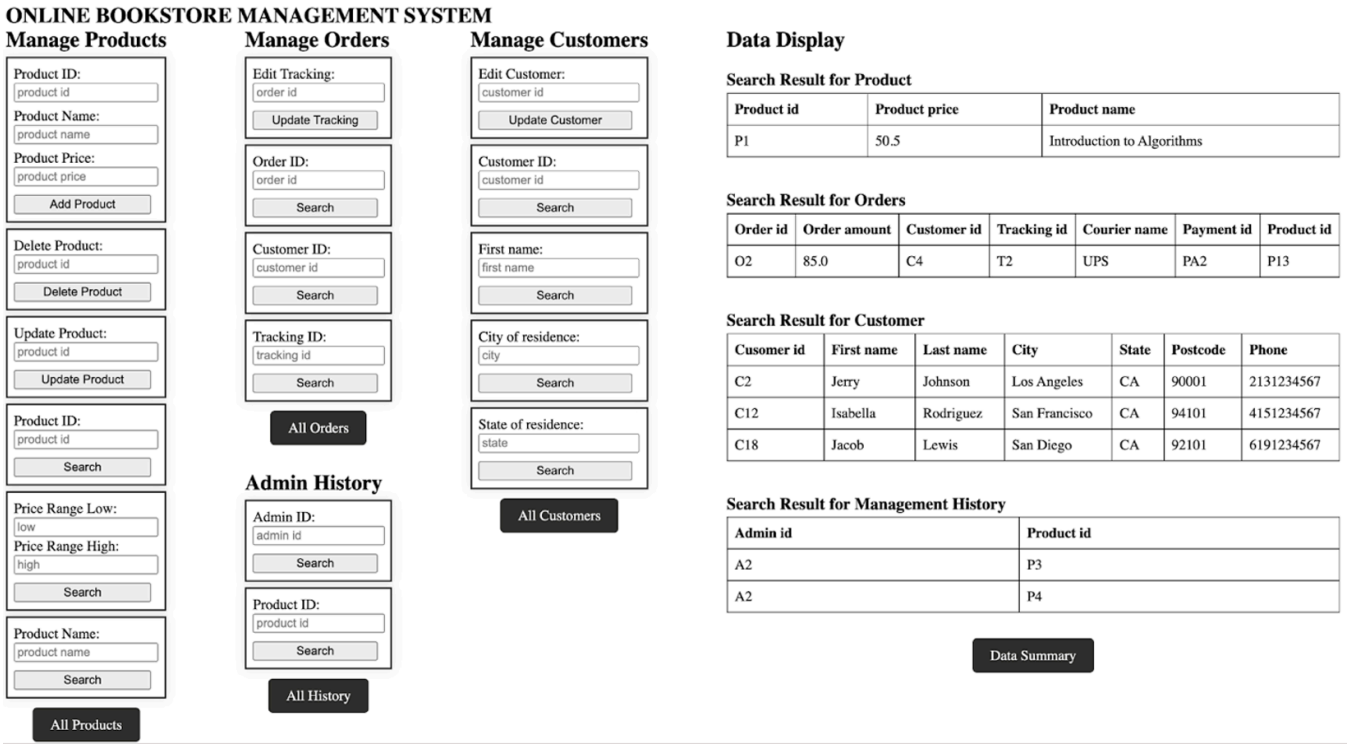
Bookstore Management System

Authors: Yiheng Liu (yl5354) Yiming Zhao (yz4579)

Application Overview



This application (http://3.136.117.103:8111/) serves as a comprehensive platform for managing the trading activities and products of the University bookstore. It empowers administrators with tools to monitor, update, and streamline store operations while maintaining a user-friendly interface. Key features include:

1. **User Authentication:**
    - Users are registered with a `user_name` and password, stored securely in the `User` dataset.
    - On first login, users receive an `Admin-id` and `Admin-name`, defining their roles and responsibilities in the management hierarchy.
2. **Product and Customer Management:**
    - Administrators have the ability to add, update, and monitor products,

including fields such as `id`, `title`, and `price`.
- Customer information, including phone numbers and addresses, is updated semi-annually to align with the start of the fall and spring semesters.
- Deletion of entities in datasets like `User`, `Admin`, `Customer`, and `Product` is restricted to predefined maintenance periods during winter and summer breaks.

