CSCI 5410
Serverless Data Processing
Group Project: DALVacationHome
Sprint – 3

Submitted By:

Abhishek Latawa: B00935048
Bhishman Desai: B00945177
Dharmil Nareshkumar Shah: B00965853
Shivang Patel: B00979841

Table of Contents

Details of research:

Following are the modules on which research and documentation has been developed by our group which is required for the project:

1. Virtual Assistant Module:

- **Google DialogFlow:** A Dialogflow agent is a virtual agent that handles concurrent conversations with your end-users. It is a natural language understanding module that understands the nuances of human language. Dialogflow translates end-user text or audio during a conversation to structured data that your apps and services can understand. You design and build a Dialogflow agent to handle the types of conversations required for your system**.** [1].

Intents

An intent categorizes an end-user's intention for one conversation turn. For each Agent, you define many intents, where your combined intents can handle a complete conversation. When an end-user writes or says something, referred to as an *end-user expression*, Dialogflow matches the end-user expression to the best intent in your agent. Matching an intent is also known as *intent classification*.

For example, you could create a weather agent that recognizes and responds to end-user questions about the weather. You would likely define an intent for questions about the weather forecast. If an end-user says, "What's the forecast?", Dialogflow would match that end-user expression to the forecast intent. You can also define your intent to extract useful information from the end-user expression, like a time or location for the desired weather forecast. This extracted data is important for your system to perform a weather query for the end-user.[1]

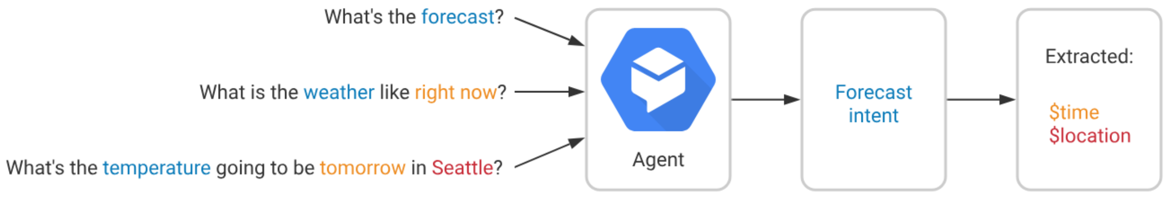

What's the forecast?
What is the weather like right now?
What's the temperature going to be tomorrow in Seattle?
Agent
Forecast intent
Extracted:
$time
$location

Figure 1: Dialogflow Implementation abstract

A basic intent contains the following:

- **Training phrases**: These are example phrases for what end-users might say. When an end-user expression resembles one of these phrases, Dialogflow matches the intent. You don't have to define every possible example, because Dialogflow's built-in machine learning expands on your list with other, similar phrases.
- **Action**: You can define an action for each intent. When an intent is matched, Dialogflow provides the action to your system, and you can use the action to trigger certain actions defined in your system.
- **Parameters**: When an intent is matched at runtime, Dialogflow provides the extracted values from the end-user expression as *parameters*. Each parameter has a type, called the entity type, which dictates exactly how the data is extracted. Unlike raw end-user input, parameters are structured data that can easily be used to perform some logic or generate responses.
- **Responses**: You define text, speech, or visual responses to return to the end-user. These may provide the end-user with answers, ask the end-user for more information, or terminate the conversation.

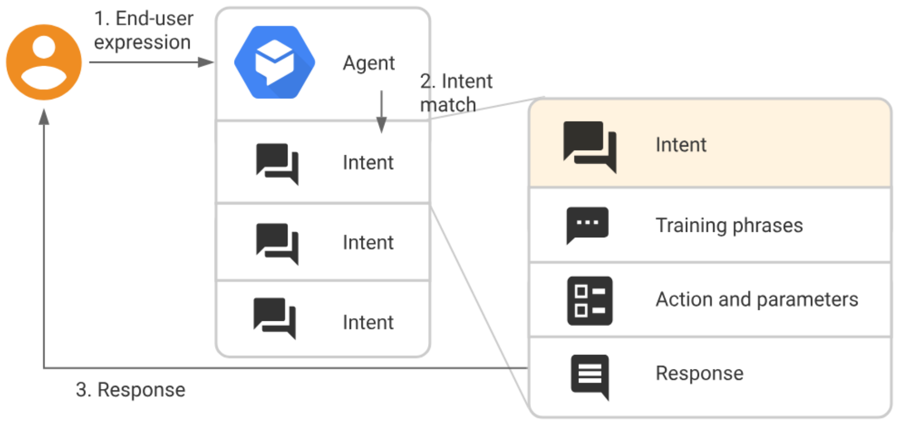


Figure 2: Intent Matching in Dialogflow

Entities

Each intent parameter has a type, called the entity type, which dictates exactly how data from an end-user expression is extracted.

Dialogflow provides predefined system entities that can match many common types of data. For example, there are system entities for matching dates, times, colors, email addresses, and so on. You can also create your own custom entities for matching custom data. For example, you could define a *vegetable* entity that can match the t ypes of vegetables available for purchase with a grocery store agent.[1]

- **AWS Lambda** integrates seamlessly with Amazon Lex for executing business logic, data retrieval, and updates. Lambda functions can be triggered by Lex to handle complex processing, such as validating user inputs, querying databases, or interacting with other AWS services. This integration allows for a fully serverless architecture, minimizing the need for managing infrastructure while ensuring scalability and reliability [3][4].

1. **Navigational Assistance (e.g., "How to register?"):**

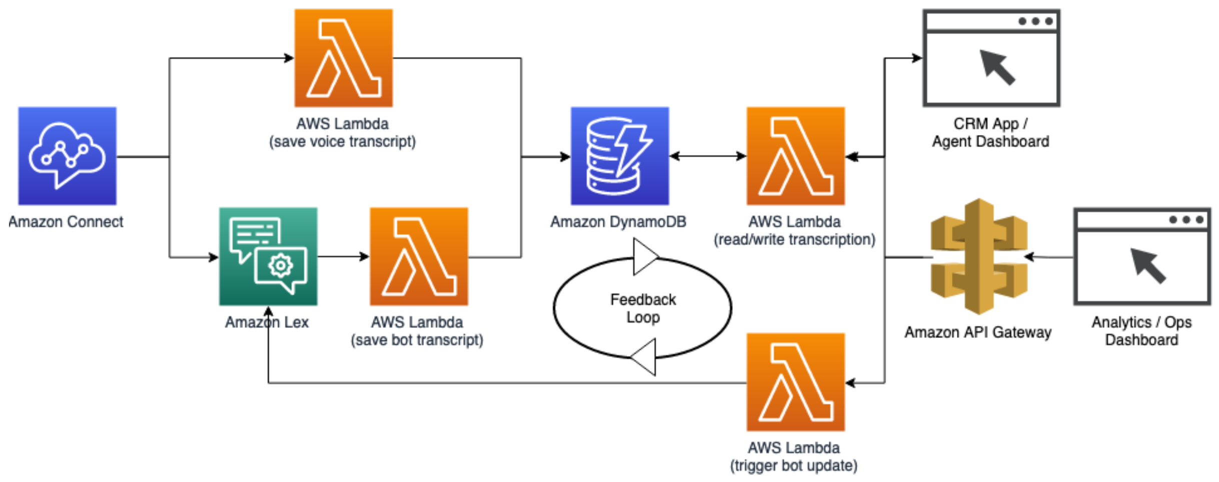
User Interaction: The user asks, "How do I register?"
AWS Lex: Captures the query and triggers an appropriate intent.
AWS Lambda: A Lambda function associated with this intent fetches the registration instructions from a predefined source or generates a response.
Response: The chatbot responds with a step-by-step guide on how to register.

2. **Searching Room Numbers and Usage Details:**

User Interaction: The user asks, "What is the usage of room number 101?" or "What is the stay duration for booking reference ABC123?"
AWS Lex: Captures the query and extracts the relevant information (room number or booking reference code).
AWS Lambda: The Lambda function queries DynamoDB to retrieve the requested details.
DynamoDB: Contains tables with room usage data and booking information.
Response: The chatbot provides the room's usage details or stay duration based on the booking reference.

3. **Accepting Customer Concerns and Forwarding to Property Agents:**

User Interaction: The user states a concern, e.g., "I have an issue with my room's cleanliness."
AWS Lex: Captures the user's concern.
AWS Lambda: The Lambda function records the concern in and triggers a workflow to post the concern on GCP Pub/Sub topic.
Firestore: Stores the customer concerns along with relevant details such as the user's contact information and concern description along with the details of property agent to whom the concern was addressed.



AWS Lambda (save voice transcript)
CRM App / Agent Dashboard
Amazon Connect
Amazon DynamoDB
AWS Lambda (read/write transcription)
Amazon Lex
AWS Lambda (save bot transcript)
Feedback Loop
Amazon API Gateway
Analytics / Ops Dashboard
AWS Lambda (trigger bot update)

Figure 3: Workflow

2. Message Passing Module:

- **Google Cloud Pub/Sub** can be used for asynchronous message passing between customers and property agents [2]. Customer concerns can be published to a Pub/Sub topic, and a Cloud Function is subscribed to this topic to forward the concerns to property agents. Additionally, the Google Natural Language API can be used for sentiment analysis of customer feedback, enhancing the feedback processing capabilities [2].

  How:

  Customer Submits Concern:
  Input: The customer submits a concern based on their booking reference code with Lex bot.
  Process: The concern is published to a GCP Pub/Sub topic.

  GCP Pub/Sub:
  Publisher: Publishes the customer concern message to a specific topic.
  Subscriber: A function subscribed to this topic receives the message.

  Processing the Concern: **Subscribed Function (GCP Cloud Function)**:
- Extracts the concern details from the message.
- Randomly selects a property agent to forward the concern.
- Logs the communication details in a NoSQL database (Firestore).
- Notifies the property agent with the concern details.

  NoSQL Database Logging:
  Logs the entire communication process, including the original concern, the assigned property agent, and timestamps for tracking purposes.

3. Notifications Module:

For sending notifications, AWS SNS (Simple Notification Service) and SQS (Simple Queue Service) are pivotal:

- **AWS SNS** is used to send notifications via email or SMS. For example, we can configure SNS to send an email to users upon successful registration or login [3].
- **AWS SQS** works with SNS to handle message queues between different application components. For instance, a booking request can be placed in an SQS queue, which triggers a Lambda function for booking approval [5][6]. This decouples the booking request process from the approval process, enhancing the system's scalability and reliability.

1. When user logs in or register successfully:
AWS SNS:
- When a user successfully registers or logs in, an event is triggered.
- This event is captured by a Lambda function.
- The Lambda function uses SNS to send an email notification to the registered user's email address.

Example:
- Upon successful registration, the user receives an email saying, "Thank you for registering!"
- Upon successful login, the user receives an email saying, "You have successfully logged in!"

2. Booking Confirmation and Failure Notifications:

AWS SNS:
- When a booking is successfully processed or fails, an event is triggered.
- A Lambda function handles this event.
- The Lambda function uses SNS to send an email notification with the booking status to the user.

Example
- Upon successful booking, the user receives an email saying, "Your booking has been confirmed. Booking ID: 12345."
- If the booking fails, the user receives an email saying, "We are sorry, but your booking could not be completed."

4. Data Analysis & Visualization:

In this section, we detailed the implementation of data analysis and visualization for the DALVacationHome project. This module provides crucial insights and feedback for property agents, guests, and customers, leveraging Google Cloud Platform (GCP) services and Google Natural Language API for sentiment analysis.

Data Collection and Storage
- **Feedback Storage:**
  - Customer feedback is collected and stored in Firestore through Lambda call in AWS using API Gateway.

    - Each feedback entry contains the customer's username, review description, and the corresponding room ID.

Sentiment Analysis

- **Google Natural Language API:**
    - Feedback descriptions are sent to the Google Natural Language API for sentiment analysis.
    - The API returns a sentiment score and magnitude, which helps in understanding the overall sentiment (positive, neutral, or negative) of the feedback.

