

E-Commerce site for HomeMade Pickles & Snacks: Taste the Best

Project Description:

Home Made Pickles & Snacks — Taste the Best is a cloud-based culinary platform revolutionizing access to authentic, handcrafted pickles and snacks. Addressing the growing demand for preservative-free, traditional recipes, this initiative combines artisanal craftsmanship with cutting-edge technology to deliver farm-fresh flavors directly to consumers. Built on Flask for backend efficiency and hosted on AWS EC2 for scalable performance, the platform offers seamless browsing, ordering, and subscription management. DynamoDB ensures real-time inventory tracking and personalized user experiences, while fostering sustainability through partnerships with local farmers and eco-friendly packaging. From tangy regional pickles to wholesome snacks, every product celebrates heritage recipes, nutritional integrity, and convenience—proving that tradition and innovation can coexist deliciously. "Preserving Traditions, One Jar at a Time."

Scenarios:

Scenario 1: Scalable Order Management for High Demand

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

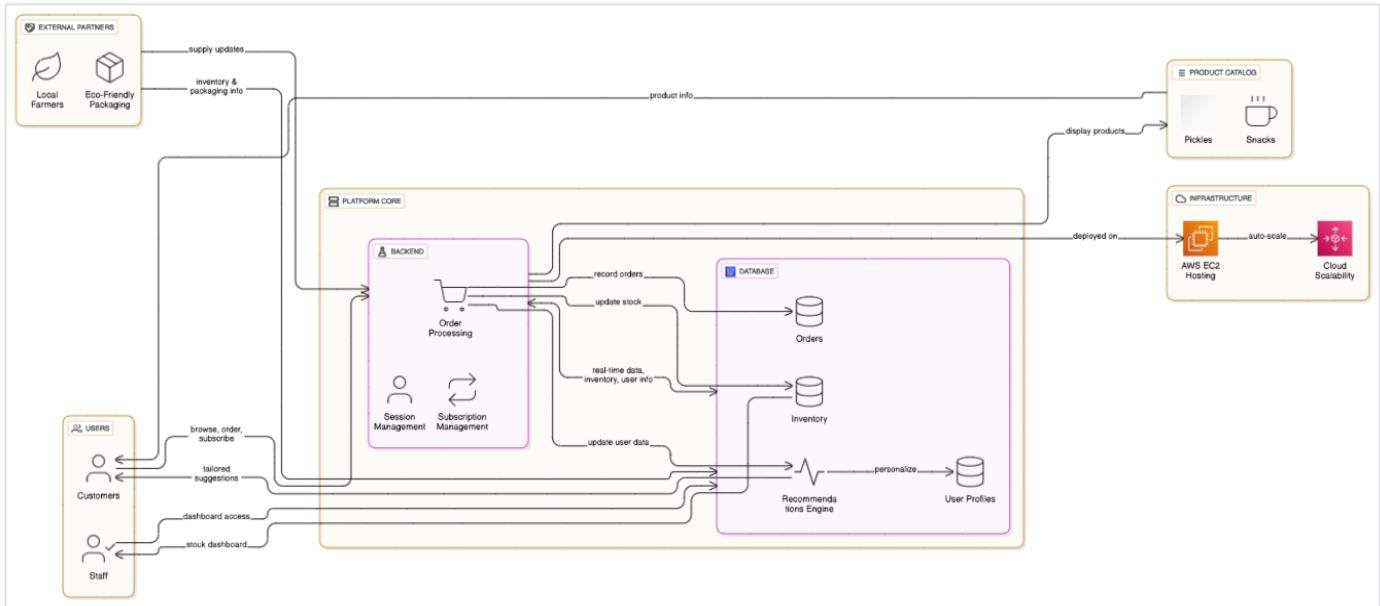
Scenario 2: Real-Time Inventory Tracking and Updates

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

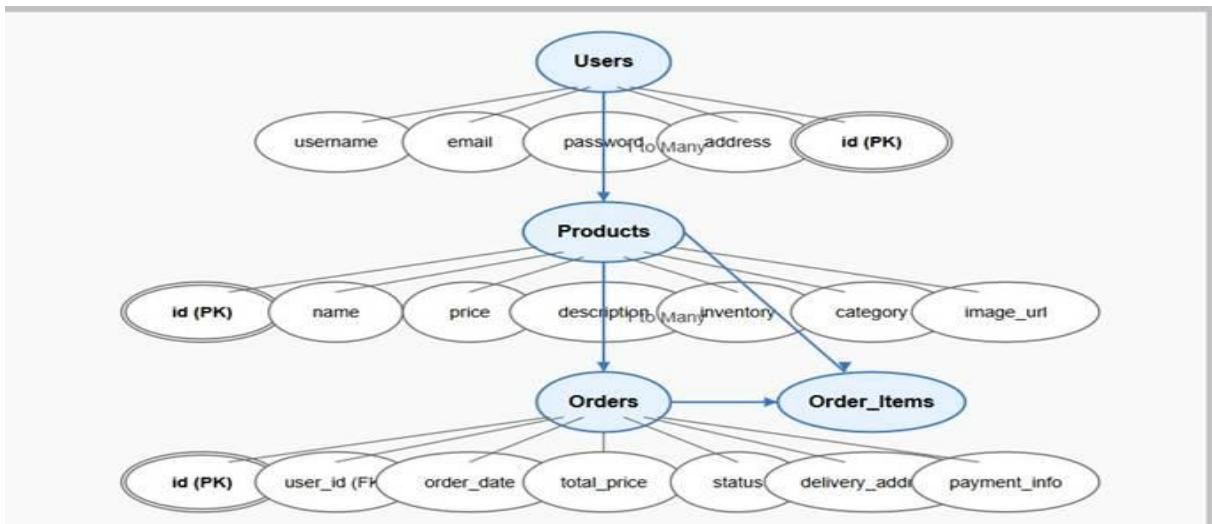
Scenario 3: Personalized User Experience and Recommendations

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

AWS ARCHITECTURE



Entity Relationship (ER) Diagram:



Pre-requisites:

- AWS Account Setup:** <https://docs.aws.amazon.com/accounts/latest/reference/getting-started.html>
- Understanding IAM:** <https://docs.aws.amazon.com/IAM/latest/UserGuide/introduction.html>
- Amazon EC2 Basics:** <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts.html>

4. **DynamoDB Basics:**

<https://docs.aws.amazon.com/amazondynamodb/Introduction.html>

5. **Git Documentation:** <https://git-scm.com/doc>

6. **VS Code Installation:** <https://code.visualstudio.com/download>

Project WorkFlow :

Activity 1: Backend Development and Application Setup

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

Activity 2: AWS Account Setup and Login

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

Activity 3: DynamoDB Database Creation and Setup

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

Activity 4: SNS Notification Setup

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

Activity 5: IAM Role Setup

- **Create IAM Role**
- **Attach Policies**

Activity 6: EC2 Instance Setup

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

Activity 7: Deployment on EC2

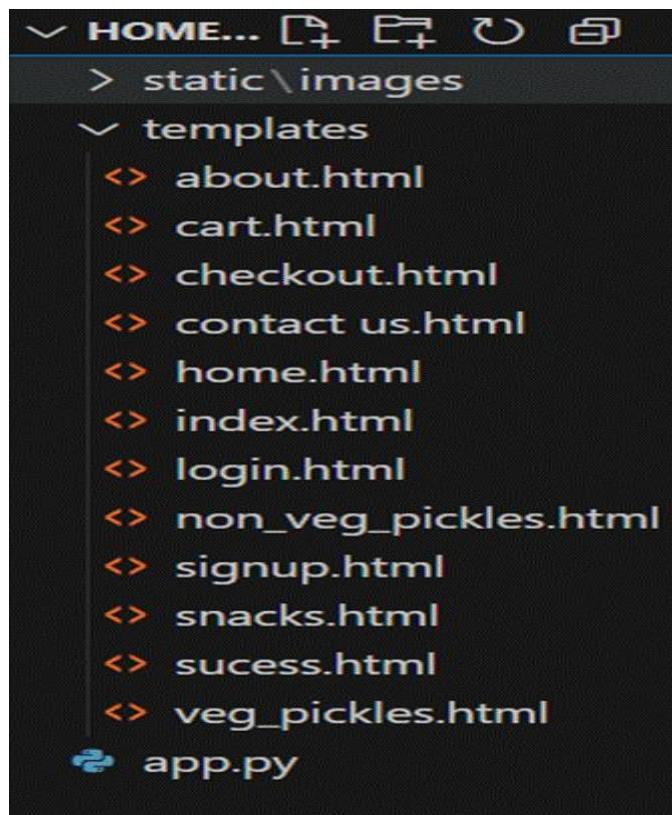
- Upload Flask Files
- Run the Flask App

Activity 8: Testing and Deployment

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

Milestone 1 : Web Application Development and Setup

- Activity 1.1: Develop the Backend code Using Flask.
 - File Explorer Structure



Description of the code :

Flask App Initialization:

```
from flask import Flask, render_template, request, redirect, url_for, session
from werkzeug.security import generate_password_hash, check_password_hash
import boto3
from datetime import datetime
import json,uuid
```

```
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.

```
dynamodb = boto3.resource('dynamodb', region_name='ap-south-1') # e.g., 'us-east-1'
users_table = dynamodb.Table('Users')
orders_table = dynamodb.Table(['Orders'])
```

```
products = {
    'non_veg_pickles': [
        {'id': 1, 'name': 'Chicken Pickle', 'weights': {'250': 600, '500': 1200, '1000': 1800}},
        {'id': 2, 'name': 'Fish Pickle', 'weights': {'250': 200, '500': 400, '1000': 800}},
        {'id': 3, 'name': 'Gongura Mutton', 'weights': {'250': 400, '500': 800, '1000': 1600}},
        {'id': 4, 'name': 'Mutton Pickle', 'weights': {'250': 400, '500': 800, '1000': 1600}},
        {'id': 5, 'name': 'Gongura Prawns', 'weights': {'250': 600, '500': 1200, '1000': 1800}},
        {'id': 6, 'name': 'Chicken Pickle (Gongura)', 'weights': {'250': 350, '500': 700, '1000': 1050}}
    ],
    'veg_pickles': [
        {'id': 7, 'name': 'Traditional Mango Pickle', 'weights': {'250': 150, '500': 280, '1000': 500}},
        {'id': 8, 'name': 'Zesty Lemon Pickle', 'weights': {'250': 120, '500': 220, '1000': 400}},
        {'id': 9, 'name': 'Tomato Pickle', 'weights': {'250': 130, '500': 240, '1000': 450}},
        {'id': 10, 'name': 'Kakarakaya Pickle', 'weights': {'250': 130, '500': 240, '1000': 450}},
        {'id': 11, 'name': 'Chintakaya Pickle', 'weights': {'250': 130, '500': 240, '1000': 450}},
        {'id': 12, 'name': 'Spicy Pandu Mirchi', 'weights': {'250': 130, '500': 240, '1000': 450}}
    ],
    # Add your veg pickle products here
    'snacks': [
        {'id': 7, 'name': 'Banana Chips', 'weights': {'250': 300, '500': 600, '1000': 800}},
        {'id': 8, 'name': 'Crispy Aam-Papad', 'weights': {'250': 150, '500': 300, '1000': 600}},
        {'id': 9, 'name': 'Crispy Chekka Pakodi', 'weights': {'250': 50, '500': 100, '1000': 200}},
        {'id': 10, 'name': 'Boondhi Acchu', 'weights': {'250': 300, '500': 600, '1000': 900}},
        {'id': 11, 'name': 'Chekkalu', 'weights': {'250': 350, '500': 700, '1000': 1000}},
        {'id': 12, 'name': 'Ragi Laddu', 'weights': {'250': 350, '500': 700, '1000': 1000}},
        {'id': 13, 'name': 'Dry Fruit Laddu', 'weights': {'250': 500, '500': 1000, '1000': 1500}},
        {'id': 14, 'name': 'Kara Boondi', 'weights': {'250': 250, '500': 500, '1000': 750}},
        {'id': 15, 'name': 'Gavvalu', 'weights': {'250': 250, '500': 500, '1000': 750}},
        {'id': 16, 'name': 'Kaju Chikki', 'weights': {'250': 250, '500': 500, '1000': 750}}
    ]
}
```

- Routes for Web Pages
- Login Route (GET/POST):Verifies user credentials, increments login count, and redirects to the dashboard on success.

