DALHOUSIE UNIVERSITY

Sprint 1 Team Report

Team 36

Christin Saji - B00977669
Jay Jagdishbhai Patel - B00981520
Jeet Jani - B00972192
Kuldeep Rajeshbhai Gajera – B00962793

GitLab Repo Link: https://git.cs.dal.ca/patel45/serverless_group_36

Table of Contents

Architecture-Oriented Research

Details of Research

Research Area	Source	URL
Serverless application best practices	AWS Blog Home - Best practices for organizing larger serverless applications	https://aws.amazon.com/blogs/compute/best-practices-for-organizing-larger-serverless-applications/
AWS Serverless Architectural Patterns and Best Practices	Medium	https://medium.com/aws-serverless-microservices-with-patterns-best/aws-serverless-architectural-patterns-and-best-practices-d2d446375924
Overview of Serverless Architecture	Research Gate	https://www.researchgate.net/publication/339680821_Overview_Of_Serverless_Architecture_Research
A Review on Serverless Architecture of Web Application	International joint conference on computing sciences	https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4495958
Serverless Architecture patterns and best practices	Free Code Camp	https://www.freecodecamp.org/news/serverless-architecture-patterns-and-best-practices/

Learning Outcomes of Architecture Oriented Research

Our journey through serverless architecture has led to very important lessons in its transformational possibilities in cloud computing. We review some of the research papers and industry articles to gain an in-depth understanding of the key principles, benefits, patterns, and best practices of serverless computing. This report summarises our learning and its implications against modern application development [3], [5].

Overview of Serverless Architecture

Serverless is a combination of back-end as a service (BaaS) and function as a service (FaaS). The main representation form of serverless is FaaS; hence, serverless computing is considered a "function as a service (FaaS)" or a "function-driven event" [2].

It is based on the developer's code for the correct resource allocation, and resources on the platform are started when a predefined event is triggered [2].

Below are some Advantages of using Serverless Architecture:

API Gateway & Lambda Integration: Utilize API Gateway for routing and security, while Lambda functions handle application-specific logic. Benefits include rapid deployment, on-demand scalability, and pay-per-use pricing [2].

RESTful Microservice Pattern: Implement synchronous communication for backend services using AWS Lambda with REST API, CRUD endpoints, and data persistence in AWS DynamoDB [1].

Fan-Out Pattern: Manage massive workloads (e.g., image processing) by triggering multiple Lambda functions to process smaller task chunks simultaneously [3].

Publish/Subscribe Pattern: Enable one-to-many communication where a publisher sends messages to multiple subscribed services via a message broker [2].

Messaging Pattern: Facilitate asynchronous communication between serverless functions using message queues, allowing independent message processing by each function [4].

Topic-Queue Chaining & Load Balancing Pattern: Use a queue as a buffer to handle asynchronous invocations, preventing data loss and balancing the load among consumer processes [1].

Amazon SQS for Decoupling: Employ Amazon SQS to decouple microservices and process events asynchronously, enhancing system resilience and scalability [2], [5].

Serverless Architecture Patterns and Best Practices

API Gateway & Lambda Integration: Utilize API Gateway for routing and security, while Lambda functions handle application-specific logic. Benefits include rapid deployment, on-demand scalability, and pay-per-use pricing.

RESTful Microservice Pattern: Implement synchronous communication for backend services using AWS Lambda with REST API, CRUD endpoints, and data persistence in AWS DynamoDB.

Fan-Out Pattern: Manage massive workloads (e.g., image processing) by triggering multiple Lambda functions to process smaller task chunks simultaneously.

Publish/Subscribe Pattern: Enable one-to-many communication where a publisher sends messages to multiple subscribed services via a message broker.

Messaging Pattern: Facilitate asynchronous communication between serverless functions using message queues, allowing independent message processing by each function.

Topic-Queue Chaining & Load Balancing Pattern: Use a queue as a buffer to handle asynchronous invocations, preventing data loss and balancing the load among consumer processes.

Amazon SQS for Decoupling: Employ Amazon SQS to decouple microservices and process events asynchronously, enhancing system resilience and scalability.

Organizing Larger Serverless Applications

1. **Infrastructure as Code (IaC)**

   Use Infrastructure as Code tools to provision and manage serverless resources automatically with AWS CloudFormation and AWS CDK [1]. IaC is used to automate creation and management processes to ensure that consistency and repeatability in creating the resources are maintained [4].

2. **Event-Driven Architecture:**
   One other approach could be the event-driven architecture, where different services communicate with each other through events. AWS services, such as EventBridge and SNS, largely work on the implementation of these architectures, making services scalable and resilient [5].

Application Perspective Research

Details of Research

Research Area	Source	URL
Hotel Booking System built using Angular 4	GitHub	https://github.com/shwetajoshi601/aws-serverless-hotelbooking
Serverless Bed & Breakfast	GitHub	https://github.com/AnkushMudgal/serverless_bnb
Bookstore Demo App built on AWS	GitHub	https://github.com/aws-samples/aws-bookstore-demo-app
Running a low-cost hotel booking application with AWS Serverless services	Hieu's Technical Blog - A year of running a hotel booking application on AWS Serverless services for $0.8/month	https://hieudd.substack.com/p/a-year-of-running-a-hotel-booking
TripAdvisor on AWS	High Scalability - An Epic TripAdvisor Update: Why Not Run on the Cloud? The Grand Experiment	https://highscalability.com/an-epic-tripadvisor-update-why-not-run-on-the-cloud-the-gran/

Learning Outcomes of Application Perspective Research

Serverless Bed & Breakfast

Our team has conducted extensive research on the Serverless Bed & Breakfast (B&B), a cloud-based hotel reservation system designed to enhance customer experience through advanced cloud technologies [6], [7]. This system facilitates hotel accommodation reservations, food orders from the restaurant, and personalized trip package requests from tour operators [7]. The primary components of this system include Customers, Hotel Management, Kitchen, and Tour Operators [6].

Customers must register or log in to access the services, enabling them to book rooms, order food, and arrange tours [6]. Additionally, the system supports customer feedback, booking management, and navigation assistance [7]. The Hotel Management service handles room reservations and customer interactions with the kitchen [9]. The Kitchen service manages customer orders, billing, meal preparation (breakfast only), inventory access, and order scheduling [6]. Tour Operator services offer customized tour packages, sending information to customers via email [7].

Serverless B&B employs a multi-cloud architecture using Google Cloud Platform (GCP) and Amazon Web Services (AWS) to ensure high availability [6]. The React frontend's static assets are hosted in an AWS S3 bucket, facilitating online access [9]. User data and application information are stored in Cloud Firestore and Amazon DynamoDB, providing user registration and multi-factor authentication [6]. Amazon Cognito manages and authorizes users, supported by DynamoDB and Firestore databases [9].

The application's virtual help system, powered by Amazon Lex, utilizes AWS Lambda functions to process user requests using DynamoDB data [9]. AWS Lambda and Google Cloud Functions handle most of the application's logical processing, replacing traditional server-oriented design [6]. Google Cloud Pub/Sub supports communication and concurrent request processing [7]. These cloud services are securely accessed via API Gateway services from both GCP and AWS [9].

To create personalized tour packages and analyze customer feedback, the application leverages machine learning techniques from GCP, specifically Vertex AI and NLP API services [9]. Amazon CloudTrail and CloudWatch cloud services generate reports on user login statistics [9].

Our research highlights the seamless integration of these technologies to deliver a robust and efficient hotel reservation system, enhancing the overall customer experience [6].

Trip Advisor

Our team has researched TripAdvisor, a cloud-based travel platform that uses serverless cloud technologies to improve customer experience [46]. This platform helps with hotel reservations, restaurant bookings, and custom travel packages from tour operators [46]. The main parts of the system are Users, Hotel Management, Restaurants, and Tour Operators [46].

Users must register or log in to access services, allowing them to book accommodations, reserve tables at restaurants, and arrange tours [46]. The system also supports user reviews, booking management, and travel advice [46]. The Hotel Management service handles room reservations and user interactions [46]. The Restaurant service manages table reservations, billing, and menu selections [46]. Tour Operator services offer custom travel packages and send information to users via email [46].

TripAdvisor uses a multi-cloud architecture with Google Cloud Platform (GCP) and Amazon Web Services (AWS) for high availability [46]. The React frontend's static assets are hosted in an AWS S3 bucket for online access [46]. User data and application info are stored in Cloud Firestore and Amazon DynamoDB, supporting user registration and multi-factor authentication [46]. Amazon Cognito manages and authorizes users, with support from DynamoDB and Firestore databases [46].

The application's virtual help system, powered by Amazon Lex, uses AWS Lambda functions to process user requests with DynamoDB data [46]. AWS Lambda and Google Cloud Functions handle most of the application's logic, replacing traditional server-based design [46]. Google Cloud Pub/Sub supports communication and concurrent request processing [46]. These cloud services are securely accessed through API Gateway services from both GCP and AWS [46].

To create personalized travel packages and analyze user reviews, the application leverages machine learning techniques from GCP, specifically Vertex AI and NLP API services [46]. Amazon CloudTrail and CloudWatch cloud services generate reports on user login statistics [46].

Our research highlights the seamless integration of these technologies to deliver a robust and efficient travel platform, enhancing the overall user experience [46].

