

## 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.

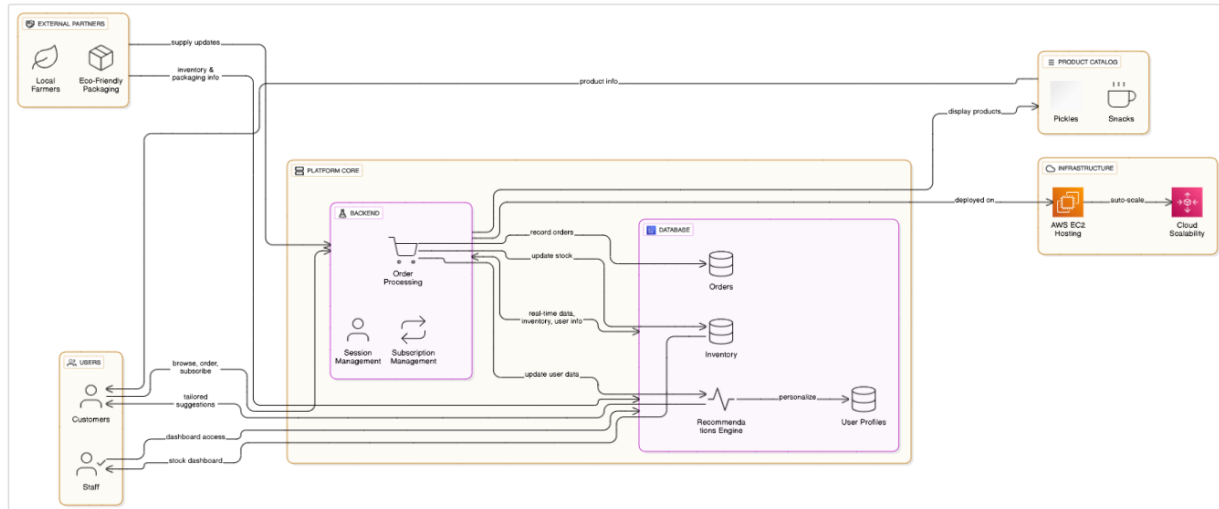
### Scenario 2: Real-Time Inventory Tracking and Updates

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.

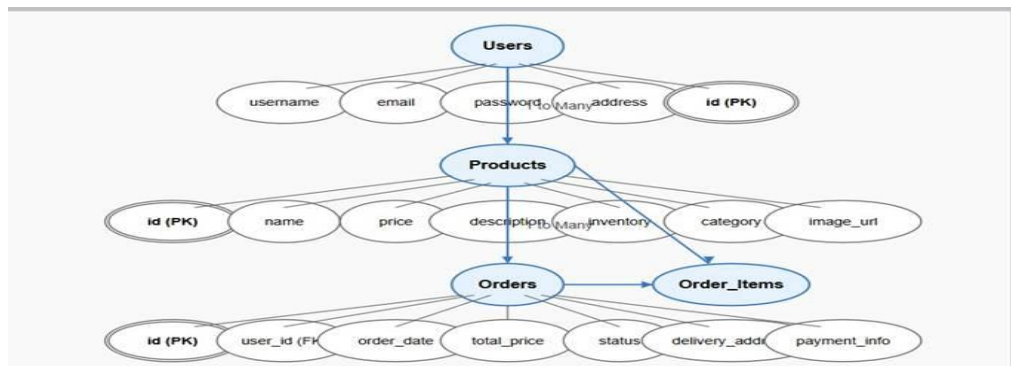
### Scenario 3: Personalized User Experience and Recommendations

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

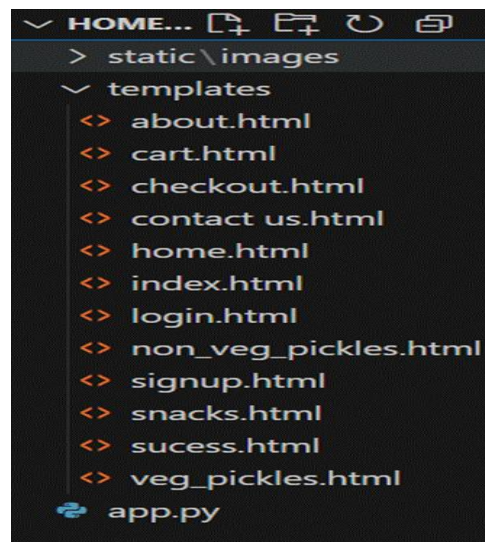
### **Milestone 8. Testing and Deployment**

- 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.



**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, '1000': 1050}},
  ],
  'veg_pickles': [
    {'id': 7, 'image': 'traditional_mango_pickle.jpg', 'name': 'Traditional Mango Pickle', 'weights': {'250': 150, '500': 280, '1000': 400}},
    {'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': 'karakakaya_pickle.jpg', 'name': 'Karakakaya 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:**

```
@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:**

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

- **Non-Veg Pickles Route:**

```
@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:**

```
@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:**

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

- **About Route:**

```
@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.

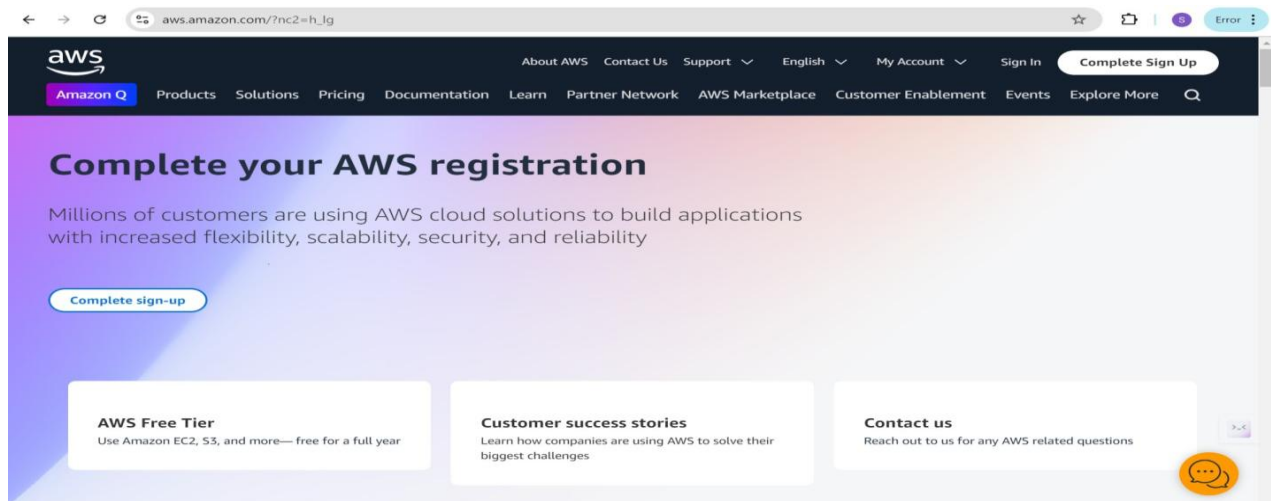
- **.env file:**

```
SENDER_EMAIL= kancharlamythili612@gmail.com  
SENDER_PASSWORD=efhw bvoc mrvg udgz  
ENABLE_EMAIL=True  
  
SMTP_SERVER=smtp.gmail.com  
SMTP_PORT=587  
|
```

