

# Homemade Pickles & Snacks: Taste The Best

## Project Description:

This project is a user-friendly website designed to showcase and sell a variety of homemade pickles and snacks. It features rich visuals, engaging descriptions, and an easy-to-navigate layout that highlights the authenticity and flavour of each product. Customers can explore product categories like spicy pickles, sweet preserves, and crunchy munchies, all made with traditional recipes. The site includes a secure shopping cart system and responsive design for mobile accessibility. A blog section shares homemade tips, recipe stories, and cultural food traditions to deepen customer connection. Users can rate products, leave reviews, and sign up for seasonal bundle offers. SEO-optimized content boosts discoverability across search engines. It's a Flavourful blend of heritage, tech, and convenience. If you're planning to build or document this site further, I can help structure sections like product listings, customer experience features, or a compelling homepage narrative.

## Scenario 1: Efficient Order Placement by Customers

In the "Flavour Cart" homemade pickles and snacks website, AWS EC2 powers a reliable and scalable infrastructure, allowing multiple customers to browse and order products simultaneously. For example, a user visits the site, explores the pickles section, and places an order for a mango pickle jar. Flask handles the backend workflow, processing user selections, updating inventory, and confirming orders in real time. The cloud-based architecture ensures a seamless and fast shopping experience, even during peak demand like festivals or seasonal sales.

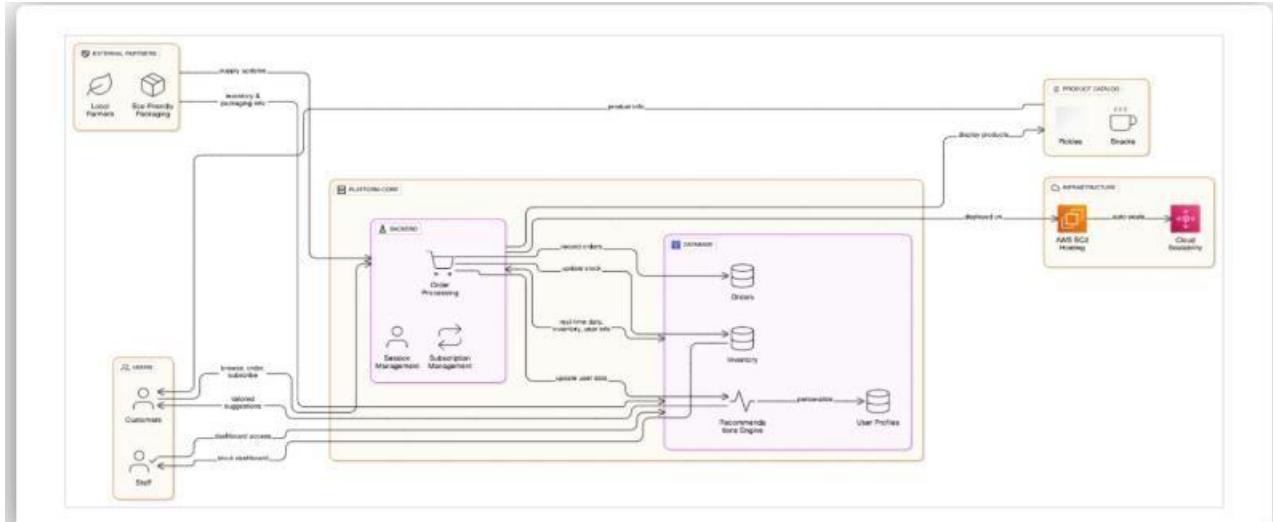
## Scenario 2: Seamless Order Notifications for Kitchen Staff and Customers

When an order is placed, the system uses AWS SNS to instantly notify both the customer and the in-house kitchen team. Flask captures the order details and triggers an SNS email or SMS alert: the customer receives a confirmation with order tracking info, while the kitchen staff is alerted with the order details to initiate preparation. All order information is securely stored in AWS DynamoDB, ensuring timely fulfilment and status monitoring.

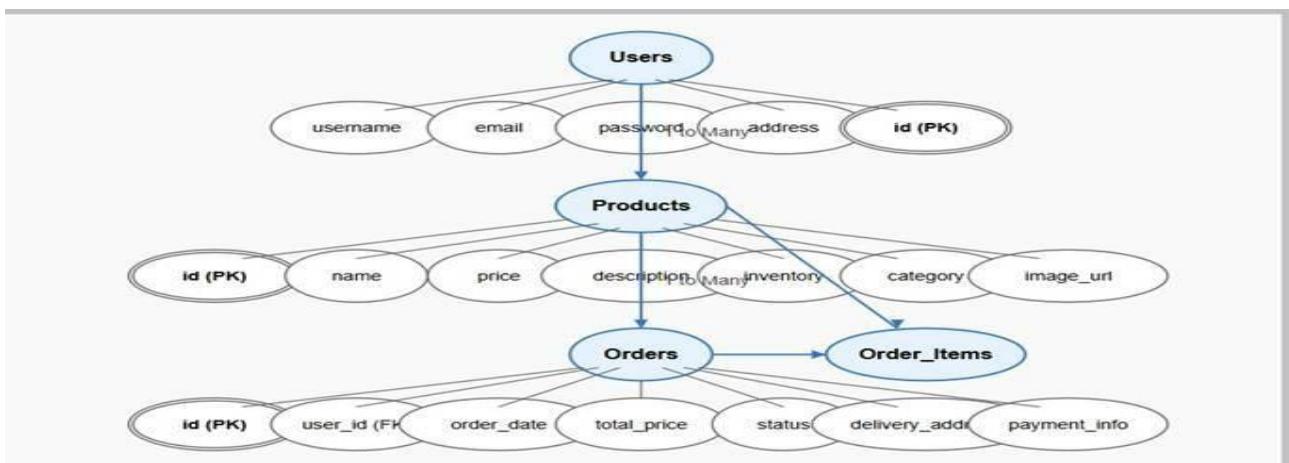
## Scenario 3: Real-Time Inventory and Product Availability

The website allows users to explore the entire product range—spicy pickles, sweet treats, crunchy snacks—with up-to-date availability status. Flask fetches real-time data from DynamoDB, showing users whether an item is in stock or on backorder. Thanks to EC2, the site delivers a consistently smooth and responsive experience, even when accessed by many users across mobile or desktop platforms, ensuring customers always have access to the latest listings.

## AWS ARCHITECTURE



Entity Relationship (ER)Diagram:



### Pre-requisites:

1. **AWS Account Setup:** [AWS Account Setup](#)
2. **Understanding IAM:** [IAM Overview](#)
3. **Amazon EC2 Basics:** [EC2 Tutorial](#)
4. **DynamoDB Basics:** [DynamoDB Introduction](#)
5. **VS Code Installation:** [VISUALSTUDIO.COM](#)
6. **Git Version Control:** [Git Documentation](#)

## Project Work Flow:

### 1. AWS Account Setup and Login

**Activity 1.1:** Set up an AWS account if not already done.

**Activity 1.2:** Log in to the AWS Management Console.

### 2. DynamoDB Database Creation and Setup

**Activity 2.1:** Create a DynamoDB Table.

**Activity 2.2:** Configure Attributes for User Data and Book Requests

### 3. SNS Notification Setup

**Activity 3.1:** Create SNS topics for book request notifications.

**Activity 3.2:** Subscribe users and library staff to SNS email notifications.

### 4. Backend Development and Application Setup

**Activity 4.1:** Develop the Backend Using Flask..

**Activity 4.2:** Integrate AWS Services Using boto3.

### 5. IAM Role Setup

**Activity 5.1:** Create IAM Role

**Activity 5.2:** Attach Policies

### 6. EC2 Instance Setup

**Activity 6.1:** Launch an EC2 instance to host the Flask application.

**Activity 6.2:** Configure security groups for HTTP, and SSH access.

### 7. Deployment on EC2

**Activity 7.1:** Upload Flask Files

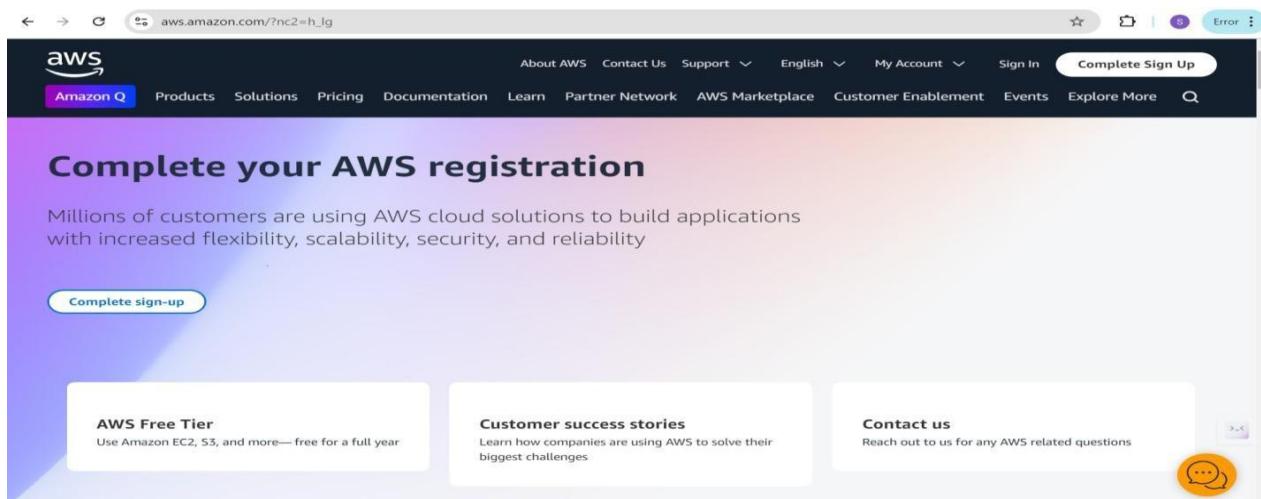
**Activity 7.2:** Run the Flask App

## 8. Testing and Deployment

**Activity 8.1: Conduct functional testing to verify user signup, login, orders and notifications.**

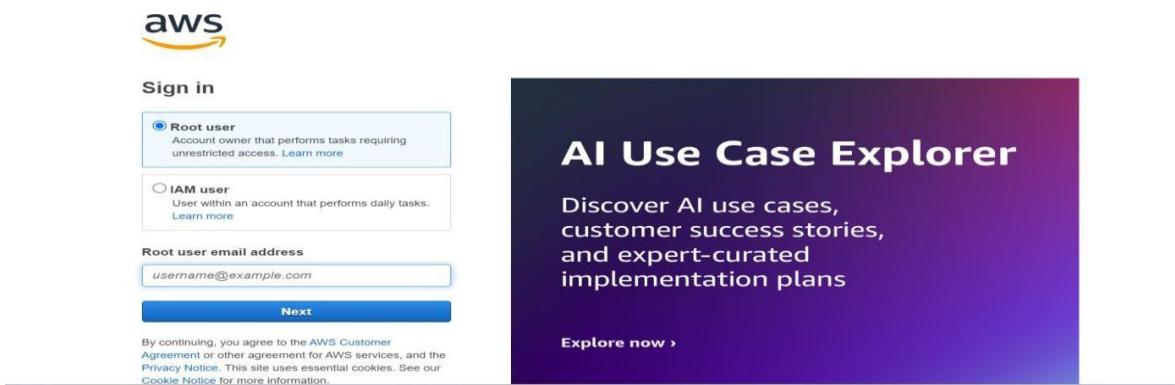
### Milestone 1: AWS Account Setup and Login

- **Activity 1.1: Set up an AWS account if not already done.**
  - Sign up for an AWS account and configure billing settings.



- **Activity 1.2: Log in to the AWS Management Console**

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

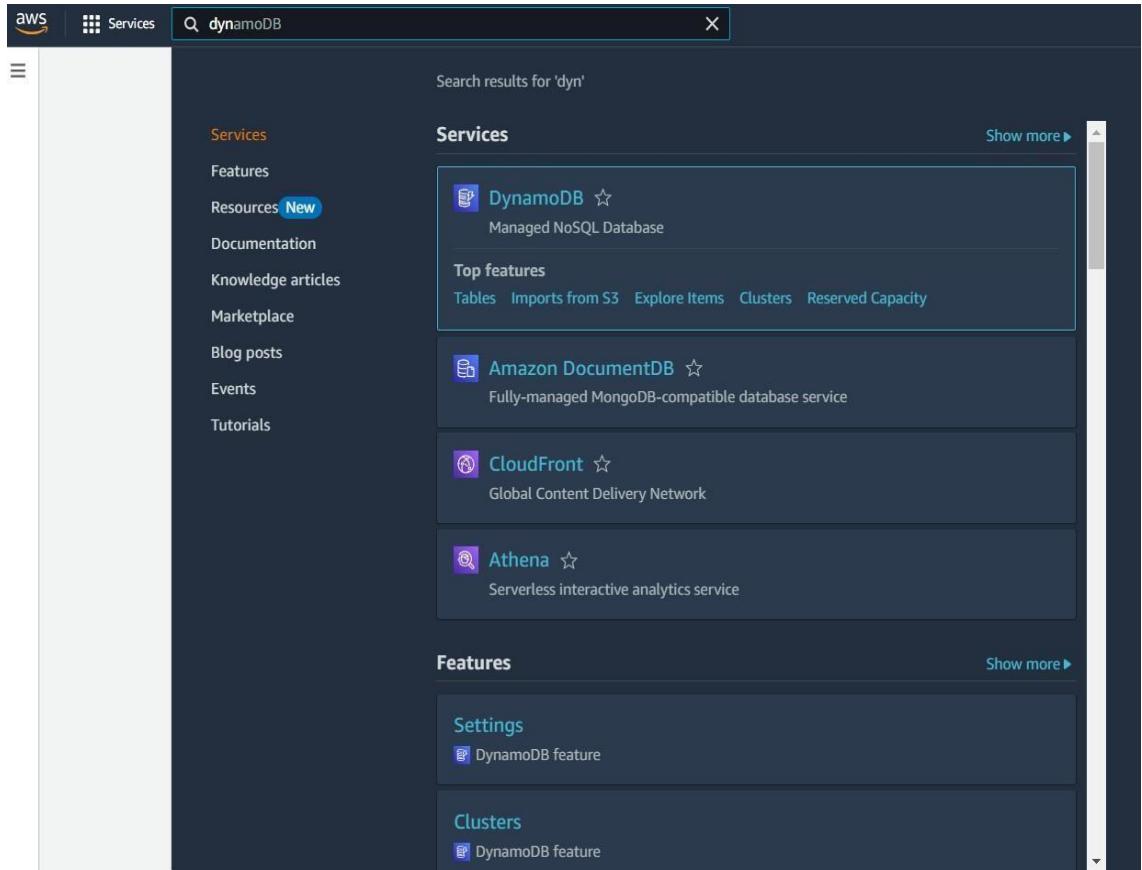


The image contains two parts. On the left is a screenshot of the AWS sign-in page. It features the AWS logo at the top, followed by a 'Sign in' section. It offers two options: 'Root user' (selected) and 'IAM user'. Below each option is a brief description. A field for 'Root user email address' is filled with 'username@example.com', and a 'Next' button is below it. At the bottom, there's a small note about cookie usage. On the right is a promotional card for the 'AI Use Case Explorer'. The card has a dark background with a purple-to-blue gradient. The title 'AI Use Case Explorer' is at the top in white. Below it, the text reads 'Discover AI use cases, customer success stories, and expert-curated implementation plans'. At the bottom, there's a 'Explore now >' button.

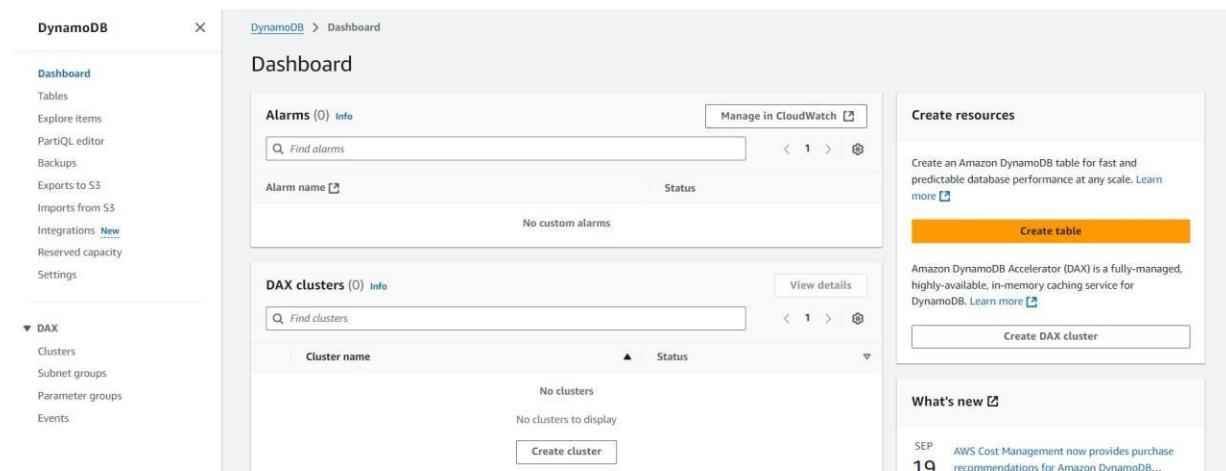
## Milestone 2: DynamoDB Database Creation and Setup

- **Activity 2.1: Navigate to the DynamoDB**

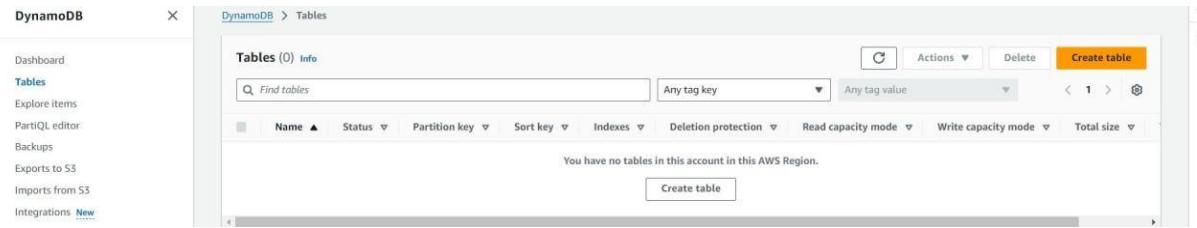
- In the AWS Console, navigate to DynamoDB and click on create tables.



