

HomeMade Pickles & Snacks: Taste the Best

Project Description:

Home Made 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."

Scenarios:

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.

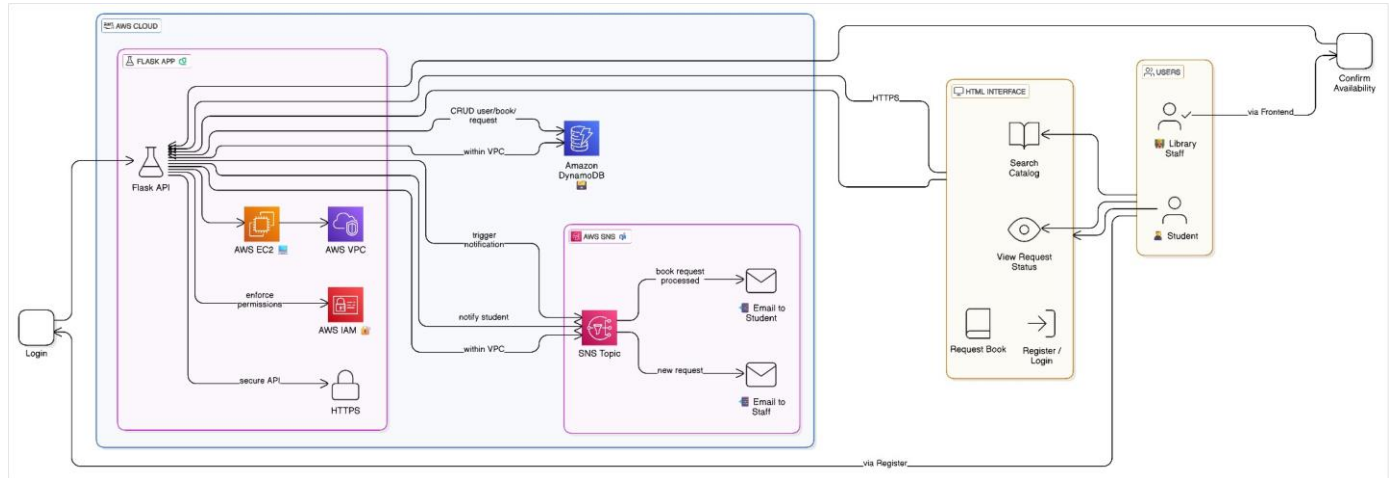
Scenario 2: Real-Time Inventory Tracking and Updates

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.

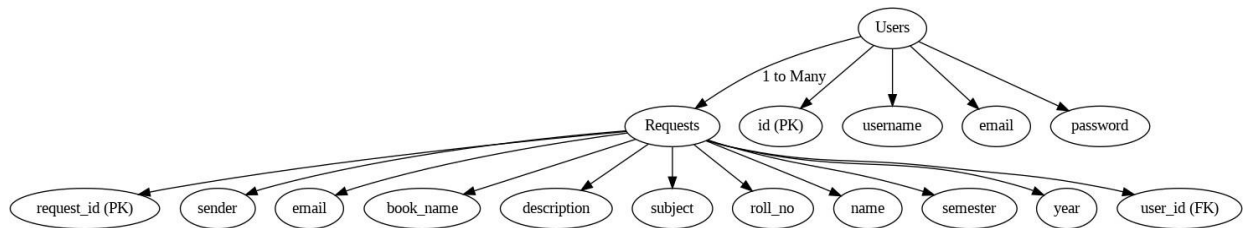
Scenario 3: Personalized User Experience and Recommendations

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. **SNS Overview:** [SNS Documentation](#)
6. **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 Database Creation 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

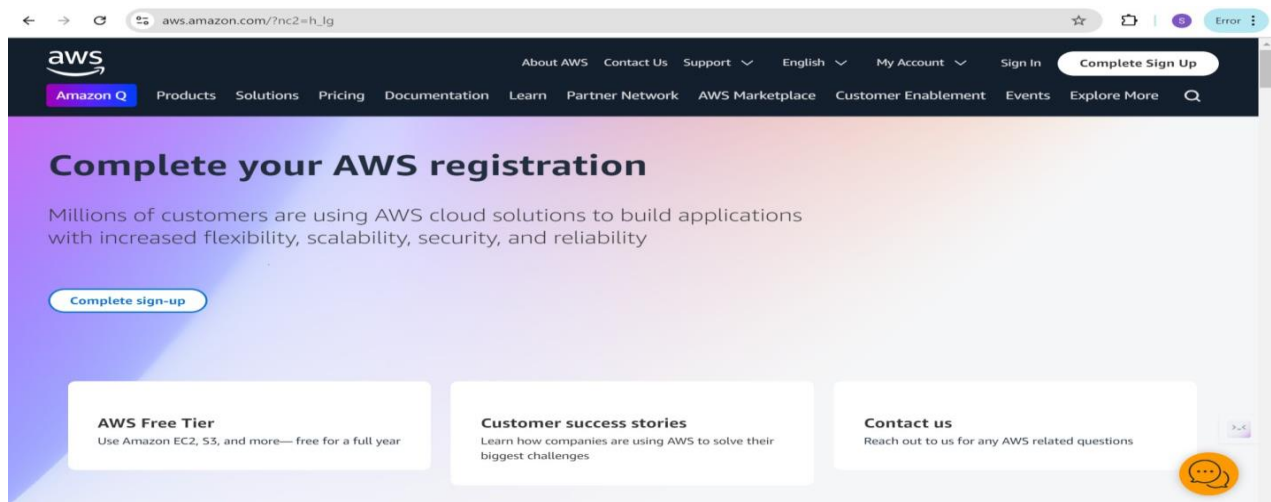
Activity 7.2: Run the Flask App

8. Testing and Deployment

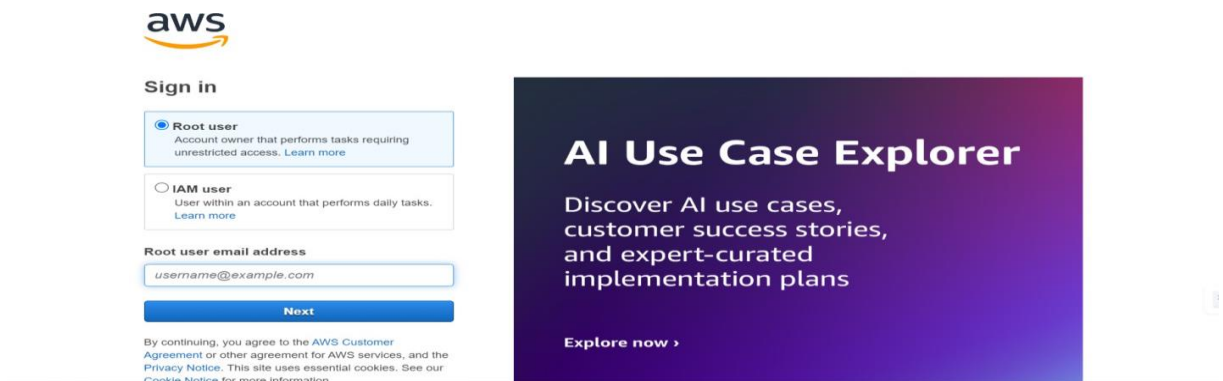
Activity 8.1: Conduct functional testing to verify user registration, login, book requests, and notifications.

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.

