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.

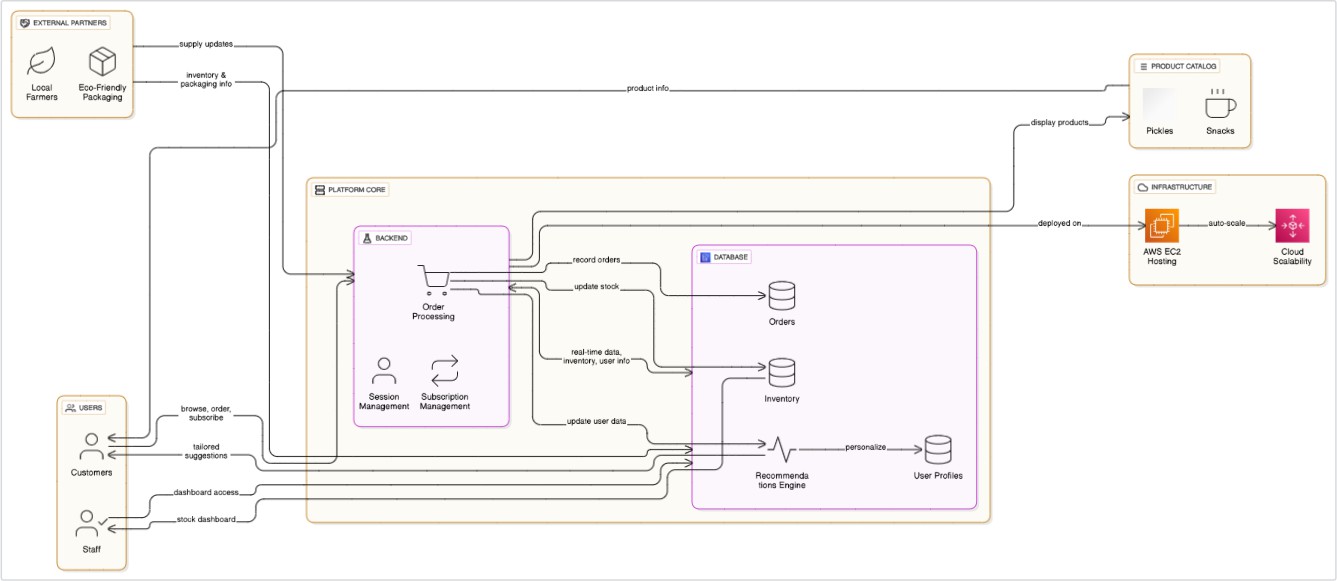
## Scenario 2: Real-Time Inventory Tracking and Updates

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.

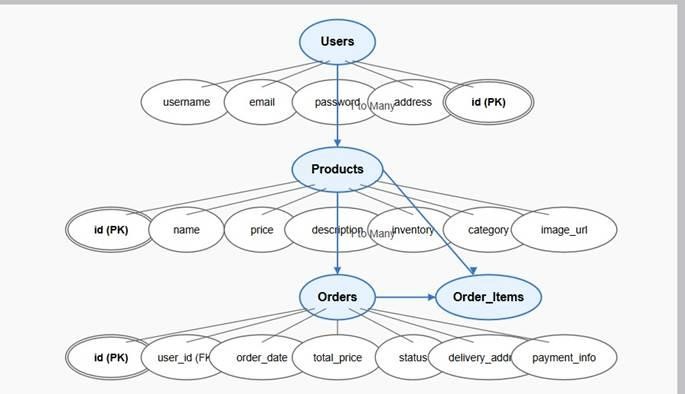
## Scenario 3: Personalized User Experience and Recommendations

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](https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/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

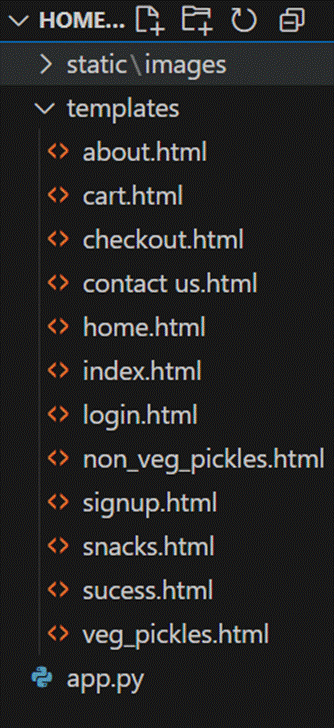
**1: Web Application Development and Setup** **Milestone 1: Web Application Development and Setup**

## 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.

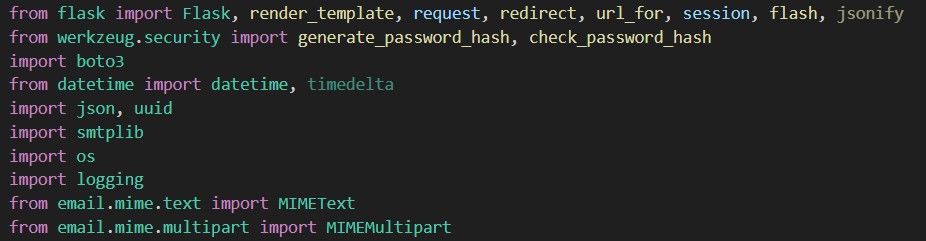
* + Implement core features like user registration, login, and data submission using local storage. Ensure smooth navigation between pages and basic input validation on both frontend and backend.



**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

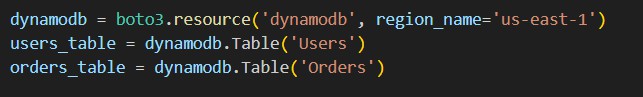
****

**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.



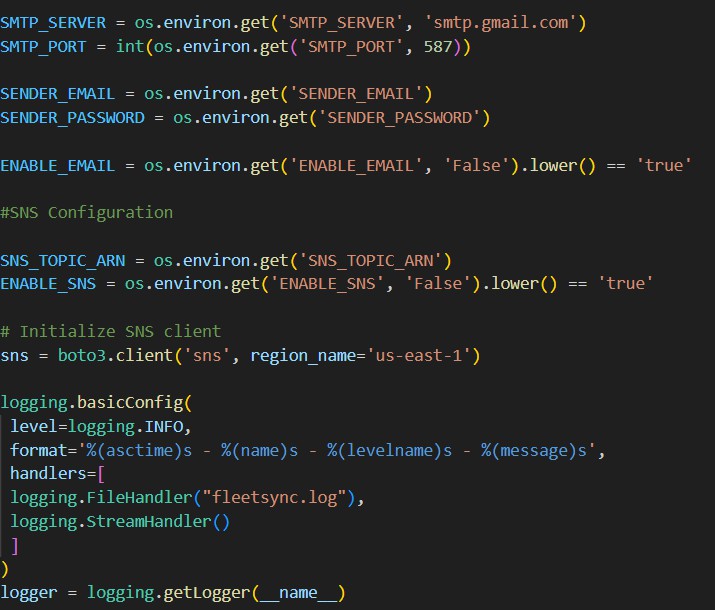
**Description:** initialize the Flask application instance using Flask(name) to start building the web app.

## Dynamodb Setup:

****

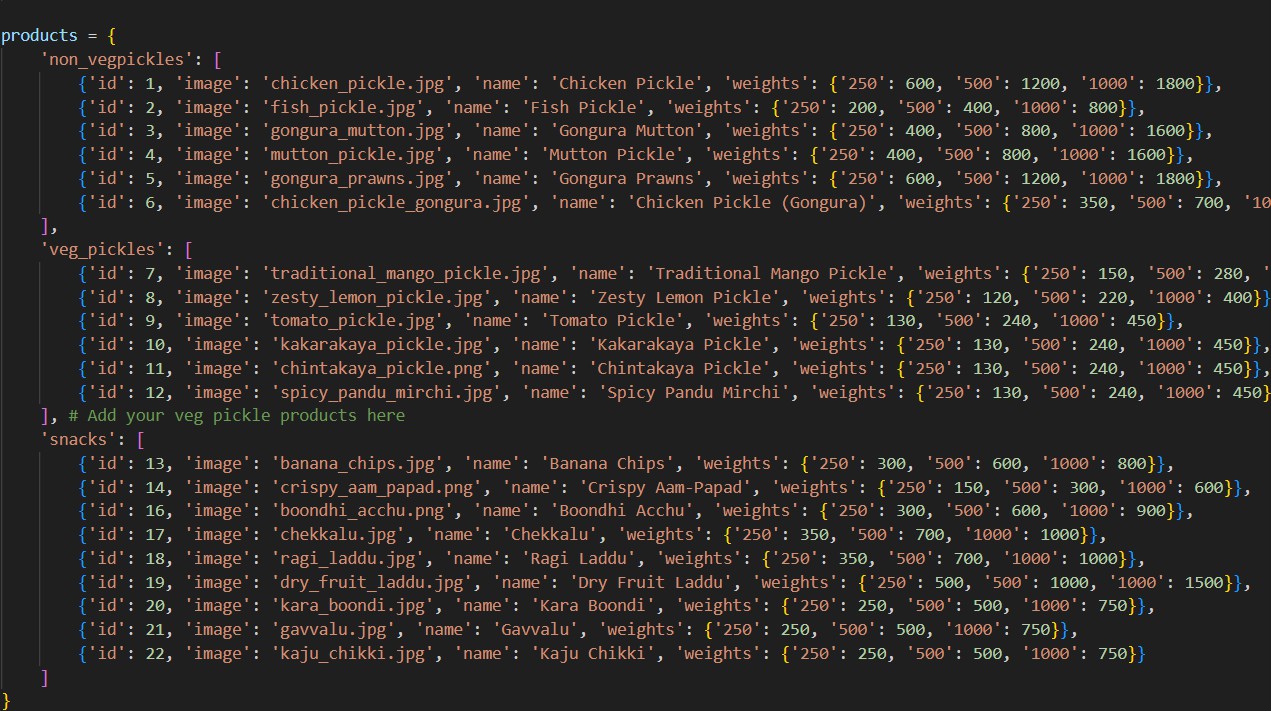
**Description:** initialize the DynamoDB resource for the us-east-1 region and set up access to the Users and Orders tables for storing user details and Orders requests.

## SNS Connection

****

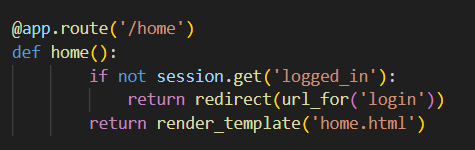
**Description:** Configure **SNS** to send notifications when a book request is submitted. Paste your stored ARN link in the sns\_topic\_arn space, along with the region\_name where the SNS topic is created. Also, specify the chosen email service in SMTP\_SERVER (e.g., Gmail, Yahoo, etc.) and enter the subscribed email in the SENDER\_EMAIL section. Create an ‘App password’ for the email ID and store it in the SENDER\_PASSWORD section.

