# Task4.2C

## Overview

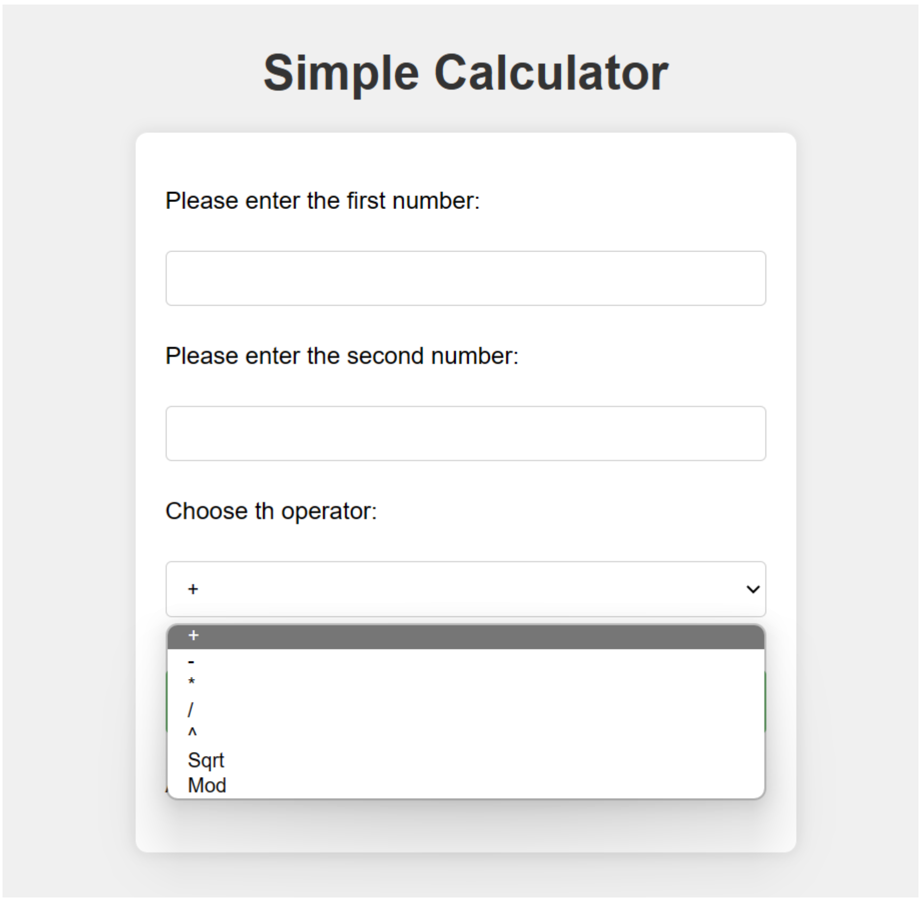
This web application offers a straightforward and user-friendly interface for performing a variety of mathematical calculations. Beyond basic arithmetic operations such as addition, subtraction, multiplication, and division, it also supports exponentiation, square root calculations, and modulo operations. The design prioritizes ease of use, allowing users to quickly input numbers, select a desired operation, and obtain results.

## GitHub Link

[userstarwind/sit323-737-2024-t1-prac4c (github.com)]

## Front-end

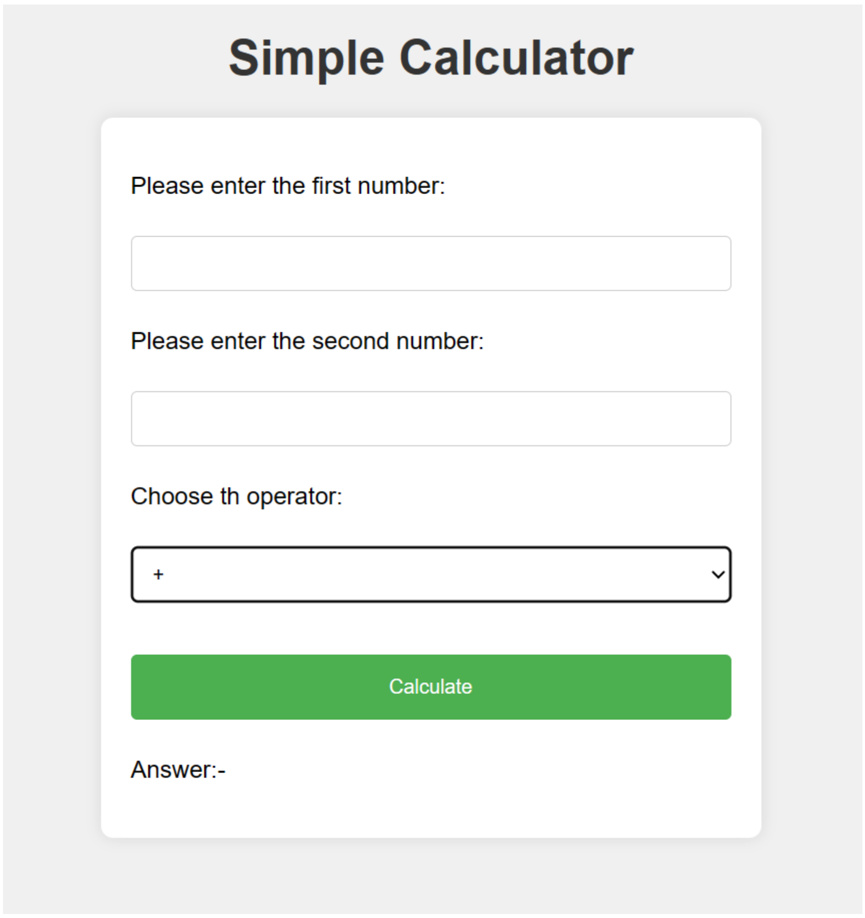
Developed using HTML, the front end presents a clean layout where users input two numbers and choose from a selection of operations, including addition (+), subtraction (-), multiplication (*), division (/), exponentiation (^), square root (Sqrt), and modulo (Mod). Users initiate calculations by clicking the "Calculate" button, which triggers the backend server to compute and return the result.


