

## 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.

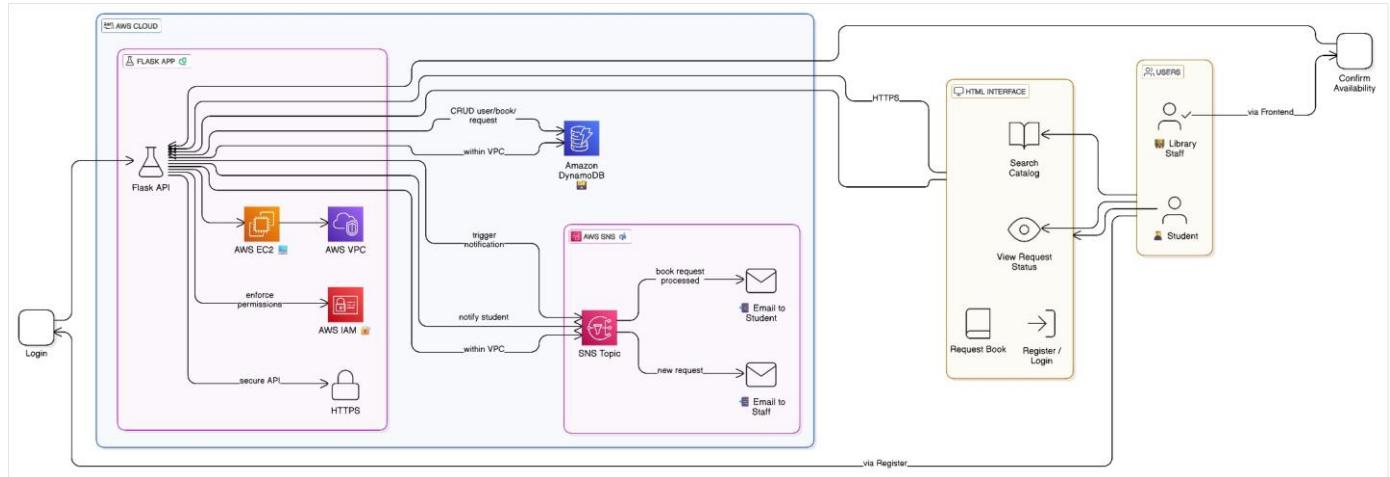
#### Scenario 2: Real-Time Inventory Tracking and Updates

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.

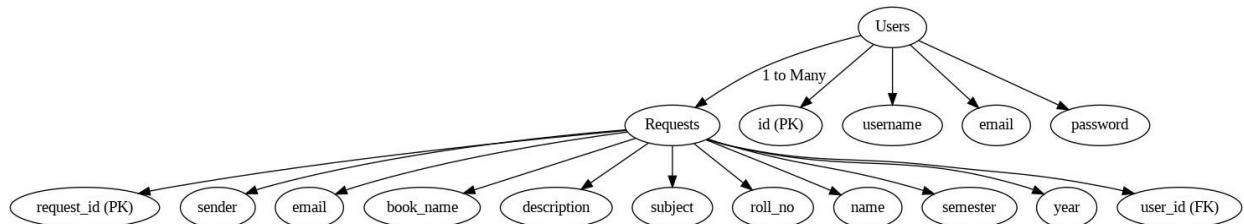
#### Scenario 3: Personalized User Experience and Recommendations

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

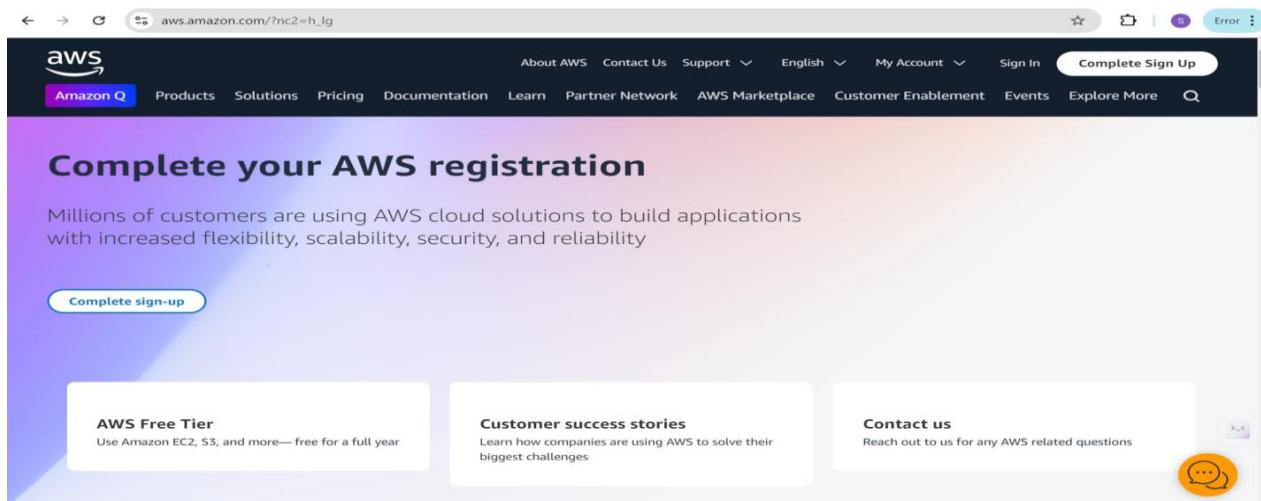
**Activity 7.2:** Run the Flask App

## 8. Testing and Deployment

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

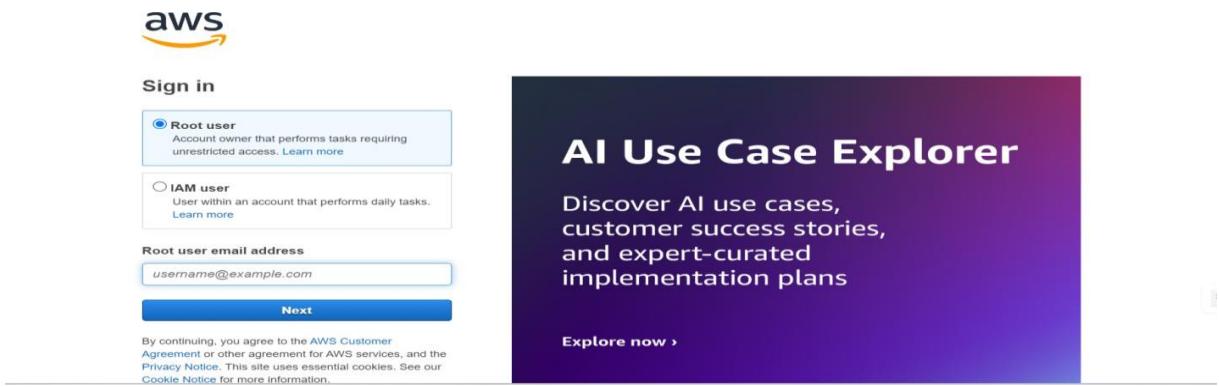
### Milestone 1: AWS Account Setup and Login

- **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**

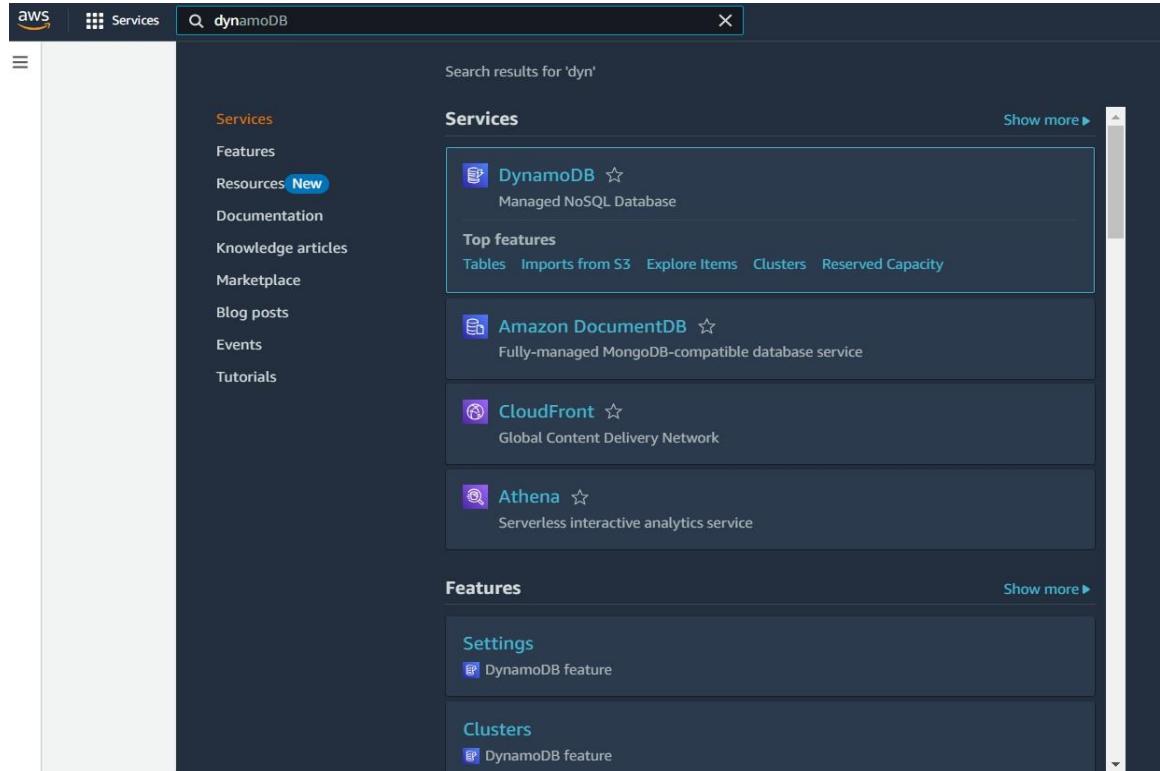
- After setting up your account, log in to the [AWS Management Console](#).



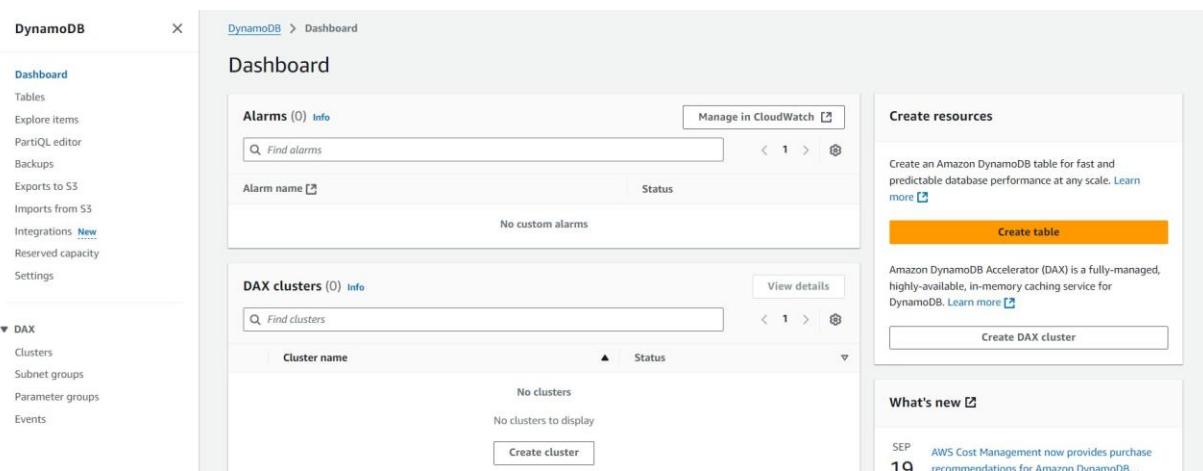
## Milestone 2: DynamoDB Database Creation and Setup

- **Activity 2.1: Navigate to the DynamoDB**

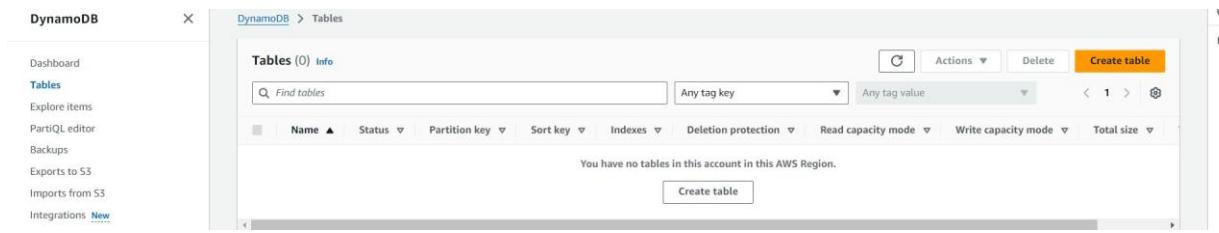
- In the AWS Console, navigate to DynamoDB and click on create tables.



