CSC7071 Cloud Computing

**Assessment 2: WebWordCount App**

Submission

Student Number: 40313770

Name: Kevin Brolly

**Task A**

Use the following structure for each of the functions you have implemented, if necessary please feel free to include limited additional information such as code, screenshots, or anything else you want to maximise demonstrating how you meet/exceed marking criteria (in most cases this isn’t required). Note the “anything else” is entirely optional just in case something didn’t fit into implementation or testing. Please also note we will mark just the FOUR functions given here (i.e. it’s better to have four more complete services than submit six less so, only four will marked!).

**Function One**

Function to check total word count in the given paragraph:

Repository URL: <http://gitlab.hal.davecutting.uk/40313770/wwc-wcount>

Live Service URL: [http://wordcount.40313770.qpc.hal.davecutting.uk/?paragraph=this%20 is%20a%20test%20paragraph](http://wordcount.40313770.qpc.hal.davecutting.uk/?paragraph=this%20%20is%20a%20test%20paragraph)

Description of Implementation (language, methods, paradigm, etc):  
  
PHP was utilised to develop this function.

Text

Description automatically generated Text

Description automatically generated

*Left Figure: The application can identify and count the number of test instances despite poor formatting.*

*Right Figure: Depicts keyword count function script processing user data before being sent to back end function.*

Graphical user interface, text, application

Description automatically generated  
*Figure: The keyword count function invoked by the front end designed with FaaS in mind.*

Description of Testing:  
PHP was utilised to develop this functions unit test.

A screenshot of a computer

Description automatically generated Graphical user interface, text, application

Description automatically generated

*Left Figure: keyword count back-end unit test result.   
Right Figure: Depicts keyword count function unit test script, due to front end data pre-processing only a positive test was checked.*

Graphical user interface, text, application

Description automatically generated Graphical user interface, application

Description automatically generated  
*Left Figure: word count test script. Right Figure: word count variables added in URL printed to screen.*

Anything else to highlight:

**Function Two**

Function to check the total number of times the keyword appears in the paragraph:

Repository URL: <http://gitlab.hal.davecutting.uk/40313770/wwc-kwcount>   
Live Service URL:[http://keywordcount.40313770.qpc.hal.davecutting.uk/?paragraph=this% 20test%20tests%20the%20number%20of%20instances%20of%20test%20based%20on%20the% 20test%20keyword%20test?&word=test](http://keywordcount.40313770.qpc.hal.davecutting.uk/?paragraph=this%25%2020test%20tests%20the%20number%20of%20instances%20of%20test%20based%20on%20the%25%2020test%20keyword%20test?&word=test)

Description of Implementation (language, methods, paradigm, etc):  
PHP was utilised to develop this function.

Text

Description automatically generated Text

Description automatically generated

*Left Figure: The application can identify and count the number of test instances despite poor formatting.*

*Right Figure: Depicts keyword count function script processing user data before being sent to back end function.*

Graphical user interface, text, application

Description automatically generated

*Figure: The keyword count function invoked by the front end designed with FaaS in mind.*

Description of Testing:  
  
PHP was utilised to develop this functions unit test.

A screenshot of a computer

Description automatically generated Graphical user interface, text, application

Description automatically generated

*Left Figure: keyword count back end unit test result.   
Right Figure: Depicts keyword count function unit test script, due to front end data pre-processing only a positive test was checked.*

A screenshot of a computer

Description automatically generated Graphical user interface, application

Description automatically generated

*Left Figure: keyword count test script. Right Figure: keyword count variables added in URL printed to screen and tested.*

Anything else to highlight:

**Task B**

**Frontend Usability and Functionality Improvements**

Repository URL: <http://gitlab.hal.davecutting.uk/40313770/wwc-frontend>

Live Service URL: <http://webapp.40313770.qpc.hal.davecutting.uk/>

**Design Philosophy**To process and clean data before it is passed to a particular function in the back end and keept the back end functions minimalist in design and function.

**Implemented Language List**  
Html, CSS, JavaScript, php unit testing

**Implementation Designs & Methodology**

CSS was relocated to a separate file for easier development now and in the future.  
Linked backend services to front end buttons

A screenshot of a computer

Description automatically generated Graphical user interface, text

Description automatically generated

*Figure : Front end container file structure. Figure : Backend Services connected to the front end.*

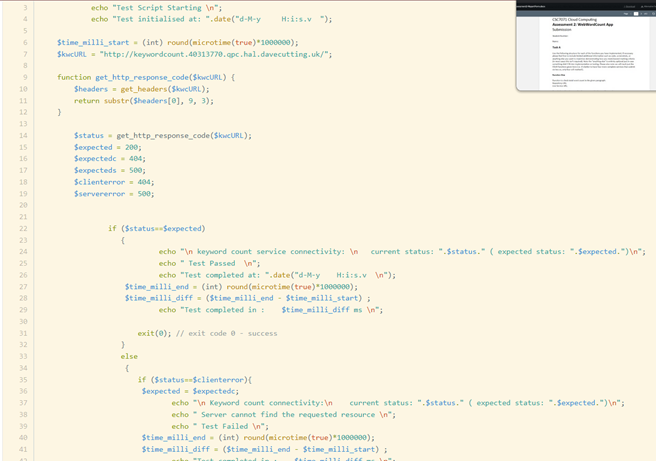
Error handling implemented to prevent backend errors and crashes   
as described in the brief, users are warned if a backend service is attempted to be invoked when:  
Keyword or paragraph is null.  
Keyword or paragraph consists of white space.  
Multiple words are used as the keyword.

