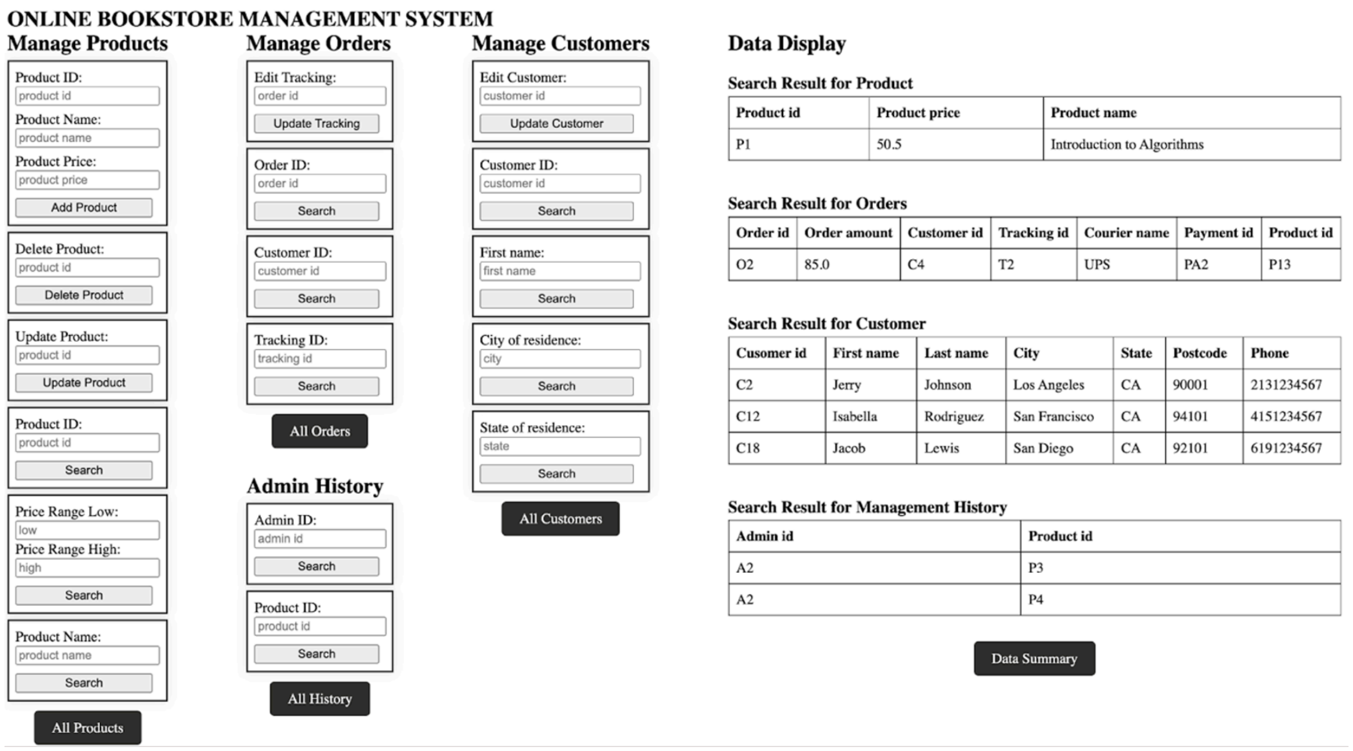
Bookstore Management System

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

1. Application Overview



Search Result for Product

Product id	Product price	Product name
P1	50.5	Introduction to Algorithms

Search Result for Orders

Order id	Order amount	Customer id	Tracking id	Courier name	Payment id	Product id
O2	85.0	C4	T2	UPS	PA2	P13

Search Result for Customer

Cusomer id	First name	Last name	City	State	Postcode	Phone
C2	Jerry	Johnson	Los Angeles	CA	90001	2131234567
C12	Isabella	Rodriguez	San Francisco	CA	94101	4151234567
C18	Jacob	Lewis	San Diego	CA	92101	6191234567

Search Result for Management History

Admin id	Product id
A2	P3
A2	P4

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.