3. **Application Constraints:**
    - The schema is designed to prevent unauthorized deletions outside maintenance windows, ensuring data integrity.

User Interface Design

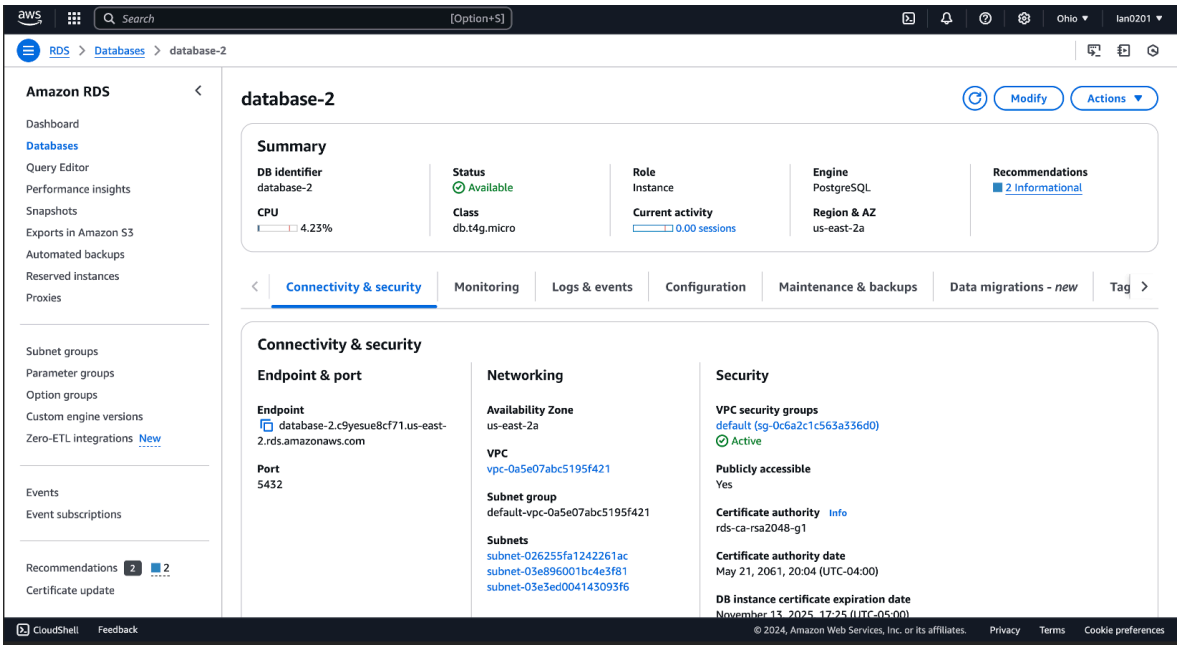
1. **Login Page:**
    - Provides users with fields to input their `username` and `password`, validated against the `User` dataset.
      `Username: Alice`
      `Password: password123`
    - Directs users to a personalized welcome page displaying: "Welcome, <>".
2. **Home Page:**
    - Features four management modules: `Manage Product`, `Manage Orders`, `Manage Customers`, and `Admin History`.
    - Includes a real-time data display section to provide immediate feedback and updates.
3. **Functional Features:**
    - **Product Management:**
        - Allows administrators to add products by entering `Product ID`, `Product Name`, and `Product Price`.
        - IDs are auto-formatted to uppercase for consistency.
        - Invalid entries trigger pop-up notifications with redirection options.
    - **Editing Data:**
        - Provides editing pages to modify tracking and customer data.

        - Displays current database values to aid accurate updates.
        - Input validation ensures compliance with required formats.
- **Search Functionality:**
        - Supports precise searches by specific fields, displaying results on the home page.
        - Facilitates quick reference with a button for comprehensive table views.

