Home Made Pickles & Snacks: Taste the Best

Project Description:

Home Made Pickles & Snacks — Taste the Best is a cloud-based culinary platform revolutionizing access to authentic, handcrafted pickles and snacks. Addressing the growing demand for preservative-free, traditional recipes, this initiative combines artisanal craftsmanship with cutting-edge technology to deliver farm-fresh flavours 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

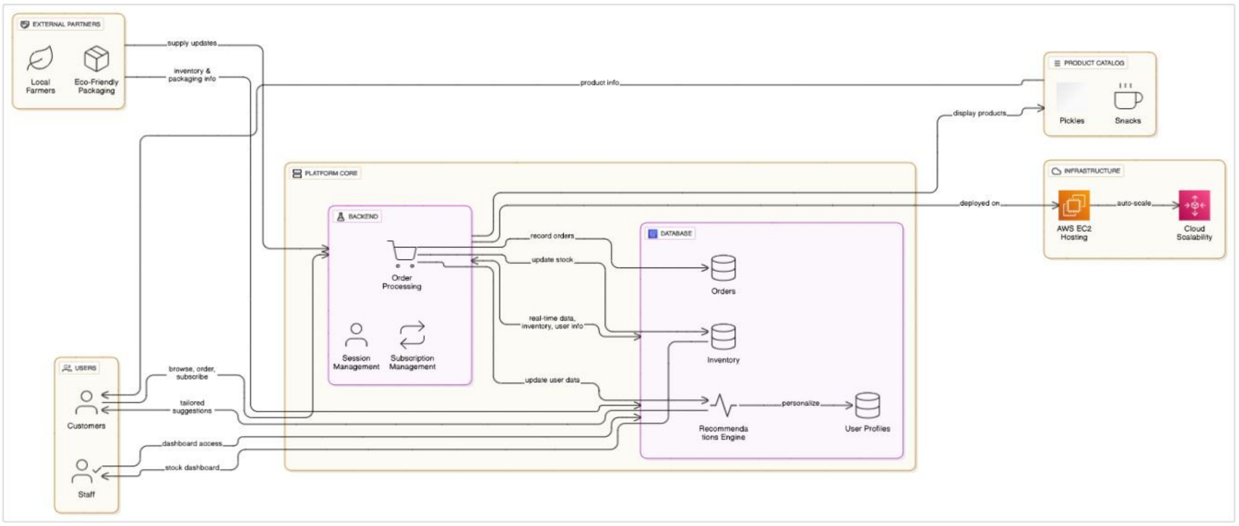
When a customer places an order for a product, the system instantly updates stock levels and records transaction details. For example, a user purchases an item, triggering automatic inventory deduction and order confirmation. Staff members receive updated dashboards to monitor stock availability and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

Scenario 3: Personalized User Experience and Recommendations

The platform leverages user behaviour 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.

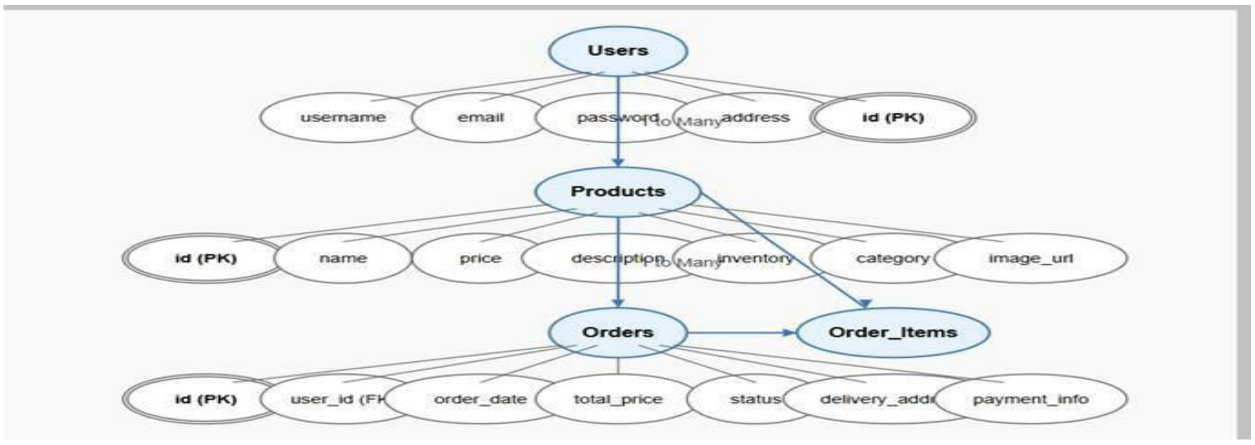
AWS ARCHITECTURE

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 Work Flow:

Milestone 1. Backend Development and Application Setup

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

Milestone 2. AWS Account Setup and Login

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

Milestone 3. DynamoDB Database Creation and Setup

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

Milestone 4. SNS Notification Setup

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

Milestone 5. IAM Role Setup

- Create IAM Role
- Attach Policies

Milestone 6. EC2 Instance Setup

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

Milestone 7. Deployment on EC2

- Upload Flask Files
- Run the Flask App

Milestone 8. Testing and Deployment

- Conduct functional testing to verify user signup

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.

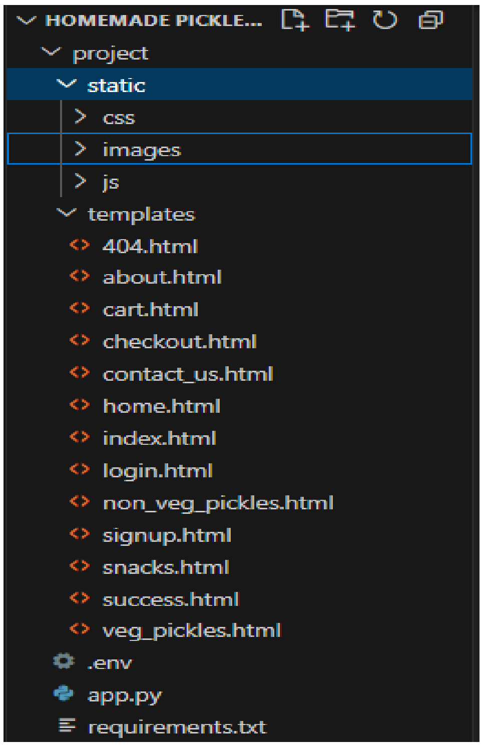
Post Troven Access Activation:

- 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



```
HOMEMADE PICKLE...
  project
    static
      css
      images
      js
    templates
      404.html
      about.html
      cart.html
      checkout.html
      contact_us.html
      home.html
      index.html
      login.html
      non_veg_pickles.html
      signup.html
      snacks.html
      success.html
      veg_pickles.html
    .env
    app.py
    requirements.txt
```

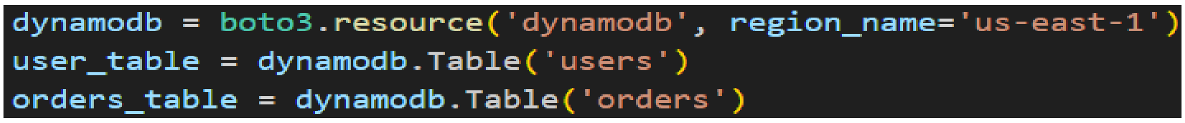
Description of the code:

- **Flask App Initialization**

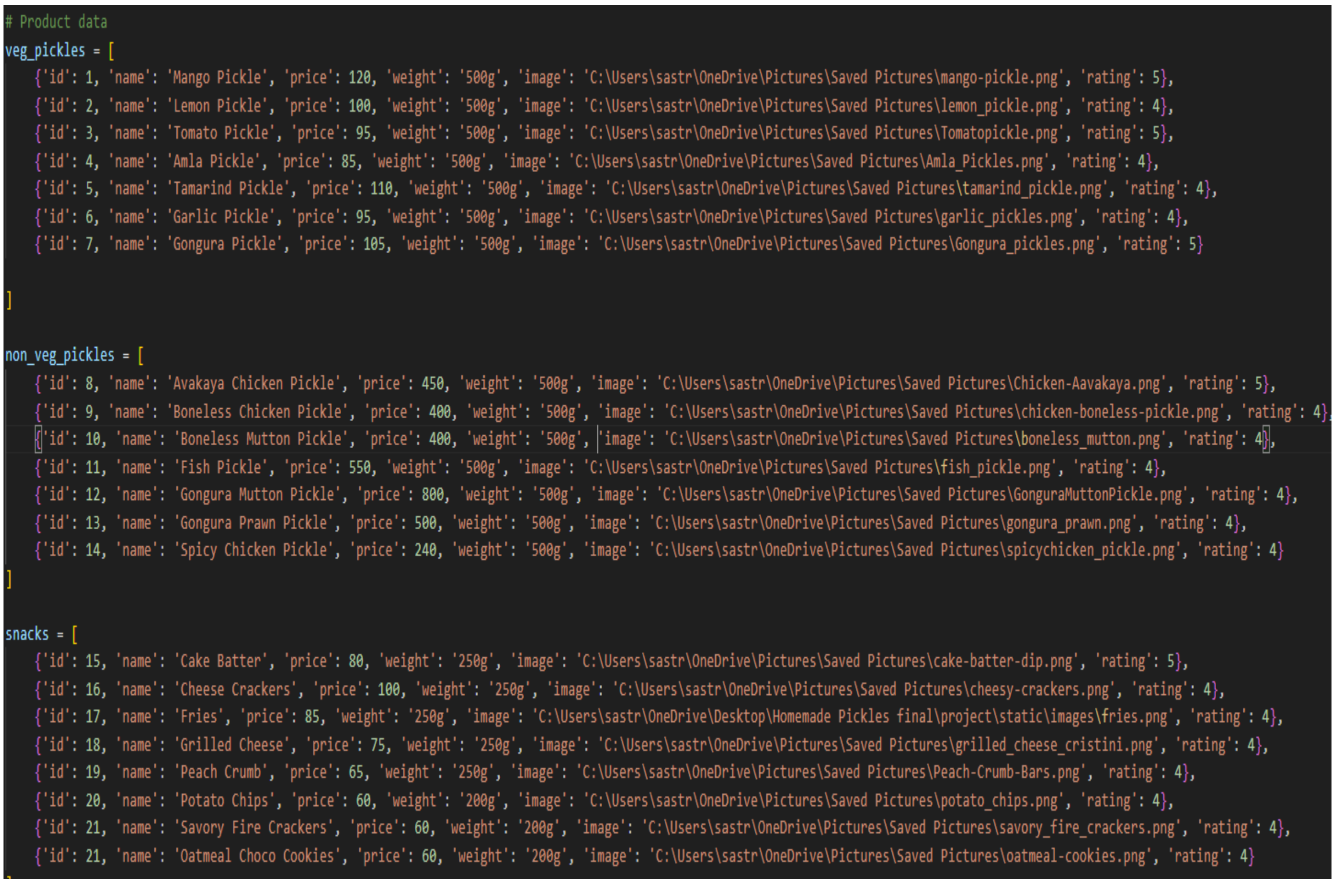
```
from flask import Flask, render_template, request, redirect, url_for, session, flash
import boto3
from boto3.dynamodb.conditions import Key, Attr
import uuid
from datetime import datetime
import smtplib
from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart
from bcrypt import hashpw, gensalt, checkpw
import os
from decimal import Decimal
```

```
app = Flask(__name__)
app.secret_key = os.urandom(24)
```

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

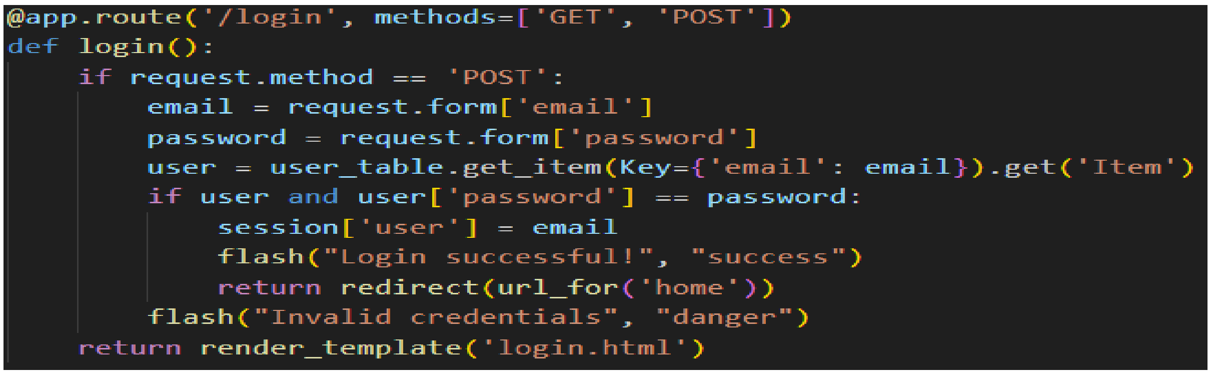


```python
dynamodb = boto3.resource('dynamodb', region_name='us-east-1')
user_table = dynamodb.Table('users')
orders_table = dynamodb.Table('orders')
```



