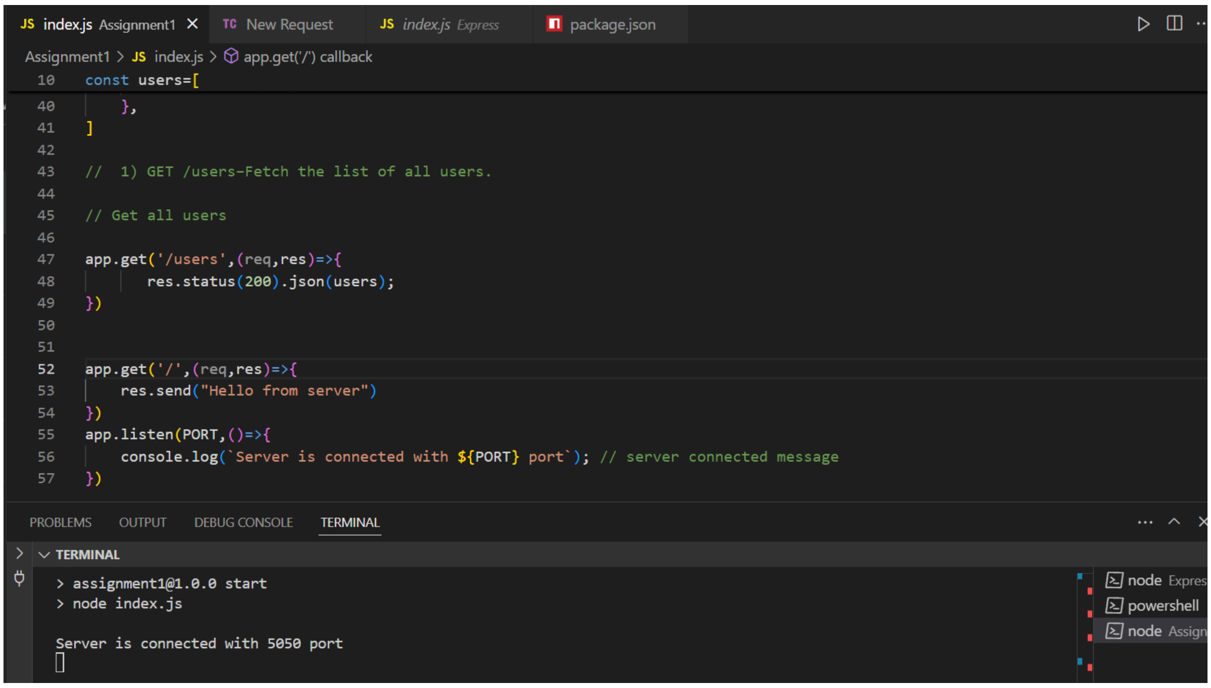
```javascript

const express = require('express');  // initilize express

const app = express(); // to run express for whole app

const PORT=5050;  // port number to run server

app.get('/',(req,res)=>{
    res.send("Hello from server")
})
app.listen(PORT,()=>{
    console.log(`Server is connected with ${PORT} port`); // server connected message
})
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

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

found 0 vulnerabilities
PS C:\Users\Sangeetha\Desktop\Node js-learning\Assignment1> npm start

> assignment1@1.0.0 start
> node index.js

Server is connected with 5050 port
```

Task 1

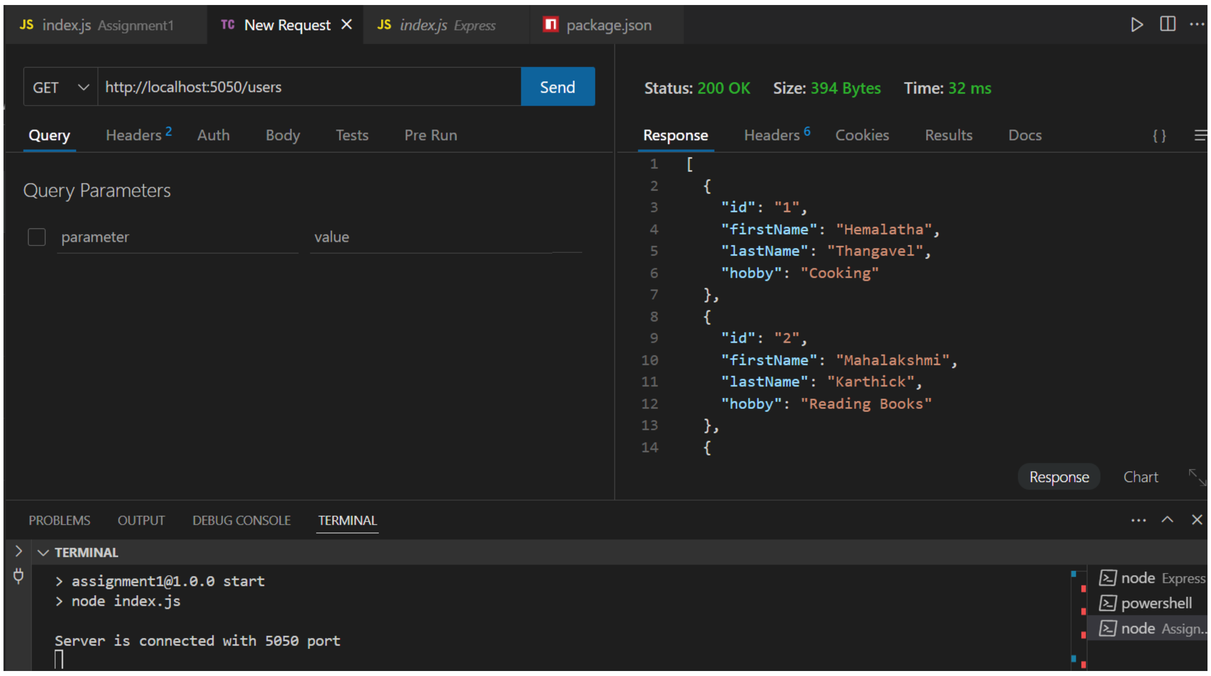
GET /users Fetch the list of all users



```
const users=[
    },
]

//  1) GET /users-Fetch the list of all users.

// Get all users

app.get('/users',(req,res)=>{
        res.status(200).json(users);
})


app.get('/',(req,res)=>{
    res.send("Hello from server")
})
app.listen(PORT,()=>{
    console.log(`Server is connected with ${PORT} port`); // server connected message
})
```

```
> assignment1@1.0.0 start
> node index.js

Server is connected with 5050 port
```

