

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

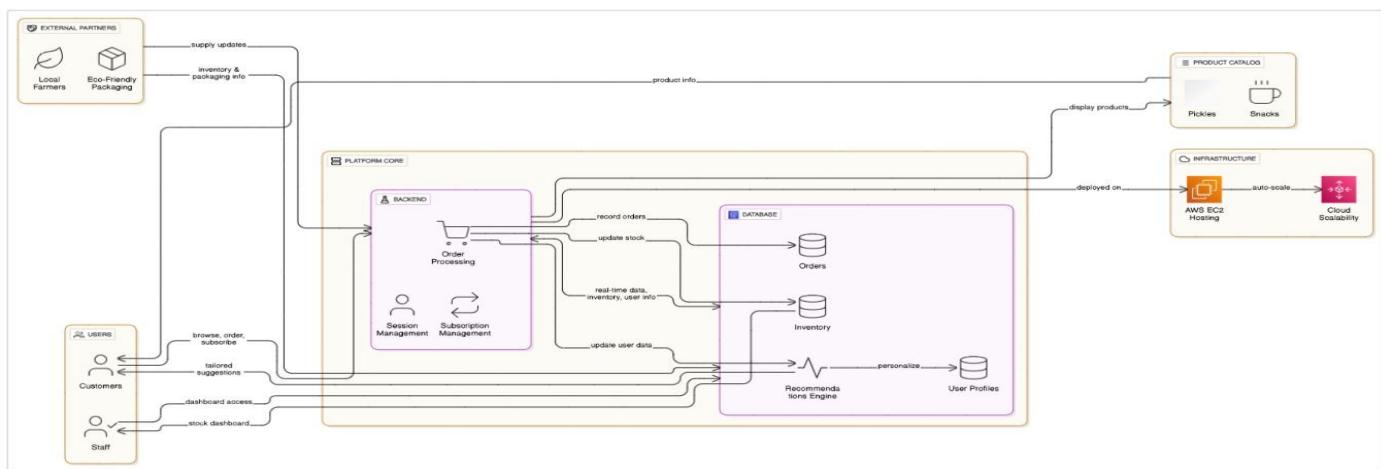
When a customer places an order for a product, the system instantly updates stock levels and records transaction details. For example, a user purchases an item, triggering automatic inventory deduction and order confirmation. Staff members receive updated dashboards to monitor stock availability and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

Scenario 3: Personalized User Experience and Recommendations

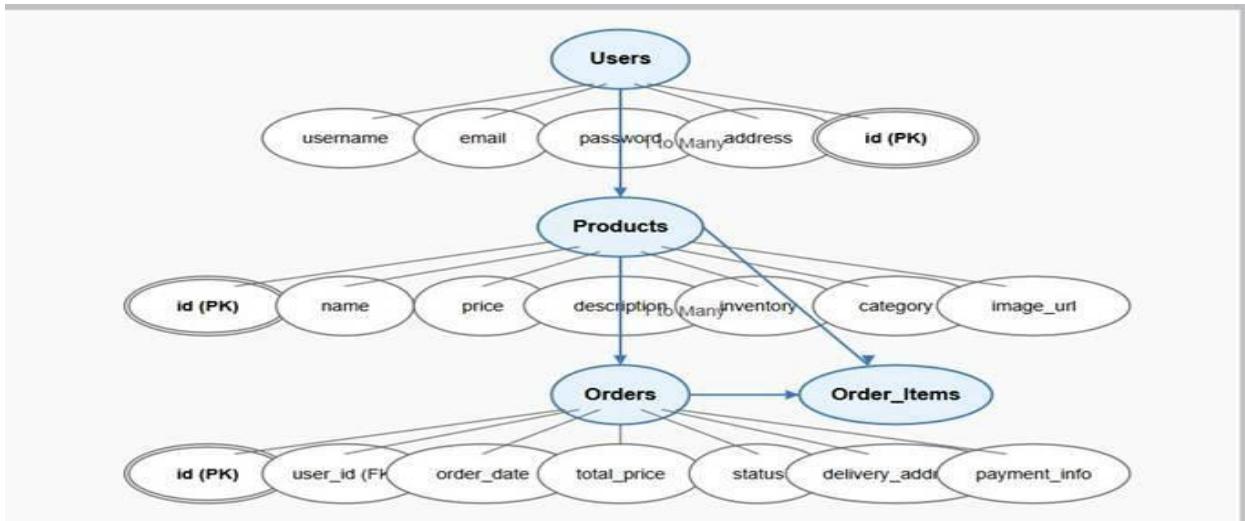
The platform leverages user behavior data to enhance engagement. A returning customer, for instance, views tailored recommendations based on past purchases and browsing history. The system dynamically adjusts suggestions in real-time, while maintaining fast response rates even during high traffic, creating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

AWS ARCHITECTURE

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:



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 Book 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

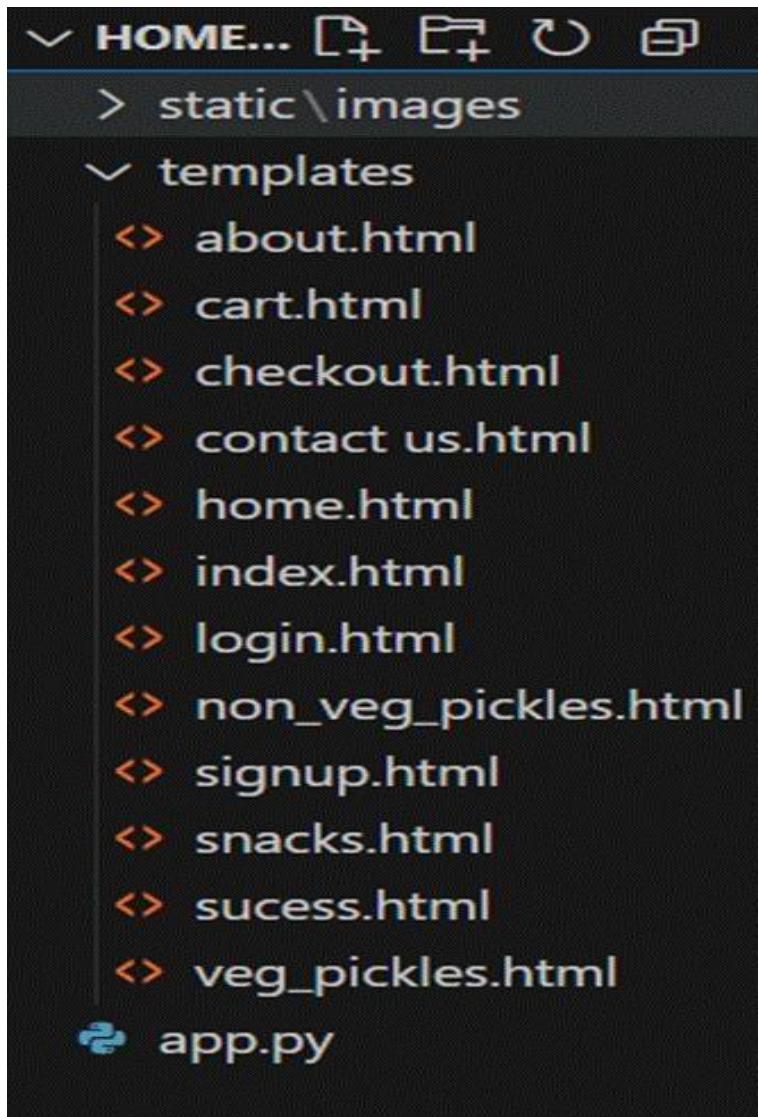
Milestone 8. Testing and Deployment

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

Milestone 1: Backend Development and Application SetUp

- **Activity 1.1:** Develop the Backend Using Flask

1. 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.

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

Description of the code :

? Flask App Initialization

```
app.py > ...
1  from flask import Flask, render_template, request, redirect, url_for, flash, session
2  from werkzeug.security import generate_password_hash, check_password_hash
3  from datetime import datetime
4  import boto3
5  import smtplib
6  import logging
7  import uuid
8  from email.mime.text import MIMEText
9  import os
```

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

```

app.py > ...
39
40 dynamodb = boto3.resource('dynamodb', region_name=AWS_REGION)
41 orders_table = dynamodb.Table('PickleOrders')
42 users_table = dynamodb.Table('users')
43 sns = boto3.client('sns', region_name=AWS_REGION)
44
45 #Email settings (loaded securely from .env)
46 EMAIL_HOST = os.getenv('EMAIL_HOST', 'smtp.gmail.com')
47 EMAIL_PORT = int(os.getenv('EMAIL_PORT', 587))
48 EMAIL_USER = os.getenv('EMAIL_USER')
49 EMAIL_PASSWORD = os.getenv('EMAIL_PASSWORD')
50
51
52 @app.context_processor
53 def inject_theme():
54     return {"color": app.config["THEME_COLOR"], "year": datetime.now().year}
55
56 # ----- Updated Product Inventory -----
57 products = {
58     "mango": {"name": "Mango Pickle", "price": 249, "stock": 10, "image": "mango.jpg"},
59     "lemon": {"name": "Lemon Pickle", "price": 199, "stock": 8, "image": "lemon.jpg"},
60     "gongura": {"name": "Gongura Pickle", "price": 229, "stock": 7, "image": "gongura.jpg"},
61     "chicken": {"name": "Chicken Pickle", "price": 349, "stock": 12, "image": "chicken.jpg"},
62     "fish": {"name": "Fish Pickle", "price": 329, "stock": 9, "image": "fish.jpg"},
63     "prawns": {"name": "Prawns Pickle", "price": 399, "stock": 6, "image": "prawns.jpg"},
64     "murukulu": {"name": "Murukulu", "price": 149, "stock": 20, "image": "murukulu.jpg"},
65     "nippattu": {"name": "Nippattu", "price": 129, "stock": 15, "image": "nippattu.jpg"},
66     "hot_maida_biscuit": {"name": "Hot Maida Biscuit", "price": 109, "stock": 25, "image": "hot_maida_biscuit.jpg"},
67 }
68
69 def get_products(prefix=None):

```

- 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"]
        pwd = request.form["password"]
        user = users.get(email)
        if user and check_password_hash(user["hash"], pwd):
            session["user"] = email
            flash("Logged in!", "success")
            return redirect(url_for("home"))
        flash("Invalid credentials", "danger")
    return render_template("login.html")

@app.route("/logout")
```

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

```
@app.route("/signup", methods=["GET", "POST"])
def signup():
    if request.method == "POST":
        email = request.form["email"]
        pwd = request.form["password"]
        confirm = request.form["confirm"]
        if pwd != confirm:
            flash("Passwords don't match", "warning")
            return redirect(url_for("signup"))
        if email in users:
            flash("User exists, log in", "info")
            return redirect(url_for("login"))
        users[email] = {"hash": generate_password_hash(pwd)}
        flash("Signup complete - please log in", "success")
        return redirect(url_for("login"))
    return render_template("signup.html")
```

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

```
# ----- Routes -----
@app.route("/")
def home():
    best = dict(list(get_products().items())[:6])
    return render_template("index.html", items=best)

@app.route("/veg")
def veg():
    return render_template("veg.html", items={
        k: v for k, v in products.items() if k in ["mango", "lemon", "gongura"]
    })

