**Harnessing AWS for Next-Gen Online Food Delivery**

### Project Description:

The "**Real-time Food Delivery** Solution using AWS" project focuses on building a scalable and responsive platform that addresses the dynamic needs of food providers and customers. By leveraging the robust capabilities of AWS services, including EC2 for scalable computing power, RDS for efficient database management, and S3 for securely storing food-related assets, this solution creates a seamless environment for food ordering and delivery.

Utilizing Flask as the backend framework, the project facilitates real-time user interactions, enabling instantaneous order placements and notifications. This allows customers to receive updates promptly, enhancing their overall experience. The platform's architecture is designed to handle varying loads, ensuring optimal performance during peak ordering periods.

The project aims to create an efficient system for food providers, offering tools for menu management, order tracking, and analytics, while also streamlining the ordering processes for customers. By leveraging cloud capabilities, the solution enhances food delivery coordination, reduces operational overhead, and provides a user-friendly interface that improves engagement. Ultimately, this project aspires to revolutionize the food delivery landscape, making it more accessible and efficient for all stakeholders involved.

### Scenario 1: Scalable Food Delivery for Peak Demand

In food delivery scenarios, the application must handle surges in traffic, especially during high-demand periods like holiday seasons or major promotions. AWS EC2 provides the elasticity needed to scale computing resources up or down based on user demand. During peak ordering times, the application can automatically scale to accommodate high volumes of concurrent users without compromising performance. Flask manages user sessions, order placements, and real-time updates, ensuring a seamless experience even during busy periods.

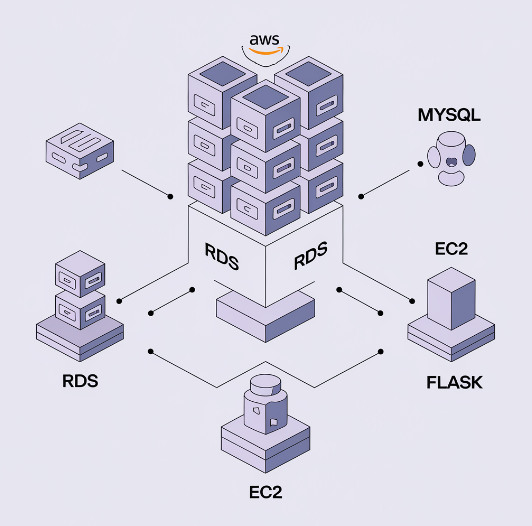
### Scenario 2: Optimized Database for Real-time Order Updates

### For effective food delivery management, Amazon RDS with MySQL ensures real-time handling of orders, customer data, and delivery statistics. With RDS, the platform benefits from automated backups, performance optimization, and high availability, ensuring a smooth and reliable ordering experience. The system efficiently tracks available menu items, customer information, and order status, supporting a large user base while ensuring data accuracy for providers and customers.

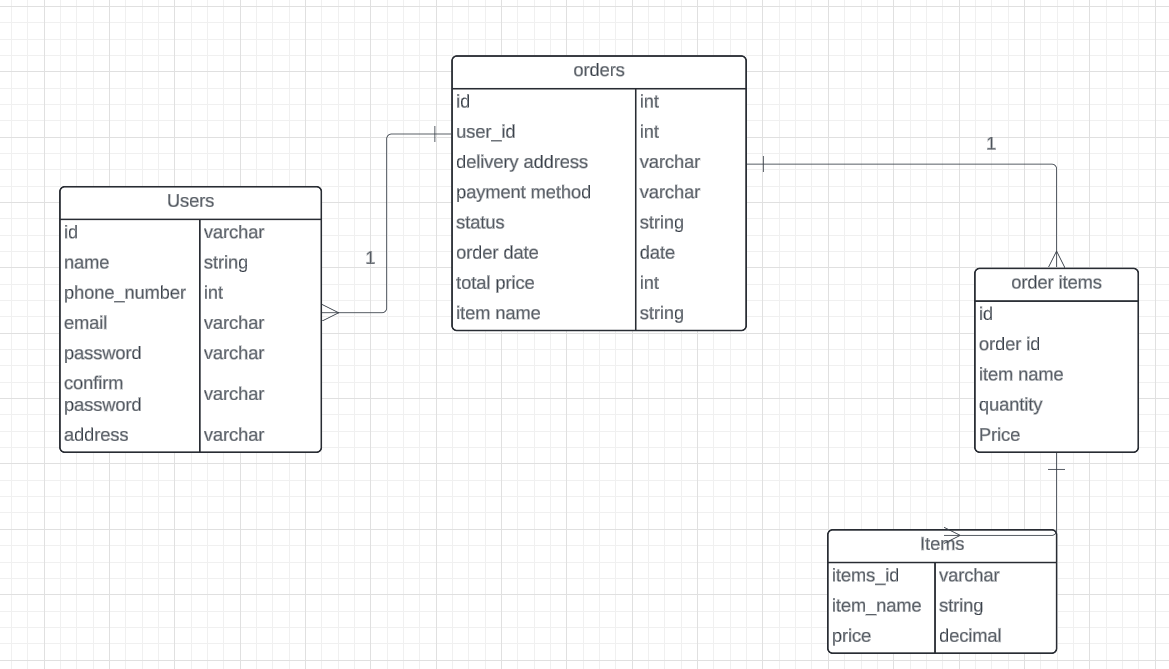
### Scenario 3: Secure Storage and Access for Food Media

AWS S3 offers a secure and scalable storage solution for food-related media files, such as promotional images, videos, and menus. With integrated IAM, access to media and sensitive data can be restricted to authorized users only. S3’s scalability ensures that even large media files can be efficiently stored and retrieved, supporting food providers in managing their promotional content.

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>

### 

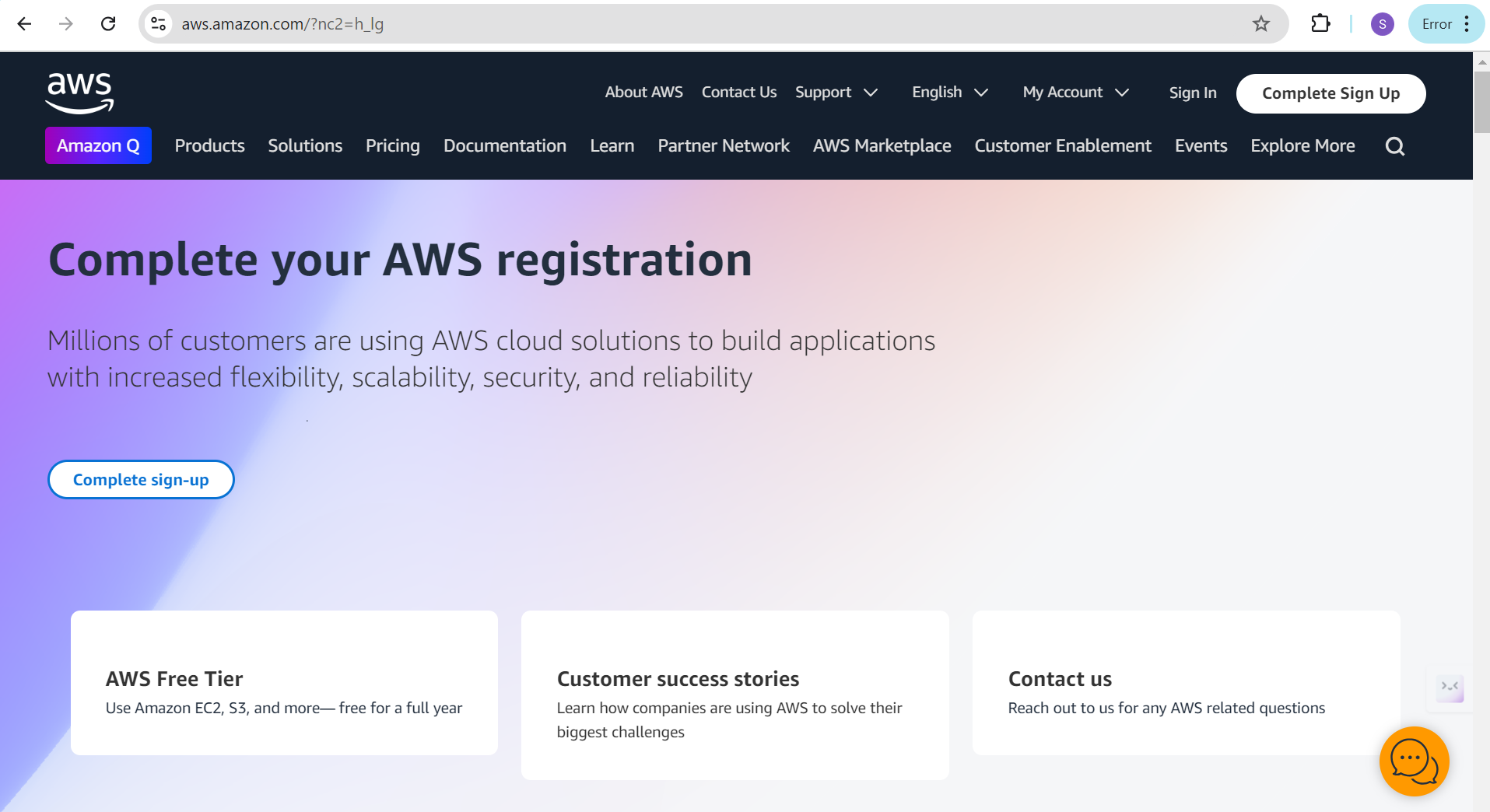
### 

### Project Flow

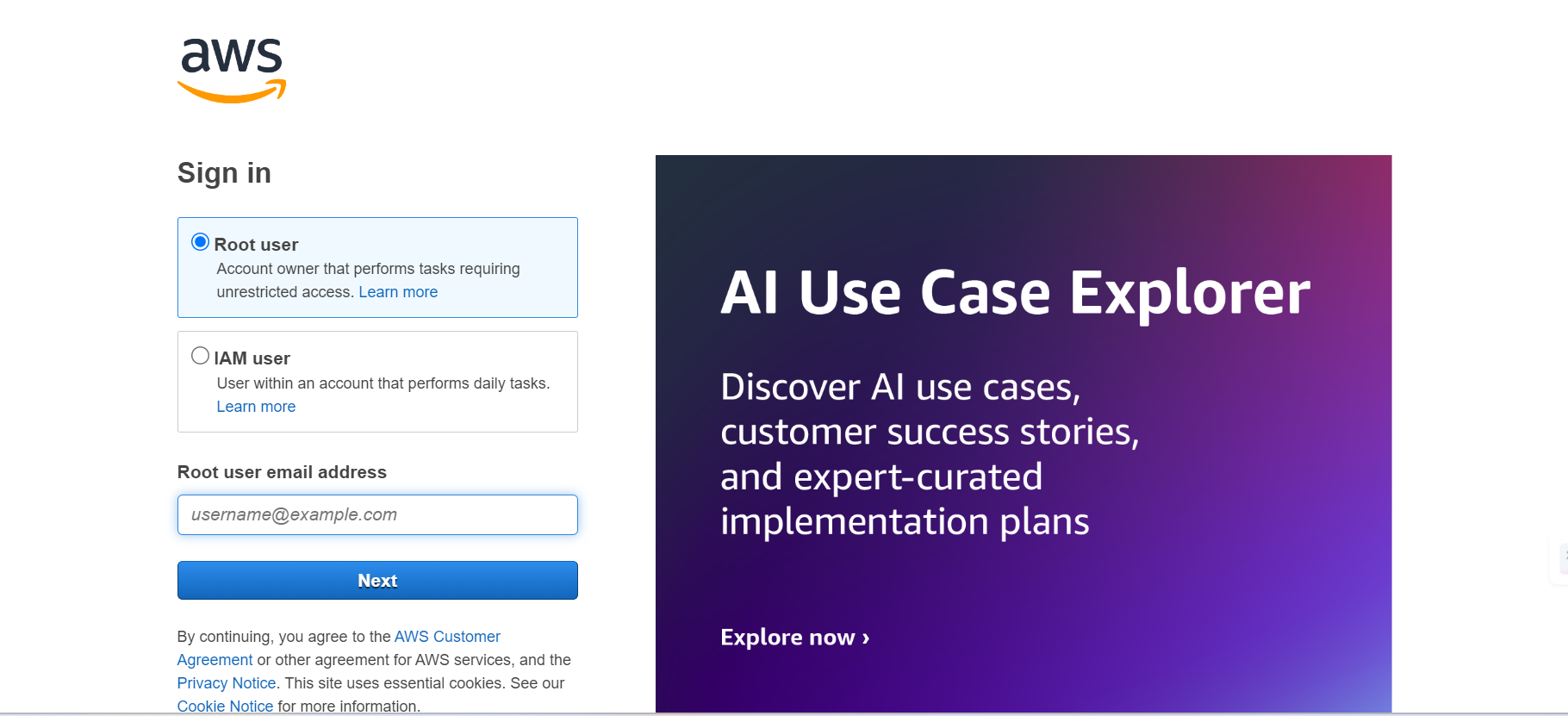
* **AWS Account Setup and Login**
  + **Activity 1.1:** Set up an AWS account if not already done.
  + **Activity 1.2:** Log in to the AWS Management Console to manage resources.
* **RDS Database Creation and Setup**
  + **Activity 2.1:Create an RDS Instance.**
* Navigate to the RDS service in the AWS 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.

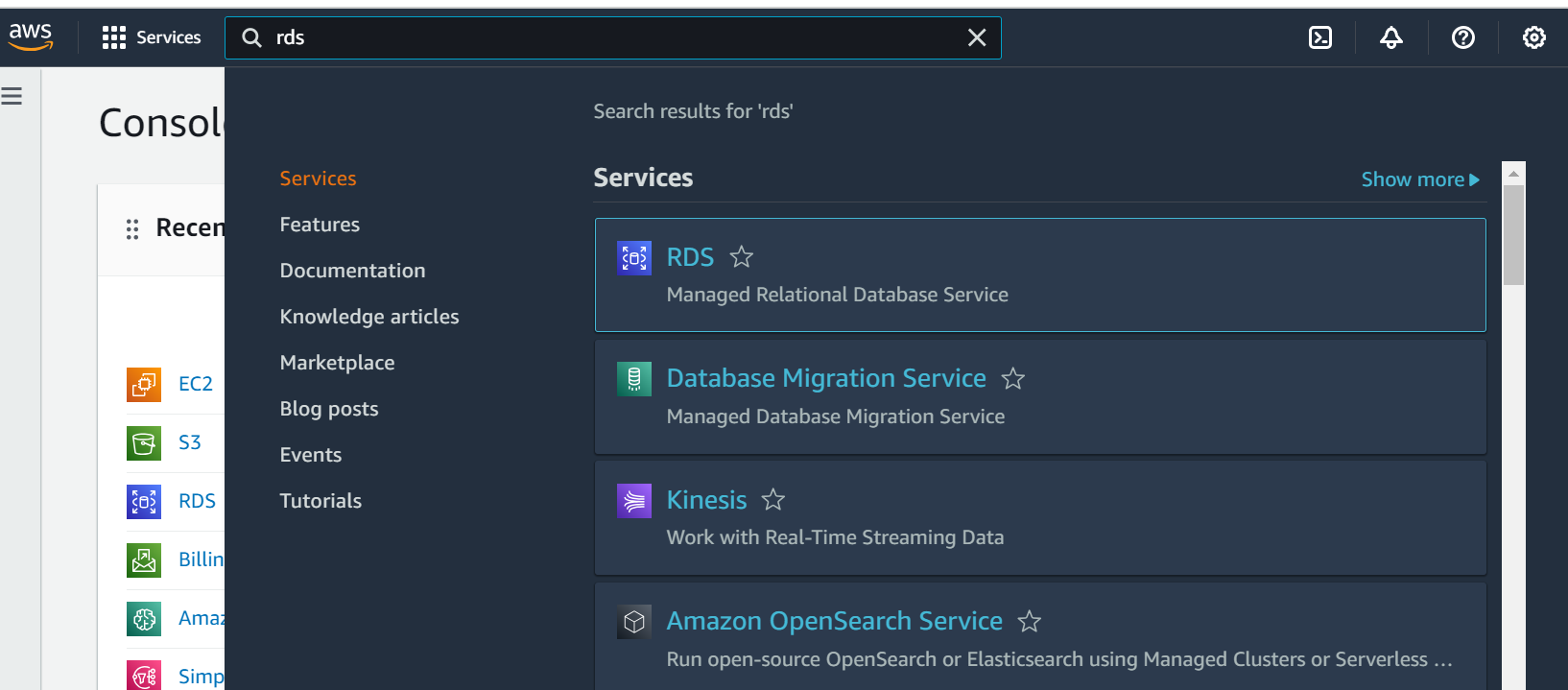


* **Activity 1.2: Log in to AWS Management Console**
  + Access the AWS Management Console using your login credentials.

