CSCI-5410 SERVERLESS DATA PROCESSING

FINAL REPORT

GROUP-6

GROUP MEMBERS:

Name	B00####
Kavan Patel	**B00869224**
Navya Jayapal	**B00886554**
Neelansh Gulati	**B00892946**
Prit Sorathiya	**B00890175**
Qiwei Sun	**B00780054**
Samarth Jariwala	**B00899380**

Course Instructor: Dr. Saurabh Dey

Table of Contents

GitLab Repo: https://git.cs.dal.ca/psorathiya/csci5410_group6.git

Application Link: https://serverlessbnb-frontend-7vz2vc6y4a-uc.a.run.app/

1. Project Overview

Our project is a cloud-based hotel management system built on concepts of serverless B&B and multi cloud architecture. The application contains features like selecting tour packages for customers, booking available rooms with food catering services. It also includes message passing services to enable seamless communication between customers and the hotel management. The system provides customers with online virtual support that can swiftly respond to their questions, and the application also enables message transfer capability between some permitted systems.

The system is also equipped with chatbot feature which provides the customers with virtual assistance with anything regarding navigation of website, room availability check for the customer/guest and managing room bookings and ordering food services for authorized customers. The application will be serverless in nature to reduce development and project operating costs. The application will use cloud services from both AWS and Google Cloud Platform. We use the agile methodology to build, test, and deploy the application since it is easy to modify as the requirements change.

2. Cloud Architecture

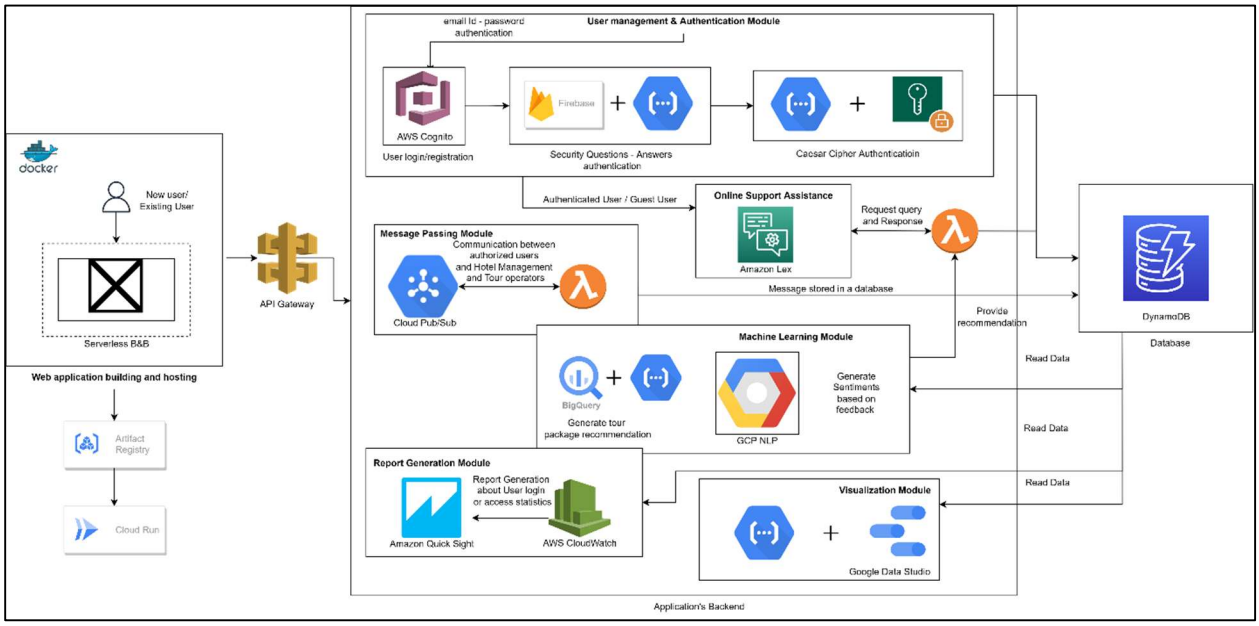


Figure 1 Final Cloud Architecture

The architecture of the Serverless B&B application is depicted in Figure 1. The application

will be composed of a frontend developed using the React.js framework and a backend built with Express.js, Node.js, and various Amazon Web Services [1] (AWS) and Google Cloud Platform [2] (GCP) Services.

Figure 1 depicts the project's final cloud architecture. This project is a serverless web application, and the front end has been built with the ReactJS framework. We have hosted the application using the Google Cloud Run service. Users can use 3-way MFA to register or authenticate in Serverless B&B. (Multifactor Authentication). We have opted to utilize AWS Cognito for user authentication through ID and password, and GCP Firestore for question/answer verification to achieve this feature.

Amazon Lex has been used as virtual support by Serverless B&B. Lex is able to answer queries from users about room availability, reserving accommodations, ordering meals, and so on. Amazon Lex is using AWS Lambda in the background to compute query replies. As a result, that lambda function will interact with Amazon DynamoDB and retrieve the necessary information. Furthermore, message passing is the application's essential functionality, and we have implemented it using Google cloud pub/sub, which functions as a messaging service. All the messages passing, and communications are saved in AWS DynamoDB. .

Also, we have used GCP BigQuery to train a machine learning model and generate predictions using it for providing recommendations to the user about the tour packages. The application also has the functionality to provide feedback, so based on those feedbacks' sentiment analyses have been generated using the GCP NLP service. Google Data Studio has also been utilized to view the data in real-time and produce the Customer booking graph, customer meal orders, and so on.

3. Core Components

3.1 User Management Module

The application has two sorts of users: guests who can see the available rooms and search tours by places and number of days and registered users who, after logging in, may book the rooms, order meals, book a tour and leave comments once they checkout. In our application, we provide a user management interface that allows users to do various operations such as creating an account and signing into an existing account.

AWS Cognito is used for user management because it supports security APIs. It also allows us to create customized email templates [2] for account verification. When a user registers for our program, a customer number is created at random and issued to the registered user. Moreover, we are collecting security question and answer with cipher key from the user during signup. Cognito also synchronizes user data across services. GCP Firestore is used to store and manage user information. The key rationale for using AWS Cognito for user management and GCP Firestore is that they both offer a variety of functionalities. Amazon Cognito allows you to manage all your users in a single location. With the help of AWS

Cognito, we can handle user signup, login, forgotten passwords, and other security features such as multi-factor authentication and compromised credentials.

We are storing security question and answer along with the unique customer id generated, and the cipher key provided by the user during signup in GCP Firestore which will be used to authenticate the user in 3 MFA when the user tries to login into our serverless B&B application.

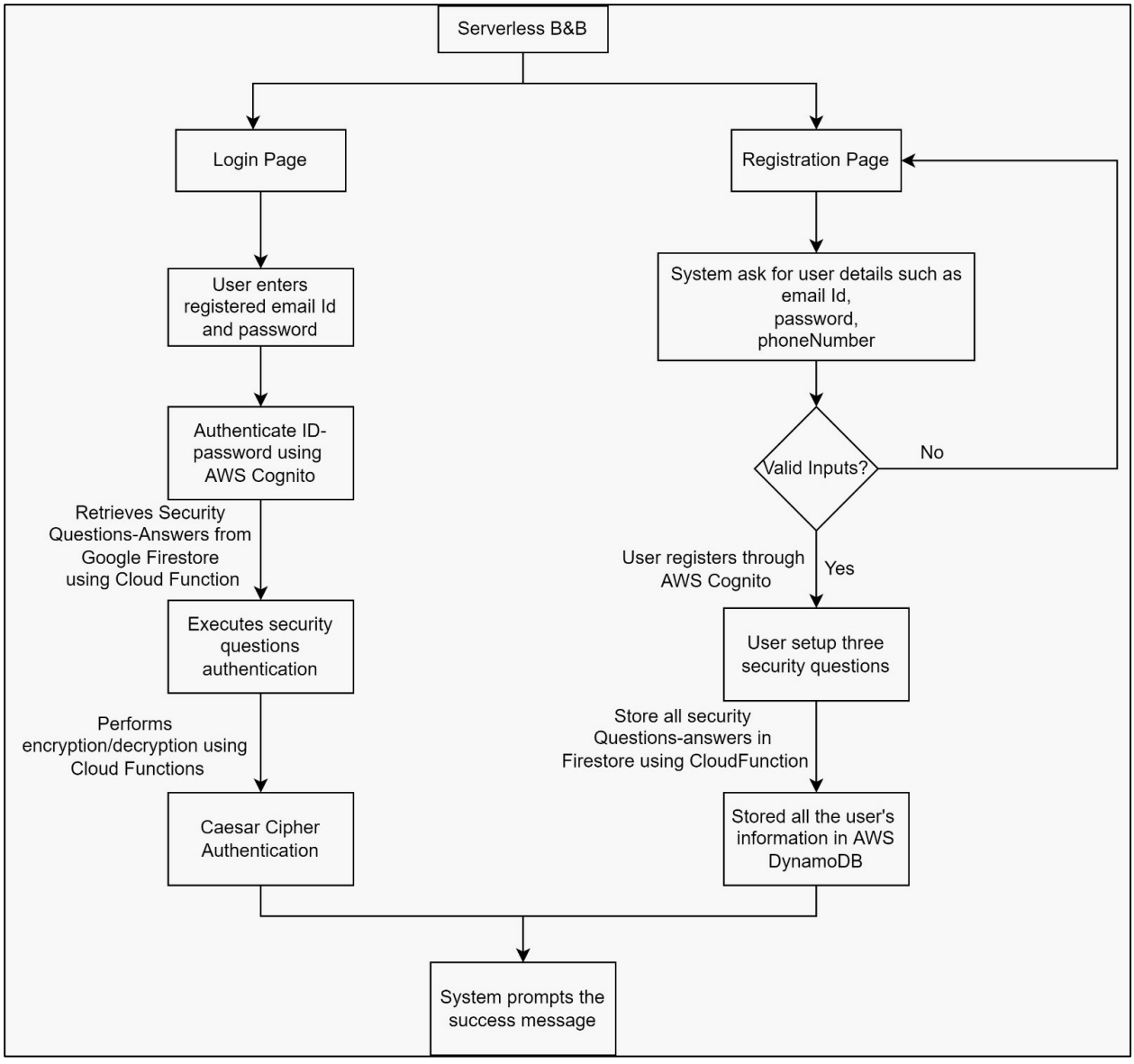


Figure 2 Flowchart of User Authentication and Management Module

If a user is not already registered, they must register in the application by entering information

such as their email address, password, and other data. If the user does not submit the correct information, the user will be routed to the registration page until the user provides all the necessary and legitimate information to create an account. If a user submits valid information, AWS Cognito will store all the user's information. Apart from this, the user must configure security questions and answers, which will be saved in Google Firestore through cloud features.

The application will display a success message confirming that registration was successful and will redirect users to the login page as soon as the details are saved in the database. The login procedure will be divided into three main parts. The user will initially input an email address and a password. The user's email address and password will be validated against data in AWS Cognito. If the user's data is correct, the user will be asked a security question, which will be fetched from Google Firestore using Cloud Function.

Pseudo Code:

Step 1: The user registers by providing their Username, Firstname, Lastname, EmaidID, Password, Confirm Password, Security Question, Answer, and Cipher key.

Step 2: All information provided by the user will be checked in Cognito and GCP Firestore to see if another user with the same username or email is present in user groups and dataset; if so, an error will be displayed to the user; if not, the user data including username, first name, last name, and email will be stored as a new record in Cognito, and username, security question, answer, and Cipher key will be stored in GCP Firestore as a new dataset, and a verification email will be sent to the user's email.

Step3: Cognito sends the user a verification link, which the user clicks on to verify the account.

Step 4: The user will enter a registered username and password to access the web application. In Cognito, this username and password entered by the user will be validated. If the authentication is successful, the user will be routed to the second step of 3FA, where they must input a response to the shown security question.

Step 5: The user will click the forgot password link to reset their password.

Step 6: On the forgot password page, the user will input their registered email address in the field. The user's email address will be validated in Cognito; if it is invalid, an error message will be displayed; otherwise, the user will be forwarded to the reset page and a verification code will be sent to the user's email address.

Step 7: To reset the password, the user will input the verification code sent by Cognito via email, as well as the email and new password.

Screenshots and Test Cases:

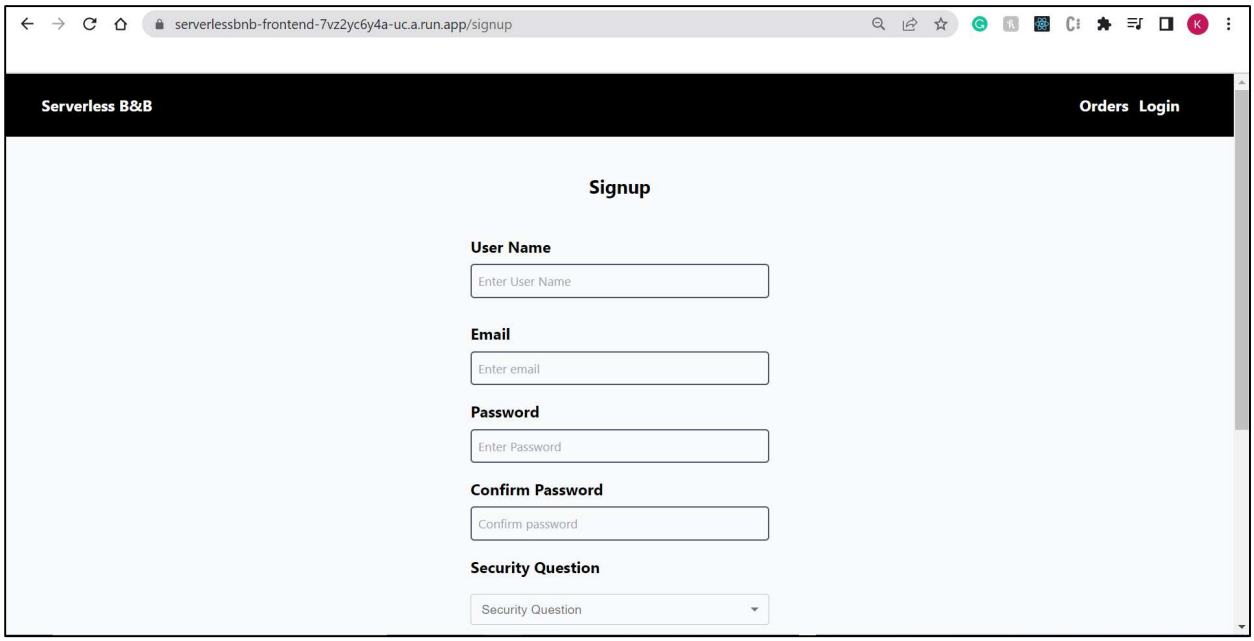


Figure 3 Signup Page

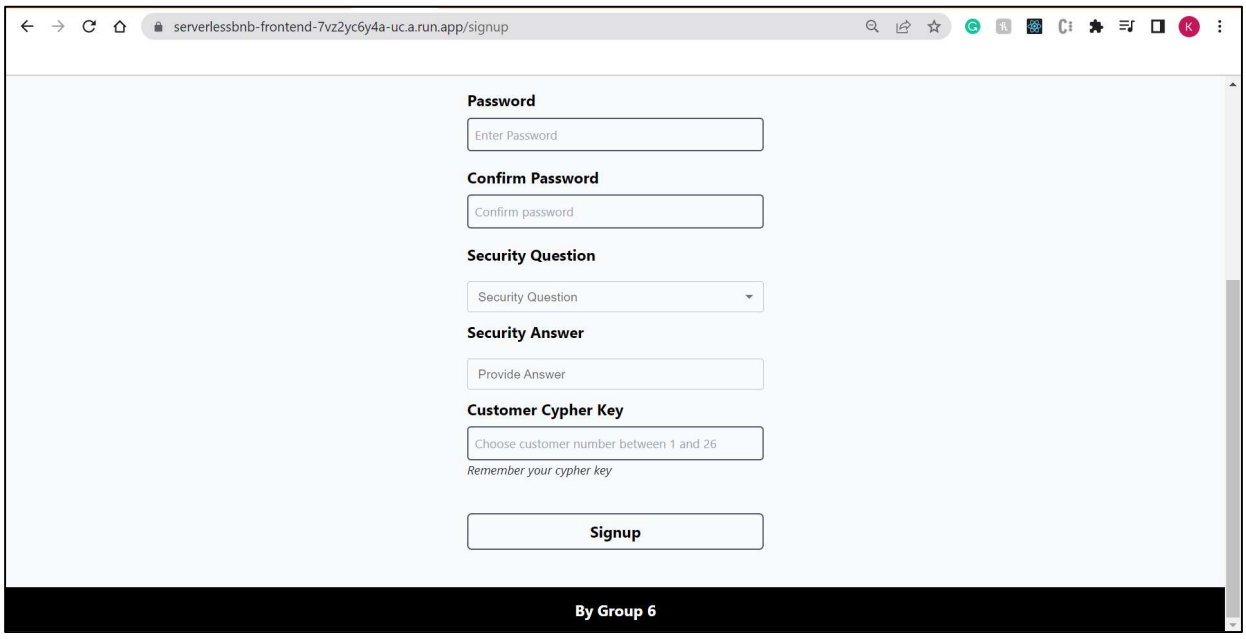


Figure 4 Signup Page

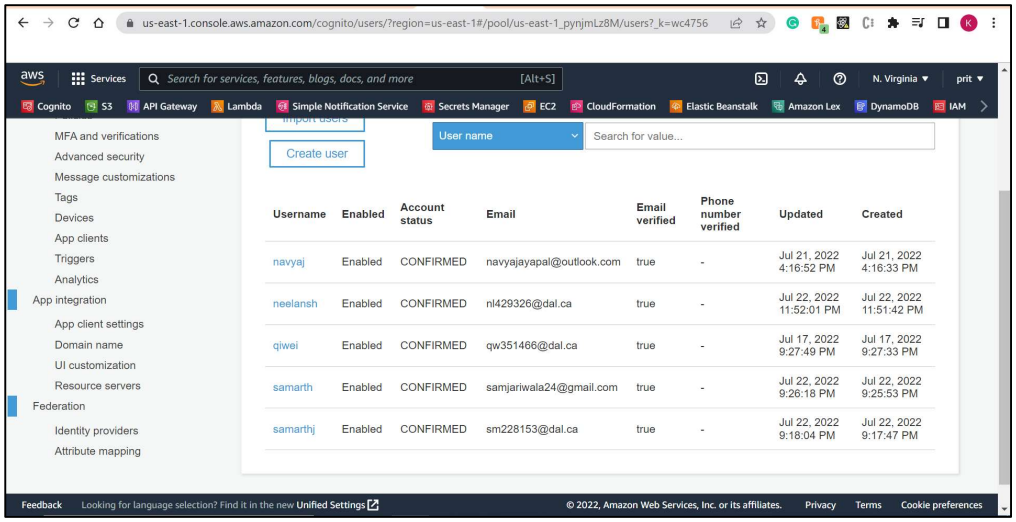


Figure 5 No user named "kavan" in cognito

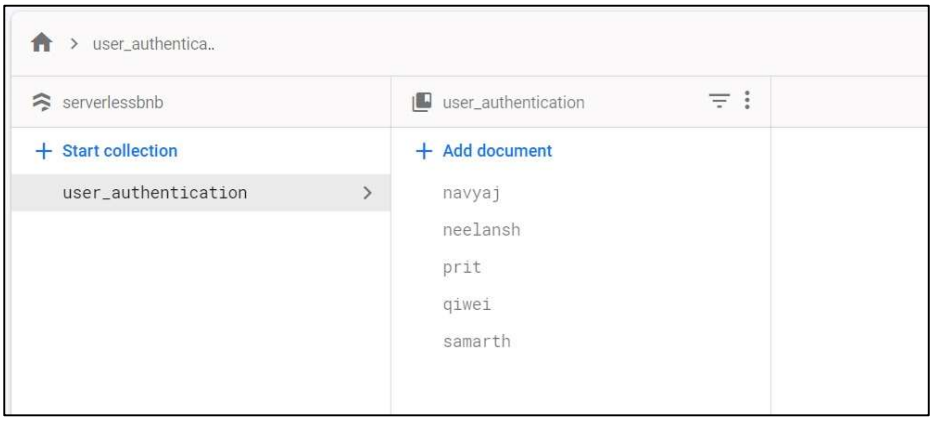


Figure 6 No user named "kavan" in Firestore

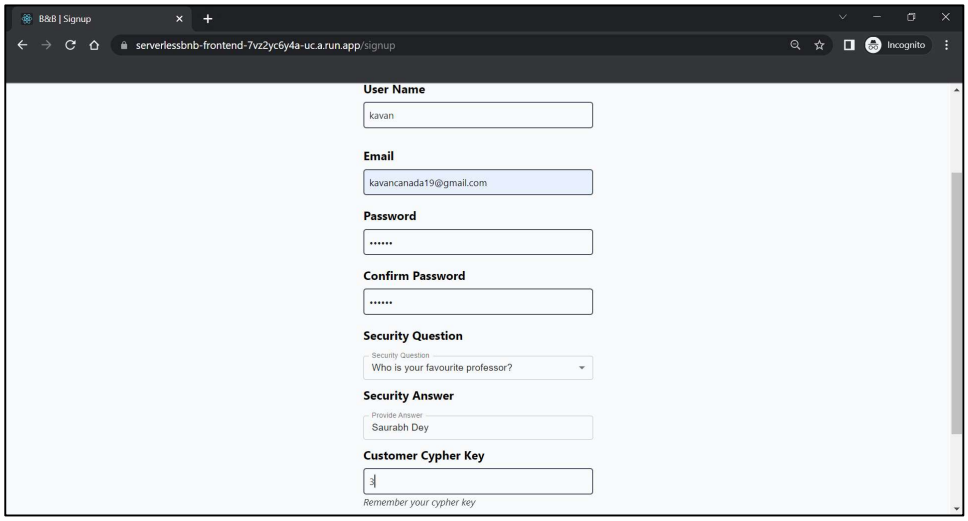


Figure 7 Registering new user with the above details

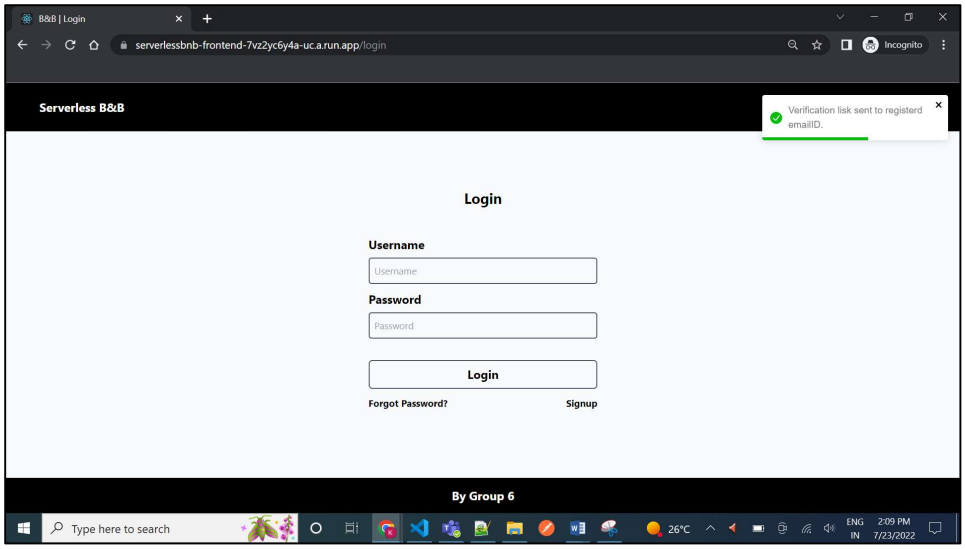


Figure 8 Verification link sent successfully

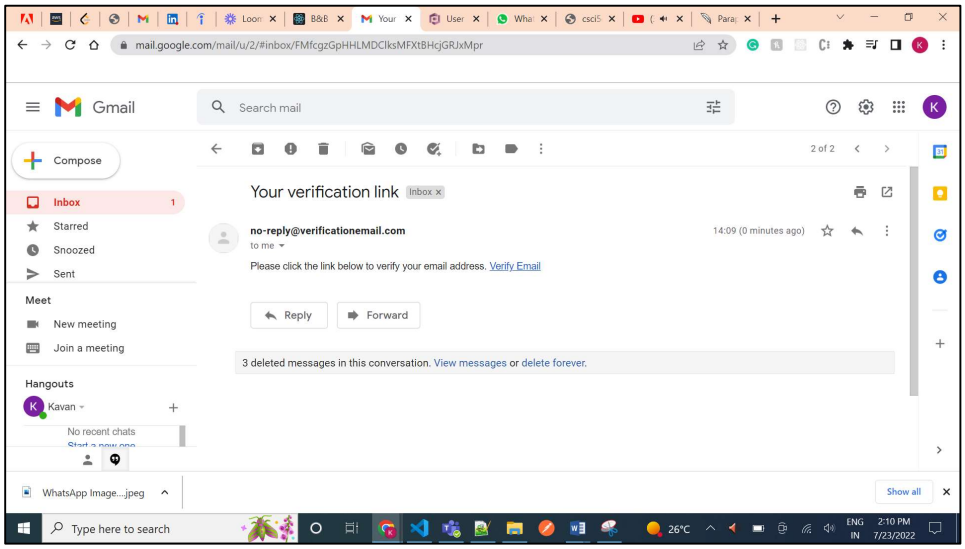


Figure 9 Mail received in Gmail account

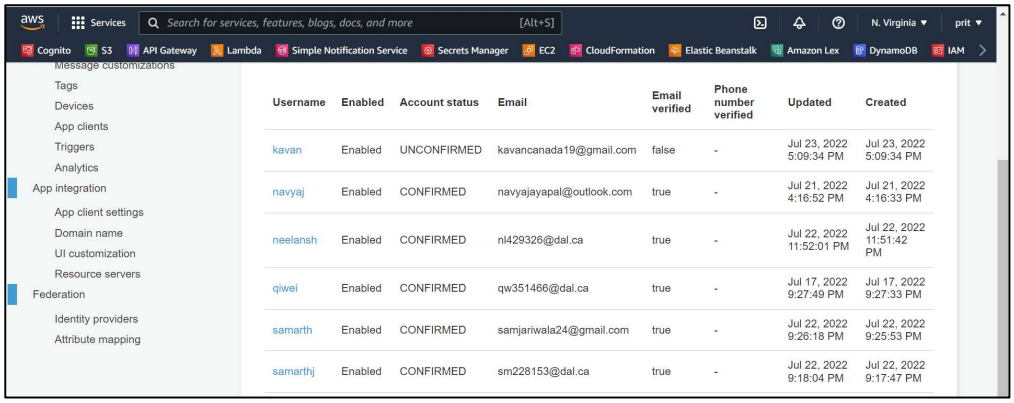


Figure 10 : Details of user "kavan" stored successfully in Cognito but Account Status for user "kavan" is unconfirmed as I have not verified the account yet

