

HomeMade Pickles & Snacks: Taste The Best

Project Description:

HomeMade Pickles & Snacks Taste the Best is a cloud-based culinary platform revolutionizing access to authentic, handcrafted pickles and snacks. Addressing the growing demand for preservative-free, traditional recipes, this initiative combines artisanal craftsmanship with cutting-edge technology to deliver farm-fresh flavors directly to consumers. Built on Flask for backend efficiency and hosted on AWS EC2 for scalable performance, the platform offers seamless browsing, ordering, and subscription management. DynamoDB ensures real-time inventory tracking and personalized user experiences, while fostering sustainability through partnerships with local farmers and eco-friendly packaging. From tangy regional pickles to wholesome snacks, every product celebrates heritage recipes, nutritional integrity, and convenience—proving that tradition and innovation can coexist deliciously. "Preserving Traditions, One Jar at a Time."

Scenario 1: Scalable Order Management for High Demand

A cloud-based system ensures seamless order processing during peak user activity. For instance, during a promotional event, hundreds of users simultaneously access the platform to place orders. The backend efficiently processes requests, updates inventory in real-time, and manages user sessions. The cloud infrastructure handles traffic spikes without performance degradation, ensuring smooth transactions and minimizing wait times.

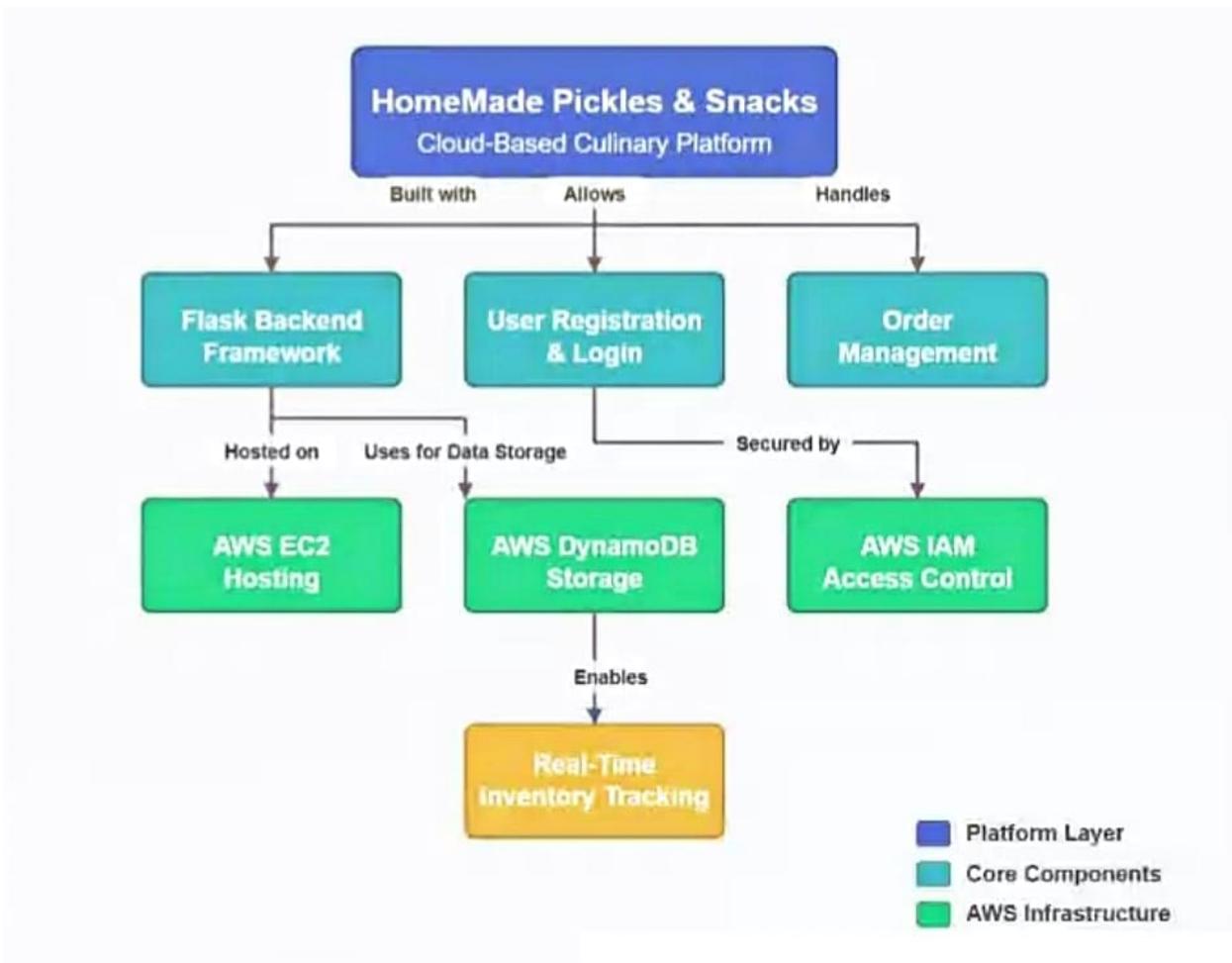
Scenario 2: Real-Time Inventory Tracking and Updates

When a customer places an order for a product, the system instantly updates stock levels and records transaction details. For example, a user purchases an item, triggering automatic inventory deduction and order confirmation. Staff members receive updated dashboards to monitor stock availability and fulfillment progress, ensuring timely restocking and minimizing overselling risks.

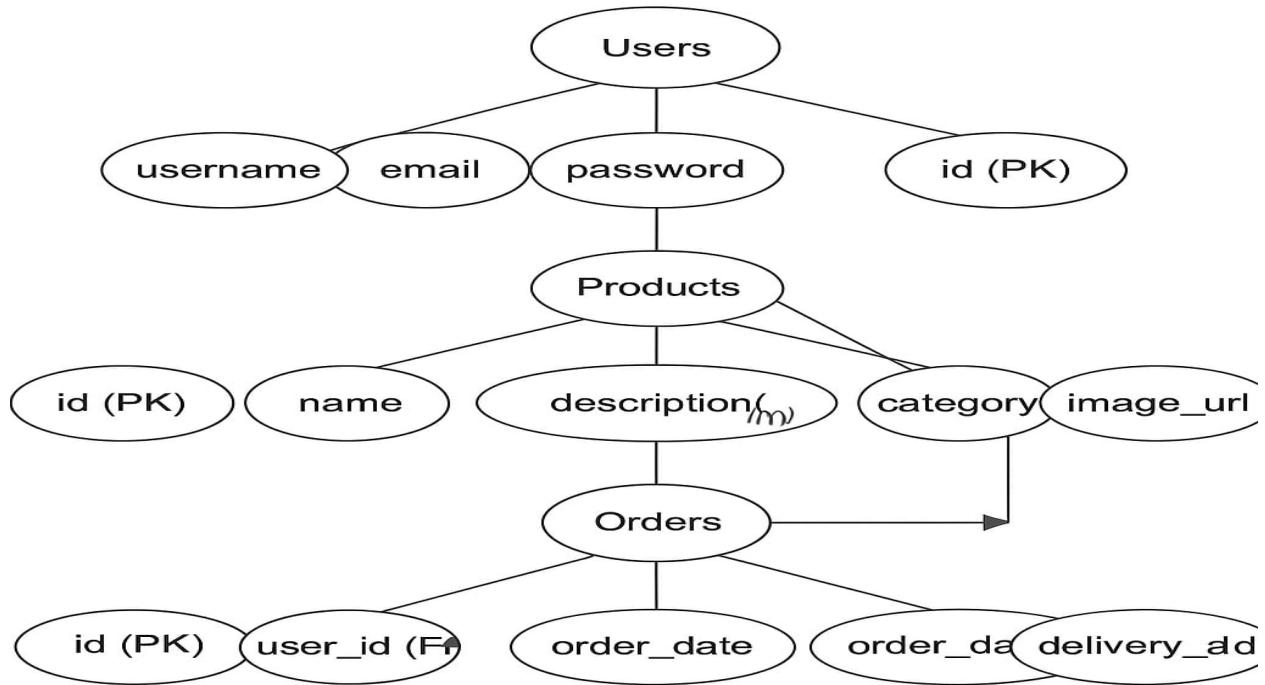
Scenario 3: Personalized User Experience and Recommendations:

The platform leverages user behavior data to enhance engagement. A returning customer, for instance, views tailored recommendations based on past purchases and browsing history. The system dynamically adjusts suggestions in real-time, while maintaining fast response rates even during high traffic, creating a frictionless and intuitive shopping experience.

AWS ARCHITECTURE :



Entity Relationship(ER) Diagram :



Pre-requisites :

1. AWS Account Setup: [AWS Account Setup](#)
2. Understanding IAM: [IAM Overview](#)
3. Amazon EC2 Basics: [EC2 Tutorial](#)
4. DynamoDB Basics: [DynamoDB Introduction](#)
5. Git Version Control: [Git Documentation](#)

Project WorkFlow:

1. AWS Account Setup and Login

Activity 1.1: Set up an AWS account if not already done.

