

HomeMade Pickles & Snacks: Taste the Best

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

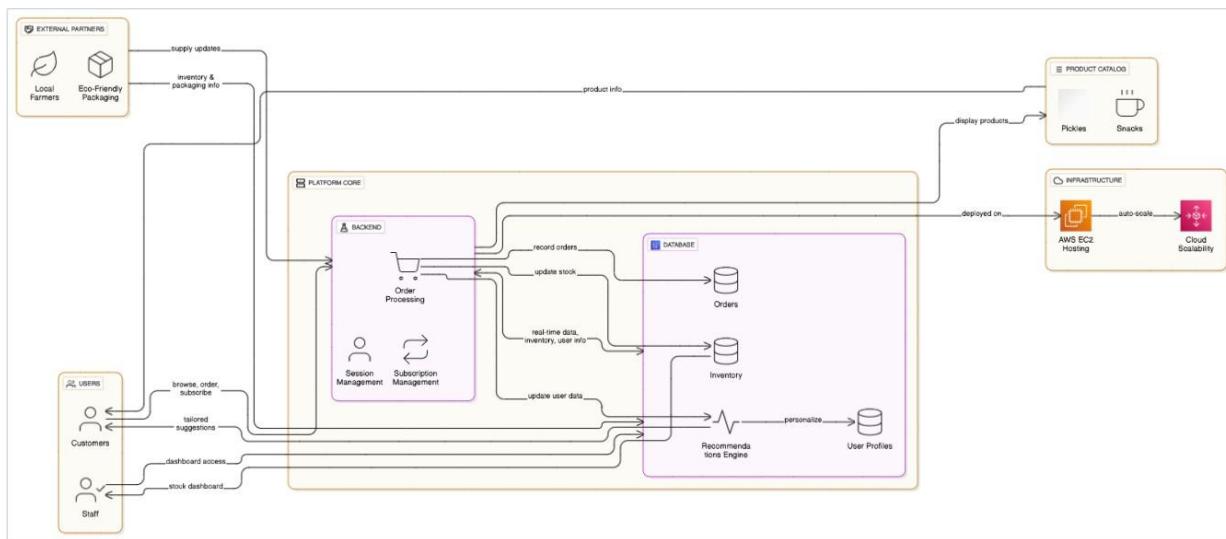
When a customer places an order for a product, the system instantly updates stock levels and records transaction details. For example, a user purchases an item, triggering automatic inventory deduction and order confirmation. Staff members receive updated dashboards to monitor stock availability and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

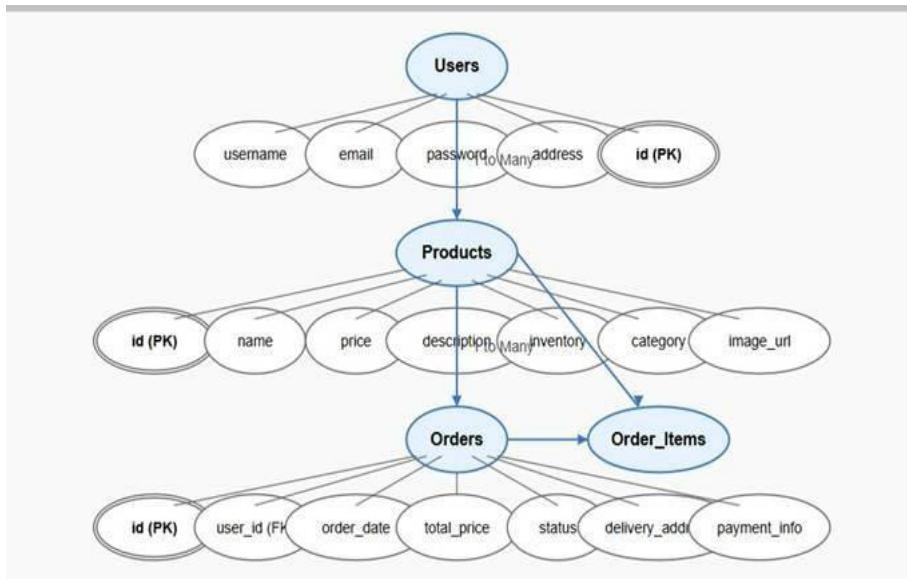
Architecture

This AWS-based architecture powers a scalable and secure web application using Amazon EC2 for hosting the backend, with a lightweight framework like Flask handling core logic. Application data is stored in Amazon DynamoDB, ensuring fast, reliable access, while user access is managed through AWS IAM for secure authentication and control. Real-time alerts and system notifications are enabled via Amazon SNS, enhancing communication and user engagement.



Entity Relationship (ER) Diagram

An ER (Entity-Relationship) diagram visually represents the logical structure of a database by defining entities, their attributes, and the relationships between them. It helps organize data efficiently by illustrating how different components of the system interact and relate. This structured approach supports effective database normalization, data integrity, and simplified query design.



Pre-requisites

- AWS Account Setup:
<https://docs.aws.amazon.com/accounts/latest/reference/getting-started.html>
- AWS IAM (Identity and Access Management):
<https://docs.aws.amazon.com/IAM/latest/UserGuide/introduction.html>
- AWS EC2 (Elastic Compute Cloud):
<https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts.html>
- AWS DynamoDB:
<https://docs.aws.amazon.com/amazondynamodb/Introduction.html>
- Git Documentation:
<https://git-scm.com/doc>
- VS Code Installation: (download the VS Code using the below link or you can get that in Microsoft store)
<https://code.visualstudio.com/download>

Project WorkFlow

Milestone 1. Backend Development and Application Setup

- Develop the Backend Using Flask.
- Integrate AWS Services Using boto3.

Milestone 2. AWS Account Setup and Login

- Set up an AWS account if not already done.
- Log in to the AWS Management Console

Milestone 3. DynamoDB Database Creation and Setup

- Create a DynamoDB Table.
- Configure Attributes for User Data and order Requests.

Milestone 4. SNS Notification Setup

- Create SNS topics for book request notifications.
- Subscribe users and library staff to SNS email notifications.

Milestone 5. IAM Role Setup

- Create IAM Role
- Attach Policies

Milestone 6. EC2 Instance Setup

- Launch an EC2 instance to host the Flask application.
- Configure security groups for HTTP, and SSH access.

Milestone 7. Deployment on EC2

- Upload Flask Files
- Run the Flask App

Milestone 8. Testing and Deployment

- Conduct functional testing to verify user signup, login, buy/sell stocks and notifications.

Milestone 1 : Backend Development and Application Setup

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.

Important Instructions:

Start by creating the necessary HTML pages and Flask routes (app.py) to build the core functionality of your application.

- During the initial development phase, store and retrieve data using Python dictionaries or lists locally. This will allow you to design, test, and validate your application logic without external database dependencies.

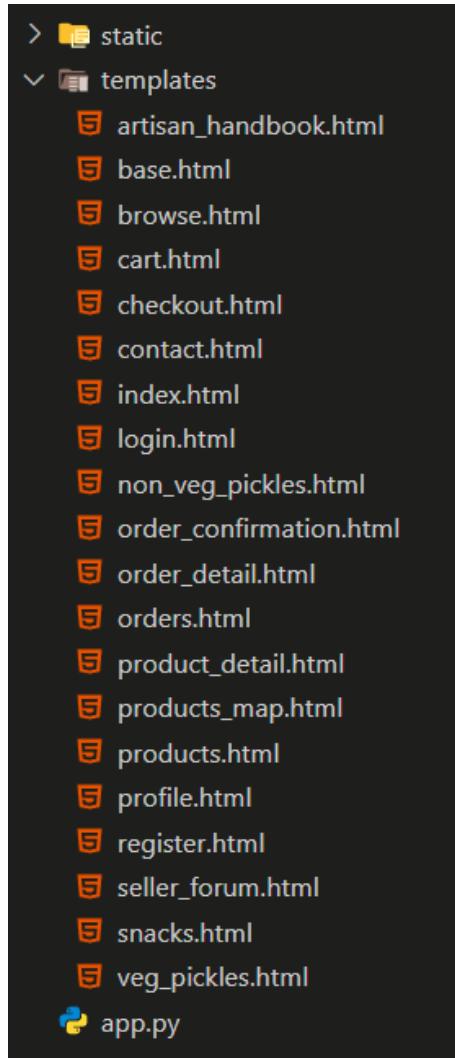
- Ensure your app runs smoothly with local data structures before integrating any cloud services.

Post Troven Access Activation:

- Once Troven Labs access is provided (valid for 3 hours), you must immediately proceed with Milestone 1 of your Guided Project instructions.
- At this point, modify your app.py and replace local dictionary/list operations with AWS services (such as DynamoDB, RDS, or others as per project requirements).
- Using the temporary credentials provided by Troven Labs, securely connect your application to AWS resources.
- Since the AWS configuration is lightweight and already instructed in the milestones, you should be able to complete the cloud integration efficiently within the allotted time.

LOCAL DEPLOYMENT

- File Explorer Structure



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
```

- Use boto3 to connect to DynamoDB for handling user registration, Order details database operations and also mention region_name where Dynamodb tables are created.

```
aws_session = boto3.Session(
    region_name=os.environ.get('AWS_REGION', 'ap-south-1')
)

dynamodb = aws_session.resource('dynamodb')
users_table = dynamodb.Table('PickleApp_Users')
products_table = dynamodb.Table('PickleApp_Products')
orders_table = dynamodb.Table('PickleApp_Orders')
cart_table = dynamodb.Table('PickleApp_Cart')
```

```
products = {
    'non_veg_pickles': [
        {'id': 1, 'name': 'Chicken Pickle', 'weights': {'250': 600, '500': 1200, '1000': 1800}},
        {'id': 2, 'name': 'Fish Pickle', 'weights': {'250': 200, '500': 400, '1000': 800}},
        {'id': 3, 'name': 'Gongura Mutton', 'weights': {'250': 400, '500': 800, '1000': 1600}},
        {'id': 4, 'name': 'Mutton Pickle', 'weights': {'250': 400, '500': 800, '1000': 1600}},
        {'id': 5, 'name': 'Prawns Pickle', 'weights': {'250': 600, '500': 1200, '1000': 1800}},
        {'id': 6, 'name': 'Chicken Pickle (Gongura)', 'weights': {'250': 350, '500': 700, '1000': 1050}},
    ],
    'veg_pickles': [
        {'id': 7, 'name': 'Traditional Mango Pickle', 'weights': {'250': 150, '500': 280, '1000': 500}},
        {'id': 8, 'name': 'Zesty Lemon Pickle', 'weights': {'250': 120, '500': 220, '1000': 400}},
        {'id': 9, 'name': 'Tomato Pickle', 'weights': {'250': 130, '500': 240, '1000': 450}},
        {'id': 10, 'name': 'Mix Veg Pickle', 'weights': {'250': 130, '500': 240, '1000': 450}},
        {'id': 12, 'name': 'Spicy Pandu Mirchi', 'weights': {'250': 130, '500': 240, '1000': 450}},
    ],
    'snacks': [
        {'id': 13, 'name': 'Banana Chips', 'weights': {'250': 300, '500': 600, '1000': 800}},
        {'id': 14, 'name': 'Crispy Aam-Papad', 'weights': {'250': 150, '500': 300, '1000': 600}},
        {'id': 16, 'name': 'Sweet Boondhi', 'weights': {'250': 300, '500': 600, '1000': 900}},
        {'id': 17, 'name': 'Chekkalu', 'weights': {'250': 350, '500': 700, '1000': 1000}},
        {'id': 18, 'name': 'Ragi Laddu', 'weights': {'250': 350, '500': 700, '1000': 1000}},
        {'id': 19, 'name': 'Dry Fruit Laddu', 'weights': {'250': 500, '500': 1000, '1000': 1500}},
        {'id': 20, 'name': 'Kara Boondi', 'weights': {'250': 250, '500': 500, '1000': 750}},
        {'id': 21, 'name': 'Gavvalu', 'weights': {'250': 250, '500': 500, '1000': 750}},
    ]
}
```

- Routes for Web Pages
- Login Route (GET/POST): Verifies user credentials, increments login count, and redirects to the dashboard on success.

```

@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        email = request.form['email']
        password = request.form['password']
        if not email or not password:
            flash("Email and password are required!", "danger")
            return redirect(url_for('login'))
        user, mode = get_user_by_email(email)
        if not user or not checkpw(password.encode('utf-8'), user['password'].encode('utf-8')):
            flash("Incorrect email or password! Please try again.", "danger")
            return redirect(url_for('login'))
        if mode == 'dynamodb':
            try:
                users_table.update_item(
                    Key={'email': email},
                    UpdateExpression='SET login_count = if_not_exists(login_count, :zero) + :inc',
                    ExpressionAttributeValues={':inc': 1, ':zero': 0}
                )
            except Exception as e:
                print(f"DynamoDB update error: {e}")
        elif mode == 'local':
            users = load_users()
            for u in users:
                if u['email'] == email:
                    u['login_count'] = u.get('login_count', 0) + 1
            save_users(users)
        session['user_email'] = user['email']
        session['user_name'] = user['name']
        session['phone'] = user.get('phone', '')
        flash("Login successful!", "success")
        next_page = request.form.get('next')
        if next_page and next_page.startswith('/'):
            return redirect(next_page)
        return redirect(url_for('profile'))
    return render_template('login.html')

```

- SignUp route: Collecting registration data, hashes the password, and stores user details in the database.