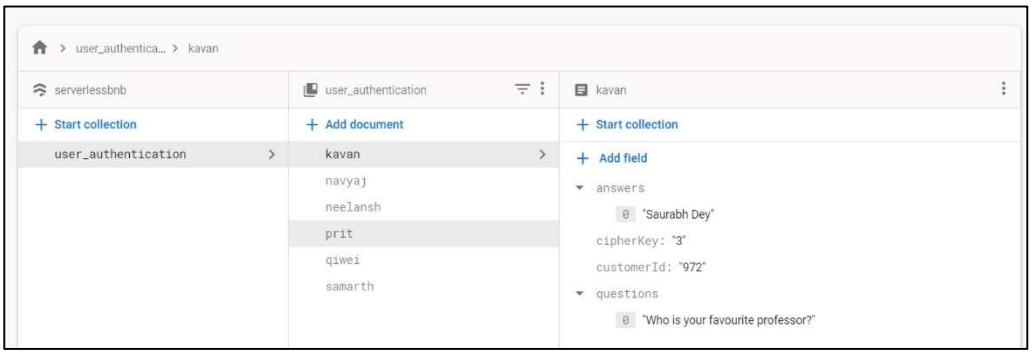


Figure 11 Details of user "kavan" stored successfully in Firestore

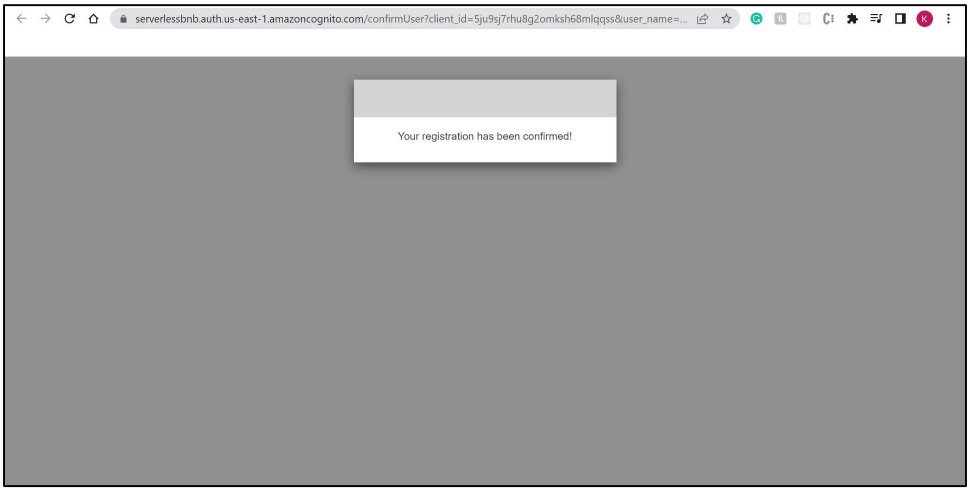


Figure 12 Account verified successfully by clicking on Verify link

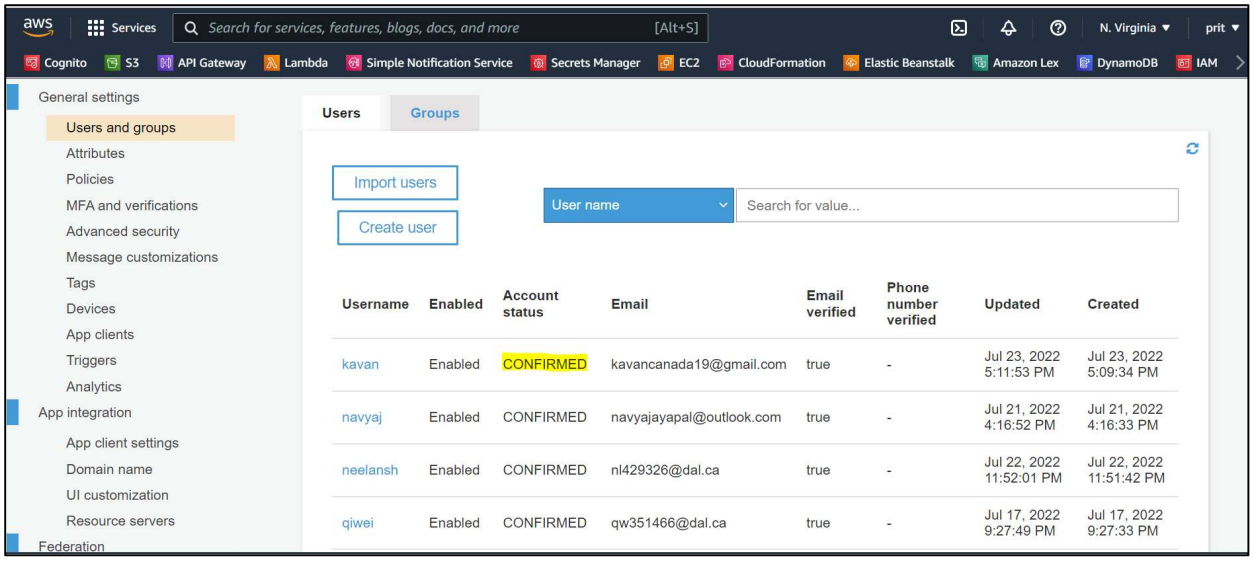


Figure 13 Account Status changed to "confirmed" after verifying the account

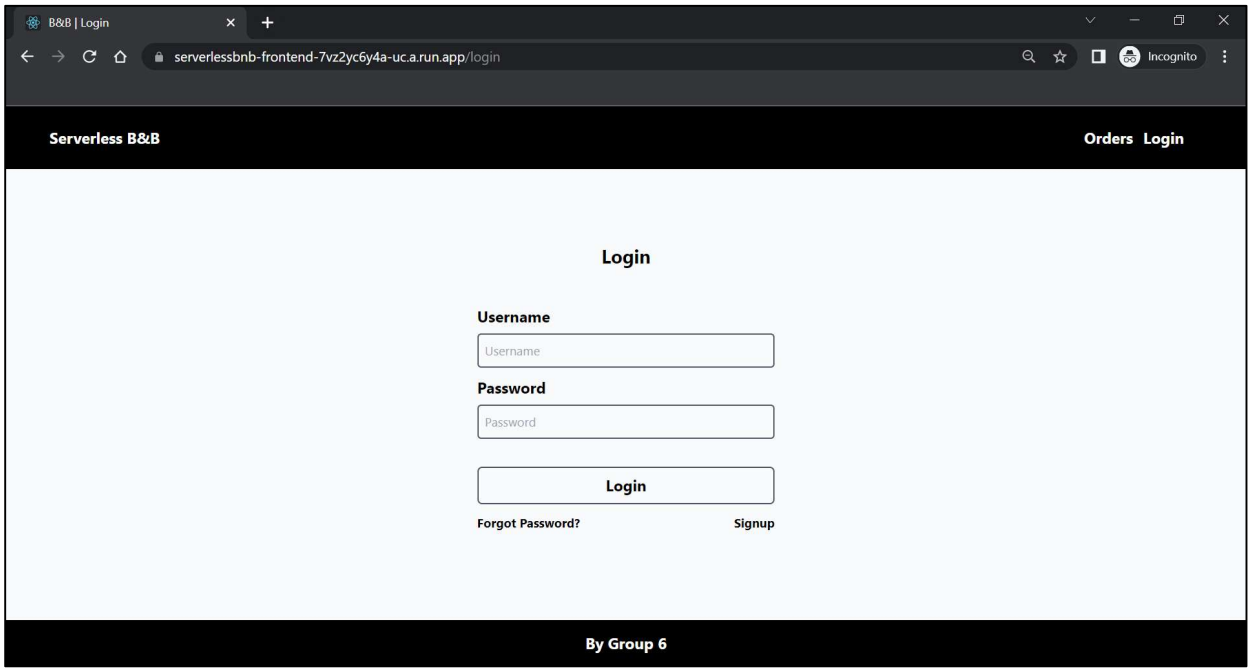


Figure 14 Login Page

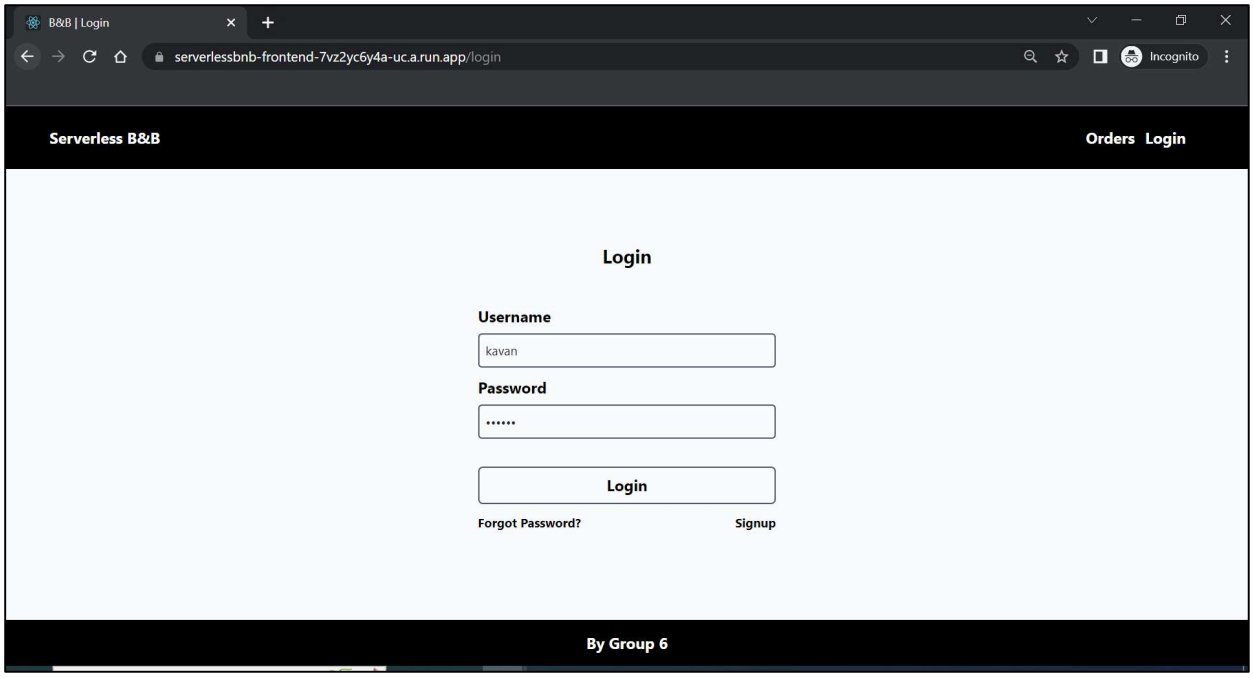


Figure 15 Inputting correct credentials

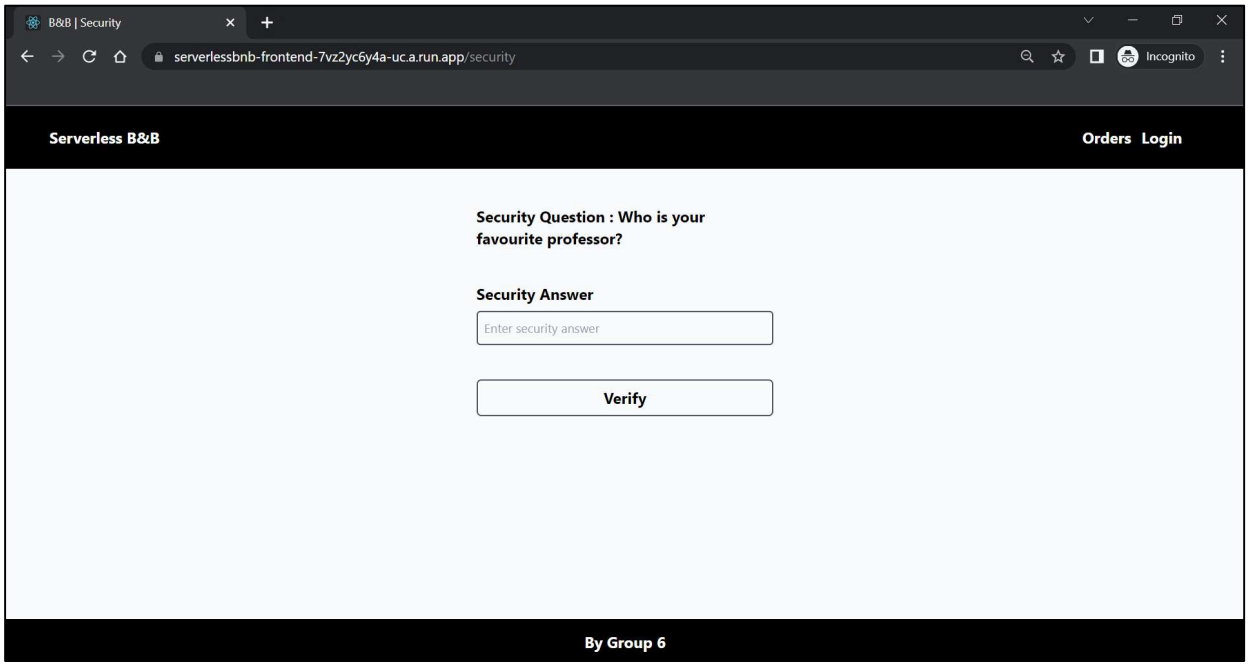


Figure 16 Correct security question for user got fetched from Firestore

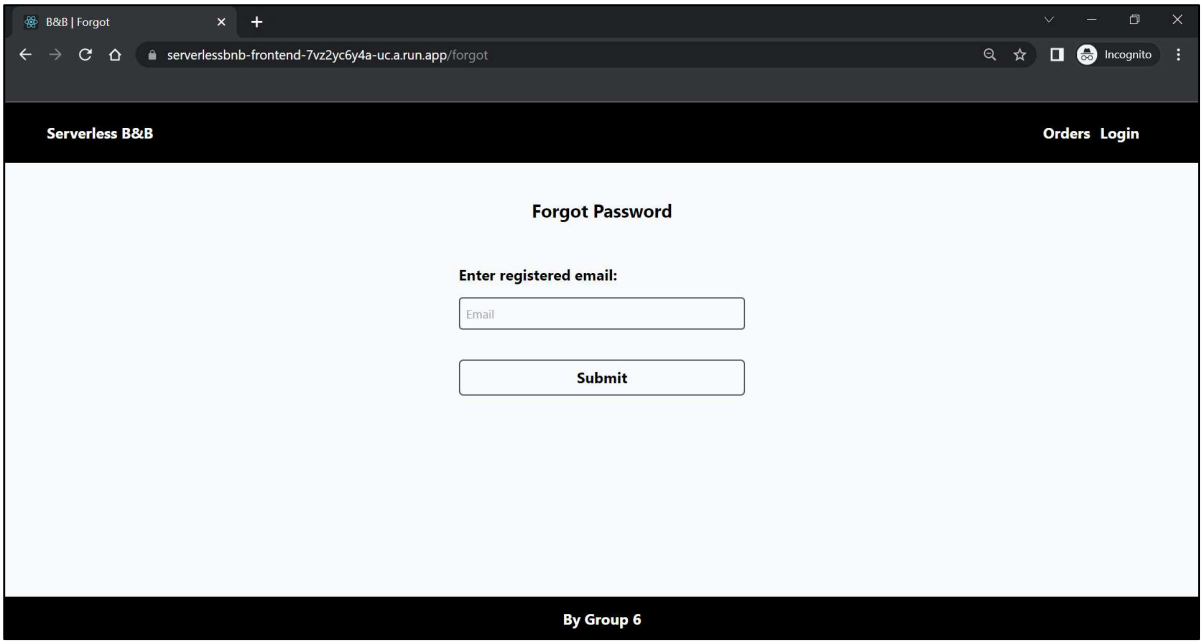


Figure 17 Forgot Password Page

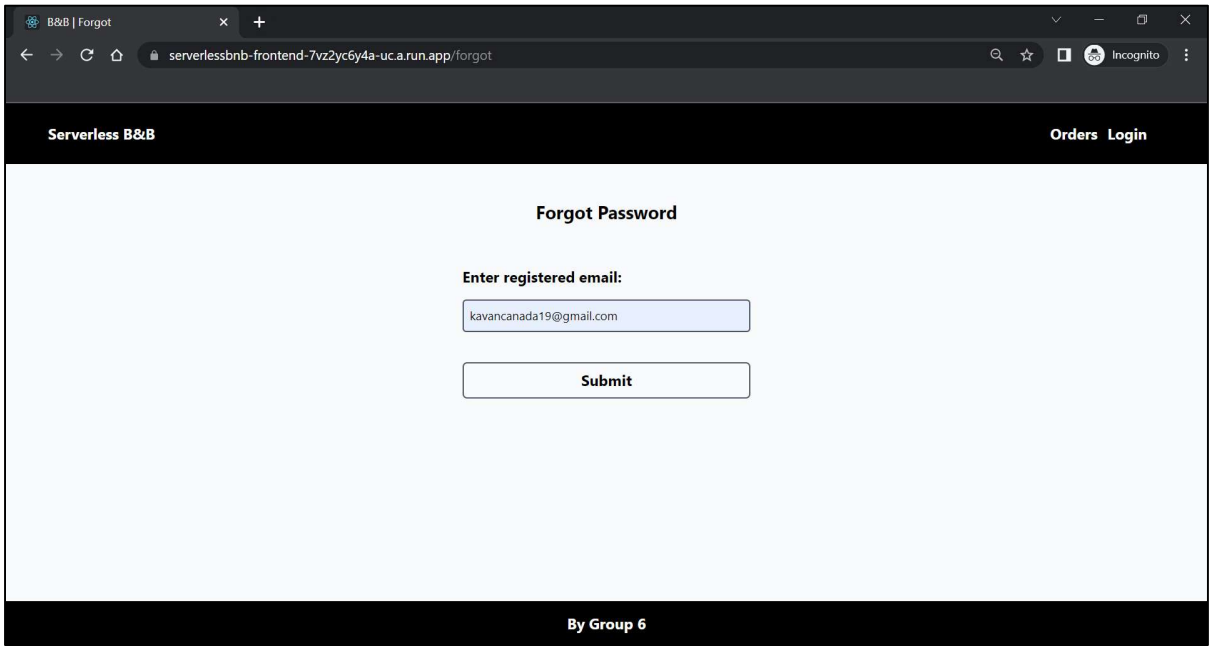


Figure 18 Entering a correct email

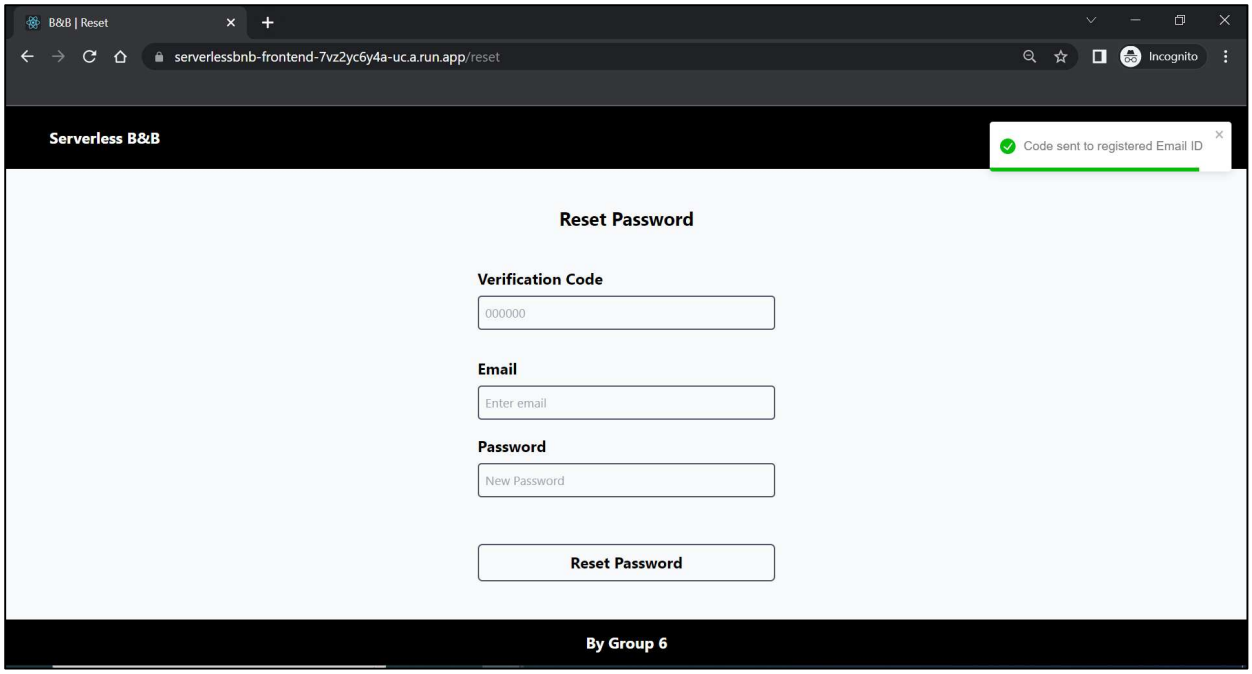


Figure 19 Verification code sent successfully to registered emailID

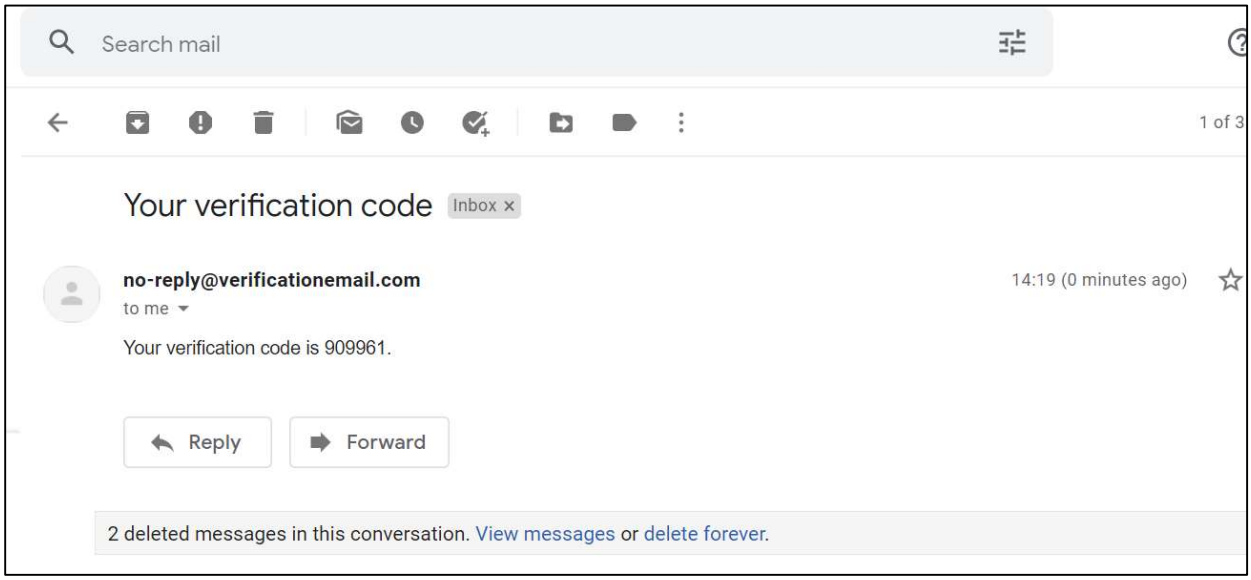


Figure 20 Verification code received in Gmail

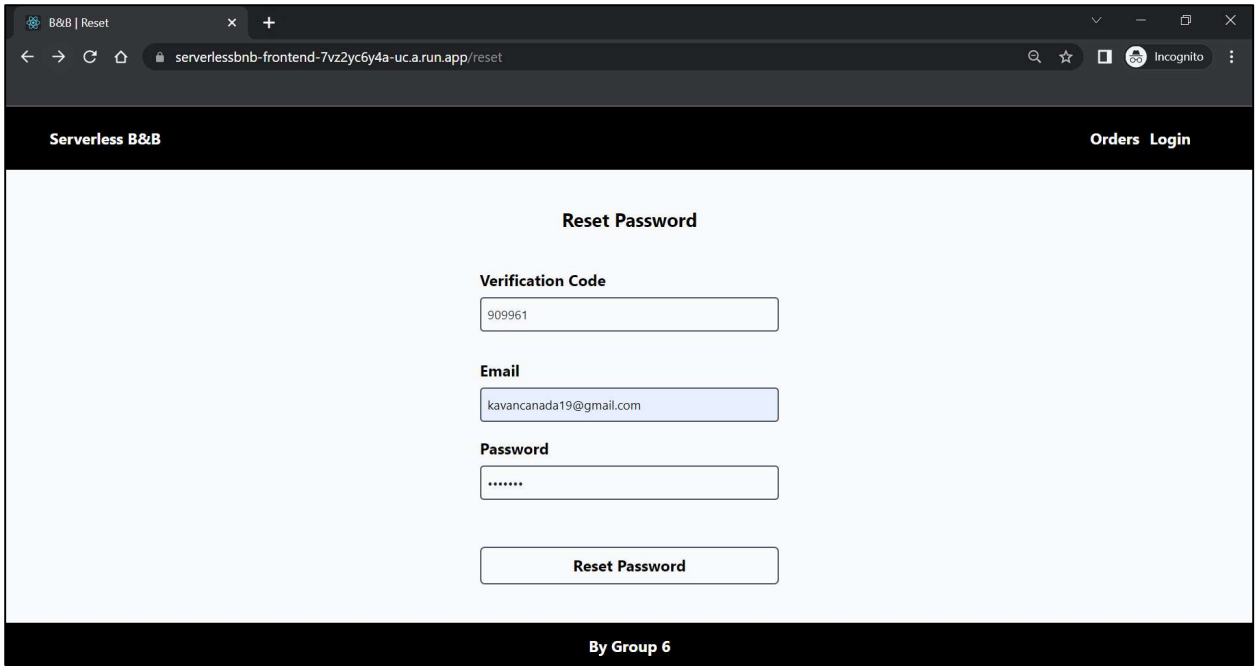


Figure 21 Inputting details to reset the password

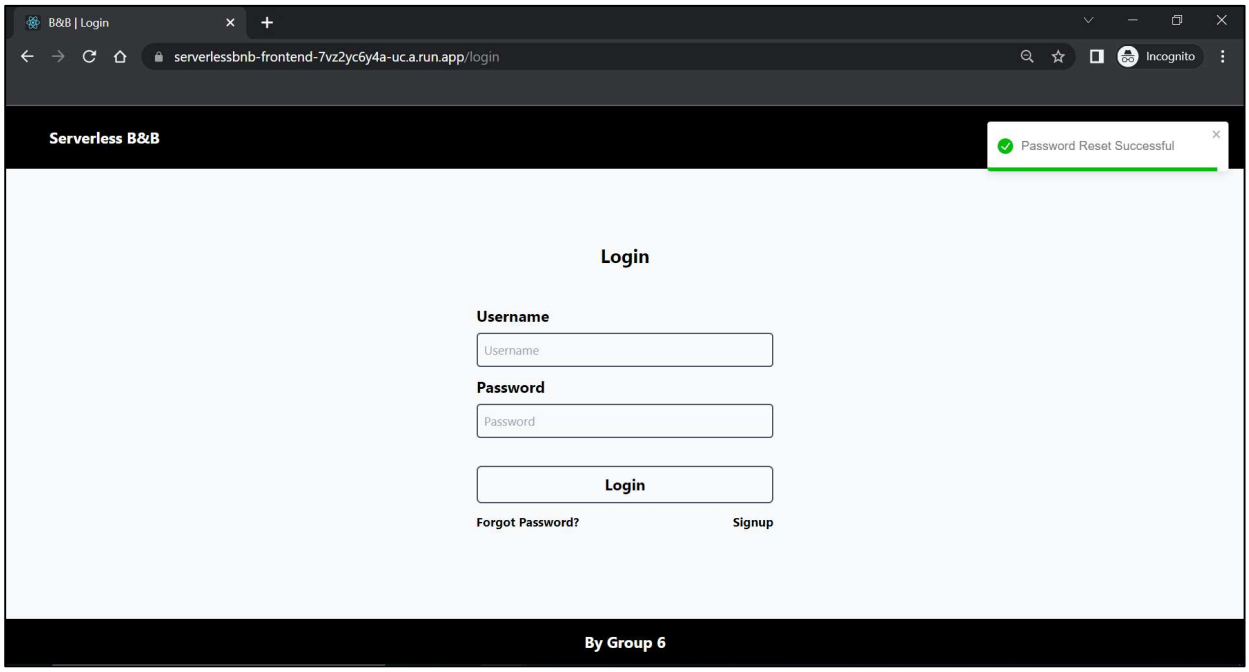


Figure 22 Password reset successfully

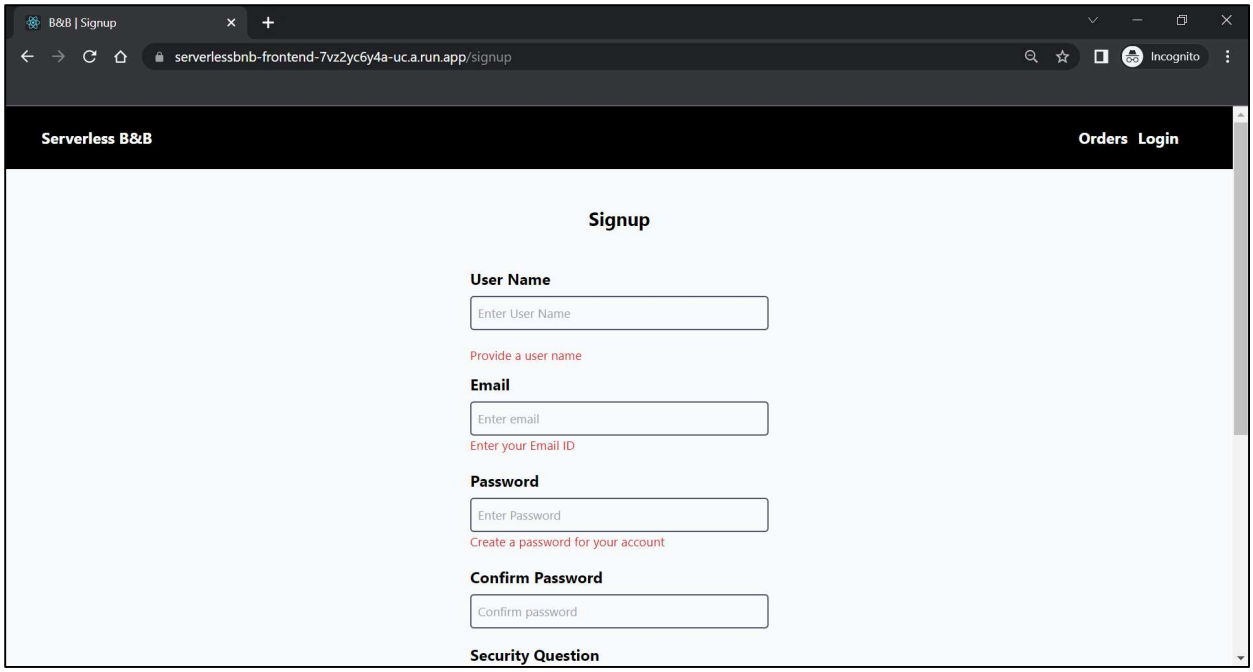


Figure 23 Validations for the Signup page if the user directly clicks on signup button

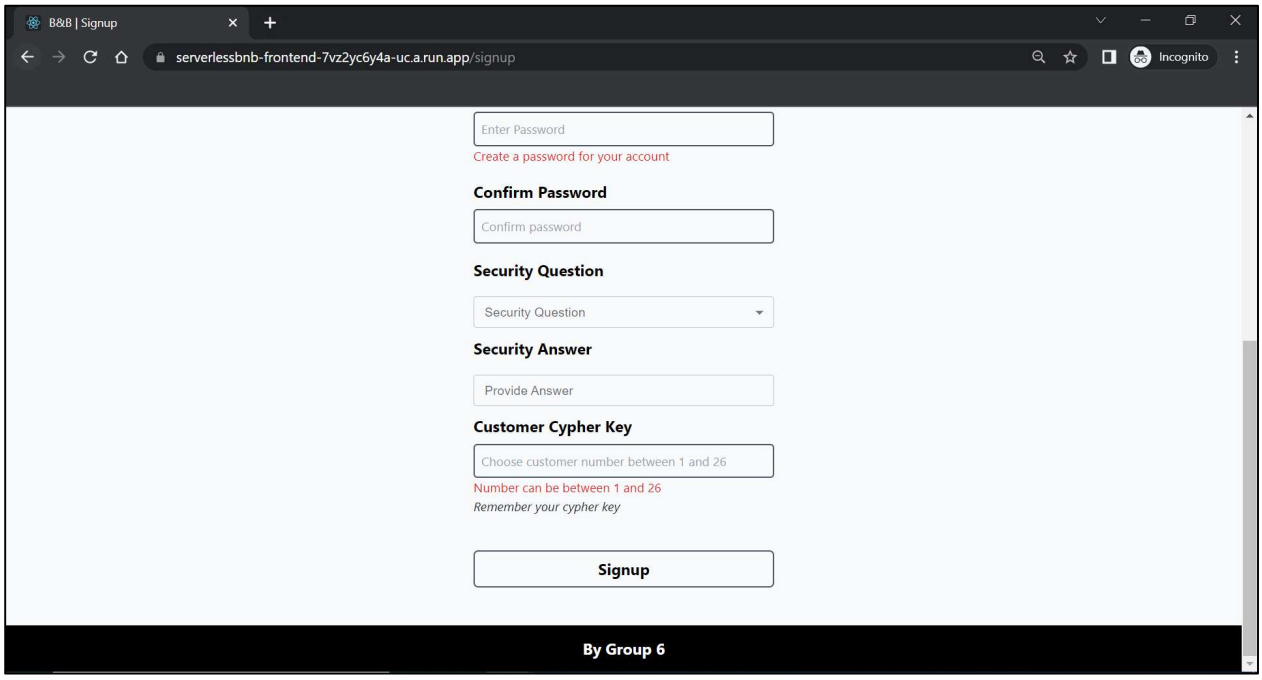


Figure 24 More Validations for the Signup page

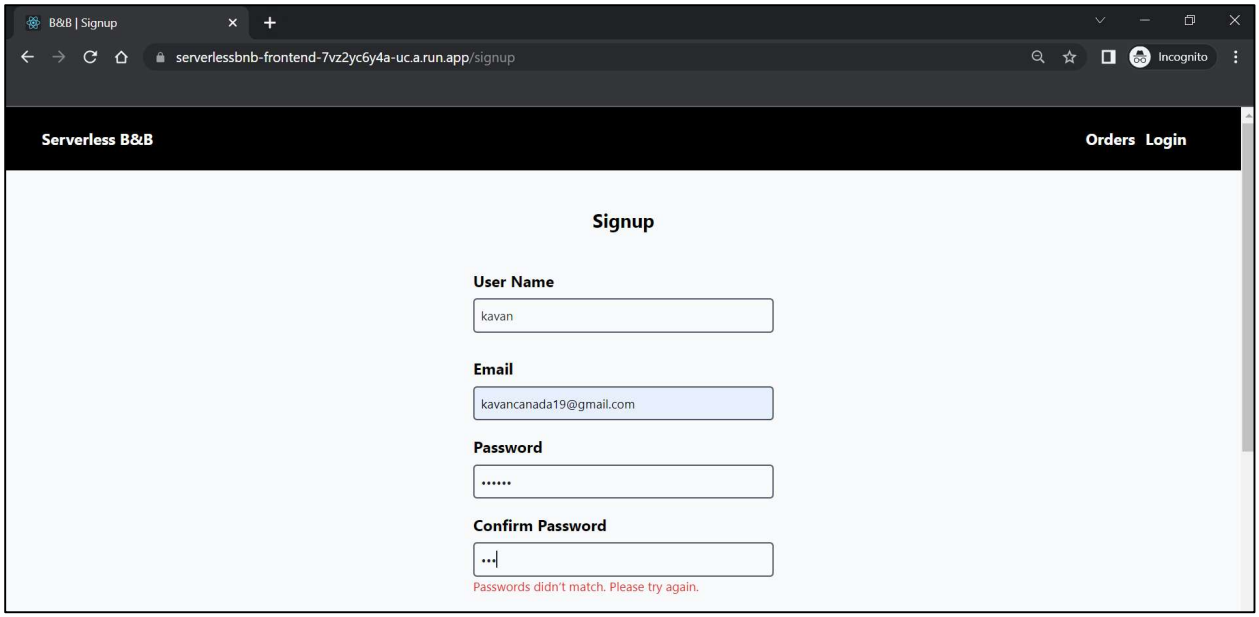


Figure 25 More validations for Signup page

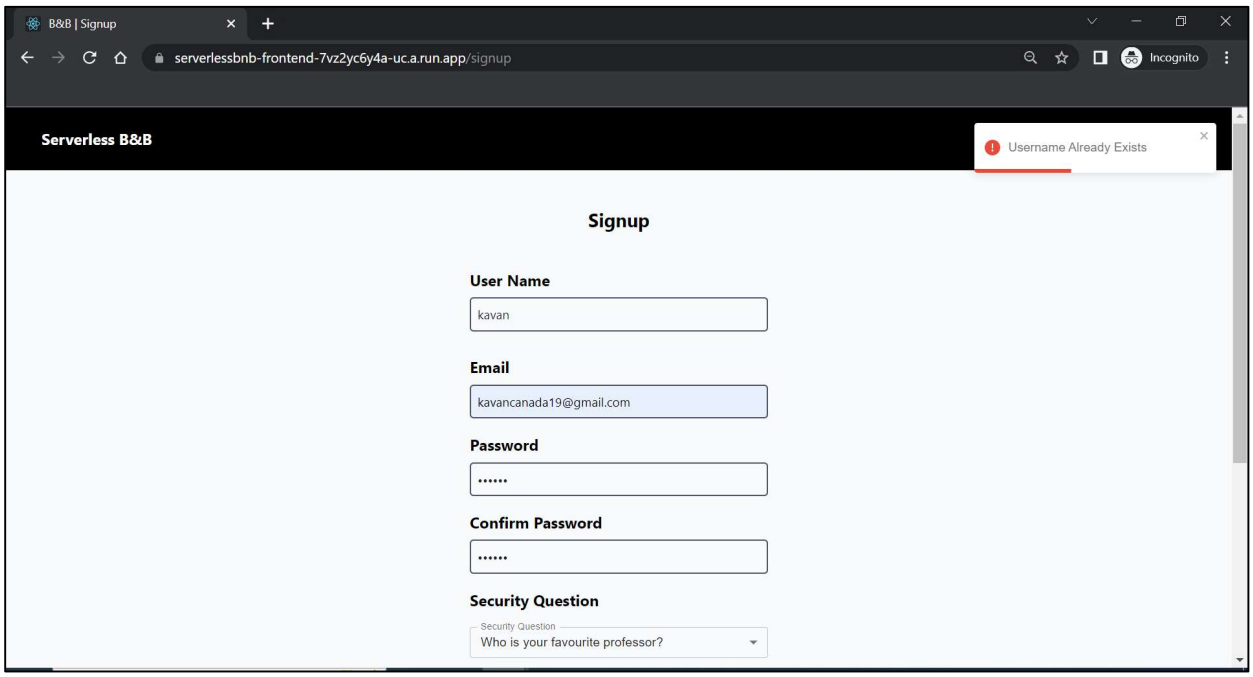


Figure 26 If the user uses same username which is already registered

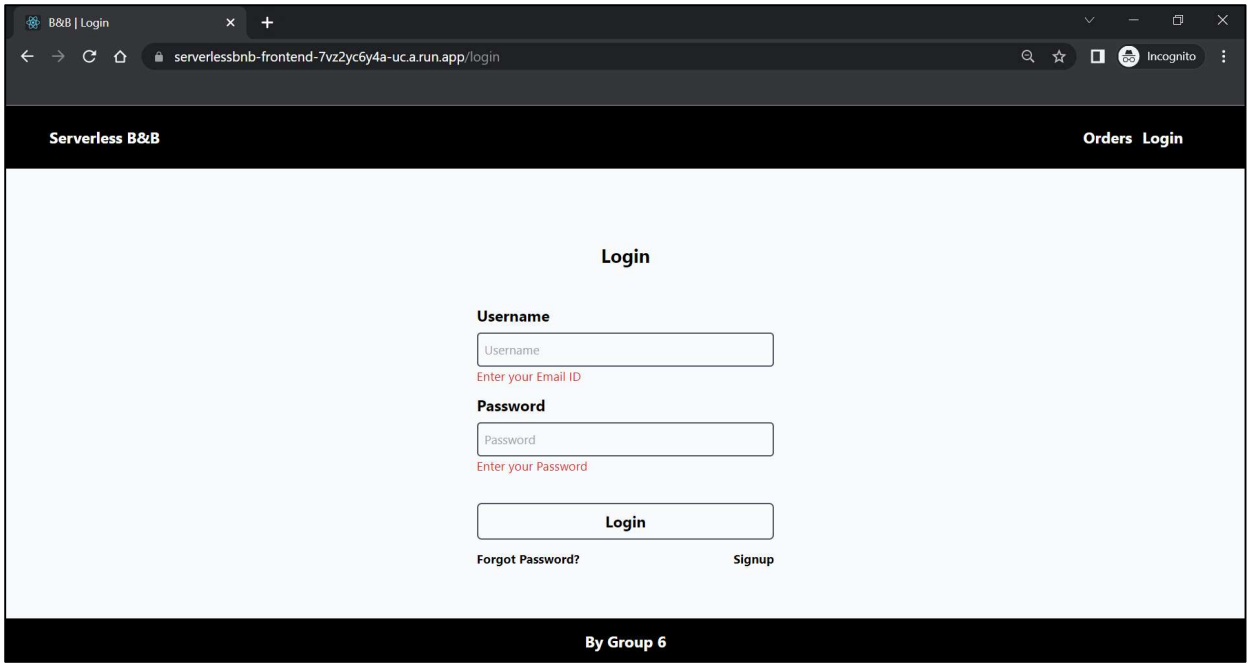


Figure 27 Validations for Login page if the user directly clicks on login

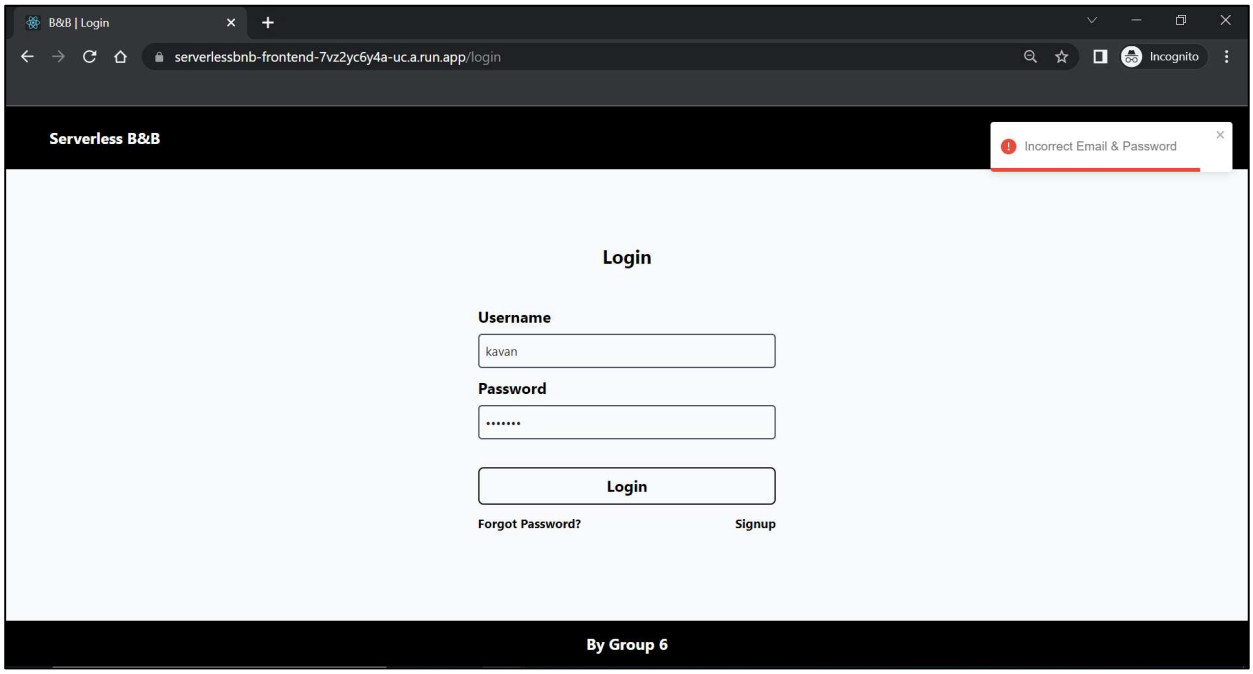


Figure 28 If the user enters an invalid username or password

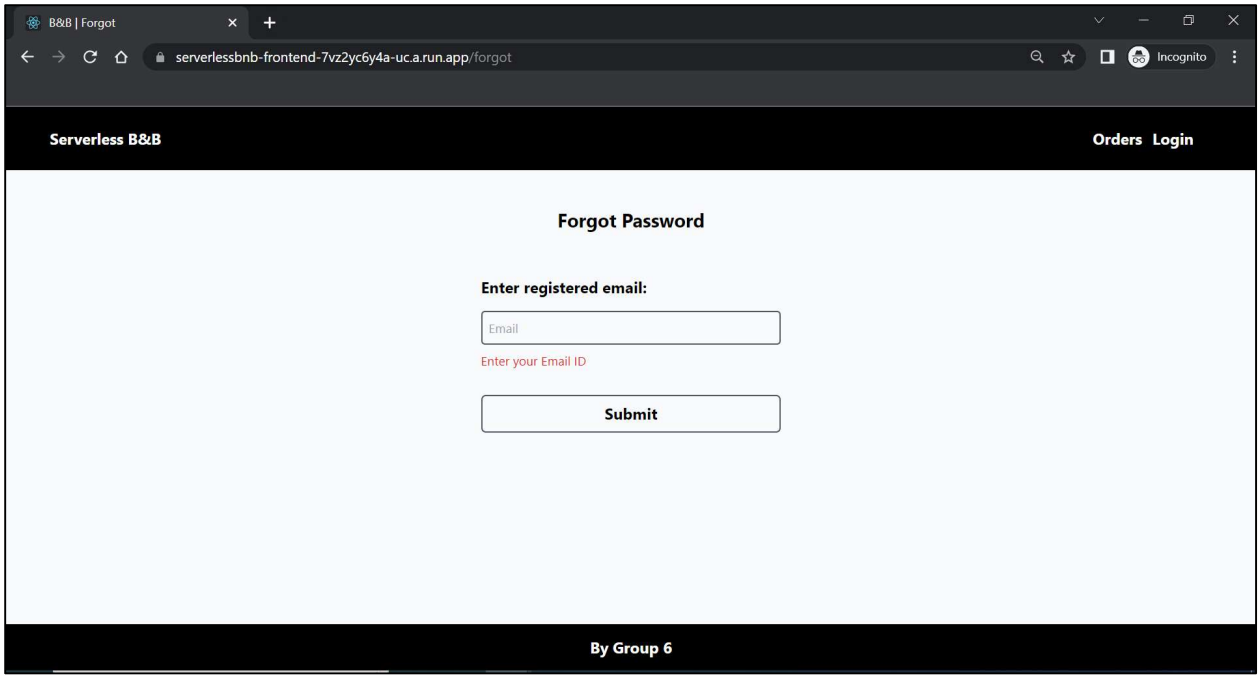


Figure 29 Validations for forgot password page if the user directly clicks on submit

