PROG2003: CLOUD SYSTEMS DEVELOPMENT

Summary

Title	Assessment 1
Type	Programming
Individual or Group	Individual
Due Date	Monday Week 3 11:59 pm AEST
Length	N/A
Weighting	40%
Academic Integrity	Contract cheating and any breach of the GenAI policy as outlined at the end of this document, is strictly prohibited. Any breach may have severe consequences. Please read carefully the "Academic Integrity" section below.
Submission	AWS application submission in Blackboard link. The application must be present in the AWS workspace.
Unit Learning Outcomes	This assessment task maps to the following ULOs: ULO1: integrate cloud APIs and services to design and develop cloud applications

Rationale

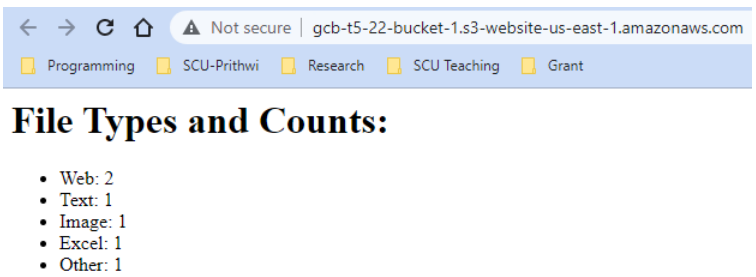
Learning and integrating cloud computing APIs would allow you to create real-world applications. This assessment particularly asks to utilise cloud storage API and its features to create a website to display a set of information. This is not a straightforward task, and you are required to understand the API features and the programs shown in the classes clearly. You must transfer and combine the knowledge they would learn in the class and from the material to complete this assessment. A real-world cloud developer would start learning fundamental cloud APIs, and create web application using them. This assessment asks you to do the same task.

Task Description

Assume that you are a cloud developer at Southern Cross University and tasked to develop a website that counts and displays the number of different types of files stored in an AWS cloud storage bucket (i.e., S3 bucket). This can be done with a backend app (i.e., an S3 app) that reads the files in an S3 bucket and counts them according to their file types (e.g., txt, xlsx, png, jpg). Later, the backend app "writes" the count information as a part of the HTML pages that are used to host the website. You will learn details about these in the class. Please see below the example screenshots of the website's home page.

Example scenario and problem

An example of how your website would look like is given below:



You can divide the whole project into two parts – Part A would be the website, and Part B would be the backend Cloud9 app.

Note: You will use an AWS account provided by the university for this project, not a personal account. This ensures that all resources and data are managed within the university's AWS ecosystem.

Task Instructions

Part A – Website

Complete the following tasks:

1. **Create a cloud storage service bucket (aligns with ULO1):**
   Create an S3 bucket and upload some test files of different types – web, text, image, excel, and others into the bucket. Configure the bucket with the appropriate "bucket policy" and "public access" to ensure public access to the website.

2. **Design a static website using cloud storage service (aligns with ULO1):**
   Create a static website with two pages – home and error, using corresponding HTML files (i.e., home.html, error.html). The home page will print (display) the number of files available within the S3 bucket for each file type. You need to consider the following types and respective file extensions.
   - Web (.html)
   - Text (.txt)
   - Image (.jpg)
   - Excel (.xlsx)
   - Other (.pdf, .xml, etc.)

   The home page must reflect any changes to the files (rename, add, or delete file) in the bucket. The error page will be displayed upon entering invalid website URL and will display a simple error message, e.g., "404: Page not found".

Part B – S3 App

Complete the following tasks:

1. **Develop S3 app by integrating cloud storage service API (aligns with ULO1):**
   Develop an S3 application to add dynamic functionalities to the static website. The S3 app must read the list of existing files, count them, and write this information into the corresponding html file used as the home page. The home page should dynamically update to show any modifications (such as renaming, adding, or deleting files) made in the bucket. The S3 app must be run after any changes to files in the bucket. You need to develop the following functionalities with detailed comments. Use separate class methods to implement the functionalities.
   a) Reading and counting the number of each file type (use the file types mentioned above).

b) Crafting HTML content for 'home.html'. This file will display data regarding the types of files and their quantities. All this information should be encapsulated within a formatted string according to HTML standards.
c) Create (write) the 'home.html' in the bucket, which will overwrite the existing 'home.html'.
d) Implement suitable exception handling mechanisms and utilise loops as needed.

Requirements

Your website and S3 app must fulfil the following requirements.