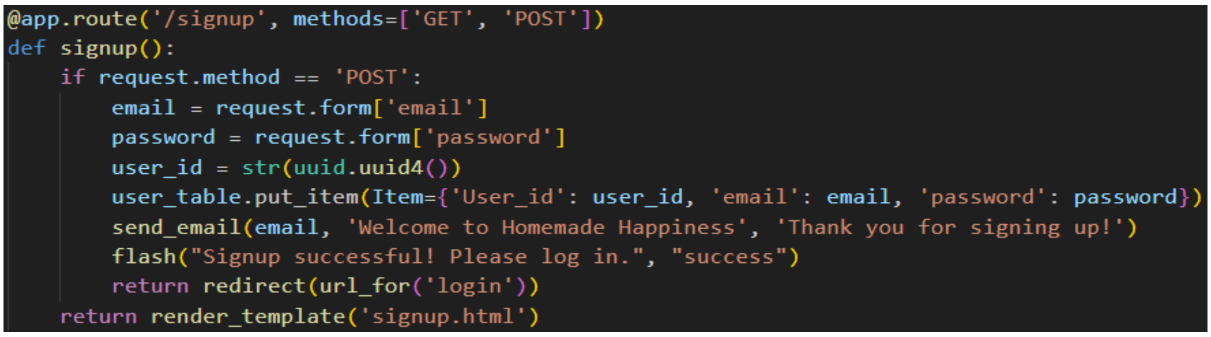
```python
# Product data
veg_pickles = [
    {'id': 1, 'name': 'Mango Pickle', 'price': 120, 'weight': '500g', 'image': 'C:\Users\sastr\OneDrive\Pictures\Saved Pictures\mango-pickle.png', 'rating': 5},
    {'id': 2, 'name': 'Lemon Pickle', 'price': 100, 'weight': '500g', 'image': 'C:\Users\sastr\OneDrive\Pictures\Saved Pictures\lemon_pickle.png', 'rating': 4},
    {'id': 3, 'name': 'Tomato Pickle', 'price': 95, 'weight': '500g', 'image': 'C:\Users\sastr\OneDrive\Pictures\Saved Pictures\Tomatopickle.png', 'rating': 5},
    {'id': 4, 'name': 'Amla Pickle', 'price': 85, 'weight': '500g', 'image': 'C:\Users\sastr\OneDrive\Pictures\Saved Pictures\Amla_Pickles.png', 'rating': 4},
    {'id': 5, 'name': 'Tamarind Pickle', 'price': 110, 'weight': '500g', 'image': 'C:\Users\sastr\OneDrive\Pictures\Saved Pictures\tamarind_pickle.png', 'rating': 4},
    {'id': 6, 'name': 'Garlic Pickle', 'price': 95, 'weight': '500g', 'image': 'C:\Users\sastr\OneDrive\Pictures\Saved Pictures\garlic_pickles.png', 'rating': 4},
    {'id': 7, 'name': 'Gongura Pickle', 'price': 105, 'weight': '500g', 'image': 'C:\Users\sastr\OneDrive\Pictures\Saved Pictures\Gongura_pickles.png', 'rating': 5}


]

non_veg_pickles = [
    {'id': 8, 'name': 'Avakaya Chicken Pickle', 'price': 450, 'weight': '500g', 'image': 'C:\Users\sastr\OneDrive\Pictures\Saved Pictures\Chicken-Aavakaya.png', 'rating': 5},
    {'id': 9, 'name': 'Boneless Chicken Pickle', 'price': 400, 'weight': '500g', 'image': 'C:\Users\sastr\OneDrive\Pictures\Saved Pictures\chicken-boneless-pickle.png', 'rating': 4},
    {'id': 10, 'name': 'Boneless Mutton Pickle', 'price': 400, 'weight': '500g', 'image': 'C:\Users\sastr\OneDrive\Pictures\Saved Pictures\boneless_mutton.png', 'rating': 4},
    {'id': 11, 'name': 'Fish Pickle', 'price': 550, 'weight': '500g', 'image': 'C:\Users\sastr\OneDrive\Pictures\Saved Pictures\fish_pickle.png', 'rating': 4},
    {'id': 12, 'name': 'Gongura Mutton Pickle', 'price': 800, 'weight': '500g', 'image': 'C:\Users\sastr\OneDrive\Pictures\Saved Pictures\GonguraMuttonPickle.png', 'rating': 4},
    {'id': 13, 'name': 'Gongura Prawn Pickle', 'price': 500, 'weight': '500g', 'image': 'C:\Users\sastr\OneDrive\Pictures\Saved Pictures\gongura_prawn.png', 'rating': 4},
    {'id': 14, 'name': 'Spicy Chicken Pickle', 'price': 240, 'weight': '500g', 'image': 'C:\Users\sastr\OneDrive\Pictures\Saved Pictures\spicychicken_pickle.png', 'rating': 4}
]

snacks = [
    {'id': 15, 'name': 'Cake Batter', 'price': 80, 'weight': '250g', 'image': 'C:\Users\sastr\OneDrive\Pictures\Saved Pictures\cake-batter-dip.png', 'rating': 5},
    {'id': 16, 'name': 'Cheese Crackers', 'price': 100, 'weight': '250g', 'image': 'C:\Users\sastr\OneDrive\Pictures\Saved Pictures\cheesy-crackers.png', 'rating': 4},
    {'id': 17, 'name': 'Fries', 'price': 85, 'weight': '250g', 'image': 'C:\Users\sastr\OneDrive\Desktop\Homemade Pickles final\project\static\images\fries.png', 'rating': 4},
    {'id': 18, 'name': 'Grilled Cheese', 'price': 75, 'weight': '250g', 'image': 'C:\Users\sastr\OneDrive\Pictures\Saved Pictures\grilled_cheese_cristini.png', 'rating': 4},
    {'id': 19, 'name': 'Peach Crumb', 'price': 65, 'weight': '250g', 'image': 'C:\Users\sastr\OneDrive\Pictures\Saved Pictures\Peach-Crumb-Bars.png', 'rating': 4},
    {'id': 20, 'name': 'Potato Chips', 'price': 60, 'weight': '200g', 'image': 'C:\Users\sastr\OneDrive\Pictures\Saved Pictures\potato_chips.png', 'rating': 4},
    {'id': 21, 'name': 'Savory Fire Crackers', 'price': 60, 'weight': '200g', 'image': 'C:\Users\sastr\OneDrive\Pictures\Saved Pictures\savory_fire_crackers.png', 'rating': 4},
    {'id': 21, 'name': 'Oatmeal Choco Cookies', 'price': 60, 'weight': '200g', 'image': 'C:\Users\sastr\OneDrive\Pictures\Saved Pictures\oatmeal-cookies.png', 'rating': 4}
```

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



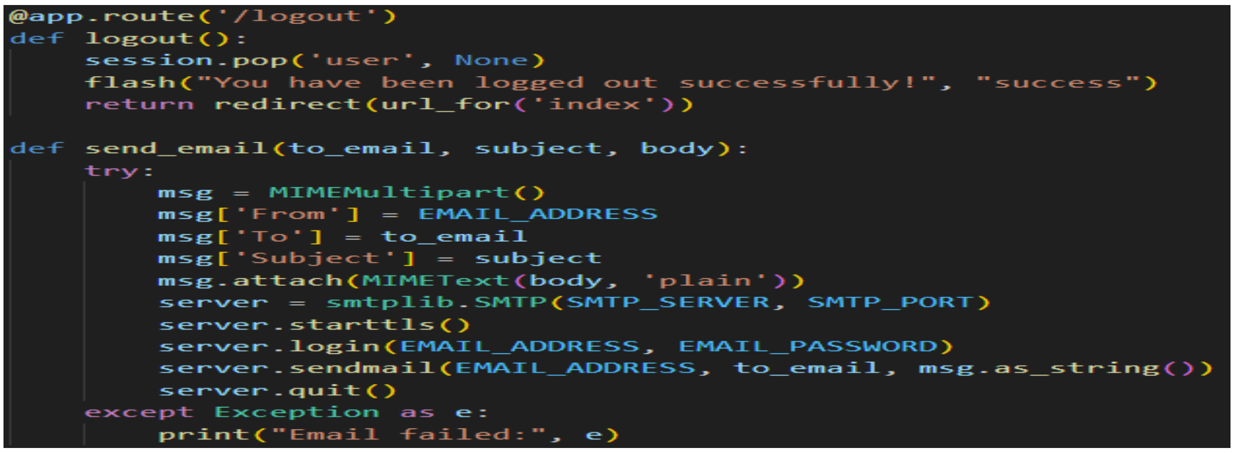
```python
@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        email = request.form['email']
        password = request.form['password']
        user = user_table.get_item(Key={'email': email}).get('Item')
        if user and user['password'] == password: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

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



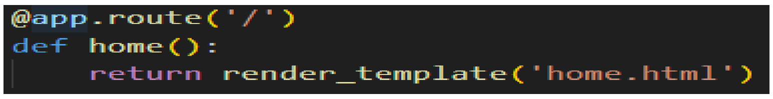
```python
@app.route('/signup', methods=['GET', 'POST'])
def signup():
    if request.method == 'POST':
        email = request.form['email']
        password = request.form['password']
        user_id = str(uuid.uuid4())
        user_table.put_item(Item={'User_id': user_id, 'email': email, 'password': password})
        send_email(email, 'Welcome to Homemade Happiness', 'Thank you for signing up!')
        flash("Signup successful! Please log in.", "success")
        return redirect(url_for('login'))
    return render_template('signup.html')
```

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



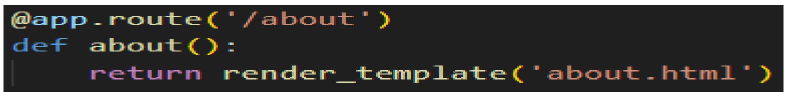
```python
@app.route('/logout')
def logout():
    session.pop('user', None)
    flash("You have been logged out successfully!", "success")
    return redirect(url_for('index'))

def send_email(to_email, subject, body):
    try:
        msg = MIMEMultipart()
        msg['From'] = EMAIL_ADDRESS
        msg['To'] = to_email
        msg['Subject'] = subject
        msg.attach(MIMEText(body, 'plain'))
        server = smtplib.SMTP(SMTP_SERVER, SMTP_PORT)
        server.starttls()
        server.login(EMAIL_ADDRESS, EMAIL_PASSWORD)
        server.sendmail(EMAIL_ADDRESS, to_email, msg.as_string())
        server.quit()
    except Exception as e:
        print("Email failed:", e)
```

- Home Route:



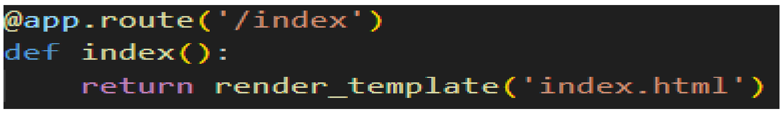
```python
@app.route('/')
def home():
    return render_template('home.html')
```

- About Route:



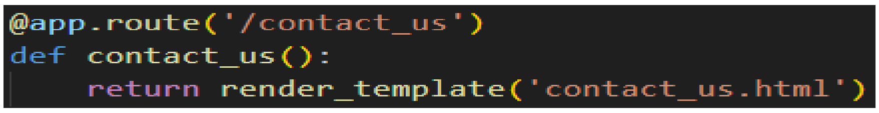
```python
@app.route('/about')
def about():
    return render_template('about.html')
```

- Index Route:



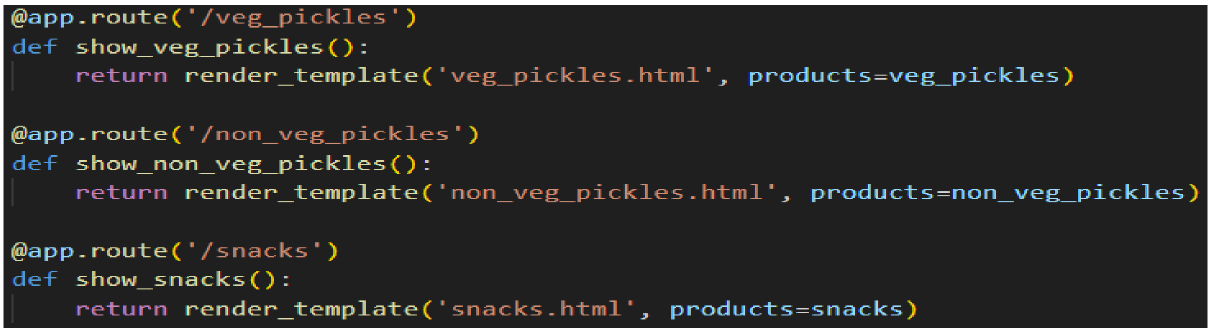
```python
@app.route('/index')
def index():
    return render_template('index.html')
```

- Contact_us Route:

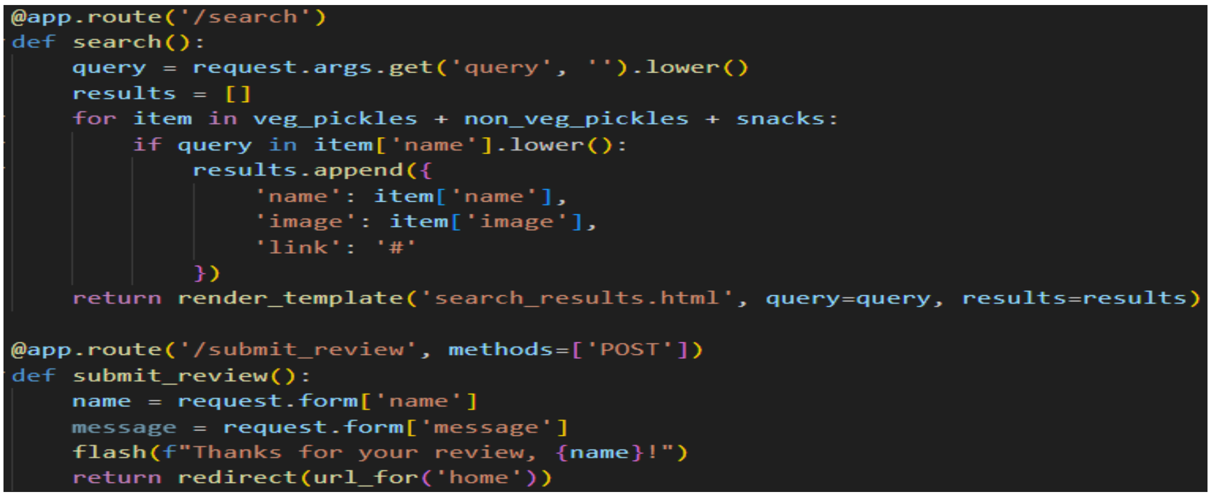


```python
@app.route('/contact_us')
def contact_us():
    return render_template('contact_us.html')
```

• Veg_Pickles, Non_Veg_Pickles, Snacks Route:

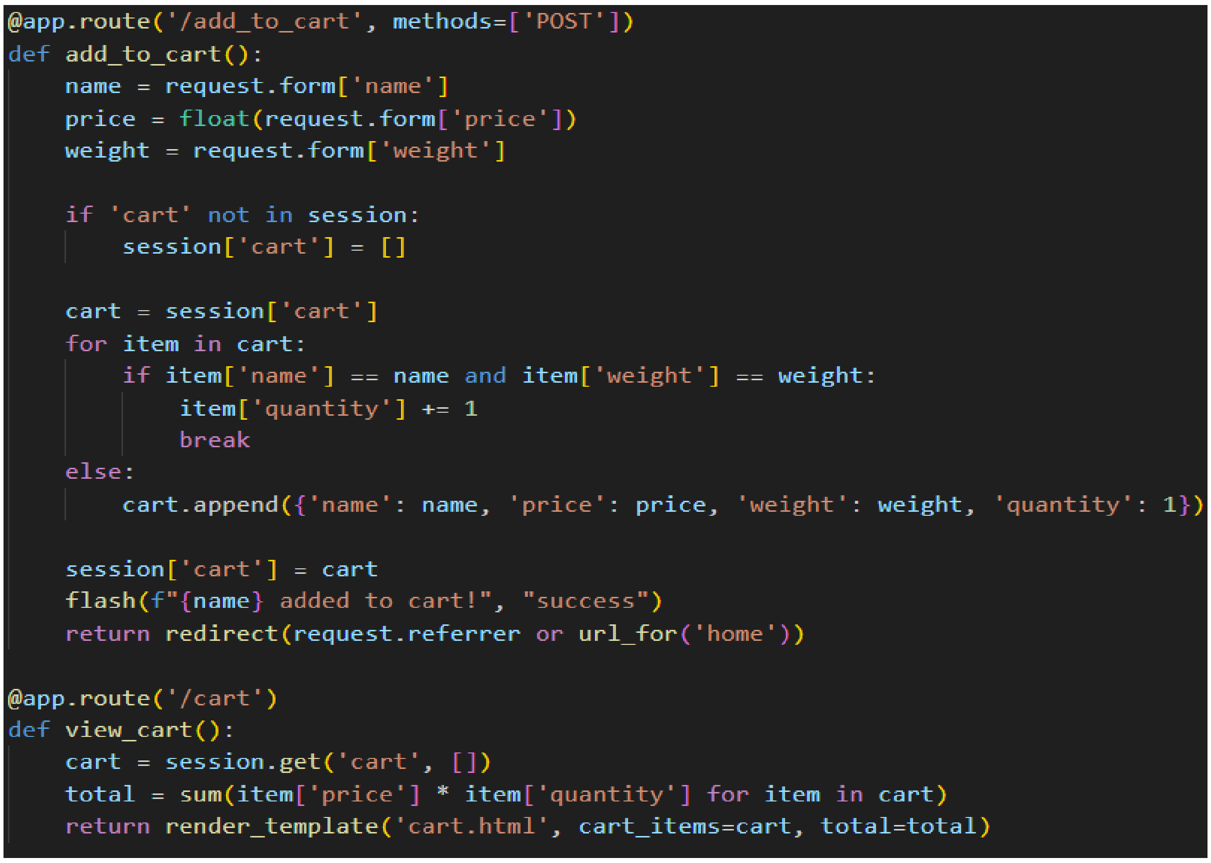


```python
@app.route('/veg_pickles')
def show_veg_pickles():
    return render_template('veg_pickles.html', products=veg_pickles)

@app.route('/non_veg_pickles')
def show_non_veg_pickles():
    return render_template('non_veg_pickles.html', products=non_veg_pickles)