Activity 1.2: Log in to the AWS Management Console

2. DynamoDB DatabaseCreation and Setup

Activity 2.1: Create a DynamoDB Table.

Activity 2.2: Configure Attributes for User Data and Book Requests.

3.SNS Notification Setup

Activity 3.1: Create SNS topics for book request notifications.

Activity 3.2: Subscribe users and library staff to SNS email notifications.

4.Backend Development and Application Setup

Activity 4.1:Develop the Backend Using Flask.

Activity 4.2: Integrate AWS Services Using boto3.

5.IAM Role Setup

Activity 5.1: Create IAM Role

Activity 5.2: Attach Policies

6.EC2 Instance Setup

Activity 6.1: Launch an EC2 instance to host the Flask application.

Activity 6.2: Configure security groups for HTTP, and SSH access.

7.Deployment on EC2

Activity 7.1:Upload Flask Files

Activity 7.2: Run the Flask App

8.Testing and Deployment

Activity 8.1: Conduct functional testing to verify user registration, login, Pickles and snacks orders, and notifications.

Milestone 1: AWS Account Setup and Login

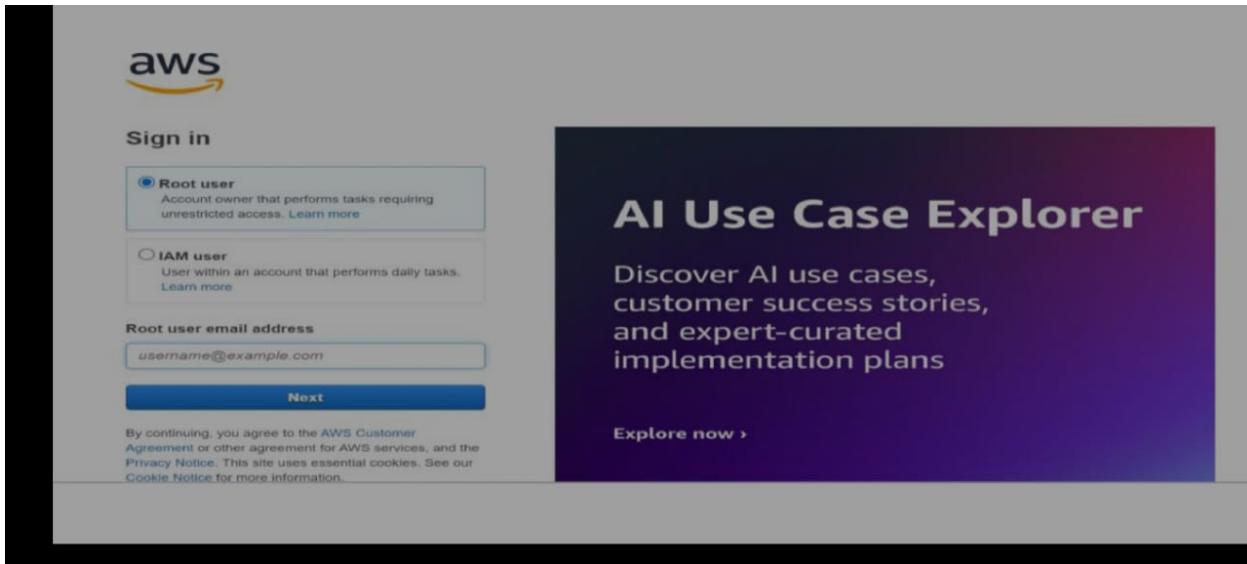
Activity 1.1: Set up an AWS account if not already done.

Sign up for an AWS account and configure billing settings



Activity 1.2: Log in to the AWS Management Console

After setting up your account, log in to the AWS Management Console.



Milestone 2: DynamoDB Database Creation and Setup

L Activity 2.1: Navigate to the DynamoDB

- In the AWS Console, navigate to DynamoDB and click on create tables.

A screenshot of the AWS EC2 Instances page. On the left, the navigation menu is open, showing 'EC2' selected under 'Services'. The main content area displays a table with one row of data. The table columns include 'Last updated', 'Connect', 'Instance state', 'Actions', 'Launch instances', 'Alarm status', 'Availability Zone', 'Public IPv4 DNS', and 'Public IPv4 ...'. The single instance listed has a Public IPv4 DNS of 'ec2-54-90-214-21.com...' and a Public IPv4 IP of '54.90.214.21'. Other details shown include 'View alarms +', 'us-east-1c', and 'Auto Scaling Group name'. The status bar at the bottom indicates the instance was last updated 7 minutes ago.

DynamoDB

Dashboard

Tables

Explore items

PartiQL editor

Backups

Exports to S3

Imports from S3

Integrations

Reserved capacity

Settings

DAX

Clusters

Subnet groups

Parameter groups

Events

Dashboard

Favorite tables

No favorite tables

To get started, click the star icon on the tables page or table details page to favorite a table.

Alarms (0)

No custom alarms

DAX clusters (0)

Create resources

Create an Amazon DynamoDB table for fast and predictable database performance at any scale. Learn more

Create table

Create DAX cluster

Amazon DynamoDB Accelerator (DAX) is a fully-managed, highly-available, in-memory caching service for DynamoDB. Learn more

What's new

DynamoDB

Tables

Explore items

PartiQL editor

Backups

Exports to S3

Imports from S3

Integrations

Reserved capacity

Settings

Tables (0)

Name

Status

Partition key

Sort key

Indexes

Replication Regions

Deletion protection

Favorite

Read capacity mode

Write capacity mode

Total size

Table d

Any tag key

Any tag value

Actions

Delete

Create table

You have no tables in this account in this AWS Region.

Create table

Activity 2.2: Create a DynamoDB table for storing registration details and pickles and snacks request.

Create Users table with partition key "Email" with type String and click on create tables.

The EcomOrders table was created successfully.

Name	Status	Partition key	Sort key	Indexes	Replication Regions	Deletion protection	Favorite	Read capacity mode	Write capacity mode
EcomOrders	Active	order_id (\$)	-	0 0	Off	☆	On-demand	On-demand	
EcomProducts	Active	id (\$)	-	0 0	Off	☆	On-demand	On-demand	
EcomUsers	Active	email (\$)	-	0 0	Off	☆	On-demand	On-demand	

- Follow the same steps to create a orders table with Email as the primary key for pickles order data.

Milestone 3: SNS Notification Setup

- Activity 3.1: Create SNS topics for sending email notifications to users and Customers .**
- In the AWS Console, search for SNS and navigate to the SNS Dashboard.

Search results for 'sns'

Services

Simple Notification Service ☆
SNS managed message topics for Pub/Sub

Route 53 Resolver
Resolve DNS queries in your Amazon VPC and on-premises network.

Route 53 ☆
Scalable DNS and Domain Name Registration

AWS End User Messaging ☆
Engage your customers across multiple communication channels

Show more ▶

Features

Events

ElastiCache feature

SMS

AWS End User Messaging feature

Hosted zones

Amazon SNS

New Feature

Amazon SNS now supports in-place message archiving and replay for FIFO topics. [Learn more](#)

Dashboard

Topics

Subscriptions

Mobile

Push notifications

Text messaging (SMS)

Application Integration

Amazon Simple Notification Service

Pub/sub messaging for microservices and serverless applications.

Amazon SNS is a highly available, durable, secure, fully managed pub/sub messaging service that enables you to decouple microservices, distributed systems, and event-driven serverless applications. Amazon SNS provides topics for high-throughput, push-based, many-to-many messaging.

Create topic

Topic name

A topic is a message channel. When you publish a message to a topic, it fans out the message to all subscribed endpoints.

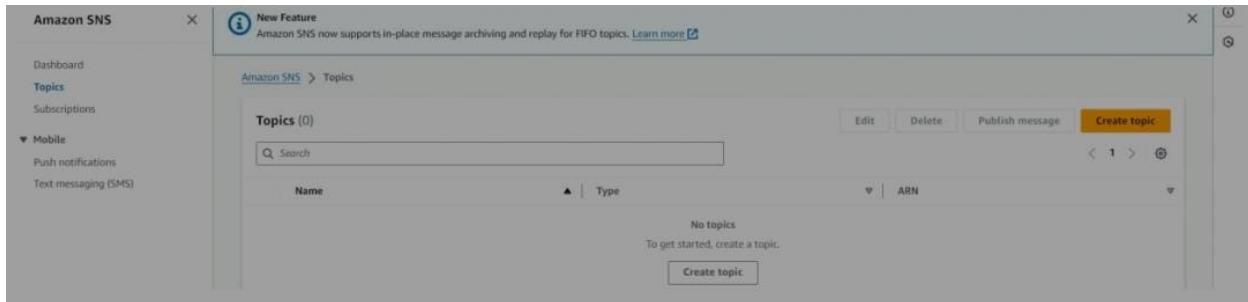
MyTopic

Next step

Start with an overview

Pricing

Click on Create Topic and choose a name for the topic.



Choose Standard type for general notification use cases and Click on Create Topic.

