

HomeMade Pickles & Snacks: Taste the Best

Project Description:

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

Scenarios:

Scenario 1: Scalable Order Management for High Demand

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

Scenario 2: Real-Time Inventory Tracking and Updates

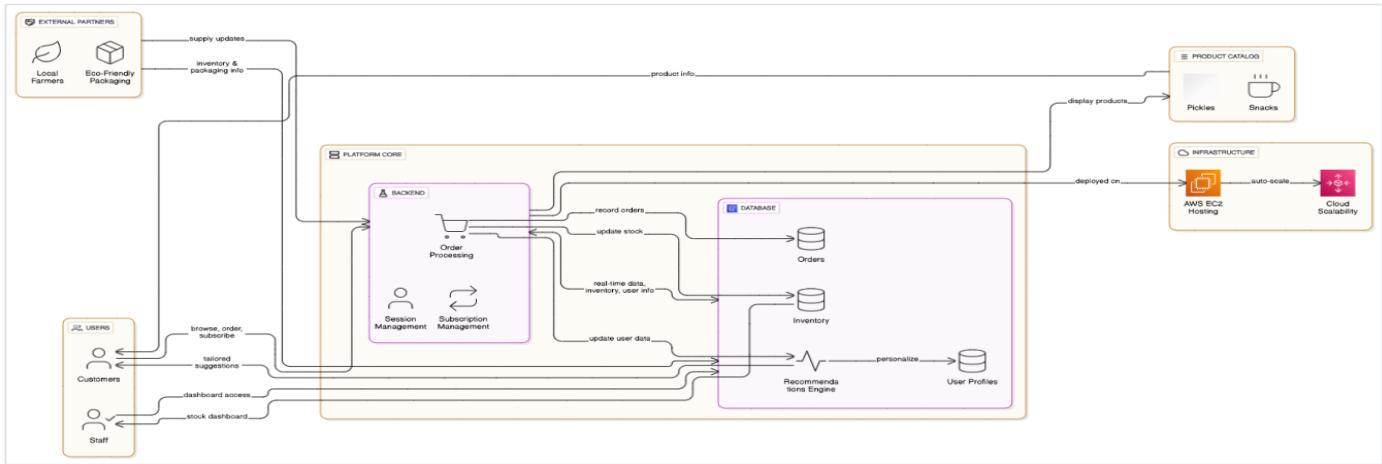
When a customer places an order for a product, the system instantly updates stock levels and records transaction details. For example, a user purchases an item, triggering automatic inventory deduction and order confirmation. Staff members receive updated dashboards to monitor stock availability and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

Scenario 3: Personalized User Experience and Recommendations

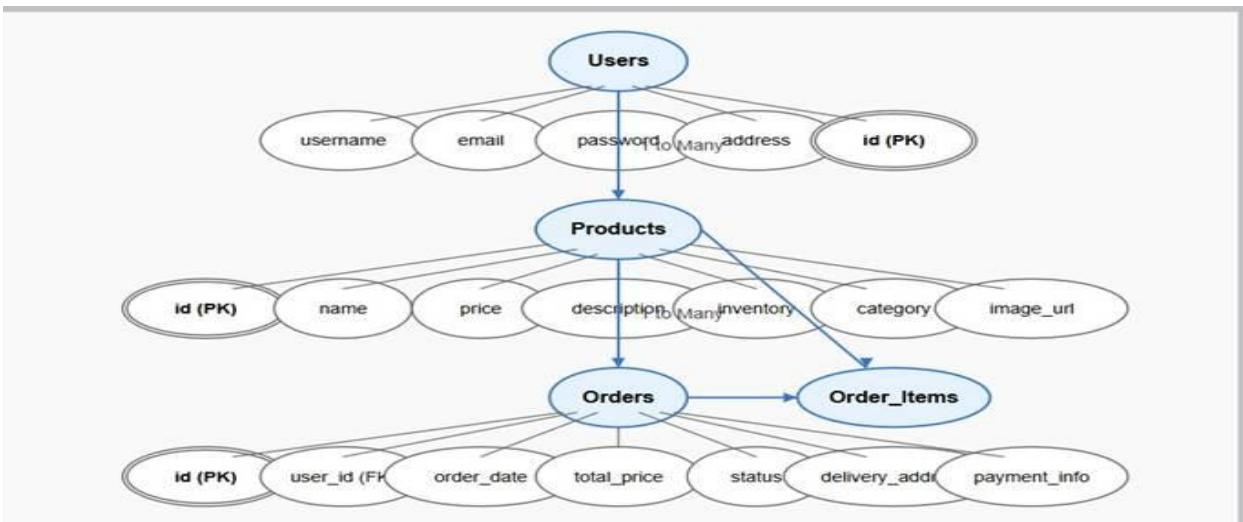
The platform leverages user behavior data to enhance engagement. A returning customer, for instance, views tailored recommendations based on past purchases and browsing history. The system dynamically adjusts suggestions in real-time, while maintaining fast response rates even during high traffic, creating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

AWS ARCHITECTURE

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:



Pre-requisites:

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

Project WorkFlow:

Milestone 1. Backend Development and Application Setup

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

Milestone 2. AWS Account Setup and Login

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

Milestone 3. DynamoDB Database Creation and Setup

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

Milestone 4. SNS Notification Setup

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

Milestone 5. IAM Role Setup

- Create IAM Role
- Attach Policies

Milestone 6. EC2 Instance Setup

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

Milestone 7. Deployment on EC2

- Upload Flask Files
- Run the Flask App

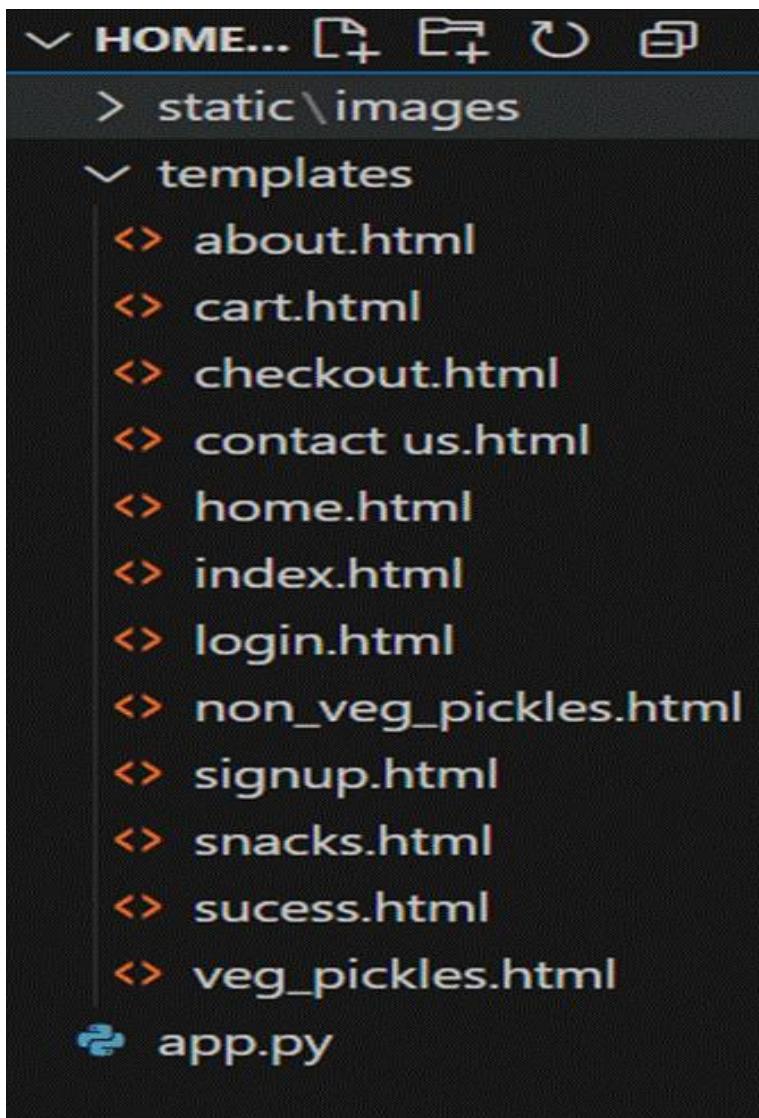
Milestone 8. Testing and Deployment

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

Milestone 1: Backend Development and Application SetUp

- **Activity 1.1:** Develop the Backend Using Flask

1. File Explorer Structure



Description: 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

Description of the code :

? Flask App Initialization

```
app.py > ...
1  from flask import Flask, render_template, request, redirect, url_for, flash, session
2  from werkzeug.security import generate_password_hash, check_password_hash
3  from datetime import datetime
4  import boto3
5  import smtplib
6  import logging
7  import uuid
8  from email.mime.text import MIMEText
9  import os
```

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

```
app.py > ...
39
40 dynamodb = boto3.resource('dynamodb', region_name=AWS_REGION)
41 orders_table = dynamodb.Table('PickleOrders')
42 users_table = dynamodb.Table('users')
43 sns = boto3.client('sns', region_name=AWS_REGION)
44
45 #Email settings (loaded securely from .env)
46 EMAIL_HOST = os.getenv('EMAIL_HOST', 'smtp.gmail.com')
47 EMAIL_PORT = int(os.getenv('EMAIL_PORT', 587))
48 EMAIL_USER = os.getenv('EMAIL_USER')
49 EMAIL_PASSWORD = os.getenv('EMAIL_PASSWORD')
50
51
52 @app.context_processor
53 def inject_theme():
54     return {"color": app.config["THEME_COLOR"], "year": datetime.now().year}
55
56 # ----- Updated Product Inventory -----
57 products = {
58     "mango": {"name": "Mango Pickle", "price": 249, "stock": 10, "image": "mango.jpg"},
59     "lemon": {"name": "Lemon Pickle", "price": 199, "stock": 8, "image": "lemon.jpg"},
60     "gongura": {"name": "Gongura Pickle", "price": 229, "stock": 7, "image": "gongura.jpg"},
61     "chicken": {"name": "Chicken Pickle", "price": 349, "stock": 12, "image": "chicken.jpg"},
62     "fish": {"name": "Fish Pickle", "price": 329, "stock": 9, "image": "fish.jpg"},
63     "prawns": {"name": "Prawns Pickle", "price": 399, "stock": 6, "image": "prawns.jpg"},
64     "murukulu": {"name": "Murukulu", "price": 149, "stock": 20, "image": "murukulu.jpg"},
65     "nippattu": {"name": "Nippattu", "price": 129, "stock": 15, "image": "nippattu.jpg"},
66     "hot_maida_biscuit": {"name": "Hot Maida Biscuit", "price": 109, "stock": 25, "image": "hot_maida_biscuit.jpg"},
67 }
68
69 def get_products(prefix=None):
```

- Routes for Web Pages

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

```

@app.route("/login", methods=["GET", "POST"])
def login():
    if request.method == "POST":
        email = request.form["email"]
        pwd = request.form["password"]
        user = users.get(email)
        if user and check_password_hash(user["hash"], pwd):
            session["user"] = email
            flash("Logged in!", "success")
            return redirect(url_for("home"))
        flash("Invalid credentials", "danger")
    return render_template("login.html")

@app.route("/logout")

```

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

```

@app.route("/signup", methods=["GET", "POST"])
def signup():
    if request.method == "POST":
        email = request.form["email"]
        pwd = request.form["password"]
        confirm = request.form["confirm"]
        if pwd != confirm:
            flash("Passwords don't match", "warning")
            return redirect(url_for("signup"))
        if email in users:
            flash("User exists, log in", "info")
            return redirect(url_for("login"))
        users[email] = {"hash": generate_password_hash(pwd)}
        flash("Signup complete - please log in", "success")
        return redirect(url_for("login"))
    return render_template("signup.html")

```

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

```
# ----- Routes -----
@app.route("/")
def home():
    best = dict(list(get_products().items())[:6])
    return render_template("index.html", items=best)

@app.route("/veg")
def veg():
    return render_template("veg.html", items={
        k: v for k, v in products.items() if k in ["mango", "lemon", "gongura"]
    })

@app.route("/nonveg")
def nonveg():
    return render_template("nonveg.html", items={
        k: v for k, v in products.items() if k in ["chicken", "fish", "prawns"]
    })

@app.route("/snacks")
def snacks():
    return render_template("snacks.html", items={
        k: v for k, v in products.items() if k in ["murukulu", "nippattu", "hot_maida_biscuit"]
    })

@app.route("/cart")
def cart():
    items, total = {}, 0
    if session.get("cart"):
        for pid, qty in session["cart"].items():
            if pid in products:
                items[pid] = {"products[pid]": "qty": qty}
                total += products[pid]["price"] * qty
    return render_template("cart.html", items=items, total=total)
```