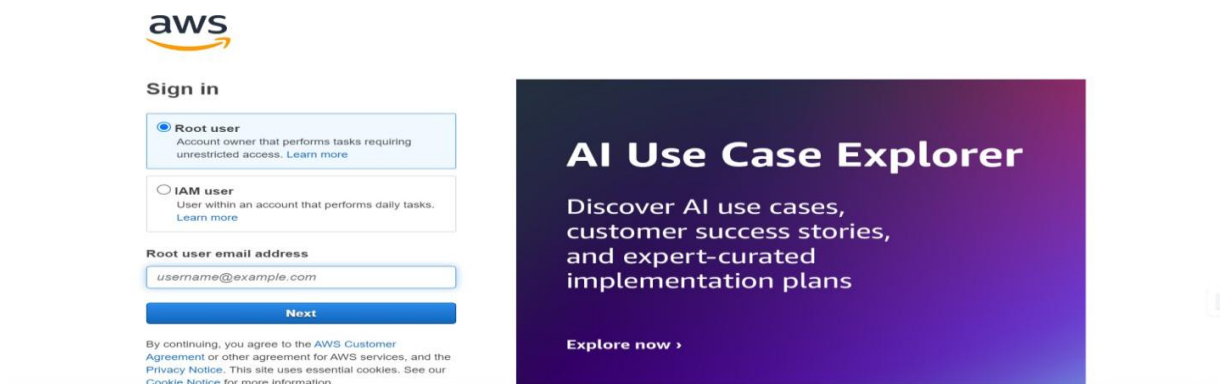
## 2: AWS Account Setup

### 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.



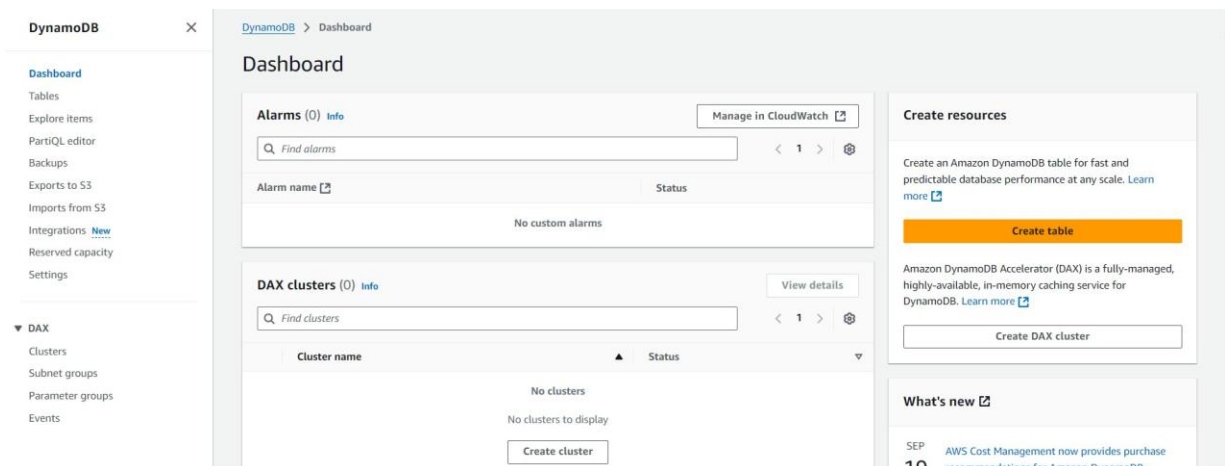
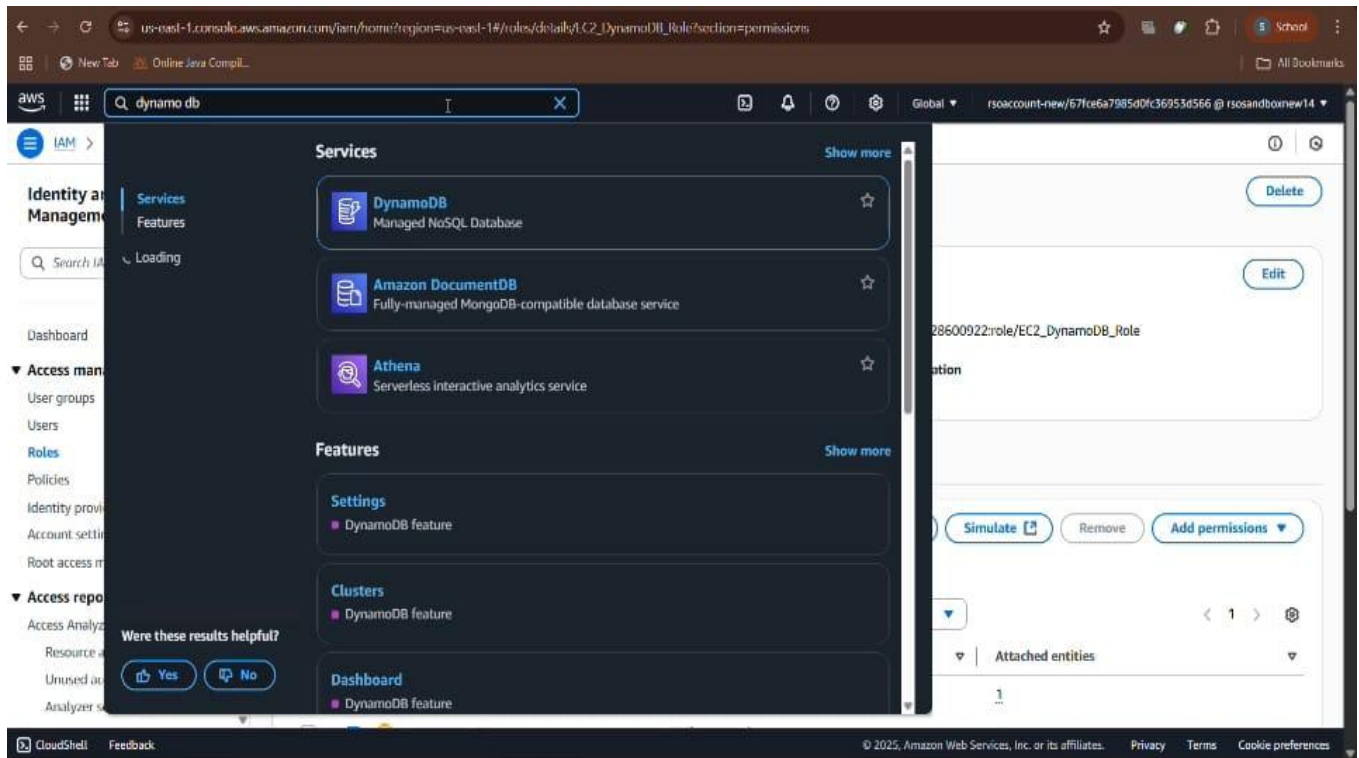
- **Activity 2.2: Log in to the AWS Management Console**
  - After setting up your account, log in to the [AWS Management Console](#).

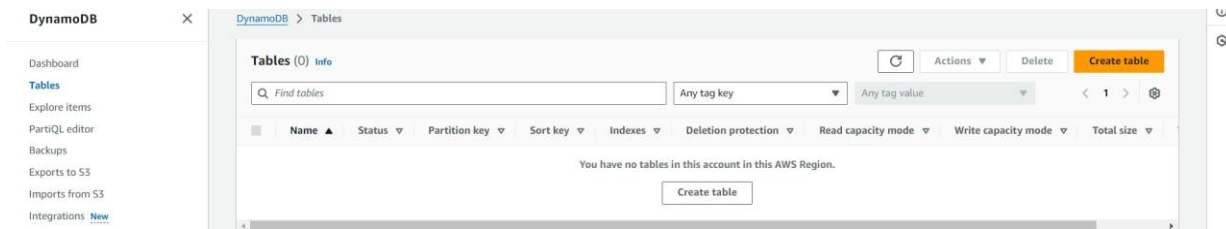


### 3: DynamoDB Database Creation and Setup

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

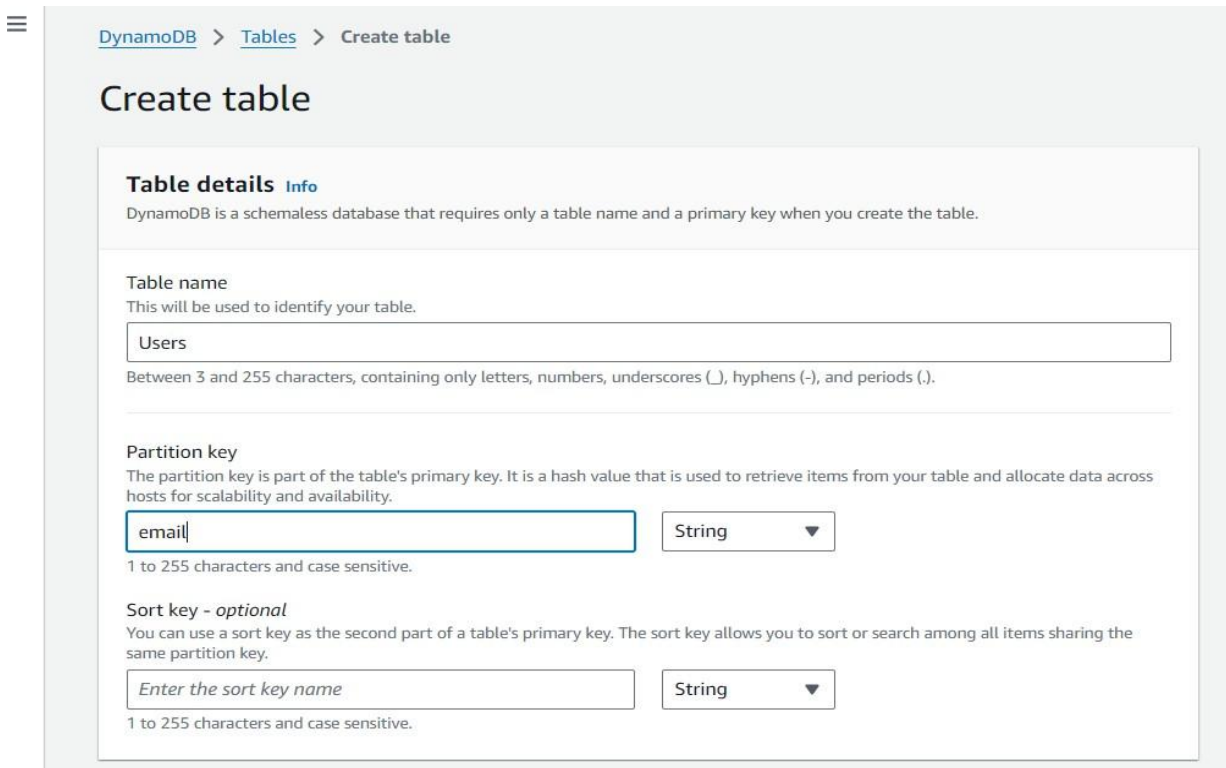
- **Activity 3.1: Navigate to the DynamoDB**
  - In the AWS Console, navigate to DynamoDB and click on create tables.





### • 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.



**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.

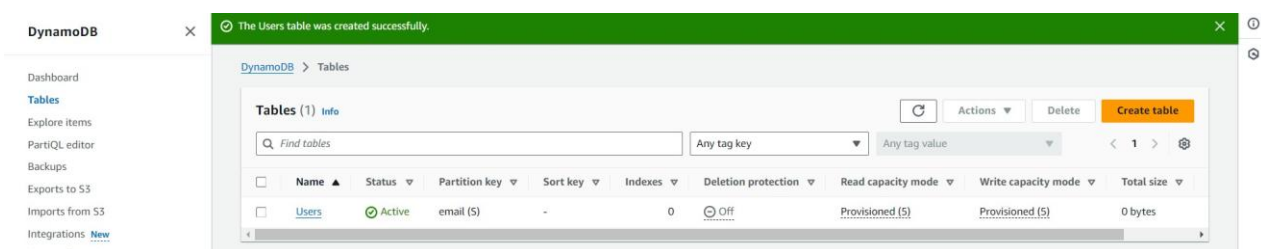
email String

1 to 255 characters and case sensitive.

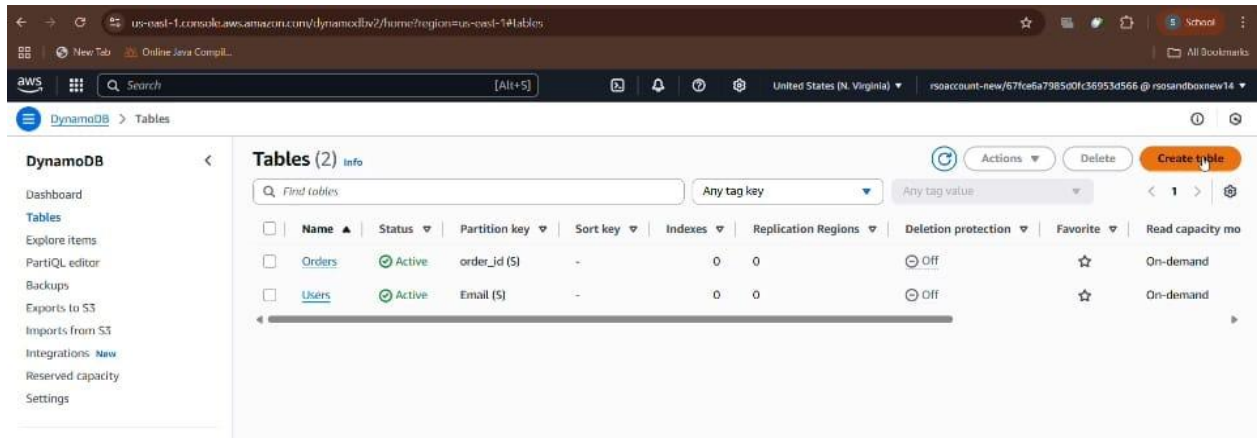
**Sort key - optional**  
You can use a sort key as the second part of a table's primary key. The sort key allows you to sort or search among all items sharing the same partition key.

Enter the sort key name String

1 to 255 characters and case sensitive.



