

# Home Made Pickles and Snacks

## 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.

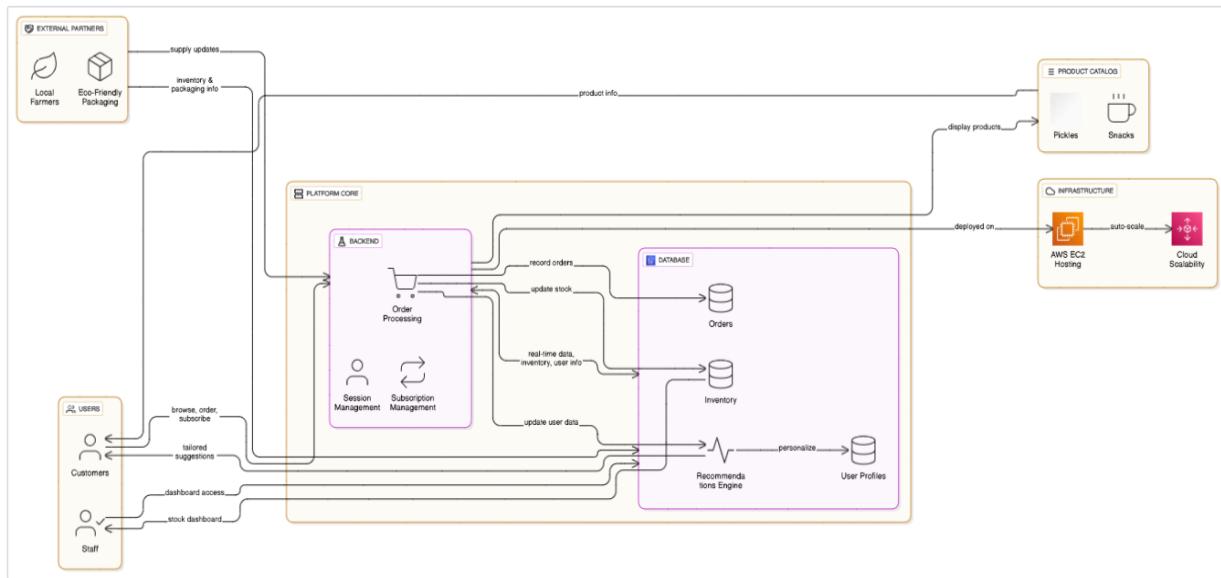
### Scenario 3: Personalized User Experience and Recommendations

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

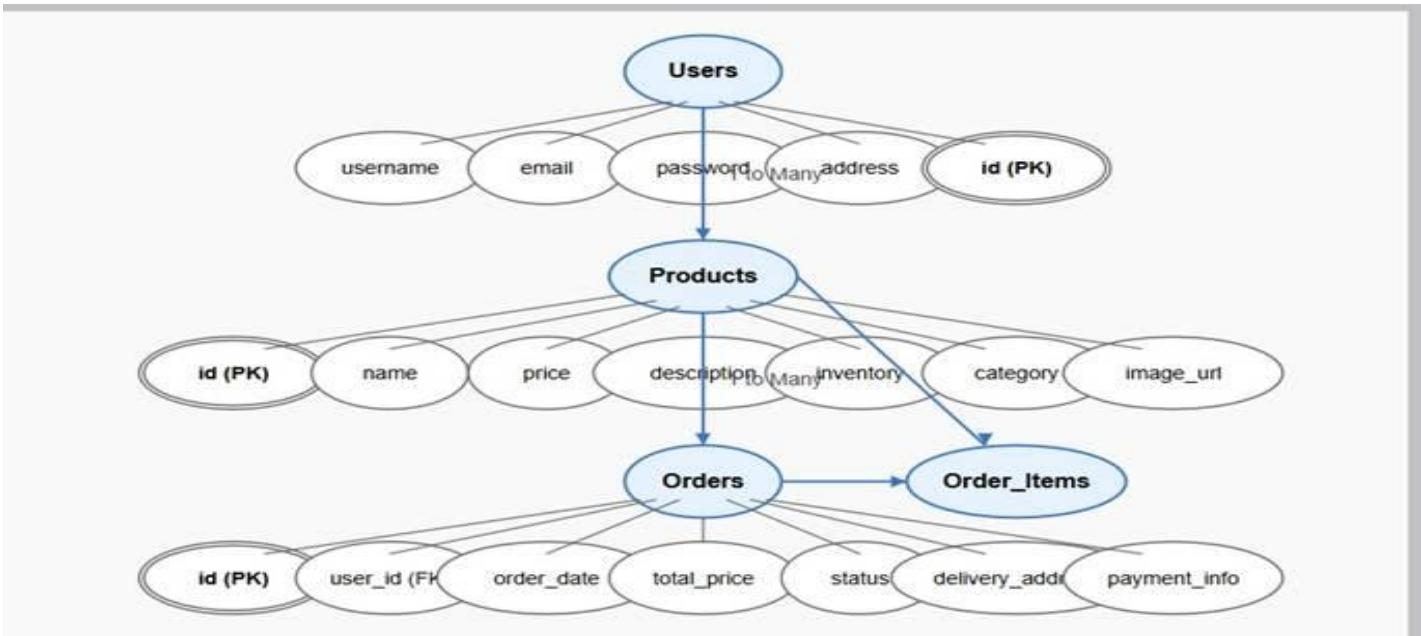
This AWS-based architecture powers a scalable and secure web application using Amazon EC2 for hosting the backend, with a lightweight framework like Flask handling core logic.

Application data is stored in Amazon DynamoDB, ensuring fast, reliable access, while user access is managed through AWS IAM for secure authentication and control. Real-time alerts and system notifications are enabled via Amazon SNS, enhancing communication and user engagement.



## Entity Relationship (ER) Diagram:

An ER (Entity-Relationship) diagram visually represents the logical structure of a database by defining entities, their attributes, and the relationships between them. It helps organize data efficiently by illustrating how different components of the system interact and relate. This structured approach supports effective database normalization, data integrity, and simplified query design.



## Pre-requisites

- ❖ AWS Account Setup:  
<https://docs.aws.amazon.com/accounts/latest/reference/getting-started.html>
- ❖ AWS IAM (Identity and Access Management):  
<https://docs.aws.amazon.com/IAM/latest/UserGuide/introduction.html>
- ❖ AWS EC2 (Elastic Compute Cloud):  
<https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts.html>
- ❖ AWS DynamoDB:  
<https://docs.aws.amazon.com/amazondynamodb/Introduction.html>
- ❖ Git Documentation:  
<https://git-scm.com/doc>
- ❖ VS Code Installation: (download the VS Code using the below link or you can get that in Microsoft store)  
<https://code.visualstudio.com/download>

# Project 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 it is not done yet.
- 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 receive 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: Web Application Development and Setup

Backend Development and Application Setup focuses on establishing the core structure of the application. This includes configuring the backend framework, setting up routing, and integrating database connectivity. It lays the groundwork for handling user interactions, data management, and secure access.

