CSCI 5410 – Serverless Data Processing
Project Design Document
Project: Serverless B&B

Course Instructor: Dr. Saurabh Dey

Submitted by: Group 29

Group Member Name	Banner ID
Aditya Deepak Mahale	B00867619
Jayasree Kulothungan	B00894354
Rishika Bajaj	B00902713
Sai Chand Kolloju	B00897214
Sourav Malik	B00839958
Udit Gandhi	B00889579

Table of Contents

Serverless B&B Application Architecture

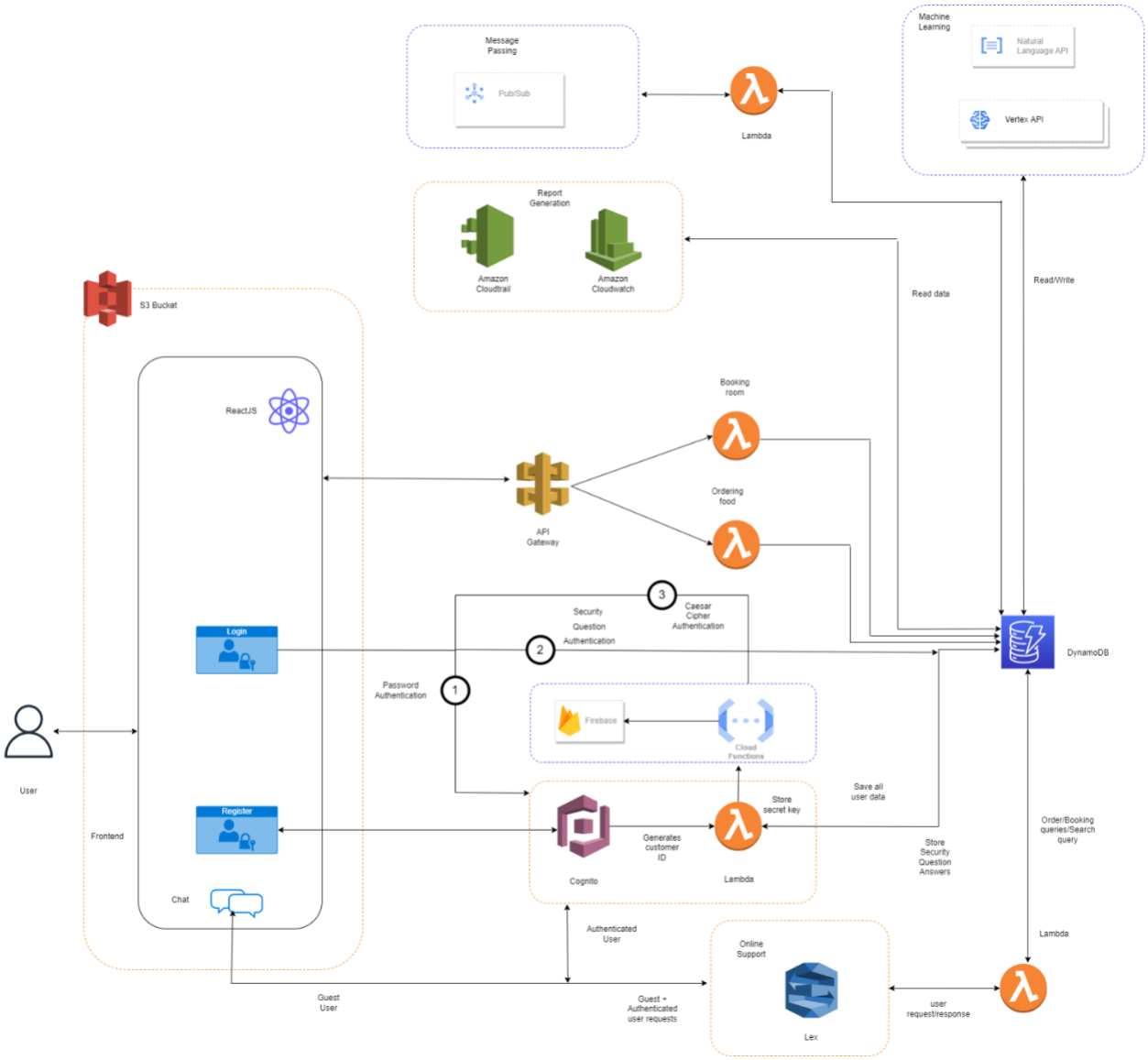

Figure 1: Architecture Diagram built using draw.io [1]