The screenshot shows the AWS Services search results page. The search bar at the top contains the text 'dynamoDB'. The results are categorized under 'Services' and 'Features'. Under 'Services', 'DynamoDB' is listed as a Managed NoSQL Database. Other services listed include Amazon DocumentDB, CloudFront, and Athena. Under 'Features', there are sections for 'Settings' and 'Clusters', both of which mention 'DynamoDB feature'. A 'Show more ▶' link is visible at the top right of each category section.



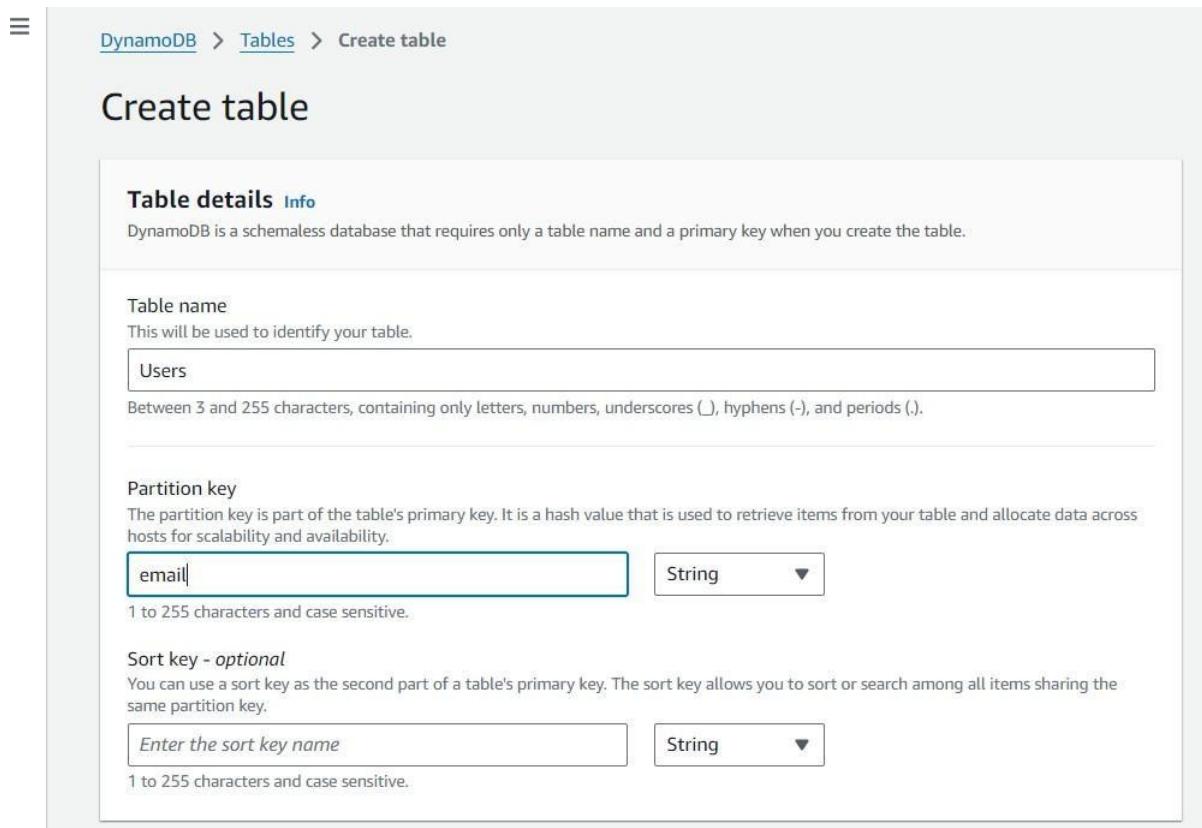
The screenshot shows the DynamoDB Dashboard. On the left, there is a sidebar with links for 'Dashboard', 'Tables', 'Explore items', 'PartiQL editor', 'Backups', 'Exports to S3', 'Imports from S3', 'Integrations New', 'Reserved capacity', 'Settings', 'DAX', 'Clusters', 'Subnet groups', 'Parameter groups', and 'Events'. The main dashboard area has two sections: 'Alarms (0) Info' and 'DAX clusters (0) Info'. Both sections show a table with columns for 'Cluster name' and 'Status'. Below these sections are buttons for 'Manage in CloudWatch' and 'View details' for alarms, and 'Create cluster' for DAX clusters. To the right, there is a 'Create resources' section with a large orange 'Create table' button. Below it, there is information about Amazon DynamoDB Accelerator (DAX) and a 'Create DAX cluster' button. At the bottom, there is a 'What's new' section with a note about AWS Cost Management providing purchase recommendations for Amazon DynamoDB.



The screenshot shows the AWS DynamoDB 'Tables' page. On the left, there is a sidebar with options like 'Dashboard', 'Tables' (which is selected), 'Explore items', 'PartiQL editor', 'Backups', 'Exports to S3', 'Imports from S3', and 'Integrations'. The main area has a heading 'Tables (0) info' with a search bar and filters for 'Any tag key' and 'Any tag value'. A table header includes columns for 'Name', 'Status', 'Partition key', 'Sort key', 'Indexes', 'Deletion protection', 'Read capacity mode', 'Write capacity mode', and 'Total size'. Below the table, a message says 'You have no tables in this account in this AWS Region.' At the bottom right of the table area is a 'Create table' button.

- **Activity 2.2: Create a DynamoDB table for storing Signup details and pickle requests.**

- Create Users table with partition key "Email" with type String and click on create tables.



The screenshot shows the 'Create table' wizard. The top navigation bar shows 'DynamoDB > Tables > Create table'. The main title is 'Create table'. Under 'Table details', it says 'DynamoDB is a schemaless database that requires only a table name and a primary key when you create the table.' The 'Table name' field is set to 'Users'. Below it, a note says 'This will be used to identify your table.' The 'Partition key' section shows a field with 'email' and a type dropdown set to 'String'. A note below says 'The partition key is part of the table's primary key. It is a hash value that is used to retrieve items from your table and allocate data across hosts for scalability and availability.' The 'Sort key - optional' section shows a field with 'Enter the sort key name' and a type dropdown set to 'String'. A note below says '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.' Both fields have a note below them stating '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	
<input type="checkbox"/>	Users	Active	email (S)	-	0	Off	Provisioned (5)	Provisioned (5)	0 bytes	

- Follow the same steps to create a requests table with Email as the primary key for pickle order, contact requests data.

DynamoDB > Tables > Create table

## Create table

### Table details Info

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

#### Table name

This will be used to identify your table.

Between 3 and 255 characters, containing letters, numbers, and periods ..

#### Partition key

The partition key part of the table's primary key. It is a hash value that is used to retrieve items from your table hosts for scalability and availability.

String ▾

1 to 255 characters and case sensitive.

#### Sort key - optional

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

String ▾

1 to 255 characters and case sensitive.

**Default settings**

This option optimizes your table based on AWS

**Customize settings**

Use this option to specify or make DynamoDB

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.

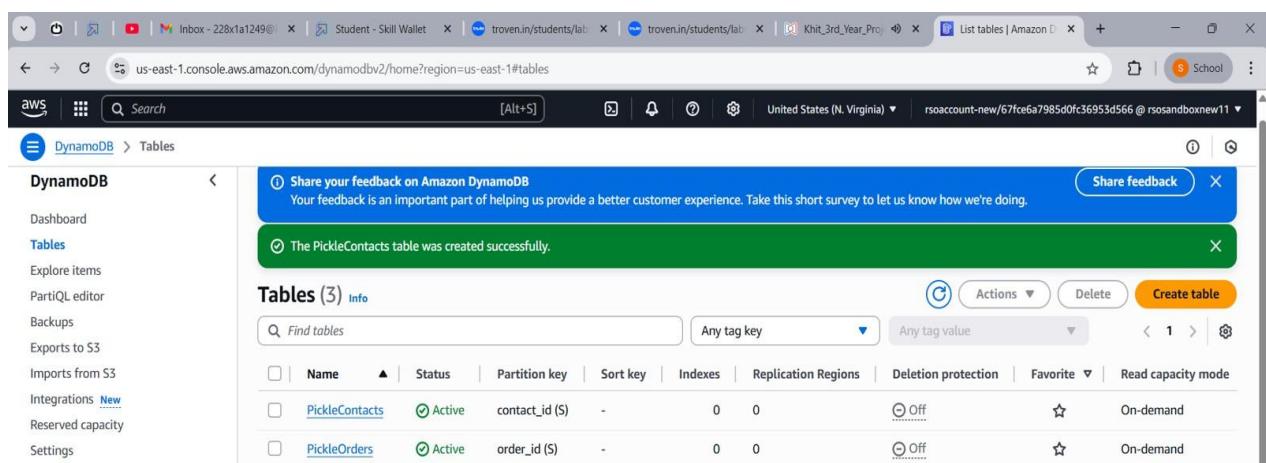
No tags are associated with the resource.

[Add new tag](#)

You can add 50 more tags.

[Cancel](#)

[Create table](#)



The screenshot shows the AWS DynamoDB console interface. The left sidebar has a navigation menu with options like Dashboard, Tables, Explore items, PartiQL editor, Backups, Exports to S3, Imports from S3, Integrations, and Settings. The main area is titled 'Tables (3)'. It lists two tables: 'PickleContacts' and 'PickleOrders'. Both tables are shown as active with 'On-demand' read capacity mode. A success message at the top right says 'The PickleContacts table was created successfully.' There is also a blue banner at the top asking for feedback on the service.

Name	Status	Partition key	Sort key	Indexes	Replication Regions	Deletion protection	Favorite	Read capacity mode
PickleContacts	Active	contact_id (\$)	-	0	0	Off	☆	On-demand
PickleOrders	Active	order_id (\$)	-	0	0	Off	☆	On-demand

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](#)
 ⓘ Share your feedback on Amazon DynamoDB

Your feedback is an important part of helping us provide a better customer experience. Take this short survey to let us know how we're doing.

[Share feedback](#)
 ⓘ The PickleContacts table was created successfully.

### Tables (3) [Info](#)


[Actions ▾](#)
[Delete](#)
[Create table](#)
 Find tables

Any tag key

Any tag value

&lt;

1

&gt;

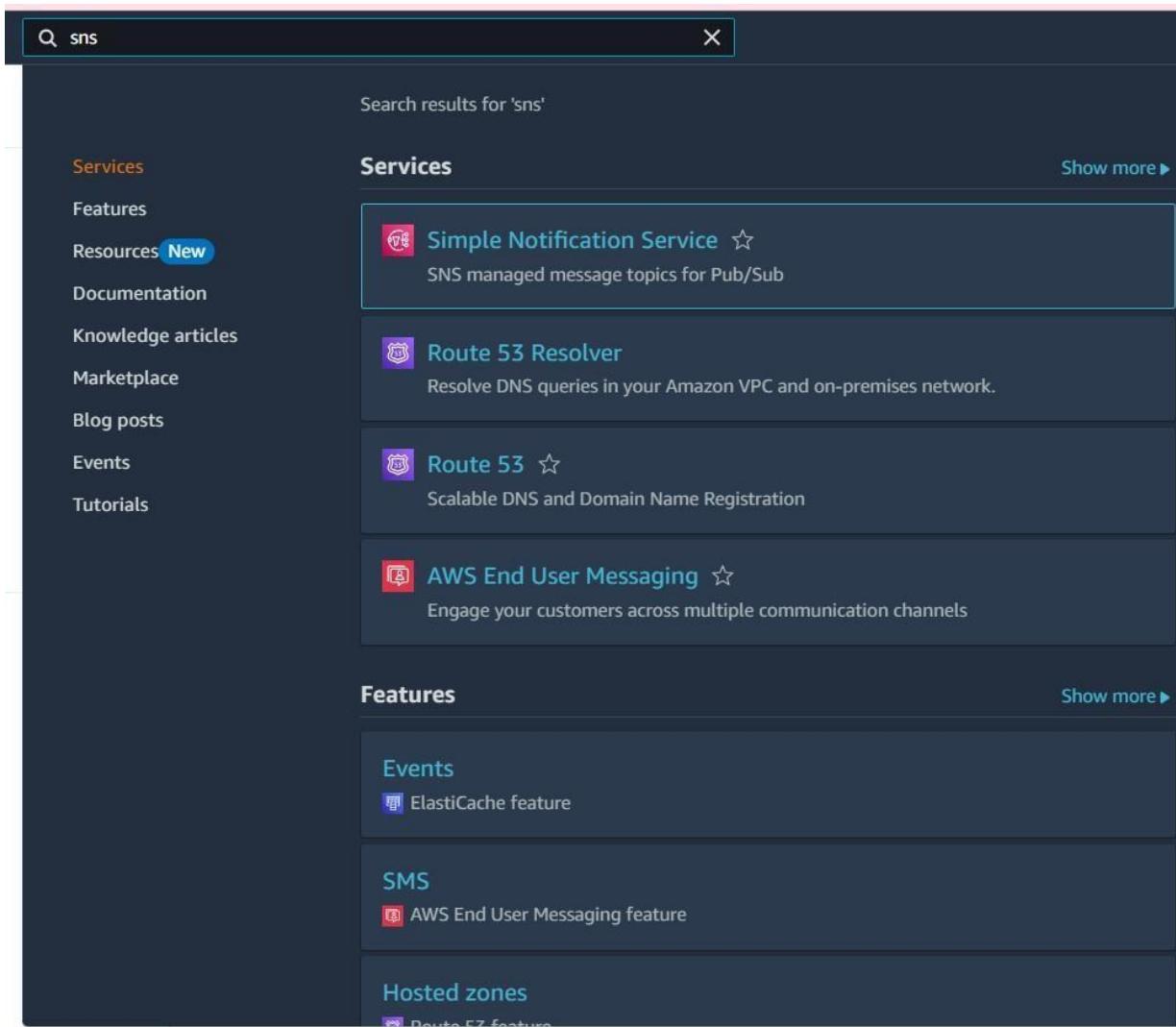


<input type="checkbox"/>	Name	Status	Partition key	Sort key	Indexes	Replication Regions	Deletion protection	Favorite	Read capacity mode
<input type="checkbox"/>	PickleContacts	Active	contact_id (\$)	-	0	0	Off	☆	On-demand
<input type="checkbox"/>	PickleOrders	Active	order_id (\$)	-	0	0	Off	☆	On-demand
<input type="checkbox"/>	PickleUsers	Active	email (\$)	-	0	0	Off	☆	On-demand

## Milestone 3: SNS Notification Setup

- **Activity 3.1: Create SNS topics for sending email notifications to users and library staff.**

In the AWS Console, search for SNS and navigate to the SNS Dashboard



The screenshot shows the AWS search results for 'sns'. The search bar at the top contains 'sns'. Below it, the results are displayed under the 'Services' category. The first result is 'Simple Notification Service' (marked with a star), described as 'SNS managed message topics for Pub/Sub'. The second result is 'Route 53 Resolver', described as 'Resolve DNS queries in your Amazon VPC and on-premises network.'. The third result is 'Route 53', described as 'Scalable DNS and Domain Name Registration'. The fourth result is 'AWS End User Messaging', described as 'Engage your customers across multiple communication channels'. Below the 'Services' section, there is a 'Features' section with three items: 'Events' (ElastiCache feature), 'SMS' (AWS End User Messaging feature), and 'Hosted zones' (Route 53 feature). A 'Show more ▶' link is located at the top right of the 'Services' section.

Amazon SNS

New Feature  
Amazon SNS now supports in-place message archiving and replay for FIFO topics. [Learn more](#)

Application Integration

## Amazon Simple Notification Service

Pub/sub messaging for microservices and serverless applications.

Amazon SNS is a highly available, durable, secure, fully managed pub/sub messaging service that enables you to decouple microservices, distributed systems, and event-driven serverless applications. Amazon SNS provides topics for high-throughput, push-based, many-to-many messaging.

**Create topic**

Topic name  
A topic is a message channel. When you publish a message to a topic, it fans out the message to all subscribed endpoints.

**Next step**

[Start with an overview](#)

**Pricing**

- Click on **Create Topic** and choose a name for the topic.

Amazon SNS

New Feature  
Amazon SNS now supports in-place message archiving and replay for FIFO topics. [Learn more](#)

Amazon SNS > Topics

Name	Type	ARN
No topics		
To get started, create a topic.		
<a href="#">Create topic</a>		

- Choose Standard type for general notification use cases and Click on Create Topic.

## Milestone 4:Backend Development and Application Setup

- **Activity 4.1: Develop the backend using Flask**

File Explorer Structure

**Description:** set up the INSTANT LIBRARY project with an app.py file, a static/ folder for images, and a templates/ directory containing all required HTML pages like home, login, signup, pickle-specific pages (e.g., Veg\_Pickles.html, Non\_Veg\_Pickles.html, Snacks.html), and utility pages (e.g., order.html, contact.html).