## Important Instructions:

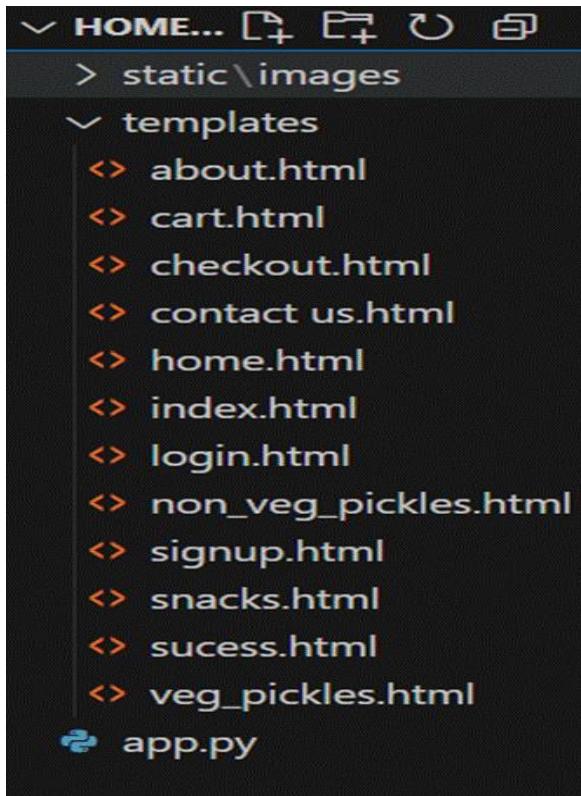
- Start by creating the necessary HTML pages and Flask routes (app.py) to build the core functionality of your application.
- During the initial development phase, store and retrieve data using Python dictionaries or lists locally. This will allow you to design, test, and validate your application logic without external database dependencies
- Ensure your app runs smoothly with local data structures before integrating any cloud services.

## Post Troven Access Activation:

- Once Troven Labs access is provided (valid for 3 hours), you must immediately proceed with Milestone 1 of your Guided Project instructions.
- At this point, modify your app.py and replace local dictionary/list operations with AWS services (such as DynamoDB, RDS, or others as per project requirements).
- Using the temporary credentials provided by Troven Labs, securely connect your application to AWS resources.
- Since the AWS configuration is lightweight and already instructed in the milestones, you should be able to complete the cloud integration efficiently within the allotted time.

## LOCAL DEPLOYMENT

- 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 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': 'Kaiju Chikkis', '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('home'))
    
```

- 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'])

```

 Restart Visual Studio

- 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')           ⏪ 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('success', 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('/success')
def success():
    return render_template('success.html')