Serverless B&B Application Architecture Overview

Serverless B&B follows the multi-cloud architectural model that employs cloud services provided by Amazon Web Services (AWS) and Google Cloud Platform (GCP) cloud providers (See Figure 1). This architectural model adds to the high availability of our application. The application is deployed to an S3 bucket [2] on AWS that hosts the static assets produced by the React frontend of our application. This allows the users to access the application over the internet. The application and user data shall be stored on Cloud Firestore [3] and Amazon DynamoDB [4] that supports user registration and multi-factor user authentication to name a

few. Users are authenticated and managed using Amazon Cognito [5] backed by the DynamoDB and Firestore databases. The application's virtual assistance system is managed by Amazon Lex [6] which makes use of AWS Lambda [7] functions to handle user requests utilizing the data on DynamoDB. AWS Lambda Functions and Google Cloud Functions [8] handle most of the logical processing in the application including booking rooms and ordering food to name a few replacing the traditional server-oriented architecture. This processing is supported by the Google Cloud Pub/Sub [9] service that enables communication and simultaneous processing of requests across the application. These cloud services are exposed securely to the application through API Gateway service provided by both GCP and AWS cloud providers. The application uses machine learning algorithms provided by GCP, more specifically, the Vertex AI [10] and NLP API [11] services to build personalized tour packages and analyse customer feedback respectively. Amazon CloudTrail [12] and Cloudwatch [13] cloud services are used to generate reports pertaining to user login statistics.

Modules with their decided services

The application will be developed using the below mentioned AWS and GCP services:

Module	Services
User Management	AWS: Cognito, Lambda, DynamoDB GCP: Cloud Functions, Firestore
Authentication	AWS: Cognito, DynamoDB, Lambda, GCP: Cloud Functions, Firestore
Online Support	AWS: S3, Lex, Lambda, DynamoDB
Message Passing	GCP: Cloud functions, Pub/Sub, Firestore
Machine Learning	GCP: NLP API, Vertex AI
Web Application Building & Hosting	React and GitLab CI/CD AWS: S3
Other Essential Modules: 1. Report Generation Module 2. Visualization	AWS: CloudTrail, CloudWatch Google Data Studio

Roadmap

User Registration

Unregistered users can sign up by filing the signup forms on the registering page.

1. The user information is collected and stored in Amazon Cognito [5].

2. The user information is validated through preSignUp triggers set up using Amazon Lambda.
3. Amazon Lambda is also used to create a customer id.
4. After the information is validated, all the data is stored in Amazon DynamoDB [4] and the secret key is stored in Google Firestore [3] to perform authentication.
5. Amazon Lambda is used to store the details in DynamoDB, and Google Cloud Function [8] is used to store in Google Firestore [3].

Figure 2 shows the user registration flow diagram.

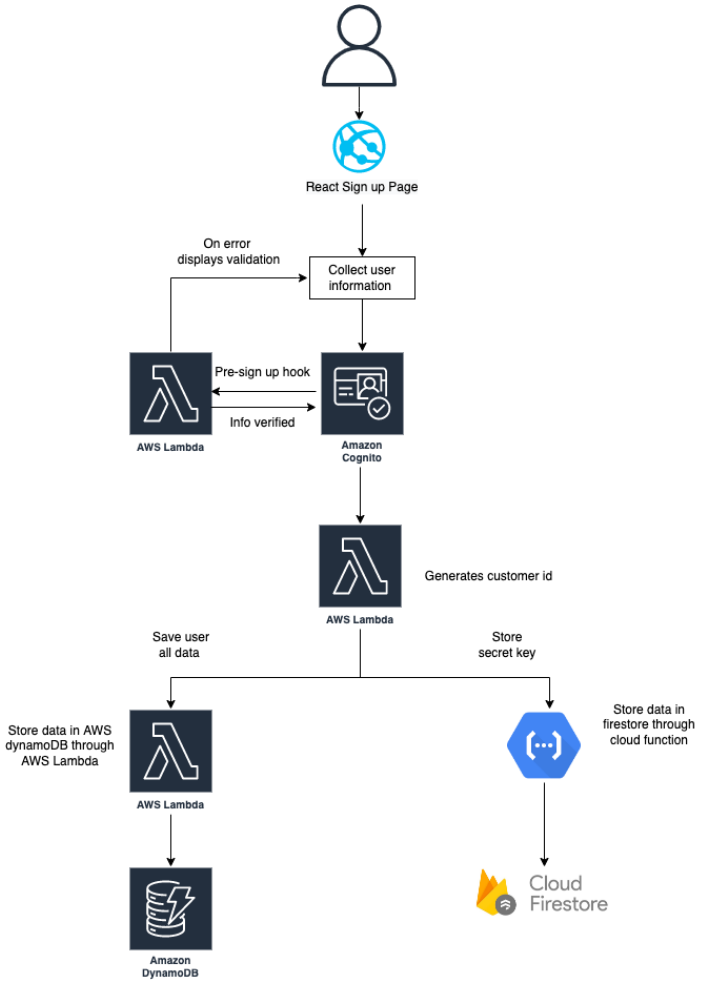

*Figure 2: User registration flow d*iagram built using draw.io [1]

User Login

Registered users are authenticated by using a 3-factor authentication which includes:

1. ID-password: Users will be provided with a screen to input userID and password. Once entered, the credentials will be validated using AWS Cognito [5].
2. Security Questions: On successful authentication of ID-password, next step is authenticating user by verifying their answers to the security questions which were provided during the time of registration. For this, we are using AWS Lambda [7] and Amazon DynamoDB [4] services.
3. Caesar Cipher: Once the above two steps are completed successfully, user will be provided with a string which needs to be encrypted using the secret key that they shared with the system during the time of registration. The system will perform the same encryption from its end and the generated cipher from both sides will be matched for the final step of authentication. For this, we are using Google Cloud Functions [8] and Cloud Firestore [3].

