DALHOUSIE
UNIVERSITY

CSCI 5410
SERVERLESS DATA PROCESSING

“Quick Data Processor”

Sprint 1 Report

SDP#Team8

Submitted By:

- Rushil Borad [rs519505@dal.ca]
- Mansi Kalathiya [mn410013@dal.ca]
- Bhavya Dave [bh392017@dal.ca]
- Avadh Rakholiya [AV786964@DAL.ca]

GitLab Repository: https://git.cs.dal.ca/rakholiya/serverless_project

Table of Contents

Table of Figures

Background Research

Backend-as-a-Service (BaaS)

- Backend-as-a-Service (BaaS) platforms allow developers to integrate backend services like data storage, user authentication, and server-side logic into their web and mobile applications without managing infrastructure. This enables them to focus on frontend and core application development. Examples include AWS Lambda, Google Cloud Functions, and Azure Functions, which allow for serverless execution of backend code without provisioning servers.

Amazon API Gateway

- It provides a fully managed service to handle API call processing, including traffic management, authorization, and monitoring. This supports efficient API development with scalable performance and flexible security.

AWS Lambda

- It offers serverless compute service, executing backend logic in response to events like DynamoDB updates or S3 triggers. It supports multiple languages such as Node.js, Python, and Java, and includes extensive documentation for setup and management.

Amazon DynamoDB

- It is a NoSQL database that supports both document and key-value models with automatic scaling for storage and throughput. It integrates seamlessly with AWS Lambda, allowing direct function triggering on table updates.

Google Cloud Functions

- It is a serverless execution environment on GCP, allowing developers to deploy single-purpose functions triggered by cloud events. It integrates with GCP services like Cloud Storage and Firestore, supporting event-driven workflows. Google Cloud Functions can also interact with AWS services, enabling multi-cloud applications.

AWS Cognito

- It simplifies user authentication and access control for web and mobile applications. It includes features like multi-factor authentication and advanced security options. Pricing varies based on the number of monthly active users and security features used.

AWS SNS (Simple Notification Service)

- It provides messaging services using a publish-subscribe model, allowing applications and devices to send and receive notifications. It supports various notification formats and includes a free tier of 1 million requests per month.

AWS SQS (Simple Queue Service)

- It is a message queuing service that enables the decoupling and scaling of microservices and serverless applications. It supports both standard and FIFO queues, with a free tier of 1 million requests per month.

Introduction to Project

Project Overview:

- Briefly describe the “Quick Data Processor” application and its goal to provide customized data processing services using a serverless multi-cloud deployment model.

Key Features:

- Virtual assistance for query resolution.
- Message passing functionality between agents and users.

User Types:

- Guests: Limited data processing without registration.
- Registered Customers: Full access with multi-factor authentication.
- QDP Agents: Administrative capabilities.

Use Case Analysis

Guest Users:

- Process up to two files without registration.
- Access to basic navigation assistance.
- View overall feedback polarity.

Registered Customers:

- Process unlimited files with multi-factor authentication.
- Access virtual assistant and previous processing results.
- Provide feedback on data processing options.

QDP Agents:

- Manage registered users and services.
- Communicate with customers regarding data processing concerns.
- Use multi-factor authentication for secure access.

Module Overview

User Management & Authentication:

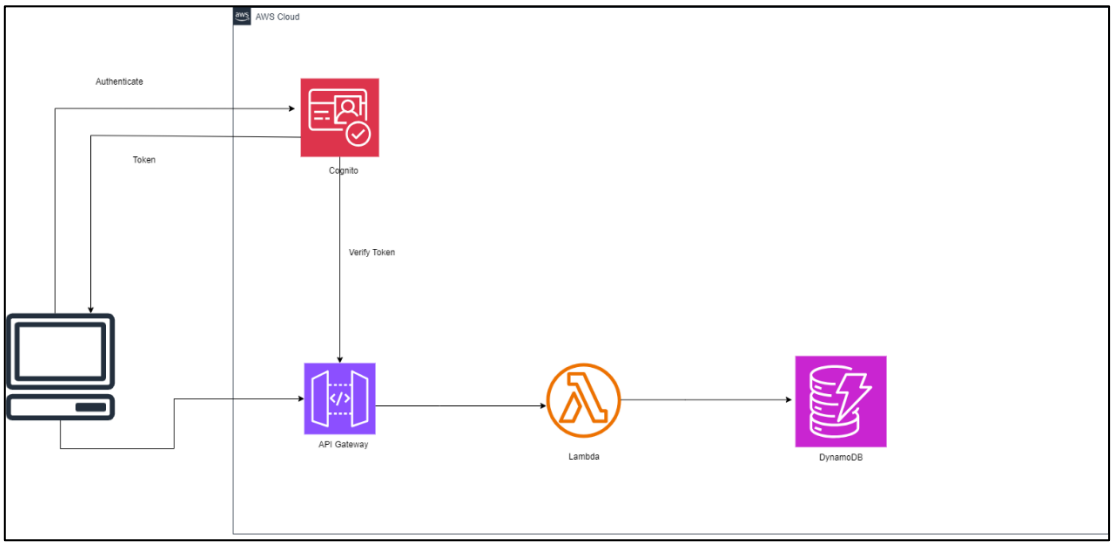

Figure 1: AWS Cognito implementation architecture

- We investigated the use of **AWS Cognito** and **AWS Lambda** to develop a secure user management and authentication system for the QuickDataProcessor project. We also explored how AWS Cognito facilitates secure sign-up and sign-in processes with multi-factor authentication (MFA) using user ID, password, and question-answer security mechanisms, ensuring a secure authentication flow for all users.
- We also examined how **DynamoDB** can be used for storing user details securely and efficiently, ensuring that all user credentials and security questions are properly encrypted and managed with high availability. This allows the system to handle high traffic and ensure data consistency across different regions.
- Error handling and session management were key focus areas in our research, where **AWS Lambda** was used for handling the third factor of authentication (math-based verification) and validating user session tokens for improved security.
- Finally, our research highlighted the importance of **IAM roles** and **Cognito Identity Pools** for securing access to resources during the authentication process, ensuring that only authorized users can access specific services in the application

Virtual Assistant:

- We will implement the chatbot using **GCP Dialogflow**, training it to handle user queries like "How to register?" and "Where are my files?" By defining intents and entities, the assistant will understand natural language and respond accurately. **Webhooks** will be used to trigger **Cloud Functions** for dynamic data retrieval and backend processing.
- Google **Firestore** will store processed file data and user concerns. We will implement a collection structure that links reference codes to file results, enabling the assistant to fetch results dynamically. Firestore's real-time database capabilities will ensure prompt access to user data and efficient logging of customer concerns.
- We will use GCP Cloud Functions to handle complex backend logic, such as querying Firestore for file details and forwarding customer concerns to **QDP agents**. Cloud Functions allow for serverless execution, ensuring scalability and cost-effectiveness by only running code when triggered by **Dialogflow** or **Firestore** events.

Message Passing:

We will use GCP Cloud Functions to handle complex backend logic, such as querying Firestore for file details and forwarding customer concerns to QDP agents. Cloud Functions allow for serverless execution, ensuring scalability and cost-effectiveness by only running code when triggered by Dialogflow or Firestore events.

- We will implement **GCP Pub/Sub** to enable asynchronous communication between different components. By publishing messages to topics and subscribing to them in real-time, this approach will ensure reliable and scalable message delivery across the system, facilitating smooth communication without direct coupling between services.
- We will log message exchanges using either Firestore or DynamoDB to maintain a detailed record of communication between services. This ensures traceability and auditing of messages, with structured data storage supporting real-time updates and scalable logging based on project requirements.
- **Cloud Functions** will be triggered by **Pub/Sub messages** to handle business logic and process the communication flow. This serverless approach provides a cost-effective, scalable solution that only executes code in response to new messages, enhancing system efficiency and responsiveness.

