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	15/4/2024	**Deadline Date**	19/5/2025		
Student Name	Lee Xerri	**ID Number**	446203L	**Class / Group**	SWD-6.3A

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	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 19, 2025
- 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 vle
  - https://vle.mcast.edu.mt/mod/assign/view.php?id=62496

This website is about Cloud-Based Incident Management System for IT Support. The following is a user story gathered from the client to better understand how it works.

As an **employee**, Sarah often faces technical issues, like her email client not syncing or her laptop running slow. Instead of emailing IT, she logs into the system using her **company Google account (OAuth Login)**.

- **Scenario: Reporting an Issue** ○ Sarah logs in and fills out a form describing her issue: *"My email is not syncing with Outlook."* ○ She selects **priority level** (High, Medium, Low) and attaches two screenshots.
    - ○ After submitting, she sees a confirmation message and can track her ticket's status.

Mike is part of the IT support team, responsible for solving employee issues.

- **Scenario: Managing Assigned Tickets** ○ Mike logs in and sees a **dashboard** listing open tickets all marked as High Priority.
    - ○ He clicks on Sarah's ticket and checks the **issue details and attached screenshots** (stored in Cloud Buckets).
- **Scenario: Updating Ticket Status** ○ If the issue is resolved, Mike marks the ticket as **"Closed"**.
    - ○ Ticket still shows on Mike screen until a week from date of opening passes.

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) A user (e.g. Sarah) through his/her profile can report an issue by opening a ticket. An ticket can be accompanied by a number of screenshots and hence the feature has to allow the user to upload a number of image file(s) [2]. Note: Files should be stored in a bucket which allows only a selection of email addresses/users to access them - permissions to be set at a later stage (Assessed in AA4.4);
b) Find a way to show a progress bar with the status of the upload which is synchronized with the actual upload(s); [3] hint: you have to use jquery/javascript

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

---

Note: Marks here are related to the saving of data only

a) Info about the ticket: title, description, date uploaded, user email address who is opening the ticket, images related (which can be more than 1), priority and a default status should queued in a PubSub topic called *tickets-topic*. Marks here are related to the saving of this info in the appropriate data storage [2]
b) Use priority as a filter attribute when storing the ticket in the *tickets-topic*: **high, medium, low** [1]
c) Files has to be stored in a bucket [2] and permissions applied (see aa4.4) – marks here are given only for the storage of the files in the bucket
d) In a NoSql database you have to keep a list of users and their roles being either a **User** or a **Technician** [2] Note: You can manually set a technician, everyone else is a user

Note: more marks are given to these if these same features work on the deployed website. See Se2.3.

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

---

The following features should work in the hosted website on the CLOUD; You may host in a container or a VM, upload to container registry/compute engine and run it (if container it has to be run using Google Cloud Run)

- Authentication [2]
- Upload of screenshots in buckets [2]
- Function (Se4.6) is hosted on the cloud and working [3]
- Closing off a ticket [3]

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

---

An Http-Function has to be developed to act as a manager to filter and sort out these prioritized tickets. Therefore

a) Create an Http Function which accesses the *tickets-topic;*
b) It should start by reading the **High** prioritized tickets, using the filter attribute applied in AA2.1 (b). If it finds any, acknowledges them and… [2]
c) It should save them in the redis-cache [assessed in KU4.1 (b)];
d) Also it should send an email (tip: Use MailGun) to the technicians about this ticket; [2] Sending email will also be assessed in KU3.1.
e) Log into Google Cloud Logging any emails sent while using the ticket id as a **correlation key**, also saving in the log a timestamp and the recipient [2]
f) If the next time it runs there are no **High** priority tickets in the topic and all existing **high** priority tickets have been resolved, the function should start fetching **Medium** priority tickets and apply the same sequential procedure as in (b), (c), (d), (e). [2]

g) If the next time it runs, there are no **Medium** priority tickets and all existing **medium** priority tickets have been resolved, the function should be intelligent enough to start fetching **Low** priority tickets and apply the same sequential procedure as in (b), (c), (d), (e). [2]

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

---

a) Read from cache: When a technician logs in into his/her account a list of tickets/reports which are ***not*** more than 1 week old OR are still marked "open" should be loaded from the cache and presented onto his screen – NOT from the database. [2]
b) Write in cache: When the Http-Function (described in SE4.6) is triggered, it should save them in the cache[3].
c) If there are tickets info in the cache, whereby ticket has been "resolved/closed" and it has been open for more than 1 week old, it should be removed from the cache and transferred in the NoSql database for record keeping. (Refer to AA4.2 (b)) [Assessed in AA4.2(b)]

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

---

a) The user must login/ register using Oauth 2.0 for the login; Identity classes which facilitate relational database access are not to be used since in SE4.6 a NoSql database has been opted for [2]
b) In his/her UI ,the technician must have the possibility to close off a ticket. Upon doing that cache must be updated as explained in KU4.1(c). Therefore if a ticket just closed has been opened for more than 1 week, it should be removed and transferred into a Tickets Archive in the NoSql Database, also stating the name of the technician who resolved the ticket. [4]
c) Two Authorization attributes must be applied and working: one on the Users' controller and one on the Technicians' Controller to allow the respective roles in [1]

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

---

a) Secret key for Oauth login (AA4.2) should be loaded from Secret Manager [3]
b) Screenshots in the bucket should be assigned different emails as readers, i.e. who uploaded the file and all the emails of technicians [4]

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

---

a) Construct a cron job that triggers the Http Function every hour mentioned in SE4.6 [3]
b) Marks will also be given if the cron job is manually triggered and an email is received indicating any pending ticket [2]

Rubric

KU4.3	a) Submission of ticket [2] b) Progress bar working [3]	
AA2.1	• Ticket stored in pubsub [2] • Priority applied [1] • Files stored in bucket [2] • Roles stored in a nosql database [2]	
SE2.3	These features should be working in the hosted app ○ Authentication [2] ○ Upload of screenshots in buckets [2] ○ Function (Se4.6) is hosted on the cloud and working [3] ○ Closing off a ticket [3]	
SE4.6	a) Reading of tickets [2] b) Send an email[2] c) Logs[2] d) Same process applied to Medium tickets but after there are no high [2] e) Same process applied to Low tickets but after there are no more Medium and High [2]	
Ku4.1	• Read from redis accordingly and following the 1 week/ open conditions [2] • Write into the cache successfully from the function [3]	
Aa4.2	• User oauth login [2] • Close ticket and update cache and NoSql db [4] • Authorization on top of 2 controllers [1]	
AA4.4	• Secret key or any other type of key loaded from Secret Vaults [3] • Read permissions dynamically set on each file [4]	
KU3.1	• 1 working cronjobs [3] • Email(s) is/are sent containing pending tickets as a result of manually triggering the cron job [2]	