CheckOut

```
app.route("/checkout", methods=["GET", "POST"])
def checkout():
    cart = session.get("cart", {})
    if not cart:
        flash("cart is empty", "warning")
        return redirect(url_for("home"))

    if request.method == "POST":
        name = request.form["name"]
        email = request.form["email"]
        address = request.form["address"]
        order_id = str(uuid.uuid4())
        order_time = datetime.now().isoformat()

        items = [
            {"product": pid, "price": products[pid]["price"], "quantity": qty}
            for pid, qty in cart.items()
        ]
        total = sum(p["price"] * p["quantity"] for p in items)

    order_data = {
        "order_id": order_id,
        "name": name,
        "email": email,
        "address": address,
        "order_time": order_time,
        "items": items,
        "total": total
    }

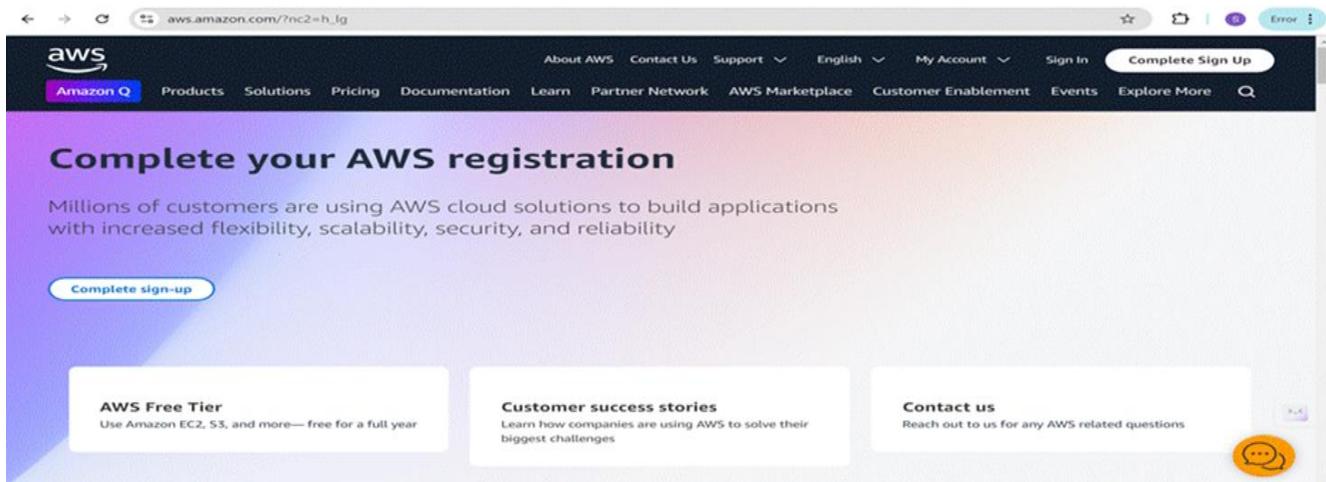
    save_order_to_dynamodb(order_data)

    summary = f"Order ID: {order_id}\nName: {name}\nTotal: ₹{total}\n\nThank you for your order!"
    send_order_email(email, summary)
    send sns notification(f"New order placed: {order_id}")

    session.pop("cart", None)
    flash("Order placed successfully!", "success")
    return redirect(url_for("success"))
return render_template("checkout.html")
```

- **Milestone 2: AWS Account Setup and Login.**

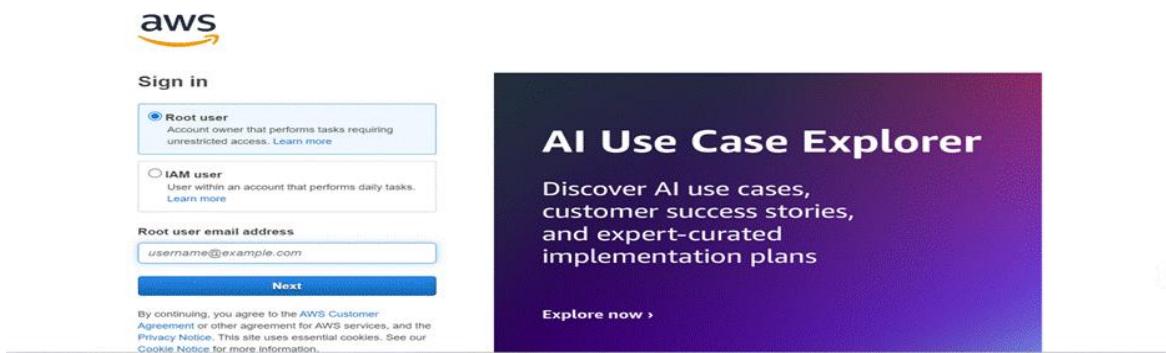
- **Activity 2.1:** Set up an AWS account if not already done.
 - Sign up for an AWS account and configure billing settings.



- Click on the "Create an AWS Account" button.
- Follow the prompts to enter your email address and choose a password.
- Provide the required account information, including your name, address, and phone number.
- Enter your payment information. (Note: While AWS offers a free tier, a credit card or debit card is required for verification.)
- Complete the identity verification process.
- Choose a support plan (the basic plan is free and sufficient for starting).
- Once verified, you can sign in to your new AWS accounts.

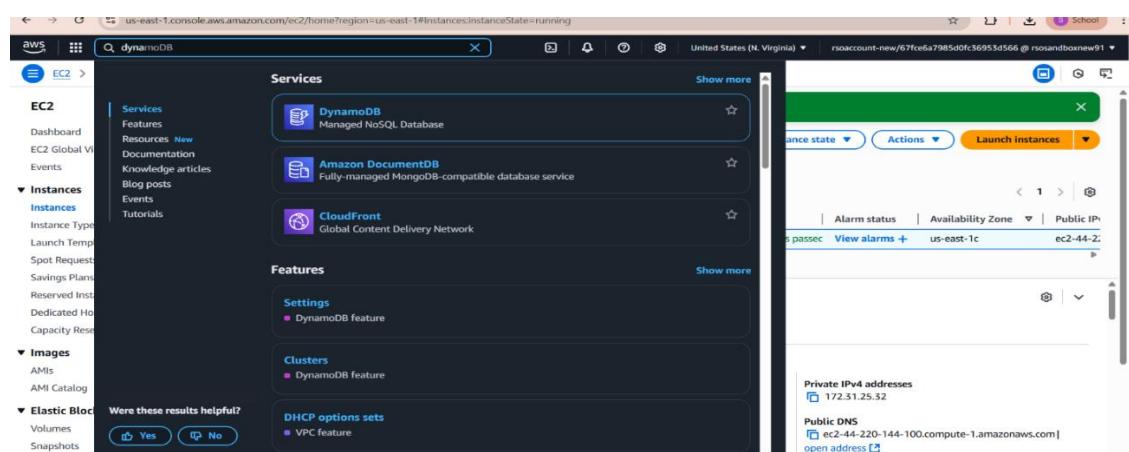


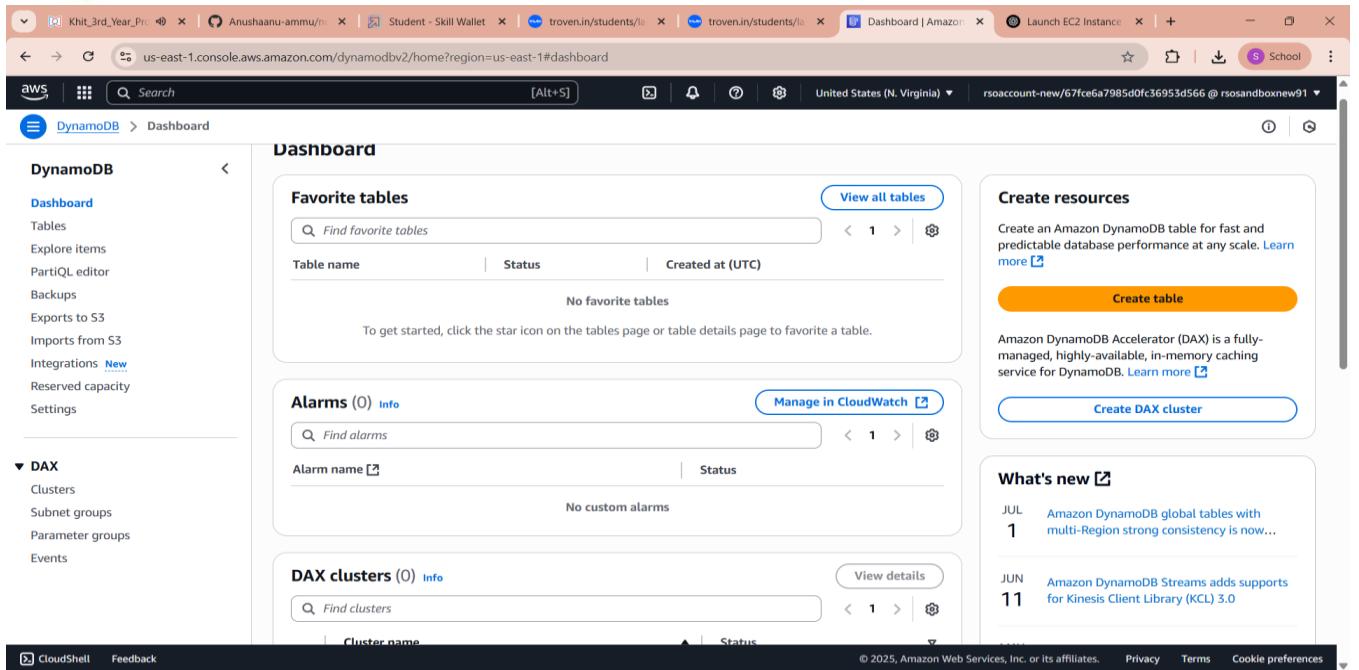
- **Activity2.2 : Log in to the AWS Management Console**
 - After setting up your account, log in to the [AWS Management Console](#).



Milestone 3: DynamoDB Database Creation and Setup

- **Activity 3.1: Navigate to the DynamoDB**
 - In the AWS Console, navigate to DynamoDB and click on create tables.





The screenshot shows the AWS DynamoDB Dashboard. On the left, there's a sidebar with 'DynamoDB' selected, showing options like 'Tables', 'Explore items', 'PartiQL editor', etc. Under 'DAX', it shows 'Clusters', 'Subnet groups', 'Parameter groups', and 'Events'. At the bottom of the sidebar are 'CloudShell' and 'Feedback' buttons.

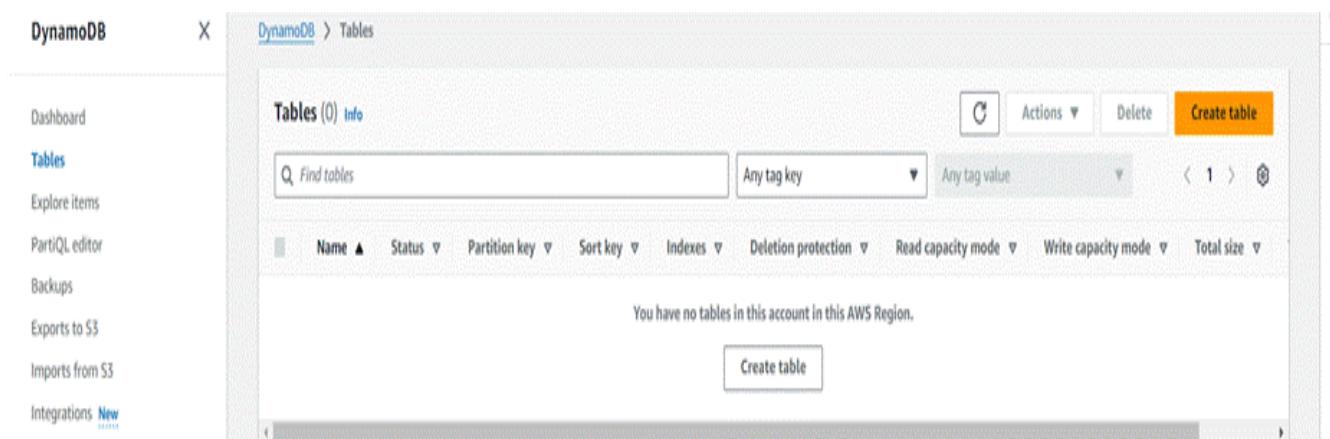
The main content area has three sections:

- Favorite tables:** A table with columns 'Table name', 'Status', and 'Created at (UTC)'. It says 'No favorite tables'.
- Alarms (0):** A table with columns 'Alarm name' and 'Status'. It says 'No custom alarms'.
- DAX clusters (0):** A table with columns 'Cluster name' and 'Status'. It says 'View details'.

