IT3122 Cloud Application Development Project (25%)

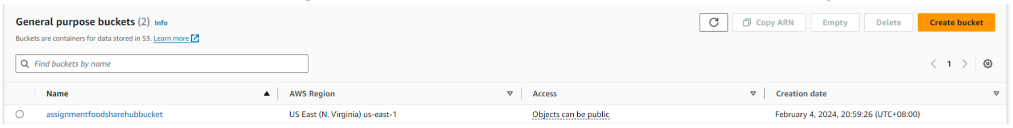
Name: Liu JiaJun
Admin No: 211283E
Module Group: IT3122-01
Credentials to your AWS account: AWS Academy Learner Lab [69324]
[Original Theme]
In this project, you will be developing a lost-and-found website for NYP using AWS Cloud Services. This website will display all items that have been returned to the lost-and-found department.

[My Theme]
For this project, I am creating a website called "FoodShareHub" specifically for the NYP community, utilizing AWS Cloud Services. The purpose of the site is to facilitate the donation of unwanted food items among students, allowing them to share with others who may need or want them. The website will showcase all the available donated food items. If a student finds something they are interested in, they can simply visit the FoodShareHub booth to collect the item.

Part 1 – Setup and host the website on S3 Bucket (7%)

Create a website that allows visitors to the site to display all available donated items. You should design your website to be user friendly and easy to navigate.

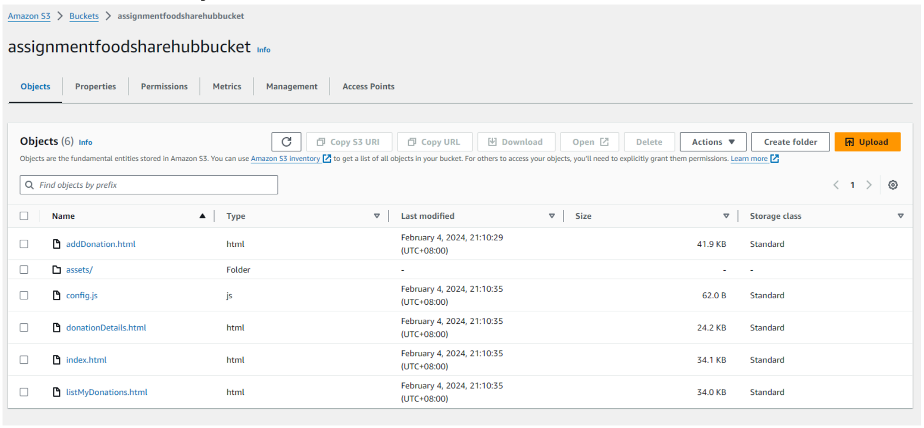
(a) Screenshot your website – listing the various pages that you created
1. Screenshots of the S3 Bucket created and objects inside
   a. S3 Bucket named “assignmentfoodsharehubbucket” successfully created



General purpose buckets (2) Info
Buckets are containers for data stored in S3. Learn more

Name	AWS Region	Access	Creation date
assignmentfoodsharehubbucket	US East (N. Virginia) us-east-1	Objects can be public	February 4, 2024, 20:59:26 (UTC+08:00)

   b. Here are the objects stored in the S3 Bucket

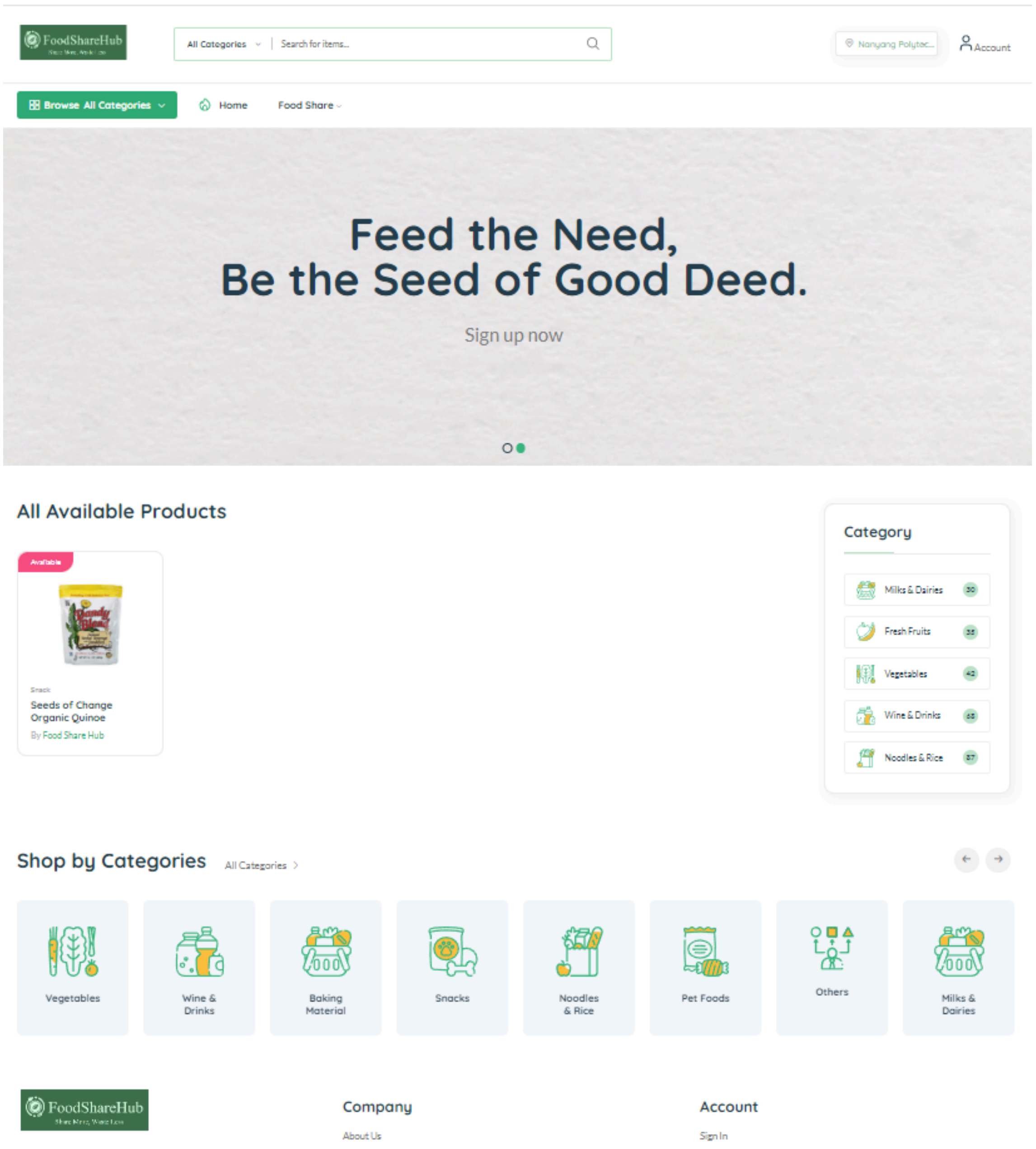


assignmentfoodsharehubbucket Info

Objects (6) Info
Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

Name	Type	Last modified	Size	Storage class
addDonation.html	html	February 4, 2024, 21:10:29 (UTC+08:00)	41.9 KB	Standard
assets/	Folder	-	-	-
config.js	js	February 4, 2024, 21:10:35 (UTC+08:00)	62.0 B	Standard
donationDetails.html	html	February 4, 2024, 21:10:35 (UTC+08:00)	24.2 KB	Standard
index.html	html	February 4, 2024, 21:10:35 (UTC+08:00)	34.1 KB	Standard
listMyDonations.html	html	February 4, 2024, 21:10:35 (UTC+08:00)	34.0 KB	Standard

2. List All Available Items Page (Home Page)



All Categories | Search for items...

Nanyang Polytec... Account

Browse All Categories | Home | Food Share

Feed the Need,
Be the Seed of Good Deed.

Sign up now

All Available Products

Available
Snack
Seeds of Change Organic Quinoa
By Food Share Hub

Category

Milks & Dairies	30
Fresh Fruits	35
Vegetables	42
Wine & Drinks	68
Noodles & Rice	87

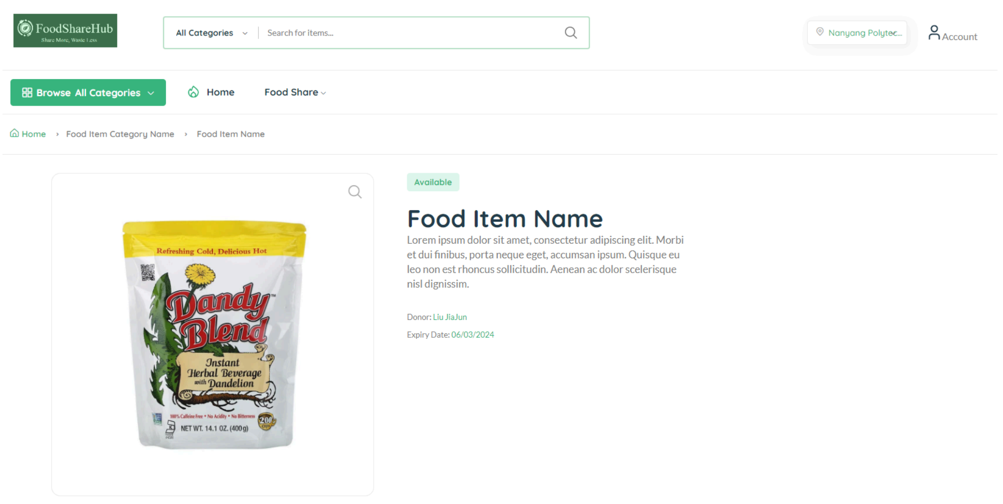
Shop by Categories All Categories

Vegetables | Wine & Drinks | Baking Material | Snacks | Noodles & Rice | Pet Foods | Others | Milks & Dairies

Company
About Us

Account
Sign In

3. Donation Details Items Page



Home › Food Item Category Name › Food Item Name

Available

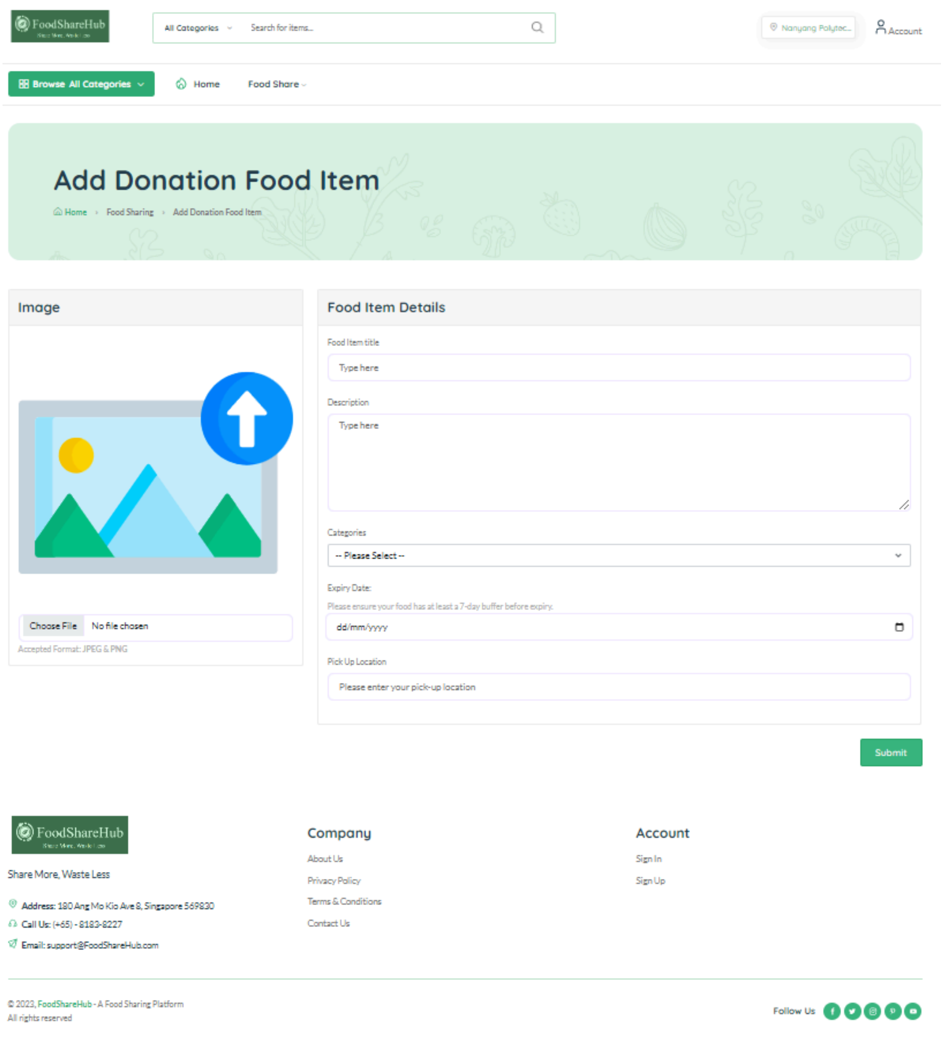
Food Item Name

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Morbi et dui finibus, porta neque eget, accumsan ipsum. Quisque eu leo non est rhoncus sollicitudin. Aenean ac dolor scelerisque nisl dignissim.

