Home Made 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."

Scenario 1: Scalable Order Management for High Demand

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

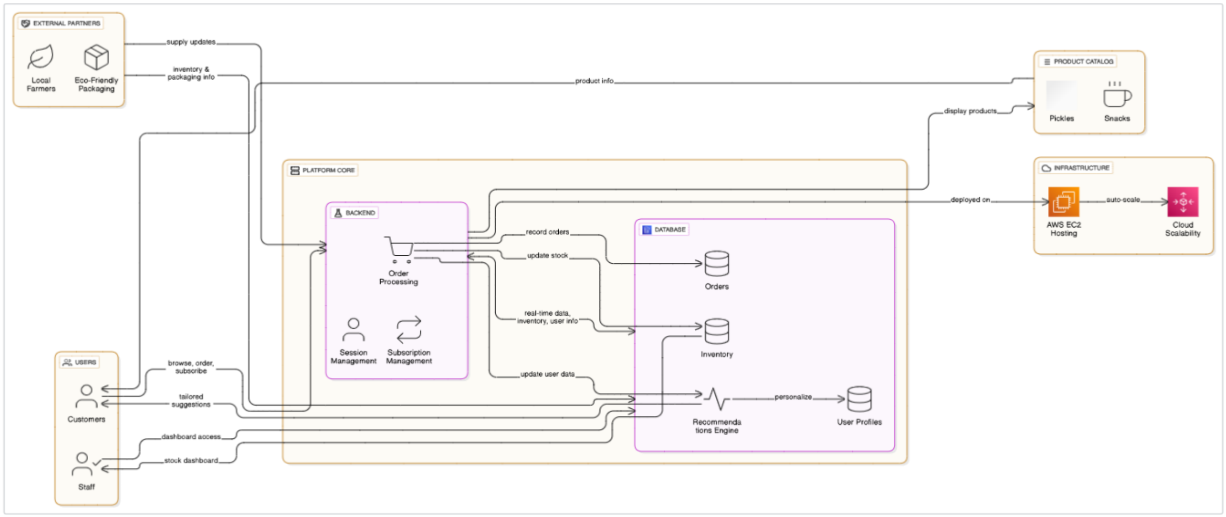
Scenario 2: Real-Time Inventory Tracking and Updates

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

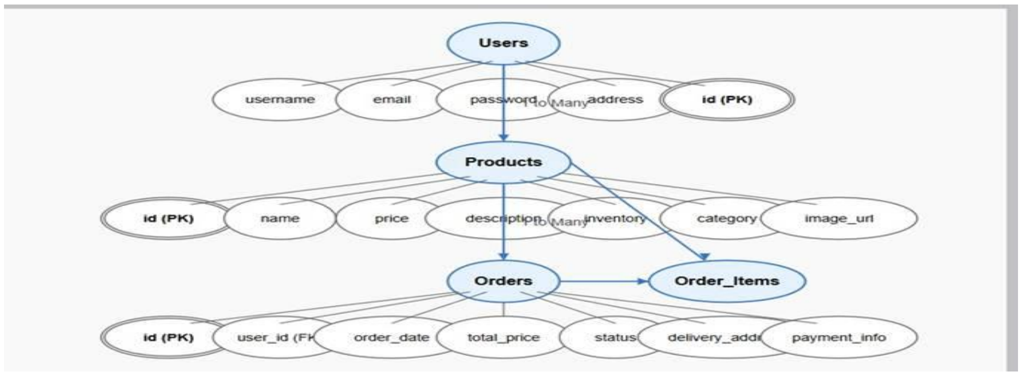
Scenario 3: Personalized User Experience and Recommendations

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

AWS ARCHITECTURE



Entity Relationship (ER)Diagram:



Pre-requisites:

- 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 Work Flow:

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

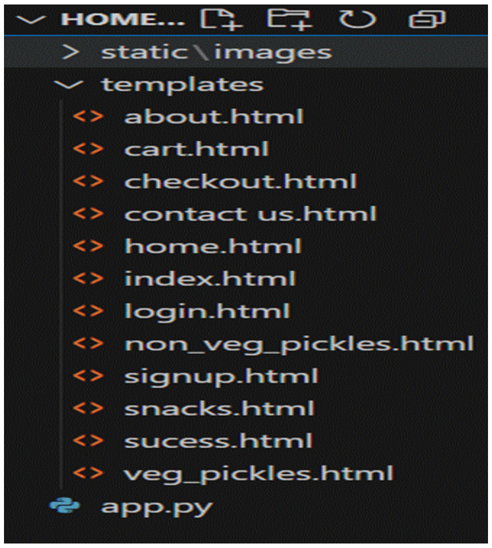
- Conduct functional testing to verify user signu

1: Web Application Development and Setup

Milestone 1: Web Application Development and Setup

- **Activity 1.1: Set up an AWS account if not already done.**
  - Begin by building essential HTML pages and Flask routes using local Python dictionaries or lists for data storage. This allows testing and validation of core functionality before integrating cloud services.

- **Activity 1.2: Core Functionalities and User Interaction.**
  - Implement core features like user registration, login, and data submission using local storage. Ensure smooth navigation between pages and basic input validation on both frontend and backend.



```
HOME...
  > static\images
  ∨ templates
      about.html
      cart.html
      checkout.html
      contact us.html
      home.html
      index.html
      login.html
      non_veg_pickles.html
      signup.html
      snacks.html
      sucess.html
      veg_pickles.html
  app.py
```

Description: set up the Home-Made Pickles project with an app.py file, a static/ folder for assets, and a templates/ directory containing all required HTML pages like home, login, register, products page etc.

Description of the code:

- **Flask App Initialization**

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

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

```
app=Flask(__name__)
app.secret_key = os.urandom(24)
```

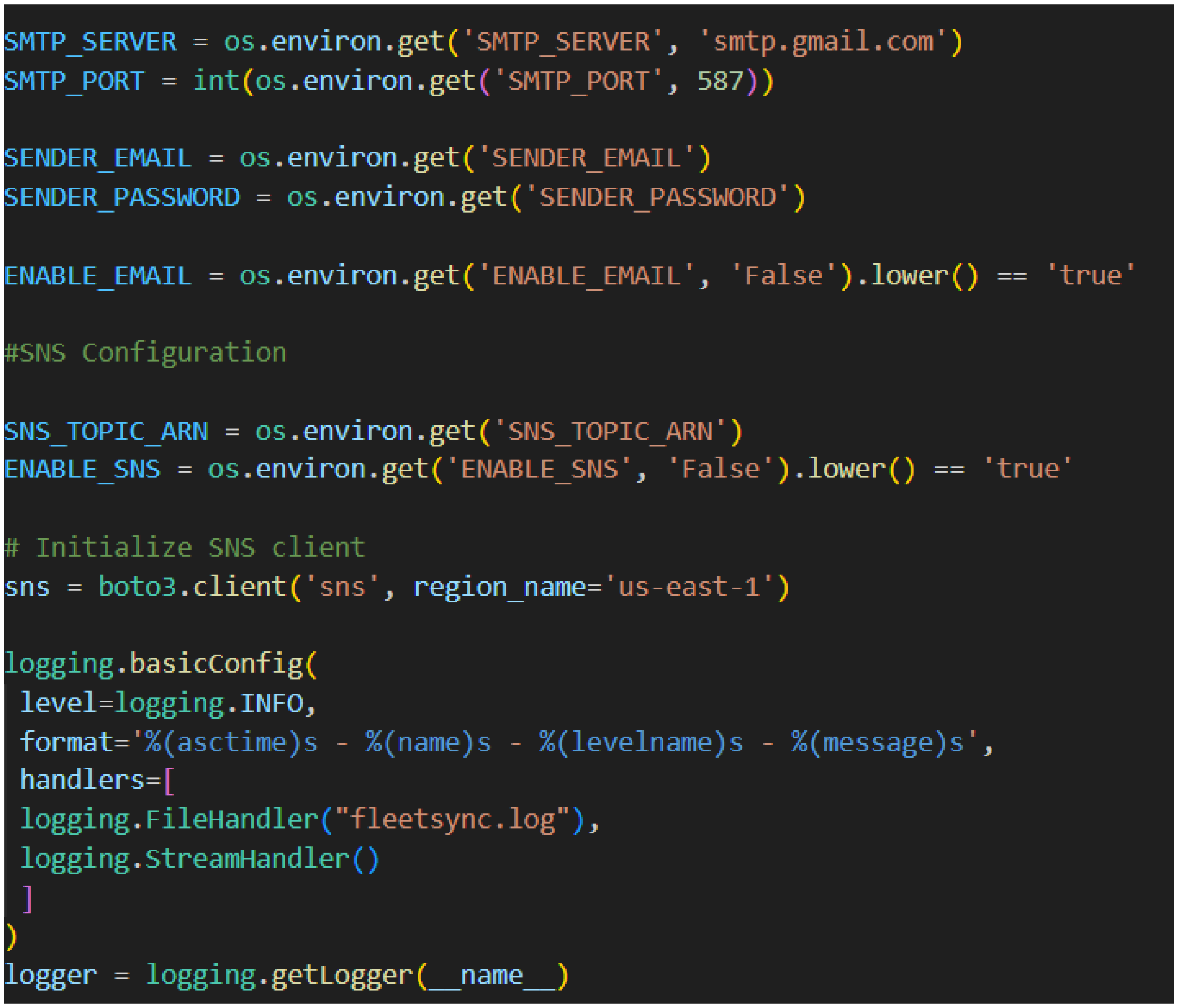
Description: initialize the Flask application instance using Flask(_name_) to start building the web app.

- **Dynamodb Setup:**

```
dynamodb = boto3.resource('dynamodb', region_name='us-east-1')
users_table = dynamodb.Table('Users')
orders_table = dynamodb.Table('Orders')
```

Description: initialize the DynamoDB resource for the us-east-1 region and set up access to the Users and Orders tables for storing user details and Orders requests.

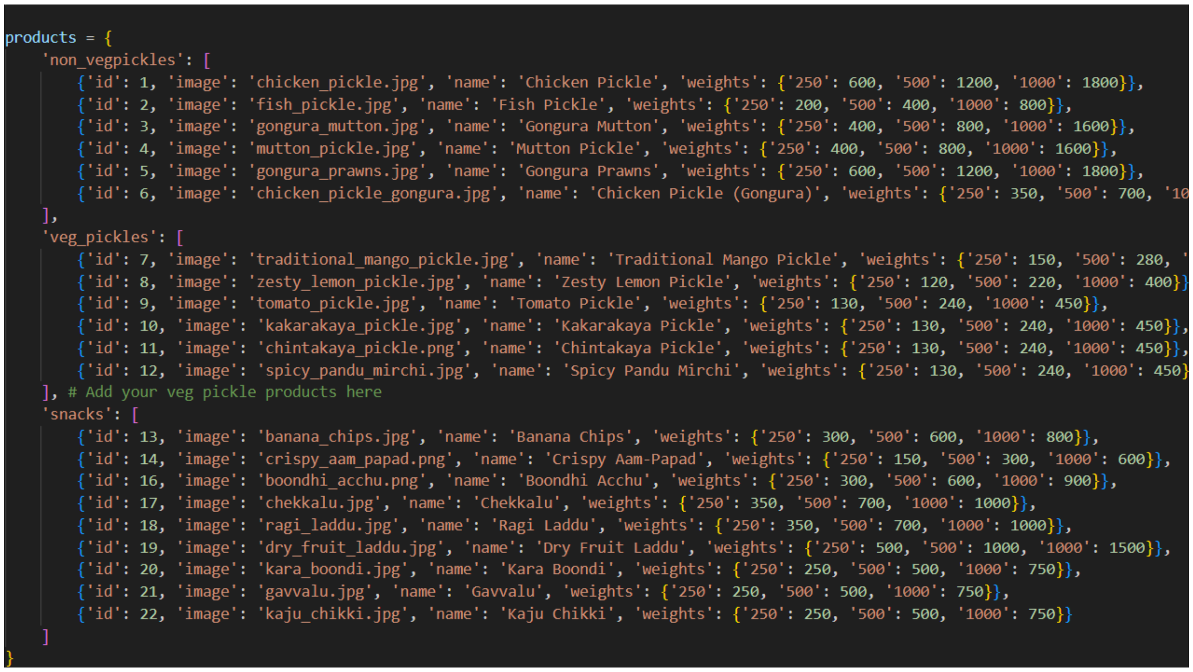
- **SNS Connection**



```
SMTP_SERVER = os.environ.get('SMTP_SERVER', 'smtp.gmail.com')
SMTP_PORT = int(os.environ.get('SMTP_PORT', 587))

SENDER_EMAIL = os.environ.get('SENDER_EMAIL')
SENDER_PASSWORD = os.environ.get('SENDER_PASSWORD')

ENABLE_EMAIL = os.environ.get('ENABLE_EMAIL', 'False').lower() == 'true'

#SNS Configuration

SNS_TOPIC_ARN = os.environ.get('SNS_TOPIC_ARN')
ENABLE_SNS = os.environ.get('ENABLE_SNS', 'False').lower() == 'true'

# Initialize SNS client
sns = boto3.client('sns', region_name='us-east-1')

logging.basicConfig(
 level=logging.INFO,
 format='%(asctime)s - %(name)s - %(levelname)s - %(message)s',
 handlers=[
 logging.FileHandler("fleetsync.log"),
 logging.StreamHandler()
 ]
)
logger = logging.getLogger(__name__)
```

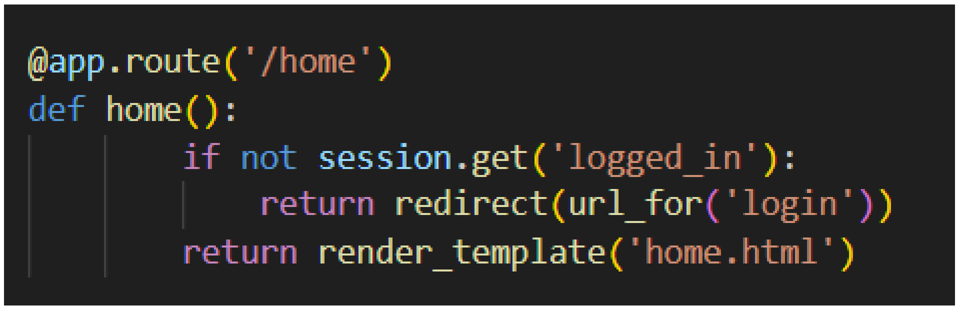
Description: Configure **SNS** to send notifications when a book request is submitted. Paste your stored ARN link in the sns_topic_arn space, along with the region_name where the SNS topic is created. Also, specify the chosen email service in SMTP_SERVER (e.g., Gmail, Yahoo, etc.) and enter the subscribed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