@app.route("/nonveg")
def nonveg():
    return render_template("nonveg.html", items={
        k: v for k, v in products.items() if k in ["chicken", "fish", "prawns"]
    })

@app.route("/snacks")
def snacks():
    return render_template("snacks.html", items={
        k: v for k, v in products.items() if k in ["murukulu", "nippattu", "hot_maida_biscuit"]
    })

@app.route("/cart")
def cart():
    items, total = {}, 0
    if session.get("cart"):
        for pid, qty in session["cart"].items():
            if pid in products:
                items[pid] = {"product": pid, "price": products[pid]["price"], "quantity": qty}
                total += products[pid]["price"] * qty
    return render_template("cart.html", items=items, total=total)
```

CheckOut

```
app.route("/checkout", methods=["GET", "POST"])
def checkout():
    cart = session.get("cart", {})
    if not cart:
        flash("Cart is empty", "warning")
        return redirect(url_for("home"))

    if request.method == "POST":
        name = request.form["name"]
        email = request.form["email"]
        address = request.form["address"]
        order_id = str(uuid.uuid4())
        order_time = datetime.now().isoformat()

        items = [
            {"product": pid, "price": products[pid]["price"], "quantity": qty}
            for pid, qty in cart.items()
        ]
        total = sum(p["price"] * p["quantity"] for p in items)

    order_data = {
        "order_id": order_id,
        "name": name,
        "email": email,
        "address": address,
        "order_time": order_time,
        "items": items,
        "total": total
    }

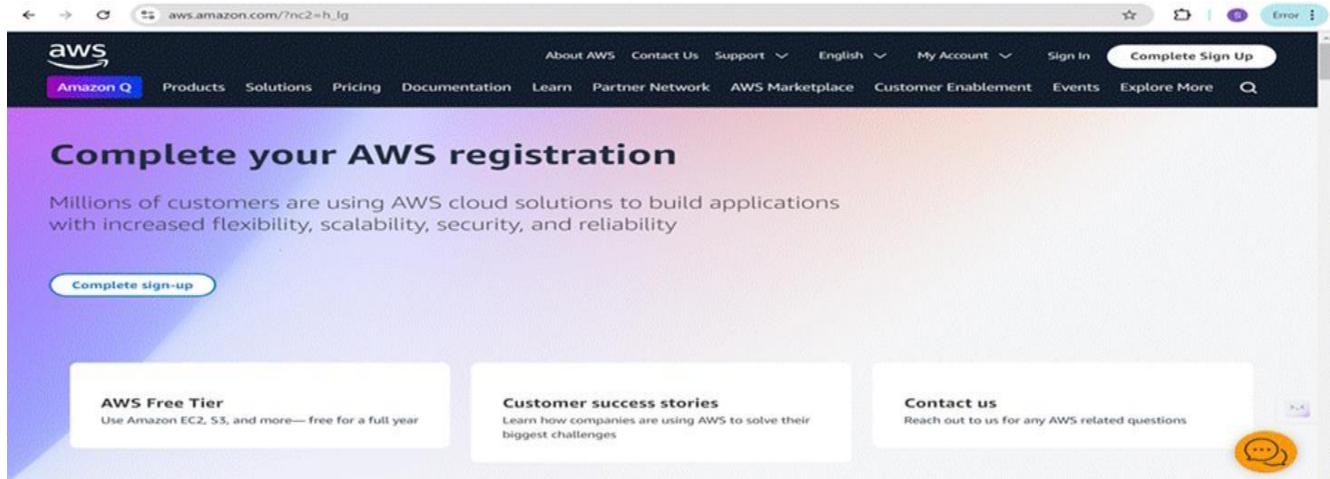
    save_order_to_dynamodb(order_data)

    summary = f"Order ID: {order_id}\nName: {name}\nTotal: ₹{total}\n\nThank you for your order!"
    send_order_email(email, summary)
    send sns notification(f"New order placed: {order_id}")