```

@app.route('/register', methods=['GET', 'POST'])
def register():
    if request.method == 'POST':
        name = request.form['name']
        email = request.form['email']
        password = request.form['password']
        confirm_password = request.form['confirm_password']
        phone = request.form['phone']
        if not name or not email or not password or not confirm_password or not phone:
            flash("All fields are mandatory! Please fill out the entire form.", "danger")
            return redirect(url_for('register'))
        if password != confirm_password:
            flash("Passwords do not match! Please try again.", "danger")
            return redirect(url_for('register'))
        user, mode = get_user_by_email(email)
        if user:
            flash("User already exists! Please log in.", "info")
            return redirect(url_for('login'))
        hashed_password = hashpw(password.encode('utf-8'), gensalt()).decode('utf-8')
        user_obj = {
            'email': email,
            'name': name,
            'password': hashed_password,
            'phone': phone,
            'login_count': 0,
            'street': '',
            'city': '',
            'pincode': '',
            'country': 'India',
            'registration_date': datetime.now().strftime("%Y-%m-%d %H:%M:%S")
        }
        save_user(user_obj, 'dynamodb' if mode == 'dynamodb' else 'local')
        try:
            sns.publish(
                TopicArn=sns_topic_arn,
                Message=f'New user registered: {name} ({email})',
                Subject='New User Registration'
            )
        except Exception as e:
            print(f"SNS error: {e}")
        try:
            send_email(email, "Welcome to PickleApp!", f"Hi {name},\n\nThank you for registering at PickleApp!")
        except Exception as e:
            print(f"Email error: {e}")
        session['user_email'] = email
        session['user_name'] = name
        session['phone'] = phone
        flash("Registration successful!", "success")
        return redirect(url_for('profile'))
    return render_template('register.html')

```

- Logout route: The user can Logout so that the user can get back to the Login Page

```
@app.route('/logout')
def logout():
    session.clear()
    flash("Logged out successfully!", "success")
    return redirect(url_for('login'))
```

- Home Route: Home page contains the routing for different categories which are Veg_pickles,Non_Veg_pickles,Snacks.

```
@app.route('/veg-pickles')
def veg_pickles():
    products = get_products_by_category('veg_pickles')
    return render_template('veg_pickles.html', products=products, category='veg_pickles')

@app.route('/non-veg-pickles')
def non_veg_pickles():
    products = get_products_by_category('non_veg_pickles')
    return render_template('non_veg_pickles.html', products=products, category='non_veg_pickles')

@app.route('/snacks')
def snacks():
    products = get_products_by_category('snacks')
    return render_template('snacks.html', products=products, category='snacks')
```

- Check out Route:

-

```
@app.route('/checkout')
def checkout():
    if not session.get('user_email'):
        flash("Please log in to proceed with checkout.", "danger")
        return redirect(url_for('login', next='/checkout'))
    return render_template('checkout.html')

@app.route('/order-confirmation')
def order_confirmation():
    if not session.get('user_email'):
        flash("Please log in to view order confirmation.", "danger")
        return redirect(url_for('login', next='/order-confirmation'))
    return render_template('order_confirmation.html')
```

Description: These are the image routes are used to fall back in the local host for the static images in the static folder .

Image Routes:

```
@app.route('/api/product/<int:product_id>')
def api_product(product_id):
    time.sleep(1)
    img_map = {
        'Chicken Pickle': '/static/img/chicken-pickle.jpg',
        'Fish Pickle': '/static/img/fish-pickle.jpg',
        'Gongura Mutton': '/static/img/gongura-mutton-pickle.webp',
        'Mutton Pickle': '/static/img/mutton-pickle.jpg',
        'Prawns Pickle': '/static/img/prawn-pickle.webp',
        'Chicken Pickle (Gongura)': '/static/img/gongura-chicken-pickle.jpg',
        'Traditional Mango Pickle': '/static/img/mango-pickle.jpg',
        'Zesty Lemon Pickle': '/static/img/zesty-lemon-pickle.png',
        'Tomato Pickle': '/static/img/tomato-pickle.webp',
        'Mix Veg Pickle': '/static/img/mixed-veg-pickle.jpg',
        'Spicy Pandu Mirchi': '/static/img/red-chilli-pickle.jpg',
        'Banana Chips': '/static/img/banana-chips.jpg',
        'Crispy Aam-Papad': '/static/img/crispy-aam-papad.jpg',
        'Sweet Boondhi': '/static/img/sweet-boondhi.jpg',
        'Chekkalu': '/static/img/chekkalu.jpg',
        'Ragi Laddu': '/static/img/ragi-ladoo.jpg',
        'Dry Fruit Laddu': '/static/img/dry-fruit.jpg',
        'Kara Boondi': '/static/img/kaaraa-boondhi.jpg',
        'Gavvalu': '/static/img/gavvalu.jpg',
    }
    desc_map = {
        'non_veg_pickles': 'Authentic homemade non-veg pickles, rich in flavor and tradition.',
        'veg_pickles': 'Classic vegetarian pickles made with fresh ingredients.',
        'snacks': 'Crispy, crunchy, and delicious snacks for every occasion.',
    }
```

Milestone 2 : AWS Account Setup

Important Notice: Use Troven Labs for AWS Access

Students are strictly advised not to create their own AWS accounts, as doing so may incur charges. Instead, we have set up a dedicated section called “Labs” on the Troven platform, which provides temporary and cost-free access to AWS services.

Once your website is locally deployed and fully functional, you must proceed with integrating AWS services only through the Troven Labs environment. This ensures secure, controlled access to AWS resources without any risk of personal billing.

All steps involving AWS (such as deploying to EC2, connecting to DynamoDB, or using SNS) must be carried out within the Troven Labs platform, as we've configured temporary credentials for each student.

Please follow the provided guidelines and access AWS exclusively through Troven to avoid unnecessary issues.

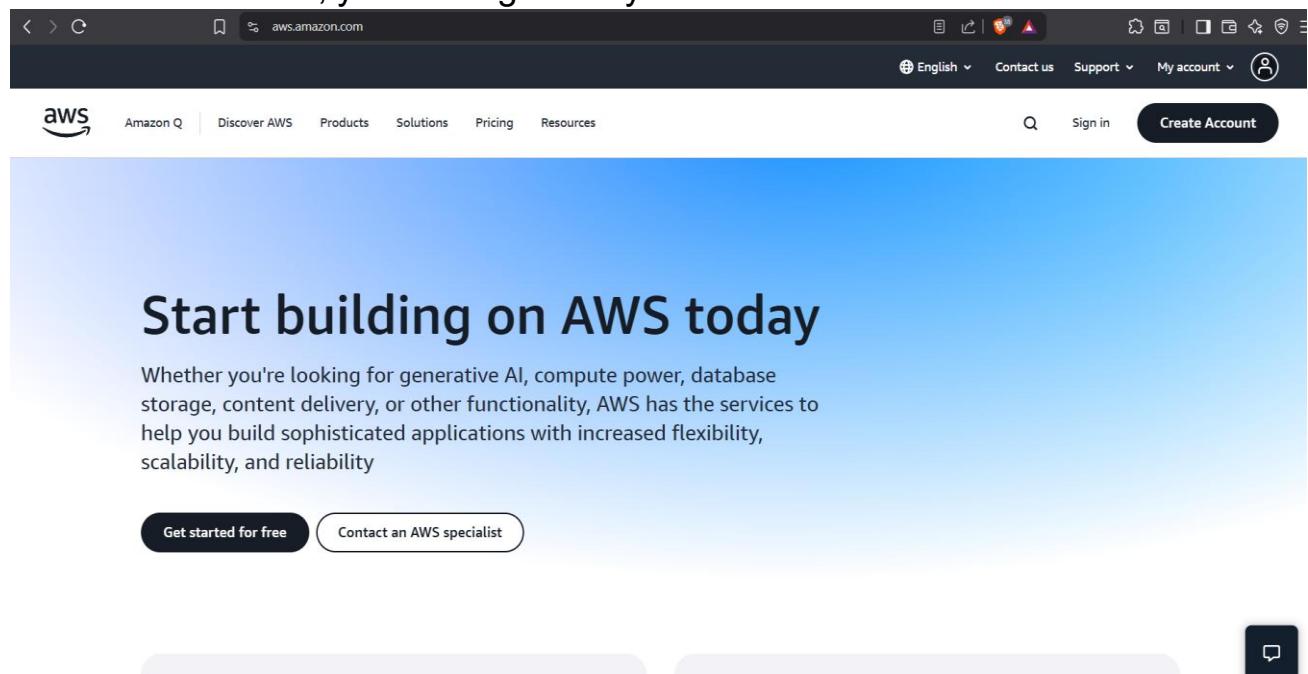
Please refer the below link -

<https://drive.google.com/file/d/1HzWc7AMJ2BrxhV-uaw5s0vWtcd-28qgl/view?usp=sharing>

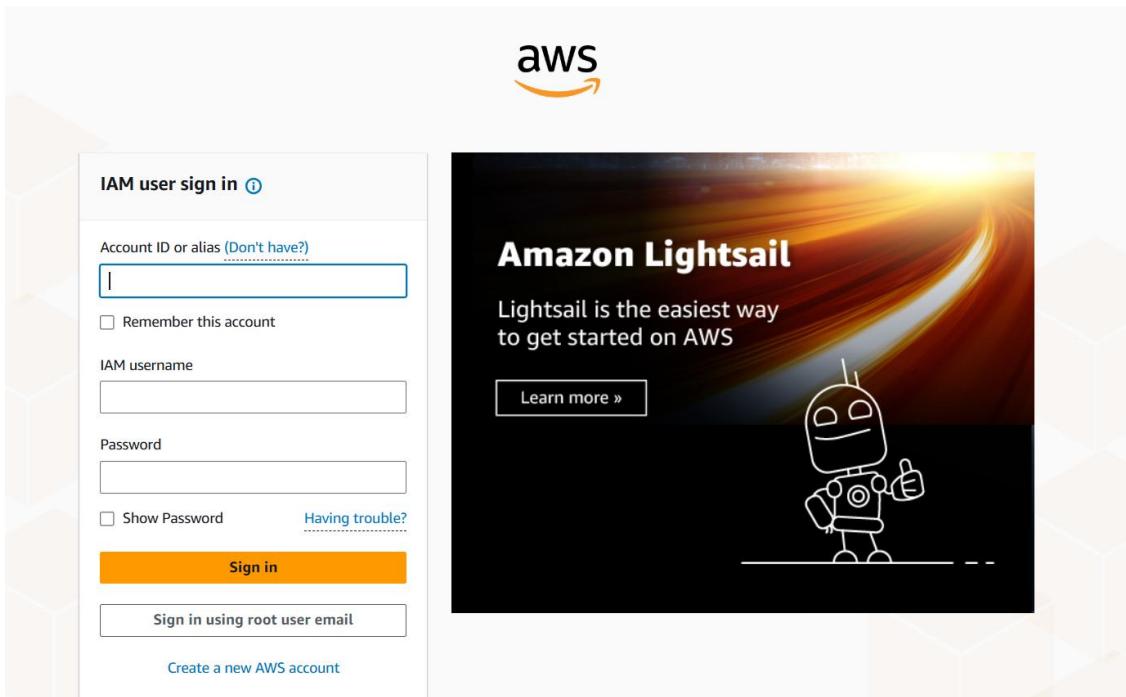
AWS Account Setup and Login

This is for your understanding only, please refrain from creating an AWS account. A temporary account will be provided via Troven.

- Go to the AWS website (<https://aws.amazon.com/>).
- Click on the "Create an AWS Account" button.
- Follow the prompts to enter your email address and choose a password.
- Provide the required account information, including your name, address, and phone number.
- Enter your payment information. (Note: While AWS offers a free tier, a credit card or debit card is required for verification.)
- Complete the identity verification process.
- Choose a support plan (the basic plan is free and sufficient for starting).
- Once verified, you can sign in to your new AWS accounts.



The screenshot shows the AWS homepage. At the top, there is a dark navigation bar with various icons and links. Below it, the main header features the AWS logo and the text "Start building on AWS today". A descriptive paragraph follows, stating: "Whether you're looking for generative AI, compute power, database storage, content delivery, or other functionality, AWS has the services to help you build sophisticated applications with increased flexibility, scalability, and reliability". At the bottom of the main content area, there are two buttons: "Get started for free" and "Contact an AWS specialist".



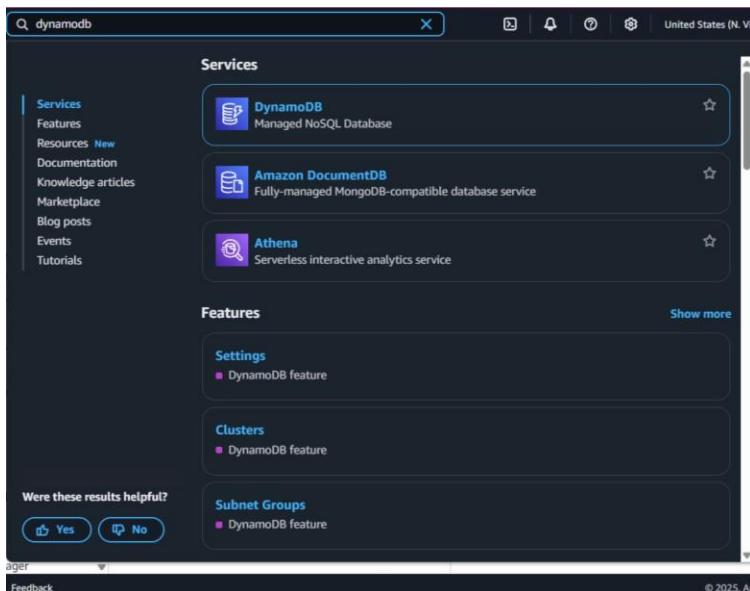
- Log in to the AWS Management Console
- After setting up your account, log in to the [AWS Management Console](#).

Milestone 3 : DynamoDB Database Creation and Setup

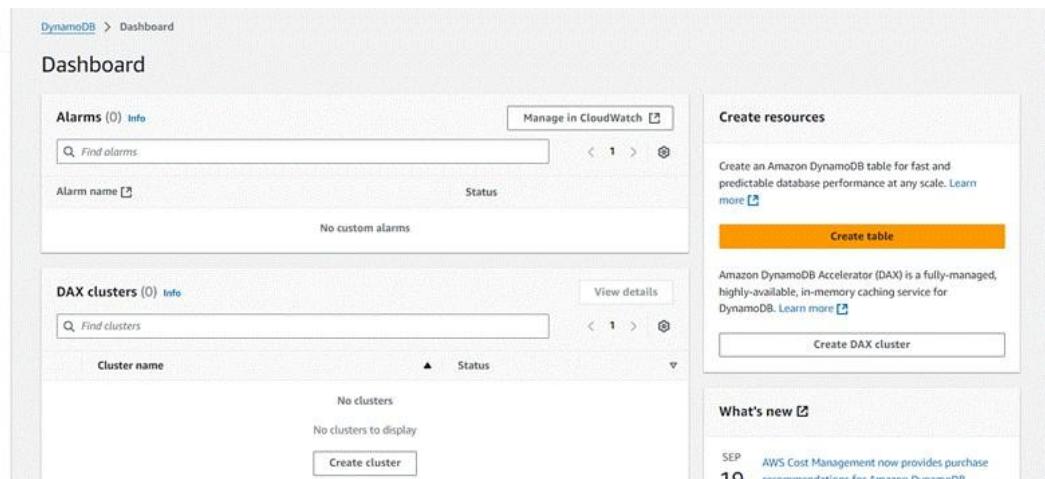
Database Creation and Setup involves initializing a cloud-based NoSQL database to store and manage application data efficiently. This step includes defining tables, setting primary keys, and configuring read/write capacities. It ensures scalable, high- performance data storage for seamless backend operations.

Navigate to the DynamoDB

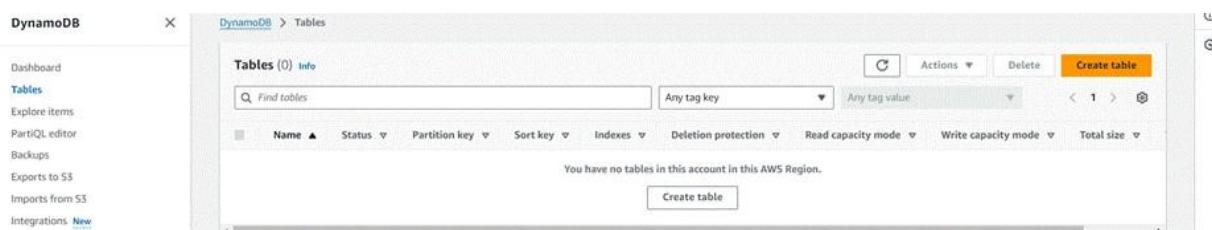
- In the AWS Console, navigate to DynamoDB and click on create tables.



The screenshot shows the AWS search interface with the query 'dynamodb' entered. The top result is 'DynamoDB Managed NoSQL Database'. Other results include 'Amazon DocumentDB Fully-managed MongoDB-compatible database service' and 'Athena Serverless interactive analytics service'. On the left sidebar, there are sections for 'Services', 'Features', and 'Documentation'. A feedback section at the bottom asks if the results were helpful, with 'Yes' and 'No' buttons.



The screenshot shows the DynamoDB dashboard. The left sidebar includes options like 'Dashboard', 'Tables', 'Explore items', 'PartiQL editor', 'Backups', 'Exports to S3', 'Imports from S3', 'Integrations', 'Reserved capacity', 'Settings', 'Clusters', 'Subnet groups', 'Parameter groups', and 'Events'. The main area displays 'Alarms (0) Info' and 'DAX clusters (0) Info'. A 'Create resources' section on the right allows users to 'Create table' or 'Create DAX cluster'. A 'What's new' section at the bottom right mentions 'AWS Cost Management now provides purchase recommendations for Amazon DynamoDB...'.



The screenshot shows the 'Tables' page under the 'DynamoDB' service. The left sidebar is identical to the dashboard. The main area shows a table header with columns for 'Name', 'Status', 'Partition key', 'Sort key', 'Indexes', 'Deletion protection', 'Read capacity mode', 'Write capacity mode', and 'Total size'. A message at the bottom states 'You have no tables in this account in this AWS Region.' A prominent 'Create table' button is located at the bottom center.

Create a DynamoDB table for storing data

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

DynamoDB > Tables > Create table

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.

Table class	DynamoDB Standard	Yes
Capacity mode	Provisioned	Yes
Provisioned read capacity	5 RCU	Yes
Provisioned write capacity	5 WCU	Yes
Auto scaling	On	Yes
Local secondary indexes	-	No
Global secondary indexes	-	Yes
Encryption key management	Owned by Amazon DynamoDB	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.

Create table

- Follow the same steps to create an PickleApp_Cart table

DynamoDB > Tables > Create table

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.

Table class	DynamoDB Standard	Yes
Capacity mode	Provisioned	Yes
Provisioned read capacity	5 RCU	Yes
Provisioned write capacity	5 WCU	Yes
Auto scaling	On	Yes
Local secondary indexes	-	No
Global secondary indexes	-	Yes
Encryption key management	Owned by Amazon DynamoDB	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.

Cancel **Create table**

- Now for the PickleApp_Orders and for the PickleApp_Products.

DynamoDB > Tables > Create table

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.
 Number ▾
 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 class	DynamoDB Standard	Yes
Capacity mode	Provisioned	Yes
Provisioned read capacity	5 RCU	Yes
Provisioned write capacity	5 WCU	Yes
Auto scaling	On	Yes
Local secondary indexes	-	No
Global secondary indexes	-	Yes
Encryption key management	Owned by Amazon DynamoDB	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.

 You can add 50 more tags.

DynamoDB > Tables > Create table

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.

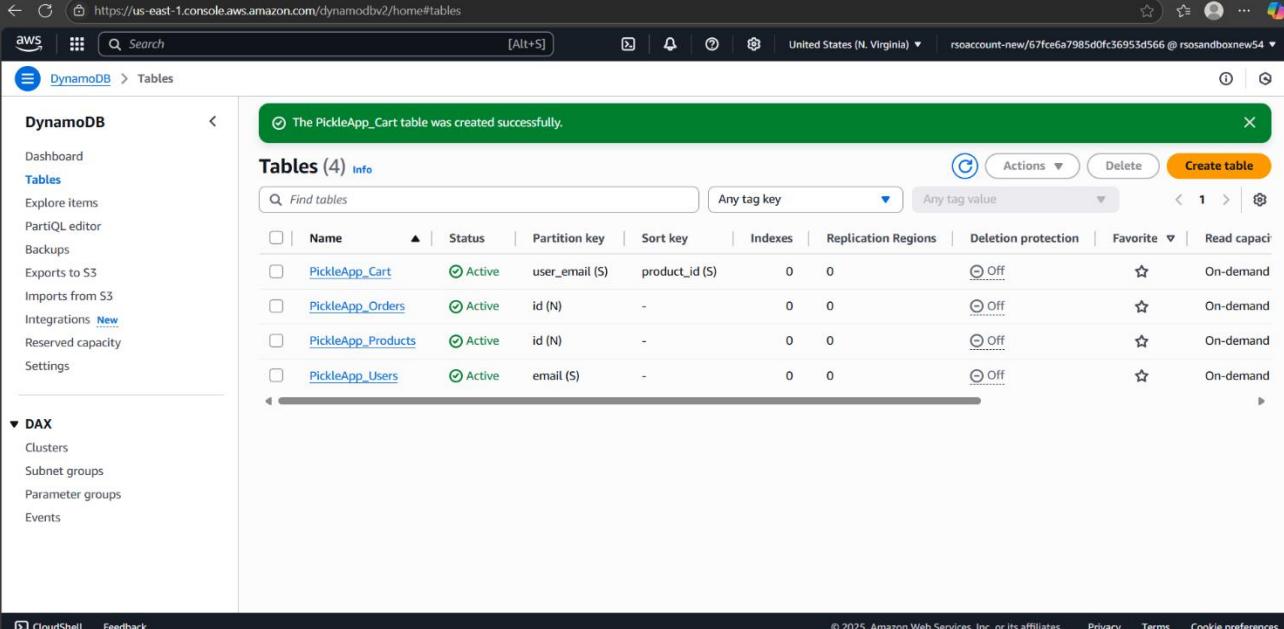
Table class	DynamoDB Standard	Yes
Capacity mode	Provisioned	Yes
Provisioned read capacity	5 RCU	Yes
Provisioned write capacity	5 WCU	Yes
Auto scaling	On	Yes
Local secondary indexes	-	No
Global secondary indexes	-	Yes
Encryption key management	Owned by Amazon DynamoDB	Yes
Deletion protection	Off	Yes
Resource-based 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.

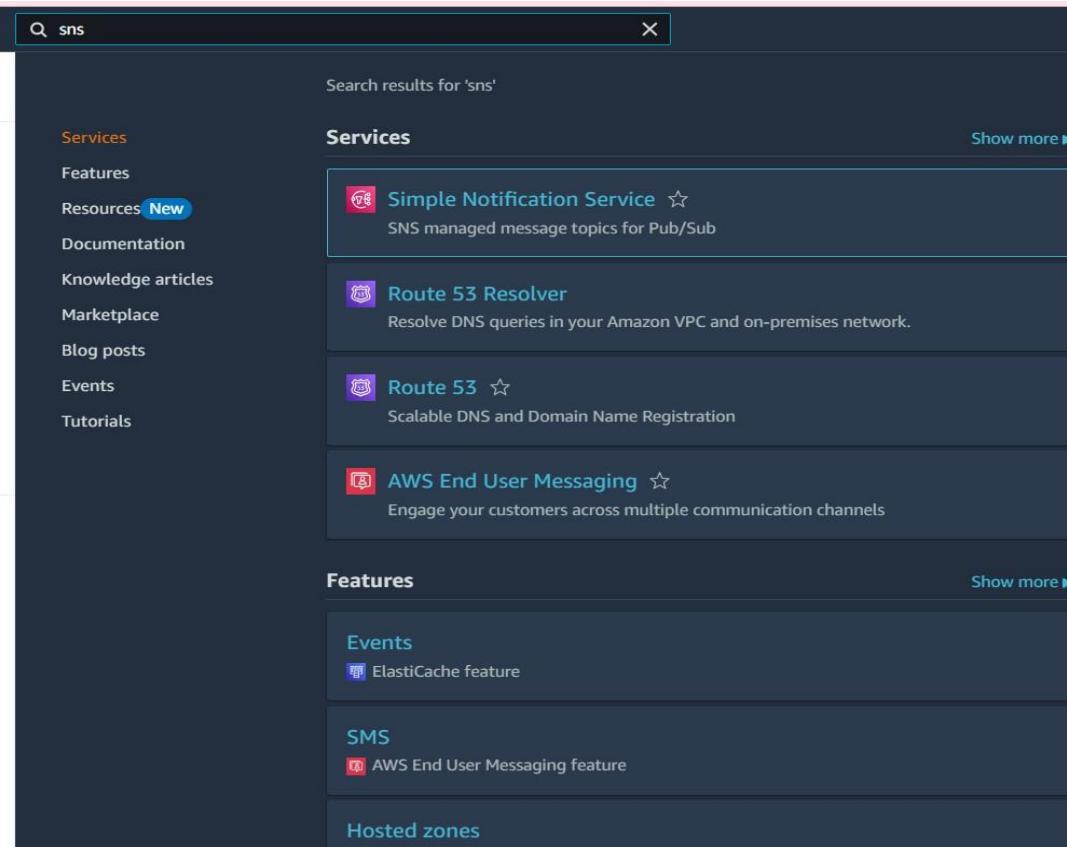
You can add 50 more tags.



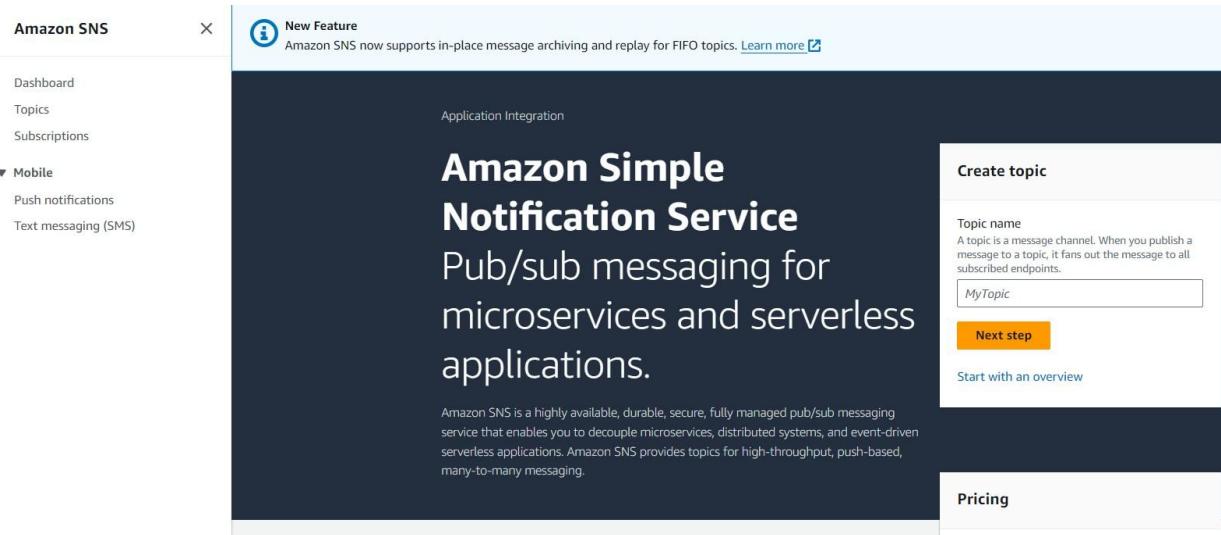
The screenshot shows the AWS DynamoDB console with the URL <https://us-east-1.console.aws.amazon.com/dynamodbv2/home#tables>. A success message at the top right states: "The PickleApp_Cart table was created successfully." The main table view displays four tables: PickleApp_Cart, PickleApp_Orders, PickleApp_Products, and PickleApp_Users. Each table has its status set to Active, with various attributes like partition key, sort key, and indexes defined.

Milestone 4 : SNS Notification Setup:

- Create SNS topics for sending email notifications for ordering alerts
 - In the AWS Console, search for SNS and navigate to the SNS Dashboard.

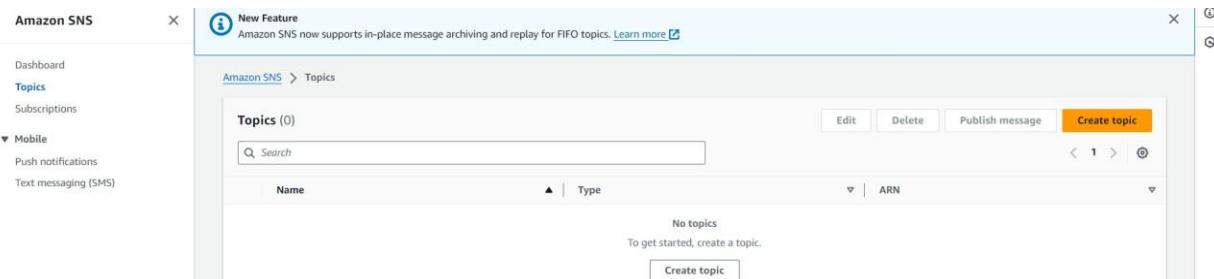