The screenshot shows the AWS Services search results page. The search bar at the top contains 'dynamoDB'. The left sidebar has a 'Services' section with links to Features, Resources (New), Documentation, Knowledge articles, Marketplace, Blog posts, Events, and Tutorials. The main content area is titled 'Services' and lists several services: **DynamoDB** (Managed NoSQL Database), **Amazon DocumentDB** (Fully-managed MongoDB-compatible database service), **CloudFront** (Global Content Delivery Network), and **Athena** (Serverless interactive analytics service). Below this, there are sections for 'Features' with 'Settings' and 'Clusters'.

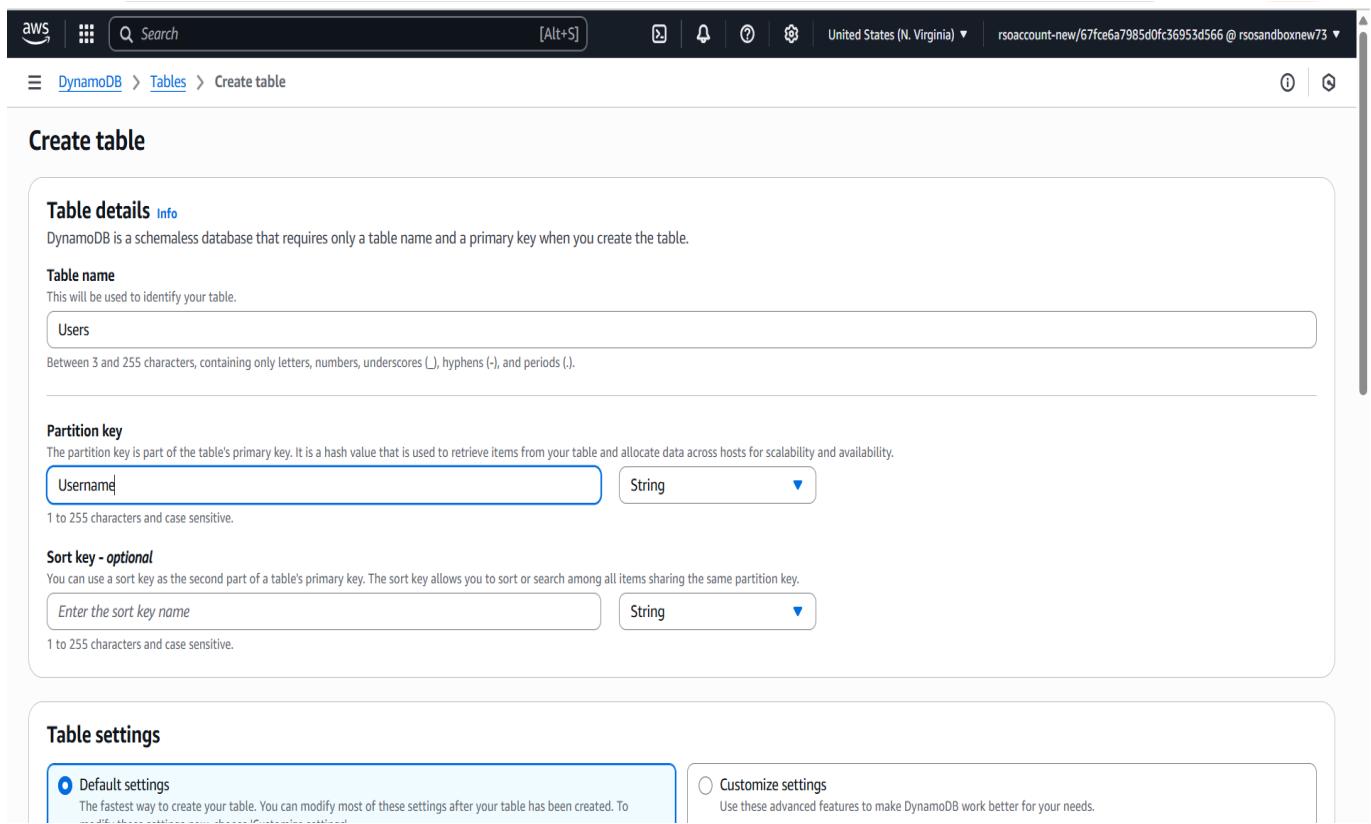


The screenshot shows the DynamoDB Dashboard. The left sidebar has a 'Dashboard' section with links to Tables, Explore items, PartiQL editor, Backups, Exports to S3, Imports from S3, Integrations (New), Reserved capacity, and Settings. It also has a 'DAX' section with links to Clusters, Subnet groups, Parameter groups, and Events. The main dashboard area has two main sections: 'Alarms (0)' and 'DAX clusters (0)'. Both sections have search bars and status indicators. To the right, there is a 'Create resources' section with a 'Create table' button and information about DAX. At the bottom, there is a 'What's new' section with a note about AWS Cost Management.



## ● Activity 2.2: Create a DynamoDB table

- Create Users table with partition key “Username” with type String and click on create 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.

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.

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...

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

### 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 Users table was created successfully.

### Tables (1) Info

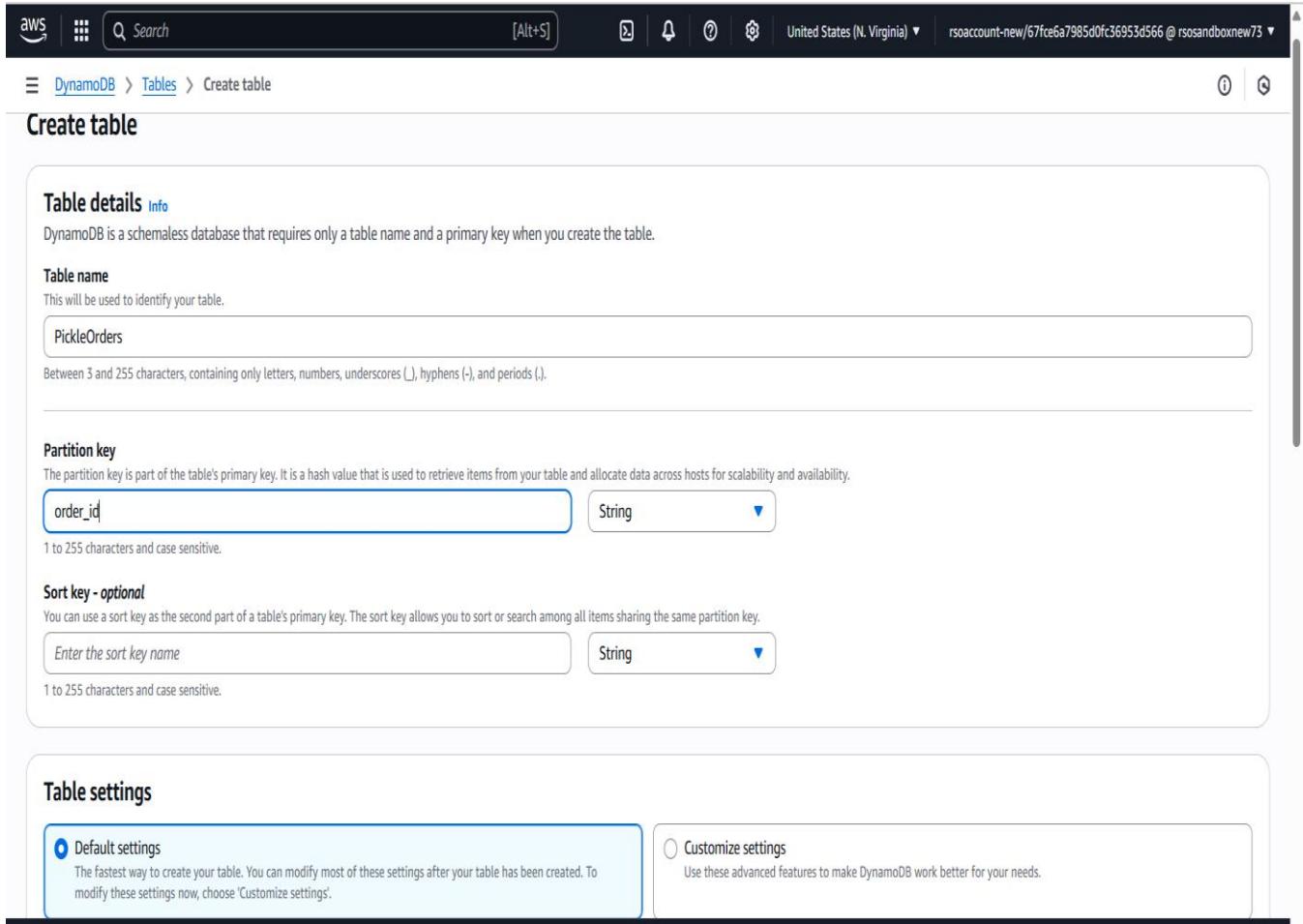

[Actions](#) ▾

[Delete](#)
[Create table](#)
Q Find tables Any tag key ▾ Any tag value ▾

☐ Name ▲ Status Partition key Sort key Indexes Replication Regions Deletion protection Favorite ▾ Read capacity mode W

☐ [Users](#) Active Username (\$) - 0 0 Off On-demand 0

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



The screenshot shows the AWS DynamoDB 'Create table' interface. At the top, there's a navigation bar with the AWS logo, a search bar, and account information ('United States (N. Virginia) rsoaccount-new/67fce6a7985d0fc36953d566 @ rsosandboxnew73'). Below the navigation is a breadcrumb trail: 'DynamoDB > Tables > Create table'. The main section is titled '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.

<input type="text" value="order_id"/>	Type: <b>String</b>
---------------------------------------	---------------------

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.

<input type="text" value="Enter the sort key name"/>	Type: <b>String</b>
--	---------------------

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

aws | Search [Alt+S] | United States (N. Virginia) rsoaccount-new/67fce6a7985d0fc36953d566 @ rsosandboxnew73

DynamoDB > Tables

The PickleOrders table was created successfully.

