**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 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**

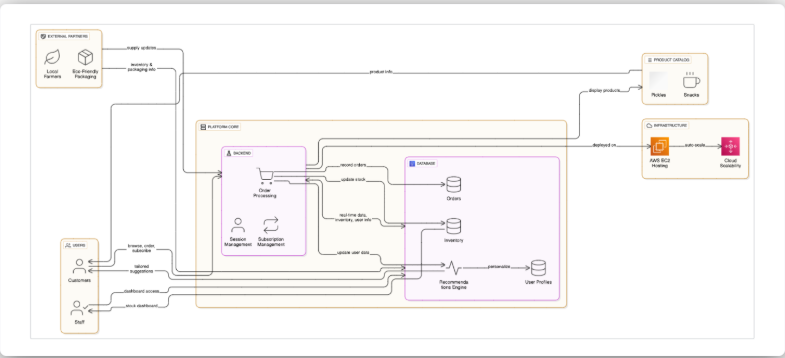
When a customer places an order for a product, the system instantly updates stock levels and records transaction details. For example, a user purchases an item, triggering automatic inventory deduction and order confirmation. Staff members receive updated dashboards to monitor stock availability and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

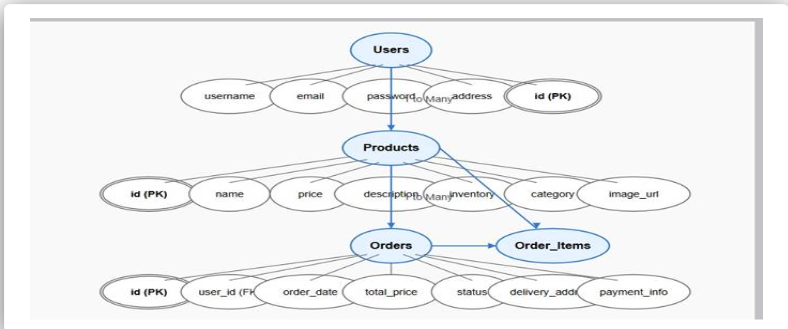
**Scenario 3: Personalized User Experience and Recommendations**

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

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](https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/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:  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.

# Milestone 2 : AWS Account Setup

# Important Notice: Use Troven Labs for AWS Access

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

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

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

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

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

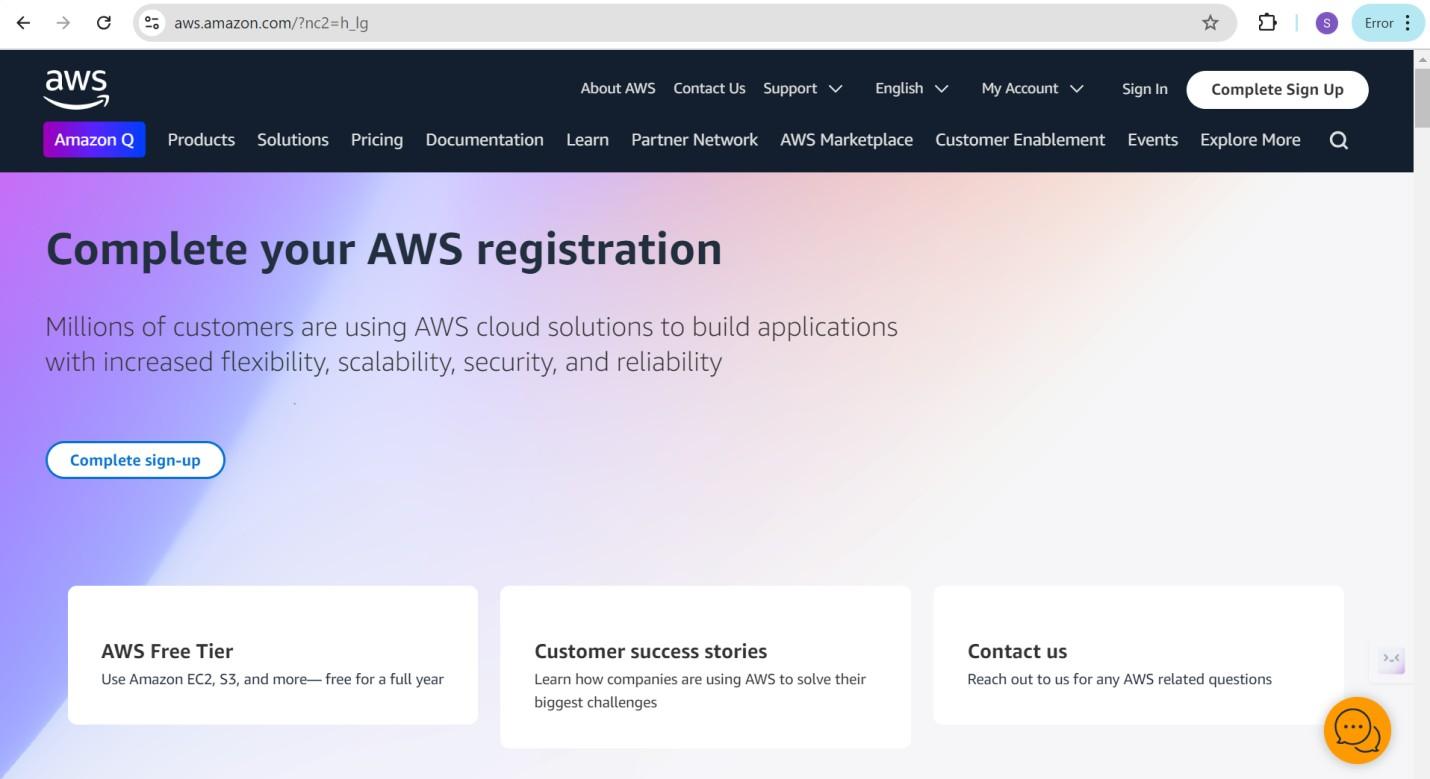
# Please refer the below link -

# <https://drive.google.com/file/d/1HzWc7AMJ2BrxhV-uaw5s0vWtcd-28qgI/view?usp=sharing>

**AWS Account Setup and Login**

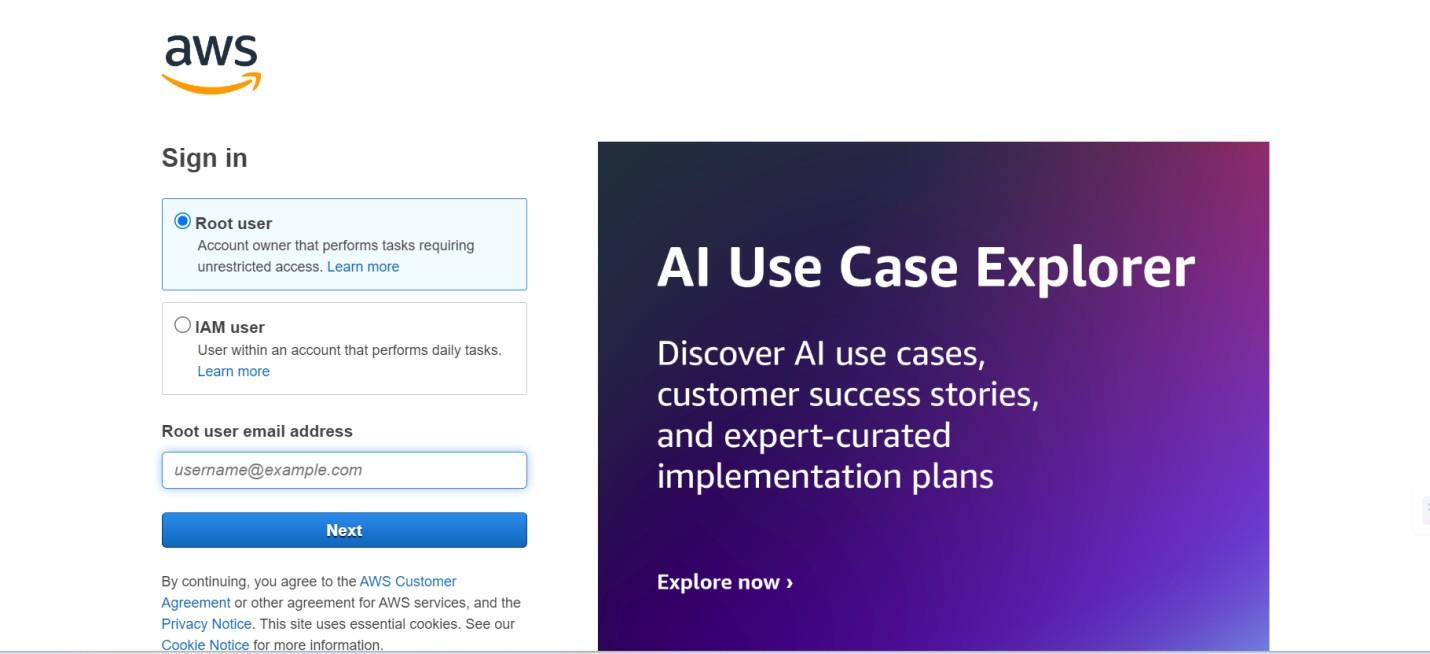
**This is for your understanding only, please refrain from creating an AWS account. A temporary account will be provided via Troven.**

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

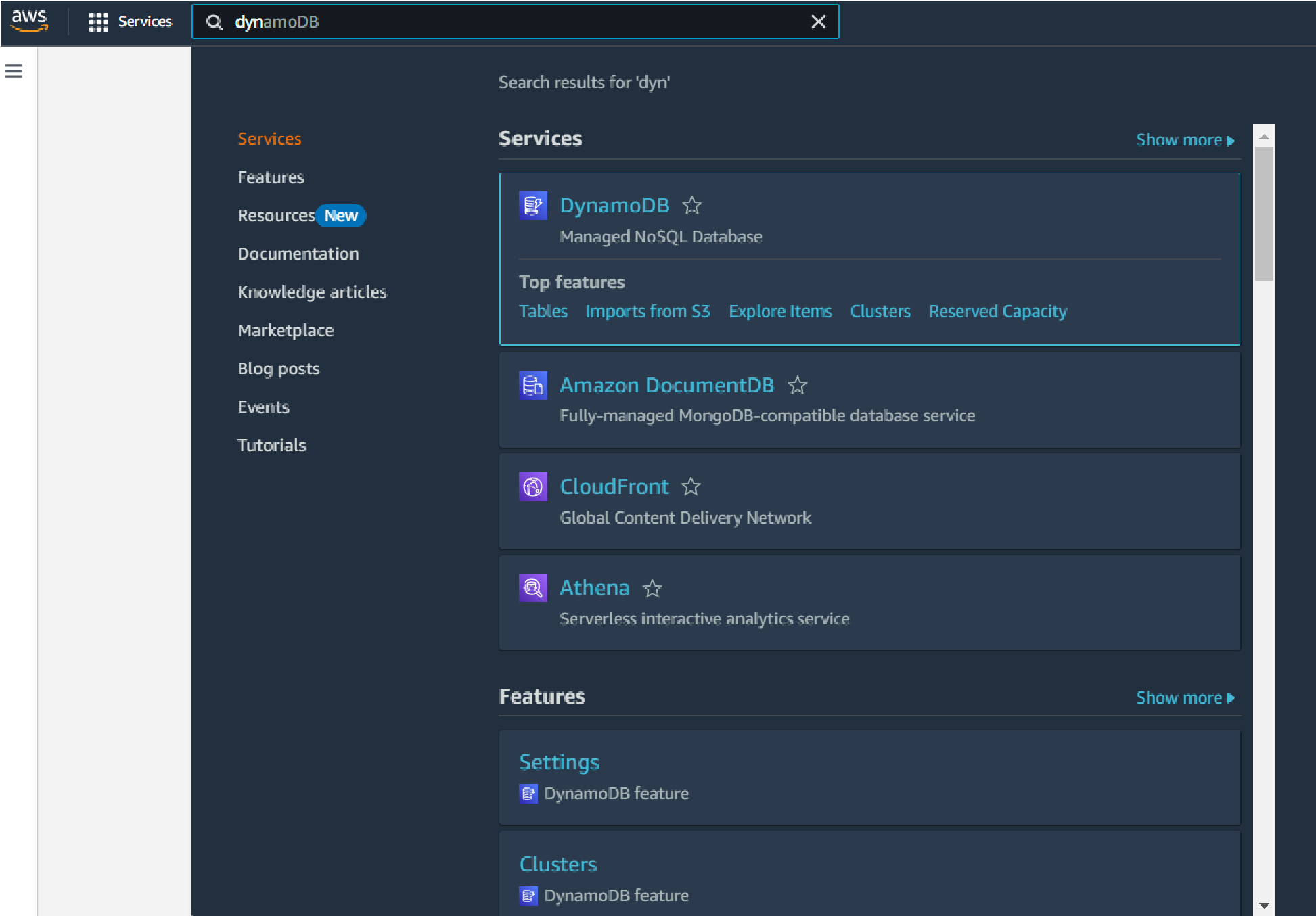


* **Activity 2.2: Log in to the AWS Management Console**

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

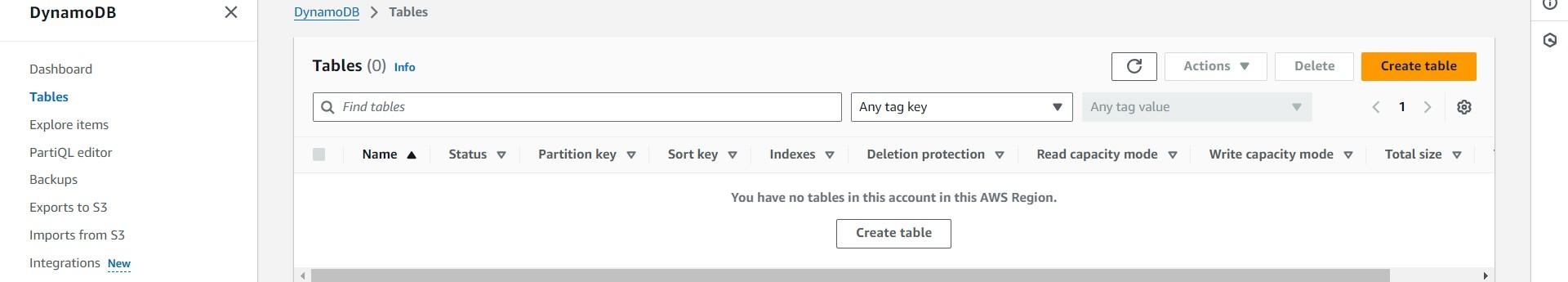


# Milestone 3: DynamoDB Database Creation and Setup

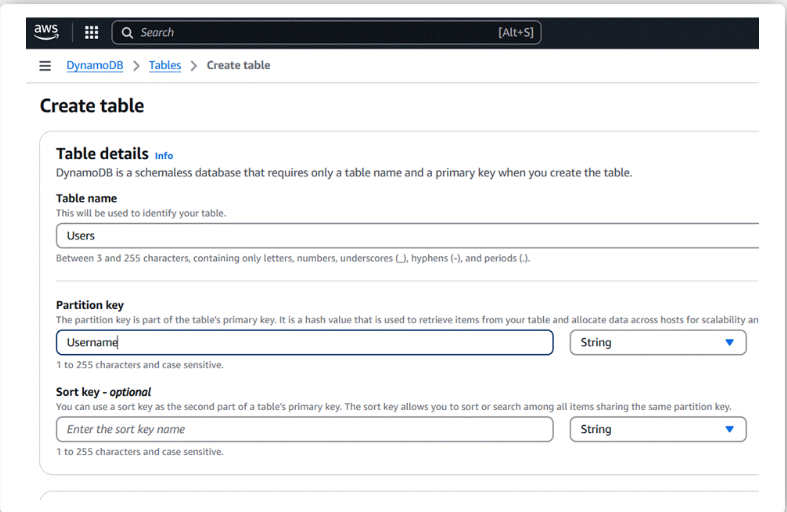
* **Activity 3.1:Navigate to the DynamoDB** 
  1. In the AWS Console, navigate to DynamoDB and click on create tables.