Visualization
Admin Dashboard:

- **Looker Studio:**
    - A dashboard which visualizes reservations, total number of users, total number of rooms published and the feedback analysis in chart form.
    - The dashboard is embedded within an iframe on the admin page for seamless integration. [7]

Customer Feedback:

- **Frontend Display:**
    - Feedback is displayed given by the users for specific room, showing the username, review description, and sentiment’s polarity.

Sentiment Visualization:

- Sentiment scores are visually represented using color codes with the text of “POSITIVE”, “NEGATIVE” and “NEUTRAL” to quickly convey the feedback's nature. [8]

5. Web Application Building and Deployment:

Overview

For the web application building and deployment, we have utilized React for the frontend and Terraform to automate the deployment process on GCP Cloud Run. [9]

Terraform Script Explanation

Service Account Creation

A service account specific to the project is created. This account is given a unique identifier and linked to the project, with an appropriate display name and description. [10]

IAM Binding for Permissions

The service account is assigned the **roles/editor** role, which provides the necessary permissions for logging and pushing to the artifact registry.

Cloud Build Trigger from GitHub

A Cloud Build trigger is defined for continuous deployment. This trigger activates on pushes to the main branch of the specified GitHub repository. This GitHub is a mirror repository of Dal GitLab repository. It uses Docker to build the application image and pushes it to Google Container Registry.

Null Resource to Run the Cloud Build Trigger

A null resource ensures that the Cloud Build trigger runs after being defined. This is done using a local-exec provisioner to automated the run of the Cloud Build trigger.

Cloud Run Service Definition

The built Docker image is deployed to Cloud Run. The Cloud Run service is configured with scaling options, container settings, environment variables, and resource limits. Traffic is set to 100% for the latest deployment.

IAM Policy for Unauthenticated Access

An IAM policy is created to allow unauthenticated access to the Cloud Run service. This policy grants the **roles/run.invoker** role to all users and is applied to the Cloud Run service.

All in all, the Terraform script automates the creation and deployment of a web application on GCP Cloud Run. It sets up a service account, configures permissions, triggers a build on GitHub changes, and deploys the application using Cloud Run.

Architecture for the Project:

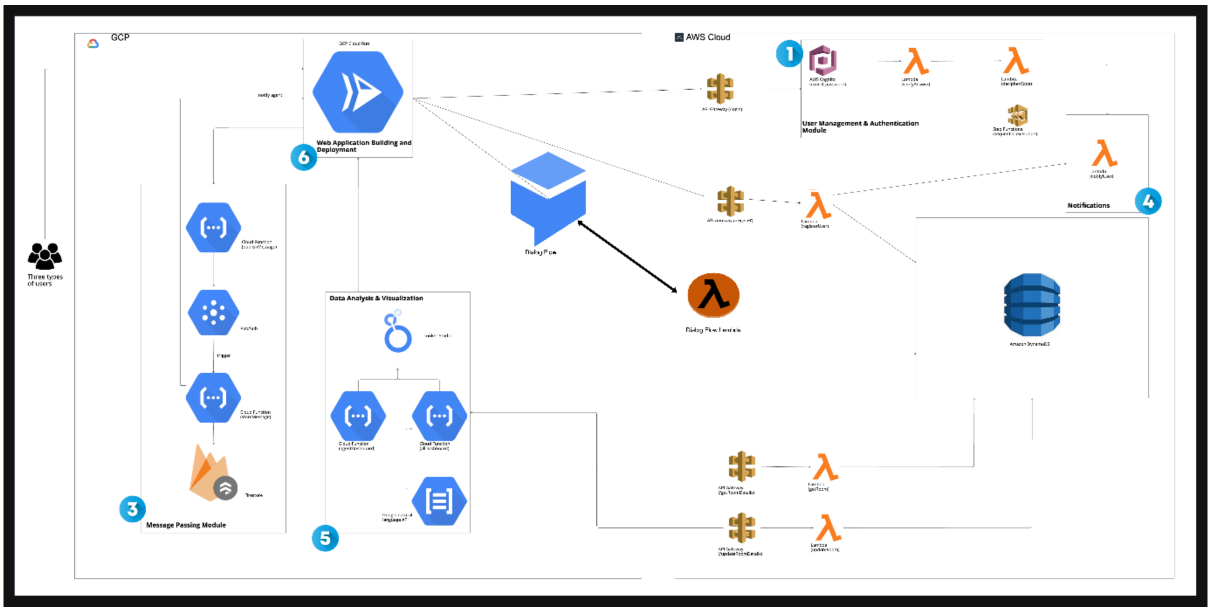

GCP
AWS Cloud
Web Application Building and Deployment
User Management & Authentication Module
Notifications
Three types of users
Dialog Flow
Data Analysis & Visualization
Message Passing Module
1
3
4
5
6

Figure 4: Architecture Diagram [11]

Pseudo Code

Authentication**:**

Login:
<u>State Variables</u>

- “username” to store the entered username
- “password" to store the entered password
- “usernameError” to store any validation error for the username
- “passwordError" to store any validation error for the password
- “loginError” to store any error message from the login attempt

<u>Functions:</u>

- Prevents default
- Checks for the status
- If response status 200, then it will proceed else produces error for incorrect credentials

<u>Render Form:</u>

- Display a form with TextField components for username and password
- Display error messages if usernameError, passwordError, or loginError are set
- Include a submit Button that triggers the handleSubmit function

Signup:

<u>State Variables:</u>

- ‘username’ to store the entered username
- ‘Name’ to store the entered name
- ‘email’ to store the entered email

- ‘password’ to store the entered password
- ‘role’ to store the selected role (default to "User")
- ‘usernameError’ to store any validation error for the username
- ‘nameError’ to store any validation error for the name
- ‘emailError’ to store any validation error for the email
- ‘passwordError’ to store any validation error for the password
- ‘signupError’ to store any error message from the signup attempt

Functions:

- Prevents default
- Checks for the status
- If response status 200, then it will proceed else produces error for empty textfields

Render Form:

- Display a form with TextField components for name, username, email, and password
- Display a Select component for role
- Display error messages if nameError, usernameError, emailError, passwordError, or signupError are set
- Include a submit Button that triggers the handleSubmit function

Message Passing:

Producer Cloud Function:

```
Step 1: Set environment variable GOOGLE_APPLICATION_CREDENTIALS to path-to-service-account-key.json
Step 2: Initialize project_id , topic_id and Initialize PublisherClient
Step 3: Extract message from event body
Step 4: If message is empty:
        Return {'statusCode': 400, 'body': 'Message not provided.'}
Step 5: Encode message as UTF-8 and publish to Pub/Sub:
        Try:
            Publish message to topic_path
            Get message_id from future.result()
            Return {'statusCode': 200, 'body': f'Message {message_id} published.'}
        Catch Exception as e:
            Print 'Error publishing message: e'
            Return {'statusCode': 500, 'body': 'Error publishing message.'}
```

Consumer Cloud Function:

```
Step 1: Initialize Firestore client
Step 2: Extract message from base64 encoded data in cloud event Parse message as JSON into parsedMessage
Step 3: Retrieve propertyAgents array from environment variables Randomly select an agent from propertyAgents as chosenAgent
Step 4: Send email to chosenAgent with parsedMessage.clientId and parsedMessage.complaint
Step 5: Try: Add a document to 'complaint_logs' collection in Firestore with:
        - clientId: parsedMessage.clientId
        - agentId: chosenAgent
        - message: parsedMessage.complaint
        - timestamp: current server timestamp Log 'Complaint logged successfully.'
Step 6: Catch error: Log 'Error logging complaint: ' + error
```

Notifications and Room Bookings**:**

1. Adding a room

```
INITIALIZE DynamoDB client
GET reference to the DynamoDB table 'Rooms'
 TRY:
    PARSE the request body from the incoming event
    EXTRACT roomId from request body
    EXTRACT Features from request body
    EXTRACT type from request body
    EXTRACT Price from request body
 CREATE a new room object with:
 roomid = roomId
        type = type
        features = Features
        price = Price
STORE the new room object in the DynamoDB table using put_item
  RETURN success response with status code 200 and message 'Room added successfully'
CATCH any exception:
RETURN error response with status code 500 and the error message
```

2. Fetch Rooms:

```
INITIALIZE DynamoDB client
GET reference to the DynamoDB table 'Rooms'
TRY:
SCAN the 'Rooms' table to retrieve all items
EXTRACT room items from the response
RETURN success response with status code 200 and room data, converting Decimal to float
CATCH any exception:
 RETURN error response with status code 500 and the error message
```

3. Delete rooms

```
INITIALIZE DynamoDB client
GET reference to the DynamoDB table 'Rooms'
TRY:
    PARSE the event body to extract room details
```

```
        - roomId
        - Features
        - type
        - Price
 CREATE a new room item dictionary with extracted details
ADD the new room item to the 'Rooms' table using put_item
 RETURN success response with status code 200 and success message
CATCH any exception:
RETURN error response with status code 500 and the error message
```

4. Update Rooms

```
INITIALIZE DynamoDB client
GET reference to the DynamoDB table 'Rooms'
 TRY:
    PARSE the event body to extract update details
        - roomId
        - type (optional)
        - features (optional)
        - price (optional)
 INITIALIZE update expression string
  INITIALIZE dictionaries for attribute values and names
  FOR each field (type, features, price) in the request body:
 IF the field is present:
APPEND to the update expression
ADD the field to the expression attribute values
ADD the field to the expression attribute names
 REMOVE trailing comma from update expression
 UPDATE the room item in the 'Rooms' table using update_item with the built update expression
 RETURN success response with status code 200 and success message
CATCH any exception:
RETURN error response with status code 500 and the error message
```

5. Room Booking Request

```
INITIALIZE SQS client using boto3
 RETRIEVE QUEUE_URL from environment variables
 EXTRACT booking details from event:
    - customerId
    - roomId
    - startDate
    - endDate
 CREATE booking_request object with extracted details
 SEND booking_request to SQS queue using send_message
 RETURN success response with status code 200 and success message
```

6. Book Room Approval

```
INITIALIZE DynamoDB and SNS clients
 RETRIEVE environment variables
 DEFINE lambda_handler(event, context):
    GET DynamoDB table
 FOR each record in event['Records']:
TRY:
PARSE SQS message for booking details
GET room from DynamoDB
 IF room does not exist or date overlap:
 RAISE exception
 ADD new booking to room
UPDATE room in DynamoDB
SEND confirmation notification via SNS
RETURN success response
EXCEPT Exception as e:
SEND failure notification via SNS
RETURN error response
```

7. ChatBot- DialogFlow

```
INITIALIZE DynamoDB and Requests clients

RETRIEVE environment variables for booking table and API URLs

DEFINE lambda_handler(event, context):

  GET DynamoDB table

  FOR each record in event['Records']:

    TRY:
      PARSE message from the record for booking details, complaint, and description

      EXTRACT booking_reference, complaint, description, and intent

      IF booking_reference is present:
        FETCH booking details from DynamoDB using booking_reference
        IF booking does not exist:
          PREPARE response indicating booking details not found
        ELSE:
          PREPARE response with booking details (username, startDate, endDate)
        RETURN response with booking details or error message

      IF complaint is present:
        PREPARE payload with Client ID and complaint
        SEND POST request to external API to register complaint
        IF POST request is successful:
          PREPARE success response indicating complaint registration
        ELSE:
          PREPARE failure response indicating POST request failure
        RETURN response
    IF description is present:
        PREPARE payload with description
        SEND POST request to external API to send complaint details
        IF POST request is successful:
            PREPARE success response indicating complaint details sent
        ELSE:
            PREPARE failure response indicating POST request failure
        RETURN response

    IF intent is "Fetch Messages":
        SEND GET request to external API to fetch messages
        IF GET request is successful:
            APPEND messages to messages list
            PREPARE response indicating messages are being fetched
        ELSE:
            PREPARE failure response indicating GET request failure
        RETURN response

    IF intent is "Display messages":
        CHECK if messages list is empty
        IF not empty:
            FORMAT and join messages
            CLEAR messages list
            PREPARE response with formatted messages
        ELSE:
            PREPARE response indicating no messages available
        RETURN response

    ELSE:
        PREPARE response indicating invalid request with missing parameters
        RETURN response

EXCEPT Exception as e:
    PREPARE error response indicating an exception occurred
```