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

```

## Milestone 2: AWS Account Setup

### Important Notice: Use Troven Labs for AWS Access

Students are strictly advised not to create their own AWS accounts, as doing so may incur charges. Instead, we have set up a dedicated section called “Labs” on the Troven platform, which provides temporary and cost-free access to AWS services.

Once your website is locally deployed and fully functional, you must proceed with integrating AWS services only through the Troven Labs environment. This ensures secure, controlled access to AWS resources without any risk of personal billing.

All steps involving AWS (such as deploying to EC2, connecting to DynamoDB, or using SNS) must be carried out within the Troven Labs platform, as we've configured temporary credentials for each student.

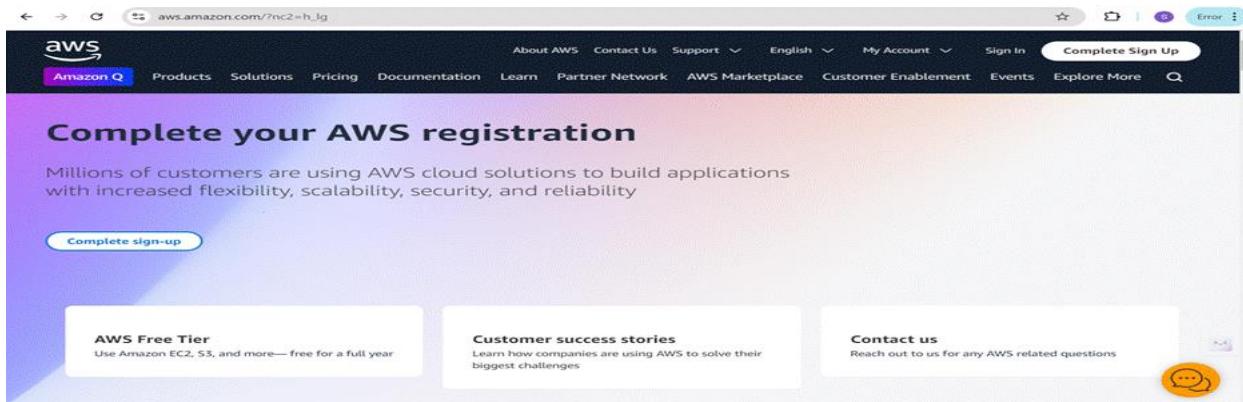
**Reminder: You must complete the Web Development task before gaining access to Troven. Once accessed, the AWS Console via Troven is available for only 3 hours—please plan your work accordingly.**

Please follow the provided guidelines and access AWS exclusively through Troven to avoid unnecessary issues.

### AWS Account Setup and Login

**This is for your understanding only. Please refrain from creating an AWS account. A temporary account will be provided via Troven.**

- 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 account.



**Sign up for AWS**

**Explore Free Tier products with a new AWS account.**

To learn more, visit [aws.amazon.com/free](https://aws.amazon.com/free).



**Root user email address**  
Used for account recovery and as described in the [AWS Privacy Notice](#)

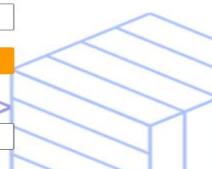
**AWS account name**  
Choose a name for your account. You can change this name in your account settings after you sign up.

**Verify email address**

OR

**Sign in to an existing AWS account**



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

**Sign in**

**Root user**  
Account owner that performs tasks requiring unrestricted access. [Learn more](#)

**IAM user**  
User within an account that performs daily tasks. [Learn more](#)

**Root user email address**

**Next**

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

## AI Use Case Explorer

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

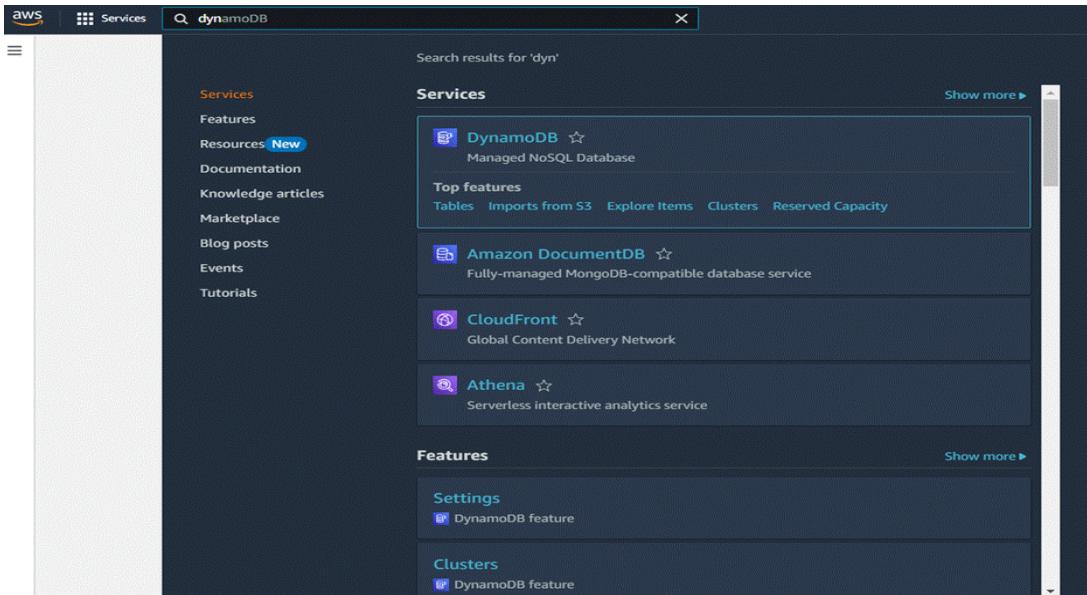
**Explore now >**

## Milestone 3: DynamoDB Database Creation and Setup

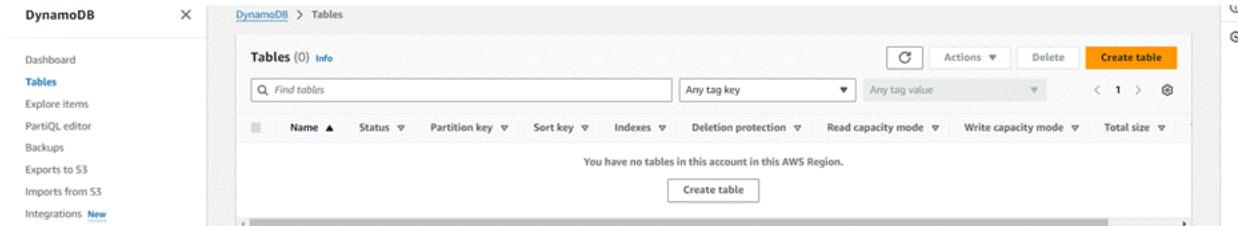
Database Creation and Setup involves initializing a cloud-based NoSQL database to store and manage application data efficiently. This step includes defining tables, setting primary keys, and configuring read/write capacities. It ensures scalable, high-performance data storage for seamless backend operations.

## Navigate to the DynamoDB

- In the AWS Console, navigate to DynamoDB and click Create Tables.



A screenshot of the DynamoDB Dashboard. The left sidebar shows navigation links for "Tables", "Explore items", "PartiQL editor", "Backups", "Exports to S3", "Imports from S3", "Integrations", "Reserved capacity", and "Settings". A collapsed section for "DAX" lists "Clusters", "Subnet groups", "Parameter groups", and "Events". The main dashboard area has two sections: "Alarms (0)" and "DAX clusters (0)". On the right side, there is a "Create resources" section with a large orange "Create table" button. Below it, there is information about the Amazon DynamoDB Accelerator (DAX) and a "Create DAX cluster" button. At the bottom right, there is a "What's new" section with a note about AWS Cost Management providing 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 tables.

The screenshot shows the 'Create table' wizard. At the top, there's a navigation bar with the AWS logo, a search bar, and a 'Create table' button. Below the navigation bar, the path 'DynamoDB > Tables > Create table' is visible. The main section is titled 'Create table'. Under 'Table details', it says 'Table name' and has a field containing 'Users'. Below the table name, there's a note: 'Between 3 and 255 characters, containing only letters, numbers, underscores (\_), hyphens (-), and periods (.).'. Under 'Partition key', it says 'Username' and 'String'. Below the partition key, there's a note: '1 to 255 characters and case sensitive.'. Under 'Sort key - optional', it says 'Enter the sort key name' and 'String'. Below the sort key, there's a note: '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** X Tables (1) Info Actions ▾ Delete Create table

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.

**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.

**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](#)

## Tables (2) [Info](#)



Action

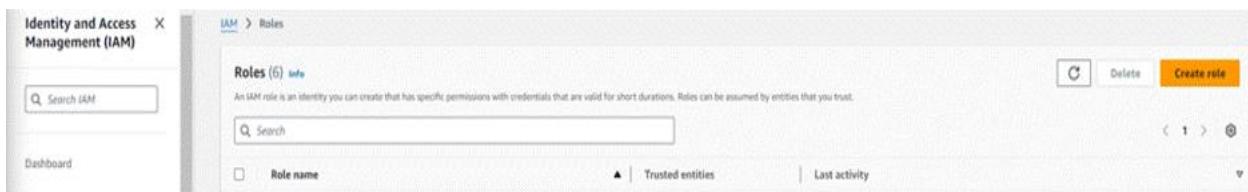
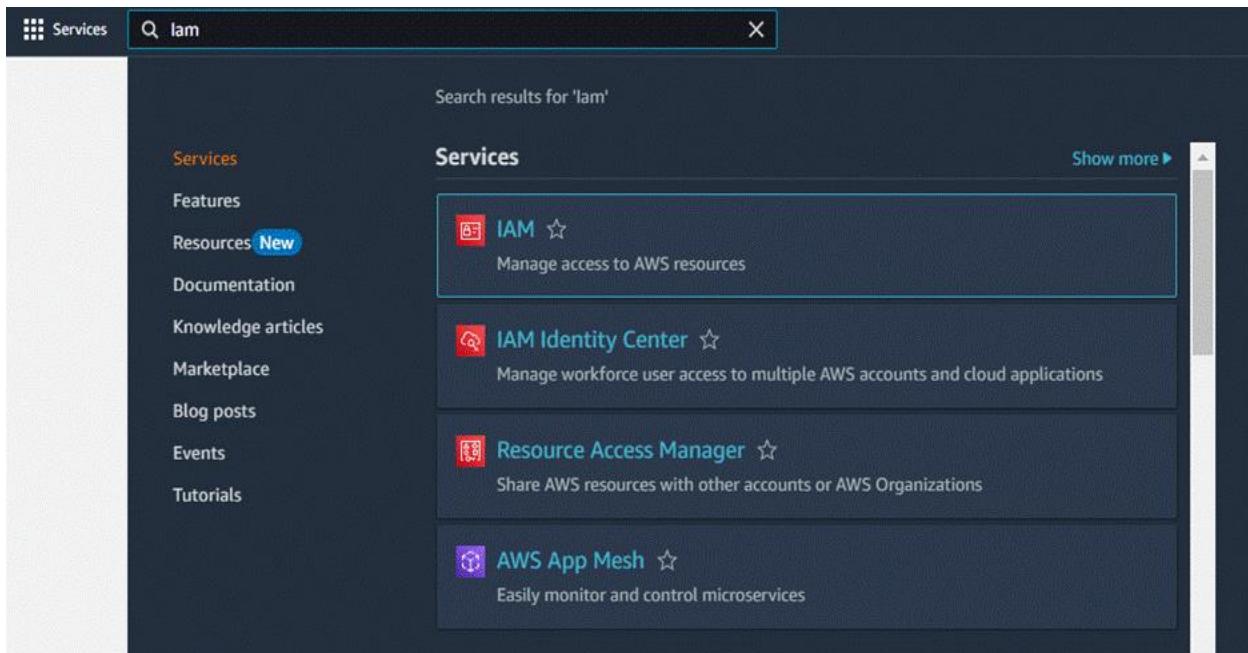
<input type="checkbox"/>	Name ▲	Status ▼	Partition key ▼	Sort key ▼	Indexes ▼	Replication Regions ▼	Deletion protecti
<input type="checkbox"/>	<a href="#">Orders</a>	<span>Active</span>	order_id (\$)	-	0	0	<span>Off</span>
<input type="checkbox"/>	<a href="#">Users</a>	<span>Active</span>	username (\$)	-	0	0	<span>Off</span>

## Milestone 4: IAM Role Setup

The IAM (Identity and Access Management) role setup involves creating roles that define specific permissions for AWS services. To set it up, you create a role with the required policies, assign it to users or services, and ensure the role has appropriate access to resources like EC2, S3, or RDS. This allows controlled access and ensures security best practices in managing AWS resources.

