

Home Made Pickles & Snacks: Taste the Best – Cloud-Powered E-Commerce Platform

Project Description

The growing demand for authentic homemade pickles and snacks has highlighted the need for an efficient, scalable e-commerce solution to cater to customers across India. The **Home Made Pickles & Snacks** platform addresses this need by providing a modern web-based application that enables customers to browse, add products to the cart, and place orders seamlessly.

This project uses **Flask** for backend development, **AWS DynamoDB** for storing user and order data, **AWS SNS** for real-time order notifications, and **EC2** for deployment. The platform offers an intuitive UI for customers to explore Veg Pickles, Non-Veg Pickles, and Snacks with pricing and cart functionality.

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.

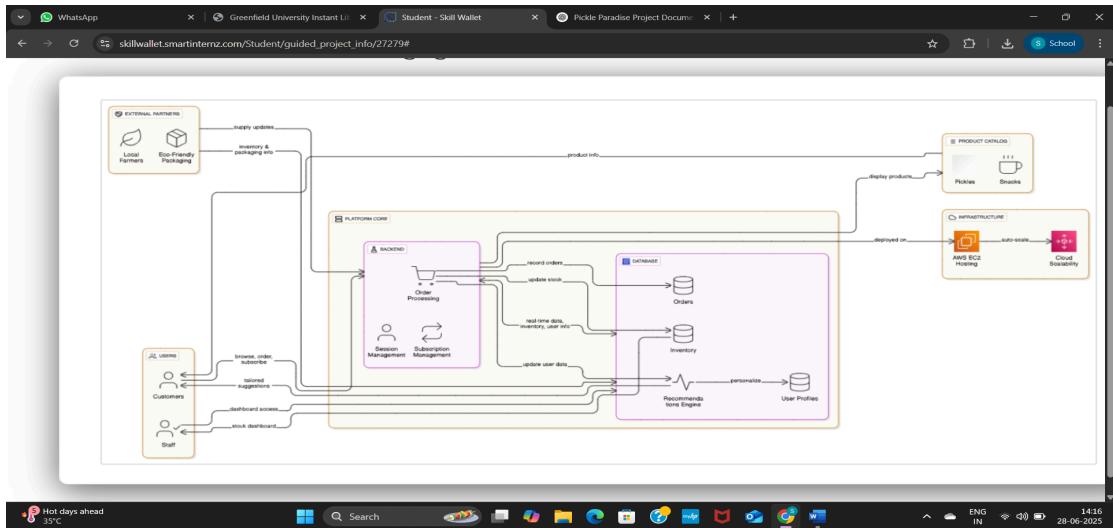
Scenario 2: Real-Time Inventory Tracking and Updates

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.

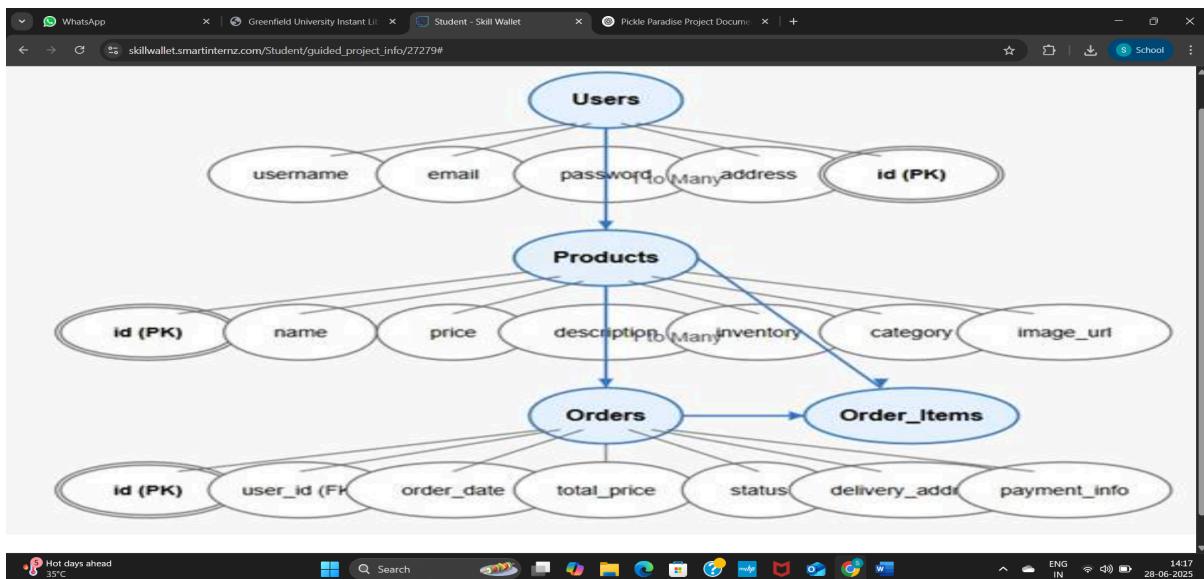
Scenario 3: Personalized User Experience and Recommendations

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: [AWS Account Setup](#)
- AWS IAM (Identity and Access Management): [AWS IAM](#)
- AWS EC2 (Elastic Compute Cloud): [EC2](#)
- AWS DynamoDB: [Dynamodb](#)
- Git Documentation: [Git Documentation](#)
- VS Code Installation: [VS Code download](#)

Project WorkFlow:

Milestone 1. Backend Development and Application Setup

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

Milestone 2. AWS Account Setup and Login

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

Milestone 3. DynamoDB Database Creation and Setup

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

Milestone 4. SNS Notification Setup

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

Milestone 5. IAM Role Setup

- Create IAM Role
- Attach Policies

Milestone 6. EC2 Instance Setup

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

Milestone 7. Deployment on EC2

- Upload Flask Files
- Run the Flask App

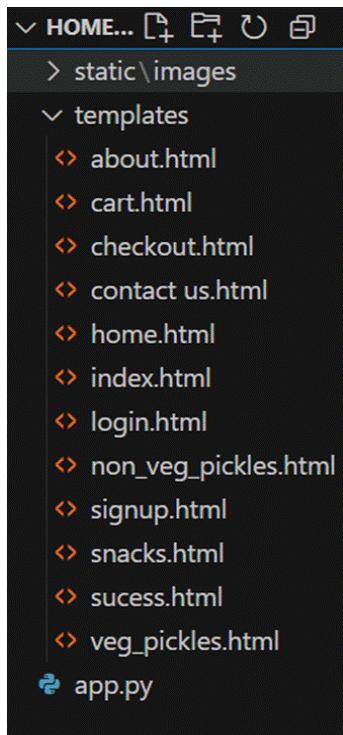
Milestone 8. Testing and Deployment

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

Milestone 1. Backend Development and Application Setup

Develop the Backend Using Flask.

