PROG2003: CLOUD SYSTEMS DEVELOPMENT

Summary

Title	Assessment 2
Type	Programming
Individual or Group	Individual
Due Date	Monday 3 June 11:59 pm AEST (Week 6 Monday)
Length	N/A
Weighting	50%
Academic Integrity	Contract cheating and any use of GenAI, such as ChatGPT, in this assignment are 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 ULO2: analyse cloud computing problems to compare and propose solutions ULO3: design and evaluate cloud computing systems

Rationale

Learning cloud computing APIs and integrating them would allow you to create real-world applications. This assignment particularly asks to utilise cloud storage, database, and serverless computing APIs and their features. The purpose of the assignment is to test if you can develop a serverless computing system that can automatically interact with cloud storage and databases. Besides, this assessment will also check your skill in creating data processing pipeline. 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 learn key cloud APIs, and create advanced cloud application using them. This assessment asks you to do the same task.

Task Description

Assume that you are a cloud engineer at Harvey Norman and tasked to develop a serverless compute application to collate all reviews of their product items. The application is intended to get notified when a particular text or json object (file) is created in a cloud storage bucket. Upon getting notified, the application must read the content within the created object and insert the content into a cloud database table. Apart from this you need to develop a data processing pipeline using a cloud data processing service.

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 – Serverless compute application

Complete the following tasks:

1. **Create a cloud storage service bucket and database table (aligns with ULO1):**
   Create an S3 bucket. You only can upload JSON and TXT files in this bucket. Create a DynamoDB table, "ProductReview" for storing the below data attributes.

   Identifier, *Product name*, *Price* (in AUD), *Review comment*, and *Rating* (out of 5)

   Use appropriate data types and attribute names for the database table accordingly. The TXT and JSON files you will upload must contain records containing values for all these attributes (except Identifier), in text and JSON format.

   For example, the JSON file content can be like this.

   [{"ProductName": "Sony TV", "Price": 12000, "Review": "I loved this product, have been using for 5 years, no issue", "Rating": 4.85},{"ProductName": "Bravia Microwave", "Price":350, "Review": "Good value, does the job", "Rating": 4.12},{"ProductName": "Lenovo Thinpad", "Price": 2400, "Review": "Not happy with, had to return", "Rating": 2.45}]

   The TEXT file content can be like this, as plain text.

   ProductName: Sony TV, Price: 12000, Review: I loved this product, have been using for 5 years, no issue, Rating: 4.85; ProductName: Bravia Microwave, Price:350, Review: Good value, does the job, Rating: 4.12; ProductName: Lenovo Thinpad, Price: 2400, Review: Not happy with, had to return, Rating: 2.45

2. **Create serverless compute service (aligns with ULO1, ULO3):**
   Create a Lambda function (in AWS Lambda) with an appropriate execution role and a "create object" trigger for the above S3 bucket, which will work as the serverless compute application. A Lambda function executes a piece of Java program known as "handler".

3. **Design and develop handler application (aligns with ULO1, ULO3):**
   The program a Lambda function executes is actually written with a Cloud9 app. You must create a Cloud9 app that would work as the handler for the above Lambda function. The app must implement the following functionalities:

   - Gets notified when an object (any file regardless of .txt or .json) is created/uploaded to the S3 bucket. Stores the names of the bucket and uploaded file into a log file.
   - Reads the content of the object/file and parse the JSON/text content to retrieve each review record.
   - Computes the current value of the identifier field. Inserts the respective details of each record into the corresponding fields of the "ProductReview" table.

Note: You can test the Lambda function by just simply uploading text and/or json into the specified bucket.

Part B – Data processing pipeline

Complete the following tasks:

1. **Propose a data processing solution (aligns with ULO2):**
   Create a Glue DataBrew project to process NYC taxi trip data, download the dataset from this link https://www.kaggle.com/datasets/anandaramg/taxi-trip-data-nyc. Upload about 5000 randomly selected samples to your Glue DataBrew project. You then need to apply the appropriate pre-processing tools on the uploaded dataset. You must not replace the original data with the new results, instead augment the existing data with new columns.
   a. Replace all missing values in three appropriate data columns which have missing values. You can get idea which columns have missing about missing instances from this above link.
   b. Identify the duplicate rows and flag them.
   c. Flag outliers for any two appropriate columns.
   d. Compute the total fare amount for each weekday.
   e. Compute the total number of trips for each weekday.
   f. Compute the total number of trips for each month.

2. **Package solution (aligns with ULO2):**
   Create your recipe to apply the above methods and publish your recipe.

Requirements

- You will use a single bucket for the whole assignment, named as “yourusernamea2bucket”. Your handler app must be named as “yourusernamea2app”.
- Handler app must be implemented using Java and AWS SDK 1.x. The Java program must be implemented with separate methods for each functionality, exception handling and loop(s). The program must have detailed and clear comments to understand how the code works.

Resources

Resources required to complete the assessment task

- Modules 2, 3, and 4 (for Part A), and Module 5 (for Part B) to complete this assignment
- JSON array and object can be parsed using the “org.json” package. Please check this link: https://stleary.github.io/JSON-java/index.html. You can use the latest maven dependency from here: https://mvnrepository.com/artifact/org.json/json.

Task Submission

1. Download the final handler app from Cloud9 workspace and submit the zipped app as "yourscuusernameA2App.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. The recipe of the data processing pipeline as JSON file must be submitted to the MySCU site submission link.