```

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

@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']

    try:
        # Fetch user from DynamoDB
        response = users_table.get_item(Key={'username': username})

        if 'Item' not in response:
            return render_template('login.html', error='User not found')

        user = response['Item']

        # Ensure password field exists in the DB
        if 'password' not in user:
            return render_template('login.html', error='Password not found in database')

        # Verify password
        if check_password_hash(user['password'], password):
            session['logged_in'] = True
            session['username'] = username
            session.setdefault('cart', []) # Initialize cart if not set
            return redirect(url_for('index'))
    
```

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

```

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

    try:
        # Check if username exists
        response = users_table.get_item(Key={'username': username})
        if 'Item' in response:
            return render_template('signup.html', error='Username already exists')

        # Hash password before storing
        hashed_password = generate_password_hash(password)

        # Store new user in DynamoDB
        users_table.put_item(
            Item={
                'username': username,
                'email': email,
                'password': hashed_password # Store hashed password
            }
        )

        return redirect(url_for('login'))

    except Exception as e:
        app.logger.error(f"Signup error: {str(e)}")

```

```

        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')

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

```

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

```

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

```

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

```

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

@app.route('/non_veg_pickles')
def non_veg_pickles():
    if not session.get('logged_in'):
        return redirect(url_for('login'))

    return render_template('non_veg_pickles.html', products=products['non_veg_pickles'])

@app.route('/veg_pickles')
def veg_pickles():
    if not session.get('logged_in'):
        return redirect(url_for('login'))

    # Simply pass all products without filtering
    return render_template('veg_pickles.html', products=products['veg_pickles'])

@app.route('/snacks')
def snacks():
    if not session.get('logged_in'):
        return redirect(url_for('login'))

    return render_template('snacks.html', products=products['snacks'])

```

Check out Route:

```

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

    error_message = None # Variable to hold error messages

    if request.method == 'POST':
        try:
            # Extract form data safely
            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()

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

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

            # Get cart data from hidden inputs
            cart_data = request.form.get('cart_data', '[]')
            total_amount = request.form.get('total_amount', '0')

```

(i) Restart Visual Studio Code to apply the late

```

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

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

        # Store order in DynamoDB
        try:
            orders_table.put_item(
                Item={
                    'order_id': str(uuid.uuid4()),
                    'username': session.get('username', 'Guest'),
                    'name': name,
                    'address': address,
                    'phone': phone,
                    'items': cart_items,
                    'total_amount': total_amount,
                    'payment_method': payment_method,
                    'timestamp': datetime.now().isoformat()
                }
            )
        except Exception as db_error:
            print(f'DynamoDB Error: {db_error}')
            return render_template('checkout.html', error="Failed to save order. Please try again later.")

```

```
# Redirect to success page with success message
return redirect(url_for('sucess', message="Your order has been placed successfully!"))

except Exception as e:
    print(f"Checkout error: {str(e)}")
    return render_template('checkout.html', error="An unexpected error occurred. Please try again.")

return render_template('checkout.html') # Render checkout page for GET request

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

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

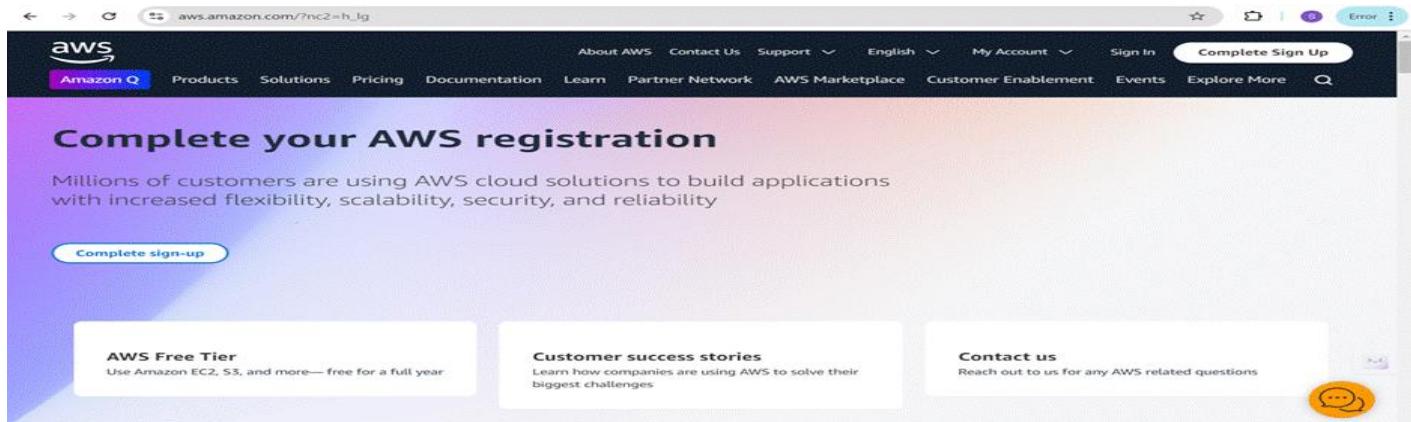
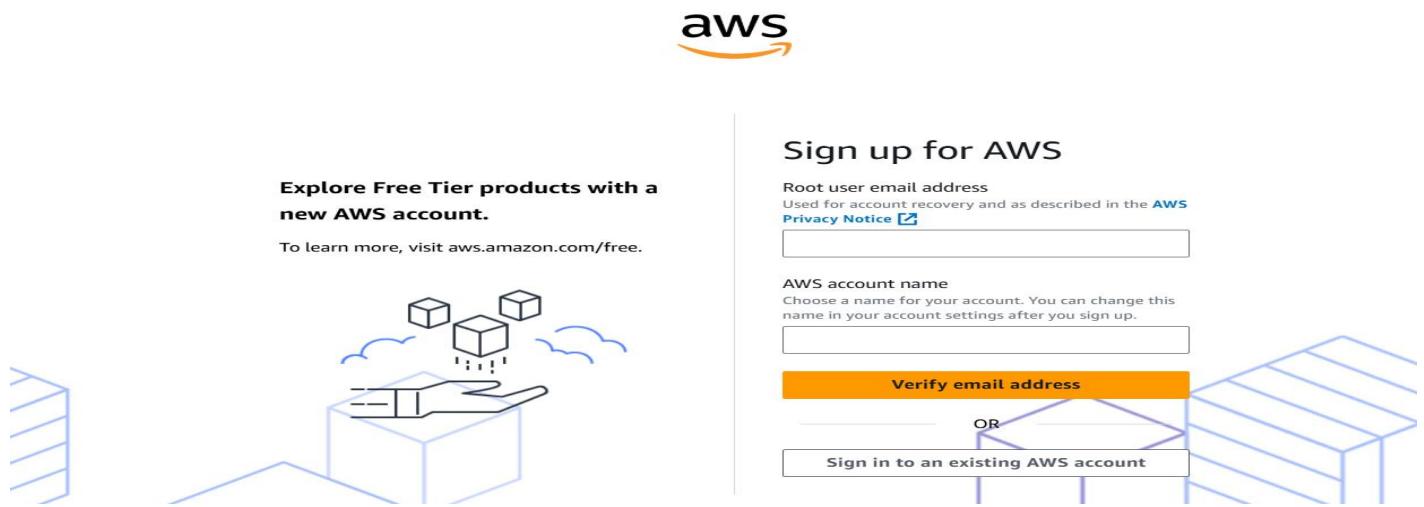
AWS - Local Deployment Sample Code -

<https://docs.google.com/document/d/1sFF7tJ6IgWtRbawWoA4W3PkxEFrSJZhKzULgLsxo/edit?usp=sharing>

Milestone 2: AWS Account Setup and Login

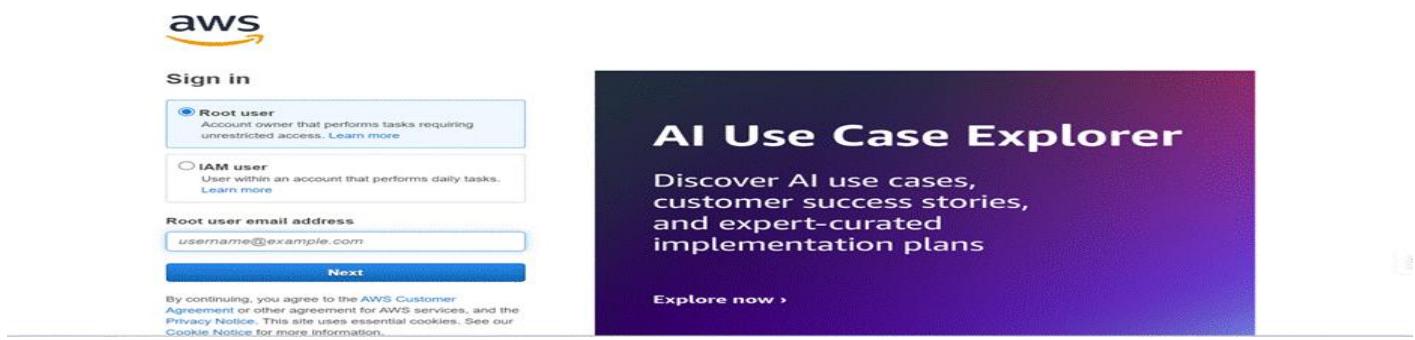
- Go to the AWS website (<https://aws.amazon.com/>).
- Click on the "Create an AWS Account" button.
- Follow the prompts to enter your email address and choose a password.
- Provide the required account information, including your name, address, and phone number.
- Enter your payment information. (Note: While AWS offers a free tier, a credit card or debit card is required for verification.)
- Complete the identity verification process.

- Choose a support plan (the basic plan is free and sufficient for starting).
- Once verified, you can sign in to your new AWS accounts.

The diagram illustrates the user journey. It starts with the 'Complete your AWS registration' page, which leads to the 'Sign up for AWS' step. This step includes fields for 'Root user email address' (with a note about account recovery and privacy) and 'AWS account name' (with a note about changing it later). There are two options: 'Verify email address' (highlighted in orange) and 'Sign in to an existing AWS account'. Arrows point from the registration page to the sign-up page, and from the sign-up page to both verification and sign-in options. The background features a large blue wireframe cube and a hand holding a stack of three smaller cubes.