* + **Products**

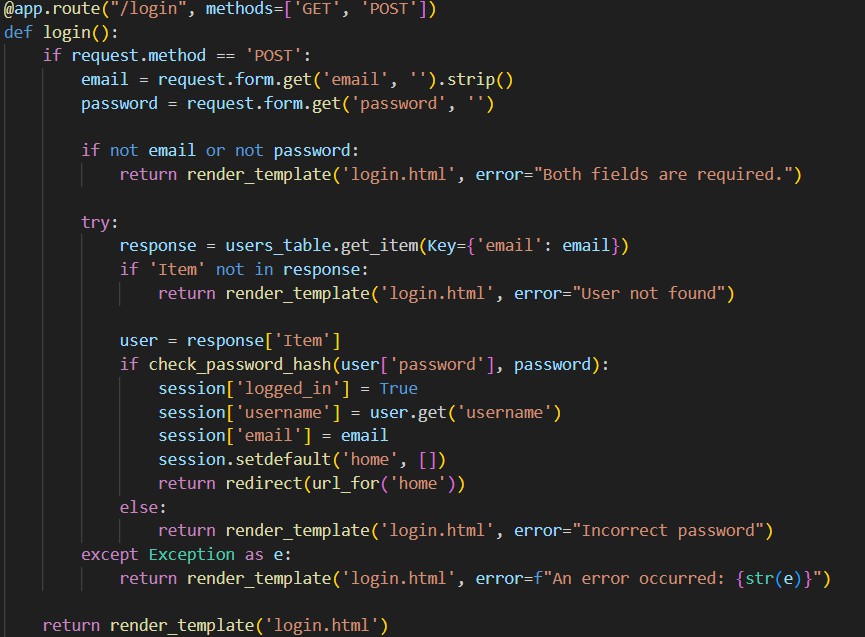
****

# Routes for Web Pages

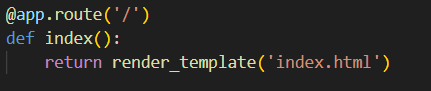
* + - **Home Route:**

****

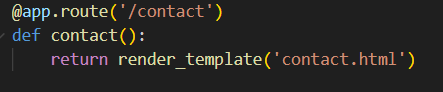
* + - **Login Route:**

****

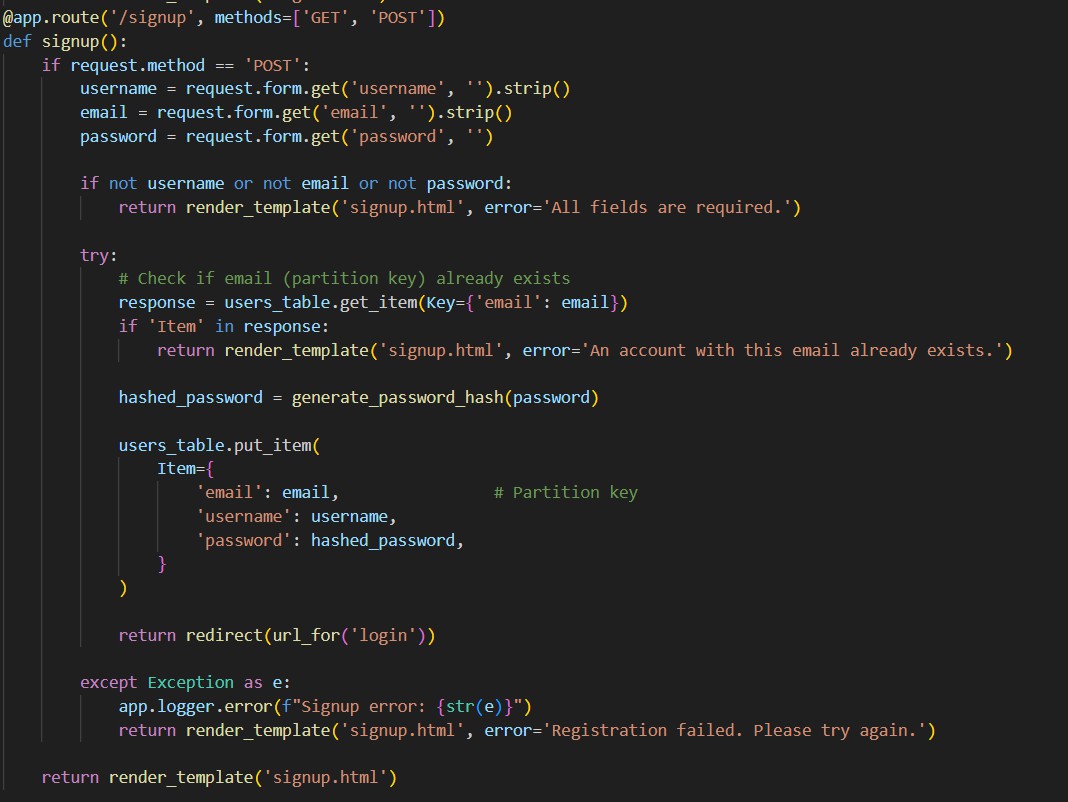
* + - **Index Route:**

****

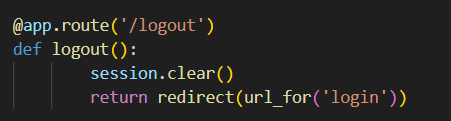
* + - **Contact Route:**

****

* + - **Sign Up Route:**

****

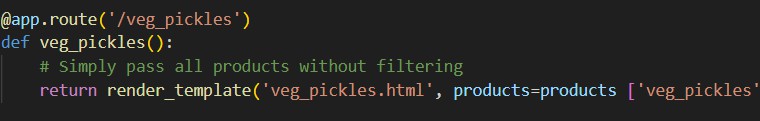
* + - * **Log Out Route:**

****

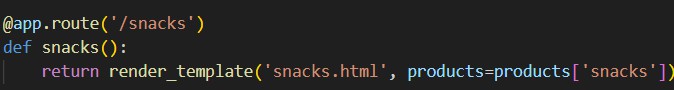
* + - * + **Non-Veg Pickles Route:**

****

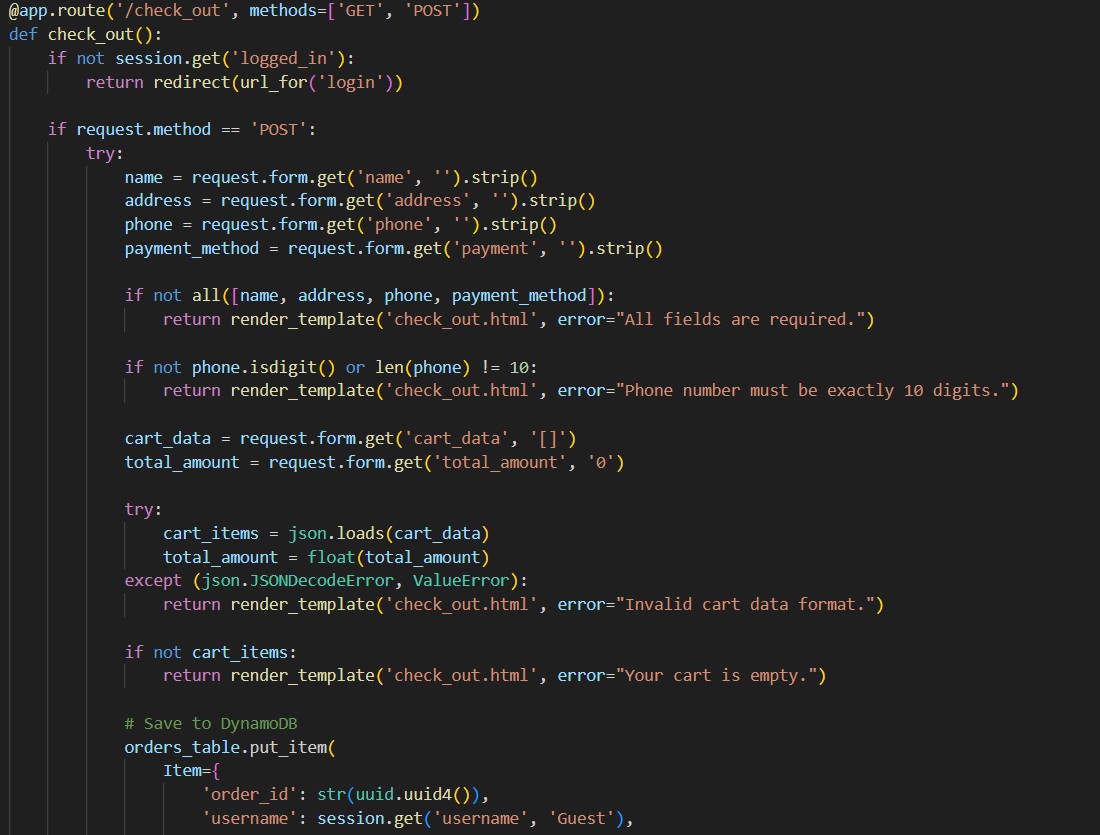
* + - * **Veg Pickles Route:**

****

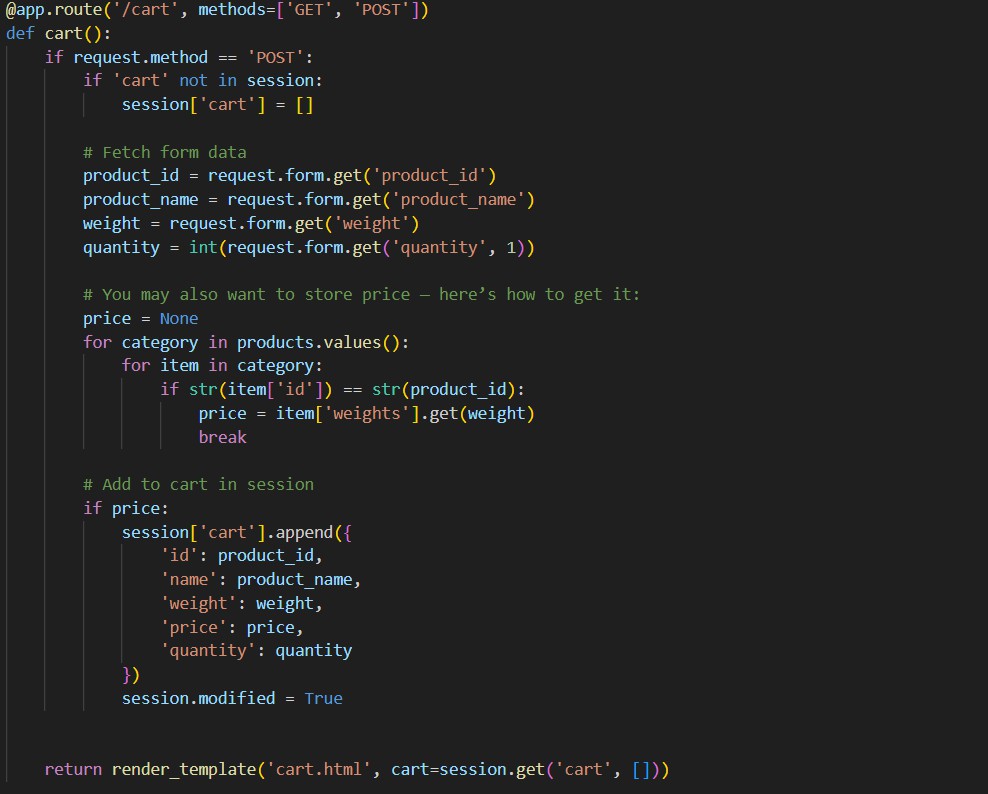
* + - **Snacks Route:**

****

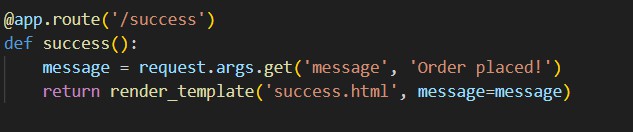
* + - * **Checkout Route:**

****

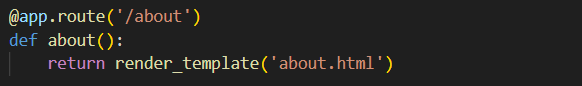
* + - **Cart Route:**

