CSC3065 Cloud Computing
Assessment 2: Web Calculator
Submission

Student Number: 40204578

Name: Ciaran Duffin

Task A

Function One

Mathematical Function: Multiply
Repository URL: http://gitlab.hal.davecutting.uk/40204578/webcalc-multiply
Live Service URL: http://webcalc-multiply.40204578.qpc.hal.davecutting.uk/

Description of Implementation (language, methods, paradigm, etc):

I used Python and Flask to create a rest API to handle multiplication. I defined a function 'multiply' that accepts two parameters x and y and multiplies the two and returns the result. The endpoint accepts two parameters x and y and validates them checking if they are not null, are numeric inputs and finally if they are whole numbers. If either of the input parameters does not pass the validation tests, a 400 response is returned, the 'error' field on the response is set to true, and the 'string' field is set to a useful error message detailing why the request failed. If both input parameters are valid then the calculation is carried out by the previously mentioned function, 'error' field on the response is set to false, the 'string' field is set to a text description of the calculation carried out and the result of the calculation is assigned to the 'answer' field.

The content-type of the response is set to 'application/json' and the Access-Control-Allow-Origin is set to '*'.

Appropriate changes were made to the frontend to allow the function to be used, a multiply button was added with an appropriate 'Onclick' handler to ensure the rest API is hit.

Description of Testing:

I included a .gitlab-ci.yml file in my repository in order to implement continuous integration using the GitLab runner. Tests in the multiplyTest.py file will run every time a commit is pushed to the master branch.

I added unit tests for my 'multiply' function which tests the function three times, using a combination of both positive and negative numbers.

Integration tests were also added to test the actual endpoint works appropriately. Three aspects of the endpoint were tested in isolation, response codes, response bodies, and the content-type of the response. For both testing the response bodies and response codes, 6 tests were carried out, 3 valid tests, testing how the API responds with positive and negative valid inputs and 3 invalids tests, testing how the API responds to no input, decimal number input and non-numeric input. For testing the content type only two tests were carried out, one test with valid input and one test with invalid input.

Summary of Expected Results:
Response Codes:

- Valid tests should return response code 200
- Invalid tests should return response code 400

Response Bodes:

- All response bodies should make the predefined 'expectedBody' variable

Content-Type:

- All responses should be of type 'application/json'

Of 17 tests run, both unit and integration, all 17 tests passed.

Anything else to highlight:
Screenshots and code snippets included below highlight the logic of the API endpoint, how validation was carried out, some example tests and proof of continuous integration.

```
@app.route('/')
def mult():
    statusCode = 200

    x = request.args.get('x')
    y = request.args.get('y')

    jsonResponse = validateInputs(x,y)

    if (jsonResponse.get("error") is True):
        statusCode = 400
    else:
        answer = multiply.multiply(int(x), int(y))
        jsonResponse = {
            "error": False,
            "string": f"{x}*{y}={answer}",
            "answer": answer,
        }

    reply = json.dumps(jsonResponse)
    r = Response(response=reply, status=statusCode)
    r.headers["Content-Type"]="application/json"
    r.headers["Access-Control-Allow-Origin"] = "*"
    return r
```

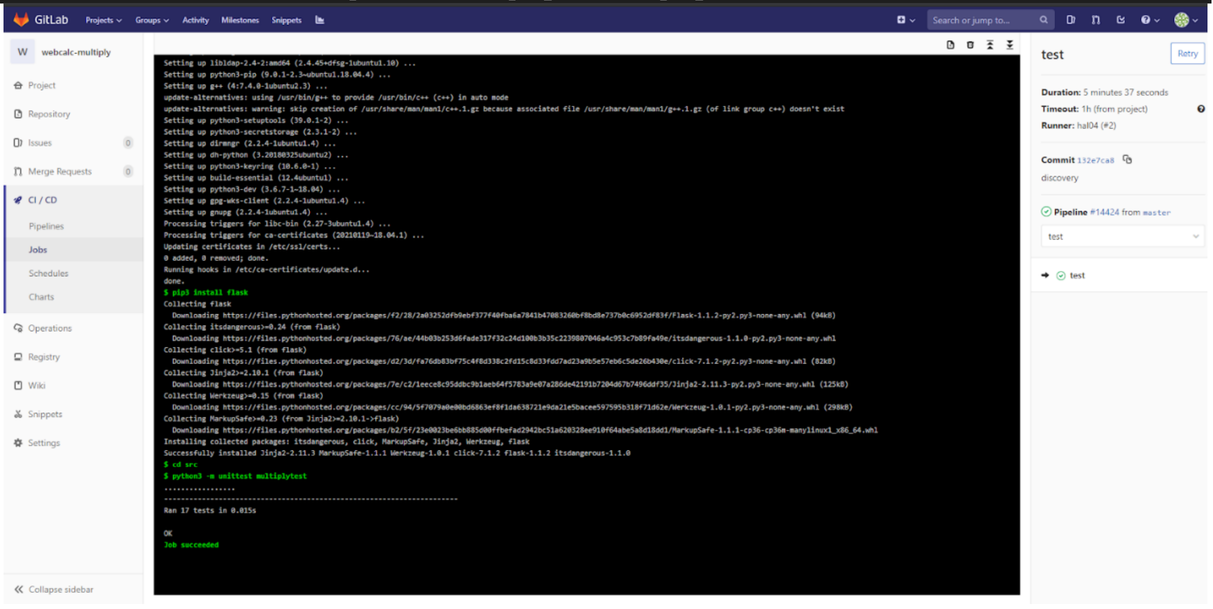
```
def validateInputs(x,y):
    jsonResponse = {
        "error": False,
        "string": "",
        "answer":0,
    }

    #Check inputs not null
    if x is None or y is None:
        jsonResponse = {
        "error": True,
        "string": "One or both required parameters (x and y) are
missing.",
        "answer":0,
        }

    #Check inputs are numeric
    elif not helpers.string_is_number(x) or not
helpers.string_is_number(y):
        jsonResponse = {
        "error": True,
        "string": "One or both required parameters (x and y) are non
numeric.",
        "answer":0,
        }

    #Check inputs are whole numbers
    elif not helpers.string_is_int(x) or not helpers.string_is_int(y):
        jsonResponse = {
        "error": True,
        "string": "One or both required parameters (x and y) are not
whole numbers.",
        "answer":0,
        }

    return jsonResponse
```

```
#Check response 400
    def test_noParams_returns400(self):
        response = self.tester.get("/")
        statuscode = response.status_code
        self.assertEqual(statuscode, 400)
```

```
def test_missingParams_returnsCorrectBody(self):
        response = self.tester.get("/")
        body = response.data.decode("utf-8")

        expectedBody = '{"error": true, "string": "One or both required
parameters (x and y) are missing.", "answer": 0}'

        self.assertEqual(body, expectedBody)
```

```
def test_validParams_bothPos_returnsCorrectContentType(self):
        response = self.tester.get("/?x=4&y=2")
        contentType = response.content_type

        expectedContentType = 'application/json'

        self.assertEqual(contentType, expectedContentType)
```

```
#||--Unit Tests--||
    def test_multiply_bothPos(self):
        self.assertEqual(multiply.multiply(2,4),8)
```



```
Setting up libldap-2.4-2:amd64 (2.4.45+dfsg-1ubuntu1.10) ...
Setting up python3-pip (9.0.1-2.3~ubuntu1.18.04.4) ...
Setting up g++ (4:7.4.0-1ubuntu2.3) ...
update-alternatives: using /usr/bin/g++ to provide /usr/bin/c++ (c++) in auto mode
update-alternatives: warning: skip creation of /usr/share/man/man1/c++.1.gz because associated file /usr/share/man/man1/g++.1.gz (of link group c++) doesn't exist
Setting up python3-setuptools (39.0.1-2) ...
Setting up python3-secretstorage (2.3.1-2) ...
Setting up dirmngr (2.2.4-1ubuntu1.4) ...
Setting up dh-python (3.20180325ubuntu2) ...
Setting up python3-keyring (10.6.0-1) ...
Setting up build-essential (12.4ubuntu1) ...
Setting up python3-dev (3.6.7-1~18.04) ...
Setting up gpg-wks-client (2.2.4-1ubuntu1.4) ...
Setting up gnupg (2.2.4-1ubuntu1.4) ...
Processing triggers for libc-bin (2.27-3ubuntu1.4) ...
Processing triggers for ca-certificates (20210119~18.04.1) ...
Updating certificates in /etc/ssl/certs...
0 added, 0 removed; done.
Running hooks in /etc/ca-certificates/update.d...
done.
$ pip3 install flask
Collecting flask
  Downloading https://files.pythonhosted.org/packages/f2/28/2a03252dfb9ebf377f40fba6a7841b47083260bf8bd8e737b0c6952df83f/Flask-1.1.2-py2.py3-none-any.whl (94kB)
Collecting itsdangerous>=0.24 (from flask)
  Downloading https://files.pythonhosted.org/packages/76/ae/44b03b253d6fade317f32c24d100b3b35c2239807046a4c953c7b89fa49e/itsdangerous-1.1.0-py2.py3-none-any.whl
Collecting click>=5.1 (from flask)
  Downloading https://files.pythonhosted.org/packages/d2/3d/fa76db83bf75c4f8d338c2fd15c8d33fdd7ad23a9b5e57eb6c5de26b430e/click-7.1.2-py2.py3-none-any.whl (82kB)
Collecting Jinja2>=2.10.1 (from flask)
  Downloading https://files.pythonhosted.org/packages/7e/c2/1eece8c95ddbc9b1aeb64f5783a9e07a286de42191b7204d67b7496ddf35/Jinja2-2.11.3-py2.py3-none-any.whl (125kB)
Collecting Werkzeug>=0.15 (from flask)
  Downloading https://files.pythonhosted.org/packages/cc/94/5f7079a0e00bd6863ef8f1da638721e9da21e5bacee597595b318f71d62e/Werkzeug-1.0.1-py2.py3-none-any.whl (298kB)
Collecting MarkupSafe>=0.23 (from Jinja2>=2.10.1->flask)
  Downloading https://files.pythonhosted.org/packages/b2/5f/23e0023be6bb885d00ffbefad2942bc51a620328ee910f64abe5a8d18dd1/MarkupSafe-1.1.1-cp36-cp36m-manylinux1_x86_64.whl
Installing collected packages: itsdangerous, click, MarkupSafe, Jinja2, Werkzeug, flask
Successfully installed Jinja2-2.11.3 MarkupSafe-1.1.1 Werkzeug-1.0.1 click-7.1.2 flask-1.1.2 itsdangerous-1.1.0
$ cd src
$ python3 -m unittest multiplytest
.................
----------------------------------------------------------------------
Ran 17 tests in 0.015s

OK
Job succeeded
```

Duration: 5 minutes 37 seconds
Timeout: 1h (from project)
Runner: hal04 (#2)

Commit 132e7ca8
discovery

Pipeline #14424 from master

test

→ test

Function Two

Mathematical Function: Modulo
Repository URL: http://gitlab.hal.davecutting.uk/40204578/webcalc-modulo
Live Service URL: http://webcalc-modulo.40204578.qpc.hal.davecutting.uk/

Description of Implementation (language, methods, paradigm, etc):

I used Golang to create a rest API to handle the modulo function on the calculator. I defined a function modulo that accepts two parameters x and y and performs the modulo function using them and returns the result. The endpoint accepts two parameters x and y and validates them checking if they are not null, are numeric and finally if they are whole numbers. If either of the input parameters does not pass the validation tests, a 400 response is returned, the 'error' field on the response is set to true, and the 'string' field is set to a useful error message detailing why the request failed. If both input parameters are valid then the calculation is carried out by the previously mentioned function, the 'error' field on the response is set to false, the 'string' field is set to a text description of the calculation carried out and the result of the calculation is assigned to the 'answer' field.

The content-type of the response is set to 'application/json' and the Access-Control-Allow-Origin is set to '*'.

Appropriate changes were made to the frontend to allow the function to be used, a modulo button was added with an appropriate 'Onclick' handler to ensure the rest API is hit.

Description of Testing:

I included a .gitlab-ci.yml file in my repository in order to implement continuous integration using the GitLab runner. Tests in the main_test.go file will run every time a commit is pushed to the master branch.

I added unit tests for my 'modulo' function which tests the function seven times using a combination of both positive and negative numbers.

Integration tests were also added to test the actual endpoint works appropriately. Three aspects of the endpoint were tested in isolation, response codes, response bodies, and the content-type of the response. When testing the response codes, 4 tests were carried out, 1 valid and 3 invalid. The valid test ensured valid input would return a response of 200 and the three invalid tests ensured that passing no data, non numeric data or non whole number data would return a response code of 400. When testing the response bodies 6 tests were carried out, 3 valid and 3 invalid. The valid test if the endpoint response body was correct when passing both negative and positive numbers while the invalid tests ensure responses are correct when passing none, non numeric and non whole number inputs. When testing content-type one valid test was used and one invalid test was used.

Summary of Expected Results:
Response Codes:

- Valid tests should return response code 200

- Invalid tests should return response code 400

Response Bodes:

- All response bodies should make the predefined ‘expectedBody’ variable

Content-Type:

- All responses should be of type ‘application/json’

Of 19 tests run, both unit and integration, all 19 tests passed.

Anything else to highlight:
Screenshots and code snippets included below highlight the logic of the API endpoint, how validation was carried out, some example tests and proof of continuous integration.

