DALHOUSIE UNIVERSITY

CSCI 5410– Serverless Data Processing

A Project Proposal On

HalifaxFoodie

Project Plan & Worksheet

Submitted by - Group 12

Group Members

Vaghasia Sagarkumar Pankajbhai – B00878629

Natarajan Aishwarya – B00882118

Sri Ramya Basam -- B00900307

Abstract

The application HalifaxFoodie is a serverless food delivery system that will be developed using multi-cloud deployment model and backend-as-a-service. The application supports restaurant owners to upload their menu with various dishes and recipes, and the customers to place an order and provide rating. Both restaurant owners and customers should sign up where restaurant owners are provided additional features like, uploading their top recipe for checking similarity score. The platform will provide the virtual assistant functionality that customers use to chat with restaurant's customer representatives. Restaurant owners can use Data Processing feature to extract key ingredients from a recipe.

Feature Specifications

User Management:

This module has two parts registration and authentication. 3-Factor authentication process will be enabled for both customers and restaurant owners. User signup and validation will be done using AWS Cognito. Amazon Cognito service provides APIs and infrastructure for key features in user management space such as authentication, authorization, and managing user repository with different operations. We will create user pool to log users registered and Details that are required in first factor authentication will be stored in DynamoDb using AWS Lambda. As second factor authentication we ask for question – answer that will be stored in GCP Firestore using Cloud function. For third factor authentication key and plain text will be accepted from the user. AWS lambda will be used to compute cipher text and the Cipher text will be returned to the user and the plain text will be stored in dynamodb.

Test Case	**Expected Behaviour**
User enters valid details while registering	Ask for 3-factor authentication detail
User enters invalid details (for example incorrect email format) while registering	Ask to re-enter details
User enters invalid userid while logging in	Ask to re-enter details. Do not proceed to 2nd factor auth
User enters invalid password while logging in	Ask to re-enter details. Do not proceed to 2nd factor auth
User enters valid details while logging in	Proceed to 2nd factor auth
User enters valid question and answer while logging in	Proceed to 3rd factor auth
User enters invalid question and answer while logging in	Ask to re-enter details. Do not proceed to 3rd factor auth
User do not enter all details while registering	Ask to enter all details
User enters invalid or empty key while registering	Ask to re-enter details

User enters wrong key during logging in	Ask to re-enter key
User enters valid key during logging in	User successfully gains access to the system

Cloud services used:
AWS cognito [1], DynamoDB, AWS lambda [2], Cloud function [3], GCP firestore.

Online Support:

This module helps to answer the most common questions of customer. The cloud services used to construct this module are AWS Lex and AWS Lambda Functions and data is retrieved from DynamoDB. Front end will be built using React.js.

For registered users, we will validate the userID, then a customer can track his/her orders by providing order number. The customer can also rate the order received based on order number. If the customer is not satisfied with his order chat module will be initiated to provide an option to chat with human. Registered restaurant owners will be able to add recipe names, prices etc. We use intents that is the action user wants to perform to develop this module. We use dynamic intents to fetch order details and store rating, recipe details into DynamoDB.

Test Case	**Expected Behaviour**
User enters valid userID	User can access features like order tracking, order rating
User enters invalid userID	Show error message “invalid userID”
User enters valid orderID to track order	Show status of the order
User enters invalid orderID to track order	Show error message that “orderID do not exits”

Cloud Services Used:
AWS Lex [4], AWS Lambda[2], DynamoDB

Chat Module:

This will be an instant messaging engine where Customer and Restaurant Manager should be able to chat in case of issues related to order or service. A separate chatroom is created for each problem using GCP Firebase. Each message is a document in firebase database and the chat module is created using React.js. This module requires multiple logged in sessions to represent user and restaurant owner. Firebase Authentication sessions manages the sessions across different chats. We will re-authenticate the user after the session is expired and the user will be asked to login again. We may use client-side logic to log out the user explicitly.

Test Case	**Expected Behaviour**
Chatroom is successfully created for customer and restaurant owner	User can chat with owner regarding order issue
Chatroom creation failed as session expired	Ask user to login again

Cloud Services Used: GCP Firebase [5]

Data Processing:

This module can be used to develop microservices that extract culinary recipes. Restaurant operators will utilise this service to identify the most important ingredients in a recipe. The application should allow restaurant owners to upload recipes (based on the word limits), which will be saved as text documents in S3 buckets or the Cloud Store. After submitting the recipes, the restaurant owner may use a button to extract the recipe's title or important ingredients (named entities), which will be saved as metadata in DynamoDb or Firestore along with the recipe's name for convenient search. A machine learning module might make use of this data. This module can be created using Lambda and Comprehend.