Test code using Thunder- Client



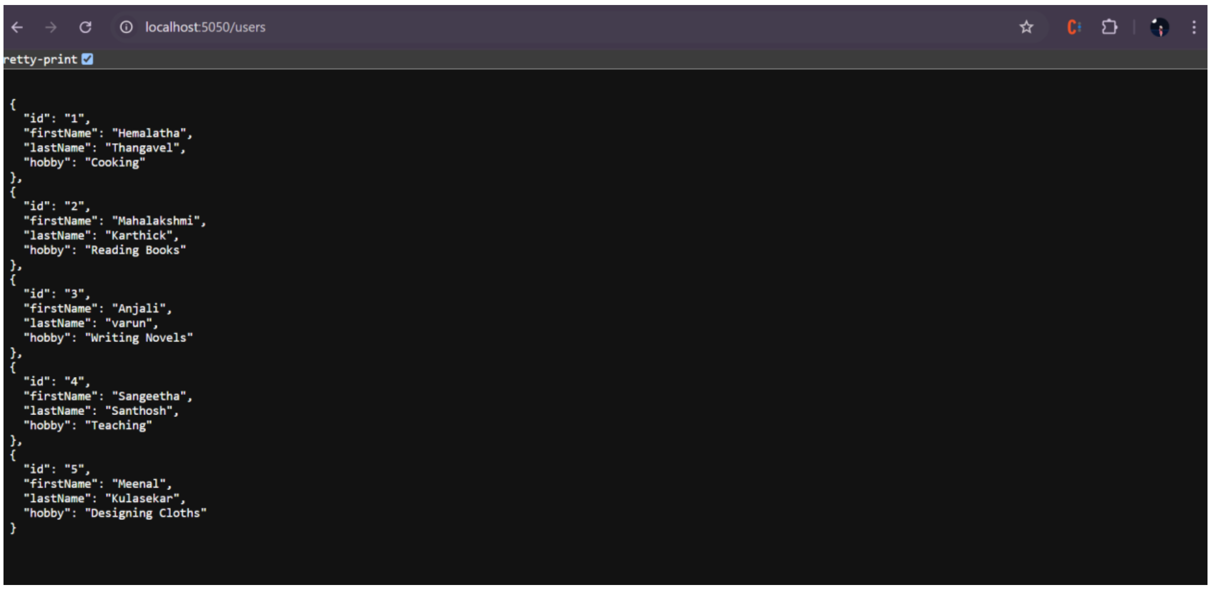
GET http://localhost:5050/users

Status: 200 OK   Size: 394 Bytes   Time: 32 ms

```
[
  {
    "id": "1",
    "firstName": "Hemalatha",
    "lastName": "Thangavel",
    "hobby": "Cooking"
  },
  {
    "id": "2",
    "firstName": "Mahalakshmi",
    "lastName": "Karthick",
    "hobby": "Reading Books"
  },
  {
```

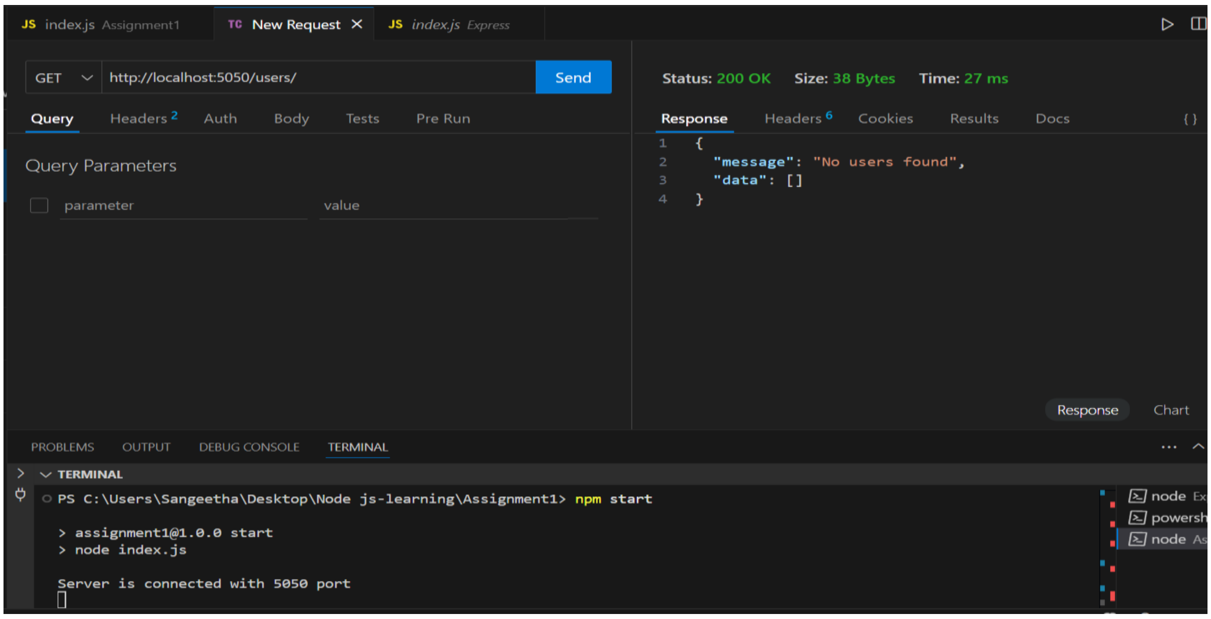
```
> assignment1@1.0.0 start
> node index.js

Server is connected with 5050 port
```

API view in browser



```
{
  "id": "1",
  "firstName": "Hemalatha",
  "lastName": "Thangavel",
  "hobby": "Cooking"
},
{
  "id": "2",
  "firstName": "Mahalakshmi",
  "lastName": "Karthick",
  "hobby": "Reading Books"
},
{
  "id": "3",
  "firstName": "Anjali",
  "lastName": "varun",
  "hobby": "Writing Novels"
},
{
  "id": "4",
  "firstName": "Sangeetha",
  "lastName": "Santhosh",
  "hobby": "Teaching"
},
{
  "id": "5",
  "firstName": "Meenal",
  "lastName": "Kulasekar",
  "hobby": "Designing Cloths"
}
```