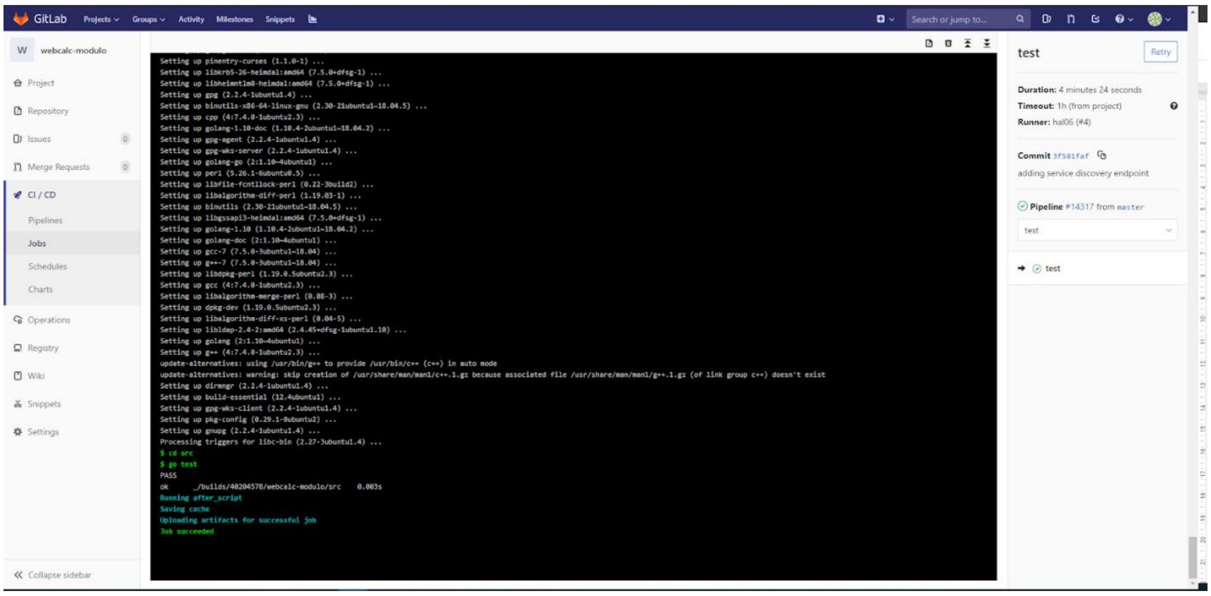
```
func HandleModuloRequest(w http.ResponseWriter, r *http.Request){
    statusCode := 200

    query := r.URL.Query()
    x := query.Get("x")
    y := query.Get("y")

    jsonResponse := validateInputs(x,y)

    if (jsonResponse.Error == true){
        statusCode = 400
    } else {
        x_int, _ := strconv.Atoi(x)
        y_int, _ :=strconv.Atoi(y)

        answer := Modulo(x_int, y_int)
        jsonResponse.Answer = answer
        jsonResponse.Error = false
        jsonResponse.String = x+" modulo "+y+"="+strconv.Itoa(answer)
    }

    w.Header().Set("Access-Control-Allow-Origin", "*")
    w.Header().Set("Content-Type", "application/json")

    if (statusCode == 200){
        w.WriteHeader(http.StatusOK)
    }else{
        w.WriteHeader(http.StatusBadRequest)
    }


    json.NewEncoder(w).Encode(jsonResponse)
}
```

```
func validateInputs(x string, y string) Output {
    jsonResponse := Output{
        Error: false,
        String: "",
        Answer: 0,
    }

    //Check inputs not null
    //Check input are numeric
    //Check inputs are whole numbers

    if (x == "" || y == ""){
        jsonResponse = Output{
            Error: true,
            String: "One or both required parameters (x and y) are
missing.",
            Answer: 0,
        }
    } else if(!isNumeric(x) || !isNumeric(y)) {
        jsonResponse = Output{
            Error: true,
            String: "One or both required parameters (x and y) are non
numeric.",
            Answer: 0,
        }
    } else if(!isInteger(x) || !isInteger(y)) {
        jsonResponse = Output{
            Error: true,
            String: "One or both required parameters (x and y) are not
whole numbers.",
            Answer: 0,
        }
    }


    return jsonResponse;
}
```

```
var moduloResults = []ModuloResult{
    {4,3,1},
    {8,7,1},
    {7,8,7},
    {4,2,0},
    {6,-2,0},
    {-6,-2,0},
    {-6,2,0},
}


//||--Unit Tests--||

func TestModulo(t *testing.T){
    for _, test := range moduloResults {
        result := Modulo(test.x, test.y)
        if result != test.expected {
            t.Fatal("Failed. Expected:"+
string(test.expected)+"\nActual:"+strconv.Itoa(result))
        }
    }
}
```

```
func Test_ValidInput_Returns200ResponseCode(t *testing.T){
    req, err := http.NewRequest("GET", "/?x=8&y=2", nil)
    if err != nil {
        t.Fatal(err)
    }

    rr := httptest.NewRecorder()
    handler := http.HandlerFunc(HandleModuloRequest)

    handler.ServeHTTP(rr, req)


    if status := rr.Code; status != http.StatusOK {
        t.Errorf("handler returned wrong status code: got %v want
%v",status,http.StatusOK)
    }
}
```

```
func Test_ValidInput8And6_ReturnsCorrectBody(t *testing.T){
    req, err := http.NewRequest("GET", "/?x=8&y=6", nil)
    if err != nil {
        t.Fatal(err)
    }

    rr := httptest.NewRecorder()
    handler := http.HandlerFunc(HandleModuloRequest)

    handler.ServeHTTP(rr, req)

    json_repsonse := strings.TrimRight(string(rr.Body.String()), "\n")

    if status := rr.Code; status != http.StatusOK {
        t.Errorf("handler retured wrong status code: got %v want
%v",status,http.StatusOK)
    }

    expected := `{"error":false,"string":"8 modulo 6=2","answer":2}`
    if json_repsonse != expected {
        t.Errorf("Failed. Handler returned unexpected body.
\nReturned:%v \nExpected:%v", json_repsonse, expected)
    }
}
```



```
Setting up pinentry-curses (1.1.0-1) ...
Setting up libkrb5-26-heimdal:amd64 (7.5.0+dfsg-1) ...
Setting up libheimntlm0-heimdal:amd64 (7.5.0+dfsg-1) ...
Setting up gpg (2.2.4-1ubuntu1.4) ...
Setting up binutils-x86-64-linux-gnu (2.30-21ubuntu1~18.04.5) ...
Setting up cpp (4:7.4.0-1ubuntu2.3) ...
Setting up golang-1.10-doc (1.10.4-2ubuntu1~18.04.2) ...
Setting up gpg-agent (2.2.4-1ubuntu1.4) ...
Setting up gpg-wks-server (2.2.4-1ubuntu1.4) ...
Setting up golang-go (2:1.10~4ubuntu1) ...
Setting up perl (5.26.1-6ubuntu0.5) ...
Setting up libfile-fcntllock-perl (0.22-3build2) ...
Setting up libalgorithm-diff-perl (1.19.03-1) ...
Setting up binutils (2.30-21ubuntu1~18.04.5) ...
Setting up libgssapi3-heimdal:amd64 (7.5.0+dfsg-1) ...
Setting up golang-1.10 (1.10.4-2ubuntu1~18.04.2) ...
Setting up golang-doc (2:1.10~4ubuntu1) ...
Setting up gcc-7 (7.5.0-3ubuntu1~18.04) ...
Setting up g++-7 (7.5.0-3ubuntu1~18.04) ...
Setting up libdpkg-perl (1.19.0.5ubuntu2.3) ...
Setting up gcc (4:7.4.0-1ubuntu2.3) ...
Setting up libalgorithm-merge-perl (0.08-3) ...
Setting up dpkg-dev (1.19.0.5ubuntu2.3) ...
Setting up libalgorithm-diff-xs-perl (0.04-5) ...
Setting up libldap-2.4-2:amd64 (2.4.45+dfsg-1ubuntu1.10) ...
Setting up golang (2:1.10~4ubuntu1) ...
Setting up g++ (4:7.4.0-1ubuntu2.3) ...
update-alternatives: using /usr/bin/g++ to provide /usr/bin/c++ (c++) in auto mode
update-alternatives: warning: skip creation of /usr/share/man/man1/c++.1.gz because associated file /usr/share/man/man1/g++.1.gz (of link group c++) doesn't exist
Setting up dirmngr (2.2.4-1ubuntu1.4) ...
Setting up build-essential (12.4ubuntu1) ...
Setting up gpg-wks-client (2.2.4-1ubuntu1.4) ...
Setting up pkg-config (0.29.1-0ubuntu2) ...
Setting up gnupg (2.2.4-1ubuntu1.4) ...
Processing triggers for libc-bin (2.27-3ubuntu1.4) ...
$ cd src
$ go test
PASS
ok      _/builds/40204578/webcalc-modulo/src    0.003s
Running after_script
Saving cache
Uploading artifacts for successful job
Job succeeded
```

test

Duration: 4 minutes 24 seconds
Timeout: 1h (from project)
Runner: hal06 (#4)

Commit 3f581faf
adding service discovery endpoint

Pipeline #14317 from master

test

→ test

Function Three

Mathematical Function: Divide
Repository URL: http://gitlab.hal.davecutting.uk/40204578/webcalc-divide
Live Service URL: http://webcalc-divide.40204578.qpc.hal.davecutting.uk/

Description of Implementation (language, methods, paradigm, etc):

I used C# and .NET Core to create a rest API to handle the division. I define a function 'Divide' that accepts two parameters x and y and divides x by y and returns the result. The endpoint accepts two parameters, x and y, validates them checking if they are not null, are numeric inputs and finally if they are whole numbers. If either of the input parameters does not pass the validation tests, a 400 response is returned, the 'error' field on the response is set to true, and the 'string' field is set to a useful error message detailing why the request failed. If both input parameters are valid then the calculation is carried out by the previously mentioned function, 'error' field on the response is set to false, the 'string' field is set to a text description of the calculator carried out and the result of the calculation is assigned to the 'answer' field.

The content-type of the response is set to 'application/json' and the Access-Control-Allow-Origin is set to '*'.

Appropriate changes were made to the frontend to allow the function to be used, a divide button was added with an appropriate 'Onclick' handler to ensure the rest API is hit.

Description of Testing:

I included a .gitlab-ci.yml file in my repository in order to implement continuous integration using the GitLab runner. Tests in the DivisionTests.cs file will run every time a commit is pushed to the master branch.

I added units tests for my 'divide' function which tests the function 9 times, using a combination of both positive and negative numbers.

Integration tests were also added to test the actual endpoint works appropriately. Three aspects of the endpoint were tested in isolation, response codes, response bodies, and the content-type of the response. For both testing the response bodies and response codes, 6 tests were carried out, 3 valid tests, testing how the API responds with positive and negative valid inputs and 3 invalids tests, testing how the API responds to no input, decimal number input and non-numeric input. For testing the content type only two tests were carried out, one test with valid input and one test with invalid input.

Summary of Expected Results:
Response Codes:

- Valid tests should return response code 200
- Invalid tests should return response code 400

Response Bodes:

- All response bodies should make the predefined 'expectedBody' variable

Content-Type:

- All responses should be of type ‘application/json’

Of 24 tests run, both unit and integration, all 24 tests passed.

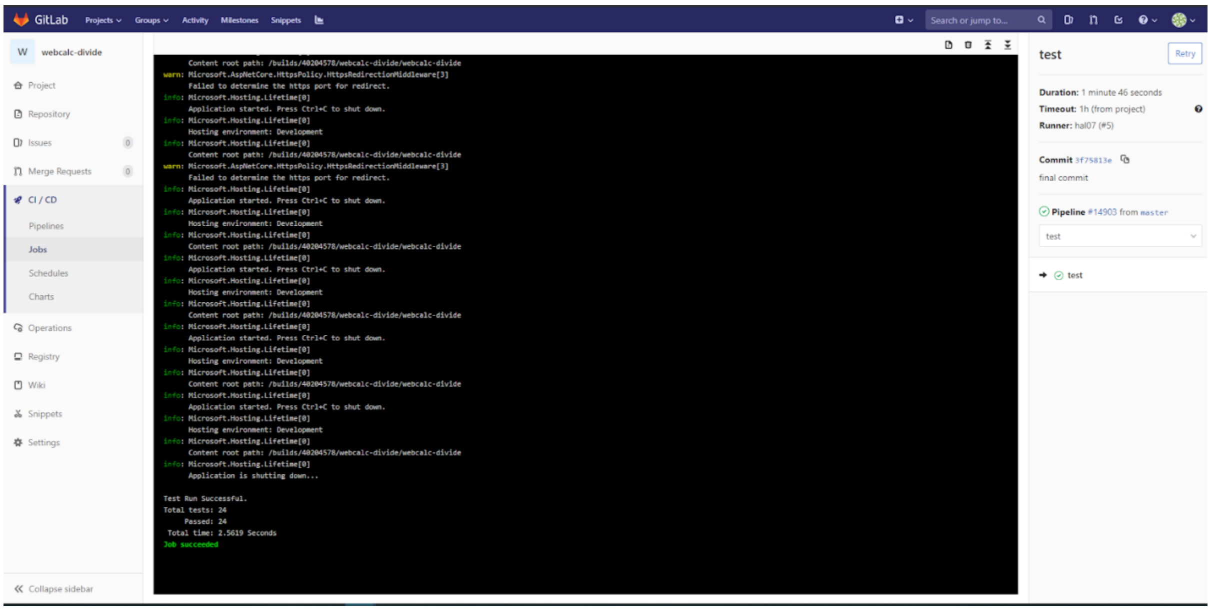
Anything else to highlight:
Screenshots and code snippets included below highlight the logic of the API endpoint, how validation was carried out, some example tests and proof of continuous integration.

```
[HttpGet]
        public IActionResult Get([FromQuery(Name = "x")] string x,
[FromQuery(Name = "y")] string y)
        {
            var statusCode = 200;


            var output = _divisionService.ValidateInputs(x, y);

            if (output.Error)
            {
                statusCode = 400;
            }
            else
            {
                var answer = _divisionService.Divide(Int32.Parse(x),
Int32.Parse(y));
                output = new OperationOutput
                {
                    Error = false,
                    String = $"{x}/{y}={answer}",
                    Answer = answer,
                };
            }

            if (statusCode == 200)
            {
                return Ok(output);
            }
            else
            {
                return BadRequest(output);
            }


        }
```

```
 public OperationOutput ValidateInputs(string x, string y)
        {
            var jsonResponse = new OperationOutput
            {
                Error = false,
                String = "",
                Answer = 0,
            };

            //Check inputs not null
            if (String.IsNullOrEmpty(x) || String.IsNullOrEmpty(y))
            {
                jsonResponse = new OperationOutput
                {
                    Error = true,
                    String = "One or both required parameters (x and y)
are missing.",
                    Answer = 0,
                };

            }

            //Check inputs are numeric
            else if (!(double.TryParse(x, out double a)) ||
!(double.TryParse(y, out double b)))
            {
                jsonResponse = new OperationOutput
                {
                    Error = true,
                    String = "One or both required parameters (x and y)
are non numeric.",
                    Answer = 0,
                };
            }

            //Check inputs are whole numbers
            else if (!(int.TryParse(x, out int c)) || !(int.TryParse(y,
out int d)))
            {
                jsonResponse = new OperationOutput
                {
                    Error = true,
                    String = "One or both required parameters (x and y)
are not whole numbers.",
                    Answer = 0,
                };
            }

            return jsonResponse;

        }
```

```
[Theory]
        [InlineData(4, 2)]
        [InlineData(6, 3)]
        [InlineData(8, 4)]
        [InlineData(10, 5)]
        [InlineData(12, 6)]
        public void  Divide_NumberDividedByItsHalf_ReturnsTwo(int x,
int y)
        {
            Assert.Equal(2, _divisionService.Divide(x, y));
        }
```

```
//<--Response Codes-->
        [Theory]
        [InlineData("/?x=foo&y=bar")]
        [InlineData("/")]
        [InlineData("/?x=4.2&y=3.4")]
        public async Task InValidInputsReturn400(string query)
        {
            var response = await _testClient.GetAsync(query);

            Assert.Equal(HttpStatusCode.BadRequest,
response.StatusCode);

        }



        [Theory]
        [InlineData("/?x=4&y=2")]
        [InlineData("/?x=6&y=-2")]
        [InlineData("/?x=-4&y=-8")]
        public async Task ValidInputsReturn200(string query)
        {
            var response = await _testClient.GetAsync(query);

            Assert.Equal(HttpStatusCode.OK, response.StatusCode);

        }
```



```
      Content root path: /builds/40204578/webcalc-divide/webcalc-divide
warn: Microsoft.AspNetCore.HttpsPolicy.HttpsRedirectionMiddleware[3]
      Failed to determine the https port for redirect.
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Development
info: Microsoft.Hosting.Lifetime[0]
      Content root path: /builds/40204578/webcalc-divide/webcalc-divide
warn: Microsoft.AspNetCore.HttpsPolicy.HttpsRedirectionMiddleware[3]
      Failed to determine the https port for redirect.
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Development
info: Microsoft.Hosting.Lifetime[0]
      Content root path: /builds/40204578/webcalc-divide/webcalc-divide
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Development
info: Microsoft.Hosting.Lifetime[0]
      Content root path: /builds/40204578/webcalc-divide/webcalc-divide
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Development
info: Microsoft.Hosting.Lifetime[0]
      Content root path: /builds/40204578/webcalc-divide/webcalc-divide
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Development
info: Microsoft.Hosting.Lifetime[0]
      Content root path: /builds/40204578/webcalc-divide/webcalc-divide
info: Microsoft.Hosting.Lifetime[0]
      Application is shutting down...

Test Run Successful.
Total tests: 24
     Passed: 24
 Total time: 2.5619 Seconds
Job succeeded
```

test

Duration: 1 minute 46 seconds
Timeout: 1h (from project)
Runner: hal07 (#5)

Commit 3f75813e
final commit

Pipeline #14903 from master
test

test

Function Four

Mathematical Function: Square
Repository URL: http://gitlab.hal.davecutting.uk/40204578/webcalc-square
Live Service URL: http://webcalc-square.40204578.qpc.hal.davecutting.uk/

Description of Implementation (language, methods, paradigm, etc):

I used Dart and Aqueduct to create a rest API to handle the square function. I defined a function 'square' that takes one parameter x and multiplies it by itself and returns the result. The endpoint accepts one parameter x and validates it to check if it is not null, numeric and finally if it is a whole number. If x doesn't pass the validation checks then a 400 response is returned, the 'error' field on the response is set to true, and the 'string' field is set to a useful error message detailing why the request failed. If both input parameters are valid then the calculation is carried out by the previously mentioned function, the 'error' field on the response is set to false, the 'string' field is set to a text description of the calculation carried out and the result of the calculation is assigned to the 'answer' field.