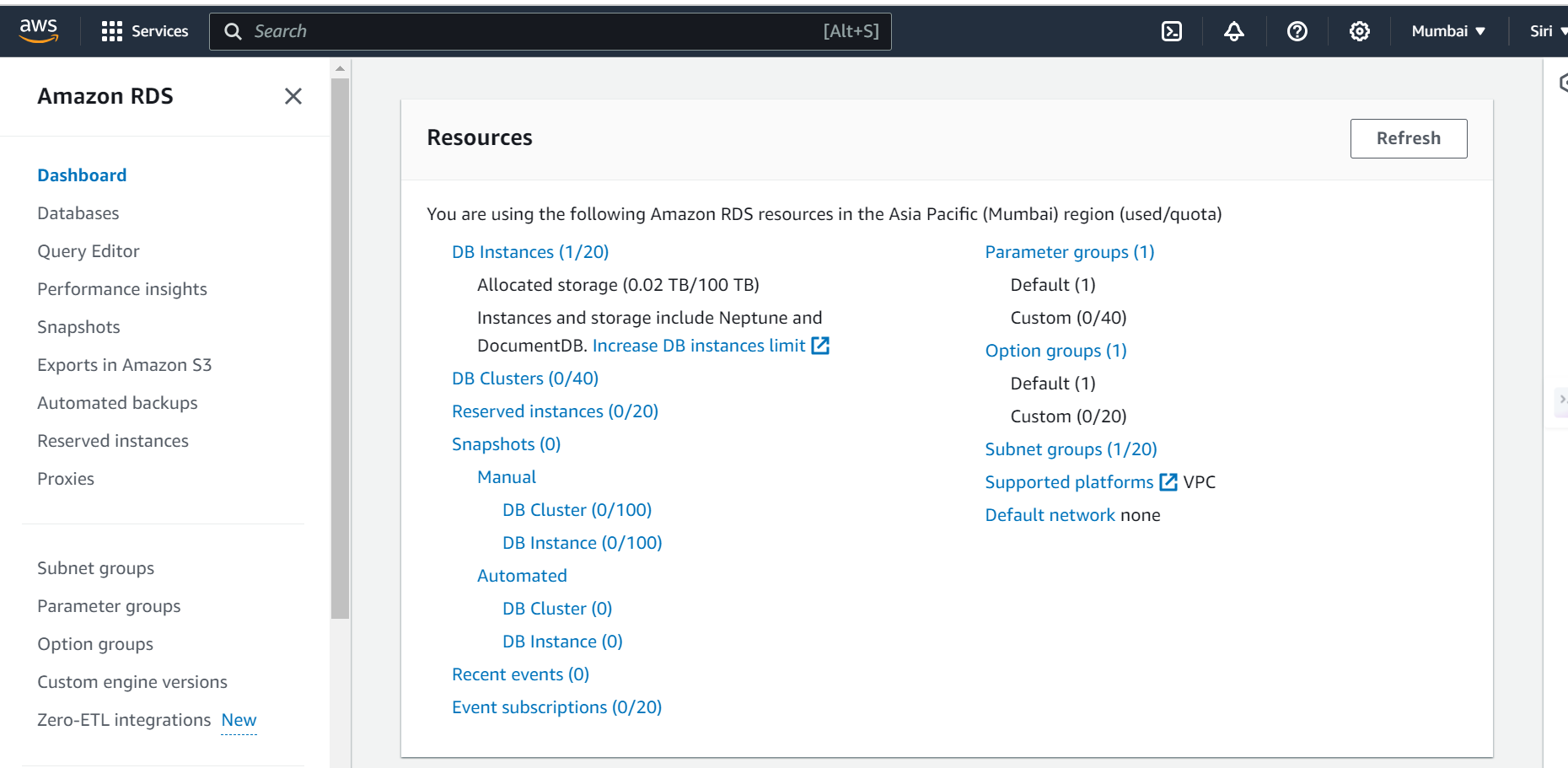


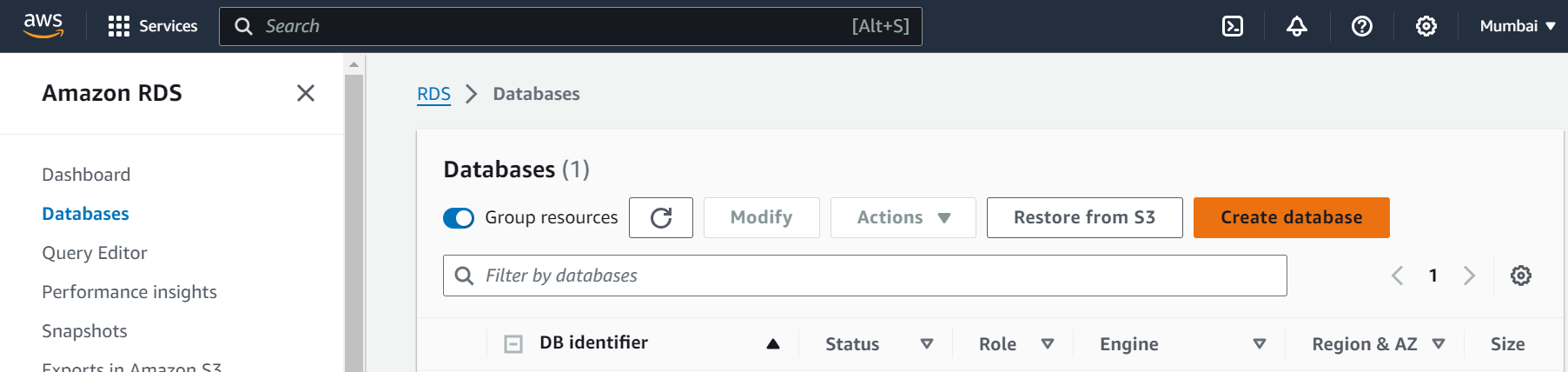
**Milestone 2: RDS Database Creation and Setup**

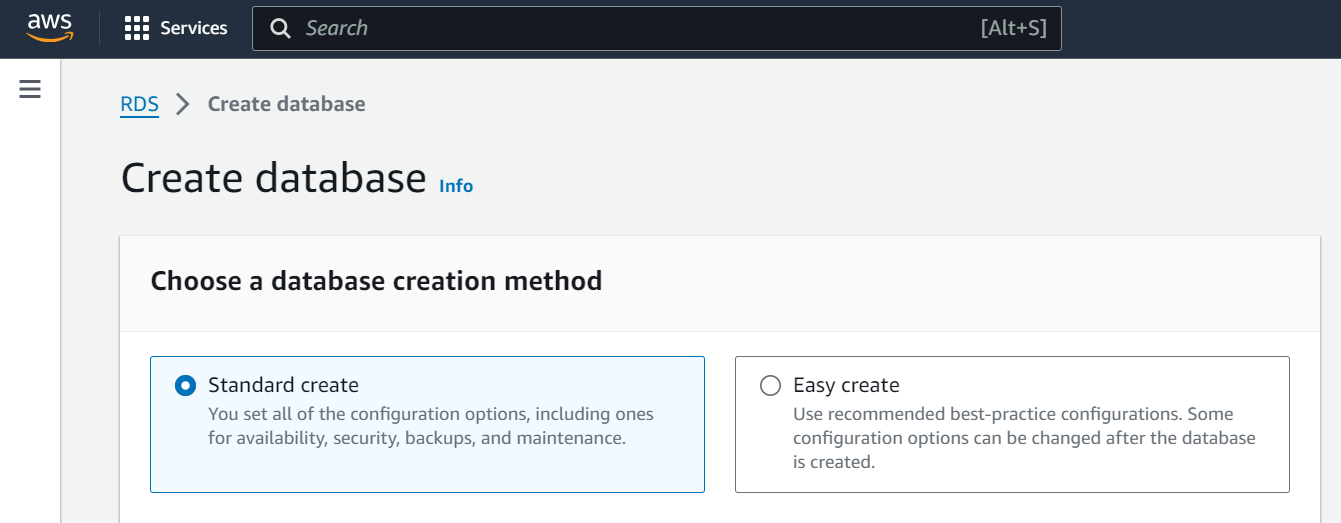
* **Activity 2.1: Create an RDS Instance**
  + Choose the RDS service from the AWS Management Console.

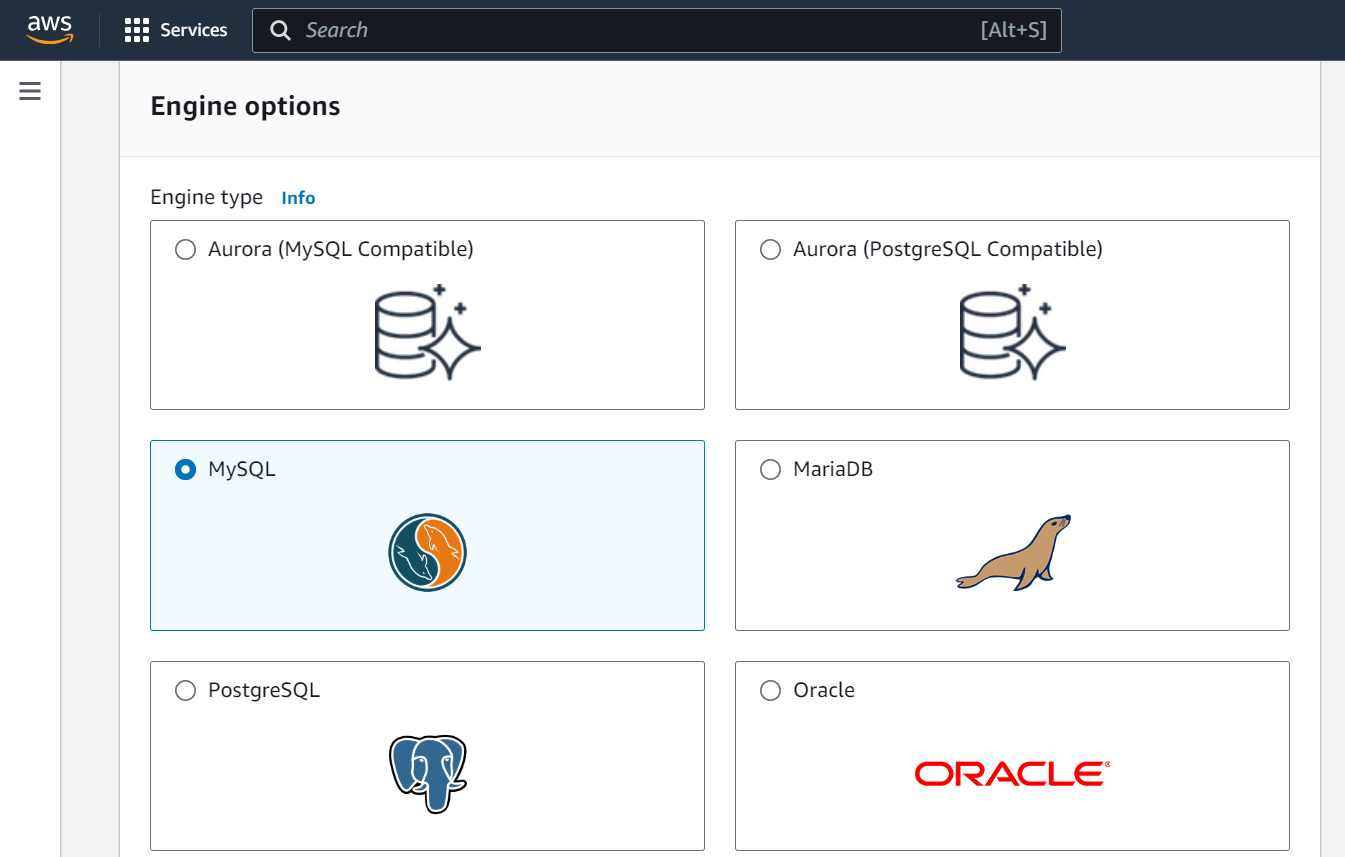


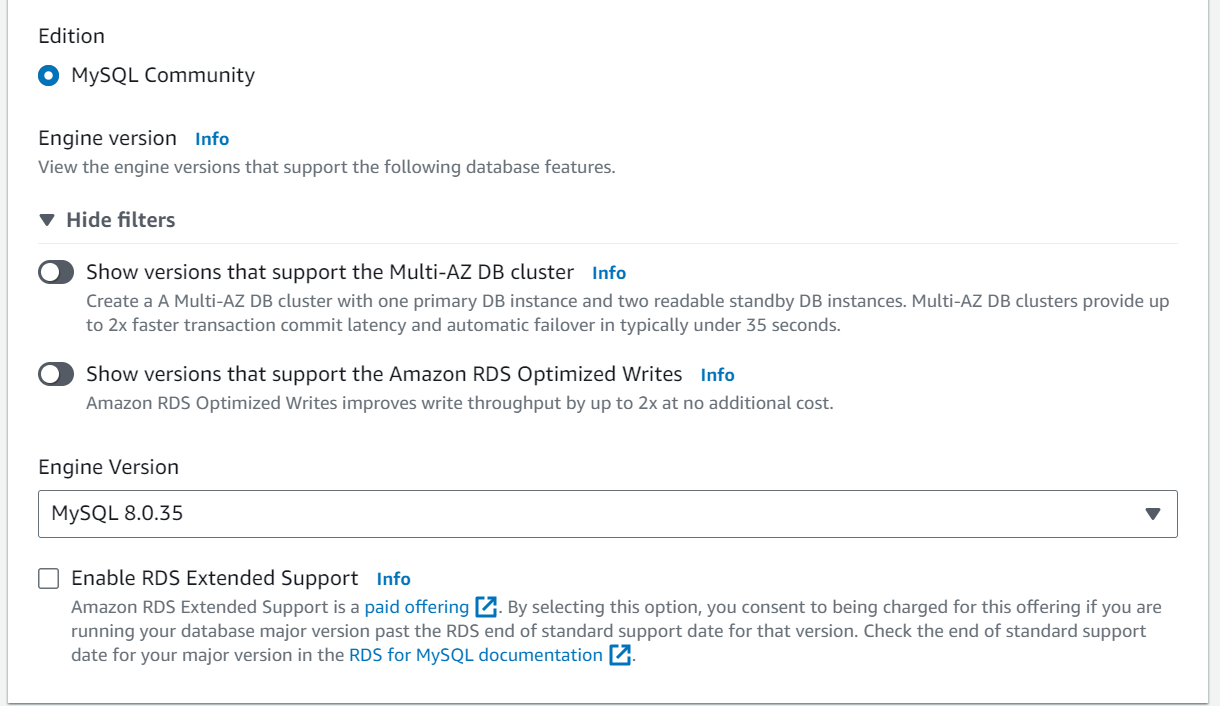
* + Select MySQL as the database engine, configure the instance settings (e.g., storage, instance class), and launch the RDS instance.