Cloud Application-Based Research

Details of Research

Module 1: User Management and Authentication		
Research Area	Source	URL
AWS DynamoDB	AWS DynamoDB Documentation	https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Introduction.html
How to connect Lambda functions to DynamoDB	Baeldung	https://www.baeldung.com/aws-lambda-dynamodb-java
AWS Cognito	AWS Cognito Documentation	https://docs.aws.amazon.com/cognito/latest/developerguide/what-is-amazon-cognito.html
How to use AWS Cognito Authentication with Auth0	Medium	https://hyprstack.medium.com/reactjs-aws-cognito-authentication-with-auth0-f3f16094fa8
AWS Lambda	AWS Lambda Documentation	https://docs.aws.amazon.com/lambda/latest/dg/welcome.html
How to create and deploy AWS Lambda functions for authentication	DEV Community Blog	https://dev.to/lauracarballo/serverless-authentication-with-aws-lambda-427m
Module 2: Virtual Assistant		
Research Area	Source	URL
Creating chatbots using Amazon Lex	Amazon Lex V1 Developer Guide - Create an Amazon Lex Bot (Console)	https://docs.aws.amazon.com/lex/latest/dg/gs-bp-create-bot.html
Creating intents and custom slot types for chatbots	Amazon Lex V2 Developer Guide - Creating intents and slot types	https://docs.aws.amazon.com/lexv2/latest/dg/designing-create.html
Chatbot conversational flow	Userlike - Steps for Creating a Smooth Chatbot Conversation Flow	https://www.userlike.com/en/blog/chatbot-conversation-flow

Configuring intents to trigger AWS Lambda	Amazon Lex V2 Developer Guide - Setting intent context	https://docs.aws.amazon.com/lexv2/latest/dg/context-mgmt-active-context.html
Query processing and fetching data from Amazon DynamoDB using AWS Lambda	AWS Lambda Developer Guide - Using AWS Lambda with Amazon DynamoDB	https://docs.aws.amazon.com/lambda/latest/dg/with-ddb.html
Module 3: Passing Message		
Research Area	Source	URL
Working of pub/sub systems	Google Cloud Pub/Sub - Architectural overview of Pub/Sub	https://cloud.google.com/pubsub/architecture#:~:text=Pub%2FSub%20sends%20an%20acknowledgement,they%20have%20processed%20the%20message.
Creating a pub/sub topic	Google Cloud Pub/Sub - Create a topic	https://cloud.google.com/pubsub/docs/create-topic
Publishing messages to a topic	Publish messages to topics	https://cloud.google.com/pubsub/docs/publisher
Autoscaling of publisher forwarders and subscriber forwarders	Stack Overflow - PubSub: horizontally scaling subscriber forwarders	https://cloud.google.com/pubsub/architecture#:~:text=Pub%2FSub%20sends%20an%20acknowledgement,they%20have%20processed%20the%20message.
Connecting Cloud Function as a subscriber to GCP Pub/Sub	Google Cloud Functions - Pub/Sub triggers	https://cloud.google.com/functions/docs/calling/pubsub
Connecting Cloud Function to Amazon DynamoDB table	N8n - AWS DynamoDB and Google Cloud integration	https://n8n.io/integrations/aws-dynamodb/and/google-cloud/
Connecting Cloud Function to Google Firestore	Firebase – Firestore - Extend Cloud Firestore with Cloud Functions	https://firebase.google.com/docs/firestore/extend-with-functions
Connecting AWS to GCP	Revology - How to establish a connection between GCP and AWS	https://www.revolgy.com/insights/blog/how-to-establish-a-connection-between-gcp-and-aws
Creating a HA VPN	Google Cloud Network Connectivity - Create HA VPN connections	https://cloud.google.com/network-connectivity/docs/vpn/tutorial

	between Google Cloud and AWS	s/create-ha-vpn-connections-google-cloud-aws
Module 4: Notifications		
Research Area	Source	URL
AWS SNS	AWS SNS Documentation	https://docs.aws.amazon.com/sdk-for-javascript/v2/developer-guide/sns-examples.html
Sending Email Notifications with Amazon SNS	Mailtrap	https://mailtrap.io/blog/amazon-sns-guide/
AWS SQS	AWS SQS Documentation	https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/welcome.html
Accessing queues from Lambdas or other containers	LocalStack	https://docs.localstack.cloud/user-guide/aws/sqs/
Use SNS and SQS to Distribute and Throttle Events	Jeremy Daly	https://www.jeremydaly.com/how-to-use-sns-and-sqs-to-distribute-and-throttle-events/
Importance of security, error handling, and monitoring in the deployment and operation of SNS and SQS.	AWS	https://docs.aws.amazon.com/IAM/latest/UserGuide/best-practices.html
Module 5: Data Analytics and Visualization		
Research Area	Source	URL
Features	Locker Studio Documentation	https://developers.looker.com/embed/getting-started
AWS DynamoDB	AWS DynamoDB Documentation	https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Introduction.html
AWS Lambda Functions	AWS Lambda Documentation	https://docs.aws.amazon.com/lambda/latest/dg/welcome.html
AWS API Gateway	AWS API Gateway Docs	https://docs.aws.amazon.com/apigateway/latest/developerguide/welcome.html

Creating Serverless Applications with AWS Lambda and API Gateway	AWS Tutorials	https://docs.aws.amazon.com/apigateway/latest/developerguide/getting-started-lambda-non-proxy-integration.html
Connect Google Sheets with Locker Studio for Data Analytics and Visualizations	Coupler.io Blogs	https://blog.coupler.io/how-to-connect-google-sheets-to-looker-studio/
Steps to Embedded Locker Studio in the ReactJS application using iFrame.	Locker Studio Support	https://support.google.com/looker-studio/answer/7450249?hl=en#zippy=%2Cin-this-article
How to Access data stored in DynamoDB using Lambda function and create API gateway to expose that lambda function	AWS Documentation	https://docs.aws.amazon.com/apigateway/latest/developerguide/http-api-dynamo-db.html
How to work with API and Which Endpoints to call to analyze data.	Google Cloud Natural Language Processing Documentation	https://cloud.google.com/natural-language/docs/basics#interpreting_sentiment_analysis_values
Post Authentication Lambda Triggers	AWS Cognito Documentation	https://docs.aws.amazon.com/cognito/latest/developerguide/user-pool-lambda-post-authentication.html
Module 6: Web Application Building and Deployment		
Research Area	Source	URL
Deploying React App on Google Cloud Run and set up CI/CD from GitLab	DEV Community Blog	https://dev.to/rushi-patel/deploy-react-app-to-google-cloud-run-with-github-actions-cicd-a-complete-guide-52pf
Google Cloud Run	Google Cloud Run Documentation	https://cloud.google.com/build/docs/deploying-builds/deploy-cloud-run
Using Terraform IAC to deploy Images to Cloud Run	Google Cloud Run Documentation	https://cloud.google.com/run/docs/deploying#terraform
Using Google Cloud Build Service to build Docker	Google Cloud Build Documentation	https://cloud.google.com/build/docs/building/build-containers

Image and push to Google Cloud Repository		

Learning Outcomes of the Research

Module 1: User Management and Authentication

In our research for the User Management & Authentication module, we focused on understanding and planning the use of several key technologies. We began with Amazon DynamoDB, a NoSQL database service that is ideal for storing user details due to its low latency and scalability [10]. We learned how to set up and manage DynamoDB tables, connect Lambda functions to DynamoDB, and ensure efficient data operations [11], [14].

Next, we explored Amazon Cognito for implementing multi-factor authentication (MFA). Cognito simplifies user authentication with built-in support for various methods [12]. We learned to configure user pools, enable MFA, and integrate Cognito with a front-end application for secure user authentication [13].

For the second authentication factor using questions and answers, we focused on AWS Lambda. Lambda functions allow us to execute backend code in response to specific events without managing servers [14]. We learned how to set up and configure Lambda functions for authentication processes [15].

Finally, for the third authentication factor, we looked into implementing a Caesar cipher using AWS Lambda. This involves creating a Lambda function to encrypt and decrypt text using the Caesar cipher algorithm [15]. We learned how to develop and deploy this cryptographic function in a serverless environment [14].

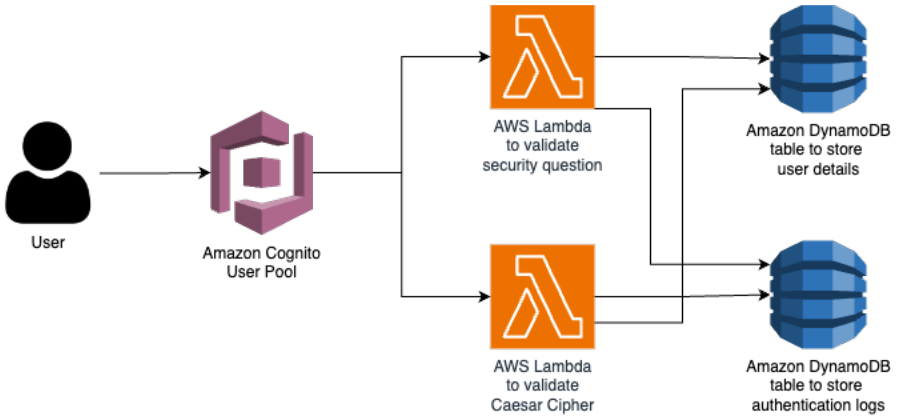

Figure 1: Architecture Diagram of User Management and Authentication Module

Module 2: Virtual Assistant Module

While researching how to create interactive chatbots and how they respond to queries, we found that Amazon Lex makes it very straightforward to create a chatbot [16]. We do not need to deal with learning natural language processing and training the model; rather, Amazon Lex does it all for us using its Natural Language Understanding (NLU) [16], [17].

