**`** Citizen AI – Intelligent Citizen Engagement Platform

## Project Description:

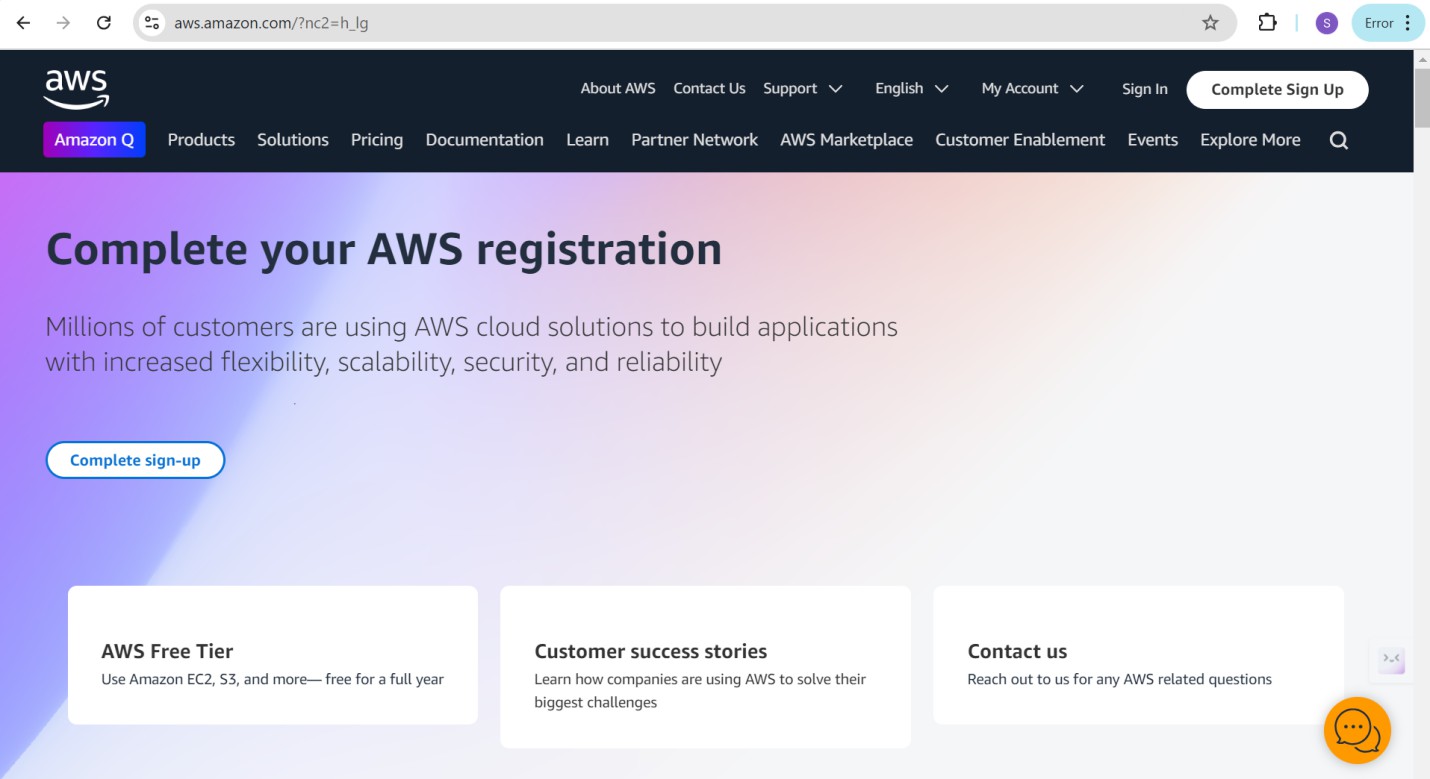
### Testing and Deployment

**Activity 8.1**: Conduct functional testing to verify user registration, login, book requests, and notifications.

# Milestone 1: AWS Account Setup and Login

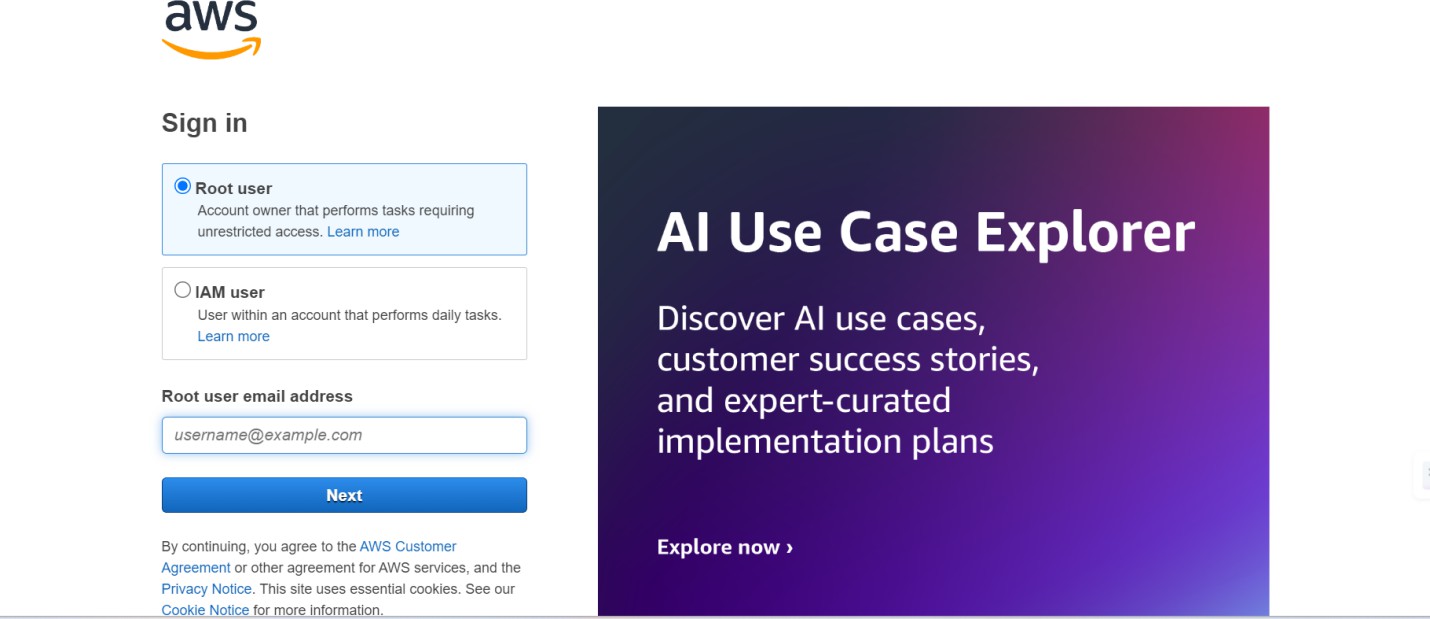
### Activity 1.1: Set up an AWS account if not already done.

* + - Sign up for an AWS account and configure billing settings.



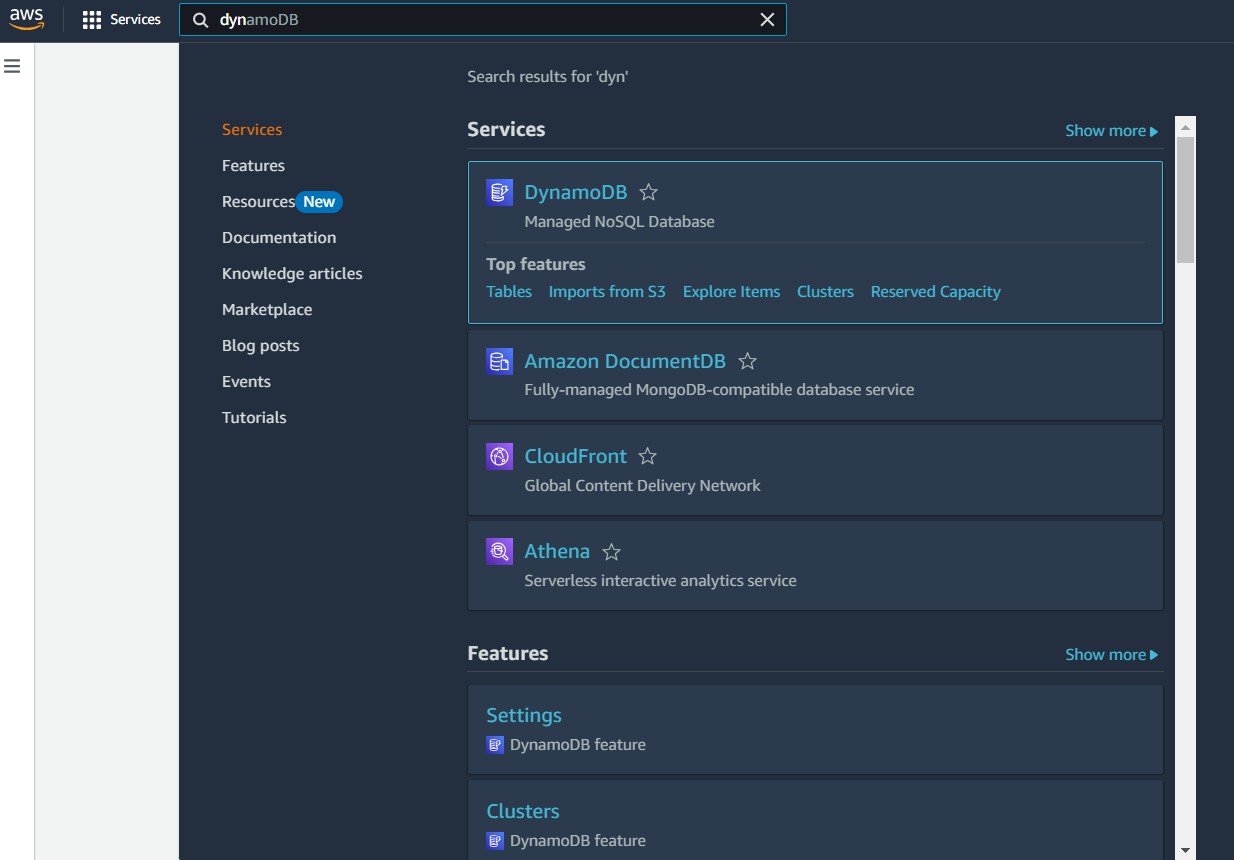
### Activity 1.2: Log in to the AWS Management Console

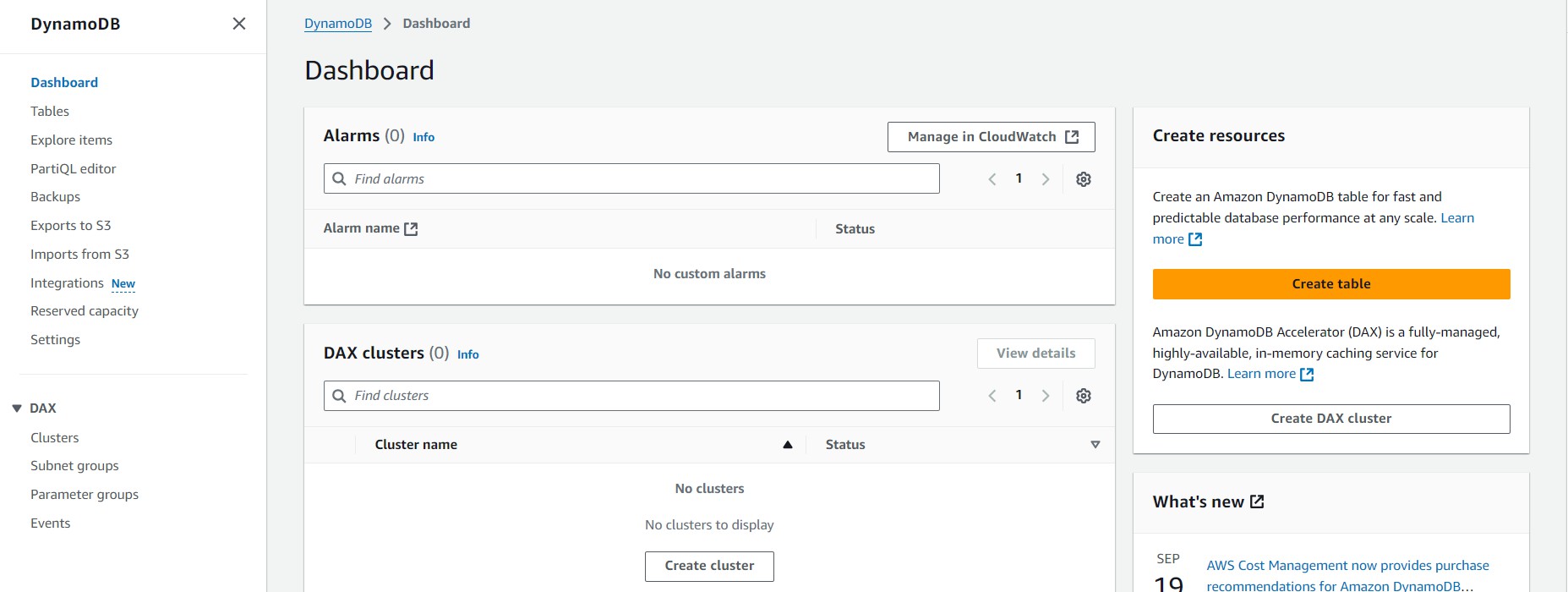
* + - After setting up your account, log in to the [AWS Management Console](https://aws.amazon.com/console/).