Cloud Infrastructure

Database Setup

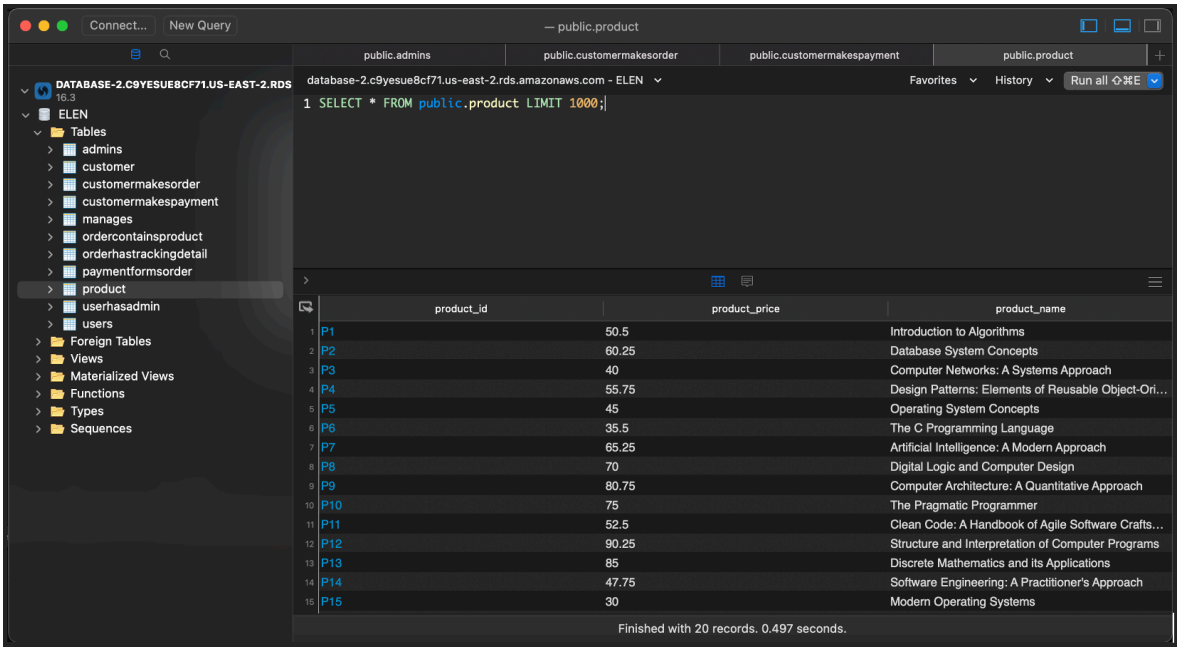
1. **Database Construction:**



    - Access the AWS RDS Console to configure a new PostgreSQL database:
        - **Instance Class:** `db.t2.micro` (Free tier eligible).
        - **Storage:** Minimal allocation to optimize cost.
        - **Credentials:** Specify a robust master username and password.
    - Ensure the database is publicly accessible for development, with