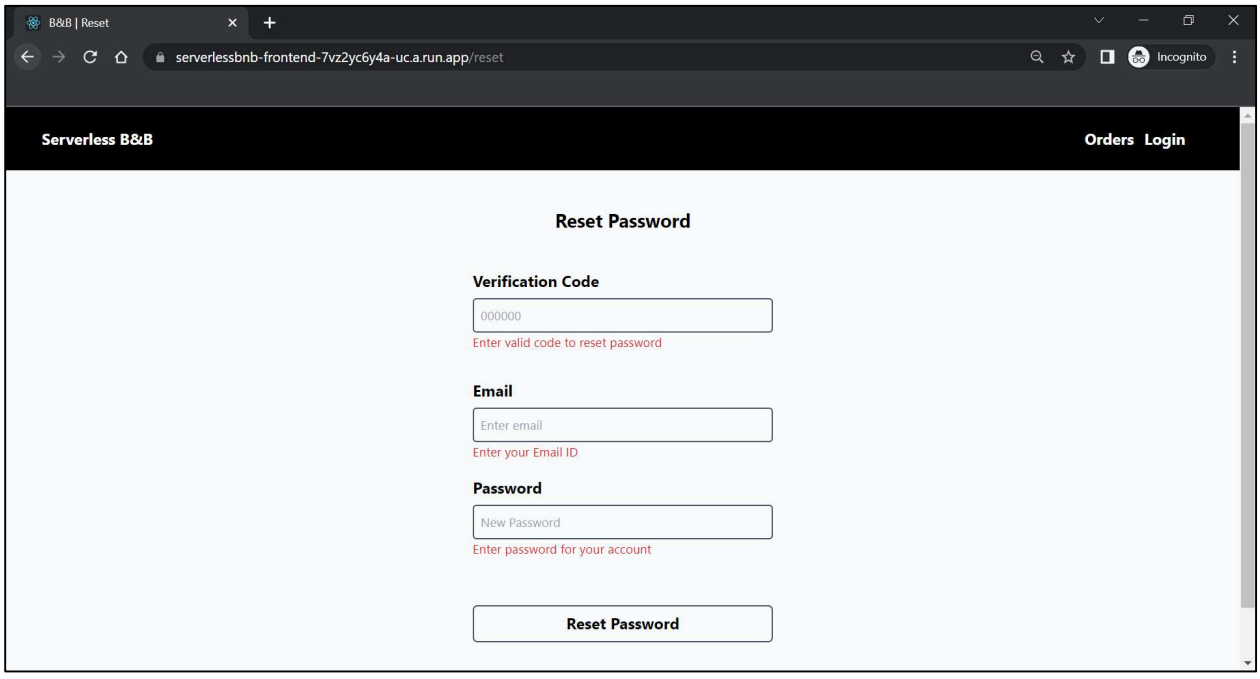


Figure 30 Validations for reset page if the user directly clicks on the reset button

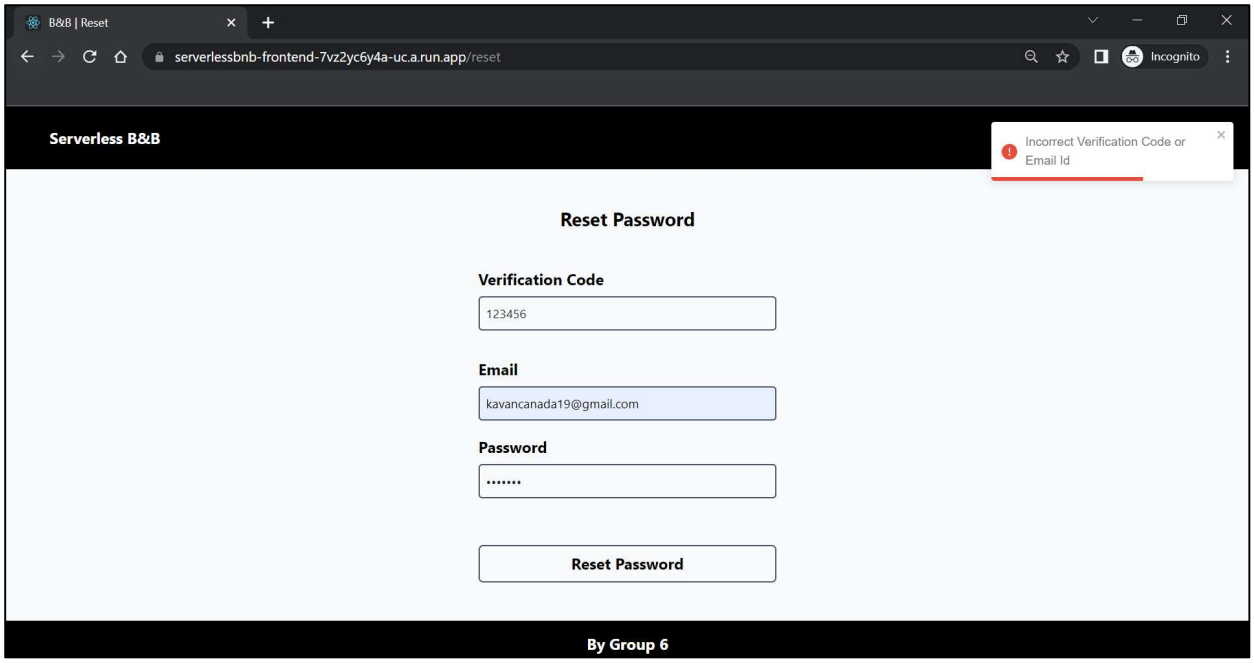


Figure 31 If the user enters an invalid verification code

3.2 Authentication Module

AWS Cognito has been used in this module for user ID and password validation. GCP Firestore and CloudFunction have been configured for a security question and answer authentication, as well as a separate GCP CloudFunction, has been created for implementing Caesar-cipher-based authentication. The Multi-Cloud Model is used in this module. The use of many public cloud providers, known as the multi-cloud approach, is necessary to achieve ultra-high availability.

The main reason we have chosen AWS Cognito and GCP Firestore + CloudFunction for user data authentication and encryption is because AWS Cognito allows us to create customized email templates for account verification [2]. We can manage user signup, login, and other security features such as multi-factor authentication and compromised credentials, whereas GCP Firestore, a NoSQL document database, offers a variety of benefits such as easy scalability, built-in live synchronization, security, and data validation. The GCP Cloud function was chosen for authentication because Cloud Functions is a scalable, pay-as-you-go functions as a service (FaaS) platform that allows you to construct and link event-driven applications using simple, single-purpose code. Cloud Functions uses open-source Function as a service framework that runs multiple functions across multiple environments which eventually avoids lock-in.

Figure 2 also indicates the flowchart for the authentication module. The procedure has been divided into three main parts. The user initially must authenticate using the registered email

address and a password. The user's email address and password will be validated against data in AWS Cognito. If the user's data is correct, the user will be asked a security question, which will be fetched from Google Firestore using Cloud Function. The user's answer will be authenticated with an answer linked to that specific question in Firestore, and if it's valid, the user will be asked to convert the given plain text to encrypt text usi'g the caser cipher key, and this encrypted text, along with the plain text, will be checked at the backend using the Lambda function. If the text is correct, the user will be redirected t' the home page. If the user's information is invalid at any point, the user will be routed to the login page until the user provides proper authentication data.

Pseudo Code:

Step 1: The user first must enter the registered Id and password to authenticate itself.

Step 2: If the Id and password are valid, then Cognito verifies the user and redirected the user security question-answer page, which is 2nd stage of MFA (multi-factor authentication). Otherwise, the system prompts the error message.

Step 3: On the 3rd stage of authentication, the system displays the random plain text and asks the user to enter the valid encrypted text which has been generated using the registered cipher key.

Step 4: If the user enters the invalid cipher text, then the system will prompt the error message and request to provide valid cipher text. While, on the insertion of the valid cipher text, the user will be redirected to the home page of the application and can access all the functionality of the application.

Screenshots and Test Cases:

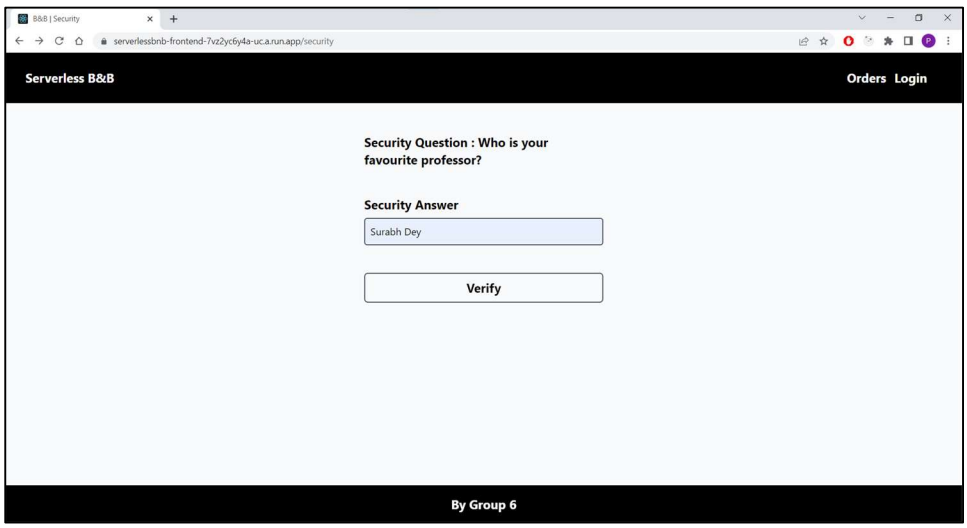


Figure 32 Security question-answer page, 2nd stage of authentication

Figure 32 displays the security QA page of the serverless B&B application. The page contains the user's registered security question and provides the text input box for allowing users to enter the security answer. Now, at the click of the verify button, the verification will be performed by the system as well as backend APIs will be called which are responsible for checking that weather provided answer is right or not. Therefore, once the verification has been done, the user will be moved to the next stage of the authentication process.

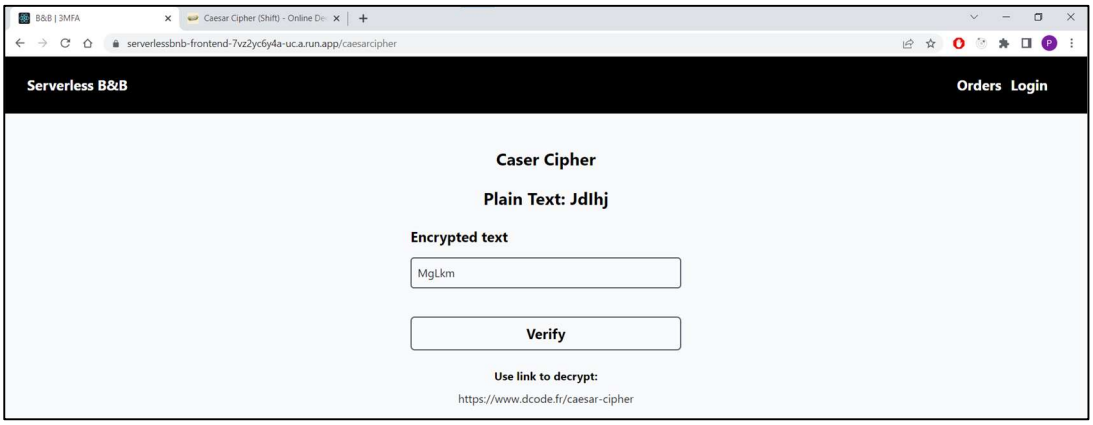


Figure 33 Caeser cipher page, 3rd stage of authentication

Figure 33 displays the security caser cipher page of the serverless B&B application. The page displays the randomly generated plain text and provides the text input box for allowing users to enter the encrypted text. Now, at the click of the verify button, the verification will be performed by the system as well as backend APIs will be called which are responsible for checking that weather provided cipher text is right or not. So, once the verification has been done, the user will be redirected to the home page of the application. Where use finds the multiple options that have been provided by the application.

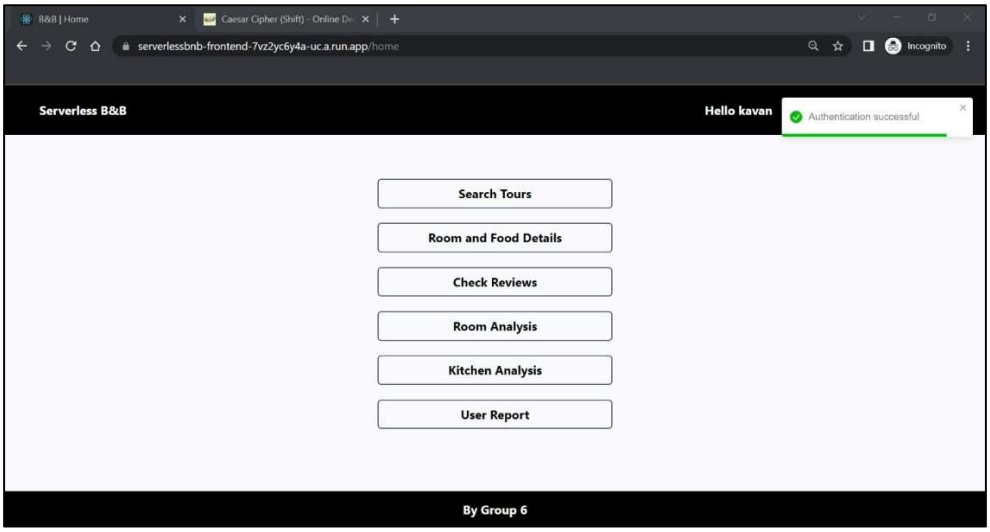


Figure 34 Home Page of the serverless B&B application

Test case 1: Try to access the application without entering the security answer

On the security QA page, the user has to enter a valid security answer. So, if the user tries to access the application without it or clicks on the verify button without entering any answer then system will display the error message indicating “Enter answer” to the user.

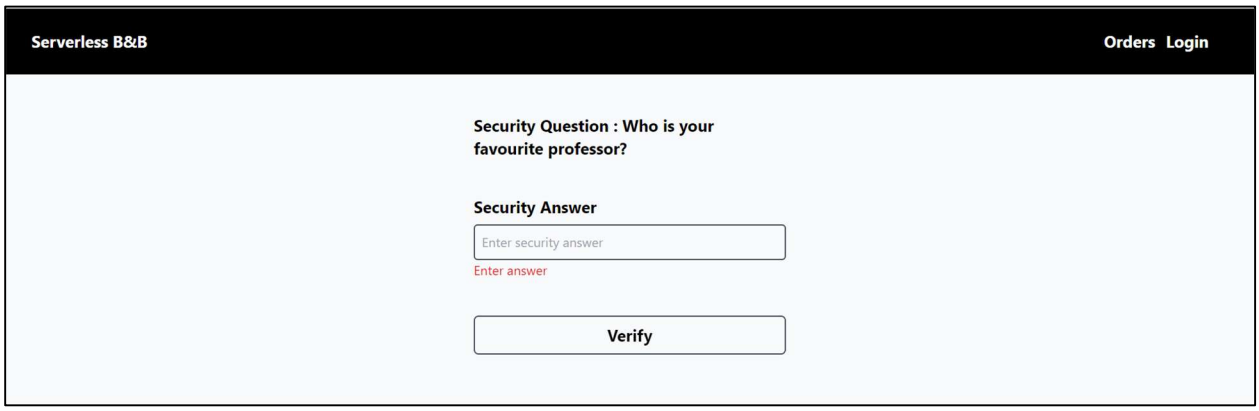


Figure 35 Trying without entering an answer, signup page prompts an error message

Test case 2: Try to access the application using an invalid security answer

If the user enters the invalid security answer, then the system will prompt the user with the error message “invalid security answer” and request to provide a valid answer to verify itself.

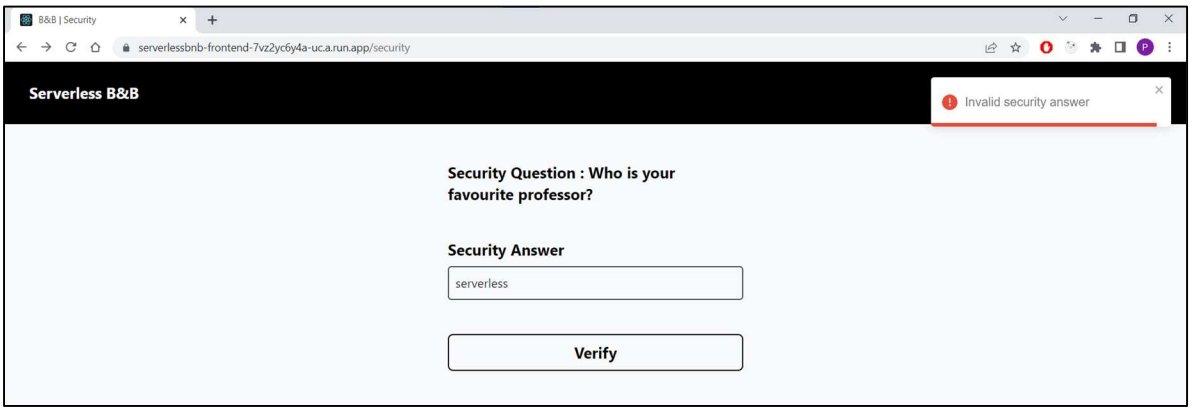


Figure 36 Trying with the invalid answer, the signup page prompts an error message

Test case 3: Try to access the application without entering the cipher text

On the caser cipher page, the user has to enter the valid encrypted text that has been generated using the caser cipher key. So, if the user tries to access the application without it or clicks on the verify button without entering any answer then the system will display the error message indicating “Provide a valid encryption text” to the user.

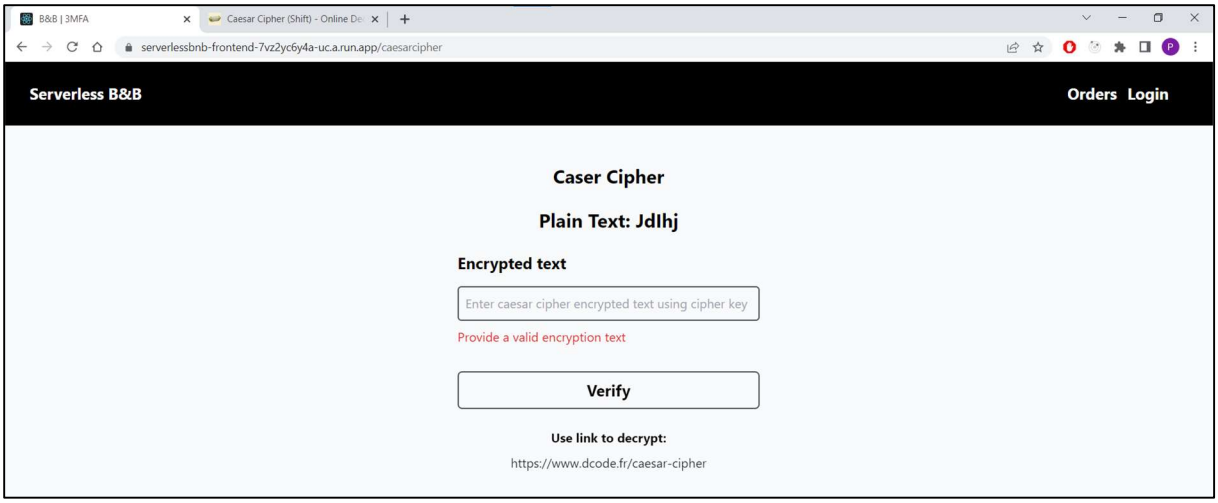


Figure 37 Trying without entering cipher text, the caesar cipher page prompts an error message

Test case 4: try to access the application using invalid encryption text

If the user enters the invalid cipher text, then the system will prompt the user with the error message "invalid encrypted text! Please try again" and request to provide valid text to verify itself.

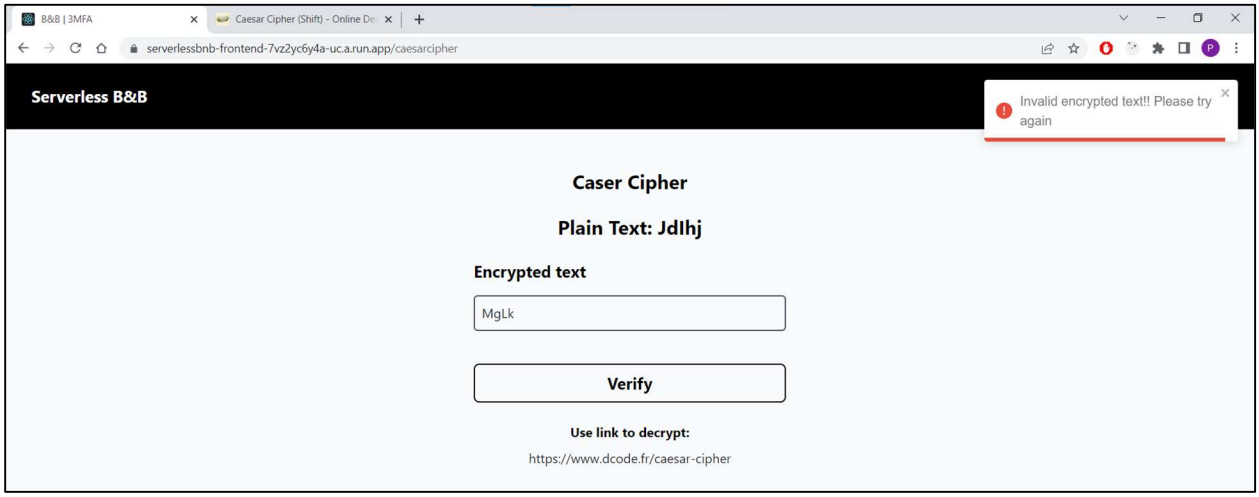


Figure 38 Trying with invalid cipher text, the caesar cipher page prompts an error message

3.3 Online Support Module

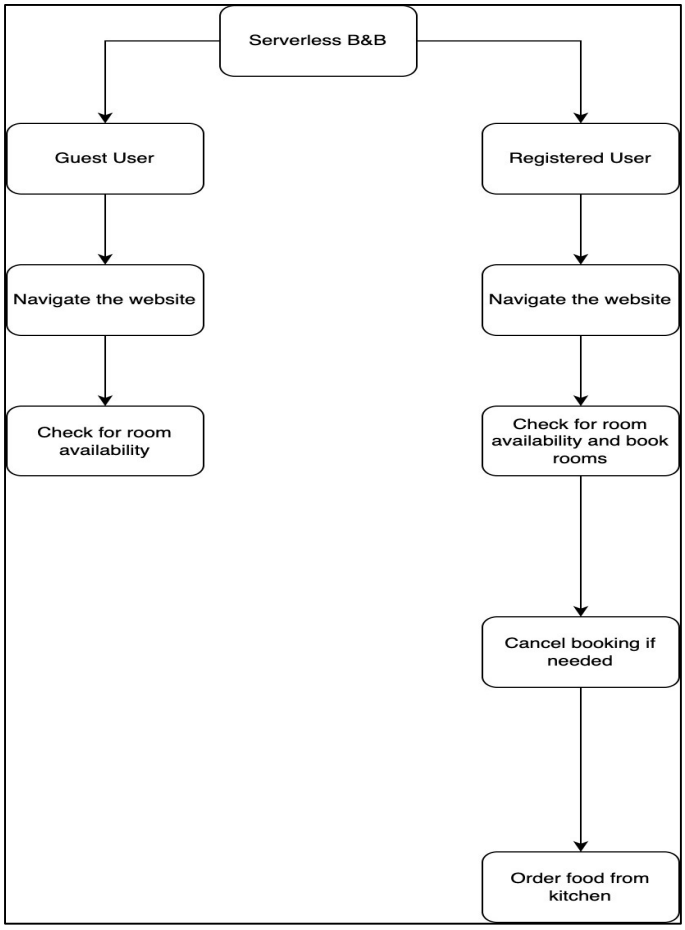


Figure 39 Functionalities for different user having access to online support

Online Support module would be handling requests of registered and guest user, each of which would be provided with different functionality. We will be integrating Amazon Lex with ReactJS and provide users with the above functionalities (room, kitchen, and navigation functionalities).

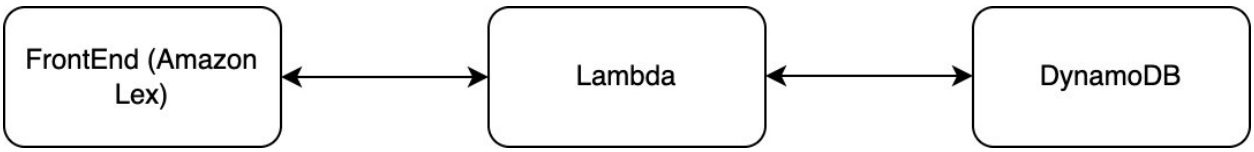


Figure 40 Flow of online support module

Users will communicate with Amazon Lex which will trigger lambda functions and perform CRUD operation on DynamoDB and store details about room booking and kitchen orders.

Pseudo Code:

Step 1: The user first has to enter the registered Id and password to authenticate itself.

Step 2: If the user is not valid user, then user can navigate through the website only.

Step 3: If the user is valid user, then user can order food from specified menu and quantity limited from 1-10 by validating through Lambda function and on fulfilment entering data in DynamoDB.

Step 4: If the user is valid user, then user can book room from specified types by validating through Lambda function and on fulfilment entering data in DynamoDB.

Step 5: If the user is valid user, then user can cancel book room if the room number specified by user belongs to the user by validating through Lambda function and on fulfilment entering data in DynamoDB.

Step 6: If the user is valid user, then user can navigate through the website as well.

Screenshots and Test Cases:

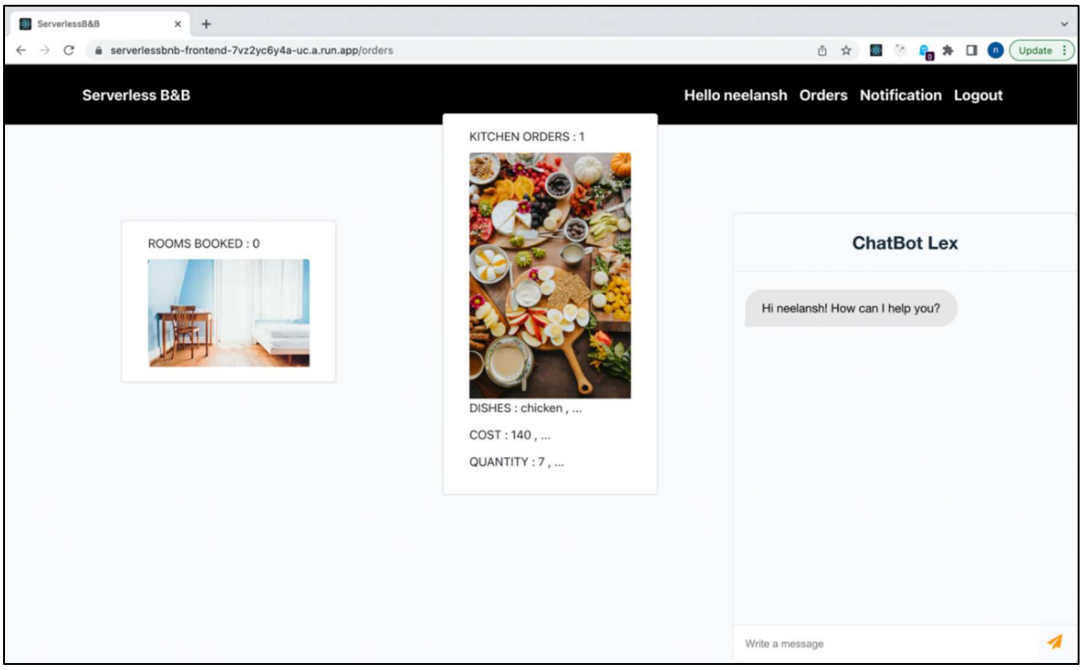


Figure 41 Logged in user using Chatbot

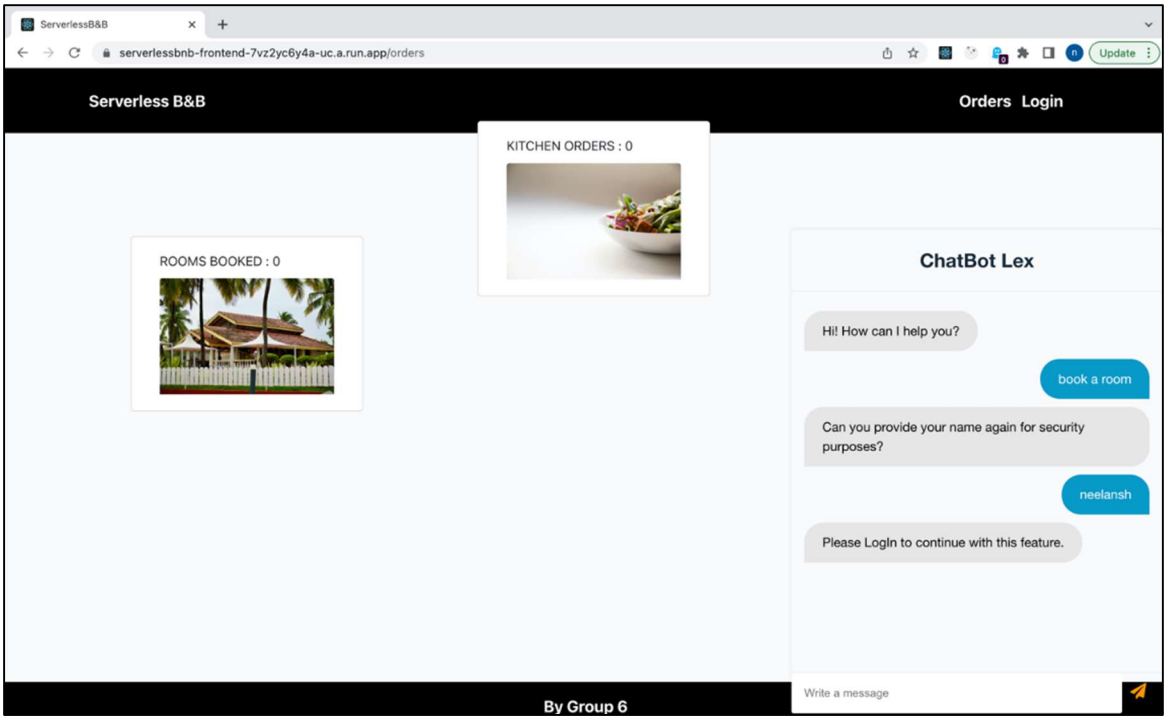


Figure 42 Guest user using Chatbot

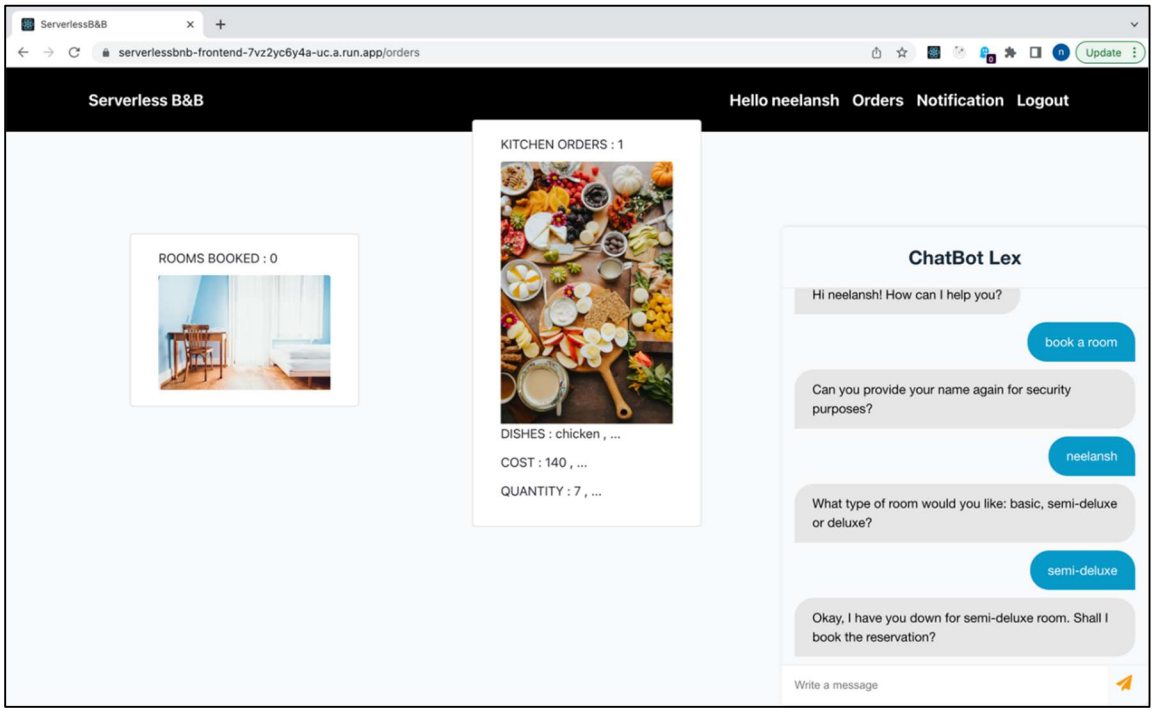


Figure 43 LoggedIn user booking a room

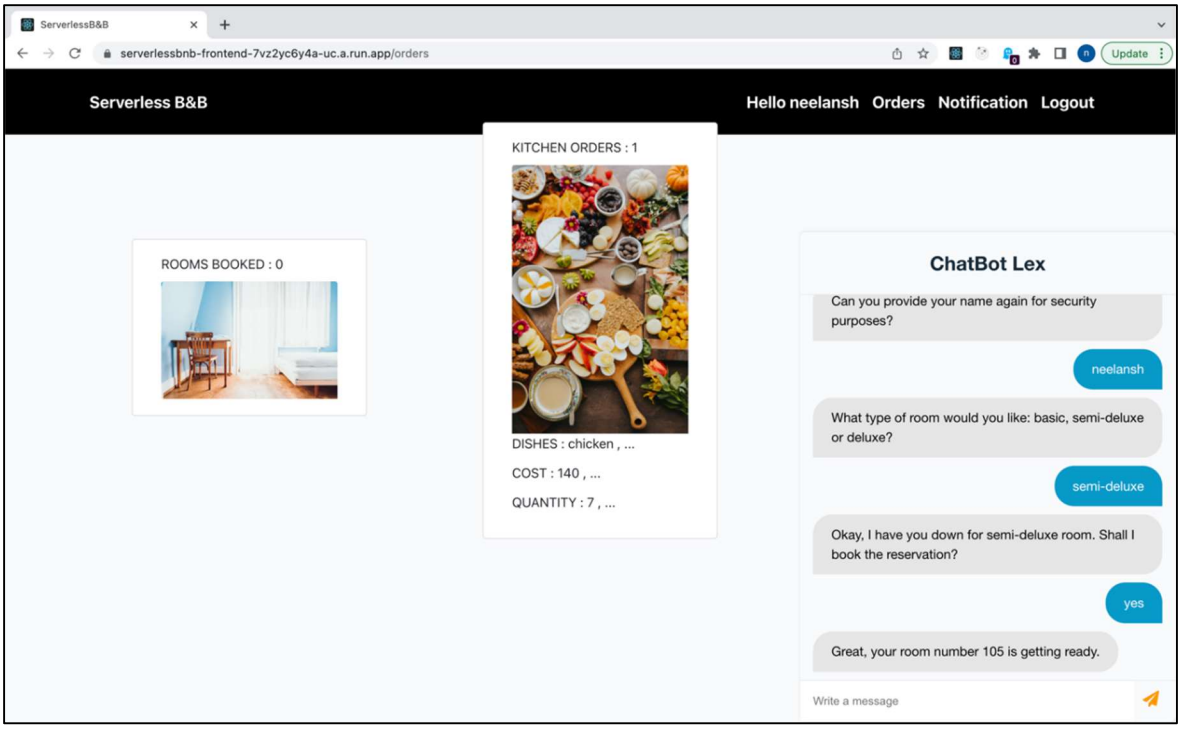


Figure 44 : LoggedIn user successful booking of room

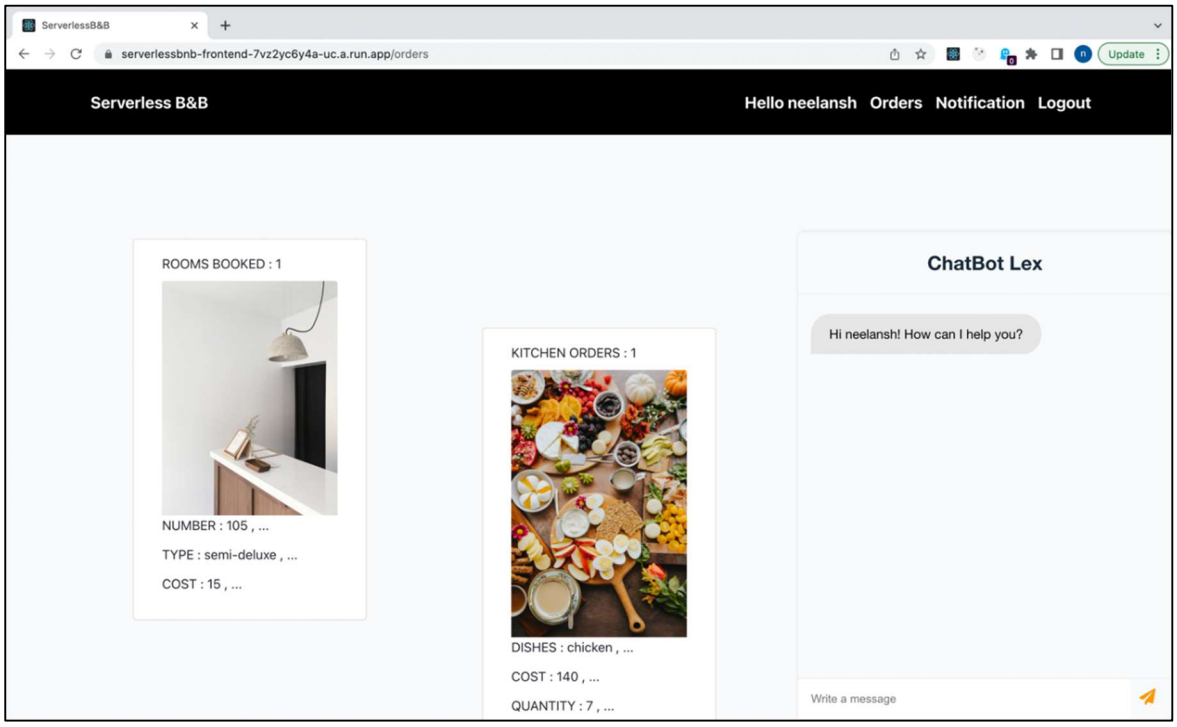


Figure 45 Order Page on successful booking of room

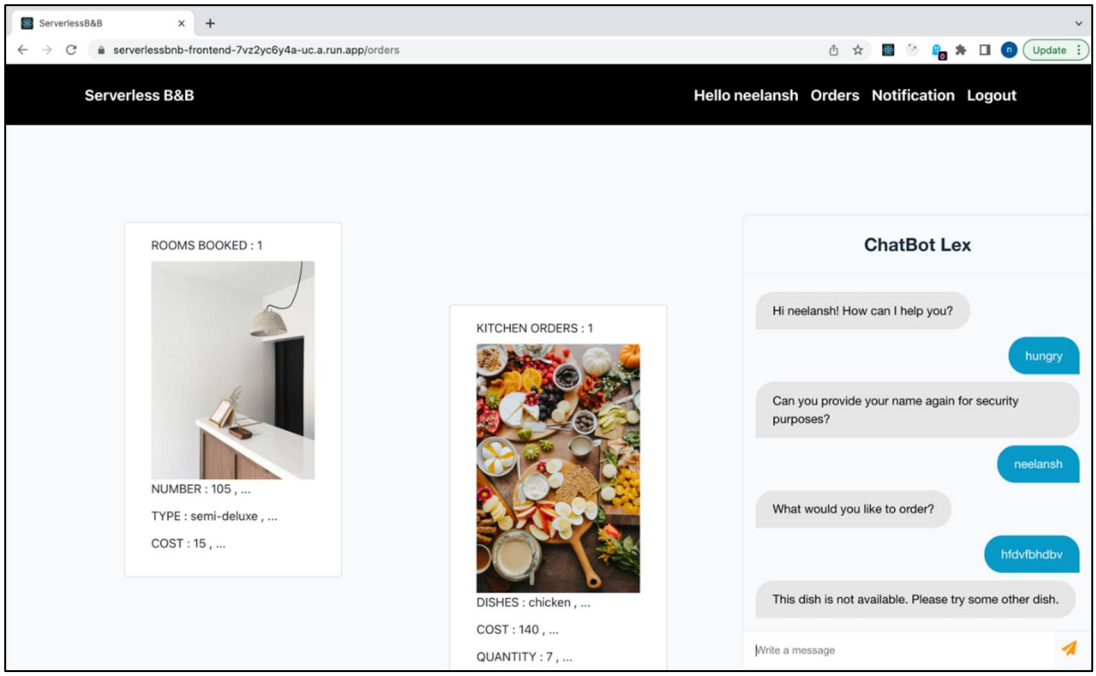


Figure 46 LoggedIn user ordering food from kitchen

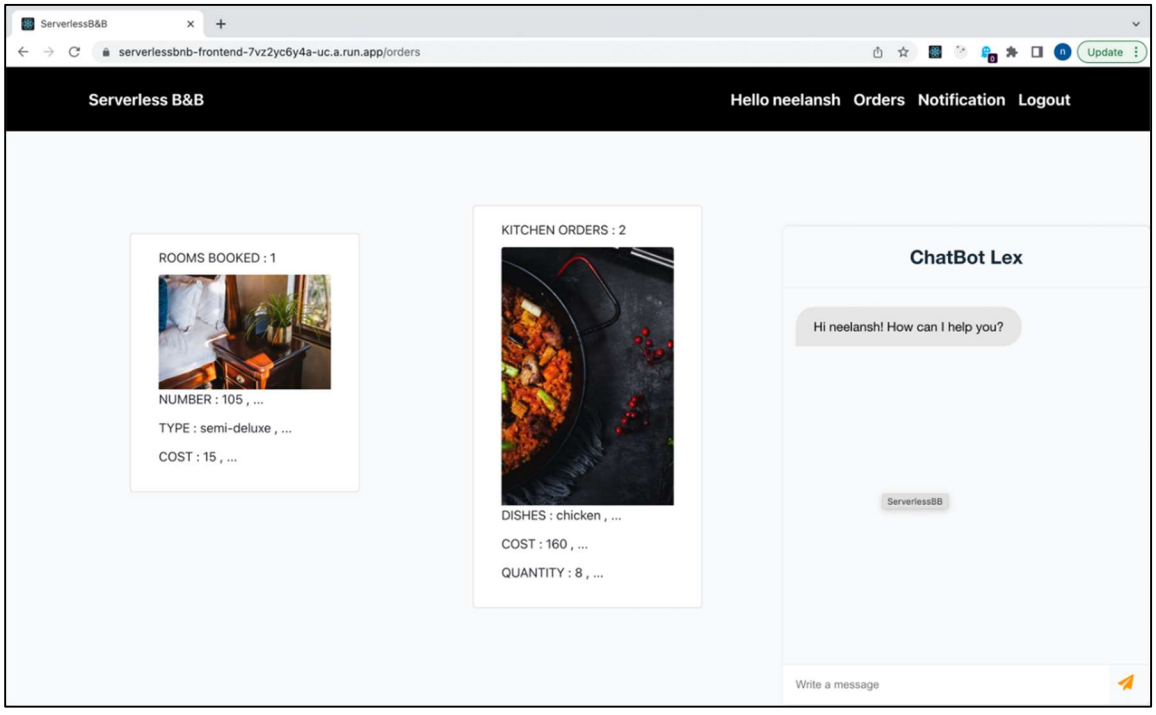


Figure 47 LoggedIn user on successfully ordering food

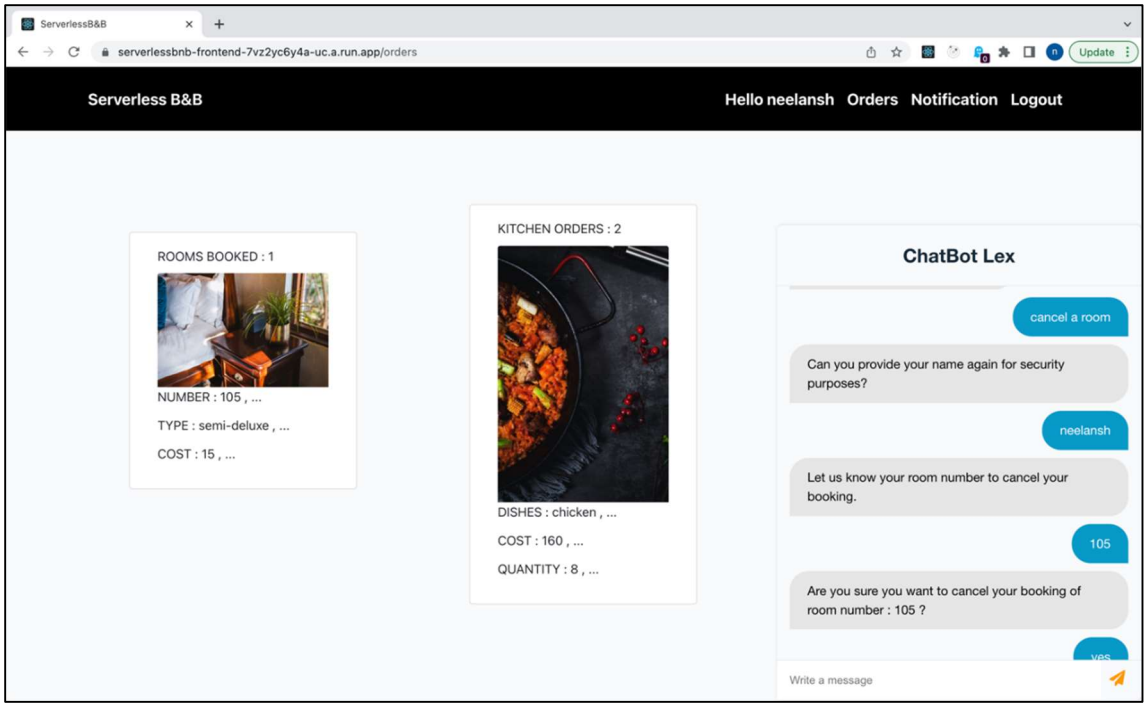


Figure 48 LoggedIn user cancelling room booking

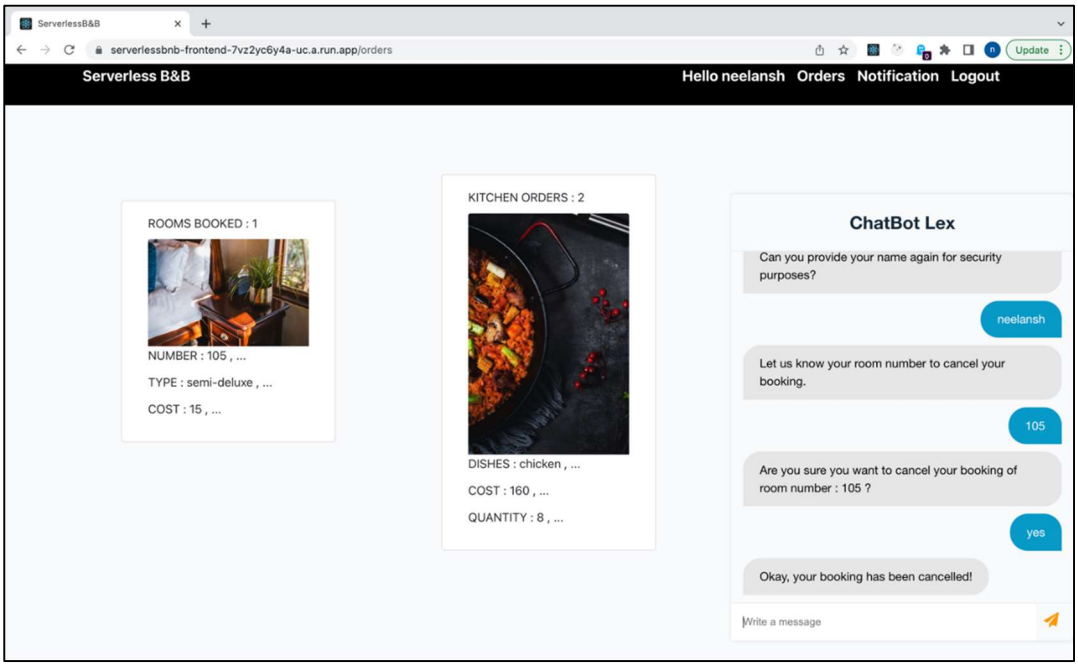


Figure 49 LoggedIn user on successfully cancelling room booking

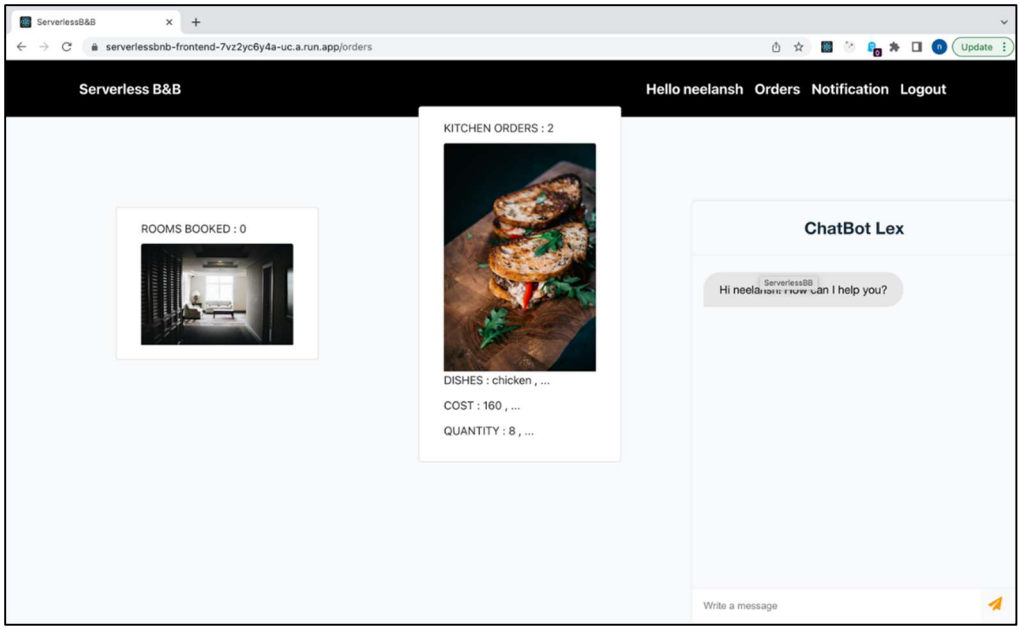


Figure 50 Order Page on successful cancellation of room

3.4 Tour Management

Both guests and registered users will be able to obtain information about the tour package. They will inquire about tour package specifics in Lex or through User Interface. As soon as Lex receives a query from a user, it will invoke a lambda function, which will request the tour data from GCP BigQuery ML. The request received by BigQuery ML is then processed, and the classification result produced through analysis is returned to the lambda function. When lambda receives the analytical data from BigQuery ML, it sends the response to Lex, who then displays it to the user.

The app will make use of Google Cloud BigQuery ML, which takes data from a DynamoDB table and automatically creates a prediction or analysis report by training, testing, and deploying a machine learning model. The tour service will provide the input for the BigQuery ML, and the data will be processed and turned into a CSV table that will be consumed by the BigQuery ML service. There are two forms of analysis that are carried out. The initial analysis is to determine the place client wants to go and second study the duration of the stay. After getting a place and the number of days from the user, the model will predict and determine the tour package based on customer input and provide a suitable list of tours. The user can then book any tour by clicking on the book button and a confirmation email for the tour will be sent to the user on the registered email through the SNS service. For enabling mail confirmation, the user needs to confirm the mail confirmation request sent during signup. Once confirmed user will start receiving confirmation emails from the SNS.

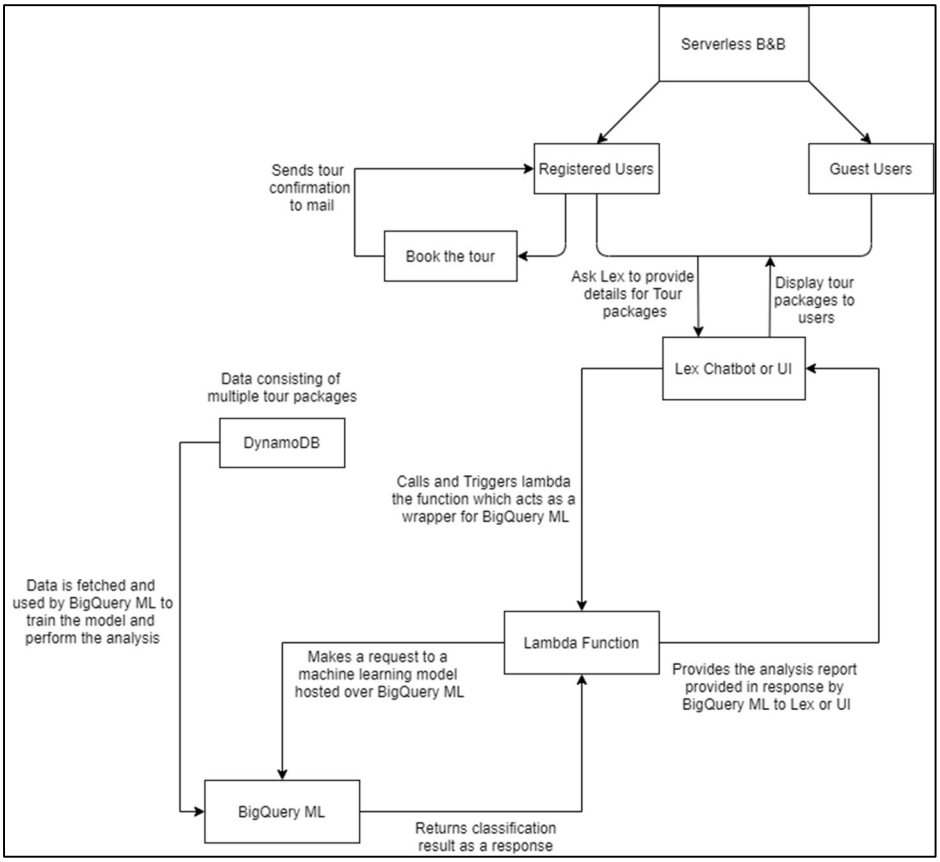


Figure 51 Flowchart of Tour Management Module

Pseudo Code:

Step 1: Create a DynamoDB table with a list of different tour packages.

Step 2: A GCP cloud function will retrieve the data from the DynamoDB database, generate a CSV file, and save it in GCP Cloud Storage.

Step 3: In GCP BigQuery, create a dataset with a table.

Step 4: This table will make use of the CSV file that was prepared to train the model.

Step 5: Once data is trained, the model is ready to provide statistical analysis.

Step 6: When a user inputs a location and number of days and hits the submit button, a cloud function is triggered that retrieves statistical analysis provided by the model based on the user's input and shares it with the user. Whereas Lex will first call a lambda function, which will initiate the cloud function responsible for obtaining and returning the classification result as a response.

Step 7: If the user has previously subscribed to the SNS service during signup, a confirmation email will be sent to the registered email id after the user clicks on a book tour.

Screenshots and Test Cases:

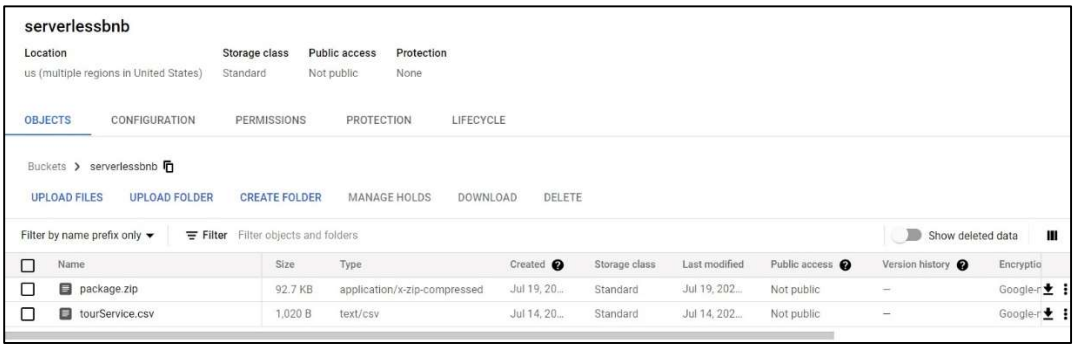


Figure 52 tourService.csv created using Cloud function by using the DynamoDB table

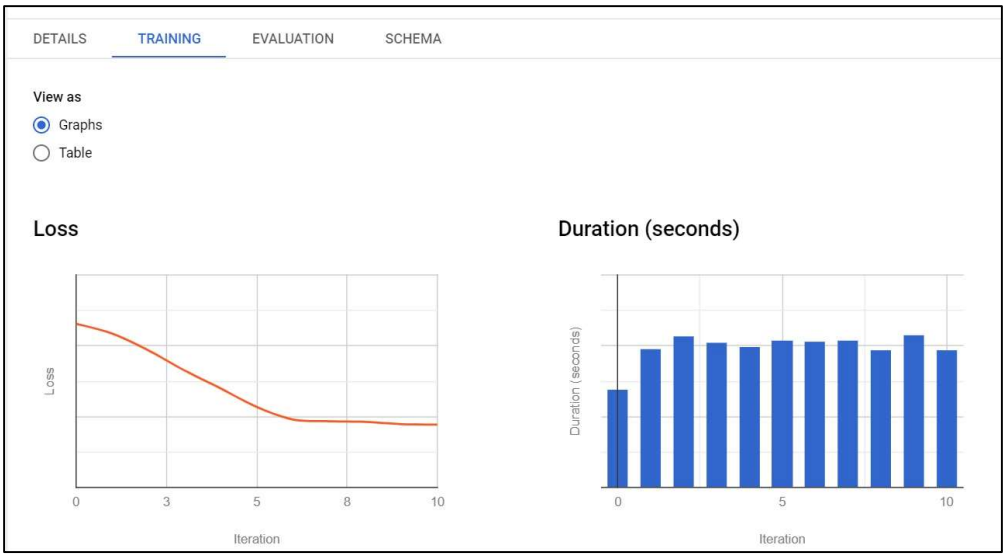
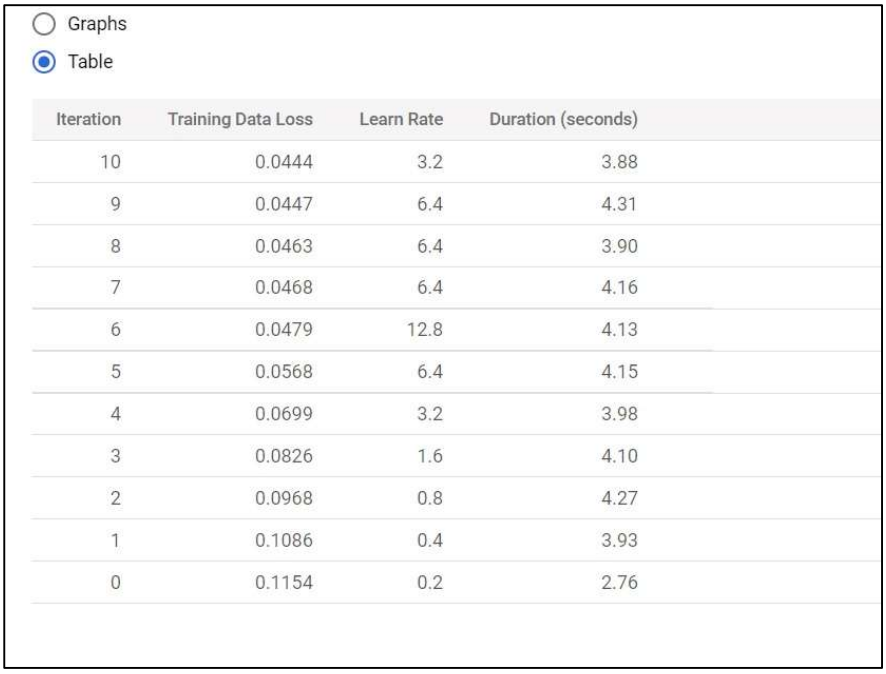


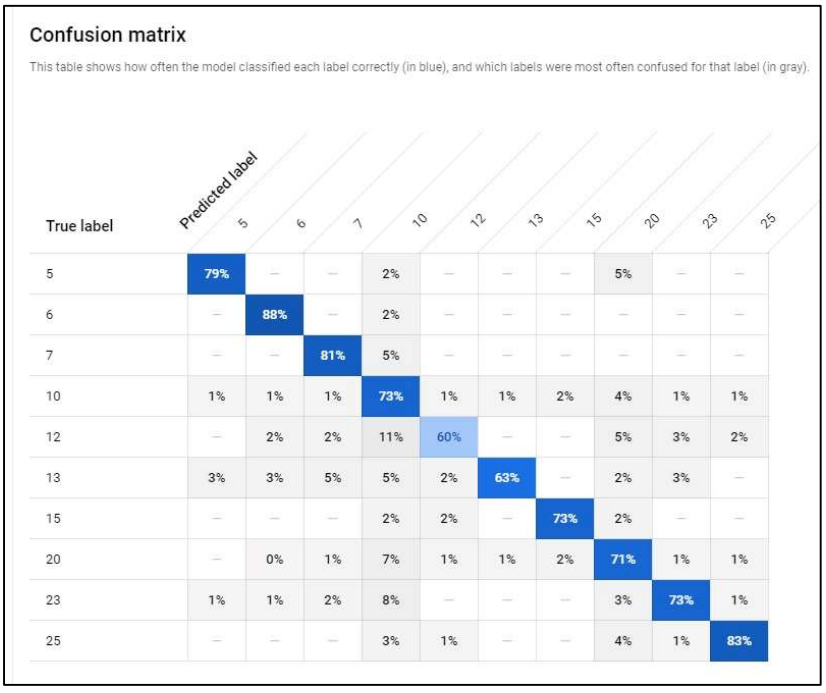
Figure 53 Graph view of trained model



Graphs
Table

Iteration	Training Data Loss	Learn Rate	Duration (seconds)
10	0.0444	3.2	3.88
9	0.0447	6.4	4.31
8	0.0463	6.4	3.90
7	0.0468	6.4	4.16
6	0.0479	12.8	4.13
5	0.0568	6.4	4.15
4	0.0699	3.2	3.98
3	0.0826	1.6	4.10
2	0.0968	0.8	4.27
1	0.1086	0.4	3.93
0	0.1154	0.2	2.76

Figure 54 Table view of trained model



Confusion matrix

This table shows how often the model classified each label correctly (in blue), and which labels were most often confused for that label (in gray).