Figure 3 shows the user login flow diagram.

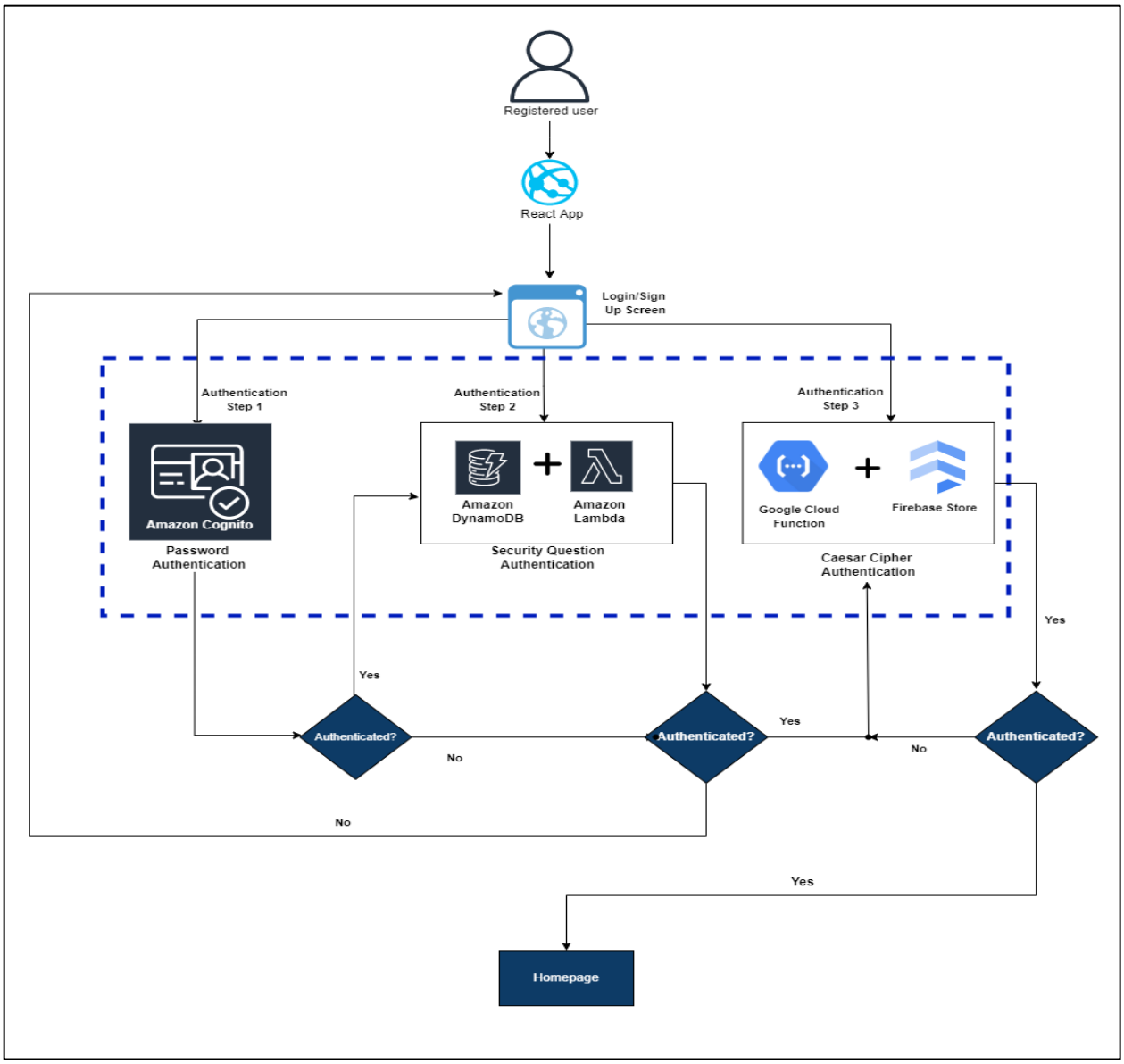

Figure 3: User login flow diagram built using draw.io [1]

Ordering food through kitchen service

Customer who has logged in, can order the food from the restaurant. These are the steps for processing the food order.

1. Logged in person click on placing the food order.
2. Google API gateway food order URL is invoked.
3. GCP cloud function runs to validate the order.
4. Once the order is validated, message is passed to the pub sub.
5. If the queue is full, message waits to be processed by another cloud function.
6. Order gets updated inside the Firestore [3], i.e., updates the inventory.

7. Once the update in Firestore [3] is done, invoice is sent to the hotel management by a cloud function.

Figure 4 shows the order food flow diagram.