To the right, there's a 'Create resources' section with a 'Create table' button. Below it, another 'Create DAX cluster' button. At the bottom right, there's a 'What's new' section with two items:

- JUL 1 Amazon DynamoDB global tables with multi-Region strong consistency is now...
- JUN 11 Amazon DynamoDB Streams adds supports for Kinesis Client Library (KCL) 3.0

At the very bottom, there's a footer with links for 'Privacy', 'Terms', and 'Cookie preferences'.



The screenshot shows the 'Tables' page under 'DynamoDB'. The sidebar shows 'Tables' selected, with other options like 'Explore items', 'PartiQL editor', etc. The main area shows 'Tables (0)' with a 'Create table' button. It says 'You have no tables in this account in this AWS Region.' There are filters for 'Find tables', 'Any tag key', 'Any tag value', and pagination controls.

Activity 3.2: Create a DynamoDB table for storing registration details and book requests.

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

us-east-1.console.aws.amazon.com/dynamodbv2/home?region=us-east-1#create-table

DynamoDB > Tables > Create table

Create table

Table details Info

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

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

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

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

Table settings

Default settings
 The fastest way to create your table. You can modify most of these settings after your table has been created. To

Customize settings
 Use these advanced features to make DynamoDB work better for your needs.

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

us-east-1.console.aws.amazon.com/dynamodbv2/home?region=us-east-1#tables

DynamoDB > Tables

The users table was created successfully.

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

DAX Clusters Subnet groups Parameter groups Events

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

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

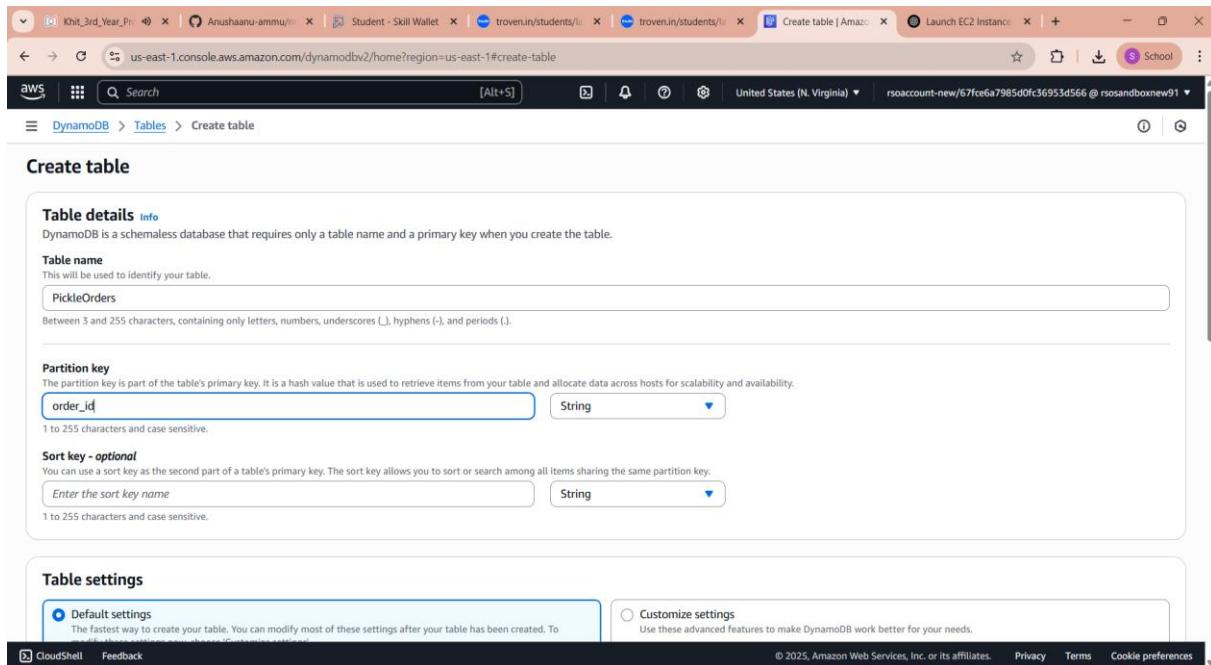


Table details Info

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

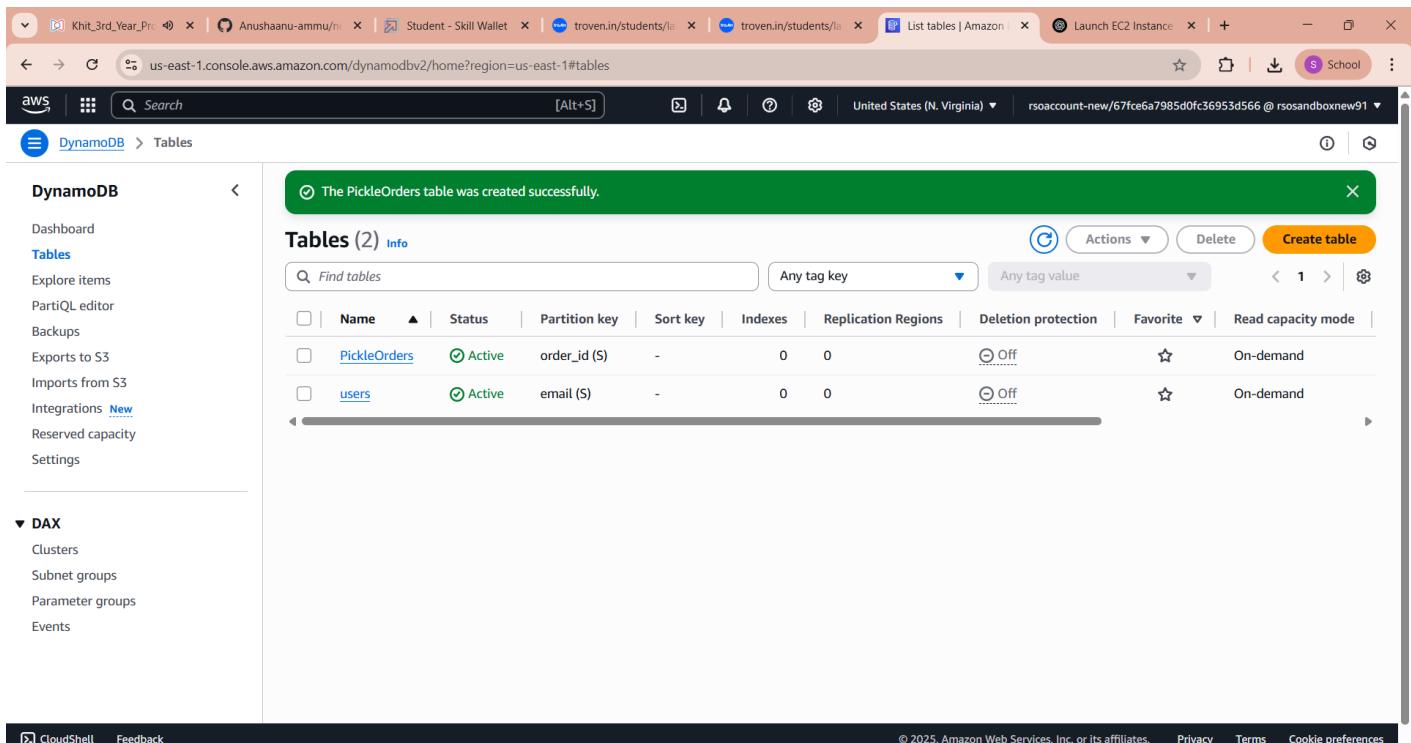
Table name
 This will be used to identify your table.
 Between 3 and 255 characters, containing only letters, numbers, underscores (_), hyphens (-), and periods (.)

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

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

Table settings

Default settings The fastest way to create your table. You can modify most of these settings after your table has been created. To **Customize settings** Use these advanced features to make DynamoDB work better for your needs.



DynamoDB

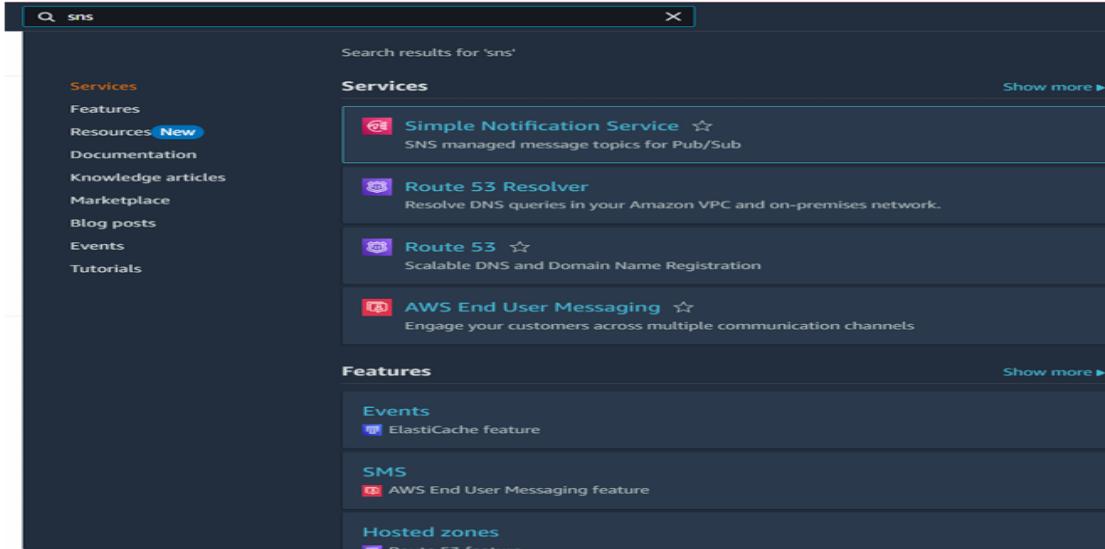
Tables

The PickleOrders table was created successfully.

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

Milestone 4. SNS Notification Setup

- **Activity4.1:** Create SNS topics for book request notifications.



The screenshot shows the AWS search interface with the query 'sns' entered. The results are categorized under 'Services' and 'Features'. Under 'Services', the 'Simple Notification Service' is listed first, described as 'SNS managed message topics for Pub/Sub'. Other services like Route 53 Resolver, Route 53, and AWS End User Messaging are also listed. Under 'Features', there are sections for 'Events' (ElasticCache feature), 'SMS' (AWS End User Messaging feature), and 'Hosted zones' (Route 53 feature).



The screenshot shows the 'Amazon Simple Notification Service' landing page. On the left, a sidebar lists 'Dashboard', 'Topics', 'Subscriptions', and 'Mobile' (with 'Push notifications' and 'Text messaging (SMS)'). The main content area features a large heading 'Amazon Simple Notification Service' with the subtext 'Pub/sub messaging for microservices and serverless applications.' A 'New Feature' banner at the top right states 'Amazon SNS now supports in-place message archiving and replay for FIFO topics. Learn more' with a link. To the right, a 'Create topic' form has a 'Topic name' field containing 'MyTopic' and a 'Next step' button. At the bottom, there are links for 'Start with an overview' and 'Pricing'.

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

Amazon SNS > Topics > Create topic

Create topic

Details

Type | [Info](#)

Topic type cannot be modified after topic is created

FIFO (first-in, first-out)

- Strictly-preserved message ordering
- Exactly-once message delivery
- High throughput, up to 300 publishes/second
- Subscription protocols: SQS

Standard

- Best-effort message ordering
- At-least once message delivery
- Highest throughput in publishes/second
- Subscription protocols: SQS, Lambda, HTTP, SMS, email, mobile application endpoints

Name

BookRequestNotifications

Maximum 256 characters. Can include alphanumeric characters, hyphens (-) and underscores (_).

Display name - *optional* | [Info](#)

To use this topic with SMS subscriptions, enter a display name. Only the first 10 characters are displayed in an SMS message.

My Topic

Maximum 100 characters.

► Access policy - *optional* [Info](#)

This policy defines who can access your topic. By default, only the topic owner can publish or subscribe to the topic.

► Data protection policy - *optional* [Info](#)

This policy defines which sensitive data to monitor and to prevent from being exchanged via your topic.

► Delivery policy (HTTP/S) - *optional* [Info](#)

The policy defines how Amazon SNS retries failed deliveries to HTTP/S endpoints. To modify the default settings, expand this section.

► Delivery status logging - *optional* [Info](#)

These settings configure the logging of message delivery status to CloudWatch Logs.

► Tags - *optional*

A tag is a metadata label that you can assign to an Amazon SNS topic. Each tag consists of a key and an optional value. You can use tags to search and filter your topics and track your costs. [Learn more](#)

► Active tracing - *optional* [Info](#)

Use AWS X-Ray active tracing for this topic to view its traces and service map in Amazon CloudWatch. Additional costs apply.

[Cancel](#)

Create topic

- Configure the SNS topic and note down the Topic ARN

Amazon SNS

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

Topic BookRequestNotifications created successfully.
You can create subscriptions and send messages to them from this topic.