Test Case	Expected Behaviour
Recipe uploaded to the storage	Restaurant owners can access easily
Empty recipe loaded	Show error message
Recipe not uploaded	Owners must try again by shortening the recipe
Clicks the button on recipe title	Recipe should pop up

Cloud Services Used: DynamoDB, GCP firestore

Machine Learning:

- **Similarity Score**

In this module the similarity of the recipes that have been uploaded will be evaluated. To evaluate the similarity scores of the recipes, GCP AutoML [6] will be used. A machine learning model would be trained using the dataset of previously uploaded recipes until the target level of accuracy is attained. Future predictions will be made using this trained model, which will be deployed to an API endpoint.

- **Polarity**

This module conducts sentiment analysis of the feedbacks which had been provided by user for the restaurants. AWS Comprehend [7] is utilised as a service to measure polarity, and AWS QuickSight [8] is used to show the results. The owner of the business can see the polarity that was examined. An API call to Comprehend will be made upon clicking, allowing for the analysis of feedback.

Test Case	Expected Behaviour
Recipe having greater than 60% similarity score.	Similar recipes.
Recipe having less than 60% similarity score.	Not considered as similar recipes.

Visualization:

This module can be used to visualise login statistics, which show how often a user connects into the application, and traffic, such as the number of active orders, so that a user can see how long their orders will take to process. The restaurant owners should publish their recipes in a graphical format so that innovative recipes can be uploaded quickly and therefore they can keep track of any additional components that are added and roll the recipe back if necessary. With the use of the embed data studio dashboard feature of GCP Data Studio, these graphical charts are accessible.

Test Case	**Expected Behaviour**
Load login statistics	Users can view their login history
Number of orders	Determine the order number
Recipe charts	History of recipes

Message Passing:

This module will operate in the background when consumers can lodge a complaint. A chatroom service can be started by AWS Lex utilising Firebase so that customers can communicate with the restaurant representative about problems that occurred during their orders or transactions. Once the problem has been fixed, their chat history will be kept under communication history, which will be kept up as a cloud service, for future use. Here, the establishment of chatrooms and conversations would be seen as one GCP service, while the preservation of Communication history would be regarded as a separate service for future consideration.

Test Case	**Expected Behaviour**
Chatroom creation for customer and restaurant representative	User can chat with representative regarding order issue
Chatroom creation failed as session expired	Ask user to login again
Restaurant rep busy with other customers	Waiting time message pop

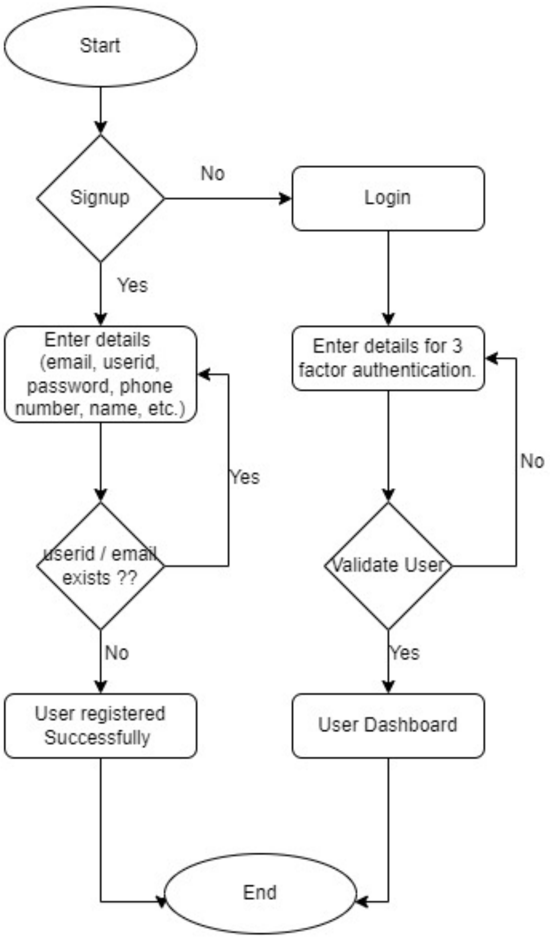
Cloud Services Used:

AWS Lex [4], GCP Firebase [5]

Application Roadmap:

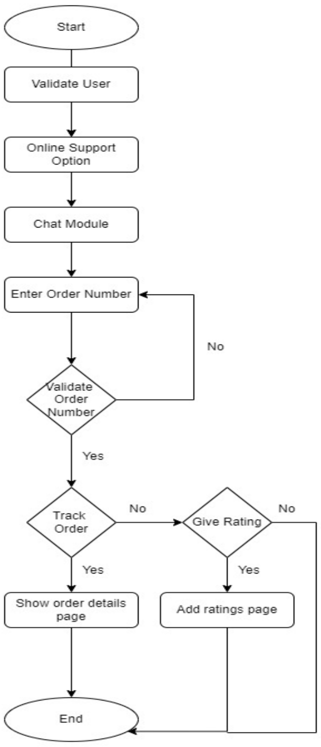
User Registration and Authentication

There will be mainly two kinds of users: customers and restaurant owners. The fromt end for the types of users will be different but the registration and authentications and process remains same for both.

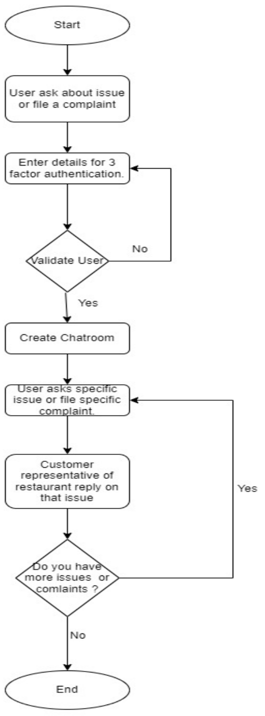


Start
Signup
No
Login
Yes
Enter details (email, userid, password, phone number, name, etc.)
Enter details for 3 factor authentication.
Yes
No
userid / email exists ??
Validate User
No
Yes
User registered Successfully
User Dashboard
End

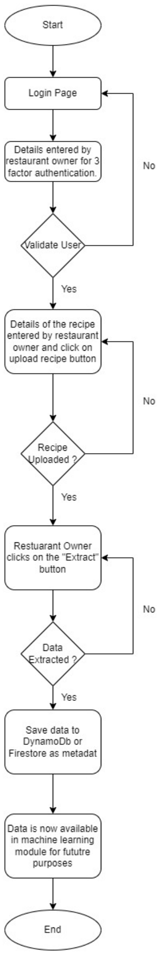
Online Support



Chat Module

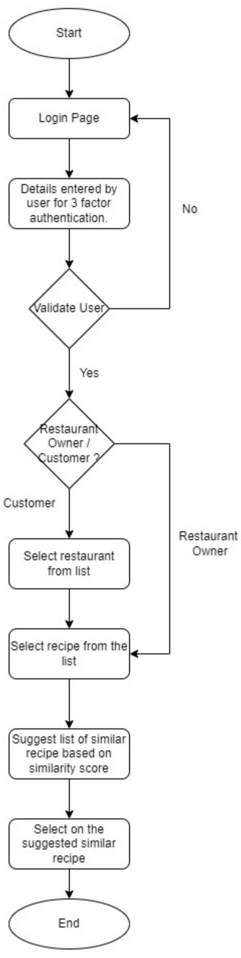

Start
User ask about issue or file a complaint
Enter details for 3 factor authentication.
Validate User
No
Yes
Create Chatroom
User asks specific issue or file specific complaint.
Customer representative of restaurant reply on that issue
Yes
Do you have more issues or comlaints ?
No
End

Data Processing

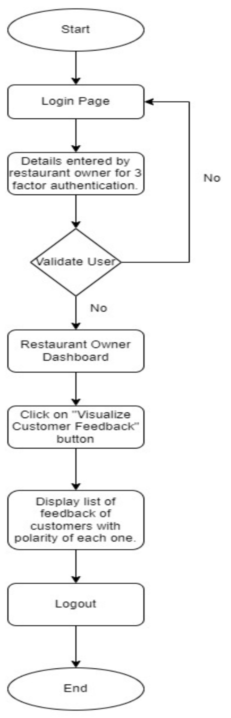

Start
Login Page
Details entered by restaurant owner for 3 factor authentication.
No
Validate User
Yes
Details of the recipe entered by restaurant owner and click on upload recipe button
No
Recipe Uploaded ?
Yes
Restuarant Owner clicks on the "Extract" button
No
Data Extracted ?
Yes
Save data to DynamoDb or Firestore as metadat
Data is now available in machine learning module for fututre purposes
End