File Explorer Structure:



Description of the code :

? Flask App Initialization

```
app.py > ...
1  from flask import Flask, render_template, request, redirect, url_for, session, flash
2  import boto3
3  import uuid
4  import smtplib
5  from email.mime.text import MIMEText
6  from email.mime.multipart import MIMEMultipart
7  import os
8
app = Flask(__name__)
```

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

```

11
12 # AWS Configuration
13 dynamodb = boto3.resource('dynamodb', region_name='us-east-1')
14 user_table = dynamodb.Table('Users')
15 orders_table = dynamodb.Table('PickleOrders')
16

```

```

# Product Data - Unchanged
veg_items = [
    {"name": "Mango Pickle", "price": 150, "img": "mango-pickle.jpg"}, 
    {"name": "Lemon Pickle", "price": 120, "img": "lemon-pickle.jpg"}, 
    {"name": "Tomato Pickle", "price": 180, "img": "tomato-pickle.jpg"}, 
    {"name": "Gongura Pickle", "price": 160, "img": "gongura-pickle.jpg"}, 
]

non_veg_items = [
    {"name": "Chicken Pickle", "price": 300, "img": "chicken-pickle.jpg"}, 
    {"name": "Mutton Pickle", "price": 350, "img": "mutton-pickle.jpg"}, 
    {"name": "Fish Pickle", "price": 320, "img": "fish-pickle.jpg"}, 
    {"name": "Prawn Pickle", "price": 340, "img": "prawn-pickle.jpg"}, 
]

snack_items = [
    {"name": "Snack Combo", "price": 200, "img": "snacks-combo.jpg"}, 
    {"name": "Murukku", "price": 100, "img": "murukku.jpg"}, 
    {"name": "Mixture", "price": 120, "img": "mixture.jpg"}, 
    {"name": "Chakli", "price": 130, "img": "chakli.jpg"}, 
]

```

- Routes for Web Pages

Login Route (GET/POST): Verifies user credentials, increments login count, and redirects to the dashboard on success

```

@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        email = request.form['email']
        password = request.form['password']
        user = user_table.get_item(Key={'email': email}).get('Item')
        if user and user['password'] == password:
            session['user'] = email
            flash("Login successful!", "success")
            return redirect(url_for('index'))
        flash("Invalid credentials", "danger")
    return render_template('login.html')

@app.route('/signup', methods=['GET', 'POST'])

```

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

```
def signup():
    if request.method == 'POST':
        email = request.form['email']
        password = request.form['password']
        user_table.put_item(Item={'email': email, 'password': password})
        send_email(email, "Welcome to Pickle Paradise", "Thank you for signing up!")
        flash("Signup successful! Please login.", "success")
        return redirect(url_for('login'))
    return render_template('signup.html')
```

Activate Wind
Go to Settings to a

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

```
150
151     @app.route('/logout')
152     def logout():
153         session.pop('user', None)
154         return redirect(url_for('index'))
155
```

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

```
42
43     # Routes
44     @app.route('/')
45     def index():
46         return render_template('index.html')
47
48     @app.route('/veg-pickles')
49     def veg_pickles():
50         return render_template('veg_pickles.html', items=veg_items)
51
52     @app.route('/non-veg-pickles')
53     def non_veg_pickles():
54         return render_template('non_veg_pickles.html', items=non_veg_items)
55
56     @app.route('/snacks')
57     def snacks():
58         return render_template('snacks.html', items=snack_items)
59
```

Check out Route:

```

81     return render_template('cart.html', cart=cart_items, total=total)
82
83 @app.route('/checkout', methods=['GET', 'POST'])
84 def checkout():
85     if request.method == 'POST':
86         name = request.form['fullname']
87         email = request.form['email']
88         address = request.form['address']
89         phone = request.form['phone']
90         payment = request.form['payment']
91         cart_items = session.get('cart', [])
92         total = sum(int(i['price']) for i in cart_items)
93         order_id = str(uuid.uuid4())
94
95         orders_table.put_item(Item={
96             'order_id': order_id,
97             'name': name,
98             'email': email,
99             'address': address,
100            'phone': phone,
101            'payment': payment,
102            'total': total,
103            'items': cart_items
104        })
105
106     send_email(email, "Order Confirmation", f"Thank you {name} for your order! Total")
107
108     session.pop('cart', None)
109     return render_template('success.html', name=name, order_id=order_id)
110
111     return render_template('checkout.html')

```

Activate Win
Go to Settings to...

```

9
0     @app.route('/add_to_cart_route', methods=['POST'])
1     def add_to_cart_route():
2         item = request.form.to_dict()
3         cart = session.get('cart', [])
4         cart.append(item)
5         session['cart'] = cart
6         return redirect(request.referrer)
7
8     @app.route('/remove_from_cart', methods=['POST'])
9     def remove_from_cart():
0         index = int(request.form['index'])
1         cart = session.get('cart', [])
2         if 0 <= index < len(cart):
3             cart.pop(index)
4         session['cart'] = cart
5         return redirect(url_for('cart'))
6
7     @app.route('/cart')
8     def cart():

```

```

@app.route('/sucess')
def success():
    return render_template('sucess.html')

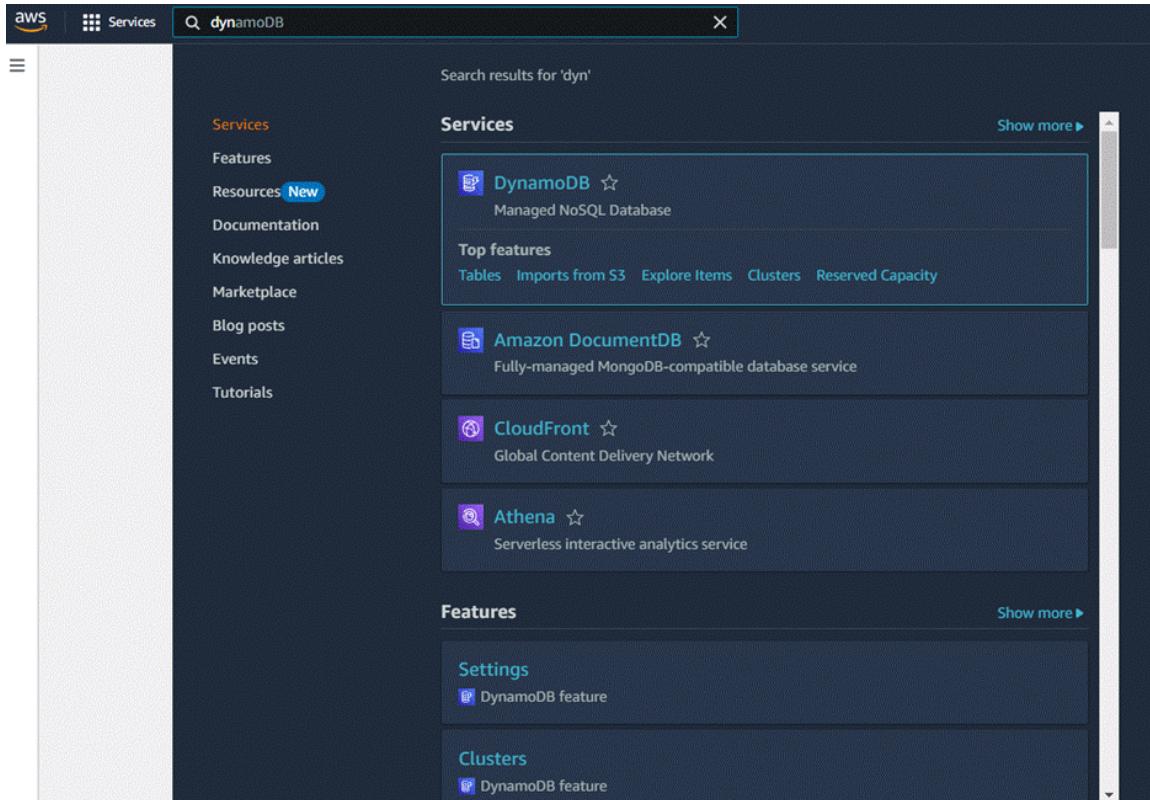
```