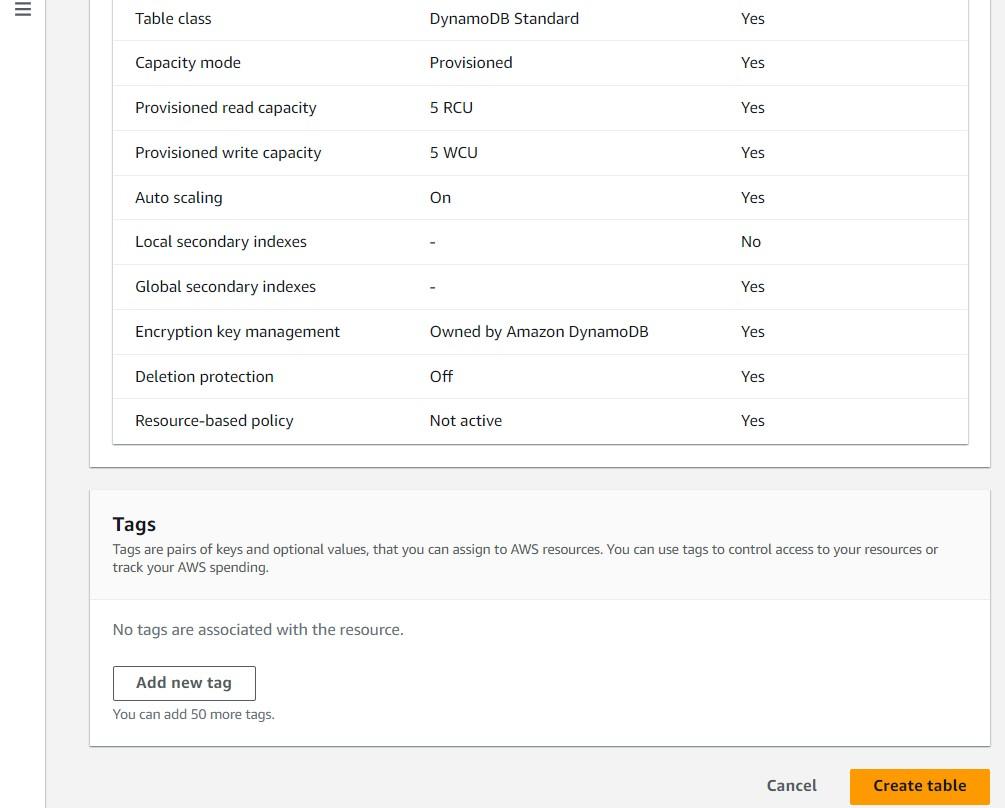
○

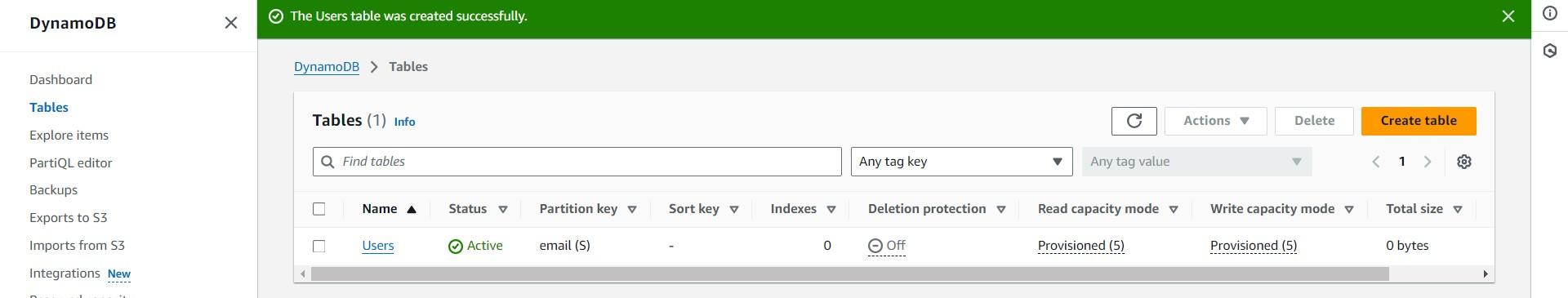




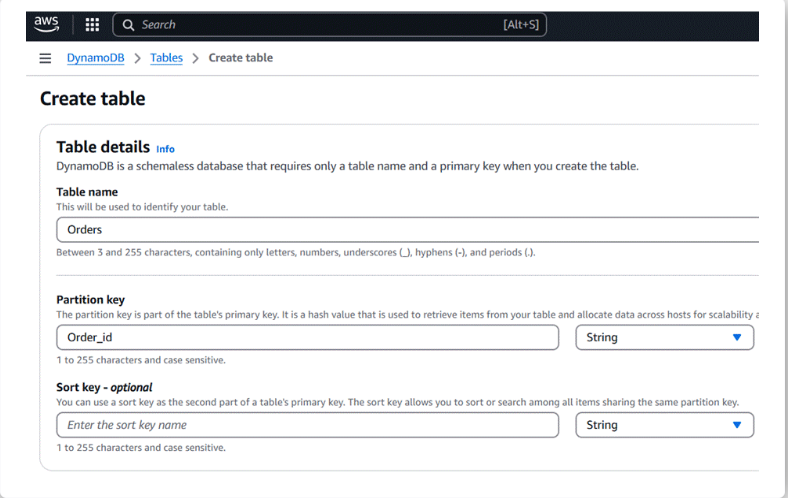
* **Activity 3.2:Create a DynamoDB table for storing registration details and book requests.** 
  1. Create Users table with partition key “Username” with type String and click on create tables.

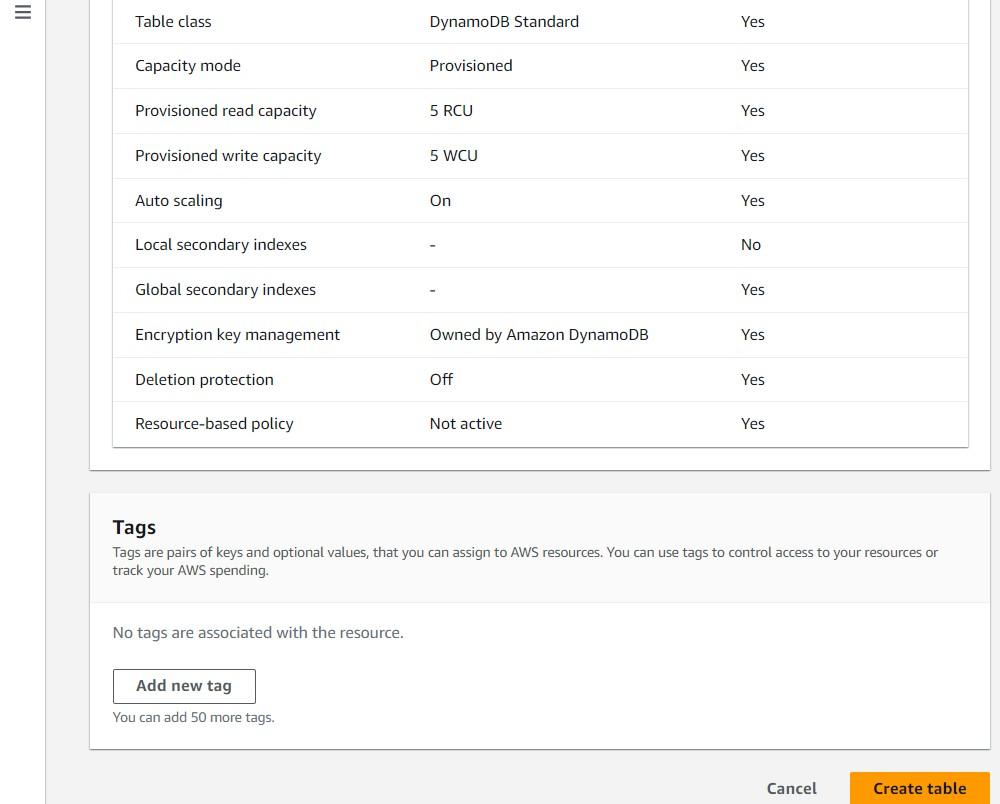


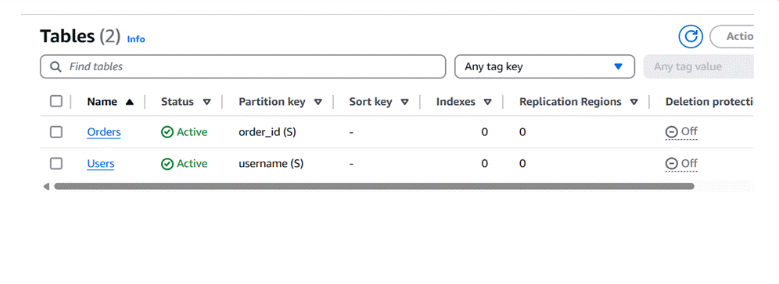




* ○  Follow the same steps to create an Orders table with Order\_id as the primary key to store Order details.



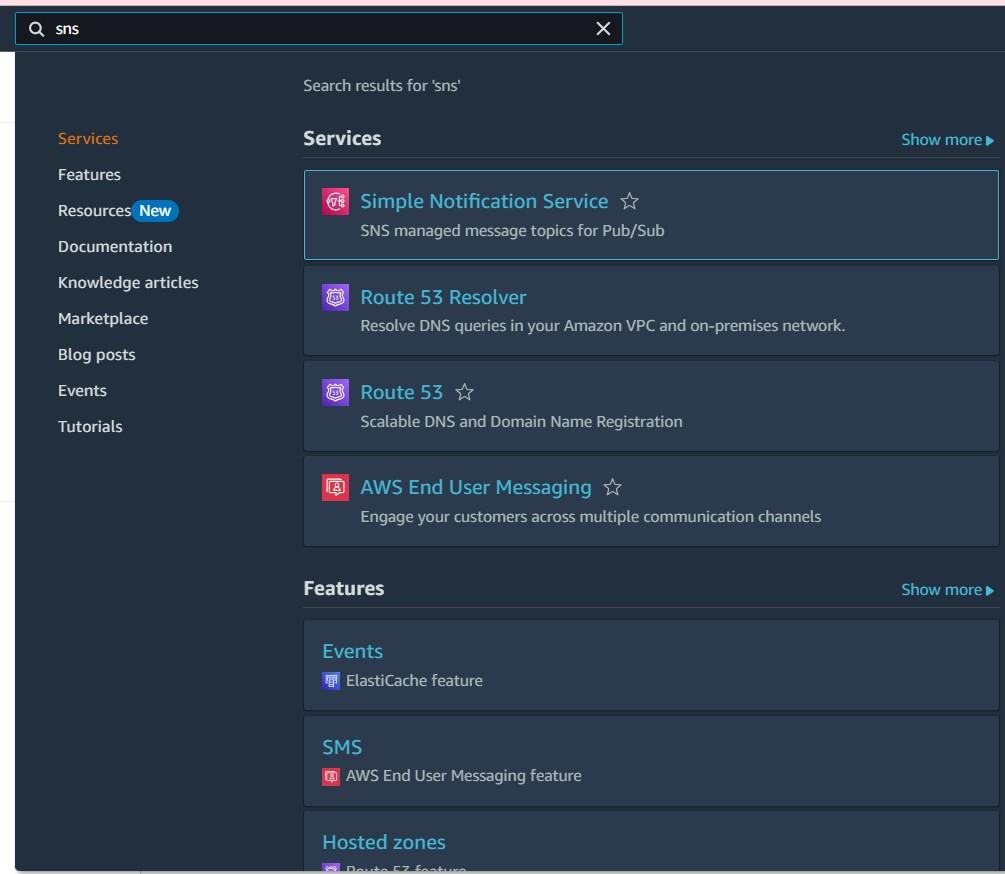


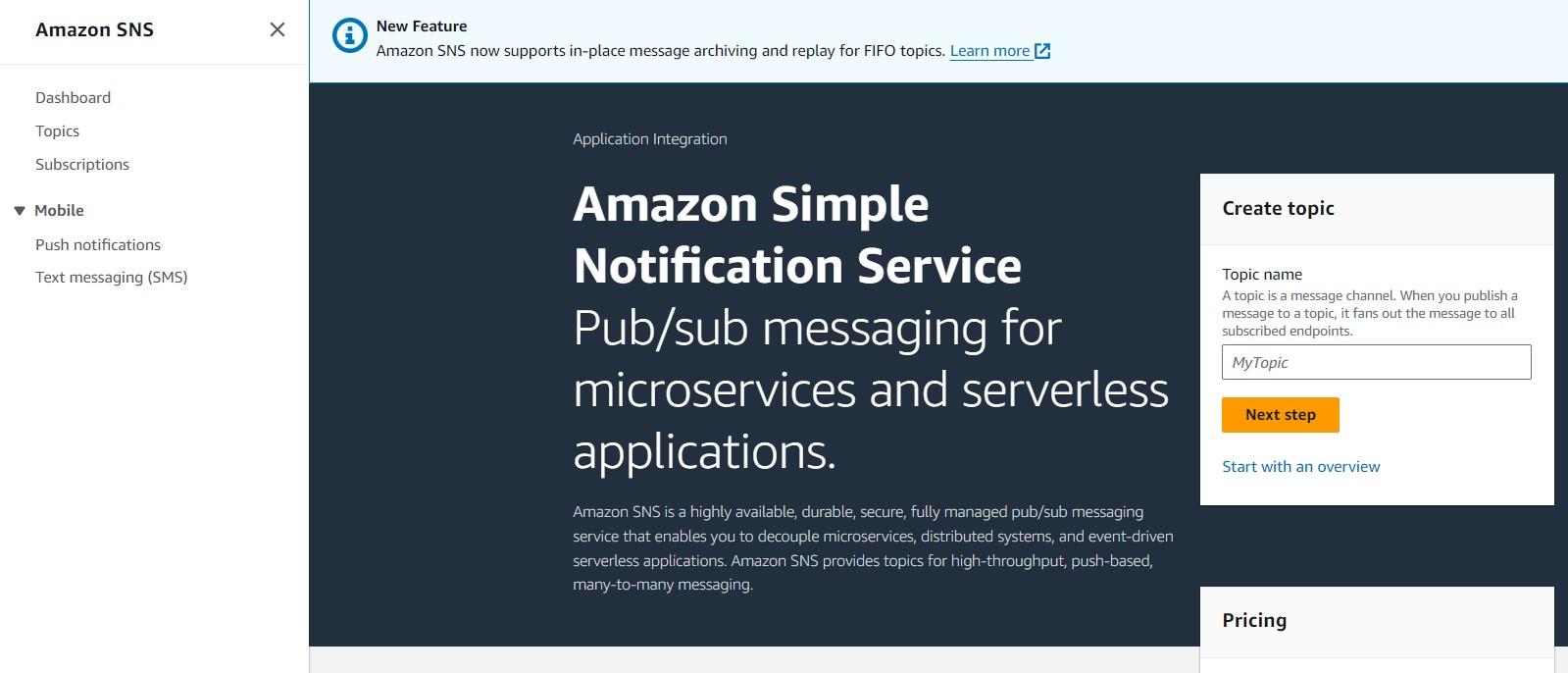


**Milestone 4: SNS Notification Setup**

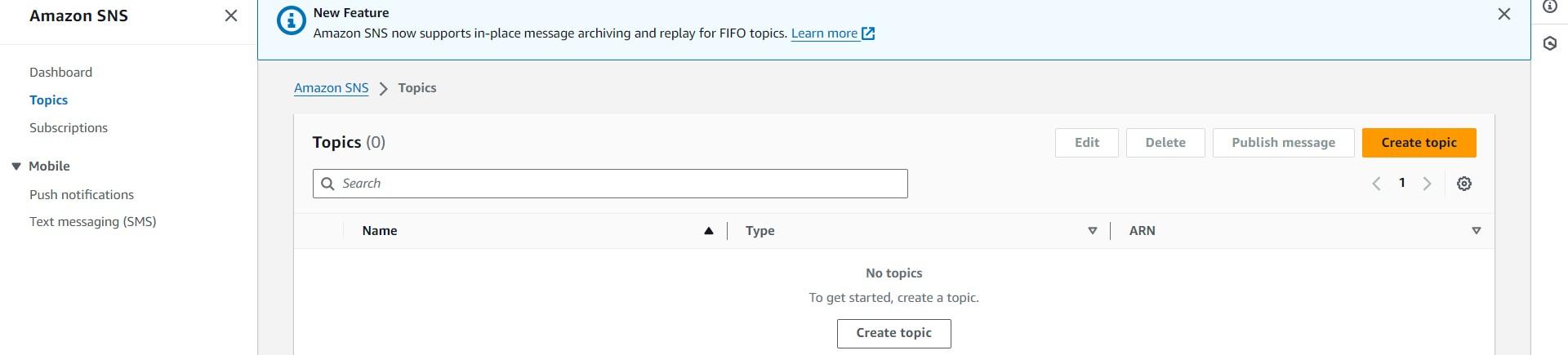
* **Activity 4.1: Create SNS topics for sending email notifications to users and library staff.**

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

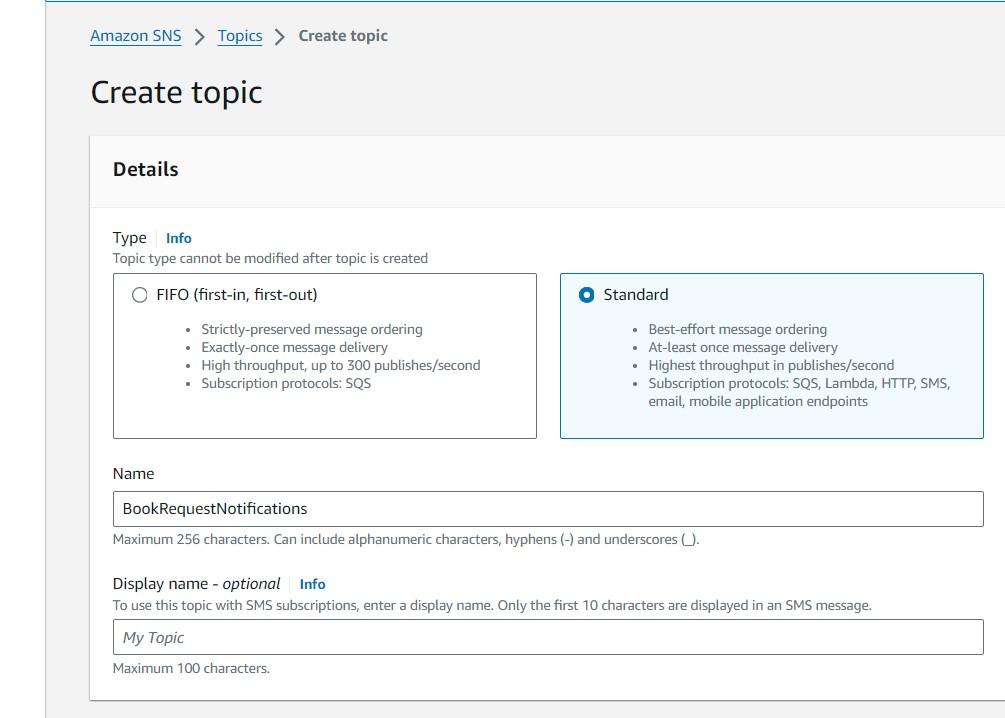


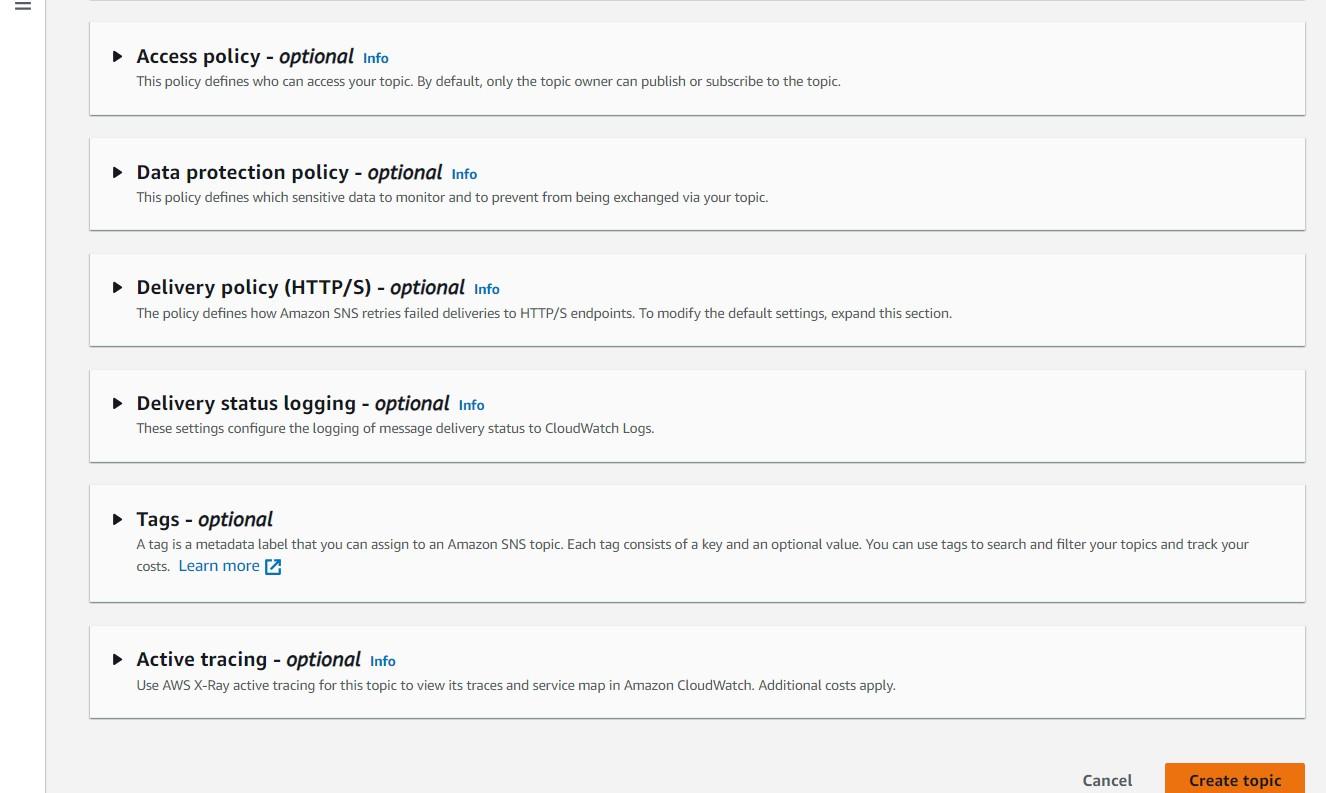


○ Click on **Create Topic** and choose a name for the topic.

