**Home Made Pickles and Snacks**

**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**

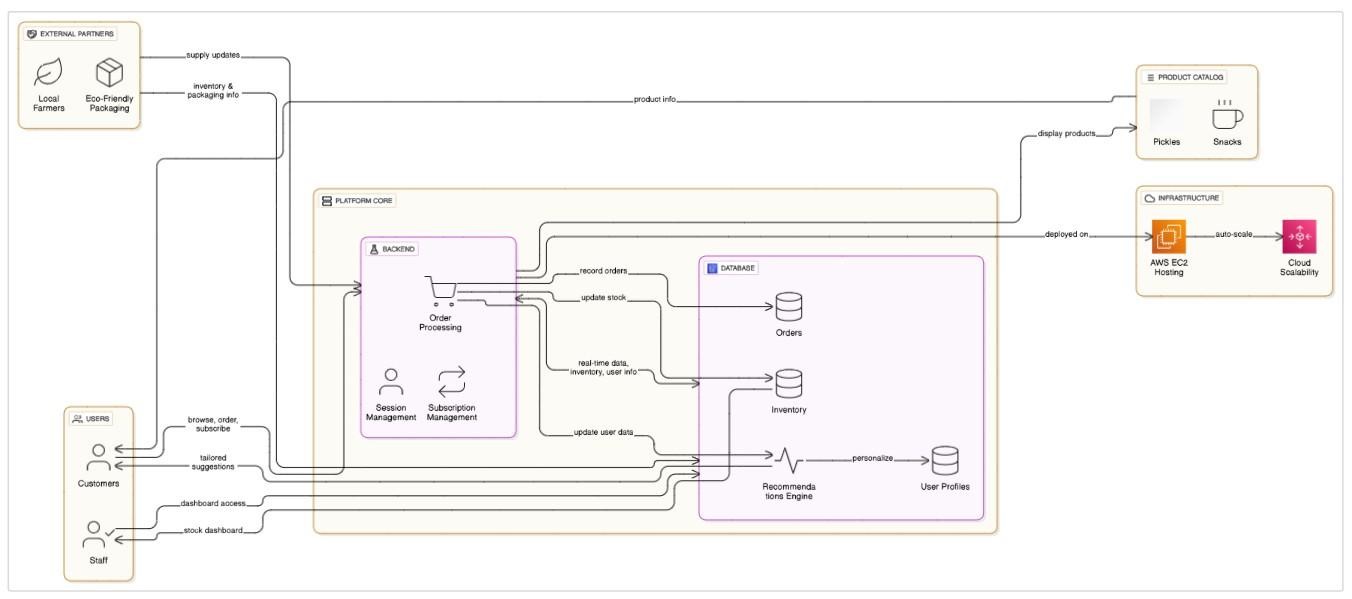
When a customer places an order for a product, the system instantly updates stock levels and records transaction details. For example, a user purchases an item, triggering automatic inventory deduction and order confirmation. Staff members receive updated dashboards to monitor stock availability and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

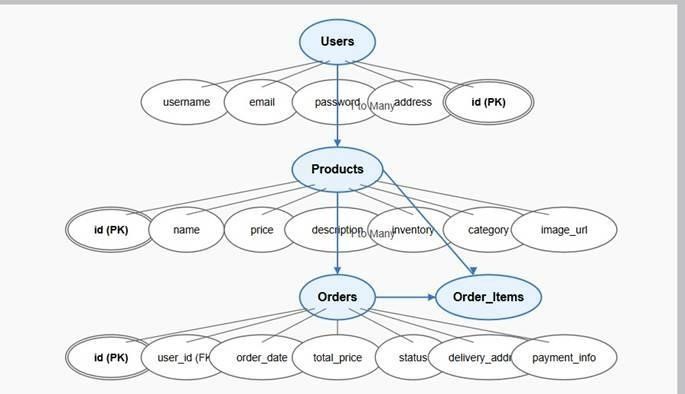
**AWS ARCHITECTURE**

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:**

An ER (Entity-Relationship) diagram visually represents the logical structure of a database by defining entities, their attributes, and the relationships between them. It helps organize data efficiently by illustrating how different components of the system interact and relate. This structured approach supports effective database normalization, data integrity, and simplified query design.



**Pre-requisites**

* AWS Account Setup:

https://docs.aws.amazon.com/accounts/latest/reference/gettingstarted.ht ml

* 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 it is not done yet.
* 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 receive 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.

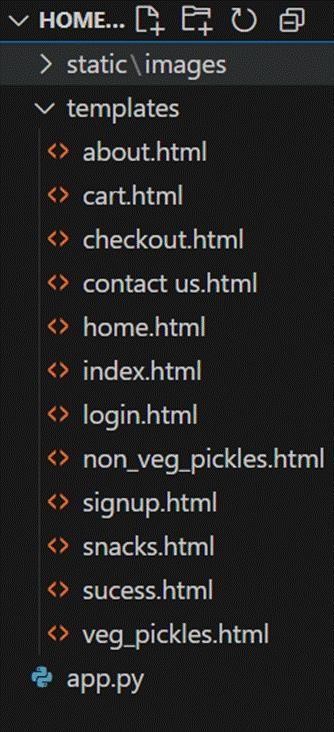
* At this point, modify your app.py and replace local dictionary/list operations with AWS services (such as DynamoDB, RDS, or others as per project requirements).

* 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.

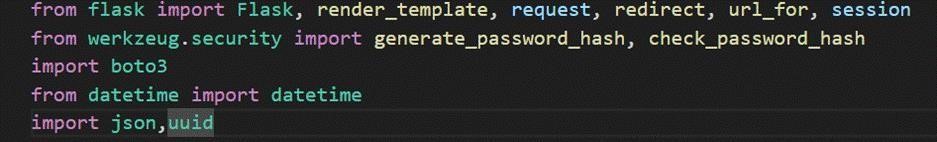
## LOCAL DEPLOYMENT

* File Explorer Structure

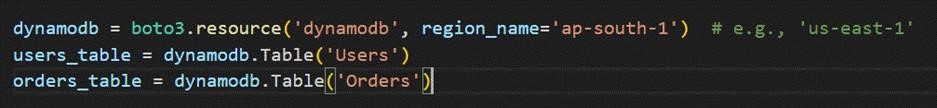


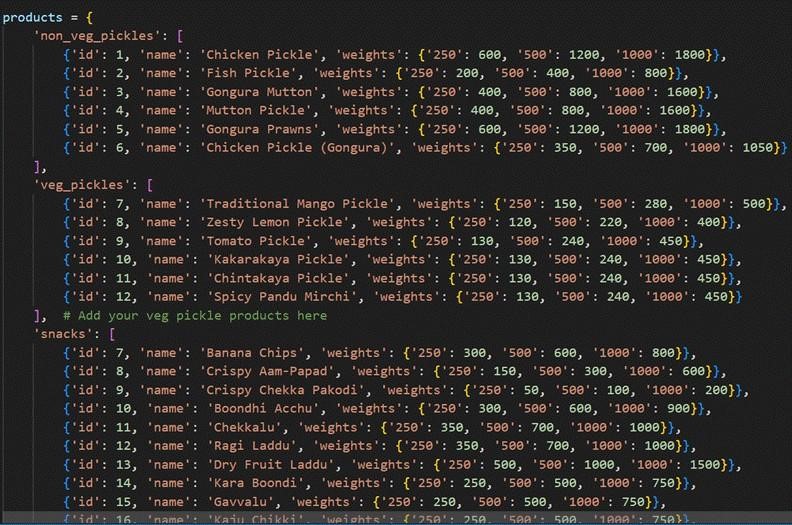
Description of the code:

? Flask App Initialization

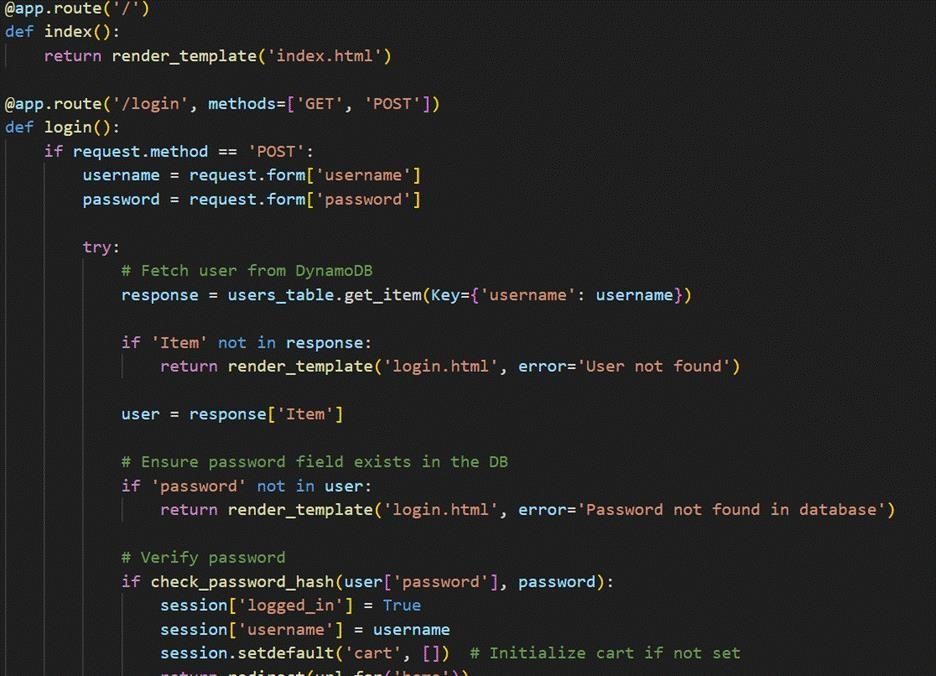


* Use boto3 to connect to DynamoDB for handling user registration, Order details database operations and mention region name where DynamoDB tables are created.

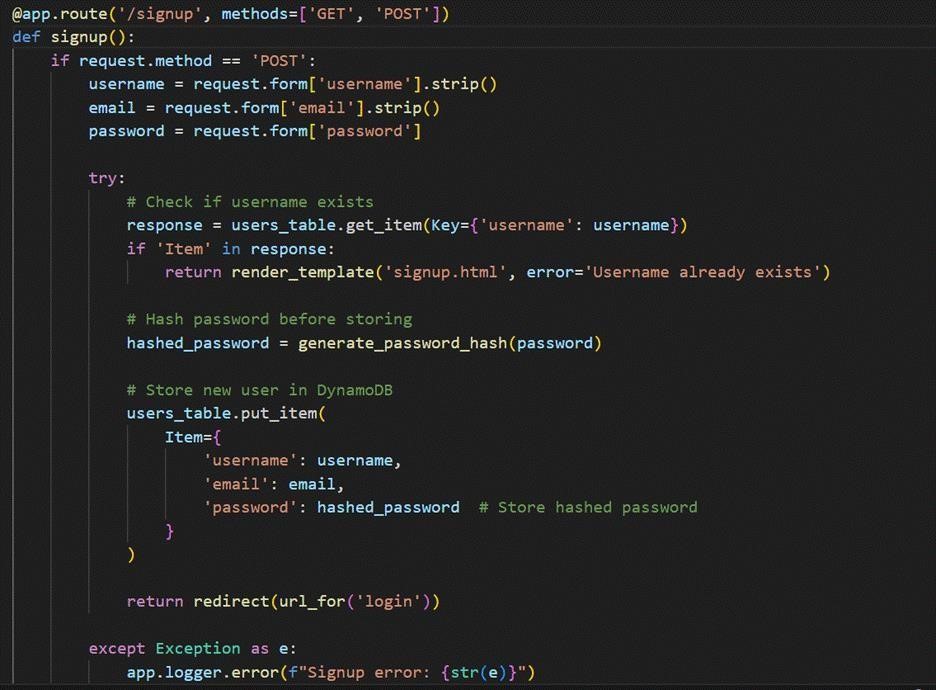


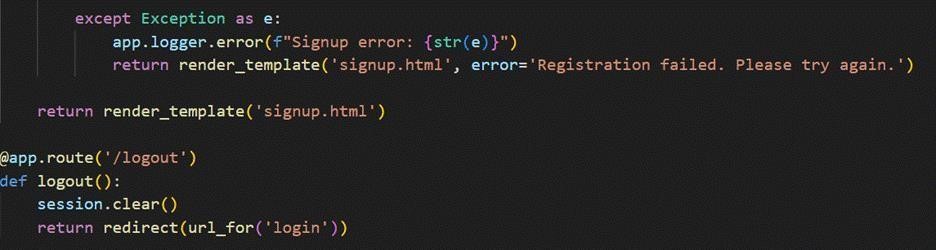


* Routes for Web Pages
* Login Route (GET/POST): Verifies user credentials, increments login count, and redirects to the dashboard on success.