Milestone 2. AWS Account Setup and Login

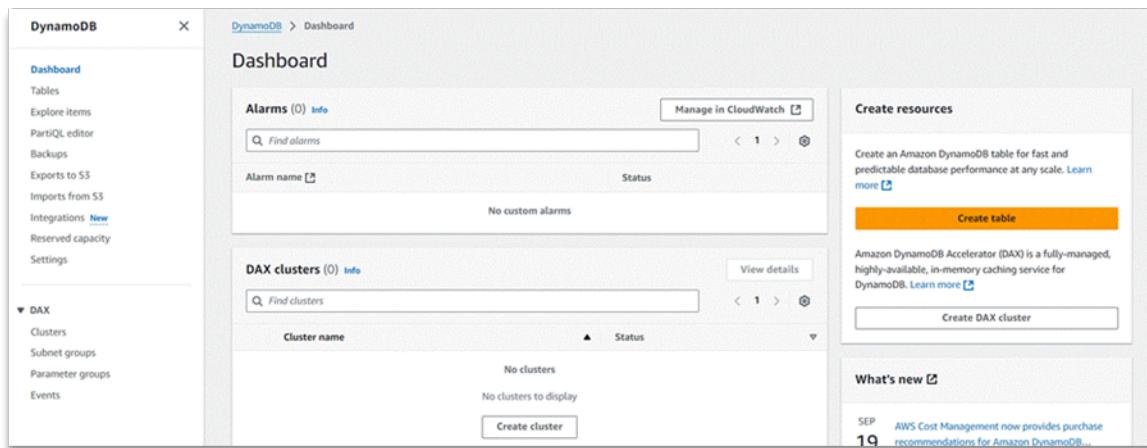
Milestone 3 : DynamoDB Database Creation and Setup

Navigate to the DynamoDB

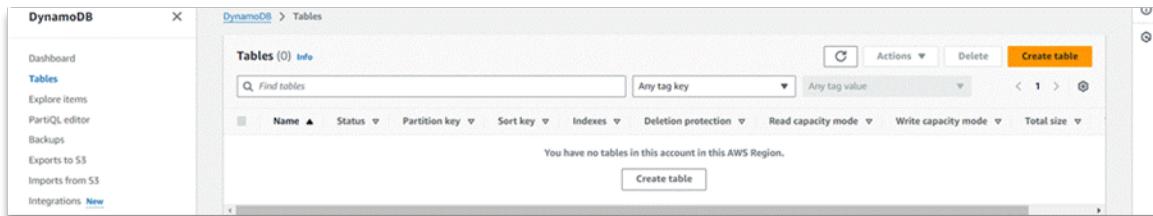
- In the AWS Console, navigate to DynamoDB and click on create tables.



The screenshot shows the AWS Services search results page. The search bar at the top contains "dynamoDB". The results are listed under the "Services" section. The first result is "DynamoDB" with the subtext "Managed NoSQL Database". Below it are other services: "Amazon DocumentDB", "CloudFront", and "Athena". There are also sections for "Features" (Settings, Clusters) and "Tutorials".



The screenshot shows the DynamoDB Dashboard. On the left, there's a sidebar with links like "Tables", "Explore items", "PartiQL editor", etc. The main dashboard area has sections for "Alarms" (0), "DAX clusters" (0), and "What's new". On the right, there's a "Create resources" section with a prominent orange "Create table" button. Below it, there's information about Amazon DynamoDB Accelerator (DAX) and a "Create DAX cluster" button. A "What's new" section at the bottom right mentions "AWS Cost Management now provides purchase recommendations for Amazon DynamoDB...".



Create a DynamoDB table for storing data

- Create Users table with partition key “Username” with type String and click on create table

The screenshot shows the 'Create table' wizard. At the top, there's a breadcrumb navigation: aws | DynamoDB > Tables > Create table. Below it, the title 'Create table' is displayed. The first section is 'Table details' with a sub-link 'Info'. It contains a note: 'DynamoDB is a schemaless database that requires only a table name and a primary key when you create the table.' Under 'Table name', the input field is filled with 'Users'. A note below says: 'This will be used to identify your table.' The second section is 'Partition key'. It contains a note: '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 consistency.' The input field is filled with 'Username', and the type is set to 'String'. A note below says: '1 to 255 characters and case sensitive.' The third section is 'Sort key - optional'. It contains a note: '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.' The input field is filled with 'Enter the sort key name', and the type is set to 'String'. A note below says: '1 to 255 characters and case sensitive.'

Table class	DynamoDB Standard	Yes
Capacity mode	Provisioned	Yes
Provisioned read capacity	5 RCU	Yes
Provisioned write capacity	5 WCU	Yes
Auto scaling	On	Yes
Local secondary indexes	-	No
Global secondary indexes	-	Yes
Encryption key management	Owned by Amazon DynamoDB	Yes
Deletion protection	Off	Yes
Resource-based policy	Not active	Yes

Tags

Tags are pairs of keys and optional values, that you can assign to AWS resources. You can use tags to control access to your resources or track your AWS spending.

No tags are associated with the resource.

[Add new tag](#)

You can add 50 more tags.

[Cancel](#) [Create table](#)

The Users table was created successfully.

[DynamoDB](#) > [Tables](#)

Tables (1) Info

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

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

aws | Search [Alt+S]

DynamoDB > Tables > Create table

Create table

Table details Info

DynamoDB is a schemaless database that requires only a table name and a primary key when you create the table.

Table name
This will be used to identify your table.

Between 3 and 255 characters, containing only letters, numbers, underscores (_), hyphens (-), and periods (.)

Partition key
The partition key is part of the table's primary key. It is a hash value that is used to retrieve items from your table and allocate data across hosts for scalability.
 String
1 to 255 characters and case sensitive.

Sort key - optional
You can use a sort key as the second part of a table's primary key. The sort key allows you to sort or search among all items sharing the same partition key.
 String
1 to 255 characters and case sensitive.

Table class	DynamoDB Standard	Yes
Capacity mode	Provisioned	Yes
Provisioned read capacity	5 RCU	Yes
Provisioned write capacity	5 WCU	Yes
Auto scaling	On	Yes
Local secondary indexes	-	No
Global secondary indexes	-	Yes
Encryption key management	Owned by Amazon DynamoDB	Yes
Deletion protection	Off	Yes
Resource-based policy	Not active	Yes

Tags
Tags are pairs of keys and optional values, that you can assign to AWS resources. You can use tags to control access to your resources or track your AWS spending.

No tags are associated with the resource.
[Add new tag](#)
You can add 50 more tags.

[Cancel](#) [Create table](#)

Tables (2) Info

Any tag key ▾ Any tag value ▾

<input type="checkbox"/>	Name	Status	Partition key	Sort key	Indexes	Replication Regions	Deletion protection
<input type="checkbox"/>	Orders	Active	order_id (\$)	-	0	0	<input checked="" type="radio"/> Off
<input type="checkbox"/>	Users	Active	username (\$)	-	0	0	<input checked="" type="radio"/> Off

Milestone 4 : IAM Role Setup

Create IAM Role.

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

The screenshot shows the AWS Services Catalog interface. At the top left is a navigation bar with icons for Services, Home, and Help. A search bar contains the text 'Q. Iam'. To the right of the search bar is a close button ('X'). Below the search bar, the text 'Search results for "Iam"' is displayed. On the left side, there is a sidebar with links: Services (highlighted in orange), Features, Resources (marked as New), Documentation, Knowledge articles, Marketplace, Blog posts, Events, and Tutorials. The main content area is titled 'Services' and shows four service cards: 'IAM' (Manage access to AWS resources), 'IAM Identity Center' (Manage workforce user access to multiple AWS accounts and cloud applications), 'Resource Access Manager' (Share AWS resources with other accounts or AWS Organizations), and 'AWS App Mesh' (Easily monitor and control microservices). Each card includes a star icon indicating popularity.