Donor: Liu JiaJun

Expiry Date: 06/03/2024

4. Add Donation Page



Add Donation Food Item

Home › Food Sharing › Add Donation Food Item

Image

Choose File No file chosen

Accepted Format: JPEG & PNG

Food Item Details

Food Item title: Type here

Description: Type here

Categories: -- Please Select --

Expiry Date:
Please ensure your food has at least a 7-day buffer before expiry.
dd/mm/yyyy

Pick Up Location: Please enter your pick-up location

Submit

FoodShareHub

Share More, Waste Less

Address: 180 Ang Mo Kio Ave 8, Singapore 569830
Call Us: (+65) - 8183-8227
Email: support@FoodShareHub.com

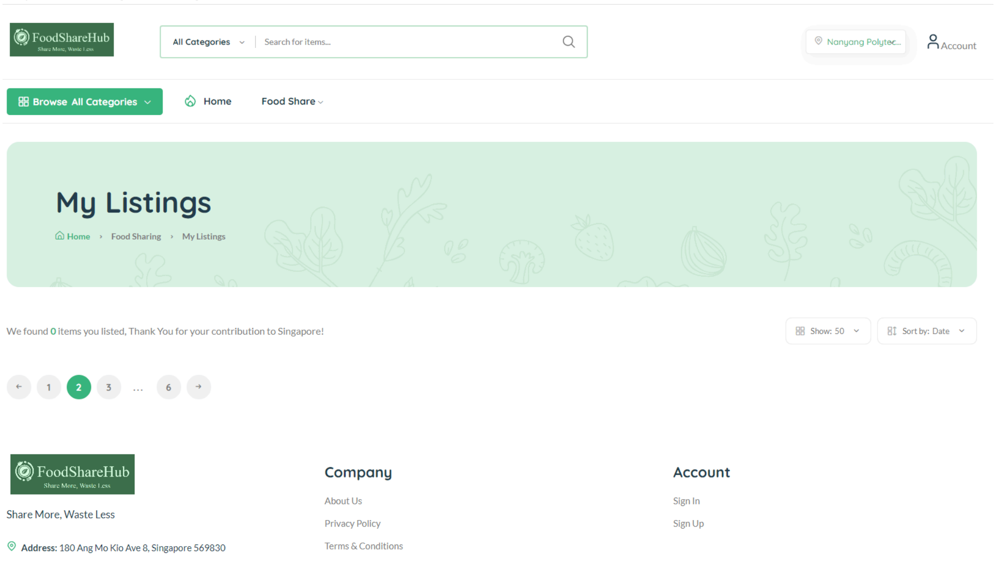
Company
About Us
Privacy Policy
Terms & Conditions
Contact Us

Account
Sign In
Sign Up

© 2023, FoodShareHub - A Food Sharing Platform
All rights reserved

Follow Us

5. My Listings Page



My Listings

We found 0 items you listed, Thank You for your contribution to Singapore!

Share More, Waste Less

Address: 180 Ang Mo Kio Ave 8, Singapore 569830

(b) Zip up all the source code for website (label Part 1)

Remarks: In Step1, Changes made including:

1. Host the "FoodShareHub" on S3 Bucket
2. API Gateway Mock Endpoint are created, and used to render the available FoodItem Categories
3. Index Page Object URL: https://assignmentfoodsharehubbucket.s3.amazonaws.com/index.html
4. The others API Gateway Endpoint will be implemented later at Part 2

Part 2 – Add a food item and list it on website (8%)

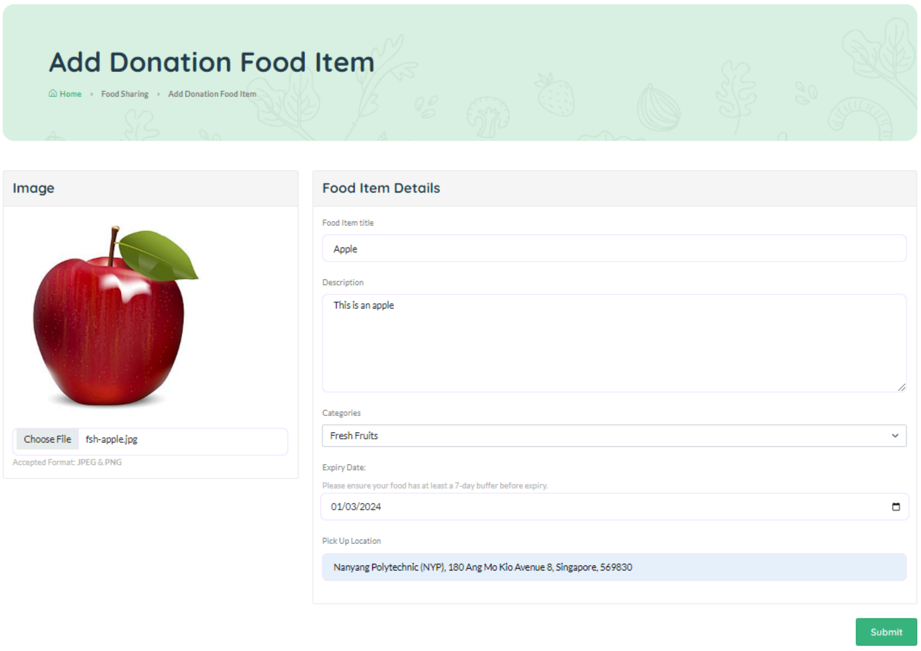
When donors want to donate something, they bring it to the FoodShareHub counter and these food items will be added to the website. As more and more items were donated, the FoodShareHub decided to design and streamline the process. New items can be added to the site, by uploading images of the food items with the appropriate descriptions. A database is now needed to maintain and categorize these items for easy searching and listing. Design and develop your improved website.

(c) Screenshot your website – listing the various pages that you created and showing all the features of the improved website

Changes includes:

1. Lambda Functions allow user to upload food item with image
2. Display the items uploaded on the HomePage
3. RDS MySQL to make data storage persistent and relational
4. Data Validation when user uploads a fooditem
5. A new S3 Bucket to store zip files of lambda code

1. Create a donation



Add Donation Food Item

Home › Food Sharing › Add Donation Food Item

Image

Choose File fsh-apple.jpg

Accepted Format: JPEG & PNG

Food Item Details

Food Item title: Apple

Description: This is an apple

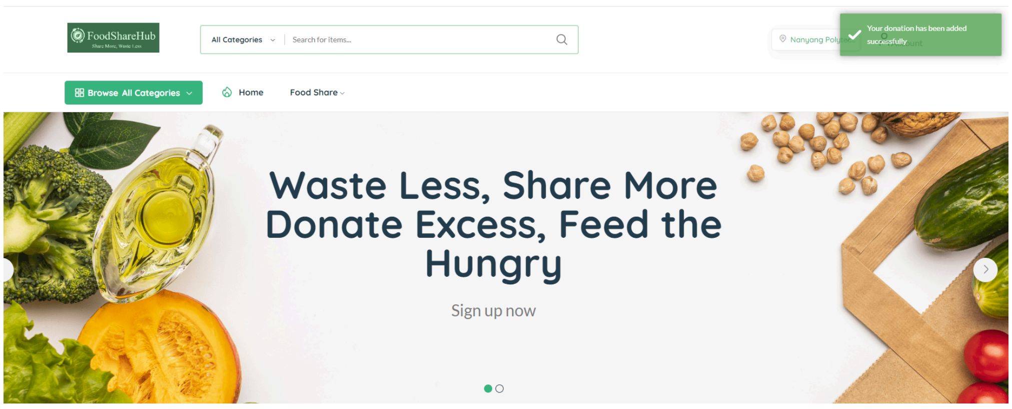
Categories: Fresh Fruits

Expiry Date:
Please ensure your food has at least a 7-day buffer before expiry.
01/03/2024

Pick Up Location: Nanyang Polytechnic (NYP), 180 Ang Mo Kio Avenue 8, Singapore, 569830

Submit

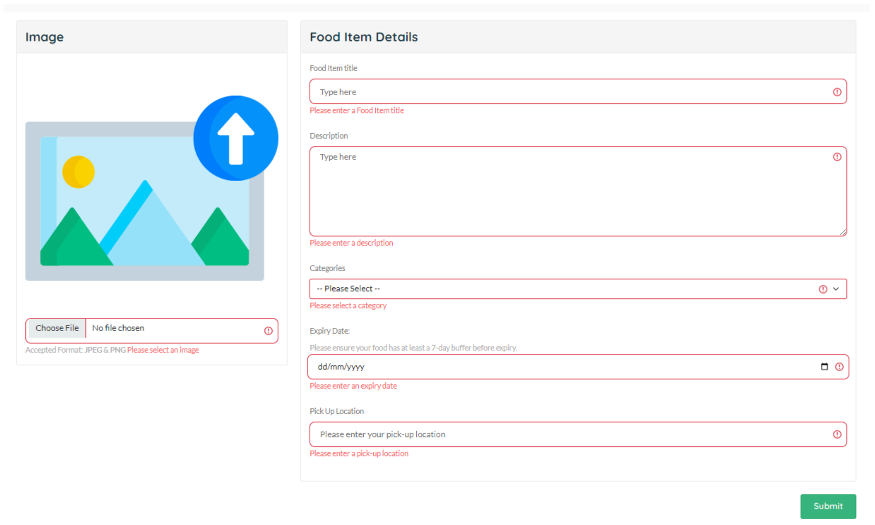
2. Redirect to index.html and displays a notification at the top right



Your donation has been added successfully

Waste Less, Share More Donate Excess, Feed the Hungry

3. Data validation



Image

Choose File No file chosen

Accepted Format: JPEG & PNG Please select an image

Food Item Details

Food Item title: Type here
Please enter a Food Item title

Description: Type here
Please enter a description

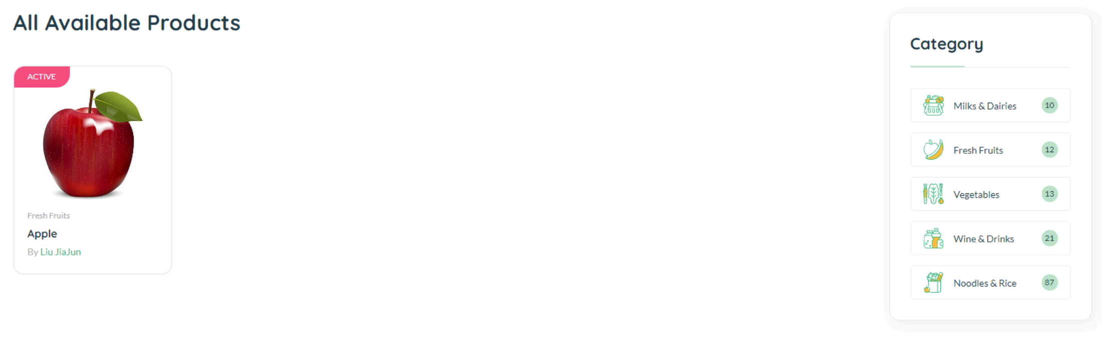
Categories: -- Please Select --
Please select a category

Expiry Date:
Please ensure your food has at least a 7-day buffer before expiry.
dd/mm/yyyy
Please enter an expiry date

Pick Up Location: Please enter your pick-up location
Please enter a pick-up location

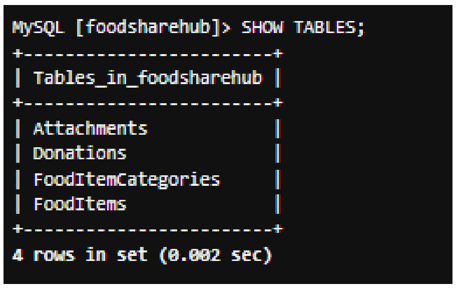
Submit

4. Display the created food item on the index.html



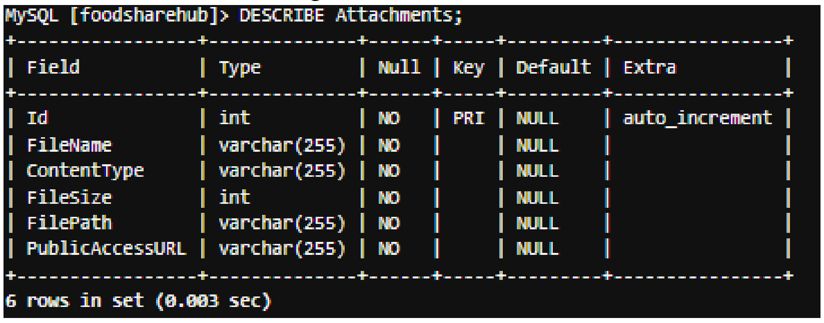
(d) Screenshot of the database design (you may use SQL or NoSQL)

1. Tables



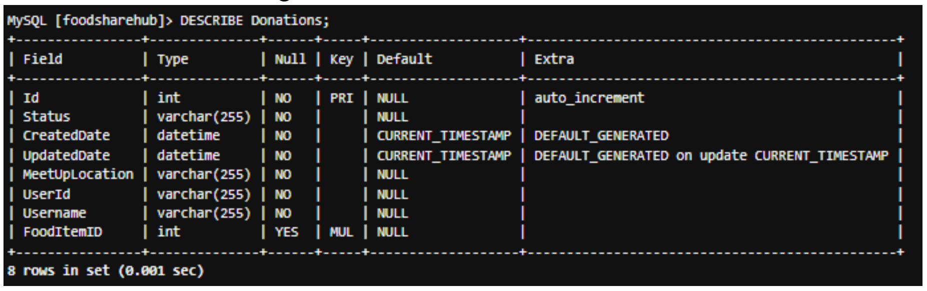
```
MySQL [foodsharehub]> SHOW TABLES;
+-----------------------+
| Tables_in_foodsharehub |
+-----------------------+
| Attachments           |
| Donations             |
| FoodItemCategories    |
| FoodItems             |
+-----------------------+
4 rows in set (0.002 sec)
```

2. Attachments Table Design



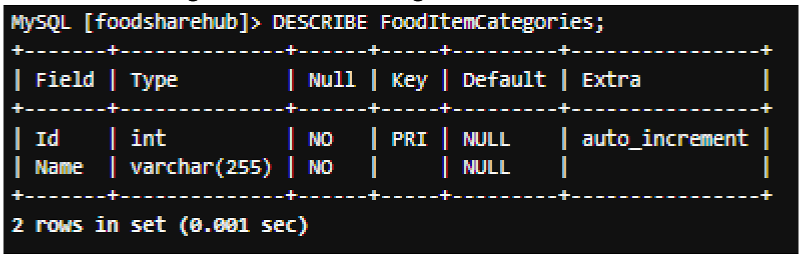
```
MySQL [foodsharehub]> DESCRIBE Attachments;
+-----------------+--------------+------+-----+---------+----------------+
| Field           | Type         | Null | Key | Default | Extra          |
+-----------------+--------------+------+-----+---------+----------------+
| Id              | int          | NO   | PRI | NULL    | auto_increment |
| FileName        | varchar(255) | NO   |     | NULL    |                |
| ContentType     | varchar(255) | NO   |     | NULL    |                |
| FileSize        | int          | NO   |     | NULL    |                |
| FilePath        | varchar(255) | NO   |     | NULL    |                |
| PublicAccessURL | varchar(255) | NO   |     | NULL    |                |
+-----------------+--------------+------+-----+---------+----------------+
6 rows in set (0.003 sec)
```