@app.route('/snacks')
def show_snacks():
    return render_template('snacks.html', products=snacks)
```

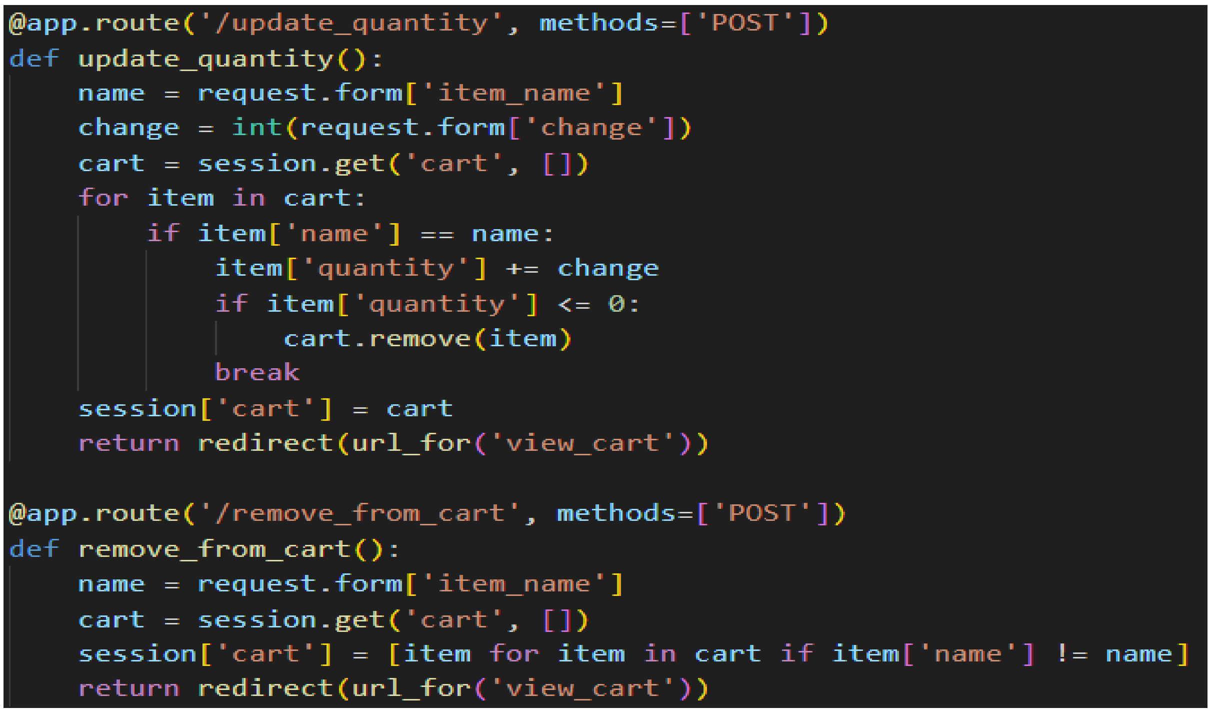


```python
@app.route('/search')
def search():
    query = request.args.get('query', '').lower()
    results = []
    for item in veg_pickles + non_veg_pickles + snacks:
        if query in item['name'].lower():
            results.append({
                'name': item['name'],
                'image': item['image'],
                'link': '#'
            })
    return render_template('search_results.html', query=query, results=results)

@app.route('/submit_review', methods=['POST'])
def submit_review():
    name = request.form['name']
    message = request.form['message']
    flash(f"Thanks for your review, {name}!")
    return redirect(url_for('home'))
```

• Cart Route:

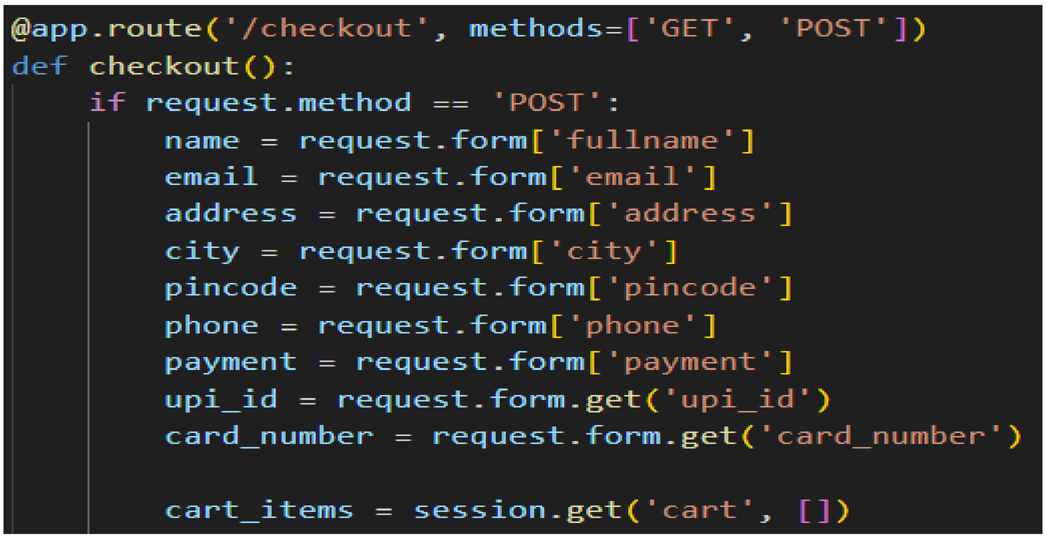


```python
@app.route('/add_to_cart', methods=['POST'])
def add_to_cart():
    name = request.form['name']
    price = float(request.form['price'])
    weight = request.form['weight']

    if 'cart' not in session:
        session['cart'] = []

    cart = session['cart']
    for item in cart:
        if item['name'] == name and item['weight'] == weight:
            item['quantity'] += 1
            break
    else:
        cart.append({'name': name, 'price': price, 'weight': weight, 'quantity': 1})

    session['cart'] = cart
    flash(f"{name} added to cart!", "success")
    return redirect(request.referrer or url_for('home'))

@app.route('/cart')
def view_cart():
    cart = session.get('cart', [])
    total = sum(item['price'] * item['quantity'] for item in cart)
    return render_template('cart.html', cart_items=cart, total=total)
```



```python
@app.route('/update_quantity', methods=['POST'])
def update_quantity():
    name = request.form['item_name']
    change = int(request.form['change'])
    cart = session.get('cart', [])
    for item in cart:
        if item['name'] == name:
            item['quantity'] += change
            if item['quantity'] <= 0:
                cart.remove(item)
            break
    session['cart'] = cart
    return redirect(url_for('view_cart'))

@app.route('/remove_from_cart', methods=['POST'])
def remove_from_cart():
    name = request.form['item_name']
    cart = session.get('cart', [])
    session['cart'] = [item for item in cart if item['name'] != name]
    return redirect(url_for('view_cart'))
```

• Checkout Route:



```python
@app.route('/checkout', methods=['GET', 'POST'])
def checkout():
    if request.method == 'POST':
        name = request.form['fullname']
        email = request.form['email']
        address = request.form['address']
        city = request.form['city']
        pincode = request.form['pincode']
        phone = request.form['phone']
        payment = request.form['payment']
        upi_id = request.form.get('upi_id')
        card_number = request.form.get('card_number')