Amazon SNS > Topic > BookRequestNotifications

BookRequestNotifications

Details

Name: BookRequestNotifications	Display name:
ARN: arn:aws:sns:ap-south-1:557690616836:BookRequestNotifications	Topic owner: 557690616836
Type: Standard	

Subscriptions | Access policy | Data protection policy | Delivery policy (HTTP/S) | Delivery status logging | Encryption | Tags | Integrations

Subscriptions (0)

No subscriptions found.
You don't have any subscriptions to this topic.

Create subscription

- Subscribe users and library staff to SNS email notifications.

Amazon SNS > Subscriptions > Create subscription

Create subscription

Details

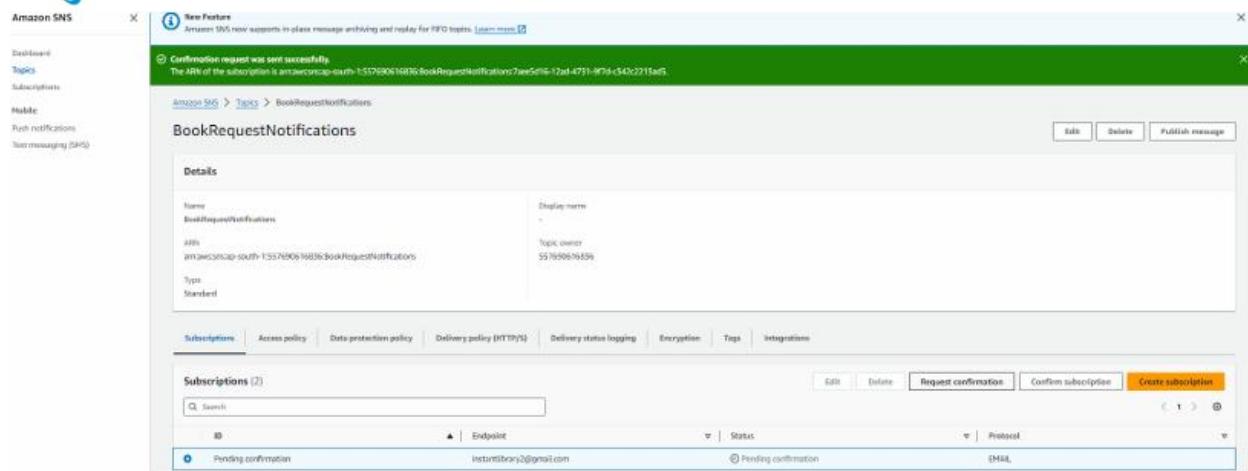
Topic ARN: arn:aws:sns:ap-south-1:557690616836:BookRequestNotifications	X
Protocol: Email	▼
Endpoint: instantlibrary2@gmail.com	

After your subscription is created, you must confirm it. [Info](#)

Subscription filter policy - optional [Info](#)
This policy filters the messages that a subscriber receives.

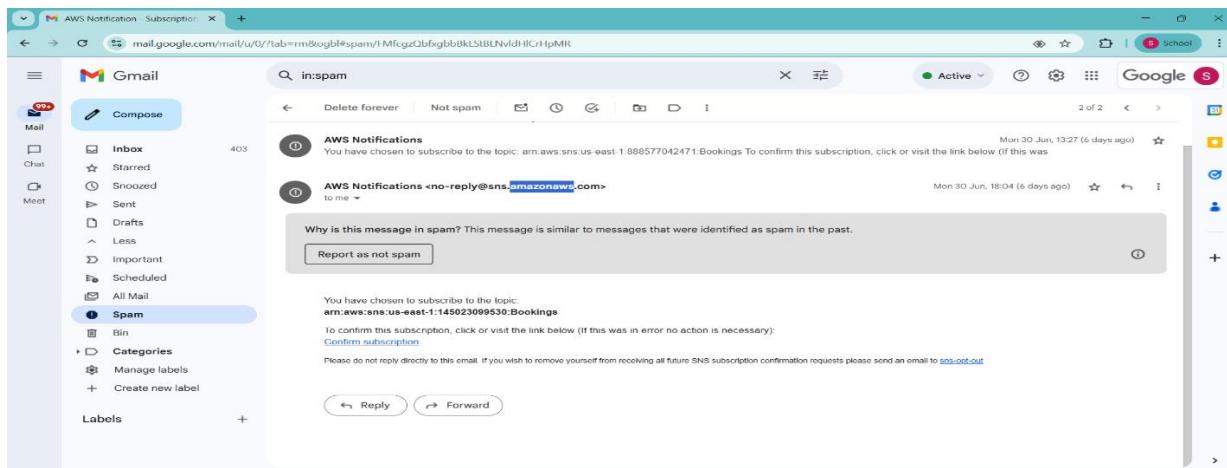
Redrive policy (dead-letter queue) - optional [Info](#)
Send undeliverable messages to a dead-letter queue.

Cancel **Create subscription**



The screenshot shows the Amazon SNS console. A success message at the top states: "Confirmation request was sent successfully. The ARN of the subscription is arn:aws:sns:ap-south-1:557690616836:BookRequestNotifications:7ae5d9f6-12ad-47f1-8f7d-c540c2211a65." Below this, the "BookRequestNotifications" topic details are shown, including its ARN and type. The "Subscriptions" tab is selected, displaying one pending confirmation subscription for the email "internzlibrary2@gmail.com".

- After Confirmation of Subscription going to Mail for confirm Mail .



The screenshot shows a Gmail inbox with two messages from "AWS Notifications". The first message is from "AWS Notifications" and says: "You have chosen to subscribe to the topic: arn:aws:sns:us-east-1:880577042471:Bookings To confirm this subscription, click or visit the link below (if this was a mistake)." The second message is from "AWS Notifications <no-reply@sns.amazonaws.com> to me" and says: "Why is this message in spam? This message is similar to messages that were identified as spam in the past. Report as not spam". Both messages are marked as "Pending confirmation".



Simple Notification Service

Subscription confirmed!

You have successfully subscribed.

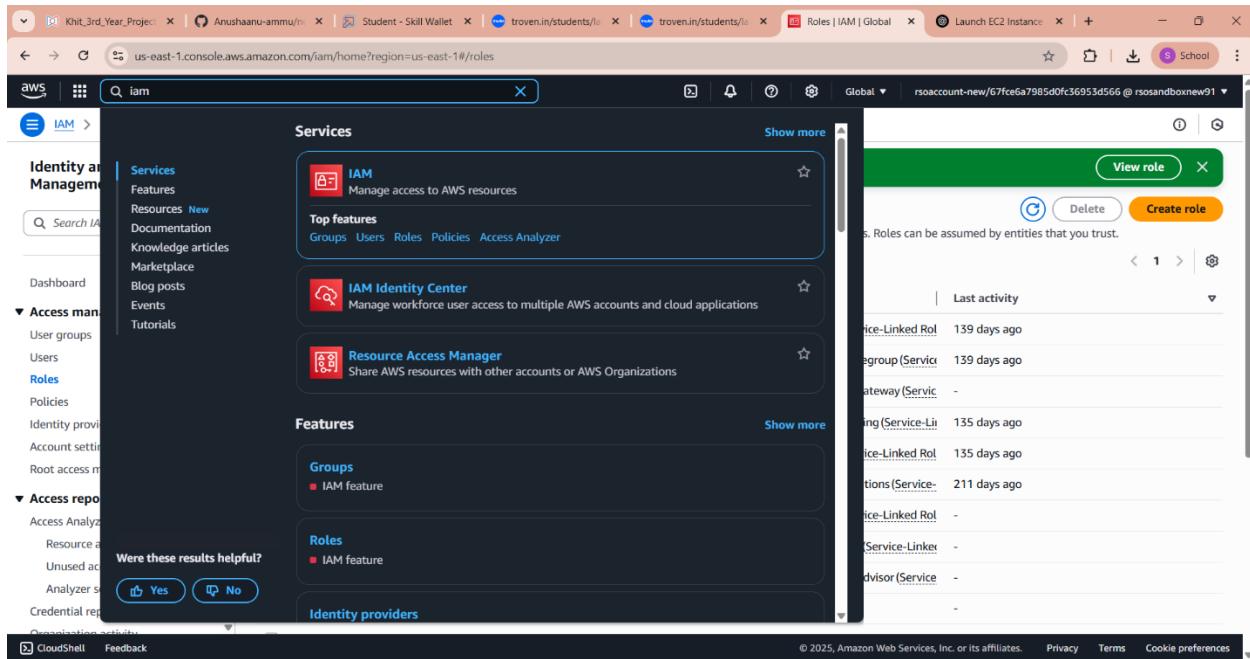
Your subscription's id is:

arn:aws:sns:ap-south-1:557690616836:BookRequestNotifications:d78e0371-9235-404d-952c-85c2743607c4

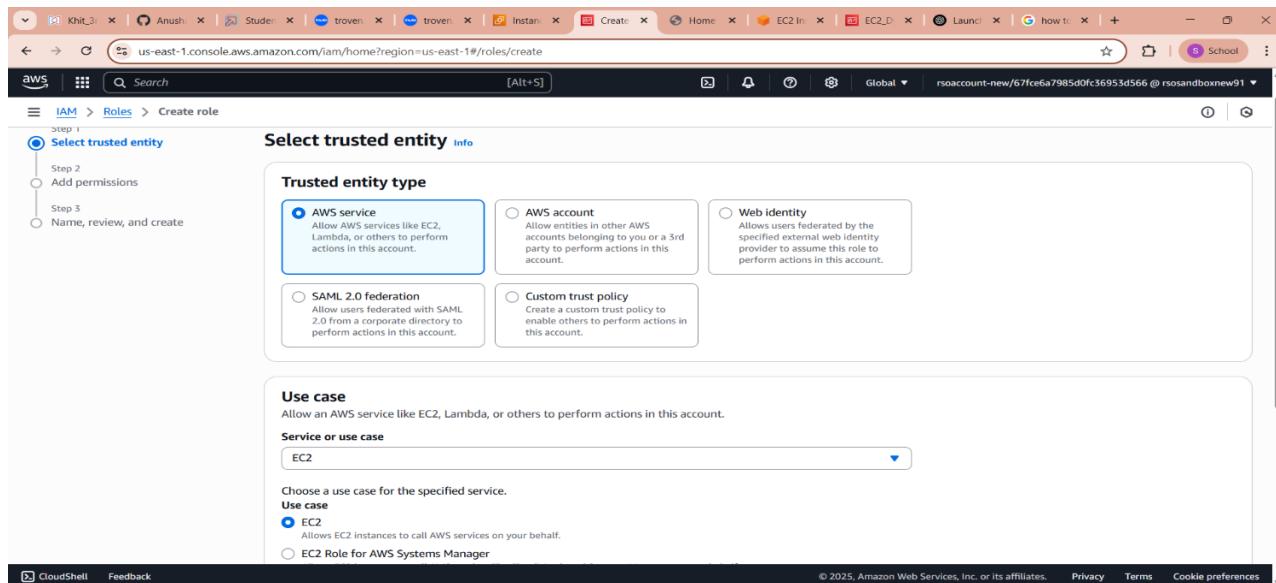
If it was not your intention to subscribe, [click here to unsubscribe](#).

Milestone 5 : IAM Role Setup.

- **Activity 5.1 :** In the AWS Console, go to IAM and create a new IAM Role for EC2 to interact with DynamoDB .



The screenshot shows the AWS IAM service page. On the left, there is a sidebar with navigation links for Identity and Management, Access management, and Access reporting. The main content area displays a list of services under 'Services' and 'Features'. Under 'Services', the 'IAM' service is highlighted, showing its description: 'Manage access to AWS resources'. Below it are 'Top features' like Groups, Users, Roles, Policies, and Access Analyzer. Under 'Features', there are sections for 'Groups' and 'Roles', both labeled as 'IAM feature'. A 'Were these results helpful?' poll is present. On the right, a modal window titled 'View role' is open, showing a list of roles with their last activity dates. At the bottom, there are links for CloudShell and Feedback.



The screenshot shows the 'Create role' wizard, Step 1: Select trusted entity. It has three steps: Step 1 (Select trusted entity), Step 2 (Add permissions), and Step 3 (Name, review, and create). The first step is active. It shows a 'Trusted entity type' section with five options: 'AWS service' (selected), 'AWS account', 'Web identity', 'SAML 2.0 federation', and 'Custom trust policy'. Below this is a 'Use case' section with a dropdown set to 'EC2' and a 'Service or use case' dropdown also set to 'EC2'. At the bottom, there are 'Choose a use case for the specified service.' and 'Use case' sections, with 'EC2' selected.