    session.pop("cart", None)
    flash("Order placed successfully!", "success")
    return redirect(url_for("success"))
return render_template("checkout.html")
```

Milestone 2: AWS Account Setup and Login.

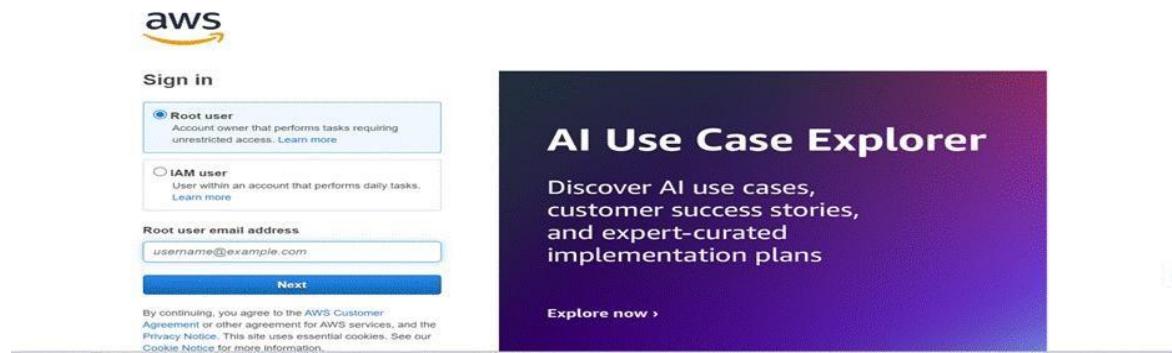
- **Activity 2.1:** Set up an AWS account if not already done.
- Sign up for an AWS account and configure billing settings.



- 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.



- **Activity2.2** : 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

- **Activity 3.1:** Navigate to the DynamoDB ○ In the AWS Console, navigate to DynamoDB and click on create tables.

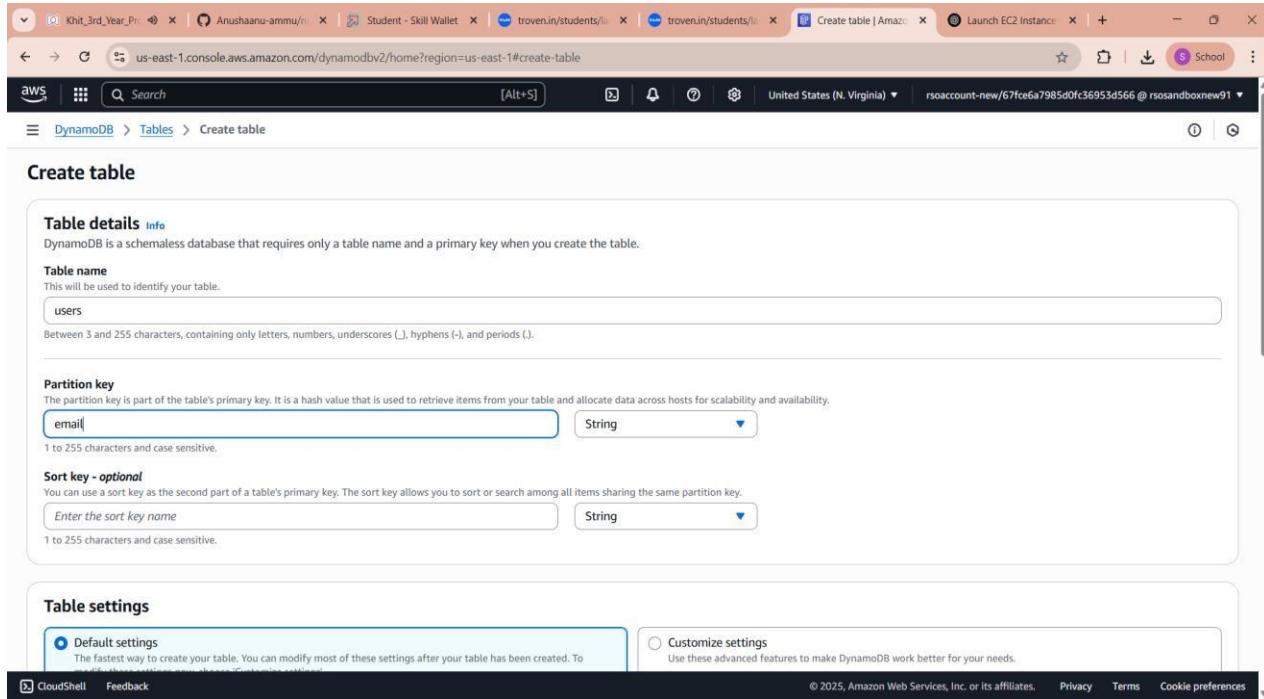
Screenshot of the AWS CloudWatch Metrics Dashboard showing metrics for EC2 instances. The dashboard includes a search bar, a navigation menu, and several visualizations.

The main visualization shows a line chart of CPU Utilization over time, with a red line indicating the average utilization. Below the chart, there are three detailed views of specific EC2 instances: "ec2-44-220-144-100", "ec2-44-220-144-101", and "ec2-44-220-144-102". Each instance view provides details like CPU Utilization, Network In, Network Out, and Status.

Below the main chart, there are two more sections: "CloudWatch Metrics" and "CloudWatch Metrics Insights".

Activity 3.2: Create a DynamoDB table for storing registration details and book requests.

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



The screenshot shows the 'Create table' wizard in the AWS DynamoDB console. The table name is 'users'. The partition key is 'email' of type String. There is no sort key defined. Under 'Table settings', the 'Default settings' option is selected. The page includes standard AWS navigation and footer links.

Table details

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

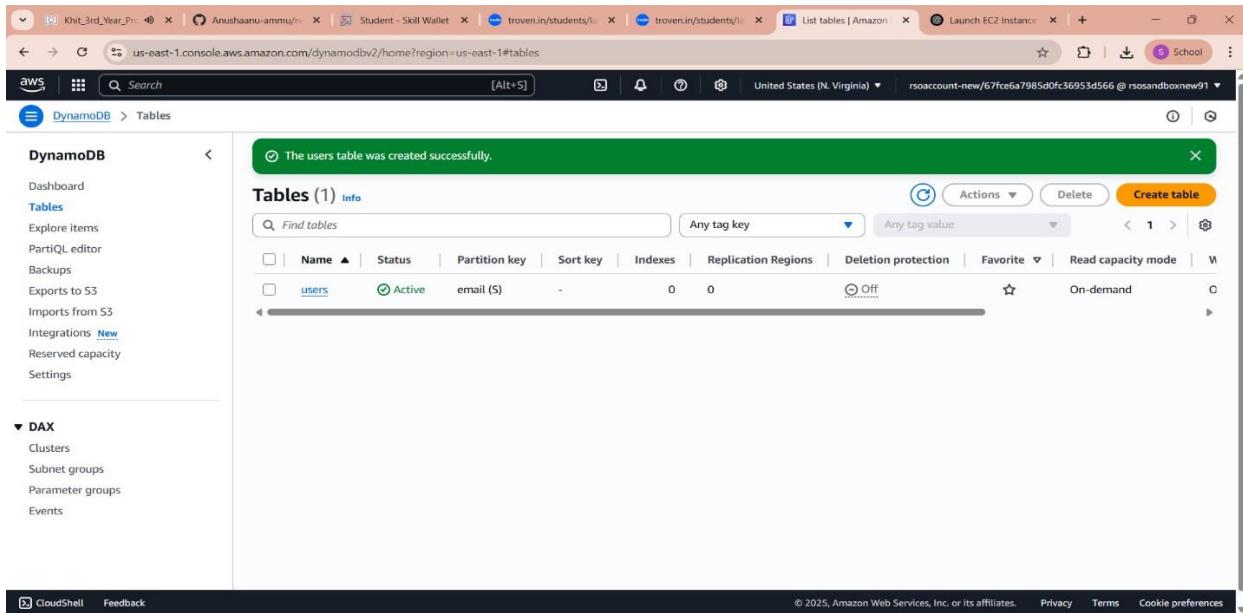
Default settings

The fastest way to create your table. You can modify most of these settings after your table has been created. To learn more about table settings, see Table settings.

Customize settings

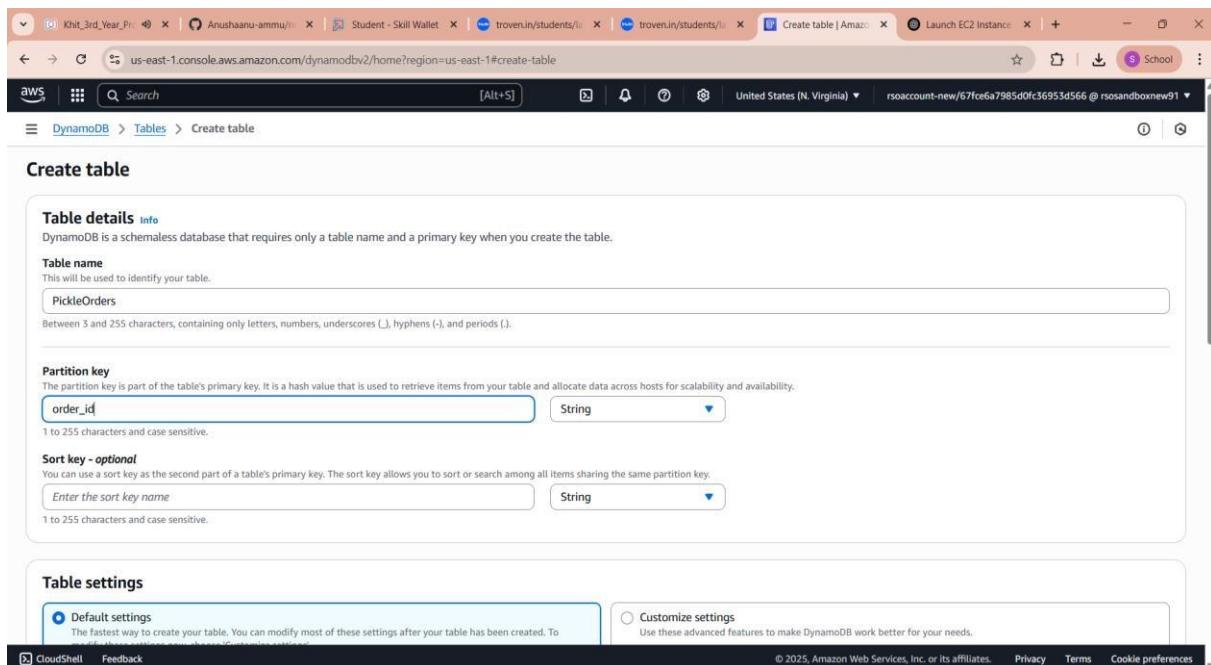
Use these advanced features to make DynamoDB work better for your needs.

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

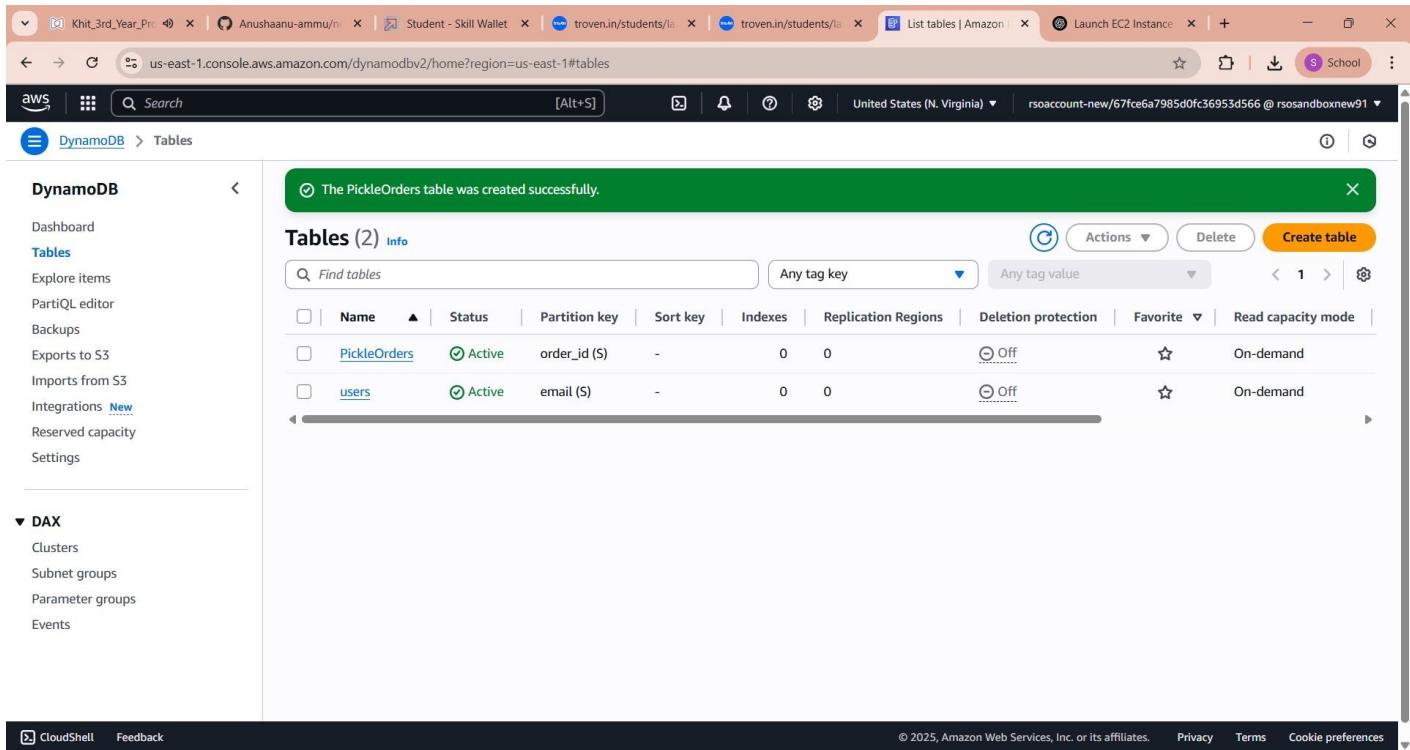


The screenshot shows the AWS DynamoDB console with a success message: "The users table was created successfully." The "Tables" section lists one table named "users" with the status "Active". The table has a partition key "email (S)" and no sort key, indexes, or replication regions. It is set to "Off" for deletion protection and is in "On-demand" read capacity mode.

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



The screenshot shows the "Create table" wizard in the AWS DynamoDB console. In the "Table details" step, a table named "PickleOrders" is being created with a primary key "order_id" of type String. In the "Partition key" step, "order_id" is selected as the partition key. In the "Sort key - optional" step, there is an empty field for a sort key name. In the "Table settings" step, the "Default settings" option is selected, indicating it's the fastest way to create the table.

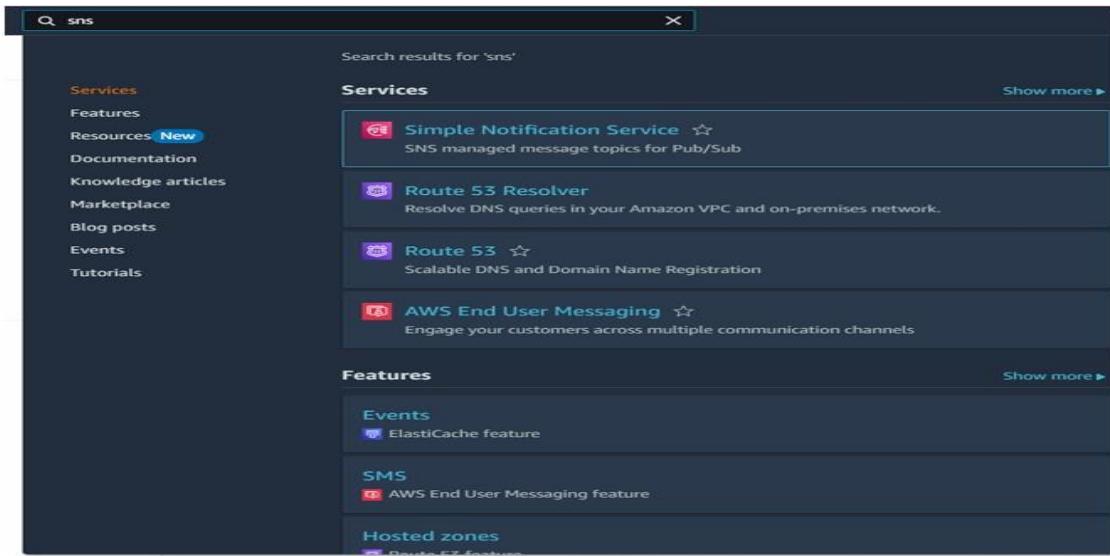


The screenshot shows the AWS DynamoDB console with the following details:

- Tables (2)**: The table list shows two tables:
 - PickleOrders**: Active, Partition key `order_id ($)`, Sort key `-`, 0 indexes, 0 replication regions, Deletion protection Off, Favorite star, On-demand Read capacity mode.
 - users**: Active, Partition key `email ($)`, Sort key `-`, 0 indexes, 0 replication regions, Deletion protection Off, Favorite star, On-demand Read capacity mode.
- Actions**: Buttons for Actions, Delete, and Create table.
- Search**: A search bar at the top.
- Navigation**: AWS logo, navigation icons, and tabs for United States (N. Virginia).
- Footer**: CloudShell, Feedback, © 2025, Amazon Web Services, Inc. or its affiliates., Privacy, Terms, Cookie preferences.

Milestone 4. SNS Notification Setup

- **Activity 4.1:** Create SNS topics for book request notifications.



The screenshot shows the AWS search results for 'sns' with the following details:

- Search results for 'sns'**
- Services** section:
 - Simple Notification Service** (starred): SNS managed message topics for Pub/Sub.
 - Route 53 Resolver**: Resolve DNS queries in your Amazon VPC and on-premises network.
 - Route 53** (starred): Scalable DNS and Domain Name Registration.
 - AWS End User Messaging** (starred): Engage your customers across multiple communication channels.
- Features** section:
 - Events**: ElastiCache feature.
 - SMS**: AWS End User Messaging feature.
 - Hosted zones**: Route 53 feature.

Amazon SNS X

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

Dashboard
Topics
Subscriptions
▼ Mobile
Push notifications
Text messaging (SMS)

Application Integration

Amazon Simple Notification Service

Pub/sub messaging for microservices and serverless applications.

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

Create topic

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

Next step

Start with an overview

Pricing



- Click on Create Topic and choose a name for the 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

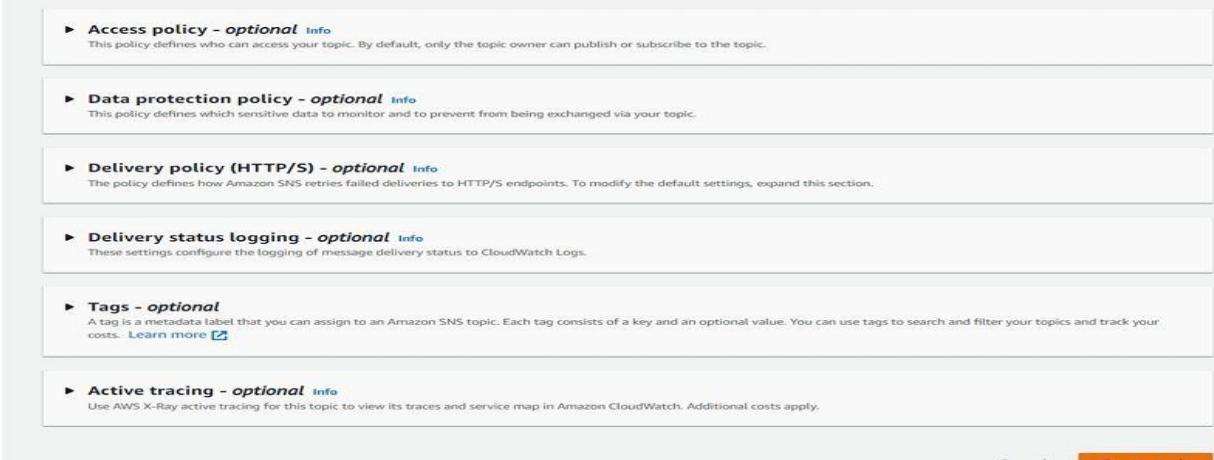
- Best-effort message ordering
- At-least once message delivery
- Highest throughput in publishes/second
- Subscription protocols: SQS, Lambda, HTTP, SMS, email, mobile application endpoints

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**

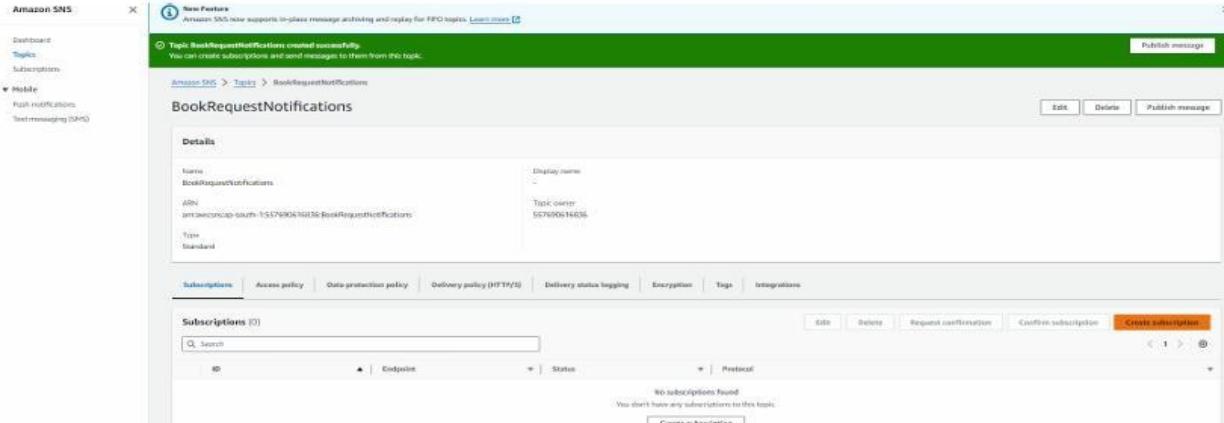
A tag is a metadata label that you can assign to an Amazon SNS topic. Each tag consists of a key and an optional value. You can use tags to search and filter your topics and track your costs. [Learn more](#)

► **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



Amazon SNS

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

Topic BookRequestNotifications created successfully. You can create subscriptions and send messages to them from this topic.

Amazon SNS > Topics > BookRequestNotifications

BookRequestNotifications

Details

Name: BookRequestNotifications	Display name: -
ARN: arn:aws:sns:ap-south-1:51799031638:BookRequestNotifications	Topic owner: 51799031638
Type: Standard	

Subscriptions [0]

ID	Endpoint	Status	Protocol
No subscriptions found.			

[Edit](#) [Delete](#) [Publish message](#) [Create subscription](#)

- Subscribe users and library staff to SNS email notifications.

Amazon SNS > Subscriptions > Create subscription

Create subscription

Details

Topic ARN X

Protocol ▼

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

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

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

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

Cancel Create subscription

Amazon SNS

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

Confirmation report was sent successfully. The ARN of the subscription is arn:aws:sns:ap-south-1:557690616836:BookRequestNotifications:7ee5d16-12a1-47f1-9791-c5dc2215ad5. X

Amazon SNS > Topics > BookRequestNotifications

BookRequestNotifications

Edit Delete Publish message

Details

Name	BookRequestNotifications	Display name	-
ARN	arn:aws:sns:ap-south-1:557690616836:bookRequestNotifications	topic owner	557690616836
Type	Standard		

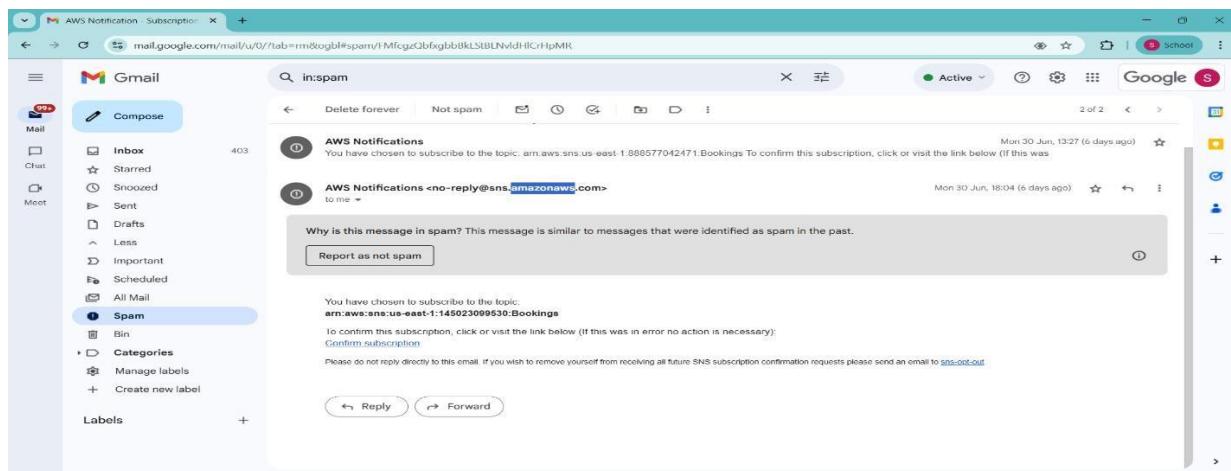
Subscriptions Access policy Data protection policy Delivery policy (HTTP/S) Delivery status logging Encryption Tags Integrations

Subscriptions (2)

ID	Endpoint	Status	Protocol
1	instantlibrary2@gmail.com	Pending confirmation	Email

Edit Delete Request confirmation Confirm subscription Create subscription

- After Confirmation of Subscription going to Mail for comfirm Mail .



aws Simple Notification Service

Subscription confirmed!

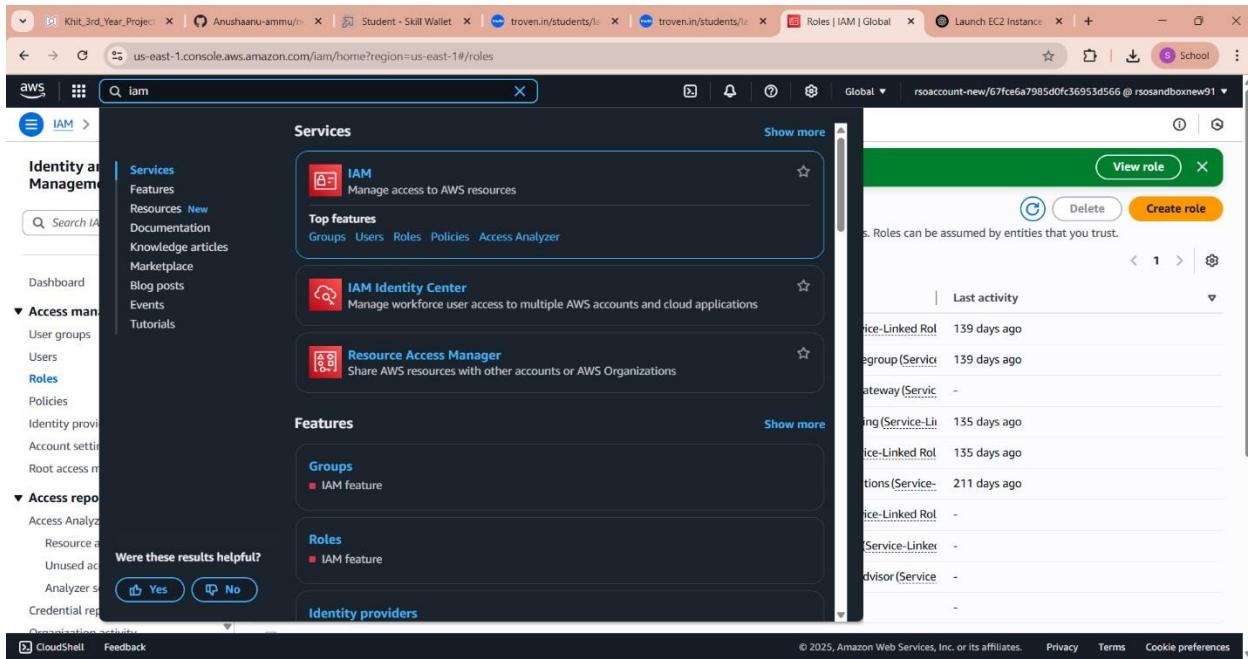
You have successfully subscribed.

Your subscription's id is:
`arn:aws:sns:ap-south-1:557690616836:BookRequestNotifications:d78e0371-9235-404d-952c-85c2743607c4`

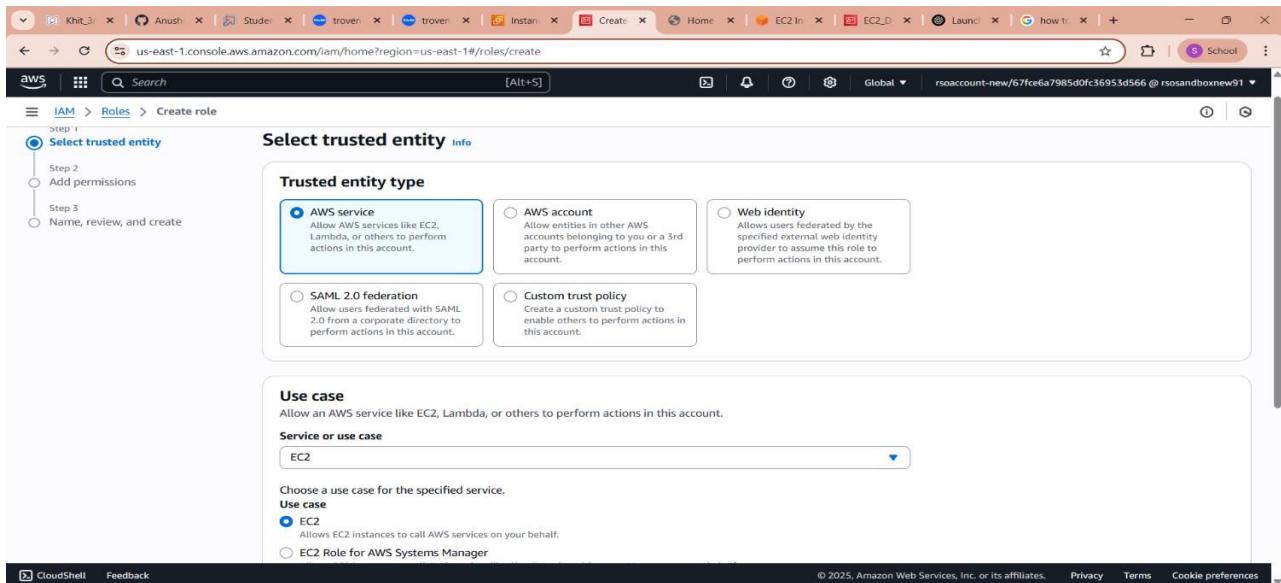
If it was not your intention to subscribe, [click here to unsubscribe](#).

Milestone 5 : IAM Role Setup.

- **Activity 5.1 :** In the AWS Console, go to IAM and create a new IAM Role for EC2 to interact with DynamoDB .



The screenshot shows the AWS IAM service console. The left sidebar has a tree view with 'Identity and Management' expanded, showing 'Services', 'Features', 'Resources', 'Documentation', 'Knowledge articles', 'Marketplace', 'Blog posts', 'Events', and 'Tutorials'. Under 'Access management', 'Roles' is selected, showing 'Policies', 'Identity providers', and 'Access reports'. A 'Were these results helpful?' poll is present. The main content area is titled 'Services' and lists 'IAM', 'IAM Identity Center', and 'Resource Access Manager'. It also has a 'Features' section with 'Groups' and 'Roles'. On the right, a table lists roles with their last activity and actions. At the bottom, there are links for CloudShell and Feedback.



