ASSESSMENT AND INTERNAL VERIFICATION FRONT SHEET (Individual Criteria)

(Note: This version is to be used for an assignment brief issued to students via Classter)

Course Title	**B.Sc. (Hons.) Software Development**		**Lecturer Name & Surname**	**Ryan Attard** **David Deguara**	
Unit Number & Title	**ITSFT-606-1620-Programming for the Cloud**				
Assignment Number, Title / Type	**Building a cloud based website**				
Date Set	10/3/2022	**Deadline Date**	8/5/2023		
Student Name		**ID Number**		**Class / Group**	

Assessment Criteria		Maximum Mark
AA2.1	Choose and apply appropriate data storage solution(s) for a given scenario	7
Ku3.1	Construct a scheduled job in a given scenario	5
KU4.1	Construct a service while making use of a cloud service feature that can be consumed by a 3rd party	5
AA4.2	Analyse a number of requirements with respect to your application needs (such as caching, website availability, etc) and configure these in your context	7
KU4.3	Arrange a way how to easily upload your artifacts on the cloud	5
AA4.4	Apply a number of security configurations to secure your artifacts (e.g. APIs, Data,etc)	7
SE2.3	Establish and Design a way how your application can handle large data delivery	10
SE4.6	Design and develop a web application that makes use of a number of cloud services while also consuming an implemented API service residing in a different location	10
Total Mark		56

Notes to Students:

- This assignment brief has been approved and released by the Internal Verifier through Classter.
- Assessment marks and feedback by the lecturer will be available online via Classter (Http://mcast.classter.com) following release by the Internal Verifier
- Students submitting their assignment on Moodle/Turnitin will be requested to confirm online the following statements:

Student's declaration prior to handing-in of assignment

- ❖ I certify that the work submitted for this assignment is my own and that I have read and understood the respective Plagiarism Policy

Student's declaration on assessment special arrangements

- ❖ I certify that adequate support was given to me during the assignment through the Institute and/or the Inclusive Education Unit.
- ❖ I declare that I refused the special support offered by the Institute.

Assignment Guidelines

Read the following instructions carefully before you start the assignment. If you do not understand any of them, ask your invigilator.

- This assignment is a HOME assignment.
- Copying is **Strictly Prohibited** and will be penalised according to disciplinary procedures.
- Use the given cloud account responsibly
- Deadline: May 8, 2023 @ 8am
- Note: There are marks associated with every technology. You can still use alternative ways to achieve the same outcome to meet other criteria but marks will be lost for that criterion addressing the requested technology.
- Submission must be done through Moodle
  - Code must be supplied via a Git repository link
  - Document/Text containing the public link to your website (if you managed to answer the relevant task)

This website allows the user to upload mp4 video files and transcribe them.

Note: all the following functions must be implemented using the Cloud services that were introduced in the course. Other solutions which bypass cloud technologies are not considered acceptable.

KU4.3 - Arrange a way how to easily upload your artifacts on the cloud

- A user through his/her profile can upload a video/movie file. Files should be stored in a bucket (standard type); [2]
- Find a way to show a progress bar with the status of the upload which is synchronized with the actual upload; [3] hint: you have to use jquery/javascript

AA2.1 - Choose and apply appropriate data storage solution(s) for a given scenario

- Some info about the movie file being uploaded such as, name of file, date uploaded, owner, in which bucket it is located/uri, thumbnail image and status should be stored in a NoSql database. [3]
- Find the best structure for your data so that when the user logs in he/she is shown only his/her data including the thumbnail image (shown properly NOT AS TEXT) and buttons to allow the user to Transcribe or Download the file. [2]
- User may download back the file anytime he wants before it is Transcribed. A note of every date and time when it was downloaded should be recorded in a nested collection under the video document [2]

SE2.3 - Establish and Design a way how your application can handle large data delivery

- When the user selects a movie to be transcribed, the job has to be published (queued) to a *topic* using Pub/Sub. [2]
- Hosting:
    - Use 2 containers/2 vms registered in the Container registry together with Google Cloud Run to host and run these:
        - The frontend (Website)
            - Upload works [1]
            - Listing works [1]
            - Srt can be downloaded [1]
            - Login works without issues [1]
        - The backend engine (processing the transcription). [4] – 4 marks cannot be split here so they are only given in case the transcription process starts until it is saved back in the database.

SE4.6 - Design and develop a web application that makes use of a number of cloud services while also consuming an implemented API service residing in a different location

In addition to the above criterion SE2.3 see the below tasks.

Subscribed to the same queue (topic) as in Se2.3(a) should be another service/application developed using any language of your choice (note: python executes the following processes faster than C#) to download the data queued, and starts transcribing the movie using Google Speech-to-Text and save the transcription back in the database (noted in AA2.1) and making the transcription data downloadable

Process is divided as follows: *Transcribing a movie file requires you to*

- Convert the movie file to audio (e.g. .wav) – this can be done either by a 3rd party service or using ffmpeg [2]
- Convert the .wav to .flac [2]
- Upload the .flac in a storage location (e.g. another bucket) [2]
- Call the right code to commence the transcription and save back the transcription successfully in the NoSql database. [2] https://codelabs.developers.google.com/codelabs/cloud-speech-text-csharp#5
- Update the status of the file in the NoSql database and it reflects in the listing of data [1]
- Delete the .flac file which is now no longer needed [1]

  The above 4 points have to be run in a different service/app NOT in the Web Application you've been building for the upload and listing of information, triggered by a cron http request – see ku3.1. Marks related to this are assessed in se2.3

KU4.1 - Construct a service while making use of a cloud service feature that can be consumed by a 3rd party

- Build a feature where the transcription output from Se4.6 can be converted in an SRT file. The feature has to be triggered using either a Pub/Sub forwarded to a cloud-event function or an Http Function hosted on the cloud. Result has to be stored in the NoSql database and can be downloaded anytime by the user. Once done, it needs not run again when the user clicks the button "Generate SRT" but downloaded from the database straight away [5]

AA4.2 - Analyse a number of requirements with respect to your application needs (such as caching, website availability, etc) and configure these in your context

- The user must login/ register using Oauth 2.0 for the login; Identity classes which facilitate relational database access are not to be used [3]
- Authorization on all the user accessed actions has to be applied; [1]
- SE4.6 must be logged in detail (after each executed step) on the cloud; [3]

AA4.4 - Apply a number of security configurations to secure your artifacts (e.g. APIs, Data,etc)

- Secret key for Oauth login (AA4.2) should be loaded from Secret Manager [3.5]
- SSL and a top domain should be configured on the hosted website [You may use noip.com] [3.5]

KU3.1 - Construct a scheduled job in a given scenario

Construct a cron job using Cloud Scheduler which calls an HTTP function/method (once every few minutes), to complete se4.6.