- The functionalities of the S3 app must be implemented with AWS SDK 2.x and Java (shown in tutorials).
- Names of the bucket and S3 app must be "yourscuusernamea1bucket" and "yourscuusernameA1App". You need to use a single bucket to host the website and develop S3 app.
- Your Cloud9 app must have three separate Java methods to implement reading/counting, preparing, and creating.
- You must implement the assignment in the UA-provided AWS account, personal AWS account will not be accepted.

Resources

Resources required to complete the assessment task

- Contents from Modules 1 and 2 of the unit site**.** You will learn how to create new file and list existing files, hosting static website during the tutorials.
- You need additional resource to craft the HTML content for the 'home.html' file. You can use an HTML list element to display/print the file counts and types to the home page. Please check this link: https://www.w3schools.com/html/html_lists_unordered.asp.

Task Submission

1. Download the final S3 app from Cloud9 workspace and submit the zipped app as "yourscuusernameA1App.zip" to the Blackboard submission link in the unit site. The app must also be present in the AWS workspace provided by the UA. The marker will test your app functionality from this workspace.
2. Fill out the *GenAI Usage Acknowledgment Form* and upload it with your zipped file via the Blackboard submission link in the unit site. This must be done even if you did not use GenAI for your assessment. Please make sure to read the **GenAI policy for Assessment** 1 at the end of this document.
3. Add the website URL to the comment field with your submission.

Note: Multiple submissions are allowed until the deadline.

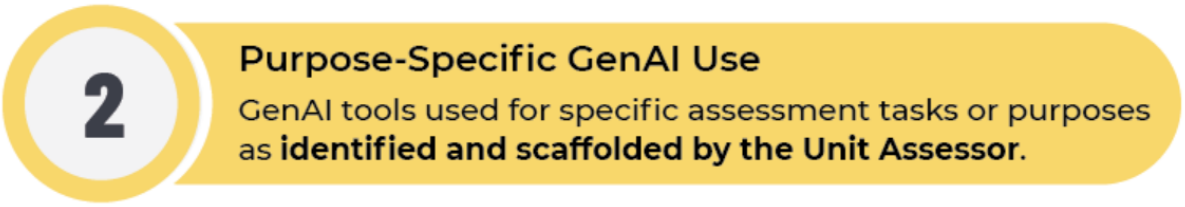
Academic Integrity

At Southern Cross University, academic integrity means behaving with the values of honesty, fairness, trustworthiness, courage, responsibility and respect in relation to academic work.

The Southern Cross University Academic Integrity Framework aims to develop a holistic, systematic and consistent approach to addressing academic integrity across the entire University. For more information, see: SCU Academic Integrity Framework

NOTE: **Academic Integrity breaches include** unacceptable use of generative artificial intelligence (GenAI) tools, the use of GenAI has not been appropriately acknowledged or is beyond the acceptable limit as defined in the Assessment, poor referencing, not identifying direct quotations correctly, close paraphrasing, plagiarism, recycling, misrepresentation, collusion, cheating, contract cheating, fabricating information.

GenAI policy for Assessment 1



Generative artificial intelligence (GenAI) tools, such as ChatGPT, **can be used for the following in Assessment 1:**

- You can use GenAI to get help with planning the assessment or setting up your environment.
- You can use GenAI to help write your code as long as you do the following:
    - Comment all of the code you created using GenAI
    - Acknowledge the code you created using GenAI in the *GenAI submission document*.

Please note that if your marker has any suspicion that you had help with your work or is not your own or you used GenAI without acknowledgement you will be asked to come to a meeting with your marker to explain your work. Any student who is unable to explain their code will be submitted for academic misconduct.

Special Consideration

Please refer to the Special Consideration section of Policy.
https://policies.scu.edu.au/document/view-current.php?id=140

Late Submissions & Penalties

Please refer to the Late Submission & Penalties section of Policy.
https://policies.scu.edu.au/view.current.php?id=00255

Grades & Feedback

Assessments that have been submitted by the due date will receive an SCU grade. Grades and feedback will be posted to the 'Grades and Feedback' section on the Blackboard unit site. Please allow 7 days for marks to be posted.