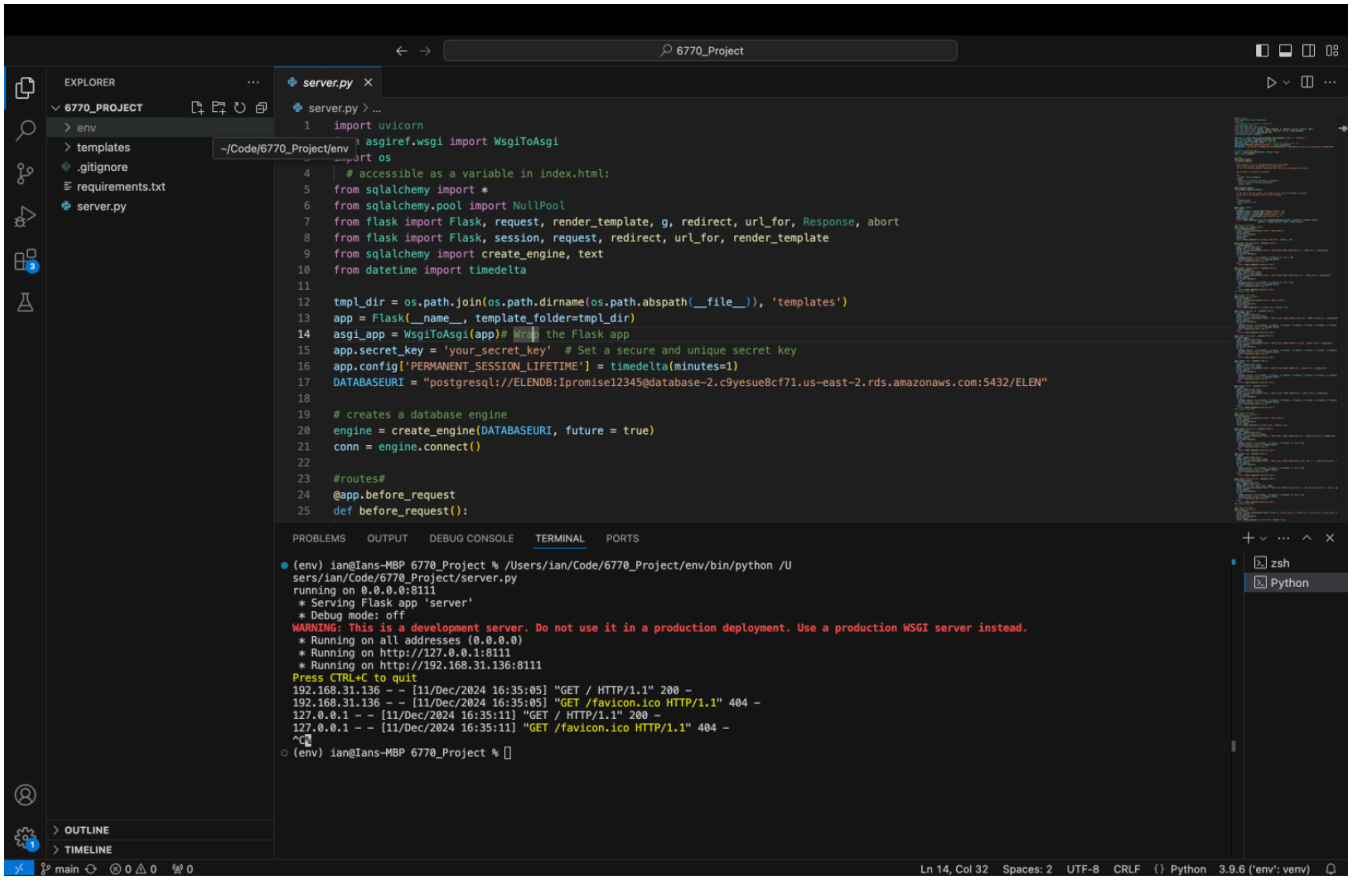
security groups restricting IP-based access.

2. **Table and Relationship Creation:**



- Use SQL Pro Studio or equivalent tools to connect to the RDS database with appropriate credentials.
- Define tables and relationships with PostgreSQL commands, incorporating constraints and references to maintain data consistency.

Backend Development (Flask)



1. **Project Initialization:**
    - Establish a Python virtual environment and install required dependencies:

      ```
      pip install flask flask_sqlalchemy flask_bcrypt flask_jwt_extended
      ```

    - Develop API endpoints for user registration, login, and CRUD operations for data management.
2. **Database Integration:**
    - Set up the database URI using the RDS endpoint:

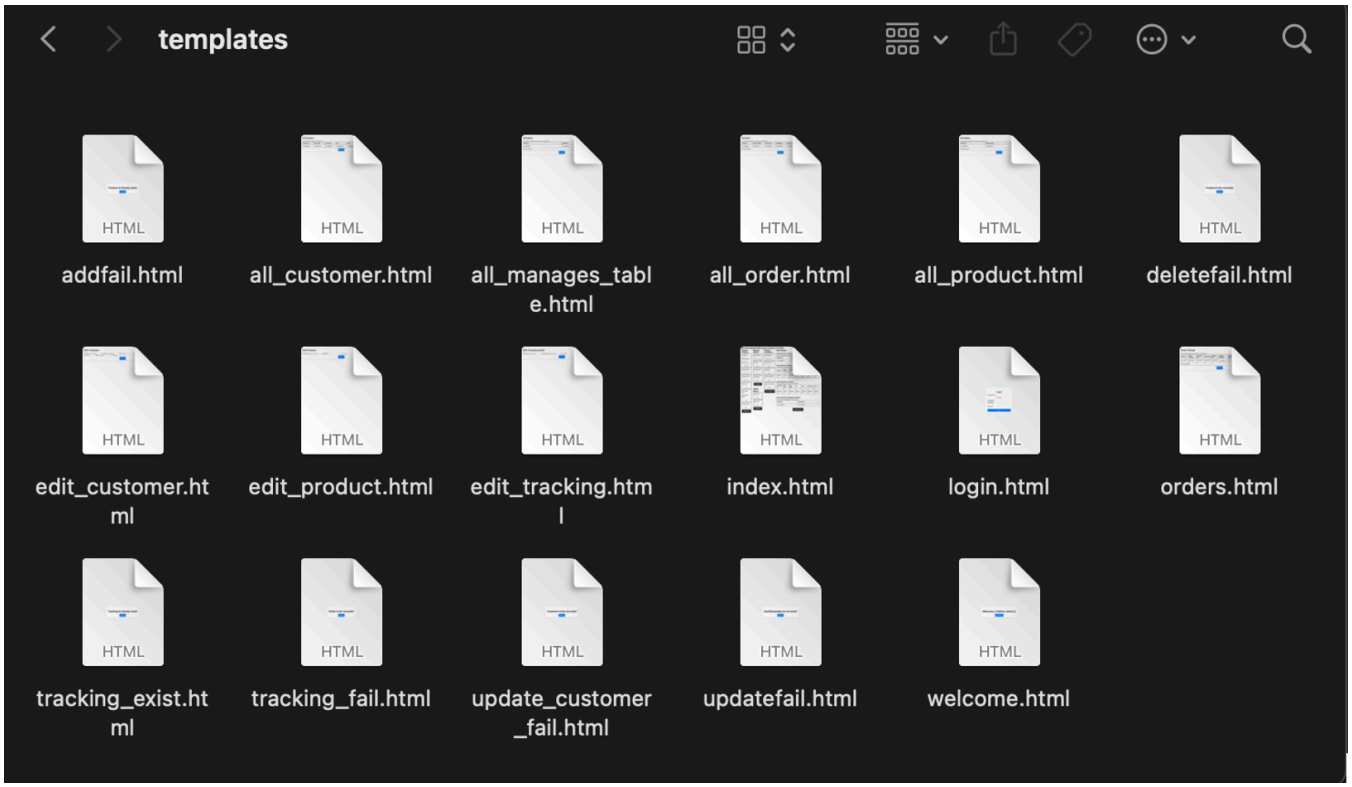
      ```
      DATABASEURI = "postgresql://ELENDB:Ipromise12345@database-2.c9yesue8
      ```

    - Implement SQLAlchemy for seamless database interactions.
3. **API Testing:**
    - Validate API functionality using tools like Postman or curl to simulate

client requests.