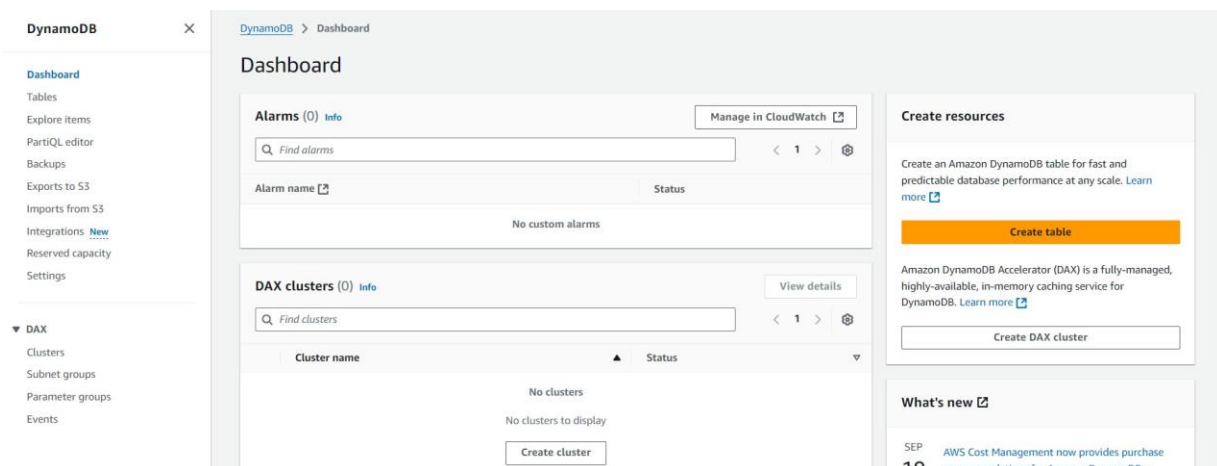
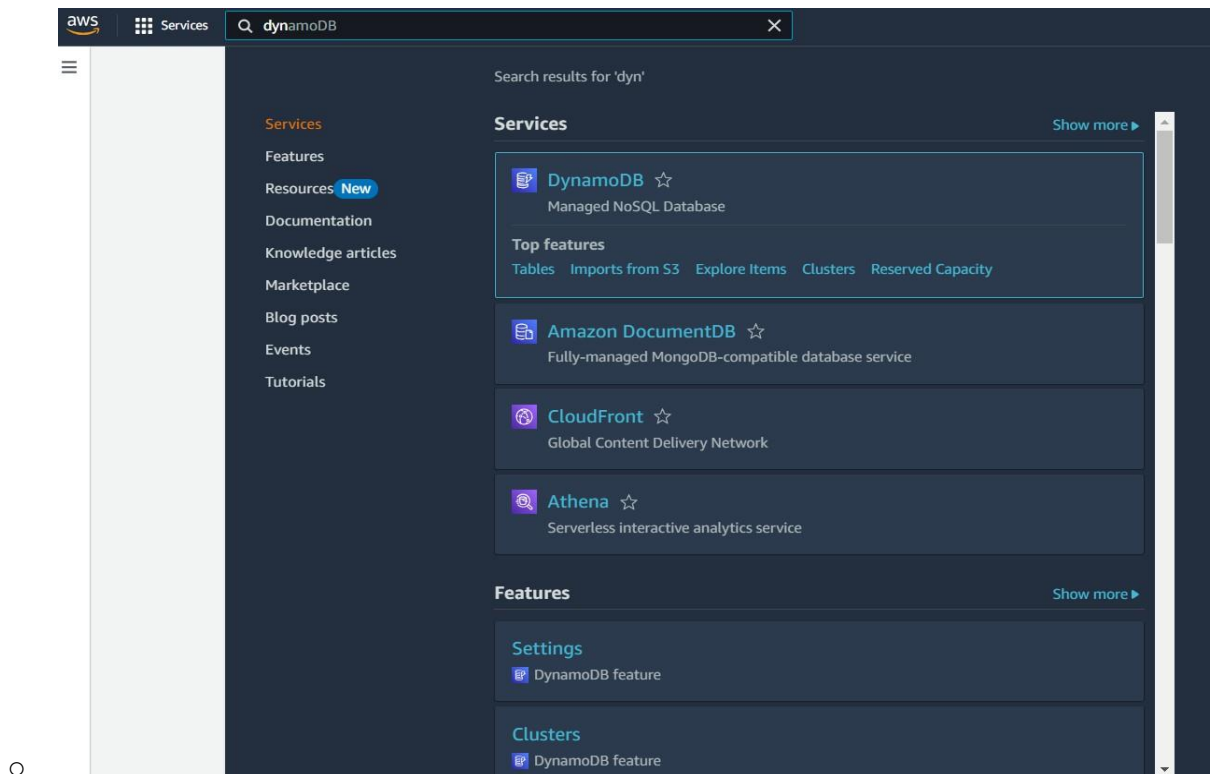


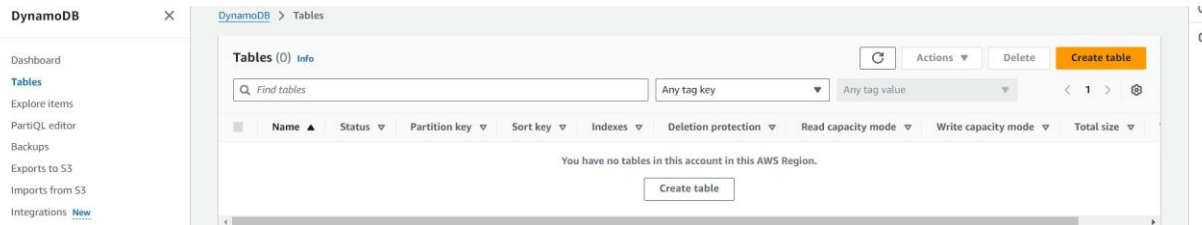
- **Activity 1.2: Log in to the AWS Management Console**
 - After setting up your account, log in to the [AWS Management Console](#).



Milestone 2: DynamoDB Database Creation and Setup

- **Activity 2.1: Navigate to the DynamoDB**
 - In the AWS Console, navigate to DynamoDB and click on create tables.





- **Activity 2.2: Create a DynamoDB table**

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

DynamoDB
Tables
Create table

Table details Info

DynamoDB is a schemaless database that requires only a table name and a primary key when you create the table.

Table name
This will be used to identify your table.

Between 3 and 255 characters, containing only letters, numbers, underscores (`_`), hyphens (`-`), and periods (`.`).

Partition key
The partition key is part of the table's primary key. It is a hash value that is used to retrieve items from your table and allocate data across hosts for scalability and availability.

String

1 to 255 characters and case sensitive.

Sort key - optional
You can use a sort key as the second part of a table's primary key. The sort key allows you to sort or search among all items sharing the same partition key.

String

1 to 255 characters and case sensitive.

Default table settings

These are the default settings for your new table. You can change some of these settings after creating the table.

Setting	Value	Editable after creation
Table class	DynamoDB Standard	Yes
Capacity mode	On-demand	Yes
Maximum read capacity units	-	Yes
Maximum write capacity units	-	Yes
Local secondary indexes	-	No
Global secondary indexes	-	Yes
Encryption key management	AWS owned key	Yes
Deletion protection	Off	Yes
Resource-based policy	Not active	Yes

Tags

Tags are pairs of keys and optional values, that you can assign to AWS resources. You can use tags to control access to your resources or track your AWS spending.

No tags are associated with the resource.

[Add new tag](#)

You can add 50 more tags.

 This table will be created with auto scaling deactivated. You do not have permissions to turn on auto scaling.

[Cancel](#)

[Create table](#)

 [DynamoDB](#) > [Tables](#)

DynamoDB

[Dashboard](#)

[Tables](#)

[Explore items](#)

[PartiQL editor](#)

[Backups](#)


[Exports to S3](#)

[Imports from S3](#)

[Integrations](#) [New](#)

[Reserved capacity](#)

[Settings](#)

 The user1 table was created successfully.

Tables (1/1) [Info](#)

Any tag key

Any tag value

< 1 > 

<input checked="" type="checkbox"/>	Name ▲	Status	Partition key	Sort key	Indexes	Replication Regions	Deletion protection	Favorite ▼	Re
<input checked="" type="checkbox"/>	user1	 Active	user name (S)	-	0	0	 Off		Or

- Follow the same steps to create an Orders table with Order_id as the primary key to store Order details.

DynamoDB > Tables > Create table

DynamoDB is a schemaless database that requires only a table name and a primary key when you create the table.

Table name

This will be used to identify your table.

Between 3 and 255 characters, containing only letters, numbers, underscores (_), hyphens (-), and periods (.).

Partition key

The partition key is part of the table's primary key. It is a hash value that is used to retrieve items from your table and allocate data across hosts for scalability and availability.

String

1 to 255 characters and case sensitive.

Sort key - optional

You can use a sort key as the second part of a table's primary key. The sort key allows you to sort or search among all items sharing the same partition key.

String

1 to 255 characters and case sensitive.

Table settings

☒ Default settings

The fastest way to create your table. You can modify most of these settings after your table has been created. To modify these settings now, choose 'Customize settings'.

☐ Customize settings

Use these advanced features to make DynamoDB work better for your needs.

[DynamoDB](#) > [Tables](#) > Create table

Default table settings

These are the default settings for your new table. You can change some of these settings after creating the table.

Setting	Value	Editable after creation
Table class	DynamoDB Standard	Yes
Capacity mode	On-demand	Yes
Maximum read capacity units	-	Yes
Maximum write capacity units	-	Yes
Local secondary indexes	-	No
Global secondary indexes	-	Yes
Encryption key management	AWS owned key	Yes
Deletion protection	Off	Yes
Resource-based policy	Not active	Yes

[DynamoDB](#) > [Tables](#)

[Dashboard](#)
[Tables](#)
[Explore items](#)
 [PartiQL editor](#)
[Backups](#)
[Exports to S3](#)
[Imports from S3](#)
[Integrations](#) [New](#)
[Reserved capacity](#)
[Settings](#)

DAX
[Clusters](#)
[Subnet groups](#)
[Parameter groups](#)
[Events](#)

The pickleorders table was created successfully.