        cart_items = session.get('cart', [])
```

Milestone 2: AWS Account Setup

Imortant 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.

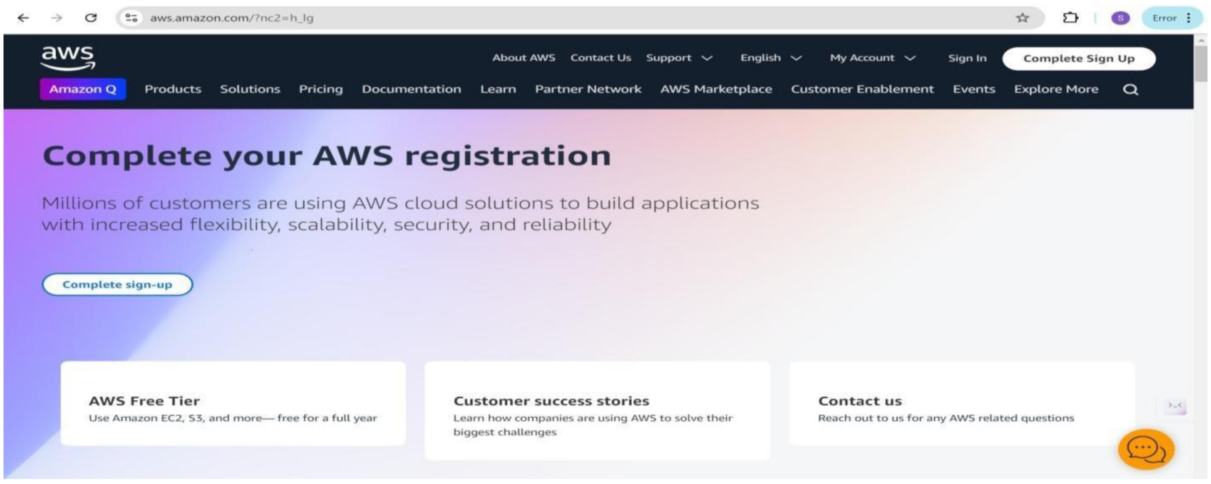
• Provide the required account information, including your name, address, and phone number.

• 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.



Complete your AWS registration

Millions of customers are using AWS cloud solutions to build applications with increased flexibility, scalability, security, and reliability

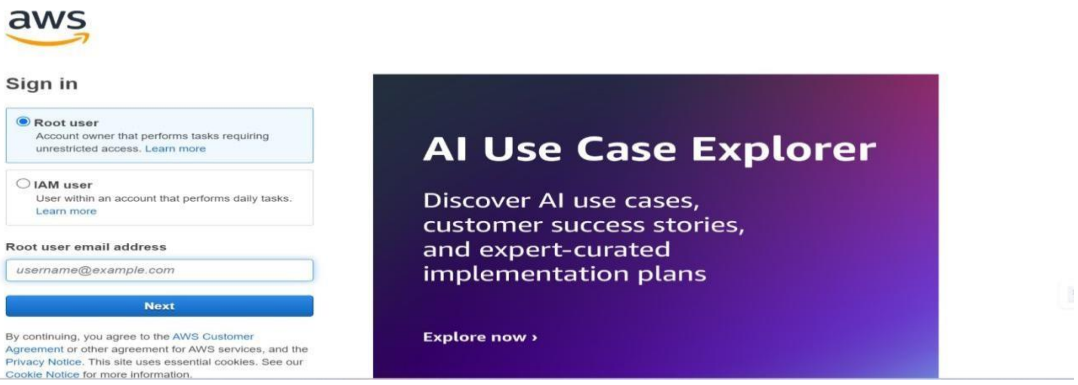
AWS Free Tier
Use Amazon EC2, S3, and more— free for a full year

Customer success stories
Learn how companies are using AWS to solve their biggest challenges

Contact us
Reach out to us for any AWS related questions

• Log in to the AWS Management Console

• After setting up your account, log in to the AWS Management Console.

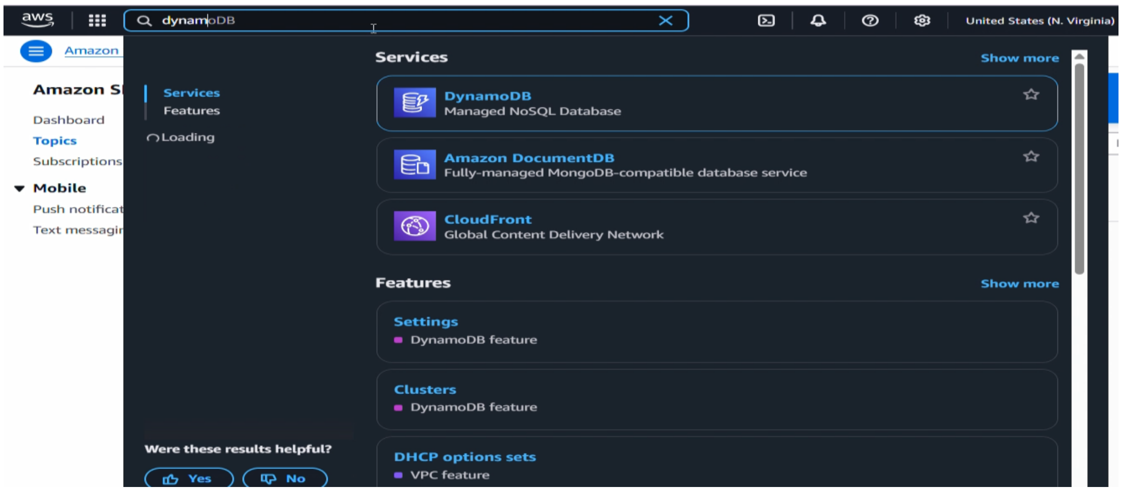

aws
Sign in
Root user
Account owner that performs tasks requiring unrestricted access. Learn more
IAM user
User within an account that performs daily tasks. Learn more
Root user email address
username@example.com
Next
By continuing, you agree to the AWS Customer Agreement or other agreement for AWS services, 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 ›

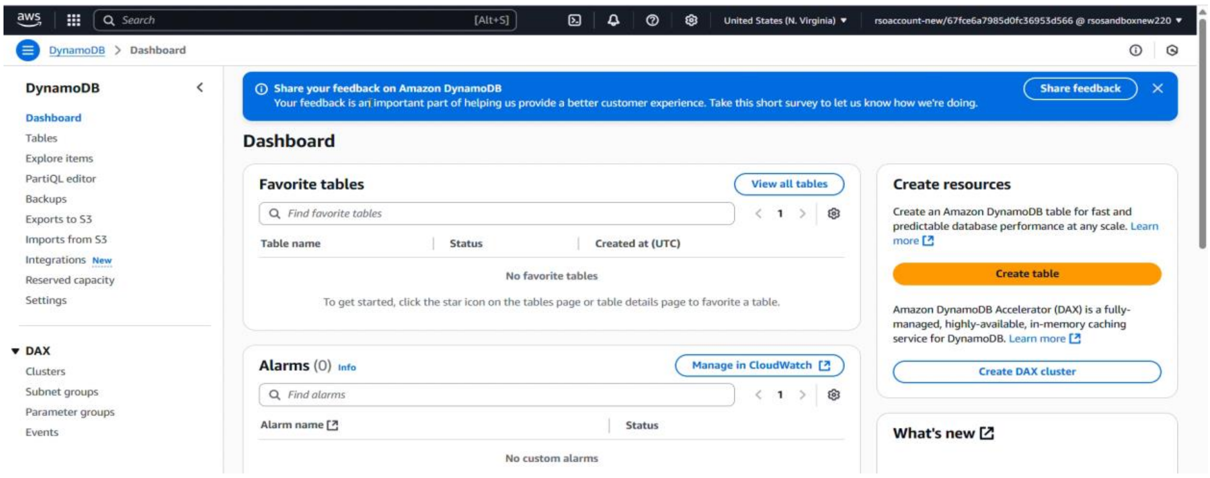
Milestone 3: DynamoDB Database Creation and Setup

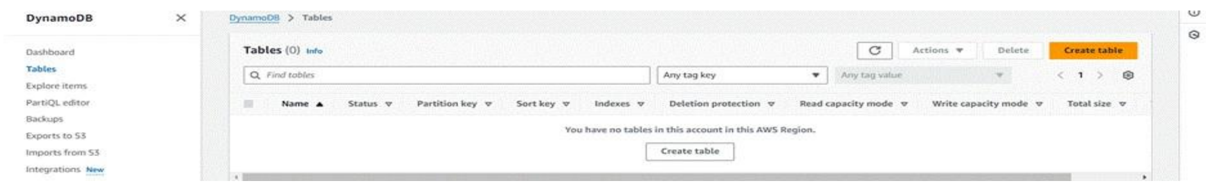
Database Creation and Setup involves initializing a cloud-based NoSQL database to store and manage application data efficiently. This step includes defining tables, setting primary keys, and configuring read/write capacities. It ensures scalable, high-performance data storage for seamless backend operations.

Navigate to the DynamoDB

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




aws
Search
[Alt+S]
United States (N. Virginia)
rsoaccount-new/67fce6a7985d0fc36953d566 @ rsosandboxnew220
DynamoDB
Dashboard
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
DynamoDB
Dashboard
Tables
Explore items
PartiQL editor
Backups
Exports to S3
Imports from S3
Integrations New
Reserved capacity
Settings
DAX
Clusters
Subnet groups
Parameter groups
Events
Dashboard
Favorite tables
View all tables
Find favorite tables
Table name
Status
Created at (UTC)
No favorite tables
To get started, click the star icon on the tables page or table details page to favorite a table.
Alarms (0) Info
Manage in CloudWatch
Find alarms
Alarm name
Status
No custom alarms
Create resources
Create an Amazon DynamoDB table for fast and predictable database performance at any scale. Learn more
Create table
Amazon DynamoDB Accelerator (DAX) is a fully-managed, highly-available, in-memory caching service for DynamoDB. Learn more
Create DAX cluster
What's new



DynamoDB

Dashboard
Tables
Explore items
PartiQL editor
Backups
Exports to S3
Imports from S3
Integrations New

DynamoDB > Tables

Tables (0) Info

Actions | Delete | Create table

Find tables | Any tag key | Any tag value

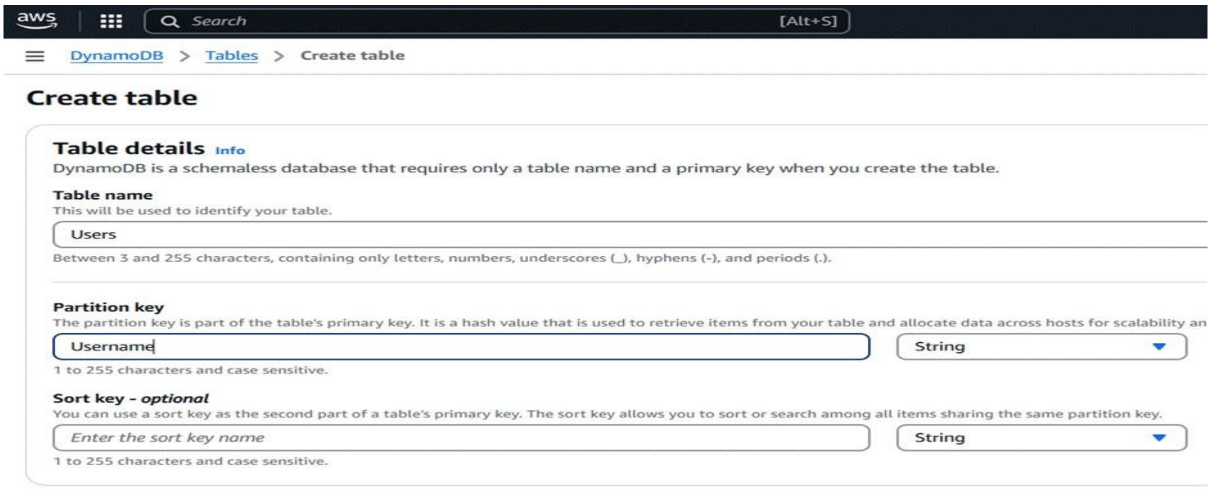
	Name	Status	Partition key	Sort key	Indexes	Deletion protection	Read capacity mode	Write capacity mode	Total size

You have no tables in this account in this AWS Region.

Create table

Create a DynamoDB table for storing data

• Create Users table with partition key “Username” with type String and click on create tables.



aws | Search [Alt+S]

DynamoDB > Tables > Create table

Create table

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

Table name
This will be used to identify your table.
Users
Between 3 and 255 characters, containing only letters, numbers, underscores (_), hyphens (-), and periods (.).

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

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

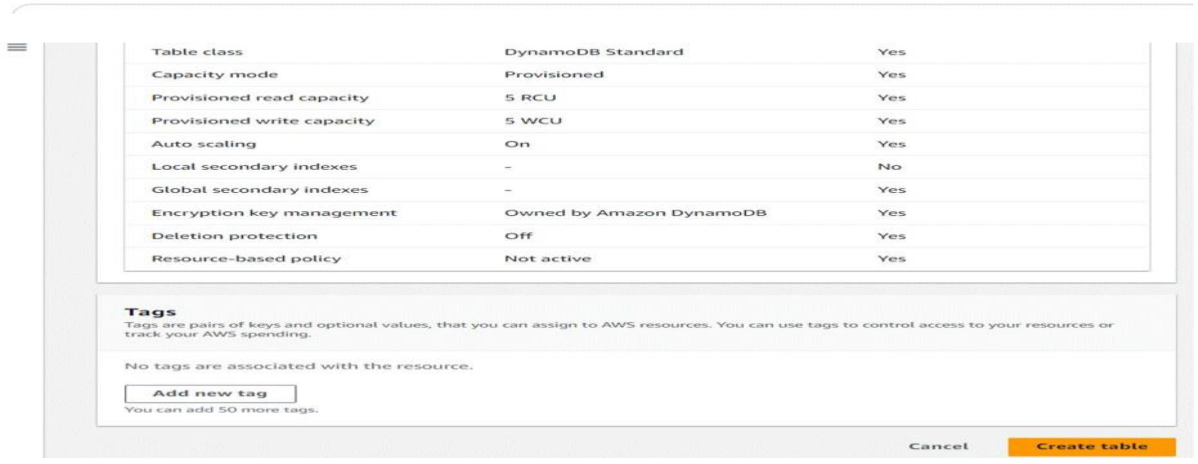


Table class	DynamoDB Standard	Yes
Capacity mode	Provisioned	Yes
Provisioned read capacity	5 RCU	Yes
Provisioned write capacity	5 WCU	Yes
Auto scaling	On	Yes
Local secondary indexes	-	No
Global secondary indexes	-	Yes
Encryption key management	Owned by Amazon DynamoDB	Yes
Deletion protection	Off	Yes
Resource-based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

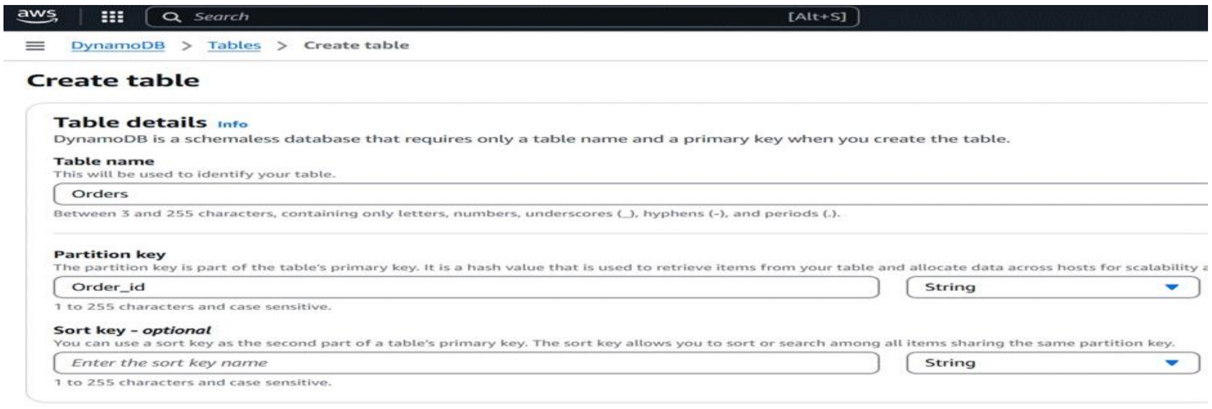
Tags
Tags are pairs of keys and optional values, that you can assign to AWS resources. You can use tags to control access to your resources or track your AWS spending.

No tags are associated with the resource.

Add new tag
You can add 50 more tags.

Cancel | Create table

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



aws | Search [Alt+S]

DynamoDB > Tables > Create table

Create table

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

Table name
This will be used to identify your table.
Orders
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 a
Order_id | String
1 to 255 characters and case sensitive.

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

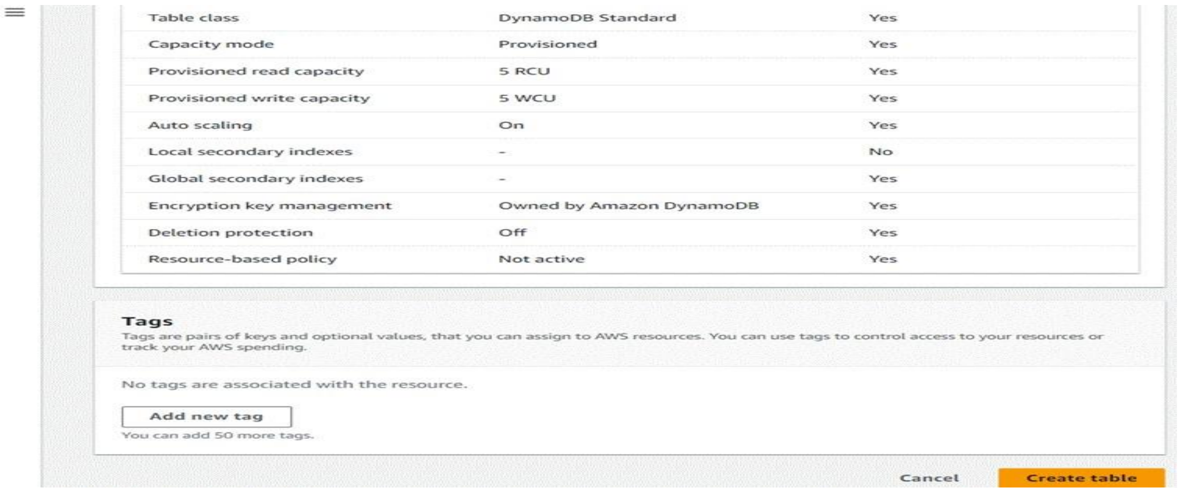


Table class	DynamoDB Standard	Yes