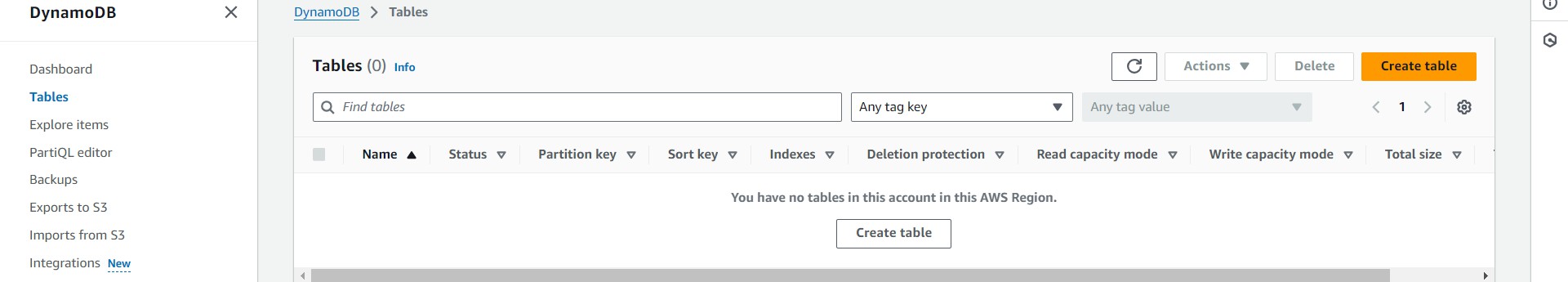


# Milestone 2: DynamoDB Database Creation and Setup

### Activity 2.1:Navigate to the DynamoDB

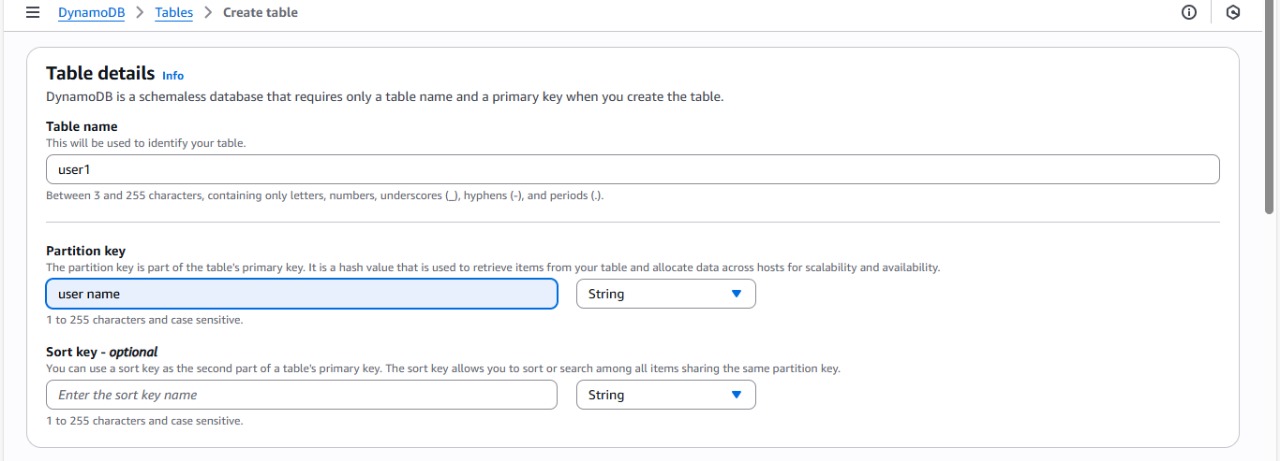
* + - In the AWS Console, navigate to DynamoDB and click on create tables.
    - 

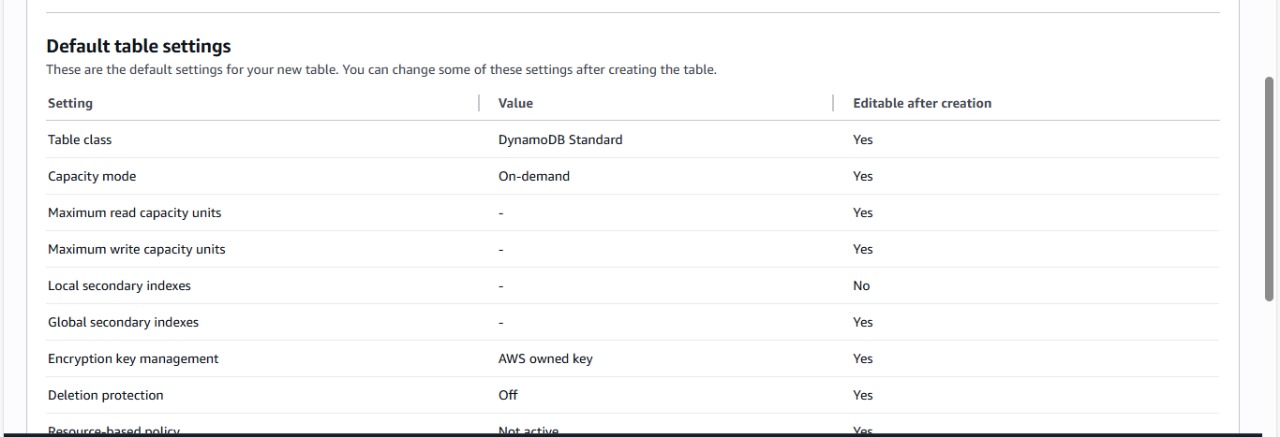


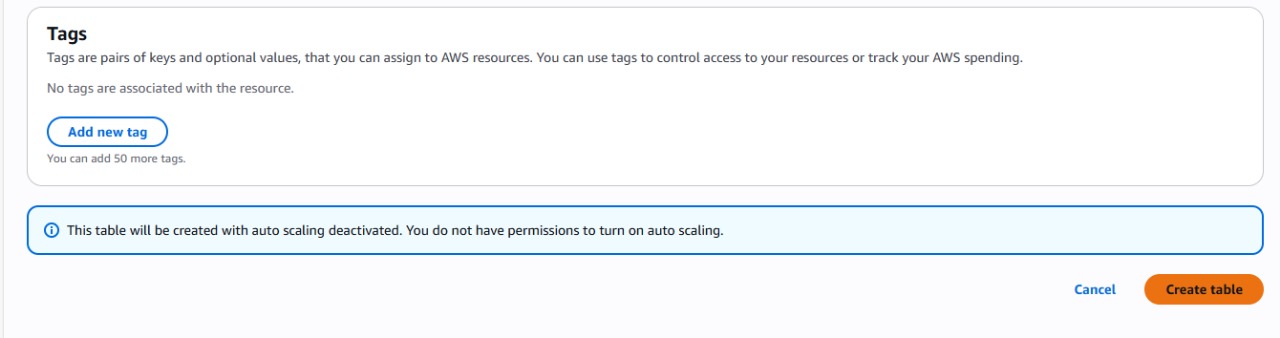


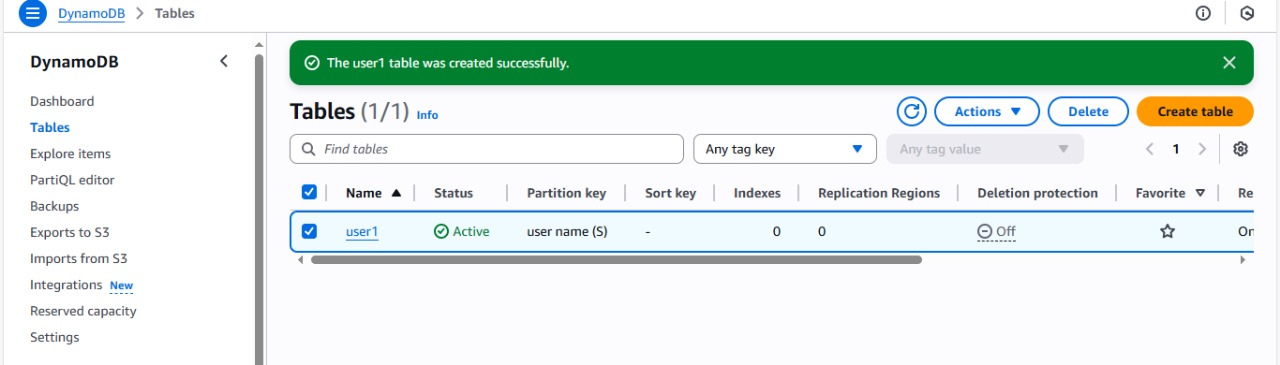
### Activity 2.2:Create a DynamoDB table

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

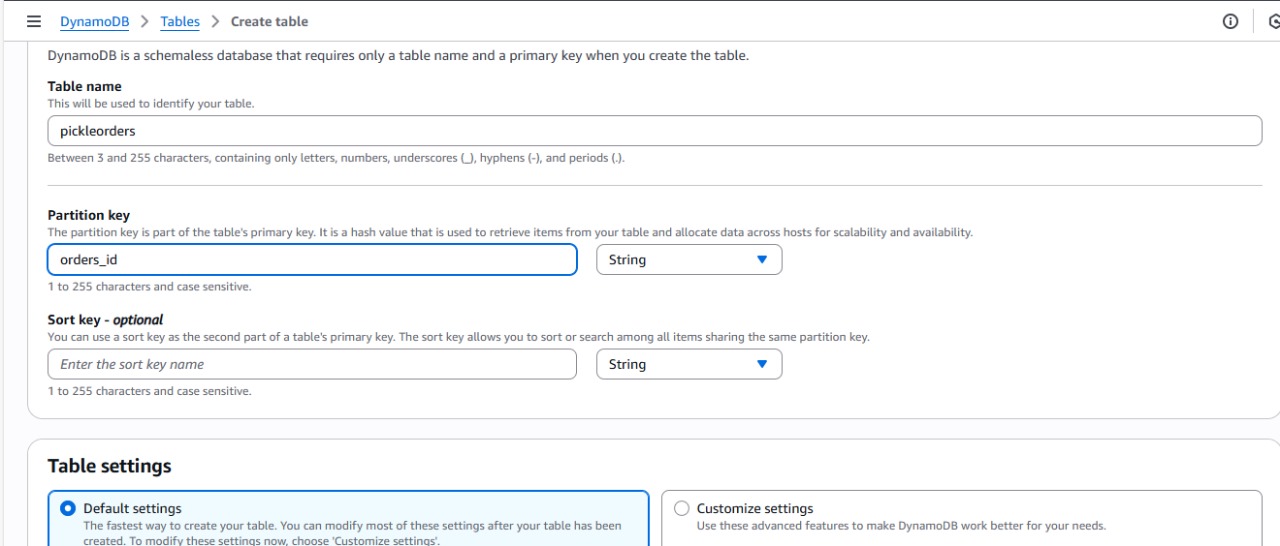


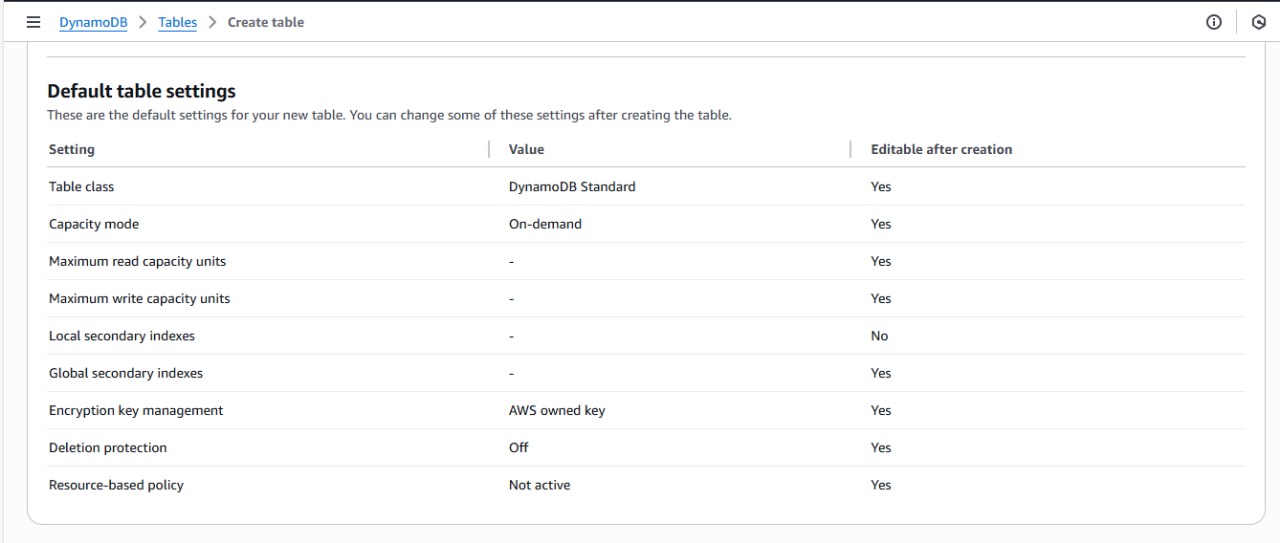


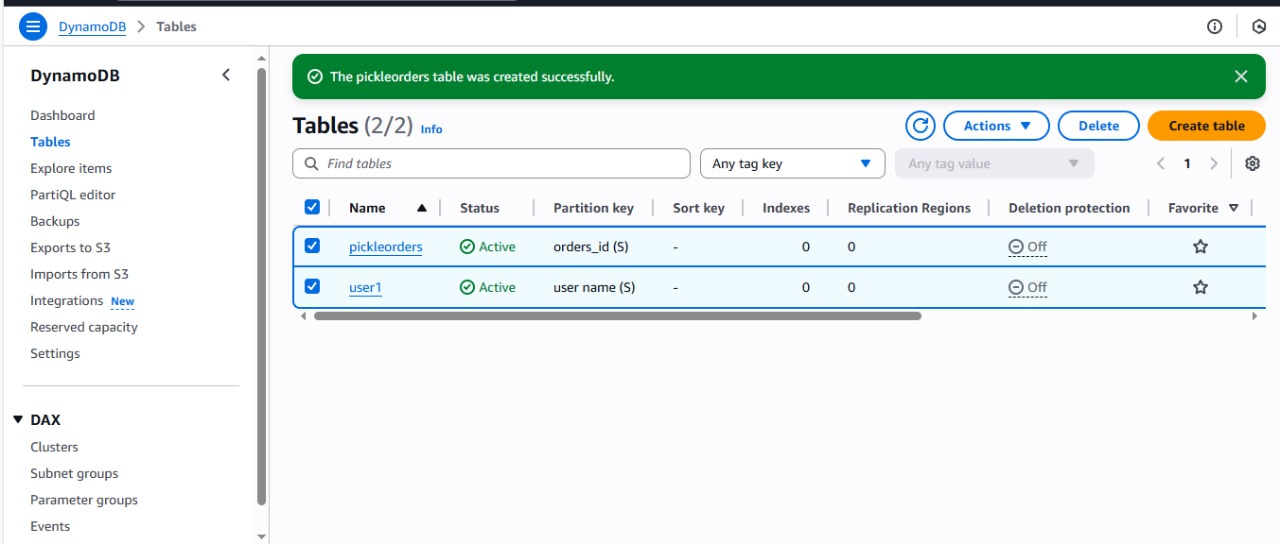




* Follow the same steps to create an Orders table with Order\_id as the primary key to store Order details.