Error : If the data array is empty – display error message

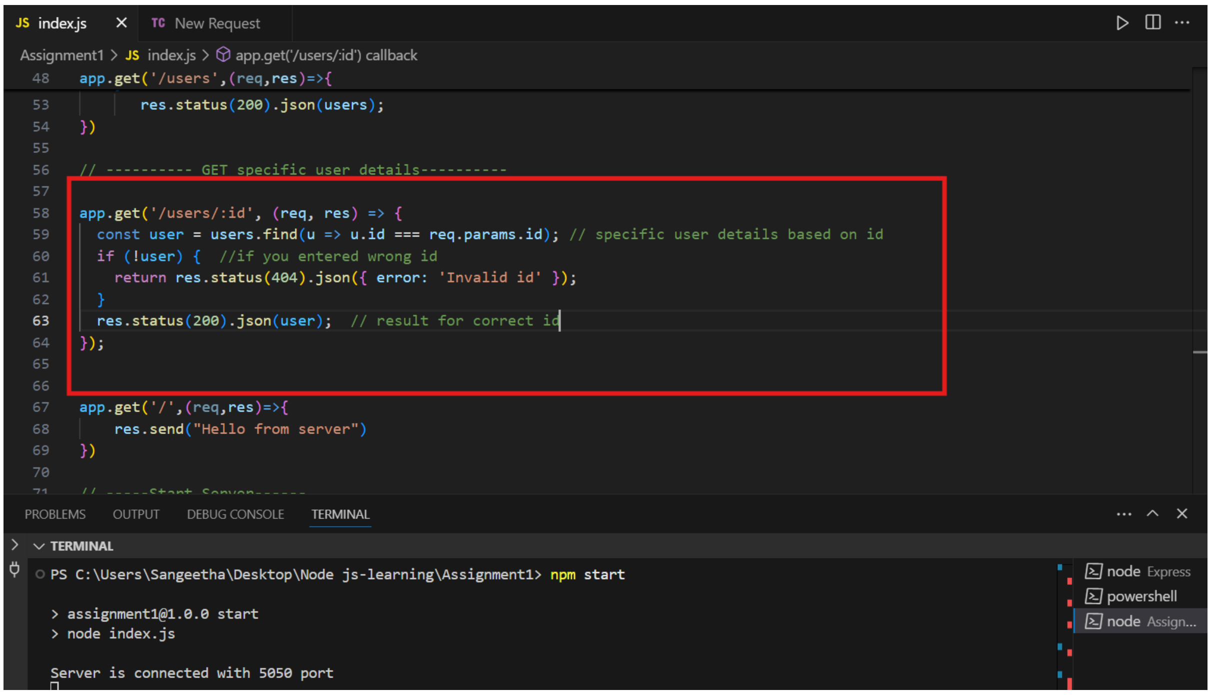


```
{
  "message": "No users found",
  "data": []
}
```

```
PS C:\Users\Sangeetha\Desktop\Node js-learning\Assignment1> npm start

> assignment1@1.0.0 start
> node index.js

Server is connected with 5050 port
```

Task – 2

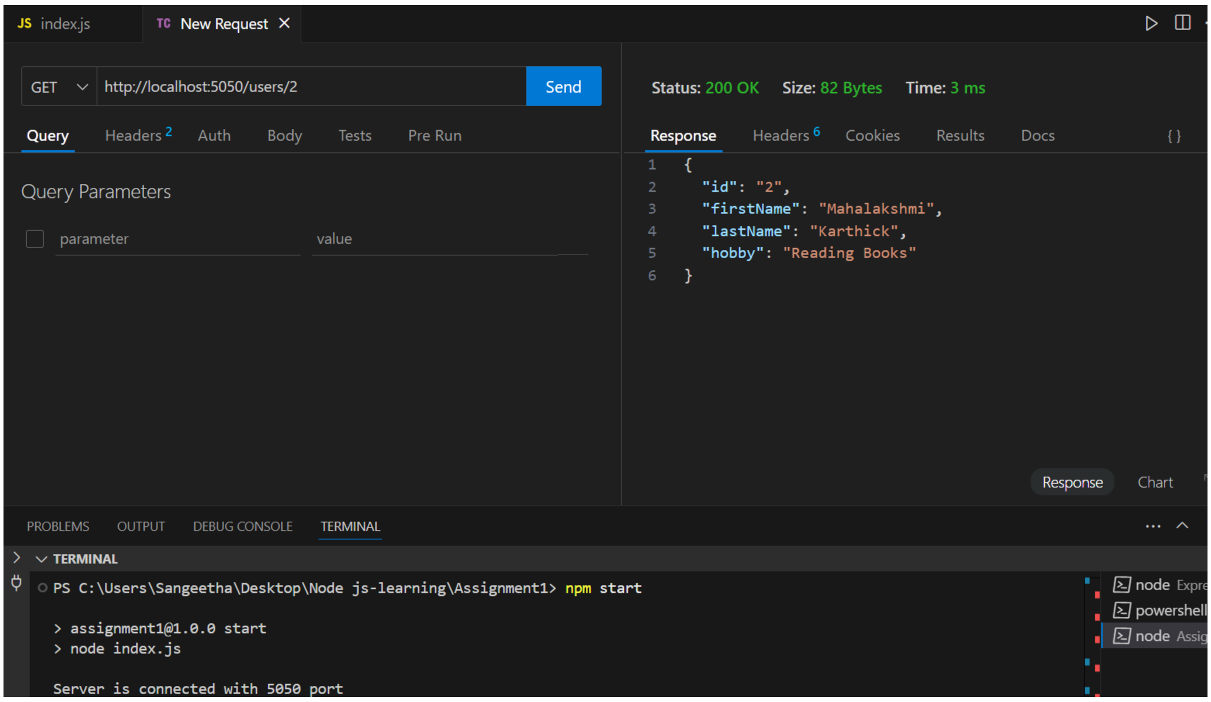
GET /users/:id–Fetch details of a specific user by ID



```
app.get('/users',(req,res)=>{
        res.status(200).json(users);
})

// --------- GET specific user details---------

app.get('/users/:id', (req, res) => {
  const user = users.find(u => u.id === req.params.id); // specific user details based on id
  if (!user) {  //if you entered wrong id
    return res.status(404).json({ error: 'Invalid id' });
  }
  res.status(200).json(user);  // result for correct id
});


app.get('/',(req,res)=>{
    res.send("Hello from server")
})
```

```
PS C:\Users\Sangeetha\Desktop\Node js-learning\Assignment1> npm start

> assignment1@1.0.0 start
> node index.js

Server is connected with 5050 port
```