Tables (2/2) [Info](#)

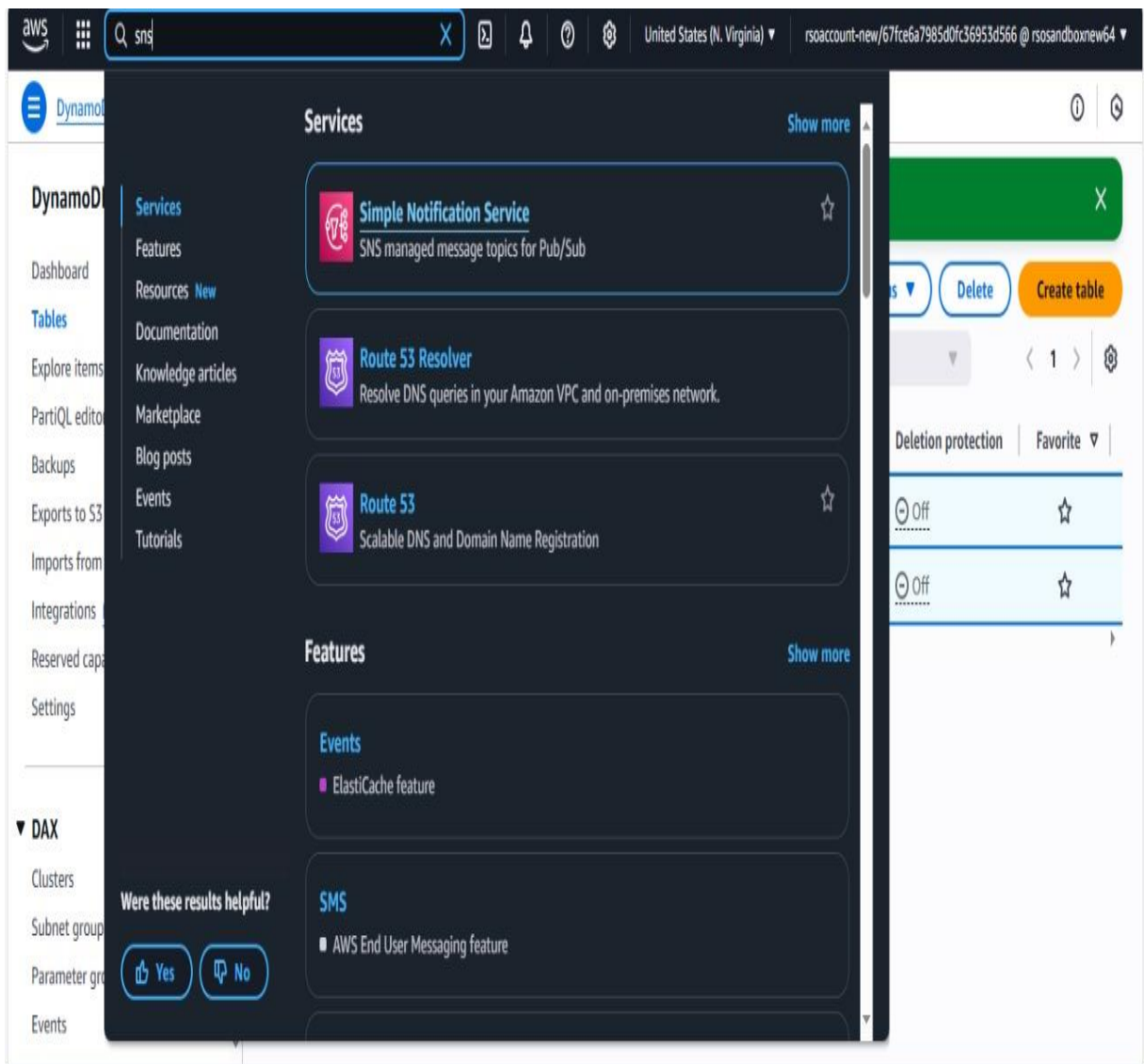
[Actions](#) [Delete](#) [Create table](#)

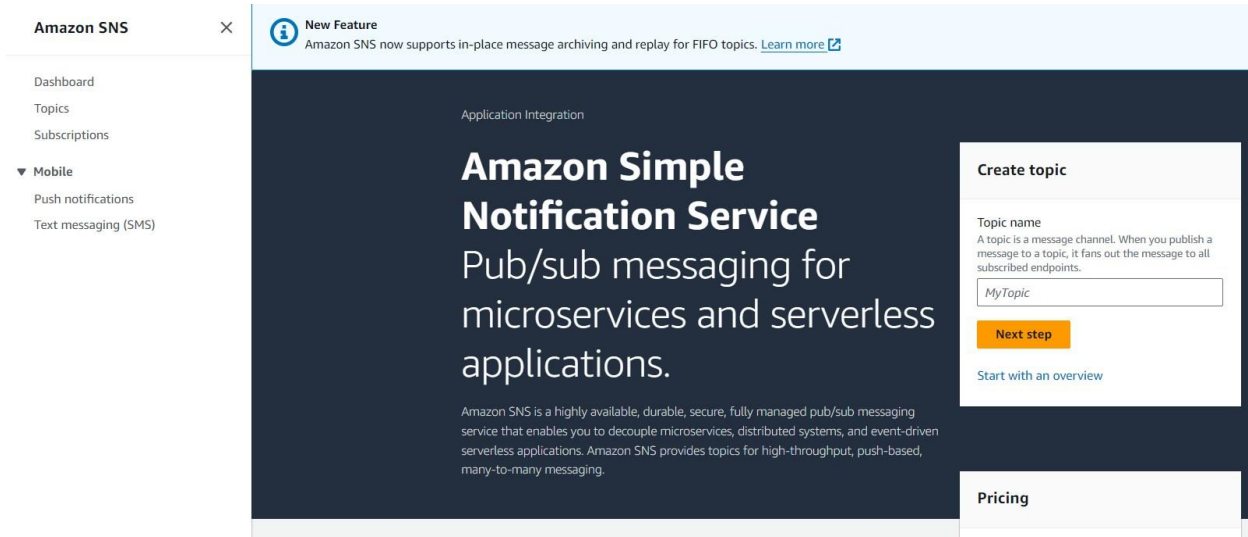
<input checked="" type="checkbox"/>	Name	Status	Partition key	Sort key	Indexes	Replication Regions	Deletion protection	Favorite
<input checked="" type="checkbox"/>	pickleorders	Active	orders_id (S)	-	0	0	Off	☆
<input checked="" type="checkbox"/>	user1	Active	user name (S)	-	0	0	Off	☆

Milestone 3: SNS Notification Setup

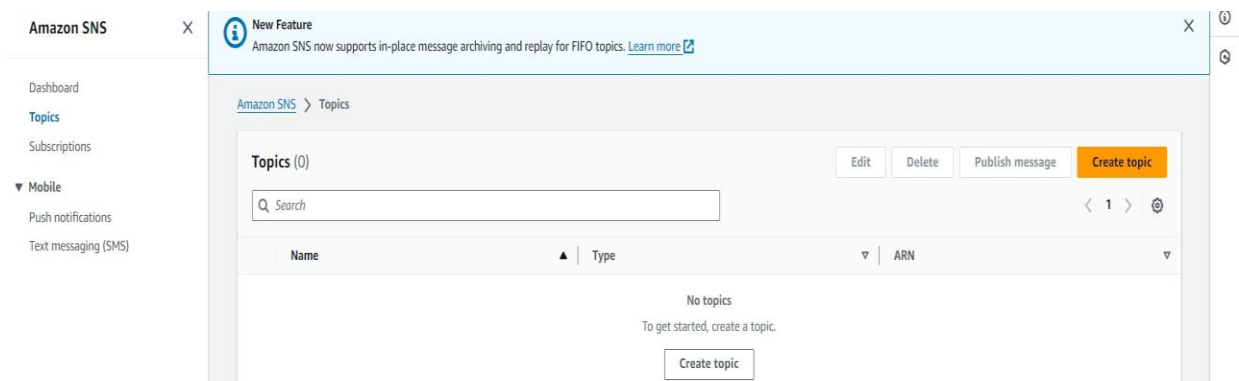
- **Activity 3.1: Create SNS topics for sending email notifications to users and library staff.**

- In the AWS Console, search for SNS and navigate to the SNS Dashboard.





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



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

Amazon SNS > Topics > Create topic

Details

Type

Info

Topic type cannot be modified after topic is created

☐ FIFO (first-in, first-out)

- Strictly-preserved message ordering
- Exactly-once message delivery
- Subscription protocols: SQS

☒ Standard

- Best-effort message ordering
- At-least once message delivery
- Subscription protocols: SQS, Lambda, Data Firehose, HTTP, SMS, email, mobile application endpoints

Name

homamadepickles

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

Display name - optional

Info

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**.

Amazon SNS

Dashboard

Topics

Subscriptions

▼ Mobile

Push notifications

Text messaging (SMS)

homemadepickles

Details

Name	homemadepickles	Display name	-
ARN	arn:aws:sns:us-east-1:216989138822:homemadepickles	Topic owner	216989138822
Type	Standard		

Subscriptions

Access policy

Data protection policy

Delivery policy (HTTP/S)

Delivery status logging

Subscriptions (0)

Edit

Delete

Request confirmation

Confirm subscription

Create subscription

- **Activity 3.2: Subscribe users and staff to relevant SNS topics to receive real-time notifications when a order request is made.**
 - Subscribe users (or customers) to this topic via Email. When a order request is made, notifications will be sent to the subscribed emails.
 -

Details

Topic ARN

arn:aws:sns:us-east-1:216989138822:homemadepickles

Protocol

The type of endpoint to subscribe

Email

Endpoint

An email address that can receive notifications from Amazon SNS.

vangadeepika929@gmail.com

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](#)

Amazon SNS

- Dashboard
- Topics
- Subscriptions

▼ **Mobile**

- Push notifications
- Text messaging (SMS)

New Feature

Amazon SNS now supports High Throughput FIFO topics. [Learn more](#)

Subscription to homemadepickles created successfully.