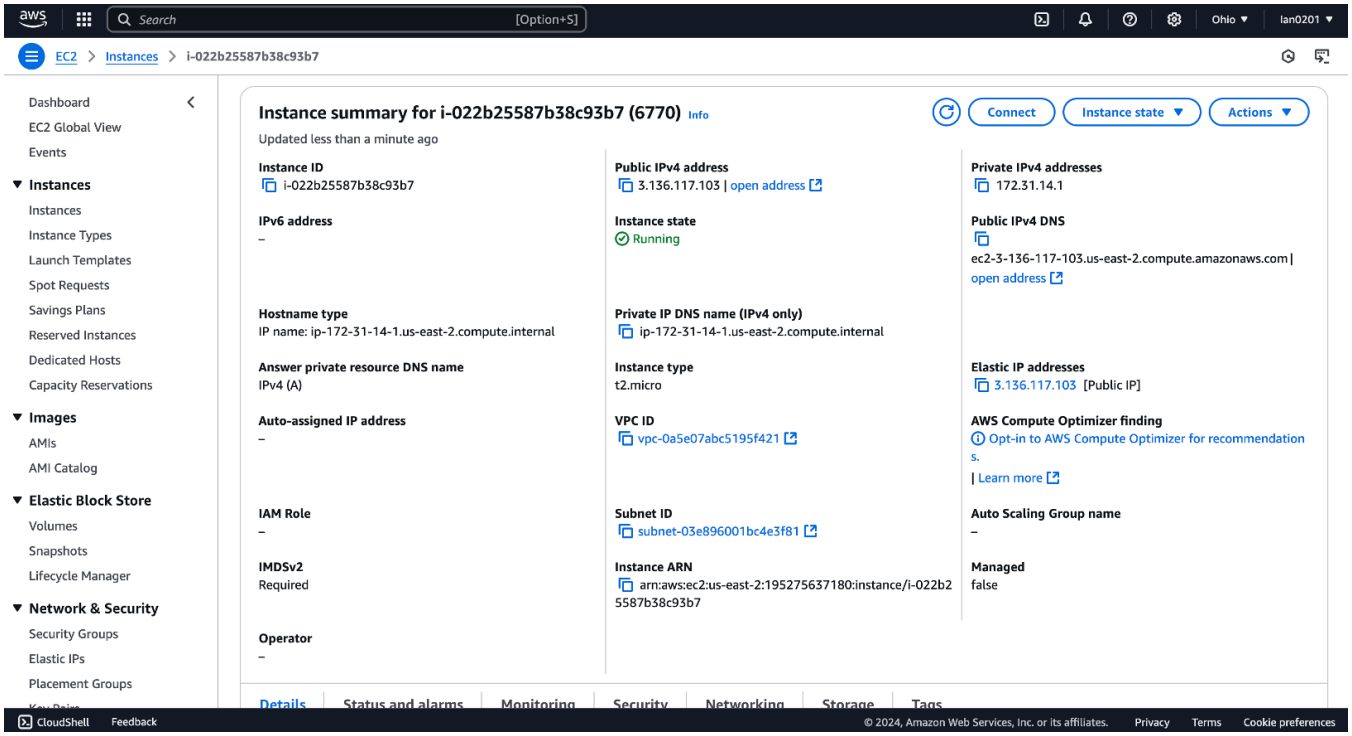
Frontend Development

The front-end interface is crafted using HTML templates and styled with CSS to ensure accessibility and usability for administrative users. Responsive design principles are applied to enhance user experience across devices.



Deployment on AWS EC2

EC2 Instance Setup



1. **Instance Initialization:**
    - Launch an Ubuntu-based EC2 instance under the free tier and connect via SSH:

```
ssh -i "<key_pair>.pem" ubuntu@<ec2_public_ip>
```

2. **Environment Preparation:**
    - Update the system and install essential packages:

```
sudo apt update
sudo apt install python3-pip python3-dev libpq-dev postgresql postgre
```

3. **Application Deployment:**
    - Clone the GitHub repository containing the project code:

```
git clone https://github.com/IAN0201/6770-Project-Bookstore-Managemer
```

    - Install application dependencies:

```
pip install -r requirements.txt
```

- Run the application locally to confirm functionality before deployment.

Production Configuration

1. **Gunicorn Setup:**
   - Define a systemd service file for Gunicorn:

     ```
     sudo nano /etc/systemd/system/gunicorn.service
     ```

     Add the following configuration:

     ```
     [Unit]
     Description=Gunicorn instance to serve Flask app
     After=network.target

     [Service]
     User=ubuntu
     Group=www-data
     WorkingDirectory=/home/ubuntu/apiv1
     Environment="PATH=/home/ubuntu/apiv1/env/bin"
     ExecStart=/home/ubuntu/apiv1/env/bin/gunicorn -w 4 -k uvicorn.workers

     [Install]
     WantedBy=multi-user.target
     ```

   - Enable and start the service to ensure persistent availability:

     ```
     sudo systemctl start gunicorn
     sudo systemctl enable gunicorn
     ```

2. **Nginx Configuration:**
   - Configure Nginx to act as a reverse proxy for Gunicorn:

     ```
     sudo nano /etc/nginx/sites-available/api
     ```

     Insert the following block:

```
server {
    listen 80;
    server_name <server_ip>;

    location / {
        proxy_pass http://127.0.0.1:8111;
    }
}
```

- Test the configuration and restart the Nginx service:

```
sudo nginx -t
sudo systemctl restart nginx
```

3. **Elastic IP Allocation:**
    - Allocate an Elastic IP to ensure a consistent public-facing address for the application.

Accessing the Application

Access the fully deployed application using the public IP: http://3.136.117.103:8111/