- **Products**



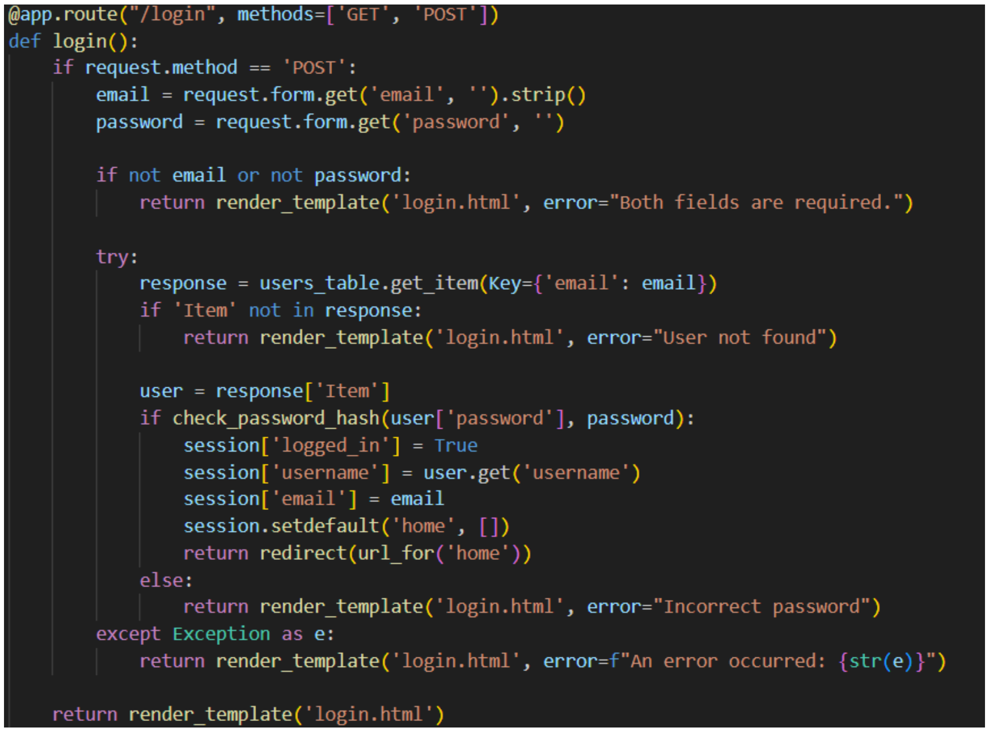
```
products = {
    'non_vegpickles': [
        {'id': 1, 'image': 'chicken_pickle.jpg', 'name': 'Chicken Pickle', 'weights': {'250': 600, '500': 1200, '1000': 1800}},
        {'id': 2, 'image': 'fish_pickle.jpg', 'name': 'Fish Pickle', 'weights': {'250': 200, '500': 400, '1000': 800}},
        {'id': 3, 'image': 'gongura_mutton.jpg', 'name': 'Gongura Mutton', 'weights': {'250': 400, '500': 800, '1000': 1600}},
        {'id': 4, 'image': 'mutton_pickle.jpg', 'name': 'Mutton Pickle', 'weights': {'250': 400, '500': 800, '1000': 1600}},
        {'id': 5, 'image': 'gongura_prawns.jpg', 'name': 'Gongura Prawns', 'weights': {'250': 600, '500': 1200, '1000': 1800}},
        {'id': 6, 'image': 'chicken_pickle_gongura.jpg', 'name': 'Chicken Pickle (Gongura)', 'weights': {'250': 350, '500': 700, '10
    ],
    'veg_pickles': [
        {'id': 7, 'image': 'traditional_mango_pickle.jpg', 'name': 'Traditional Mango Pickle', 'weights': {'250': 150, '500': 280, '
        {'id': 8, 'image': 'zesty_lemon_pickle.jpg', 'name': 'Zesty Lemon Pickle', 'weights': {'250': 120, '500': 220, '1000': 400}}
        {'id': 9, 'image': 'tomato_pickle.jpg', 'name': 'Tomato Pickle', 'weights': {'250': 130, '500': 240, '1000': 450}},
        {'id': 10, 'image': 'kakarakaya_pickle.jpg', 'name': 'Kakarakaya Pickle', 'weights': {'250': 130, '500': 240, '1000': 450}},
        {'id': 11, 'image': 'chintakaya_pickle.png', 'name': 'Chintakaya Pickle', 'weights': {'250': 130, '500': 240, '1000': 450}},
        {'id': 12, 'image': 'spicy_pandu_mirchi.jpg', 'name': 'Spicy Pandu Mirchi', 'weights': {'250': 130, '500': 240, '1000': 450}
    ], # Add your veg pickle products here
    'snacks': [
        {'id': 13, 'image': 'banana_chips.jpg', 'name': 'Banana Chips', 'weights': {'250': 300, '500': 600, '1000': 800}},
        {'id': 14, 'image': 'crispy_aam_papad.png', 'name': 'Crispy Aam-Papad', 'weights': {'250': 150, '500': 300, '1000': 600}},
        {'id': 16, 'image': 'boondhi_acchu.png', 'name': 'Boondhi Acchu', 'weights': {'250': 300, '500': 600, '1000': 900}},
        {'id': 17, 'image': 'chekkalu.jpg', 'name': 'Chekkalu', 'weights': {'250': 350, '500': 700, '1000': 1000}},
        {'id': 18, 'image': 'ragi_laddu.jpg', 'name': 'Ragi Laddu', 'weights': {'250': 350, '500': 700, '1000': 1000}},
        {'id': 19, 'image': 'dry_fruit_laddu.jpg', 'name': 'Dry Fruit Laddu', 'weights': {'250': 500, '500': 1000, '1000': 1500}},
        {'id': 20, 'image': 'kara_boondi.jpg', 'name': 'Kara Boondi', 'weights': {'250': 250, '500': 500, '1000': 750}},
        {'id': 21, 'image': 'gavvalu.jpg', 'name': 'Gavvalu', 'weights': {'250': 250, '500': 500, '1000': 750}},
        {'id': 22, 'image': 'kaju_chikki.jpg', 'name': 'Kaju Chikki', 'weights': {'250': 250, '500': 500, '1000': 750}}
    ]
}
```

- **Routes for Web Pages**
    - **Home Route:**



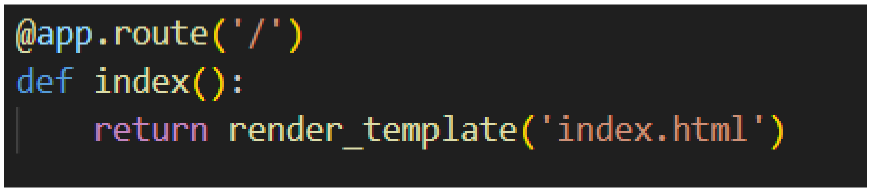
```
@app.route('/home')
def home():
        if not session.get('logged_in'):
            return redirect(url_for('login'))
        return render_template('home.html')
```

- **Login Route:**



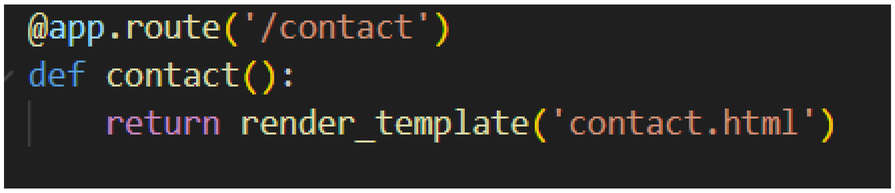
```
@app.route("/login", methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        email = request.form.get('email', '').strip()
        password = request.form.get('password', '')

        if not email or not password:
            return render_template('login.html', error="Both fields are required.")

        try:
            response = users_table.get_item(Key={'email': email})
            if 'Item' not in response:
                return render_template('login.html', error="User not found")

            user = response['Item']
            if check_password_hash(user['password'], password):
                session['logged_in'] = True
                session['username'] = user.get('username')
                session['email'] = email
                session.setdefault('home', [])
                return redirect(url_for('home'))
            else:
                return render_template('login.html', error="Incorrect password")
        except Exception as e:
            return render_template('login.html', error=f"An error occurred: {str(e)}")

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

- **Index Route:**



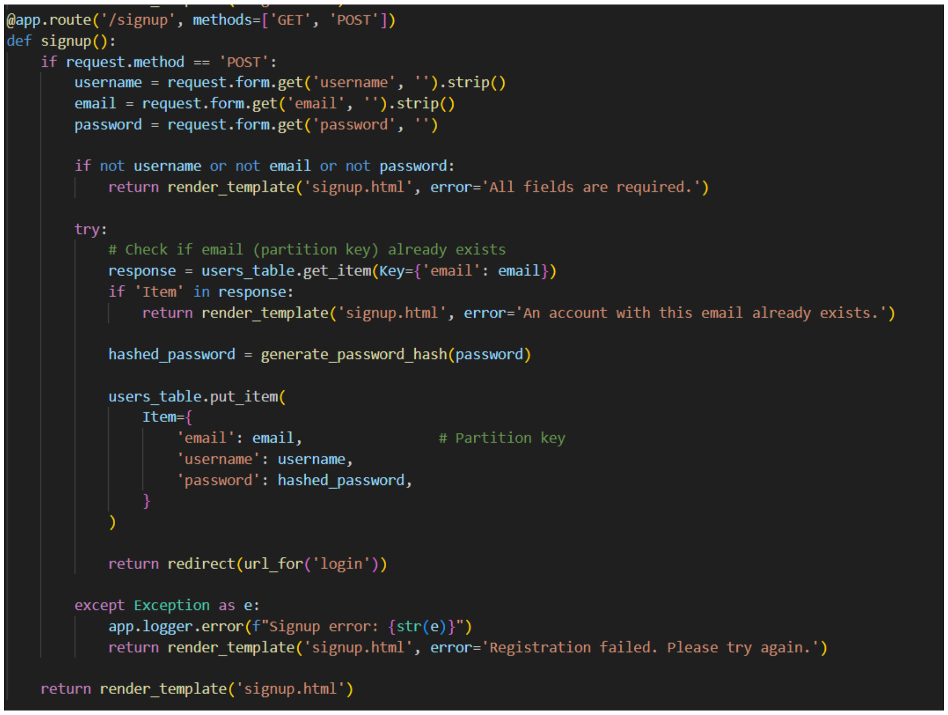
```
@app.route('/')
def index():
    return render_template('index.html')
```

- **Contact Route:**



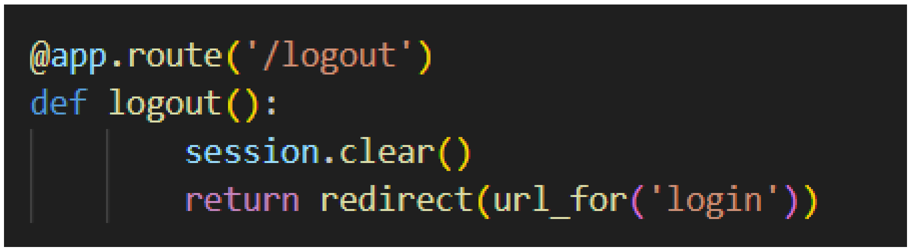
```
@app.route('/contact')
def contact():
    return render_template('contact.html')
```

- **Sign Up Route:**



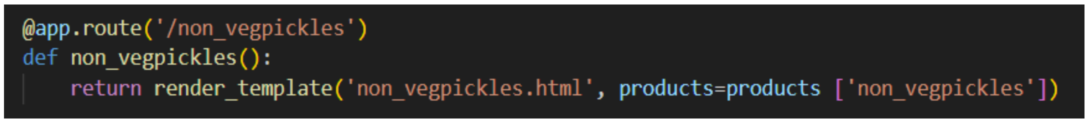
```python
@app.route('/signup', methods=['GET', 'POST'])
def signup():
    if request.method == 'POST':
        username = request.form.get('username', '').strip()
        email = request.form.get('email', '').strip()
        password = request.form.get('password', '')

        if not username or not email or not password:
            return render_template('signup.html', error='All fields are required.')

        try:
            # Check if email (partition key) already exists
            response = users_table.get_item(Key={'email': email})
            if 'Item' in response:
                return render_template('signup.html', error='An account with this email already exists.')

            hashed_password = generate_password_hash(password)

            users_table.put_item(
                Item={
                    'email': email,                 # Partition key
                    'username': username,
                    'password': hashed_password,
                }
            )

            return redirect(url_for('login'))

        except Exception as e:
            app.logger.error(f"Signup error: {str(e)}")
            return render_template('signup.html', error='Registration failed. Please try again.')

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

- **Log Out Route:**



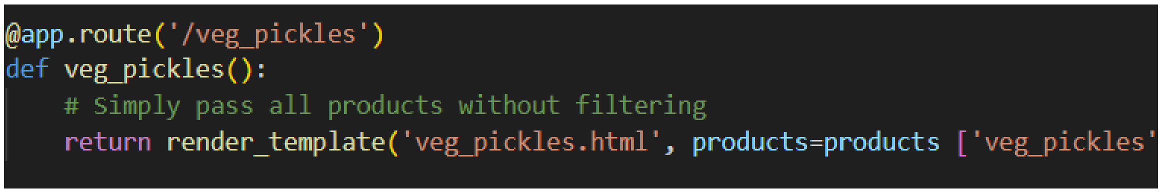
```python
@app.route('/logout')
def logout():
        session.clear()
        return redirect(url_for('login'))
```

- **Non-Veg Pickles Route:**



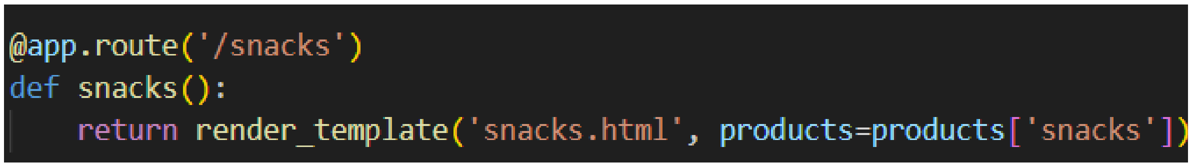
```python
@app.route('/non_vegpickles')
def non_vegpickles():
    return render_template('non_vegpickles.html', products=products ['non_vegpickles'])
```

- **Veg Pickles Route:**



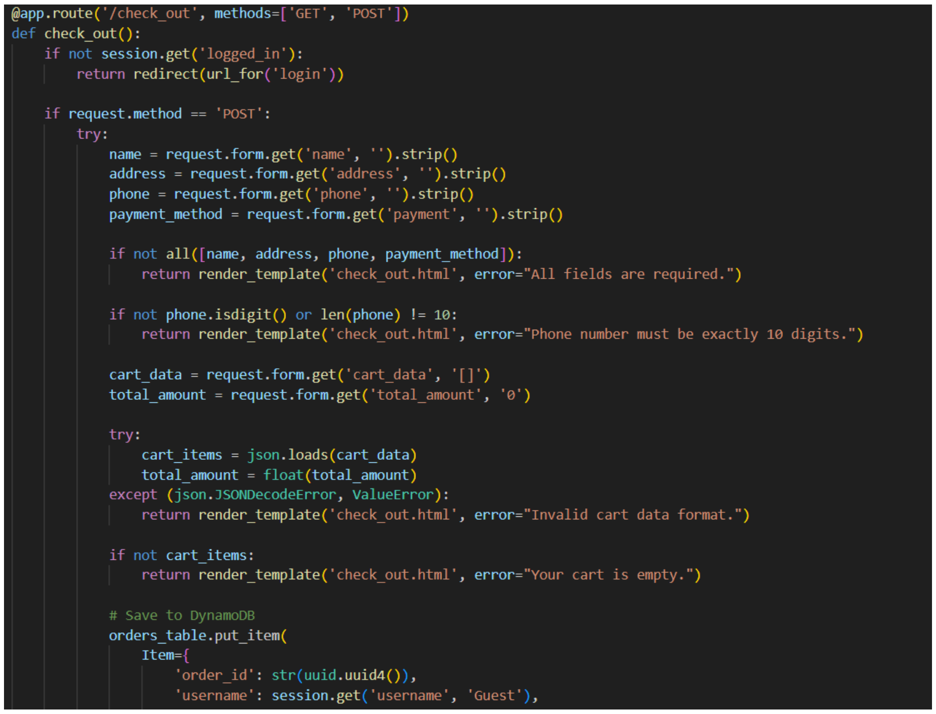
```
@app.route('/veg_pickles')
def veg_pickles():
    # Simply pass all products without filtering
    return render_template('veg_pickles.html', products=products ['veg_pickles'
```

- **Snacks Route:**



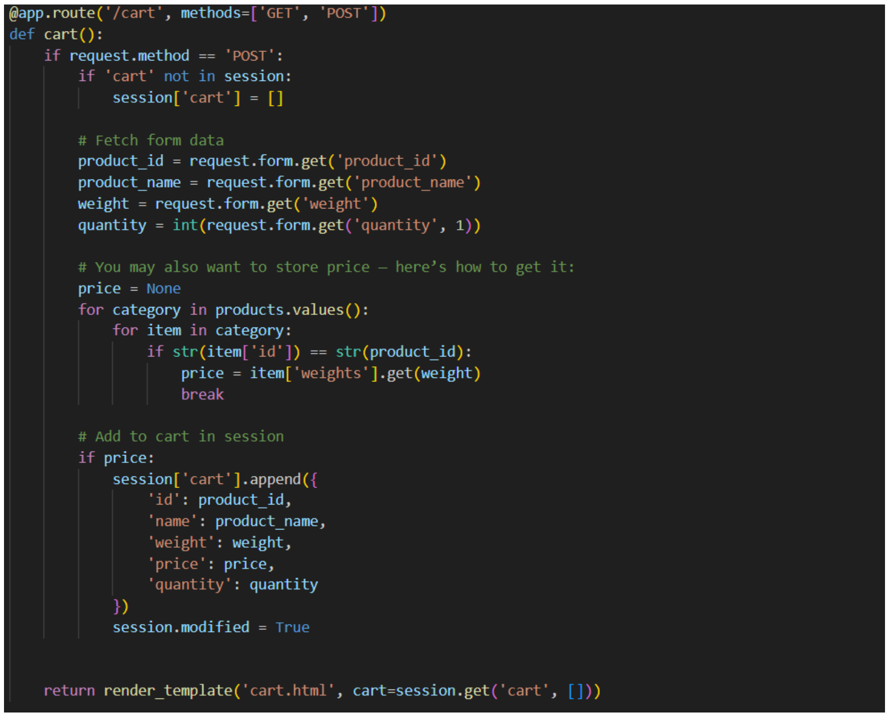
```
@app.route('/snacks')
def snacks():
    return render_template('snacks.html', products=products['snacks'])
```

- **Checkout Route:**



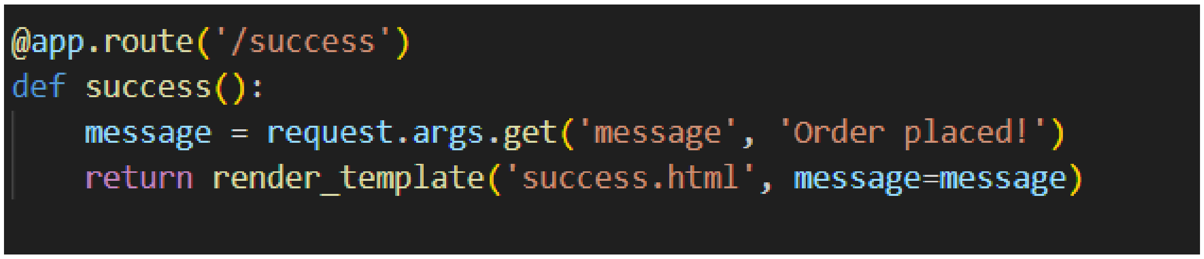
```
@app.route('/check_out', methods=['GET', 'POST'])
def check_out():
    if not session.get('logged_in'):
        return redirect(url_for('login'))

    if request.method == 'POST':
        try:
            name = request.form.get('name', '').strip()
            address = request.form.get('address', '').strip()
            phone = request.form.get('phone', '').strip()
            payment_method = request.form.get('payment', '').strip()

            if not all([name, address, phone, payment_method]):
                return render_template('check_out.html', error="All fields are required.")

            if not phone.isdigit() or len(phone) != 10:
                return render_template('check_out.html', error="Phone number must be exactly 10 digits.")

            cart_data = request.form.get('cart_data', '[]')
            total_amount = request.form.get('total_amount', '0')

            try:
                cart_items = json.loads(cart_data)
                total_amount = float(total_amount)
            except (json.JSONDecodeError, ValueError):
                return render_template('check_out.html', error="Invalid cart data format.")

            if not cart_items:
                return render_template('check_out.html', error="Your cart is empty.")

            # Save to DynamoDB
            orders_table.put_item(
                Item={
                    'order_id': str(uuid.uuid4()),
                    'username': session.get('username', 'Guest'),
```