The ARN of the subscription is arn:aws:sns:us-east-1:216989138822:homemadepickles:e89352eb-94a7-4d73-8290-c0acafe98d1d.

Subscription: e89352eb-94a7-4d73-8290-c0acafe98d1d

Edit Delete

Details

<p>ARN</p> <p>arn:aws:sns:us-east-1:216989138822:homemadepickles:e89352eb-94a7-4d73-8290-c0acafe98d1d</p>	<p>Status</p> <p>⌚ Pending confirmation</p>
<p>Endpoint</p> <p>vangadeepika929@gmail.com</p>	<p>Protocol</p> <p>EMAIL</p>
<p>Topic</p> <p>homemadepickles</p>	
<p>Subscription Principal</p> <p>arn:aws:iam::216989138822:role/rsoaccount-new</p>	

- After subscription request for the mail confirmation

Amazon SNS

- Dashboard
- Topics
- Subscriptions

▼ **Mobile**

- Push notifications
- Text messaging (SMS)

New Feature

Amazon SNS now supports High Throughput FIFO topics. [Learn more](#)

Subscription: e89352eb-94a7-4d73-8290-c0acafe98d1d

Edit Delete

Details

<p>ARN</p> <p>arn:aws:sns:us-east-1:216989138822:homemadepickles:e89352eb-94a7-4d73-8290-c0acafe98d1d</p>	<p>Status</p> <p>✅ Confirmed</p>
<p>Endpoint</p> <p>vangadeepika929@gmail.com</p>	<p>Protocol</p> <p>EMAIL</p>
<p>Topic</p> <p>homemadepickles</p>	
<p>Subscription Principal</p> <p>arn:aws:iam::216989138822:role/rsoaccount-new</p>	

Subscription filter policy Redrive policy (dead-letter queue)

- 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 ▾

14:17 (0 minutes ago)

You have chosen to subscribe to the topic:
arn:aws:sns:us-east-1:216989138822:homemadepickles

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](#)



Simple Notification Service

Subscription confirmed!

You have successfully subscribed.

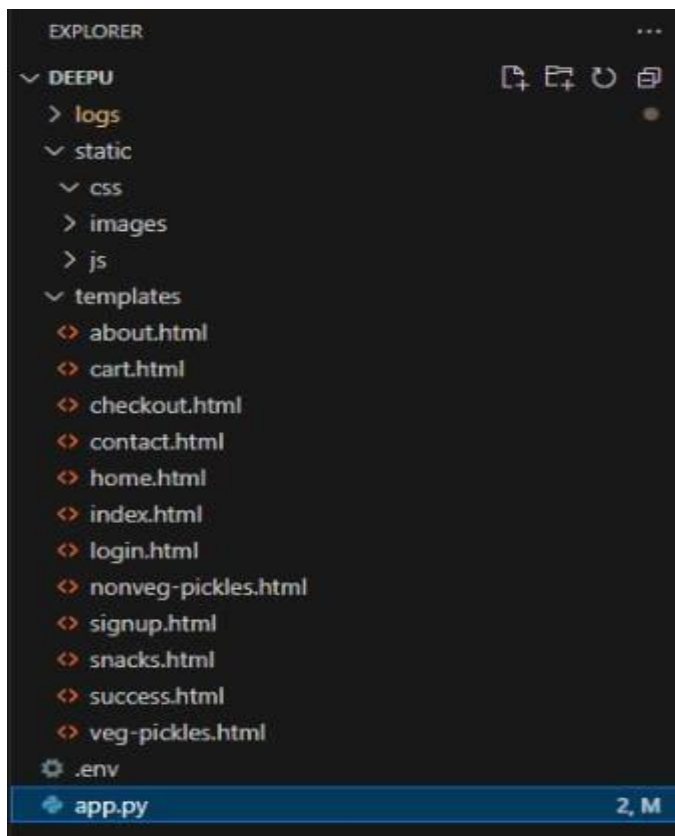
Your subscription's id is:
arn:aws:sns:us-east-1:216989138822:homemadepickles:e89352eb-94a7-4d73-8290-c0acafe98d1d

If it was not your intention to subscribe, [click here to unsubscribe](#).

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

Milestone 4: Backend Development and Application Setup

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



Description:

Backend Development and Application Setup focuses on establishing the core structure of the application. This includes configuring the backend framework, setting up routing, and integrating database connectivity. It lays the groundwork for handling user interactions, data management, and secure access.

Description of the code :

- Flask App Initialization

```
from flask import Flask, render_template, request, redirect, url_for
import boto3
from boto3.dynamodb.conditions import Key
import smtplib
from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart
from bcrypt import hashpw, gensalt, checkpw
```

Description: import essential libraries including Flask utilities for routing, Boto3 for DynamoDB operations, SMTP and email modules for sending mails, and Bcrypt for password hashing and verification

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

Description: initialize the Flask application instance using Flask(__name__) to start building the web app.

- Dynamodb Setup:

```
#Initialize DynamoDB resource
dynamodb = boto3.resource('dynamodb', region_name='us-east-1')

#DynamoDB Tables
orders_table = dynamodb.Table('PickleOrders')
users_table = dynamodb.Table('users')
```

Description: initialize the DynamoDB resource for the ap-south-1 region and set up access to the Users and Requests tables for storing user details and book requests.

- **SNS Connection**

```
#SNS Topic ARN
sns = boto3.client('sns', region_name='us-east-1')
SNS_TOPIC_ARN = 'arn:aws:sns:us-east-1:216989138822:homemadepickles'

# Email settings
EMAIL_HOST = os.getenv('EMAIL_HOST', 'smtp.gmail.com')
EMAIL_PORT = int(os.getenv('EMAIL_PORT', 587))
EMAIL_USER = os.getenv('EMAIL_USER')
EMAIL_PASSWORD = os.getenv('EMAIL_PASSWORD')
```

Description: Configure **SNS** to send notifications when a book request is submitted. Paste your stored ARN link in the sns_topic_arn space, along with the region_name where the SNS topic is created. Also, specify the chosen email service in SMTP_SERVER (e.g., Gmail, Yahoo, etc.) and enter the subscribed email in the SENDER_EMAIL section. Create an 'App password' for the email ID and store it in the SENDER_PASSWORD section.

- **Routes for Web Pages**

- **Register Route:**

```
@app.route("/signup.html", methods=["GET", "POST"])
def signup():
    if request.method == 'POST':
        name = request.form.get('name')
        email = request.form.get('email')
        password = request.form.get('password')
        confirm = request.form.get('confirm')

        if password != confirm:
            return render_template('signup.html', error="Passwords do not match!")

        # Save user to dictionary (in real app, use database)
        users[email] = {
            'name': name,
            'password': password # For security, use hashing (next step)
        }

        flash("Signup successful. Please log in.", "success")
        return redirect(url_for('login'))

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

Description: define /register route to validate registration form fields, hash the user password using Bcrypt, store the new user in DynamoDB with a login count, and send an SNS notification on successful registration

- **login Route (GET/POST):**

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

        user = users.get(email)
        if user and user['password'] == password:
            session['user'] = email
            return redirect(url_for('home'))
        else:
            return render_template('login.html', error="Invalid email or password")

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

-

Description: define /login route to validate user credentials against DynamoDB, check the password using Bcrypt, update the login count on successful authentication, and redirect users to the home page

- **Home routes:**

```
# ----- Routes -----

@app.route('/')
def home():
    user = session.get('user') # ✅ FIXED: get user safely
    if user:
        return render_template('home.html', user=user)
    return redirect(url_for('login'))

@app.route('/index.html')
def index():
    return render_template("index.html")
```

Description:

define the home route / to automatically redirect users to the register page when they access the base URL.

.

- **Request Routes:**

```
# Email settings
EMAIL_HOST = os.getenv('EMAIL_HOST', 'smtp.gmail.com')
EMAIL_PORT = int(os.getenv('EMAIL_PORT', 587))
EMAIL_USER = os.getenv('EMAIL_USER')
EMAIL_PASSWORD = os.getenv('EMAIL_PASSWORD')

@app.context_processor
def inject_theme():
    return {"color": app.config["THEME_COLOR"], "year": datetime.now().year}

