FeedMe: A Cloud-Based Restaurant Management System

Developed by
Vamsi Gadiraju (VG2383)
Vishruth Nair (VN2286)
Harsh Yadav (HY2616)

TABLE OF CONTENTS

Abstract:

As restaurants increasingly move towards prioritizing delivery across delivery services, there is a growing need to effectively facilitate customer engagement. Restaurant SEO is heavily based upon review count and overall reviews. Unfortunately, each delivery service has its own app for customer engagement and none of them heavily focus on guiding restaurant goers to leave reviews. This would require using 5 logins for 1 restaurant to handle customer engagement for a single branch, 10 logins for 2 branches and so on. There are three goals of FeedMe: 1) centralize user engagement through the development of aggregated feeds across all delivery platforms and for all branches of a single restaurant, 2) spur increased review numbers through sending restaurant goers automated messages with tailored links associated with the review sites alongside a nice message, and 3) create a web-based chatting service that can communicate with restaurant goers through text messages (we also hope to allow for group chats to allow for mass marketing messaging for delivery coupons or new menu updates). We've created the initial version of the application using our cloud architecture and are excited to share it with you.

Introduction:

With the food delivery market expected to grow to $365 billion globally by 2030, restaurateurs—be they franchises or individuals—have been eagerly searching for ways to capitalize on the delivery boom, both because it is operationally prudent, but also because customers are demanding it. Third-party delivery platforms, like Uber Eats and DoorDash, have accelerated this growth, so much so that an entirely novel type of restaurant has been increasingly popping up in major cities—delivery-only/virtual restaurants and cloud kitchens. For restaurants, SEO optimization on Google and within these apps are crucial as this determines their positioning on listings for users and users are much more likely to choose restaurants with more reviews and higher on listings.
The goal is to design an interaction management system for these restaurants to allow them to effectively gauge feedback/reviews from consumers and unique drive brand development. This approach has been extremely successful for small to medium businesses (SMBs) and the same can be replicated for delivery restaurants. Once a restaurant gets an email or text indicating that an order has been placed and has been fulfilled, this data will be scraped to generate a text message or email that will be sent to thank the user and offer them a link to a page that will allow them to:

- Leave unique feedback (or menu recommendations) for the restaurant through text messaging the restaurant
- Leave reviews on Google Reviews, Yelp, or Open Table

- (in the future) Talk to a chatbot about their restaurant experience (how the customer's experience can be improved? Any difficulties faced during the process? etc).

This page will have an easy to use interface to guide individuals through each of these workflows.

From the restaurant's perspective, they will have three interfaces that can help them understand their customer engagement:

1. A user dashboard that shows the number of overall reviews, stars, and growth metrics for engagement.
2. An engagement flow that shows the responses left by each user. When a specific user is clicked, the restaurant can see the message left, any conversational information, and can also text/email the user back to discuss the experience overall.
3. A feedback page showing the unique feedback and menu recommendations left to the user.

Restaurants can also offer rewards to users who leave reviews through this mechanism. This tool is incredibly useful for restaurants that deliver through multiple locations and multiple delivery services as it decreases the amount of research work needed to drive this feedback and respond to it. This application would be a one-stop shop for all customer engagement for restaurants.

External Links:

Youtube Video: https://youtu.be/-HEBxCQ29pI

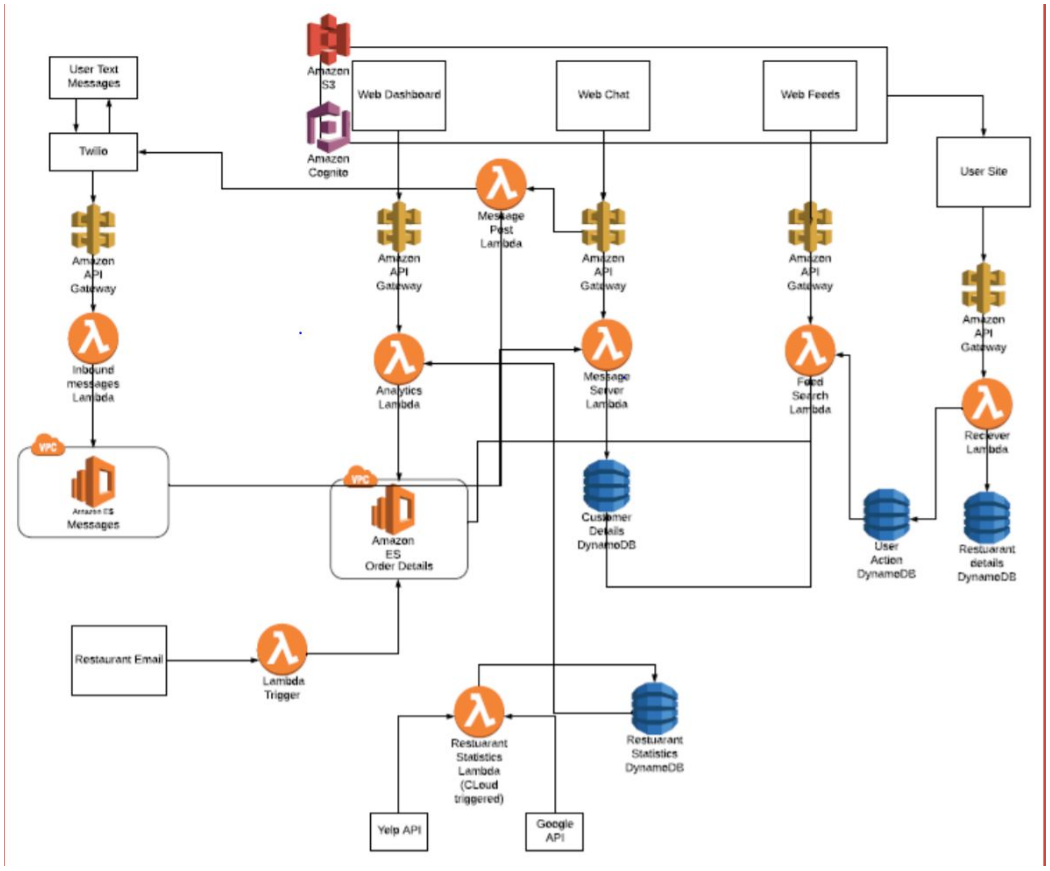
Presentation Link and Demo Links:
https://drive.google.com/file/d/1pXfmYND_rZcHwZipd875rPzOs6nxPVEF/view?usp=sharing

Code: Code was included in this submission as a zip file with the lambda code and front-end code all included.

Architecture:

From an architecture perspective, the entire web application is built using a very AWS-heavy stack. As the main intention of this application required high scalability and low latency (to fulfill real-time engagement and messaging), it is highly dependent on serverless products like Lambda, provided by AWS. As mentioned earlier, from the restaurant's perspective, there are three key perspectives: the analytics dashboard, the engagement feeds, and the messaging functionality. These 3 interfaces are hosted as static pages on S3 behind Cognito for User Authentication/Security. We use a Swagger-based API gateway to interface with our microservice-focused serverless architecture (as can be seen in the diagram below). Our

Lambdas extensively use DynamoDB or ElasticSearch to facilitate our key functionalities. We used ElasticSearch mainly for chat storage, search optimization, analytics numbers and other functions that were very search heavy. We used Dynamo for more traditional storage. Aside from these views, we had key functionalities driven by lambda for analytics, chatting, and order onboarding. We used SES to receive inbound emails to registered email accounts for our theoretical restaurants. On an email receipt, the data is written to SQS from which a Lambda will be triggered to scrap the data, extract the key information, and write it to the pertinent databases. Additionally, a text message with a unique order-based link including links for leaving reviews will be sent out. We also used CloudWatch to configure weekly-run lambdas that scrape review analytics on our restaurants from Google, OpenTable, and Yelp and write them to our analytics tables (this lambda is also used to run aggregations on our own data for further analytics on order types, popular delivery services, etc). All of our texting services are driven through Twilio. For outbound messages, we use the simple SDK. For inbound messages, we have a Lambda-based webhook that appropriately scrapes the message metadata, assigns it to the proper reviewer and restaurant, and proceeds to handle the data properly (write to ElasticSearch, SQS, etc). We use SQS extensively to populate new user notifications based on new engagements/inbound messages to allow for easy user understanding.



Implementation and User Flows:

1. The user logs into the application using the Cognito authentication framework.
2. Upon logging in, the application pulls data from the DynamoDB tables concerning the restaurant with its user id. This is loaded into the dashboard page that displays the analytics. The dashboard analytics include number of reviews over time, as well as the performance of the restaurant based on delivery services. All this metadata is serviced through API gateway GET calls (lambda behind the Gateway) and ElasticSearch. As mentioned earlier, the analytics data is computed by a weekly run lambda.
3. Another view in the application fetches the user engagement data which includes the order id, customer name, delivery service, order date, and restaurant name/branch .
4. This component also presents the capability to the restaurant to filter reviews by delivery service, customer name, or order date.
5. Each review is an individual component that opens to a screen displaying all the information regarding the order and the corresponding review.
6. Another key feature of the system is to allow users to chat with the reviewer on a device agnostic platform. To that end we built a chat system that displays the chat history to the restaurant and allows sending texts/emails to the reviewer.

