CSC3065 Cloud Computing 2023/24

Assessment 2: QUBengage

Assessment Briefing Version:	2 (06/11/2023)
Weighting:	60%
Set By:	Esha Barlaskar
Moderated By:	David Cutting

Date Released:	**06/11/2023**
Submission Due:	**1700 on 11/12/2023**

Late submission penalties and rules will be applied in accordance with the QUB policy on late submission. For more information on this or any other QUB policy with regards to assessment please see:

https://www.qub.ac.uk/directorates/AcademicStudentAffairs/AcademicAffairs/ExaminationsandAssessment/MarkSchemesandClassifications/

Please be aware that this is an individual assignment and as such will be checked for plagiarism. You should ensure that the work is yours and yours alone, citing any third-party sources as applicable. Plagiarism is a serious academic offense. Submission of your work implies your claim that it is your individual work and has not previously been submitted for academic credit. Also, student must include acknowledge any use of generative AI in their reports and must include this academic integrity statement in their report – *"I certify that that the submission is my own work, all sources are correctly attributed, and the contribution of any AI technologies is fully acknowledged."*

If you have any questions about the assignment, please see the module organiser Esha Barlaskar in the first instance or use any of the support options listed in this document.

1. Assessment Details

The ***QUBengage*** is the world's most over-engineered yet deeply unreliable student engagement monitoring app.

Your assignment is to start with this as a base and to add more functionality as specified in this document. There is some potential for a choice for what specifically you wish to implement so you must make decisions on how to make these improvements.

The purpose of this ***Student Engagement Monitoring (i.e., the QUBengage)*** microservice application is to allow the students to check their engagement in a module. The app should take total hours of student attendance in different sessions (lecture, lab, support) and the total hours of canvas activities, and using these input values the app should perform a number of functionalities.

Currently the QUBengage App can perform the following two tasks:

- **Find Maximum and Minimum Attendance –** to identify the session/activities with maximum and minimum hours of attendance.
- **Sort Attendance** – to sort the sessions/activities in descending order of hours of attendance.

The app performs these tasks through three different container services:

- ***QUBengage-frontend*** – this is a simple container that hosts the frontend of the QUBengage App. This is designed using HTML, CSS and JavaScript. The GUI in the frontend provides textboxes to enter the attendance/activities in hours as input, and buttons to submit the requests for finding the maximum and minimum attendance hours, and sorting the sessions/activities in terms of their attendance. The grey coloured buttons are inactive, and they are provided as placeholders for the new functionalities. For example, when the 'Maximum and Minimum Attendance Hours' or 'Sort Attendance' 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 (i.e., the worker service) via XMLHttpRequest in Javascript, and response from the service is received in JSON format. The endpoints it uses for these requests are configured within the Javascript.

  Repository: https://repository.hal.davecutting.uk/root/qubengage-frontend

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

- ***QUBengage-maxmin*** – this is a PHP based container that hosts PHP scripts to process one of the requests from the frontend, which is to find the maximum and minimum attendance hours out of all the sessions/activities that are entered in the frontend GUI. The container takes the input as HTTP GET parameters ('item_1', 'item _2', 'item _3', 'item _4', 'attendance_1', 'attendance _2', 'attendance _3', 'attendance _4') and returns the results ('error','items','attendance',' max_item','min_item') in JSON format. CI testing is not implemented in this version, and it's expected from the students to implement the CI testing to check the correctness of the results.

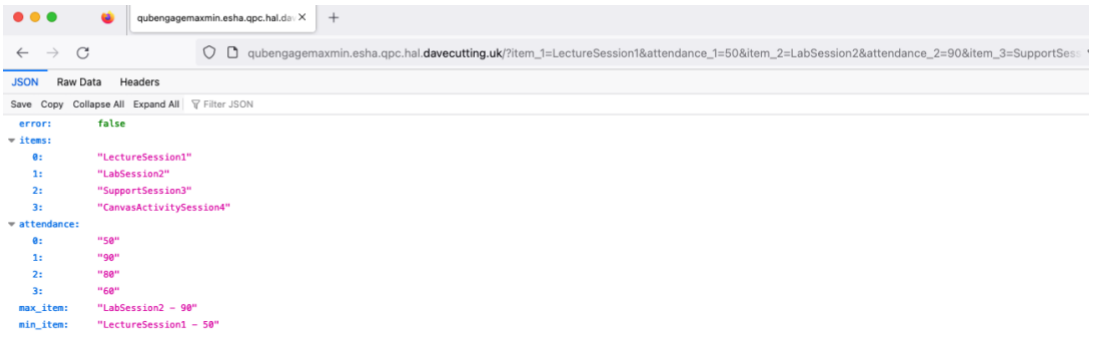
Repository: https://repository.hal.davecutting.uk/root/qubengage-maxmin