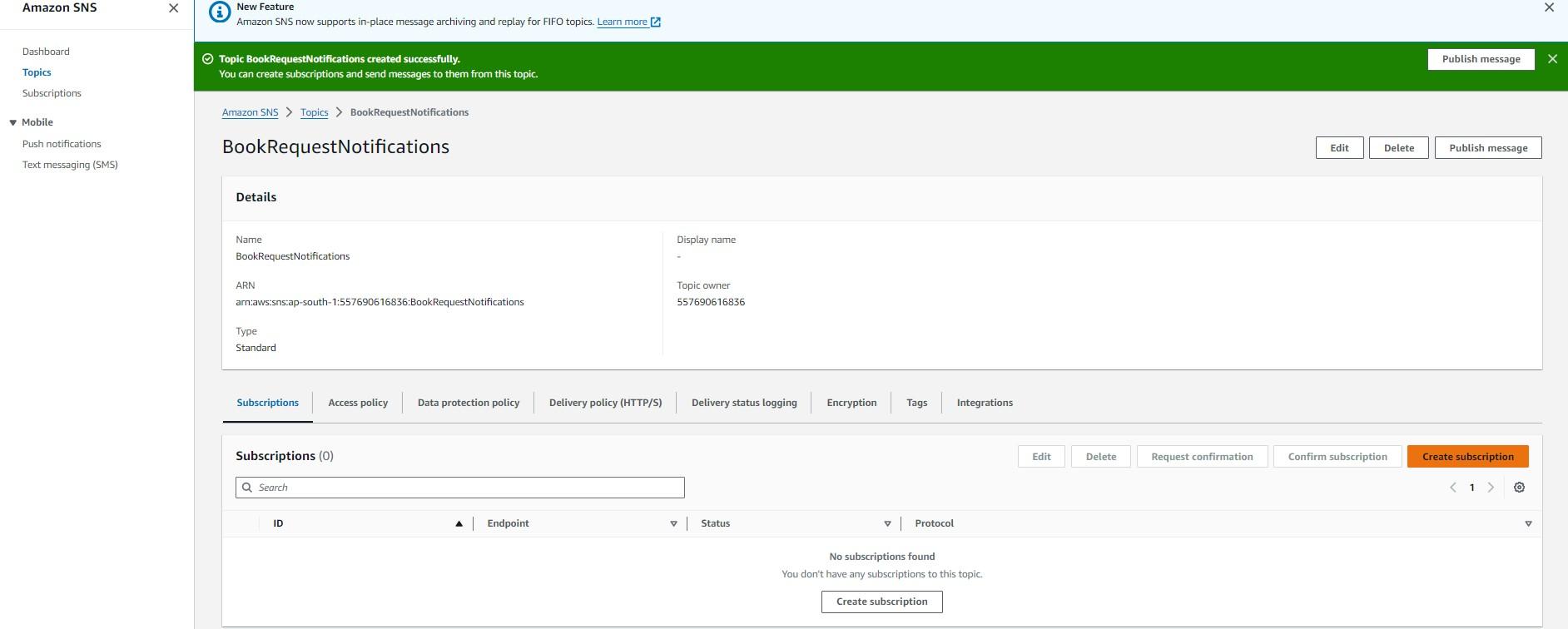


○ Choose Standard type for general notification use cases and Click on Create Topic.



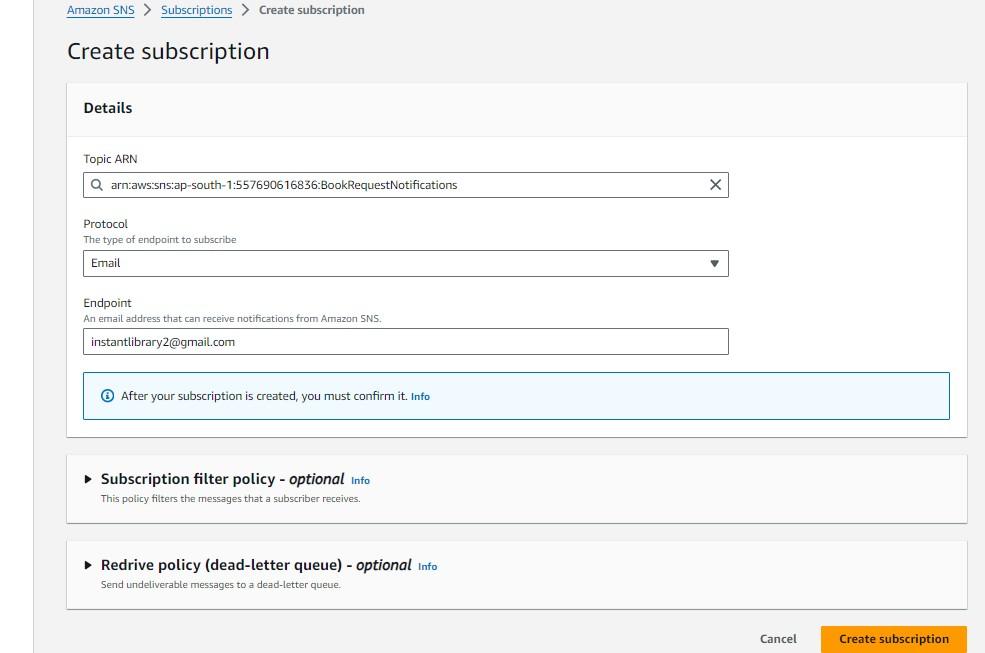


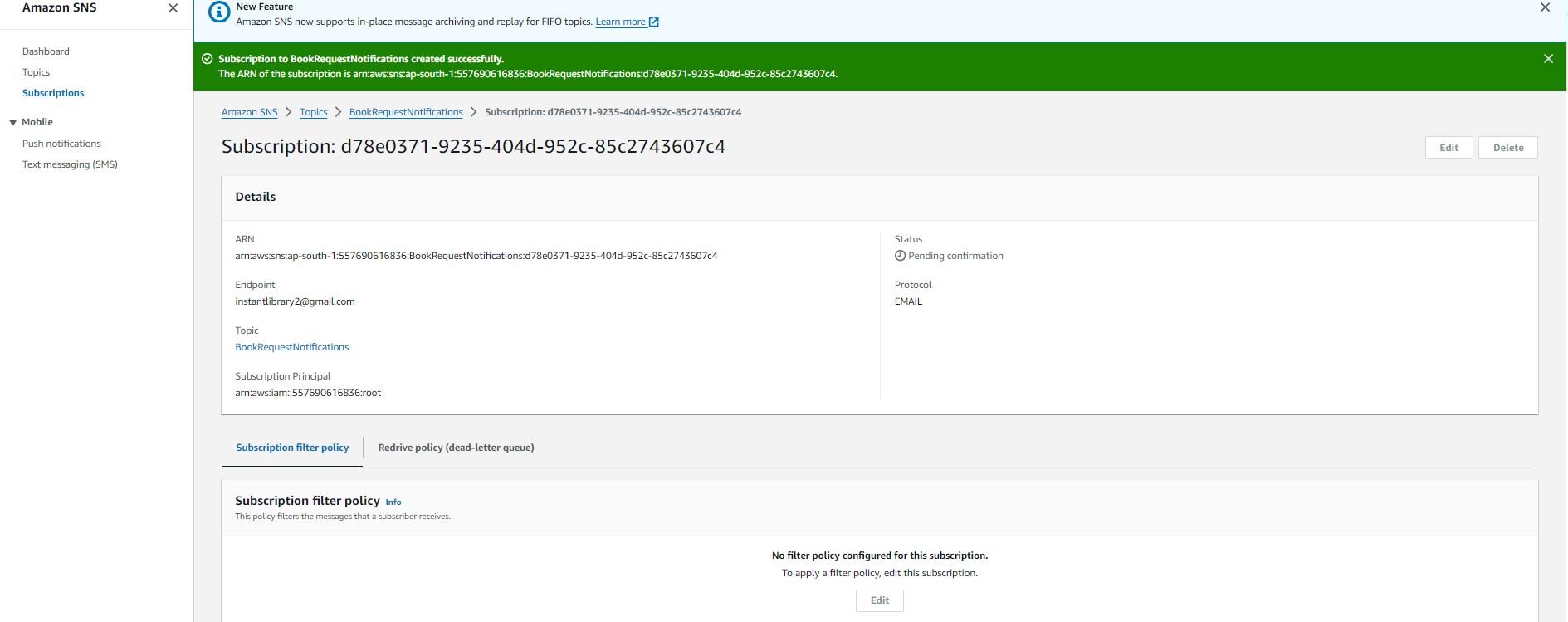
○ Configure the SNS topic and note down the **Topic ARN**.



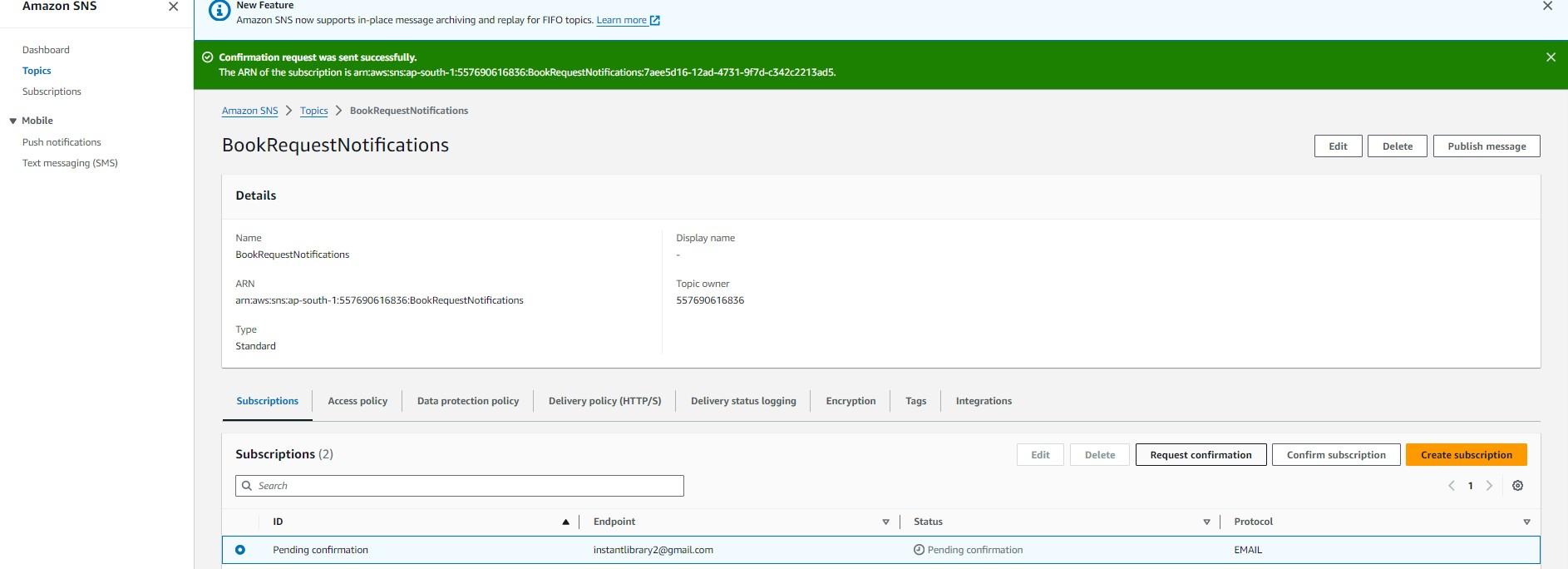
* **Activity 4.2: Subscribe users and staff to relevant SNS topics to receive real-time notifications when a book request is made.**

* 1. Subscribe users (or admin staff) to this topic via Email. When a book request is made, notifications will be sent to the subscribed emails**.**

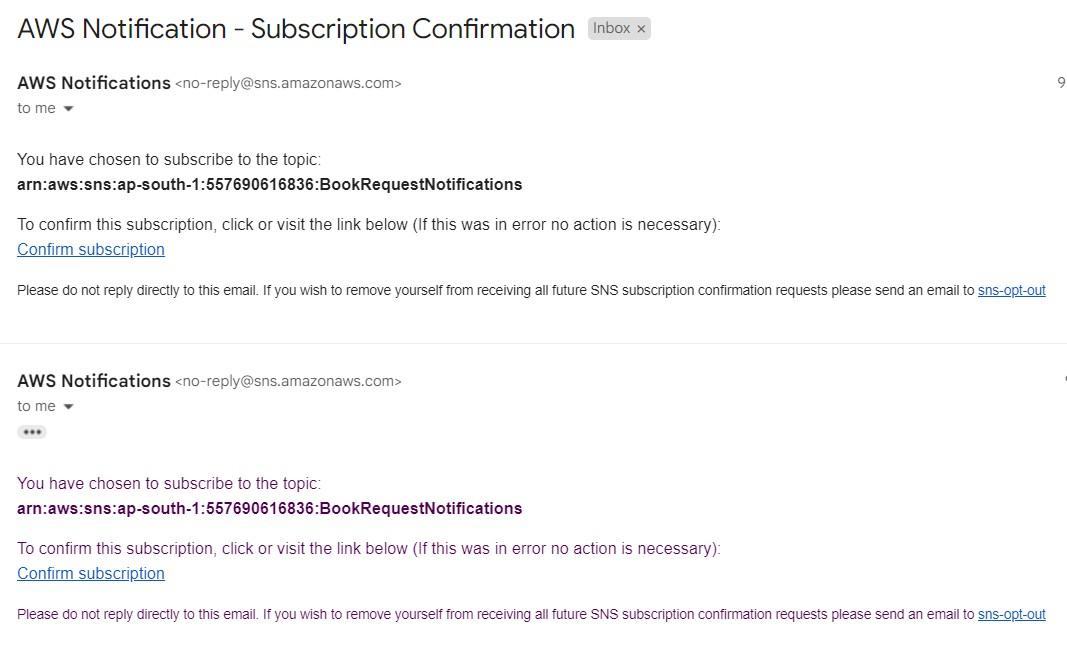


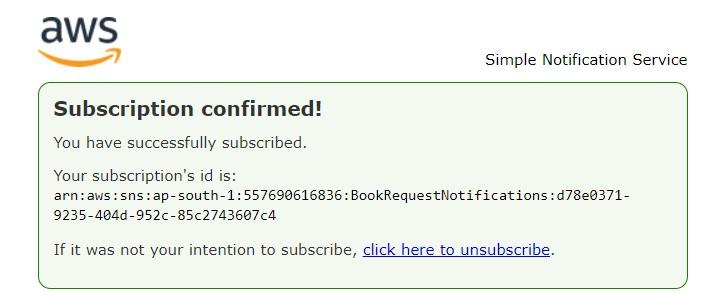


○ After subscription request for the mail confirmation

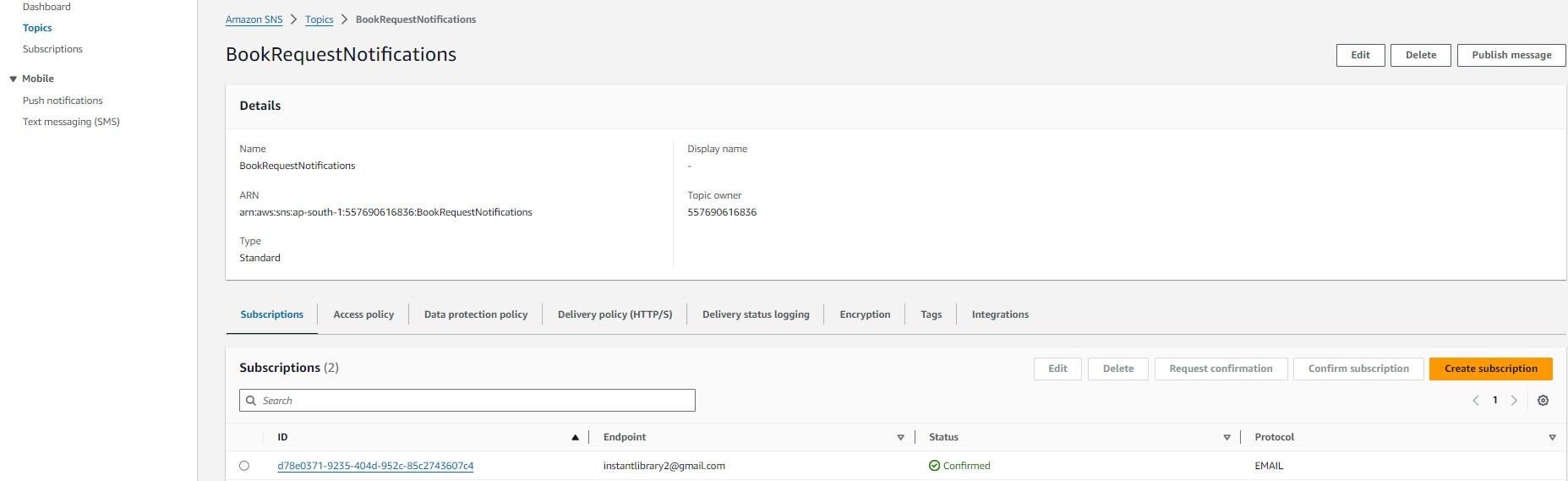


○ Navigate to the subscribed Email account and Click on the confirm subscription in the AWS Notification- Subscription Confirmation mail.