The screenshot shows the 'Create topic' wizard in the 'Details' step. It has two tabs: 'Type' (selected) and 'Info'. Under 'Type', there are two options: 'FIFO (first-in, first-out)' and 'Standard'. 'Standard' is selected and highlighted with a green border. Both options have associated bullet points. Below the type selection is a 'Name' field containing 'BookRequestNotifications' with a note about character limits. At the bottom is a 'Display name - optional' field with a note about SMS display names and a character limit of 100.

Type Topic type cannot be modified after topic is created

FIFO (first-in, first-out)

- Strictly-preserved message ordering
- Exactly-once message delivery
- High throughput, up to 300 publishes/second
- Subscription protocols: SQS

Standard

- Best-effort message ordering
- At-least once message delivery
- Highest throughput in publishes/second
- Subscription protocols: SQS, Lambda, HTTP, SMS, email, mobile application endpoints

Name

BookRequestNotifications

Maximum 256 characters. Can include alphanumeric characters, hyphens (-) and underscores (_).

Display name - optional To use this topic with SMS subscriptions, enter a display name. Only the first 10 characters are displayed in an SMS message.

My Topic

Maximum 100 characters.

▶ **Access policy - optional** [Info](#)
This policy defines who can access your topic. By default, only the topic owner can publish or subscribe to the topic.

▶ **Data protection policy - optional** [Info](#)
This policy defines which sensitive data to monitor and to prevent from being exchanged via your topic.

▶ **Delivery policy (HTTP/S) - optional** [Info](#)
The policy defines how Amazon SNS retries failed deliveries to HTTP/S endpoints. To modify the default settings, expand this section.

▶ **Delivery status logging - optional** [Info](#)
These settings configure the logging of message delivery status to CloudWatch Logs.

▶ **Tags - optional**
A tag is a metadata label that you can assign to an Amazon SNS topic. Each tag consists of a key and an optional value. You can use tags to search and filter your topics and track your costs. [Learn more](#)

▶ **Active tracing - optional** [Info](#)
Use AWS X-Ray active tracing for this topic to view its traces and service map in Amazon CloudWatch. Additional costs apply.

Cancel

Create topic

Configure the SNS topic and note down the Topic ARN.

Activity 3.2: Subscribe users and customers to relevant SNS topics to receive real-time notifications when a snacks order is made.

- Subscribe users (or admin) to this topic via Email. When a snacks reque is made, notifications will be sent to the subscribed emails.

Amazon SNS > Subscriptions > Create subscription

Create subscription

Details

Topic ARN
 X

Protocol
 The type of endpoint to subscribe
 ▼

Endpoint
 An email address that can receive notifications from Amazon SNS.

i After your subscription is created, you must confirm it. [Info](#)

▶ **Subscription filter policy - optional** [Info](#)
 This policy filters the messages that a subscriber receives.

▶ **Redrive policy (dead-letter queue) - optional** [Info](#)
 Send undeliverable messages to a dead-letter queue.

Cancel Create subscription

Amazon SNS X

i **New Feature:** Amazon SNS now supports in-place message archiving and replay for FIFO topics. [Learn more](#) X

i Subscription to BookRequestNotifications created successfully.
 The ARN of the subscription is arn:aws:sns:ap-south-1:557690616836:BookRequestNotifications:d78e0371-9235-404d-952c-85c2743607c4.

Amazon SNS > Topics > BookRequestNotifications > Subscription: d78e0371-9235-404d-952c-85c2743607c4

Details	
ARN	Status
arn:aws:sns:ap-south-1:557690616836:BookRequestNotifications:d78e0371-9235-404d-952c-85c2743607c4	i Pending confirmation
Endpoint	Protocol
instantlibrary2@gmail.com	Email
Topic	
BookRequestNotifications	
Subscription Principal	
arn:aws:sns:ap-south-1:557690616836:BookRequestNotifications:d78e0371-9235-404d-952c-85c2743607c4	

▶ **Subscription filter policy** [Info](#)
 This policy filters the messages that a subscriber receives.

No filter policy configured for this subscription.
 To apply a filter policy, edit this subscription.

Edit

□ After subscription request for the mail confirmation

New Feature
Amazon SNS now supports in-place message archiving and replay for FIFO topics. [Learn more](#)

Confirmation request was sent successfully.
The ARN of the subscription is arn:aws:sns:ap-south-1:557690616836:BookRequestNotifications:7ae5d16-12ad-4731-97f0-c543c2213ad5.

Amazon SNS > Topics > BookRequestNotifications

BookRequestNotifications

Details

Name: BookRequestNotifications Display name:

ARN: arn:aws:sns:ap-south-1:557690616836:BookRequestNotifications

Type: Standard

Subscriptions | Access policy | Data protection policy | Delivery policy (HTTP/HTTPS) | Delivery status logging | Encryption | Tags | Integrations

Subscriptions (1)

ID: Pending confirmation Endpoint: imstundlibrary2@gmail.com Status: Pending confirmation Protocol: EMAIL

Edit | Delete | Request confirmation | Confirm subscription | Create subscription

- Navigate to the subscribed Email account and Click on the confirm subscription in the AWS Notification- Subscription Confirmation mail.

AWS Notification - Subscription Confirmation Inbox x

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

You have chosen to subscribe to the topic:
arn:aws:sns:ap-south-1:557690616836:BookRequestNotifications

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

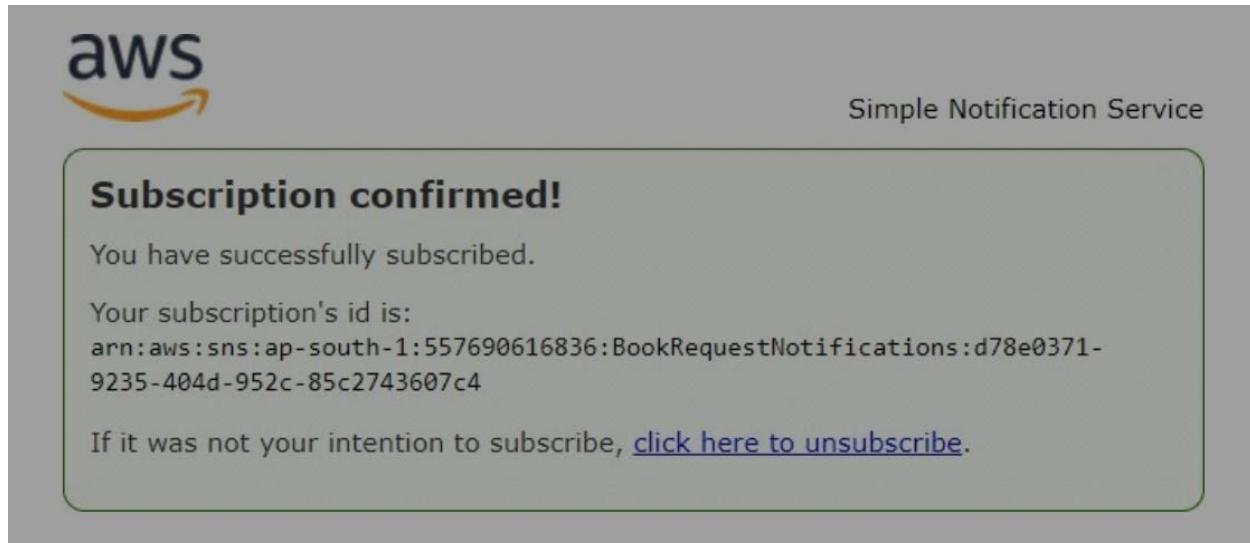
Please do not reply directly to this email. If you wish to remove yourself from receiving all future SNS subscription confirmation requests please send an email to [sns-opt-out](#)

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

You have chosen to subscribe to the topic:
arn:aws:sns:ap-south-1:557690616836:BookRequestNotifications

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](#)



- ❑ Successfully done with the SNS mail subscription and setup, now store the ARN link.

This screenshot shows the AWS SNS "BookRequestNotifications" topic details page. The left sidebar has navigation links for Dashboard, Topics, Subscriptions, and Mobile (Push notifications, Text messaging (SMS)). The main content area shows the topic name "BookRequestNotifications" and its ARN "arn:aws:sns:ap-south-1:557690616836:BookRequestNotifications". The "Type" is listed as "Standard". On the right, there are buttons for Edit, Delete, and Publish message. Below the topic details, a "Subscriptions" section lists one subscription: "d78e0371-9235-404d-952c-85c2743607c4" with the endpoint "mailto:library2@gmail.com" and status "Confirmed". There are also buttons for Edit, Delete, Request confirmation, Confirm subscription, and Create subscription.