- **Cart Route:**



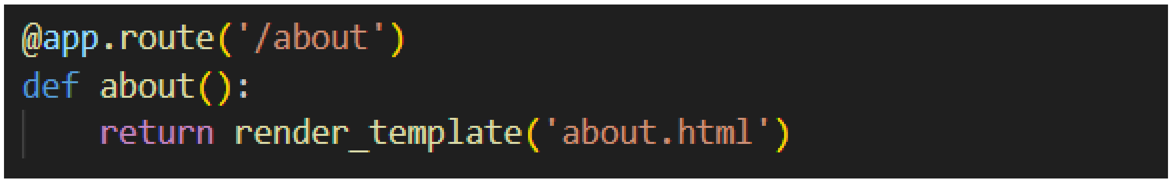
```python
@app.route('/cart', methods=['GET', 'POST'])
def cart():
    if request.method == 'POST':
        if 'cart' not in session:
            session['cart'] = []

        # Fetch form data
        product_id = request.form.get('product_id')
        product_name = request.form.get('product_name')
        weight = request.form.get('weight')
        quantity = int(request.form.get('quantity', 1))

        # You may also want to store price — here’s how to get it:
        price = None
        for category in products.values():
            for item in category:
                if str(item['id']) == str(product_id):
                    price = item['weights'].get(weight)
                    break

        # Add to cart in session
        if price:
            session['cart'].append({
                'id': product_id,
                'name': product_name,
                'weight': weight,
                'price': price,
                'quantity': quantity
            })
            session.modified = True


    return render_template('cart.html', cart=session.get('cart', []))
```

- **Success Route:**



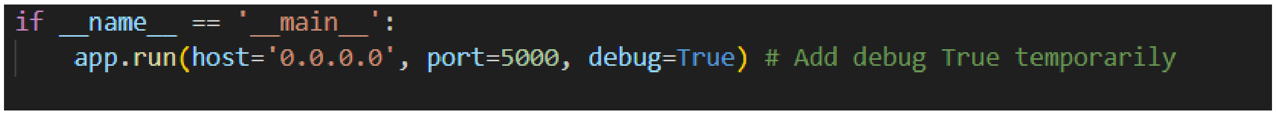
```python
@app.route('/success')
def success():
    message = request.args.get('message', 'Order placed!')
    return render_template('success.html', message=message)
```

- **About Route:**



```python
@app.route('/about')
def about():
    return render_template('about.html')
```

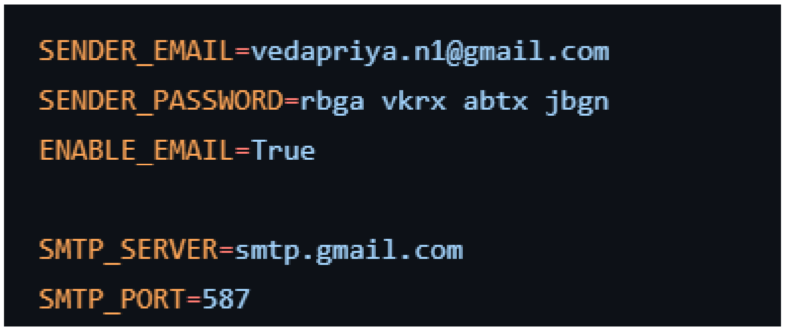
- **Deployment of code:**



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

Description: start the Flask server to listen on all network interfaces (`0.0.0.0`) at port `5000` with debug mode enabled for development and testing.

- **.env file:**

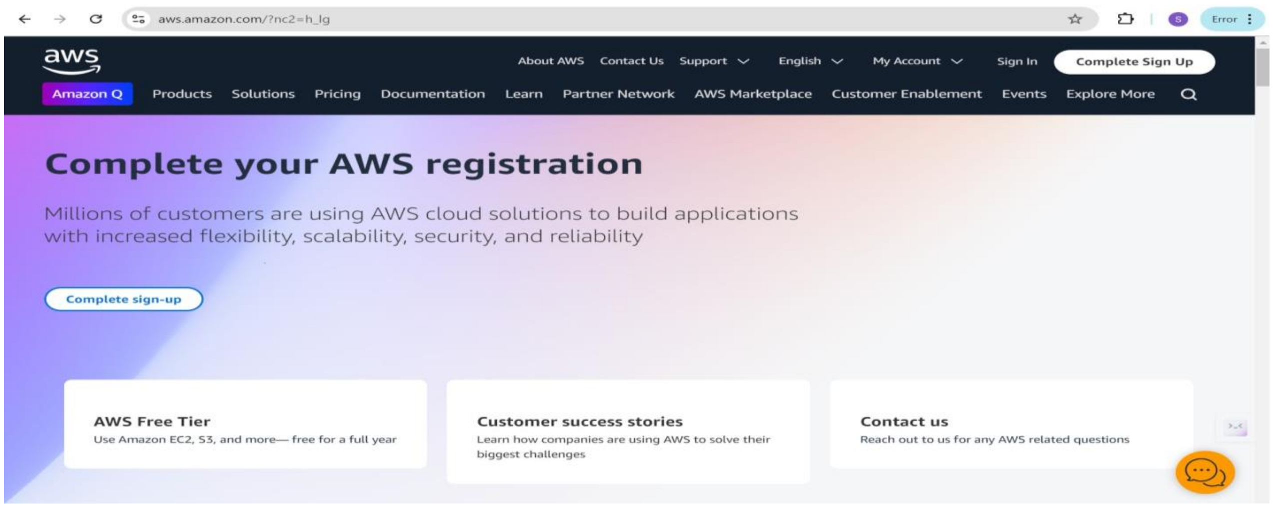


```
SENDER_EMAIL=vedapriya.n1@gmail.com
SENDER_PASSWORD=rbga vkrx abtx jbgn
ENABLE_EMAIL=True

SMTP_SERVER=smtp.gmail.com
SMTP_PORT=587
```

2: AWS Account Setup

Milestone 2: AWS Account Setup

- **Activity 2.1: Set up an AWS account if not already done.**
    - Begin by building essential HTML pages and Flask routes using local Python dictionaries or lists for data storage. This allows testing and validation of core functionality before integrating cloud services.



Complete your AWS registration

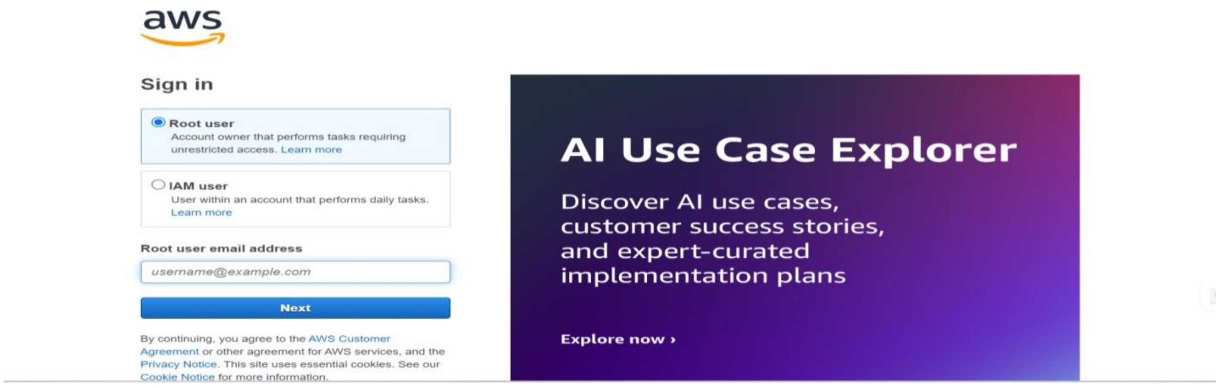
Millions of customers are using AWS cloud solutions to build applications with increased flexibility, scalability, security, and reliability

AWS Free Tier
Use Amazon EC2, S3, and more— free for a full year

Customer success stories
Learn how companies are using AWS to solve their biggest challenges

Contact us
Reach out to us for any AWS related questions

- **Activity 2.2: Log in to the AWS Management Console**
    - After setting up your account, log in to the AWS Management Console.



Account owner that performs tasks requiring unrestricted access. Learn more

User within an account that performs daily tasks. Learn more

By continuing, you agree to the AWS Customer Agreement or other agreement for AWS services, and the Privacy Notice. This site uses essential cookies. See our Cookie Notice for more information.

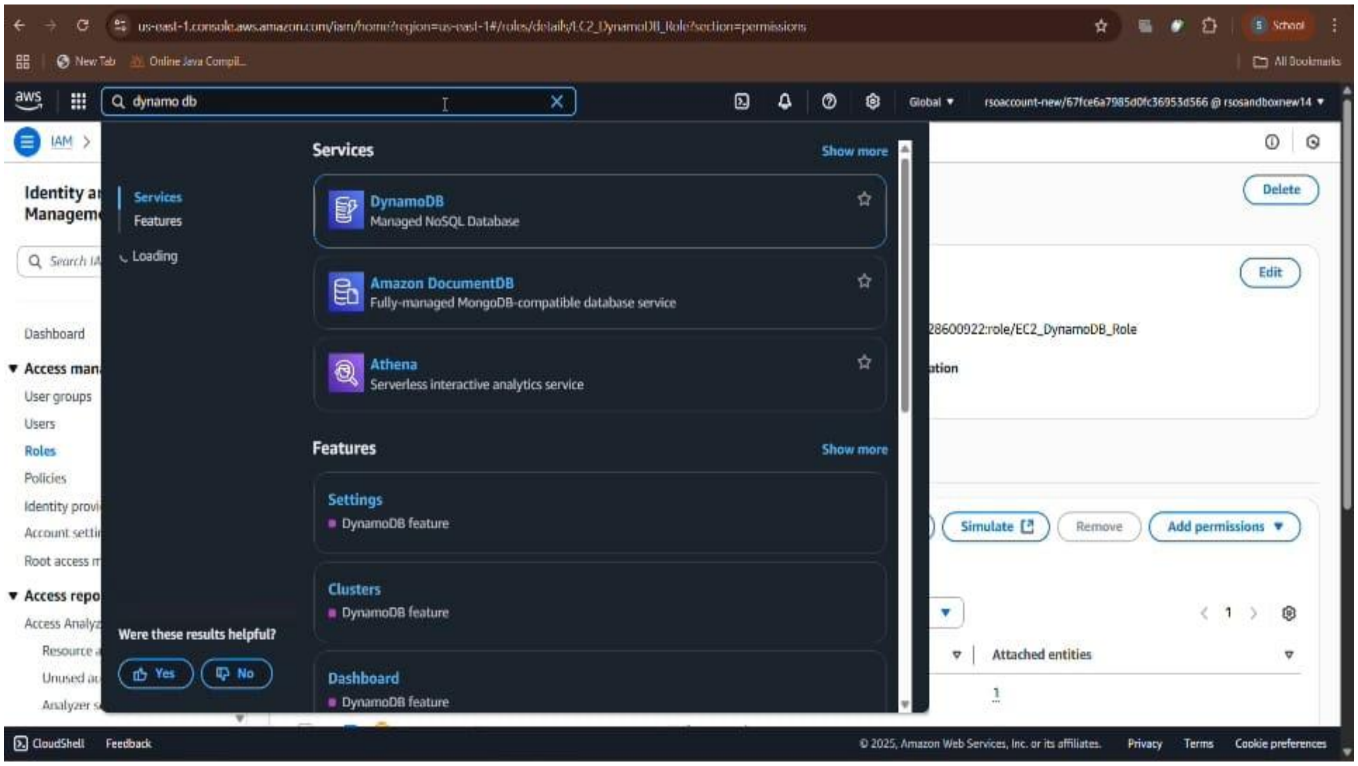
AI Use Case Explorer

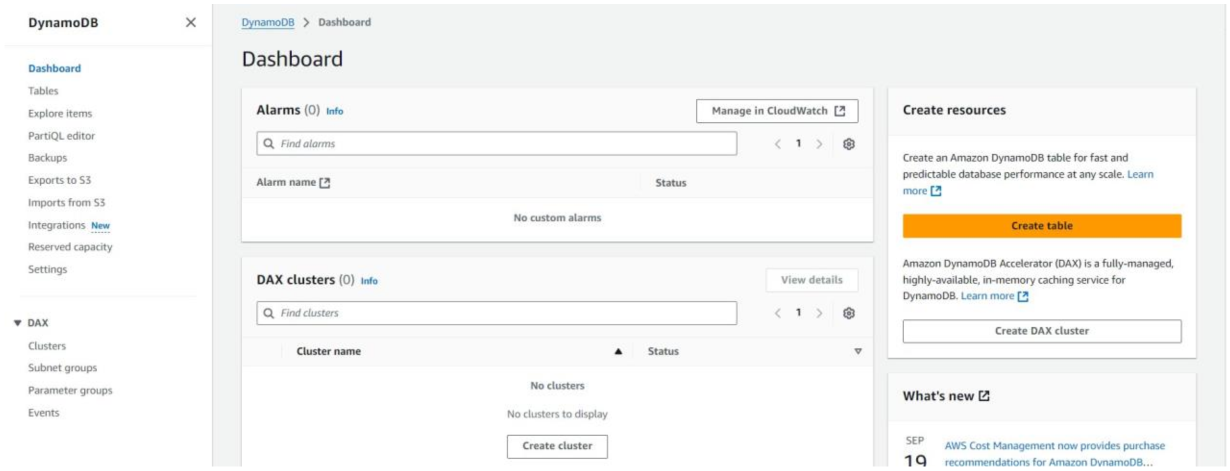
Discover AI use cases, customer success stories, and expert-curated implementation plans

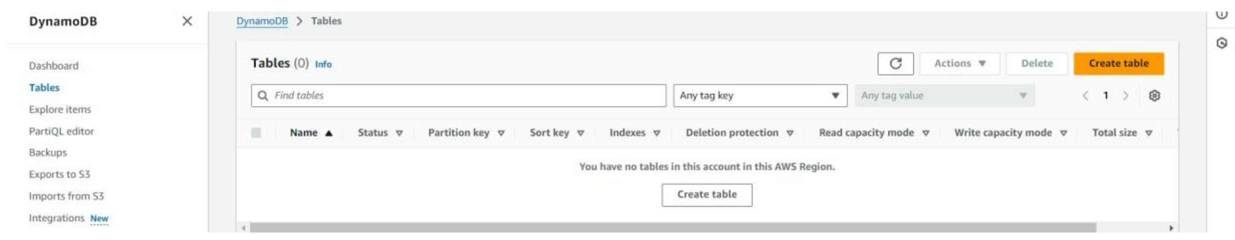
3: DynamoDB Database Creation and Setup

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.







DynamoDB ✕

Dashboard
Tables
Explore items
PartiQL editor
Backups
Exports to S3
Imports from S3
Integrations New

DynamoDB > Tables

Tables (0) Info

Actions ▼ | Delete | Create table

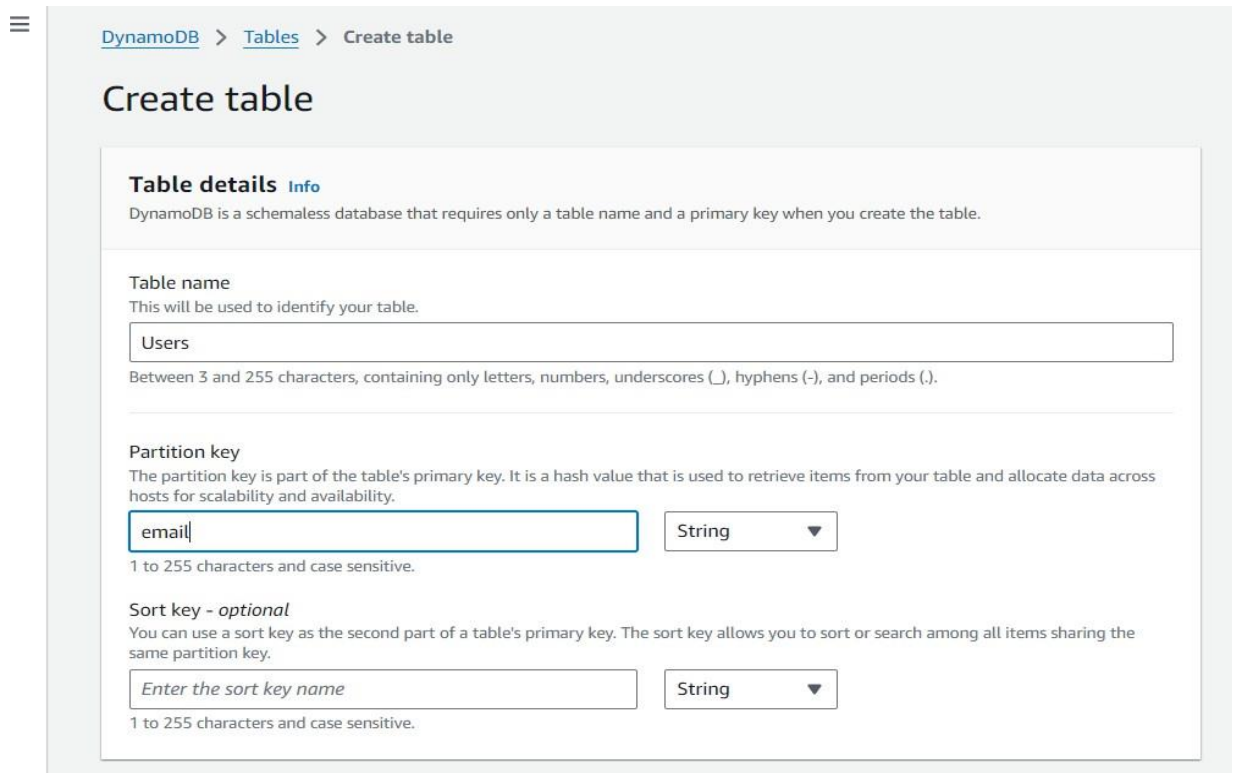
Find tables | Any tag key | Any tag value | < 1 >

	Name ▲	Status	Partition key	Sort key	Indexes	Deletion protection	Read capacity mode	Write capacity mode	Total size

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

Create table

- **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.