LOG the exceptio RETURN error response

Testing Evidence:

Module 1: Authentication

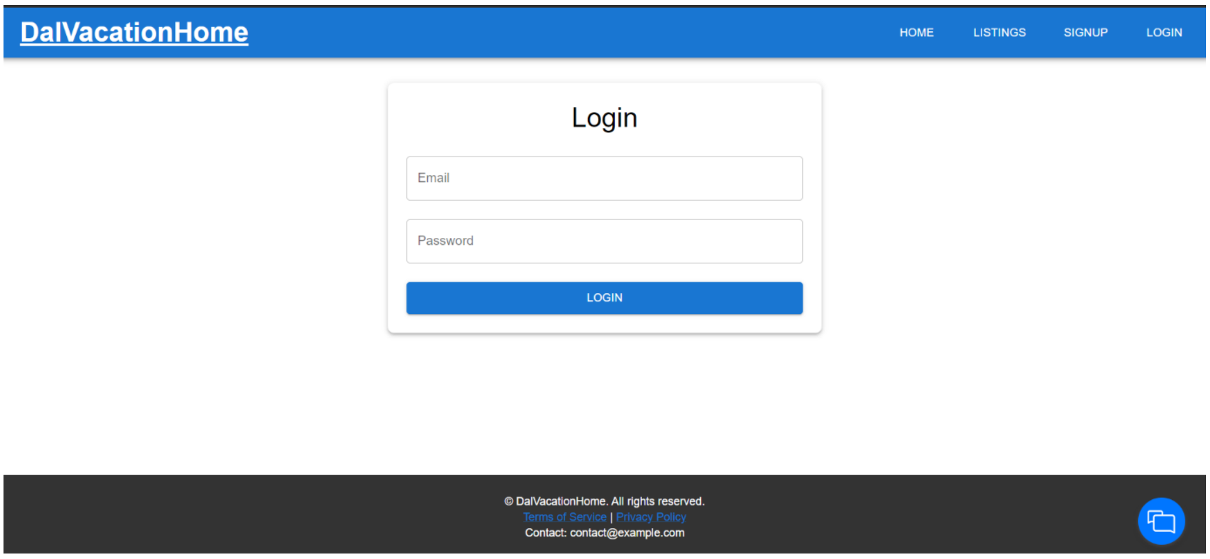

DalVacationHome
HOME
LISTINGS
SIGNUP
LOGIN
Login
Email
Password
LOGIN
© DalVacationHome. All rights reserved.
Terms of Service | Privacy Policy
Contact: contact@example.com

Figure 5: Login Page

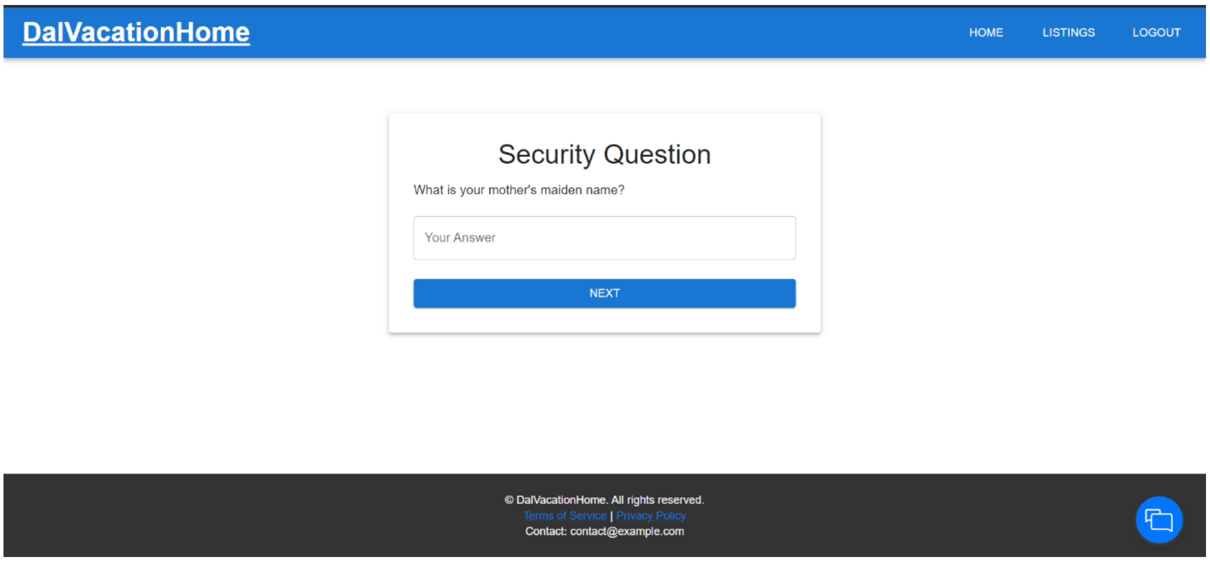
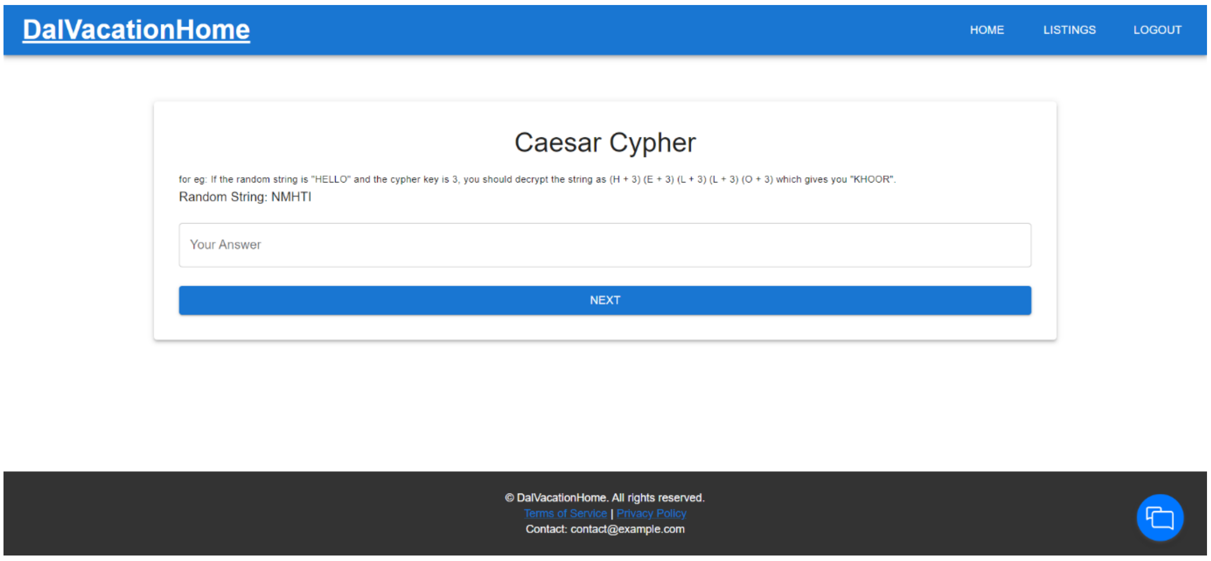

DalVacationHome
HOME
LISTINGS
LOGOUT
Security Question
What is your mother's maiden name?
Your Answer
NEXT
© DalVacationHome. All rights reserved.
Terms of Service | Privacy Policy
Contact: contact@example.com

Figure 6: Login Security Authentication


DalVacationHome
HOME LISTINGS LOGOUT

Caesar Cypher

for eg: If the random string is "HELLO" and the cypher key is 3, you should decrypt the string as (H + 3) (E + 3) (L + 3) (L + 3) (O + 3) which gives you "KHOOR".

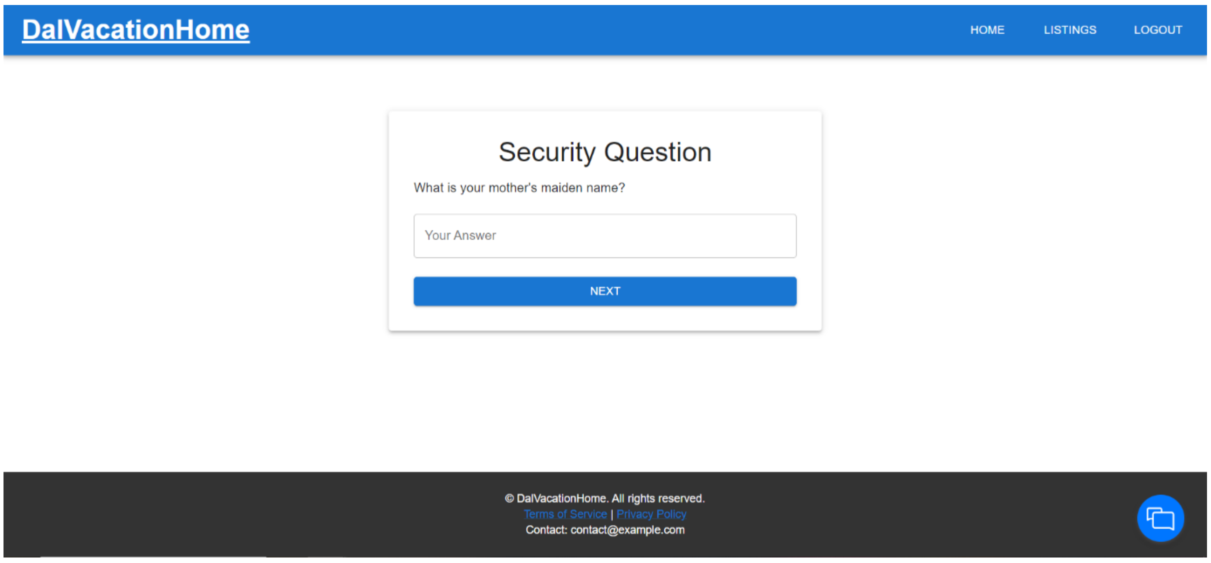
Random String: NMHTI

Your Answer

NEXT

© DalVacationHome. All rights reserved.
Terms of Service | Privacy Policy
Contact: contact@example.com

Figure 7: Ceaser Cypher during Login


DalVacationHome
HOME LISTINGS LOGOUT

Security Question

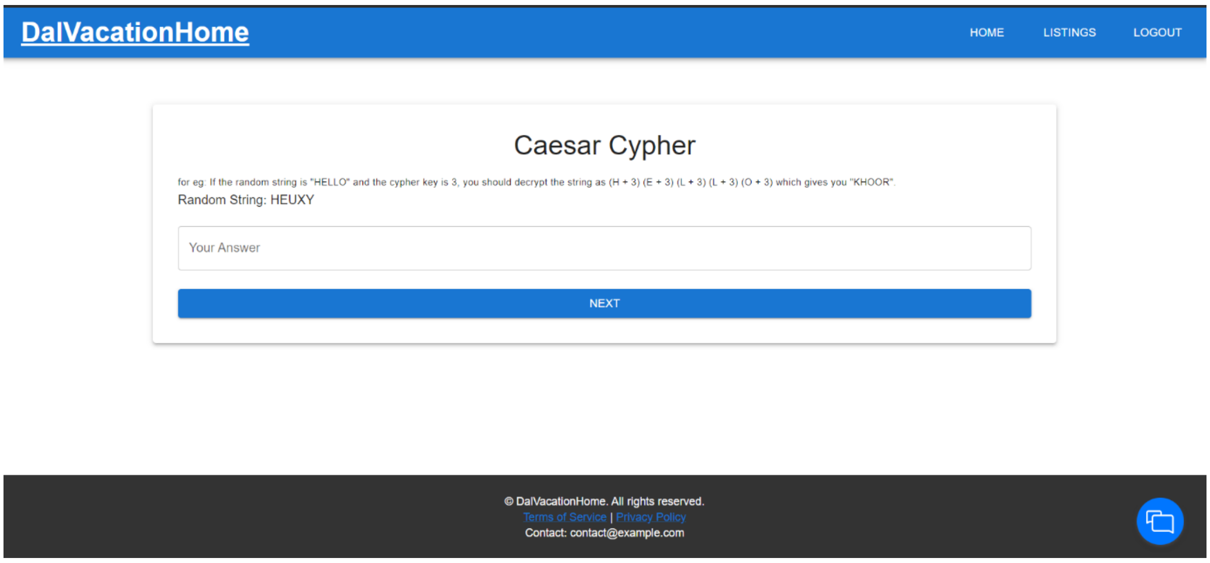
What is your mother's maiden name?