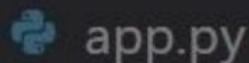
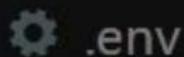
Milestone 4:Backend Development and Application Setup

- ❑ **Activity 4.1:** Develop the backend using Flask
- ❑ **File Explorer Structure**

✓ HOMEMADE PICKLES & SNACKS TASTE THE BEST

✓ templates

- <> base.html
- <> category.html
- <> index.html
- <> login.html
- <> products.html
- <> signup.html



Description of the code :

❑ Flask App Initialization

```
app.py > ...  
from flask import Flask, request, session, redirect, url_for, render_template, flash, jsonify  
import boto3  
from werkzeug.security import generate_password_hash, check_password_hash  
from datetime import datetime  
import uuid  
import os  
import logging  
import smtplib  
from email.mime.text import MIMEText  
from email.mime.multipart import MIMEMultipart  
from dotenv import load_dotenv
```

```
app = Flask(__name__)
```

❑ Dynamodb Setup:

```

# AWS Resources
#
# -----
dynamodb = boto3.resource('dynamodb', region_name=AWS_REGION)
sns = boto3.client('sns', region_name=AWS_REGION)

products_table = dynamodb.Table(PRODUCTS_TABLE)
users_table = dynamodb.Table(USER_TABLE)
orders_table = dynamodb.Table(ORDERS_TABLE)

# -----

```

SNS Connection

```

# SNS Topic ARN (create the SNS topic in AWS and provide the ARN here)
sns = boto3.client('sns', region_name='ap-south-1')
sns_topic_arn = 'arn:aws:sns:ap-south-1:557690616836:BookRequestNotifications'

# Email settings (for sending emails)
SMTP_SERVER = "smtp.gmail.com"
SMTP_PORT = 587
SENDER_EMAIL = "instantlibrary2@gmail.com"
SENDER_PASSWORD = "luut dsih nyvq dgzv" # Your app password

# Function to send email
def send_email(to_email, subject, body):
    msg = MIMEText(body)
    msg['From'] = SENDER_EMAIL
    msg['To'] = to_email
    msg['Subject'] = subject
    msg.attach(MIMEText(body, 'plain'))

    try:
        server = smtplib.SMTP(SMTP_SERVER, SMTP_PORT)
        server.starttls()
        server.login(SENDER_EMAIL, SENDER_PASSWORD)
        text = msg.as_string()
        server.sendmail(SENDER_EMAIL, to_email, text)
        server.quit()
        print("Email sent successfully")
    except Exception as e:
        print(f"Failed to send email: {e}")

```

Routes for Web Pages

Home Route:

```

"""
# Routes
# -----
@app.route('/')
def home():
    try:
        response = products_table.scan(Limit=3)
        products = response.get('Items', [])
    except Exception as e:
        logger.error(f"Failed to load products: {e}")
        products = []
    return render_template('index.html', featured_products=products)

```



Signup Route:

```

1 @app.route('/signup', methods=['GET', 'POST'])
2 def signup():
3     if request.method == 'POST':
4         email = request.form['email']
5         name = request.form['name']
6         password = generate_password_hash(request.form['password'])
7
8         try:
9             existing = users_table.get_item(Key={'email': email})
10            if 'Item' in existing:
11                flash("Email already registered.", "danger")
12                return redirect(url_for('signup'))
13
14            users_table.put_item(Item={
15                'email': email,
16                'name': name,
17                'password': password,
18                'created_at': datetime.utcnow().isoformat()
19            })
20
21            session['email'] = email
22            session['name'] = name
23
24            subject = "Welcome to Our Pickle & Snacks Store!"
25            body = f"Hello {name},\n\nThank you for signing up! Enjoy shopping your favorite pickles and snacks.\n\nTeam PickleMart"
26            send_email(email, subject, body)
27
28            flash("Account created!", "success")
29            return redirect(url_for('home'))
30        except Exception as e:
31            logger.error(f"Signup error: {e}")
32            flash("Error creating account.", "danger")
33
34    return render_template('signup.html')

```



login Route (GET/POST):

```
@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        email = request.form['email']
        password = request.form['password']

    try:
        result = users_table.get_item(Key={'email': email})
        user = result.get('Item')

        if user and check_password_hash(user['password'], password):
            session['email'] = email
            session['name'] = user['name']
            flash("Logged in successfully!", "success")
            return redirect(url_for('home'))
        else:
            flash("Invalid credentials.", "danger")
    except Exception as e:
        logger.error(f"Login error: {e}")
        flash("Login failed.", "danger")

    return render_template('login.html')
```

□ **logout route:**

```
@app.route('/logout')
def logout():
    session.clear()
    flash("Logged out.", "info")
    return redirect(url_for('home'))
```

□ **Products Routes:**

```

@app.route('/products')
def products():
    category = request.args.get('category')
    try:
        if category:
            response = products_table.scan(
                FilterExpression="category = :c",
                ExpressionAttributeValues={":c": category}
            )
        else:
            response = products_table.scan()
    products = response.get('Items', [])
    except Exception as e:
        logger.error(f"Product load error: {e}")
        products = []
    return render_template('products.html', products=products)

```

Add to Cart Route:

```

@app.route('/add-to-cart', methods=['POST'])
def add_to_cart():
    if not is_logged_in():
        return jsonify({'success': False, 'message': 'Login required'})

    email = session['email']
    product_id = request.json.get('product_id')
    quantity = int(request.json.get('quantity', 1))

    if email not in cart_db:
        cart_db[email] = {}

    cart_db[email][product_id] = cart_db[email].get(product_id, 0) + quantity
    return jsonify({'success': True, 'message': 'Added to cart'})

```

Remove from Cart :

```
@app.route('/remove-from-cart', methods=['POST'])
def remove_from_cart():
    if not is_logged_in():
        return jsonify({'success': False, 'message': 'Login required'})

    email = session['email']
    product_id = request.json.get('product_id')

    try:
        if email in cart_db and product_id in cart_db[email]:
            del cart_db[email][product_id]
            return jsonify({'success': True, 'message': 'Item removed from cart'})
        else:
            return jsonify({'success': False, 'message': 'Item not found in cart'})
    except Exception as e:
        logger.error(f"Remove from cart error: {e}")
    return jsonify({'success': False, 'message': 'Error removing item'})
```

Update Cart:

```
@app.route('/update-cart', methods=['POST'])
def update_cart():
    if not is_logged_in():
        return jsonify({'success': False, 'message': 'Login required'})

    email = session['email']
    product_id = request.json.get('product_id')
    quantity = int(request.json.get('quantity', 1))

    try:
        if email not in cart_db:
            cart_db[email] = {}

        if quantity > 0:
            cart_db[email][product_id] = quantity
            return jsonify({'success': True, 'message': 'Cart updated'})
        else:
            # Remove item if quantity is 0 or less
            if product_id in cart_db[email]:
                del cart_db[email][product_id]
            return jsonify({'success': True, 'message': 'Item removed from cart'})
    except Exception as e:
        logger.error(f"Update cart error: {e}")
        return jsonify({'success': False, 'message': 'Error updating cart'})
```

Checkout route:

The screenshot shows a code editor window with the following details:

- Title Bar:** HomeMade Pickles & Snacks Taste The Best
- File Menu:** File, Edit, Selection, View, Go, ...
- Toolbar:** Standard icons for file operations.
- Explorer:** Shows a project structure:
 - HOME MADE PICKLES & SNACKS TASTE THE BEST
 - templates (base.html, category.html, index.html, login.html, products.html, signup.html)
 - .env
 - app.py
- Code Editor:** The app.py file contains Python code for a Flask application. It includes routes for checkout, session management, and order processing. It uses a database table named orders_table and interacts with AWS services like SNS and S3. The code is annotated with line numbers (311-369).
- Status Bar:** Shows the current file is app.py, line 28, column 65, spaces 4, encoding UTF-8, CR/LF, and the system status (27°C, 07:30 PM, 03-07-2025).

Deployment Code:

```
if __name__ == '__main__':
    app.run(debug=True, host='0.0.0.0', port=5000)
```

Milestone 5: IAM Role Setup

Activity 5.1: Create IAM Role.

- In the AWS Console, go to IAM and create a new IAM Role for EC2 to interact with DynamoDB and SNS.

The screenshot illustrates the AWS Identity and Access Management (IAM) service interface. At the top, a search bar shows the results for 'iam'. The left sidebar lists various navigation options: Services, Features, Resources (New), Documentation, Knowledge articles, Marketplace, Blog posts, Events, and Tutorials. The main content area displays four service cards:

- IAM** (Manage access to AWS resources)
- IAM Identity Center** (Manage workforce user access to multiple AWS accounts and cloud applications)
- Resource Access Manager** (Share AWS resources with other accounts or AWS Organizations)
- AWS App Mesh** (Easily monitor and control microservices)

Below this, the 'Identity and Access Management (IAM)' dashboard is shown. It features a search bar for 'Search IAM' and a 'Roles' section with a search bar for 'Role name'. The 'Trusted entities' and 'Last activity' filters are also present. A large central window titled 'Select trusted entity' shows the 'Trusted entity type' dropdown set to 'AWS account'. It lists several AWS accounts as potential trusted entities. The 'Add account' button is visible at the bottom right.

The bottom portion of the screenshot shows the 'Create role' wizard. Step 1, 'Select trusted entity', is completed. Step 2, 'Add permissions', is currently active. It displays the 'Permissions policies' section with two managed policies selected: 'AmazonCloudWatchLogsFullAccess' and 'AmazonCloudWatchLogsWriteAccess'. The 'Filter by Type' dropdown is set to 'All types' and shows '2 matches'. The 'Type' column indicates both are 'AWS managed'. A note at the bottom states 'Set permissions boundary - optional'.