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.

Users

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.

email | String ▼

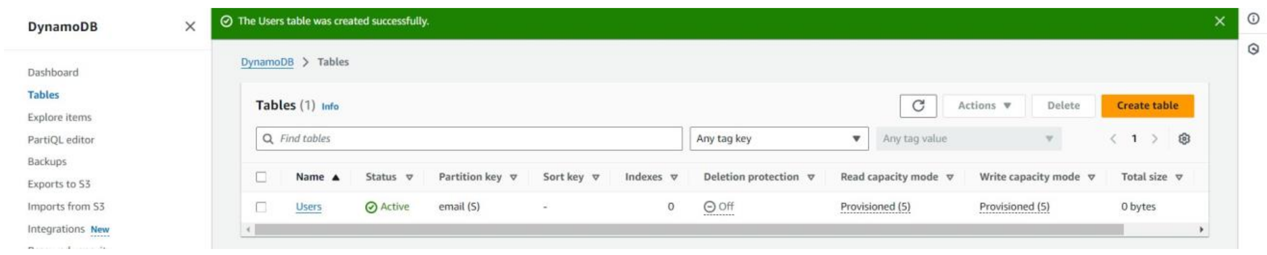
1 to 255 characters and case sensitive.

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.

Enter the sort key name | String ▼

1 to 255 characters and case sensitive.



The Users table was created successfully.

DynamoDB ✕

Dashboard
Tables
Explore items
PartiQL editor
Backups
Exports to S3
Imports from S3
Integrations New

DynamoDB > Tables

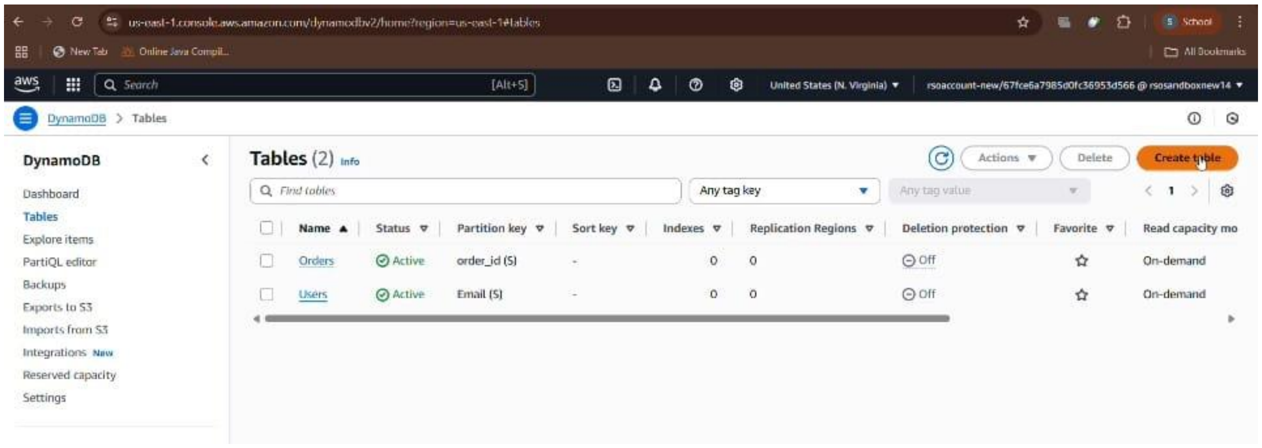
Tables (1) Info

Actions ▼ | Delete | Create table

Find tables | Any tag key | Any tag value | < 1 >

	Name ▲	Status	Partition key	Sort key	Indexes	Deletion protection	Read capacity mode	Write capacity mode	Total size
☐	Users	Active	email (S)	-	0	Off	Provisioned (5)	Provisioned (5)	0 bytes

○ Follow the same steps to create a requests table with E-mail as the primary key for book requests data.



us-east-1.console.aws.amazon.com/dynamodbv2/home?region=us-east-1#tables

DynamoDB > Tables

Tables (2) Info

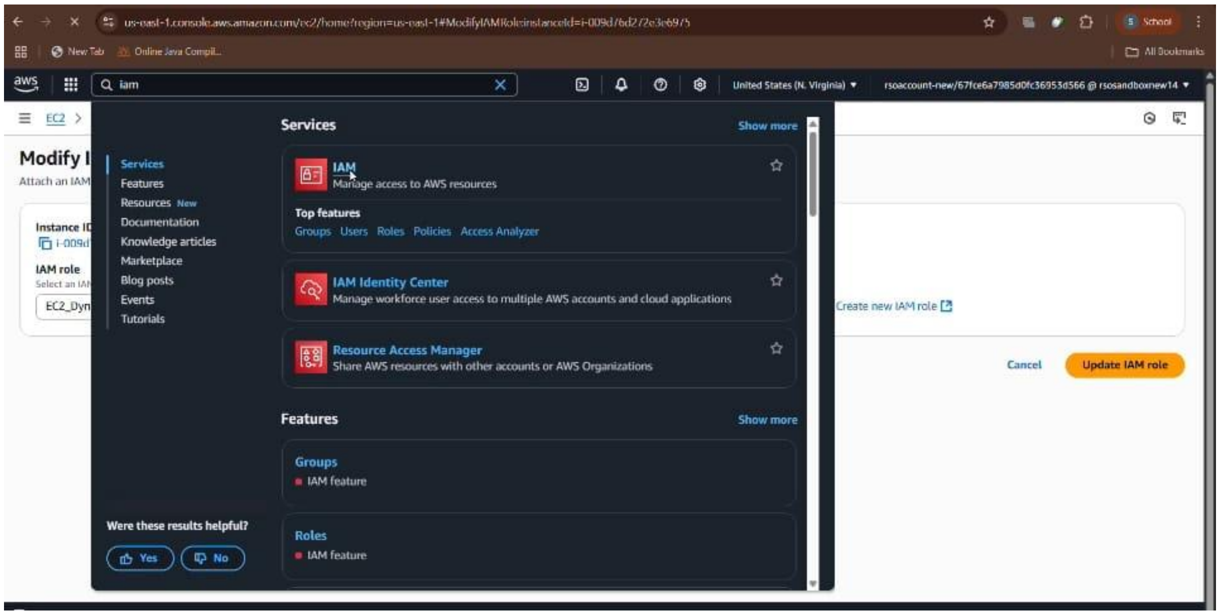
	Name ▲	Status	Partition key	Sort key	Indexes	Replication Regions	Deletion protection	Favorite	Read capacity mo
☐	Orders	Active	order_id (S)	-	0	0	Off	☆	On-demand
☐	Users	Active	Email (S)	-	0	0	Off	☆	On-demand

4: IAM Role Setup

Milestone 4: IAM Role Setup

- **Activity 4.1: Create IAM Role**

  ○ In the AWS Console, navigate to IAM and create a new IAM Role for EC2 to allow interaction with DynamoDB.



us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#ModifyIAMRole:instanceId=i-009d76d272e3e6975

Search: iam

Services

IAM — Manage access to AWS resources
Top features: Groups Users Roles Policies Access Analyzer

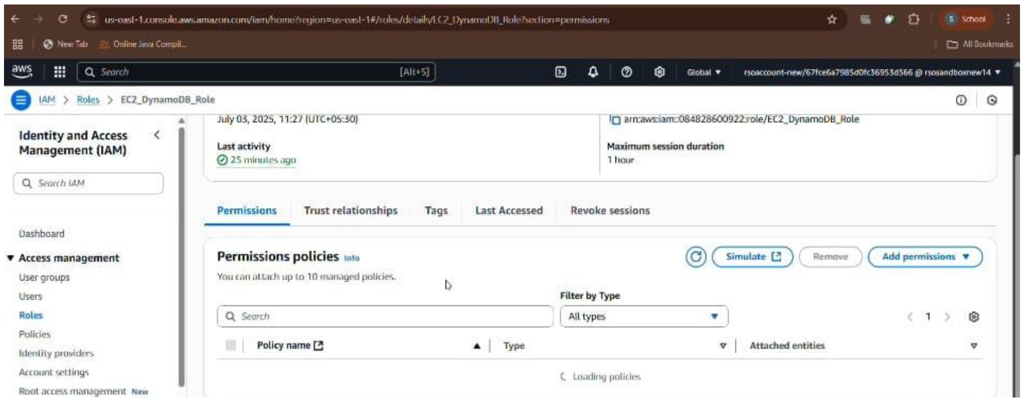
IAM Identity Center — Manage workforce user access to multiple AWS accounts and cloud applications

Resource Access Manager — Share AWS resources with other accounts or AWS Organizations

Features

Groups — IAM feature

Roles — IAM feature



July 03, 2025, 11:27 (UTC+05:30)

Last activity
25 minutes ago

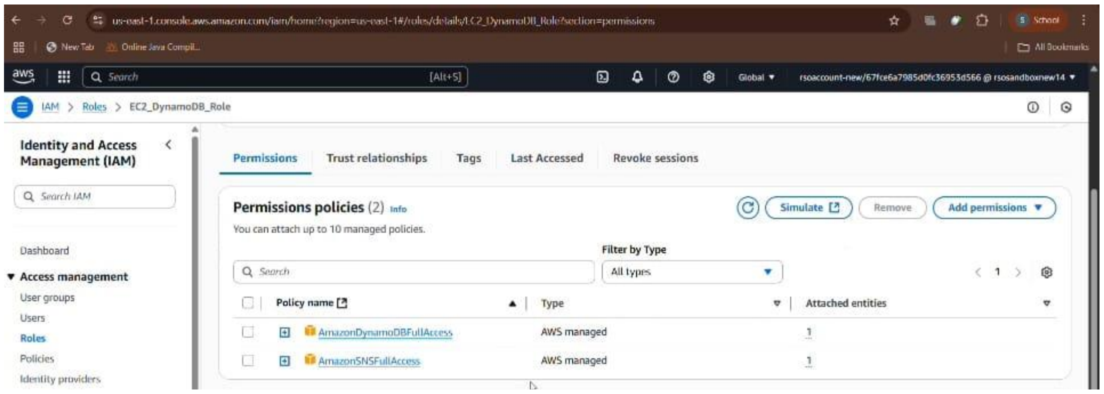
Maximum session duration
1 hour

Permissions policies
You can attach up to 10 managed policies.

Policy name | Type | Attached entities

Loading policies

- **Activity 4.2: Attach Policies**
  - Attach the **AmazonDynamoDBFullAccess** and **AmazonSNSFullAccess** policy to the role. This grants EC2 instances permission to perform read and write operations on DynamoDB.



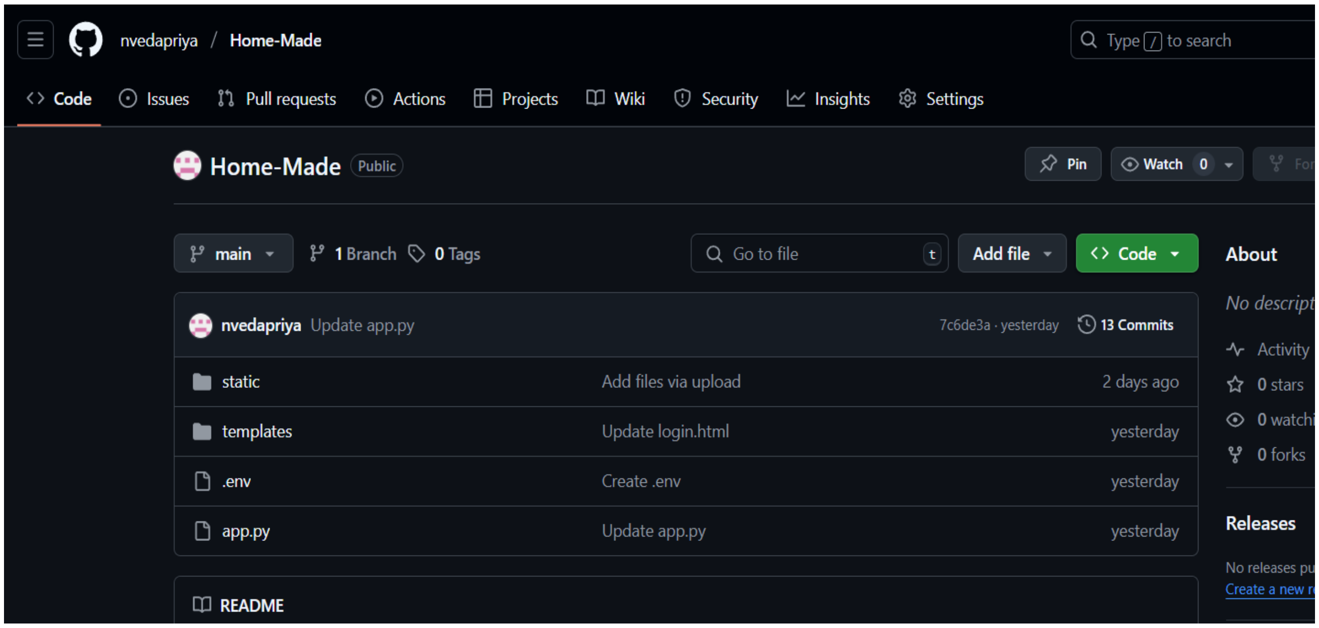
Permissions policies (2)
You can attach up to 10 managed policies.

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

5: EC2 Instance Setup

Milestone 5: EC2 Instance Setup

- **Activity 5.1: Load Project Files to GitHub**
  ○ Upload your Flask application and HTML files to a GitHub repository.
  Note: This will allow easy access and deployment to the EC2 instance.


nvedapriya / Home-Made

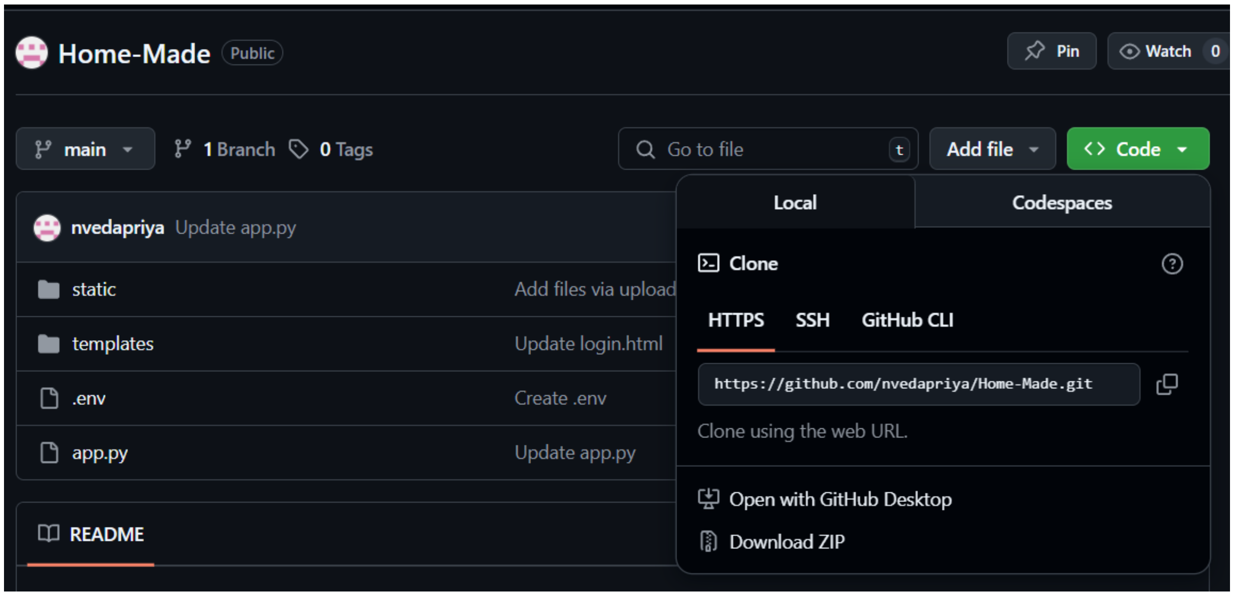
Home-Made Public

main · 1 Branch · 0 Tags

nvedapriya Update app.py · 7c6de3a · yesterday · 13 Commits

static	Add files via upload	2 days ago
templates	Update login.html	yesterday
.env	Create .env	yesterday
app.py	Update app.py	yesterday

README


Home-Made Public

nvedapriya Update app.py

static	Add files via upload
templates	Update login.html
.env	Create .env
app.py	Update app.py

Clone

HTTPS SSH GitHub CLI

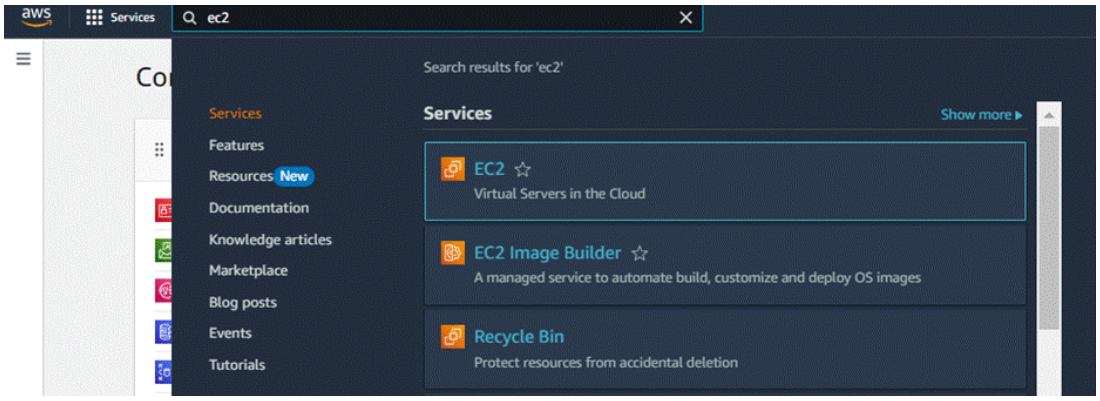
```
https://github.com/nvedapriya/Home-Made.git
```

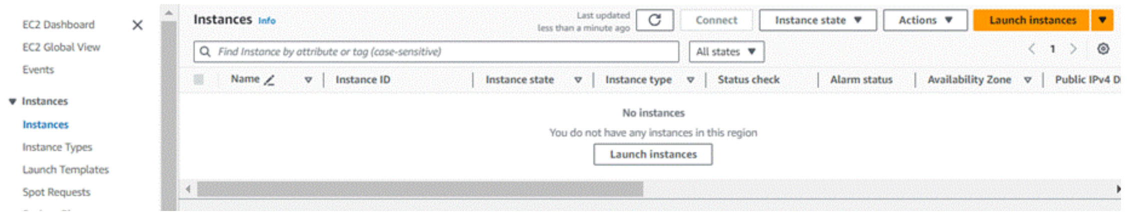
Clone using the web URL.