The screenshot shows the 'Create role' wizard, Step 1: Select trusted entity. The left sidebar shows steps: Step 1 (Select trusted entity), Step 2 (Add permissions), and Step 3 (Name, review, and create). The main content area is titled 'Select trusted entity'. It shows a 'Trusted entity type' section with five options: 'AWS service' (selected), 'AWS account', 'Web identity', 'SAML 2.0 federation', and 'Custom trust policy'. Below this is a 'Use case' section with a dropdown menu set to 'EC2'. At the bottom, there are sections for 'Service' and 'Use case', with 'EC2' selected. The footer includes CloudShell, Feedback, and standard links.

- **Activity 5.2 : Attach Policies**
- **AmazonDynamoDBFullAccess:** Allows EC2 to perform read/write operations on DynamoDB.
- **AmazonSNSFullAccess:** Grants EC2 the ability to send notifications via SNS.

IAM > Roles > Create role

Step 1: Select trusted entity

Step 2: Add permissions

Step 3: Name, review, and create

Add permissions

Permissions policies (1/955) Info

Choose one or more policies to attach to your new role.

Filter by Type: All types | 2 matches

Policy name	Type
<input checked="" type="checkbox"/>  AmazonDynamoDBFullAccess	AWS managed
<input type="checkbox"/>  AmazonDynamoDBReadOnlyAccess	AWS managed

▶ Set permissions boundary - optional

Cancel Previous Next

IAM > Roles > Create role

Step 1: Select trusted entity

Step 2: Add permissions

Step 3: Name, review, and create

Add permissions

Permissions policies (2/955) Info

Choose one or more policies to attach to your new role.

Filter by Type: All types | 5 matches

Policy name	Type
<input checked="" type="checkbox"/>  AmazonSNSFullAccess	AWS managed
<input type="checkbox"/>  AmazonSNSReadOnlyAccess	AWS managed
<input type="checkbox"/>  AmazonSNSRole	AWS managed
<input type="checkbox"/>  AWSLambdaBasicExecutionRole	AWS managed
<input type="checkbox"/>  AWSIoTDeviceDefenderPublishFindingsToSNSMigrationAction	AWS managed

▶ Set permissions boundary - optional

Cancel Previous Next

us-east-1.console.aws.amazon.com/iam/home?region=us-east-1#/roles/details/EC2_DynamoDB_Role?section=permissions

IAM > Roles > EC2_DynamoDB_Role

EC2_DynamoDB_Role Info

Allows EC2 instances to call AWS services on your behalf.

Summary

Creation date
July 03, 2025, 12:37 (UTC+05:30)

Last activity
43 minutes ago

ARN
arn:aws:iam::463470967337:role/EC2_DynamoDB_Role

Maximum session duration
1 hour

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

Permissions policies (3) Info

You can attach up to 10 managed policies.

Filter by Type: All types

Policy name	Type	Attached entities
<input type="checkbox"/>  AmazonDynamoDBFullAccess	AWS managed	1
<input type="checkbox"/>  AmazonEC2FullAccess	AWS managed	1

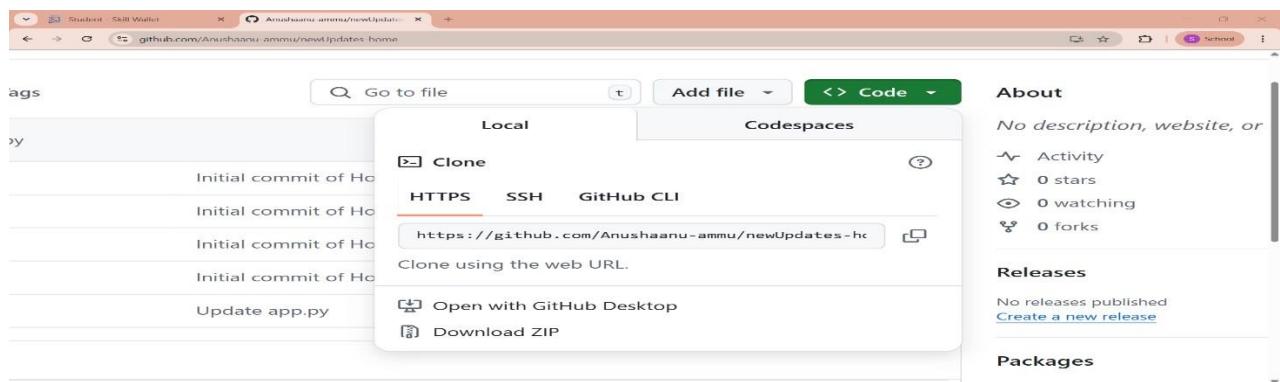
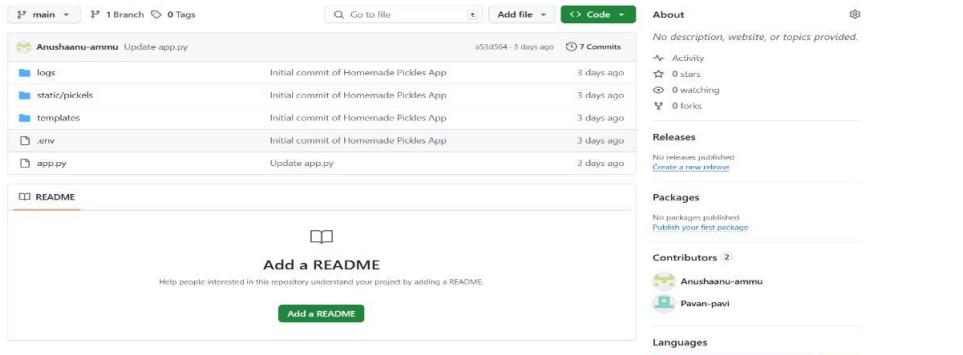
(   )

CloudShell Feedback

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

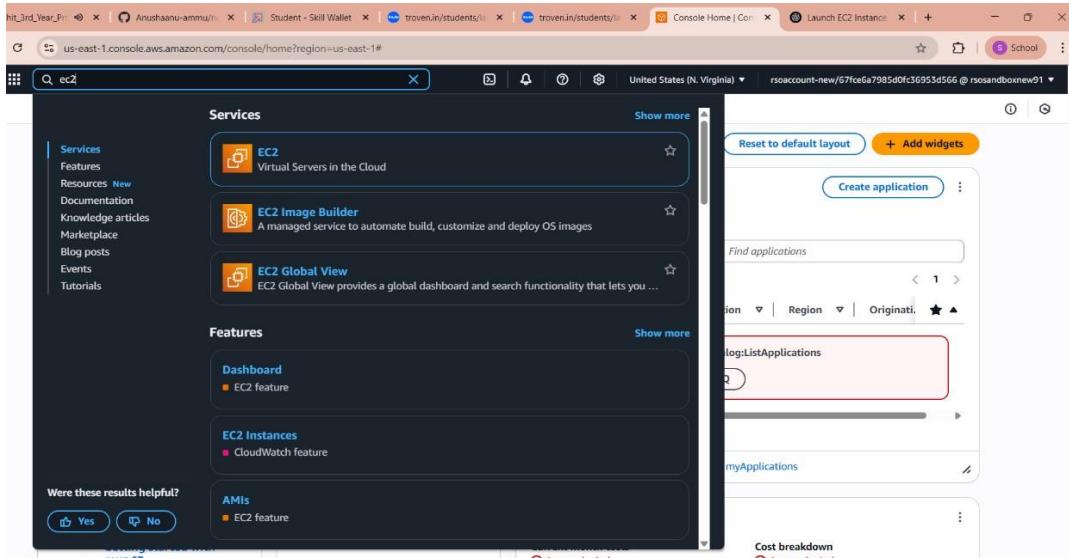
○ Milestone 6. EC2 Instance Setup

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



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

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





AWS | Search [Alt+S] United States (N. Virginia) rsoaccount-new/67fce6a7985d0fc36953d566 @ rsosandboxnew91

EC2 > Instances > Launch an instance

Launch an instance Info

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

Name and tags Info

Name Add additional tags

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

Recent | Quick Start

Amazon Linux macOS Ubuntu Windows Red Hat SUSE Linux Debian

Browse more AMIs Including AMIs from AWS, Marketplace and the Community

Amazon Machine Image (AMI)

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

AWS | Search [Alt+S] United States (N. Virginia) rsoaccount-new/67fce6a7985d0fc36953d566 @ rsosandboxnew91

EC2 > Instances > Launch an instance

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 Create new key pair

Network settings Info

Network Info vpc-06b1983fd7eaa46c7

Subnet Info No preference (Default subnet in any availability zone)

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. Add rules to allow specific traffic to reach your instance.

Create security group Select existing security group

We'll create a new security group called "launch-wizard-2" with the following rules:

Allow SSH traffic from Anywhere 0.0.0.0/0

Allow HTTPS traffic from the internet

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

EC2 > Security Groups > sg-0f1fc765e39312322 - launch-wizard-1

sg-0f1fc765e39312322 - launch-wizard-1

Details

Security group name	sg-0f1fc765e39312322
Owner	463470967537
Inbound rules count	4 Permission entries
Description	launch-wizard-1 created 2025-07-03T07:00:14.084Z
VPC ID	vpc-06b1983fd7eaa46c7

Inbound rules | Outbound rules | Sharing - new | VPC associations - new | Tags

Inbound rules (4)

Name	Security group rule ID	IP version	Type	Protocol	Port range
-	sgr-0eff08b1b9dd0920d	IPv4	HTTP	TCP	80
-	sgr-0116436ad44282685	IPv4	SSH	TCP	22
-	sgr-0fef928cff38eb8ea	IPv4	Custom TCP	TCP	5000
-	sgr-06e009f2b9c9b8208	IPv4	HTTPS	TCP	443

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

AWS Search United States (N. Virginia) rsoaccount-new/G7fce6a7985d0fc36953d566 @ rsosandboxnew91 ▾

EC2 Instances

Instances (1) Info

Find instance by attribute or tag (case-sensitive)

Instance state = running | Clear filters

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4
HomeMadePic...	i-0bc9fb84cdde8e83a	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1c	ec2-44-220-144-100

Select an instance

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

AWS Search United States (N. Virginia) rsoaccount-new/67fce6a7985d0fc36953d566 @ rsosandboxnew91 ▾

EC2 Instances

Instances (1/1) Info

Find instance by attribute or tag (case-sensitive)

Instance state = running | Clear filters

Name	Instance ID	Instance state	Instance type	Security
HomeMadePic...	i-0bc9fb84cdde8e83a	Running	t2.micro	Change security groups Get Windows password Modify IAM role

i-0bc9fb84cdde8e83a (HomeMadePickles)

Details Status and alarms Monitoring Security Networking Storage Tags

Instance summary

Instance ID	Public IPv4 address	Private IPv4 addresses
i-0bc9fb84cdde8e83a	44.220.144.100 open address	172.31.25.32
IPv6 address	Instance state	Public DNS
-	Running	ec2-44-220-144-100.compute-1.amazonaws.com open address

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

AWS Search United States (N. Virginia) rsoaccount-new/67fce6a7985d0fc36953d566 @ rsosandboxnew91 ▾

EC2 > Instances > i-0bc9fb84cdde8e83a > Modify IAM role

Modify IAM role Info

Attach an IAM role to your instance.

Instance ID: i-0bc9fb84cdde8e83a (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 Create new IAM role

Cancel Update IAM role

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

us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#InstancesinstanceState=running

EC2 > Instances

Successfully attached EC2_DynamoDB_Role to instance i-0bc9fb84cdde8e83a

Instances (1/1) Info

Find Instance by attribute or tag (case-sensitive)

All states

Instance state = running

Clear filters

Name: HomeMadePic... Instance ID: i-0bc9fb84cdde8e83a Instance state: Running Instance type: t2.micro Status check: ... Alarm status: ... Availability Zone: us-east-1c Public IPv4: ec2-44-220-

i-0bc9fb84cdde8e83a (HomeMadePickles)

Details Status and alarms Monitoring Security Networking Storage Tags

Instance summary

Instance ID: i-0bc9fb84cdde8e83a Public IPv4 address: 44.220.144.100 | open address Private IP4 addresses: 172.31.25.32 Public DNS: ec2-44-220-144-100.compute-1.amazonaws.com | open address

IPv6 address: - Instance state: Running

CloudShell Feedback

aws Search [Alt+S]

EC2 > Instances > i-0bc9fb84cdde8e83a > 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-0bc9fb84cdde8e83a (HomeMadePickles)

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

Public IPv4 address: 44.220.144.100

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

https://us-east-1.console.aws.amazon.com/ec2/instance/connect/tsh?region=us-east-1&connType=standard&instanceId=i-0bc9fb84cdde8e83a&osUser=ec2-user&sshPort=22&addressFamily=ipv4

Amazon Linux 2023

https://aws.amazon.com/linux/amazon-linux-2023

{ec2-user@ip-172-31-25-32 ~} \$

Make Chrome faster

Memory Saver frees up memory from inactive tabs so it can be used by active tabs and other apps

Turn on No thanks

i-0bc9fb84cdde8e83a (HomeMadePickles)

Public IPs: 44.220.144.100 Private IPs: 172.31.25.32

CloudShell Feedback

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

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/NeelimaKayasiri/Neelima.git' .

This will download the Project to Ec2 instance .

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