3. Donations Table Design



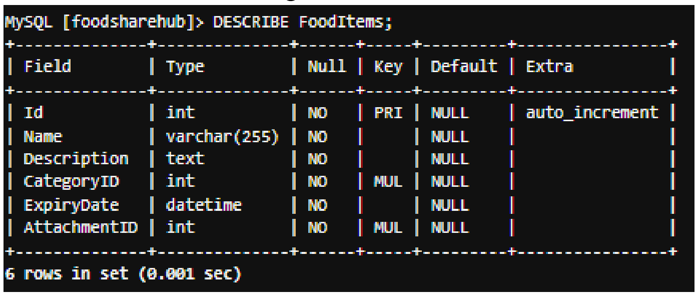
```
MySQL [foodsharehub]> DESCRIBE Donations;
+----------------+--------------+------+-----+-------------------+-----------------------------------------------+
| Field          | Type         | Null | Key | Default           | Extra                                         |
+----------------+--------------+------+-----+-------------------+-----------------------------------------------+
| Id             | int          | NO   | PRI | NULL              | auto_increment                                |
| Status         | varchar(255) | NO   |     | NULL              |                                               |
| CreatedDate    | datetime     | NO   |     | CURRENT_TIMESTAMP | DEFAULT_GENERATED                             |
| UpdatedDate    | datetime     | NO   |     | CURRENT_TIMESTAMP | DEFAULT_GENERATED on update CURRENT_TIMESTAMP |
| MeetUpLocation | varchar(255) | NO   |     | NULL              |                                               |
| UserId         | varchar(255) | NO   |     | NULL              |                                               |
| Username       | varchar(255) | NO   |     | NULL              |                                               |
| FoodItemID     | int          | YES  | MUL | NULL              |                                               |
+----------------+--------------+------+-----+-------------------+-----------------------------------------------+
8 rows in set (0.001 sec)
```

4. FoodItemCategories Table Design



```
MySQL [foodsharehub]> DESCRIBE FoodItemCategories;
+-------+--------------+------+-----+---------+----------------+
| Field | Type         | Null | Key | Default | Extra          |
+-------+--------------+------+-----+---------+----------------+
| Id    | int          | NO   | PRI | NULL    | auto_increment |
| Name  | varchar(255) | NO   |     | NULL    |                |
+-------+--------------+------+-----+---------+----------------+
2 rows in set (0.001 sec)
```

5. FoodItems Table Design



```
MySQL [foodsharehub]> DESCRIBE FoodItems;
+--------------+--------------+------+-----+---------+----------------+
| Field        | Type         | Null | Key | Default | Extra          |
+--------------+--------------+------+-----+---------+----------------+
| Id           | int          | NO   | PRI | NULL    | auto_increment |
| Name         | varchar(255) | NO   |     | NULL    |                |
| Description  | text         | NO   |     | NULL    |                |
| CategoryID   | int          | NO   | MUL | NULL    |                |
| ExpiryDate   | datetime     | NO   |     | NULL    |                |
| AttachmentID | int          | NO   | MUL | NULL    |                |
+--------------+--------------+------+-----+---------+----------------+
6 rows in set (0.001 sec)
```

(e) Write a brief explanation on the choice of database used and the design.

My Choice: RDS (MySQL)

Reason:

1. Structured Data and Relationships:
   Relational databases are ideal for handling structured data and relationships between different data entities. For FoodShareHub, where you have clear relationships between donations, categories, and items, a relational database can manage these relations efficiently through foreign keys and joins.

2. ACID Transactions:
   Relational databases provide ACID (Atomicity, Consistency, Isolation, Durability) transactions which are critical for ensuring data integrity, especially in applications like FoodShareHub where transactions may involve multiple tables (e.g., updating a donation record and the associated food item record).

3. Schema Enforcement:
   RDS enforces a schema, which means the database structure is defined and controlled, ensuring data consistency and making it easier to enforce data integrity rules. This can be particularly important when dealing with a variety of food items and categories where certain data fields are mandatory.

Database Design Explanation:

My RDS instance includes four main tables:

1. Attachments:
   Stores metadata for images and other files associated with food items. Separating attachments from food items allows for flexible management and retrieval of media content.

2. Donations:
   Tracks each donation record, including donor information, the donated food items, and the status of the donation. This table can be used to monitor the lifecycle of donations from receipt to distribution.

3. FoodItemCategories:
   Provides a categorization system for food items, enabling users to browse and search for food items based on categories. This is helpful for enhancing user experience and navigation on the site.

4. FoodItems:

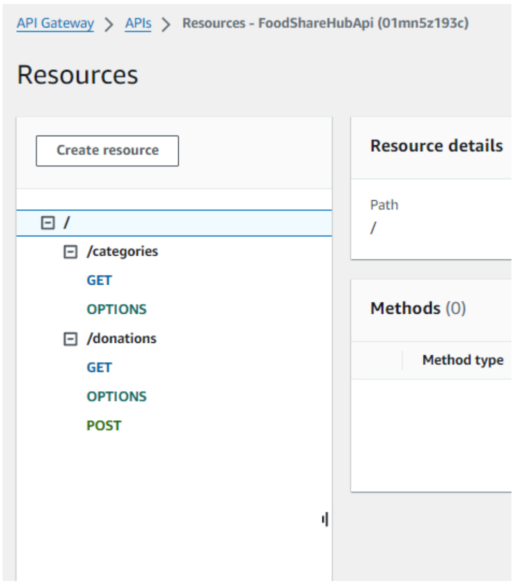
Contains detailed information about food items such as the name, description, category, and any related attachments. This is a central part of the site, offering users information about what donated items are available.

By using RDS and this design, the FoodShareHub website can efficiently handle and display donated food items, allowing both users and staff to easily manage and access this information. The design emphasizes organization and consistency of data while maintaining scalability and ease of use for the database.

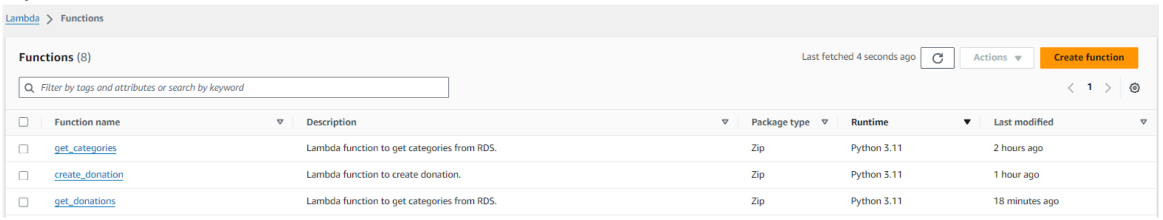
(f) Zip up all the source code for website and database (label Part 2)

Remarks:

1. To learn more about RDS table attributes, check out the "create_tables.sql" file in the AWS folder.
2. Regarding MyListings Page and DonationDetail Page, I did not make any changes to it as I focused on Retrieve and Create only.
3. My API Gateway so far



4. My Lambda Functions


Lambda > Functions
Functions (8)
Last fetched 4 seconds ago
Actions
Create function
Filter by tags and attributes or search by keyword

Function name	Description	Package type	Runtime	Last modified
get_categories	Lambda function to get categories from RDS.	Zip	Python 3.11	2 hours ago
create_donation	Lambda function to create donation.	Zip	Python 3.11	1 hour ago
get_donations	Lambda function to get categories from RDS.	Zip	Python 3.11	18 minutes ago

   1. Create Donation
   2. Get Donations
   3. Get Categories

Part 3 (5%)

As the site grew, more and more items were donated and collected. The team decided to make it easier for users to find food items. Users can create notifications whenever new items are listed on the site or when an item that matches the category of item is uploaded to the site. The team was also exploring other features to improve user experience and automation (eg using AI to recognize items etc)

Design and utilize any AWS service to achieve this feature.

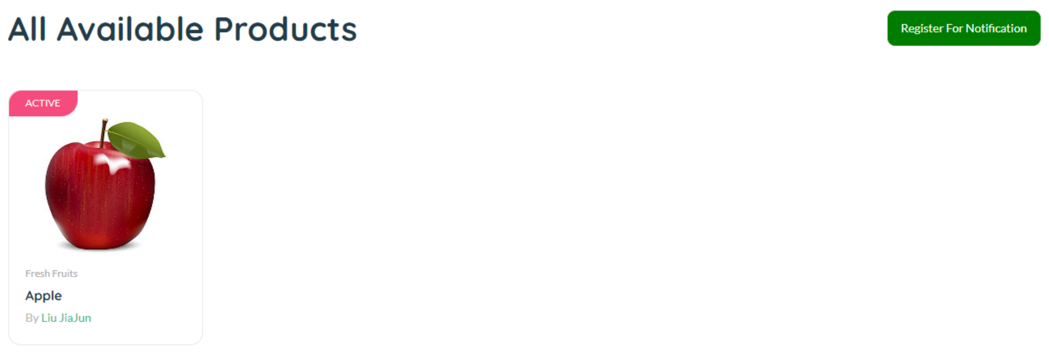
(g) Screenshot the setting of your improvements and the respective settings. Show evidence that these features are working.

Improvement includes:

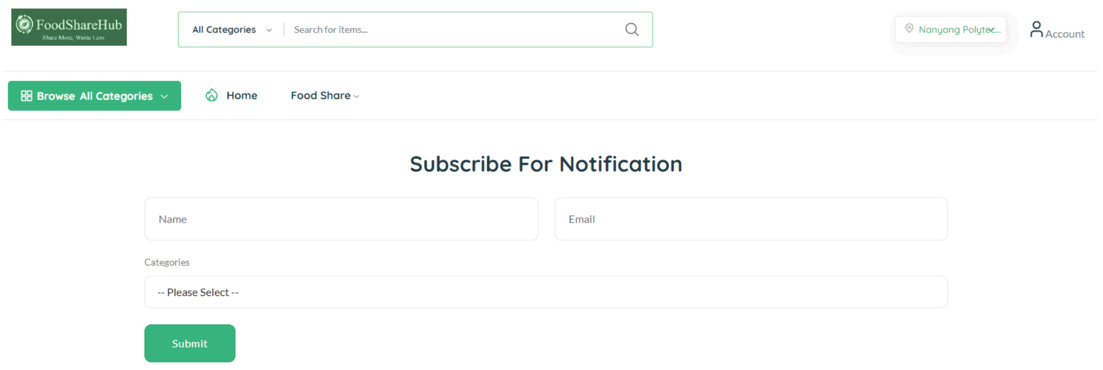
1. Create a web page that allows users to select their preferred categories
2. Use Amazon Simple Notification Service (SNS) to send emails to users when a new item matching their selected category is uploaded.
3. Use Amazon Recognition to moderate and detect the food item image and auto filling the web form.

Simple Notification Service:

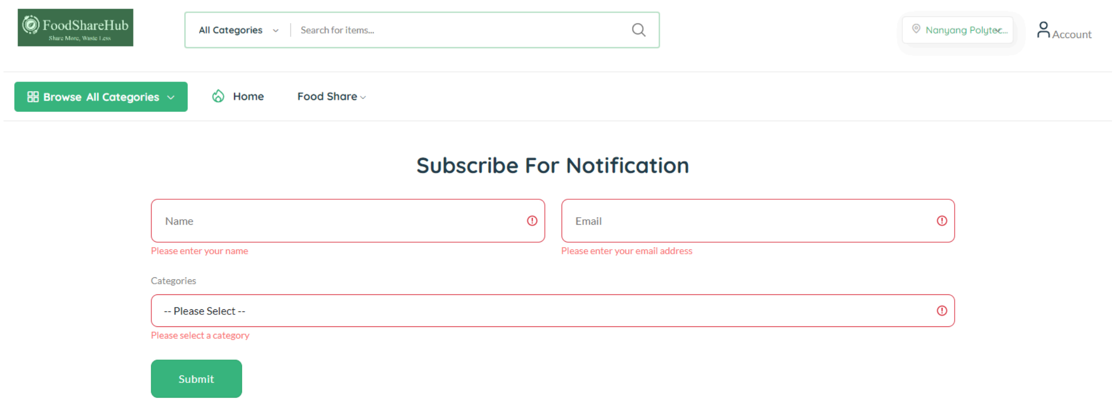
1. Click register for notification on the index.html



2. Register for notification on registerNotification.html
   The subscription form includes following field:
   a. Name
   b. Email
   c. Category



Validation for empty fields



Subscribe For Notification

Name

Please enter your name

Email

Please enter your email address

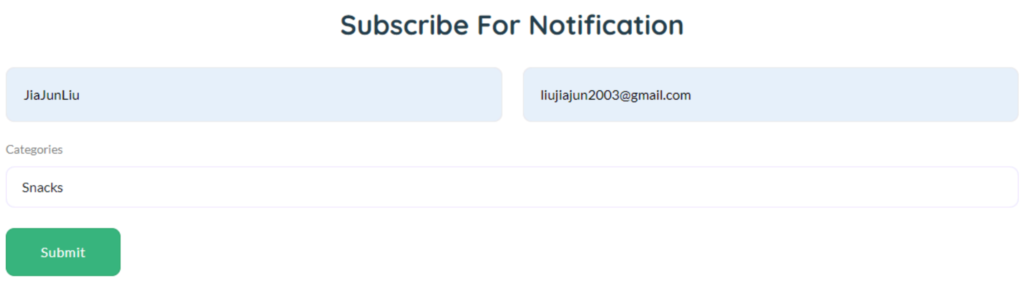
Categories

-- Please Select --

Please select a category

Submit

Upon successful submission of the form, users will be redirected to the index.html page, where a success message will be displayed in the top right corner. SNS Subscription Confirmation will be send to user if the user have not subscribed yet



Subscribe For Notification

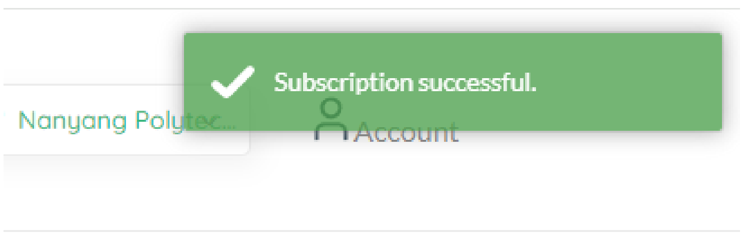
JiaJunLiu

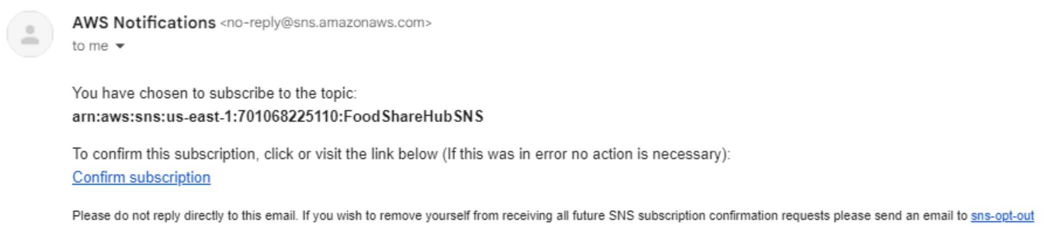
liujiajun2003@gmail.com

Categories

Snacks

Submit



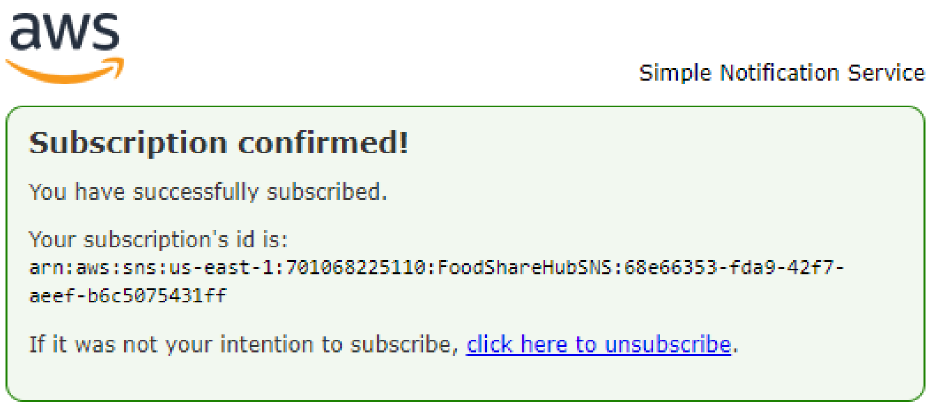


AWS Notifications <no-reply@sns.amazonaws.com>
to me

You have chosen to subscribe to the topic:
arn:aws:sns:us-east-1:701068225110:FoodShareHubSNS

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



aws

Simple Notification Service

Subscription confirmed!

