



**Faculty of Computer Science**

**Dalhousie University**

**CSCI 5410 – Serverless Data Processing**

**FINAL REPORT**

**A Multi-Cloud based Serverless Food Delivery System**

Harsh Samirbhai Bhatt	<b>B00877053</b>
Vishnu Sumanth Dampetla	<b>B00866237</b>
Venkata Kanakayya Prashant Vadlamani	<b>B00883901</b>
Johanan Abhishek Prabhurai	<b>B00869532</b>

## TABLE OF CONTENTS

PROJECT REQUIREMENTS .....	5
KNOWLEDGE TRANSFER AND MEETING DETAILS .....	5
MEETING 1: 5-25-2021 .....	5
MEETING 2: 05-29-2021 .....	6
MEETING 3: 06-05-2021 .....	6
MEETING 4: 06-08-2021 .....	6
MEETING 5: 06-13-2021 .....	7
MEETING 6: 06-16-2021 .....	7
MEETING 7: 06-23-2021 .....	7
MEETING 8: 06-30-2021 .....	8
MEETING 9: 07-03-2021 .....	8
MEETING 10: 07-04-2021 .....	8
MEETING 11: 07-30-2021 .....	9
PROJECT TIMELINE .....	9
ARCHITECTURE DIAGRAM .....	10
SERVICES OVERVIEW AND DETAIL .....	11
1. User Management Module .....	11
2. Authentication Module.....	12
3. Online Support Module.....	13
4. Messaging service .....	14
5. Data Processing .....	14
6. Machine Learning .....	15
7. Web Application Building and Hosting .....	16
8. Testing.....	17
9. Visualization .....	22
10. Application – Service Interaction .....	23
API Gateway (AWS), Cloud Functions (GCP) .....	23
USE CASES.....	23
DATABASE DESIGN.....	25
TEAM COORDINATION.....	26
PROGRAMMING FRAMEWORK .....	26
Front-end.....	26

Back-end .....26

Database .....26

CHALLENGES .....27

    Credits .....27

    Security .....27

    Integration .....27

USER INTERFACE .....28

REFERENCES.....42

## TABLE OF FIGURES

Figure 1: Architecture Diagram of Project.....	10
Figure 2: User Management Module .....	11
Figure 3: Authentication Module .....	12
Figure 4: Chat Module .....	13
Figure 5: Data Processing Module.....	14
Figure 6 Machine Learning Module .....	15
Figure 7: Web Application Building and hosting Module.....	16
Figure 8: Ratings Visualization generated using Data Studio .....	22
Figure 9: Database design of project.....	25
Figure 10: Login Screen.....	28
Figure 11: Signup Screen .....	28
Figure 12: Restaurant Signup Lock Screen.....	29
Figure 13: Restaurant Signup Screen.....	29
Figure 14: Reset Password Screen .....	30
Figure 15: Security Question Screen .....	30
Figure 16: User Dashboard Screen .....	31
Figure 17: Profile Screen .....	31
Figure 18: AWS Lex Bot Live Chat .....	32
Figure 19: Pub/Sub Chat Screen .....	32
Figure 20: List of Restaurants Screen .....	33
Figure 21: Restaurant Screen .....	33
Figure 22: Order History Screen.....	34
Figure 23: Review Order Screen.....	34
Figure 24: Order Detail Screen .....	35
Figure 25: Track Order Screen.....	35
Figure 26: My Cart.....	36
Figure 27: Coupon Code .....	36
Figure 28: Word Cloud of frequently ordered dishes .....	37
Figure 29: Similar Dishes Suggestions .....	37
Figure 30: Restaurant Dashboard Screen.....	38
Figure 31: Featured Food Item.....	38
Figure 32: Delete Food Item .....	39
Figure 33: Add new dish to Menu.....	39
Figure 34: Order list for Restaurant .....	40
Figure 35: Feature to change order status .....	40
Figure 36: Order can be made from one restaurant at a time.....	41

## PROJECT REQUIREMENTS

Dalsoft5410 is planning to build serverless Food Delivery System using multi-cloud and Backend-as-a-service (BaaS). The main aim of this application is to a virtual assistance online to all the registered customers to get their questions answered. The application also provides the following features:

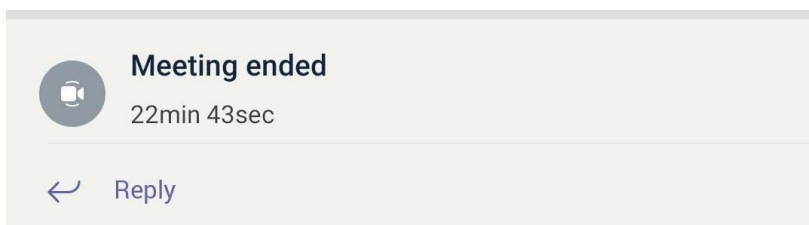
- User Authentication
- Customer Feedback
- Rating the restaurant and food
- Food Recommendation for users
- Chat Functionality between a customer and service representative
- Offering discount coupons to the customers

For the development of the application, we have decided to build the application on two cloud platforms – AWS and GCP. We have received accounts for both GCP and AWS which could help us in building and testing the application. The development approach for each of the features is explained in the Feasibility Study including the services and the technologies used.

## KNOWLEDGE TRANSFER AND MEETING DETAILS

We have come up with all the approaches possible for the development of the application. We discussed about our strengths in the programming language and concluded an approach to develop the application. We have been meeting frequently to discuss about the idea and discussed about the services required to build each module for the application. The meeting logs of the discussion is shown below.

### MEETING 1: 5-25-2021



### Discussed agendas

- It was an introductory session.
- We discussed programming languages and cloud services overview that can be implemented efficiently.

## MEETING 2: 05-29-2021

← Reply



Meeting ended

23min 32sec

← Reply

## Discussed agendas

- In the second meeting, we figured out the services needed to accomplish each module.
- We came up with few services and explored them in detail.

## MEETING 3: 06-05-2021



Meeting ended

1 hr 23min 25sec

← Reply



## Discussed agendas

- In the third meeting, we discussed the feasibility of each module by the cloud service we had chosen.

## MEETING 4: 06-08-2021

HB

Harsh Samirbhai Bhatt 15:45

Harsh

User Management  
Authentication  
Web Application & Hosting

Vishnu

Online Support Module  
Live chat  
Machine Learning

Prashant

Other Essential  
Project Overview  
Knowledge Transfer

Abhishek

Data Processing  
Gantt Chart

See less

← Reply

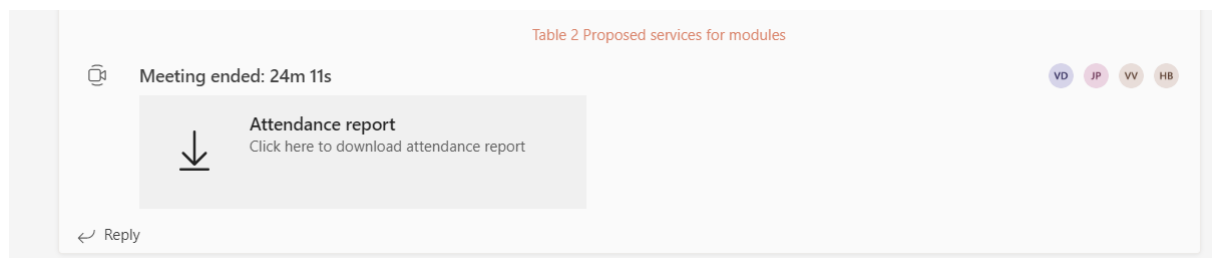
Meeting ended: 35m 46s

14

## Discussed agendas

- In the fourth meeting, we worked on dividing the work for documentation.
- We share knowledge about the cloud services we know with each other.

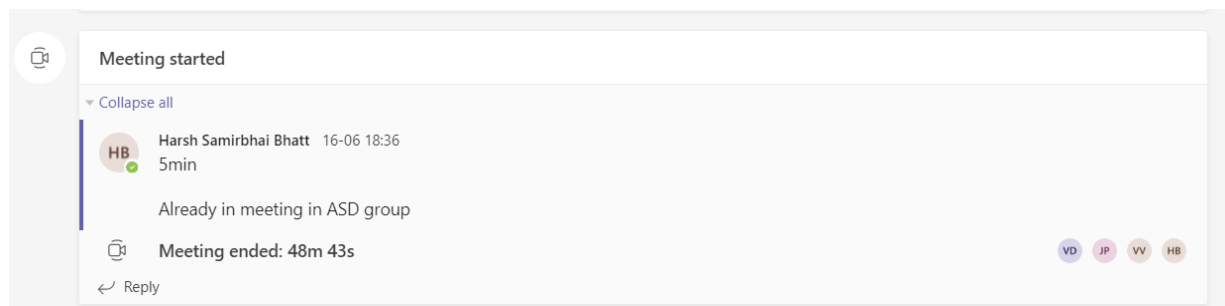
## MEETING 5: 06-13-2021



## Discussed agendas

- We talked about how we're going to execute the project modules and what problems we might face during this meeting.

## MEETING 6: 06-16-2021



## Discussed agendas

- In his meeting, we tracked the work progress and discuss the next tasks.

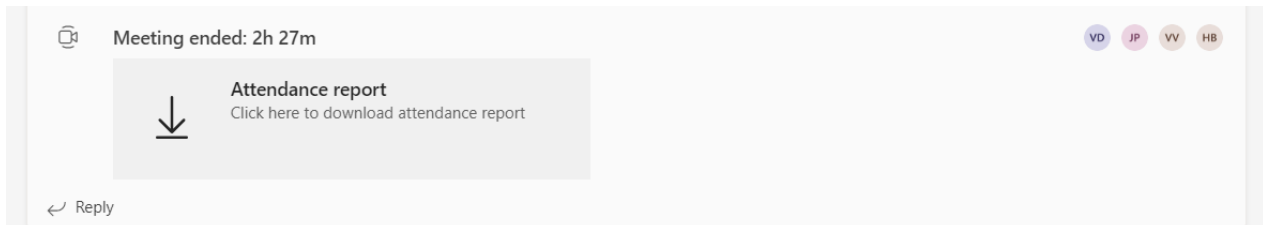
## MEETING 7: 06-23-2021



## Discussed agendas

- We worked on the design document task and checked the viability of cloud services that we had decided to work on at this conference.

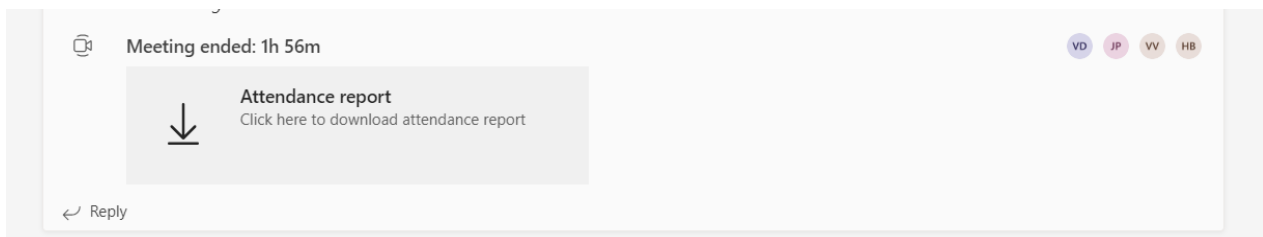
### MEETING 8: 06-30-2021



## Discussed agendas

- We kept track of the design document's progress and discussed what was remained and what could be improved.

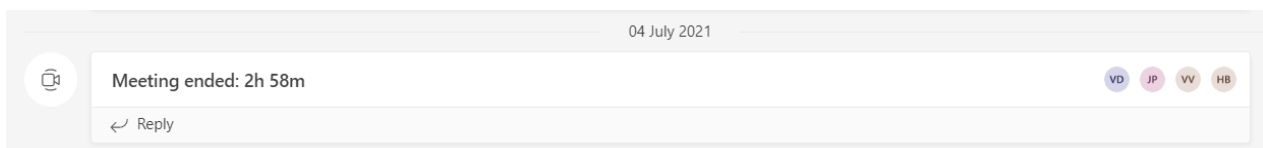
### MEETING 9: 07-03-2021



## Discussed agendas

- We prepared, checked, and improved the design document quality at this meeting before submitting it to the Brightspace portal.

### MEETING 10: 07-04-2021

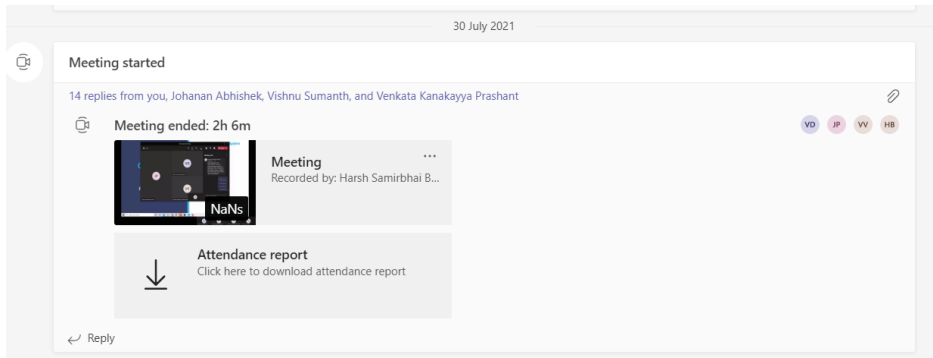


## Discussed agendas

- Before the Q&A session, we tested our project to confirm that it was working properly and fixed the possible bugs.



## MEETING 11: 07-30-2021



### Discussed agendas

- We recorded our project demonstration video and submitted it to the Brightspace portal during this conference.

## PROJECT TIMELINE

*Table 1: Timeline for project deliverables*

No.	Deliverables	Timeline
1	Feasibility Analysis	14 <sup>th</sup> June 2021
	Knowledge Transfer	
	Requirements Gathering	
2	Exploring the Services SDK	18 <sup>th</sup> June 2021
	Project Plan & Database Design	
3	Implementation of Modules	15 <sup>th</sup> July 2021
4	Integration of Modules	26 <sup>th</sup> July 2021
5	Regression Testing	28 <sup>th</sup> July 2021
6	Project Deployment	30 <sup>th</sup> July 2021
	Project Demo	

	Project Final Report	
--	----------------------	--

ARCHITECTURE DIAGRAM

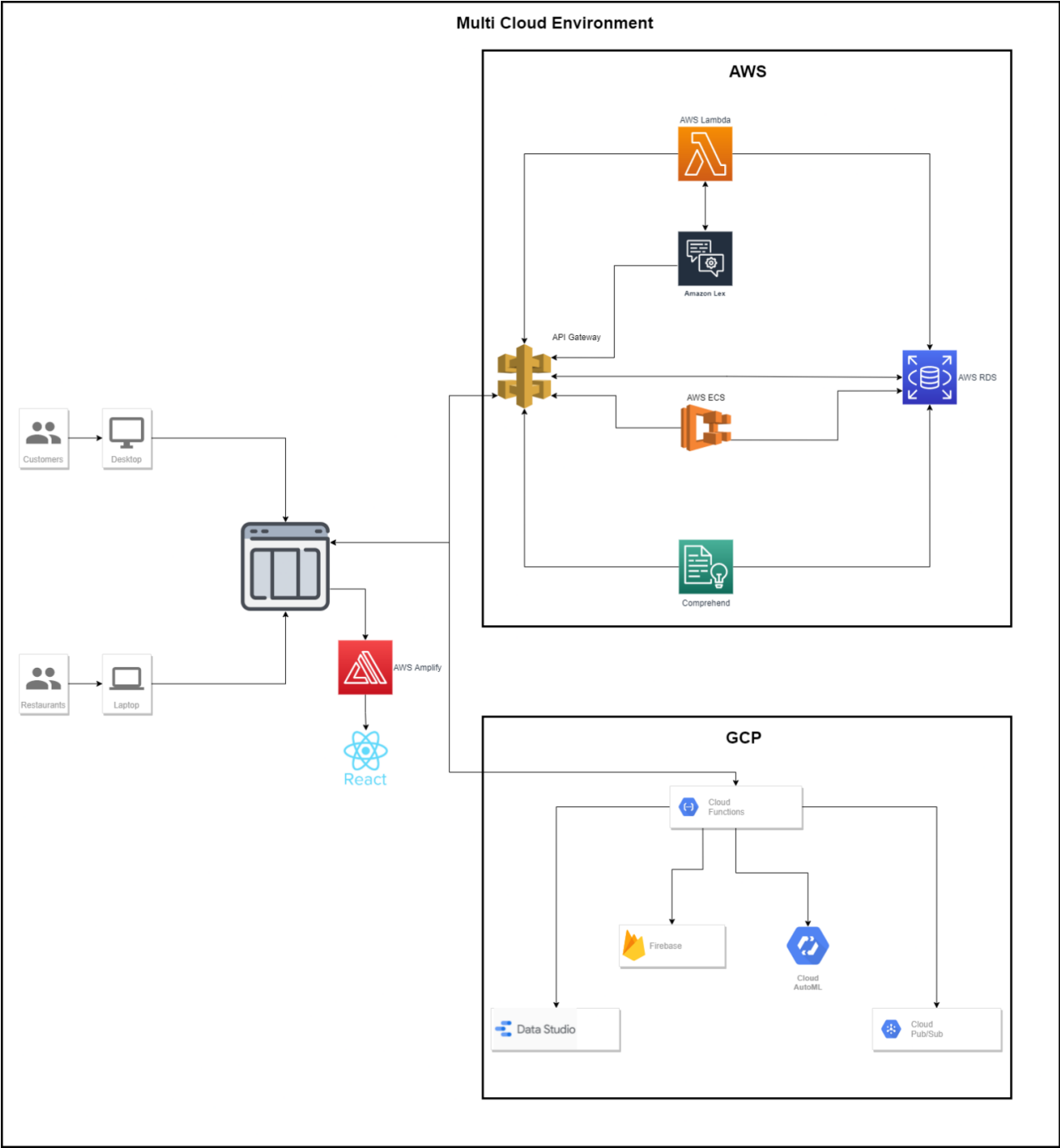


Figure 1: Architecture Diagram of Project

Figure 1 depicts the tentative multi-cloud architecture diagram of the application to be developed using various cloud services offered by GCP and AWS.

## SERVICES OVERVIEW AND DETAIL

Table 1: Services for modules

Modules	Service	Cloud Service	Service Provider	Framework
User Management	Signin & Signup	Identity Platform	GCP	-
	Managing and storing User details	AWS RDS	AWS	-
Authentication	User authentication	Identity Platform	GCP	-
	Security Question	Lambda	AWS	-
Online Support	Chat Bots	Amazon Lex	AWS	-
Live Chat	Chat between Customer and Restaurant Manager	Pub/Sub	GCP	-
Data Processing	Extract named entities from food ratings	Amazon QuickSight	AWS	-
	Build and Manage Container	Amplify, API Gateway	AWS	-
Machine Learning	Identify the similarity of the recipe and add appropriate tags	AutoML	GCP	-
Web Application Building and hosting	Frontend	-	-	React.js
	Backend	-	-	Node.js/Python/Cloud Services
	Hosting	Amplify	AWS	-
Others	Testing	Lambda	AWS	-
	Report Generation	Amazon QuickSight	AWS	-
	Visualization Module	Data Studio	GCP	-

Table 1 gives an idea about the services offered by the respective cloud service providers (GCP, AWS), to be used for each module in the application.