****

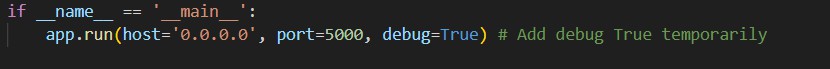
* + - **Success Route:**

****

* + **About Route:**

****

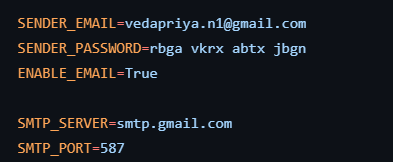
* + **Deployment of code:**

****

**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:**

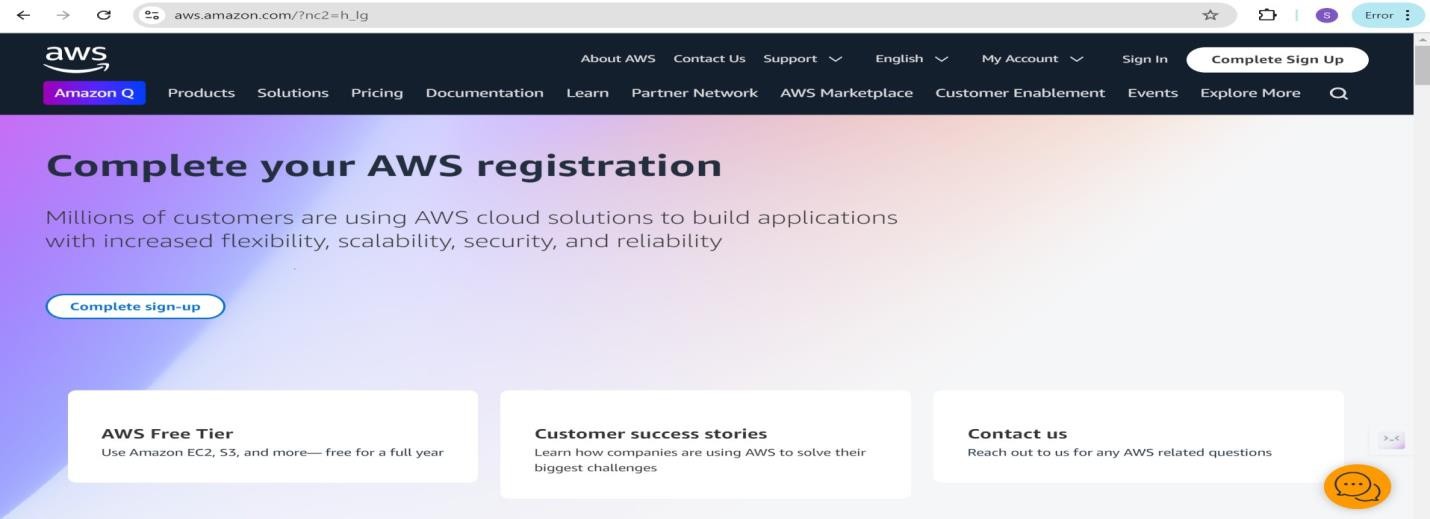
****

**2: AWS Account Setup**

**Milestone 2: AWS Account Setup**

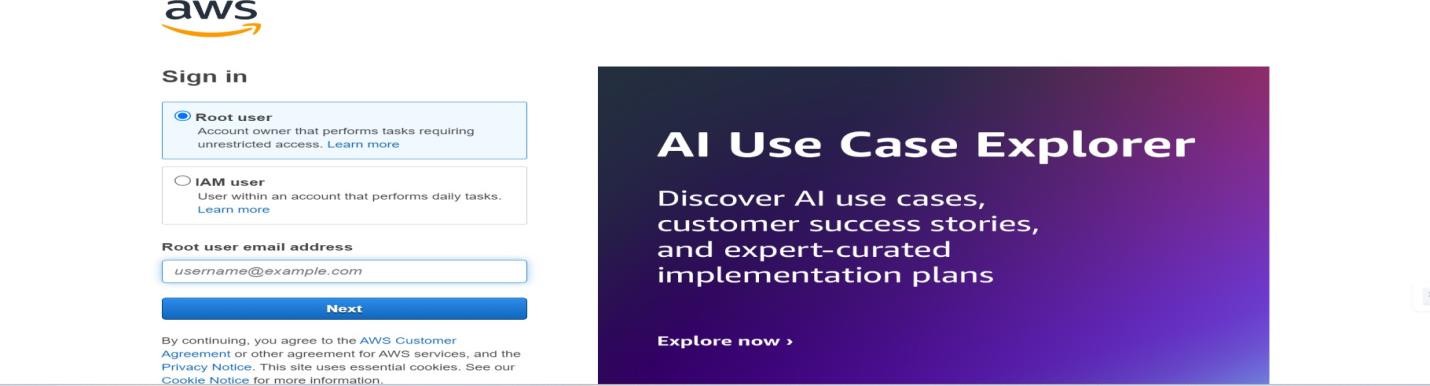
## Activity 2.1: Set up an AWS account if not already done.

* + 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.



## Activity 2.2: Log in to the AWS Management Console

* After setting up your account, log in to the [AWS Management Console.](https://aws.amazon.com/console/)

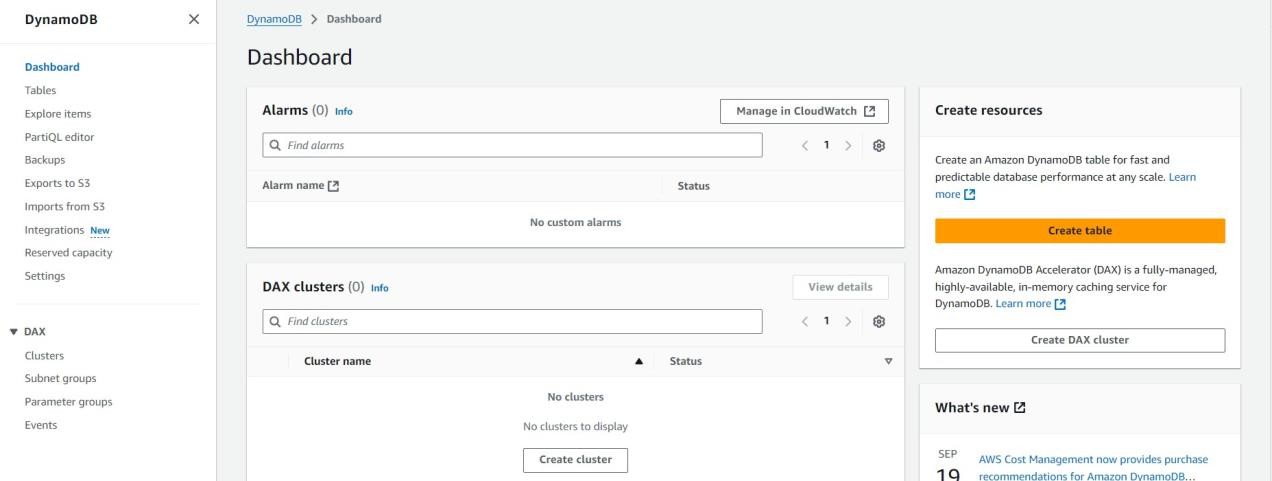
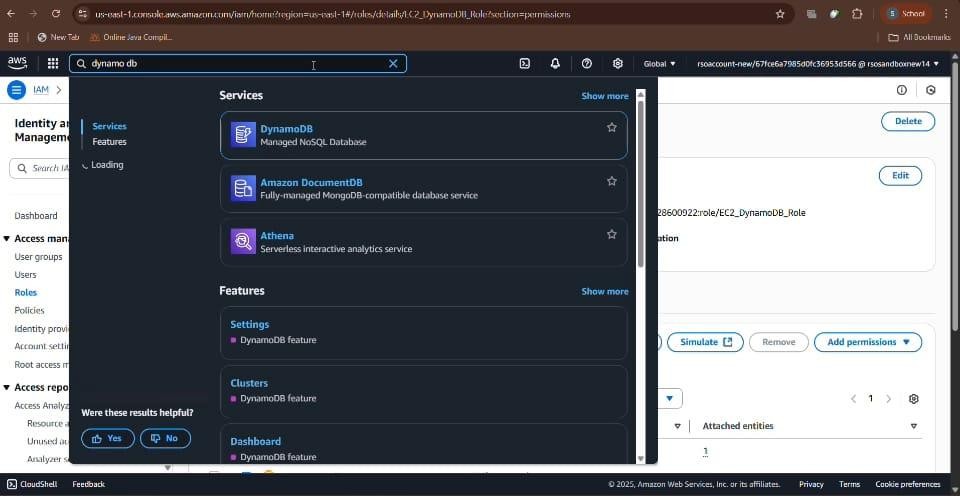


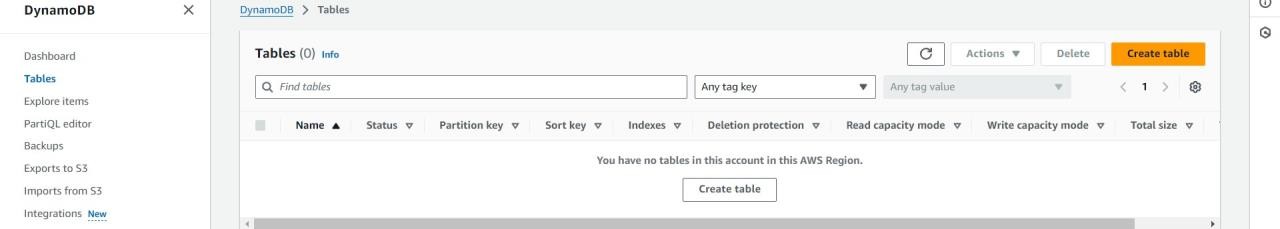
**3: DynamoDB Database Creation and Setup**