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

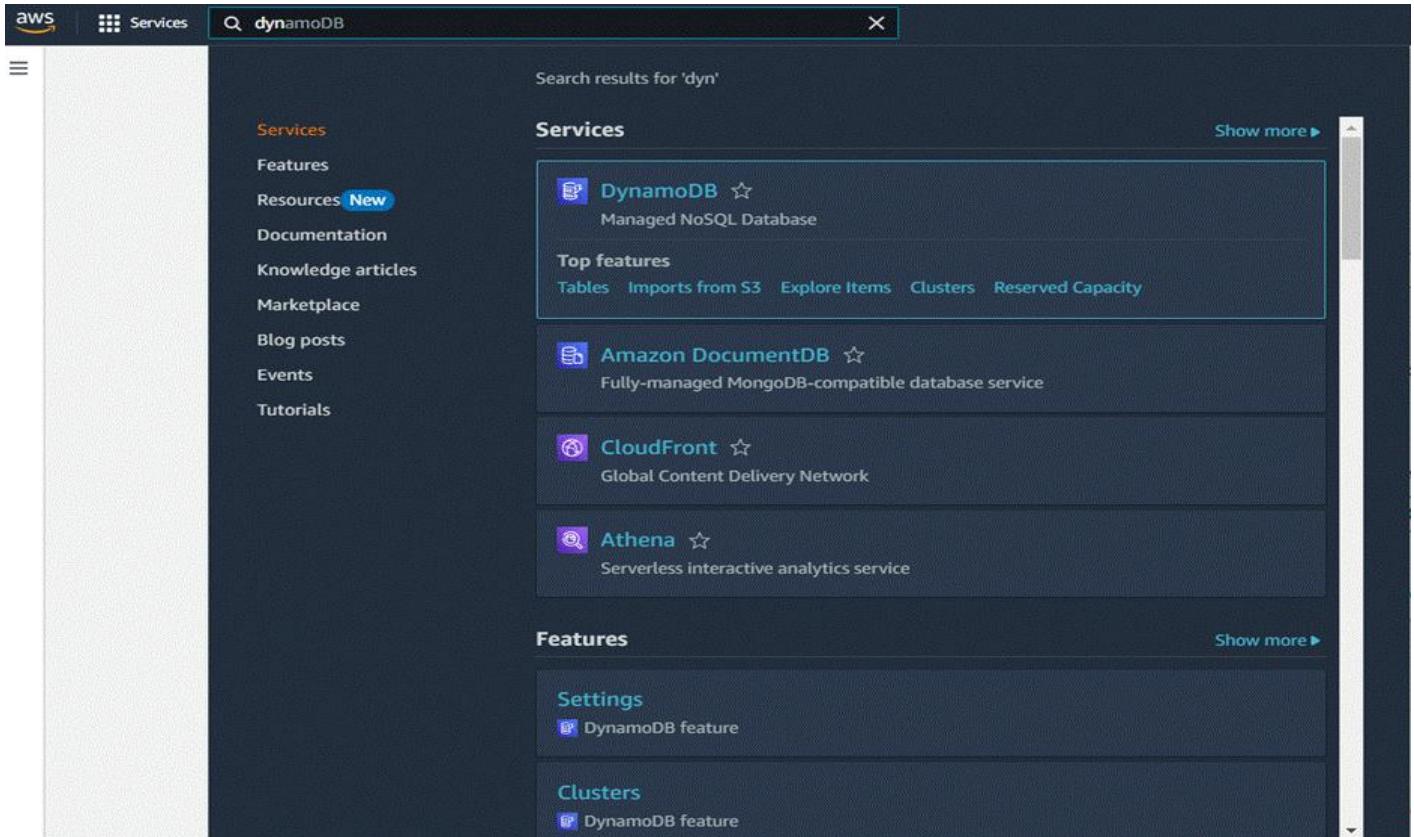


The screenshot shows the AWS Sign In interface. It asks the user to choose between 'Root user' (selected) and 'IAM user'. It also asks for a 'Root user email address' (username@example.com) and a 'Next' button. Below the form is a small legal notice. To the right is a dark purple banner for the 'AI Use Case Explorer' with the text: 'Discover AI use cases, customer success stories, and expert-curated implementation plans. Explore now >'. The banner has a small yellow speech bubble icon in the bottom right corner.

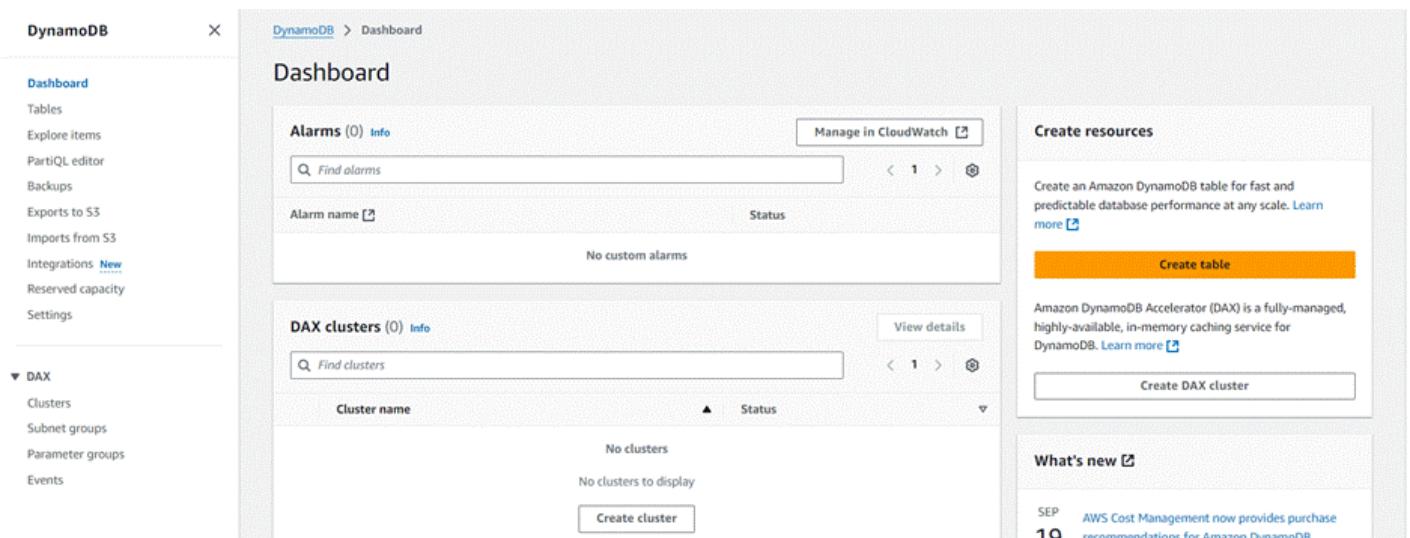
Milestone 3 : DynamoDB Database Creation and Setup

- **Navigate to the DynamoDB**

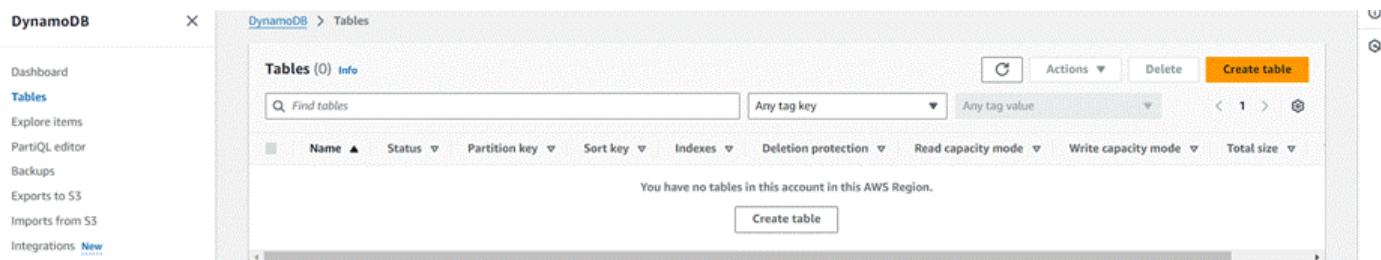
- In the AWS Console, navigate to DynamoDB and click on create tables.



The screenshot shows the AWS Services search results page. The search bar at the top contains 'dynamoDB'. The results are filtered under the 'Services' category. The first result is 'DynamoDB' (Managed NoSQL Database), which is highlighted with a blue border. Below it are other services: 'Amazon DocumentDB' (Fully-managed MongoDB-compatible database service), 'CloudFront' (Global Content Delivery Network), and 'Athena' (Serverless interactive analytics service). Further down, under 'Features', there is a section for 'Settings' (DynamoDB feature) and 'Clusters' (DynamoDB feature). On the left sidebar, there are links for 'Features', 'Resources New', 'Documentation', 'Knowledge articles', 'Marketplace', 'Blog posts', 'Events', and 'Tutorials'.

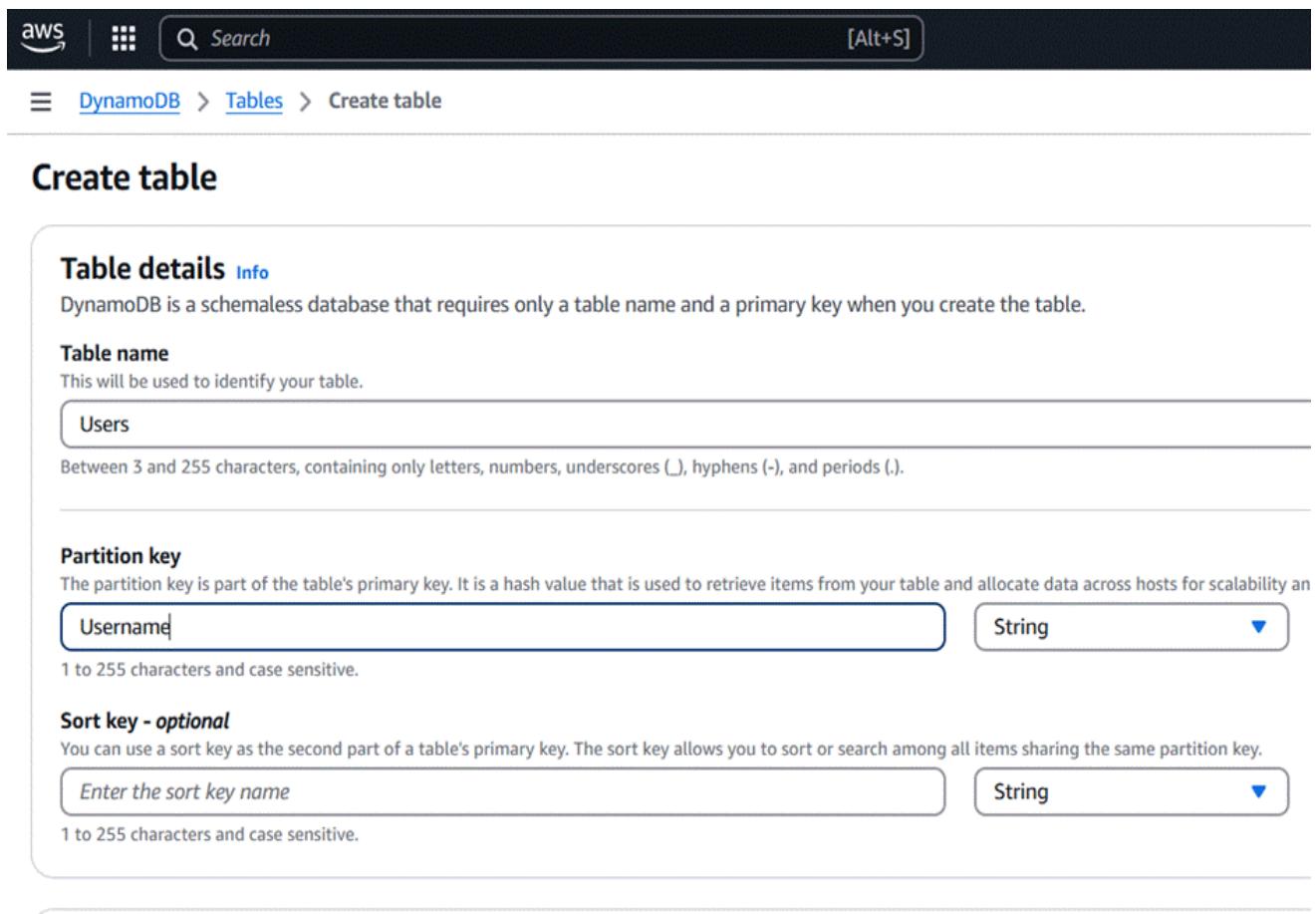


The screenshot shows the DynamoDB Dashboard. The left sidebar has a 'Dashboard' tab selected, along with other options like 'Tables', 'Explore items', 'PartiQL editor', 'Backups', 'Exports to S3', 'Imports from S3', 'Integrations New', 'Reserved capacity', and 'Settings'. A 'DAX' section is also present with 'Clusters', 'Subnet groups', 'Parameter groups', and 'Events'. The main dashboard area has two sections: 'Alarms (0)' and 'DAX clusters (0)'. Both sections have a 'Find' bar, a table with columns for 'Cluster name' and 'Status', and a 'Create' button ('Create table' for alarms, 'Create DAX cluster' for clusters). To the right, there is a 'Create resources' section for creating a new DynamoDB table, a 'What's new' section with a note about AWS Cost Management, and a 'What's new' section for Amazon DynamoDB Accelerator (DAX).



