PROJECT PROPOSAL

CSCI 5410 Serverless Data Processing

Submitted To: Dr. Saurabh Dey

Submitted By: Group 2

Ketul Patel - B00900957

Parul Raich - B00903203

Faiza Umatiya – B00899642

Contents

1. Abstract

The application “HalifaxFoodie” is a serverless Food delivery service system with restaurant owners and customers being target users. The application would be provided as “Backend as a Service” (BaaS) and each individual functionality of the application would be broken into small individual micro-services, deployed over multi-cloud services. The front-end of our application would be a react application and would be utilizing the various microservices to present the processed data and cater user requests. In addition to other the primary individual basic functionality, to make the system synchronized and fault-tolerant while communicating between multiple services we will be using messaging services.

2. Feature Specifications

2.1 User Management Module

2.1.1 Registration

The system will offer the registration process for multiple users at the same time. 3-Factor authentication system will be implemented to be to register user successfully database. Firstly, information of users such as email, password, name, phone number, etc. will be validated using **AWS Cognito** [4]. The second factor will be questions and answers, where questions will be fixed, and users need to provide an answer for 1 question. Then, the question and answer for that user will be stored in **DynamoDB** [3] database using **CloudFunction** [5]. Lastly, the User provided 4-character key and plain text will be stored in **DynamoDB** [3] database. Following, User will receive cipher text computed from **CloudFunction** [5] using Columnar Cipher that can be used during authentication.

Test Cases:

Table 1: Test case scenarios for Registration Module

Test Scenarios	Expected behavior
Name is in invalid format.	Alert user with proper message.
Email-id is in invalid format.	Alert user with proper message.
Phone number in invalid format.	Alert user with proper message.
Password is in invalid format.	Alert user with proper message.
Email-id already exists.	Alert user with proper message.
All basic details are valid.	Send user to next sign-up process.
Question not selected in question-answer.	Alert user with proper message.
Answer is not valid in question-answer.	Alert user with proper message.

Question-Answer is valid.	Store selected question and it' answer in **DynamoDB** [3] for user and send user to next sign-up process.
Character key is in invalid format.	Alert user with proper message.
Plaint text is in invalid format.	Alert user with proper message.
Character key and plain text are valid.	Send use cipher text and store key and plain text in **DynamoDB** [3].
Successfully Registered user.	Provide Success information with proper message and redirect user to login page.

2.1.2 Authentication

The login page for the serverless Food service web app is where the user ends up after registering. The user must initially register to use all of the system's features. Upon entering the app, the registered user authenticates themselves. There are multiple steps involved in the authentication.

The "**AWS Cognito"** will first be used to validate the user's login information [4]. The control shifts to the second layer of security if the first step is completed successfully. The "**GCP Firestore"** [12], and "**GCP Cloud Functions"**, which carry out question-answer checking, are now in charge in this tier [5].

After this layer is successfully passed, "**columnar cypher"** validation is carried out [14]. "**AWS Lamba"** [11] are used for this validation. The user will successfully log in once all three layers have been verified.

Test Cases:

Table 2: Test case scenarios for Authentication Module

Test case	Expected behavior
User enters invalid ID or password during first stage of authentication.	Return to login page and alert user to enter proper login credentials.
User enters proper login credentials during first stage of authentication.	Direct user to 2nd factor authentication page.
User enters incorrect answers during 2nd factor authentication.	Alert user with message stating "incorrect answer" asks to re-enter again.
User enters correct answer during 2nd factor authentication.	Direct user to 3rd factor authentication page.
User enters an incorrect key or plain text during 3rd factor authentication.	Alert user by stating "login failed".

2.2 Online Support Module

This module was created using the cloud services “**AWS Lex”** [13], and “**AWS Lambda”** Function [11].

The module's primary goal is to provide the user with virtual support by sending out bots to address all of their questions. All user types can make use of the func tionality.

The chatbot verifies a user's identification when they register before allowing them to continue their order online. These users can rate an order based on its order number and ask for navigational assistance. Visitors can inquire about directions from the chatbot. Restaurant owners should be allowed to provide information in their dynamic responses regarding the ingredients they use as well as their expenses. If the customer or user's question is regarding the complaint, the chatbot should start a new conversation using the chat module.

Implementation Details:

- After users register, the bot will assist them with navigation and solve any registration problems. The bot will authenticate the userID for logged-in users. After authentication, the bot will respond to the queries regarding order tracking and reviews. after “**AWS Lambda”** is activated [11], All of the data is retrieved using “**AWS DynamoDB”** [3]. Following receipt of all the information from the client, a second AWS Lambda function will be called, which will display all the tracking data. The names and costs of recipes can be added by restaurant owners. The name of the recipe will be entered and inserted into the database using an “**AWS Lambda”** function [11].
- If the problem or complaint is related to an order, the bot will end the current virtual help session and offer the "Chat with our associate" option. Once the user clicks the chat button, the chat module will start.