### Create an IAM Role.

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



**Step 1: Select trusted entity**

**Trusted entity type**

- AWS service: Allow AWS services like EC2, 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.
- Custom trust policy: Create a custom trust policy to enable others to perform actions in this account.

**Service or use case**

EC2

**Use case**

Allow an AWS service like EC2 Lambda, or others to perform actions in this account.

Allow EC2 instances to call AWS Lambda on your behalf.

EC2 Role for AWS Systems Manager

EC2 Spot Fleet Role

EC2 Auto Scaling Role

EC2 Spot Fleet Tagging Role

EC2 Spot Instances Role

EC2 Spot Instances Role with Tags

EC2 Spot Instances Role with Tags & Launch Permissions

EC2 Scheduled Instances Role

EC2 Scheduled Instances Role with Tags

**Next Step**

**Step 2: Add permissions**

**Add permissions**

**Permissions policies (1/955)**

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

Filter by Type: All types | 2 matches

Policy name:

AmazonDynamoDBFullAccess

AmazonDynamoDBFullAccessPolicy

**Set permissions boundary - optional**

**Cancel** **Previous** **Next**

## Attach Policies

Attach the following policies to the role:

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

**Step 3: Name, review, and create**

**Name, review, and create**

**Role details**

**Role name**:

**Description**:

**Select trusted entities**

**Step 2: Add permissions**

**Permissions policy summary**

Policy name	Type	Attached as
<a href="#">AmazonDynamoDBFullAccess</a>	AWS managed	Permissions policy
<a href="#">AmazonDynamoDBFullAccessPolicy</a>	AWS managed	Permissions policy

**Step 3: Add tags**

Add tags - optional

Tags are key-value pairs that you can use to identify, organize, or search for resources.

No tags associated with this resource.

**Add new tag**

You can add up to 50 tags.

**Cancel** **Previous** **Create role**

The screenshot shows the AWS IAM Roles page for the role 'sns\_Dynamodb\_role'. The 'Summary' tab is selected, displaying details like ARN, creation date (October 13, 2024, 23:06 UTC+05:30), and last activity (6 days ago). The 'Permissions' tab is also visible, showing two managed policies attached: 'AmazonDynamoDBFullAccess' and 'AmazonSNSFullAccess'.

## Milestone 5: EC2 Instance Setup

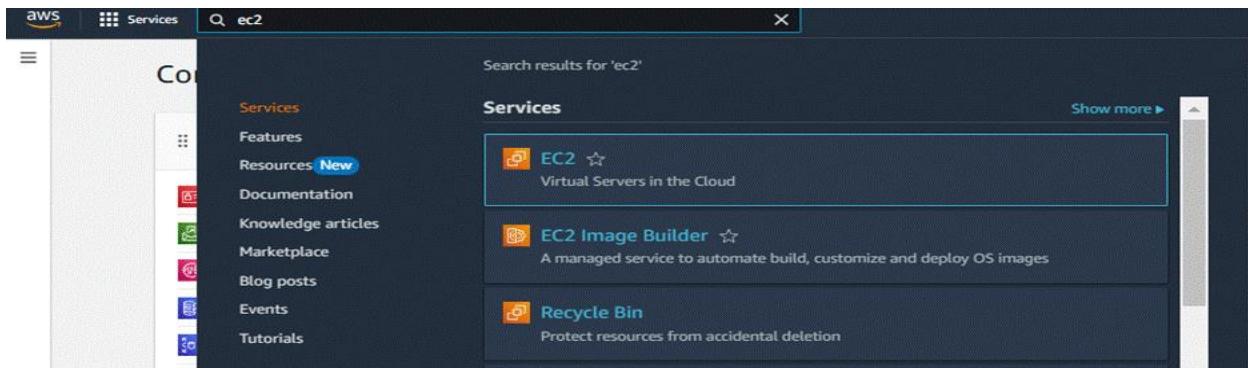
To set up a public EC2 instance, choose an appropriate Amazon Machine Image (AMI) and instance type. Ensure the security group allows inbound traffic on necessary ports (e.g., HTTP/HTTPS for web applications). After launching the instance, associate it with an Elastic IP for consistent public access,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

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

The screenshot shows a GitHub repository interface. On the left, there's a list of files: 'static', 'templates', '.env', and 'app.py'. The 'Create style.css' file has a dropdown menu open, showing 'Local' and 'Codespaces' tabs. Under 'Local', there are options to 'Clone' via 'HTTPS', 'SSH', or 'GitHub CLI'. The 'HTTPS' link is highlighted. Below that, there's a button to 'Clone using the web URL'. At the bottom of the dropdown, there are links for 'Open with GitHub Desktop' and 'Download ZIP'.

## Launch an EC2 instance to host the Flask

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



- Click on Launch instance to launch EC2 instance

EC2 Dashboard    Instances Info

Last updated less than a minute ago    Connect    Instance state ▾    Actions ▾    Launch instances ▾

Find Instance by attribute or tag (case-sensitive)    All states ▾

No instances  
You do not have any instances in this region

Launch instances

**EC2 > Instances > Launch an instance**

It seems like you may be new to launching instances in EC2. Take a walkthrough to learn about EC2, how to launch instances and about best practices    Do not show me this again

### Launch an instance Info

Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

**Name and tags Info**

Name  Add additional tags

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

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    macOS    Ubuntu    Windows    Red Hat    SUSE Linux    Debian

Browse more AMIs    Including AMIs from

▼ Sum  
Number of 1  
Software 1 Amazon Li ami-002f6e5  
Virtual ser t2.micro  
Firewall 5 New secur  
Storage 1 volume(s)  
Free acc t2.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** 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**

**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)**

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

ⓘ 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 this again](#)

## Launch an instance [Info](#)

Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

### Name and tags [Info](#)

Name

[Add additional tags](#)

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

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

[Recent](#)
[Quick Start](#)

[macOS](#)
[Ubuntu](#)
[Windows](#)
[Red Hat](#)
[SUSE Linux](#)
[Debian](#)
[Browse more AMIs](#)  
 Including AMIs from
 

ⓘ Free account  
t2.micro  
+2 more

[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

64-bit (x86)

Boot mode

uefi-preferred

AMI ID

ami-078264b8ba71bc45e

Username

ec2-user

[Verified provider](#)

### ▼ 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

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

#### Instance type

t2.micro

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

All generations

[Compare instance types](#)