Your Answer

NEXT

© DalVacationHome. All rights reserved.
Terms of Service | Privacy Policy
Contact: contact@example.com

Figure 8: Signup Security Questions



Caesar Cypher

for eg: If the random string is "HELLO" and the cypher key is 3, you should decrypt the string as (H + 3) (E + 3) (L + 3) (L + 3) (O + 3) which gives you "KHOOR".
Random String: HEUXY

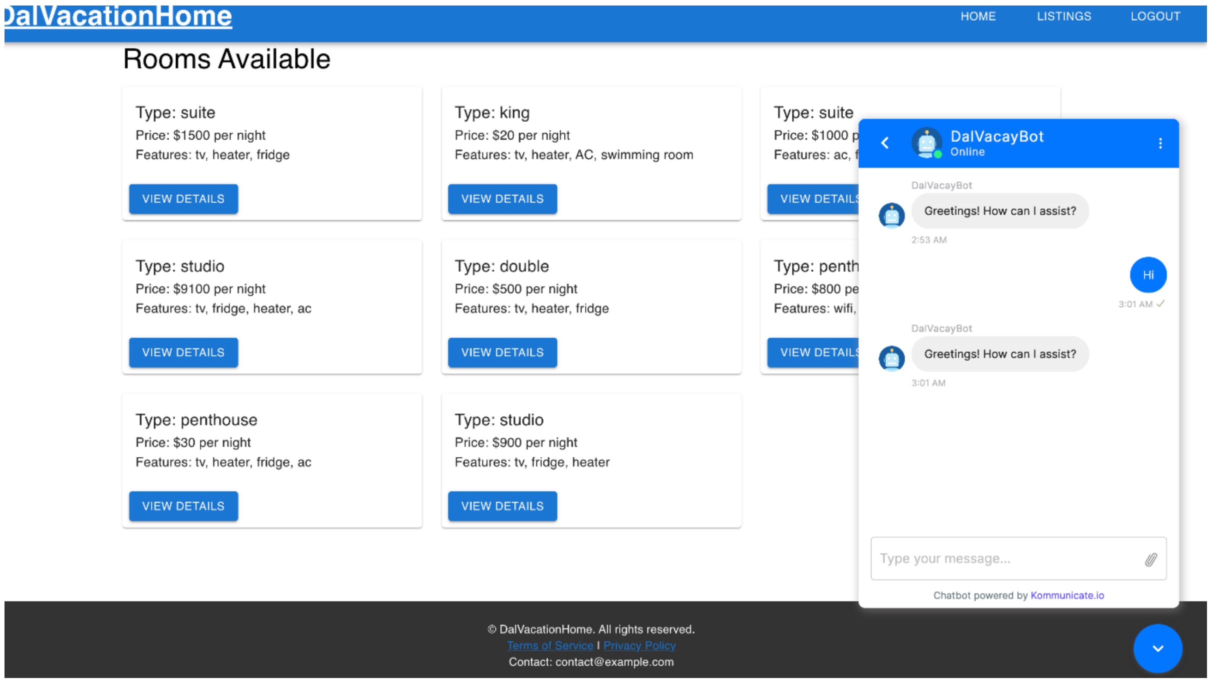
Your Answer

© DalVacationHome. All rights reserved.
Terms of Service | Privacy Policy
Contact: contact@example.com

Figure 9: Signup Ceaser Cipher

Module 2 and 3: Virtual Assistant and Pub/Sub

1. Welcome Intent



Rooms Available

Type: suite
Price: $1500 per night
Features: tv, heater, fridge

Type: king
Price: $20 per night
Features: tv, heater, AC, swimming room

Type: suite
Price: $1000 p
Features: ac, f

Type: studio
Price: $9100 per night
Features: tv, fridge, heater, ac

Type: double
Price: $500 per night
Features: tv, heater, fridge

Type: penth
Price: $800 pe
Features: wifi,

Type: penthouse
Price: $30 per night
Features: tv, heater, fridge, ac

Type: studio
Price: $900 per night
Features: tv, fridge, heater

DalVacayBot
Online

DalVacayBot
Greetings! How can I assist?
2:53 AM

Hi
3:01 AM ✓

DalVacayBot
Greetings! How can I assist?
3:01 AM

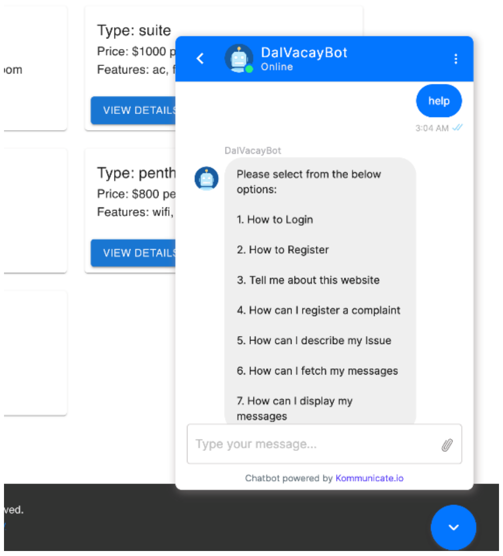
Type your message...

Chatbot powered by Kommunicate.io

© DalVacationHome. All rights reserved.
Terms of Service | Privacy Policy
Contact: contact@example.com

Figure 10: Welcome Intent

2. Help Intent



help

3:04 AM

DalVacayBot

Please select from the below options:

1. How to Login

2. How to Register

3. Tell me about this website

4. How can I register a complaint

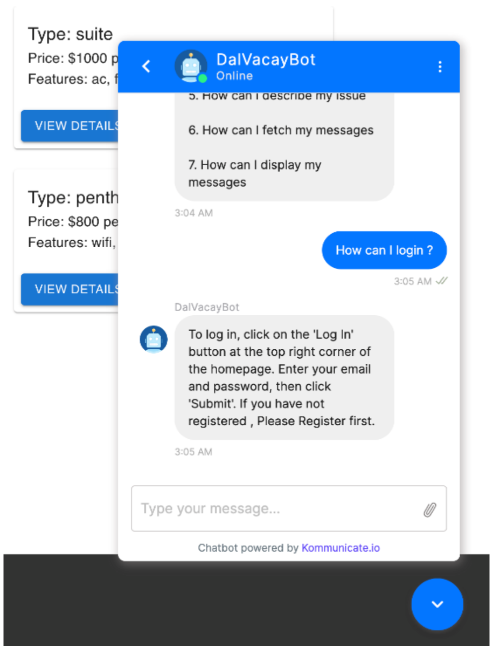
5. How can I describe my Issue

6. How can I fetch my messages

7. How can I display my messages

Figure 11: Help Intent

3. Login Intent



5. How can I describe my Issue

6. How can I fetch my messages

7. How can I display my messages

3:04 AM

How can I login ?

3:05 AM

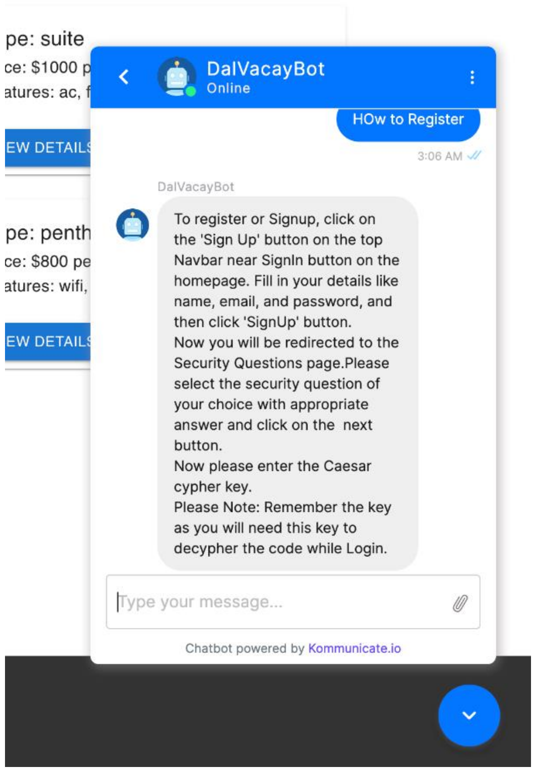
DalVacayBot

To log in, click on the 'Log In' button at the top right corner of the homepage. Enter your email and password, then click 'Submit'. If you have not registered , Please Register first.

3:05 AM

Figure 12: Login Intent

4. Register Intent



HOw to Register

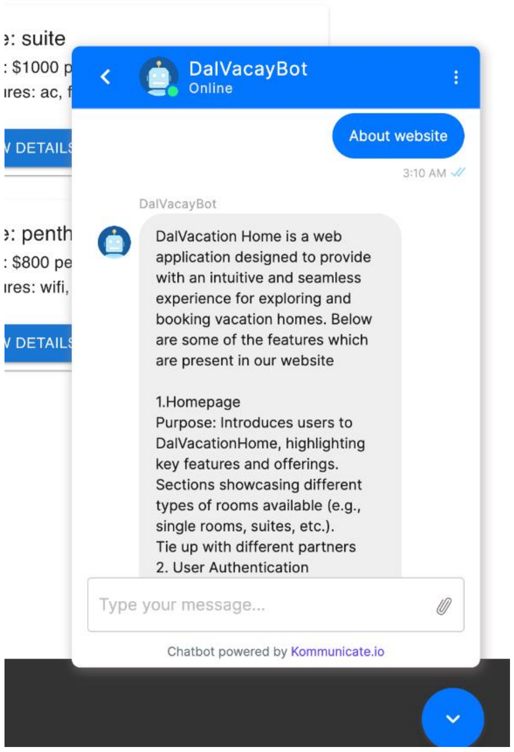
3:06 AM

DalVacayBot

To register or Signup, click on the 'Sign Up' button on the top Navbar near SignIn button on the homepage. Fill in your details like name, email, and password, and then click 'SignUp' button.
Now you will be redirected to the Security Questions page.Please select the security question of your choice with appropriate answer and click on the next button.
Now please enter the Caesar cypher key.
Please Note: Remember the key as you will need this key to decypher the code while Login.