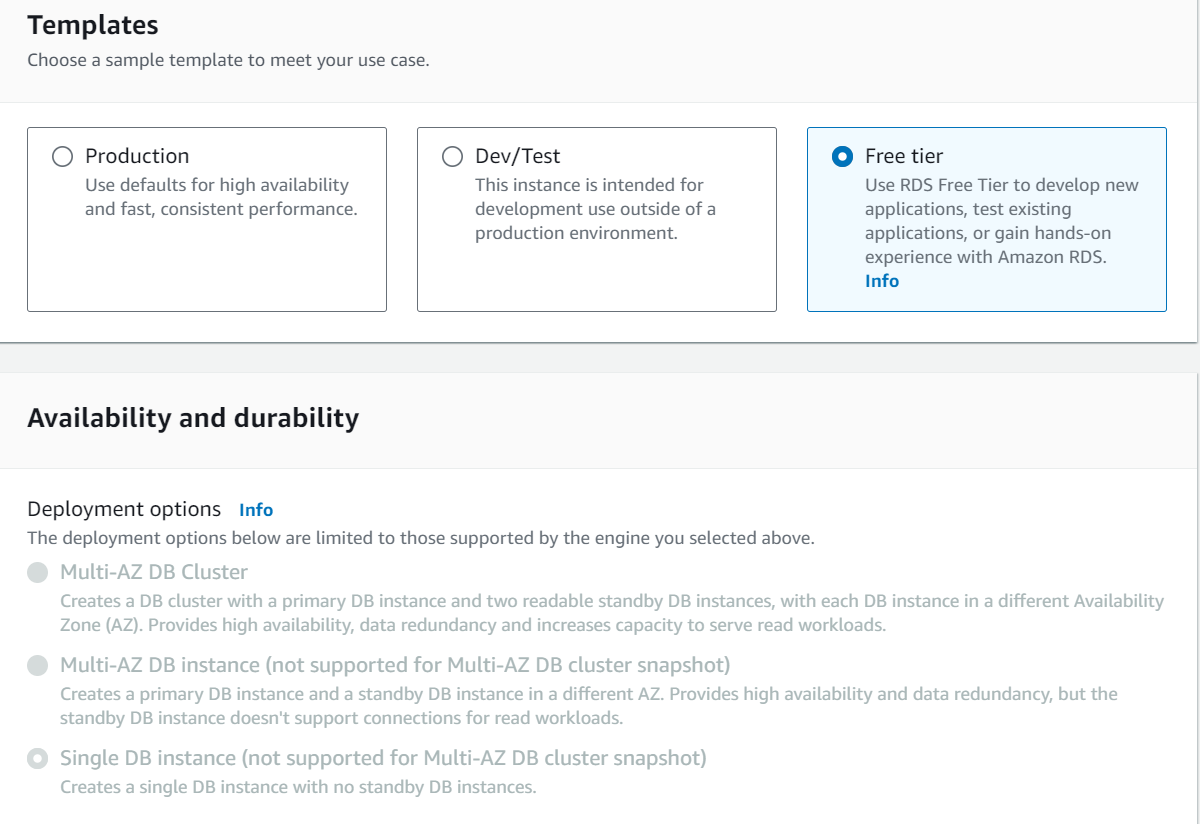


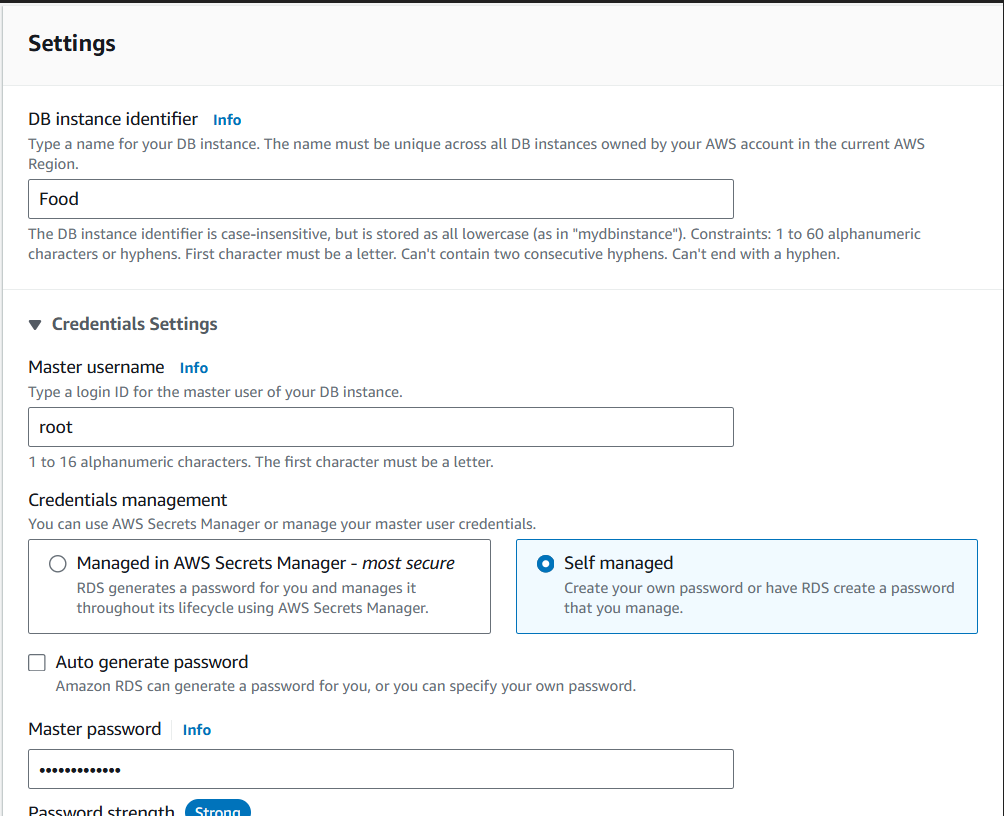


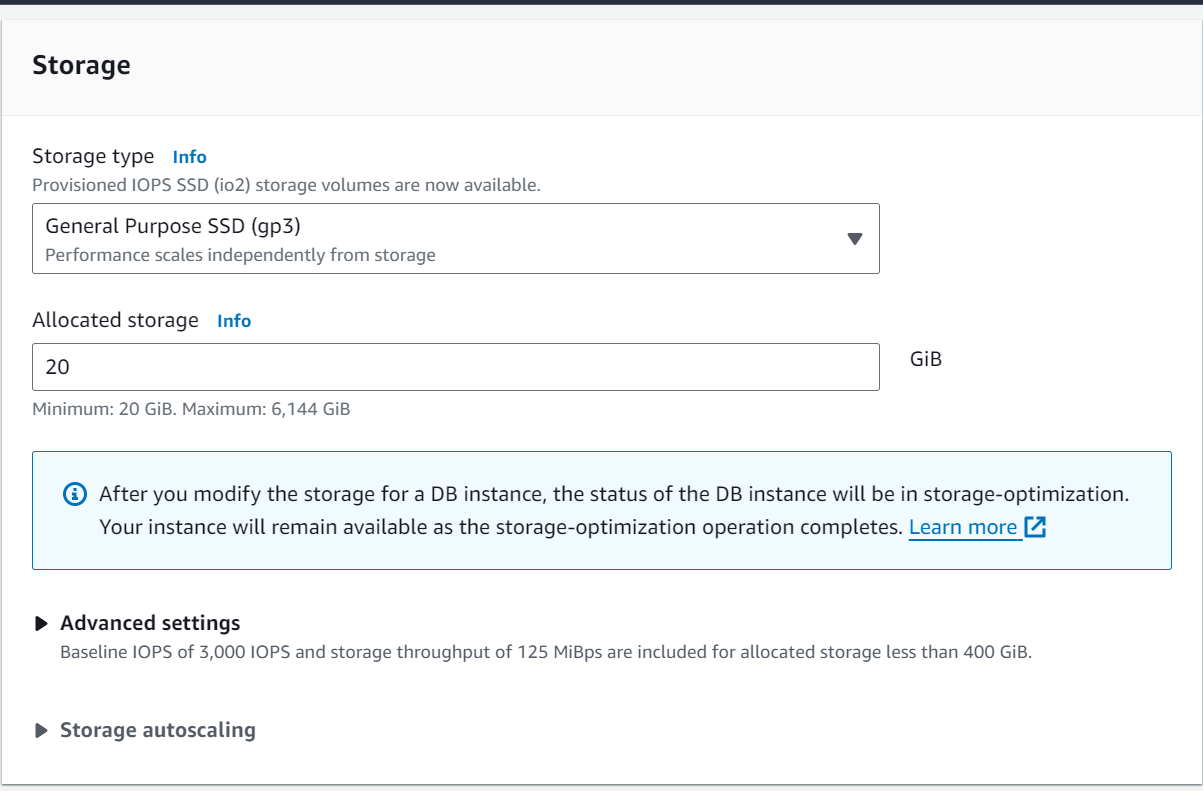


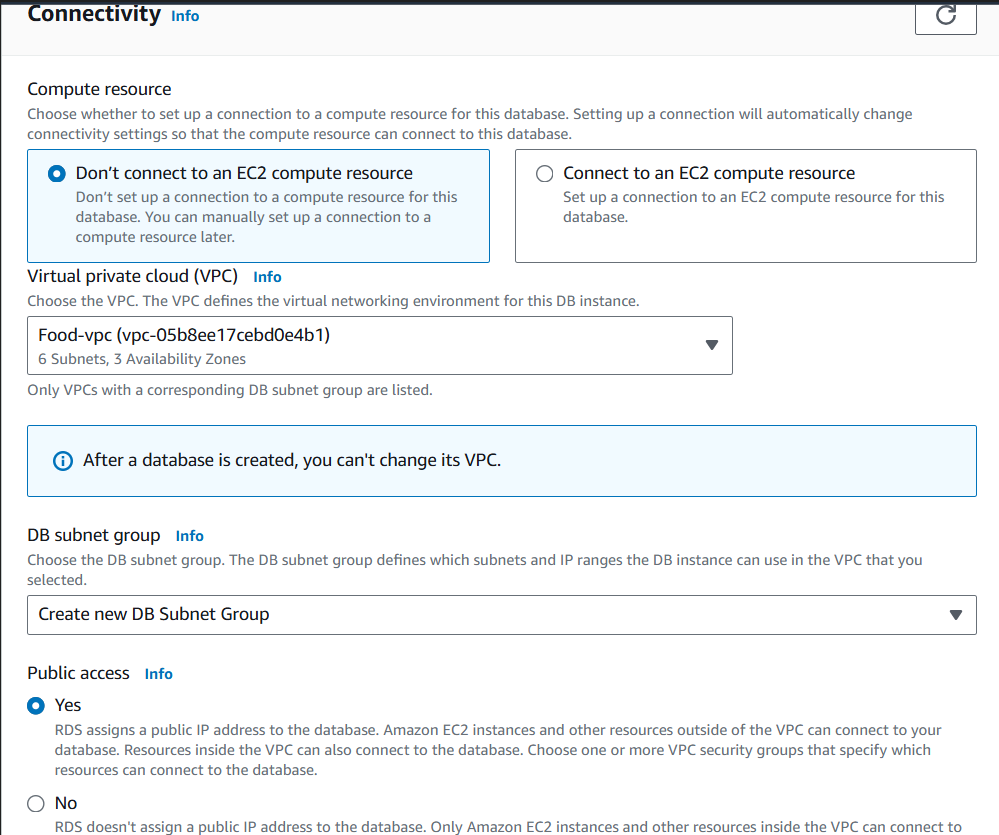


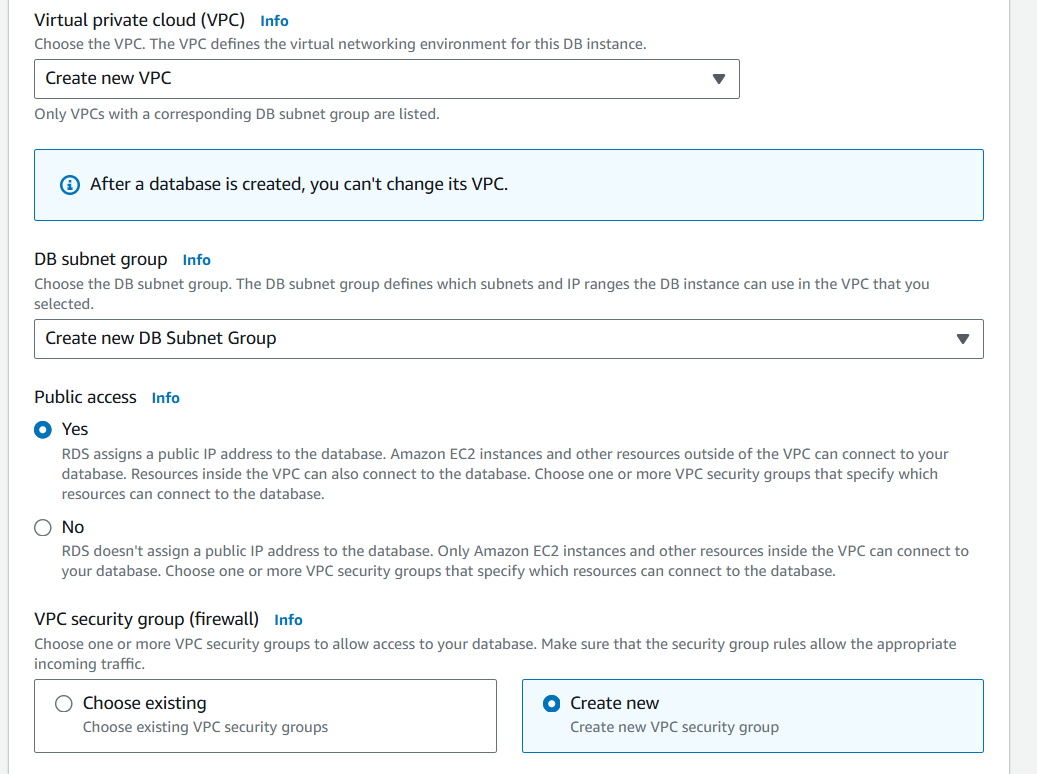


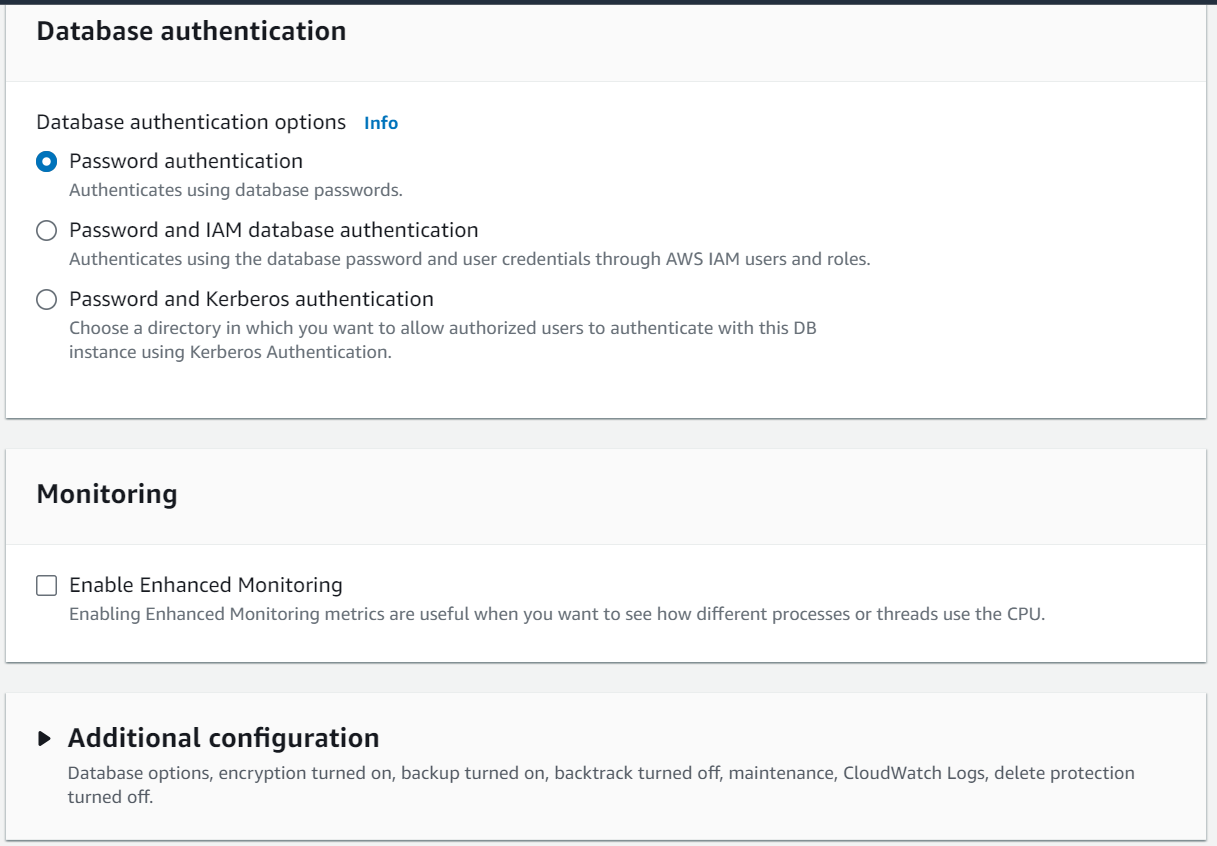


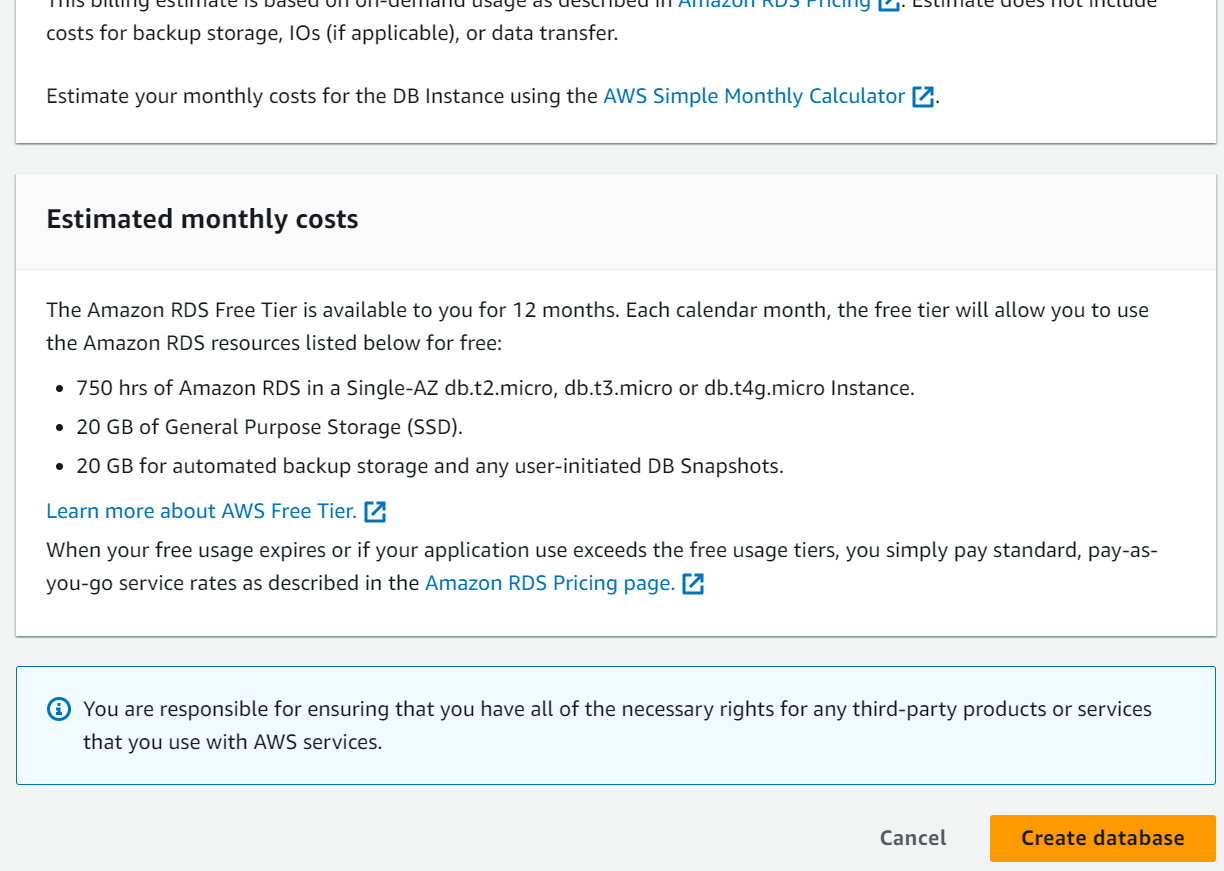




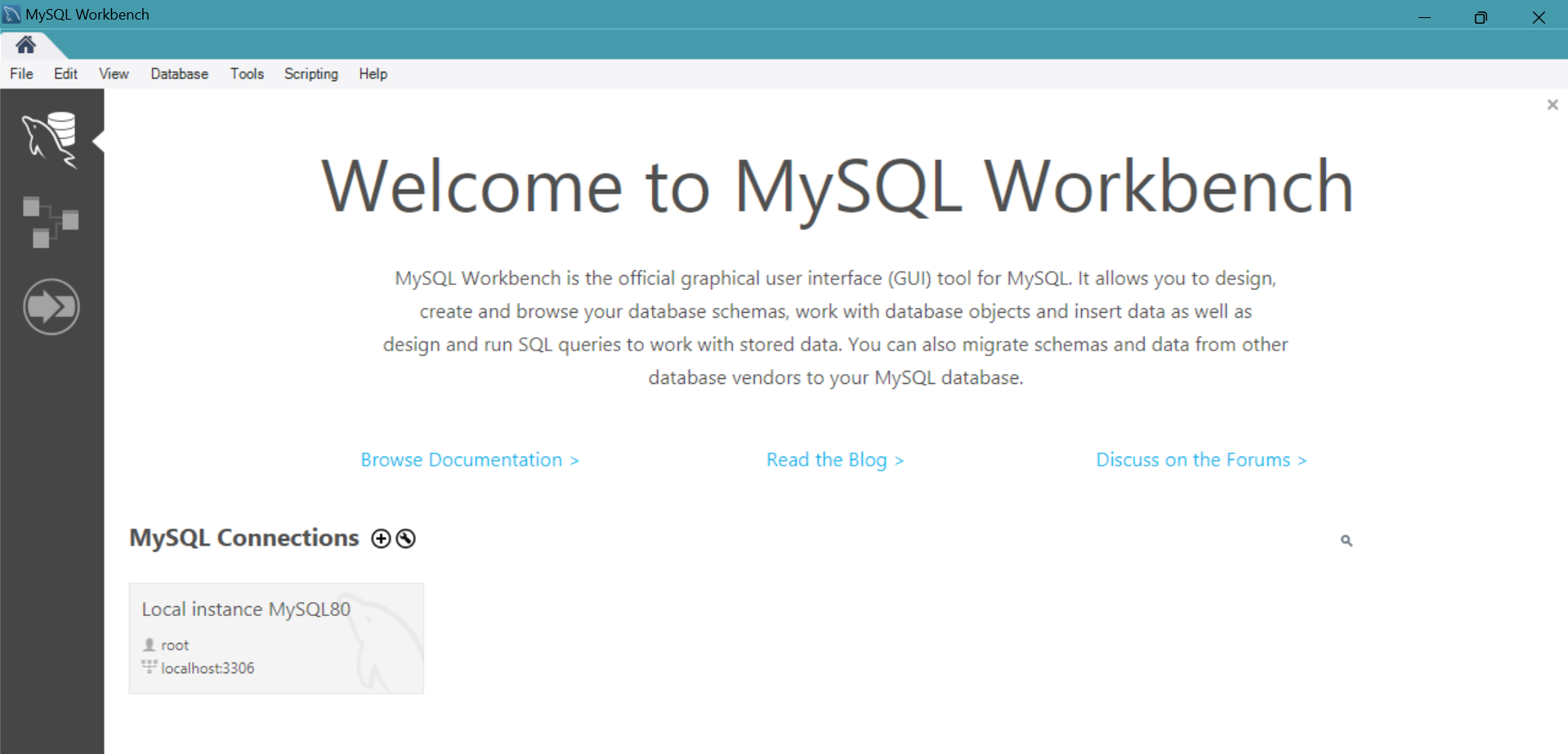
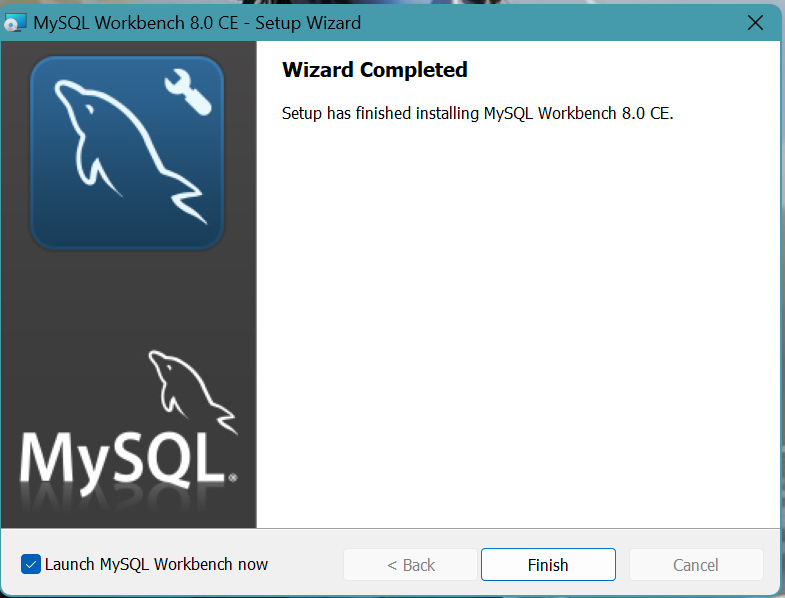