ⓘ **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 GiB of bandwidth to the internet.

### ▼ 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](#)

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

## Configure security groups for HTTP and SSH access.

**▼ Network settings** [Info](#)

VPC – required [Info](#)  
vpc-03cdc7b6f19dd7211 (default) ▾ [Edit](#)

Subnet [Info](#)  
No preference ▾ [Edit](#) [Create new subnet](#) [Edit](#)

Auto-assign public IP [Info](#)  
Enable ▾ [Edit](#)

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 [Edit](#)  Select existing security group [Edit](#)

Security group name – required  
 [Edit](#)

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](#)  
 [Edit](#)

**Inbound Security Group Rules**

**▼ Security group rule 1 (TCP, 22, 0.0.0.0/0)** [Remove](#)

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

**▼ Security group rule 2 (TCP, 80, 0.0.0.0/0)** [Remove](#)

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

**▼ Security group rule 3 (TCP, 5000, 0.0.0.0/0)** [Remove](#)

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

[Add security group rule](#)

EC2 > ... > Launch an instance

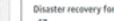
**Success**  
Successfully initiated launch of instance i-001861022fbac290

▶ 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 >

Create billing and free tier usage alerts  Once your instance is running, log into it from your local computer.  Create billing alerts   Connect to instance 	Connect to your instance  Configure the connection between an EC2 instance and a database to allow traffic flow between them.  Connect to RDS database   Create a new RDS database   Learn more 	Create an RDS database  Create EBS snapshot policy  Manage detailed monitoring  Create Load Balancer	Manage detailed monitoring  Create EBS snapshot policy  Create a policy that automates the creation, retention, and deletion of EBS snapshots  Manage detailed monitoring 	Manage detailed monitoring  Enable or disable detailed monitoring for the instance. If you enable detailed monitoring, the Amazon EC2 console displays monitoring graph with a 1-minute period.  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 the 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) 											
Last updated  Connect Instance state Actions 											
Find Instance by attribute or tag (case-sensitive) 											
Name 	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP	IPv6 IPs	Monitoring
<input checked="" type="checkbox"/> InstantLibrary... 	i-001861022fbac290	 Stopped  	t2.micro	-	 View alarms 	ap-south-1b	-	-	-	-	disabled  launch-w...

[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	172.31.5.5
IPv6 address	-	Instance state	Stopped	Public IPv4 DNS	-
Hostname type	IP name: ip-172-31-5-5.ap-south-1.compute.internal	Private IP DNS name (IPv4 only)	ip-172-31-5-5.ap-south-1.compute.internal	AWS Compute Optimizer finding	<a href="#">Opt-in to AWS Compute Optimizer for recommendations.   Learn more</a>
Answer private resource DNS name	ipv4 (A)	Instance type	t2.micro	Auto Scaling Group name	-
Auto-assigned IP address	-	VPC ID	vpc-03cdc7b6f19dd7211	Elastic IP addresses	-
IAM Role	<a href="#">sns_Dynamodb_role</a>	Subnet ID	subnet-0d9fa3144480cc9a9	AWS Compute Optimizer finding	<a href="#">Opt-in to AWS Compute Optimizer for recommendations.   Learn more</a>
IMDSv2	Required	Instance ARN	arn:aws:ec2:ap-south-1:557690616836:instance/i-001861022fbcac290	Auto Scaling Group name	-

[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	172.31.5.5
IPv6 address	-	Instance state	Stopped	Public IPv4 DNS	-
Hostname type	IP name: ip-172-31-5-5.ap-south-1.compute.internal	Private IP DNS name (IPv4 only)	ip-172-31-5-5.ap-south-1.compute.internal	AWS Compute Optimizer finding	<a href="#">Opt-in to AWS Compute Optimizer for recommendations.   Learn more</a>
Answer private resource DNS name	ipv4 (A)	Instance type	t2.micro	Auto Scaling Group name	-
Auto-assigned IP address	-	VPC ID	vpc-03cdc7b6f19dd7211	Elastic IP addresses	-
IAM Role	<a href="#">sns_Dynamodb_role</a>	Subnet ID	subnet-0d9fa3144480cc9a9	AWS Compute Optimizer finding	<a href="#">Opt-in to AWS Compute Optimizer for recommendations.   Learn more</a>
IMDSv2	Required	Instance ARN	arn:aws:ec2:ap-south-1:557690616836:instance/i-001861022fbcac290	Auto Scaling Group name	-

[Connect](#) [Actions ▾](#)

- [Connect](#)
- [Manage instance state](#)
- [Instance settings](#)
- [Networking](#)
- Security**
- [Change security groups](#)
- [Get Windows password](#)
- [Modify IAM role](#)
- [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.
<input style="width: 100%; height: 30px; margin-bottom: 5px; border: 1px solid #ccc; padding: 5px; font-size: 14px; border-radius: 5px;" type="text" value="sns_Dynamodb_role"/> <span style="font-size: 14px; border: 1px solid #ccc; padding: 2px 10px; border-radius: 5px; color: #333;">▼</span> <span style="font-size: 14px; border: 1px solid #ccc; padding: 2px 10px; border-radius: 5px; color: #333;">C</span> <span style="font-size: 14px; border: 1px solid #ccc; padding: 2px 10px; border-radius: 5px; color: #333;">Create new IAM role</span>	

[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

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

Connection Type

[Connect using EC2 Instance Connect](#)

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

[Public IPv4 address](#)

[13.200.229.59](#)

[IPv6 address](#)

—

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.

ec2-user X

i **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.

[Cancel](#)

[Connect](#)