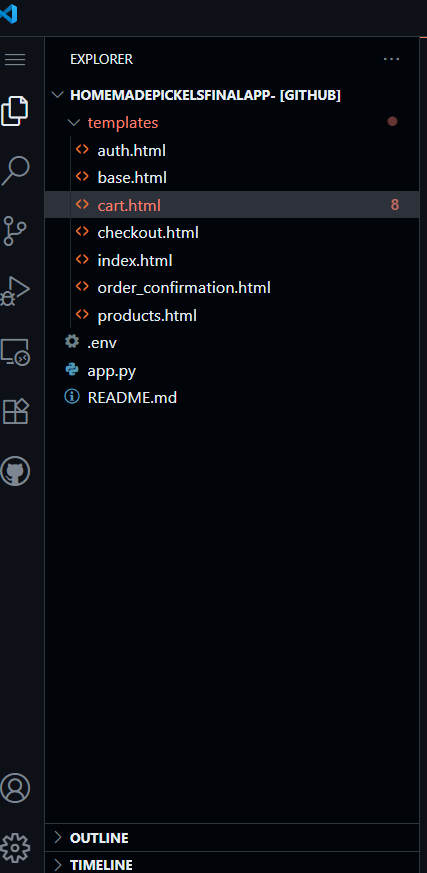
○ Successfully done with the SNS mail subscription and setup, now store the ARN link.



# Milestone 5:Backend Development and Application Setup

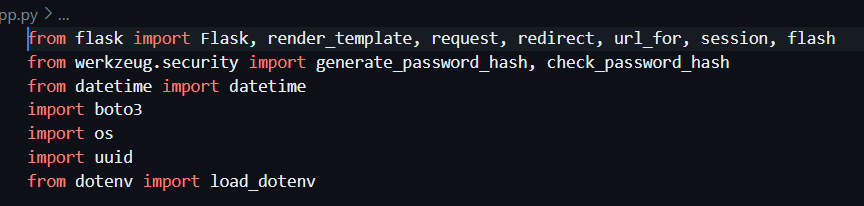
* **Activity 5.1: Develop the backend using Flask**

○ File Explorer Structure

 **Description:** set up the Home Made Pickles & Snacks: Taste the Best

This project is a Flask-based e-commerce platform for selling pickles and snacks, supporting both local and AWS (DynamoDB, SNS) deployments. It includes an app.py file for routing and logic, a templates/ folder for HTML pages like index.html, auth.html, cart.html, and a static/ folder for CSS, JS, and images. Environment config is managed through .env to toggle AWS or local mode.

**Description of the code :**

* **Flask App Initialization**
* 

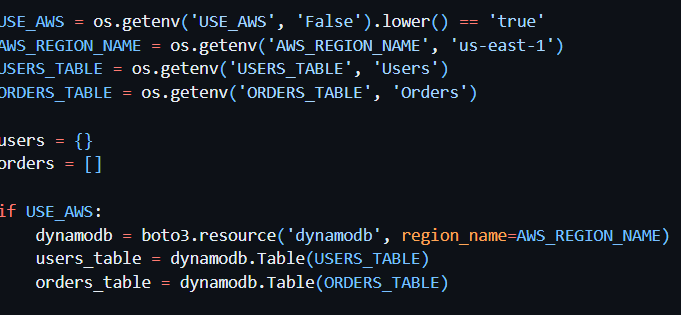
**Description:**

This project is a Flask-based web app for selling pickles and snacks, supporting both local and AWS deployments via DynamoDB and SNS. It includes app.py with routing logic, user authentication, and order management using Bcrypt for password hashing and Boto3 for database operations. The structure includes a templates/ folder for HTML pages and a static/ folder for assets, with .env handling configuration.



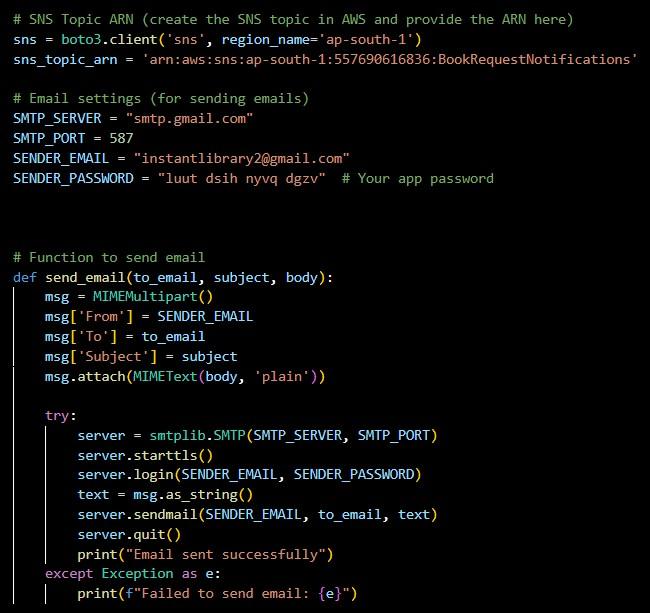
**Description:** initialize the Flask application instance using Flask(\_\_name\_\_) to start building the web app.

* **Dynamodb Setup:**



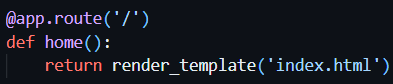
**Description:** initialize the DynamoDB resource for the ap-south-1 region and set up access to the Users and Requests tables for storing user details and book requests.

* **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**

* **Home Route:**

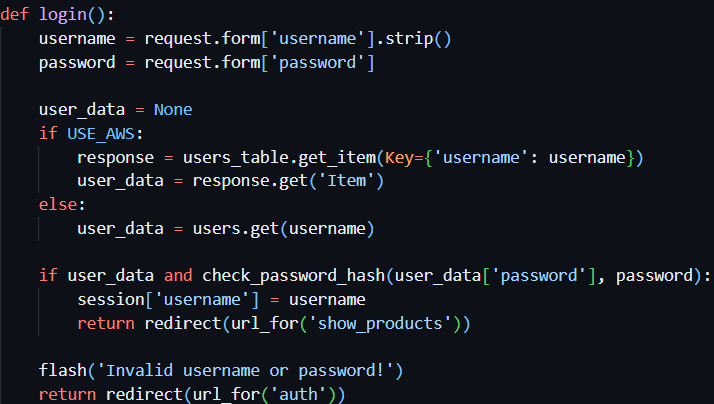


**Description:** define the home route / to automatically redirect users to the register page when they access the base URL.

* **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



**Deployment Code:**

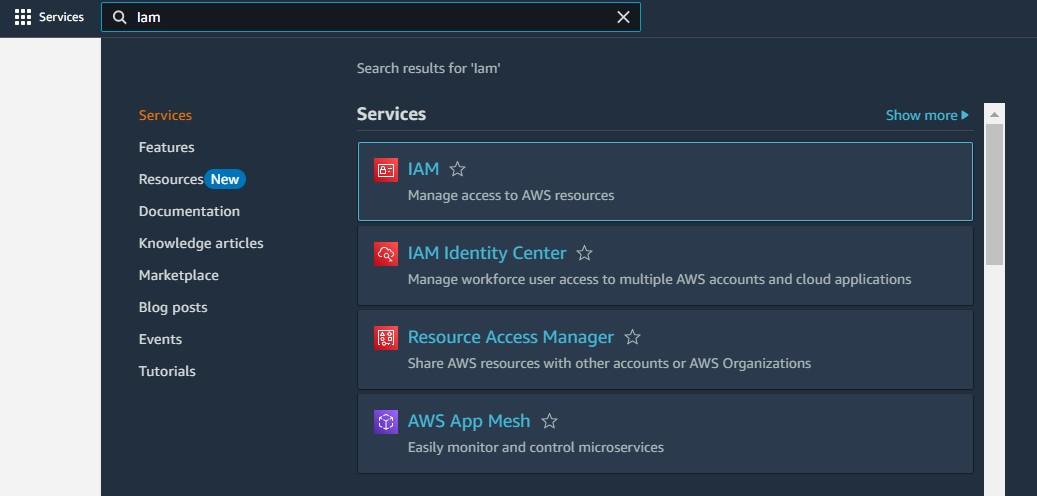


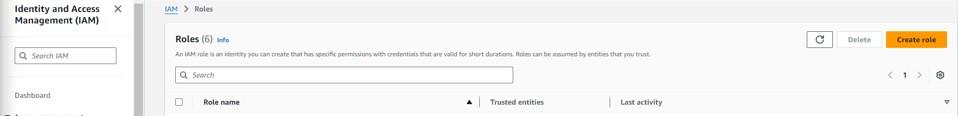
**Description:** start the Flask server to listen on all network interfaces (0.0.0.0) at port 80 with debug mode enabled for development and testing.

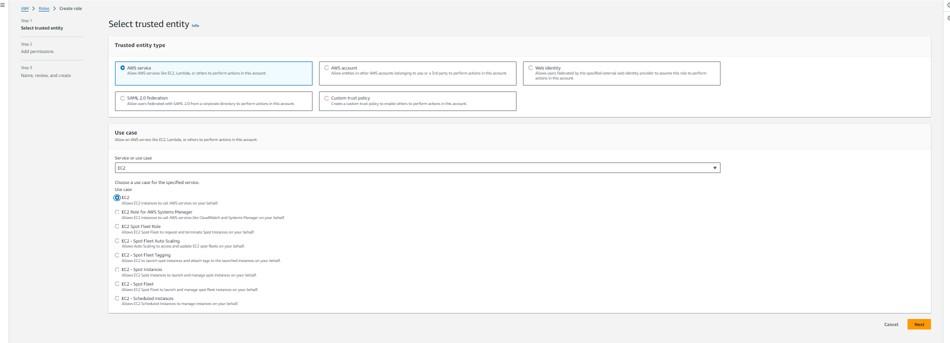
# Milestone 6: IAM Role Setup

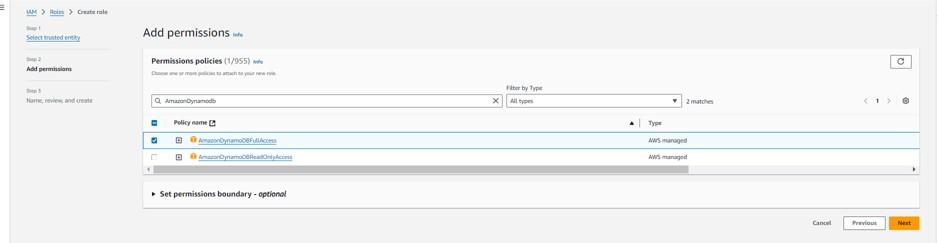
* **Activity 6.1:Create IAM Role.**

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





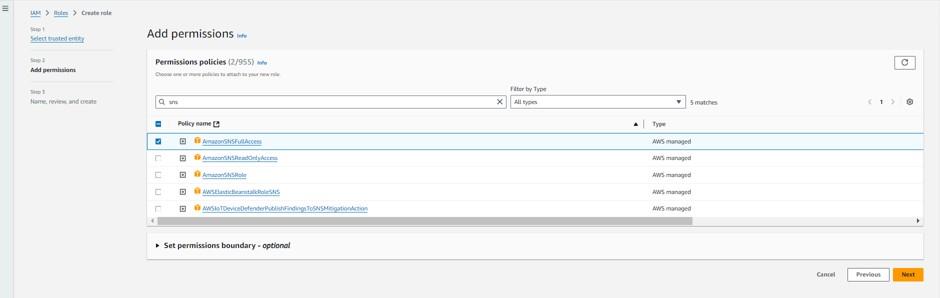


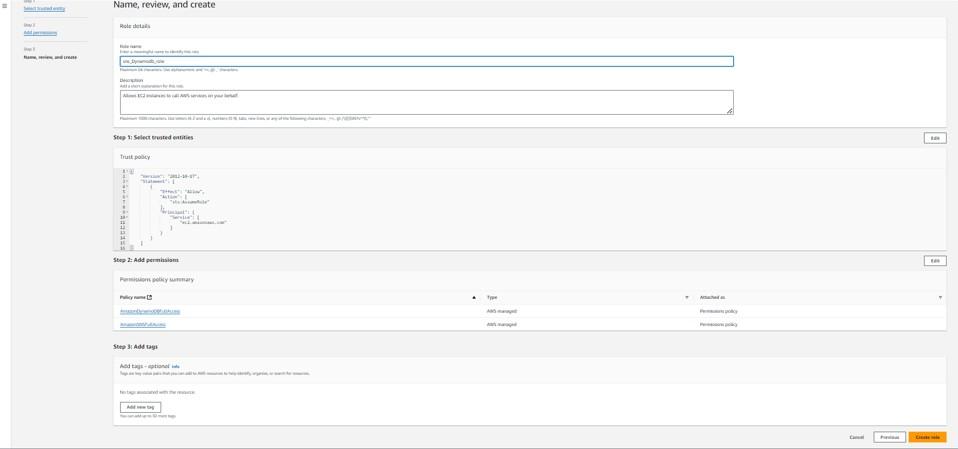


* **Activity 6.2: Attach Policies.**

Attach the following policies to the role:

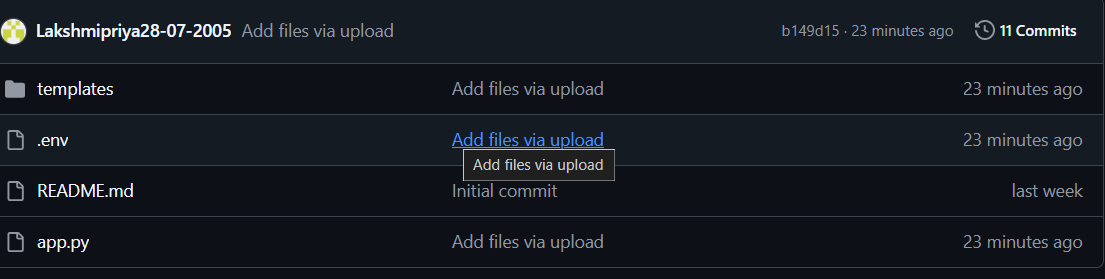
* AmazonDynamoDBFullAccess: Allows EC2 to perform read/write operations on DynamoDB.
* AmazonSNSFullAccess: Grants EC2 the ability to send notifications via SNS.