You have successfully subscribed.

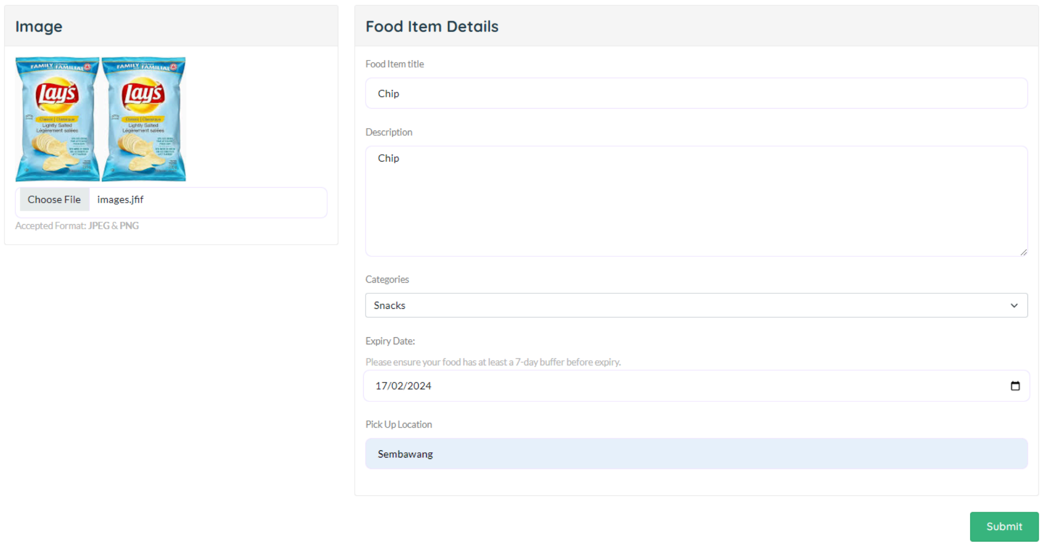
Your subscription's id is:
`arn:aws:sns:us-east-1:701068225110:FoodShareHubSNS:68e66353-fda9-42f7-aeef-b6c5075431ff`

If it was not your intention to subscribe, click here to unsubscribe.

Remarks: If the user wishes to modify their preferred category, they have the option to resubmit the form selecting a new category.

3. Once a new food item is added, and if it matches the user's selected category, an email will be sent to them.

   It has been observed that the user is registered for the "Snacks" category. Consequently, I will proceed to include a new food item within the "Snacks" category.



Image

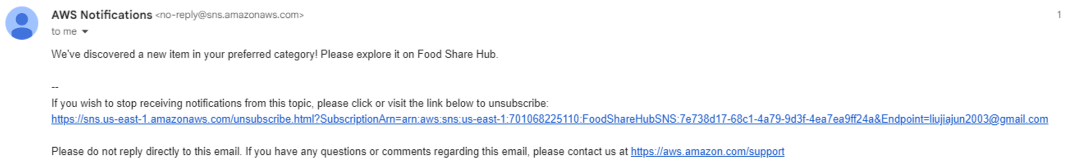
Choose File Images.jfif

Accepted Format: JPEG & PNG

Food Item Details

Food Item title	Chip
Description	Chip
Categories	Snacks
Expiry Date: Please ensure your food has at least a 7-day buffer before expiry.	17/02/2024
Pick Up Location	Sembawang

Submit



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

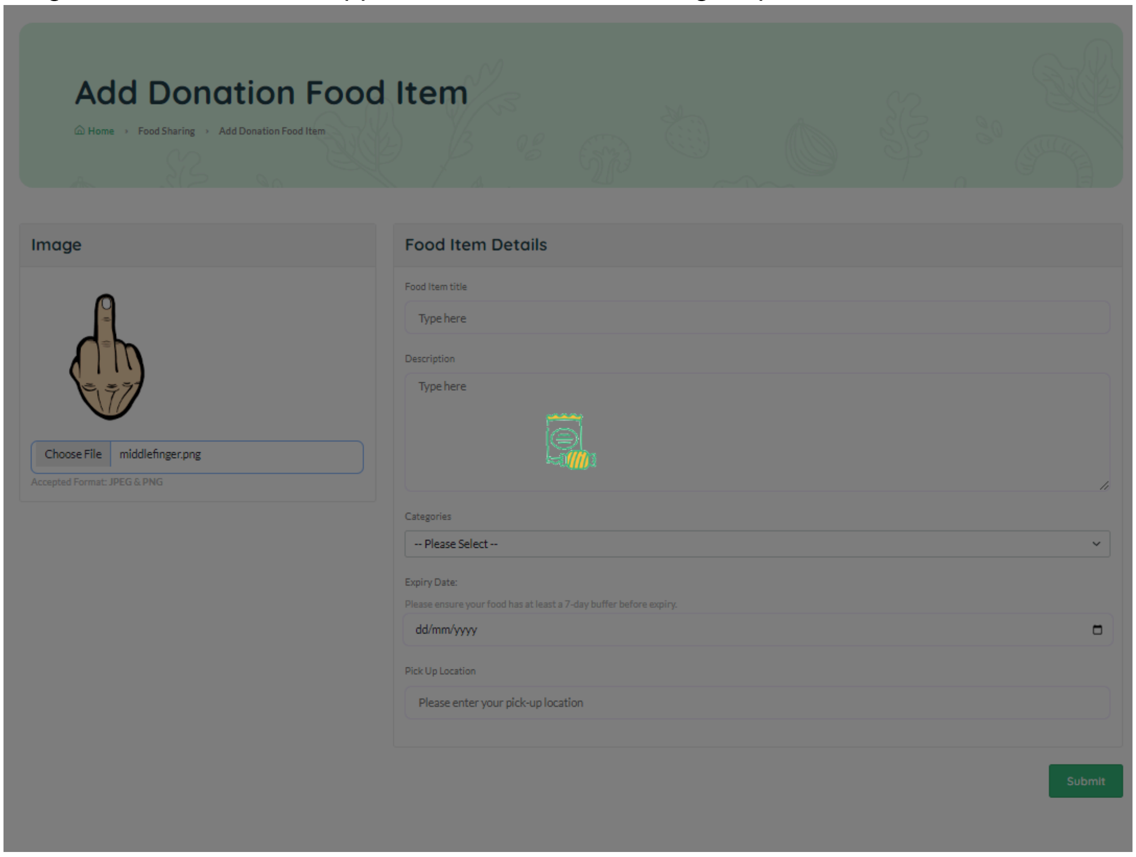
We've discovered a new item in your preferred category! Please explore it on Food Share Hub.

--
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-1:701068225110:FoodShareHubSNS:7e738d17-68c1-4a79-9d3f-4ea7ea9ff24a&Endpoint=liujiajun2003@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

Amazon Recognition:

Uploading a "middle finger" image will result in a warning. And once an inappropriate image detected, the web application will reset the image input



Add Donation Food Item

Home · Food Sharing · Add Donation Food Item

Image

Choose File middlefinger.png

Accepted Format: JPEG & PNG

Food Item Details

Food Item title: Type here

Description: Type here

Categories: -- Please Select --

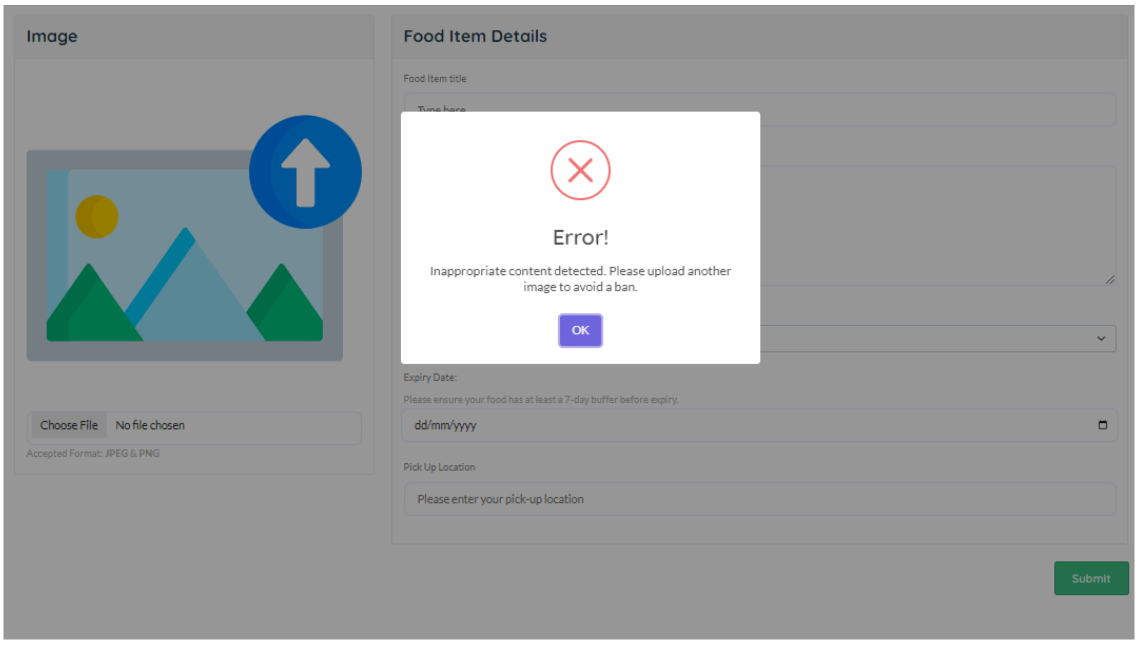
Expiry Date:

Please ensure your food has at least a 7-day buffer before expiry.

dd/mm/yyyy

Pick Up Location: Please enter your pick-up location

Submit



Image

Choose File No file chosen

Accepted Format: JPEG & PNG

Food Item Details

Error!

Inappropriate content detected. Please upload another image to avoid a ban.

OK

Expiry Date:

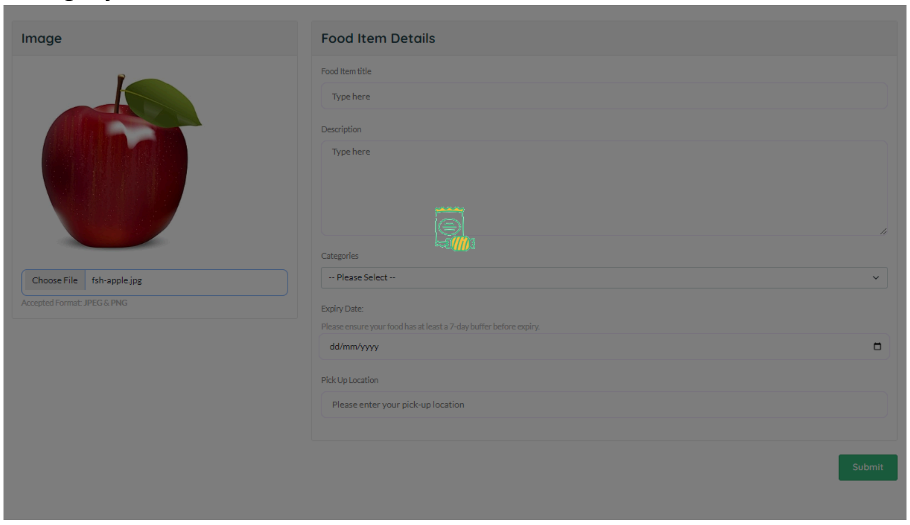
Please ensure your food has at least a 7-day buffer before expiry.

dd/mm/yyyy

Pick Up Location: Please enter your pick-up location

Submit

Upload an Apple image, and the online application will automatically fill in the name and categories depending on the recognition returned labels.
In this case:
Name: Apple
Category: Fresh Fruits



Image

Choose File fsh-apple.jpg

Accepted Format: JPEG & PNG

Food Item Details

Food Item title: Type here

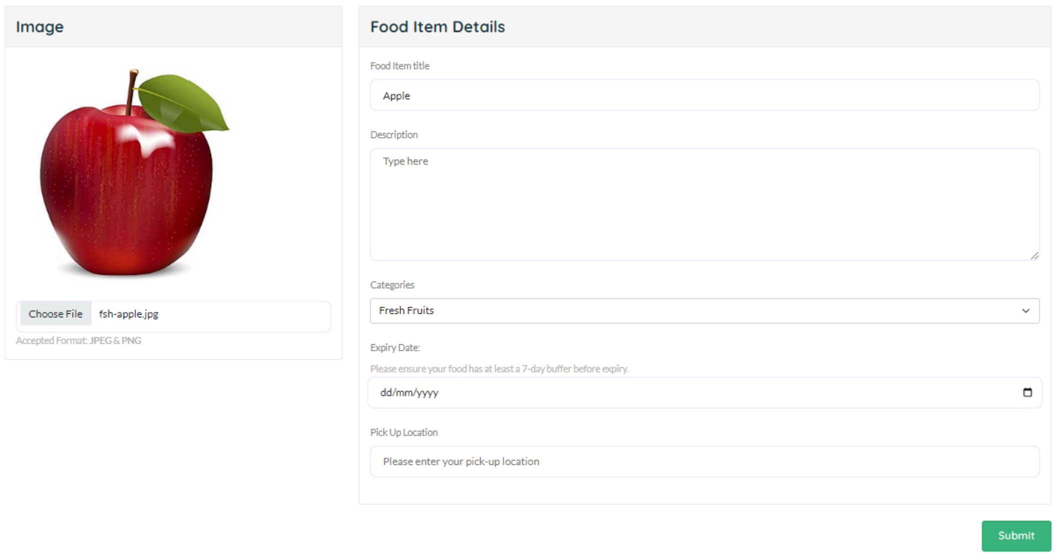
Description: Type here

Categories: -- Please Select --

Expiry Date:
Please ensure your food has at least a 7-day buffer before expiry.
dd/mm/yyyy

Pick Up Location: Please enter your pick-up location

Submit



Image

Choose File fsh-apple.jpg

Accepted Format: JPEG & PNG

Food Item Details

Food Item title: Apple

Description: Type here

Categories: Fresh Fruits

Expiry Date:
Please ensure your food has at least a 7-day buffer before expiry.
dd/mm/yyyy

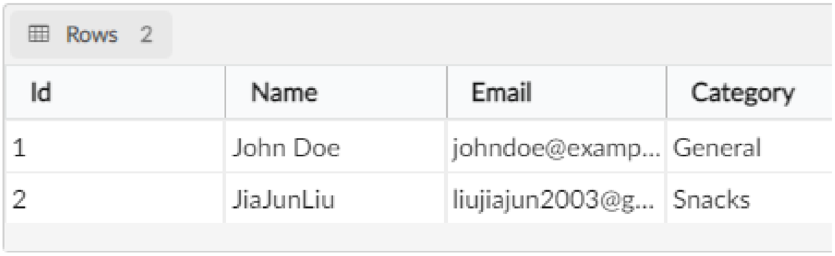
Pick Up Location: Please enter your pick-up location

Submit

(h) Document your design and steps with the necessary descriptions and screenshots.