**Milestone 3: DynamoDB Database Creation and Setup**

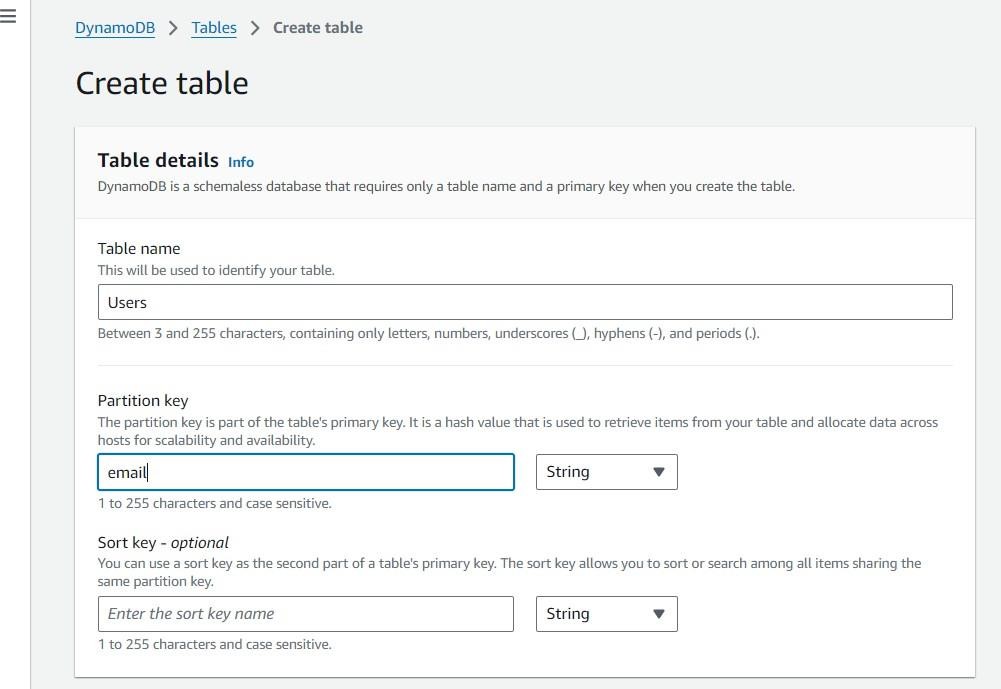
## Activity 3.1: Navigate to the DynamoDB

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

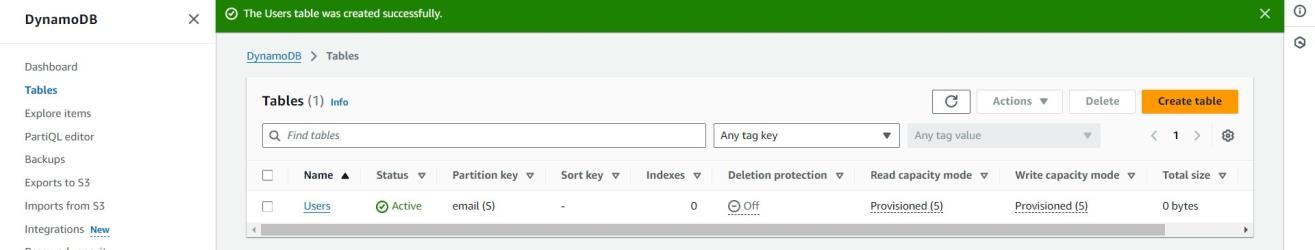




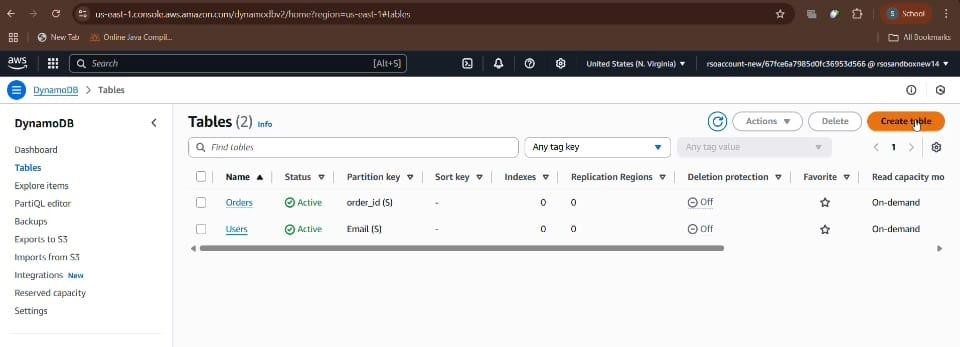
## Activity 3.2: Create a DynamoDB table for storing registration details and book requests.

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

○



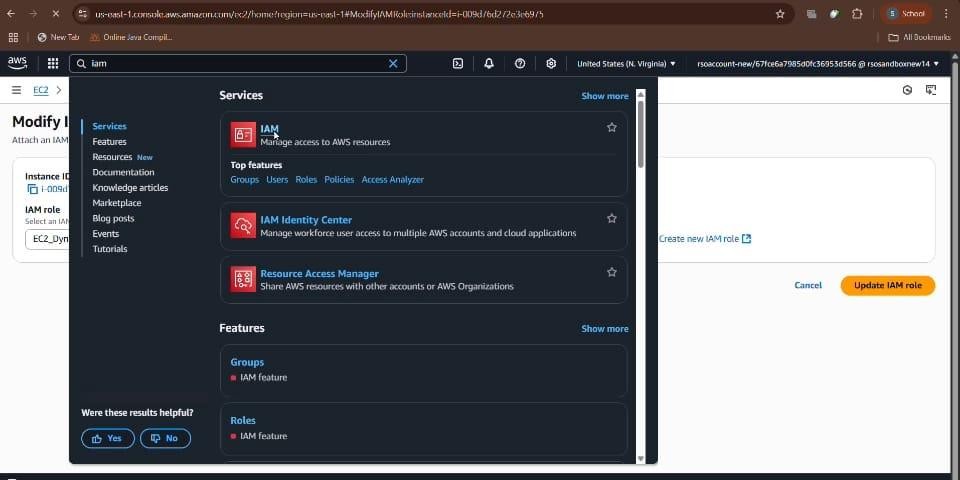
* + Follow the same steps to create a requests table with E-mail as the primary key for book requests data.

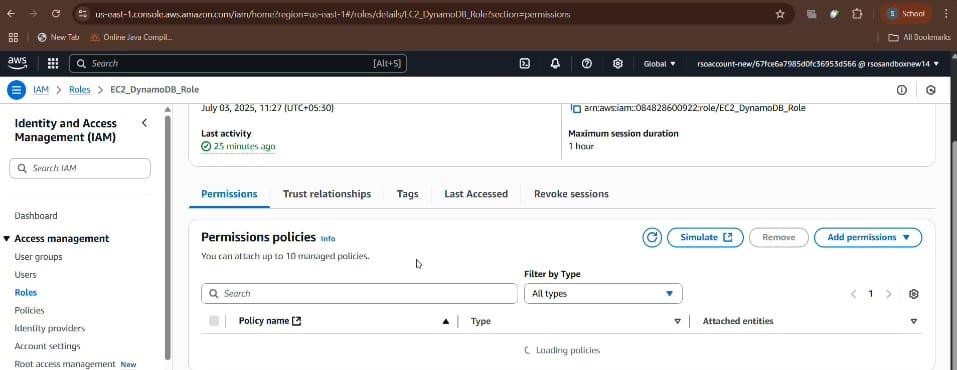


**4: IAM Role Setup Milestone 4: IAM Role Setup**

## Activity 4.1: Create IAM Role

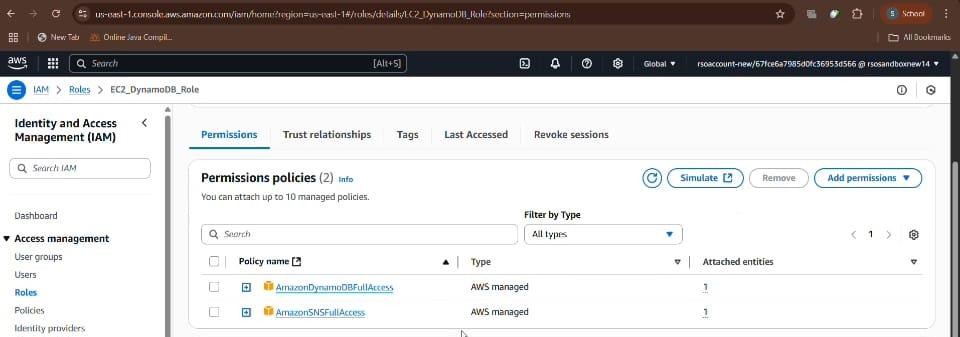
* + In the AWS Console, navigate to IAM and create a new IAM Role for EC2 to allow interaction with DynamoDB.





## 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.



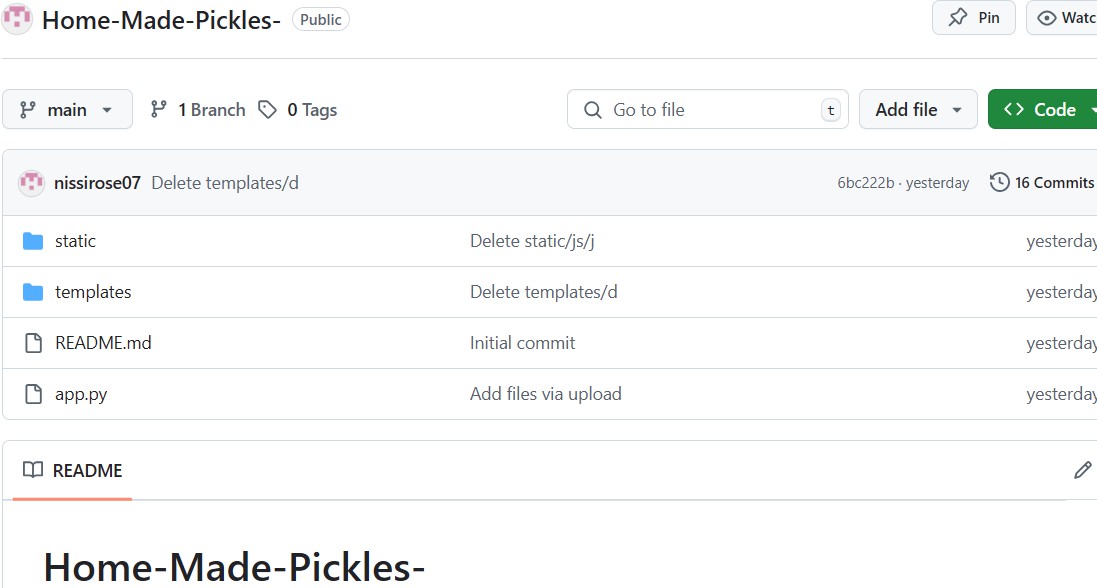
**5: EC2 Instance Setup**

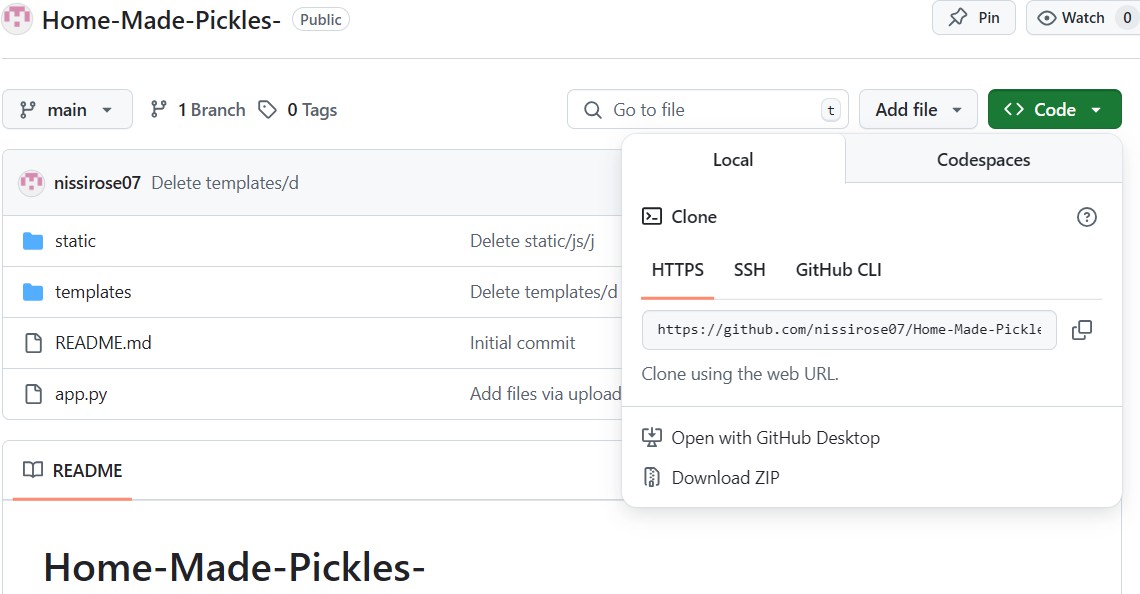
**Milestone 5: EC2 Instance Setup**

## Activity 5.1: Load Project Files to GitHub

* + Upload your Flask application and HTML files to a GitHub repository.