Result in Thunder-client



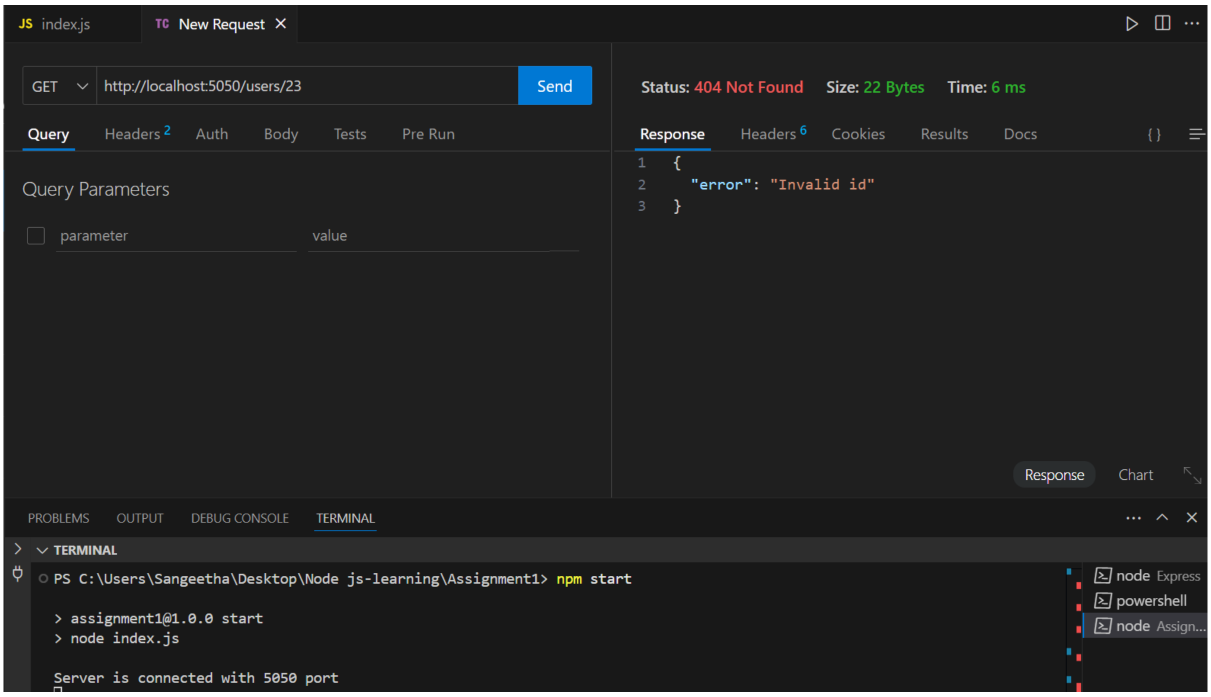
http://localhost:5050/users/2

Status: 200 OK   Size: 82 Bytes   Time: 3 ms

```
{
  "id": "2",
  "firstName": "Mahalakshmi",
  "lastName": "Karthick",
  "hobby": "Reading Books"
}
```

```
PS C:\Users\Sangeetha\Desktop\Node js-learning\Assignment1> npm start

> assignment1@1.0.0 start
> node index.js

Server is connected with 5050 port
```

Error message – If you entered invalid user id



GET http://localhost:5050/users/23 Send

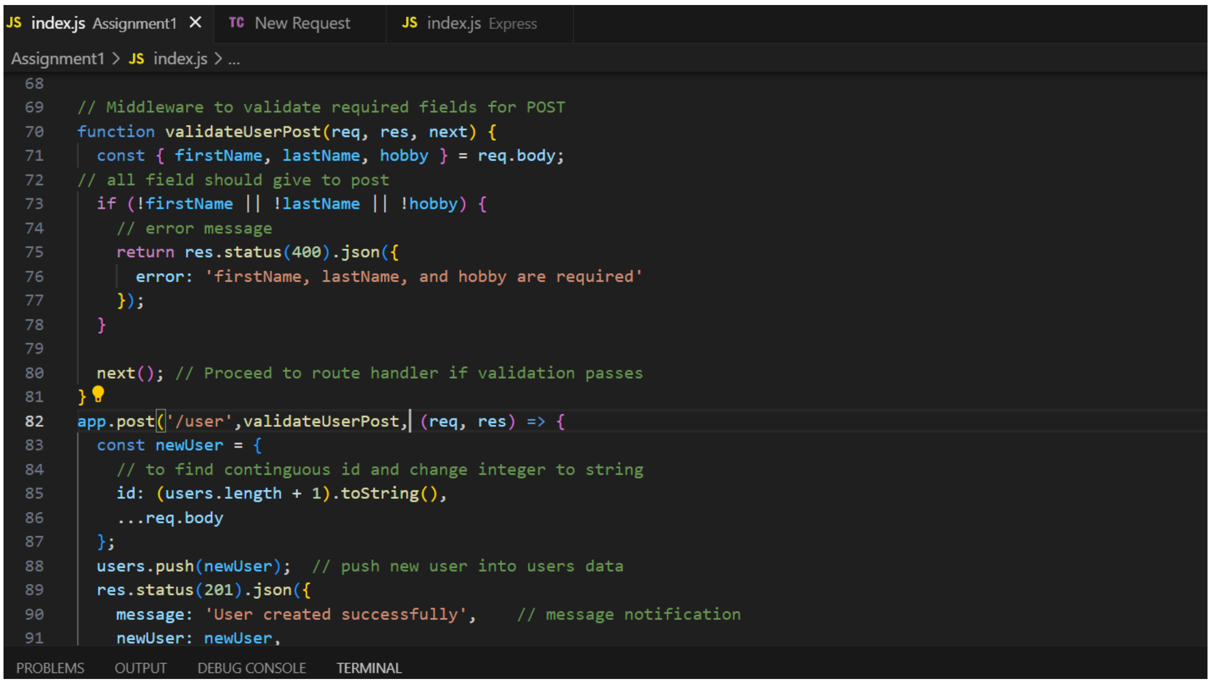
Status: 404 Not Found Size: 22 Bytes Time: 6 ms

```
{
  "error": "Invalid id"
}
```

```
PS C:\Users\Sangeetha\Desktop\Node js-learning\Assignment1> npm start

> assignment1@1.0.0 start
> node index.js

Server is connected with 5050 port
```