Test Cases:

Table 3: Test case scenarios for Online Support Module

Test Scenarios	Expected behavior
Customer tracking the online order	Bot should validate the UserID and provide the required information.
The owner tries to add the recipe and its prices.	Bot should guide the restaurant owner to add recipes, name and price with dynamic response.

2.3 Chat Module

With the help of this module, customers and the restaurant owner can interact in a live chat room. Customers use this feature to communicate issues with meal delivery to restaurant owners or customer service representatives. For instance, if customers receive their orders late, they can use a chatroom to speak with the restaurant manager who oversees that. There will be several logged-in sessions for this module, including client and restaurant operator sessions. A minimum of two logged-in accounts is needed to create a chat room; one must be a customer and the other must be a restaurant customer care professional. To make the dynamic chat chatrooms we would be using react applications coupled with "**Google Firebase"** at the backend [12].

Implementation Details:

- The logged-in customers will access the system in the first stage. Initially, they will select "Chat with us" from the menu if they have any problems with the current order. By pressing this button, the AWS Lambda function will start the chat room on the customer's end. The key-value pair containing all the information required for the setup of the chat room and the conversation will be kept in "**Google's Firebase"** [12] database.
- Another "**AWS Lambda"** function will be executed after receiving all the client-provided information, allowing the restaurant's customer service person to enter the chat session [11]. This module can only be used by authorised users. The necessary information will be retrieved via lambda functions from the order information pertaining to the customer and restaurant. Text messages will be used by them to communicate with one another.

Test Cases:

Table 4: Test case scenarios for Chat Module

Test Scenarios	Expected behavior
Customer/user tries to initiate a chat. (Customer and Customer service representative is online).	A chat should be initiated with the required restaurant's customer service representative.
Customer/user tries to end the chat.	Chat should be terminated from both ends (customer and restaurant's customer service representative).

2.4 Data Processing Module

In this feature, an "**AWS lambda function"** would be triggering the services required to extract and process the data to be used for quick retrieval [11]. This service will be limited to the restaurant owners wherein they would be allowed to extract "key" ingredients from the recipe. restaurant owners using lambda function, would upload recipe through front-end, which would be saved in the **"S3"** bucket [6]. Then, Amazon "**Comprehend**" will analyze the data from S3 to extract the title or the key ingredients from the recipe (**named entities**) [8], and store them in "**AWS DynamoDB"** along with name of the recipe for easy searching [3]. Machine learning module could use this data for further analyses. The uppercase entities would be considered as "named" entities.

Test Cases:

Table 5: Test case scenarios for Data Processing Module

Test Scenarios	Expected behavior
Named entities correctly identified	Shows the requested recipe and ingredients to restaurant owners.
Named entities exists but not identified	Named entity should have been identified and displayed.
Named entities falsely identified	Entities those were not part of named entities criteria were identified.
No named entity exists and not identified	No ingredients or recipe will be shown to restaurant owner.

2.5 Machine Learning Module

2.5.1 Similarity Score

In this feature, similarity between two recipes uploaded in "**AWS S3"** bucket will be identified [6]. "**GCP AutoML"** will be used to calculate similarity of recipes based on different measurements such as Euclidean distance [7]. The same recipes will be classified under the same group based on the threshold value. The value for threshold will be decided after checking model performance, available data, and features.

Test Cases:

Table 6: Test case scenarios for Machine Learning Module (Similarity Score)

Test Scenarios	Expected behavior
No similar recipe is found for recipe.	Maximum of 3 Random recipes of current Restaurant watched by user will be displayed.
One similar recipe is found.	Display only that recipe.
Two similar recipes found.	Display only those 2 similar recipes.
More than two similar recipes found.	Display only top 3 similar recipes.

2.5.2 Polarity

The main aim of the feature is to find the polarity of the feedback provided by customers**. "AWS Comprehend"** will be used to calculate the polarity [8], and will be visualized using **AWS "QuickSight"** [9]. Restaurant owners will be able to check polarity by clicking on the option of view visualize customer feedback polarity.

Test Cases:

Table 7: Test case scenarios or Machine Learning Module (Polarity)

Test Scenarios	Expected behavior
No customer feedback available.	Alert user with proper message.
Customer feedback polarity is positive.	Display green color happy face icon.
Customer feedback polarity is negative.	Display red color sad face icon.
Customer feedback polarity is neutral.	Display yellow color neutral face icon.

2.6 Visualization Module

2.6.1 Login Statistics

The feature will provide graphs/charts of traffic of Restaurant and Users, and login activity of user. It will be visualized using "**GCP Data Studio"** [10]. The graphs/charts will be displayed on the dashboard page of Restaurant admin.