```
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
Last login: Tue Oct 15 04:17:59 2024 from 13.233.177.3
[ec2-user@ip-172-31-3-5 ~]$ █
```

i-001861022fbcac290 (InstantLibraryApp)

PublicIPs: 13.201.74.42 PrivateIPs: 172.31.3.5

## Milestone 6: Deployment on EC2

Deployment on an EC2 instance involves launching a server, configuring security groups for public access, and uploading your application files. After setting up necessary dependencies and environment variables, start your application and ensure it's running on the correct port. Finally, bind your domain or use the public IP to make the application accessible online.

### 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

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

- Run: '[git clone:https://github.com/kilarukusuma-12/Homemadepicklesandsnacks.git](https://github.com/kilarukusuma-12/Homemadepicklesandsnacks.git)'
- Note: change your-github-username and your-repository-name with your credentials here: '<https://github.com/kilarukusuma-12/Homemadepicklesandsnacks.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

```
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
'   _#_#
  _\###_
  \###|
  \#/  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 ~]$ git clone https://github.com/AlekhyaPembakula/InstantLibrary.git
fatal: destination path 'InstantLibrary' already exists and is not an empty directory.
[ec2-user@ip-172-31-3-5 ~]$ cd InstantLibrary
[ec2-user@ip-172-31-3-5 InstantLibrary]$ cd InstantLibrary
[ec2-user@ip-172-31-3-5 InstantLibrary]$ flask run --host=0.0.0.0 --port=80
 * Debug mode: off
Permission denied
[ec2-user@ip-172-31-3-5 InstantLibrary]$ ^C
[ec2-user@ip-172-31-3-5 InstantLibrary]$ ^C
[ec2-user@ip-172-31-3-5 InstantLibrary]$ sudo flask run --host=0.0.0.0 --port=80
 * Debug mode: off
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://127.0.0.1:80
 * Running on http://172.31.3.5:80
Press CTRL+C to quit
^C[ec2-user@ip-172-31-3-5 InstantLibrary]$
[ec2-user@ip-172-31-3-5 InstantLibrary]$ sudo flask run --host=0.0.0.0 --port=80
 * Debug mode: off
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://127.0.0.1:80
 * Running on http://172.31.3.5:80
Press CTRL+C to quit
183.82.125.56 - - [22/Oct/2024 07:42:00] "GET / HTTP/1.1" 302 -
183.82.125.56 - - [22/Oct/2024 07:42:01] "GET /register HTTP/1.1" 200 -
183.82.125.56 - - [22/Oct/2024 07:42:01] "GET /static/images/library3.jpg HTTP/1.1" 200 -
183.82.125.56 - - [22/Oct/2024 07:42:01] "GET /favicon.ico HTTP/1.1" 404 -
183.82.125.56 - - [22/Oct/2024 07:42:16] "GET /login HTTP/1.1" 200 -
183.82.125.56 - - [22/Oct/2024 07:42:16] "GET /static/images/library3.jpg HTTP/1.1" 304 -
183.82.125.56 - - [22/Oct/2024 07:42:21] "POST /login HTTP/1.1" 200 -
183.82.125.56 - - [22/Oct/2024 07:42:24] "GET /login HTTP/1.1" 200 -
183.82.125.56 - - [22/Oct/2024 07:42:27] "POST /login HTTP/1.1" 302 -
183.82.125.56 - - [22/Oct/2024 07:42:28] "GET /home-page HTTP/1.1" 200 -

```

i-001861022fbcac290 (InstantLibraryApp)  
PublicIPs: 13.201.74.42 PrivateIPs: 172.31.3.5

Verify the Flask app is running:

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

- Run the Flask app on the EC2 instance