The screenshot shows the AWS DynamoDB 'Tables' page. The left sidebar has links for Dashboard, Tables (which is selected), Explore items, PartiQL editor, Backups, Exports to S3, Imports from S3, and Integrations. The main area title is 'Tables (0) Info'. It includes search and filter fields for 'Find tables', 'Any tag key', 'Any tag value', and sorting by 'Name', 'Status', 'Partition key', 'Sort key', 'Indexes', 'Deletion protection', 'Read capacity mode', 'Write capacity mode', and 'Total size'. A message says 'You have no tables in this account in this AWS Region.' A prominent orange 'Create table' button is at the bottom.

- **Create a DynamoDB table for storing registration details and book requests.**
 - Create Users table with partition key “Username” with type String and click on create tables.



The screenshot shows the 'Create table' wizard in the AWS DynamoDB console. The top navigation bar includes the AWS logo, a search bar, and a keyboard shortcut [Alt+S]. Below it, the breadcrumb navigation shows 'DynamoDB > Tables > Create table'. The main section is titled '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 and consistency.

1 to 255 characters and case sensitive.

Sort key - optional

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

1 to 255 characters and case sensitive.

Table 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.

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 a requests table with Email as the primary key for book requests data.

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.
 Type: 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.
 Type: 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](#)

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

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.

Services X

Search results for 'Iam'

Services

- Features
- Resources **New**
- Documentation
- Knowledge articles
- Marketplace
- Blog posts
- Events
- Tutorials

Show more ▶

 **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

 **AWS App Mesh** ☆ Easily monitor and control microservices

Identity and Access Management (IAM) X

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
...
...
...
...
...

Delete Create role

Step 1 Select trusted entity Info

Trusted entity type

- AWS service

Allow AWS services like S3, Lambda, or others to perform actions in this account.
- AWS account

Allow entities in other AWS accounts belonging to you or a 3rd party to perform actions in this account.
- IAM user federated with SAML 2.0 from a separate directory to perform actions in this account.
- Custom trust policy

Create a custom trust policy to enable others to perform actions in this account.

Use case
 Select an IAM role for the EC2 Lambda, or others to perform actions in this account.

Service or use case

Choose a use case for the specified service.

Use case
 EC2

Allow EC2 instances to call AWS services on your behalf.

- EC2 Role for AWS Systems Manager

Allow EC2 instances to call AWS services like CloudWatch and Systems Manager on your behalf.
- EC2 Spot Fleet Role

Allow EC2 Spot Fleet to request and terminate Spot Instances on your behalf.
- EC2 Spot Fleet Auto Scaling

Allow EC2 Spot Fleet Auto Scaling and update EC2 spot fleets on your behalf.
- EC2 Spot Fleet Trigger

Allow EC2 to search spot instances and attach tags to the launched instances on your behalf.
- EC2 Spot Instances

Allow EC2 Spot Instances to launch and manage spot instances on your behalf.
- EC2 Scheduled Instances

Allow EC2 Scheduled Instances to manage instances on your behalf.

Cancel Next

Step 1 Select trusted entity Info

Step 2 Add permissions Info

Permissions policies (1/955) Info

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

Filter by Type: All types 2 matches

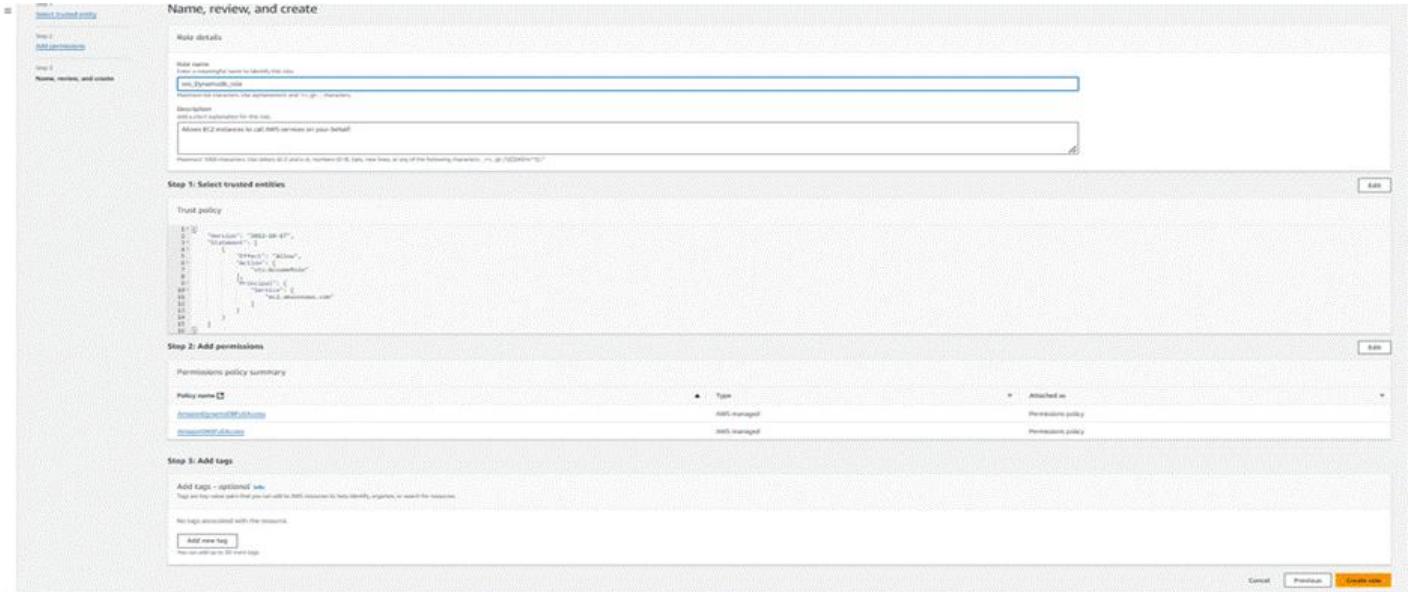
Policy name	Type
<input checked="" type="checkbox"/> AmazonDynamoDBFullAccess	AWS managed
<input type="checkbox"/> AmazonDynamoDBReadOnlyAccess	AWS managed

Set permissions boundary - optional

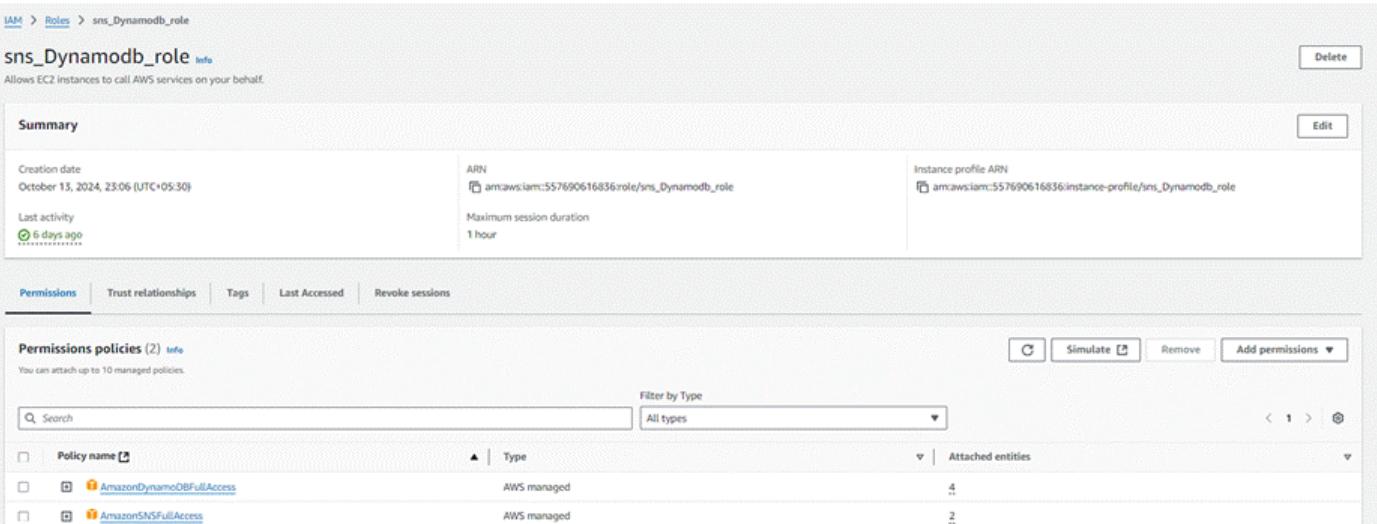
Cancel Previous Next

○ Attach Policies

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



The screenshot shows the "Name, review, and create" step of the IAM role creation wizard. It includes three main sections: Step 1: Select trusted entities, Step 2: Add permissions, and Step 3: Add tags. In Step 1, the trust policy is set to "AWS Lambda". In Step 2, the policy name is "sns_Dynamodb_role" and it is attached to the role. In Step 3, there are no tags added.



The screenshot shows the "sns_Dynamodb_role" role details page. It includes a "Summary" section with creation date (October 13, 2024), last activity (6 days ago), ARN (arn:aws:iam::557690616836:role/sns_Dynamodb_role), and instance profile ARN (arn:aws:iam::557690616836:instance-profile/sns_Dynamodb_role). Below this is a "Permissions" tab showing two managed policies: "AmazonDynamoDBFullAccess" and "AmazonSNSFullAccess".

Milestone 5: EC2 Instance Set Up

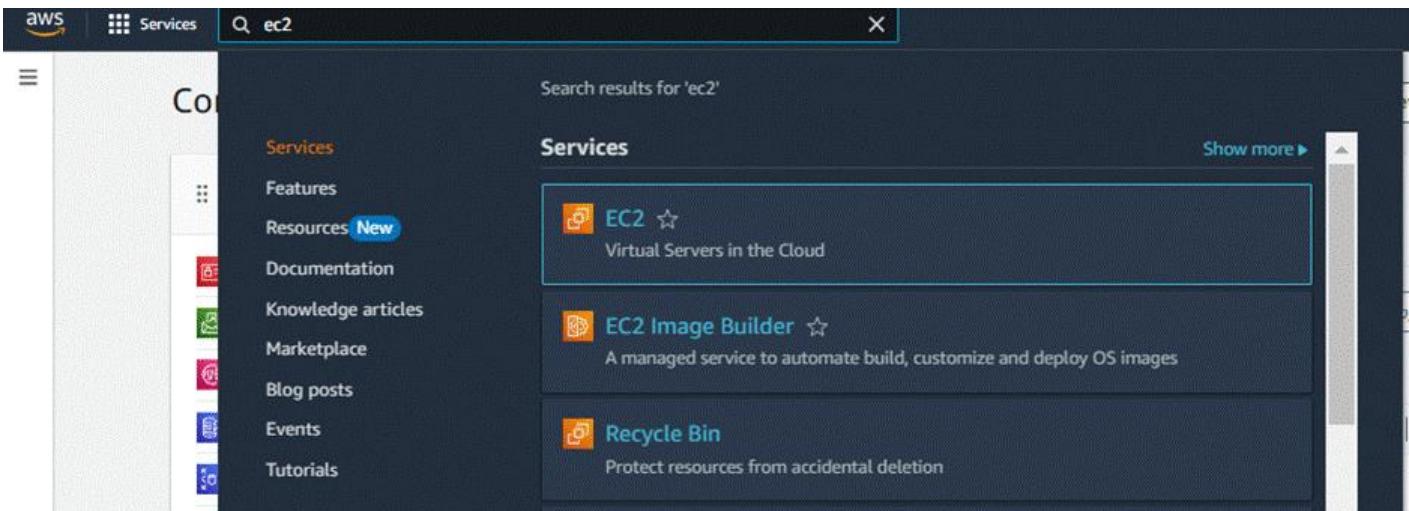
Load your Flask app and Html files into GitHub repository.

 static	Initial commit
 templates	Update statistics.html
 app.py	Update app.py

- Launch an EC2 instance to host the Flask

- Launch EC2 Instance.