Identity and Access Management (IAM) > Roles

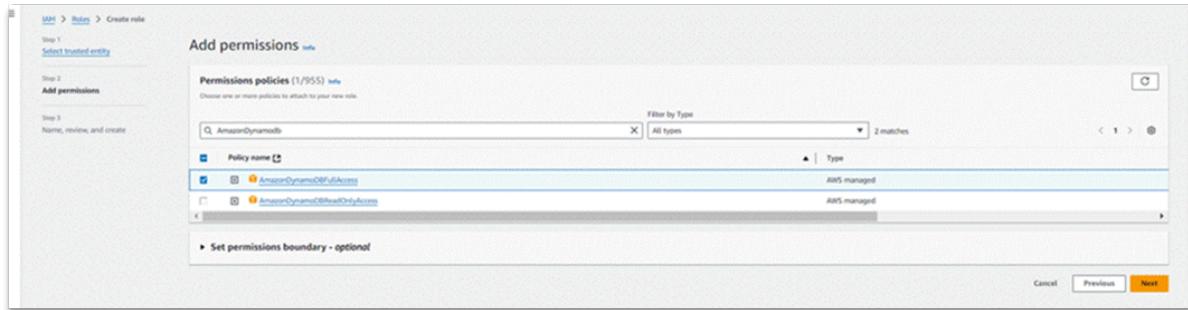
Roles (6) Info

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

Search

Role name	Trusted entities	Last activity
Administrator role	Amazon S3	1 hour ago
Administrator role	Amazon S3	1 hour ago
Administrator role	Amazon S3	1 hour ago
Administrator role	Amazon S3	1 hour ago
Administrator role	Amazon S3	1 hour ago
Administrator role	Amazon S3	1 hour ago

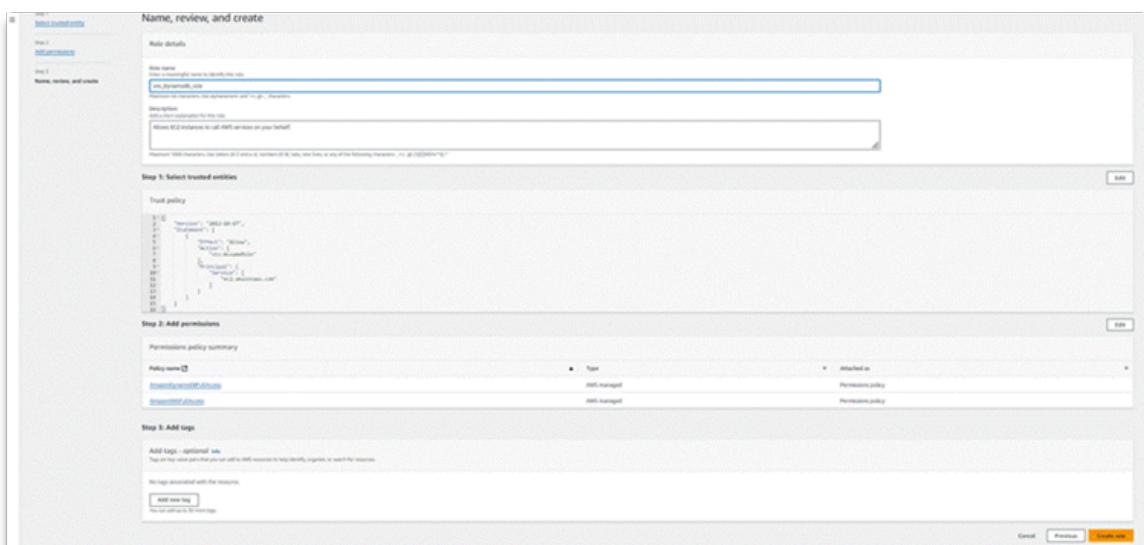
C Delete Create role



Attach Policies

Attach the following policies to the role:

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



EC2_DynamoDB_Role

Allows EC2 instances to call AWS services on your behalf.

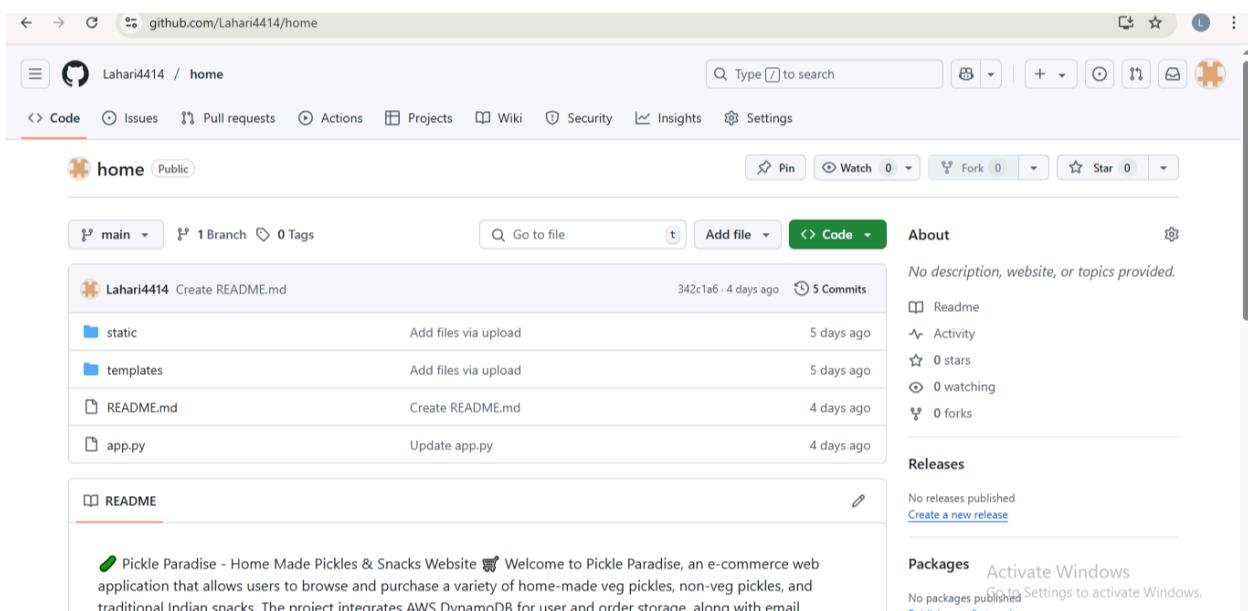
Summary	ARN	Instance profile ARN									
<p>Creation date July 03, 2025, 11:27 (UTC+05:30)</p> <p>Last activity -</p>	arn:aws:iam::084829400932:role/EC2_DynamoDB_Role	arn:aws:iam::084829400932:instance-profile/EC2_DynamoDB_Role									
<p>Permissions</p> <p>Permissions policies (2)</p> <p>You can attach up to 10 managed policies.</p> <table border="1"> <thead> <tr> <th>Policy name</th> <th>Type</th> <th>Attached entities</th> </tr> </thead> <tbody> <tr> <td>AmazonDynamoDBFullAccess</td> <td>AWS managed</td> <td>3</td> </tr> <tr> <td>AmazonDynamoDBAccess</td> <td>AWS managed</td> <td>3</td> </tr> </tbody> </table> <p>Permissions boundary (not set)</p> <p>Access denied to access-analyzer.ListPolicyGenerations</p> <p>You don't have permission to access-analyzer.ListPolicyGenerations. To request access, copy the following text and send it to your AWS administrator. Learn more about troubleshooting access denied errors.</p> <pre>User arn:aws:sts:084829400932:assumed-role/iamaccount-new-%7fr6z798510936953d566 Action: access-analyzer.ListPolicyGenerations On resource: arn:aws:access-analyzer:us-east-1:084829400932:*</pre>			Policy name	Type	Attached entities	AmazonDynamoDBFullAccess	AWS managed	3	AmazonDynamoDBAccess	AWS managed	3
Policy name	Type	Attached entities									
AmazonDynamoDBFullAccess	AWS managed	3									
AmazonDynamoDBAccess	AWS managed	3									