**Description of the code :**

- **Flask App Initialization**

```
from flask import Flask, request, render_template, redirect, url_for, session, flash, jsonify
import boto3
import uuid
import os
import hashlib
from datetime import datetime
from botocore.exceptions import ClientError
```

**Description:** import essential libraries including Flask utilities for routing, Boto3 for DynamoDB operations, SMTP and email modules for sending mails, and Bcrypt for password hashing and verification.

```
app = Flask(__name__)
```

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

- **Dynamodb Setup:**

```
# DynamoDB tables - Production ready
order_table = dynamodb.Table('PickleOrders')
user_table = dynamodb.Table('PickleUsers')
contact_table = dynamodb.Table('PickleContacts')
```

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

- **SNS Connection**

```
@app.route('/notify')
def notify():
    # Send a sample SNS message
    sns.publish(
        TopicArn=SNS_TOPIC_ARN,
        Message='A new order was received on Homemade Pickles & Snacks!',
        Subject='New Pickle Order Alert'
    )
    return "SNS notification sent!"
```

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

- Routes for Web Pages

- Home Route:

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

**Description:** define the home route / to automatically redirect users to the home page when they access the base URL.

- Signup Route :

```
@app.route('/signup', methods=['GET', 'POST'])
def signup():
    if request.method == 'POST':
        try:
            name = request.form['name']
            email = request.form['email']
            password = request.form['password']
            hashed_password = hash_password(password)
            timestamp = datetime.utcnow().isoformat()

            # Check if user already exists
            response = user_table.get_item(Key={'email': email})
            if 'Item' in response:
                flash('User already exists', 'error')
                return render_template('signup.html')

            # Save user to DynamoDB
            user_table.put_item(Item={
                'email': email,
                'name': name,
                'password': hashed_password,
                'created_at': timestamp,
                'status': 'active'
            })
```

```

# Send welcome email
welcome_message = f"Dear {name},\n\nWelcome to Homemade Pickles &
Snacks!\n\nYour account has been created successfully. You can now login and
start ordering our delicious homemade pickles and snacks.\n\nThank you for
joining us!\n\nBest regards,\nHomemade Pickles & Snacks Team"
send_email_notification(email, 'Welcome to Homemade Pickles & Snacks!', 
welcome_message)

flash('Account created successfully! Please login.', 'success')
return redirect(url_for('index'))
except Exception as e:
    flash(f'Signup failed: {str(e)}', 'error')
return render_template('signup.html')

```

**Description:** define signup route to validate registration form fields, hash the user password using Bcrypt, store the new user in DynamoDB with a login count, and send an SNS notification on successful registration

- **login Route (GET/POST):**

```

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

            # Check user in DynamoDB
            response = user_table.get_item(Key={'email': email})
            if 'Item' in response:
                stored_password = response['Item'].get('password')
                if stored_password == hashed_password:
                    session['user_email'] = email
                    session['user_name'] = response['Item'].get('name')
                    return redirect(url_for('home'))

                flash('Invalid email or password', 'error')
            except Exception as e:
                flash(f'Login failed: {str(e)}', 'error')
        return render_template('login.html')

```

**Description:** define login route to validate user credentials against DynamoDB, check the password using encryption, update the login count on successful authentication, and redirect users to the home page.

- **Index Route :**

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

**Description:** The index page serves as the homepage, welcoming users with an overview of your brand and guiding them to key sections like products, contact, or login.

- **Order route:**

```
@app.route('/order', methods=['GET', 'POST'])
def order():
    if request.method == 'POST':
        try:
            name = request.form['name']
            email = request.form['email']
            phone = request.form['phone']
            address = request.form['address']
            city = request.form['city']
            pincode = request.form['pincode']
            item = request.form['item']
            quantity = int(request.form['quantity'])
            notes = request.form.get('notes', '')
            order_id = str(uuid.uuid4())
            timestamp = datetime.utcnow().isoformat()

            # Save to DynamoDB with full order details
            order_table.put_item(Item={
                'order_id': order_id,
                'name': name,
                'email': email,
                'phone': phone,
                'address': address,
                'city': city,
                'pincode': pincode,
                'item': item,
                'quantity': quantity,
```

```

def order():
    'pincode': pincode,
    'item': item,
    'quantity': quantity,
    'notes': notes,
    'timestamp': timestamp,
    'status': 'pending',
    'total_amount': quantity * 100 # Sample pricing
}

# Send email notifications
customer_message = f"Dear {name},\n\nYour order has been placed
successfully!\n\nOrder ID: {order_id}\nItem: {item}\nQuantity: {quantity}
\n\nWe'll contact you soon for delivery details.\n\nThank you for choosing
Homemade Pickles & Snacks!"
admin_message = f"New Order Received!\n\nOrder ID: {order_id}\nCustomer:
{name}\nEmail: {email}\nPhone: {phone}\nItem: {item}\nQuantity: {quantity}
\nAddress: {address}, {city} - {pincode}\nNotes: {notes}"

send_email_notification(email, 'Order Confirmation - Homemade Pickles &
Snacks', customer_message)
send_email_notification(ADMIN_EMAIL, f'New Order - {order_id}', admin_message)

session['last_order_id'] = order_id
return redirect(url_for('success'))
except Exception as e:
    flash(f'Order processing failed: {str(e)}', 'error')
    return render_template('order.html')
return render_template('order.html')

```

**Description:** The order page allows customers to review their selected items, enter shipping details, and confirm payment to complete the purchase. It acts as the final step in the checkout process, ensuring a smooth and secure transaction.

- Aws-info Route:

```
@app.route('/aws-info')
def aws_info():
    """Display AWS service information"""
    try:
        # Get EC2 instance info
        instance_id = get_instance_info()

        # Get IAM role info
        try:
            sts = boto3.client('sts', region_name='us-east-1')
            identity = sts.get_caller_identity()
            account_id = identity['Account']
            role_arn = identity.get('Arn', 'No role attached')
        except:
            account_id = 'Unknown'
            role_arn = 'No role attached'

        info = {
            'instance_id': instance_id,
            'account_id': account_id,
            'role_arn': role_arn,
            'region': 'us-east-1'
        }
        return jsonify(info)
    except Exception as e:
        return jsonify({'error': str(e)}), 500
```

**Description:** The Aws-info route displays details about your website's integration with AWS, such as database status, storage usage, or service configurations. It helps admins monitor backend health and ensure smooth operation of the homemade pickles and snacks platform.

- **About route:**

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

**Description:** The about route displays a heartfelt introduction to your homemade pickles and snacks brand, sharing its origin, mission, and values. It builds trust and connection by telling your story and celebrating what makes your products unique.

- **Success route:**

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

**Description:** The success route displays a confirmation message after a successful order or form submission, reassuring users that their action was completed. It adds a friendly closing touch to the user journey with gratitude or next-step instructions.

- **Cart route:**

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

**Description:** The cart route displays all items a user has added for purchase, allowing them to review quantities and total cost before checkout. It acts as a shopping basket, giving users the option to update or remove products before finalizing their order

- **Logout route:**

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

**Description:** The logout route securely ends the user's session, logging them out of their account. It helps protect personal information and ensures a fresh login is required for future activity.

- **Deployment Code:**

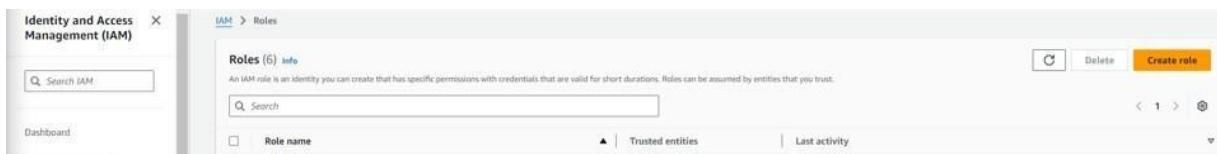
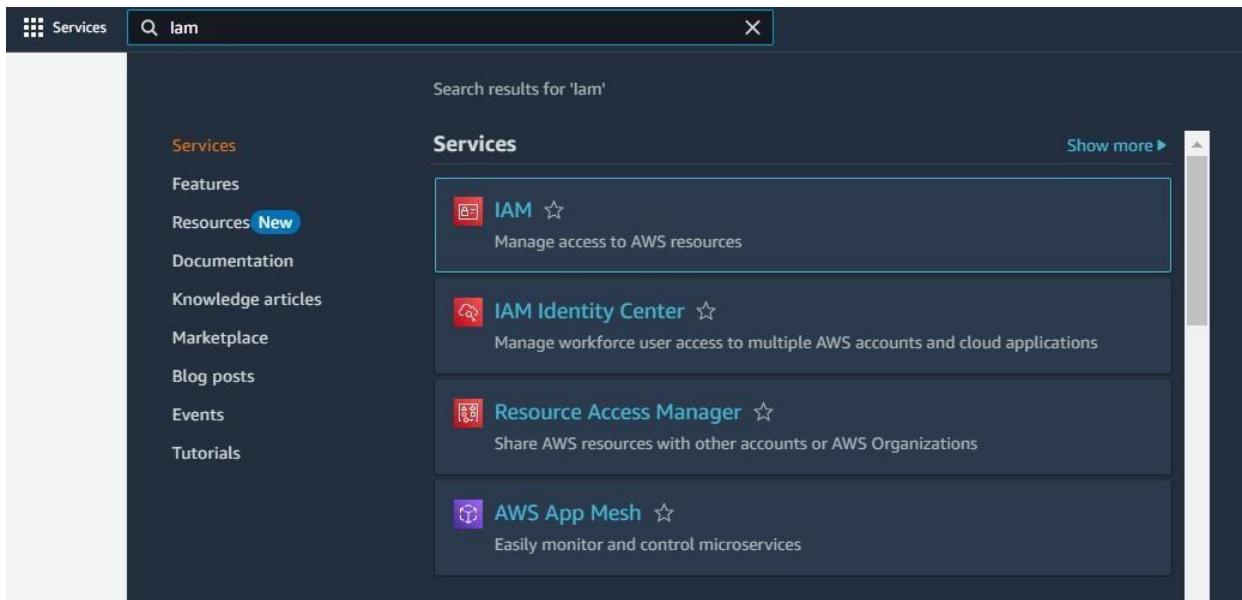
```
if __name__ == '__main__':
    port = int(os.environ.get('PORT', 5000))
    debug = os.environ.get('DEBUG', 'False').lower() == 'true'
    app.run(host='0.0.0.0', port=port, debug=debug)
```

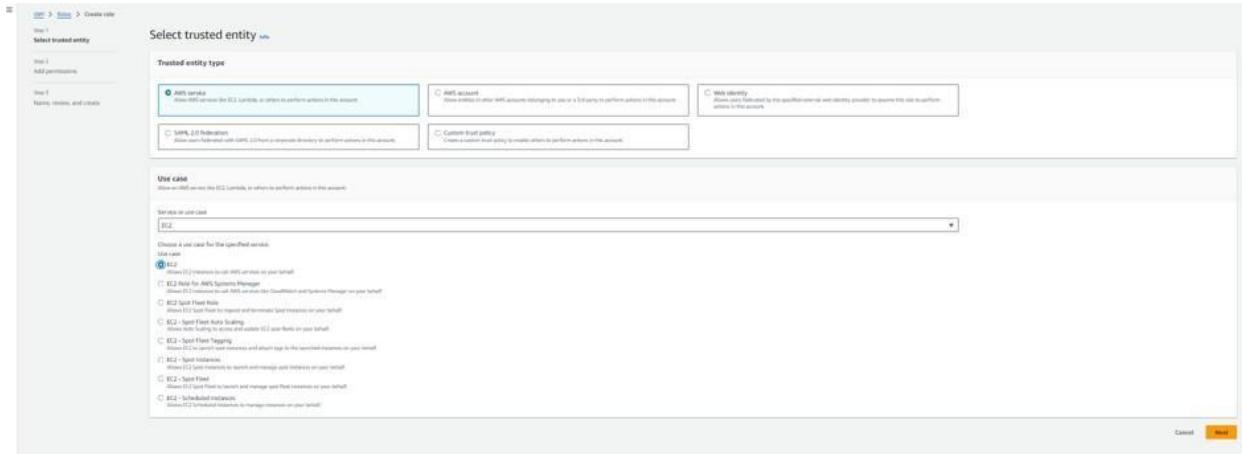
**Description:** The deployment code sets up and launches the homemade pickles and snacks website on a cloud platform like AWS or Heroku, ensuring public access. It typically includes configurations for web hosting, environment variables, and service integration to keep the app running smoothly.

## Milestone 5: IAM Role Setup

- **Activity 5.1: Create IAM Role.**

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





This screenshot shows the 'Select trusted entity' step of the 'Create role' wizard. It includes sections for 'Trusted entity type' (selected: 'AWS service'), 'User case' (selected: 'EC2'), and a detailed list of EC2 permissions.

**Trusted entity type:**

- AWS service: Allows this role to use the EC2 service, or others to perform actions in this account.
- AWS account: Allows entities in other AWS accounts belonging to you or a 3rd party to perform actions in this account.
- IAM user: Allows users federated by the specified external identity provider to assume this role to perform actions in this account.
- Custom trust policy: Creates a custom trust policy to enable others to perform actions in this account.

**User case:**

Allows entities to access the EC2 Lambda, or others to perform actions in this account.

Service or use case: **EC2**

Choose a use case for the specified service:

EC2: Allows EC2 instances to be used on your behalf.

EC2 Role for AWS Systems Manager: Allows EC2 instances to use the AWS Systems Manager service (CloudWatch and Systems Manager) on your behalf.

EC2 Spot Price Rule: Allows EC2 Spot Price to inquire and terminate Spot Instances on your behalf.

EC2 - Agent Fleet Auto Scaling: Allows EC2 Agent Fleet to manage EC2 user fleets on your behalf.

EC2 - Spot Fleet Tagging: Allows EC2 Spot Fleet instances and agents to be tagged in the launched instances on your behalf.

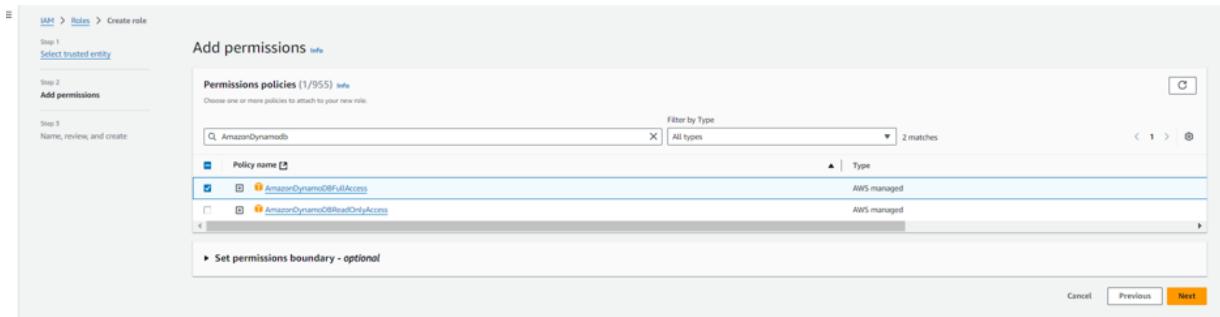
EC2 - Spot Fleet: Allows EC2 Spot Fleet to inquire and manage spot fleet instances on your behalf.

EC2 - Scheduled Instances: Allows EC2 Scheduled Instances to manage instances on your behalf.

## ● Activity 5.2: Attach Policies.

Attach the following policies to the role:

- **AmazonDynamoDBFullAccess:** Allows EC2 to perform read/write operations on DynamoDB.
- **AmazonSNSFullAccess:** Grants EC2 the ability to send notifications via SNS.



This screenshot shows the 'Add permissions' step of the 'Create role' wizard. It displays a list of available permissions, with two specifically selected: 'AmazonDynamoDBFullAccess' and 'AmazonDynamoDBReadOnlyAccess'.

**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"/> <a href="#">AmazonDynamoDBFullAccess</a>	AWS managed
<input checked="" type="checkbox"/> <a href="#">AmazonDynamoDBReadOnlyAccess</a>	AWS managed

Set permissions boundary - optional

Step 1: Select trusted entity

### Add permissions

Permissions policies (2/955) [Info](#)

Please use one or more policies to attach to your new role.

Policy name	Type
<input checked="" type="checkbox"/>  <a href="#">AmazonSNSFullAccess</a>	AWS managed
<input type="checkbox"/>  <a href="#">AmazonVPCReadOnlyAccess</a>	AWS managed
<input type="checkbox"/>  <a href="#">AmazonS3Role</a>	AWS managed
<input type="checkbox"/>  <a href="#">AWSLambdaBasicExecutionRole</a>	AWS managed
<input type="checkbox"/>  <a href="#">AWSIoTDevice DefenderPublishFindingsToSNSLogstashAction</a>	AWS managed

Set permissions boundary - optional

[Cancel](#) [Previous](#) [Next](#)

Step 1: Select trusted entities

Name, review, and create

Role details

Role name:  Maximum 255 characters. Use alphanumeric and \_ (underline) characters.

Description:  Maximum 1000 characters. Use letters A-Z and a-z, numbers 0-9, underscores, !@#\$%^&\*-~\_, and any of the following characters: !@#\$%^&\*-~\_

Step 2: Add permissions

Permissions policy summary

Policy name	Type	Attached as
<input checked="" type="checkbox"/> <a href="#">AmazonSNSFullAccess</a>	AWS managed	Permissions policy
<input type="checkbox"/> <a href="#">AmazonVPCReadOnlyAccess</a>	AWS managed	Permissions policy

Step 3: Add tags

Add tags - optional [Info](#)

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

No tags associated with the resource.

[Add new tag](#) You can add up to 50 tags. Help

[Cancel](#) [Previous](#) [Create role](#)

IAM > Roles > sns\_Dynamodb\_role

### sns\_Dynamodb\_role Info

Allows EC2 Instances to call AWS services on your behalf.