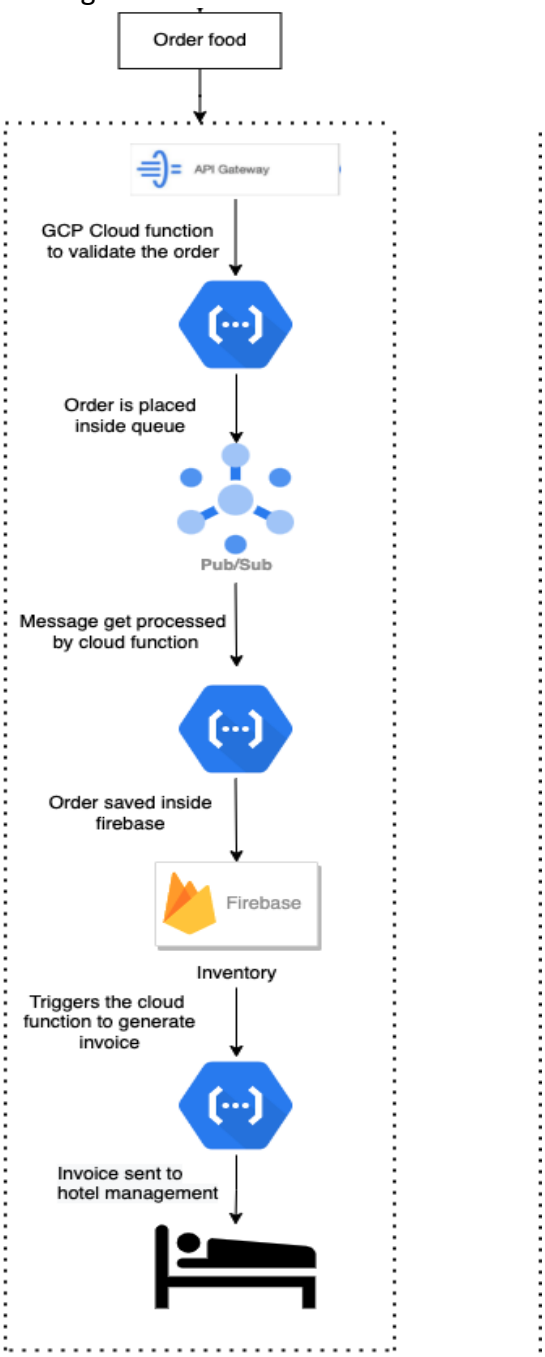

Figure 4: Order food flow diagram built using draw.io [1]

Tour Operator service

The tour operator service is responsible for taking in tour requests from registered customers and propose a tour package that best suits the requirements requested by the user. Following are the sequence of actions that would take place in the tour operator service:

1. Authorized users would be able to request a tour package by specifying their required length of the tour from the user interface for the project.
2. The request submitted is placed in a Pub/Sub [9] queue which is constantly watched by a cloud function. The Pub/Sub queue is secured using API gateway which exposes the service only to authorized users.
3. The cloud function processes the tour requests from the Pub/Sub [9] queue and creates ideal tour package for the user.
4. The cloud function makes use of the machine learning algorithms provided by GCP to build the tour package.
5. The GCP machine learning algorithms are trained to build the tour package by identifying similarities in the stay durations with the data stored on Firestore [3].
6. Once the tour package is built, the cloud function makes use of the Firebase Cloud Messaging [14] service to send out an email to the customer that requested for a tour package with the tour package details. Firebase Messaging Service shall use SendGrid [15] as its email client to send out emails.

Figure 5 shows the flow diagram for tour operator service.

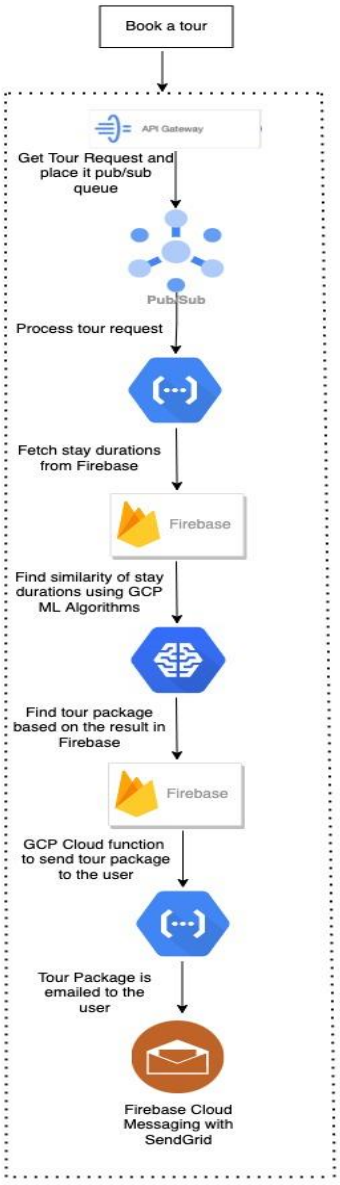

Figure 5: Tour operator service flow diagram built using draw.io [1]

Customer Feedback and Feedback Analysis

Authorized customers are allowed to give feedback on the hotel. The machine learning algorithms used by this service are provided by Vertex AI [10] and NLP API [11] services on GCP. Following are the sequence of actions that would take place in the feedback process:

1. Authorized users would be able to submit feedback on the hotel from the user interface of the project.
2. The submitted feedback is received by a cloud function which stores it in the Firestore database.
3. The machine learning algorithms provided by GCP are used to calculate the polarity of the feedback and assign a score to it.