After creating a chatbot, we need to specify intents, sample utterances, and slots [16]. Intents are the queries a user can make, e.g., booking a hotel room or connecting a user to a representative [17]. Sample utterances are the phrases that initiate an intent. For example, “Book 3 nights in Paris for 4 people” [16]. Slots refer to the information required to fulfill an intent. In the above example, to fulfill a room-booking intent, the bot needs to know the city, number of nights, check-in date, room type, and number of people [18]. Some of these might be included in sample utterances. For the rest, the bot asks prompt questions [19].

Once all the slots are known, Amazon Lex extracts the relevant parameters from the query and triggers an AWS Lambda function, which processes the necessary logic to store, query, and retrieve the data [14]. For instance, it will store the customer's name, the concern, and the contact information in a CustomerConcerns DynamoDB table, or publish a message to all the subscribers through a GCP Pub/Sub topic [20].

Lastly, the function will return the response to Lex, which will then update the user through the chat interface [17], [19].

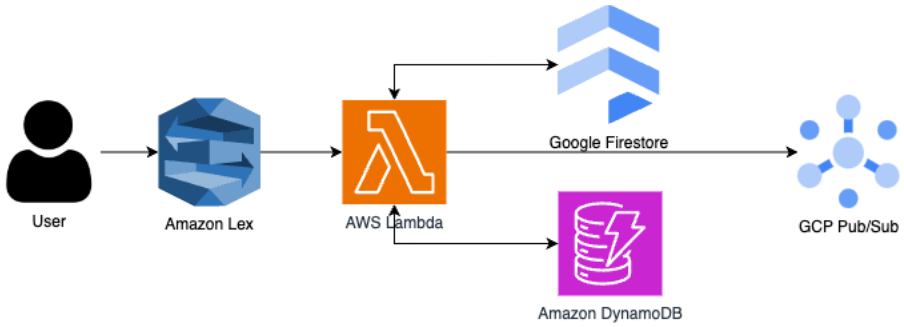

Figure 2: Architecture Diagram of Virtual Assistant Module

Module 3: Passing Messages

While trying to figure out what the GCP Pub/Sub architecture looks like, we found a few interesting aspects of it. GCP Pub/Sub is a distributed system with four main layers: publishers, publishing forwarders, subscribing forwarders, and subscribers [21]. In our case, the publisher would be AWS Lambda [14]. As discussed in the previous section, AWS Lambda will publish the query message to a set of subscribers for a topic [21]. Then, publishing forwarders instantly write the message and its metadata to the storage [21].

Following that, when subscribers are ready, subscribing forwarders request publishing forwarders to hand over the message, which they then pass over to subscribers [21]. In this case, the subscribers will be Cloud Functions, which will deal with processing the query, storing it in Amazon DynamoDB or Google Firestore, fetching agent information from Amazon DynamoDB or Google Firestore, assigning a random agent for customer support, and sending a notification to the assigned agent using Amazon SNS and/or Amazon SQS [22], [23], [24].

The choice between Amazon DynamoDB and Google Firestore will be based on the database redundancy and data security requirements, which will be explored in the next sprint [25], [26].

The most difficult part of the project will be implementing a multi-cloud deployment model [27]. We found that one way to establish direct communication across two cloud platforms is by creating Highly Available Virtual Private Networks (HA VPNs) [28]. This can be achieved with the help of various services like Cloud Router, HA VPN Gateway, VPN Tunnels, and Peer VPN Gateways [28]. More on this architecture is in the scope of the next sprint [27].

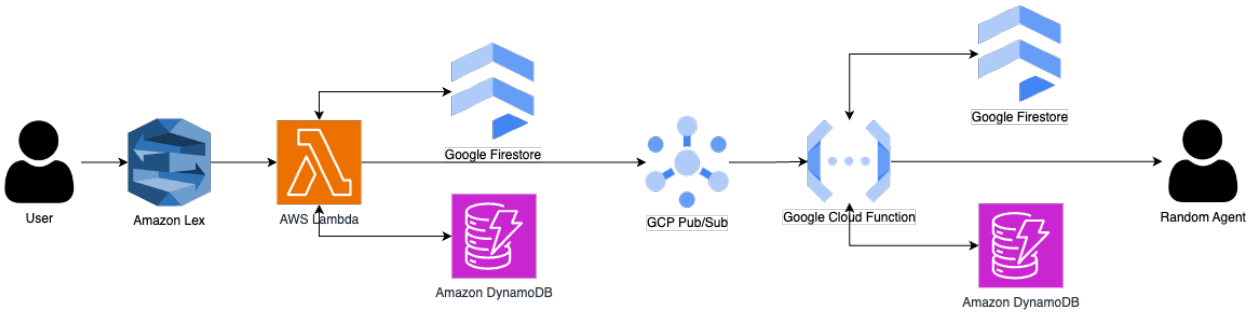

Figure 3: Architecture Diagram of Passing Messages Module

Module 4: Notifications

Our research focused on integrating AWS Simple Notification Service (SNS) and Simple Queue Service (SQS) to improve the notification system of our cloud-based application [29], [30]. We explored using SNS for sending emails about key events like successful registrations and logins and learned how to create and manage SNS topics for various notifications [29], [31].

We then examined AWS SQS for managing asynchronous communication between application components and setting up a queue for handling room booking requests [30]. These requests are processed by an AWS Lambda function, which decides on booking approvals or denials and communicates the outcomes via SNS [14], [32].

Our findings emphasized the importance of security, error handling, and monitoring, using IAM roles and policies to secure SNS topics and SQS queues, and implementing robust error handling strategies [33].

Figure 4: Architecture Diagram of Notification Module

Module 5: Data Analytics and Visualization

Going over the official documentation of various cloud services, we got an overall idea of how we will be implementing the Data Analytics and Visualization module [34], [35].

We learned about Google Looker Studio, which allows us to present our raw data in informative, easy-to-read ways such that we can get insights from the data by

visualizing the data using different diagrams like bar graphs, pie charts, histograms, etc. [37]. This tool is used for Data Analytics and Visualization purposes [36].

This tool provides a feature to visualize Google Sheets, so we are planning to write a Google App Script that will periodically fetch data from Lambda function via API Gateway and populate the Google Sheets [35]. We will then use Looker Studio to analyze that Google Sheet and then embed that Looker Studio Report in React App using an iFrame [37].

According to the requirement, we need to display Login Statistics of the users on the Admin Page [35]. We learned about storing login statistics in the AWS DynamoDB after Login [10]. After the successful login into the application using AWS Cognito, we can set up a post-authentication trigger which will run a lambda function to store data about Login Statistics of the user in DynamoDB [12], [40].

We also need to display feedback of each room in a tabular structure in the frontend according to the requirement [14]. To accomplish this, we learned about how to access data stored in DynamoDB using a Lambda function and how to expose that lambda function using API Gateway service of AWS [38]. This way, we can request that API endpoint which will fetch data from DynamoDB using Lambda function [14], [20].

Moreover, we also need to analyze all feedback and show the sentiment of each feedback in the front-end [39]. For this, we have gone through the Google Natural Language Processing API Documentation [39]. The Natural Language API has several methods for performing analysis and annotation on your text [39]. Out of all, the analyzeSentiment method takes given text as input and in response provides score (-1 to 1) and magnitude (0 to 1) values [39]. Different score values give different sentiment levels like Clearly positive (above 0.8), Clearly Negative (below -0.6), Neutral (0.1), Mixed (0.0) [39].

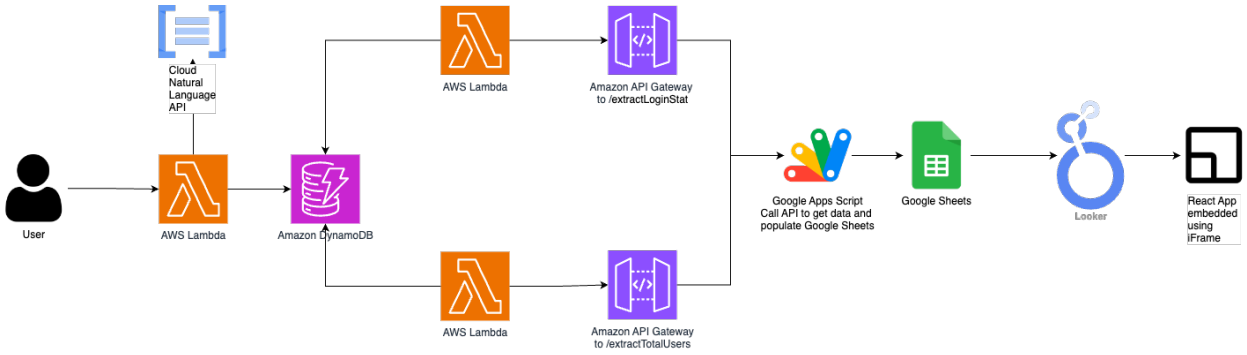

Figure 5: Architecture Diagram of Data Analytics and Visualization Module

Module 6: Web Application Building and Deployment

Going over the official documentation of Google Cloud Run and blogs on how to deploy React App on Cloud Run, we learned about how to configure CI/CD pipeline from GitLab and deploy React App on Google Cloud Run [41], [42].