2. 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.

3. Major Software Components

System Architecture

1. **Frontend:**
    - Provides login and management interfaces using responsive design principles.
    - Interacts with backend APIs for data operations.
2. **Backend:**
    - Implements user authentication, CRUD operations, and API endpoints using Flask.
    - Manages database operations via SQLAlchemy ORM.
3. **Database:**
    - PostgreSQL database hosted on AWS RDS.
    - Contains structured tables for `Users`, `Products`, `Orders`, and `Admin` data.
4. **Cloud Deployment:**
    - AWS EC2 instance serves as the primary hosting environment.
    - Nginx acts as a reverse proxy, and Gunicorn serves the Flask application.

4. Technologies Used

Frontend:

- HTML, CSS, and JavaScript for web interfaces.

Backend:

- **Flask Framework:** Python-based microframework.
- **SQLAlchemy:** Object Relational Mapper (ORM) for database operations.
- **Gunicorn:** Python WSGI HTTP server.

Database:

- **PostgreSQL:** Relational database management system hosted on AWS RDS.

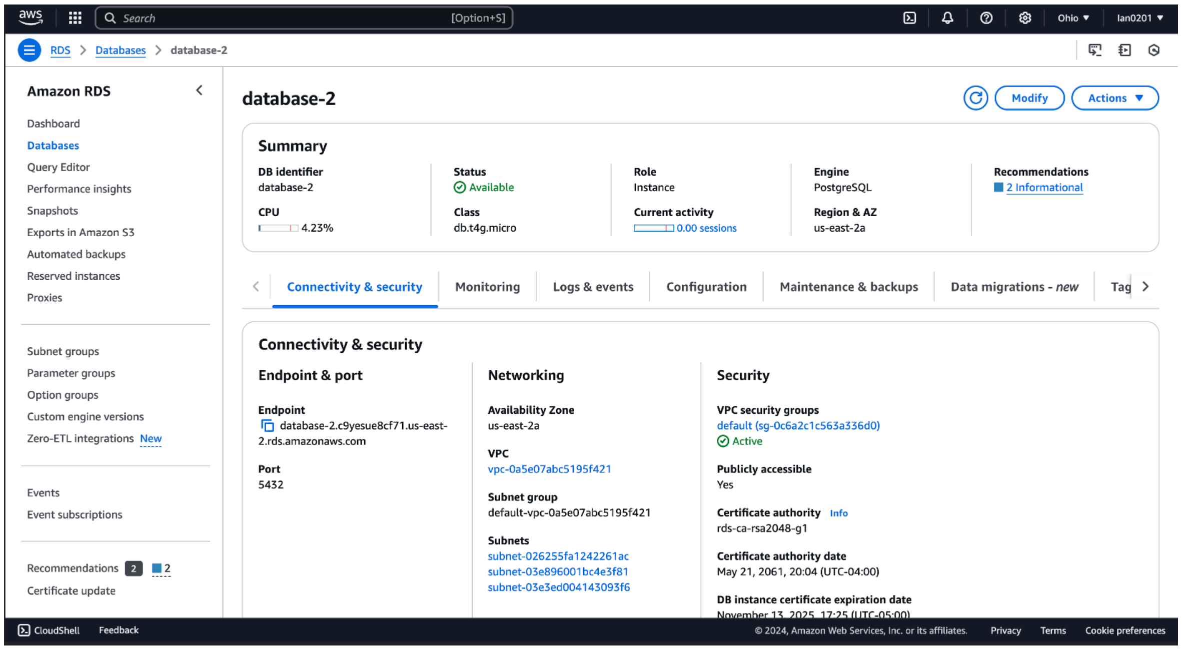
Cloud Services:

- **AWS EC2:** Virtual server for hosting the application.
- **AWS RDS:** Managed database service for PostgreSQL.
- **AWS Elastic IP:** Ensures consistent public IP for the EC2 instance.
- **Nginx:** High-performance HTTP server for reverse proxying.