Use Amazon SNS to alert users via email when a new item in their chosen category is listed.

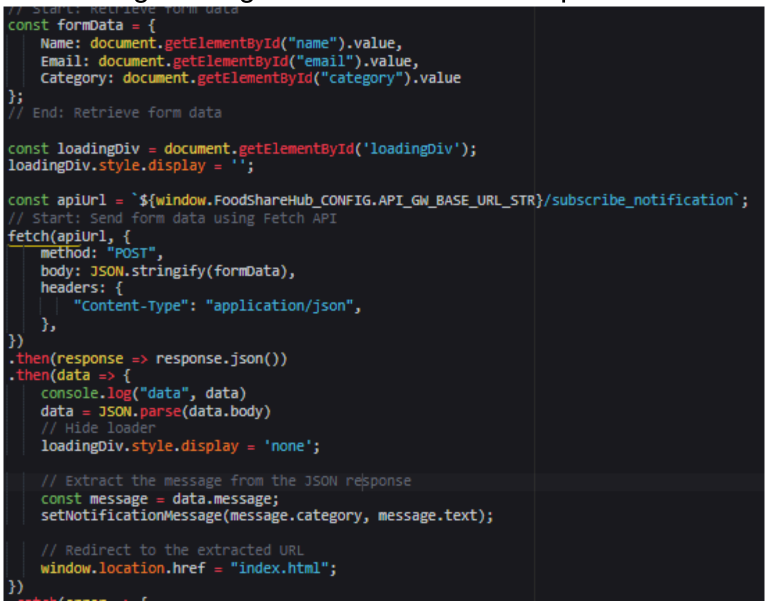
1. Create “NotificationSubscribers” Table



Rows 2

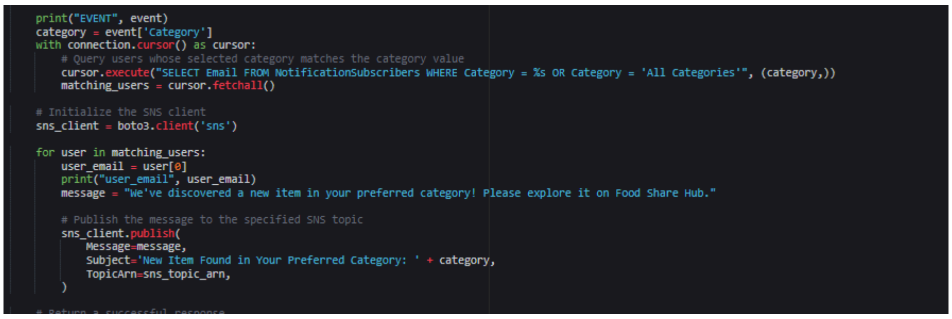
Id	Name	Email	Category
1	John Doe	johndoe@examp...	General
2	JiaJunLiu	liujiajun2003@g...	Snacks

2. Add a new “registerNotification.html”page allowing user to register for notification
3. Create the “subscribe_notification_code.py” and “subscribe_notification_wrapper.py” to create lambda function
4. Edit the register registerNotification.html to pass form data



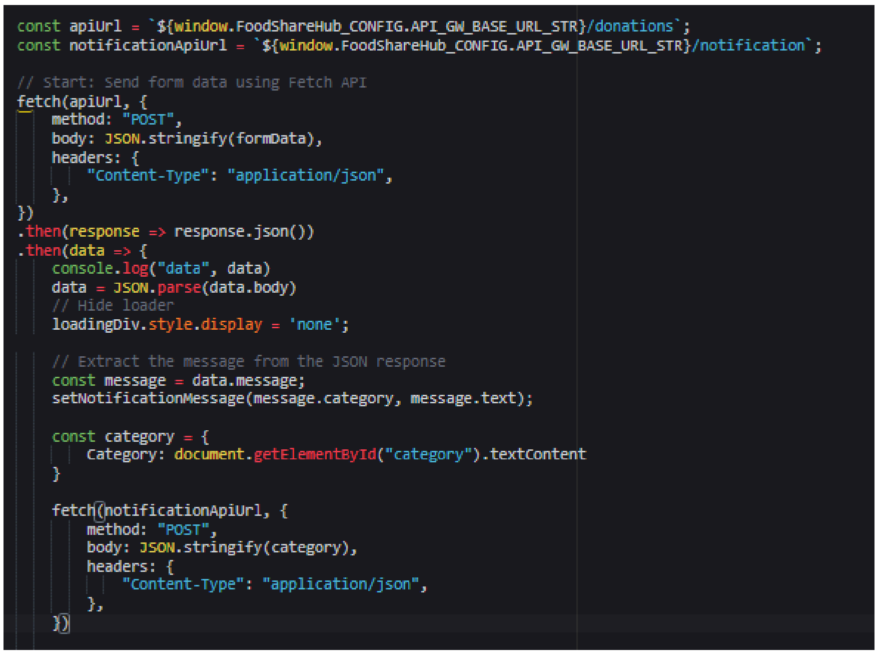
```
// Start: Retrieve form data
const formData = {
    Name: document.getElementById("name").value,
    Email: document.getElementById("email").value,
    Category: document.getElementById("category").value
};
// End: Retrieve form data

const loadingDiv = document.getElementById('loadingDiv');
loadingDiv.style.display = '';

const apiUrl = `${window.FoodShareHub_CONFIG.API_GW_BASE_URL_STR}/subscribe_notification`;
// Start: Send form data using Fetch API
fetch(apiUrl, {
    method: "POST",
    body: JSON.stringify(formData),
    headers: {
        "Content-Type": "application/json",
    },
})
.then(response => response.json())
.then(data => {
    console.log("data", data)
    data = JSON.parse(data.body)
    // Hide loader
    loadingDiv.style.display = 'none';

    // Extract the message from the JSON response
    const message = data.message;
    setNotificationMessage(message.category, message.text);

    // Redirect to the extracted URL
    window.location.href = "index.html";
})
```

5. Create the “send_notification_code.py” and “send_notification_wrapper.py” to create a lambda function that sends email notification.



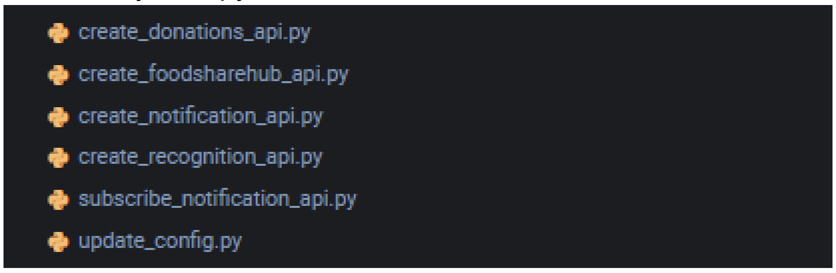
```
    print("EVENT", event)
    category = event['Category']
    with connection.cursor() as cursor:
        # Query users whose selected category matches the category value
        cursor.execute("SELECT Email FROM NotificationSubscribers WHERE Category = %s OR Category = 'All Categories'", (category,))
        matching_users = cursor.fetchall()

    # Initialize the SNS client
    sns_client = boto3.client('sns')

    for user in matching_users:
        user_email = user[0]
        print("user_email", user_email)
        message = "We've discovered a new item in your preferred category! Please explore it on Food Share Hub."

        # Publish the message to the specified SNS topic
        sns_client.publish(
            Message=message,
            Subject='New Item Found in Your Preferred Category: ' + category,
            TopicArn=sns_topic_arn,
        )
```

6. Edit the “addDonation.html” to trigger this lambda function after a new item is added to FoodShareHub



```
const apiUrl = `${window.FoodShareHub_CONFIG.API_GW_BASE_URL_STR}/donations`;
const notificationApiUrl = `${window.FoodShareHub_CONFIG.API_GW_BASE_URL_STR}/notification`;

// Start: Send form data using Fetch API
fetch(apiUrl, {
    method: "POST",
    body: JSON.stringify(formData),
    headers: {
        "Content-Type": "application/json",
    },
})
.then(response => response.json())
.then(data => {
    console.log("data", data)
    data = JSON.parse(data.body)
    // Hide loader
    loadingDiv.style.display = 'none';

    // Extract the message from the JSON response
    const message = data.message;
    setNotificationMessage(message.category, message.text);

    const category = {
        Category: document.getElementById("category").textContent
    }

    fetch(notificationApiUrl, {
        method: "POST",
        body: JSON.stringify(category),
        headers: {
            "Content-Type": "application/json",
        },
    })
```

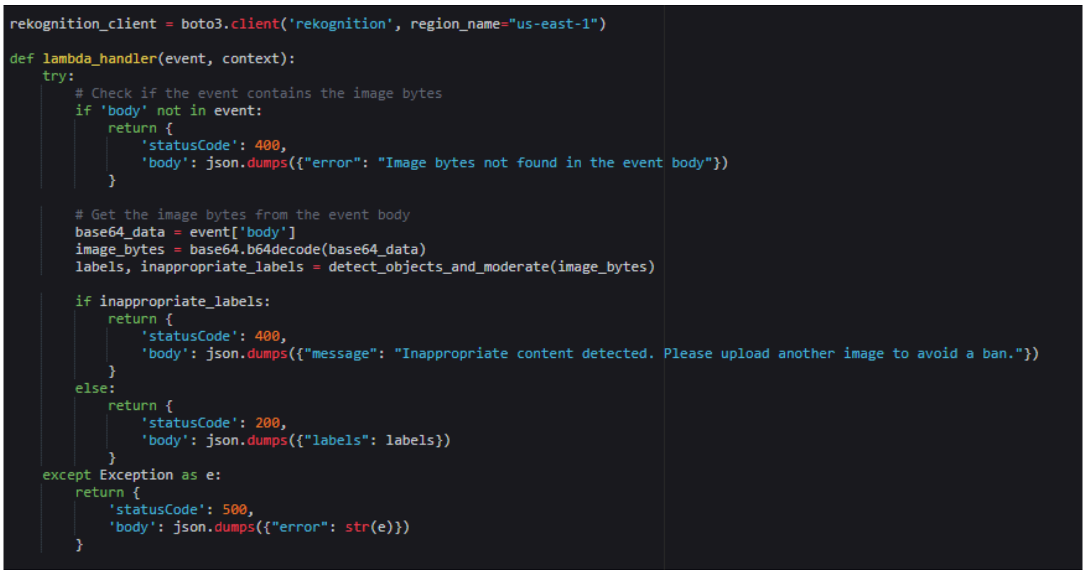
7. Additionally, use python file to create APIs



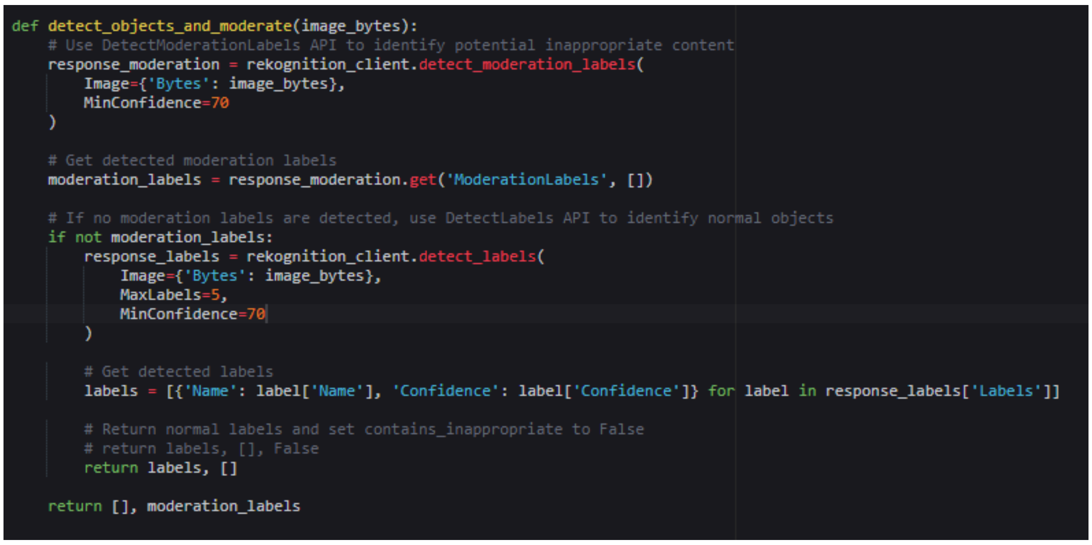
create_donations_api.py
create_foodsharehub_api.py
create_notification_api.py
create_recognition_api.py
subscribe_notification_api.py
update_config.py

Utilize Amazon Rekognition to analyze food images, automatically identify the item, and categorize it.