Milestone 5 : EC2 Instance Setup

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

Load your Project Files to GitHub

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

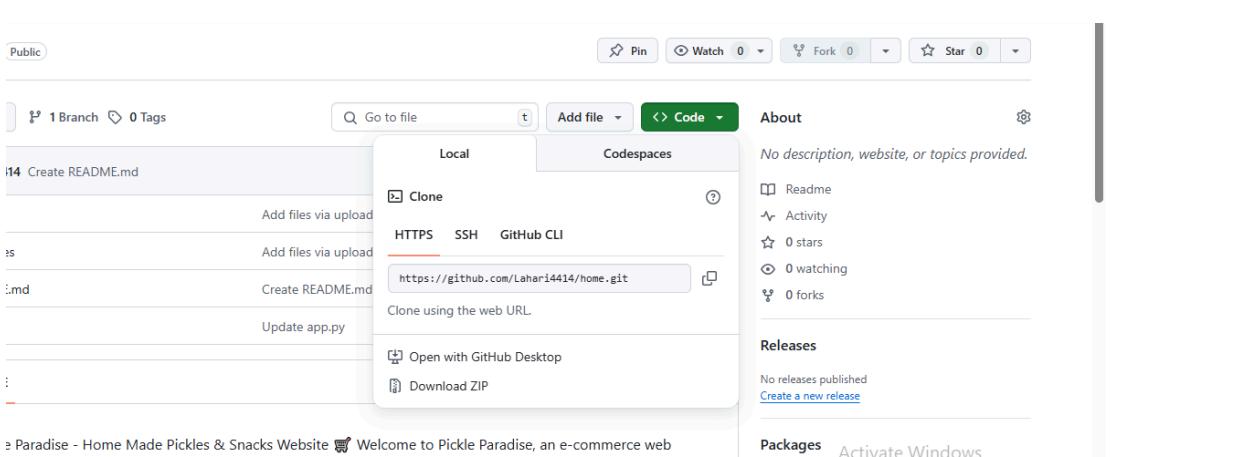


The screenshot shows a GitHub repository page for a user named Lahari4414. The repository is titled 'home' and is marked as 'Public'. The commit history shows five commits, all made by Lahari4414. The commits are:

- Create README.md (342c1a6, 4 days ago)
- Add files via upload (5 days ago)
- Add files via upload (5 days ago)
- Create README.md (4 days ago)
- Update app.py (4 days ago)

The README file contains the following text:

Pickle Paradise - Home Made Pickles & Snacks Website. Welcome to Pickle Paradise, an e-commerce web application that allows users to browse and purchase a variety of home-made veg pickles, non-veg pickles, and traditional Indian snacks. The project integrates AWS DynamoDB for user and order storage, along with email



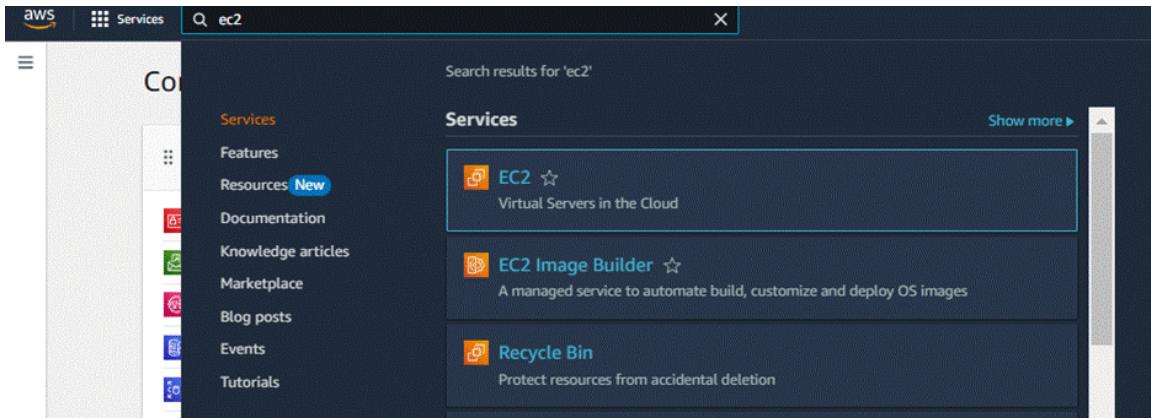
The screenshot shows the same GitHub repository page for Lahari4414/home. A modal window is open over the repository details, specifically the 'Code' section. The modal provides cloning options:

- Local
- Codespaces
- Clone
- HTTPS
- SSH
- GitHub CLI

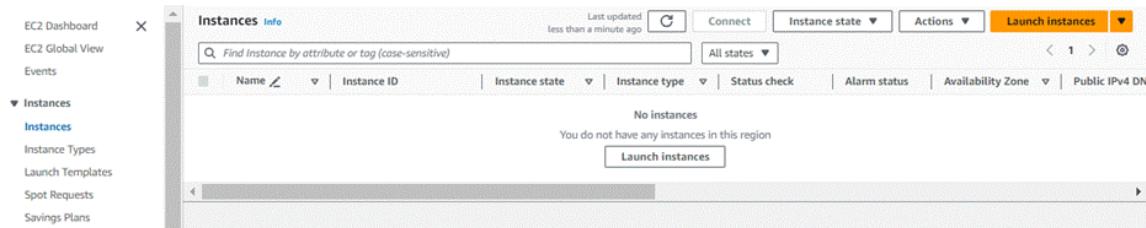
The 'Clone' option is highlighted, and the URL <https://github.com/Lahari4414/home.git> is displayed. Below the cloning options, there are links to 'Open with GitHub Desktop' and 'Download ZIP'.

On the right side of the page, there are sections for 'About', 'Releases', and 'Packages'. The 'About' section notes 'No description, website, or topics provided.' The 'Releases' section says 'No releases published' and has a link to 'Create a new release'. The 'Packages' section says 'Activate Windows' and 'No packages published' with a link to 'Publish your first package'.

Launch an EC2 instance to host the Flask.



● Click on Launch instance to launch EC2 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](#)

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

Search our full catalog including 1000s of application and OS images

Recents

Quick Start



Browse more AMIs

Including AMIs from

▼ Sum

Number of

1

Software I

Amazon Li
ami-002f6eef

Virtual ser

t2.micro

Firewall (s

New securi

Storage (v

1 volume(s)

Free

acc
t2.m
+2 m

Cancel

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

Amazon Machine Image (AMI)

Amazon Linux 2023 AMI
ami-02b49a24cfb95941c (64-bit (x86), uefi-preferred) / ami-04ad8c7fcc828fad4 (64-bit (Arm), uefi)
Virtualization: hvm ENA enabled: true Root device type: ebs

Free tier eligible

Description

Amazon Linux 2023 is a modern, general purpose Linux-based OS that comes with 5 years of long term support. It is optimized for AWS and designed to provide a secure, stable and high-performance execution environment to develop and run your cloud applications.

Architecture: 64-bit (x86) Boot mode: uefi-preferred AMI ID: ami-02b49a24cfb95941c Verified provider

● Create and download the key pair for Server access.