Figure 13: Register Intent

5. About Website Intent



About website

3:10 AM

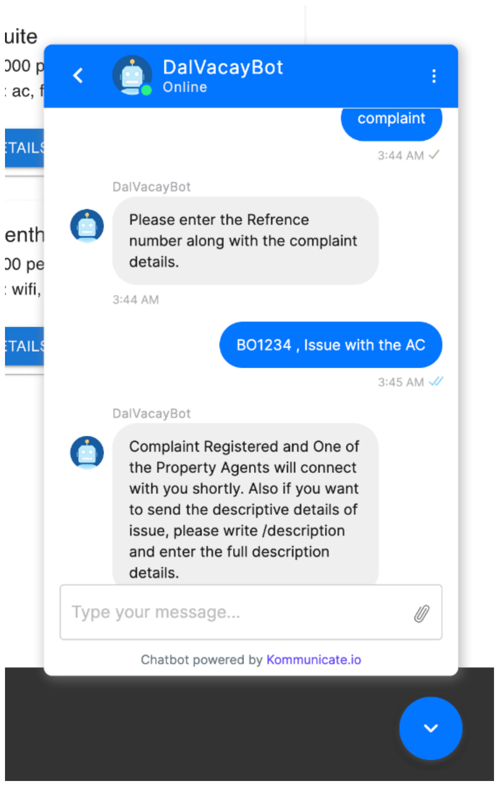
DalVacayBot

DalVacation Home is a web application designed to provide with an intuitive and seamless experience for exploring and booking vacation homes. Below are some of the features which are present in our website

1.Homepage
Purpose: Introduces users to DalVacationHome, highlighting key features and offerings.
Sections showcasing different types of rooms available (e.g., single rooms, suites, etc.).
Tie up with different partners
2. User Authentication

Figure 14: About the website intent

6. Complaint Intent



complaint

3:44 AM

DalVacayBot

Please enter the Refrence number along with the complaint details.

3:44 AM

BO1234 , Issue with the AC

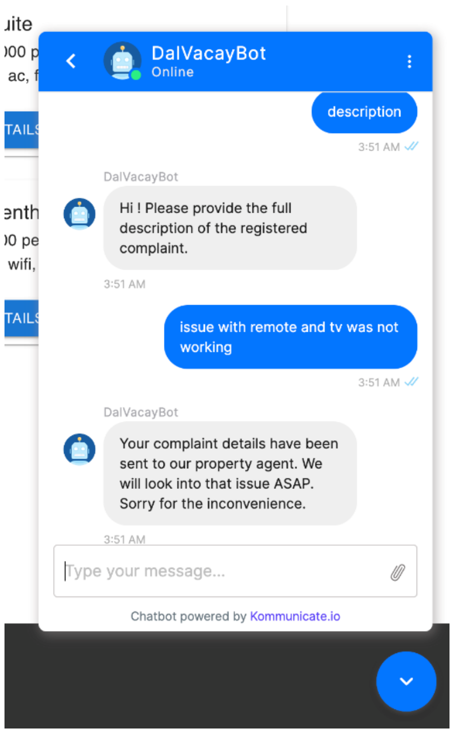
3:45 AM

DalVacayBot

Complaint Registered and One of the Property Agents will connect with you shortly. Also if you want to send the descriptive details of issue, please write /description and enter the full description details.

Figure 15: Complaint Intent

7. Describe Issue Intent



description

3:51 AM

DalVacayBot

Hi ! Please provide the full description of the registered complaint.

3:51 AM

issue with remote and tv was not working

3:51 AM

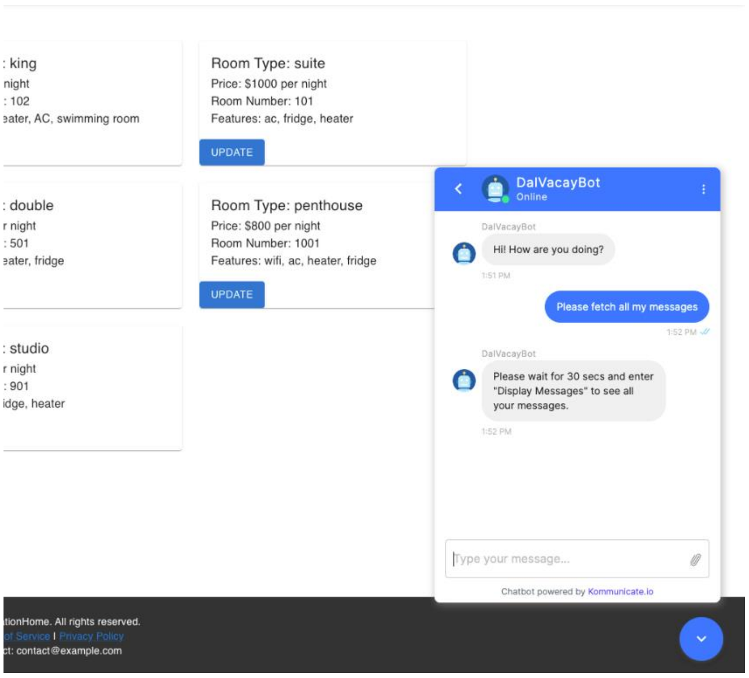
DalVacayBot

Your complaint details have been sent to our property agent. We will look into that issue ASAP. Sorry for the inconvenience.

3:51 AM

Figure 16: Describe Issue Intent

8. Fetch and display messages



: king
night
: 102
eater, AC, swimming room

Room Type: suite
Price: $1000 per night
Room Number: 101
Features: ac, fridge, heater

: double
r night
: 501
eater, fridge

Room Type: penthouse
Price: $800 per night
Room Number: 1001
Features: wifi, ac, heater, fridge

: studio
r night
: 901
idge, heater

DalVacayBot: Hi! How are you doing? 1:51 PM

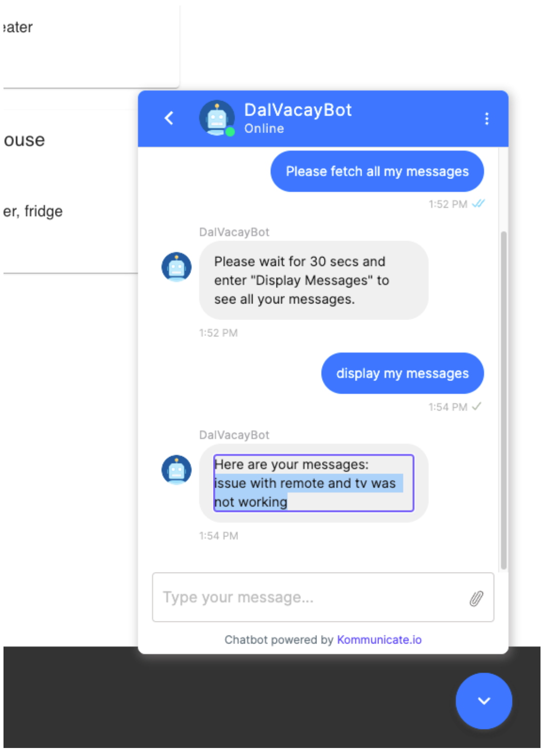
Please fetch all my messages 1:52 PM

DalVacayBot: Please wait for 30 secs and enter "Display Messages" to see all your messages. 1:52 PM

tionHome. All rights reserved.
of Service | Privacy Policy
ct: contact@example.com

Figure 17: Displaying messages

Display Messages



Please fetch all my messages 1:52 PM

DalVacayBot: Please wait for 30 secs and enter "Display Messages" to see all your messages. 1:52 PM

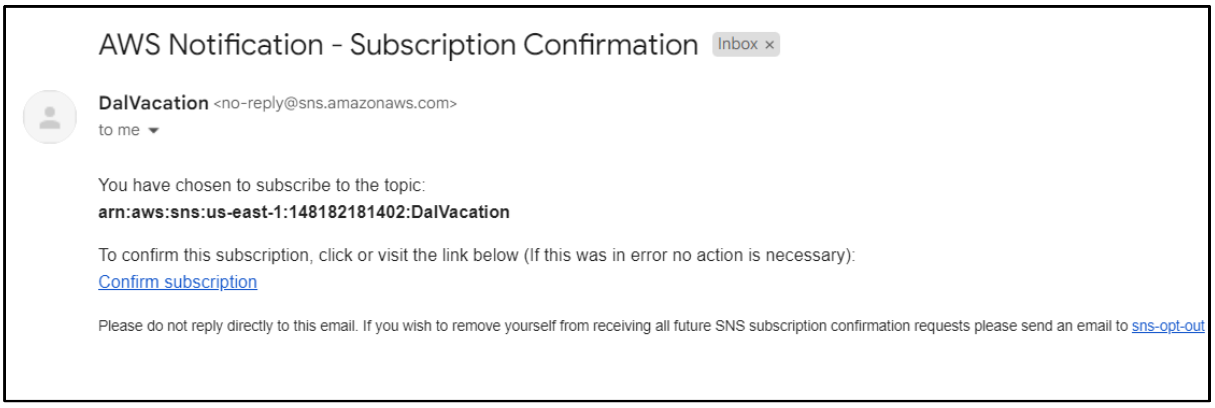
display my messages 1:54 PM

DalVacayBot: Here are your messages:
issue with remote and tv was not working
1:54 PM

Figure 18: Messages

Module 4: Notifications

Authentication:


AWS Notification - Subscription Confirmation Inbox ×

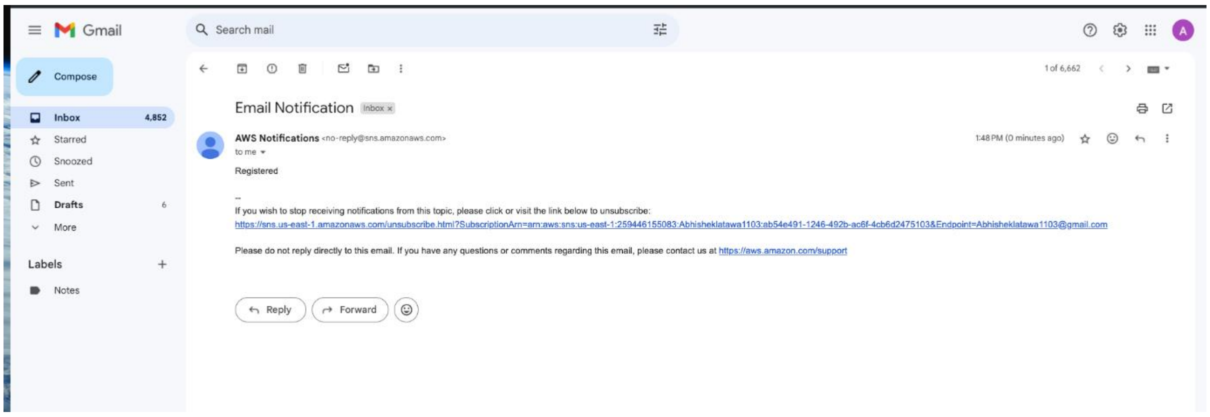
DalVacation <no-reply@sns.amazonaws.com>
to me

You have chosen to subscribe to the topic:
arn:aws:sns:us-east-1:148182181402:DalVacation

To confirm this subscription, click or visit the link below (If this was in error no action is necessary):
Confirm subscription

Please do not reply directly to this email. If you wish to remove yourself from receiving all future SNS subscription confirmation requests please send an email to sns-opt-out

Figure 19: Subscription notification



Email Notification Inbox ×

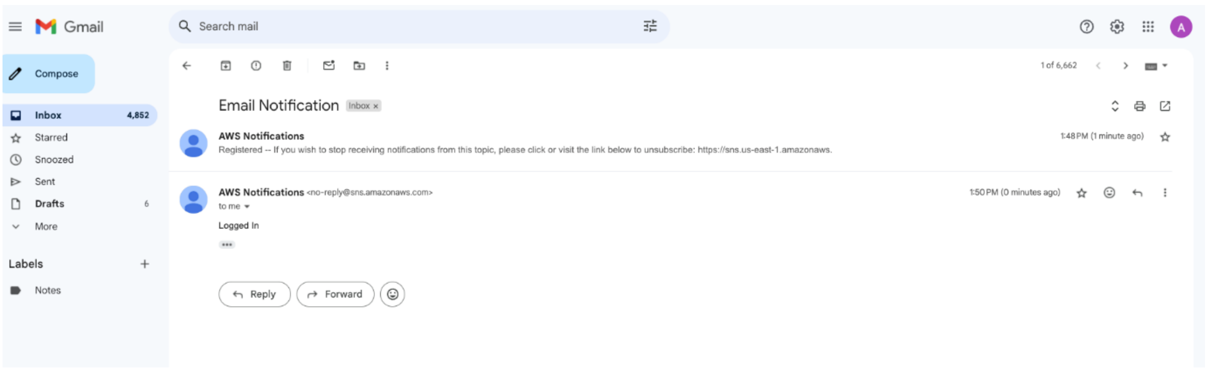
AWS Notifications <no-reply@sns.amazonaws.com>
to me
1:48 PM (0 minutes ago)