1. Create the Lambda Code: **detect_image_code.py** to detect image and return food item labels.

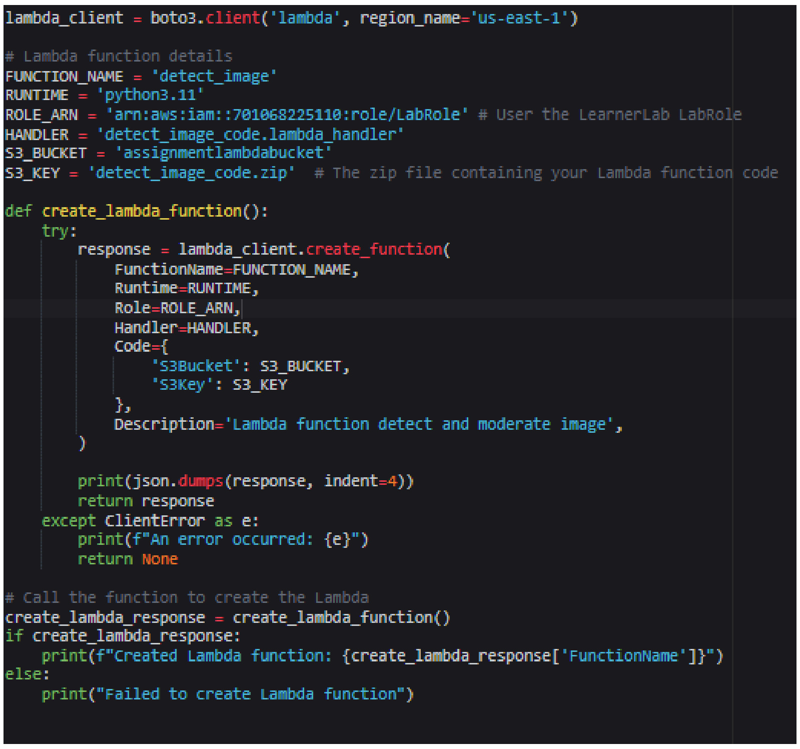


```python
rekognition_client = boto3.client('rekognition', region_name="us-east-1")

def lambda_handler(event, context):
    try:
        # Check if the event contains the image bytes
        if 'body' not in event:
            return {
                'statusCode': 400,
                'body': json.dumps({"error": "Image bytes not found in the event body"})
            }

        # Get the image bytes from the event body
        base64_data = event['body']
        image_bytes = base64.b64decode(base64_data)
        labels, inappropriate_labels = detect_objects_and_moderate(image_bytes)

        if inappropriate_labels:
            return {
                'statusCode': 400,
                'body': json.dumps({"message": "Inappropriate content detected. Please upload another image to avoid a ban."})
            }
        else:
            return {
                'statusCode': 200,
                'body': json.dumps({"labels": labels})
            }
    except Exception as e:
        return {
            'statusCode': 500,
            'body': json.dumps({"error": str(e)})
        }
```



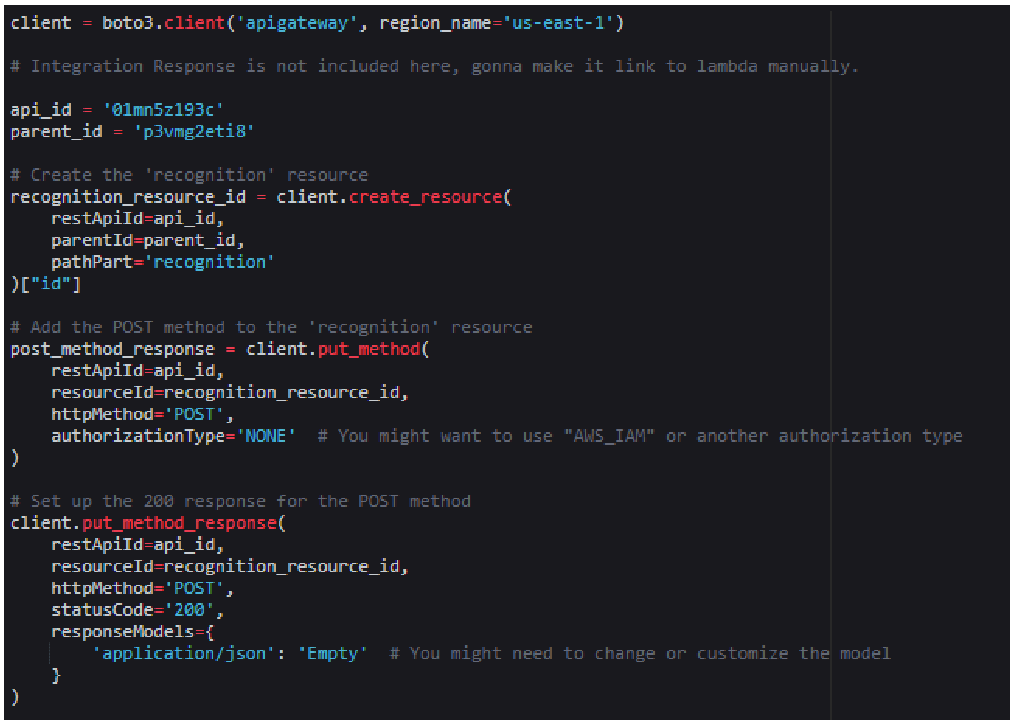
```python
def detect_objects_and_moderate(image_bytes):
    # Use DetectModerationLabels API to identify potential inappropriate content
    response_moderation = rekognition_client.detect_moderation_labels(
        Image={'Bytes': image_bytes},
        MinConfidence=70
    )

    # Get detected moderation labels
    moderation_labels = response_moderation.get('ModerationLabels', [])

    # If no moderation labels are detected, use DetectLabels API to identify normal objects
    if not moderation_labels:
        response_labels = rekognition_client.detect_labels(
            Image={'Bytes': image_bytes},
            MaxLabels=5,
            MinConfidence=70
        )

        # Get detected labels
        labels = [{'Name': label['Name'], 'Confidence': label['Confidence']} for label in response_labels['Labels']]

        # Return normal labels and set contains_inappropriate to False
        # return labels, [], False
        return labels, []

    return [], moderation_labels
```

2. Create Lambda Wrapper: **detect_image_wrapper.py** to create the lambda function.



```python
lambda_client = boto3.client('lambda', region_name='us-east-1')

# Lambda function details
FUNCTION_NAME = 'detect_image'
RUNTIME = 'python3.11'
ROLE_ARN = 'arn:aws:iam::701068225110:role/LabRole' # User the LearnerLab LabRole
HANDLER = 'detect_image_code.lambda_handler'
S3_BUCKET = 'assignmentlambdabucket'
S3_KEY = 'detect_image_code.zip'  # The zip file containing your Lambda function code

def create_lambda_function():
    try:
        response = lambda_client.create_function(
            FunctionName=FUNCTION_NAME,
            Runtime=RUNTIME,
            Role=ROLE_ARN,
            Handler=HANDLER,
            Code={
                'S3Bucket': S3_BUCKET,
                'S3Key': S3_KEY
            },
            Description='Lambda function detect and moderate image',
        )

        print(json.dumps(response, indent=4))
        return response
    except ClientError as e:
        print(f"An error occurred: {e}")
        return None

# Call the function to create the Lambda
create_lambda_response = create_lambda_function()
if create_lambda_response:
    print(f"Created Lambda function: {create_lambda_response['FunctionName']}")
else:
    print("Failed to create Lambda function")
```

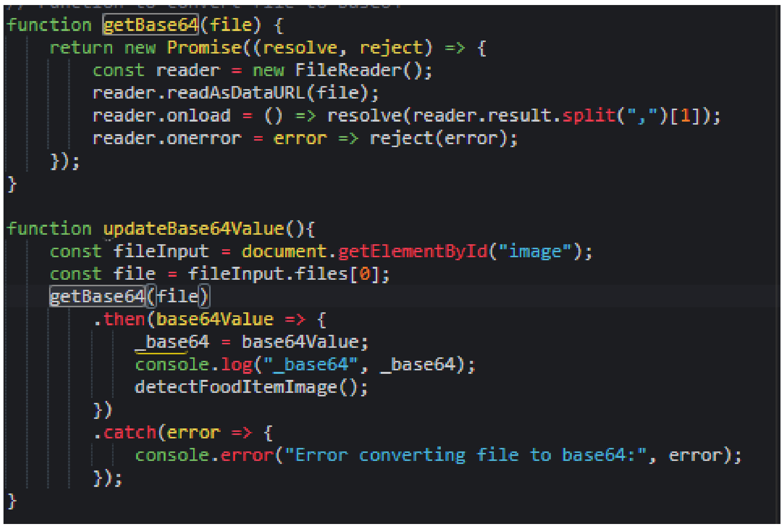
3. Create Recognition API: **create_recognition_api.py** and set up integration with lambda function “detect_image” manually through the amazon management console.



```python
client = boto3.client('apigateway', region_name='us-east-1')

# Integration Response is not included here, gonna make it link to lambda manually.

api_id = '01mn5z193c'
parent_id = 'p3vmg2eti8'

# Create the 'recognition' resource
recognition_resource_id = client.create_resource(
    restApiId=api_id,
    parentId=parent_id,
    pathPart='recognition'
)["id"]

# Add the POST method to the 'recognition' resource
post_method_response = client.put_method(
    restApiId=api_id,
    resourceId=recognition_resource_id,
    httpMethod='POST',
    authorizationType='NONE'  # You might want to use "AWS_IAM" or another authorization type
)

# Set up the 200 response for the POST method
client.put_method_response(
    restApiId=api_id,
    resourceId=recognition_resource_id,
    httpMethod='POST',
    statusCode='200',
    responseModels={
        'application/json': 'Empty'  # You might need to change or customize the model
    }
)
```

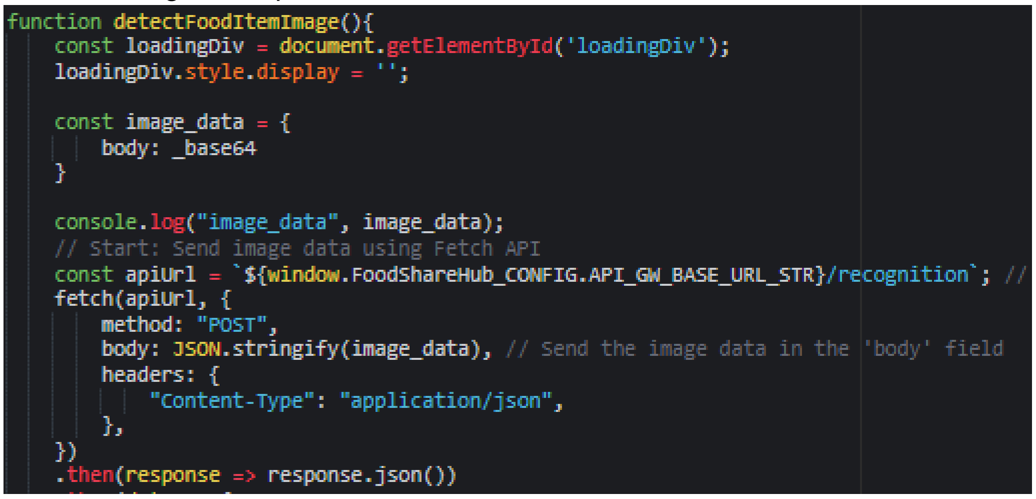
4. Modify the HTML Page: **addDonation.html**. Once an image is uploaded it will get the _base64 value first and then pass to the recognition API which calls the "detect_image" lambda function for image moderation and detection. After that based on the return, prompt error message or automatically filling up food item name and category.

   Get Base64 Value



```
function getBase64(file) {
    return new Promise((resolve, reject) => {
        const reader = new FileReader();
        reader.readAsDataURL(file);
        reader.onload = () => resolve(reader.result.split(",")[1]);
        reader.onerror = error => reject(error);
    });
}

function updateBase64Value(){
    const fileInput = document.getElementById("image");
    const file = fileInput.files[0];
    getBase64(file)
        .then(base64Value => {
            _base64 = base64Value;
            console.log("_base64", _base64);
            detectFoodItemImage();
        })
        .catch(error => {
            console.error("Error converting file to base64:", error);
        });
}
```

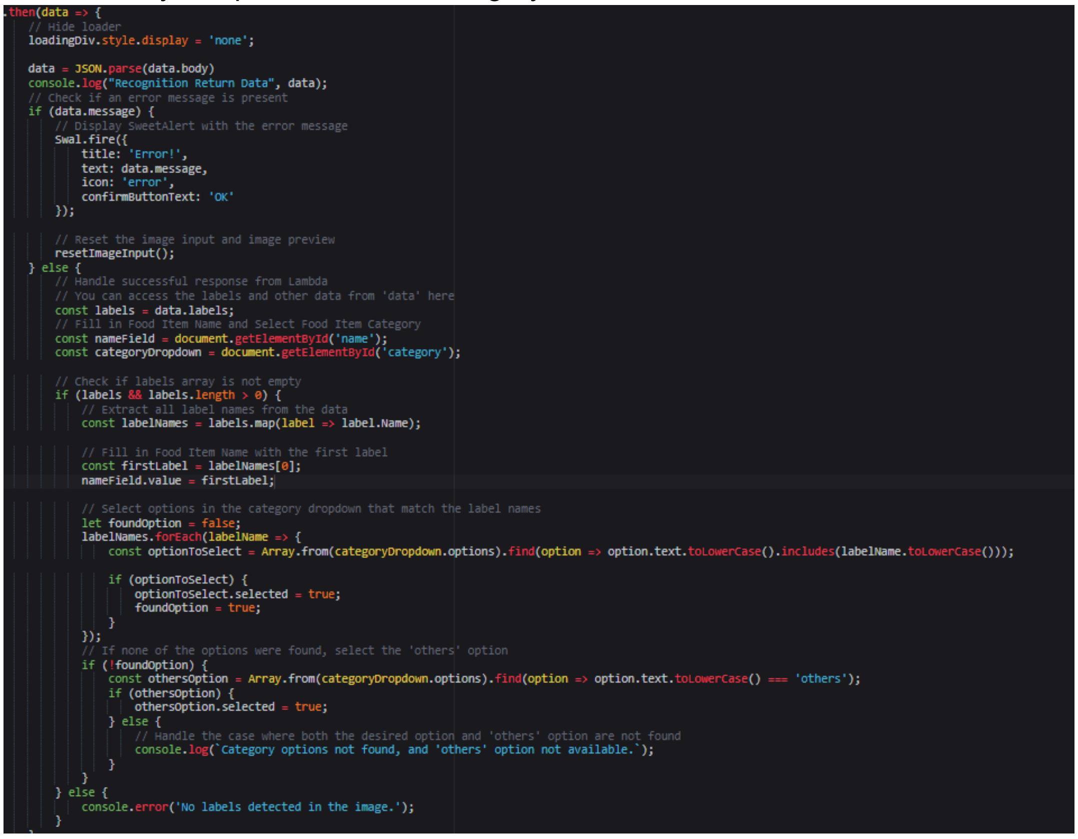
   Call the recognition api



```
function detectFoodItemImage(){
    const loadingDiv = document.getElementById('loadingDiv');
    loadingDiv.style.display = '';

    const image_data = {
        body: _base64
    }

    console.log("image_data", image_data);
    // Start: Send image data using Fetch API
    const apiUrl = `${window.FoodShareHub_CONFIG.API_GW_BASE_URL_STR}/recognition`; //
    fetch(apiUrl, {
        method: "POST",
        body: JSON.stringify(image_data), // Send the image data in the 'body' field
        headers: {
            "Content-Type": "application/json",
        },
    })
    .then(response => response.json())
```

Execute actions according to the responses, clear the image input field, and display an error message using SweetAlert if an inappropriate image is detected.