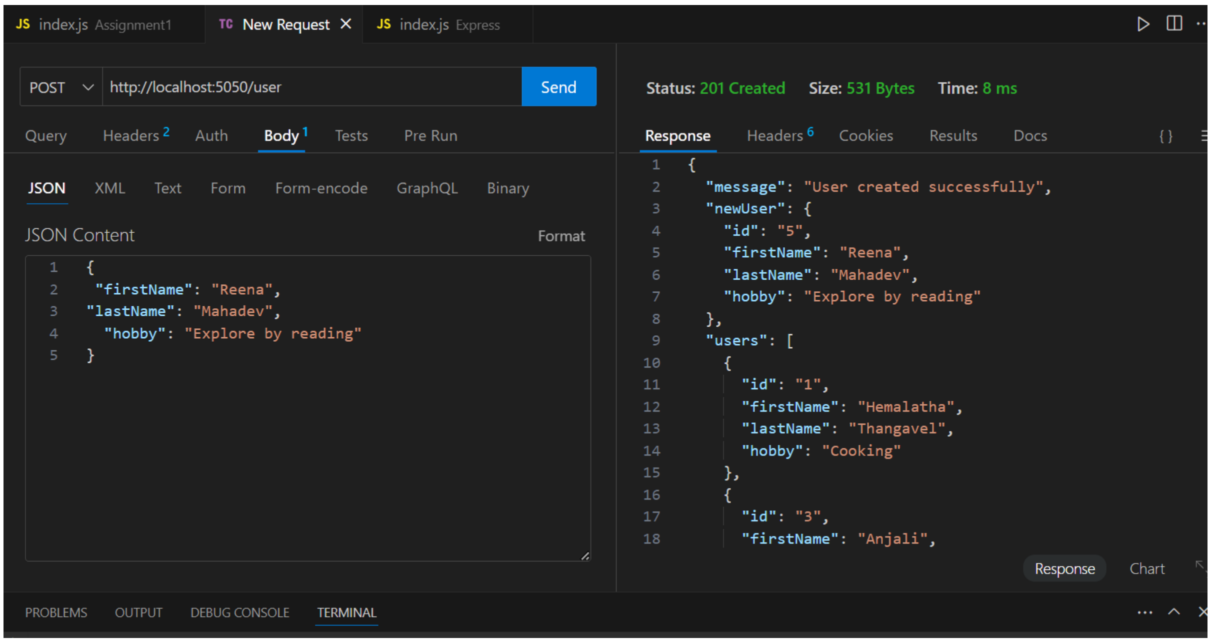
Task – 3

POST /user–Add a new user



```javascript

// Middleware to validate required fields for POST
function validateUserPost(req, res, next) {
  const { firstName, lastName, hobby } = req.body;
// all field should give to post
  if (!firstName || !lastName || !hobby) {
    // error message
    return res.status(400).json({
      error: 'firstName, lastName, and hobby are required'
    });
  }

  next(); // Proceed to route handler if validation passes
}
app.post('/user',validateUserPost, (req, res) => {
  const newUser = {
    // to find continguous id and change integer to string
    id: (users.length + 1).toString(),
    ...req.body
  };
  users.push(newUser);  // push new user into users data
  res.status(201).json({
    message: 'User created successfully',    // message notification
    newUser: newUser,
```

Response in Thunder-client



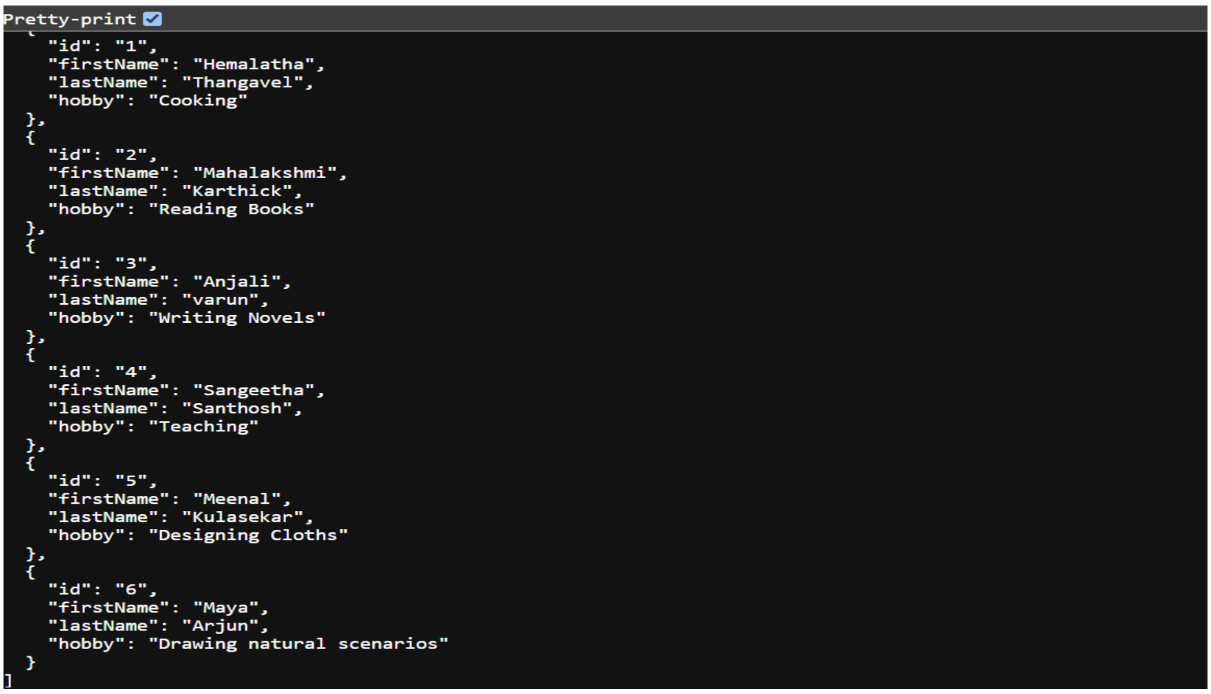
POST http://localhost:5050/user

```json
{
  "firstName": "Reena",
"lastName": "Mahadev",
    "hobby": "Explore by reading"
}
```

Status: 201 Created   Size: 531 Bytes   Time: 8 ms

```json
{
  "message": "User created successfully",
  "newUser": {
    "id": "5",
    "firstName": "Reena",
    "lastName": "Mahadev",
    "hobby": "Explore by reading"
  },
  "users": [
    {
      "id": "1",
      "firstName": "Hemalatha",
      "lastName": "Thangavel",
      "hobby": "Cooking"
    },
    {
      "id": "3",
      "firstName": "Anjali",
```