Test Cases:

Table 8: Test case scenarios for Visualization Module (Login Statistics)

Test Scenarios	Expected behavior
No data is available.	No graphs will be displayed with proper message.
Data is available.	Related charts/graphs will be displayed.

2.6.2 Recipe

Recipe uploaded by will be highlighted using proper graphs. The examples of the graphs can be top rated recipes, most liked recipes, most profitable recipes, etc. "**GCP Data Studio"** will be used to show graphs on the dashboard page of Restaurant admin [10].

Test Cases:

Table 9: Test case scenarios for Visualization Module (Recipe)

Test Scenarios	Expected behavior
No data is available.	No graphs will be displayed with proper message.
Data is available.	Related charts/graphs will be displayed.

2.7 Message Passing Module

This module is "messaging-oriented middleware" that could be used as a queue to exchange data and enable communication "asynchronously and "reliably" between different micro-services [2]. Different applications could be publisher or subscriber or both to create or respond to an event to send or receive data. In this step, the publisher applications would publish data to a topic ("predefined" or "dynamic" based on the context of message passing) , and subscriber would subscribe to the topics and ingest data whenever needed [2].We will be using "**GCP pub/sub"** in our project to achieve the messaging passing and queuing [2].

Test Cases:

Table 10: Test case scenarios for Message Passing Module

Test Scenarios	Expected behavior
Publisher published the data to the topic and subscriber accessed the data when needed from the topic.	The publisher publishes the data and generates event to let the subscribers know the about update in the subscribed topic. The data is read correctly by the subscriber.
Publisher published the data but the subscriber could not access data when needed from topic.	The publisher publishes the data and generates event to let the subscribers know the about update in the subscribed topic. The data could not be read by the subscriber when it tried to access it.
Publisher could not publish the data	Publish could not publish data and subscriber not notified about the event.

3. Application Roadmap

3.1 Registration

The users of the application are of two types: customer and restaurant owners, Although the front-end page presented to both the users will be different, the data collection for authentication would be same.

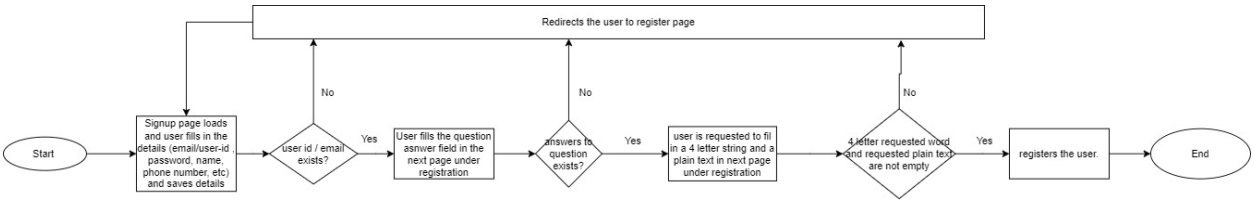

Figure 1: Application Module for Registration Module [1]

3.2 Authentication

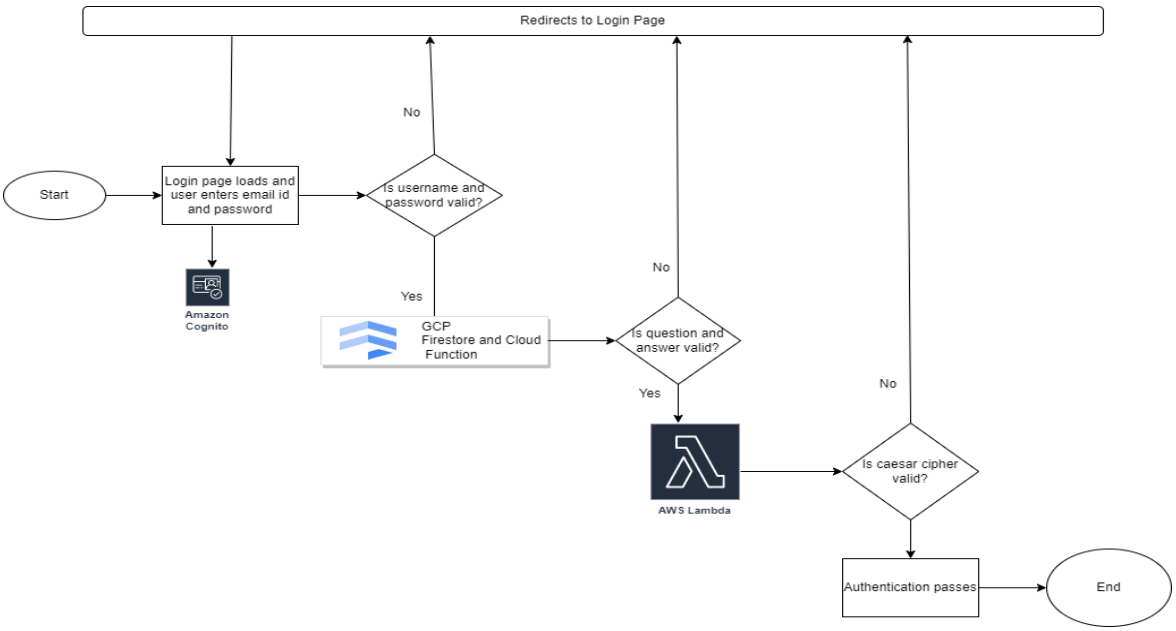

Figure 2: Application Module for Authentication Module [1]