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

An example of direct call to the service: (tried on Firefox!)

http://qubengagemaxmin.esha.qpc.hal.davecutting.uk/?item_1=LectureSession1&attendance_1=50&item_2=LabSession2&attendance_2=90&item_3=SupportSession3&attendance_3=80&item_4=CanvasActivitySession4&attendance_4=60

JSON result should display the following:



qubengagemaxmin.esha.qpc.hal.davecutting.uk/?item_1=LectureSession1&attendance_1=50&item_2=LabSession2&attendance_2=90&item_3=SupportSes

```
error:        false
▾ items:
    0:        "LectureSession1"
    1:        "LabSession2"
    2:        "SupportSession3"
    3:        "CanvasActivitySession4"
▾ attendance:
    0:        "50"
    1:        "90"
    2:        "80"
    3:        "60"
  max_item:   "LabSession2 - 90"
  min_item:   "LectureSession1 - 50"
```

- ***QUBengage-sort*** – this is a PHP based container that hosts PHP scripts to process one of the requests from the frontend, which is to sort all the sessions/activities that are entered in the frontend GUI in descending order of their attendance. The container takes the input as HTTP GET parameters ('item_1', 'item _2', 'item _3', 'item _4', 'attendance_1', 'attendance _2', 'attendance _3', 'attendance _4') and returns the results ('error','items','attendance','sorted_attendance') in JSON format. , CI testing is not implemented in this version, and it's expected from the students to implement the CI testing to check the correctness of the results.

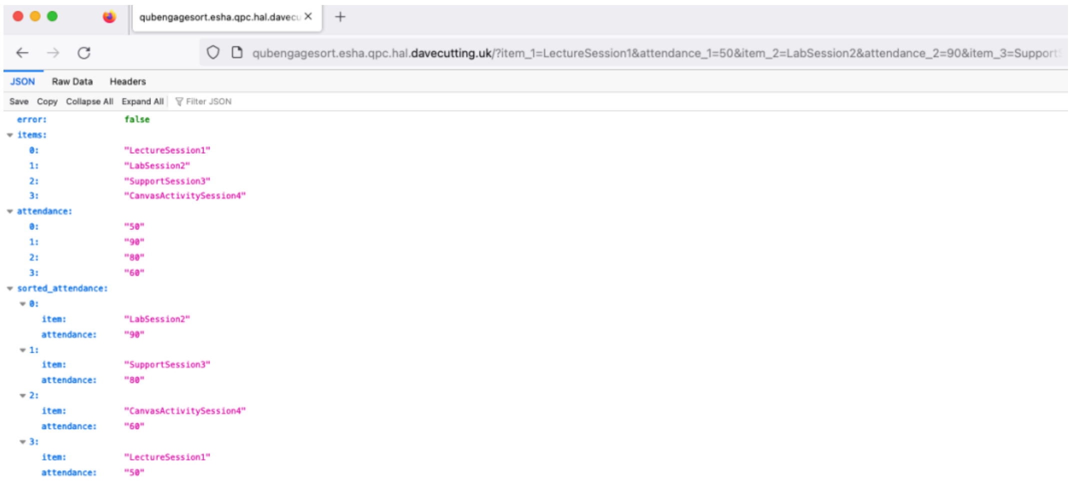
  Repository: https://repository.hal.davecutting.uk/root/qubengage-sort

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

  An example of direct call to the service: (tried on Firefox!)

  http://qubengagesort.esha.qpc.hal.davecutting.uk/?item_1=LectureSession1&attendance_1=50&item_2=LabSession2&attendance_2=90&item_3=SupportSession3&attendance_3=80&item_4=CanvasActivitySession4&attendance_4=60