Notifications:

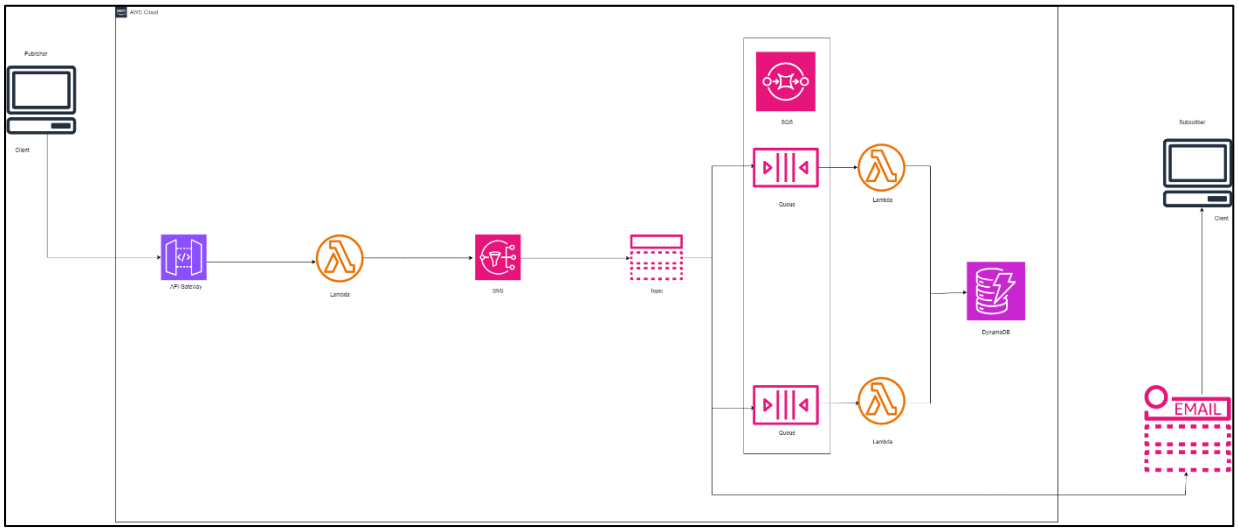

Figure 2 : AWS SNS with SQS for Notification

We will use **AWS Simple Notification Service (SNS)** and **Simple Queue Service (SQS)** to handle notifications and message queuing. Notifications will be sent to users when certain events occur, such as successful registration, successful login, or when their file processing is completed.

Service Components:

- **AWS SNS:** This service will send push notifications and emails.
    - For example, after a user successfully registers or logs in, SNS will automatically trigger an email or app notification confirming the action. Additionally, users will be notified when their files are successfully processed or if an error occurs.
- **AWS SQS:** For communication between backend services, SQS will queue requests such as data processing approvals or feedback submissions. These requests will then be picked up by AWS Lambda for further processing.

When a registered user completes the sign-up process, an SNS notification is sent to their email to confirm their **successful registration**. Similarly, once a file has been processed (e.g., a JSON file converted to CSV), an SNS notification is sent to the user, informing them that their file is ready for download. For internal processes, such as sending feedback to agents, SQS will ensure that the feedback request is queued and processed asynchronously, ensuring system efficiency.