Result in Browser

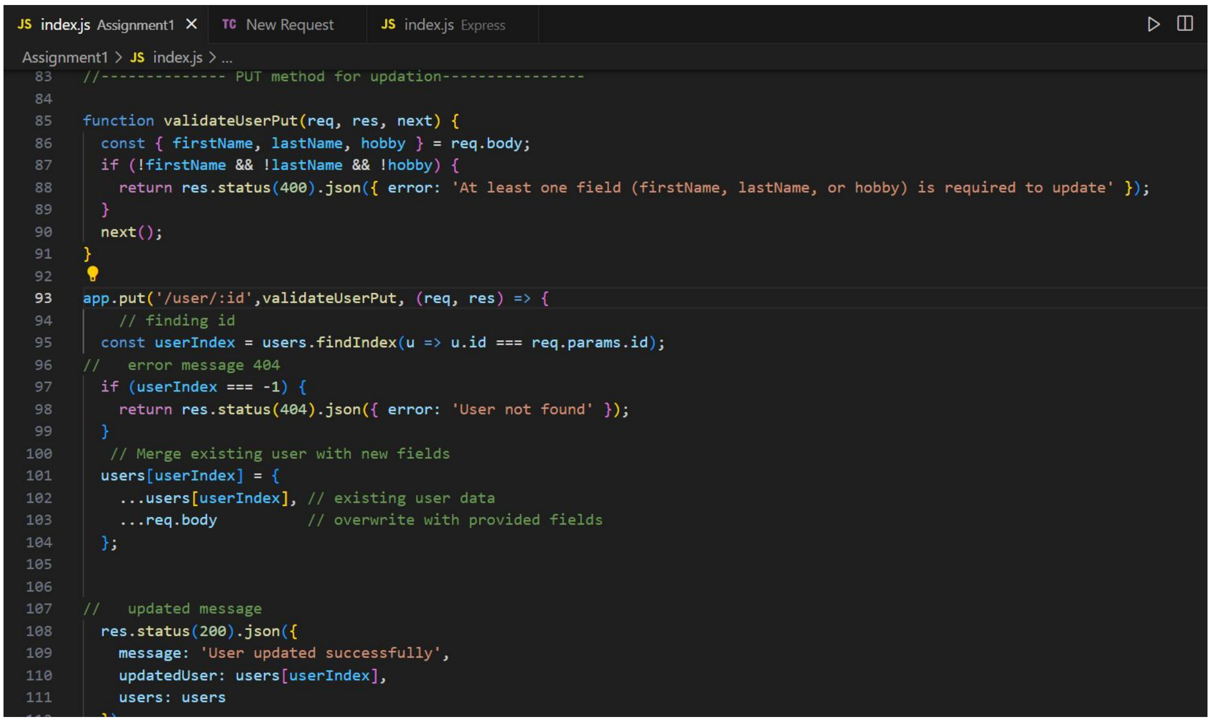


Pretty-print ✓

```
    "id": "1",
    "firstName": "Hemalatha",
    "lastName": "Thangavel",
    "hobby": "Cooking"
  },
  {
    "id": "2",
    "firstName": "Mahalakshmi",
    "lastName": "Karthick",
    "hobby": "Reading Books"
  },
  {
    "id": "3",
    "firstName": "Anjali",
    "lastName": "varun",
    "hobby": "Writing Novels"
  },
  {
    "id": "4",
    "firstName": "Sangeetha",
    "lastName": "Santhosh",
    "hobby": "Teaching"
  },
  {
    "id": "5",
    "firstName": "Meenal",
    "lastName": "Kulasekar",
    "hobby": "Designing Cloths"
  },
  {
    "id": "6",
    "firstName": "Maya",
    "lastName": "Arjun",
    "hobby": "Drawing natural scenarios"
  }
]
```

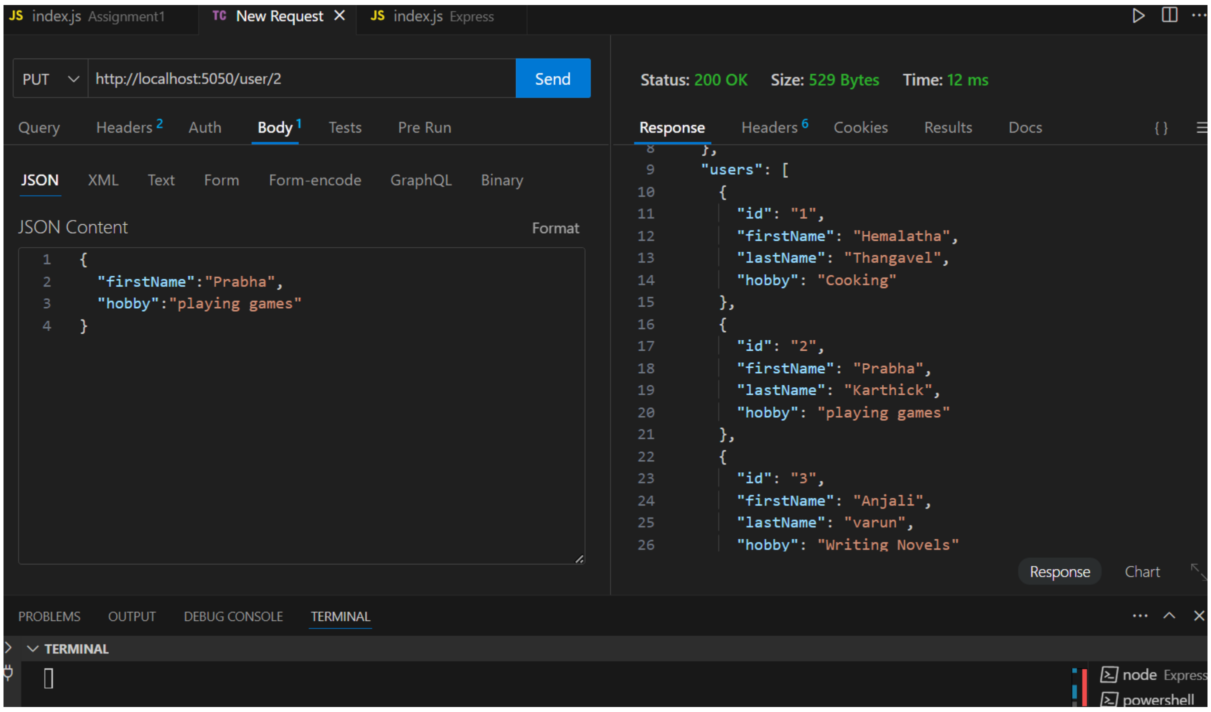
Task – 4

PUT /user/: id–Update details of an existing user



```
//------------- PUT method for updation---------------

function validateUserPut(req, res, next) {
  const { firstName, lastName, hobby } = req.body;
  if (!firstName && !lastName && !hobby) {
    return res.status(400).json({ error: 'At least one field (firstName, lastName, or hobby) is required to update' });
  }
  next();
}

app.put('/user/:id',validateUserPut, (req, res) => {
    // finding id
  const userIndex = users.findIndex(u => u.id === req.params.id);
//   error message 404
  if (userIndex === -1) {
    return res.status(404).json({ error: 'User not found' });
  }
   // Merge existing user with new fields
  users[userIndex] = {
    ...users[userIndex], // existing user data
    ...req.body          // overwrite with provided fields
  };


//   updated message
  res.status(200).json({
    message: 'User updated successfully',
    updatedUser: users[userIndex],
    users: users
```

