CSCI 5410
Serverless Data Processing

Sprint 1 Report

Submitted to

Dr. Saurabh Dey

Department of Computer Science

Dalhousie University.

Submitted by Team 4

Alishan Ali

Harsh Vaghasiya

Rutvik Mansukhbhai Vaghani

Sivasubramanian Venkatasubramanian

Table of Contents

User Management & Authentication Module

The user management and authentication module are an essential and starting point for any application, based on the given requirements for 1st factor authentication we will be using AWS Cognito [1]. It requires to have a user identity pool which then can be accessed through any frontend application using the Cognito SDK [2]. The SDK is also responsible to provide basic user related CRUD operations.

Moreover, for Question/Answer based authentication, we are required to use both AWS Lambda functions [3] and Dynamo DB [4]. We will be using Python programming language for the lambda function which will validate the questions/answers, and for each user we are going to store the security answers in the dynamo db.

Lastly, for mathematical based 3rd factor authentication, we are going to use AWS lambda functions as per the requirements. The AWS lambda functions give us the flexibility to only execute on demand, just like any REST API call.

The flow for this module will first validate the user identity using the AWS Identity pool, then it will validate the username and password stored in the DynamoDB. Similarly, for security questions and answers it will to the same. Using the AWS lambda functions, any random math question will be asked for user to complete 3rd factor authentication.

In Conclusion, this module will be implemented mainly using the AWS lambda functions that will be interacting with DynamoDB's Client APIs. Overall, using Cognito, Lambda functions and DynamoDB will provide a secure authentication mechanism to the overall serverless application.

Virtual Assistant Module

The virtual assistant will serve as a primary interaction tool for guests, registered users, and QDP agents, allowing users to navigate the platform, process data, and obtain results from previous sessions.

Technology Stack and Services:

- **Google Cloud Dialogflow**: For building conversational interfaces that allow the virtual assistant to respond to user queries [5].
- **Google Cloud Functions**: To handle backend processes triggered by Dialogflow interactions, such as navigating the site, retrieving past processed data, or escalating issues to QDP agents [6].
- **Firestore**: Chosen as the unified data store for user interactions, past processing results, and messages. Firestore allows fast read/write operations, making it ideal for both real-time chats and long-term storage [7].

Flow for Virtual Assistant Module:

1. **User Query**: A user (guest or registered) asks a question or submits a query using the virtual assistant.
2. **Dialogflow Interaction**: The query is passed to Google Cloud Dialogflow, which processes the natural language input.
3. **Cloud Function Trigger**: Dialogflow triggers a Google Cloud Function to fetch necessary data or execute an action (e.g., returning processing results or submitting a customer concern) [6].
4. **Firestore Integration**: If the query involves fetching results from previous data processing sessions, the Cloud Function will retrieve the data from Firestore and send it back to the user [7].
5. **Response to User**: Dialogflow provides the processed response back to the user in a conversational format [4].

Message Passing Module

This module ensures asynchronous communication between customers and QDP agents, handling customer concerns or inquiries based on their processing reference code.

Technology Stack and Services:

- **Google Cloud Pub/Sub**: For publishing customer messages and concerns to the appropriate QDP agents [8].
- **Google Cloud Functions**: For subscribing to messages from Pub/Sub and routing them to the appropriate agent [6].
- **Firestore**: As the central data store, Firestore will store all message exchanges, ensuring consistency across the Virtual Assistant and Message Passing modules [7].

Flow for Message Passing Module:

1. **Customer Submission**: A customer submits a concern or question related to their data processing reference code through the virtual assistant or another interface [5].
2. **Pub/Sub Trigger**: The concern is published to the Google Cloud Pub/Subtopic [8].
3. **Cloud Function Execution**: A subscribed Cloud Function is triggered, routing the message to the appropriate QDP agent. If no agent is available, the concern is queued [6].
4. **Message Logging**: All messages and responses are logged in Firestore for future reference [7].
5. **Agent Response**: Once an agent responds, the message is logged in Firestore, and the response is sent back to the customer through the virtual assistant [5].