JSON result should display the following:



```
qubengagesort.esha.qpc.hal.davecutting.uk/?item_1=LectureSession1&attendance_1=50&item_2=LabSession2&attendance_2=90&item_3=Support
```

```
error:               false
▼ items:
    0:               "LectureSession1"
    1:               "LabSession2"
    2:               "SupportSession3"
    3:               "CanvasActivitySession4"
▼ attendance:
    0:               "50"
    1:               "90"
    2:               "80"
    3:               "60"
▼ sorted_attendance:
  ▼ 0:
      item:          "LabSession2"
      attendance:    "90"
  ▼ 1:
      item:          "SupportSession3"
      attendance:    "80"
  ▼ 2:
      item:          "CanvasActivitySession4"
      attendance:    "60"
  ▼ 3:
      item:          "LectureSession1"
      attendance:    "50"
```

Currently, the ***QUBengage App*** 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, docker containers, microservices, 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. You are required to modify and extend this QUBengage App as described in the following tasks.

Make sure you read and understand the instructions carefully, especially those on submission! **The report is what is marked (code and the video will only be sample checked as needed to verify a claim made in the report or for moderation – anything in the code or video that is not in the report will not be marked)**.

1.1 Tasks

A. New Functionalities using Container Services

Add a minimum of **four** new functions to the application including the changes to the front-end as well as back-end implementations for each. See the assessment criteria for more information but in general higher marks are given for use of new languages (beyond those provided) and completeness including testing with CI pipelines. Points can also be awarded for use of novel technologies, i.e., deployment using FaaS for one function as an example (new technology or language points are only awarded once, i.e., if you use FaaS twice it only counts as novel once even if you use different languages or providers).

Three mandatory functions to add to the QUBengage App are:

I. **Total Attendance Hours** - calculate the total hours by adding the hours of attendance from all the sessions/activities.

II. **Student Engagement Score** – calculate the engagement score using the following equation inside a function:

$$StudentEngagementScore = \frac{lec * \ lec_w}{lec_{total}} + \frac{lab * \ lab_w}{lab_{total}} + \frac{supp * \ supp_w}{Supp_{total}} + \frac{can * \ can_w}{can_{total}}$$

$Lec \ = \ total\ attendance\ hours\ in\ lecture\ sessions$
$Lab \ = \ total\ attendance\ hours\ in\ lab\ sessions$
$Supp \ = \ total\ attendance\ hours\ in\ support\ sessions$
$Can \ = \ total\ student\ activity\ hours\ on\ canvas$
$Lec_{total} \ = \ total\ hours\ in\ lecture\ sessions$
$Lab_{total} \ = \ total\ hours\ in\ lab\ sessions$
$Supp_{total} \ = \ total\ hours\ in\ support\ sessions$
$Can_{total} \ = \ total\ expected\ activity\ hours\ on\ canvas$
$lec_w = weight\ for\ lecture\ sessions\ (e.g.\ 0.3)$
$lab_w = weight\ for\ lab\ sessions\ (e.g.\ 0.4)$
$supp_w = weight\ for\ support\ sessions\ (e.g.\ 0.15)$
$can_w = weight\ for\ canvas\ activities\ (e.g.\ 0.15)$
$Total\ weights \ = \ 1.0$

III. **Risk of Student Failure** – identify if the student is at risk to fail (in the academic exam/assignments) in the module by checking the student engagement score (student can use function (d) to get that score); if the score is below the cut-off score (provided as an input), the student is at risk of failure.

Note: total hours for lecture/lab/support sessions and for the canvas activities can be imaginary values.
The buttons (in the frontend GUI) that can be linked to these three services are in grey colour. In addition to the three mandatory ones, you must implement one more function of your choice. Please note, only **four** functions will be marked, handing in work with more than four functions will result in the first four being marked and others ignored.

B. Improvement to Current Implementation

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

- No error handling in the frontend (what if a backend service returns an error result?)
- No error handling by either of the provided backend example services (for example, if the user inputs, i.e., session/activity names and attendance are empty and an attempt is made to check maximum and minimum attendance hours, no alert message is displayed, or if any other error condition occurs, such as attendance hours are not integers, or attendance hors more then the total assigned hours, etc.; then these errors are not reported in the frontend thus making it difficult to identify the problems.
- No CI testing for the provided backend services (sort and maxmin), i.e., there is no unit tests on the functions as well as there is no HTTP request to check web API functionalities.
- Static configuration of routes to services and/or proxies in the frontend (they are lines of code inside the source, would be much better to have some external configuration file or service to work). This applies only to the frontend services and/or proxies.