### 1. User Management Module

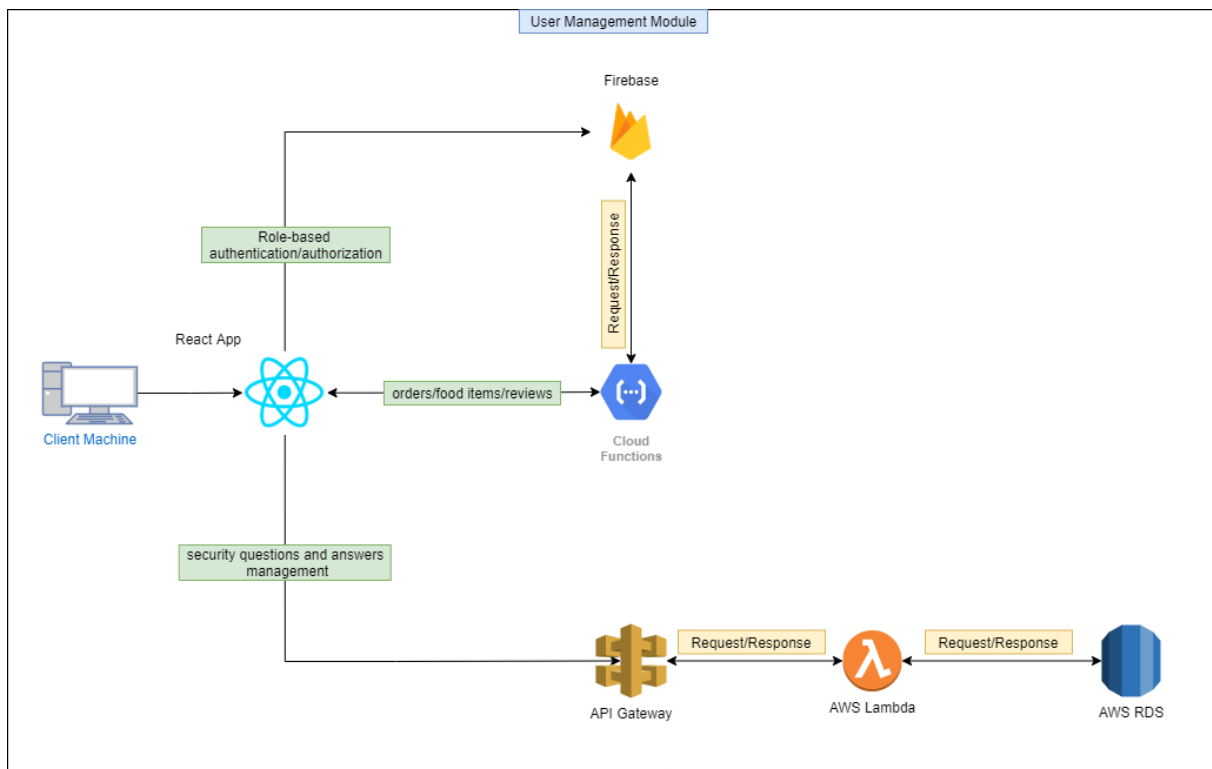


Figure 2: User Management Module

The user administration module is shown in action in the diagram above. In our system, we have two roles. The first is **User**, while the second is **Admin**.

The user is the one who orders the food, reviews it, can speak with Lex Bot or a restaurant manager/employee in real-time, and tracks the order.

The admin is the restaurant's owner or manager, and he or she oversees all administrative tasks such as adding the restaurant's menu, handling & managing new orders, creating food items to be featured in the popular dishes list, chatting with a client in case of an issue, and so on.

Using API Gateway [1] and Cloud Functions [2], we were able to integrate AWS and GCP services. We've set up a firebase project with cloud functions to enable a multi-service integration environment, and the firebase Identity Platform handles all authentication and authorization-related services. All user-related information is transmitted to AWS RDS for storage after the user is authenticated using Firebase. All order-related information, such as placing an order, providing feedback and ratings, and adding menu items, is handled by cloud functions, while storage is handled by AWS RDS.

## 2. Authentication Module

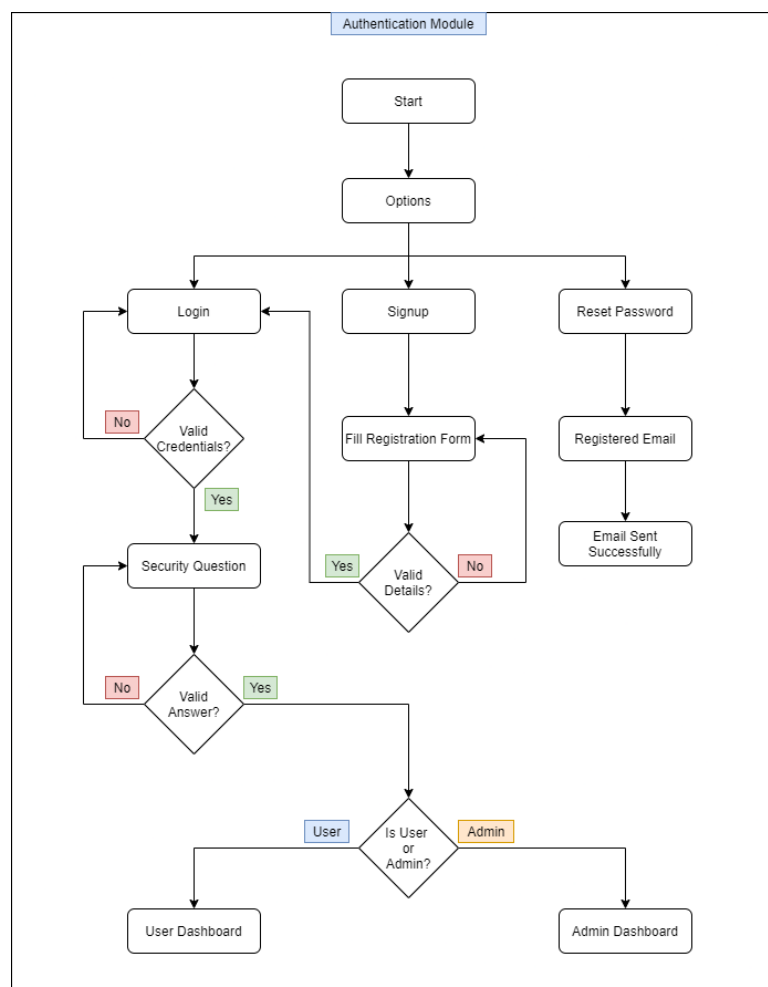


Figure 3: Authentication Module

As mentioned above, we have two roles in our online food ordering system. i.e., User and Admin.

Users will be able to choose from a variety of options, as shown in the flowchart. If a user attempts to log into the system using improper credentials, an appropriate notification will be displayed. If the credentials are legitimate, the flow continues, and the user is asked to answer the security question they chose during registration.

The lambda function is used to retrieve the answer from AWS RDS. This necessitates the use of the user id to uniquely identify the answer to the query in question. If the response is valid, the system will extract the user role, and the associated dashboard will be loaded based on the role, i.e., for the user role, the user dashboard will be loaded, and for the admin role, the admin dashboard will be loaded. Because it is role-based authentication and authorization, there will be a limited number of options available to users that are not available to administrators and vice versa.

### 3. Online Support Module

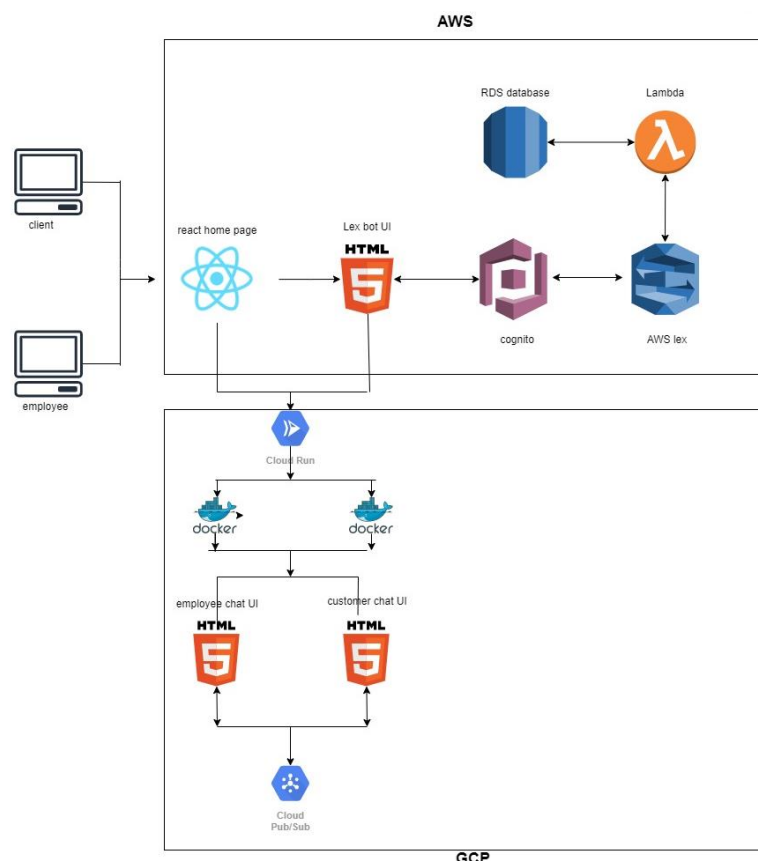


Figure 4: Chat Module

For a bot, we have used the AWS Lex service provided by AWS. To make the bot more dynamic. We connected it to the lambda function where switch cases for each intent are present.

The lambda will fetch information from the AWS RDS database and returns the response object to the Lex. There are two intents that are dynamic. We used Cognito and generated an identity pool id that will allow clients to access AWS services within the rules that were given. There is a escalate button in both UI, which will redirect the user to messaging service. This button is only present on the customer's home page. There is no bot available on the restaurant home page. The bot will fetch the status of the order, shows all the available restaurants dynamically, and helps the user navigate along with the homepage.

#### 4. Messaging service

We have used GCP's pub/sub service for the chat module. The chat module consists of two web pages that work individually. The employee chat module has its topic and is subscribed to the customer's topic and the customer's chat module has its topic and is subscribed to the hotel's topic. These two webpages are containerized and deployed in Cloud run. We have reduced the message retention period to 10 minutes so that the messages are only stored for 10 minutes. The way the user is directed to these web pages is also different, it depends on the type of user i.e., customer or employee. If the user is a customer, he/she/they must press the escalate button in BOT UI to chat with the employee or manager. If the user is an employee, he/she/they must press the chat button on the home page to chat with the customer.

#### 5. Data Processing

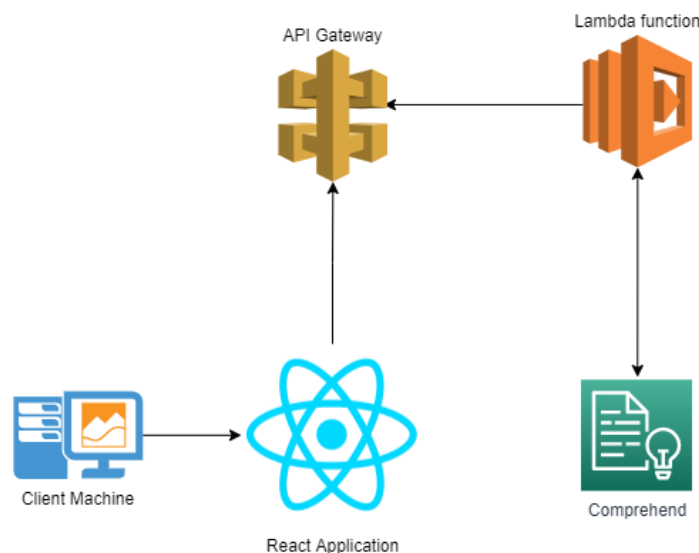


Figure 5: Data Processing Module

For data processing, we have used the services Amazon Comprehend, MySQL instance of Amazon RDS, Amazon API Gateway, and Amazon S3. We have trained a list of food items in Amazon Comprehend by creating a custom entity recogniser [16]. For training the data we have stored the list of food items in a csv file which is stored in Amazon S3 bucket. Amazon comprehend access the csv file in S3 bucket to train the data. Once the custom entity recogniser is trained, we have retrieved the food items from the order review table which is stored in MySQL instance of Amazon RDS [2]. This food items are sent as an input to the custom entity recogniser which was trained previously to retrieve the recognised food items. These food items are retrieved are stored in a json format which would be used as an input to create a word cloud. We have created a lambda function in python to return the food items [3]. Now, we have created an API using API gateway to integrate the data to our web application [12]. For the creation of word cloud, we have used WordCloud API by rapid API [17]. The food items retrieved are taken as an input to this word cloud API and a word cloud of the food items is generated.

## 6. Machine Learning

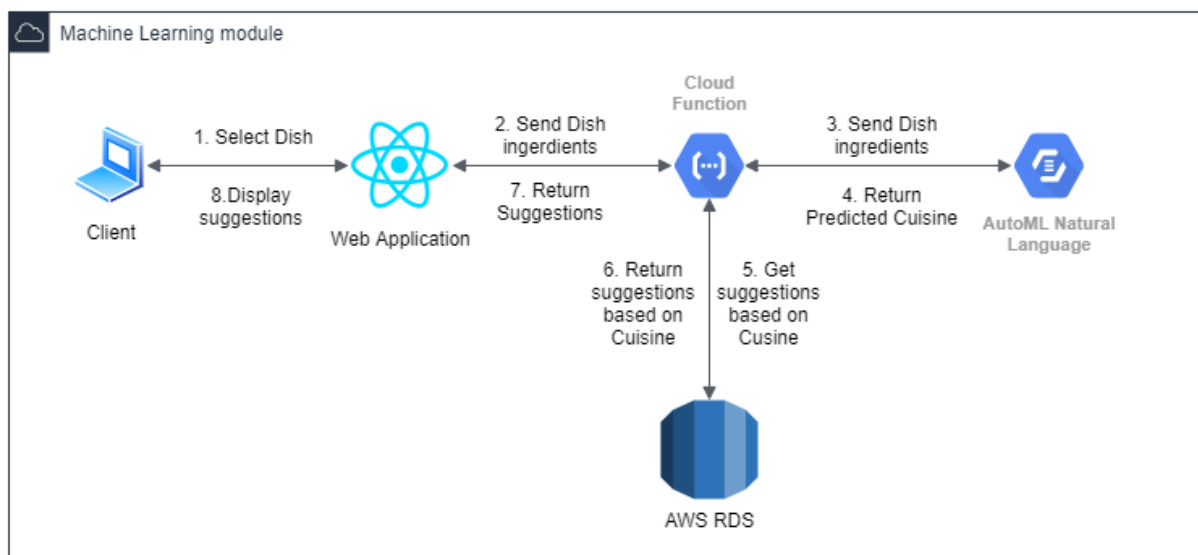


Figure 6 Machine Learning Module

Figure 6 shows the components and the control flow of the Machine learning module. The website uses AutoML Natural Language [3] in this module. The user is given an option to find similar dishes to the dish they select. After the dish is selected and show similar option is selected. The website sends a GET API call to a cloud function [2] with the ingredients of the selected dish in the Request message body. These ingredients are redirected to the AutoML trained model (currently trained to recognise French and Indian cuisine) to predict the type of cuisine selected. Once the cuisine is predicted, the predicted result is sent back to the cloud function that invoked AutoML. The predicted cuisine is then looked up into the AWS RDS MYSQL [4] instance to find similar dishes based on the predicted cuisine. Once the results are

retrieved from the RDS instance, the cloud functions return these values as a response to the API call embedded within the response body. This is then displayed to the User in the website.

## 7. Web Application Building and Hosting

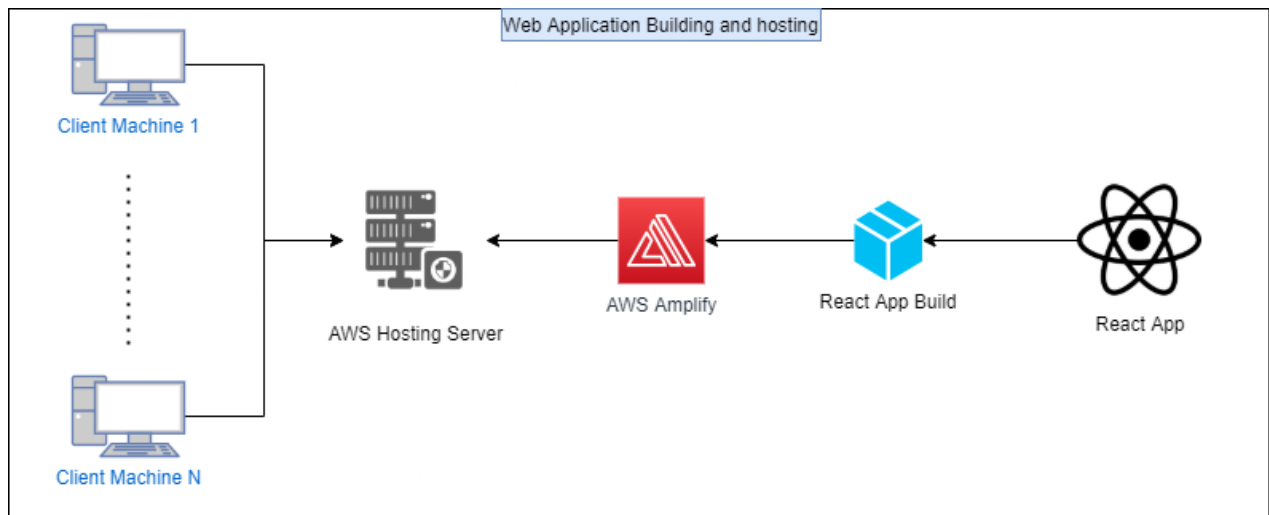


Figure 7: Web Application Building and hosting Module

The application's front-end is developed using React.js and the backend using Node.js with Express.js. Using Node.js the entire user requests are forwarded to the respective services using API calls.