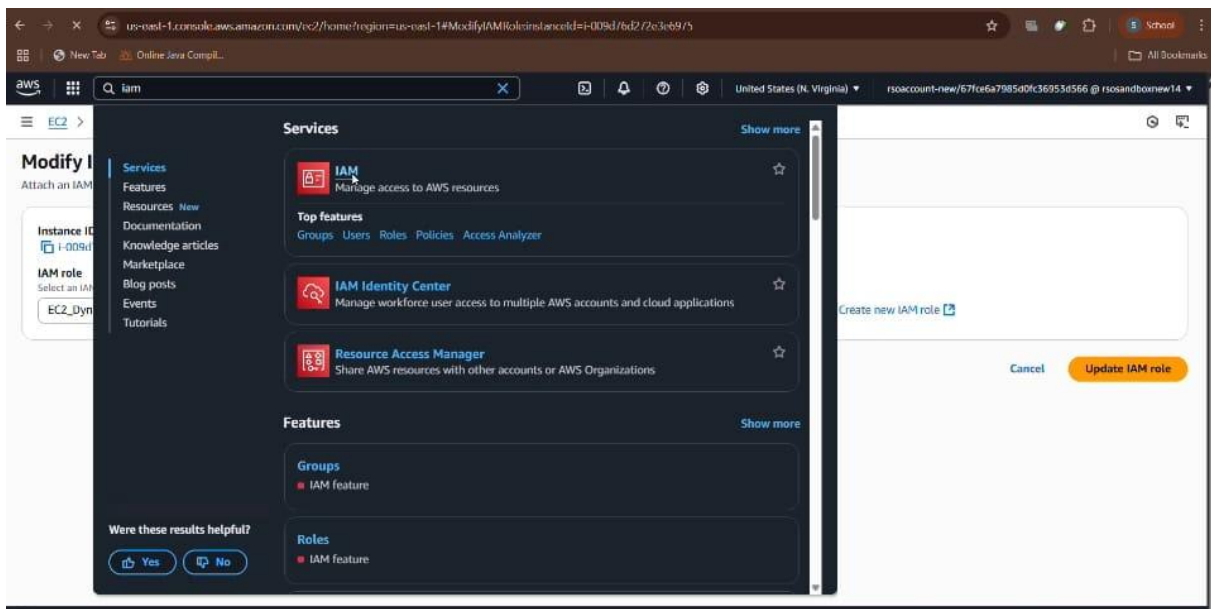
- Follow the same steps to create a requests table with E-mail as the primary key for book requests data.



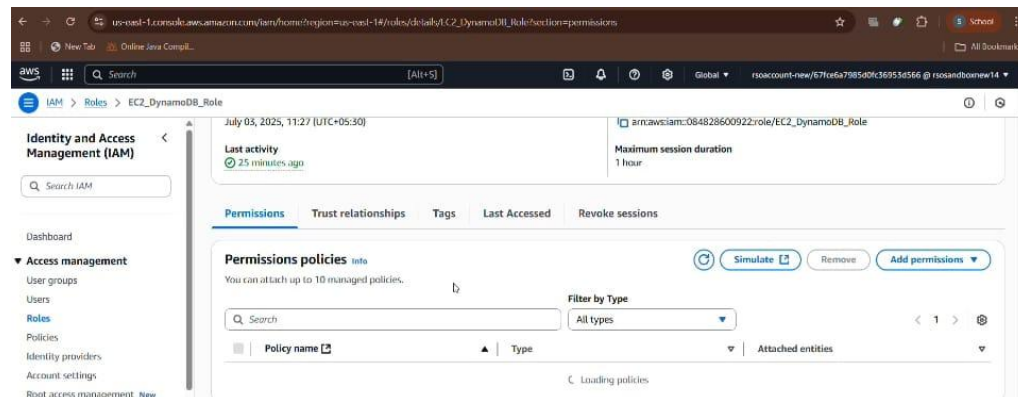
## 4: IAM Role Setup

### 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.

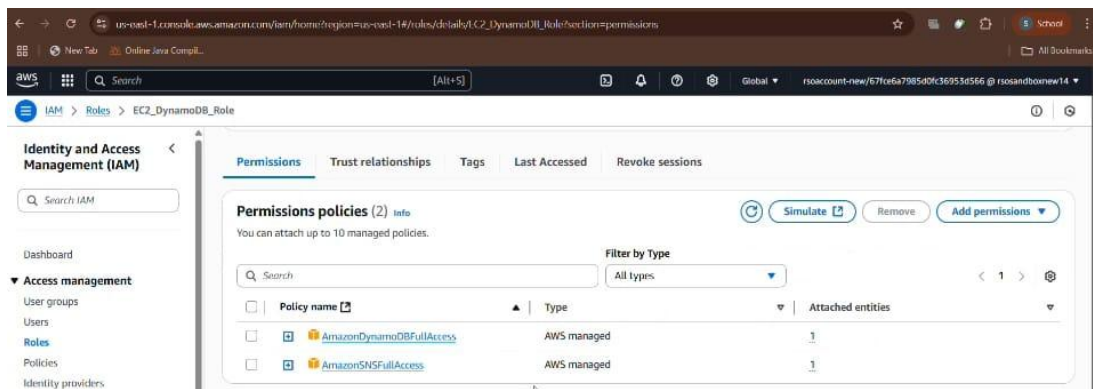






- **Activity 4.2: Attach Policies**

- 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.*

MythiliKancharia / pickleshomemade

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

pickleshomemade Public

main 1 Branch 0 Tags

Go to file

Add file Code

About

No description, website, or topics provided.

Readme

Activity

0 stars

0 watching

0 forks

Releases

No releases published

Create a new release

Packages

No packages published

Publish your first package

Languages

HTML 56.9% Python 29.9% CSS 13.2%

Suggested workflows

Based on your tech stack

MythiliKancharia Add files via upload 16/5/22 4 hours ago 6 Commits

File	Commit	Time
static	Add files via upload	5 days ago
templates	Add files via upload	5 days ago
.env	Add files via upload	4 hours ago
README.md	Initial commit	5 days ago
app.py	Update app.py	2 days ago
style.css	Add files via upload	5 days ago

pickleshomemade

Pull requests Actions Projects Wiki Security Insights Settings

pickleshomemade Public

main 1 Branch 0 Tags

Go to file

Add file Code

About

No description, website, or topics provided.

Readme

Activity

0 stars

0 watching

0 forks

Releases

No releases published

Create a new release

Packages

No packages published

Publish your first package

Languages

HTML 56.9% Python 29.9% CSS 13.2%

MythiliKancharia Add files via upload

File	Commit	Time
static	Add files via upload	
templates	Add files via upload	
.env	Add files via upload	
README.md	Initial commit	
app.py	Update app.py	
style.css	Add files via upload	

pickleshomemade

Local Codespaces

Clone

HTTPS SSH GitHub CLI

https://github.com/MythiliKancharia/pickleshomemade

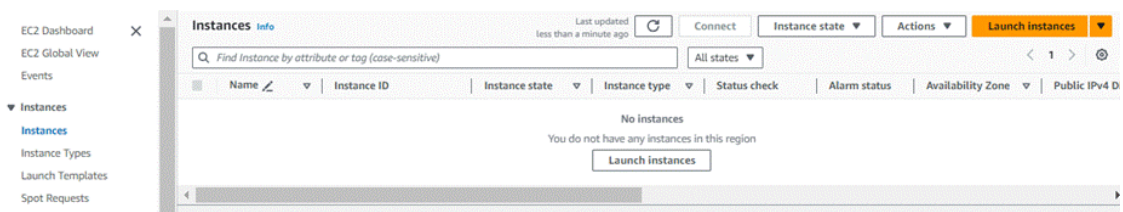
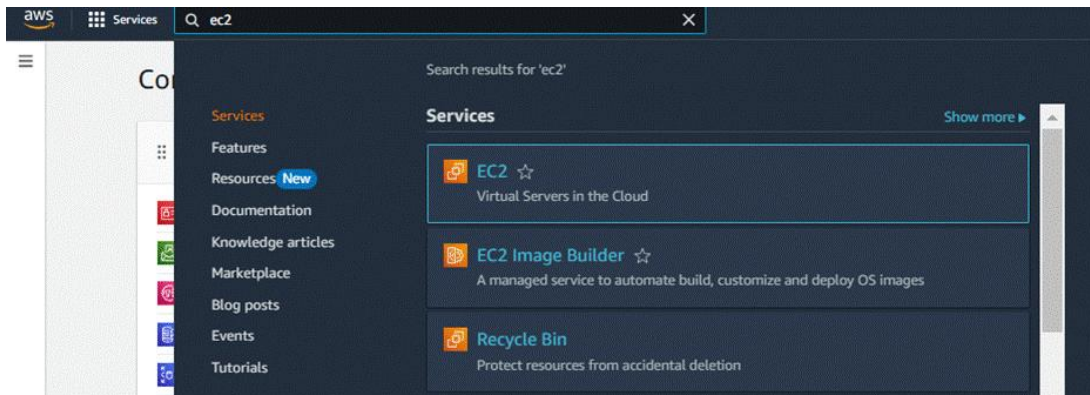
Clone using the web URL

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

[Add additional tags](#)

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

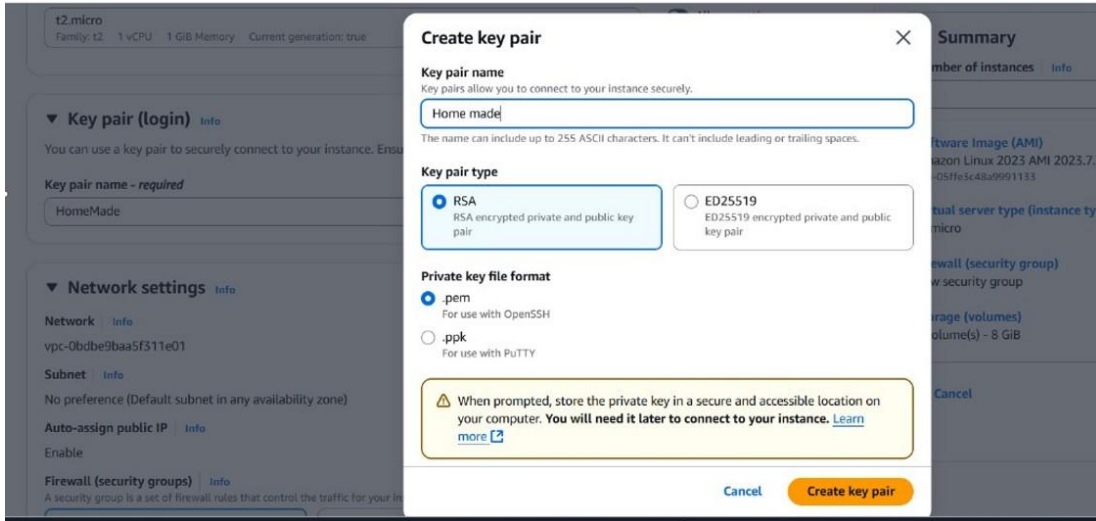
### ▼ Summary

Number of instances

Software: Amazon Linux 2  
ami-002f6e...