- **Activity 5.2 : Attach Policies**
- AmazonDynamoDBFullAccess: Allows EC2 to perform read/write operations on DynamoDB.
- AmazonSNSFullAccess: Grants EC2 the ability to send notifications via SNS.

IAM > Roles > Create role

Step 1 Select trusted entity

Step 2 Add permissions

Step 3 Name, review, and create

Add permissions Info

Permissions policies (1/955) Info

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

Filter by Type All types 2 matches

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

Set permissions boundary - optional

Cancel Previous Next

IAM > Roles > Create role

Step 1 Select trusted entity

Step 2 Add permissions

Step 3 Name, review, and create

Add permissions Info

Permissions policies (2/955) Info

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

Filter by Type All types 5 matches

Policy name	Type
<input checked="" type="checkbox"/>  AmazonSNSFullAccess	AWS managed
<input type="checkbox"/>  AmazonSNSReadOnlyAccess	AWS managed
<input type="checkbox"/>  AmazonSNSRole	AWS managed
<input type="checkbox"/>  AmazonSNSLambdaRoleSNS	AWS managed
<input type="checkbox"/>  AmazonSNSDeviceDefenderPublishFindingsToSNSMigrationAction	AWS managed

Set permissions boundary - optional

Cancel Previous Next

us-east-1.console.aws.amazon.com/iam/home?region=us-east-1#/roles/details/EC2_DynamoDB_Role?section=permissions

IAM > Roles > EC2_DynamoDB_Role

EC2_DynamoDB_Role Info

Allows EC2 instances to call AWS services on your behalf.

Summary

Creation date: July 03, 2025, 12:37 (UTC+05:30)

Last activity: 43 minutes ago

ARN: arn:aws:iam::463470967337:role/EC2_DynamoDB_Role

Maximum session duration: 1 hour

Permissions Trust relationships Tags Last Accessed Revoke sessions

Permissions policies (3) Info

You can attach up to 10 managed policies.

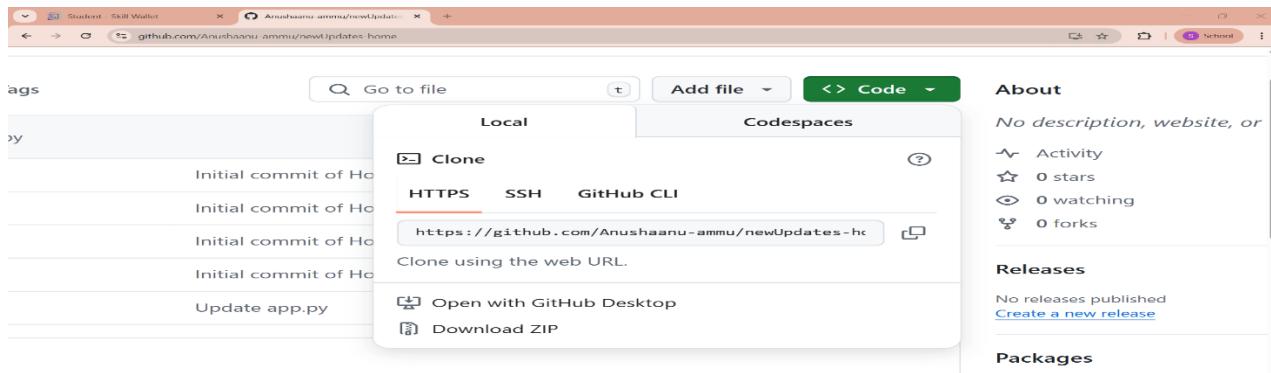
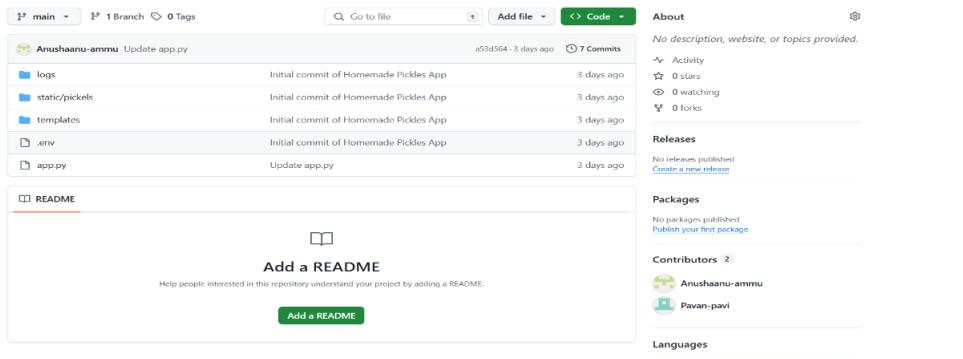
Filter by Type All types

Policy name	Type	Attached entities
<input checked="" type="checkbox"/>  AmazonDynamoDBFullAccess	AWS managed	1
<input checked="" type="checkbox"/>  AmazonEC2FullAccess	AWS managed	1

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

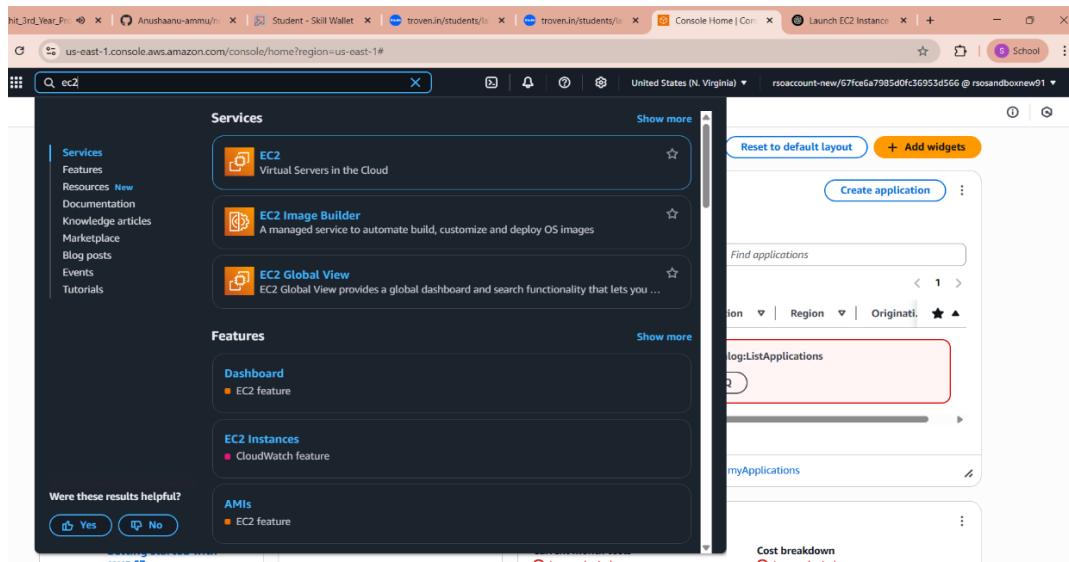
- **Milestone 6. EC2 Instance Setup**

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



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

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





aws Search [Alt+S] United States (N. Virginia) rsoaccount-new/67fce6a7985d0fc36953d566 @ rsoandboxnew91

EC2 > Instances > Launch an instance

Launch an instance [Info](#)

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

Name and tags [Info](#)

Name Add additional tags

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

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

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

Recent AMIs: Amazon Linux, macOS, Ubuntu, Windows, Red Hat, SUSE Linux, Debian, AWS Mac, ubuntu®, Microsoft, Red Hat, SUSE, Debian

Browse more AMIs Including AMIs from AWS, Marketplace and the Community

Summary

Number of instances [Info](#)

1

Software Image (AMI)

Amazon Linux 2023 AMI 2023.7.2... [read more](#)
ami-05ffe3c48a9991133

Virtual server type (instance type)

t2.micro

Firewall (security group)

New security group

Storage (volumes)

1 volume(s) - 8 GB

Launch instance [Preview code](#)

Cancel

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

aws Search [Alt+S] United States (N. Virginia) rsoaccount-new/67fce6a7985d0fc36953d566 @ rsoandboxnew91

EC2 > Instances > Launch an instance

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 Create new key pair

Network settings [Info](#)

Network [Edit](#)
vpc-06b1983fd7eaa46c7

Subnet [Info](#)
No preference (Default subnet in any availability zone)

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

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

We'll create a new security group called "launch-wizard-2" with the following rules:

Allow SSH traffic from Anywhere
Helps you connect to your instance

Allow HTTPS traffic from the internet

Summary

Number of instances [Info](#)

1

Software Image (AMI)

Amazon Linux 2023 AMI 2023.7.2... [read more](#)
ami-05ffe3c48a9991133

Virtual server type (instance type)

t2.micro

Firewall (security group)

New security group

Storage (volumes)

1 volume(s) - 8 GB

Launch instance [Preview code](#)

Cancel

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

EC2 > Security Groups > sg-0f1fc765e39312322 - launch-wizard-1 Actions

sg-0f1fc765e39312322 - launch-wizard-1

Details

Security group name	sg-0f1fc765e39312322
Owner	463470967337
Inbound rules count	4 Permission entries
Description	launch-wizard-1 created 2025-07-03T07:00:14.084Z
Outbound rules count	1 Permission entry
VPC ID	vpc-06b1983fd7eaa46c7

Inbound rules [Manage tags](#) [Edit inbound rules](#)

Name	Security group rule ID	IP version	Type	Protocol	Port range
-	sgr-0eff08b1b9dd0920d	IPv4	HTTP	TCP	80
-	sgr-0116436ad44282685	IPv4	SSH	TCP	22
-	sgr-0efef928cff38eb8ea	IPv4	Custom TCP	TCP	5000
-	sgr-06e009f2b9c9b8208	IPv4	HTTPS	TCP	443

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



AWS | Search [Alt+S] | United States (N. Virginia) | rsoaccount-new/67fce6a7985d0fc36953d566 @ rsosandboxnew91

EC2 > Instances

Instances (1) Info

Find Instance by attribute or tag (case-sensitive)

Instance state = running | Clear filters

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4
HomeMadePic...	i-0bc9fb84cdde8e83a	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1c	ec2-44-220

Select an instance

CloudShell Feedback

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

AWS | Search [Alt+S] | United States (N. Virginia) | rsoaccount-new/67fce6a7985d0fc36953d566 @ rsosandboxnew91

EC2 > Instances

Instances (1/1) Info

Find Instance by attribute or tag (case-sensitive)

Instance state = running | Clear filters

Name	Instance ID	Instance state	Instance type	Change security groups
HomeMadePic...	i-0bc9fb84cdde8e83a	Running	t2.micro	Get Windows password Modify IAM role

i-0bc9fb84cdde8e83a (HomeMadePickles)

Details Status and alarms Monitoring Security Networking Storage Tags

Instance summary Info

Instance ID	Public IPv4 address	Private IPv4 addresses
i-0bc9fb84cdde8e83a	44.220.144.100 open address	172.31.25.32

IPv6 address
Instance state
Running

Public DNS
ec2-44-220-144-100.compute-1.amazonaws.com | open address

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

AWS | Search [Alt+S] | United States (N. Virginia) | rsoaccount-new/67fce6a7985d0fc36953d566 @ rsosandboxnew91

EC2 > Instances > i-0bc9fb84cdde8e83a > Modify IAM role

Modify IAM role Info

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.

Instance ID: i-0bc9fb84cdde8e83a (HomeMadePickles)

IAM role: EC2_DynamoDB_Role

Create new IAM role

Cancel Update IAM role

CloudShell Feedback

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



Khet_3rd_Year_Pri Anushaenu-ammu/ Student - Skill Wallet troven.in/students/ Instances | EC2 | us-east-1 Launch EC2 Instance School

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

AWS Search [Alt+S] United States (N. Virginia) rsoaccount-new/67fce6a7985d0fc36953d566 @ rsosandboxnew91

EC2 Instances

EC2 Dashboard EC2 Global View Events

Instances Instance Types Launch Templates Spot Requests Savings Plans Reserved Instances Dedicated Hosts Capacity Reservations

Images AMIs AMI Catalog

Elastic Block Store Volumes Snapshots Lifecycle Manager

CloudShell Feedback

Successfully attached EC2_DynamoDB_Role to instance i-0bc9fb84cdde8e83a

Instances (1/1) Info

Find Instance by attribute or tag (case-sensitive)

Instance state = running Clear filters All states

Name: HomeMadePic... Instance ID: i-0bc9fb84cdde8e83a Instance state: Running Instance type: t2.micro Status check: ... Alarm status: ... Availability Zone: us-east-1c Public IPv4: ec2-44-220

i-0bc9fb84cdde8e83a (HomeMadePickles)

Details Status and alarms Monitoring Security Networking Storage Tags

Instance summary Info

Instance ID: i-0bc9fb84cdde8e83a Public IPv4 address: 44.220.144.100 [open address] Instance state: Running Public IPv6 address: - Public DNS: ec2-44-220-144-100.compute-1.amazonaws.com [open address]

© 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

aws Search [Alt+S] United States (N. Virginia) rsoaccount-new/67fce6a7985d0fc36953d566 @ rsosandboxnew91

EC2 Instances i-0bc9fb84cdde8e83a Connect to instance

Connect Info

Connect to an instance using the browser-based client.