```
cd InstantLibrary
```

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
```

aws | ⚡ Search [Alt+S] United States (N. Virginia) rsoaccount-new/67fce6a7985d0fc36953d566 @ rsosandboxnew91

```

Amazon Linux 2023
https://aws.amazon.com/linux/amazon-linux-2023

[ec2-user@ip-172-31-25-32 ~]$ sudo yum update -y
Amazon Linux 2023 Kernel Livepatch repository
Dependencies resolved.
Nothing to do.
Complete!
[ec2-user@ip-172-31-25-32 ~]$ sudo yum install python3 git -y
Last metadata expiration check: 0:00:30 ago on Thu Jul 3 07:28:40 2025.
Package python3-3.9.23-1.amzn2023.0.1.x86_64 is already installed.
Dependencies resolved.

      Package           Architecture     Version          Repository      Size
Installing:   git                  x86_64        2.47.1-1.amzn2023.0.3
Installing dependencies: git-core      x86_64        2.47.1-1.amzn2023.0.3
git-core-doc  noarch            2.47.1-1.amzn2023.0.3
perl-Error   noarch            1:0.17029-5.amzn2023.0.2

i-0bc9fb84cdde8e83a (HomeMadePickles)
Public IPs: 44.220.144.100 Private IPs: 172.31.25.32