The screenshot shows the AWS search results for 'sns'. The search bar at the top contains 'sns'. Below it, the results are displayed under the 'Services' section. The first result is 'Simple Notification Service' (marked with a star), described as 'SNS managed message topics for Pub/Sub'. Other services listed include 'Route 53 Resolver', 'Route 53', and 'AWS End User Messaging'. Below the services, there are sections for 'Events' (ElastiCache feature) and 'SMS' (AWS End User Messaging feature). A 'Show more' link is visible at the top right of each service section.



The screenshot shows the Amazon SNS dashboard. On the left, there is a sidebar with options: Dashboard, Topics, Subscriptions, Mobile (Push notifications, Text messaging (SMS)), and a New Feature notice about FIFO topics. The main content area has a dark header "Application Integration" and a large title "Amazon Simple Notification Service" with a subtitle "Pub/sub messaging for microservices and serverless applications." Below this is a descriptive paragraph about Amazon SNS. To the right, a white modal window titled "Create topic" is open, containing a "Topic name" input field with "MyTopic" typed into it, a "Next step" button, and a "Start with an overview" link. At the bottom right of the main content area, there is a "Pricing" link.

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



The screenshot shows the "Topics" page within the Amazon SNS service. The left sidebar includes "Dashboard", "Topics" (which is selected and highlighted in blue), "Subscriptions", and "Mobile" (Push notifications, Text messaging (SMS)). The main content area shows a table header "Topics (0)" with columns "Name", "Type", and "ARN". A search bar and a "Create topic" button are at the top of the table. Below the table, a message says "No topics" and "To get started, create a topic." with another "Create topic" button.

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

Amazon SNS > Topics > Create topic

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
- High throughput, up to 300 publishes/second
- Subscription protocols: SQS

Standard

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

Name
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.

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.

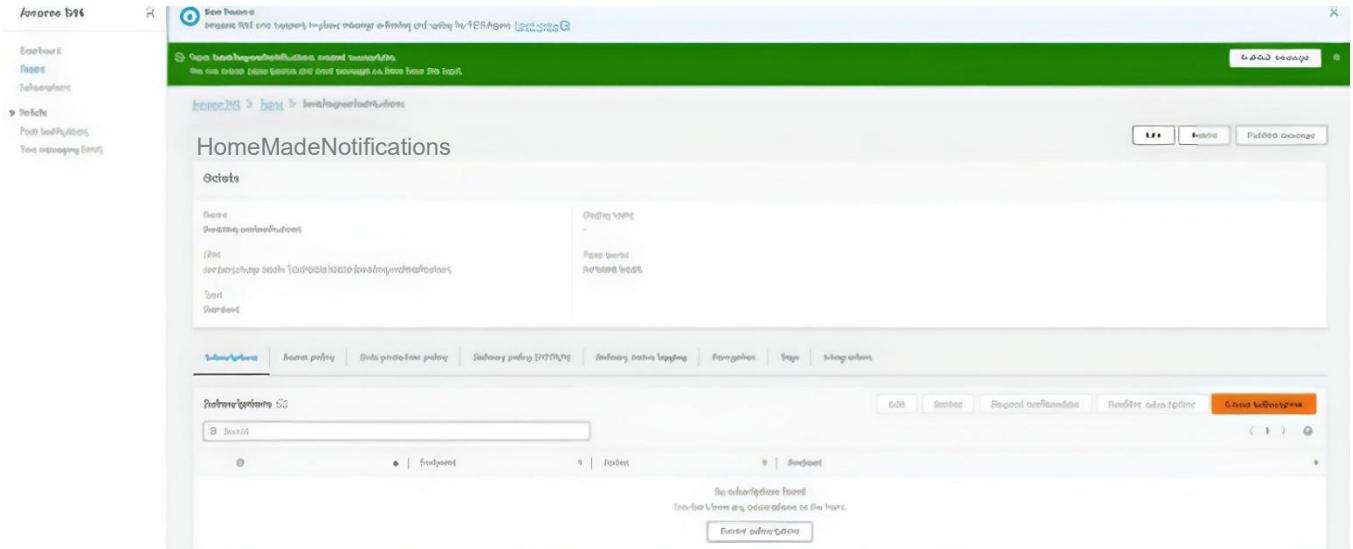
► **Delivery status logging - optional** [Info](#)
These settings configure the logging of message delivery status to CloudWatch Logs.

► **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**.



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

Amazon SNS > Subscriptions > Create subscription

Create subscription

Details

Topic ARN: X

Protocol:

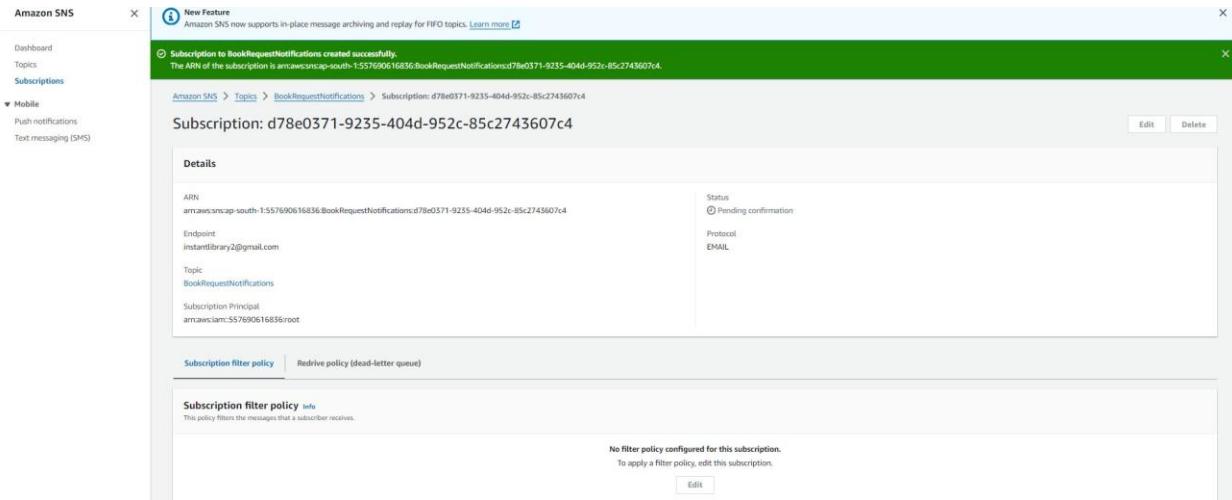
Endpoint:

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

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

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

Cancel Create subscription



The screenshot shows the Amazon SNS Subscription Details page. A green banner at the top indicates a new feature: "Amazon SNS now supports in-place message archiving and replay for FIFO topics." Below this, a success message says "Subscription to BookRequestNotifications created successfully." The ARN of the subscription is listed as arn:aws:sns:ap-south-1:557690616836:BookRequestNotifications:d78e0371-9235-404d-952c-85c2743607c4. The main section shows the subscription details: ARN, Endpoint (instantlibrary2@gmail.com), Topic (BookRequestNotifications), and Subscription Principal (arn:aws:iam::557690616836:root). The status is "Pending confirmation" and the protocol is "EMAIL". There is a "Details" tab and a "Subscription filter policy" tab. The "Subscription filter policy" tab notes "No filter policy configured for this subscription." and "To apply a filter policy, edit this subscription." with an "Edit" button.

- After subscription request for the mail confirmation
- 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 ▾

9

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

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

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

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

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

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

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



Simple Notification Service

Subscription confirmed!

You have successfully subscribed.

Your subscription's id is:

arn:aws:sns:ap-south-1:557690616836:BookRequestNotifications:d78e0371-9235-404d-952c-85c2743607c4

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

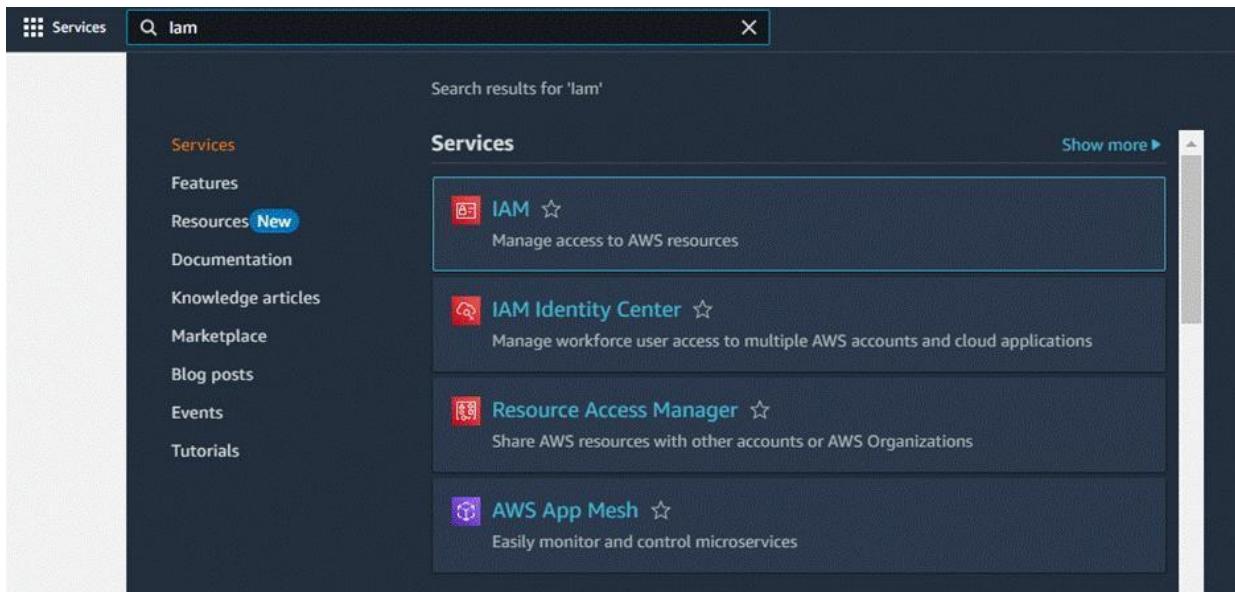
- Successfully done with the SNS mail subscription and setup, now store the ARN link.

Milestone 5 : IAM Role Setup

IAM (Identity and Access Management) role setup involves creating roles that define specific permissions for AWS services. To set it up, you create a role with the required policies, assign it to users or services, and ensure the role has appropriate access to resources like EC2, S3, or RDS. This allows controlled access and ensures security best practices in managing AWS resources.

Create IAM Role.

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



Search results for 'Iam'

Services

Show more ▾

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



Identity and Access Management (IAM)

IAM > Roles

Roles (6) info

An IAM role is an identity you can create that has specific permissions with credentials that are valid for short durations. Roles can be assumed by entities that you trust.

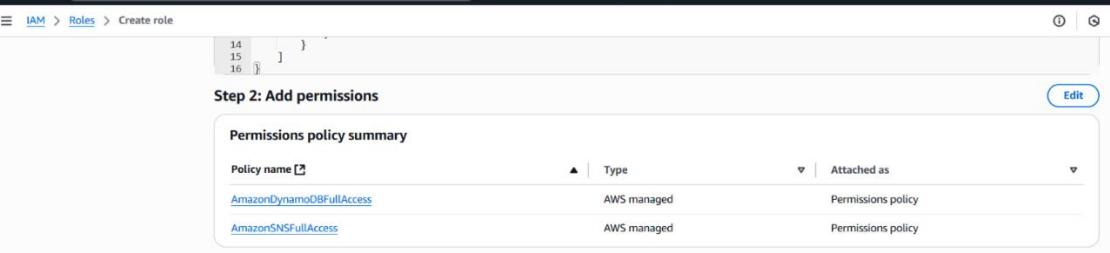
Role name	Type	Attached as
AmazonDynamoDBFullAccess	AWS managed	Permissions policy
AmazonSNSFullAccess	AWS managed	Permissions policy
EC2-Sample-Role	Customer role	Customer role
EC2-Sample-Role-1	Customer role	Customer role
EC2-Sample-Role-2	Customer role	Customer role

Attach

Policies

Attach the following policies to the role:

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



Step 2: Add permissions

Permissions policy summary

Policy name	Type	Attached as
AmazonDynamoDBFullAccess	AWS managed	Permissions policy
AmazonSNSFullAccess	AWS managed	Permissions policy

Step 3: Add tags

Add tags - optional Info

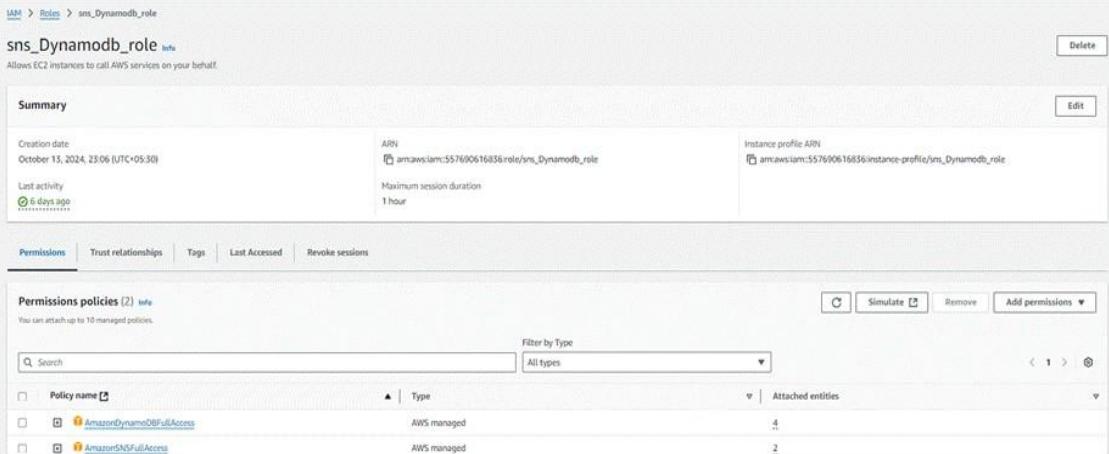
Tags are key-value pairs that you can add to AWS resources to help identify, organize, or search for resources.

No tags associated with the resource.

Add new tag

You can add up to 50 more tags.

Create role



Summary

Creation date: October 13, 2024, 23:06 (UTC+05:30)
Last activity: 6 days ago

ARN: arn:aws:iam::55760616836:role/smr_Dynamodb_role
Instance profile ARN: arn:aws:iam::55760616836:instance-profile/smrt_Dynamodb_role

Permissions

Permissions policies (2) info

You can attach up to 10 managed policies.

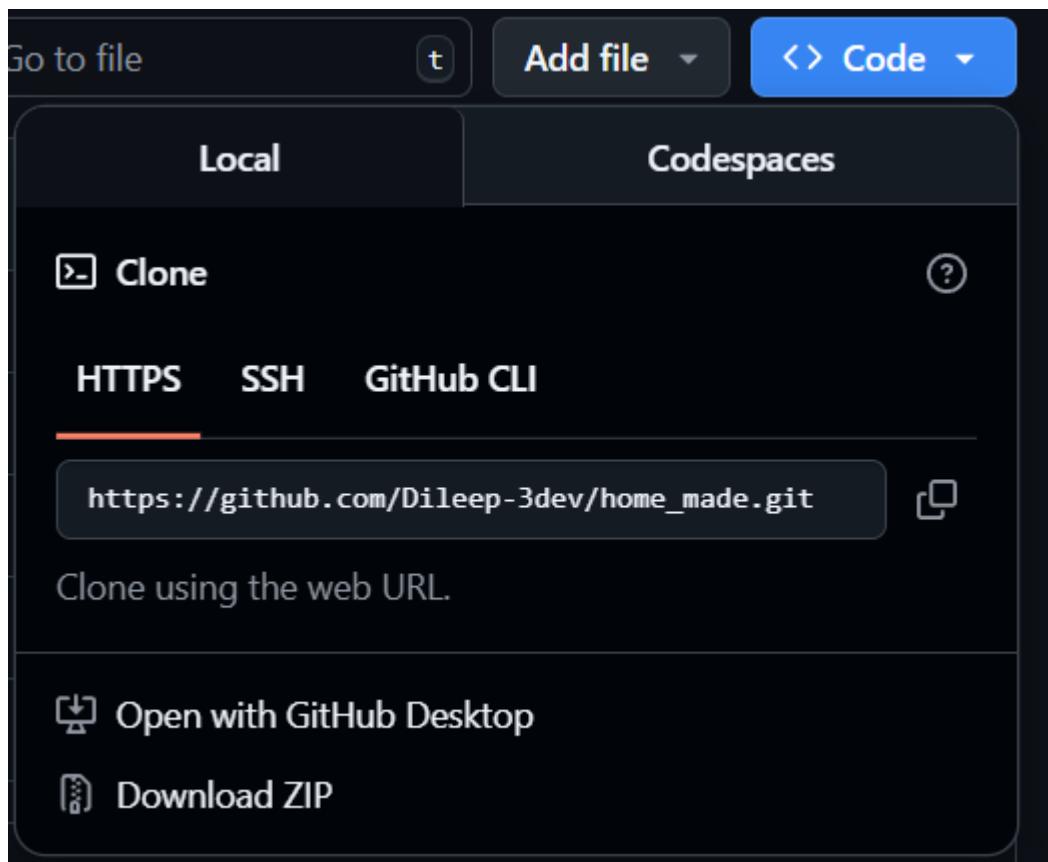
Policy name	Type	Attached entities
AmazonDynamoDBFullAccess	AWS managed	1
AmazonSNSFullAccess	AWS managed	2

Milestone 6 : EC2 Instance Setup

To set up a public EC2 instance, choose an appropriate Amazon Machine Image (AMI) and instance type. Ensure the security group allows inbound traffic on necessary ports (e.g., HTTP/HTTPS for web applications). After launching the instance, associate it with an Elastic IP for consistent public access, and configure your application or services to be publicly accessible.

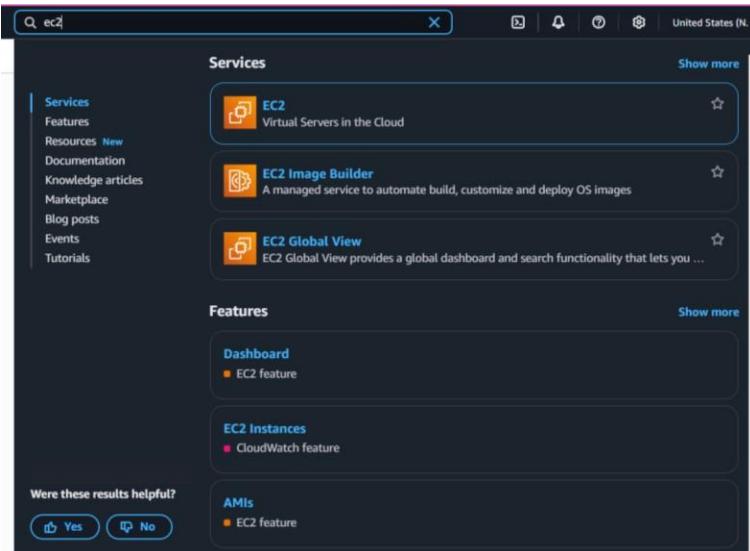
- Note: Load your Flask app and Html files into GitHub repository.

 static	Initial Commit
 templates	Initial Commit
 app.py	Update app.py



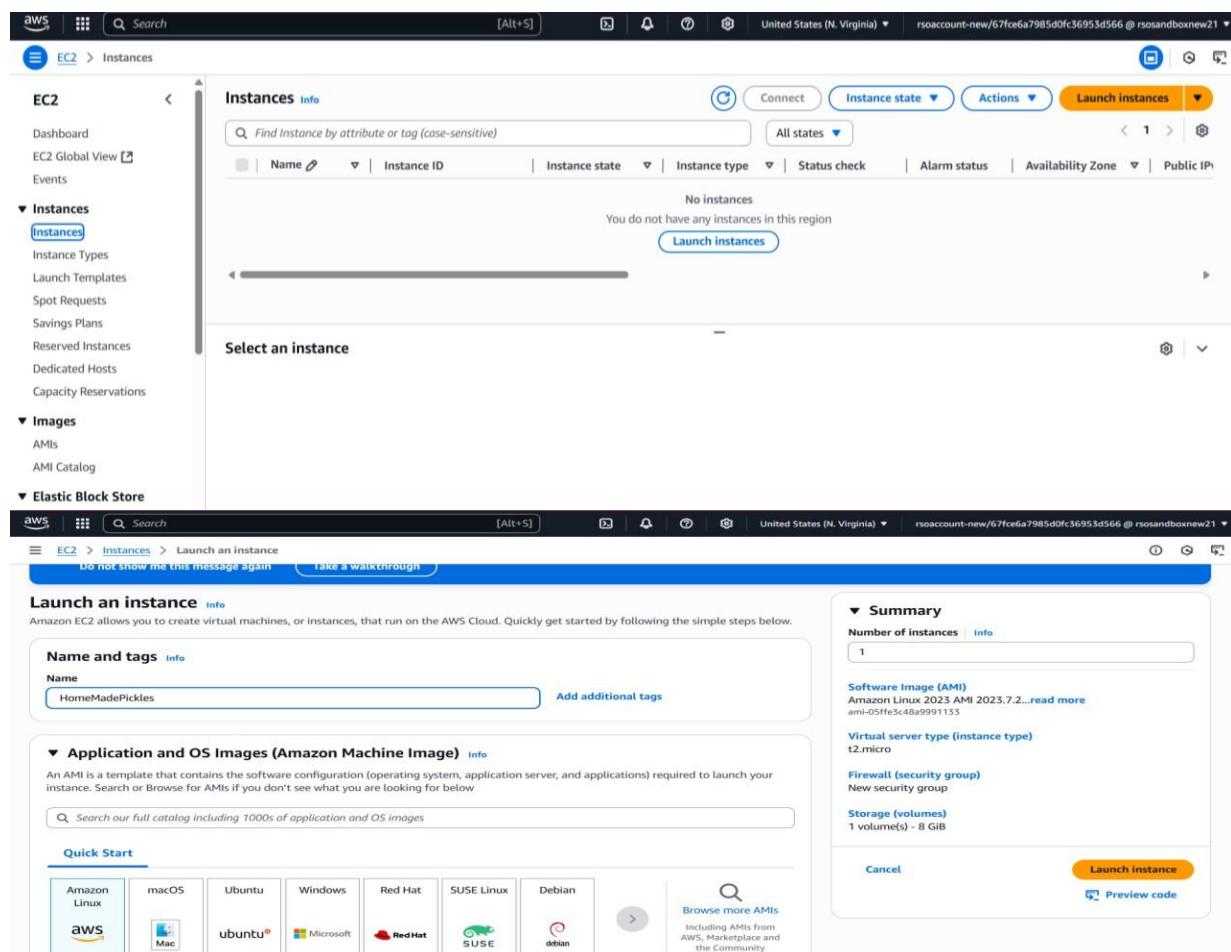
Launch an EC2 instance to host the Flask

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



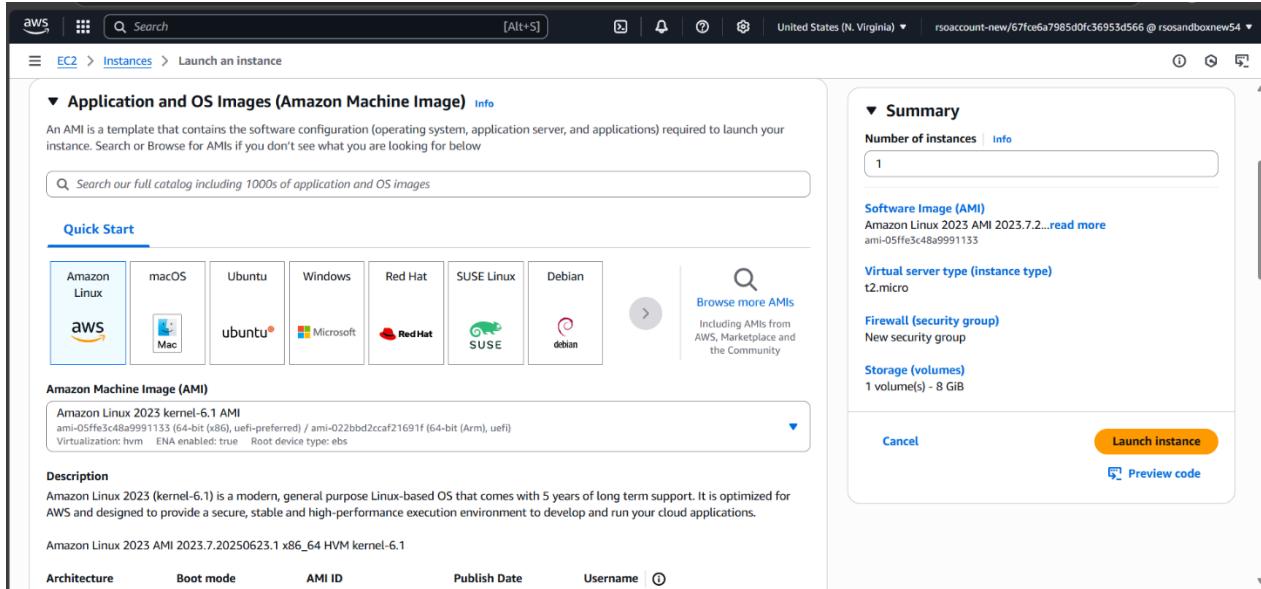
The screenshot shows the AWS search results for 'ec2'. The top result is the EC2 service card, which includes a thumbnail icon, the service name, and a brief description: 'Virtual Servers in the Cloud'. Below the service cards, there are sections for 'Features' and 'AMIs'. The 'Features' section contains cards for 'Dashboard' (an EC2 feature) and 'EC2 Instances' (a CloudWatch feature). The 'AMIs' section contains a card for 'AMIs' (an EC2 feature). At the bottom left, there are 'Yes' and 'No' buttons for a feedback survey.

- Click on Launch instance to launch EC2 instance



The screenshots show the AWS EC2 Instances page. The first screenshot shows the main 'Instances' list with a 'Launch instances' button highlighted. The second screenshot shows the 'Select an instance' step, which is currently empty. The third screenshot shows the 'Launch an instance' configuration wizard. It includes fields for 'Name' (set to 'HomeMadePickles'), 'Software Image (AMI)' (set to 'Amazon Linux 2023 AMI 2023.7.2...'), 'Virtual server type (instance type)' (set to 't2.micro'), and a 'Launch instance' button. The configuration wizard also includes sections for 'Name and tags', 'Application and OS Images (Amazon Machine Image)', 'Quick Start' (with options for various operating systems), and a 'Summary' section.