[Delete](#)

Summary		<a href="#">Edit</a>
Creation date	October 13, 2024, 23:06 (UTC+05:30)	ARN
Last activity	 6 days ago	Maximum session duration
		1 hour
<a href="#">Instance profile ARN</a>		
<a href="#">arn:aws:iam::557690616836:instance-profile/sns_Dynamodb_role</a>		

[Permissions](#) [Trust relationships](#) [Tags](#) [Last Accessed](#) [Revoke sessions](#)

**Permissions policies (2) Info**

You can attach up to 10 managed policies.

[Add permissions](#)

Filter by Type	
Search	All types
<input type="checkbox"/> Policy name 	Type
<input type="checkbox"/>  AmazonDynamoDBFullAccess	AWS managed
<input type="checkbox"/>  AmazonSNSFullAccess	AWS managed

us-east-1.console.aws.amazon.com/iam/home?region=us-east-1#roles

[AWS](#) [Global](#) [School](#)

[IAM](#) > Roles

### Roles (1/8) 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.

Role name	Trusted entities	Last activity
<a href="#">AWSServiceRoleForOrganizations</a>	AWS Service: organizations (Service-Linked Role)	218 days ago
<a href="#">AWSServiceRoleForSSO</a>	AWS Service: sso (Service-Linked Role)	-
<a href="#">AWSServiceRoleForSupport</a>	AWS Service: support (Service-Linked Role)	-
<a href="#">AWSServiceRoleForTrustedAdvisor</a>	AWS Service: trustedadvisor (Service-Linked Role)	-
<a href="#">DCEPrincipal-bunnytf</a>	Account: 058264256896	-
<a href="#">EC2_DynamoDB_Role</a> 	AWS Service: ec2	-
<a href="#">OrganizationAccountAccessRole</a>	Account: 058264256896	1 hour ago
<a href="#">rsoaccount-new</a>	Account: 058264256896	6 minutes ago

[Create role](#)

### Roles Anywhere Info

Authenticate your non AWS workloads and securely provide access to AWS services.

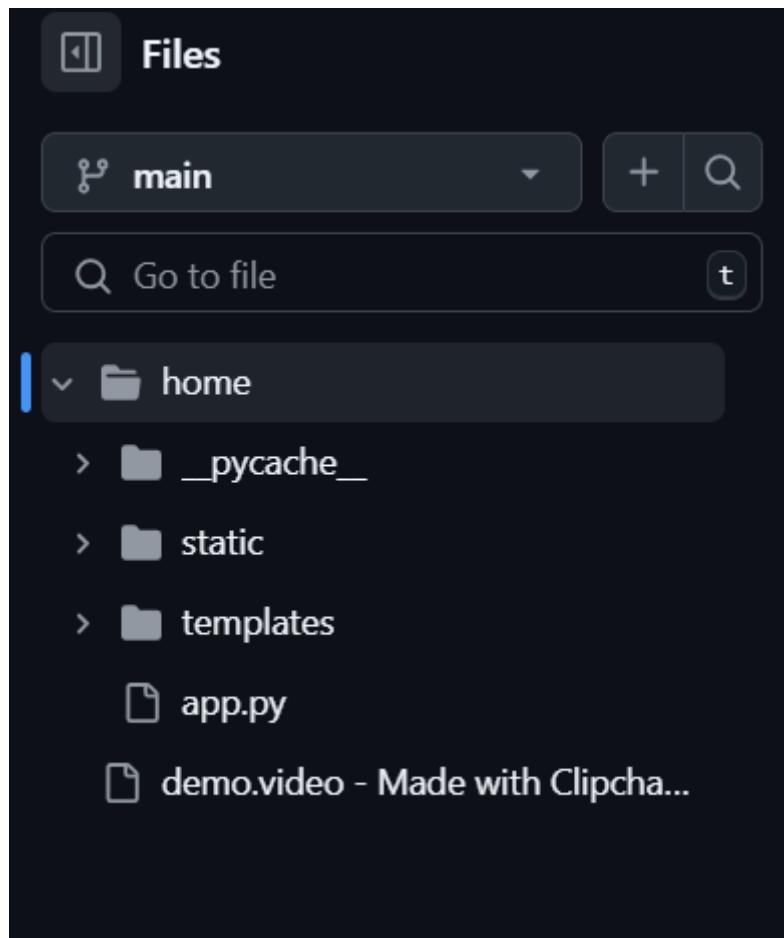
[Manage](#)

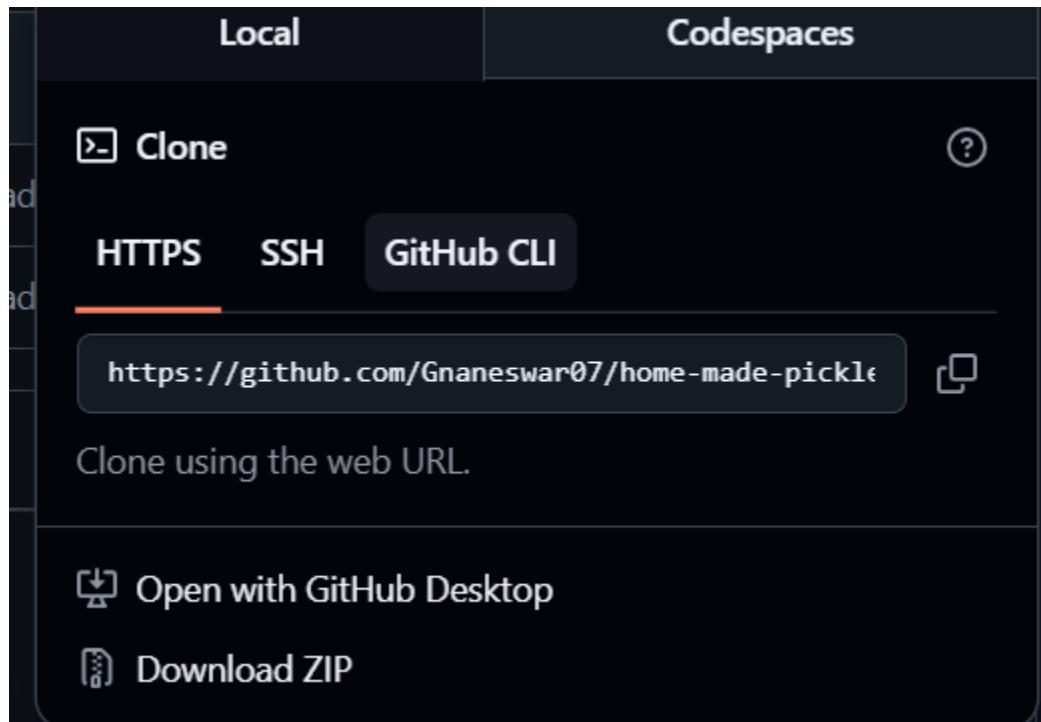
© 2025, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

CloudShell Feedback  500312 +1.60%  ENG IN 11:56 03-07-2025

## Milestone 6: EC2 Instance Setup

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

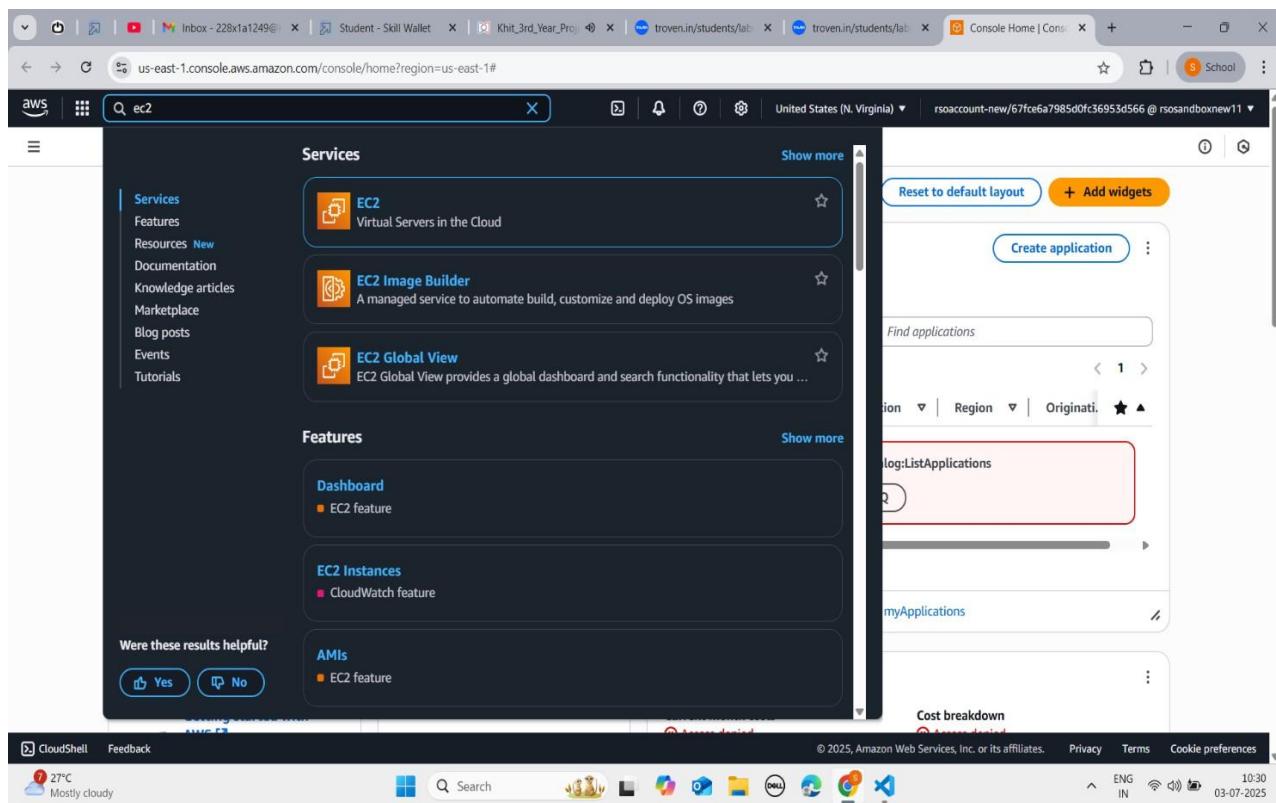




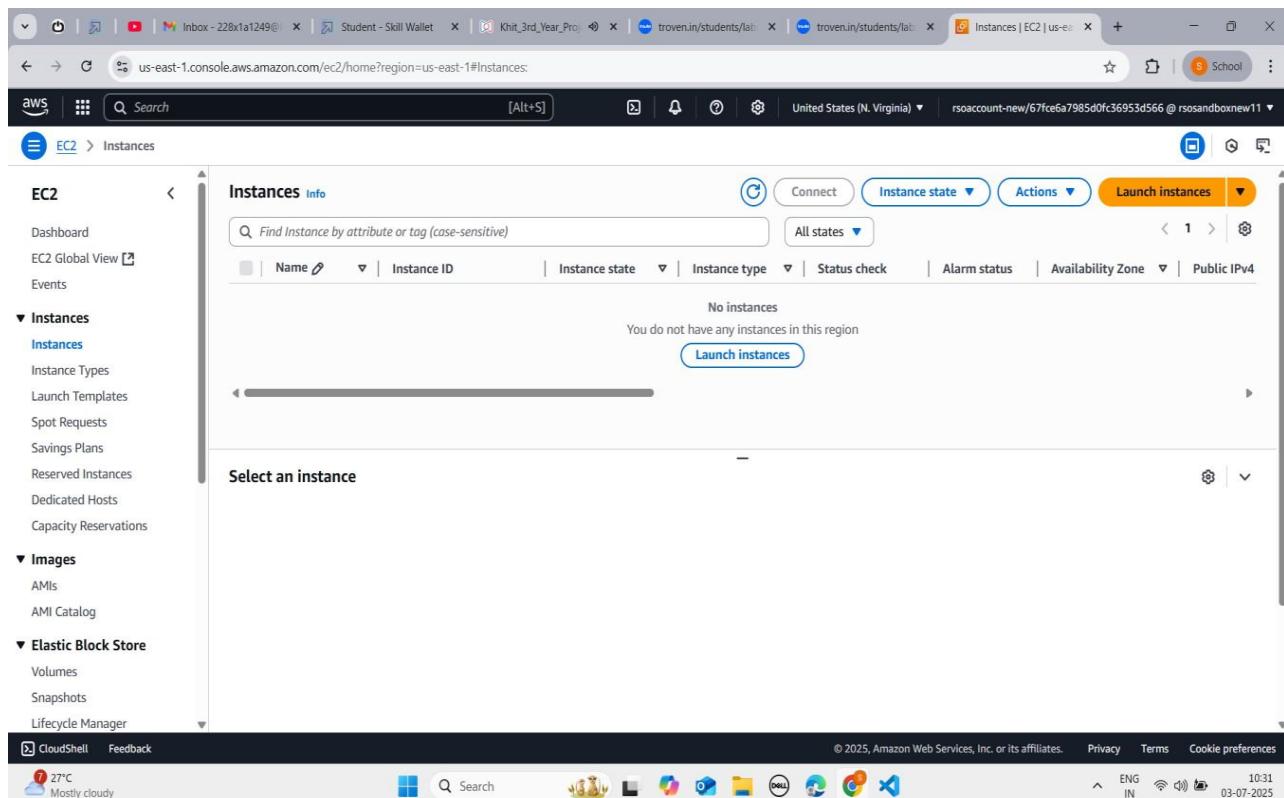
- **Activity 6.1: Launch an EC2 instance to host the Flask application.**

- **Launch EC2 Instance**

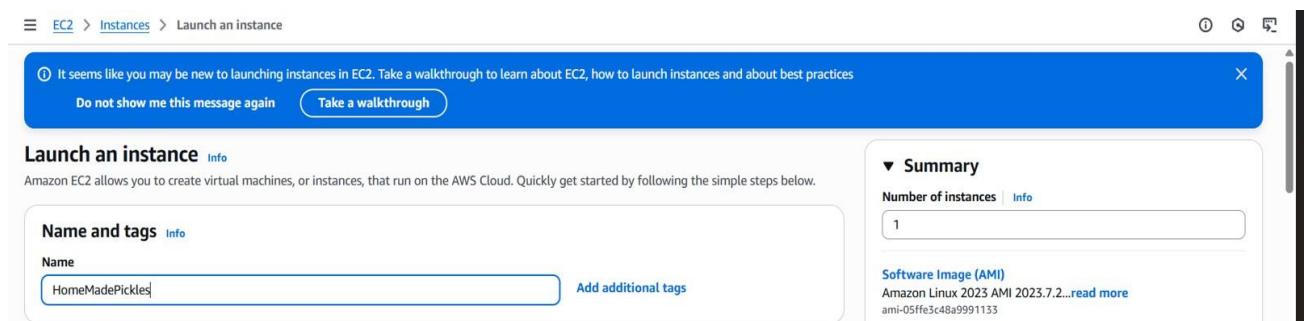
- In the AWS Console, navigate to EC2 and launch a new instance.



- Click on Launch instance to launch EC2 instance



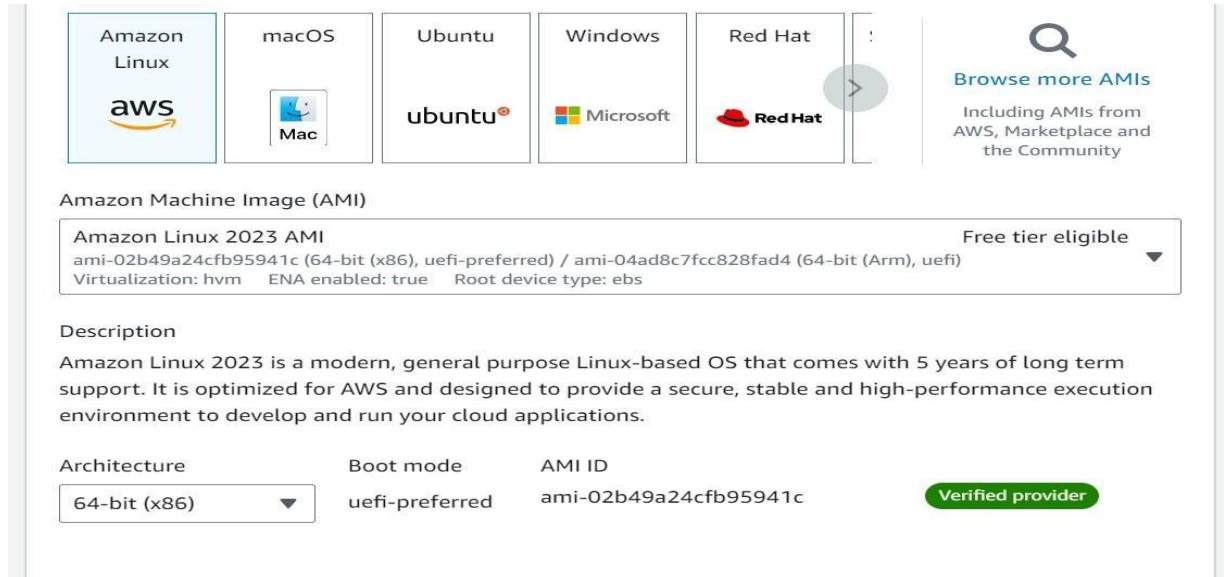
The screenshot shows the AWS EC2 Instances page. On the left, there's a navigation sidebar with sections like Dashboard, EC2 Global View, Events, Instances (which is expanded to show Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, and Capacity Reservations), Images, and Elastic Block Store. The main content area has a heading 'Instances Info' with a search bar and filters for Name, Instance ID, Instance state, Instance type, Status check, Alarm status, Availability Zone, and Public IPv4. A message says 'No instances' and 'You do not have any instances in this region'. Below this is a large blue button labeled 'Launch instances'. At the bottom of the page, there's a footer with links for CloudShell, Feedback, and various AWS services like Lambda, S3, and CloudWatch.



