1. Assessment Details

The WebWordCount App is a web-based word count application that is designed with minimal features and without providing much attention into its reliability or user friendlessness. It has been implemented using several cloud technologies including stateless, containers, CI, and is currently deployed on the QPC.

Your assignment is to start with this as a base and to add more functionalities as specified in this document.

Currently the WebWordCount App consists of two container components:

webwordcount-frontend – this is a simple container that hosts the frontend of the WebWordCount application which is designed using some static HTML and JavaScript. The GUI in the frontend provides the textboxes to enter the input and buttons to submit the requests for the keyword check and word count. The GUI is designed to be able to submit three different kinds of requests: (i) checking total word count, (ii) checking whether the keyword appears in the given paragraph, and (iii) checking total number of times the keyword appears in the paragraph. Out of these three requests only the second one works: whether keyword appears in the paragraph. This request is taken by the green button in the GUI.

When the "check keyword appearance" button is clicked the frontend takes this request and passes it to a container that hosts the service that can process the request. The request is sent to the container service via XMLHttpRequest in Javascript and response from the service is received in JSON format.

Repository:
http://gitlab.hal.davecutting.uk/esha/webwordcount/tree/master/webwordcount-frontend-master

Note: This repository access will be given in one of the lectures in week 3.

Live Demo: http://webwordcount.esha.qpc.hal.davecutting.uk/

webwordcount-check – this is a PHP based container that hosts a PHP script to process one of the requests from the frontend, which is to check whether a keyword is present in the paragraph that is provided in the frontend GUI. The container takes two HTTP GET parameters ('paragraph' and 'word') and returns the result of the check in JSON format.

There is a rudimentary CI testing (unit testing of the function that is used, but not of the service as a whole). Answer is '1' when the paragraph contains the keyword, its '0' otherwise.

Repository:
http://gitlab.hal.davecutting.uk/esha/webwordcount/tree/master/webwordcount-check-master
Live Demo:
http://check.webwordcount.esha.qpc.hal.davecutting.uk/?paragraph=eshabarlaskar&word=esha [paragraph contains the keyword – answer is 1]
http://check.webwordcount.esha.qpc.hal.davecutting.uk/?paragraph=eshabarlaskar&word=david [paragraph does not contain the keyword – answer is 0]

You are required to modify and extend this WebWordCount App as described in the following tasks.

1.1 Tasks

A. New Container Services

Add two new container services to the WebWordCount App which can complete the pending requests in the frontend:

(i) Checking total word count in the given paragraph.
(ii) Checking the total number of times the keyword appears in the paragraph.

The buttons (in the frontend GUI) that can be linked to these services are in grey colour. See the assessment criteria for more information but in general higher marks are given not just for a working service but for advanced modern techniques such as use of CI pipelines.

Points will also be awarded for use of novel technologies, e.g., deployment using FaaS, however, there will be no additional points for using new languages.

B. Improvement to Current Implementation

There are a number of issues with how the current WebWordCount App works which include (but are likely not limited to):

- No error handling in the frontend, for example, if the user input (i.e., the paragraph or the keyword) is empty and an attempt is made to check keyword appearance a request will be sent to the back end service which will crash (or return an error) but no alert message is displayed, or if any other error condition occurs in the backend, such as having multiple words for the keyword, then this is not reported in the frontend thus making it difficult to identify the problems.
- The keyword matching in the application is case sensitive: it has to be case insensitive
- CI testing for the provided backend service is very limited as it does unit tests on the function alone, i.e., it does not actually perform an HTTP request to check web API functionality
- Frontend service failure handler – the frontend currently stores a single URL for each service. This challenge is to update the frontend to support multiple URLs for each

service. If a specific endpoint falls offline or errors, the frontend should be able to continue using other endpoints instead. This may also include load balancing while multiple endpoints are available. In combination with Task C (see below) this would refer to multiple URLs for the proxy service provided by Task C. Please note "frontend" as always refers to the HTML+JS component (the in-browser)

Plan and implement improvements to address some or all of the above, or other shortcomings or issues you perceive within the provided system. You should aim to make four improvements (either the above or other changes; if you are unsure of the scope of your planned changes speak to a member of the teaching team). See the assessment criteria and submission instructions for more information on what may be assessed and how to show this working.

C. Custom proxy router

Custom proxy router – currently each service endpoint has a custom URL configured in the frontend client. For this challenge you will build a self-container web proxy as a container to which the frontend will pass all calls to (including the 'paragraph' and 'word'). This proxy will then itself hold the specific endpoints for different methods and make the request to the actual service before returning the answer.

