Enhanced Functionality of Microservice

Github Link:

https://github.com/31Husain31/SIT323-2025-prac4c

Process followed for building the calculator microservice:

Step 1: Project Setup

1. Created a new project folder named 4.1C and duplicated 4.1P files for additional operations.
2. Opened this folder using Visual Studio Code (VS Code).
3. Added endpoints clearly following consistent URL patterns previously established: /power, /sqrt, and /modulo.

Step 2: Designing API Endpoints

Planned and created four main REST API endpoints clearly for the calculator functionality:

Endpoint	Method	Operation	URL Example
/add	GET	Add num1 and num2	/add?num1=5&num2=10
/subtract	GET	Subtract num1 and num2	/subtract?num1=5&num2=10
/multiply	GET	Multiply num1 and num2	/multiply?num1=5&num2=10
/divide	GET	Divide num1 and num2	/divide?num1=10&num2=2
Exponentiation	GET	num1 raised to num2	/power?num1=2&num2=3 → returns 8
Square root	GET	num	/sqrt?num=9 → returns 3
Modulo	GET	Remainder of num1 and num2	/modulo?num1=10&num2=3 → returns 1

Additionally, the helpful root endpoint (/) was updated to guide users clearly.

Step 3: Implementing the Microservice

Created a file called index.js in the project root. Coded the endpoints using Express:

- Parsed inputs (num1, num2) from the URL query parameters using parseFloat().

- Added error handling to check if inputs are valid numbers (isNaN()) and to prevent division by zero.
- For square root, checked if input is negative (since negative numbers don't produce real square roots).
- For modulo, ensured the second parameter is never zero (avoiding mathematical errors).

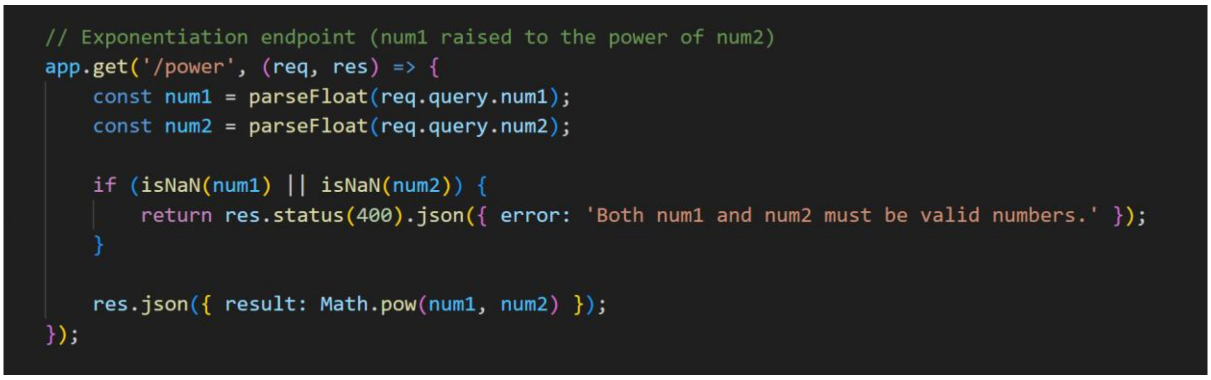
Step 4: Testing the API Endpoints

- Started the microservice locally using: node index.js
- Clearly tested each endpoint via browser URLs:
    - Addition: http://localhost:3000/add?num1=5&num2=10
    - Subtraction: http://localhost:3000/subtract?num1=10&num2=4
    - Multiplication: http://localhost:3000/multiply?num1=6&num2=7
    - Division: http://localhost:3000/divide?num1=14&num2=2
    - Exponent: http://localhost:3000/power?num1=2&num2=3
    - Square root: http://localhost:3000/sqrt?num=16
    - Modulo: http://localhost:3000/modulo?num1=10&num2=3
- Also verified error cases:
    - Non-numeric input (num1=abc)
    - Division by zero (num2=0)
    - Square root of negative number
    - Modulo by 0