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



Application and OS Images (Amazon Machine Image) [Info](#)

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below.

[Search our full catalog including 1000s of application and OS images](#)

Quick Start

Amazon Linux	macOS	Ubuntu	Windows	Red Hat	SUSE Linux	Debian

Amazon Machine Image (AMI)

Amazon Linux 2023 kernel-6.1 AMI
 ami-05ffe3c48a9991133 (64-bit (x86), uefi-preferred) / ami-022bbd2ccaf21691f (64-bit (Arm), uefi)
 Virtualization: hvm ENA enabled: true Root device type: ebs

Description

Amazon Linux 2023 (kernel-6.1) 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.

Amazon Linux 2023 AMI 2023.7.20250623.1 x86_64 HVM kernel-6.1

Architecture **Boot mode** **AMI ID** **Publish Date** **Username** [\(i\)](#)

Summary

Number of instances [Info](#)
 1

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

Virtual server type (instance type)
 t2.micro

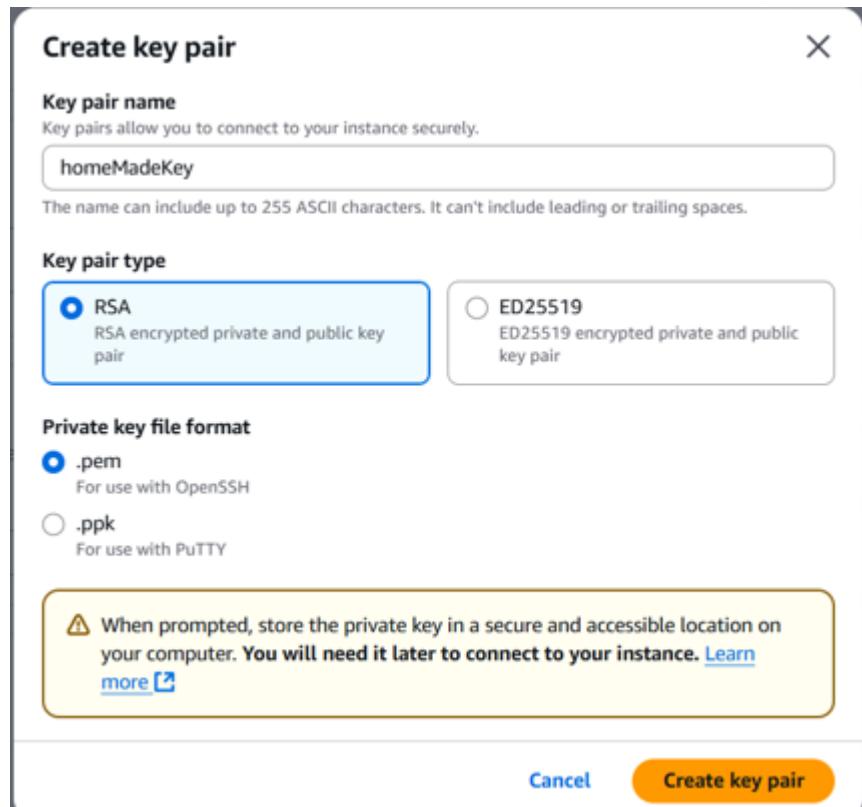
Firewall (security group)
 New security group

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

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

- Create and download the key pair for Server access.

-



Create key pair

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

The name can include up to 255 ASCII characters. It can't include leading or trailing spaces.

Key pair type

RSA
 RSA encrypted private and public key pair

ED25519
 ED25519 encrypted private and public key pair

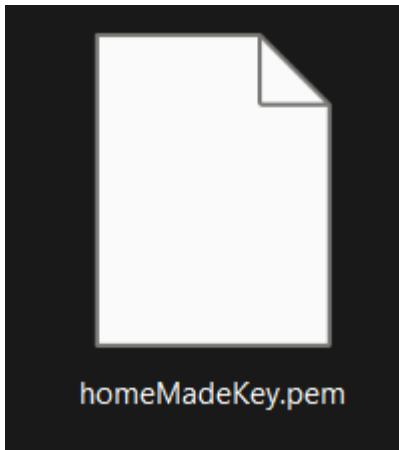
Private key file format

.pem
 For use with OpenSSH

.ppk
 For use with PuTTY

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

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



← ⌂ https://us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#LaunchInstances:

aws | Search [Alt+S] United States (N. Virginia) rsaccount-new/67fce6a7985d0fc36953d566 @ rsosandboxnew54 ▾

EC2 > Instances Launch an instance

Architecture: 64-bit (x86) Boot mode: uefi-preferred AMI ID: ami-05ffe3c48a9991133 Publish Date: 2025-06-20 Username: ec2-user Verified provider

Instance type: t2.micro Family: t2 1 vCPU 1 GiB Memory Current generation: true All generations Compare instance types

Key pair (login): homeMadeKey Create new key pair

Network settings: Network: vpc-044374f9e806e72d9 Edit

Summary: Number of instances: 1 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

Launch instance Preview code

Configure security groups for HTTP, and SSH access.

Inbound Security Group Rules

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

Type: ssh	Protocol: TCP	Port range: 22	Remove
Source type: Anywhere	Description - optional: e.g. SSH for admin desktop		
0.0.0.0/0 X			

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

Type: HTTP	Protocol: TCP	Port range: 80	Remove
Source type: Custom	Description - optional: e.g. SSH for admin desktop		
0.0.0.0/0 X			

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

Type: Custom TCP	Protocol: TCP	Port range: 5000	Remove
Source type: Custom	Description - optional: e.g. SSH for admin desktop		
0.0.0.0/0 X			

Add security group rule

EC2 > Launch an Instance

Success
Successfully initiated launch of instance i-00186102fbac290

Launch log

Next Steps

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	Connect to your instance	Connect an RDS database	Create EBS snapshot policy	Manage detailed monitoring	Create Load Balancer
Once your instance is running, log into it from your local computer.	Configure the connection between an EC2 instance and a database to allow traffic flow between them.	Create a policy that automates the creation, retention, and deletion of EBS snapshots.	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.	Creates a application network gateway or classic Elastic Load Balancer	
Create billing alerts	Connect to instance	Connect an RDS database	Create EBS snapshot policy	Manage detailed monitoring	Create Load Balancer
Learn more	Learn more	Learn more	Learn more	Learn more	Learn more

Create AWS budget	Manage CloudWatch alarms	Disaster recovery for your instances	Monitor for suspicious runtime activities	Get instance screenshot	Get system log
AWS Budgets allows you to create budgets, forecast spend, and take action on your costs and usage from a single location.	Create or update Amazon CloudWatch alarms for the instance.	Recover the instances you just launched into a different Availability Zone or a different Region using AWS Elastic Disaster Recovery (EDR).	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.	Capture a screenshot from the instance and view it as an image. This is useful for troubleshooting an unresponsive instance.	View the instance's system log to troubleshoot issues.
Create AWS budget	Manage CloudWatch alarms	Disaster recovery for your instances	Monitor for suspicious runtime activities	Get instance screenshot	Get system log
Learn more	Learn more	Learn more	Learn more	Learn more	Learn more

[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.

https://us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#Instances:sort=instanceState

aws | Search [Alt+S] United States (N. Virginia) rsoaccount-new/67fce6a7985d0fc36953d566 @ rsosandboxnew54

EC2 > Instances

Success Successfully attached EC2_DynamoDB_Role to instance i-0a759096dd59bc99d

Instances (1/2) [Info](#)

Find Instance by attribute or tag (case-sensitive)

All states ▾

Connect	Instance state	Actions	Launch instances
Connect	Running	Actions	Launch instances
	Terminated		

Name Instance ID Instance state Instance type Status check Alarm status Availability Zone

| HomeMadePickles i-0a759096dd59bc99d Running t2.micro 2/2 checks passed View alarms + us-east-1d |
| HomeDeleted i-0d8e6b4b7c470bf03 Terminated - View alarms + us-east-1d |

i-0a759096dd59bc99d (HomeMadePickles)

EC2 > Instances > i-001861022fbcac290

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

Updated less than a minute ago

Instance ID	i-001861022fbcac290	Public IPv4 address	-	Private IPv4 addresses	172.31.3.5
IPv6 address	-	Instance state	Stopped	Public IPv4 DNS	-
Hostname type	IP name: ip-172-31-3-5.ap-south-1.compute.internal	Private IP DNS name (IPv4 only)	i-172-31-3-5.ap-south-1.compute.internal	Elastic IP addresses	-
Answer private resource DNS name	IPV4 (A)	Instance type	t2.micro	AWS Compute Optimizer finding	Opt-in to AWS Compute Optimizer for recommendations. Learn more [?]
Auto-assigned IP address	-	VPC ID	vpc-03cdc7b6f19dd7211 [?]	Auto Scaling Group name	-
IAM Role	arn:aws:iam::123456789012:role/EC2_DynamoDB_Role	Subnet ID	subnet-0df9fa3144480cc9a9 [?]	Amazon CloudWatch Metrics	-
IMDSv2	Required	Instance ARN	arn:aws:ec2:ap-south-1:557690616836:instance/i-001861022fbcac290	Amazon CloudWatch Logs	-

[Details](#) | [Status and alarms](#) | [Monitoring](#) | [Security](#) | [Networking](#) | [Storage](#) | [Tags](#)

EC2 > Instances > i-001861022fbcac290

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

Updated less than a minute ago

Instance ID	i-001861022fbcac290	Public IPv4 address	-	Private IPv4 addresses	172.31.3.5
IPv6 address	-	Instance state	Stopped	Public IPv4 DNS	-
Hostname type	IP name: ip-172-31-3-5.ap-south-1.compute.internal	Private IP DNS name (IPv4 only)	ip-172-31-3-5.ap-south-1.compute.internal	Elastic IP addresses	-
Answer private resource DNS name	IPV4 (A)	Instance type	t2.micro	AWS Compute Optimizer finding	Opt-in to AWS Compute Optimizer for recommendations. Learn more [?]
Auto-assigned IP address	-	VPC ID	vpc-03cdc7b6f19dd7211 [?]	Auto Scaling Group name	-
IAM Role	arn:aws:iam::123456789012:role/EC2_DynamoDB_Role	Subnet ID	subnet-0df9fa3144480cc9a9 [?]	Amazon CloudWatch Metrics	-
IMDSv2	Required	Instance ARN	arn:aws:ec2:ap-south-1:557690616836:instance/i-001861022fbcac290	Amazon CloudWatch Logs	-

[Connect](#) | [Instance state](#) ▾ | [Actions](#) ▾

- [Manage instance state](#)
- [Instance settings](#)
- [Networking](#)
- Security**
- [Image and templates](#)
- [Monitor and troubleshoot](#)

[Modify IAM role](#)

EC2 > Instances > i-0a759096dd59bc99d > Modify IAM role

Modify IAM role [Info](#)

Attach an IAM role to your instance.

Instance ID: i-0a759096dd59bc99d (HomeMadePickles)

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.

EC2_DynamoDB_Role [\[?\]](#) [Create new IAM role](#) [\[?\]](#)

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

[CloudShell](#) [Feedback](#)

© 2025, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

EC2 > Instances > i-0a759096dd59bc99d > Connect to instance

arn:aws:ssm:us-east-1:504925312407:0: because no identity-based policy allows the ssm:DescribeInstanceInformation action

Connect info

Connect to an instance using the browser-based client.

EC2 Instance Connect Session Manager SSH client EC2 serial console

Instance ID
 i-0a759096dd59bc99d (HomeMadePickles)

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
54.92.203.98

IPv6 address

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

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

Cancel **Connect**

[CloudShell](#) [Feedback](#)

© 2025, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