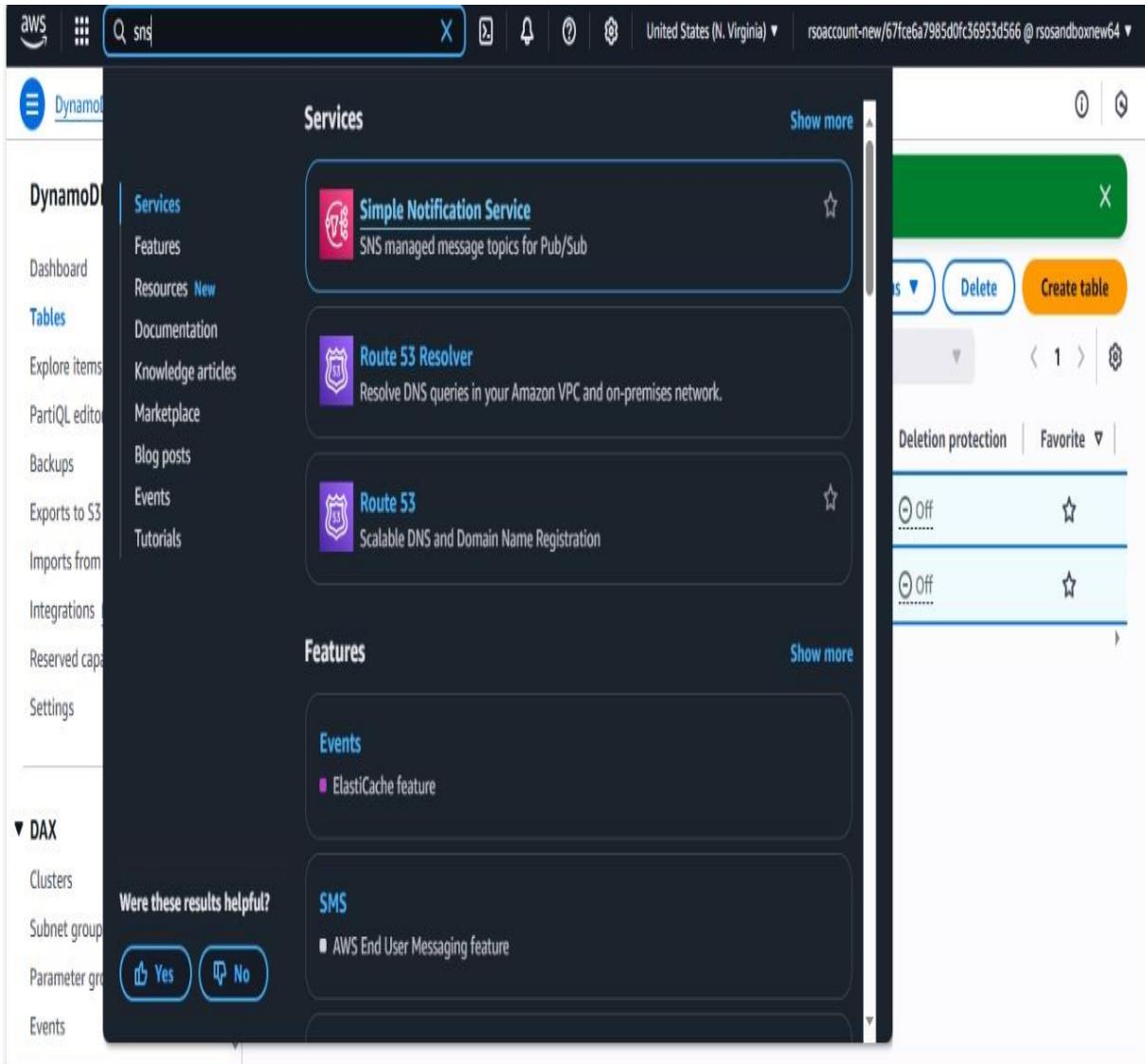
**Tables (2) Info**

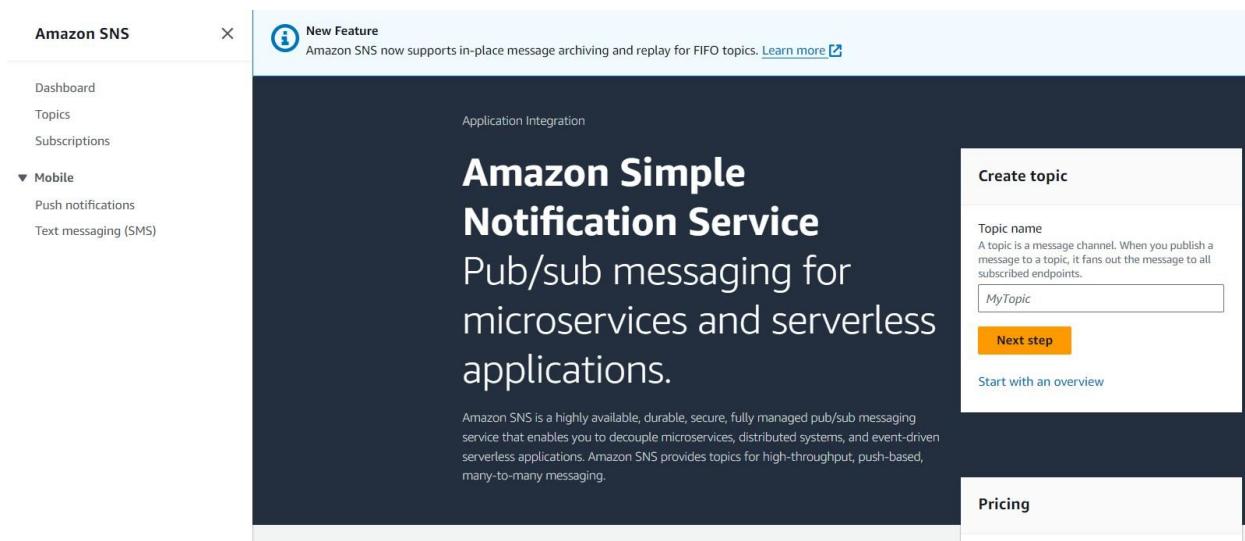
Name	Status	Partition key	Sort key	Indexes	Replication Regions	Deletion protection	Favorite	Read capacity mode
PickleOrders	Active	order_id (\$)	-	0	0	Off	☆	On-demand
Users	Active	Username (\$)	-	0	0	Off	☆	On-demand

## 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.





Amazon SNS

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

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.

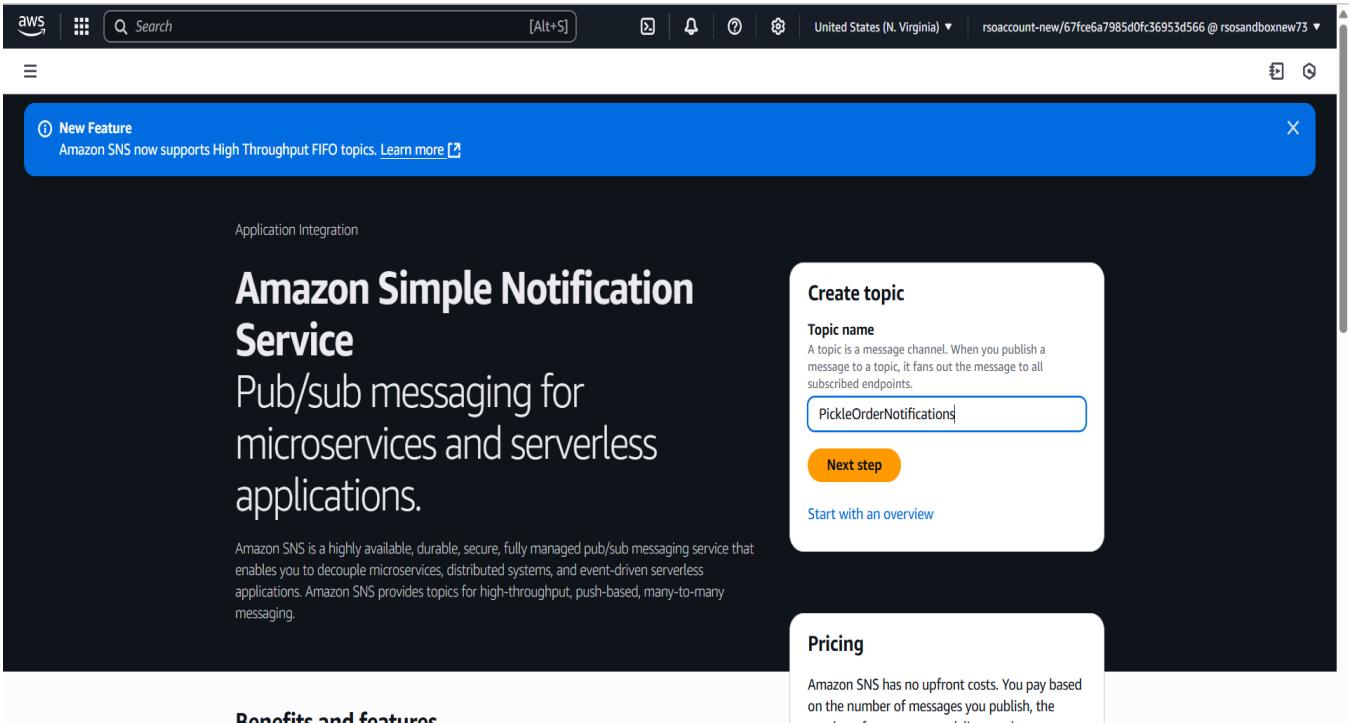
MyTopic

Next step

Start with an overview

Pricing

- Enter Topic Name and Click Next Step



aws | Search [Alt+S] | ⋮ | ? | United States (N. Virginia) ▾ | rsoaccount-new/67fce6a7985d0fc36953d566 @ rsosandboxnew73 ▾

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

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.

PickleOrderNotifications

Next step

Start with an overview

Pricing

Benefits and features

Handler: HelloWorldFunction.handler

Role: Lambda execution role - rsosandboxnew73

Code

File upload

Environment

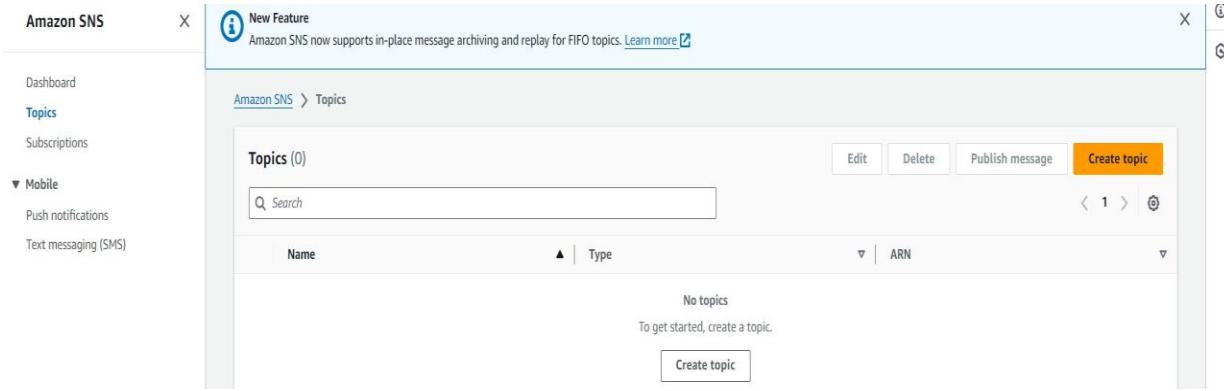
AWS\_LAMBDA\_FUNCTION\_NAME: HelloWorldFunction

AWS\_LAMBDA\_FUNCTION\_MEMORY\_SIZE: 128

Logs

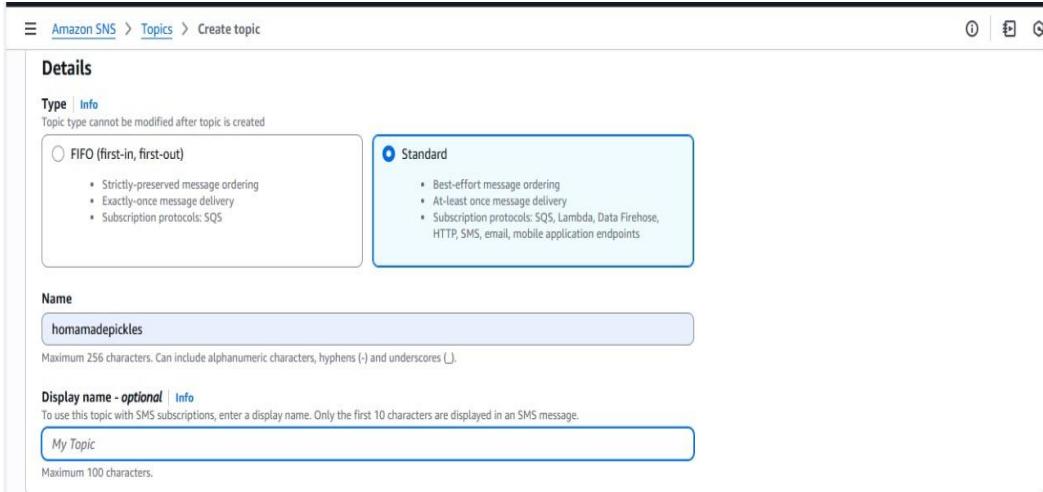
2018-07-10T10:45:00.000Z HelloWorldFunction[128]: Hello world!

