

Task 4.1P

Calculator Microservice – Documentation

This document provides a **step-by-step guide** on setting up, running, and understanding the **Calculator Microservice** built using **Node.js** and **Express.js**. The microservice includes **four arithmetic operations** (addition, subtraction, multiplication, division) and **logs requests and errors** using **Winston**.

1. Project Overview

The **Calculator Microservice** is a **RESTful API** that performs basic arithmetic operations. It is implemented using:

- **Node.js & Express.js** – Handles API requests
- **Winston** – Logs all API calls and errors
- **GitHub** – Stores the source code for collaboration

2. Project Folder Structure

```
calculator-microservice/  
|-- logs/          <-- Stores log files  
|  |-- combined.log <-- Logs all API requests  
|  |-- error.log   <-- Logs errors only  
|-- server.js      <-- Main API code  
|-- package.json   <-- Project dependencies and metadata  
|-- node_modules/  <-- Installed dependencies
```

3. Step-by-Step Setup Guide

- Step 1: Install Node.js
Verify **Node.js** is installed using:
 - `node -v`
 - `npm -v`
- Step 2: Initialize the Node.js Project
Navigate to the project folder and initialize a Node.js project:
 - `npm init -y`This creates a `package.json` file.
- Step 3: Install Dependencies
Install the required packages:
 - `npm install express Winston`**express** – For handling HTTP requests
winston – For logging API requests and errors

4. API Implementation (server.js)

- Import Required Modules

```
const express = require('express');  
const winston = require('winston');
```

express – Creates the API server

winston – Logs API requests and errors

- Initialize Express App

```
const app = express();  
const PORT = 3000;
```

app – Stores the Express application

PORT – Defines the port number for the server

- Configure Winston Logging

```
const logger = winston.createLogger({  
  level: 'info',  
  format: winston.format.json(),  
  defaultMeta: { service: 'calculator-microservice' },  
  transports: [  
    new winston.transports.Console({ format: winston.format.simple() }),  
    new winston.transports.File({ filename: 'logs/error.log', level: 'error' }),  
    new winston.transports.File({ filename: 'logs/combined.log' }),  
  ],  
});
```

Console Transport – Displays logs in the terminal

File Transports – Saves logs in *logs/error.log* and *logs/combined.log*

- Middleware to Log Incoming Requests

```
app.use((req, res, next) => {  
  logger.info(`Received ${req.method} request for ${req.url}`);  
  next();  
});
```

Logs each request to *combined.log*

- API Endpoints

Each endpoint takes **two query parameters** (*num1* and *num2*), performs the respective operation, and returns the result.

- Addition Endpoint

```
app.get('/add', (req, res) => {  
  const num1 = parseFloat(req.query.num1);  
  const num2 = parseFloat(req.query.num2);  
  if (isNaN(num1) || isNaN(num2)) {  
    logger.error('Invalid input for addition');  
    return res.status(400).json({ error: 'Invalid numbers' });  
  }  
  const result = num1 + num2;  
  res.json({ operation: 'addition', result });  
});
```

Validates numbers

Adds two numbers

Returns JSON response

- Subtraction Endpoint

```
app.get('/subtract', (req, res) => {  
  const num1 = parseFloat(req.query.num1);  
  const num2 = parseFloat(req.query.num2);  
  if (isNaN(num1) || isNaN(num2)) {  
    logger.error('Invalid input for subtraction');  
    return res.status(400).json({ error: 'Invalid numbers' });  
  }  
  const result = num1 - num2;  
  res.json({ operation: 'subtraction', result });  
});
```

Validates numbers

Subtracts two numbers

Returns JSON response

- Multiplication Endpoint

```
app.get('/multiply', (req, res) => {  
  const num1 = parseFloat(req.query.num1);  
  const num2 = parseFloat(req.query.num2);  
  if (isNaN(num1) || isNaN(num2)) {  
    logger.error('Invalid input for multiplication');  
    return res.status(400).json({ error: 'Invalid numbers' });  
  }  
  const result = num1 * num2;  
  res.json({ operation: 'multiplication', result });  
});
```

```

    }
    const result = num1 * num2;
    res.json({ operation: 'multiplication', result });
  });