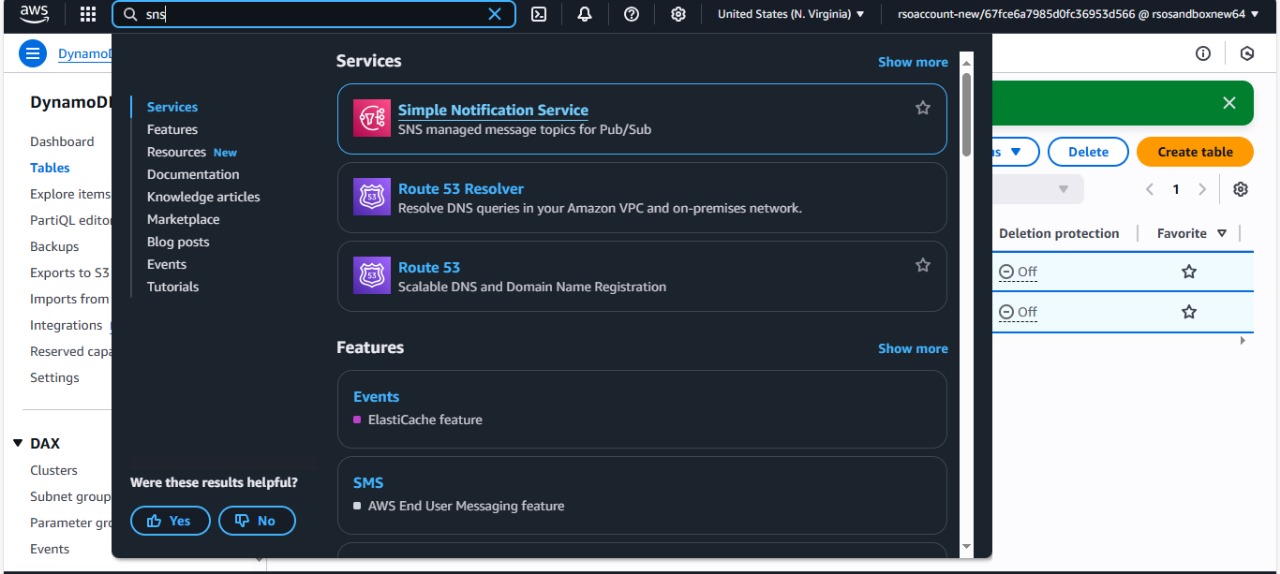


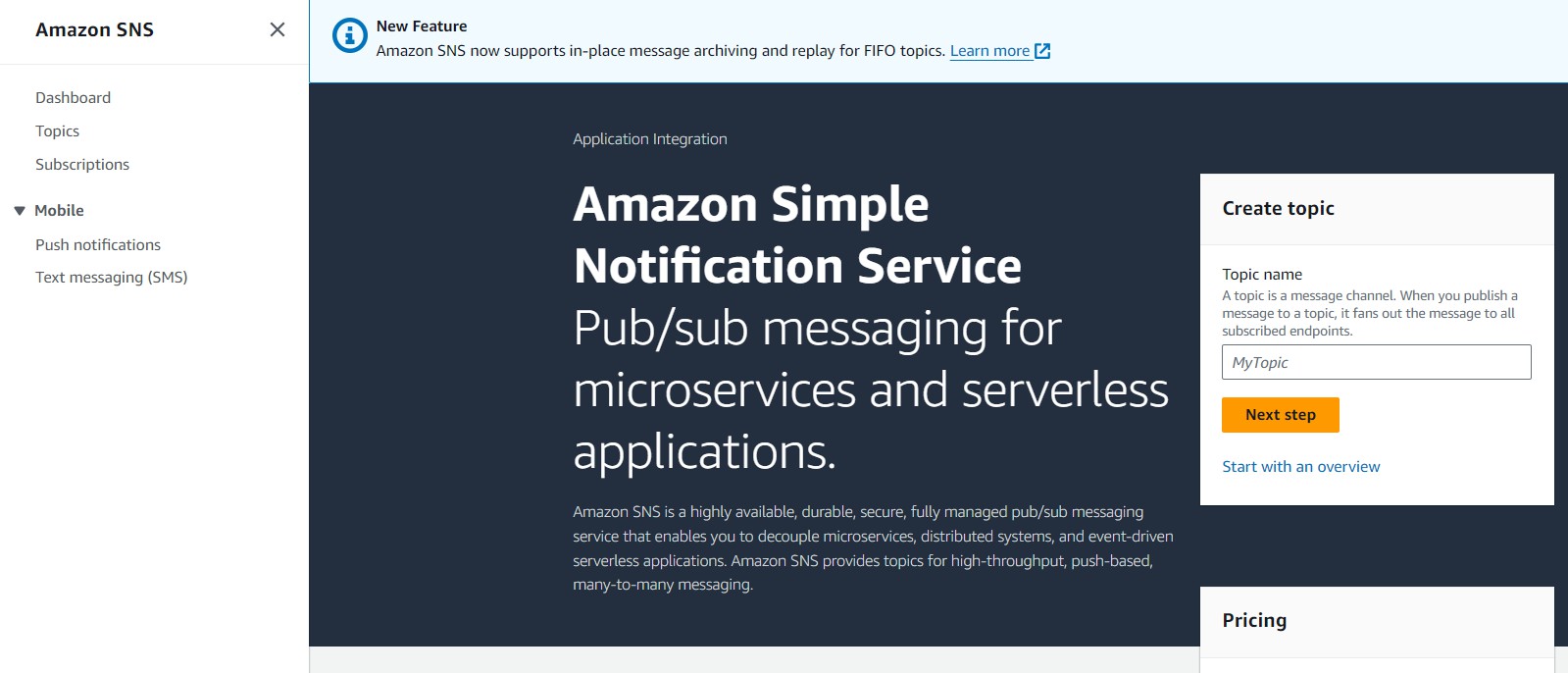


## Milestone 3: SNS Notification Setup

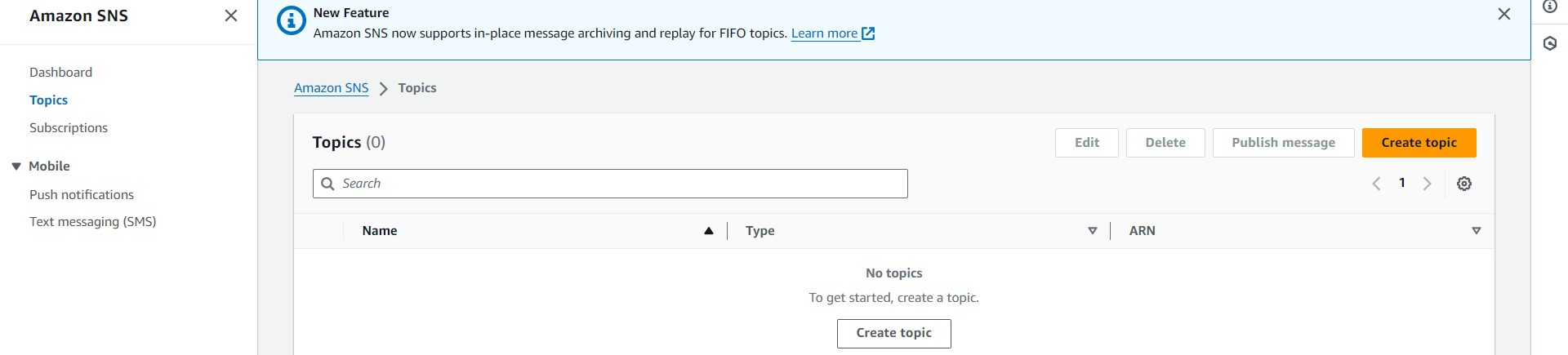
### Activity 3.1: Create SNS topics for sending email notifications to users and library staff.

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

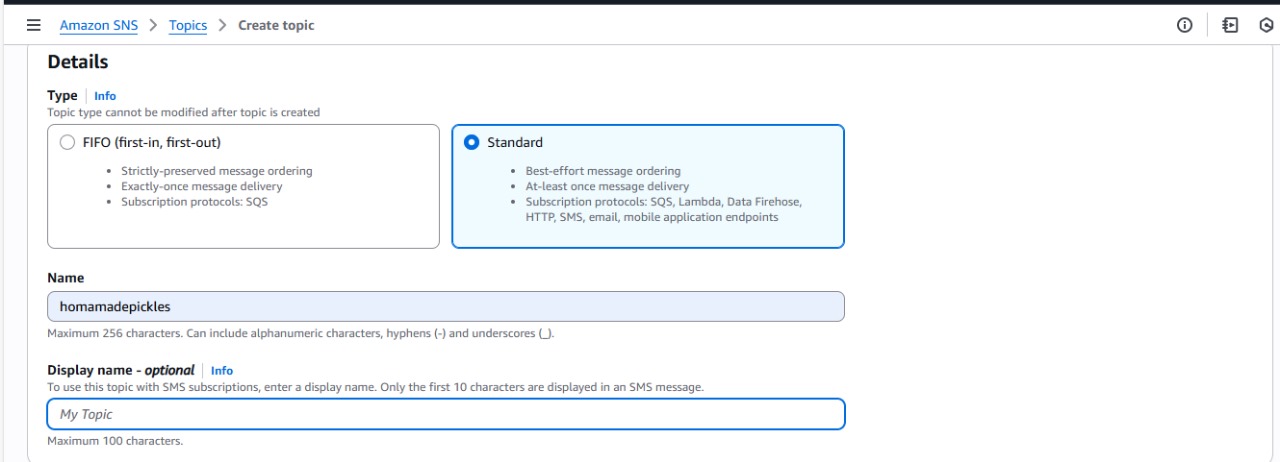


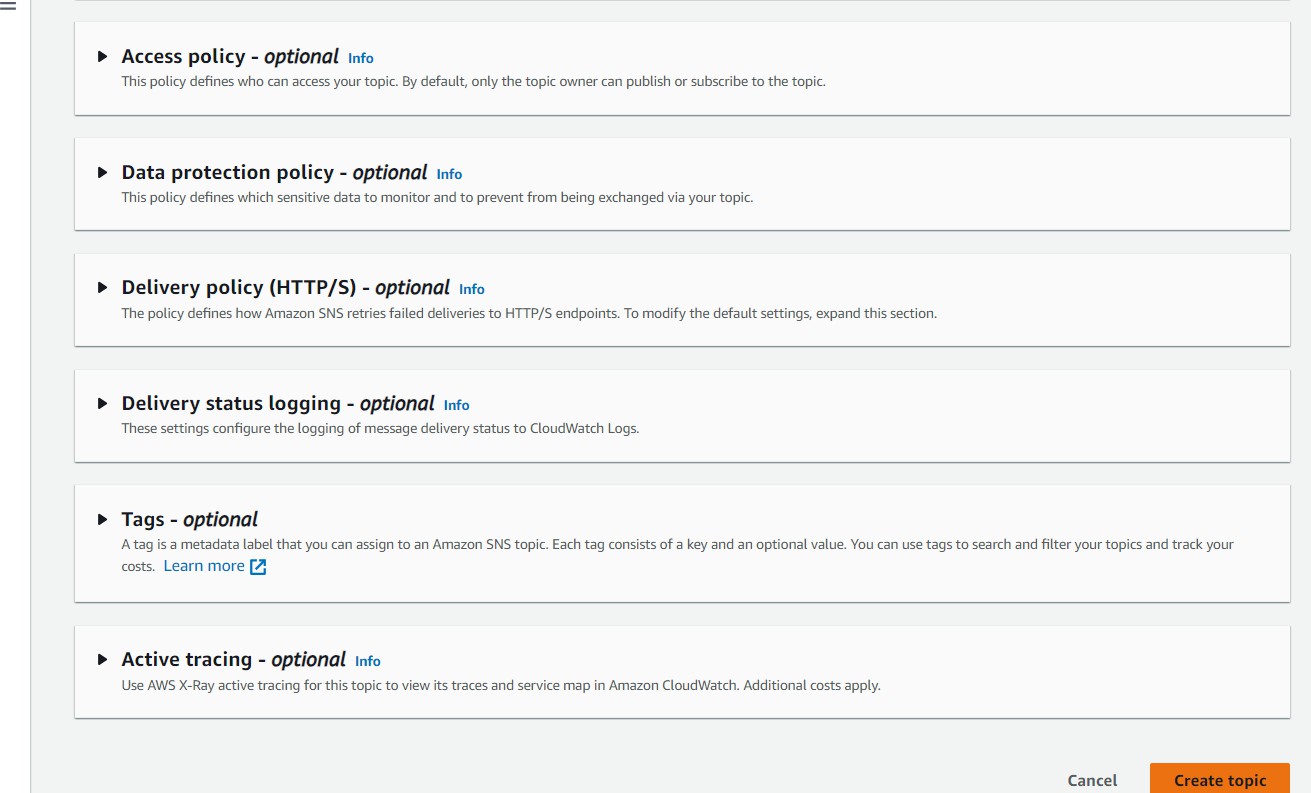


* + Click on **Create Topic** and choose a name for the topic.

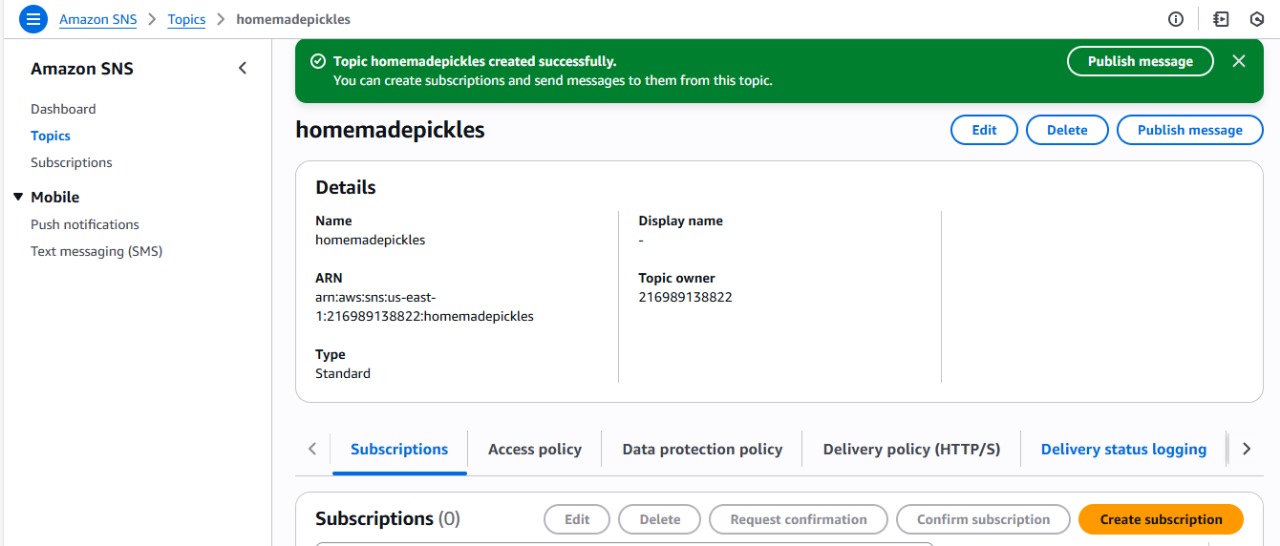


* + Choose Standard type for general notification use cases and Click on Create Topic.