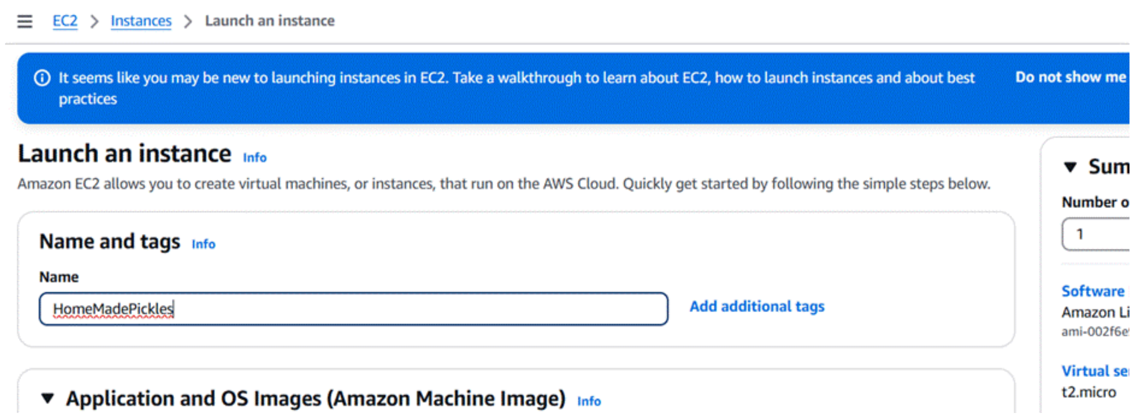
Open with GitHub Desktop

Download ZIP

- **Activity 5.2: Launch an EC2 Instance**
  - In the AWS Console, go to EC2 and click **"Launch Instance"**.
  - Choose **Amazon Linux 2** or **Ubuntu** as the AMI and select **t2.micro** (Free-tier eligible).






EC2 > Instances > Launch an instance

It seems like you may be new to launching instances in EC2. Take a walkthrough to learn about EC2, how to launch instances and about best practices

Do not show me

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

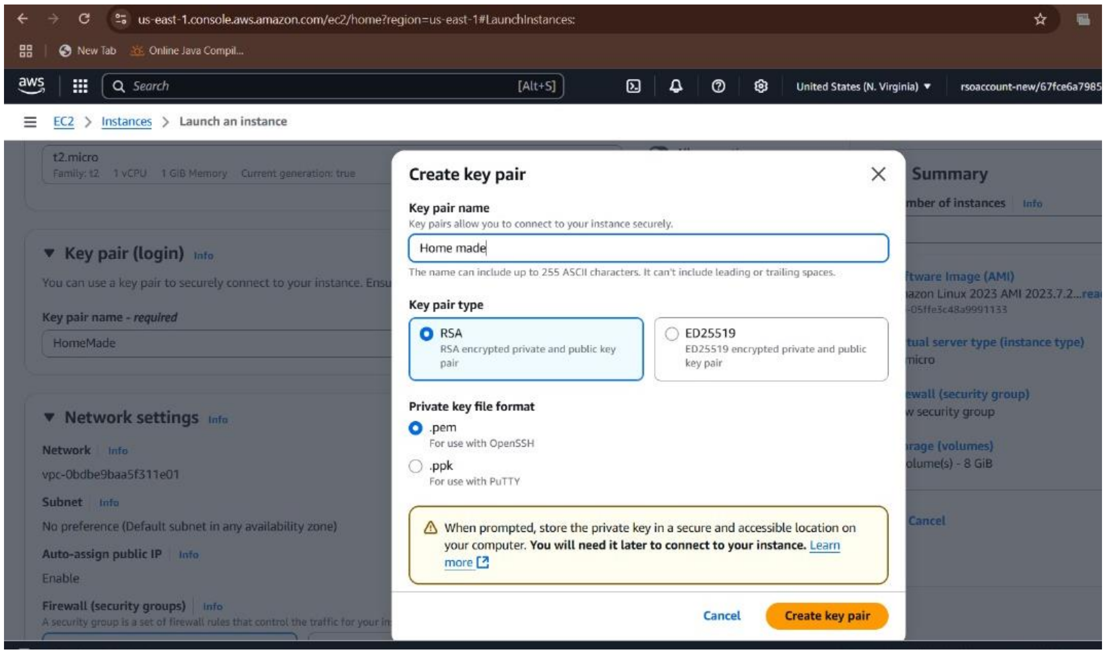
Name: HomeMadePickles

Add additional tags

▼ **Application and OS Images (Amazon Machine Image)** Info

▼ Sum
Number o
1
Software
Amazon Li
ami-002f6e
Virtual se
t2.micro

○ Create and download a **key pair** for secure SSH access.



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

EC2 > Instances > Launch an instance

t2.micro
Family: t2 1 vCPU 1 GiB Memory Current generation: true

Key pair (login) Info
You can use a key pair to securely connect to your instance. Ensu
Key pair name - *required*
HomeMade

Network settings Info
Network Info
vpc-0bdbe9baa5f311e01
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 in

Create key pair

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

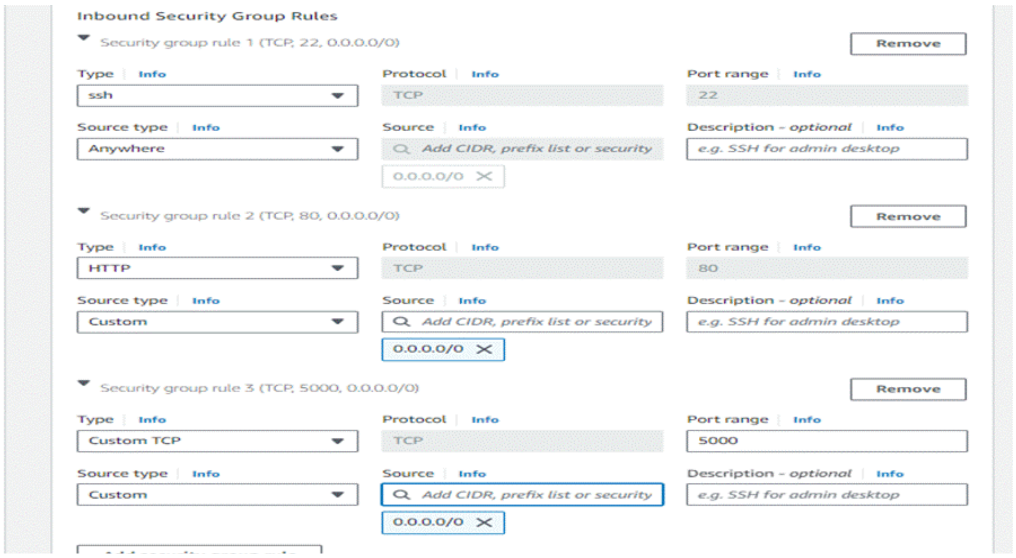
Private key file format
[✓ .pem] For use with OpenSSH
.ppk For use with PuTTY

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

Summary
Number of instances Info
Software Image (AMI)
Amazon Linux 2023 AMI 2023.7.2...read
-05ffe3c48a9991133
Virtual server type (instance type)
t2.micro
Firewall (security group)
New security group
Storage (volumes)
1 volume(s) - 8 GiB

- **Activity 5.3: Configure Security Groups**

  ○ Allow **HTTP (port 80)** and **SSH (port 22)** inbound traffic.

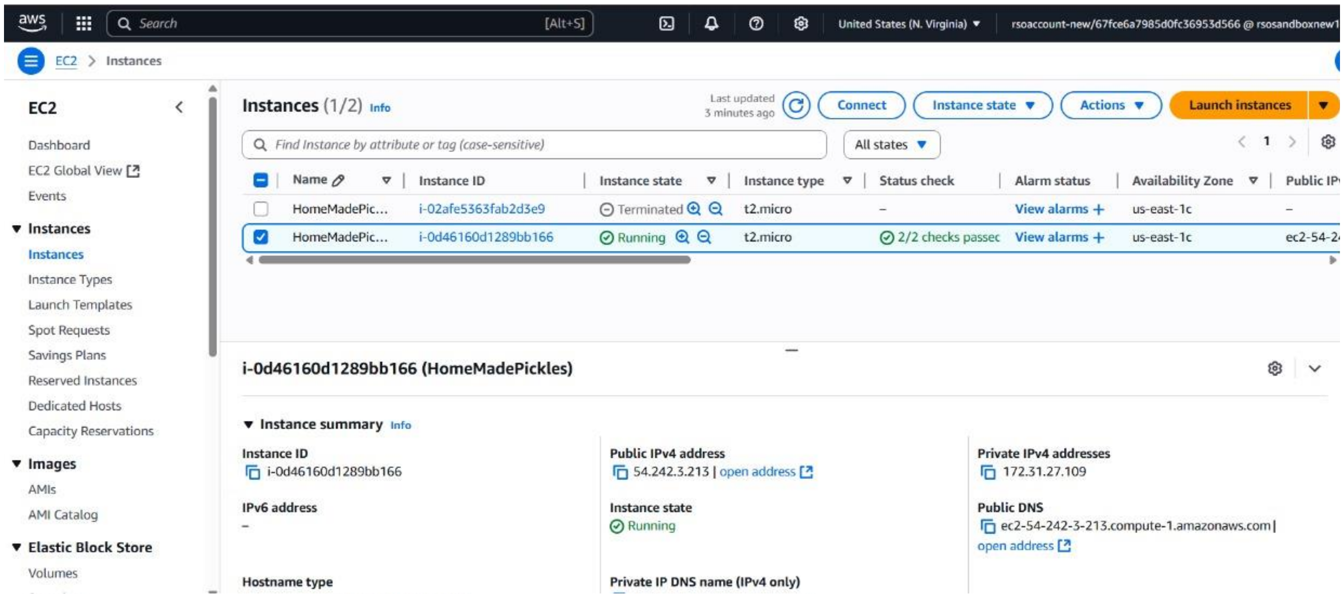


Inbound Security Group Rules

Security group rule 1 (TCP, 22, 0.0.0.0/0)
Type: ssh | Protocol: TCP | Port range: 22
Source type: Anywhere | Source: 0.0.0.0/0 | Description - *optional*: e.g. SSH for admin desktop

Security group rule 2 (TCP, 80, 0.0.0.0/0)
Type: HTTP | Protocol: TCP | Port range: 80
Source type: Custom | Source: 0.0.0.0/0 | Description - *optional*: e.g. SSH for admin desktop

Security group rule 3 (TCP, 5000, 0.0.0.0/0)
Type: Custom TCP | Protocol: TCP | Port range: 5000
Source type: Custom | Source: 0.0.0.0/0 | Description - *optional*: e.g. SSH for admin desktop



EC2 > Instances

Instances (1/2) Info

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IP
HomeMadePic...	i-02afe5363fab2d3e9	Terminated	t2.micro	–	View alarms +	us-east-1c	–
HomeMadePic...	i-0d46160d1289bb166	Running	t2.micro	2/2 checks passec	View alarms +	us-east-1c	ec2-54-2

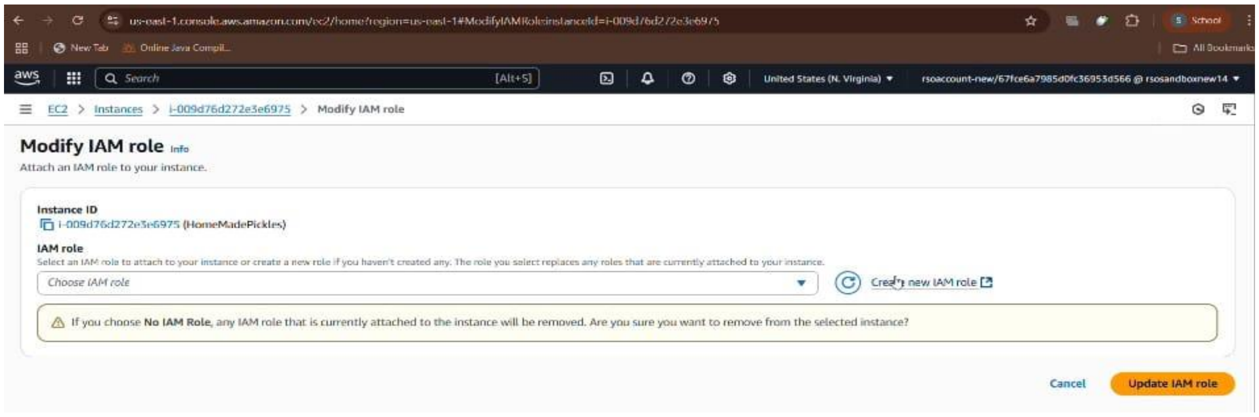
i-0d46160d1289bb166 (HomeMadePickles)

Instance summary Info

Instance ID: i-0d46160d1289bb166	Public IPv4 address: 54.242.3.213 | open address	Private IPv4 addresses: 172.31.27.109
IPv6 address: –	Instance state: Running	Public DNS: ec2-54-242-3-213.compute-1.amazonaws.com | open address
Hostname type	Private IP DNS name (IPv4 only)	

- **Activity 5.4: Attach IAM Role**

  ○ Attach the IAM Role created earlier to your EC2 instance by selecting your instance → **Actions → Security → Modify IAM Role**.



EC2 > Instances > i-009d76d272e3e6975 > Modify IAM role

Modify IAM role Info

Attach an IAM role to your instance.

Instance ID
i-009d76d272e3e6975 (HomeMadePickles)

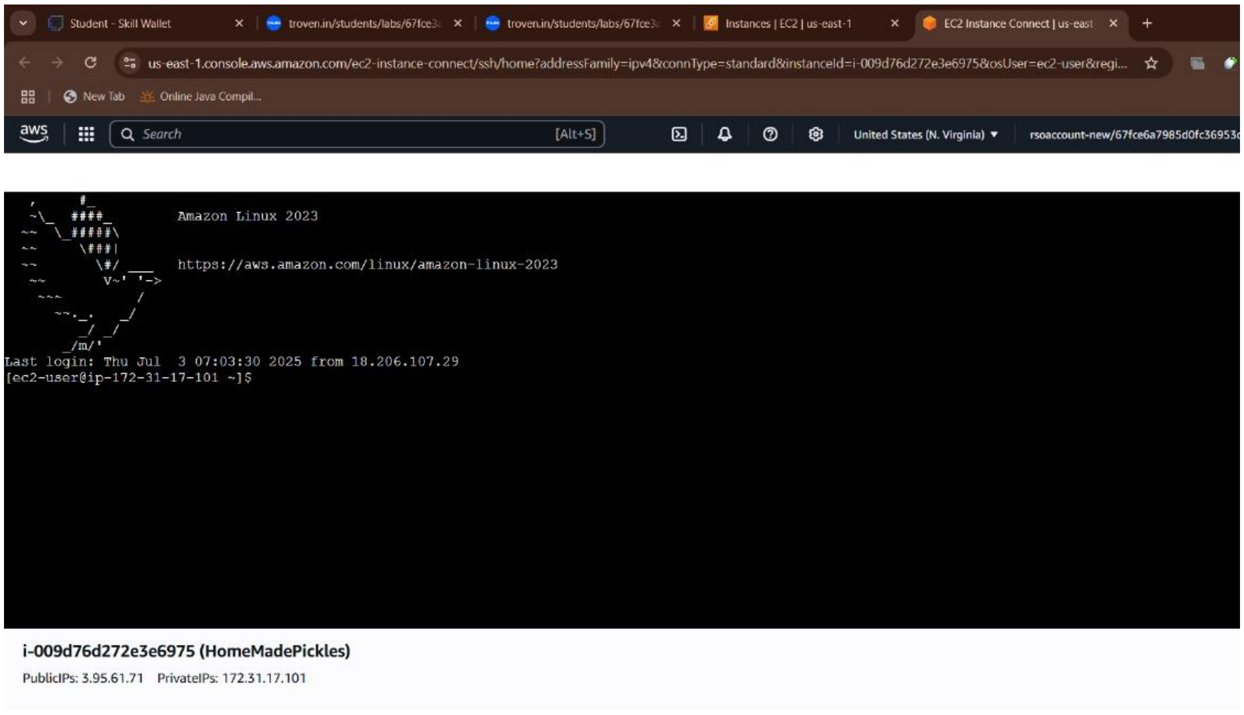
IAM role
Select an IAM role to attach to your instance or create a new role if you haven't created any. The role you select replaces any roles that are currently attached to your instance.