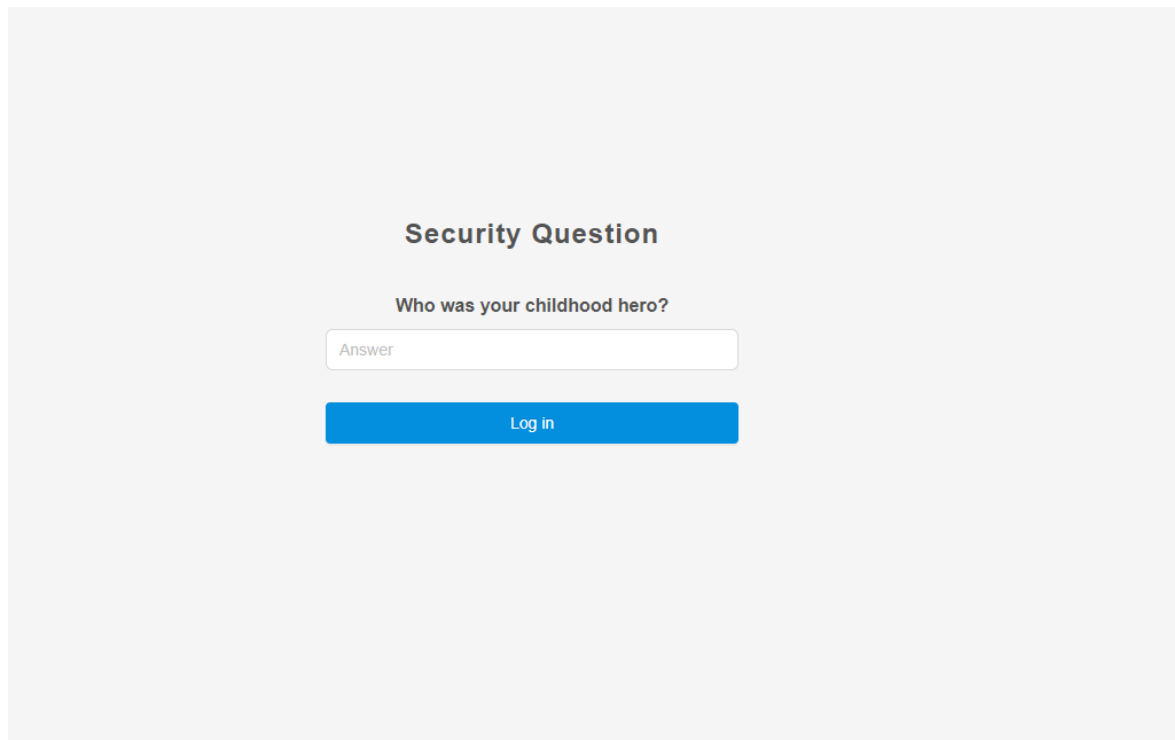
We are unable to host our GitLab repository on AWS Amplify [5] because we have self-hosted GitLab at Dal. However, we were able to find a workaround. When we were ready to test and deploy to the development or production environment, we produced a build. The application was then hosted on AWS Amplify using the build output directory.



## 8. Testing

### Login using valid credentials:

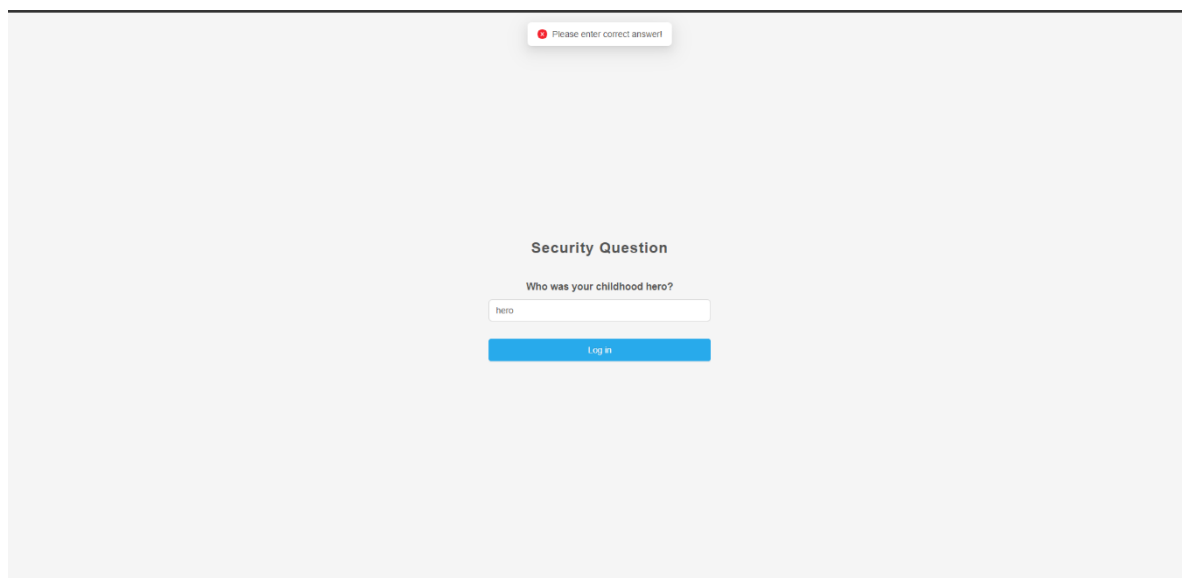
When a user logs in to the application and enters valid credentials then the user is navigated to a security question page.



The image shows a web page titled "Security Question". Below the title is the question "Who was your childhood hero?". There is a text input field with the placeholder text "Answer". Below the input field is a blue button labeled "Log in".

### Invalid security answer:

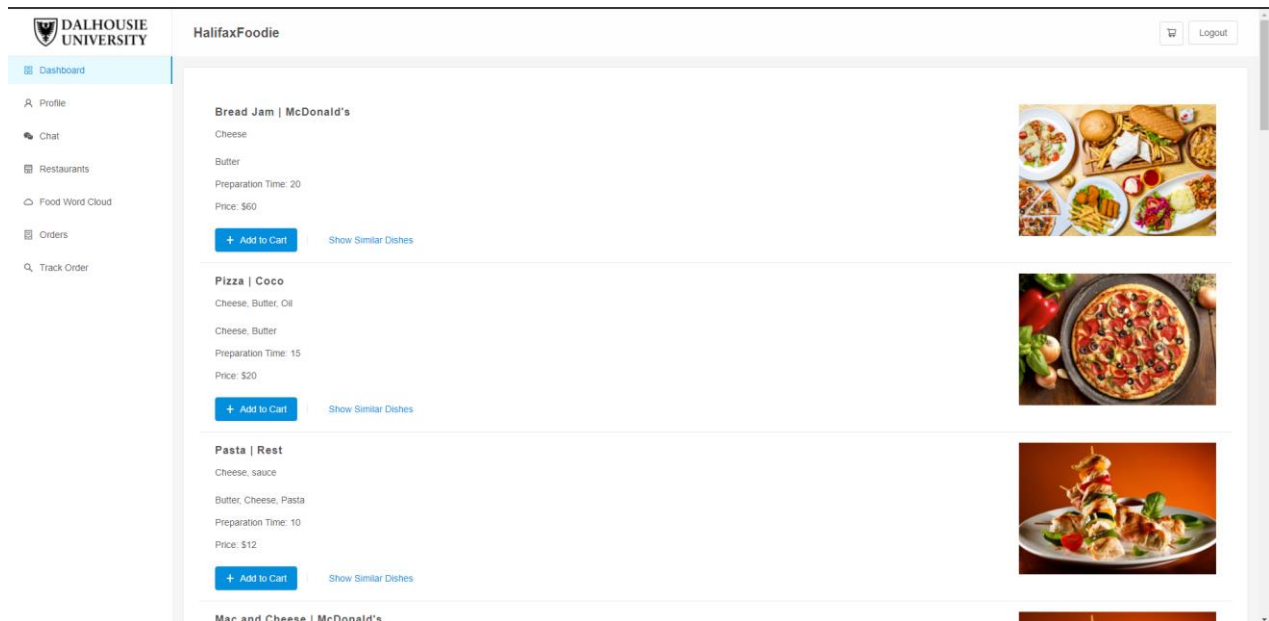
When a user enters valid login credentials and enters invalid security answer then an error message is displayed to the user.



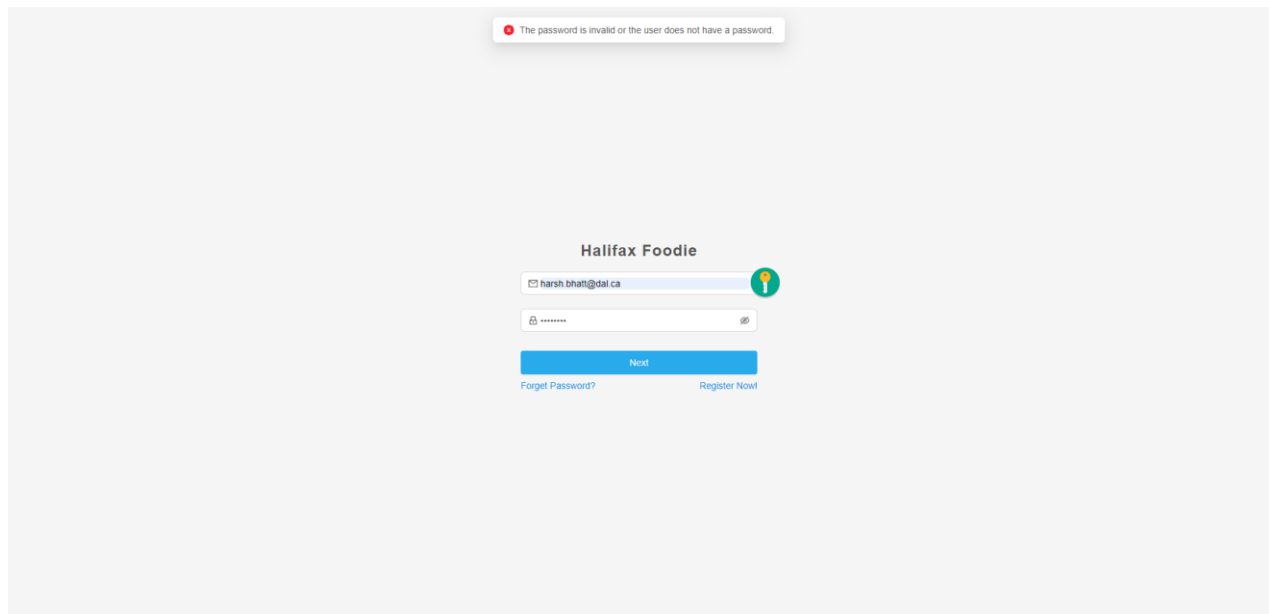
The image shows the same "Security Question" page as above, but with an error message displayed at the top: "Please enter correct answer!". The input field now contains the text "hero". The "Log in" button is still present.

**Valid security answer:**

When a user enters valid login credentials and enters a valid security answer user must be navigated to homepage

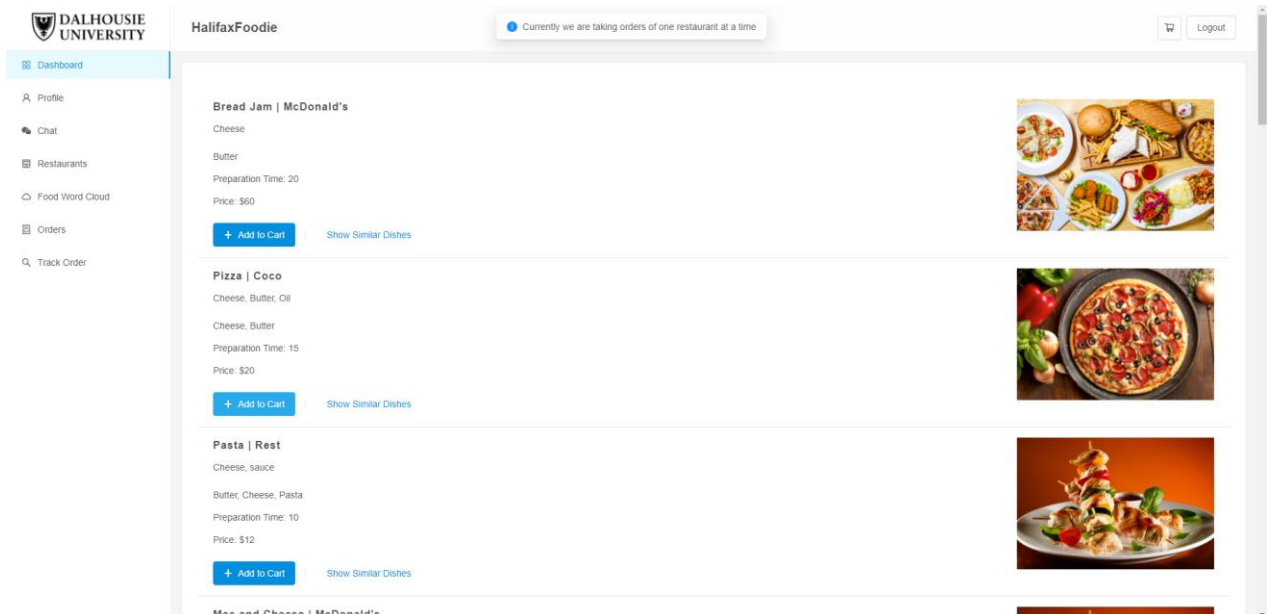
**Invalid credentials**

When a user tries to login with invalid credentials then an error message is shown to the user



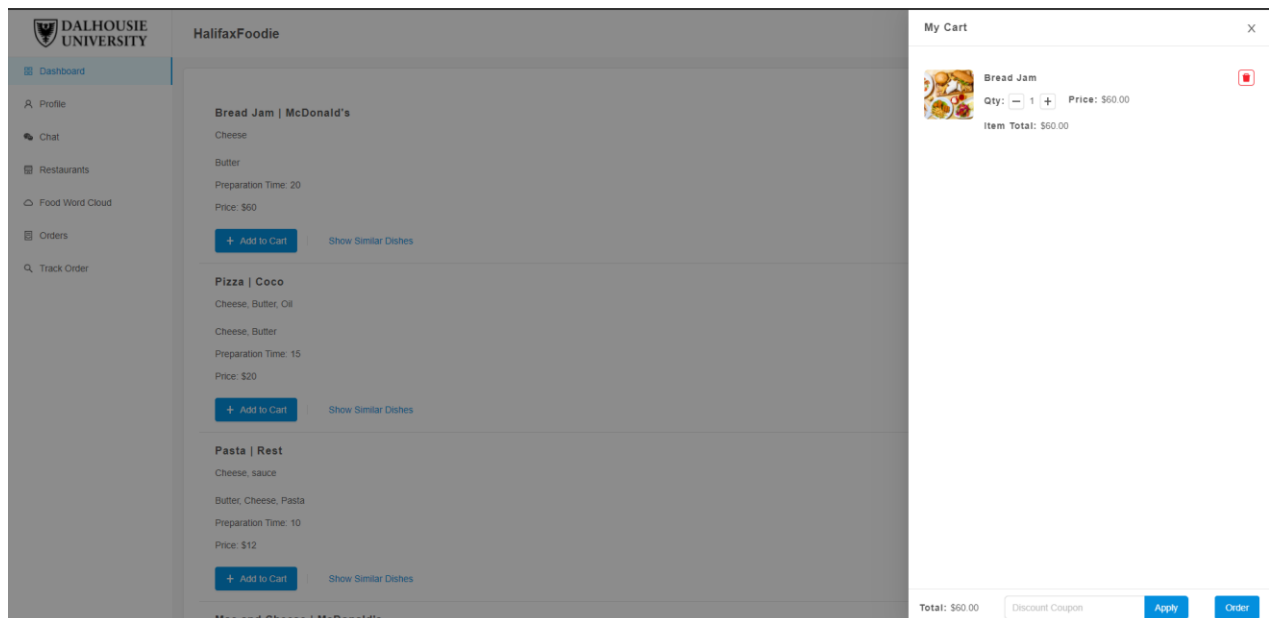
## Ordering from multiple restaurants:

When a user tries to order food from multiple restaurants then an error message is displayed to the user



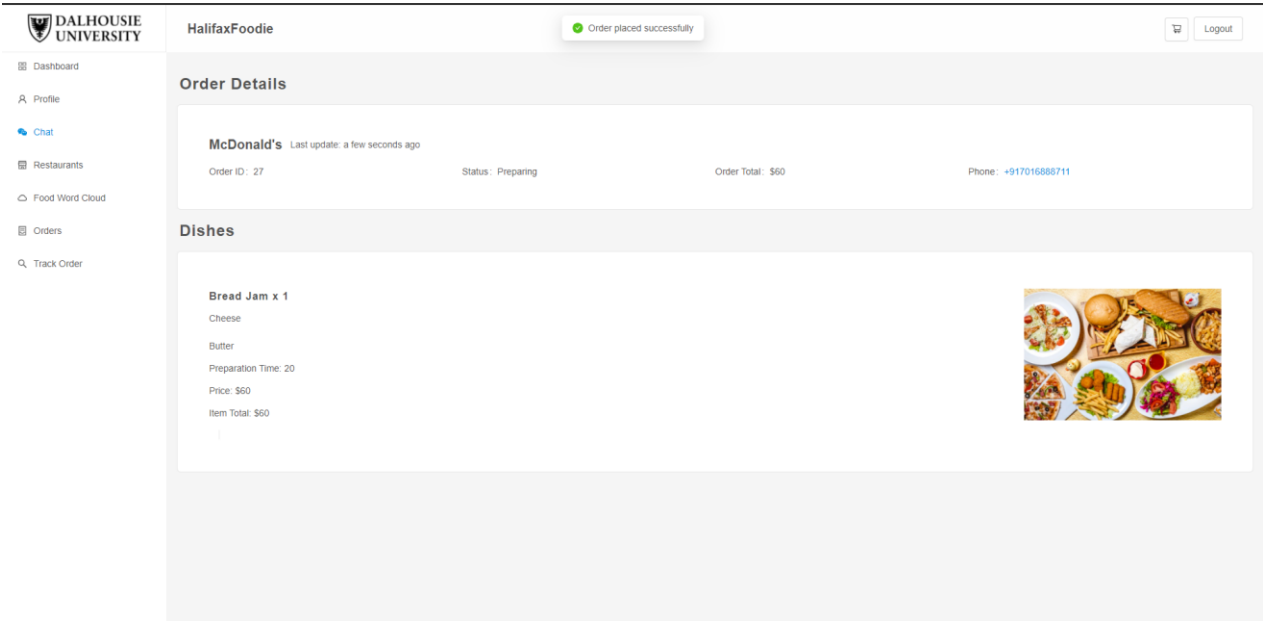
## Adding to cart:

When a user adds an item to a cart then that item should be displayed in cart



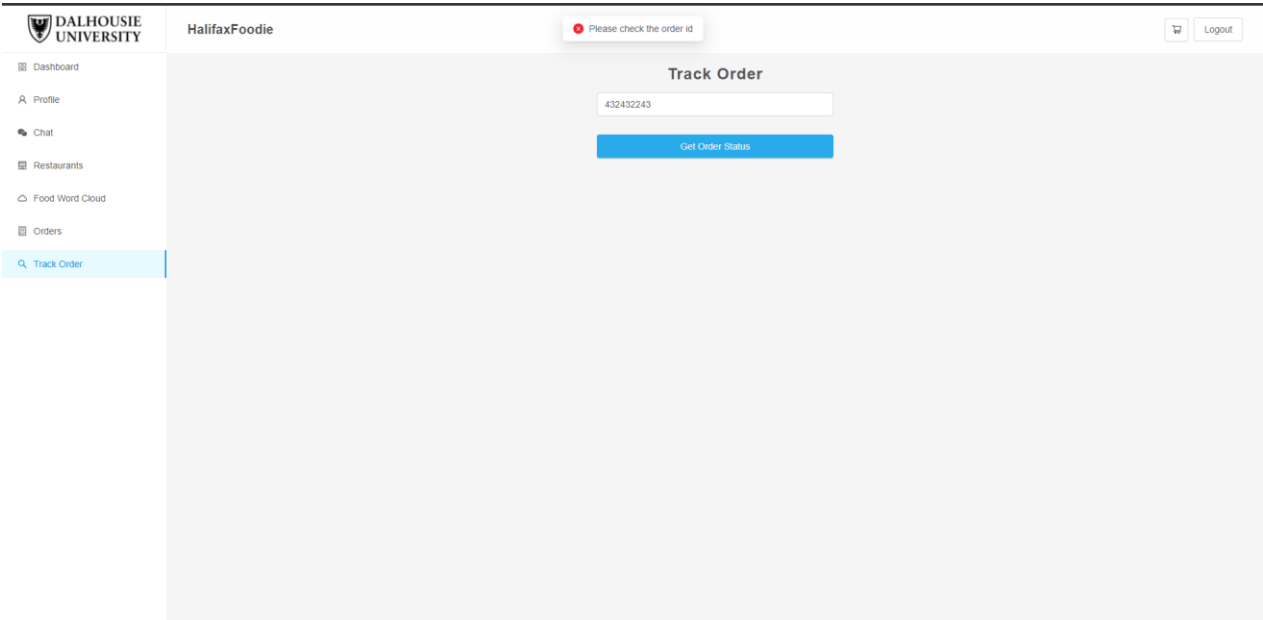
Order placed:

When a user places an order, then that order should be displayed in orders tab



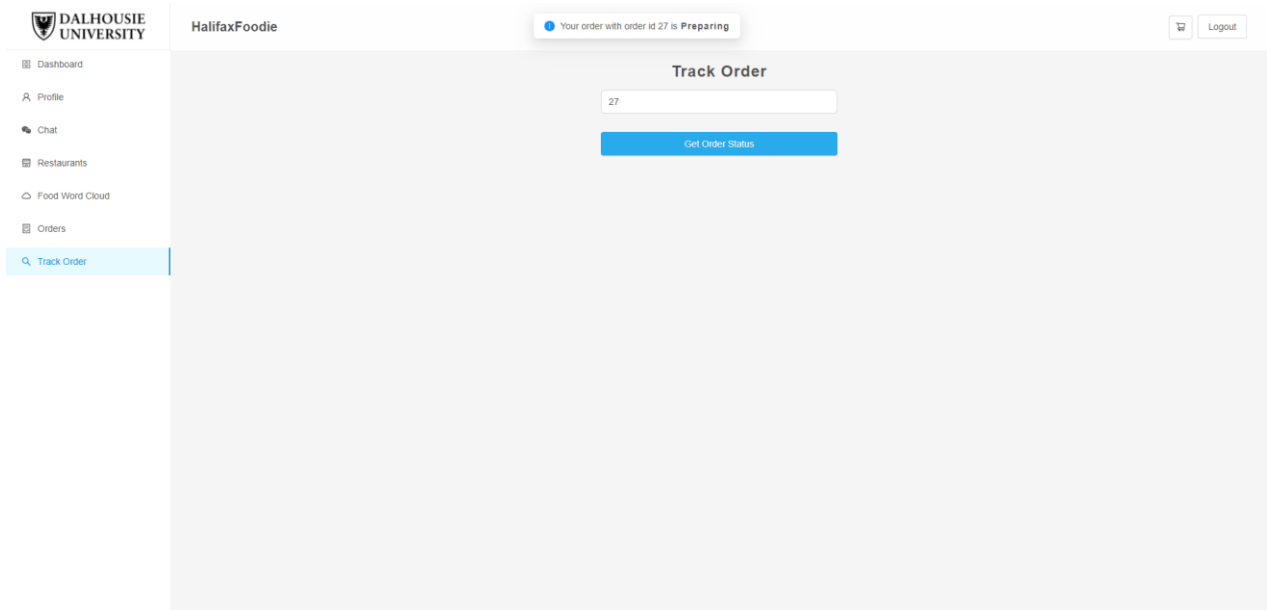
Tracking order with invalid order number:

When a user tracks the order with an invalid order number then an error message is displayed

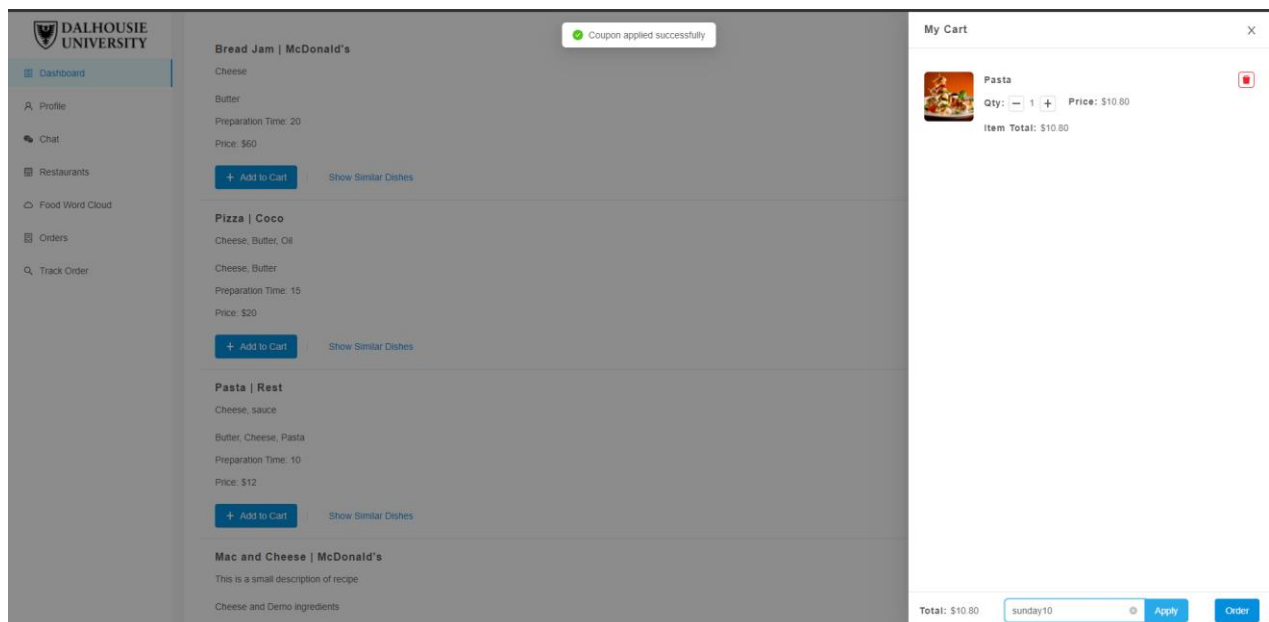


**Track order with valid order number:**

When a user tracks order with valid order number then the status of the order is displayed to the user

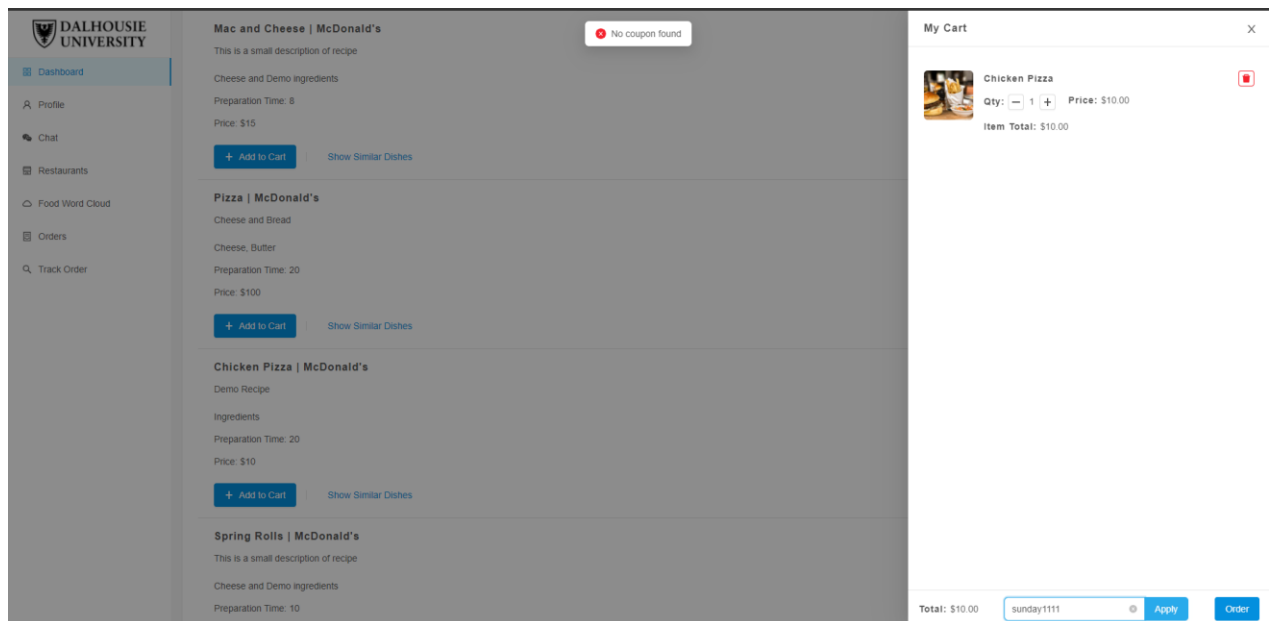
**Valid discount coupon:**

When a user enters valid discount coupon, then a discount is applied to the order



## Invalid discount coupon:

When a user enters invalid discount coupon, then an error message should be displayed



## 9. Visualization

We used Data Studio to visualize the data. To accomplish this, we will connect our RDS MySQL database. We will select the necessary table to visualize the data. For example, to study, which user is ordering what food item frequently, we will load the food item table and see the count of users based on the food item ordered. This way we can analyse, which food item is ordered frequently by a particular customer.

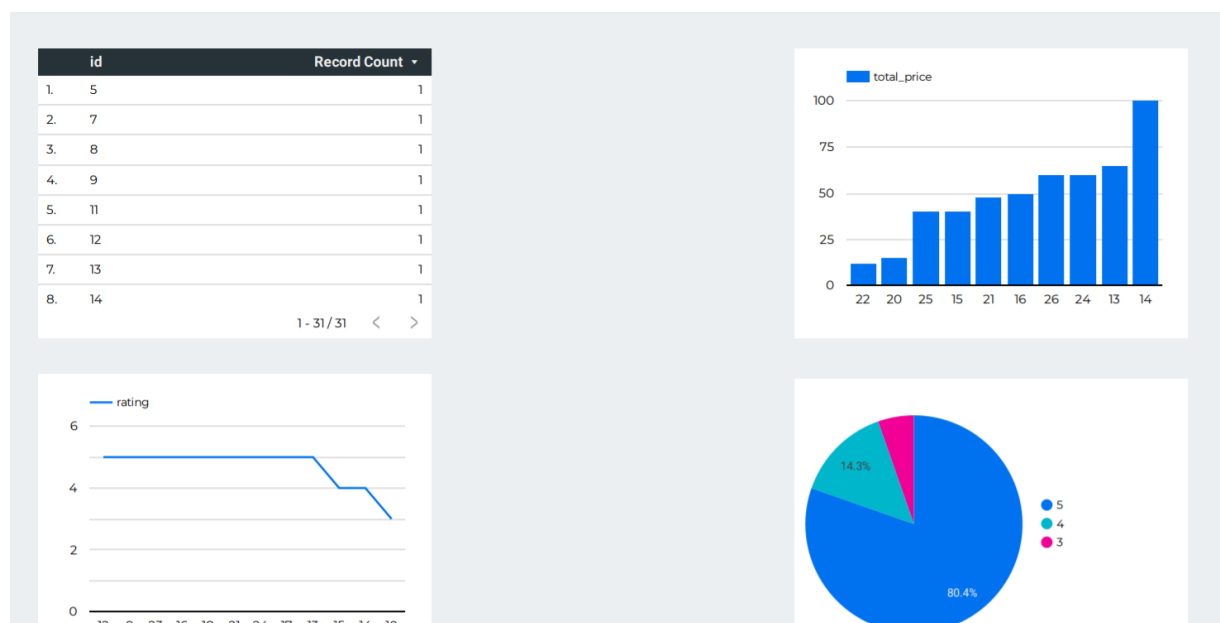


Figure 8: Ratings Visualization generated using Data Studio

We've made a graphic representation of the order's ratings as an example. As can be seen, **80.4** percent of all orders are rated 5 stars, whereas **0** percent of orders are rated 1 or 2. We plotted a bar and line graph for the same to demonstrate diversity. With the help of Data Studio, we can also see data from the other tables.

## 10. Application – Service Interaction

### API Gateway (AWS), Cloud Functions (GCP)

Whenever requests are made to the application through the website (user interface), the application in the backend will interact with the respective services that are able to provide to these requests. It could be a group/combination of services that will be used or a single service. In any case, the application will be able to communicate/interact with these services only through API Gateway/Cloud Functions [2] [1]. In case of intra-service interaction, it is still going to be the same, API gateway/Cloud Functions will be used if the service is present in the AWS cloud or GCP cloud environment, respectively. These services act as a single point of entry to their respective environments which increases security of services since the services are not exposed to public access directly, which in turn increases the overall security of the application.

## USE CASES

1. **Scenario:** A customer would like to know about different restaurants, dishes available in Halifax. He finds our application interesting and would like to start using it right away, to do so he must first register.

**Task Flow:** The user navigates to our application site. The user finds the “Register” options and selects it. The user enters all valid details. The system validates the input values. If all the data is valid, a user account is created for the user or else an appropriate message is shown to the user.

2. **Scenario:** A restaurant would like to use an online platform to provide service to their customers. The restaurant owner finds HalifaxFoodie apt and want to use it. As a new user he must register first.

**Task Flow:** The restaurant owner navigates to the HalifaxFoodie site. He then selects “Register”. He fills in the required details. The system validates the data. If the data is valid, they system creates a new restaurant tagged user or else an appropriate message is shown to the user.

3. **Scenario:** A customer is already registered and would like to see the different restaurants and their dishes available. To use the application, he must first login.

**Task Flow:** The customer navigates to HalifaxFoodie webpage. He then enters username and password. The system validates the given data. If the data is valid, the system then prompts the user to answer a security question or an error message with instructions are displayed to the user. When the user answers the security question, if the answer is correct the system authenticates the user and logs him in. or else the system prompts the user to try again.

4. **Scenario:** A customer has ordered few food items/dishes from a particular restaurant. He now wants to find the status of the order. He can find this with the help of HalifaxFoodie's smart assistant.

**Task Flow:** After the customer successfully logs into the application, He then navigates to online support – (HalifaxFoodie's smart assistant) and selects that option. The system provides a chatbot to the user. The system asks for details such as the order id and any other information that is needed. The user responds to those questions with the appropriate details. The status of the customer's order is given by the system when the user gives valid information.

5. **Scenario:** A customer is not satisfied with the online assistant and wants to escalate this to the respective restaurants' manager.

**Task Flow:** The user selects the option to escalate to restaurant manager. The system prompts the user for confirmation. If the user confirms, a notification is sent to the restaurant manager with a chatbot link to connect with the user. After the manager accepts the notification, the user is given a notification that the manager is available. The user connects with the manager through the live chat option.

6. **Scenario:** A customer is interested in finding out the most popular dishes prepared by different restaurants. He wants to get this information from HalifaxFoodie since it hosts multiple restaurants.

**Task flow:** After the customer log into the website successfully, the user navigates to "Most Popular items" and selects it. The system then displays a word cloud of different item names. The item name with the biggest size is the most popular dish and the smallest being the least among all the restaurants.

7. **Scenario:** A customer really enjoyed a particular dish. He now wants to try different dishes that are like the dish he liked. He wants this information on the HalifaxFoodie website.

**Task Flow:** The customer logs into the website successfully. He then navigates to "Recommendations" and selects it. The system picks up the last orders of the user and finds similar items. These items are displayed in a rank format based on their feedbacks and ratings.



## DATABASE DESIGN

Figure 2 illustrates the different tables, their attributes, and the relationship. The design is speculative and will change as we progress in development of the application.

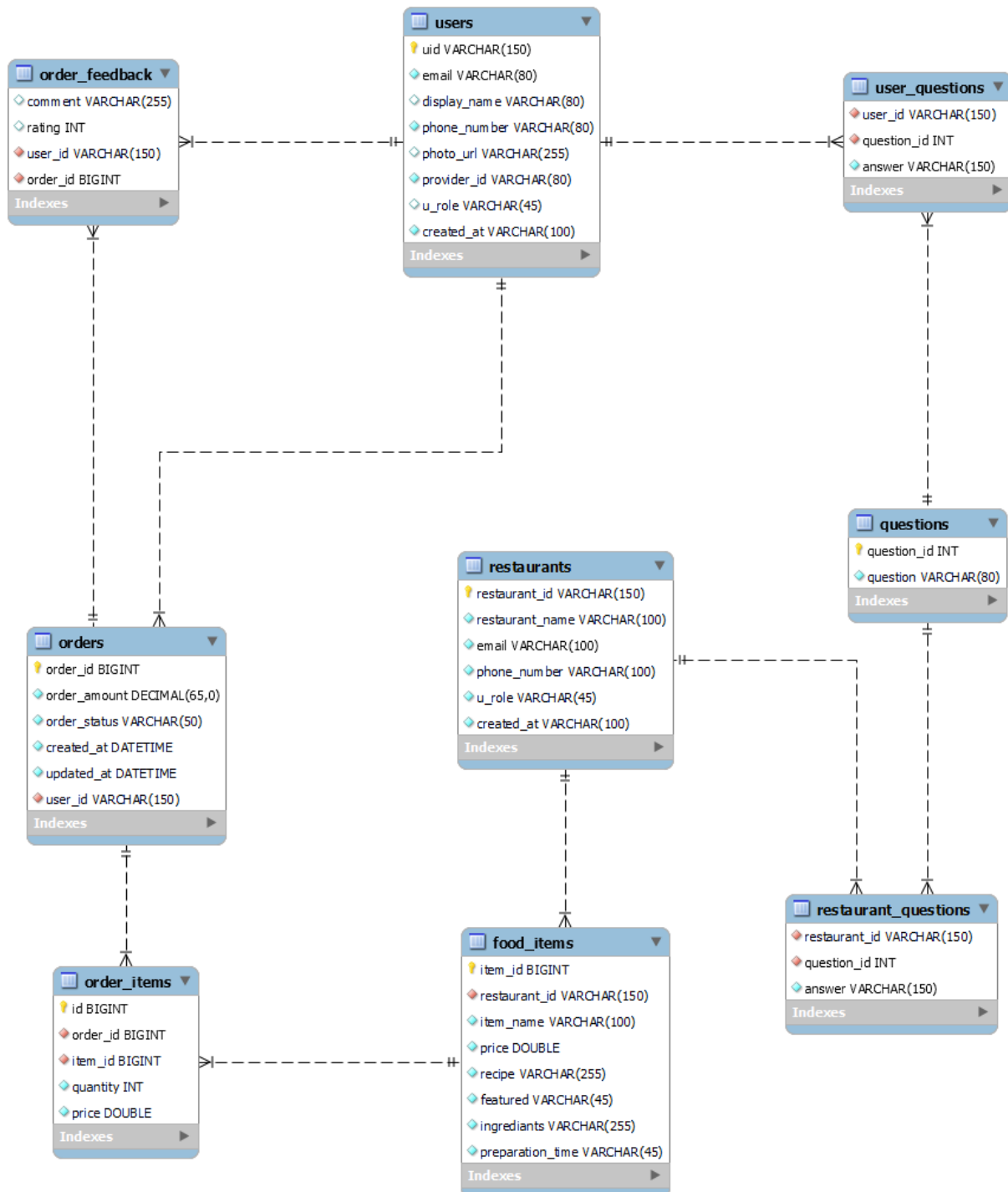


Figure 9: Database design of project

## TEAM COORDINATION

- As a team, we coordinated all the project related code at GitLab.
- We had different branch for each developer and one Development (dev) branch to merge code of all the developers and if all tests are successful, we then merge it to the Master (master) branch.
- The meeting logs demonstrate that the team members are well-coordinated.
- Despite a few ups and downs, we emerged as a strong team with each other's support.
- Because blockages are resolved at regular intervals, no one has to wait for others for an extended amount of time, allowing each developer to maximize their time.
- Harsh is excelled at React.js and Cloud Functions. Johanan shown exceptional competence in developing the model for calculating similarity scores. Vishnu accepted the challenge of finishing the chatting module and Lex bot. Prashant worked hard on the data processing task and finished it on time.
- Everyone in the group contributed to the creation and integration of the application by offering suggestions.