```

A newer release of "Amazon Linux" is available.
Version 2023.6.20241010:
Run "/usr/bin/dnf check-release-update" for full release and version update info
  #          Amazon Linux 2023
  #          https://aws.amazon.com/linux/amazon-linux-2023
  ~
  ~
  ~
  ~
  ~
  ~
  ~
Last login: Tue Oct 15 04:17:59 2024 from 13.233.177.3
[ec2-user@ip-172-31-3-5 ~]$ █

```

```

3.228.99.42 - [03/Jul/2025 07:43:24] "GET /favicon.ico HTTP/1.1" 404 -
ynamoDB error: An error occurred (ResourceNotFoundException) when calling the Scan operation: Requested resource not found
3.228.99.42 - [03/Jul/2025 07:43:26] "GET /api/products HTTP/1.1" 200 -
3.228.99.42 - [03/Jul/2025 07:43:28] "GET /static/img/chicken-pickle.jpg HTTP/1.1" 200 -
3.228.99.42 - [03/Jul/2025 07:43:28] "GET /static/img/fish-pickle.jpg HTTP/1.1" 200 -
3.228.99.42 - [03/Jul/2025 07:43:49] "GET / HTTP/1.1" 200 -
3.228.99.42 - [03/Jul/2025 07:43:49] "GET /static/css/general.css HTTP/1.1" 304 -

```

i-0a759096dd59bc99d (HomeMadePickles)

Public IPs: [54.92.203.98](#) Private IPs: 172.31.28.232

Milestone 7 : Deployment on EC2

Deployment on an EC2 instance involves launching a server, configuring security groups for public access, and uploading your application files. After setting up necessary dependencies and environment variables, start your application and ensure it's running on the correct port. Finally, bind your domain or use the public IP to make the application accessible online.

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

Clone Your Flask Project from GitHub

Run: 'git clone https://github.com/Dileep-3dev/home_made.git'

- This will download your project to the EC2 instance.

Here : 'git clone https://github.com/Dileep-3dev/home_made.git'

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=5000
```

- This will download your project to the EC2 instance.