Simple Calculator

Please enter the first number:

Please enter the second number:

Choose th operator:


Simple Calculator
Please enter the first number:
Please enter the second number:
Choose th operator:
+
Calculate
Answer:-

## Back-end

The backend, implemented with Node.js and Express, features a well-structured API with endpoints corresponding to each mathematical operation. It leverages CORS middleware to handle cross-origin requests, facilitating seamless communication between the front-end and back-end across different origins.

The server validates input data for each operation, ensuring inputs are non-empty, non-null, and numerically valid. Special conditions are also checked, such as non-zero denominators for division and positive numbers for square root operations. Following validation, the server calculates the result and returns it to the front end. Error handling is robust, with meaningful error messages and status codes returned for invalid inputs or server issues.

## Logging and Error Handling

Logging is managed through Winston, a versatile logging library for Node.js, configured to write logs to files and, in development environments, to the console. This setup aids in monitoring the application's operation and troubleshooting.

## Related Code

index.html

```html
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Simple Calculator</title>
    <link rel="stylesheet" href="index.css">
</head>

<body>
    <script src="index.js" defer></script>
    <h1>Simple Calculator</h1>
    <div>
        <p>Please enter the first number:</p>
        <input name="num1">
        <p>Please enter the second number:</p>
        <input name="num2">
        <br>
        <p>Choose th operator:</p>
        <select name="operator">
            <option value="addition">+</option>
            <option value="subtraction">-</option>
            <option value="multiplication">*</option>
            <option value="division">/</option>
            <option value="exponentiation">^</option>
            <option value="squareroot">Sqrt</option>
            <option value="modulo">Mod</option>
        </select>
        <br>
        <br>
        <button name="calculate" id="calculateButton">Calculate</button>
        <br>
        <p>Answer:<span id="result">-</span></p>
    </div>
</body>

</html>
```

index.js

```
document.querySelector('#calculateButton').addEventListener('click', (event) => {
    event.preventDefault();
    const num1Input = document.querySelector('input[name="num1"]').value;
    const num2Input = document.querySelector('input[name="num2"]').value;
    const operator = document.querySelector('select[name="operator"]').value;
    const data = {
        "num1": num1Input,
        "num2": num2Input,
    };

    let apiUrl = `http://localhost:8080/${operator}`;

    fetch(apiUrl, {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json',
        },
        body: JSON.stringify(data),
    })
    .then(response => {
        if (!response.ok) {
            return response.json().then(errorData => {
                throw new Error(errorData.error || 'Unknown Error');
            });
        }
        return response.json();
    })
    .then(data => {
        console.log('Success:', data);
        document.getElementById('result').textContent = data.result;
    })
    .catch((error) => {
        console.error('Error:', error);
        alert(error.message);
    });
});

document.addEventListener('DOMContentLoaded', function() {
    var operatorSelect = document.querySelector('select[name="operator"]');
    var num2Input = document.querySelector('input[name="num2"]');

    operatorSelect.addEventListener('change', function() {
        if (this.value === 'squareroot') {
            num2Input.disabled = true;
            num2Input.value = 'None';
        } else {
            num2Input.disabled = false;
        }
    });
});
```

