UIT2602 WEB PROGRAMMING
Exercise 8 - Implementation of Calculator RestAPI

Name: S. Selcia
Reg. No.: 3122215002098
Section: IT B

1. Aim:

To implement a calculator application using RestAPI Framework.

2. Required Web Tools and Methodology:

- Express: A Node.js framework for API development. (`npm install express`)
- HTML, CSS, Javascript for frontend
- Git for version control
- Web Browser: To test the functionality of the website.

3. Implementation Procedure:

a. Get the operands using input tag and the operator using option tag.
b. The div enclosing the calculator is directed to the corresponding app and controller.
c. Have a calculate button whose ng-click is directed to the calculate function in the controller.
d. Have a tag for the answer which is linked to the result passed by the controller.
e. Inside the controller get the operator and pass it to the $http.get function and pass the first and second operands as query params in the get function.
f. Store the result in $scope.answer

frontend/index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Calculator</title>
    <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
    <style>
body {
    font-family: 'Arial', sans-serif;
    margin: 0;
    padding: 0;
    display: flex;
    justify-content: center;
```

```css
    align-items: center;
    min-height: 100vh;
    background-color: #f8f9fa;
}
.calculator {
    width: 300px;
    padding: 20px;
    background-color: #ffffff;
    border-radius: 10px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
}
h1 {
    text-align: center;
    color: #333333;
    margin-bottom: 20px;
}
.input-group {
    display: flex;
    flex-direction: column;
    gap: 10px;
}
input[type="number"],
select,
button {
    width: 100%;
    padding: 10px;
    border: 1px solid #ced4da;
    border-radius: 5px;
    font-size: 16px;
    outline: none;
}
button {
    background-color: #007bff;
    color: #ffffff;
    border: none;
    border-radius: 5px;
    font-size: 16px;
    cursor: pointer;
    transition: background-color 0.3s;
}
button:hover {
    background-color: #0056b3;
}
.result {
    margin-top: 20px;
    text-align: center;
    font-size: 24px;
    font-weight: bold;
    color: #333333;
}
    </style>
</head>
<body ng-app="calculatorApp" ng-controller="calculatorController">
    <div>
        <label for="">Operand 1:</label>
        <input type="number" ng-model="operand1">
    </div>
    <div>
        <label for="">Operator:</label>
        <select ng-model="operator">
            <option value="add">+</option>
            <option value="subtract">-</option>
            <option value="multiply">x</option>
            <option value="divide">/</option>
        </select>
    </div>
    <div>
        <label for="">Operand 2:</label>
        <input type="number" ng-model="operand2">
    </div>
    <div>
        <button ng-click="calculate()">Calculate</button>
    </div>
    <div ng-show="showResult">
        <label for="">Result:</label>
        <span>{{ answer }}</span>
    </div>
    <script src="app.js"></script>
</body>
</html>
```

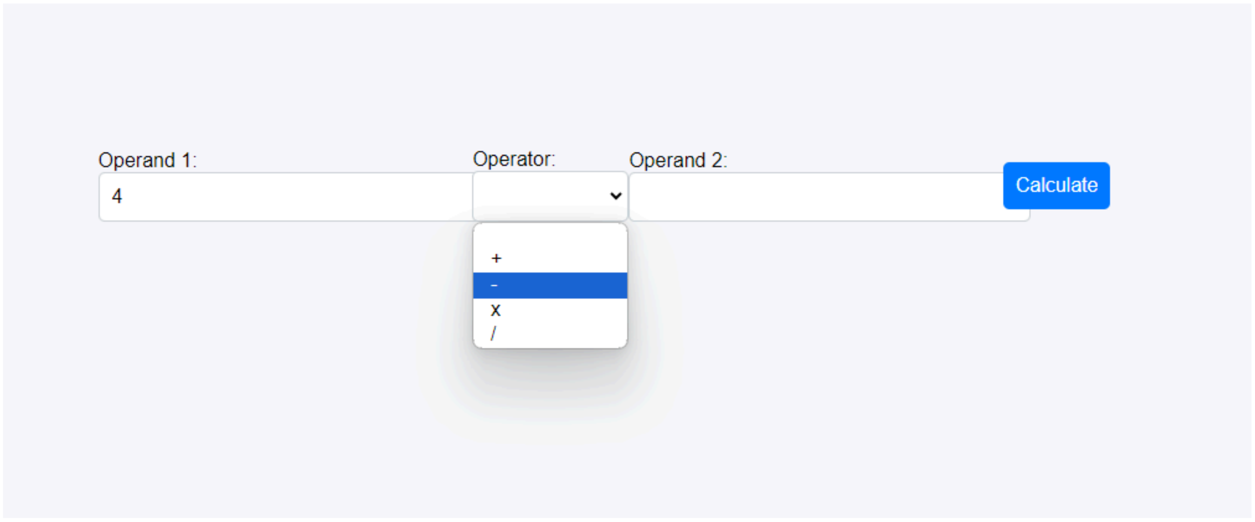
<u>frontend/app.js</u>

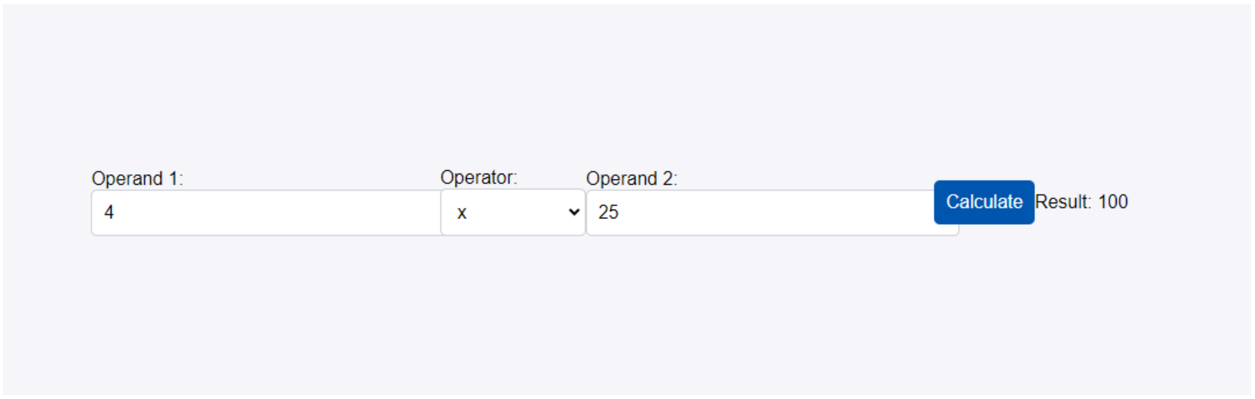
```
var app = angular.module('calculatorApp', []);
app.controller('calculatorController', function($scope, $http) {
    $scope.showResult = false;
    $scope.calculate = function() {
        $http.get('/calculate', {
            params: {
                first: $scope.operand1,
                second: $scope.operand2,
                operator: $scope.operator
            }
        }).then(function(response) {
            $scope.answer = response.data.result;
            $scope.showResult = true;
        });
    };
});
```

<u>**backend/script.js**</u>