Registered

--
If you wish to stop receiving notifications from this topic, please click or visit the link below to unsubscribe:
https://sns.us-east-1.amazonaws.com/unsubscribe.html?SubscriptionArn=arn:aws:sns:us-east-1:259446155083:Abhisheklatawa1103:ab54e491-1246-492b-ac6f-4cb6d2475103&Endpoint=Abhisheklatawa1103@gmail.com

Please do not reply directly to this email. If you have any questions or comments regarding this email, please contact us at https://aws.amazon.com/support

Figure 20: Registration Notification



Email Notification Inbox ×

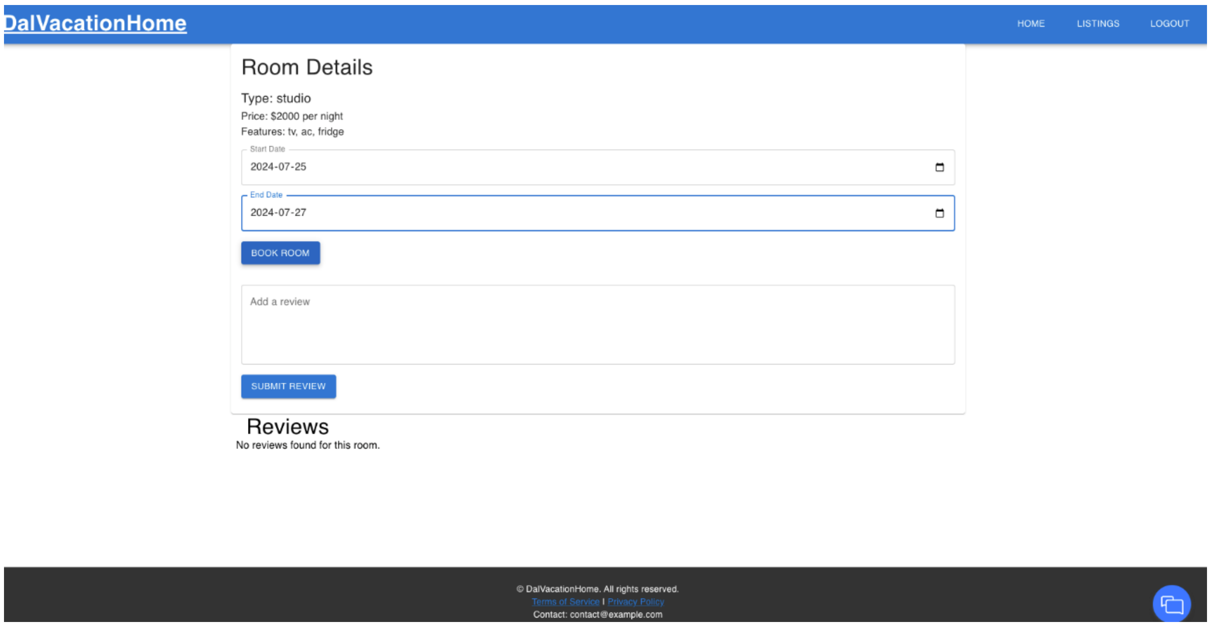
AWS Notifications
1:48 PM (1 minute ago)
Registered -- If you wish to stop receiving notifications from this topic, please click or visit the link below to unsubscribe: https://sns.us-east-1.amazonaws.

AWS Notifications <no-reply@sns.amazonaws.com>
to me
1:50 PM (0 minutes ago)

Logged In

Figure 21: Login Notification

Room Booking Success:


DalVacationHome

HOME LISTINGS LOGOUT

Room Details

Type: studio

Price: $2000 per night

Features: tv, ac, fridge

Start Date 2024-07-25

End Date 2024-07-27

BOOK ROOM

Add a review

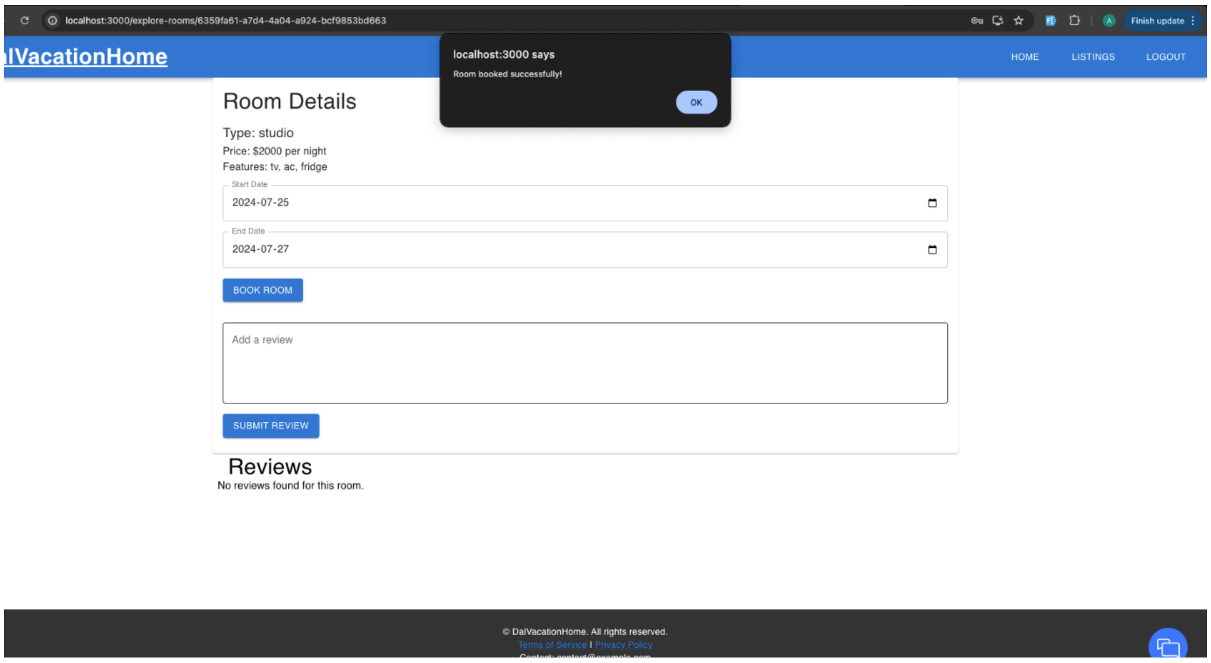
SUBMIT REVIEW

Reviews

No reviews found for this room.

© DalVacationHome. All rights reserved.
Terms of Service | Privacy Policy
Contact: contact@example.com

Figure 22: Room booking process


localhost:3000/explore-rooms/6359fa61-a7d4-4a04-a924-bcf9853bd663

localhost:3000 says

Room booked successfully!

OK

Room Details

Type: studio

Price: $2000 per night

Features: tv, ac, fridge

Start Date 2024-07-25

End Date 2024-07-27

BOOK ROOM

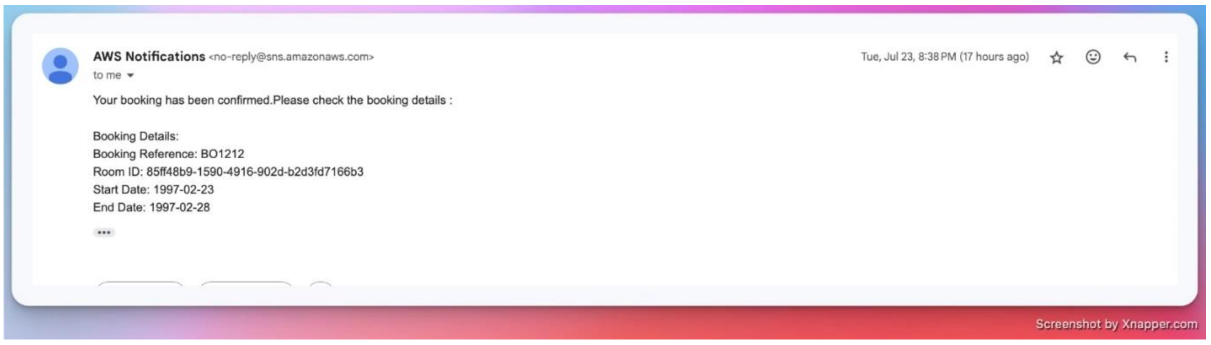
Add a review

SUBMIT REVIEW

Reviews

No reviews found for this room.

Figure 23: Successful room booking


AWS Notifications <no-reply@sns.amazonaws.com>
to me

Tue, Jul 23, 8:38 PM (17 hours ago)

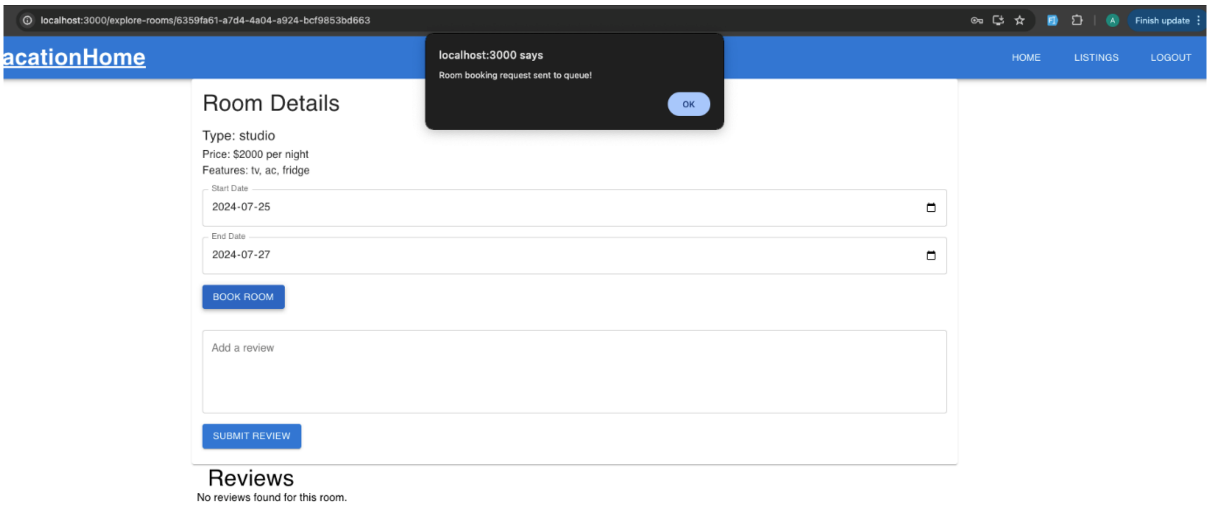
Your booking has been confirmed.Please check the booking details :

Booking Details:
Booking Reference: BO1212
Room ID: 85ff48b9-1590-4916-902d-b2d3fd7166b3
Start Date: 1997-02-23
End Date: 1997-02-28

Screenshot by Xnapper.com

Figure 24: Booking details confirmation

Room Booking Failure



localhost:3000 says
Room booking request sent to queue!