3.3 Online Support

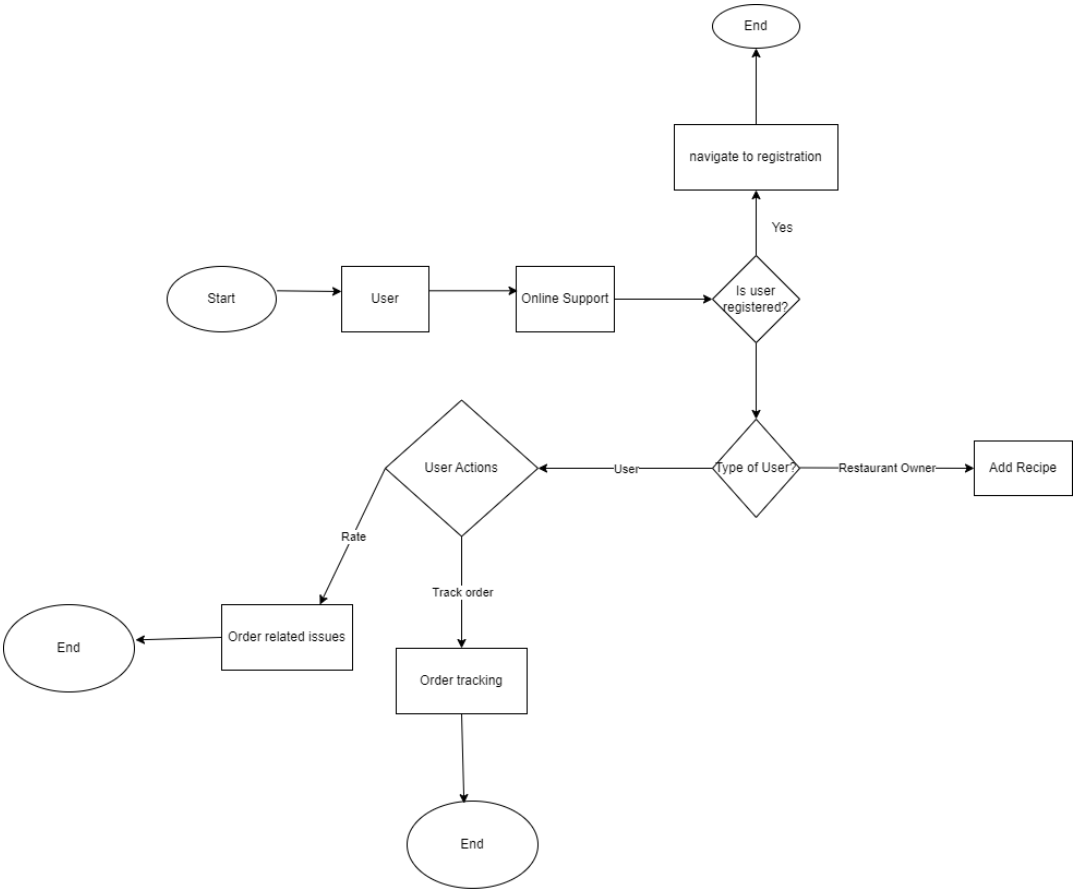

Figure 3: Application diagram for Online Support Module [1]