```

CloudShell Feedback [Alt+S] United States (N. Virginia) rsoaccount-new/67fce6a7985d0fc36953d566 @ rsosandboxnew91

```

(8/8): g1c-0016-2-47-1-1-amzn2023.0.3.x86_64.rpm
(6/8): perl-Git-2.47.1-1.amzn2023.0.3.noarch.rpm
(7/8): perl_TermReadKey-2.38-9.amzn2023.0.2.x86_64.rpm
(8/8): perl-lib-0.65-477.amzn2023.0.7.x86_64.rpm

Total 4.0 MB/s | 4.2 MB 00:00
1.2 MB/s | 40 kB 00:00
1.1 MB/s | 36 kB 00:00
670 kB/s | 15 kB 00:00
44 MB/s | 7.5 MB 00:00

Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
Preparing:
  Installing : git-core-2.47.1-1.amzn2023.0.3.x86_64
  Installing : git-core-doc-2.47.1-1.amzn2023.0.3.noarch
  Installing : perl-Git-2.47.1-1.amzn2023.0.3.x86_64
  Installing : perl-TermReadKey-2.38-9.amzn2023.0.2.x86_64
  Installing : perl-file-Find-1.37-477.amzn2023.0.7.noarch
  Installing : perl-Error-1:0.17029-5.amzn2023.0.2.noarch
  Installing : perl-Git-2.47.1-1.amzn2023.0.3.noarch
  Installing : perl-Git-2.47.1-1.amzn2023.0.3.noarch
  Running scriptlets: git-2.47.1-1.amzn2023.0.3.x86_64
  Verifying   : git-2.47.1-1.amzn2023.0.3.x86_64
  Verifying   : git-core-2.47.1-1.amzn2023.0.3.x86_64
  Verifying   : git-core-doc-2.47.1-1.amzn2023.0.3.noarch
  Verifying   : perl-Git-2.47.1-1.amzn2023.0.3.noarch
  Verifying   : perl-TermReadKey-2.38-9.amzn2023.0.2.noarch
  Verifying   : perl-file-Find-1.37-477.amzn2023.0.7.noarch
  Verifying   : perl-Error-1:0.17029-5.amzn2023.0.2.noarch
  Verifying   : perl-lib-0.65-477.amzn2023.0.7.x86_64
  Verifying   : perl-lib-0.65-477.amzn2023.0.7.x86_64

Installed:
  git-2.47.1-1.amzn2023.0.3.x86_64
  perl-Error-1:0.17029-5.amzn2023.0.2.noarch
  perl-TermReadKey-2.38-9.amzn2023.0.2.x86_64

git-core-2.47.1-1.amzn2023.0.3.x86_64
perl-file-Find-1.37-477.amzn2023.0.7.noarch
perl-Git-2.47.1-1.amzn2023.0.3.noarch
perl-TermReadKey-2.38-9.amzn2023.0.2.x86_64

Total 4.0 MB/s | 4.2 MB 00:00
1.2 MB/s | 40 kB 00:00
1.1 MB/s | 36 kB 00:00
670 kB/s | 15 kB 00:00
44 MB/s | 7.5 MB 00:00

CloudShell Feedback [Alt+S] United States (N. Virginia) rsoaccount-new/67fce6a7985d0fc36953d566 @ rsosandboxnew91
```

aws | ⚡ Search [Alt+S] United States (N. Virginia) rsoaccount-new/67fce6a7985d0fc36953d566 @ rsosandboxnew91

```

$ export AWS_REGION=us-east-1
$ export FLASK_APP=app.py
$ sudo python3 -m flask run --host=0.0.0.0 --port=80
* bash: cd: newUpdates-home: No such file or directory
* Tip: There are .env files present. Install python-dotenv to use them.
Usage: python -m flask run [OPTIONS]
Try 'python -m flask run --help' for help.

Error: While importing 'app', an ImportError was raised:

Traceback (most recent call last):
  File "/usr/local/lib/python3.9/site-packages/flask/cli.py", line 245, in locate_app
    import (module_name)
  File "/home/ec2-user/newUpdates-home/app.py", line 10, in <module>
    from dotenv import load_dotenv
ModuleNotFoundError: No module named 'dotenv'

[ec2-user@ip-172-31-25-32 newUpdates-home]$ sudo pip3 install python-dotenv
Collecting python-dotenv
  Downloading python_dotenv-1.1.1-py3-none-any.whl (20 kB)
Installing collected packages: python-dotenv
Successfully installed python-dotenv-1.1.1
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/warnings/venv
[ec2-user@ip-172-31-25-32 newUpdates-home]$ export AWS_REGION=us-east-1
$ export FLASK_APP=app.py
$ sudo python3 -m flask run --host=0.0.0.0 --port=80
2025-07-03 08:04:13,993 - INFO - Found credentials from IAM Role: EC2_DynamoDB_Role
* Debug mode: off
2025-07-03 08:04:14,095 - 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://127.0.0.1:80
* Running on http://172.31.25.32:80
2025-07-03 08:04:14,096 - INFO - Press CTRL+C to quit

```

Access the website through:

Public IPs: <http://44.220.144.100/>

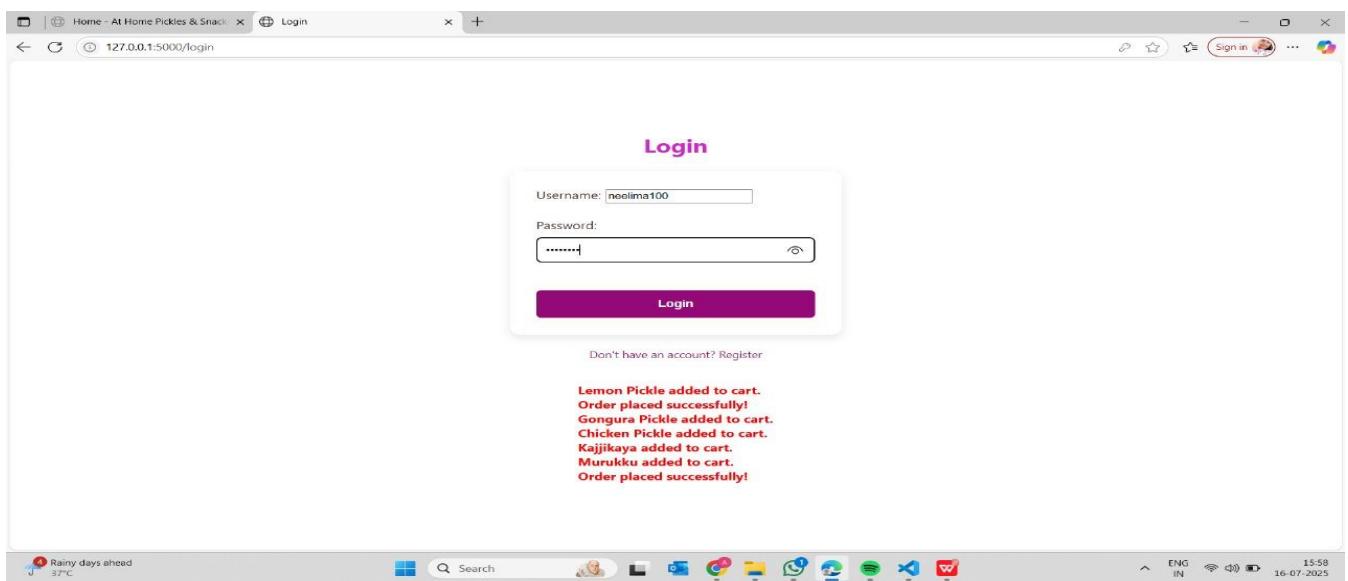
Milestone 8. Testing and Deployment

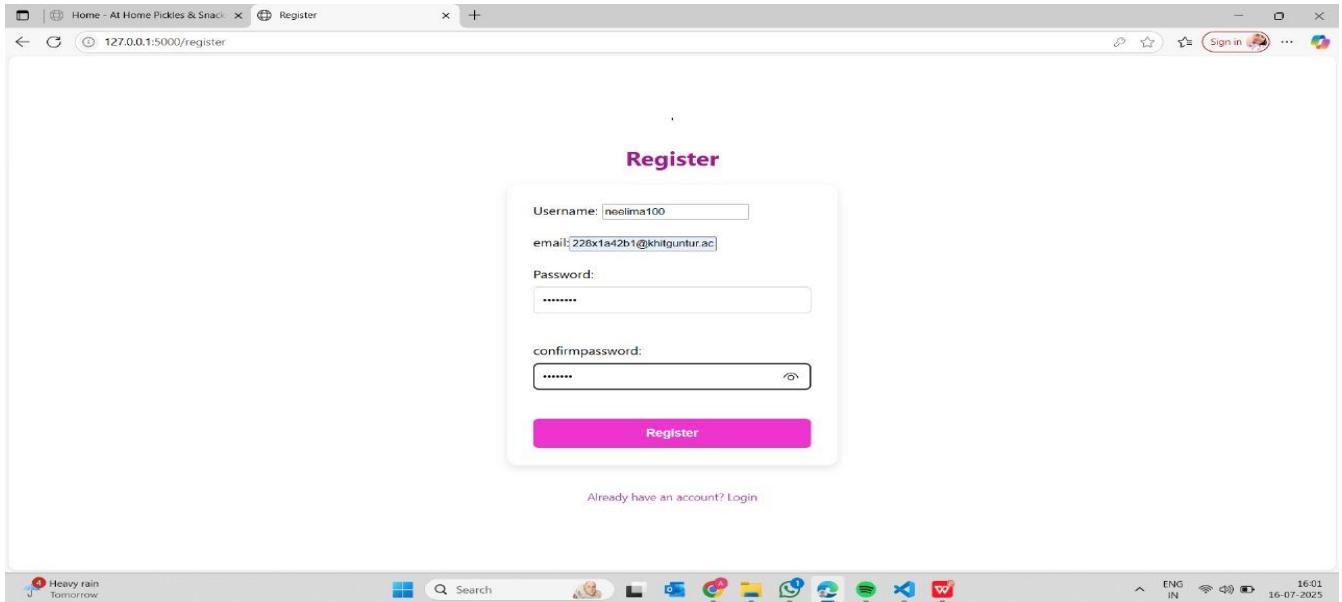
Activity 8.1 : Conduct functional testing to verify user signup, login, buy/sell stocks and notifications.

HOME PAGE:



LOGIN PAGE :



SIGNUP PAGE :


Username: neelima100

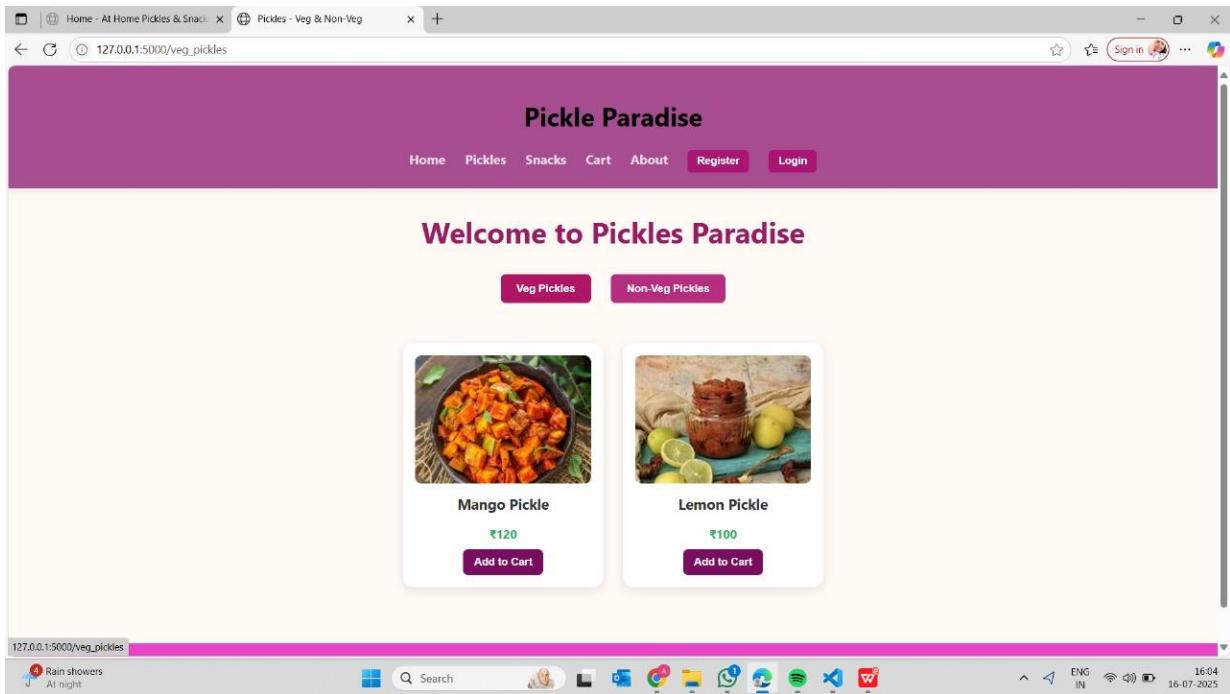
email: 228x1a42b1@khitguntur.ac

Password:

confirm password:

[Register](#)

Already have an account? [Login](#)

VEG PAGE :


Pickle Paradise

Home Pickles Snacks Cart About Register Login

Welcome to Pickles Paradise

Veg Pickles Non-Veg Pickles

Mango Pickle ₹120 Add to Cart

Lemon Pickle ₹100 Add to Cart

127.0.0.1:5000/veg_pickles

NON VEG PAGE :

Home - At Home Pickles & Snack X Pickles - Veg & Non-Veg X +

127.0.0.1:5000/non_veg_pickles

Pickle Paradise

Home Pickles Snacks Cart About Register Login

Welcome to Pickles Paradise

Veg Pickles Non-Veg Pickles



Chicken Pickle

₹180

Add to Cart



Prawn Pickle

₹200

Add to Cart

Rain showers At night

Search

ENG IN 16-07-2025

SNACKS PAGE :

Home - At Home Pickles & Snack X Snacks X +

127.0.0.1:5000/snacks

Delicious Homemade Snacks

Home Pickles Snacks Cart About Register Login



Murukku

₹60

★★★★★

Add to Cart



Mixture

₹70

★★★★★

Add to Cart



Kajjikaya

₹80

★★★★★

Add to Cart



Chekkalu

₹50

★★★★★

Add to Cart



Boondi

₹65

★★★★★

Add to Cart

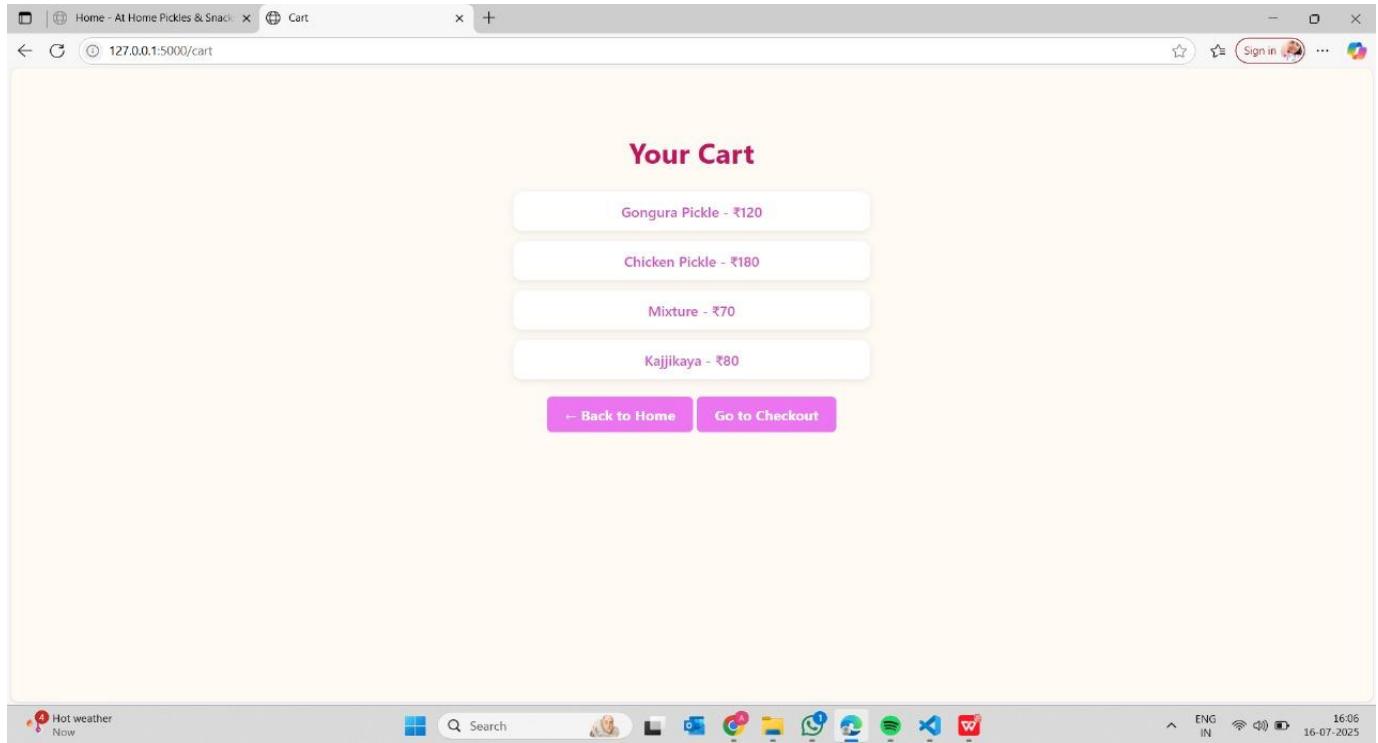


Chakodi

Hot days ahead 37°C

Search

ENG IN 16-07-2025

CART PAGE :