Response in Thunder-client



```
PUT  http://localhost:5050/user/2
```

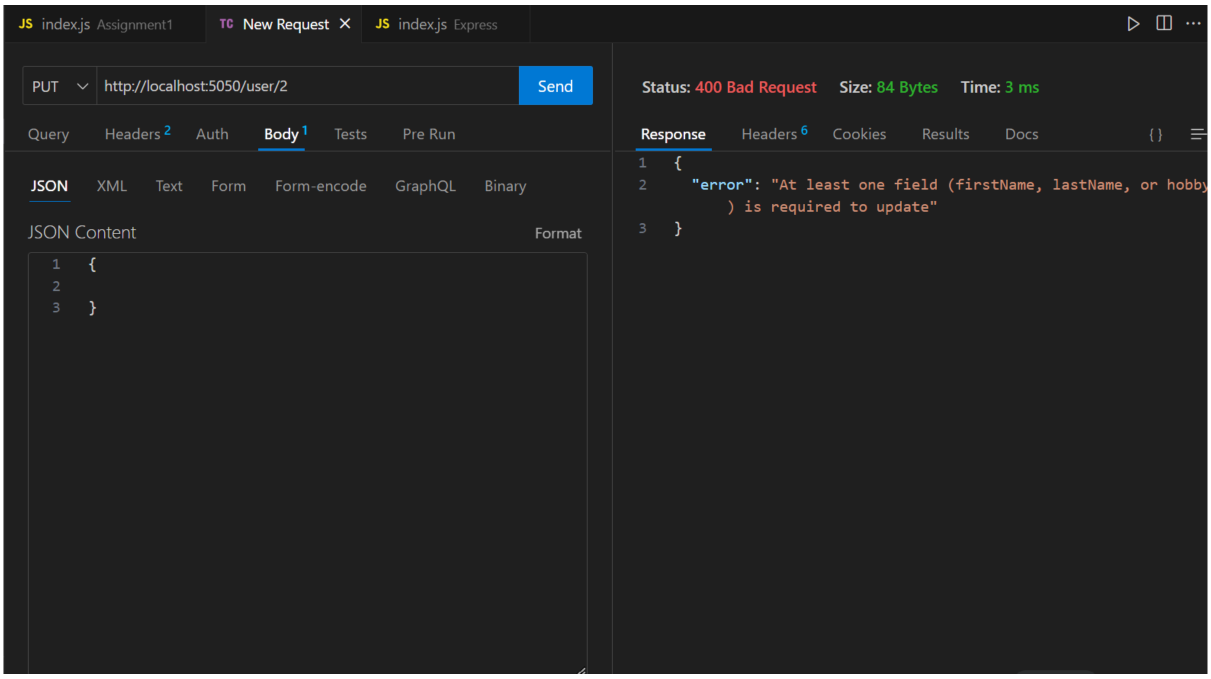
```
{
  "firstName":"Prabha",
  "hobby":"playing games"
}
```

Status: 200 OK   Size: 529 Bytes   Time: 12 ms

```
  "users": [
    {
      "id": "1",
      "firstName": "Hemalatha",
      "lastName": "Thangavel",
      "hobby": "Cooking"
    },
    {
      "id": "2",
      "firstName": "Prabha",
      "lastName": "Karthick",
      "hobby": "playing games"
    },
    {
      "id": "3",
      "firstName": "Anjali",
      "lastName": "varun",
      "hobby": "Writing Novels"
```

Error message

If no field is entered and try to make PUT request


PUT http://localhost:5050/user/2

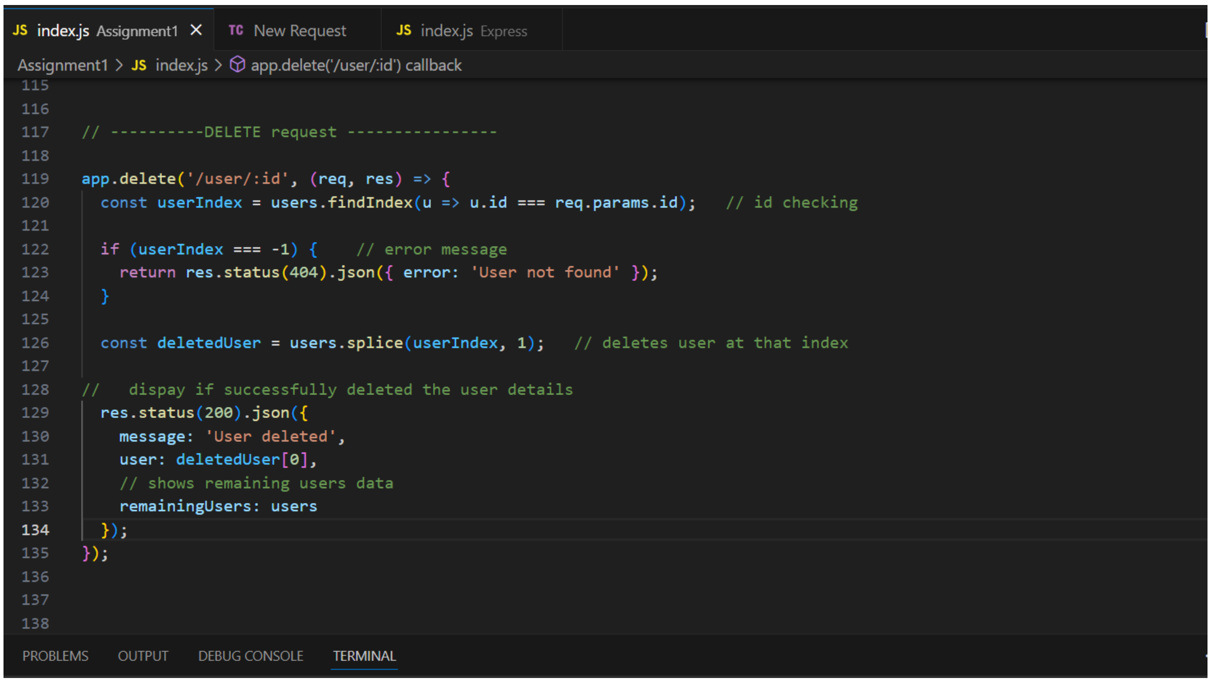
```
{

}
```

Status: 400 Bad Request   Size: 84 Bytes   Time: 3 ms

```
{
  "error": "At least one field (firstName, lastName, or hobby
    ) is required to update"
}
```