3.4 Chat Module

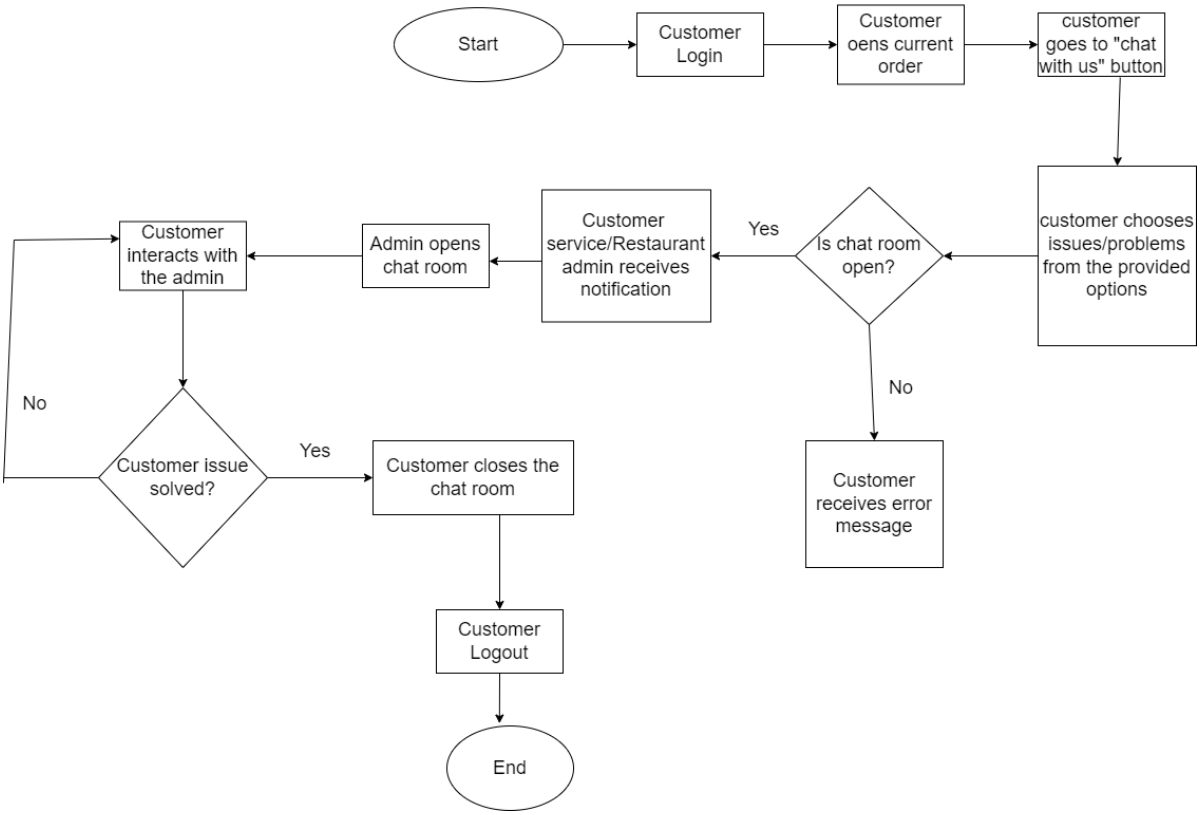

Figure 4: Application diagram for Chat Module [1]

3.5 Data Processing

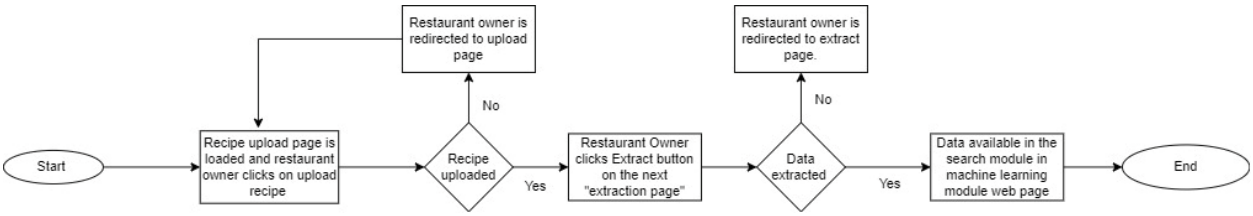

Figure 5: Application diagram for Data processing Module [1]