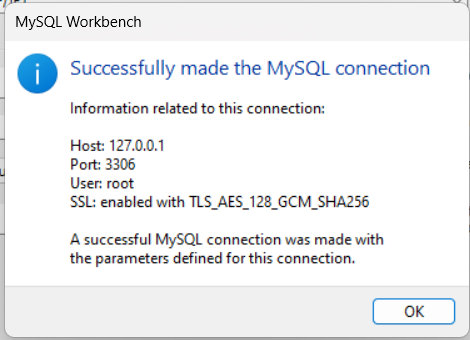
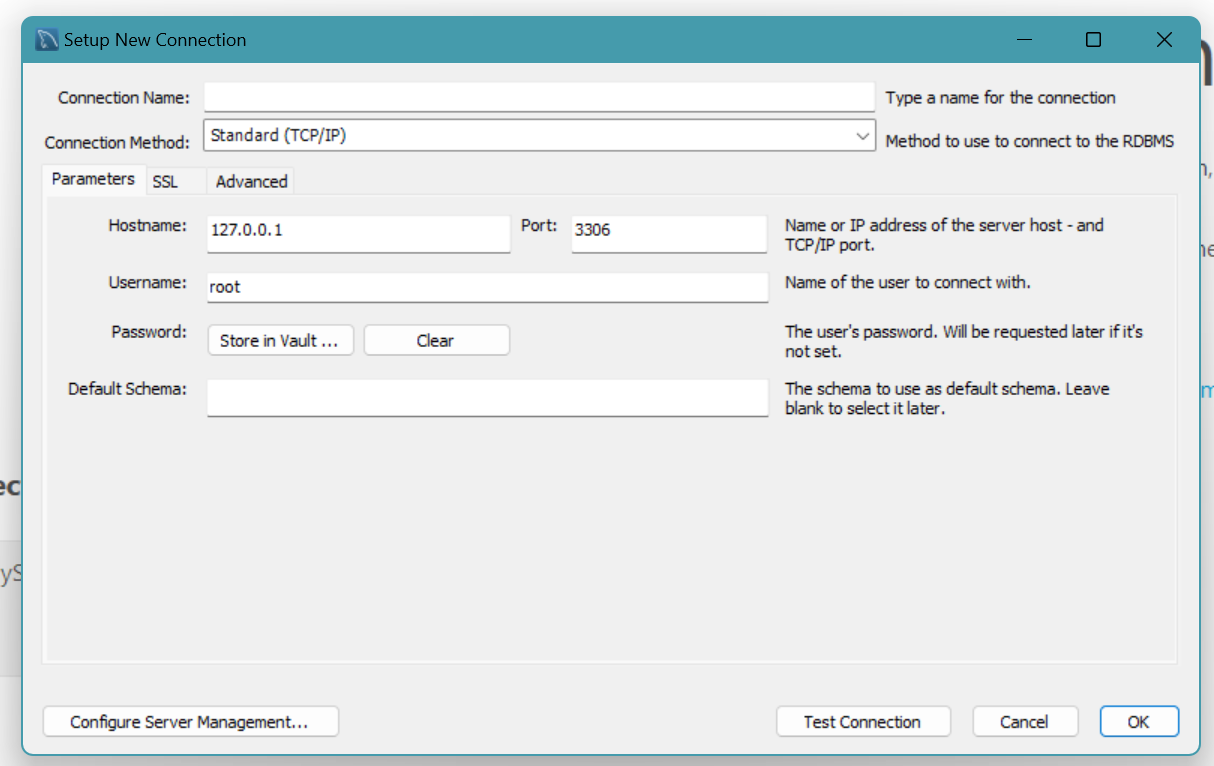


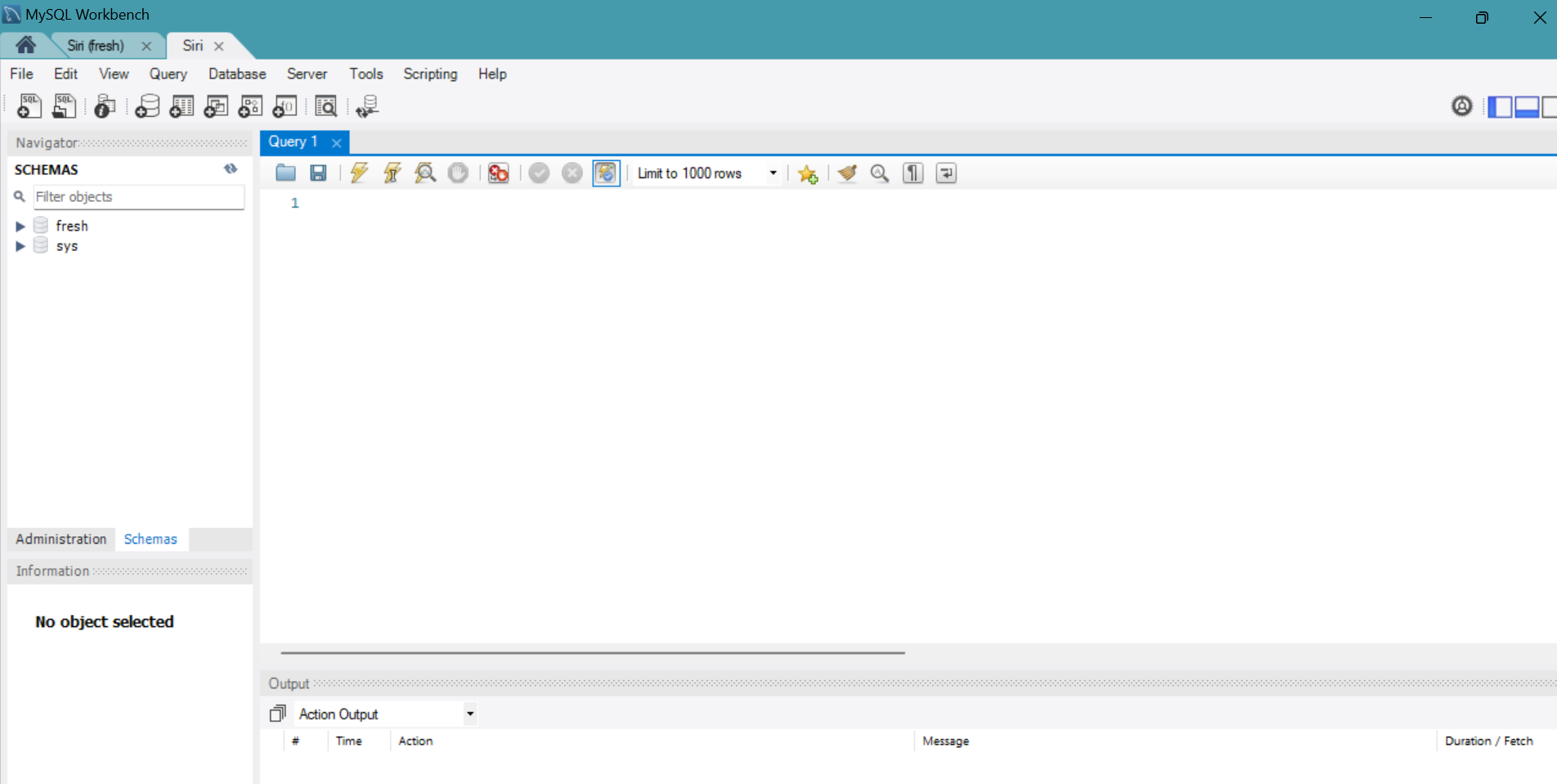




* **Activity 2.2: Configure Database Access**
  + Set up security groups, create database credentials, and configure access policies to ensure secure connectivity to the database.
* **Activity 2.3: Install MySQL Workbench**
  + Download and install MySQL Workbench on your local machine for database management.

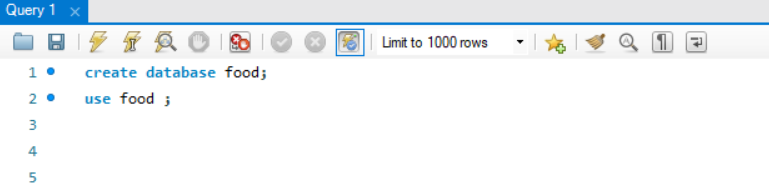


* + Connect to the RDS instance via MySQL Workbench using the endpoint and credentials from AW
  + 
* Give a connection name.
* Copy the endpoint from the RDS database that is created in AWS and paste it in **Hostname.**
* Write the username and enter the password , then click on **Test Connection**.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



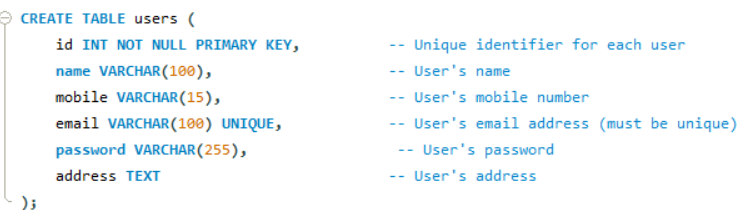
**Activity 2.4:** Create the Database and the tables which are required.

* **Create a basic database schema for an Food Delivery platform**



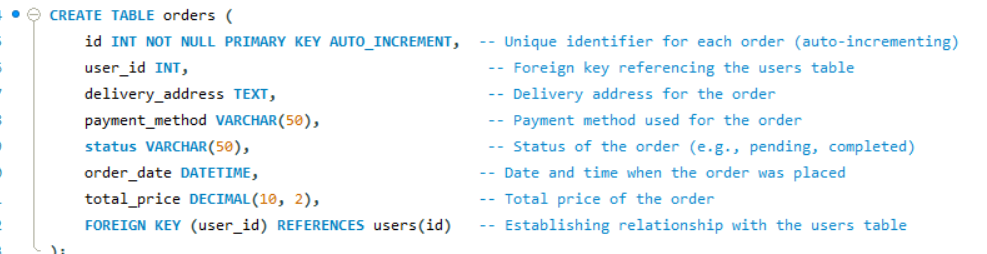
**users:**

* 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.



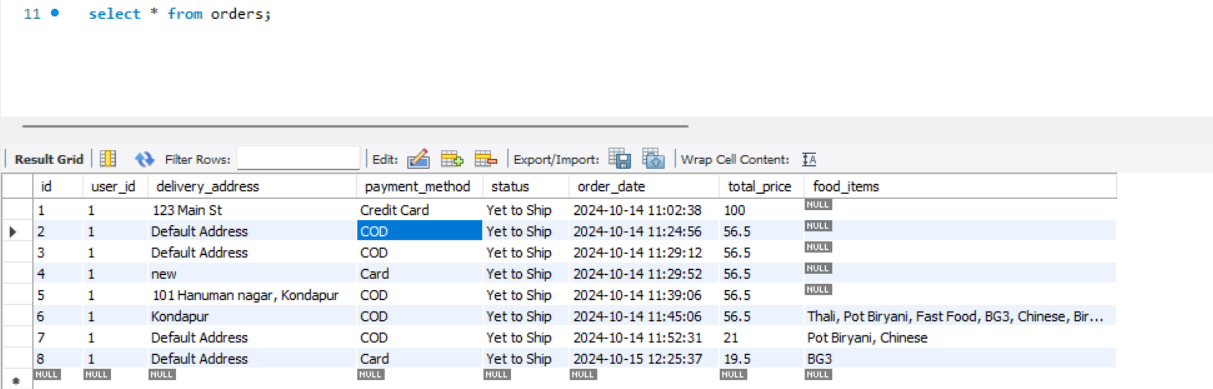
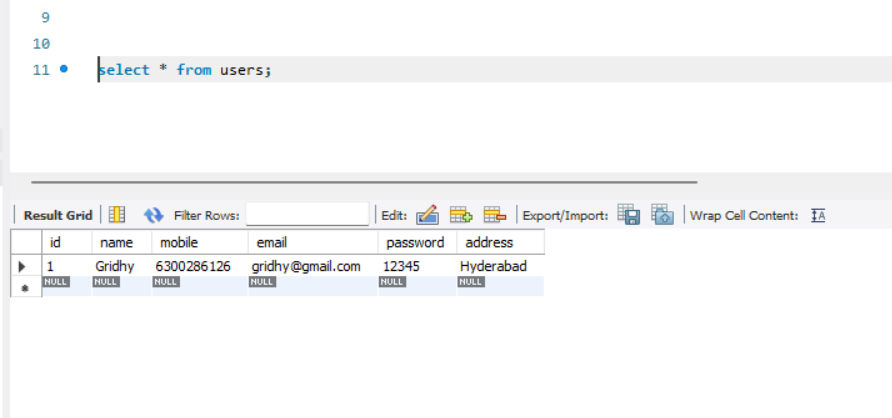
**bookings:**

* 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.

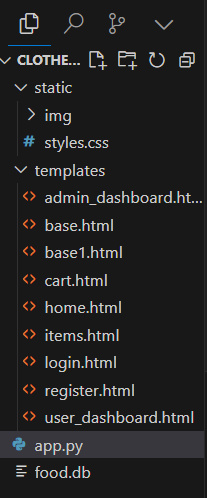


### Data Operations:

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

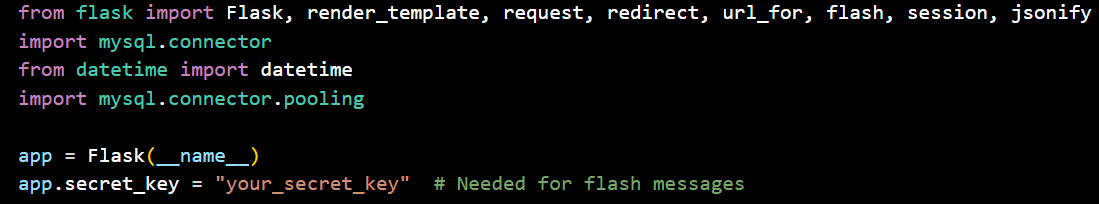


**Milestone 3: Frontend Development and Application Setup**

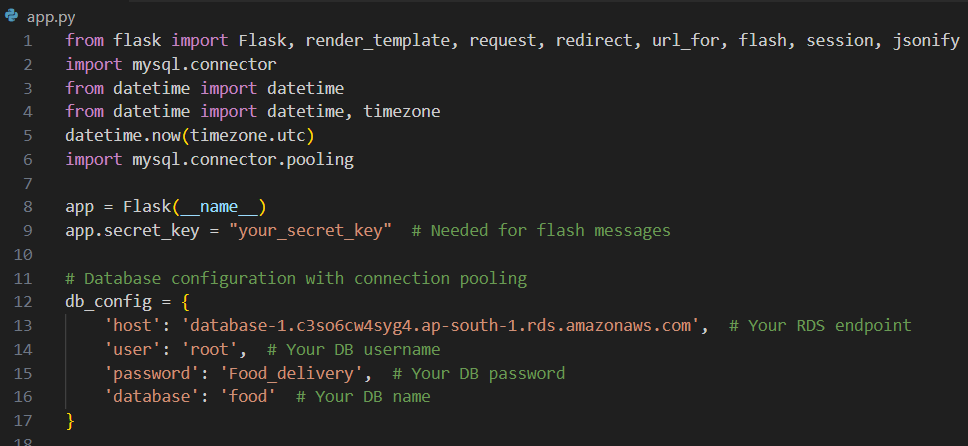
* **Activity 3.1: Build the Frontend**
  + Develop HTML, CSS, and Python-based Flask application files for Food Delivery
* **Activity 3.2: Integrate Application with RDS**
  + Connect app.py (Flask application) to the MySQL RDS database by configuring database connection settings and verifying connectivity.

**Description of the code :**

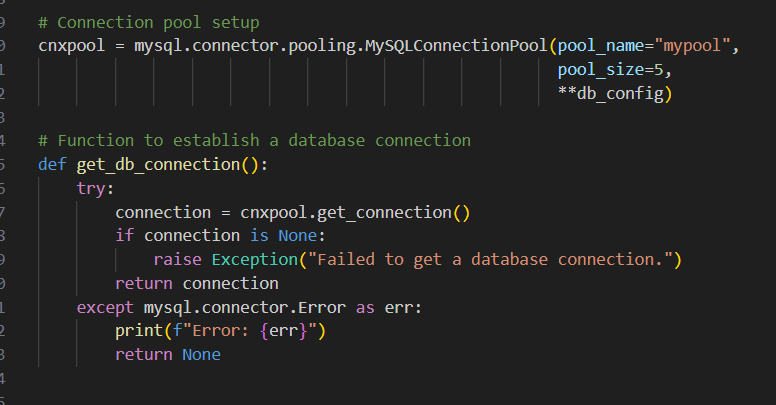
**1.Flask App Initialization**: Initializes a Flask application with secret key for sessions.



**2.Database Configuration**: Configures MySQL RDS with connection pooling for efficient database access.



**3.Connection Pool**: Uses MySQL connection pooling to handle multiple database connections.



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

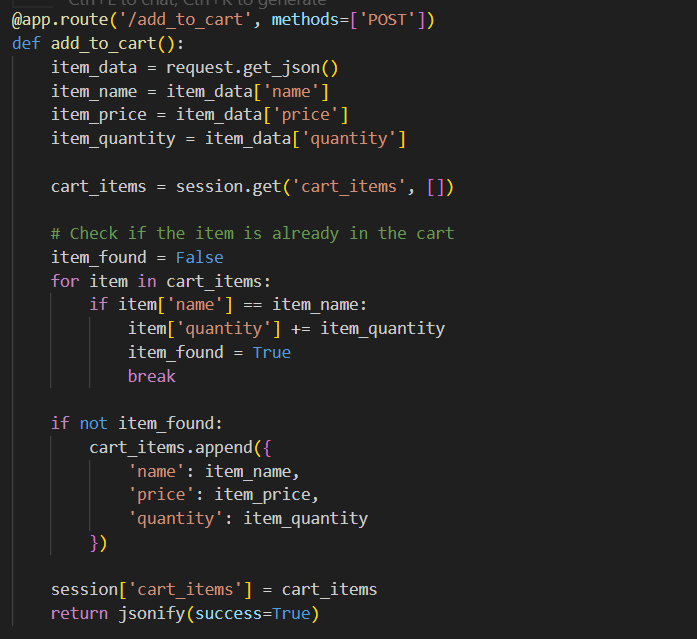
**5.signup Route (GET/POST)**: Handles user registration, inserts user data into the database



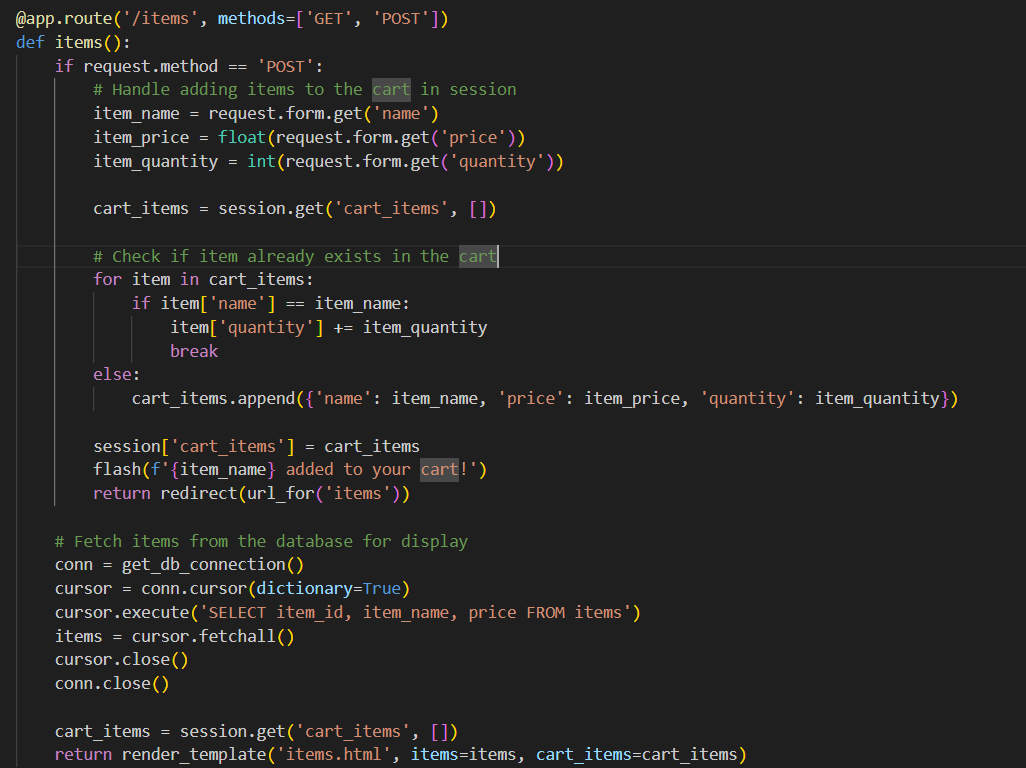
**6.Login Route (GET/POST)**: Authenticates user with email and password, creates session on success.