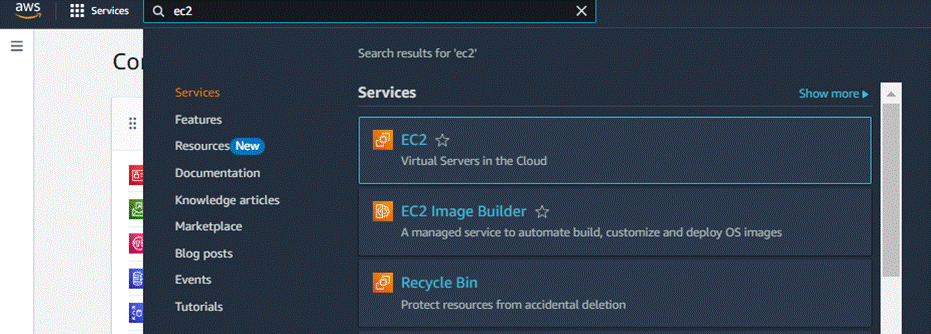
*Note: This will allow easy access and deployment to the EC2 instance.*

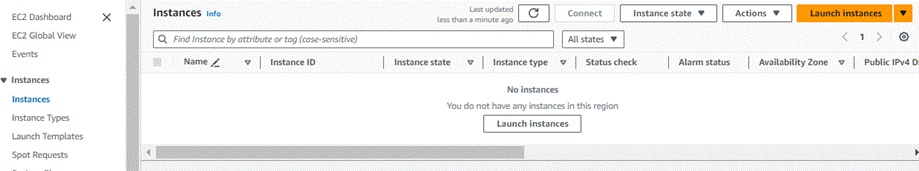


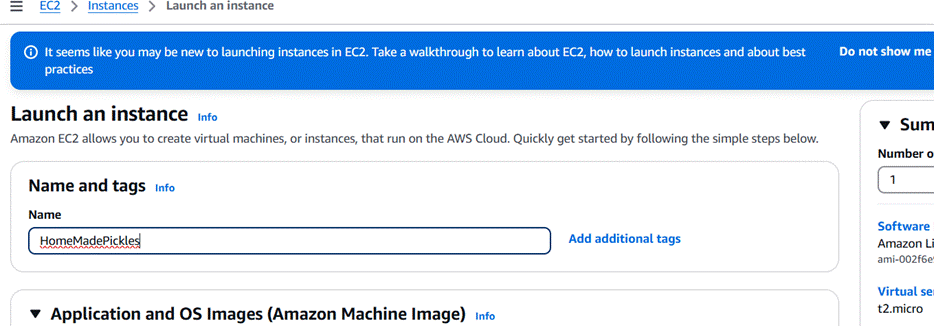
**

### Activity 5.2: Launch an EC2 Instance

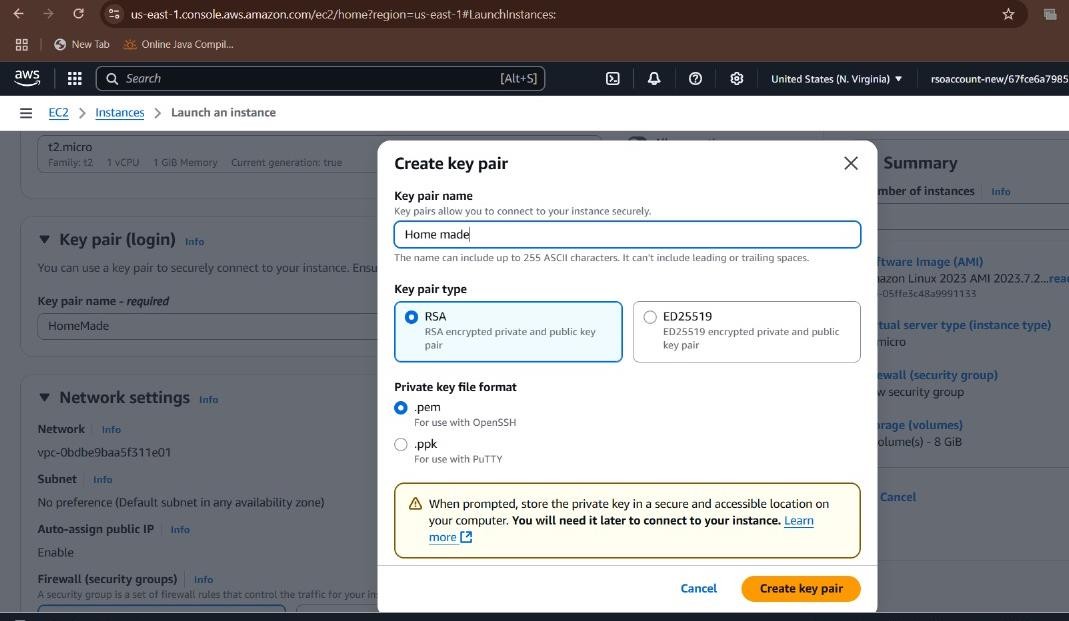
* In the AWS Console, go to EC2 and click **"Launch Instance"**.
* Choose **Amazon Linux 2** or **Ubuntu** as the AMI and select **t2.micro** (Free-tier eligible).





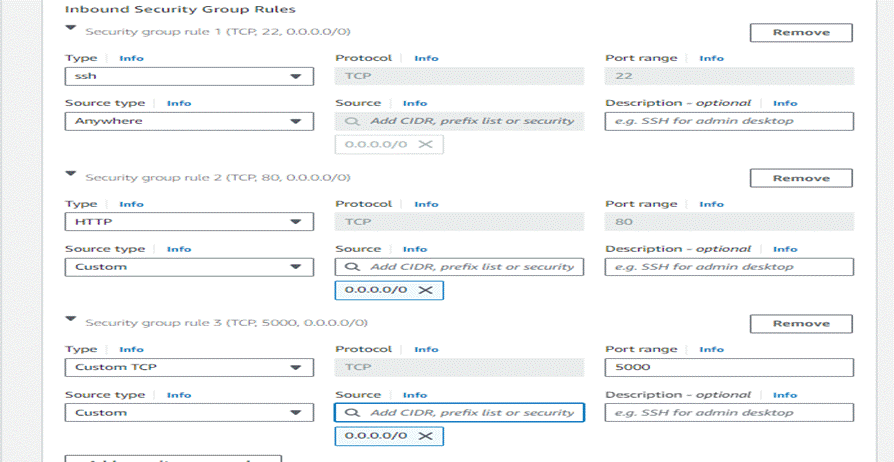


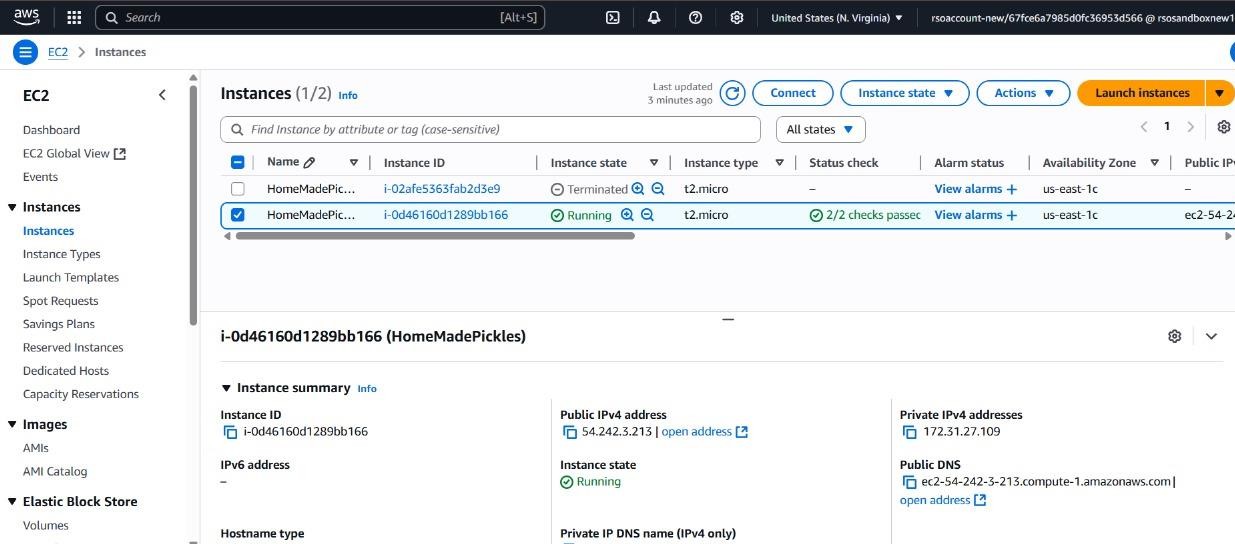
* Create and download a **key pair** for secure SSH access.



## Activity 5.3: Configure Security Groups

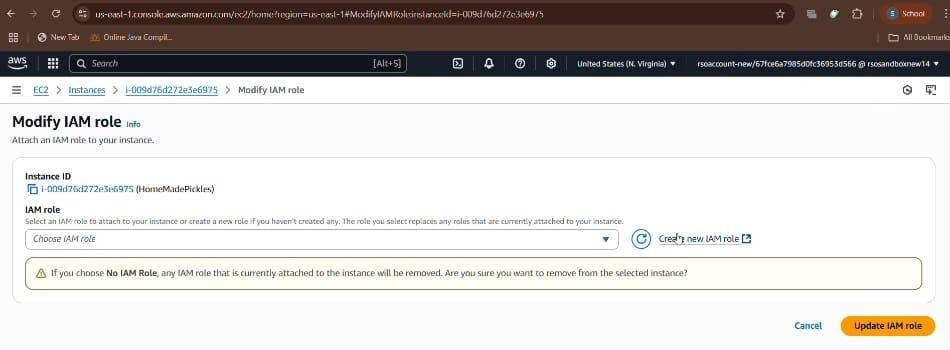
* + - Allow **HTTP (port 80)** and **SSH (port 22)** inbound traffic.