Capacity mode	Provisioned	Yes
Provisioned read capacity	5 RCU	Yes
Provisioned write capacity	5 WCU	Yes
Auto scaling	On	Yes
Local secondary indexes	-	No
Global secondary indexes	-	Yes
Encryption key management	Owned by Amazon DynamoDB	Yes
Deletion protection	Off	Yes
Resource-based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

Tags

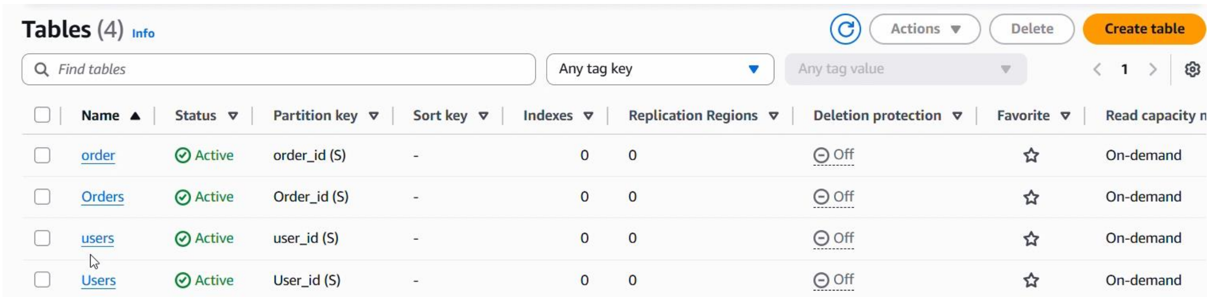
Tags are pairs of keys and optional values, that you can assign to AWS resources. You can use tags to control access to your resources or track your AWS spending.

No tags are associated with the resource.

Add new tag

You can add 50 more tags.

Cancel Create table



Tables (4) Info

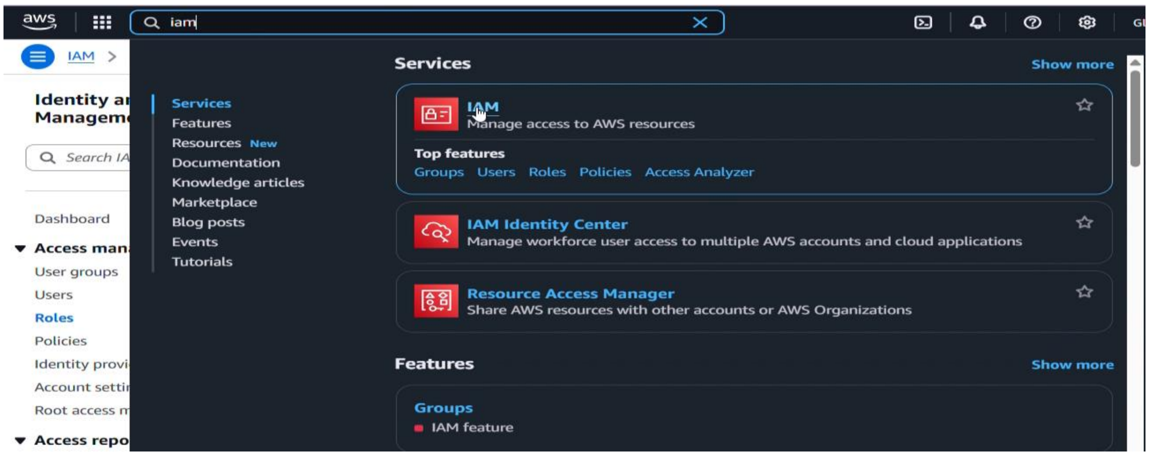
Name	Status	Partition key	Sort key	Indexes	Replication Regions	Deletion protection	Favorite	Read capacity m
order	Active	order_id (S)	-	0	0	Off	☆	On-demand
Orders	Active	Order_id (S)	-	0	0	Off	☆	On-demand
users	Active	user_id (S)	-	0	0	Off	☆	On-demand
Users	Active	User_id (S)	-	0	0	Off	☆	On-demand

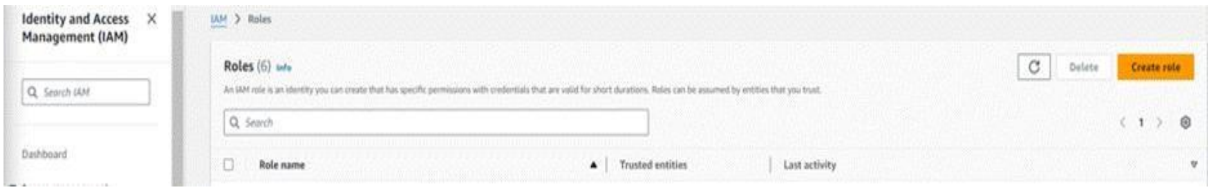
Milestone 4: IAM Role Setup

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.



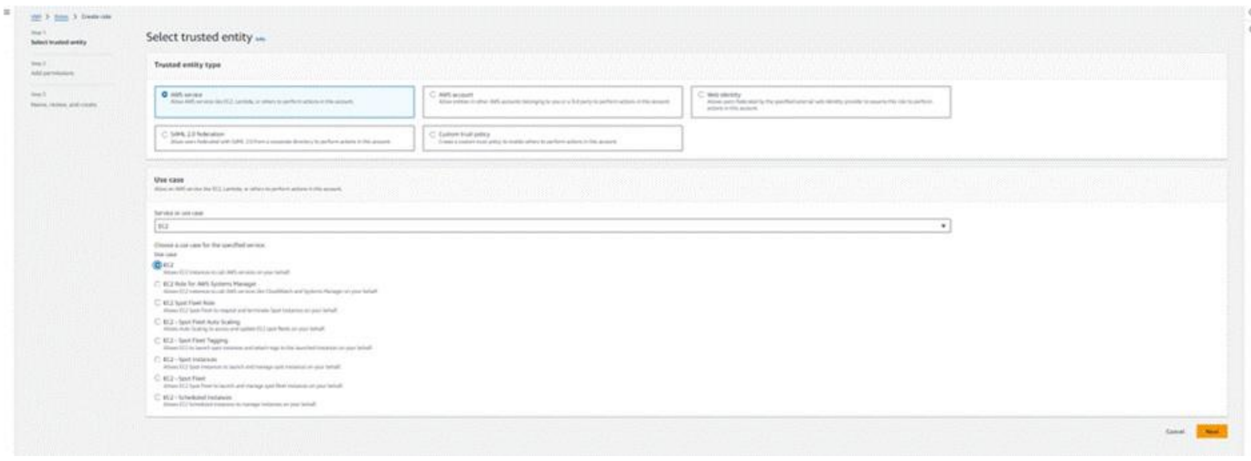

Identity and Access Management (IAM)

IAM > Roles

Roles (6) Info

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

Role name	Trusted entities	Last activity


Select trusted entity

Trusted entity type

[✓ AWS service] AWS account Web identity SAML 2.0 federation Custom trust policy

Use case

Service or use case: EC2

[✓ EC2]

EC2 Role for AWS Systems Manager

EC2 Spot Fleet Role

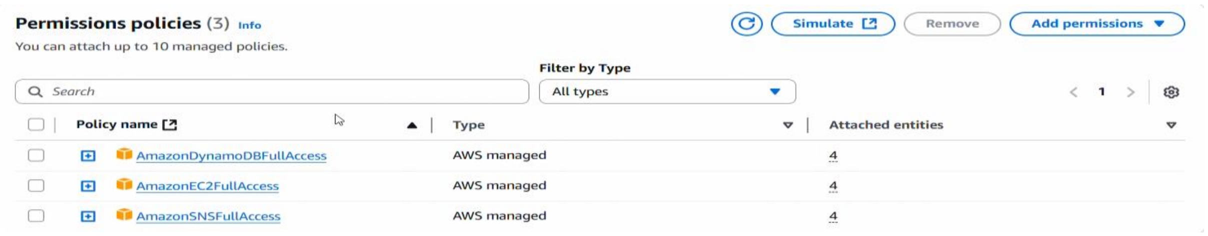
EC2 - Spot Fleet Auto Scaling

EC2 - Spot Fleet Tagging

EC2 - Spot Instances

EC2 - Spot Fleet

EC2 - Scheduled Instances


Permissions policies (3) Info

You can attach up to 10 managed policies.

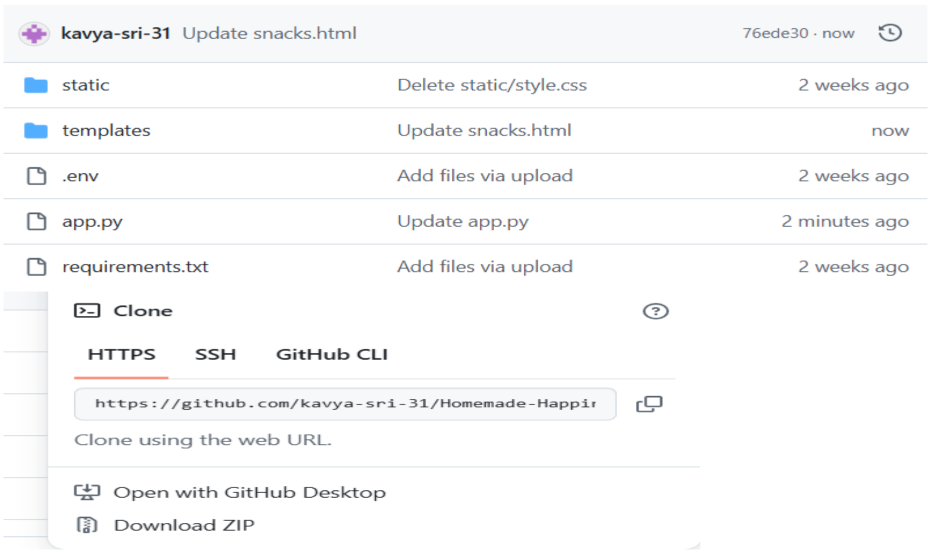
Filter by Type: All types

Policy name	Type	Attached entities
AmazonDynamoDBFullAccess	AWS managed	4
AmazonEC2FullAccess	AWS managed	4
AmazonSNSFullAccess	AWS managed	4

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.


kavya-sri-31 Update snacks.html 76ede30 · now

static	Delete static/style.css	2 weeks ago
templates	Update snacks.html	now
.env	Add files via upload	2 weeks ago
app.py	Update app.py	2 minutes ago
requirements.txt	Add files via upload	2 weeks ago

Clone

HTTPS SSH GitHub CLI

https://github.com/kavya-sri-31/Homemade-Happi

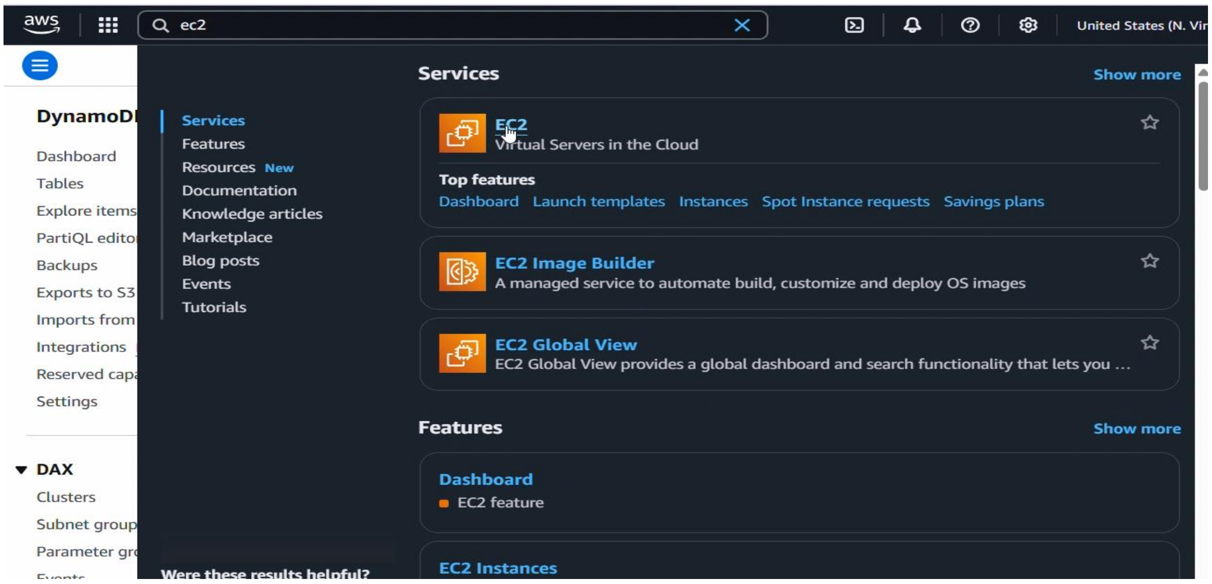
Clone using the web URL.

Open with GitHub Desktop

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.



Search: ec2

Services — Show more

EC2 — Virtual Servers in the Cloud
Top features: Dashboard, Launch templates, Instances, Spot Instance requests, Savings plans

EC2 Image Builder — A managed service to automate build, customize and deploy OS images

EC2 Global View — EC2 Global View provides a global dashboard and search functionality that lets you …

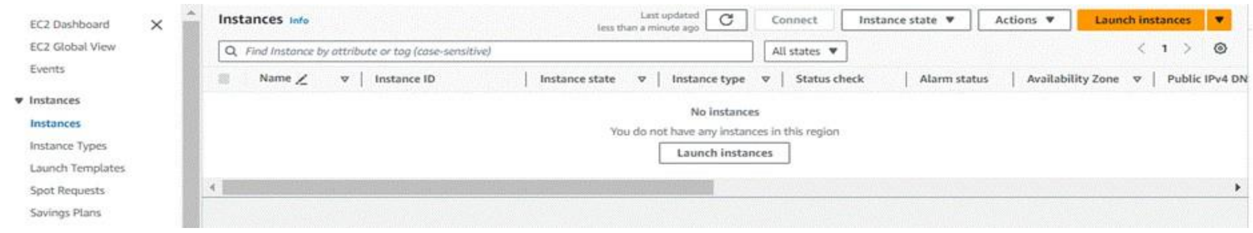
Features — Show more

Dashboard — EC2 feature

EC2 Instances

Were these results helpful?

• Click on Launch instance to launch EC2 instance

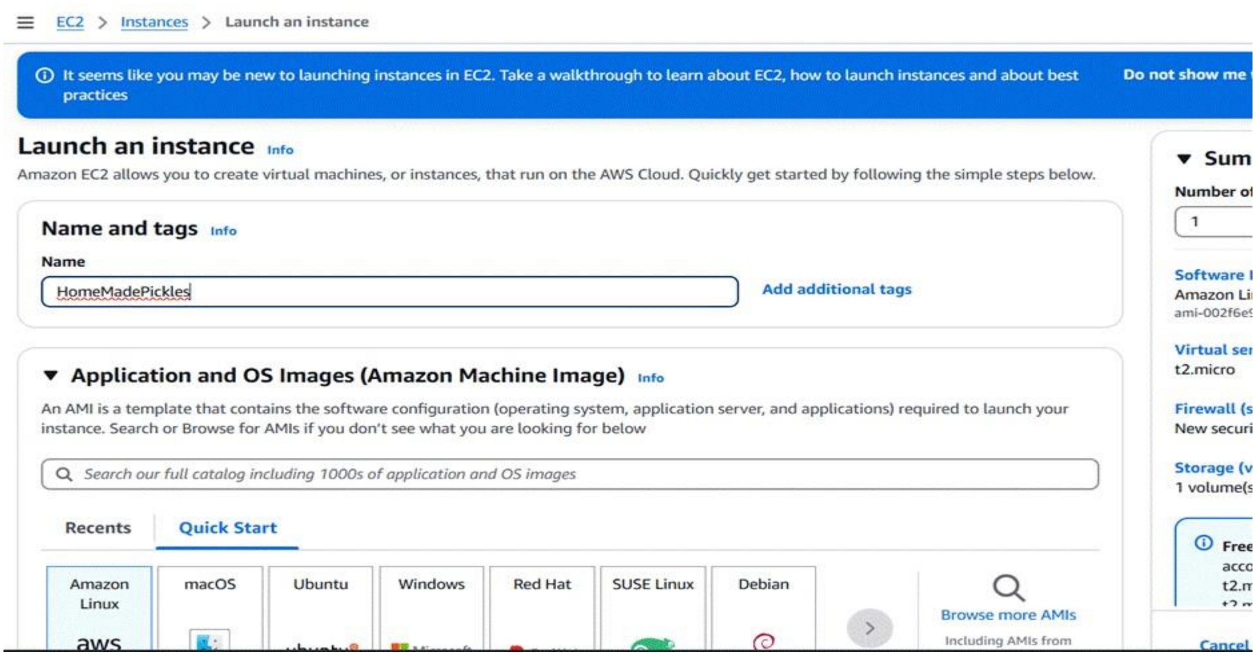


Instances Info — Last updated less than a minute ago

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

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DN

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

Launch an instance Info

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

Name and tags Info

Name: HomeMadePickles — Add additional tags

Application and OS Images (Amazon Machine Image) Info

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

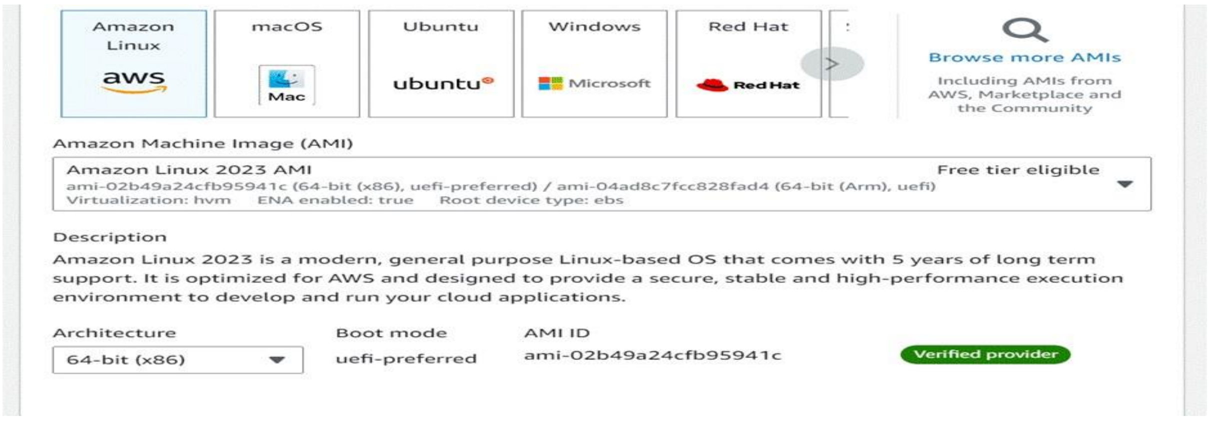
Search our full catalog including 1000s of application and OS images

Recents | Quick Start

Amazon Linux | macOS | Ubuntu | Windows | Red Hat | SUSE Linux | Debian | Browse more AMIs — Including AMIs from

Summary
Number of: 1
Software Amazon Li ami-002f6e
Virtual ser t2.micro
Firewall (s New securi
Storage (v 1 volume(s
Free acco t2.m
Cancel

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



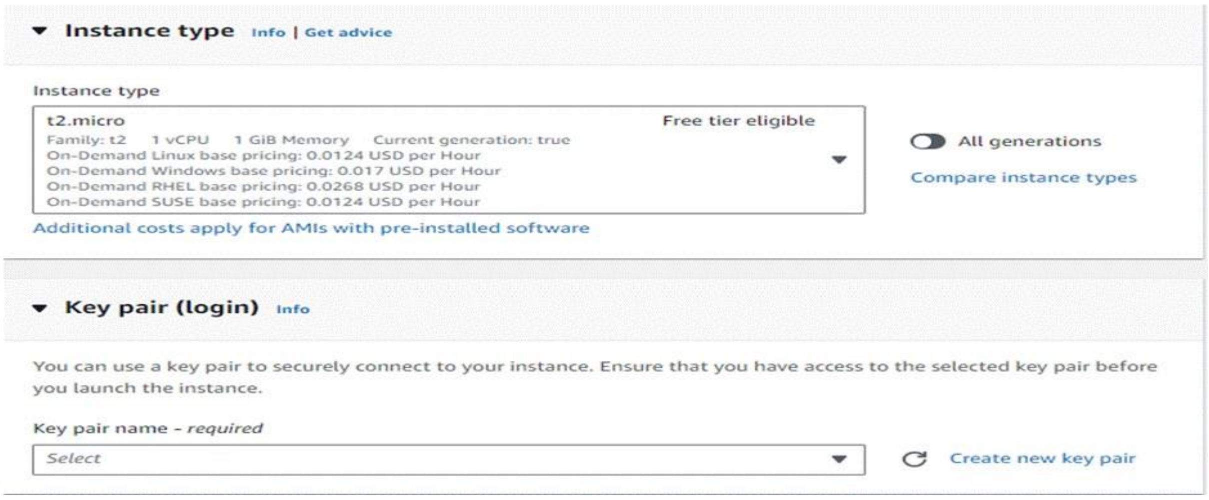
Amazon Machine Image (AMI)

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 Info | Get advice

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

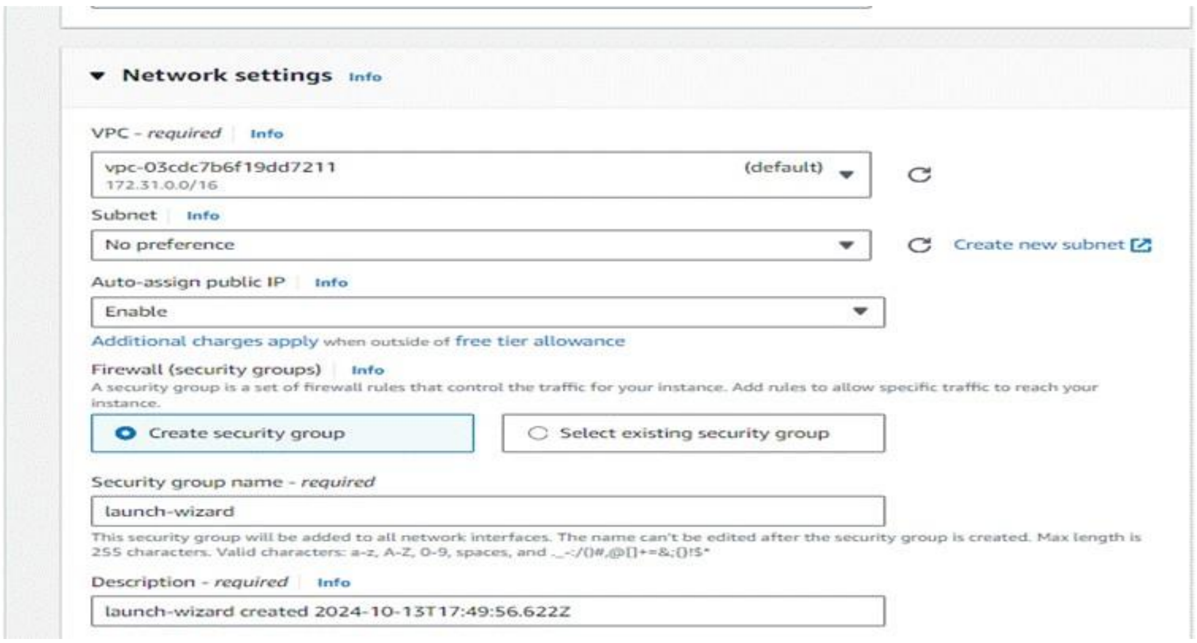
All generations
Compare instance types
Additional costs apply for AMIs with pre-installed software

Key pair (login) Info

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

Key pair name - *required*
Select
Create new key pair

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
Create new subnet

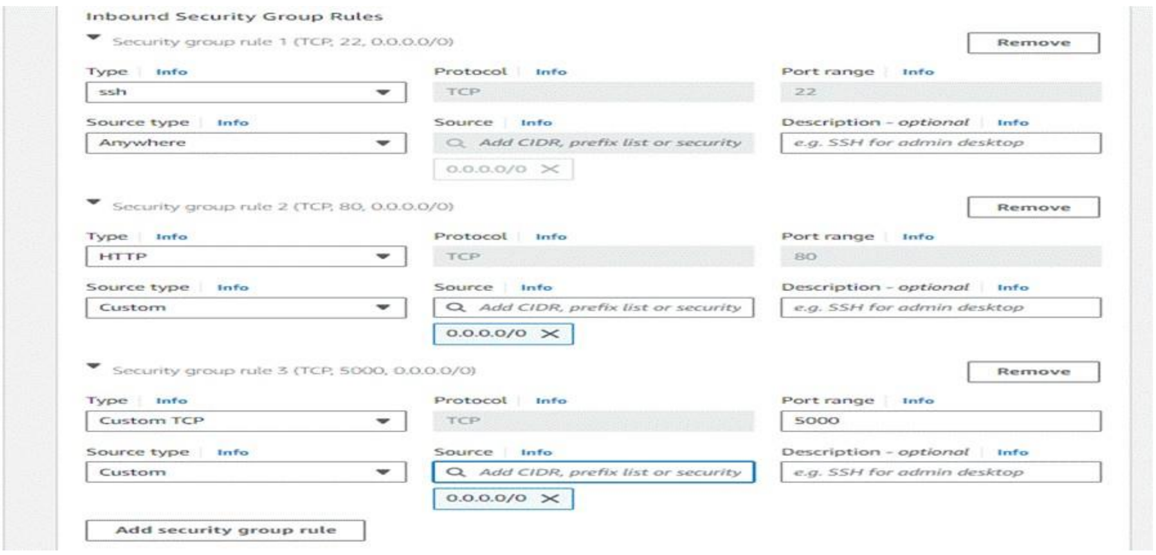
Auto-assign public IP Info
Enable
Additional charges apply when outside of free tier allowance

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

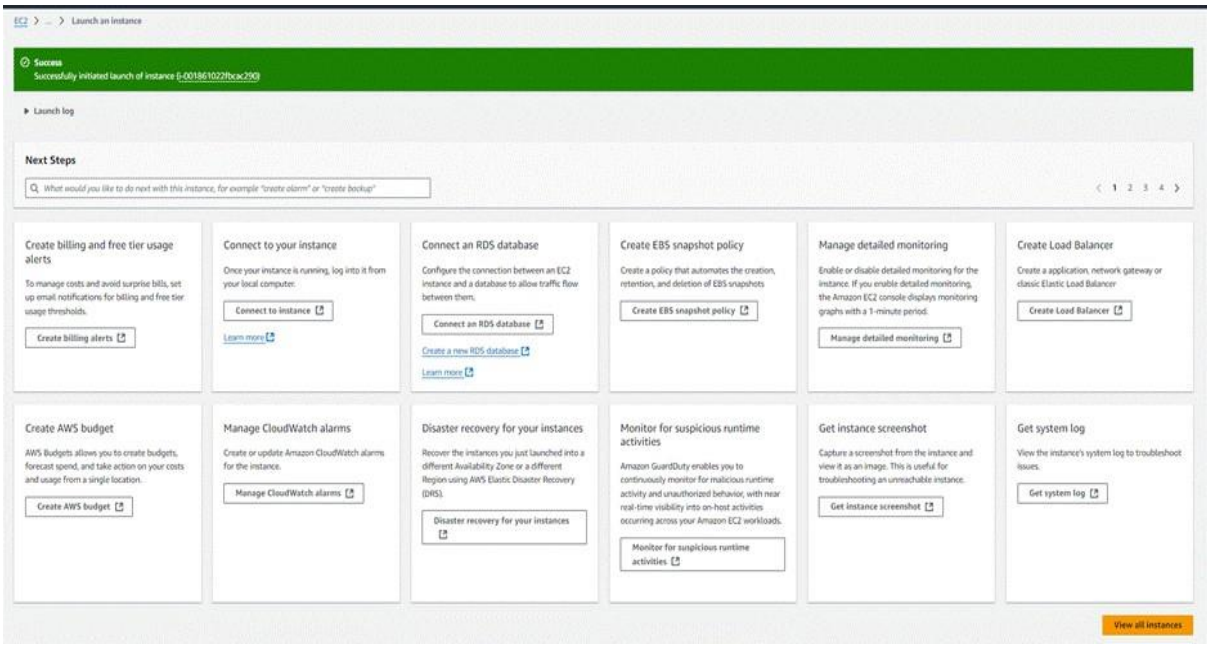

Inbound Security Group Rules

Security group rule 1 (TCP, 22, 0.0.0.0/0) — Remove
Type: ssh | Protocol: TCP | Port range: 22