3.6 Machine learning

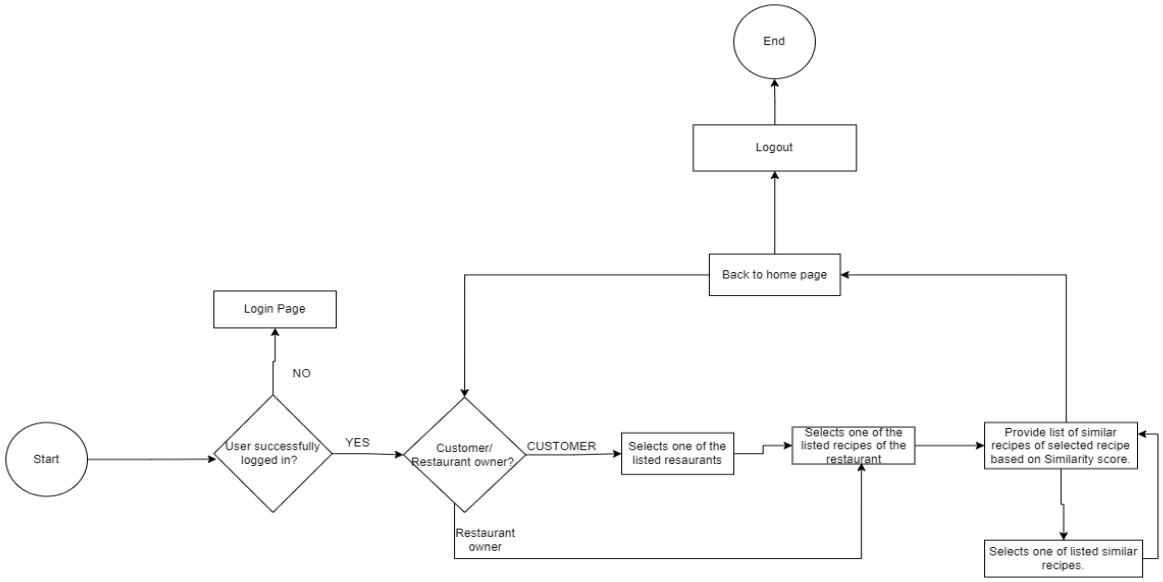

Figure 6: Application Diagram for Similarity Score [1]

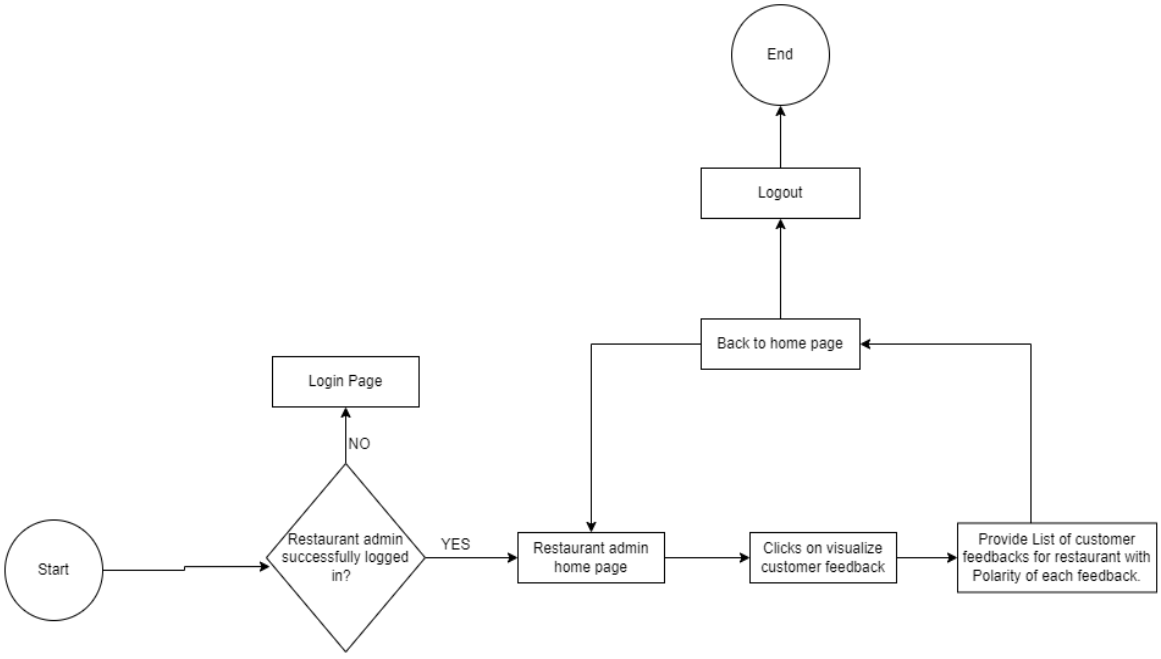

Figure 7: Application Diagram for Polarity of Customer feedback [1]

3.7 Visualization

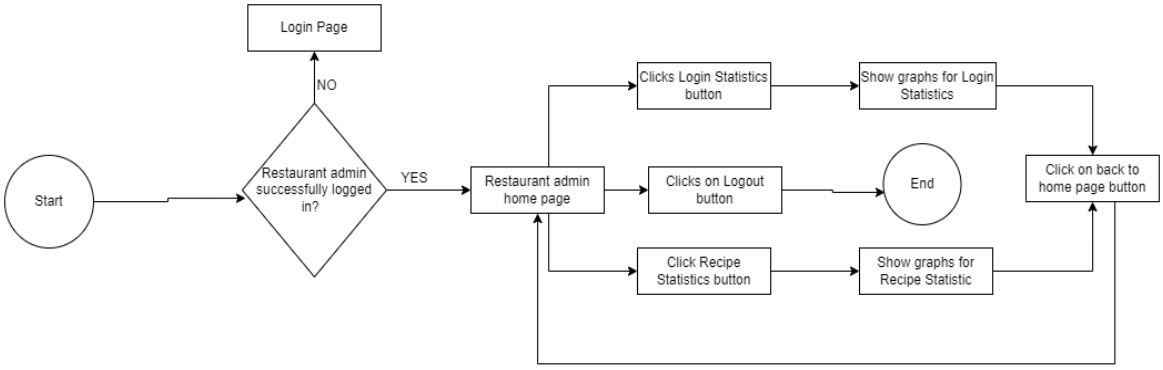

Figure 8: Application Diagram for Visualization [1]

3.8 Message Passing

No application diagram for this feature as it will not be accessible from the front end and is internal to the system.