This screenshot shows the first step of the 'Launch an instance' wizard. It has a blue header bar with a message: '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' and buttons for 'Do not show me this message again' and 'Take a walkthrough'. The main content area is titled 'Launch an instance' and contains a sub-section 'Name and tags'. It shows a text input field with 'HomeMadePickles' and a link 'Add additional tags'. To the right, there's a summary section with a table:

Summary	
Number of instances	<a href="#">Info</a>
1	
Software Image (AMI)	
Amazon Linux 2023 AMI 2023.7.2... <a href="#">read more</a>	
ami-05ffe3c48a9991133	

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



Amazon Machine Image (AMI)

**Amazon Linux 2023 AMI**

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

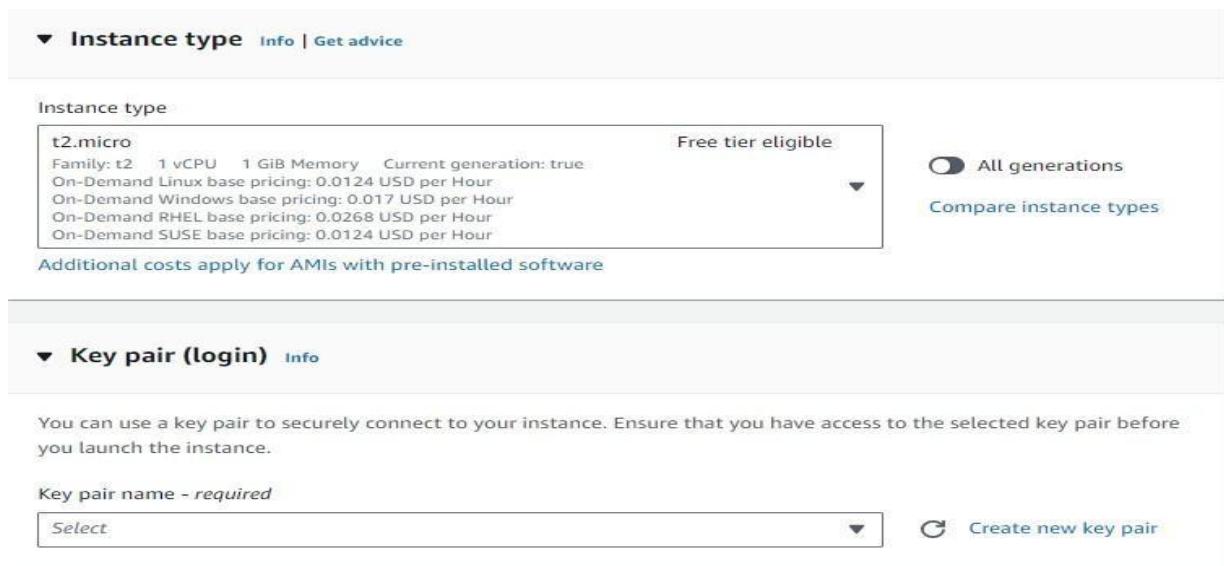
Free tier eligible ▾

Description

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

Architecture	Boot mode	AMI ID	Verified provider
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 ▾	<input checked="" type="button"/> Create new key pair
----------	---

us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#LaunchInstances:

aws Search [Alt+S] United States (N. Virginia) rsoaccount-new/67fce6a7985d0fc36953d566 @ rsosandboxnew11 ▾

EC2 > Instances > Launch an instance

64-bit (x86) uefi-preferred ami-05fdec48a9991133 2025-08-20 ec2-user verified provider

**Create key pair**

**Key pair name**  
Key pairs allow you to connect to your instance securely.

The name can include up to 255 ASCII characters. It can't include leading or trailing spaces.

**Key pair type**  
 RSA RSA encrypted private and public key pair  
 ED25519 ED25519 encrypted private and public key pair

**Private key file format**  
 .pem For use with OpenSSH  
 .ppk For use with PuTTY

**Summary**  
Number of instances Info

Software Image (AMI)  
Amazon Linux 2023 AMI 2023.7.2...read more  
05ffe3c48a9991133

Virtual server type (instance type)  
micro

Network security group  
security group

Storage (volumes)  
Volume(s) - 8 GiB

Cancel Launch instance Preview code

When prompted, store the private key in a secure and accessible location on your computer. You will need it later to connect to your instance. [Learn more](#)

Cancel Create key pair

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

27°C Mostly cloudy ENG IN 10:40 03-07-2025



- **Activity 6.2:Configure security groups for HTTP, and SSH access.**

**▼ Network settings [Info](#)**

VPC - required [Info](#)  
 vpc-03cdc7b6f19dd7211 (default)  
 172.31.0.0/16

Subnet [Info](#)  
 No preference [▼](#) [C](#) Create new subnet [\[x\]](#)

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) [Remove](#)

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

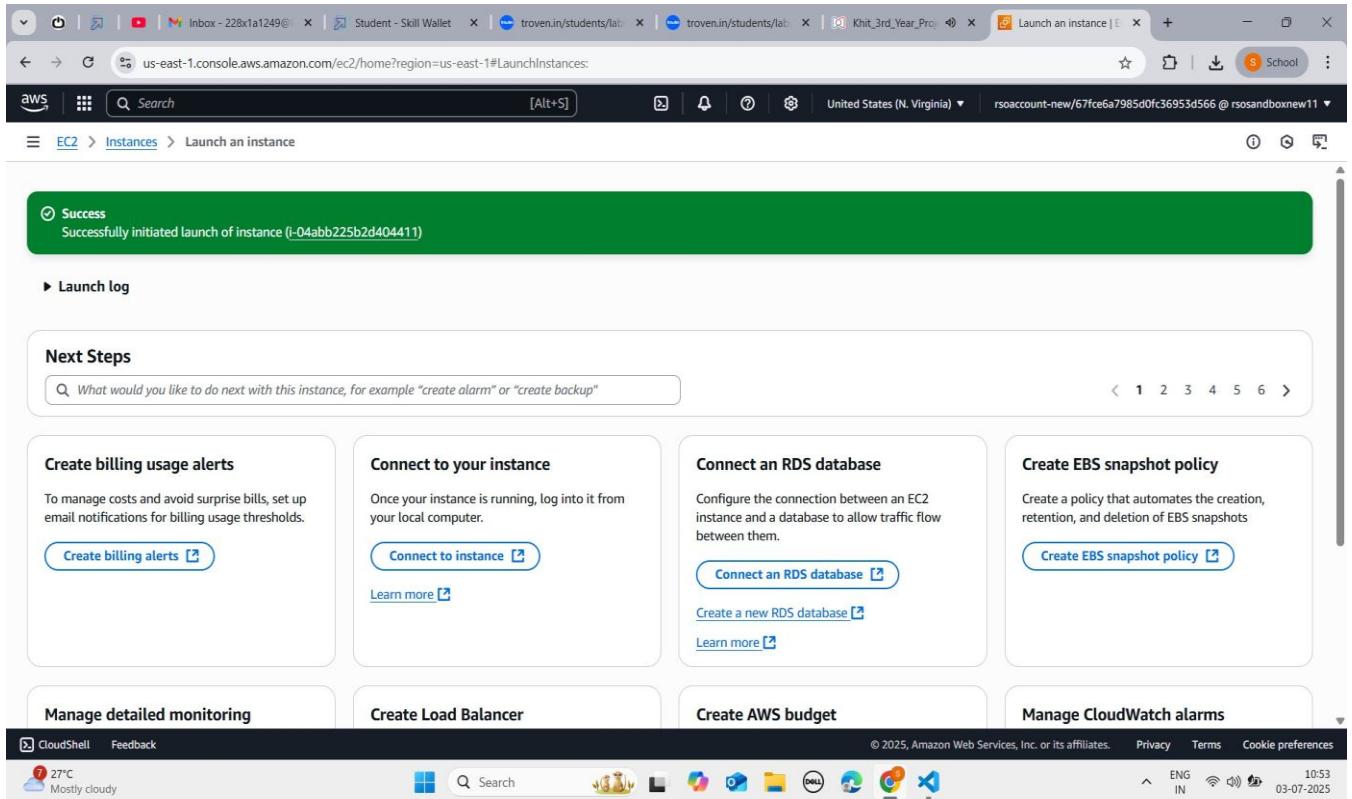
▼ Security group rule 2.(TCP, 80, 0.0.0.0/0) [Remove](#)

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

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

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

[Add security group rule](#)



The screenshot shows the AWS EC2 Instances Launch log page. At the top, there is a green success message: "Successfully initiated launch of instance (i-04abb225b2d404411)". Below this, there is a "Launch log" section. Further down, under "Next Steps", there are several options: "Create billing usage alerts", "Connect to your instance", "Connect an RDS database", "Create EBS snapshot policy", "Manage detailed monitoring", "Create Load Balancer", "Create AWS budget", and "Manage CloudWatch alarms". The bottom of the page includes standard browser navigation and status bars.

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.

us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#Instances:

Successfully attached EC2\_DynamoDB\_Role to instance i-04abb225b2d404411

**Instances (1/1)**

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IP
HomeMadePic...	i-04abb225b2d404411	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1b	ec2-34-2...

**i-04abb225b2d404411 (HomeMadePickles)**

**Details** Status and alarms Monitoring Security Networking Storage Tags

**Instance summary**

Instance ID	Public IPv4 address	Private IPv4 addresses
i-04abb225b2d404411	34.235.165.139   open address	172.31.28.8
IPv6 address	Instance state	Public DNS
-	Running	ec2-34-235-165-139.compute-1.amazonaws.com   open address

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences 27°C Mostly cloudy ENG IN 11:34 03-07-2025

- 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.

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

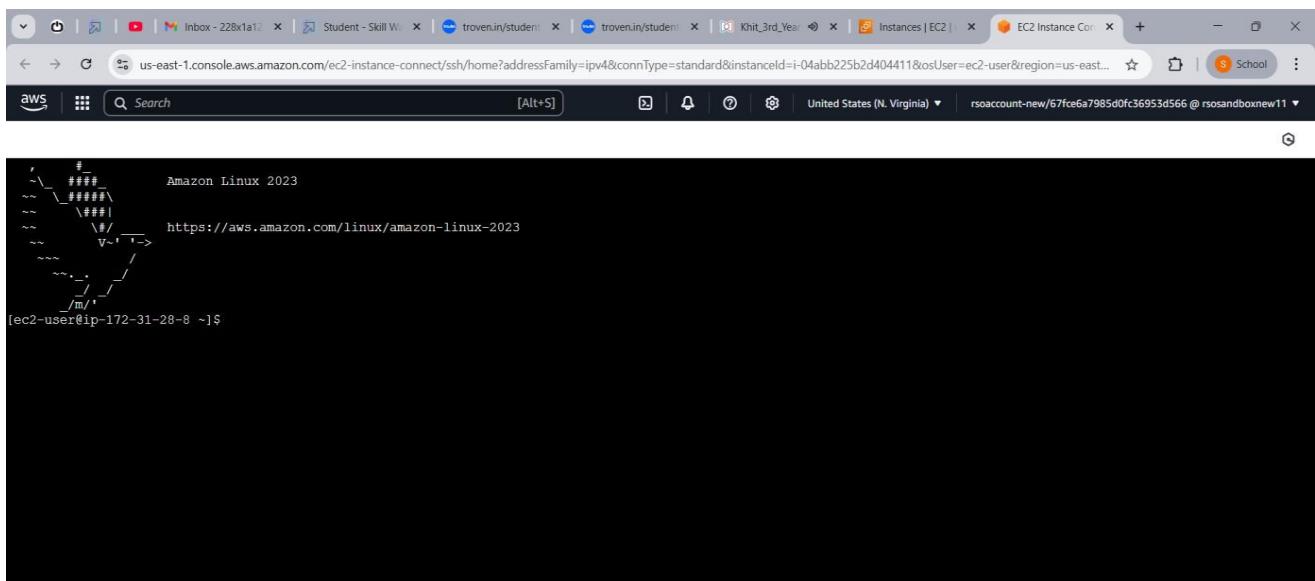
Public IPv4 address

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.

**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**



```

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

[ec2-user@ip-172-31-28-8 ~]$

```

i-04abb225b2d404411 (HomeMadePickles)

Public IPs: 34.235.165.139 Private IPs: 172.31.28.8

us-east-1.console.aws.amazon.com/ec2-instance-connect/ssh/home?addressFamily=ipv4&connType=standard&instanceId=i-04abb225b2d404411&osUser=ec2-user&region=us-east... School

aws | Search [Alt+S] United States (N. Virginia) ▾ rsoaccount-new/67fce6a7985d0fc36953d566 @ rsosandboxnew11 ▾

```

# Amazon Linux 2023
# https://aws.amazon.com/linux/amazon-linux-2023
[ec2-user@ip-172-31-28-8 ~]$ sudo yum update -y
sudo yum install python3 git
sudo pip3 install flask boto3
sudo yum install python3-pip -y
pip3 install Flask
pip3 install boto3
pip3 install python-dotenv
Amazon Linux 2023 Kernel Livepatch repository
Dependencies resolved.
Nothing to do.
Complete!
Last metadata expiration check: 0:00:03 ago on Thu Jul  3 07:06:08 2025.
Package python3-3.9.23-1.amzn2023.0.1.x86_64 is already installed.
Dependencies resolved.

=====
Package           Architecture      Version          Repository      Size
=====
Installing:
i-04abb225b2d404411 (HomeMadePickles)

```

164 kB/s | 17 kB 00:00

i-04abb225b2d404411 (HomeMadePickles)  
 Public IPs: 34.235.165.139 Private IPs: 172.31.28.8

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

30°C Mostly cloudy

CloudShell Search

```

Install 8 Packages

Total download size: 7.5 M
Installed size: 37 M
Is this ok [y/N]: y
Downloading Packages:
(1/8): git-2.47.1-1.amzn2023.0.3.x86_64.rpm
(2/8): perl-Error-0.17029-5.amzn2023.0.2.noarch.rpm
(3/8): git-core-doc-2.47.1-1.amzn2023.0.3.noarch.rpm
(4/8): perl-File-Find-1.37-477.amzn2023.0.7.noarch.rpm
(5/8): perl-Git-2.47.1-1.amzn2023.0.3.noarch.rpm
(6/8): perl-TermReadKey-2.38-9.amzn2023.0.2.x86_64.rpm
(7/8): git-core-2.47.1-1.amzn2023.0.3.x86_64.rpm
(8/8): perl-lib-0.65-477.amzn2023.0.7.x86_64.rpm

```

## Milestone 7: Deployment on EC2

### Activity 7.1: 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
sudo yum install python3-pip -y
pip3 install Flask
pip3 install boto3
pip3 install python-dotenv
```

Verify Installations:

```
flask --version
```

```
git --version
```

### Activity 7.2: Clone Your Flask Project from GitHub

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

Run: 'git clone https://github.com/Chandra123-123/Homemade.git'

Note: change your-github-username and your-repository-name with your credentials

here: 'git clone https://github.com/Chandra123-123/Homemade.git'

- This will download your project to the EC2 instance.

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

```
cd Homemade
```

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

#### Run the Flask Application

```
sudo flask run --host=0.0.0.0 --port=5000
```



```
aws [Q] Search [Alt+S] United States (N. Virginia) ▾ rsaccount-new/67fce6a7985d0fc36953d566 @ rsandboxnew11 ▾

Downloading boto3-1.39.2-py3-none-any.whl (139 kB)
|██████████| 139 kB 10.9 MB/s
Requirement already satisfied: jmespath<2.0.0,>=0.7.1 in /usr/lib/python3.9/site-packages (from boto3) (0.10.0)
Collecting botocore<1.40.0,>=1.39.2
  Downloading botocore-1.39.2-py3-none-any.whl (13.8 MB)
|██████████| 13.8 MB 16.8 MB/s
Collecting s3transfer<0.14.0,>=0.13.0
  Downloading s3transfer-0.13.0-py3-none-any.whl (85 kB)
|██████████| 85 kB 5.6 MB/s
Requirement already satisfied: urllib3<1.27,>=1.25.4 in /usr/lib/python3.9/site-packages (from botocore<1.40.0,>=1.39.2->boto3) (1.25.10)
Requirement already satisfied: python-dateutil<3.0.0,>=2.1 in /usr/lib/python3.9/site-packages (from botocore<1.40.0,>=1.39.2->boto3) (2.8.1)
Requirement already satisfied: six>=1.5 in /usr/lib/python3.9/site-packages (from python-dateutil<3.0.0,>=2.1->botocore<1.40.0,>=1.39.2->boto3) (1.15.0)
Installing collected packages: botocore, s3transfer, boto3
Successfully installed boto3-1.39.2 botocore-1.39.2 s3transfer-0.13.0
Defaulting to user installation because normal site-packages is not writeable
Collecting python-dotenv
  Downloading python_dotenv-1.1.1-py3-none-any.whl (20 kB)
Installing collected packages: python-dotenv
Successfully installed python-dotenv-1.1.1
[ec2-user@ip-172-31-28-8 ~]$ git clone https://github.com/Chandra123/Homemade.git
Cloning into 'Homemade'...
remote: Enumerating objects: 85, done.
remote: Counting objects: 100% (85/85), done.
remote: Compressing objects: 100% (79/79), done.
remote: Total 85 (delta 23), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (85/85), 734.70 KiB | 25.33 MiB/s, done.
Resolving deltas: 100% (23/23), done.
[ec2-user@ip-172-31-28-8 ~]$
```

i-04abb225b2d404411 (HomeMadePickles)

Public IPs: 34.235.165.139 Private IPs: 172.31.28.8