Technology Choice Justification: Firestore and Google Cloud Functions

- **Firestore** was chosen because it provides a flexible and scalable NoSQL database that works well for both the Virtual Assistant and Message Passing modules. Since both modules interact with chat-related data, consolidating everything into Firestore ensures consistency and eliminates data silos. It supports real-time synchronization, making it ideal for chat and messaging [4].
- **Google Cloud Functions** is used instead of AWS Lambda due to its seamless integration with other GCP services like Dialogflow, Pub/Sub, and Firestore. By keeping everything within the Google ecosystem, the architecture remains consistent and efficient, reducing overhead and integration complexity [6].

Overall Architecture Flow:

1. **User Query or Concern**: Users interact with the virtual assistant (Dialogflow), submitting queries or concerns [5].
2. **Backend Processing**: Dialogflow triggers a Cloud Function to fetch or process data, interacting with Firestore as needed [6].
3. **Pub/Sub Messaging**: For concerns requiring agent intervention, the Message Passing Module uses Pub/Sub to forward the query to a QDP agent [8].
4. **Data Storage**: Firestore logs all interactions, including chat data, concerns, and agent responses [7].
5. **Response to User**: The virtual assistant responds to the user with data or notifies them about the status of their concern [5]

Notifications Module

This module ensures that registered users are notified via email about their activities, such as login, registration, and data processing results.

Technology Stack and Services:

- **AWS SQS**: This service is used to send, store, and receive messages between different modules at any volume and the main part is that no messages would be lost [11].
- **AWS SNS**: This service provides message delivery functionality between two entities publisher and subscribers. Publishers can communicate asynchronously with subscribers, i.e. subscribers can subscribe to a topic, and when a publisher sends something to a topic the subscribers would receive the content from the topic [12].

How the Notifications Module Works:

1. **User Registration and Login**:
    a) When a new user gets registered, A notification regarding the activity is directly sent to email used for registration using AWS SNS.
    b) Similarly, when the user logs into the application using AWS SNS, a notification is sent to their respective emails.

2. **Data Processing and Approval Request:**
    a) In case of 5 to 10 requests for processing a file, there won't be any issues. But in the case of millions of requests, it would be difficult for lambda functions to execute it efficiently, and if we also increase the specs of the services, it would be a waste of resources.
    b) To solve the above problem AWS SQS would be used to act as a buffer for the millions of requests and the lambda function would take requests from the queue as it finishes.
    c) When a user uploads their document, the message is published in a sns topic and is fanned out to multiple queues, whose subscribers would be different data processing services.
    d) Once all the different services of data processing request get executed a notification is sent to the registered user stating that data processing is complete it could be either success or failure.

Data Processing Module

This module requires the following three different types of data processing:

1. JSON to CSV file conversion using AWS Glue [13].
2. Extraction of the Named entities using the AWS Lambda function.
3. Word cloud generation using GCP Locker Studio [9].

Based on the user role, registered users can have access to all three of these data processing while the guest user would have option to only use Data Processing type 1 (JSON to CSV). The restriction can be handled by validating the user role then, we can hide the data processing option on the frontend layer.

For the file upload mechanism, separate S3 buckets can be created to store files for different data processing types. Based on this, relevant Lambda functions will be triggered to execute the job processing.

For type 1 processing, an event would trigger after validating the uploaded file type. This would run the AWS glue job and after successful completion it will store the result in a new outputs folder.

For type 2 processing, after the successful upload to txt file, a lambda function will read all the words, and it will maintain the count for all of them. Then, we can remove the help words or identify named words using Python library like SpaCy, then this result will be stored in an output folder for GCP to perform further processing [14].

Lastly, using the outputs of type 2 processing, the GCP locker studio will read the text file. It will do word analysis to generate the word cloud visualization.

Data Analysis Module

The Data Analysis module is designed to be useful for all types of users, including **QDP Agents**, **Guests**, and **Customers**. Its primary function is to display user statistics and analyze customer feedback, helping agents understand user engagement and overall satisfaction.