Note: All your development resources for Part A and B, including bucket, database table, Lambda function, handler app, data project, recipe, must be present in your AWS workspace (UA-provided) to be tested and marked. 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 is may not be used for Assessment 2

Generative artificial intelligence (GenAI) tools, such as ChatGPT, **must not be used** for this assessment task. **You are required to demonstrate that you have developed the unit's skills and knowledge without any external support.** If you use GenAI tools in your assessment task, it will result in an academic integrity breach against you, as described in the Student Academic and Non-Academic Misconduct Rules, Section 3.

Since you are mastering fundamental skills, you are permitted to work from the examples in the MySCU site or study guide, but you must acknowledge assistance from other textbooks, classmates, or online resources. In particular, you must not use online material or help from others, as this would prevent you from mastering these concepts.

This diagram will help you understand where you can get help:

Lecturer Tutors	GenAI e.g., ChatGPT	Online Forums
Relatives	Students outside unit	Hired coders

Classmates	Private Tutors	Other

Encouraged

Attribution Required (in the report)

Ask tutor

Not acceptable

Please note that if your marker has any suspicion that you had help with your work or is not your own 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 and database services to set up a serverless compute application (ULO1) – 5%	Executes a Lambda function with meticulous configuration, incorporating the requisite runtime environment, and seamlessly integrates an S3 trigger into the serverless architecture. Configures the database table with accurate attributes and data types.	Configures a Lambda function appropriately with necessary runtime environment, and integrates an S3 trigger effectively into the serverless architecture. Ensures basic configuration of the database table with correct attributes and data types, with minor naming issues.	Establishes a Lambda function with a functional runtime environment, and integrates an S3 trigger adequately into the serverless architecture. The configuration of the database table includes basic attributes and data types. Contains minor naming and configuration issues.	Sets up a Lambda function with basic configuration, incorporating a rudimentary runtime environment, and integrates an S3 trigger minimally into the serverless architecture. The configuration of the database table may lack precision in attributes and data types.	Fails to deploy Lambda function with proper configuration, struggles with S3 trigger integration, and lacks adequate database table configuration.
Implement a serverless compute application (ULO1, ULO3) – 40%	Implements program with exemplary proficiency incorporating accurate retrieval and storing of bucket and key information, format validation, parsing JSON and text data, executing DynamoDB operations for both reading and inserting data flawlessly. Showcases exceptional coding practices by employing robust exception handling, well-structured loops, modular methods, and	Implements program with high level of competence by effectively handling bucket and key retrieval with format validation, accurately parsing JSON and text data, performing DynamoDB operations for reading and inserting data proficiently. Demonstrates strong coding practices with reliable exception handling, structured loops, modular methods, and detailed comments.	Implements program with competency in handling bucket and key retrieval with format validation, parsing JSON and text data with minor inaccuracies, executing DynamoDB operations for reading and inserting data satisfactorily. Employs decent coding practices with functional exception handling, organized loops, modular methods, and sufficient comments.	Implements program with basic requirements, incorporating bucket and key retrieval and format validation, parses JSON and text data adequately with some errors, executes DynamoDB operations for reading and inserting data adequately with minor issues. Employs basic coding practices with minimal exception handling, simple loops, modular methods, and limited comments.	Attempts insufficient proficiency in handling file retrieval and format validation, parsing JSON and text data inaccurately, executing erroneous DynamoDB operations for reading and inserting data. Lacks effective coding practices with inadequate exception handling, poorly structured loops, unorganized methods, and insufficient comments.

	comprehensive comments.				
Develop a serverless compute app using cloud storage and database services (ULO1, ULO3) – 35%	Develops a Lambda app that executes flawlessly without compile or runtime errors, accurately logs bucket and key, retrieves the JSON and text content, inserts the data accurately into DynamoDB table.	Develops a Lambda app that runs smoothly with no compile or runtime errors, reliably logs bucket and key, retrieves the JSON and text content, inserts the data into DynamoDB table, with some minor issues in presenting information.	Constructs a Lambda n app that runs without compile or runtime errors but offers limited functionality, with one key feature not operating correctly, indicating a foundational level of performance yet room for improvement.	Builds a Lambda app free from compile or runtime errors, yet it exhibits restricted functionality due to several key features malfunctioning, suggesting a basic operational level with significant scope for enhancement.	Attempts to develop a Lambda app but encounters significant compile or runtime errors, failing to meet the project's core requirements.
Analyse a data processing problem and develop a cloud solution (ULO2) – 20%	Implements a DataBrew recipe with exceptional proficiency that meticulously replaces missing values, flags duplicate rows, and identifies outliers. It excels in computing total fare amounts and trip numbers for weekdays and months showcasing advanced analytical capabilities.	Implements a DataBrew recipe that demonstrates a high level of proficiency by effectively replacing missing values, flagging duplicate rows, and identifying outliers. It proficiently computes total fare amounts and trip numbers for weekdays and months, reflecting advanced analytical skills, with ignorable issues.	Implements a DataBrew recipe that adequately handles data quality concerns by replacing missing values, flagging duplicate rows, and identifying outliers with satisfactory proficiency. It adequately computes total fare amounts and trip numbers for weekdays and months, with minor inaccuracies.	Implements a DataBrew recipe addressing basic data quality challenges by replacing missing values, flagging duplicate rows, and identifying outliers to a minimal extent. It calculates total fare amounts and trip numbers for weekdays and months with basic proficiency, with multiple missing functionalities.	Attempts a DataBrew recipe but fails to effectively address data quality issues, lacking proficiency in replacing missing values, flagging duplicate rows, and identifying outliers. It inadequately computes total fare amounts and trip numbers for weekdays and months, suggesting a fundamental misunderstanding.

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.