```
| [ 13.8 MB 16.8 MB/s ]  
Collecting s3transfer<0.14.0,>=0.13.0  
  Downloading s3transfer-0.13.0-py3-none-any.whl (85 kB)  
| [ 85 kB 5.6 MB/s ]  
Requirement already satisfied: urllib3<1.27.,>=1.25.4 in /usr/lib/python3.9/site-packages (from botocore<1.40.0,>=1.39.2->boto3) (1.25.10)  
Requirement already satisfied: python-dateutil<3.0.0,>=2.1 in /usr/lib/python3.9/site-packages (from botocore<1.40.0,>=1.39.2->boto3) (2.8.1)  
Requirement already satisfied: six>=1.5 in /usr/lib/python3.9/site-packages (from python-dateutil<3.0.0,>=2.1->botocore<1.40.0,>=1.39.2->boto3) (1.15.0)  
Installing collected packages: botocore, s3transfer, boto3  
Successfully installed boto3-1.39.2 botocore-1.39.2 s3transfer-0.13.0  
Defaulting to user installation because normal site-packages is not writable  
Collecting python-dotenv  
  Downloading python_dotenv-1.1.1-py3-none-any.whl (20 kB)  
Installing collected packages: python-dotenv  
Successfully installed python-dotenv-1.1.1  
[ec2-user@ip-172-31-28-8 ~]$ git clone https://github.com/Chandral23-123/Homemade.git  
Cloning into 'Homemade'...  
remote: Enumerating objects: 85, done.  
remote: Counting objects: 100% (85/85), done.  
remote: Compressing objects: 100% (79/79), done.  
remote: Total 85 (delta 23), reused 0 (delta 0), pack-reused 0 (from 0)  
Receiving objects: 100% (85/85), 734.70 KiB | 25.33 MiB/s, done.  
Resolving deltas: 100% (23/23), done.  
[ec2-user@ip-172-31-28-8 ~]$ cd Homemade  
[ec2-user@ip-172-31-28-8 Homemade]$ python3 app.py  
* Serving Flask app 'app'  
* 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)
```

j-04abb225b2d404411 (HomeMadePickles)

Public IPs: 34.235.165.139 Private IPs: 172.31.28.8





```
[ec2-user@ip-172-31-28-8 ~]$ cd Homemade
[ec2-user@ip-172-31-28-8 Homemade]$ python3 app.py
 * Serving Flask app "app"
 * 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://127.0.0.1:5000
 * Running on http://172.31.28.8:5000
Press CTRL+C to quit
164.90.141.165 - - [03/Jul/2025 07:36:00] "GET / HTTP/1.1" 200 -
164.90.141.165 - - [03/Jul/2025 07:36:00] "GET /favicon.ico HTTP/1.1" 404 -
□
```

i-04abb225b2d404411 (HomeMadePickles)

Public IPs: 34.235.165.139 Private IPs: 172.31.28.8

```
Resolving deltas: 100% (23/23), done.
[ec2-user@ip-172-31-28-8 ~]$ cd Homemade
[ec2-user@ip-172-31-28-8 Homemade]$ python3 app.py
 * Serving Flask app 'app'
 * 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://127.0.0.1:5000
 * Running on http://172.31.28.8:5000
Press CTRL+C to quit
164.90.141.165 - - [03/Jul/2025 07:36:00] "GET / HTTP/1.1" 200 -
164.90.141.165 - - [03/Jul/2025 07:36:00] "GET /favicon.ico HTTP/1.1" 404 -
223.228.99.42 - - [03/Jul/2025 07:37:52] "GET / HTTP/1.1" 200 -
223.228.99.42 - - [03/Jul/2025 07:37:52] "GET /favicon.ico HTTP/1.1" 404 -
223.228.99.42 - - [03/Jul/2025 07:38:14] "GET /signup HTTP/1.1" 200 -
49.205.96.120 - - [03/Jul/2025 07:38:56] "GET / HTTP/1.1" 200 -
49.205.96.120 - - [03/Jul/2025 07:38:56] "GET /favicon.ico HTTP/1.1" 404 -
223.228.99.42 - - [03/Jul/2025 07:39:00] "GET / HTTP/1.1" 200 -
49.43.218.175 - - [03/Jul/2025 07:39:22] "GET / HTTP/1.1" 200 -
49.43.218.175 - - [03/Jul/2025 07:39:22] "GET /favicon.ico HTTP/1.1" 404 -
223.228.104.102 - - [03/Jul/2025 07:40:40] code 400, message Bad request version ('(\x00\x12\x00\x10\x04\x03\x08\x04\x04\x01\x05\x03\x08\x08\x05\x05\x01\x08\x06\x06\x01\x003\x04!\x04!')
```

i-04abb225b2d404411 (HomeMadePickles)

Public IPs: 34.235.165.139 Private IPs: 172.31.28.8

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

6 30°C Search DELL 13:11  
Mostly cloudy ENG IN 03-07-2025

us-east-1.console.aws.amazon.com/ec2-instance-connect/ssh/home?addressFamily=ipv4&connType=standard&instanceId=i-04abb225b2d404411&osUser=ec2-user&region=us-east...

Search [Alt+5] United States (N. Virginia) rsoaccount-new/67fce6a7985d0fc36953d566 @ rsosandboxnew11

```

  _###_ Amazon Linux 2023
  ~\###\
  ~~ \###|
  ~~ \|/ V~-->
  ~~ /`/
  ~~ /`/
  _m/`/
[ec2-user@ip-172-31-28-8 ~]$ sudo yum update -y
sudo yum install python3 git
sudo pip3 install flask boto3
sudo yum install python3-pip -y
pip3 install Flask
pip3 install boto3
pip3 install python-dotenv
Amazon Linux 2023 Kernel Livepatch repository
Dependencies resolved.
Nothing to do.
Complete!
Last metadata expiration check: 0:00:03 ago on Thu Jul 3 07:06:08 2025.
Package python3-3.9.23-1.amzn2023.0.1.x86_64 is already installed.
Dependencies resolved.

=====
Package           Architecture      Version       Repository      Size
=====
Installing:
i-04abb225b2d404411 (HomeMadePickles)

Public IPs: 34.235.165.139 Private IPs: 172.31.28.8

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences
30°C Mostly cloudy Search ENG IN 13:35 03-07-2025

[ec2-user@ip-172-31-28-8 ~]$ cd Homemade
[ec2-user@ip-172-31-28-8 Homemade]$ python3 app.py
 * Serving Flask app 'app'
 * 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://127.0.0.1:5000
 * Running on http://172.31.28.8:5000
Press CTRL+C to quit
164.90.141.165 - - [03/Jul/2025 07:36:00] "GET / HTTP/1.1" 200 -
164.90.141.165 - - [03/Jul/2025 07:36:00] "GET /favicon.ico HTTP/1.1" 404 -

```

**i-04abb225b2d404411 (HomeMadePickles)**

Public IPs: 34.235.165.139 Private IPs: 172.31.28.8

## Verify the Flask app is running:

<http://13.218.225.254>

Run the Flask app on the EC2

instance



```
aws Search [Alt+S] United States (N. Virginia) rsoaccount-new/67fce6a7985d0fc36953d566 @rsosandboxnew11 ▾