Activity 5.2: Attach Policies.

Attach the following policies to the role:

AmazonDynamoDBFullAccess: Allows EC2 to perform read/write operations on DynamoDB.

AmazonSNSFullAccess: Grants EC2 the ability to send notifications via SNS.

The screenshot shows the 'Add permissions' step of the IAM role creation wizard. In the search bar, 'AmazonDynamoDBFullAccess' is typed. The results list several AWS managed policies, with 'AmazonDynamoDBFullAccess' being selected. Other visible policies include 'AmazonVPCFullAccess', 'AmazonSNSRole', 'AWSLambdaBasicExecutionRole', and 'AWSLambdaVPCExecutionRole'. Below the search results, there is a section for setting a permissions boundary, which is currently empty. At the bottom right, there are 'Cancel', 'Previous', and 'Next' buttons.

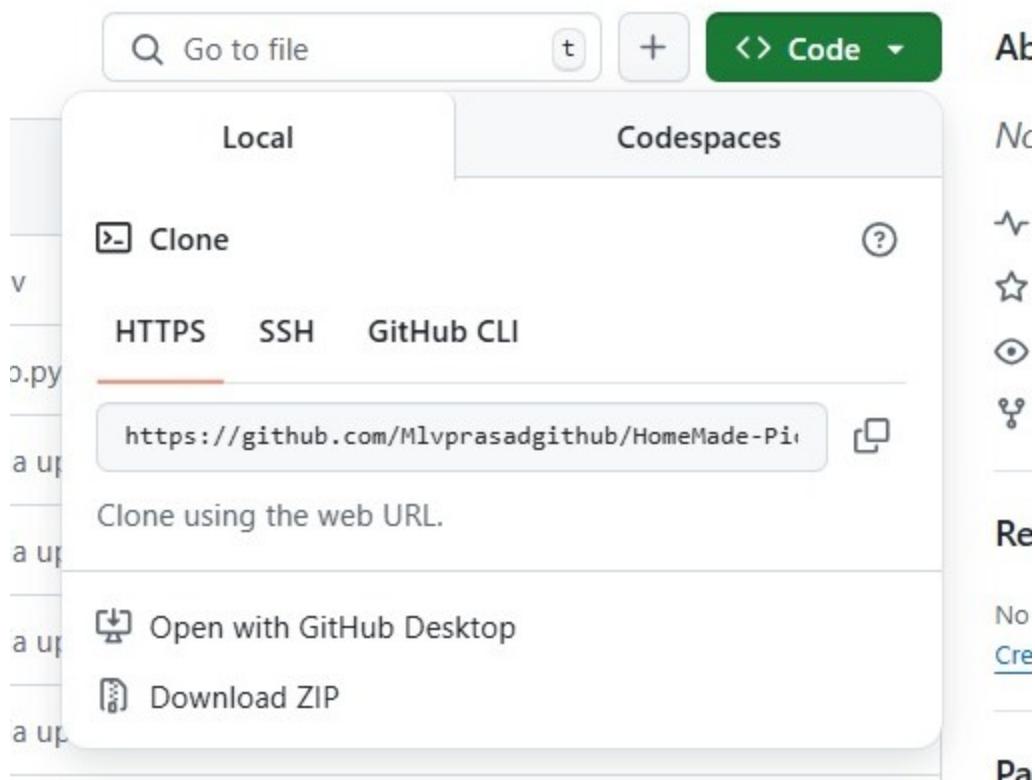
The screenshot shows the final 'Name, review, and create' step of the IAM role creation wizard. It includes sections for 'Role details' (role name: 'EC2-SampleRole'), 'Trust policy' (JSON trust policy allowing EC2 to assume the role), 'Add permissions' (summary of attached policies: 'AmazonDynamoDBFullAccess' and 'AmazonSNSFullAccess'), and 'Add tags' (empty tag list). At the bottom right, there are 'Cancel', 'Previous', and 'Create role' buttons.

The screenshot shows the AWS IAM Roles page with the role 'sns_Dynamodb_role' selected. The 'Summary' tab is active, displaying details such as creation date (October 13, 2024, 23:06 UTC+05:30), ARN (arn:aws:iam::557690616836:role/sns_Dynamodb_role), instance profile ARN (arn:aws:iam::557690616836:instance-profile/sns_Dynamodb_role), and last activity (6 days ago). The 'Permissions' tab is selected, showing two managed policies attached: 'AmazonDynamoDBFullAccess' and 'AmazonSNSFullAccess'. There are buttons for 'Edit', 'Delete', and 'Add permissions'.

Milestone 6: EC2 Instance Setup

Note: Load your Flask app and Html files into GitHub repository.

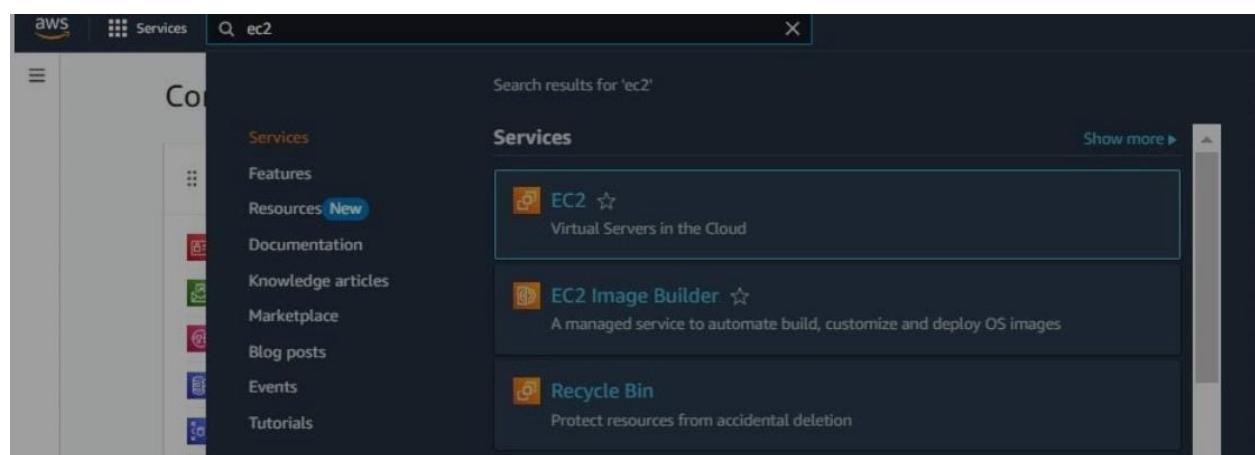
Mlvprasadbgithub Update .env		
	5a61891 · 5 hours ago	31 Commits
.env	Update .env	5 hours ago
app.py	Update app.py	5 hours ago
base.html	Add files via upload	11 hours ago
category.html	Add files via upload	11 hours ago
index.html	Add files via upload	11 hours ago
login.html	Add files via upload	11 hours ago
products.html	Add files via upload	11 hours ago
signup.html	Add files via upload	11 hours ago



☐ **Activity 6.1: Launch an EC2 instance to host the Flask application.**

☐ **Launch EC2 Instance**

☐ In the AWS Console, navigate to EC2 and launch a new instance.



☐ Click on Launch instance to launch EC2 instance

Screenshot of the AWS EC2 Instances page showing no instances and a "Launch instances" button. Below this, the "Launch an instance" wizard is shown with a "Name and tags" step. The "Name" field contains "InstantLibraryApp". To the right, a summary shows 1 instance, AMI details (Amazon Linux 2023.5.2), instance type (t2.micro), and security group (Firewall).

Launch an instance

Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

Name and tags

Name: InstantLibraryApp

Add additional tags

Summary

Number of instances: 1

Software Image (AMI): Amazon Linux 2023.5.2...read more
ami-078264b8ba71bc45e

Virtual server type (instance type): t2.micro

Firewall (security group)

Amazon Machine Image (AMI)

Amazon Linux 2023 AMI

Free tier eligible

ami-02b49a24cfb95941c (64-bit (x86), uefi-preferred) / ami-04ad8c7fcc828fad4 (64-bit (Arm), uefi)
Virtualization: hvm ENA enabled: true Root device type: ebs

Description

Amazon Linux 2023 is a modern, general purpose Linux-based OS that comes with 5 years of long term support. It is optimized for AWS and designed to provide a secure, stable and high-performance execution environment to develop and run your cloud applications.

Architecture	Boot mode	AMI ID	Verified provider
64-bit (x86)	uefi-preferred	ami-02b49a24cfb95941c	Verified provider

Browse more AMIs

Including AMIs from AWS, Marketplace and the Community

- Create and download the key pair for Server access.

▼ Instance type [Info](#) | [Get advice](#)

Instance type

t2.micro	Free tier eligible		
Family: t2	1 vCPU	1 GiB Memory	Current generation: true
On-Demand Linux base pricing: 0.0124 USD per Hour			
On-Demand Windows base pricing: 0.017 USD per Hour			
On-Demand RHEL base pricing: 0.0268 USD per Hour			
On-Demand SUSE base pricing: 0.0124 USD per Hour			

Additional costs apply for AMIs with pre-installed software

▼ Key pair (login) [Info](#)

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

Key pair name - required

Select [Create new key pair](#)

Create key pair X

Key pair name
Key pairs allow you to connect to your instance securely.
 The name can include up to 255 ASCII characters. It can't include leading or trailing spaces.

