Task 4.1P

Github link: https://github.com/vhh064/sit323_737-2024-t1-prac4p.git

Overview

This document covers the implementation of a basic arithmetic microservice using Node.js and the Express framework. The microservice provides endpoints for performing addition, subtraction, multiplication, and division on two numbers.

CODE EXPLANATION

Step 1: Server Set Up

First we need set up the Express server and Winston for logging

```
JS Logger.js > ...
const express = require('express');
const winston = require('winston');
```

Step 2: Setup Logger with Winston

Configure Winston to log messages in JSON format. Logs are written to two files: **error.log** for errors and **combined.log** for all logs. Additionally, logs are printed to the console in non-production environments.

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

if (process.env.NODE_ENV !== 'production') {
  logger.add(new winston.transports.Console({
    format: winston.format.simple(),
  }));
}
```

Step 3: Define Arithmetic Operation Function

'PerformOperation' is a function that takes an operation name and two numbers. It performs the specified arithmetic operation on the numbers. It also includes error handling for division by

```
// Arithmetic operation function
const performOperation = (operation, num1, num2) => {
  switch(operation) {
    case 'add':
      return num1 + num2;
    case 'subtract':
      return num1 - num2;
    case 'multiply':
      return num1 * num2;
    case 'divide':
      if(num2 === 0) throw new Error('Division by zero is not allowed.');
      return num1 / num2;
    default:
      throw new Error('Invalid operation');
  }
};
```

zero. In this task we will do 4 basic operation : ‘add’ , ‘subtract’, ‘divide’, and ‘multiplies’

Step 4: Implement Route Handler

A dynamic route handler is set up to match any operation included in the URL path (**/:operation**). It extracts the operation and number parameters from the request, validates them, performs the requested operation using ‘**performOperation’**, and sends back the result. Error handling is implemented for invalid inputs and internal errors, with appropriate HTTP status codes.

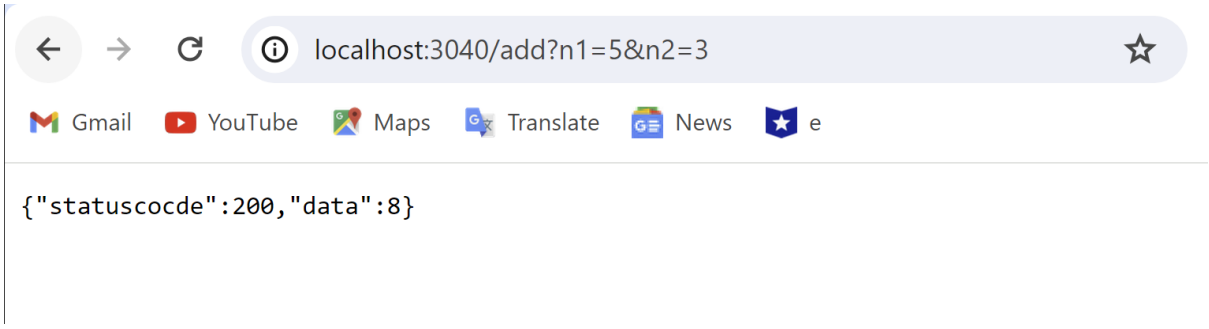
```
// Route handler
app.get('/:operation', (req, res) => {
  try {
    const { operation } = req.params;
    const num1 = parseFloat(req.query.n1);
    const num2 = parseFloat(req.query.n2);

    if(isNaN(num1) || isNaN(num2)) {
      throw new Error('Please provide valid numbers for n1 and n2.');
    }