5. Cloud Infrastructure

Database Setup

1. **Database Construction:**



database-2

Summary

DB identifier	Status	Role	Engine	Recommendations
database-2	Available	Instance	PostgreSQL	2 Informational
CPU	**Class**	**Current activity**	**Region & AZ**	
4.23%	db.t4g.micro	0.00 sessions	us-east-2a	

Connectivity & security

Endpoint & port

Endpoint
database-2.c9yesue8cf71.us-east-2.rds.amazonaws.com

Port
5432

Networking

Availability Zone
us-east-2a

VPC
vpc-0a5e07abc5195f421

Subnet group
default-vpc-0a5e07abc5195f421

Subnets
subnet-026255fa1242261ac
subnet-03e896001bc4e3f81
subnet-03e3ed004143093f6

Security

VPC security groups
default (sg-0c6a2c1c563a336d0)
Active

Publicly accessible
Yes

Certificate authority Info
rds-ca-rsa2048-g1

Certificate authority date
May 21, 2061, 20:04 (UTC-04:00)

DB instance certificate expiration date
November 13, 2025, 17:25 (UTC-05:00)

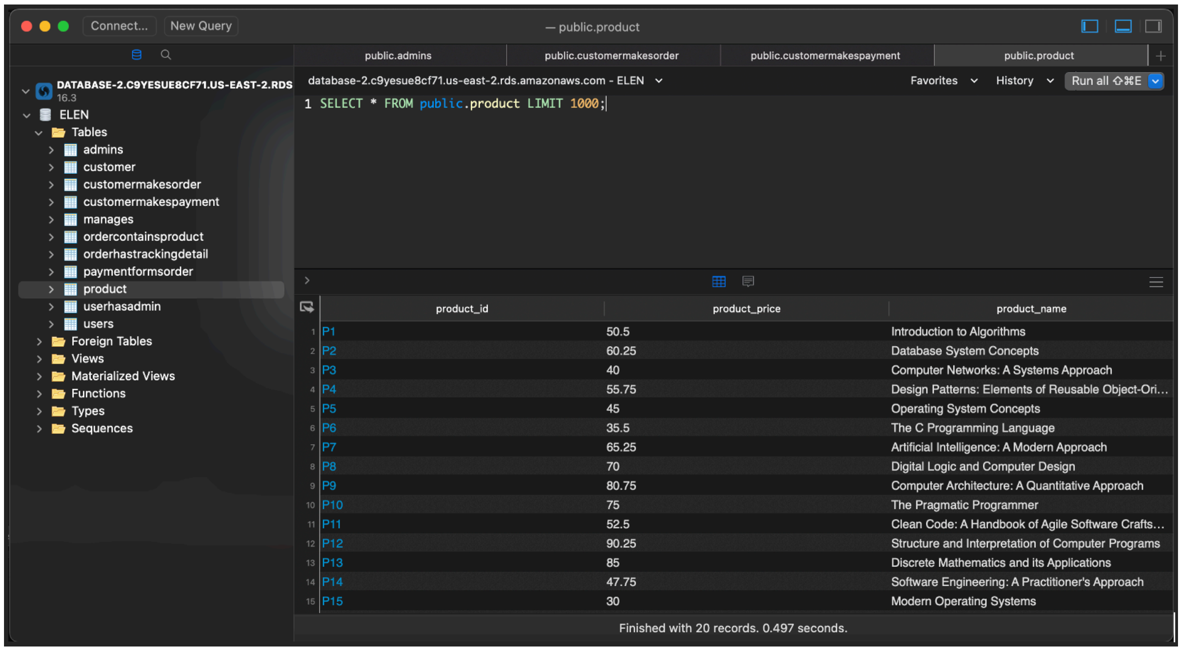
© 2024, Amazon Web Services, Inc. or its affiliates.