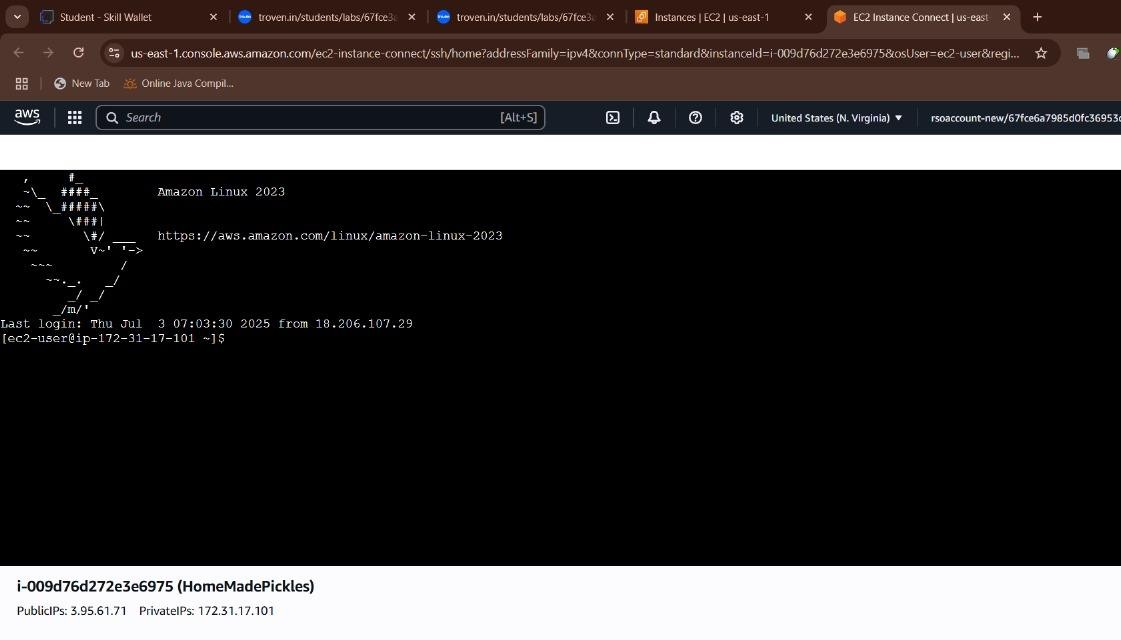
## Activity 5.4: Attach IAM Role

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



## Activity 5.5: Connect to EC2 Instance

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



**6: Deployment on EC2**

**Milestone 6: Deployment on EC2**

## Activity 6.1: Install Required Software

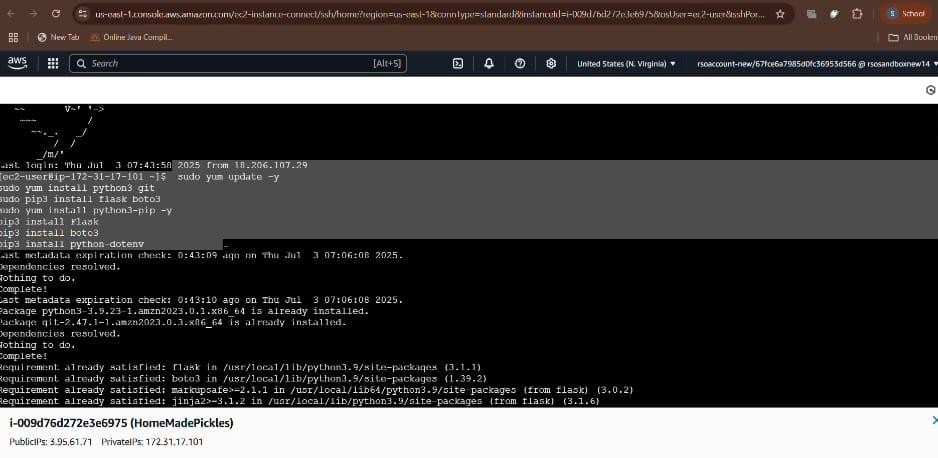
* + Run the following commands to install necessary packages: sudo yum update -y

sudo yum install python3 git sudo pip3 install flask boto3

* + Verify installations:

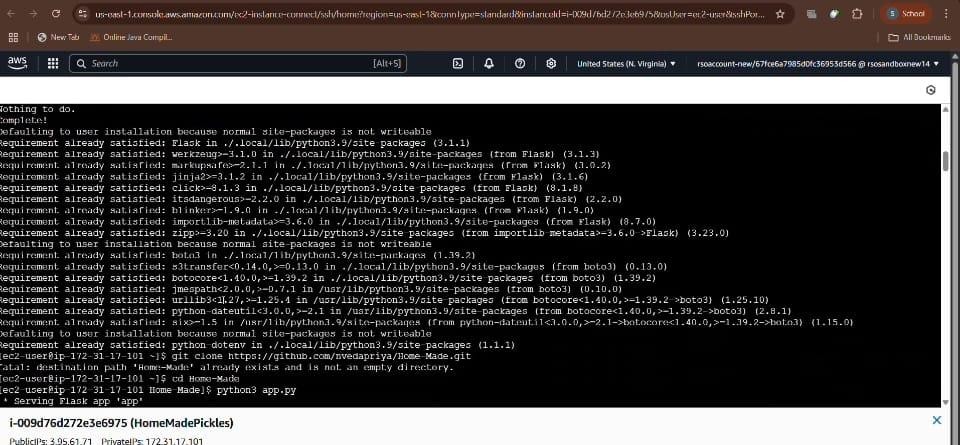
bash

Copy code flask --version git --version



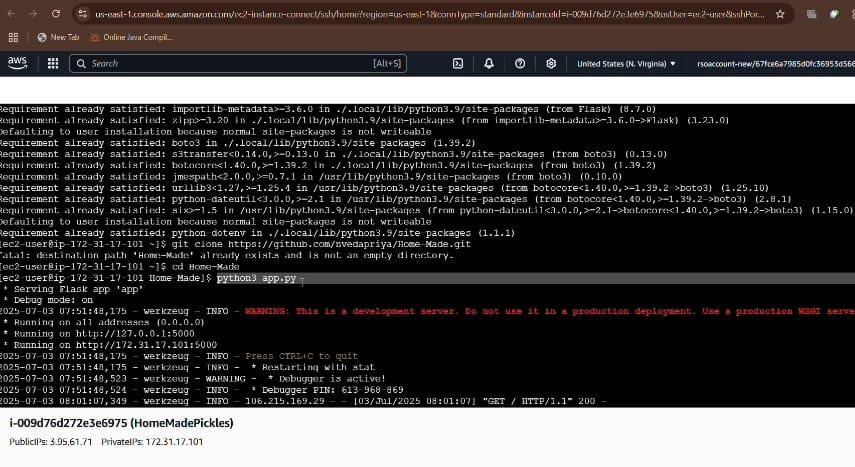
## Activity 6.2: Clone Flask Project from GitHub

* + Run: git clone https://github.com/nissirose07/Home-Made-Pickles-.git
  + Navigate to the project folder: cd [**Home-Made**](https://github.com/nvedapriya/Home-Made)

****

## Activity 6.3: Run the Flask Application

* + - * Run: python3 app.py



## Activity 6.6: Access the Website

* + Open your browser and go to: [http://127.0.0.1:5000](http://127.0.0.1:5000/)