    const result = performOperation(operation, num1, num2);
    res.status(200).json({ statusCode: 200, result: result });
    logger.info(`Operation performed: ${operation} with n1: ${num1}, n2: ${num2}, result: ${result}`);
  } catch (error) {
    logger.error(`Error performing operation: ${req.params.operation}. Error: ${error.message}`);
    const statusCode = error.message === 'Division by zero is not allowed.' || error.message === 'Please provide valid numbers for n1 and n2.' ? 400 : 500;
    res.status(statusCode).json({ statusCode: statusCode, message: error.message });
  }
});
```

Step 5: Start the Server

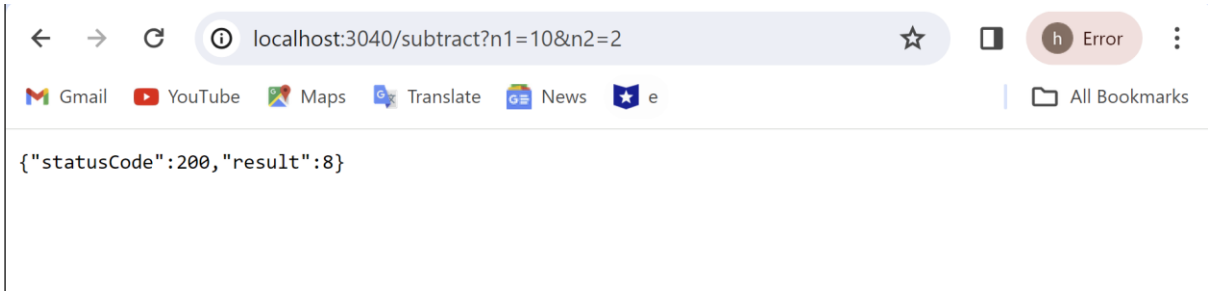
```
// Start the server
app.listen(port, () => {
  logger.info(`Arithmetic service listening at http://localhost:${port}`);
});
```

Output:

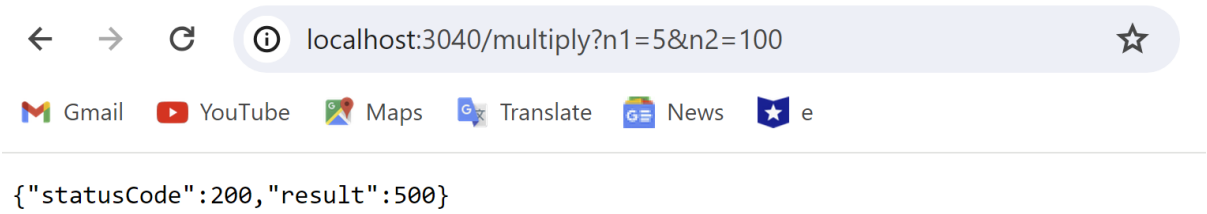
The ‘/add’ endpoint



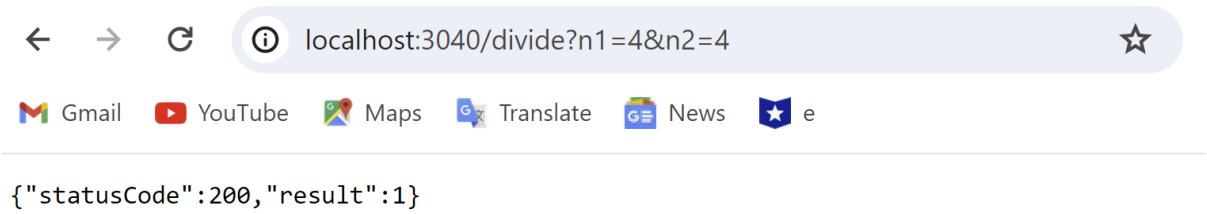
The ‘/subtract’ endpoint



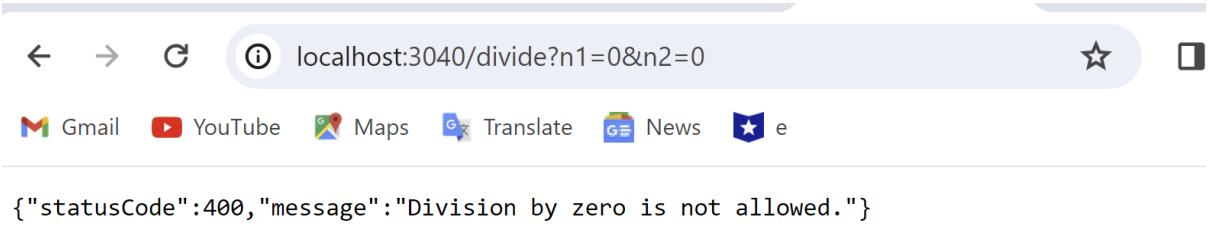
The '/multiply' endpoint



The '/divide' endpoint



Error:



localhost:3040/divide?n2=0&n2=0

Gmail YouTube Maps Translate News e

{"statusCode":400,"message":"Please provide valid numbers for n1 and n2."}