CSCI 5410

SERVERLESS DATA PROCESSING

PROJECT REPORT

GROUP 4

Table of Contents

Table of Figures 2
1. Gitlab Repository Link 5
2. Hosted Web Application Link 5
3. Implementation Details 5
3.1 User Management Module [1] 5
3.2 User Authentication Module 6
3.3 Online Support Module [1] 6
3.4 Message Passing Module [1] 7
3.5 Machine Learning Module [1] 8
3.6 Web Application Building and Hosting [1] 9
3.7 Other essential modules-testing, report generation and visualizations 9
5. Pseudo Code 11
5.1 User Management 11
5.2 User Authentication (3 Stage Verification) 11
5.4 Message Passing Module 13
Tour Package Recommender: 13
Sentiment Analysis: 14
5.6 Web Application Building and Hosting 14
Room Service 14
Kitchen Service (Get Food): 15
Kitchen Service (OrderFood): **15**
Tour Service 15
5.7 Other Essential Module (Report/Visualization) 16
6. Flowchart/Activity Diagram [1] 17
7. Individual and Team Contribution 21
8. Meeting Logs 22
9. Test Cases and Evidence of Testing 26
9.1 User Login and Registration 26
9.4 Message Passing 36
9.5 Machine Learning 37
9.6 Web Application Building and Hosting 41

9.7 Other Essential Module (Report/Visualization) 64
10. Limitations 67
11. References 68

Table of Figures

Figure 1: Cloud Architecture Diagram [1][2] 10
Figure 2 Unregistered Customer Roadmap [3][4][5] 17
Figure 3 Registered Customer Roadmap [3][4][5] 17
Figure 4 Admin Roadmap [3][4][5] 18
Figure 5 Flow diagram[2] of services like Amazon Lex[10] and Lambda function[8] interacting with each other for unauthorised users. 19
Figure 6 Flow diagram[2] of services like Amazon Lex[10], Lambda function[8], and DynamoDB[7] and interacting with each other for authorised users. 19
Figure 7 Meeting 2 Log 22
Figure 8 Meeting 3 Log 22
Figure 9 Meeting log 4. 23
Figure 10 Meeting log 5 23
Figure 11 Meeting log 6 24
Figure 12 Meeting log 7 24
Figure 13 Login Authentication Failed Test Case 26
Figure 14 User Already Signed Up Test case 26
Figure 15 Password policy not matched 27
Figure 16 Caesar cipher failed. 27
Figure 17 Screenshot of chatbot successfully deploying on the website. 28
Figure 18 Screenshot of chatbot successfully deploying on the website. 28
Figure 19 . Screenshot of chatbot performing unauthorized function: room details. 29
Figure 20 . Screenshot of chatbot performing unauthorized function: navigation/help. 29
Figure 21 Screenshot of chatbot performing unauthorized function: navigation/help 30
Figure 22 Screenshot of activation of authorized chatbot after login. 31
Figure 23 Screenshot of chatbot performing authorized function: order food. 31
Figure 24 Screenshot of chatbot performing authorized function: order food with confirmation message. 32
Figure 25 Screenshot of chatbot performing authorized function: book room. 32
Figure 26 Screenshot of chatbot performing authorized function: book room with confirmation message. 33
Figure 27 User Notifications 36
Figure 28 Booking Confirmation Email 36
Figure 29 Notifications in Table 37
Figure 30 Tour Package Recommender Front End 37

Figure 31 Tour Package Recommender Front End 38
Figure 32 Tour Package Recommender Results 38
Figure 33 Tour Package Recommender Cloud Function 39
Figure 34 Tour Package Recommended Vertex AI 39
Figure 35 Sentiment Analysis Front End 40
Figure 36 Sentiment Analysis Results in AWS DynamoDB 40
Figure 37 Sentiment Analysis Cloud Function 40
Figure 38 Room booking home page 41
Figure 39 : User enters the room booking details 41
Figure 40 Room booked successfully message 42
Figure 41 Validations for the fields in the room booking form 42
Figure 42 Validations for the fields in the room booking form 43
Figure 43 Room booking details in the Amazon DynamoDB table 43
Figure 44 Tour package home page for user 44
Figure 45 Tour package home page for user 44
Figure 46 Booking confirmation check for the user 45
Figure 47 Booking date confirmation for the user 45
Figure 48 Data inserted in Amazon DynamoDB table for Tour booking 46
Figure 49 Food Order home page 46
Figure 50 Order confirmation details form for the user 47
Figure 51 Validations in the order food form 47
Figure 52 Food order details stored in Amazon DynamoDB table 48
Figure 53 Lambda Functions for all the modules 48
Figure 54 API created in AWS for all the modules 49
Figure 55 API gateway trigger added for room booking backend lambda function 49
Figure 56 GET, POST, PUT and DELETE API requests for the routes 50
Figure 57 Integrations of the requests with AWS Lambda function 50
Figure 58 API gateway trigger for food order lambda function 51
Figure 59 Routes for the food order lambda function API's 51
Figure 60 Integrations of the requests with AWS Lambda function in food order 52
Figure 61 API gateway trigger for the Tour Booking lambda function 52
Figure 62 Routes for tour booking 53
Figure 63 Integrations of the requests with AWS Lambda function in tour booking 53
Figure 64 Caesar Cipher Lambda function and API gateway trigger 54
Figure 65 API details of Caesar cipher lambda function 54
Figure 66 Integrations of the requests with AWS Lambda function in caesar cipher 55
Figure 67 feedback lambda function 55
Figure 68 orderMyfood lambda function 56
Figure 69 available rooms lambda function 56
Figure 70 API gateway trigger for security answer lambda function 57
Figure 71 Integrations of the requests with security answer lambda function 58
Figure 72 Website navigation lambda function 58
Figure 73 room booking lambda function 59
Figure 74 Active tables in DynamoDB 59

Figure 75 Caesar Cipher table in DynamoDB 60
Figure 76 User feedback table in dynamodb 60
Figure 77 Food details table in dynamodb 61
Figure 78 Food order table in dynamodb 61
Figure 79 room details table in dymaodb 62
Figure 80 roomdetails table in dynamodb 62
Figure 81 security answer table in dynamodb 63
Figure 82 Tour booking details table in dynamodb 63
Figure 83 Room Bookings 1 64
Figure 84 Room Bookings 2 64
Figure 85 Food Orderings 1 65
Figure 86 Food Orderings 2 65
Figure 87 Tour Bookings 1 66
Figure 88 Tour Bookings 2 66

1. Gitlab Repository Link

Group 4 GitLab Repository

2. Hosted Web Application Link

Serverless B&B

3. Implementation Details

3.1 User Management Module [1]

- The main functionality of this module is to handle the user registration process along with dynamically assigning the User to the specific room booked by the user. The AWS Cognito [7] will be used for authentication, authorization, and user management. The users will be able to sign in using their username and password. After this step, the user will be asked to setup the security questions followed by the Caesar Cipher key. This key will be generated randomly, and the user is supposed to remember the key which will be useful at the time of login. The user pool and Identity pool will be used together to provide the essential services for the application. Furthermore, we will be also securely storing and maintaining the user details. AWS DynamoDB [8] service will be utilized to store the information of these users. The application initially will display the home page where the user will either log in to the application or can register themselves.

- If the user chooses to register themselves then the application will ask the user for details like Username, Phone No, Email ID (It will be a unique User ID), Password, Security Questions, and the Caesar Cipher Key for encryption of the text. There will be on-the-go validations done for the details entered by the user like for a valid email or for a strong password.

- User registration logic for Serverless B&B application:

1. According to the requirements, when the user enters the username, email and other details, the next section will be followed by the Security Questions which the user has to remember at the time of login. Along with that, the final part which will be require for authentication is the Caesar cipher key which will be randomly generated by the system. The user is supposed to remember this key which the user will be using to encrypt the plain text that the system generated at the time of login. When the plain text is generated in the third part of authentication, then the user will encrypt the plain text with the randomly generated key and perform Caesar cipher on the text. If the entered details match with that in the database, then the user will be able to successfully login and access the application to order food, book rooms and even book for tours.

- Now, all the information entered by the User, such as the username, email and password, security questions and the Caesar cipher key along with the information regarding the room bookings, food orders and the tour bookings that are provided by the user is stored securely in the AWS DynamoDB [8].

3.2 User Authentication Module

Users must successfully complete all three phases of authentication before being granted access to any of the services offered by the B&B application, and the user authentication logic uses a multi-cloud paradigm to enable multi-factor authentication for registered users.

1. **ID and Password Validation (AWS Cognito):**

   The first step requests the user's login email and password, then checks it against the information in AWS Cognito.

2. **Question and Answer verification (AWS Lambda):**

   The second stage displays the user's chosen question from the two options they were given on registering and checks the response. The user-selected query and response are saved in AWS DynamoDb, and the API calls the Cloud Function to validate the outcomes before proceeding to the third phase of authentication.

3. **Caesar Cipher Validation (AWS DynamoDB):**

   The user will be required to utilise the key that was generated and given to them when they registered to decrypt the Caesar-ciphered text as the last step in the authentication process.

3.3 Online Support Module [1]

Online assistance is a web-based technique for offering customer service. It is currently more popular than calling or going in person to communicate with businesses. Online support is a part of customer relationship management. The best feature of online support is the round-the-clock customer service it provides. The consumer can instantly get in touch with the agent if there is a problem or if they have any questions about the service or product.

The module's responsibility is to provide online help using Amazon Lex to both authenticated and unauthenticated users [10]. Lex will assist with developing clever comments to the recorded key words. To answer accurately and assist consumers in the best way possible, the service will offer a range of user contact scenarios.

As per the project specifications, two bots are created:

1. **BnbAuth:**
This bot will be for authorized users, i.e., activated after user has registered/login. This bot will help in booking room by taking all the details like email, arrival date, departure date, room numbers and finally gives a confirmation message of booking the room with message with all the details.

2. **BnbUnauth:**
This bot will be for unauthorized users, i.e., users who have not yet logged into the website portal. It will help in navigating through the website by giving the options like login, signup and more by giving direct links to the pages. Also, it will help to get details about any room like what kind of rooms the website offers.

For deploying the bot into the application, an amazon deployment document [16] was followed that involved creating Cloud Formation [3] stack using the available template. The formation had a snippet code which was added in the index.html for successful deployment of IU for chatbot.

3.4 Message Passing Module [1]

Giving authorized users access to the hotel management i.e. food ordering and room booking, and tour operators via this message passing module is helpful. Using the service, Pub/Sub, provided by GCP, we can build systems of event producers and customers, known as publishers and subscribers. Without considering how or when these events should be processed, publishers transmit events to the Pub/Sub service. Then, Pub/Sub distributes events to all the services that must respond to them. As a result, if employed as a messaging service, this service can be incredibly beneficial and effective.

In the Serverless B&B application, whenever the users use services such as room booking, food ordering or tour booking, the front-end, hosted on GCP CloudRun, will send a request to the Lambda function to store the details of the order in the DynamoDB table. Once the successful response comes to the application, it will send another request to the GCP Cloud Function which passes the message to the GCP Pub/Sub service. Whenever there's a message available in the Publisher queue, the Subscriber service would fetch the message from the queue and send it to the AWS Lambda function. This Lambda function will create a confirmation message and store it in the notification table in the DynamoDB database to store the user-specific notifications. After storing the notification in the table, the Lambda function would send an email containing the order details to the registered user's email address.

3.5 Machine Learning Module [1]

To identify the similarity of stay duration of customers and propose a tour package.

Google cloud platform offers various Machine Learning services which club into Vertex AI service [6]. To implement this module, we will use Vertex AI's Auto ML services [6]. Auto ML provides various methods to train, test, and evaluate the model. From the choice of many, we will be using the Tabular form of data to identify the suitable tour package and recommend it to the customers [6]. We will be using a classification algorithm to train the model. The process of developing this module is described below:

- We will create the dataset that will contain information on the stay duration of customers, and we will also set the pipeline that will continue updating the database with new entries about customers' stay duration.
- Clean that database and modify it to best fit the model.
- Apply classification algorithm on Vertex AI's Auto ML service [6].
- Train the model with the training dataset.
- Test the model with the test dataset and evaluate the performance by considering the Confusion matrix and Feature importance matrix [6].
- Generate the Endpoint to access the trained model and do the prediction.
- By entering the data into the model, we will receive a prediction from the model that will tell us what package is suitable for the customer based on their stay duration [6].

To identify the polarity of customer feedback and to add appropriate scores.

We will be using Vertex AI's Auto ML service to build this module [5]. Here, we must calculate the polarity of customer feedback with its score of it [5]. Therefore, we need to use the Text format of the data. We can use Text format with Auto ML and can build the machine learning model which takes text data and can give the polarity of customer feedback [5]. The implementation of this module is described below:

- We will create a Text dataset by adding data from various users to our application's database and collecting those data into one directory [4].
- After collecting the data, we have to label all the text files and divide them based on the polarity of the feedback [4].
- Cleaning of the data by removing stopwords, and other less important details from the data files.
- Build the model by using Sentiment Analysis on customer feedback data [4].
- Train the model using the training dataset.

- Test the model by using a test dataset with various matrix that defines the accuracy of the model.
- Generate the endpoint from which we can predict the result of the input data [1].

3.6 Web Application Building and Hosting [1]

We are going to use ReactJS for the front-end development and NodeJS with different Cloud Function as well as Lambda Function for the API connection as a back-end environment. Users will interact with the web application via the front-end module [4][5]. A good choice of front-end and back-end technologies is necessary when developing dynamic web applications [4][5]. Therefore, to achieve application flexibility and scalability, we have chosen to design the front-end using React JS. To access the functionality of GCP Cloud function services on the back end, we will be using Express JS, Node JS, and Lambda services [2][3]. This web application will be hosted in Google Cloud Run via a docker image.

3.7 Other essential modules-testing, report generation and visualizations

For our application, we ran validation tests and function tests. Username-password checks, operations that can only be carried out by authorised users, whether a guest user can carry out tasks that are not assigned to them, whether the chatbot can provide the users with the relevant information they are looking for, whether the users are notified in the appropriate situations, whether the users can obtain specific booking ids and order ids when booking the room and ordering the food, and many other validations and tests. All potential test cases necessary for the application to operate effectively are included in the testing module.

For the report generation module, the AWS QuickSight service is used, which comes with business intelligence dashboards and applications, for the report generation module. Reports on user login and access statistics are generated by this module. The report can be generated by the application's authorized users. Each user activity will be recorded using the AWS CloudWatch service. The Lambda function will store all the user login activities in the DynamoDB database from AWS CloudWatch. The QuickSight service receives the user activity data in order to produce the reports.

To visualize the statistical data reports, we have used GCP Data Studio, BigQuery, AWS Lambda functions, GCP Cloud functions, AWS S3 and GCP CloudStorage. We have used the Lambda function to extract the data from the DynamoDB tables and store it in the AWS S3 buckets. For example, to generate the room booking visualization, the function will extract the room booking data from DynamoDB and store it in CSV format in the S3 bucket. Another function will send this file to CloudStorage. Whenever the data arrives in the CloudStorage bucket, a cloud function will be triggered. This cloud function will extract the file from the GCP bucket and create a data set and table in BigQuery for report visualization. From the BigQuery service, the data is fetched into the GCP Data Studio to visualize in different forms such as tabular forms, pie charts, donut charts etc.

4. Cloud Architecture Diagram [1]

Figure 1 depicts the cloud architecture of the application. The diagram is designed using draw.io [2] by gathering knowledge from sources Google Cloud Documentation [2], and AWS Documentation [3]. As we can see in the figure, the users will interact with the front-end of the application developed in ReactJS. The front-end of the application is hosted on GCP Cloud Run via the docker image which is stored in the Container Registry of GCP. AWS API gateway is used to navigate the requests from various services to AWS Lambda and GCP API Gateway is used to direct the requests to the GCP Cloud Functions. Amazon DynamoDB is used for the storage of data for the application [1].

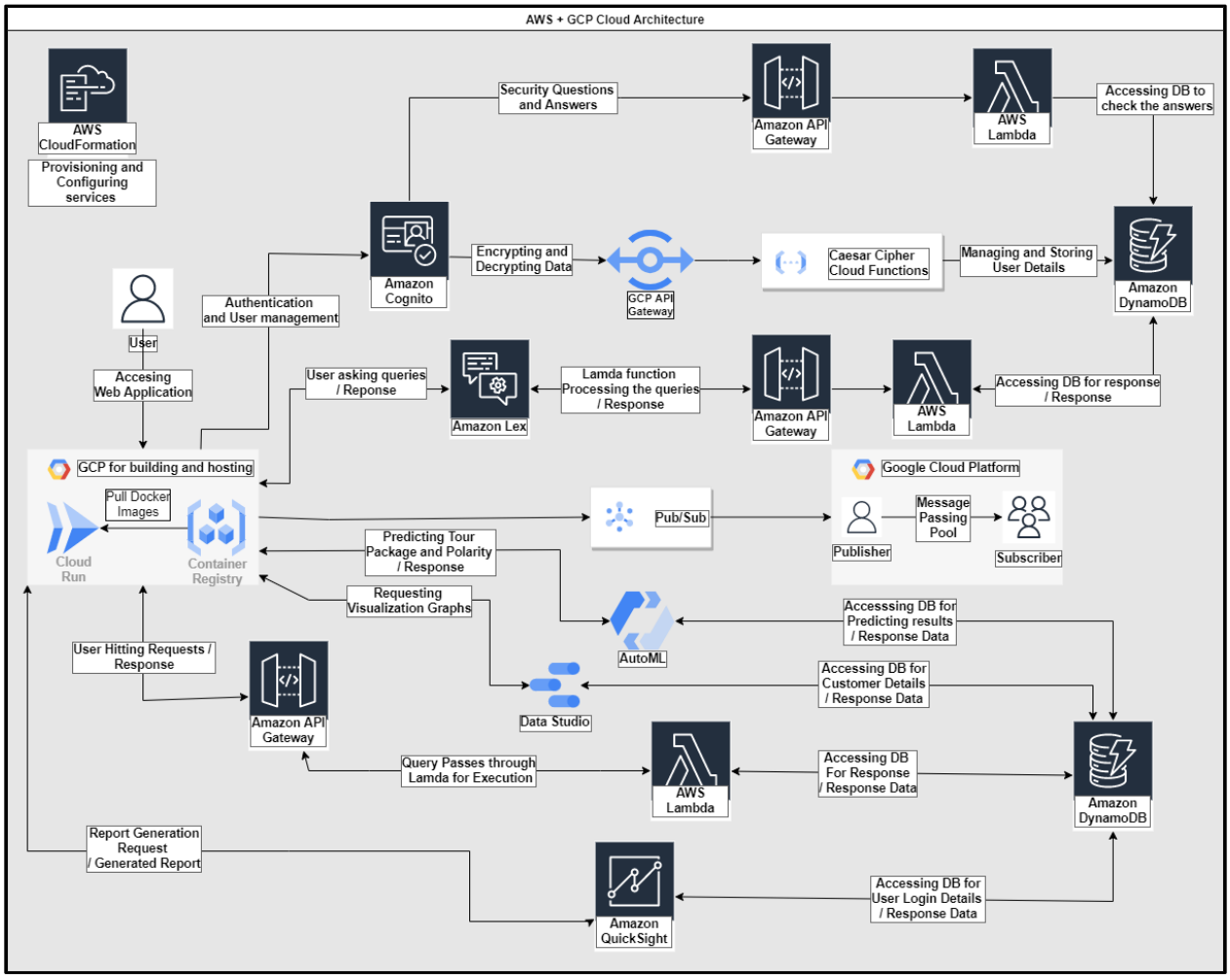

Figure 1: Cloud Architecture Diagram [1][2]

5. Pseudo Code

5.1 User Management

1. User enters the username, email, password
2. The entered username, email and password is checked for validations and for proper format for email and password.
3. If the format is wrong, the system throws an error; If not, the user is proceeded to the next phase.
4. In the next step, the user enters the answers for the security questions asked by the system.
5. The user must remember the answers to the questions required when they try to login.
6. In the next step, the system generates a key for the Caesar cipher verification.
7. The user is supposed to remember the key generated by the system.
8. When all the details are correctly entered by the user which match the requirements of the system, the user is successfully registered to the system.
9. The registered data is stored in the database.
10. If the details are not valid and do not follow the system requirements, then the user is thrown an error and redirected back to the registration page.
11. On successful registration the user is redirected to the login page.
12. If the user is unregistered, then also the user can access the system
13. In case of unregistered user, the user will get minimum access wherein the user can access a smaller number of features in the application.

5.2 User Authentication (3 Stage Verification)

1. Request the person's email address as both their username and password.
2. Call the AWS Cognito API to verify your credentials.
3. If the credentials are wrong, go back to step 1 or continue to step 4.
4. Request the user to enter their security question and response.
5. If the response to the security questions is wrong, go back to step 4; otherwise, continue.

6. Request that the user use the registration-generated key to decrypt the Caesar cypher text.

7. Retry step 1 if the user is unable to decrypt the text. Otherwise, direct the visitor to the application's main page.

5.3 Online Support Module:

Pseudo code for Lambda functions:

bookMyRoom.js

1. First, it will scan through the ‘roomDetails’ table where information related to room is stored and roomNumber is stored as the key.
2. After the taking all the inputs from the user in lex intent(s) [as it is set as fulfilment trigger for BnbAuth’s BookRoom intent], it will record all the data and combine data of the intent and step 1 as an ‘item’.
3. This ‘item’ will be passed as an argument to putItem and the details will be recorded into ‘room’ table where all the room booking information resides.\

orderMyFood.js

1. First, it will scan through the ‘foodDetails’ table where information related to food is stored and foodName is stored as the key.
2. After the taking all the inputs from the user in lex intent(s) [as it is set as fulfilment trigger for BnbAuth’s OrderFood intent], it will record all the data and combine data of the intent and step 1 as an ‘item’.
3. This ‘item’ will be passed as an argument to putItem and the details will be recorded into ‘foodorder’ table where all the food order information resides.

navigateWebsite.js

1. This function will just scan the input by the user in the ‘NavigateWebsite’ intent and send the message i.e., the link of the page user desired from the menu in chatbot UI.

5.4 Message Passing Module

1. User clicks on room booking/food ordering/tour booking.
2. System makes a request with required data to Lambda function.
3. Lambda function stores the data in DynamoDB and sends back the success message.
4. System makes another request with data to Cloud Function.
5. Cloud Function publish the message to the relevant topic.
6. Subscriber service fetches message from the topic queue and pass it to AWS Lambda function.
7. Lambda function prepares notification message and store the notification to a DynamoDB table.
8. Once the data is stored, the Lambda function sends email to the user.
9. User navigates to notification page.
10. System makes a call to Lambda function.
11. Lambda function fetches the user's notification from DynamoDB and sends to the application.
12. System displays the notifications.

5.5 Machine Learning Module

The below implementation has been done by referring the official documentation provided from sources [13][14][15].

Tour Package Recommender:

1. User clicks enter data from the front-end application and post request will be hit to the cloud function.
2. Cloud function will be invoked and fetch the request json body and get all the parameters.
3. Cloud function will hit the machine learning model's endpoint and send the instance_dict to that model to get the prediction.
4. Here, function will get all the prediction and store them in dictionary format.
5. Form here, function will determine which class has the best score and return that result to the front-end.
6. Result of tour recommendation model will be displayed to the user.

Sentiment Analysis:

1. User enter the feedback from the front-end.
2. After hitting the submit the cloud function will be invoked.
3. Cloud function get the text data from the requet JSON body.
4. Cloud function calls Cloud Natural Language Processing library provided by GCP and fetch the sentiment score of the text.
5. Cloud function save that score and add the sentiment score to the dynamodb table.
7. Return with the 200 status.

5.6 Web Application Building and Hosting

Room Service

1. User clicks on room booking then a list of rooms which are available will be displayed on the page.
2. Once the user selects a particular room and book it then that room with the given user id will get stored in the dynamodb.
3. When the user clicks on the book button then a call wil be made to the API Gateway as different routes/endpoints are assigned in the roomapi on the aws.
4. Each API endpoints are integrated with the Lambda function and the lambda function will post the data in the database.
5. User clicks on the book button.
6. API endpoint will be hit and a dedicated function will be called.
7. At the successful storing of user information with room info and date from which the reservation is made.
8. If everything is ok then room booked successfully message will be seen on the website.
9. Once the room is reserved then that room will be deducted from the available room database.

Kitchen Service (Get Food):

1. Loading the NodeJS AWS-SDK.
2. Using aws.config.update to configure AWS credentials.
3. Making a service object for DynamoDB.
4. Create a JSON object with the parameters needed to search for items in the table, such as TableName: "getfood"
5. Calling the procedure getMenu while supplying a JSON object
6. In particular Function
    a. Calling the scan method of the DynamoDB service object

    b. Creating a JSON menu string by iterating the response.

    c. Give the menu back.

7. In response to an API request, returning the menu.

Kitchen Service (OrderFood):

1. Loading the NodeJS AWS-SDK.

2. Using aws.config.update to configure AWS credentials.

3. Making a service object for DynamoDB.

4. Getting order items from a JSON request

5. Iterating the order items to figure out how long it takes to prepare an order.

5. Making an order ID.

7. Extracting order information from the JSON request.

8. Making a JSON object with the parameters required to store the object in DynamoDB.

9. Call the function that orders orders from DynamoDB.

Tour Service

1. User clicks on tour booking then a list of tours which are available on the season will be displayed on the page.

2. The following images of tour and its information will be get or displayed using the getapi which takes information from DynamoDB.

3. When the user clicks on the book button then a call will be made to the API Gateway as different routes/endpoints are assigned in the tourapi on the aws.
4. Each API endpoints are integrated with the Lambda function and the lambda function will post the data in the database.
5. User clicks on the book button.
6. Api endpoint will be hit and a dedicated function will be called.
7. At the successful storing of user information with tour info and date from which the reservation is made.
8. If everything is ok then tour will be booked successfully and a message will be seen on the website.

5.7 Other Essential Module (Report/Visualization)

1. Lambda functions checks periodically and extracts data from DynamoDB to store it in AWS S3.
2. Another function sends the extracted data from S3 to GCP Cloud Storage.
3. Cloud function gets triggered as the data arrives in GCP Cloud Storage.
4. Cloud function takes data from storage and creates a dataset and table in the BigQuery.
5. BigQuery service performs query on the generated table.
6. GCP Data Studio takes data from BigQuery.
7. Data Studio generates visuals from the data.
8. User clicks on the visualisation.
9. The system displays the visualisations created by the Data Studio.

6. Flowchart/Activity Diagram [1]

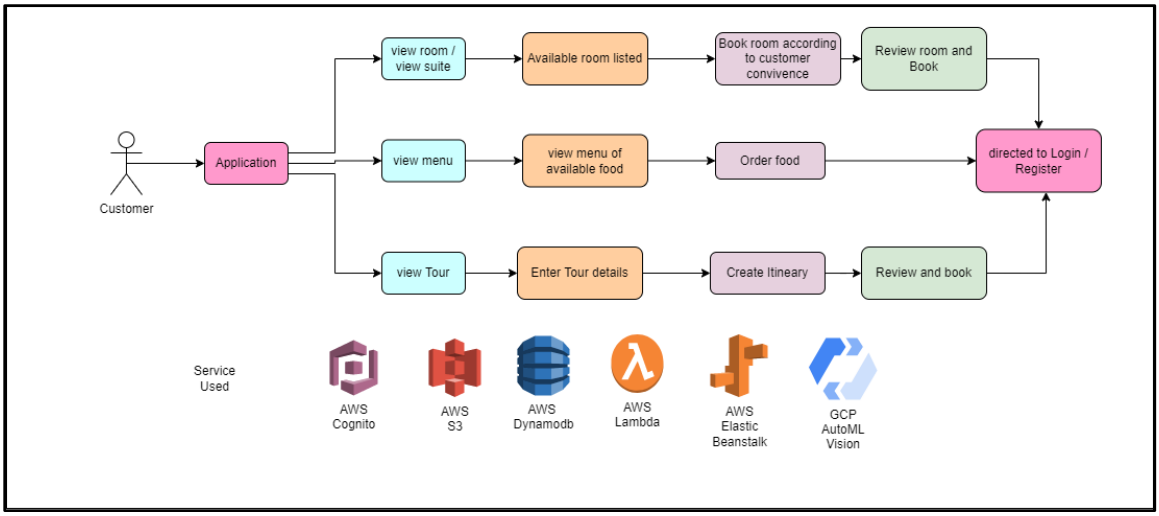

Figure 2 Unregistered Customer Roadmap [3][4][5]

- **Non-registered users**

  Users who are not signed up will be able to browse rooms, menus, and tours. The user will see all the available rooms and their specifications when they access the rooms. The user will be sent to the login/registration page to proceed when attempting to select and book the room [4][5]. The menu can also be viewed by the user. When a user examines the menu, all the available breakfast options are displayed along with information like pricing. Additionally, the user can view recommended tours by selecting the view tours option. It will request the user's stay time and then display results based on how closely that duration matches the user's input [4][5].

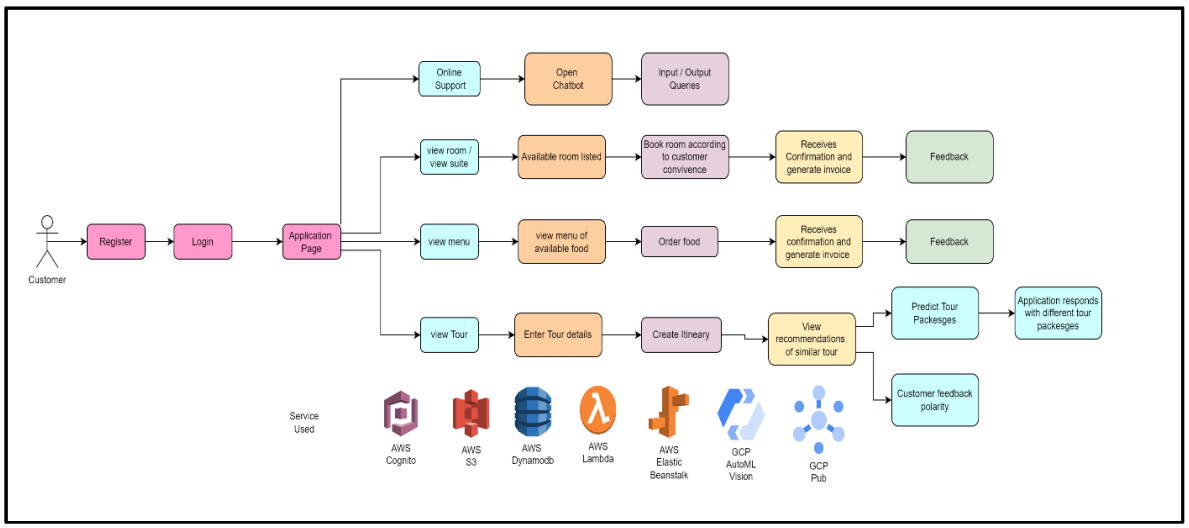

Figure 3 Registered Customer Roadmap [3][4][5]

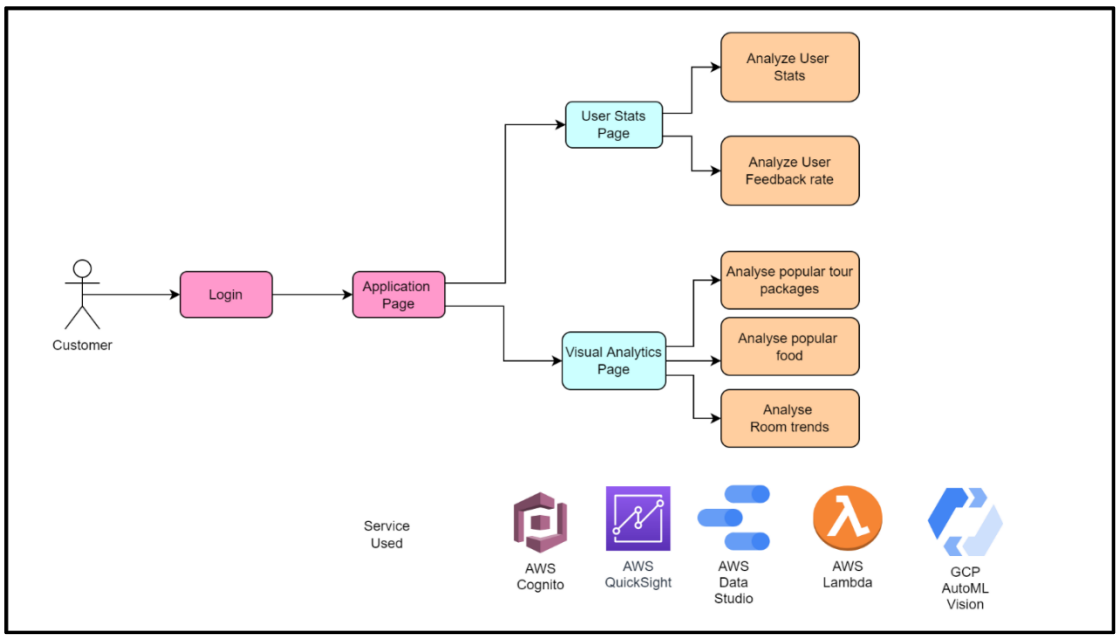

Figure 4 Admin Roadmap [3][4][5]

- **Registered users**

  The register is chosen by the user from the navigation bar. They answer security questions and input their login and password. After submitting the form, Cognito will be used by a Lambda function through an AWS gateway to register the user [4][5]. The responses to the security question will also be kept in DynamoDB by Lambda. Additionally, upon successful registration, they will receive a random number that will be saved in DynamoDB [4][5].