## PROGRAMMING FRAMEWORK

### Front-end

React.js

### Back-end

Node.js/Express.js, Python, GCP and AWS Services

### Database

MySQL (AWS RDS)

## CHALLENGES

### Credits

GCP and AWS both have different billing structure. Considering the fact, AWS is more cost effective as compared to GCP. To utilize credit efficiently, we have decided to run and host our app on AWS as Cloud Run is not an optimal solution in this case. It is quite difficult to determine the resources to be utilized by project specifications and this can only be effectively managed once we start de development.

### Security

Security is always a concern when developing something on cloud platform. In addition, security & privacy of the critical data such as user information will be a major concern. Moreover, user information such as password is handled by Identity Platform (GCP), and we have no control over it.

### Integration

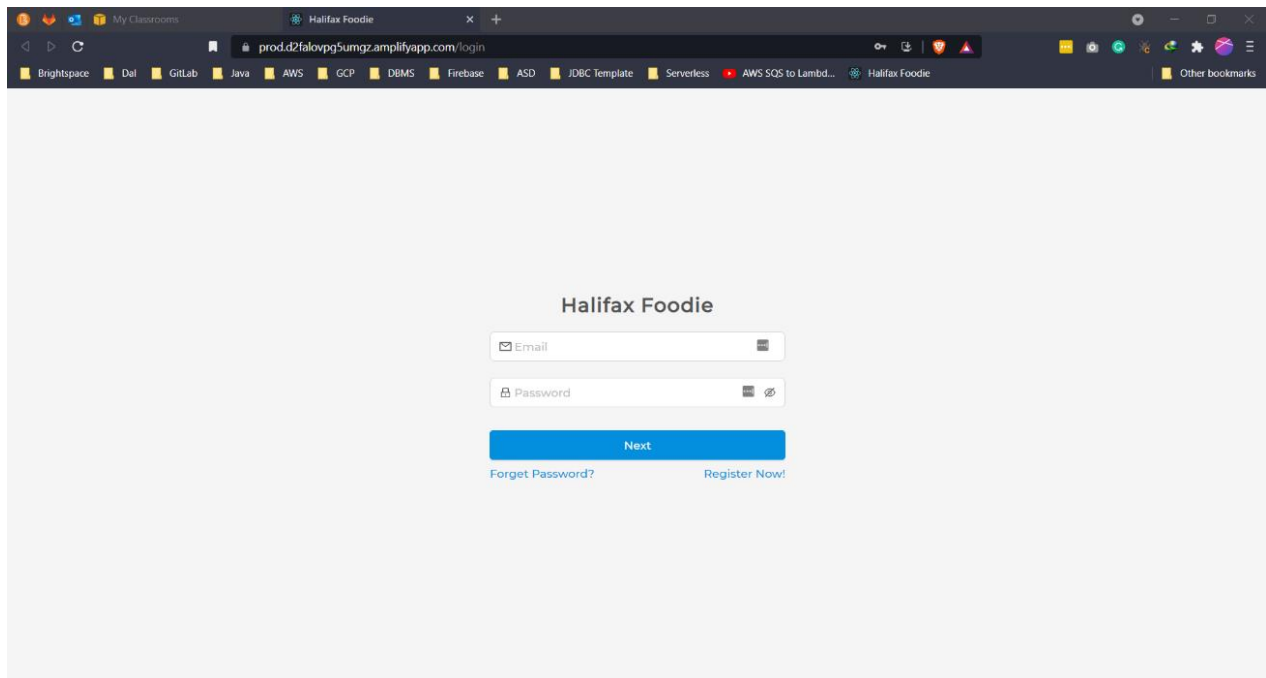
This is the most challenging part as we are developing our app using GCP ns AWS services i.e., multi-cloud platform. We must effectively implement an integration logic to have a seamless user experience throughout the session and robust error handling if one of the cloud services throws an error.

### GitLab Repository Link

<https://git.cs.dal.ca/hbhatt/saursbhdey-csci-5410-group-7-may-2021-project>

# USER INTERFACE

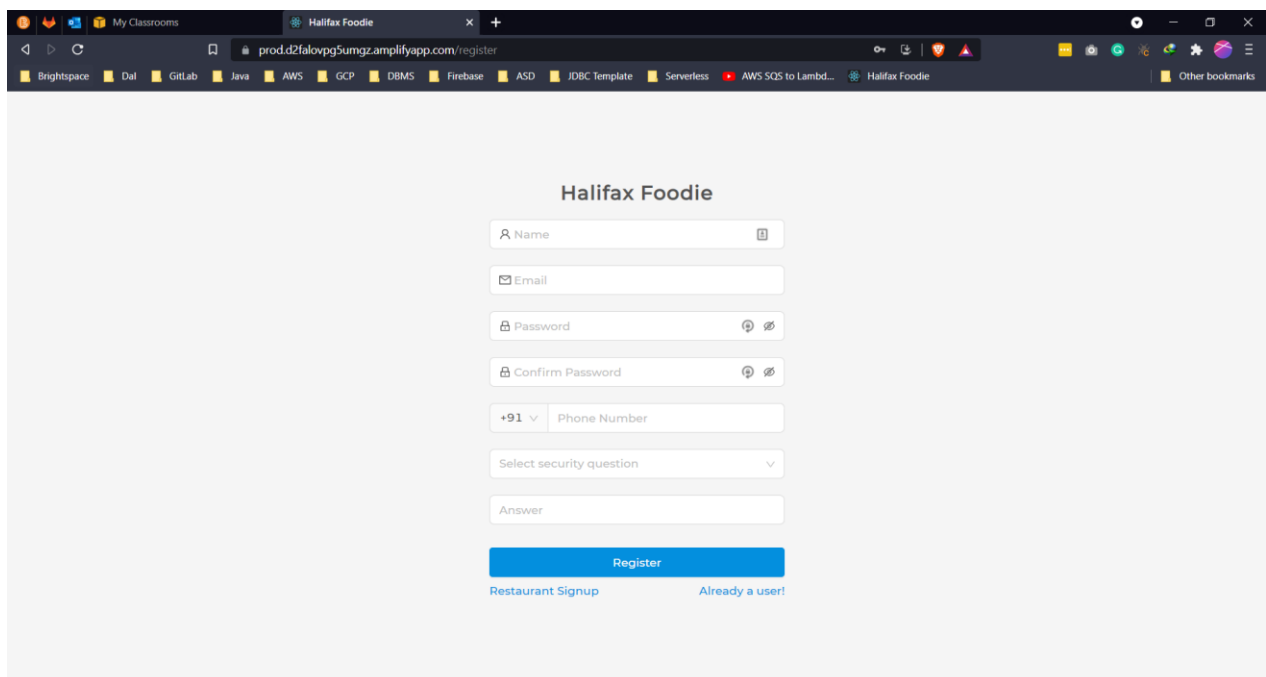
## 1. Login



The screenshot shows a web browser window with the URL `prod.d2falovpg5umgz.amplifyapp.com/login`. The page title is "Halifax Foodie". The login form consists of two input fields: "Email" and "Password", each with a small icon on the right. Below the fields is a blue "Next" button. At the bottom of the form, there are two links: "Forgot Password?" and "Register Now!".

Figure 10: Login Screen

## 2. Customer Signup



The screenshot shows a web browser window with the URL `prod.d2falovpg5umgz.amplifyapp.com/register`. The page title is "Halifax Foodie". The signup form consists of several input fields: "Name", "Email", "Password", "Confirm Password", "Phone Number" (with a dropdown for "+91"), "Select security question", and "Answer". Below the fields is a blue "Register" button. At the bottom of the form, there are two links: "Restaurant Signup" and "Already a user?".

Figure 11: Signup Screen

### 3. Restaurant Signup Protection

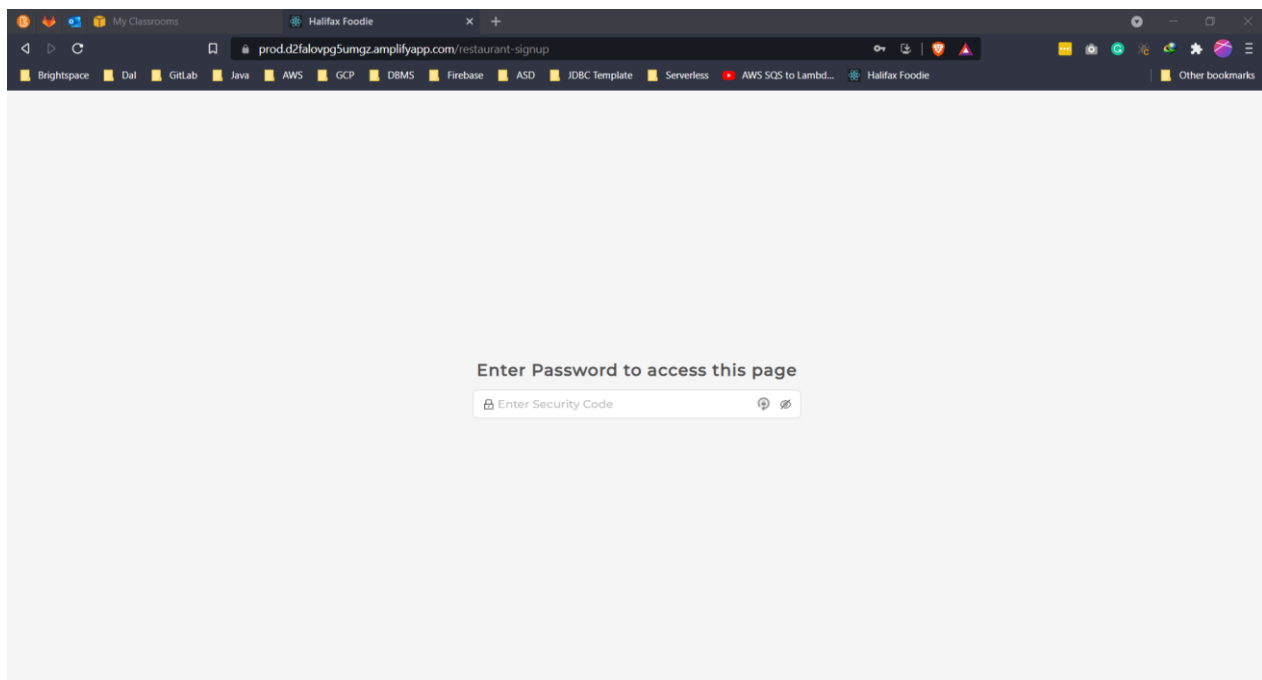


Figure 12: Restaurant Signup Lock Screen

### 4. Restaurant Signup

A screenshot of a web browser window showing the 'Restaurant Signup' form. The form is centered on a light gray background. It contains the following fields: 'Restaurant Name' (with a user icon), 'Email' (with an envelope icon), 'Password' (with a lock icon and a toggle for visibility), 'Confirm Password' (with a lock icon and a toggle for visibility), 'Phone Number' (with a dropdown for country code set to '+91' and a phone icon), 'Select security question' (a dropdown menu), and 'Answer' (a text input field). Below the fields is a blue 'Register' button. At the bottom of the form, there is a link that says 'Already a user?'. The browser's address bar and bookmark bar are visible at the top.