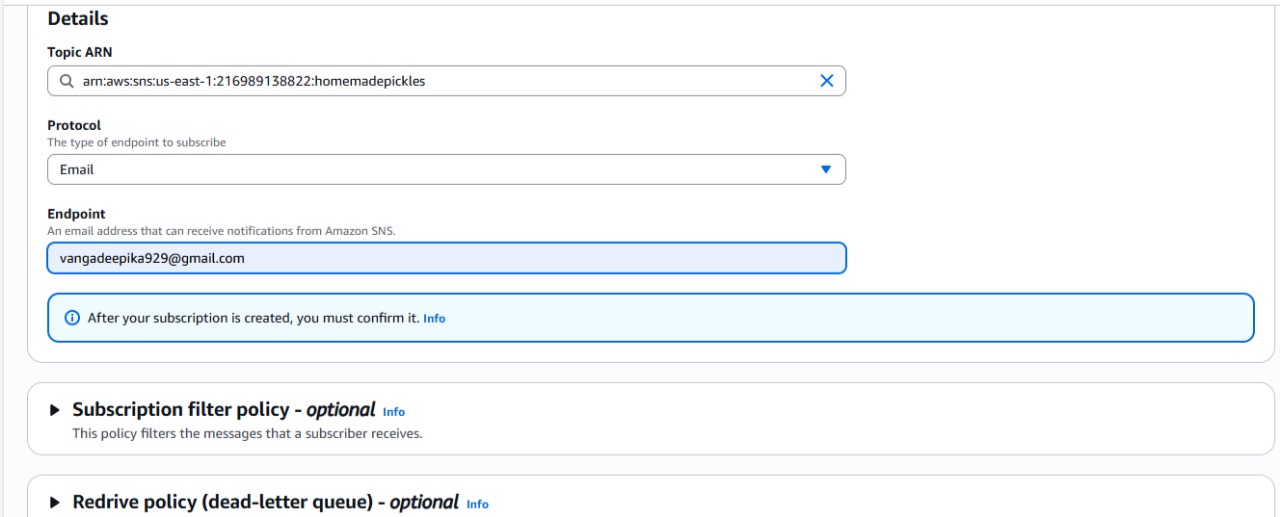


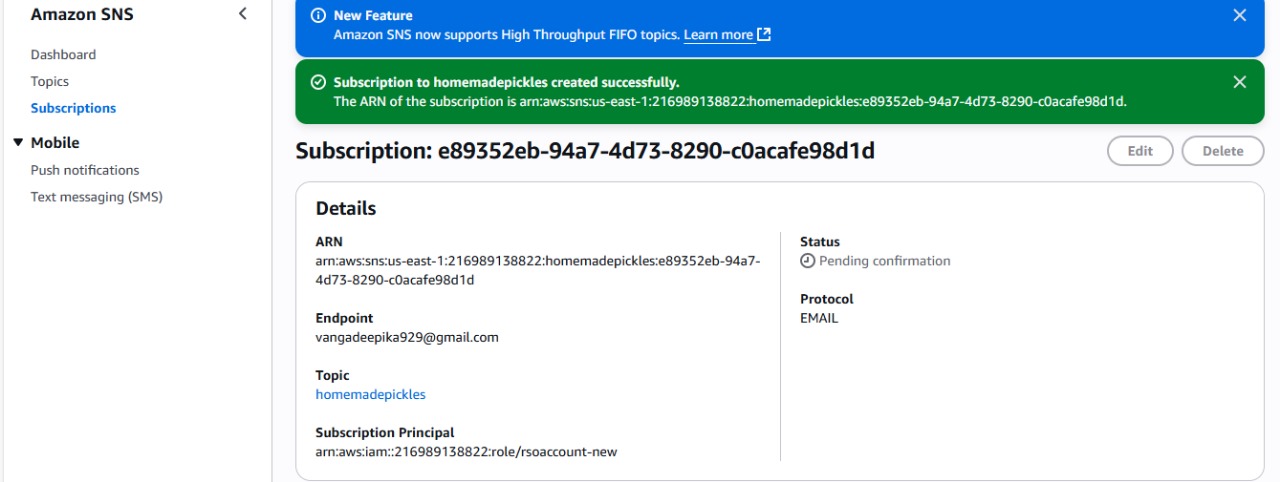
* + Configure the SNS topic and note down the **Topic ARN**.



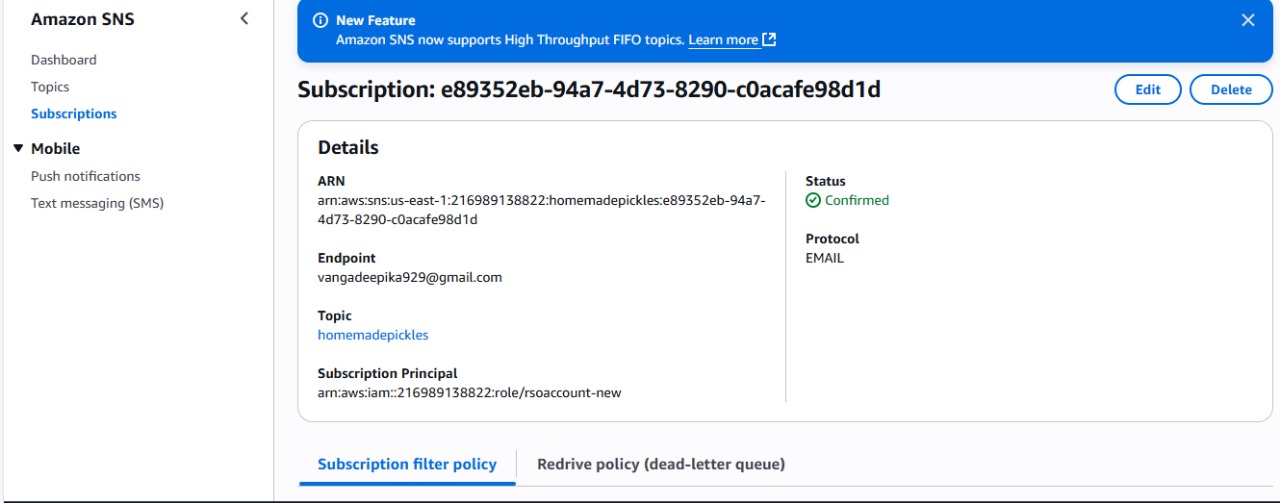
### Activity 3.2: Subscribe users and staff to relevant SNS topics to receive real-time notifications when a order equest is made.

* + Subscribe users (or customers) to this topic via Email. When a order request is made, notifications will be sent to the subscribed emails**.**

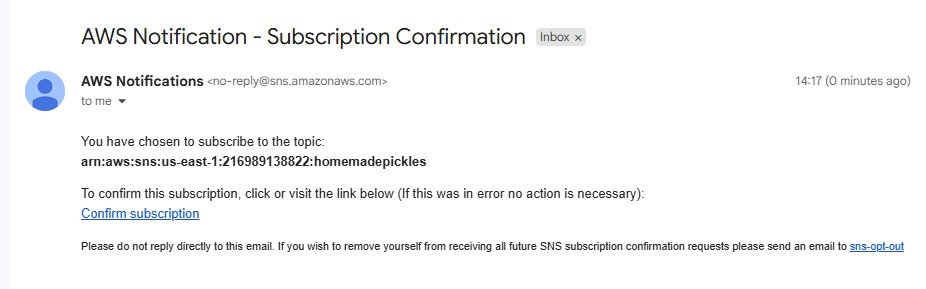
****

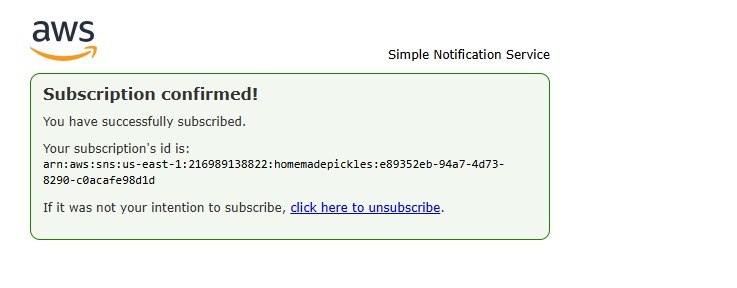


* + After subscription request for the mail confirmation



* + Navigate to the subscribed Email account and Click on the confirm subscription in the AWS Notification- Subscription Confirmation mail.



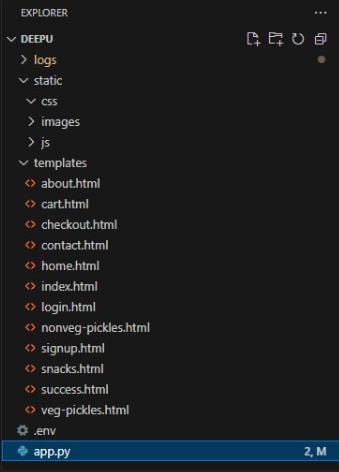


* + Successfully done with the SNS mail subscription and setup, now store the ARN link.

# Milestone 4:Backend Development and Application Setup

### Activity 4.1: Develop the backend using Flask

* + File Explorer Structure

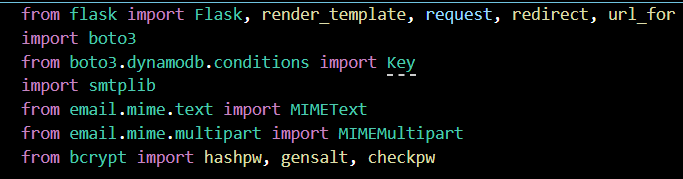


**Description:**

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**.**

## Description of the code :

### Flask App Initialization

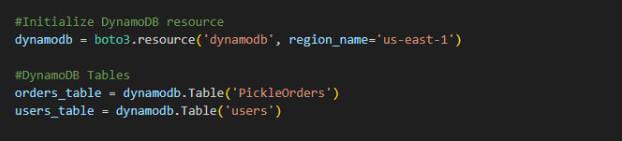
****

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



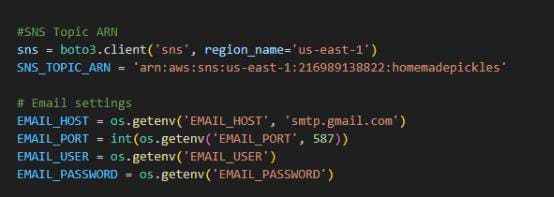
**Description:** initialize the Flask application instance using Flask( name ) to start building the web app.

### Dynamodb Setup:

****

**Description:** initialize the DynamoDB resource for the ap-south-1 region and set up access to the Users and Requests tables for storing user details and book requests.

### SNS Connection

****

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

## Routes for Web Pages

.

### Register Route:

* 

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

### login Route (GET/POST):

* 

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

### Home routes:

****

**Description:**

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

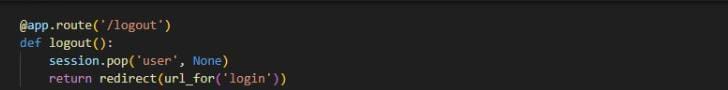
.

* **Request Routes:**



**Description:** define /request-form route to capture book request details from users, store the request in DynamoDB, send a thank-you email to the user, notify the admin, and confirm submission with a success message.

**LogoutRoute**:

****

**Description:** define /logout route to render the exit.html page when the user chooses to leave or close the application.

**Deployment Code:**

****

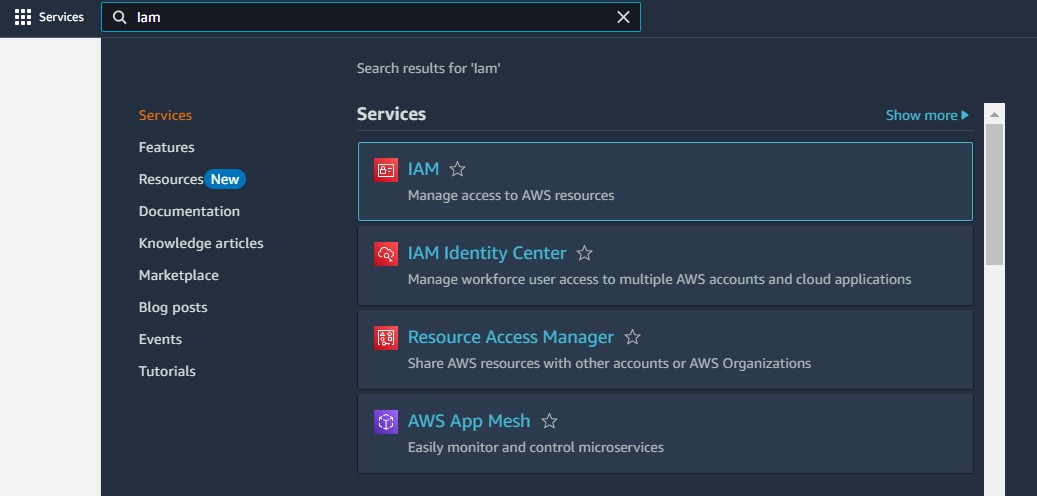
**Description:** start the Flask server to listen on all network interfaces (0.0.0.0) at port 80

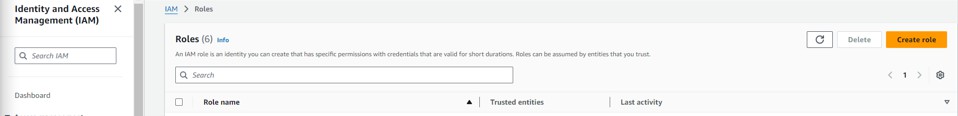
with debug mode enabled for development and testing.

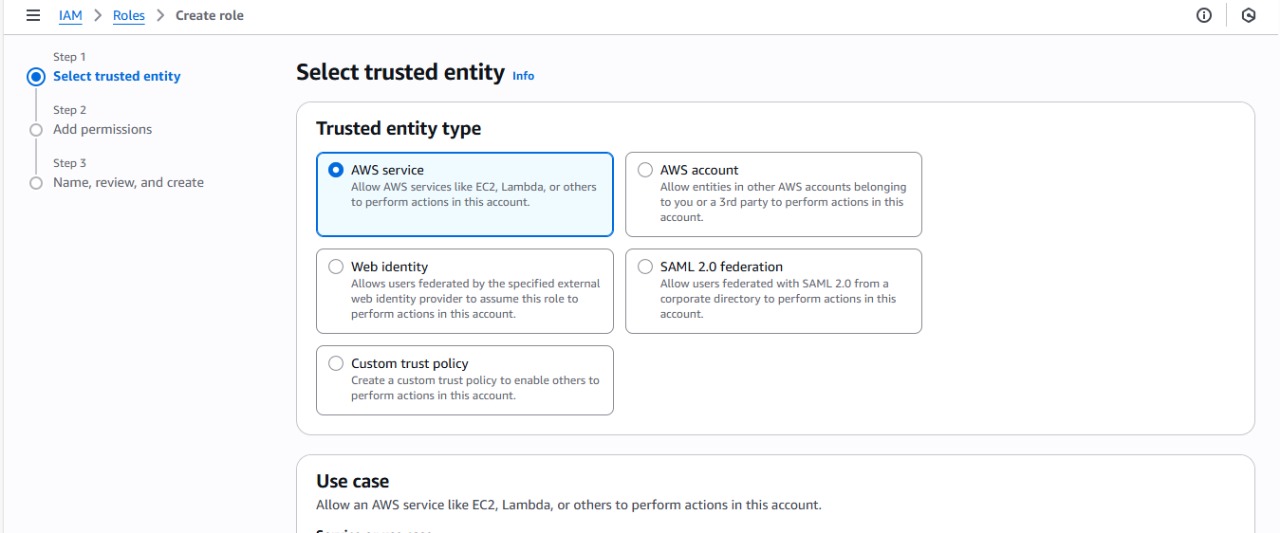
# Milestone 5: IAM Role Setup

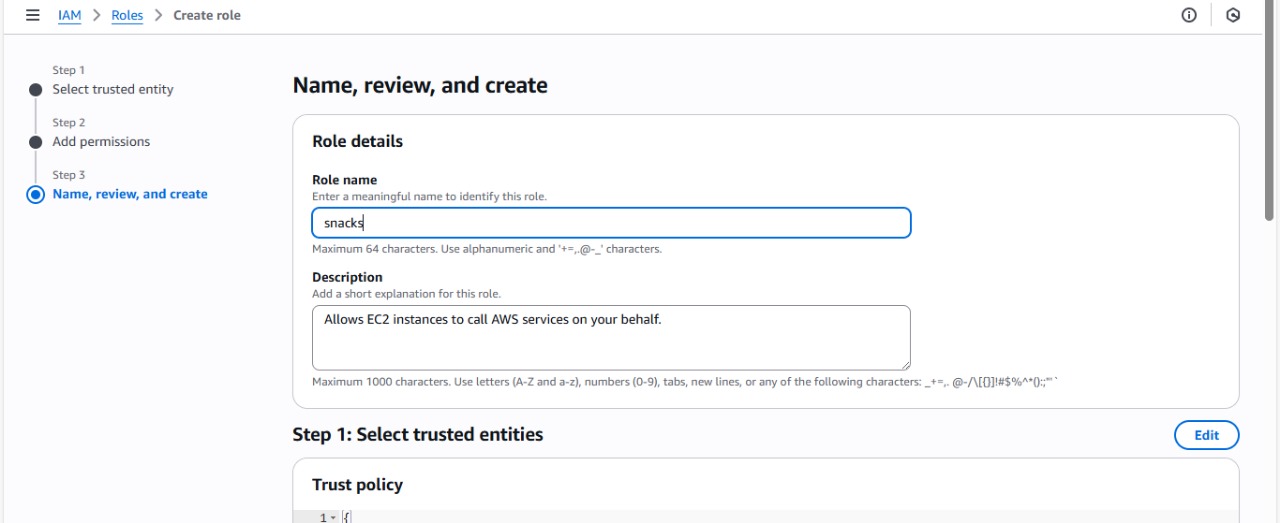
### Activity 5.1:Create IAM Role.