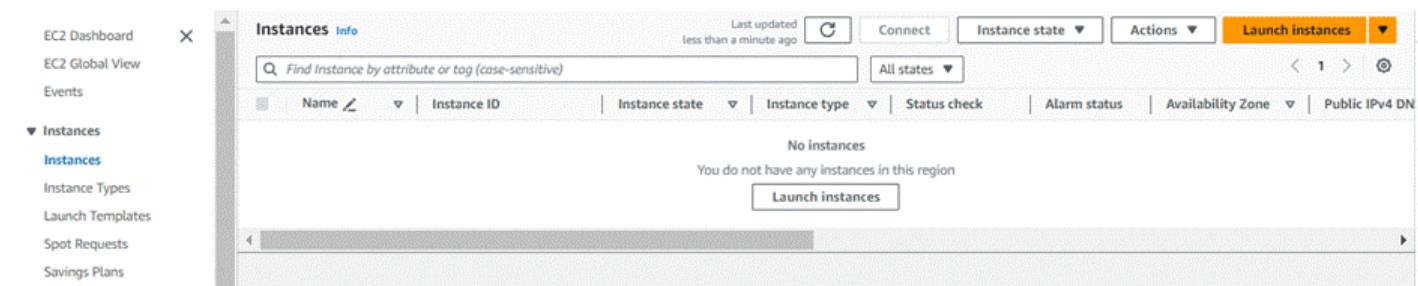
In the AWS Console, navigate to EC2 and launch a new instance.



The screenshot shows the AWS CloudSearch interface with a search query of 'ec2'. The results include:

- EC2** Virtual Servers in the Cloud
- EC2 Image Builder** A managed service to automate build, customize and deploy OS images
- Recycle Bin** Protect resources from accidental deletion

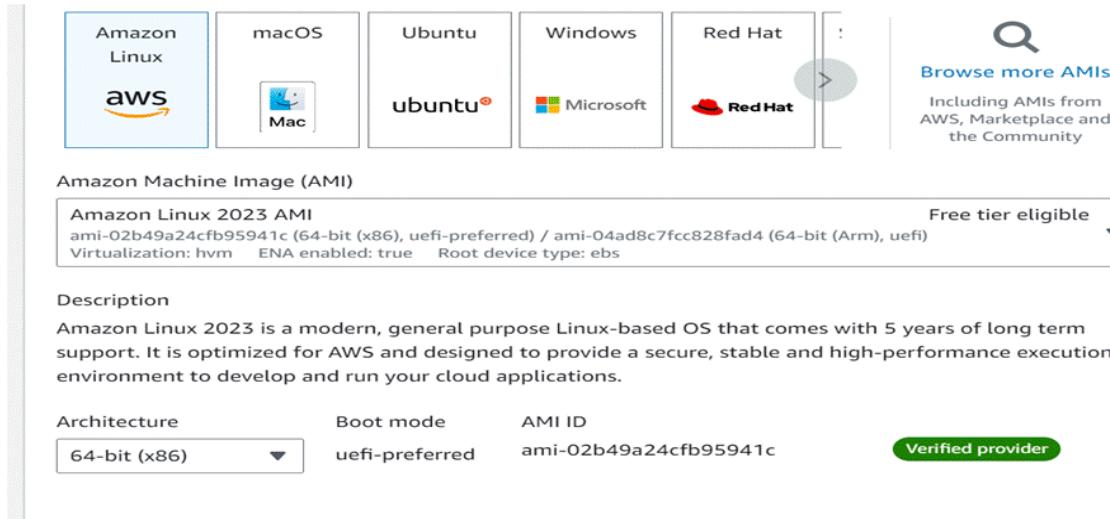
Click on Launch instance to launch EC2 instance



The screenshot shows the AWS EC2 Instances page. The left sidebar includes links for EC2 Dashboard, EC2 Global View, Events, Instances (selected), Instances, Instance Types, Launch Templates, Spot Requests, and Savings Plans. The main content area displays the following information:

- Instances Info**: Shows a search bar, a 'Launch instances' button, and a message: "You do not have any instances in this region".
- Launch an instance**: A blue banner with the text: "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".
- Name and tags**: A section where the user can enter a name (e.g., "HomeMadePickles") and add additional tags.
- Application and OS Images (Amazon Machine Image)**: A section where users can search for AMIs. Recent AMIs listed include Amazon Linux, macOS, Ubuntu, Windows, Red Hat, SUSE Linux, and Debian.
- Summary**: On the right side, it shows 1 item under Software (Amazon Linux), 1 virtual server (t2.micro), 1 firewall (New security group), and 1 volume (1 volume(s)).

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

Free tier eligible

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

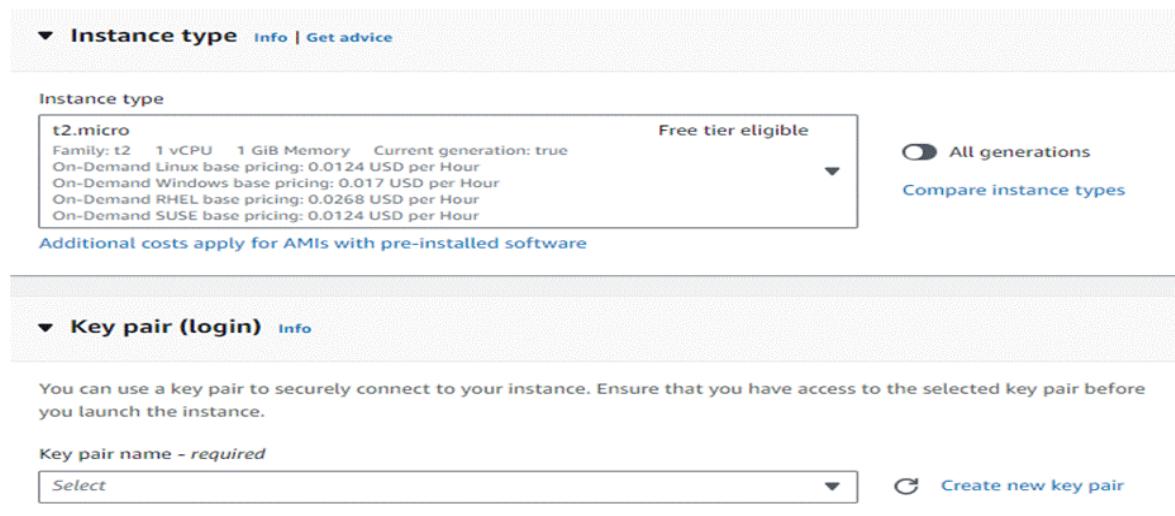
Description

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
64-bit (x86)	uefi-preferred	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

All generations

Compare instance types

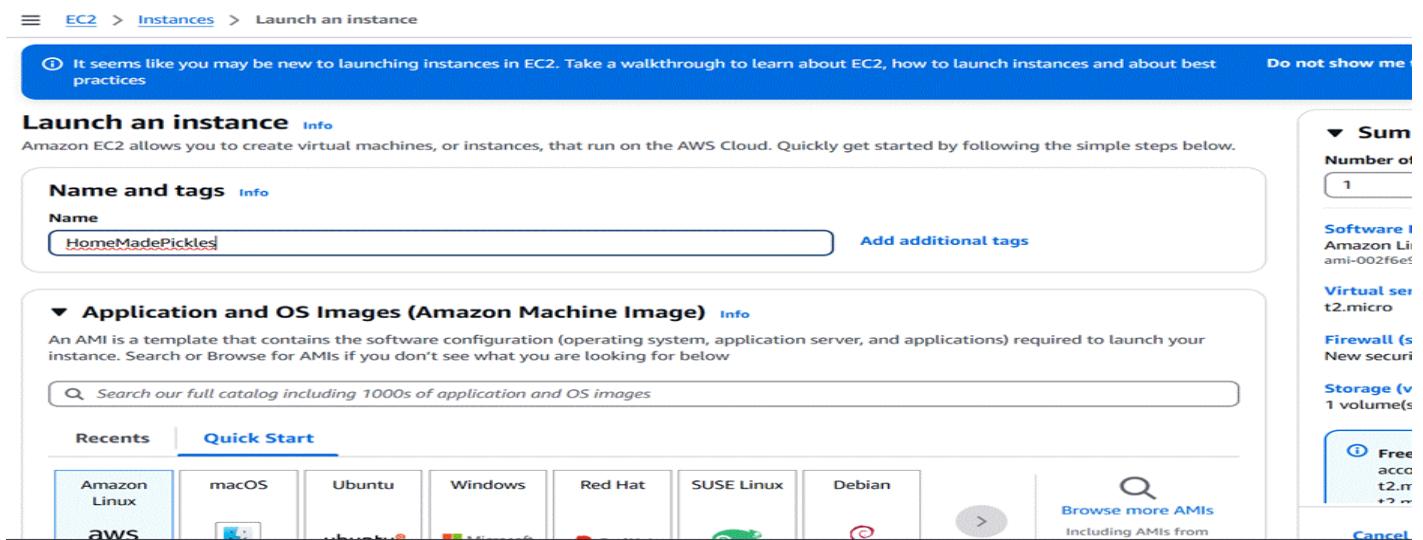
Additional costs apply for AMIs with pre-installed software

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



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

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

Amazon Linux aws macOS Ubuntu Windows Red Hat SUSE Linux Debian

Browse more AMIs Including AMIs from

Sum
 Number of 1

Software I
 Amazon Li ami-002f6ef

Virtual ser
 t2.micro

Firewall (s
 New securi

Storage (v
 1 volume(s)

Free acc
 t2.m +2 ..

Cancel

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-078264b8ba71bc45e	ec2-user	Verified provider

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

Instance type	Free tier eligible
t2.micro	 All generations
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 

Summary

Number of instances | [Info](#)
1

Software Image (AMI)
Amazon Linux 2023 AMI 2023.5.2...[read more](#)
ami-078264b8ba71bc45e

Virtual server type (instance type)
t2.micro

Firewall (security group)
New security group

Storage (volumes)
1 volume(s) - 8 GiB

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 I/Os, 1 GB of snapshots, and 100 GB of bandwidth to the internet. 

Launch instance

- Configure security groups for HTTP, and SSH access.

Network settings [Info](#)

VPC - required [Info](#)

vpc-03cdc7b6f19dd7211 (default) 

Subnet [Info](#)

No preference 

Auto-assign public IP [Info](#)

Enable 

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

launch-wizard

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](#)

launch-wizard created 2024-10-13T17:49:56.622Z

Inbound Security Group Rules

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

Type Info	Protocol Info	Port range Info
ssh	TCP	22
Source type Info	Source Info	Description - optional Info
Anywhere	Add CIDR, prefix list or security	e.g. SSH for admin desktop
0.0.0.0/0 X		
- ▼ Security group rule 2 (TCP, 80, 0.0.0.0/0)

Type Info	Protocol Info	Port range Info
HTTP	TCP	80
Source type Info	Source Info	Description - optional Info
Custom	Add CIDR, prefix list or security	e.g. SSH for admin desktop
0.0.0.0/0 X		
- ▼ Security group rule 3 (TCP, 5000, 0.0.0.0/0)

Type Info	Protocol Info	Port range Info
Custom TCP	TCP	5000
Source type Info	Source Info	Description - optional Info
Custom	Add CIDR, prefix list or security	e.g. SSH for admin desktop
0.0.0.0/0 X		

[Add security group rule](#)

EC2 > ... > Launch an instance

 Success Successfully initiated launch of instance i-001861022fbca:290

▶ Launch log

Next Steps

Q. What would you like to do next with this instance, for example "create alarm" or "create backup"?

< 1 2 3 4 >

Creating billing and free tier usage alerts To manage costs and avoid surprise bills, set up email notifications for billing and free tier usage thresholds. Create billing alerts	Connect to your instance Once your instance is running, log into it from your local computer. Connect to instance Learn more	Connect an RDS database Configure the connection between an EC2 instance and a database to allow traffic flow between them. Connect an RDS database Create a new RDS database Learn more	Create EBS snapshot policy Create a policy that automates the creation, retention, and deletion of EBS snapshots. Create EBS snapshot policy	Manage detailed monitoring Enable or disable detailed monitoring for the instance. If you enable detailed monitoring, the Amazon EC2 console displays monitoring graphs with a 1-minute period. Manage detailed monitoring	Create Load Balancer Create a application, network gateway or classic Elastic Load Balancer. Create Load Balancer
Create AWS budget AWS Budgets allows you to create budgets, forecast spend, and take action on your costs and usage from a single location. Create AWS budget	Manage CloudWatch alarms Create or update Amazon CloudWatch alarms for the instance. Manage CloudWatch alarms	Disaster recovery for your instances Recover EC2 instances you just launched into a different Availability Zone or a different Region using AWS Elastic Disaster Recovery (EDR). Disaster recovery for your instances	Monitor for suspicious runtime activities Amazon GuardDuty enables you to continuously monitor for malicious runtime activity and unauthorized behavior, with near real-time visibility into on-host activities occurring across your Amazon EC2 workloads. Monitor for suspicious runtime activities	Get instance screenshot Capture a screenshot from the instance and view it as an image. This is useful for troubleshooting an unresponsive instance. Get Instance screenshot	Get system log View the instance's system log to troubleshoot issues. Get system log

[View all instances](#)

- 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.