# ----- Product Inventory -----
products = {
    "mango": {"name": "Mango Pickle", "price": 200, "stock": 10, "image": "mango pickle.webp"},
    "tomato": {"name": "Tomato Pickle", "price": 150, "stock": 7, "image": "Tomato pickle.webp"},
    "lemon": {"name": "Lemon Pickle", "price": 180, "stock": 8, "image": "Lemon pickle.jpg"},
    "chicken": {"name": "Chicken Pickle", "price": 250, "stock": 9, "image": "chicken pickle.webp"},
    "fish": {"name": "Fish Pickle", "price": 250, "stock": 6, "image": "Fish pickle.webp"},
    "mutton": {"name": "Mutton Pickle", "price": 300, "stock": 7, "image": "Mutton pickle.webp"},
    "banana_chips": {"name": "Banana Chips", "price": 100, "stock": 8, "image": "Banana Chips.jpg"},
    "ama_papad": {"name": "Ama Papad", "price": 80, "stock": 8, "image": "Aam papad.jpg"},
    "chekka_pakodi": {"name": "Chekka Pakodi", "price": 110, "stock": 5, "image": "Chekka Pakodi.jpg"}
}

def get_products(prefix=None):
    if not prefix:
        return products
    return {k: v for k, v in products.items() if k.startswith(prefix)}
```

Description: define /request-form route to capture book request details from users, store the request in DynamoDB, send a thank-you email to the user, notify the admin, and confirm submission with a success message.

LogoutRoute:

```
@app.route('/logout')
def logout():
    session.pop('user', None)
    return redirect(url_for('login'))
```

Description: define /logout route to render the exit.html page when the user chooses to leave or close the application.

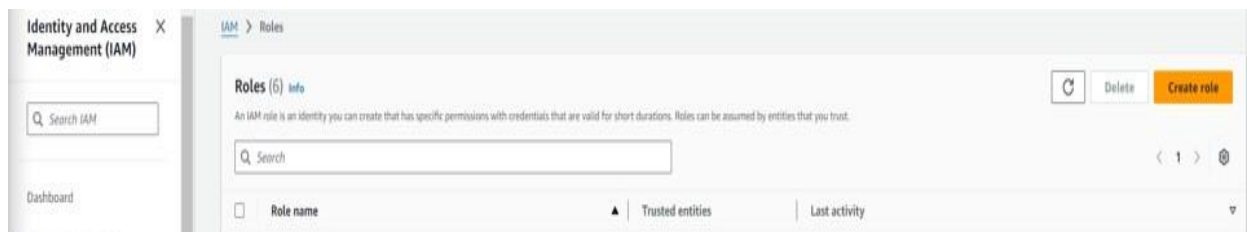
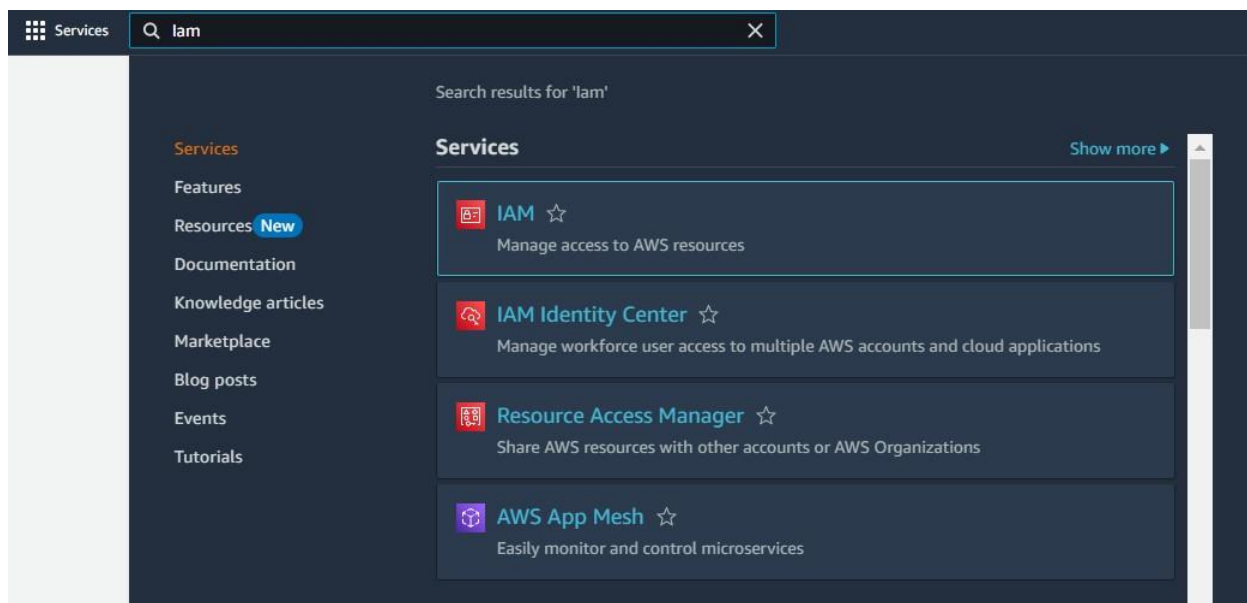
Deployment Code:

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

Description: start the Flask server to listen on all network interfaces (0.0.0.0) at port 80 with debug mode enabled for development and testing.

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.



IAM > Roles > Create role

Step 1
☒ **Select trusted entity**
 Step 2
☐ Add permissions
 Step 3
☐ Name, review, and create

Select trusted entity Info

Trusted entity type

☒ **AWS service**
 Allow AWS services like EC2, Lambda, or others to perform actions in this account.

☐ **AWS account**
 Allow entities in other AWS accounts belonging to you or a 3rd party to perform actions in this account.

☐ **Web identity**
 Allows users federated by the specified external web identity provider to assume this role to perform actions in this account.

☐ **SAML 2.0 federation**
 Allow users federated with SAML 2.0 from a corporate directory to perform actions in this account.

☐ **Custom trust policy**
 Create a custom trust policy to enable others to perform actions in this account.

Use case
 Allow an AWS service like EC2, Lambda, or others to perform actions in this account.

IAM > Roles > Create role

Step 1
☐ Select trusted entity
 Step 2
☐ Add permissions
 Step 3
☒ **Name, review, and create**

Name, review, and create

Role details

Role name
 Enter a meaningful name to identify this role.

 Maximum 64 characters. Use alphanumeric and '+', '@', '-' characters.

Description
 Add a short explanation for this role.

 Maximum 1000 characters. Use letters (A-Z and a-z), numbers (0-9), tabs, new lines, or any of the following characters: _ + = , @ - / [] ! \$ % ^ * () ; ' " ` ~ . , < > .

Step 1: Select trusted entities Edit

Trust policy
 1 of 1

● 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.

IAM > Roles > Create role

14
15
16

Step 2: Add permissions

Edit

Permissions policy summary

Policy name	Type	Attached as
AmazonDynamoDBFullAccess	AWS managed	Permissions policy
AmazonEC2FullAccess	AWS managed	Permissions policy
IAMFullAccess	AWS managed	Permissions policy

IAM > Roles > Create role

Step 1
● Select trusted entity
Step 2
● Add permissions
Step 3
● **Name, review, and create**

Name, review, and create

Role details

Role name
Enter a meaningful name to identify this role.

Maximum 64 characters. Use alphanumeric and '+', '@', '-' characters.

Description
Add a short explanation for this role.

Maximum 1000 characters. Use letters (A-Z and a-z), numbers (0-9), tabs, new lines, or any of the following characters: '_', '+', '@', '-', '/', '[', ']', '#', '\$', '%', '^', '*', '~', '!', '&', '(', ')', '=', '<', '>', '“', '”', '‑', '–', '—', '‘', '’'.

Step 1: Select trusted entities

Edit

Trust policy




1

IAM > Roles > sns_Dynamodb_role

sns_Dynamodb_role [Info](#) Delete


Allows EC2 instances to call AWS services on your behalf.

Summary Edit

Creation date October 13, 2024, 23:06 (UTC+05:30)	ARN  amawsiam::557690616836:role/sns_Dynamodb_role	Instance profile ARN  amawsiam::557690616836:instance-profile/sns_Dynamodb_role
Last activity  6 days ago	Maximum session duration 1 hour	





Permissions | Trust relationships | Tags | Last Accessed | Revoke sessions

Permissions policies (2) [Info](#)

🔄 Simulate  Remove Add permissions ▼





You can attach up to 10 managed policies.

Filter by Type
 All types ▼
< 1 > ⚙️

<input type="checkbox"/>	Policy name 🔗	Type	Attached entities
<input type="checkbox"/>	  AmazonDynamoDBFullAccess	AWS managed	4
<input type="checkbox"/>	  AmazonSNSFullAccess	AWS managed	2

Milestone 6: EC2 Instance Setup

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


 static/images	init
 templates	Updated log deletion code
 .env	Updated log deletion code
 app.py	Updated log deletion code

Local

Codespaces

 Clone 

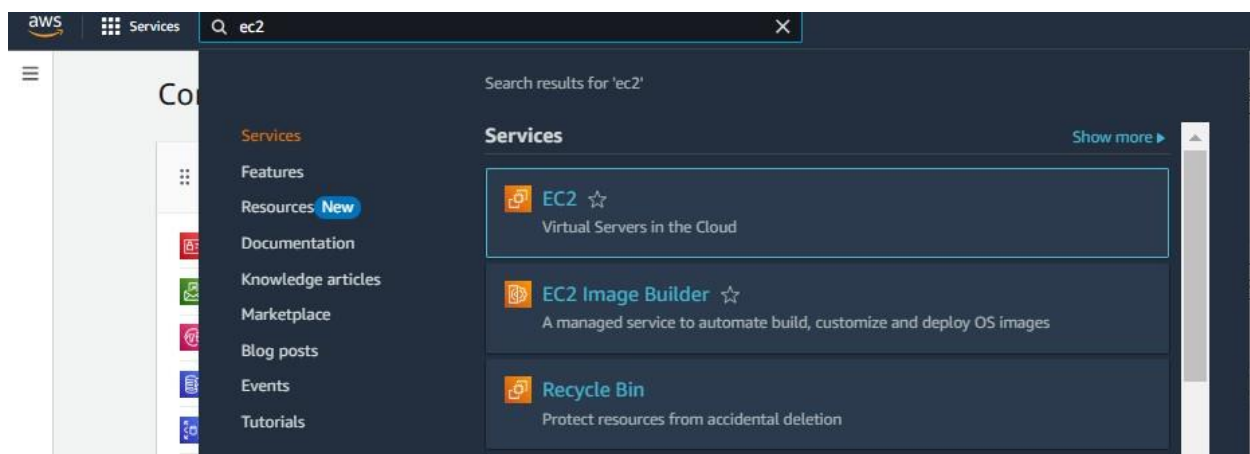
HTTPS SSH GitHub CLI

`https://github.com/deepika846/Homemade-snacks-recipe` 

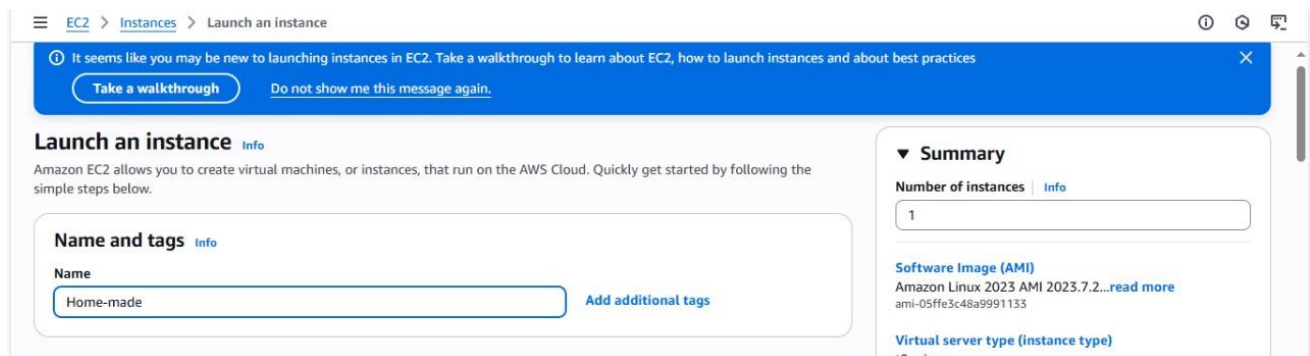
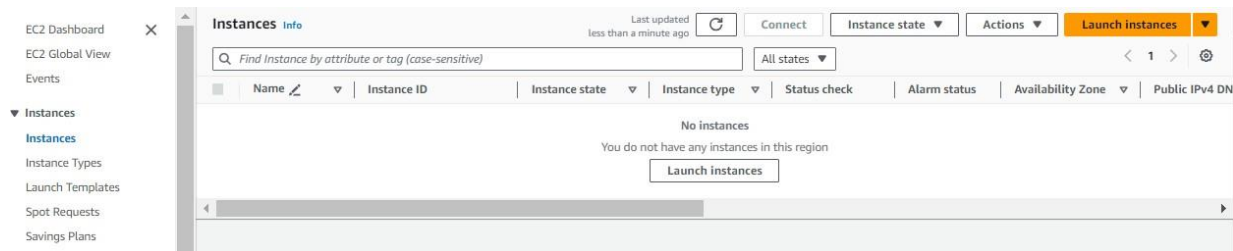
Clone using the web URL.

 Open with GitHub Desktop Download ZIP

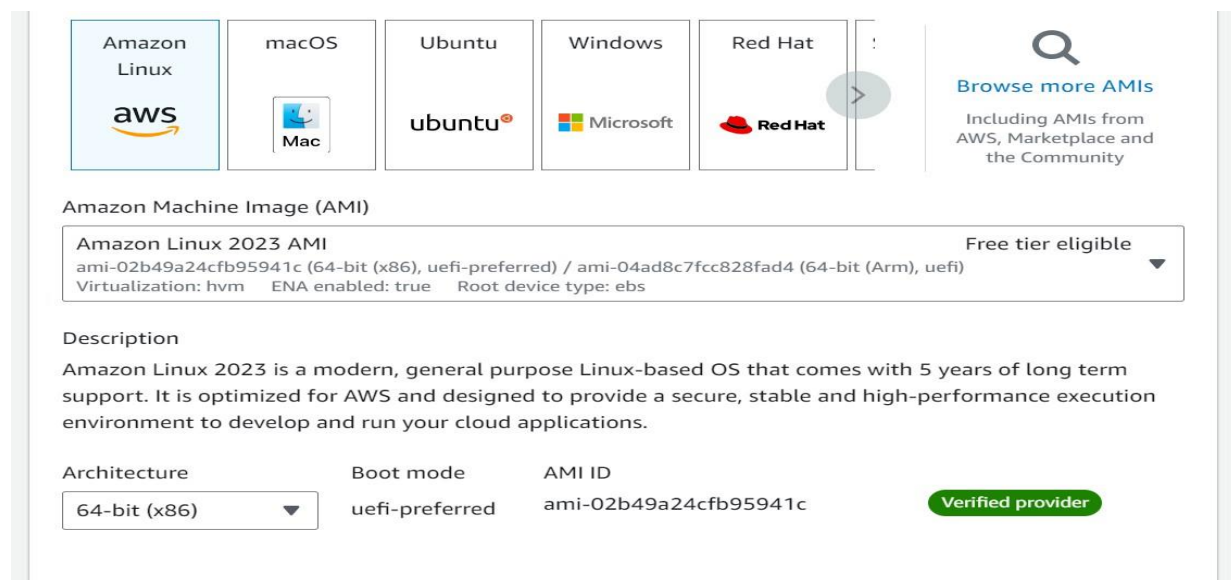
- **Activity 6.1: Launch an EC2 instance to host the Flask application.**
 - **Launch EC2 Instance**
 - In the AWS Console, navigate to EC2 and launch a new instance.



- Click on Launch instance to launch EC2 instance



- Choose Amazon Linux 2 or Ubuntu as the AMI and t2.micro as the instance type (free-tier eligible).



- Create and download the key pair for Server access.

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

Instance type

t2.micro

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

Free tier eligible

☐ All generations

[Compare instance types](#)

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](#)

EC2 > [Instances](#) > Launch an instance

launch the instance.

Key pair name - *required*

Select

▼ Network settings [Info](#)

VPC - *required* [Info](#)

vpc-0c5d5eab8432c1904

172.31.0.0/16

Subnet [Info](#)

No preference

Availability Zone [Info](#)

No preference

Auto-assign public IP [Info](#)

Enable

Firewall (security groups) [Info](#)

A security group is a set of firewall rules that control the traffic to and from your instances.

Create security group

Create key pair

Key pair name

Key pairs allow you to connect to your instance securely.

taste_the_best

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

Summary

[View all instances](#) [Info](#)

Image (AMI)

Linux 2023 AMI 2023.7.2...[read more](#)

3c48a9991133

server type (instance type)

o

(security group)

curity group

(volumes)

ne(s) - 8 GiB

cel

[Launch instance](#)

[Preview code](#)



InstantLibrary.pem

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

64-bit (x86)

Boot mode

uefi-preferred

AMI ID

ami-078264b8ba71bc45e

Username

ec2-user

Verified provider

▼ Instance type

Info | Get advice

Instance type

t2.micro

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

Free tier eligible

All generations

Compare instance types