Google Cloud Build is used for building images of applications, and that image is deployed on Cloud Run [42], [43]. We first need to create a Dockerfile to create the image of the React App and then push that image to Google Cloud Build or Google Cloud Repository, and then from that, we run a container created out of that image on Cloud Run [42], [43]. Every step is automated using GitLab CI/CD pipeline and Terraform for automatic deployment of Google Cloud Resources and React App on Google Cloud Run [42], [44].

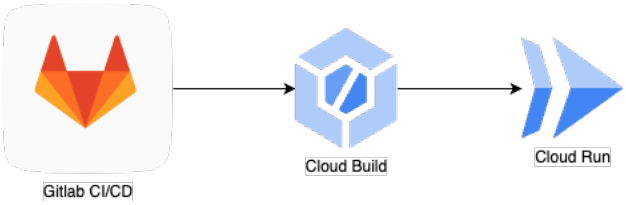

Figure 6: Architecture Diagram of Web Application Building and Deployment Module

Architecture Diagram of the Application

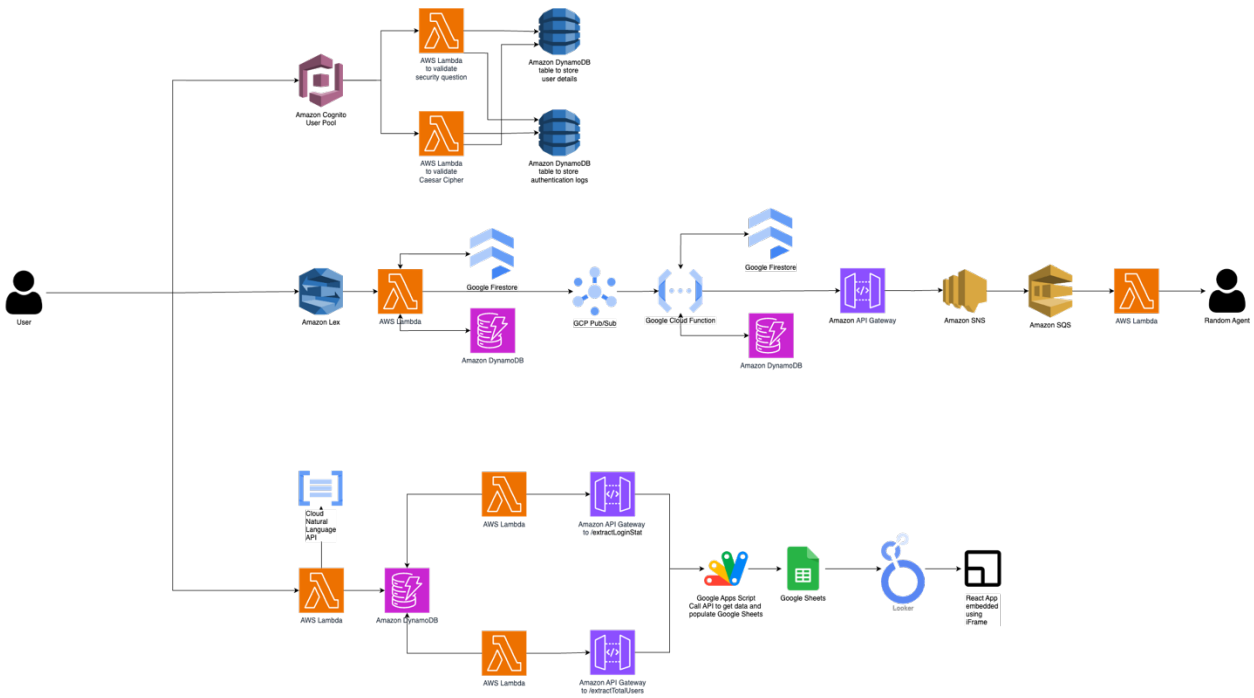

Figure 7: Architecture Diagram of the Application

Project Management Tool

We have selected GitLab for our project management. GitLab is ideal for our needs due to its comprehensive features for tracking tasks, managing workflows, and facilitating team collaboration. The platform allows us to create and assign tasks, set deadlines, and monitor progress efficiently. Our team is already familiar with GitLab, which makes it easier to use and ensures a smooth workflow.

Sprint Planning:

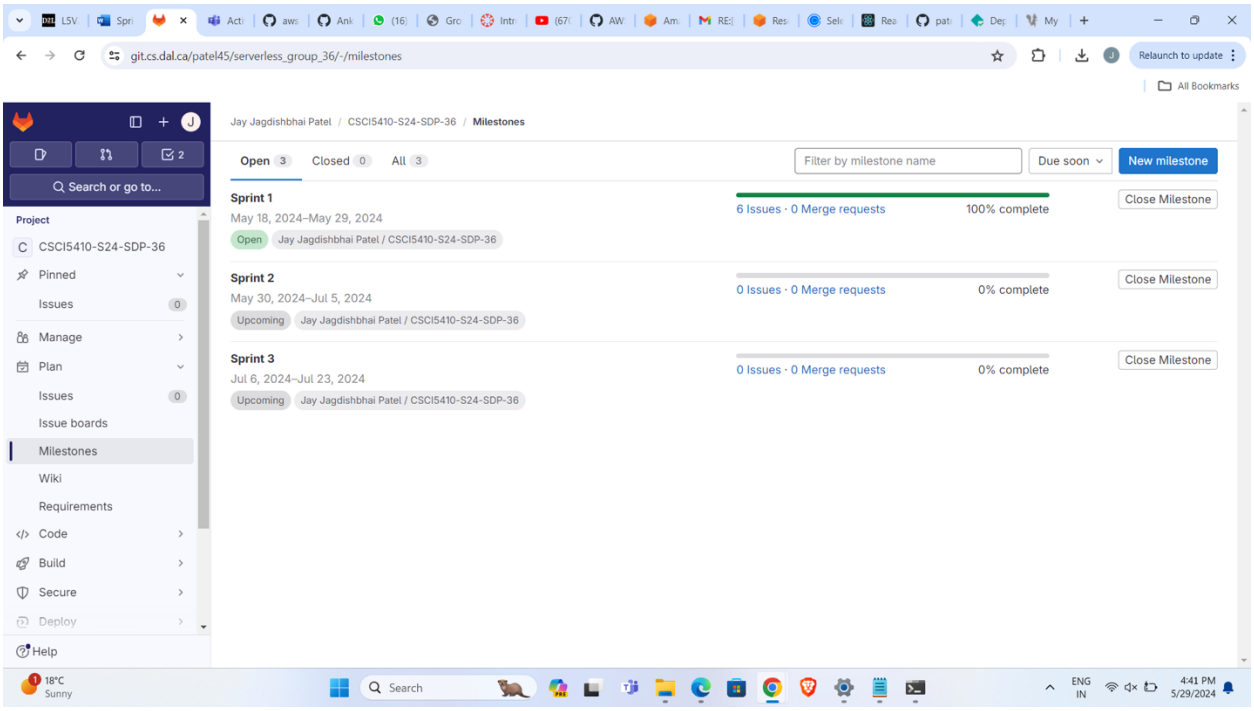

Figure 8: Milestone created for each Sprint

Sprint 1:

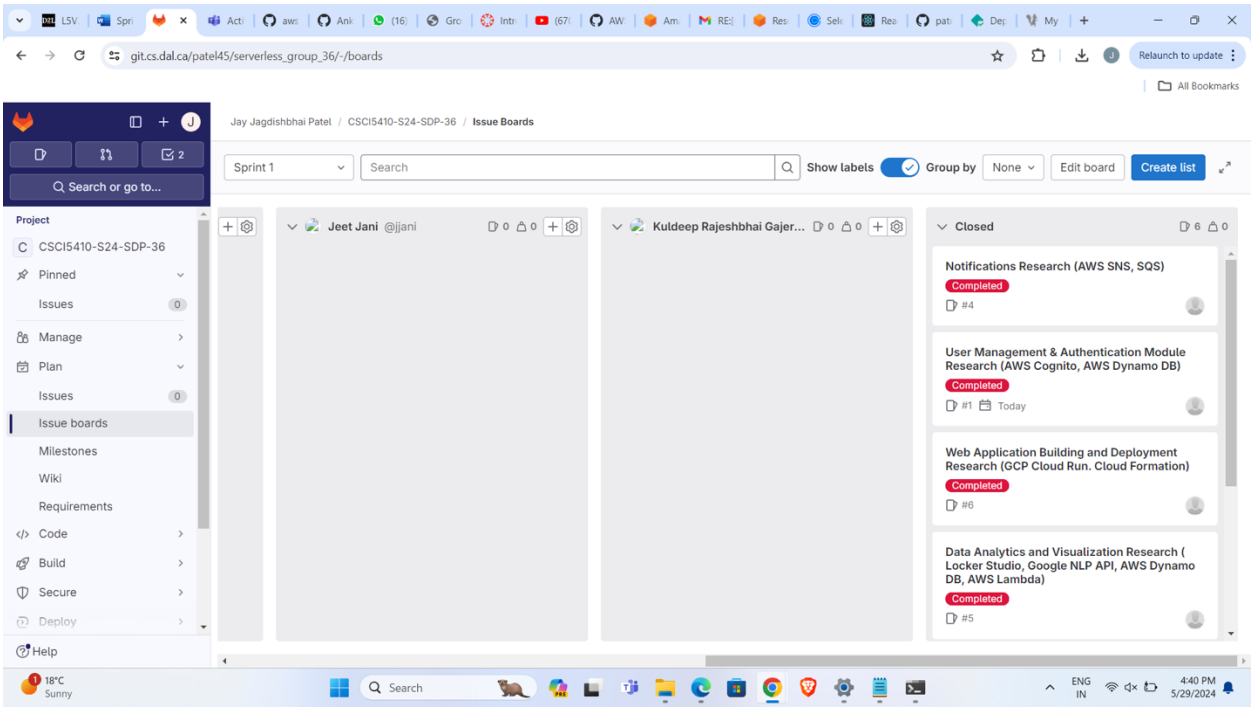

Figure 9: Sprint 1 Dashboard on 29th May, 2024

Member Contribution

Member	Tasks Taken
Christin Saji	Module 1, Task Allocation and Workflow, Project Management Tool, Meeting Logs.
Jay Jagdishbhai Patel	Module 5, Module 6, Architecture-Oriented Research, Application Perspective Research, Gantt Chart.
Jeet Jani	Module 2, Module 3, Architecture Diagrams, Application Perspective Research.
Kuldeep Rajeshbhai Gajera	Module 4, References, Citation, Captions.

Christin Saji:

- Conducted in-depth research on Module 1 (User Management and Authentication), including understanding the use of AWS Cognito, DynamoDB, and Lambda for multi-factor authentication.
- Created a detailed architecture diagram for the User Management and Authentication module, illustrating the flow and integration of various components such as AWS services and the front-end application.
- Provided a brief description of the Gantt chart and developed the Task Allocation and Workflow plan, outlining the key tasks, their durations, and the order in which they need to be completed.
- Explained the selection of GitLab as our project management tool, detailing its features and benefits for tracking tasks, managing workflows, and facilitating team collaboration.

Jay Jagdishbhai Patel:

- I have done research regarding Serverless Architecture, which helps our team to understand what Serverless Architecture is, its best practices and different design patterns to follow for different tasks.
- Second, research about one similar serverless application, which helped our team to get clarity around how serverless applications and different AWS services can be structured and used to build complete application.
- Also, I have done research regarding the Data Analytics and Visualization module and Web Application and Deployment module, which provides our team with a brief overview about how these two modules can be implemented.
- At last, Setup GitLab Repo and Issue Board and Created Gantt Chart for project Tracking and management purpose.

Jeet Jani

- Worked on Cloud application-based research on building chatbot and integrating it with the serverless architecture of hotel management application.
- Researched on implementing multi-cloud deployment using HA VPN with AWS and GCP.
- Undertook the task of finding out how GCP Pub/Sub can be utilized to pass event driven messages.
- Designed Architecture diagrams for all modules.
- Researched on Trip Advisor, an existing hotel management application on AWS.
- Contributed to formatting the document.

Kuldeep Rajeshbhai Gajera

- My contribution to the Notifications module involved conducting thorough research and preparing a detailed report on implementing AWS Simple Notification Service (SNS) and Simple Queue Service (SQS). I explored how to leverage SNS for sending emails about key events like successful registrations and logins and examined SQS for its capability to manage asynchronous communication between application components, particularly for room booking requests.
- Additionally, I focused on the importance of security, error handling, and monitoring, proposing the use of IAM roles and policies to safeguard these services.
- I contributed to project documentation and management, compiling a comprehensive research table, adding references, inline citations, and handling the overall report formatting to ensure clarity and coherence in our project deliverables.

Gantt Chart

The Gantt chart below illustrates the timeline for the project, indicating the start and end dates for each task. There are no overlapping tasks, ensuring that each phase is completed before the next one begins.

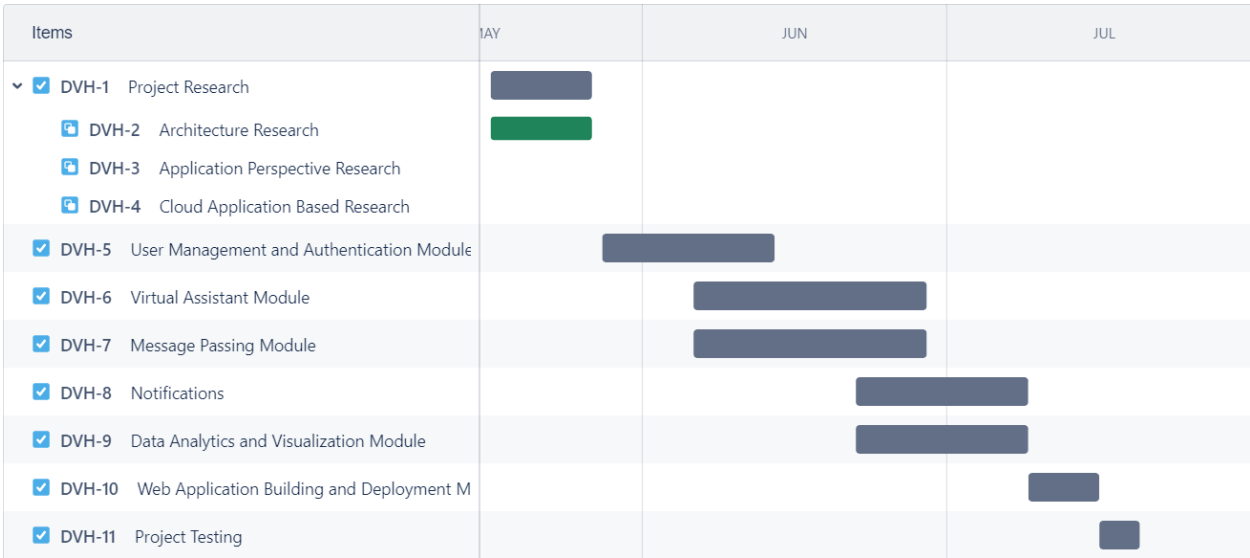

Figure 10: Clear and sequential task timeline for project execution

Using GitLab, we will continuously update the status of tasks, track the completion of milestones, and ensure that the project is progressing as planned. This approach provides a clear structure and overview of the technical specifications and intended tasks, promoting accountability and collaboration among team members.

Task Allocation and Workflow

The project is divided into several key tasks. Below is an overview of the tasks and their respective durations:

1. **Project Research:**
    - **Task:** Conduct architecture, application perspective, and cloud application research.
    - **Duration:** May 17, 2024 – May 29, 2024
2. **User Management and Authentication Module:**
    - **Task:** Implement user registration, validation, and multi-factor authentication.
    - **Duration:** May 28, 2024 – June 13, 2024

3. **Virtual Assistant Module:**
    - **Task:** Develop a virtual assistant using AWS Lex and integrate it with the system.
    - **Duration:** June 6, 2024 – June 28, 2024
4. **Message Passing Module:**
    - **Task:** Implement message-passing functionality using GCP Pub/Sub.
    - **Duration:** June 6, 2024 – June 28, 2024
5. **Notifications:**
    - **Task:** Set up notifications for user actions using AWS SNS and SQS.
    - **Duration:** June 22, 2024 – July 8, 2024
6. **Data Analytics and Visualization Module:**
    - **Task:** Analyze data and create visualizations using Google Natural Language API and Looker Studio.
    - **Duration:** June 22, 2024 – July 8, 2024
7. **Web Application Building and Deployment:**
    - **Task:** Build and deploy the front-end application using React and GCP Cloud Run.
    - **Duration:** July 9, 2024 – July 15, 2024
8. **Project Testing:**
    - **Task:** Perform comprehensive testing of all modules.
    - **Duration:** July 15, 2024 – July 19, 2024

Meeting Logs

Agenda	Outcome of discussion	Link to meeting recordings
1. Review project requirements 2. Discuss tasks for Sprint 1 and the overall project 3. Share knowledge on key concepts from the project requirements	• Reviewed and understood the project requirements. • Identified tasks for Sprint 1. • Discussed the overall project timeline and deliverables. • Shared knowledge and clarified concepts mentioned in the requirements.	Call with Serverless Team 36-20240518_113442-Meeting Recording.mp4
1. Review completed documentation for Sprint 1 2. Identify and assign remaining tasks 3. Plan next steps and timeline for completion	• Reviewed documentation completed for Sprint 1, including module-based research, application perspective research, and architecture-oriented research. • Identified parts that needed to be added or edited, including the architecture diagram, Gantt chart, and meeting log. • Divided the remaining work among team members and set deadlines for completion.	Call with Serverless Team 36-20240526_151700-Meeting Recording.mp4