4. Architecture

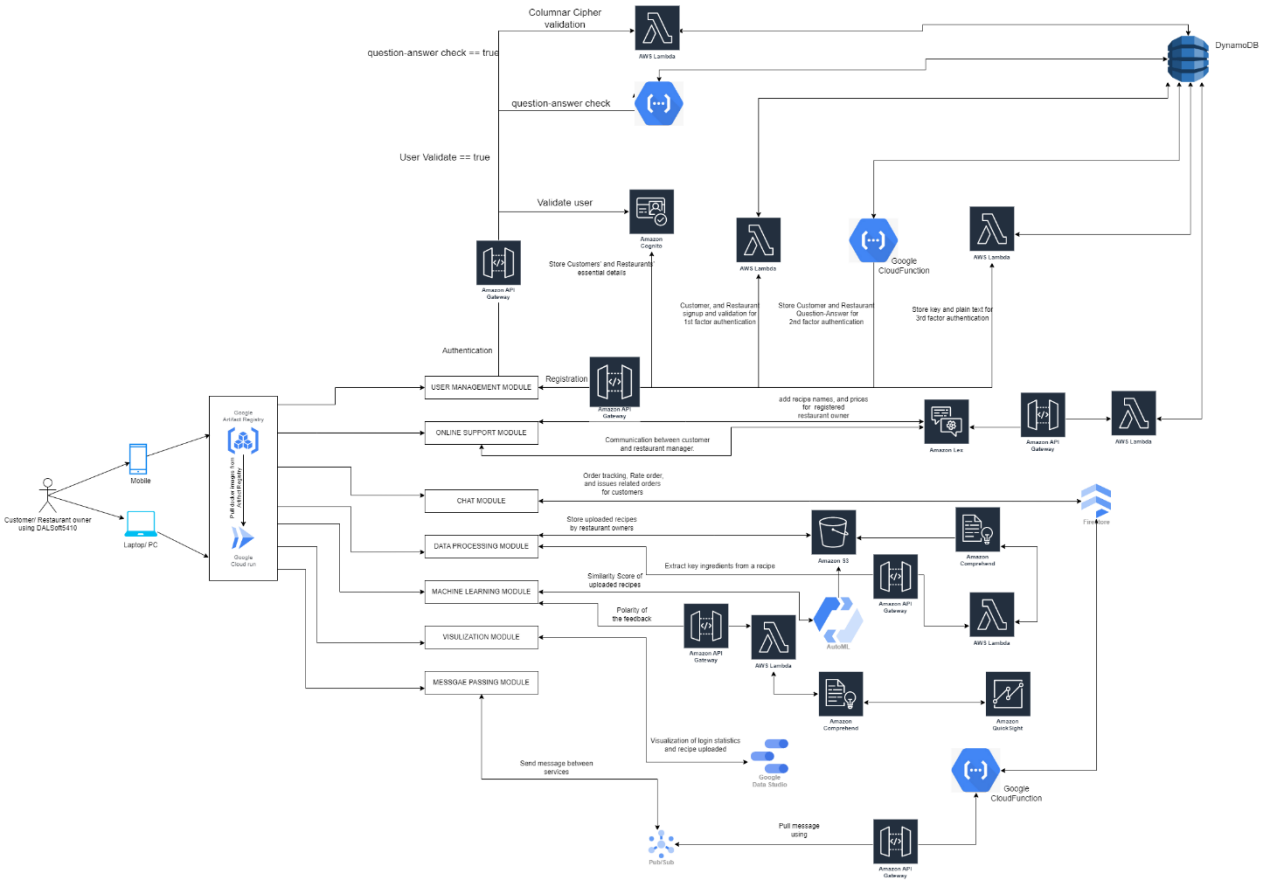

Figure 9: Cloud *Architecture for DALSoft5410 [1].*

5. Worksheet

Table 11: Worksheet

Module	Team member
User management – Registration-Customer	Ketul Patel
User management – Registration-Restaurant	Parul Raich
User management – Authentication	Faiza Umatiya
Online Support	Faiza Umatiya
Chat	Faiza Umatiya
Data processing	Parul Raich
Machine learning	Ketul Patel
Visualization	Ketul Patel
Message passing	Parul Raich
Documentation	All team members

6. Sprint Plan

Table 12: Sprint Plan

Sprint	Date	Task
1	Oct 01 - Oct 10	• Meet and greet for project planning • Requirement gathering and project analysis • Exploring cloud technologies and distributing modules amongst us.
2	Oct 12 - Oct 21	• Setting up the environment and Project configuration • Architecture planning and designing project structure.
3	Oct 23 - Nov 04	• User management Module implementation • Chat module implementation
4	Nov 05- Nov 14	• Building the front-end of the application • Preliminary stage of deployment and testing.
5	Nov 15- Nov 25	• Final testing and deployment • Start with the Report documentation and visualization.
6	Nov 26 - Dec 02	• Preparation of final Report and design documentation. • Preparation of final video presentation.