Lambdas in Use:

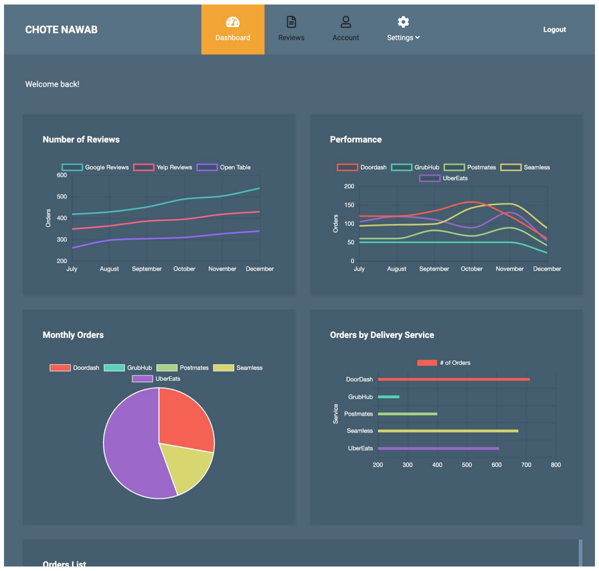
- Text Message Webhook
- Review aggregator
- Chat handler
- Dashboard Formatter
- Order Creator
- Message Link handler
- Analytics Processor
- Review Handler
- 7 API Gateway related-lambdas

API Gateway Design:

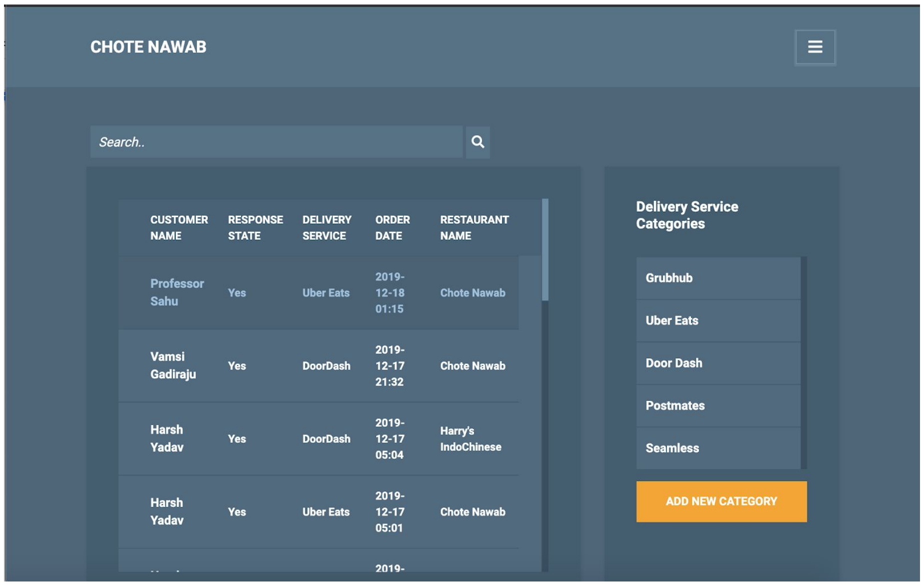
1. Dashboard retrieval /dashboard (GET)
    a. Retrieve all data analytics pertaining to restaurants reviews, provide most recent notifications, and give unread messages
    b. Validate token that is sent as query parameter)
2. Review Feed retrieval /reviews (GET)
    a. Retrieve all engagements in proper ordering and under proper format (opened vs unopened reviews)
    b. Provide general information on each engagement to display and link to individual chat pages or group chat pages
    c. Handle search for specific reviews through query parameter
    d. Validate restaurant token that is sent as a query parameter

3. Individual review retrieval /review/ (GET, POST)
    a. Retrieve individual engagement data (previous messages, user link metadata, user click data (did they click on our link, what review app did they click to handle) and all customer data pertaining to individual engagement
    b. Verify restaurant through token in query parameter
    c. Send outbound message by text to specific user through POST request with token validation
4. User data web page /reviewer (GET)
    a. Retrieve information pertaining to a user order and gather proper links pertaining to restaurant's review sites (Google Reviews Link, Yelp Link, and OpenTable link)

General Implementation Images:



Initial Dashboard w/ several analytics and order list


CHOTE NAWAB

Search..