Data Processing:

- Our research focused on integrating **AWS Glue** and **AWS Lambda** to enhance the **Data Processing Module**. We have explored how AWS Glue can be used to convert data files (such as JSON to CSV) automatically which makes it a highly efficient and scalable tool for processing user-uploaded files in real-time. This helped us understand the capabilities of serverless data transformation in a cloud-based environment.
- We also researched using **AWS Lambda** for processing text files to extract Named Entities and to display results such as word clouds. Lambda's event-driven architecture allows for quick processing of uploaded data files without the need for manual intervention, improving system efficiency.
- Lastly, we evaluated the use of **GCP Looker Studio** for real-time data visualization, enabling users to view processed data insights, such as word clouds, directly within the application. This combination of AWS and GCP services ensures that the data processing workflow is automated, secure, and user-friendly

Data Analysis:

- We will be using **Google Looker Studio** and **Google Cloud Natural Language API** to develop the **Data Analysis Module.**
- We saw how **Looker Studio** can be integrated with the frontend to provide real-time visualization of data processing results, such as user feedback and login statistics, offering intuitive insights for QDP agents and users.
- We also saw on how to use the **Google Cloud Natural Language API** to automatically analyze customer feedback and determine sentiment, which would then be displayed in a clear, tabular format within the application. This ensures that both guests and registered users can view feedback and gain insights into processing performance.

Web Application Building and Deployment

- We will be using **React** and **GCP CloudRun** to implement the **Web Application Building and Deployment Module**. We saw how **React** can be used to build a lightweight and responsive frontend for communication with backend services, allowing users to easily move through the application and perform tasks like file uploads, viewing results, and receiving notifications
- For deployment, we researched **GCP CloudRun**, which offers a fully managed serverless platform for deploying the web application. This ensures that the frontend can scale dynamically based on user demand without requiring manual server management, making it cost-effective and highly scalable.

Project Planning & Management

GitLab Issue Board:

We have created a GitLab repository for the project and set up an issue board to maintain the progress of the project. We assigned three labels to the issues. Currently, all the issues are in the assigned state as no work has started yet. Each issue has its description as shown in Figure 4 below.