Technology Stack and Services:

- **LookerStudio**: This tool is used to create visual dashboards, allowing QDP Agents to see data such as the total number of users and login statistics on the admin page [9].
- **Google Natural Language API**: This service is responsible for analyzing customer feedback automatically, determining whether the feedback is positive, negative, or neutral [10].
- **DynamoDB/S3/Firestore/GCP Storage**: One of these storage systems will be used to store and retrieve customer feedback, offering flexibility depending on your choice of infrastructure [4].

How the Data Analysis Module Works:

3. **QDP Agent Insights**:
    a. **QDP Agents** can log into the admin page and access user statistics through a dashboard powered by **LookerStudio**, which might be embedded directly into the page [9].

4. **Viewing Customer Feedback**:
    a. **Guests, Customers, and QDP Agents** can view **customer feedback** about their experiences with data processing in an easy-to-read table on the frontend [5].
    b. The feedback is retrieved from a flexible data source such as **DynamoDB**, **S3**, **Firestore**, or **GCP Storage**, ensuring scalability [4].

5. **Automatic Sentiment Analysis**:
    a. Once the feedback is submitted, the **Google Natural Language API** runs in the backend to automatically analyze the content.
    b. The feedback is classified based on sentiment (e.g., positive, negative, or neutral), making it easier for users to gauge how customers feel about their experience [10].
6. **Presenting Data to Users**:
    a. All users (QDP Agents, Guests, and Customers) can see the sentiment analysis results on the frontend, giving a clear picture of customer satisfaction [9].

Why These Technologies Were Chosen:

- **LookerStudio** was selected for its ability to create detailed and user-friendly dashboards, allowing QDP Agents to view important statistics about users and their activity [9].
- **Google Natural Language API** is ideal for automating the sentiment analysis process, saving time by removing the need for manual feedback reviews. It's a powerful tool for understanding the emotional tone of feedback [10].
- **DynamoDB/S3/Firestore/GCP Storage** provides various options for data storage, ensuring that the module can be adapted to different systems and easily scale to handle larger amounts of data as needed [4].

How the Module Works End-to-End:

1. **User Access**:
    a. **QDP Agents** can view key statistics through the admin dashboard.
    b. **Guests, Customers, and QDP Agents** can all view customer feedback on the frontend, fetched from one of the supported storage systems.

2. **Backend Processing**:
    a. The **Google Natural Language API** analyzes the feedback, assessing the sentiment behind each comment.
3. **Displaying the Results**:
    a. The analyzed feedback is shown in a user-friendly way on the frontend, helping all users understand how customers feel about their data processing experience.

Web Application Building and Deployment:

This module entirely focuses on deploying the frontend of the application.

Technology Stack and Services:

- **React**: The frontend will be built using React for its component-based architecture, enabling efficient development and maintainability [14].
- **GCP CloudRun**: This service would be used to deploy the containerized frontend application [15].
- **Terraform**: Terraform is an ideal choice for provisioning multi-cloud services, as it supports both AWS and GCP, making it the best infrastructure-as-code (IaC) tool for efficiently managing the infrastructure of this application across multiple cloud providers [16].

Individual Contribution:

Table 1: Shows Individual Contribution on Sprint 1 research and report.

Member	Contribution
Alishan Ali	• Module 1 (User Authentication) • Module 5 (Data Processing)
Rutvik Vaghani	• Module 6 (Data Analysis) • Background Research About Programming Language.
Harsh Vaghasiya	• Module 2 (Virtual Assistant) • Module 3 (Message Passing)
Sivasubramanian Venkatasubramanian	• Module 4 (Notifications) • Module 7 (Web Application Building and Deployment)

Meeting Logs:

Table 2: Shows Team 4 meeting logs for sprint 1.