▼ Instance type [Info](#) | [Get advice](#)

Instance type

t2.micro	Free tier eligible
Family: t2	1 vCPU 1 GiB Memory Current generation: true
On-Demand Linux base pricing: 0.0124 USD per Hour	
On-Demand Windows base pricing: 0.017 USD per Hour	
On-Demand RHEL base pricing: 0.0268 USD per Hour	
On-Demand SUSE base pricing: 0.0124 USD per Hour	

Additional costs apply for AMIs with pre-installed software

▼ Key pair (login) [Info](#)

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

Key pair name - required

Select [Create new key pair](#)

Description

Amazon Linux 2023 is a modern, general purpose Linux-based OS that comes with 5 years of long term support. It is optimized for AWS and designed to provide a secure, stable and high-performance execution environment to develop and run your cloud applications.

Architecture	Boot mode	AMI ID	Username
64-bit (x86)	uefi-preferred	ami-078264b8ba71b	ec2-user
Verified provider			
c4se			

▼ Instance type [Info](#) | [Get advice](#)

Instance type

t2.micro	Free tier eligible
Family: t2	1 vCPU 1 GiB Memory Current generation: true
On-Demand Linux base pricing: 0.0124 USD per Hour	
On-Demand Windows base pricing: 0.017 USD per Hour	
On-Demand RHEL base pricing: 0.0268 USD per Hour	
On-Demand SUSE base pricing: 0.0124 USD per Hour	

Additional costs apply for AMIs with pre-installed software

▼ Key pair (login) [Info](#)

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

Key pair name - required

InstantLibrary [Create new key pair](#)

▼ Summary

Number of instances [Info](#)

1

Software Image (AMI)

Amazon Linux 2023 AMI 2023.5.2... [read more](#)
ami-078264b8ba71b
c4se

Virtual server type (instance type)

t2.micro

Firewall (security group)

New security group

Storage (volumes)

1 volume(s) - 8 GiB

① Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free tier AMIs per month, 750 hours of public IPv4 address usage per month, 30 GiB of EBS storage, 2 million IOPS, 1 GB of snapshots, and 100 GiB of bandwidth to the internet.

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

Configure security groups for HTTP, and SSH access.

▼ Network settings [Info](#)

VPC - required [Info](#)
 (default) [▼](#) [Remove](#)

Subnet [Info](#)
 [▼](#) [Create new subnet](#) [▼](#)

Auto-assign public IP [Info](#)
 [▼](#)

Additional charges apply when outside of free tier allowance

Firewall (security groups) [Info](#)
A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

Create security group [▼](#) Select existing security group [▼](#)

Security group name - required

This security group will be added to all network interfaces. The name can't be edited after the security group is created. Max length is 255 characters. Valid characters: a-z, A-Z, 0-9, spaces, and _-:/()#,@+=&;!\$^*

Description - required [Info](#)

Inbound Security Group Rules

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

Type Info <input type="text" value="ssh"/> ▼	Protocol Info <input type="text" value="TCP"/> ▼	Port range Info <input type="text" value="22"/> ▼	Remove
Source type Info <input type="text" value="Anywhere"/> ▼	Source Info <input type="text" value="Add CIDR, prefix list or security"/> ▼	Description - optional Info <input type="text" value="e.g. SSH for admin desktop"/> ▼	<input type="text" value="0.0.0.0/0"/> ▼

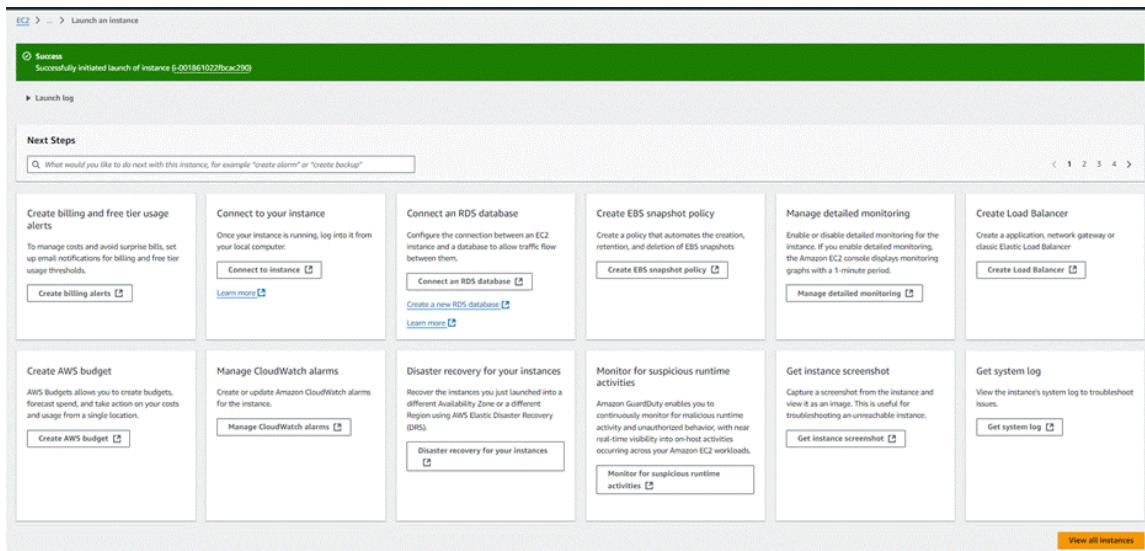
▼ Security group rule 2 (TCP; 80, 0.0.0.0/0)

Type Info <input type="text" value="HTTP"/> ▼	Protocol Info <input type="text" value="TCP"/> ▼	Port range Info <input type="text" value="80"/> ▼	Remove
Source type Info <input type="text" value="Custom"/> ▼	Source Info <input type="text" value="Add CIDR, prefix list or security"/> ▼	Description - optional Info <input type="text" value="e.g. SSH for admin desktop"/> ▼	<input type="text" value="0.0.0.0/0"/> ▼

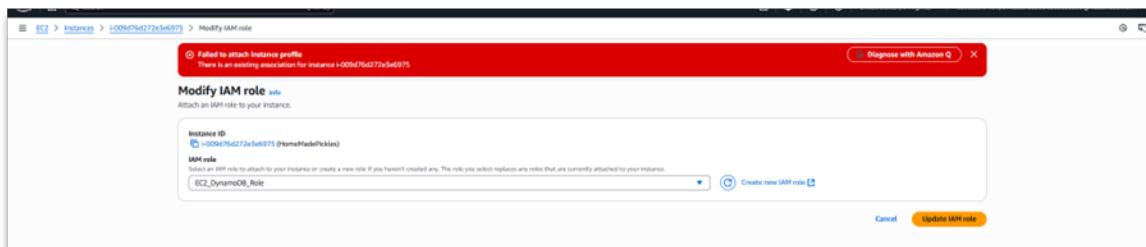
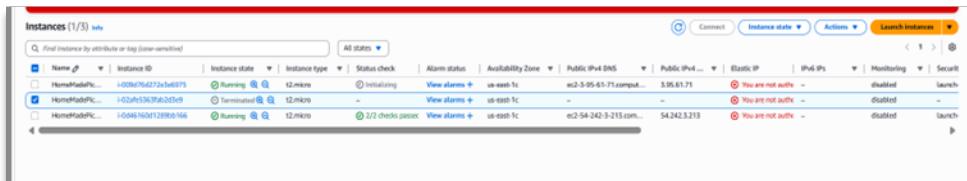
▼ Security group rule 3 (TCP; 5000, 0.0.0.0/0)