Choose IAM role

If you choose **No IAM Role**, any IAM role that is currently attached to the instance will be removed. Are you sure you want to remove from the selected instance?

- **Activity 5.5: Connect to EC2 Instance**

  ○ Use **EC2 Instance Connect** via AWS Console to open a terminal session.



```
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
Last login: Thu Jul  3 07:03:30 2025 from 18.206.107.29
[ec2-user@ip-172-31-17-101 ~]$
```

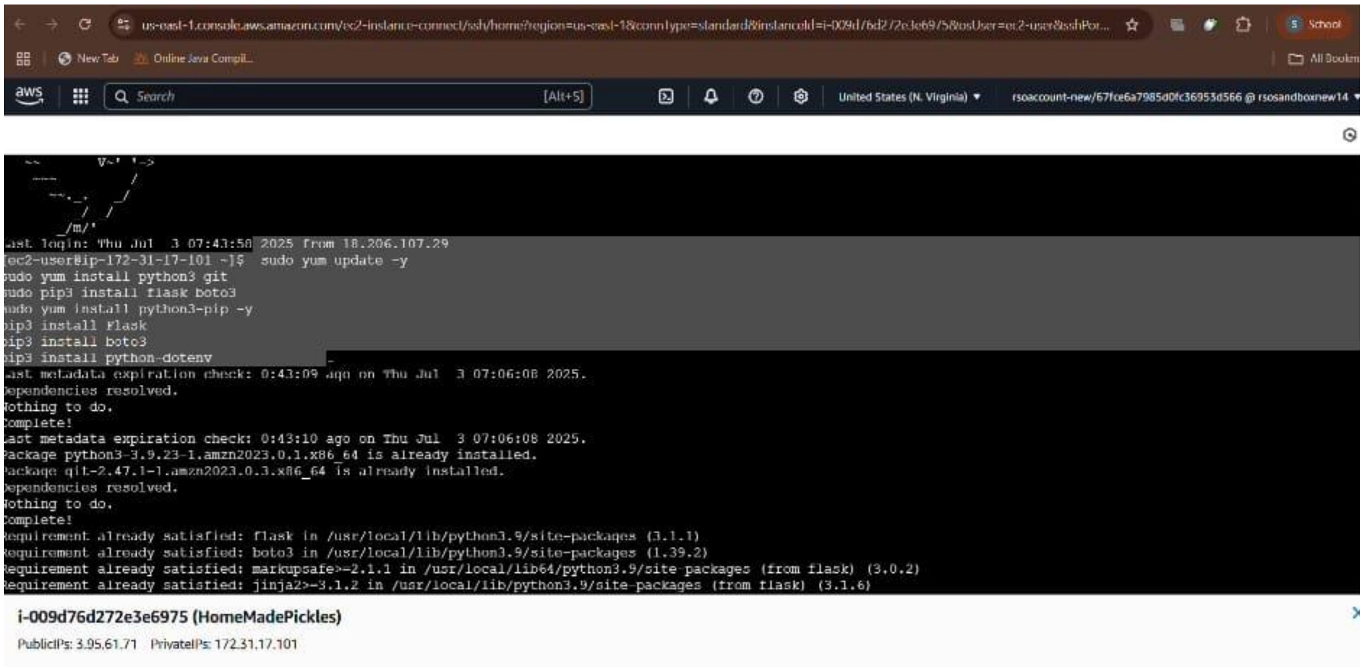
i-009d76d272e3e6975 (HomeMadePickles)

PublicIPs: 3.95.61.71 PrivateIPs: 172.31.17.101

6: Deployment on EC2

Milestone 6: Deployment on EC2

- **Activity 6.1: Install Required Software**
  - Run the following commands to install necessary packages:
    sudo yum update -y
    sudo yum install python3 git
    sudo pip3 install flask boto3
  - Verify installations:
    bash
    Copy code
    flask --version
    git --version

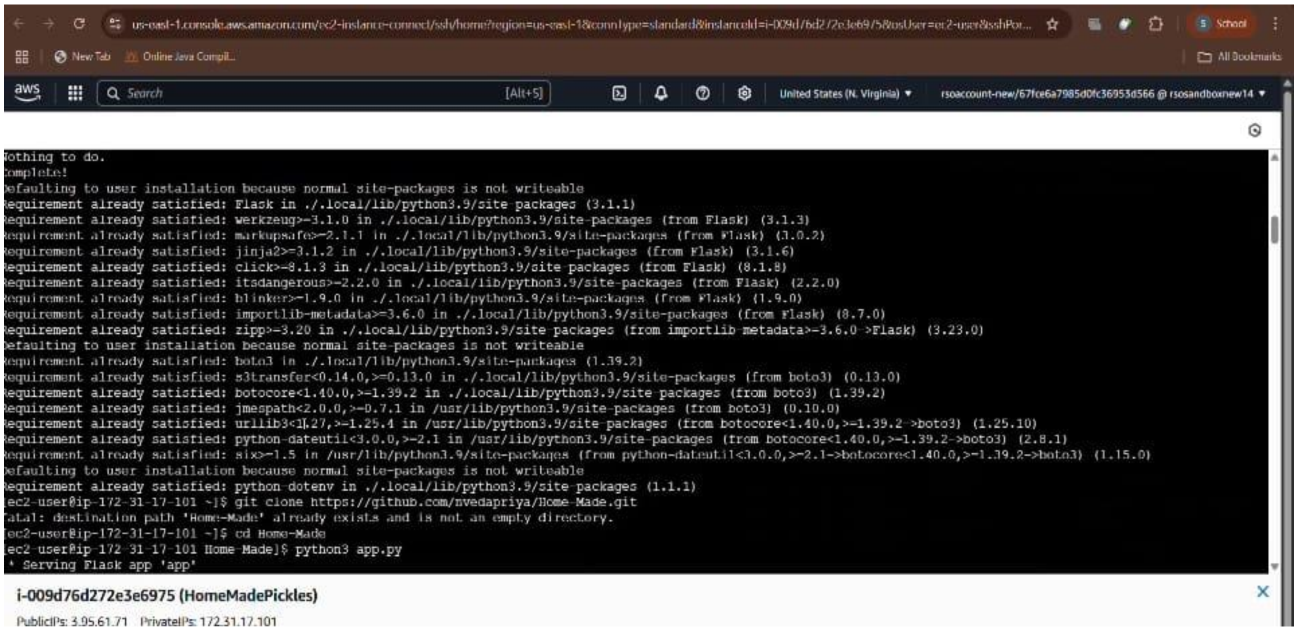


```
ast login: Thu Jul  3 07:43:58 2025 from 18.206.107.29
ec2-user@ip-172-31-17-101 ~]$  sudo yum update -y
udo yum install python3 git
udo pip3 install flask boto3
udo yum install python3-pip -y
ip3 install Flask
ip3 install boto3
ip3 install python-dotenv
ast metadata expiration check: 0:43:09 ago on Thu Jul  3 07:06:08 2025.
ependencies resolved.
othing to do.
omplete!
ast metadata expiration check: 0:43:10 ago on Thu Jul  3 07:06:08 2025.
ackage python3-3.9.23-1.amzn2023.0.1.x86_64 is already installed.
ackage git-2.47.1-1.amzn2023.0.3.x86_64 is already installed.
ependencies resolved.
othing to do.
omplete!
equirement already satisfied: flask in /usr/local/lib/python3.9/site-packages (3.1.1)
equirement already satisfied: boto3 in /usr/local/lib/python3.9/site-packages (1.39.2)
equirement already satisfied: markupsafe>=2.1.1 in /usr/local/lib64/python3.9/site-packages (from flask) (3.0.2)
equirement already satisfied: jinja2>=3.1.2 in /usr/local/lib/python3.9/site-packages (from flask) (3.1.6)
```

i-009d76d272e3e6975 (HomeMadePickles)

PublicIPs: 3.95.61.71 PrivateIPs: 172.31.17.101

- **Activity 6.2: Clone Flask Project from GitHub**
  - Run: git clone https://github.com/nvedapriya/Home-Made.git
  - Navigate to the project folder: cd **Home-Made**

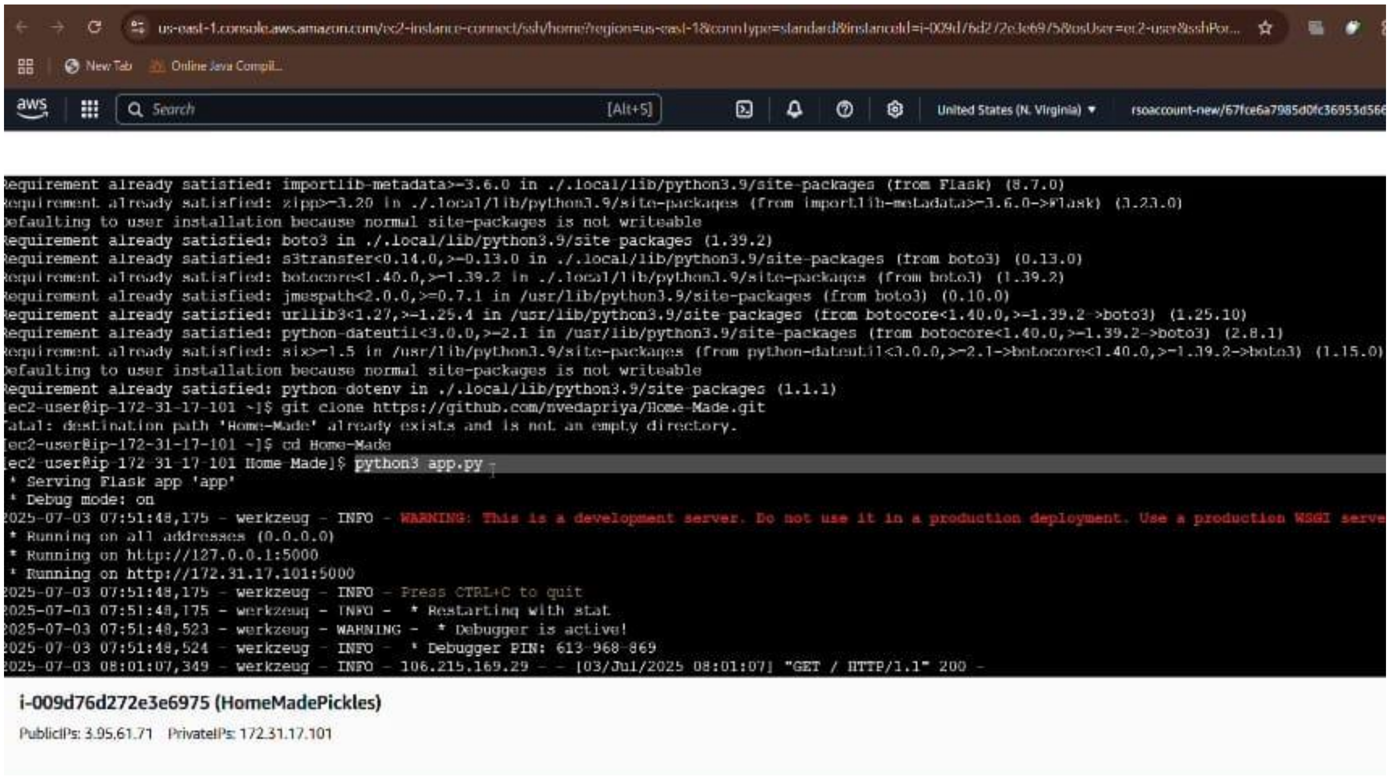


```
othing to do.
omplete!
efaulting to user installation because normal site-packages is not writeable
equirement already satisfied: Flask in ./.local/lib/python3.9/site packages (3.1.1)
equirement already satisfied: werkzeug>=3.1.0 in ./.local/lib/python3.9/site-packages (from Flask) (3.1.3)
equirement already satisfied: markupsafe>=2.1.1 in ./.local/lib/python3.9/site-packages (from Flask) (3.0.2)
equirement already satisfied: jinja2>=3.1.2 in ./.local/lib/python3.9/site-packages (from Flask) (3.1.6)
equirement already satisfied: click>=8.1.3 in ./.local/lib/python3.9/site packages (from Flask) (8.1.8)
equirement already satisfied: itsdangerous>=2.2.0 in ./.local/lib/python3.9/site-packages (from Flask) (2.2.0)
equirement already satisfied: blinker>=1.9.0 in ./.local/lib/python3.9/site-packages (from Flask) (1.9.0)
equirement already satisfied: importlib-metadata>=3.6.0 in ./.local/lib/python3.9/site-packages (from Flask) (8.7.0)
equirement already satisfied: zipp>=3.20 in ./.local/lib/python3.9/site packages (from importlib metadata>=3.6.0 >Flask) (3.23.0)
efaulting to user installation because normal site-packages is not writeable
equirement already satisfied: boto3 in ./.local/lib/python3.9/site-packages (1.39.2)
equirement already satisfied: s3transfer<0.14.0,>=0.13.0 in ./.local/lib/python3.9/site-packages (from boto3) (0.13.0)
equirement already satisfied: botocore<1.40.0,>=1.39.2 in ./.local/lib/python3.9/site packages (from boto3) (1.39.2)
equirement already satisfied: jmespath<2.0.0,>=0.7.1 in /usr/lib/python3.9/site-packages (from boto3) (0.10.0)
equirement already satisfied: urllib3<1.27,>=1.25.4 in /usr/lib/python3.9/site packages (from botocore<1.40.0,>=1.39.2->boto3) (1.25.10)
equirement already satisfied: python-dateutil<3.0.0,>=2.1 in /usr/lib/python3.9/site-packages (from botocore<1.40.0,>=1.39.2->boto3) (2.8.1)
equirement already satisfied: six>=1.5 in /usr/lib/python3.9/site-packages (from python-dateutil<3.0.0,>=2.1->botocore<1.40.0,>=1.39.2->boto3) (1.15.0)
efaulting to user installation because normal site-packages is not writeable
equirement already satisfied: python-dotenv in ./.local/lib/python3.9/site-packages (1.1.1)
ec2-user@ip-172-31-17-101 ~]$ git clone https://github.com/nvedapriya/Home-Made.git
atal: destination path 'Home-Made' already exists and is not an empty directory.
ec2-user@ip-172-31-17-101 ~]$ cd Home-Made
ec2-user@ip-172-31-17-101 Home-Made]$ python3 app.py
 * Serving Flask app 'app'
```

i-009d76d272e3e6975 (HomeMadePickles)

PublicIPs: 3.95.61.71 PrivateIPs: 172.31.17.101

- **Activity 6.3: Run the Flask Application**
  - Run: python3 app.py



```
equirement already satisfied: importlib-metadata>=3.6.0 in ./.local/lib/python3.9/site-packages (from Flask) (8.7.0)
equirement already satisfied: zipp>=3.20 in ./.local/lib/python3.9/site-packages (from importlib-metadata>=3.6.0->Flask) (3.23.0)
efaulting to user installation because normal site-packages is not writeable
equirement already satisfied: boto3 in ./.local/lib/python3.9/site-packages (1.39.2)
equirement already satisfied: s3transfer<0.14.0,>=0.13.0 in ./.local/lib/python3.9/site-packages (from boto3) (0.13.0)
equirement already satisfied: botocore<1.40.0,>=1.39.2 in ./.local/lib/python3.9/site-packages (from boto3) (1.39.2)
equirement already satisfied: jmespath<2.0.0,>=0.7.1 in /usr/lib/python3.9/site-packages (from boto3) (0.10.0)
equirement already satisfied: urllib3<1.27,>=1.25.4 in /usr/lib/python3.9/site-packages (from botocore<1.40.0,>=1.39.2->boto3) (1.25.10)
equirement already satisfied: python-dateutil<3.0.0,>=2.1 in /usr/lib/python3.9/site-packages (from botocore<1.40.0,>=1.39.2->boto3) (2.8.1)
equirement already satisfied: six>=1.5 in /usr/lib/python3.9/site-packages (from python-dateutil<3.0.0,>=2.1->botocore<1.40.0,>=1.39.2->boto3) (1.15.0)
efaulting to user installation because normal site-packages is not writeable
equirement already satisfied: python-dotenv in ./.local/lib/python3.9/site-packages (1.1.1)
ec2-user@ip-172-31-17-101 ~]$ git clone https://github.com/nvedapriya/Home-Made.git
atal: destination path 'Home-Made' already exists and is not an empty directory.
ec2-user@ip-172-31-17-101 ~]$ cd Home-Made
ec2-user@ip-172-31-17-101 Home-Made]$ python3 app.py
 * Serving Flask app 'app'
 * Debug mode: on
025-07-03 07:51:48,175 - werkzeug - INFO - WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI serve
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5000
 * Running on http://172.31.17.101:5000
025-07-03 07:51:48,175 - werkzeug - INFO - Press CTRL+C to quit
025-07-03 07:51:48,175 - werkzeug - INFO -  * Restarting with stat
025-07-03 07:51:48,523 - werkzeug - WARNING -  * Debugger is active!
025-07-03 07:51:48,524 - werkzeug - INFO -  * Debugger PIN: 613-968-869
025-07-03 08:01:07,349 - werkzeug - INFO - 106.215.169.29 - - [03/Jul/2025 08:01:07] "GET / HTTP/1.1" 200 -
```

i-009d76d272e3e6975 (HomeMadePickles)