server.js

```
const express = require('express');
const cors = require('cors');
const app = express();
const PORT = 8080;
const winston = require('winston');

app.use(express.json());
app.use(cors());

const logger = winston.createLogger({
    level: 'info',
    format: winston.format.json(),
    defaultMeta: { service: 'user-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(),
    }));
}

const validateNumbers = (num1, num2, allowZeroInSecond = true) => {
    if (num1 === '' || num2 === '' || num1 == null || num2 == null) {
        return "One or both of the numbers is missing or empty.";
    }
    if (isNaN(num1) || isNaN(num2)) {
        return 'One or both inputs are not valid numbers.';
    }
    if (!allowZeroInSecond && parseFloat(num2) === 0) {
        return 'The second number can not be zero.';
    }
    return false;
};

const validateSquareRootNumber = (num1) => {
    if (num1 === '' || num1 == null ) {
        return "The number is missing or empty.";
    }
    if (isNaN(num1)) {
        return 'The input is not a valid number.';
    }
    if (parseFloat(num1) < 0) {
        return 'Square root of a negative number is not defined in real
numbers.';
    }
    return false;
};

app.post('/addition', (req, res) => {
    try {
        const { num1, num2 } = req.body;
        logger.info(`Addition request: ${num1}, ${num2}`);
        const error = validateNumbers(num1, num2);
        if (error) {
            return res.status(400).json({ error });
        }
        let result = parseFloat(num1) + parseFloat(num2);
        res.json({ result });
    } catch (e) {
        logger.error(`Error processing addition: ${e.message}`);
        res.status(500).json({ error: 'An unexpected error occurred.' });
    }
});

app.post('/subtraction', (req, res) => {
    try {
        const { num1, num2 } = req.body;
        logger.info(`Subtraction request: ${num1}, ${num2}`);
        const error = validateNumbers(num1, num2);
        if (error) {
            return res.status(400).json({ error });
        }
        let result = parseFloat(num1) - parseFloat(num2);
        res.json({ result });
    } catch (e) {
        logger.error(`Error processing addition: ${e.message}`);
        res.status(500).json({ error: 'An unexpected error occurred.' });
    }
});

app.post('/multiplication', (req, res) => {
    try {
        const { num1, num2 } = req.body;
        logger.info(`Multiplication request: ${num1}, ${num2}`);
        const error = validateNumbers(num1, num2);
        if (error) {
            return res.status(400).json({ error });
        }
        let result = parseFloat(num1) * parseFloat(num2);
        res.json({ result });
    } catch (e) {
        logger.error(`Error processing addition: ${e.message}`);
        res.status(500).json({ error: 'An unexpected error occurred.' });
    }
});

app.post('/division', (req, res) => {
    try {
        const { num1, num2 } = req.body;
        logger.info(`Division request: ${num1}, ${num2}`);
        const error = validateNumbers(num1, num2, false);
        if (error) {
            return res.status(400).json({ error });
        }
        let result = parseFloat(num1) / parseFloat(num2);
        res.json({ result });
    } catch (e) {
        logger.error(`Error processing addition: ${e.message}`);
        res.status(500).json({ error: 'An unexpected error occurred.' });
    }
});

app.post('/exponentiation', (req, res) => {
    try {
        const { num1, num2 } = req.body;
        logger.info(`Exponentiation request: ${num1}, ${num2}`);
        const error = validateNumbers(num1, num2);
        if (error) {
            return res.status(400).json({ error });
        }
        let result = Math.pow(parseFloat(num1), parseFloat(num2));
        res.json({ result });
    } catch (e) {
        logger.error(`Error processing addition: ${e.message}`);
        res.status(500).json({ error: 'An unexpected error occurred.' });
    }
});

app.post('/squareroot', (req, res) => {
    try {
        const { num1 } = req.body;
        logger.info(`Squareroot request: ${num1}`);
        const error = validateSquareRootNumber(num1);
        if (error) {
            return res.status(400).json({ error });
        }
        let result = Math.sqrt(parseFloat(num1));
        res.json({ result });
    } catch (e) {
        logger.error(`Error processing addition: ${e.message}`);
        res.status(500).json({ error: 'An unexpected error occurred.' });
    }
});

app.post('/modulo', (req, res) => {
    try {
        const { num1, num2 } = req.body;
        logger.info(`Modulo request: ${num1}, ${num2}`);
        const error = validateNumbers(num1, num2, false);
        if (error) {
            return res.status(400).json({ error });
        }
        let result = parseFloat(num1) % parseFloat(num2);
        res.json({ result });
    } catch (e) {
        logger.error(`Error processing addition: ${e.message}`);
        res.status(500).json({error: 'An unexpected error occurred.' });

    }
});

app.listen(PORT, () => {
    console.log(`Server is running on http://localhost:${PORT}`);
});
```