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

InstantLibrary

Create new key pair

▼ Summary

Number of instances

Info

1

Software Image (AMI)

Amazon Linux 2023 AMI 2023.5.2...read more

ami-078264b8ba71bc45e

Virtual server type (instance type)

t2.micro

Firewall (security group)

New security group

Storage (volumes)

1 volume(s) - 8 GiB

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 IOs, 1 GB of snapshots, and 100 GB of bandwidth to the internet.

Can cel

Preview code

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

Create new subnet

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.

☒ Create security group
☐ 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)
Remove

Type Info
ssh

Protocol Info
TCP

Port range Info
22

Source type Info
Anywhere

Source Info
Add CIDR, prefix list or security group
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)
Remove

Type Info
HTTP

Protocol Info
TCP

Port range Info
80

Source type Info
Custom

Source Info
Add CIDR, prefix list or security group
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)
Remove

Type Info
Custom TCP

Protocol Info
TCP

Port range Info
5000

Source type Info
Custom

Source Info
Add CIDR, prefix list or security group
0.0.0.0/0

Description - optional Info
e.g. SSH for admin desktop

Add security group rule

EC2 > ... > Launch an instance

Success
Successfully initiated launch of instance [i-001861027fbac290]

► Launch log

Next Steps
Q: What would you like to do next with this instance, for example "create alarm" or "create backup"

Create billing and free tier usage alerts

To manage costs and avoid surprise bills, set up email notifications for billing and free tier usage thresholds.

Create billing alerts

Connect to your instance

Once your instance is running, log into it from your local computer.

Connect to instance

Learn more

Connect an RDS database

Configure the connection between an EC2 instance and a database to allow traffic flow between them.

Connect an RDS database

Create a new RDS database

Learn more

Create EBS snapshot policy

Create a policy that automates the creation, retention, and deletion of EBS snapshots.

Create EBS snapshot policy

Manage detailed monitoring

Enable or disable detailed monitoring for the instance. If you enable detailed monitoring, the Amazon EC2 console displays monitoring graphs with a 1-minute period.

Manage detailed monitoring

Create Load Balancer

Create an application, network gateway or classic Elastic Load Balancer.

Create Load Balancer

Create AWS budget

AWS Budgets allows you to create budgets, forecast spend, and take action on your costs and usage from a single location.

Create AWS budget

Manage CloudWatch alarms

Create or update Amazon CloudWatch alarms for the instance.

Manage CloudWatch alarms

Disaster recovery for your instances

Recover the instances you just launched into a different Availability Zone or a different Region using AWS Elastic Disaster Recovery (DRS).

Disaster recovery for your instances

Monitor for suspicious runtime activities

Amazon GuardDuty enables you to continuously monitor for malicious runtime activity and unauthorized behavior, with near real-time visibility into on-host activities occurring across your Amazon EC2 workloads.

Monitor for suspicious runtime activities

Get instance screenshot

Capture a screenshot from the instance and view it as an image. This is useful for troubleshooting an unreachable instance.

Get instance screenshot

Get system log

View the instance's system log to troubleshoot issues.

Get system log

[View all instances](#)

- 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.

EC2 > Instances

EC2

Dashboard
EC2 Global View
Events

▼ **Instances**

Instances
Instance Types
Launch Templates
Spot Requests
Savings Plans
Reserved Instances
Dedicated Hosts
Capacity Reservations

▼ **Images**

AMIs
AMI Catalog

▼ **Elastic Block Store**

Instances (1/1) Info Last updated less than a minute ago

[Connect](#) [Instance state](#) [Actions](#) [Launch instances](#)

Find Instance by attribute or tag (case-sensitive) All states

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability
Home-made	i-038809ae92e30fa30	Running	t2.micro	2/2 checks passed	View alarms	us-east-1c

i-038809ae92e30fa30 (Home-made)

Details | Status and alarms | Monitoring | **Security** | Networking | Storage | Tags

▼ **Security details**

IAM Role	Owner ID	Launch time
-	216989138822	Tue Jul 08 2025 14:24:58 GMT+0530 (India Standard Time)

Security groups

EC2 > Instances > i-001861022fbcac290

Instance summary for i-001861022fbcac290 (InstantLibraryApp) [Info](#)

Updated less than a minute ago

Instance ID i-001861022fbcac290 IPv4 address -- Hostname type IP name: ip-172-31-3-5-ap-south-1.compute.internal Answer private resource DNS name IPv4 (A) -- Auto-assigned IP address -- IAM Role sns_Dynamodb_role IMDSv2 Required	Public IPv4 address -- Instance state Stopped Private IP DNS name (IPv4 only) ip-172-31-3-5-ap-south-1.compute.internal Instance type t2.micro VPC ID vpc-05cdc7b6f19dd7211 Subnet ID subnet-0d9fa3144480cc9a9 Instance ARN arn:aws:ec2:ap-south-1:557690616836:instance/i-001861022fbcac290	Private IPv4 addresses 172.31.3.5 Public IPv4 DNS -- Elastic IP addresses -- AWS Compute Optimizer finding Opt-in to AWS Compute Optimizer for recommendations. Learn more Auto Scaling Group name --	Actions Connect Instance state Manage instance state Instance settings Networking Security Image and templates Monitor and troubleshoot Change security groups Get Windows password Modify IAM role
---	--	---	--

EC2 > Instances > i-038809ae92e30fa30 > Modify IAM role

Modify IAM role [Info](#)

Attach an IAM role to your instance.

Instance ID
 i-038809ae92e30fa30 (Home-made)

IAM role
 Select an IAM role to attach to your instance or create a new role if you haven't created any. The role you select replaces any roles that are currently attached to your instance.

snacks [Create new IAM role](#)

[Cancel](#) [Update IAM role](#)

- 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

i-001861022fbcac290 (InstantLibraryApp)

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

13.200.229.59

- IPv6 address

Username

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

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