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

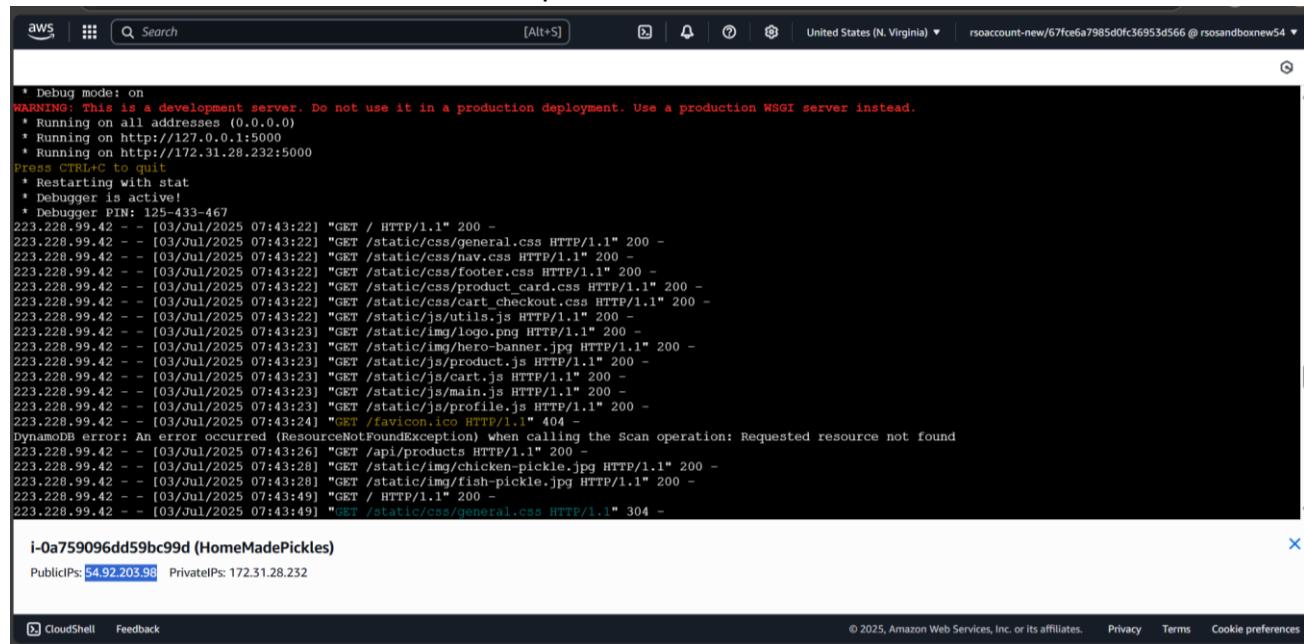
- cd Homemadepicklesandsnacks
- cd "Home Made Pickles1"

Create a Virtual Environment:

- python3 -m venv
- source venv/bin/activate
- sudo yum install python3 git
- sudo pip3 install flask boto3

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=5000



```
* Debug mode: on
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://172.31.28.232:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 125-433-467
223.228.99.42 - - [03/Jul/2025 07:43:22] "GET / HTTP/1.1" 200 -
223.228.99.42 - - [03/Jul/2025 07:43:22] "GET /static/css/general.css HTTP/1.1" 200 -
223.228.99.42 - - [03/Jul/2025 07:43:22] "GET /static/css/nav.css HTTP/1.1" 200 -
223.228.99.42 - - [03/Jul/2025 07:43:22] "GET /static/css/footer.css HTTP/1.1" 200 -
223.228.99.42 - - [03/Jul/2025 07:43:22] "GET /static/css/product_card.css HTTP/1.1" 200 -
223.228.99.42 - - [03/Jul/2025 07:43:22] "GET /static/css/cart_checkout.css HTTP/1.1" 200 -
223.228.99.42 - - [03/Jul/2025 07:43:22] "GET /static/js/utils.js HTTP/1.1" 200 -
223.228.99.42 - - [03/Jul/2025 07:43:23] "GET /static/img/logo.png HTTP/1.1" 200 -
223.228.99.42 - - [03/Jul/2025 07:43:23] "GET /static/img/hero-banner.jpg HTTP/1.1" 200 -
223.228.99.42 - - [03/Jul/2025 07:43:23] "GET /static/js/product.js HTTP/1.1" 200 -
223.228.99.42 - - [03/Jul/2025 07:43:23] "GET /static/js/cart.js HTTP/1.1" 200 -
223.228.99.42 - - [03/Jul/2025 07:43:23] "GET /static/js/main.js HTTP/1.1" 200 -
223.228.99.42 - - [03/Jul/2025 07:43:23] "GET /static/js/profile.js HTTP/1.1" 200 -
223.228.99.42 - - [03/Jul/2025 07:43:24] "GET /favicon.ico HTTP/1.1" 404 -
DynamoDB error: An error occurred (ResourceNotFoundException) when calling the Scan operation: Requested resource not found
223.228.99.42 - - [03/Jul/2025 07:43:26] "GET /api/products HTTP/1.1" 200 -
223.228.99.42 - - [03/Jul/2025 07:43:28] "GET /static/img/chicken-pickle.jpg HTTP/1.1" 200 -
223.228.99.42 - - [03/Jul/2025 07:43:28] "GET /static/img/fish-pickle.jpg HTTP/1.1" 200 -
223.228.99.42 - - [03/Jul/2025 07:43:49] "GET / HTTP/1.1" 200 -
223.228.99.42 - - [03/Jul/2025 07:43:49] "GET /static/css/general.css HTTP/1.1" 304 -

```

i-0a759096dd59bc99d (HomeMadePickles)
 PublicIPs: 54.92.203.98 PrivateIPs: 172.31.28.232

Verify the Flask app is running:

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

- Run the Flask app on the EC2 instance

```

ModuleNotFoundError: No module named 'boto3'

[ec2-user@ip-172-31-15-149 Home made pickles]$ sudo pip3 install boto3
Collecting boto3
  Downloading boto3-1.37.23-py3-none-any.whl (139 kB)
    ━━━━━━━━━━| 139 kB 8.1 MB/s
Collecting s3transfer<0.12.0,>=0.11.0
  Downloading s3transfer-0.11.4-py3-none-any.whl (84 kB)
    ━━━━━| 84 kB 6.7 MB/s
Requirement already satisfied: jmespath<2.0.0,>=0.7.1 in /usr/lib/python3.9/site-packages (from boto3) (0.10.0)
Collecting botocore<1.38.0,>=1.37.23
  Downloading botocore-1.37.23-py3-none-any.whl (13.4 MB)
    ━━━━━| 13.4 MB 38.3 MB/s
Requirement already satisfied: urllib3<1.27,>=1.25.4 in /usr/lib/python3.9/site-packages (from botocore<1.38.0,>=1.37.23->boto3) (1.25.10)
Requirement already satisfied: python-dateutil<3.0.0,>=2.1 in /usr/lib/python3.9/site-packages (from botocore<1.38.0,>=1.37.23->boto3) (2.8.1)
Requirement already satisfied: six<1.5 in /usr/lib/python3.9/site-packages (from python-dateutil<3.0.0,>=2.1->botocore<1.38.0,>=1.37.23->boto3) (1.15.0)
Installing collected packages: botocore, s3transfer, boto3
Successfully installed boto3-1.37.23 botocore-1.37.23 s3transfer-0.11.4
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system package manager. It is recommended to use a virtual environment instead: https://pip.pypa.io/warningsenv
[ec2-user@ip-172-31-15-149 Home made pickles]$ sudo flask run --host=0.0.0.0 --port=5000
 * 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://127.0.0.1:5000
 * Running on http://172.31.15.149:5000
Press CTRL+C to quit

```

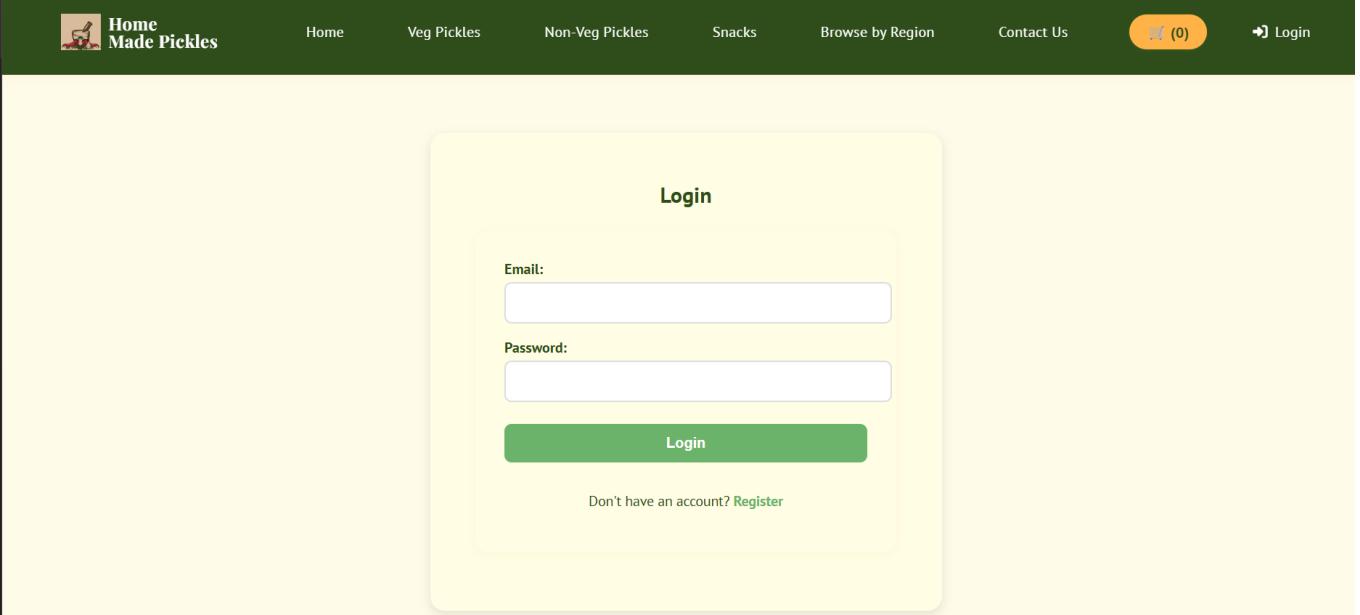
Acess the website through:

PublicIPs: <http://54.92.203.98:5000>

Milestone 8 : Testing and Deployment

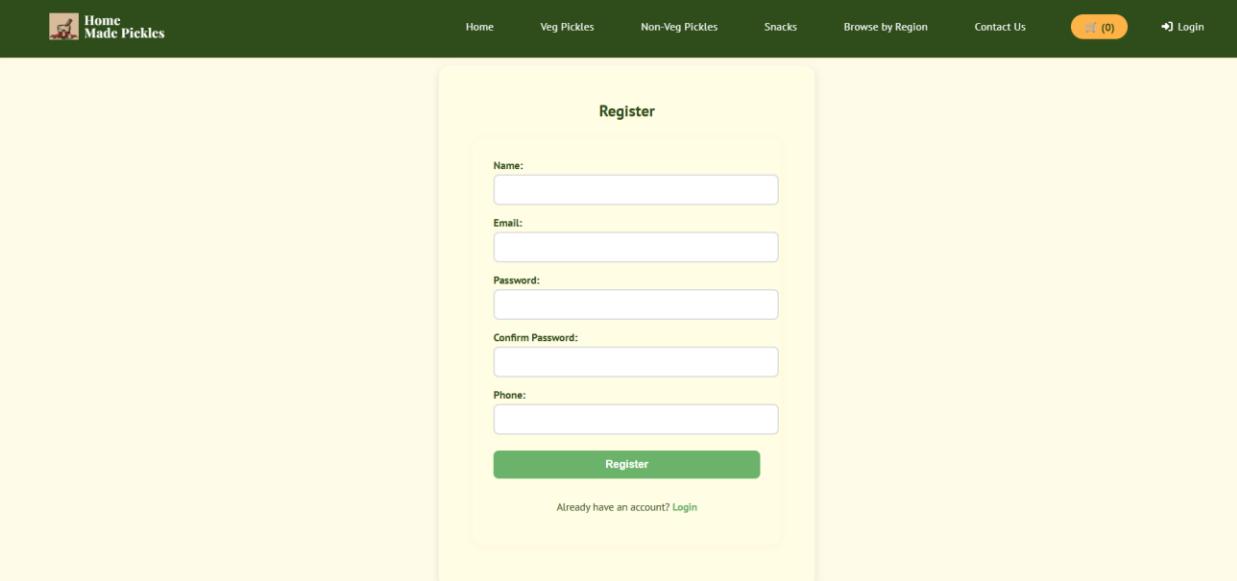
Testing and deployment involve verifying that your application works as expected before making it publicly accessible. Start by testing locally or on a staging environment to catch bugs and ensure functionality. Once tested, deploy the application to an EC2 instance, configure necessary services, and perform a final round of live testing to confirm everything runs smoothly in the production environment.