Virtual size: t2.micro

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



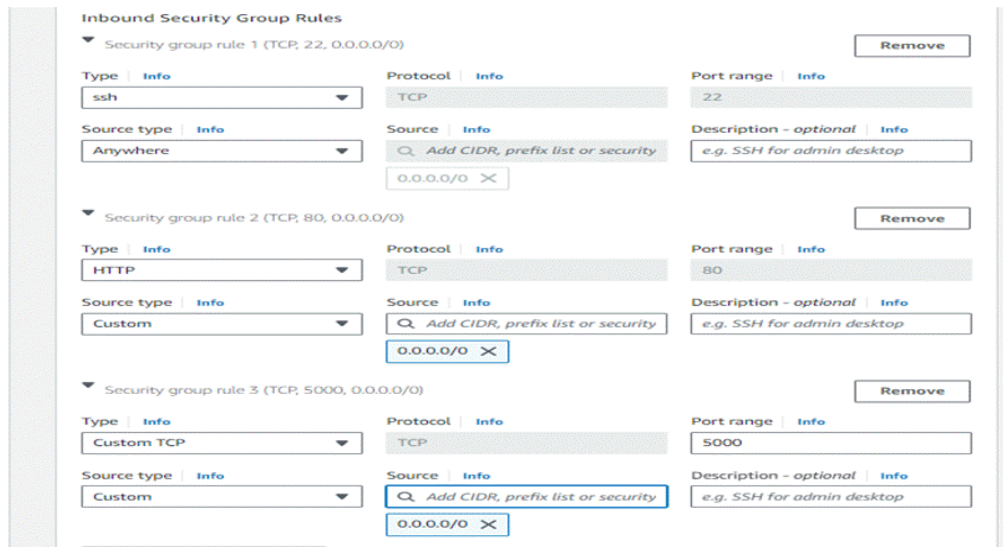
The screenshot shows the 'Create key pair' dialog box in the AWS Management Console. The dialog is titled 'Create key pair' and has a close button (X) in the top right corner. It contains the following sections:

- Key pair name:** A text input field with the value 'Home made'. Below it, a note states: 'The name can include up to 255 ASCII characters. It can't include leading or trailing spaces.'
- Key pair type:** Two radio button options:
  - RSA:** Selected. Description: 'RSA encrypted private and public key pair'.
  - ED25519:** Description: 'ED25519 encrypted private and public key pair'.
- Private key file format:** Two radio button options:
  - .pem:** Selected. Description: 'For use with OpenSSH'.
  - .ppk:** Description: 'For use with PuTTY'.
- Warning:** A yellow box with a warning icon and text: '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](#)'.
- Buttons:** 'Cancel' and 'Create key pair' (highlighted in orange).

In the background, parts of the AWS console are visible, including the 't2.micro' instance details and the 'Key pair (login)' section.

## ● Activity 5.3: Configure Security Groups

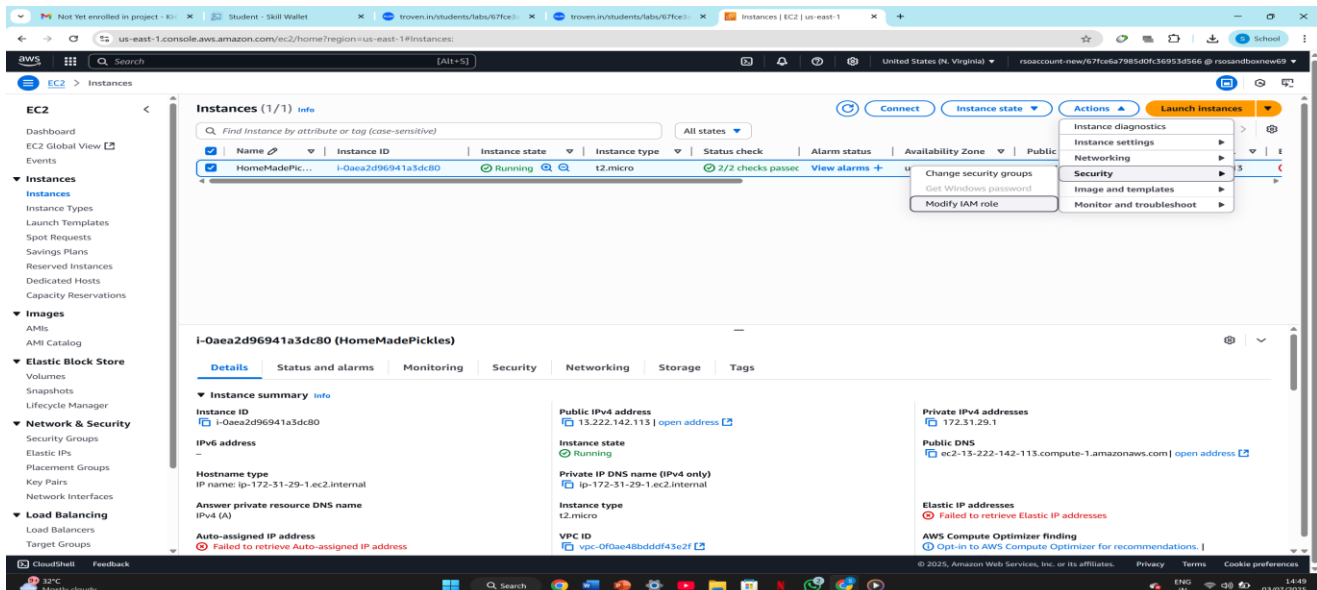
- Allow **HTTP (port 80)** and **SSH (port 22)** inbound traffic.



The screenshot shows the 'Inbound Security Group Rules' configuration page in the AWS Management Console. It displays three security group rules for inbound traffic:

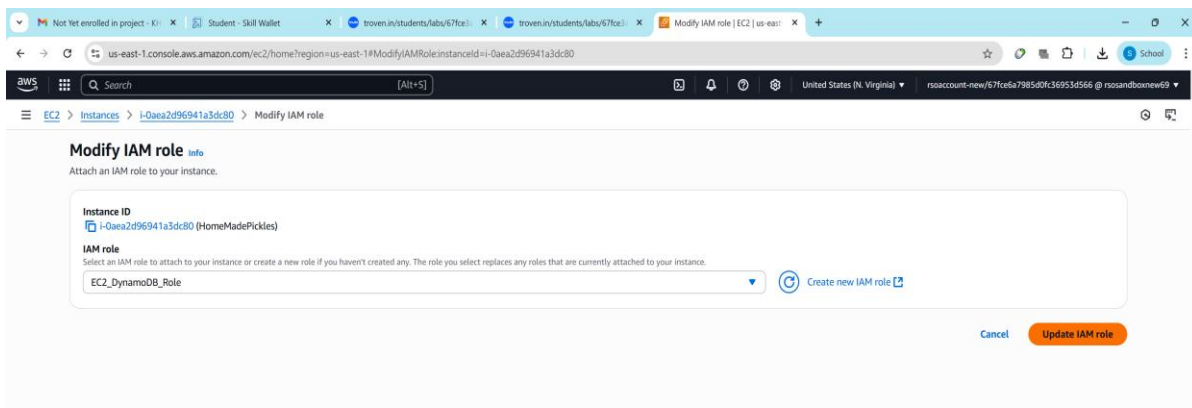
Rule ID	Type	Protocol	Port range	Source type	Source	Description - optional
Security group rule 1 (TCP, 22, 0.0.0.0/0)	ssh	TCP	22	Anywhere	0.0.0.0/0	e.g. SSH for admin desktop
Security group rule 2 (TCP, 80, 0.0.0.0/0)	HTTP	TCP	80	Custom	0.0.0.0/0	e.g. SSH for admin desktop
Security group rule 3 (TCP, 5000, 0.0.0.0/0)	Custom TCP	TCP	5000	Custom	0.0.0.0/0	e.g. SSH for admin desktop

Each rule has a 'Remove' button to its right. The 'Source' field for each rule is currently set to '0.0.0.0/0', which is highlighted with a blue box in the screenshot. The 'Description' field for each rule contains the text 'e.g. SSH for admin desktop'.