- 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:**



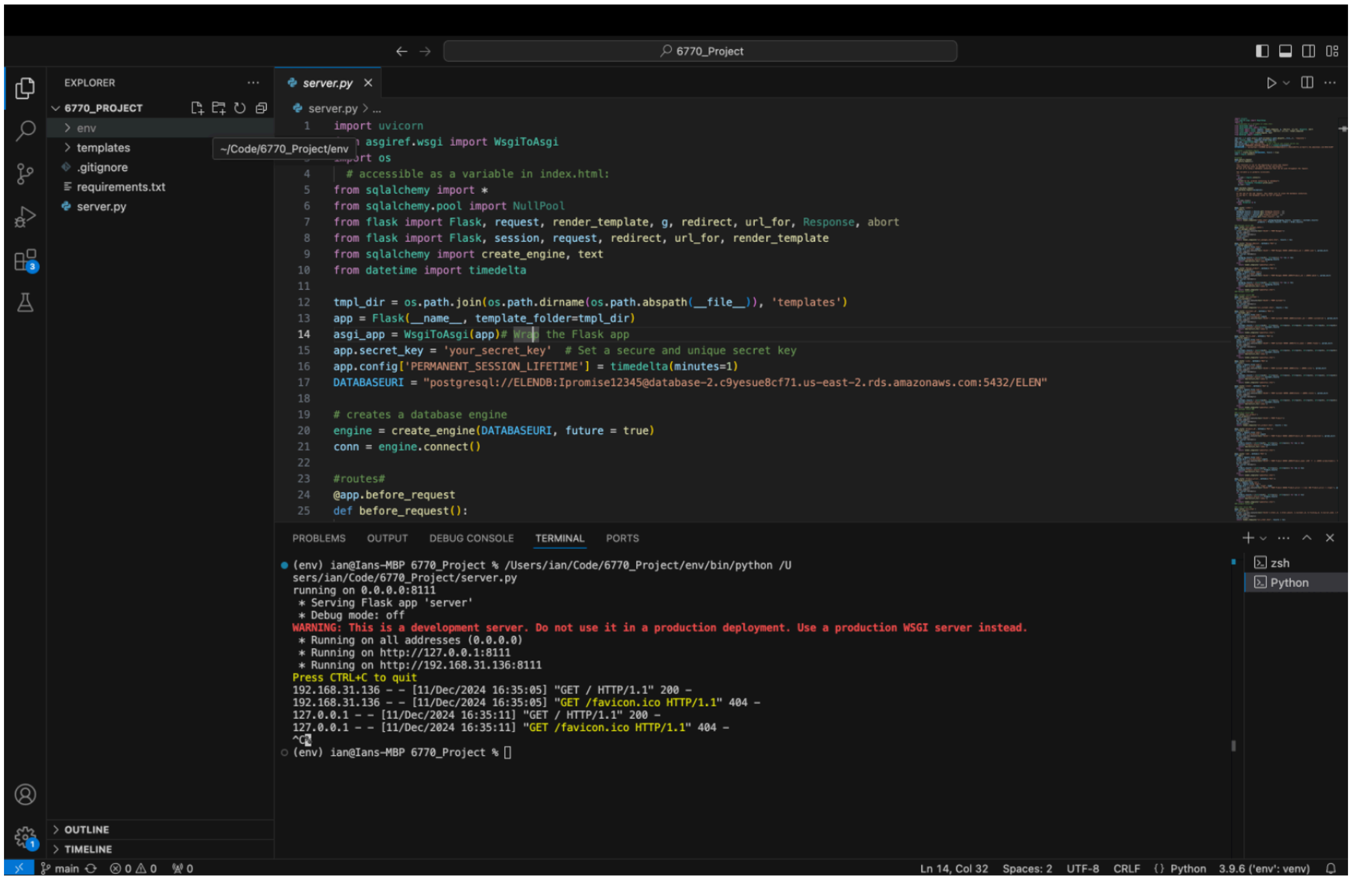
```
SELECT * FROM public.product LIMIT 1000;
```