Instances (1/2) Info											Last updated less than a minute ago	Connect	Instance state ▾	Actions ▾	Launch instances ▾
Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP	IPv6 IPs	Monitoring	Security			
InstantLibrary...	i-001861022fbcac290	Stopped	t2.micro	-	View alarms +	ap-south-1b	-	-	-	-	disabled	launch-wi			

EC2 > Instances > i-001861022fbcac290

Instance summary for i-001861022fbcac290 (InstantLibraryApp) [Info](#)

Updated less than a minute ago

Instance ID i-001861022fbcac290	Public IPv4 address -	Private IPv4 addresses 172.31.3.5
IPv6 address -	Instance state Stopped	Public IPv4 DNS -
Hostname type IP name: ip-172-31-3-5.ap-south-1.compute.internal	Private IP DNS name (IPv4 only) ip-172-31-3-5.ap-south-1.compute.internal	Elastic IP addresses -
Answer private resource DNS name IPv4 (A)	Instance type t2.micro	AWS Compute Optimizer finding Opt-in to AWS Compute Optimizer for recommendations. Learn more
Auto-assigned IP address -	VPC ID vpc-03cdc7b6f19dd7211	Auto Scaling Group name -
IAM Role sns_Dynamodb_role	Subnet ID subnet-0d9fa3144480cc9a9	
IMDSv2 Required	Instance ARN arn:aws:ec2:ap-south-1:1557690616836:instance/i-001861022fbcac290	

[Details](#) | [Status and alarms](#) | [Monitoring](#) | [Security](#) | [Networking](#) | [Storage](#) | [Tags](#)

EC2 > Instances > i-001861022fbcac290

Instance summary for i-001861022fbcac290 (InstantLibraryApp) [Info](#)

Updated less than a minute ago

Instance ID i-001861022fbcac290	Public IPv4 address -	Private IPv4 addresses 172.31.3.5
IPv6 address -	Instance state Stopped	Public IPv4 DNS -
Hostname type IP name: ip-172-31-3-5.ap-south-1.compute.internal	Private IP DNS name (IPv4 only) ip-172-31-3-5.ap-south-1.compute.internal	Elastic IP addresses -
Answer private resource DNS name IPv4 (A)	Instance type t2.micro	AWS Compute Optimizer finding Opt-in to AWS Compute Optimizer for recommendations. Learn more
Auto-assigned IP address -	VPC ID vpc-03cdc7b6f19dd7211	Auto Scaling Group name -
IAM Role sns_Dynamodb_role	Subnet ID subnet-0d9fa3144480cc9a9	
IMDSv2 Required	Instance ARN arn:aws:ec2:ap-south-1:1557690616836:instance/i-001861022fbcac290	

[Connect](#) | [Instance state ▾](#) | [Actions ▾](#)
[Manage instance state](#) | [Instance settings](#) | [Networking](#) | [Security](#) | [Image and templates](#) | [Monitor and troubleshoot](#)

EC2 > Instances > i-001861022fbcac290 > Modify IAM role

Modify IAM role [Info](#)

Attach an IAM role to your instance.

Instance ID
[i-001861022fbcac290 \(InstantLibraryApp\)](#)

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

[▼](#) [Create new IAM role](#)

[Cancel](#) [Update IAM role](#)

- Now connect the EC2 with the files

Connect to instance Info

Connect to your instance i-001861022fbcac290 (InstantLibraryApp) using any of these options

EC2 Instance Connect Session Manager SSH client EC2 serial console

⚠ Port 22 (SSH) is open to all IPv4 addresses

Port 22 (SSH) is currently open to all IPv4 addresses, indicated by **0.0.0.0/0** in the inbound rule in [your security group](#). For increased security, consider restricting access to only the EC2 Instance Connect service IP addresses for your Region: 13.233.177.0/29. [Learn more](#).

Instance ID

Connection Type

Connect using EC2 Instance Connect
Connect using the EC2 Instance Connect browser-based client, with a public IPv4 or IPv6 address.

Public IPv4 address

IPv6 address

Connect using EC2 Instance Connect Endpoint
Connect using the EC2 Instance Connect browser-based client, with a private IPv4 address and a VPC endpoint.

Username Enter the username defined in the AMI used to launch the instance. If you didn't define a custom username, use the default username, ec2-user.

X

ⓘ Note: In most cases, the default username, ec2-user, is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI username.

```
A newer release of "Amazon Linux" is available.
Version 2023.6.20241010:
Run "/usr/bin/dnf check-release-update" for full release and version update info
      #
      #####      Amazon Linux 2023
      #####\      https://aws.amazon.com/linux/amazon-linux-2023
      #\      V~.--->
      ~\      /m/
Last login: Tue Oct 15 04:17:59 2024 from 13.233.177.3
[ec2-user@ip-172-31-3-5 ~]$ █
```

Milestone 6: Deployment on EC2

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: 'git clone <https://github.com/your-github-username/your-repository-name.git>'

Note: change your-github-username and your-repository-name with your credentials here: 'git clone'

[https://github.com/AlekhyaPenubakula/InstantLibrary.git'](https://github.com/AlekhyaPenubakula/InstantLibrary.git)

- This will download your project to the EC2 instance.

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

- cd Homemadepicklesandsnacks

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

Run the Flask Application

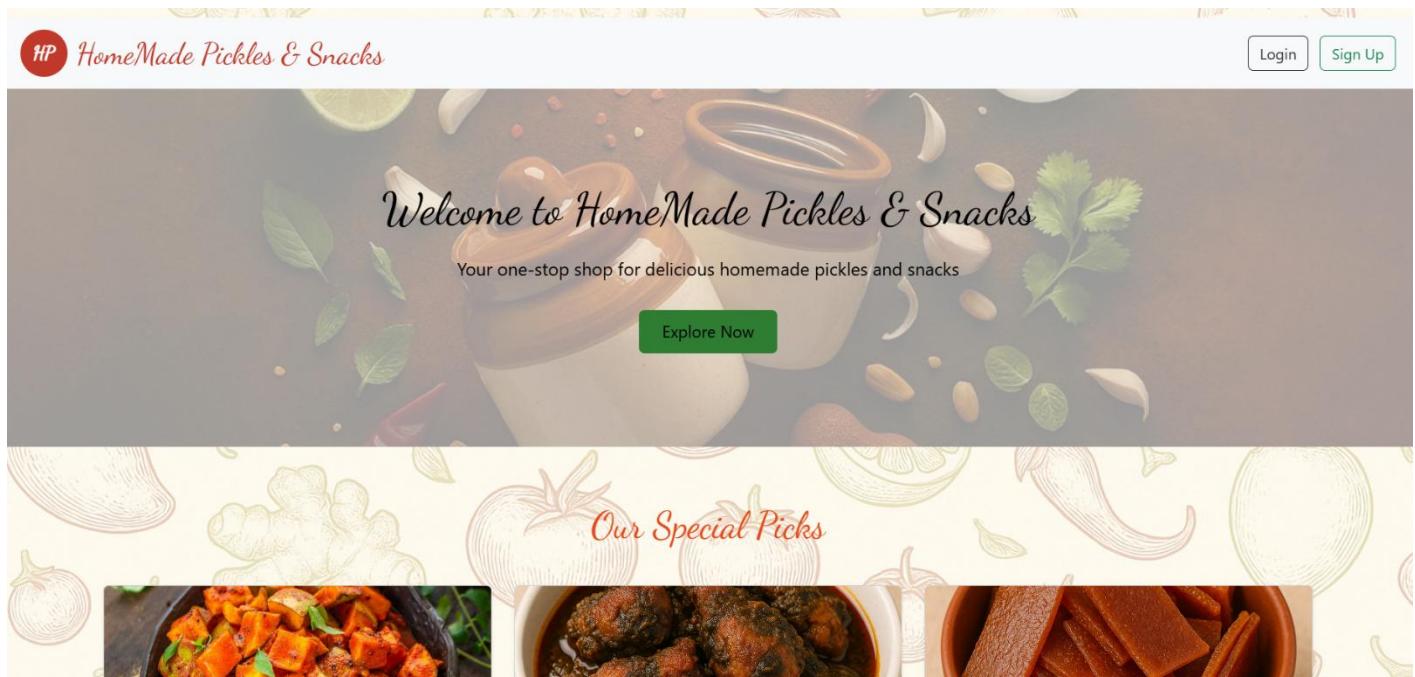
```
sudo flask run --host=0.0.0.0 --port=80
```

```
Microsoft Windows [Version 10.0.26100.4652]
(c) Microsoft Corporation. All rights reserved.

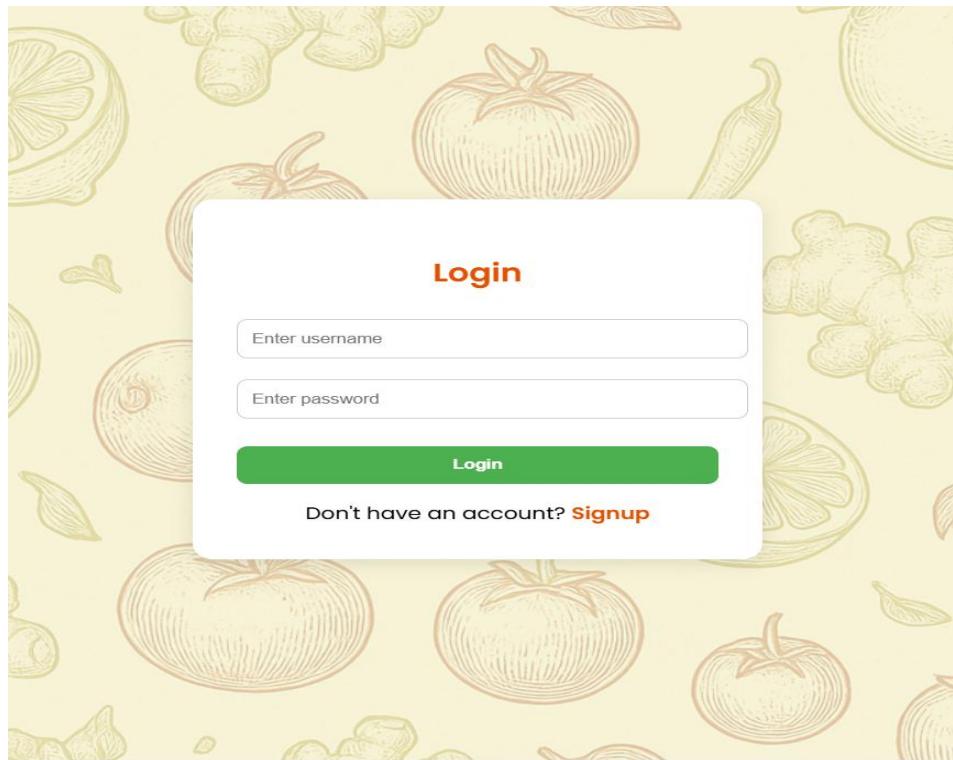
C:\Users\srava\OneDrive\Desktop\pr1>python app.py
 * Tip: There are .env files present. Install python-dotenv to use them.
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Tip: There are .env files present. Install python-dotenv to use them.
 * Debugger is active!
 * Debugger PIN: 517-909-220
```