Machine Learning

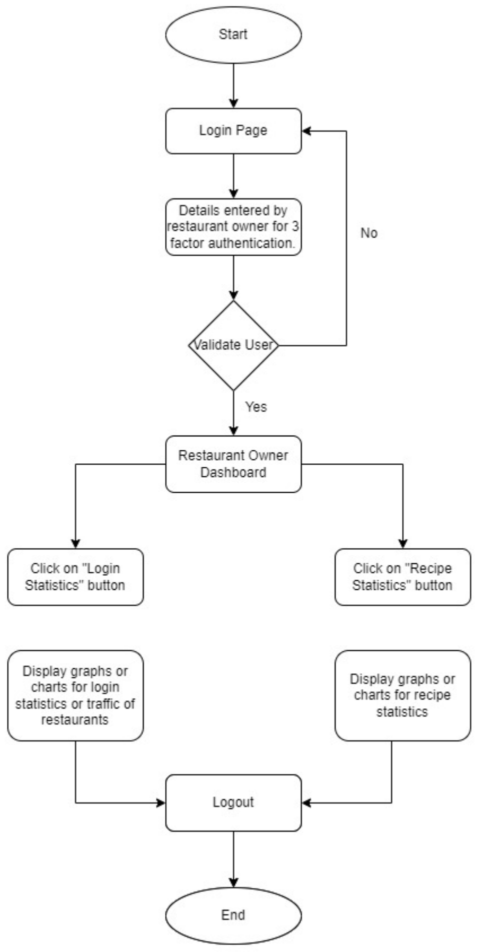
Similarity Score


Start
Login Page
Details entered by user for 3 factor authentication.
No
Validate User
Yes
Restaurant Owner / Customer ?
Customer
Restaurant Owner
Select restaurant from list
Select recipe from the list
Suggest list of similar recipe based on similarity score
Select on the suggested similar recipe
End

Polarity


Start
Login Page
Details entered by restaurant owner for 3 factor authentication.
No
Validate User
No
Restaurant Owner Dashboard
Click on "Visualize Customer Feedback" button
Display list of feedback of customers with polarity of each one.
Logout
End