Meeting Date	Meeting Link	Attendees
24/9/2024	https://dalu-my.sharepoint.com/:v:/g/personal/hr971794_dal_ca/EXcoJYb2XkdHraUCwHgeGGABbeKTzvakcflF-xw0lLeA6A?referrer=Teams.TEAMS-ELECTRON&referrerScenario=MeetingChicletGetLink.view	Everyone from the team joined
29/9/2024	https://dalu-my.sharepoint.com/personal/al459703_dal_ca/_layouts/15/stream.aspx?id=%2Fpersonal%2Fal459703%5Fdal%5Fca%2FDocuments%2FRecordings%2FSprint%201%20%2D%20Meeting%202%2D20240929%5F233543%2DMeeting%20Recording%2Emp4&referrer=StreamWebApp%2EWeb&referrerScenario=AddressBarCopied%2Eview%2E076323b2%2Df3ba%2D4d46%2D957c%2D17de4decc443	Everyone from the team joined
2/10/2024	https://dalu-my.sharepoint.com/:v:/g/personal/hr971794_dal_ca/EcZU9ANbdrpKsz9a8ugnN1EBpCzML_5ZoICwHUUsoMe1ww?referrer=Teams.TEAMS-ELECTRON&referrerScenario=MeetingChicletGetLink.view	Everyone from the team joined

References:

1. AWS, "Getting started with user pools," Amazon Web Services, 2023. [Online]. Available: https://docs.aws.amazon.com/cognito/latest/developerguide/getting-started-user-pools.html. [Accessed: 02-Oct-2024].
2. AWS, "Class: AWS.CognitoIdentityServiceProvider," Amazon Web Services, 2023. [Online]. Available: https://docs.aws.amazon.com/AWSJavaScriptSDK/latest/AWS/CognitoIdentityServiceProvider.html. [Accessed: 02-Oct-2024].
3. AWS, "Tutorial: Using AWS Lambda with Amazon API Gateway," Amazon Web Services, 2023. [Online]. Available: https://docs.aws.amazon.com/lambda/latest/dg/services-apigateway-tutorial.html. [Accessed: 02-Oct-2024].
4. "DynamoDB Documentation," AWS [Online]. Available: https://docs.aws.amazon.com/amazondynamodb/latest/developerguide. [Accessed: Sep 29, 2024].
5. "Dialogflow Documentation," Google Cloud [Online]. Available: https://cloud.google.com/dialogflow/docs. [Accessed: Sep 29, 2024].
6. "Cloud Functions Documentation," Google Cloud [Online]. Available: https://cloud.google.com/functions/docs. [Accessed: Sep 29, 2024].
7. "Firestore Documentation," Google Cloud [Online]. Available: https://cloud.google.com/firestore/docs. [Accessed: Sep 29, 2024].
8. "Pub/Sub Documentation," Google Cloud [Online]. Available: https://cloud.google.com/pubsub/docs. [Accessed: Sep 29, 2024].
9. "LookerStudio Documentation," Google Cloud [Online]. Available: https://cloud.google.com/looker/docs. [Accessed: Sep 29, 2024].
10. "Natural Language API Documentation," Google Cloud [Online]. Available: https://cloud.google.com/natural-language/docs. [Accessed: Sep 29, 2024].
11. "Amazon Simple Queue Service," AWS [Online]. Available: https://aws.amazon.com/sqs. [Accessed: Oct 2, 2024].
12. "What is Amazon SNS?," AWS Docs [Online]. Available: https://docs.aws.amazon.com/sns/latest/dg/welcome.html. [Accessed: Oct 2, 2024].
13. AWS, "Defining Tables in the AWS Glue Data Catalog," Amazon Web Services, 2023. [Online]. Available: https://docs.aws.amazon.com/glue/latest/dg/start-data-catalog.html. [Accessed: 02-Oct-2024].
14. React, "The library for web and native user interfaces," React, 2023. [Online]. Available: https://react.dev/. [Accessed: 02-Oct-2024].
15. GCP, "Build apps or websites quickly on a fully managed platform," Google Cloud, 2023. [Online]. Available: https://cloud.google.com/run/. [Accessed: 02-Oct-2024].

16. Terraform, " Infrastructure as code," Amazon Web Services, 2023. [Online]. Available: https://www.terraform.io/use-cases/infrastructure-as-code. [Accessed: 02-Oct-2024].