Figure 13: Restaurant Signup Screen

## 5. Reset Password

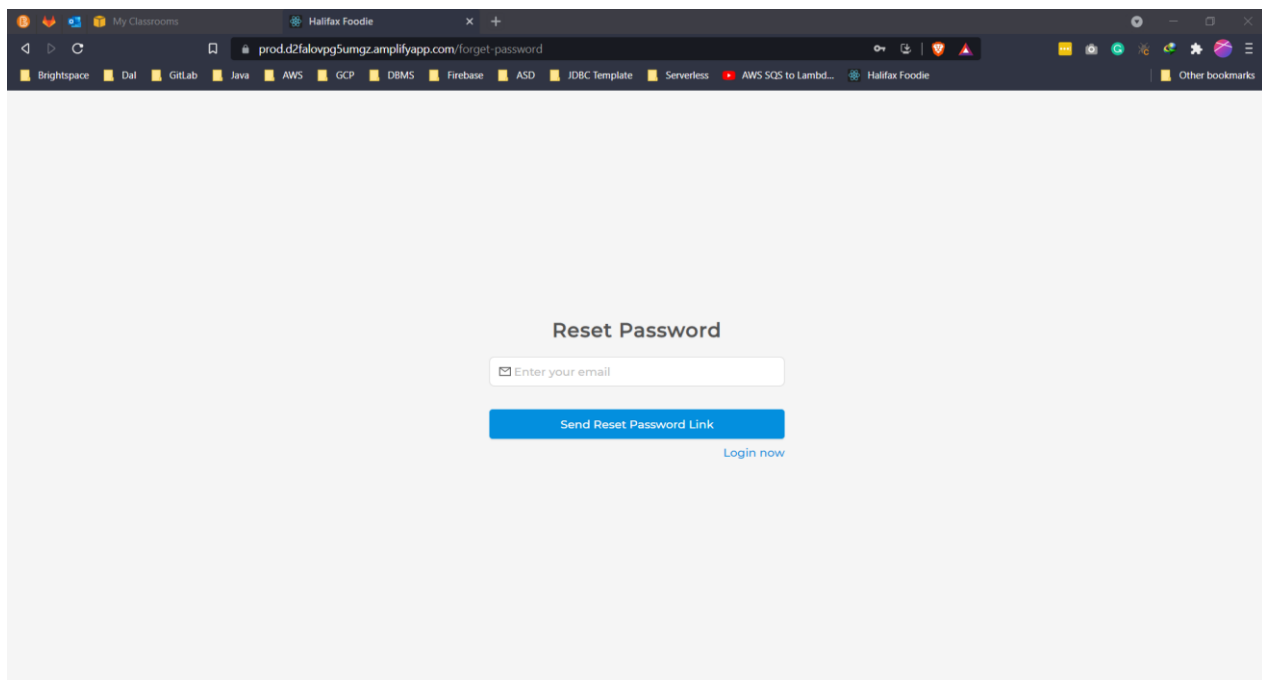


Figure 14: Reset Password Screen

## 6. Security Question

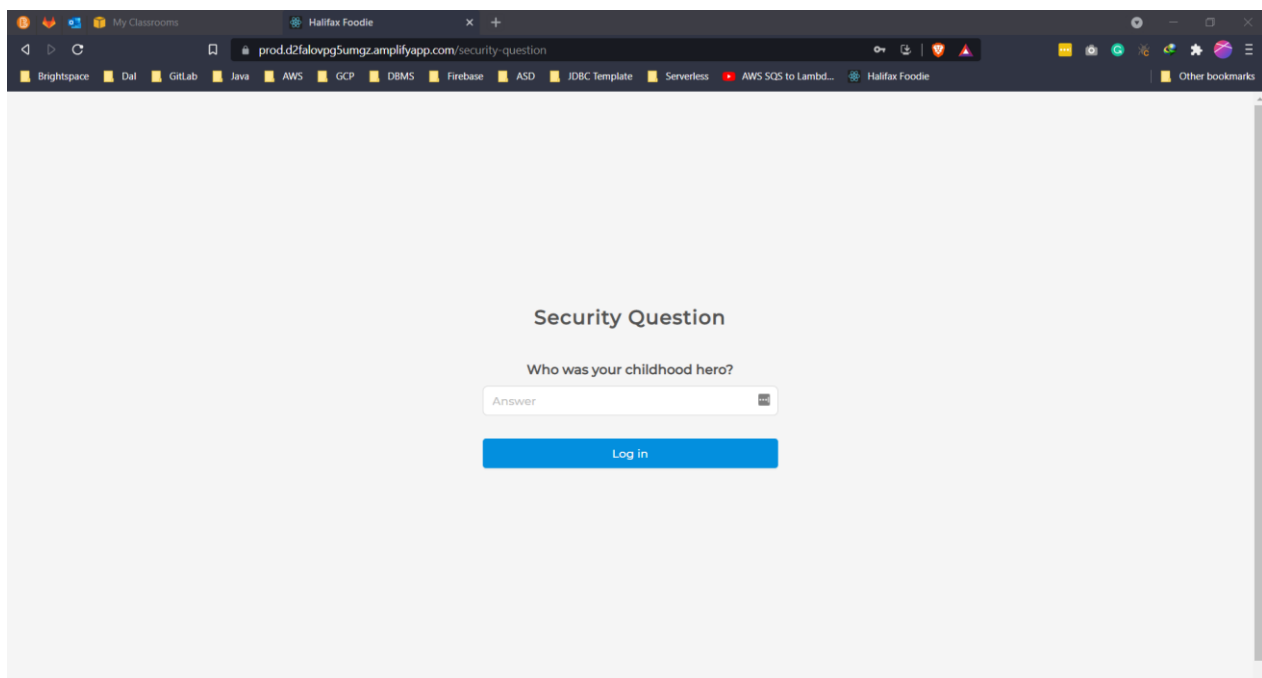


Figure 15: Security Question Screen

7. User Dashboard

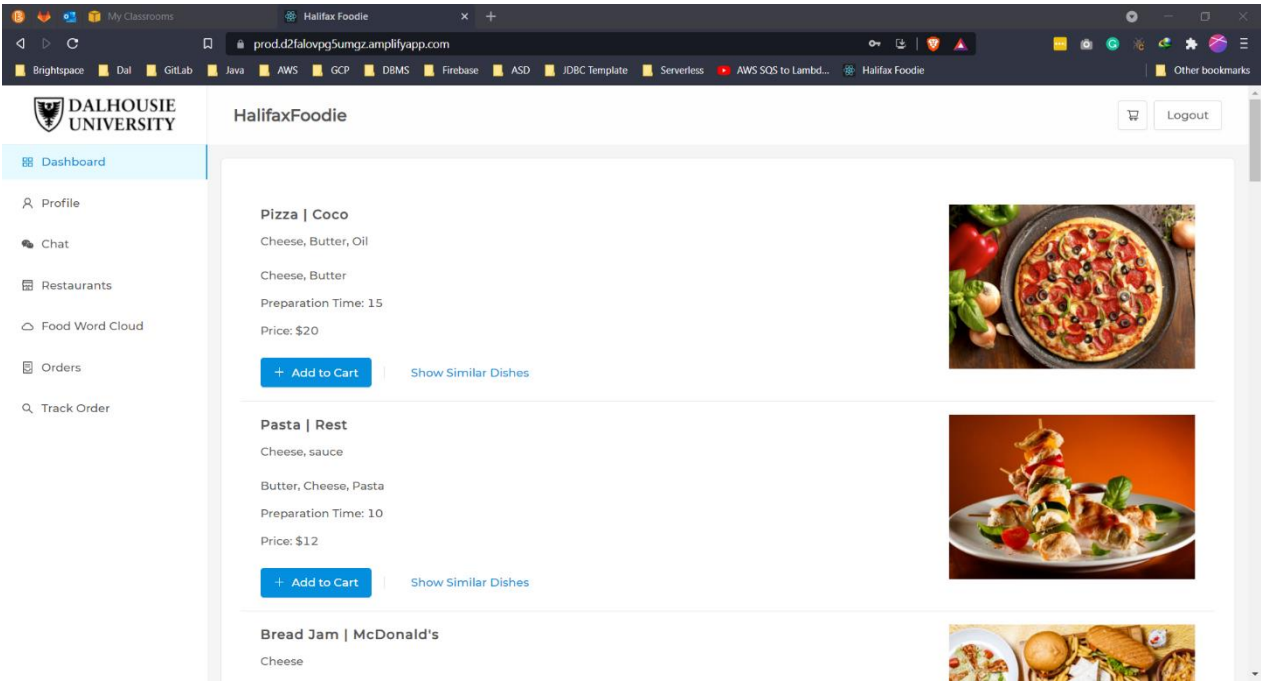


Figure 16: User Dashboard Screen

8. Customer/Restaurant Profile

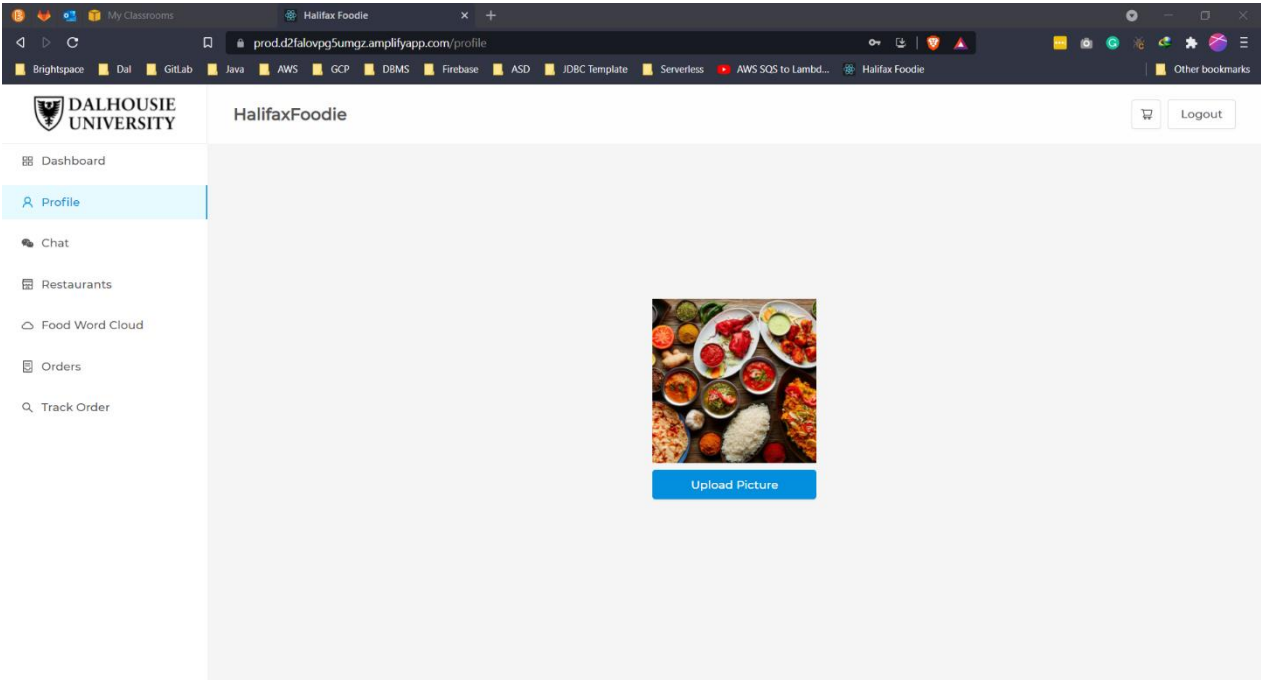


Figure 17: Profile Screen

## 9. Live Chat Support using Lex Bot

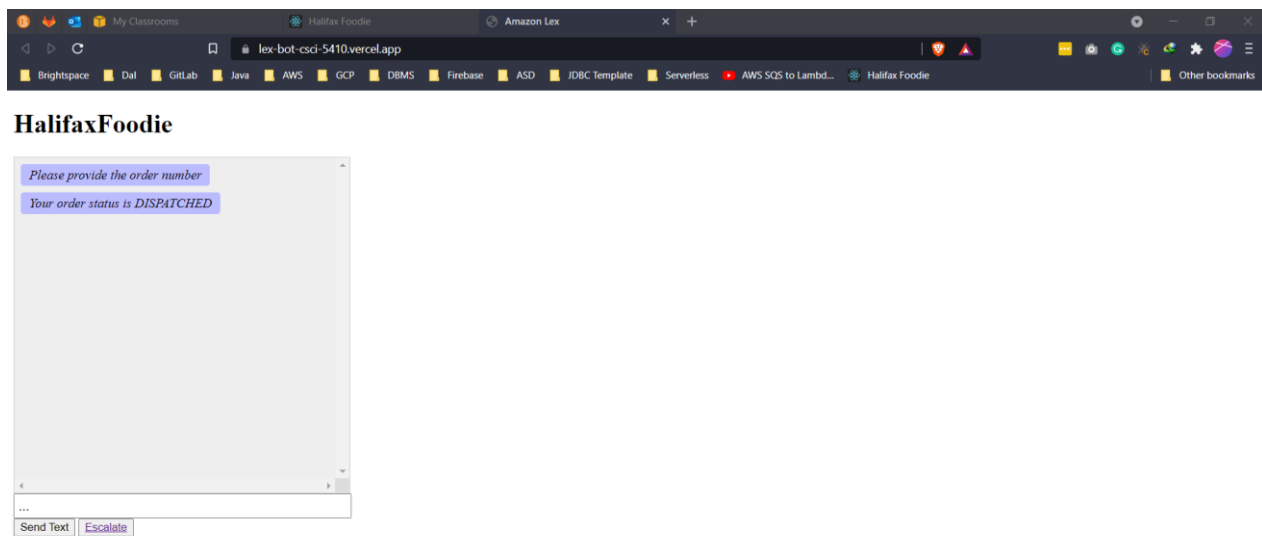


Figure 18: AWS Lex Bot Live Chat

## 10. Customer and Restaurant Employee/Manager Chat using Pub/Sub

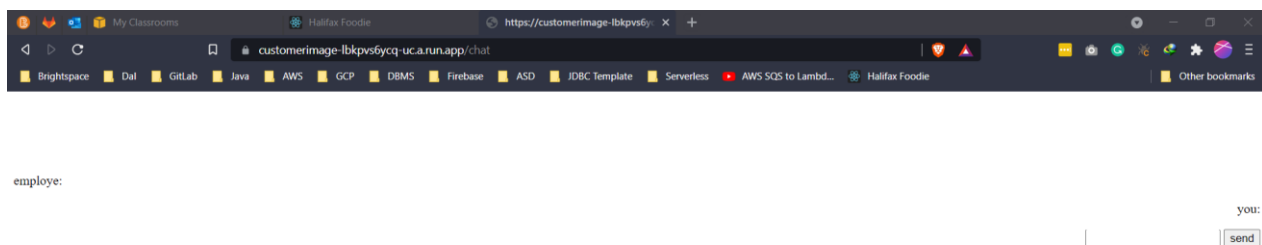


Figure 19: Pub/Sub Chat Screen



## 11. List of restaurants

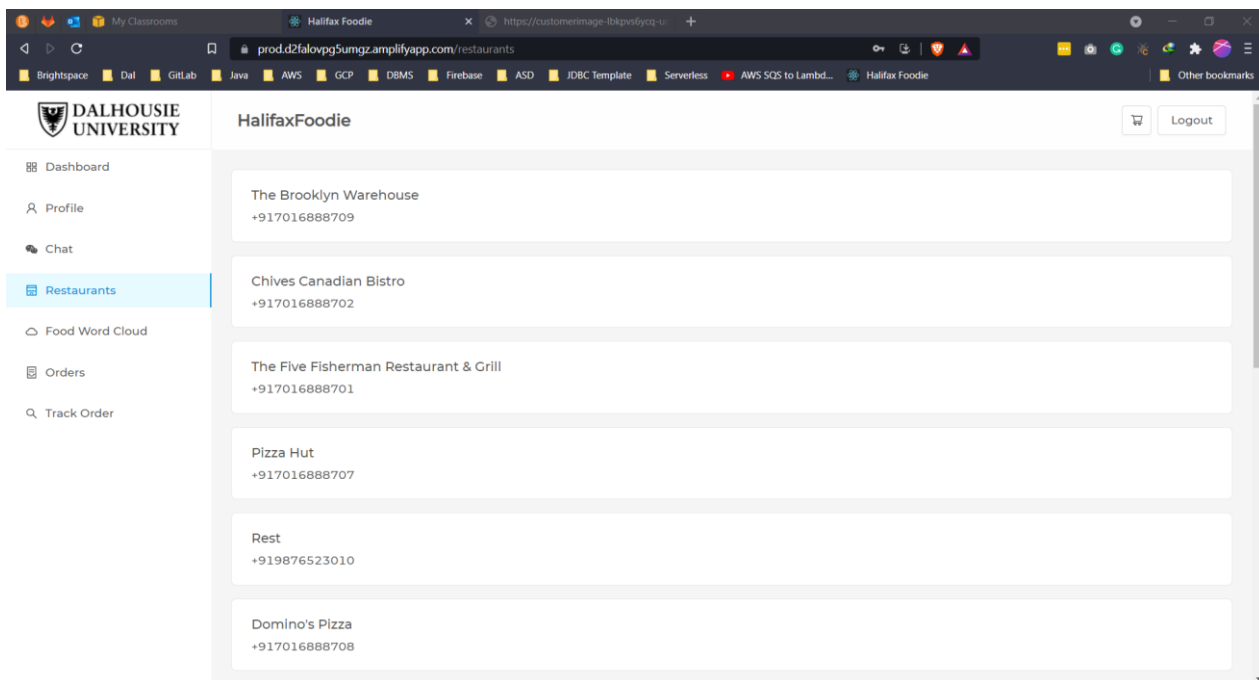


Figure 20: List of Restaurants Screen

## 12. Restaurant page

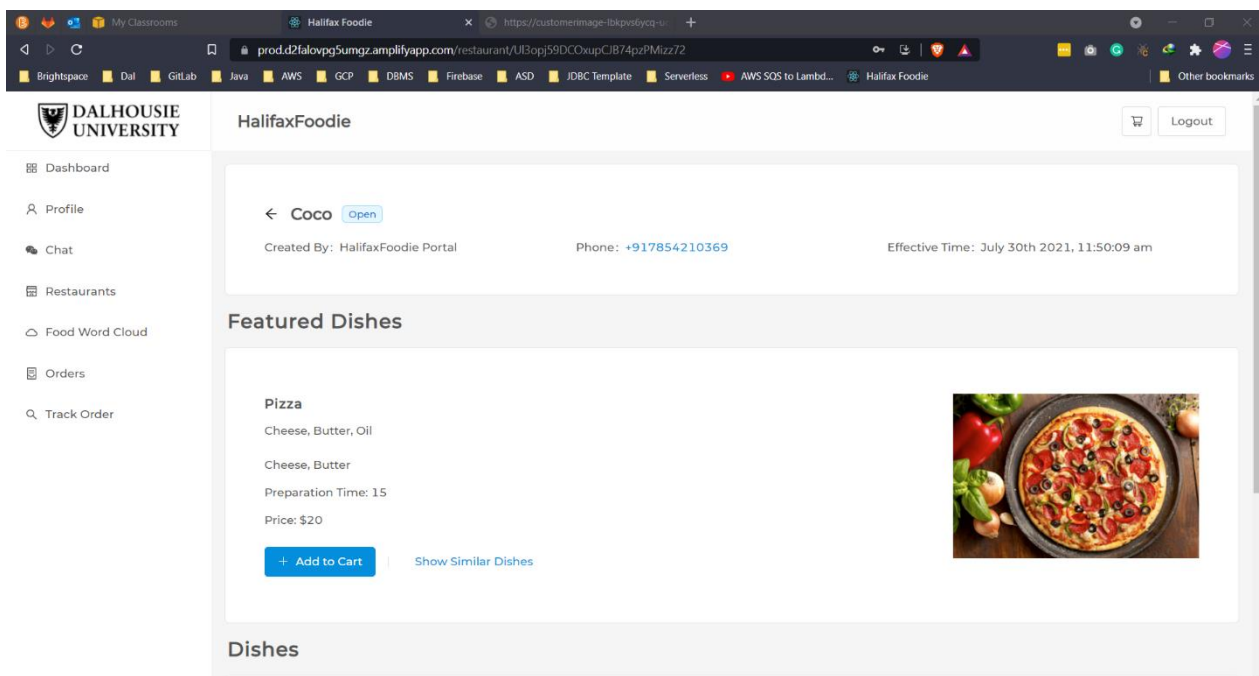


Figure 21: Restaurant Screen

13. Orders List page for Customer

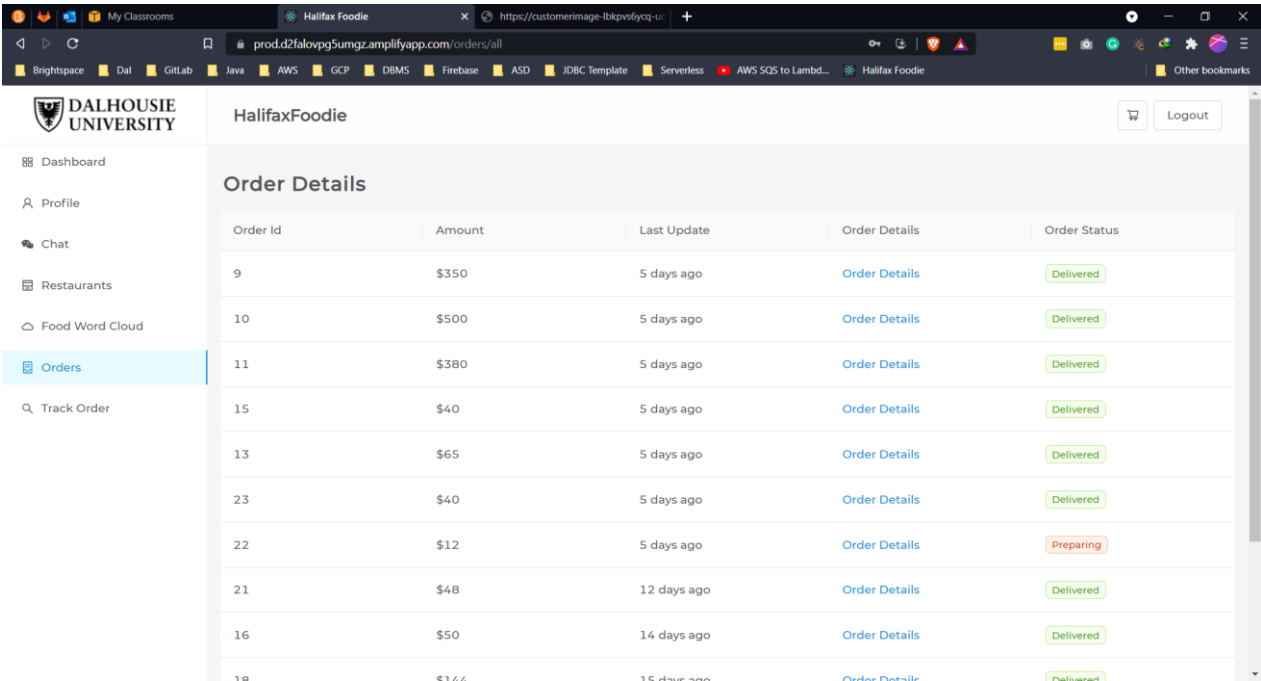


Figure 22: Order History Screen

14. Order review page for Customer

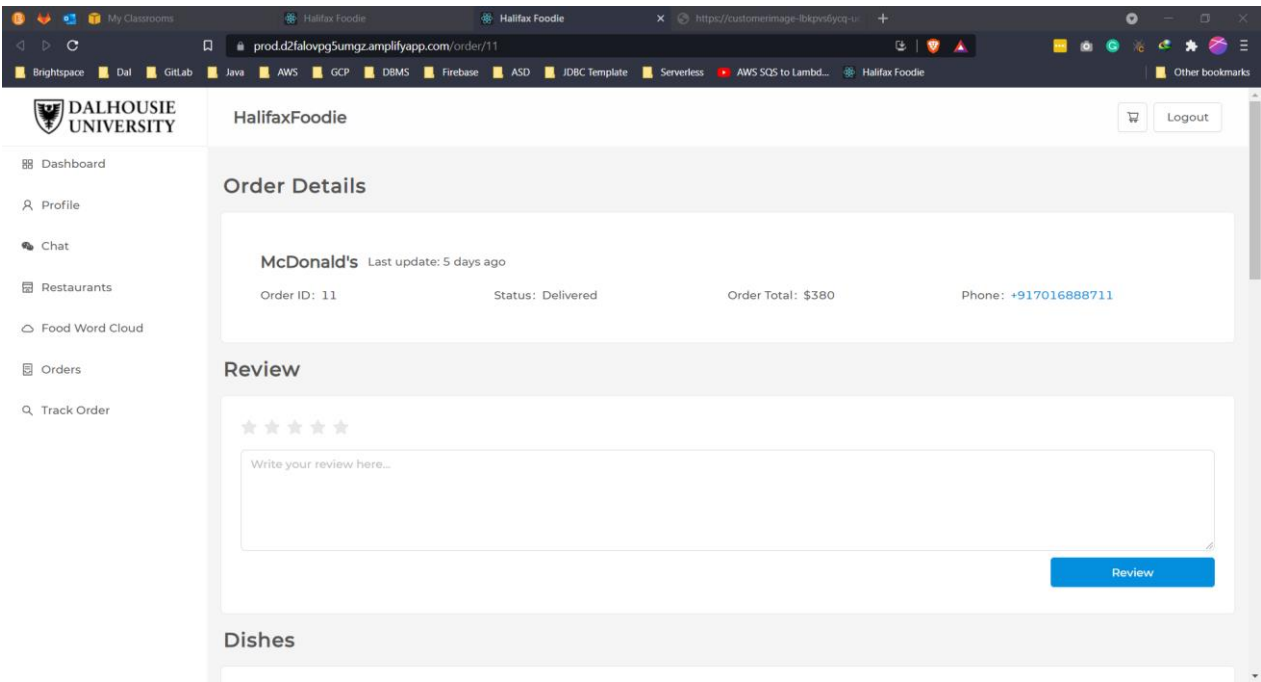


Figure 23: Review Order Screen

## 15. Order detail page

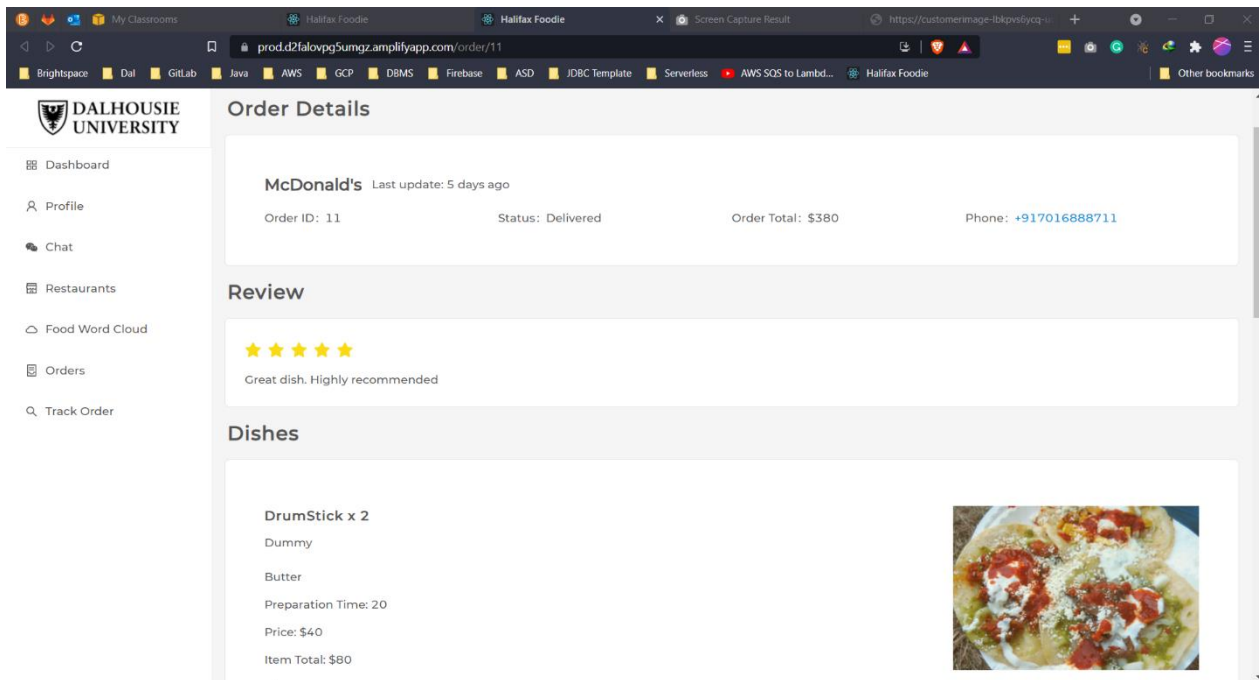


Figure 24: Order Detail Screen

## 16. Track order

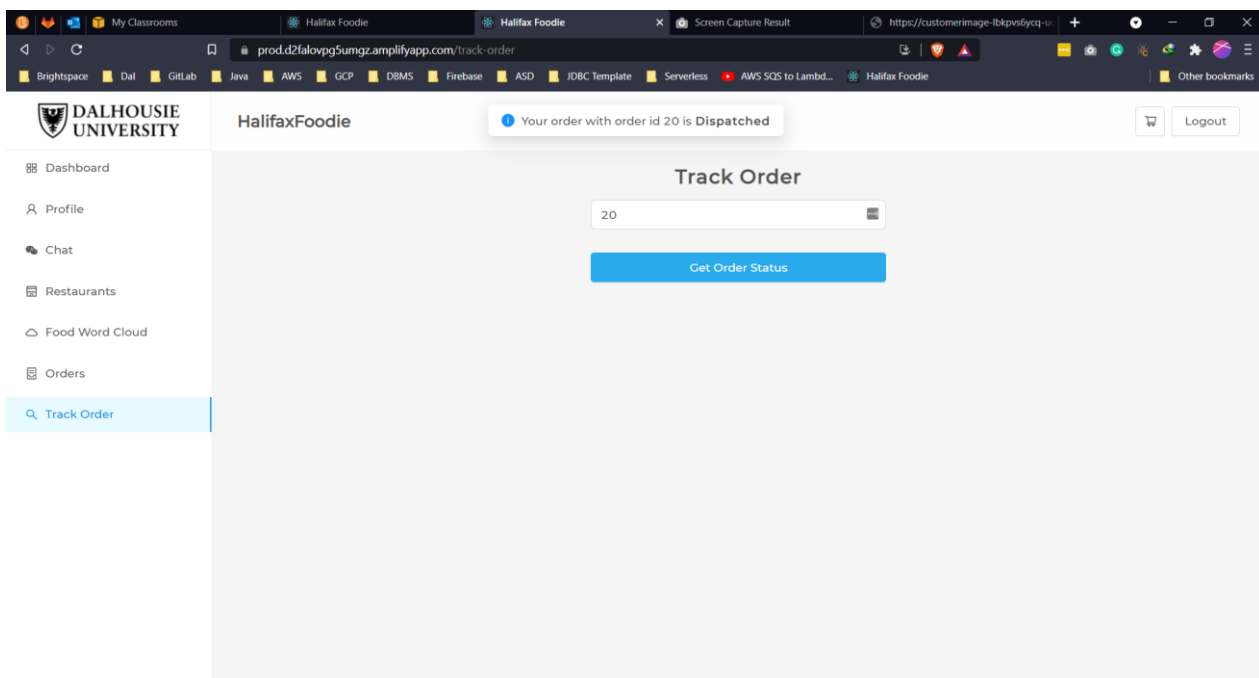


Figure 25: Track Order Screen

## 17. My Cart

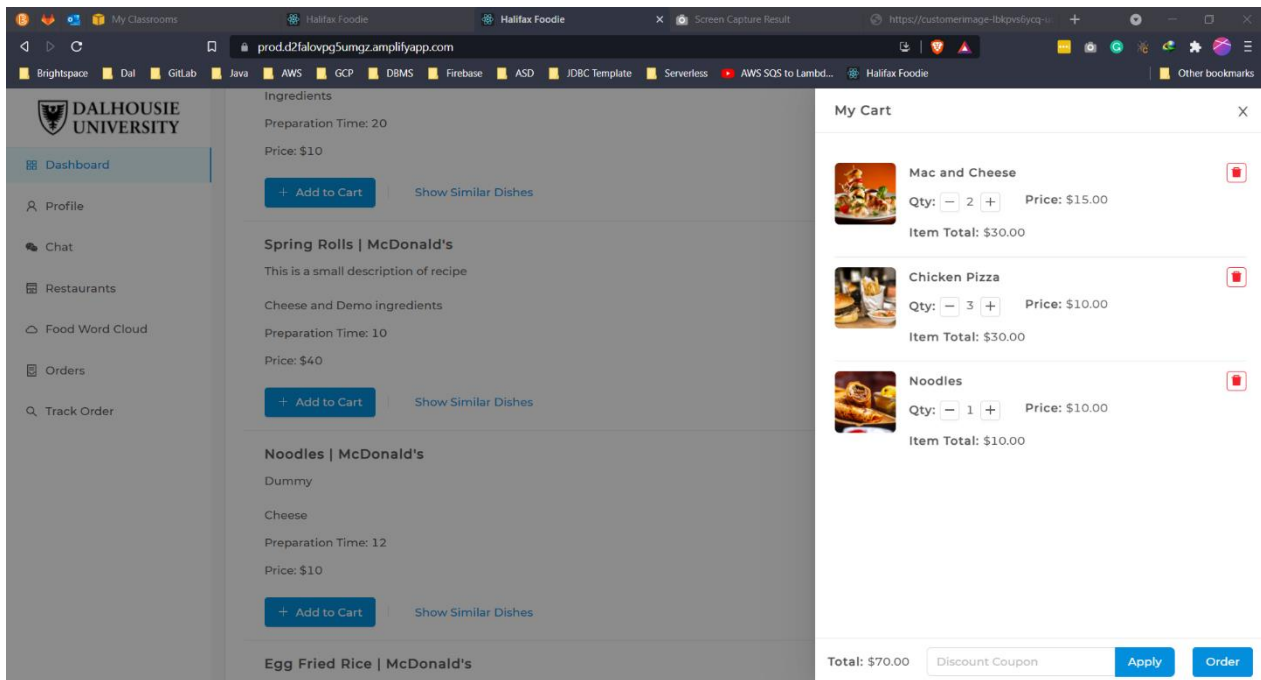


Figure 26: My Cart

## 18. Coupon Code

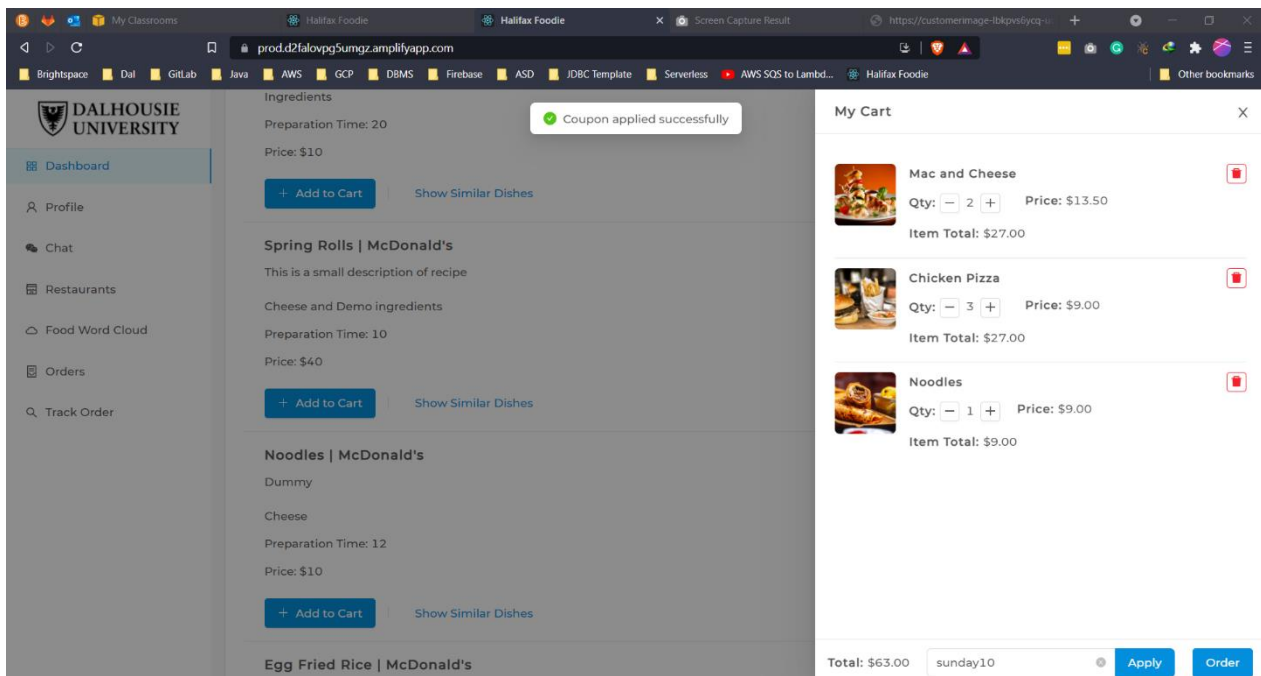


Figure 27: Coupon Code

## 19. WordCloud of frequently ordered dishes



Figure 28: Word Cloud of frequently ordered dishes