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



The screenshot shows the Amazon SNS Topics page. On the left, there is a sidebar with links: Dashboard, Topics (which is selected and highlighted in blue), Subscriptions, Mobile (with Push notifications and Text messaging (SMS)), and a search bar. The main content area has a header 'Topics (0)' with buttons for Edit, Delete, Publish message, and Create topic (which is highlighted in orange). Below this is a search bar and a table with columns Name, Type, and ARN. A message at the bottom says 'No topics' and 'To get started, create a topic.' with a 'Create topic' button.

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



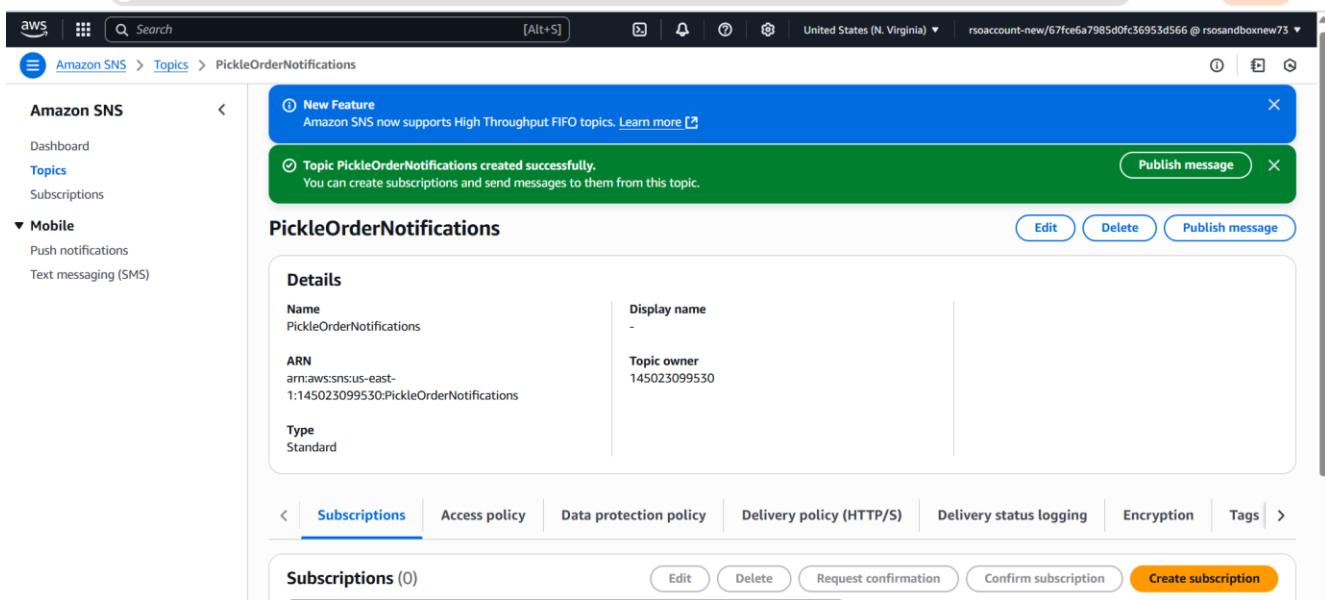
The screenshot shows the 'Create topic' wizard. At the top, it says 'Amazon SNS > Topics > Create topic'. Below that is a 'Details' section with tabs for Type (selected) and Info. It says 'Topic type cannot be modified after topic is created'. There are two options: 'FIFO (first-in, first-out)' and 'Standard'. 'Standard' is selected and highlighted in blue. Under 'Name', the value 'homamadepickles' is entered. Under 'Display name - optional', the value 'My Topic' is entered. Both fields have character limits: 256 and 100 respectively.

=

- ▶ **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 > Topics > PickleOrderNotifications

**PickleOrderNotifications**

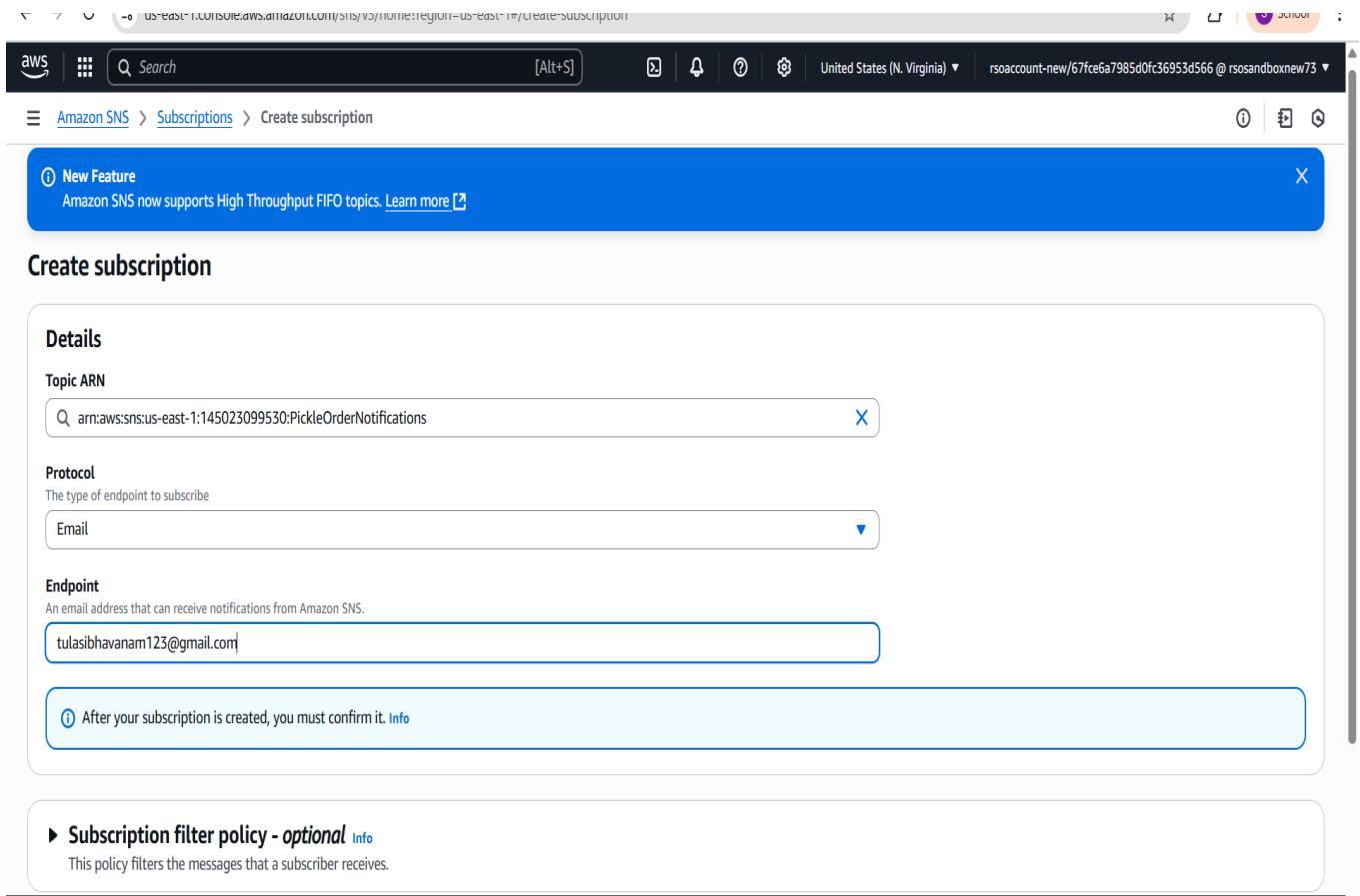
**Details**

Name	PickleOrderNotifications	Display name	
ARN	arnaws:sns:us-east-1:145023099530:PickleOrderNotifications	Topic owner	145023099530
Type	Standard		

**Subscriptions (0)** [Edit](#) [Delete](#) [Request confirmation](#) [Confirm subscription](#) [Create subscription](#)

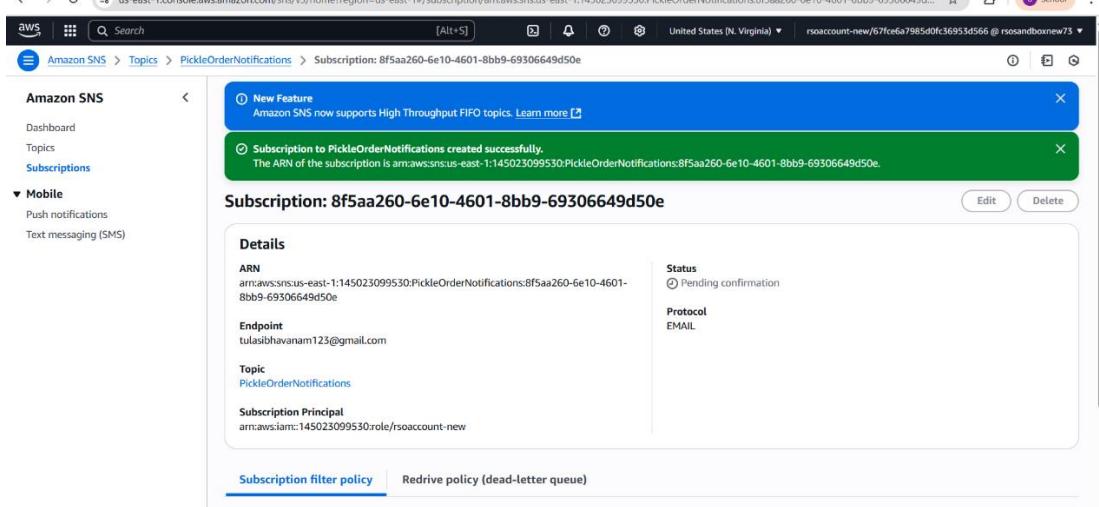
- **Activity 3.2: Subscribe users and staff to relevant SNS topics to receive real-time notifications when a order equest is made.**