Automatically fill up ‘Name’ and “Category” Fields based on the return labels



```
.then(data => {
    // Hide loader
    loadingDiv.style.display = 'none';

    data = JSON.parse(data.body)
    console.log("Recognition Return Data", data);
    // Check if an error message is present
    if (data.message) {
        // Display SweetAlert with the error message
        Swal.fire({
            title: 'Error!',
            text: data.message,
            icon: 'error',
            confirmButtonText: 'OK'
        });

        // Reset the image input and image preview
        resetImageInput();
    } else {
        // Handle successful response from Lambda
        // You can access the labels and other data from 'data' here
        const labels = data.labels;
        // Fill in Food Item Name and Select Food Item Category
        const nameField = document.getElementById('name');
        const categoryDropdown = document.getElementById('category');

        // Check if labels array is not empty
        if (labels && labels.length > 0) {
            // Extract all label names from the data
            const labelNames = labels.map(label => label.Name);

            // Fill in Food Item Name with the first label
            const firstLabel = labelNames[0];
            nameField.value = firstLabel;

            // Select options in the category dropdown that match the label names
            let foundOption = false;
            labelNames.forEach(labelName => {
                const optionToSelect = Array.from(categoryDropdown.options).find(option => option.text.toLowerCase().includes(labelName.toLowerCase()));

                if (optionToSelect) {
                    optionToSelect.selected = true;
                    foundOption = true;
                }
            });
            // If none of the options were found, select the 'others' option
            if (!foundOption) {
                const othersOption = Array.from(categoryDropdown.options).find(option => option.text.toLowerCase() === 'others');
                if (othersOption) {
                    othersOption.selected = true;
                } else {
                    // Handle the case where both the desired option and 'others' option are not found
                    console.log(`Category options not found, and 'others' option not available.`);
                }
            }
        } else {
            console.error('No labels detected in the image.');
        }
```

Remarks:
You can test Amazon Rekognition for content moderation and image detection at: assignmentfoodsharehubbucket.s3.amazonaws.com/addDonation.html.

Part 4 (5%) – Advanced automation and features

The project was a great success. Other polytechnics heard about this project and have approached your team for assistance to implement a similar project.

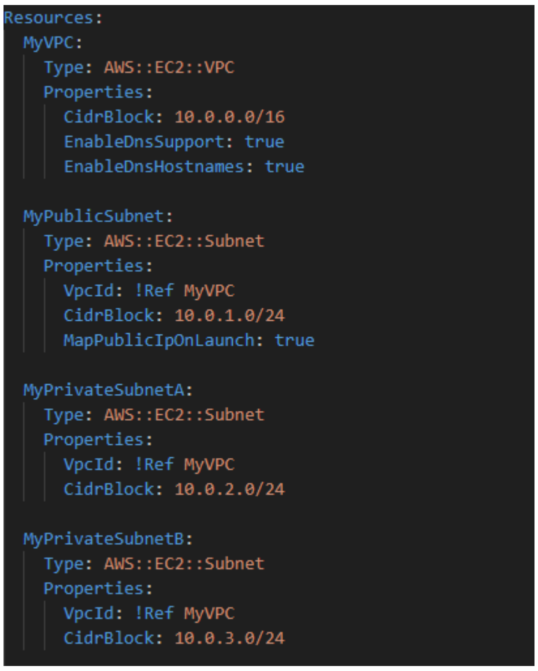
Design, document and implement your automation plan.

Your team had other plans and ideas for enhancement and improvement to the site, these enhancements should be included

(i) Document your design and steps with the necessary descriptions and screenshots for the automation process.

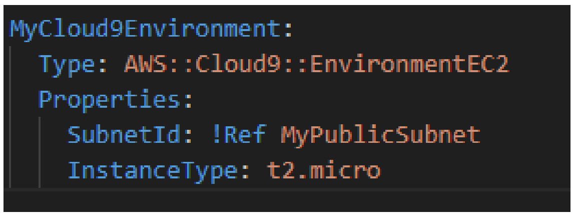
Serverless Application Model
Things to create by SAM

1. VPC



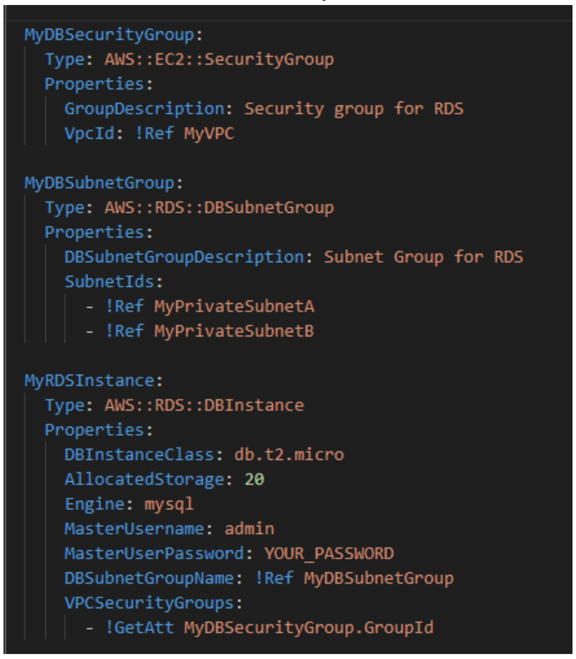
```
Resources:
  MyVPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
      EnableDnsSupport: true
      EnableDnsHostnames: true

  MyPublicSubnet:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId: !Ref MyVPC
      CidrBlock: 10.0.1.0/24
      MapPublicIpOnLaunch: true

  MyPrivateSubnetA:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId: !Ref MyVPC
      CidrBlock: 10.0.2.0/24

  MyPrivateSubnetB:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId: !Ref MyVPC
      CidrBlock: 10.0.3.0/24
```

2. Cloud9 Environment, Place in Public Subnet of the VPC



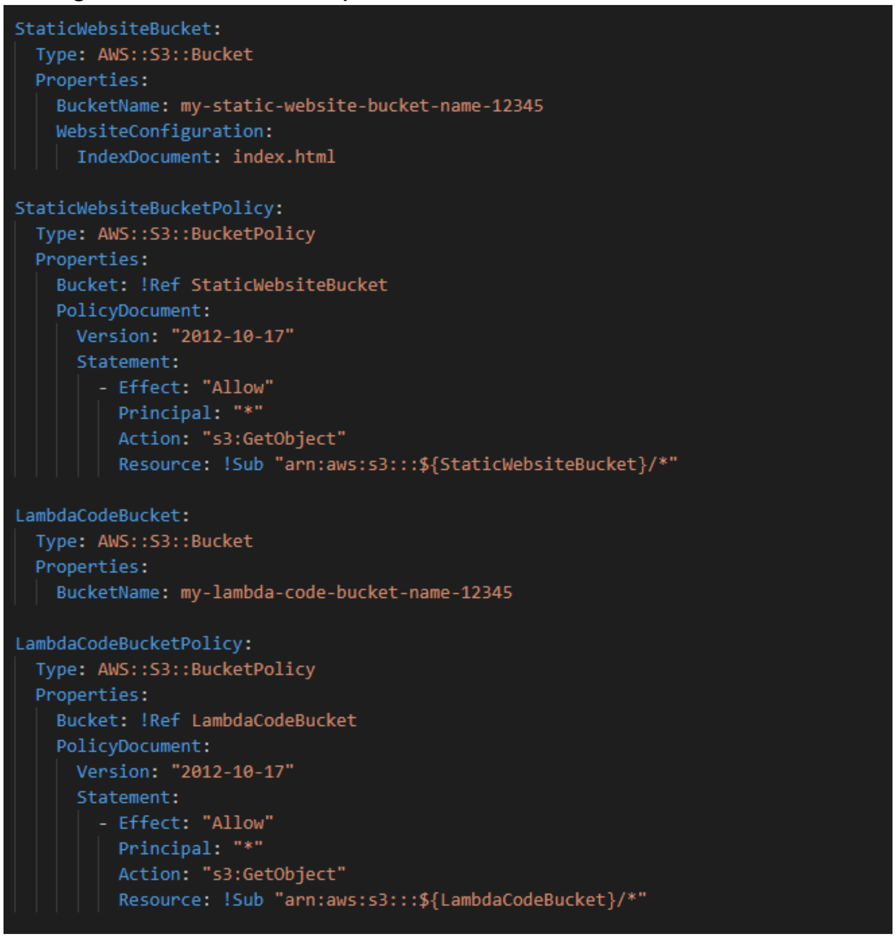
```
MyCloud9Environment:
  Type: AWS::Cloud9::EnvironmentEC2
  Properties:
    SubnetId: !Ref MyPublicSubnet
    InstanceType: t2.micro
```

3. Create SG, Subnet Group and Create RDS



```yaml
MyDBSecurityGroup:
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupDescription: Security group for RDS
    VpcId: !Ref MyVPC

MyDBSubnetGroup:
  Type: AWS::RDS::DBSubnetGroup
  Properties:
    DBSubnetGroupDescription: Subnet Group for RDS
    SubnetIds:
      - !Ref MyPrivateSubnetA
      - !Ref MyPrivateSubnetB

MyRDSInstance:
  Type: AWS::RDS::DBInstance
  Properties:
    DBInstanceClass: db.t2.micro
    AllocatedStorage: 20
    Engine: mysql
    MasterUsername: admin
    MasterUserPassword: YOUR_PASSWORD
    DBSubnetGroupName: !Ref MyDBSubnetGroup
    VPCSecurityGroups:
      - !GetAtt MyDBSecurityGroup.GroupId
```

4. Use one Amazon S3 bucket for hosting static web content, and a second S3 bucket for storing Lambda function zip files.

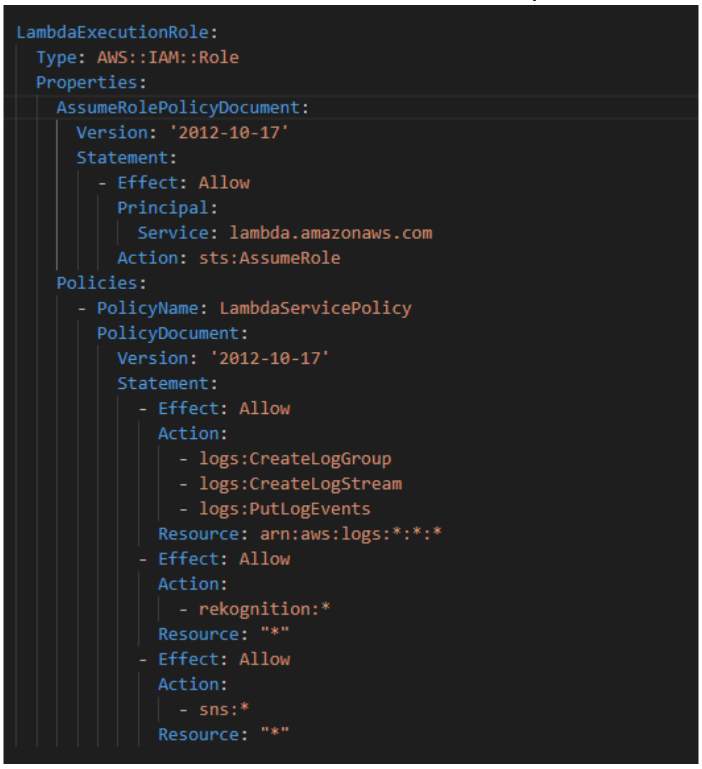


```yaml
StaticWebsiteBucket:
  Type: AWS::S3::Bucket
  Properties:
    BucketName: my-static-website-bucket-name-12345
    WebsiteConfiguration:
      IndexDocument: index.html

StaticWebsiteBucketPolicy:
  Type: AWS::S3::BucketPolicy
  Properties:
    Bucket: !Ref StaticWebsiteBucket
    PolicyDocument:
      Version: "2012-10-17"
      Statement:
        - Effect: "Allow"
          Principal: "*"
          Action: "s3:GetObject"
          Resource: !Sub "arn:aws:s3:::${StaticWebsiteBucket}/*"

LambdaCodeBucket:
  Type: AWS::S3::Bucket
  Properties:
    BucketName: my-lambda-code-bucket-name-12345

LambdaCodeBucketPolicy:
  Type: AWS::S3::BucketPolicy
  Properties:
    Bucket: !Ref LambdaCodeBucket
    PolicyDocument:
      Version: "2012-10-17"
      Statement:
        - Effect: "Allow"
          Principal: "*"
          Action: "s3:GetObject"
          Resource: !Sub "arn:aws:s3:::${LambdaCodeBucket}/*"
```

5. Create Lambda Functions

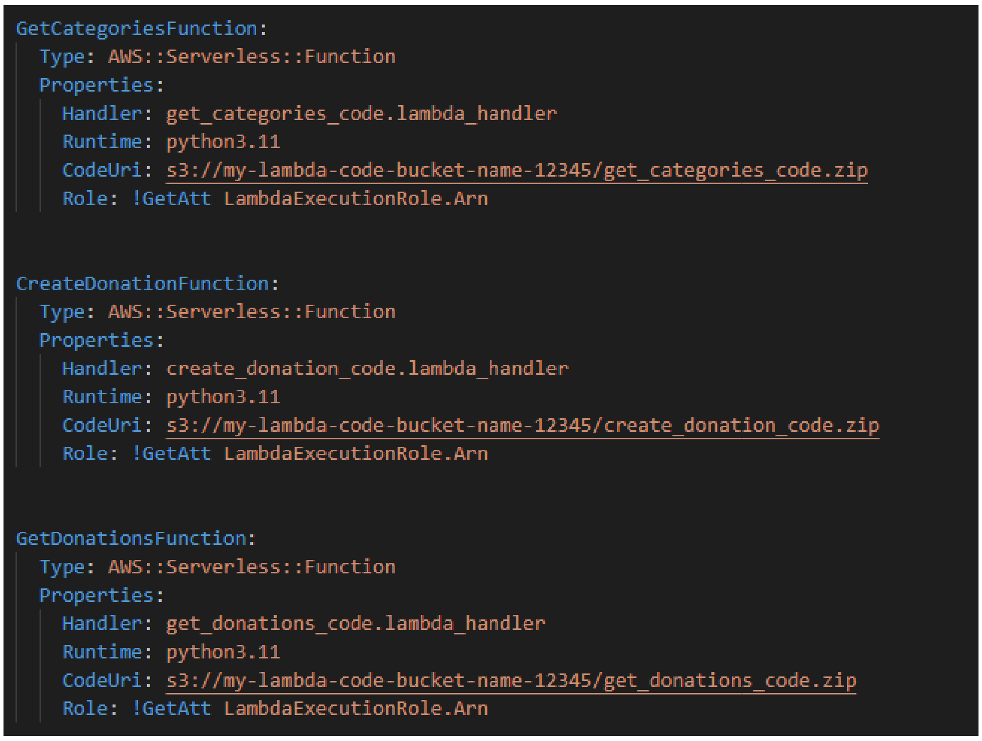
To ensure the Lambda function can access its code, the user must manually upload the code files to the LambdaCodeBucket.