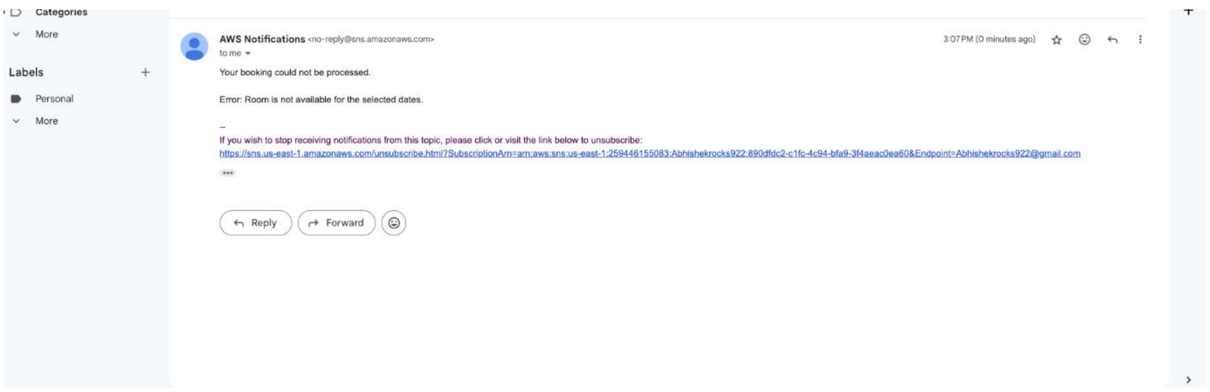
Room Details

Type: studio
Price: $2000 per night
Features: tv, ac, fridge

Start Date: 2024-07-25
End Date: 2024-07-27

Reviews
No reviews found for this room.

Figure 25: Room booking sent to queue



AWS Notifications <no-reply@sns.amazonaws.com>
to me
3:07 PM (0 minutes ago)

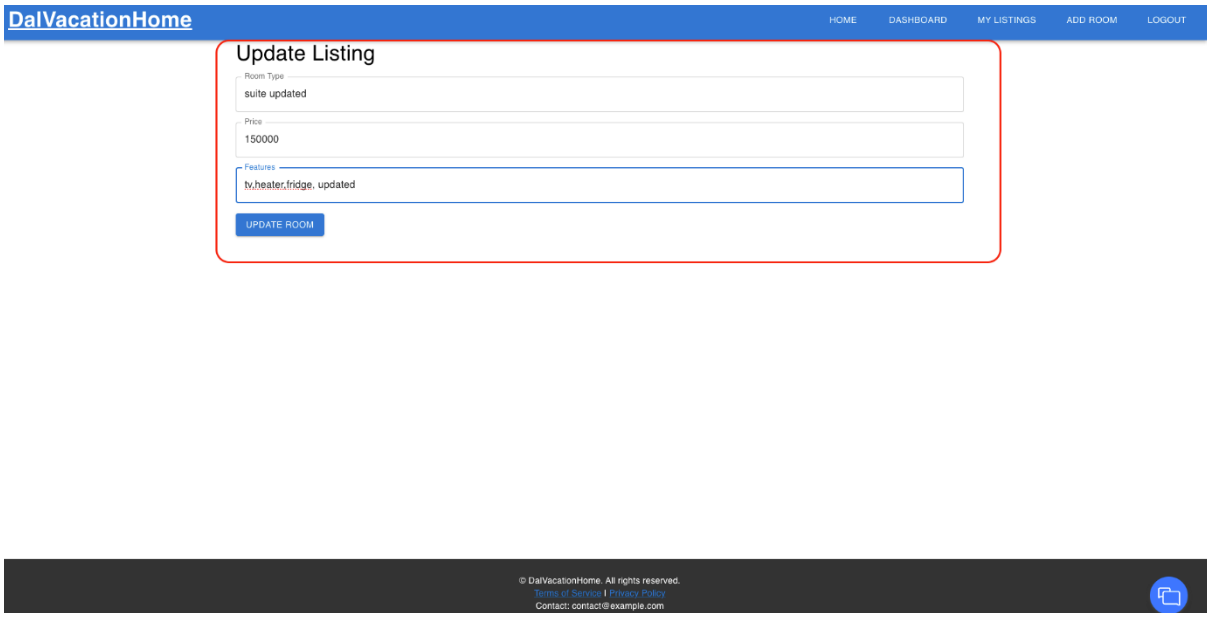
Your booking could not be processed.

Error: Room is not available for the selected dates.

--
If you wish to stop receiving notifications from this topic, please click or visit the link below to unsubscribe:
https://sns.us-east-1.amazonaws.com/unsubscribe.html?SubscriptionArn=arn:aws:sns:us-east-1:259446155083:Abhishekrocks922:890dfdc2-c1fc-4c94-bfa9-3f4aeac0ea60&Endpoint=Abhishekrocks922@gmail.com

Figure 26: Room booking failure notification

Update a Room:



Update Listing

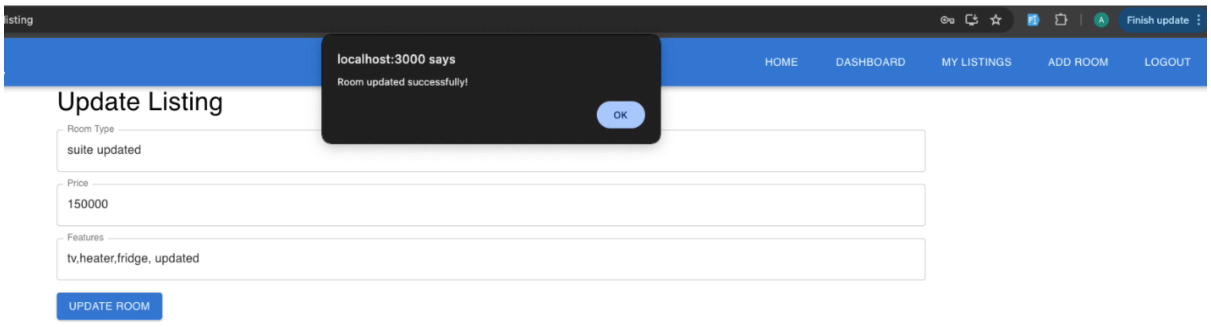
Room Type: suite updated

Price: 150000

Features: tv,heater,fridge, updated

© DalVacationHome. All rights reserved.
Terms of Service | Privacy Policy
Contact: contact@example.com

Figure 27: Update listing



localhost:3000 says

Room updated successfully!

Update Listing

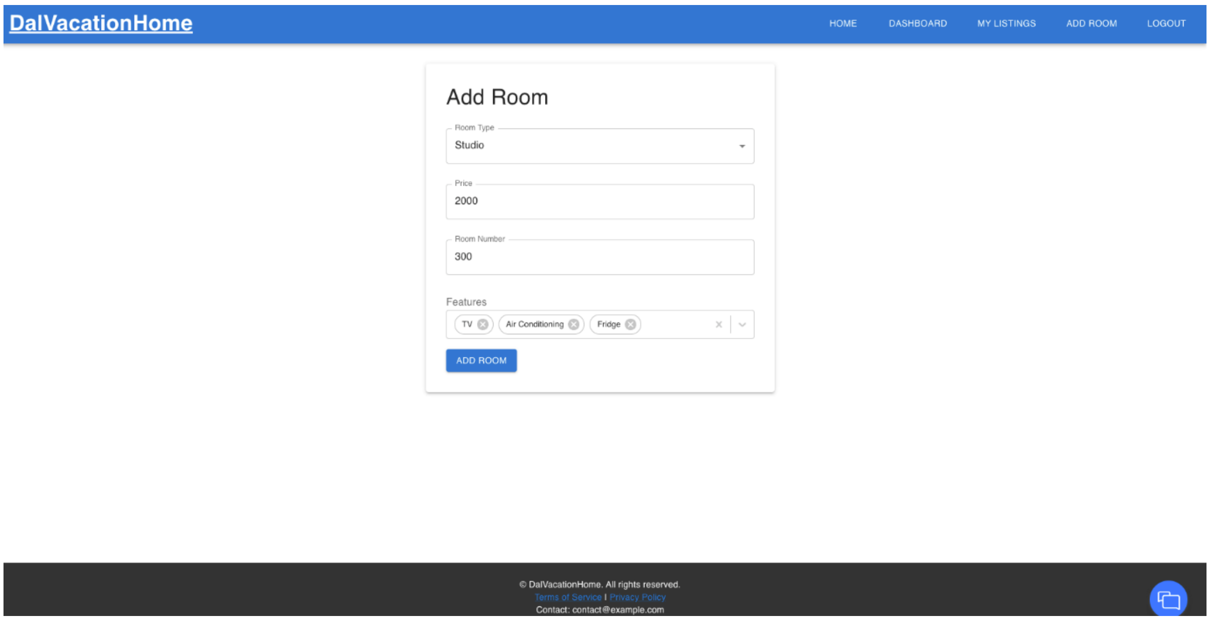
Room Type: suite updated

Price: 150000

Features: tv,heater,fridge, updated

Figure 28: Update room success

Add a Room:



DalVacationHome
HOME
DASHBOARD
MY LISTINGS
ADD ROOM
LOGOUT

Add Room

Room Type: Studio

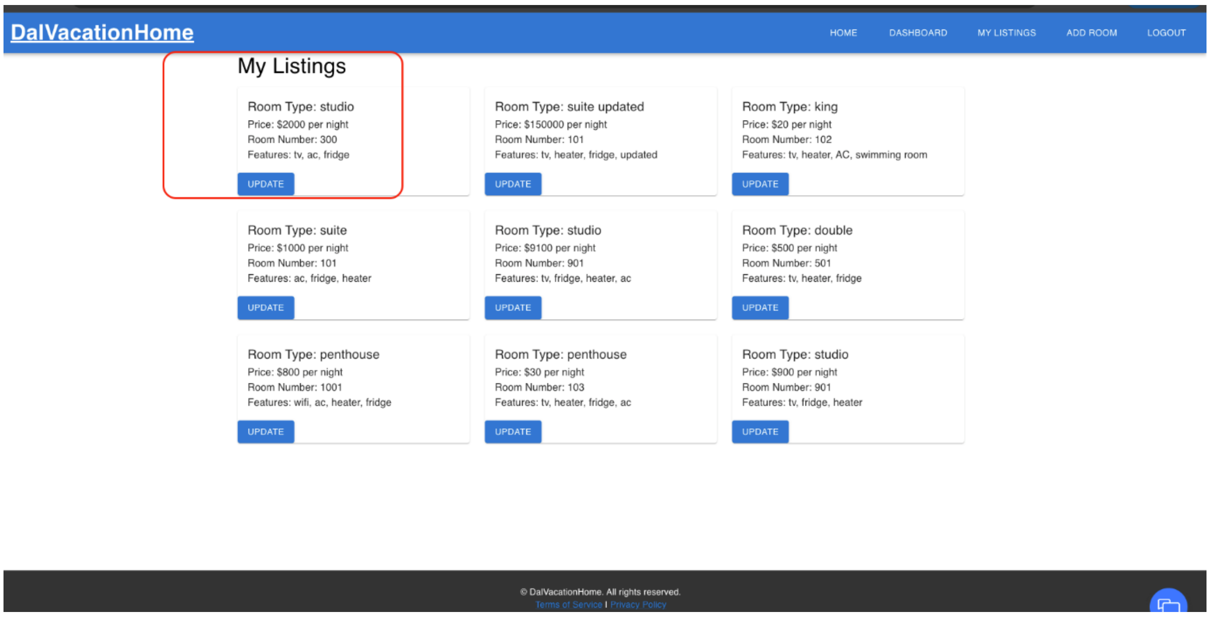
Price: 2000

Room Number: 300

Features: TV, Air Conditioning, Fridge

ADD ROOM
© DalVacationHome. All rights reserved.
Terms of Service | Privacy Policy
Contact: contact@example.com

Figure 29: Adding room



My Listings

Room Type: studio
Price: $2000 per night
Room Number: 300
Features: tv, ac, fridge

Room Type: suite updated
Price: $150000 per night
Room Number: 101
Features: tv, heater, fridge, updated

Room Type: king
Price: $20 per night
Room Number: 102
Features: tv, heater, AC, swimming room

Room Type: suite
Price: $1000 per night
Room Number: 101
Features: ac, fridge, heater

Room Type: studio
Price: $9100 per night
Room Number: 901
Features: tv, fridge, heater, ac

Room Type: double
Price: $500 per night
Room Number: 501
Features: tv, heater, fridge