```

Validates numbers

Multiply two numbers

Returns JSON response

- Division Endpoint

```

app.get('/divide', (req, res) => {
  const num1 = parseFloat(req.query.num1);
  const num2 = parseFloat(req.query.num2);
  if (isNaN(num1) || isNaN(num2) || num2 === 0) {
    logger.error('Invalid input for division');
    return res.status(400).json({ error: 'Invalid numbers or division by zero' });
  }
  const result = num1 / num2;
  res.json({ operation: 'division', result });
});

```

Validates numbers

Checks for division by zero

Multiply two numbers

Returns JSON response

- Start the Server

```

app.listen(PORT, () => {
  console.log(`Calculator microservice running at http://localhost:${PORT}`);
});

```

Starts the server on **port 3000**

Testing the services

- Step 1: Start the Server

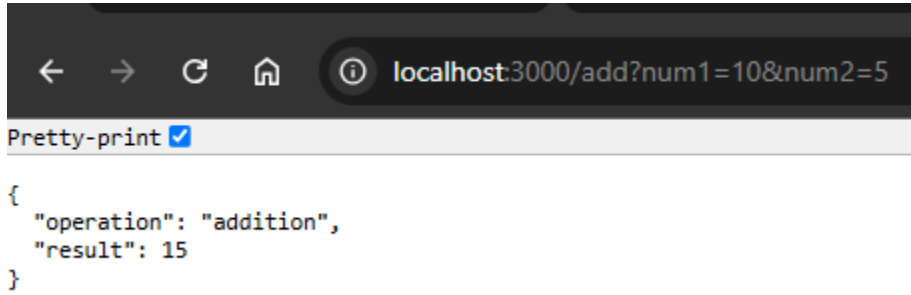
`node server.js`

Gives the output:

Calculator microservice running at <http://localhost:3000>

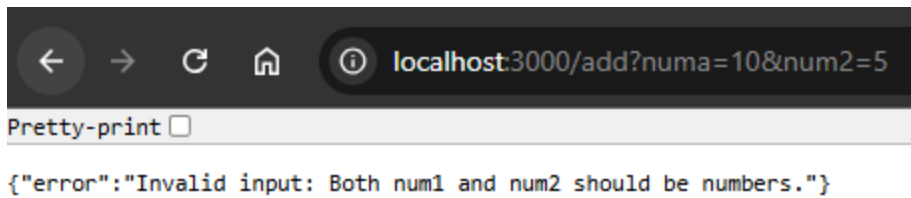
- Step 2: Test API Endpoints
I used the browser for testing

- Addition



A screenshot of a web browser's developer console. The address bar shows the URL `localhost:3000/add?num1=10&num2=5`. Below the address bar, the 'Pretty-print' checkbox is checked. The console displays a JSON object:

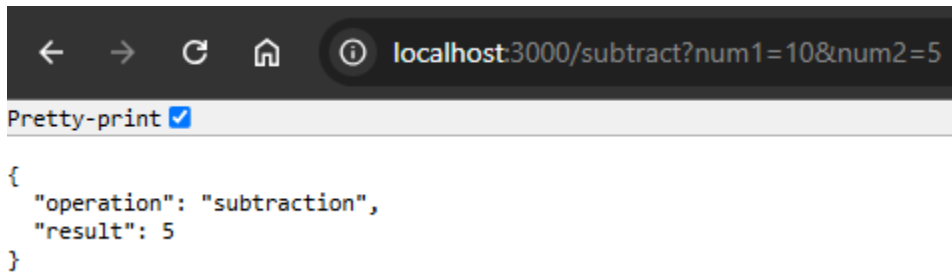
```
{  "operation": "addition",  "result": 15}
```



A screenshot of a web browser's developer console. The address bar shows the URL `localhost:3000/add?numa=10&num2=5`. Below the address bar, the 'Pretty-print' checkbox is unchecked. The console displays an error message:

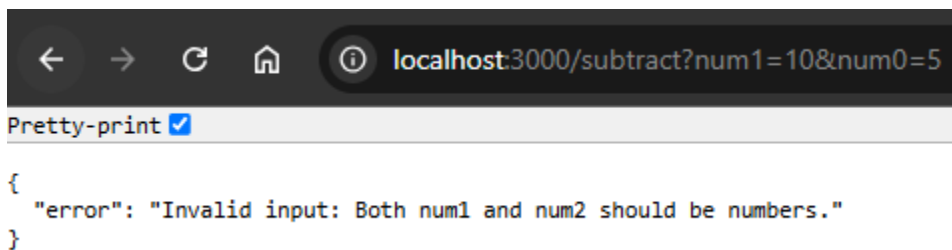
```
{"error": "Invalid input: Both num1 and num2 should be numbers."}
```