LambdaExecutionRole for the lambda to perform action related to Recognition and SNS

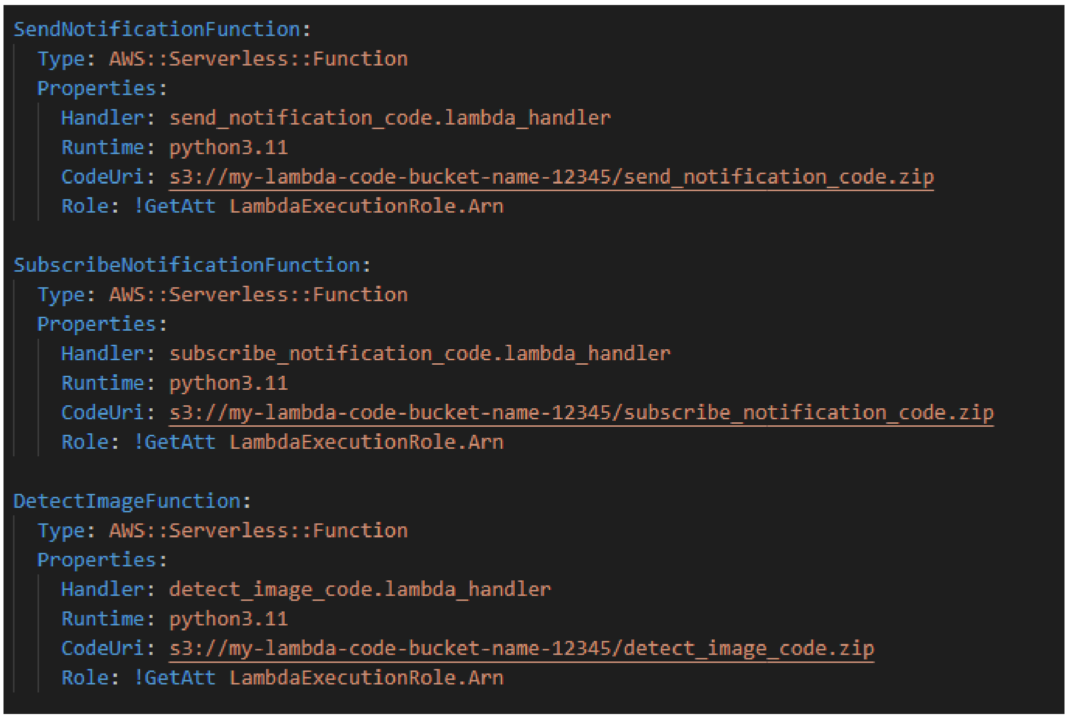


```yaml
LambdaExecutionRole:
  Type: AWS::IAM::Role
  Properties:
    AssumeRolePolicyDocument:
      Version: '2012-10-17'
      Statement:
        - Effect: Allow
          Principal:
            Service: lambda.amazonaws.com
          Action: sts:AssumeRole
    Policies:
      - PolicyName: LambdaServicePolicy
        PolicyDocument:
          Version: '2012-10-17'
          Statement:
            - Effect: Allow
              Action:
                - logs:CreateLogGroup
                - logs:CreateLogStream
                - logs:PutLogEvents
              Resource: arn:aws:logs:*:*:*
            - Effect: Allow
              Action:
                - rekognition:*
              Resource: "*"
            - Effect: Allow
              Action:
                - sns:*
              Resource: "*"
```

Functions:

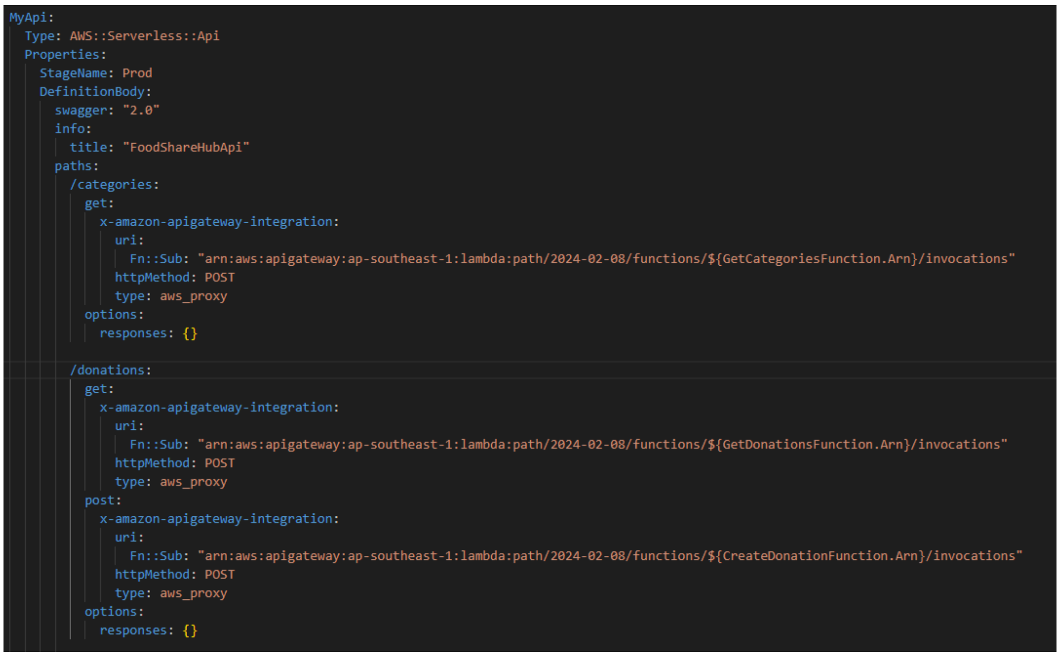


```yaml
GetCategoriesFunction:
  Type: AWS::Serverless::Function
  Properties:
    Handler: get_categories_code.lambda_handler
    Runtime: python3.11
    CodeUri: s3://my-lambda-code-bucket-name-12345/get_categories_code.zip
    Role: !GetAtt LambdaExecutionRole.Arn


CreateDonationFunction:
  Type: AWS::Serverless::Function
  Properties:
    Handler: create_donation_code.lambda_handler
    Runtime: python3.11
    CodeUri: s3://my-lambda-code-bucket-name-12345/create_donation_code.zip
    Role: !GetAtt LambdaExecutionRole.Arn


GetDonationsFunction:
  Type: AWS::Serverless::Function
  Properties:
    Handler: get_donations_code.lambda_handler
    Runtime: python3.11
    CodeUri: s3://my-lambda-code-bucket-name-12345/get_donations_code.zip
    Role: !GetAtt LambdaExecutionRole.Arn
```

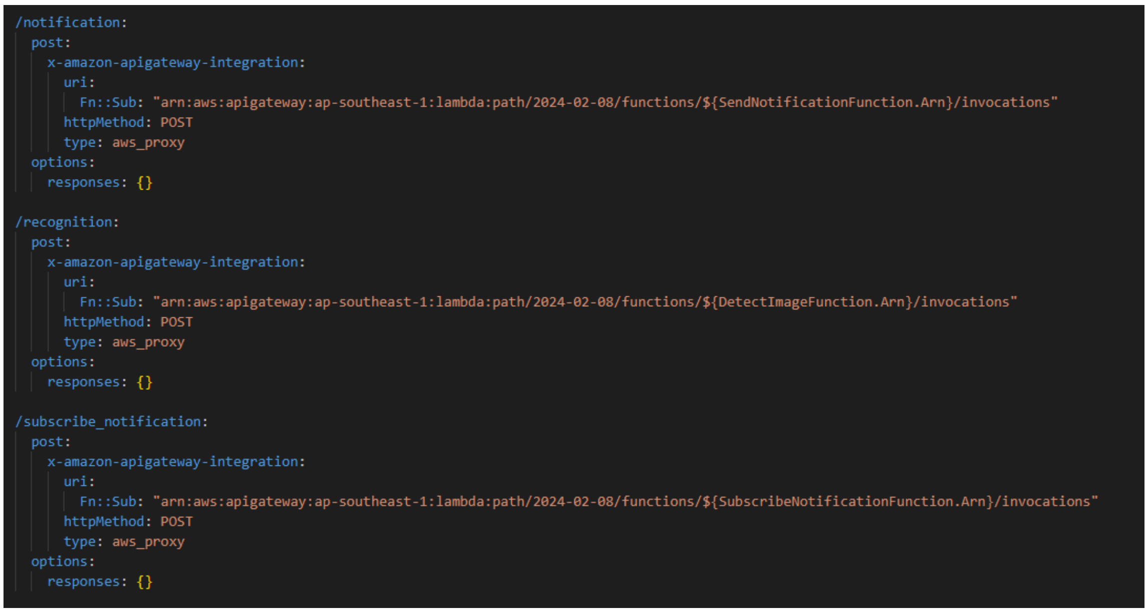


```yaml
SendNotificationFunction:
  Type: AWS::Serverless::Function
  Properties:
    Handler: send_notification_code.lambda_handler
    Runtime: python3.11
    CodeUri: s3://my-lambda-code-bucket-name-12345/send_notification_code.zip
    Role: !GetAtt LambdaExecutionRole.Arn

SubscribeNotificationFunction:
  Type: AWS::Serverless::Function
  Properties:
    Handler: subscribe_notification_code.lambda_handler
    Runtime: python3.11
    CodeUri: s3://my-lambda-code-bucket-name-12345/subscribe_notification_code.zip
    Role: !GetAtt LambdaExecutionRole.Arn

DetectImageFunction:
  Type: AWS::Serverless::Function
  Properties:
    Handler: detect_image_code.lambda_handler
    Runtime: python3.11
    CodeUri: s3://my-lambda-code-bucket-name-12345/detect_image_code.zip
    Role: !GetAtt LambdaExecutionRole.Arn
```

6. API Gateways and Integrate API with the lambda functions created



```yaml
MyApi:
  Type: AWS::Serverless::Api
  Properties:
    StageName: Prod
    DefinitionBody:
      swagger: "2.0"
      info:
        title: "FoodShareHubApi"
      paths:
        /categories:
          get:
            x-amazon-apigateway-integration:
              uri:
                Fn::Sub: "arn:aws:apigateway:ap-southeast-1:lambda:path/2024-02-08/functions/${GetCategoriesFunction.Arn}/invocations"
              httpMethod: POST
              type: aws_proxy
          options:
            responses: {}

        /donations:
          get:
            x-amazon-apigateway-integration:
              uri:
                Fn::Sub: "arn:aws:apigateway:ap-southeast-1:lambda:path/2024-02-08/functions/${GetDonationsFunction.Arn}/invocations"
              httpMethod: POST
              type: aws_proxy
          post:
            x-amazon-apigateway-integration:
              uri:
                Fn::Sub: "arn:aws:apigateway:ap-southeast-1:lambda:path/2024-02-08/functions/${CreateDonationFunction.Arn}/invocations"
              httpMethod: POST
              type: aws_proxy
          options:
            responses: {}
```



```yaml
/notification:
  post:
    x-amazon-apigateway-integration:
      uri:
        Fn::Sub: "arn:aws:apigateway:ap-southeast-1:lambda:path/2024-02-08/functions/${SendNotificationFunction.Arn}/invocations"
      httpMethod: POST
      type: aws_proxy
  options:
    responses: {}

/recognition:
  post:
    x-amazon-apigateway-integration:
      uri:
        Fn::Sub: "arn:aws:apigateway:ap-southeast-1:lambda:path/2024-02-08/functions/${DetectImageFunction.Arn}/invocations"
      httpMethod: POST
      type: aws_proxy
  options:
    responses: {}

/subscribe_notification:
  post:
    x-amazon-apigateway-integration:
      uri:
        Fn::Sub: "arn:aws:apigateway:ap-southeast-1:lambda:path/2024-02-08/functions/${SubscribeNotificationFunction.Arn}/invocations"
      httpMethod: POST
      type: aws_proxy
  options:
    responses: {}
```

(j) Write a brief explanation on the future enhancement plan of your “FoodShareHub” project.

1. **Login System**: Create a login system to increase security and allow staff to manage food item listings more effectively. This enhancement aims to improve the platform's security and administrative functionality, ensuring a more controlled and secure environment for both the staff managing the platform and the users benefiting from it.
2. **Advanced Search and Filtering**: Enhance the search functionality with more advanced filters, such as location-based filtering, item category. This will help users find what they need more efficiently.
3. **Feedback and Rating System**: Introduce a feedback and rating system for transactions to ensure the quality of food items shared and foster trust among users. Users could rate their experience and the condition of the food items, with the option to leave comments.

Appendix – Rubrics

CATEGORY	A	B	C	D
Implementation (45%)	Completed all systems with persistence storage integration and all managed cloud service that is aligned to the problem statement. The system is user friendly and caters to all groups of users fully.	Partial completion of system with persistence storage integration and some managed cloud service that is aligned to the problem statement. The system has some user-friendly features implemented.	Incomplete system with persistence storage integration and minimum or no managed cloud service that is aligned to the problem statement. The system has basic user navigation components.	Incomplete system without persistence storage integration and no managed cloud service
Cloud Native Development (20%)	Demonstrate effective use of at least 2 managed cloud services. All the managed cloud services are fully integrated	Demonstrate effective use of 1 managed cloud service. The clouod services used are partially integrated	Able to identify and use basic cloud service suchs as compute, storage or database to implement the solution. Some form of integration has been used.	Unable to use any cloud service to implement the system
Infrastructure as Code (15%)	Demonstrate effective use of infrastructure as code to create the cloud infrastructure automatically	Demonstrate partial use of infrastructure as code to create the cloud infrastruction partially	Able to create cloud infrastructure using the management console	Unable to completely create the cloud infrastructure
Explanation (20%)	Confident with good illustration and explanation to explain the work well. Logical and feasible proposals	Somewhat confident with decent illustration and explanation to somewhat explain the work done	Lack of confident with some illustration and explanation, but unable to explain the work well	No confidence, and little illustration and explanation, unable to describe the work done

<~The END~>