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







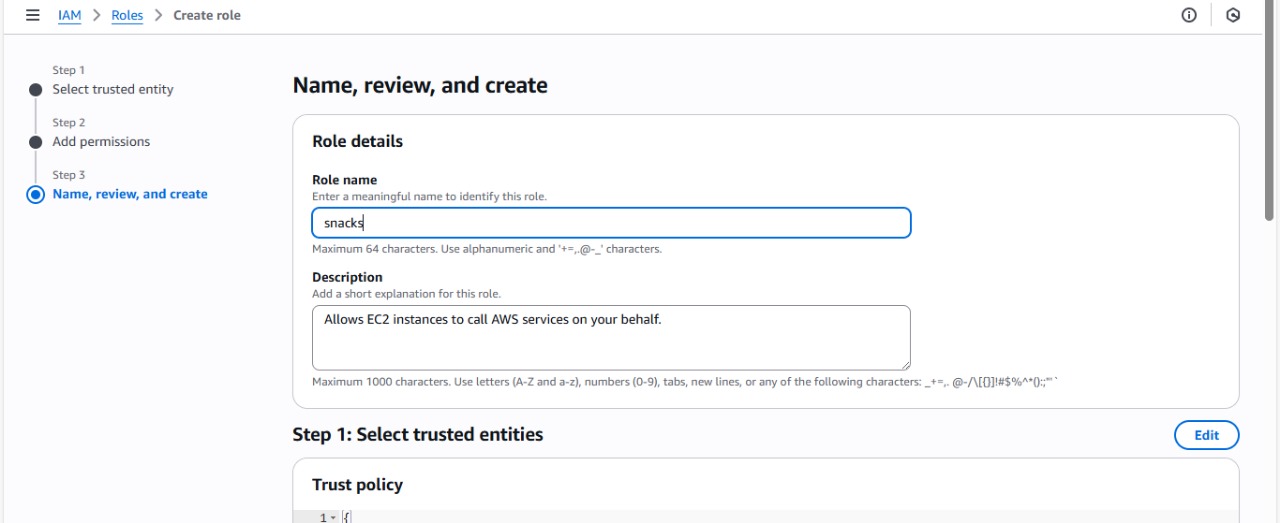


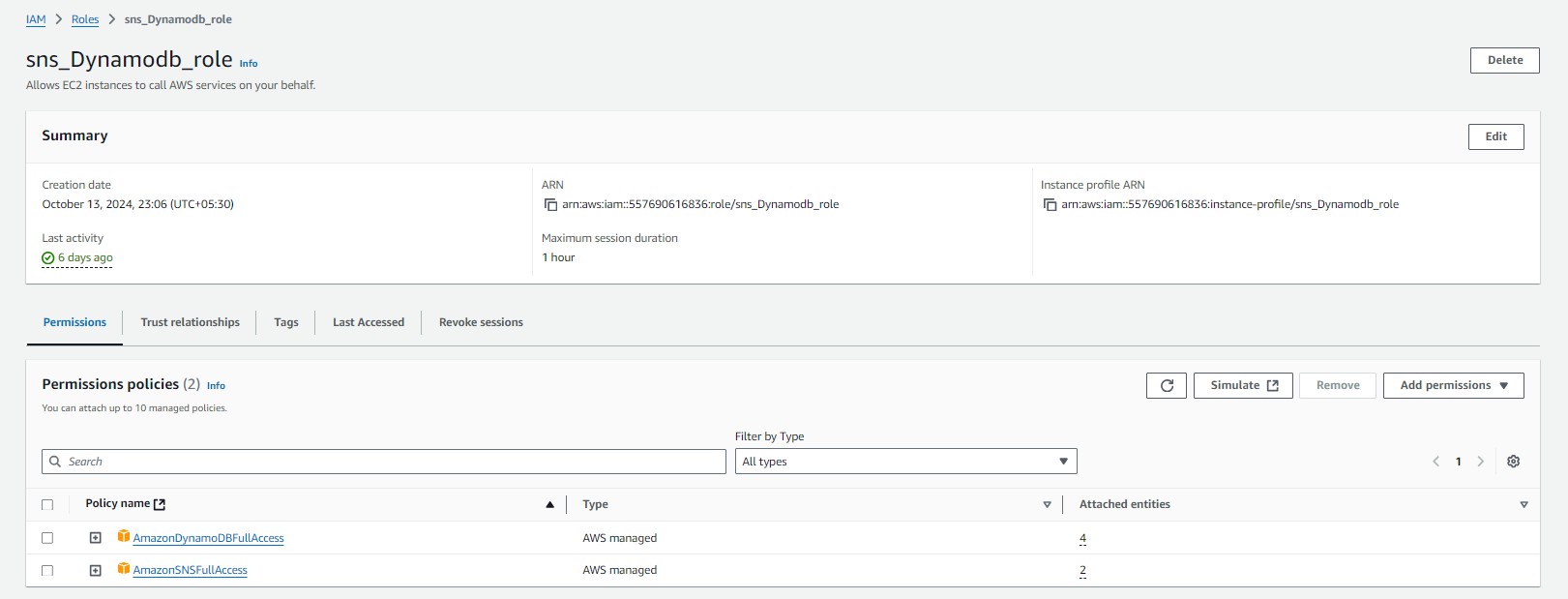
### Activity 5.2: Attach Policies.

Attach the following policies to the role:

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

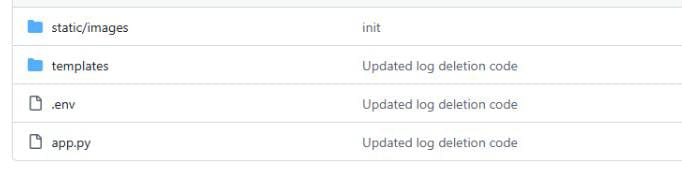


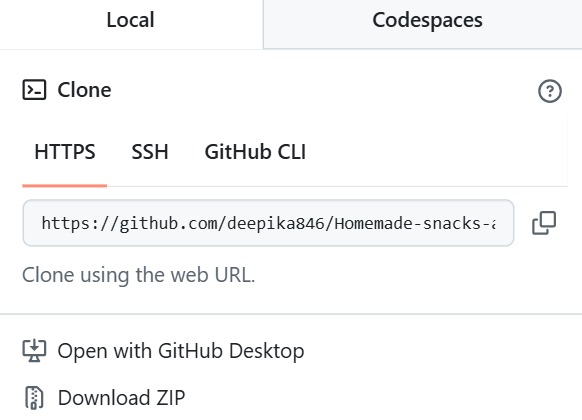




# Milestone 6: EC2 Instance Setup

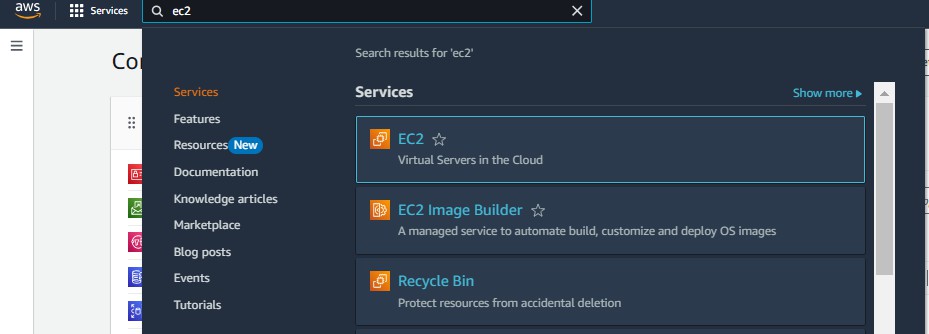
* **Note: Load your Flask app and Html files into GitHub repository.**

****

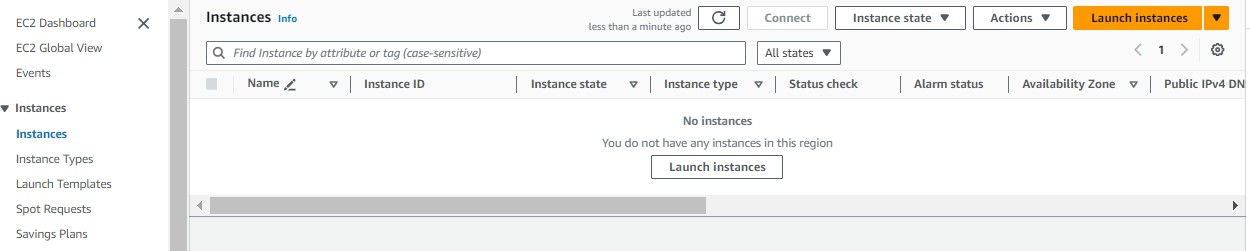


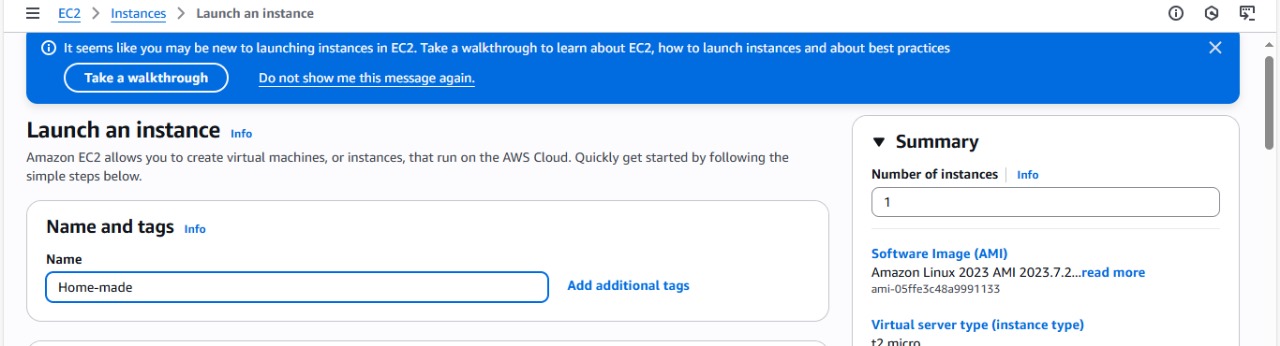
### Activity 6.1: Launch an EC2 instance to host the Flask application.