- Subtraction



A screenshot of a web browser's developer console. The address bar shows the URL `localhost:3000/subtract?num1=10&num2=5`. Below the address bar, the 'Pretty-print' checkbox is checked. The console displays a JSON object:

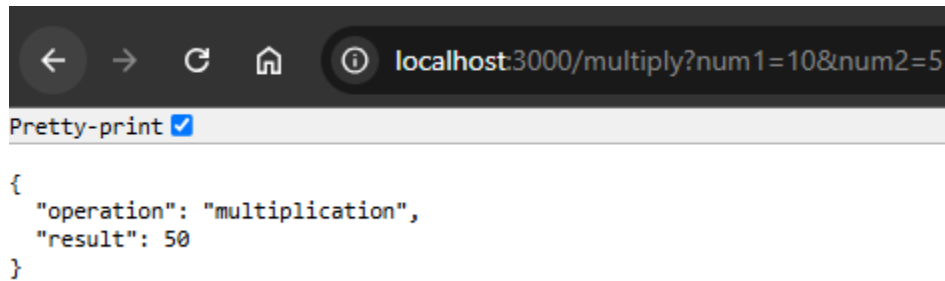
```
{  "operation": "subtraction",  "result": 5}
```



A screenshot of a web browser's developer console. The address bar shows the URL `localhost:3000/subtract?num1=10&num0=5`. Below the address bar, the 'Pretty-print' checkbox is checked. The console displays an error message:

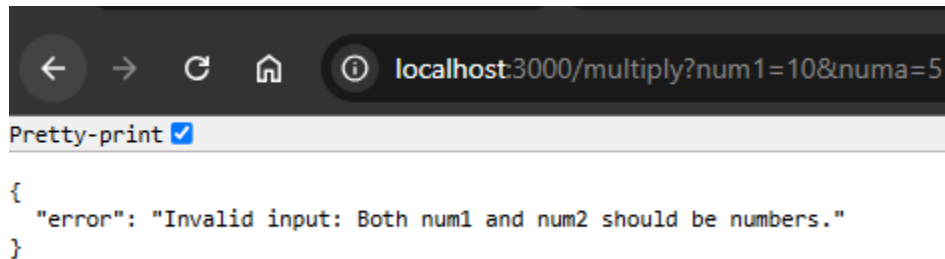
```
{  "error": "Invalid input: Both num1 and num2 should be numbers."}
```

- Multiplication



A screenshot of a web browser window. The address bar shows the URL `localhost:3000/multiply?num1=10&num2=5`. Below the address bar, there is a checkbox labeled "Pretty-print" which is checked. The main content area displays a JSON response:

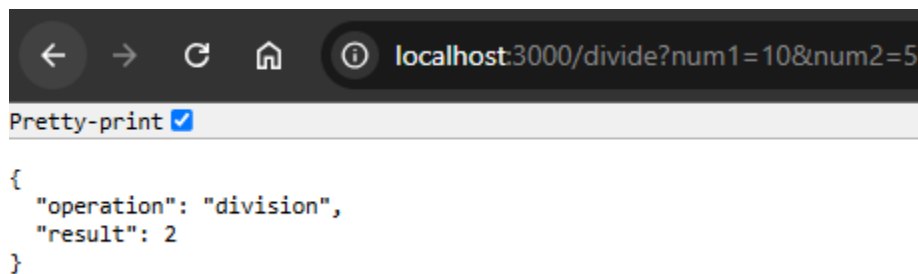
```
{  "operation": "multiplication",  "result": 50}
```



A screenshot of a web browser window. The address bar shows the URL `localhost:3000/multiply?num1=10&numa=5`. Below the address bar, there is a checkbox labeled "Pretty-print" which is checked. The main content area displays a JSON error response:

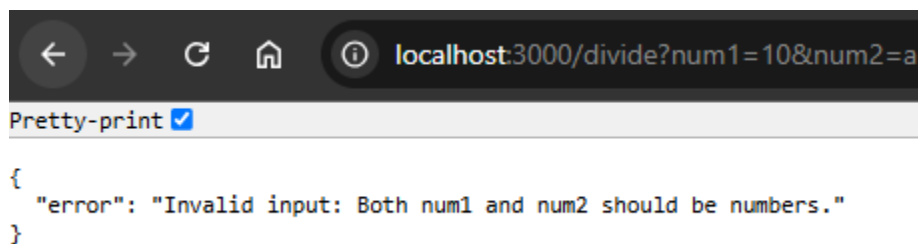
```
{  "error": "Invalid input: Both num1 and num2 should be numbers."}
```

- Division



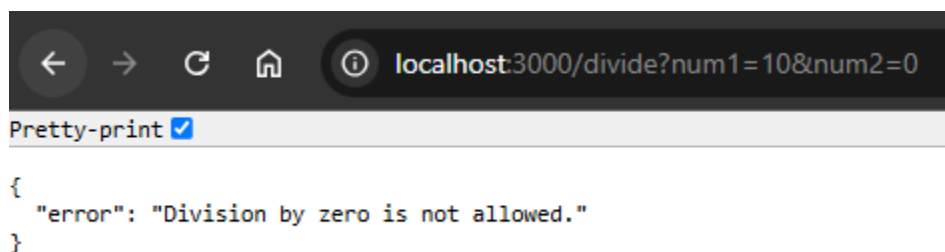
A screenshot of a web browser window. The address bar shows the URL `localhost:3000/divide?num1=10&num2=5`. Below the address bar, there is a checkbox labeled "Pretty-print" which is checked. The main content area displays a JSON response:

```
{  "operation": "division",  "result": 2}
```



A screenshot of a web browser window. The address bar shows the URL `localhost:3000/divide?num1=10&num2=a`. Below the address bar, there is a checkbox labeled "Pretty-print" which is checked. The main content area displays a JSON error response:

```
{  "error": "Invalid input: Both num1 and num2 should be numbers."}
```



A screenshot of a web browser window. The address bar shows the URL `localhost:3000/divide?num1=10&num2=0`. Below the address bar, there is a checkbox labeled "Pretty-print" which is checked. The main content area displays a JSON error response:

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

Viewing Logs

- Step 1: View Logs in Real-Time

Run:

Get-Content logs/combined.log

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

PS E:\Deakin\Semester 4\Cloud Native Applications\Task 4.1P\calculator-microservice> Get-Content logs/combined.log -Wait
{"level":"info","message":"Addition requested: 10 + 5 = 15","service":"calculator-microservice"}
{"level":"error","message":"Invalid input: Both num1 and num2 should be numbers.","service":"calculator-microservice"}
{"level":"info","message":"Subtraction requested: 10 - 5 = 5","service":"calculator-microservice"}
{"level":"error","message":"Invalid input: Both num1 and num2 should be numbers.","service":"calculator-microservice"}
{"level":"info","message":"Multiplication requested: 10 * 5 = 50","service":"calculator-microservice"}
{"level":"error","message":"Invalid input: Both num1 and num2 should be numbers.","service":"calculator-microservice"}
{"level":"info","message":"Division requested: 10 / 5 = 2","service":"calculator-microservice"}
{"level":"error","message":"Error: Division by zero is not allowed.","service":"calculator-microservice"}
{"level":"error","message":"Invalid input: Both num1 and num2 should be numbers.","service":"calculator-microservice"}

```

- Step 2: View Error Logs

Run:

Get-Content logs/error.log

```
PS E:\Deakin\Semester 4\Cloud Native Applications\Task 4.1P\calculator-microservice> Get-Content logs/error.log -Wait
{"level":"error","message":"Invalid input: Both num1 and num2 should be numbers.","service":"calculator-microservice"}
{"level":"error","message":"Invalid input: Both num1 and num2 should be numbers.","service":"calculator-microservice"}
{"level":"error","message":"Invalid input: Both num1 and num2 should be numbers.","service":"calculator-microservice"}
{"level":"error","message":"Error: Division by zero is not allowed.","service":"calculator-microservice"}
{"level":"error","message":"Invalid input: Both num1 and num2 should be numbers.","service":"calculator-microservice"}

```

Conclusion

This microservice successfully:

- Implements **addition, subtraction, multiplication, and division**
- Uses **Express.js** for API handling
- Uses **Winston** for logging
- Handles **error conditions**
- Stores logs for monitoring