EC2 Instance Connect Session Manager SSH client EC2 serial console

Instance ID: i-0bc9fb84cdde8e83a (HomeMadePickles)

Connect using a Public IP Connect using a public IPv4 or IPv6 address

Public IPv4 address: 44.220.144.100 IPv6 address

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

https://us-east-1.console.aws.amazon.com/ec2/instance/connect/sh?region=us-east-1&connType=standard&instanceId=i-0bc9fb84cdde8e83a&ossUser=ec2-user&sshPort=22&addressFamily=ipv4

aws Search [Alt+S] United States (N. Virginia) rsoaccount-new/67fce6a7985d0fc36953d566 @ rsosandboxnew91

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

(ec2-user@ip-172-31-25-32 ~)\$

Make Chrome faster

Memory Saver frees up memory from inactive tabs so it can be used by active tabs and other apps

Turn on No thanks

i-0bc9fb84cdde8e83a (HomeMadePickles)

Public IPs: 44.220.144.100 Private IPs: 172.31.25.32

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

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**
- **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/Anushaanu-ammu/newUpdates-home>' .

This will download the Project to Ec2 instance .

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

```
cd InstantLibrary
```

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

Run the Flask Application

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



```
aws | #### | Search [Alt+S] | X | 🔍 | ⓘ | United States (N. Virginia) | rsoaccount-new/67fce6a7985d0fc36953d566@rso sandboxnew91

[ec2-user@ip-172-31-25-32 ~]$ sudo yum update -y
Amazon Linux 2023 Kernel Livepatch repository
Dependencies resolved.
Nothing to do.
Complete!
[ec2-user@ip-172-31-25-32 ~]$ sudo yum install python3 git -y
Last metadata expiration check: 0:00:30 ago on Thu Jul  3 07:28:40 2025.
Package python3-3.9.23-1.amzn2023.0.1.x86_64 is already installed.
Dependencies resolved.

      Package          Architecture      Version       Repository      Size
Installing:
git                  x86_64        2.47.1-1.amzn2023.0.3      amazonlinux    52 k
Installing dependencies:
git-core             x86_64        2.49.1-1.amzn2023.0.3      amazonlinux   4.5 M
git-core-doc         noarch       2.49.1-1.amzn2023.0.3      amazonlinux   2.0 M
perl_Error           noarch       1:0.17029-5.amzn2023.0.2      amazonlinux   41 k

i-0bc9fb84cdde8e83a (HomeMadePickles)
Public IPs: 44.220.144.100 Private IPs: 172.31.25.32
```

```
(9/8): git-core-2.47.1-1.amzn2023.0.3.x86_64.rpm
(6/8): perl-Git-2.47.1-1.amzn2023.0.3.noarch.rpm
(7/8): perl-TermReadkey-2.38-9.amzn2023.0.2.x86_64.rpm
(8/8): perl-lib-0.65-477.amzn2023.0.7.x86_64.rpm

Total 44 MB/s | 7.5 MB 00:00

Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
Preparing : 1/1
Installing : git-core-2.47.1-1.amzn2023.0.3.x86_64 1/8
Installing : git-core-doc-2.47.1-1.amzn2023.0.3.noarch 2/8
Installing : perl-Git-2.47.1-1.amzn2023.0.3.noarch 3/8
Installing : perl-TermReadkey-2.38-9.amzn2023.0.2.x86_64 4/8
Installing : perl-File-Find-1.37-477.amzn2023.0.7.noarch 5/8
Installing : perl-Error-1.0.17029-5.amzn2023.0.2.noarch 6/8
Installing : perl-Git-2.47.1-1.amzn2023.0.3.noarch 7/8
Installing : perl-lib-0.65-477.amzn2023.0.7.x86_64 8/8
Running scriptlets: 1/1
Verifying : git-2.47.1-1.amzn2023.0.3.x86_64 1/8
Verifying : git-2.47.1-1.amzn2023.0.3.noarch 2/8
Verifying : perl-Git-2.47.1-1.amzn2023.0.3.noarch 3/8
Verifying : perl-TermReadkey-2.38-9.amzn2023.0.2.x86_64 4/8
Verifying : perl-File-Find-1.37-477.amzn2023.0.7.noarch 5/8
Verifying : perl-Error-1.0.17029-5.amzn2023.0.2.noarch 6/8
Verifying : perl-Git-2.47.1-1.amzn2023.0.3.noarch 7/8
Verifying : perl-lib-0.65-477.amzn2023.0.7.x86_64 8/8

Installed:
git-2.47.1-1.amzn2023.0.3.x86_64
perl-Error-1.0.17029.5.amzn2023.0.2.noarch
perl-TermReadkey-2.38-9.amzn2023.0.2.x86_64

Uninstalling:
git-core-2.47.1-1.amzn2023.0.3.x86_64
perl-File-Find-1.37-477.amzn2023.0.7.noarch
perl-lib-0.65-477.amzn2023.0.7.x86_64
```

```
aws | [Q] Search [Alt+S] | United States (N. Virginia) | rsoaccount-new/67fce6a7985d0fc36953d566 @rsoandboxnew91

export AWS_REGION=us-east-1
export FLASK_APP=app.py
sudo python3 -m flask run --host=0.0.0.0 --port=80
-bash: cd: newUpdates-home: No such file or directory
* Tip: There are .env files present. Install python-dotenv to use them.
Usage: python -m flask run [OPTIONS]
Try 'python -m flask run --help' for help.

Error: While importing 'app', an ImportError was raised:

Traceback (most recent call last):
  File "/usr/local/lib/python3.9/site-packages/flask/cli.py", line 245, in locate_app
    import _module_name
  File "/home/ec2-user/newUpdates-home/app.py", line 10, in <module>
    from dotenv import load_dotenv
ModuleNotFoundError: No module named 'dotenv'

[ec2-user@ip-172-31-25-32 newUpdates-home]$ sudo pip3 install python-dotenv
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
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system package manager. It is recommended to use a virtual environment instead: https://pip.pypa.io/warnings/venv
[ec2-user@ip-172-31-25-32 newUpdates-home]$ export AWS_REGION=us-east-1
export FLASK_APP=app.py
sudo python3 -m flask run --host=0.0.0.0 --port=80
2025-07-03 08:04:13,993 - INFO - Found credentials from IAM Role: EC2_DynamoDB_Role
* Debug mode: off
2025-07-03 08:04:14,095 - INFO - WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:80
* Running on http://172.31.25.32:80
2025-07-03 08:04:14,096 - INFO - Press CTRL+C to quit
```

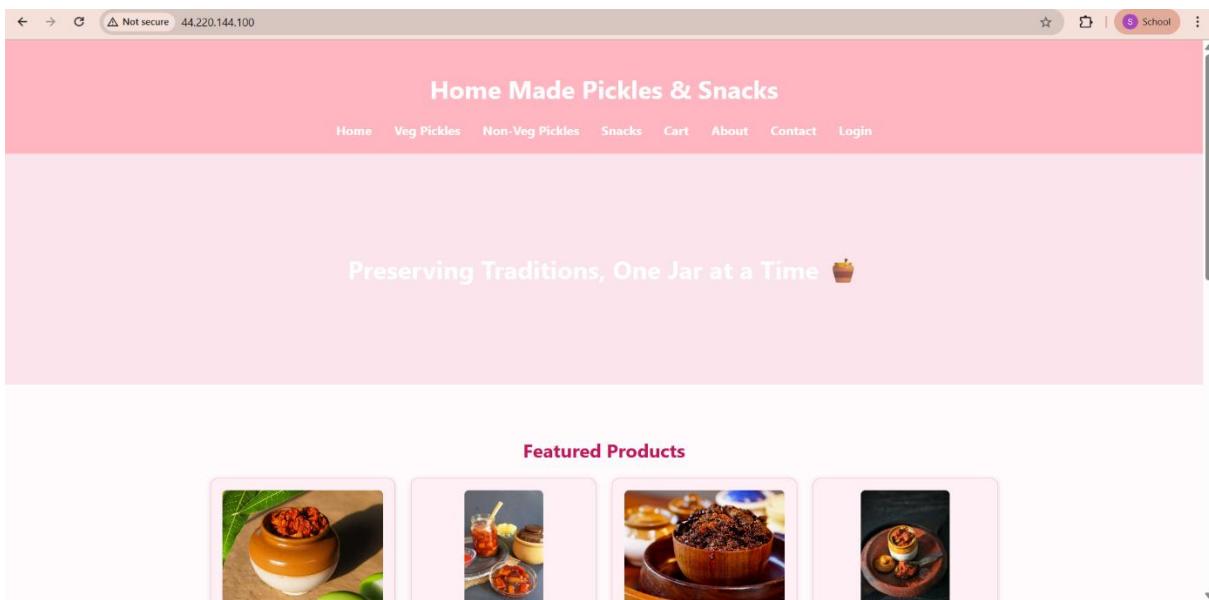
Access the website through:

Public IPs: <http://44.220.144.100/>

Milestone 8. Testing and Deployment

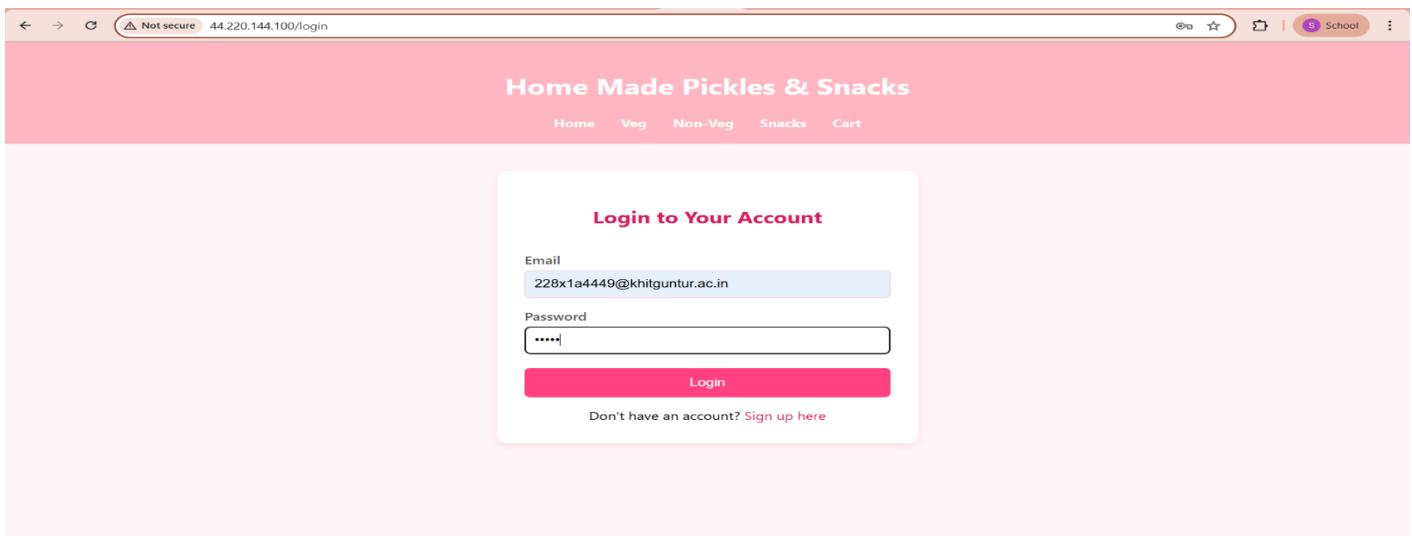
Activity 8.1 : Conduct functional testing to verify user signup, login, buy/sell stocks and notifications.