The content-type of the response is set to 'application/json' and the Access-Control-Allow-Origin is set to '*'.

Appropriate changes were made to the frontend to allow the function to be used, a square button was added with an appropriate 'Onclick' handler to ensure the rest API is hit.

Description of Testing:

I included a .gitlab-ci.yml file in my repository in order to implement continuous integration using the GitLab runner. Tests in the square_test.dart file will run every time a commit is pushed to the master branch.

I added unit tests for my 'square' function which tests the function twice, using a positive number and then a negative number.

Integration tests were also added to test the actual endpoint works appropriately. Three aspects of the endpoint were tested in isolation, response codes, response bodies, and the content-type of the response. For both testing the response bodies and response codes, 5 tests were carried out, 2 valid tests, testing how the API responds with a positive and a negative valid input and 3 invalids tests, testing how the API responds to no input, a decimal number input and a non-numeric input. For testing the content type only two tests were carried out, one test with valid input and one test with invalid input.

Summary of Expected Results:
Response Codes:
- Valid tests should return response code 200
- Invalid tests should return response code 400

Response Bodes:
- All response bodies should make the predefined 'expectedBody' variable

Content-Type:

- All responses should be of type ‘application/json’

Of 14 tests run, both unit and integration, all 14 tests passed.

Anything else to highlight:
Screenshots and code snippets included below highlight the logic of the API endpoint, how validation was carried out, some example tests and proof of continuous integration.

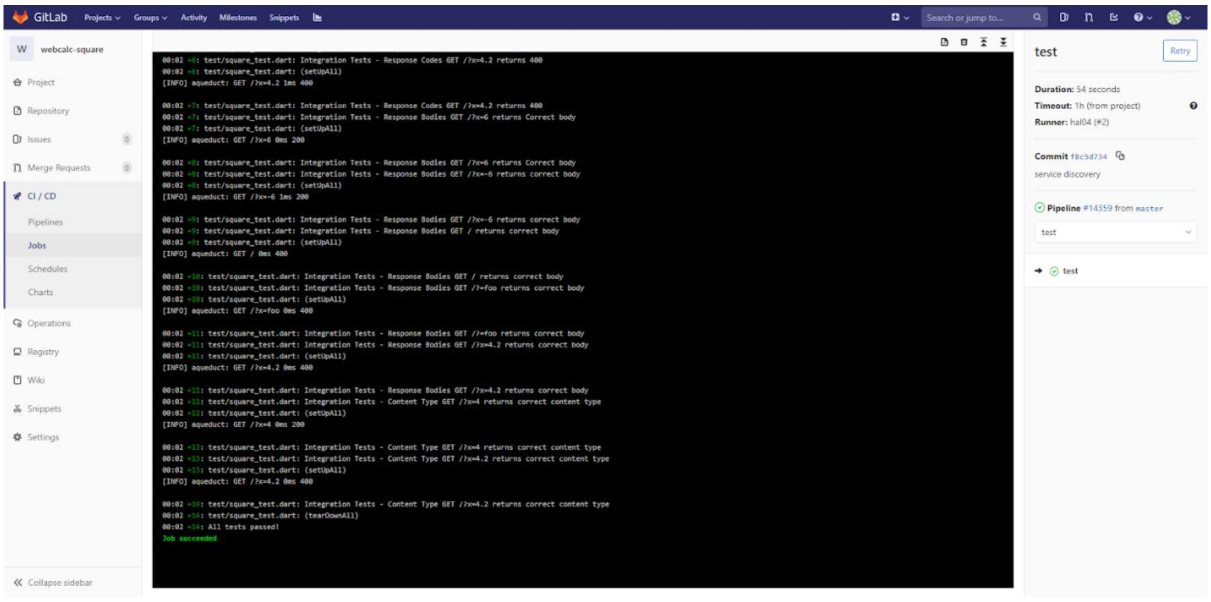
```
@Operation.get()
  Future<Response> getSquare({@Bind.query('x') String x}) async {
    var statusCode = 200;

    var jsonResponse = validateInput(x);

    if (jsonResponse['error'] == true) {
      statusCode = 400;
    } else {
      final answer = square(int.parse(x));
      jsonResponse = {
        "error": false,
        "string": "$x^2=$answer",
        "answer": answer,
      };
    }

    final headers = {
      'Access-Control_Allow_Origin': '*',
      'Content-Type': 'application/json',
    };

    final response = Response(statusCode, headers, jsonResponse);

    return response;
  }
```

```
Map<String, Object> validateInput(String x) {
  var jsonResponse = {
    "error": false,
    "string": "",
    "answer": 0,
  };

  //Check x is not null
  if (x == null) {
    jsonResponse = {
      "error": true,
      "string": "Parameter x is missing.",
      "answer": 0,
    };
  }
  //Check x is numeric
  else if (!isNumeric(x)){
    jsonResponse = {
      "error": true,
      "string": "Parameter x is non numeric.",
      "answer": 0,
    };
  }
  //Check x is a whole number
  else if (!isInteger(x)){
    jsonResponse = {
      "error": true,
      "string": "Parameter x is not a whole number.",
      "answer": 0,
    };
  }

  return jsonResponse;
}
```

```
group("Unit Tests", () {
    test("square a positive number works", () {
      expect(square(2), 4);
    });
    test("square a negative number works", () {
      expect(square(-2), 4);
    });
  });
```

```
group("Integration Tests - Response Codes", () {
    test("GET /?x=6 returns 200", () async {
      final response = await harness.agent.get("/?x=6");

      expectResponse(response, 200);
    });
```

```
group("Integration Tests - Response Bodies", () {
    test("GET /?x=6 returns Correct body", () async {
      final response = await harness.agent.get("/?x=6");

      final expectedBody = {'error': false, 'string': '6^2=36',
'answer': 36};

      expectResponse(response, 200, body: expectedBody);
    });
```



```
00:02 +6: test/square_test.dart: Integration Tests - Response Codes GET /?x=4.2 returns 400
00:02 +6: test/square_test.dart: (setUpAll)
[INFO] aqueduct: GET /?x=4.2 1ms 400

00:02 +7: test/square_test.dart: Integration Tests - Response Codes GET /?x=4.2 returns 400
00:02 +7: test/square_test.dart: Integration Tests - Response Bodies GET /?x=6 returns Correct body
00:02 +7: test/square_test.dart: (setUpAll)
[INFO] aqueduct: GET /?x=6 0ms 200

00:02 +8: test/square_test.dart: Integration Tests - Response Bodies GET /?x=6 returns Correct body
00:02 +8: test/square_test.dart: Integration Tests - Response Bodies GET /?x=-6 returns correct body
00:02 +8: test/square_test.dart: (setUpAll)
[INFO] aqueduct: GET /?x=-6 1ms 200

00:02 +9: test/square_test.dart: Integration Tests - Response Bodies GET /?x=-6 returns correct body
00:02 +9: test/square_test.dart: Integration Tests - Response Bodies GET / returns correct body
00:02 +9: test/square_test.dart: (setUpAll)
[INFO] aqueduct: GET / 0ms 400

00:02 +10: test/square_test.dart: Integration Tests - Response Bodies GET / returns correct body
00:02 +10: test/square_test.dart: Integration Tests - Response Bodies GET /?=foo returns correct body
00:02 +10: test/square_test.dart: (setUpAll)
[INFO] aqueduct: GET /?x=foo 0ms 400

00:02 +11: test/square_test.dart: Integration Tests - Response Bodies GET /?=foo returns correct body
00:02 +11: test/square_test.dart: Integration Tests - Response Bodies GET /?x=4.2 returns correct body
00:02 +11: test/square_test.dart: (setUpAll)
[INFO] aqueduct: GET /?x=4.2 0ms 400

00:02 +12: test/square_test.dart: Integration Tests - Response Bodies GET /?x=4.2 returns correct body
00:02 +12: test/square_test.dart: Integration Tests - Content Type GET /?x=4 returns correct content type
00:02 +12: test/square_test.dart: (setUpAll)
[INFO] aqueduct: GET /?x=4 0ms 200

00:02 +13: test/square_test.dart: Integration Tests - Content Type GET /?x=4 returns correct content type
00:02 +13: test/square_test.dart: Integration Tests - Content Type GET /?x=4.2 returns correct content type
00:02 +13: test/square_test.dart: (setUpAll)
[INFO] aqueduct: GET /?x=4.2 0ms 400

00:02 +14: test/square_test.dart: Integration Tests - Content Type GET /?x=4.2 returns correct content type
00:02 +14: test/square_test.dart: (tearDownAll)
00:02 +14: All tests passed!
Job succeeded
```

Duration: 54 seconds
Timeout: 1h (from project)
Runner: hal04 (#2)

Commit f8c5d734
service discovery

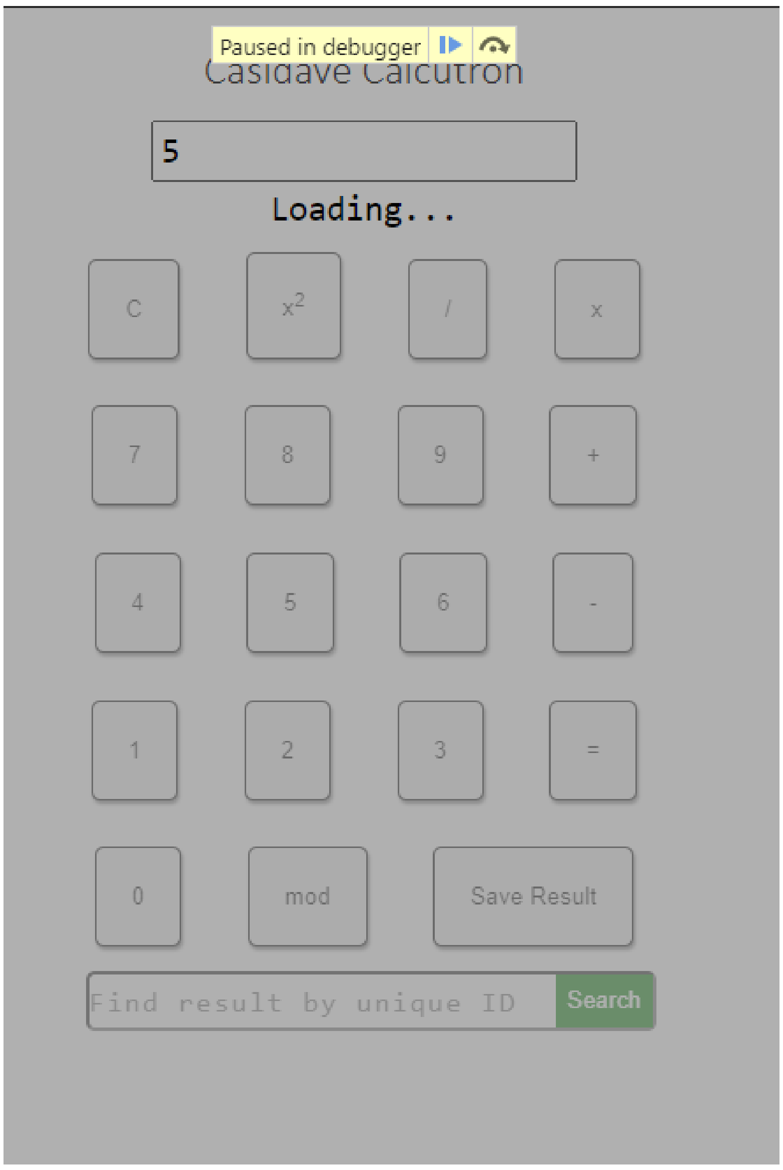
Pipeline #14359 from master

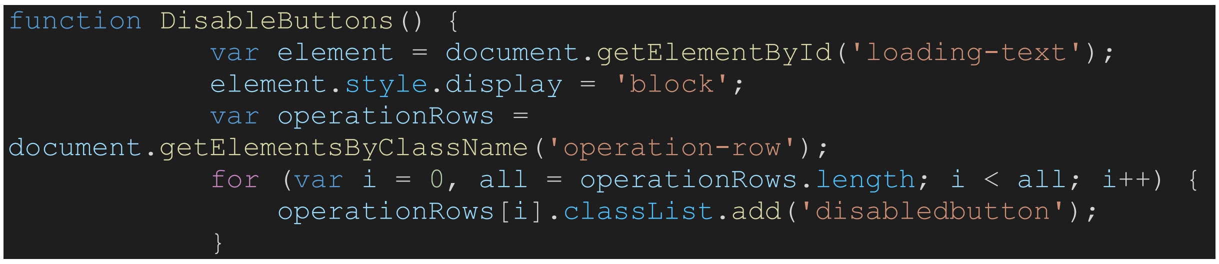
test

test

Task B

Improvement 1 - Handled actions happening while waiting for asynchronous calls
As soon as the equals button is clicked CSS styles are added to the actionable buttons to prevent users from clicking them until the response for the currently sent request is received. This stops multiple requests being sent at once which could result in incorrect answers. It also stops requests being blocked by too many requests in transit at once. Below shows the calculator paused while I request is in transit, as well as some of the logic required to do this.





```
function DisableButtons() {
            var element = document.getElementById('loading-text');
            element.style.display = 'block';
            var operationRows =
document.getElementsByClassName('operation-row');
            for (var i = 0, all = operationRows.length; i < all; i++) {
                operationRows[i].classList.add('disabledbutton');
            }
```

```
        }

        function EnableButtons() {
            var element = document.getElementById('loading-text');
            element.style.display = 'none';
            var operationRows =
document.getElementsByClassName('operation-row');
            for (var i = 0, all = operationRows.length; i < all; i++) {
                operationRows[i].classList.remove('disabledbutton');
            }
        }
```

```
function Equals() {
            DisableButtons();
            HideErrorMessage();

            if (operation == ''){
                EnableButtons();
                return;
            }


            y = value;

            CalculateOperationResult();
        }
```

```
function HandleResponseResult(responseData, responseStatusCode) {
            if (responseStatusCode == 200 &&
IsValidResponse(responseData)) {
                var j = responseData;
                x = 0;
                y = 0;
                operation = '';
                value = j.answer;
                Display();
                EnableButtons();
            }
            else if (responseStatusCode == 400) {
                var j = responseData;
                x = 0;
                y = 0;
                operation = '';
                value = 0;

                ShowErrorMessage('Bad Request - ' + j.string);
                Display();
                EnableButtons();
            }
            else {
                x = 0;
                y = 0;
                operation = '';
                value = 0;

                ShowErrorMessage('Unexpected Exception Occurred');
                Display();
                EnableButtons();
            }
        }
```