Resolving deltas: 100% (23/23), done.
[ec2-user@ip-172-31-28-8 ~]$ cd Homemade
[ec2-user@ip-172-31-28-8 Homemade]$ python3 app.py
 * Serving Flask app 'app'
 * 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://127.0.0.1:5000
 * Running on http://172.31.28.8:5000
Press CTRL+C to quit
164.90.141.165 - - [03/Jul/2025 07:36:00] "GET / HTTP/1.1" 200 -
164.90.141.165 - - [03/Jul/2025 07:36:00] "GET /favicon.ico HTTP/1.1" 404 -
223.228.99.42 - - [03/Jul/2025 07:37:52] "GET / HTTP/1.1" 200 -
223.228.99.42 - - [03/Jul/2025 07:37:52] "GET /favicon.ico HTTP/1.1" 404 -
223.228.99.42 - - [03/Jul/2025 07:38:14] "GET /signmp HTTP/1.1" 200 -
49.205.96.120 - - [03/Jul/2025 07:38:56] "GET / HTTP/1.1" 200 -
49.205.96.120 - - [03/Jul/2025 07:38:56] "GET /favicon.ico HTTP/1.1" 404 -
223.228.99.42 - - [03/Jul/2025 07:39:00] "GET / HTTP/1.1" 200 -
49.43.218.175 - - [03/Jul/2025 07:39:22] "GET / HTTP/1.1" 200 -
49.43.218.175 - - [03/Jul/2025 07:39:22] "GET /favicon.ico HTTP/1.1" 404 -
223.228.104.102 - - [03/Jul/2025 07:40:40] code 400, message Bad request version ('(\x00\x12\x00\x10\x04\x03\x08\x04\x04\x01\x05\x03\x08\x05\x05\x01\x08\x06\x06\x01\x00\x3\x04!\x04!')
```

i-04abb225b2d404411 (HomeMadePickles)

Public IPs: 34.235.165.139 Private IPs: 172.31.28.8

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences 6 30°C Search 13:11 Mostly cloudy ENG IN 03-07-2025

**Access the website through:**

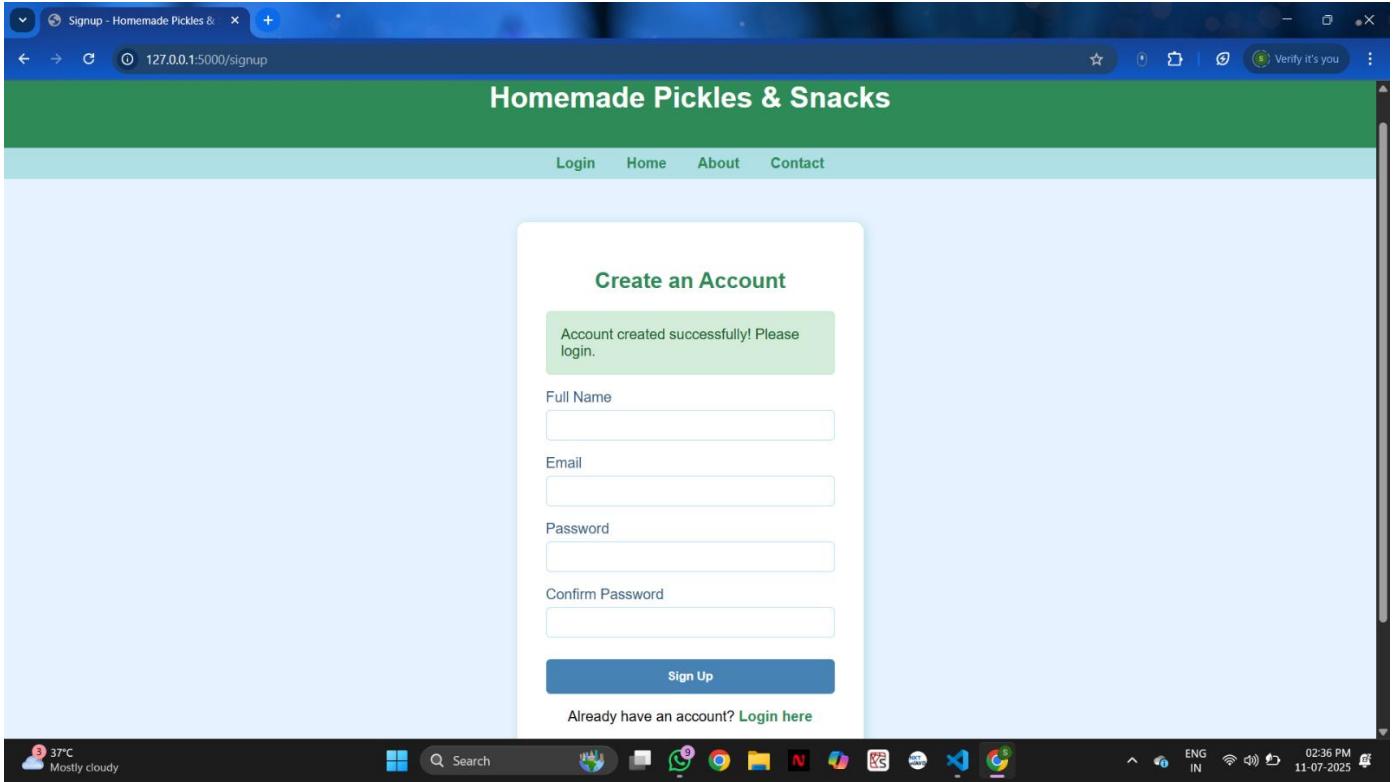
Public IPs:

<http://13.218.225.254>

## Milestone 8: Testing and Deployment

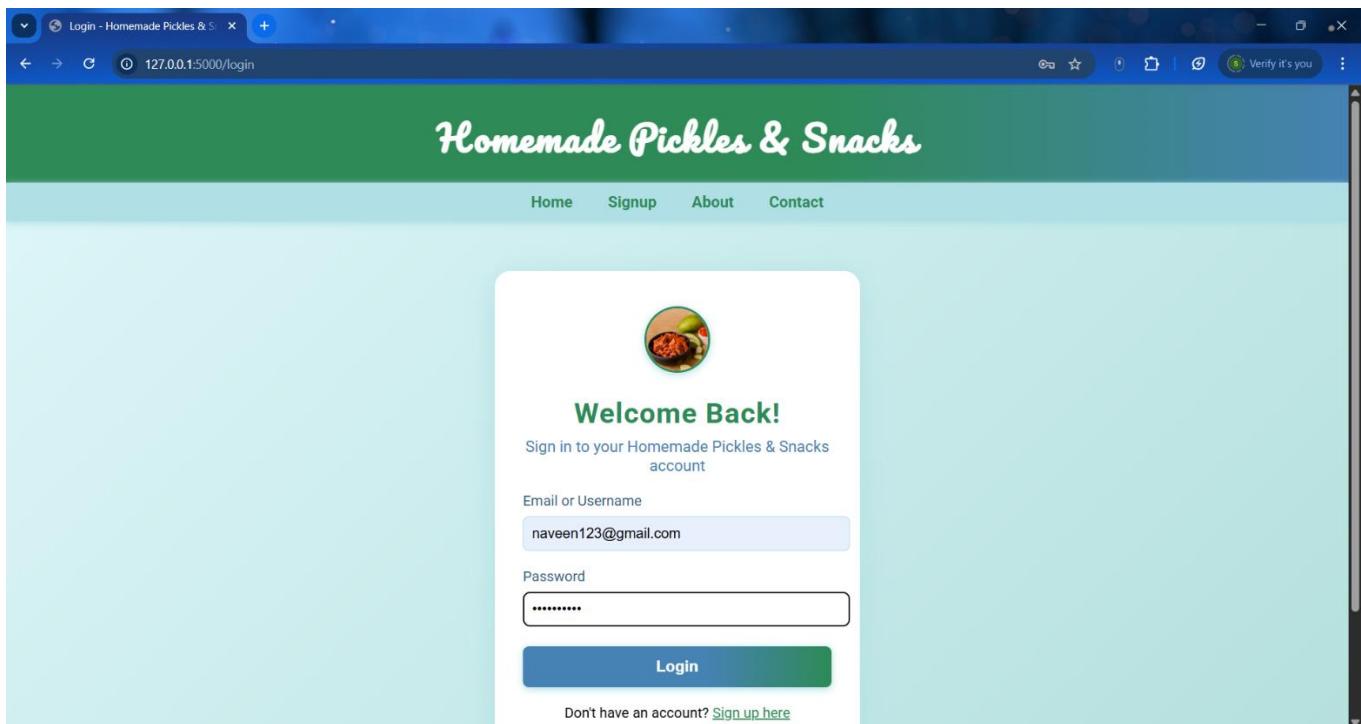
- **Activity 8.1: Conduct functional testing to verify user signup, login, pickles requests, and notifications.**

### Signup Page:

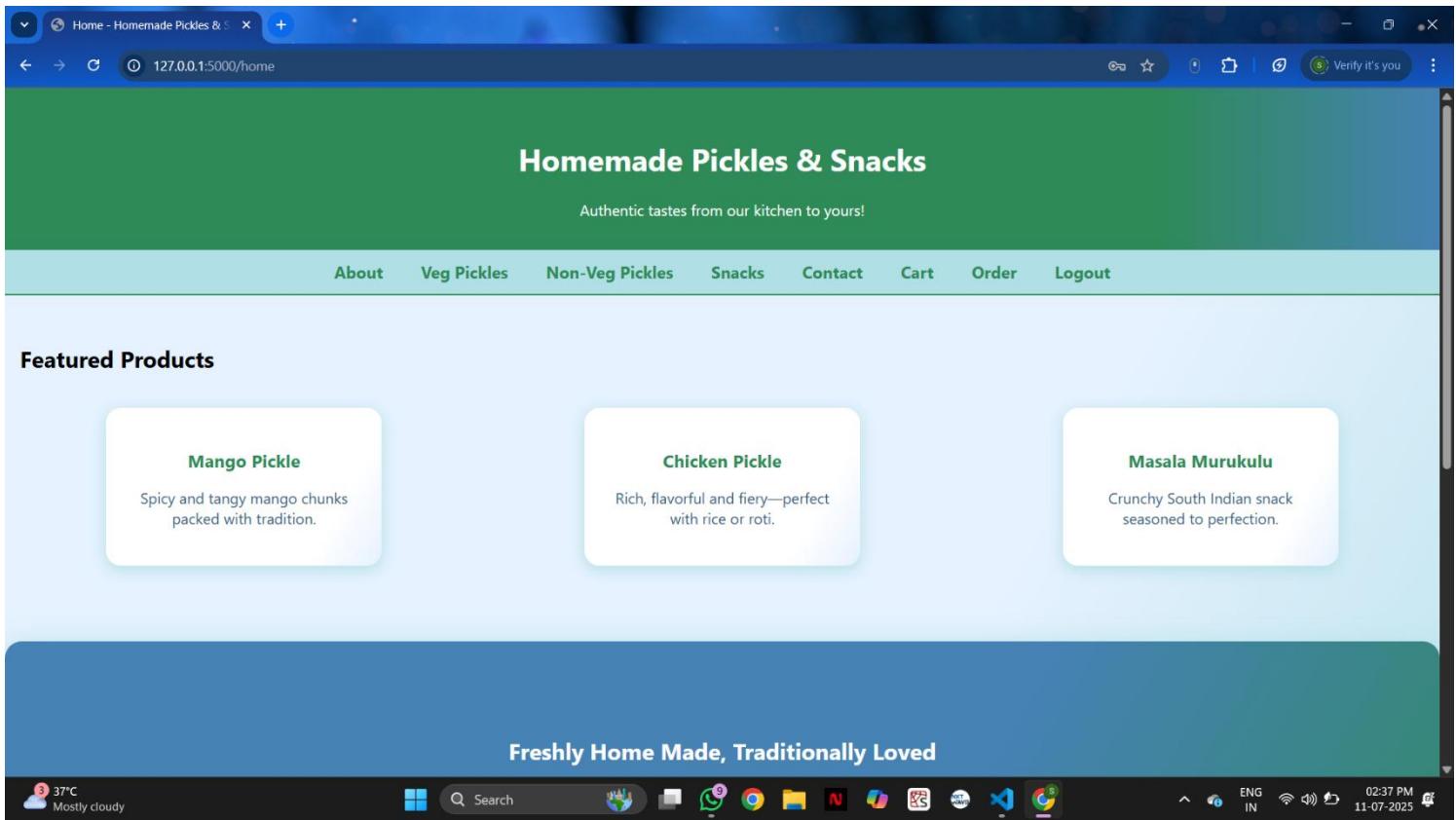


The screenshot shows a web browser window with the title "Signup - Homemade Pickles & Snacks". The URL in the address bar is "127.0.0.1:5000/signup". The page has a green header bar with the text "Homemade Pickles & Snacks" and a navigation menu with links "Login", "Home", "About", and "Contact". Below the header is a "Create an Account" form. A green success message box contains the text "Account created successfully! Please login.". The form includes input fields for "Full Name", "Email", "Password", and "Confirm Password", each with a corresponding placeholder text. At the bottom of the form is a blue "Sign Up" button. Below the form, a link says "Already have an account? [Login here](#)". The browser's taskbar at the bottom shows various open tabs and system icons, including weather information (37°C, Mostly cloudy), search, file explorer, messaging, and other application icons. The date and time in the bottom right corner are "11-07-2025 02:36 PM".

## Login Page:



## Home page:



The screenshot shows a web browser window displaying a homepage for "Homemade Pickles & Snacks". The title bar indicates the URL is 127.0.0.1:5000/home. The page has a green header with the title "Homemade Pickles & Snacks" and a subtitle "Authentic tastes from our kitchen to yours!". A navigation bar below the header includes links for About, Veg Pickles, Non-Veg Pickles, Snacks, Contact, Cart, Order, and Logout. The main content area features three sections under the heading "Featured Products": "Mango Pickle" (spicy and tangy mango chunks packed with tradition), "Chicken Pickle" (rich, flavorful and fiery—perfect with rice or roti), and "Masala Murukulu" (crunchy South Indian snack seasoned to perfection). At the bottom of the page is a blue footer bar with the text "Freshly Home Made, Traditionally Loved". The system tray at the bottom of the screen shows weather (37°C, mostly cloudy), search, file explorer, taskbar icons, and system status.

Home - Homemade Pickles & Snacks

127.0.0.1:5000/home

Verify it's you

## Homemade Pickles & Snacks

Authentic tastes from our kitchen to yours!

About    Veg Pickles    Non-Veg Pickles    Snacks    Contact    Cart    Order    Logout

### Featured Products

**Mango Pickle**  
Spicy and tangy mango chunks packed with tradition.

**Chicken Pickle**  
Rich, flavorful and fiery—perfect with rice or roti.

**Masala Murukulu**  
Crunchy South Indian snack seasoned to perfection.

Freshly Home Made, Traditionally Loved

37°C  
Mostly cloudy

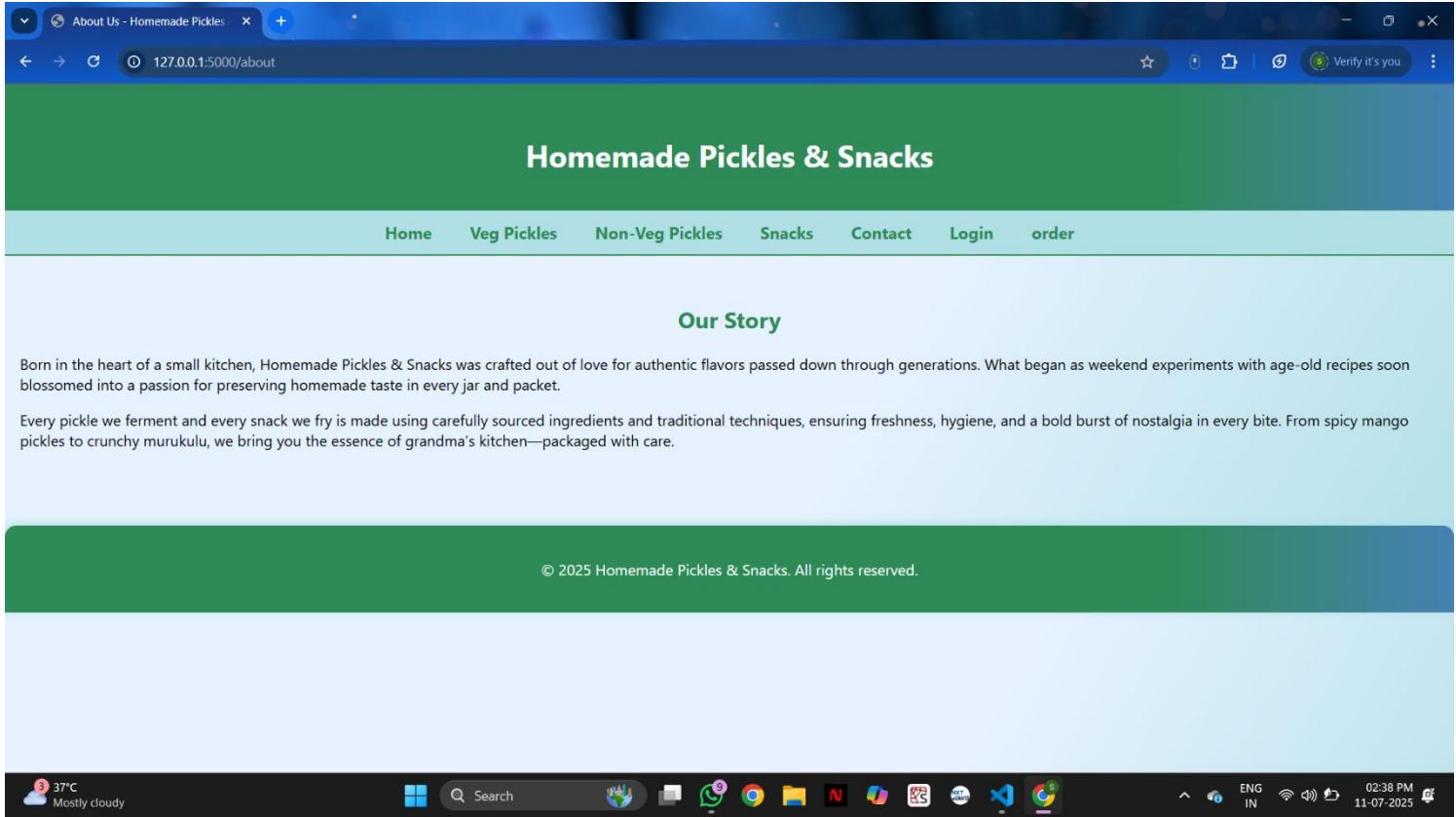
Search

File Explorer

Taskbar Icons

System Status

## About Us page:



The screenshot shows a web browser window with the title "About Us - Homemade Pickles" and the URL "127.0.0.1:5000/about". The page has a green header with the text "Homemade Pickles & Snacks". Below the header is a navigation bar with links: Home, Veg Pickles, Non-Veg Pickles, Snacks, Contact, Login, and order. A section titled "Our Story" contains text about the company's history and ingredients. At the bottom, there is a footer with the copyright notice "© 2025 Homemade Pickles & Snacks. All rights reserved." and a taskbar at the bottom of the screen.

**Our Story**

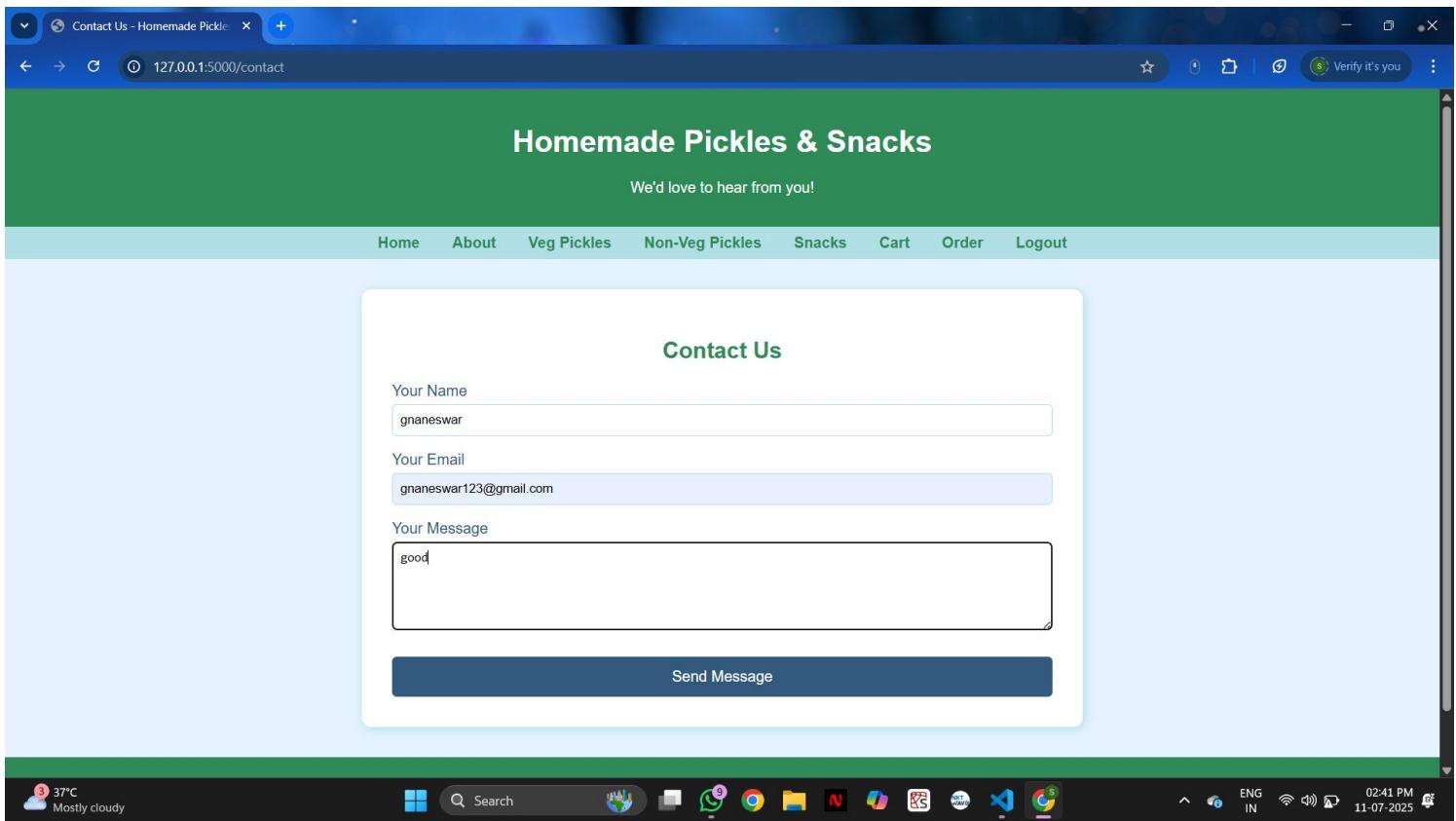
Born in the heart of a small kitchen, Homemade Pickles & Snacks was crafted out of love for authentic flavors passed down through generations. What began as weekend experiments with age-old recipes soon blossomed into a passion for preserving homemade taste in every jar and packet.

Every pickle we ferment and every snack we fry is made using carefully sourced ingredients and traditional techniques, ensuring freshness, hygiene, and a bold burst of nostalgia in every bite. From spicy mango pickles to crunchy murukulu, we bring you the essence of grandma's kitchen—packaged with care.

© 2025 Homemade Pickles & Snacks. All rights reserved.

37°C Mostly cloudy      Search      11-07-2025      02:38 PM

## Contact Page:



The screenshot shows a contact form on a website for "Homemade Pickles & Snacks". The page has a green header with the title "Homemade Pickles & Snacks" and a subtext "We'd love to hear from you!". Below the header is a navigation bar with links: Home, About, Veg Pickles, Non-Veg Pickles, Snacks, Cart, Order, and Logout. The main content area contains a "Contact Us" form with three input fields: "Your Name" (filled with "gnaneswar"), "Your Email" (filled with "gnaneswar123@gmail.com"), and "Your Message" (filled with "good"). A "Send Message" button is at the bottom of the form. The browser's address bar shows the URL "127.0.0.1:5000/contact". The taskbar at the bottom of the screen displays various application icons and system status information.