  Once the user is registered then he/she does not have to sign up again as data is already present in the DynamoDB so the user will sign in the next time he /she is using the application [4][5]. Now we will illustrate different scenarios of the interaction of users with the services available on our application [4][5].

I. User interaction with a chatbot:

The user will be able to communicate with a chatbot after logging in. Users can use this bot to search for available rooms and browse the website [4][5]. Additionally, it will give them suggestions and useful guidance for booking reservations and exclusive placing meal orders [4][5].

- **Unauthorised users:** The flow diagram below shows how various services run and communicate with others in cloud environment for BnbUnauth bot.

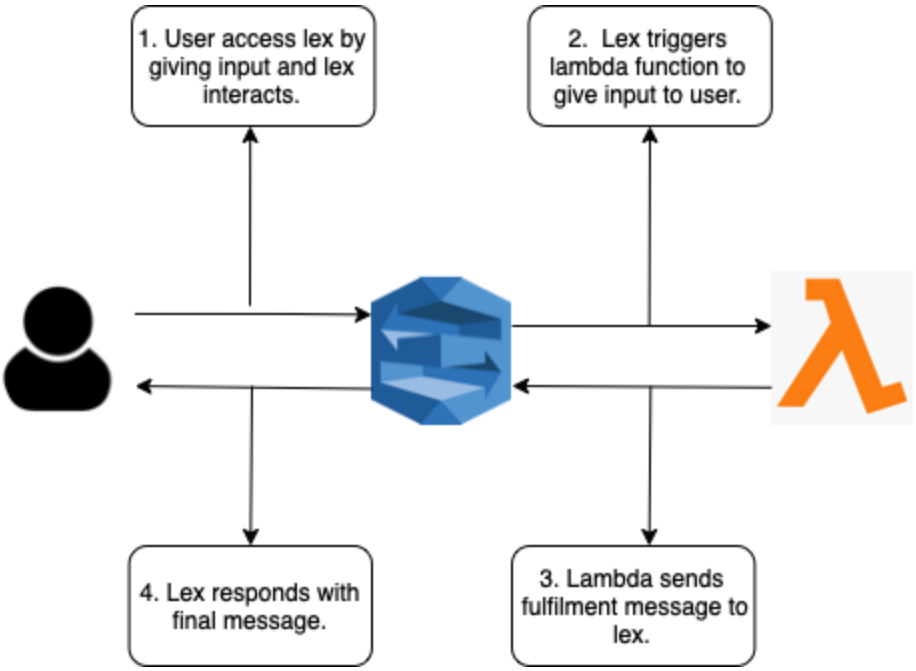

Figure 5 Flow diagram[2] of services like Amazon Lex[10] and Lambda function[8] interacting with each other for unauthorised users.

- **Authorised users:** The flow diagram below shows how various services run and communicate with others in cloud environment for BnbAuth bot.

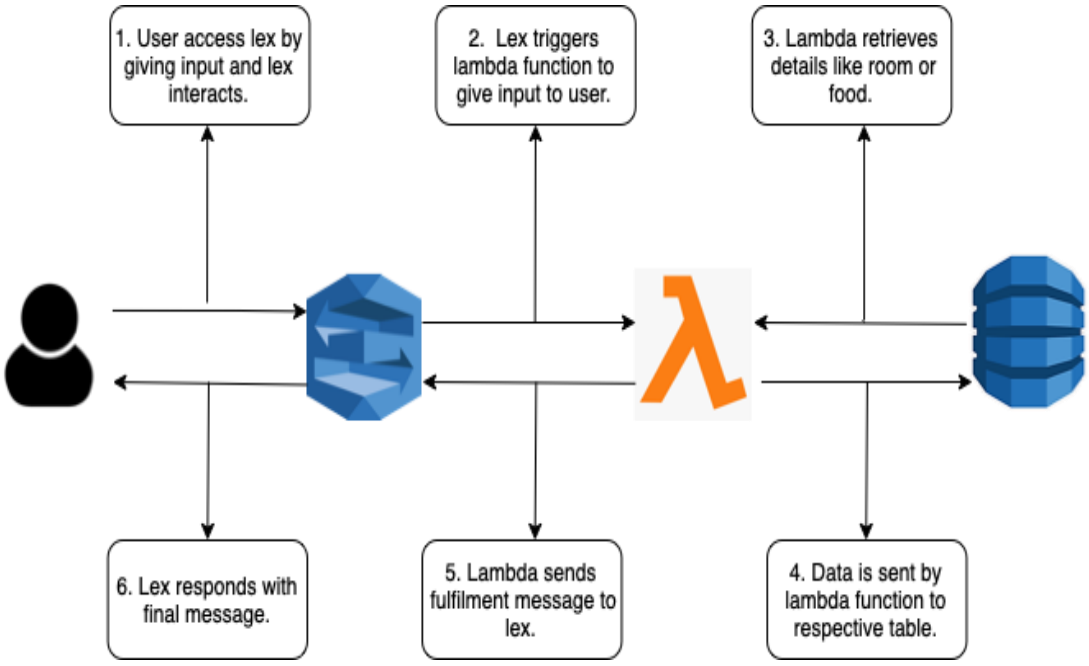

Figure 6 Flow diagram[2] of services like Amazon Lex[10], Lambda function[8], and DynamoDB[7] and interacting with each other for authorised users.

II. User booking a room:

The chatbot will let the user book a room or a bed as they interact with it. Once the user selects the option they want to use, this will happen automatically. Once the confirmation is received, the information about the reservation will be updated in DynamoDB [4][5]. After the reservation, the consumer will be asked for feedback. Sentiment analysis will be done on the feedback, which will be stored in DynamoDB [4][5].

III. User ordering food(breakfast):

Users will be able to examine a pre-defined menu that includes information on the breakfast alternatives that are offered, as well as their specifics and cost. Users will be able to choose from the menu and place meal orders. After a successful order, DynamoDB will save the order data, and the user will get an order confirmation notification [4][5]. They will also be able to see the order's current status, including whether it is being placed, prepared, ready, or delivered. Invoices are produced and saved to S3 buckets after orders are delivered [4][5].

IV. User viewing suggested tours:

Following the completion of the order, the customer will be requested to provide feedback [4][5]. This feedback will be kept in DynamoDB and based on the data, sentiment analysis will be performed to determine if the feedback was good or negative [4][5].

7. Individual and Team Contribution

Module name	Team member
User Management Module	Adarsh Kannan Iyengar
User Authentication Module	Meet Patel
Online Virtual Assistance Module	Ruhi Tyagi
Message Passing Module	Kunj Patel
Machine Learning Module	Fenil Parmar
Room Booking, Tour Booking Module	Jenish Patel
Web Application building and hosting	All Team Members
Other Essential Module	Kunj Patel
Reports and Documentations	All Team Members
Demo Video	All Team Members

8. Meeting Logs

Meeting 1

Date: 17th May 2022

Agenda: Team Introduction

Meeting 2

Date: 25th May 2022

Agenda: Discussion about the project specifications and getting to know the requirements.

Figure 7 Meeting 2 Log

Meeting 3

Date: 29th May 2022

Agenda: Task distribution and discussion regarding the cloud services that are to be used in the project.

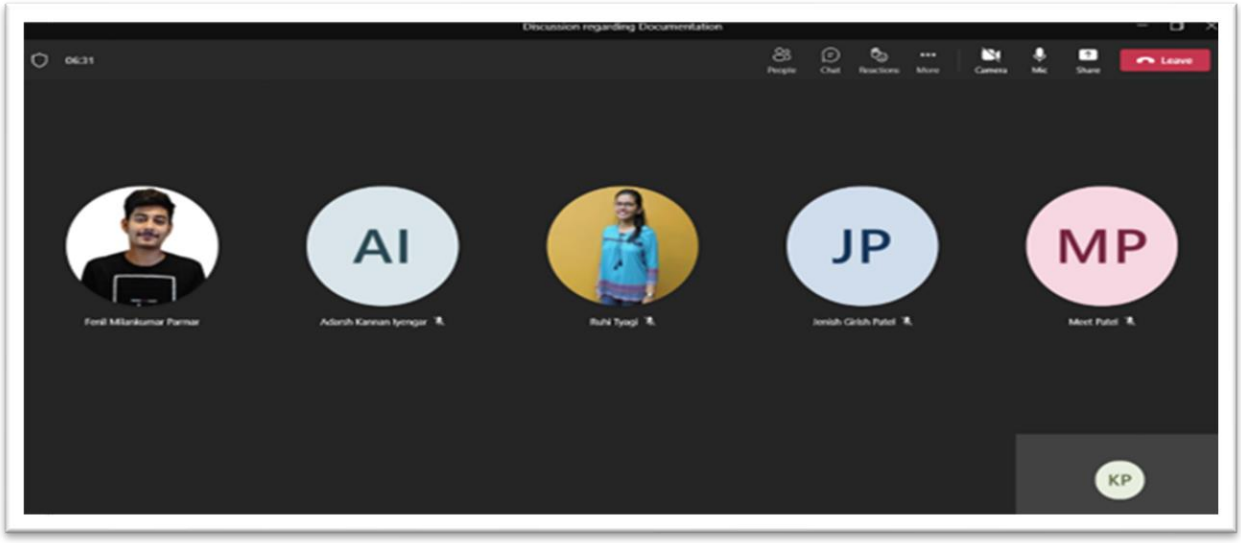

Figure 8 Meeting 3 Log

Meeting 4

Date: 1st June 2022

Agenda: Reviewing the conceptual design report

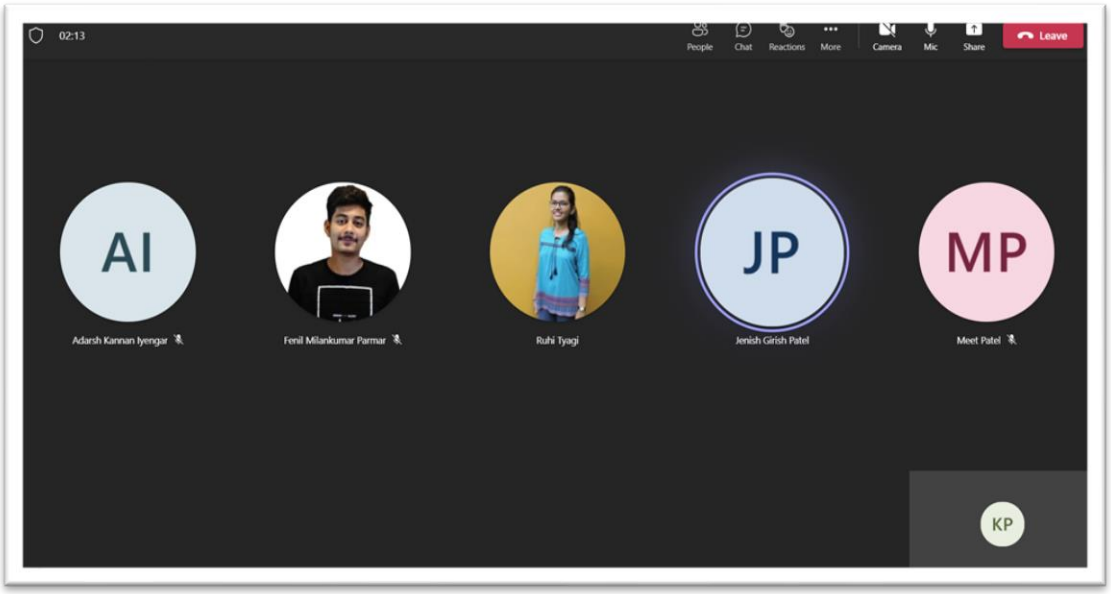

Figure 9 Meeting log 4.

Meeting 5

Date: 28th June 2022

Agenda: Discussion regarding conception report feedback and dividing the modules amongst the team members.

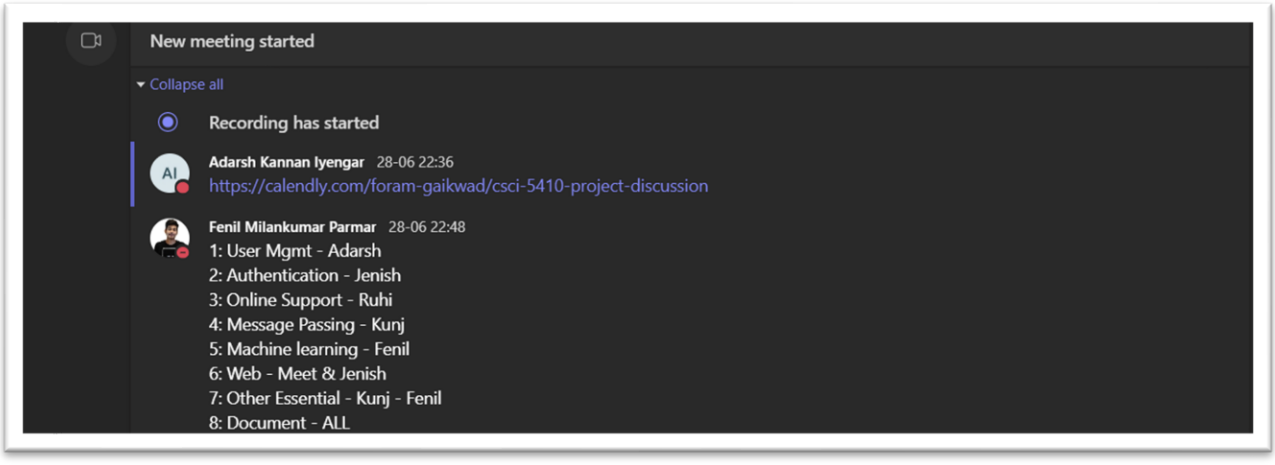

Figure 10 Meeting log 5

Meeting 6

Date: 4th July 2022

Agenda: Discussion regarding the design document and dividing the documentation tasks amongst the team members.

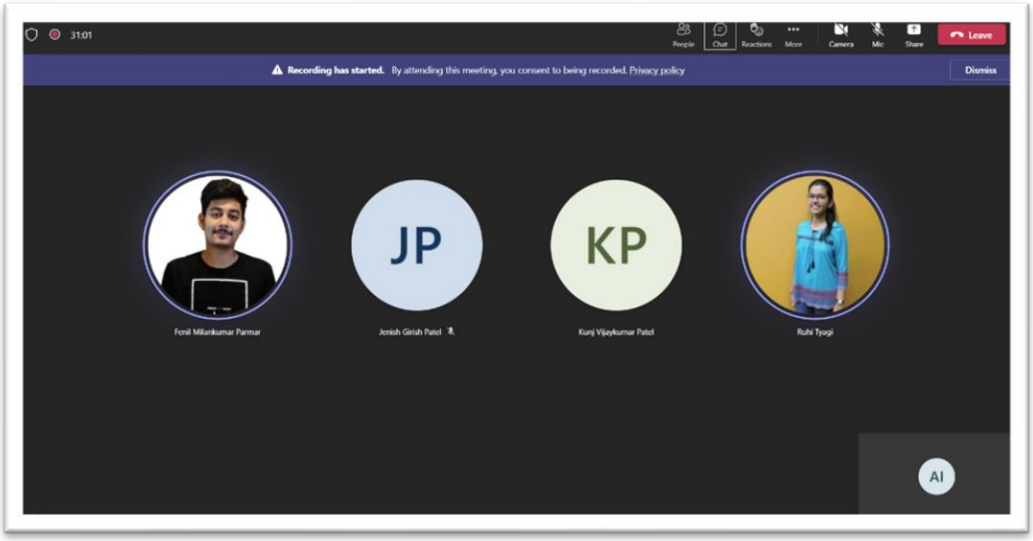

Figure 11 Meeting log 6

Meeting 7

Date: 5th July 2022

Agenda: Final meeting regarding the design document formatting and submission

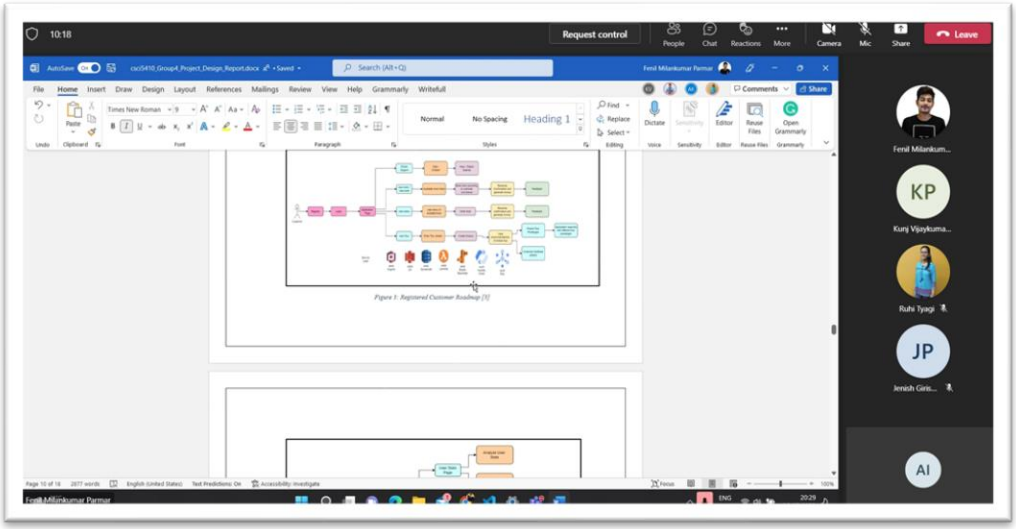

Figure 12 Meeting log 7

Meeting 8

Date: 18th July 2022

Agenda: In Person Meeting regarding project implementation and integration

Meeting 9

Date: 23rd July 2022

Agenda: In Person meeting regarding demo video and final report documentation

9. Test Cases and Evidence of Testing

9.1 User Login and Registration

Figure 13 Login Authentication Failed Test Case

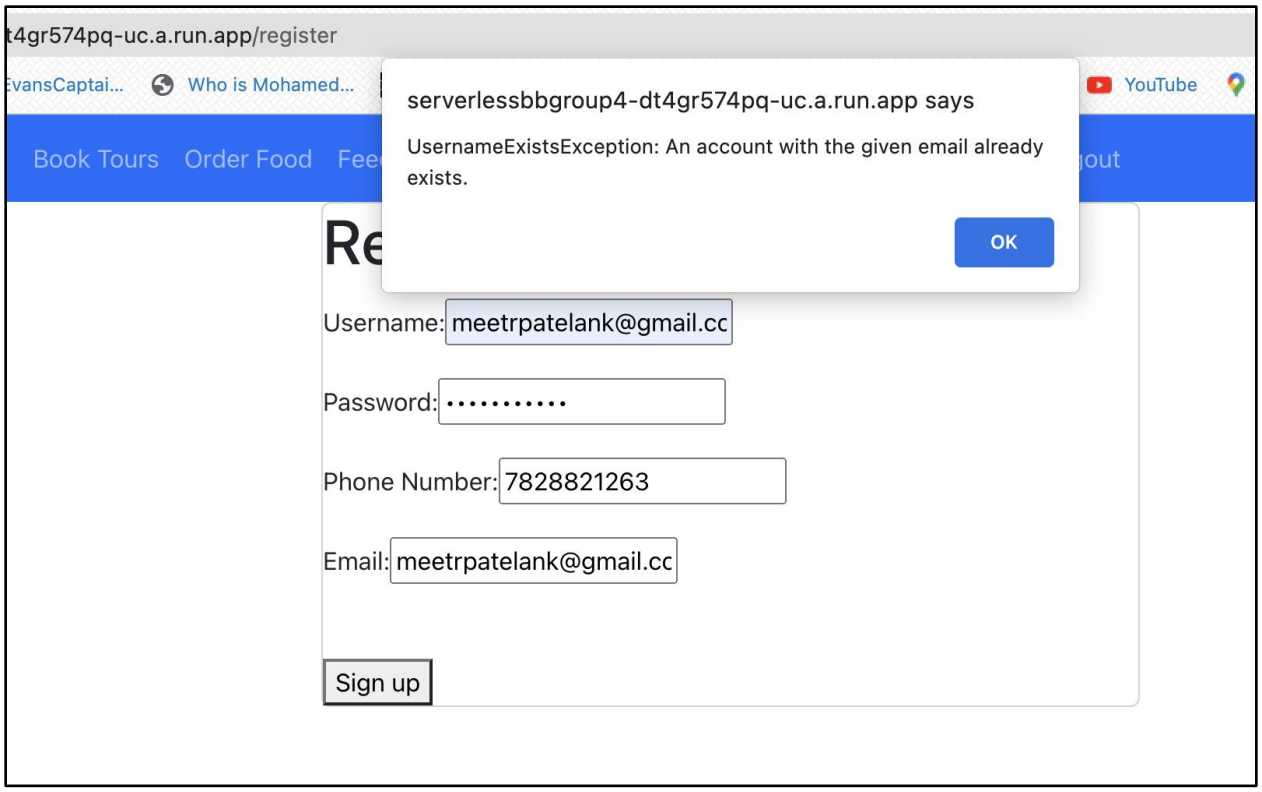

Figure 14 User Already Signed Up Test case

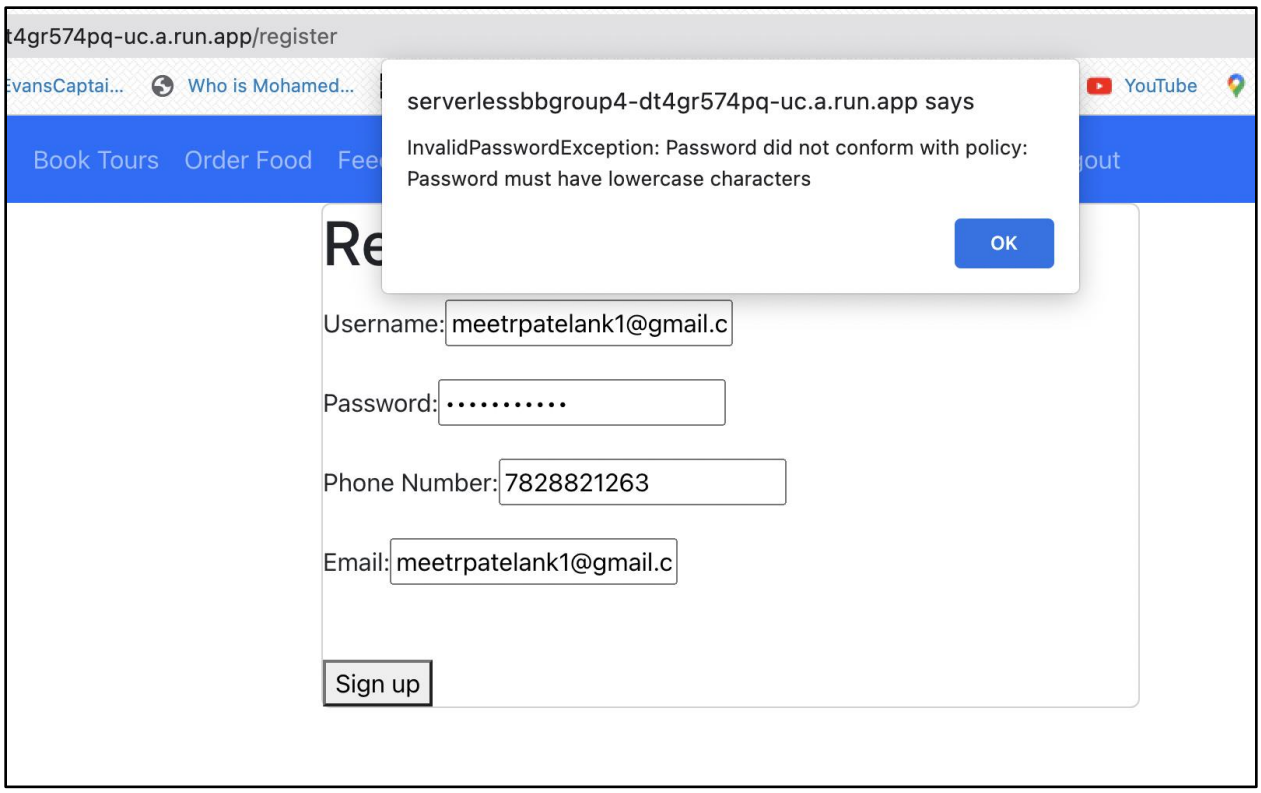

Figure 15 Password policy not matched

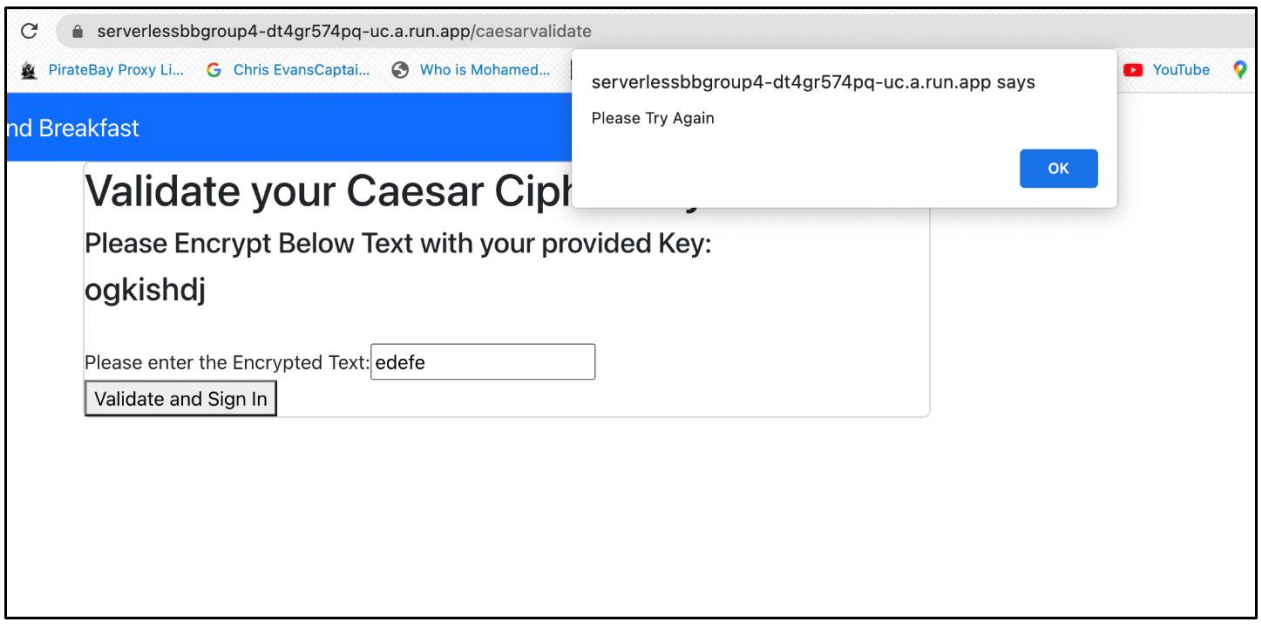

Figure 16 Caesar cipher failed.

9.3 Online Support Module:

Screenshot 17-21 show the unauthorized functionalities of bot.

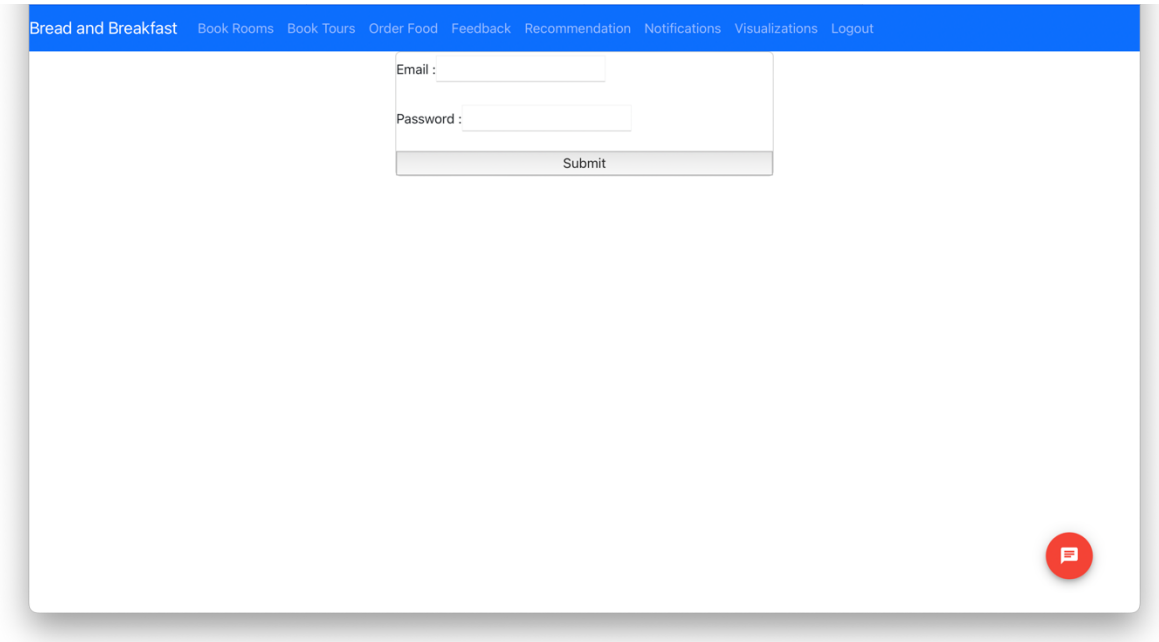

Figure 17 Screenshot of chatbot successfully deploying on the website.

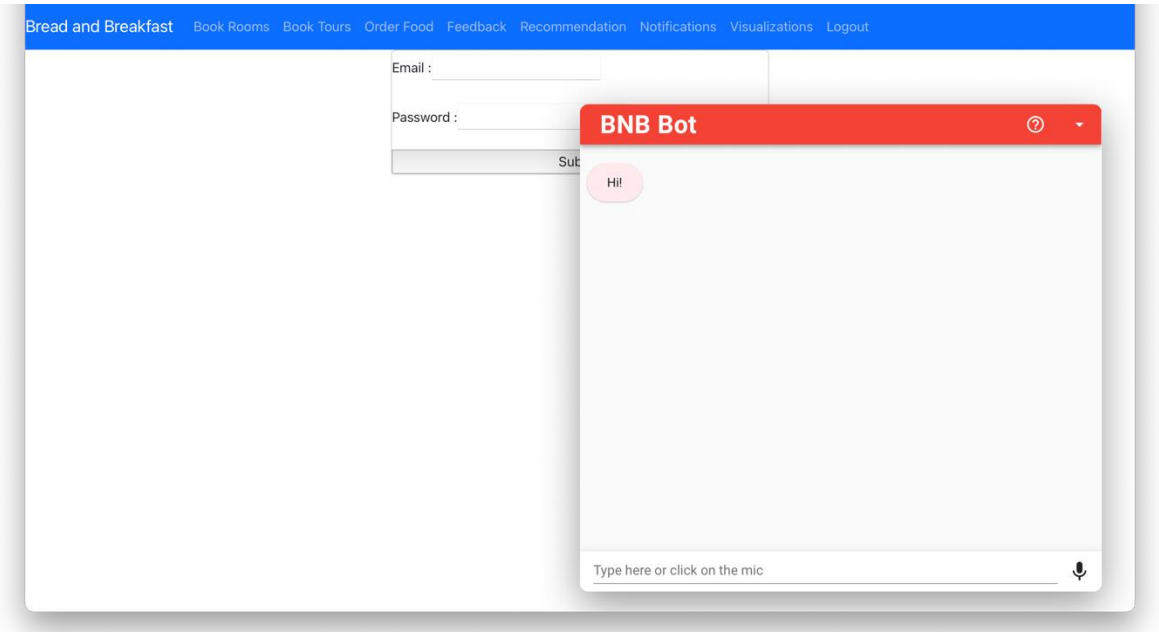

Figure 18 Screenshot of chatbot successfully deploying on the website.