True label \ Predicted label	5	6	7	10	12	13	15	20	23	25
5	79%	–	–	2%	–	–	–	5%	–	–
6	–	88%	–	2%	–	–	–	–	–	–
7	–	–	81%	5%	–	–	–	–	–	–
10	1%	1%	1%	73%	1%	1%	2%	4%	1%	1%
12	–	2%	2%	11%	60%	–	–	5%	3%	2%
13	3%	3%	5%	5%	2%	63%	–	2%	3%	–
15	–	–	–	2%	2%	–	73%	2%	–	–
20	–	0%	1%	7%	1%	1%	2%	71%	1%	1%
23	1%	1%	2%	8%	–	–	–	3%	73%	1%
25	–	–	–	3%	1%	–	–	4%	1%	83%

Figure 55 Confusion matrix of model

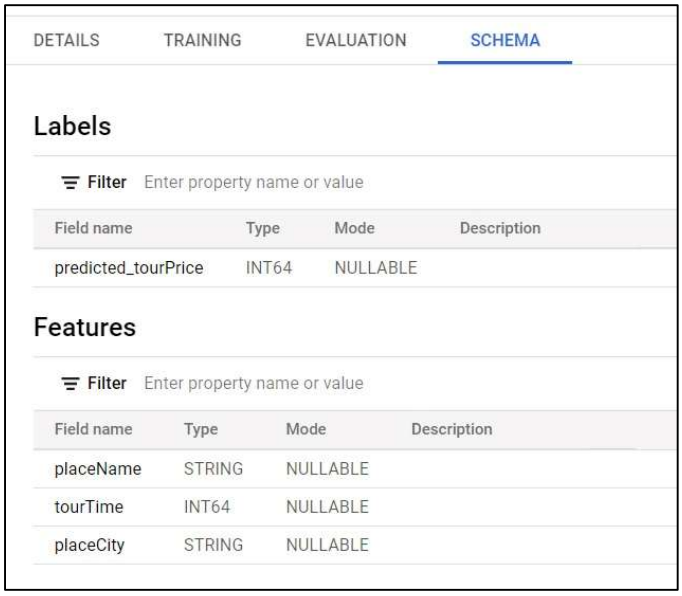


Figure 56 Schema of the model

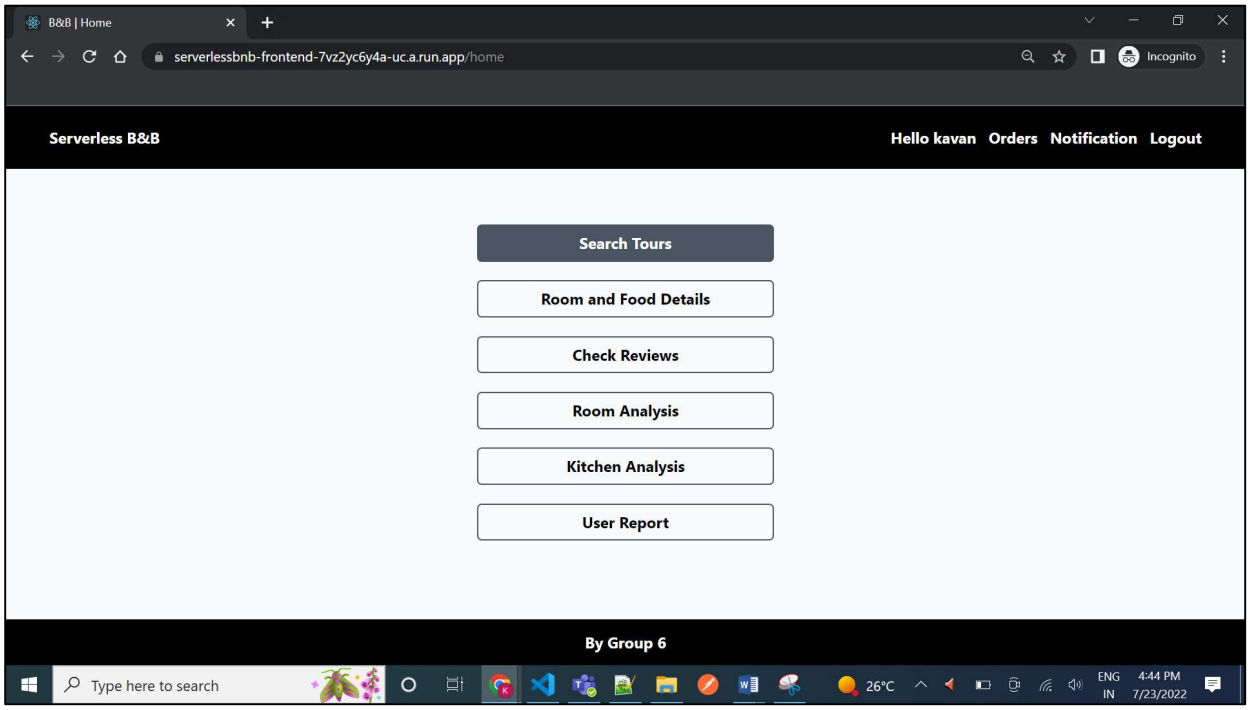


Figure 57 Dashboard to redirect to tour page

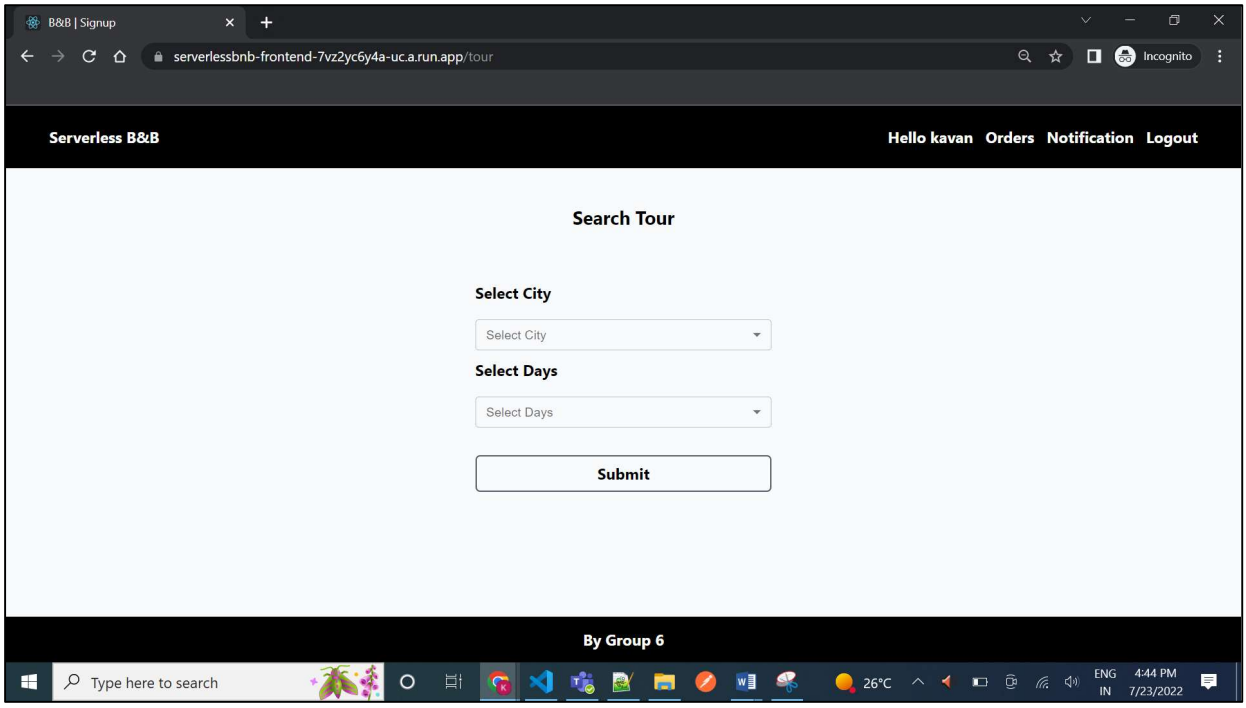


Figure 58 UI of the Tour Page

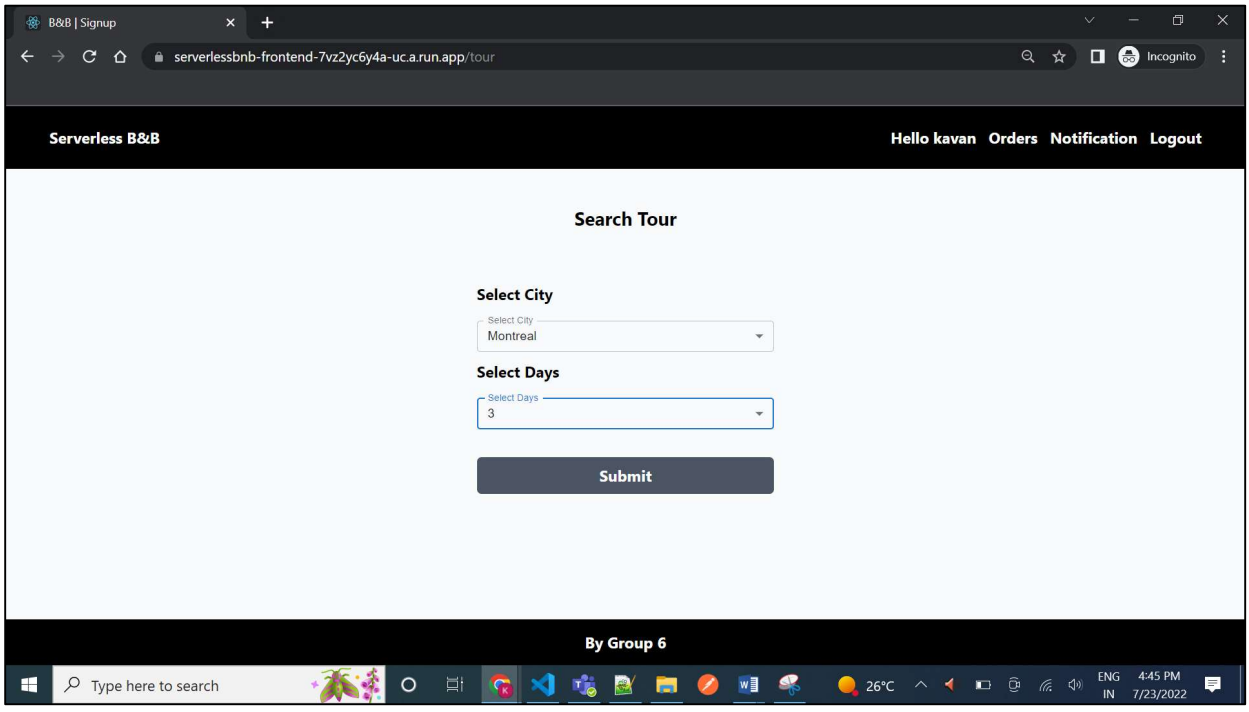


Figure 59 User wants to get tour for City Montreal for 3 days

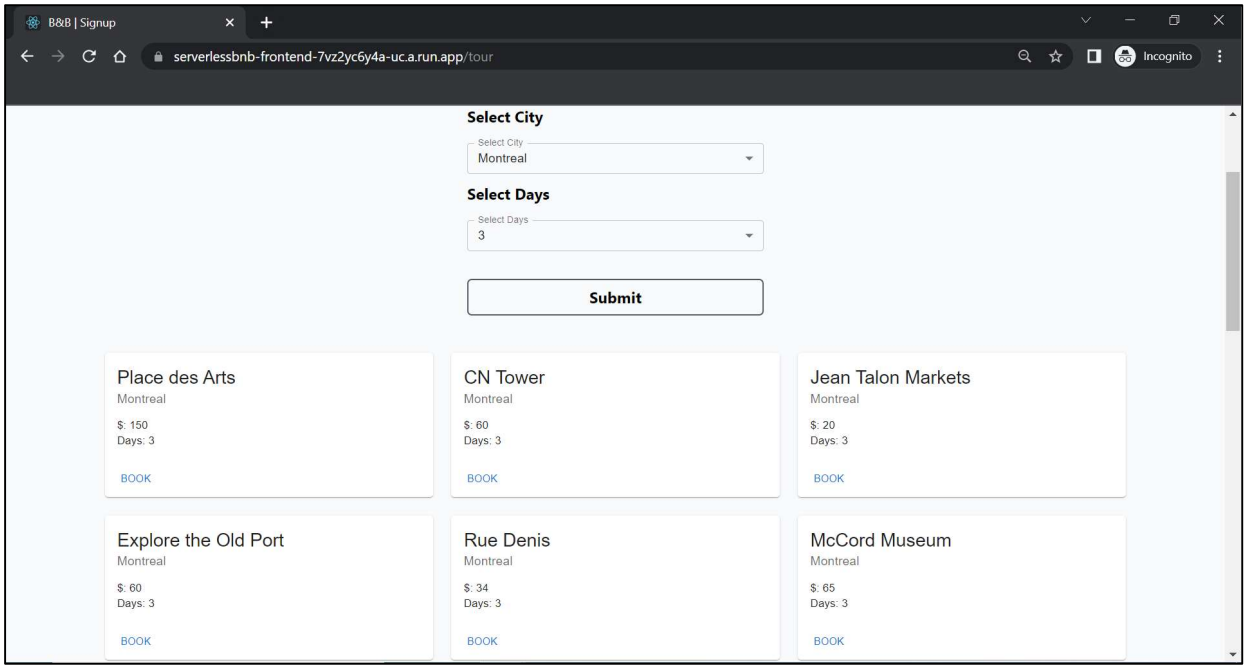


Figure 60 Predicted list of tours gets displayed

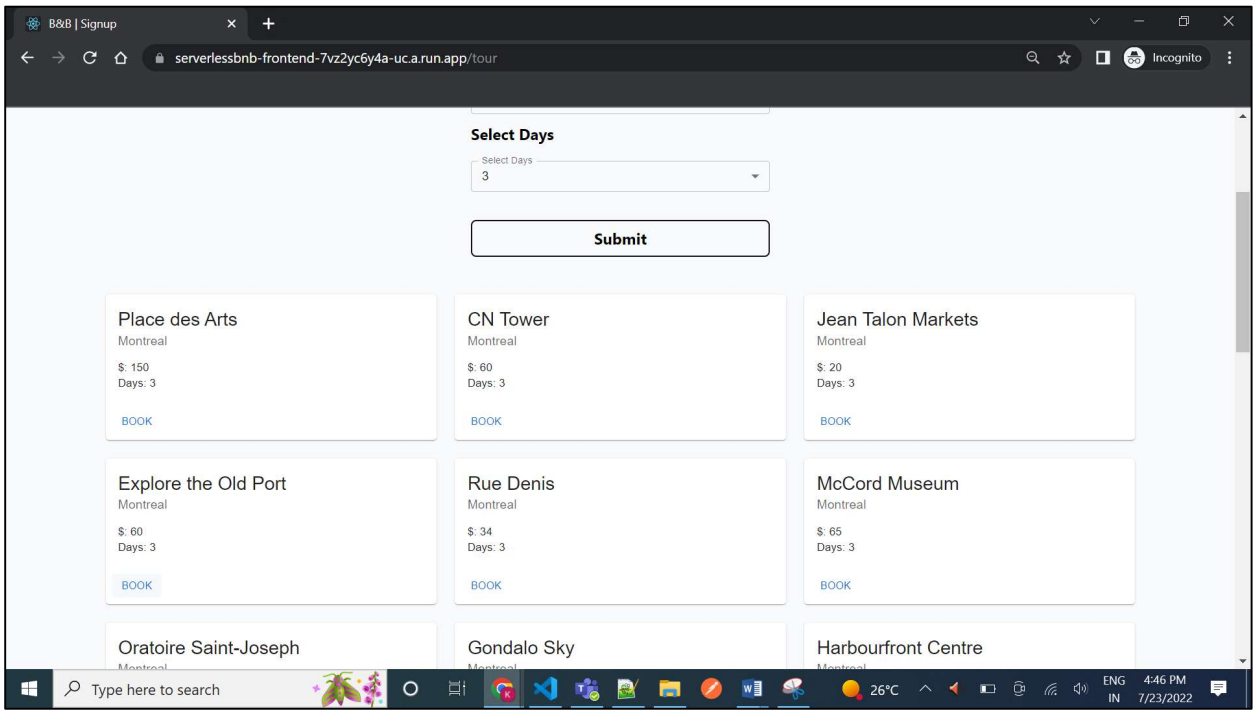


Figure 61 User wants to book tour "Explore the Old Port"

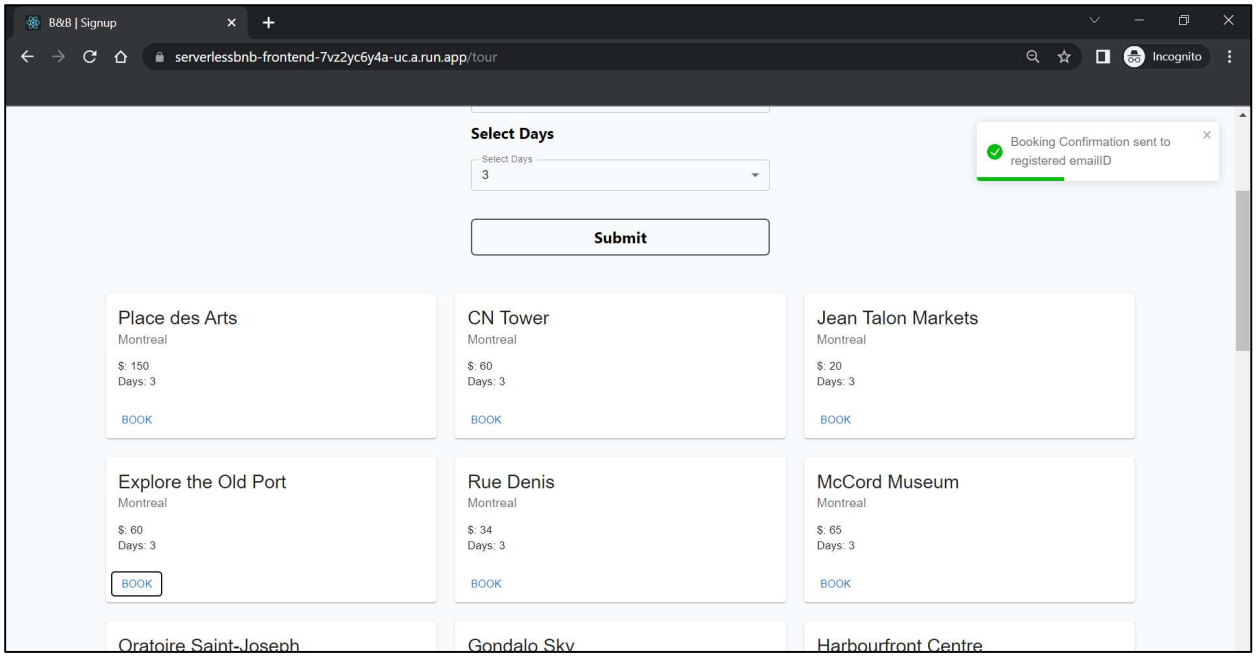


Figure 62 Once clicking on book button, booking confirmation gets displayed and confirmation mail is sent to registered email id

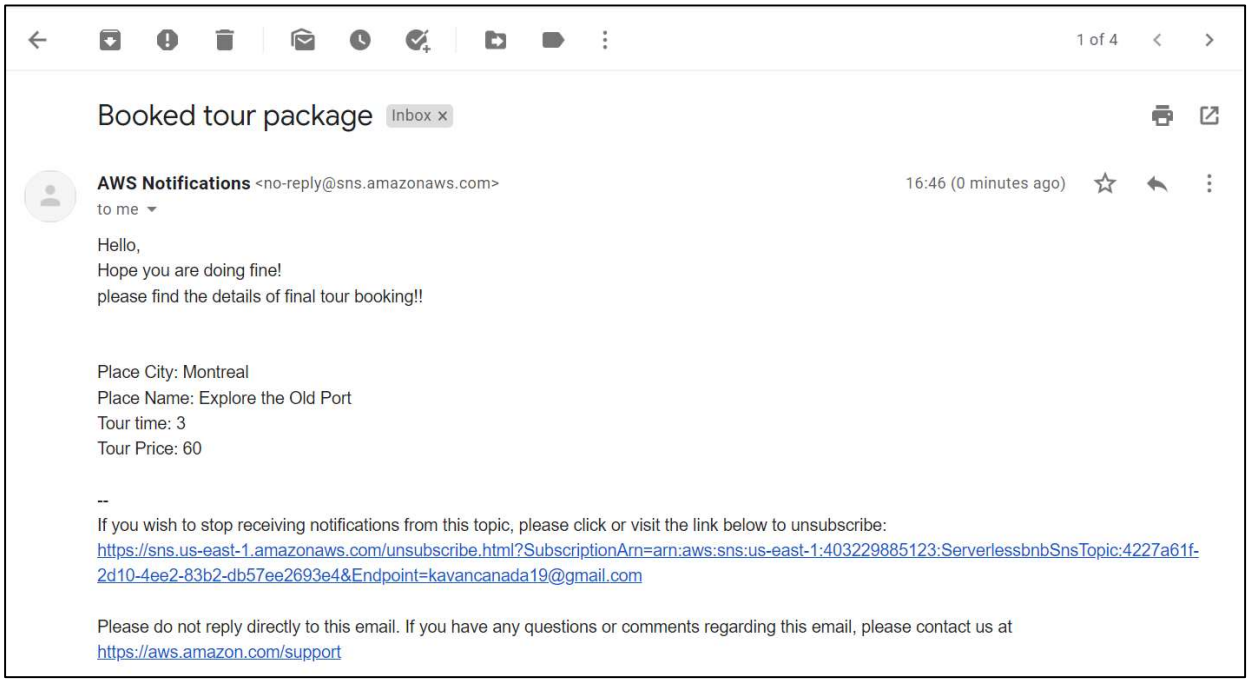


Figure 63 Confirmation mail for tour confirmation

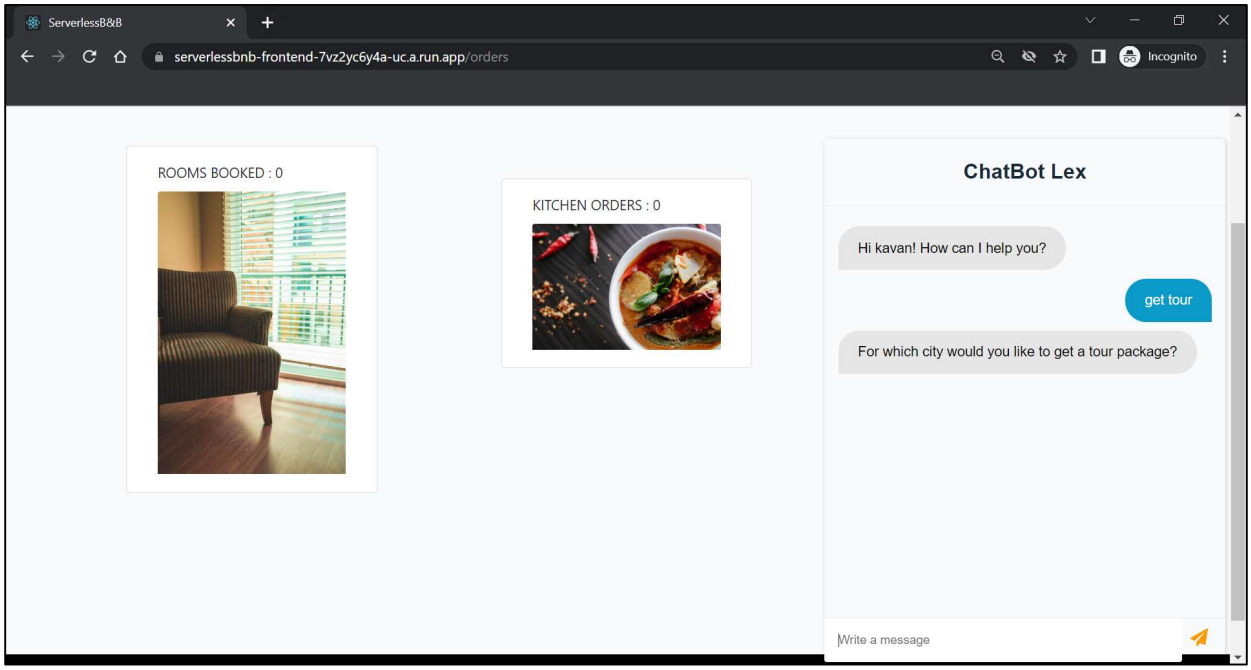


Figure 64 Tour service using Lex chatbot for guest users

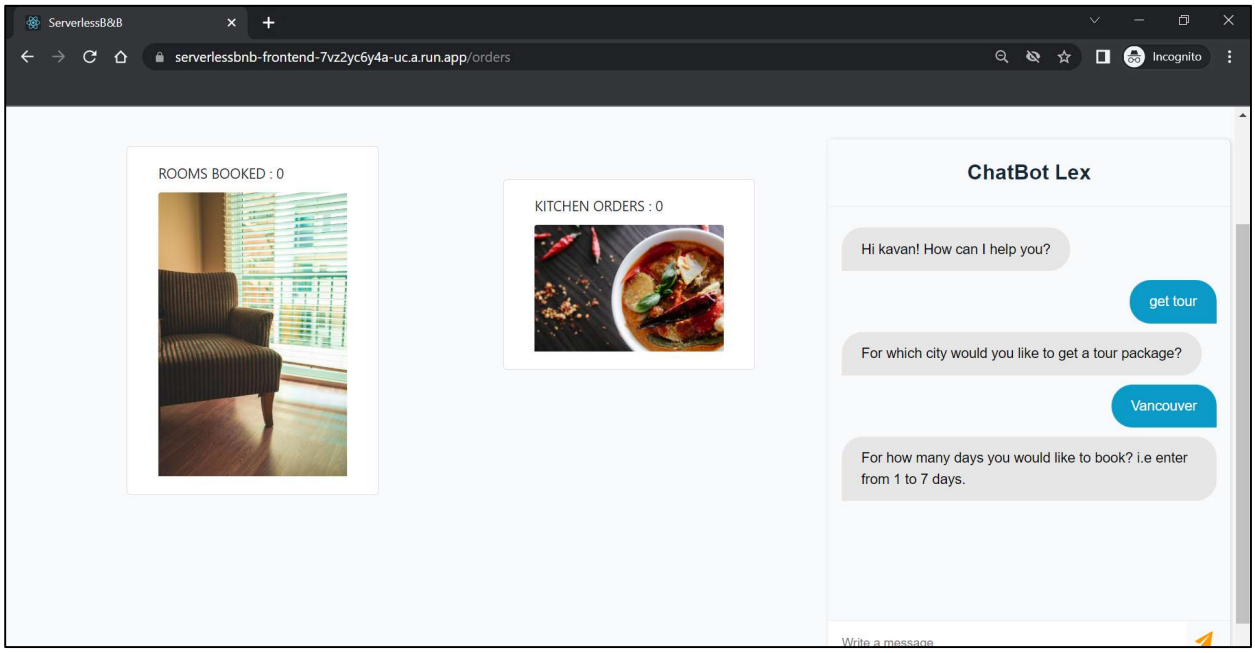


Figure 65 Guest users enters Vancouver as city

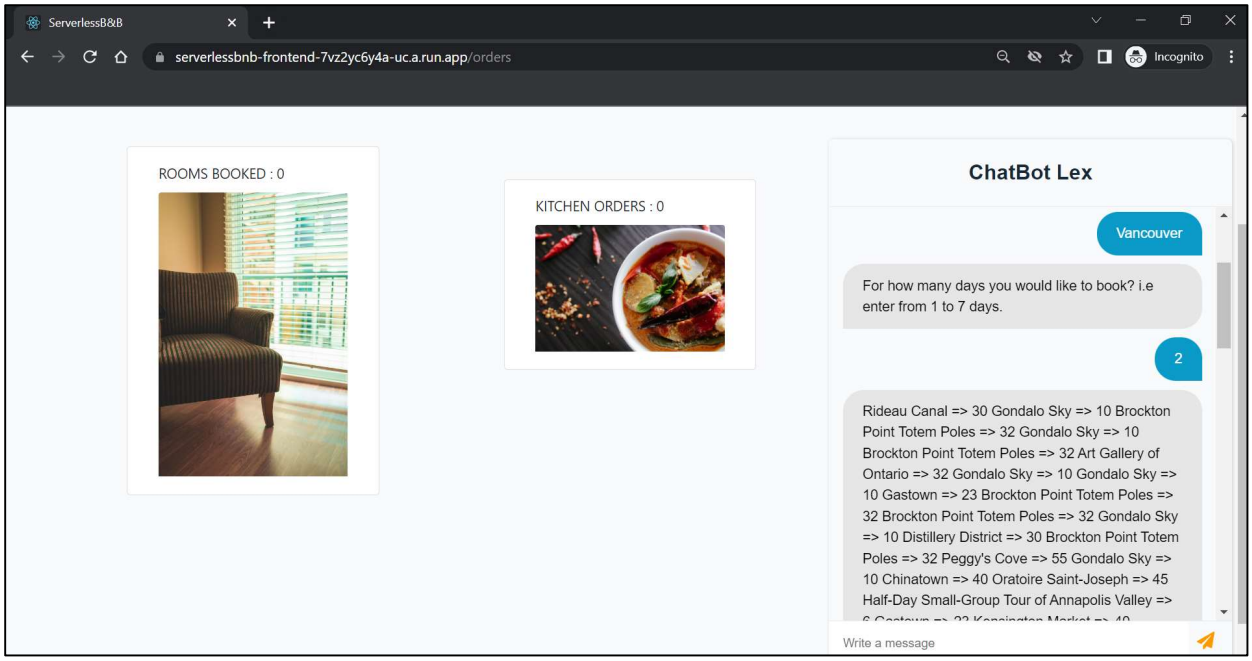


Figure 66 Guest users enter 2 as the number of the days and lex provides the list of tours fetched from BigQuery ML

Screenshots and Test Cases:

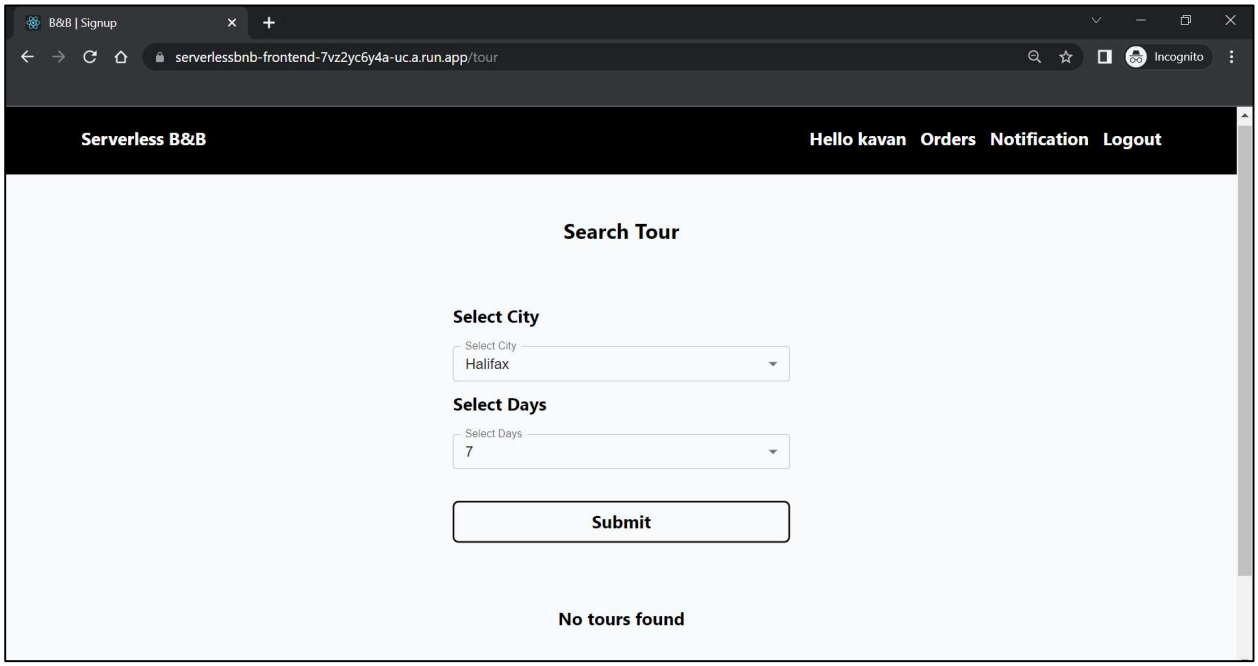


Figure 67 No tours found for given inputs

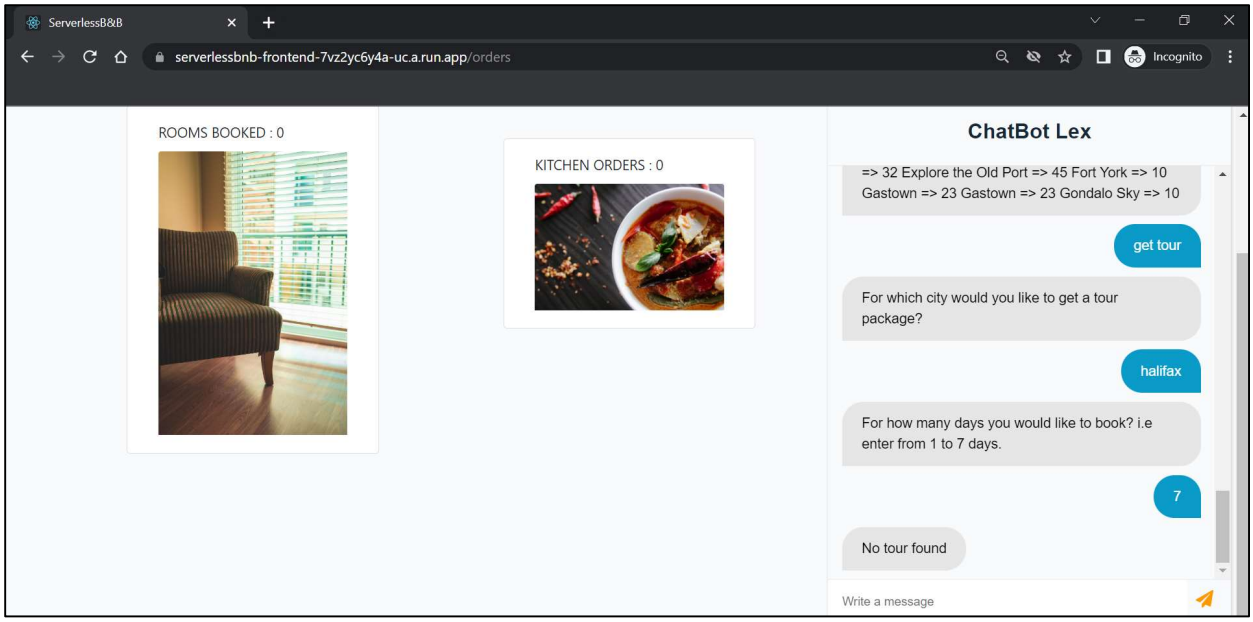


Figure 68 No tours found for given inputs through Lex

3.5 Message Passing Module

This module decided to use gcp pub/sub services to allow the authenticate user to communicate with tour operator and hotel management. Unauthenticated are not allowed to use these services. Authenticate user can book a tour, book a room/bed, and order a breakfast with serverless B&B. If the service book successfully it acknowledged by tour operator, hotel management and kitchen. Figure 69 shows how the message passing module works. It starts from tour page or hotel page. From here use can book a tour or book a room. Then it will automatically check the availability of the tour or room. If the service is available, then the tour or room will book and respond to user while GCP pub/sub will also send a message to tour operator, hotel management and kitchen. Tour operator, hotel management and kitchen can acknowledge services are created, so they can prepare the service.

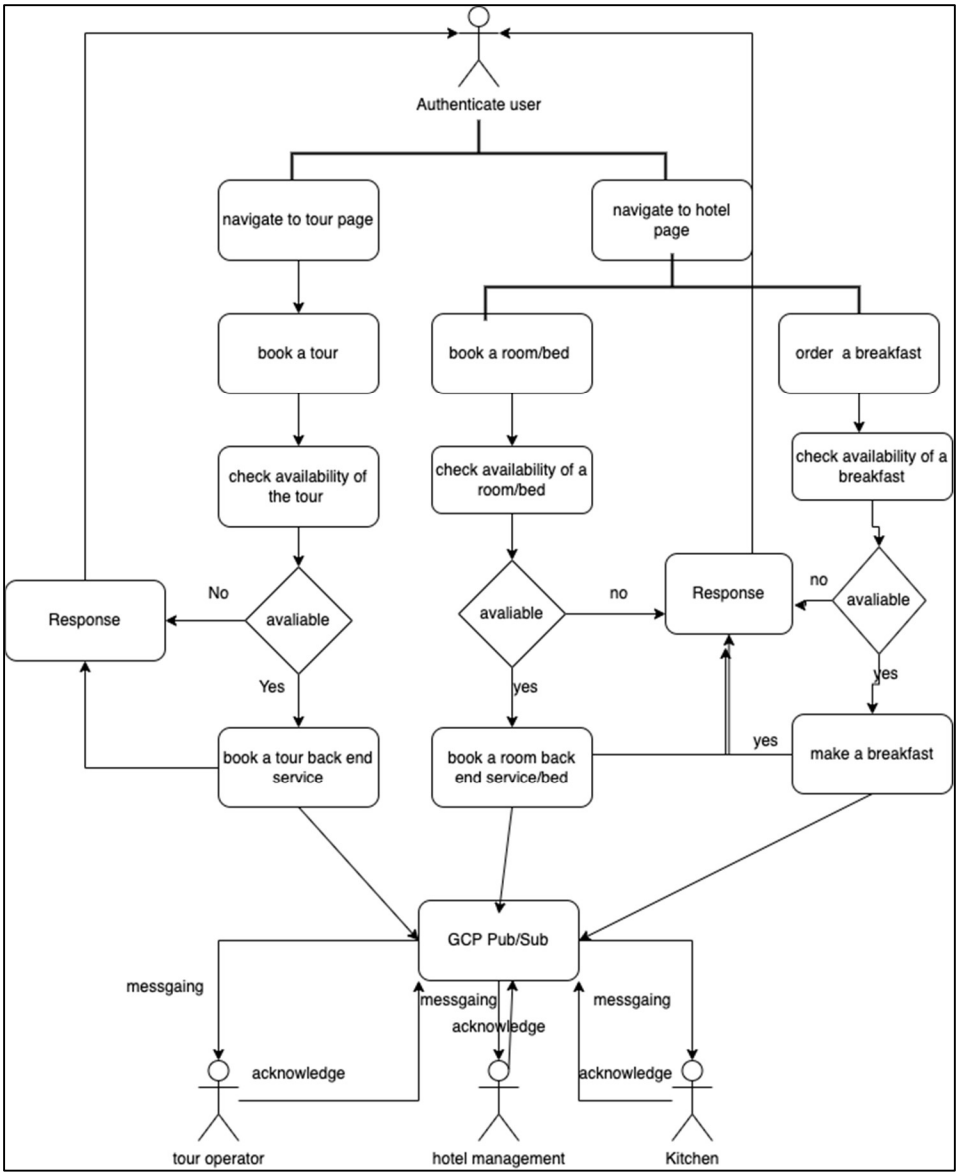


Figure 69 Flow chart of message passing module

Pseudo Code:

Step 1: When the use fills the data from and sent requires by api call

Step 2: Correspond aws lambda backend service will be trigged

Step 3: Inside this lambda function, create a gcp pub client

Step 4: Use gcp pub client to publish the message into queue for later access.