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

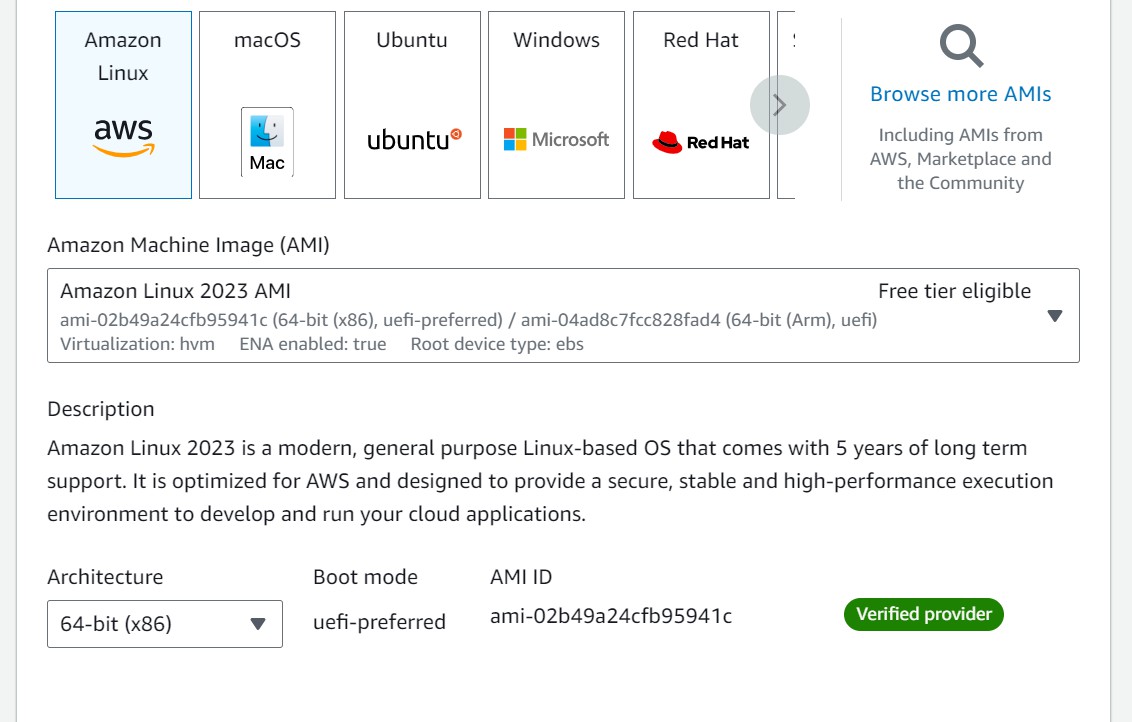


* Click on Launch instance to launch EC2 instance

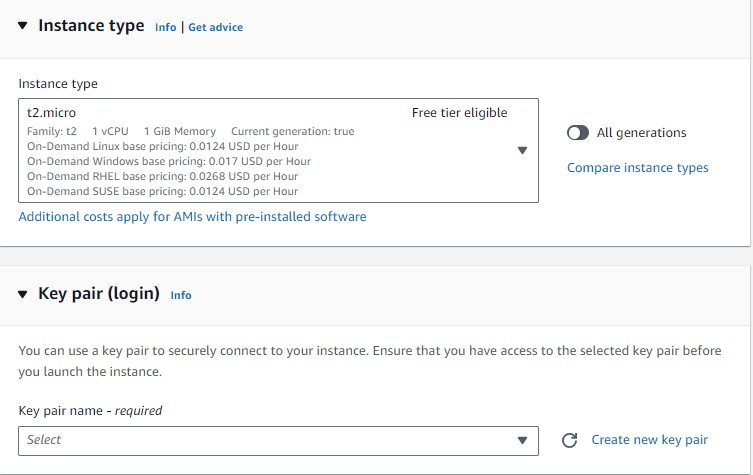


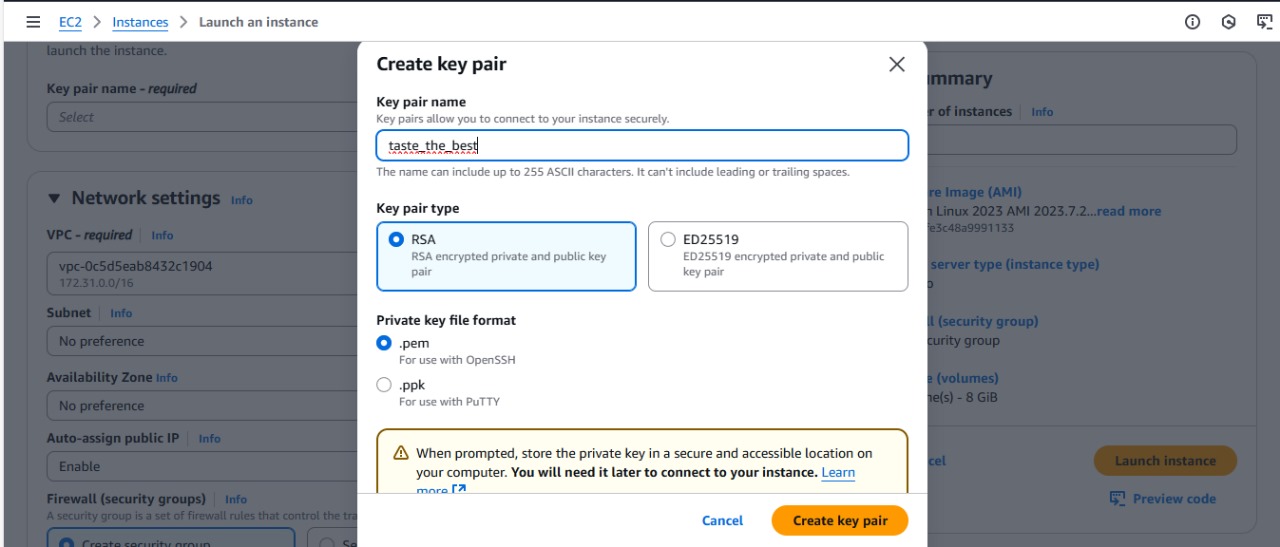


* + Choose Amazon Linux 2 or Ubuntu as the AMI and t2.micro as the instance type (free-tier eligible).

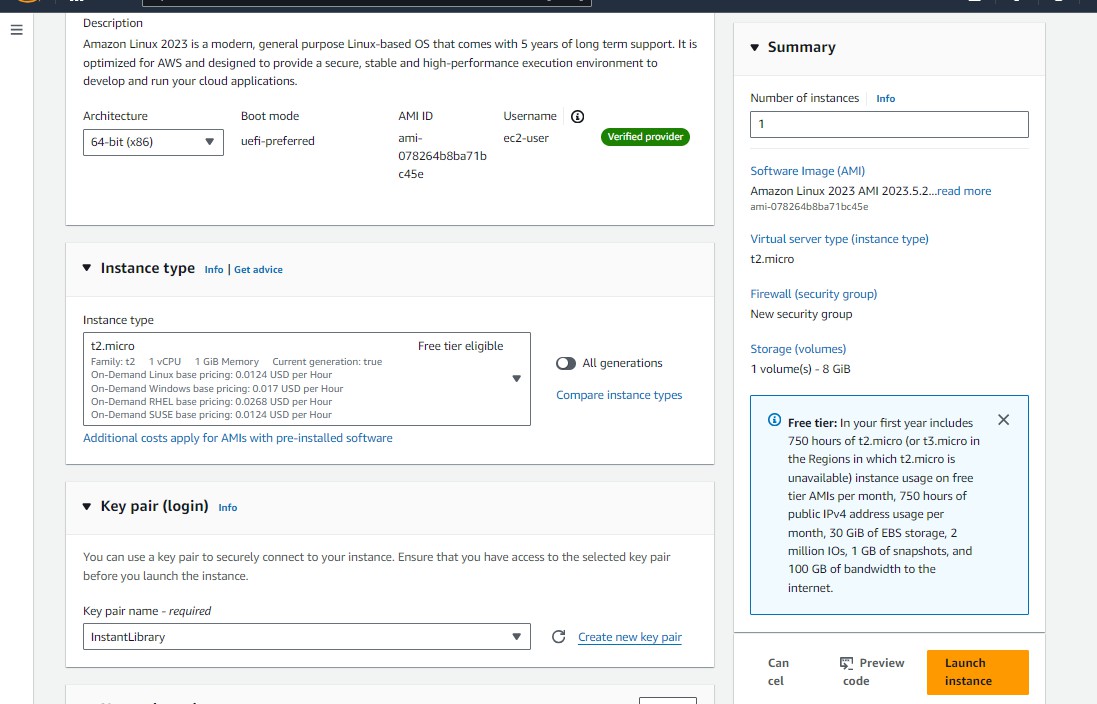


* Create and download the key pair for Server access.

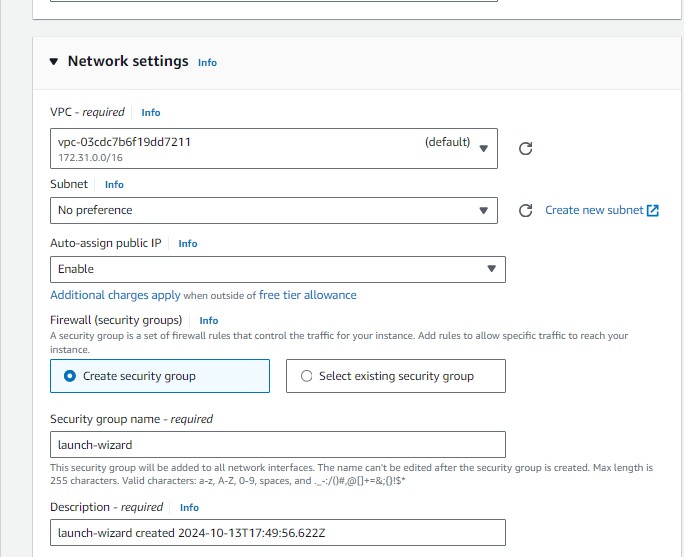


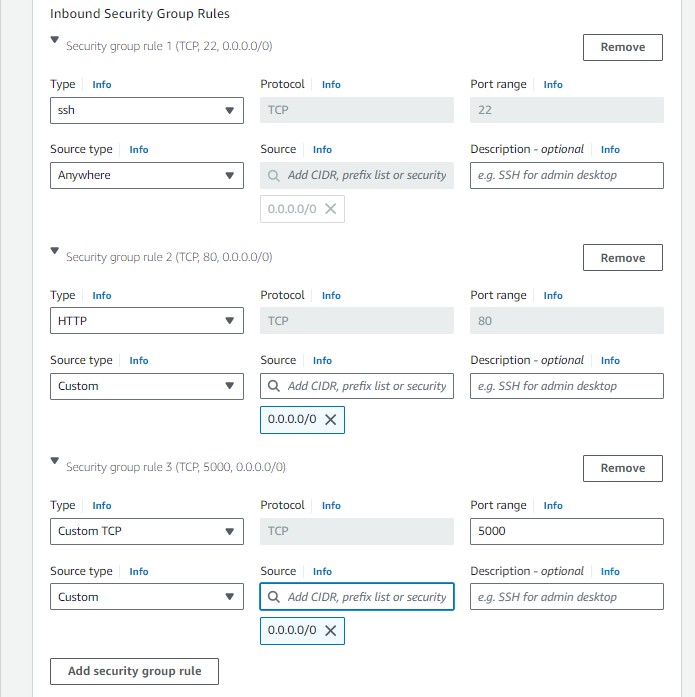


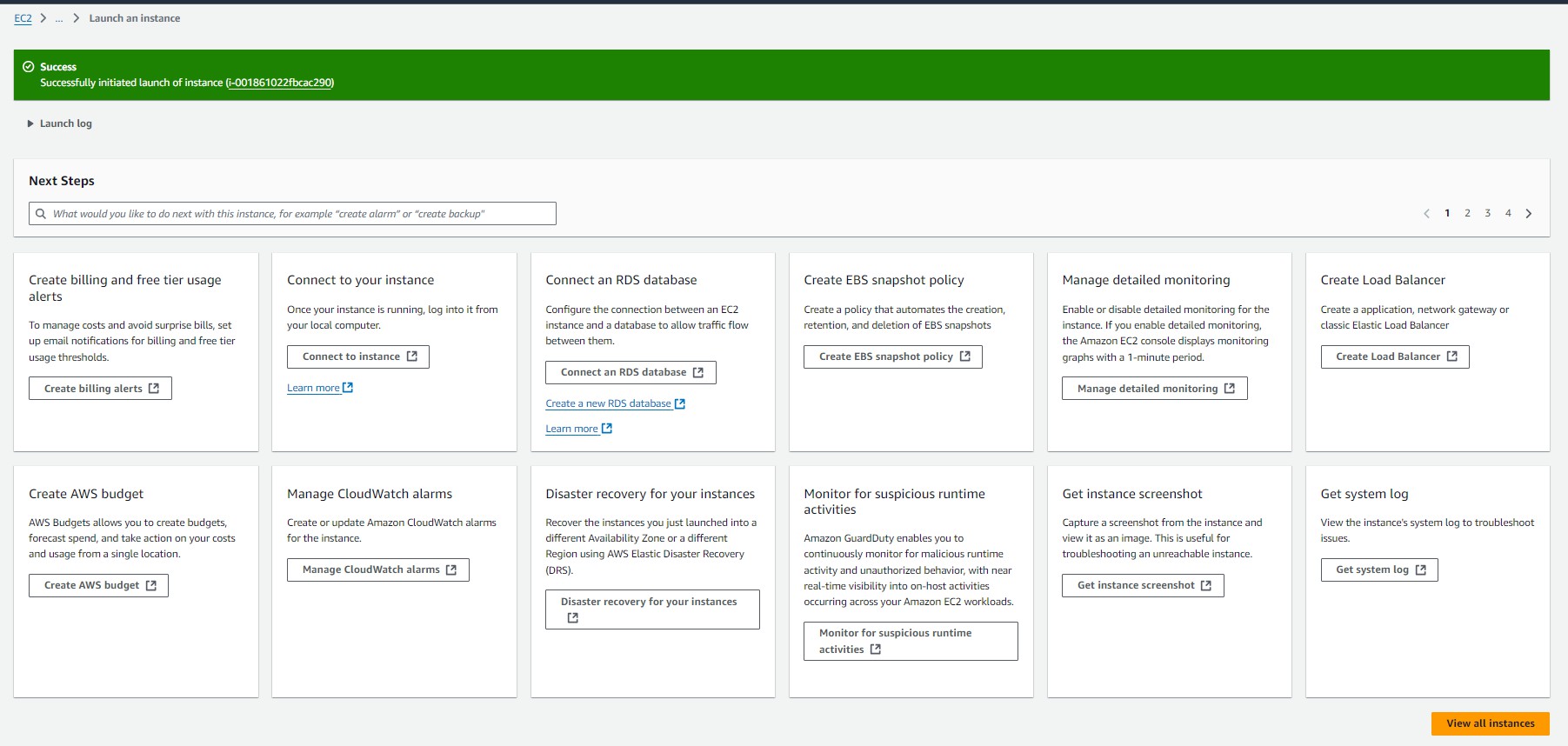




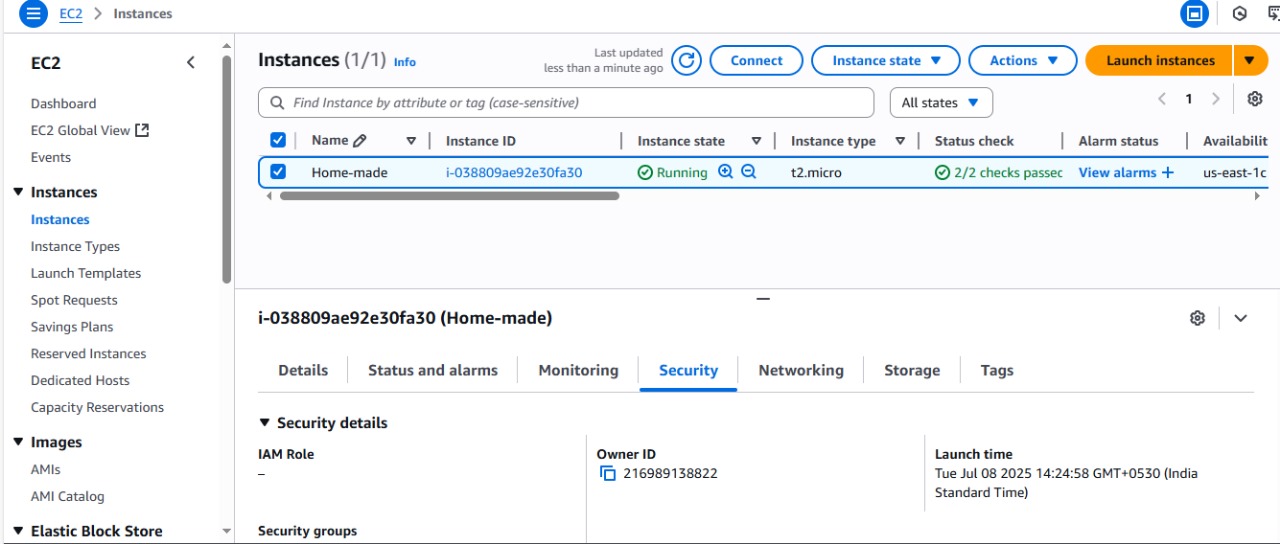
### Activity 6.2:Configure security groups for HTTP, and SSH access.