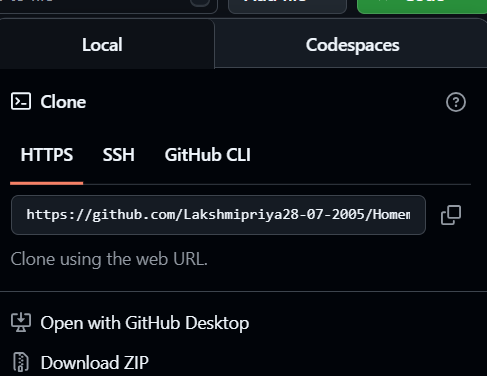




# Milestone 7: EC2 Instance Setup

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

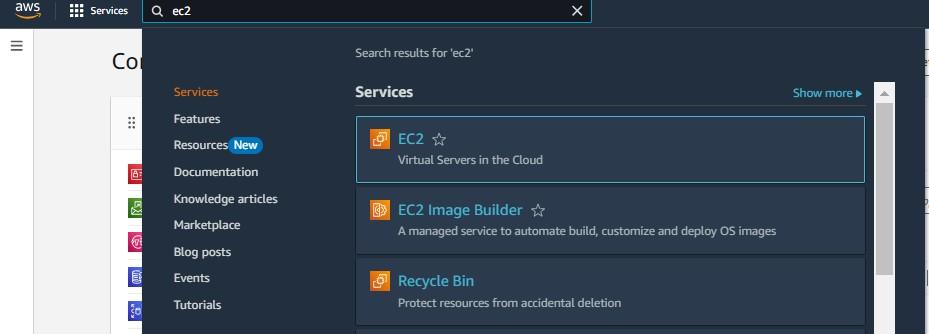




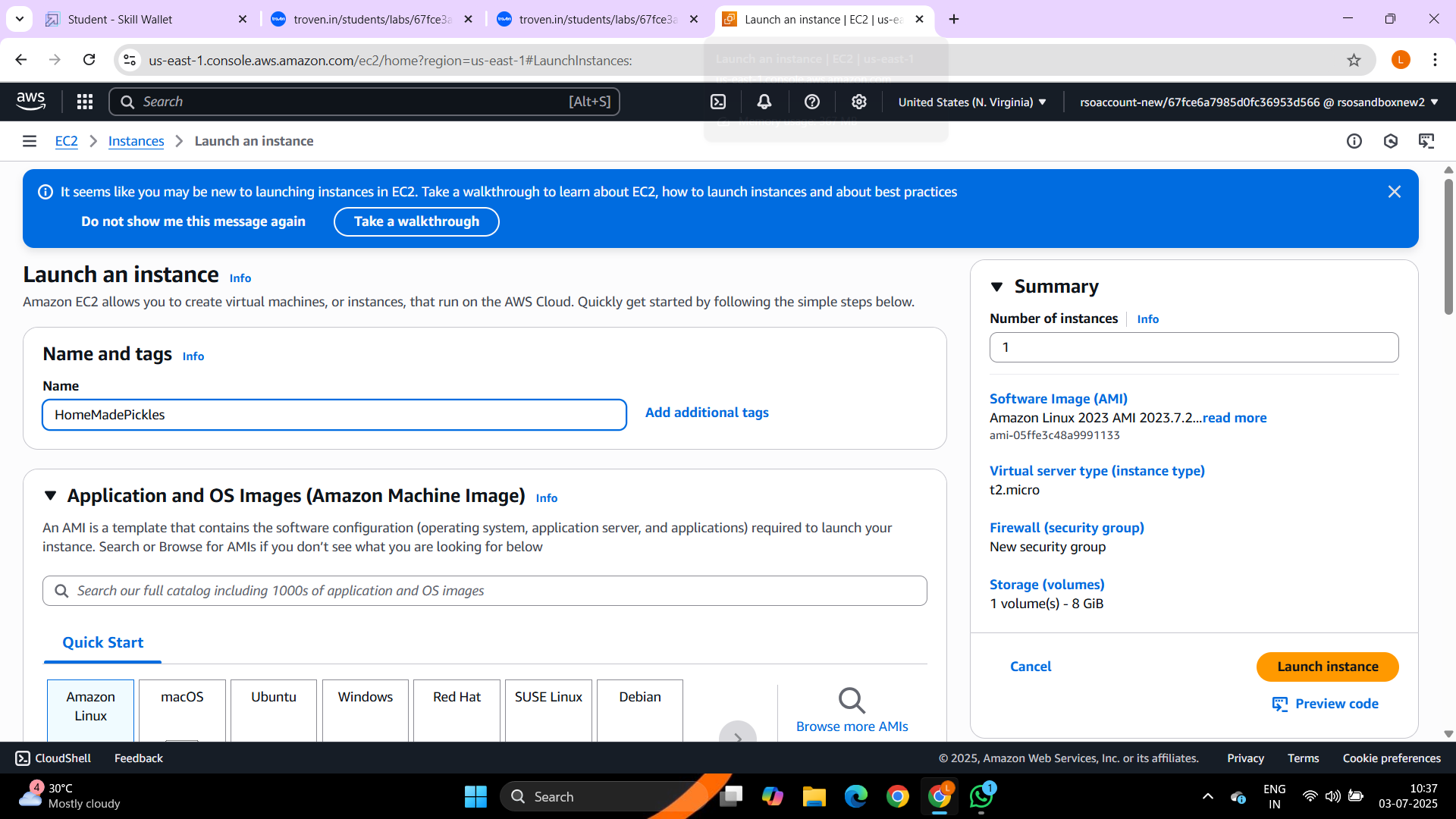
* **Activity 7.1: Launch an EC2 instance to host the Flask application.**

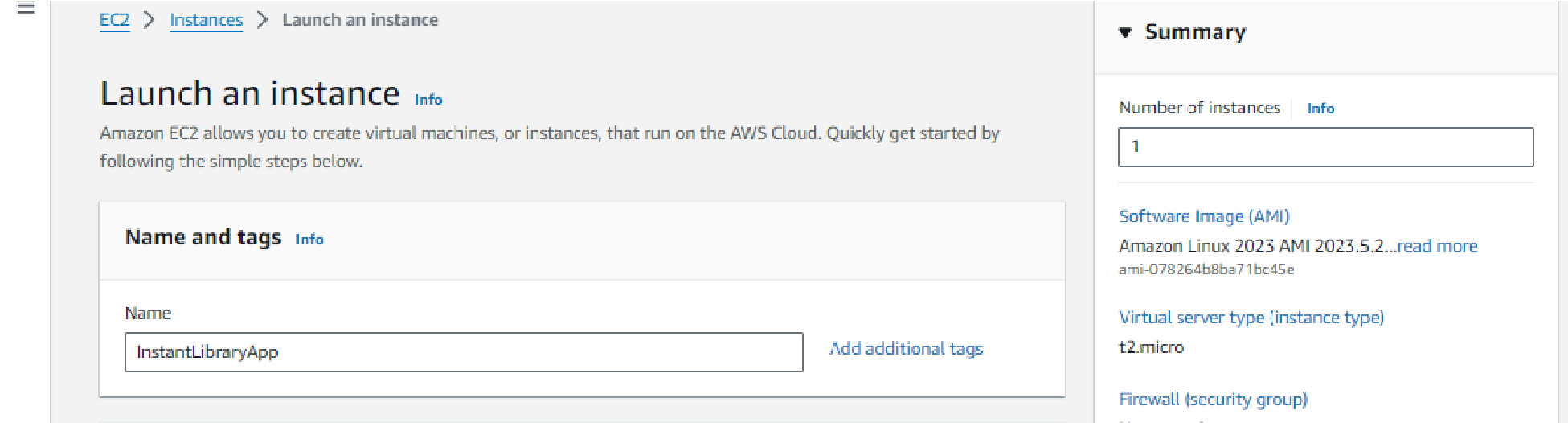
* **Launch EC2 Instance**

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

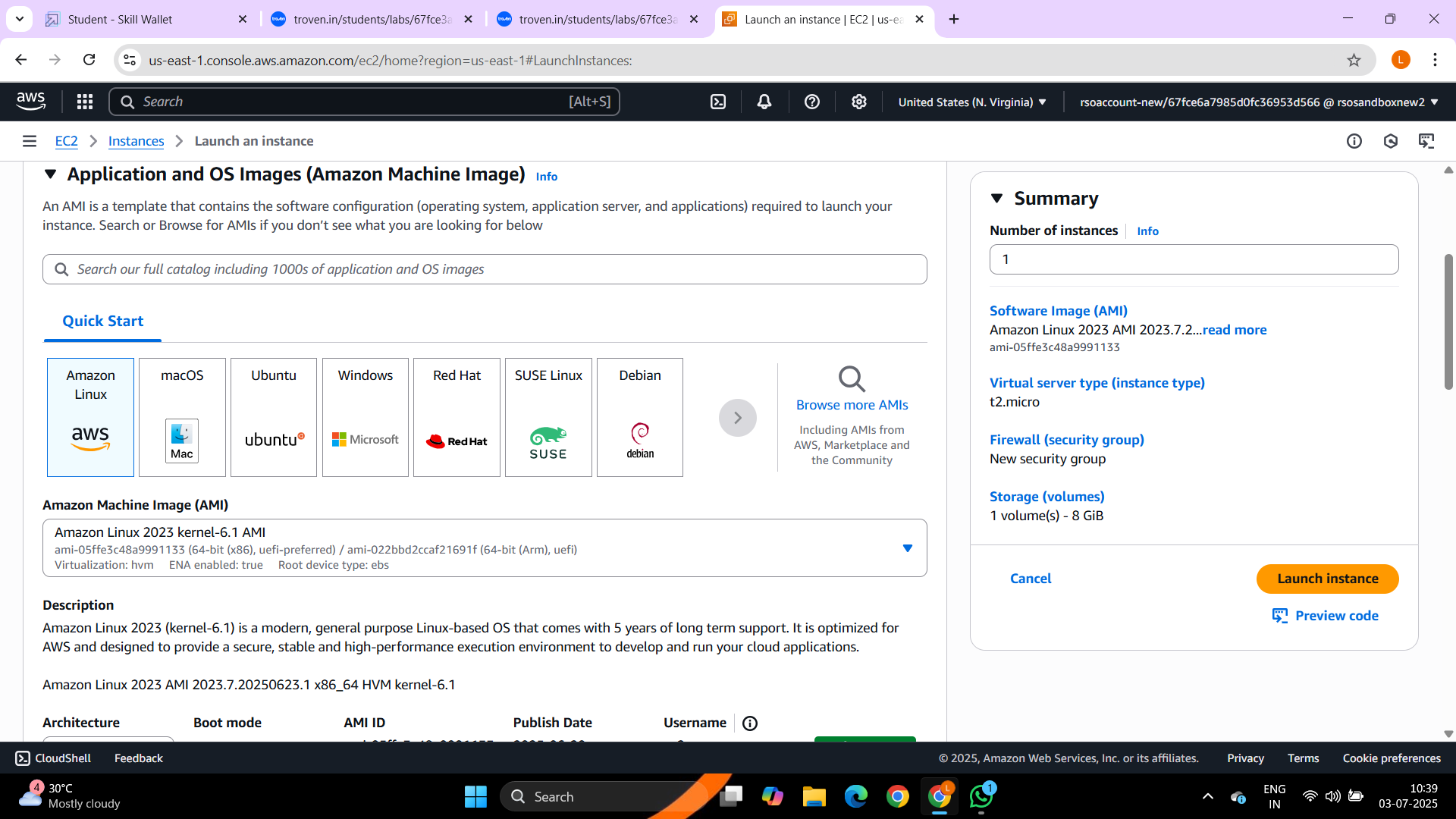


* Click on Launch instance to launch EC2 instance

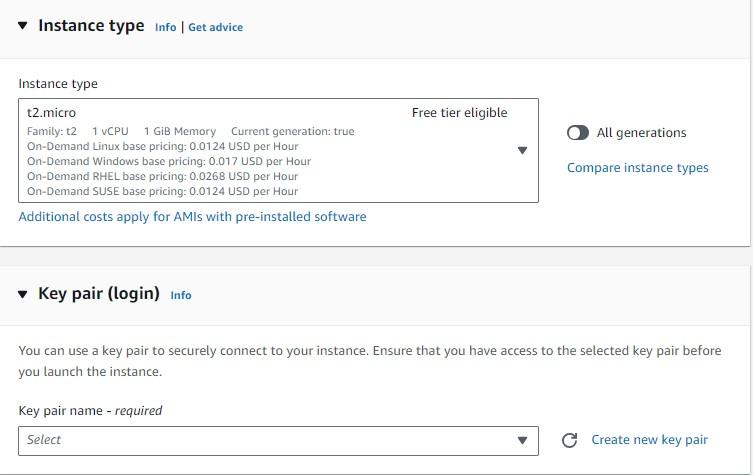


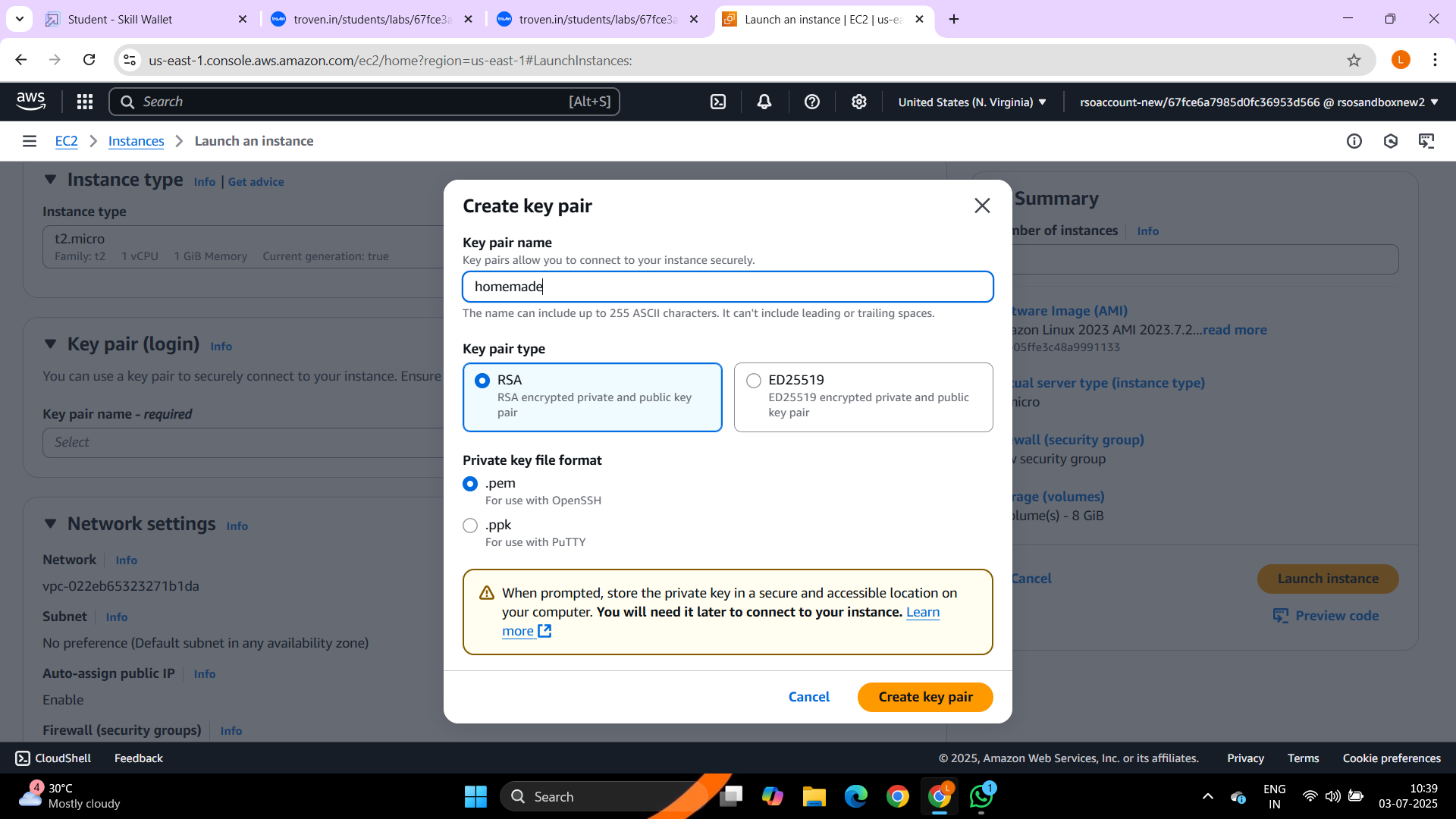


* Choose Amazon Linux 2 or Ubuntu as the AMI and t2.micro as the instance type (free-tier eligible).



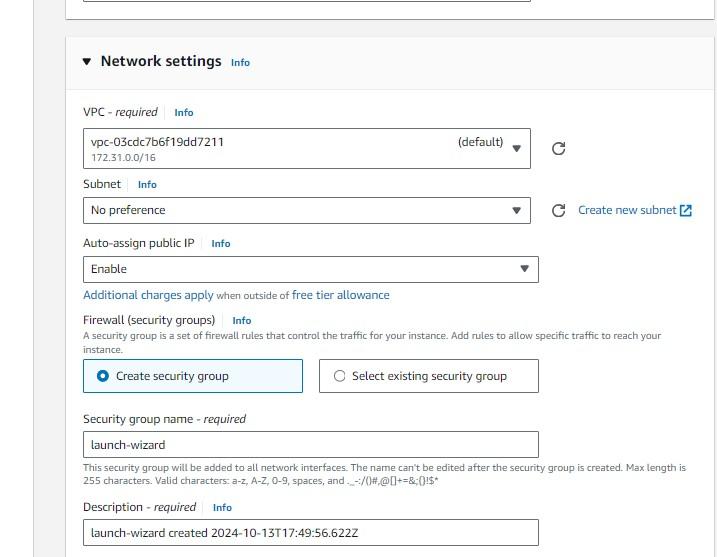
* Create and download the key pair for Server access.