Text

Description automatically generated  
*Figure : implemented error handling and data processing for check() function. Error handling occurs and processing occurs before data is sent to the backend services which have been implemented in a FaaS manner.*  
keywords and paragraphs are case insensitive; they are all turned lowercase  
Strip special characters and numbers from word and paragraph as this impacts word count prevents SQL injection and if future additional functions such as character count are implemented.

**Description of Testing:**  
The testing assesses the front end connection to backend services in Gitlab utilising the get\_http\_response\_code(“backend service URL”) function, to pass a status code of 200 is expected, also include coverage of 404 and 500.

A screenshot of a computer

Description automatically generated 

*Left Figure: Frontend connectivity test results to backend services. Right Figure: Depicts keyword count function test script.*

**Task C**

For the custom proxy router that you have built, complete the following:

Custom Proxy router Code and Title:

Implementation Details (how you implemented):

Testing Details (how you tested):

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

Anything else to highlight (optional):

**Task D**

**Monitoring & Metrics**

Repository URL: <http://gitlab.hal.davecutting.uk/40313770/wwc-monmet>

Live Service URL: <http://monmet.40313770.qpc.hal.davecutting.uk/>

Implementation Details

* Php scripts written to test the connection to backend services
* Bash script and crontab configured to run these scripts every 15mins, report the results to the front-end monitoring metrics and append to their respective .csv file
* Configured mail service to send a system alert when a connection to a backend service results in anything but a 200 status
* Monitoring front end reads the .csv files and creates a log of the test time, status, description, test end time and test duration. The label for each test is colour coded as the php script reads the files based on status code, 200 results are green, 404 orange and red when a 500 status code is red.

Graphical user interface, application

Description automatically generated Graphical user interface, website

Description automatically generated

*Left Figure: Monitoring metrics repo testing and recording services. Right Figure: Depicts keyword count test record and email function.*

Graphical user interface, text

Description automatically generated A screenshot of a computer

Description automatically generated

*Left Figure: Monitoring front end connectivity script. Right Figure: Monitoring and Metrics Dockerfile image.*

Graphical user interface, text, application

Description automatically generated Graphical user interface, text

Description automatically generated

*Left Figure: Crontab executing loggingtasks.sh every 15min. Right Figure: Cron tab wrapper script to execute these scripts.*

Implementation Limitations

* Sendmail is not a particularly secure service – would have liked to use a 3rd party like PearMail but did not have the time to perfect the configuration
* Multiple instances of executable privileges and ‘sudo-like’ privileges given to non-root users to allow for successful configuration
* Would have liked to have a separate mail container but need 2 pods for each container so the container limit (10) would have been exceeded
* I only managed to implement a status check on connectivity and not correctness, due to a lack of time I would have liked to include a function to input test words and paragraphs, displaying the input, expected and actual results post processing and as the connectivity monitoring function enabled, the ability to time the processing time of a request. A configurable unit test such as those utilised in the Gitlab.
* Ideally, I would have liked to also include an external SQL database to store and manipulate record data presented in the UI.

**Conclusion**

Summary of your report or any concluding remarks.

**References**

<https://www.php.net/manual/en/function.fopen.php>   
<https://stackoverflow.com/questions/38320663/multiple-php-scripts-run-at-same-time-in-cron>   
<https://unix.stackexchange.com/questions/287166/how-to-run-multiple-scripts-at-a-same-time>   
<https://www.cloudsavvyit.com/9033/how-to-use-cron-with-your-docker-containers/>   
<https://futurestud.io/tutorials/ubuntu-debian-how-to-test-a-cron-job>   
<https://unix.stackexchange.com/questions/212355/where-is-my-logfile-of-crontab>   
<https://stackoverflow.com/questions/21731745/how-to-add-shebang-with-php-script-on-linux>   
<https://stackoverflow.com/questions/21731745/how-to-add-shebang-with-php-script-on-linux>   
<https://www.php.net/manual/en/function.fopen.php>   
<https://stackoverflow.com/questions/44716612/docker-php-permissions>   
<https://www.w3schools.com/php/php_json.asp>   
<https://www.w3schools.com/php/php_callback_functions.asp>   
<http://etutorials.org/Server+Administration/Sendmail/Part+I+Build+and+Install/Chapter+4.+Configure+sendmail.cf+with+m4/FEATUREauthinfo/>   
<http://etutorials.org/Server+Administration/Sendmail/Part+I+Build+and+Install/Chapter+4.+Configure+sendmail.cf+with+m4/FEATUREuse_ct_file/>   
<https://www.oreilly.com/openbook/linag2/book/ch18.html>