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



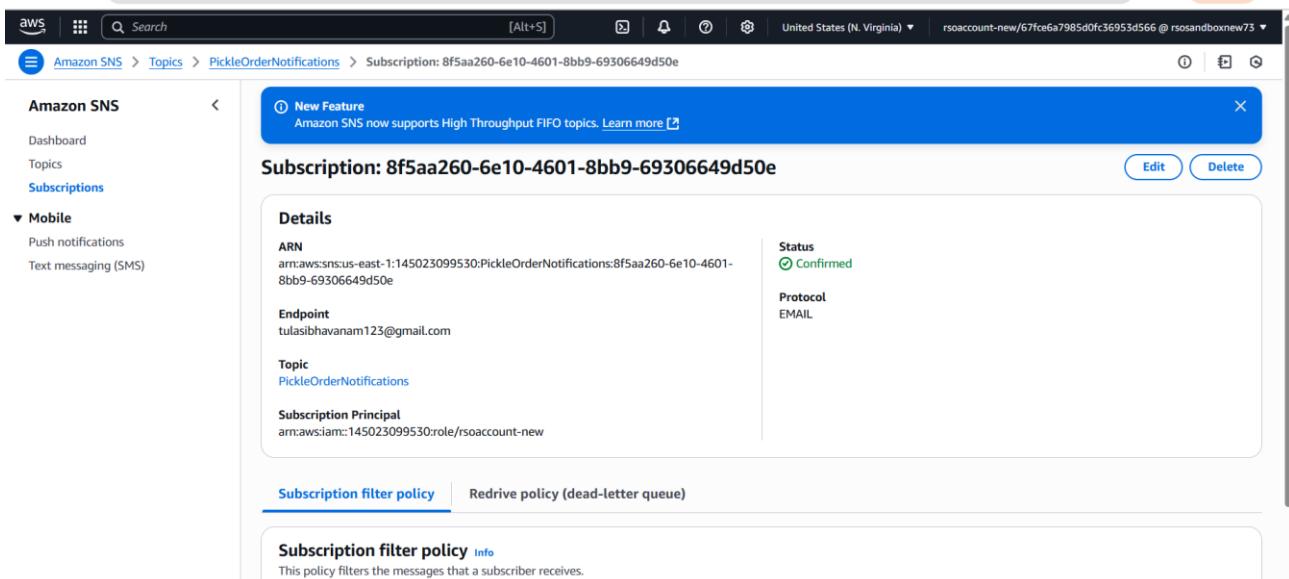
The screenshot shows the 'Create subscription' page in the AWS SNS console. The URL in the browser bar is `us-east-1.console.aws.amazon.com/sns/v3/home?region=us-east-1#/create-subscription`. The top navigation bar includes the AWS logo, search bar, and various icons. The main content area has a blue header bar with the text 'New Feature' and 'Amazon SNS now supports High Throughput FIFO topics. Learn more'. Below this, the title 'Create subscription' is displayed. The form fields are as follows:

- Topic ARN:** `arn:aws:sns:us-east-1:145023099530:PickleOrderNotifications`
- Protocol:** `Email`
- Endpoint:** `tulasibhavanam123@gmail.com`
- Note:** A message states 'After your subscription is created, you must confirm it.' with a link to 'Info'.
- Subscription filter policy - optional:** A note says 'This policy filters the messages that a subscriber receives.'



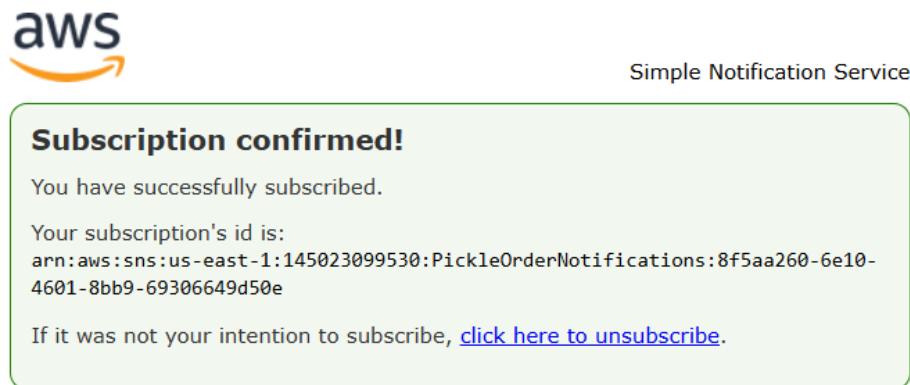
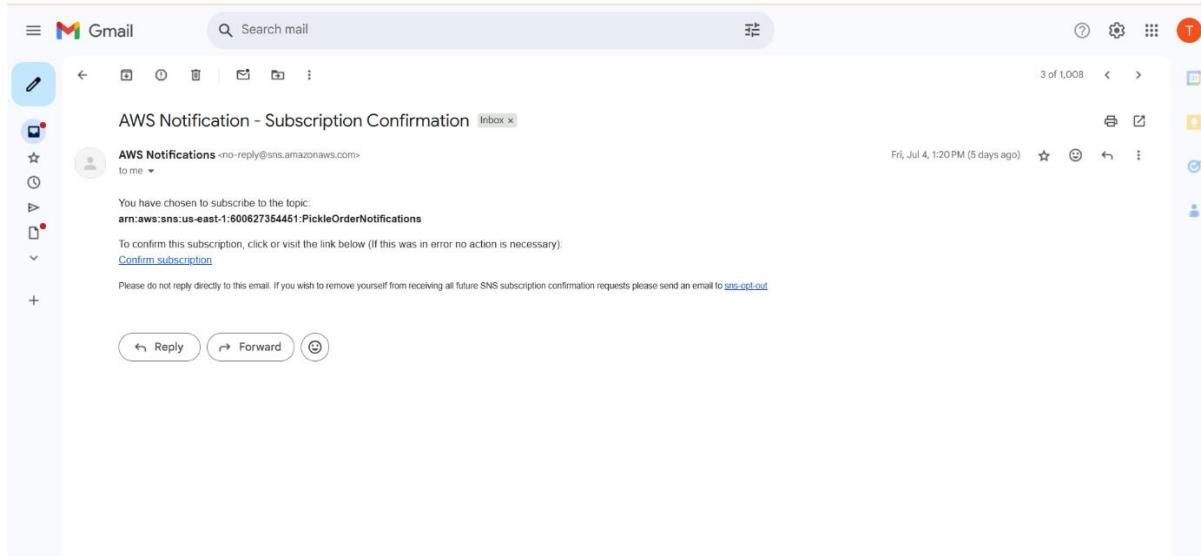
The screenshot shows the AWS SNS console. In the left sidebar, under 'Amazon SNS', 'Subscriptions' is selected. The main content area displays a success message: 'Subscription to PickleOrderNotifications created successfully.' Below this, the subscription details are shown for 'Subscription: 8f5aa260-6e10-4601-8bb9-69306649d50e'. The 'Details' section includes fields for ARN, Endpoint, Topic, and Subscription Principal. The status is listed as 'Pending confirmation'. At the bottom, there are tabs for 'Subscription filter policy' and 'Redrive policy (dead-letter queue)'. Buttons for 'Edit' and 'Delete' are also present.

- After subscription request for the mail confirmation



This screenshot shows the same AWS SNS subscription details as the previous one, but with a different status. The 'Status' field now shows 'Confirmed' with a green checkmark. All other details (ARN, Endpoint, Topic, Subscription Principal) remain the same. The 'Details' section and the tabs at the bottom are identical to the first screenshot.

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

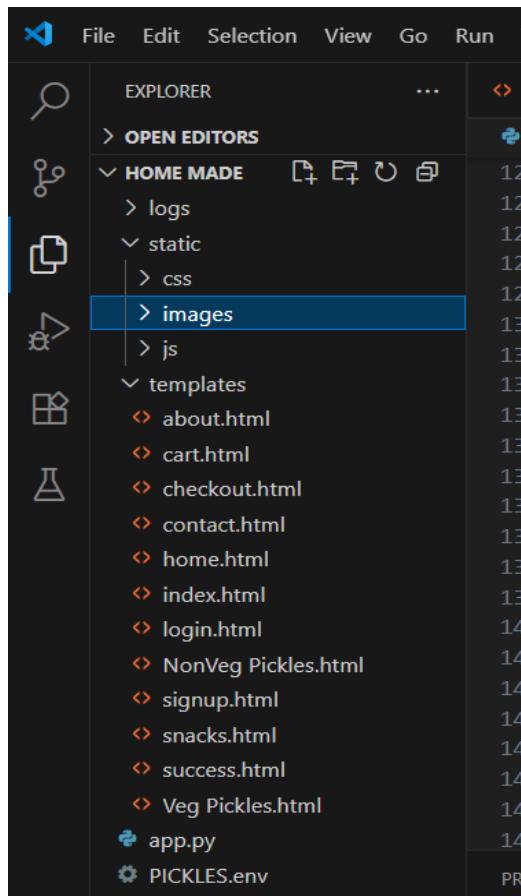


- Successfully done with the SNS mail subscription and setup, now store the ARN link.

## 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:

```
AWS_REGION = 'us-east-1'
SNS_TOPIC_ARN = 'arn:aws:sns:us-east-1:600627354451:PickleOrderNotifications'

dynamodb = boto3.resource('dynamodb', region_name=AWS_REGION)
orders_table = dynamodb.Table('PickleOrders')
users_table = dynamodb.Table('users')
sns = boto3.client('sns', region_name=AWS_REGION)
```

**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**

```
# 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:** 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:

```
@app.route('/checkout.html', methods=['GET', 'POST'])
def checkout():
    if 'user' not in session:
        flash('Please login to checkout', 'warning')
        return redirect(url_for('login'))

    if request.method == 'POST':
        name = request.form.get('name')
        email = request.form.get('email')
        phone = request.form.get('phone')
        address = request.form.get('address')
        notes = request.form.get('notes')
        payment = request.form.get('payment')

        if not name or not email or not phone or not address or not payment:
            flash('All fields including payment method are required!', 'danger')
            return redirect(url_for('checkout'))

        order_id = str(uuid.uuid4())[:8].upper()
        session['cart'] = []

        flash(f'Order #{order_id} placed successfully!', 'success')
        return redirect(url_for('success'))