Step 5: Then create another lambda function to consumer the message.

Step 6: Create a lambda function trigger by an api gate way.

Step 7: Inside this lambda function create a sub function to pull the message.

Step 8: Pass the message to the http response body

Step 9: Frontend application receive the http response and parse the data

Step 10: Render the date and show the message to the user.

Screenshots and Test Cases:

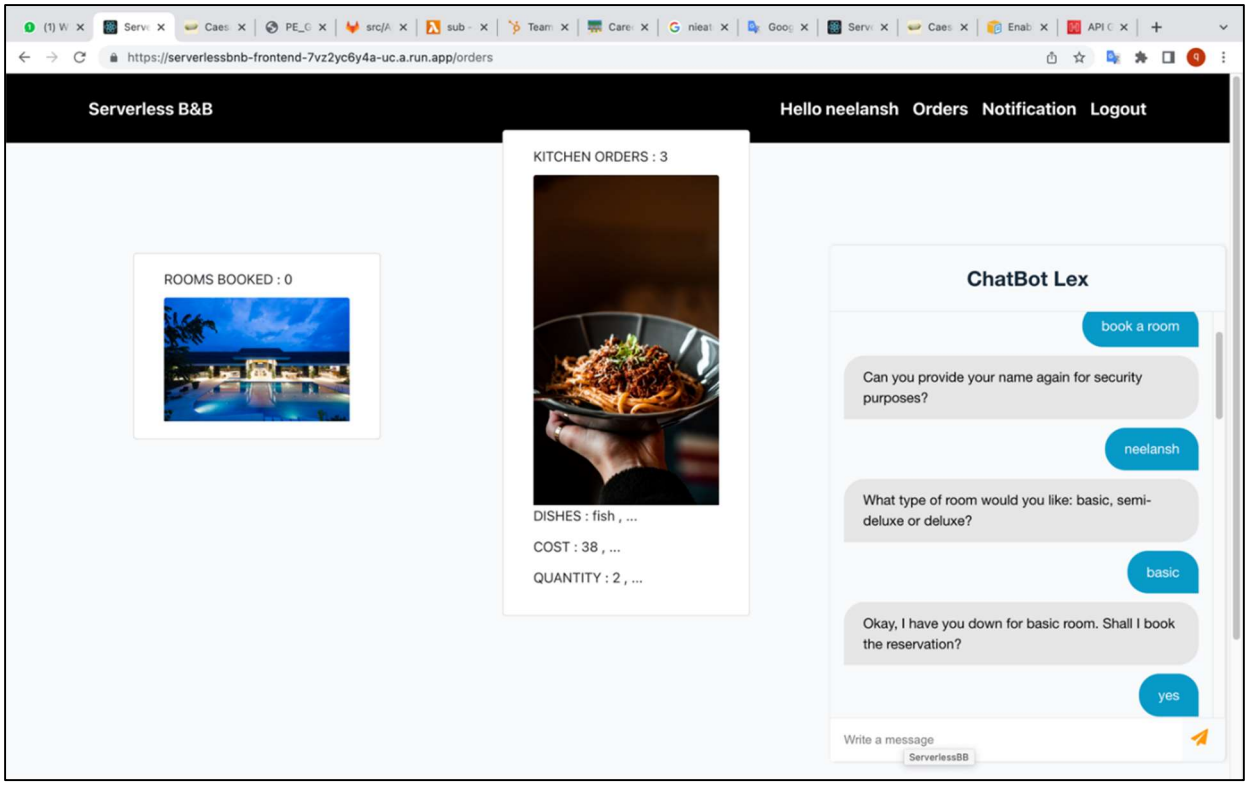


Figure 70 book a room using Chatbot Lex option

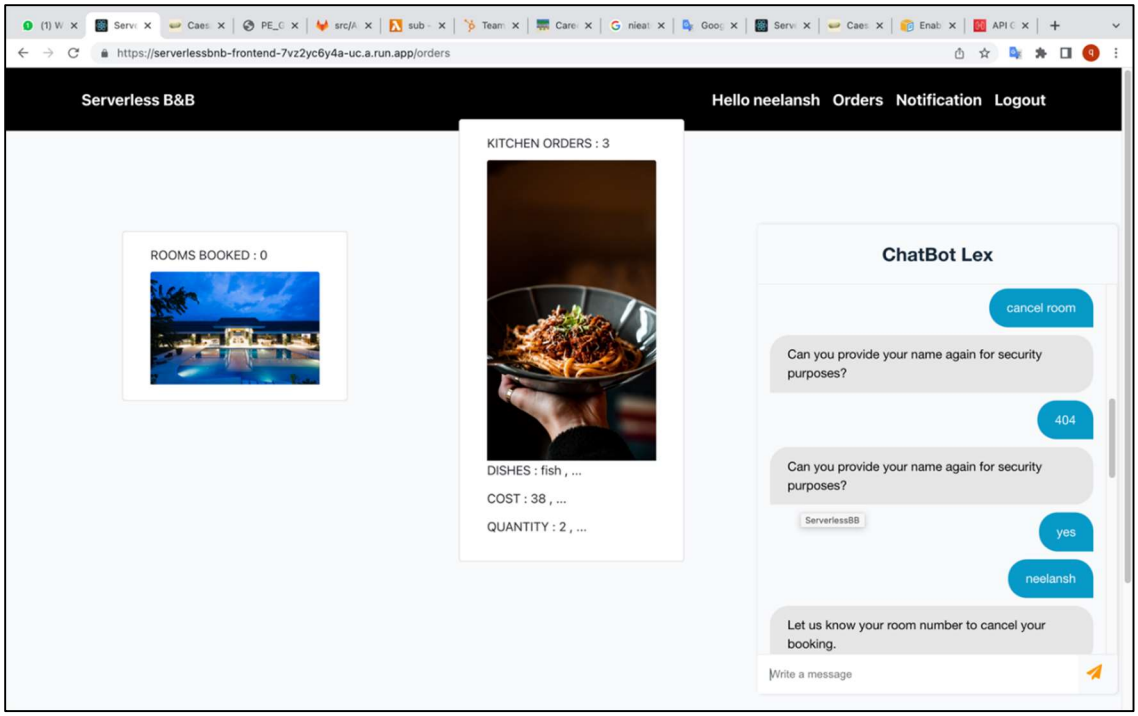


Figure 71 cancel a room using Chatbot Lex option

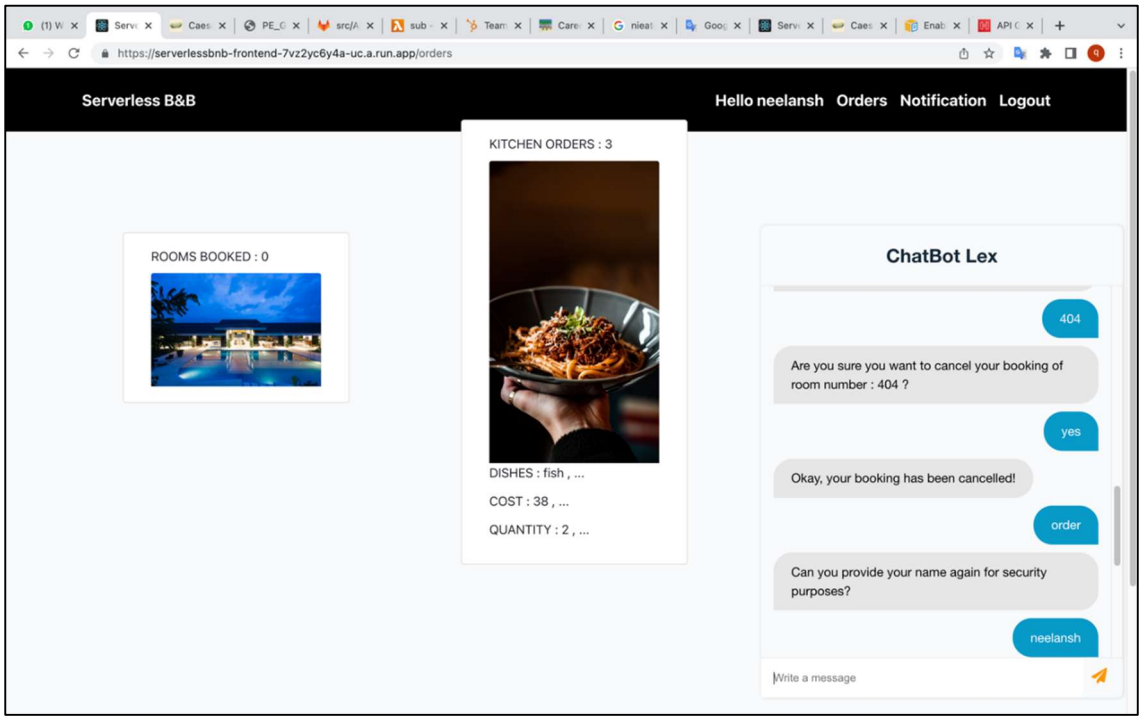


Figure 232 cancel a room using Chatbot Lex option

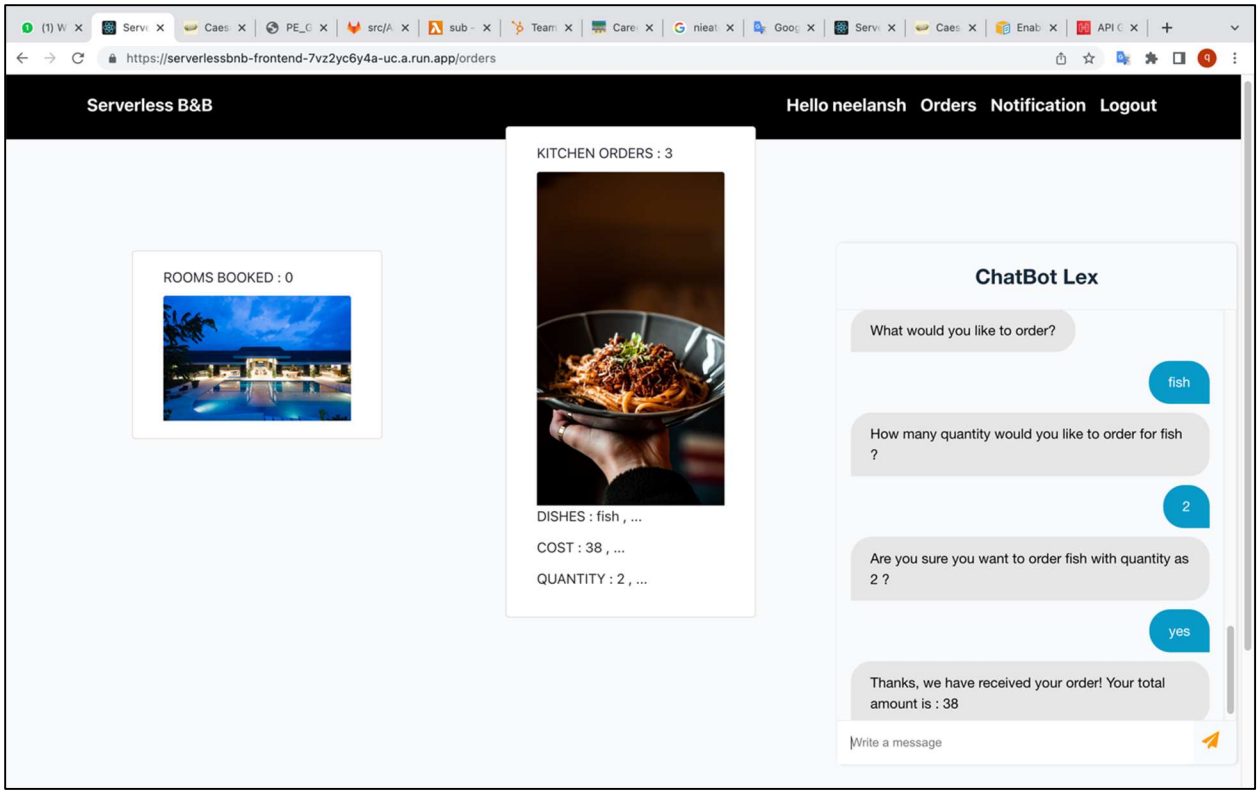


Figure 73 order a breakfast using Chatbot Lex option

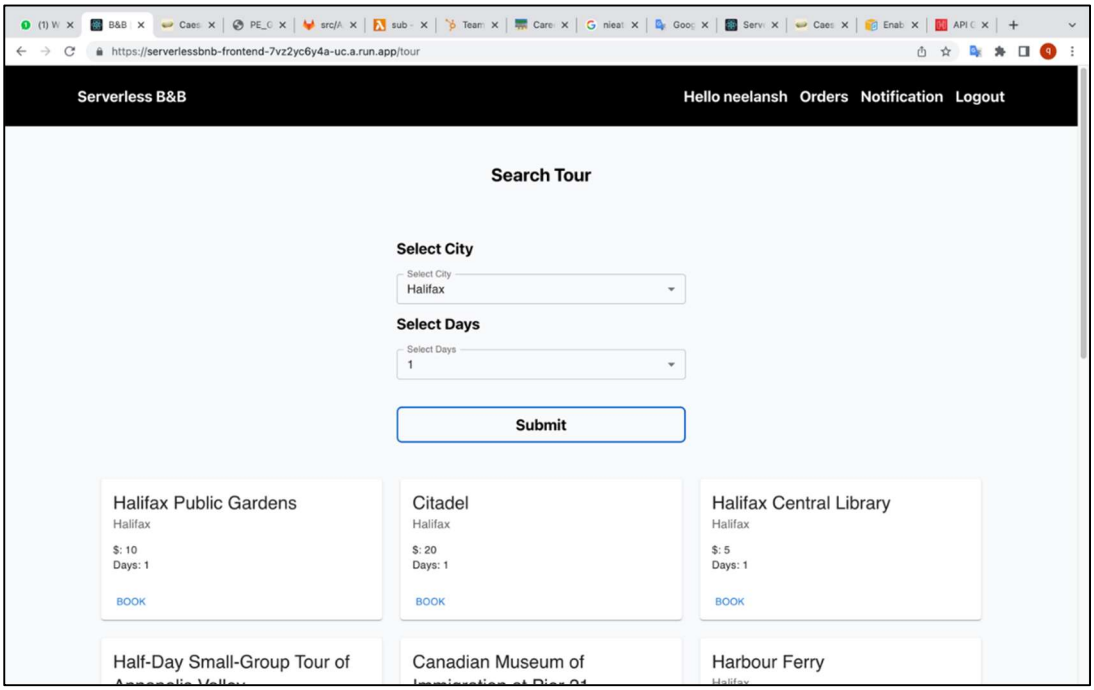


Figure 74 book a tour using UI

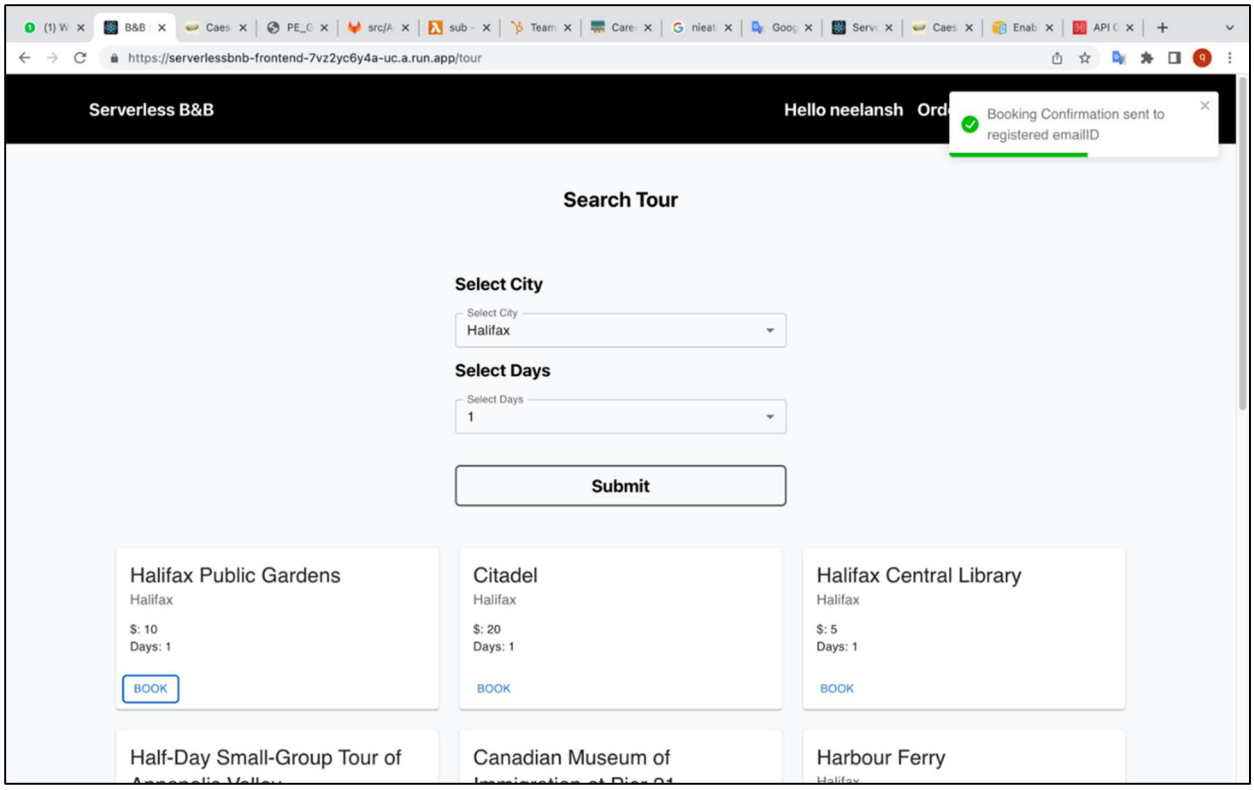


Figure 75 book a tour confirmation

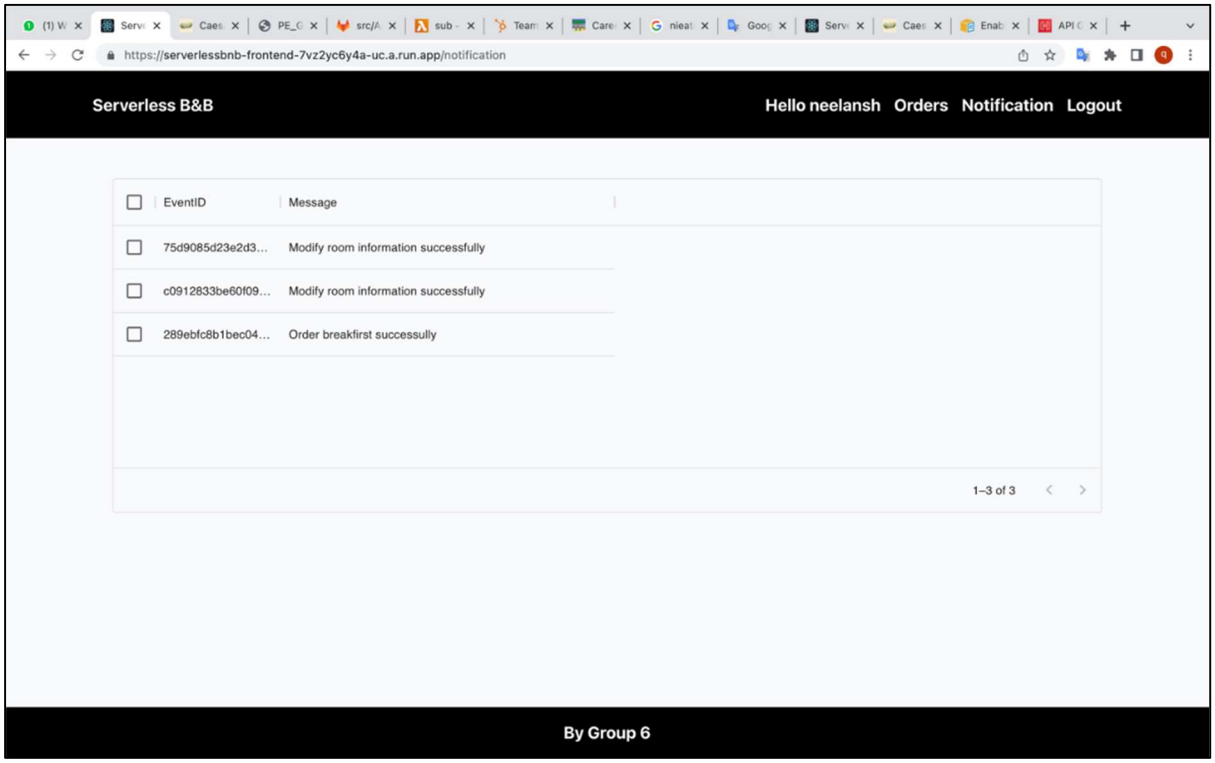


Figure 76 show the message in notification

3.6 Machine Learning Module

The software will be able to compile client feedback and provide a thorough analysis from it. The program will make use of the Google Cloud Natural Language API, which uses Lex data to train, test, and deploy a machine learning model that produces a prediction or analysis report automatically. Using structured feedback data provided by the endpoints, it automatically creates and deploys ML models. It automates modelling using many different forms of data [15].

There are two different kinds of analysis to be done. The first step in the analysis is to determine which customers' stays are comparable so that the application may suggest a trip package. Based on customer input, the second study will determine which tour package is the most well-liked by consumers [15]. This information will be used to assign a score that will assist management decide how to enhance their services. This is accomplished using techniques based on logistic regression from Google Cloud's big query ML.

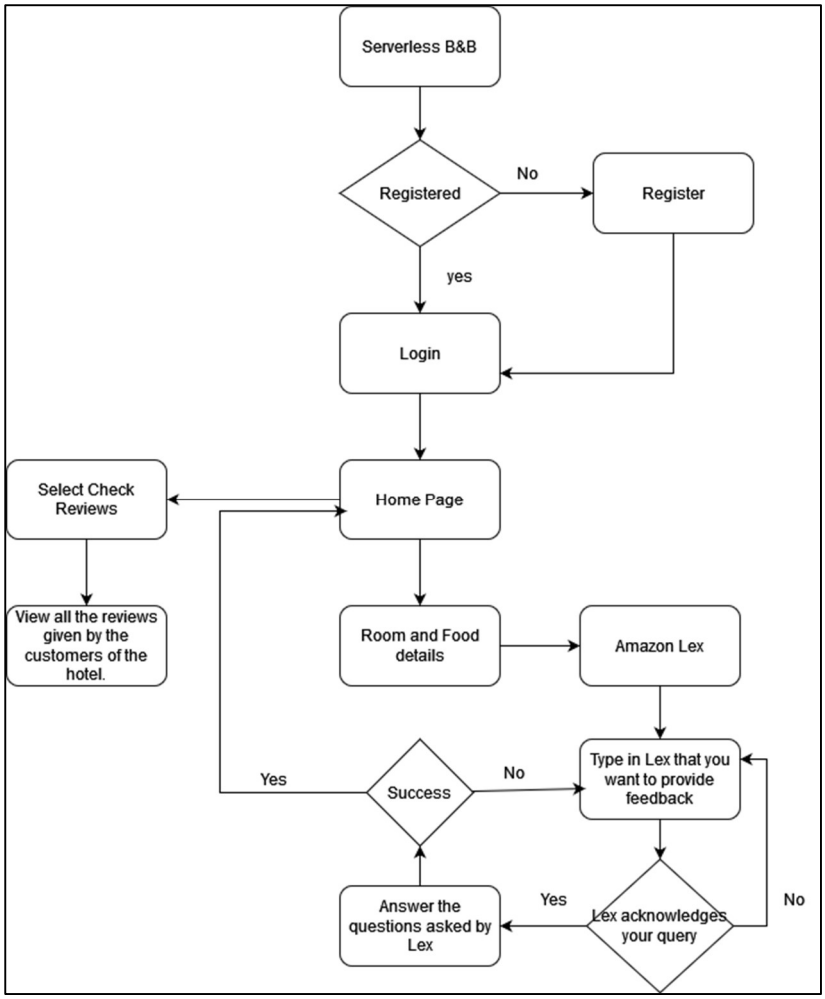


Figure 77 Flow chart for feedback module

Pseudo Code:

Step 1: The user first has to enter the registered Id and password to authenticate itself.

Step 2: If the user is not valid user, then user can navigate through the website only or register to use the services offered by serverless B&B.

Step 3: From Homepage, the user navigates to chatbot by clicking the book room and food services tab.

Step 4: The user can use Lex to provide feedback about the hotel. Lex contents intent which would listen to sentences related to review/feedback. After answering the questions provided by lex, from system side the lambda function gets triggered and receives answers to the queries as the input for the lambda. User can navigate to the homepage to see the review posted by him.

Step 5: The lambda calls the google cloud function which contains natural language api which would help in sentiment analysis of the feedback, and it returns the response back to lambda. The lambda stores the responses in dynamo db.

Step 6: User can view all the reviews about the hotel services by going to the check reviews tab.

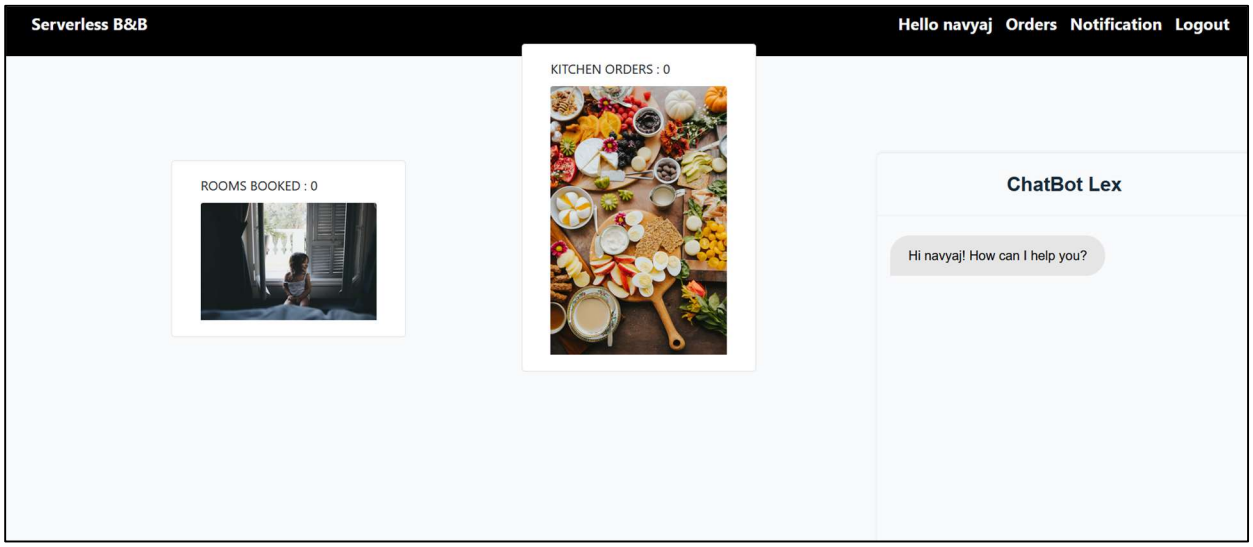


Figure 78 In Book room and food page

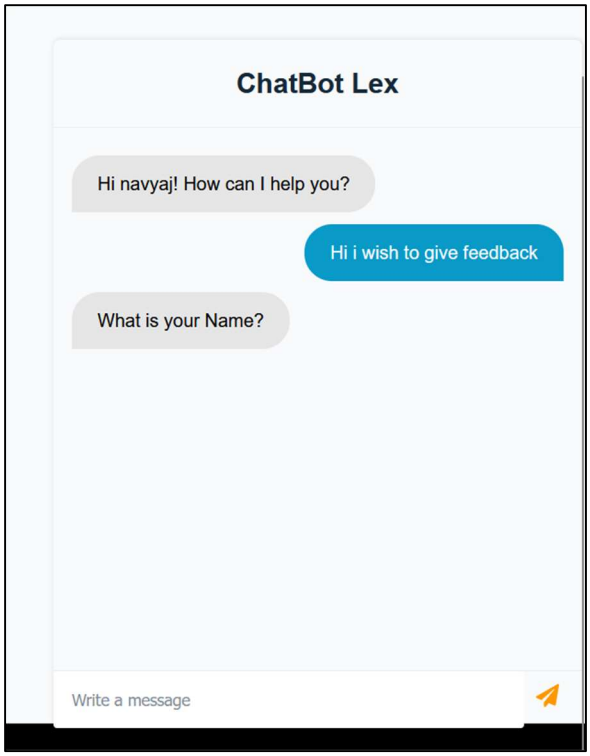


Figure 79 Lex Prompts on feedback

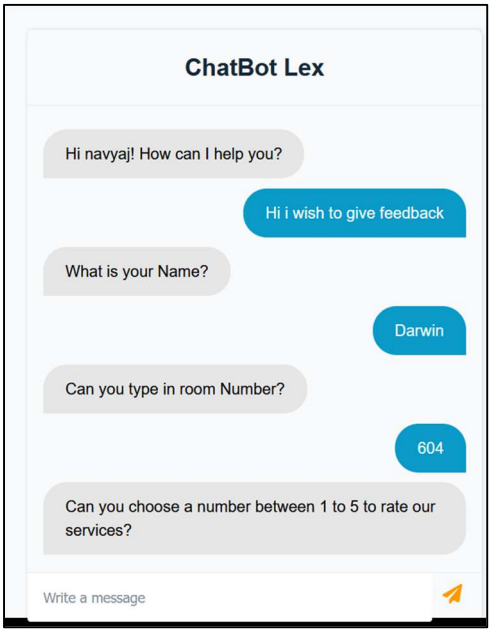


Figure 80 Lex prompts on feedback contd.

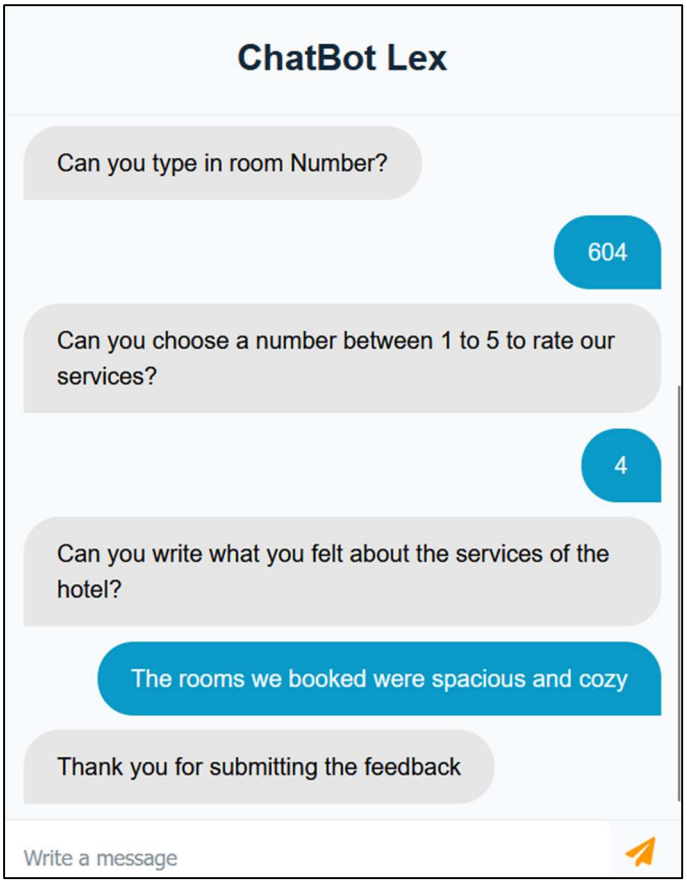


Figure 81 Full functioning conversation between lex and user.

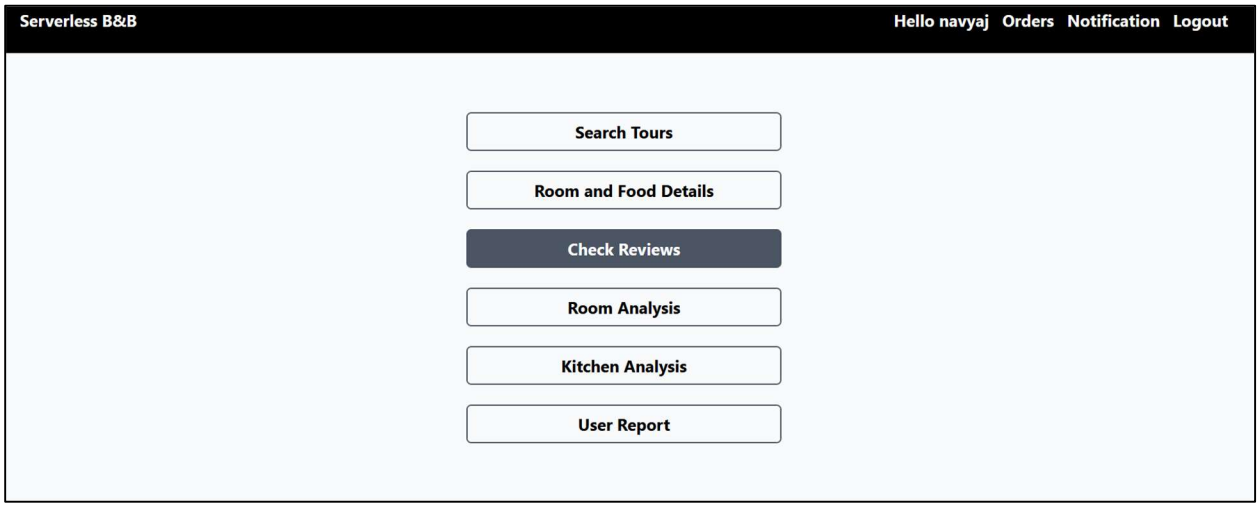


Figure 82 Home page with check review tab.

Figure 83 Inside check review page

3.7 Web Application Building and Hosting

We have created the front end of the application using the ReactJS web development framework, and we have used both NodeJS and python to build the back end of this serverless application. The reason to choose ReactJS is that it provides an interactive layout of any UI. Furthermore, it enables rapid and high-quality application development that saves time for both client and developer. ReactJS is a perfect combination of JavaScript and HTML tags. The syntax of JavaScript and HTML is always used which simplifies writing the code. Apart from that, it is extremely competent, easy to adopt, makes template design easy, and provides good developer tools.