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

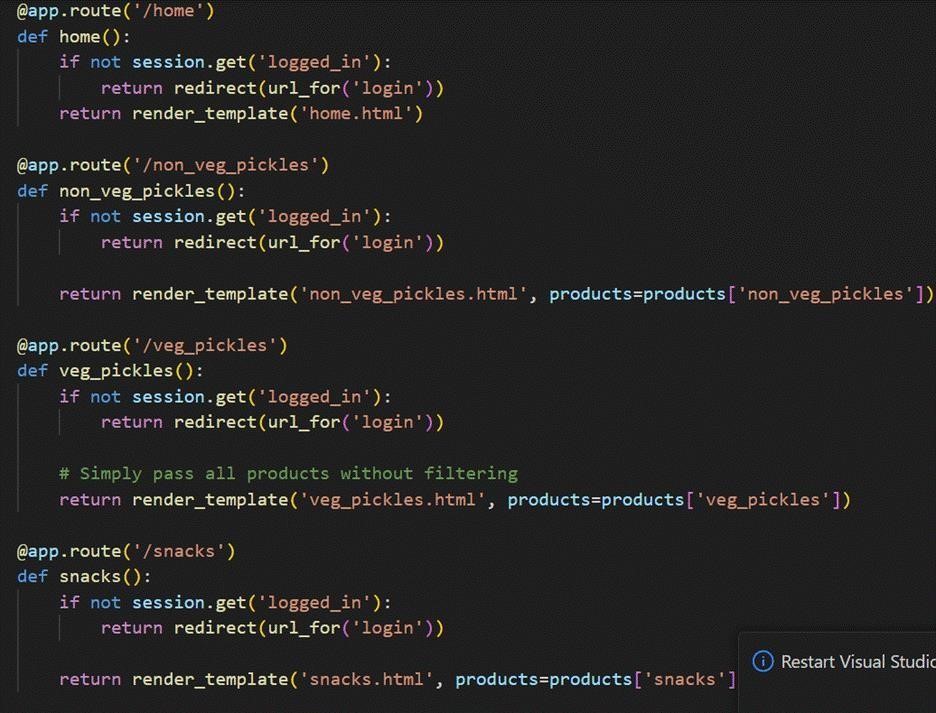




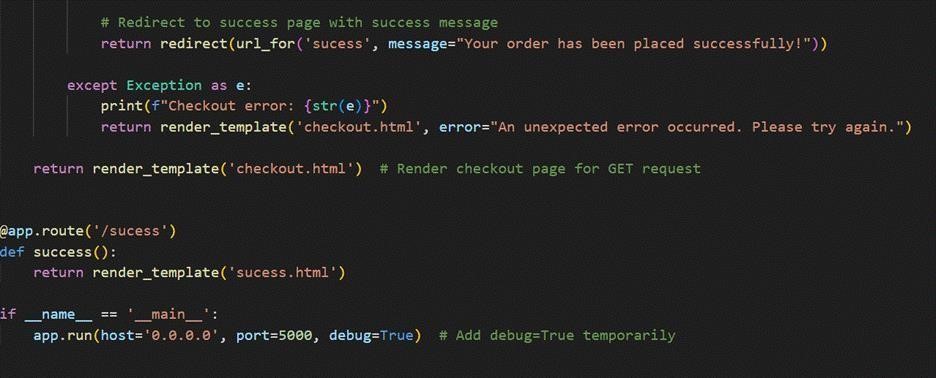
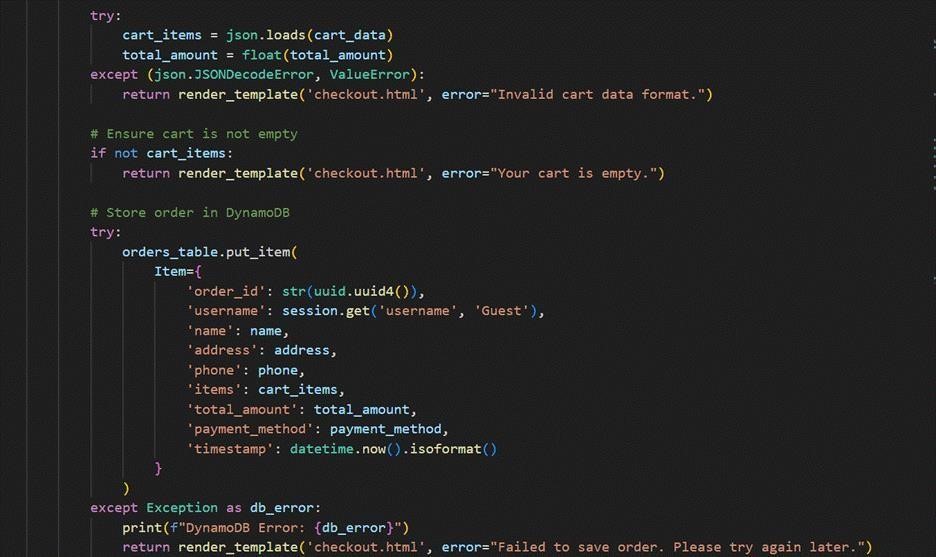
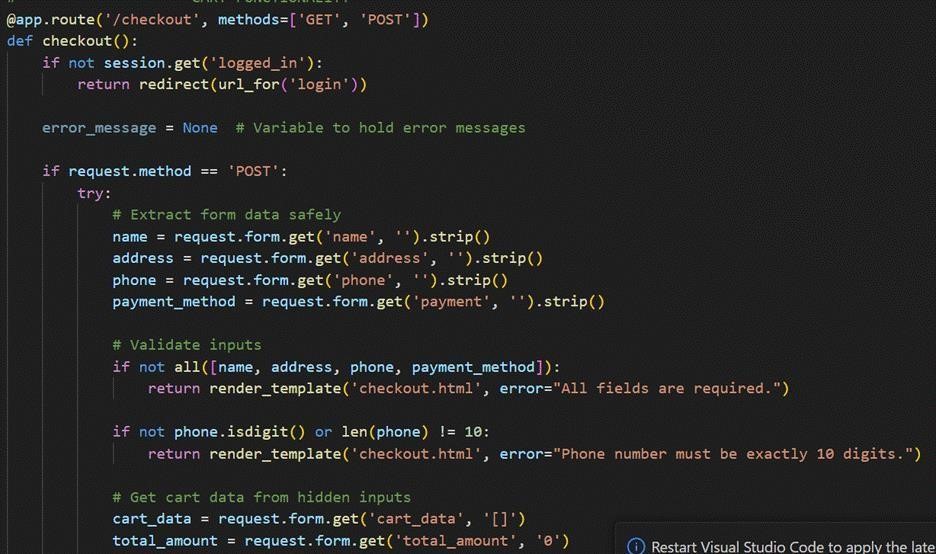
* Logout route: The user can Logout so that the user can get back to the Login Page



* Home Route: Home page contains the routing for different categories which are Veg\_pickles, Non\_Veg\_pickles,Snacks.



* Check out Route:



## 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.

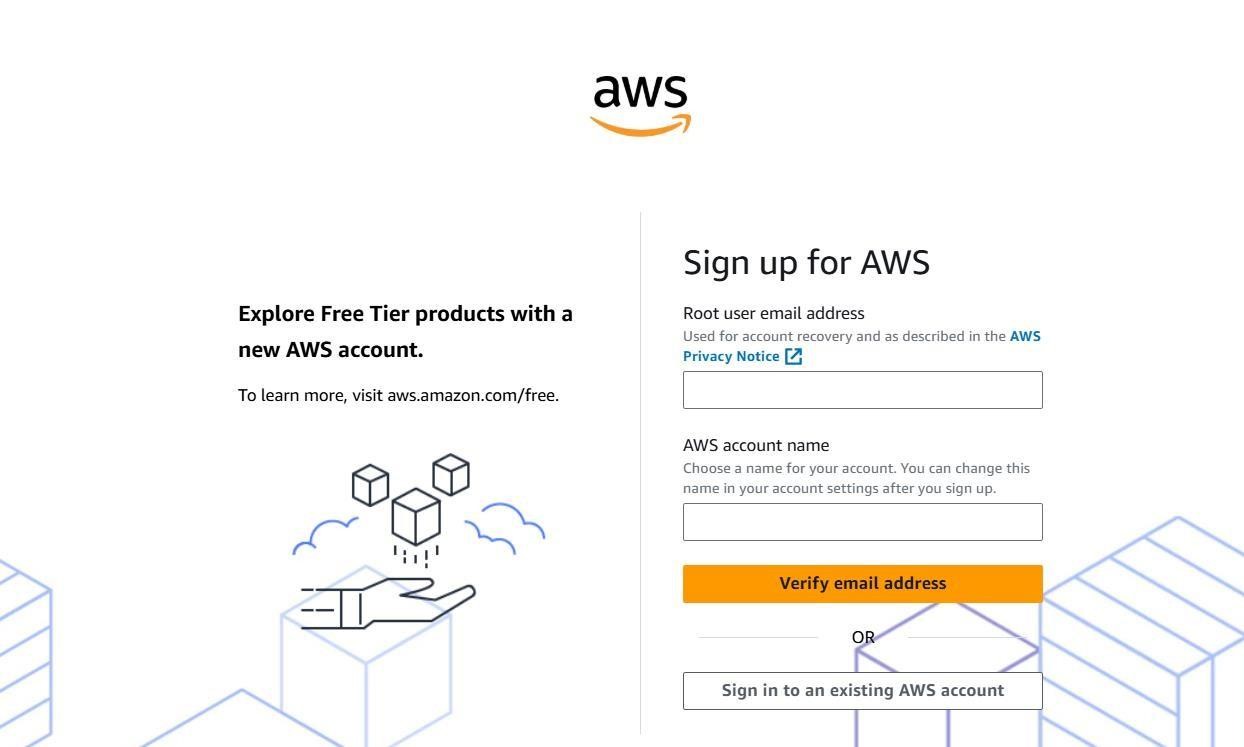
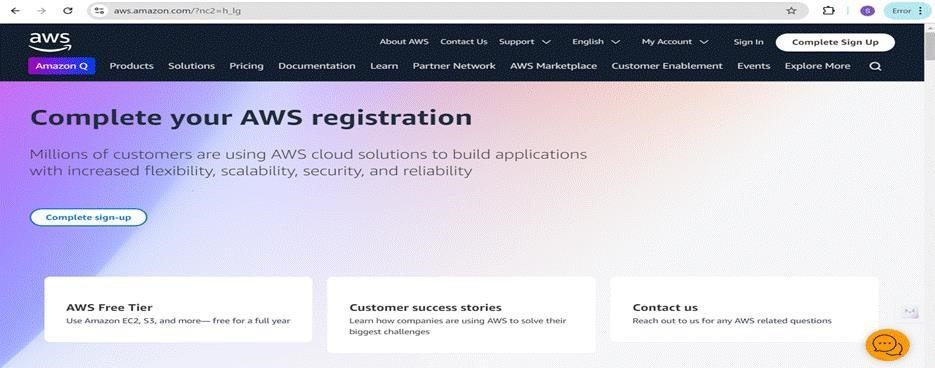
**Reminder: You must complete the Web Development task before gaining access to Troven. Once accessed, the AWS Console via Troven is available for only 3 hours—please plan your work accordingly.**

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

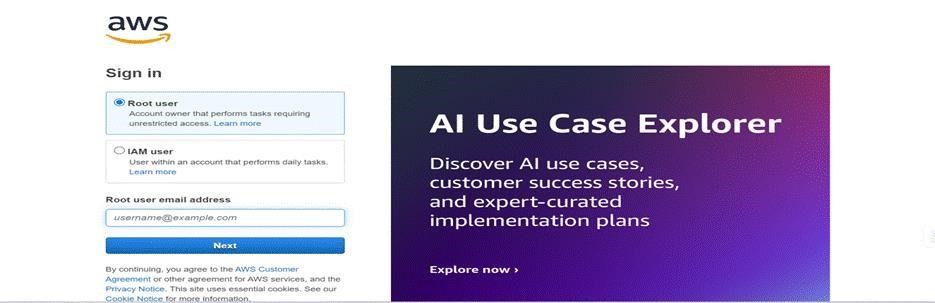
### AWS Account Setup and Login

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

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



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

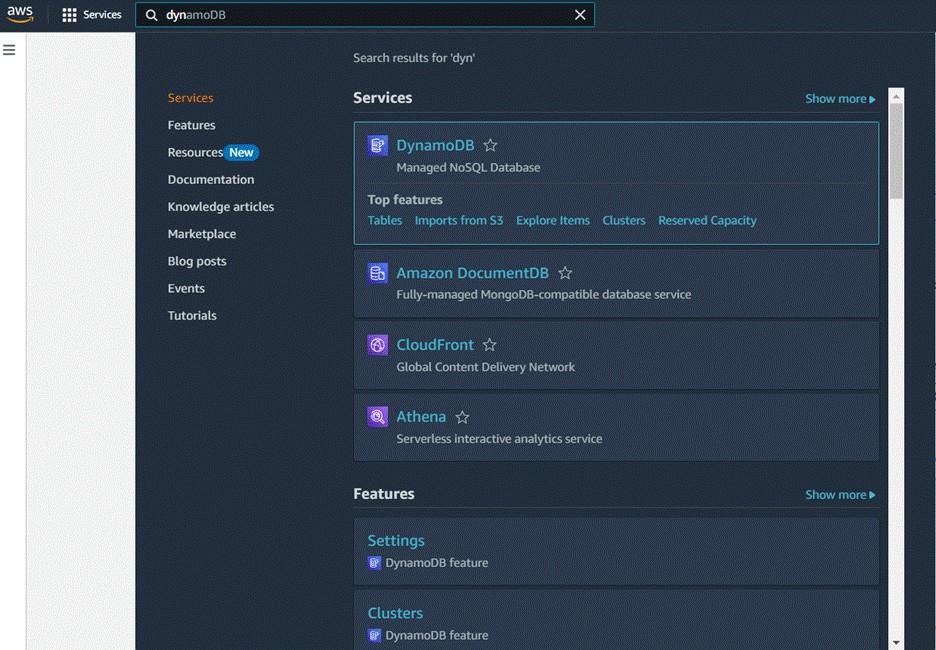


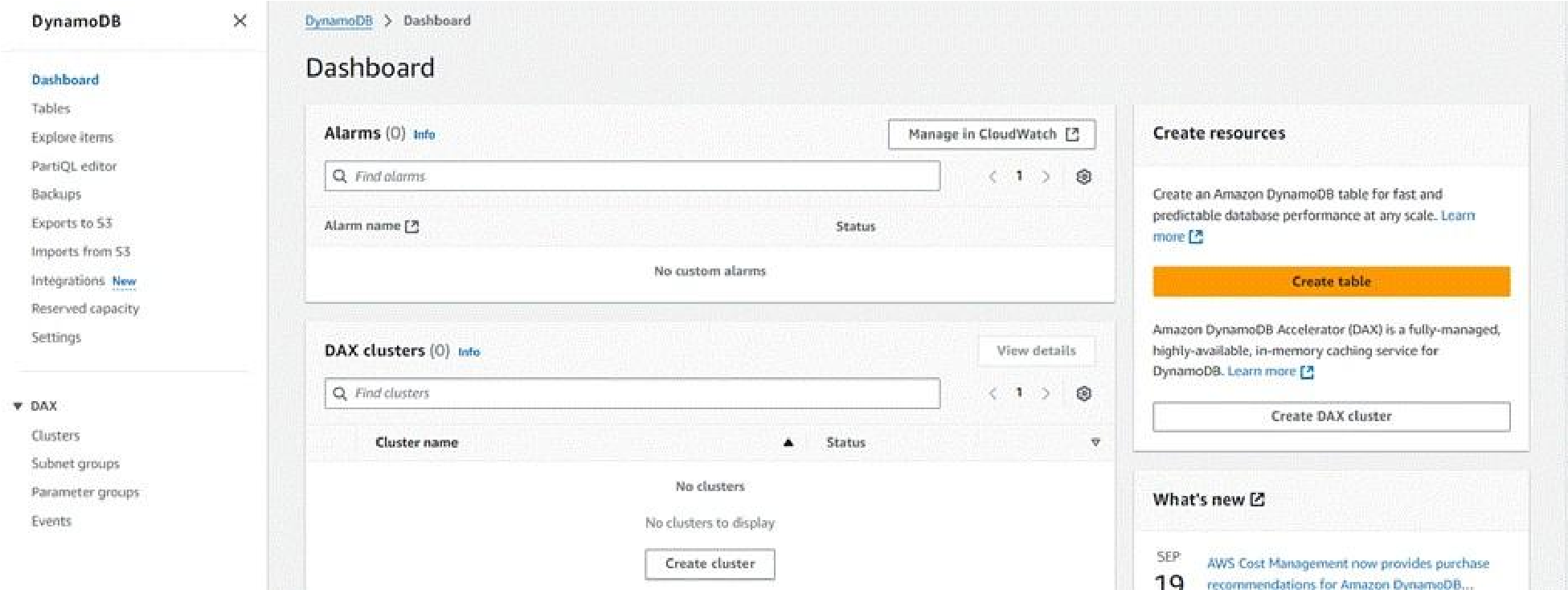
## Milestone 3: DynamoDB Database Creation and Setup

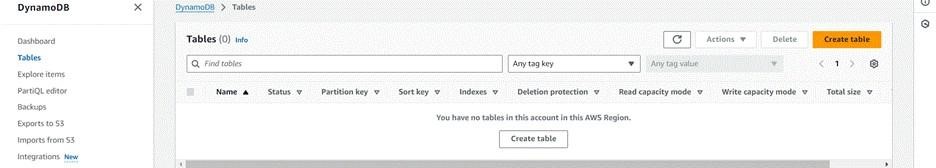
Database Creation and Setup involves initializing a cloud-based NoSQL database to store and manage application data efficiently. This step includes defining tables, setting primary keys, and configuring read/write capacities. It ensures scalable, high-performance data storage for seamless backend operations.

### Navigate to the DynamoDB

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

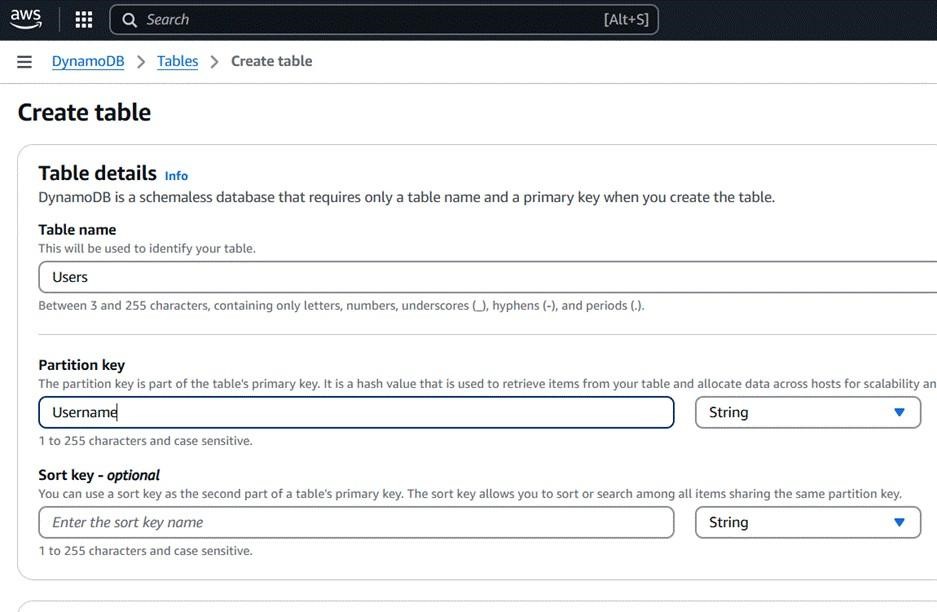


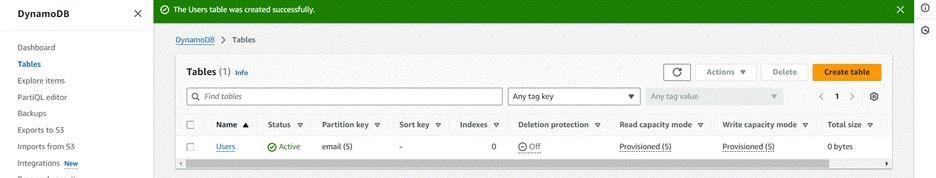
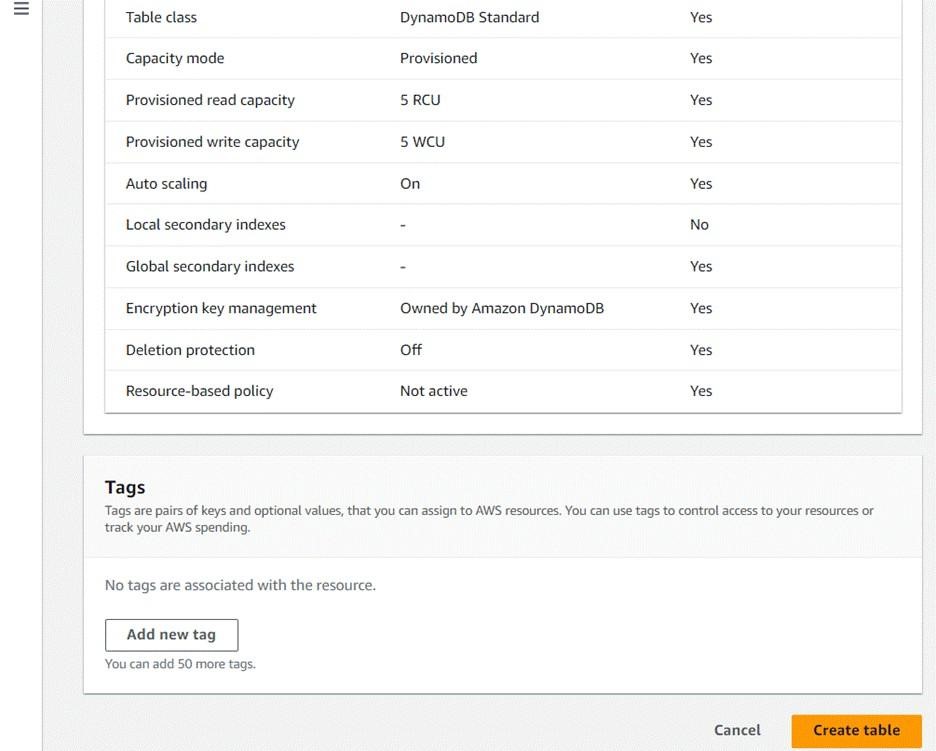




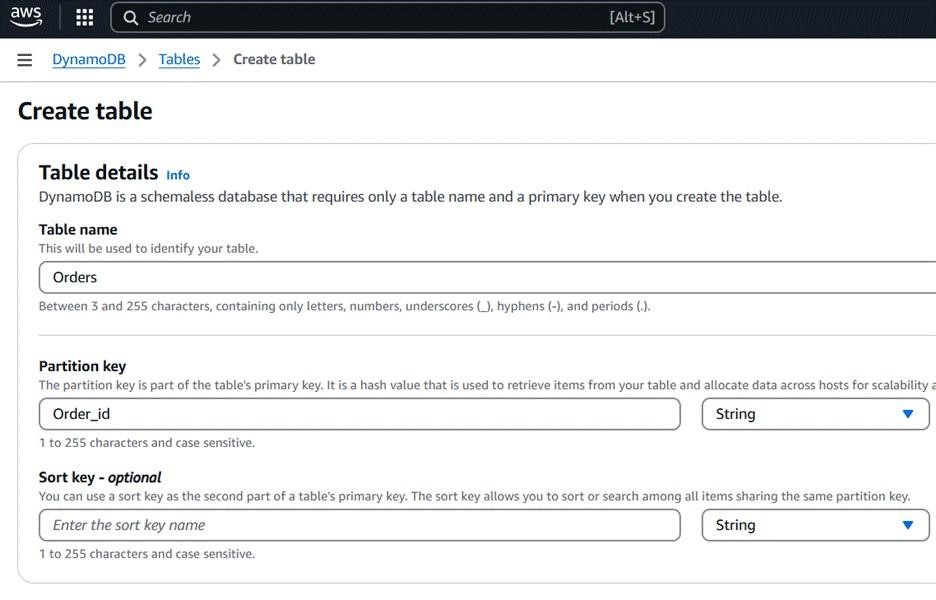
### Create a DynamoDB table for storing data

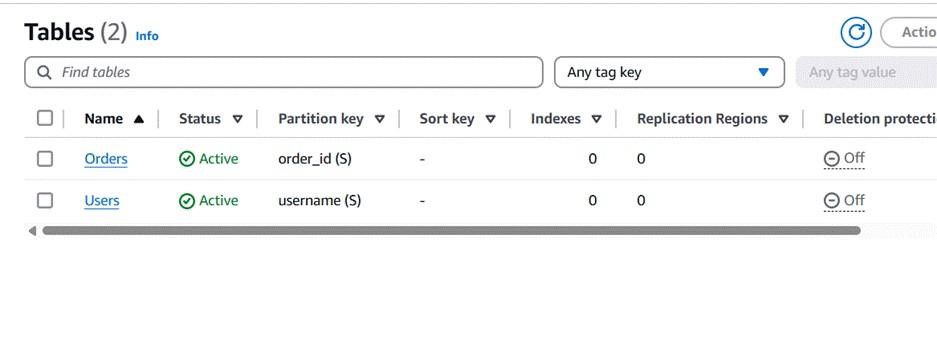
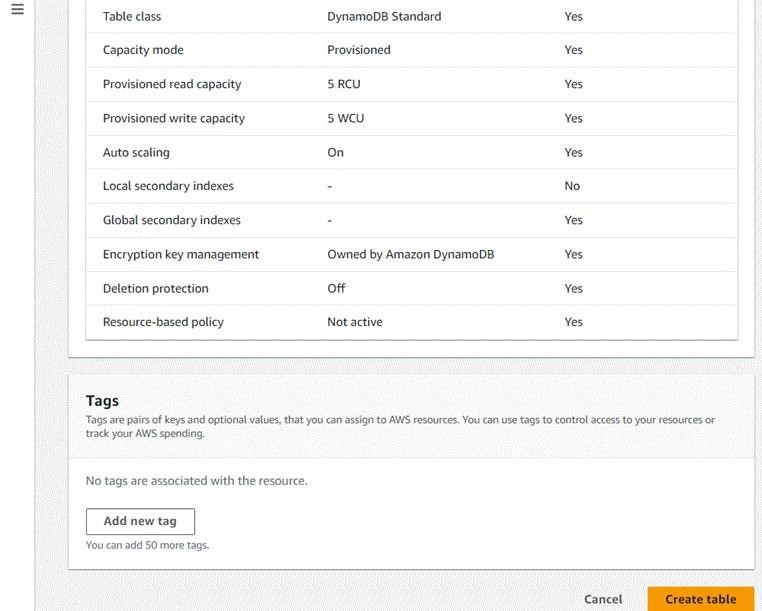
* Create Users table with partition key “Username” with type String and click on create tables.





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



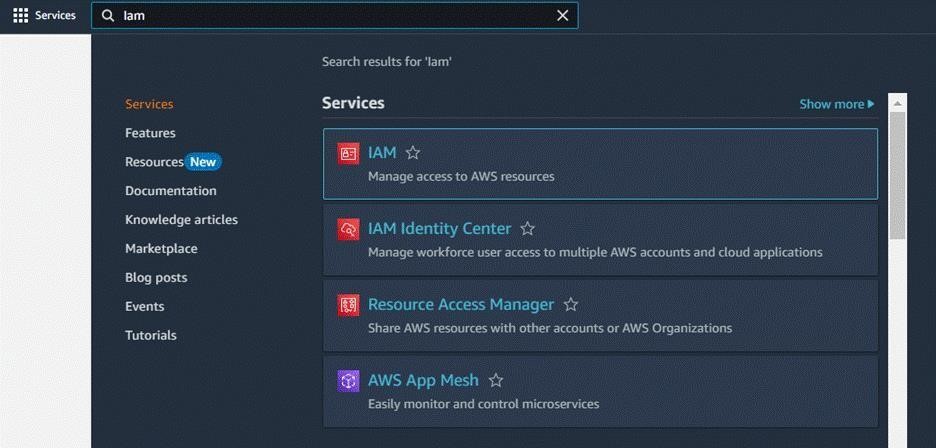


## Milestone 4: IAM Role Setup

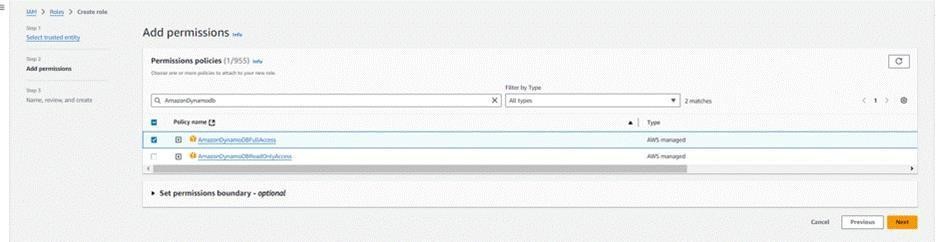
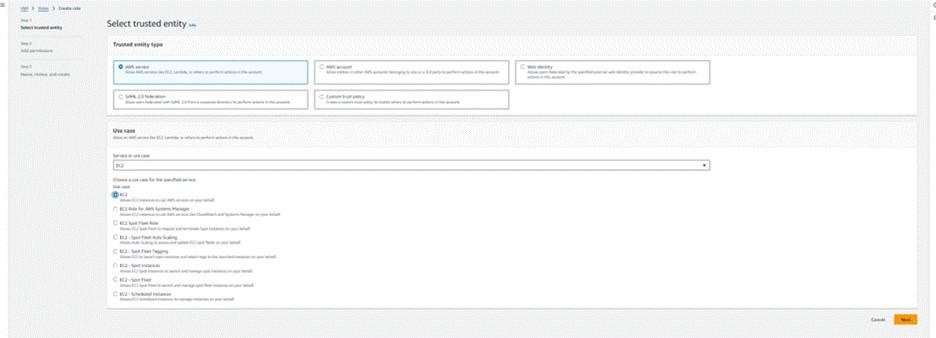
The IAM (Identity and Access Management) role setup involves creating roles that define specific permissions for AWS services. To set it up, you create a role with the required policies, assign it to users or services, and ensure the role has appropriate access to resources like EC2, S3, or RDS. This allows controlled access and ensures security best practices in managing AWS resources.

**Create an IAM Role.**

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



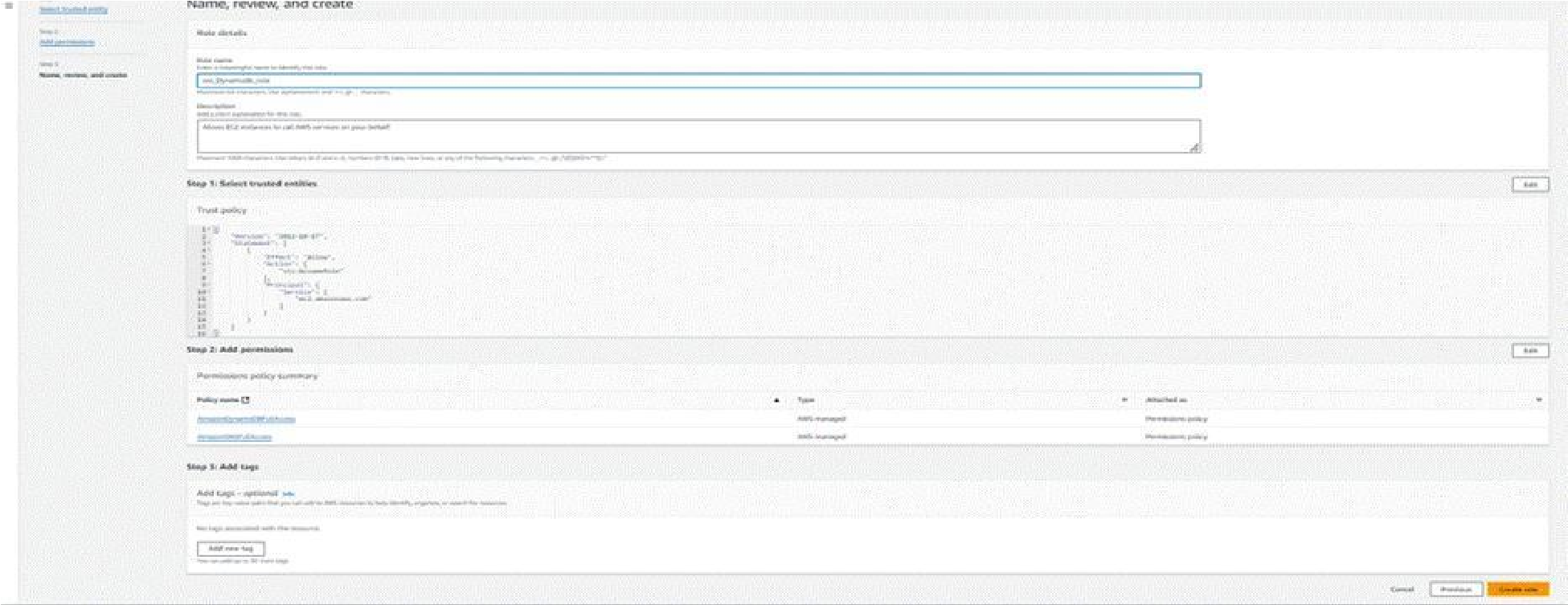




### Attach Policies

Attach the following policies to the role:

• AmazonDynamoDBFullAccess: Allows EC2 to perform read/write operations on DynamoDB.

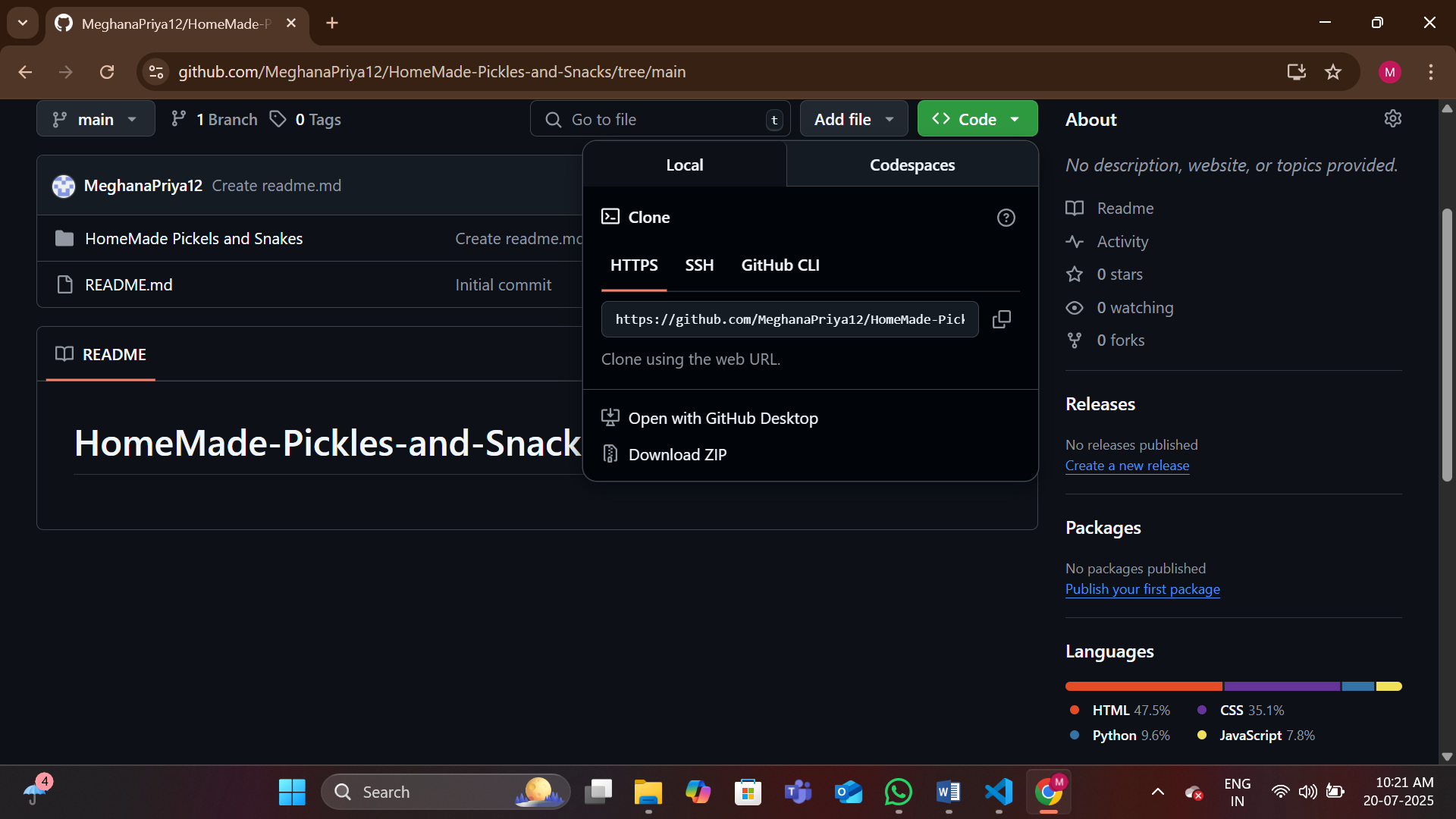




### Milestone 5: EC2 Instance Setup

To set up a public EC2 instance, choose an appropriate Amazon Machine Image (AMI) and instance type. Ensure the security group allows inbound traffic on necessary ports (e.g., HTTP/HTTPS for web applications). After launching the instance, associate it with an Elastic IP for consistent public access, and configure your application or services to be publicly accessible.

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

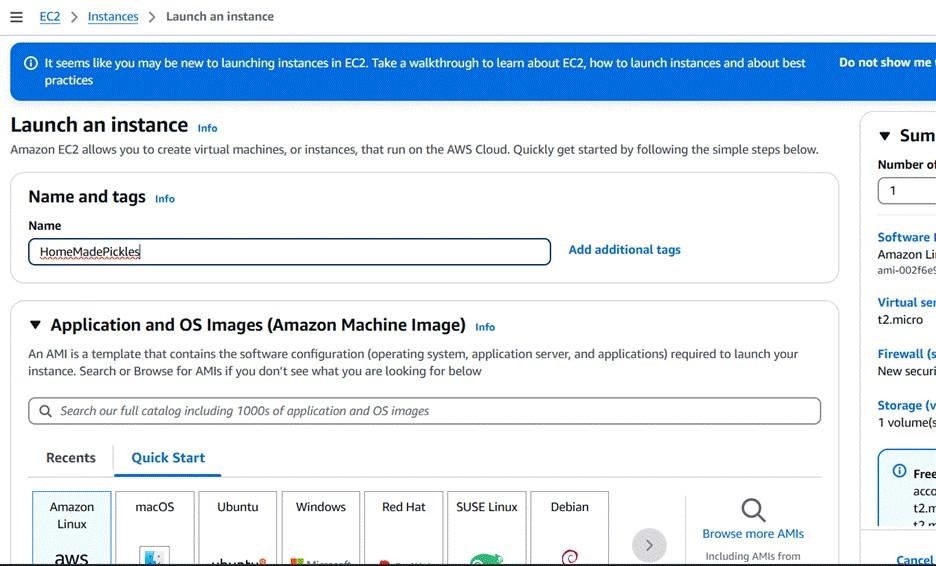
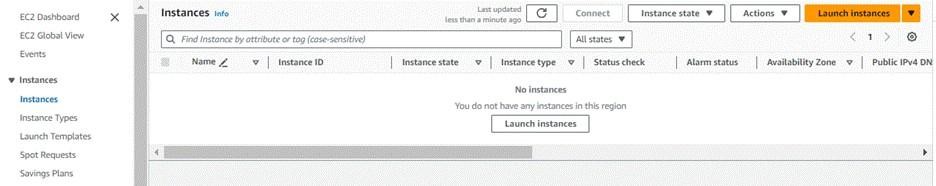


### Launch an EC2 instance to host the Flask

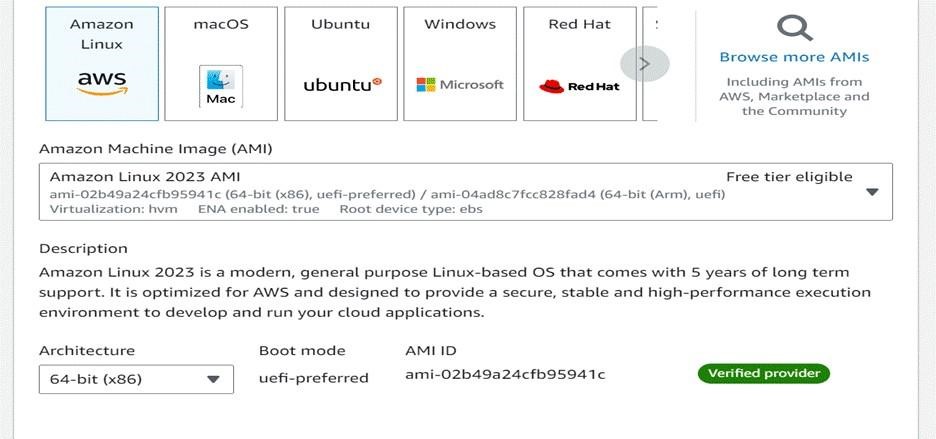
* Launch EC2 Instance
* In the AWS Console, navigate to EC2 and launch a new instance.



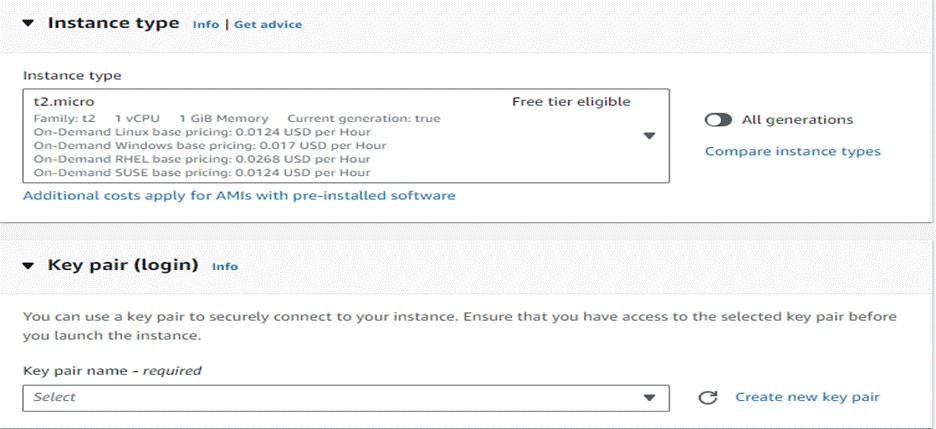
* Click on Launch instance to launch EC2 instance

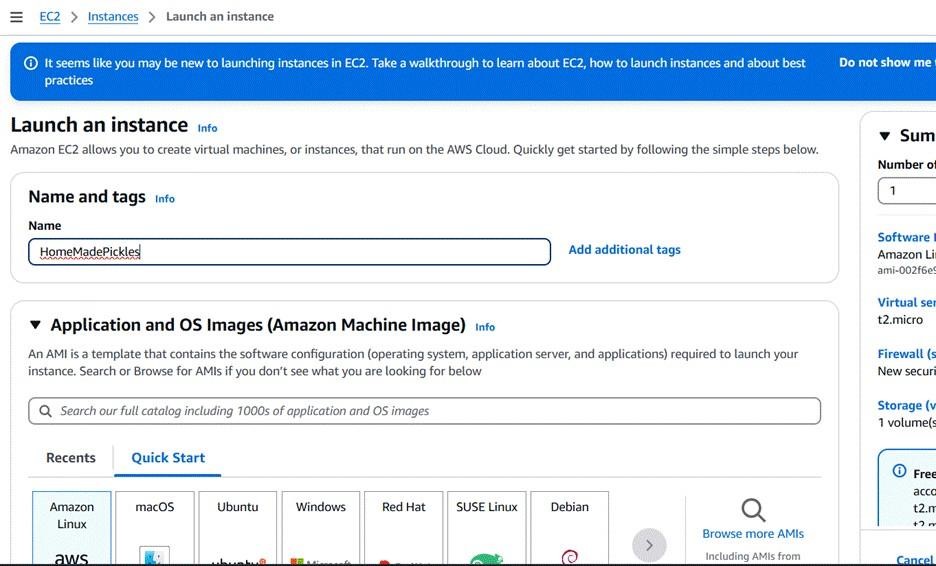


? Choose Amazon Linux 2 or Ubuntu as the AMI and t2. micro as the instance type (free-tier eligible).

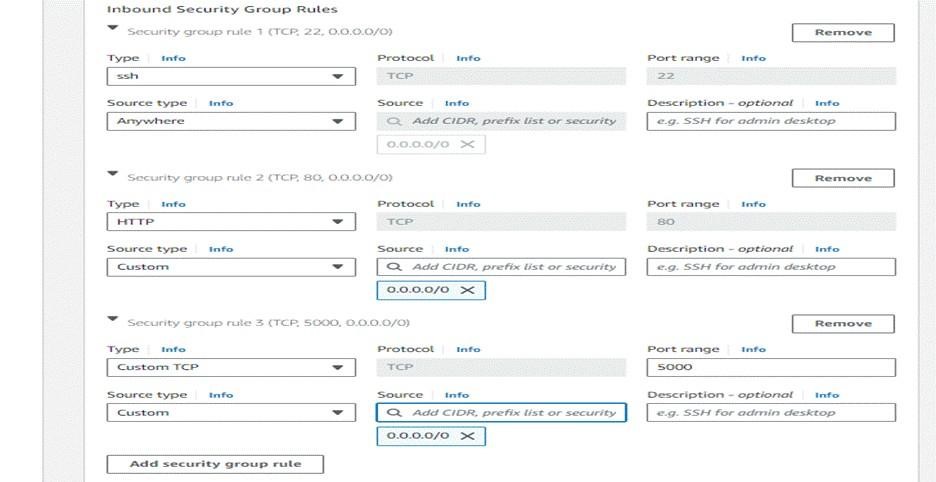
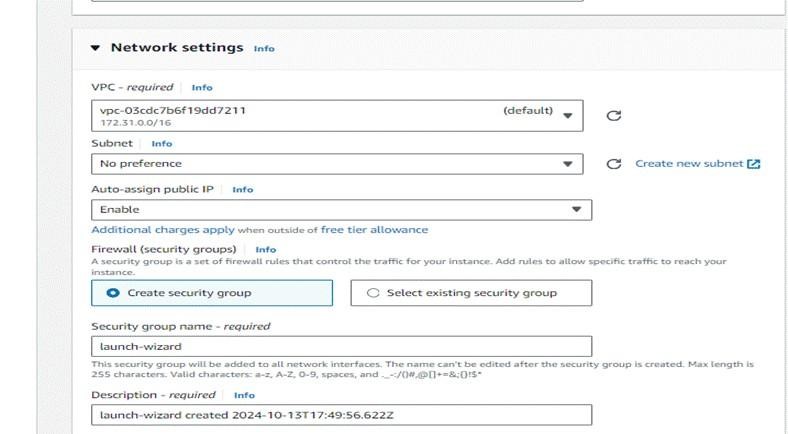


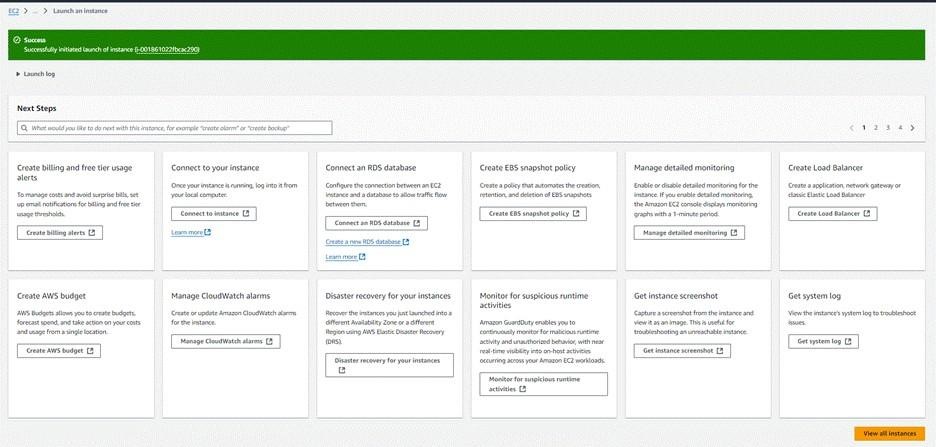
* Create and download the key pair for Server access.





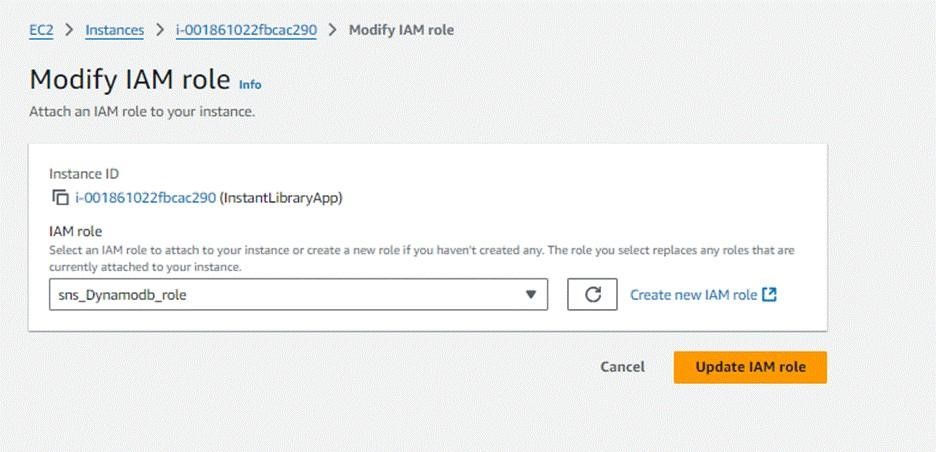
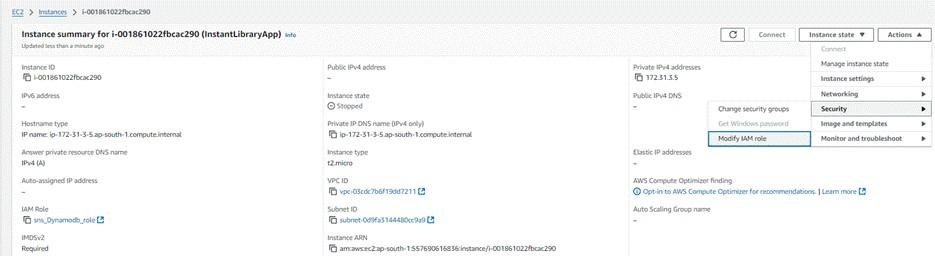
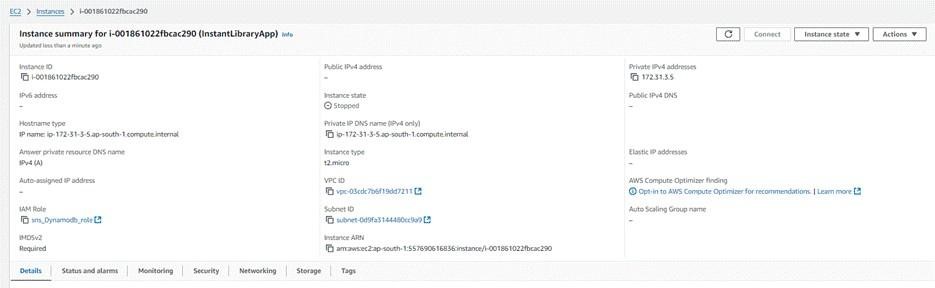
**Configure security groups for HTTP and SSH access.**



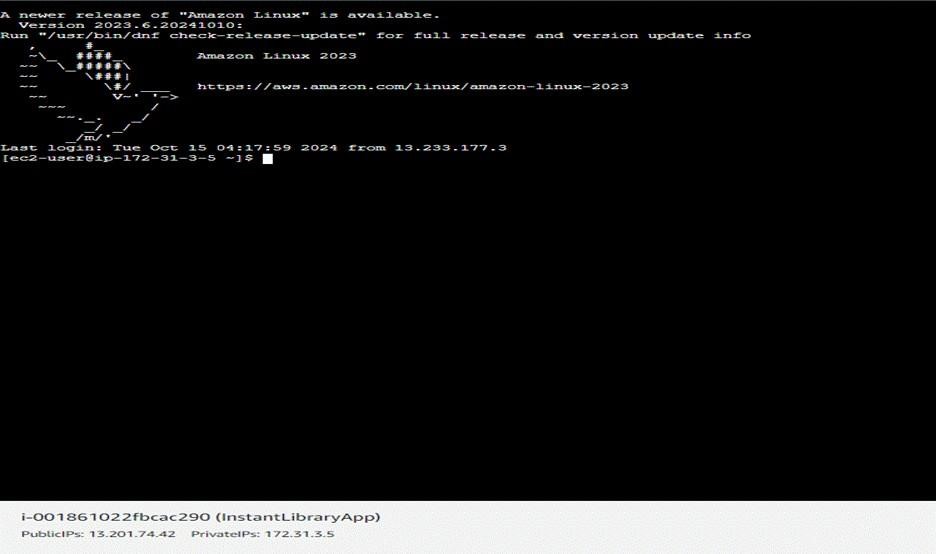
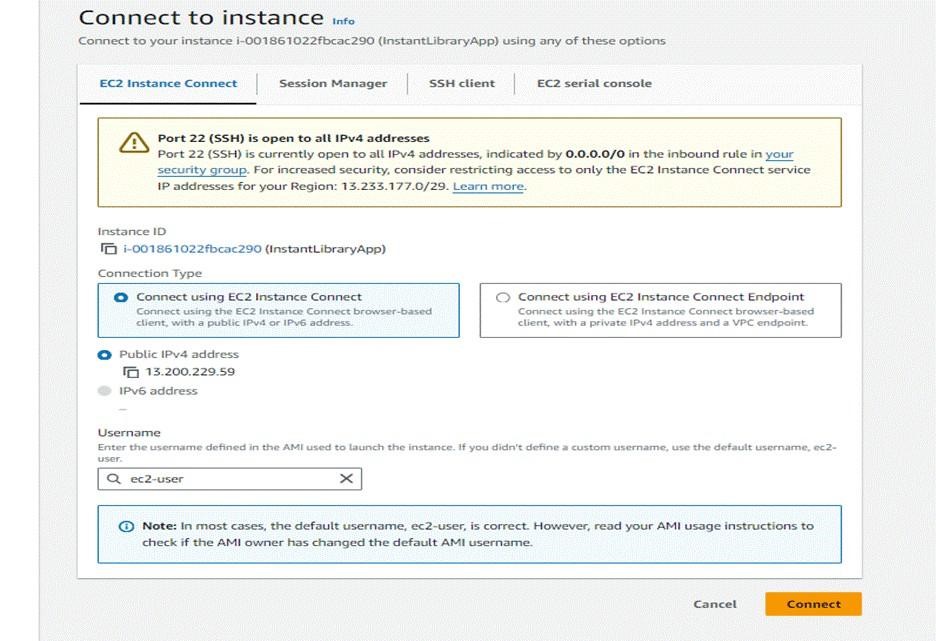


* To connect to EC2 using EC2 Instance Connect, start by ensuring that an IAM role is attached to your EC2 instance. You can do this by selecting your instance, clicking on Actions, then navigating to Security and selecting Modify IAM Role to attach the appropriate role. After the IAM role is connected, 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 6: Deployment on EC2

Deployment on an EC2 instance involves launching a server, configuring security groups for public access, and uploading your application files. After setting up necessary dependencies and environment variables, start your application and ensure it's running on the correct port. Finally, bind your domain or use the public IP to make the application accessible online.

**Install Software on the EC2 Instance**

Install Python3, Flask, and Git:

**On Amazon Linux 2:**

* sudo yum update -y
* sudo yum install python3 git
* sudo pip3 install flask boto3

**Verify Installations:**

* flask --version
* git –version

### Clone Your Flask Project from GitHub

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

* Run: ‘https://github.com/MeghanaPriya12/HomeMade-Pickles-and-Snacks.git

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

credentials

here:‘ <https://github.com/MeghanaPriya12/HomeMade-Pickles-and-Snacks.git>

This will download your project to the EC2 instance.

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

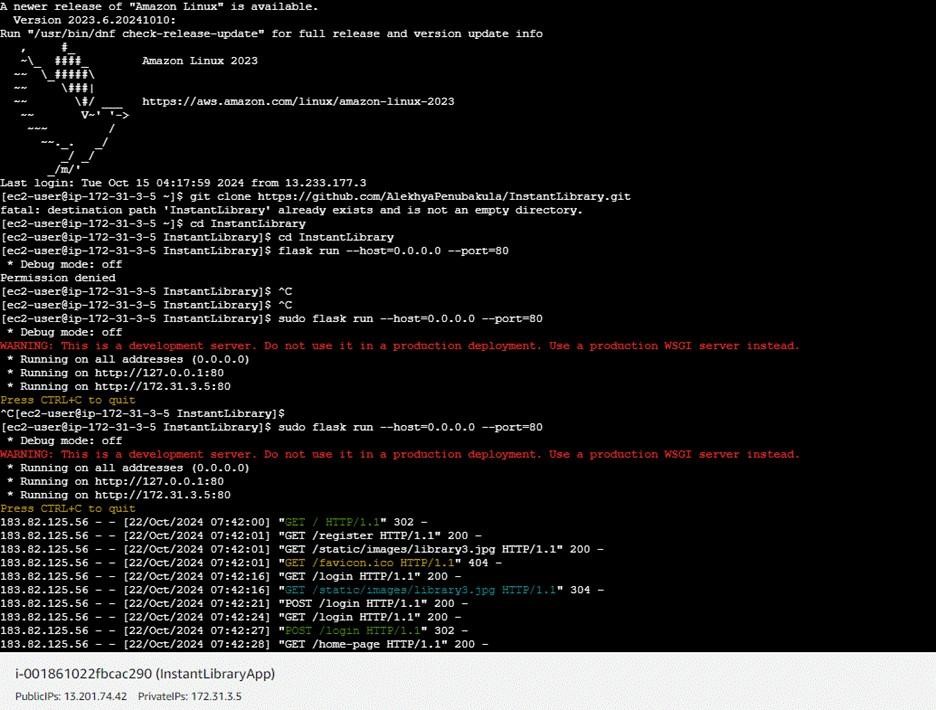
* cd Homemade pickles and snacksss
* cd “Home Made Pickles1”

Create a Virtual Environment:

* python3 -m venv venv
* source venv/bin/activate
* sudo yum install python3 git
* sudo pip3 install flask boto3

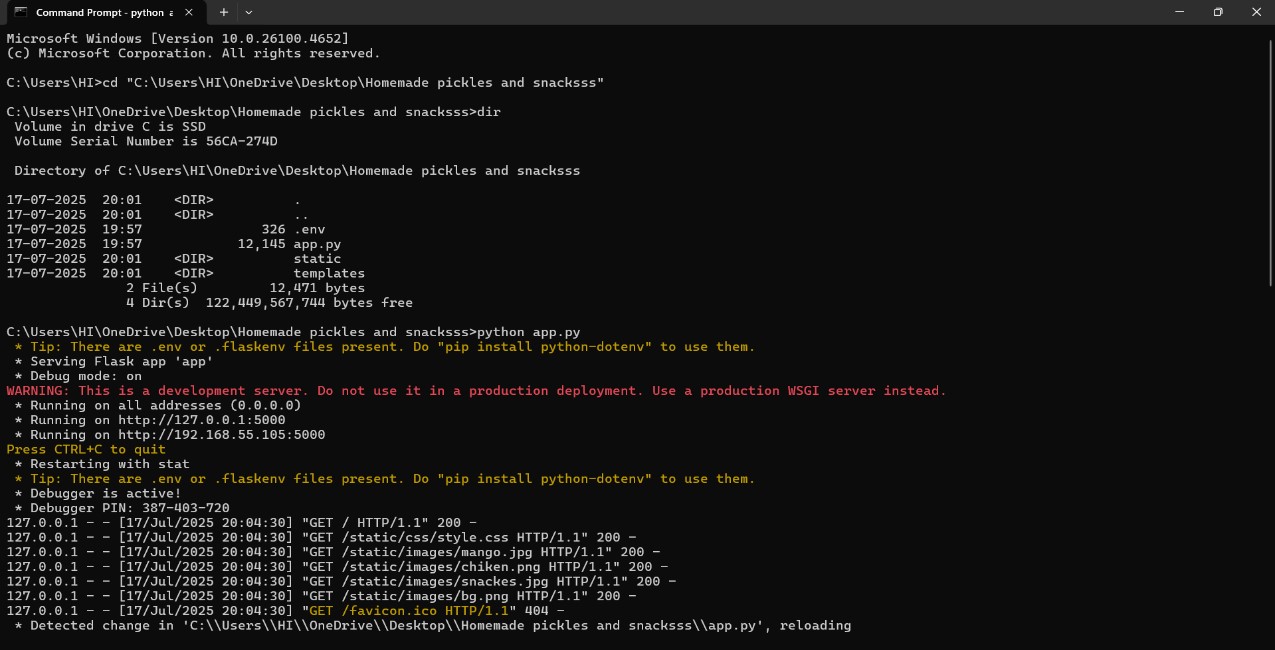
Once inside the project directory, configure and run the Flask application by executing the following command with elevated privileges:

* Run the Flask Application
* sudo flask run --host=0.0.0.0 --port=5000



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

* Run the Flask app on the EC2 instance



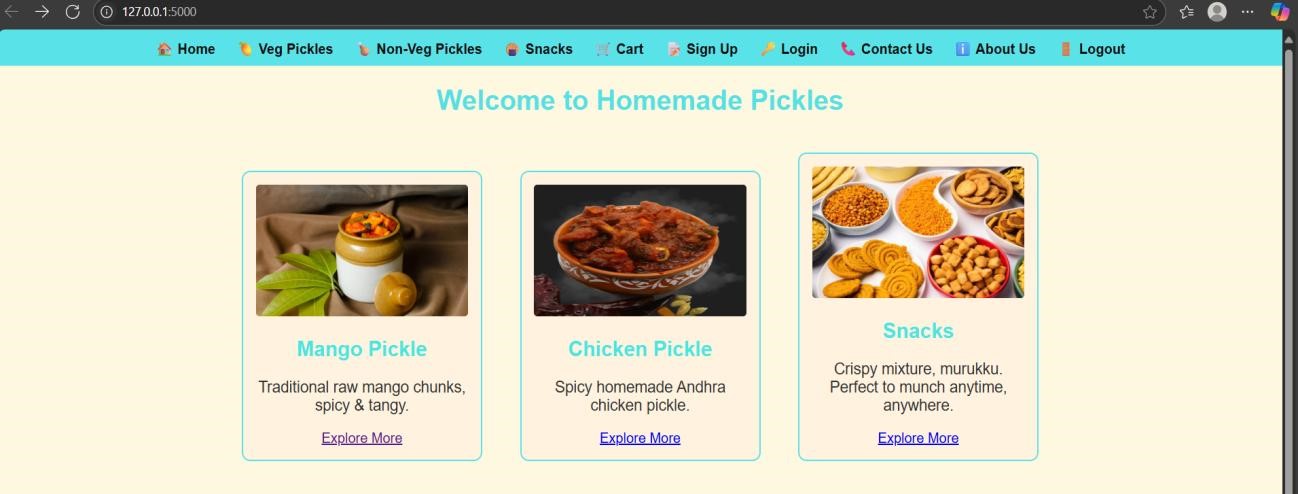
Access the website through:

PublicIPs: [http://127.0.0.1:5000](http://127.0.0.1:5000/)

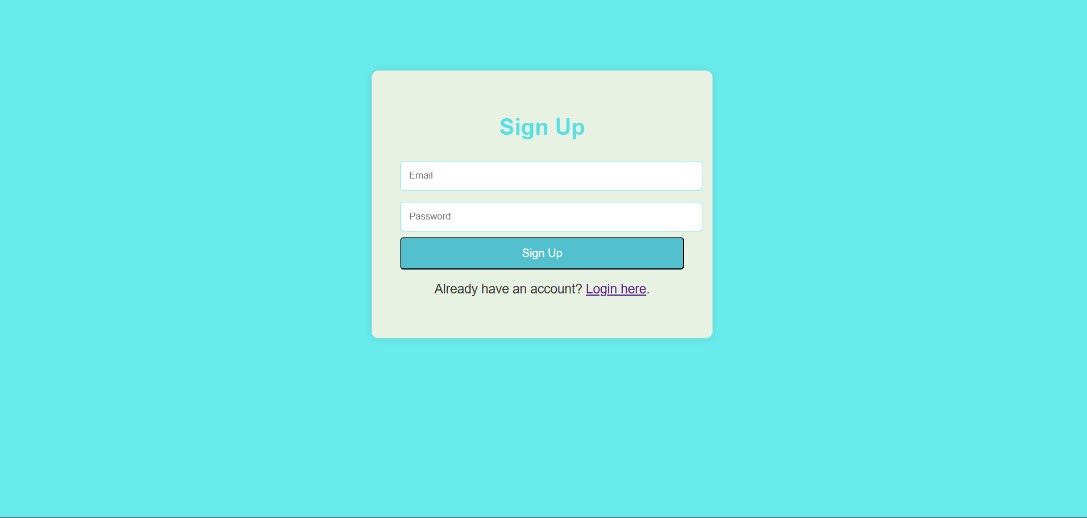
### Milestone 7: Testing and Deployment

Testing and deployment involve verifying that your application works as expected before making it publicly accessible. Start by testing locally or in a staging environment to catch bugs and ensure functionality.Once tested, deploy the application to an EC2 instance, configure necessary services, and perform a final round of live testing to confirm everything runs smoothly in the production environment.

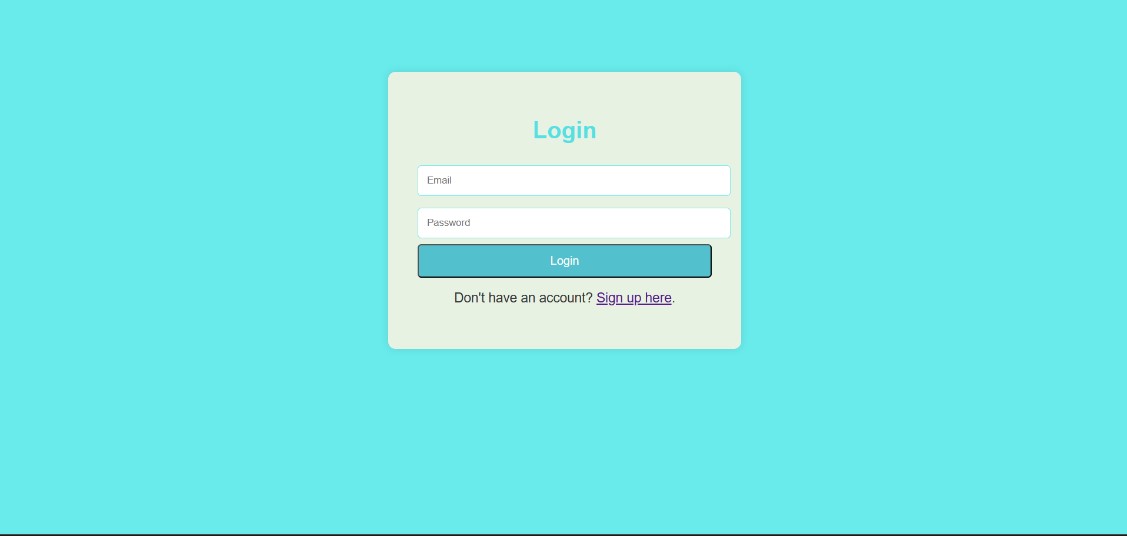
**Functional testing to verify the Project** Welcome page:



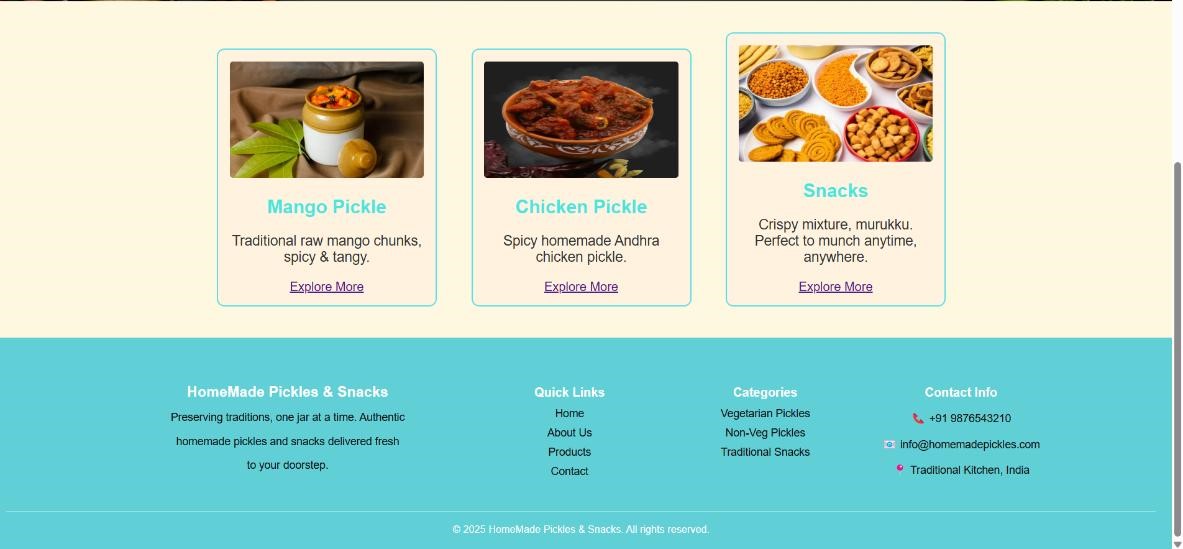
Signup page :



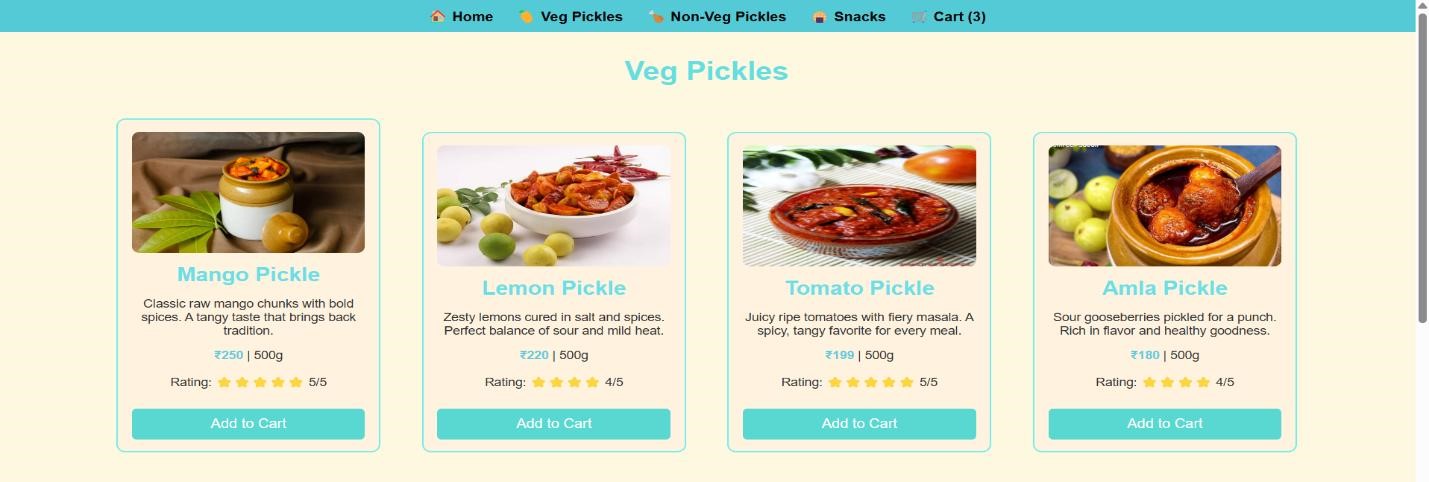
Login Page:

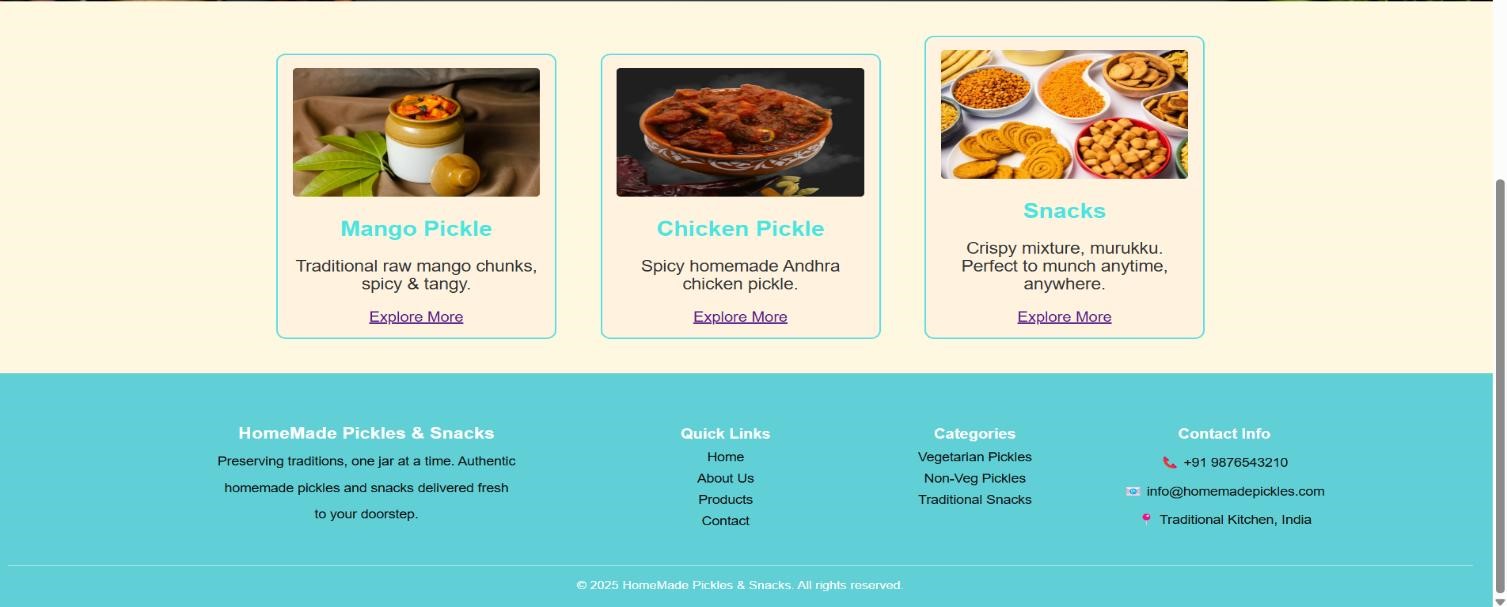


Home Page:

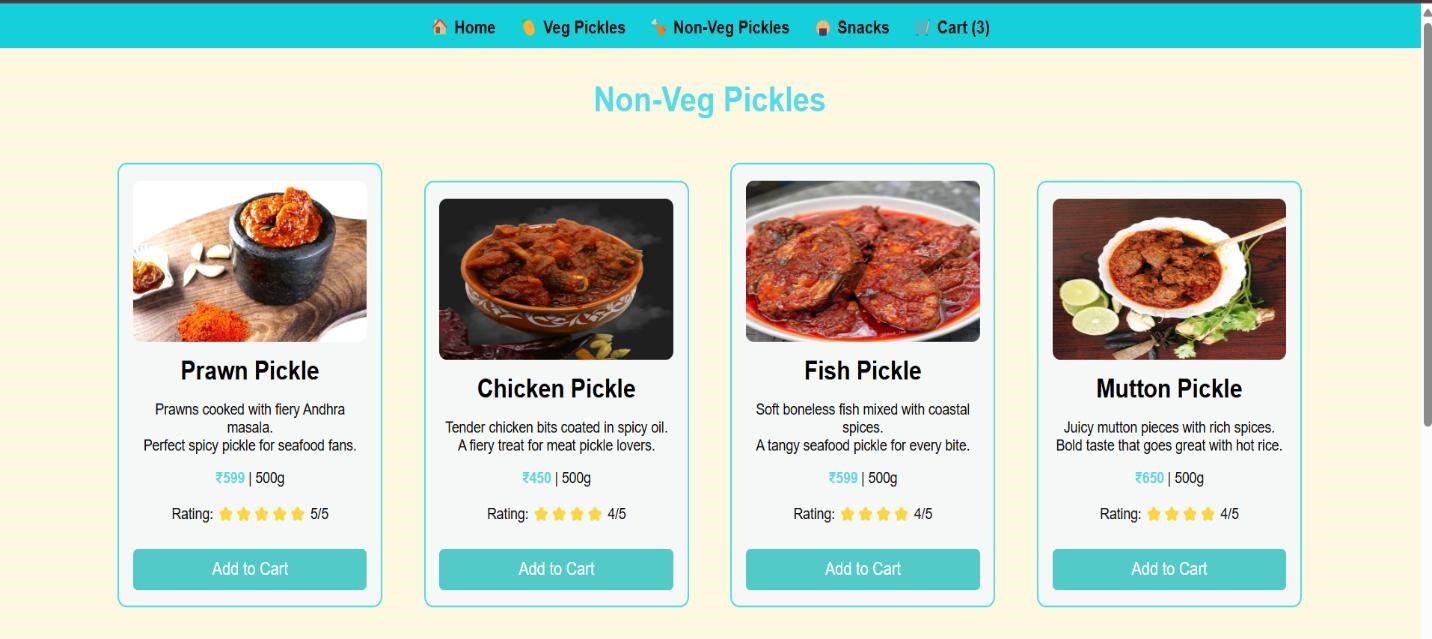


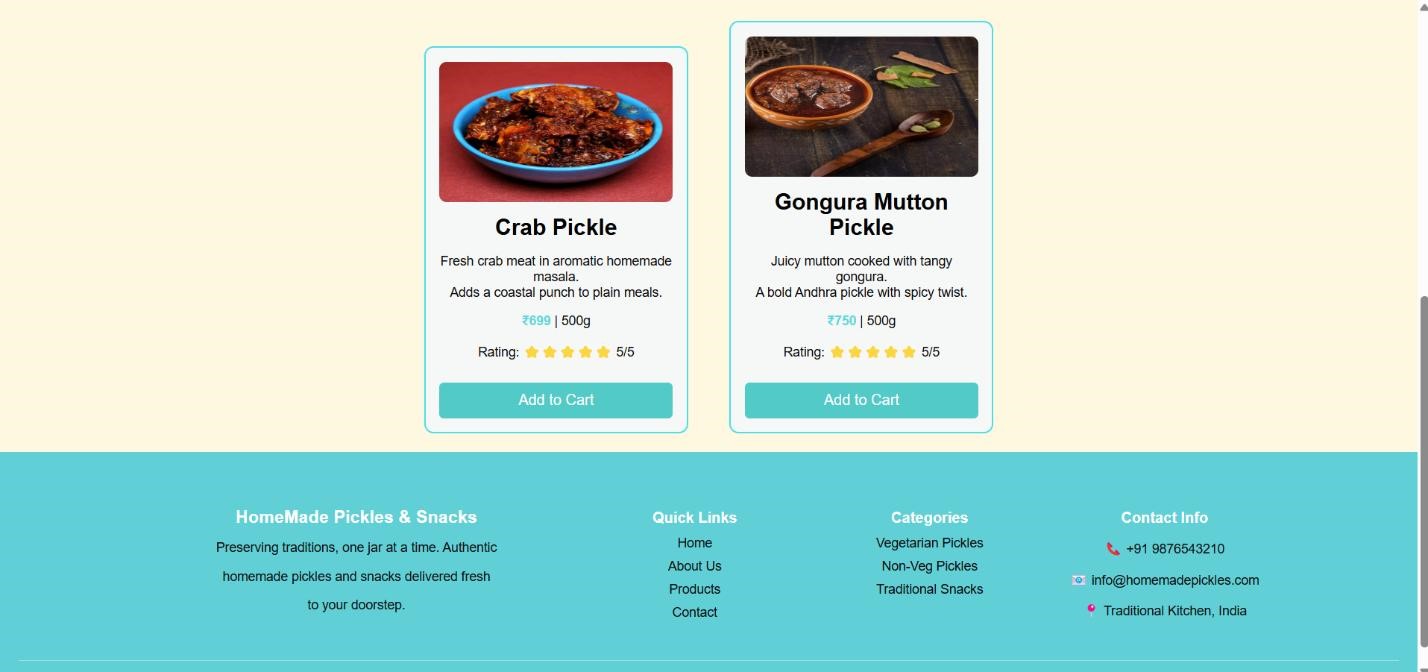
Veg\_Pickles Page:



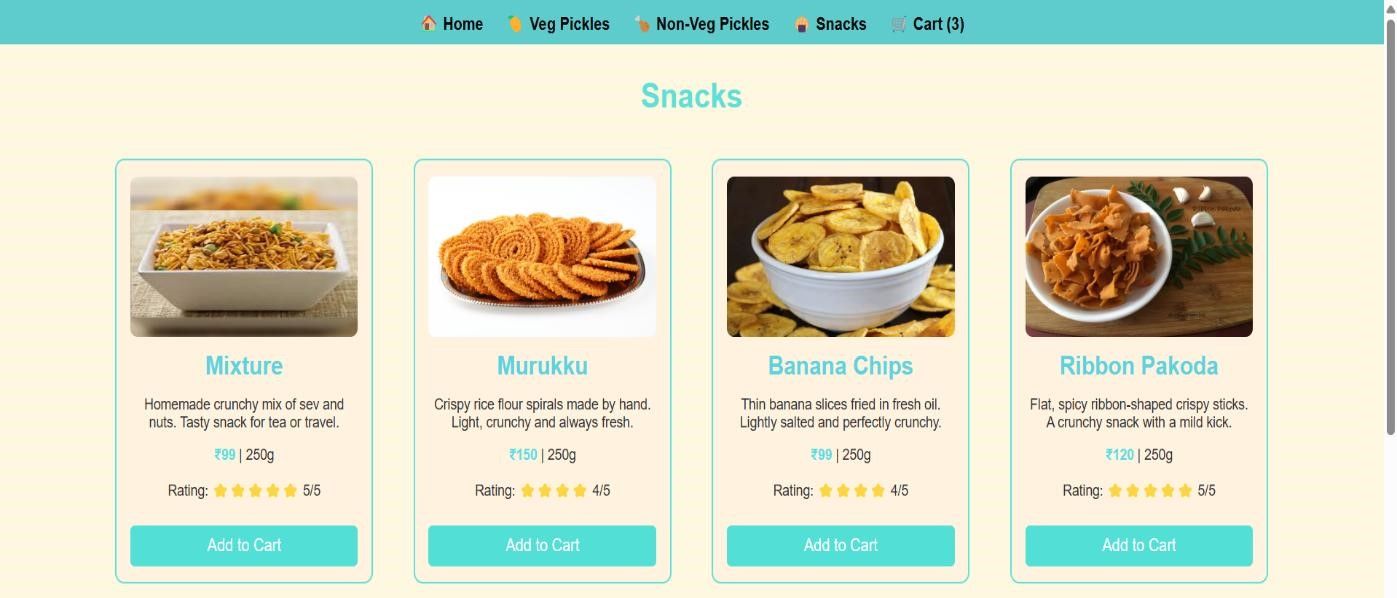


Non\_veg Pickles Page:

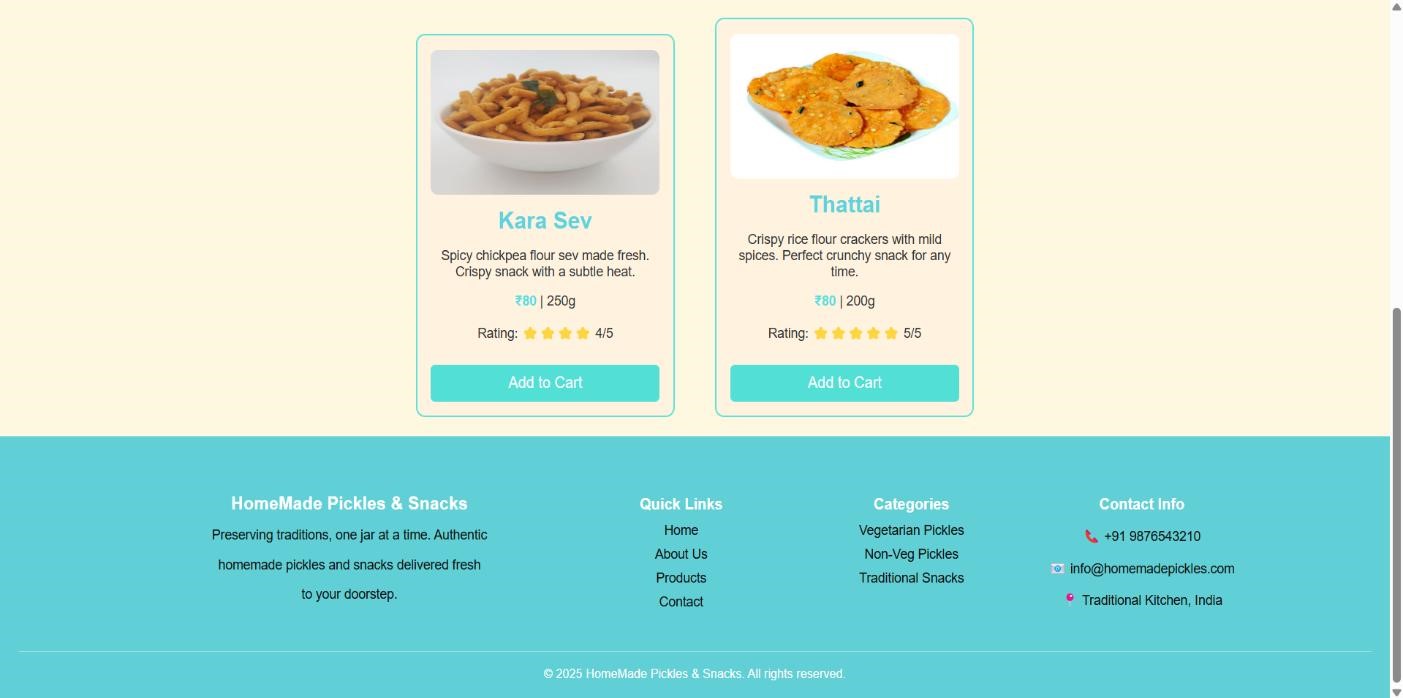




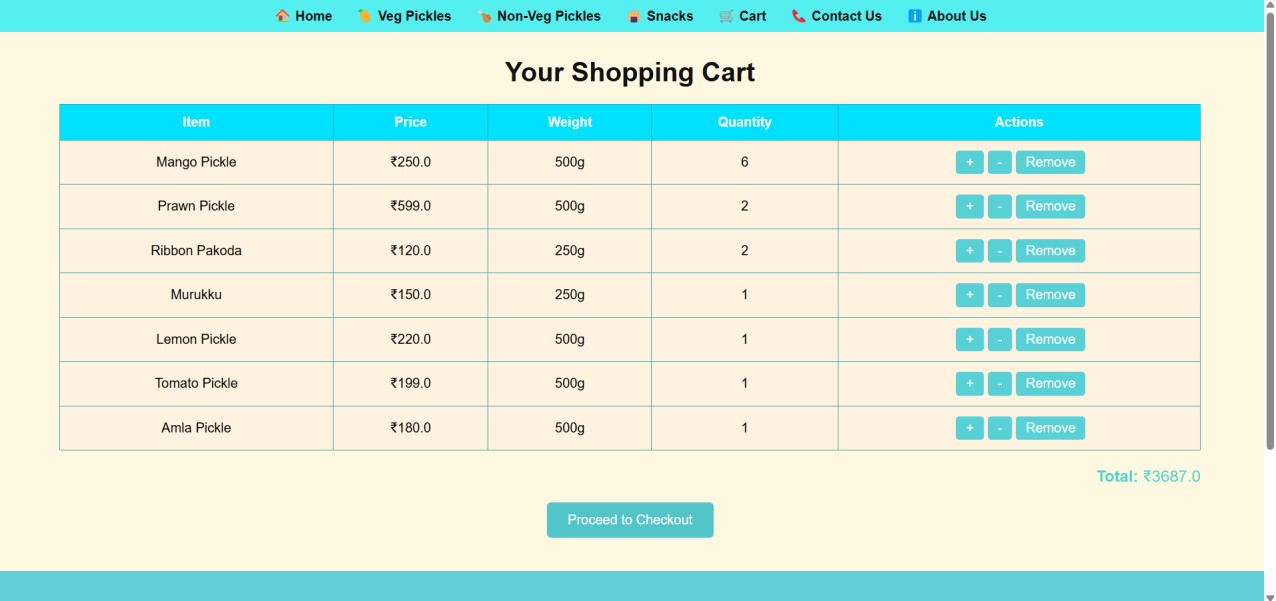
Snacks Page:



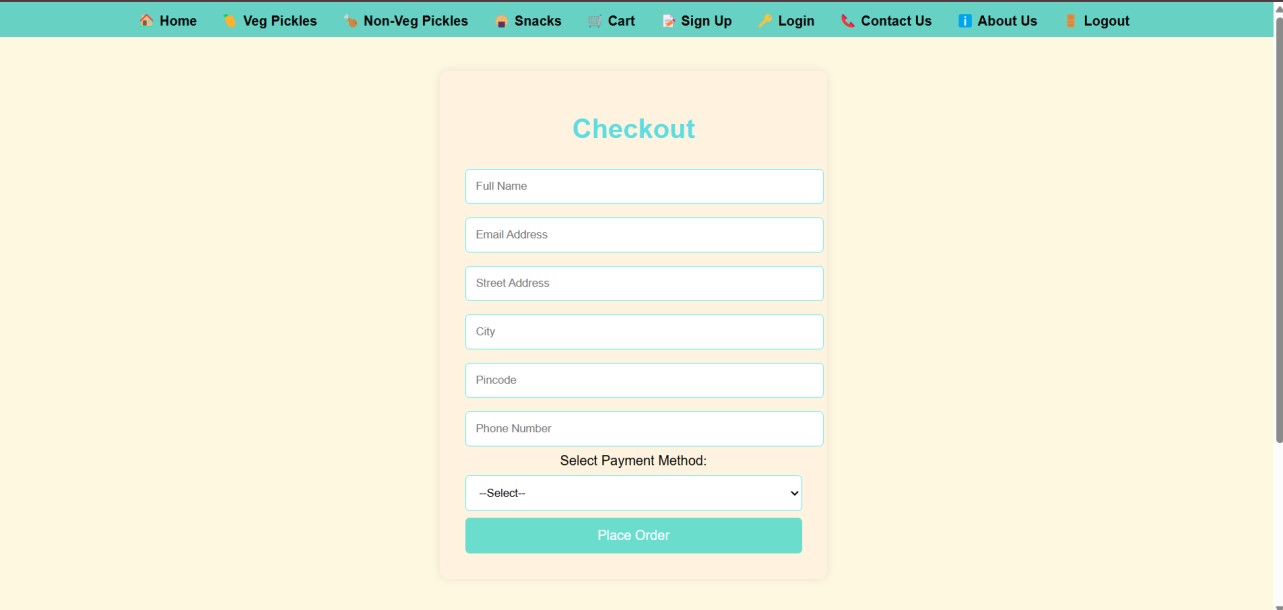
+



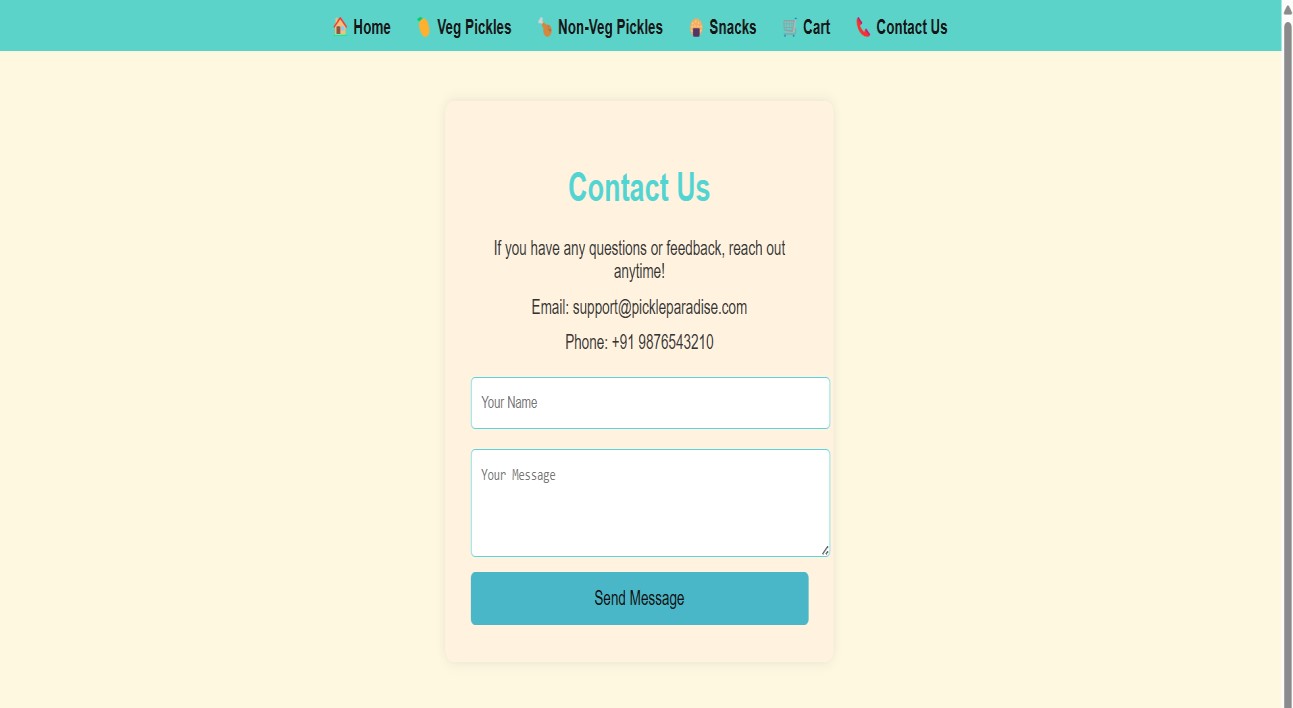
Cart Page:



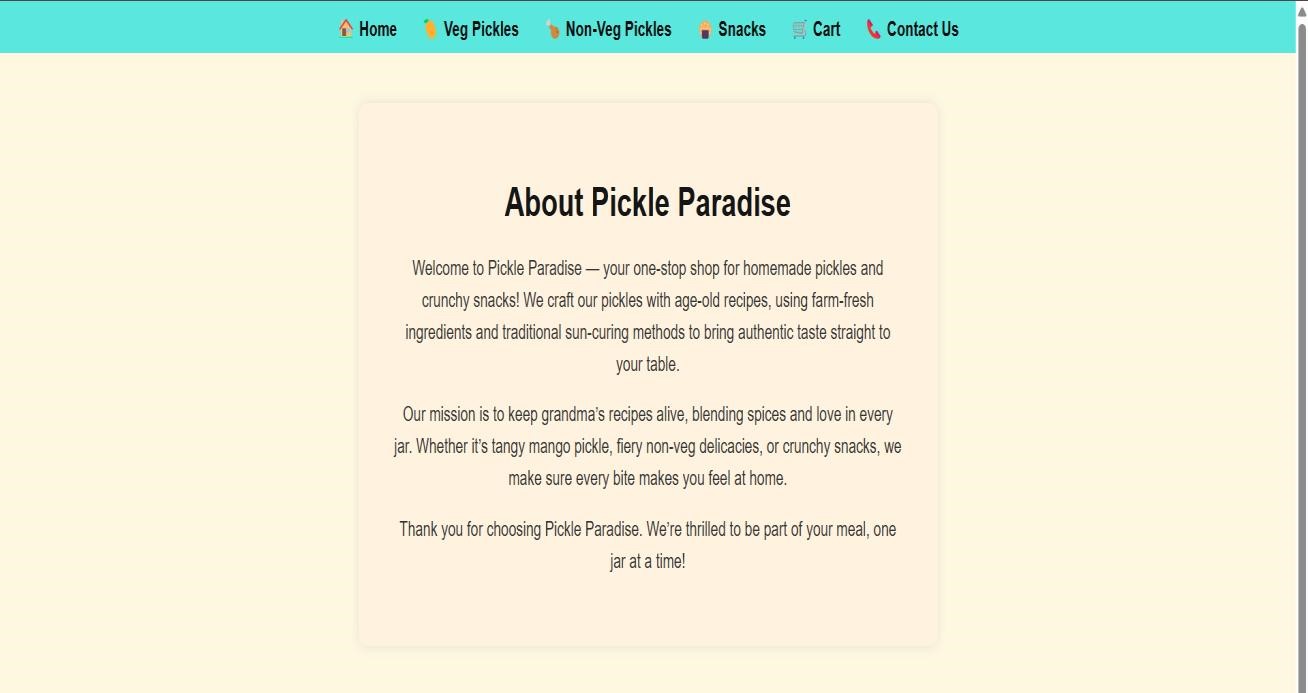
Checkout Page:



Contact Us Page:



About Us Page:



After Logout :