4. The cloud function sends the feedback with the polarity back to the React frontend to display the feedback to the users.

Figure 6 shows the customer feedback and feedback analysis flow diagram

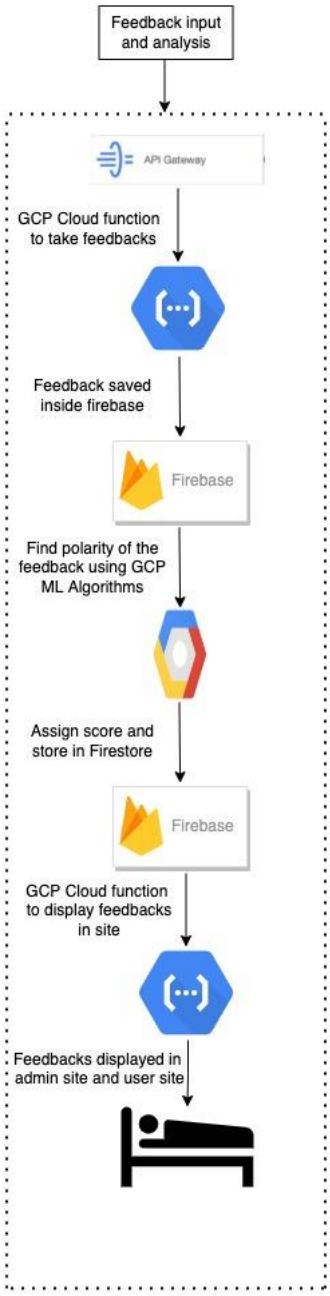

Figure 6: Customer feedback and feedback analysis flow diagram built using draw.io [1]

Searching Rooms

When an unregistered user accesses the website, they can search for the available room on the homepage.

1. The React app deployed on S3 bucket [2] sends the static content to the user's browser.
2. The website displays all the available rooms on the home page by calling the GET method for the /rooms endpoint.
3. The request gets passed to the API Gateway.
4. The API gateway passes this request to the lambda function for processing this information.
5. The lambda function cleans and processes this information and then queries the DynamoDB database.
6. DynamoDB [4] returns the list of available rooms to the lambda function.
7. The lambda function returns this information to the API gateway.
8. The API gateway sends the response back to the React frontend.

Booking Room

When registered user accesses the website, they can book an available room on the website.

1. The React app deployed on S3 [2] sends the static content to the user's browser.
2. The website displays all the available rooms on the home page by calling the GET method for the /rooms endpoint.
3. The user clicks on the book button on one of the available rooms.
4. The request gets passed to the API Gateway.
5. The API gateway passes this request to the lambda function for processing this information.
6. The lambda function cleans and processes this information and then updates the DynamoDB [4].
7. The lambda function returns this success status information to the API gateway.
8. The API gateway sends the response back to the React frontend.

Figure 7 shows the flow diagram for searching and booking room flows.

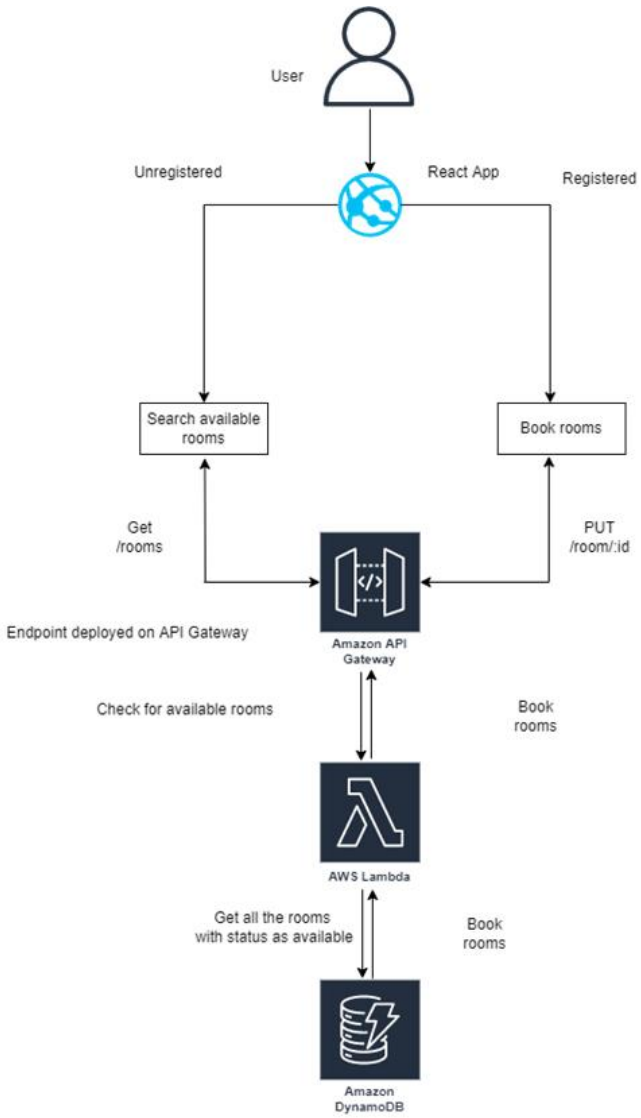

Figure 7: Searching and booking rooms flow diagram built using draw.io [1]

Chatbot Interactions

A user can utilize the chatbot on the website both as a guest user and after logging in. A guest user can only search for available rooms, while a logged-in user can book a room and order food using the chatbot.