For the deployment of the application, we have used the GCP Cloud Run service. It develops and deploys highly scalable containerized applications using JavaScript a fully managed serverless platform. We have also used the GCP Artifact Repository for storing the application's centralization image, which has been used by the Cloud Run during the hosting of the application.

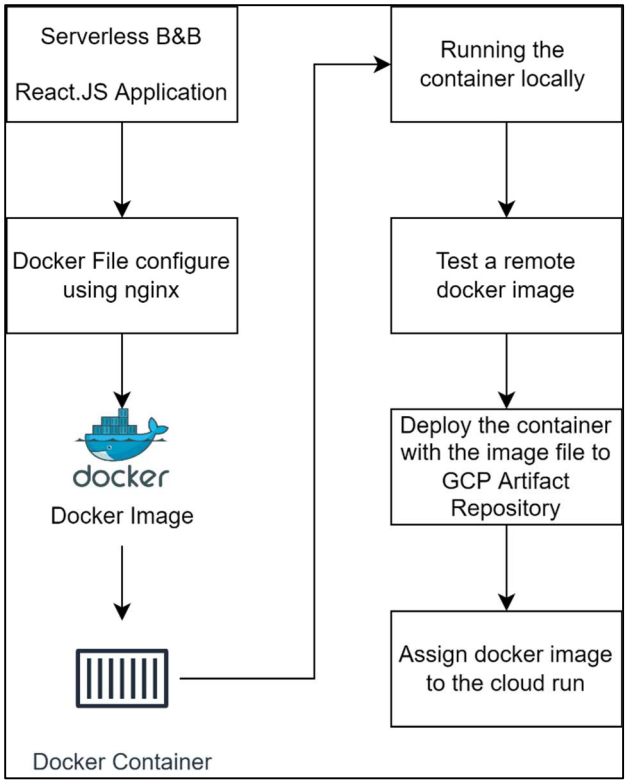


Figure 84 Flowchart of Deployment module

Pseudo Code:

Step 1: Cloud Run required the docker image for the deployment of the application. So, for that, first, we have created a docker configuration file that contains the Nginx engine to build and run the react application.

Step 2: Generated docker file used to build the docker image in docker container locally.

Step 3: Run the created container locally and test whether the application is running as per the expectation or not.

Step 4: Once the created image is successfully tested, that has been pushed to the GCP Artifact Repository which stored the application's Docker image.

Step 5: Finally, that docker has been assigned to the cloud run which will use it and deploy the application.

Screenshots:

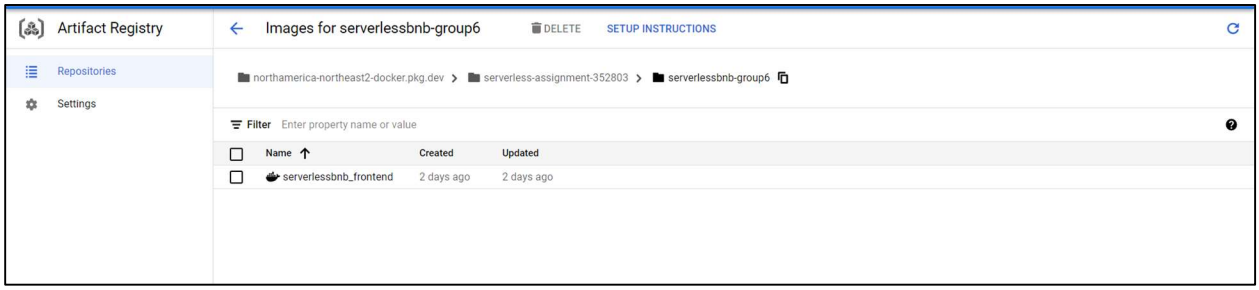


Figure 85 Pushed docker image in Artifact Repository

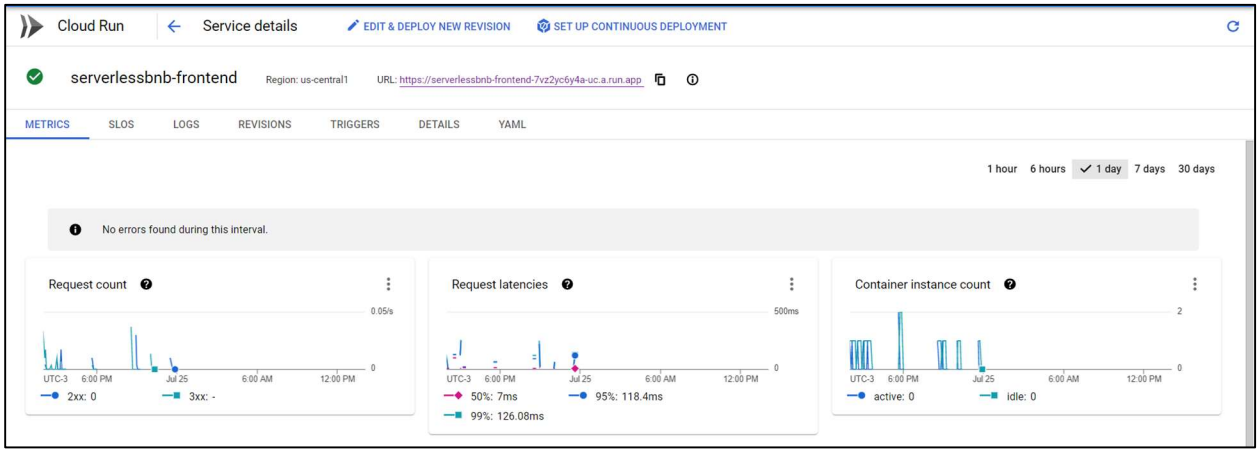


Figure 86 Deployed the application through cloud run

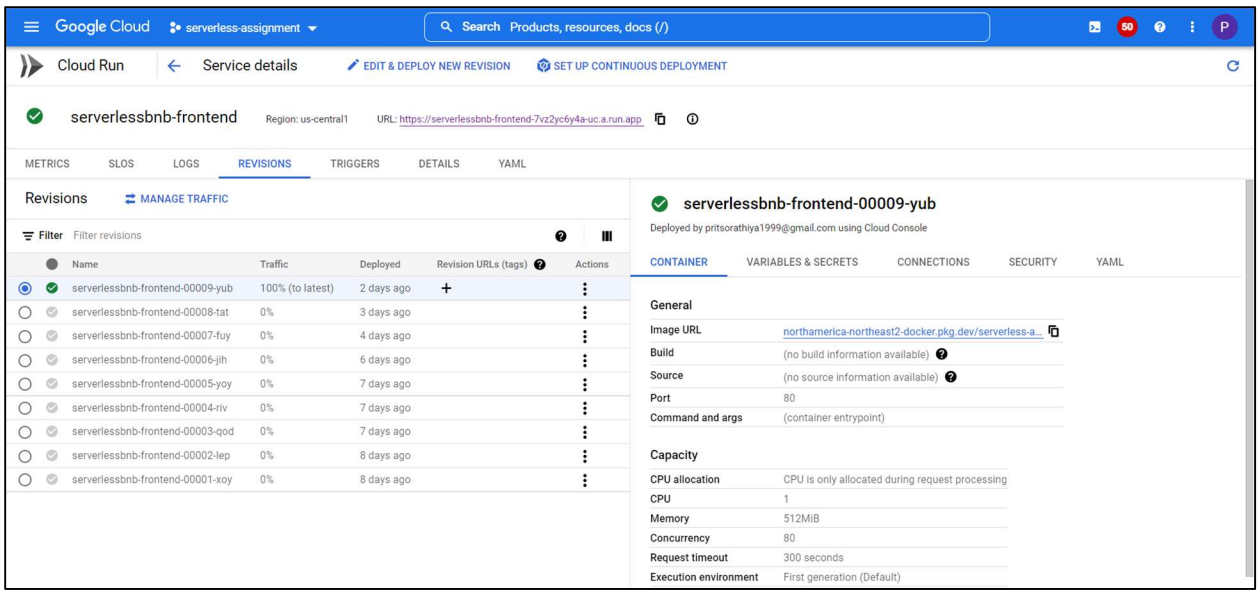


Figure 87 Configuration details of the deployment

3.8 Report Generation and Visualization Modules

For visualization – building graphs, charts etc. we are using Google' data studio visualization. It is an online tool that helps visualize data for free. The AI of Google Data Studio combines data from sources, and it automatically analyze those data and enables to make interactive charts, graphs [13].

Initially we were going for AWS Quicksight for report generation. We had studied completely how report should be generated from Quicksight. The report should be of user activity and our user activity data is stored in Dynamo DB. But Quicksight does not fetch data directly from Dynamo DB. It fetches from S3 bucket.

Therefore, we must also store data in S3 bucket as the data from Dynamo DB is used by the lex module. The other option was to create a data pipeline for streaming data from dunamo to S3. But it is not cost effective. After receiving feedback from professor, to make application cost effective, we decided to do the report generation in google data studio. Another small advantage of data studio is it creates better visuals than quicksight.

For creating report in data studio, we just needed another lambda function as the data is stored in firestore database of Room and Kitchen.
Data pipeline:
Running on AWS: High Frequency: $1/month, Low Frequency: $0.60/month.
Running on Premise: High Frequency: $2.50/month, Low Frequency: $1.50/month.

Lambda function:
Duration: $0.0000166667 for every GB-second, Requests: $0.20 per 1M requests.

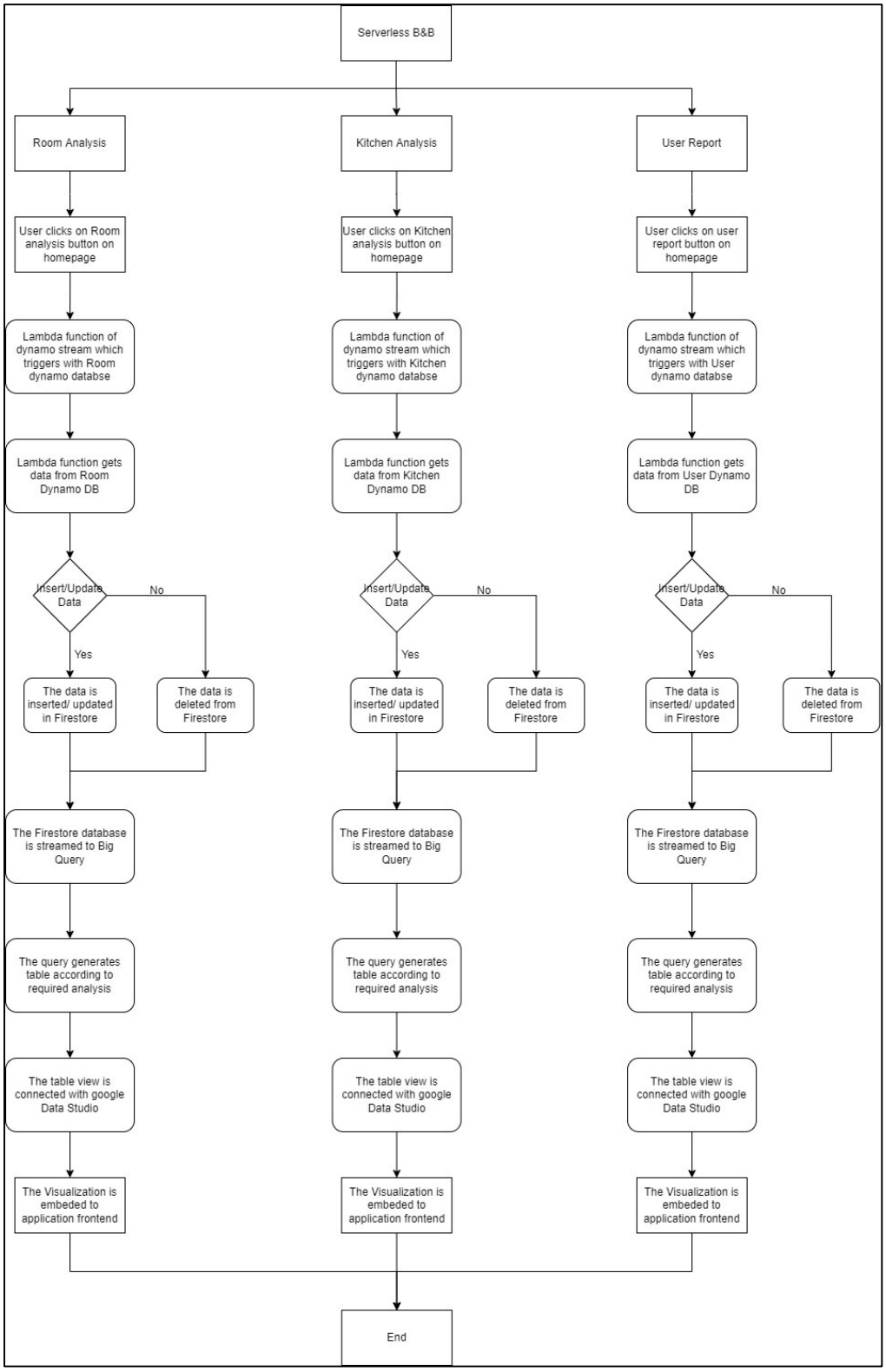


Figure 88 Flowchart of visualization module

Screenshots and Test Cases:

The flow remains same for all three analyses, that is, Room, Kitchen and User report.

Step 1: Logging in the homepage.

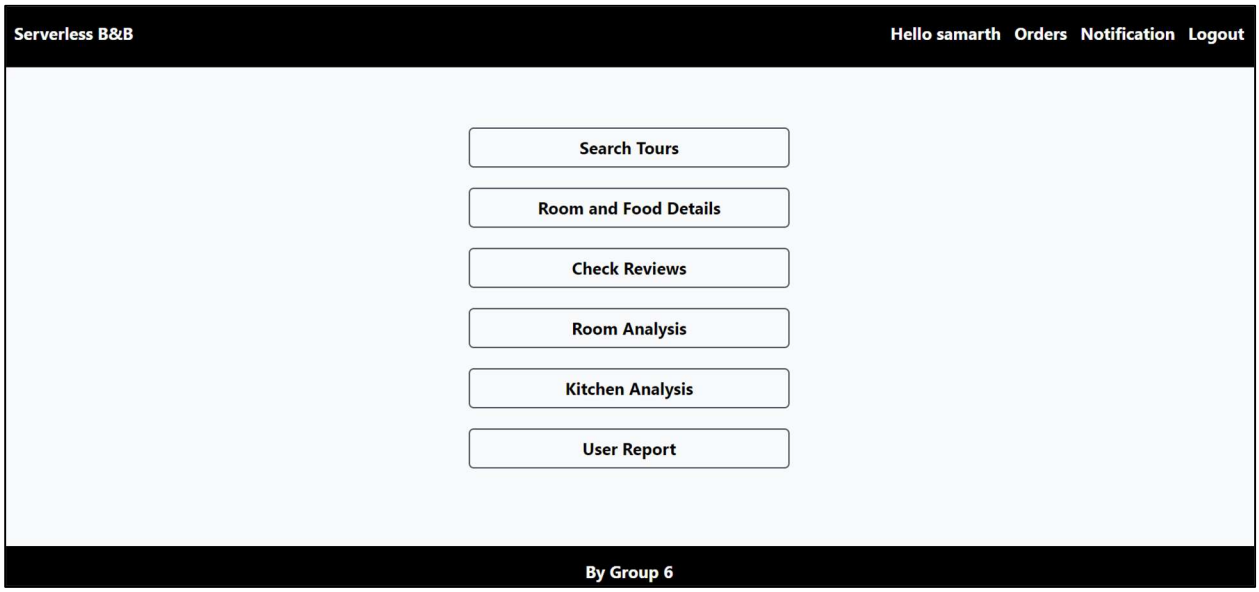


Figure 89 User homepage

Step 2: Ordering food to get updated analyses on kitchen.

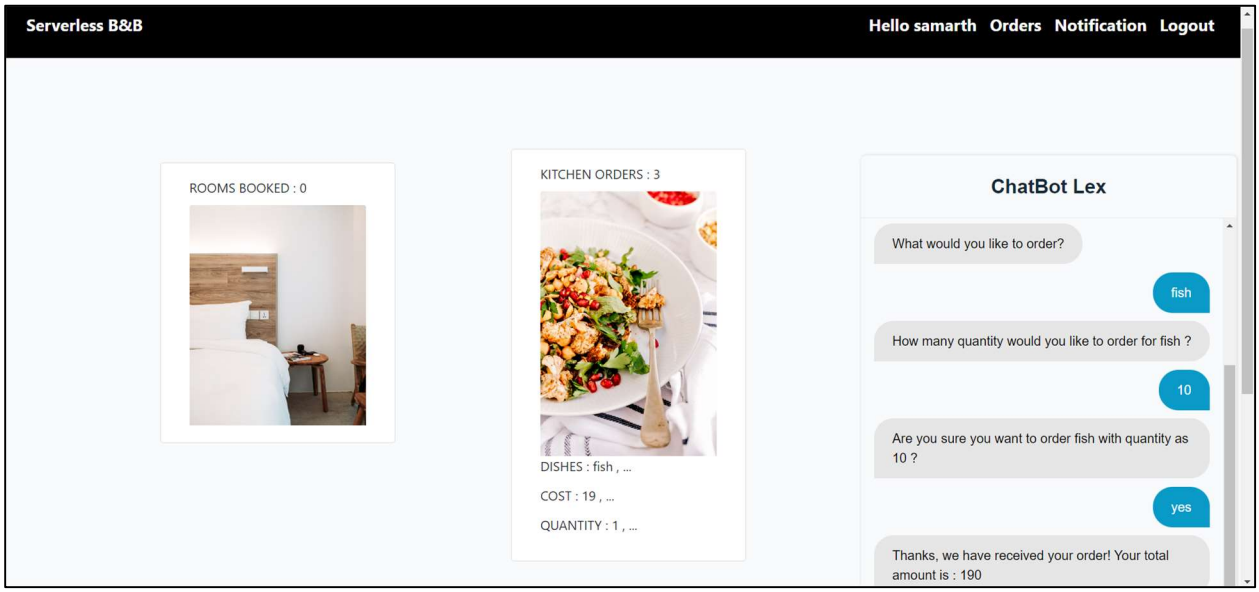


Figure 90 Ordering food for kitchen analysis

Step 3: Lambda function which triggers on dynamo DB data stream. It fetches the latest update and sends data to Firestore.

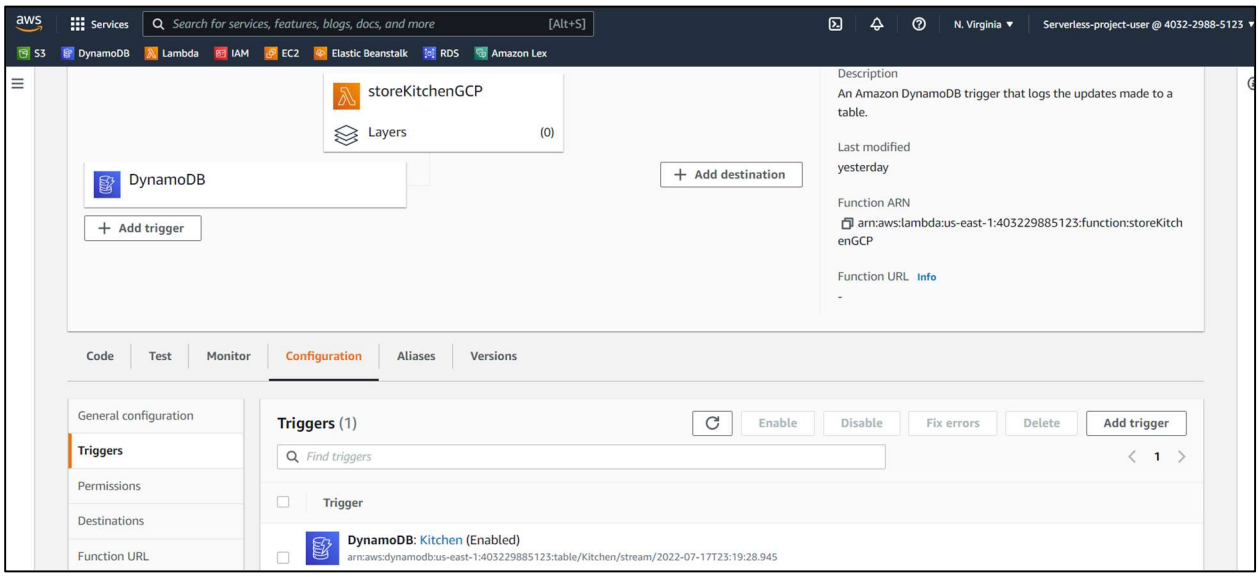


Figure 91 Lambda is triggered as there is any update in Kitchen database

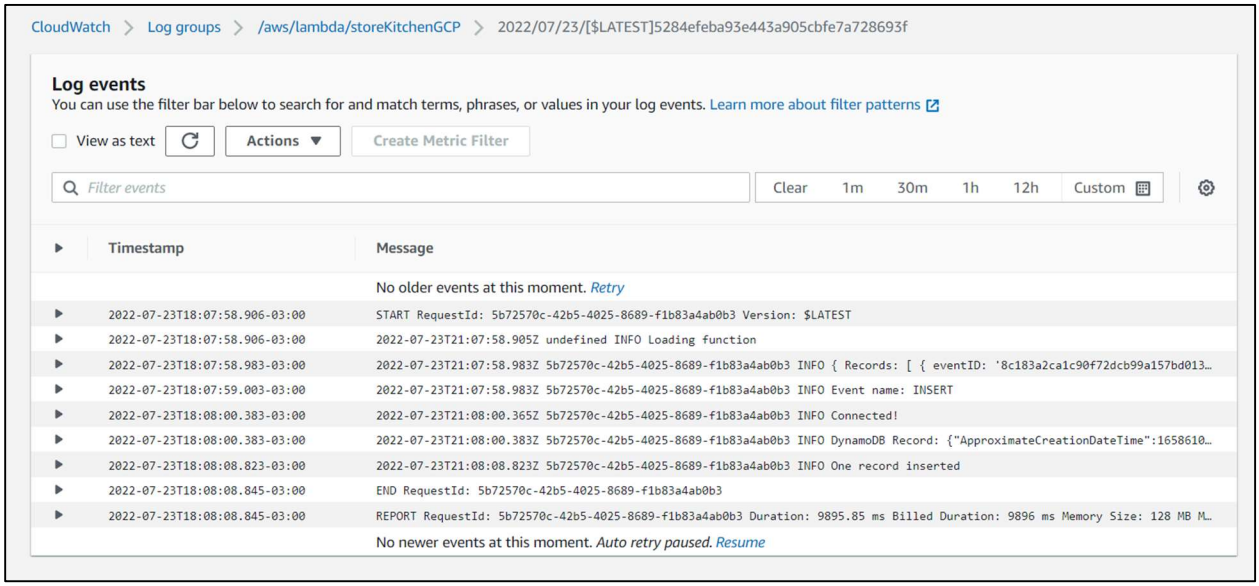


Figure 92 Cloud watch logs for the data inserted

Step 4: The data is updated in fire store. That is, in kitchen collection with dish name as fish and quantity 10.

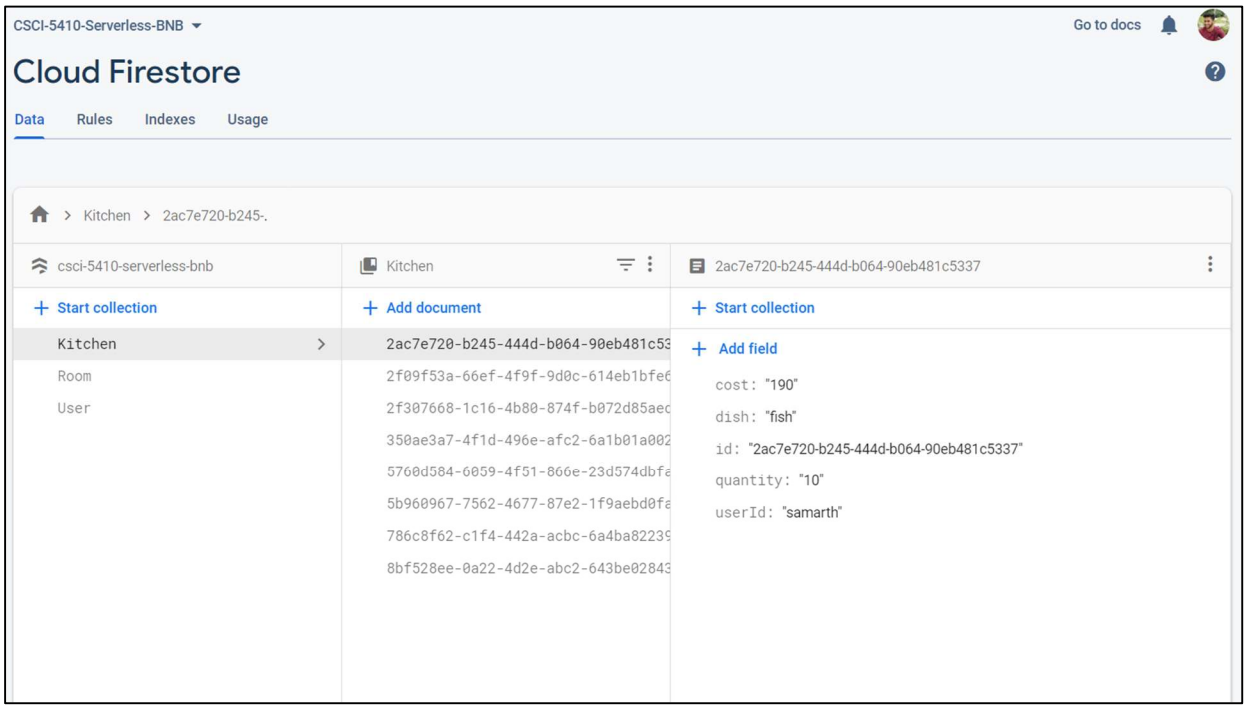


Figure 93 Data updated in Firestore

Step 5: The collection is streamed to big query to create table and table view, as google data studio does not do visualization on collections and documents.

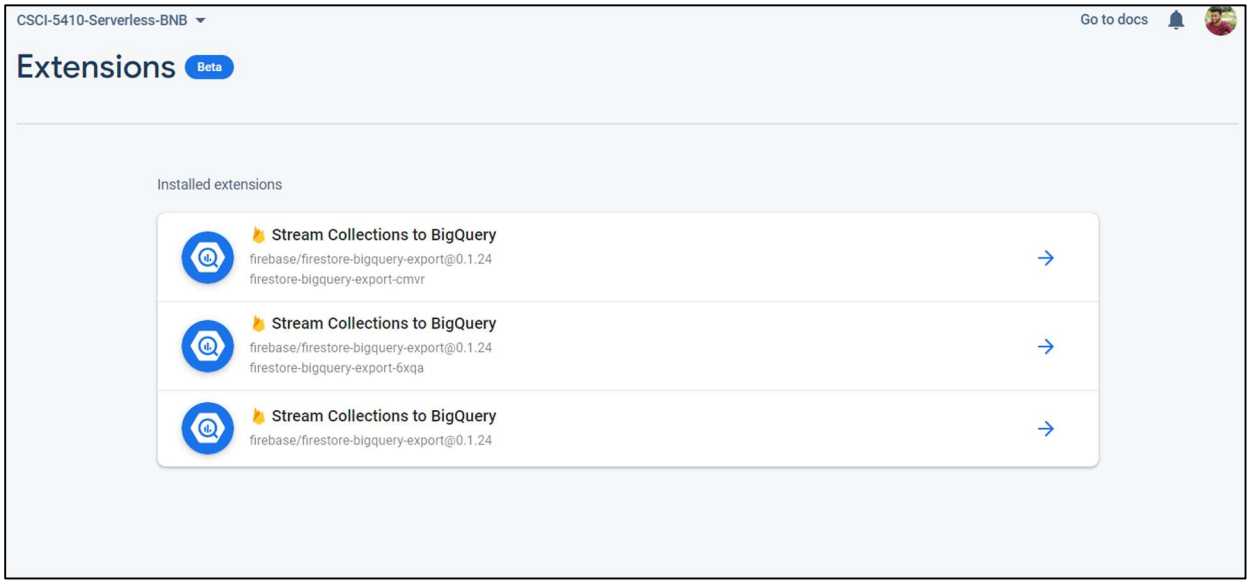


Figure 94 The firestore database is streamed to big query

Step 6: The saved query in big query creates table for room, kitchen, and user. The table is imported in google data studio to stream visuals.

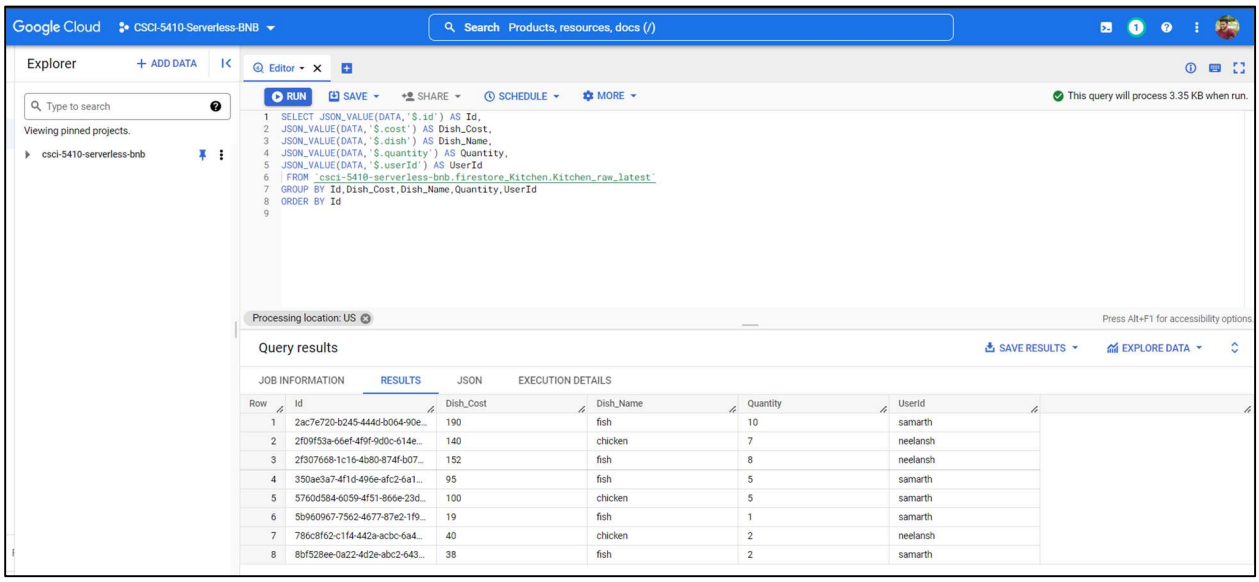


Figure 95 Query to create table view

Step 7: The visualization is performed in google data studio.

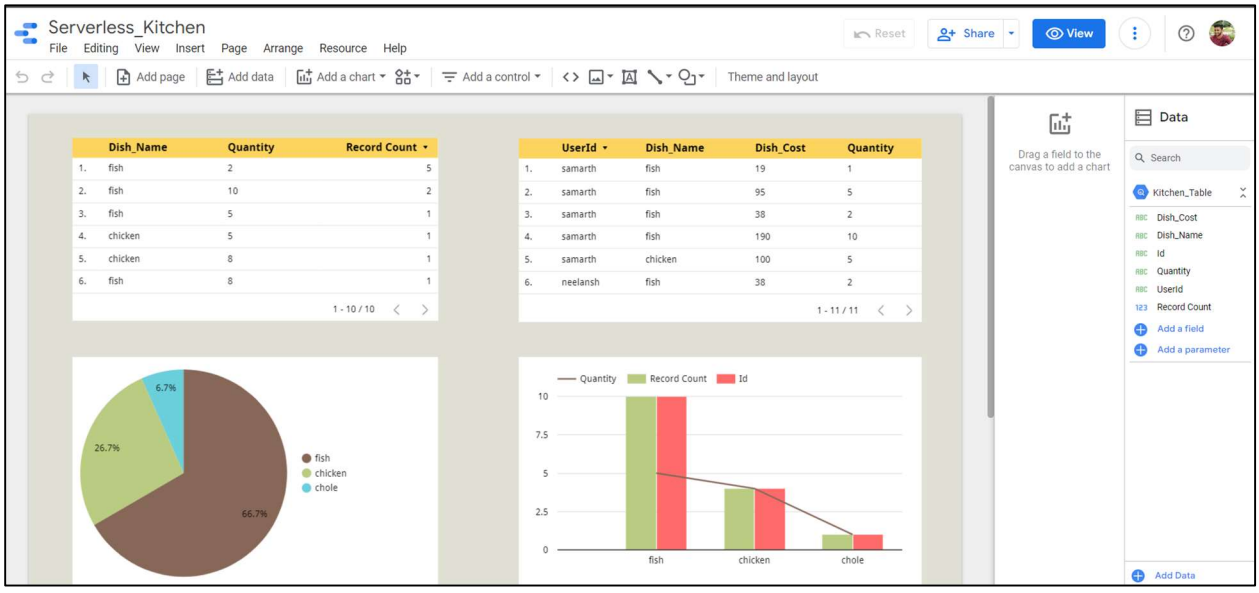


Figure 96 Table view imported in google data studio

Step 8: The visualization is embedded to the application. The order for 10 fish is now updated in kitchen analysis.