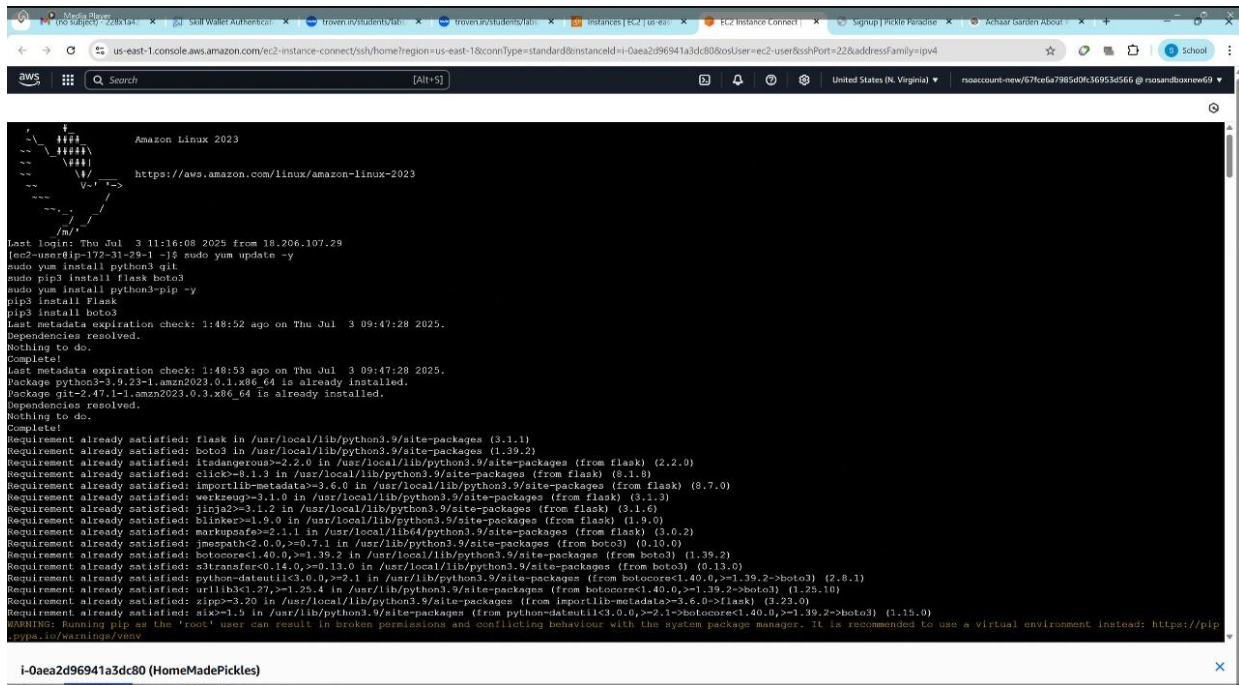
## ● Activity 5.4: Attach IAM Role

- 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.



```

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

last login: Thu Jul 3 11:16:08 2025 from 18.206.107.29
[ec2-user@ip-172-31-29-1 ~]$ sudo yum update -y
sudo yum install python3 git
sudo pip3 install flask boto3
sudo yum install python3-pip -y
pip3 install Flask
pip3 install boto3
Last metadata expiration check: 1:48:52 ago on Thu Jul 3 09:47:28 2025.
Dependencies resolved.
Nothing to do.
Complete!
Last metadata expiration check: 1:48:53 ago on Thu Jul 3 09:47:28 2025.
Package git-2.47.1-1.amzn2023.0.1.x86_64 is already installed.
Dependencies resolved.
Nothing to do.
Complete!
Requirement already satisfied: flask in /usr/local/lib/python3.9/site-packages (3.1.1)
Requirement already satisfied: boto3 in /usr/local/lib/python3.9/site-packages (1.39.2)
Requirement already satisfied: itsdangerous>=2.2.0 in /usr/local/lib/python3.9/site-packages (from flask) (2.2.0)
Requirement already satisfied: click>=8.1.3 in /usr/local/lib/python3.9/site-packages (from flask) (8.1.0)
Requirement already satisfied: importlib-metadata>=3.6.0 in /usr/local/lib/python3.9/site-packages (from flask) (8.7.0)
Requirement already satisfied: werkzeug>=3.1.0 in /usr/local/lib/python3.9/site-packages (from flask) (3.1.3)
Requirement already satisfied: Jinja2>=3.1.2 in /usr/local/lib/python3.9/site-packages (from flask) (3.1.0)
Requirement already satisfied: blinker>=1.9.0 in /usr/local/lib/python3.9/site-packages (from flask) (1.9.0)
Requirement already satisfied: MarkupSafe>=2.1.1 in /usr/local/lib/python3.9/site-packages (from flask) (3.0.2)
Requirement already satisfied: jmespath<2.0.0,>=0.7.1 in /usr/local/lib/python3.9/site-packages (from boto3) (0.10.0)
Requirement already satisfied: botocore<1.40.0,>=1.39.2 in /usr/local/lib/python3.9/site-packages (from boto3) (1.39.2)
Requirement already satisfied: s3transfer<0.14.0,>=0.13.0 in /usr/local/lib/python3.9/site-packages (from boto3) (0.13.0)
Requirement already satisfied: python-dateutil<3.0.0,>=2.1 in /usr/local/lib/python3.9/site-packages (from botocore<1.40.0,>=1.39.2->boto3) (2.8.1)
Requirement already satisfied: urllib3<3.0,>=1.25.4 in /usr/local/lib/python3.9/site-packages (from botocore<1.40.0,>=1.39.2->boto3) (1.25.10)
Requirement already satisfied: zipp>=3.20 in /usr/local/lib/python3.9/site-packages (from importlib-metadata>=3.6.0->flask) (3.23.0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.9/site-packages (from python-dateutil<3.0.0,>=2.1->botocore<1.40.0,>=1.39.2->boto3) (1.15.0)
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
  
```

## 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
          
```



```

AWS
Search [Alt+S]
United States (N. Virginia)
us-east-1.console.aws.amazon.com/ec2-instance-connect/ssh/home?region=us-east-1&connType=standard&instanceId=i-Oaea2d96941a3dc80&source=ec2-user@sshPort=22&addressFamily=ipv4

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

Last login: Thu Jul 3 11:14:08 2025 from 18.206.107.29
[ec2-user@ip-172-31-29-1 ~]$ sudo yum update -y
sudo yum install python3 git
sudo pip3 install flask boto3
sudo yum install python3-pip -y
pip3 install flask
pip3 install boto3
Last metadata expiration check: 1:48:52 ago on Thu Jul 3 09:47:28 2025.
Dependencies resolved.
Nothing to do.
Complete!
Last metadata expiration check: 1:48:53 ago on Thu Jul 3 09:47:28 2025.
Package python3-1.9.23-1.amzn2023.0.1.x86_64 is already installed.
Package git-2.47.1-1.amzn2023.0.3.x86_64 is already installed.
Dependencies resolved.
Nothing to do.
Complete!
Requirement already satisfied: flask in /usr/local/lib/python3.9/site-packages (3.1.1)
Requirement already satisfied: boto3 in /usr/local/lib/python3.9/site-packages (1.39.2)
Requirement already satisfied: itsdangerous<2.2.0 in /usr/local/lib/python3.9/site-packages (from flask) (2.2.0)
Requirement already satisfied: click>8.1.3 in /usr/local/lib/python3.9/site-packages (from flask) (8.1.8)
Requirement already satisfied: importlib-metadata>3.6.0 in /usr/local/lib/python3.9/site-packages (from flask) (8.7.0)
Requirement already satisfied: werkzeug>3.1.0 in /usr/local/lib/python3.9/site-packages (from flask) (3.1.3)
Requirement already satisfied: Jinja2>3.1.2 in /usr/local/lib/python3.9/site-packages (from flask) (3.1.6)
Requirement already satisfied: blinker<1.9.0 in /usr/local/lib/python3.9/site-packages (from flask) (1.5.0)
Requirement already satisfied: MarkupSafe>2.1.1 in /usr/local/lib/python3.9/site-packages (from flask) (3.0.2)
Requirement already satisfied: botocore<1.40.0, >=1.39.2 in /usr/local/lib/python3.9/site-packages (from boto3) (1.39.2)
Requirement already satisfied: s3transfer<0.14.0, >=0.13.0 in /usr/local/lib/python3.9/site-packages (from boto3) (0.13.0)
Requirement 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)
Requirement 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)
Requirement already satisfied: sipp>3.20 in /usr/local/lib/python3.9/site-packages (from importlib-metadata>3.6.0->flask) (3.23.0)
Requirement 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)
WARNING: Running 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

i-Oaea2d96941a3dc80 (HomeMadePickles)
  
```

## • Activity 6.2: Clone Flask Project from GitHub

- Run: git clone <https://github.com/MythiliKancharla/pickleshomemade.git>
- Navigate to the project folder: cd [pickleshomemade](#)

```

AWS
Search [Alt+S]
United States (N. Virginia)
us-east-1.console.aws.amazon.com/ec2-instance-connect/ssh/home?region=us-east-1&connType=standard&instanceId=i-Oaea2d96941a3dc80&source=ec2-user@sshPort=22&addressFamily=ipv4

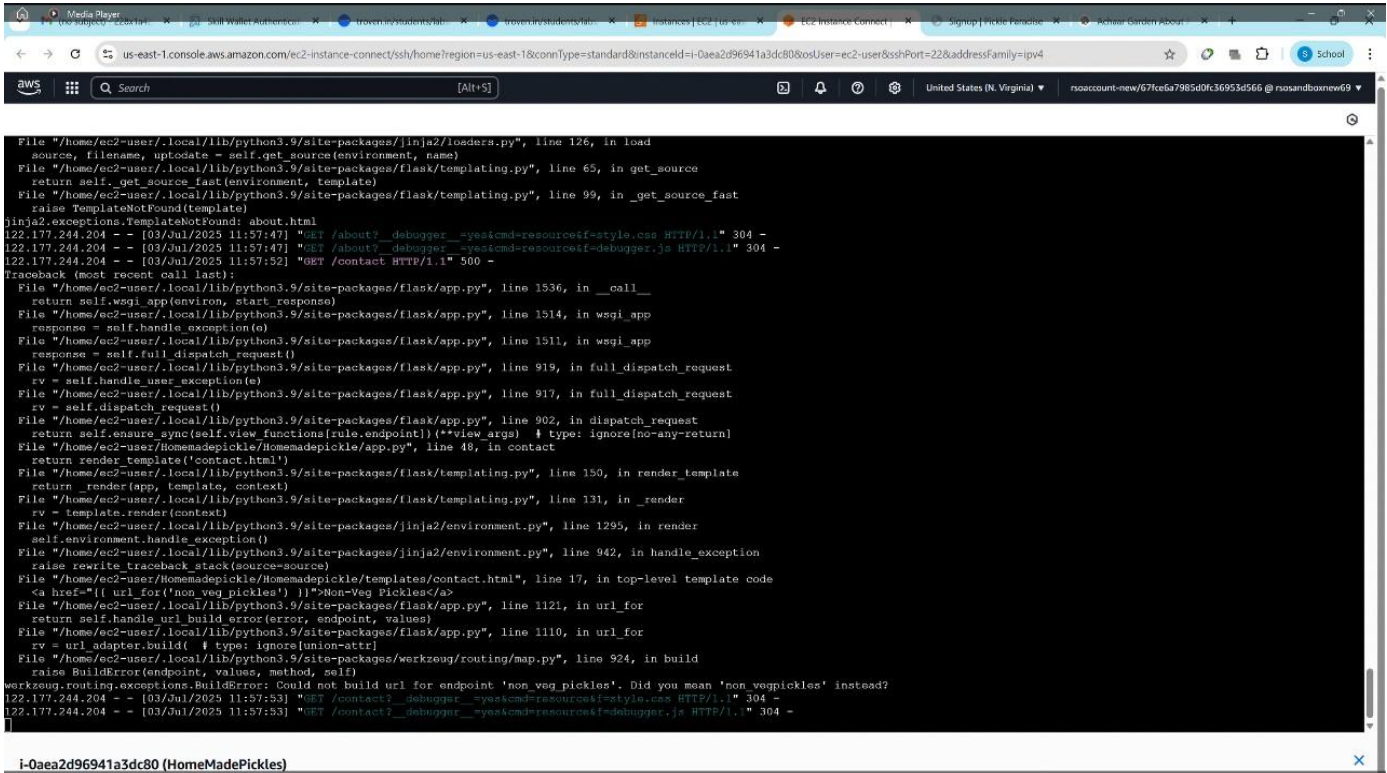
Traceback (most recent call last):
  File "/home/ec2-user/.local/lib/python3.9/site-packages/flask/app.py", line 1536, in __call__
    return self.wsgi_app(environ, start_response)
  File "/home/ec2-user/.local/lib/python3.9/site-packages/flask/app.py", line 1514, in wsgi_app
    response = self.handle_exception(e)
  File "/home/ec2-user/.local/lib/python3.9/site-packages/flask/app.py", line 1511, in wsgi_app
    response = self.full_dispatch_request()
  File "/home/ec2-user/.local/lib/python3.9/site-packages/flask/app.py", line 919, in full_dispatch_request
    rv = self.handle_user_exception(e)
  File "/home/ec2-user/.local/lib/python3.9/site-packages/flask/app.py", line 917, in full_dispatch_request
    rv = self.dispatch_request()
  File "/home/ec2-user/.local/lib/python3.9/site-packages/flask/app.py", line 902, in dispatch_request
    return self.ensure_sync(self.view_functions[rule.endpoint])(**view_args)  # type: ignore[no-any-return]
  File "/home/ec2-user/~/HomemadePickles/HomemadePickles/app.py", line 198, in about
    return render_template("about.html")
  File "/home/ec2-user/.local/lib/python3.9/site-packages/flask/templating.py", line 149, in render_template
    template = app.jinja_env.get_or_select_template(template_name_or_list)
  File "/home/ec2-user/.local/lib/python3.9/site-packages/jinja2/environment.py", line 1087, in get_or_select_template
    return self.get_template(template_name_or_list, parent, globals)
  File "/home/ec2-user/.local/lib/python3.9/site-packages/jinja2/environment.py", line 1016, in get_template
    return self._load_template(name, globals)
  File "/home/ec2-user/.local/lib/python3.9/site-packages/jinja2/environment.py", line 975, in _load_template
    template = self.loader.load(self, name, self.make_globals(globals))
  File "/home/ec2-user/.local/lib/python3.9/site-packages/jinja2/loaders.py", line 126, in load
    source, filename, uptodate = self.get_source(environment, name)
  File "/home/ec2-user/.local/lib/python3.9/site-packages/flask/templating.py", line 65, in get_source
    return self._get_source_fast(environment, template)
  File "/home/ec2-user/.local/lib/python3.9/site-packages/flask/templating.py", line 99, in _get_source_fast
    raise TemplateNotFound(template)
jinja2.exceptions.TemplateNotFound: about.html
122.177.244.204 - [03/Jul/2025 11:44:54] "GET /about?_debugger__yes&cmd=resource&ef=style.css HTTP/1.1" 200 -
122.177.244.204 - [03/Jul/2025 11:44:54] "GET /about?_debugger__yes&cmd=resource&ef=debugger.js HTTP/1.1" 200 -
122.177.244.204 - [03/Jul/2025 11:44:55] "GET /about?_debugger__yes&cmd=resource&ef=console.png HTTP/1.1" 200 -
122.177.244.204 - [03/Jul/2025 11:47:17] "GET /about HTTP/1.1" 500
Traceback (most recent call last):
  File "/home/ec2-user/.local/lib/python3.9/site-packages/flask/app.py", line 1536, in __call__
    return self.wsgi_app(environ, start_response)
  File "/home/ec2-user/.local/lib/python3.9/site-packages/flask/app.py", line 1514, in wsgi_app
    response = self.handle_exception(e)
  File "/home/ec2-user/.local/lib/python3.9/site-packages/flask/app.py", line 1511, in wsgi_app
    response = self.full_dispatch_request()
  File "/home/ec2-user/.local/lib/python3.9/site-packages/flask/app.py", line 919, in full_dispatch_request
    rv = self.handle_user_exception(e)
  File "/home/ec2-user/.local/lib/python3.9/site-packages/flask/app.py", line 917, in full_dispatch_request
    rv = self.dispatch_request()
  
```

i-Oaea2d96941a3dc80 (HomeMadePickles)

PublicIPs: 15.222.162.115 PrivateIPs: 172.31.20.1

- **Activity 6.3: Run the Flask Application**

- Run: `python3 app.py`