Team Conversations Screenshots

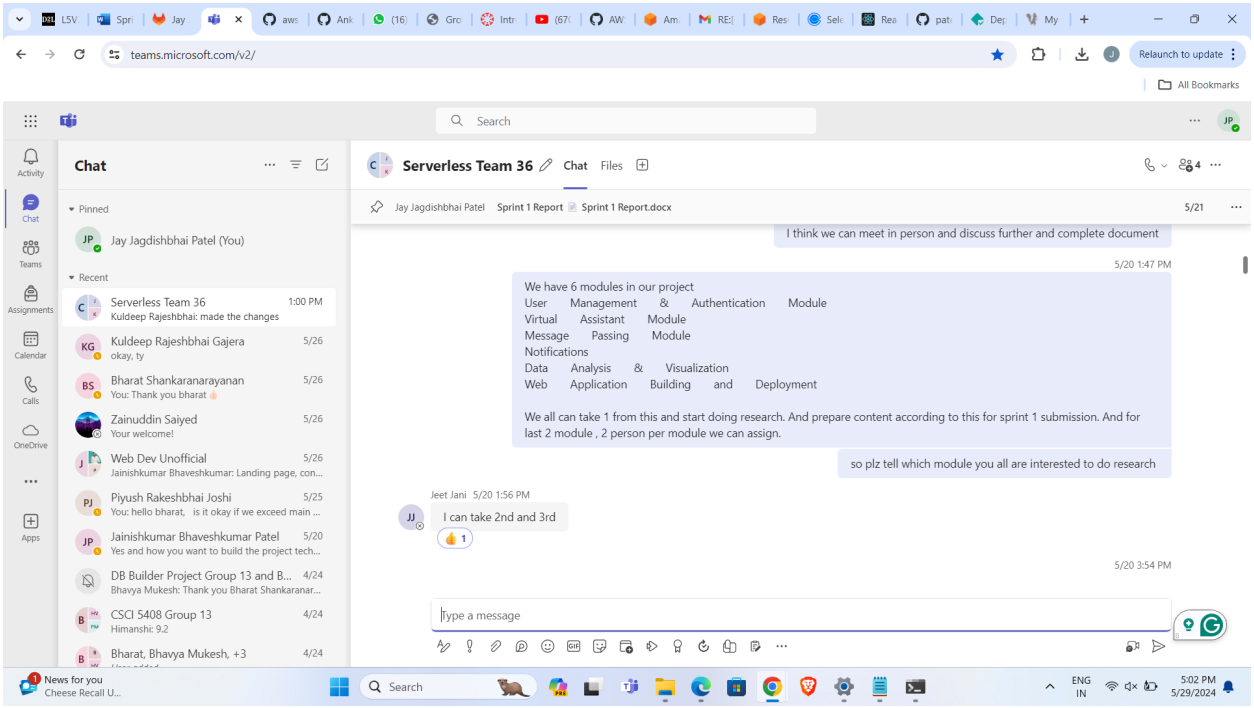

Figure 11: MS Teams Chat Screenshot

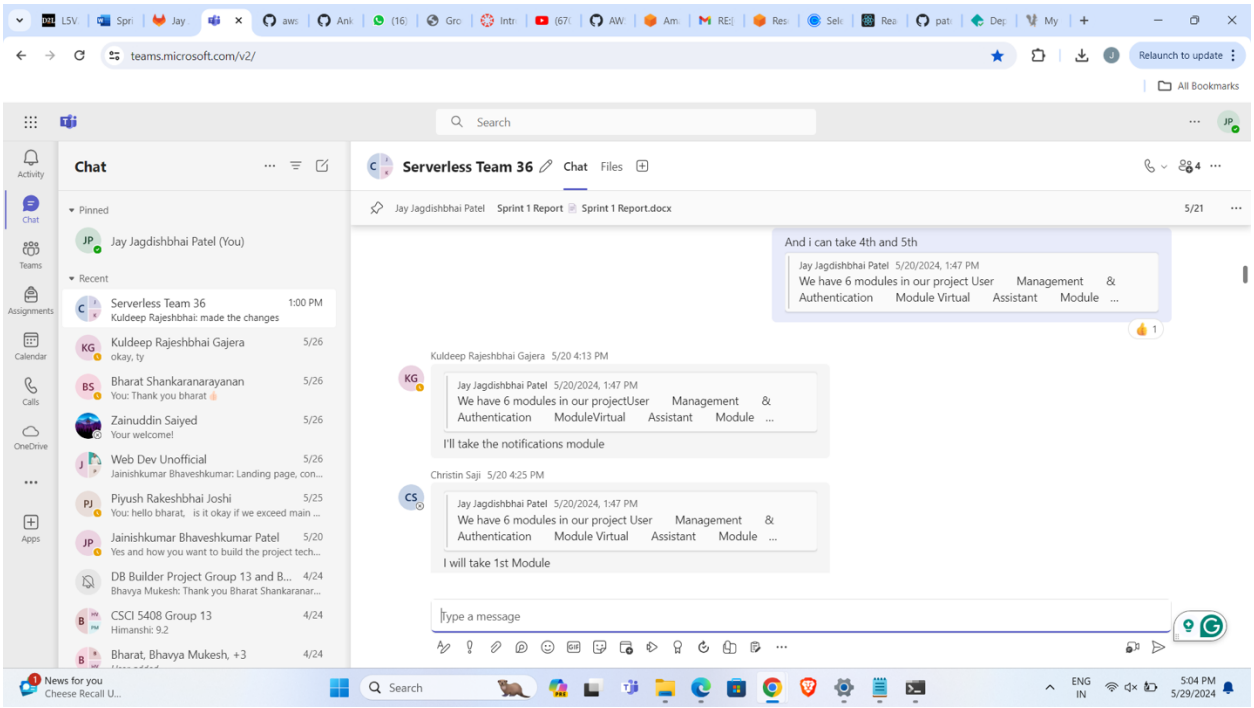

Figure 12: MS Teams Chat Screenshot

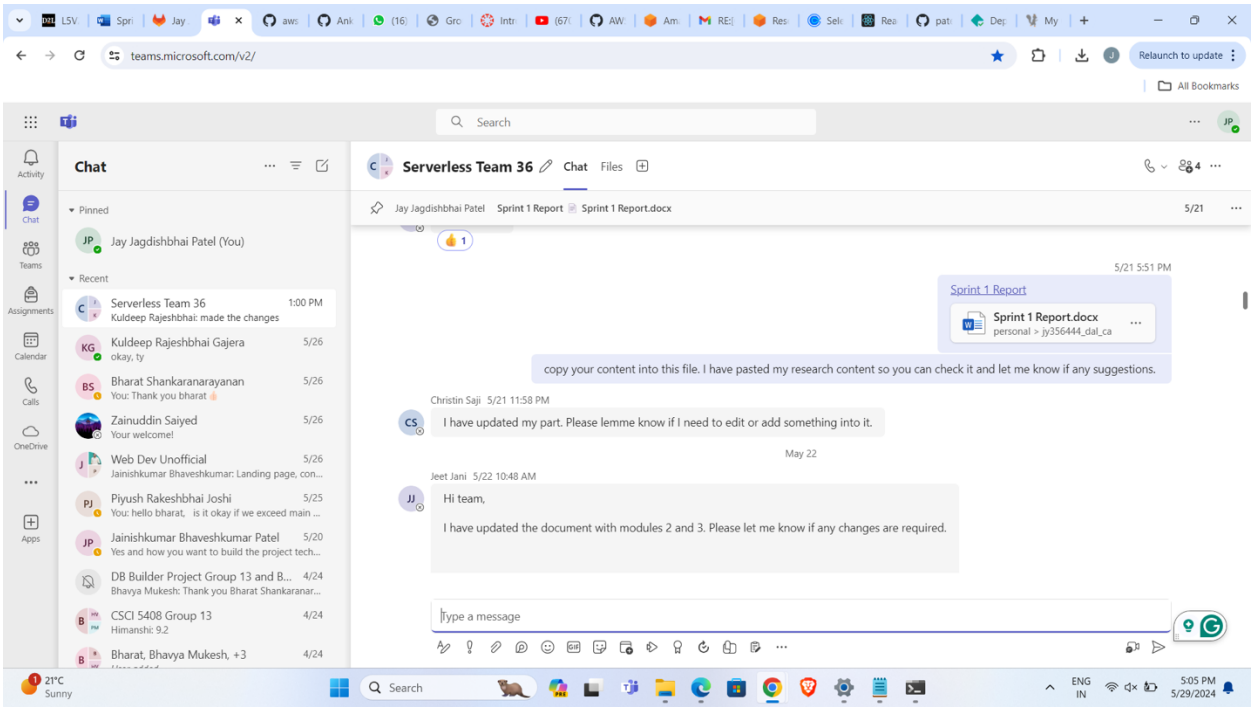

Figure 13: MS Teams Chat Screenshot

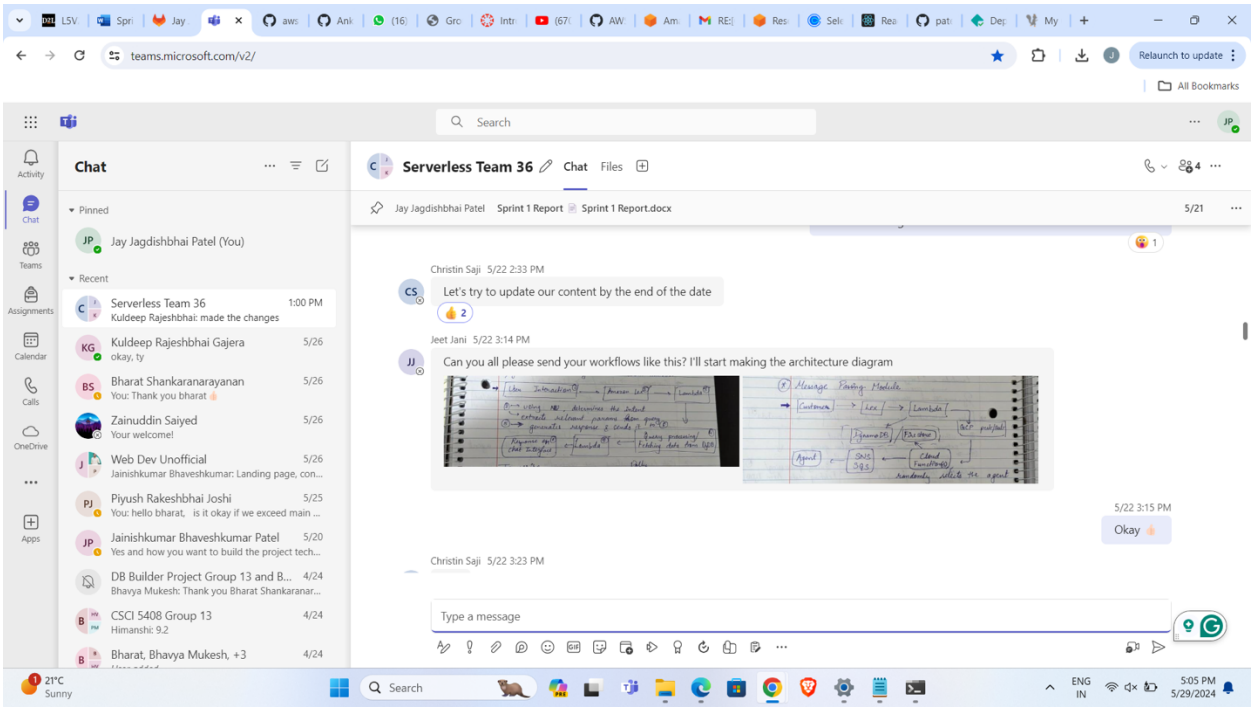

Figure 14: MS Teams Chat Screenshot

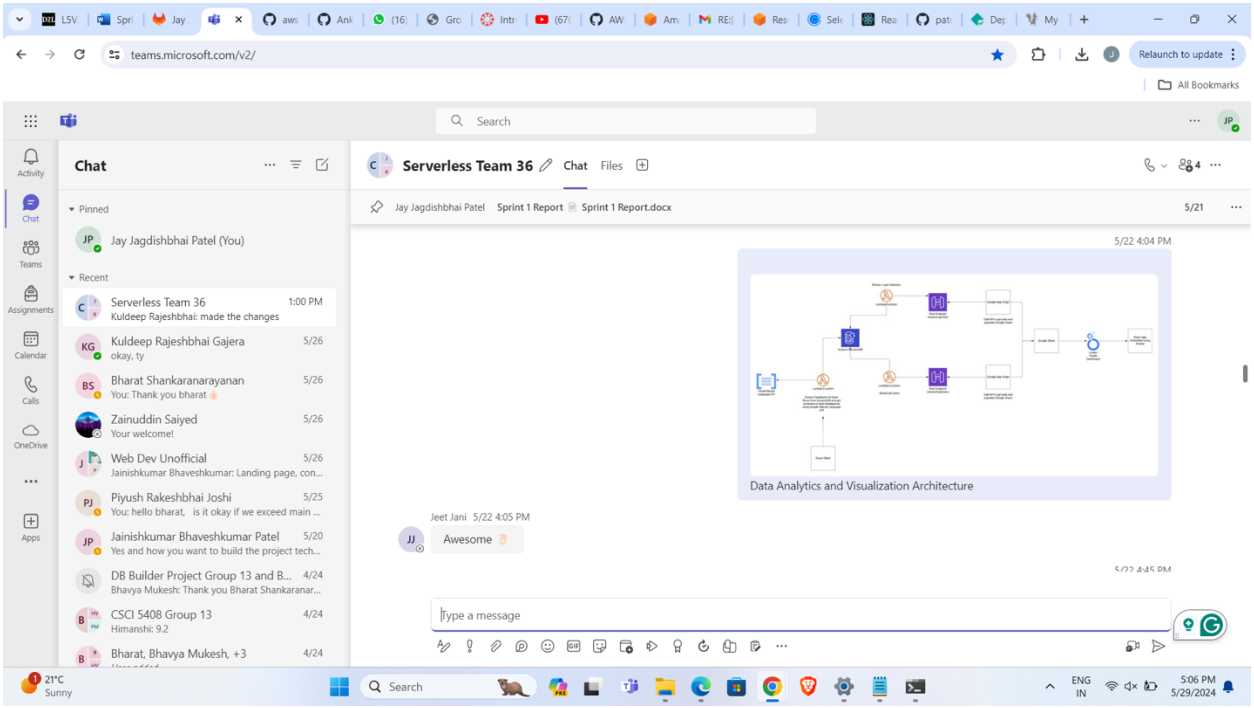

Figure 15: MS Teams Chat Screenshot

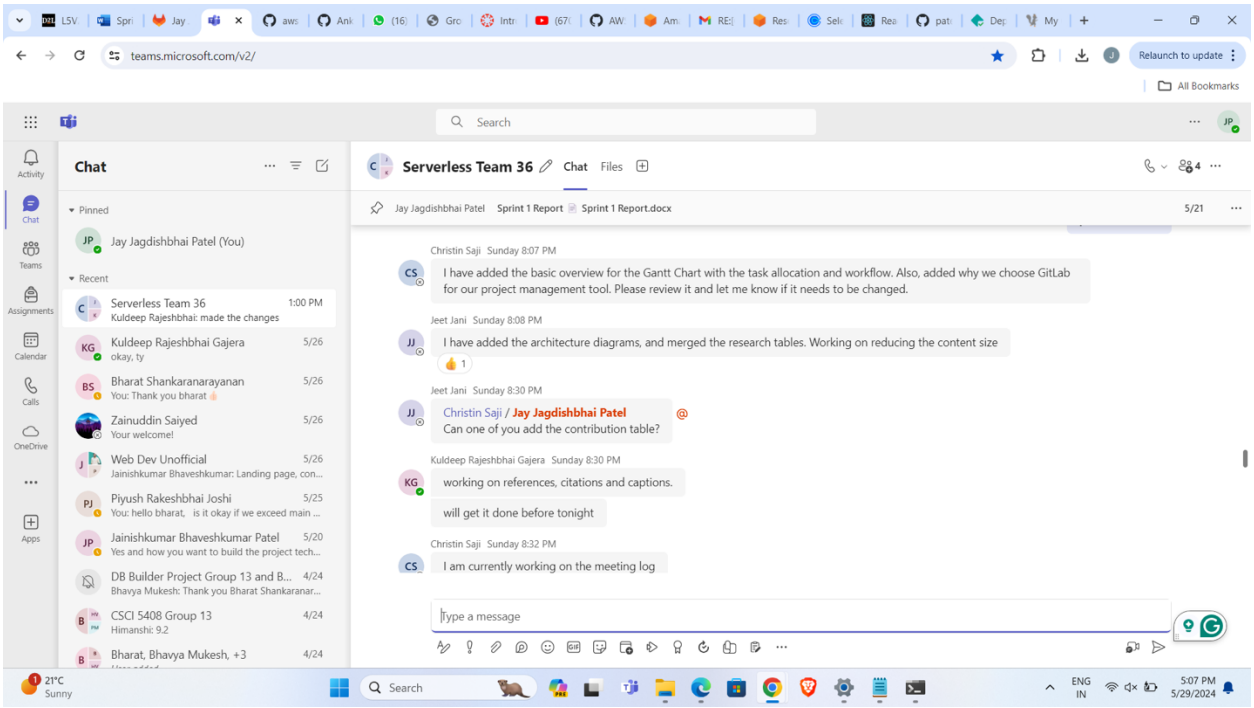

Figure 16: MS Teams Chat Screenshot

References

[1]	*J. Beswick, "Best practices for organizing larger serverless applications," Amazon.com*. [Online]. Available: https://aws.amazon.com/blogs/compute/best-practices-for-organizing-larger-serverless-applications/. [Accessed: May 20, 2024].
[2]	M. Ozkaya, "AWS Serverless Architectural Patterns and Best Practices," *AWS Serverless Microservices with Patterns & Best Practices*, 12-Mar-2022. [Online]. Available: https://medium.com/aws-serverless-microservices-with-patterns-best/aws-serverless-architectural-patterns-and-best-practices-d2d446375924. [Accessed: May 20, 2024].
[3]	*L. Jiang, Y. Pei, J. Zhao, "Overview Of Serverless Architecture Research," Researchgate.net*. [Online]. Available: https://www.researchgate.net/publication/339680821_Overview_Of_Serverless_Architecture_Research. [Accessed: May 20, 2024].
[4]	*J. Saji, A. Kumar, "A Review Paper on Serverless Architecture of Web Applications," Ssrn.com*. [Online]. Available: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4495958. [Accessed: May 20, 2024].
[5]	F. Oyama, "Serverless architecture patterns and best practices," *freecodecamp.org*, 09-Jan-2024. [Online]. Available: https://www.freecodecamp.org/news/serverless-architecture-patterns-and-best-practices/. [Accessed: May 20, 2024].
[6]	Shweta Joshi, "Hotel Booking System built using Angular 4," GitHub. [Online]. Available: https://github.com/shwetajoshi601/aws-serverless-hotelbooking. [Accessed: May 27, 2024].
[7]	Ankush Mudgal, "Serverless Bed & Breakfast," GitHub. [Online]. Available: https://github.com/AnkushMudgal/serverless_bnb. [Accessed: May 27, 2024].
[8]	*AWS Samples, "Bookstore Demo App built on AWS," GitHub. [Online]. Available: https://github.com/aws-samples/aws-bookstore-demo-app. [Accessed:* May 27, 2024*].*
[9]	H. Do, "A year of running a hotel booking application on AWS Serverless services for $0.8/month," *Hieu's Technical Blog*, 29-Oct-2023. [Online]. Available: https://hieudd.substack.com/p/a-year-of-running-a-hotel-booking. [Accessed: May 21, 2024].
[10]	"What is Amazon DynamoDB?" *Amazon.com*. [Online]. Available: https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Introduction.html. [Accessed: May 21, 2024].

[11]	*Baeldung, "AWS Lambda Using DynamoDB With Java," Baeldung.com*. [Online]. Available: https://www.baeldung.com/aws-lambda-dynamodb-java. [Accessed: May 21, 2024].
[12]	"What is Amazon Cognito?" *Amazon.com*. [Online]. Available: https://docs.aws.amazon.com/cognito/latest/developerguide/what-is-amazon-cognito.html. [Accessed: May 21, 2024].
[13]	M. Mendes, "ReactJs — AWS Cognito Authentication with Auth0," *Medium*, 21-Nov-2019. [Online]. Available: https://hyprstack.medium.com/reactjs-aws-cognito-authentication-with-auth0-f3f16094fa8. [Accessed: May 21, 2024].
[14]	"What is AWS Lambda?" *Amazon.com*. [Online]. Available: https://docs.aws.amazon.com/lambda/latest/dg/welcome.html. [Accessed: May 22, 2024].
[15]	L. Carballo, "Serverless authentication with AWS lambda," *DEV Community*, 06-Apr-2021. [Online]. Available: https://dev.to/lauracarballo/serverless-authentication-with-aws-lambda-427m. [Accessed: May 22, 2024].
[16]	"Create an Amazon Lex Bot (Console)" *Amazon.com*. [Online]. Available: https://docs.aws.amazon.com/lex/latest/dg/gs-bp-create-bot.html. [Accessed: May 22, 2024].
[17]	"Creating intents and slot types" *Amazon.com*. [Online]. Available: https://docs.aws.amazon.com/lexv2/latest/dg/designing-create.html. [Accessed: May 22, 2024].
[18]	Leah, "6 steps for creating a smooth chatbot conversation flow," *Userlike*, 14-Mar-2024. [Online]. Available: https://www.userlike.com/en/blog/chatbot-conversation-flow. [Accessed: May 22, 2024].
[19]	"Setting intent context" *Amazon.com*. [Online]. Available: https://docs.aws.amazon.com/lexv2/latest/dg/context-mgmt-active-context.html. [Accessed: May 22, 2024].
[20]	"Using AWS Lambda with Amazon DynamoDB" *Amazon.com*. [Online]. Available: https://docs.aws.amazon.com/lambda/latest/dg/with-ddb.html. [Accessed: May 22, 2024].
[21]	"Architectural overview of pub/sub," *Google Cloud*. [Online]. Available: https://cloud.google.com/pubsub/architecture. [Accessed: May 23, 2024].
[22]	"Create a topic," *Google Cloud*. [Online]. Available: https://cloud.google.com/pubsub/docs/create-topic. [Accessed: May 23, 2024].
[23]	"Publish messages to topics," *Google Cloud*. [Online]. Available: https://cloud.google.com/pubsub/docs/publisher. [Accessed: May 23, 2024].

[24]	"Pub/Sub triggers," *Google Cloud*. [Online]. Available: https://cloud.google.com/functions/docs/calling/pubsub. [Accessed: May 23, 2024].
[25]	"AWS DynamoDB and Google Cloud low-code integration," *N8n.io*. [Online]. Available: https://n8n.io/integrations/aws-dynamodb/and/google-cloud/. [Accessed: May 23, 2024].
[26]	"Extend cloud firestore with cloud functions," *Firebase*. [Online]. Available: https://firebase.google.com/docs/firestore/extend-with-functions. [Accessed: May 23, 2024].
[27]	"How to establish a connection between GCP and AWS," *Revolgy.com*. [Online]. Available: https://www.revolgy.com/insights/blog/how-to-establish-a-connection-between-gcp-and-aws. [Accessed: May 23, 2024].
[28]	"Create HA VPN connections between Google Cloud and AWS," *Google Cloud*. [Online]. Available: https://cloud.google.com/network-connectivity/docs/vpn/tutorials/create-ha-vpn-connections-google-cloud-aws. [Accessed: May 23, 2024].
[29]	P. Malek, "Amazon SNS tutorial for sending email notifications," *Mailtrap*, May 20, 2024.
[30]	"What is Amazon Simple Queue Service" *Amazon.com*. [Online]. Available: https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/welcome.html. [Accessed: May 20, 2024].
[31]	"Simple Queue Service (SQS)," *Docs*. [Online]. Available: https://docs.localstack.cloud/user-guide/aws/sqs/. [Accessed: May 24, 2024].