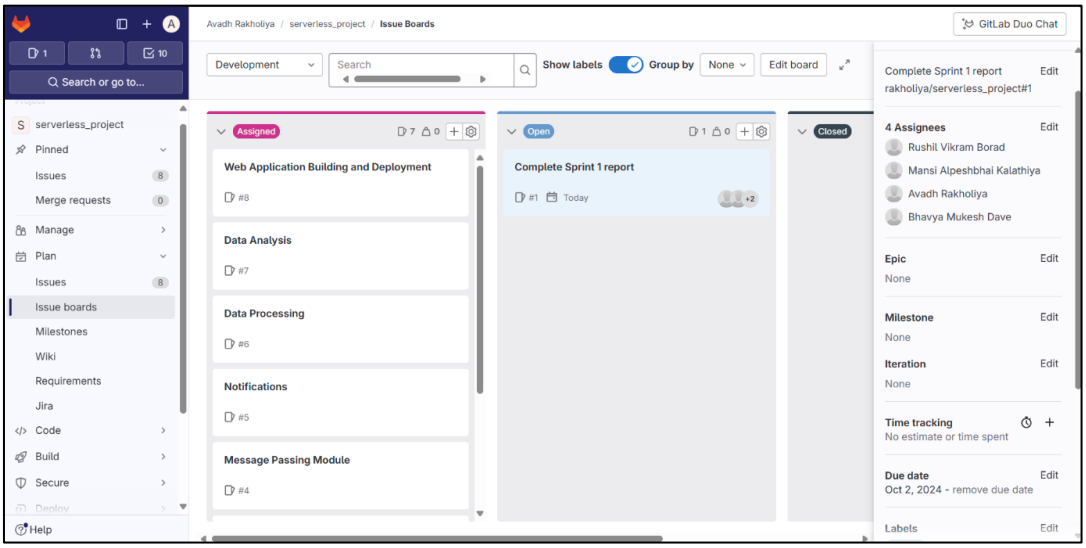

Figure 3 : GitLab Issue Board created

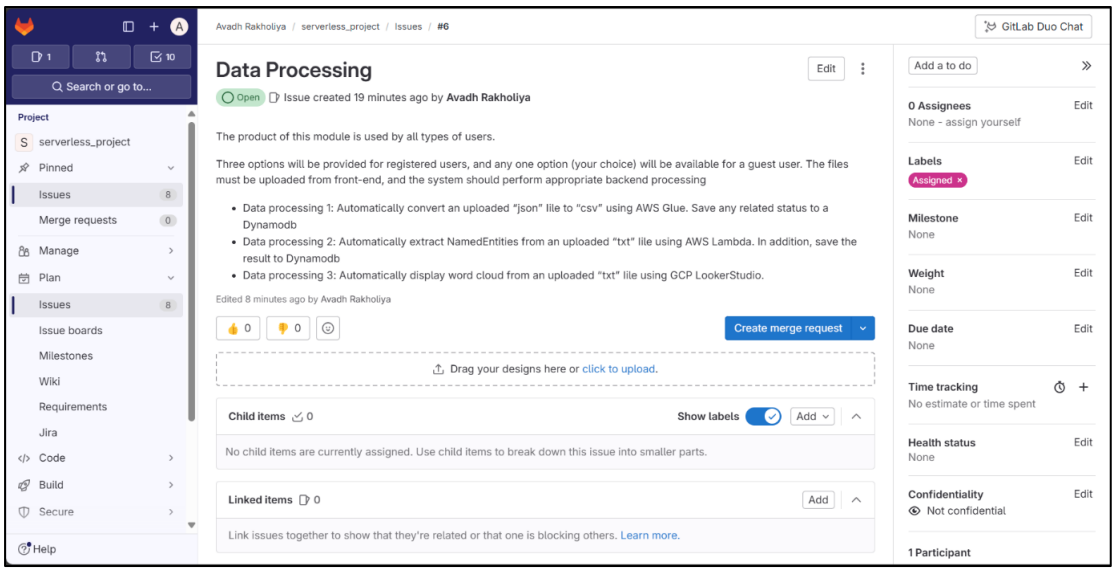

Figure 4 : GitLab Issue Board

Gantt Chart:

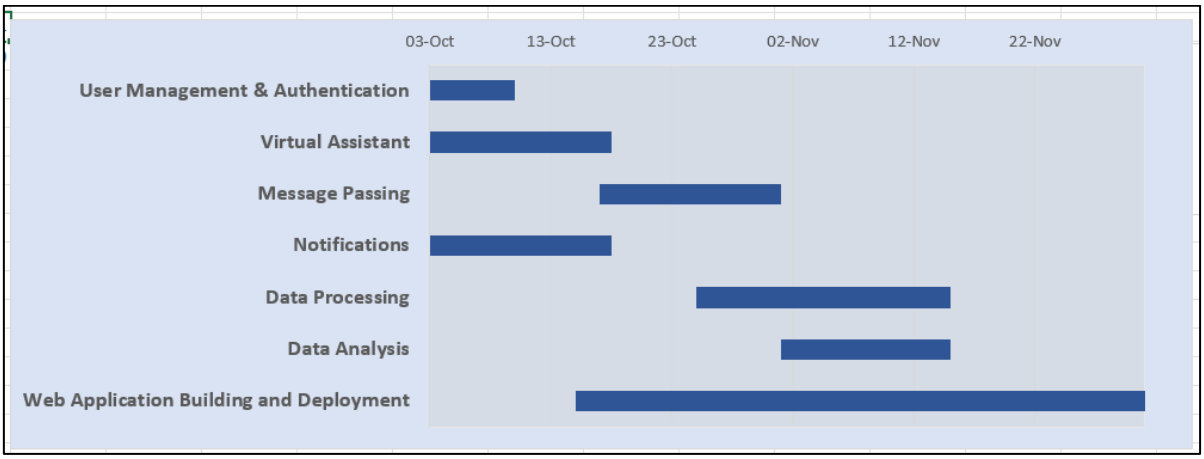

Figure 5 : Project Gantt chart tentative planning

Meeting Logs

- **Date: September 23, 2024**

Topic of Discussion: Project Kick-off and Requirement Overview

Overview (MOMs):