```
#  
~\##### Amazon Linux 2023  
~~~\#####  
~~~~\#####  
~~~~#/  
~~~~V-!> https://aws.amazon.com/linux/amazon-linux-2023  
~~~~  
~~~~+  
~~~~+  
~~~~+  
~~~~m/<
```

[ec2-user@ip-172-31-90-136 ~]\$

```

~#
~\#####
~~\#####\
~~\#####|
~~\#/
~~v~'~> https://aws.amazon.com/linux/amazon-linux-2023

~.._
~/m/'

[ec2-user@ip-172-31-90-136 ~]$ sudo yum update -y
sudo yum install python3 git
sudo pip3 install flask boto3
sudo yum install python3-pip -y
pip3 install Flask
pip3 install boto3
pip3 install python-dotenv
```

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.

Run: 'git clone <https://github.com/your-github-username/your-repository-name.git>'

Note: change your-github-username and your-repository-name with your credentials

here: `git clone https://github.com/Deepika-9755/Homemade-snacks-and-pickles.git`

- This will download your project to the EC2 instance.

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

```
cd Homemade-snacks-and-pickles
```

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
```

Verify the Flask app is running:

<http://your-ec2-public-ip>

- Run the Flask app on the EC2 instance

```
[ec2-user@ip-172-31-3-5 InstantLibrary]$ sudo flask run --host=0.0.0.0 --port=80
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:80
* Running on http://172.31.3.5:80
Press CTRL+C to quit
183.82.125.56 - - [22/Oct/2024 07:42:00] "GET / HTTP/1.1" 302 -
183.82.125.56 - - [22/Oct/2024 07:42:01] "GET /register HTTP/1.1" 200 -
183.82.125.56 - - [22/Oct/2024 07:42:01] "GET /static/images/library3.jpg HTTP/1.1" 200 -
183.82.125.56 - - [22/Oct/2024 07:42:01] "GET /favicon.ico HTTP/1.1" 404 -
183.82.125.56 - - [22/Oct/2024 07:42:16] "GET /login HTTP/1.1" 200 -
183.82.125.56 - - [22/Oct/2024 07:42:16] "GET /static/images/library3.jpg HTTP/1.1" 304 -
183.82.125.56 - - [22/Oct/2024 07:42:21] "POST /login HTTP/1.1" 200 -
183.82.125.56 - - [22/Oct/2024 07:42:24] "GET /login HTTP/1.1" 200 -
183.82.125.56 - - [22/Oct/2024 07:42:27] "POST /login HTTP/1.1" 302 -
183.82.125.56 - - [22/Oct/2024 07:42:28] "GET /home-page HTTP/1.1" 200 -
```

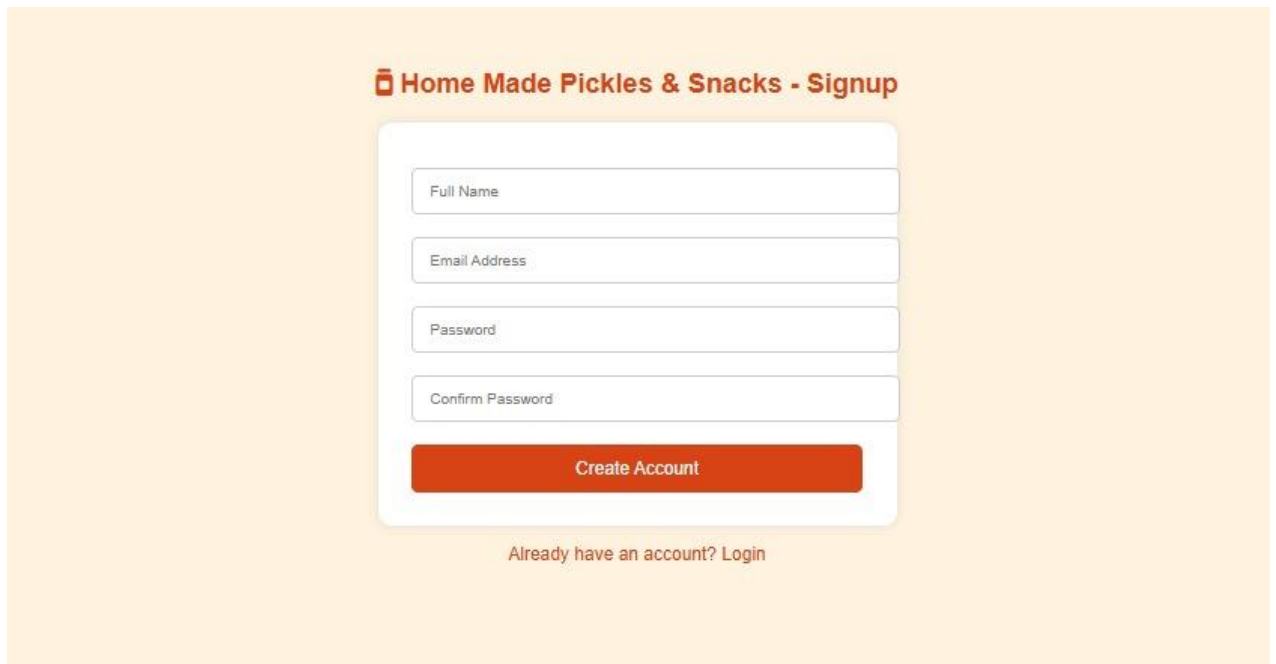
Access the website through:

PUBLIC IP : 184.72.205.71/5000

Milestone 8: Testing and Deployment

- **Activity 8.1: Conduct functional testing to verify user registration, login, order success ,order requests, and notifications.**

Sign Up page



Home Made Pickles & Snacks - Signup

Full Name

Email Address

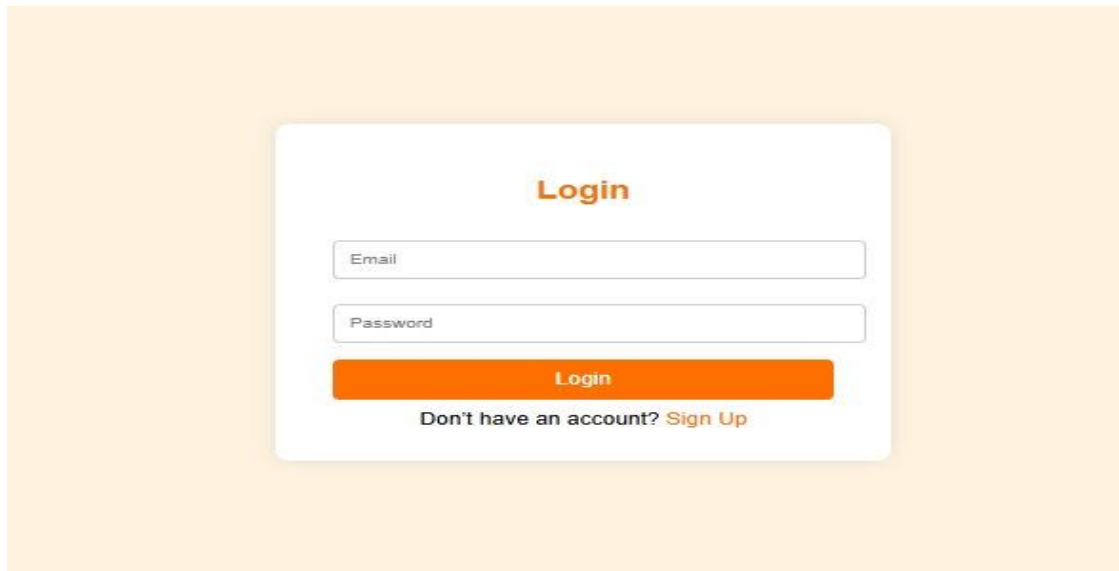
Password

Confirm Password

Create Account

Already have an account? [Login](#)

Login Page:



Login

Email

Password

Login

Don't have an account? [Sign Up](#)

Home Page

You are successfully logged in.

[Logout](#)

[Home](#) [Veg Pickles](#) [Non-Veg Pickles](#) [Snacks](#) [Cart](#) [Checkout](#) [Login](#) [Sign Up](#) [Contact Us](#) [About Us](#)

Home Made Pickles & Snacks

Taste the Best, Straight from Tradition

[Shop Now](#)



Veg Pickles



NonVeg Pickles



Snacks

© 2025 Pickle Paradise | For Pickle Lovers

HomeMade Pickles & Snacks

[Home](#) [Veg Pickles](#) [Non-Veg Pickles](#) [Snacks](#) [Logout](#) [Cart \(0\)](#)

Welcome to HomeMade Pickles & Snacks!

Your one-stop shop for delicious homemade pickles and snacks.

© 2025 Homemade Pickles & Snacks. All rights reserved.

Veg Pickles

Veg Pickles

**Mango Pickle**

Aged to perfectin with mustard seeds and aromatic spices.

250g-₹200

[Add to Cart](#)**Tomato Pickle**

Crunchy tomatoes marinated in mustard oil and spices.

250g-₹150

[Add to Cart](#)**Lemon Pickle**

Authentic Indian-style Lemon pickle with traditional spices.

250g-₹120

[Add to Cart](#)

Non Veg Pickles

Non-Veg Pickles

**Chicken Pickle**

Spicy Chicken Pickle: A Test of Home in Every Bit.

250g-₹300

[Add to Cart](#)**Fish Pickle**

Tangy and spicy, made with succulent fish pieces and traditional spices.

250g-₹250


[Add to Cart](#)**Mutton Pickle**

Rich mutton pickle cooked with tangy gongura leaves and savory spices.

250g-₹330

[Add to Cart](#)

Snacks Page


[Homemade Snacks](#)

[Home](#)
[Cart](#)



Banana Chips
Crispy and golden, our banana chips are made from fresh ripe bananas for a perfect crunch.
250g-₹100
[Add to Cart](#)




Aam Papad
Sweet & tangy mango leather, sun-dried to perfection. A summer treat with no preservatives.
250g-₹80
[Add to Cart](#)



Chekka Pakodi
Deep-fried spicy gram flour snack, delivering crunch and savory flavor in every bite.
250g-₹110
[Add to Cart](#)

Cart Page


[Your Cart](#)

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

[Clear Cart](#)
[Proceed to Checkout](#)

Check out Page

Checkout

Choose Payment Method:

☐

Cash on Delivery

☐

UPI

☐

Credit Card

[← Back to Cart](#)

Success page



Your order has been placed successfully!

[← Back to Home](#)

About us page

Homemade Pickles & Snacks

HomeVeg PicklesNon-Veg PicklesSnacksAboutContact

About Us

Bringing the taste of tradition to your table with homemade pickles and snacks crafted with love.

Our Story

Our journey began with a passion for authentic homemade flavors. Inspired by the recipes passed down through generations, we decided to bring back the nostalgic taste of grandma's kitchen. Every jar of pickle and every bite of snack is prepared with fresh, natural ingredients and no preservatives.

What We Offer

We specialize in both veg and non-veg pickles, as well as crispy, traditional Indian snacks like banana chips, murukku, and laddus. Our products are made in small batches to ensure the highest quality and taste.

Why Choose Us?

We believe in preserving tradition while delivering convenience. Whether you're craving a spicy mutton pickle or a sweet rava laddu, our products are made with care, hygiene, and heart.

Contact Us

Homemade Pickles & Snacks

HomeVeg PicklesNon-Veg PicklesSnacksAboutContact

Contact Us

We'd love to hear from you! Please fill out the form below and we'll get in touch soon.

Full Name

Your name

Email

you@example.com

Message

Type your message here...

Send Message

Conclusion

The Homemade Pickles and Snacks platform has been meticulously crafted to deliver a seamless and delightful experience for food enthusiasts seeking authentic, handcrafted flavors. By leveraging modern web technologies such as Flask for backend logic, secure user authentication, and dynamic cart management, the platform ensures a user-friendly interface for browsing, customizing, and ordering artisanal pickles and snacks.

The integration of cloud-ready architecture (e.g., AWS for future scalability) and robust session management allows the platform to handle high traffic efficiently while maintaining real-time updates for orders and inventory. Features like weight-based pricing, category-specific searches, and instant checkout streamline the shopping process, empowering customers to explore a diverse range of traditional and innovative recipes with ease.

This project addresses the growing demand for homemade, preservative-free food products by bridging the gap between small-scale producers and discerning customers. The platform's intuitive design and secure payment workflows enhance trust and convenience, while backend tools enable effortless inventory tracking and order fulfillment for administrators.

By combining time-honored recipes with modern e-commerce capabilities, this website not only preserves culinary heritage but also adapts to the digital age, ensuring that every jar of pickle or snack reaches customers with the same care and quality as a homemade meal. As the platform evolves, it stands ready to scale, introduce new product lines, and foster a community of food lovers united by a passion for authentic flavors.

In essence, this project redefines the way homemade delicacies are shared and enjoyed, offering a flavorful bridge between tradition and technology.