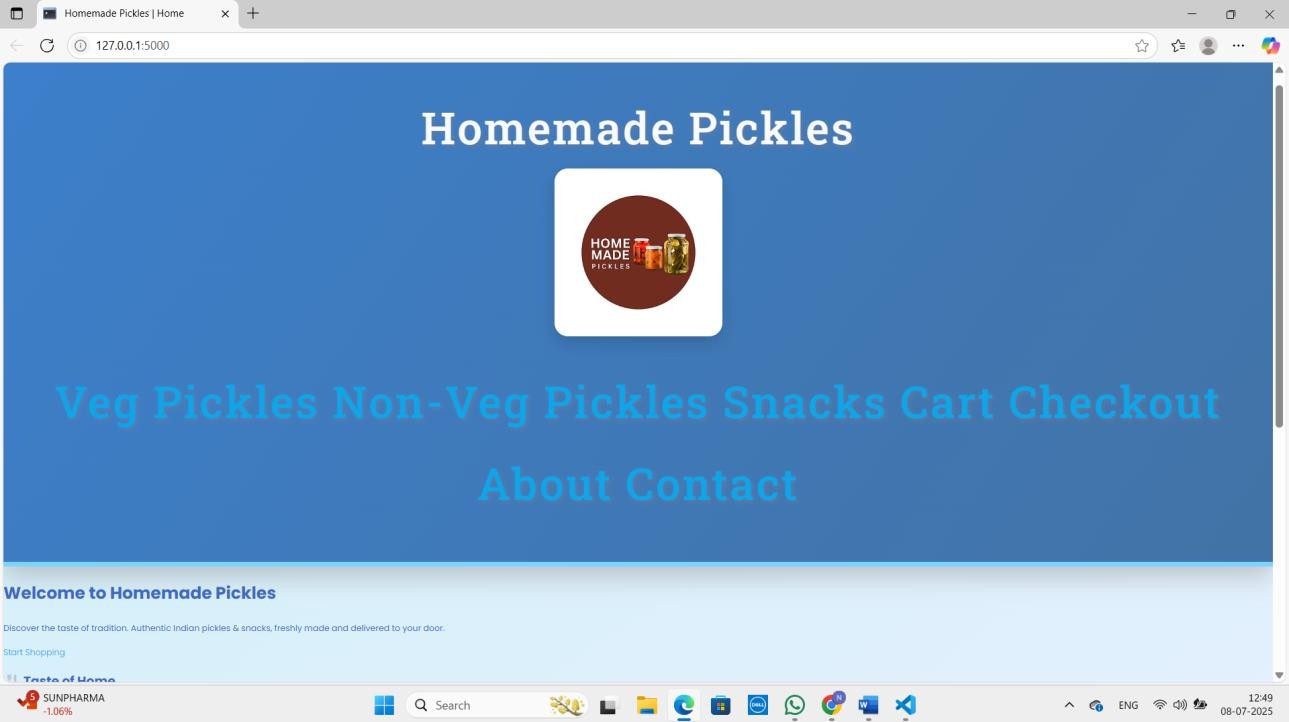
**7: Testing and Deployment**

**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:

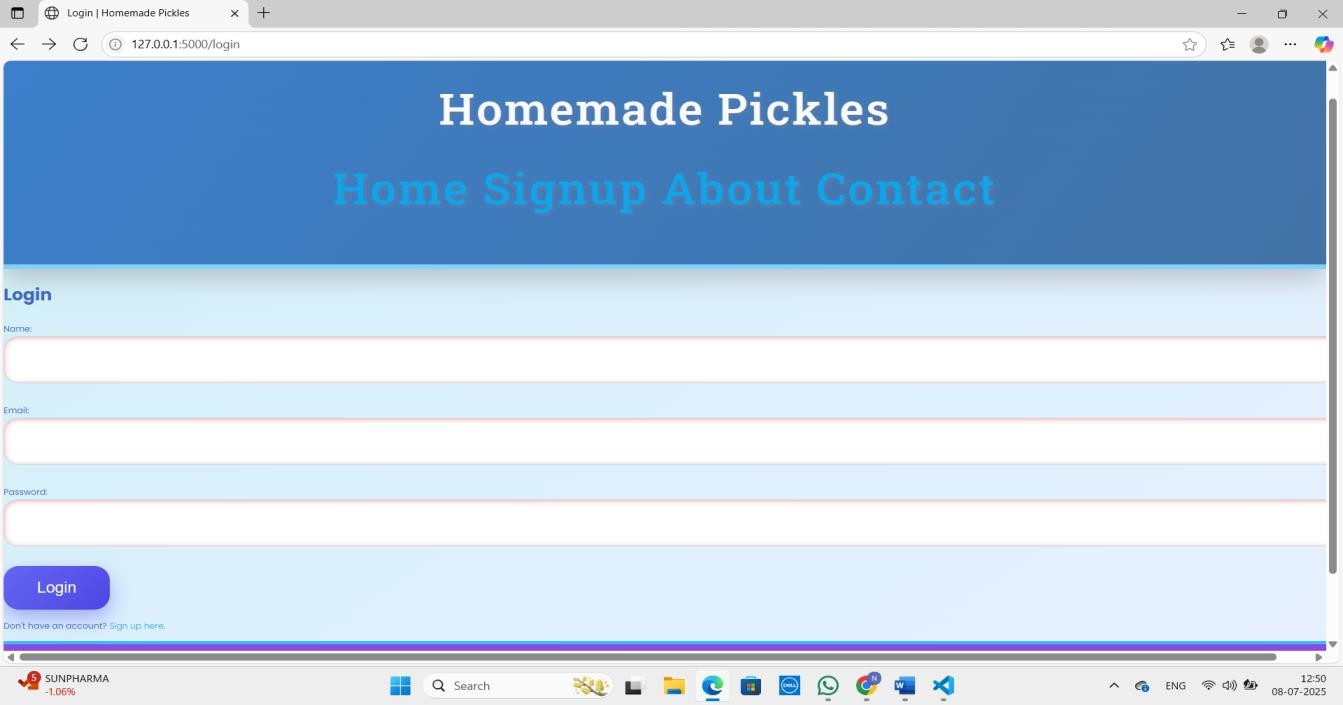
**Home Page:**

****

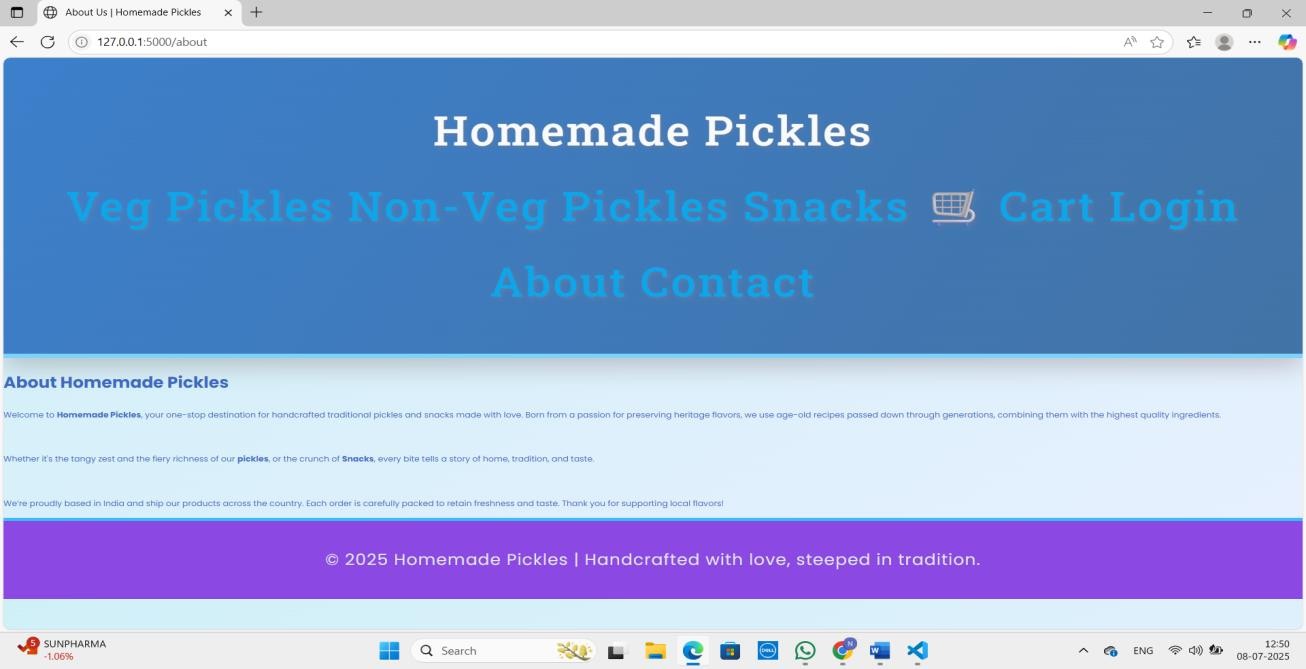
**Signup Page:**

****

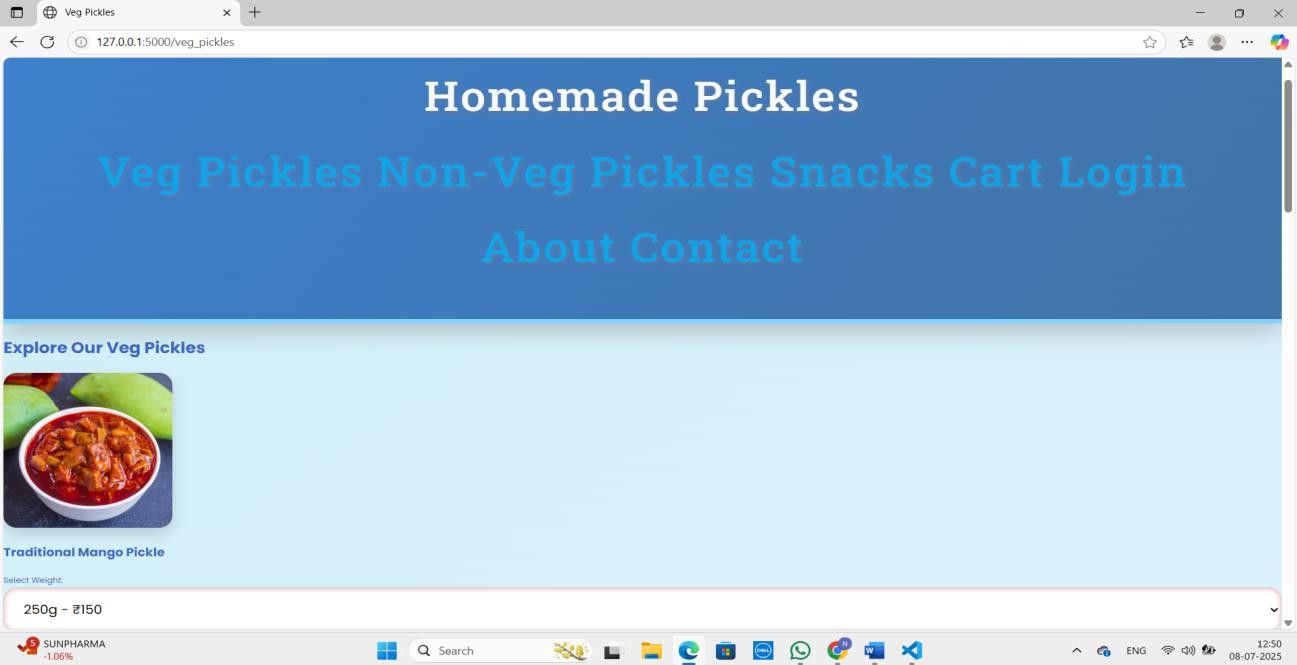
**Login Page:**

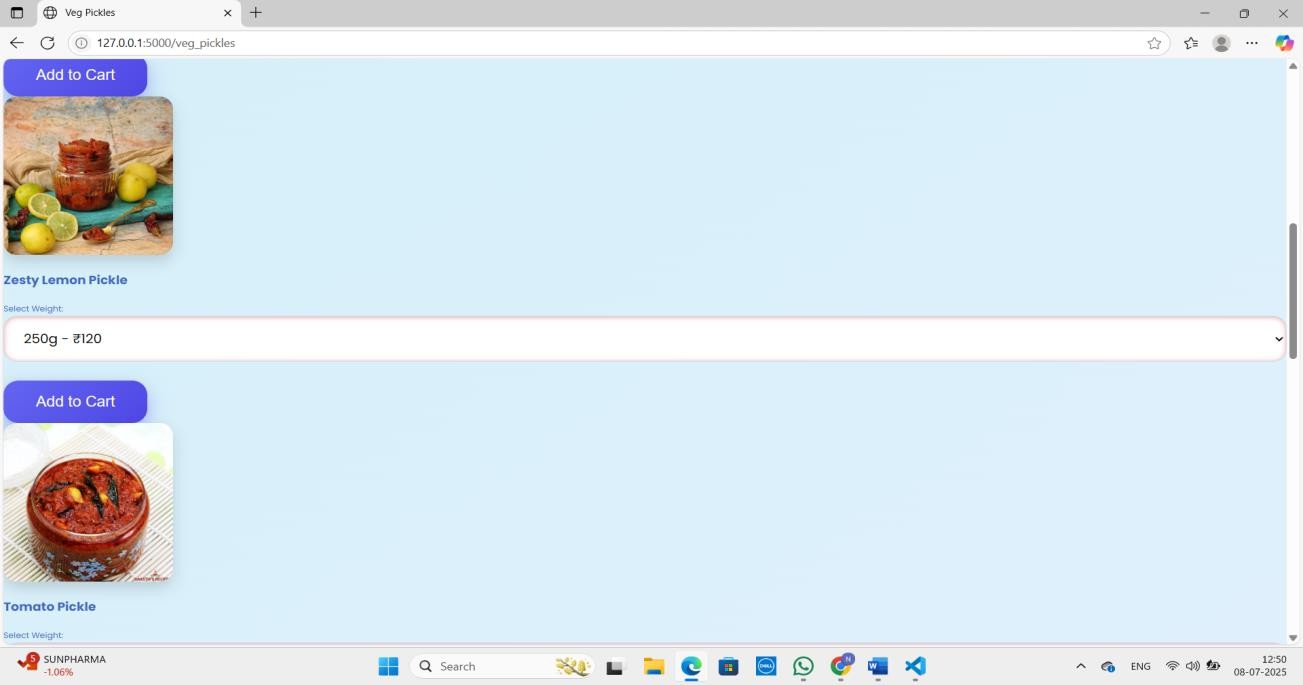