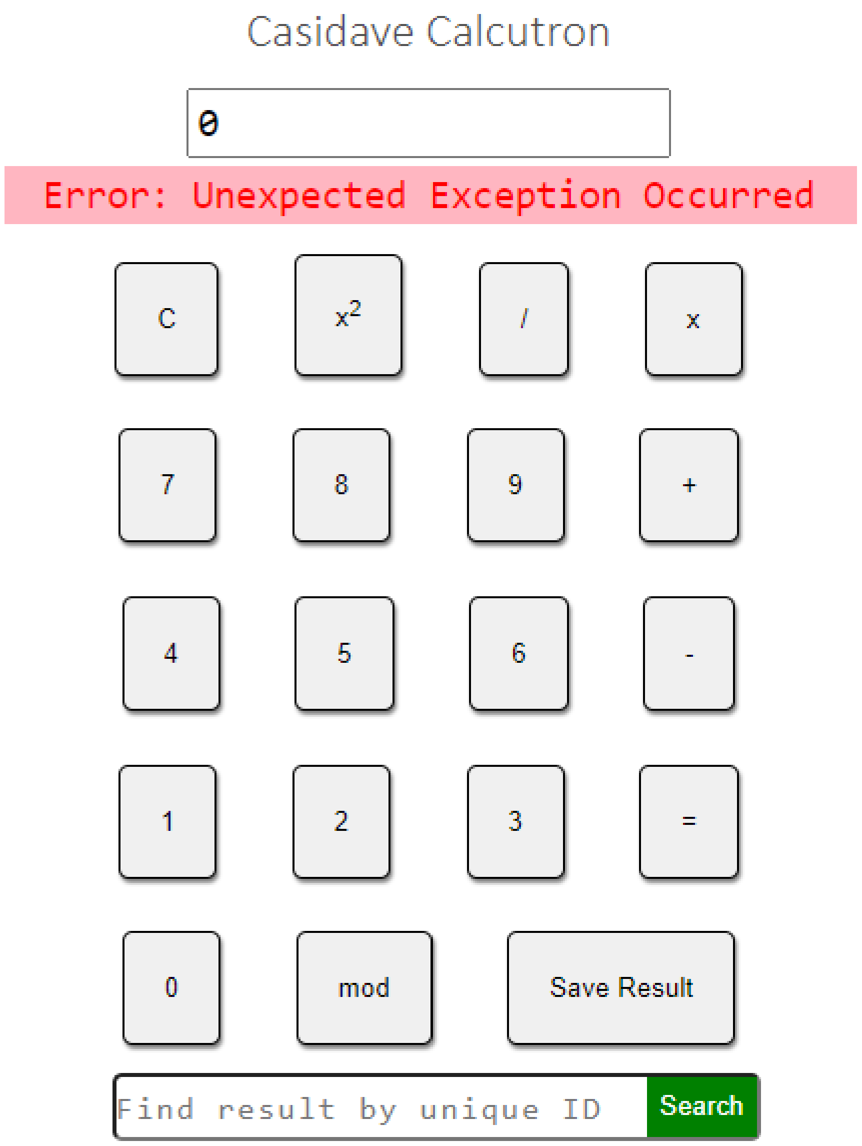
Improvement 2 - Frontend Error Handling
Added error handling to the frontend. When requests are returned the code first checks that the json returned is not null and has the relevant valid fields. Errors can be detected by either response no returning a 400 status code, or by being caught in the axios ‘catch’ call-back. If some error has occurred, it 's handled by setting ‘value’ to 0 and then displaying an appropriate error message to the user. Below are some code snippets or relevant frontend logic as well as a screenshot of what is shown to the user when an error occurs. In the case of the screenshot I have taken down the workload and this is what is causing the error.

```
async function CalculateOperationResult() {
            var finalResponseData;
            var finalResponseCode;

            var requestSuccessful = false;
            while (availableUrls.length > 0 && !requestSuccessful) {
                await 
axios.get(`${availableUrls[0]}?x=${x}&y=${y}&operator=${operation}`).th
en(response => {
                    finalResponseData = response.data;
                    finalResponseCode = response.status;
                }).catch(error => {
                    var test = error;
                    if (error.response) {
                        finalResponseData = error.response.data;
                        finalResponseCode = error.response.code;
                    } else {
                        finfinalResponseCode = 502;
                        finalResponseData = {
                            'error': true,
                            'string': error,
                            'answer': 0
                        }
                    }
                });

                requestSuccessful = finalResponseCode == 200;

                if (!requestSuccessful) {
                    failedUrls.push(availableUrls[0]);
                    availableUrls.shift();
                }
            }

            HandleResponseResult(finalResponseData, finalResponseCode);

        }
```

```
function HandleResponseResult(responseData, responseStatusCode) {
            if (responseStatusCode == 200 && 
IsValidResponse(responseData)) {
```

```
                var j = responseData;
                x = 0;
                y = 0;
                operation = '';
                value = j.answer;
                Display();
                EnableButtons();
            }
            else if (responseStatusCode == 400) {
                var j = responseData;
                x = 0;
                y = 0;
                operation = '';
                value = 0;

                ShowErrorMessage('Bad Request - ' + j.string);
                Display();
                EnableButtons();
            }
            else {
                x = 0;
                y = 0;
                operation = '';
                value = 0;

                ShowErrorMessage('Unexpected Exception Occured');
                Display();
                EnableButtons();
            }
        }
```

```
function IsValidResponse(response) {
            if (response == null) {
                return false;
            }
            else if (response.answer == null || response.string == null
|| response.error == null) {
                return false;
            }

            return true;
        }
```


Error: Unexpected Exception Occurred

Improvement 3 - Error Handling added to both backend services
Made relevant changes to the backend services so the input parameters are validated, checking inputs are not null, numeric and are whole numbers, and relevant error strings are appended to the response when necessary. A response code of 400 is also returned when an error does occur as a result of invalid parameters. Code snippets are included below of how validation is done in the two backend services as well as how it handles invalid input.

Add Service:
```
function isInteger($input){
    if(filter_var($input, FILTER_VALIDATE_INT) !== false){
        return true;
      }else{
          return false;
      }
}
```

```php
function validateInputs($x, $y){
    $output = array(
        "error" => false,
        "string" => "",
        "answer" => 0
    );

    if ($x == null || $y == null){
        $output = array(
            "error" => true,
            "string" => "One or both required parameters (x and y) are
missing.",
            "answer" => 0
        );
    }
    elseif(!is_numeric($x) || !is_numeric($y)){
        $output = array(
            "error" => true,
            "string" => "One or both required parameters (x and y) are
non numeric.",
            "answer" => 0
        );
    }
    elseif(!isInteger($x) || !isInteger($y)) {
        $output = array(
            "error" => true,
            "string" => "One or both required parameters (x and y) are
not whole numbers.",
            "answer" => 0
        );
    }

    return $output;
}
```

```php
$statusCode = 200;

    $jsonResponse = array(
        "error" => false,
        "string" => "",
        "answer" => 0
    );

    $x = $_REQUEST['x'];
    $y = $_REQUEST['y'];

    $jsonResponse = validateInputs($x, $y);

    if ($jsonResponse['error']){
        $statusCode = 400;
    }else{
        $answer=add($x,$y);

        $jsonResponse['string']=$x."+".$y."=".$answer;
        $jsonResponse['answer']=$answer;
    }
    if ($statusCode == 200){
        header("HTTP/1.1 200 OK");
    }else {
        header("HTTP/1.1 400 Bad Request");
    }

    echo json_encode($jsonResponse);
```

Subtract Service:

```
validate: function (x, y) {
        // print(typeof(x))

        var output = {
            'error': false,
            'string': '',
            'answer': 0
        };

        if (x == null || y == null) {
            var output = {
                'error': true,
                'string': 'One or both required parameters (x and y) 
are missing.',
                'answer': 0
            };
        }
        else if (isNaN(x) || isNaN(y)) {
            var output = {
                'error': true,
                'string': 'One or both required parameters (x and y) 
are non numeric.',
                'answer': 0
            };
        }
        else if (!Number.isInteger(parseFloat(x))  || 
!Number.isInteger(parseFloat(y))){
            var output = {
                'error': true,
                'string': 'One or both required parameters (x and y) 
are not whole numbers.',
                'answer': 0
            };
        }


        return output;
    }
```

```
app.get('/', (req, res) => {

    var jsonResponse = {
        'error': false,
        'string': '',
        'answer': 0
    };
    var x = req.query.x;
    var y = req.query.y;

    var jsonResponse = validation.validate(x, y);

    if (jsonResponse.error) {
        res.statusCode = 400;
    }
    else {
        res.statusCode = 200;
        var answer = sub.subtract(x, y);

        jsonResponse.string = x + '-' + y + '=' + answer;
        jsonResponse.answer = answer;
    }

    res.setHeader('Content-Type', 'application/json');
    res.setHeader('Access-Control-Allow-Origin', '*')

    res.end(JSON.stringify(jsonResponse));
});
```

Improvement 4 - Configure Routes through .json file
Routes can be configured by modifying the appsettings.json file instead of being stored as static variables in the index.html file. Below are code snippets of the appsettings.json file used to configured the URLs as well as the file being loaded in and used.

```
{
    "proxyUrls": [
        "http://webcalc-proxy.40204578.qpc.hal.davecutting.uk/",
      "http://webcalc-proxy-backup1.40204578.qpc.hal.davecutting.uk/",
        "http://webcalc-proxy-backup2.40204578.qpc.hal.davecutting.uk/"
    ]
}
```

```
let appsettings = {}

        let availableUrls = [];
        let failedUrls = [];
        init()

        function init() {
            console.log('In Init');
            loadJSON(function (response) {
                appsettings = JSON.parse(response);
                availableUrls = appsettings['proxyUrls'];
            });

            setInterval(ReinsertFailedUrls, 10000);
        }

        function loadJSON(callback) {

            var xobj = new XMLHttpRequest();
            xobj.overrideMimeType("application/json");
            xobj.open('GET', 'appsettings.json', true);
            xobj.onreadystatechange = function () {
                if (xobj.readyState == 4 && xobj.status == "200") {
                    callback(xobj.responseText);
                }
            };
            xobj.send(null);
        }
```

Improvement 5 - Integration Tests
Added integration tests to each backend service so that the actual endpoints are tested and not just small units of code. In each backend service three aspects of the API were tested in isolation, the response code, the response body and the content type of the response. For each aspect previously mentioned there were three valid and three invalid tests written. In both services all 18 integration tests that were run passed. I also expanded the unit testing, adding tests which use a mixture of positive and negative paraments. Below are some code snippets showing the integration tests that were run.

Add Service:

```
public function test_InvalidInputs_ReturnsCorrectBody(){
        $response1 = $this->http->request('GET', '/', ['http_errors' =>
false]);
        $response2 = $this->http->request('GET', '/?x=foo&y=bar',
['http_errors' => false]);
        $response3 = $this->http->request('GET', '/?x=4.2&y=2.2',
['http_errors' => false]);

        $this->assertEquals('{"error":true,"string":"One or both
required parameters (x and y) are missing.","answer":0}', $response1-
>getBody()->getContents());
        $this->assertEquals('{"error":true,"string":"One or both
required parameters (x and y) are non numeric.","answer":0}',
$response2->getBody()->getContents());
        $this->assertEquals('{"error":true,"string":"One or both
required parameters (x and y) are not whole numbers.","answer":0}',
$response3->getBody()->getContents());
    }
```

```
public function test_ValidInputs_Returns200()
    {
        $response1 = $this->http->request('GET', '/?x=4&y=2',
['http_errors' => false]);
        $response2 = $this->http->request('GET', '/?x=-4&y=-2',
['http_errors' => false]);
        $response3 = $this->http->request('GET', '/?x=4&y=-2',
['http_errors' => false]);



        $this->assertEquals(200, $response1->getStatusCode());
        $this->assertEquals(200, $response2->getStatusCode());
        $this->assertEquals(200, $response3->getStatusCode());
    }
```

Subtract Service:

```
describe("valid data returns correct content type", function () {
    it("two positives", function (done) {
        request(server)
        .get('/?x=2&y=2')
        .expect('Content-Type', 'application/json', done);
    });

    it("two negatives", function (done) {
        request(server)
        .get('/?x=-2&y=-2')
        .expect('Content-Type', 'application/json', done);
    });

    it("one positive one negative", function (done) {
        request(server)
        .get('/?x=2&y=-2')
        .expect('Content-Type', 'application/json', done);
    });
});
```

```
describe("invalid data returns correct response code", function () {
    it("no data", function (done) {
        request(server).get("/")
            .expect(400, done)
    })

    it("non numeric data", function (done) {
        request(server).get("/?x=foo&y=bar")
            .expect(400, done)
    })

    it("decimal numbers data", function (done) {
        request(server).get("/?x=4.2&y=3.2")
            .expect(400, done)
    })
})
```

Improvement 6 - Routing functionality in the add service
(reference: https://www.youtube.com/watch?v=6reEBParHzQ)
The current implementation of the add service made it difficult to add new routes to the service. I implemented a Route.php class to allow routes to be added to the service easily, which would later be used to add a new endpoint for service discovery. Below is a code snippet of the .htaccess file which was added and the new class created as well as it being used to add a new route.

```
RewriteEngine On
RewriteBase /
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d

RewriteRule ^(.+)$ index.php?uri=$1 [QSA,L]
```

```
class Route {
```

```php
    private $_uri = array();
    private $_method = array();

    public function add($uri, $method){
     $this->_uri[] = '/' . trim($uri, '/');

     if ($method != null) {
         $this->_method[] = $method;
     }
    }

    public function submit(){

       $uriGetParam = isset($_GET['uri']) ? '/' . $_GET['uri'] : '/';
       foreach( $this->_uri as $key => $value){
           if (preg_match("#^$value$#", $uriGetParam)) {
               call_user_func($this->_method[$key]);
           }
       }
    }
}
```

```php
$route->add('discovery',function(){
    $jsonResponse = array(
        "operator" => "add",
    );

    header("HTTP/1.1 200 OK");

    echo json_encode($jsonResponse);
});
```

Task C

For the challenges you have chosen complete one section for each.

Challenge One

Challenge Code and Title: Custom Proxy Router

Implementation Details (how you implemented):

I created a new service using Node.js and express. On start-up the proxy reads in URLs from the url-config.json file and sends a request to each URL's 'discovery' endpoint, the response it receives is then used to create an object map of operators (the key) to URLs (the value) based on what is returned from the service, named operatorMaps, using this the proxy now knows what each service does. Included below are the discovery requests being sent, the url-config.json contents and an example of one of the 'discovery' endpoints which exist in all six services.

```
async function setUpUrls() {
    urls = require('./url-config.json');

    for (var i = 0; i < urls.length; i++) {
        var url = urls[i];
        await sendDiscoveryRequest(url);
    }
}

async function sendDiscoveryRequest(url) {
    await axios.get(`${url}discovery`)
        .then(response => {
            var operator = response.data.operator;
            operatorUrls[`${operator}`] = url;

        })
        .catch(error => {
            if (error.response.status) {
                console.log(`URL ${url} return
${error.response.status}`);
            }
        });
}
```

```
$route->add('discovery',function(){
    $jsonResponse = array(
        "operator" => "add",
    );

    header("HTTP/1.1 200 OK");

    echo json_encode($jsonResponse);
});
```

The proxy URLs can be configured during runtime by using two different endpoints '/update' and '/remove'. If visited both endpoints will return a list of each currently configured operator and its service URL. If a value is passed to the query parameter 'newUrl' in the /update endpoint, it will perform service discovery to identify its operation, if the operator already exists, it will be replaced, otherwise a new operator will be mapped, provided the URL +has a valid 'discovery' endpoint. If a value is passed to the query parameter 'operator' in the /remove endpoint, the operator will be looked up in the operatorUrl map and that service will be removed. A code snippet of both endpoints is displayed below.l

```
app.get('/update', (req, res) => {
    res.setHeader('Content-Type', 'application/json');
    res.setHeader('Access-Control-Allow-Origin', '*')

    var url = req.query.url;

    if (url != null) {
        sendDiscoveryRequest(url).then(_ => {
            res.end('Operator URLs: \n\n' +
JSON.stringify(operatorUrls));
        });
    }
    else {
        res.end('Operator URLs: \n\n' + JSON.stringify(operatorUrls));
    }
});

app.get('/remove', (req, res) => {
    res.setHeader('Content-Type', 'application/json');
    res.setHeader('Access-Control-Allow-Origin', '*')

    var operator = req.query.operator;

    if (operator != null) {
        delete operatorUrls[`${operator}`];
    }

    res.end('Updated URLs: \n' + JSON.stringify(operatorUrls));

});
```

The proxy default route accepts three parameters, x, y and operator. The operator variable is validated to ensure that it is a valid operation which it is trying to perform, any validation on x and y is left to the individual function services. It's worth noting that the 'acceptedOperators' variable set below should actually use the operatorUrl map which was built be the service discovery, instead of heard coding valid operators.

```
checkOperatorIsValid: function (operator) {
        // should check based on what is in json
        var acceptedOperators = ['add', 'sub', 'mul', 'div', 'mod',
'sqr',];

        return acceptedOperators.includes(operator);
    },
```

If the operator passed to the proxy is valid then the operatorUrl map mentioned above is then used to select a URL which will be used to send a request to one of the function backend services with the x and y parameters included in the query string. When the service responds, the proxy then forwards the result to the frontend.

```
buildUrl: function (x, y, operator, urlsConfig) {
        var url = urlsConfig[operator];

        url += `?x=${x}`;
        if (operator != '^2') {
            url += `&y=${y}`;
        }

        return url;
    },
```

```
app.get('/', (req, res) => {
    res.setHeader('Content-Type', 'application/json');
    res.setHeader('Access-Control-Allow-Origin', '*')

    var x = req.query.x;
    var y = req.query.y;
    var operator = req.query.operator;

    // If not do something
    var isOperatorValid = functions.checkOperatorIsValid(operator);

    if (!isOperatorValid) {
        res.end(JSON.stringify(
            {
                error: true,
                string: "No or invalid operator provided",
                answer: 0,
            }));
    } else {
        var url = functions.buildUrl(x, y, operator, operatorUrls);

        //Get Operation Answer
        axios.get(url)
            .then(response => {
                res.statusCode = response.status;

                res.end(JSON.stringify(response.data));
            })
            .catch(error => {
                if (error.response) {
                    res.statusCode = error.response.status;
                    res.end(JSON.stringify(error.response.data));
                } else {
                    res.end(JSON.stringify({
                        error: true,
                        string: "An unexpected error occurred",
                        answer: 0,
                    }));
                }
```

```
                    });
            }
});
```

Testing Details (how you tested):
Most testing was initially done using postman or by manually visiting the URLs. When I was more confident that everything in the proxy worked I tested using the actionable buttons on the front end which actually triggered the requests to the proxy router.