	product_id	product_price	product_name
1	P1	50.5	Introduction to Algorithms
2	P2	60.25	Database System Concepts
3	P3	40	Computer Networks: A Systems Approach
4	P4	55.75	Design Patterns: Elements of Reusable Object-Ori…
5	P5	45	Operating System Concepts
6	P6	35.5	The C Programming Language
7	P7	65.25	Artificial Intelligence: A Modern Approach
8	P8	70	Digital Logic and Computer Design
9	P9	80.75	Computer Architecture: A Quantitative Approach
10	P10	75	The Pragmatic Programmer
11	P11	52.5	Clean Code: A Handbook of Agile Software Crafts…
12	P12	90.25	Structure and Interpretation of Computer Programs
13	P13	85	Discrete Mathematics and its Applications
14	P14	47.75	Software Engineering: A Practitioner's Approach
15	P15	30	Modern Operating Systems

Finished with 20 records. 0.497 seconds.

- 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.

6. Backend Development (Flask)



```python
import uvicorn
  n asgiref.wsgi import WsgiToAsgi
  ort os
  # accessible as a variable in index.html:
from sqlalchemy import *
from sqlalchemy.pool import NullPool
from flask import Flask, request, render_template, g, redirect, url_for, Response, abort
from flask import Flask, session, request, redirect, url_for, render_template
from sqlalchemy import create_engine, text
from datetime import timedelta

tmpl_dir = os.path.join(os.path.dirname(os.path.abspath(__file__)), 'templates')
app = Flask(__name__, template_folder=tmpl_dir)
asgi_app = WsgiToAsgi(app)# Wrap the Flask app
app.secret_key = 'your_secret_key'  # Set a secure and unique secret key
app.config['PERMANENT_SESSION_LIFETIME'] = timedelta(minutes=1)
DATABASEURI = "postgresql://ELENDB:Ipromise12345@database-2.c9yesue8cf71.us-east-2.rds.amazonaws.com:5432/ELEN"

# creates a database engine
engine = create_engine(DATABASEURI, future = true)
conn = engine.connect()

#routes#
@app.before_request
def before_request():
```

```
(env) ian@Ians-MBP 6770_Project % /Users/ian/Code/6770_Project/env/bin/python /U
sers/ian/Code/6770_Project/server.py
running on 0.0.0.0:8111
 * Serving Flask app 'server'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:8111
 * Running on http://192.168.31.136:8111
Press CTRL+C to quit
192.168.31.136 - - [11/Dec/2024 16:35:05] "GET / HTTP/1.1" 200 -
192.168.31.136 - - [11/Dec/2024 16:35:05] "GET /favicon.ico HTTP/1.1" 404 -
127.0.0.1 - - [11/Dec/2024 16:35:11] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [11/Dec/2024 16:35:11] "GET /favicon.ico HTTP/1.1" 404 -
^C
(env) ian@Ians-MBP 6770_Project %
```

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.c9yesue8c
     .us-east-2.rds.amazonaws.com:5432/ELEN"
     ```