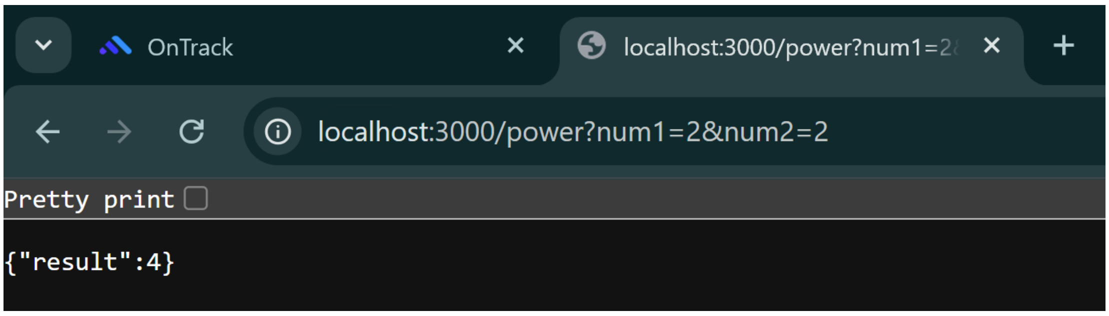
Step 5: Adding Code to GitHub

- Created a GitHub repository named SIT323-2025-prac4c.
- Initialized git locally, committed files, and pushed them to GitHub clearly using:
    - git init
    - git add .
    - git commit -m "Initial commit: Calculator microservice"
    - git branch -M main
    - git remote add origin https://github.com/31Husain31/SIT323-2025-prac4c.git
    - git push -u origin main

Some Screenshots of code and output

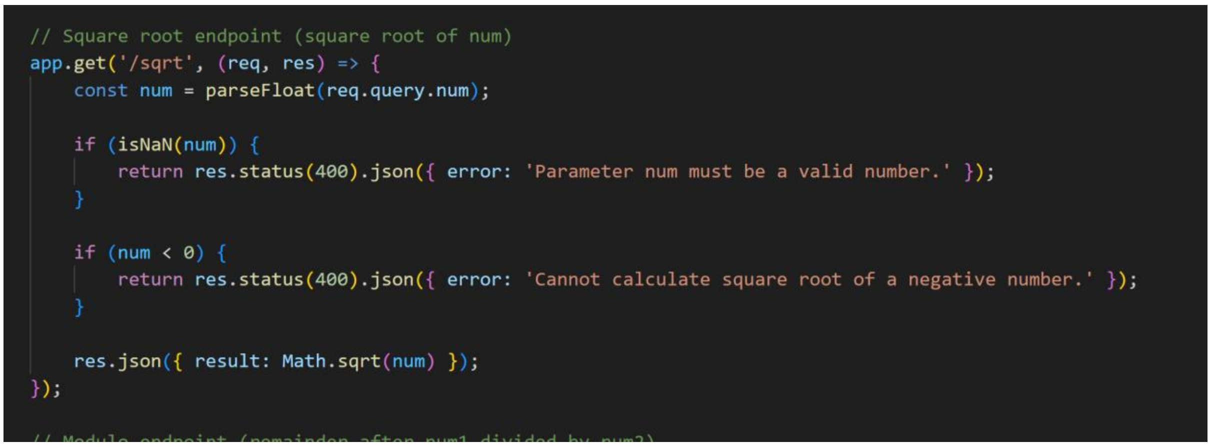


```javascript
// Exponentiation endpoint (num1 raised to the power of num2)
app.get('/power', (req, res) => {
    const num1 = parseFloat(req.query.num1);
    const num2 = parseFloat(req.query.num2);

    if (isNaN(num1) || isNaN(num2)) {
        return res.status(400).json({ error: 'Both num1 and num2 must be valid numbers.' });
    }

    res.json({ result: Math.pow(num1, num2) });
});
```