Source type: Anywhere | Source: Add CIDR, prefix list or security — 0.0.0.0/0 | Description - optional: e.g. SSH for admin desktop

Security group rule 2 (TCP, 80, 0.0.0.0/0) — Remove
Type: HTTP | Protocol: TCP | Port range: 80
Source type: Custom | Source: Add CIDR, prefix list or security — 0.0.0.0/0 | Description - optional: e.g. SSH for admin desktop

Security group rule 3 (TCP, 5000, 0.0.0.0/0) — Remove
Type: Custom TCP | Protocol: TCP | Port range: 5000
Source type: Custom | Source: Add CIDR, prefix list or security — 0.0.0.0/0 | Description - optional: e.g. SSH for admin desktop

Add security group rule


EC2 > ... > Launch an instance

Success
Successfully initiated launch of instance (i-001861022fbcac290)

Launch log

Next Steps

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

Create billing and free tier usage alerts — To manage costs and avoid surprise bills, set up email notifications for billing and free tier usage thresholds. [Create billing alerts]

Connect to your instance — Once your instance is running, log into it from your local computer. [Connect to instance] Learn more

Connect an RDS database — Configure the connection between an EC2 instance and a database to allow traffic flow between them. [Connect an RDS database] Create a new RDS database. Learn more

Create EBS snapshot policy — Create a policy that automates the creation, retention, and deletion of EBS snapshots [Create EBS snapshot policy]

Manage detailed monitoring — Enable or disable detailed monitoring for the instance. If you enable detailed monitoring, the Amazon EC2 console displays monitoring graphs with a 1-minute period. [Manage detailed monitoring]

Create Load Balancer — Create a application, network gateway or classic Elastic Load Balancer [Create Load Balancer]

Create AWS budget — AWS Budgets allows you to create budgets, forecast spend, and take action on your costs and usage from a single location. [Create AWS budget]

Manage CloudWatch alarms — Create or update Amazon CloudWatch alarms for the instance. [Manage CloudWatch alarms]

Disaster recovery for your instances — Recover the instances you just launched into a different Availability Zone or a different Region using AWS Elastic Disaster Recovery (DRS). [Disaster recovery for your instances]

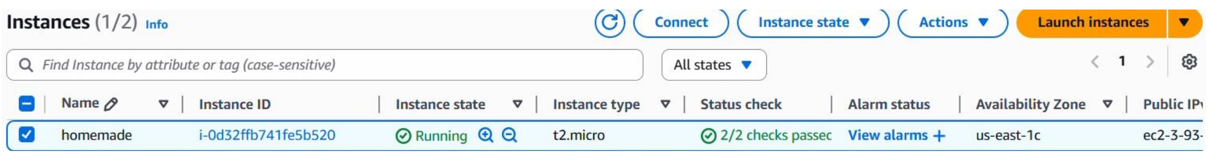
Monitor for suspicious runtime activities — Amazon GuardDuty enables you to continuously monitor for malicious runtime activity and unauthorized behavior, with near real-time visibility into on-host activities occurring across your Amazon EC2 workloads. [Monitor for suspicious runtime activities]

Get instance screenshot — Capture a screenshot from the instance and view it as an image. This is useful for troubleshooting an unreachable instance. [Get instance screenshot]

Get system log — View the instance's system log to troubleshoot issues. [Get system log]

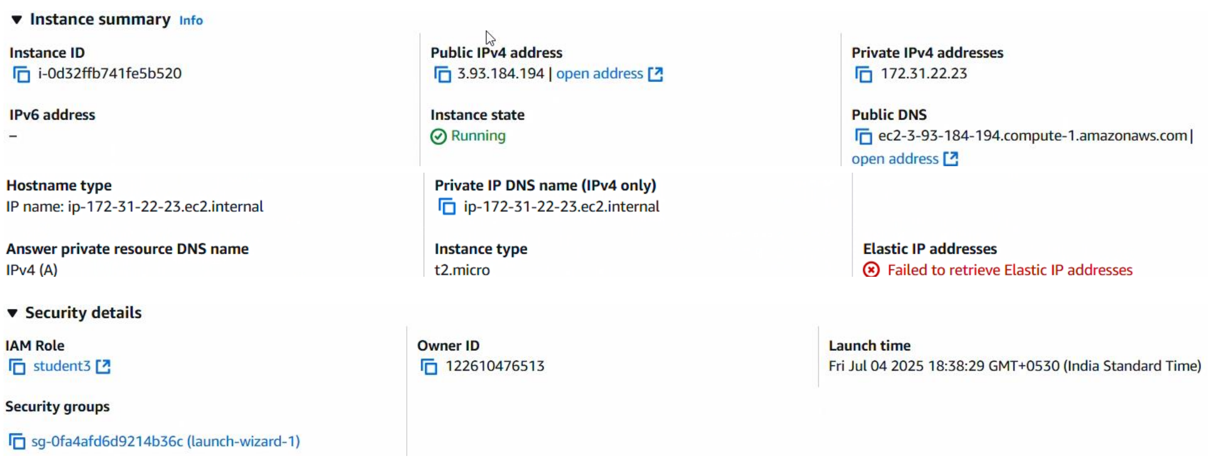
View all instances

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


Instances (1/2) Info — Connect | Instance state | Actions | Launch instances

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

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IP
homemade	i-0d32ffb741fe5b520	Running	t2.micro	2/2 checks passec	View alarms +	us-east-1c	ec2-3-93-

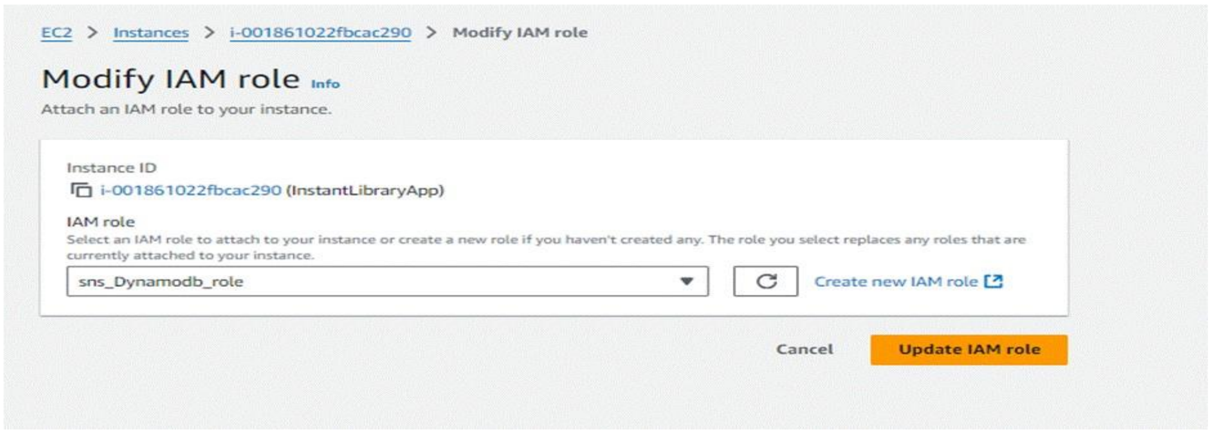


▼ Instance summary Info

Instance ID i-0d32ffb741fe5b520	Public IPv4 address 3.93.184.194 | open address	Private IPv4 addresses 172.31.22.23
IPv6 address –	Instance state Running	Public DNS ec2-3-93-184-194.compute-1.amazonaws.com | open address
Hostname type IP name: ip-172-31-22-23.ec2.internal	Private IP DNS name (IPv4 only) ip-172-31-22-23.ec2.internal	
Answer private resource DNS name IPv4 (A)	Instance type t2.micro	Elastic IP addresses Failed to retrieve Elastic IP addresses

▼ Security details

IAM Role student3	Owner ID 122610476513	Launch time Fri Jul 04 2025 18:38:29 GMT+0530 (India Standard Time)
Security groups sg-0fa4afd6d9214b36c (launch-wizard-1)		



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

Modify IAM role Info

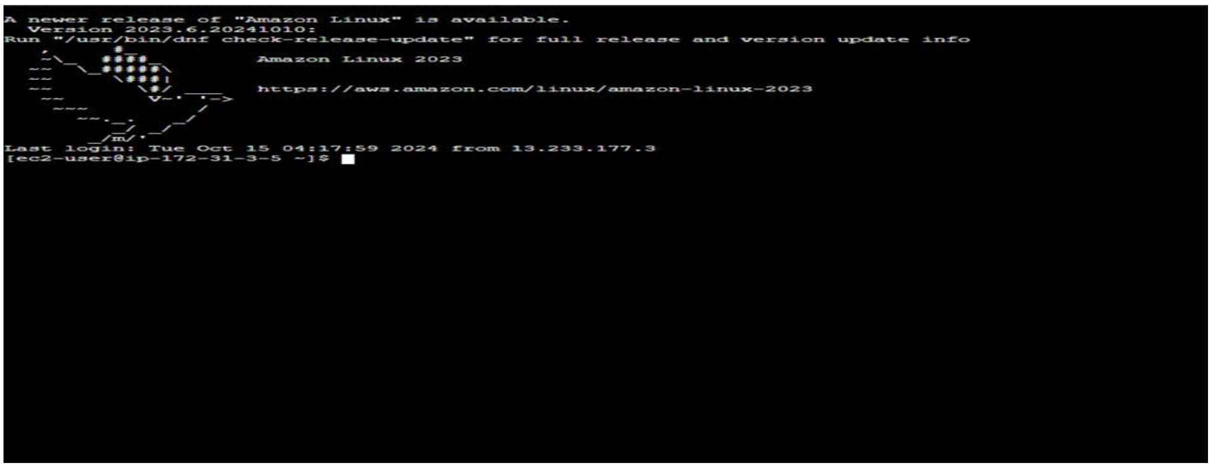
Attach an IAM role to your instance.

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.

sns_Dynamodb_role

• Now connect the EC2 with the files



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

Milestone 6: Deployment on EC2

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/kavya-sri-31/Homemade-Happiness.git”
- Note: change your-github-username and your-repository-name with your credentials here: “https://github.com/kavya-sri-31/Homemade-Happiness.git
- This will download your project to the EC2 instance.

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

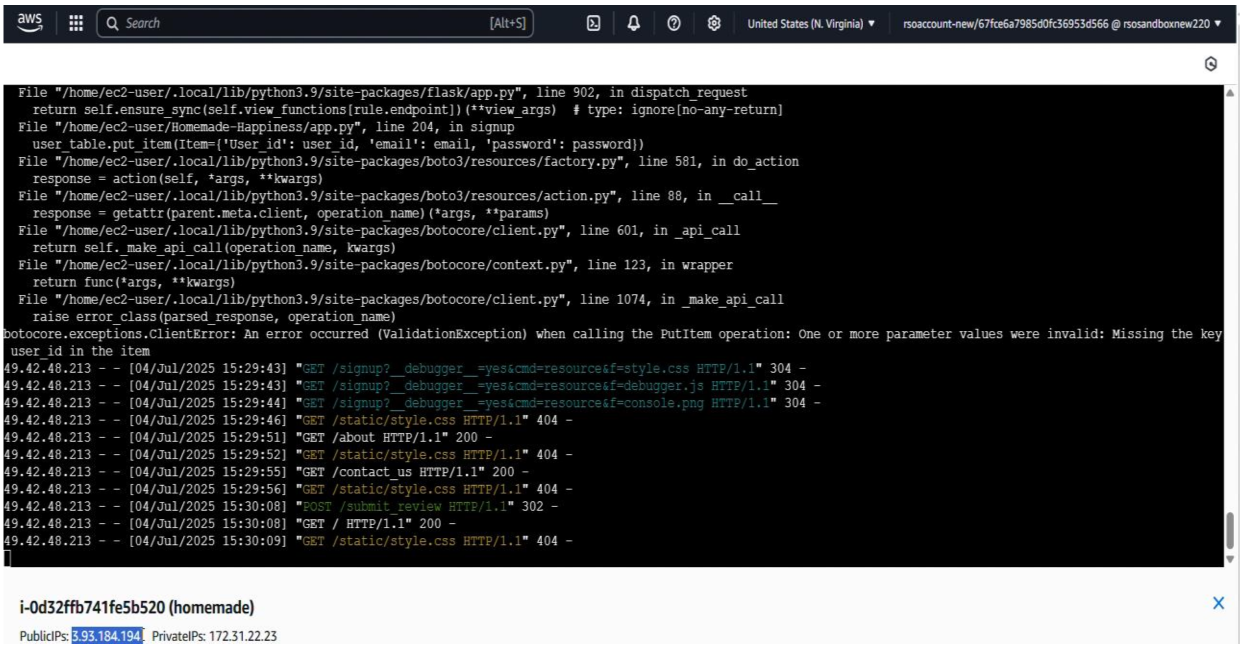
- cd Homemadepicklesandsnacks

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
- Run the Flask app on the EC2 instance


aws
Search
[Alt+S]
United States (N. Virginia) ▼
rsoaccount-new/67fce6a7985d0fc36953d566 @ rsosandboxnew220 ▼