```
const express = require('express');
const app = express();
const path = require('path');
app.use(express.static(path.join(__dirname, "../frontend")));
app.get('/', (req, res) => {
    res.sendFile(path.join(__dirname, '../frontend/index.html'));
});
// Define routes
app.get('/calculate', (req, res) => {
    const { first, second, operator } = req.query;
    const num1 = parseFloat(first);
    const num2 = parseFloat(second);
    let result;
    switch(operator) {
        case 'add':
            result = num1 + num2;
            break;
        case 'subtract':
            result = num1 - num2;
            break;
        case 'multiply':
            result = num1 * num2;
            break;
        case 'divide':
            result = num1 / num2;
            break;
        default:
            result = 'Invalid operator';
    }
    res.json({ result });
});
// Start the server
const PORT = process.env.PORT || 3000;
app.listen(PORT, () => {
    console.log(`Server is running on port ${PORT}`);
});
```

OUTPUT:





Result: 100

Conclusion:

Integrating AJAX enhances web applications by enabling dynamic features like live search, dynamic form submissions, and content updates without refreshing the page. It facilitates real-time communication between the client and server, resulting in faster responses and a smoother user experience. AJAX-driven live search provides instant results as users type, improving search efficiency. Dynamic form submissions allow seamless data submission without interrupting the user's flow. Content updates without page refreshes ensure users receive the latest information without manual reloads, enhancing interactivity.