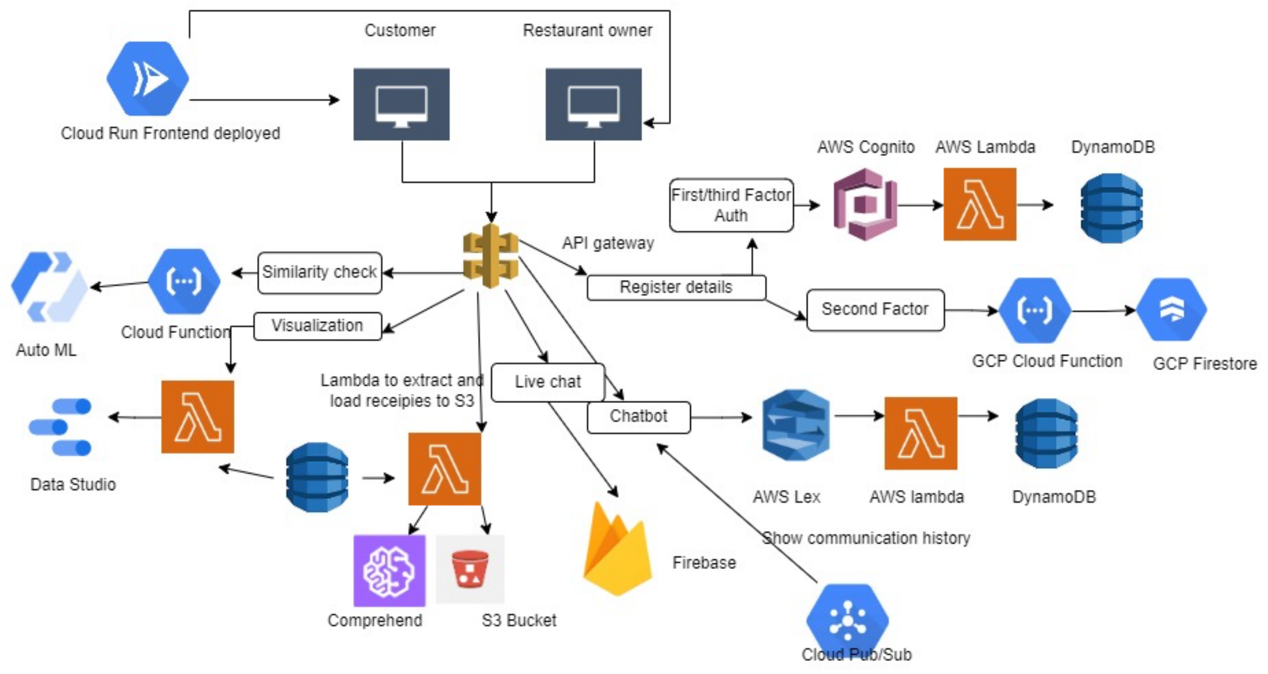
Visualization


Start
Login Page
Details entered by restaurant owner for 3 factor authentication.
No
Validate User
Yes
Restaurant Owner Dashboard
Click on "Login Statistics" button
Click on "Recipe Statistics" button
Display graphs or charts for login statistics or traffic of restaurants
Display graphs or charts for recipe statistics
Logout
End

Message Passing

There is no application roadmap diagram for message passing as it will be working in the backend where no user will interact with the services run by application.

Architecture Diagram:



Worksheet:

Module	Team Member
User management – Customer registration, Restaurant Registration, Authentication	**Ramya**
Online Support	**Ramya**
Chat	**Sagar**
Data Processing	**Aishwarya**
Machine Learning	**Aishwarya**
Visualization	**Aishwarya**
Message Passing	**Sagar**
Project Proposal	All team members

Sprint Plan:

Date	Weekly Plan
Oct 03 – Oct 14	Explore GCP Firebase, AWS Lambda, Cognito,Dynamo DB, AWS Amplify,Cloud functions
Oct 15 – Oct 22	Work on the project proposal and create worksheet plan for the same.

Oct 23 – Nov 4	Our team based on rigorous discussion would create basic project structure and start working on the assigned modules. Ramya would start working on user management module.
Nov 5 – Nov 18	Sagar to complete the Online support module and start with chat module. Ramya to start working with Authentication module. Aishwarya would start working Machine Learning, Visualization and Data processing module. Ramya, Aishwarya and Sagar to work on Message passing module.
Nov 18 – Nov 25	Start testing all the modules and rectify the issues.
Nov 26 – Dec 2	Complete all the modules and their testcases and push it to gitlab and work on the documentation part.

Meeting logs:

Date: 26 Sept
Agenda: Group Introduction and project meeting to discuss project specification V01. Met in-person.

Date: 8 Oct
Agenda: Team meeting to understand the features and transfer knowledges

Date: 15 Oct
Agenda: Meeting with Bharat to discuss on project specification V02.

Date: 21 Oct
Agenda: Team meeting to distribute tasks and prepare initial work sheet.

Date: 22 Oct
Agenda: Team meeting to review final version of the report.

References:

[1] *Amazon.com*. [Online]. Available: https://aws.amazon.com/cognito/. [Accessed: 21-Oct-2022].

[2] *Amazon.com*. [Online]. Available: https://aws.amazon.com/lambda/. [Accessed: 21-Oct-2022].

[3] "Cloud functions," *Google Cloud*. [Online]. Available: https://cloud.google.com/functions. [Accessed: 21-Oct-2022].

[4] *Amazon.com*. [Online]. Available: https://aws.amazon.com/lex/. [Accessed: 23-Oct-2022].

[5] "Firebase & Google cloud," *Firebase*. [Online]. Available: https://firebase.google.com/firebase-and-gcp. [Accessed: 23-Oct-2022].

[6] "AutoML tables documentation," Google Cloud. [Online]. Available: https://cloud.google.com/automl-tables/docs. [Accessed: 23-Oct-2022].

[7] Amazon.com. [Online]. Available: https://docs.aws.amazon.com/comprehend/latest/dg/what-is.html. [Accessed: 23-Oct-2022].

[8] Amazon.com. [Online]. Available: https://docs.aws.amazon.com/managedservices/latest/userguide/quicksight.html. [Accessed: 23-Oct-2022].