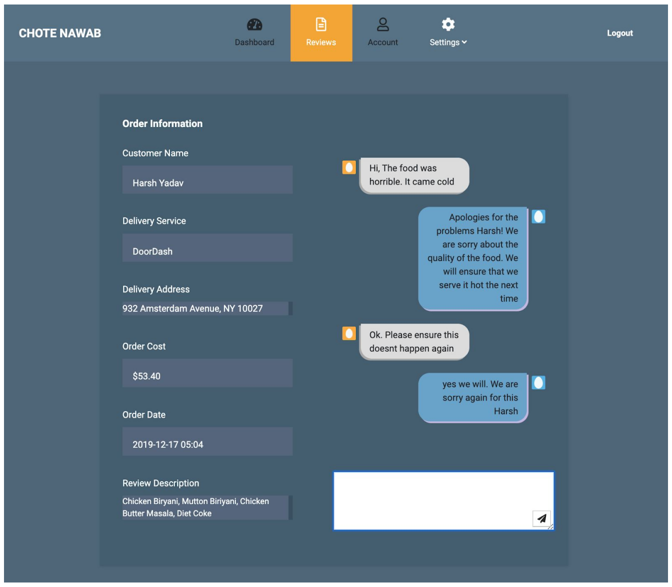
CUSTOMER NAME	RESPONSE STATE	DELIVERY SERVICE	ORDER DATE	RESTAURANT NAME
Professor Sahu	Yes	Uber Eats	2019-12-18 01:15	Chote Nawab
Vamsi Gadiraju	Yes	DoorDash	2019-12-17 21:32	Chote Nawab
Harsh Yadav	Yes	DoorDash	2019-12-17 05:04	Harry's IndoChinese
Harsh Yadav	Yes	Uber Eats	2019-12-17 05:01	Chote Nawab
			2019-	

Delivery Service Categories

- Grubhub
- Uber Eats
- Door Dash
- Postmates
- Seamless

ADD NEW CATEGORY

Engagement Feed w/ general information, search, and categories search


CHOTE NAWAB

Order Information

Customer Name: Harsh Yadav

Delivery Service: DoorDash

Delivery Address: 932 Amsterdam Avenue, NY 10027

Order Cost: $53.40

Order Date: 2019-12-17 05:04

Review Description: Chicken Biryani, Mutton Biriyani, Chicken Butter Masala, Diet Coke

Hi, The food was horrible. It came cold

Apologies for the problems Harsh! We are sorry about the quality of the food. We will ensure that we serve it hot the next time

Ok. Please ensure this doesnt happen again

yes we will. We are sorry again for this Harsh

Chatting with a user through text on our web-UI

Results

Our resulting product is a complete restaurant engagement management system. It is built to scale through our cloud architecture and heavily optimizes for scalability and low latency. It includes an analytics aggregator, inbound email processor, messaging engine built on top of Twilio and ElasticSearch, notifications platform, message searcher, and custom link generator for unique customer webpages.

When we designed the application, we heavily built in metadata generation to effectively store information on how both our users used the app and how restaurant goers engaged with our links (we store data on whether they clicked the initial link, whether they clicked any review links, which review links they clicked, and whether they used text messaging with the restaurant). This data combined with our order data, analytics data, location data, and messaging data makes us an extremely strong processor for future work. Through having this data, we can work to develop marketing strategies for restaurants, provide additional information on engagement type (assign messages neutrality scores), give restaurants insights on what menu items perform the best, and also help them understand how their users actually feel (who felt the need to leave a review based on a specific menu item; we have more meaningful data than just who ordered what). We can also implement AI customer chat bots to further improve the engagement experience.

Moving forward, we plan to use a combination of KPIs to further evaluate our overall system. Firstly, we hope to use the Yelp, Google Reviews, OpenTable APIs and our internally collected metrics to get real time data to gauge several key metrics: 1) how many people are actually leaving reviews based on our initial messages, 2) how many people are leaving recommendations, 3) are restaurant reviews being helped overall, 4) how quickly are restaurants responding to user reviews, and 5) how is SEO being impacted overall. Other key evaluation metrics that could be useful include: 1) daily active restaurant users, 2) monthly active restaurant users, 3) clickstream patterns of our users (to assess whether processes and UI is streamlined.

Conclusion

FeedMe represents a strong start to tackling the huge issue of restaurant engagement management. As virtual kitchens continue to rise in popularity and food delivery grows further, restaurants will have an implicit need for this type of system and this project represents a strong MVP towards that goal. Our heavy use of cloud technologies has given us three key benefits: 1) use of existing, robust technology, 2) modularizability of project components, and 3) high scalability with low overhead. Starting with technology, we built our application on top of extremely strong underlying resources. This gives us high scalability, availability, and function out of the box with little overhead or maintenance. Through using these cloud technologies and serverless architecture, we were able to use a CI/CD approach to development with no real downtime given our ability to release changes on the fly. Additionally, the auto-scaling of our serverless resources really reduced the burden of thought associated with release cycles and release projections (how do we scale out a release). Finally, we used extremely scalable resources that allow us to add additional resources in an extremely simple fashion. This means our product is truly scalable without us having to implement complicated logic ourselves. We hope to be able to continue work on this project moving forward and are excited by the prospect of applying more of the work that Professor Sahu taught us over the course of this semester.