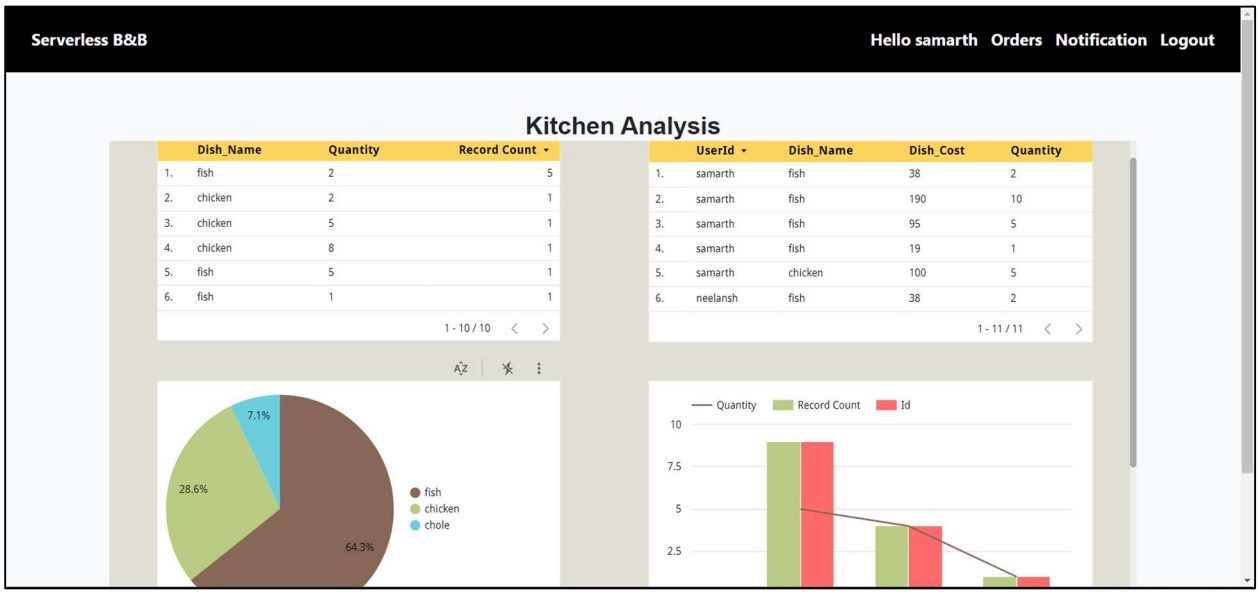


Figure 97 Visualization embedded to the application

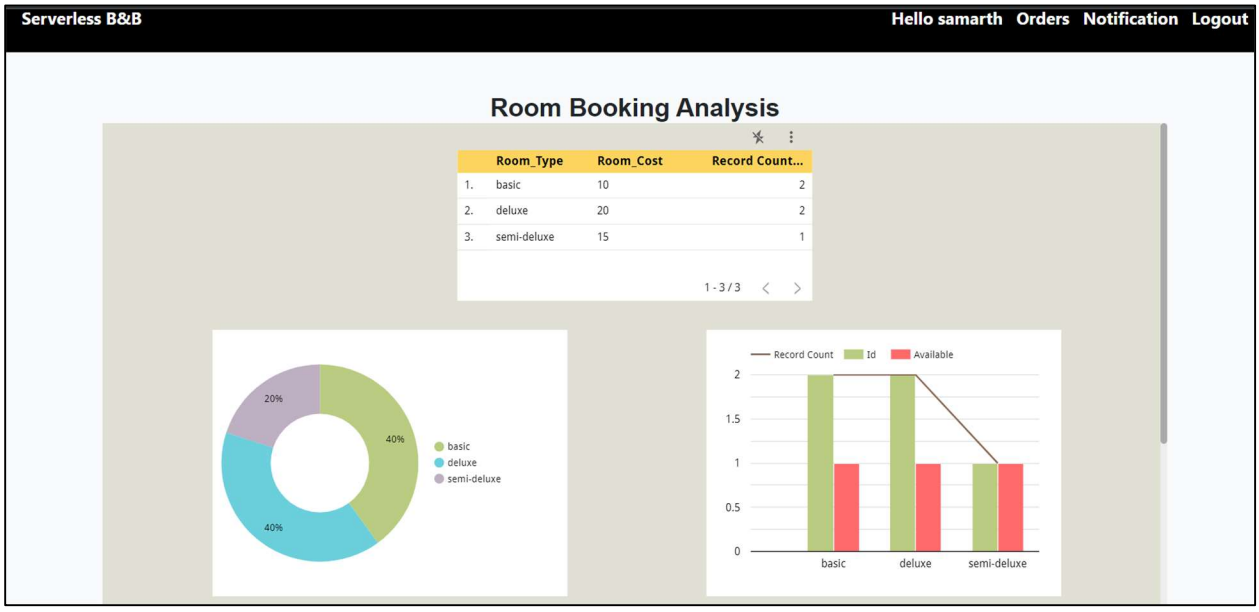


Figure 98 Room Booking Visualization

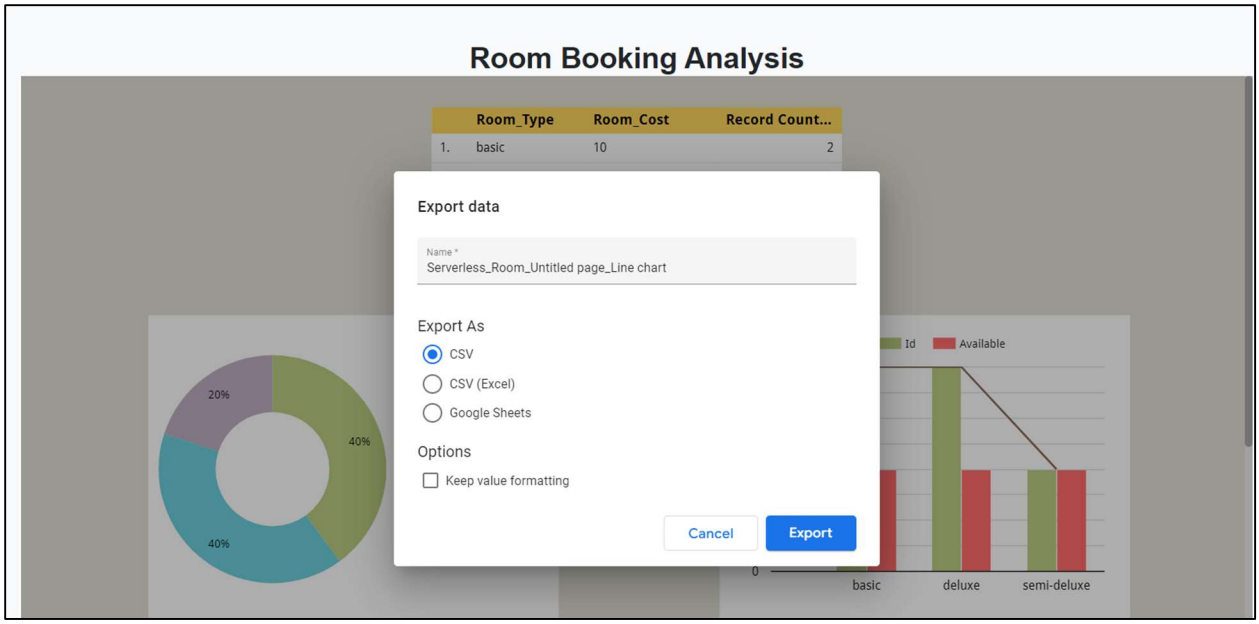


Figure 99 Graphs, charts etc. exported

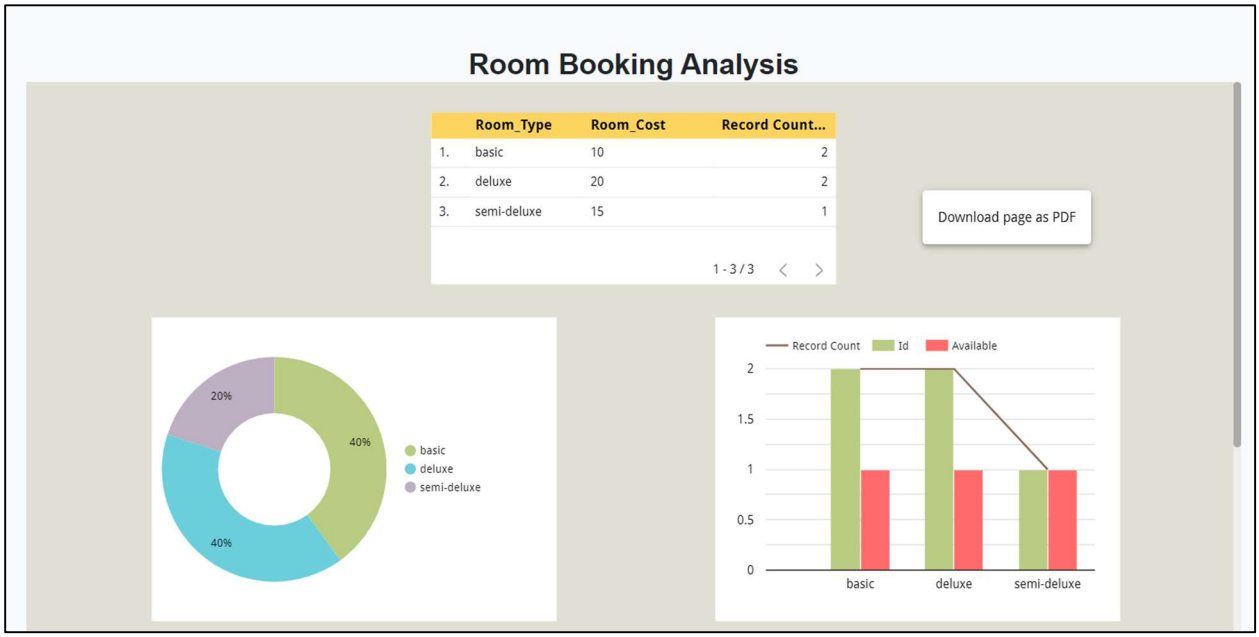


Figure 100 Analysis can be downloaded from application as PDF.

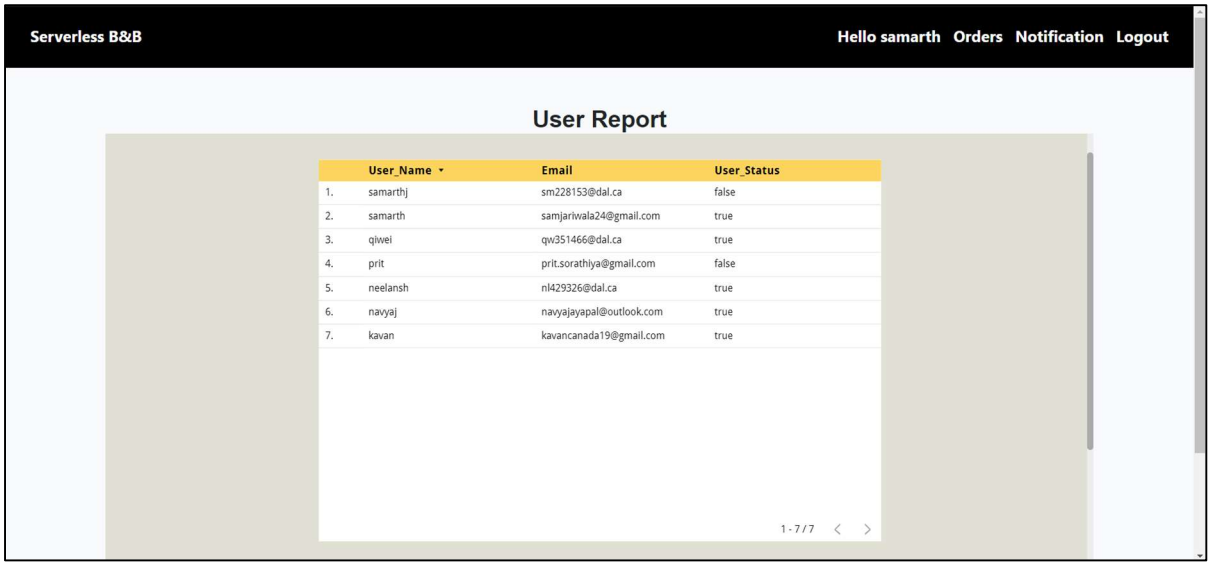


Figure 101 User Report

The data freshness if set 15 minutes, will automatically update the data every 15 minutes to your application. We can set any rate of data freshness in google studio as per our requirement.

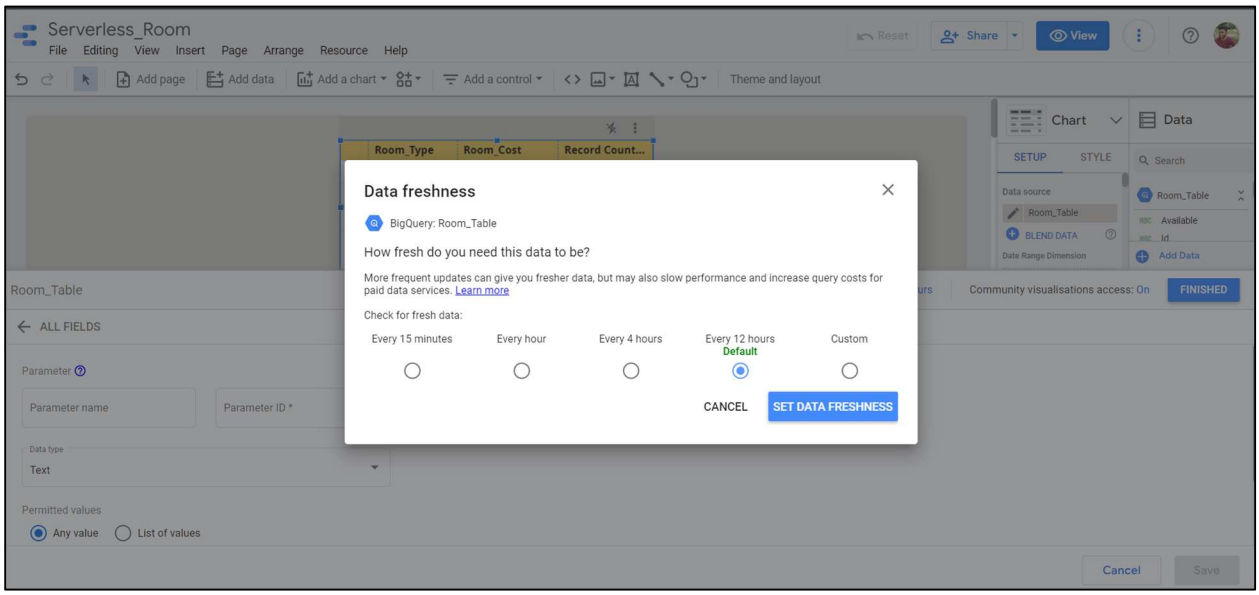


Figure 102 Set Data refreshness to set time to get updated data

Pseudo Code:

Step 1: User clicks on room analysis/ kitchen analysis/ user report.

Step 2: The data is fetched from each database of Dynamo DB through lambda function.

Step 3: The Firestore database connection is establish.

Step 4: The event record name of lambda is fetched.

Step 5: If event name is “REMOVE”. Go to step 5. Else go to step 8.

Step 6: Delete the user from the Room/Kitchen/User collection from firestore database.

Step 7: Go to step 9.

Step 8: Insert/Update the Room/Kitchen/User document from their collection in firestore database.

Step 9: Ending the function.

Step 10: The collections in firestore are streamed to big query.

Step 11: The query generates the Room/Kitchen/User table view.

Step 12: The view is imported in google data studio.

Step 13: Visualization is performed.

Step 14: The visualization is embedded to front-end where user can see Room Analysis/Kitchen Analysis/User Report.

Step 15: End of module.

4. Limitations

There are various constraints due to the architecture we utilize and how we built it. We utilize a synchronized pulling message for the message passing module maintaining a subscriber listening in a lambda function which eventually increases the cost. On the other side, it may lose messages since we limit the number of messages for each API call.

Another constraint is cost and security; we construct our application with a bigger number of API gateways. When the number of users increases, so does the number of requests. In addition, malicious users may attack our application via an API hack, wasting resources and money in the process. The managed AWS services will grow up and cost us considerably more as the number of API calls increases.

In terms of usability, consumers are not presented with UI for room and meal information, making it difficult for the user to determine what to order and which rooms to reserve.

5. Contribution

Table 1: Individual Contribution

Name	Role	Individual Contribution
Samarth Jariwala	Developer	Report Generation, Visualization.
Prit Sorathiya	Developer	User Authentication, Tour Management Module, Web Hosting
Qiwei Sun	Developer	Message passing module
Neelansh Gulati	Developer	Online Support and Web hosting
Kavan Patel	Developer	User Management, Tour Management Module, Web Hosting
Navya Jayapal	Developer	Machine learning Feedback

Table 2: Team contribution

Name	Role	Contribution
Samarth Jariwala	Developer	Respect others work and willing to make changes when needed
Prit Sorathiya	Developer	Help team member with technical issues
Qiwei Sun	Developer	Always approachable and responsible for tracking meeting
Neelansh Gulati	Developer	Active and approachable, deliver the tasks as promise
Kavan Patel	Developer	Organize team meeting and support team members
Navya Jayapal	Developer	Keep approachable and flexibility

6. Meeting Logs

Meeting Number: #1
Date of Meeting: May 12, 2022
Time: 10:00pm – 10:17pm
Place: Teams
Agenda: Meeting team members and schedule the team meetings
Discussion: Availability for the team meeting and team strength (experience and technology stacks)
Decision: Meet on Tuesday at 12:30 every week

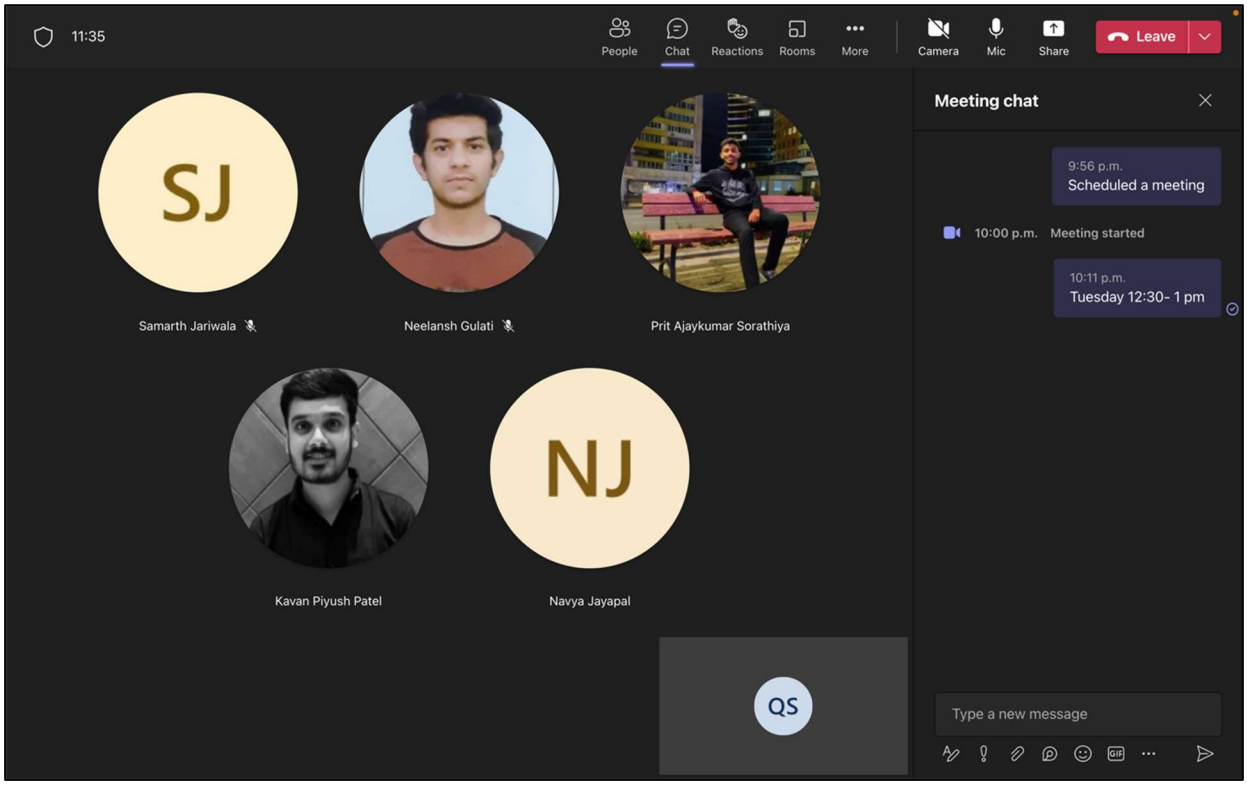


Figure 103 Meeting Log 1

Meeting Number: #2
Date of Meeting: May 17, 2022
Time: 9:00pm – 9:10pm
Place: Teams
Agenda: Pop-up meeting to discuss project
Discussion: understanding the project then distribute task for next meeting

Meeting Number: #3
Date of Meeting: May 25th, 2022
Time: 5:30pm – 7:30pm
Place: shift lab key 426
Agenda: overview about project architecture and project comments and web Tecnologies
Discussion:

Meeting Number: #4
Date of Meeting: May 30th, 2022
Time: 5:30pm – 6:40pm
Place: shift lab key room 426
Agenda: make a conceptual level understanding about the project and explore aws and gcp cloud services which we will use in our project.
Discussion:

Meeting Number: #5
Date of Meeting: June 27th, 2022
Time: 2:30pm – 3pm
Place: Teams meeting
Agenda:
Discussion:

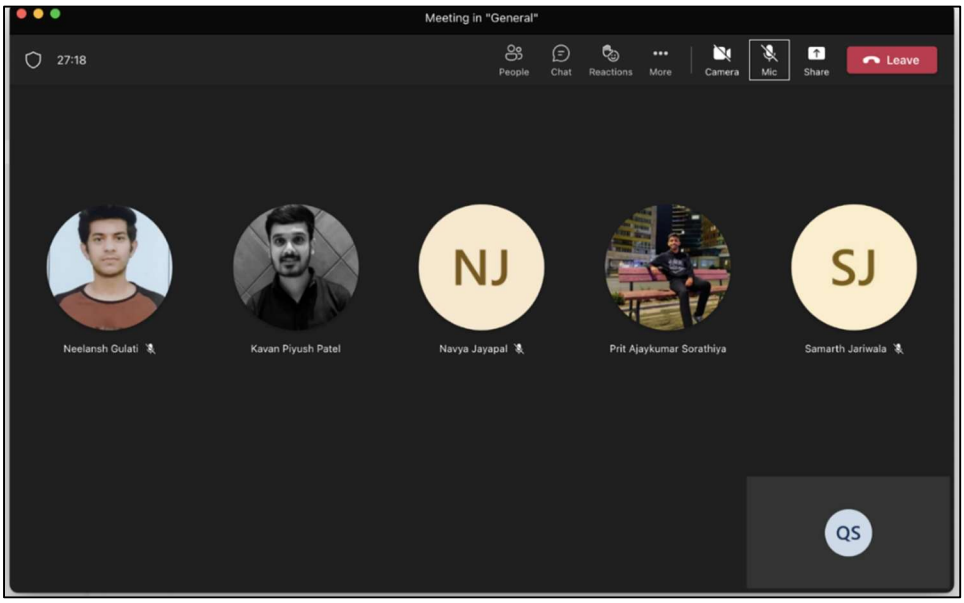


Figure 104 Meeting Log 5

Meeting Number: #6
Date of Meeting: June 28th, 2022
Time: 7:00pm – 7:30pm
Place: Teams meeting
Agenda: Doubts clarification with TA
Discussion:

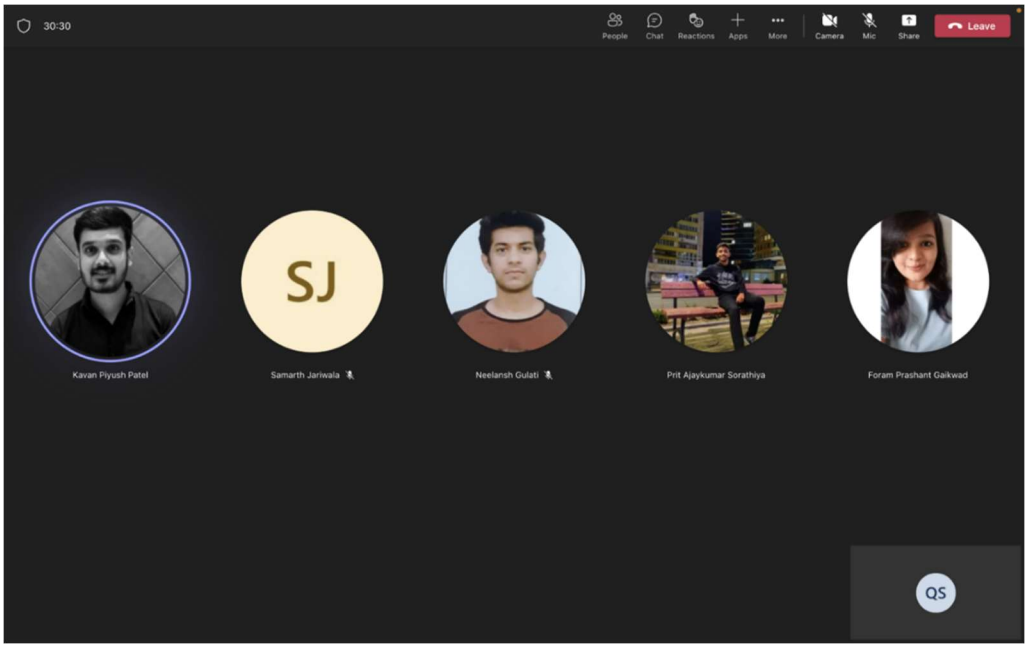


Figure 105 Meeting Log 6

Meeting Number: #7
Date of Meeting: July 2nd, 2022
Time: 2:00pm – 2:14pm
Place: Teams meeting
Agenda: project report discussion
Discussion:

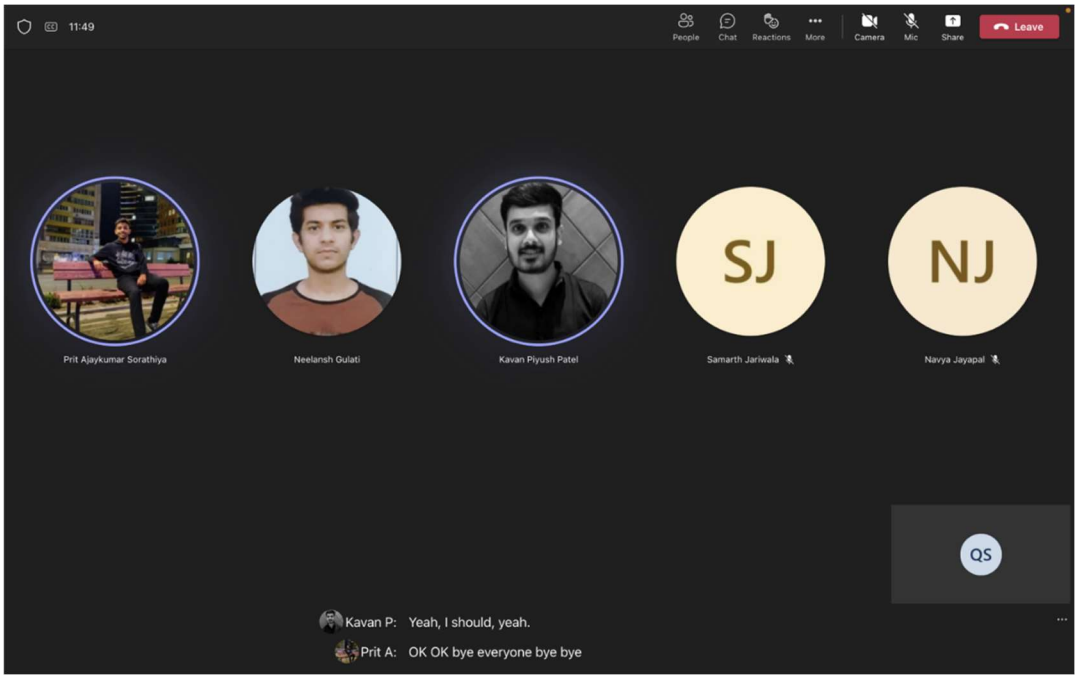


Figure 106 Meeting Log 7

Meeting Number: #8
Date of Meeting: July 2nd, 2022
Time: 11:00pm – 11:18pm
Place: Teams meeting
Agenda: project report discussion
Discussion:

Figure 107 Meeting Log 8

Meeting Number: #9
Date of Meeting: July 20st, 2022
Time: 11:00pm – 12:00pm
Place: Teams meeting
Agenda: Project demo and prepare for Q&A session
Discussion:

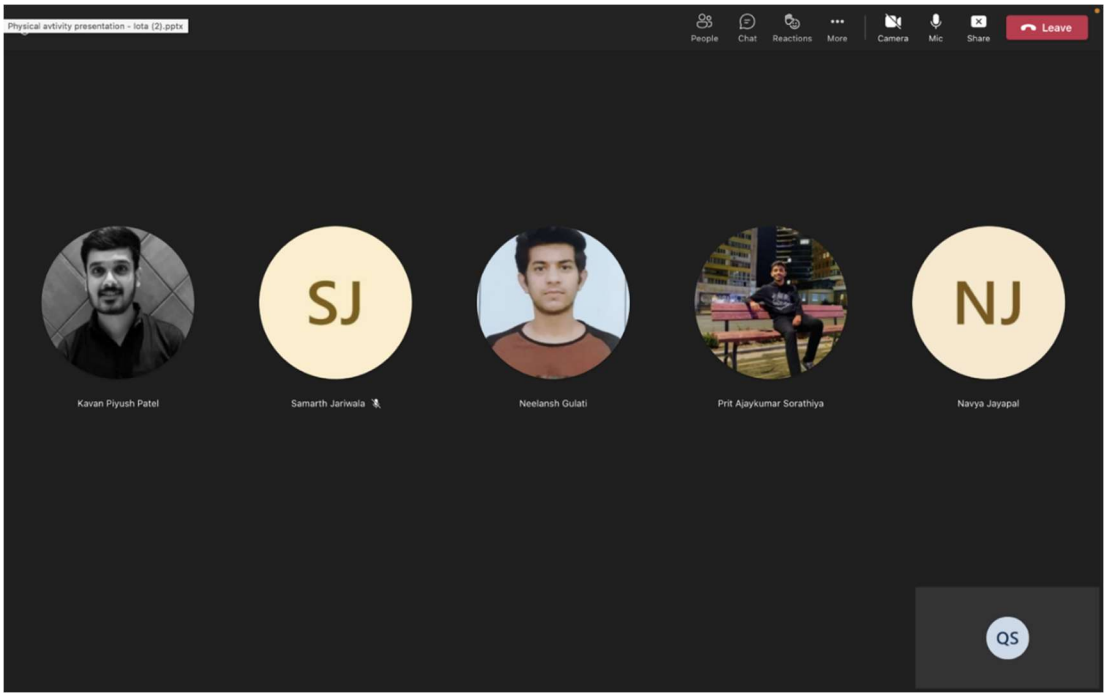


Figure 108 Meeting Log 9

Meeting Number: #10
Date of Meeting: July 21nd, 2022
Time: 11:00pm – 11:30pm
Place: Teams meeting
Agenda: Project integration and help to solve the problem

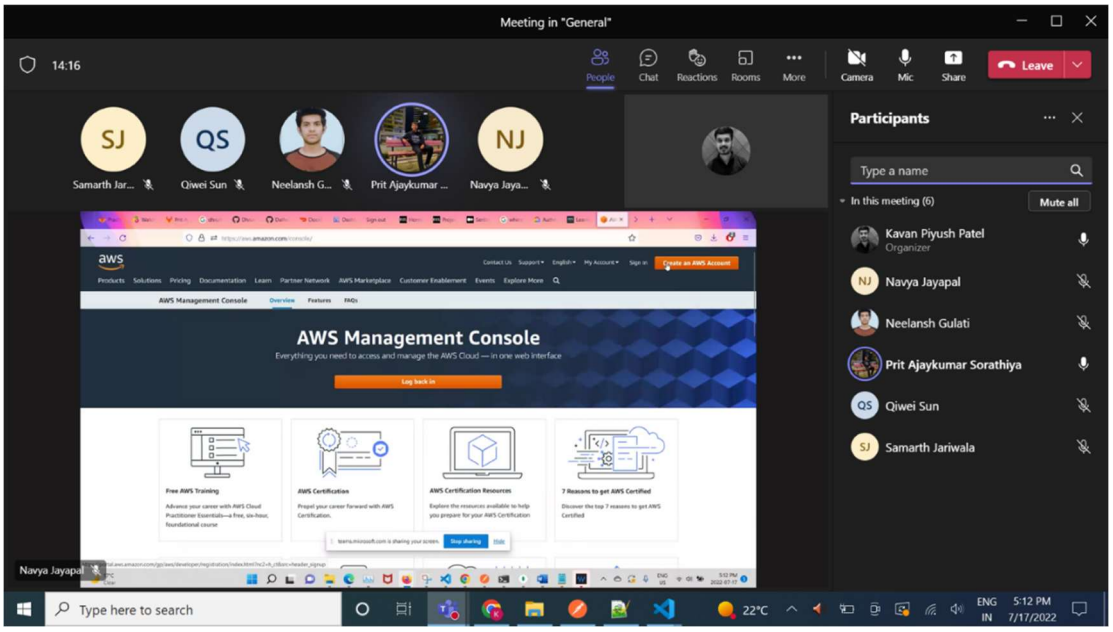


Figure 109 Meeting Log 10

Meeting Number: #11
Date of Meeting: July 22nd, 2022
Time: 11:00pm – 11:20pm
Place: Teams meeting
Agenda: Understanding the final project report and assign the task to each member

References:

[1] "Fast NoSQL Key-Value Database – Amazon DynamoDB – Amazon Web Services", *Amazon Web Services, Inc.*, 2022. [Online]. Available: https://aws.amazon.com/dynamodb/. [Accessed: 30- May- 2022].

[2] "Amazon Cognito - Simple and Secure User Sign Up & Sign In | Amazon Web Services (AWS)", *Amazon Web Services, Inc.*, 2022. [Online]. Available: https://aws.amazon.com/cognito/. [Accessed: 30- May- 2022].

[3] "Firestore: NoSQL document database | Google Cloud", *Google Cloud*, 2022. [Online]. Available: https://cloud.google.com/firestore#section-5. [Accessed: 31- May- 2022].

[4] "Cloud Functions | Google Cloud", *Google Cloud*, 2022. [Online]. Available: https://cloud.google.com/functions. [Accessed: 31- May- 2022].

[5] "What is Pub/Sub? | Cloud Pub/Sub Documentation | Google Cloud", *Google Cloud*, 2022. [Online]. Available: https://cloud.google.com/pubsub/docs/overview. [Accessed: 31- May- 2022].

[6] blazeclan. *5 key benefits of Amazon S3*. [online] Available at: https://www.blazeclan.com/blog/5-key-benefits-of-amazon-s3/ [Accessed 2 June 2022].

[7] *5 Use cases for DynamoDB.* [online] Available at: https://rockset.com/blog/5-use-cases-for-dynamodb/ [Accessed 2 June 2022].

[8] McGloin, C., *How to Overcome Intent Limitations on Amazon Lex and other NLP Engines*. [online] ServisBOT AI. Available at: https://servisbot.com/nlp-intent-limitations/ [Accessed 2 June 2022].

[9] *What is Amazon Lex?* [online] Available at: https://docs.aws.amazon.com/lex/latest/dg/what-is.html [Accessed 2 June 2022].

[10] Deshpande, A., *7 Advantages of ReactJS for Building Interactive User Interfaces*. [online] Clariontech.com. Available at: https://www.clariontech.com/blog/7-advantages-of-reactjs-for-building-interactive-user-interfaces [Accessed 2 June 2022].

[11] Amazon Web Services, Inc. *AWS Elastic Beanstalk – Deploy Web Applications*.

[online] Available at: https://aws.amazon.com/elasticbeanstalk/ [Accessed 2 June 2022].

[12] Amazon Web Services, Inc. *Amazon QuickSight Features - Business Intelligence Service - Amazon Web Services*. [online] Available at: https://aws.amazon.com/quicksight/features/ [Accessed 2 June 2022].

[13] MUO. *The 8 Best Features of Google Data Studio for Data Analysis and Visualization*. [online] Available at: https://www.makeuseof.com/best-features-google-data-studio-data-analysis-visualization/ [Accessed 2 June 2022].

[14] "How to visualize data with Google Data studio," *Content Marketing Institute*. [Online]. Available: https://contentmarketinginstitute.com/2020/03/google-data-studio-visual-content/ [Accessed: 05-Jul-2022].

[15] "Cloud natural language; google cloud," Google. [Online]. Available: https://cloud.google.com/natural-language [Accessed: 24-Jul-2022].