```
File "/home/ec2-user/.local/lib/python3.9/site-packages/jinja2/loaders.py", line 126, in load
    source, filename, uptodate = self.get_source(environment, name)
File "/home/ec2-user/.local/lib/python3.9/site-packages/flask/templating.py", line 65, in get_source
    return self._get_source_fast(environment, template)
File "/home/ec2-user/.local/lib/python3.9/site-packages/flask/templating.py", line 99, in _get_source_fast
    raise TemplateNotFound(template)
jinja2.exceptions.TemplateNotFound: about.html
122.177.244.204 - - [03/Jul/2025 11:57:47] "GET /about?_debugger__yes&cmd=resource&f=style.css HTTP/1.1" 304 -
122.177.244.204 - - [03/Jul/2025 11:57:47] "GET /about?_debugger__yes&cmd=resource&f=debugger.js HTTP/1.1" 304 -
122.177.244.204 - - [03/Jul/2025 11:57:52] "GET /contact HTTP/1.1" 500 -
Traceback (most recent call last):
  File "/home/ec2-user/.local/lib/python3.9/site-packages/flask/app.py", line 1536, in __call__
    return self.wsgi_app(environ, start_response)
  File "/home/ec2-user/.local/lib/python3.9/site-packages/flask/app.py", line 1514, in wsgi_app
    response = self.handle_exception(e)
  File "/home/ec2-user/.local/lib/python3.9/site-packages/flask/app.py", line 1511, in wsgi_app
    response = self.full_dispatch_request()
  File "/home/ec2-user/.local/lib/python3.9/site-packages/flask/app.py", line 919, in full_dispatch_request
    rv = self.handle_user_exception(e)
  File "/home/ec2-user/.local/lib/python3.9/site-packages/flask/app.py", line 917, in full_dispatch_request
    rv = self.dispatch_request()
  File "/home/ec2-user/.local/lib/python3.9/site-packages/flask/app.py", line 902, in dispatch_request
    return self.ensure_sync(self.view_functions[rule.endpoint])(**view_args)  # type: ignore[no-any-return]
  File "/home/ec2-user/.local/lib/python3.9/site-packages/flask/app.py", line 48, in contact
    return render_template("contact.html")
  File "/home/ec2-user/.local/lib/python3.9/site-packages/flask/templating.py", line 150, in render_template
    return _render(app, template, context)
  File "/home/ec2-user/.local/lib/python3.9/site-packages/flask/templating.py", line 131, in _render
    rv = template.render(context)
  File "/home/ec2-user/.local/lib/python3.9/site-packages/jinja2/environment.py", line 1295, in render
    self.environment.handle_exception()
  File "/home/ec2-user/.local/lib/python3.9/site-packages/jinja2/environment.py", line 942, in handle_exception
    raise rewrite_traceback_stack(source=source)
  File "/home/ec2-user/.local/lib/python3.9/site-packages/flask/templating.py", line 17, in top-level template code
    <a href="{% url for 'non_veg_pickles' %}" %}>Non-Veg Pickles</a>
  File "/home/ec2-user/.local/lib/python3.9/site-packages/flask/app.py", line 1121, in url_for
    return self.handle_url_build_error(error, endpoint, values)
  File "/home/ec2-user/.local/lib/python3.9/site-packages/flask/app.py", line 1110, in url_for
    rv = url_adapter.build(  # type: ignore[union-attr]
  File "/home/ec2-user/.local/lib/python3.9/site-packages/werkzeug/routing/map.py", line 924, in build
    raise BuildError(endpoint, values, method, self)
werkzeug.routing.exceptions.BuildError: Could not build url for endpoint 'non_veg_pickles'. Did you mean 'non_veg_pickles' instead?
122.177.244.204 - - [03/Jul/2025 11:57:53] "GET /contact?_debugger__yes&cmd=resource&f=style.css HTTP/1.1" 304 -
