Calculator API

We want to implement an HTTP Server evaluates given mathematical expression in a string.

Specs

```
POST /compute
{
  "computation":"(6 * 3) + 4 + (8 / 2) * 5"
}
```

StatusCode: 200

```
{
  "result":42
}
```

Execution Plan

- **Template:** First create a project template that contains
  - Libraries to add
    - HttpServer Library: Http4s: https://http4s.org/
    - Json Parsing Library: Circe: https://circe.github.io/circe/
  - Start-up script that spins up the server
  - Basic API that accepts a GET request without any parameter and returns a constant response.
    - Simple example for this API is **/health.** In general, this API accepts a GET request without any parameters as mentioned above and returns hardcoded **200.**
- **Implementation:** Now, since your server is up and ready, you can start implementing your API
  - Add library that parses the given expression.
    - https://www.lihaoyi.com/fastparse/#ExampleParsers
  - Use the output data structure to compute the final result or have the library to compute the result of the expression.

    - In this step, please assume that given expression will be always valid.

- **Validation and Exception Handling:** Once you have the implementation for happy path, we can now focus more on error cases, these are the questions that you should ask in this step?
    - What should I do if the given expression is not valid?
    - What should I do if the library or code that I have written throws an unexpected error?
    - How much user should know about these errors/validation results?

Here are a bunch of invalid examples:

Missing Closing Parenthesis

```
POST /compute
{
  "computation":"(6 * 3 + 4 + (8 / 2) * 5"
}
```

StatusCode: 400

```
{
  "Message": “Given expression is invalid!”
}
```

Expression has characters

```
POST /compute
{
  "computation":"(6 * 3 + 4 + (X / 2) * 5"
}
```

StatusCode: 400

```
{
  "Message": “Given expression is invalid!”
}
```

- **Testing:** Once you assure that you handled all possible cases, you can write some unit tests to ensure that your application has good coverage
    - You can use **scalatest** to write your unit tests in Scala.
    - Start writing a test case for **/health** to adjust into **ScalaTest.**

- Then list edge cases for your API.
- And write edge cases as a test case.