HOME PAGE:



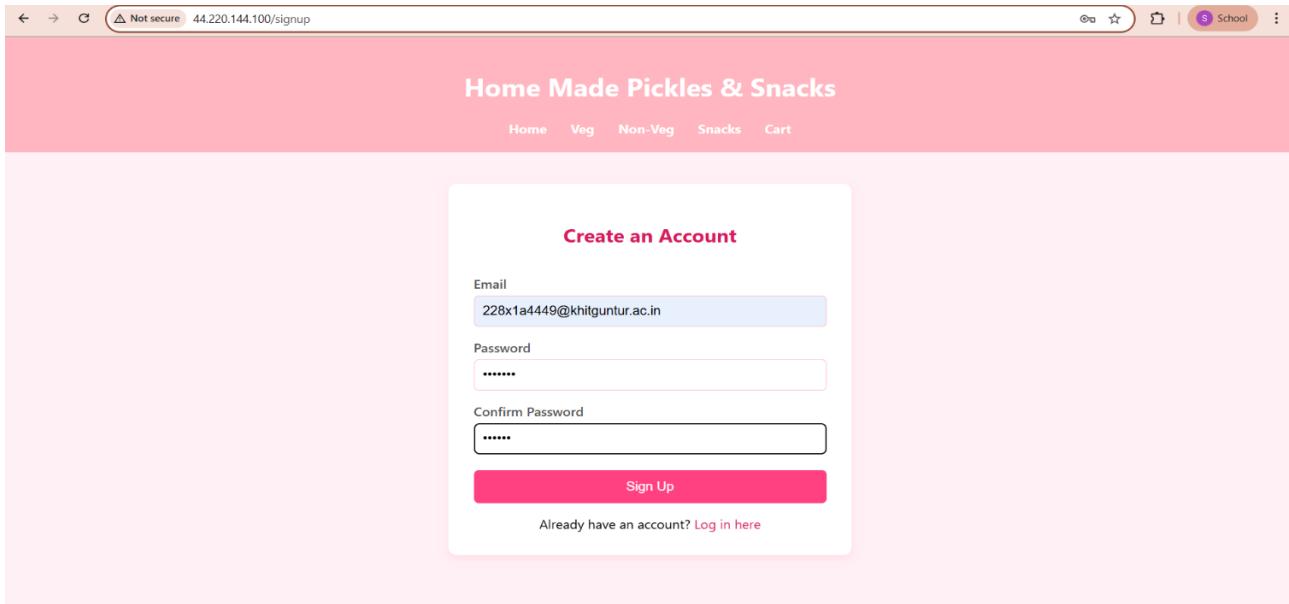
The screenshot shows a web browser window with the URL <http://44.220.144.100> in the address bar. The page has a pink header with the title "Home Made Pickles & Snacks". Below the header is a navigation menu with links: Home, Veg Pickles, Non-Veg Pickles, Snacks, Cart, About, Contact, and Login. The main content area features a large pink banner with the text "Preserving Traditions, One Jar at a Time" and a small jar icon. Below the banner is a section titled "Featured Products" showing four images of different jars of pickles and snacks.

LOGIN PAGE :

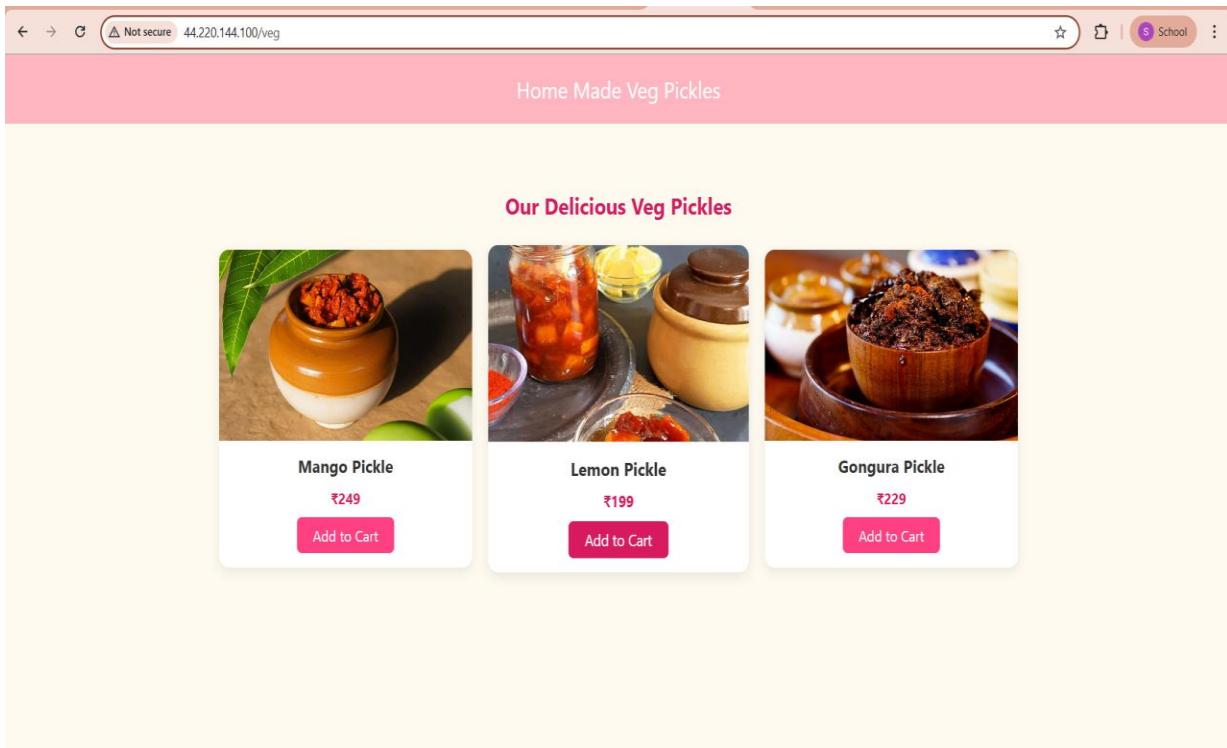


The screenshot shows a web browser window with the URL <http://44.220.144.100/login> in the address bar. The page has a pink header with the title "Home Made Pickles & Snacks". Below the header is a navigation menu with links: Home, Veg, Non-Veg, Snacks, and Cart. The main content area features a login form titled "Login to Your Account". The form includes fields for "Email" (containing "228x1a4449@khitguntur.ac.in") and "Password" (containing "....."). A "Login" button is at the bottom of the form. Below the form, a link says "Don't have an account? [Sign up here](#)".

SIGNUP PAGE :



VEG PAGE :



NON VEG PAGE :

← → ⌛ Not secure 44.220.144.100/nonveg ☆ 🗑️ School ⋮

Home Made Non-Veg Pickles

Delicious Non-Veg Pickles



Chicken Pickle
₹349

Add to Cart



Fish Pickle
₹329

Add to Cart



Prawns Pickle
₹399

Add to Cart

SNACKS PAGE :

← → ⌛ Not secure 44.220.144.100/snacks ☆ 🗑️ School ⋮

Crispy & Tasty Snacks

Freshly Made Traditional Snacks



Murukulu
₹149

Add to Cart



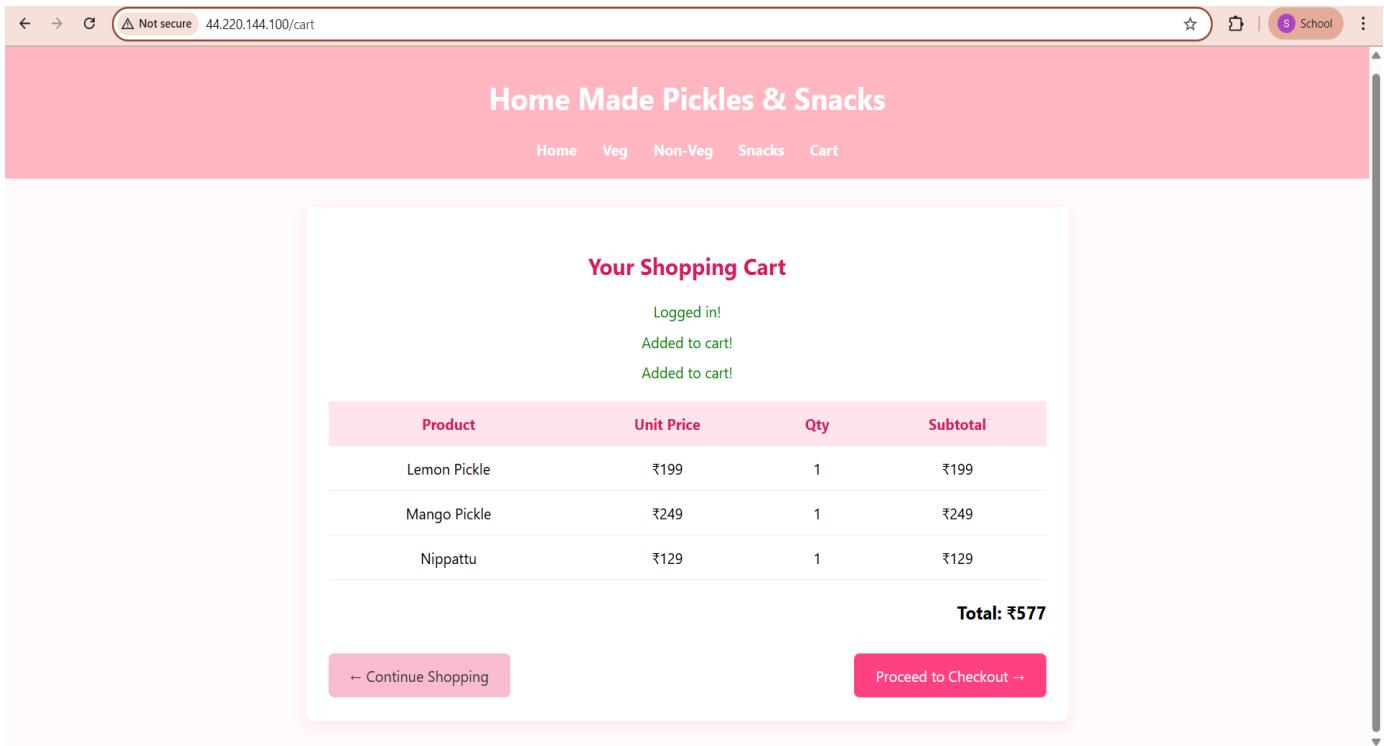
Nippattu
₹129

Add to Cart



Hot Maida Biscuit
₹109

Add to Cart

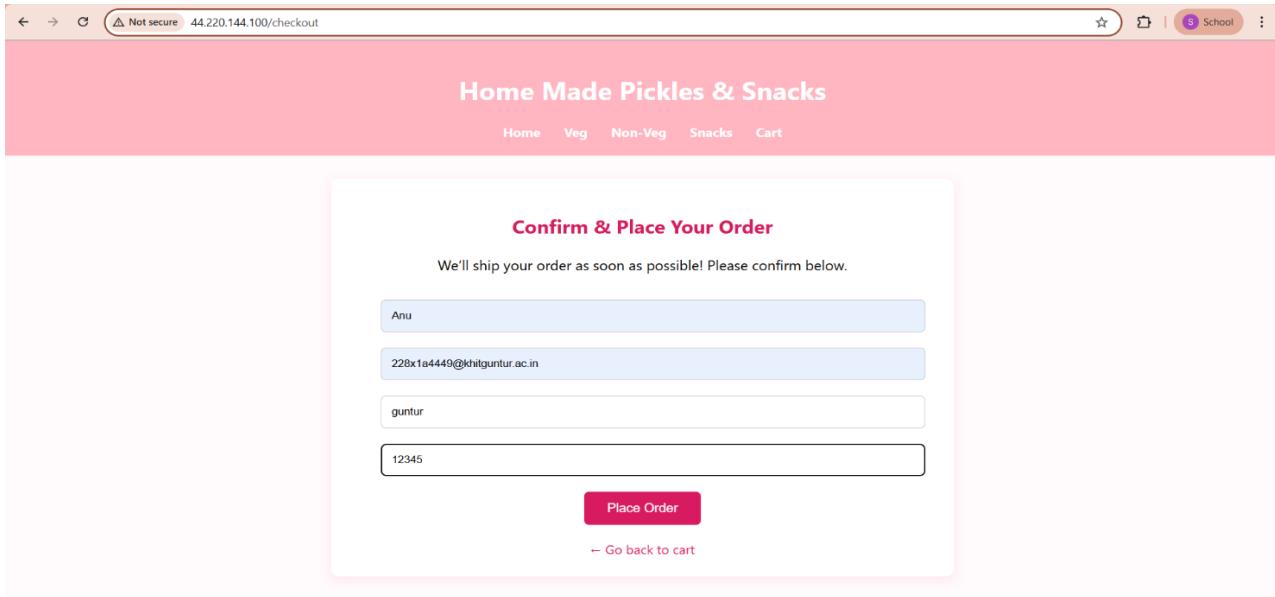
CART PAGE :


The screenshot shows a web browser window with the URL 44.220.144.100/cart. The page title is "Home Made Pickles & Snacks". A navigation bar at the top includes links for Home, Veg, Non-Veg, Snacks, and Cart. The main content area is titled "Your Shopping Cart". It displays a table with the following data:

Product	Unit Price	Qty	Subtotal
Lemon Pickle	₹199	1	₹199
Mango Pickle	₹249	1	₹249
Nippattu	₹129	1	₹129

Total: ₹577

Below the table are two buttons: "← Continue Shopping" and "Proceed to Checkout →". Above the table, there are three success messages: "Logged in!", "Added to cart!", and "Added to cart!".