Plan and implement improvements to address all of the above, or other shortcomings or issues you perceive within the provided system. 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

Currently each service endpoint has a custom URL configured in the frontend client. For this challenge you will build a self-contained reverse web proxy as a container to which the frontend will pass all calls to (including any variables).

This proxy will then itself hold the specific endpoints for different methods and make the HTTP request to the actual service before returning the answer.

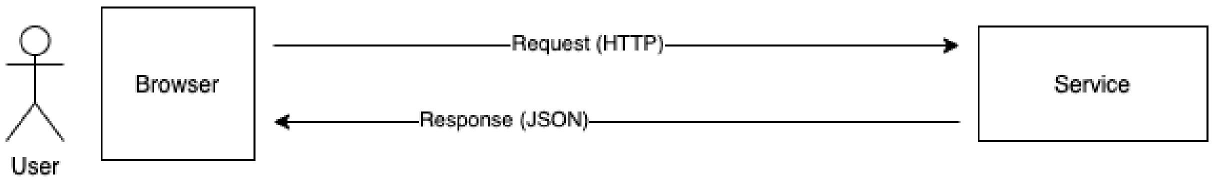


Figure 1 Requests and response without proxy (direct to service)

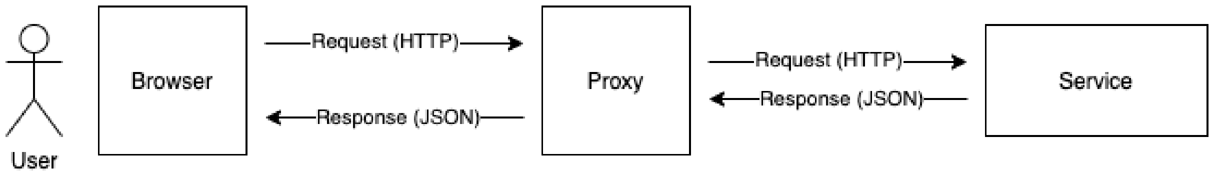


Figure 2 Requests and response with proxy, proxy makes request to relevant service

For full marks this service should be highly configurable (how do we update the end points or add new ones, can this happen dynamically while the service is running for instance), and support some service discovery options (what methods are available or automatic registration of new services 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 functionality. Service discovery features could include automatic configuration using these systems or the proxy having an ability to probe a URL and discover what options are available for example.

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

D. Frontend Service Failure Handler

The frontend currently stores a single URL for each service (after Task C this will be a single URL for the proxy). This challenge is to update the frontend to support multiple URLs for each service/proxy. If a specific endpoint (URL) 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.

As mentioned in combination with Task C 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), should you implement this handler in the proxy it will not count for marks against Task C, this only relates to the browser.

E. Monitoring and Metrics

Monitoring and testing service – Implement a separate service (this could be a container or other service) that when used makes HTTP requests with random session names and attendance hours 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).

F. Stateful Saving of Current Values

Currently there is no way to store the current values of the sessions/activities, attendance hours, and the results that are produced by the QUBengage App. For this section you should implement, using any technology (containers, FaaS, database, etc) you like a way to:

- Save the sessions/activities, attendance hours, and the results (and display) an identifier for the saved items.
- Provide the option to enter an identifier and thus recall the saved items to the front-end.
- For maximum marks, persistence over multiple sessions is preferred, that is, not using just browser storage.

G. Multi-Vendor Architecture

Complete a justified design (you are not required to implement) for how the QUBengage App could be fully deployed onto public multiple cloud vendors perhaps with a common broker/router. Include diagrams and text as appropriate demonstrating how your design would ensure utmost resilience and sustain the capacity to execute the expected operations even in the presence of an unforeseen catastrophic event.