PublicIPs: 3.95.61.71 PrivateIPs: 172.31.17.101

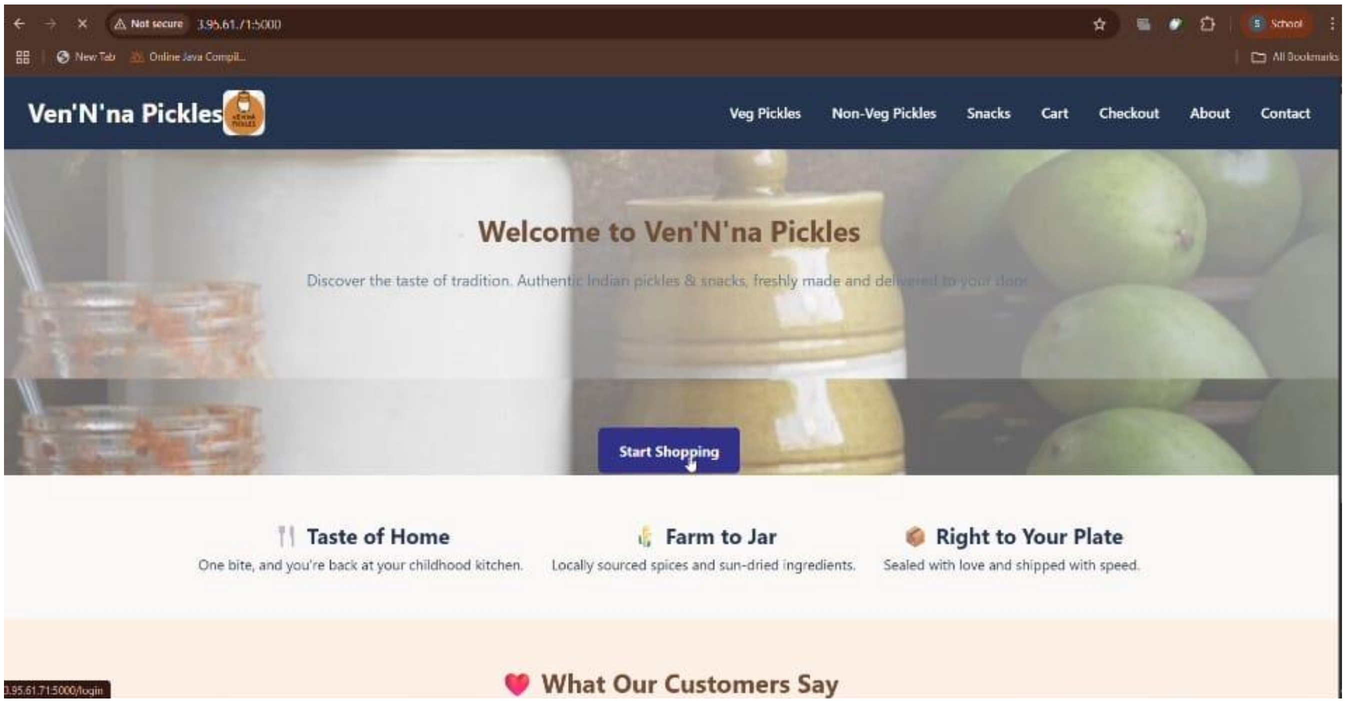
- **Activity 6.6: Access the Website**
  - Open your browser and go to: **http://3.95.61.71:5000**

7: Testing and Deployment

Milestone 7: Testing and Deployment

- **Activity 7.1: Functional Testing to Verify the Project**
  - Test each of the following pages for proper functionality, navigation, and data flow:

Home Page:



Welcome to Ven'N'na Pickles

Discover the taste of tradition. Authentic Indian pickles & snacks, freshly made and delivered to your door.

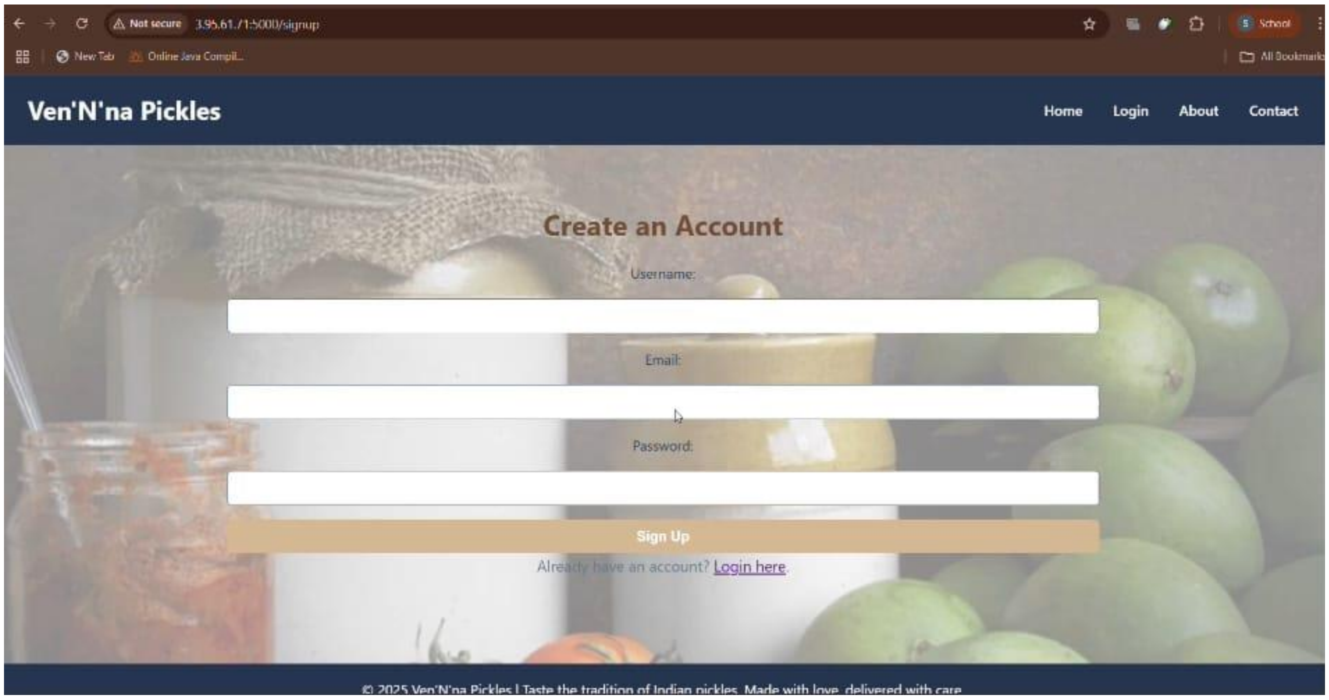
Taste of Home
One bite, and you're back at your childhood kitchen.

Farm to Jar
Locally sourced spices and sun-dried ingredients.

Right to Your Plate
Sealed with love and shipped with speed.

What Our Customers Say

Signup Page:



Create an Account

Username:

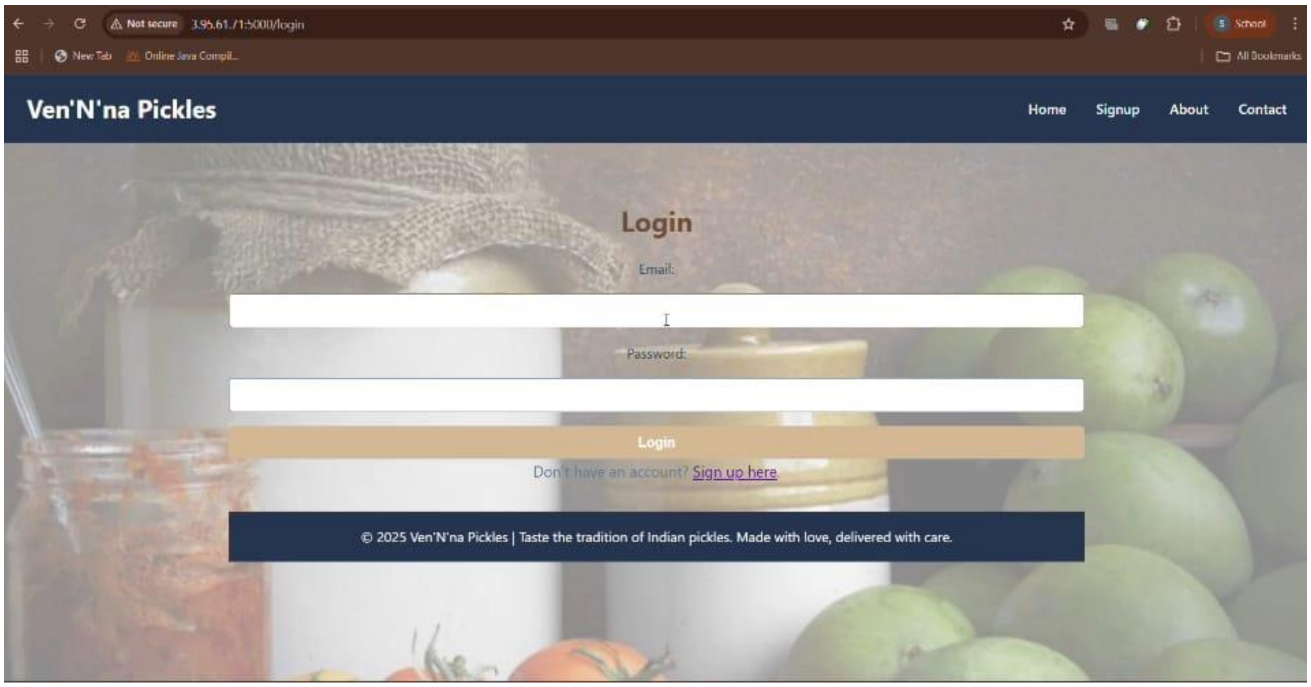
Email:

Password:

Already have an account? Login here.

© 2025 Ven'N'na Pickles | Taste the tradition of Indian pickles. Made with love, delivered with care.

Login Page:



Login

Email:

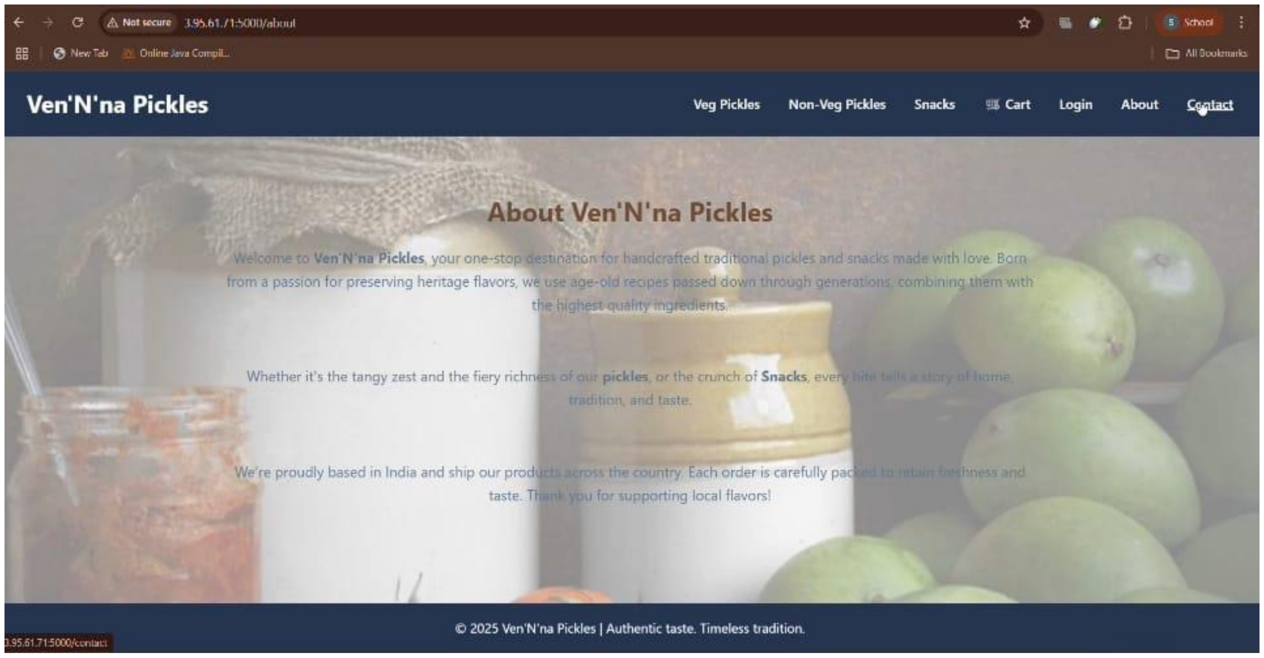
Password:

Login

Don't have an account? Sign up here

© 2025 Ven'N'na Pickles | Taste the tradition of Indian pickles. Made with love, delivered with care.

About Page:



About Ven'N'na Pickles

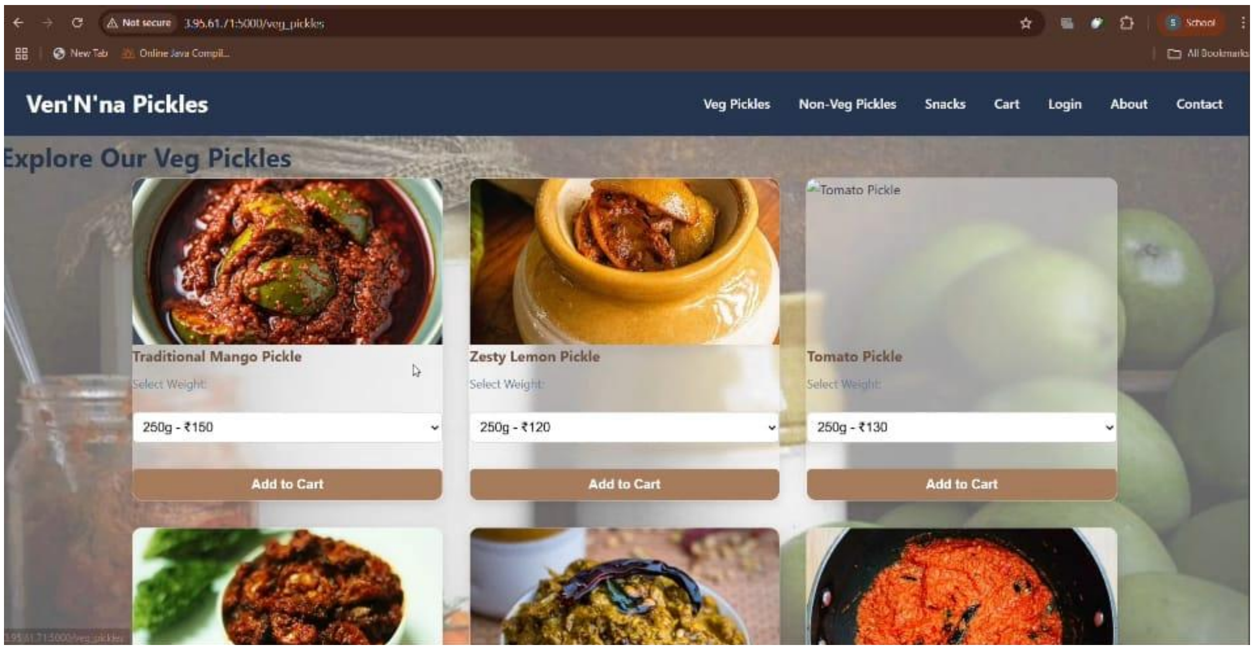
Welcome to **Ven'N'na Pickles**, your one-stop destination for handcrafted traditional pickles and snacks made with love. Born from a passion for preserving heritage flavors, we use age-old recipes passed down through generations, combining them with the highest quality ingredients.

Whether it's the tangy zest and the fiery richness of our **pickles**, or the crunch of **Snacks**, every bite tells a story of home, tradition, and taste.

We're proudly based in India and ship our products across the country. Each order is carefully packed to retain freshness and taste. Thank you for supporting local flavors!

© 2025 Ven'N'na Pickles | Authentic taste. Timeless tradition.