Figure 8 shows the flow diagram for Lex [6] chatbot interactions.

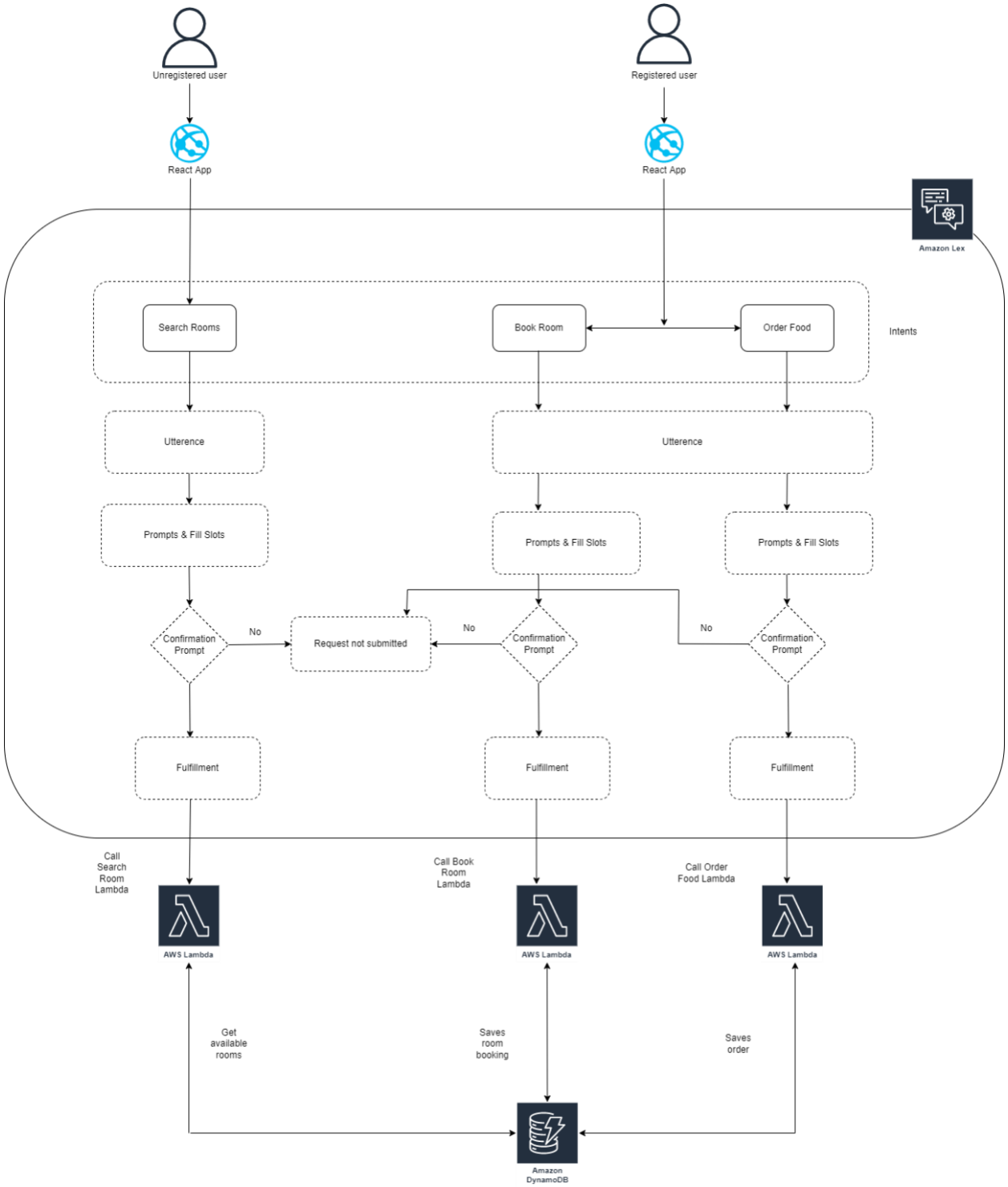

Figure 8: Chatbot flow diagram showing Lex [6] interactions built using draw.io [1]

Chatbot - Unregistered User Flow (Search Rooms)

1. When guest user accesses the website, they can access the chatbot by clicking on the chat icon at the bottom right corner.

2. After clicking on the chat icon, a chat window appears on the website.
3. A user can type in the chat window to search for the available rooms.
4. The user can type one of the following texts to get a valid response from the backend [6].

   I want to search the available rooms
   Search the available rooms
   I want to know if the room is available between ___ and ___ dates.
   Are there any deluxe rooms available?

5. Based on the text typed by the user, the chatbot will respond by asking a few more questions to get the entire information from the user.
6. The user is prompted with two questions.

   Which type of room? (Basic/Premium)
   What are the start date and end date of your stay?