When testing service discovery I open Google Chrome dev tools to see all requests were being send and returning a 200 response and then logged out the mapping that was built to the console to ensure everything was mapped appropriately.

I sent requests to the proxy from the frontend using all the actionable buttons to ensure all calculations worked as well as testing it with non-existent operators to ensure errors such as this are handled correctly.

I tested requests such as: http://webcalc-proxy.40204578.qpc.hal.davecutting.uk/update?url=http://webcalc-add-backup.40204578.qpc.hal.davecutting.uk/ which successfully remapped the 'add' URL and http://webcalc-proxy.40204578.qpc.hal.davecutting.uk/remove?operator=add which successfully remove the add map all together

Brief Review of Success (did it work, would you do it the same again [very brief!]):

All intended functionality works well, service discovery is successfully implemented to find the URLs operator on start-up, URLs can be added, edited and deleted during runtime and the proxy successfully routes all requests to the appropriate service.

Anything else to highlight (optional):

Appropriate changes were made to the frontend to integrate these changes such as updating the URL config to only include the proxy URL instead of the six individual services as routing all requests through one single request which just took three parameters x, y and operator.

Challenge Two

Challenge Code and Title: Frontend service failure handler

Implementation Details (how you implemented):

I created two backup proxy ingress controllers on QPC to allow the calculator to fall back on these if the original URL failed for some reason. When sending requests, I made sure to retry the request on two conditions if it hasn't returned a 200 response and if there are still URLs to fall back on. So if a request fails, the URL used will be added to an array of ‘failedUrls’ and removed from circulation, the request will then be tried again with one of the remaining URLs until there are no URLs left, at which point an error message will be displayed to the user. A code snippet below displays an example of a request being sent and how failure of the rest would be handled.

```
async function CalculateOperationResult() {
            var finalResponseData;
            var finalResponseCode;

            var requestSuccessful = false;
            while (availableUrls.length > 0 && !requestSuccessful) {
                await
axios.get(`${availableUrls[0]}?x=${x}&y=${y}&operator=${operation}`).th
en(response => {
                    finalResponseData = response.data;
                    finalResponseCode = response.status;
                }).catch(error => {
                    var test = error;
                    if (error.response) {
                        finalResponseData = error.response.data;
                        finalResponseCode = error.response.code;
                    } else {
                        finfinalResponseCode = 502;
                        finalResponseData = {
                            'error': true,
                            'string': error,
                            'answer': 0
                        }
                    }
                });

                requestSuccessful = finalResponseCode == 200;

                if (!requestSuccessful) {
                    failedUrls.push(availableUrls[0]);
                    availableUrls.shift();
                }
            }

            HandleResponseResult(finalResponseData, finalResponseCode);

        }
```

A background task is set using setInternal to run a function every 10 second that will retry any failed URLs with some hard coded parameters so they can be used for future requests, if a 200 response is received from the mock request, they will be removed from failedUrls list and are added to the available URLs list. A code snippet of the logic required for this is included below.

```
let failedUrls = [];
        init()

        function init() {
            console.log('In Init');
            loadJSON(function (response) {
                appsettings = JSON.parse(response);
                availableUrls = appsettings['proxyUrls'];
            });

            setInterval(ReinsertFailedUrls, 10000);
        }
```

```
async function ReinsertFailedUrls() {
            if (failedUrls.length == 0) return;

            var test_x = 2
            var test_y = 3
            var test_operator = 'add';

            var responseCode;

            var urlsToRemoveFromFailed = [];

            for (var i = 0; i < failedUrls.length; i++) {
                await axios.get(`${failedUrls[i]}?x=${test_x}&y=${test_y}&operator=${test_operator}`).then(response => {
                    responseCode = response.status;
                }).catch(error => {
                    var test = error;
                    if (error.response) {
                        responseCode = error.response.code;
                    } else {
                        responseCode = 502;
                    }
                });

                if (responseCode == 200) {
                    availableUrls.push(failedUrls[i]);
                    urlsToRemoveFromFailed.push(failedUrls[i]);
                }
            }

            for(var i = 0; i < urlsToRemoveFromFailed.length; i++){
                var index = failedUrls.indexOf(urlsToRemoveFromFailed[i]);
                if (index !== -1){
                    failedUrls.splice(index, 1);
                }
            }
        }
```

Testing Details (how you tested):

To test that the URLs were being removed I exposed incorrect ports on the each ingress controller on QPC, when trying to send a request an error message would then be displayed.

I then opened the Google Chrome dev tools to observe that every 10 seconds, each broken URL was being used to send a request to check if the URL was working again. I would then fix one of the ingress controllers port, observe that its ‘retry request’ returned a 200 using the Google Chrome dev tools and if so I would try to do another calculation which would then use the reintroduced URL and successfully complete the operation and display the result.

Brief Review of Success (did it work, would you do it the same again [very brief!]):

I think the functionality works well, URLs are used and removed from circulation appropriately and reintroduced when ready. In terms of the quality of the code, it is not tidy or practical to have the same for loop every time a request needs to be called, in the future I’d like to extract the logic of the retry in a way that would make it more reusable. I have included code snippets below of every time the failure handler logic is used to illustrate how it might be perceived as repetitive and untidy code.

```
async function GetResultById() {
            var id = document.getElementById("find-by-id").value;

            var requestSuccessful = false;
            while (availableUrls.length > 0 && !requestSuccessful) {
                var responseCode;

                var url = `${availableUrls[0]}read/?id=${id}`;

                await axios.get(url)
                    .then(function (response) {
                        responseCode = response.status;
                        ShowSuccessMessage(`Successfully read result
from the Database. \nValue: ${response.data.result}`);
                    })
                    .catch(function (error) {
                        if (error.resonse) {
                            responseCode = error.response.code;
                        }
                        else {
                            responseCode = 503;
                        }
                        ShowErrorMessage(error);
                    });

                requestSuccessful = responseCode == 200;

                if (!requestSuccessful) {
                    failedUrls.push(availableUrls[0]);
                    availableUrls.shift();
                }
            }
        }
```

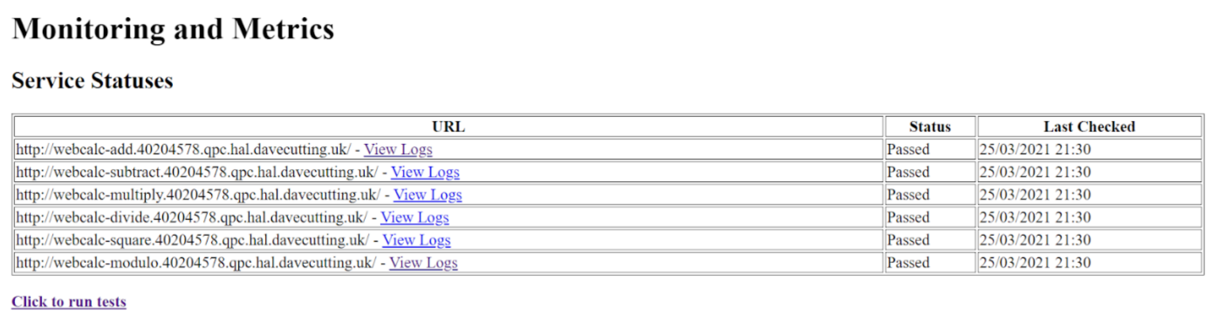
```
async function SaveResult() {
            var data = {
                result: value
            };
            var headers = { "Access-Control-Allow-Origin": "*" };

            var requestSuccessful = false;
            while (availableUrls.length > 0 && !requestSuccessful) {
                var responseCode;

                await axios.post(`${availableUrls[0]}write`, data)
                    .then(function (response) {
                        responseCode = response.status;
                        ShowSuccessMessage(`Successfully saved result 
to the Database.\nUnique ID: ${response.data.key}`);
                    })
                    .catch(function (error) {
                        if (error.response) {
                            responseCode = error.response.code;
                        }
                        else {
                            responseCode = 503;
                        }
                        ShowErrorMessage(error);
                    });

                requestSuccessful = responseCode == 200;

                if (!requestSuccessful) {
                    failedUrls.push(availableUrls[0]);
                    availableUrls.shift();
                }
            }
        }
```

Anything else to highlight (optional):

Task D

Repository URL: http://gitlab.hal.davecutting.uk/40204578/webcalc-monitoring-and-metrics
Live Service Link: http://webcalc-monitoring-and-metrics.40204578.qpc.hal.davecutting.uk/

I created a new service using Node.js to monitor and test each function service. Visiting the link above will display a table detailing the status of each function service, as well as when the last time it was tested. It is configured to run tests every half an hour, but the service also includes a link which can be clicked to run the tests manually.



Monitoring and Metrics

Service Statuses

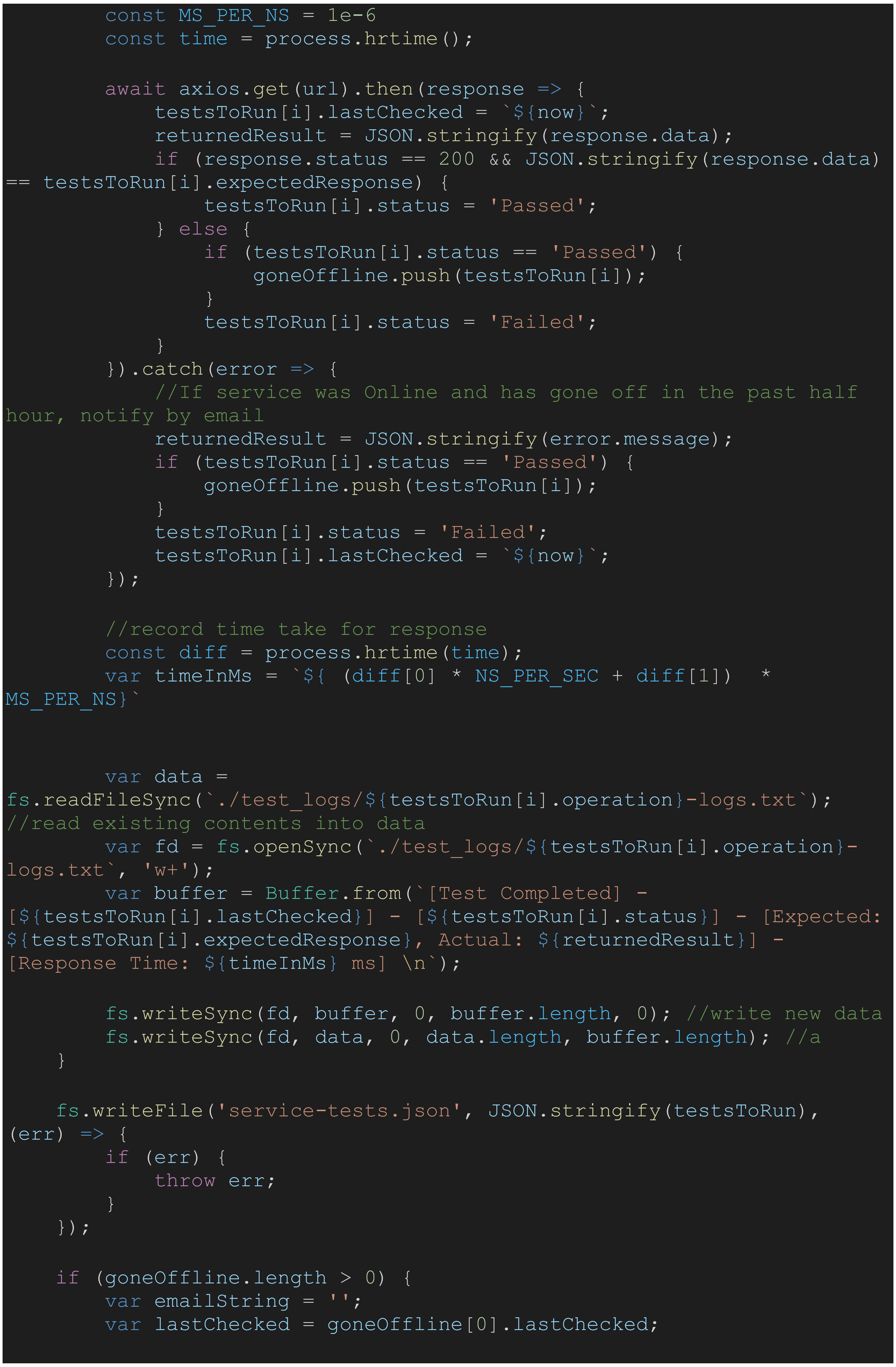
URL	Status	Last Checked
http://webcalc-add.40204578.qpc.hal.davecutting.uk/ - View Logs	Passed	25/03/2021 21:30
http://webcalc-subtract.40204578.qpc.hal.davecutting.uk/ - View Logs	Passed	25/03/2021 21:30
http://webcalc-multiply.40204578.qpc.hal.davecutting.uk/ - View Logs	Passed	25/03/2021 21:30
http://webcalc-divide.40204578.qpc.hal.davecutting.uk/ - View Logs	Passed	25/03/2021 21:30
http://webcalc-square.40204578.qpc.hal.davecutting.uk/ - View Logs	Passed	25/03/2021 21:30
http://webcalc-modulo.40204578.qpc.hal.davecutting.uk/ - View Logs	Passed	25/03/2021 21:30

Click to run tests

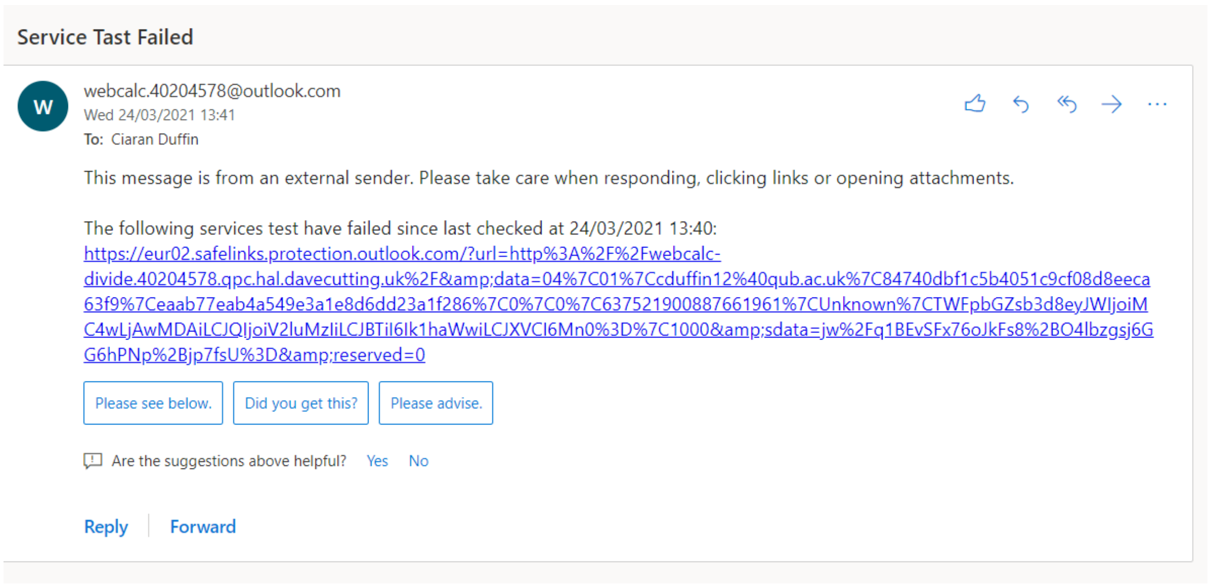
```
//Schedule tests to run every half hour on the 0th minute
const onlineCheck = schedule.scheduleJob('*/30 * * * *', async function
() {
    runTests();
});
```