****

**About Page:**

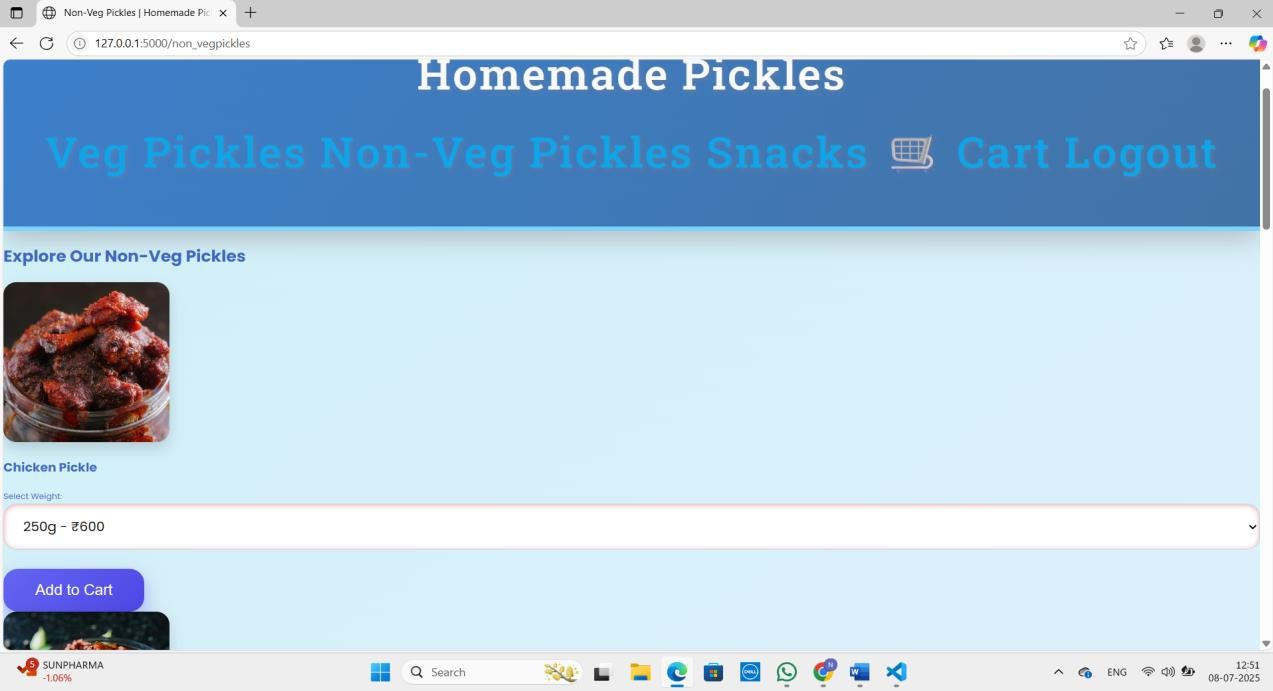
****

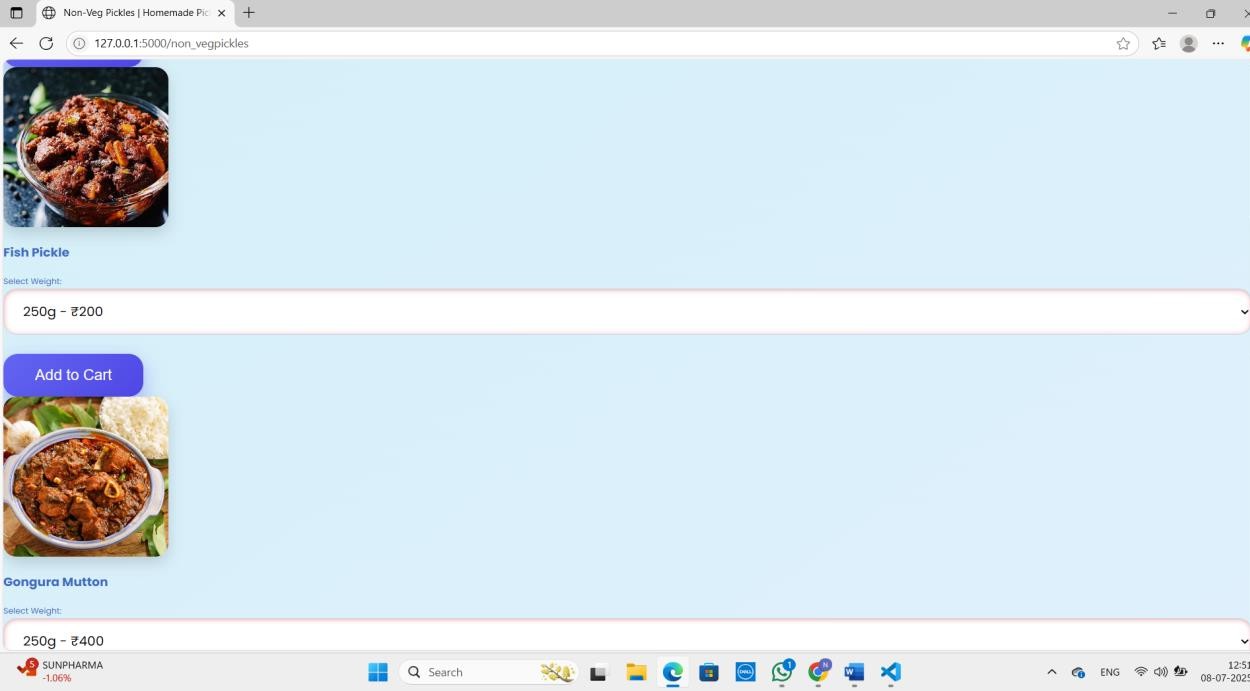
**Veg Pickles Page:**

****

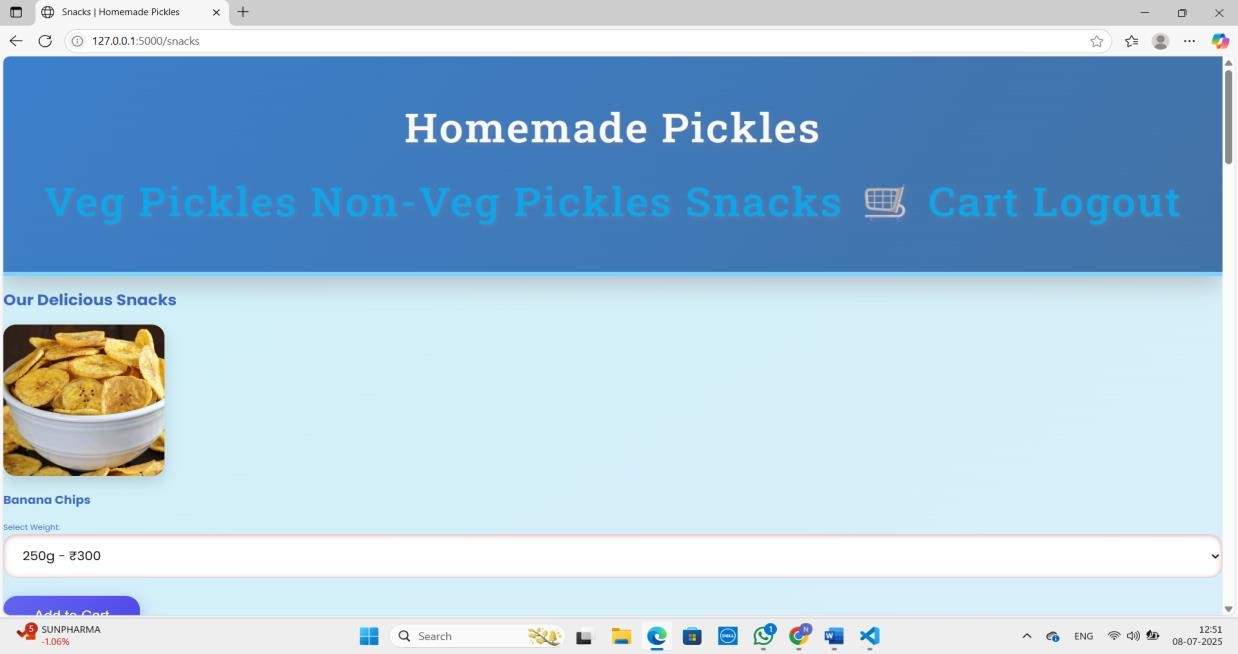
****

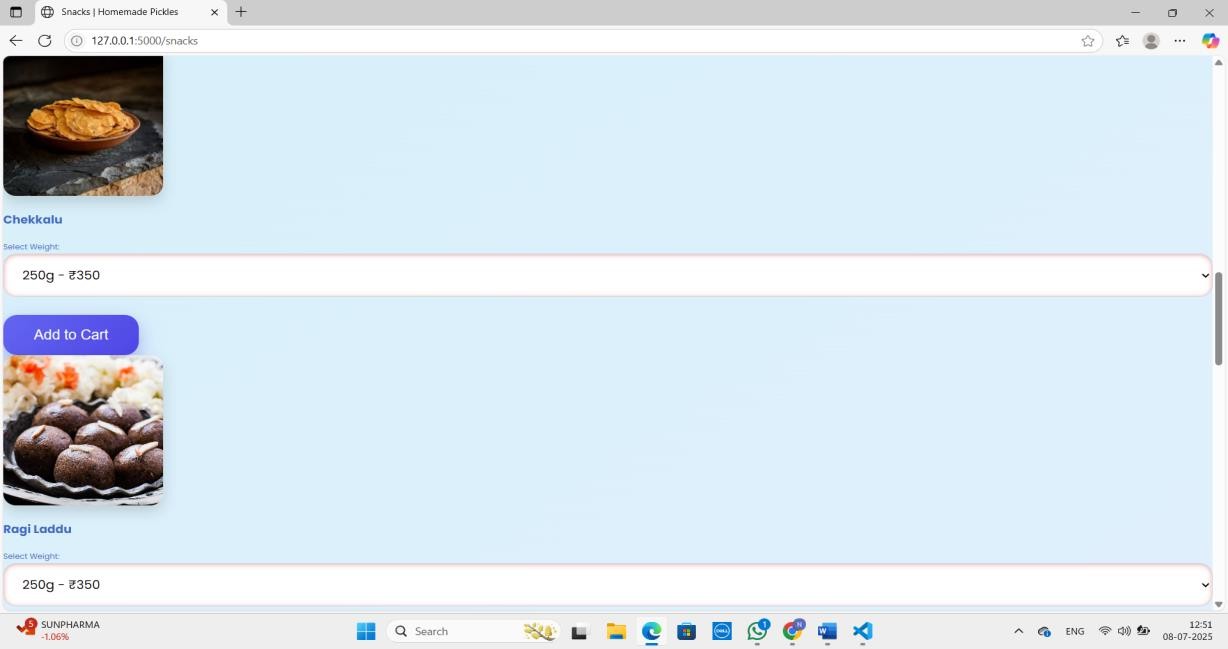
**Non-Veg Pickles Page:**

****

****

**Snacks page:**

****

****

**Cart Page:**

****

**Success Page:**

****