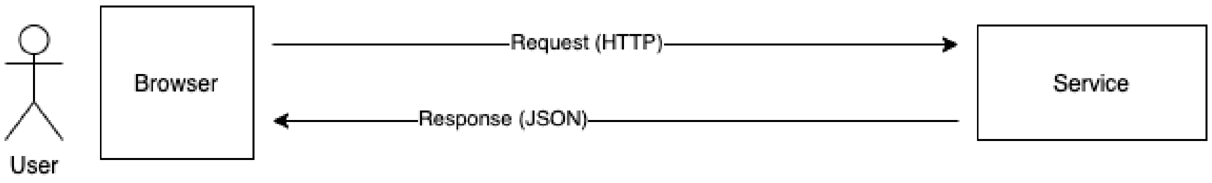

Figure 1 Requests and response without proxy (direct to service)

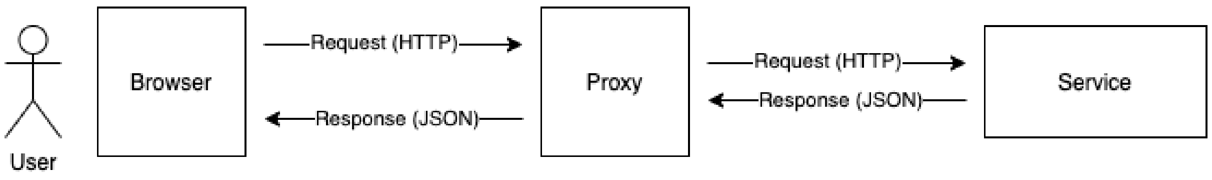

Figure 2 Requests and response with proxy, proxy makes request to relevant service

For full marks this service should be highly configurable and support some service discovery options (what methods are available for example). This means, it should be something like an ingress controller or load balancer which loads its setup from a configuration file and can update at runtime. For example, consider that we may want to change the URL a service goes to or even add another service with a new operator.

Service discovery features could include automatic configuration using these systems or the proxy having an ability to probe a URL and discover what options and operator is available for example.

Note you are required to build this yourself rather than using a third-party service or container.

D. Monitoring and Metrics

Monitoring and testing service – the testing currently happens during the build cycle with CI testing. Implement a separate service (this could be a container or other service) that when used makes HTTP requests with random texts to the services and checks (a) correctness of results against expectations and (b) overall performance in time (in an ideal world this service should periodically check performance and record results in addition to an on-demand operation). Note you are required to build this yourself rather than using a third-party service or container.

For full marks you are expected to have some form of monitoring periodically test and alert on failure of your services (this can be an external service or container as you like as long as you have done the first part from scratch yourself).

1.2 Deployment Environment

As it stands the current WebWordCount App is deployed on the QPC Kubernetes system with source code and docker registry in the QPC gitlab install. It is suggested that you continue to use this architecture for your project, or at least most of it, but this isn’t a requirement.

You are free to use any provider(s) you like and their technology stacks but be aware that (a) less support will be available and (b) you must make any code accessible when you submit (see submission section for details). It’s strongly suggested you discuss with the teaching team before selecting a platform other than QPC Kubernetes/gitlab.

2. Assessment Criteria

The following are the criteria against which your submission will be marked and their conceptual marking equivalents.

Marking Criteria

Criteria	**Outstanding 85% +**	**Excellent 70%-85%**	**Very Good 60%-70%**	**Good 50%-60%**	**Acceptable 40%-50%**	**Unacceptable < 40%**
Task A. Additional Functions 2 x 15% (total of 30% for A)	Perfect implementation of function in any language or paradigm with excellent CI tests fully covering functionality	Excellent implementation of function in any language or paradigm with excellent CI tests. Some very minor weaknesses in implementation such as a lack of sensible error conditions.	Good implementation of a function usually in any language with good CI tests. Some weaknesses in some aspects of implementation.	Function implemented well in any language or with significant updates to provided models with CI tests. Some aspects of the implementation missing or lacking.	Function implemented and working but little extension shown beyond provided examples.	Function not fully or at all implemented, significant errors may be present in submission.
Task B. Addressing shortcomings 4x6%=24% (total of 24% for B)	Excellent understanding of the key shortcomings shown with exemplary well demonstrated implementation leading to a robust functional system.	Very good understanding of the key shortcomings shown with strong well demonstrated implementation leading to a robust functional system.	Good understanding of the key shortcomings shown with well demonstrated implementation leading towards a robust functional system.	Demonstrates understanding of some key shortcomings which are well addressed in a suitable fashion leading towards an improved system.	Some shortcomings addressed correctly or showing good intention and nearly functional solutions.	Shortcomings addressed in minor part or not at all.