7. Once the user answers all the questions, the system will have all the information to process this information.
8. The information entered by the user is then passed to a lambda function. The lambda function cleans and processes the information.
9. The lambda function then calls the DynamoDB to get the list of available rooms in that category.
10. The lambda function receives this information and sends the response back to the lex chatbot.
11. The chatbot passes this information to the front-end to fulfill the user's request of searching the available rooms [6].

Chatbot - Registered User Flow (Book a Room)

1. When registered user accesses the website, they can access the chatbot by clicking on the chat icon at the bottom right corner.
2. After clicking on the chat icon, a chat window appears on the website.
3. A user can type in the chat window to book a room.
4. The user can type one of the following texts to get a valid response from the backend.

   I want to book a room
   Book a room
   I want to book a basic room.
   I want to book a deluxe room from ____ to _____.

5. Based on the text typed by the user, the chatbot will respond by asking a few more questions to get the entire information from the user.
6. The user is prompted with two questions [6].

   Which type of room? (Basic/Premium)
   What are the start date and end date of your stay?

7. Once the user answers all the questions, the system will have all the information to process this information [6].
8. The chatbot asks for confirmation [6].
9. The information entered by the user is then passed to a lambda function. The lambda function cleans and processes the information.
10. The lambda function then updates the DynamoDB to book the room.
11. The lambda function receives this information and sends the response back to the lex chatbot.
12. The chatbot passes this information to the front-end to fulfill the user's request of booking the room.

Chatbot - Registered User Flow (Order Food):

1. When registered user accesses the website, they can access the chatbot by clicking on the chat icon at the bottom right corner.
2. After clicking on the chat icon, a chat window appears on the website.
3. A user (who has already booked a room) can type in the chat window to order food.
4. The user can type one of the following texts to get a valid response from the backend.

   I want to order breakfast
   Please serve breakfast

5. The chatbot asks for confirmation [6].
6. The information entered by the user is then passed to a lambda function. The lambda function cleans and processes the information.
7. The lambda function then updates the DynamoDB to order food.
8. The lambda function receives this information and sends the response back to the lex chatbot.
9. The chatbot passes this information to the front-end to fulfill the user's request of ordering food.

Report Generation and Visualization

Our application shall utilize the Google Data Studio [16] service to generate interactive graphs and dashboards from the application data. This service shall generate graphs that display customers' booking patterns and food ordering patterns. We shall generate reports that provide insights on user login activity using AWS CloudTrail [12] and CloudWatch [13] services.

Website building and Hosting

The frontend of our project is built using the React [17] frontend library. The frontend is deployed to a S3 bucket on AWS that hosts the static assets generated by building the React frontend code. Following are the sequence of actions that take place in the deployment process:

1. A git commit is made, and the frontend code is pushed to the **main** branch of the project repository on GitLab [18].
2. The GitLab CI/CD pipeline builds the static assets from the React code using the **npm run build** command and uploads the static assets to an S3 bucket.
3. The S3 bucket [2] is configured to support website hosting. The website can be accessed using the URL generated for the bucket.

Modules and work distribution

Module Name	**Group Member Name**
User Management	Jayasree Kulothungan
Authentication	Sourav Malik
Online Support	Aditya Mahale
Message Passing	Udit Gandhi, Sai Chand Kolloju
Machine Learning	Rishika Bajaj
Web Application Building Hosting	Udit Gandhi, Sai Chand Kolloju, Aditya Mahale, Sourav Malik, Jayasree Kulothungan, Rishika bajaj Sai Chand Kolloju
Other Essential Modules: 1. Report Generation Module 2. Visualization	 Jayasree Kulothungan Sai Chand Kolloju

Meeting Logs

Figures 9 through 11 show the meeting records on Microsoft Teams.

Meeting Number: 1
Date of the meeting: June 23, 2022
Time: 9:00pm - 9:23pm
Place: Microsoft Teams
Agenda: Project conception report feedback discussion and initial database design
Discussion: Analysed the feedback given for the project conception report submission to incorporate the suggestions made in the future project submissions. We created an initial database design subject to additional changes in the future.

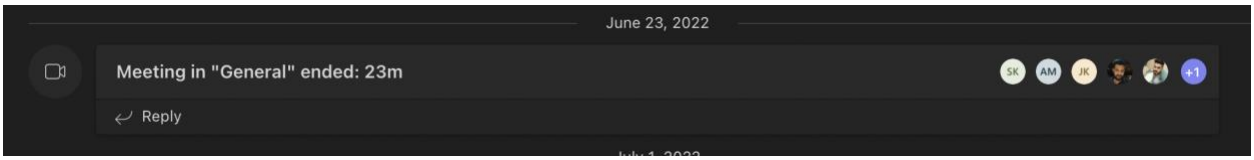

Figure 9: Microsoft Teams meeting record for meeting 1

Meeting Number: 2
Date of the meeting: July 1, 2022
Time: 10:00pm - 10:10pm
Place: Microsoft Teams
Agenda: Studying the design document requirements
Discussion: Speculated over design document requirements and formed a rough draft of the document highlighting the sections to be included in the document. We decided to meet in person to work on the document.