Task – 5

DELETE /user/:id–Delete a user by ID

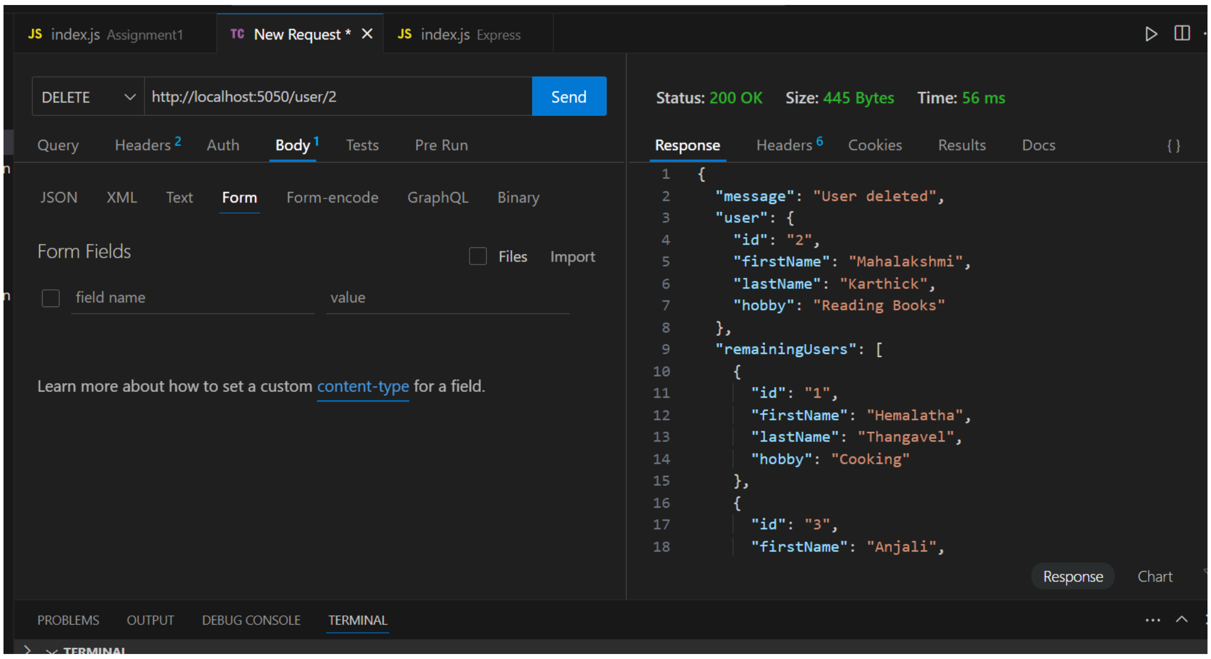


index.js Assignment1 · New Request · index.js Express

Assignment1 > index.js > app.delete('/user/:id') callback

```

// ----------DELETE request ---------------

app.delete('/user/:id', (req, res) => {
  const userIndex = users.findIndex(u => u.id === req.params.id);   // id checking

  if (userIndex === -1) {    // error message
    return res.status(404).json({ error: 'User not found' });
  }

  const deletedUser = users.splice(userIndex, 1);   // deletes user at that index

//   dispay if successfully deleted the user details
  res.status(200).json({
    message: 'User deleted',
    user: deletedUser[0],
    // shows remaining users data
    remainingUsers: users
  });
});


```

Response in Thunder-client



DELETE http://localhost:5050/user/2 Send

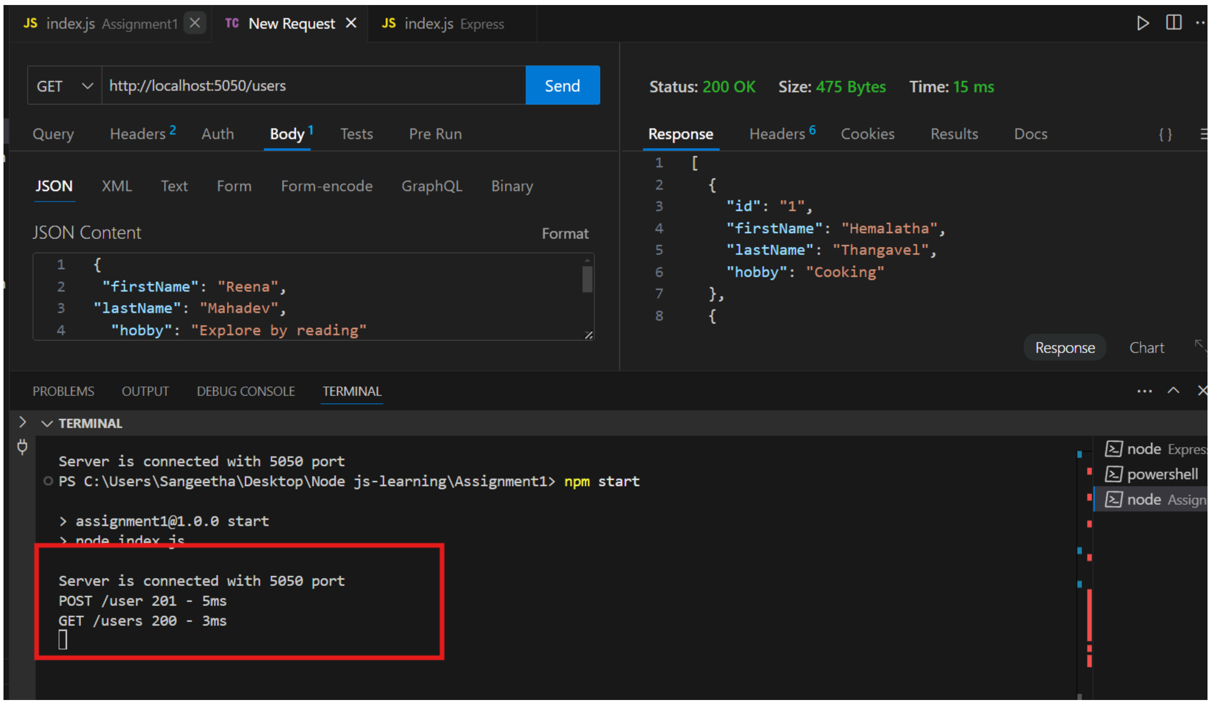
Status: 200 OK Size: 445 Bytes Time: 56 ms

Body: Form — Form Fields: field name / value

Learn more about how to set a custom content-type for a field.

```
{
  "message": "User deleted",
  "user": {
    "id": "2",
    "firstName": "Mahalakshmi",
    "lastName": "Karthick",
    "hobby": "Reading Books"
  },
  "remainingUsers": [
    {
      "id": "1",
      "firstName": "Hemalatha",
      "lastName": "Thangavel",
      "hobby": "Cooking"
    },
    {
      "id": "3",
      "firstName": "Anjali",
```

Use middleware to log the details of each request



```
{
 "firstName": "Reena",
"lastName": "Mahadev",
  "hobby": "Explore by reading"
```

```
[
  {
    "id": "1",
    "firstName": "Hemalatha",
    "lastName": "Thangavel",
    "hobby": "Cooking"
  },
  {
```

```
Server is connected with 5050 port
PS C:\Users\Sangeetha\Desktop\Node js-learning\Assignment1> npm start

> assignment1@1.0.0 start
> node index.js

Server is connected with 5050 port
POST /user 201 - 5ms
GET /users 200 - 3ms
```

Conclusion

This RESTful API allows creating, reading, updating, and deleting user data using HTTP methods.

- GET -- retrieves all users or a specific user by ID.
- POST-- adds a new user with validation.
- PUT -- updates user details partially or fully.
- DELETE-- removes a user by ID and returns the updated list.

GitHub Link

https://github.com/Hema2802/create-RESTfull-API