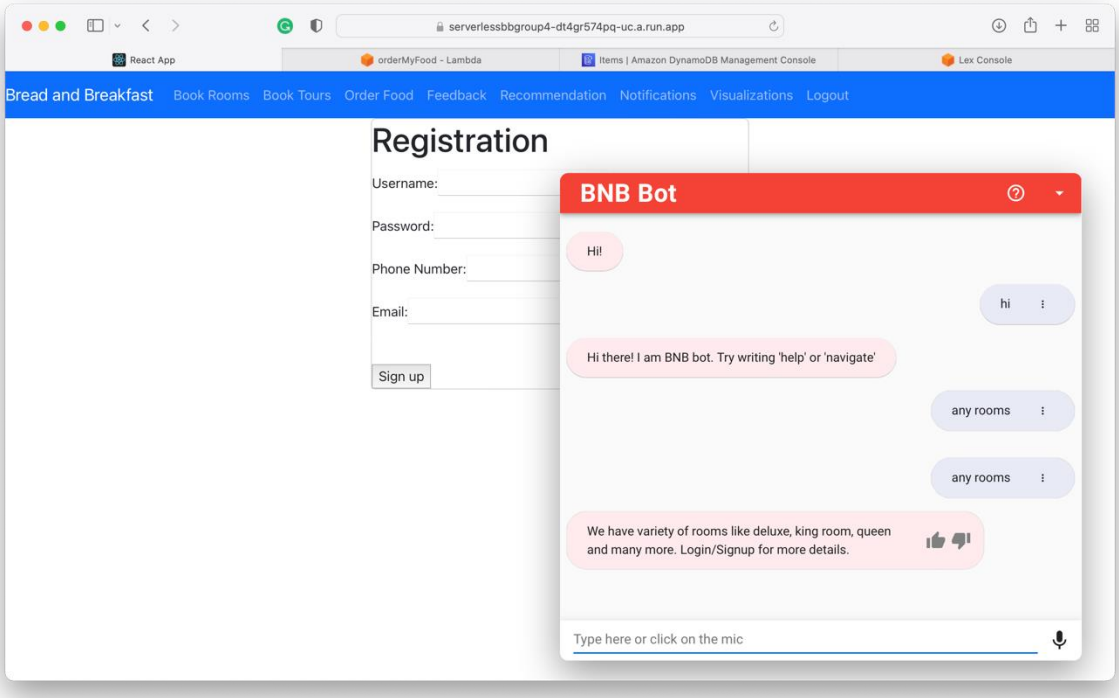

Figure 19 . Screenshot of chatbot performing unauthorized function: room details.

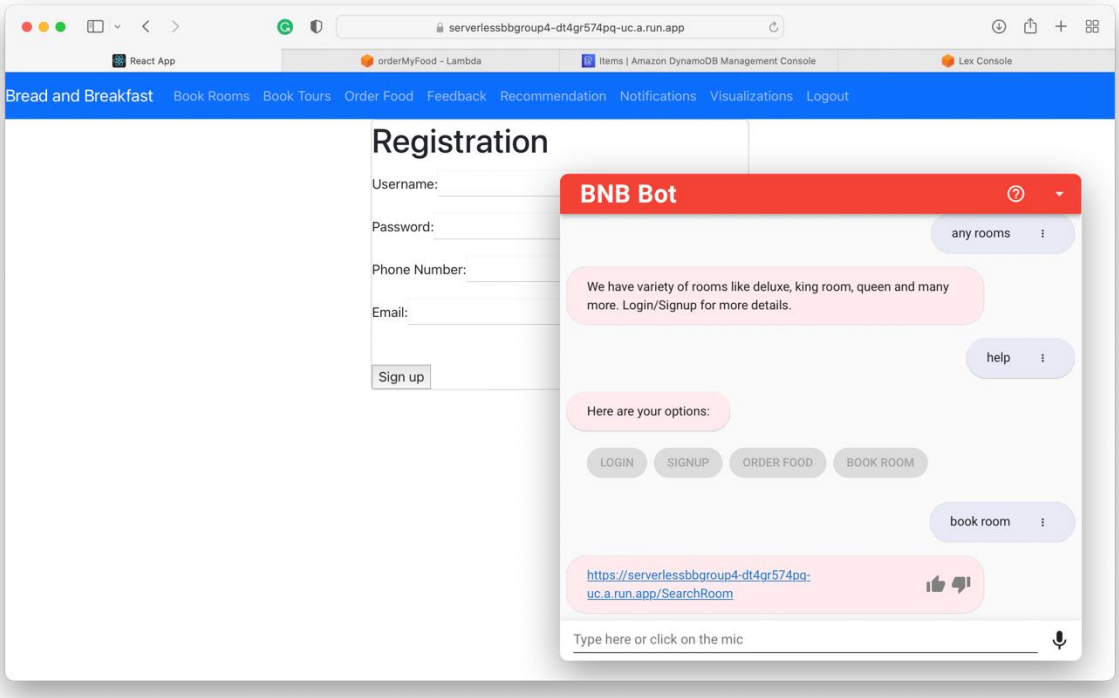

Figure 20 . Screenshot of chatbot performing unauthorized function: navigation/help.

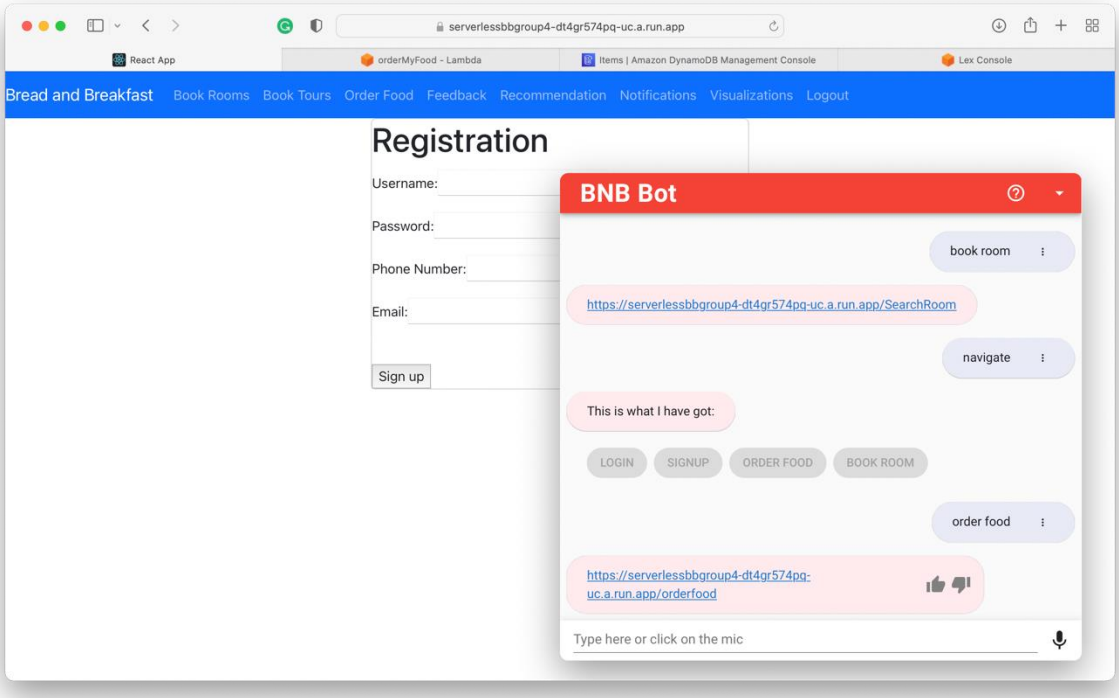

Figure 21 Screenshot of chatbot performing unauthorized function: navigation/help

Screenshot 22-26 show the authorized functionalities of bot.

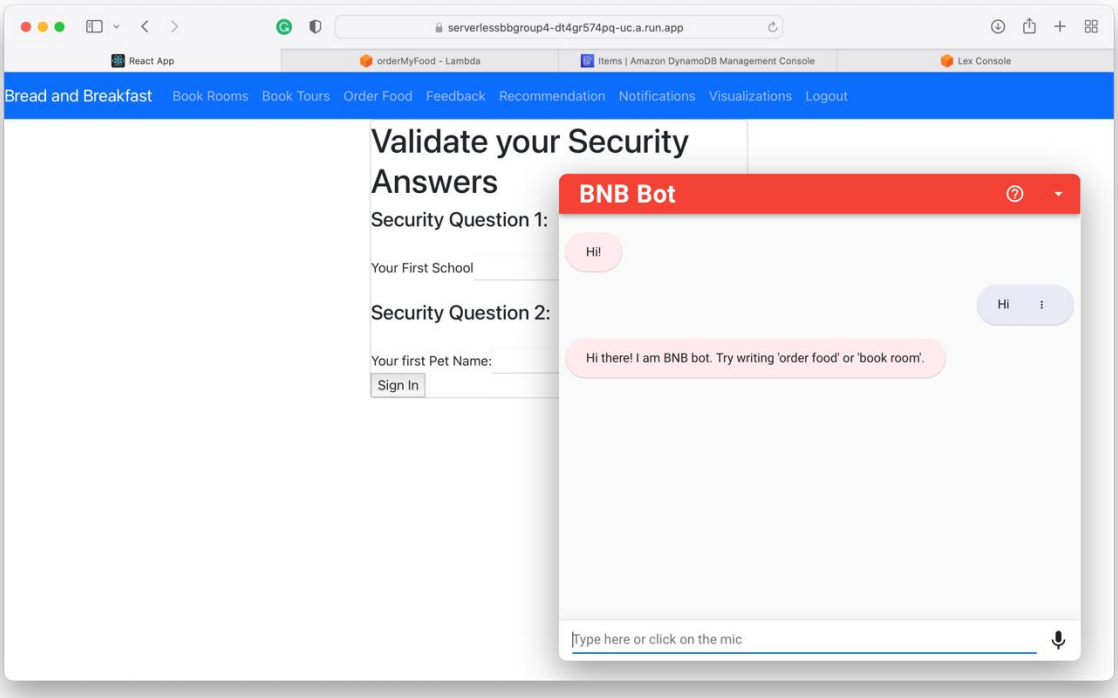

Figure 22 Screenshot of activation of authorized chatbot after login.

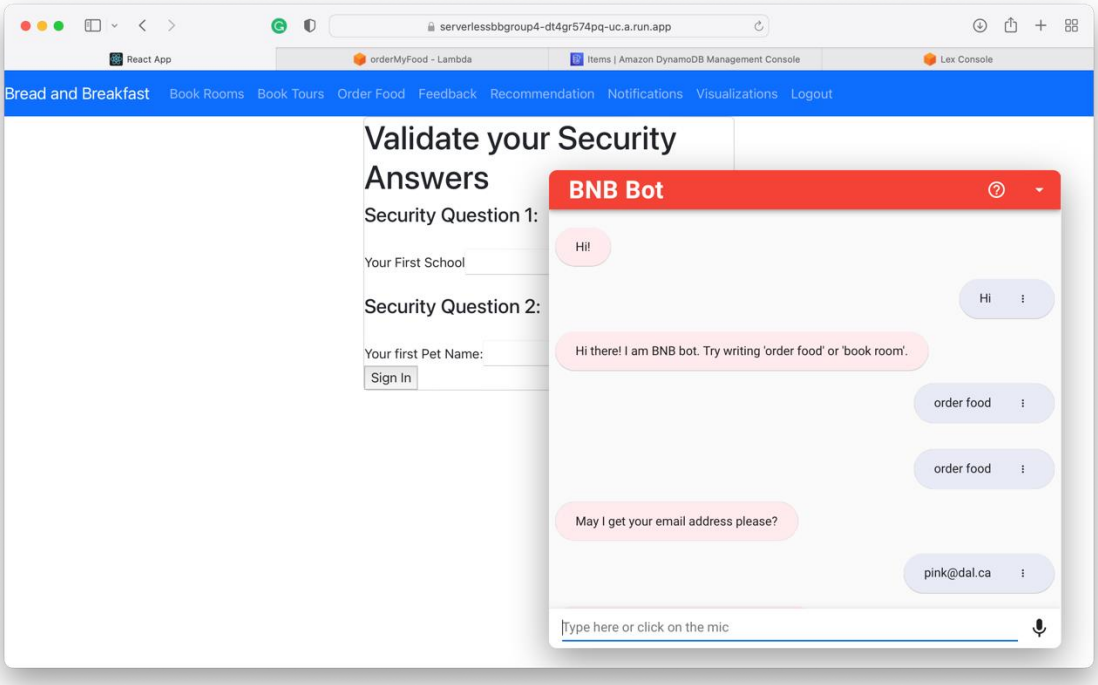

Figure 23 Screenshot of chatbot performing authorized function: order food.

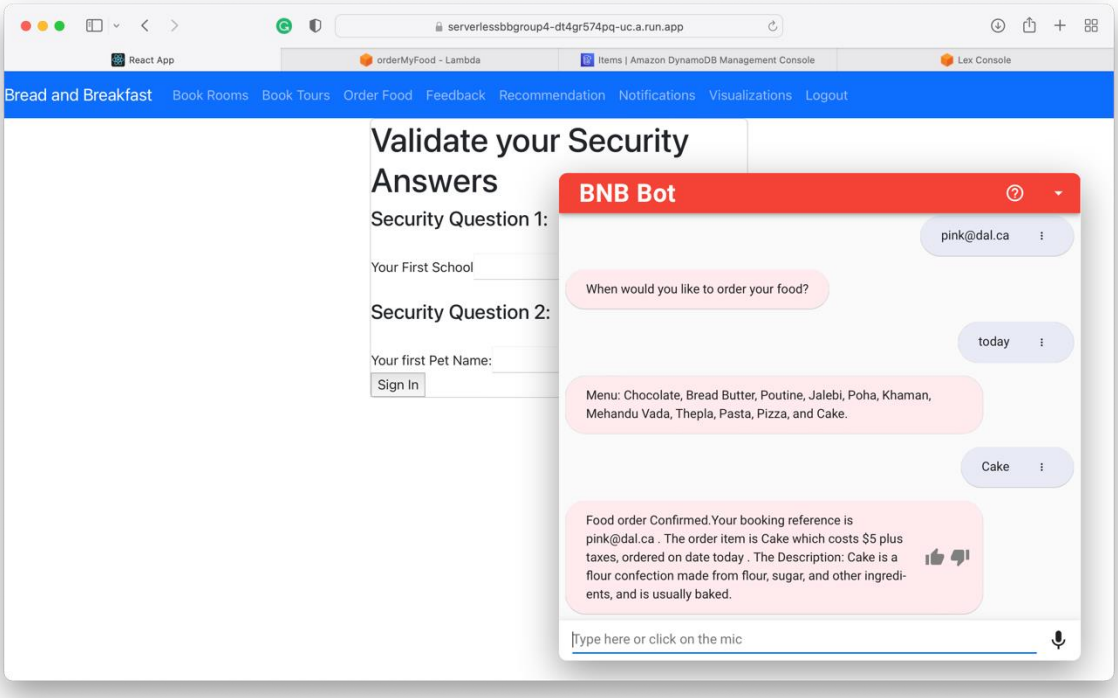

Figure 24 Screenshot of chatbot performing authorized function: order food with confirmation message.

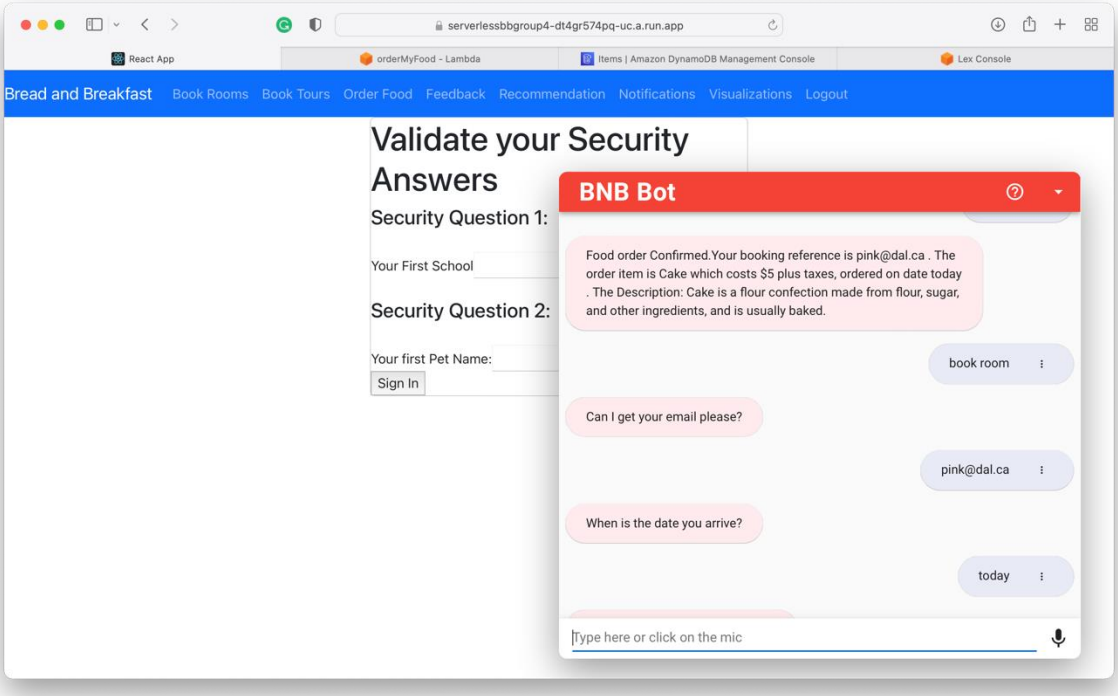

Figure 25 Screenshot of chatbot performing authorized function: book room.

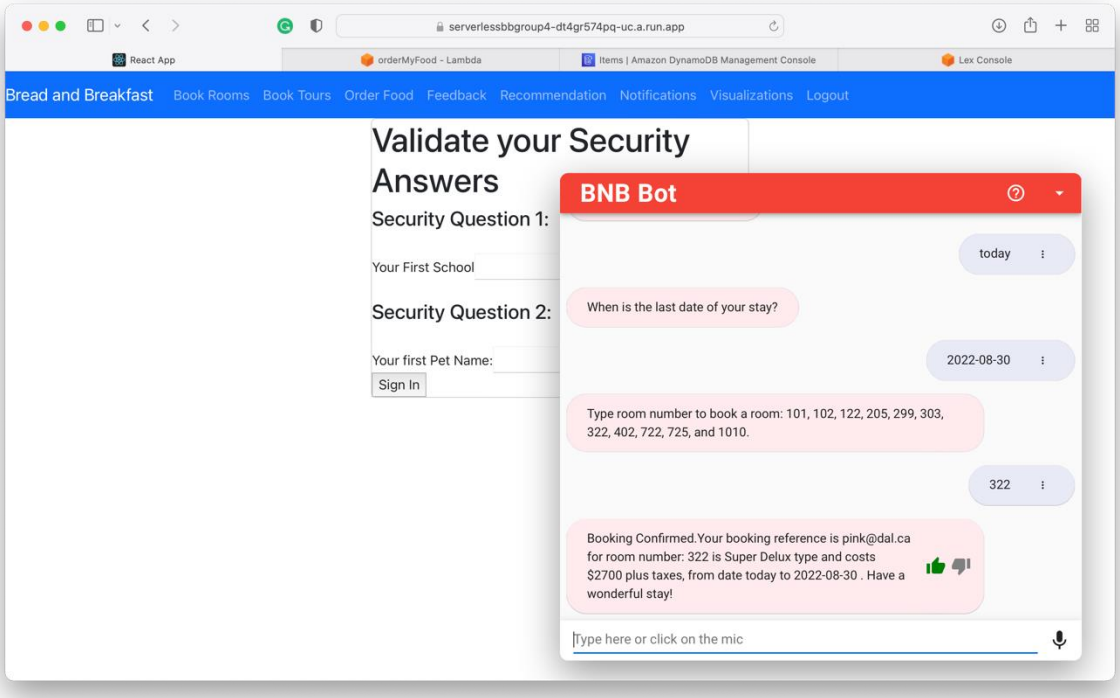

Figure 26 Screenshot of chatbot performing authorized function: book room with confirmation message.

Screenshot 27-22 show the unauthorized action attempted by user from chatbot and how website handled it.

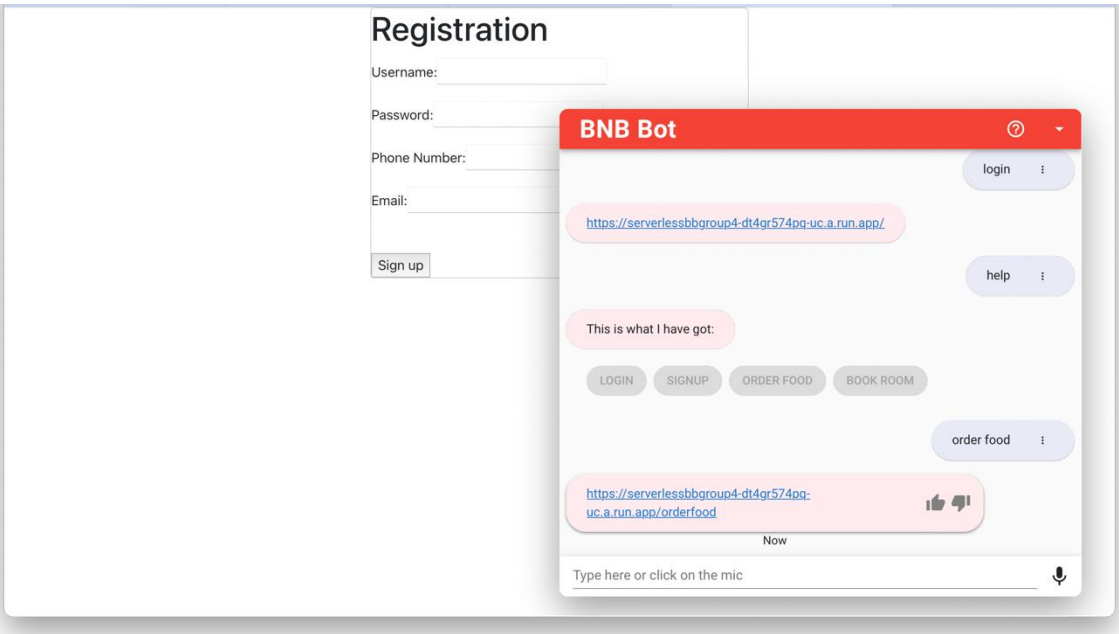

Figure 24. Screenshot of user trying to navigate to orderfood page before login.

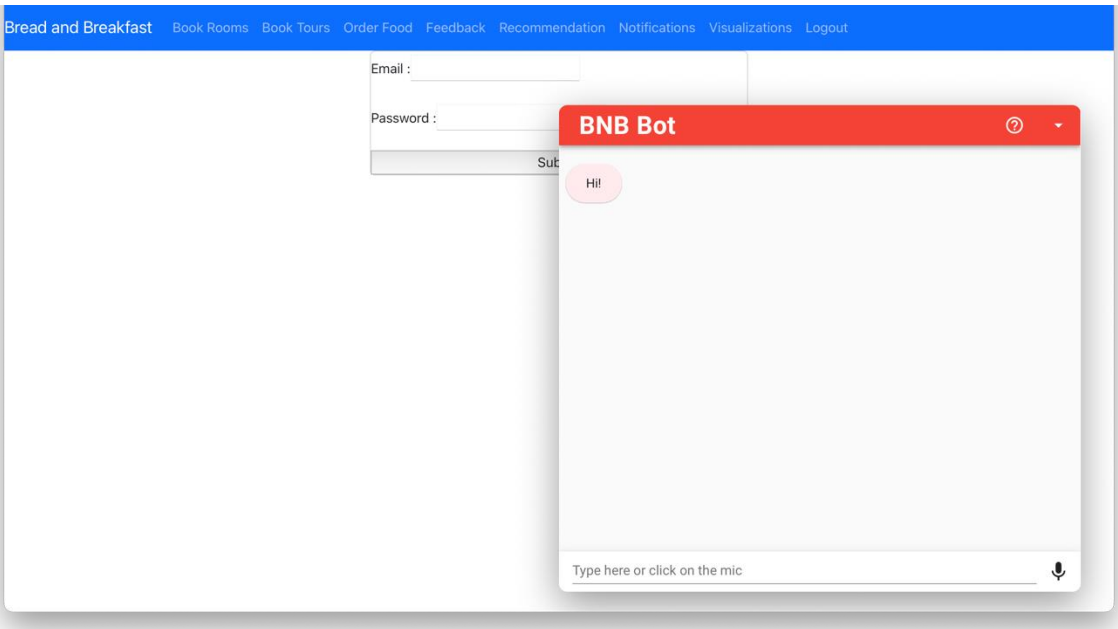

Figure 25. Screenshot of website landing on login page after user trying to access authorized functionality and chatbot successfully deployed.

Screenshot 23-24 show the entry of records taken from the chatbot into respective tables.

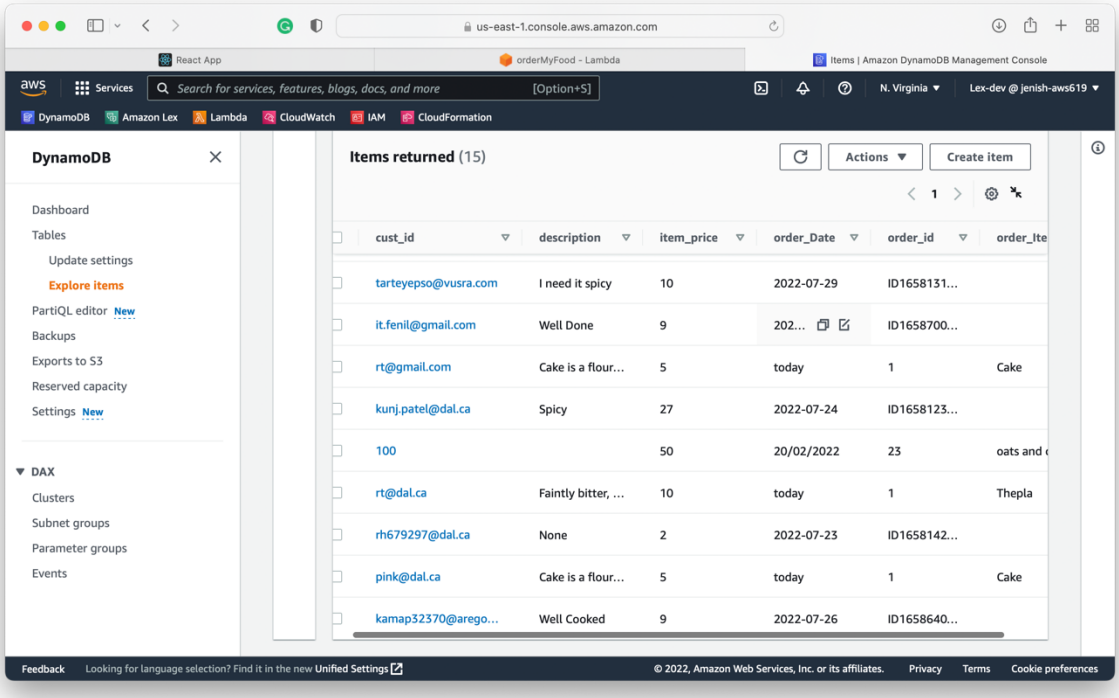

Figure 26. Screenshot of successful entry of record 'pink@dal.ca' into the foodorder table in DynamoDB.

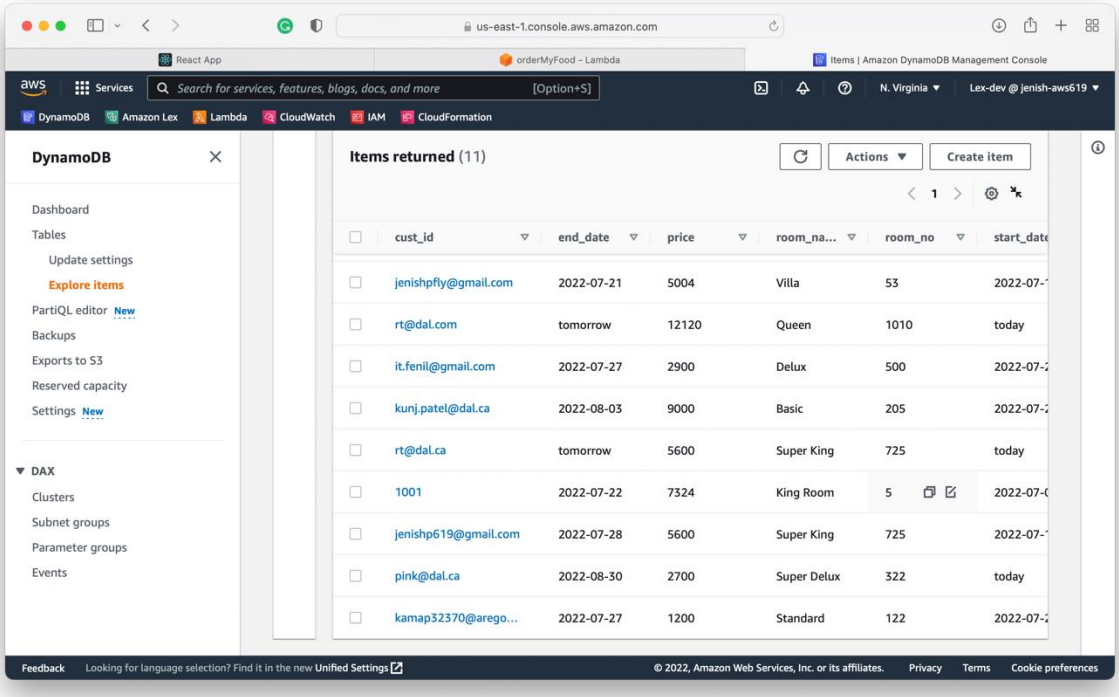

Figure 27. Screenshot of successful entry of record 'pink@dal.ca' into the room table in DynamoDB.