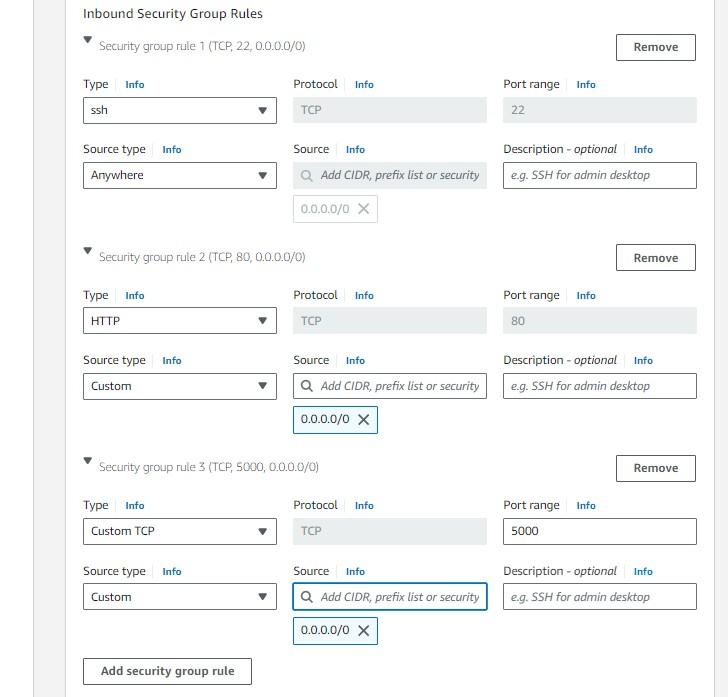


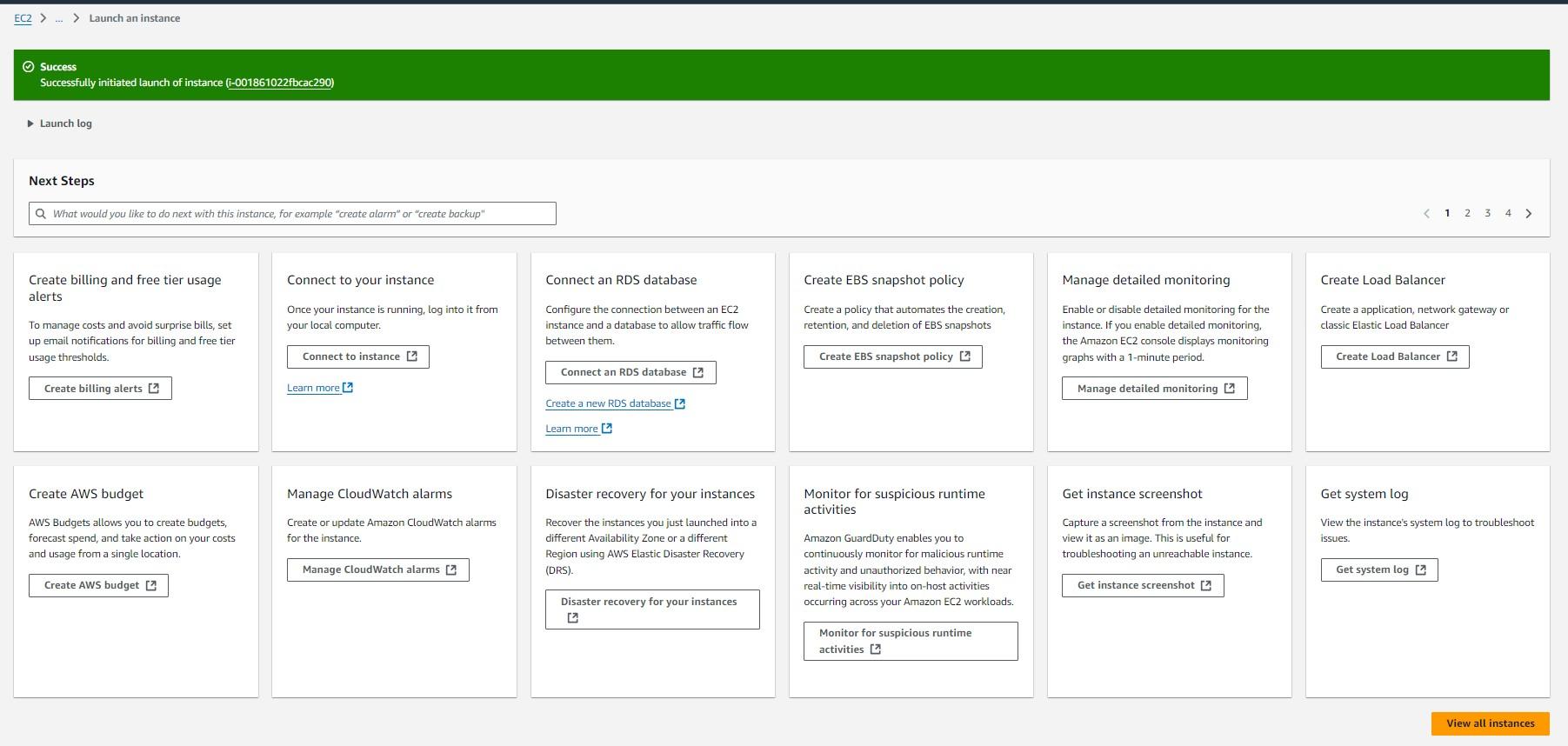




* **Activity 7.2:Configure security groups for HTTP, and SSH access.**

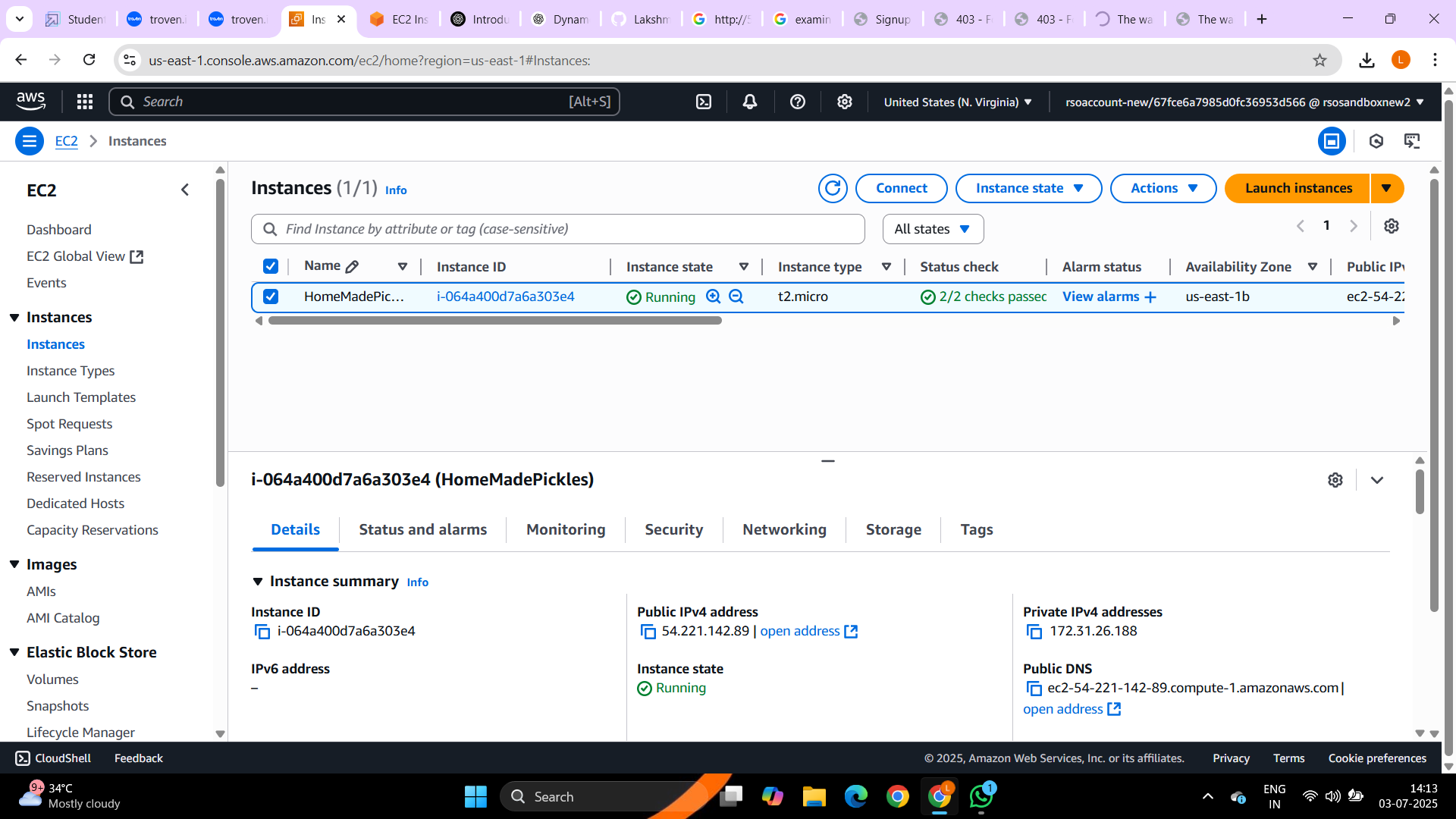


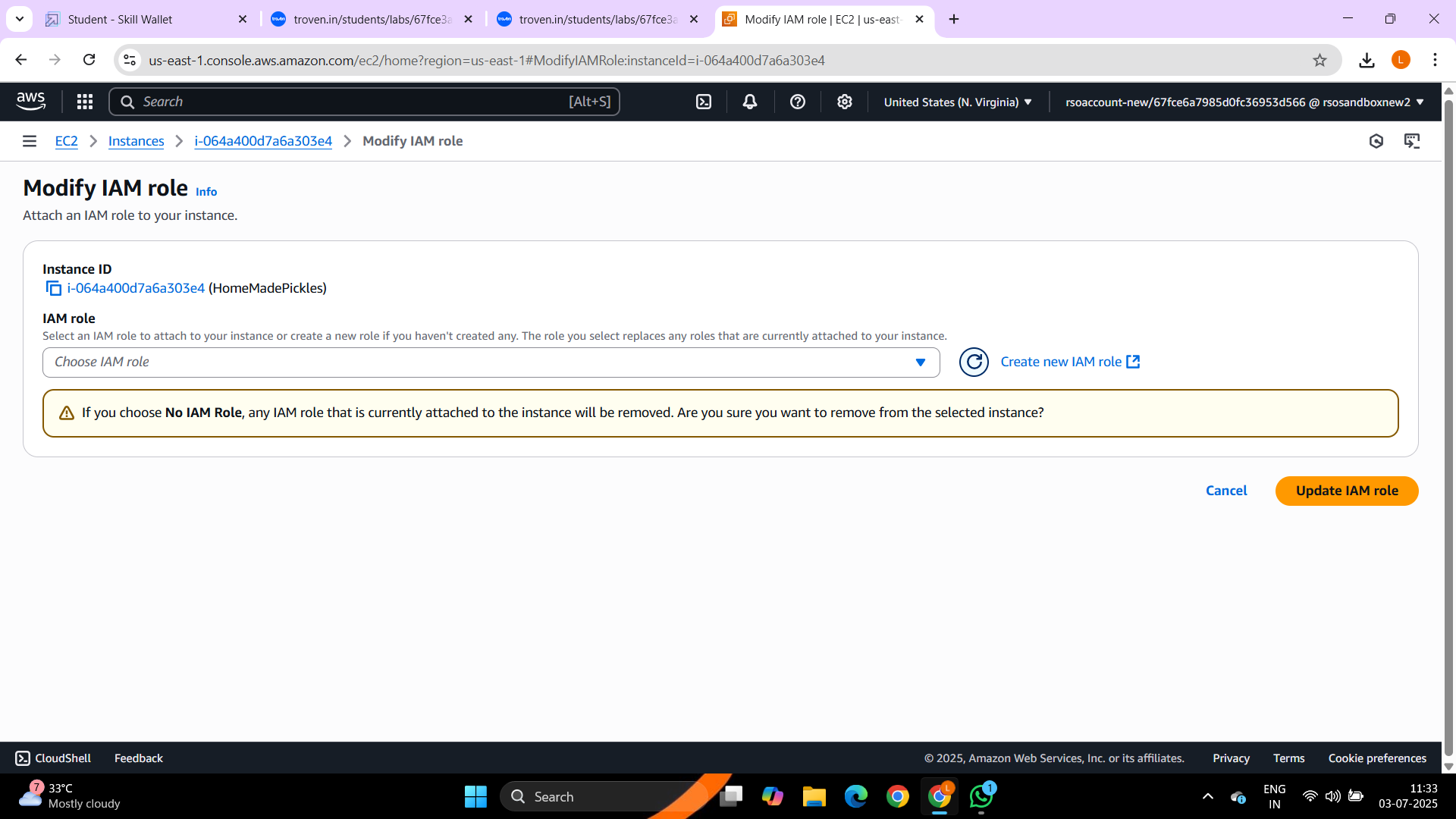




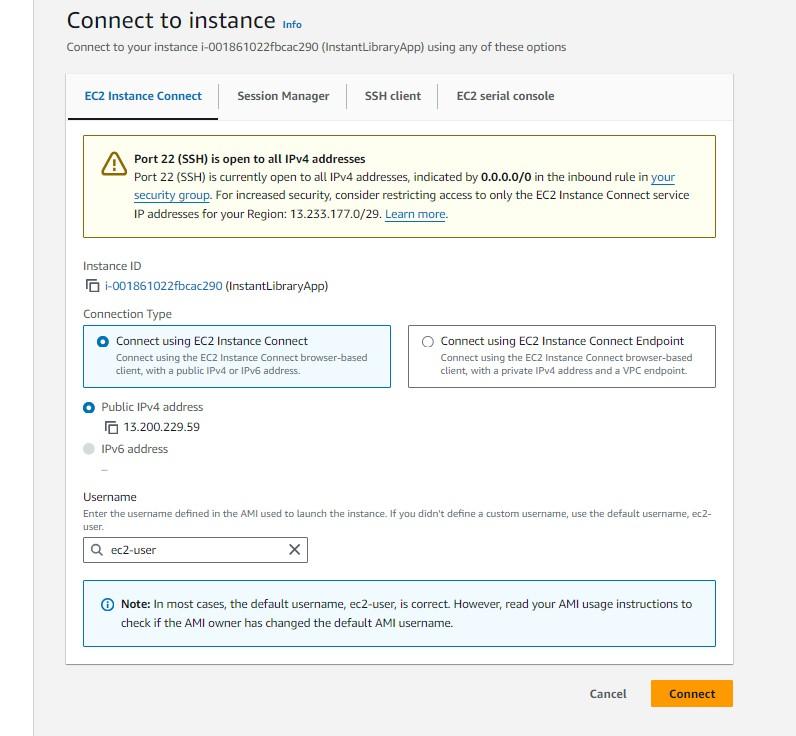
* To connect to EC2 using **EC2 Instance Connect**, start by ensuring that an **IAM role** is attached to your EC2 instance. You can do this by selecting your instance, clicking on **Actions**, then navigating to **Security** and selecting **Modify IAM Role** to attach the appropriate role. After the 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

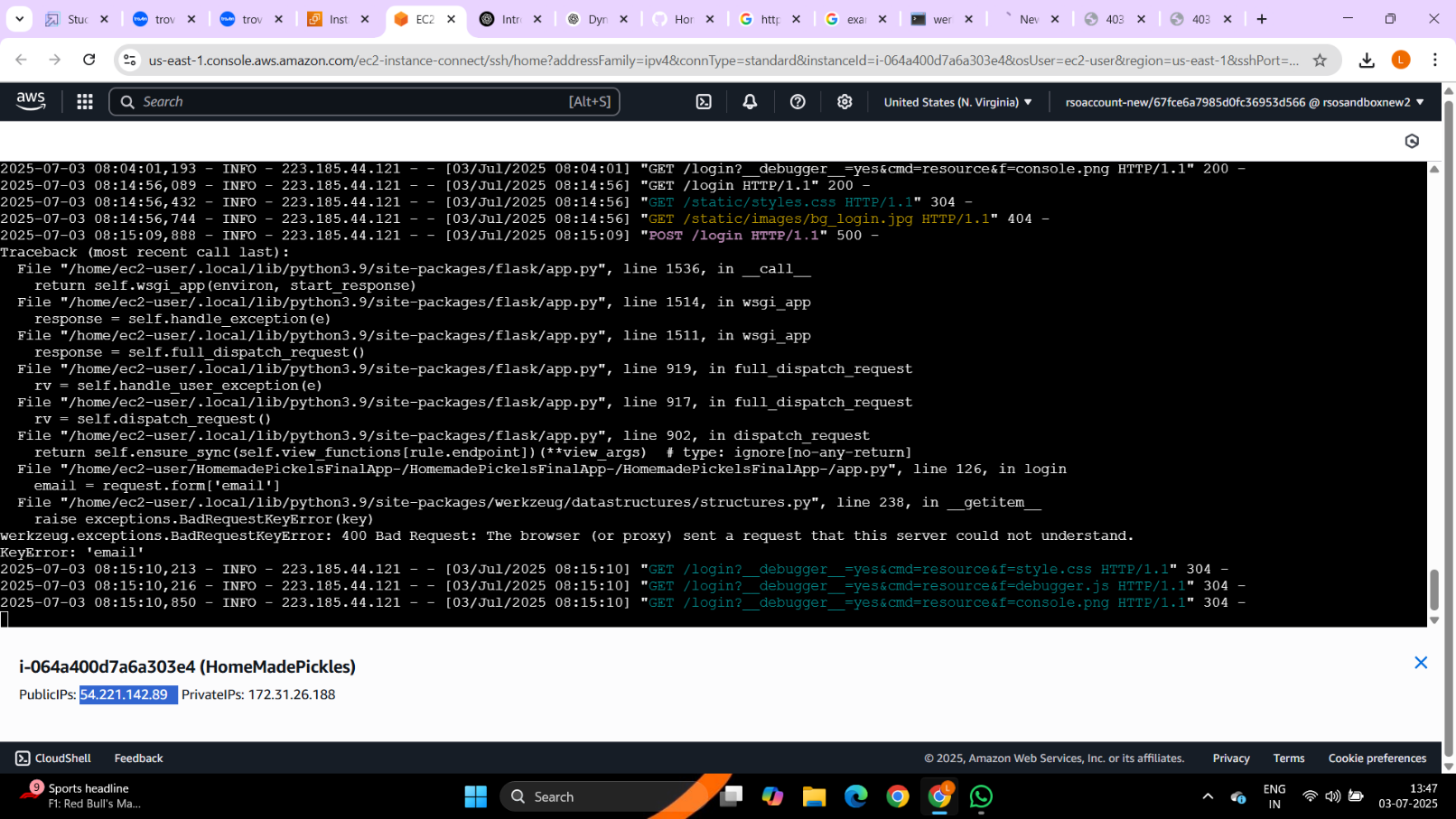
**EC2 Dashboard**, click the **Connect** button. From the connection methods presented, choose **EC2 Instance Connect**. Finally, click **Connect** again, and a new browser-based terminal will open, allowing you to access your EC2 instance directly from your browser.





* Now connect the EC2 with the files





# Milestone 8: Deployment on EC2

**Activity 8.1: 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 8.2: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:‘gitclone https://github.com/BathinaSivani/pickle-shop

● This will download your project to the EC2 instance.