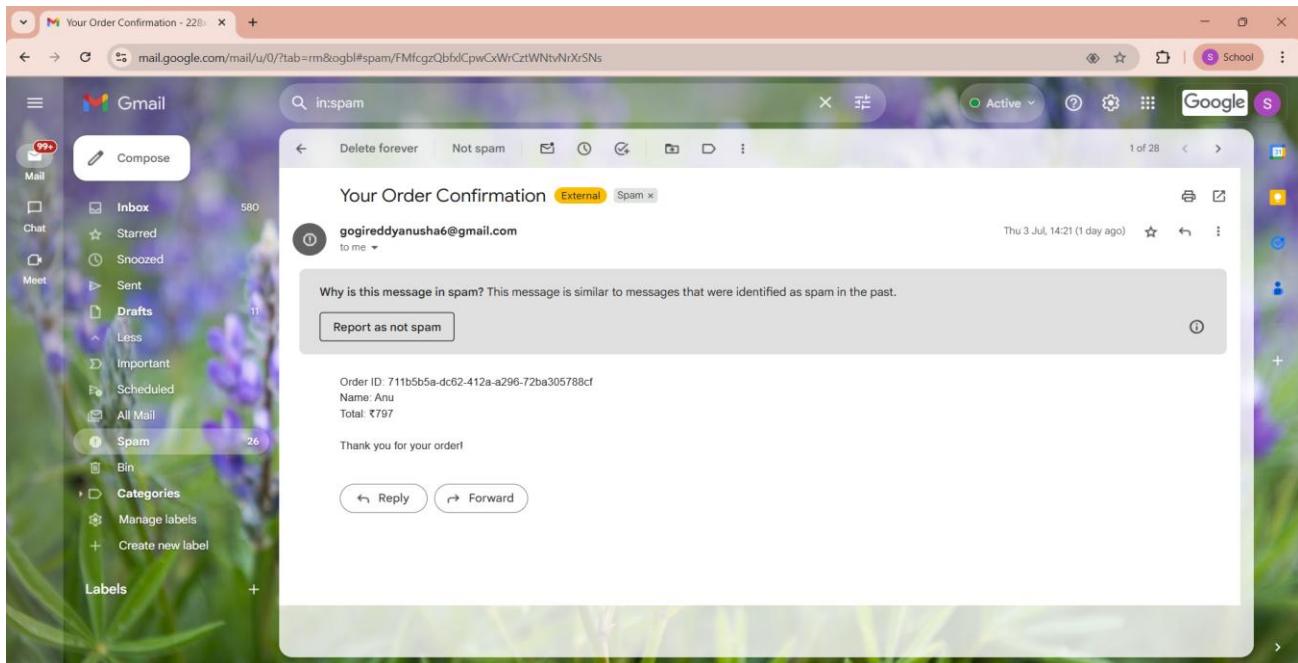
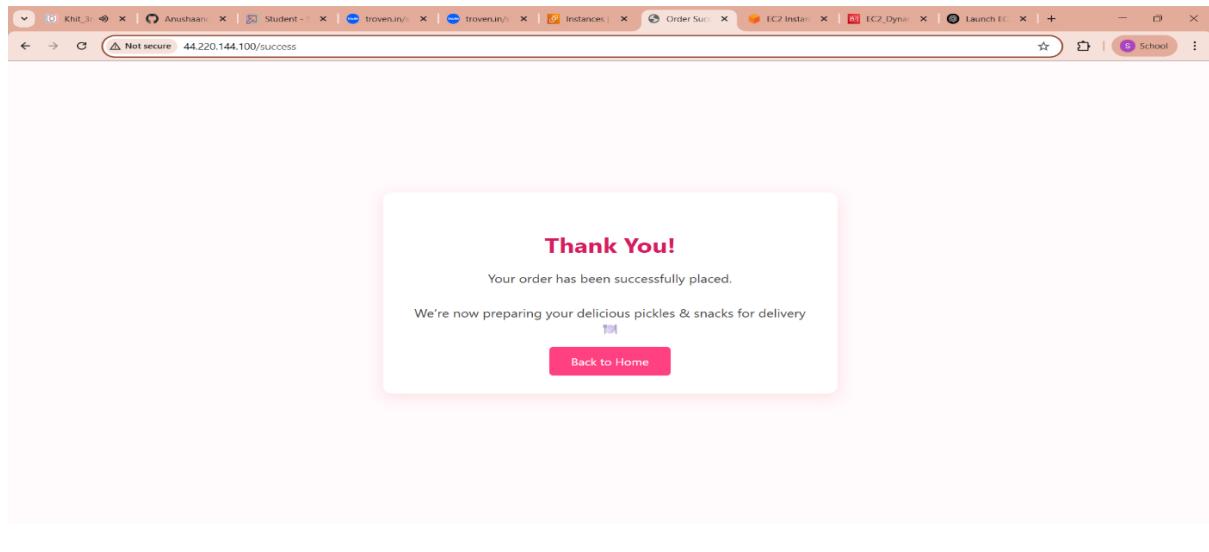
CHECK OUT PAGE :


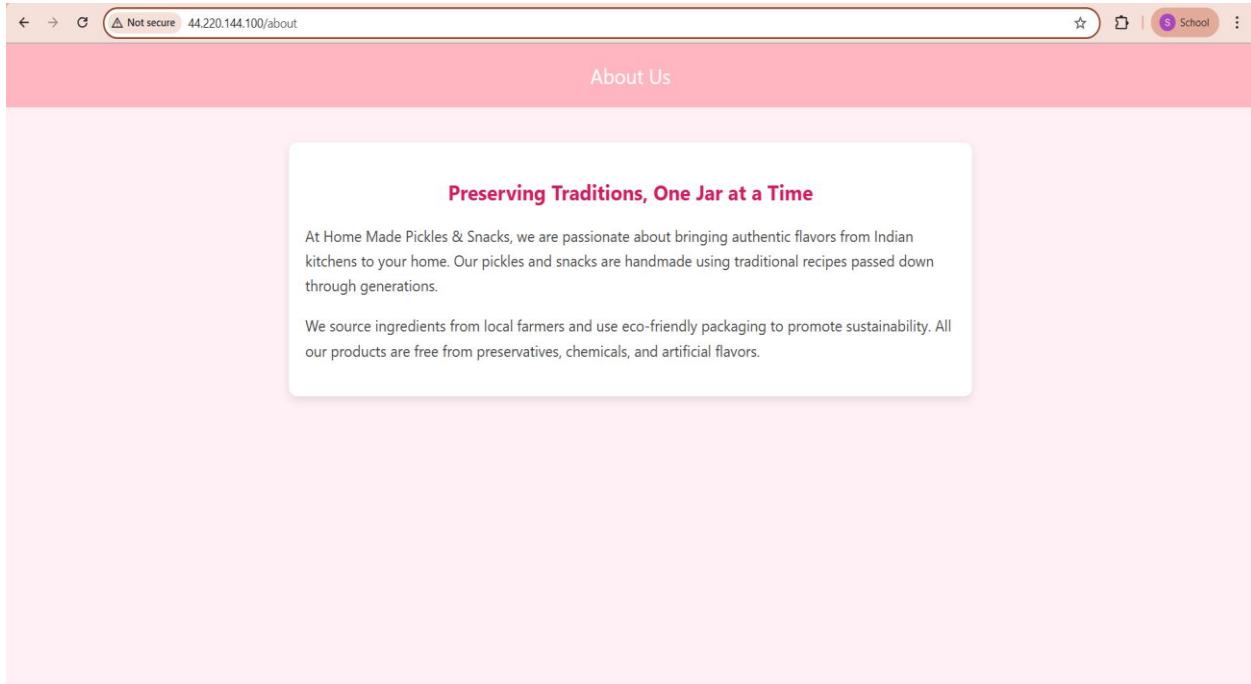
The screenshot shows a web browser window with the URL 44.220.144.100/checkout. The page title is "Home Made Pickles & Snacks". A navigation bar at the top includes links for Home, Veg, Non-Veg, Snacks, and Cart. The main content area is titled "Confirm & Place Your Order". It displays a form with the following fields:

- Name: Anu
- Email: 228x1a4449@khitguntur.ac.in
- Address: guntur
- Pincode: 12345

At the bottom of the form are two buttons: "Place Order" and "← Go back to cart". Below the form, a message states: "We'll ship your order as soon as possible! Please confirm below."

SUCCESS PAGE :



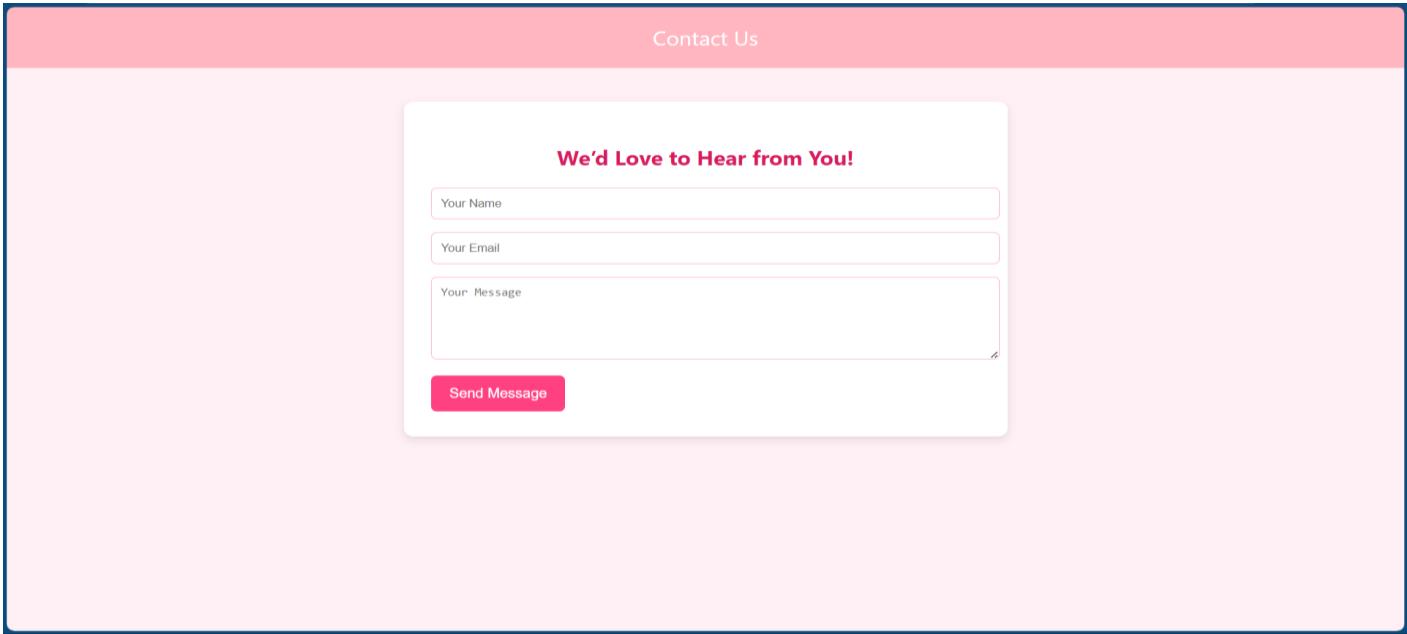
ABOUT PAGE :

The screenshot shows a web browser window with a pink header bar. The title bar reads "About Us". The main content area has a white background with a central box containing text. At the top of this box is the heading "Preserving Traditions, One Jar at a Time". Below it is a paragraph about the company's focus on authentic Indian flavors and traditional recipes. Another paragraph discusses their commitment to sustainability and eco-friendly practices.

Preserving Traditions, One Jar at a Time

At Home Made Pickles & Snacks, we are passionate about bringing authentic flavors from Indian kitchens to your home. Our pickles and snacks are handmade using traditional recipes passed down through generations.

We source ingredients from local farmers and use eco-friendly packaging to promote sustainability. All our products are free from preservatives, chemicals, and artificial flavors.

CONTACT PAGE :

The screenshot shows a web browser window with a pink header bar. The title bar reads "Contact Us". The main content area has a white background with a central box containing text. At the top of this box is the heading "We'd Love to Hear from You!". Below it are three input fields: "Your Name", "Your Email", and "Your Message". At the bottom is a red "Send Message" button.

We'd Love to Hear from You!

Your Name

Your Email

Your Message

Send Message

Conclusion:

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

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

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

By combining time-honored recipes with modern e-commerce capabilities, this website not only preserves culinary heritage but also adapts to the digital age, ensuring that every jar of pickle or snack reaches customers with the same care and quality as a homemade meal. As the platform evolves, it stands ready to scale, introduce new product lines, and foster a community of food lovers united by a passion for authentic flavors.

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

.....THE END.....