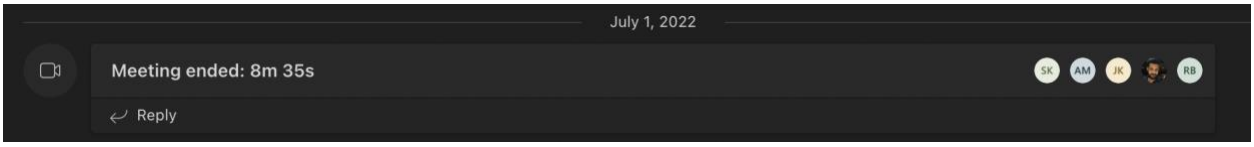

Figure 10: Microsoft Teams meeting record for meeting 2

Meeting Number: 3
Date of the meeting: July 3, 2022
Time: 1:00pm - 5:30pm
Place: Collaborative Health Education Building
Agenda: Project architecture and roadmap discussion
Discussion: Discussed how the overall architecture of the project should be designed and studied the interactions between the services used in building the project. We revisited the requirements document to better understand the scope of the project. We also made changes to our initial database design to include other related fields. We went through the design document requirements and decided to distribute the work.

Meeting Number: 4
Date of the meeting: July 4, 2022
Time: 2:00pm - 2:15pm
Place: Microsoft Teams
Agenda: Task distribution for design document
Discussion: Distributed the tasks among the team members and assigned responsibilities of completing their portion of the design document.

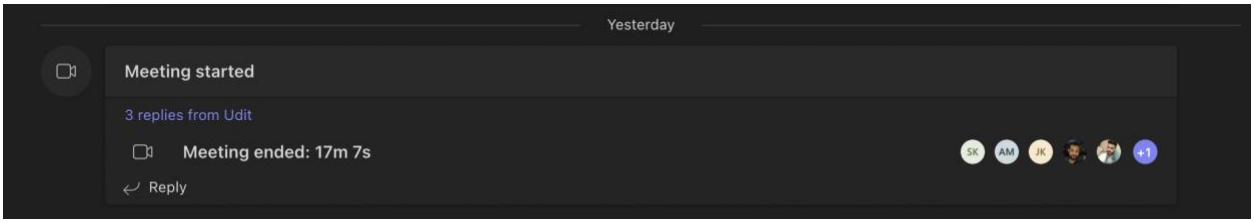

Figure 11: Microsoft Teams meeting record for meeting 4

References

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

[2] "S3," *Amazon*, 2002. [Online]. Available: https://aws.amazon.com/s3/. [Accessed: 05-Jul-2022].

[3] "Firestore: Nosql document database | google cloud," *Google*. [Online]. Available: https://cloud.google.com/firestore/. [Accessed: 05-Jul-2022].

[4] D. Rangel, "DynamoDB: Everything you need to know about Amazon Web Service's NoSQL database," *Amazon*, 2015. [Online]. Available: https://aws.amazon.com/dynamodb/. [Accessed: 05-Jul-2022].

[5] H. Roose, "Cognito," *Amazon*, 1987. [Online]. Available: https://aws.amazon.com/cognito/. [Accessed: 05-Jul-2022].

[6] "Conversational AI and Chatbots - Amazon Lex - Amazon Web Services", *Amazon Web Services, Inc.*, 2022. [Online]. Available: https://aws.amazon.com/lex/. [Accessed: 05- Jul-2022]

[7] D. A. V. I. D. MUSGRAVE, "Lambda," *Amazon*, 2022. [Online]. Available: https://aws.amazon.com/lambda/. [Accessed: 05-Jul-2022]. \

[8] "Cloud functions | google cloud," *Google*. [Online]. Available: https://cloud.google.com/functions/. [Accessed: 05-Jul-2022].

[9] "Pub/Sub for Application & Data Integration | google cloud," *Google*. [Online]. Available: https://cloud.google.com/pubsub/. [Accessed: 05-Jul-2022].

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

[11] "Cloud natural language | google cloud," *Google*. [Online]. Available: https://cloud.google.com/natural-language/. [Accessed: 05-Jul-2022].

[12] "Secure standardized logging - AWS CloudTrail - Amazon Web Services." [Online]. Available: https://aws.amazon.com/cloudtrail/. [Accessed: 06-Jul-2022].

[13] "Amazon Cloudwatch - application and Infrastructure Monitoring." [Online]. Available: https://aws.amazon.com/cloudwatch/. [Accessed: 06-Jul-2022].

[14] "Firebase Cloud messaging | send notifications across platforms at no-cost," *Google*. [Online]. Available: https://firebase.google.com/products/cloud-messaging/. [Accessed: 05-Jul-2022].

[15] "Email delivery, API, Marketing Service," *SendGrid*. [Online]. Available: https://sendgrid.com/. [Accessed: 05-Jul-2022].

[16] *Google Data studio overview*. [Online]. Available: https://datastudio.google.com/. [Accessed: 05-Jul-2022].

[17] "React – a JavaScript library for building user interfaces," – *A JavaScript library for building user interfaces*. [Online]. Available: https://reactjs.org/. [Accessed: 05-Jul-2022].

[18] "The One DevOps platform," *GitLab*. [Online]. Available: https://about.gitlab.com/. [Accessed: 05-Jul-2022].