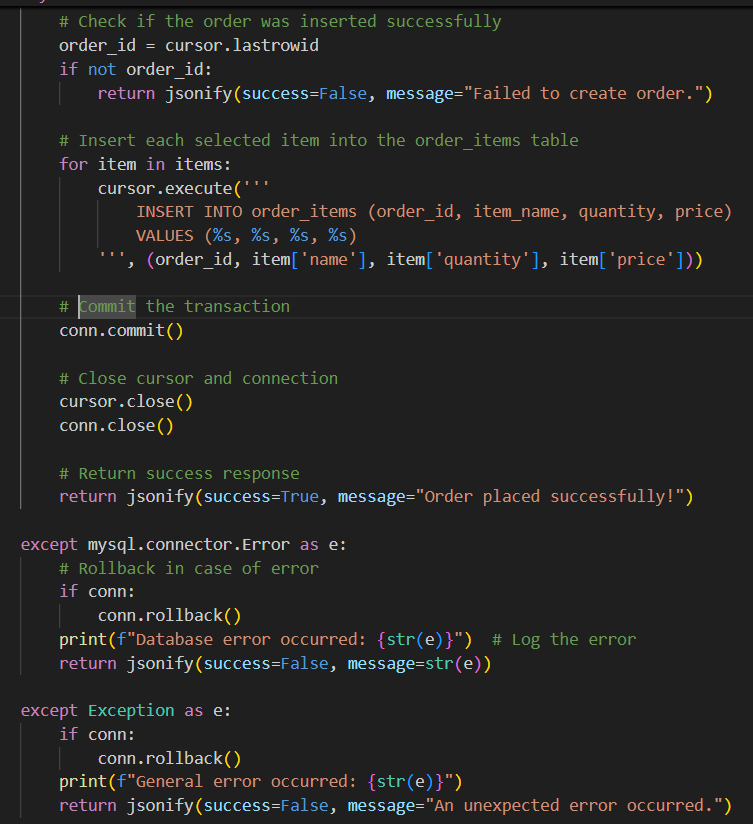
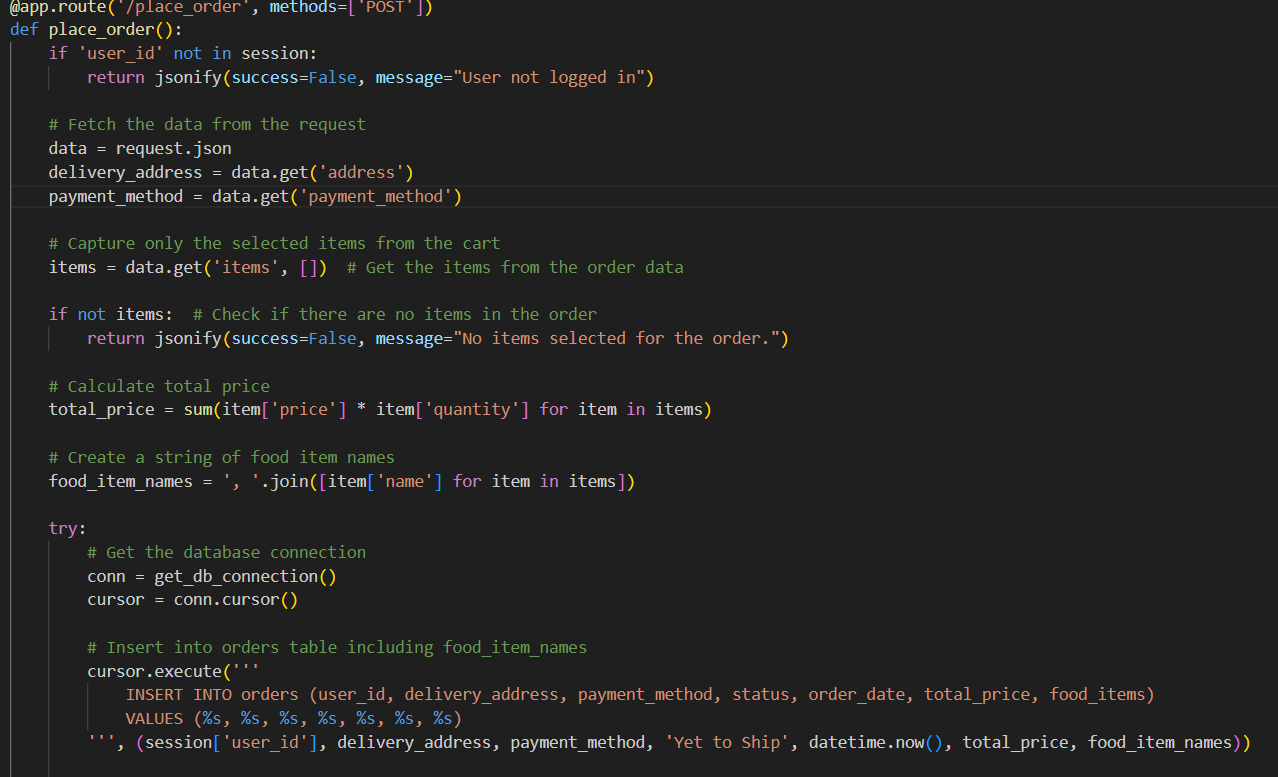
**7.Add to cart Route**:Processes customer orders by adding selected food items to the cart, preparing for checkout.



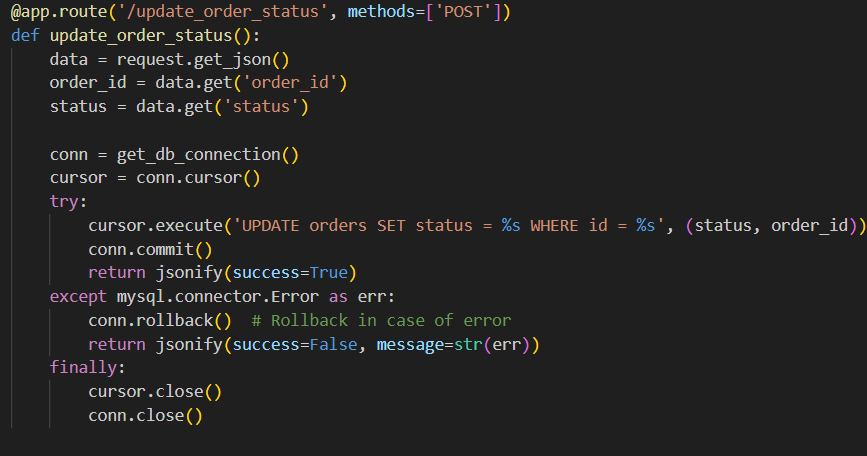
**8.Cart Page**: Displays the customer's selected food items, allowing them to review and modify their order before checkout



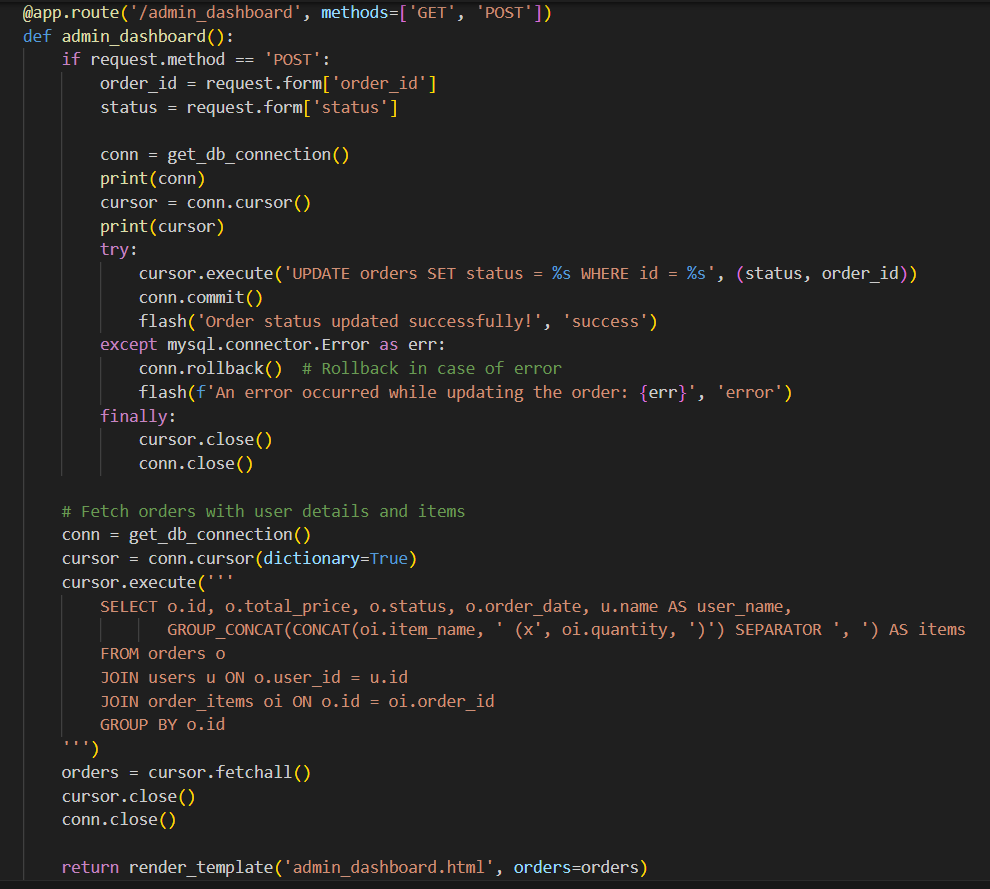
**9.Placing cart Items page:** Confirms the customer's order by placing the selected cart items and proceeding to the payment or delivery details



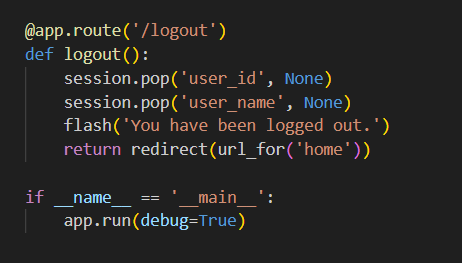
**10.Update order status route**: Shows the message “Thanks for Booking!” and redirects to the home page.



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



**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.

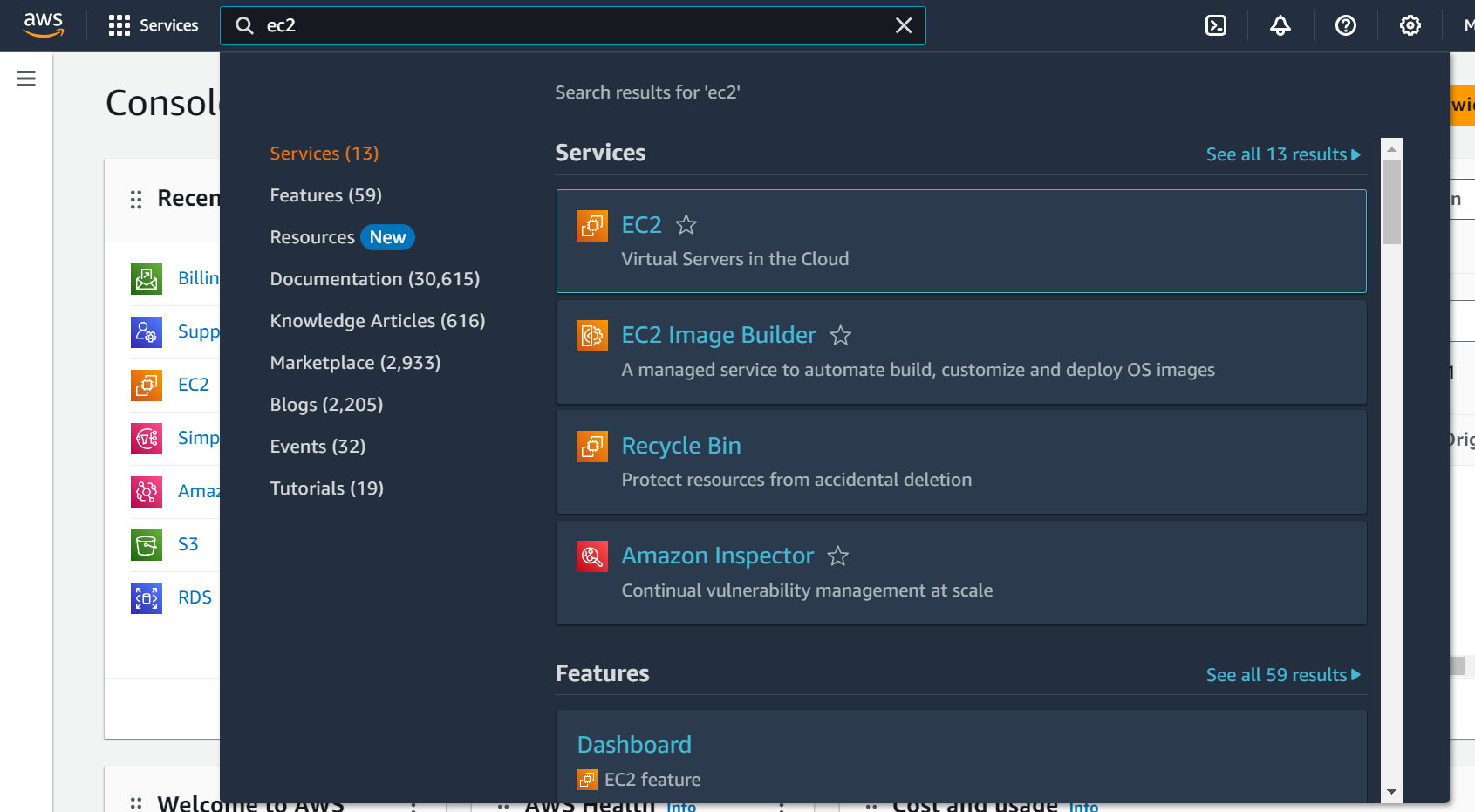
**Session Management:** Uses Flask sessions to store customer preferences and cart details for seamless navigation.

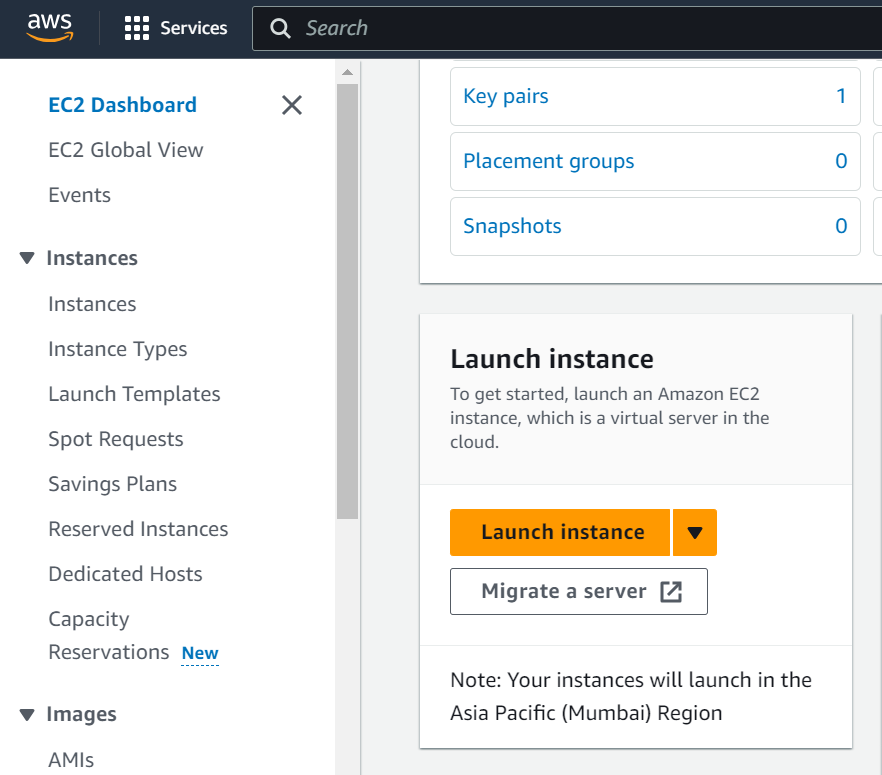
**Flash Messages:** Provides user feedback through flash messages for login, registration, order confirmation, and status updates**.**

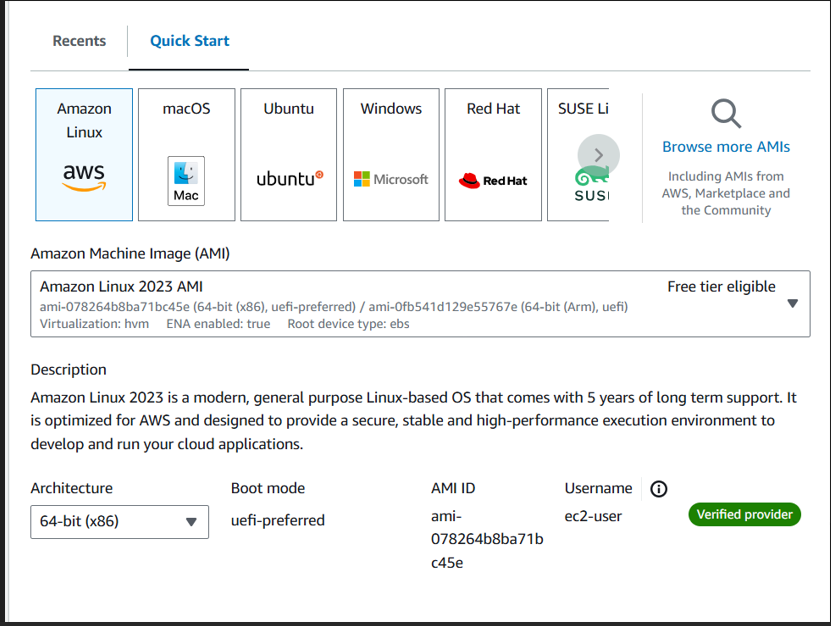
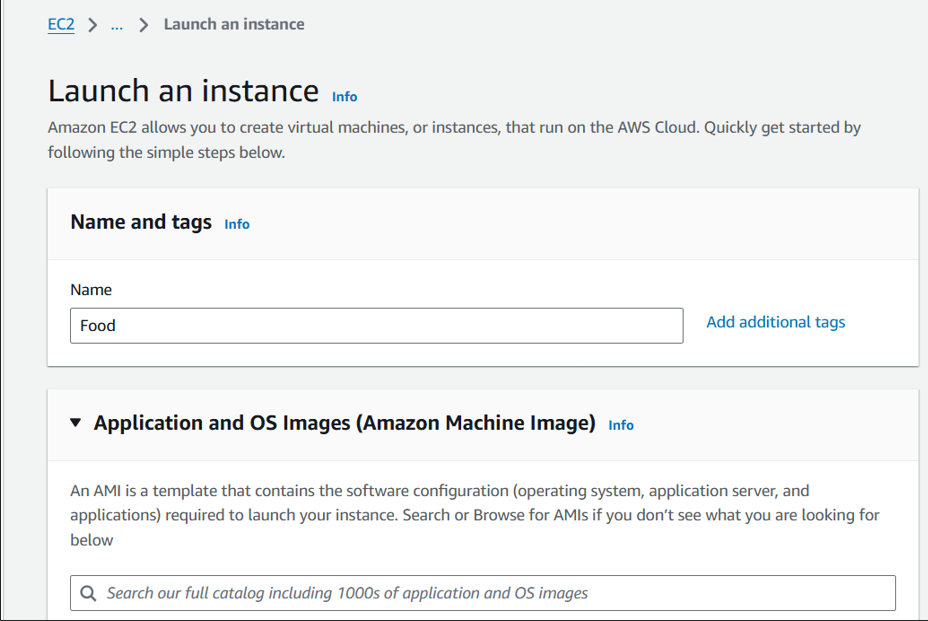
**Menu Data Fetching:** Retrieves food items, prices, and availability from the database to display on the ordering interface.