## 20. Similar dishes suggestions

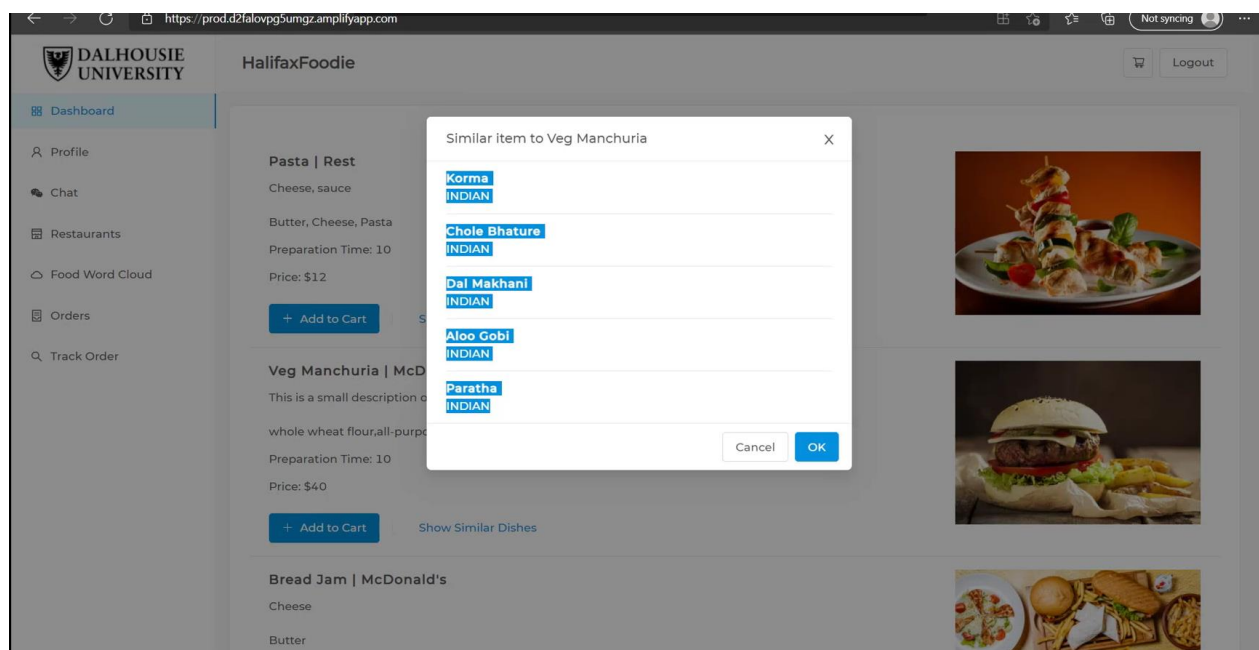


Figure 29: Similar Dishes Suggestions

21. Restaurant Dashboard

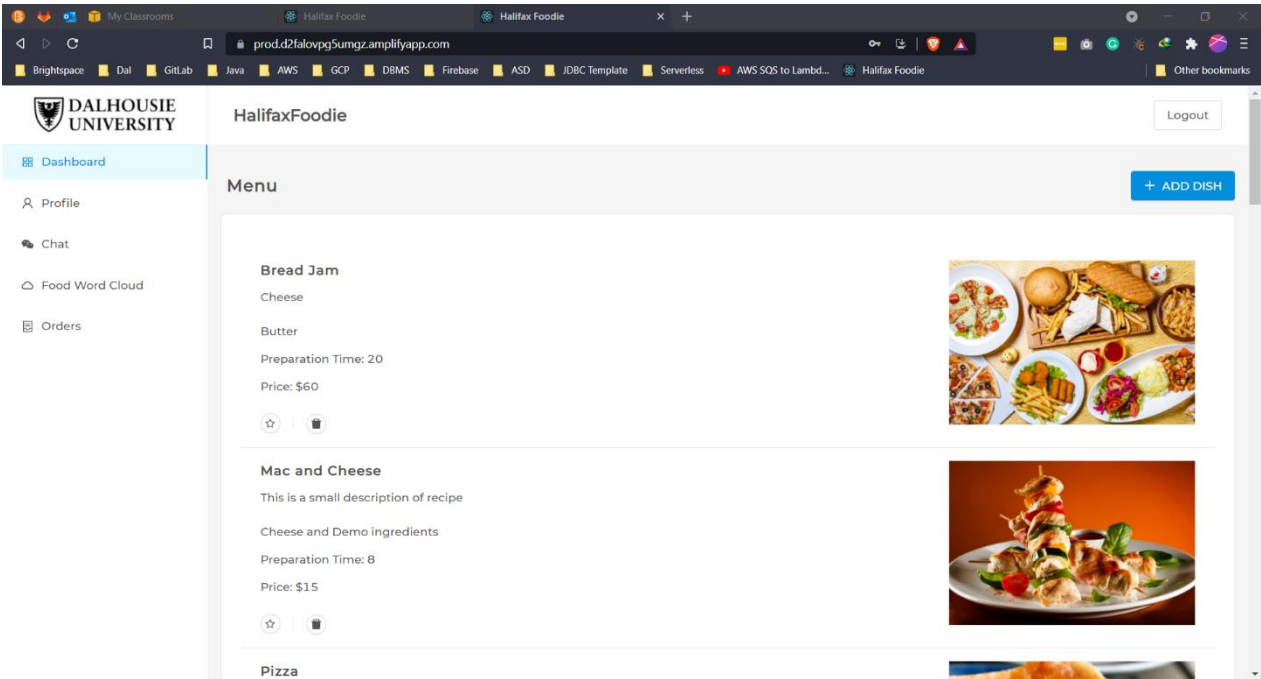


Figure 30: Restaurant Dashboard Screen

22. Featuring dish to show at the top of restaurant's menu

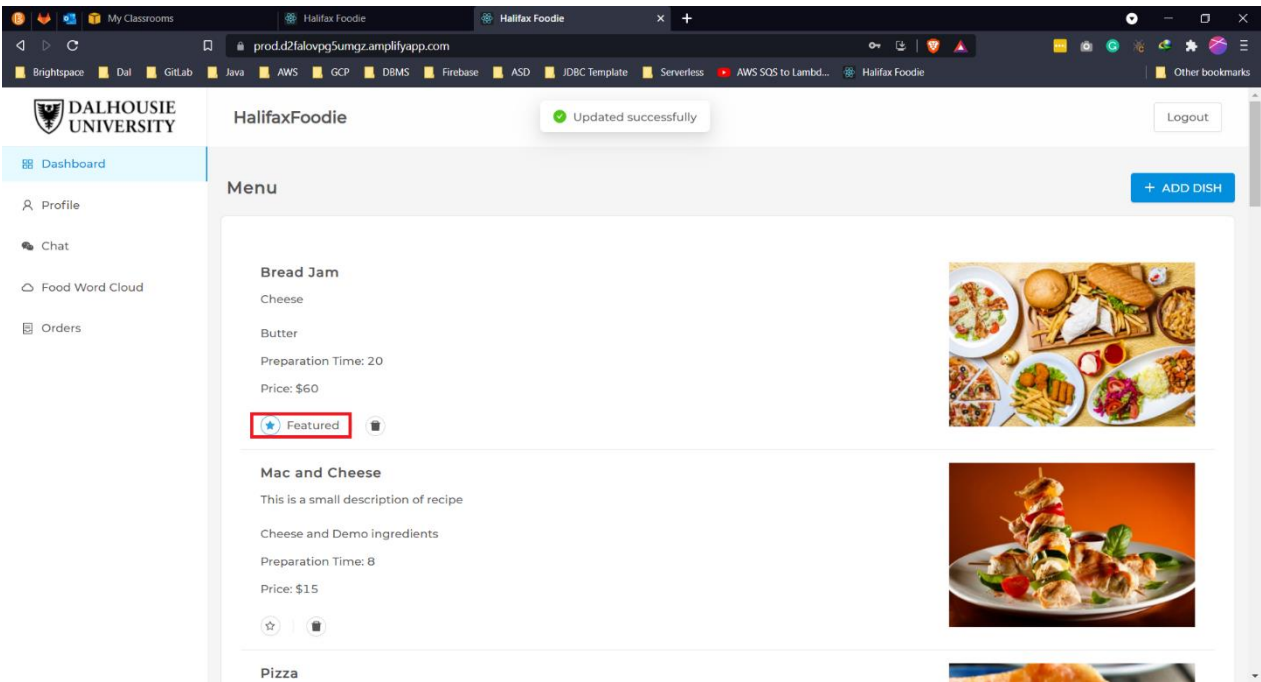


Figure 31: Featured Food Item

## 23. Delete Food Item from Menu

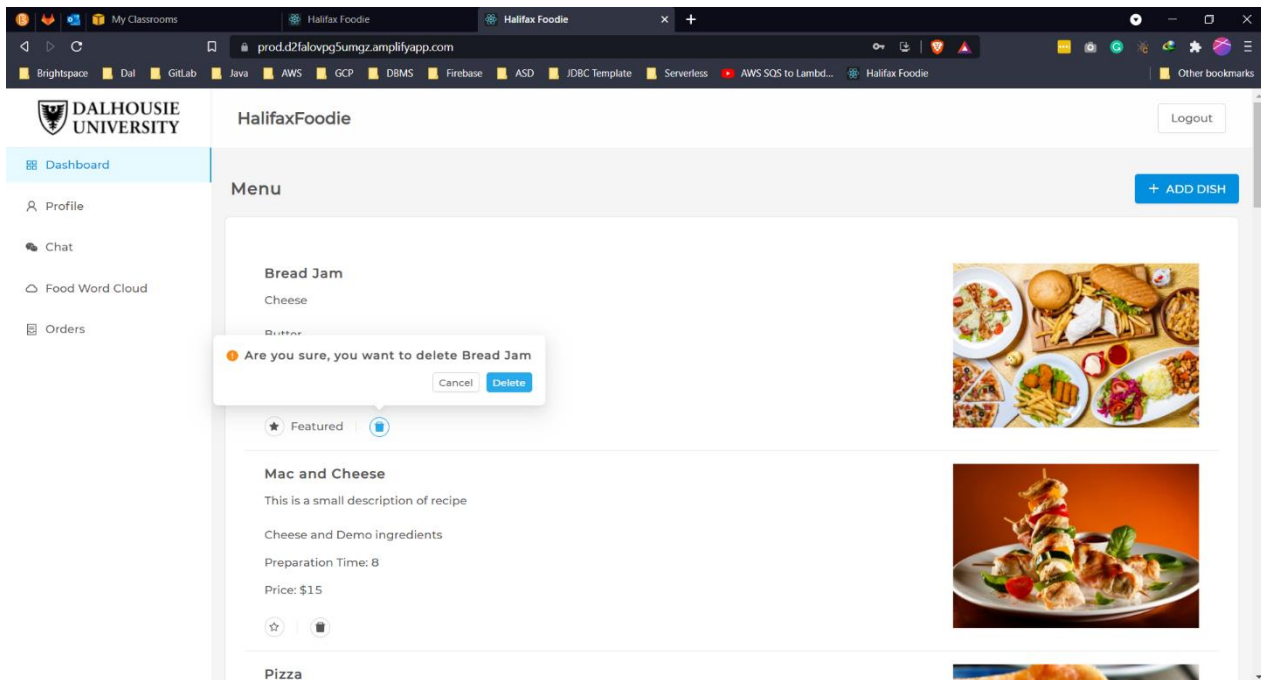


Figure 32: Delete Food Item

## 24. Adding new dish to restaurant's menu

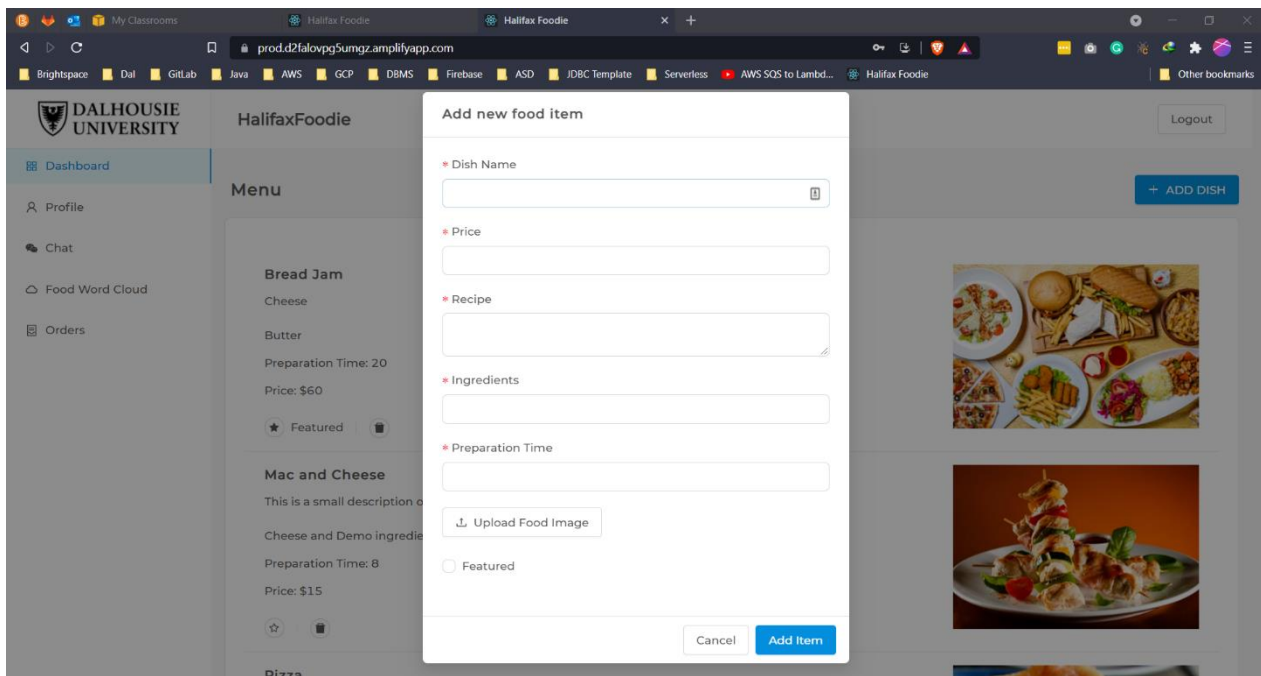


Figure 33: Add new dish to Menu



25. Orders List page for Restaurant

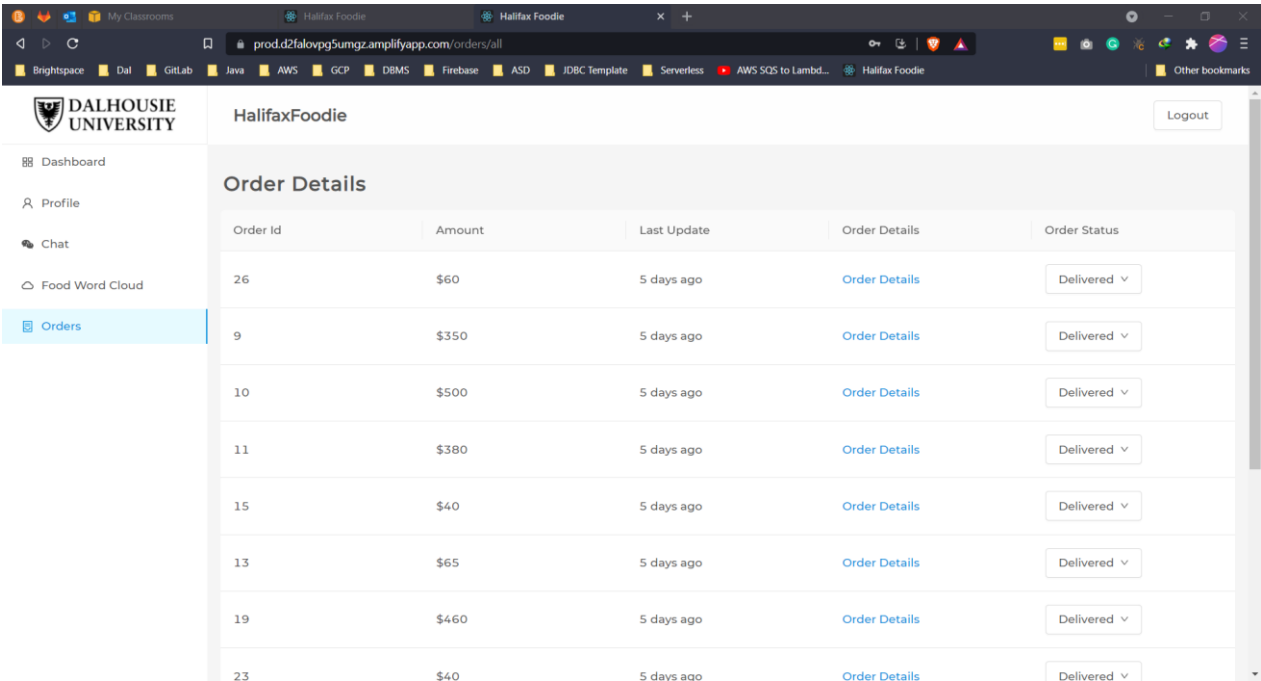


Figure 34: Order list for Restaurant

26. Changing order status of the customer

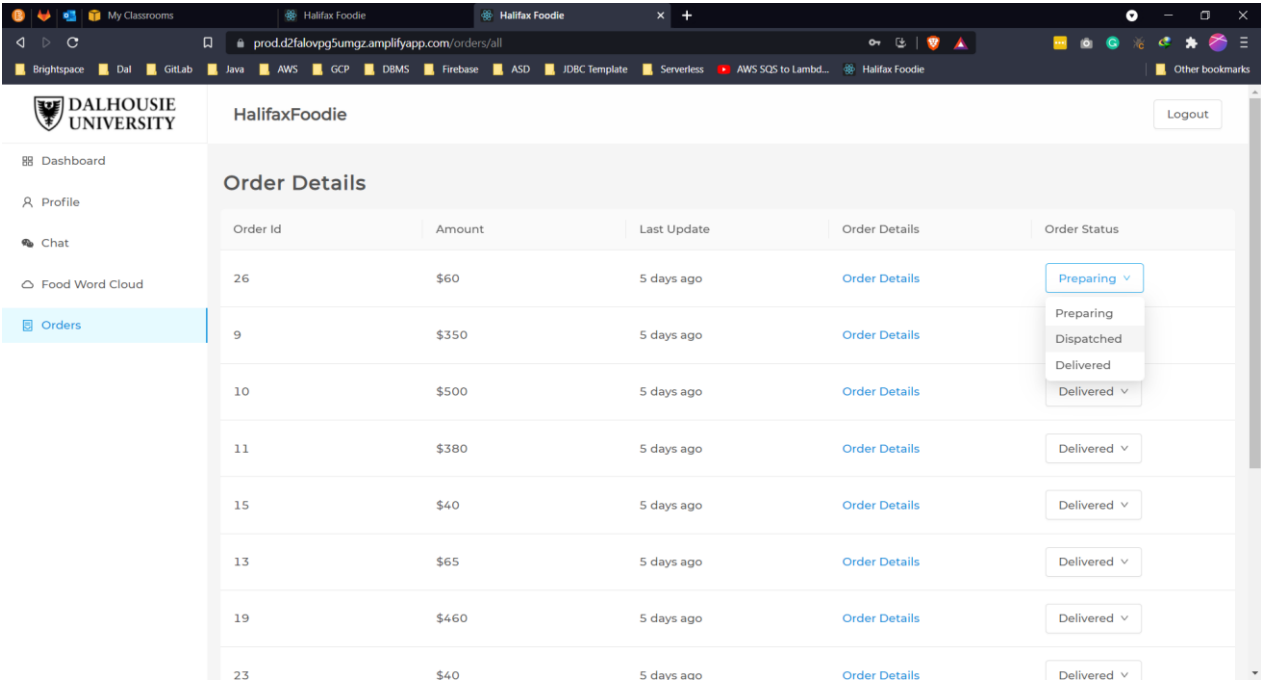


Figure 35: Feature to change order status



## 27. Order from the one restaurant at a time restriction for the customer

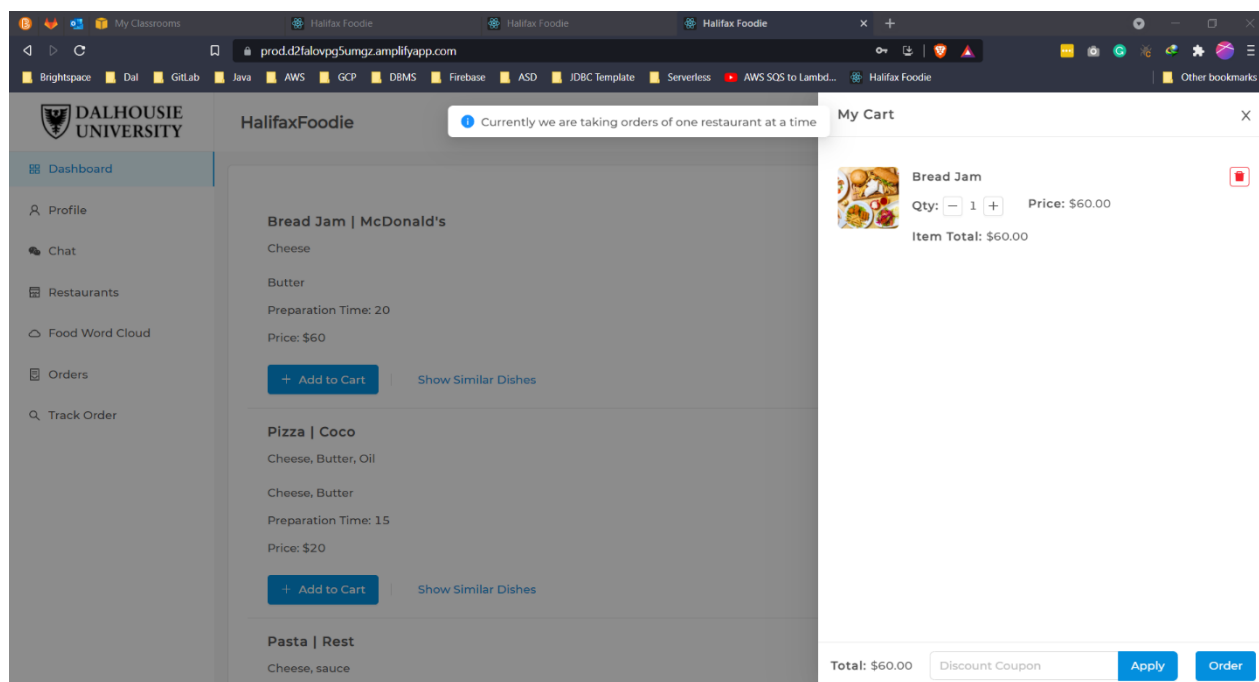


Figure 36: Order can be made from one restaurant at a time

## REFERENCES

- [1] “Amazon API Gateway | API Management | Amazon Web Services,” Amazon, [Online]. Available: <https://aws.amazon.com/api-gateway/>. [Accessed 03 07 2021].
- [2] “Cloud Functions - Google Cloud,” Google, [Online]. Available: <https://cloud.google.com/functions>. [Accessed 03 07 2021].
- [3] Google, “AutoML Natural Language documentation,” Google, [Online]. Available: <https://cloud.google.com/natural-language/automl/docs>. [Accessed 06 08 2021].
- [4] “AWS RDS,” Amazon, [Online]. Available: <https://aws.amazon.com/rds/>. [Accessed 08 06 2021].