Task C. Custom Proxy Router 16%	Excellent and perfectly demonstrated implementation of custom proxy router to a near industry standard incorporating best practices and documentation as appropriate.	Excellent and very well demonstrated implementation of custom proxy router to a very high standard incorporating best practices and documentation as appropriate.	Very good implementation of custom proxy router and well demonstrated incorporating some service discovery features and documentation as appropriate.	Good implementation of custom proxy router, clearly demonstrated and meeting the goals set well but with some weaknesses in design or implementation.	Solution meeting (or almost meeting) the requirements set for the custom proxy router but with major weaknesses in design or implementation.	Solution failing to meet the requirements, containing significant errors or lacking in functionality.
Task D. Monitoring & Metrics 30%	Excellent industry quality monitoring with all aspects considered and full periodic testing and alerting.	Excellent monitoring showing near industry quality with nearly all aspects considered and full periodic testing and alerting.	Very good monitoring showing strong consideration of nearly all aspects with periodic testing and alerting.	Good monitoring showing promise but with some significant weaknesses and/or some failings of periodic testing and alerting.	Basic monitoring in place with simple connections but with major weaknesses and/or no periodic alerting.	Fails to perform any effective monitoring or be close to operational.

3. Feedback

Feedback in the form of marks will be provided as soon as practicable after submission. Individual feedback will take the form of a numeric score against each of the assessment criteria (there may be brief comments for these criteria if appropriate), a total made from these scores weighted by section, and an overall textual comment on the totality of the submission.

Generalised feedback will be provided to the class as a whole including overall trends and areas of particular concern.

Anyone wishing to discuss their marks in more detail are welcome to do so using any of the support arrangements outlined in this document.

4. Submission

Submission will be via Canvas and also through repository access. **Please read this section carefully to avoid any mistakes** which could lead to marks being lost.

Note: **three uploads are required – unless you complete all three then this assignment will be regarded as not submitted**. It is your responsibility to check the validity/lack of corruption of any uploads.

Late penalties will be applied to this assignment based on the latest timestamp of any uploaded item.

You are required to:

1. Provide repository access if not using the QPC gitlab (gitlab.hal.davecutting.uk – if you are using this then you don't need to do anything other than provide the links in your report) so for example the EEECS gitlab or any other repository service. It is your responsibility to discuss this with a member of the teaching team to allow them access (or a link to a public repository) if this is the case.

2. Upload three items to the relevant assignment option on Canvas:
   - A completed report based on the template provided (you must use the template) named as <student_number>.pdf to the REPORT assignment on Canvas.
   - A demonstration video showing the operation of your system (see 4.1 for more details) to the VIDEO assignment on Canvas.
   - A zip file containing copy of all source code you have written (this is required in addition to the repository access for the external examiner) to the CODE assignment on Canvas.

3. Validate your uploads by viewing them via Canvas and making sure they are correct and uncorrupted. Previously students have uploaded the wrong assignments or an early version etc – it is your responsibility to ensure the correct item is uploaded.

4.1 Video Submission

To demonstrate your system working you are required to submit a video, usually a screen capture video, ideally as short as possible. This should show each of the key elements of your solution working (with inspection mode or similar to show network traffic to the endpoints).

You should use this video to highlight any specifics you wish to, for example any particularly detailed functionality included.

You are not required to specifically present or explain your work but we will not be running all code ourselves (some samples will be run) so this allows us to see it does actually operate! Be sure to cover all the features.

Please be mindful of filesize and compress/re-encode if needed. Previously we've had submissions of 1GB or more for a short video! This will take a long time to upload and upload (and check) must be completed by the deadline to avoid penalties.

5. Support Available

A number of support avenues are available throughout this project. It's suggested you try them in this order, but this is your choice and you should feel free to avail yourself of one or all.

Canvas Discussion – you can ask any questions (in general please, don't include your work as everyone can see!) on the Canvas Discussion Forum. This is very useful as everyone can see (the question and the answer!) and its possible students can help each other out.

Practical Sessions – there is a practical session Thursdays 1500-1700 where you are welcome to ask questions about anything including this assignment. Just pop up on the Teams question channel and say hi.

Module Drop-in – CSC7071 offers a virtual drop-in session every Tuesday 1600-1700. Our module demonstrators (who will be involved in the marking) will be guaranteed to be there and the module lecturer will be most often as well. Just pop up on the Teams question channel and say hi.

Open Door/Other Appointments – Esha Barlaskar operates an "open door" policy when she is available. She is happy to make one-to-one MS Teams meetings for individuals or groups as needed outside of the practical and drop-in sessions. Drop her an email or message via teams. Also, you can book a meeting slot in advance for the office hours available every Friday 1000-1200.