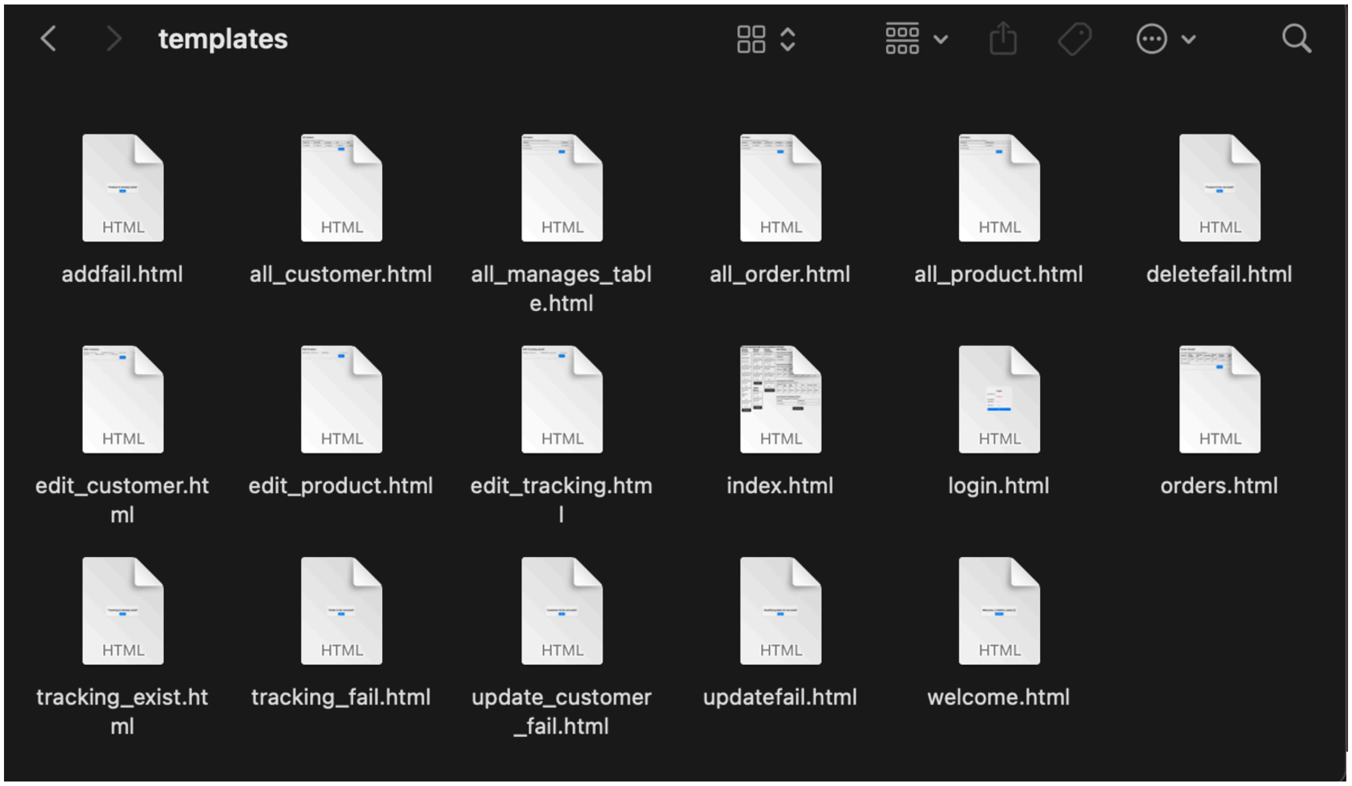
   - Implement SQLAlchemy for seamless database interactions.

3. **API Testing:**
   - Validate API functionality using tools like Postman or curl to simulate client requests.