Room Type: penthouse
Price: $800 per night
Room Number: 1001
Features: wifi, ac, heater, fridge

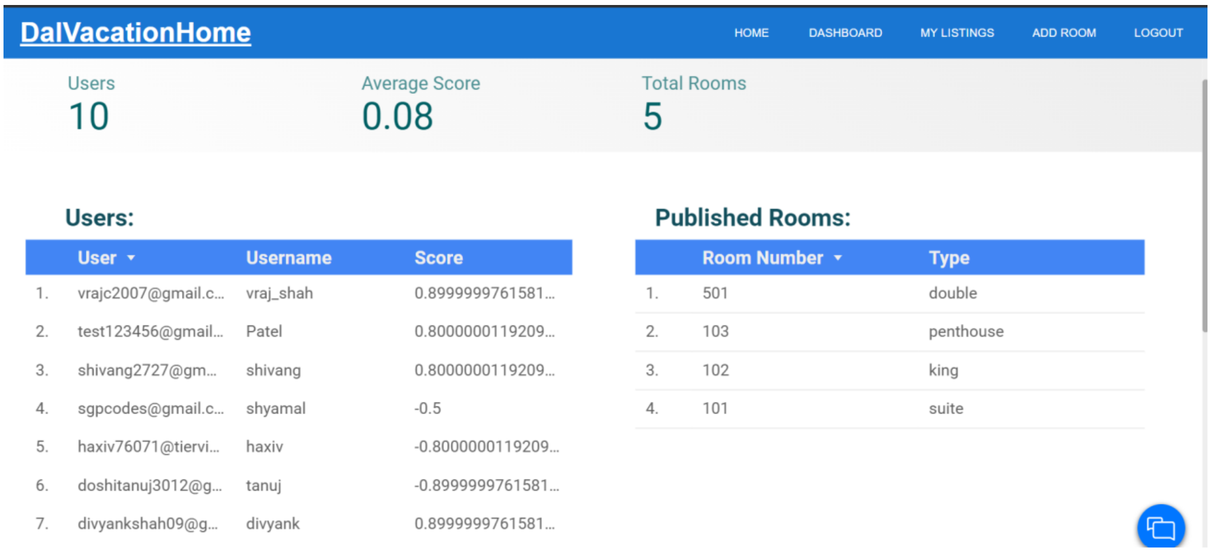
Room Type: penthouse
Price: $30 per night
Room Number: 103
Features: tv, heater, fridge, ac

Room Type: studio
Price: $900 per night
Room Number: 901
Features: tv, fridge, heater

UPDATE
© DalVacationHome. All rights reserved.
Terms of Service | Privacy Policy

Figure 30: Room added success

Module 5: Data analysis and Visualization



Users 10 | Average Score 0.08 | Total Rooms 5

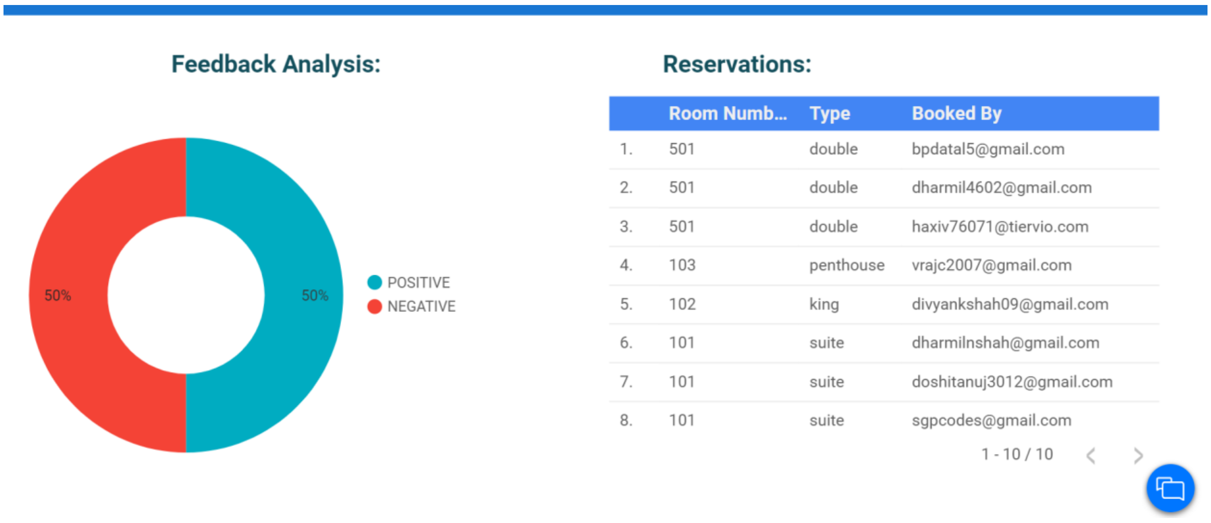
Users:

	User	Username	Score
1.	vrajc2007@gmail.c...	vraj_shah	0.8999999761581...
2.	test123456@gmail...	Patel	0.8000000119209...
3.	shivang2727@gm...	shivang	0.8000000119209...
4.	sgpcodes@gmail.c...	shyamal	-0.5
5.	haxiv76071@tiervi...	haxiv	-0.8000000119209...
6.	doshitanuj3012@g...	tanuj	-0.8999999761581...
7.	divyankshah09@g...	divyank	0.8999999761581...

Published Rooms:

	Room Number	Type
1.	501	double
2.	103	penthouse
3.	102	king
4.	101	suite

Figure 31: Dashboard 1



Reservations:

	Room Numb...	Type	Booked By
1.	501	double	bpdatal5@gmail.com
2.	501	double	dharmil4602@gmail.com
3.	501	double	haxiv76071@tiervio.com
4.	103	penthouse	vrajc2007@gmail.com
5.	102	king	divyankshah09@gmail.com
6.	101	suite	dharmilnshah@gmail.com
7.	101	suite	doshitanuj3012@gmail.com
8.	101	suite	sgpcodes@gmail.com

Figure 32: Dashboard 2

Individual Contribution:

Abhisek Latawa:

- Developed Lambda functions for the Adding, Updating, Deleting and fetching Room as a Property Agent
- Developed Lambda for the Google DialogFlow response
- Developed Lambdas for make a booking for the available rooms when logged in as a registered user.
- Worked with the SQS to develop a lambda which takes the request from the client to make Room booking request
- Worked with the SNS service and developed lambda to send the successful and failure notification to the subscribed users.
- Developed the Landing page for the Guest users in which they can see the room details.

- Worked on Google DialogFlow ChatBot
- Integrated Chatbot with frontend application and handled the user queries through chatbot

Bhishman Desai:

- Developed a comprehensive AWS CloudFormation script to automate the deployment of an Amazon Lex chatbot.
- Ensured the script included all necessary AWS resources such as Lambda functions, and IAM roles configurations.
- Designed and implemented an automated deployment pipeline for a React frontend application using Google Cloud Run.
- Utilized Terraform to manage and provision Google Cloud resources, ensuring an efficient and scalable infrastructure setup.
- Developed a robust message passing module to facilitate communication (synchronous and asynchronous) between Dialog Flow and Lamba/Cloud functions using Google Cloud Pub/Sub using Terraform.
- Implemented cross-cloud integration to enable seamless data flow and interoperability between different AWS and GCP.

Shivang Patel:

- Implemented user authentication using AWS Cognito, integrating it seamlessly with the application's frontend and backend.
- Configured Multi-Factor Authentication (MFA) to enhance security, ensuring users verify their identity using a second factor.
- Implemented encryption using Caesar Cipher for secure transmission and storage of sensitive user data and for third factor authentication.
- Worked extensively with AWS Simple Notification Service (SNS) to send notifications to users.
- Integrated SNS with Lambda functions to automate notifications based on user actions and application events.
- Configured SNS topics and subscriptions to manage and organize notification delivery efficiently.

Dharmil Shah:

- Researched on Looker Studio for dashboard implementation for displaying reviews, total users.
- Researched on Google Natural Language API.
- Designed and integrated Frontend and backend of Authentication module.
- Designed room booking frontend.
- Designed and Implemented backend of Looker studio and Google Natural Language API and integrated it in Frontend through iframe.
- Added reviews system for each room.

Meeting Logs:

Serverless DP Team-3 Meeting Logs				
Meeting Date	**Meeting Agenda**	**Minutes of Meeting**	**Attendance**	**Meeting Recordings**
18-07-2024	Discussed the updates on the divided modules.	1. Discussed about the errors and resolve them 2. Worked on the code quality 3. Worked on the individual tasks and discussed the approach	All members present	https://dalu-my.sharepoint.com/:v:/g/personal/dh939824_dal_ca/EfEVATRKpBdGiMY4vUHVFf0BqrQMZ8TDy8vEpr-y-ipOjA?referrer=Teams.TEAMS-ELECTRON&referrerScenario=MeetingChicletGetLink.view
22-07-2024	Completion of the modules	1. Discussed on the completed modules 2. Resolved errors that were faced during integration	All members present	https://dalu-my.sharepoint.com/:v:/g/personal/dh939824_dal_ca/EY3BX1i5dU9Bg22vHhEoiTEBgwUpAFxn3wTjQ8jpWpsMSg?referrer=Teams.TEAMS-ELECTRON&referrerScenario=MeetingChicletGetLink.view
23-06-2024	Final Discussion for Sprint 3	1. Combining individual tasks 2. Finalising report and meeting logs	All members present	https://dalu-my.sharepoint.com/:v:/g/personal/dh939824_dal_ca/EUkHy1rT7wBJjRti2aEP0yUBt4w0FvEJtiVH5m6XhmCgwA?referrer=Teams.TEAMS-ELECTRON&referrerScenario=MeetingChicletGetLink.view

Gitlab Code:

- https://git.cs.dal.ca/latawa/csci5410-s24-sdp-3

References:

[1] "Dialogflow ES basics," *Google Cloud.* https://cloud.google.com/dialogflow/es/docs/basics [Accessed: July 22, 2024]

[2] "What is Pub/Sub? | Pub/Sub Documentation | Google Cloud," Google Cloud. [Online]. Available: https://cloud.google.com/pubsub/docs/overview [Accessed: July 22, 2024]

[3] "What is Amazon SNS? - Amazon Simple Notification Service," Amazon Web Services, Inc. [Online]. Available: https://docs.aws.amazon.com/sns/latest/dg/welcome.html [Accessed: July 22, 2024].

[4] "Push Notification Service - Amazon Simple Notification Service - AWS," Amazon Web Services, Inc. [Online]. Available: https://aws.amazon.com/sns/. [Accessed: July 22, 2024].

[5] "Working with Amazon SQS messages - Amazon Simple Queue Service," Amazon Web Services, Inc. [Online]. Available: https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/working-with-messages.html. [Accessed: July 22, 2024].

[6] "Using Amazon SQS dead-letter queues to replay messages," Amazon Web Services, Inc. [Online]. Available: https://aws.amazon.com/blogs/compute/using-amazon-sqs-dead-letter-queues-to-replay-messages/. [Accessed: July 22, 2024].

[7] "Looker Studio: Business Insights," Google Cloud. [Online]. https://cloud.google.com/looker-studio?hl=en [Accessed: July 22, 2024].

[8] "Cloud Natural Language," Google Cloud. [Online]. https://cloud.google.com/natural-language?hl=en [Accessed: July 22, 2024].

[9] "What is Cloud Run | Cloud Run Documentation | Google Cloud," Google Cloud. [Online]. Available: https://cloud.google.com/run/docs/overview/what-is-cloud-run [Accessed: July 22, 2024].

[10] "Docs Overview," Terraform. [Online]. https://registry.terraform.io/providers/hashicorp/google/latest/docs [Accessed: July 22, 2024].

[11] "Documents & diagrams for engineering teams," Eraser.io. [Online]. Available: https://www.eraser.io/ [Accessed: July 22, 2024].