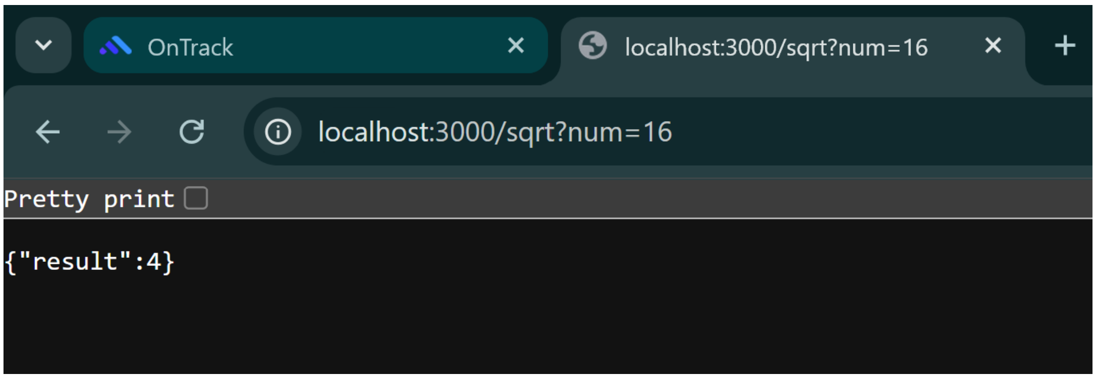


localhost:3000/power?num1=2&num2=2

```
{"result":4}
```

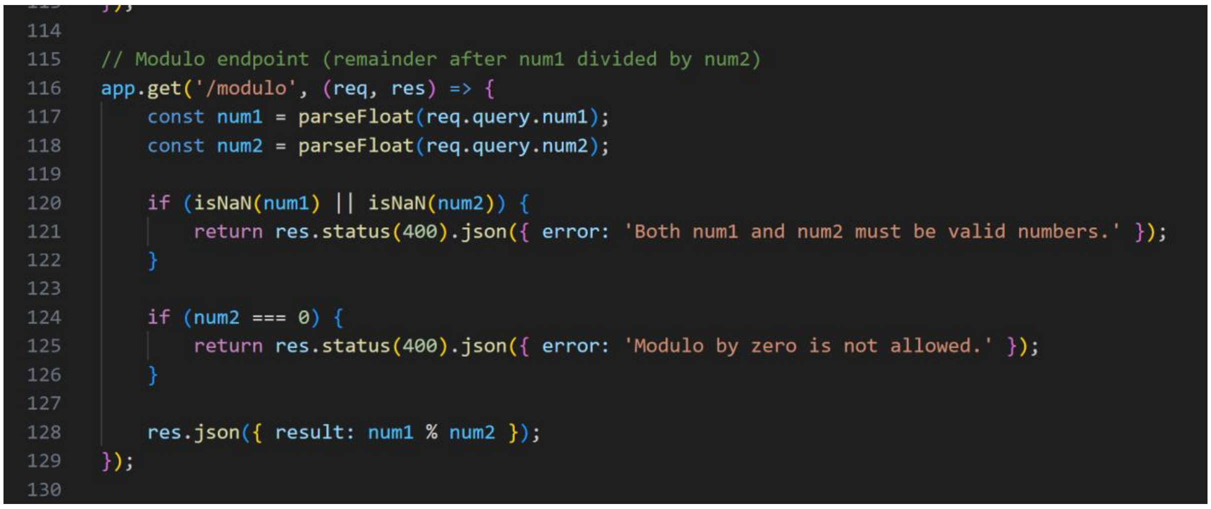


```javascript
// Square root endpoint (square root of num)
app.get('/sqrt', (req, res) => {
    const num = parseFloat(req.query.num);

    if (isNaN(num)) {
        return res.status(400).json({ error: 'Parameter num must be a valid number.' });
    }

    if (num < 0) {
        return res.status(400).json({ error: 'Cannot calculate square root of a negative number.' });
    }

    res.json({ result: Math.sqrt(num) });
});
// Modulo endpoint (remainder after num1 divided by num2)
```