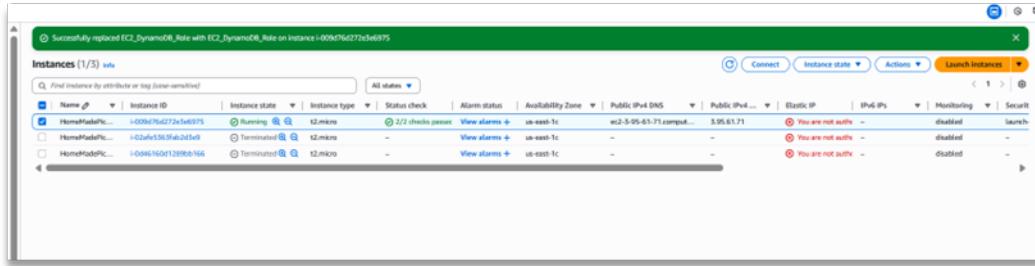
Type Info <input type="text" value="Custom TCP"/> ▼	Protocol Info <input type="text" value="TCP"/> ▼	Port range Info <input type="text" value="5000"/> ▼	Remove
Source type Info <input type="text" value="Custom"/> ▼	Source Info <input type="text" value="Add CIDR, prefix list or security"/> ▼	Description - optional Info <input type="text" value="e.g. SSH for admin desktop"/> ▼	<input type="text" value="0.0.0.0/0"/> ▼

[Add security group rule](#)



To connect to EC2 using EC2 Instance Connect, start by ensuring that an IAM role is attached to your EC2 instance. You can do this by selecting your instance, clicking on Actions, then navigating to Security and selecting Modify IAM Role to attach the appropriate role. After the IAM role is connected, navigate to the EC2 section in the AWS Management Console. Select the EC2 instance you wish to connect to. At the top of the EC2 Dashboard, click the Connect button. From the connection methods presented, choose EC2 Instance Connect. Finally, click Connect again, and a new browser-based terminal will open, allowing you to access your EC2 instance directly from your browser.





Now connect the EC2 with the files

Install Software on the EC2 Instance

Install Python3, Flask, and Git:

On Amazon Linux 2:

```
sudo yum update -y
```

```
sudo yum install python3 git
```

```
sudo pip3 install flask boto3
```

Verify Installations:

```
flask --version
```

```
git --version
```

Clone Your Flask Project from GitHub

Clone your project repository from GitHub into the EC2 instance using Git.

- Run: <https://github.com/Lahari4414/home.git>

- Note: change your-github-username and your-repository-name with your credentials here: <https://github.com/Lahari4414/home.git>
- This will download your project to the EC2 instance.

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

- cd Homemadepicklesandsnacks
- cd "Home Made Pickles1"

Create a Virtual Environment:

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

Once inside the project directory, configure and run the Flask application by executing the following command with elevated privileges:

- Run the Flask Application
- sudo flask run --host=0.0.0.0 --port=5000

```

aws Terminal Search [Alt+5] United States (N. Virginia) nsoaccount-view/67fca6a7985d9c36953d566@nosandboxnew14

[1] 11:39:22 ~ % pip3 install flask boto3
Collecting flask
  Downloading flask-1.1.1-py3-none-any.whl (5.1 kB)
Collecting werkzeug==1.0.2
  Downloading werkzeug-1.0.2-py3-none-any.whl (22.4 kB)
Collecting click==7.0.0
  Downloading click-7.0.0-py3-none-any.whl (6.6 kB)
Collecting itsdangerous==1.1.0
  Downloading itsdangerous-1.1.0-py3-none-any.whl (0.9 kB)
Collecting Jinja2==2.11.3
  Downloading Jinja2-2.11.3-py3-none-any.whl (368 kB)
Collecting MarkupSafe==1.1.1
  Downloading MarkupSafe-1.1.1-py3-none-any.whl (19 kB)
Collecting boto3==1.19.2
  Downloading boto3-1.19.2-py3-none-any.whl (13.8 MB)
Collecting botocore==1.39.2
  Downloading botocore-1.39.2-py3-none-any.whl (13.8 MB)
Collecting s3transfer==0.1.13
  Downloading s3transfer-0.1.13-py3-none-any.whl (85 kB)
Requirement already satisfied: jmespath<2.0.0,>=0.7.1 in /usr/lib/python3.9/site-packages (from boto3) (0.10.0)
Requirement already satisfied: urllib3<1.27.0,>=1.25.0 in /usr/lib/python3.9/site-packages (from botocore<1.40.0,>=1.39.2>botocore) (1.25.10)
Requirement already satisfied: pyyaml<1.0.0,>=3.12 in /usr/lib/python3.9/site-packages (from botocore<1.40.0,>=1.39.2>botocore) (2.8.1)
Requirement already satisfied: s3transfer<0.1.13,>=0.1.13.0 in /usr/lib/python3.9/site-packages (from botocore<1.40.0,>=1.39.2>botocore) (1.15.0)
Installing collected packages: botocore, s3transfer, boto3
Successfully installed botocore-1.39.2 botocore-1.39.2 s3transfer-0.13.0
[ec2-user@ip-172-31-17-101 ~]$ git clone https://github.com/yasminshaik1329/pickle-paradsie.git
Cloning into 'pickle-paradsie'...
remote: Enumerating objects: 73, done.
remote: Counting objects: 100% (73/73), done.
remote: Compressing objects: 100% (70/70), done.
remote: Total 73 (delta 13), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (73/73), 9.75 MiB | 21.24 MiB/s, done.
Resolving deltas: 100% (13/13), done.
[ec2-user@ip-172-31-17-101 ~]$ i-009d76d272e3e6975 (HomeMadePickles)
[PublicIP: 3.95.61.71 PrivateIP: 172.31.17.101]

```

Verify the Flask app is running:

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

- Run the Flask app on the EC2 instance

The screenshot shows a terminal session in AWS CloudShell. The user has run several commands to upload files to an S3 bucket, update a Lambda function code, and then deployed a Flask application to an EC2 instance. The deployment command output includes a warning about using it in production.

```

remote: Counting objects: 100% (51/51), done.
remote: Compressing objects: 100% (51/51), done.
remote: Total 51 (delta 10), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (51/51), 2.11 MiB | 29.97 MiB/s, done.
Resolving deltas: 100% (10/10), done.
[ec2-user@ip-172-31-28-52 ~]$ cd home
[ec2-user@ip-172-31-28-52 home]$ python3 app.py
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5000
 * Running on http://172.31.28.52:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 107-456-553
^C[ec2-user@ip-172-31-28-52 home]$ ^C
[ec2-user@ip-172-31-28-52 home]$ ^C
[ec2-user@ip-172-31-28-52 home]$ 
```

i-08424a3afe159d32 (HomeMadePickles)