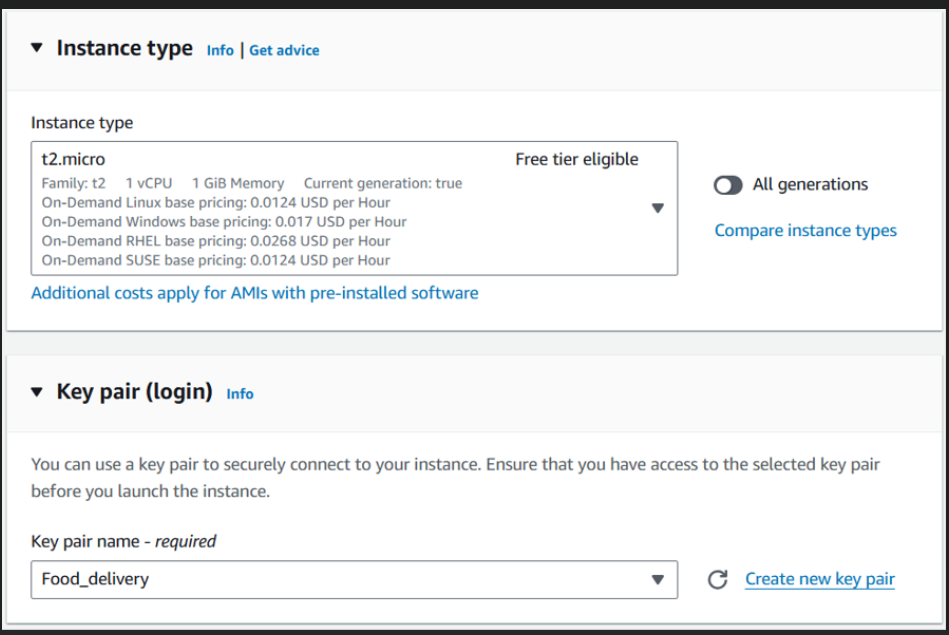
**Milestone 4: EC2 Instance Setup**

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

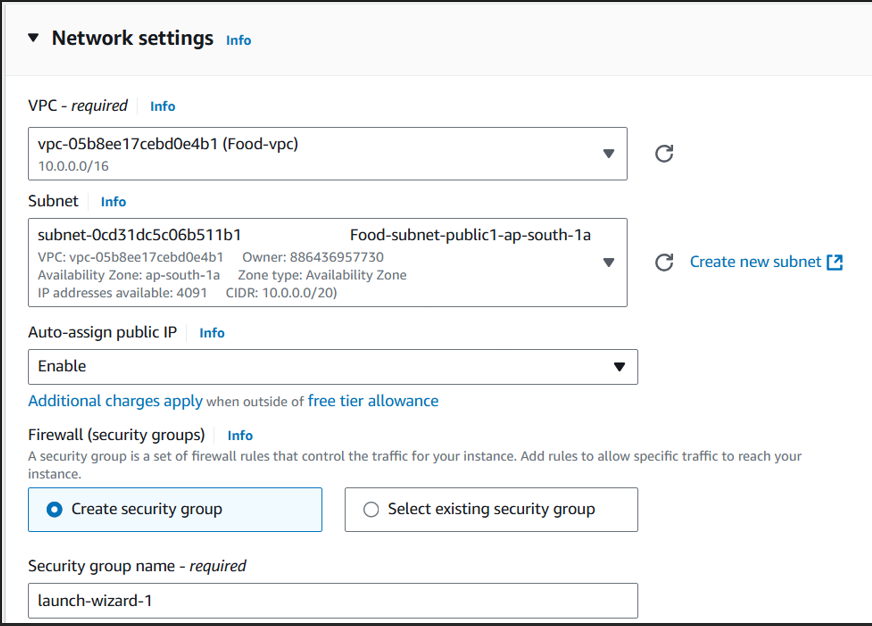






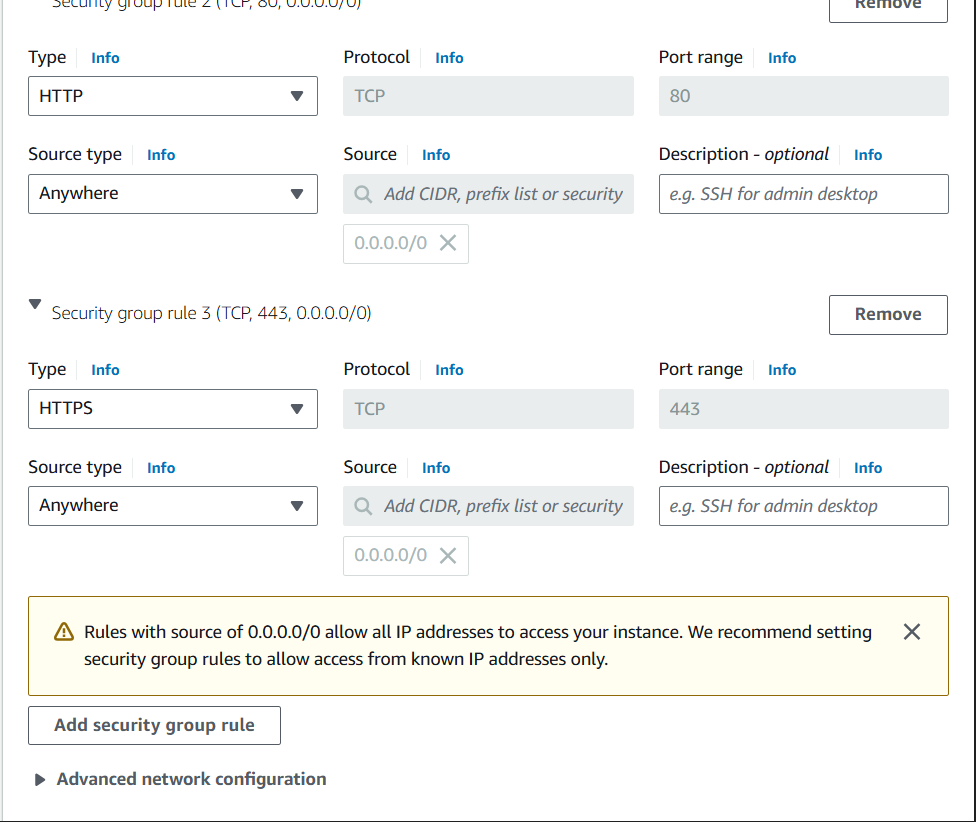
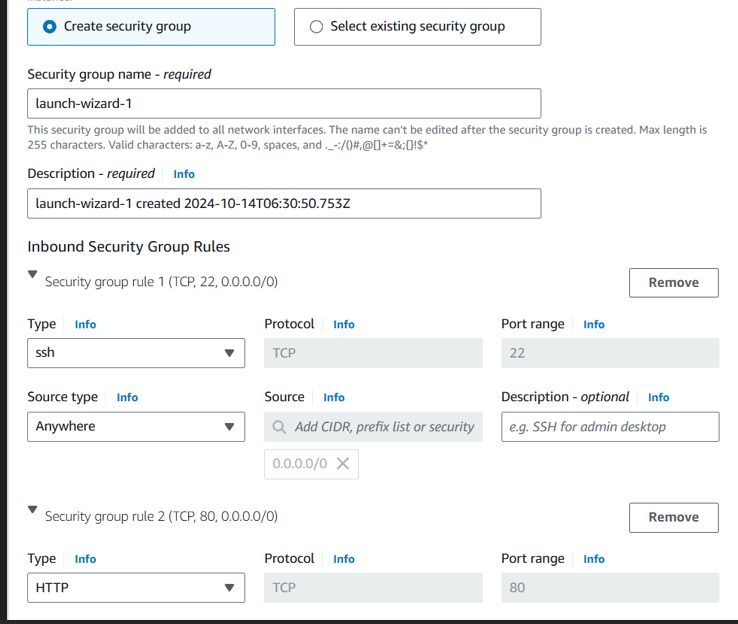


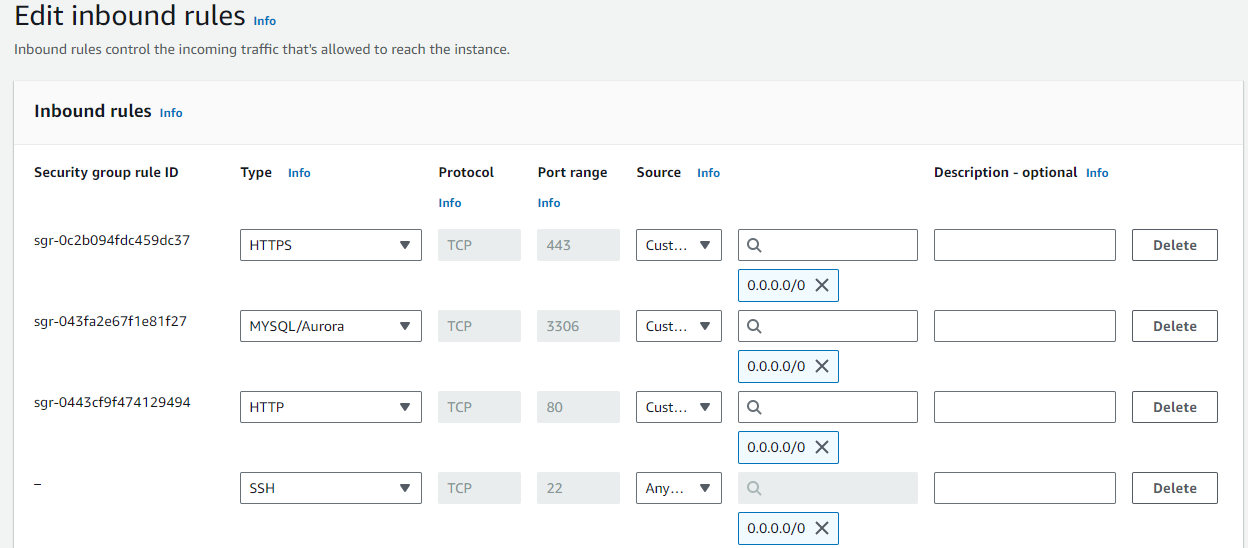
* **Activity 4.2: Configure Network Settings**
  + Set up the security group to allow HTTP, HTTPS, and SSH traffic.



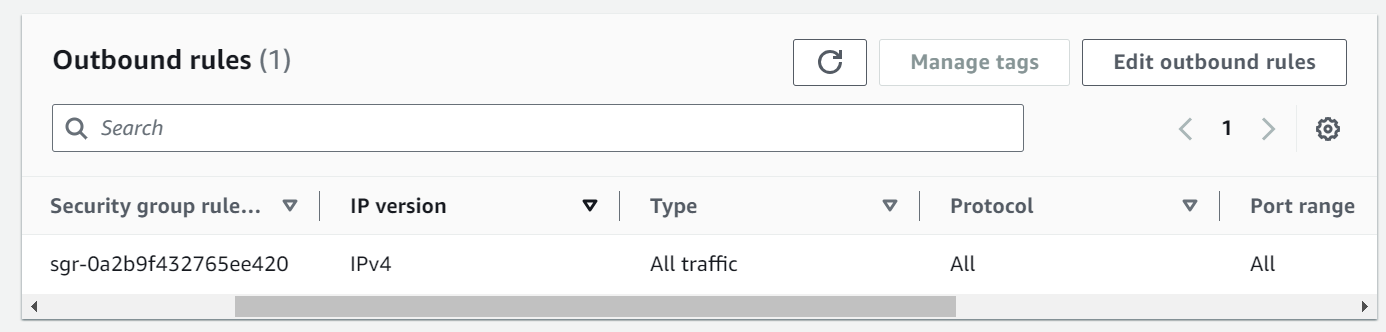
* Create and download the key pair for SSH access.

**Setting up Inbound and Outbound rules**





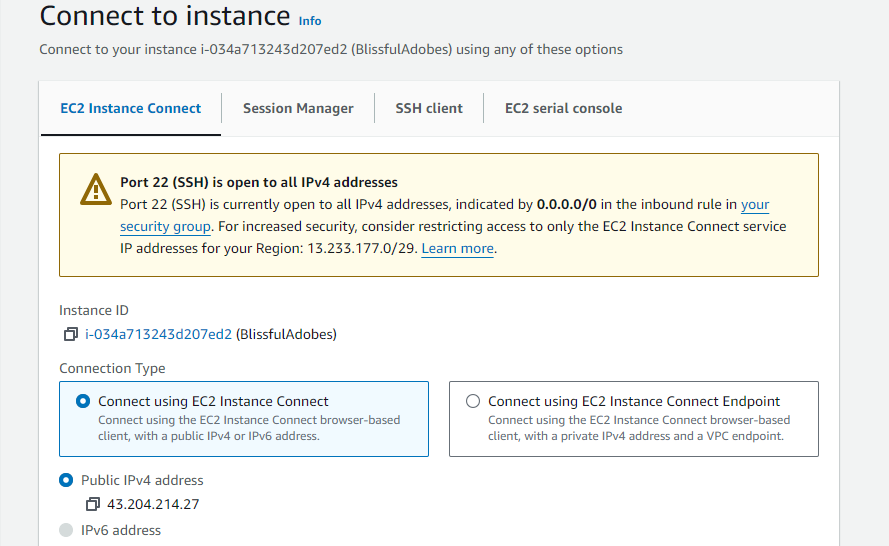
* Add Type : HTTP > Source : Anywhere
* Add Type : HTTPS > Source : Anywhere



**Milestone 5: Deployment on EC2**

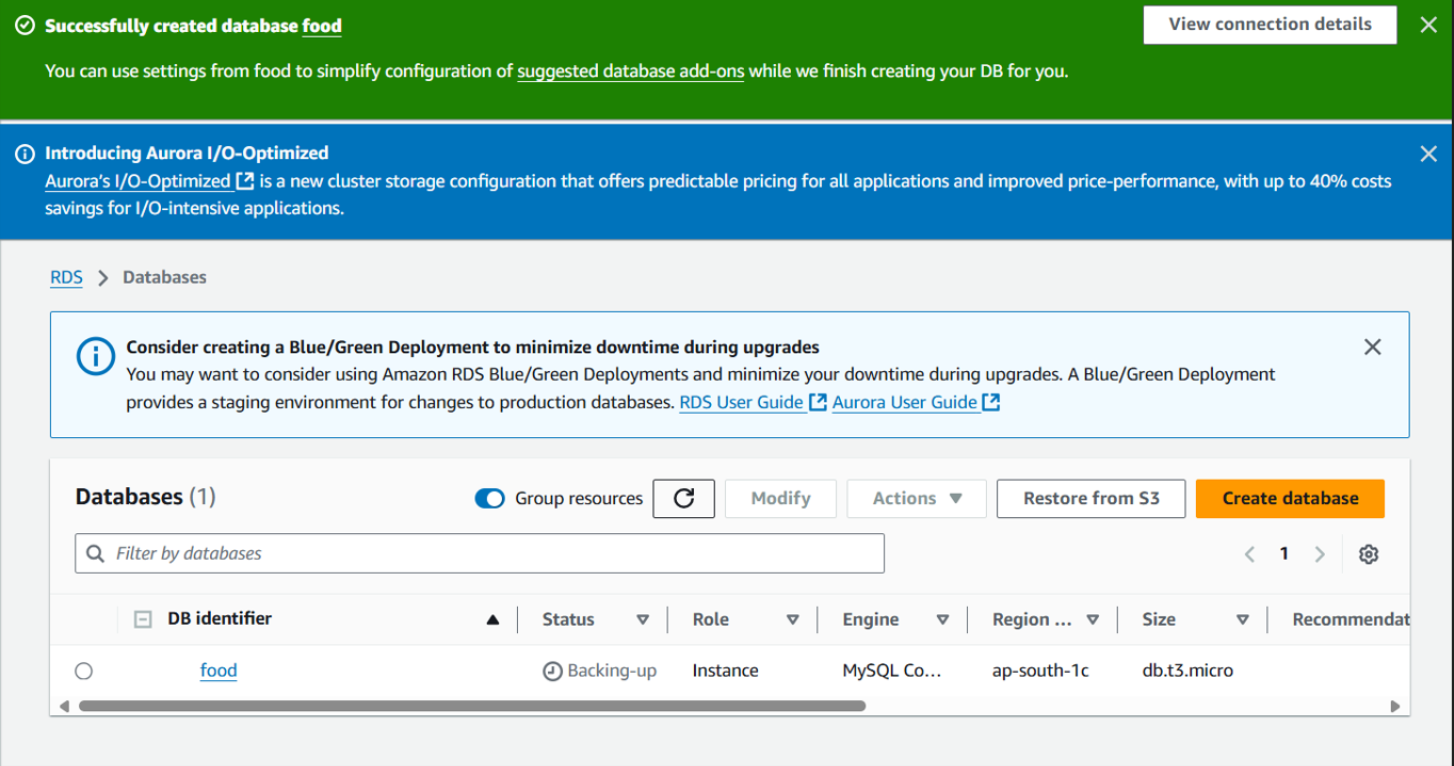
**Activity 5.1: Deploy to EC2**

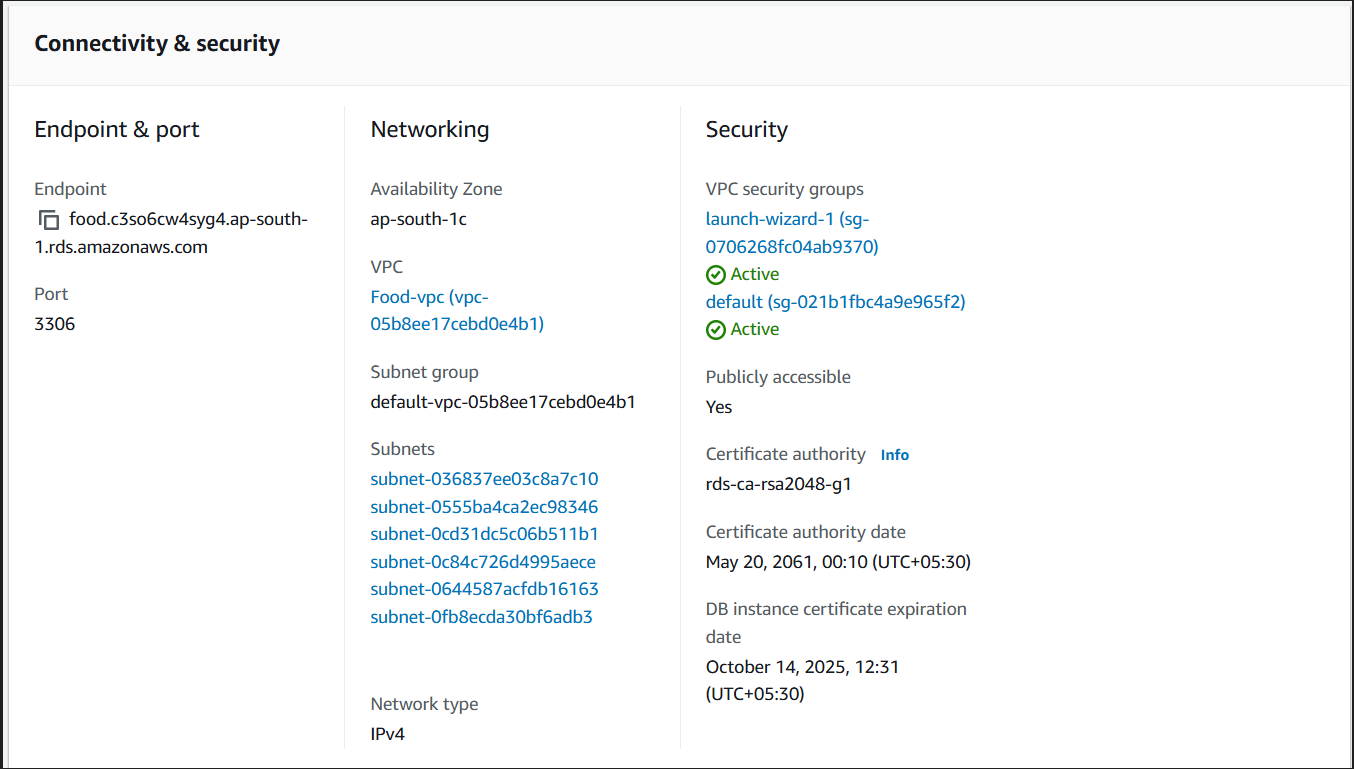
1. Connect EC2 terminal.
2. Set up any necessary environment variables, including database connection strings.
3. Configure the web server to serve your application.
4. Start your application and ensure it's accessible via the EC2 instance's public IP or domain.