```
  File "/home/ec2-user/.local/lib/python3.9/site-packages/flask/app.py", line 902, in dispatch_request
    return self.ensure_sync(self.view_functions[rule.endpoint])(**view_args)  # type: ignore[no-any-return]
  File "/home/ec2-user/Homemade-Happiness/app.py", line 204, in signup
    user_table.put_item(Item={'User_id': user_id, 'email': email, 'password': password})
  File "/home/ec2-user/.local/lib/python3.9/site-packages/boto3/resources/factory.py", line 581, in do_action
    response = action(self, *args, **kwargs)
  File "/home/ec2-user/.local/lib/python3.9/site-packages/boto3/resources/action.py", line 88, in __call__
    response = getattr(parent.meta.client, operation_name)(*args, **params)
  File "/home/ec2-user/.local/lib/python3.9/site-packages/botocore/client.py", line 601, in _api_call
    return self._make_api_call(operation_name, kwargs)
  File "/home/ec2-user/.local/lib/python3.9/site-packages/botocore/context.py", line 123, in wrapper
    return func(*args, **kwargs)
  File "/home/ec2-user/.local/lib/python3.9/site-packages/botocore/client.py", line 1074, in _make_api_call
    raise error_class(parsed_response, operation_name)
botocore.exceptions.ClientError: An error occurred (ValidationException) when calling the PutItem operation: One or more parameter values were invalid: Missing the key
 user_id in the item
49.42.48.213 - - [04/Jul/2025 15:29:43] "GET /signup?__debugger__=yes&cmd=resource&f=style.css HTTP/1.1" 304 -
49.42.48.213 - - [04/Jul/2025 15:29:43] "GET /signup?__debugger__=yes&cmd=resource&f=debugger.js HTTP/1.1" 304 -
49.42.48.213 - - [04/Jul/2025 15:29:44] "GET /signup?__debugger__=yes&cmd=resource&f=console.png HTTP/1.1" 304 -
49.42.48.213 - - [04/Jul/2025 15:29:46] "GET /static/style.css HTTP/1.1" 404 -
49.42.48.213 - - [04/Jul/2025 15:29:51] "GET /about HTTP/1.1" 200 -
49.42.48.213 - - [04/Jul/2025 15:29:52] "GET /static/style.css HTTP/1.1" 404 -
49.42.48.213 - - [04/Jul/2025 15:29:55] "GET /contact_us HTTP/1.1" 200 -
49.42.48.213 - - [04/Jul/2025 15:29:56] "GET /static/style.css HTTP/1.1" 404 -
49.42.48.213 - - [04/Jul/2025 15:30:08] "POST /submit_review HTTP/1.1" 302 -
49.42.48.213 - - [04/Jul/2025 15:30:08] "GET / HTTP/1.1" 200 -
49.42.48.213 - - [04/Jul/2025 15:30:09] "GET /static/style.css HTTP/1.1" 404 -
```

i-0d32ffb741fe5b520 (homemade)

PublicIPs: 3.93.184.194 PrivateIPs: 172.31.22.23

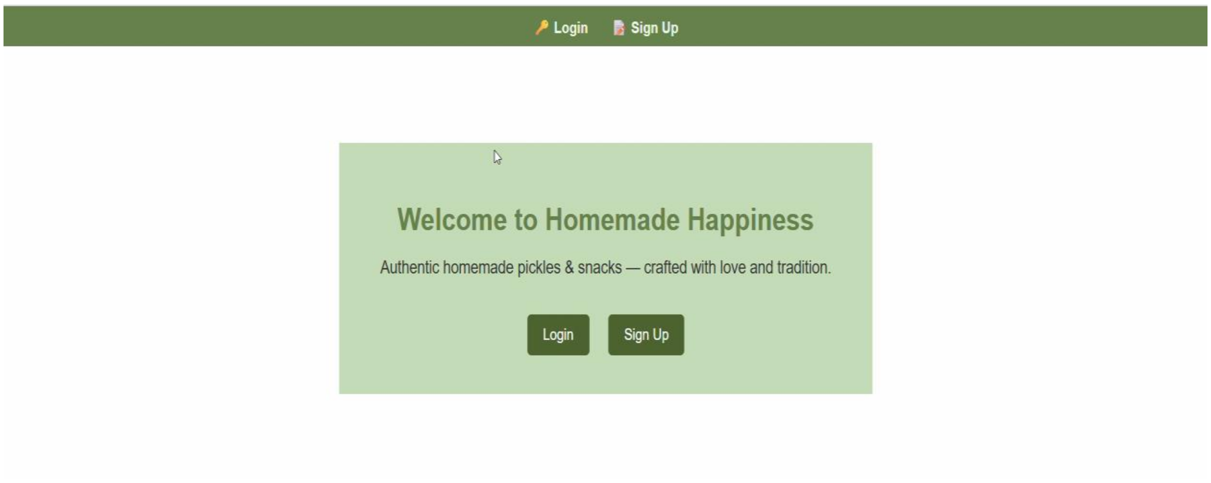
Access the website through: **PublicIPs**: http://3.93.184.194:5000

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

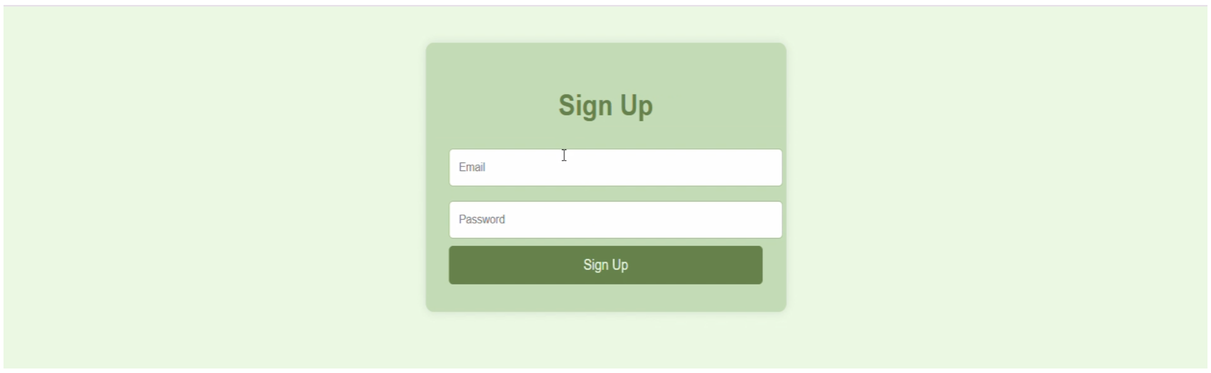
Welcome page:



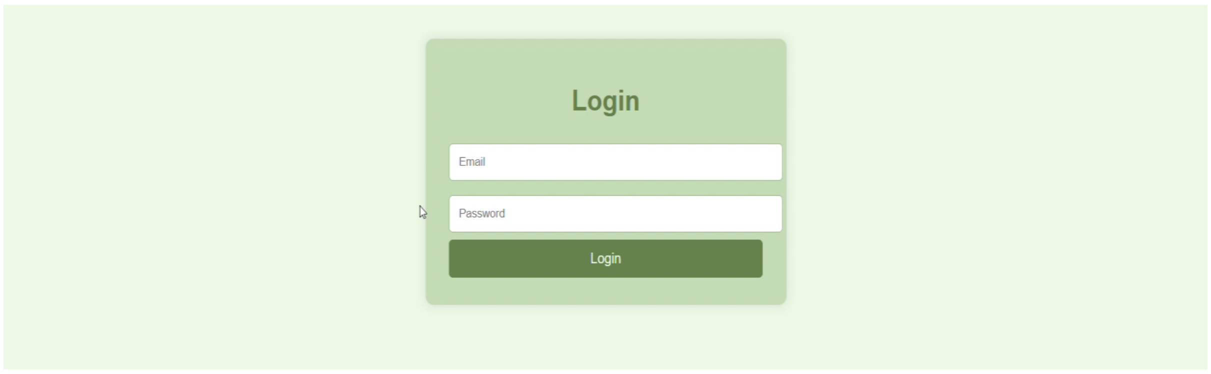
Welcome to Homemade Happiness

Authentic homemade pickles & snacks — crafted with love and tradition.

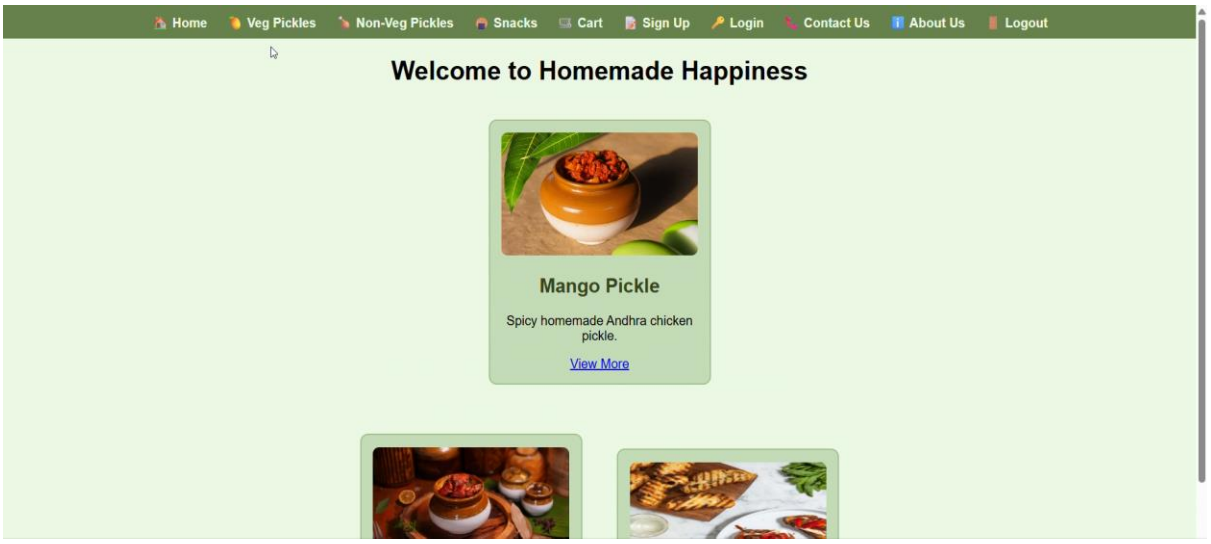
Signup page:



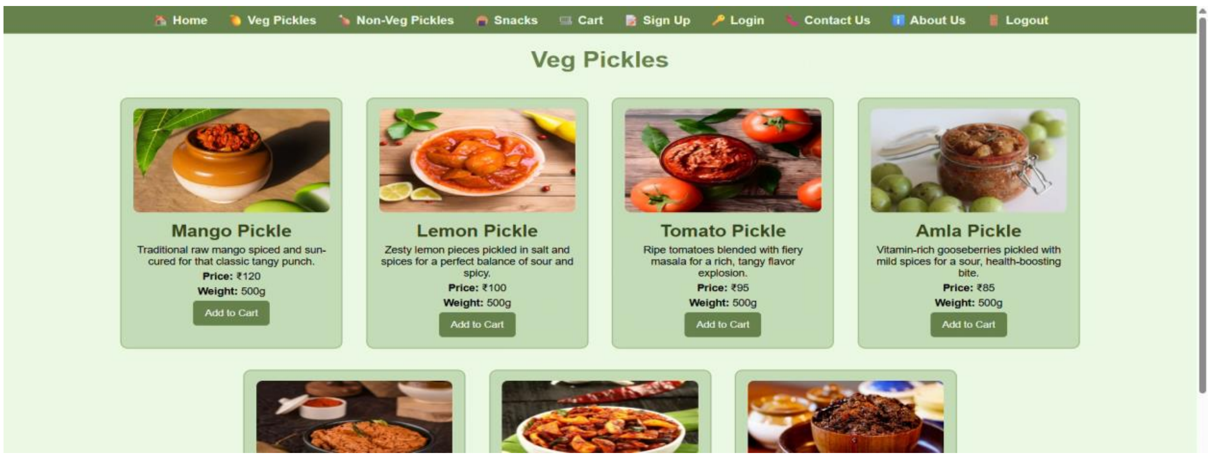
Login Page:



Home Page:



Veg_Pickles Page:



Veg Pickles

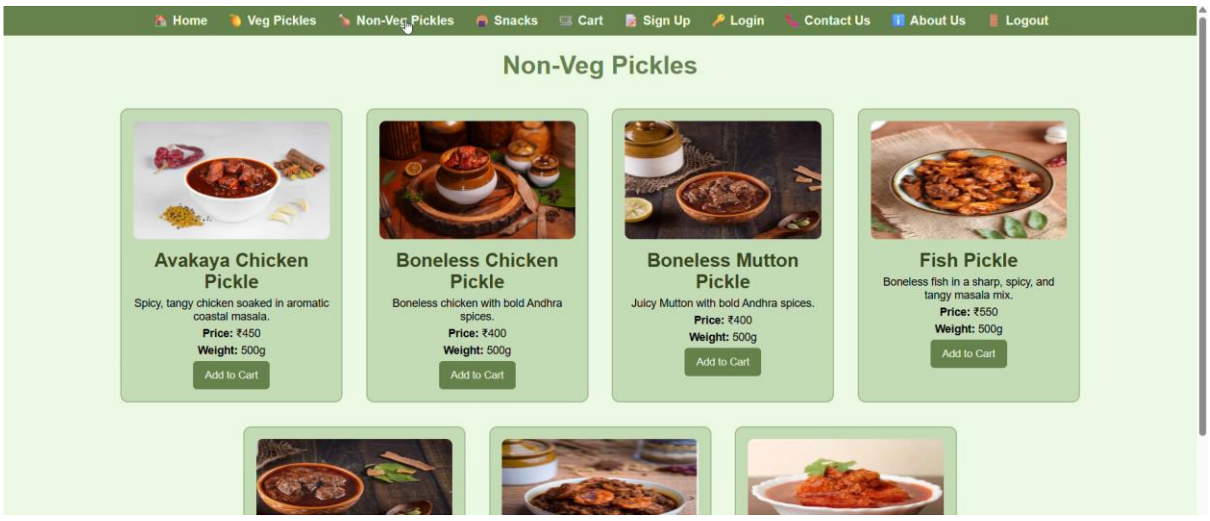
Mango Pickle
Traditional raw mango spiced and sun-cured for that classic tangy punch.
Price: ₹120
Weight: 500g
Add to Cart

Lemon Pickle
Zesty lemon pieces pickled in salt and spices for a perfect balance of sour and spicy.
Price: ₹100
Weight: 500g
Add to Cart

Tomato Pickle
Ripe tomatoes blended with fiery masala for a rich, tangy flavor explosion.
Price: ₹95
Weight: 500g
Add to Cart

Amla Pickle
Vitamin-rich gooseberries pickled with mild spices for a sour, health-boosting bite.
Price: ₹85
Weight: 500g
Add to Cart

Non_veg Pickles Page:



Non-Veg Pickles

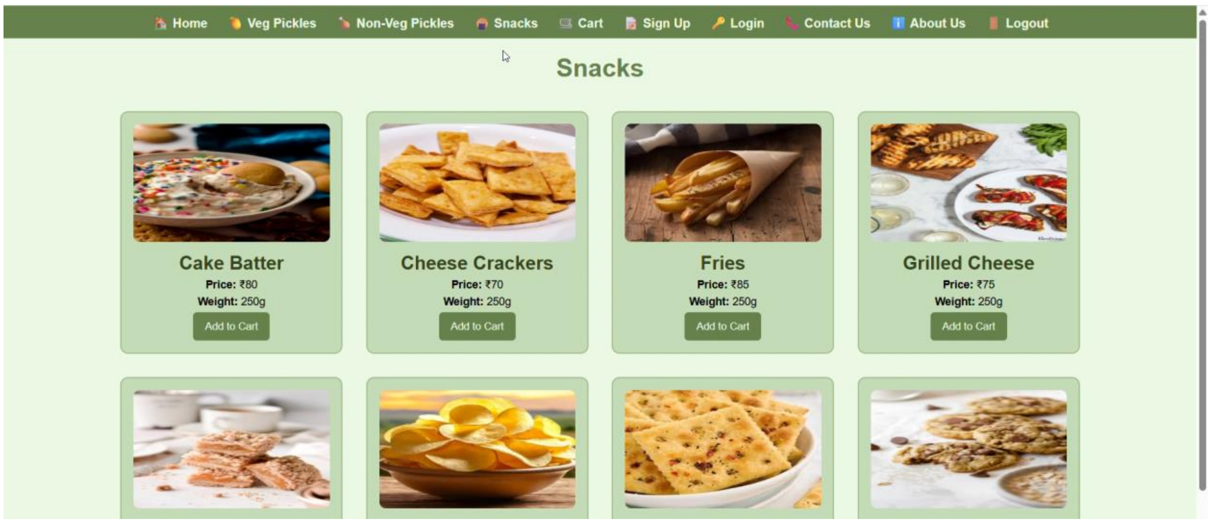
Avakaya Chicken Pickle
Spicy, tangy chicken soaked in aromatic coastal masala.
Price: ₹450
Weight: 500g
Add to Cart

Boneless Chicken Pickle
Boneless chicken with bold Andhra spices.
Price: ₹400
Weight: 500g
Add to Cart

Boneless Mutton Pickle
Juicy Mutton with bold Andhra spices.
Price: ₹400
Weight: 500g
Add to Cart

Fish Pickle
Boneless fish in a sharp, spicy, and tangy masala mix.
Price: ₹550
Weight: 500g
Add to Cart

Snacks Page:



Snacks

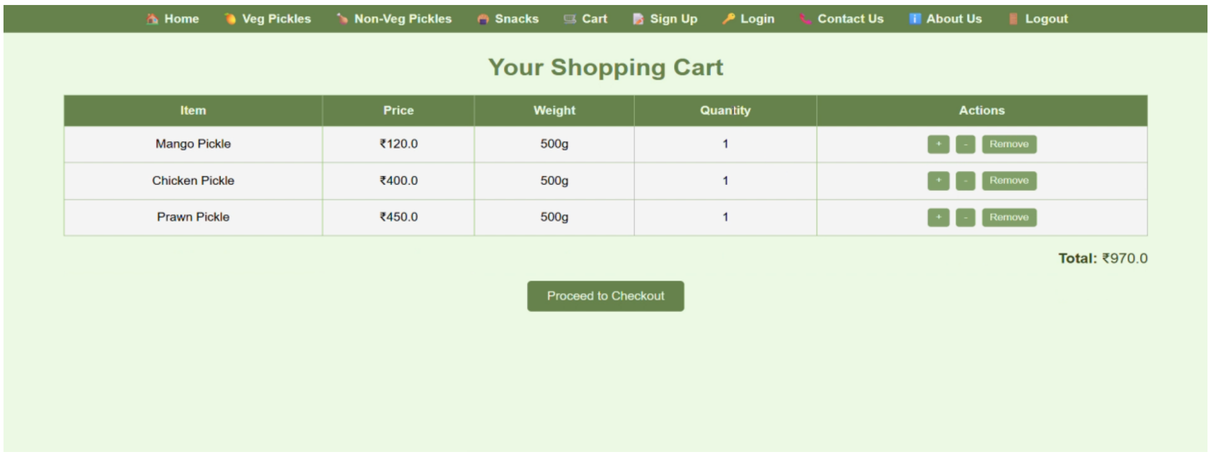
Cake Batter
Price: ₹80
Weight: 250g
Add to Cart

Cheese Crackers
Price: ₹70
Weight: 250g
Add to Cart

Fries
Price: ₹85
Weight: 250g
Add to Cart

Grilled Cheese
Price: ₹75
Weight: 250g
Add to Cart

Cart Page:


Home Veg Pickles Non-Veg Pickles Snacks Cart Sign Up Login Contact Us About Us Logout

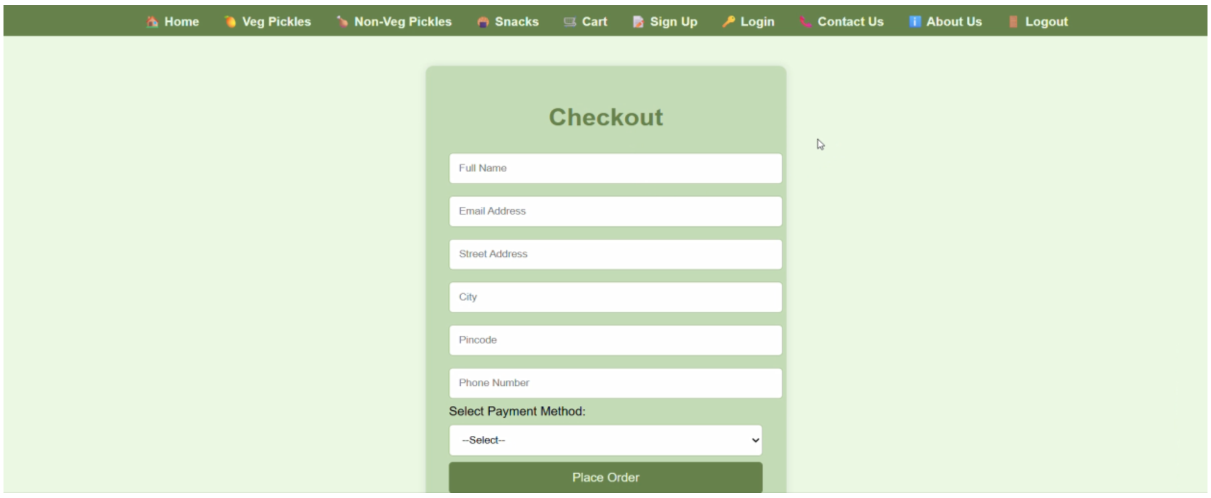
Your Shopping Cart

Item	Price	Weight	Quantity	Actions
Mango Pickle	₹120.0	500g	1	+ - Remove
Chicken Pickle	₹400.0	500g	1	+ - Remove
Prawn Pickle	₹450.0	500g	1	+ - Remove

Total: ₹970.0

Proceed to Checkout

Checkout Page:


Home Veg Pickles Non-Veg Pickles Snacks Cart Sign Up Login Contact Us About Us Logout

Checkout

Full Name

Email Address

Street Address

City

Pincode

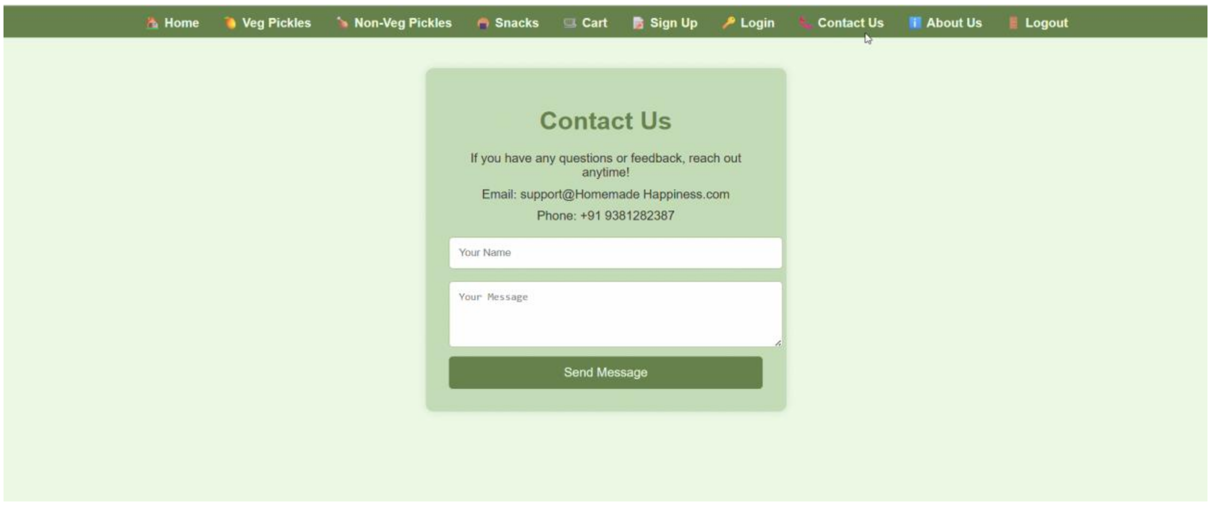
Phone Number

Select Payment Method:

--Select--

Place Order

Contact Us Page:


Home Veg Pickles Non-Veg Pickles Snacks Cart Sign Up Login Contact Us About Us Logout

Contact Us

If you have any questions or feedback, reach out anytime!

Email: support@Homemade Happiness.com

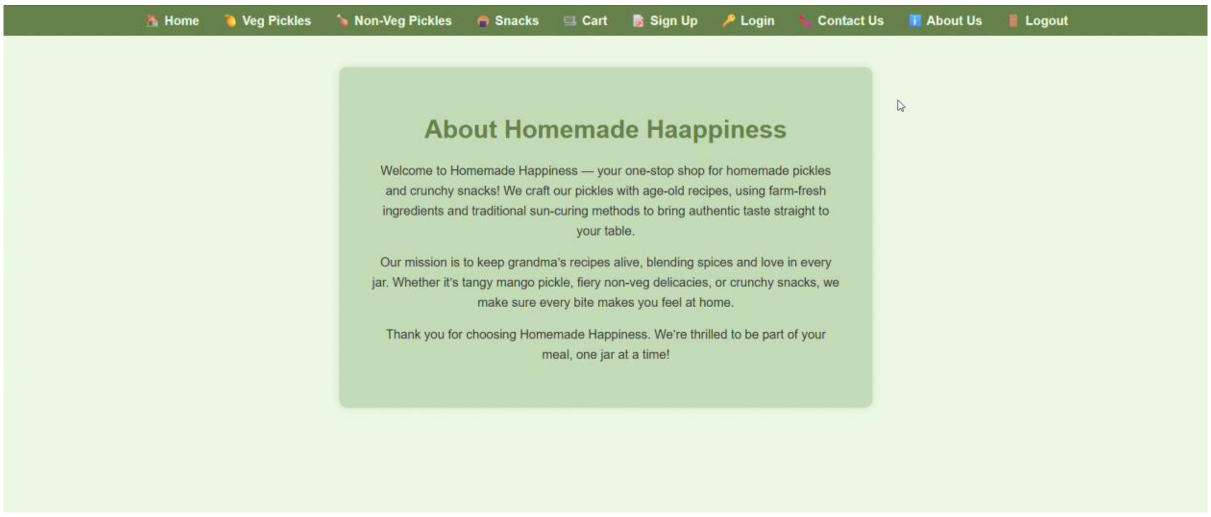
Phone: +91 9381282387

Your Name

Your Message

Send Message

About Us Page:



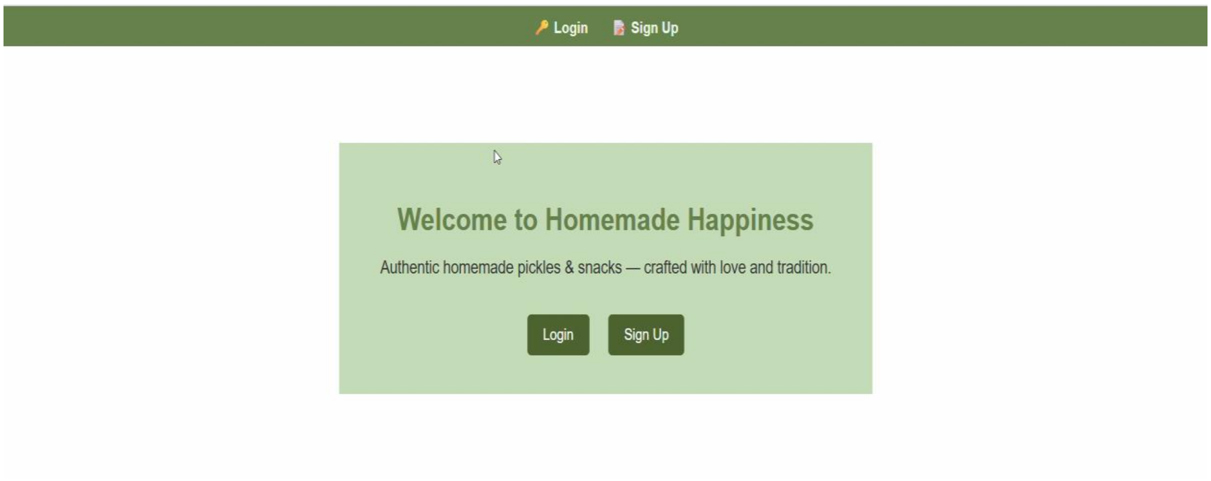
About Homemade Haappiness

Welcome to Homemade Happiness — 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.

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.

Thank you for choosing Homemade Happiness. We're thrilled to be part of your meal, one jar at a time!

After Logout



Welcome to Homemade Happiness

Authentic homemade pickles & snacks — crafted with love and tradition.