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

**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:

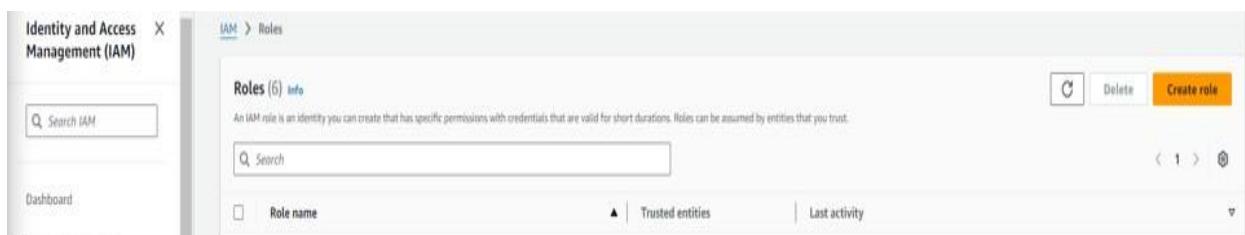
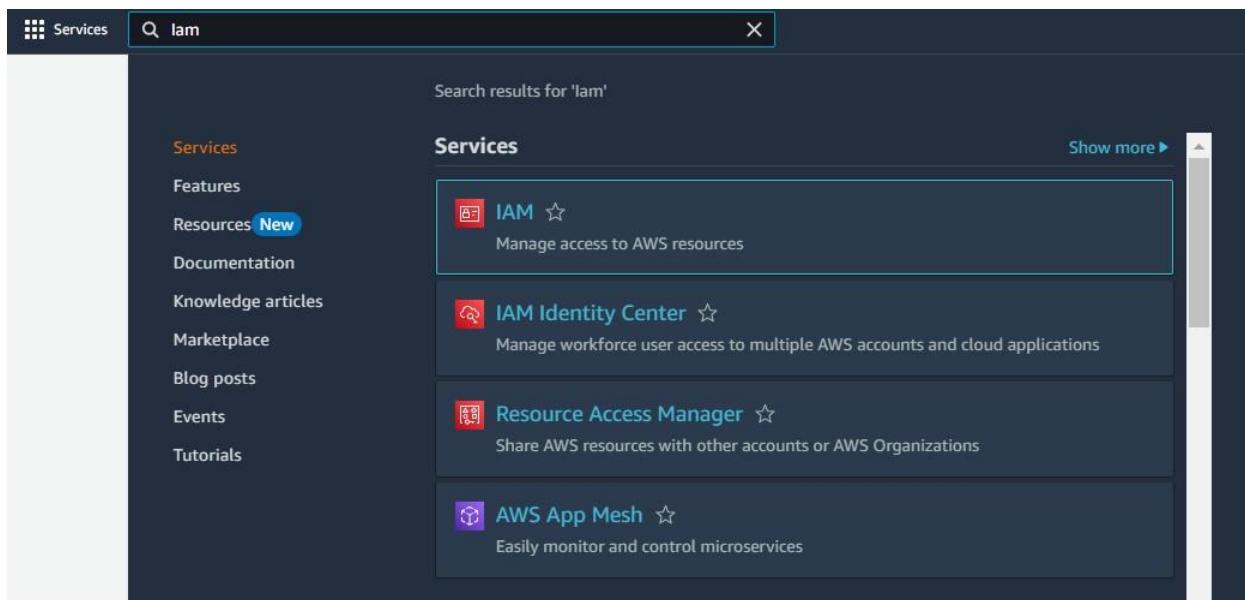
```
if __name__ == "__main__":
    app.run(host='0.0.0.0', port=5000, debug=True)
```

**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.

aws Search [Alt+S] Global rsoaccount-new/67fce6a7985d0fc3c6953d566 @ rsosandboxnew73

☰ IAM > Roles > Create role

Step 3  
Name, review, and create

**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: \_+=,. @-/[]{}#\$%^&`~-`

- **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.



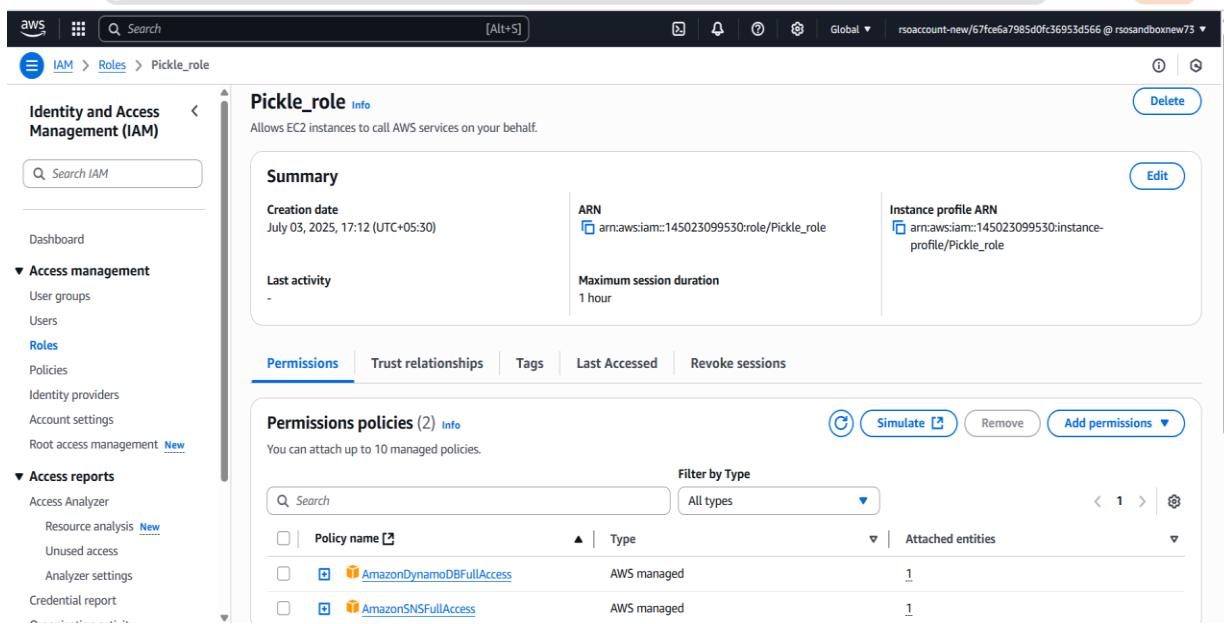
The screenshot shows the 'Add permissions' step of the 'Create role' wizard. At the top, there is a code editor window displaying a JSON policy document:

```
11 12 13 14 15 16 } ] "ec2.amazonaws.com"
```

Below the code editor, the section title 'Step 2: Add permissions' is visible, along with an 'Edit' button. The main area is titled 'Permissions policy summary' and contains a table:

Policy name	Type	Attached as
<a href="#">AmazonDynamoDBFullAccess</a>	AWS managed	Permissions policy
<a href="#">AmazonSNSFullAccess</a>	AWS managed	Permissions policy

At the bottom of the page, the 'Step 3: Add tags' section is shown, which includes an 'Add new tag' button and a note about adding up to 50 more tags.



**Pickle\_role** Info

Allows EC2 instances to call AWS services on your behalf.

Summary	
Creation date	July 03, 2025, 17:12 (UTC+05:30)
Last activity	-
ARN	<a href="#">arn:aws:iam::145023099530:role/Pickle_role</a>
Maximum session duration	1 hour
Instance profile ARN	<a href="#">arn:aws:iam::145023099530:instance-profile/Pickle_role</a>

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

**Permissions policies (2) Info**

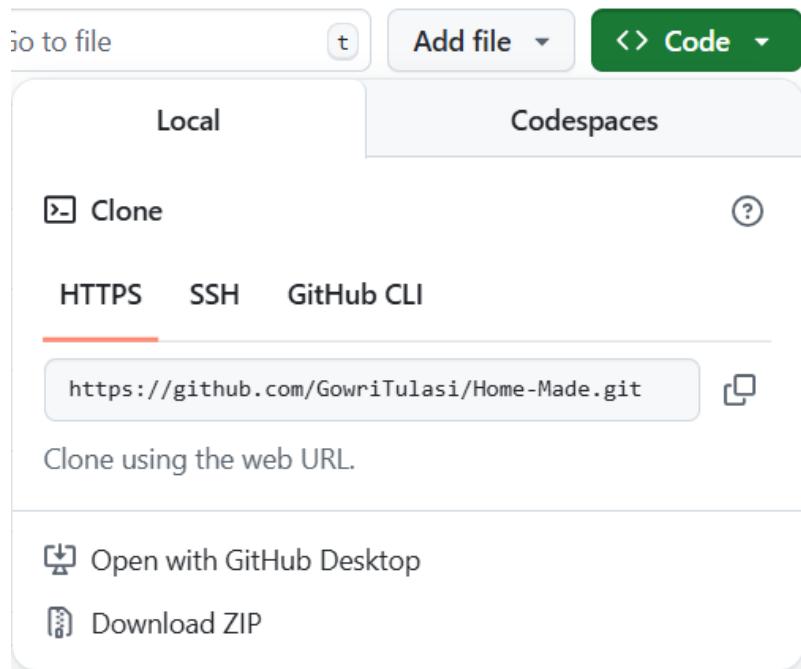
You can attach up to 10 managed policies.

Filter by Type	
Policy name	Type
<a href="#">AmazonDynamoDBFullAccess</a>	AWS managed
<a href="#">AmazonSNSFullAccess</a>	AWS managed

## Milestone 6: EC2 Instance Setup

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

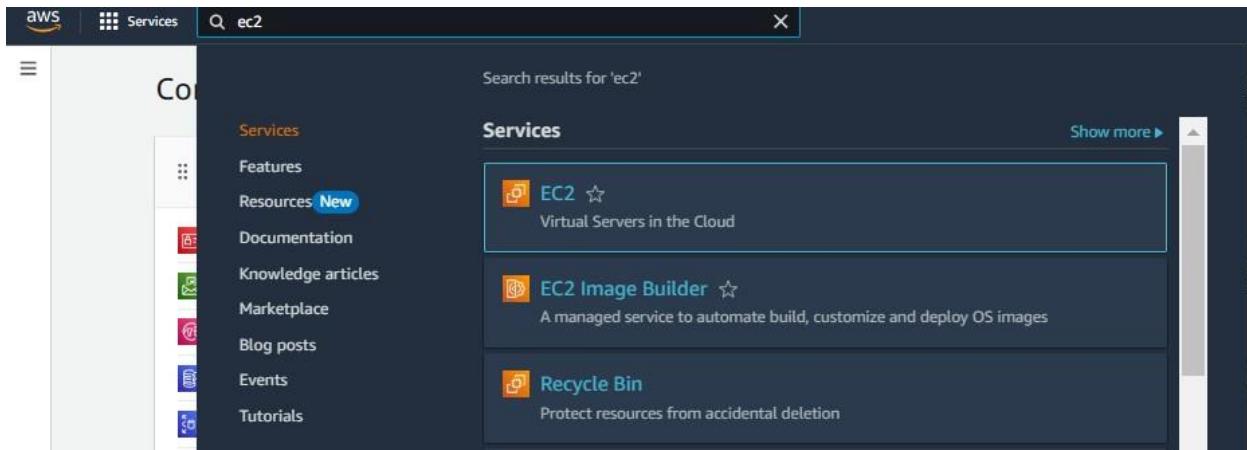
 static/images	Add files via upload
 templates	Add files via upload
 PICKLES.env	Add files via upload
 README.md	Initial commit
 app.py	Add files via upload



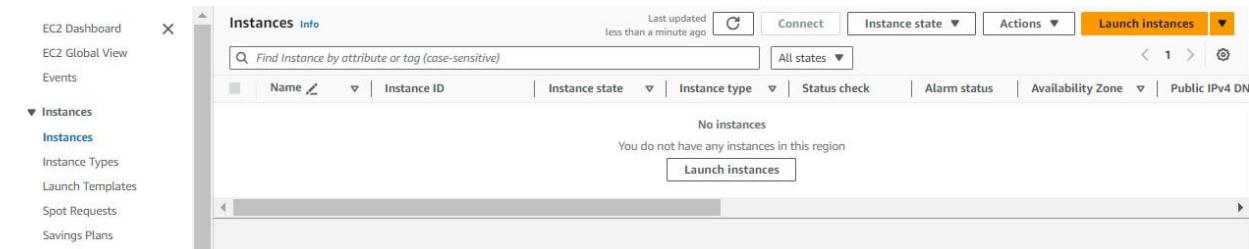
- **Activity 6.1: Launch an EC2 instance to host the Flask application.**

- **Launch EC2 Instance**

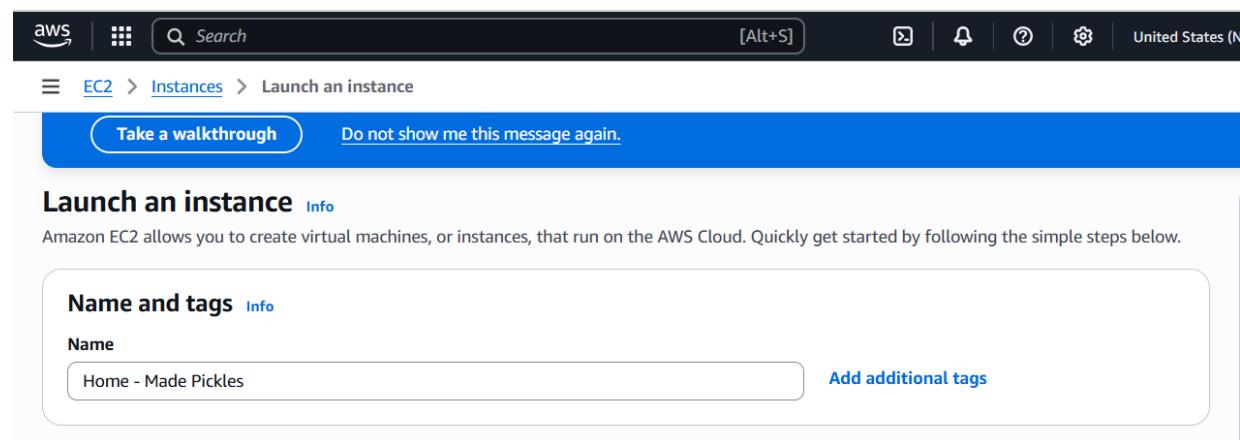
- In the AWS Console, navigate to EC2 and launch a new instance.