H. Acknowledgement

Important Note: In the report you need to provide some information as appropriate in the "Anything else to highlight" sections of each of the tasks (Task A to Task G) on how you came to certain answers or found information. This would include an analysing your use of AI tools and also other third-party sources of information all of which are available to be used and acknowledged in your work. Some examples would include answering questions like "what enhancements and modifications did you implement in developing this application beyond the suggestions and guidance provided by ChatGPT/any other third-party sources? Could you share specific examples for each of the tasks of how your creative input and expertise were integrated to improve the overall analysis?"

1.2 Deployment Environment

As it stands the current QUBengage 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 4 x 7% (total of 28% for A)	Perfect implementation of function in new language or paradigm with excellent CI tests fully covering functionality.	Excellent implementation of function in new 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 new 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 4 * 3% (total of 12% 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 10%	Excellent and perfectly demonstrated implementation of custom proxy router to industry standard incorporating dynamic configuration, service discovery and advanced features.	Excellent and very well demonstrated implementation of custom proxy nearly to an industry standard incorporating some advanced features well implemented such as dynamic configuration and service discovery.	Very good implementation of proxy and well demonstrated incorporating some service discovery features, some significant best practices and advanced features but with some limitations.	Good implementation of proxy, clearly demonstrated and meeting the goals set well (and beyond simple hard coded requests) but with some weaknesses in design or implementation.	Proxy meeting (or almost meeting) the requirements set for the custom proxy but with major weaknesses in design or implementation.	Solution failing to meet the requirements, containing significant errors or lacking in functionality.
Task D. Frontend Failure Handling 5%	Excellent and perfectly demonstrated implementation of challenge to a near industry standard incorporating best practices and documentation as appropriate.	Excellent and very well demonstrated implementation of challenge to a very high standard incorporating best practices and documentation as appropriate.	Very good solutions well demonstrated incorporating some significant best practices and documentation as appropriate.	Good solutions clearly demonstrated and meeting the goals set well but with some weaknesses in design or implementation.	Solutions meeting (or almost meeting) the challenge set but with major weaknesses in design or implementation.	Solutions failing to meet the requirements, containing significant errors or lacking in functionality.

Task E. Monitoring & Metrics 15%	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.
Task F. Stateful Saving 5%	Excellent fully-featured saving and recall including error handling and exceptions.	Excellent saving and recall including error handling.	Save and recall working well with some quality features.	Saving and recall working correctly.	Saving and recall almost working with some errors.	Fails to meet requirement to save or recall and with no or little attempt.

Task G. Architectural Design 10%	Excellent fully justified designs showing an appreciation for the core issues of high-availability over multiple vendors.	Excellent designs with strong justification showing appreciation for the core issues.	Generally excellent designs and justification with some issues or weaknesses.	Good designs and justification with some weaknesses throughout or significant weaknesses in one area.	Fair designs and justification but with weaknesses or issues throughout.	Poor quality or badly justified designs showing no real understanding of the core issues.
Task H. Acknowledgemen t 15%	Acknowledgemen t is perfect and outstanding analysis covering all aspects.	Acknowledgemen t is excellent with detailed examples and analysis.	Acknowledgemen t is very good with detailed examples and analysis.	Acknowledgemen t is good with a good number of examples.	Acknowledgemen t is acceptable but lacks a good number of examples.	Acknowledgemen t is either not provided or just a line is included saying what is being used with no detailed analysis and no examples at all.

3. Feedback

Feedback in the form of marks will be provided as soon as practicable after submission with the expectation that marking will be complete (and marks provided).

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.

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 ***<your_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. The video must be named as ***<your_student_number>.mp4***
    - 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. This zip file should be also saved with your student number.

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.

Please note: the primary purpose of the video is for validation and for external examination. The report is the marked component so you must ensure you evidence everything in the report, anything included in the video but not in the report can not be marked. The marker may not even view the video.

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

Module "Drop-in" – CSC3065 offers a virtual drop-in session every Monday 1200-1400. Our module connected learning tutor (who will be involved in the marking) and a few PhD demonstrators 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.

Office Hours/Other Appointments – Esha Barlaskar has office hours available every week. Appointments can be booked via the Office Hours link on Canvas.

Open Door/Other Appointments – Esha Barlaskar operates an "open door" policy when she is available. She is happy to make ad-hoc meetings for individuals or groups as needed outside of the office hours above. Drop her an email or message via teams if you wish to discuss anything.