```
C:\Users\kmoun\OneDrive\Desktop\Home made pickles and snacks>python 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://192.168.29.132:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 895-566-049
127.0.0.1 - - [15/Jul/2025 21:39:41] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [15/Jul/2025 21:39:42] "GET /static/css/style.css HTTP/1.1" 200 -
127.0.0.1 - - [15/Jul/2025 21:39:42] "GET /static/images/chiken.png HTTP/1.1" 304 -
127.0.0.1 - - [15/Jul/2025 21:39:42] "GET /static/images/bg.png HTTP/1.1" 304 -
127.0.0.1 - - [15/Jul/2025 21:39:42] "GET /static/images/mango.jpg HTTP/1.1" 304 -
127.0.0.1 - - [15/Jul/2025 21:39:42] "GET /static/images/snackes.jpg HTTP/1.1" 304 -
127.0.0.1 - - [15/Jul/2025 21:39:42] "GET /favicon.ico HTTP/1.1" 404 -
127.0.0.1 - - [15/Jul/2025 21:39:43] "GET /non_veg_pickles HTTP/1.1" 200 -
127.0.0.1 - - [15/Jul/2025 21:39:43] "GET /static/css/style.css HTTP/1.1" 304 -
127.0.0.1 - - [15/Jul/2025 21:39:43] "GET /static/images/chiken.png HTTP/1.1" 304 -
127.0.0.1 - - [15/Jul/2025 21:39:43] "GET /static/images/fish.png HTTP/1.1" 304 -
127.0.0.1 - - [15/Jul/2025 21:39:44] "GET /static/images/prawns.jpg HTTP/1.1" 304 -
127.0.0.1 - - [15/Jul/2025 21:39:44] "GET /static/images/crab.jpg HTTP/1.1" 304 -
127.0.0.1 - - [15/Jul/2025 21:39:44] "GET /static/images/mutton.jpg HTTP/1.1" 304 -
```

Access the website through:

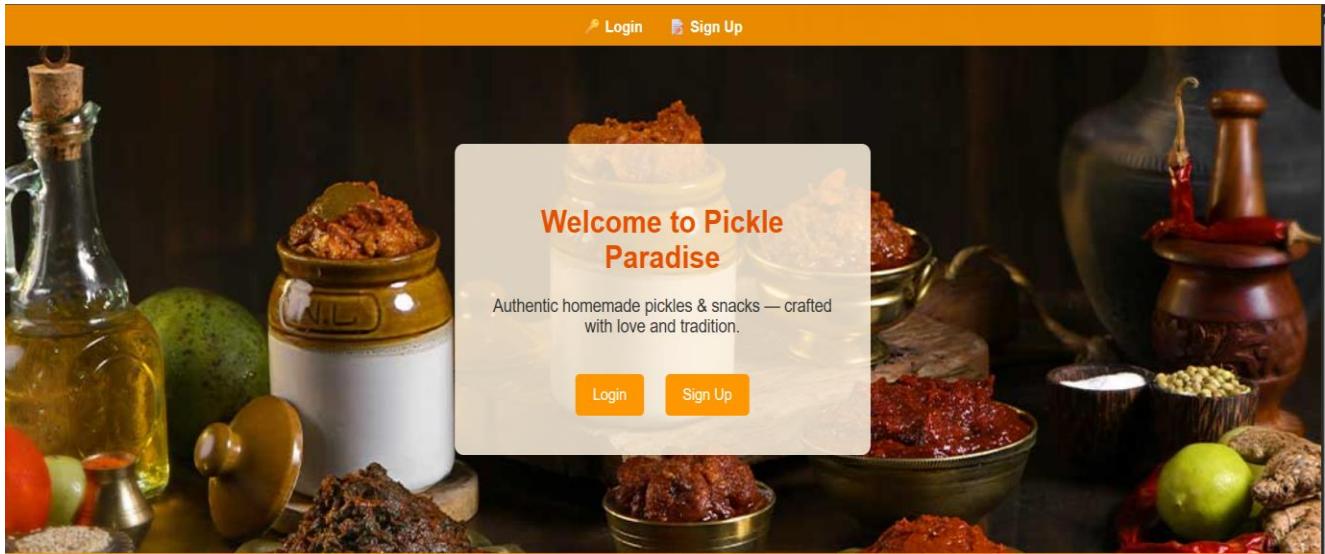
PublicIPs: <http://3.95.250.6:5000>

## 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 in 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:



Signup page

The image shows the "Sign Up" page of the "Pickle Paradise" website. The background is a light beige color. In the center, there is a white rectangular form with the following elements:

**Sign Up**

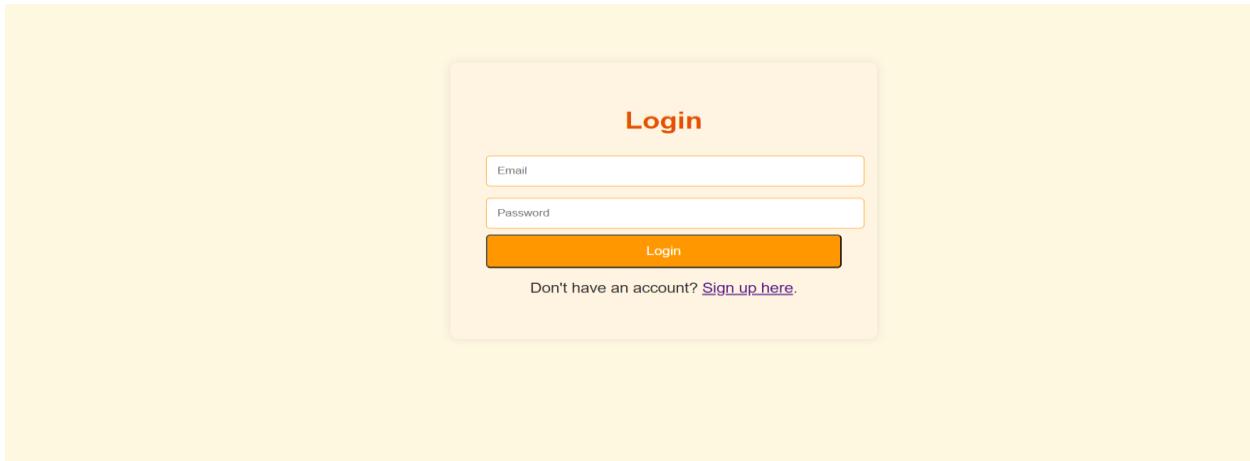
Email

Password

[Sign Up](#)

Already have an account? [Login here.](#)

Login Page:



Home Page:

The home page banner features a dark background with various ingredients like a lime, an orange slice, basil leaves, and nuts. Overlaid on the banner is a central text box containing the title "Welcome to Pickle Paradise" in white. Below the banner, there are three main product categories displayed in separate boxes: "Mango Pickle", "Chicken Pickle", and "Snacks". Each category includes a small image, a title, a brief description, and a "Explore More" link.

- Mango Pickle**  
Traditional raw mango chunks, spicy & tangy.  
[Explore More](#)
- Chicken Pickle**  
Spicy homemade Andhra chicken pickle.  
[Explore More](#)
- Snacks**  
Crispy mixture, murukku. Perfect to munch anytime, anywhere.  
[Explore More](#)

## Veg Pickles Page:

Home   Veg Pickles   Non-Veg Pickles   Snacks   Cart

### Veg Pickles



#### Mango Pickle

Classic raw mango chunks with bold spices. A tangy taste that brings back tradition.

₹250 | 500g

Rating: ★★★★★ 5/5

Add to Cart



#### Lemon Pickle

Zesty lemons cured in salt and spices. Perfect balance of sour and mild heat.

₹220 | 500g

Rating: ★★★★★ 4/5

Add to Cart



#### Tomato Pickle

Juicy ripe tomatoes with fiery masala. A spicy, tangy favorite for every meal.

₹199 | 500g

Rating: ★★★★★ 5/5

Add to Cart



#### Amla Pickle

Sour gooseberries pickled for a punch. Rich in flavor and healthy goodness.

₹180 | 500g

Rating: ★★★★★ 4/5

Add to Cart



#### Mixed Veg Pickle

Seasonal veggies spiced and preserved at home. Crunchy bites bursting with flavor.

₹199 | 500g

Rating: ★★★★★ 4/5

Add to Cart



#### Red Chilli Pickle

Bright red chillies packed with tangy spice. A bold pickle that kicks up any meal.

₹180 | 500g

Rating: ★★★★★ 5/5

Add to Cart



#### Garlic Pickle

Bold garlic cloves soaked in tangy oil. Strong flavor, pure homemade taste.

₹199 | 500g

Rating: ★★★★★ 5/5

Add to Cart



#### Gongura Pickle

Sour gongura leaves pickled Andhra style. Authentic regional taste in a jar.

₹150 | 500g

Rating: ★★★★★ 5/5

Add to Cart

**HomeMade Pickles & Snacks**  
Preserving traditions, one jar at a time. Authentic  
homemade pickles and snacks delivered fresh  
to your doorstep.

**Quick Links**  
[Home](#)  
[About Us](#)  
[Products](#)  
[Contact](#)

**Categories**  
[Vegetarian Pickles](#)  
[Non-Veg Pickles](#)  
[Traditional Snacks](#)

**Contact Info**  
 +91 9876543210  
 [Info@homemadepickles.com](mailto:info@homemadepickles.com)  
 Traditional Kitchen, India

© 2025 HomeMade Pickles & Snacks. All rights reserved.

## Non\_veg Pickles Page:

Screenshot of the Non-Veg Pickles page from the HomeMade Pickles & Snacks website.

The page features a header with navigation links: Home, Veg Pickles, Non-Veg Pickles, Snacks, and Cart.

### Non-Veg Pickles

The main content displays six non-veg pickle products in a grid format:

- Prawn Pickle**: Prawns cooked with fiery Andhra masala. Perfect spicy pickle for seafood fans. ₹599 | 500g. Rating: ★★★★★ 5/5. Add to Cart.
- Chicken Pickle**: Tender chicken bits coated in spicy oil. A fiery treat for meat pickle lovers. ₹450 | 500g. Rating: ★★★★★ 4/5. Add to Cart.
- Fish Pickle**: Soft boneless fish mixed with coastal spices. A tangy seafood pickle for every bite. ₹599 | 500g. Rating: ★★★★★ 4/5. Add to Cart.
- Mutton Pickle**: Juicy mutton pieces with rich spices. Bold taste that goes great with hot rice. ₹650 | 500g. Rating: ★★★★★ 4/5. Add to Cart.
- Crab Pickle**: Fresh crab meat in aromatic homemade masala. Adds a coastal punch to plain meals. ₹699 | 500g. Rating: ★★★★★ 5/5. Add to Cart.
- Gongura Mutton Pickle**: Juicy mutton cooked with tangy gongura. A bold Andhra pickle with spicy twist. ₹750 | 500g. Rating: ★★★★★ 5/5. Add to Cart.