Public IPs: 54.236.22.196 Private IPs: 172.31.28.52

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

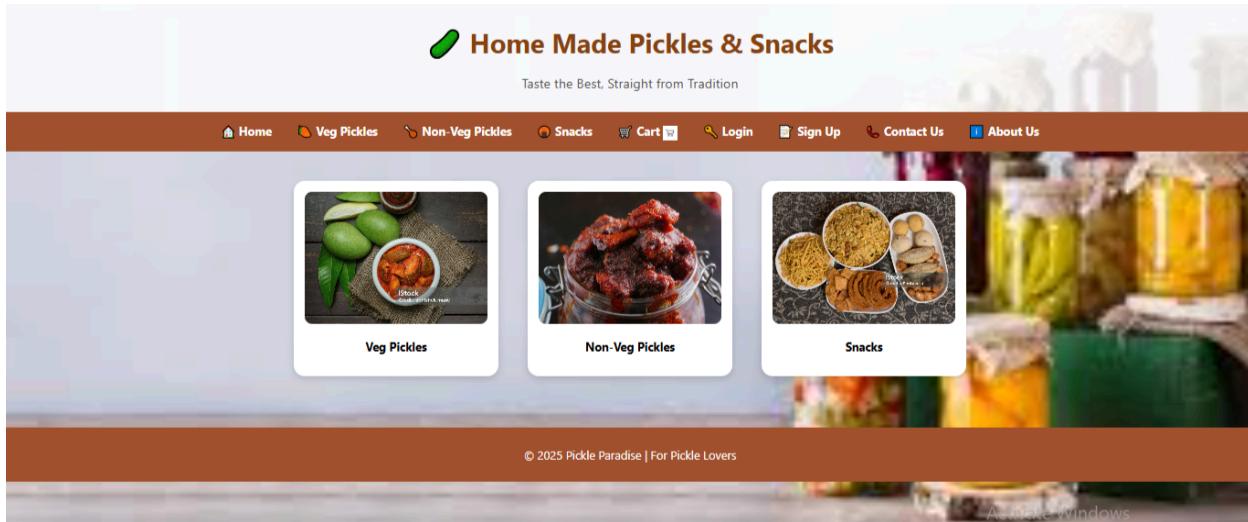
Milestone 7 : Testing and Deployment

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

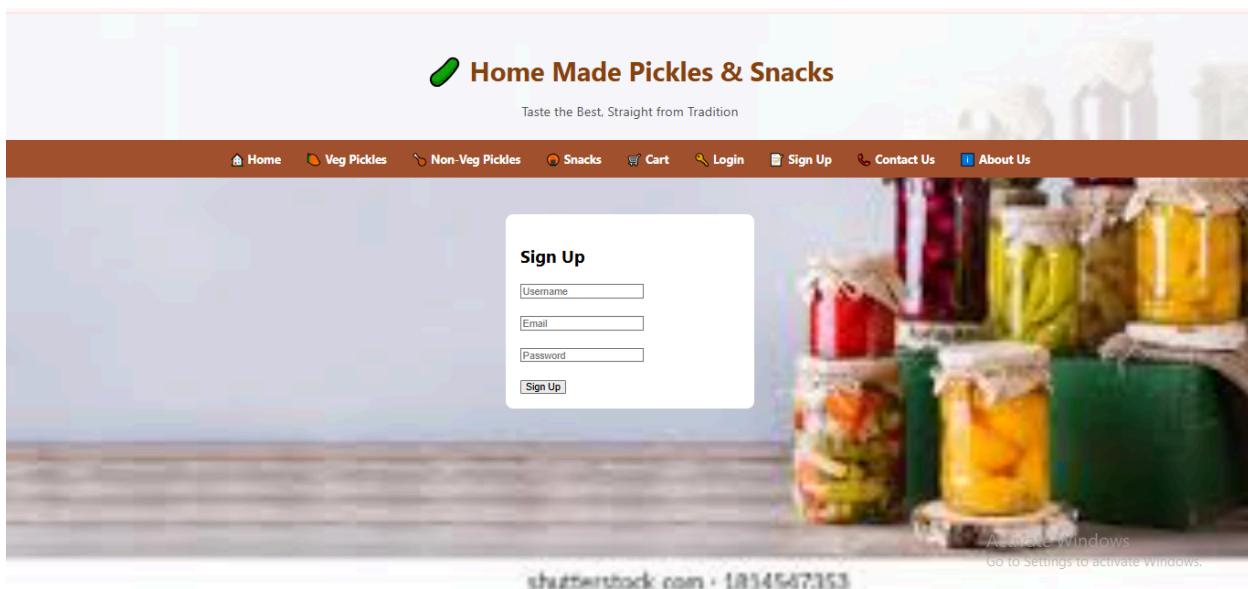
Functional testing to verify the Project

Welcome page:

Home page:



Signup page:



Login page:



Veg pickles:

A screenshot of a website's product listing page for vegetable pickles. The background is the same as the previous image, showing jars of pickles. The header "Home Made Pickles & Snacks" and the tagline "Taste the Best. Straight from Tradition" are visible. A brown navigation bar at the top includes links for Home, Veg Pickles, Non-Veg Pickles, Snacks, Cart, Login, Sign Up, Contact Us, and About Us. Below the navigation, the heading "Our Best Veg Pickles (₹)" is displayed. Four products are shown in cards: "Mango Pickle – ₹150" (with an "Add to Cart" button), "Lemon Pickle – ₹120" (with an "Add to Cart" button), "Tomato Pickle – ₹180" (with an "Add to Cart" button), and "Gongura Pickle – ₹160" (with an "Add to Cart" button). At the bottom of the page, there is a footer with the text "© 2025 Pickle Paradise | For Pickle Lovers" and a watermark that reads "Activate Windows Go to Settings to activate Windows."

Non-Veg pickles:

The screenshot shows the 'Non-Veg Pickles' section of the website. At the top, there's a navigation bar with links for Home, Veg Pickles, Non-Veg Pickles (which is the active page), Snacks, Cart, Login, Sign Up, Contact Us, and About Us. Below the navigation is a header with the text 'Home Made Pickles & Snacks' and a subtitle 'Taste the Best. Straight from Tradition'. A main heading 'Our Best Non-Veg Pickles (₹)' is centered above four product cards. Each card features an image of a pickle, its name, price, and an 'Add to Cart' button.

Product	Price (₹)	Action
Chicken Pickle	300	Add to Cart
Mutton Pickle	350	Add to Cart
Fish Pickle	320	Add to Cart
Prawn Pickle	340	Add to Cart

At the bottom of the page, there's a footer with copyright information and a note about activating Windows.

Snacks:

The screenshot shows the 'Snacks' section of the website. The layout is identical to the Non-Veg Pickles section, with a similar navigation bar and header. The main heading is 'Our Best Snacks (₹)'. It displays four snack items: Snack Combo, Murukku, Mixture, and Chakli, each with an image, price, and an 'Add to Cart' button.

Snack	Price (₹)	Action
Snack Combo	200	Add to Cart
Murukku	100	Add to Cart
Mixture	120	Add to Cart
Chakli	130	Add to Cart

The footer contains copyright and Windows activation information.

Cart Page:

Your Cart

Product	Price (₹)	Action
Murukku	100	Remove
Mutton Pickle	350	Remove

Total: ₹450

[Proceed to Checkout](#)

© 2025 Pickle Paradise | For Pickle Lovers

Activate Windows
Go to Settings to activate Windows.

Checkout Page:

Home Made Pickles & Snacks

Taste the Best. Straight from Tradition

[Home](#) [Veg Pickles](#) [Non-Veg Pickles](#) [Snacks](#) [Cart](#) [Login](#) [Sign Up](#) [Contact Us](#) [About Us](#)

Checkout

Please confirm your order and provide your details below:

Name:

Address:

Phone Number:

[Place Order](#)

© 2025 Pickle Paradise | For Pickle Lovers

Activate Windows
Go to Settings to activate Windows.

Conclusion

The **Homemade Pickles: Taste the Best** project demonstrates the successful development and deployment of a scalable, cloud-based e-commerce platform using modern web technologies and AWS services. By leveraging **Flask** for backend development, **AWS EC2** for hosting, **DynamoDB** for secure data management, and **SNS** for real-time notifications, the system provides a seamless and responsive experience for both customers and administrators.

This project not only solves the limitations of traditional small-scale selling but also opens up a professional and automated channel for reaching wider audiences. Customers can easily browse, select, and order high-quality homemade products with real-time updates and secure transactions, while sellers can manage orders more efficiently.

The architecture ensures reliability, scalability, and performance – making it future-ready as customer demand and product variety increase. Through this project, the effective integration of cloud services and Python web development is clearly showcased, reflecting practical implementation of theoretical knowledge and real-world problem-solving skills.

In conclusion, this solution provides a **modern, reliable, and efficient e-commerce experience**, supporting both digital transformation for small businesses and convenience for customers.