- Click on Launch instance to launch EC2 instance

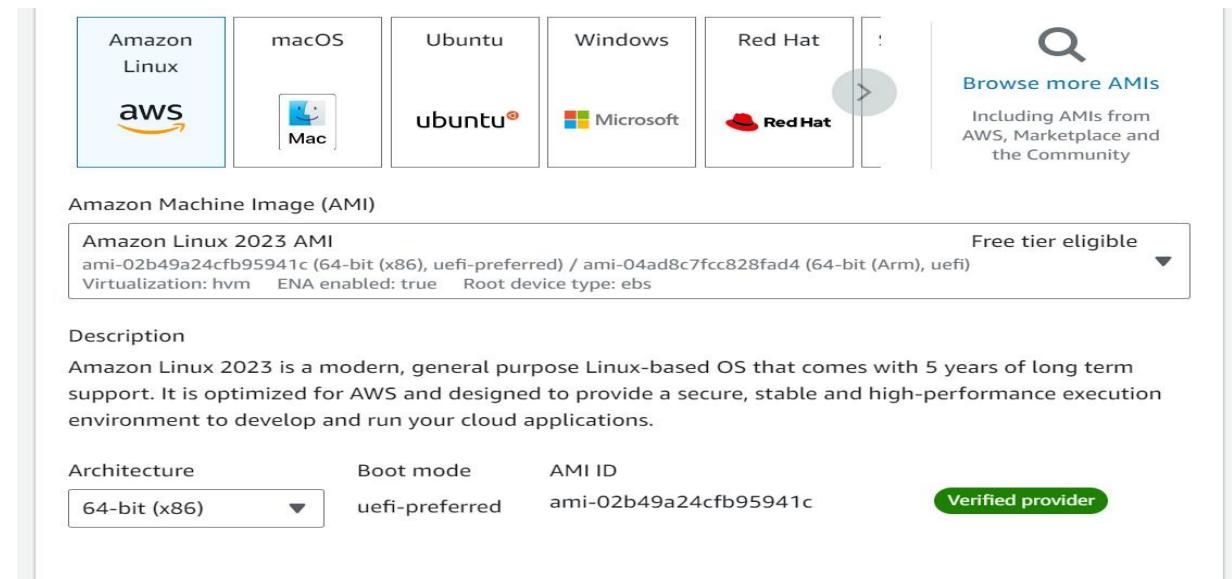


The screenshot shows the AWS EC2 Instances page. On the left, there is a sidebar with options like EC2 Dashboard, EC2 Global View, Events, Instances (selected), Instance Types, Launch Templates, Spot Requests, and Savings Plans. The main area has a heading 'Instances Info' with a search bar and filters for Name, Instance ID, Instance state, Instance type, Status check, Alarm status, Availability Zone, and Public IPv4 DNS. A message says 'No instances' and 'You do not have any instances in this region'. At the bottom right is a large 'Launch instances' button.



The screenshot shows the 'Launch an instance' wizard. The top navigation bar includes the AWS logo, a search bar, and links for EC2, Instances, Launch an instance, Take a walkthrough (buttoned), and Do not show me this message again. The main section is titled 'Launch an instance' with a 'Info' link. It says '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.' Below this is a form for 'Name and tags' with a 'Info' link. It has a 'Name' field containing 'Home - Made Pickles' and a 'Add additional tags' link.

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



The screenshot shows the 'Amazon Machine Image (AMI)' selection screen. It features a grid of icons for various AMIs: Amazon Linux, macOS, Ubuntu, Windows, Red Hat, and others. To the right is a 'Browse more AMIs' section with a magnifying glass icon and text about including AMIs from AWS, Marketplace, and the Community. Below this is a detailed view of the 'Amazon Linux 2023 AMI' (ami-02b49a24cfb95941c). It shows it is 64-bit (x86), uefi-preferred, and has an AMI ID of ami-02b49a24cfb95941c. It is marked as 'Free tier eligible'. Other details include Virtualization: hvm, ENA enabled: true, and Root device type: ebs. Below this is a 'Description' section for Amazon Linux 2023, which states it is a modern, general purpose Linux-based OS with 5 years of long term support, optimized for AWS. It also lists Architecture (64-bit (x86)), Boot mode (uefi-preferred), and AMI ID (ami-02b49a24cfb95941c), with a 'Verified provider' badge.

- Create and download the key pair for Server access.

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

**Instance type**

<b>t2.micro</b>	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	

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](#)

▼ **Network Settings** [Info](#)

VPC - required [Info](#)

vpc-0712b6d2c7cd977a4  
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 for your instance.

Create security group [Select existing](#)

Security group name - required

launch-wizard-1

This security group will be added to all network interfaces. The name can't contain spaces or punctuation marks. It must start with a letter (a-z, A-Z), 0-9, underscores (\_), or hyphens (-).

Description - required [Info](#)

launch-wizard-1 created 2025-07-03T10:53:21.831Z

Inbound Security Group Rules

### Create key pair

**Key pair name**  
Key pairs allow you to connect to your instance securely.

**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](#)


**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	Boot mode	AMI ID	Username
64-bit (x86)	uefi-preferred	ami-078264b8ba71bc45e	ec2-user

**Verified provider**

**▼ 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			

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

**Can** **Preview** **Launch**

**cel** **code** **instance**

**▼ 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 IOPS, 1 GB of snapshots, and 100 GB of bandwidth to the internet.

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

EC2 > Instances > Launch an instance

▼ Network settings [Info](#)

VPC - required [Info](#)  
vpc-0712b6d2c7cd977a4 (default) [Edit](#)

Subnet [Info](#)  
No preference [Edit](#) Create new subnet [Edit](#)

Availability Zone [Info](#)  
No preference [Edit](#) Enable additional zones [Edit](#)

Auto-assign public IP [Info](#)  
Enable [Edit](#)

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-1 [Edit](#)

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-1 created 2025-07-03T11:10:03.099Z [Edit](#)

Inbound Security Group Rules

▼ Summary

Number of instances [Info](#)  
1 [Edit](#)

Software Image (AMI)  
Amazon Linux 2023 AMI 2023.7.2... [read more](#)  
ami-05ffe3c48a9991133

Virtual server type (instance type)  
t2.micro

Firewall (security group)  
New security group

Storage (volumes)  
1 volume(s) - 8 GiB

[Cancel](#) [Launch instance](#) [Preview code](#)

Inbound Security Group Rules

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

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

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

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

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

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

[Add security group rule](#)

EC2 > ... > Launch an Instance

**Success**  
Successfully initiated launch of instance i-001861022fbac290

▶ Launch log

**Next Steps**

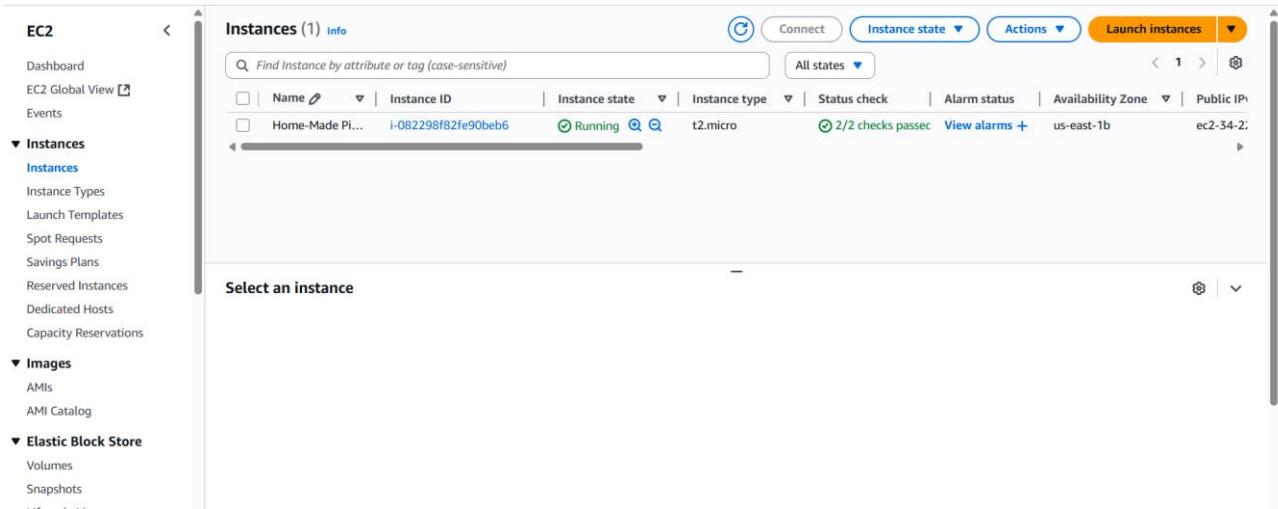
Q. What would you like to do next with this instance, for example "create alarm" or "create backup"?

< 1 2 3 4 >

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 a 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 unresponsive 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.



The screenshot shows the AWS EC2 Instances page. On the left, there is a navigation sidebar with the following menu items:

- 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
  - Volumes
  - Snapshots

The main content area displays the following information:

**Instances (1) Info**

Find Instance by attribute or tag (case-sensitive)

All states

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IP
Home-Made Pi...	i-082298f82fe90beb6	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1b	ec2-34-2...

Select an instance

☰ EC2 > Instances > i-0536cfac0174f217d > Modify IAM role

### Modify IAM role Info

Attach an IAM role to your instance.

**Instance ID**  
 ⓘ i-0536cfac0174f217d (Home-Made Pickles)

**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.

Pickle\_role ▼  ⓘ Create new IAM role

ⓘ Cancel Update IAM role

aws | ⚡ | Search [Alt+S]

IAM > Roles > Pickle\_role

ⓘ Delete

**Pickle\_role** Info

Allows EC2 instances to call AWS services on your behalf.

Summary		<span> ⓘ</span> <span>Edit</span>
Creation date	July 03, 2025, 17:12 (UTC+05:30)	ARN
		<span> ⓘ</span> arn:aws:iam::145023099530:role/Pickle_role
Last activity	-	Maximum session duration
		1 hour

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

- Now connect the EC2 with the files

☰ EC2 > Instances > i-082298f82fe90beb6 > Connect to instance

### Connect Info

Connect to an instance using the browser-based client.