122.177.244.204 - - [03/Jul/2025 11:57:53] "GET /contact?_debugger__yes&cmd=resource&f=debugger.js HTTP/1.1" 304 -
```

- **Activity 6.6: Access the Website**

- Open your browser and go to: <http://13.222.142.113:5000>

## 7: Testing and Deployment

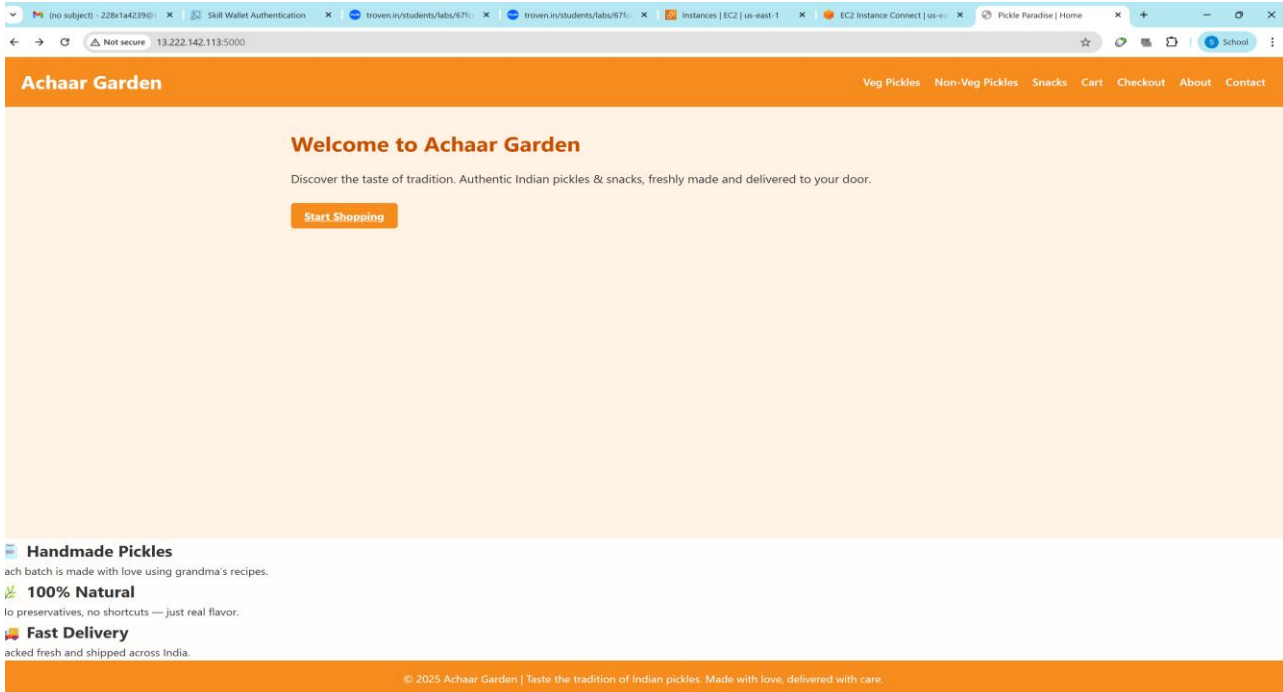
### Milestone 7: Testing and Deployment

- **Activity 7.1: Functional Testing to Verify the Project**

- Test each of the following pages for proper functionality, navigation, and data flow:

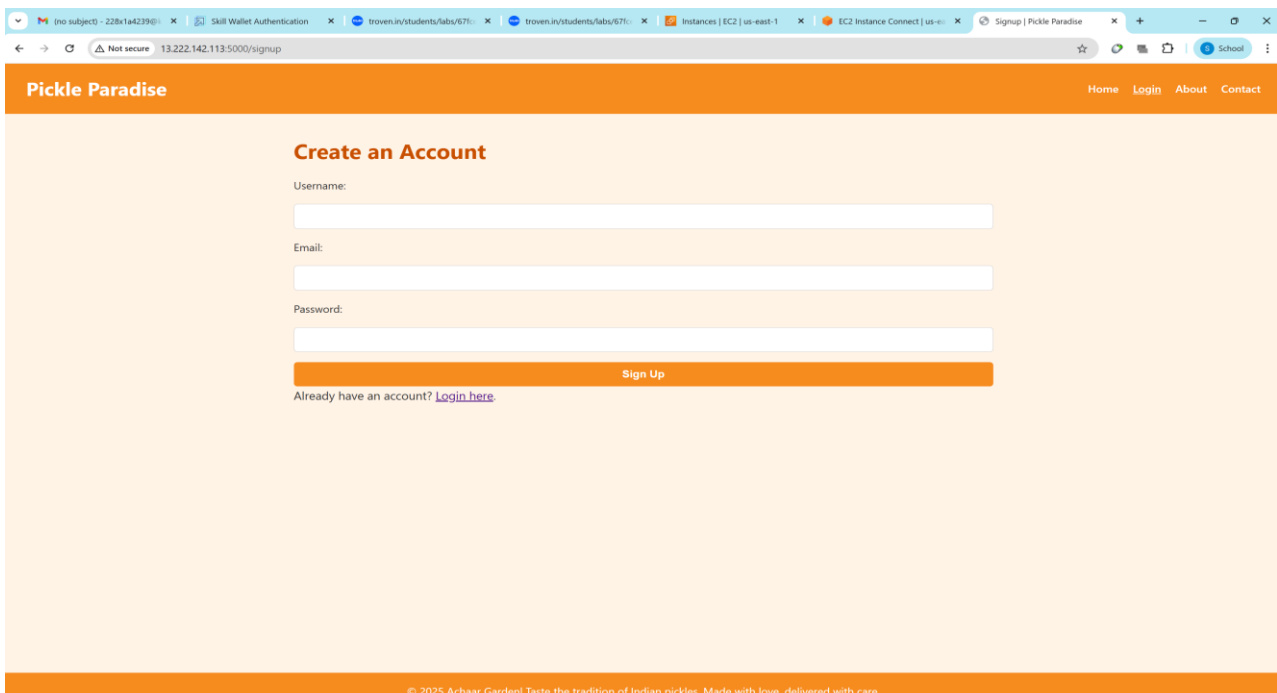


## Home Page:



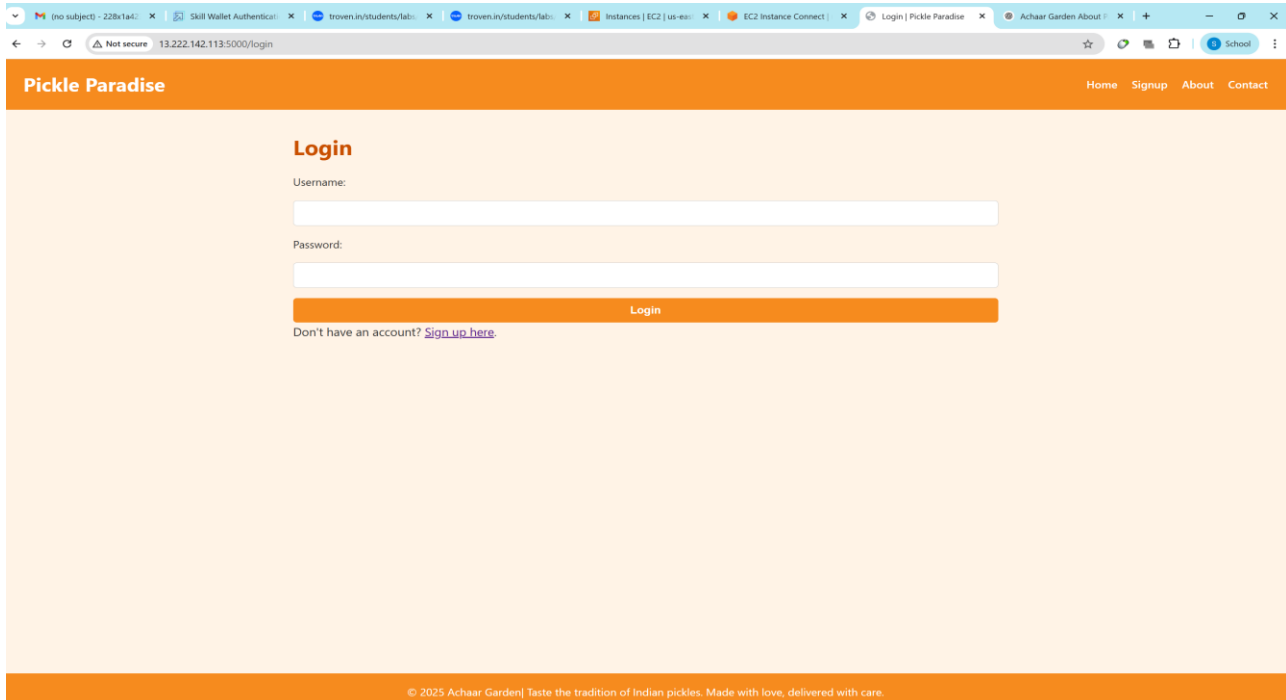
The screenshot shows the Achaar Garden website. The browser's address bar displays "13.222.142.113:5000". The website has an orange header with the name "Achaar Garden" on the left and navigation links "Veg Pickles", "Non-Veg Pickles", "Snacks", "Cart", "Checkout", "About", and "Contact" on the right. The main content area is a light orange color. It features a "Welcome to Achaar Garden" heading, followed by the text "Discover the taste of tradition. Authentic Indian pickles & snacks, freshly made and delivered to your door." and a "Start Shopping" button. Below this, there are three sections: "Handmade Pickles" with the text "Each batch is made with love using grandma's recipes.", "100% Natural" with the text "No preservatives, no shortcuts — just real flavor.", and "Fast Delivery" with the text "Packed fresh and shipped across India." The footer is orange and contains the text "© 2025 Achaar Garden | Taste the tradition of Indian pickles. Made with love, delivered with care."

## Signup Page:



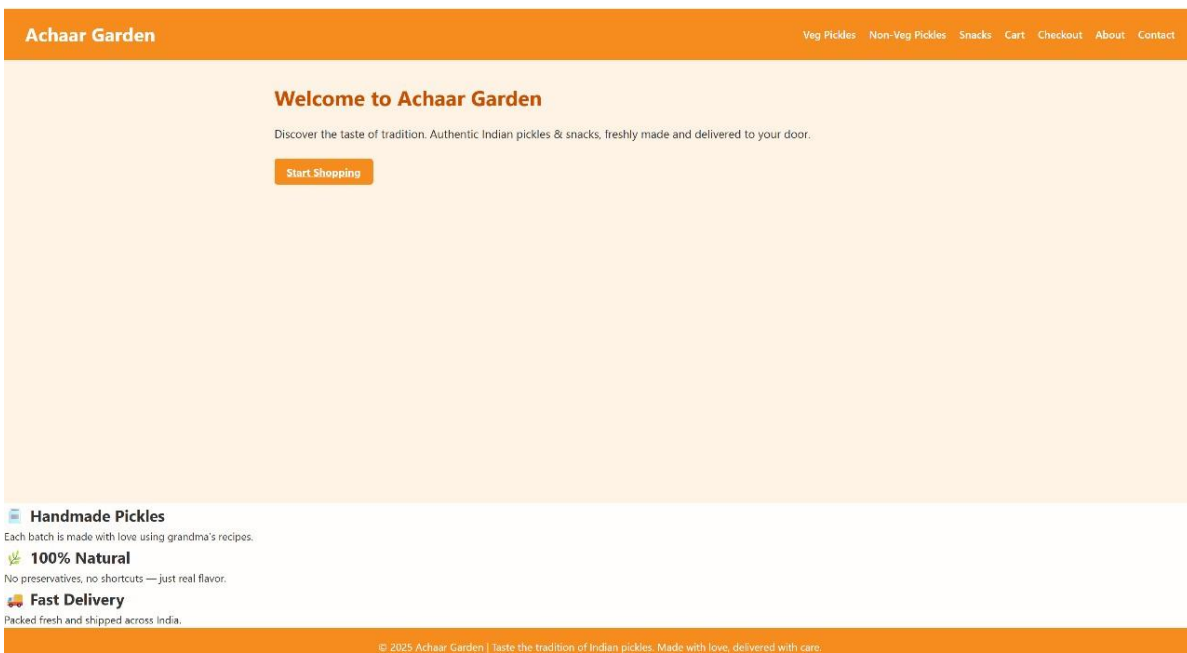
The screenshot shows the Pickle Paradise website's signup page. The browser's address bar displays "13.222.142.113:5000/signup". The website has an orange header with the name "Pickle Paradise" on the left and navigation links "Home", "Login", "About", and "Contact" on the right. The main content area is a light orange color. It features a "Create an Account" heading. Below this, there are three input fields for "Username:", "Email:", and "Password:". A "Sign Up" button is positioned below the password field. At the bottom of the form, there is a link: "Already have an account? [Login here.](#)" The footer is orange and contains the text "© 2025 Achaar Garden | Taste the tradition of Indian pickles. Made with love, delivered with care."

## Login Page:



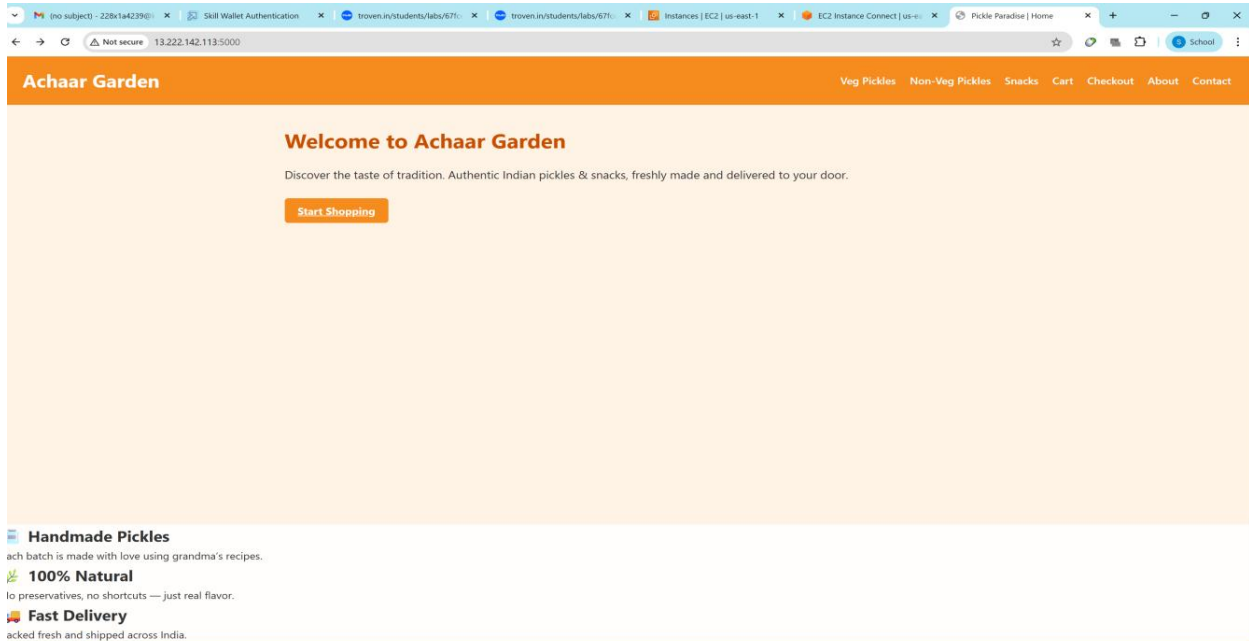
The screenshot shows a web browser window with multiple tabs. The active tab is 'Login | Pickle Paradise'. The address bar shows '13.222.142.113:5000/login'. The page has an orange header with 'Pickle Paradise' on the left and 'Home Signup About Contact' on the right. The main content area is light orange and contains a 'Login' section with a title, 'Username:' label, a text input field, 'Password:' label, another text input field, an orange 'Login' button, and a link 'Don't have an account? [Sign up here](#)'. The footer is orange with the text '© 2025 Achaar Garden | Taste the tradition of Indian pickles. Made with love, delivered with care.'

