Sweng Design Doc

Overview

- **Status:** In progress
- **Members:**
  - Neil Shevlin
  - Mamoke Arubayi

This project aims to build a simple calculator, which takes a string representing mathematical expressions and returns an answer.

Context

Goal

To demonstrate the use of software engineering best practices like use of version control, remote repositories, testing and clean code, while delivering a calculator application that meets the project specification.

Specific Features

1. Application programming interface(API) that accepts a string as an input. The string represents a mathematical expression from the user.
2. Application tests the validity of the string expression. Checking for invalid characters, improper syntax, not having duplicate operations.
3. Application returns a string with the correct answer in the case of a valid string or an appropriate error message in the case of invalid input expression.
4. The application will also run unit tests before being deployed to the main branch of the repository.

Out of Scope

1. Mathematical operations other than addition, subtraction and multiplication(ie, division is out of scope)

2. Non integer string inputs.

Proposed directory structure

```
CalculatorApplication
|   README.md
|   tsconfig.json
|   tslint.json
|   package.json
|   nodemon.json
|   .gitignore
|   .prettierrc
|
└──src
|   |   server.ts
|   |   config.ts
|   |   app.ts
|   |
|   └──routes
|   |   |   index.ts
|   |   |   calculator.ts
|   |   |
|   └──controllers
|   |   |   calculator.ts
|   |   |
|   └──utils
|       |   logger.ts
|
└──test
    |   01_index.spec.ts
    |   02_calculator.spec.ts
    |   mocha.opts
```

Description of proposal

We intend to complete this project in typescript, which is a superset of Javascript. This Application can accept http requests that are sending a string to the API. This API will parse the string, determine validity and return a response to the client user. The application will run unit tests using the mocha framework and the Chai assertion library for node.

Tasks and timeline

- [ ] Setup initial typescript project

- [ ] Configure routes to receive strings
- [ ] Connect routing to controller logic
- [ ] Develop string validity parser
- [ ] Develop mathematical operational parser
- [ ] Return correct response
- [ ] Refactor controller logic to use helper function system