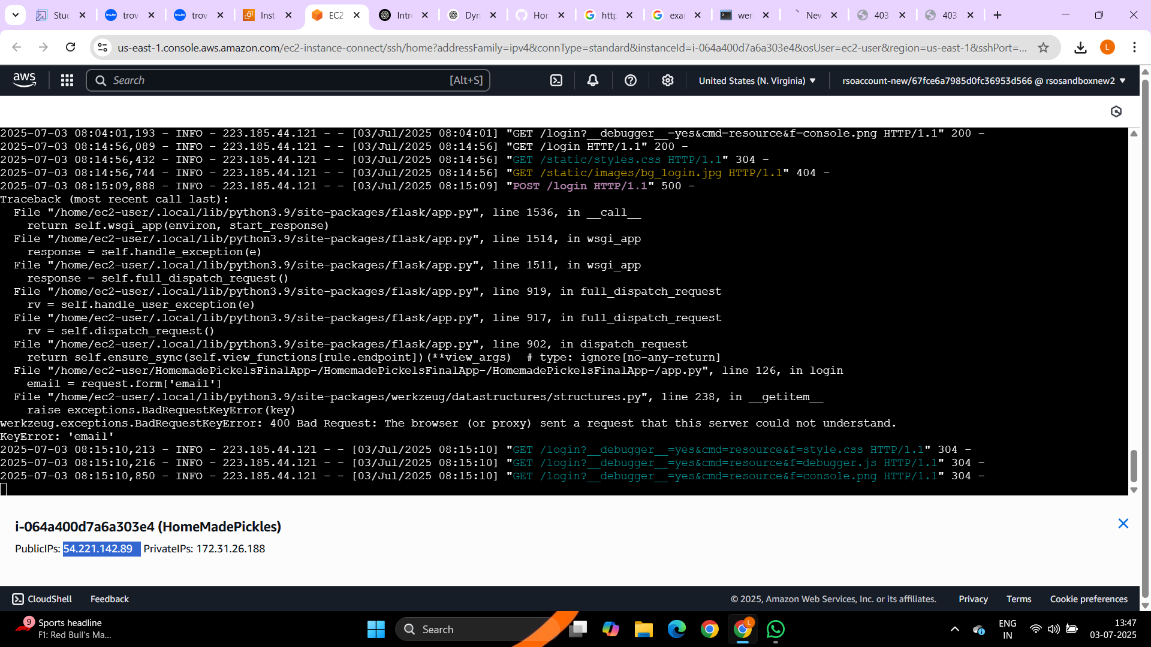
**To navigate to the project directory, run the following command:**