To run the tests a .json file is loaded in which contains details about each URL as well as some test data and an expected response, if the response obtained from sending a request to the service URL defined matched the expected response the test will be marked as passed and this will be saved to the .json file, all tests are also timestamped. If a response however is not equal to the expected response, then the status of the URL is set it failed and an email is sent, using node mailer package, to cduffin12@qub.ac.uk notifying me of any services that have failed. Below includes a code snippet of the main test function which shows how tests are run, results are validated and written to logs as well as sending alert emails where necessary, I have also included an example of the email which is sent when tests fail.

```
const runTests = async function(){
    var testsToRun = require('./service-tests.json');

    var goneOffline = [];
    var now = moment().format('DD/MM/YYYY HH:mm');

    for (var i = 0; i < testsToRun.length; i++) {
        var url =
`${testsToRun[i].url}?x=${testsToRun[i].x}&y=${testsToRun[i].y}`;

        var returnedResult;

        //start performance timer
        const NS_PER_SEC = 1e9;
```



```
        const MS_PER_NS = 1e-6
        const time = process.hrtime();

        await axios.get(url).then(response => {
            testsToRun[i].lastChecked = `${now}`;
            returnedResult = JSON.stringify(response.data);
            if (response.status == 200 && JSON.stringify(response.data)
== testsToRun[i].expectedResponse) {
                testsToRun[i].status = 'Passed';
            } else {
                if (testsToRun[i].status == 'Passed') {
                    goneOffline.push(testsToRun[i]);
                }
                testsToRun[i].status = 'Failed';
            }
        }).catch(error => {
            //If service was Online and has gone off in the past half
hour, notify by email
            returnedResult = JSON.stringify(error.message);
            if (testsToRun[i].status == 'Passed') {
                goneOffline.push(testsToRun[i]);
            }
            testsToRun[i].status = 'Failed';
            testsToRun[i].lastChecked = `${now}`;
        });

        //record time take for response
        const diff = process.hrtime(time);
        var timeInMs = `${ (diff[0] * NS_PER_SEC + diff[1])  *
MS_PER_NS}`


        var data =
fs.readFileSync(`./test_logs/${testsToRun[i].operation}-logs.txt`);
//read existing contents into data
        var fd = fs.openSync(`./test_logs/${testsToRun[i].operation}-
logs.txt`, 'w+');
        var buffer = Buffer.from(`[Test Completed] -
[${testsToRun[i].lastChecked}] - [${testsToRun[i].status}] - [Expected:
${testsToRun[i].expectedResponse}, Actual: ${returnedResult}] -
[Response Time: ${timeInMs} ms] \n`);

        fs.writeSync(fd, buffer, 0, buffer.length, 0); //write new data
        fs.writeSync(fd, data, 0, data.length, buffer.length); //a
    }

    fs.writeFile('service-tests.json', JSON.stringify(testsToRun),
(err) => {
        if (err) {
            throw err;
        }
    });

    if (goneOffline.length > 0) {
        var emailString = '';
        var lastChecked = goneOffline[0].lastChecked;
        goneOffline.forEach(element => {
            emailString += element.url +'\n';
        });

        try {
            mailer.sendServiceOfflineEmail(emailString, lastChecked);
        }catch(error){
            console.log('Email failed to send');
        }
    }
}
```



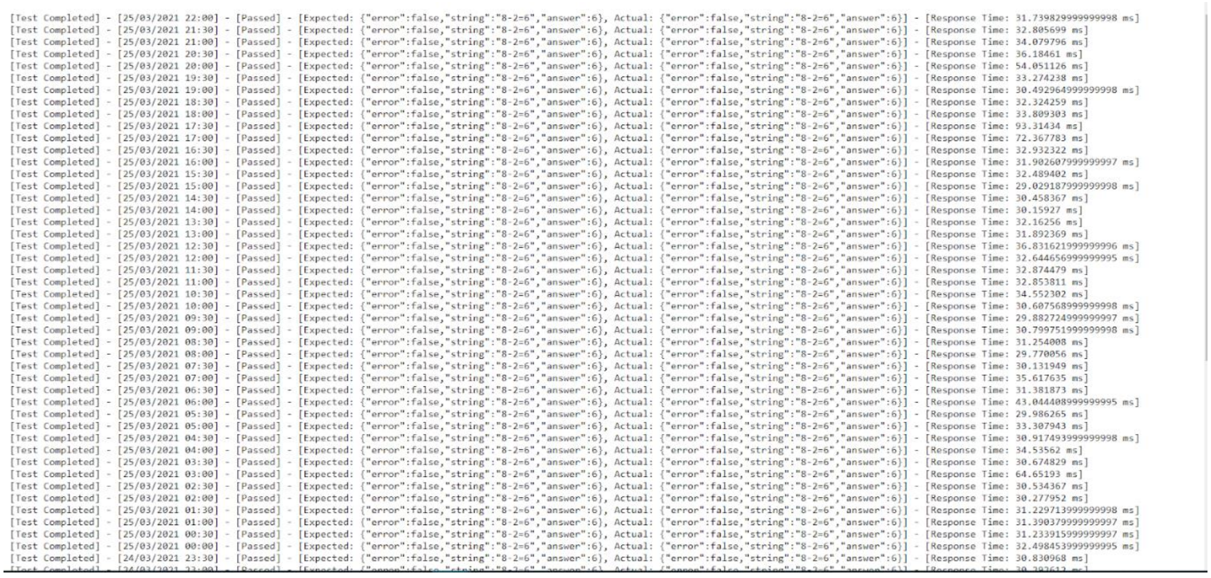
Service Tast Failed

webcalc.40204578@outlook.com
Wed 24/03/2021 13:41
To: Ciaran Duffin

This message is from an external sender. Please take care when responding, clicking links or opening attachments.

The following services test have failed since last checked at 24/03/2021 13:40:
https://eur02.safelinks.protection.outlook.com/?url=http%3A%2F%2Fwebcalc-divide.40204578.qpc.hal.davecutting.uk%2F&amp;data=04%7C01%7Ccduffin12%40qub.ac.uk%7C84740dbf1c5b4051c9cf08d8eeca63f9%7Ceaab77eab4a549e3a1e8d6dd23a1f286%7C0%7C0%7C637521900887661961%7CUnknown%7CTWFpbGZsb3d8eyJWIjoiMC4wLjAwMDAiLCJQIjoiV2luMzIiLCJBTiI6Ik1haWwiLCJXVCI6Mn0%3D%7C1000&amp;sdata=jw%2Fq1BEvSFx76oJkFs8%2BO4lbzgsj6GG6hPNp%2Bjp7fsU%3D&amp;reserved=0

Users can monitor the performance of tests over time, every time a test is run for a service the details of the test, including the time taken for the test to complete are stored in a .txt file, the contents of which can be viewed by clicking the view logs link beside the URL. Each test log is timestamped, with the most recent tests at the top, so the user can get a clear idea of the performance of the service over time.

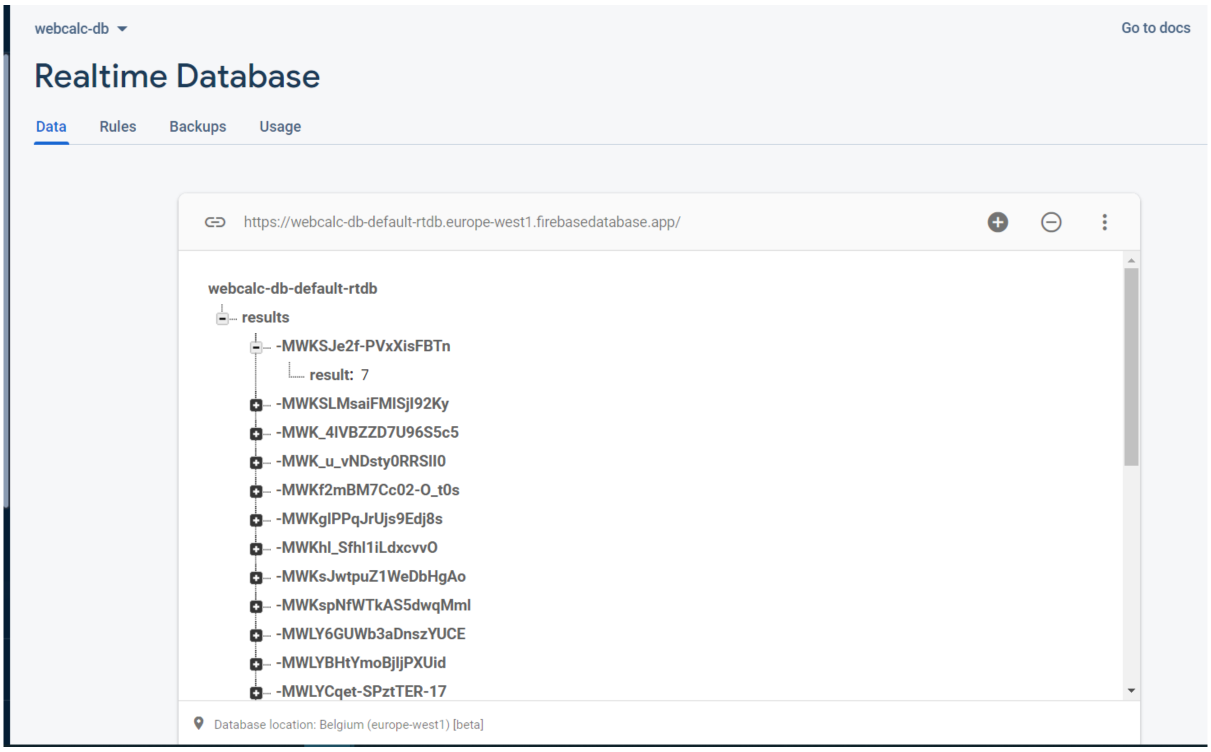


```
[Test Completed] - [25/03/2021 22:00] - [Passed] - [Expected: {"error":false,"string":"8-2=6","answer":6}, Actual: {"error":false,"string":"8-2=6","answer":6}] - [Response Time: 31.739829999999998 ms]
[Test Completed] - [25/03/2021 21:30] - [Passed] - [Expected: {"error":false,"string":"8-2=6","answer":6}, Actual: {"error":false,"string":"8-2=6","answer":6}] - [Response Time: 32.805699 ms]
[Test Completed] - [25/03/2021 21:00] - [Passed] - [Expected: {"error":false,"string":"8-2=6","answer":6}, Actual: {"error":false,"string":"8-2=6","answer":6}] - [Response Time: 34.079796 ms]
[Test Completed] - [25/03/2021 20:30] - [Passed] - [Expected: {"error":false,"string":"8-2=6","answer":6}, Actual: {"error":false,"string":"8-2=6","answer":6}] - [Response Time: 36.18461 ms]
[Test Completed] - [25/03/2021 20:00] - [Passed] - [Expected: {"error":false,"string":"8-2=6","answer":6}, Actual: {"error":false,"string":"8-2=6","answer":6}] - [Response Time: 54.051126 ms]
[Test Completed] - [25/03/2021 19:30] - [Passed] - [Expected: {"error":false,"string":"8-2=6","answer":6}, Actual: {"error":false,"string":"8-2=6","answer":6}] - [Response Time: 33.274238 ms]
[Test Completed] - [25/03/2021 19:00] - [Passed] - [Expected: {"error":false,"string":"8-2=6","answer":6}, Actual: {"error":false,"string":"8-2=6","answer":6}] - [Response Time: 30.492964999999998 ms]
[Test Completed] - [25/03/2021 18:30] - [Passed] - [Expected: {"error":false,"string":"8-2=6","answer":6}, Actual: {"error":false,"string":"8-2=6","answer":6}] - [Response Time: 32.324259 ms]
[Test Completed] - [25/03/2021 18:00] - [Passed] - [Expected: {"error":false,"string":"8-2=6","answer":6}, Actual: {"error":false,"string":"8-2=6","answer":6}] - [Response Time: 33.809303 ms]
[Test Completed] - [25/03/2021 17:30] - [Passed] - [Expected: {"error":false,"string":"8-2=6","answer":6}, Actual: {"error":false,"string":"8-2=6","answer":6}] - [Response Time: 93.31434 ms]
[Test Completed] - [25/03/2021 17:00] - [Passed] - [Expected: {"error":false,"string":"8-2=6","answer":6}, Actual: {"error":false,"string":"8-2=6","answer":6}] - [Response Time: 72.367783 ms]
[Test Completed] - [25/03/2021 16:30] - [Passed] - [Expected: {"error":false,"string":"8-2=6","answer":6}, Actual: {"error":false,"string":"8-2=6","answer":6}] - [Response Time: 32.932322 ms]
[Test Completed] - [25/03/2021 16:00] - [Passed] - [Expected: {"error":false,"string":"8-2=6","answer":6}, Actual: {"error":false,"string":"8-2=6","answer":6}] - [Response Time: 31.902607999999997 ms]
[Test Completed] - [25/03/2021 15:30] - [Passed] - [Expected: {"error":false,"string":"8-2=6","answer":6}, Actual: {"error":false,"string":"8-2=6","answer":6}] - [Response Time: 32.489402 ms]
[Test Completed] - [25/03/2021 15:00] - [Passed] - [Expected: {"error":false,"string":"8-2=6","answer":6}, Actual: {"error":false,"string":"8-2=6","answer":6}] - [Response Time: 29.029187999999998 ms]
[Test Completed] - [25/03/2021 14:30] - [Passed] - [Expected: {"error":false,"string":"8-2=6","answer":6}, Actual: {"error":false,"string":"8-2=6","answer":6}] - [Response Time: 30.458367 ms]
[Test Completed] - [25/03/2021 14:00] - [Passed] - [Expected: {"error":false,"string":"8-2=6","answer":6}, Actual: {"error":false,"string":"8-2=6","answer":6}] - [Response Time: 30.15927 ms]
[Test Completed] - [25/03/2021 13:30] - [Passed] - [Expected: {"error":false,"string":"8-2=6","answer":6}, Actual: {"error":false,"string":"8-2=6","answer":6}] - [Response Time: 32.16256 ms]
[Test Completed] - [25/03/2021 13:00] - [Passed] - [Expected: {"error":false,"string":"8-2=6","answer":6}, Actual: {"error":false,"string":"8-2=6","answer":6}] - [Response Time: 31.892369 ms]
[Test Completed] - [25/03/2021 12:30] - [Passed] - [Expected: {"error":false,"string":"8-2=6","answer":6}, Actual: {"error":false,"string":"8-2=6","answer":6}] - [Response Time: 36.831621999999996 ms]
[Test Completed] - [25/03/2021 12:00] - [Passed] - [Expected: {"error":false,"string":"8-2=6","answer":6}, Actual: {"error":false,"string":"8-2=6","answer":6}] - [Response Time: 32.644656999999995 ms]
[Test Completed] - [25/03/2021 11:30] - [Passed] - [Expected: {"error":false,"string":"8-2=6","answer":6}, Actual: {"error":false,"string":"8-2=6","answer":6}] - [Response Time: 32.874479 ms]
[Test Completed] - [25/03/2021 11:00] - [Passed] - [Expected: {"error":false,"string":"8-2=6","answer":6}, Actual: {"error":false,"string":"8-2=6","answer":6}] - [Response Time: 32.853811 ms]
[Test Completed] - [25/03/2021 10:30] - [Passed] - [Expected: {"error":false,"string":"8-2=6","answer":6}, Actual: {"error":false,"string":"8-2=6","answer":6}] - [Response Time: 34.552302 ms]
[Test Completed] - [25/03/2021 10:00] - [Passed] - [Expected: {"error":false,"string":"8-2=6","answer":6}, Actual: {"error":false,"string":"8-2=6","answer":6}] - [Response Time: 30.607568999999998 ms]
[Test Completed] - [25/03/2021 09:30] - [Passed] - [Expected: {"error":false,"string":"8-2=6","answer":6}, Actual: {"error":false,"string":"8-2=6","answer":6}] - [Response Time: 29.882724999999997 ms]
[Test Completed] - [25/03/2021 09:00] - [Passed] - [Expected: {"error":false,"string":"8-2=6","answer":6}, Actual: {"error":false,"string":"8-2=6","answer":6}] - [Response Time: 30.799751999999998 ms]
[Test Completed] - [25/03/2021 08:30] - [Passed] - [Expected: {"error":false,"string":"8-2=6","answer":6}, Actual: {"error":false,"string":"8-2=6","answer":6}] - [Response Time: 31.254008 ms]
[Test Completed] - [25/03/2021 08:00] - [Passed] - [Expected: {"error":false,"string":"8-2=6","answer":6}, Actual: {"error":false,"string":"8-2=6","answer":6}] - [Response Time: 29.770056 ms]
[Test Completed] - [25/03/2021 07:30] - [Passed] - [Expected: {"error":false,"string":"8-2=6","answer":6}, Actual: {"error":false,"string":"8-2=6","answer":6}] - [Response Time: 30.131949 ms]
[Test Completed] - [25/03/2021 07:00] - [Passed] - [Expected: {"error":false,"string":"8-2=6","answer":6}, Actual: {"error":false,"string":"8-2=6","answer":6}] - [Response Time: 35.617635 ms]
[Test Completed] - [25/03/2021 06:30] - [Passed] - [Expected: {"error":false,"string":"8-2=6","answer":6}, Actual: {"error":false,"string":"8-2=6","answer":6}] - [Response Time: 31.381873 ms]
[Test Completed] - [25/03/2021 06:00] - [Passed] - [Expected: {"error":false,"string":"8-2=6","answer":6}, Actual: {"error":false,"string":"8-2=6","answer":6}] - [Response Time: 43.044408999999995 ms]
[Test Completed] - [25/03/2021 05:30] - [Passed] - [Expected: {"error":false,"string":"8-2=6","answer":6}, Actual: {"error":false,"string":"8-2=6","answer":6}] - [Response Time: 29.986265 ms]
[Test Completed] - [25/03/2021 05:00] - [Passed] - [Expected: {"error":false,"string":"8-2=6","answer":6}, Actual: {"error":false,"string":"8-2=6","answer":6}] - [Response Time: 33.307943 ms]
[Test Completed] - [25/03/2021 04:30] - [Passed] - [Expected: {"error":false,"string":"8-2=6","answer":6}, Actual: {"error":false,"string":"8-2=6","answer":6}] - [Response Time: 30.917493999999998 ms]
[Test Completed] - [25/03/2021 04:00] - [Passed] - [Expected: {"error":false,"string":"8-2=6","answer":6}, Actual: {"error":false,"string":"8-2=6","answer":6}] - [Response Time: 34.53562 ms]
[Test Completed] - [25/03/2021 03:30] - [Passed] - [Expected: {"error":false,"string":"8-2=6","answer":6}, Actual: {"error":false,"string":"8-2=6","answer":6}] - [Response Time: 30.674829 ms]
[Test Completed] - [25/03/2021 03:00] - [Passed] - [Expected: {"error":false,"string":"8-2=6","answer":6}, Actual: {"error":false,"string":"8-2=6","answer":6}] - [Response Time: 64.65193 ms]
[Test Completed] - [25/03/2021 02:30] - [Passed] - [Expected: {"error":false,"string":"8-2=6","answer":6}, Actual: {"error":false,"string":"8-2=6","answer":6}] - [Response Time: 30.534367 ms]
[Test Completed] - [25/03/2021 02:00] - [Passed] - [Expected: {"error":false,"string":"8-2=6","answer":6}, Actual: {"error":false,"string":"8-2=6","answer":6}] - [Response Time: 30.277952 ms]
[Test Completed] - [25/03/2021 01:30] - [Passed] - [Expected: {"error":false,"string":"8-2=6","answer":6}, Actual: {"error":false,"string":"8-2=6","answer":6}] - [Response Time: 31.229713999999998 ms]
[Test Completed] - [25/03/2021 01:00] - [Passed] - [Expected: {"error":false,"string":"8-2=6","answer":6}, Actual: {"error":false,"string":"8-2=6","answer":6}] - [Response Time: 31.390379999999997 ms]
[Test Completed] - [25/03/2021 00:30] - [Passed] - [Expected: {"error":false,"string":"8-2=6","answer":6}, Actual: {"error":false,"string":"8-2=6","answer":6}] - [Response Time: 31.233915999999997 ms]
[Test Completed] - [25/03/2021 00:00] - [Passed] - [Expected: {"error":false,"string":"8-2=6","answer":6}, Actual: {"error":false,"string":"8-2=6","answer":6}] - [Response Time: 32.498453999999995 ms]
[Test Completed] - [24/03/2021 23:30] - [Passed] - [Expected: {"error":false,"string":"8-2=6","answer":6}, Actual: {"error":false,"string":"8-2=6","answer":6}] - [Response Time: 30.830968 ms]
```

