2.1 Submission process

Table of Contents

Video

Important! Ensure that your video is actually submitted. You need to click the *Submit* button after uploading your video.

Your video must be a screen capture at a high enough quality that the marker can make out details such as text. A recording from your phone is not acceptable.

The purpose of the video is to *demonstrate* functionality, not explain it. The response to criteria document is used for justifying choices. Keep things brief in the video; you only need to describe what you are demonstrating.

- Your video must be submitted to **CAB432 A2 - Video (https://canvas.qut.edu.au/courses/20367/assignments/202268)**
- Only one partner needs to submit.
- Your video must be no longer than 8 minutes

- Your video must follow the structure below and demonstrate your application's functionality as requested
- Your video must be a screen capture at a high enough quality that the marker can make out details such as text. A recording from your phone is not acceptable.
- The purpose of the video is to *demonstrate* functionality, not explain it. The response to criteria document is used for justifying choices. Keep things brief in the video; you only need to describe what you are demonstrating.

Creating your video:

- You can record short videos separately and edit them together
- If some operation takes a long time (eg. uploading a video) you can edit it down to save time
- **OBS Studio (https://obsproject.com/)** is a reasonable choice for screen capture, but you can use other software if you like.
- If you are working with a partner, you can record the video together over Zoom and use the recording for your submission
- Canvas has a built in option for screen capture which you can access through the submission page. Choose the *Record media* option.

Video structure:

Please demonstrate the following features of your application. Note, the order is important, for both a logical progression through features of your application and our sanity while marking in this order. The order matches the marking rubric.

Core - First data persistence service

- State which service is used and what data is being stored (briefly!) eg. "S3 is used for storing video files."
- In the AWS console:
  - show the details page for the service instance(s) used in your application
  - show objects/records being added to the service in response to application actions (eg. show a video file in S3 after being uploaded through the application). If a single action modifies data in more than one service then you can demonstrate them together.

Core - Second data persistence service

- State which service is used and what data is being stored (briefly!) eg. "DynamoDB is used for storing video metadata."
- In the AWS console:
  - show the details page for the service instance(s) used in your application
  - show objects/records being added to the service in response to application actions (eg. show a video file in S3 after being uploaded through the application). If a single action modifies data in more than one service then you can demonstrate them together.

Additional - Third data service

- State which service is used and what data is being stored (briefly!) eg. "EBS is used for storing machine learning models."

- In the AWS console:
  - show the details page for the service instance(s) used in your application
  - show objects/records being added to the service in response to application actions (eg. show a video file in S3 after being uploaded through the application). If a single action modifies data in more than one service then you can demonstrate them together.

Additional - S3 Pre-signed URLs

- State which application tasks use pre-signed URLs. (briefly!) eg. "Video files are uploaded and downloaded with pre-signed URLs."
- Show the web client accessing S3 using pre-signed URLs, eg. by showing the requests in the Network tab of the developer tools.

Additional - In-memory cache

- State which service is used and what for (briefly!) eg. "memcached is used to cache requests to the RDS instance."
- Demonstrate the cache being used in response to application actions. For example:
  - Install `libmemcached-tools` on an EC2 instance
  - Use `memcflush` to flush all entries from the cache
  - Perform an action in the application that will create an entry in the cache
  - Use `memcdump` to show the newly added entries
  - If using Redis, you can use the `redis-cli` command from the `redis-tools` package to similar effect

Core - Statelessness

- This is addressed in the response document only. There is no need to address it in the video.

Additional - Graceful handling of persistent connections

- This is addressed in the response document only. There is no need to address it in the video.

Core - Authentication with Cognito

- Show the user pool in the AWS console
- Demonstrate creating a new user in the app and the new user showing up in the user pool list
- Demonstrate verifying the email address in your app and the user showing up as verified in the user pool list
- Demonstrate logging in to your app using the new user

Additional - Cognito multi-factor authentication

- Demonstrate logging in to your application with multi-factor authentication

Additional - Cognito federated identities

- Demonstrate logging in to your application with federated/social login with each provider your application supports (eg. Google, Facebook, ...)

Additional - Cognito groups

- Demonstrate group functionality, for example by changing a user's group membership and showing how their access to application functionality changes.

Core - DNS with Route53

- Demonstrate the application being accessed using the custom subdomain

Additional - Parameter store

- Show the relevant parameters in the AWS console
- Briefly state what the parameters are used for (eg. "This parameter is used to store the application URL for the web client fetch requests.")
- Briefly show the source code where the parameters are used.

Additional - Secrets manager

- Show the relevant secrets in the AWS console (you don't need to show the values)
- Briefly state what the secrets are used for (eg. "This secret is used to store API keys for the YouTube API.")
- Briefly show the source code where the secrets are used.

Additional - Infrastructure as code

- This is addressed in the response document only. There is no need to address it in the video.

Additional - Other (with prior approval only)

- Briefly demonstrate the functionality.

Additional - Other (with prior approval only)

- Briefly demonstrate the functionality.

Code submission requirements

- Your code must be submitted to **CAB432 A02: Cloud services exercises (Code) (https://canvas.qut.edu.au/courses/20367/assignments/203891)**
- Your submission should not include `.git`, `node_modules`, `venv`, `__MACOSX` or other files that are not required for building and running your application (eg. video files left over from testing)
- Your code submission must include the `A2_response_to_criteria.md` document with the filename unchanged and in the original Markdown format.

These requirements will make it easier for the markers to navigate your source code and find relevant information.

- If you have trouble submitting you code, please ensure that it is no more than 100Mb in size and that you haven't included any extraneous files or directories. (such as those mentioned above.)

Response to criteria

- Your response to criteria document must be submitted along with your code
- The filename must be `A2_response_to_criteria.md` and in the original Markdown format.

- Do not delete or rearrange sections. If you did not attempt a criterion, leave the corresponding section blank

The template is available at **A2_response_to_criteria.md (https://canvas.qut.edu.au/courses/20367/files/6976160/download)**