cd store-

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

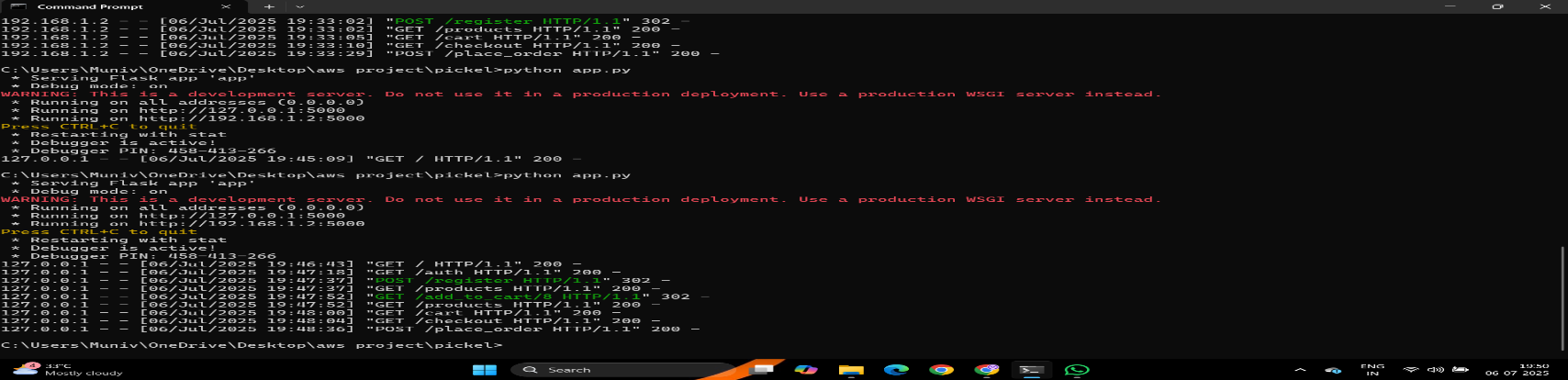
**Run the Flask Application**

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



**Verify the Flask app is running**: http://your-ec2-public-ip

## ○ Run the Flask app on the EC2 instance

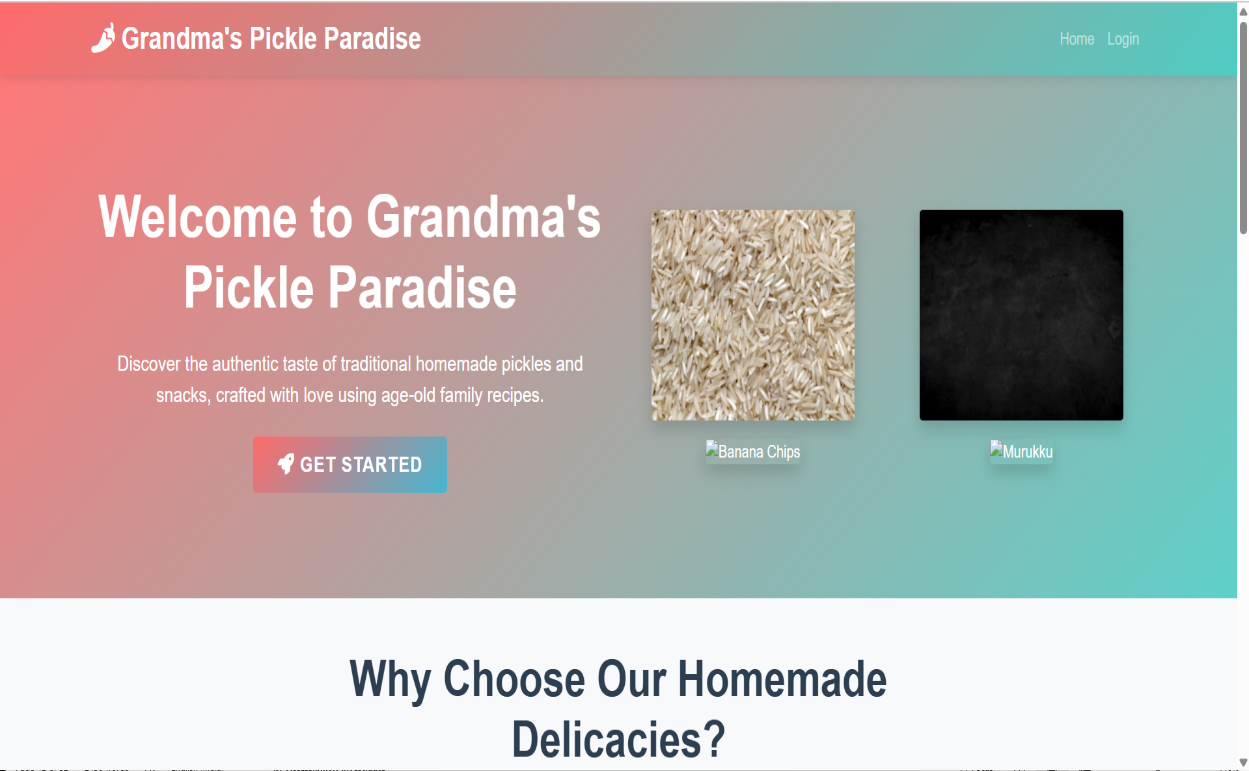


**Access the website through:**

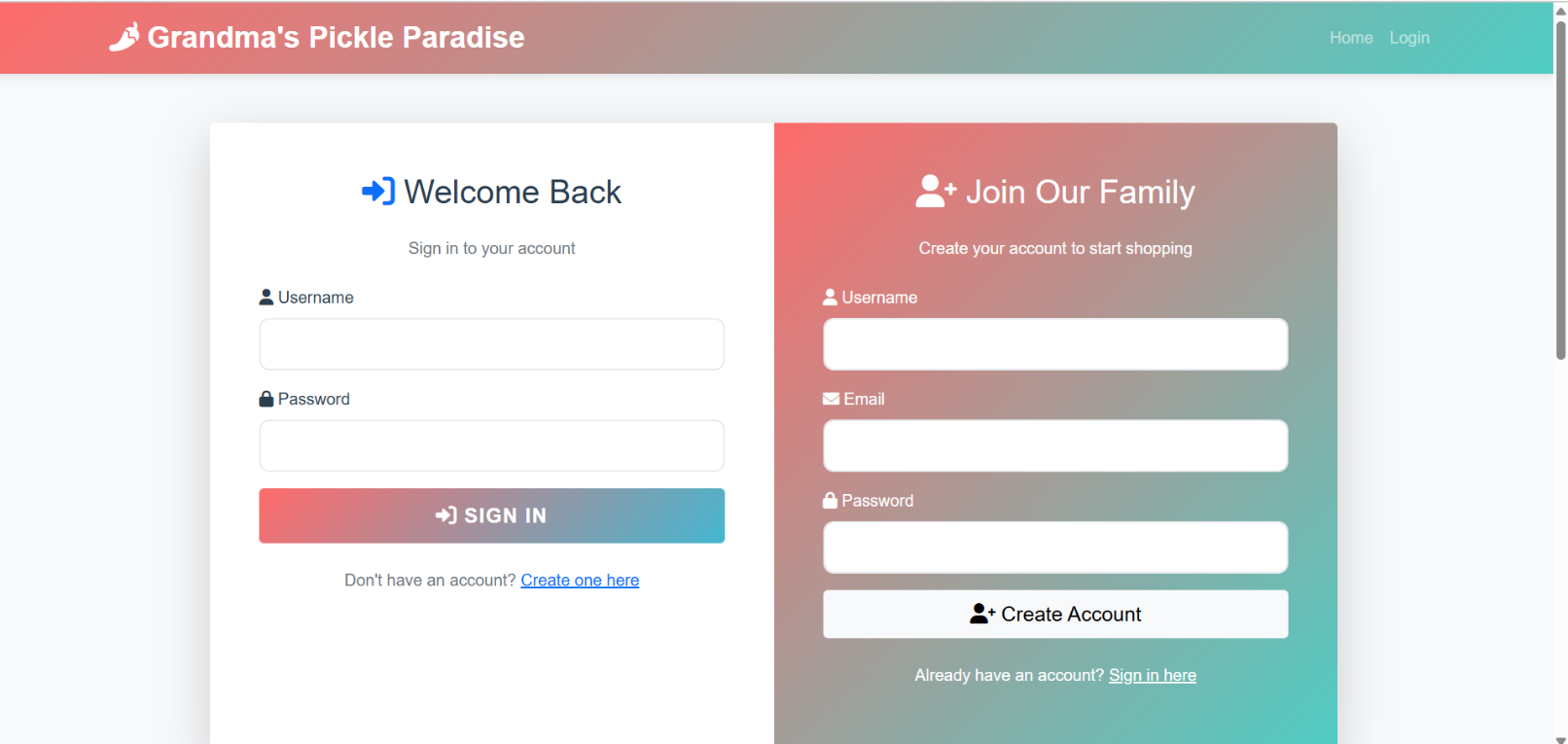
**PublicIPs:http://54.221.142.89/8000**

**Milestone 9: Testing and Deployment**

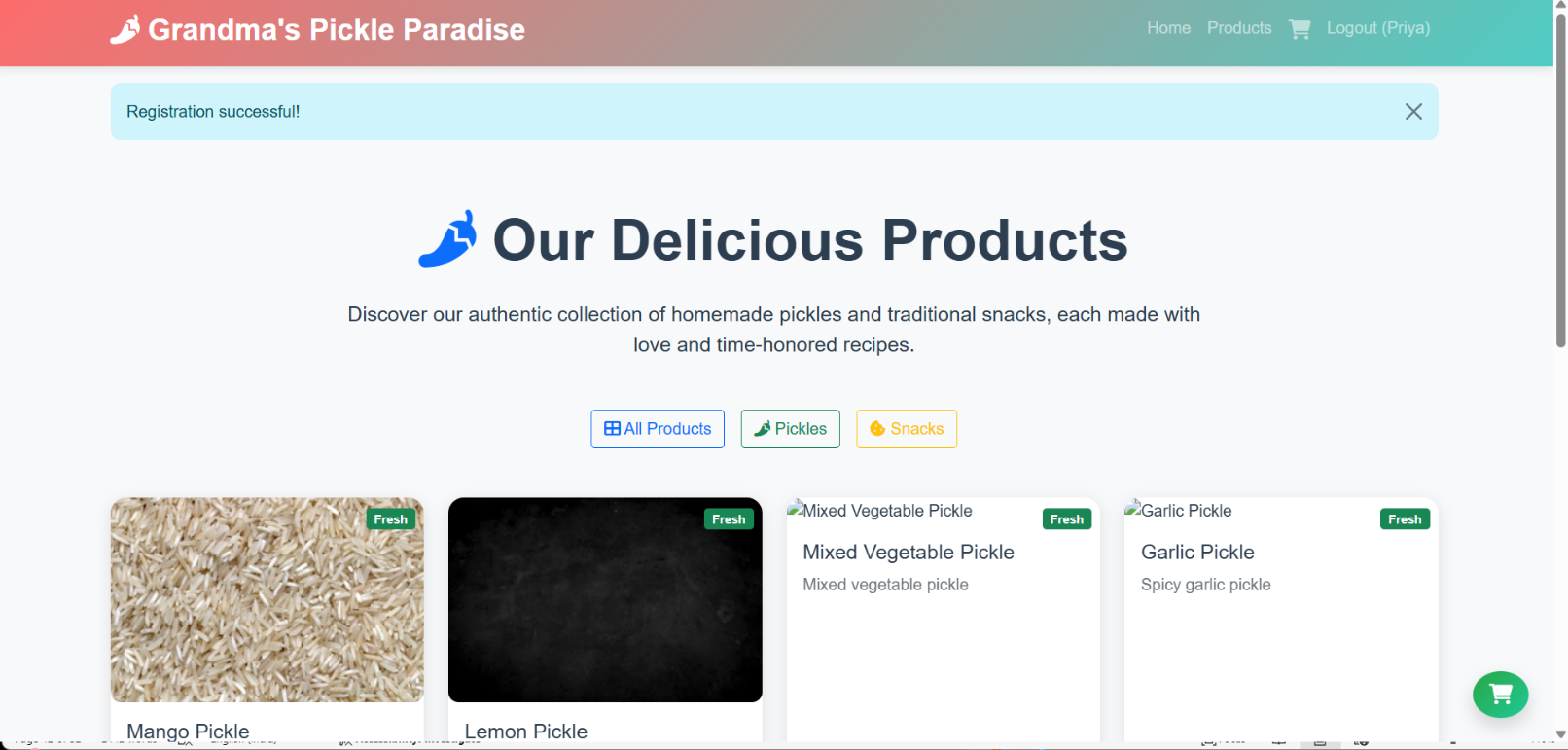
● **Activity 9.1:**



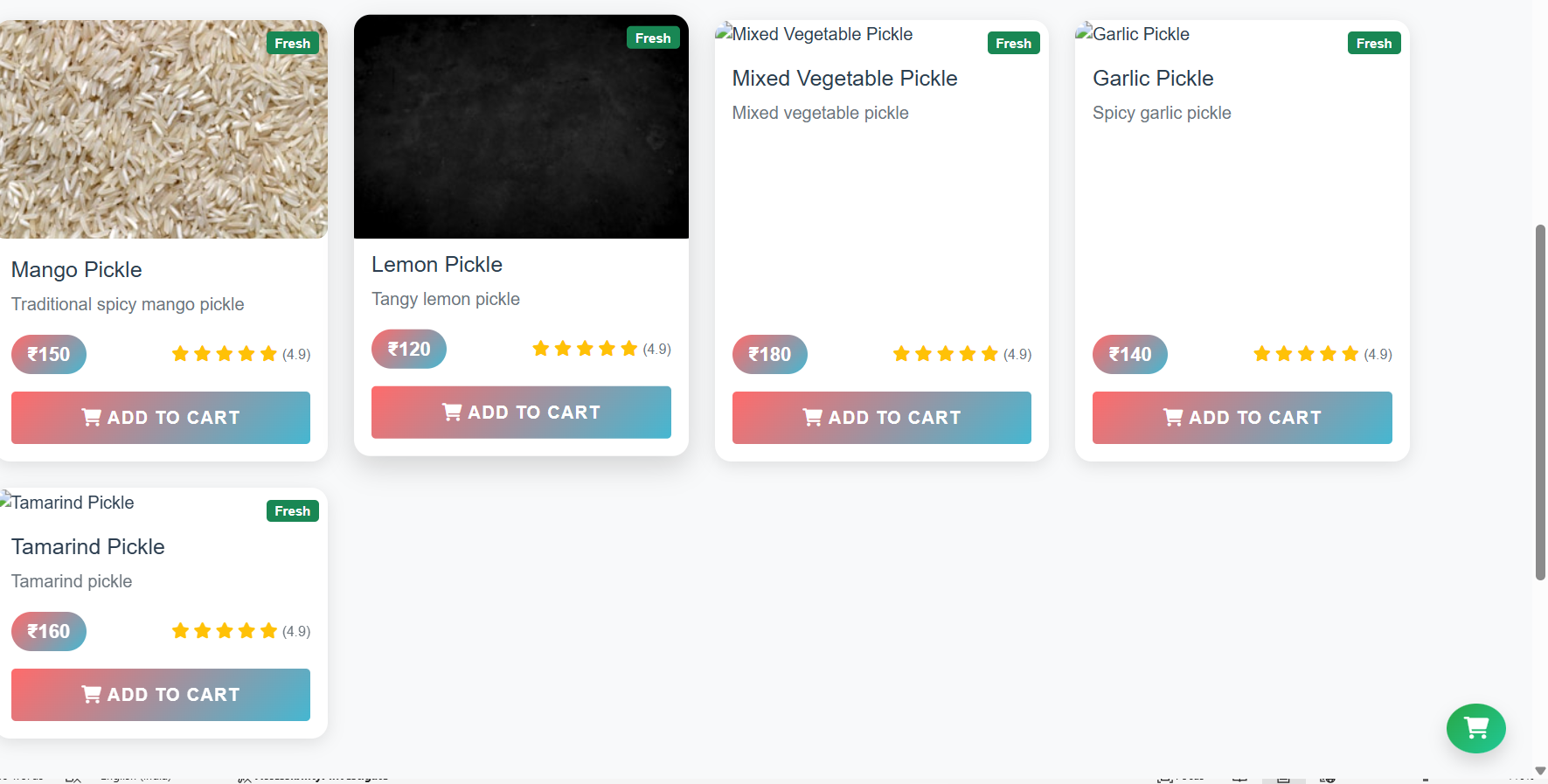
**Register Page:**



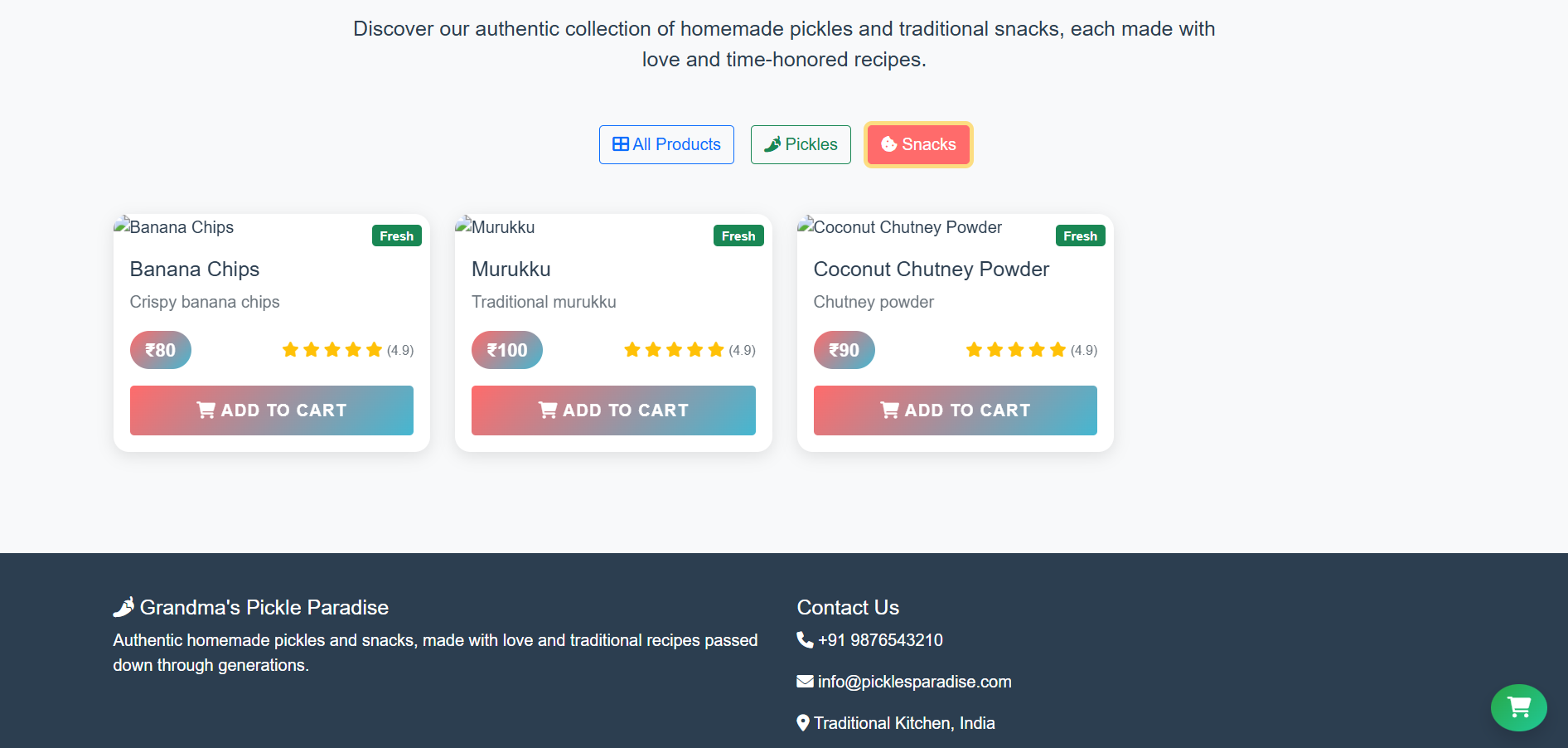
**Home page:**



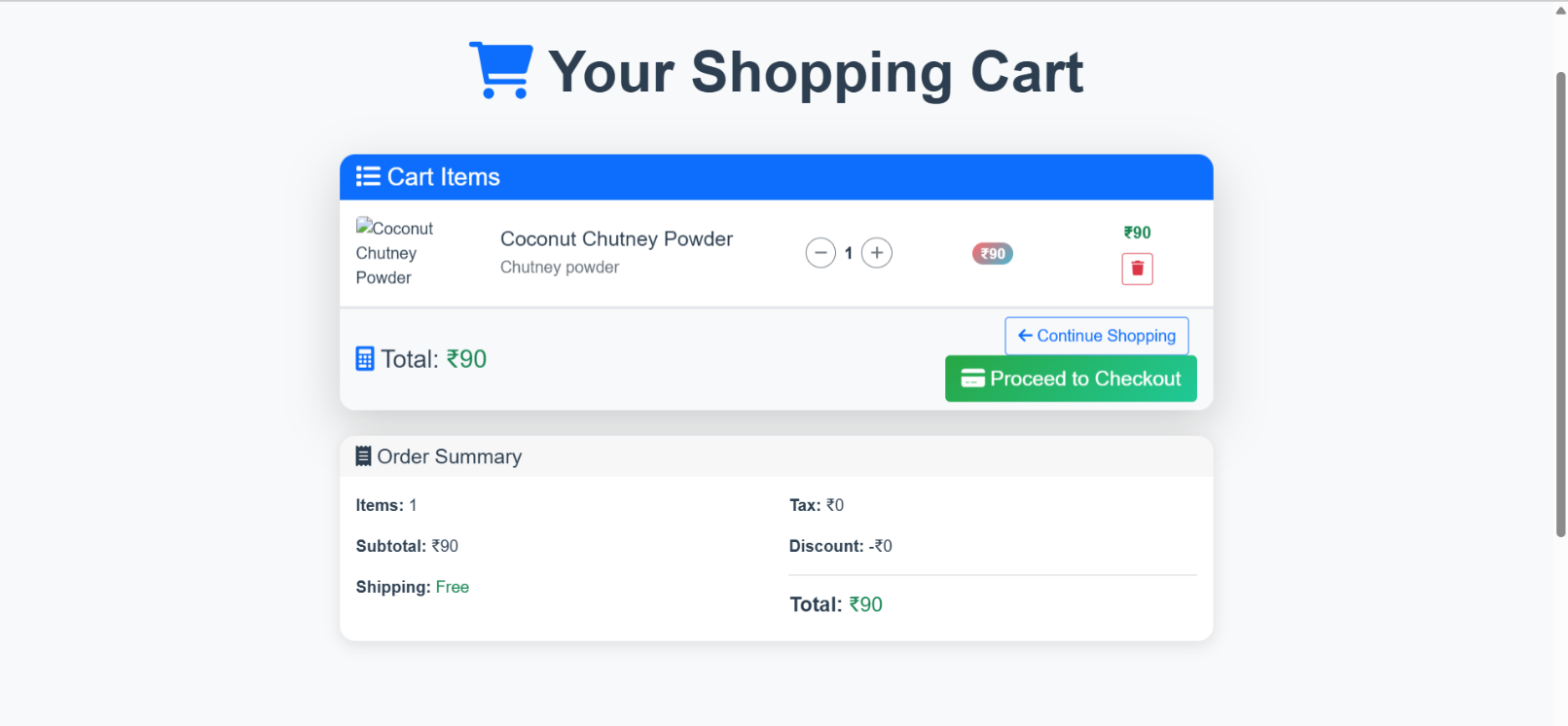
Pickels:



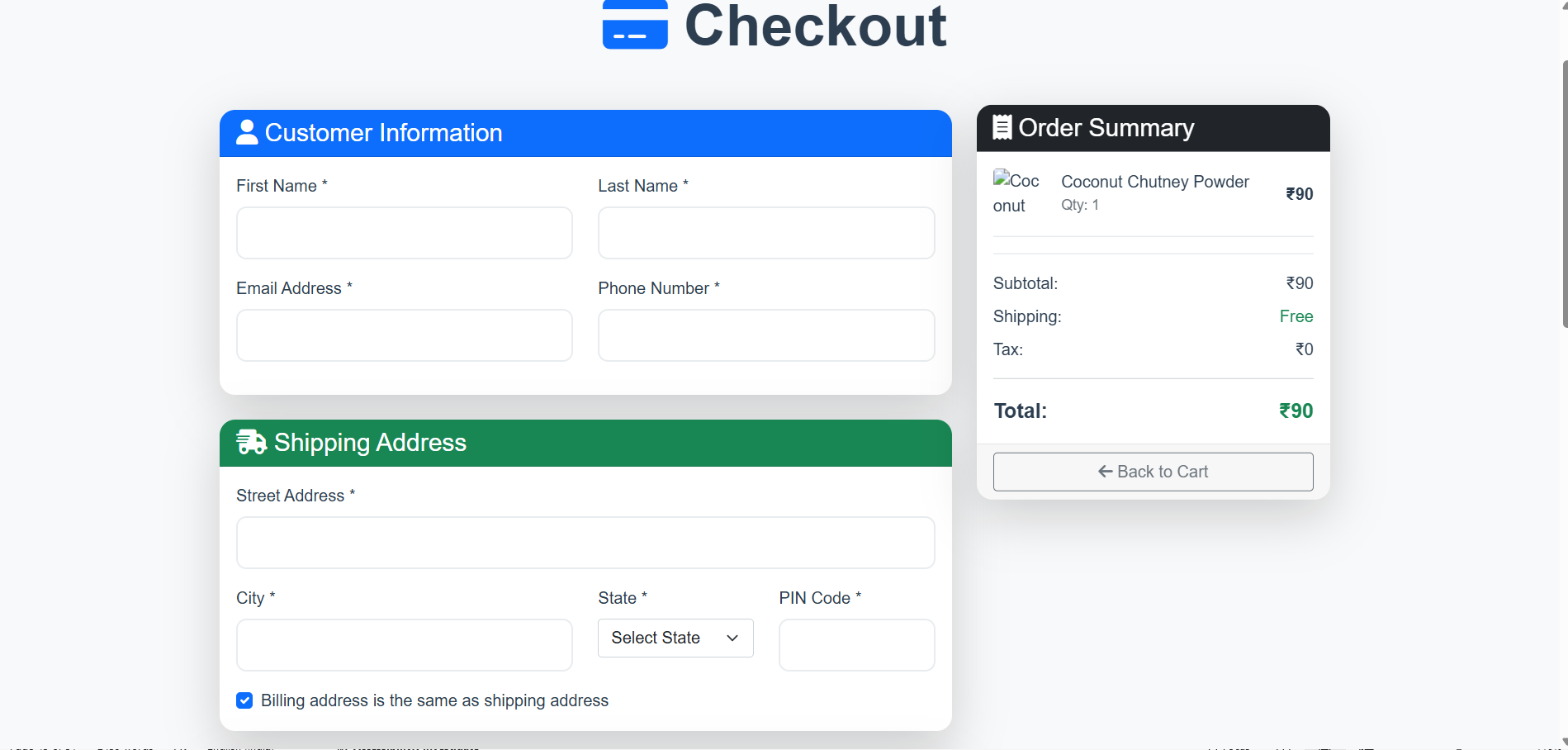
**Snacks:**

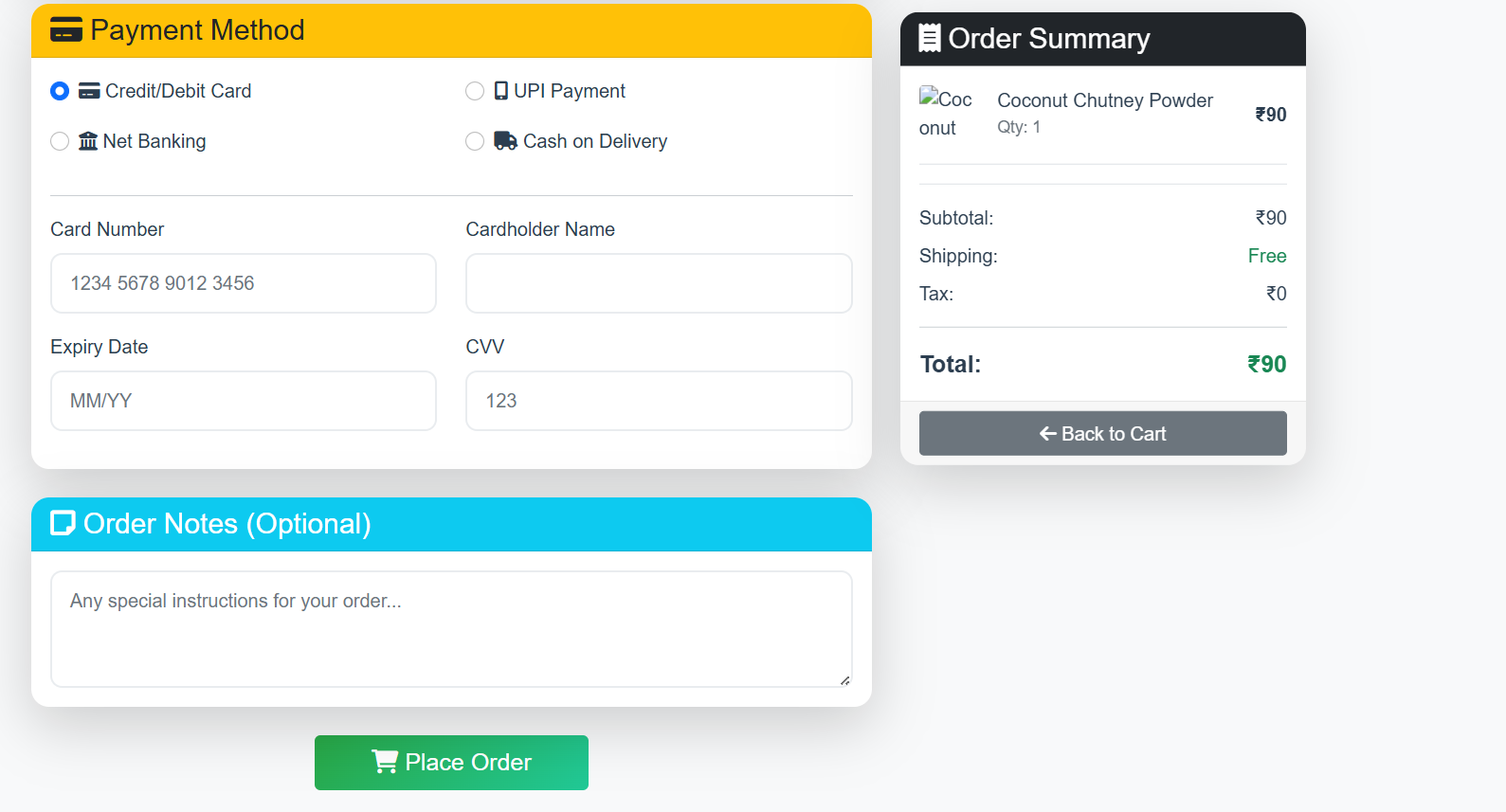


**Cart page:**

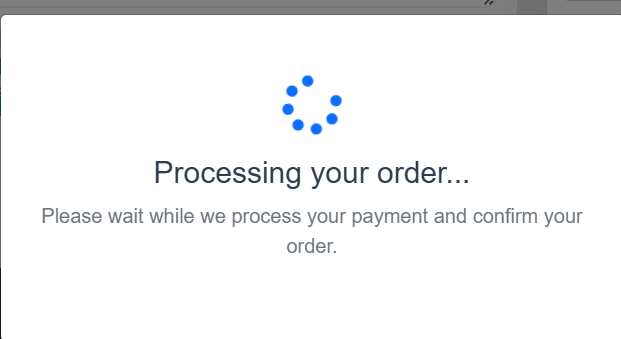


**Proceed to Checkout:**

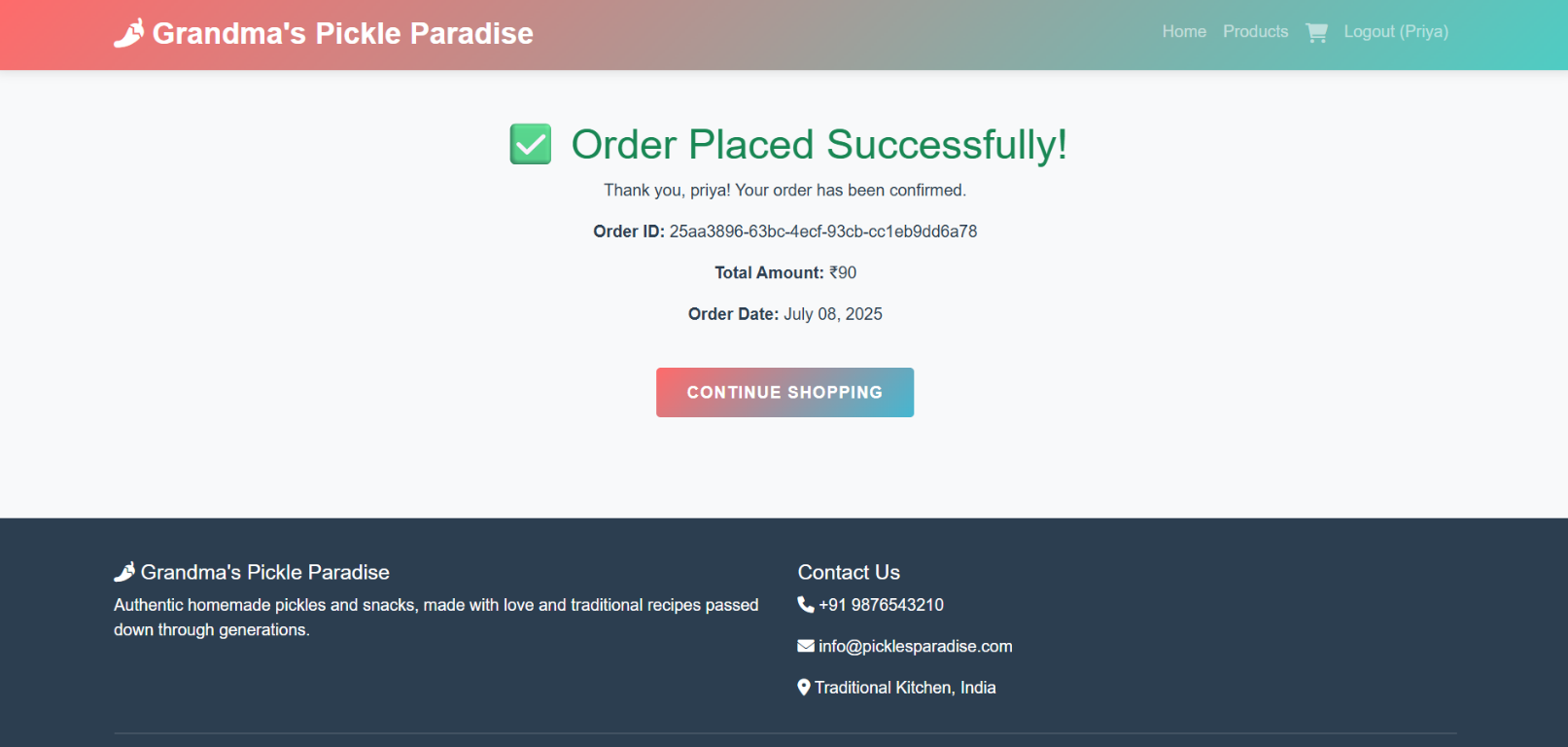




**Place order:**



**Success:**



**Exit:**



**Conclusion:**

**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.