**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.





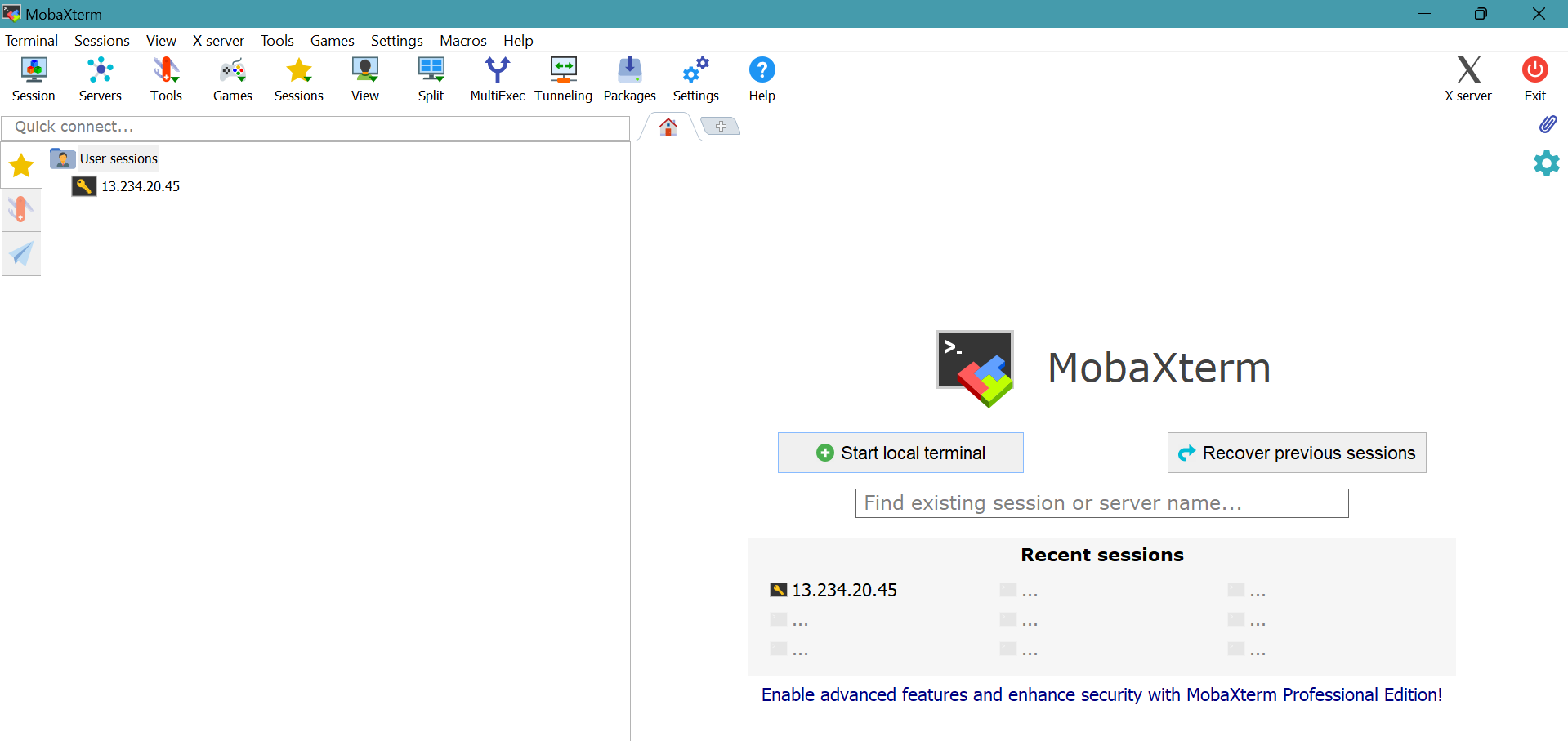
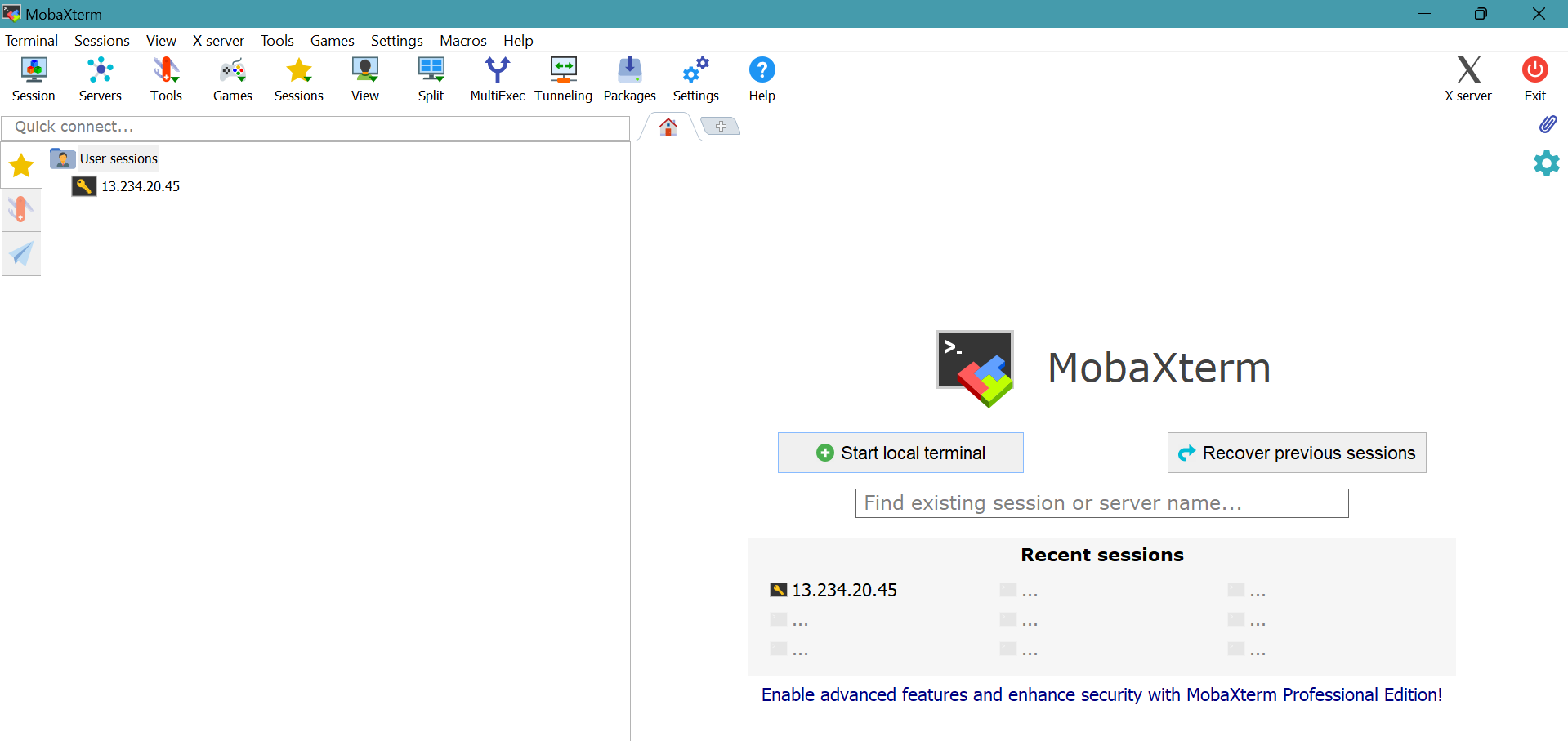
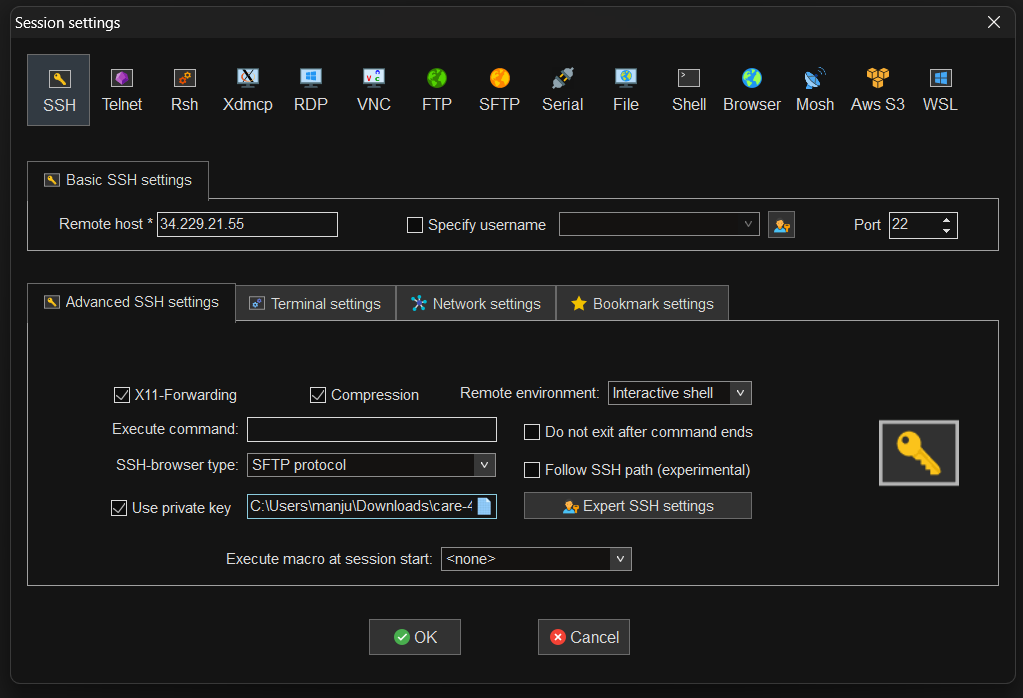
**Activity 5.3 : Launch Flask Application**

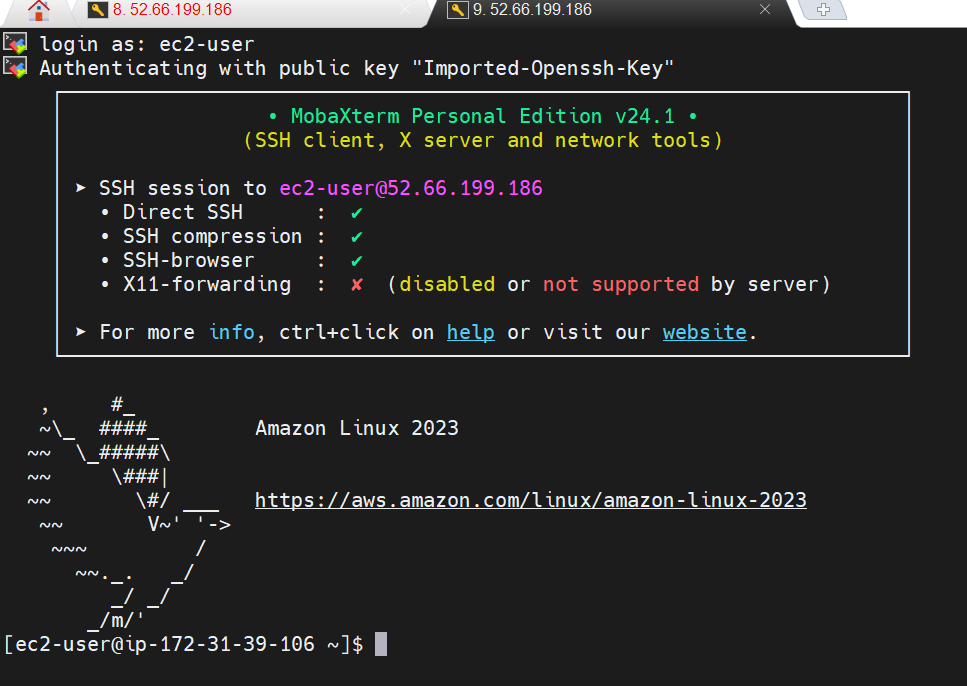
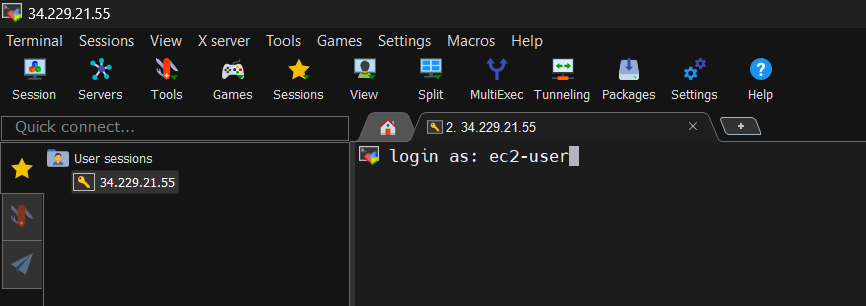
* + Run the Flask app on the EC2 instance through the SSH session to serve the order booking front-end.

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/Harnessing-AWS-for-Next-Gen-Online-Food-Delivery.git>



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

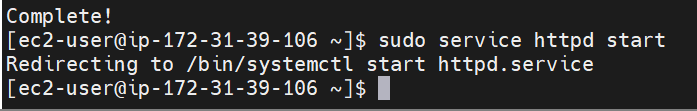
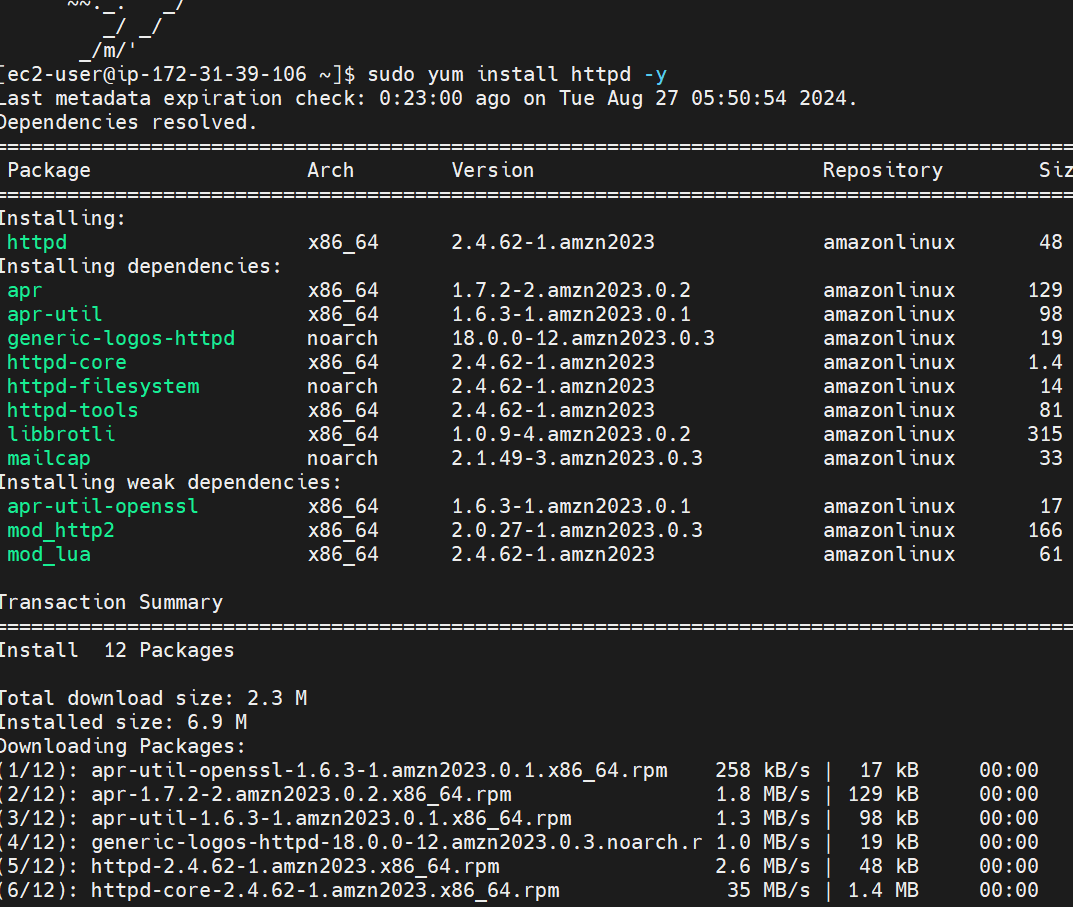
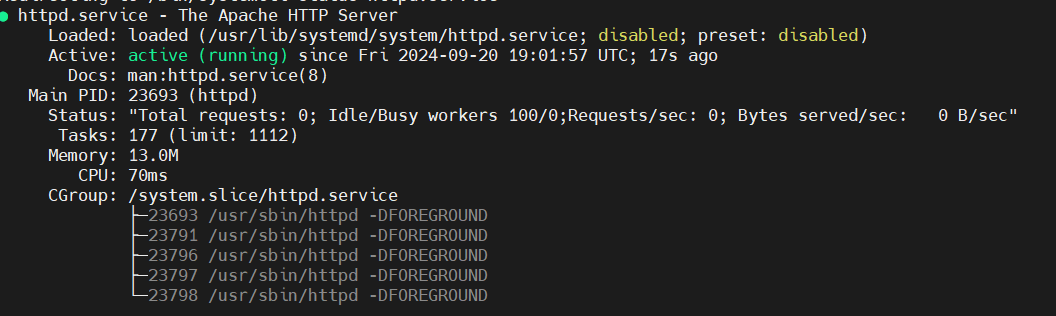
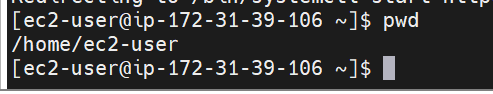
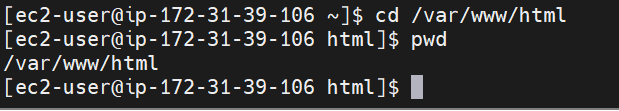
* **Activity 5.b.1: Install and Configure MobaXterm**
  + Download and install MobaXterm on your local machine.
  + Establish an SSH session with the EC2 instance by uploading the .pem key file.
  + 
  + 
  + 
  + **Activity 5.b.2: Login into the SSH Session**



**Update and Install Web Server Software**

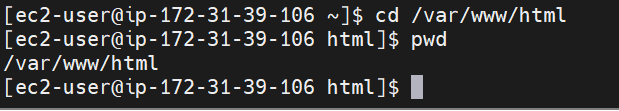
* Update package lists with sudo apt-get update (Ubuntu) or sudo yum update (Amazon Linux).