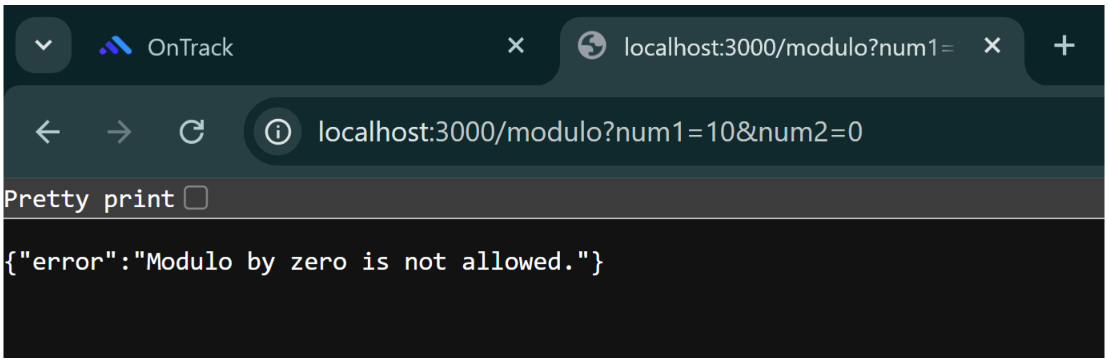


localhost:3000/sqrt?num=16

```
{"result":4}
```

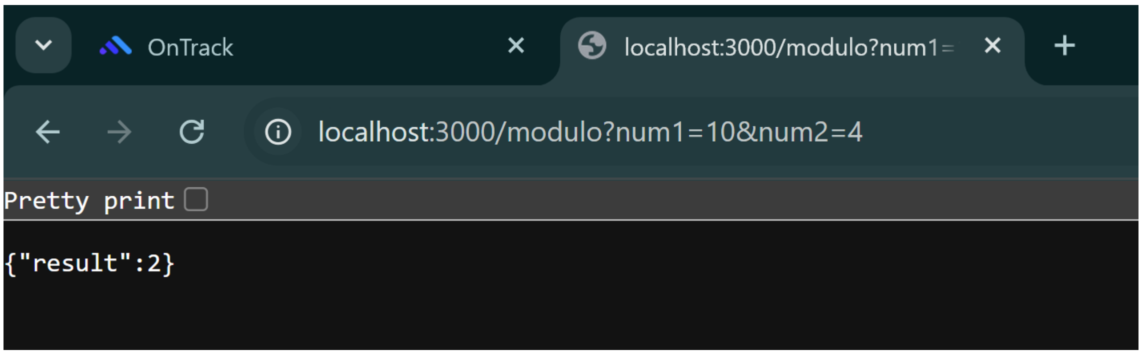


```
});

// Modulo endpoint (remainder after num1 divided by num2)
app.get('/modulo', (req, res) => {
    const num1 = parseFloat(req.query.num1);
    const num2 = parseFloat(req.query.num2);

    if (isNaN(num1) || isNaN(num2)) {
        return res.status(400).json({ error: 'Both num1 and num2 must be valid numbers.' });
    }

    if (num2 === 0) {
        return res.status(400).json({ error: 'Modulo by zero is not allowed.' });
    }

    res.json({ result: num1 % num2 });
});
```



OnTrack
localhost:3000/modulo?num1=
localhost:3000/modulo?num1=10&num2=0
Pretty print

```
{"error":"Modulo by zero is not allowed."}
```



OnTrack
localhost:3000/modulo?num1=
localhost:3000/modulo?num1=10&num2=4
Pretty print

```
{"result":2}
```