EC2 Instance Connect    Session Manager    SSH client    EC2 serial console

Instance ID  
i-082298f82fe90beb6 (Home-Made Pickles)

Connect using a Public IP  
Connect using a public IPv4 or IPv6 address

Connect using a Private IP  
Connect using a private IP address and a VPC endpoint

Public IPv4 address  
34.228.12.60

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.  
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
~~\####\_
~~\###|_
~~\#/   https://aws.amazon.com/linux/amazon-linux-2023
~~V~'__>
~~_.'_/
~~_/_/
~/m/`_
[ec2-user@ip-172-31-30-5 ~]$
```

- Enter all the commands and Run

```
'`#_
~\###_          Amazon Linux 2023
~~\####\_
~~\###|_
~~\#/   https://aws.amazon.com/linux/amazon-linux-2023
~~V~'__>
~~_.'_/
~~_/_/
~/m/`_
[ec2-user@ip-172-31-28-89 ~]$ 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
Amazon Linux 2023 Kernel Livepatch repository
Dependencies resolved.
Nothing to do.
Complete!
Last metadata expiration check: 0:00:02 ago on Fri Jul  4 08:14:02 2025.
Package python3-3.9.23-1.amzn2023.0.1.x86_64 is already installed.
Dependencies resolved.
=====
Package           Architecture      Version       Repository      Size
=====
Installing:
```

i-0536cfac0174f217d (Home-Made Pickles)

PublicIPs: 3.89.249.40 PrivateIPs: 172.31.28.89

## 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/GowriTulasi/Home-Made.git>

This will download your project to the EC2 instance.

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

```
cd Home-Made
```

**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

```
2025-07-09 18:41:40,736 - INFO - 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://127.0.0.1:5000
* Running on http://192.168.80.72:5000
2025-07-09 18:41:40,736 - INFO - Press CTRL+C to quit
2025-07-09 18:41:40,740 - INFO - * Restarting with stat
2025-07-09 18:41:45,375 - WARNING - * Debugger is active!
2025-07-09 18:41:45,375 - INFO - * Debugger PIN: 351-392-708
```

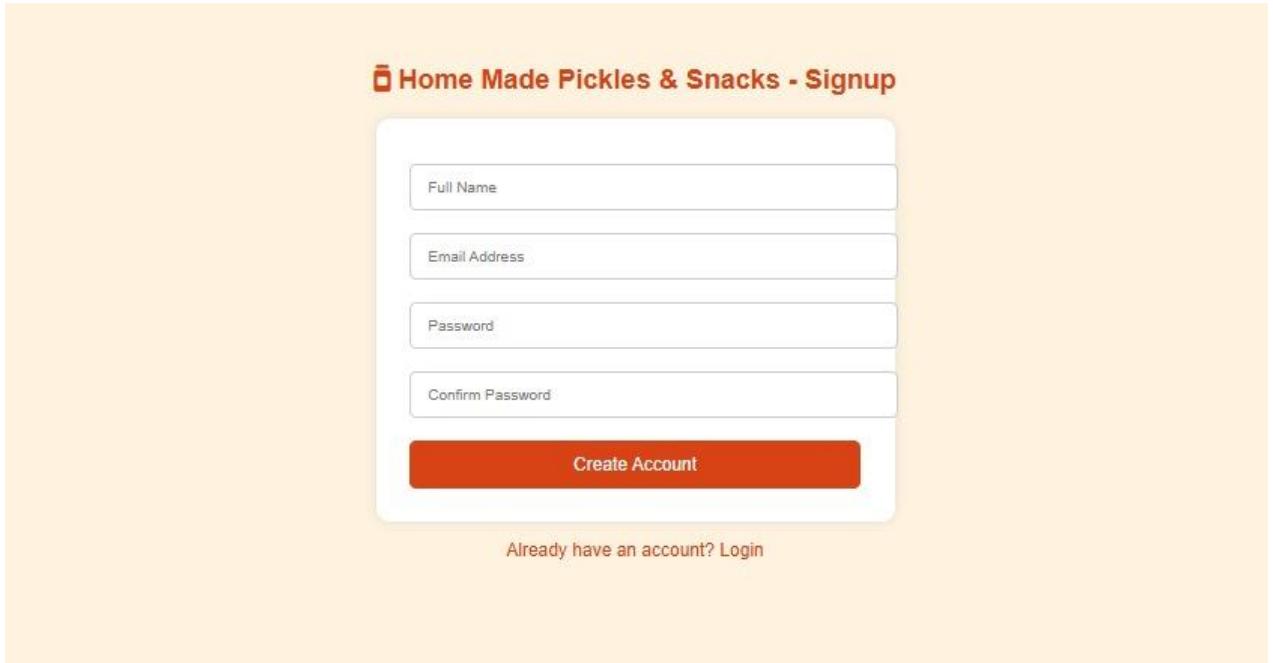
**Access the website through:**

**PUBLIC IP :<https://3.89.249.40:5000>**

## Milestone 8: Testing and Deployment

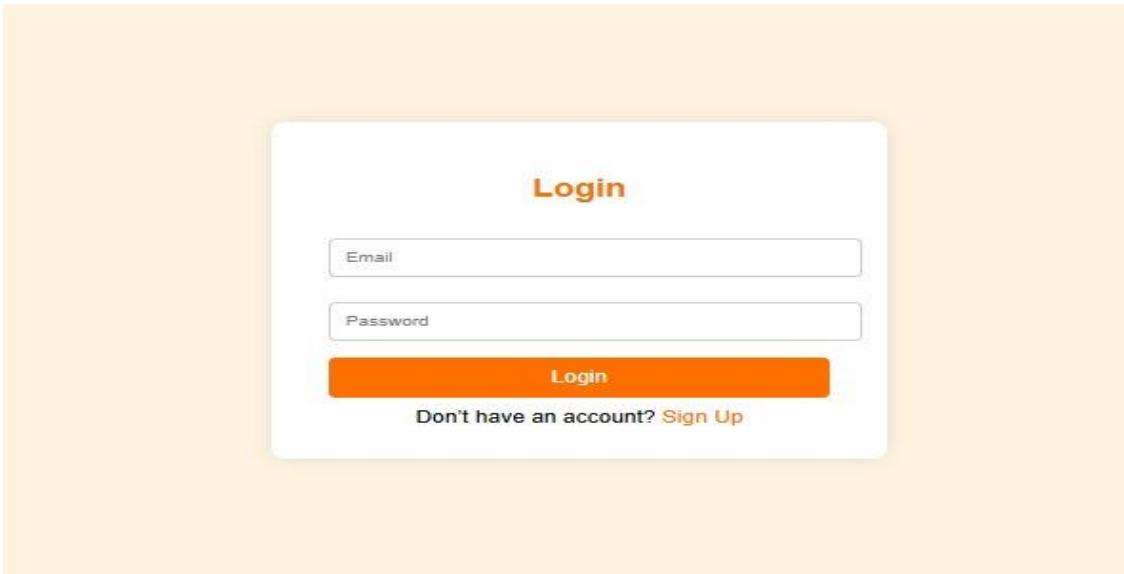
- **Activity 8.1: Conduct functional testing to verify user registration, login, order success ,order requests, and notifications.**

Sign Up page



The screenshot shows a sign-up form titled "Home Made Pickles & Snacks - Signup". The form consists of four input fields: "Full Name", "Email Address", "Password", and "Confirm Password". Below the form is a large orange "Create Account" button. At the bottom of the page, there is a link "Already have an account? Login".

Login Page:



The screenshot shows a login form titled "Login". It has two input fields: "Email" and "Password". Below the fields is an orange "Login" button. At the bottom of the form, there is a link "Don't have an account? Sign Up".

## Home Page

LOGO

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

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 perfection 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

### Snacks



**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

### Cart Items

<b>Mango Pickle</b> ₹200	<a href="#" style="border: 1px solid red; padding: 2px 10px; border-radius: 5px; color: red;">Remove</a>
<b>Chicken Pickle</b> ₹300	<a href="#" style="border: 1px solid red; padding: 2px 10px; border-radius: 5px; color: red;">Remove</a>
<b>Aam Papad</b> ₹80	<a href="#" style="border: 1px solid red; padding: 2px 10px; border-radius: 5px; color: red;">Remove</a>

Total: ₹580

[Clear Cart](#)

[Proceed to Checkout](#)

## Check out Page

### Checkout

Full Name

Email Address

Phone Number

Delivery Address

Additional Notes (optional)

**Choose Payment Method:**

Cash on Delivery

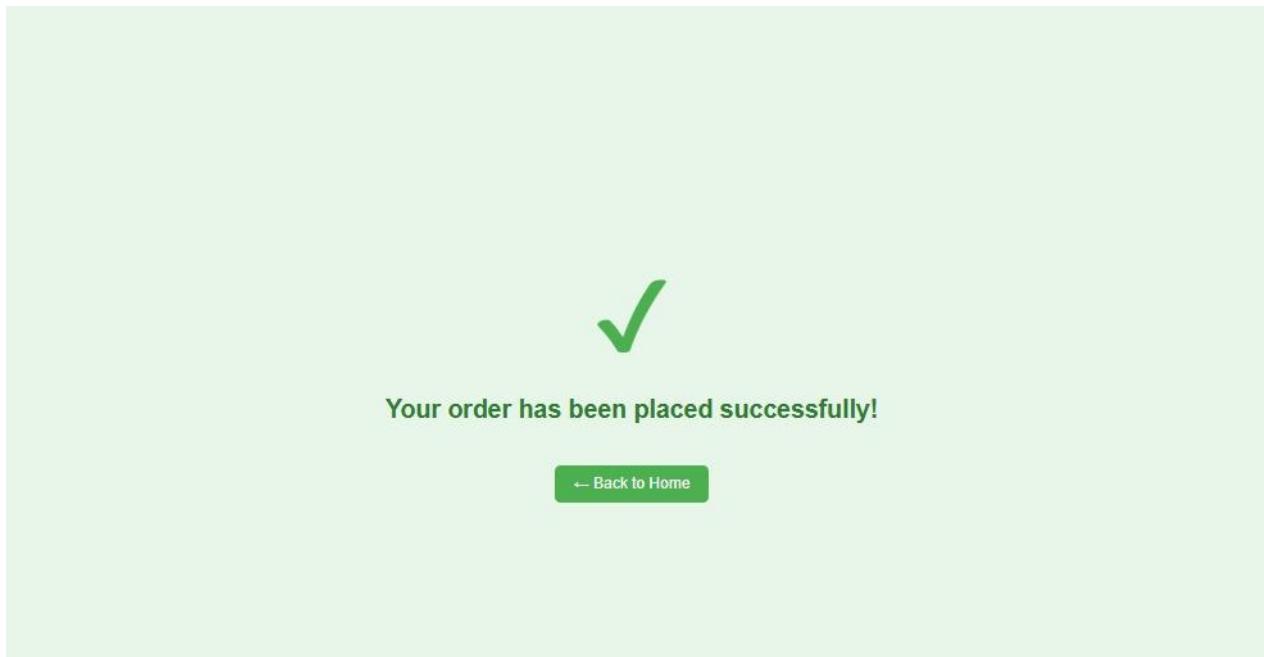
UPI

Credit Card

Place Order

[← Back to Cart](#)

### Success page



## About us page

**Homemade Pickles & Snacks**

Home   Veg Pickles   Non-Veg Pickles   Snacks   About   Contact

## 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**

Home   Veg Pickles   Non-Veg Pickles   Snacks   About   Contact

## Contact Us

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

**Full Name**

**Email**

**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.