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**

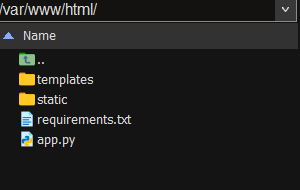
* Use MobaXterm’s SFTP functionality to transfer website files to the EC2 instance.
* Navigate to the /var/www/html directory (or the relevant directory for Nginx) and upload the files.

****

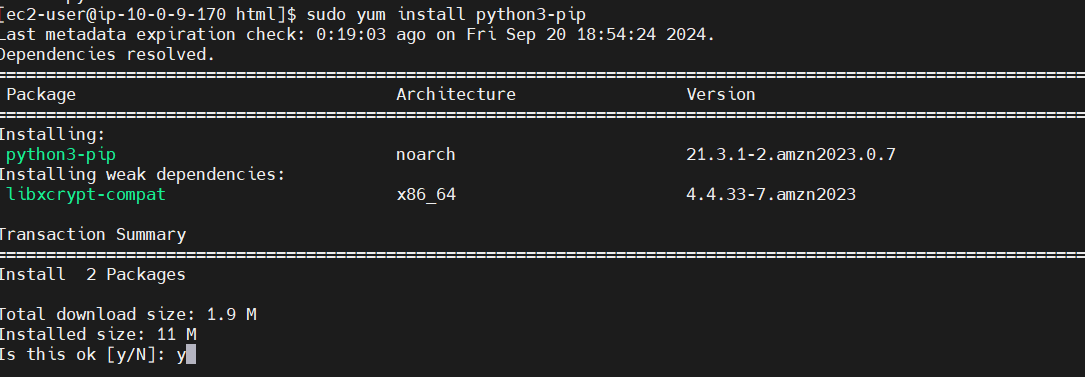
**To grant permission to accept the files into html folder.**

****

**Upload the HTML CSS and JS files**

****

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

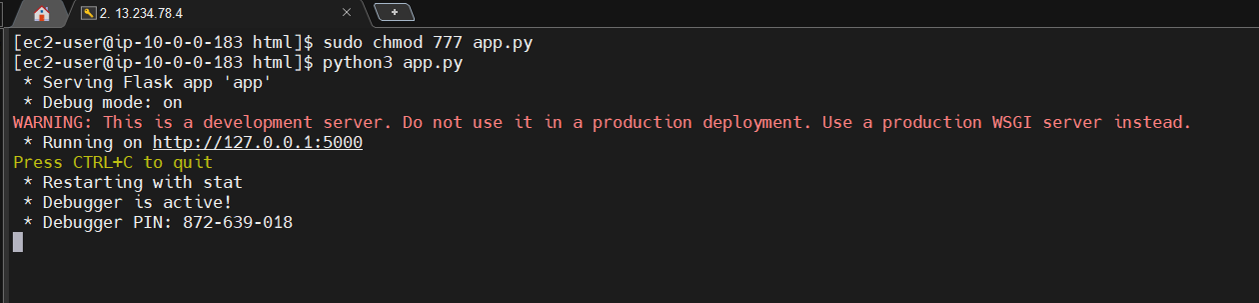
****

Install Required Libraries:

Pip install flask

pip install boto3 mysql-connector-python

**Milestone 6: Testing and Deployment**

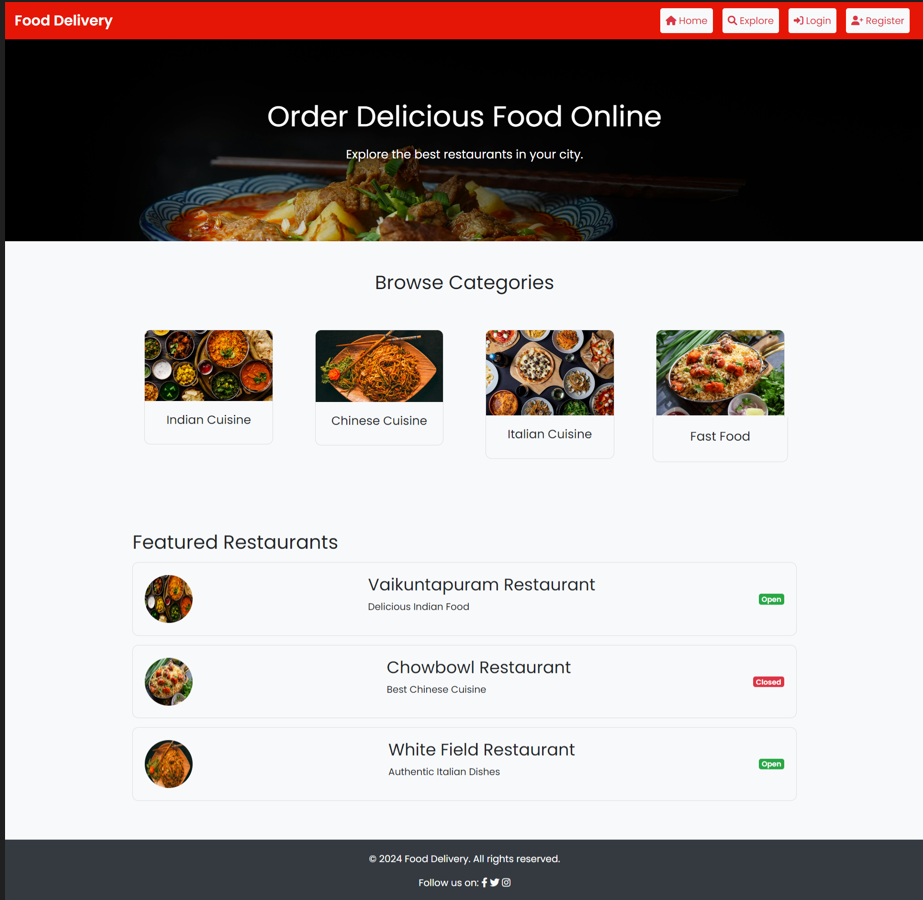
* **Activity 6.1: Functional Testing**
  + Test the FreshBasket application for functionality, including database interactions and frontend features.
  + Run the Flask app **python3 app.py**
* It will give you the link 

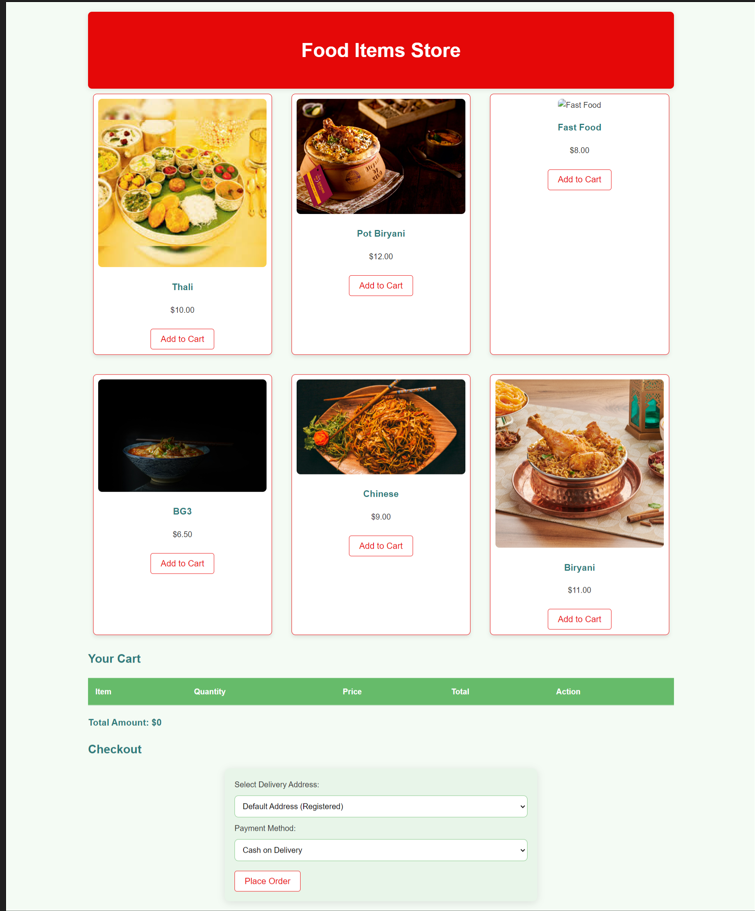
**Access the website through:  
 PublicIPs:**[**http://127.0.0.1:5000**](http://127.0.0.1:5000)

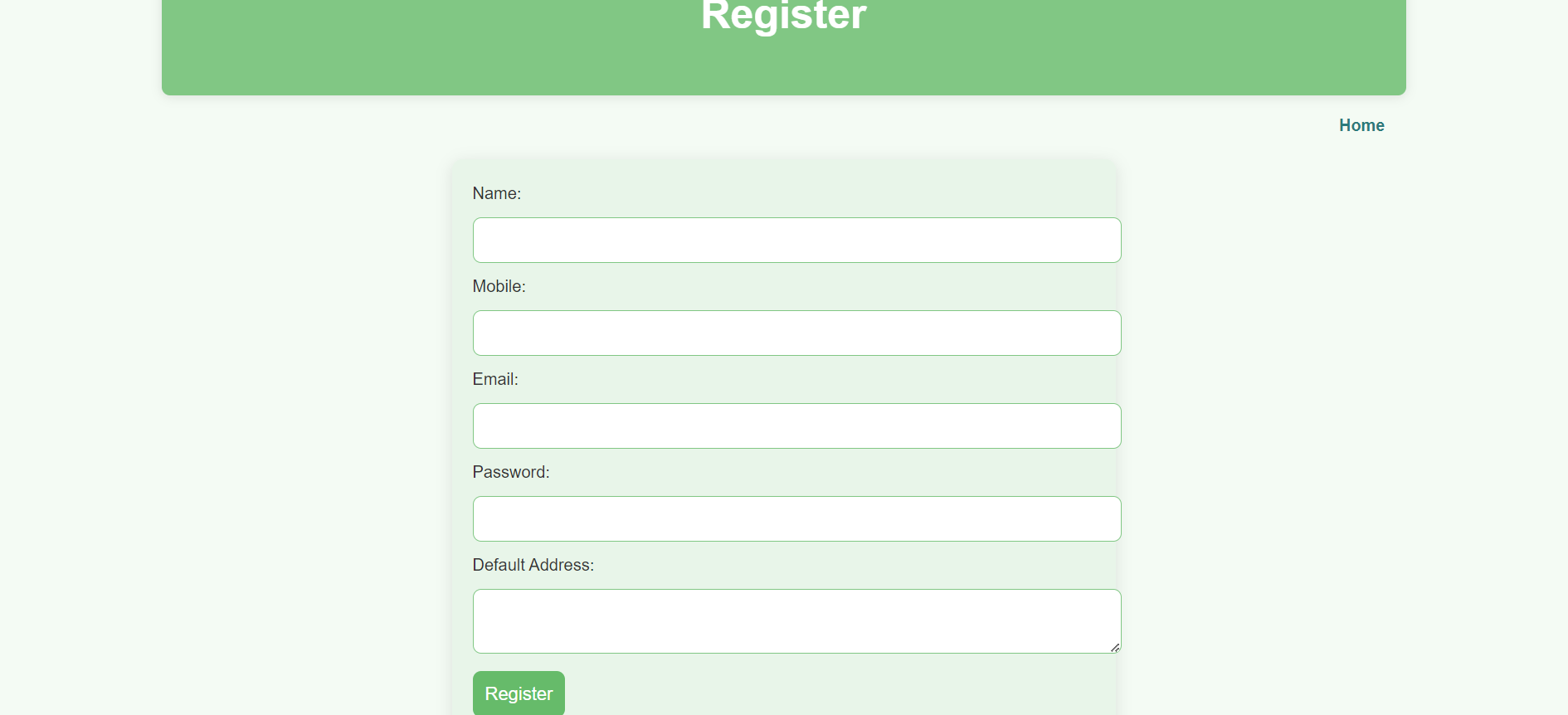
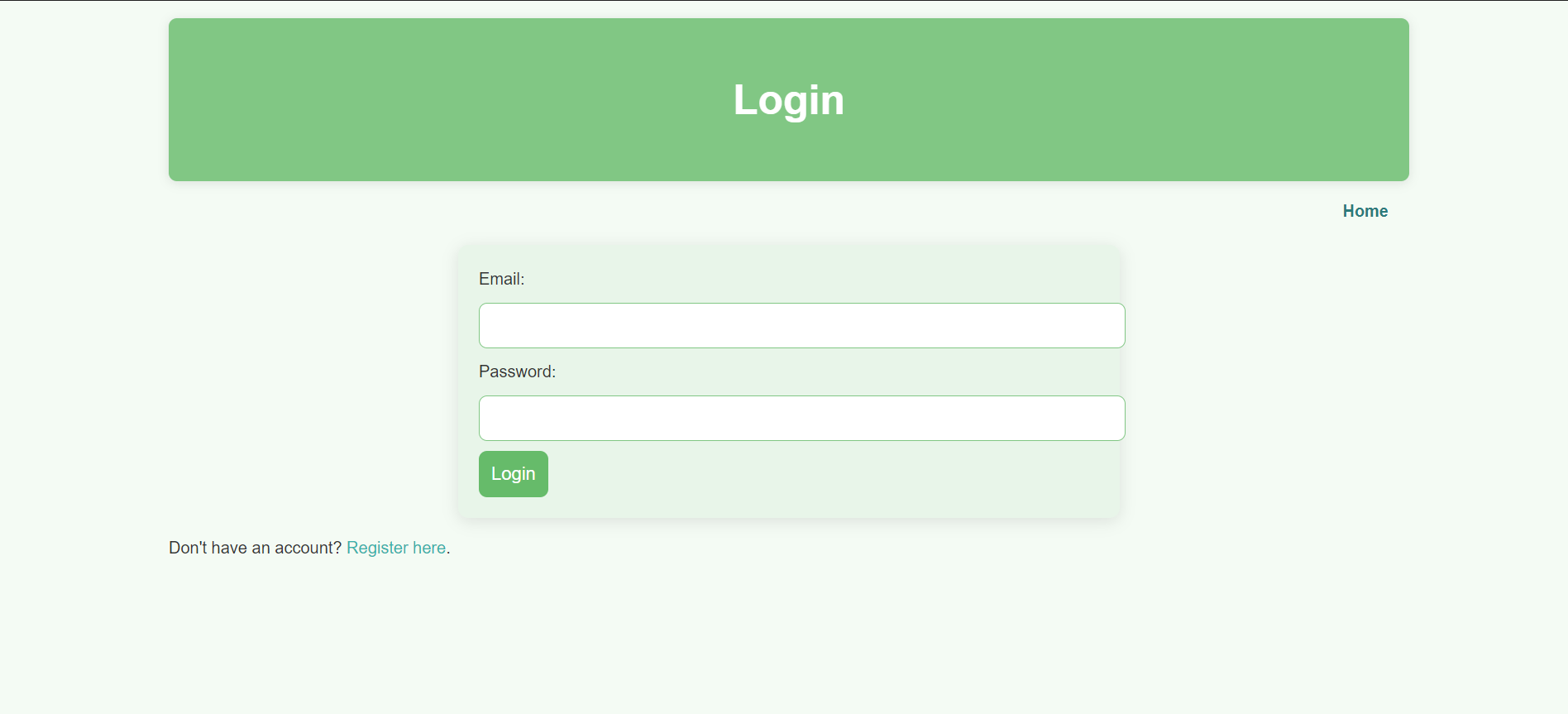
* **Activity 6.2: Deployment**
  + Deploy the application in a production environment, ensuring high availability and performance.

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

**Home:**

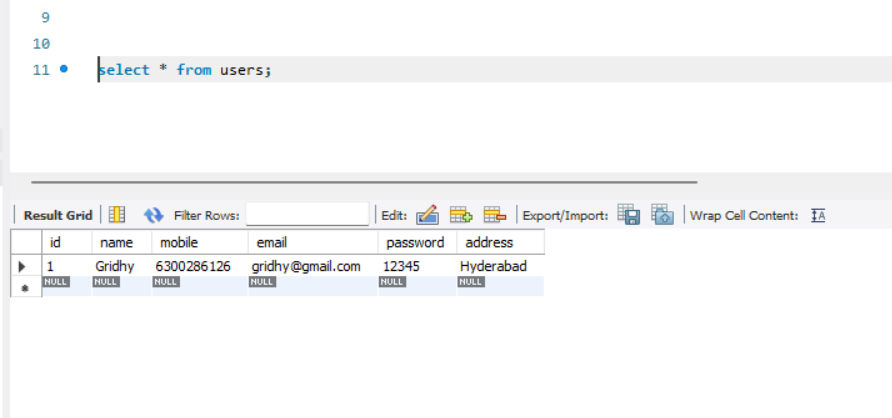
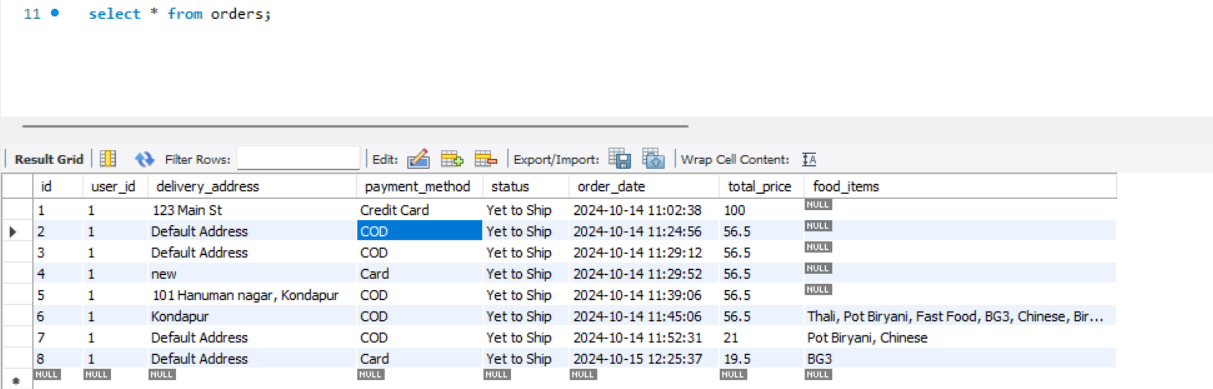
****

****

****

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

**MySql Database updations :**

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

**Conclusion:**

The Next-Gen Online Food Delivery platform showcases how AWS cloud infrastructure can be effectively utilized to create a scalable, reliable, and efficient food delivery system. By leveraging key AWS services such as EC2, RDS, and S3, this solution ensures seamless operations from order placement to delivery tracking, while providing secure data management and optimizing performance for high-traffic scenarios. The platform's ability to scale effortlessly with fluctuating demand demonstrates the technical strength and flexibility of cloud-native solutions. As the demand for fast, reliable food delivery services continues to grow, the architecture and lessons from this project will provide a solid foundation for future innovations in the industry.