Milestone 7: Testing and Deployment

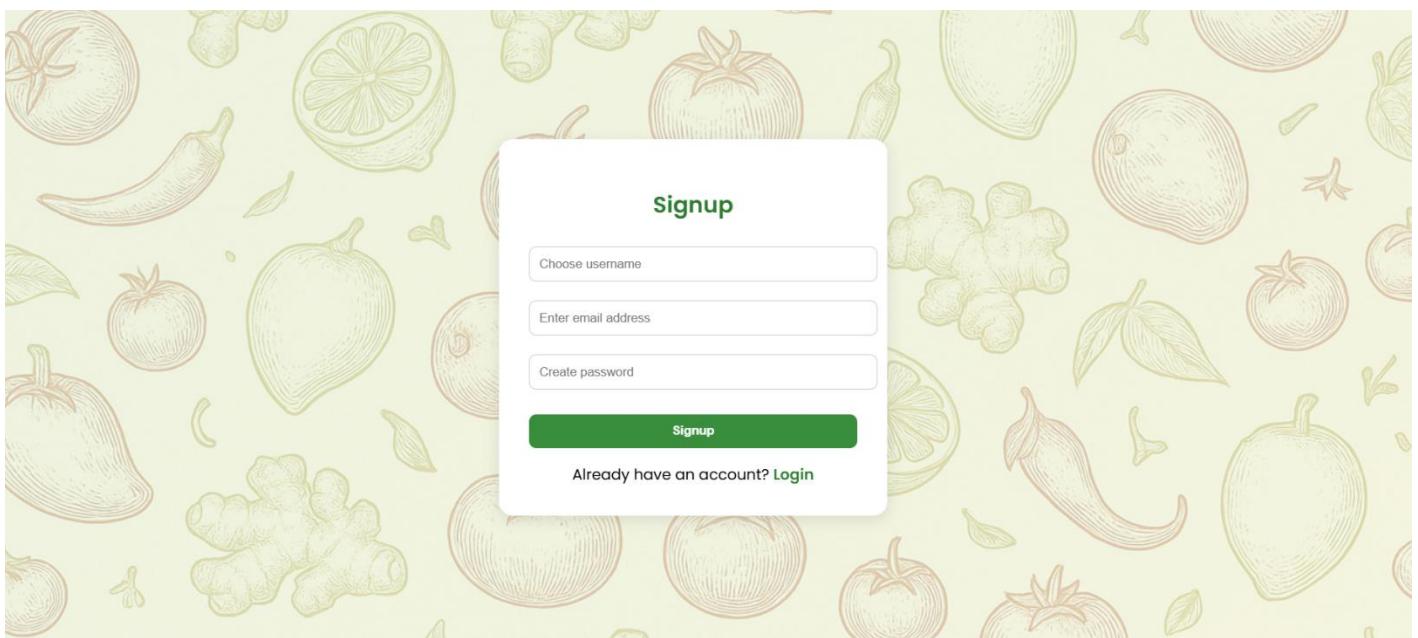
Welcome Page:



Login Page:



Sign in Page:



Home Page:

HP HomeMade Pickles & Snacks

Top Selling Pickles



Usirikaya Pickle

Weight:

250g - ₹100

Add to Cart



Aavakaya Pickle

Weight:

250g - ₹110

Add to Cart



Mirapakaya Pickle

Weight:

250g - ₹120

Add to Cart

Summer Specials

Veg Pickles Page:

HP HomeMade Pickles & Snacks

Veg Pickles



Usirikaya Pickle

Weight:

250g - ₹100

Qty:

Add to Cart



Aavakaya Pickle

Weight:

250g - ₹110

250g - ₹110

500g - ₹180

1kg - ₹310

Add to Cart



Mirapakaya Pickle

Weight:

250g - ₹120

Qty:

Add to Cart



Traditional Mango Pickle

Weight:

250g - ₹130

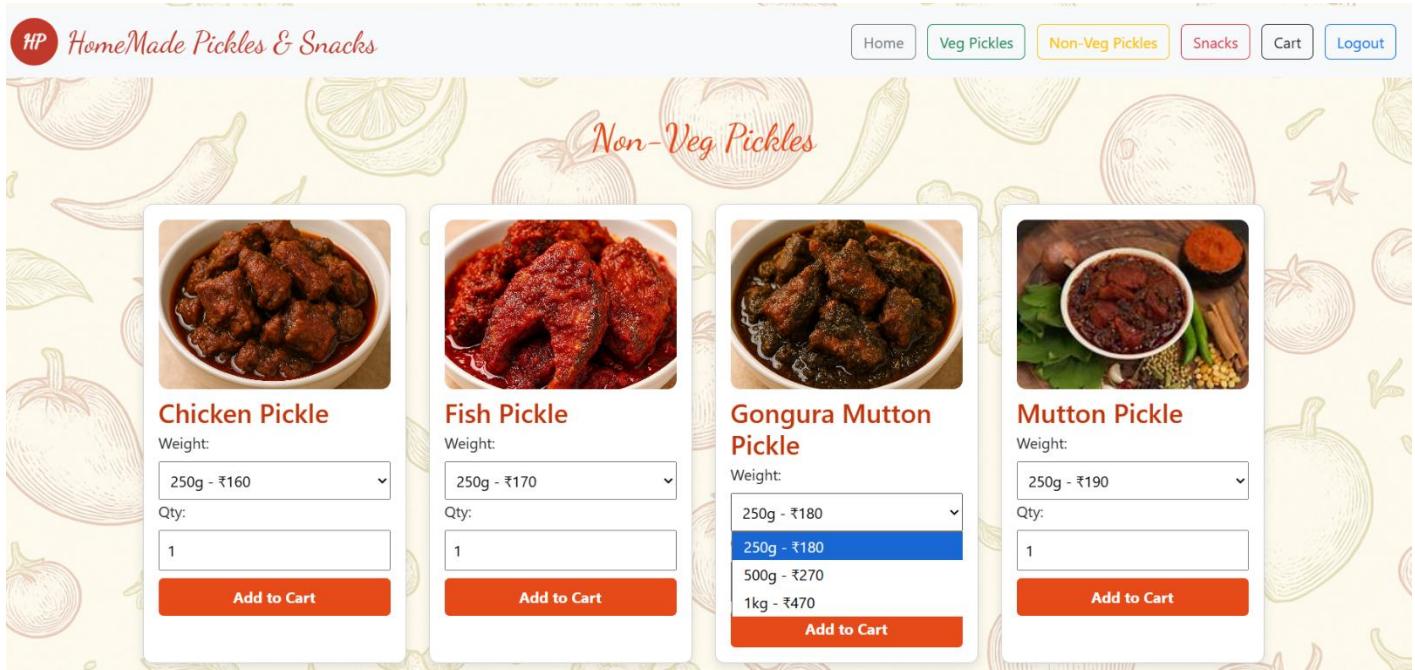
Qty:

Add to Cart

Non Veg Pickles Page:

HM HomeMade Pickles & Snacks

Non-Veg Pickles



The screenshot shows a grid of four product cards for non-veg pickles:

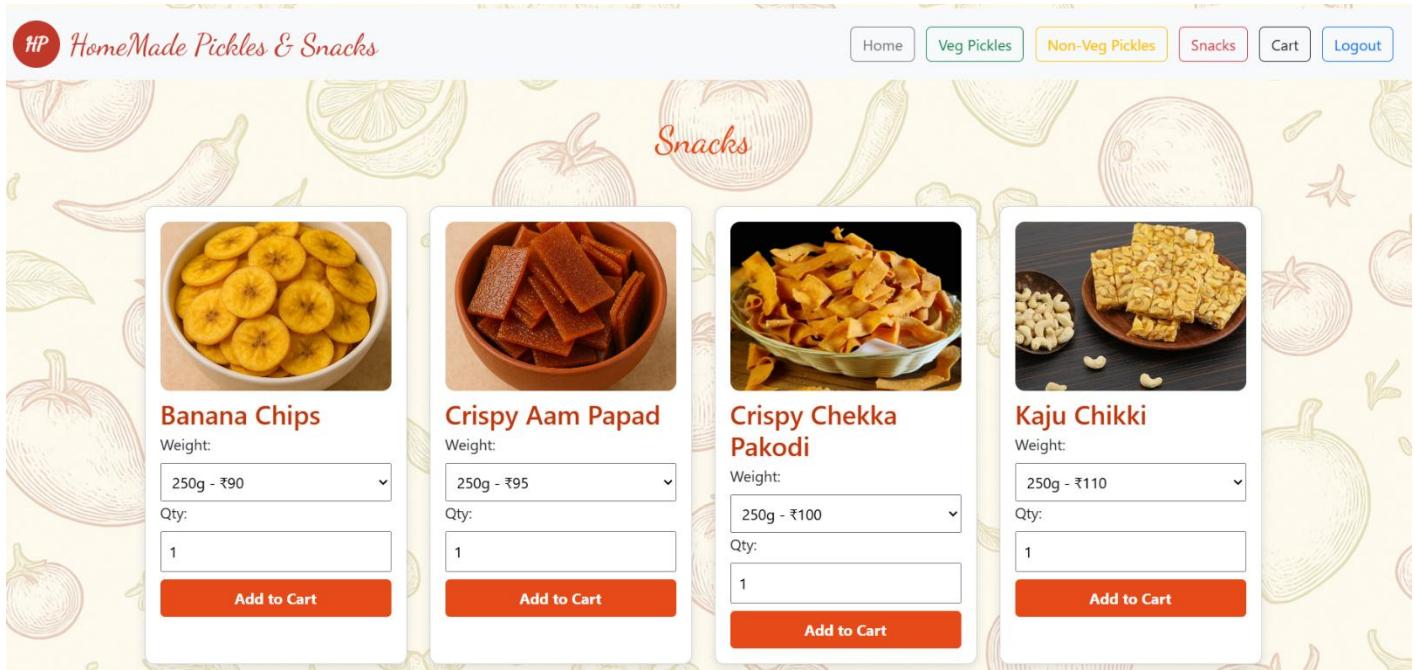
- Chicken Pickle**: Weight dropdown (250g - ₹160), Qty: 1, Add to Cart button.
- Fish Pickle**: Weight dropdown (250g - ₹170), Qty: 1, Add to Cart button.
- Gongura Mutton Pickle**: Weight dropdown (250g - ₹180 selected, 500g - ₹270, 1kg - ₹470), Qty: 1, Add to Cart button.
- Mutton Pickle**: Weight dropdown (250g - ₹190), Qty: 1, Add to Cart button.

Navigation bar at the top: Home, Veg Pickles, Non-Veg Pickles (highlighted in yellow), Snacks, Cart, Logout.