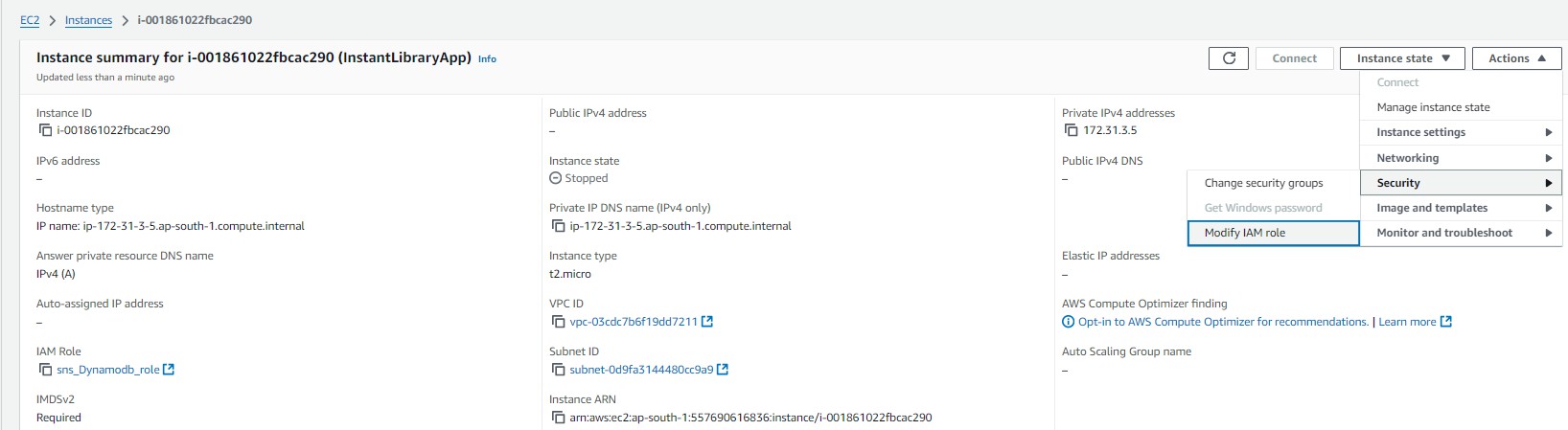
****

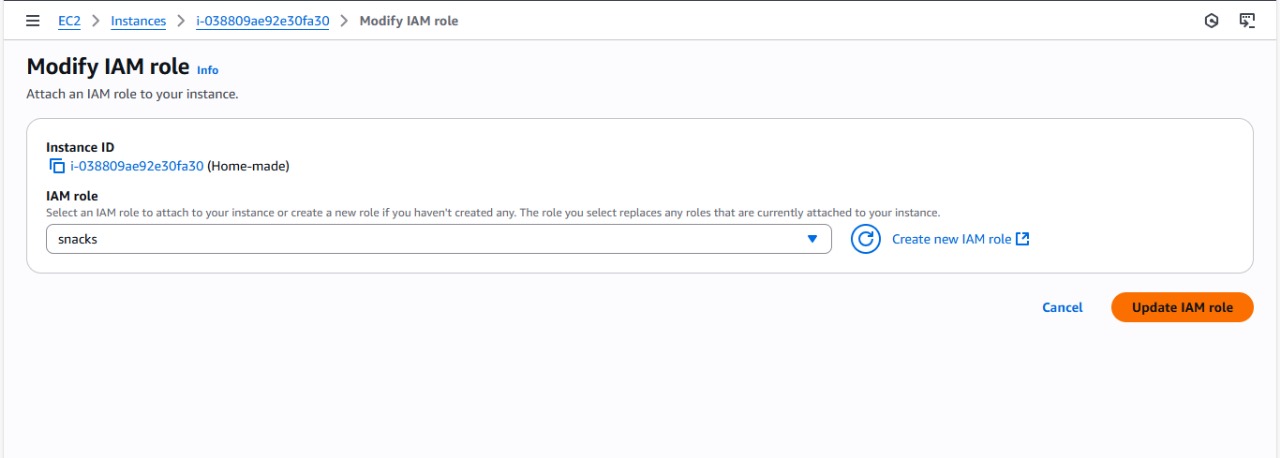
****



* 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.



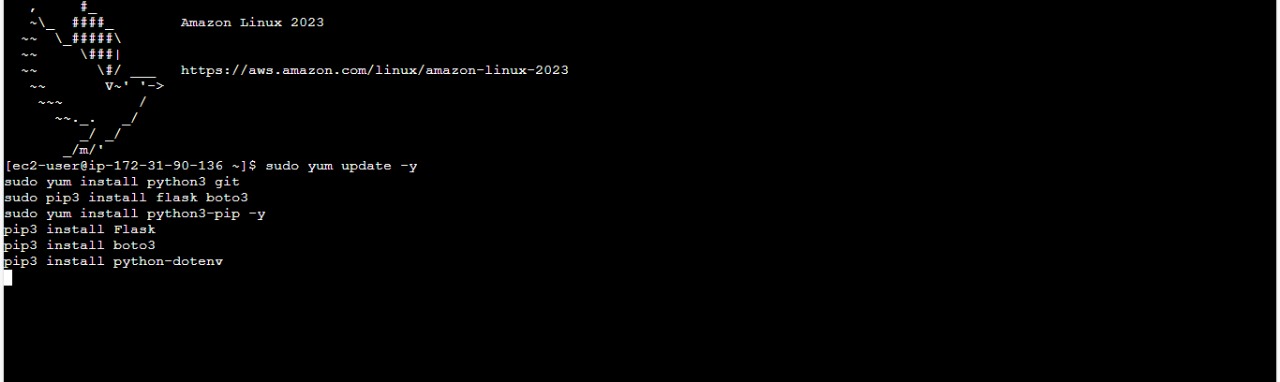




* Now connect the EC2 with the files







**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/your-github-username/your-repository-name.git>’

Note: change your-github-username and your-repository-name with your credentials here: ‘git clone https://github.com/Deepika-9755/Homemade-snacks-and-pickles.git

* This will download your project to the EC2 instance.

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

cd Homemade-snacks-and-pickles

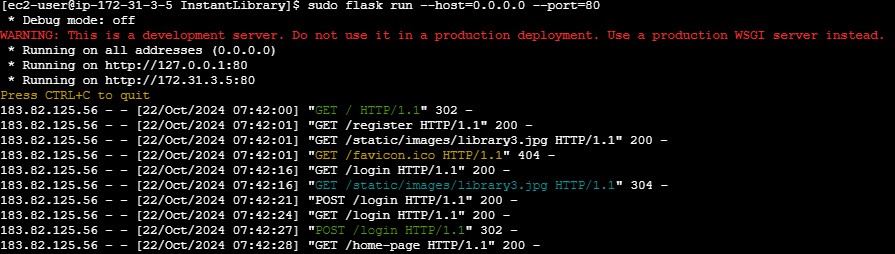
### Once inside the project directory, configure and run the Flask application by executing the following command with elevated privileges:

**Run the Flask Application**

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

**Verify the Flask app is running**: [http://your-ec2-public-ip](http://your-ec2-public-ip/)

* Run the Flask app on the EC2 instance



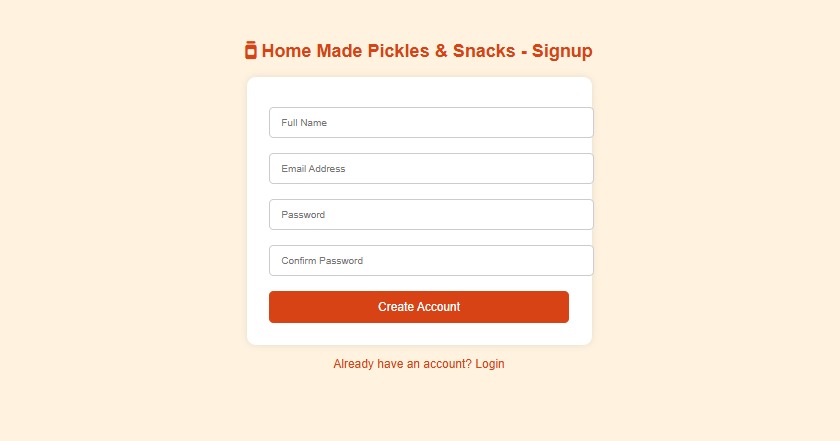
### Access the website through:

**PUBLIC IP** : 184.72.205.71/5000

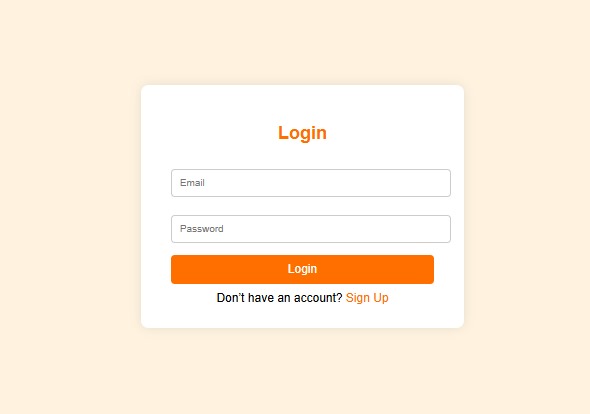
## Milestone 8: Testing and Deployment

### Activity 8.1: Conduct functional testing to verify user registration, login, order success ,order requests, and notifications.