The footer contains sections for HomeMade Pickles & Snacks, Quick Links, Categories, and Contact Info.

HomeMade Pickles & Snacks  
Preserving traditions, one jar at a time. Authentic homemade pickles and snacks delivered fresh to your doorstep.

Quick Links: Home, About Us, Products, Contact.

Categories: Vegetarian Pickles, Non-Veg Pickles, Traditional Snacks.

Contact Info: +91 9876543210, info@homemadepickles.com, Traditional Kitchen, India.

© 2025 HomeMade Pickles & Snacks. All rights reserved.

## Snacks Page:

Home Veg Pickles Non-Veg Pickles Snacks Cart (2)

### Snacks



#### Mixture

Homemade crunchy mix of sev and nuts. Tasty snack for tea or travel.

₹99 | 250g

Rating: ★★★★★ 5/5

Add to Cart



#### Murukku

Crispy rice flour spirals made by hand. Light, crunchy and always fresh.

₹150 | 250g

Rating: ★★★★★ 4/5

Add to Cart



#### Banana Chips

Thin banana slices fried in fresh oil. Lightly salted and perfectly crunchy.

₹99 | 250g

Rating: ★★★★★ 4/5

Add to Cart



#### Ribbon Pakoda

Flat, spicy ribbon-shaped crispy sticks. A crunchy snack with a mild kick.

₹120 | 250g

Rating: ★★★★★ 5/5

Add to Cart



#### Kara Sev

Spicy chickpea flour sev made fresh. Crispy snack with a subtle heat.

₹80 | 250g

Rating: ★★★★★ 4/5

Add to Cart



#### Thattai

Crispy rice flour crackers with mild spices. Perfect crunchy snack for any time.

₹80 | 200g

Rating: ★★★★★ 5/5

Add to Cart

**HomeMade Pickles & Snacks**  
Preserving traditions, one jar at a time. Authentic homemade pickles and snacks delivered fresh to your doorstep.

**Quick Links**  
[Home](#) [About Us](#) [Products](#) [Contact](#)

**Categories**  
[Vegetarian Pickles](#) [Non-Veg Pickles](#) [Traditional Snacks](#)

**Contact Info**  
 +91 9876543210  
 [info@homemadepickles.com](mailto:info@homemadepickles.com)  
 Traditional Kitchen, India

© 2025 HomeMade Pickles & Snacks. All rights reserved

## Cart Page:

The screenshot shows a shopping cart page with the following details:

Item	Price	Weight	Quantity	Actions
Crab Pickle	₹699.0	500g	1	+ - Remove
Chicken Pickle	₹450.0	500g	1	+ - Remove
Murukku	₹150.0	250g	1	+ - Remove
Mixture	₹99.0	250g	1	+ - Remove

Total: ₹1398.0

Proceed to Checkout

**HomeMade Pickles & Snacks**  
Preserving traditions, one jar at a time. Authentic homemade pickles and snacks delivered fresh to your doorstep.

**Quick Links**  
Home | About Us | Products | Contact

**Categories**  
Vegetarian Pickles | Non-Veg Pickles | Traditional Snacks

**Contact Info**  
+91 9876543210 | info@homemadepickles.com | Traditional Kitchen, India

## Checkout Page:

The screenshot shows a checkout page with the following fields:

- Full Name
- Email Address
- Street Address
- City
- Pincode
- Phone Number

Select Payment Method:  
-Select-

Place Order

## Contact Us Page:

The screenshot shows the 'Contact Us' page of a website. At the top, there is a navigation bar with links for Home, Veg Pickles, Non-Veg Pickles, Snacks, Cart, and Contact Us. The main content area has a light yellow background and features a central box titled 'Contact Us'. Inside this box, there is a message: 'If you have any questions or feedback, reach out anytime!'. Below the message are two input fields: 'Your Name' and 'Your Message'. At the bottom of the box is a large orange button labeled 'Send Message'. At the very bottom of the page, there is a footer section with four columns: 'HomeMade Pickles & Snacks' (describing the business), 'Quick Links' (with links to Home, About Us, Products, and Contact), 'Categories' (listing Vegetarian Pickles, Non-Veg Pickles, and Traditional Snacks), and 'Contact Info' (with a phone number and email address).

<b>HomeMade Pickles &amp; Snacks</b> Preserving traditions, one jar at a time. Authentic homemade pickles and snacks delivered fresh	<b>Quick Links</b> Home About Us Products Contact	<b>Categories</b> Vegetarian Pickles Non-Veg Pickles Traditional Snacks	<b>Contact Info</b> +91 9876543210 info@homemadepickles.com
---	---	--	---

## About Us Page:

The screenshot shows the 'About Us' page of the website. At the top, there is a navigation bar with links for Home, Veg Pickles, Non-Veg Pickles, Snacks, Cart, and Contact Us. The main content area has a light yellow background and features a central box titled 'About Pickle Paradise'. Inside this box, there is a welcome message: 'Welcome to Pickle Paradise — your one-stop shop for homemade pickles and crunchy snacks! We craft our pickles with age-old recipes, using farm-fresh ingredients and traditional sun-curing methods to bring authentic taste straight to your table.' Below this is another paragraph: 'Our mission is to keep grandma's recipes alive, blending spices and love in every jar. Whether it's tangy mango pickle, fiery non-veg delicacies, or crunchy snacks, we make sure every bite makes you feel at home.' At the bottom of the box is a thank you message: 'Thank you for choosing Pickle Paradise. We're thrilled to be part of your meal, one jar at a time!' At the very bottom of the page, there is a footer section with four columns: 'HomeMade Pickles & Snacks' (describing the business), 'Quick Links' (with links to Home, About Us, Products, and Contact), 'Categories' (listing Vegetarian Pickles, Non-Veg Pickles, and Traditional Snacks), and 'Contact Info' (with a phone number and email address).

<b>HomeMade Pickles &amp; Snacks</b> Preserving traditions, one jar at a time. Authentic homemade pickles and snacks delivered fresh	<b>Quick Links</b> Home About Us Products Contact	<b>Categories</b> Vegetarian Pickles Non-Veg Pickles Traditional Snacks	<b>Contact Info</b> +91 9876543210 info@homemadepickles.com
---	---	--	---

After Logout