Veg Pickles Page:

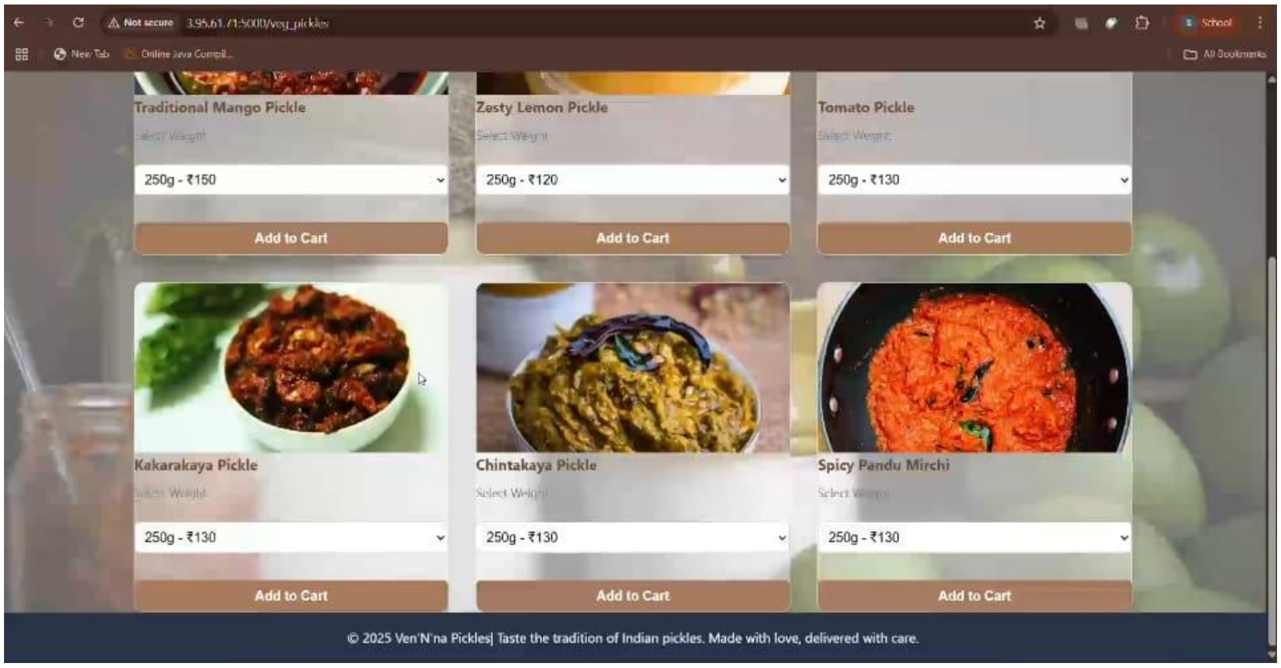


Explore Our Veg Pickles

Traditional Mango Pickle
Select Weight:
250g - ₹150

Zesty Lemon Pickle
Select Weight:
250g - ₹120

Tomato Pickle
Select Weight:
250g - ₹130



Traditional Mango Pickle
Select Weight:
250g - ₹150

Zesty Lemon Pickle
Select Weight:
250g - ₹120

Tomato Pickle
Select Weight:
250g - ₹130

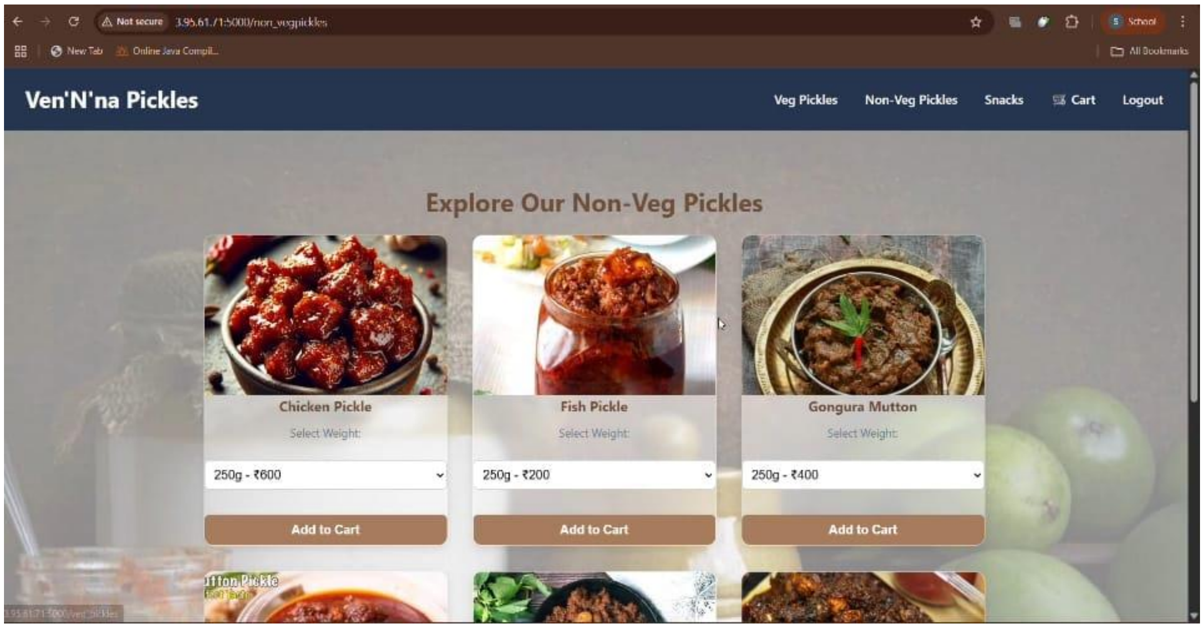
Kakarakaya Pickle
Select Weight:
250g - ₹130

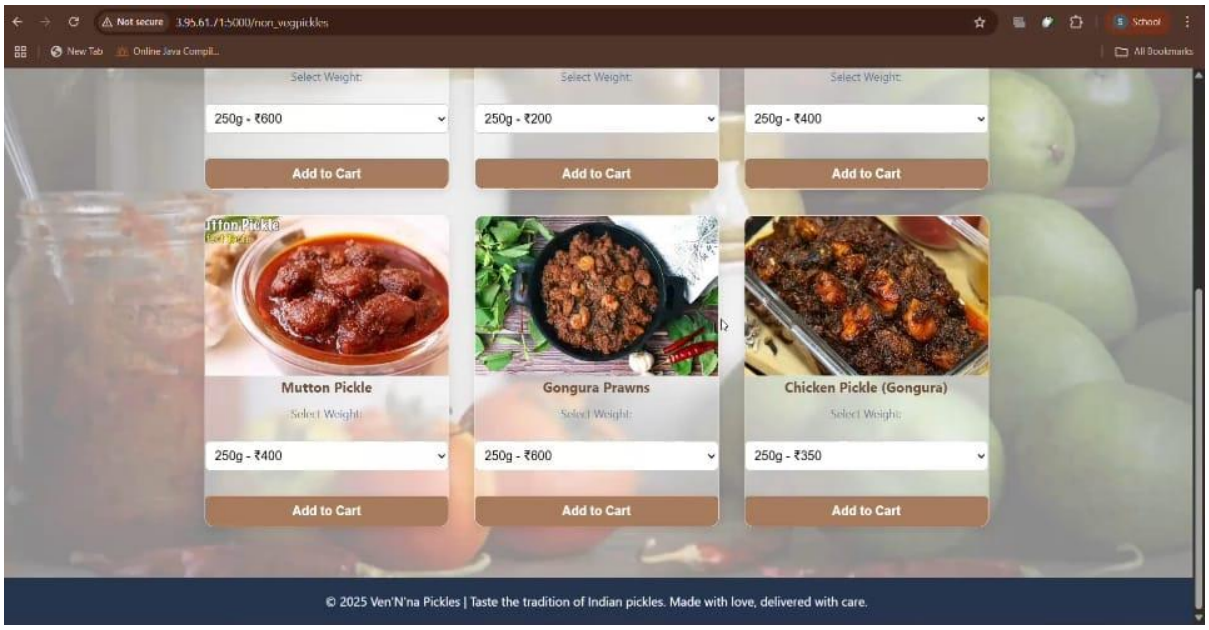
Chintakaya Pickle
Select Weight:
250g - ₹130

Spicy Pandu Mirchi
Select Weight:
250g - ₹130

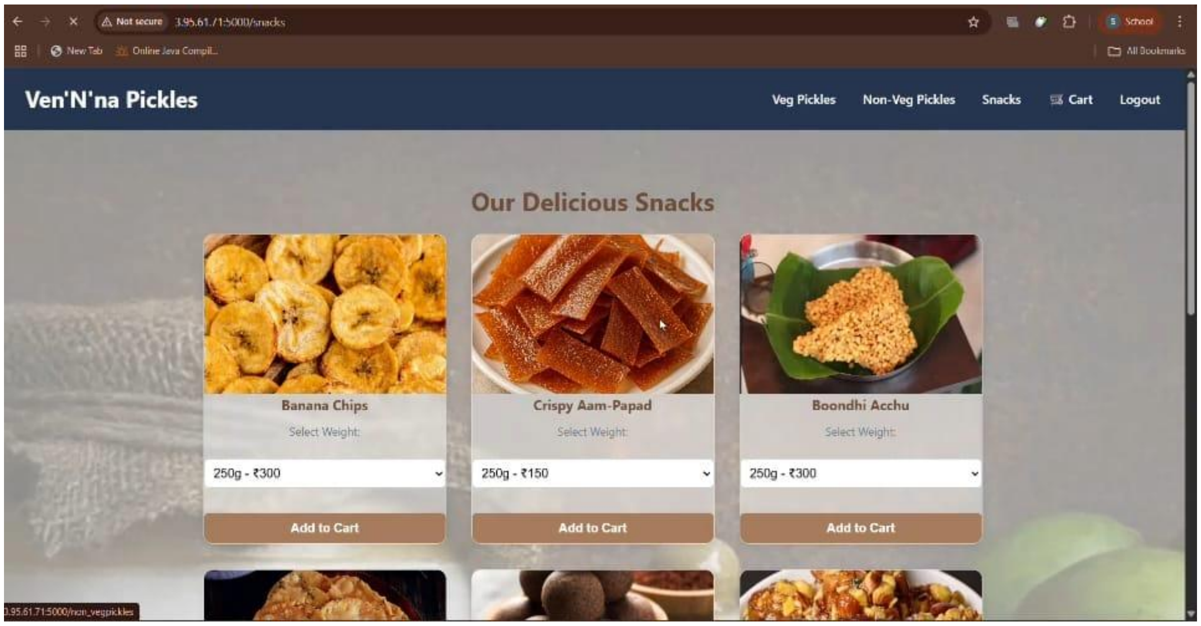
© 2025 Ven'N'na Pickles| Taste the tradition of Indian pickles. Made with love, delivered with care.

Non-Veg Pickles Page:



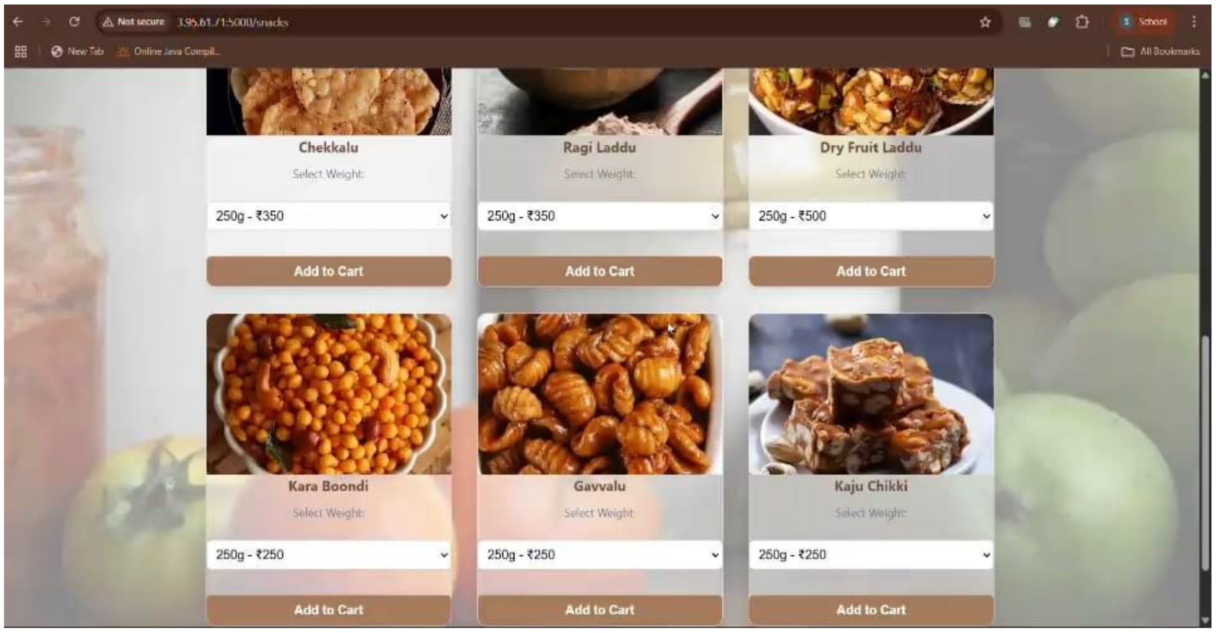


Snacks page:



Our Delicious Snacks

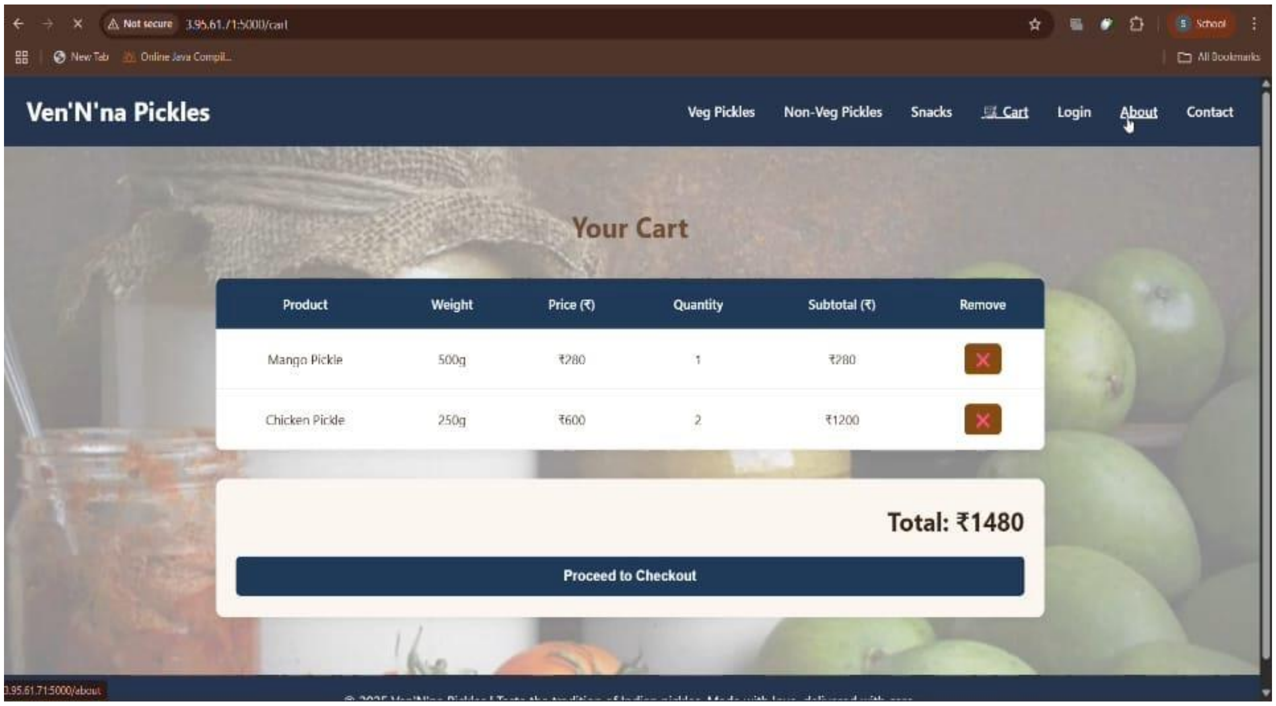
Banana Chips	Crispy Aam-Papad	Boondhi Acchu
Select Weight:	Select Weight:	Select Weight:
250g - ₹300	250g - ₹150	250g - ₹300
Add to Cart	Add to Cart	Add to Cart



Chekkalu	Ragi Laddu	Dry Fruit Laddu
Select Weight:	Select Weight:	Select Weight:
250g - ₹350	250g - ₹350	250g - ₹500
Add to Cart	Add to Cart	Add to Cart

Kara Boondi	Gavvalu	Kaju Chikki
Select Weight:	Select Weight:	Select Weight:
250g - ₹250	250g - ₹250	250g - ₹250
Add to Cart	Add to Cart	Add to Cart

Cart Page:



Ven'N'na Pickles

Veg Pickles | Non-Veg Pickles | Snacks | Cart | Login | About | Contact

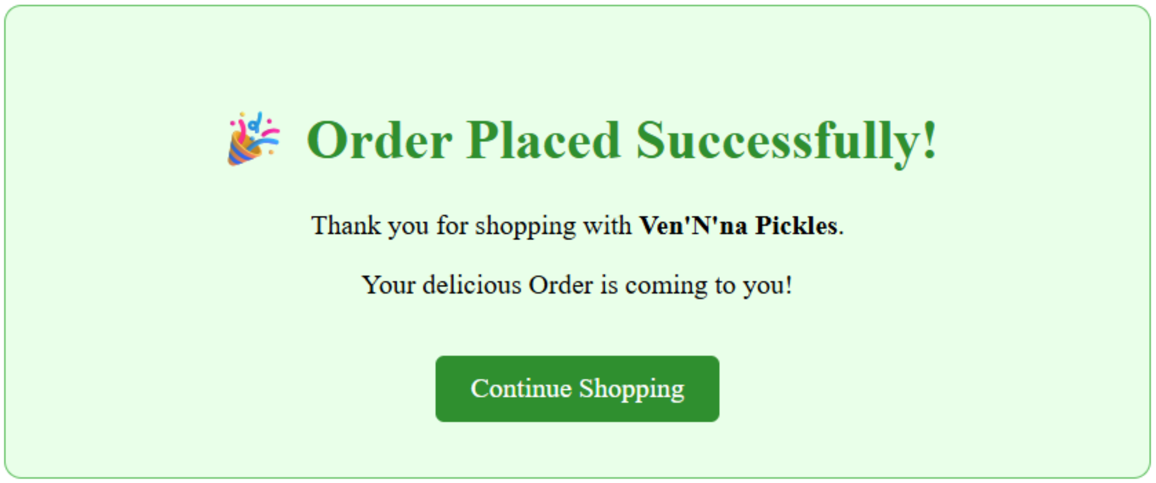
Your Cart

Product	Weight	Price (₹)	Quantity	Subtotal (₹)	Remove
Mango Pickle	500g	₹280	1	₹280	✕
Chicken Pickle	250g	₹600	2	₹1200	✕

Total: ₹1480

Proceed to Checkout

Success Page:



🎉 Order Placed Successfully!

Thank you for shopping with **Ven'N'na Pickles**.

Your delicious Order is coming to you!

Continue Shopping