Login Page:



The screenshot shows the homepage of the "Home Made Pickles" website. At the top, there is a navigation bar with links for "Home", "Veg Pickles", "Non-Veg Pickles", "Snacks", "Browse by Region", "Contact Us", a shopping cart icon with "(0)", and a "Login" button. Below the navigation bar, there is a large, semi-transparent overlay containing a login form. The form has fields for "Email:" and "Password:", both represented by input boxes. A green "Login" button is centered below the password field. At the bottom of the form, there is a link "Don't have an account? Register".

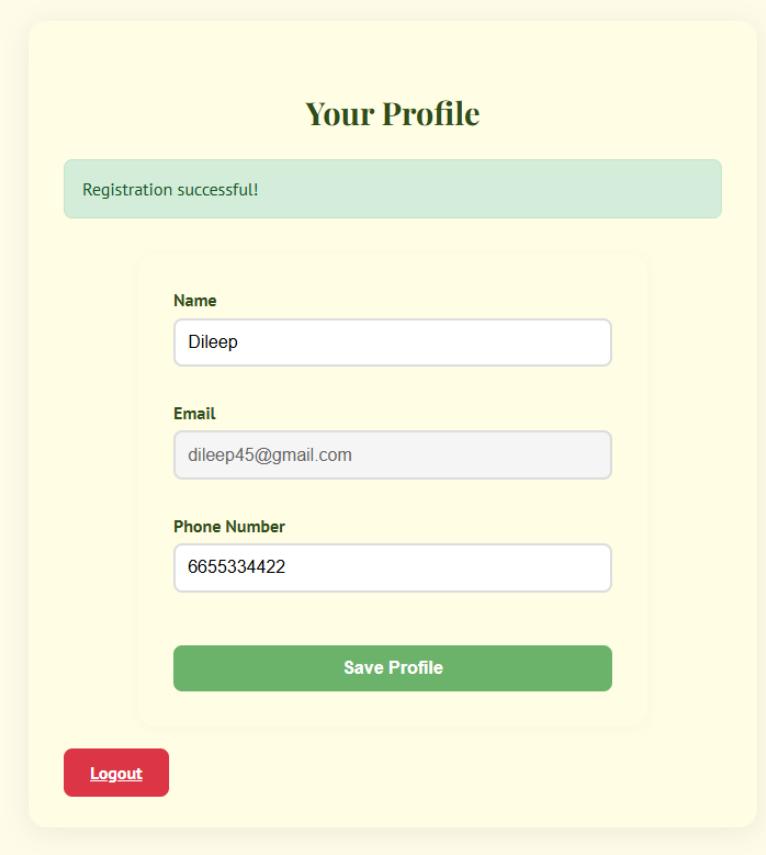
Register Page:



The screenshot shows the registration page for "Home Made Pickles". The header includes the SmartBridge logo and navigation links for Home, Veg Pickles, Non-Veg Pickles, Snacks, Browse by Region, Contact Us, a notification badge (0), and a Login button.

The main content area is titled "Register" and contains fields for Name, Email, Password, Confirm Password, and Phone, each with a corresponding input field. A green "Register" button is at the bottom, and a link to "Login" is below it.

Profile page :



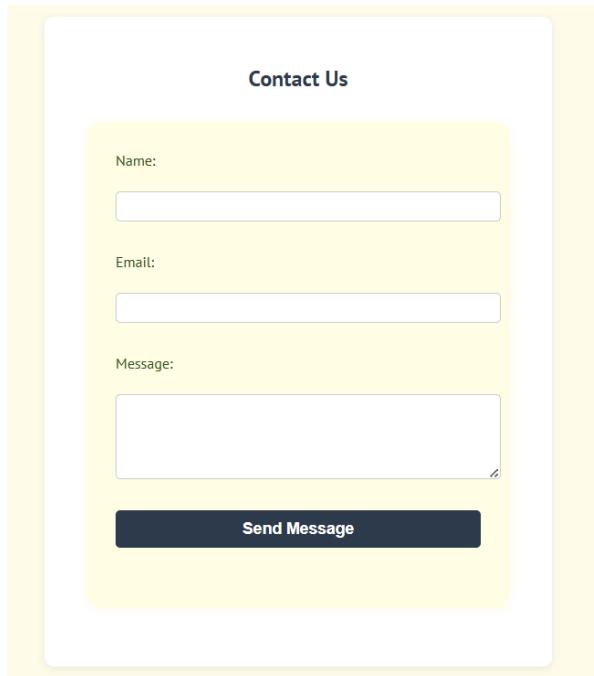
The screenshot shows the user profile page. At the top, a green banner displays the message "Registration successful!". The page features three input fields for Name ("Dileep"), Email ("dileep45@gmail.com"), and Phone Number ("6655334422"). Below these fields is a green "Save Profile" button. At the bottom left is a red "Logout" button.

Home page:



The screenshot shows the homepage of a website for "Home Made Pickles". The header features the company logo and name. Below the header, there is a main banner with the text "Home Made Pickles & Snacks - Taste the Best" and a subtext "Preserving Traditions, One Jar at a Time.". Two buttons, "Explore Homemade Goods" and "Our Specials", are visible. To the right of the text is a small illustration of a jar and some pickles. Below the banner, there is a section titled "Why Choose Us" with three icons: a yellow circle with a green leaf, a green circle with a white swirl, and a dark blue circle with a white gear.

Contact Us page:



The screenshot shows the "Contact Us" page of the website. The page has a light yellow background. At the top, it says "Contact Us". Below that are three input fields: "Name:", "Email:", and "Message:". Each field has a corresponding empty input box. At the bottom of the form is a dark blue button labeled "Send Message".

Artisan Handbook Page:



Artisan Handbook

Empowering home-based pickle & snack makers to grow, thrive, and shine!

A guide for home-based pickle and snack makers. Learn best practices, packaging, hygiene, and how to grow your artisan business online.

Best Practices

Maintain quality by using fresh, local ingredients. Keep your workspace clean and follow traditional recipes for authentic taste. Consistency and care in every batch will set you apart!

Packaging & Hygiene

Use food-safe, eco-friendly packaging. Label your products clearly and store them in a cool, dry place. Always wear gloves and hairnets while preparing food. Good hygiene builds trust and loyalty.

Growing Your Business

Promote your products online, engage with customers, and consider joining local food fairs. Share your story, use social media, and build a loyal customer base. Never stop learning and innovating!

Seller forum page:

Seller Forum

Connect with other sellers, ask questions, share tips, and grow your network in our supportive artisan community.

Post title or question

Share your question, tip, or experience...

Post

No posts yet. Be the first to ask a question or share a tip!

Veg Pickles Page :

Veg Pickles

Search products...

Veg Pickles



Traditional Mango Pickle

Classic vegetarian pickles made with fresh ingredients.

Weight: 250g - Rs.150

Add to Cart



Zesty Lemon Pickle

Classic vegetarian pickles made with fresh ingredients.

Weight: 250g - Rs.120

Add to Cart



Tomato Pickle

Classic vegetarian pickles made with fresh ingredients.

Weight: 250g - Rs.130

Add to Cart



Non-Veg Pickles page:

Non-Veg Pickles

Search products...

Non-Veg Pickles



Chicken Pickle

Authentic homemade non-veg pickles, rich in flavor and tradition.

Weight: 250g - Rs.600

Add to Cart



Fish Pickle

Authentic homemade non-veg pickles, rich in flavor and tradition.

Weight: 250g - Rs.200

Add to Cart



Gongura Mutton

Authentic homemade non-veg pickles, rich in flavor and tradition.

Weight: 250g - Rs.400

Add to Cart



Snacks Page:

Snacks

Search products...

Snacks ▾



Banana Chips

Crispy, crunchy, and delicious snacks for every occasion.

Weight: 250g - Rs.300 ▾

Add to Cart



Crispy Aam-Papad

Crispy, crunchy, and delicious snacks for every occasion.

Weight: 250g - Rs.150 ▾

Add to Cart



Sweet Boondhi

Crispy, crunchy, and delicious snacks for every occasion.

Weight: 250g - Rs.300 ▾

Add to Cart



Category Page :

×



How would you like to browse?

Choose your preferred way to discover our homemade pickles and snacks.

 Choose from Map (Regional)

 Browse Full Catalog

Cancel

Regional Page:

① Select Region

[Browse Full Catalogue Instead](#)

- [North India](#)
- [South India](#)
- [East India](#)
- [West India](#)
- [Central India](#)

Product Type

[All](#) [Pickles](#) [Snacks](#)

Products by Region



Chicken Pickle
Authentic homemade non-veg pickles, rich in flavor and tradition.

Weight: 250g - Rs.600

[Add to Cart](#)



Traditional Mango Pickle
Classic vegetarian pickles made with fresh ingredients.

Weight: 250g - Rs.150

[Add to Cart](#)



Products Page :

[All Products](#)

[All Products](#)



Chicken Pickle
Authentic homemade non-veg pickles, rich in flavor and tradition.

Weight: 250g - Rs.600

[Add to Cart](#)



Fish Pickle
Authentic homemade non-veg pickles, rich in flavor and tradition.

Weight: 250g - Rs.200

[Add to Cart](#)



Gongura Mutton
Authentic homemade non-veg pickles, rich in flavor and tradition.

Weight: 250g - Rs.400

[Add to Cart](#)



Mutton Pickle
Authentic homemade non-veg pickles, rich in flavor and tradition.

Weight: 250g - Rs.400

[Add to Cart](#)



Prawns Pickle
Authentic homemade non-veg pickles, rich in flavor and tradition.

Weight: 250g - Rs.600

[Add to Cart](#)



Chicken Pickle (Gongura)
Authentic homemade non-veg pickles, rich in flavor and tradition.

Weight: 250g - Rs.350

[Add to Cart](#)

Cart Page:

Shopping Cart

	Banana Chips 250g	Rs.300	<input type="button" value="-"/>	1	<input type="button" value="+"/>	<input type="button" value="Remove"/>
	Kara Boondi 250g	Rs.250	<input type="button" value="-"/>	1	<input type="button" value="+"/>	<input type="button" value="Remove"/>
	Crispy Aam-Papad 250g	Rs.150	<input type="button" value="-"/>	1	<input type="button" value="+"/>	<input type="button" value="Remove"/>
	Mix Veg Pickle 250g	Rs.130	<input type="button" value="-"/>	1	<input type="button" value="+"/>	<input type="button" value="Remove"/>
Subtotal: Rs.830 Shipping: Free Total: Rs.830						
<input type="button" value="Proceed to Checkout"/>						

Check-Out Page:

Checkout <div style="border: 1px solid #ccc; padding: 10px; margin-bottom: 10px;"> Contact Info <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 10px;">Dileep</div> <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 10px;">dileep45@gmail.com</div> <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 10px;">7253926453</div> <div style="background-color: #f0c21e; color: white; text-align: center; padding: 5px; border-radius: 5px;"><input type="button" value="Next"/></div> </div>	Order Summary <div style="border: 1px solid #ccc; padding: 10px; margin-bottom: 10px;"> Subtotal: Rs.830 Shipping: Free Total: Rs.830 </div>
---	--

Address Page:

📍 Shipping Address

new Road

Tekkali 532201

India

[Back](#) [Next](#)

">\$ Payment Method

Cash on Delivery

[Back](#)

Confirmation Page:

✓ Order Confirmation

Thank you for your order!

Your order ID is #1

Total: **Rs.830**

[Continue Shopping](#) [View Order Details](#)

Order History page:

Order History

Order #	Date	Total	Status	Action
#1	7/7/2025, 7:56:23 PM	Rs.830	Placed	View Details

Exit:

Session Ended

Please close this tab.

Functional testing to verify the Project Conclusion

This platform has been thoughtfully created to connect food enthusiasts with authentic, handcrafted pickles and snacks from local artisans. Using Flask as its foundation, the website provides a seamless shopping experience that makes discovering and ordering traditional recipes simple and enjoyable.

Key Features: The platform offers an intuitive browsing system with categories for vegetarian pickles, non-vegetarian pickles, and traditional snacks. Customers can choose from multiple weight options (250g to 1kg), enjoy secure user accounts, and experience smooth cart management with real-time updates. The regional discovery feature helps users explore local specialties while maintaining a mobile-friendly design that works across all devices.

Supporting Artisans and Heritage: By bringing these time-honored recipes online, the platform bridges the gap between small-scale producers and customers seeking preservative-free, authentic flavours. The cloud-ready architecture ensures reliable service as the community grows, while automated email notifications and efficient order management keep everything running smoothly.

Impact: This project demonstrates how modern technology can preserve and share culinary traditions, making it easier for people to access genuine homemade flavours. As the platform evolves, it continues to serve as a trusted marketplace where every jar represents the care and quality of traditional home cooking.

In essence, this website transforms how authentic pickles and snacks reach customers, ensuring that traditional flavours find their way to modern kitchens with the same love and quality they've always carried.