9.4 Message Passing

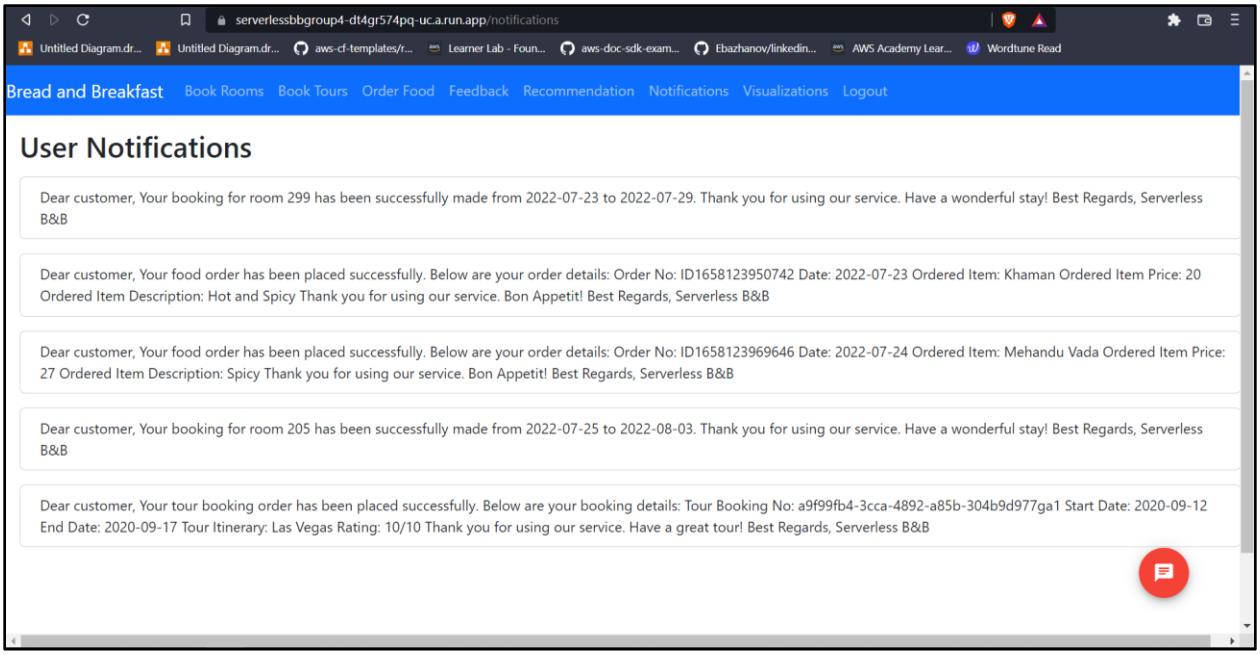

Figure 27 User Notifications

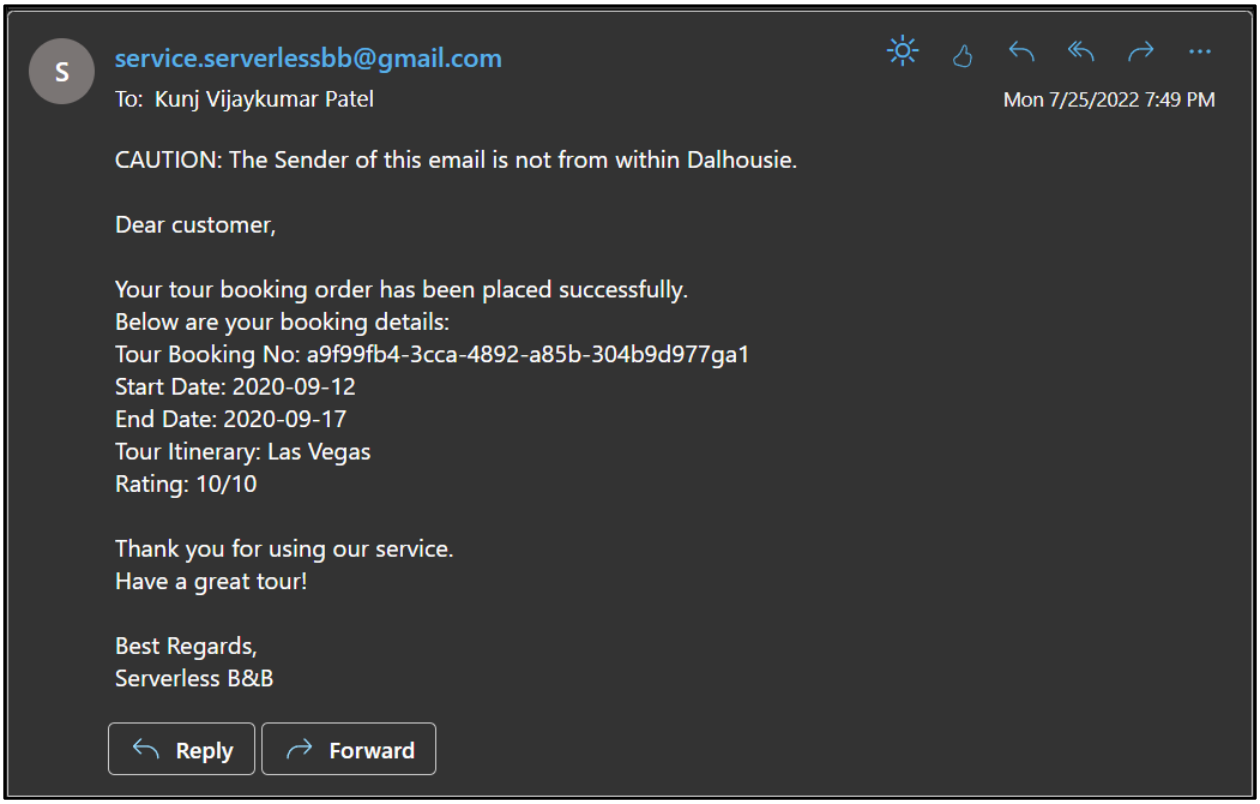

Figure 28 Booking Confirmation Email

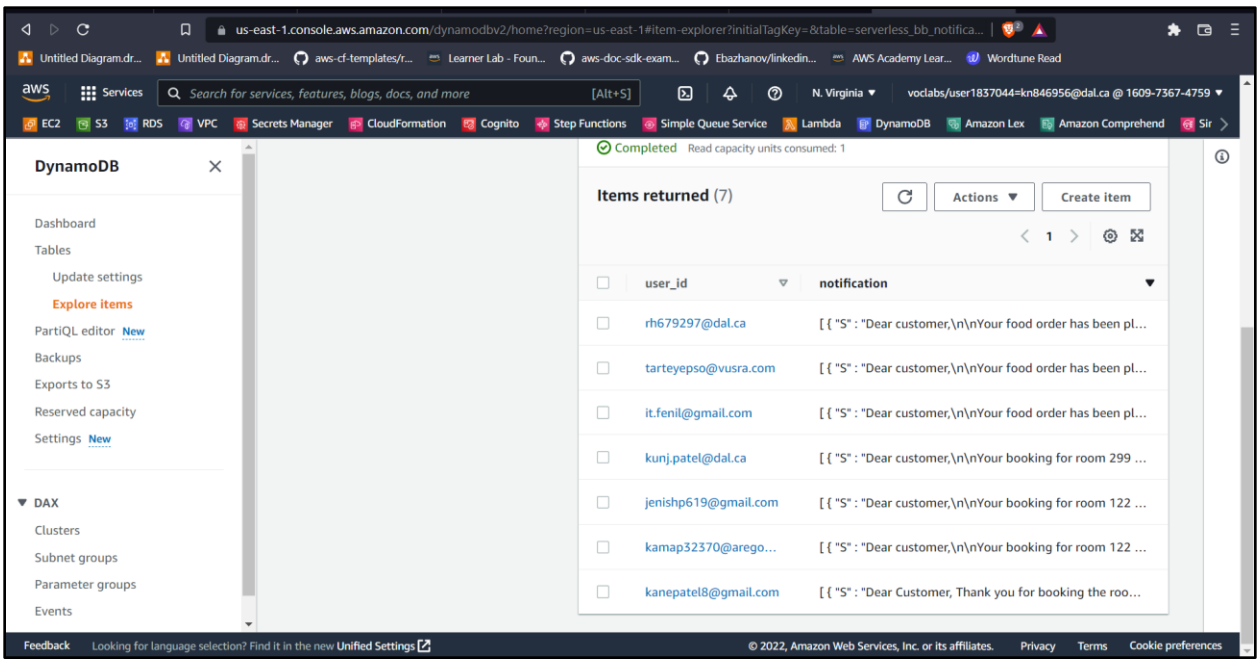

Figure 29 Notifications in Table

9.5 Machine Learning

- **Tour Recommendation**

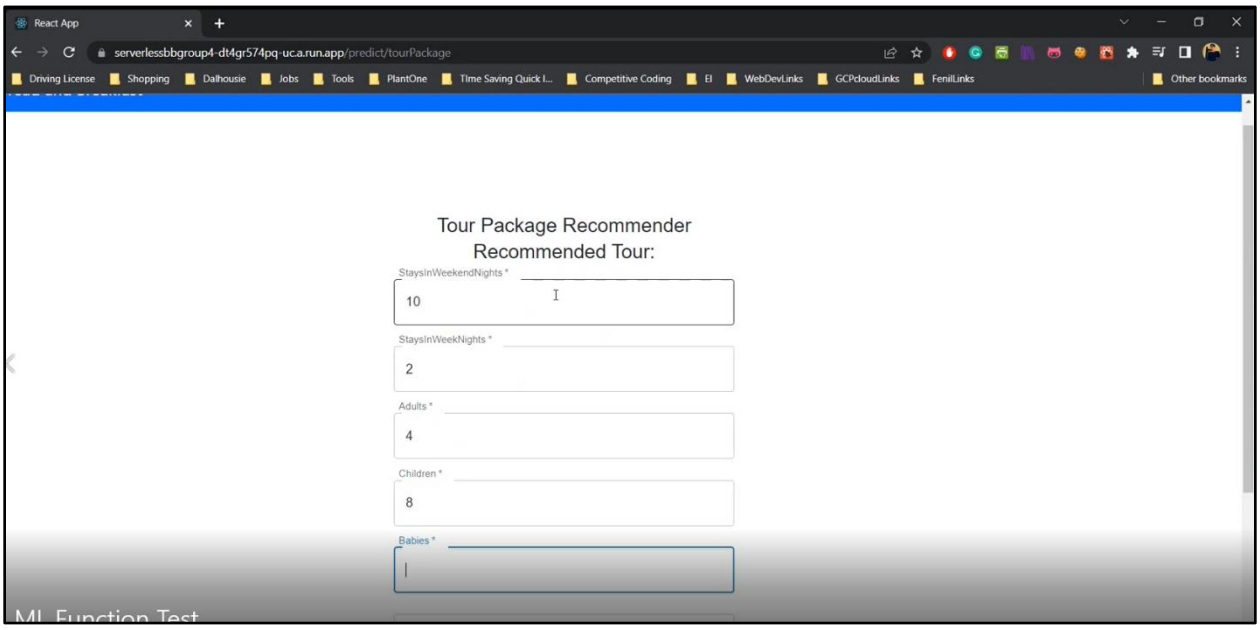

Figure 30 Tour Package Recommender Front End

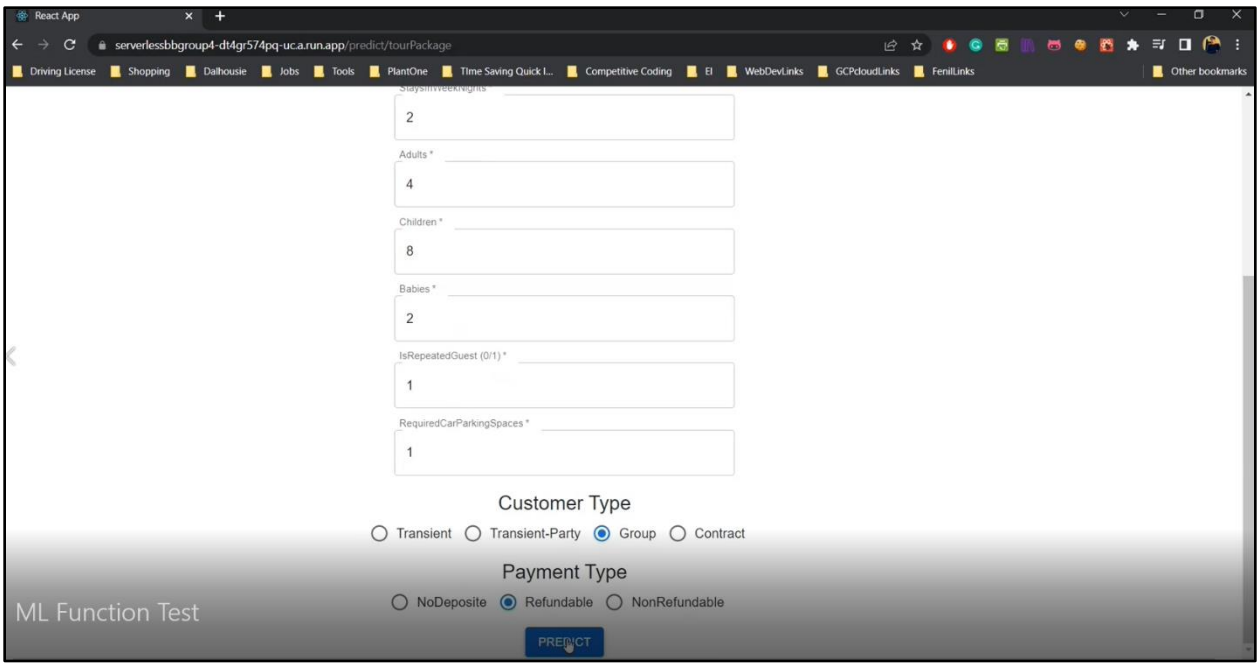

Figure 31 Tour Package Recommender Front End

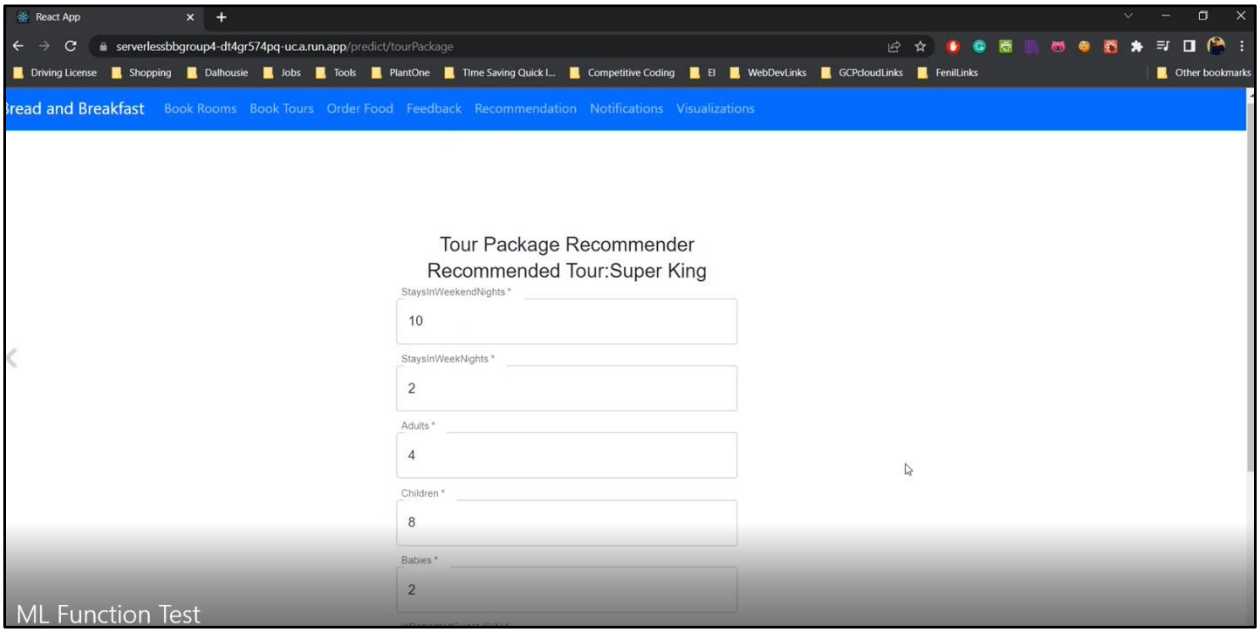

Figure 32 Tour Package Recommender Results

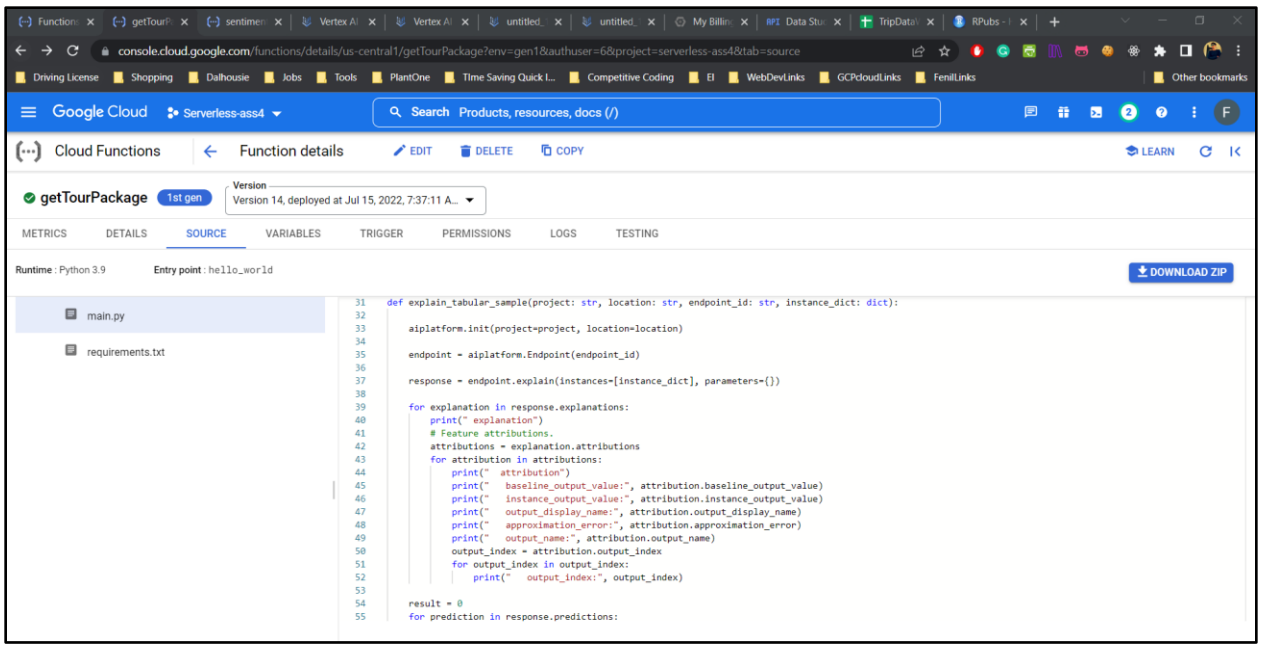

Figure 33 Tour Package Recommender Cloud Function

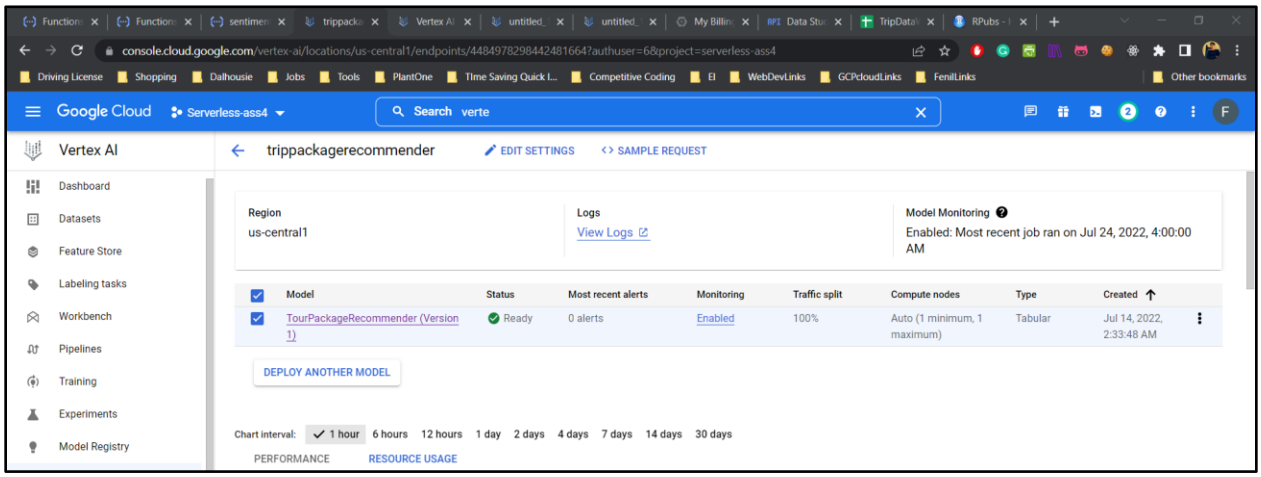

Figure 34 Tour Package Recommended Vertex AI

- **Sentiment Analysis**

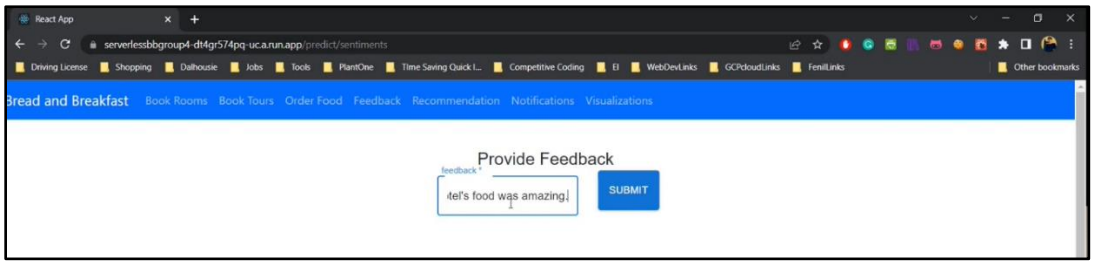

Figure 35 Sentiment Analysis Front End

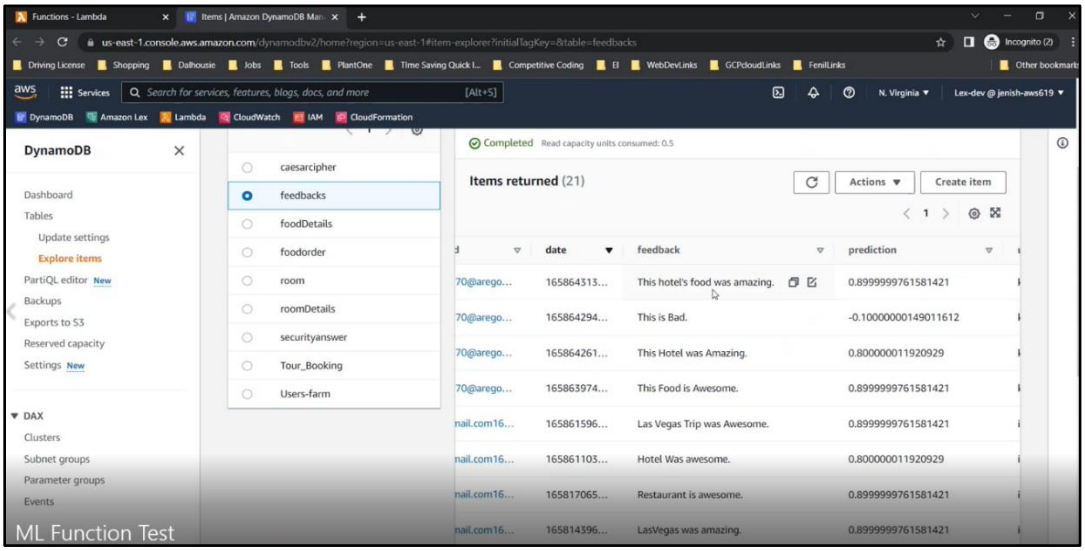

Figure 36 Sentiment Analysis Results in AWS DynamoDB

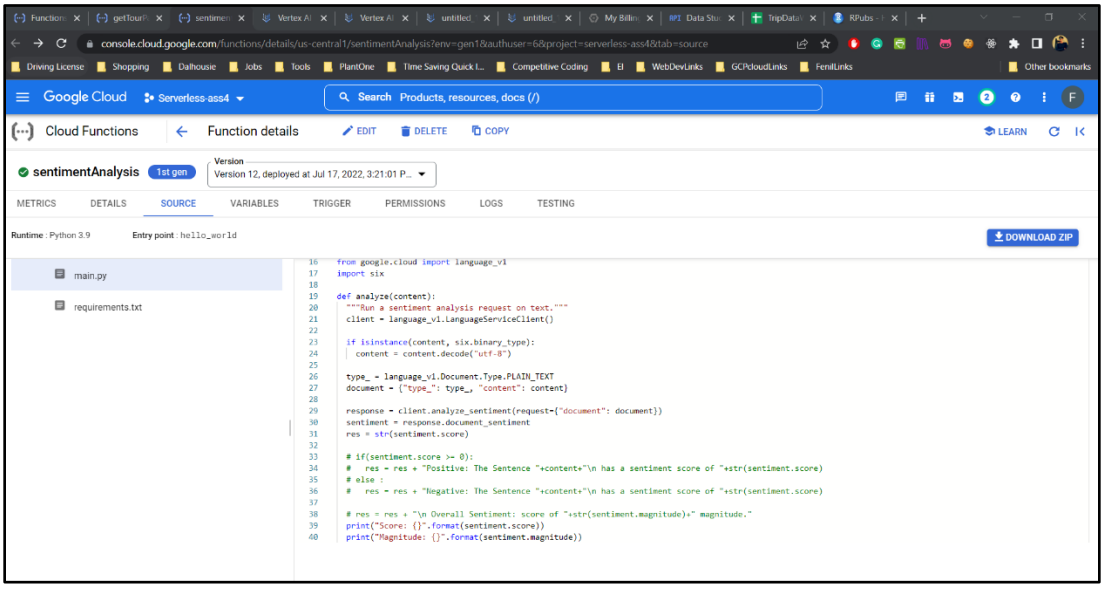

Figure 37 Sentiment Analysis Cloud Function

9.6 Web Application Building and Hosting

Room Booking:

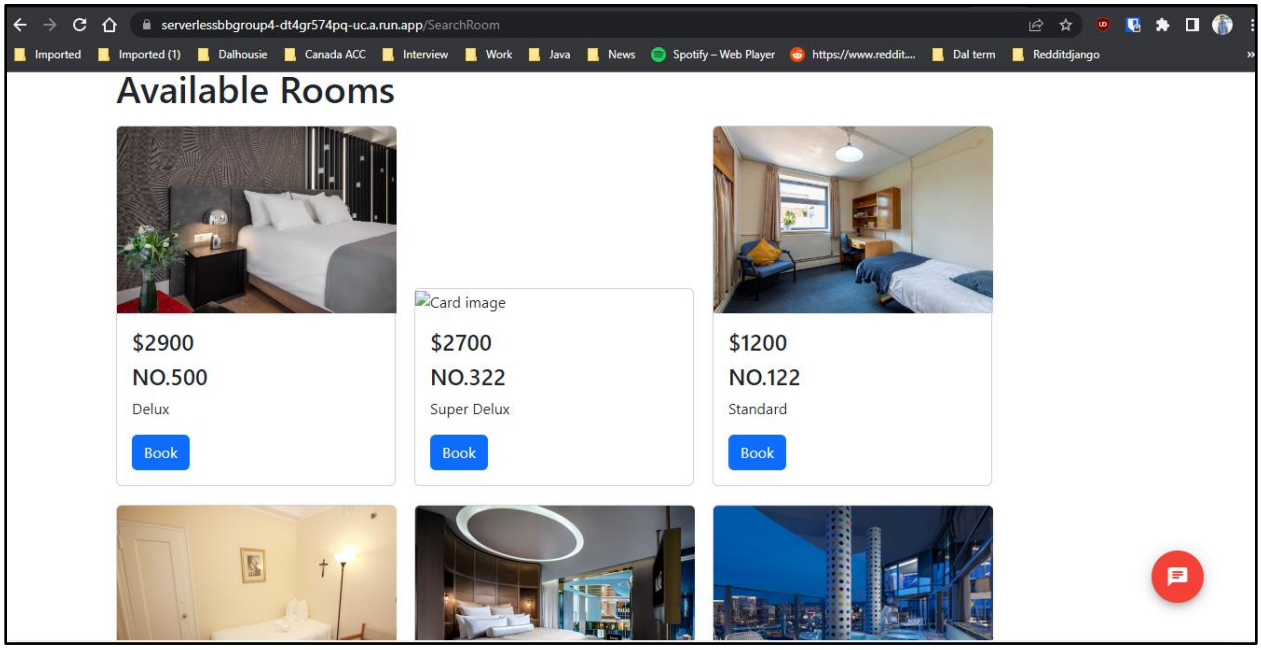

Figure 38 Room booking home page

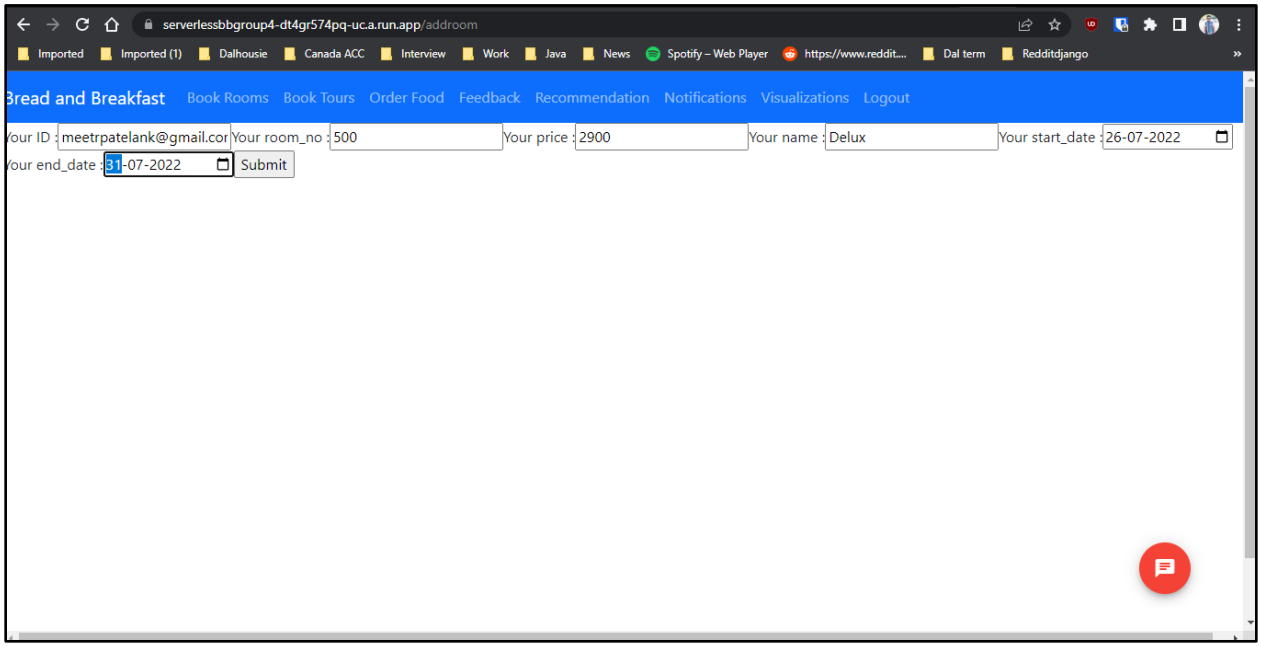

Figure 39 : User enters the room booking details

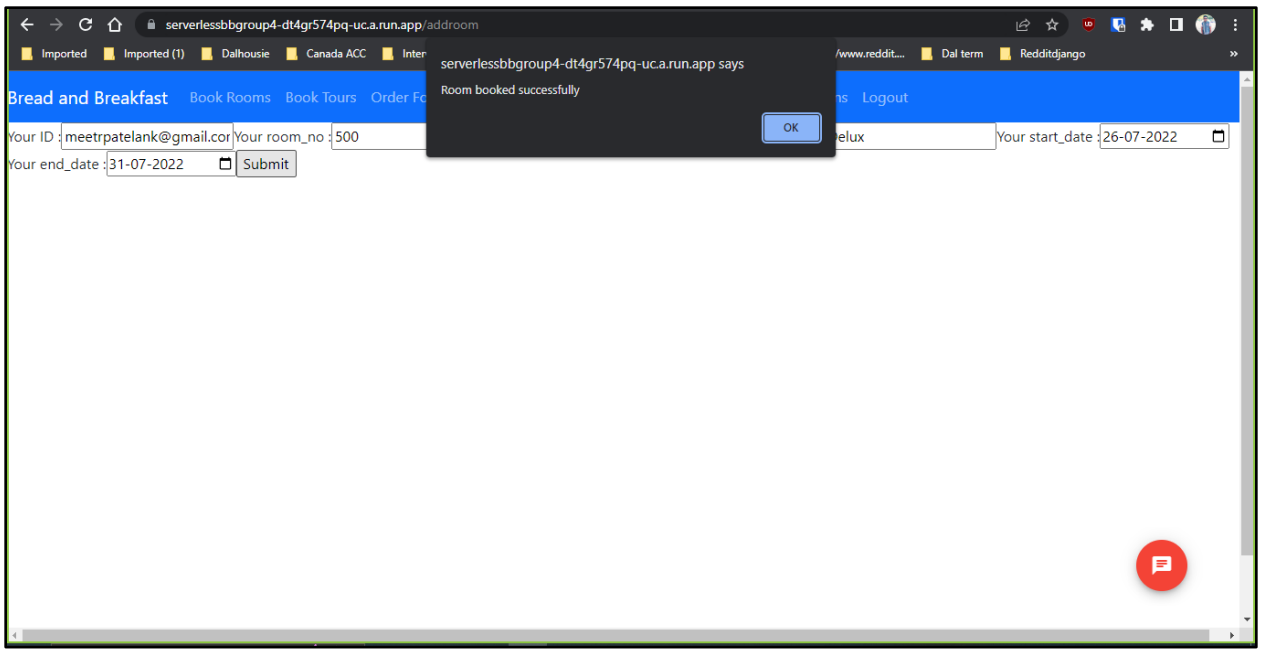

Figure 40 Room booked successfully message

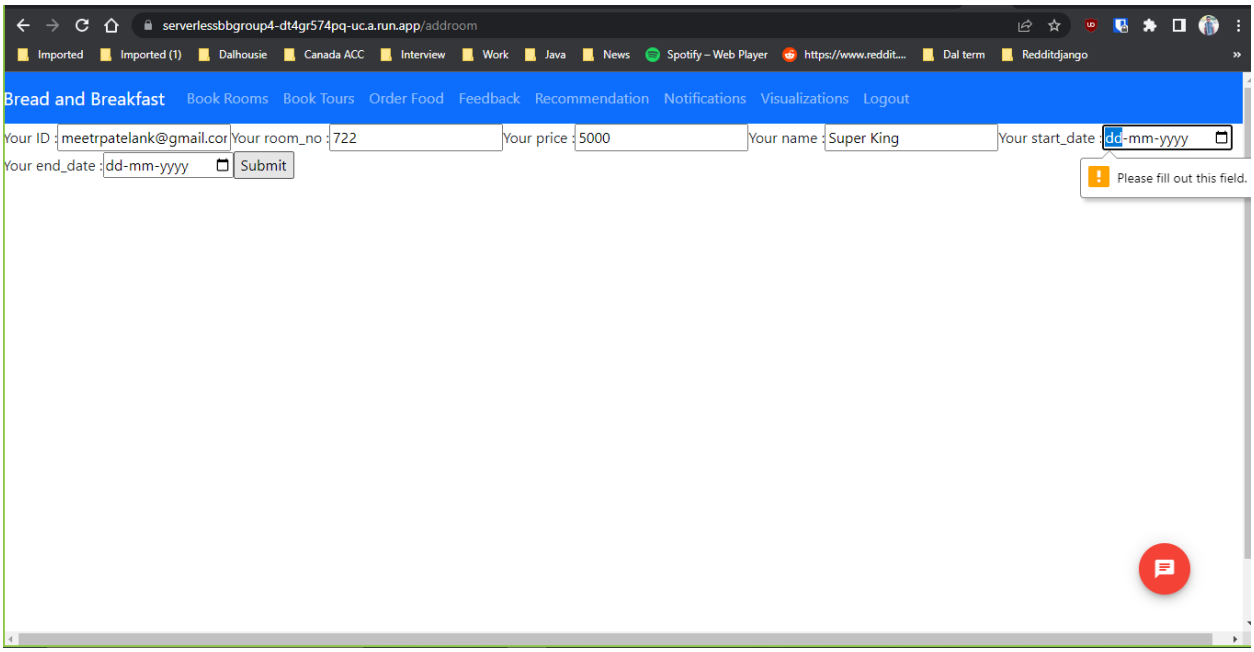

Figure 41 Validations for the fields in the room booking form

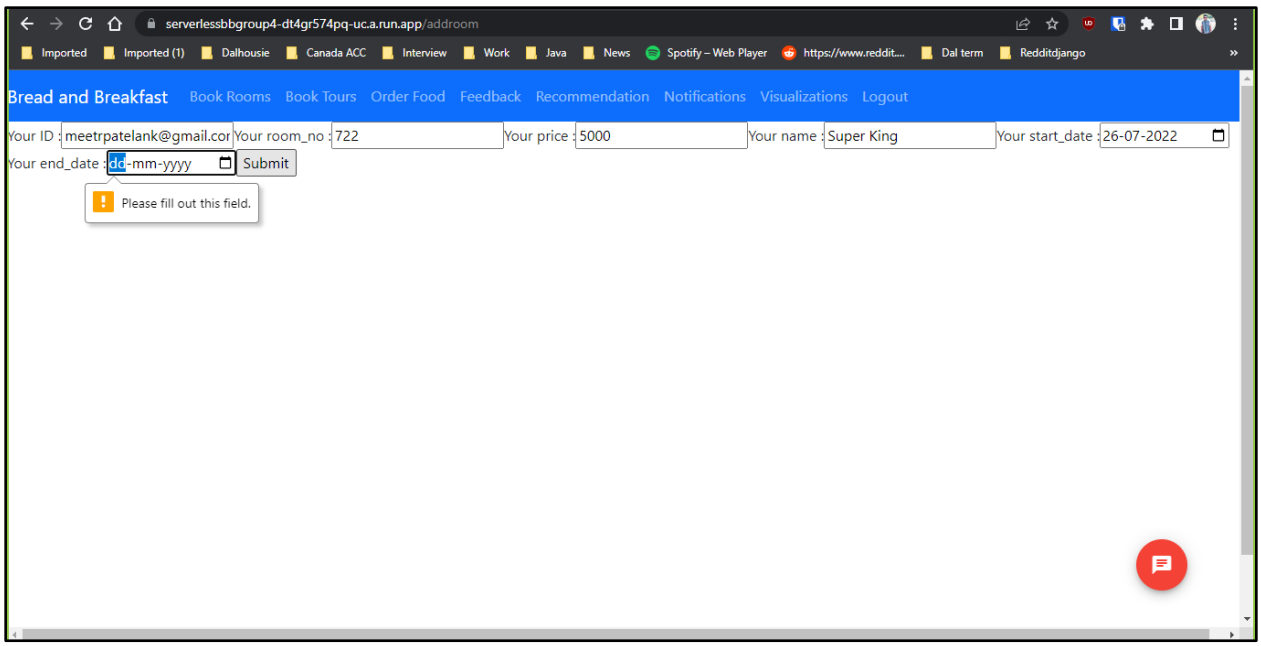

Figure 42 Validations for the fields in the room booking form

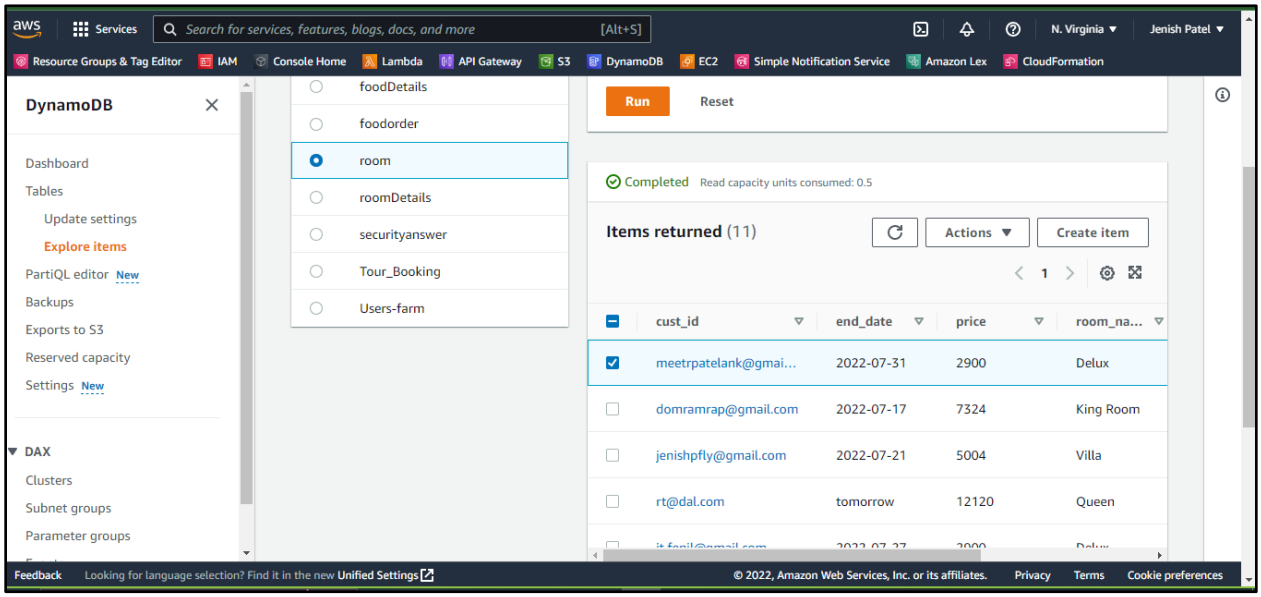

Figure 43 Room booking details in the Amazon DynamoDB table

Tour Booking:

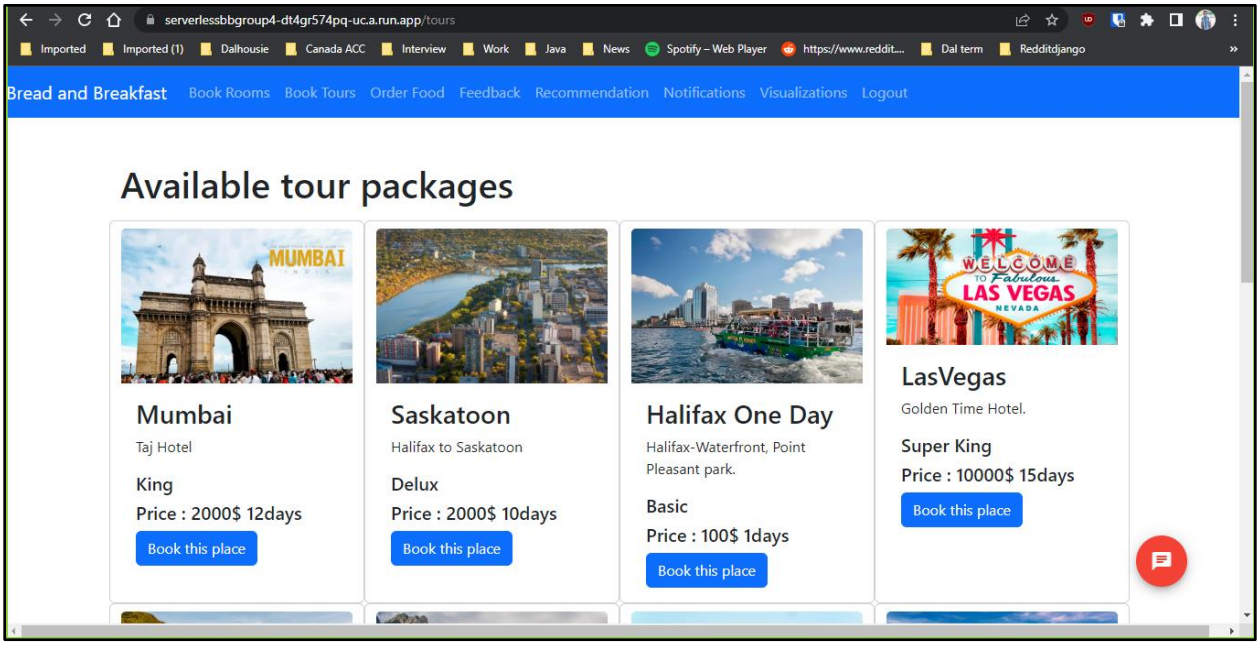

Figure 44 Tour package home page for user

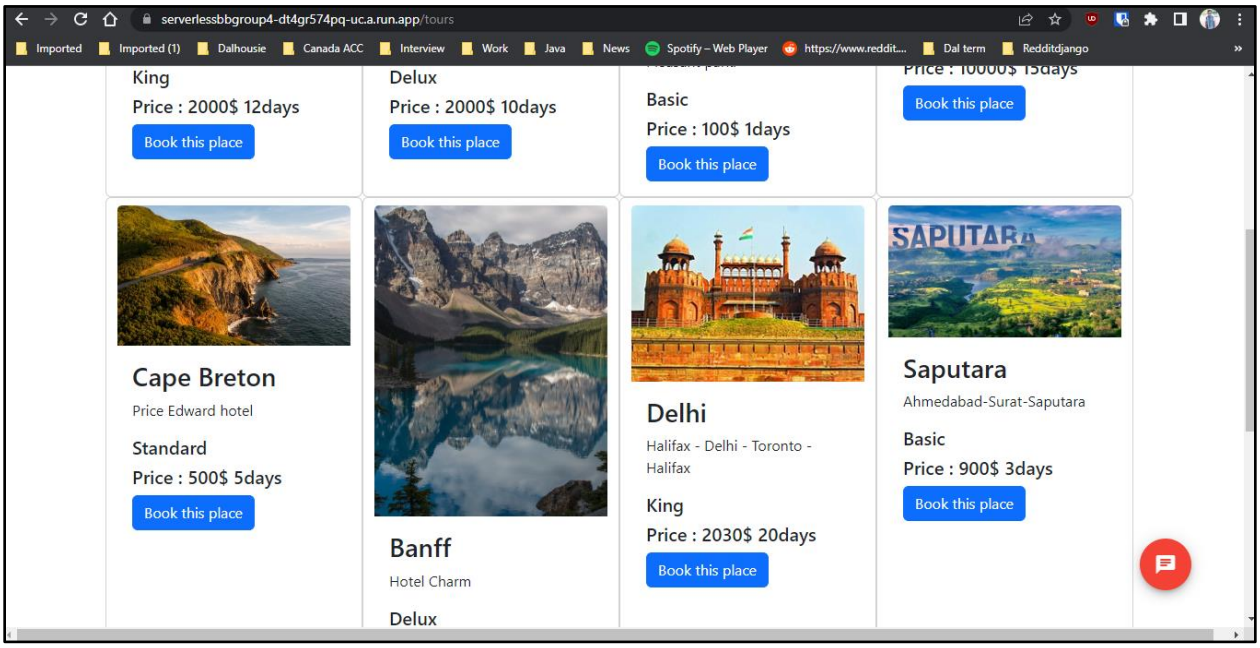

Figure 45 Tour package home page for user

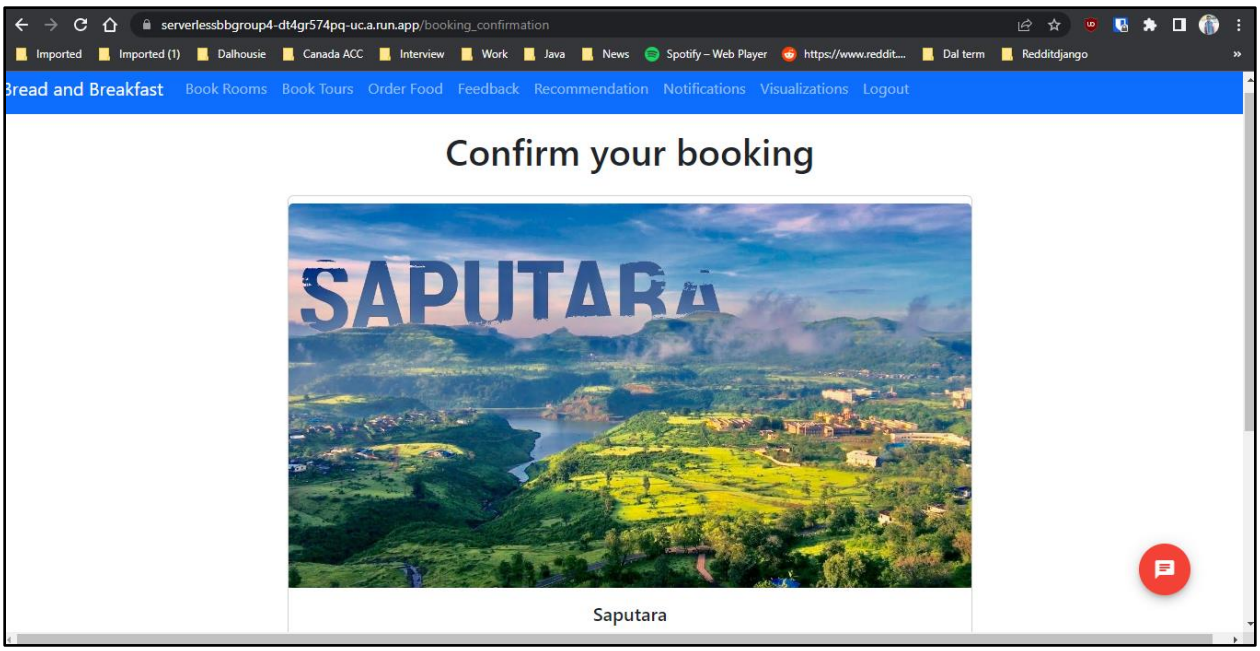

Figure 46 Booking confirmation check for the user

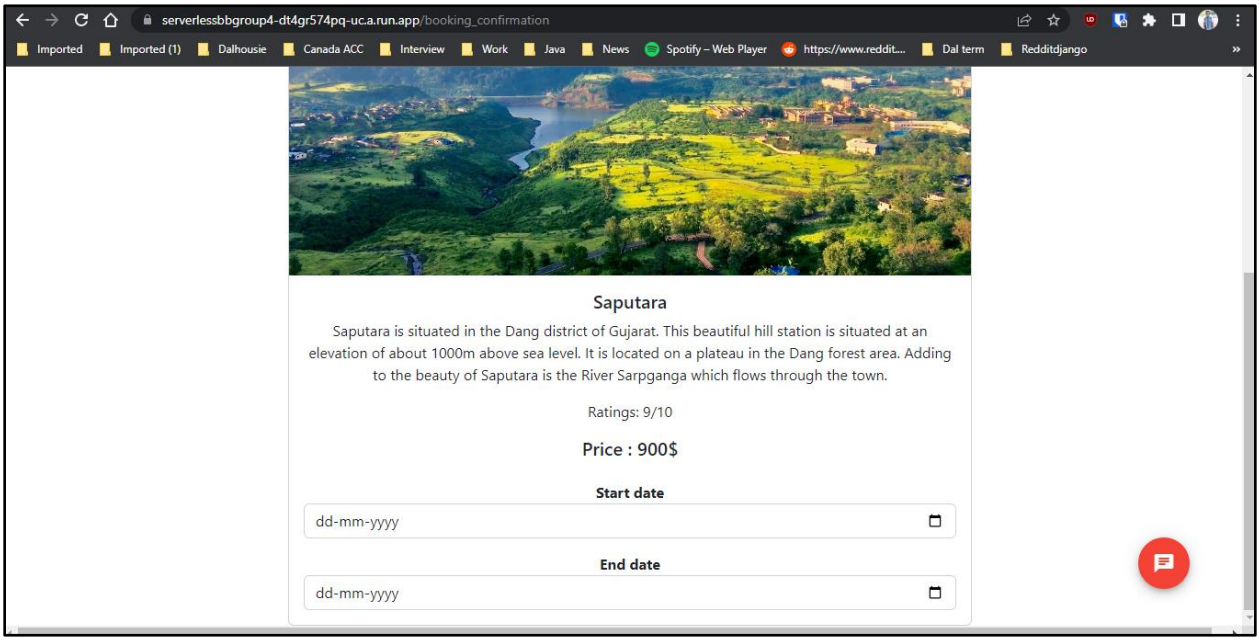

Figure 47 Booking date confirmation for the user

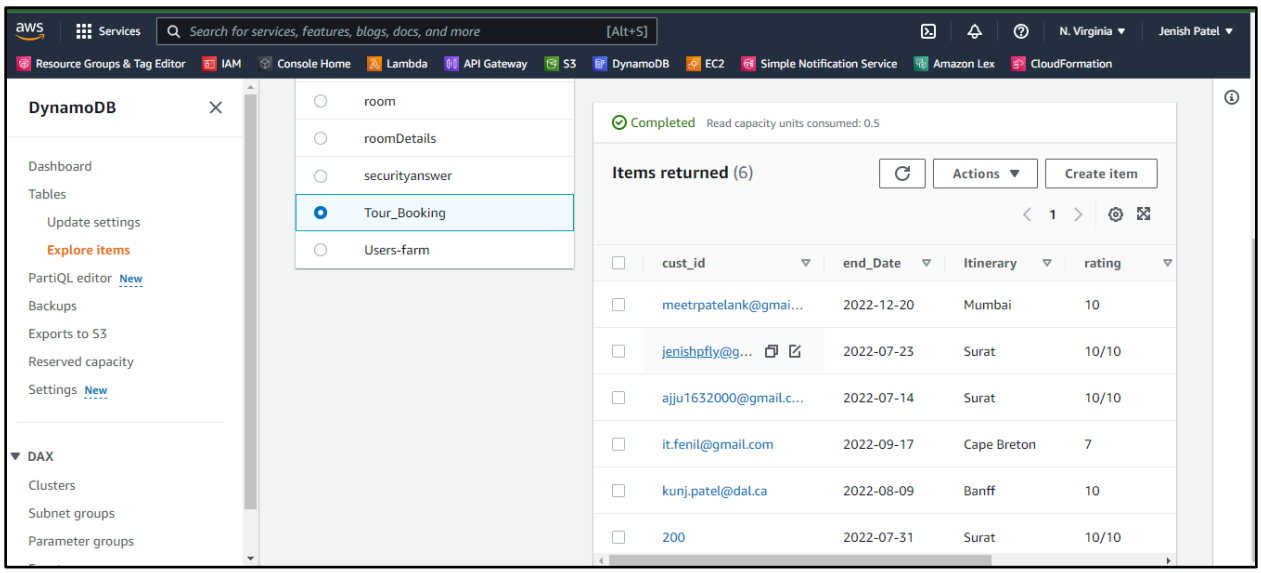

Figure 48 Data inserted in Amazon DynamoDB table for Tour booking

Food Order :

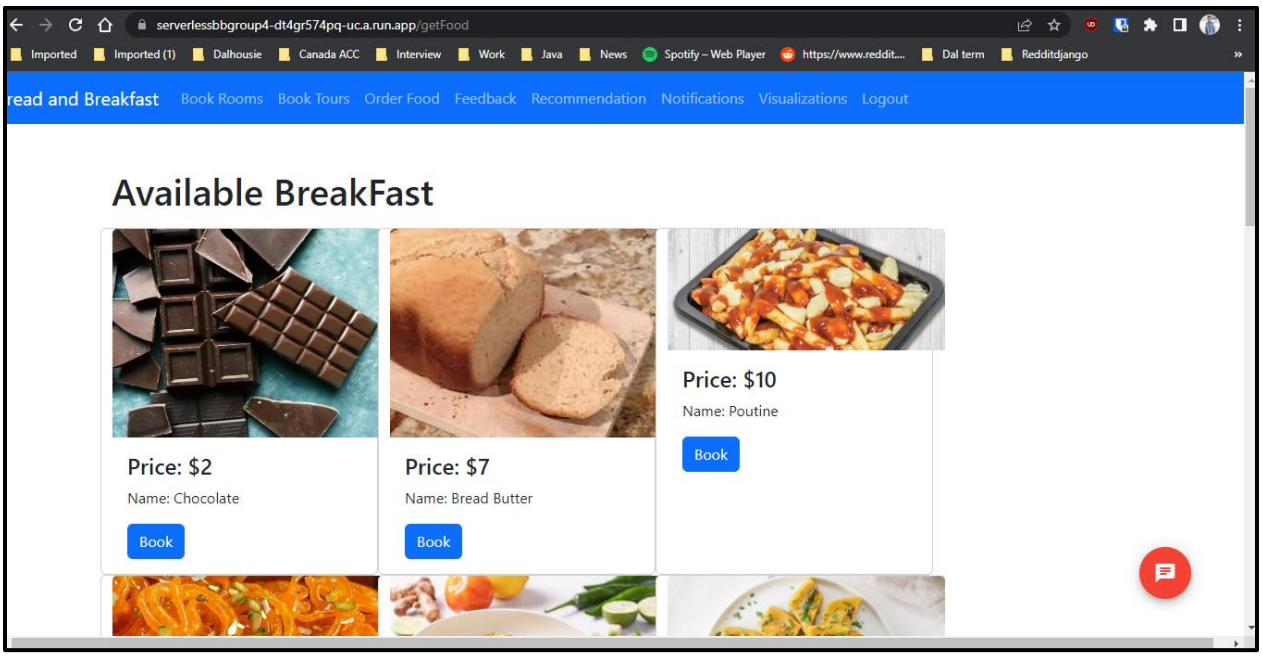

Figure 49 Food Order home page

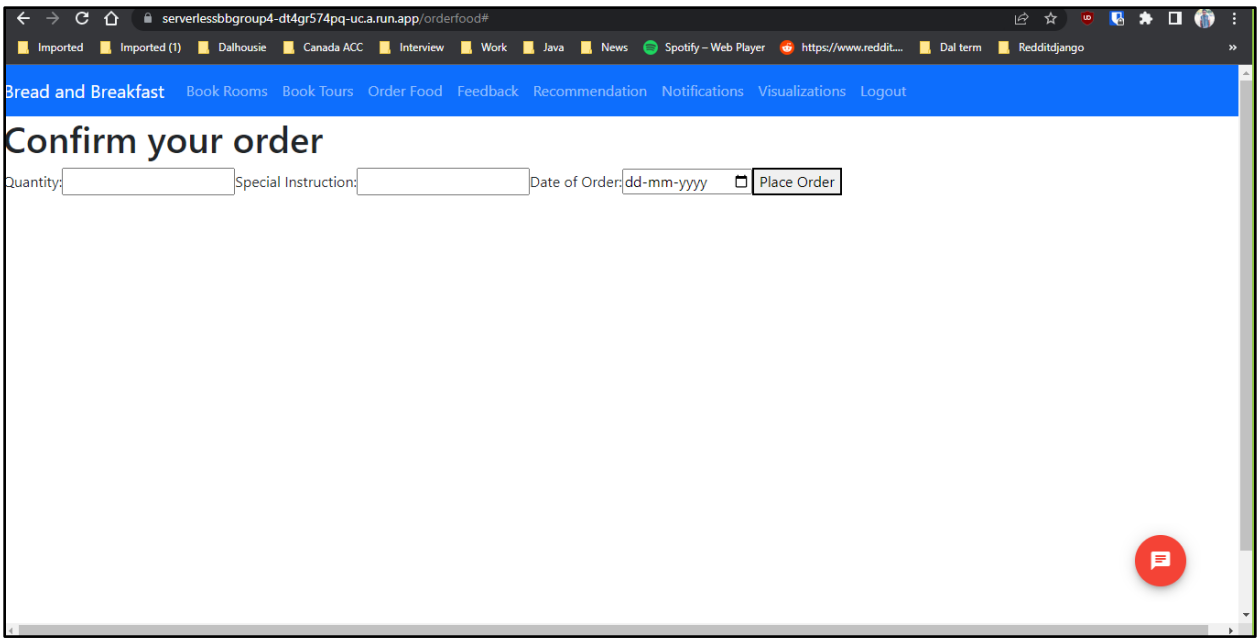

Figure 50 Order confirmation details form for the user

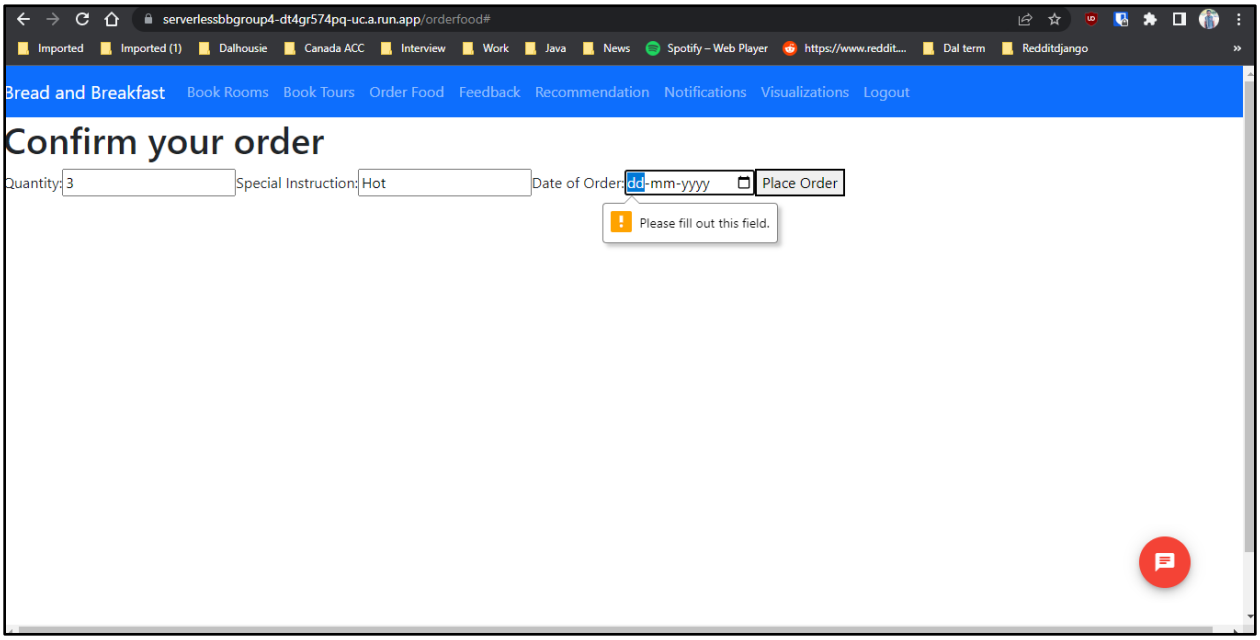

Figure 51 Validations in the order food form

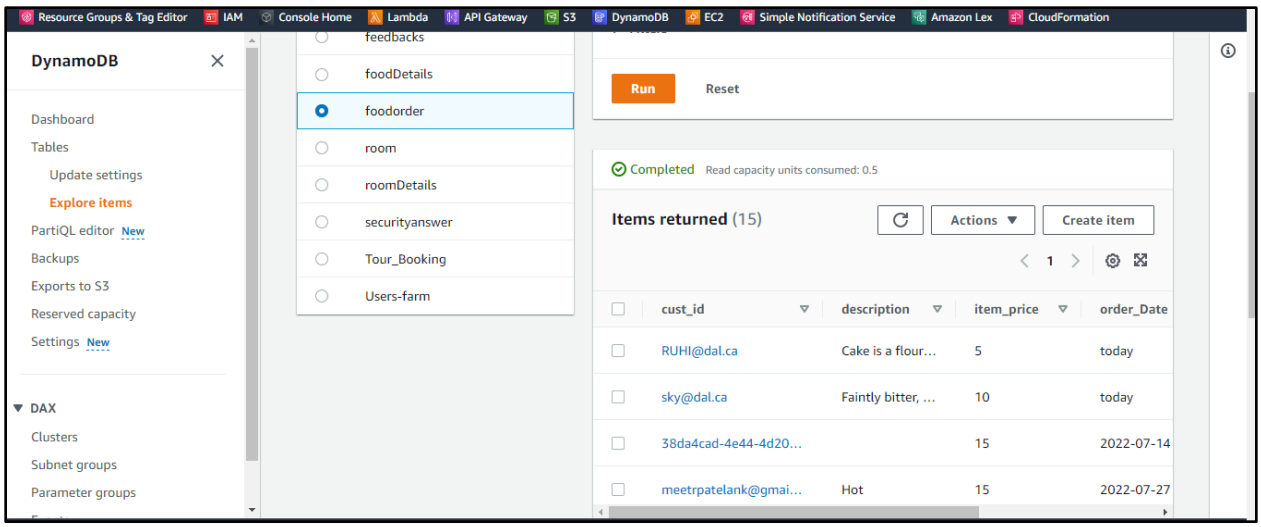

Figure 52 Food order details stored in Amazon DynamoDB table

Lambda Functions used in the Project:

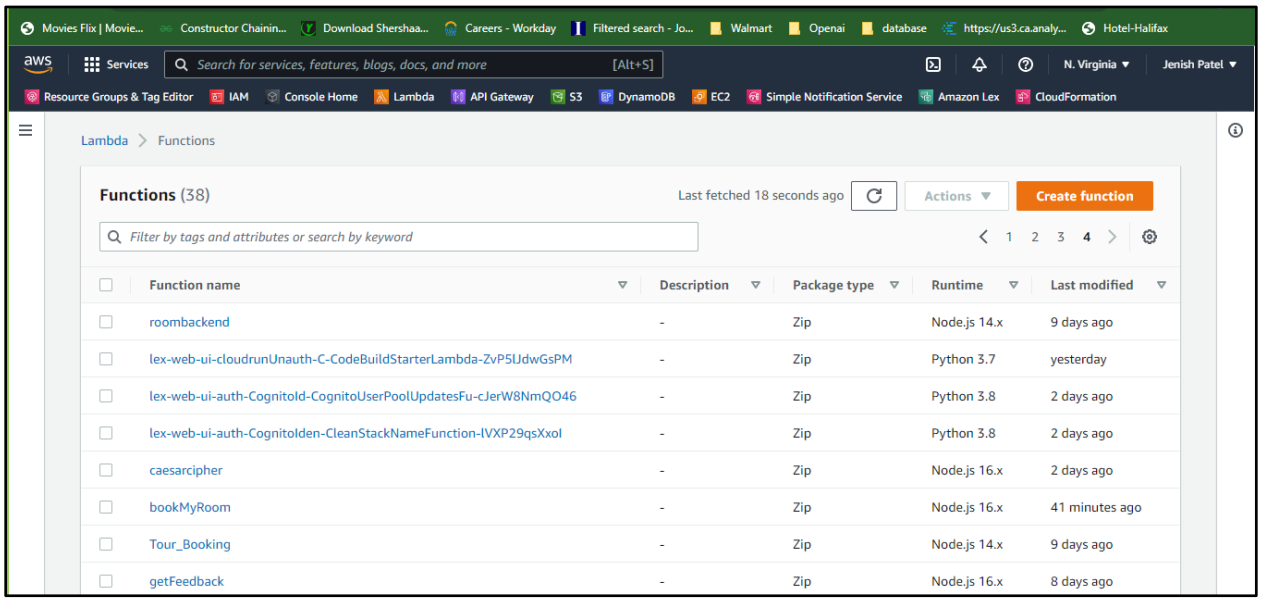

Figure 53 Lambda Functions for all the modules

APIs created in the project:

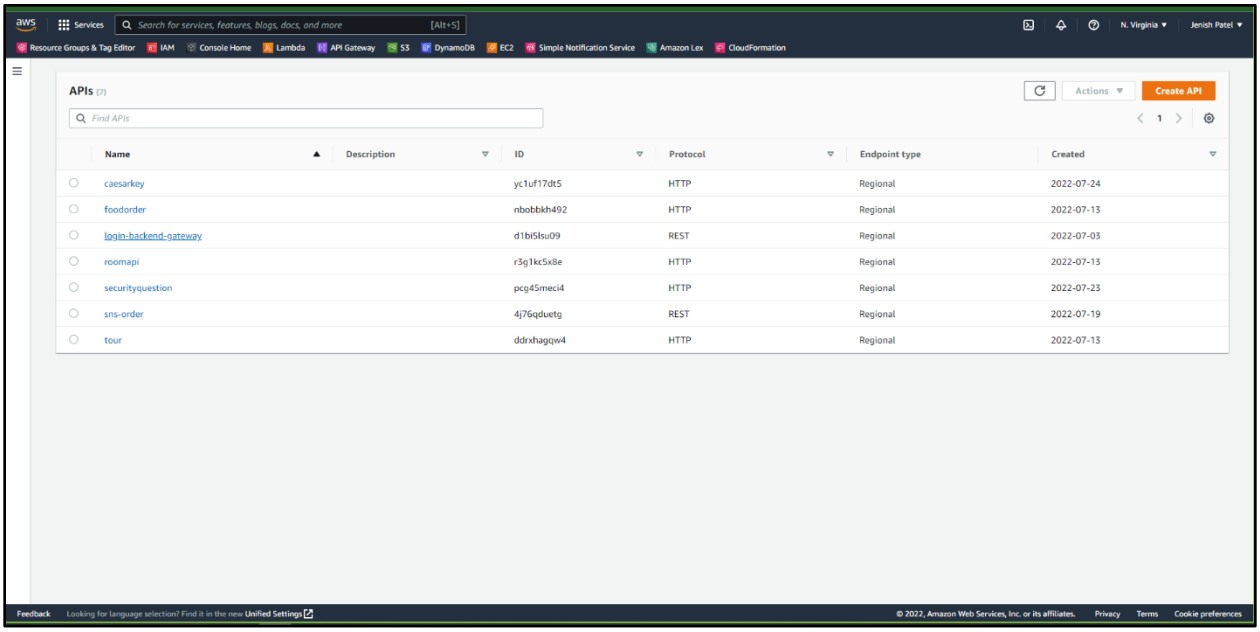

Figure 54 API created in AWS for all the modules

Room Booking Backend Process:

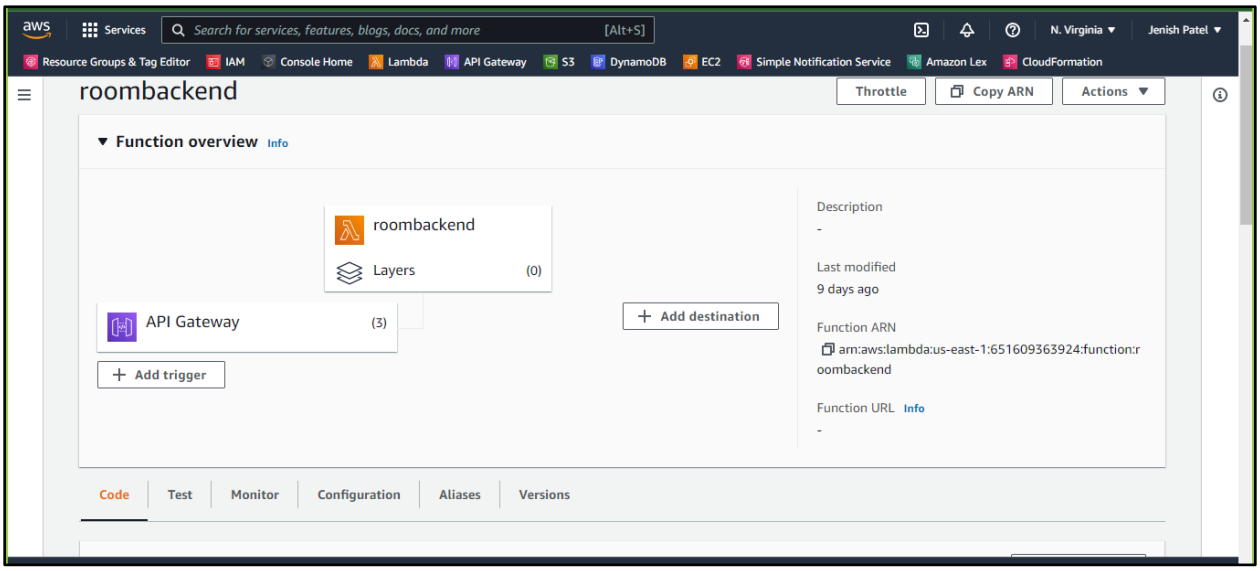

Figure 55 API gateway trigger added for room booking backend lambda function

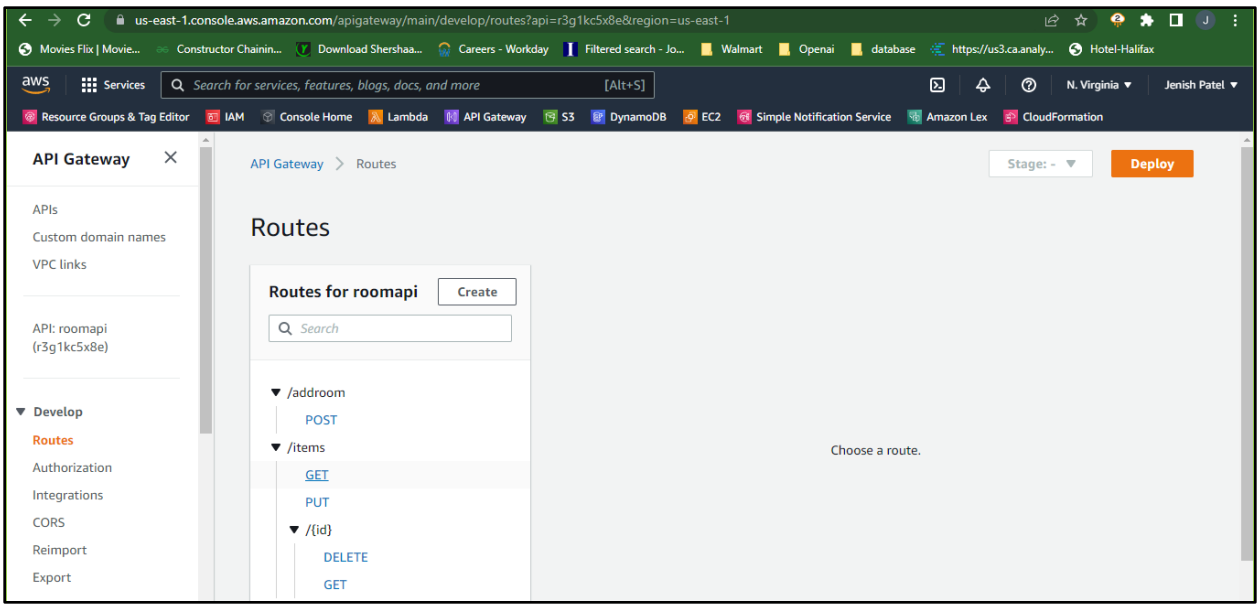

Figure 56 GET, POST, PUT and DELETE API requests for the routes

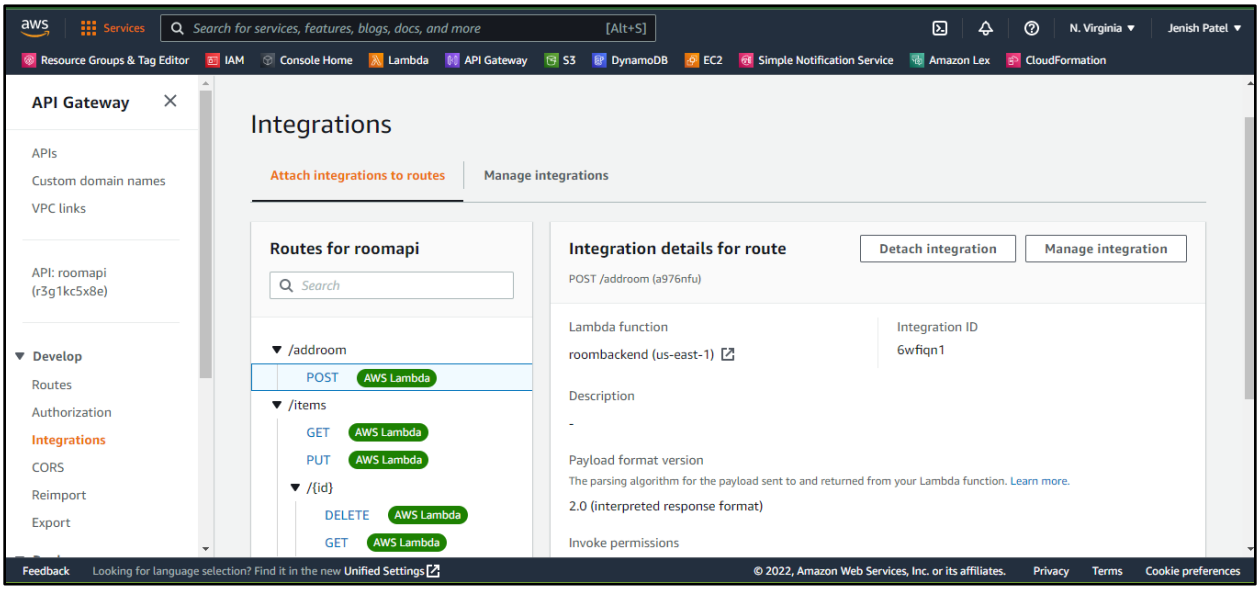

Figure 57 Integrations of the requests with AWS Lambda function

Food Order Backend Process:

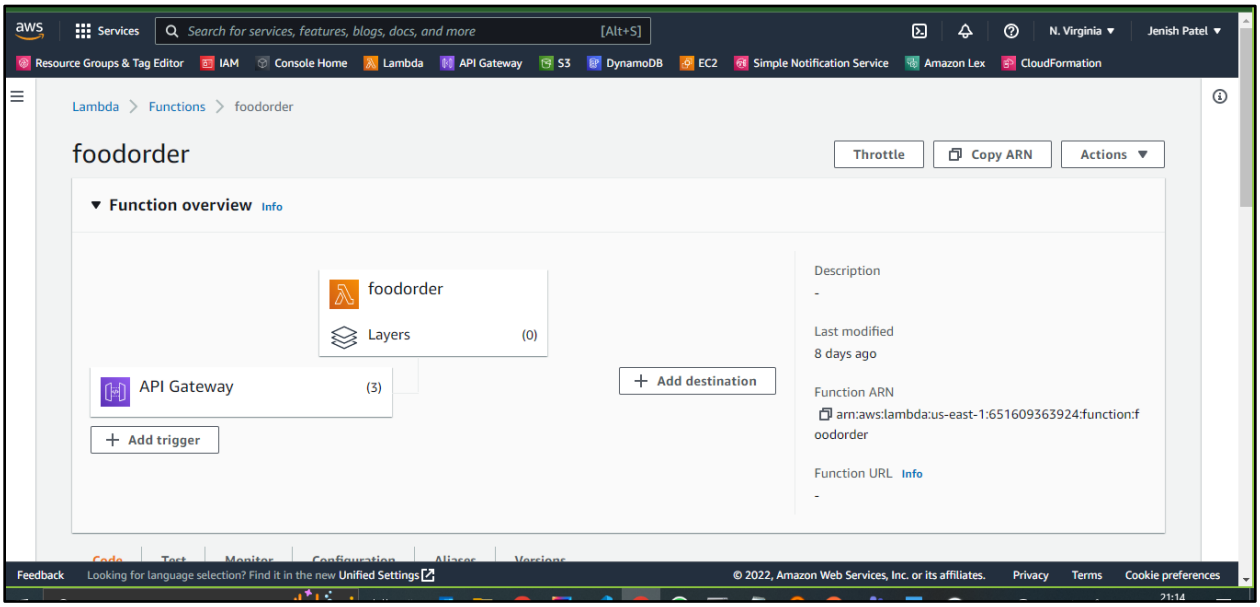

Figure 58 API gateway trigger for food order lambda function

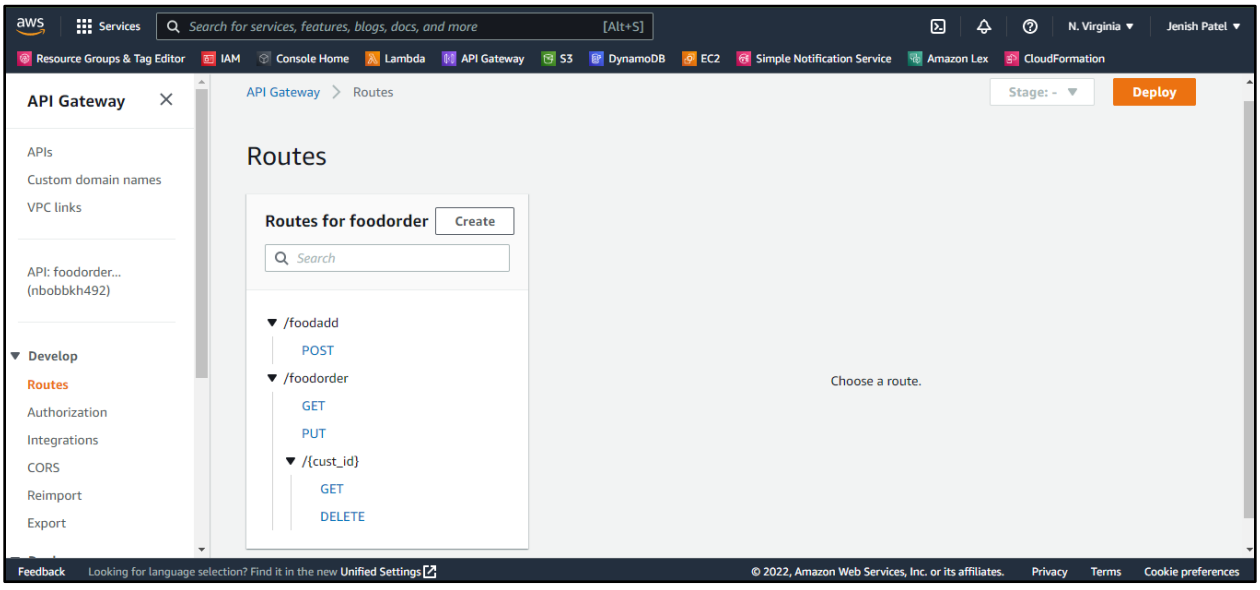

Figure 59 Routes for the food order lambda function API's

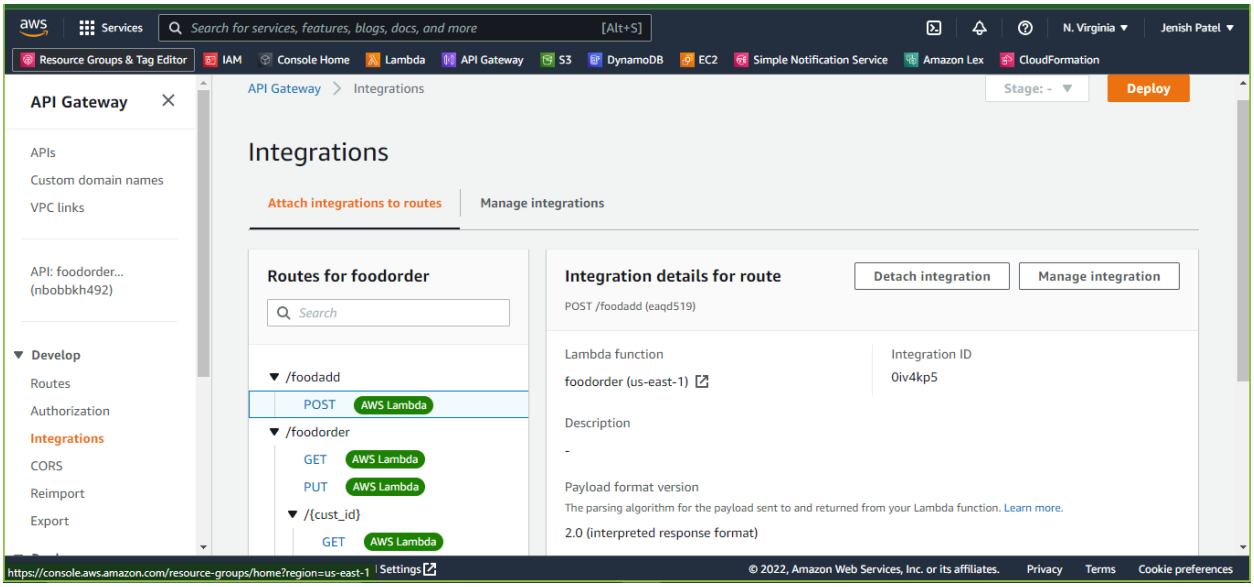

Figure 60 Integrations of the requests with AWS Lambda function in food order

Tour Booking Backend Process:

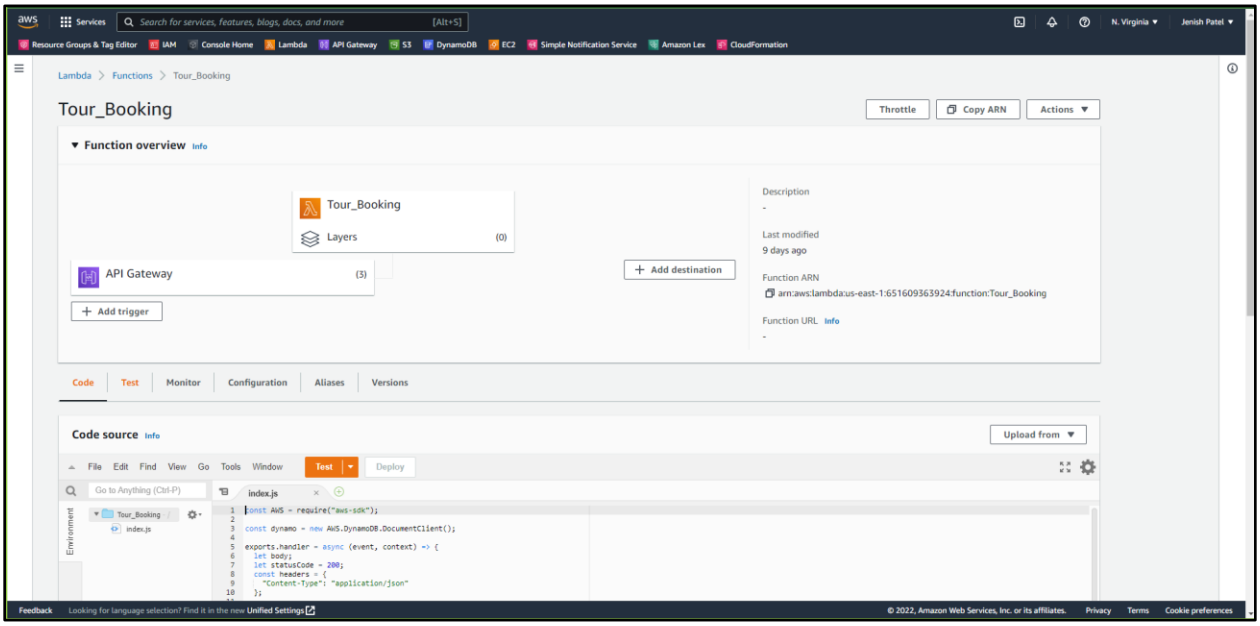

Figure 61 API gateway trigger for the Tour Booking lambda function

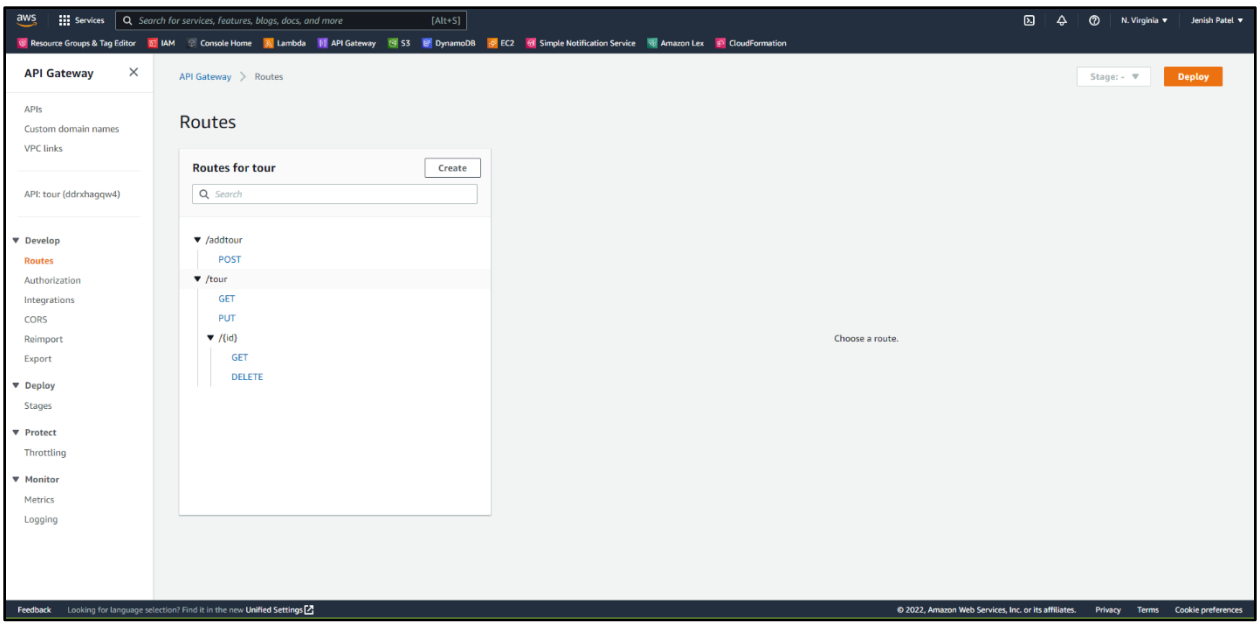

Figure 62 Routes for tour booking

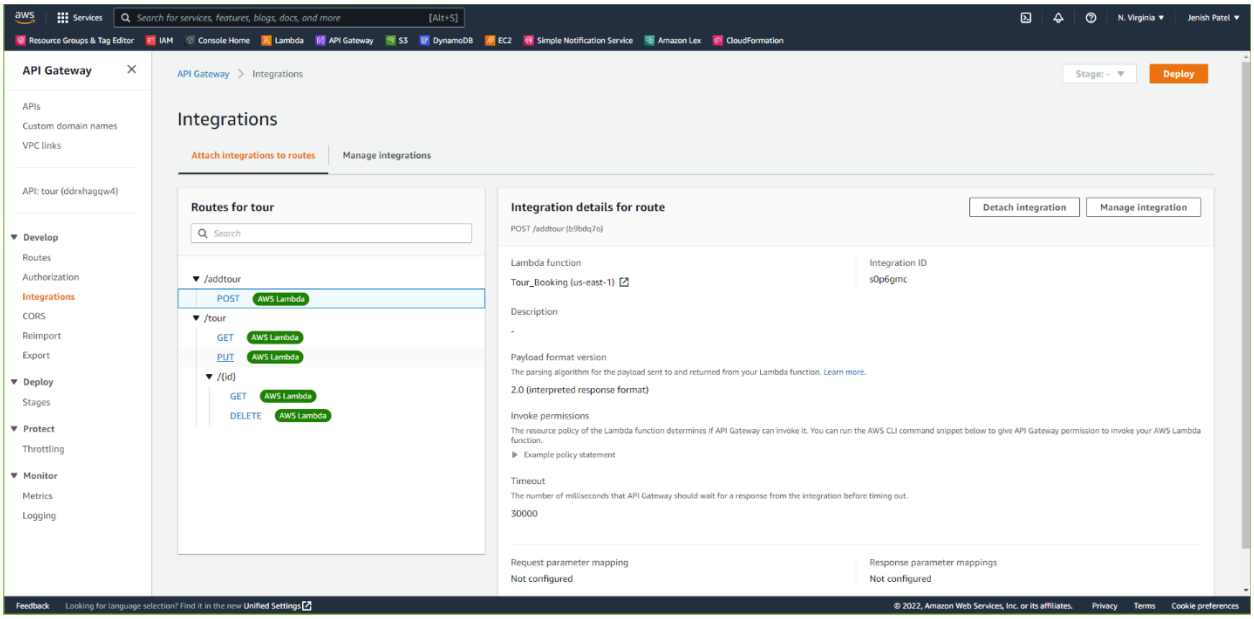

Figure 63 Integrations of the requests with AWS Lambda function in tour booking

Caesar Cipher Backend Process:

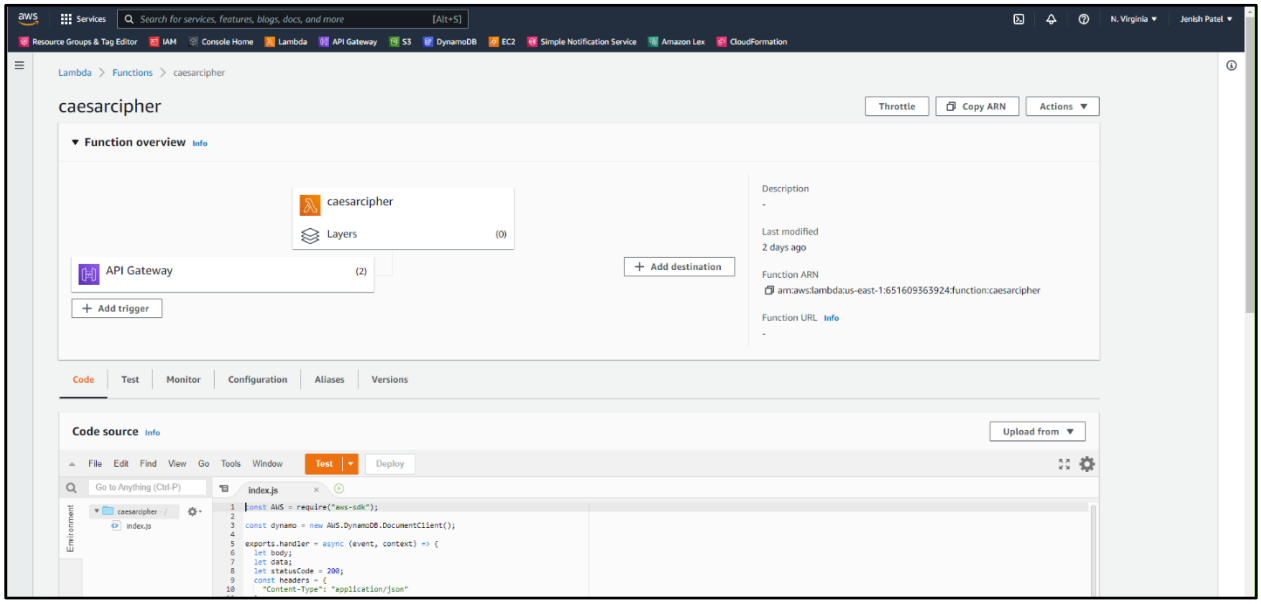

Figure 64 Caesar Cipher Lambda function and API gateway trigger

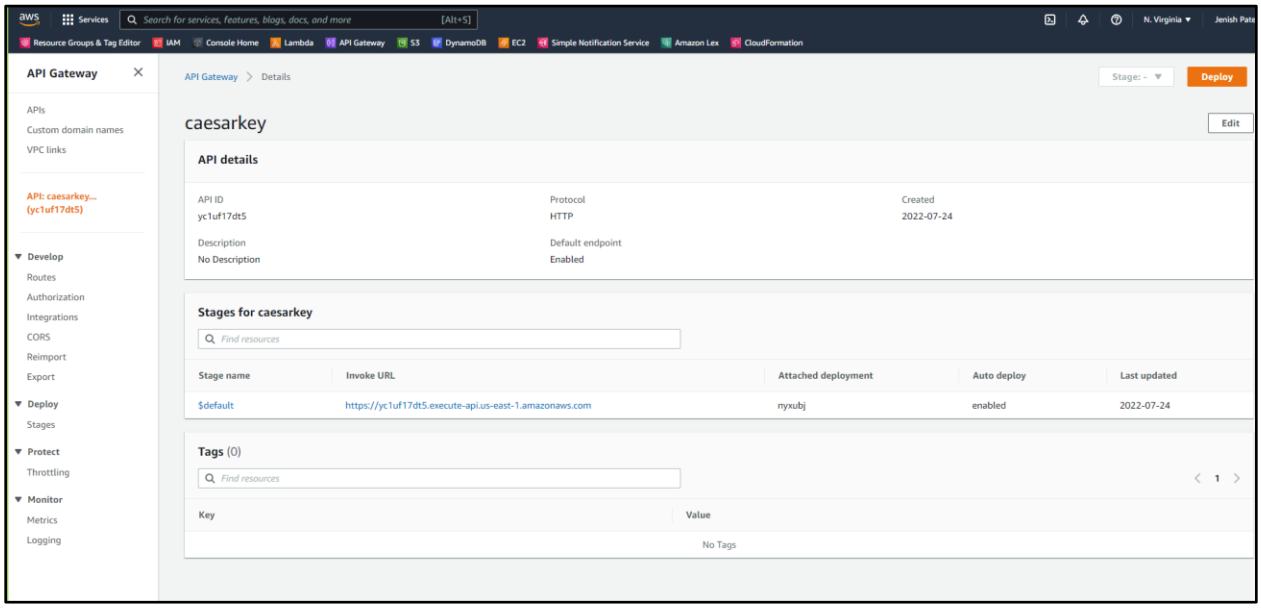

Figure 65 API details of Caesar cipher lambda function

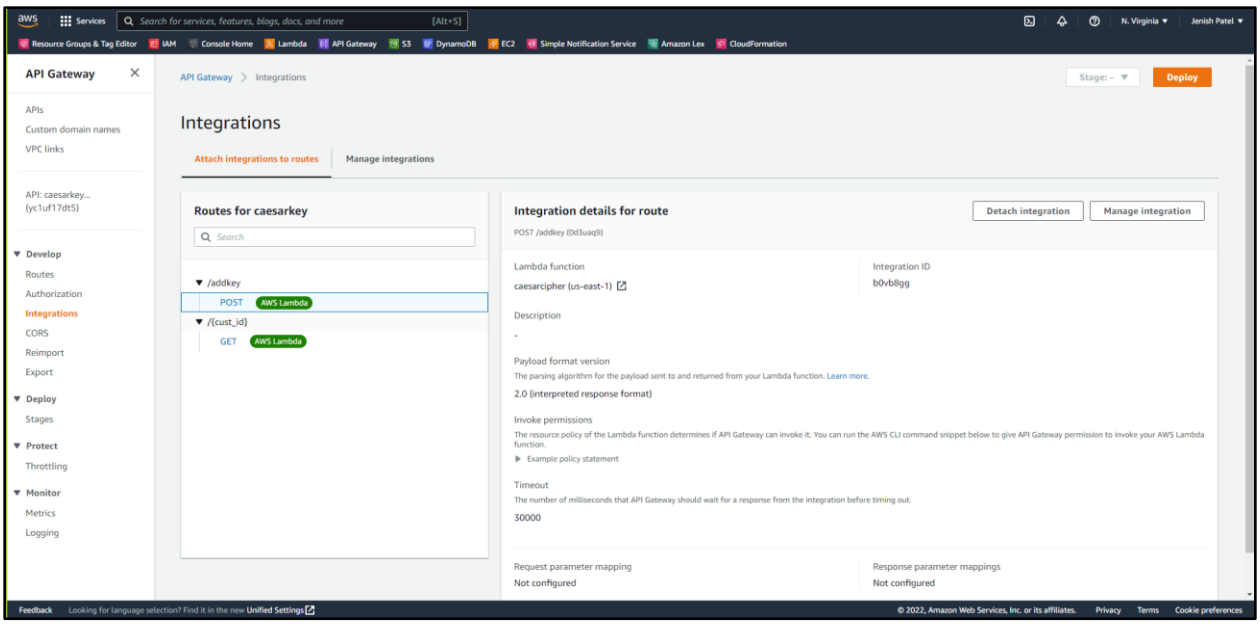

Figure 66 Integrations of the requests with AWS Lambda function in caesar cipher

User Feedback Backend Process:

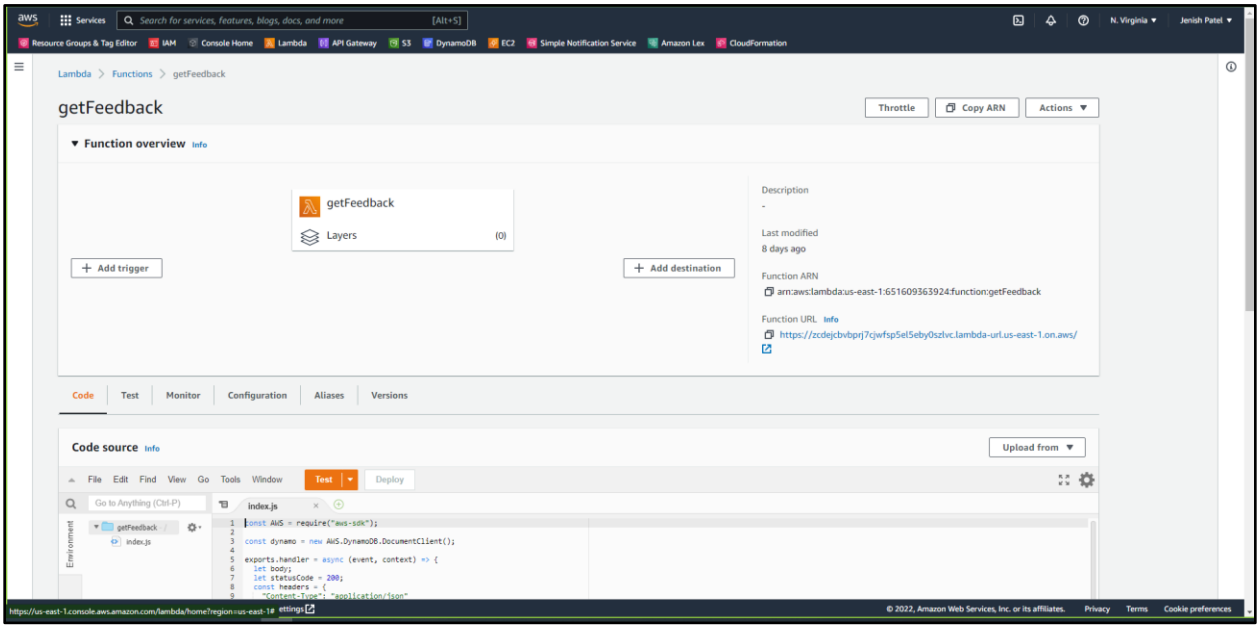

Figure 67 feedback lambda function

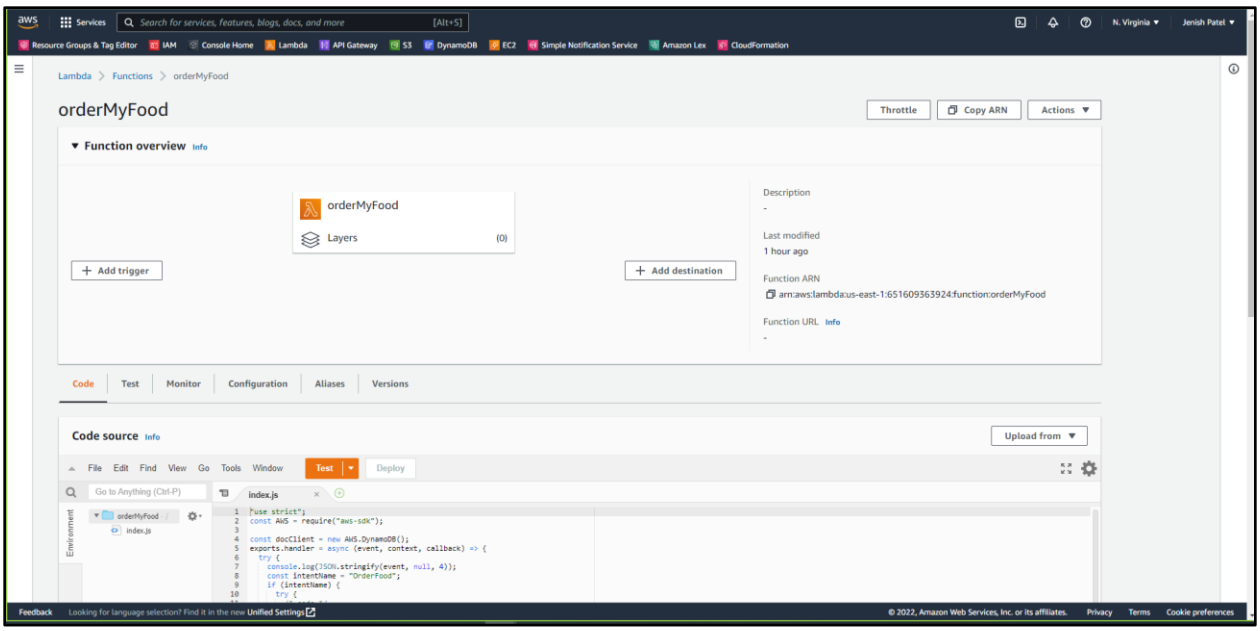

Figure 68 orderMyfood lambda function

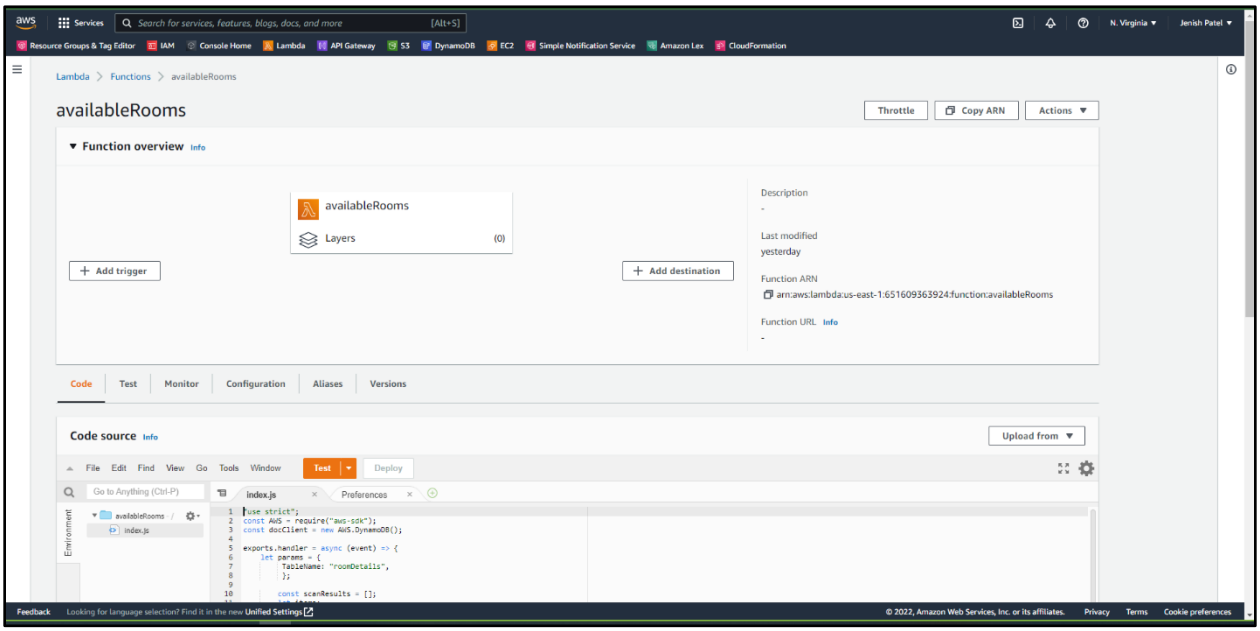

Figure 69 available rooms lambda function

Security Question Answer Backend Process:

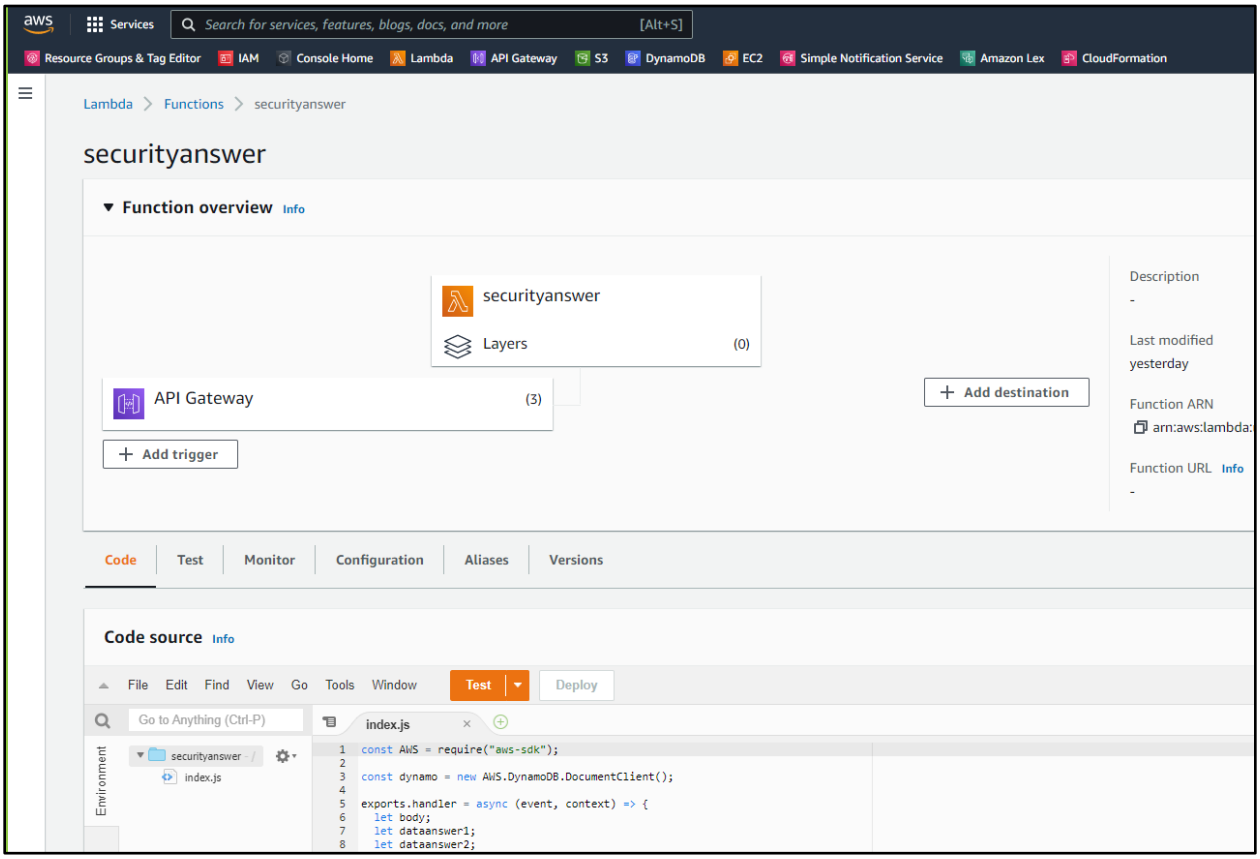

Figure 70 API gateway trigger for security answer lambda function

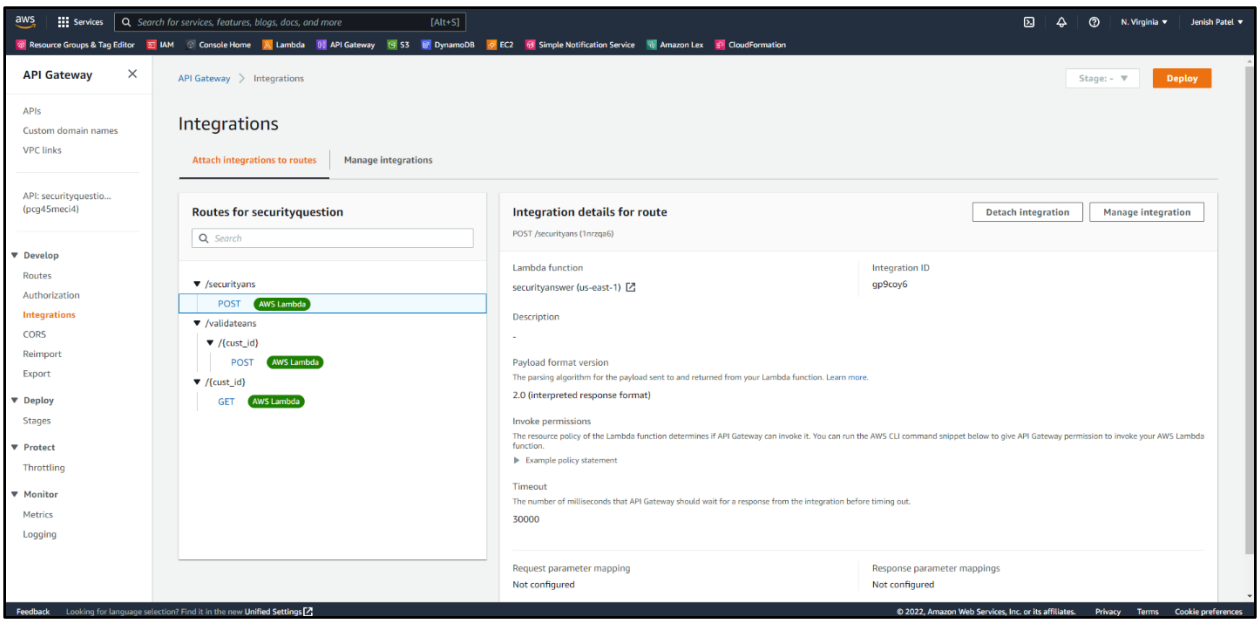

Figure 71 Integrations of the requests with security answer lambda function

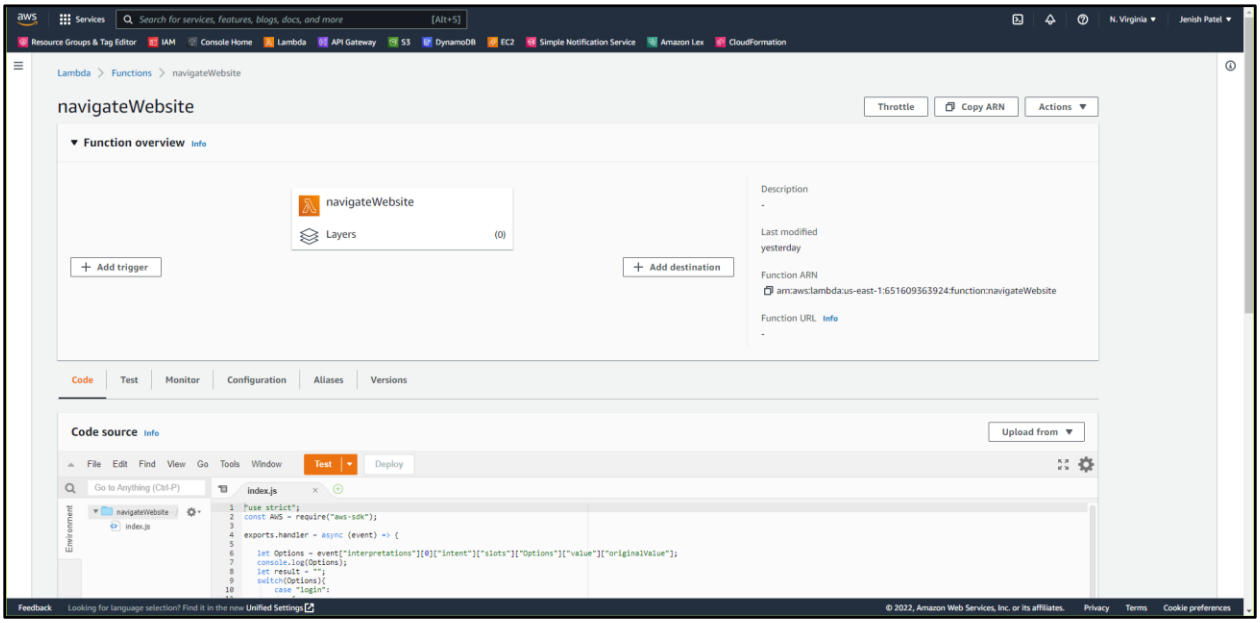

Figure 72 Website navigation lambda function

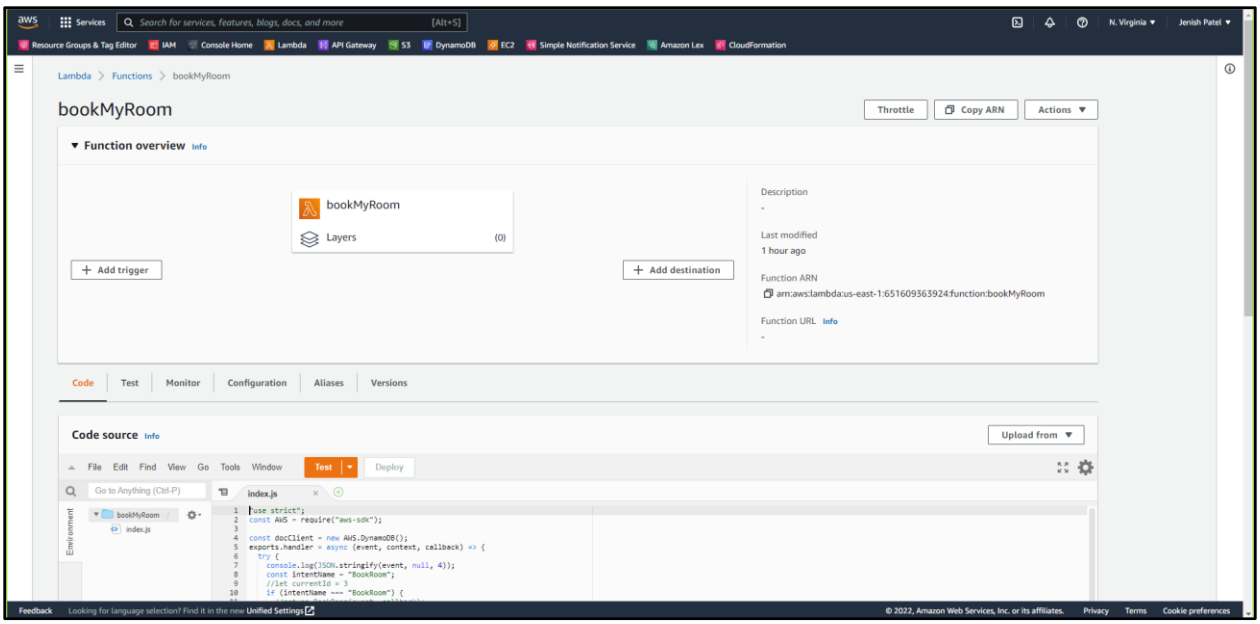

Figure 73 room booking lambda function

Tables stored in DynamoDB for the project:

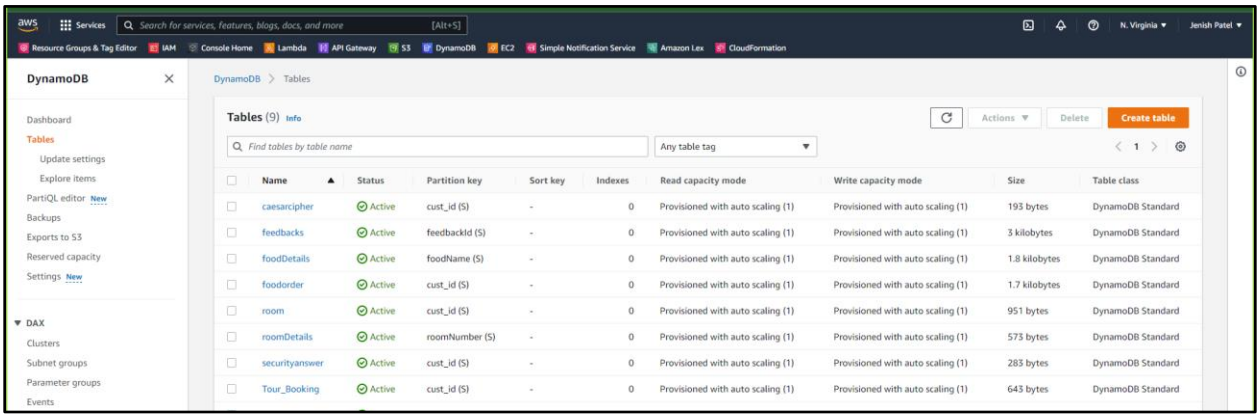

Figure 74 Active tables in DynamoDB

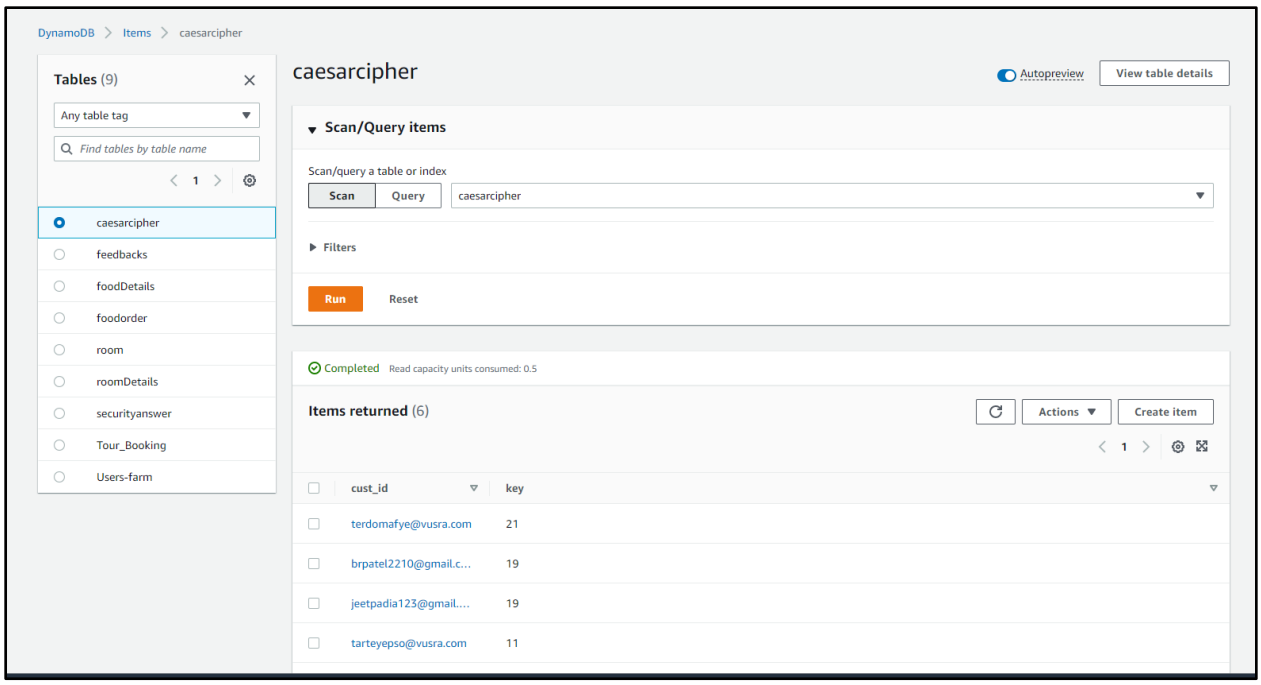

Figure 75 Caesar Cipher table in DynamoDB

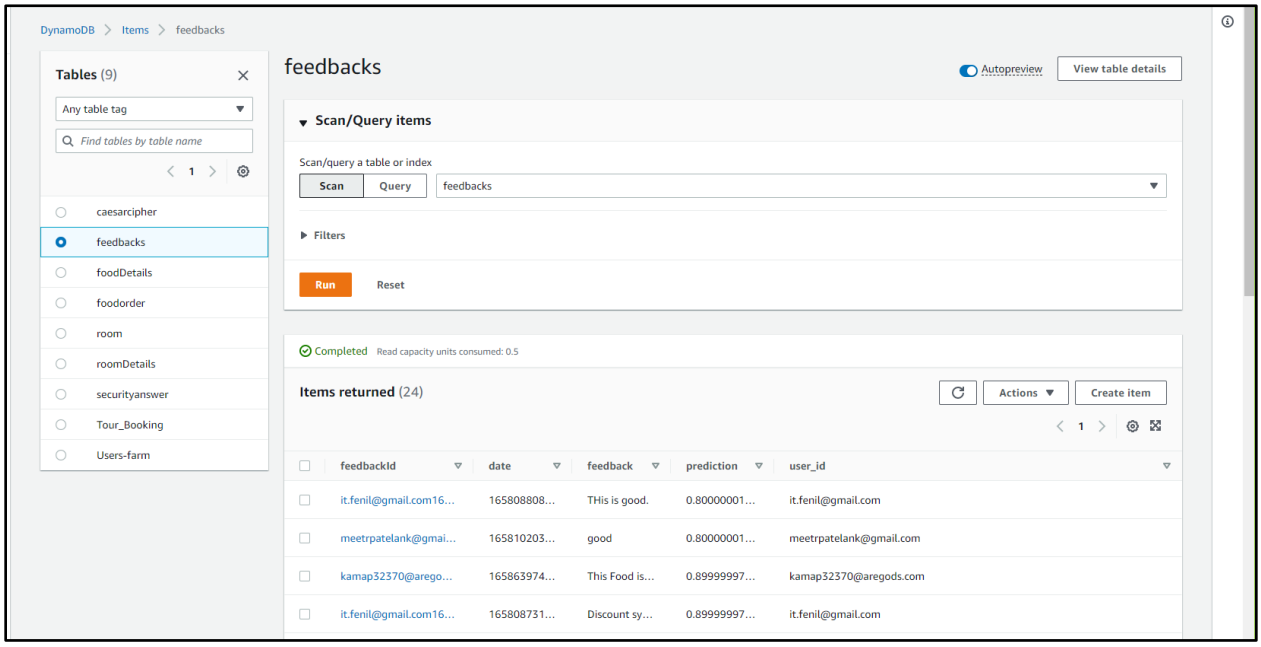

Figure 76 User feedback table in dynamodb

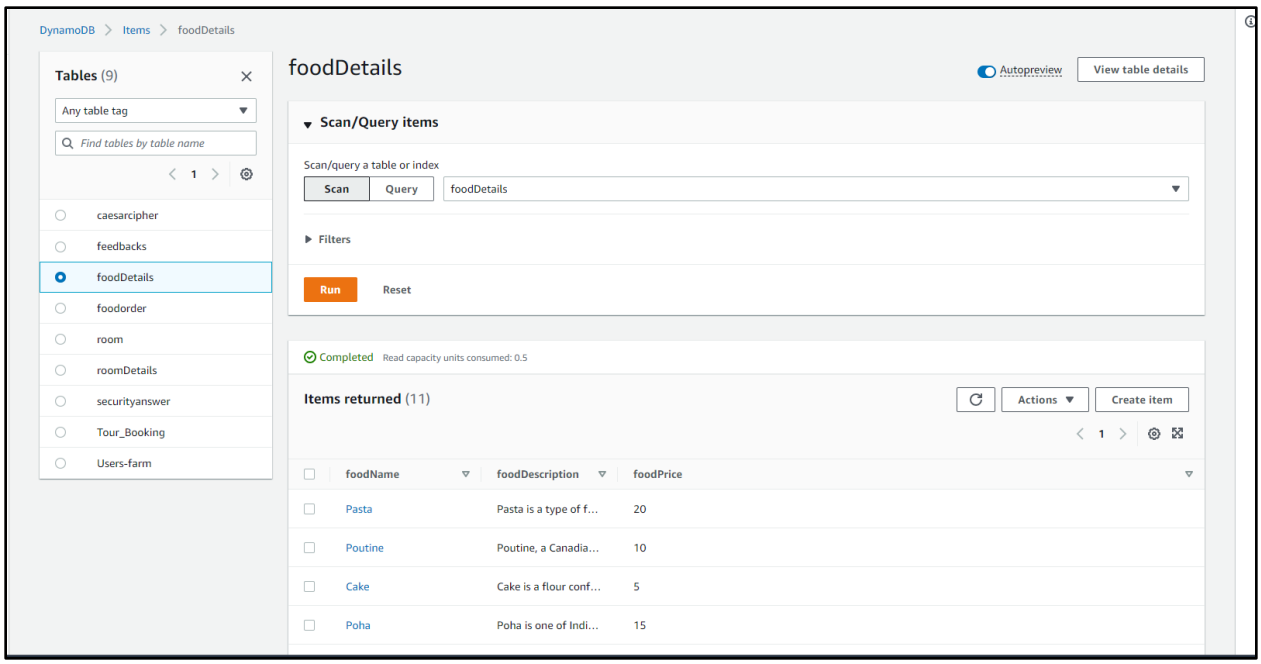

Figure 77 Food details table in dynamodb

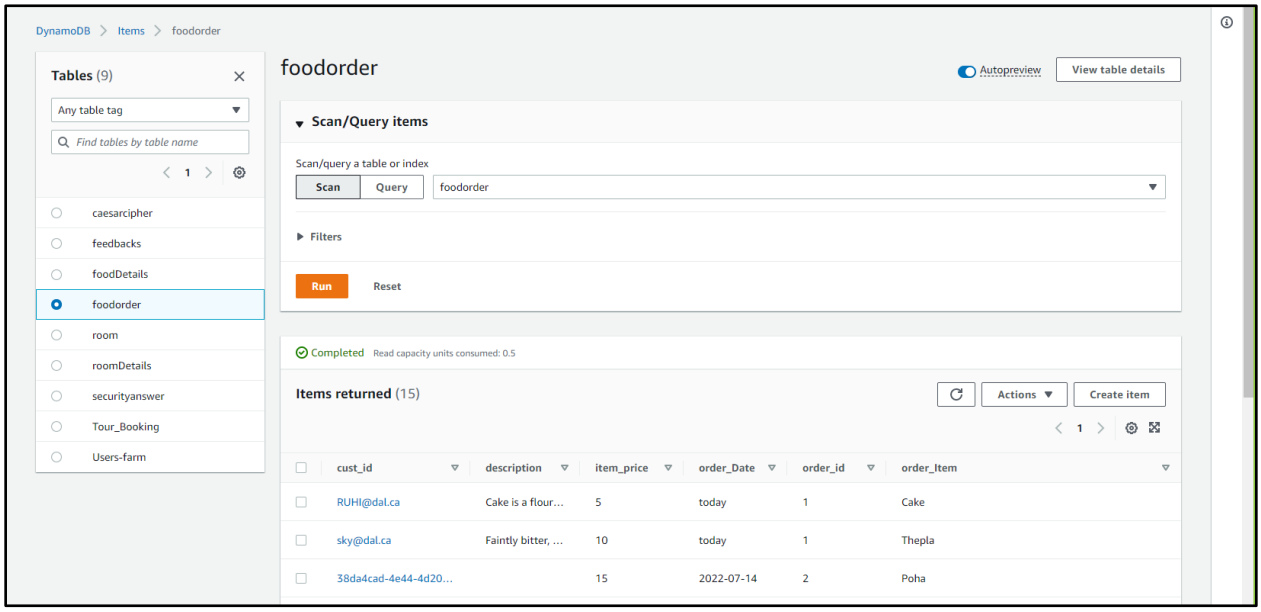

Figure 78 Food order table in dynamodb

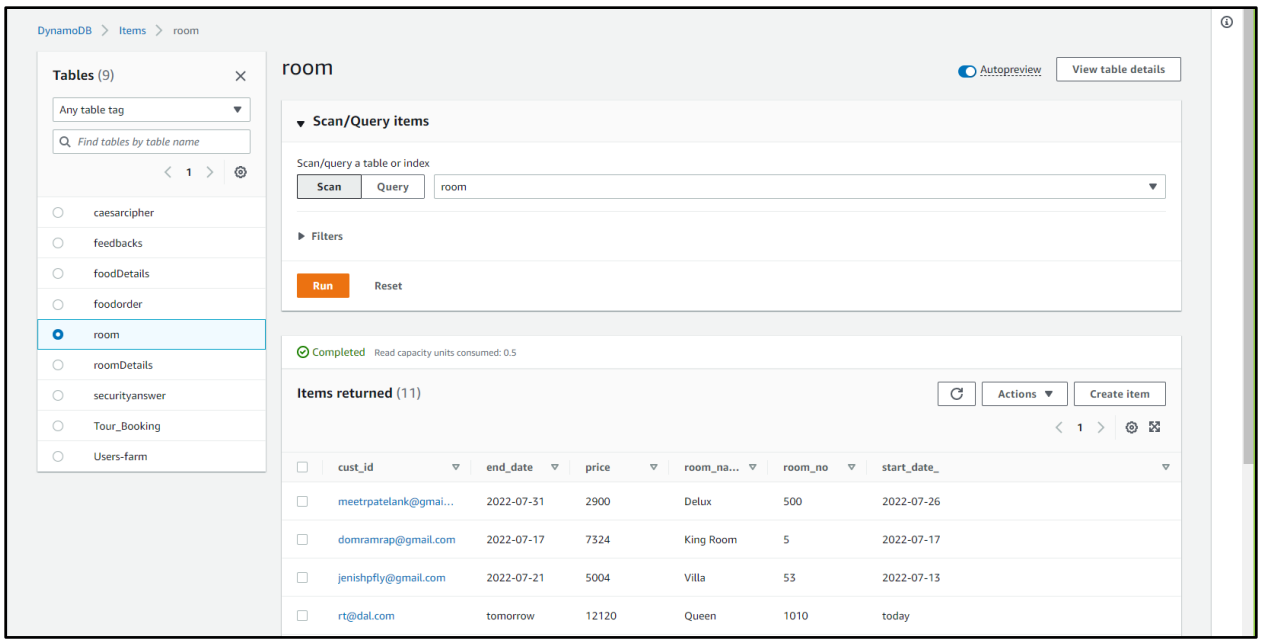

Figure 79 room details table in dymaodb

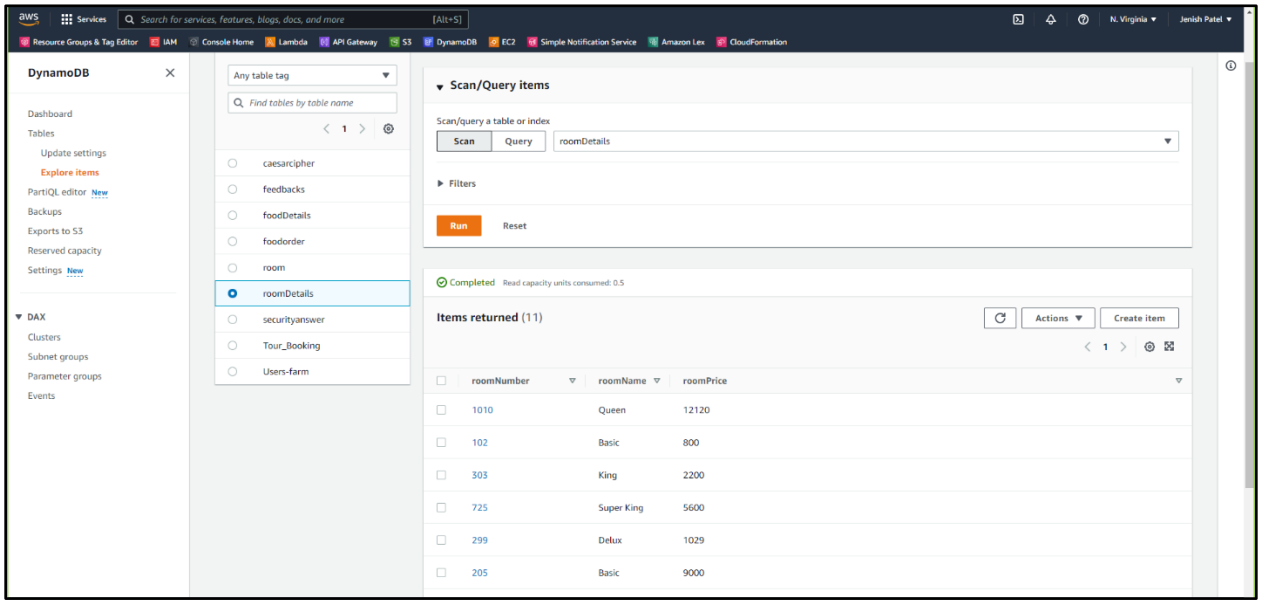

Figure 80 roomdetails table in dynamodb

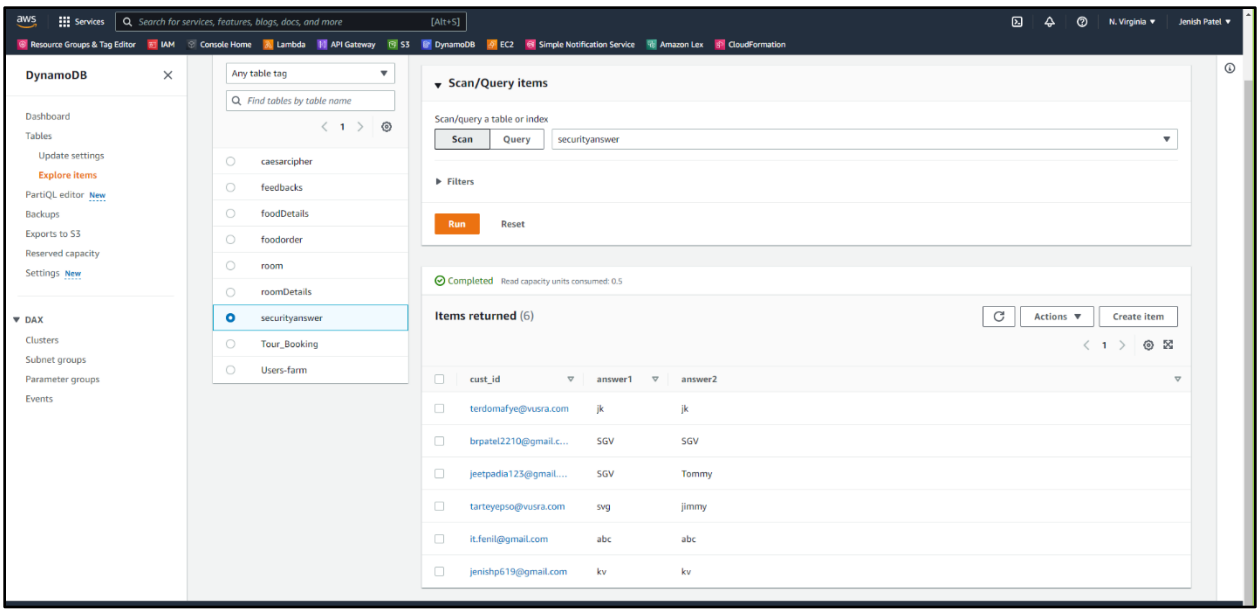

Figure 81 security answer table in dynamodb

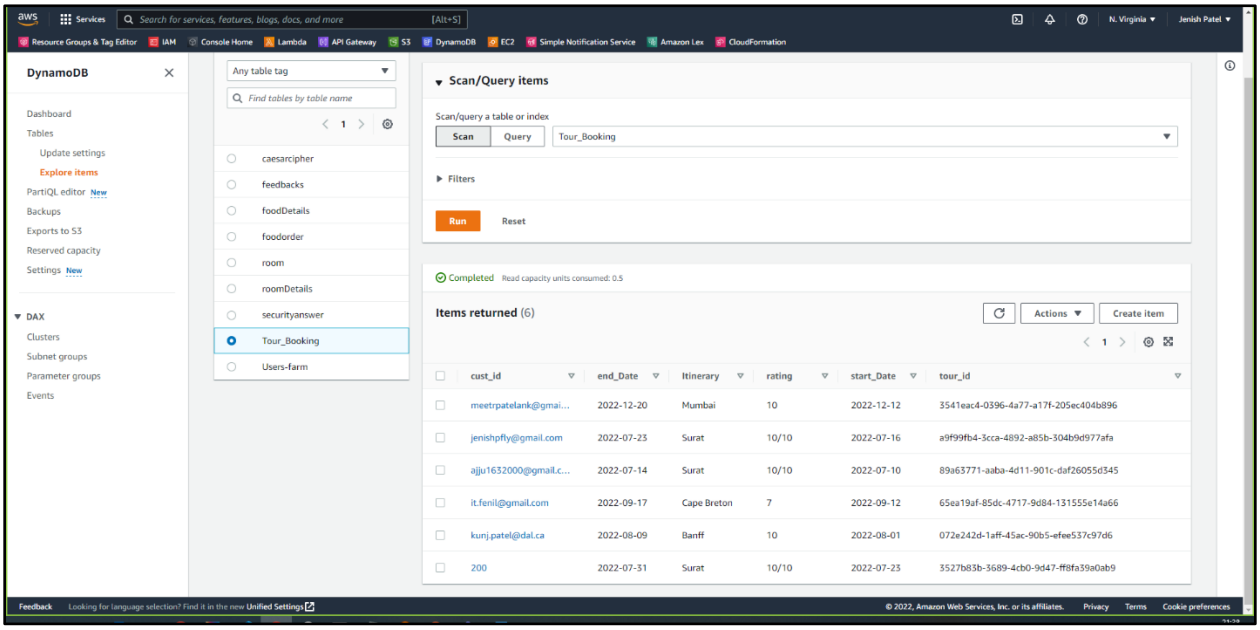

Figure 82 Tour booking details table in dynamodb

9.7 Other Essential Module (Report/Visualization)

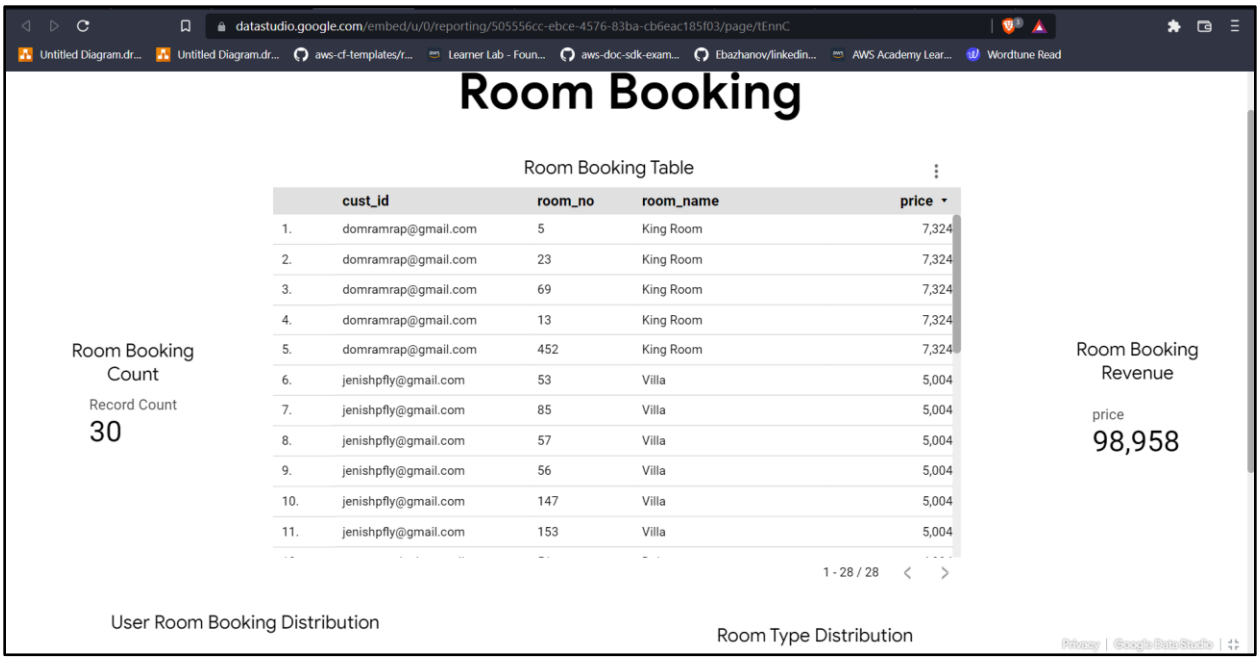

Figure 83 Room Bookings 1

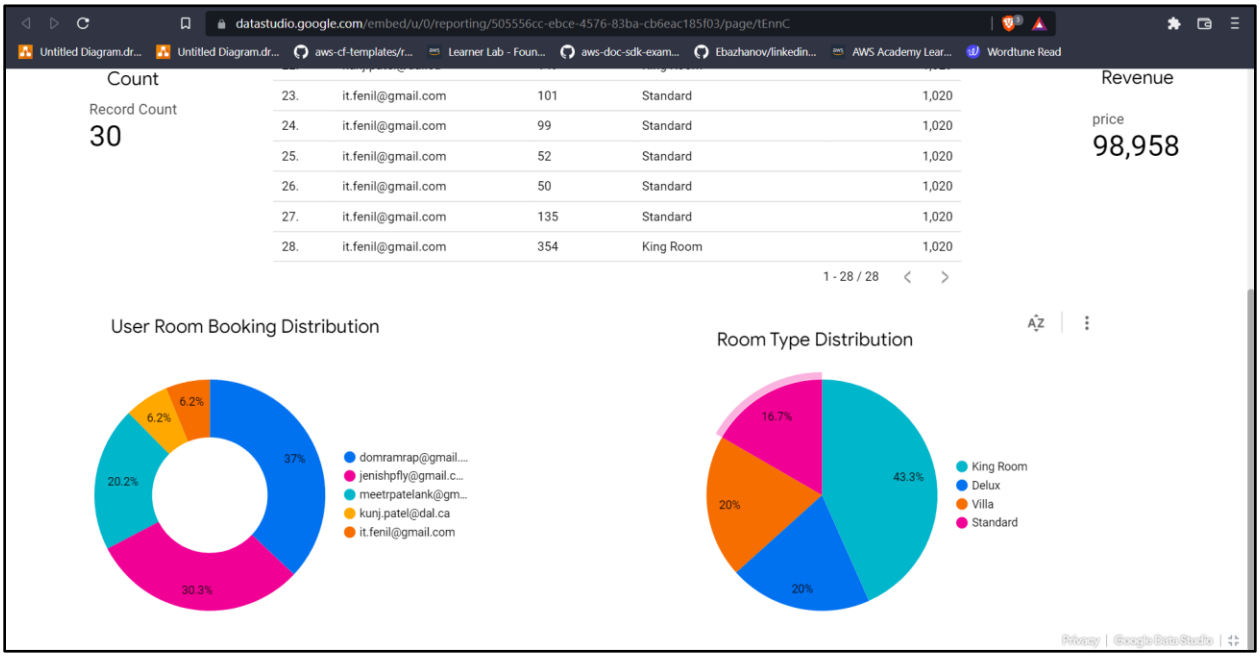

Figure 84 Room Bookings 2

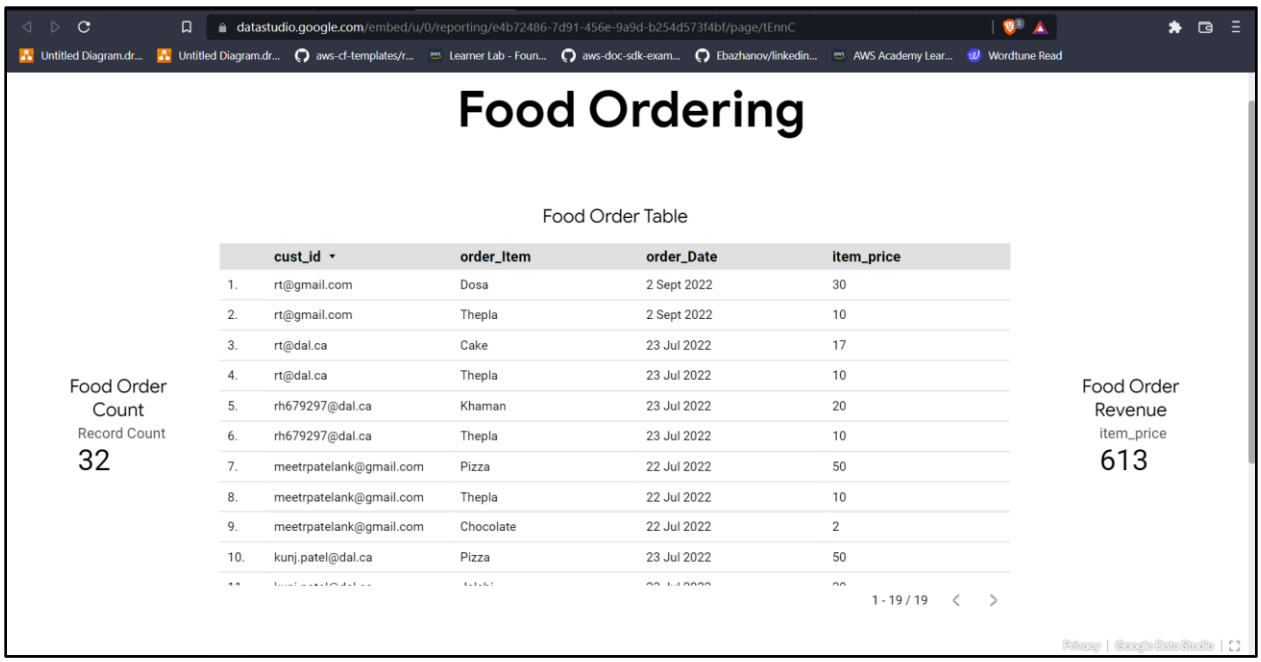

Figure 85 Food Orderings 1

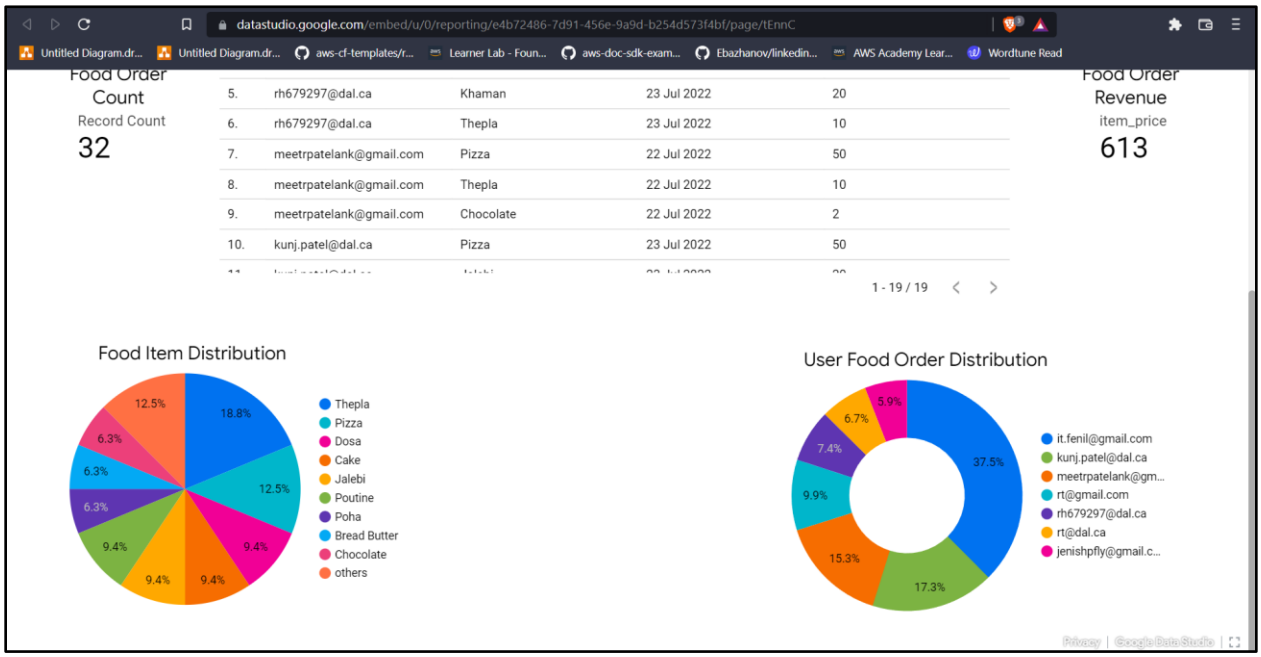

Figure 86 Food Orderings 2

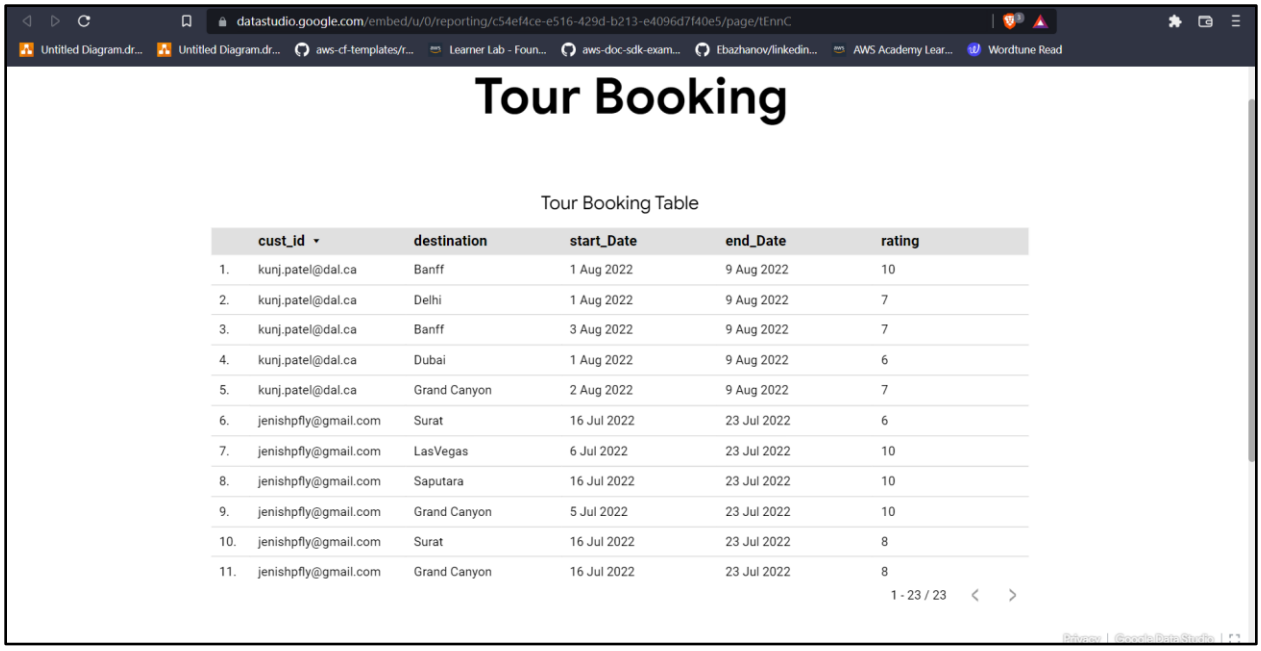

Figure 87 Tour Bookings 1

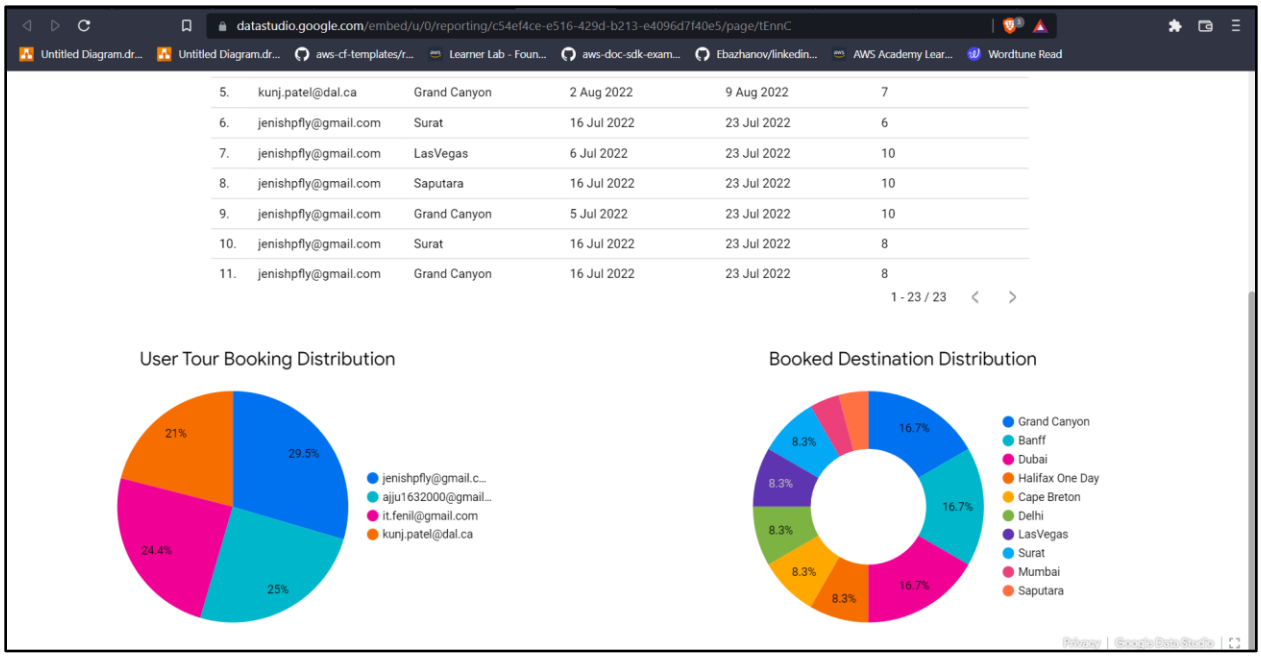

Figure 88 Tour Bookings 2

10. Limitations

Some of the limitations that we have in our application are as follows:

- Currently, we are not providing forgot password feature as it was not required by mentioned requirements documentation. However, if user still want to change their password, they cannot do from our application.

- In our application, manager or admin of the hotel must manually make the rooms available to book. That is quite stressful for admin to keep eye on the checkout data and that is also one of the limitations of our application.

- Even though we are booking rooms in accordance with user requests, we are not showing earlier reservations. The user would only see active bookings on the booking details page as a result.

- To manage sessions after a user registers or logs in, we use the local storage of the browser, which is prone to exploitation. JSON Web Tokens for data transmission across the network stream can be used to improve this.

- In the current implementation, once a user has requested to order food or book a room, we simply accept the request without integrating a payment method that can be used in real-time scenarios.

- At present, we don't currently have an interface for our application where we can show user profile information. As a result, the user lacks access to an interface for updating personal information like their address or phone number.

- For bot, it can perform limited functionalities and the connection of intents is yet to be implemented. Also the chatbot UI needs to be improved matching the website UI.

- Bot is not able to intelligently recognize the user input and trigger the respective intents.

11. References

[1] A Iyengar, K Patel, J Patel,M Patel,R Tyagi,F Parmar "*Project Design Report*," Dalhousie University, [Online document], 2022 [Accessed: Jul. 25, 2022].

[2] "Flowchart Maker & Online Diagram Software", *App.diagrams.net*, 2022. [Online]. Available: https://app.diagrams.net/. [Accessed: 05- Jul- 2022].

[3] "Google Cloud documentation | Documentation", *Google Cloud*, 2022. [Online]. Available: https://cloud.google.com/docs. [Accessed: 05-Jul-2022].

[4] "AWS Documentation", *AWS Docs*, 2022. [Online]. Available: https://docs.aws.amazon.com/. [Accessed: 05-Jul-2022].

[5] "Vertex AI," *Google Cloud*. [Online]. Available: https://cloud.google.com/vertex-ai. [Accessed: 05-Jul-2022].

[6] "AWS Cognito," Amazon.com. [Online]. Available: https://aws.amazon.com/cognito/. [Accessed: 05-Jul-2022].

[7] "AWS DynamoDB," Amazon.com. [Online]. Available: https://aws.amazon.com/dynamodb/. [Accessed: 05-Jul-2022].

[8] "AWS Lambda," Amazon.com. [Online]. Available: https://aws.amazon.com/lambda/. [Accessed: 05-Jul-2022].

[9] "What is Pub/Sub?," *Google Cloud*. [Online]. Available: https://cloud.google.com/pubsub/docs/overview. [Accessed: 05-Jul-2022].

[10] "What is Amazon Lex V2?", Amazon Lex, 2022. [Online]. Available: https://docs.aws.amazon.com/lexv2/latest/dg/what-is.html. [Accessed: 05 - Jul- 2022]

[11] Amazon.com. [Online]. Available: https://aws.amazon.com/quicksight/. [Accessed: 05-Jul-2022].

[12] *Amazon.com*. [Online]. Available: https://docs.aws.amazon.com/apigateway/latest/developerguide/http-api-dynamo-db.html. [Accessed: 20-Jul-2022].

[13] "RPubs - Hotel Demand Dataset," *Rpubs.com*. [Online]. Available: https://rpubs.com/saianne/832882. [Accessed: 26-Jul-2022].

[14] “Sentiment analysis tutorial,” *Google Cloud*. [Online]. Available: https://cloud.google.com/natural-language/docs/sentiment-tutorial. [Accessed: 26-Jul-2022].

[15] “Get online predictions from AutoML models,” *Google Cloud*. [Online]. Available: https://cloud.google.com/vertex-ai/docs/predictions/online-predictions-automl. [Accessed: 26-Jul-2022].

[16] "Deploy a Web UI for Your Chatbot | Amazon Web Services", Amazon Web Services, 2022. [Online]. Available: https://aws.amazon.com/blogs/machine-learning/deploy-a-web-ui-for-your-chatbot/. [Accessed: 22- Jul- 2022]