- We met physically on campus and discussed the overall project goals and requirements.
- We reviewed the **Project Requirement Document**, which outlined the modules to be developed, the expected timelines, and deliverables.
- Identified the core components of the project: **User Management**, **Virtual Assistant**, **Message Passing**, **Notifications**, **Data Processing**, **Data Analysis**, and **Web Application**.
- Discussed about the initial research phase on each module and assigned initial responsibilities to team members.

Team Members: Avadh, Bhavya, Mansi, Rushil.
Meeting Link: N/A (Physical Meeting)

- **Date: September 26, 2024**

Topic of Discussion: Service Selection and Architecture Finalization

Overview (MOMs):

- This meeting was held online, where we discussed the detailed design of each module.
- The team finalized the cloud services and frameworks to be used:
  - **User Management**: AWS Cognito, AWS Lambda, DynamoDB
  - **Virtual Assistant**: GCP DialogFlow, Firestore, Cloud Functions
  - **Notifications**: AWS SNS, AWS SQS
  - **Data Processing**: AWS Glue, AWS Lambda, GCP Looker Studio
  - **Frontend Framework**: **React**
- We also discussed the integration points between AWS and GCP to ensure a smooth flow in the system architecture.
- Agreed on next steps for Sprint 1: Begin implementation of the **User Management** and **Virtual Assistant** modules.

Team Members: Avadh, Bhavya, Mansi, Rushil.
Meeting Link: link

References

[1] AWS, “AWS Lambda – Serverless Compute,” Amazon Web Services, Inc., 2019. [Online]. Available: https://aws.amazon.com/lambda/. [Accessed: 02-Oct-2024].

[2] AWS, “Amazon API Gateway,” Amazon Web Services, Inc., 2019. [Online]. Available: https://aws.amazon.com/api-gateway/. [Accessed: 02-Oct-2024].

[3] AWS, “Amazon DynamoDB - Overview,” Amazon Web Services, Inc. [Online]. Available: https://aws.amazon.com/dynamodb/. [Accessed: 02-Oct-2024].

[4] AWS, “Amazon Simple Notification Service (SNS) | AWS,” Amazon Web Services, Inc., 2019. [Online]. Available: https://aws.amazon.com/sns/. [Accessed: 02-Oct-2024].

[5] AWS, “Amazon Simple Queue Service (SQS) | Message Queuing for Messaging Applications | AWS,” Amazon Web Services, Inc., 2018. [Online]. Available: https://aws.amazon.com/sqs/. [Accessed: 02-Oct-2024].

[6] AWS, “Amazon Cognito - Simple and Secure User Sign Up & Sign In | Amazon Web Services (AWS),” Amazon Web Services, Inc., 2024. [Online]. Available: https://aws.amazon.com/cognito/. [Accessed: 02-Oct-2024].

[7] Google Cloud, “Cloud Functions,” Google Cloud. [Online]. Available: https://cloud.google.com/functions. [Accessed: 02-Oct-2024].

[8] Looker Studio, “Looker Studio Overview,” lookerstudio.google.com. [Online]. Available: https://lookerstudio.google.com/u/0/navigation/reporting. [Accessed: 02-Oct-2024].

[9] AWS, “AWS Glue - Managed ETL Service - Amazon Web Services,” Amazon Web Services, Inc. [Online]. Available: https://aws.amazon.com/glue/. [Accessed: 02-Oct-2024].

[10] Google Cloud, “Google Cloud Deployment Manager documentation | Cloud Deployment Manager Documentation,” Google Cloud. [Online]. Available: https://cloud.google.com/deployment-manager/docs. [Accessed: 02-Oct-2024].