Key pair type

RSA RSA encrypted private and public key pair

ED25519 ED25519 encrypted private and public key pair

Private key file format

.pem For use with OpenSSH

.ppk For use with PuTTY

⚠️ When prompted, store the private key in a secure and accessible location on your computer. You will need it later to connect to your instance. [Learn more](#) 

[Cancel](#) [Create key pair](#)

The screenshot shows the AWS Lambda function configuration interface. At the top, there is a file icon and the function name **homemade.pem**. Below the name, there is a description of the function: "Amazon Linux 2023 is a modern, general purpose Linux-based OS that comes with 5 years of long term support. It is optimized for AWS and designed to provide a secure, stable and high-performance execution environment to develop and run your cloud applications." The configuration section includes fields for Architecture (64-bit (x86)), Boot mode (uefi-preferred), AMI ID (ami-078264b8ba71bc45e), Username (ec2-user), and a Verified provider badge. The Instance type section shows a t2.micro instance selected, with details about its family, vCPUs, memory, and current generation. It also indicates that it is free tier eligible and lists other On-Demand options. The Key pair (login) section shows InstantLibrary selected as the key pair name. The Summary section indicates 1 instance will be launched. The Software Image (AMI) section shows the Amazon Linux 2023 AMI. The Virtual server type (instance type) is set to t2.micro. The Firewall (security group) is set to New security group. The Storage (volumes) section shows 1 volume(s) - 8 GiB. A callout box provides information about the Free tier, stating: "Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free tier AMIs per month, 750 hours of public IPv4 address usage per month, 30 GiB of EBS storage, 2 million I/Os, 1 GB of snapshots, and 100 GiB of bandwidth to the internet." At the bottom, there are buttons for Cancel, Preview code, and Launch instance.



Activity 6.2:Configure security groups for HTTP, and SSH access.

▼ Network settings [Info](#)

VPC - required [Info](#)

vpc-03cdc7b6f19dd7211	(default) ▾	
172.31.0.0/16		

Subnet [Info](#)

No preference	▼	
---------------	---	--

Auto-assign public IP [Info](#)

Enable	▼
--------	---

Additional charges apply when outside of free tier allowance

Firewall (security groups) [Info](#)

A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

<input checked="" type="radio"/> Create security group	<input type="radio"/> Select existing security group
--	--

Security group name - required

launch-wizard

This security group will be added to all network interfaces. The name can't be edited after the security group is created. Max length is 255 characters. Valid characters: a-z, A-Z, 0-9, spaces, and _-:/()#@[]+=&;()!\$*

Description - required [Info](#)

launch-wizard created 2024-10-13T17:49:56.622Z
--

Inbound Security Group Rules

▼ Security group rule 1 (TCP, 22, 0.0.0.0/0)

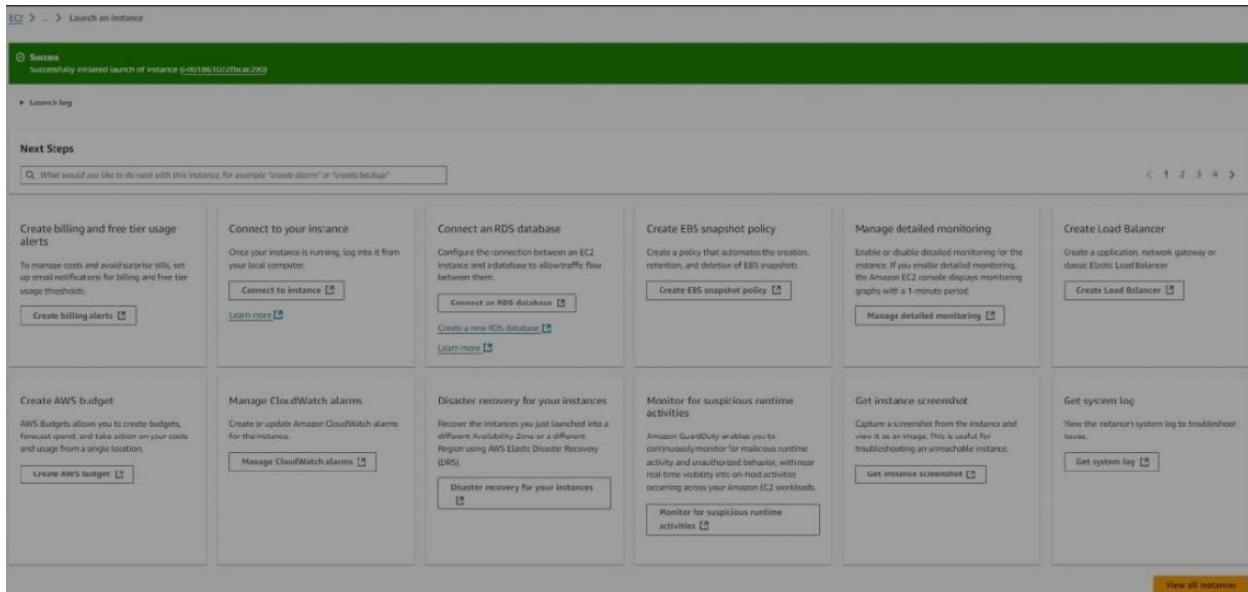
Type Info ssh	Protocol Info TCP	Port range Info 22	
Source type Info Anywhere	Source Info <input type="text"/> Add CIDR, prefix list or security 0.0.0.0/0	Description - optional Info e.g. SSH for admin desktop	

▼ Security group rule 2 (TCP, 80, 0.0.0.0/0)

Type Info HTTP	Protocol Info TCP	Port range Info 80	
Source type Info Custom	Source Info <input type="text"/> Add CIDR, prefix list or security 0.0.0.0/0	Description - optional Info e.g. SSH for admin desktop	

▼ Security group rule 3 (TCP, 5000, 0.0.0.0/0)

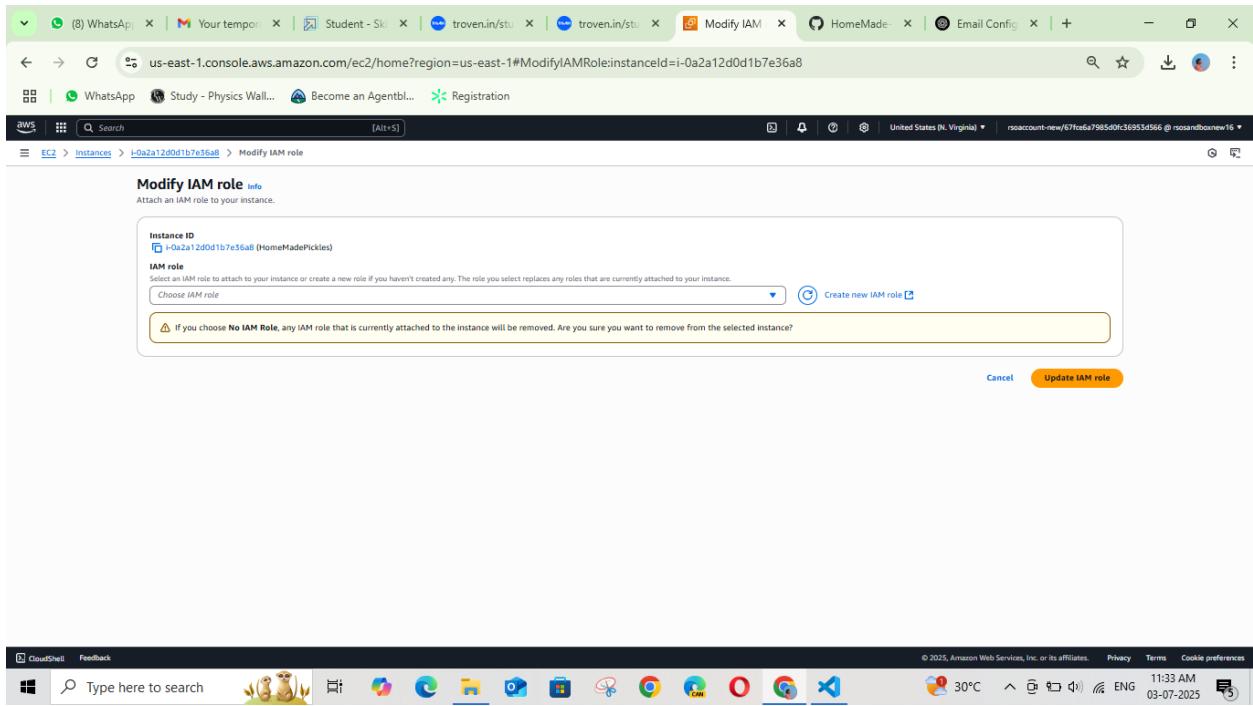
Type Info Custom TCP	Protocol Info TCP	Port range Info 5000	
Source type Info Custom	Source Info <input type="text"/> Add CIDR, prefix list or security 0.0.0.0/0	Description - optional Info e.g. SSH for admin desktop	



To connect to EC2 using **EC2 Instance Connect**, start by ensuring that an **IAM role** is attached to your EC2 instance. You can do this by selecting your instance, clicking on Actions, then navigating to **Security** and selecting **Modify IAM Role** to attach the appropriate role. After the IAM role is connected, navigate to the EC2 section in the **AWS Management Console**. Select the **EC2 instance** you wish to connect to. At the top of the **EC2 Dashboard**, click the **Connect** button. From the connection methods presented, choose **EC2 Instance Connect**. Finally, click **Connect** again, and a new browser-based terminal will open, allowing you to access your EC2 instance directly from your browser.