Your Cart

Gongura Pickle - ₹120

Chicken Pickle - ₹180

Mixture - ₹70

Kajjikaya - ₹80

[-- Back to Home](#) [Go to Checkout](#)

CHECK OUT PAGE :

Home - At Home Pickles & Snack Checkout

127.0.0.1:5000/checkout

Sign in

Checkout

Image	Item	Price (₹)
	Gongura Pickle	₹120
	Chicken Pickle	₹180
	Mixture	₹70
	Kajjikaya	₹80
Total		₹450

[Place Order](#)

SA - NZ In 1 hour

Search

16-07-2025

SUCCESS PAGE :

Home - At Home Pickles & Snack Order Successful

127.0.0.1:5000/order_success

Sign in

Thank You!

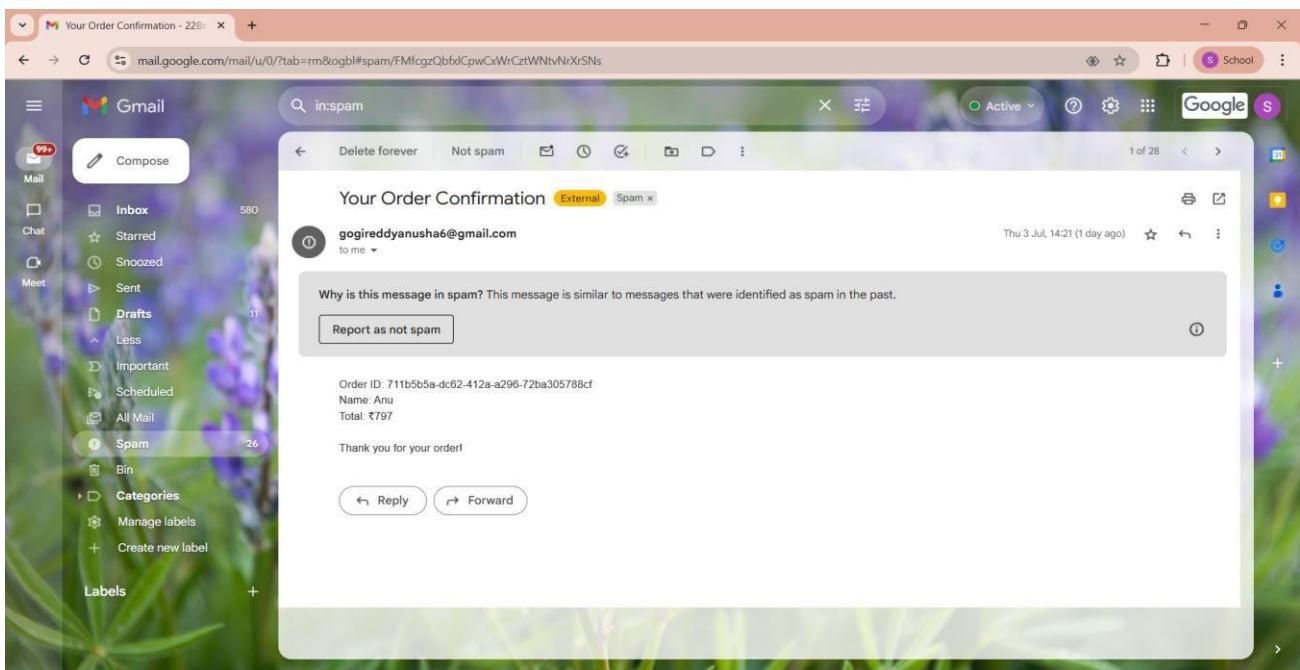
Your order was placed successfully. You will receive a confirmation shortly.

[Back to Home](#)

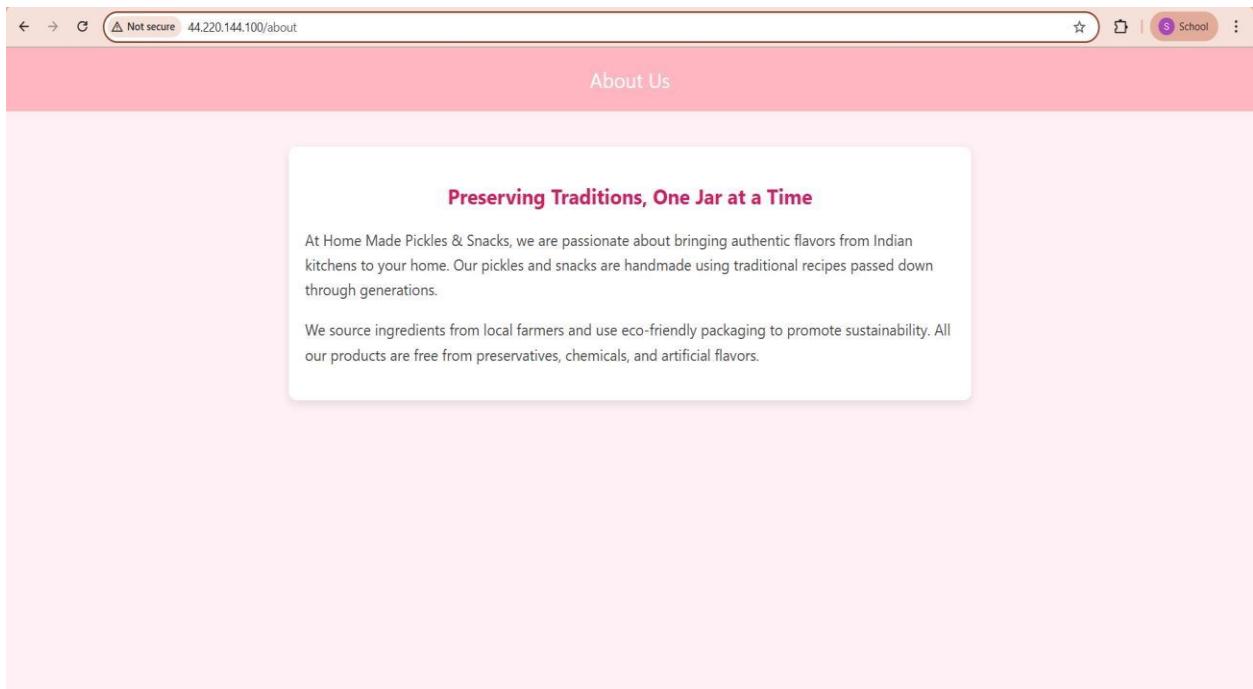
SA - NZ In 1 hour

Search

16-07-2025



ABOUT PAGE :



Not secure 44.220.144.100/about

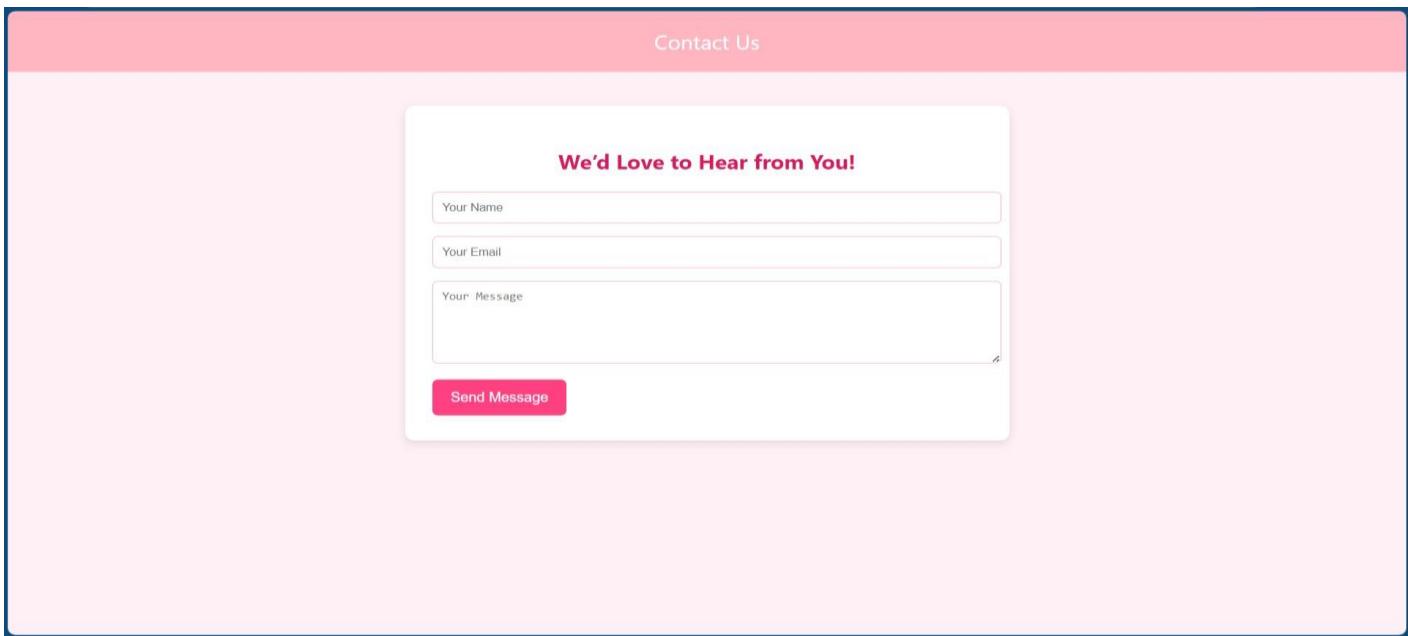
About Us

Preserving Traditions, One Jar at a Time

At Home Made Pickles & Snacks, we are passionate about bringing authentic flavors from Indian kitchens to your home. Our pickles and snacks are handmade using traditional recipes passed down through generations.

We source ingredients from local farmers and use eco-friendly packaging to promote sustainability. All our products are free from preservatives, chemicals, and artificial flavors.

CONTACT PAGE :



Contact Us

We'd Love to Hear from You!

Your Name

Your Email

Your Message

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.

.....THE END.....