Snacks Page:

HM HomeMade Pickles & Snacks

Snacks

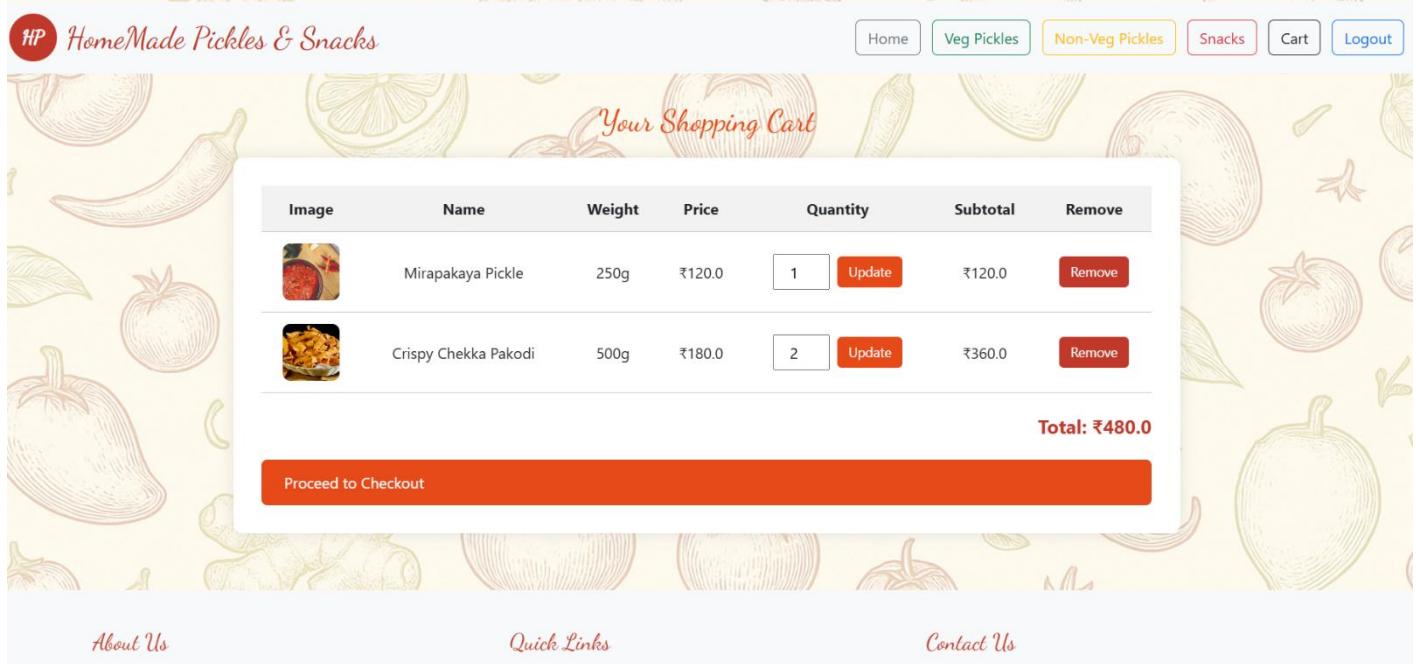


The screenshot shows a grid of four snack products:

- Banana Chips**: Weight dropdown (250g - ₹90), Qty: 1, Add to Cart button.
- Crispy Aam Papad**: Weight dropdown (250g - ₹95), Qty: 1, Add to Cart button.
- Crispy Chekka Pakodi**: Weight dropdown (250g - ₹100), Qty: 1, Add to Cart button.
- Kaju Chikki**: Weight dropdown (250g - ₹110), Qty: 1, Add to Cart button.

Navigation bar at the top: Home, Veg Pickles, Non-Veg Pickles, Snacks (highlighted in pink), Cart, Logout.

Cart Page:



The screenshot shows the 'Your Shopping Cart' page for 'HomeMade Pickles & Snacks'. The cart contains two items: 'Mirapakaya Pickle' (250g, ₹120.0) and 'Crispy Chekka Pakodi' (500g, ₹180.0). The total amount is ₹480.0. Navigation links at the bottom include 'About Us', 'Quick Links', and 'Contact Us'.

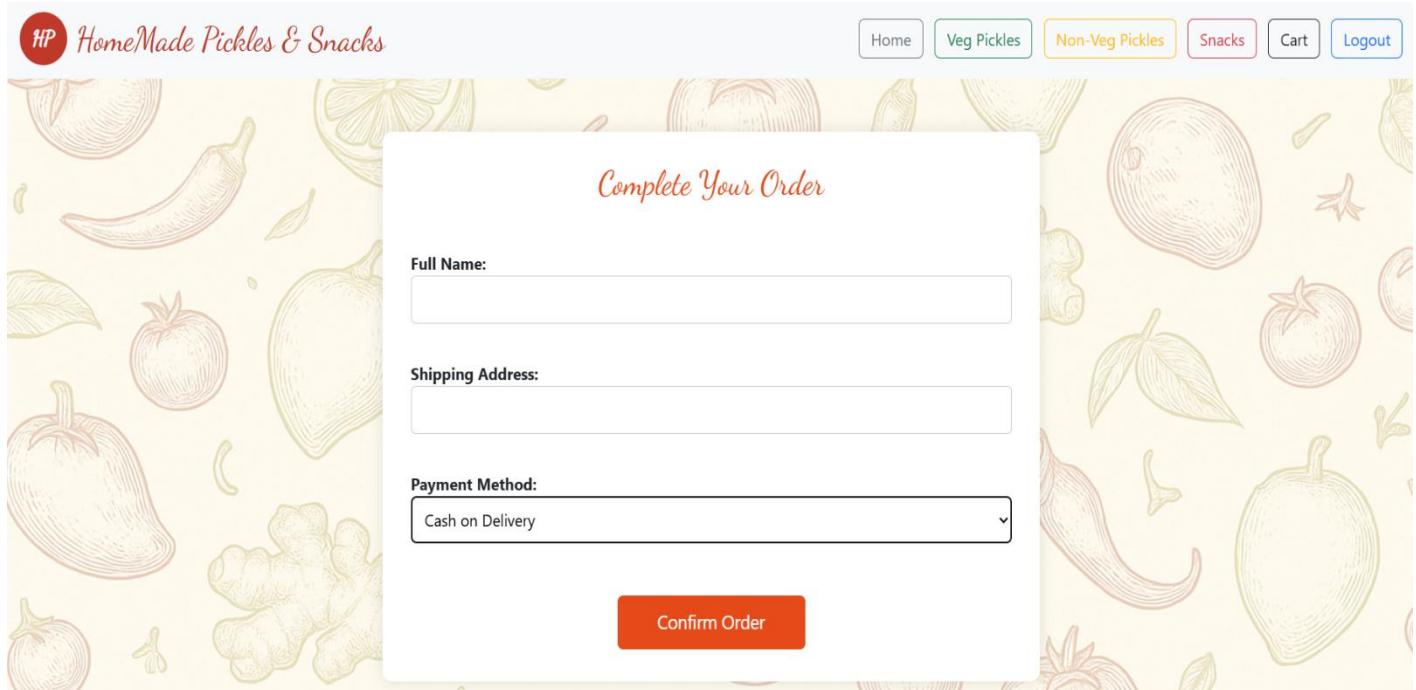
Image	Name	Weight	Price	Quantity	Subtotal	Remove
	Mirapakaya Pickle	250g	₹120.0	1	₹120.0	<button>Remove</button>
	Crispy Chekka Pakodi	500g	₹180.0	2	₹360.0	<button>Remove</button>

Total: ₹480.0

[Proceed to Checkout](#)

[About Us](#) [Quick Links](#) [Contact Us](#)

Checkout Page:



The screenshot shows the 'Complete Your Order' page for 'HomeMade Pickles & Snacks'. It includes fields for 'Full Name', 'Shipping Address', and 'Payment Method' (set to 'Cash on Delivery'). A 'Confirm Order' button is at the bottom.

Complete Your Order

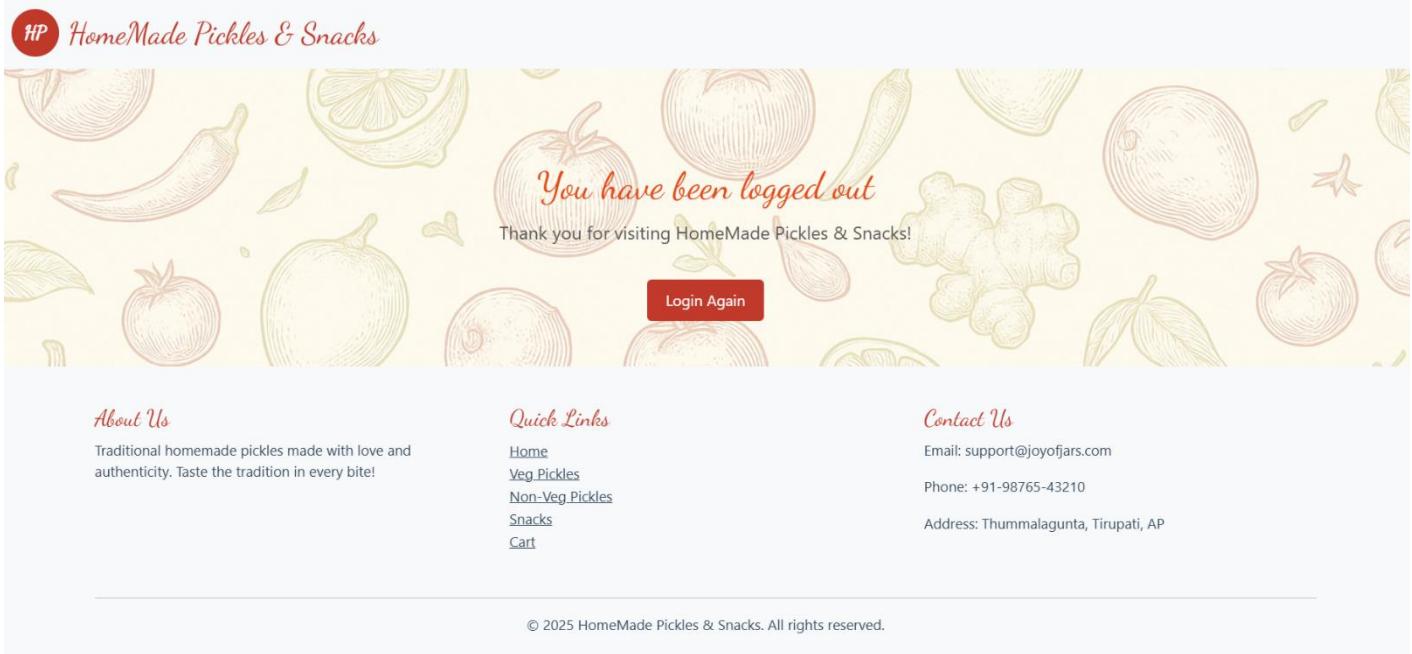
Full Name:

Shipping Address:

Payment Method:

[Confirm Order](#)

Log Out Page:



HP HomeMade Pickles & Snacks

You have been logged out

Thank you for visiting HomeMade Pickles & Snacks!

[Login Again](#)

About Us
Traditional homemade pickles made with love and authenticity. Taste the tradition in every bite!

Quick Links
[Home](#)
[Veg Pickles](#)
[Non-Veg Pickles](#)
[Snacks](#)
[Cart](#)

Contact Us
Email: support@joyofjars.com
Phone: +91-98765-43210
Address: Thummalagunta, Tirupati, AP

© 2025 HomeMade Pickles & Snacks. All rights reserved.

Conclusion:

The Homemade Pickles and Snacks platform has been meticulously crafted to deliver a seamless and delightful experience for food enthusiasts seeking authentic, handcrafted flavors. By leveraging modern web technologies such as Flask for backend logic, secure user authentication, and dynamic cart management, the platform ensures a user-friendly interface for browsing, customizing, and ordering artisanal pickles and snacks.

The integration of cloud-ready architecture (e.g., AWS for future scalability) and robust session management allows the platform to handle high traffic efficiently while maintaining real-time updates for orders and inventory. Features like weight-based pricing, category-specific searches, and instant checkout streamline the shopping process, empowering customers to explore a diverse range of traditional and innovative recipes with ease.

This project addresses the growing demand for homemade, preservative-free food products by bridging the gap between small-scale producers and discerning customers. The platform's intuitive design and secure payment workflows enhance trust and convenience, while backend tools enable effortless inventory tracking and order fulfillment for administrators.

By combining time-honored recipes with modern e-commerce capabilities, this website not only preserves culinary heritage but also adapts to the digital age, ensuring that every jar of pickle or snack reaches customers with the same care and quality as a homemade meal. As the platform



evolves, it stands ready to scale, introduce new product lines, and foster a community of food lovers united by a passion for authentic flavors.

In essence, this project redefines the way homemade delicacies are shared and enjoyed, offering a flavorful bridge between tradition and technology.