The screenshot shows the AWS EC2 Instances page. On the left, the navigation pane is open with the 'Instances' section selected. The main area displays a table titled 'Instances (1/1) Info'. A single instance is listed: 'HomeMadePic...', with the ID 'i-0a2a12d0d1b7e56a8'. The instance is 'Running' in the 't2.micro' type. It has a Public IPv4 address of '54.90.214.21' and a Private IP address of '172.31.18.29'. The Public DNS is 'ec2-54-90-214-21.compute-1.amazonaws.com'. The status bar at the bottom right shows the date and time as '03-07-2025 11:19 AM'.

The screenshot shows the AWS IAM page. The left sidebar is open with the 'Instances' section selected. The main area displays the 'Services' section, which includes links for IAM, IAM Identity Center, and Resource Access Manager. Below this is the 'Features' section with links for Groups, Roles, and Identity providers. A modal window titled 'Introducing resource search' is open at the bottom. On the right, the 'Actions' button in the top right corner of the main content area has a context menu open, showing options like 'Instance diagnostics', 'Instance settings', 'Networking', 'Security', 'Image and templates', and 'Modify IAM role'. The status bar at the bottom right shows the date and time as '03-07-2025 11:19 AM'.



 Now connect the EC2 with the files

Connect to instance Info

Connect to your instance i-001861022fbcac290 (InstantLibraryApp) using any of these options

EC2 Instance Connect | **Session Manager** | **SSH client** | **EC2 serial console**

⚠ Port 22 (SSH) is open to all IPv4 addresses

Port 22 (SSH) is currently open to all IPv4 addresses, indicated by **0.0.0.0/0** in the inbound rule in [your security group](#). For increased security, consider restricting access to only the EC2 Instance Connect service IP addresses for your Region: **13.233.177.0/29**. [Learn more](#).

Instance ID

Connection Type

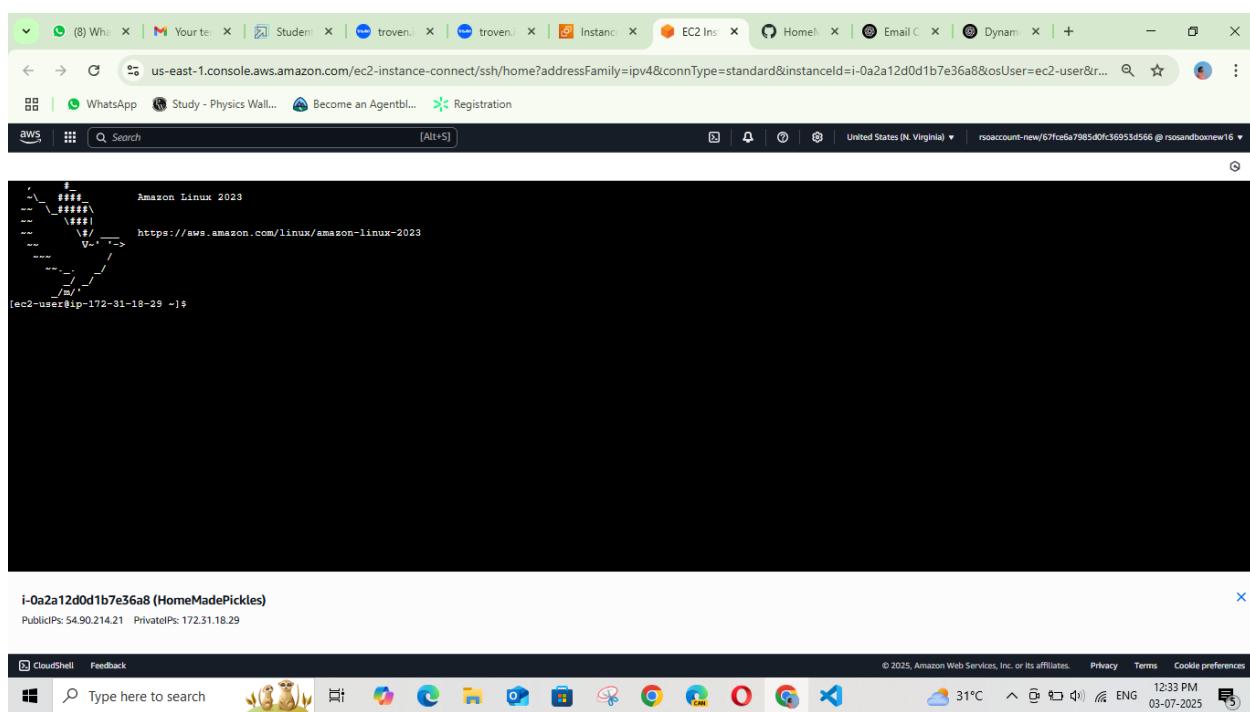
- Connect using EC2 Instance Connect**
Connect using the EC2 Instance Connect browser-based client, with a public IPv4 or IPv6 address.
- Connect using EC2 Instance Connect Endpoint**
Connect using the EC2 Instance Connect browser-based client, with a private IPv4 address and a VPC endpoint.
- Public IPv4 address**
- IPv6 address**

Username
Enter the username defined in the AMI used to launch the instance. If you didn't define a custom username, use the default username, ec2-user.

×

Note: In most cases, the default username, ec2-user, is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI username.

Cancel
Connect



Milestone 7: Deployment on EC2

Activity 7.1: Install Software on the EC2 Instance

Install Python3, Flask, and Git:

On Amazon Linux 2:

```
sudo yum update -y  
sudo yum install python3 git  
sudo pip3 install flask boto3
```

Verify Installations:

```
flask --version git --version
```

Activity 7.2: Clone Your Flask Project from GitHub

Clone your project repository from GitHub into the EC2 instance using Git.

Note: change your-github-username and your-repository-name with your credentials here: '[git clone https://github.com/Mlvprasadgithub/HomeMade-Pickles-Snacks-Taste-The-Best.git](https://github.com/Mlvprasadgithub/HomeMade-Pickles-Snacks-Taste-The-Best.git)'

 This will download your project to the EC2 instance.

To navigate to the project directory, run the following command:

```
cd HomeMade Pickles & Snacks Taste the Best
```

Once inside the project directory, configure and run the Flask application by executing the following command with elevated privileges:

Run the Flask Application

```
sudo flask run --host=0.0.0.0 --port=80
```

```

.
├── Amazon Linux 2023
│   ├── https://aws.amazon.com/linux/amazon-linux-2023
└── [ec2-user@ip-172-31-18-29 ~]$

```

i-0a2a12d0d1b7e36a8 (HomeMadePickles)
PublicIPs: 54.90.214.21 PrivateIPs: 172.31.18.29

cloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Verify the Flask app is running: <http://your-ec2-public-ip>

Run the Flask app on the EC2 instance

```

sudo pip3 install flask boto3
sudo yum install python3-pip -y
pip3 install Flask
pip3 install boto3
pip3 install https://dl.fedoraproject.org/moredeps/
Amazon Linux 2023 Kernel. Livepatch repository
Dependencies resolved.
Nothing to do.
Complete!
Last metadata expiration check: 0:00:02 ago on Thu Jul  3 07:06:26 2025.
Package python3-3.9.23-1.amzn2023.0.1.x86_64 is already installed.
Dependencies resolved.

=====
Package           Architecture      Version       Repository    Size
=====
Installing:
git              x86_64          2.47.1-1.amzn2023.0.3  amazonlinux  52 k
Installing dependencies:
git-core          x86_64          2.47.1-1.amzn2023.0.3  amazonlinux  4.5 M
git-core-doc      noarch         2.47.1-1.amzn2023.0.3  amazonlinux  2.8 M
perl-Error        noarch         1:0.17029-5.amzn2023.0.2  amazonlinux  41 k
perl-File-Find    noarch         1.37-477.amzn2023.0.7   amazonlinux  25 k
perl-Git          noarch         2.47.1-1.amzn2023.0.3   amazonlinux  40 k
perl-MIMEReadKey  x86_64          2.38-1.amzn2023.0.1   amazonlinux  30 k
perl-Lib          x86_64          0.65-477.amzn2023.0.7   amazonlinux  15 k

Transaction Summary
Install  8 Packages

Total download size: 7.5 M
Installed size: 37 M
Is this ok [y/N]: [y]

i-0a2a12d0d1b7e36a8 (HomeMadePickles)
PublicIPs: 54.90.214.21 PrivateIPs: 172.31.18.29

```

cloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Access the website through:

PublicIPs: <http://127.0.0.1:5000/>

Milestone 8: Testing and Deployment

Activity 8.1: Conduct functional testing to verify customer registration, login, pickles and snacks requests, and notifications.