[32]	"How to: Use SNS and SQS to distribute and throttle events," *Jeremydaly.com*. [Online]. Available: https://www.jeremydaly.com/how-to-use-sns-and-sqs-to-distribute-and-throttle-events/. [Accessed: May 24, 2024].
[33]	*"Security best practices in IAM" Amazon.com*. [Online]. Available: https://docs.aws.amazon.com/IAM/latest/UserGuide/best-practices.html. [Accessed: May 24, 2024].
[34]	"Get started with embedding Looker," *Looker.com*. [Online]. Available: https://developers.looker.com/embed/getting-started. [Accessed: May 25, 2024].
[35]	"Tutorial: Build a CRUD API with Lambda and DynamoDB" *Amazon.com*. [Online]. Available: https://docs.aws.amazon.com/apigateway/latest/developerguide/http-api-dynamo-db.html. [Accessed: May 27, 2024].
[36]	O. Cheban, "Connect Google Sheets to Looker Studio (Google data studio)," *Coupler.io Blog*, 29-May-2023. [Online]. Available: https://blog.coupler.io/how-to-connect-google-sheets-to-looker-studio/. [Accessed: May 25, 2024].

[37]	"Embed a report," *Google.com*. [Online]. Available: https://support.google.com/looker-studio/answer/7450249?hl=en. [Accessed: May 25, 2024].
[38]	"Tutorial: Build a CRUD API with Lambda and DynamoDB" *Amazon.com*. [Online]. Available: https://docs.aws.amazon.com/apigateway/latest/developerguide/http-api-dynamo-db.html. [Accessed: May 25, 2024].
[39]	"Natural language API basics," *Google Cloud*. [Online]. Available: https://cloud.google.com/natural-language/docs/basics. [Accessed: May 25, 2024].
[40]	"Post authentication Lambda trigger" *Amazon.com*. [Online]. Available: https://docs.aws.amazon.com/cognito/latest/developerguide/user-pool-lambda-post-authentication.html. [Accessed: May 27, 2024].
[41]	R. Patel, "Deploy react app to Google Cloud run with github actions CI/CD - A complete guide," *DEV Community*, 10-Mar-2024. [Online]. Available: https://dev.to/rushi-patel/deploy-react-app-to-google-cloud-run-with-github-actions-cicd-a-complete-guide-52pf. [Accessed: May 27, 2024]].
[42]	"Deploying to Cloud Run using Cloud Build," *Google Cloud*. [Online]. Available: https://cloud.google.com/build/docs/deploying-builds/deploy-cloud-run. [Accessed: May 27, 2024]].
[43]	"Deploying to cloud run," *Google Cloud*. [Online]. Available: https://cloud.google.com/run/docs/deploying. [Accessed: May 27, 2024]].
[44]	"Build container images," *Google Cloud*. [Online]. Available: https://cloud.google.com/build/docs/building/build-containers. [Accessed: May 27, 2024]].
[45]	"Flowchart maker & online diagram software," *Diagrams.net*. [Online]. Available: https://app.diagrams.net/. [Accessed: May 27, 2024]].
[46]	H. Scalability, "An epic TripAdvisor update: Why not run on the cloud? The grand experiment. - high scalability," *High Scalability*, 02-Oct-2012. [Online]. Available: https://highscalability.com/an-epic-tripadvisor-update-why-not-run-on-the-cloud-the-gran/. [Accessed: May 27, 2024]].
[47]	Atlassian, "Unlock your team's best work with Jira Software," *Atlassian.com*. [Online]. Available: https://jira.atlassian.com/. [Accessed: May 27, 2024]].