7. Meeting Log

7.1. Meeting 1: Offline

We had the first meeting after the class at the university face to face to get to know each other and discuss on the feature and distribute the features based on the interest.

7.2. Meeting 2: Online

We meet on 15th October to discuss about the project proposal and to discuss about the research on the technologies used in the project and choice of the final technologies out of AWS and GCP to be used in the applications.

Figure 10: Meeting log meeting 1.

7.3 Meeting 2: Online

We meet on 21st October to revise the documentation status and cohesively proofread and finalize the document.

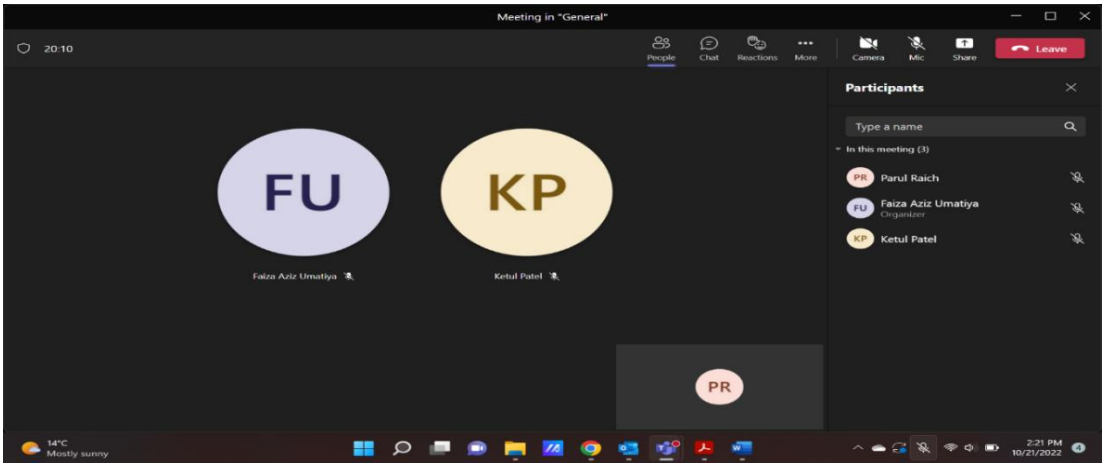

Figure 11: Meeting log meeting 2

8. References

[1] “Flowchart maker & online diagram software,” Diagrams.net. [Online]. Available: https://app.diagrams.net/. [Accessed: 24-Oct-2022].

[2] “What is Pub/Sub?,” Google Cloud. [Online]. Available: https://cloud.google.com/pubsub/docs/overview. [Accessed: 24-Oct-2022].

[3] Wikipedia contributors, “Amazon DynamoDB,” Wikipedia, The Free Encyclopedia, 09-Oct-2022. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Amazon_DynamoDB&oldid=1115015722.

[4] Amazon.com. [Online]. Available: https://aws.amazon.com/cognito/. [Accessed: 24-Oct-2022].

[5] “Cloud functions,” Google Cloud. [Online]. Available: https://cloud.google.com/functions. [Accessed: 24-Oct-2022].

[6] Amazon.com. [Online]. Available: https://aws.amazon.com/s3/. [Accessed: 24-Oct-2022].

[7] Y. Li, Z. Wang, B. Ding, and C. Zhang, “AutoML: A Perspective where Industry Meets Academy,” in Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, 2021.

[8] Amazon.com. [Online]. Available: https://aws.amazon.com/comprehend/. [Accessed: 24-Oct-2022].

[9] Amazon.com. [Online]. Available: https://aws.amazon.com/quicksight/. [Accessed: 24-Oct-2022].

[10] “Looker studio connect to data,” Google.com. [Online]. Available: https://datastudio.google.com/data. [Accessed: 24-Oct-2022].

[11] Amazon.com. [Online]. Available: https://aws.amazon.com/lambda/. [Accessed: 24-Oct-2022].

[12] “Firebase API reference,” Firebase. [Online]. Available: https://firebase.google.com/docs/reference/. [Accessed: 24-Oct-2022].

[13] *Amazon.com*. [Online]. Available: https://docs.aws.amazon.com/lexv2/latest/dg/what-is.html. [Accessed: 24-Oct-2022].

[14] Åhlén, “Columnar Transposition Cipher (online tool) | Boxentriq,” Columnar Transposition Cipher (online tool) | Boxentriq. [Online]. Available: https://www.boxentriq.com/code-breaking/columnar-transposition-cipher. [Accessed: Oct. 24, 2022]