Task E

Repository URL: Not applicable
Live Service URL: Not applicable

To save the state for the results in the calculator I decided to set up a ‘Realtime Database’ using Google’s Backend as a Service (BaaS) Firebase, which I communicate with through the proxy service. Once the database was created, I setup an endpoint, ‘/write’ in my proxy router to forward write requests to the database, all that was required to ensure the state was saved correctly was to pass an appropriate reference, which can be thought of as a table name, and some data in an API request to database and database will generate a unique ID and save it. I have included a screenshot below of how the database looks in Firebase and some code snippets of how it was configured and how the write endpoint works.



webcalc-db
Go to docs

Realtime Database

Data Rules Backups Usage

https://webcalc-db-default-rtdb.europe-west1.firebasedatabase.app/

```
webcalc-db-default-rtdb
  results
    -MWKSJe2f-PVxXisFBTn
      result: 7
    -MWKSLMsaiFMISjl92Ky
    -MWK_4lVBZZD7U96S5c5
    -MWK_u_vNDsty0RRSII0
    -MWKf2mBM7Cc02-O_t0s
    -MWKglPPqJrUjs9Edj8s
    -MWKhl_Sfhl1iLdxcvvO
    -MWKsJwtpuZ1WeDbHgAo
    -MWKspNfWTkAS5dwqMml
    -MWLY6GUWb3aDnszYUCE
    -MWLYBHtYmoBjljPXUid
    -MWLYCqet-SPztTER-17
```

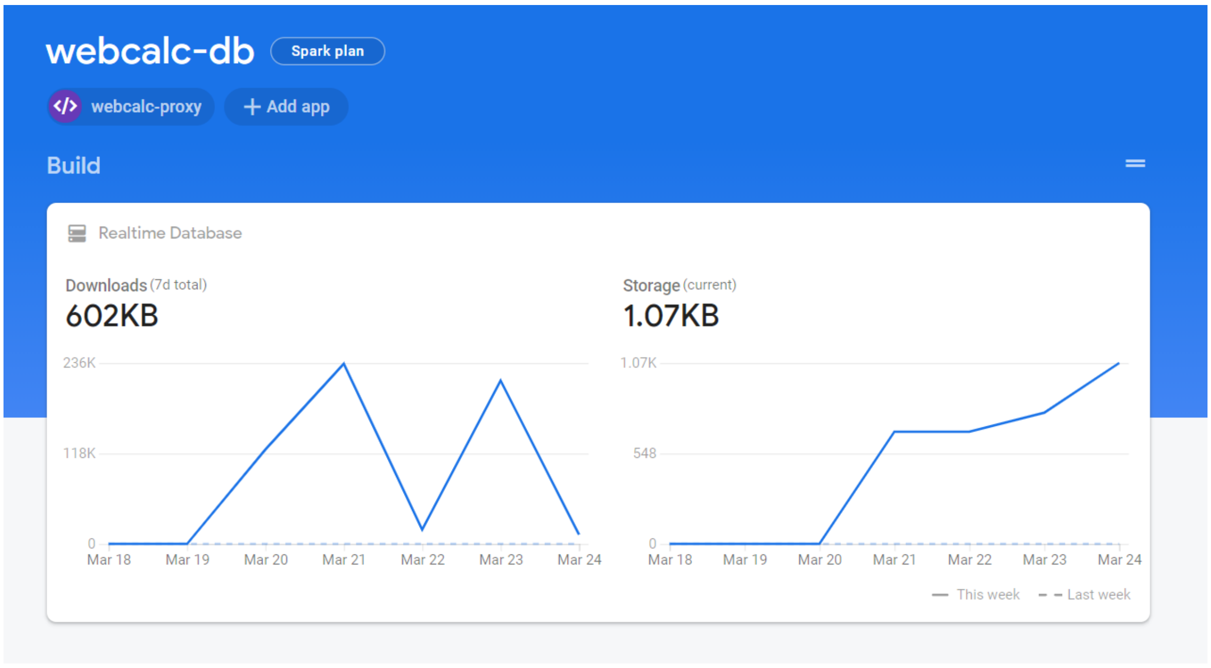
Database location: Belgium (europe-west1) [beta]

```
const firebaseConfig = {
    apiKey: "AIzaSyByz5ZEn0HriM7ICNRE41jttr-XiqIMtns",
    authDomain: "webcalc-db.firebaseapp.com",
    projectId: "webcalc-db",
    storageBucket: "webcalc-db.appspot.com",
    messagingSenderId: "56647902383",
    appId: "1:56647902383:web:c54decdb1bb7a1241e205a",
    measurementId: "G-K3RD4K74HY",
    databaseURL: "https://webcalc-db-default-rtdb.europe-
west1.firebasedatabase.app/"
};

firebase.initializeApp(firebaseConfig)
```

```
let database = firebase.database();

const resultsRef = database.ref("results");
```

```
app.post('/write', (req, res) => {
    res.setHeader('Content-Type', 'application/json');
    res.setHeader('Access-Control-Allow-Origin', '*')

    var data = {
        result: req.body.result,
    }

    var createdId = resultsRef.push(data, function (error) {
        if (error) {
            res.end(JSON.stringify({ result: "fail" }));

        } else {
            data = {
                result: "success",
            };
        }
    }).getKey();

    data.key = createdId;

    res.end(JSON.stringify(data));
});
```

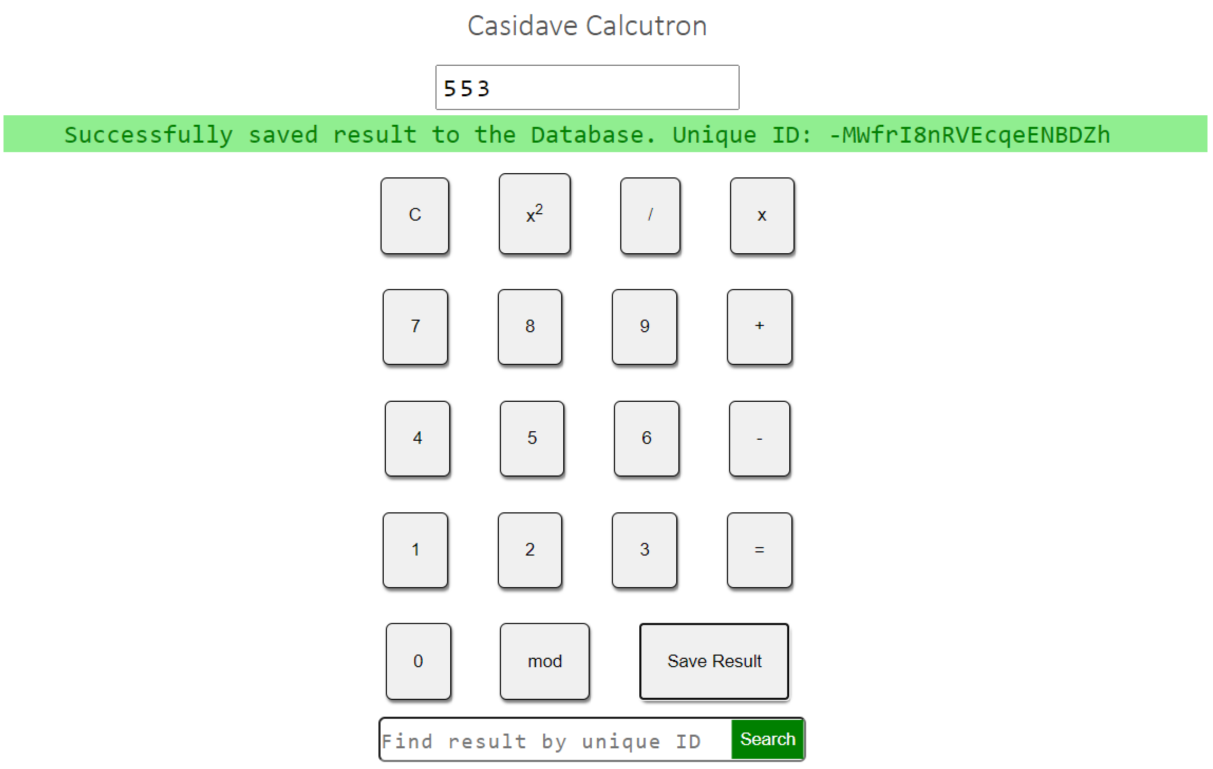
An advantage of using Firebase is usage analytics such as below are provided and can be easily consumed by administrators to evaluate the traffic the database is experiencing, in a system such as this if multiple users were constantly making calculations and saving them the database could experience heavy traffic and there may be a need to change the architecture, for example splitting the database in two, analytics such as below are useful for flagging this need.



In order to read saved values from the database, I set up an endpoint '/read' which forwards requests to firebase and retrieves a record based on the specified unique ID, once the record is retrieved from the database the proxy forwards it to the frontend, if there is no record found matching the ID passed then error is set to true and the string shows a message informing the user the record does not exist. I have included a code snippet below of how the saved record is retrieved.

```
app.get('/read', (req, res) => {
    var id = req.query.id;
    console.log('In read');

    resultsRef.on("value", function (snapshot) {
        console.log('In snapshot');
        console.log(snapshot);

        var value = snapshot.val()[id];

        console.log(value);

        if(value){
            res.end(JSON.stringify({
                error: false,
                string: "Success",
                result: value.result,
            }));
        }else{
            res.end(JSON.stringify({
                error: true,
                string: "No such record exists with that ID",
                result: 0,
            }));
        }
    }, function (_) {
        res.end(JSON.stringify({
            error: true,
            string: "An error occurred",
            result: 0,
        }));
    });

});
```

Once all the backend functionality was set up, I then made the relevant changes to the front end to allow the user to actually save values and reload them later. First I added a button (Save Result) with an OnClick handle that would send a request to the proxy with the value that was currently displayed in the calculator, the proxy in turn would send it to the database, once saved it will return the unique ID created and display it to the user so it can be used to receive the saved state later.