- [5] “Build Mobile & Web Apps Fast | AWS Amplify | Amazon Web Services,” Amazon, [Online]. Available: <https://aws.amazon.com/amplify/>. [Accessed 01 07 2021].
- [6] “Google Cloud,” Google, [Online]. Available: <https://cloud.google.com/identity-platform>. [Accessed 08 06 2021].
- [7] “AWS Lambda,” Amazon, [Online]. Available: <https://aws.amazon.com/lambda/>. [Accessed 08 06 2021].
- [8] “Amazon Lex,” Amazon, [Online]. Available: <https://aws.amazon.com/lex/>. [Accessed 10 06 2021].
- [9] “Cloud Firestore,” Google, [Online]. Available: <https://firebase.google.com/docs/firestore>. [Accessed 11 06 2021].
- [10] “Amazon QuickSight,” Amazon, [Online]. Available: <https://aws.amazon.com/quicksight/>. [Accessed 13 06 2021].
- [11] “Google Data Studio,” Google, [Online]. Available: <https://datastudio.google.com/overview>. [Accessed 13 06 2021].
- [12] “Amazon ECS,” Amazon, [Online]. Available: <https://aws.amazon.com/ecs>. [Accessed 13 06 2021].
- [13] “Firebase,” Google, [Online]. Available: <https://firebase.google.com>. [Accessed 13 06 2021].
- [14] “Draw.io,” AppDiagrams, [Online]. Available: <https://app.diagrams.net/>. [Accessed 13 06 2021].

- [15] “BigQuery ML,” Google, [Online]. Available: <https://cloud.google.com/bigquery-ml/docs/introduction>. [Accessed 12 06 2021].
- [16] “Cloud Pub/Sub | Google Cloud,” Google, [Online]. Available: <https://cloud.google.com/pubsub>. [Accessed 01 07 2021].
- [17] “Word Cloud API,” Word Cloud, [Online]. Available: <https://wordcloudapi.com/>. [Accessed 07 08 2021].