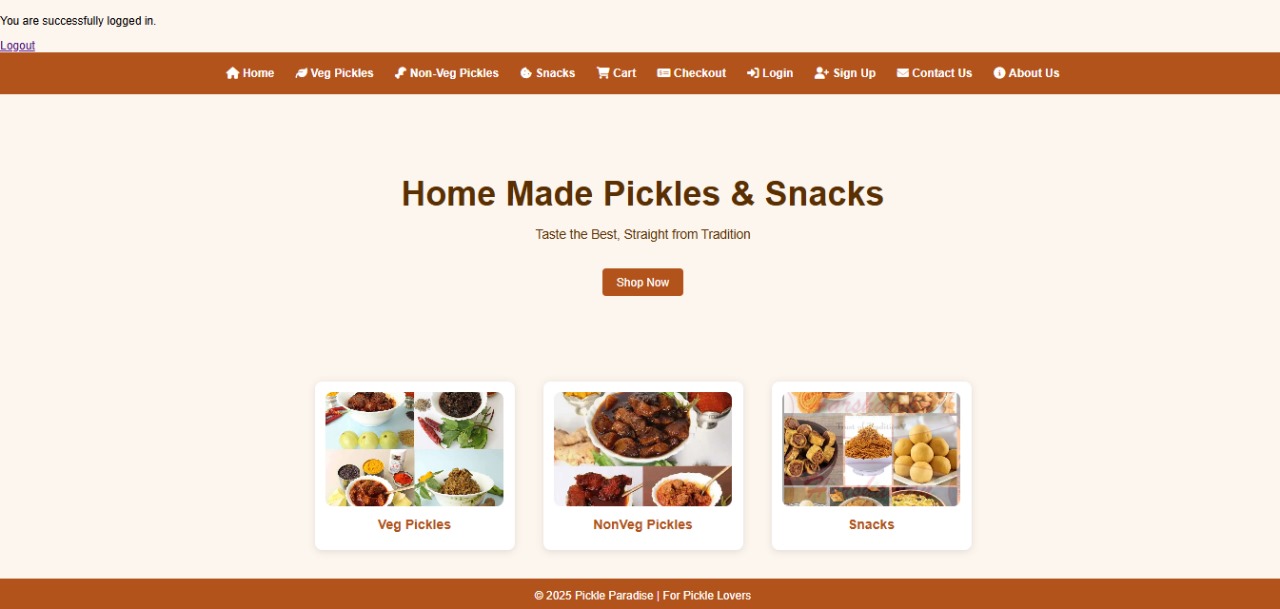
**Sign Up page**

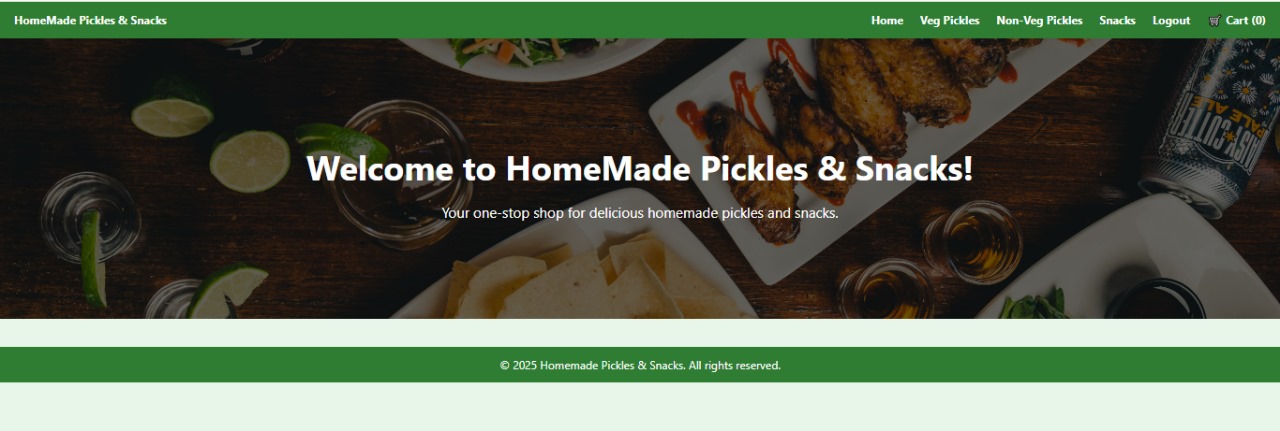
****

**Login Page:**

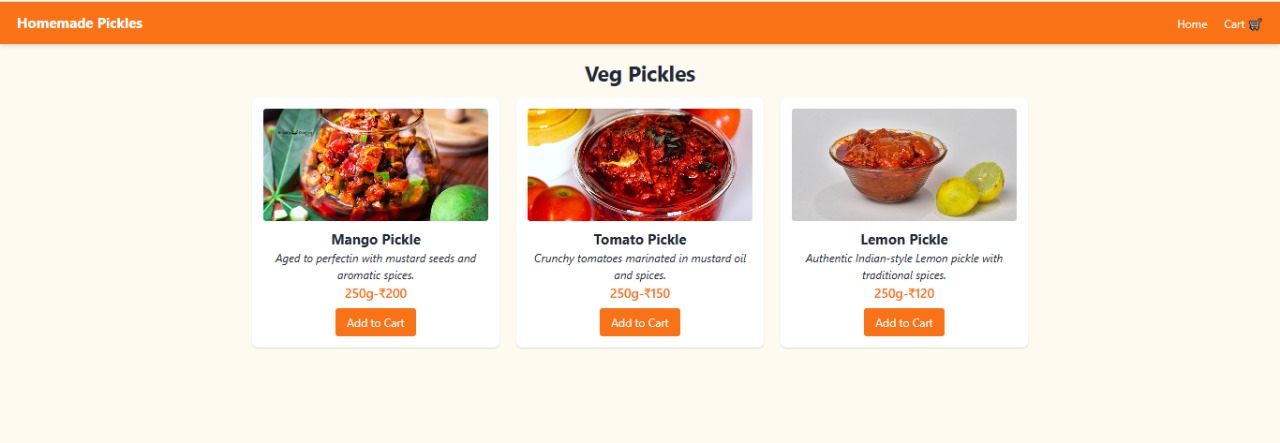
****

**Home Page**

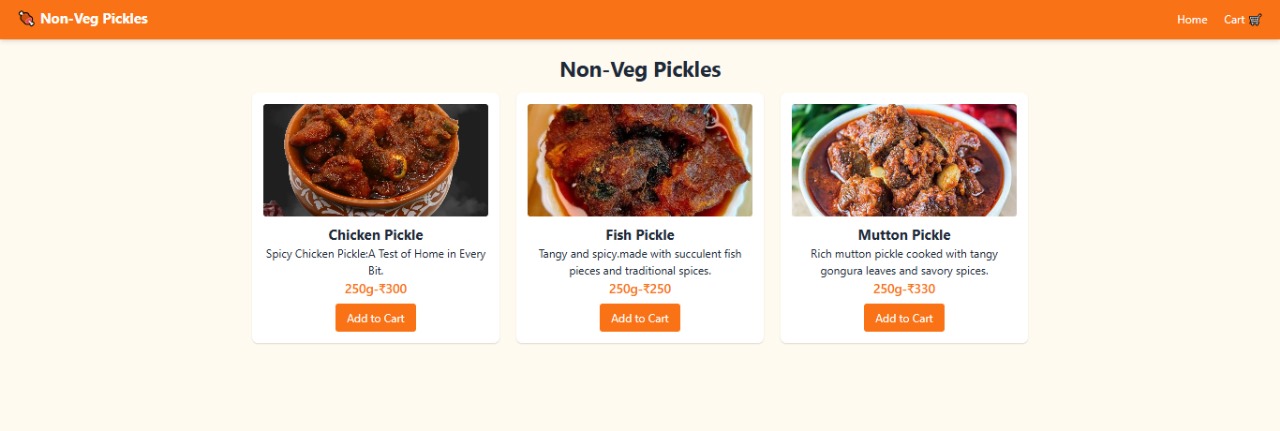
****

****

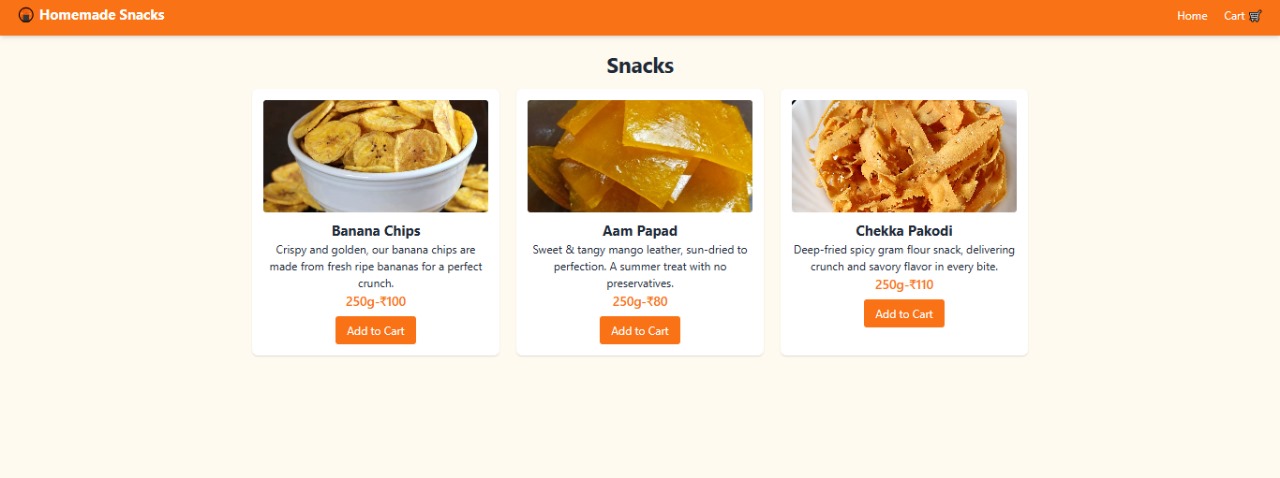
**Veg Pickles**

****

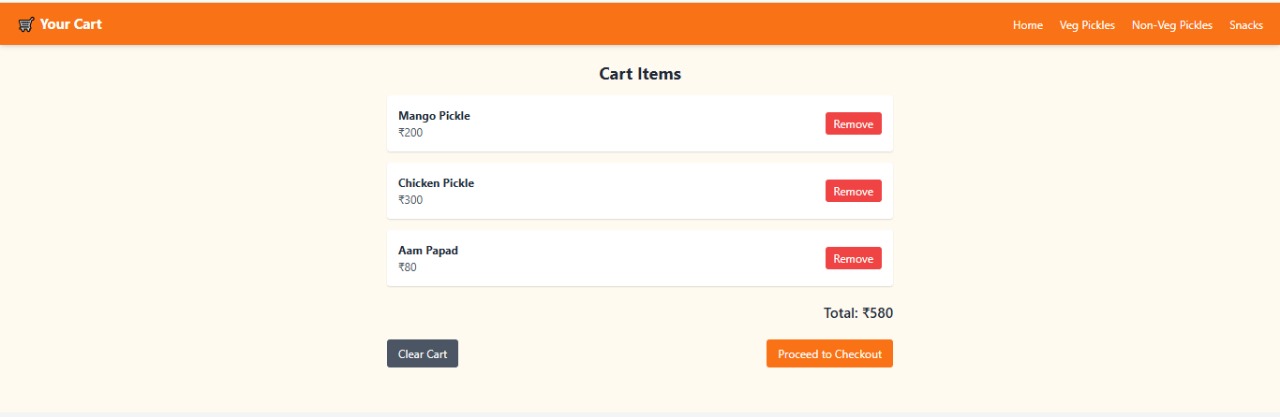
**Non Veg Pickles**

****

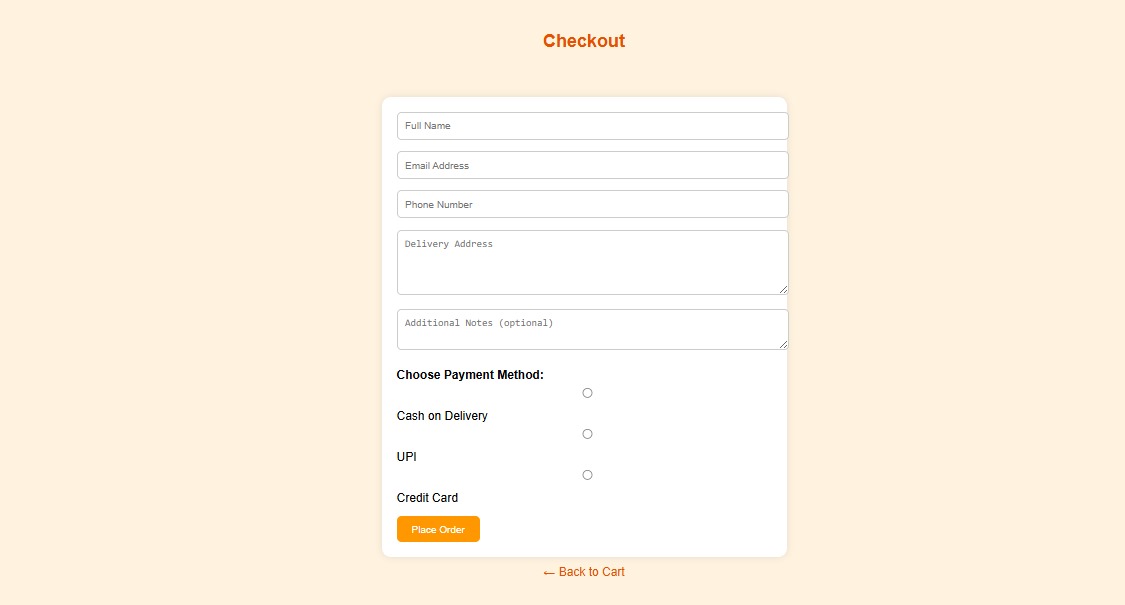
**Snacks Page**

****

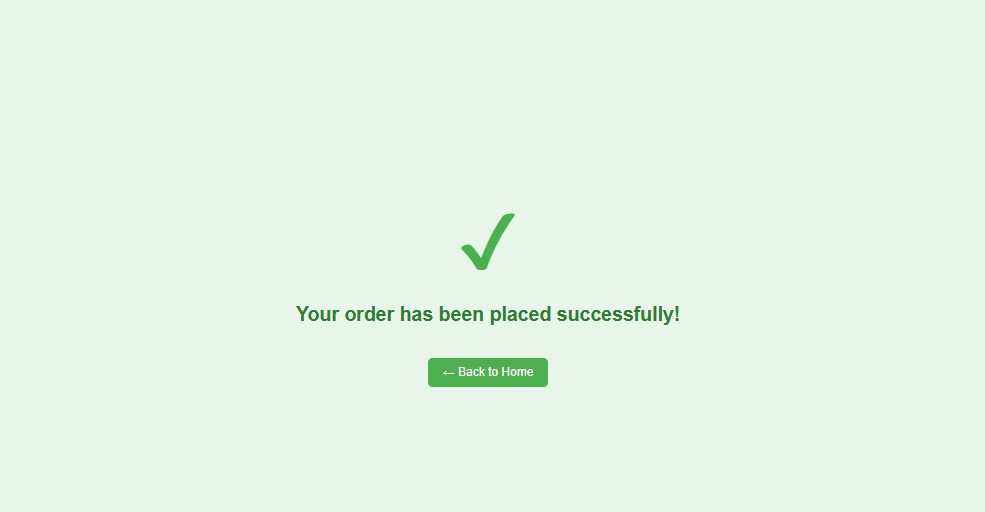
**Cart Page**

****

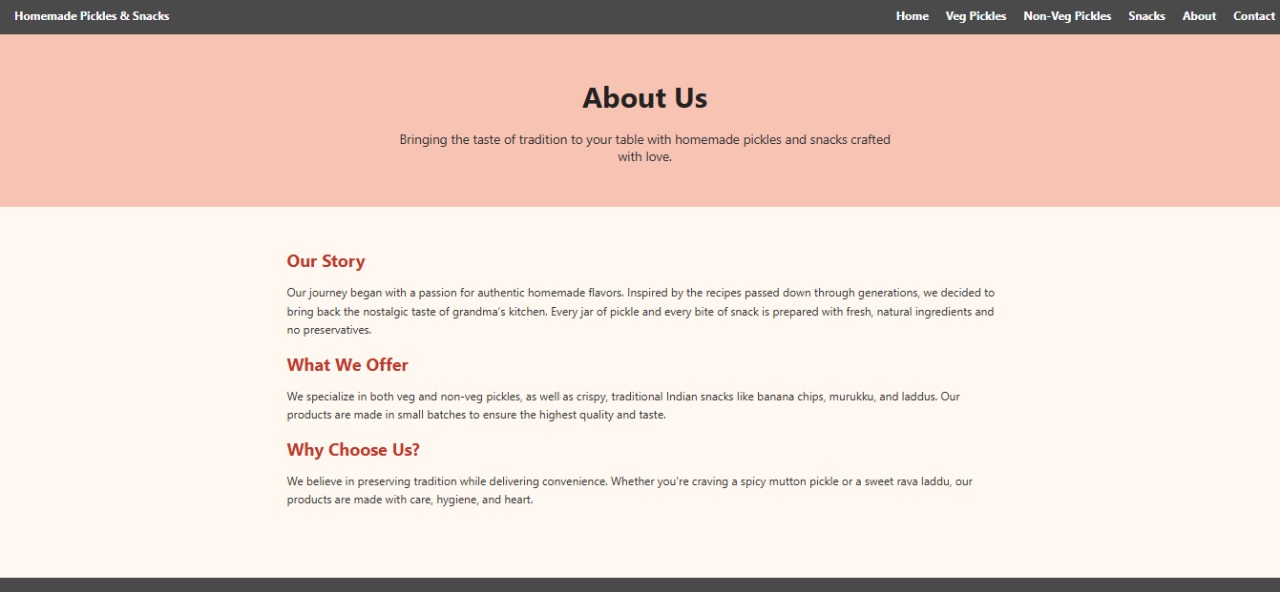
**Check out Page**

****

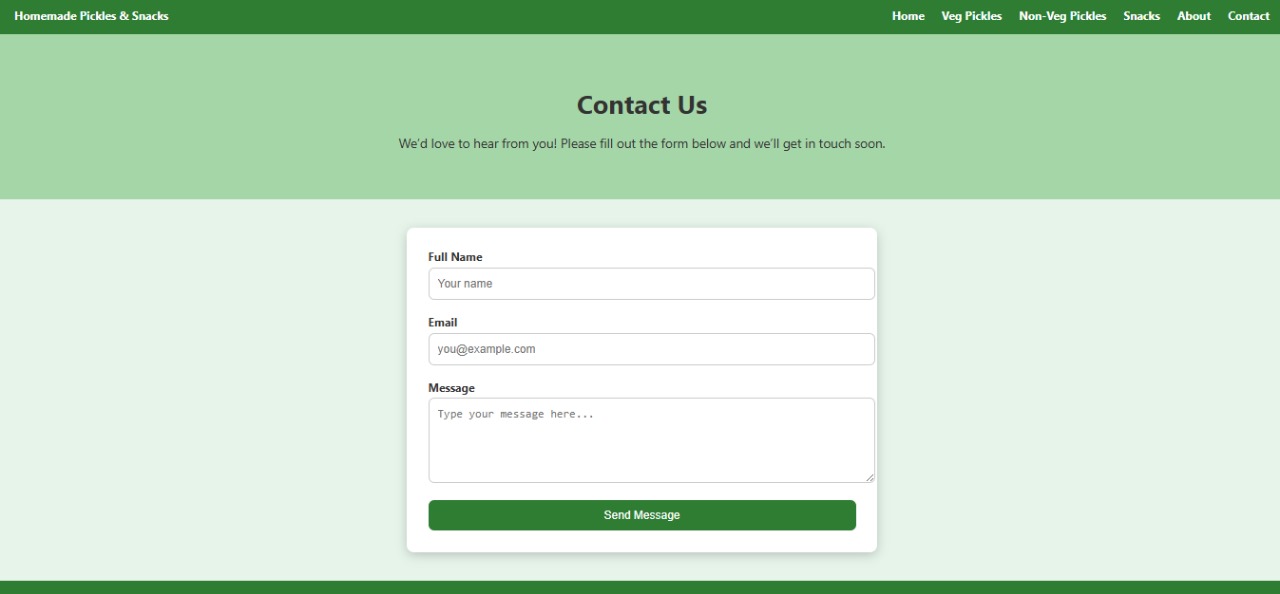
**Success page**

****

**About us page**

****

**Contact Us**

****

**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.