Casidave Calcutron

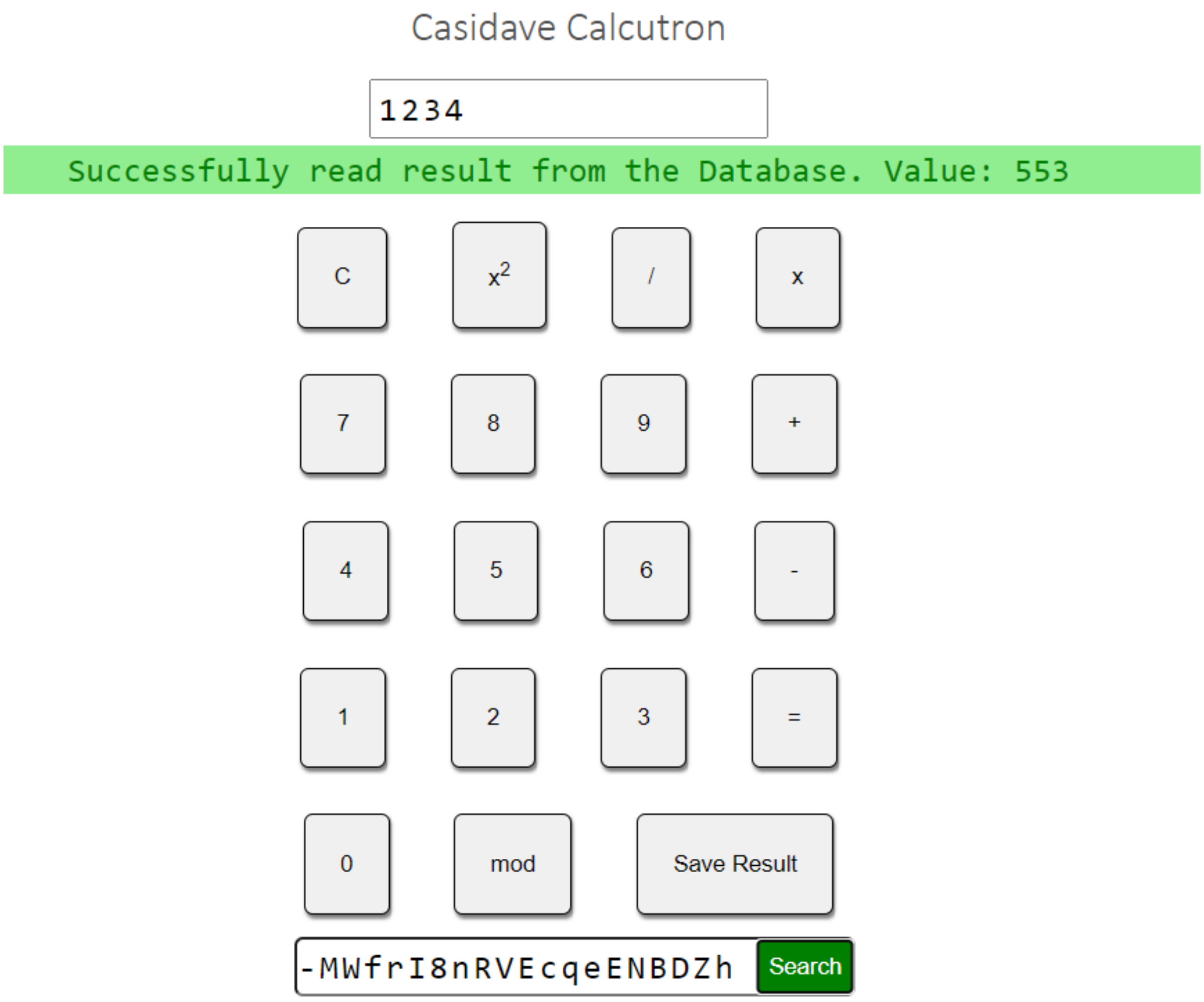
553

Successfully saved result to the Database. Unique ID: -MWfrI8nRVEcqeENBDZh

```
<button class="calc1" onclick="SaveResult();">Save Result</button>
```

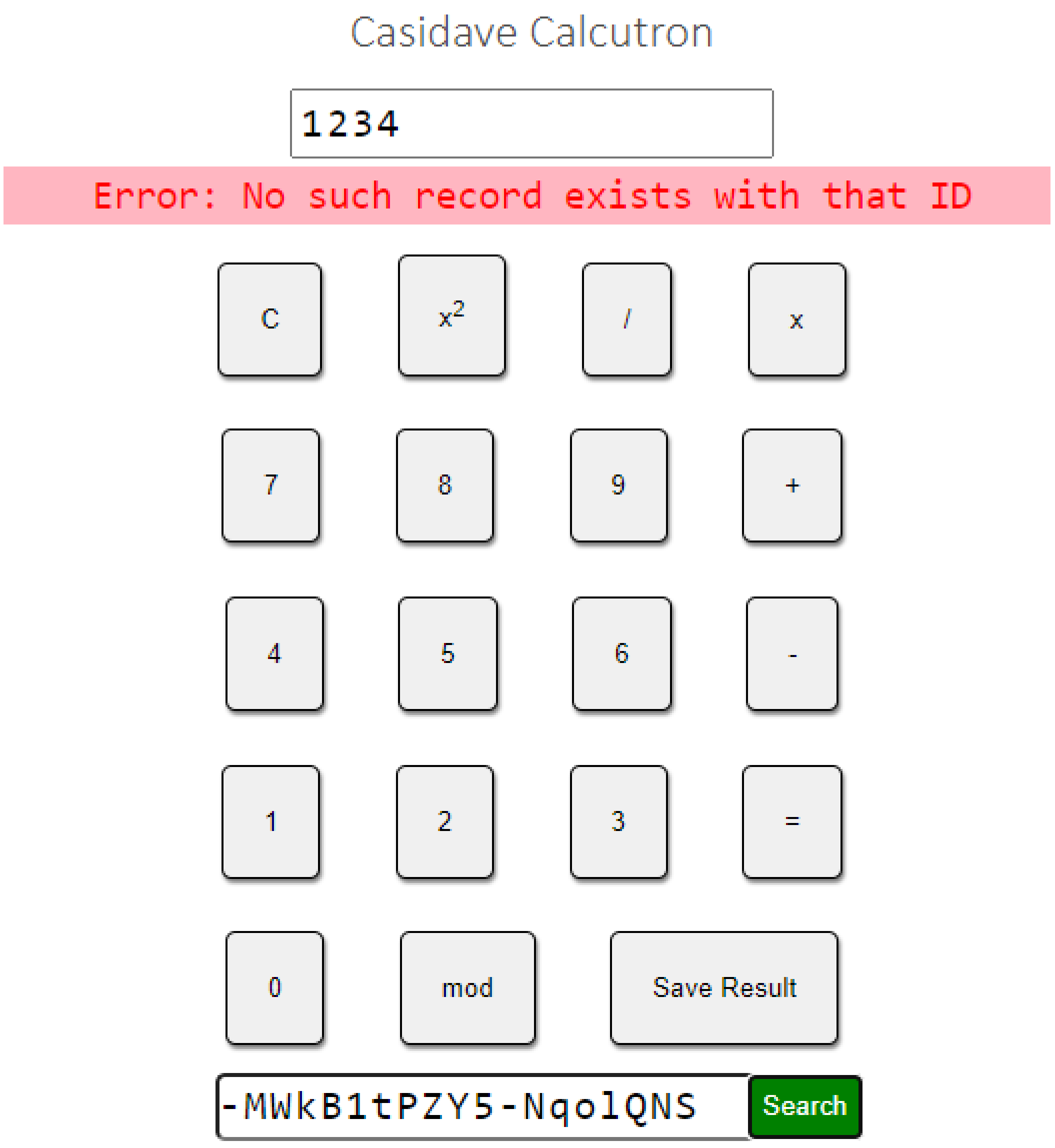
```
async function SaveResult() {
            var data = {
                result: value
            };
            var headers = { "Access-Control-Allow-Origin": "*" };

            var requestSuccessful = false;
            while (availableUrls.length > 0 && !requestSuccessful) {
                var responseCode;

                await axios.post(`${availableUrls[0]}write`, data)
                    .then(function (response) {
                        responseCode = response.status;
                        ShowSuccessMessage(`Successfully saved result
to the Database.\nUnique ID: ${response.data.key}`);
                    })
                    .catch(function (error) {
                        if (error.response) {
                            responseCode = error.response.code;
                        }
                        else {
                            responseCode = 503;
                        }
                        ShowErrorMessage(error);
                    });

                requestSuccessful = responseCode == 200;

                if (!requestSuccessful) {
                    failedUrls.push(availableUrls[0]);
                    availableUrls.shift();
                }
            }
        }
```

In order to allow the user to retrieve saved results, I added a text input to the front end and a search button with an OnClick handler, which when clicked takes the value from the text input and sends it to the '/read' endpoint in the proxy, which will retrieve it from the database and then display it at the top of the calculator in a success message.


Casidave Calcutron

1234

Successfully read result from the Database. Value: 553



Casidave Calcutron

1234

Error: No such record exists with that ID

-MWkB1tPZY5-NqolQNS

```
<div class="buttonIn">
                 <input id="find-by-id" type="text" id="enter"
placeholder="Find result by unique ID">
                 <button id="find-by-id-button"
onclick="GetResultById();">Search</button>
             </div>
```

```
async function GetResultById() {
            var id = document.getElementById("find-by-id").value;

            var requestSuccessful = false;
            while (availableUrls.length > 0 && !requestSuccessful) {
                var responseCode;

                var url = `${availableUrls[0]}read/?id=${id}`;
                await axios.get(url)
                    .then(function (response) {
                        responseCode = response.status;
                        if (response.data.error){
                            ShowErrorMessage(response.data.string);
                        }else{
                            ShowSuccessMessage(`Successfully read result from
the Database. \nValue: ${response.data.result}`);
                        }
                    })
                    .catch(function (error) {
                        if (error.resonse) {
                            responseCode = error.response.code;
                        }
                        else {
                            responseCode = 503;
                        }
                        ShowErrorMessage(error);
                    });

                requestSuccessful = responseCode == 200;

                if (!requestSuccessful) {
                    failedUrls.push(availableUrls[0]);
                    availableUrls.shift();
                }
            }
        }
```

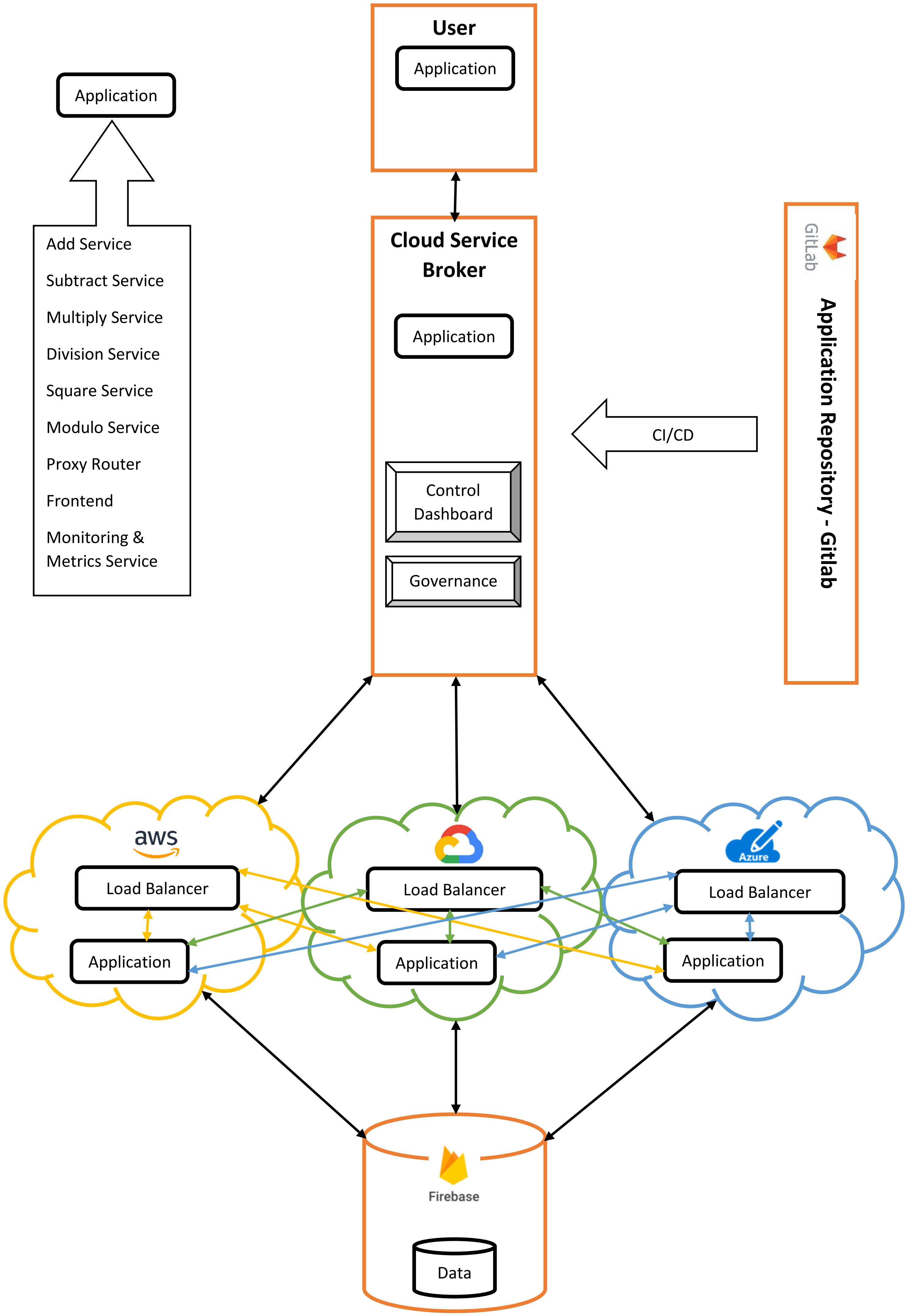
Task F

The following page displays my design on how the web calculator could be fully deployed onto multiple cloud vendors with a common broker/router. The three cloud services included in the design are AWS, GCP and Microsoft Azure. The application repositories are referenced under the same title 'Application' and will remain in Gitlab as it provides the necessary functionality for continuous integration and continuous deployment.

To deploy on multiple cloud services, software such as Terraform can be used. The 'application', the contents of which are defined below will all be deployed on three different cloud platforms. Communicating between the three independent services can be elaborate, time-consuming, and expensive, this is where the cloud broker becomes necessary. Each service will have its own dashboard however using a cloud broker will allow for a centralised control dashboard that can be used to consume information about all the cloud services as well as manage them. The control panel improves visibility for both developers and operations engineers. Operations engineers can use the centralised control dashboard to spin up and configure Kubernetes clusters, see other clusters, pods and deployments. From the developer's point of view this centralised control panel means that once the new software is containerized and ready to deploy, the service can be deployed to all three cloud services with just one command. Governance of the cloud services can be made easier by the cloud broker, a compliance policy could be pushed out to multiple clusters from the central control dashboard.

The Web Calculator's services hold some sensitive data that might need protection on a private cloud, such as the email and passwords in the Monitoring and Metrics aspect of the project. Inside these files, there are also details of the metrics of the calculator which may not want to be stored on a public cloud. Thus, I have considered the deployment of some applications on a private cloud network as well as using public cloud vendors. This would incorporate a hybrid cloud with a multi cloud topology. A private cloud stays inside an organisation's intranet network behind a firewall. It provides access to the same resources as the public model, but with less exposure to internet security risks. Thus, less susceptible to cyber-attacks and the loss of sensitive data.

The ability to endure, adapt, and grow in the face of change is referred to as resilience. In this situation, there is a need for cloud services to be delivered redundantly at design time, e.g. across at least two separate endpoints, for the survival and adaptation of cloud services. Furthermore, we need the presence of mechanisms capable of anticipating and avoiding possible service failure to survive and adapt at run-time. If a loss occurs, the mechanism must be able to rebound from it and continue to provide a reliable service that leads to the project's overall success and progress. With a multi-cloud strategy, we can respond to unplanned service outages by failing over their workloads routing them from one public cloud to another based on their locality in the world to the data centres to avoid latency issues, overall improving user experience. Linking back to resilience and the ability to grow and adapt, a useful feature of cloud vendors is an Auto Scaling feature. Auto Scaling monitors your applications and automatically adjusts capacity to maintain steady, predictable performance at the lowest possible cost. This gives you the ability to grow or shrink the number of server resources on demand, and this work is done by load balancer.



Application
Add Service
Subtract Service
Multiply Service
Division Service
Square Service
Modulo Service
Proxy Router
Frontend
Monitoring & Metrics Service
User
Application
Cloud Service Broker
Application
Control Dashboard
Governance
CI/CD
GitLab
Application Repository - Gitlab
aws
Load Balancer
Application
Load Balancer
Application
Azure
Load Balancer
Application
Firebase
Data