Contact Us

Your Name  
gnaneswar

Your Email  
gnaneswar123@gmail.com

Your Message  
good

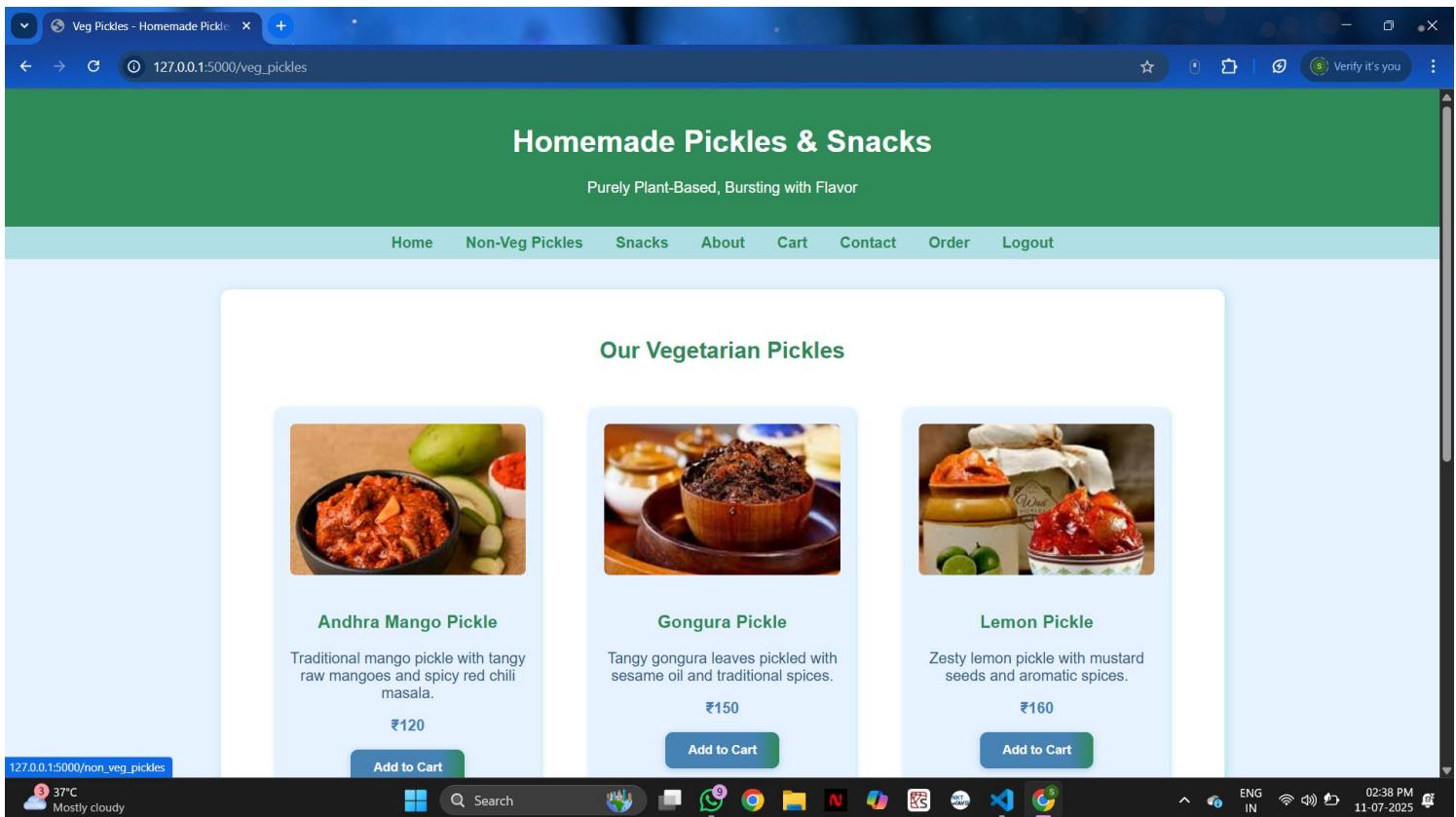
Send Message

37°C Mostly cloudy

Search

02:41 PM 11-07-2025

## Veg-Pickles page :

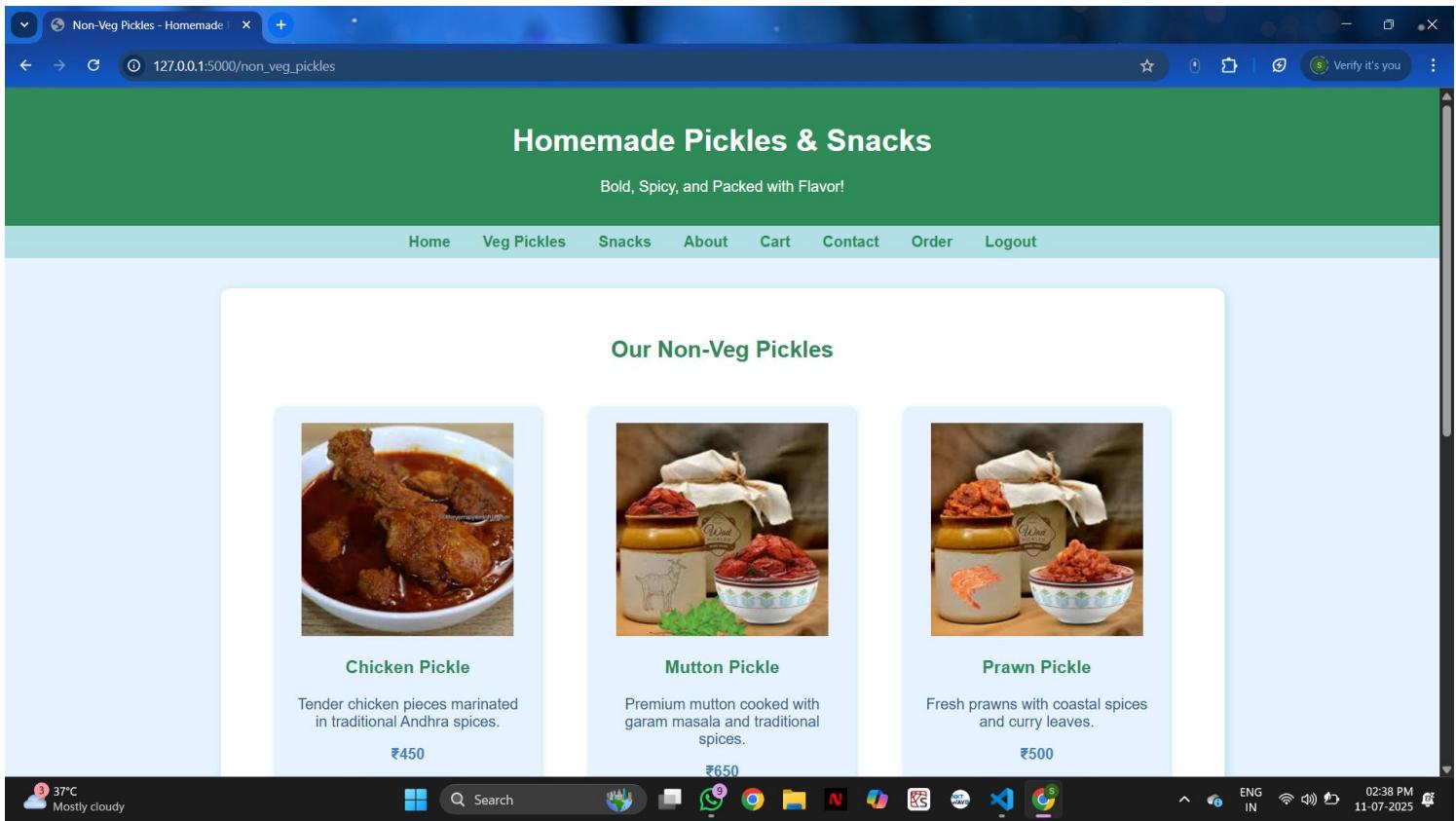


The screenshot shows a web browser displaying a homepage for "Homemade Pickles & Snacks". The header features a green banner with the title "Homemade Pickles & Snacks" and the tagline "Purely Plant-Based, Bursting with Flavor". Below the banner is a navigation bar with links for Home, Non-Veg Pickles, Snacks, About, Cart, Contact, Order, and Logout. The main content area is titled "Our Vegetarian Pickles" and displays three products:

- Andhra Mango Pickle**: Traditional mango pickle with tangy raw mangoes and spicy red chili masala. Price: ₹120. Add to Cart button.
- Gongura Pickle**: Tangy gongura leaves pickled with sesame oil and traditional spices. Price: ₹150. Add to Cart button.
- Lemon Pickle**: Zesty lemon pickle with mustard seeds and aromatic spices. Price: ₹160. Add to Cart button.

The bottom of the screen shows a taskbar with various icons and system status information, including weather (37°C, Mostly cloudy), search, and system controls.

## Non-Veg-Pickles page:



**Homemade Pickles & Snacks**

Bold, Spicy, and Packed with Flavor!

Home   Veg Pickles   Snacks   About   Cart   Contact   Order   Logout

### Our Non-Veg Pickles

**Chicken Pickle**  
Tender chicken pieces marinated in traditional Andhra spices.  
₹450

**Mutton Pickle**  
Premium mutton cooked with garam masala and traditional spices.  
₹650

**Prawn Pickle**  
Fresh prawns with coastal spices and curry leaves.  
₹500

37°C Mostly cloudy

Search

ENG IN 02:38 PM 11-07-2025

## Snacks Page:

Snacks - Homemade Pickles & Snacks

127.0.0.1:5000/snacks

### Homemade Pickles & Snacks

South Indian Crunch & Munch Specials

Home   Veg Pickles   Non-Veg Pickles   About   Cart   Contact   Order   Logout

#### Our Handmade Snacks



**Masala Murukulu**  
Crunchy rice flour spirals seasoned with traditional spices.  
₹80



**Karam Boondi**  
Spicy gram flour pearls with curry leaves and masala.



**Mixture**  
Crispy blend of sev, peanuts, and curry leaves with spices.  
₹90

37°C Mostly cloudy

Search

02:38 PM 11-07-2025

## Cart Page:

Cart - Homemade Pickles & Snacks

127.0.0.1:5000/cart

### Homemade Pickles & Snacks

Your Cart

Product	Quantity	Unit Price	Subtotal	Action
Andhra Mango Pickle	<span>-</span> <span>1</span> <span>+</span>	₹120	₹120	<span>Remove</span>
Gongura Pickle	<span>-</span> <span>1</span> <span>+</span>	₹150	₹150	<span>Remove</span>
Mutton Pickle	<span>-</span> <span>1</span> <span>+</span>	₹650	₹650	<span>Remove</span>
Chicken Pickle	<span>-</span> <span>1</span> <span>+</span>	₹450	₹450	<span>Remove</span>

Total: ₹1370

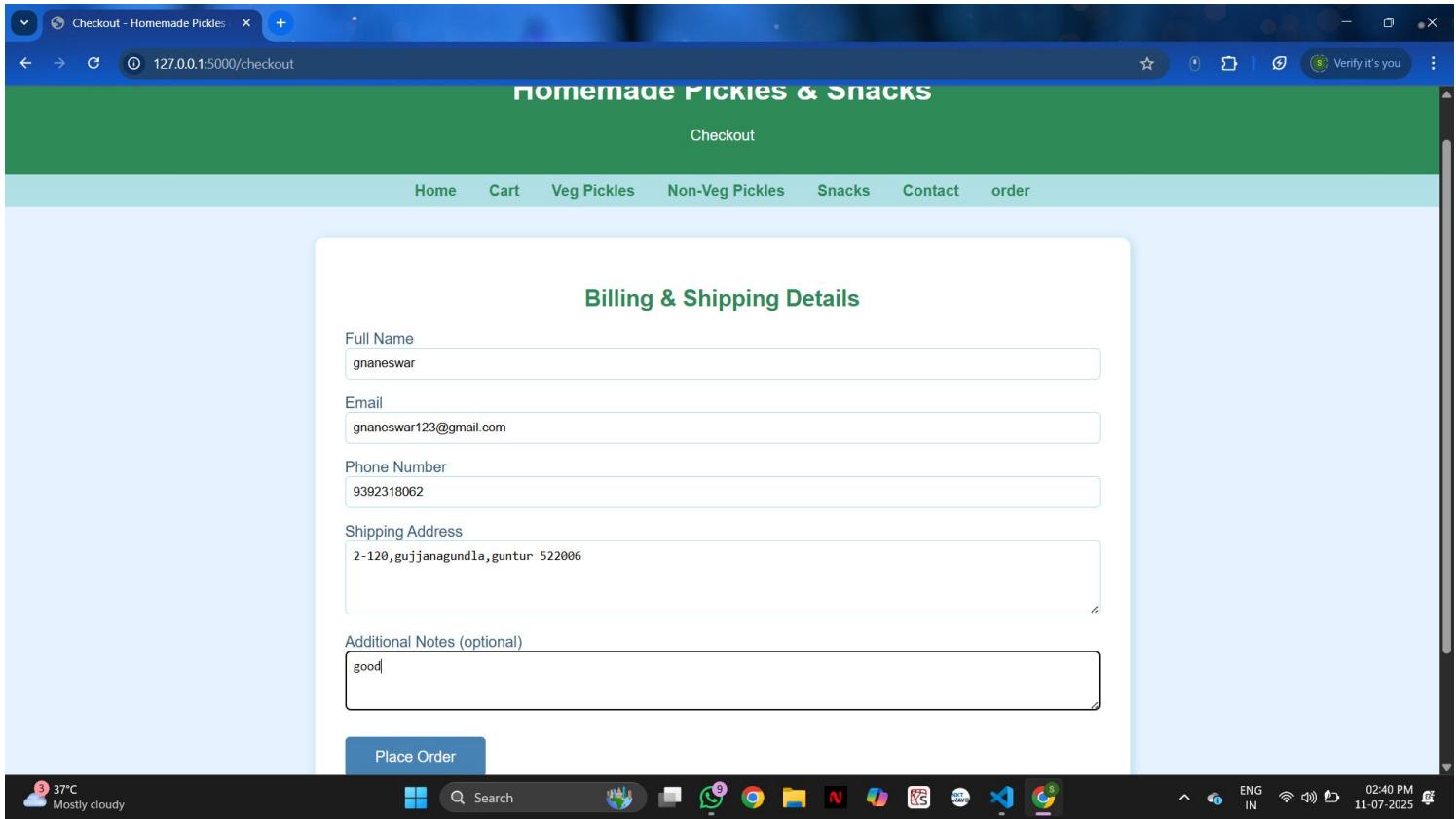
Proceed to Checkout

37°C Mostly cloudy

Search

02:39 PM 11-07-2025

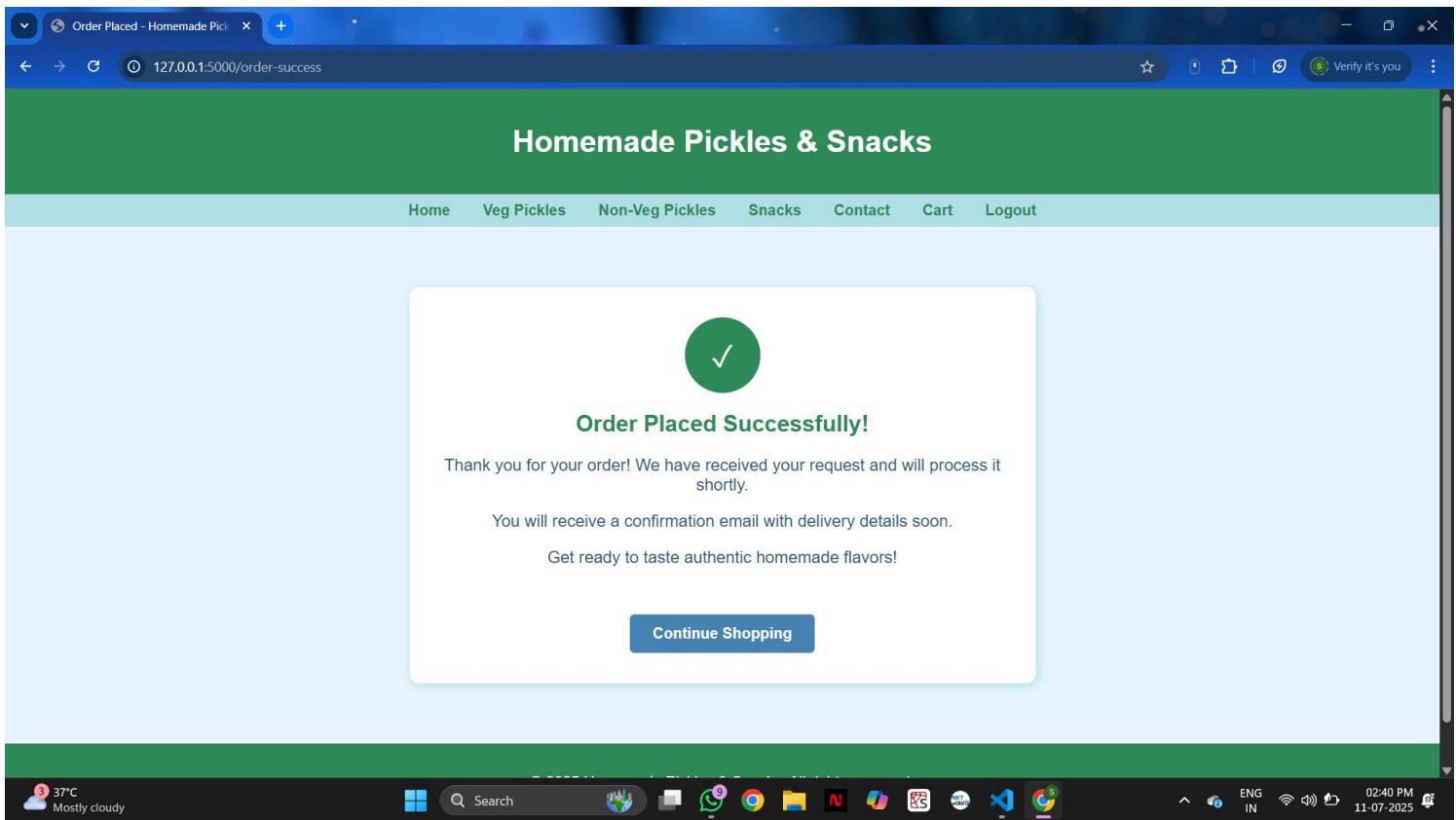
## Check-out Page:



The screenshot shows a web browser window with the following details:

- Title Bar:** Checkout - Homemade Pickles
- Address Bar:** 127.0.0.1:5000/checkout
- Page Header:** **HOMEMADE PICKLES & SNACKS**
- Page Title:** Checkout
- Navigation Bar:** Home, Cart, Veg Pickles, Non-Veg Pickles, Snacks, Contact, order
- Form Section:** **Billing & Shipping Details**
  - Full Name: gnaneswar
  - Email: gnaneswar123@gmail.com
  - Phone Number: 9392318062
  - Shipping Address: 2-120,gujanagundla,guntur 522006
  - Additional Notes (optional): good
- Buttons:** Place Order
- System Status Bar:** 37°C, Mostly cloudy, Weather icon, Search icon, Taskbar icons (File Explorer, WhatsApp, Chrome, Notepad, Paint, File Manager, Task View, Visual Studio Code, Google Sheets), Language: ENG IN, Battery: 02:40 PM, Date: 11-07-2025.

## Success Page:



The screenshot shows a web browser window with the title "Order Placed - Homemade Pickles & Snacks". The URL in the address bar is "127.0.0.1:5000/order-success". The page has a green header with the text "Homemade Pickles & Snacks" and a navigation menu with links: Home, Veg Pickles, Non-Veg Pickles, Snacks, Contact, Cart, and Logout. The main content area features a large green circle with a white checkmark. Below it, the text "Order Placed Successfully!" is displayed. A message follows: "Thank you for your order! We have received your request and will process it shortly." Another message says: "You will receive a confirmation email with delivery details soon." A final message encourages action: "Get ready to taste authentic homemade flavors!". A blue button labeled "Continue Shopping" is visible at the bottom of this message area. The browser's taskbar at the bottom shows various open applications and system status icons.

**Your order has been placed successfully! We Will get back to you soon**

**Exit:**

## Session Ended

Please close [this tab](#).

### Conclusion:

The Homemade Pickles and Snacks Website has been successfully developed and deployed using a robust cloud-native architecture to ensure high performance, scalability, and customer satisfaction. By leveraging **AWS EC2** for reliable hosting, **DynamoDB** for secure product and order data management, and **SNS** for real-time customer and staff notifications, the platform delivers a seamless end-to-end experience for home made food lovers. This system addresses the need for efficient online access to traditional food offerings, allowing customers to conveniently browse, order, and track their favourite pickles and snacks. The cloud infrastructure enables the site to scale effortlessly during high-demand periods like festive seasons or promotions, without compromising responsiveness or user experience.

The integration of **Flask with AWS services** ensures that backend operations—such as order processing, inventory updates, and customer messaging—function smoothly in real time. Thorough testing has validated the stability of all core features, from product browsing and secure checkout to order confirmation notifications. In conclusion, this website serves as a modern platform that blends tradition with technology, providing a delightful shopping experience while streamlining business operations. It stands as a strong example of how cloud-based solutions can elevate local, handmade products to a broader digital audience.