Assessment Rubric

Marking Criteria and % allocation	High Distinction (85–100%)	Distinction (75–84%)	Credit (65–74%)	Pass (50–64%)	Fail 0–49%
Integrate cloud storage service to set up and configure a static website, utilising cloud-based resources for web hosting (ULO1) – 5%	Configures a static website featuring exceptionally designed home and error pages, with an S3 bucket configured to perfection, showcasing a deep understanding of bucket policy and public access settings.	Configures a static website with well-structured home and error pages, and the S3 bucket is configured with considerable attention to detail in bucket policies and public access controls, though minor improvements could be made.	Configures a static website with functional home and error pages; the S3 bucket configuration is adequate, addressing the essential aspects of bucket policies and public access, with minor errors noted.	Configures a basic static website where the home and error pages meet the minimum criteria; the S3 bucket's configuration demonstrates an understanding of the key aspects of bucket policies and/or public access, though several errors are present.	Attempts to configure a static website; however, home and error pages are either incomplete or improperly configured. The S3 bucket setup lacks correct configuration of bucket policies and public access, failing to meet the project criteria.
Integrate cloud storage service to design a static website (ULO1) – 10%	Develops a fully functional website featuring a home page that accurately counts and dynamically updates for each file type in the bucket upon modifications, alongside an error page that precisely displays the appropriate error message.	Develops a functional website where the home page accurately tracks and updates file counts in the bucket, despite minor presentation issues. The error page conveys error messages effectively, with some presentation flaws.	Develops a satisfactory website where the home page generally reflects the count for each file type in the bucket and updates with modifications, though moderate discrepancies in the displayed information in home and error pages.	Develops a basic website with a home page that displays the file counts for each type in the bucket and attempts to update with file changes, though not always accurately. The error page includes an error message that is not fully correct.	Attempts to create a website; however, the home and error pages display inaccurate information with no reflection for any changes in files, failing to meet the project criteria.
Develop S3 app using cloud storage service (ULO1) – 40%	Develops an app that executes flawlessly without compile or runtime errors, accurately counts file types, generates detailed HTML content with count	Develops an app that runs smoothly with no compile or runtime errors, reliably counts file types, produces detailed HTML content including count data, and	Constructs an app that runs without compile or runtime errors but offers limited functionality, with one key feature not operating correctly, indicating a foundational	Builds an app free from compile or runtime errors, yet it exhibits restricted functionality due to several key features malfunctioning, suggesting a basic	Attempts to develop an app but encounters significant compile or runtime errors, failing to meet the project's core requirements.

	information, and creates a home page using this HTML.	constructs a home page with this HTML, with some minor issues in presenting information.	level of performance yet room for improvement in feature implementation.	operational level with significant scope for enhancement in feature execution.	
Implement S3 app using cloud storage service API (ULO1) – 45%	Implements an S3 app with exceptional skill, accurately listing file types and embedding file counts in HTML. The app demonstrates advanced implementation techniques using distinct Java methods, effective loops, and comprehensive exception handling. Source code is extensively commented, offering clear and insightful explanations of functionality.	Implements an S3 app with high competence, effectively listing file types and embedding file counts in HTML. Utilises well-structured Java methods, loops, and exception handling to ensure app robustness. Source code comments are detailed, providing good clarity on the app's functionality.	Implements an S3 app with good proficiency, listing file types and embedding file counts in HTML with minor inaccuracies. Java methods, loops, and exception handling are used appropriately to maintain app stability. Source code comments are adequate, explaining the functionality sufficiently.	Implements an S3 app that meets basic requirements, listing file types and embedding file counts in HTML with some errors. Java methods, loops, and exception handling are employed but lack refinement, affecting the app's robustness. Source code comments are present but lack depth in explaining functionalities.	Attempts to implement an S3 app, but fails to accurately list file types and embed file counts in HTML. Java methods, loops, and exception handling are poorly utilised, compromising the app's functionality. Source code comments are sparse or unclear, failing to elucidate the app's operations adequately.

Description of SCU Grades

High Distinction:

The student’s performance, in addition to satisfying all of the basic learning requirements, demonstrates distinctive insight and ability in researching, analysing and applying relevant skills and concepts, and shows exceptional ability to synthesise, integrate and evaluate knowledge. The student’s performance could be described as outstanding in relation to the learning requirements specified.

Distinction:

The student's performance, in addition to satisfying all of the basic learning requirements, demonstrates distinctive insight and ability in researching, analysing and applying relevant skills and concepts, and shows a well-developed ability to synthesise, integrate and evaluate knowledge. The student's performance could be described as distinguished in relation to the learning requirements specified.

Credit:

The student's performance, in addition to satisfying all of the basic learning requirements specified, demonstrates insight and ability in researching, analysing and applying relevant skills and concepts. The student's performance could be described as competent in relation to the learning requirements specified.

Pass:

The student's performance satisfies all of the basic learning requirements specified and provides a sound basis for proceeding to higher-level studies in the subject area. The student's performance could be described as satisfactory in relation to the learning requirements specified.

Fail:

The student's performance fails to satisfy the learning requirements specified.