Home Page:



Register Page:

The screenshot shows the 'Join PickleMart!' sign-up form. At the top center is a green circular icon with a white person symbol and a plus sign. Below it, the text 'Join PickleMart!' is displayed in bold. A sub-instruction 'Create your account and start shopping authentic pickles' follows. The form includes fields for 'Full Name' (placeholder 'Enter your full name'), 'Email Address' (placeholder 'Enter your email'), 'Password' (placeholder 'Create a strong password'), and 'Confirm Password' (placeholder 'Confirm your password'). Below these are two checkboxes: 'I agree to the Terms & Conditions and Privacy Policy' and 'Subscribe to our newsletter for updates and special offers'. A large green 'CREATE ACCOUNT' button with a person icon is centered at the bottom. A small note at the very bottom says 'Already have an account? [Sign in here](#)'.

Login Page:

The screenshot shows the 'Welcome Back!' login form. At the top center is a green circular icon with a white person symbol. Below it, the text 'Welcome Back!' is displayed in bold. A sub-instruction 'Sign in to your PickleMart account' follows. The form includes fields for 'Email Address' (placeholder 'Enter your email') and 'Password' (placeholder 'Enter your password'). Below these is a checkbox for 'Remember me'. A large green 'SIGN IN' button with a right-pointing arrow icon is centered at the bottom. A small note at the very bottom says 'Don't have an account? [Sign up here](#)'.

About Us Page:

PickleMart Home All Products Categories Login Sign Up

What Our Customers Say

Real reviews from happy customers



"The mango pickle tastes just like my grandmother's recipe. Absolutely authentic and delicious!"

Priya Sharma
Mumbai



"Fast delivery and excellent packaging. The snacks arrived fresh and crispy. Highly recommended!"

Rajesh Kumar
Delhi



"Amazing quality and taste. The fish pickle is outstanding. Will definitely order again!"

Anjali Patel
Bangalore



Secure Payments
100% secure and encrypted transactions



Easy Returns
Hassle-free return policy within 7 days



24/7 Support
Round-the-clock customer support



Quality Assured
Premium quality guaranteed

Contact Us Page:

PickleMart Home All Products Categories Login Sign Up



Secure Payments
100% secure and encrypted transactions



Easy Returns
Hassle-free return policy within 7 days



24/7 Support
Round-the-clock customer support



Quality Assured
Premium quality guaranteed

PickleMart

Your one-stop destination for authentic pickles and traditional snacks.

Quick Links

- [Home](#)
- [Products](#)
- [Veg Pickles](#)
- [Snacks](#)

Contact Info

- info@picklemart.com
- +91 98765 43210
-

© 2025 PickleMart. All rights reserved.

All Products Page:

All Products

Explore our complete collection of authentic pickles and traditional snacks

Search products..

All Categories

Sort by Name

Showing 10 products



Traditional Mango Pickle

Authentic homemade mango pickle with traditional spices and mustard oil...

★★★★★ 4.8 (156)

₹160.00

₹120.00

In Stock



Spicy Chicken Pickle

Tender chicken pieces marinated in aromatic spices and traditional recipe...

★★★★★ 4.5 (89)

₹200.00

₹250.00

In Stock



Mixed Vegetable Pickle

Colorful mix of seasonal vegetables pickled with authentic spices...

★★★★★ 4.2 (67)

₹95.00

₹120.00

In Stock



Crispy Samosas (6 pcs)

Golden, crispy samosas filled with spiced potatoes and peas...

★★★★★ 4.7 (134)

₹80.00

In Stock



Tangy Lemon Pickle

Fresh lemons preserved in aromatic spices and salt for the perfect tang...

★★★★★ 4.3 (78)

₹110.00

₹100.00

In Stock



Premium Mutton Pickle

Tender mutton pieces cooked in rich spices and preserved traditionally...

★★★★★ 4.9 (45)

₹380.00

₹400.00

In Stock



Spicy Garlic Pickle

Pungent garlic cloves pickled with red chillies and traditional spices...



Coastal Fish Pickle

Fresh fish marinated in coastal spices and traditional recipe...



Premium Prawn Pickle

Fresh prawns cooked in rich coconut oil with traditional coastal spices...

Sweet Carrot Pickle
Crunchy carrots pickled with jaggery and mild spices for a sweet tang...
₹85.00 In Stock

Mixed Vegetable Pickle
A colorful mix of seasonal vegetables pickled with authentic spices...
₹200.00 In Stock

Tangy Lemon Pickle
Fresh lemons preserved in aromatic spices and salt for the perfect tang...
₹320.00 1 left!

Mixed Vegetable Pickle
A colorful mix of seasonal vegetables pickled with authentic spices...
₹140.00 In Stock

Previous 1 2 3 Next

Vegetarian Pickles Page:

Traditional Mango Pickle
Authentic homemade mango pickle with traditional spices and mustard oil...
₹120.00 In Stock

Mixed Vegetable Pickle
A colorful mix of seasonal vegetables pickled with authentic spices...
₹95.00 In Stock

Tangy Lemon Pickle
Fresh lemons preserved in aromatic spices and salt for the perfect tang...
₹110.00 In Stock

Mixed Vegetable Pickle
A colorful mix of seasonal vegetables pickled with authentic spices...
₹140.00 In Stock

Non-Vegetarian Pickles Page:

Search products... Non-Veg Pickles Sort by Name

Showing 4 products



Spicy Chicken Pickle
Tender chicken pieces marinated in aromatic spices and traditional recipe...

★★★★★ 4.5 (89)
₹250.00 In Stock

Add to Cart



Premium Mutton Pickle
Tender mutton pieces cooked in rich spices and preserved traditionally...

★★★★★ 4.9 (45)
₹380.00 In Stock

Add to Cart



Coastal Fish Pickle
Fresh fish marinated in coastal spices and traditional recipe...

★★★★★ 4.4 (56)
₹200.00 In Stock

Add to Cart

Traditional Snacks Page:

Search products... Traditional Snacks Sort by Name

Showing 1 products



Crispy Samosas (6 pcs)
Golden, crispy samosas filled with spiced potatoes and peas...

★★★★★ 4.7 (134)
₹80.00 In Stock

Add to Cart

Previous 1 2 3 Next

Pickles Ordered Successfully! we will get back to you soon.

Exit:

Session Ended
Please close the tab.

Dynamodb Database Updations:

█ Users table :

	email (String)	login_count	name	password
□	ponubakulaalekhya@...	1	Alekhya	\$2b\$12\$e6CDh0Sje.nTmUF9DFSkouBvOvxGxd13MrMT3rSgmzAlphy/bT6
□	alekhya080228@gma...	1	Alekhya	\$2b\$12\$mlJrhDAh1OMt7lcSKQlau4ytsOCJPXifdQ6HdvYg6qU16G0kRhd2
□	sirichakkala@gmail.com	1	Siri Chakkala	\$2b\$12\$IVMIRmgrb0tUp8U3Kd0hhrOUhSw5/ApxgSLof7k63cvWSWqOcDgR6

█ Orders table :

	email (String)	book_na...	description	name	roll_no	semester	subject	year
□	ponubakulaalekhya@...	python	less stock	Alekhya	1234	5	Data Science	3
□	alekhya080228@gma...	applied stats	I need this bo...	Alekhya	1234	3	Statistics	2
□	sirichakkala@gmail.com	Python Pro...	Hello	Siri Chakkala	12	2	Data Science	1

Products Table :

	email (String)	book_na...	description	name	roll_no	semester	subject	year
□	ponubakulaalekhya@...	python	less stock	Alekhya	1234	5	Data Science	3
□	alekhya080228@gma...	applied stats	I need this bo...	Alekhya	1234	3	Statistics	2
□	sirichakkala@gmail.com	Python Pro...	Hello	Siri Chakkala	12	2	Data Science	1

Conclusion:

The HomeMade Pickles and Snacks - Taste the Best web application successfully demonstrates the power of modern web development using Flask and AWS services to build a scalable, user-friendly e-commerce platform. By focusing on the niche of traditional, homemade pickles and snacks, the project not only highlights cultural authenticity but also bridges local products to a broader online market.

Key accomplishments of the project include:

- ❑ A fully functional **Flask-based backend** with **DynamoDB integration** for fast and scalable data storage.
- ❑ **User-friendly interfaces** for product browsing, cart management, and order placement.
- ❑ Secure **email notifications** using **AWS SNS** and **SMTP** for order confirmations and user interactions.
- ❑ Cleanly organized project structure, designed with deployment in mind for platforms like **AWS EC2**.
- ❑ Emphasis on **security, modularity, and readability** by using environment variables and best practices.

This project provides a solid foundation for future enhancements, such as adding payment gateways, user authentication, order tracking, and more advanced analytics. It also demonstrates readiness for real-world deployment and a professional understanding of full-stack development principles.

With this project, we've created more than just a web app – we've built a digital storefront that brings the flavors of tradition to the fingertips of modern consumers.