## About Page:



The screenshot shows a web browser window with multiple tabs. The active tab is 'Achaar Garden About'. The address bar shows '13.222.142.113:5000/about'. The page has an orange header with 'Achaar Garden' on the left and 'Veg Pickles Non-Veg Pickles Snacks Cart Checkout About Contact' on the right. The main content area is light orange and contains a 'Welcome to Achaar Garden' section with a title, a paragraph 'Discover the taste of tradition. Authentic Indian pickles & snacks, freshly made and delivered to your door.', and an orange 'Start Shopping' button. Below this is a 'Handmade Pickles' section with a title, a paragraph 'Each batch is made with love using grandma's recipes.', and two sub-sections: '100% Natural' with a leaf icon and 'No preservatives, no shortcuts — just real flavor.' and 'Fast Delivery' with a truck icon and 'Packed fresh and shipped across India.'. The footer is orange with the text '© 2025 Achaar Garden | Taste the tradition of Indian pickles. Made with love, delivered with care.'

## Veg Pickles Page:



The screenshot shows a web browser window with multiple tabs. The active tab is 'Pickle Paradise | Home'. The address bar shows '13.222.142.113:5000'. The website has an orange header with the text 'Achaar Garden' on the left and a navigation menu on the right: 'Veg Pickles', 'Non-Veg Pickles', 'Snacks', 'Cart', 'Checkout', 'About', and 'Contact'. The main content area has a large orange banner with the text 'Welcome to Achaar Garden' and a subtext 'Discover the taste of tradition. Authentic Indian pickles & snacks, freshly made and delivered to your door.' Below this is a 'Start Shopping' button. At the bottom, there are three sections: 'Handmade Pickles' with a description 'Each batch is made with love using grandma's recipes.', '100% Natural' with a description 'No preservatives, no shortcuts — just real flavor.', and 'Fast Delivery' with a description 'Packed fresh and shipped across India.'

**Achaar Garden** Veg Pickles Non-Veg Pickles Snacks Cart Checkout About Contact

### Welcome to Achaar Garden

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

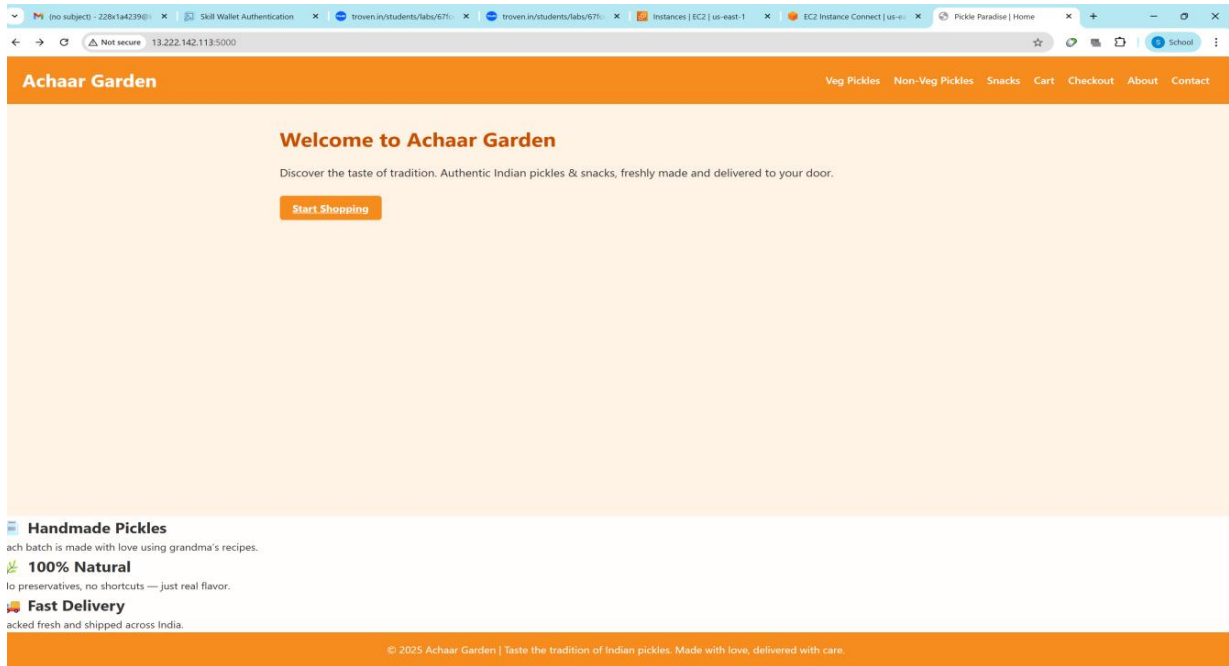
[Start Shopping](#)

**Handmade Pickles**  
Each batch is made with love using grandma's recipes.

**100% Natural**  
No preservatives, no shortcuts — just real flavor.

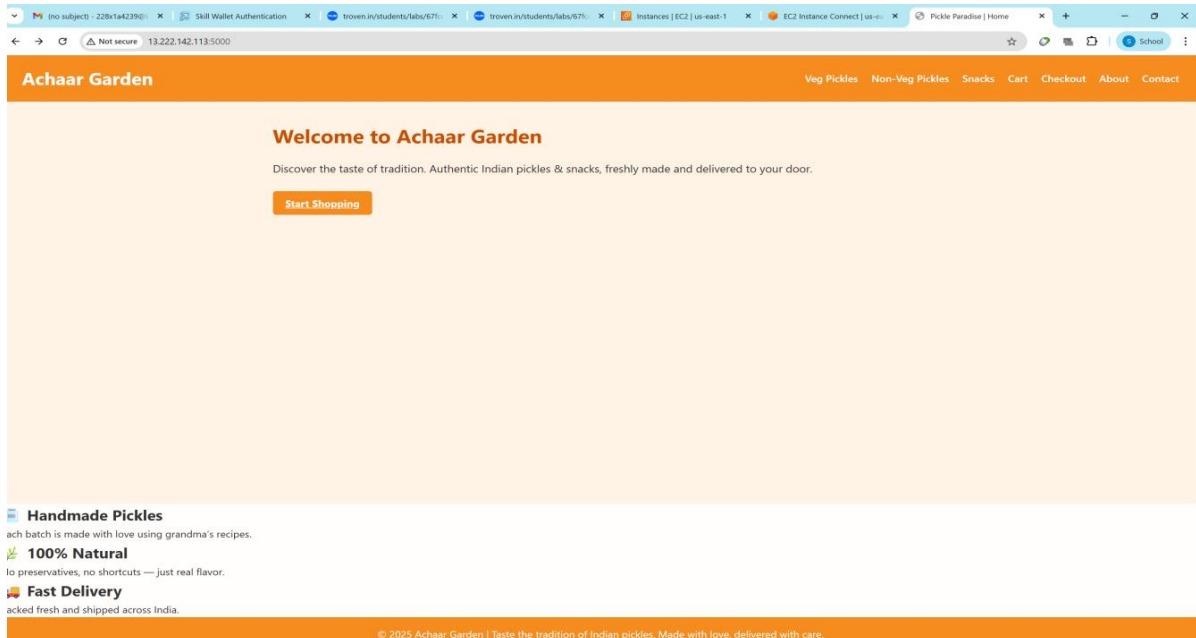
**Fast Delivery**  
Packed fresh and shipped across India.

## Non-Veg Pickles Page:

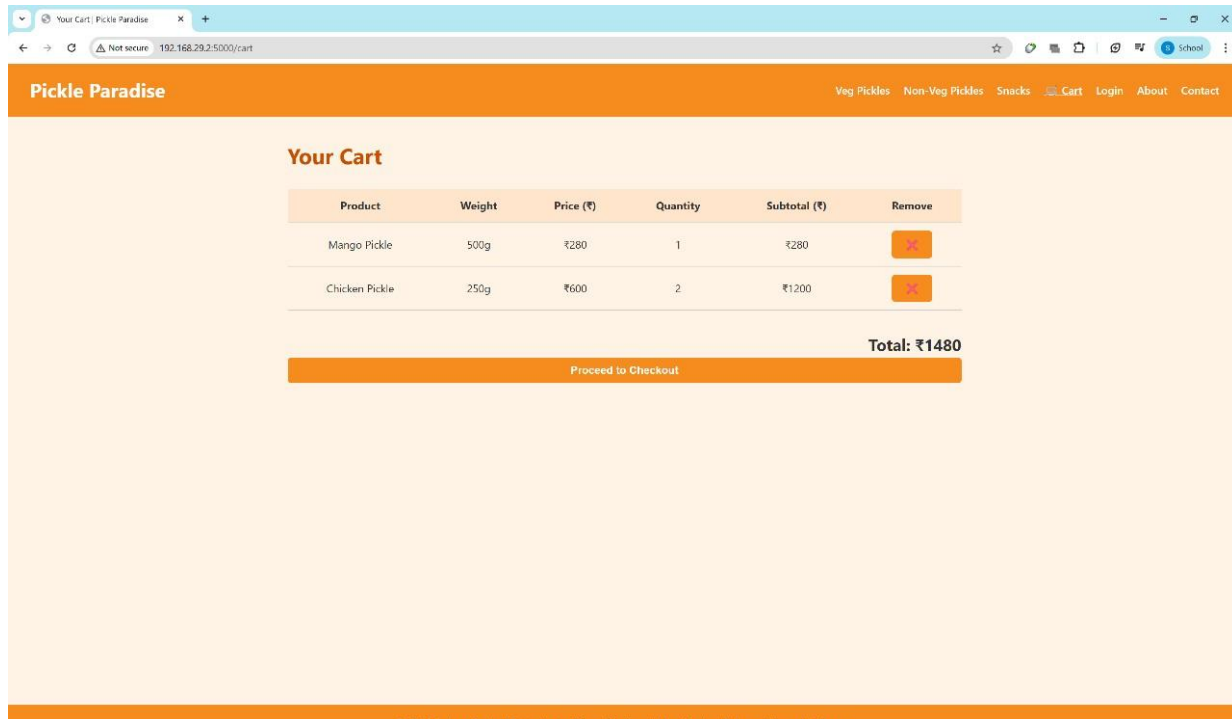


The screenshot shows a web browser window with multiple tabs. The active tab is titled "Pickle Paradise | Home". The address bar shows "13.222.142.113:5000". The website header is orange with the text "Achaar Garden" on the left and navigation links "Veg Pickles", "Non-Veg Pickles", "Snacks", "Cart", "Checkout", "About", and "Contact" on the right. The main content area has a large orange banner with the text "Welcome to Achaar Garden" and "Discover the taste of tradition. Authentic Indian pickles & snacks, freshly made and delivered to your door." Below this is a "Start Shopping" button. The footer section is white with three icons and text: "Handmade Pickles" (ach batch is made with love using grandma's recipes), "100% Natural" (No preservatives, no shortcuts — just real flavor), and "Fast Delivery" (Packaged fresh and shipped across India). The bottom orange bar contains the copyright notice: "© 2025 Achaar Garden | Taste the tradition of Indian pickles. Made with love, delivered with care."

## Snacks page:



## Cart Page:



## Success Page:

