

**Local Artisan Bakery: Powered by AWS for Fresh Bakes, Custom**

**Orders, and Easy Home Delivery**

**By**

**Bhavanam Dhana Lakshmi**

[**218x1a4208@khitguntur.ac.in**](mailto:218x1a4208@khitguntur.ac.in)

**Chavali Badrinadh**

[**218x1a4213@khitguntur.ac.in**](mailto:218x1a4213@khitguntur.ac.in)

**Kambala Pavan Kumar**

[**218x1a4257@khitguntur.ac.in**](mailto:218x1a4257@khitguntur.ac.in)

**Ashish Deelip Bhosale**

[**218x1a4251@khitguntur.ac.in**](mailto:218x1a4251@khitguntur.ac.in)

**Local Artisan Bakery: Powered by AWS for Fresh Bakes, Custom**

**Orders, and Easy Home Delivery**

### Project Description:

### In today’s fast-paced world, having access to fresh, custom-made bakery items delivered straight to your doorstep is a convenience for customers' value. This project aims to build a scalable and reliable web application that facilitates online ordering, custom requests, and seamless home delivery of freshly baked goods. By utilizing cloud-native development practices and integrating key AWS services, the platform ensures real-time order management, efficient inventory tracking, and smooth delivery coordination.

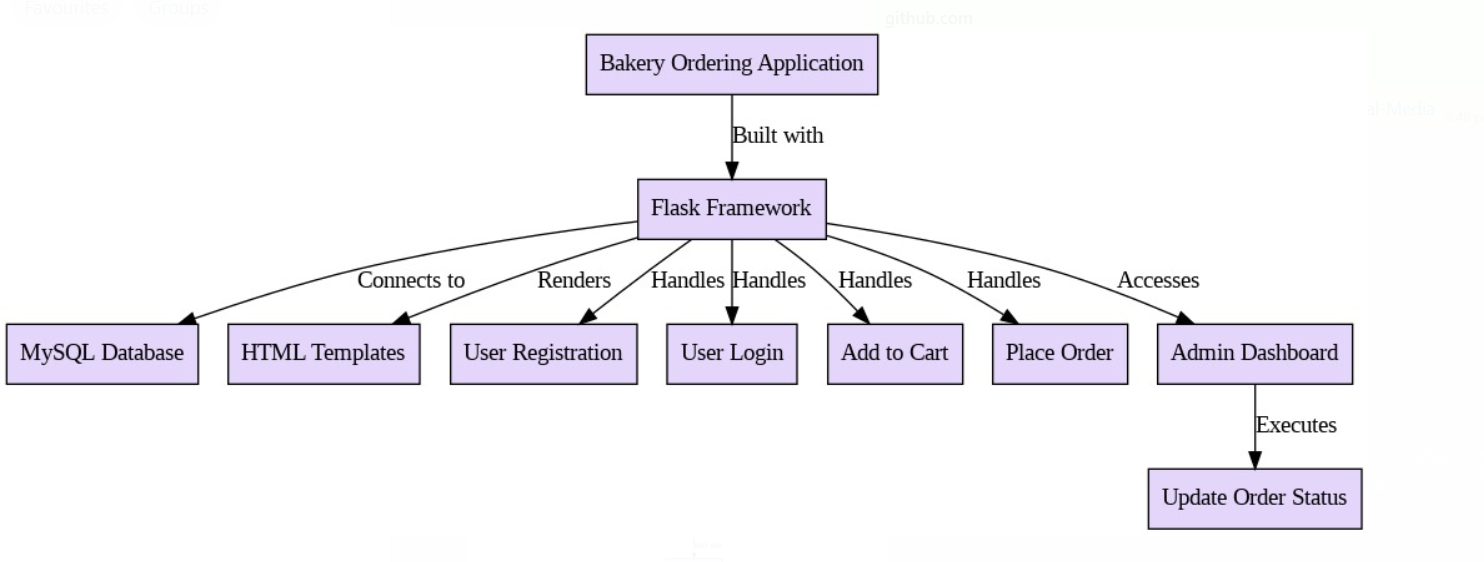
### The bakery ordering system will be developed using a lightweight web framework, hosted on Amazon EC2 for scalability. Orders and customer requests will be processed in real time using AWS Lambda, while DynamoDB will store order histories and customer preferences for future recommendations. By leveraging AWS services such as CloudWatch for performance monitoring and DynamoDB for managing order data, this project provides an efficient and scalable solution for delivering fresh, custom bakery goods right to customers' homes.

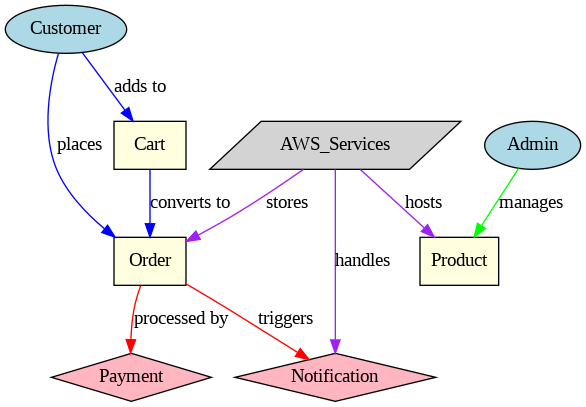
**Scenario 1: Exploring the Bakery Menu and Creating a Wishlist**A user visits the **Local Artisan Bakery** website, where they browse through a variety of fresh, custom-baked goods ranging from cakes to artisanal bread. During their exploration, they come across a beautifully crafted cake and decide to add it to their Wishlist. The backend, powered by Flask and AWS DynamoDB, securely stores the wishlist item associated with the user's profile for easy access in the future. When the user revisits the site, they can review their Wishlist and make custom orders, streamlining their experience and helping them decide on their next purchase.

**Scenario 2: Participating in the Bakery Quiz for Discounts**After browsing the bakery’s offerings, the user takes part in the **Custom Bakes and Flavors Quiz**, designed to educate customers about the bakery’s specialties. They answer questions related to different baking styles, ingredients, and flavor combinations. Upon successfully completing the quiz, the platform generates a discount coupon code as a reward. The coupon, stored in AWS DynamoDB under the user's profile, is also displayed on their dashboard for future use during checkout, incentivizing the user to make a purchase.

**Scenario 3: Receiving Discounts and Redeeming at the Bakery**Having passed the quiz, the user receives a 10% discount coupon code via email and their user dashboard. The next time they visit the **Local Artisan Bakery** in person, they redeem the coupon while picking up a custom order. The backend system validates the coupon at the counter, ensuring a seamless and efficient transaction. This integration of online engagement with in-store purchasing enhances the overall customer experience

**AWS ARCHITECTURE:**



**Entity-relationship (ER) diagram:**

### 

**Pre-requisites:**

### 1. AWS Account Setup: [https://youtu.be/CjKhQoYeR4Q?si=ui8Bvk\_M4FfVM-D](https://youtu.be/CjKhQoYeR4Q?si=ui8Bvk_M4FfVM-Dh)h

### 2. Understanding of IAM: <https://youtu.be/gsgdAyGhV0o?si=3qg-bULgkD4LXNvR>

### 3. Knowledge of Amazon EC2 :<https://youtu.be/8TlukLu11Yo?si=MUj0nEAOESRhHUIz>

4. RDS : <https://www.youtube.com/live/MPau9c7PT74?si=A8OK-zFGbSKkAFWN>

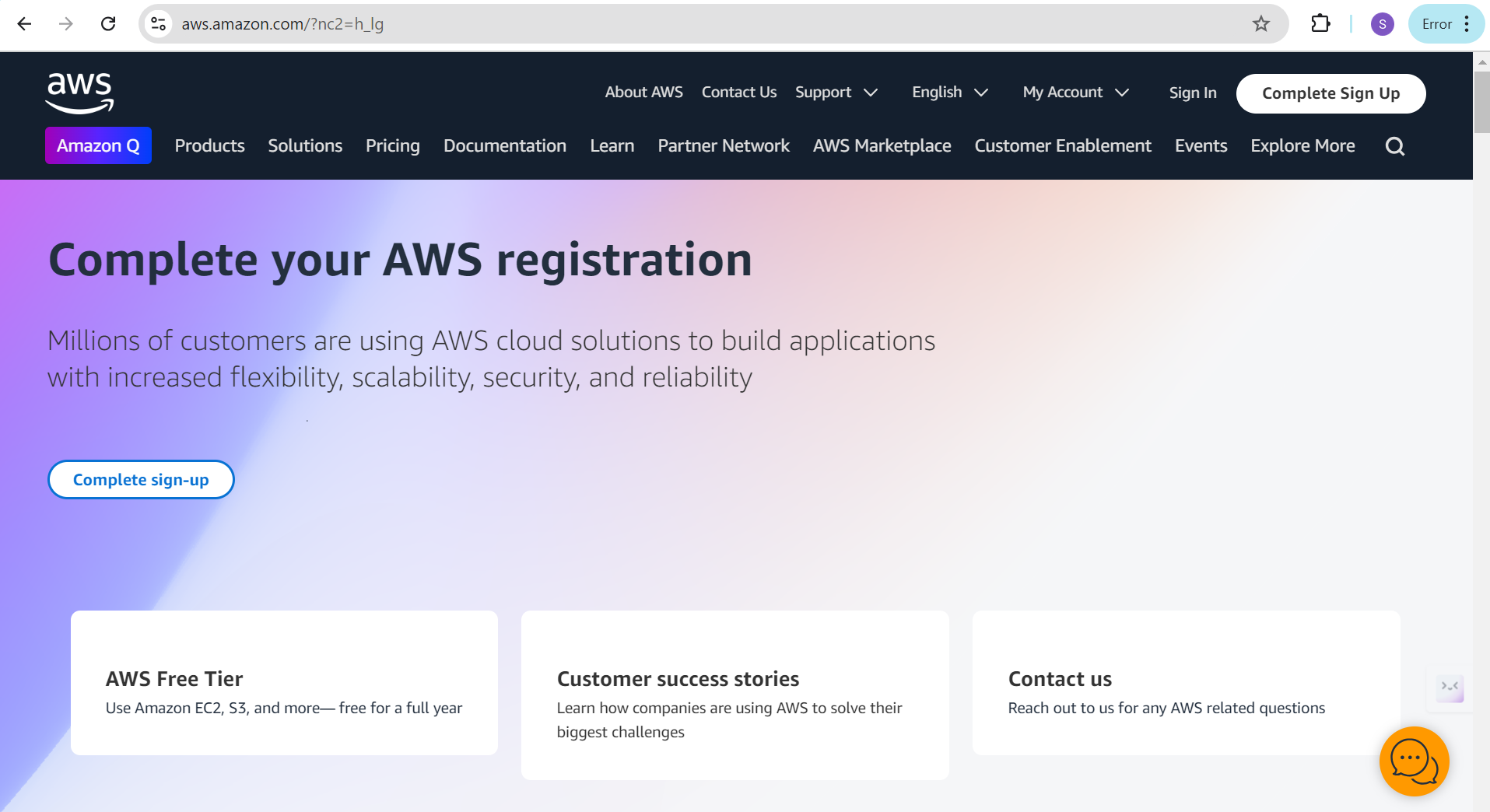
5. MySQL WorkBench: <https://youtu.be/wALCw0F8e9M?si=ovMF9qMx5rLxaznB>

### Project Flow

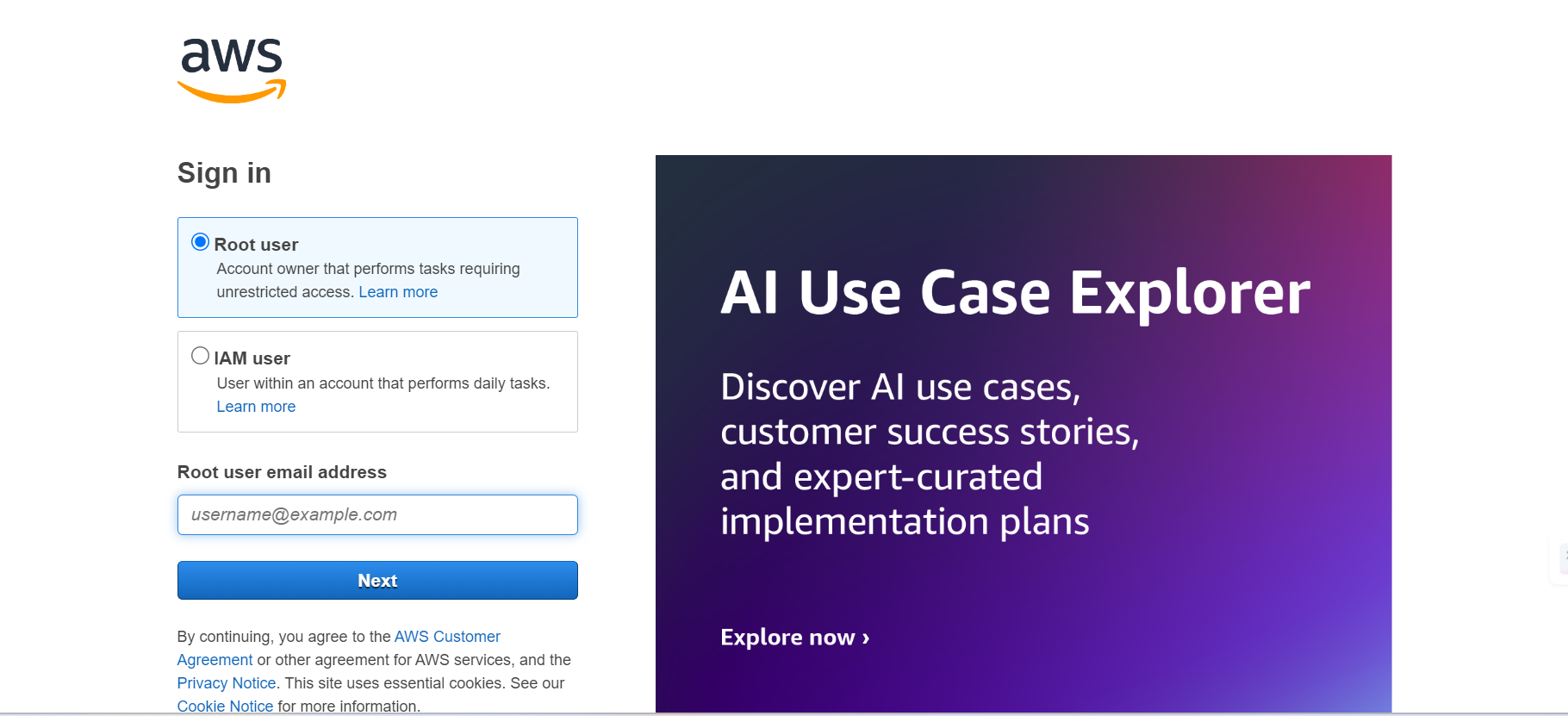
* **AWS Account Setup and Login**
  + **Activity 1.1:** Set up an AWS account if not already done.
  + **Activity 1.2:** Log in to the AWS Management Console to manage resources.
* **RDS Database Creation and Setup**
  + **Activity 2.1:Create an RDS Instance.**
* Navigate to the RDS service in the AWS Console and configure your database instance (select MySQL).
  + **Activity 2.3**: Install MySQL Workbench.
    - Download and install MySQL Workbench on your local machine.
    - Use the endpoint and credentials of your RDS instance to establish a connection via MySQL Workbench.
* **Frontend Development and Application Setup**
  + **Activity 3.1**: Build the Frontend.
    - Develop the frontend for the online food delivery application using HTML, CSS, and Flask (Python-based)
    - Ensure the structure supports the ordering functionality and user interaction.
* **EC2 Instance Setup**
  + **Activity 4.1**: Launch EC2 Instance.
    - From the AWS Console, launch a Linux-based EC2 instance.
    - SSH into the instance and prepare the environment for hosting the application.
* **Deployment on EC2**
  + **Activity 5.1**: Deploy to EC2.
    - Transfer the developed Flask application to the EC2 instance.
    - Install necessary dependencies (e.g., Flask, MySQL libraries) on EC2.
    - Configure the EC2 instance to connect to the RDS database.
    - Start the Flask application on the instance.
* **Testing and Deployment**
  + **Activity 6.1**: Functional Testing.
    - Test the full application for functionality including frontend interaction, database communication, and overall performance.
    - Run the Flask app with python3 app.py and access the link provided to verify its correct functioning.
  + **Activity 6.2**: Deployment.
    - Finalize the deployment in the production environment.
    - Ensure high availability, security, and performance optimization.

**Milestone 1: AWS Account Setup and Login**

* **Activity 1.1: Create AWS Account**
  + Sign up for an AWS account and configure billing settings.

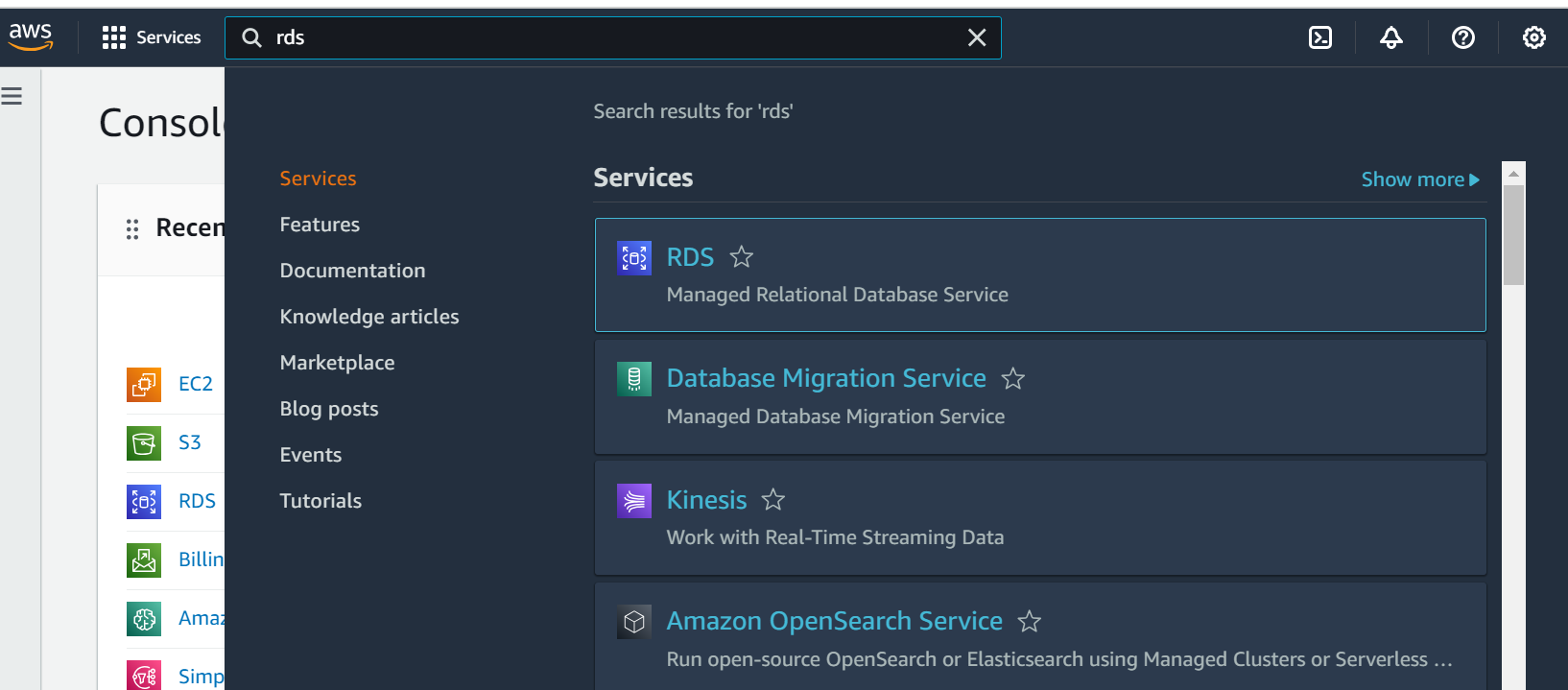


* **Activity 1.2: Login to AWS Management Console**
  + Access the AWS Management Console using your login credentials.

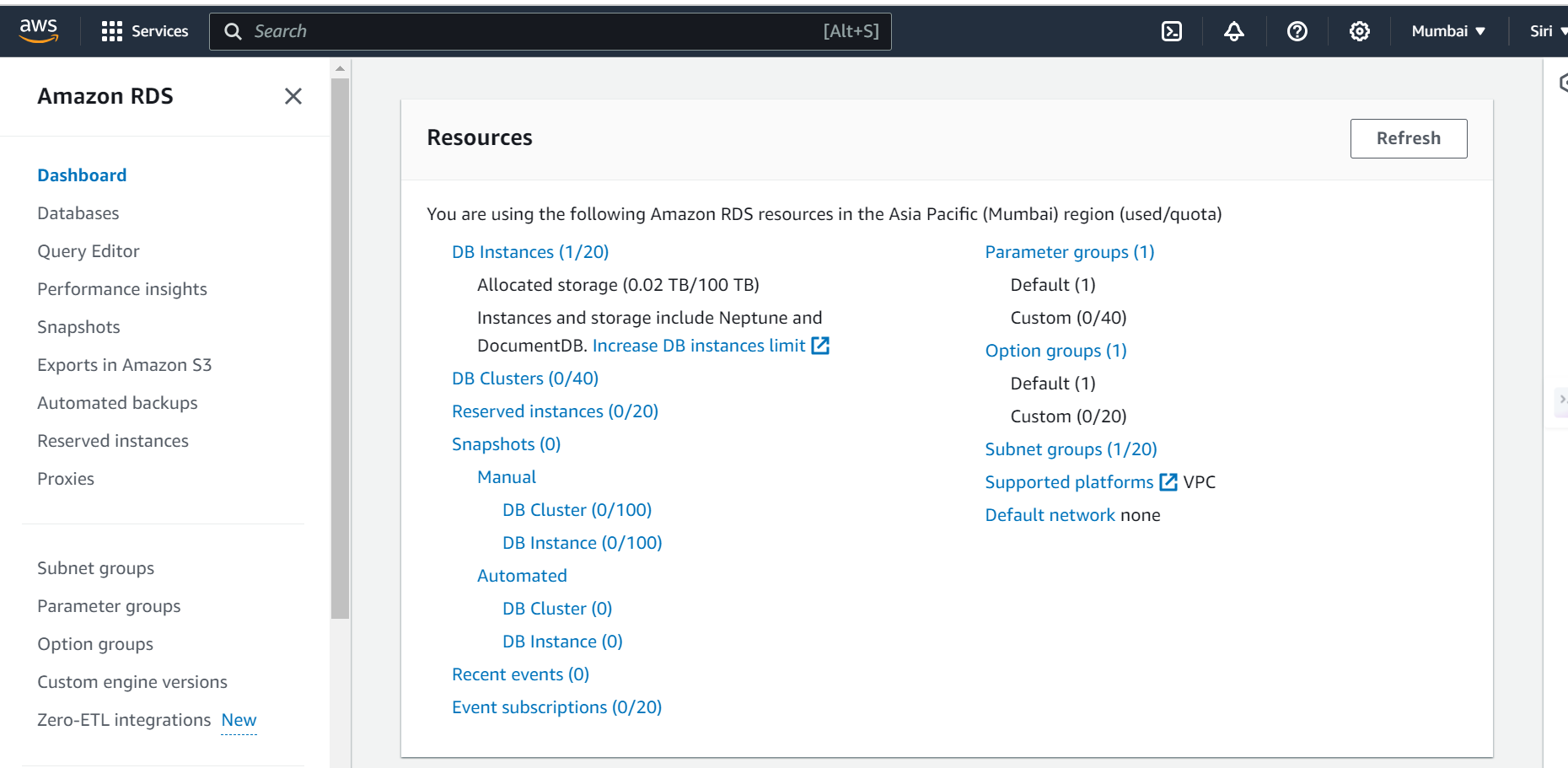


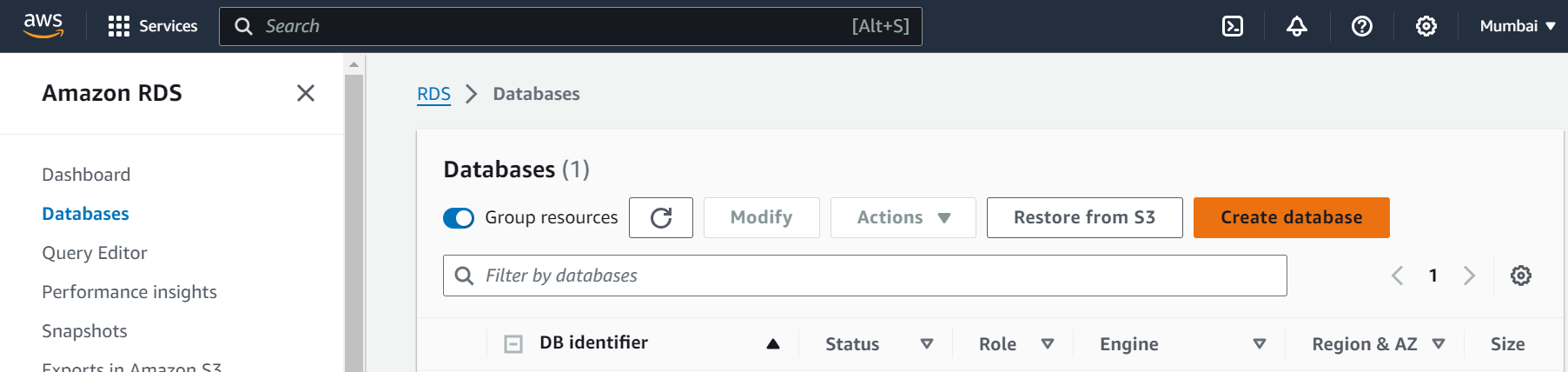
**Milestone 2: RDS Database Creation and Setup**

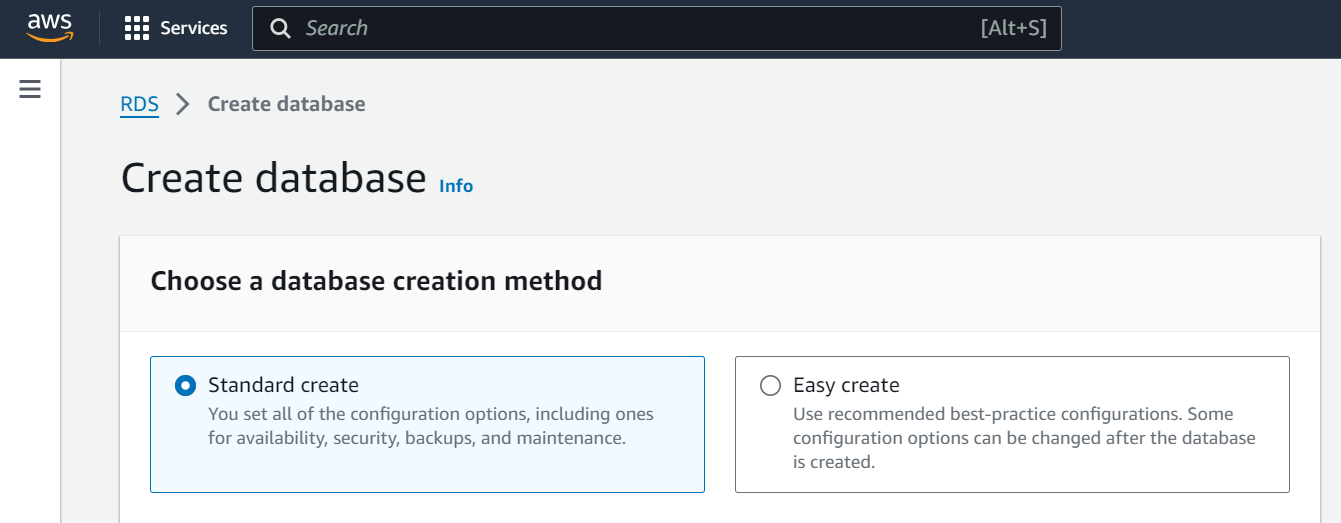
* **Activity 2.1: Create an RDS Instance**
  + Choose the RDS service from the AWS Management Console.

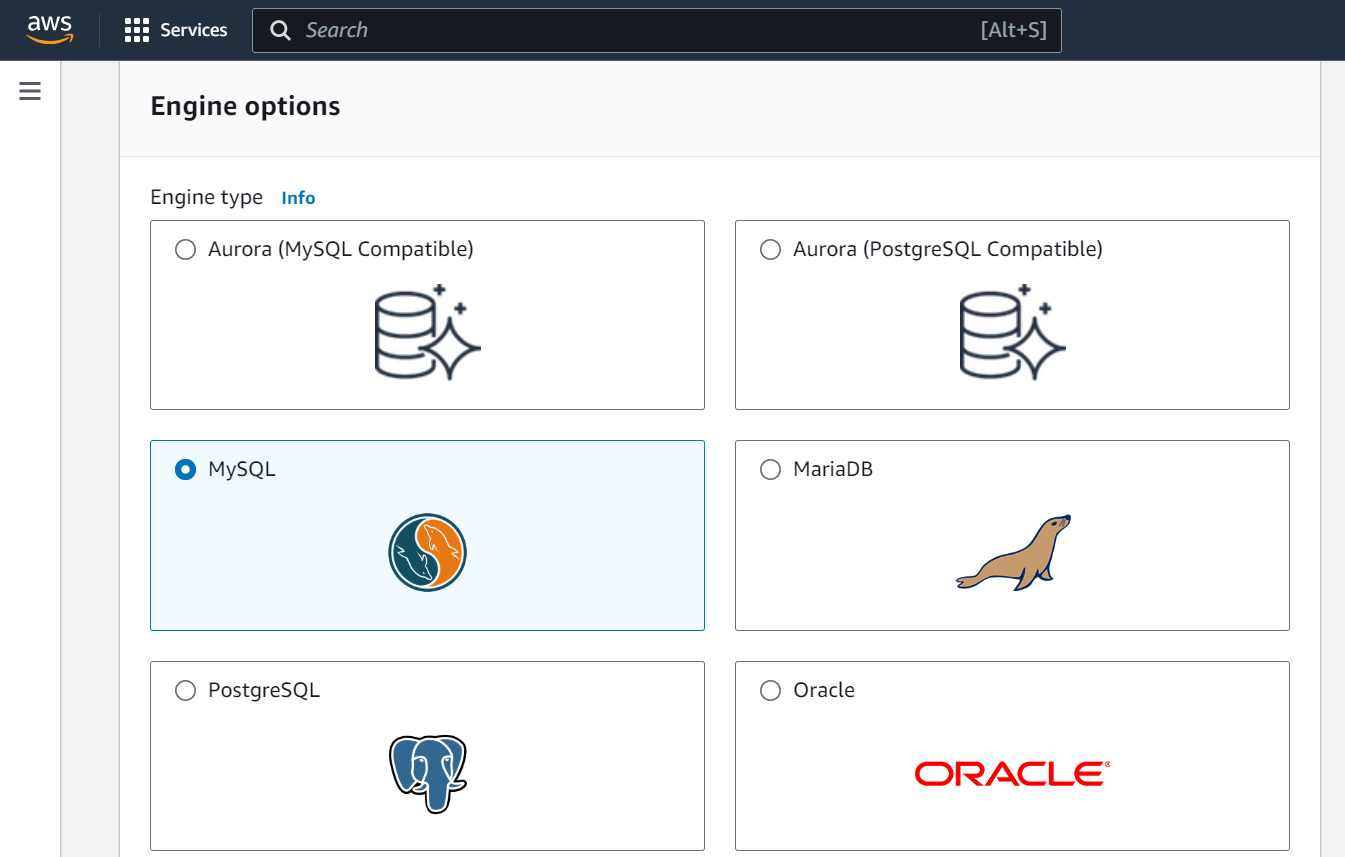


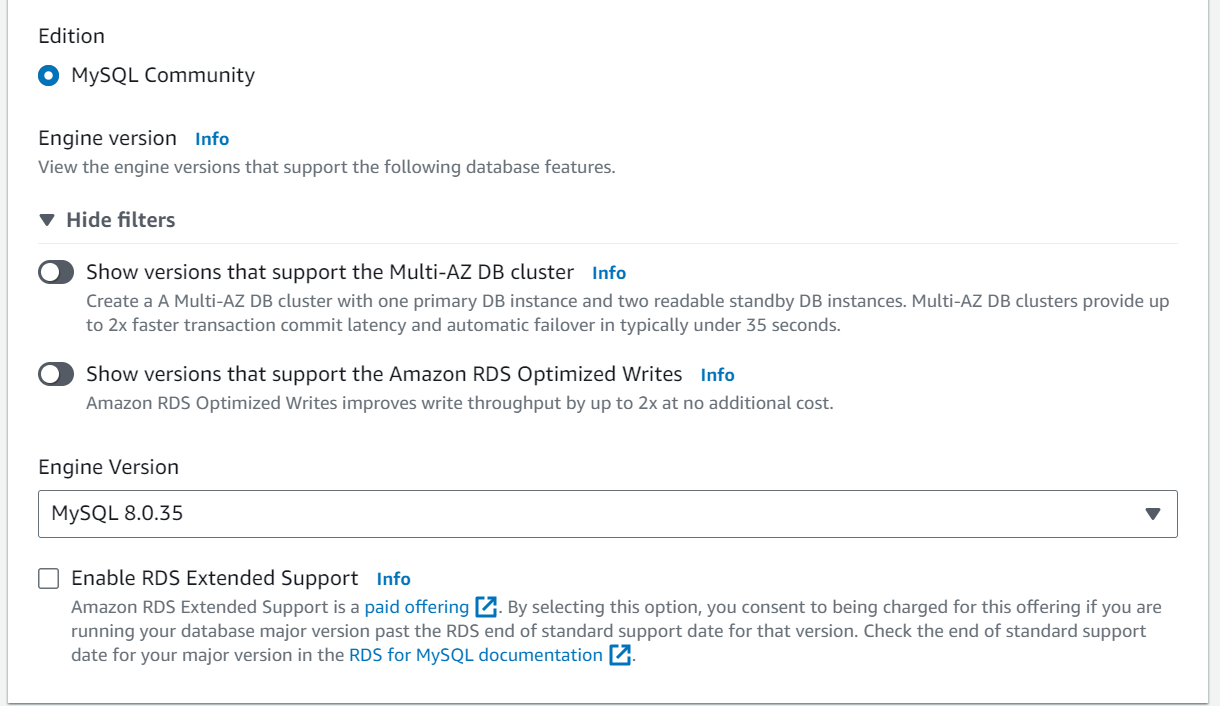
* + Select MySQL as the database engine, configure the instance settings (e.g., storage, instance class), and launch the RDS instance.

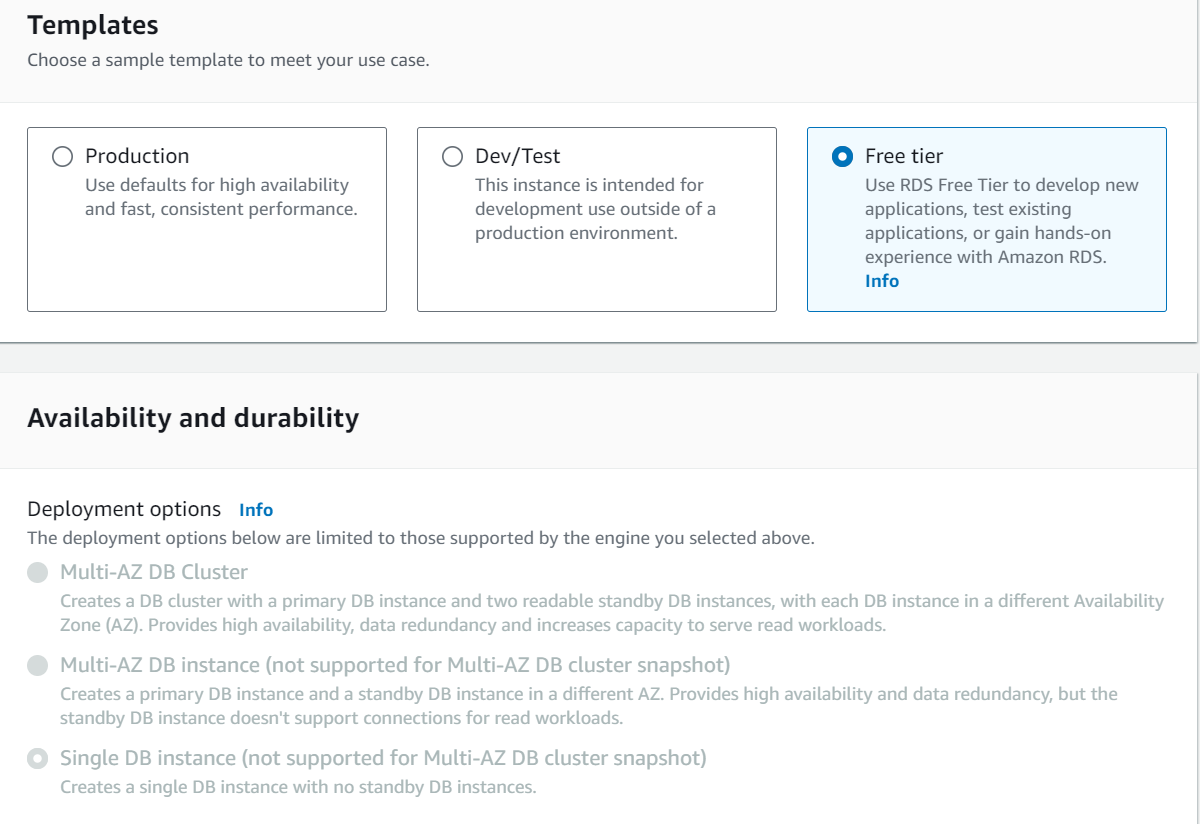


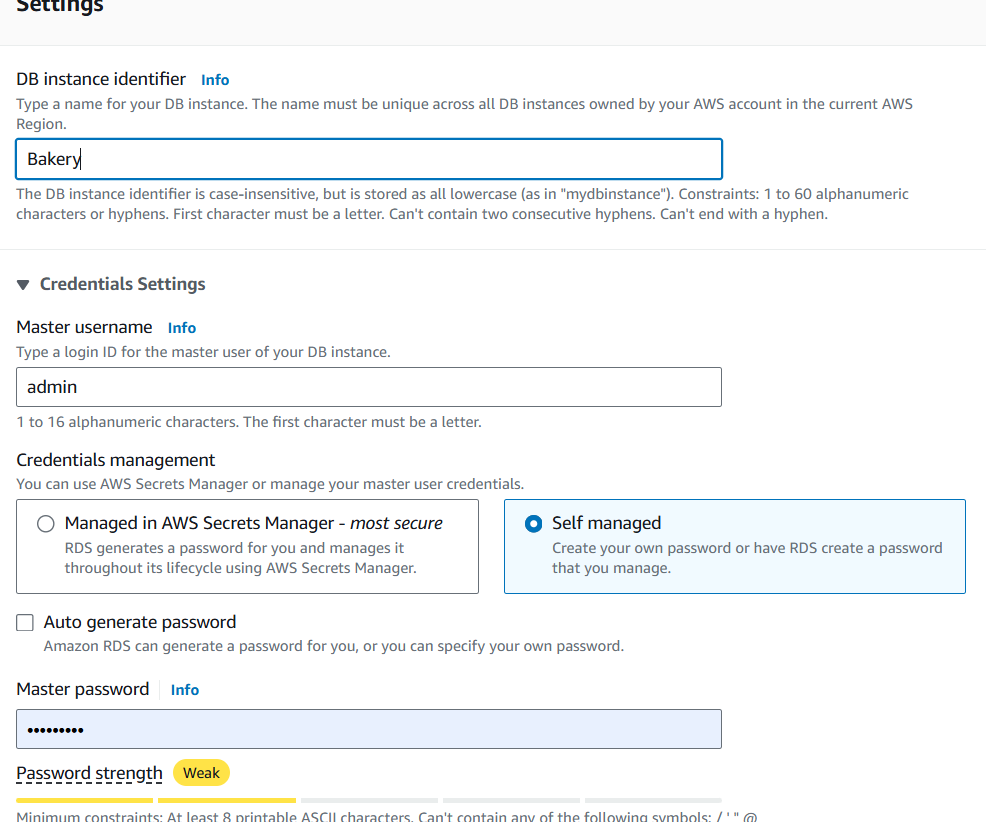


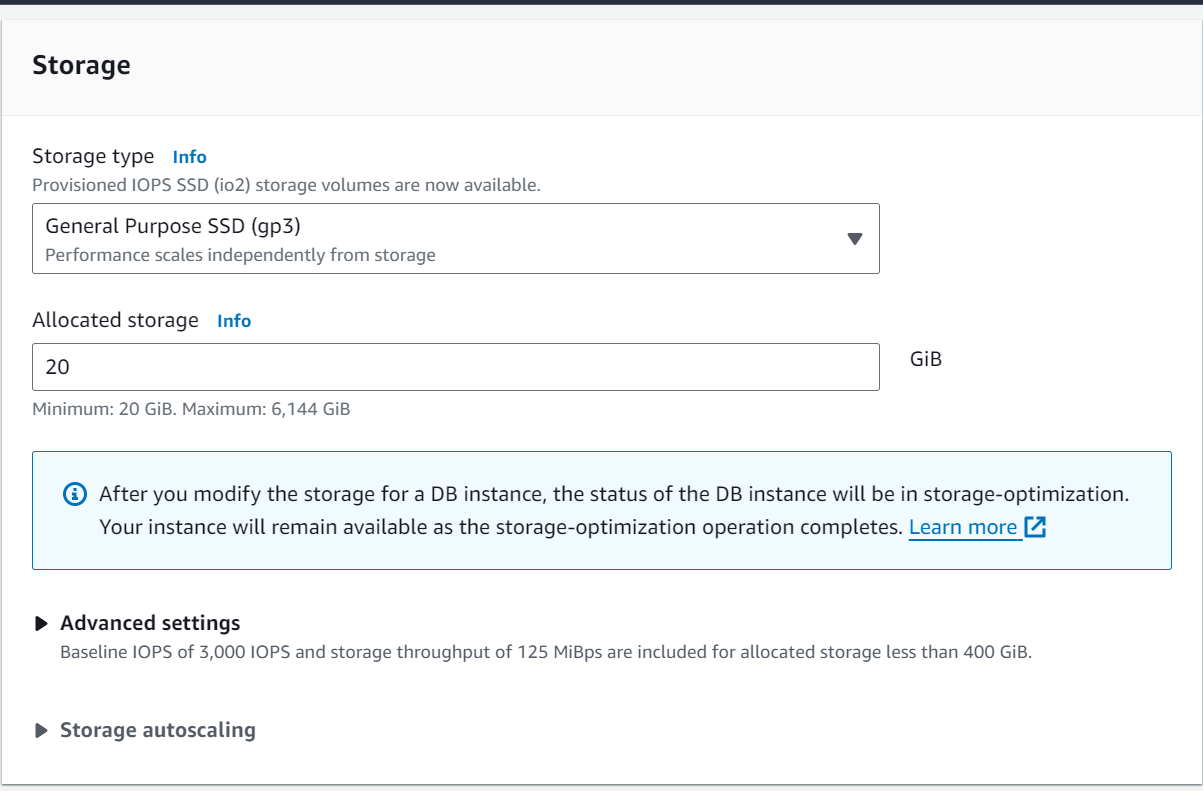


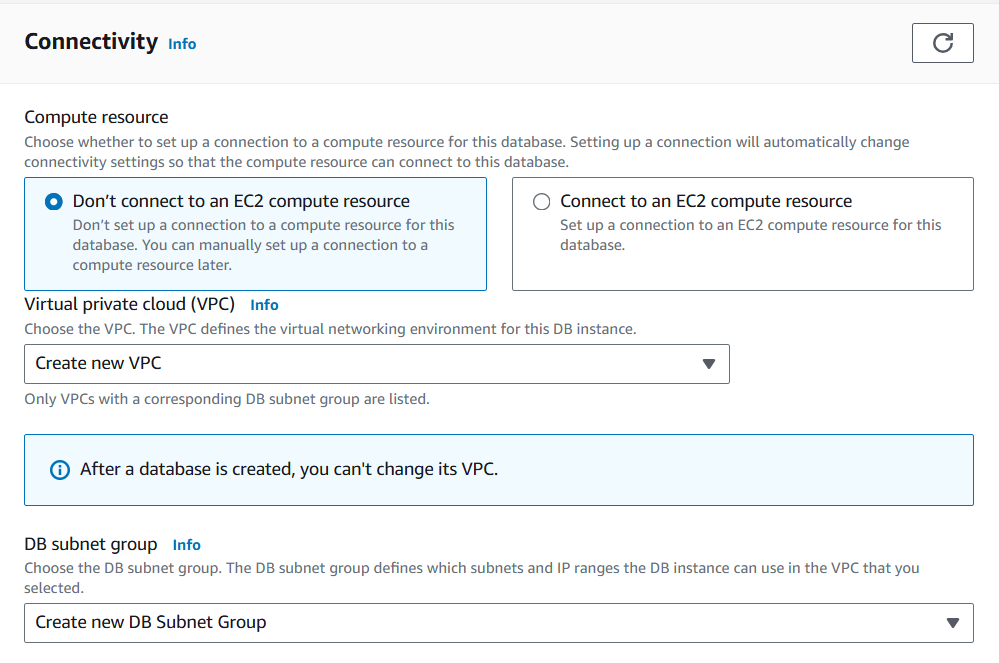


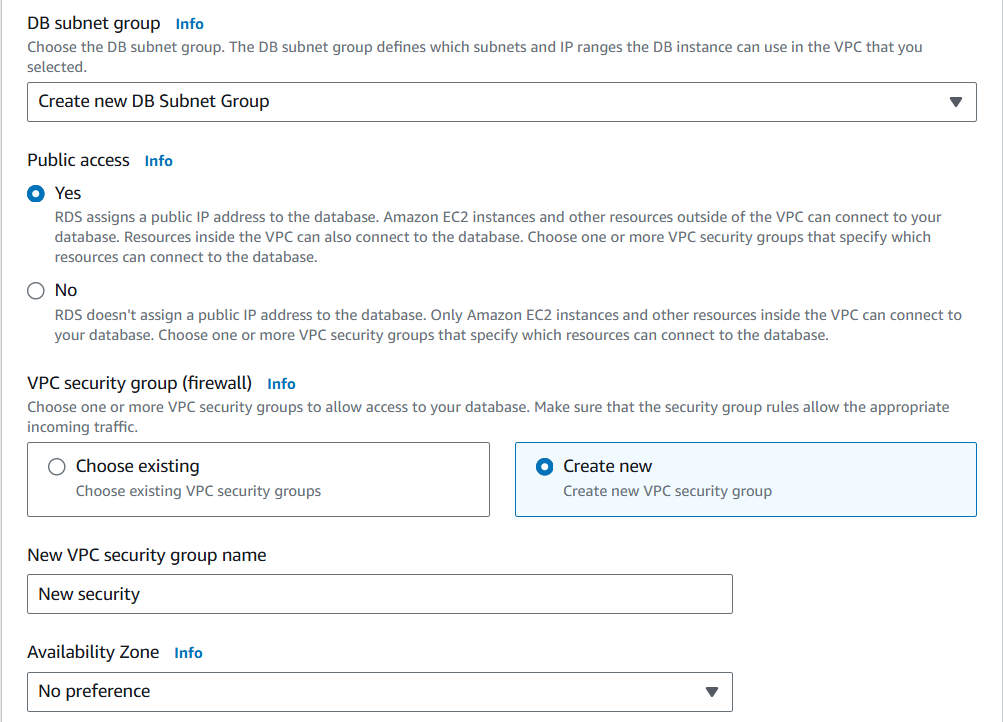


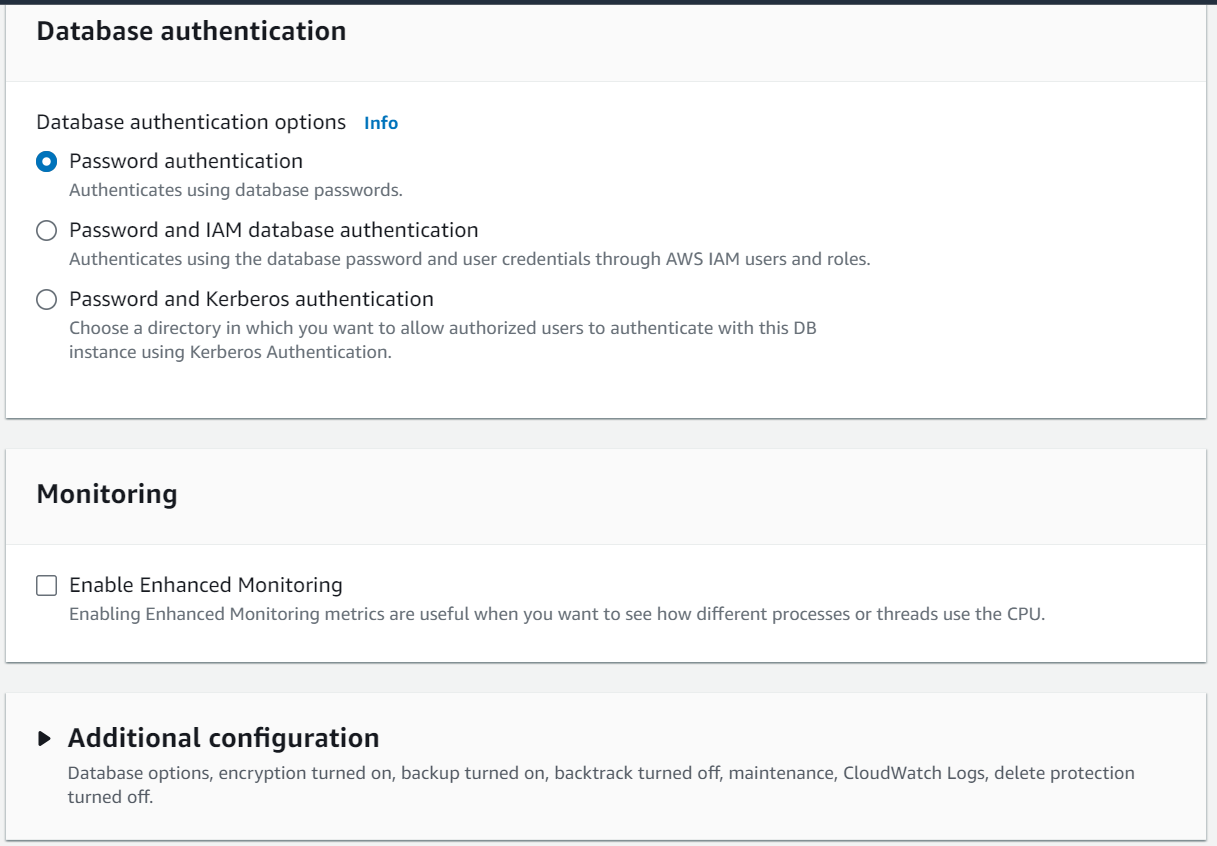


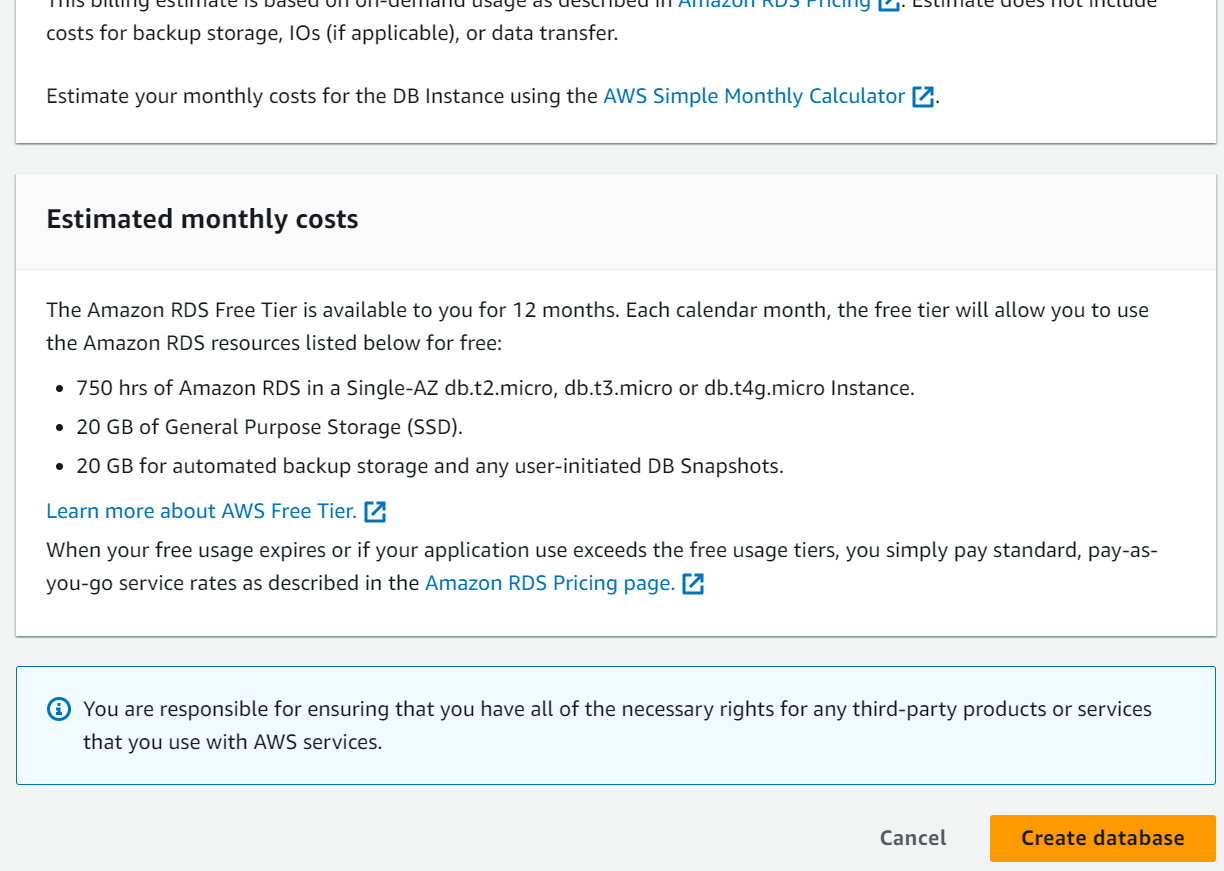




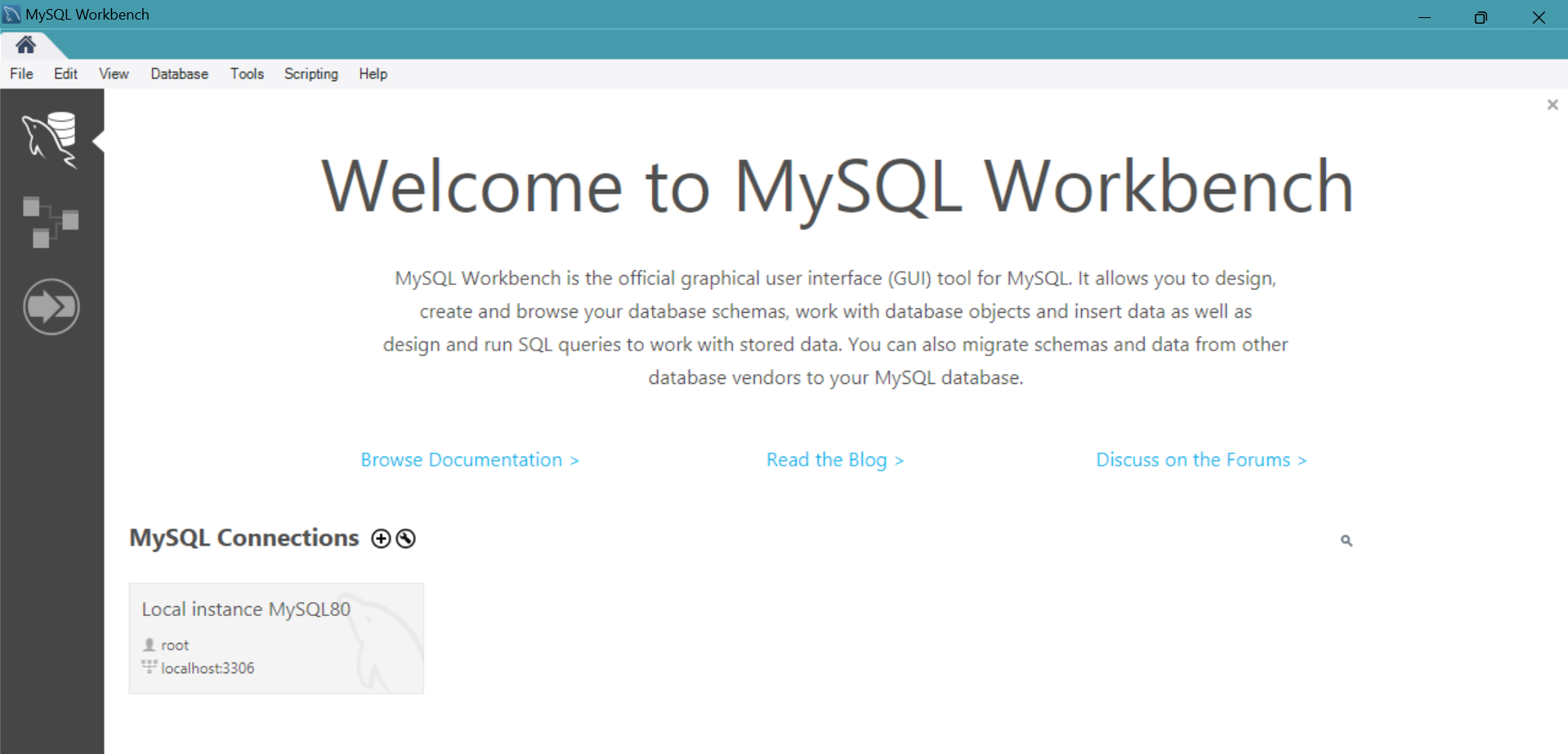
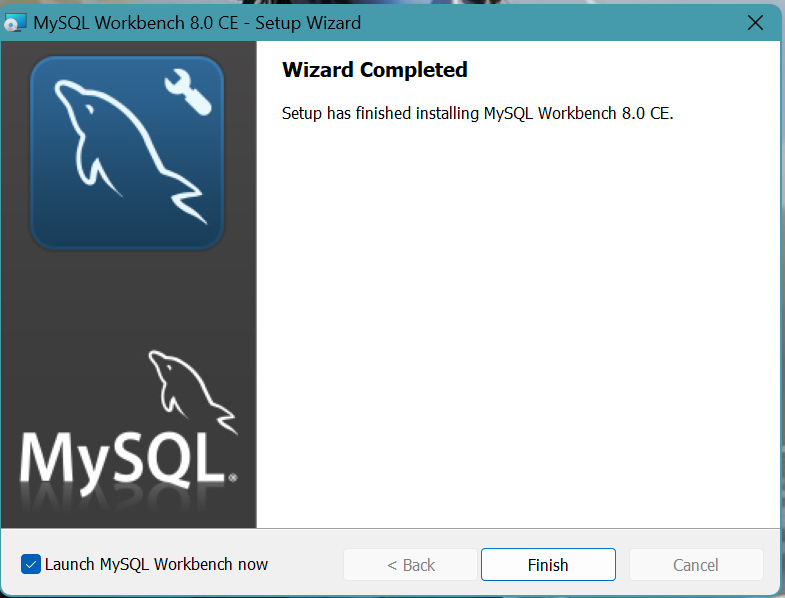


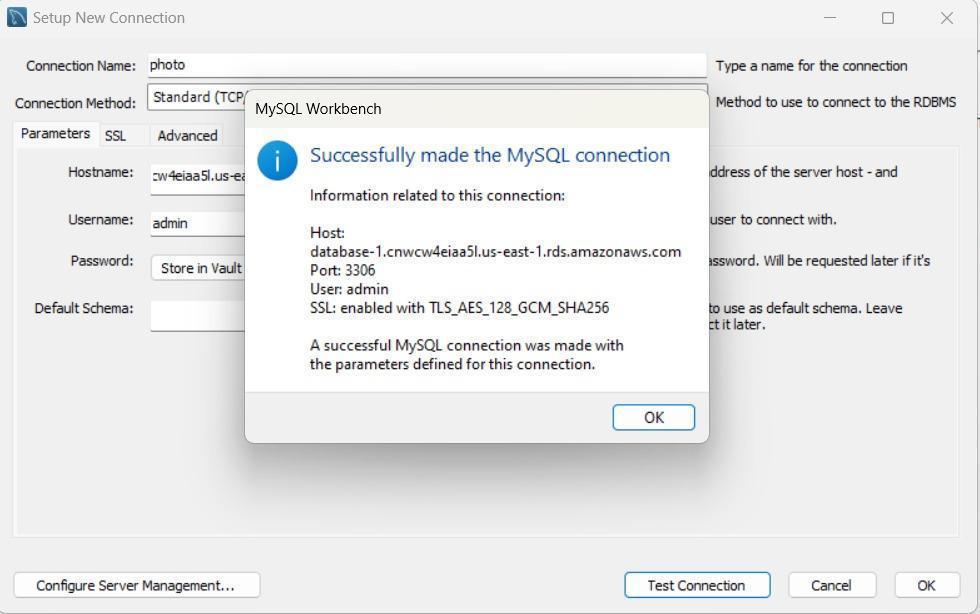
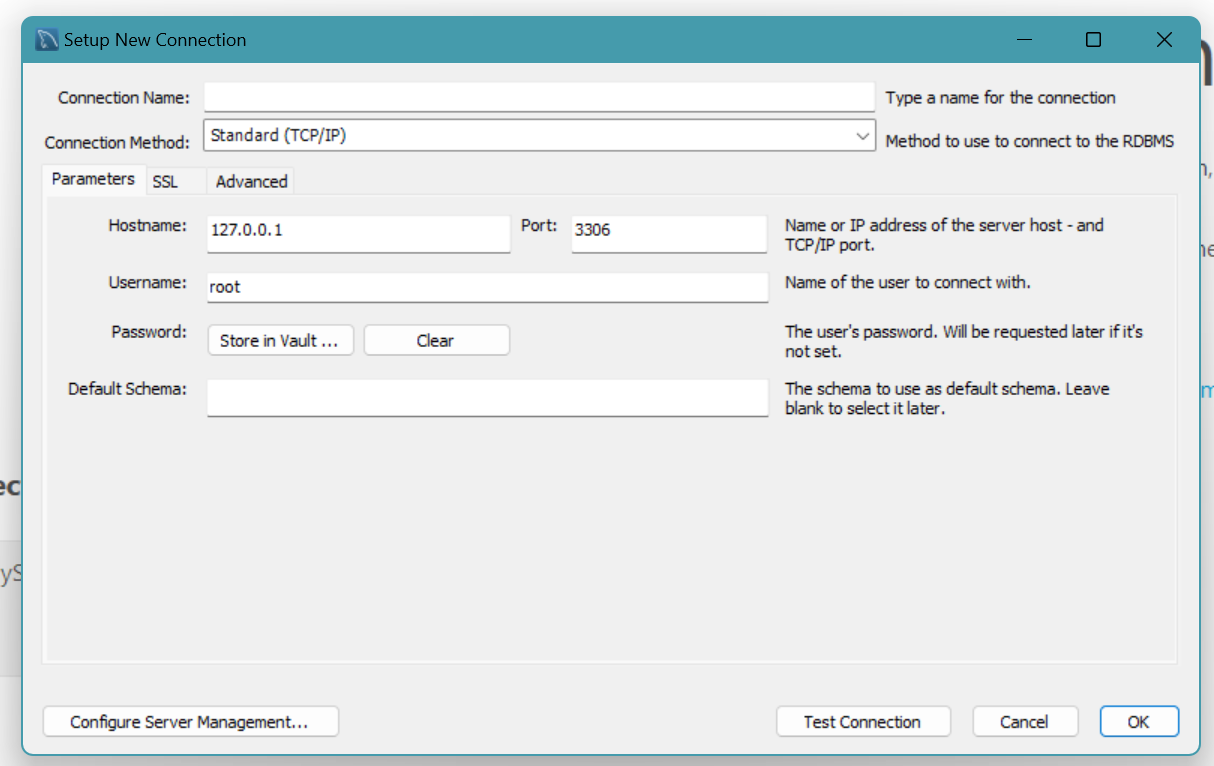


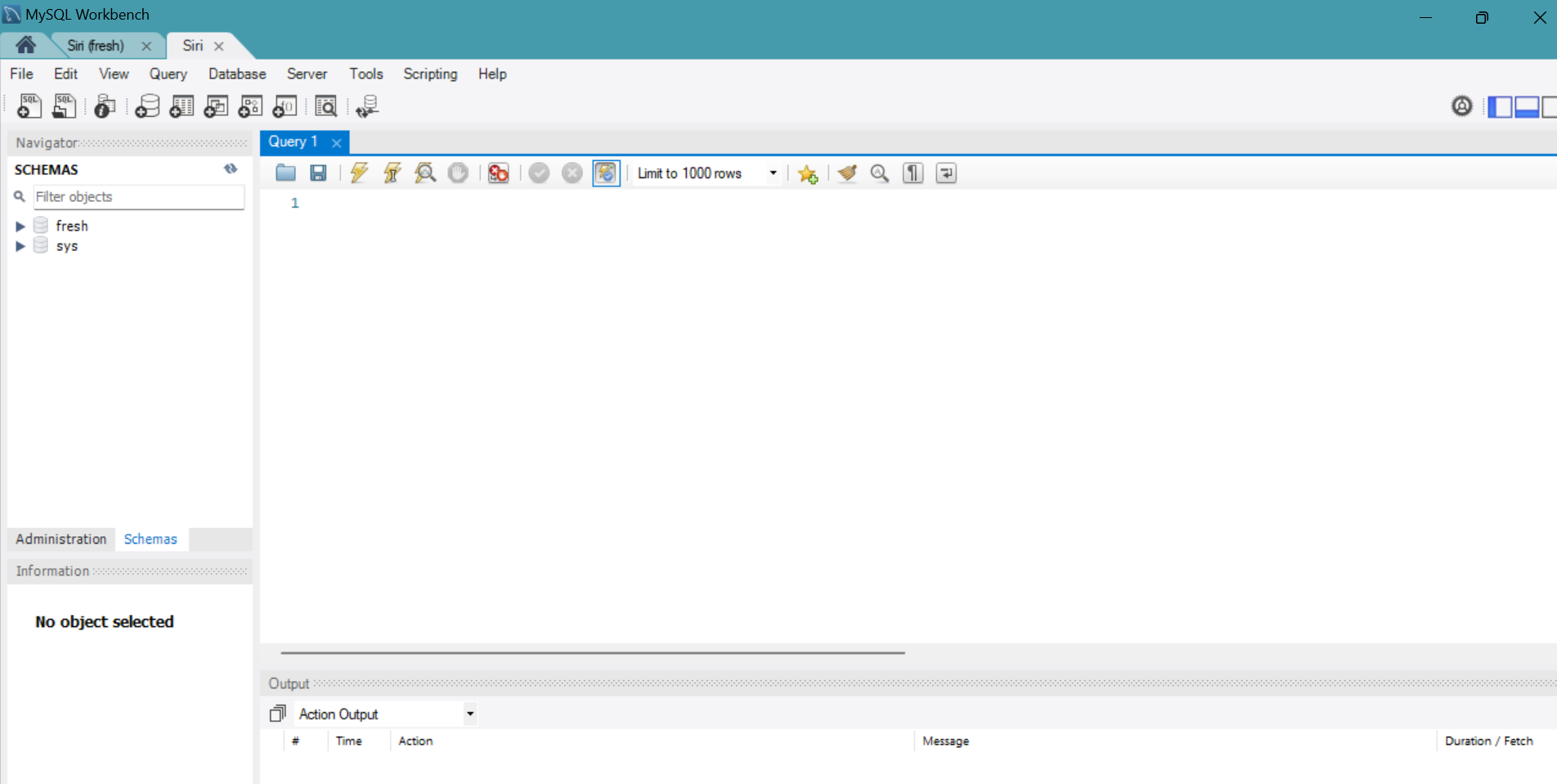




* **Activity 2.2: Configure Database Access**
  + Set up security groups, create database credentials, and configure access policies to ensure secure connectivity to the database.
* **Activity 2.3: Install MySQL Workbench**
  + Download and install MySQL Workbench on your local machine for database management.

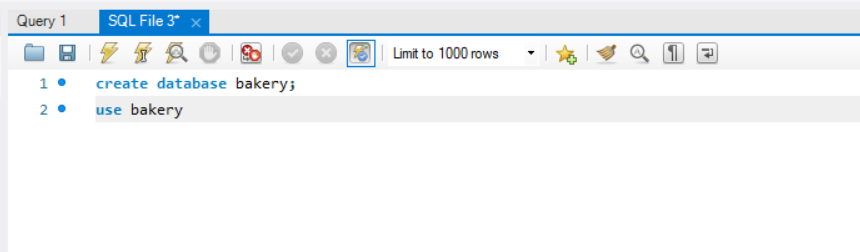


* + Connect to the RDS instance via MySQL Workbench using the endpoint and credentials from AW
  + 
* Give a connection name.
* Copy the endpoint from the RDS database that is created in AWS and paste it in **Hostname.**
* Write the username and enter the password , then click on **Test Connection**.
* Once the connection is successful, you’ll be welcomed with this interface.



**Activity 2.4:** Create the Database and the tables which are required.

* **Create a basic database schema for Local Artisan Bakery: Powered by AWS for Fresh Bakes, Custom Orders, and Easy Home Delivery**



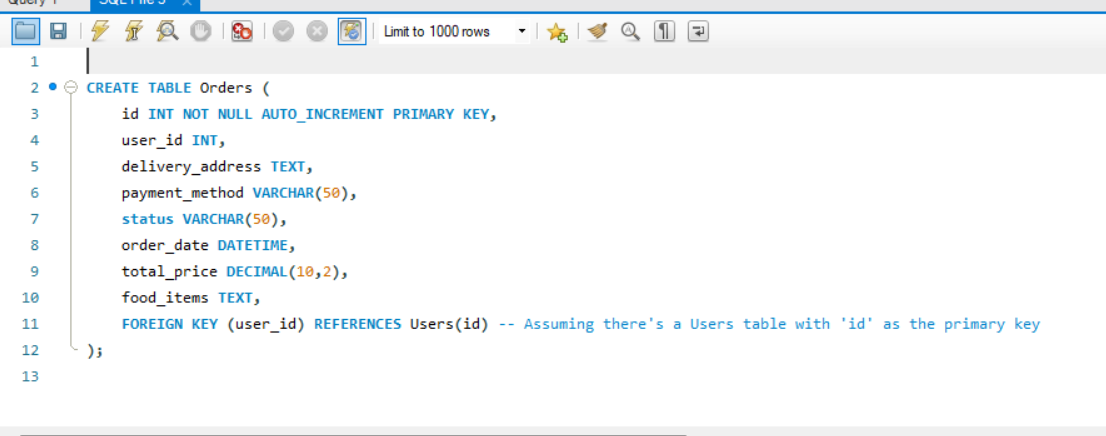
**users:**

* Stores user information such as name, email, password, and mobile number.
* Each user has a unique ID (id), which serves as the primary key.
* The email column is unique to ensure no duplicate accounts.



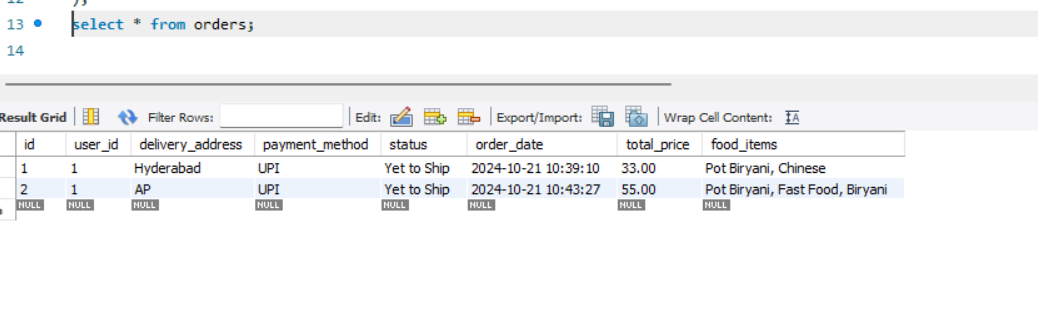
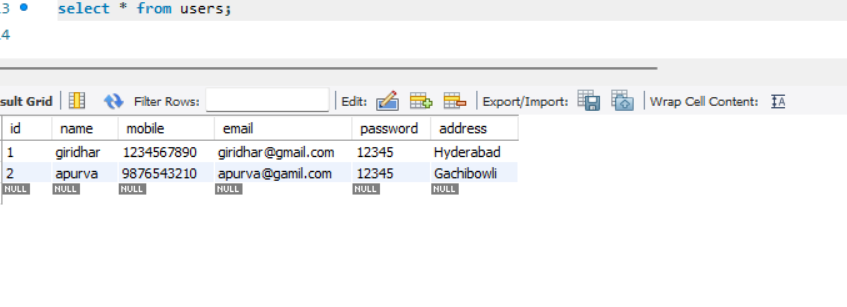
**bookings:**

* Contains register details, including the user ID (foreign key referencing the users table), food number of people, check-in/check-out dates, special requests, and total price.
* Each booking is assigned a unique identifier (id), which is the primary key.
* The user\_id column links to the users table, establishing a relationship between bookings and the corresponding user.

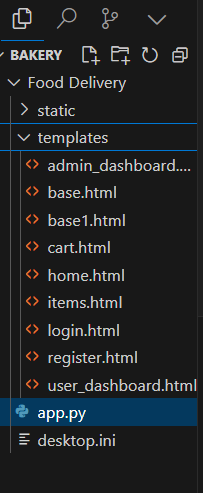


### Data Operations:

**1. Data Retrieval**: We retrieved data from each table to view the stored information, including user details, available items, and orders.

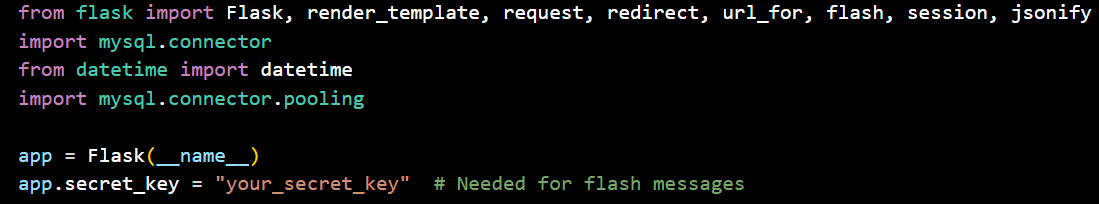


**Milestone 3: Frontend Development and Application Setup**

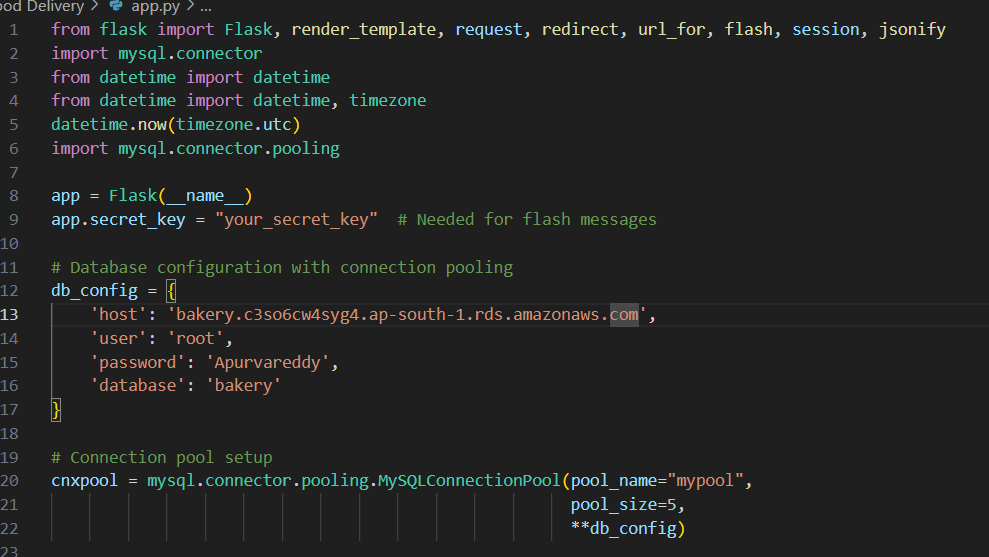
* **Activity 3.1: Build the Frontend**
  + Develop HTML, CSS, and Python-based Flask application files
* **Activity 3.2: Integrate Application with RDS**
  + Connect app.py (Flask application) to the MySQL RDS database by configuring database connection settings and verifying connectivity.

**Description of the code :**

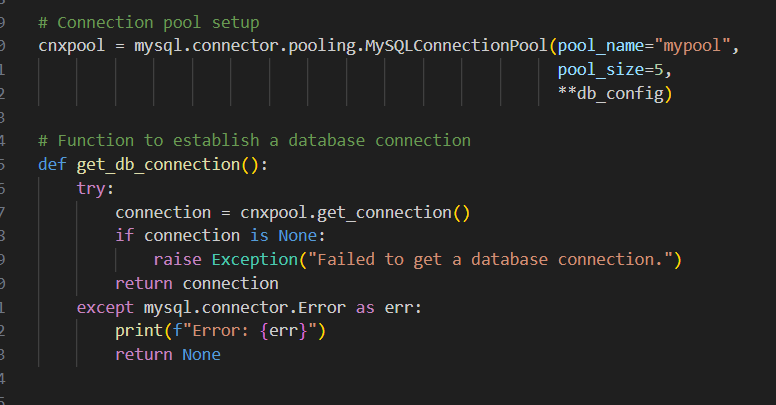
Fresh Bakes, Custom Orders, and Easy Home Delivery utilizes AWS to provide a seamless experience for customers ordering handcrafted baked goods. This integration allows for efficient management of orders, real-time inventory updates, and personalized recommendations. The application ensures secure handling of customer data, enabling fast and reliable home delivery while maintaining the quality of fresh bakes. This scalable system supports growing demand and offers a user-friendly ordering process.



**2.Database Configuration**: his setup allows multiple database connections to be efficiently managed, reducing the overhead of creating and closing connections for each request. By implementing connection pooling, the application enhances performance and scalability, ensuring a smooth user experience even during peak traffic times.

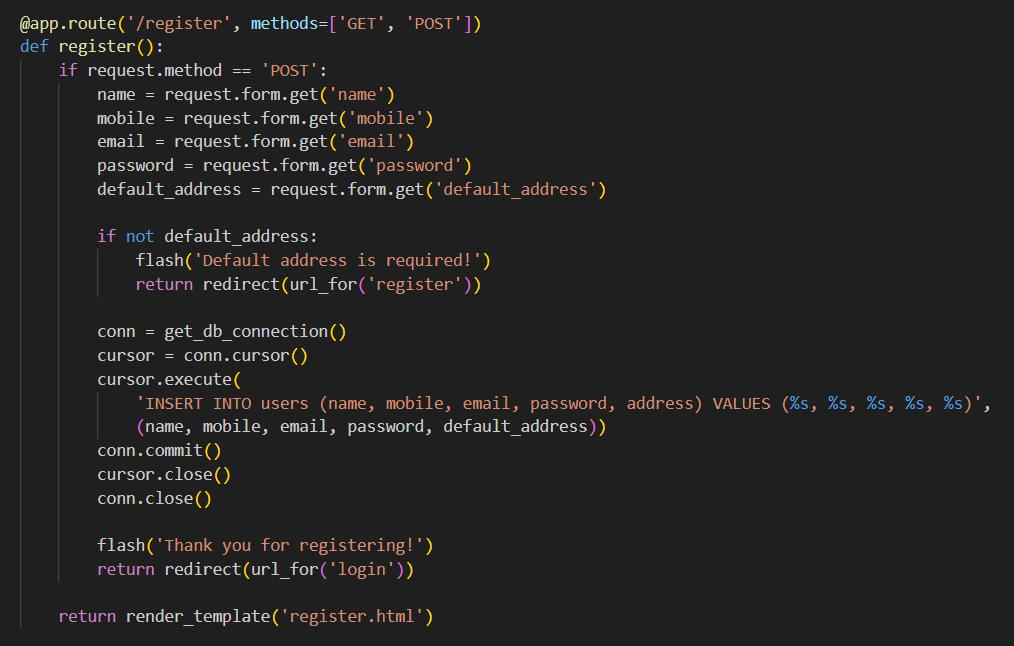


**3.Connection Pool**: Implementing MySQL connection pooling is essential for effectively managing multiple database connections. This technique allows the application to reuse existing connections, significantly reducing the overhead of establishing new ones. By utilizing connection pooling, the platform ensures optimal performance and responsiveness, particularly during high-demand periods. This approach leads to improved efficiency and a smoother experience for restaurants, enabling faster transaction processing and better resource utilization.



**4.Home Route**: Renders the home page template when the root URL is accessed.

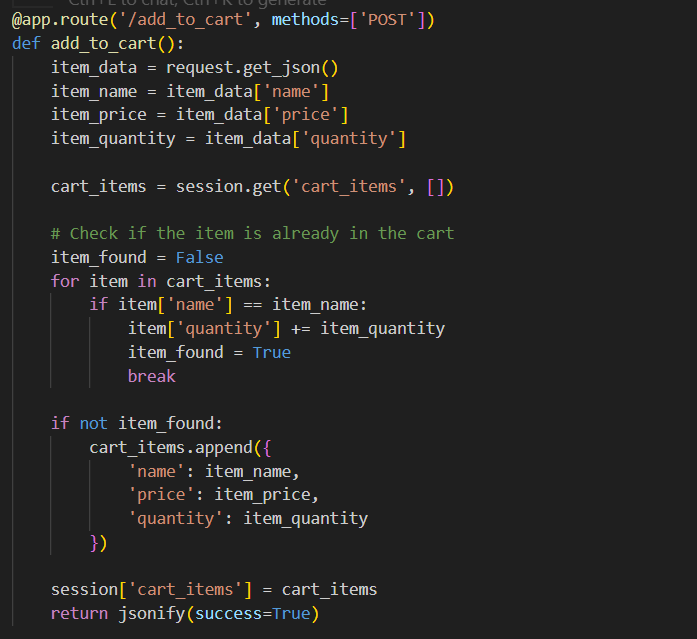
**5.signup Route (GET/POST)**: Handles user registration, inserts user data into the database



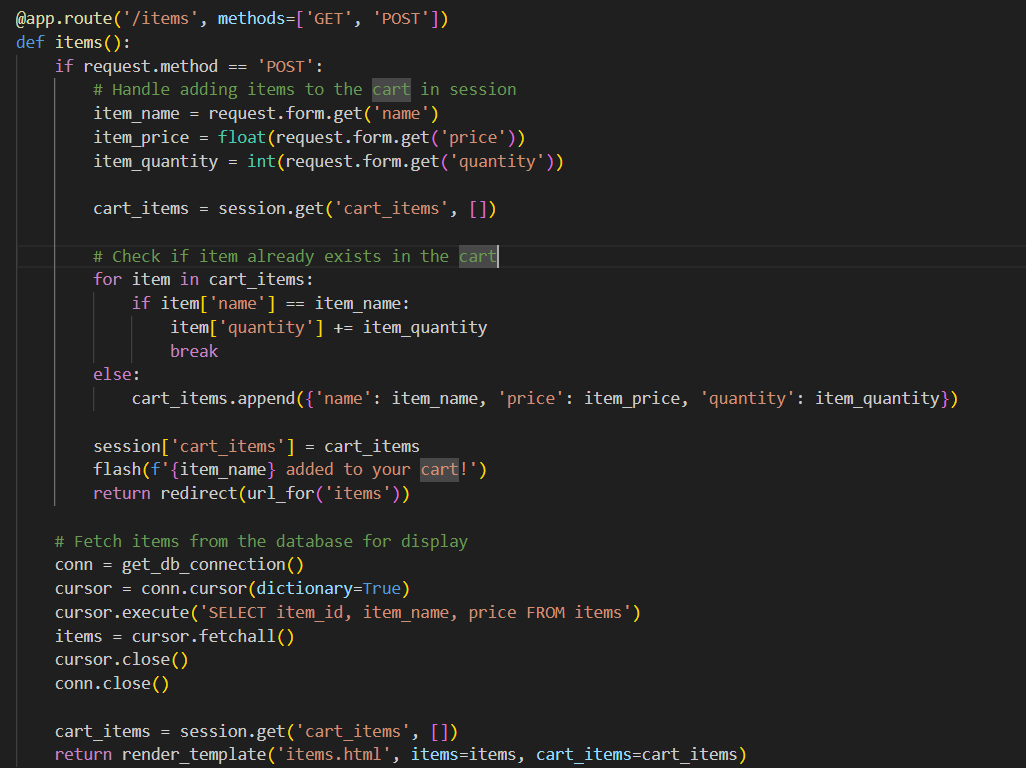
**6.Login Route (GET/POST)**enhances customer experience through secure user authentication. Upon successful login, a user session is created, granting access to order placement and delivery tracking. This secure management protects sensitive information, ensuring a seamless and confident interaction with the platform.



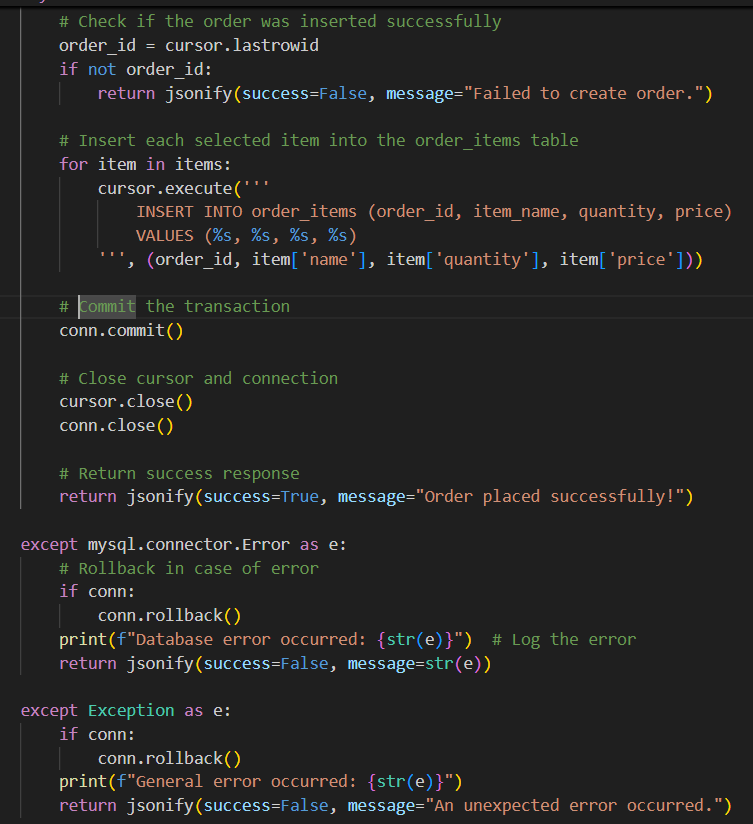
**7. Add to cart Route**: **Local Artisan Bakery: Powered by AWS for Fresh Bakes, Custom Orders, and Easy Home Delivery** enables customers to easily add their selected baked goods to their cart. This functionality streamlines the shopping experience, preparing the cart for a smooth checkout process. By efficiently updating the cart in real-time, the route ensures customers can effortlessly review their choices before finalizing their orders, enhancing overall satisfaction and convenience.



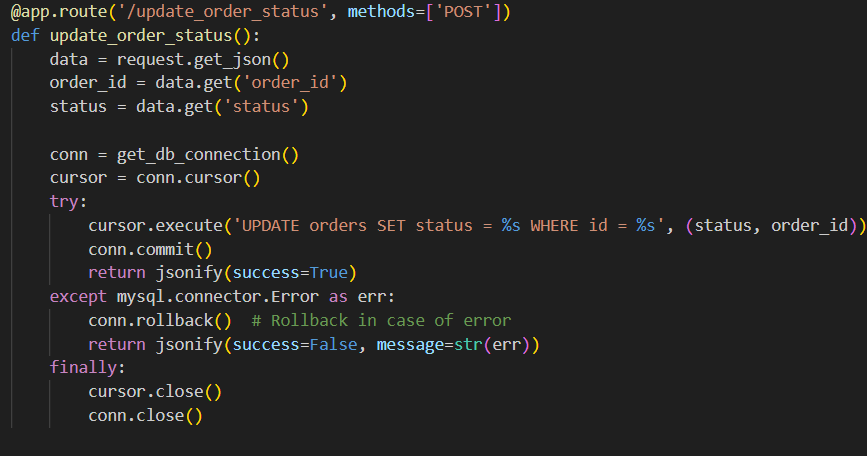
**8.Cart Page:** the customer's selected food items, providing a comprehensive overview of their order. Users can review their selections, make modifications such as adjusting quantities or removing items, and ensure everything is correct before proceeding to checkout. This functionality enhances the user experience by giving customers control over their orders, facilitating a smooth and efficient purchasing process.



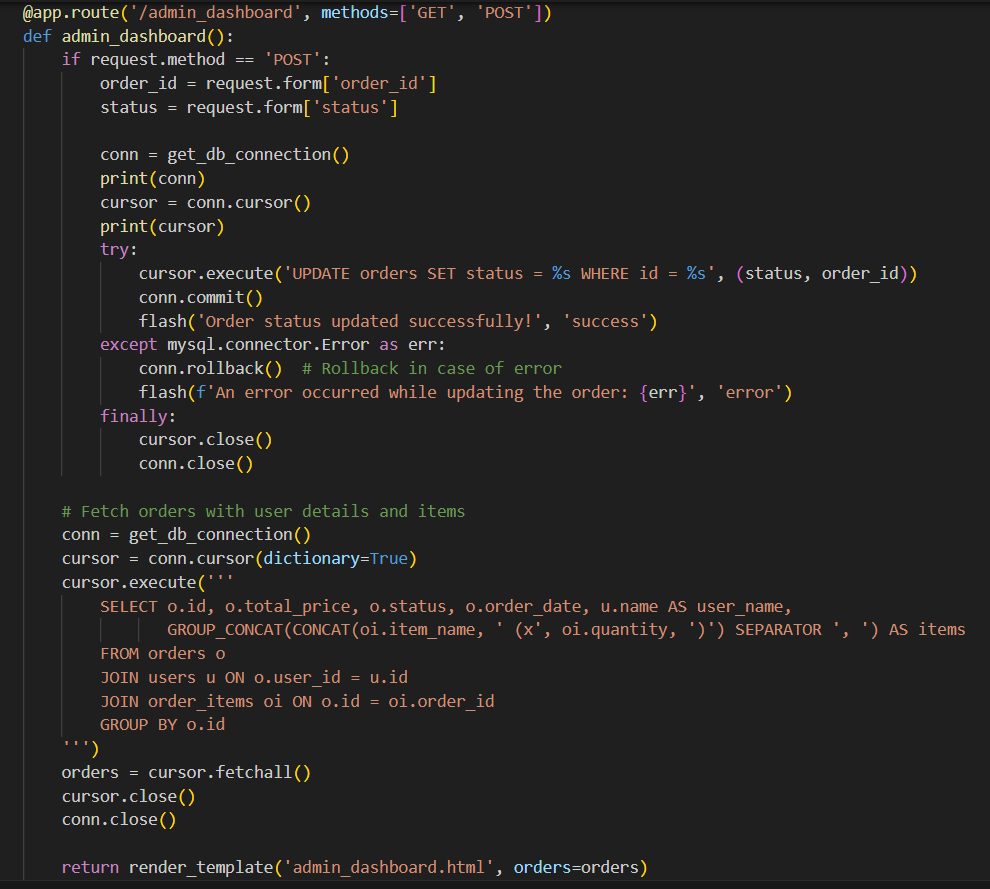
**9.Placing cart Items page:** Customer's order by finalizing the selected items in their cart. Users are prompted to proceed to payment or enter delivery details, ensuring that all necessary information is collected before completing the transaction. This page enhances the user experience by providing a clear summary of the order, facilitating a seamless transition from selection to payment, and ensuring a smooth checkout process.



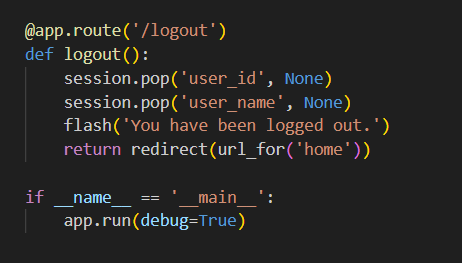
**10.Update order status route**:message “Thanks for Booking!” upon successful order placement. This route provides users with immediate feedback, confirming that their order has been received. After displaying the message, it redirects customers to the homepage, ensuring a smooth navigation experience and allowing them to continue exploring the platform.



**11**. **dashboard.html:** Provides an overview of customer orders, delivery status, and real-time updates for managing



**12.Logout route :** securely ends the user's session and redirects them to the homepage, ensuring proper session management.



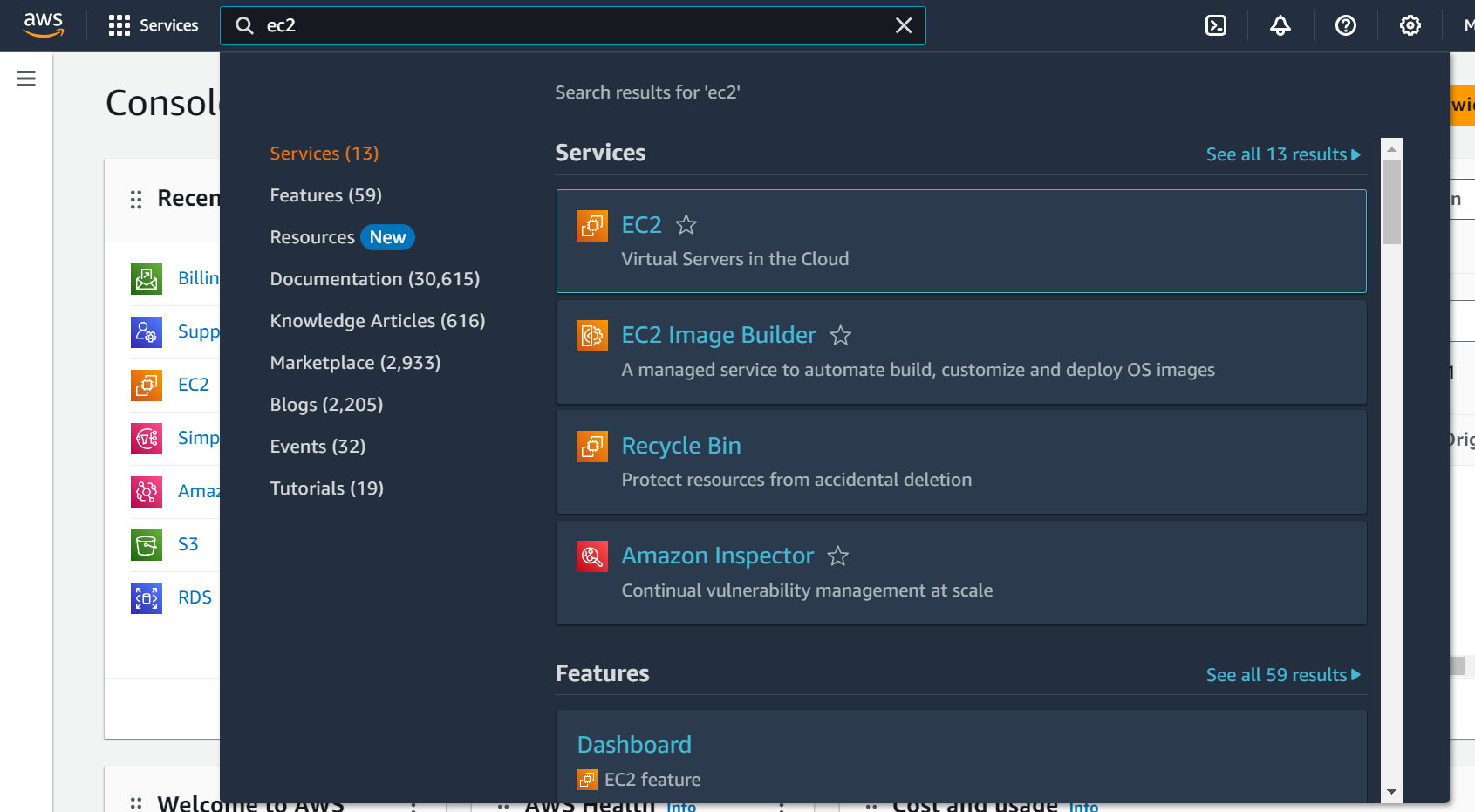
**Database Queries:** Uses SQL queries to interact with MySQL RDS for menu items, customer orders, and delivery details.

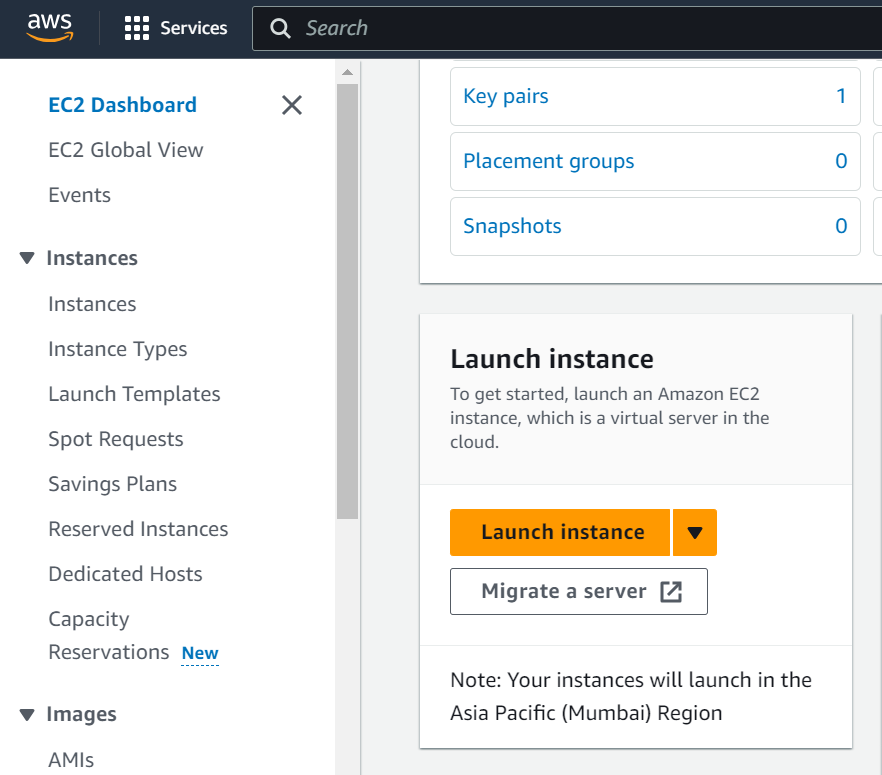
**Session Management:** Uses Flask sessions to store customer preferences and cart details for seamless navigation.

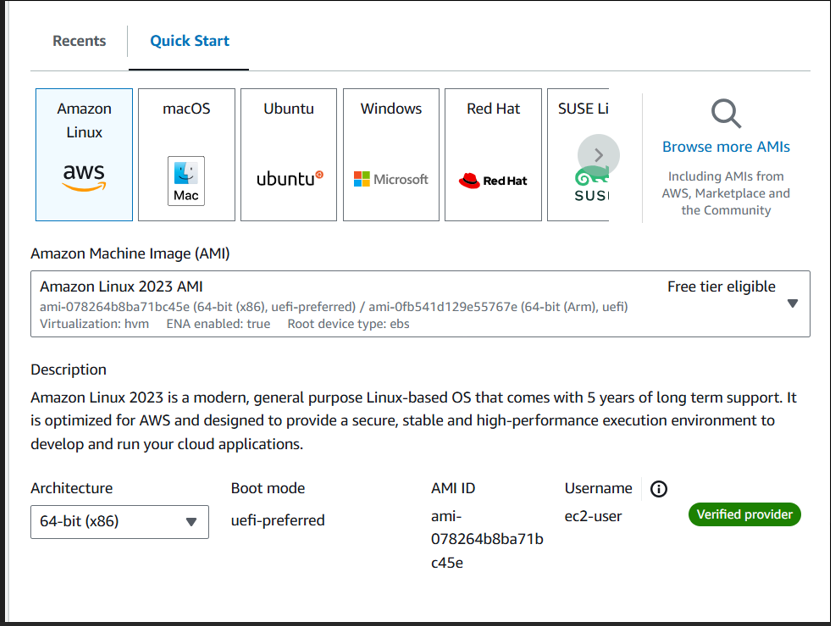
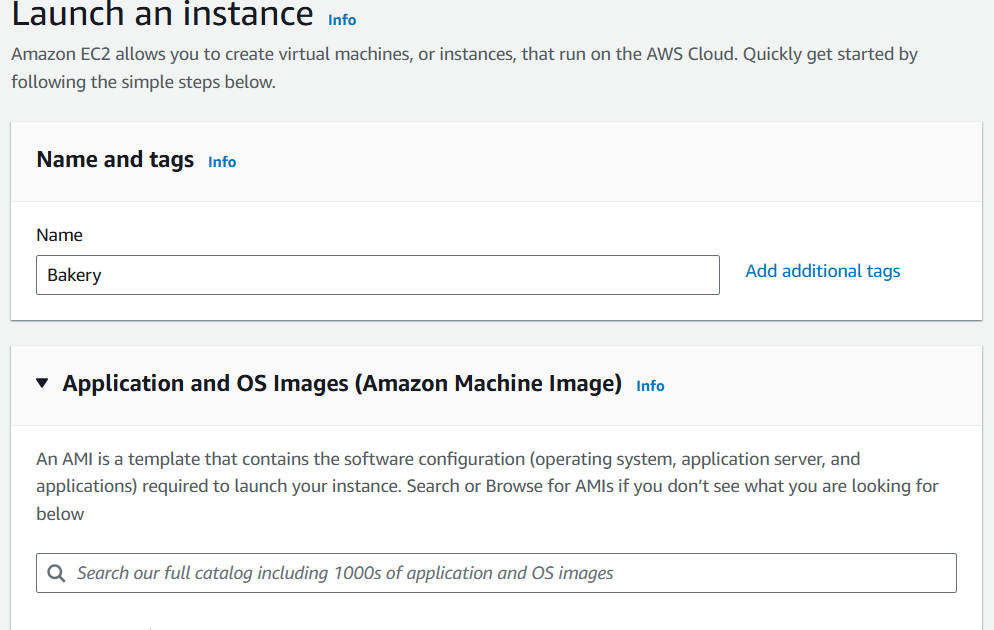
**Flash Messages:** Provides user feedback through flash messages for login, registration, order confirmation, and status updates**.**

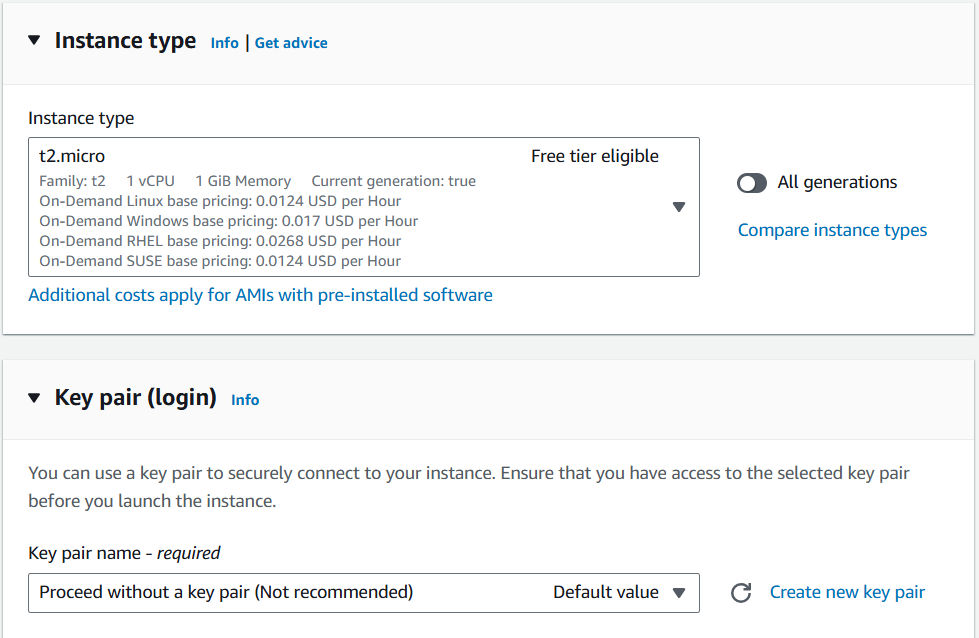
**Menu Data Fetching:** Retrieves food items, prices, and availability from the database to display on the ordering interface.

**Milestone 4: EC2 Instance Setup**

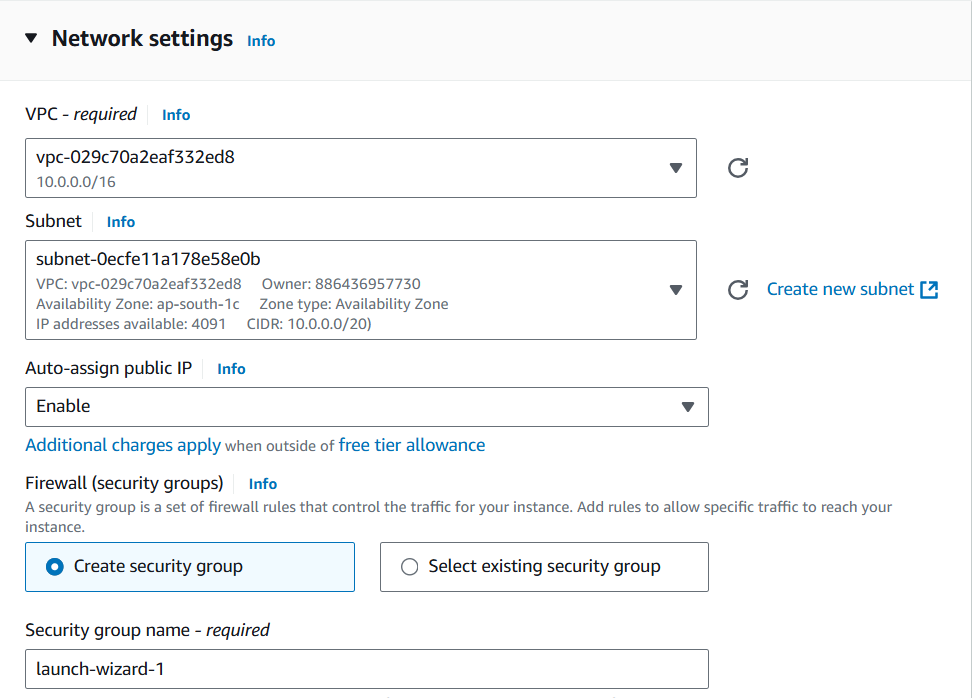
* **Activity 4.1: Launch EC2 Instance**
  + Choose a Linux-based EC2 instance from the AWS Console to host the FreshBasket application.





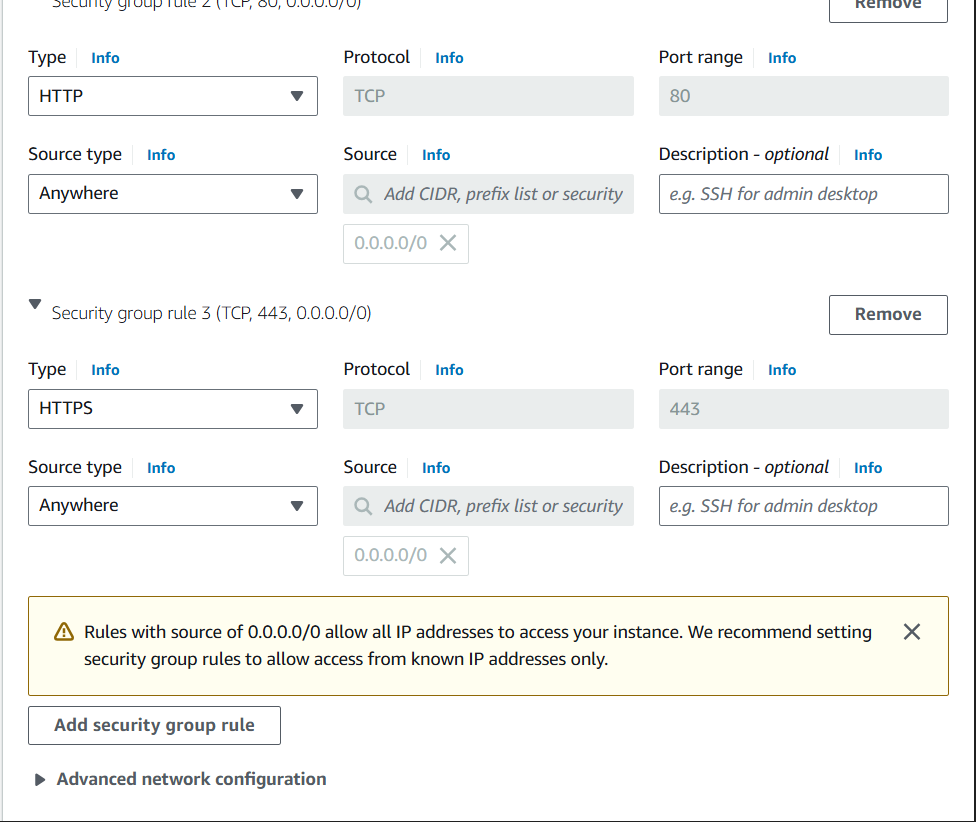
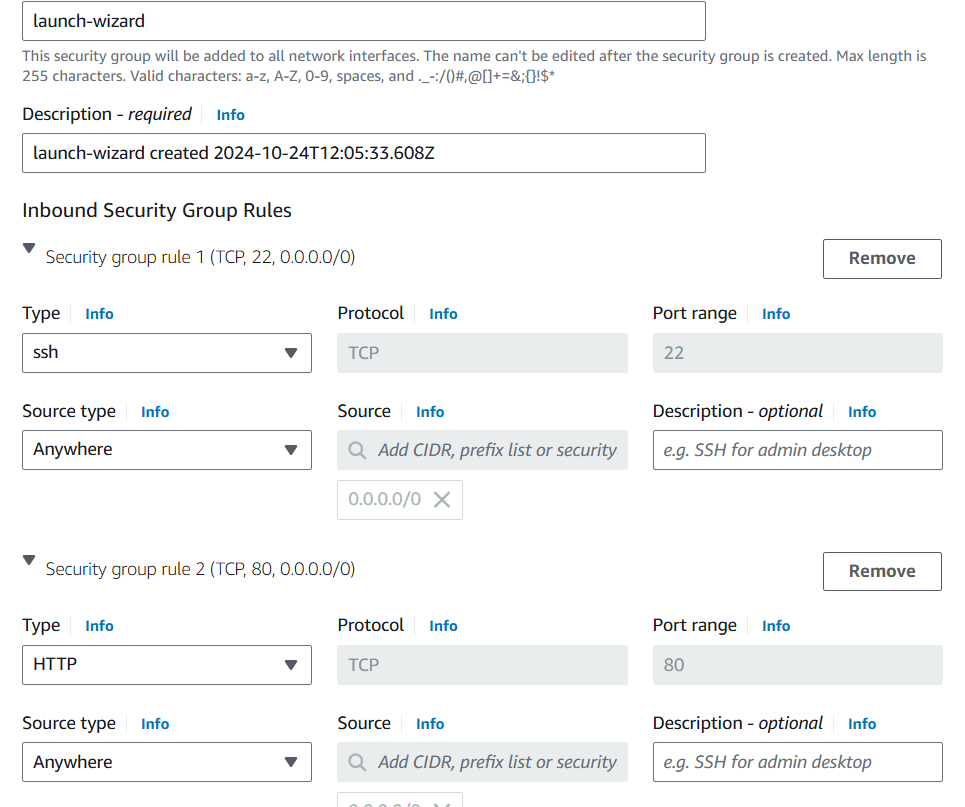


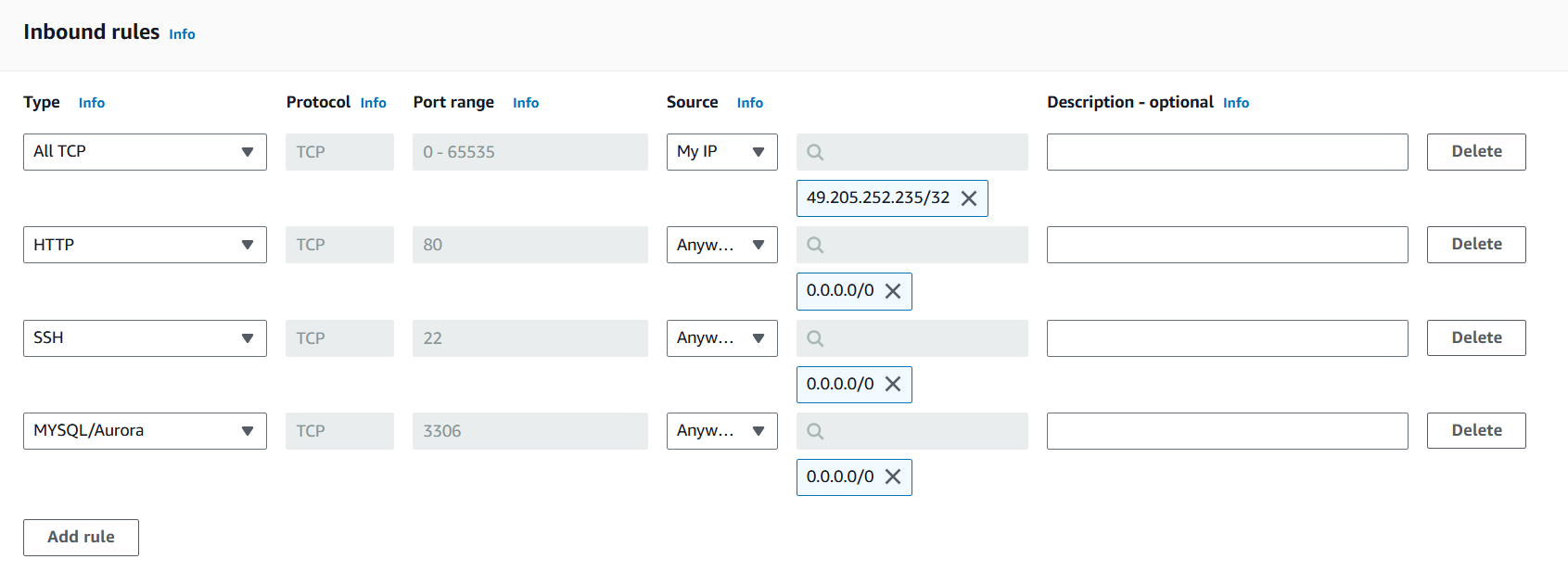
* **Activity 4.2: Configure Network Settings**
  + Set up the security group to allow HTTP, HTTPS, and SSH traffic.



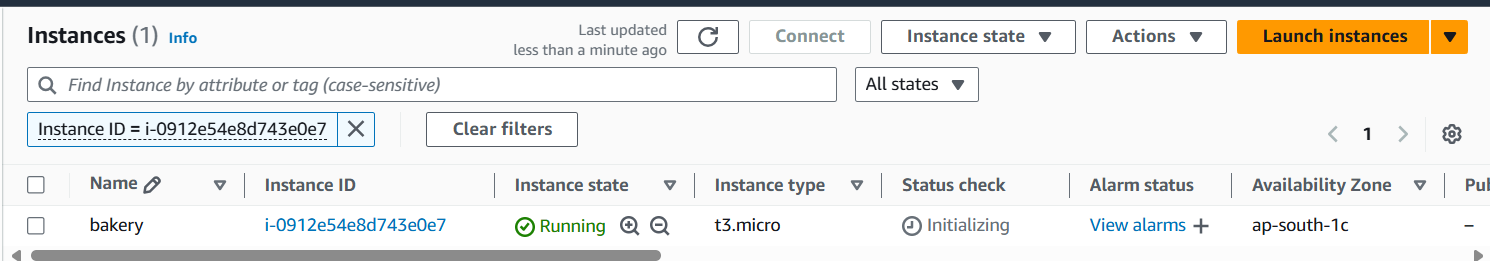
* Create and download the key pair for SSH access.

**Setting up Inbound and Outbound rules**





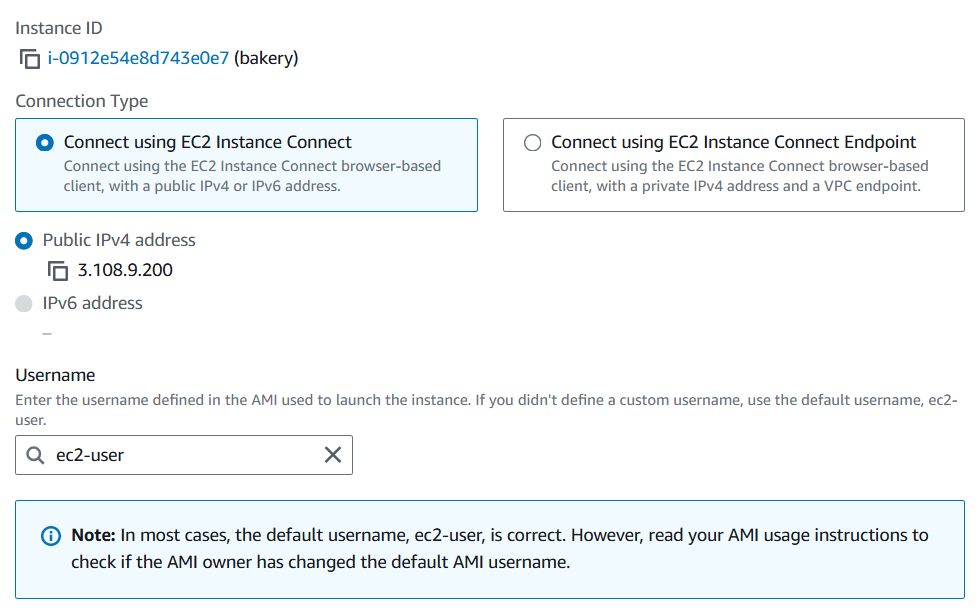
* Add Type : HTTP > Source : Anywhere
* Add Type : HTTPS > Source : Anywhere



**Milestone 5: Deployment on EC2**

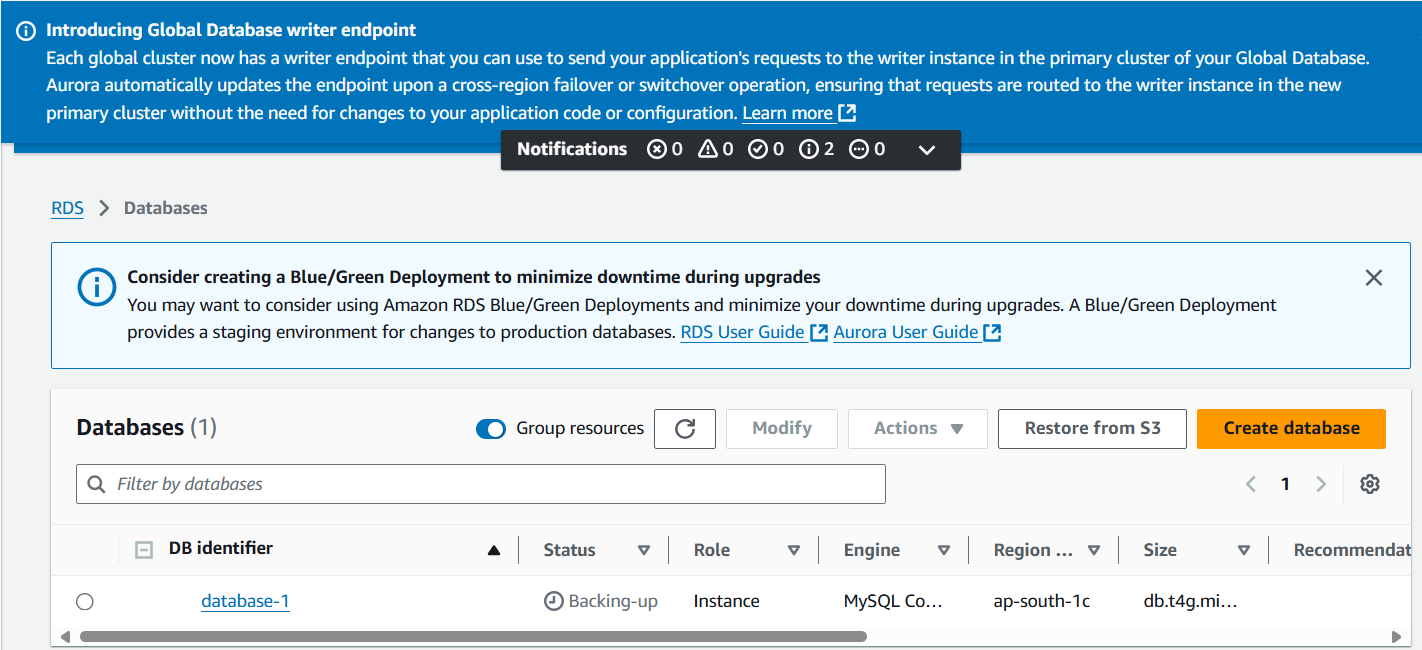
**Activity 5.1: Deploy to EC2**

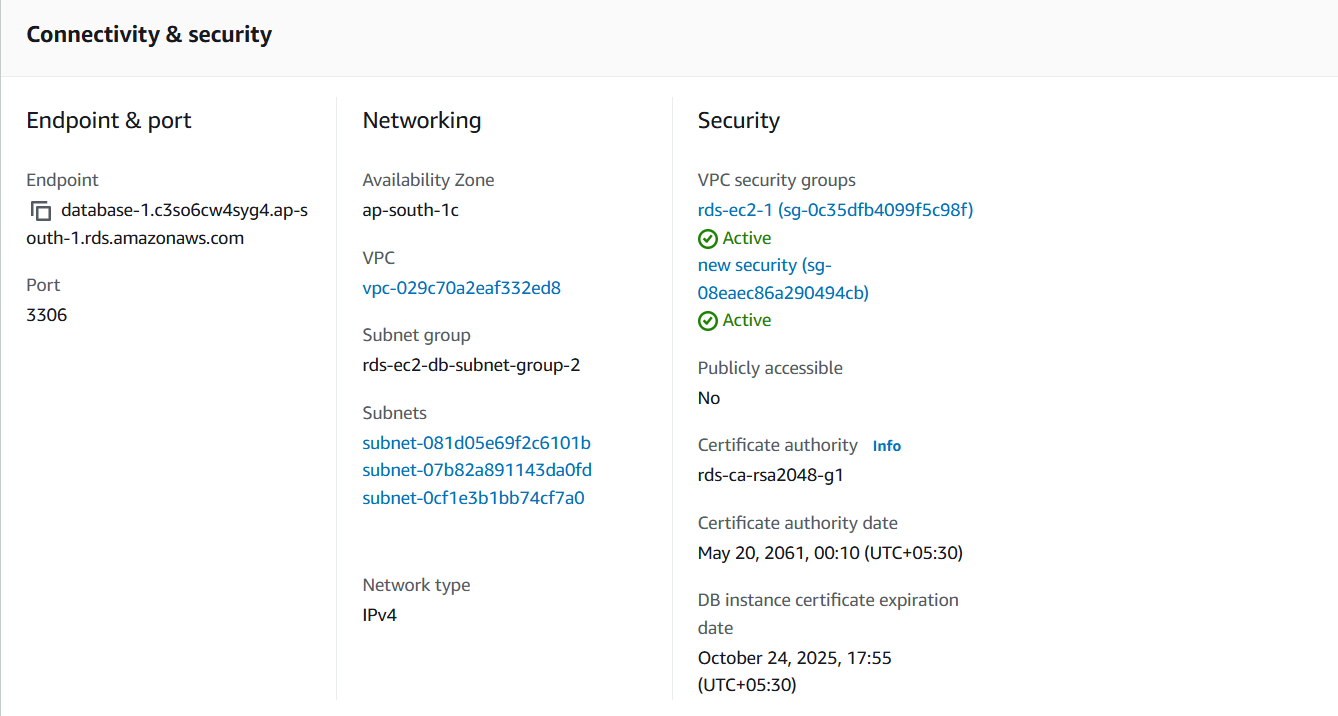
1. Connect EC2 terminal.
2. Set up any necessary environment variables, including database connection strings.
3. Configure the web server to serve your application.
4. Start your application and ensure it's accessible via the EC2 instance's public IP or domain.



**Activity 5.2: Deploy to RDS**

1. **Connect to RDS Instance**Access your Amazon RDS instance through the AWS Management Console or by using an RDS client.
2. **Set up Database Configuration**Configure environment variables in your application to include RDS database connection strings (host, port, username, password).
3. **Initialize Database Schema**Run your database schema setup or migration scripts to create the necessary tables and relations on the RDS instance.
4. **Test Database Connectivity**Verify the connection between your EC2 instance and RDS by running database queries through your application.
5. **Monitor RDS Instance**Set up monitoring and logging for your RDS instance using AWS CloudWatch to track performance and errors.

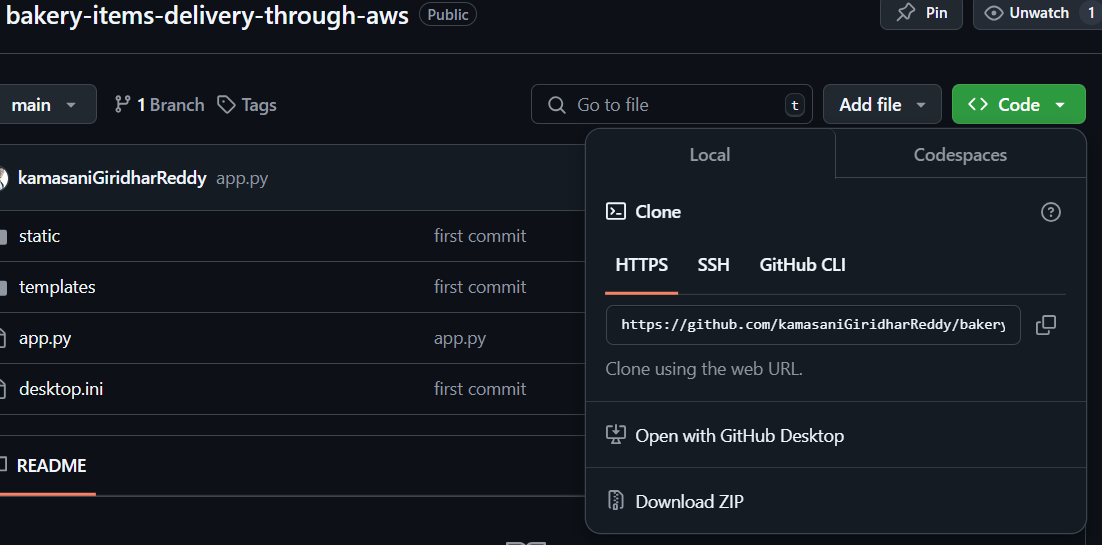




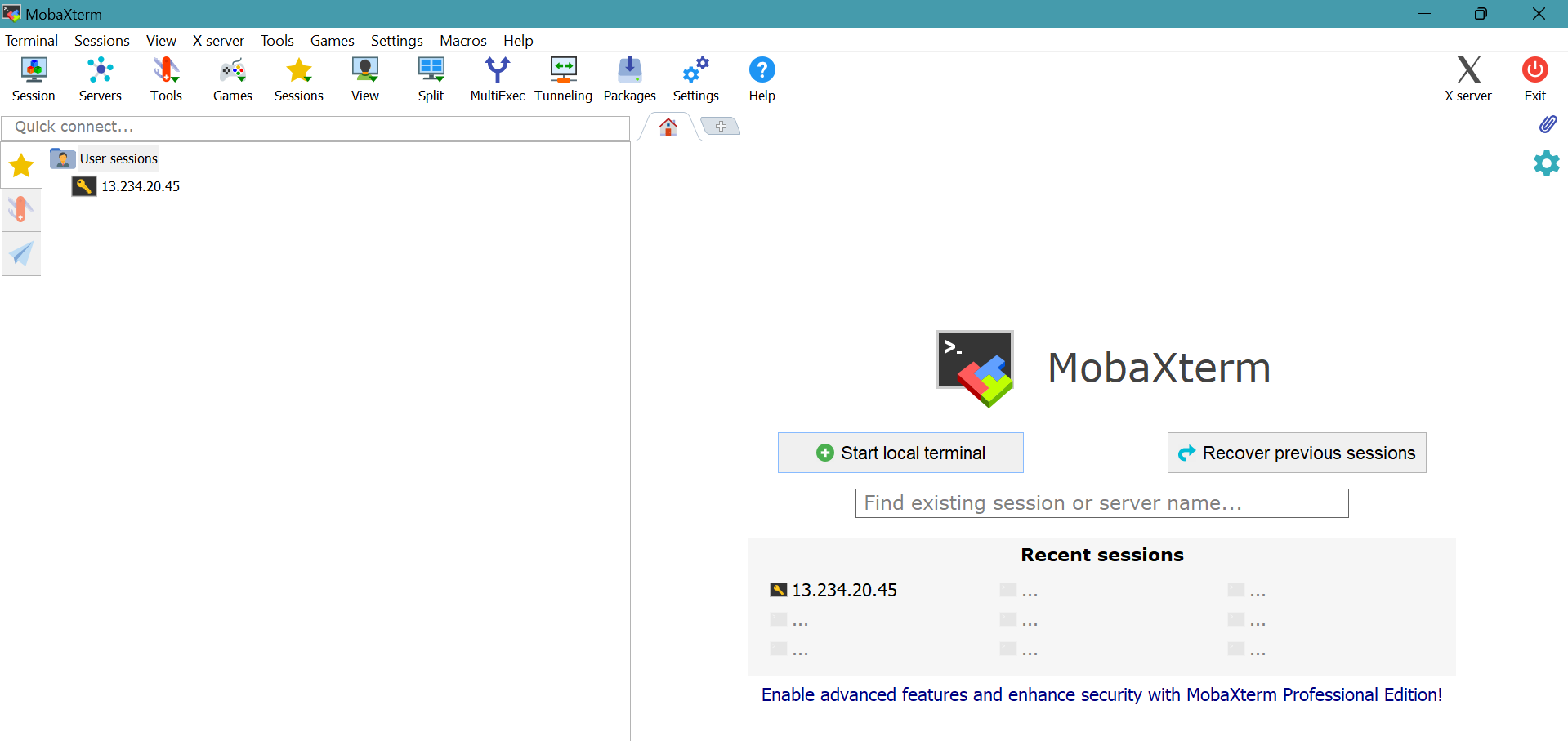
**Activity 5.3 : Launch Flask Application**

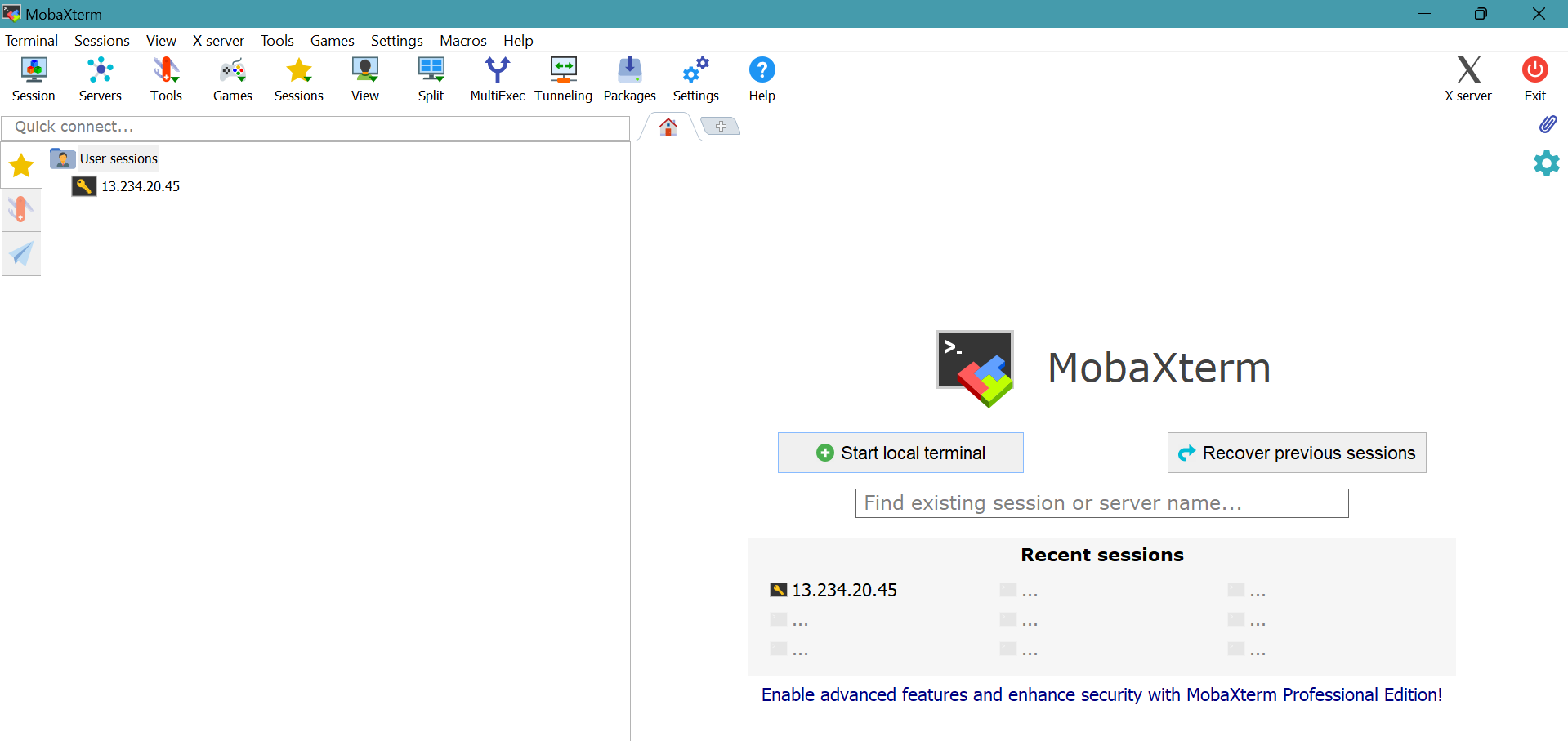
* + Run the Flask app on the EC2 instance through the SSH session to serve the order booking front-end.

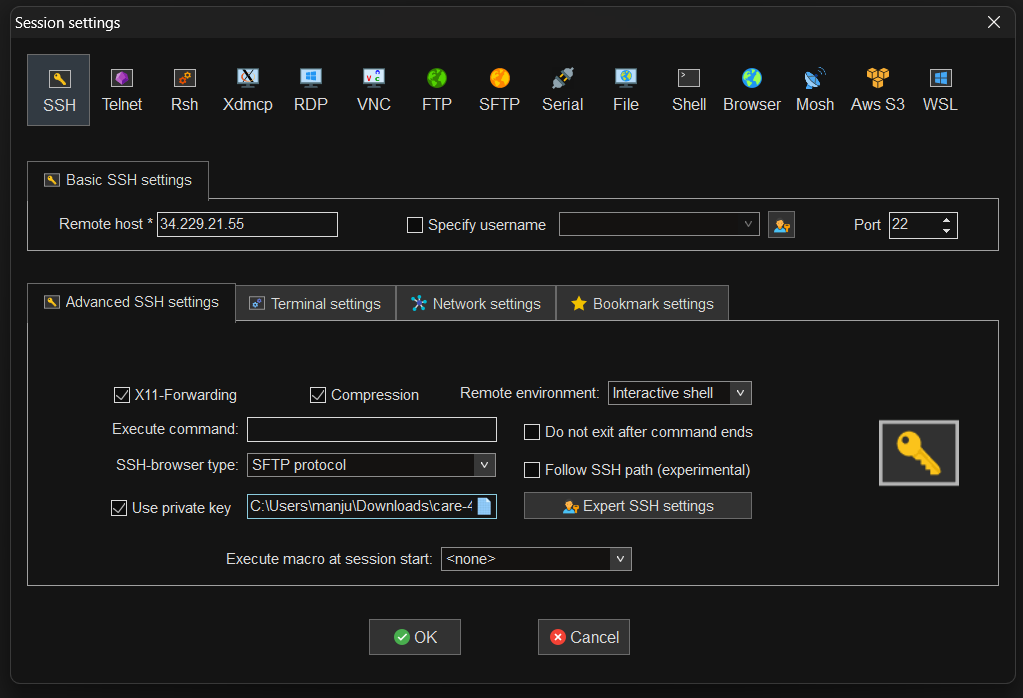
1. Transfer your application code to the EC2 instance.
2. Set up any necessary environment variables, including database connection strings.
3. Configure the web server to serve your application.
4. Start your application and ensure it's accessible via the EC2 instance's public IP or domain.
5. Run the below commands on ec2 terminal
6. sudo yum update -y
7. sudo yum install python3 -y
8. sudo pip3 install virtualenv
9. python3 -m venv venv
10. source venv/bin/activate
11. pip install flask
12. git clone <https://github.com/your-repo/your-flask-app.git>
13. cd your-flask-app
14. python3 app.py
15. <https://github.com/kamasaniGiridharReddy/bakery-items-delivery-through-aws.git>



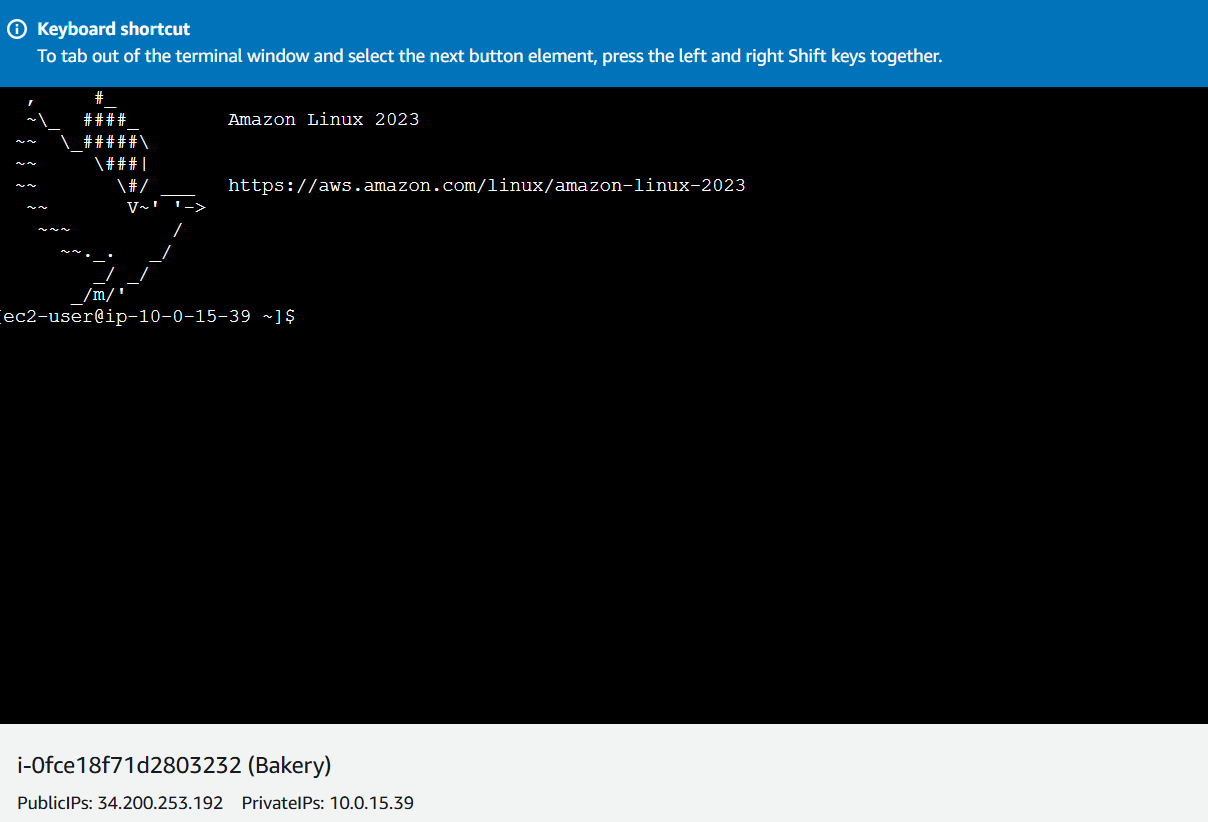
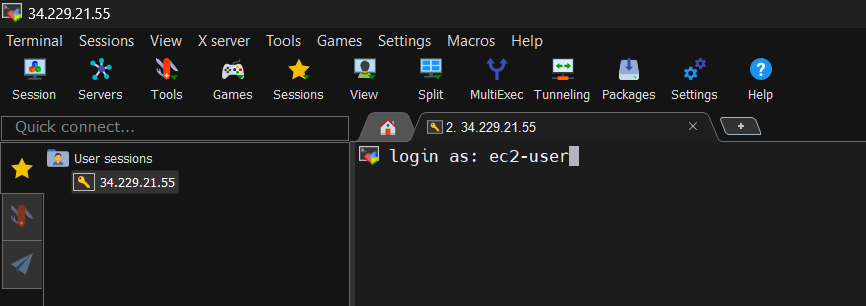
**Milestone 5(b): MobaXterm Setup and SSH Access**

* **Activity 5.b.1: Install and Configure MobaXterm**
  + Download and install MobaXterm on your local machine.
  + Establish an SSH session with the EC2 instance by uploading the .pem key file.
  + 





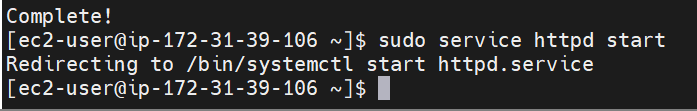
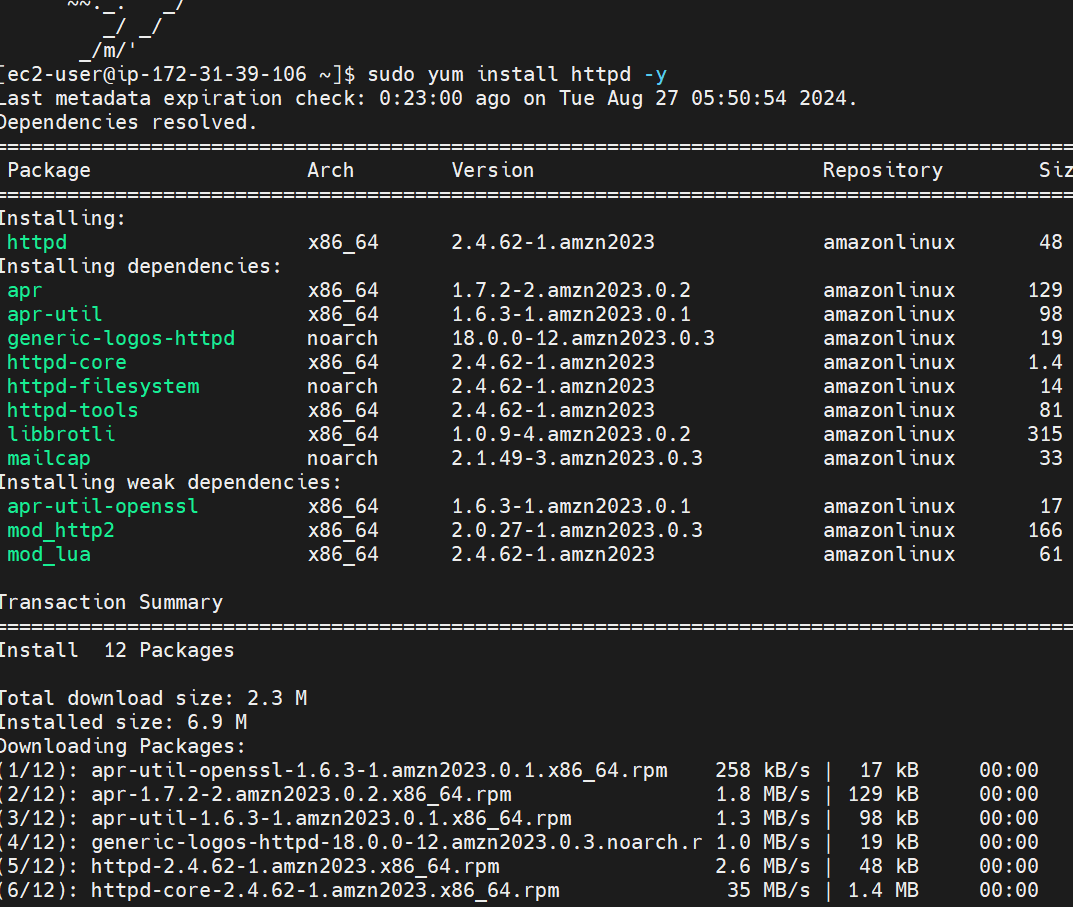
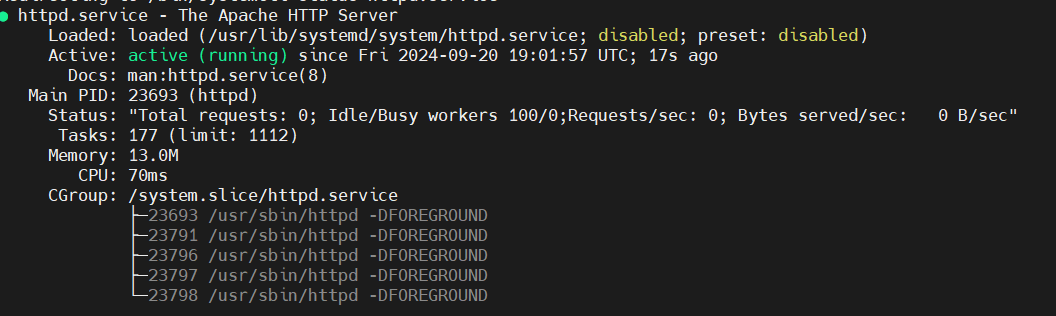
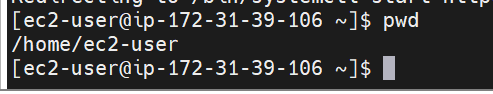
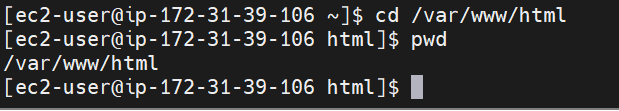
* + **Activity 5.b.2: Login into the SSH Session**



**Update and Install Web Server Software**

* Update package lists with sudo apt-get update (Ubuntu) or sudo yum update (Amazon Linux).

Install Apache or Nginx:

* For Apache: sudo apt-get install apache2 (Ubuntu) or sudo yum install httpd (Amazon Linux).
* For Nginx: sudo apt-get install nginx (Ubuntu) or sudo yum install nginx (Amazon Linux).
* To check the status of the service : sudo service httpd status
* 
* To know the Present working Directory 
* Navigating to html folder.
* 

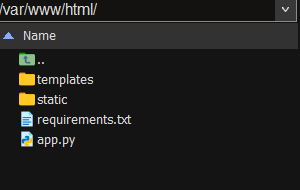
**Upload Website Files Using MobaXterm**

* Use MobaXterm’s SFTP functionality to transfer website files to the EC2 instance.
* Navigate to the /var/www/html directory (or the relevant directory for Nginx) and upload the files.

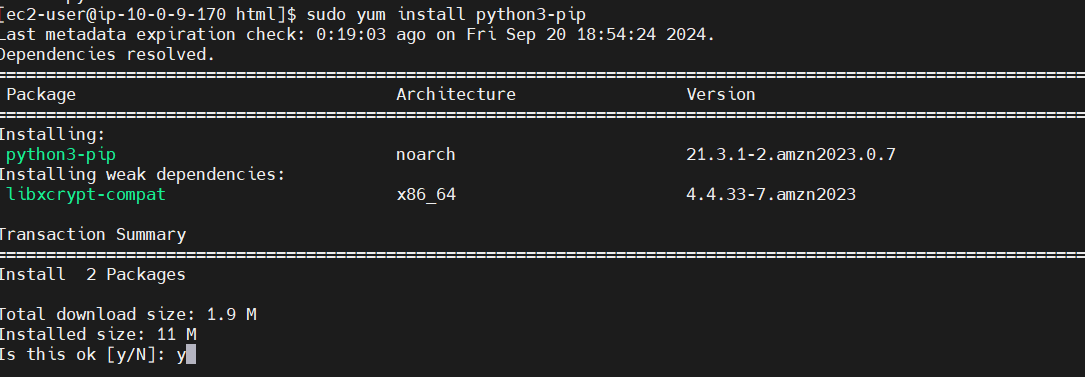
**To grant permission to accept the files into html folder.**

****

**Upload the HTML CSS and JS files**

****

**Install python and pip to run the flask application**

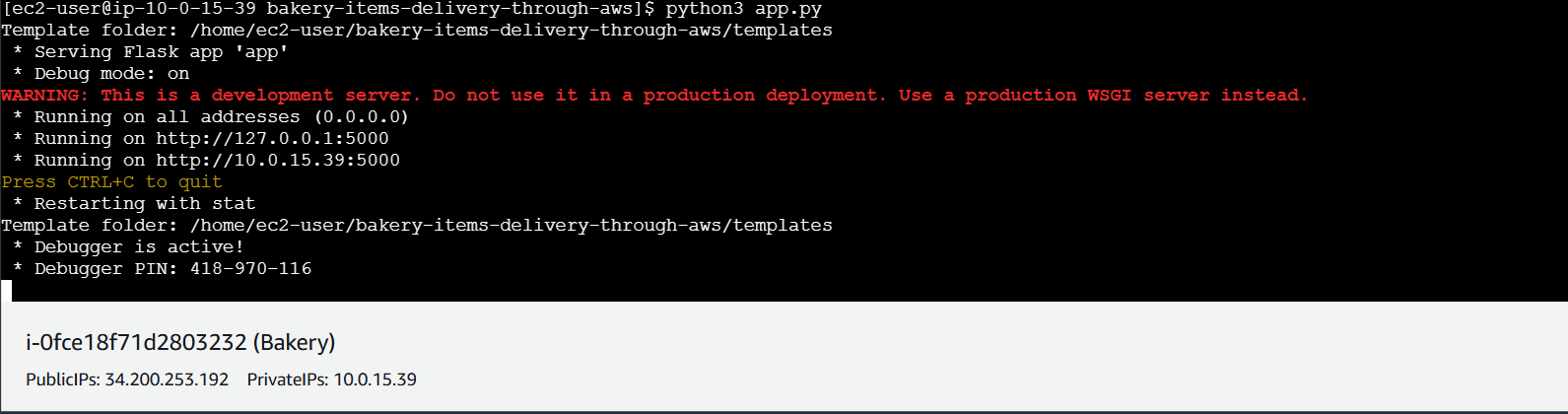
****

Install Required Libraries:

Pip install flask

pip install boto3 mysql-connector-python

**Milestone 6: Testing and Deployment**

* **Activity 6.1: Functional Testing**
  + Test the FreshBasket application for functionality, including database interactions and frontend features.
  + Run the Flask app **python3 app.py**
* It will give you the link 

**Access the website through:**

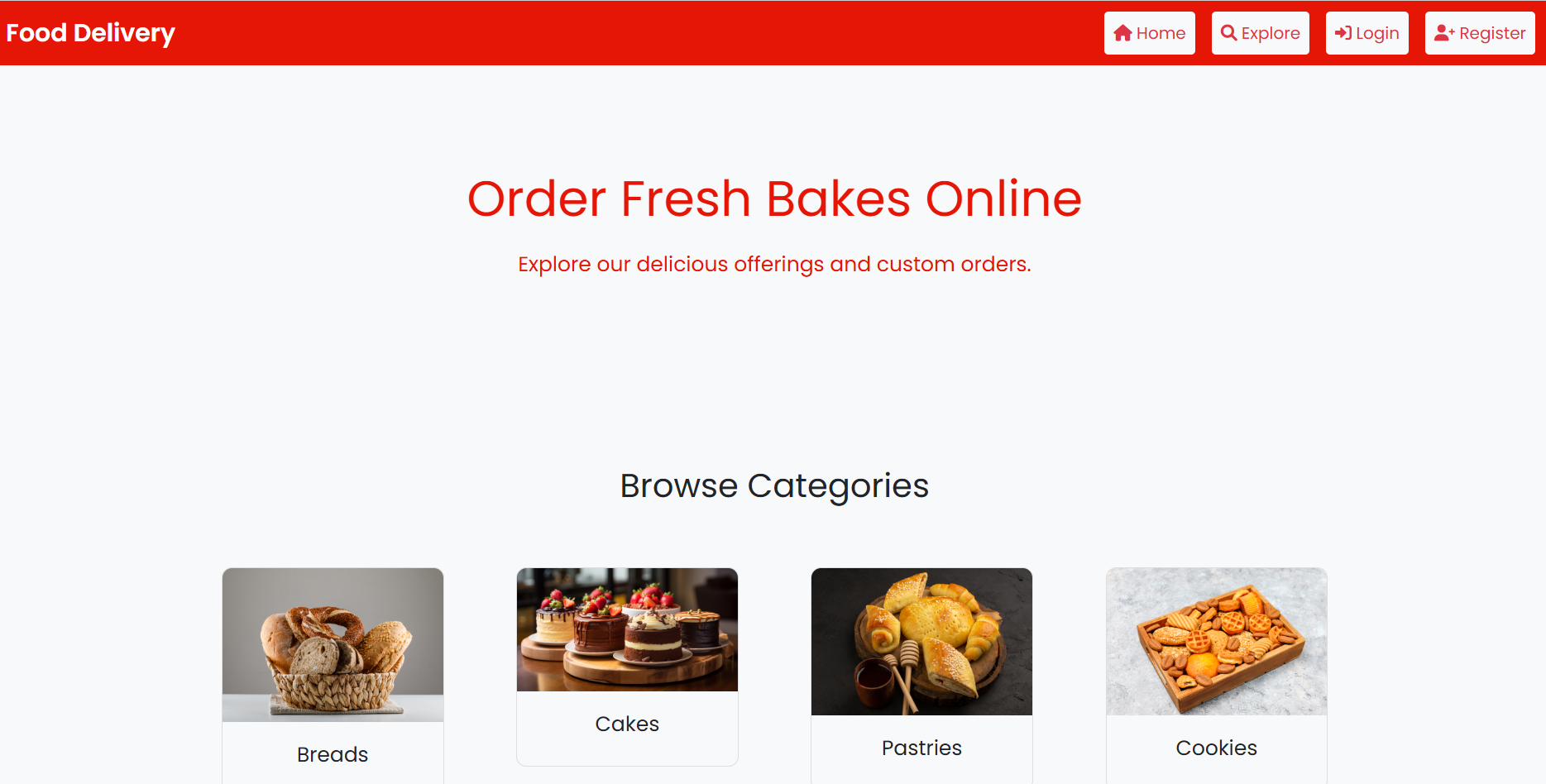
**PublicIPs:**[**http://127.0.0.1:5000**](http://127.0.0.1:5000)

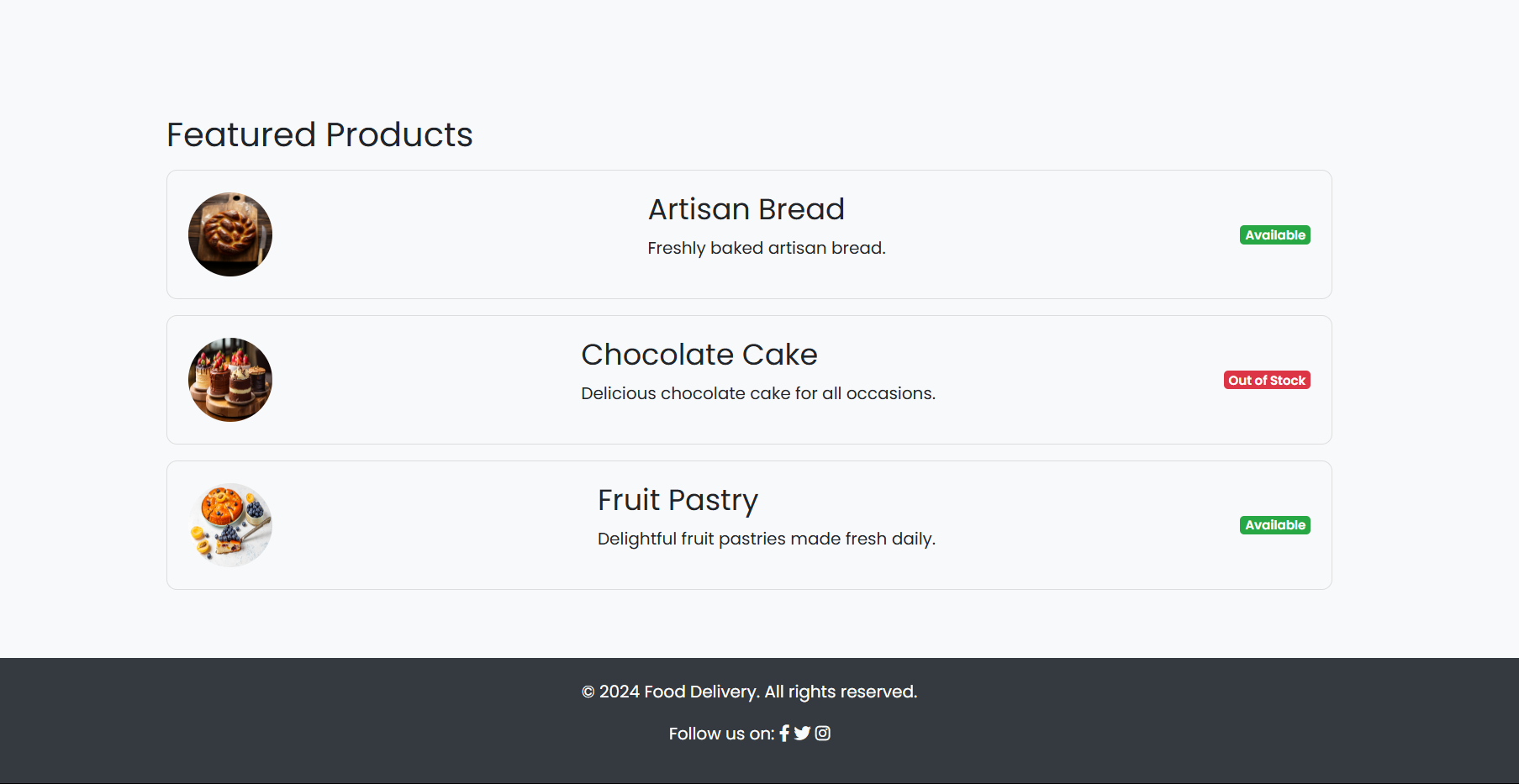
[**http://10.0.15.39:5000**](http://10.0.15.39:5000)

* **Activity 6.2: Deployment**
  + Deploy the application in a production environment, ensuring high availability and performance.

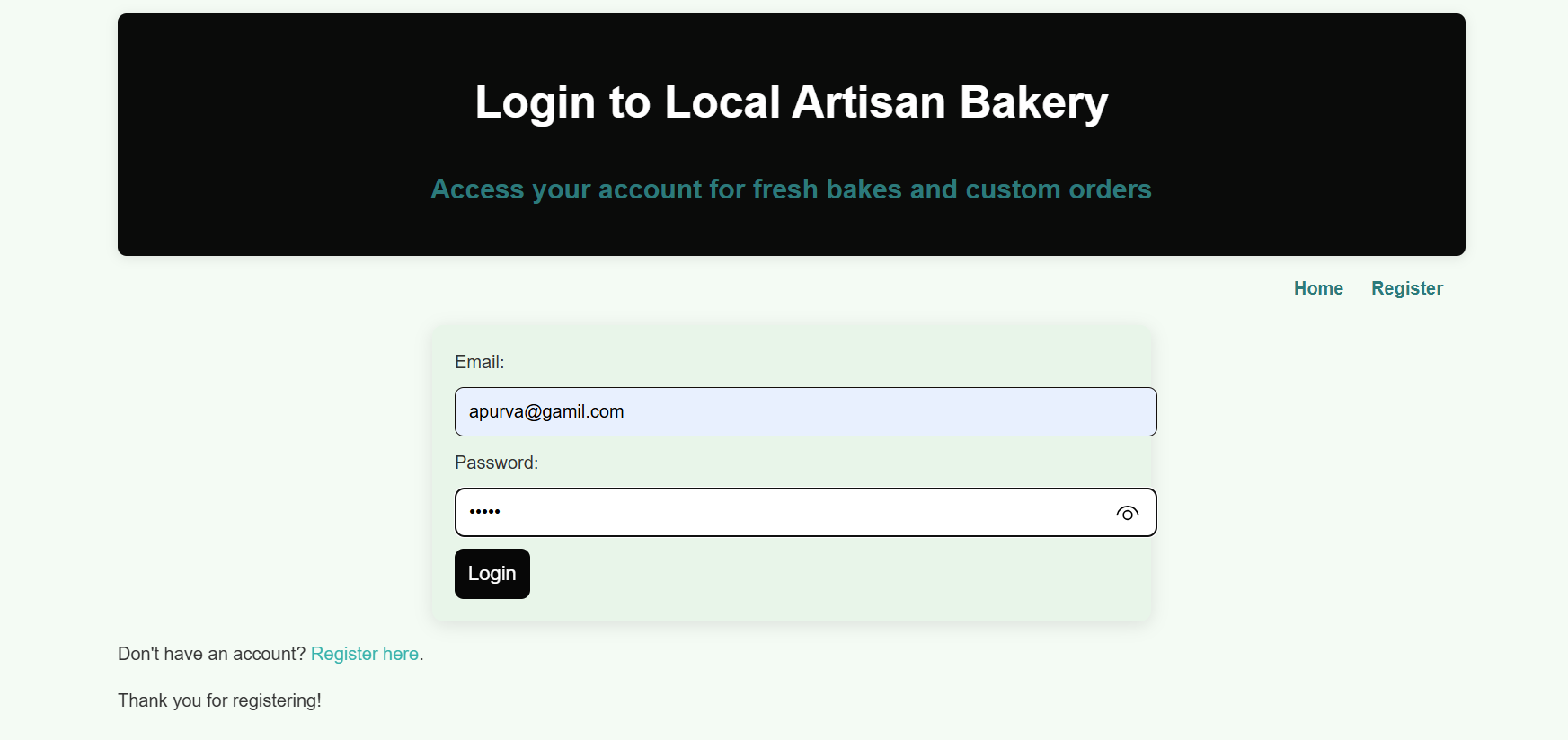
Click on the link above and it will take you to the webpage:

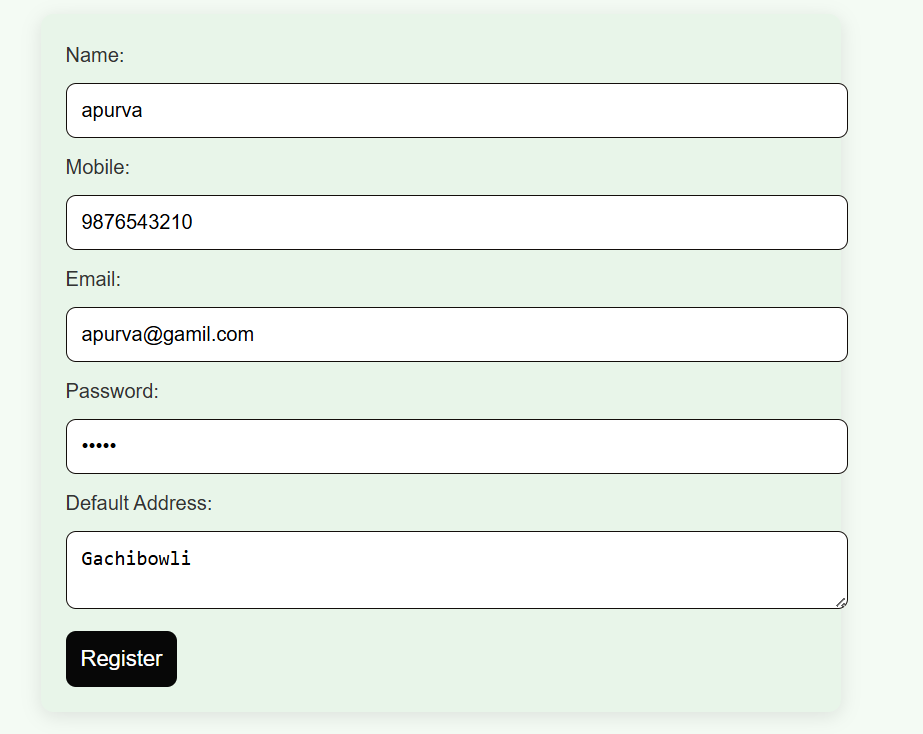
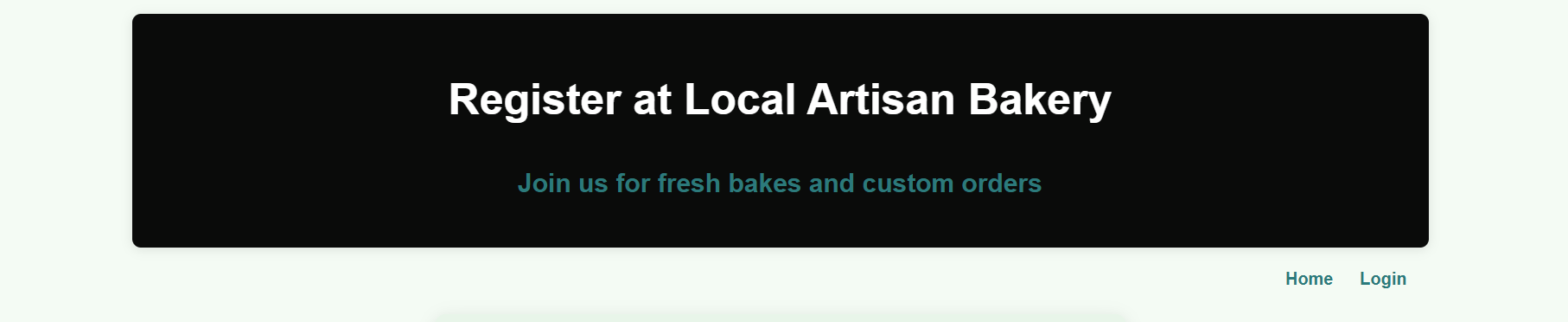
**Home:**

****

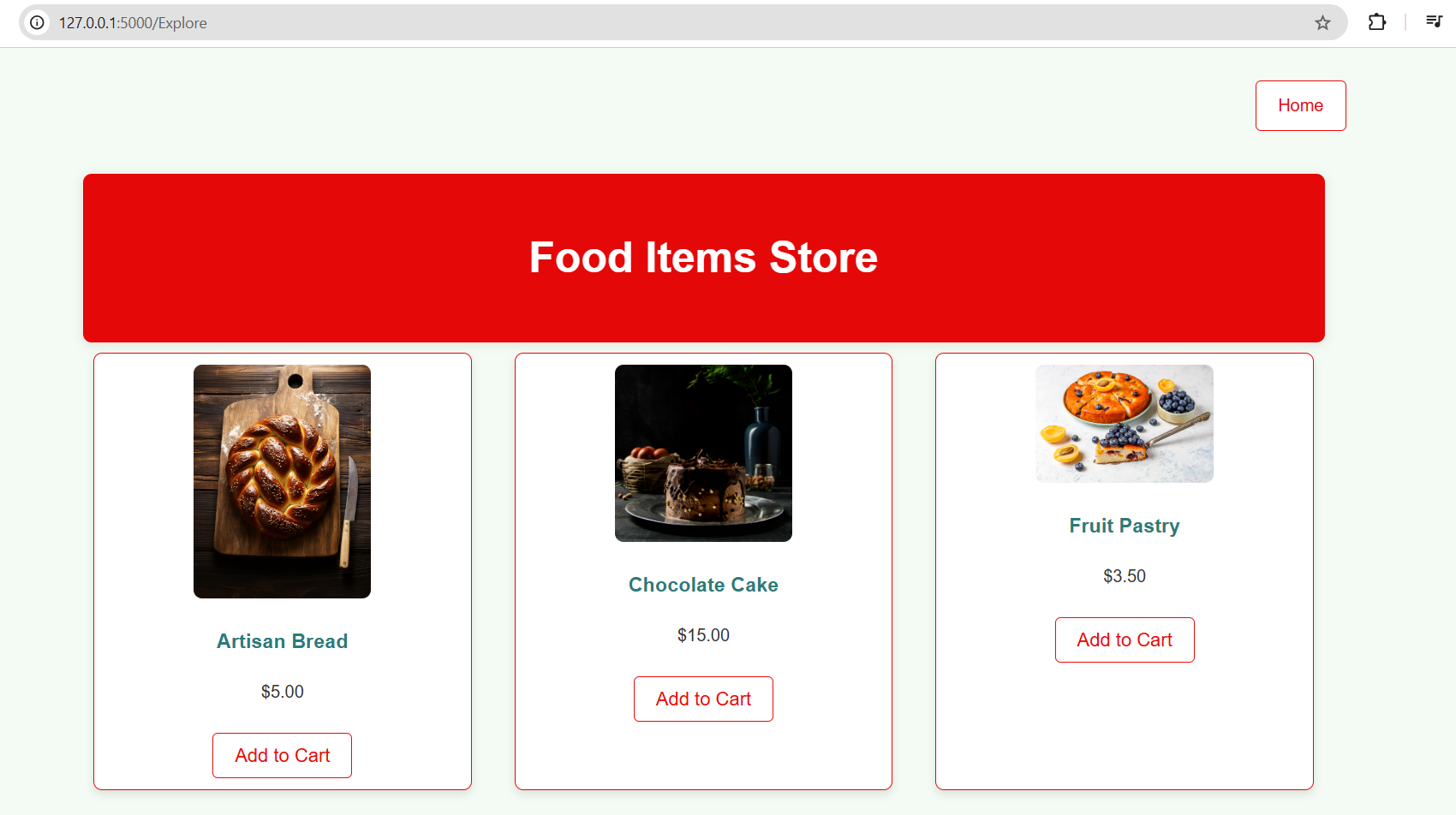
****

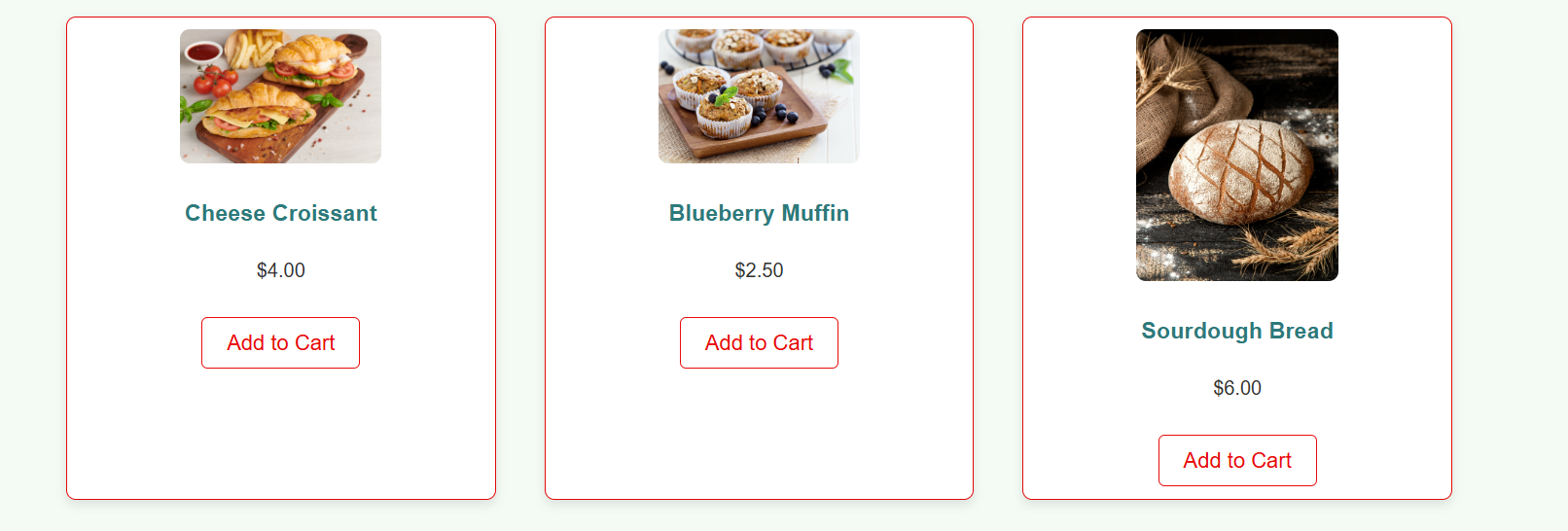
**Login page:**

****

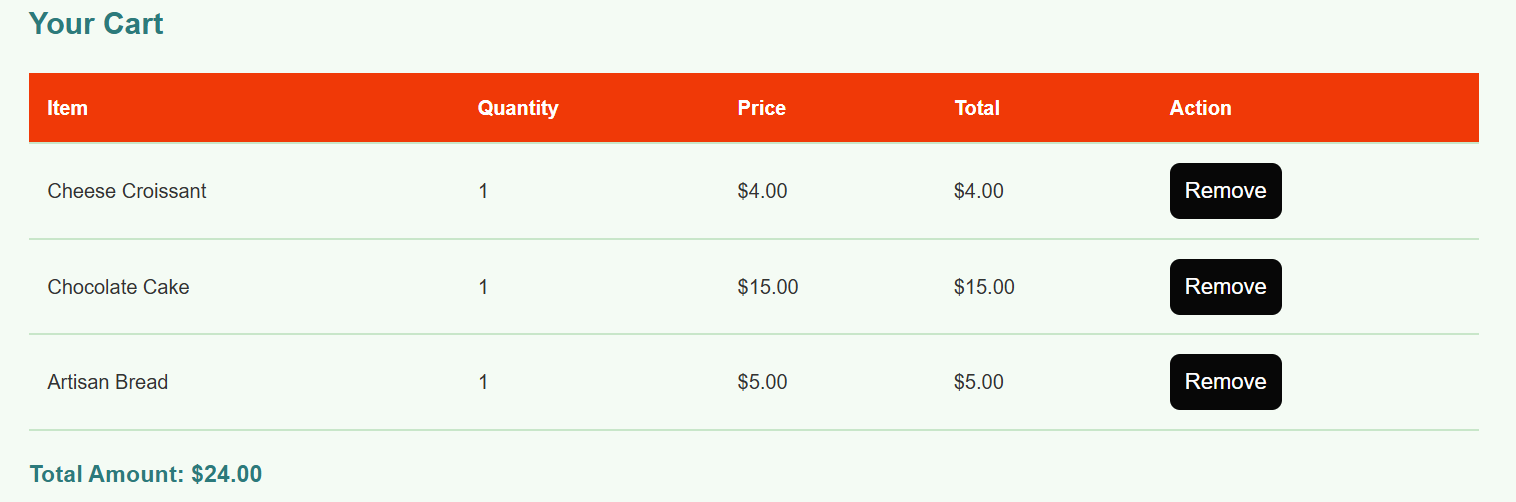
**SignUp page:   
 **

**Food Items Page:**

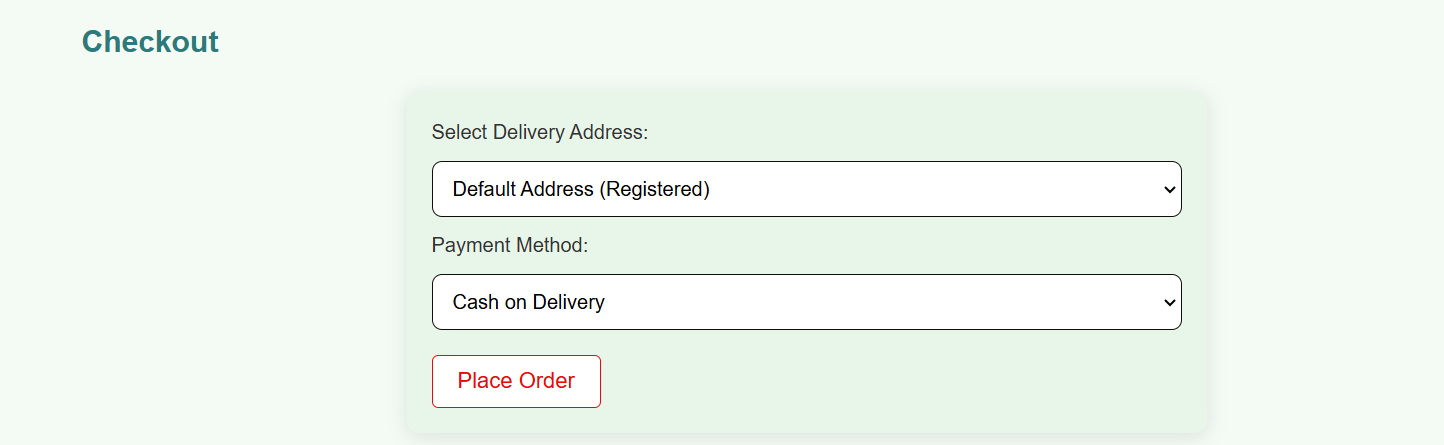
****

****

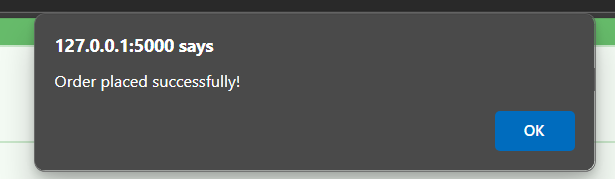
**Cart Page:**

****

**Payment page:**

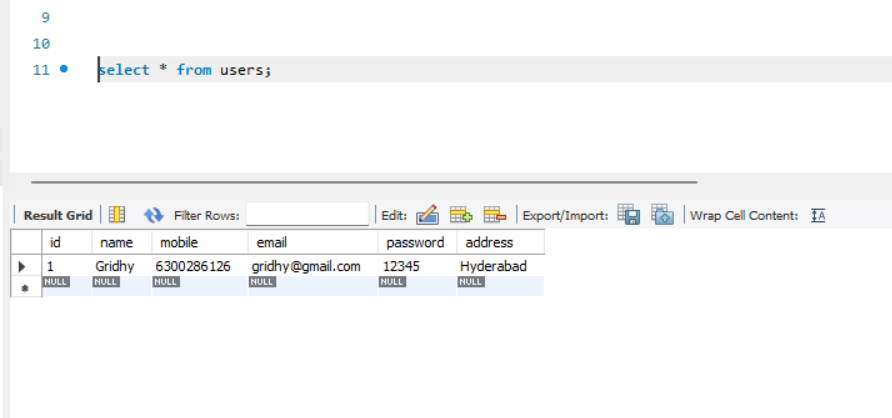
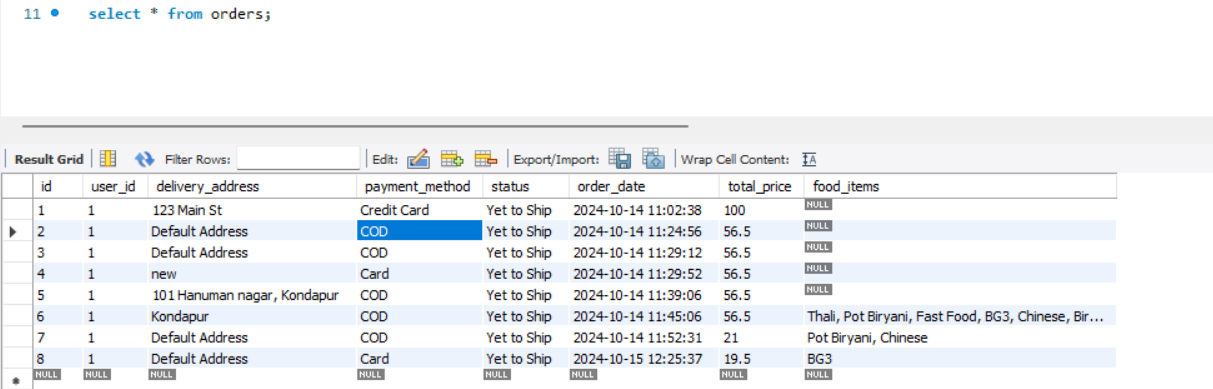
****

**Order Placed Pop up notification.**

****

* **Activity 6.3: Check the updations in the mysql database.**

**MySql Database updations :**

1. **Users table :**
2. **Bookings details :**  
   

**Conclusion:**

In conclusion, the Cloud-Enabled Bakery Ordering System provides an efficient and scalable platform for delivering fresh, custom-made bakery items directly to customers. Utilizing a lightweight web framework on Amazon EC2, the system leverages AWS Lambda for real-time order processing and DynamoDB to store customer preferences and order histories, enhancing the experience with personalized recommendations. Key AWS services such as CloudWatch ensure performance reliability, while interactive features like the Wishlist, Custom Bakes and Flavors Quiz, and in-store coupon redemption foster customer engagement. This blend of cloud-native solutions and customer-centric features offers a seamless, modernized approach to online bakery ordering and home delivery.