---

7. Frontend Development

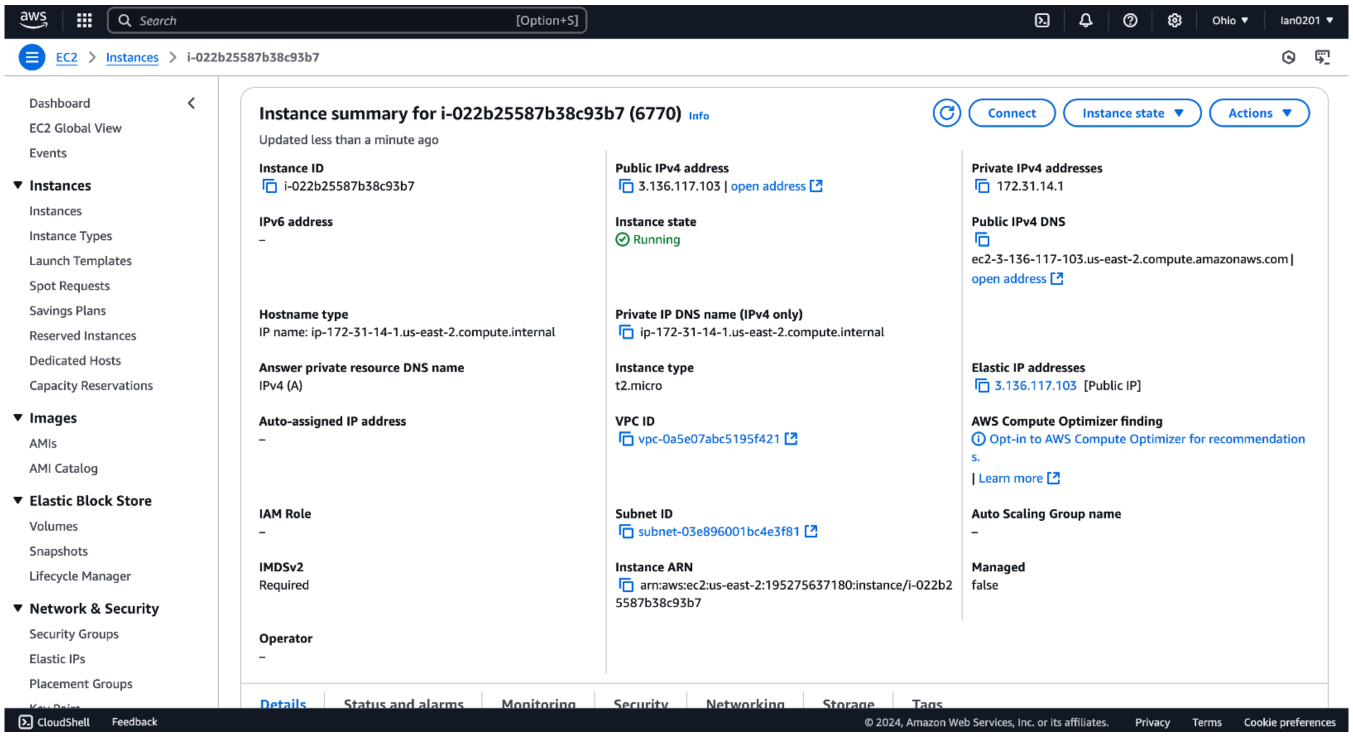
The front-end interface is crafted using HTML templates and styled with CSS to ensure accessibility and usability for administrative users. Responsive design principles are applied to enhance user experience across devices.



---

8. Deployment on AWS EC2

EC2 Instance Setup



Instance summary for i-022b25587b38c93b7 (6770) Info

Updated less than a minute ago

Instance ID i-022b25587b38c93b7	**Public IPv4 address** 3.136.117.103 | open address	**Private IPv4 addresses** 172.31.14.1
IPv6 address –	**Instance state** Running	**Public IPv4 DNS** ec2-3-136-117-103.us-east-2.compute.amazonaws.com | open address
Hostname type IP name: ip-172-31-14-1.us-east-2.compute.internal	**Private IP DNS name (IPv4 only)** ip-172-31-14-1.us-east-2.compute.internal	
Answer private resource DNS name IPv4 (A)	**Instance type** t2.micro	**Elastic IP addresses** 3.136.117.103 [Public IP]
Auto-assigned IP address –	**VPC ID** vpc-0a5e07abc5195f421	**AWS Compute Optimizer finding** Opt-in to AWS Compute Optimizer for recommendations. | Learn more
IAM Role –	**Subnet ID** subnet-03e896001bc4e3f81	**Auto Scaling Group name** –
IMDSv2 Required	**Instance ARN** arn:aws:ec2:us-east-2:195275637180:instance/i-022b25587b38c93b7	**Managed** false
Operator –		

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 postgresql-contrib nginx curl
   ```

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

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

- 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
     UvicornWorker -b 0.0.0.0:8111 server:asgi_app

     [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;
      }
  }
  ```

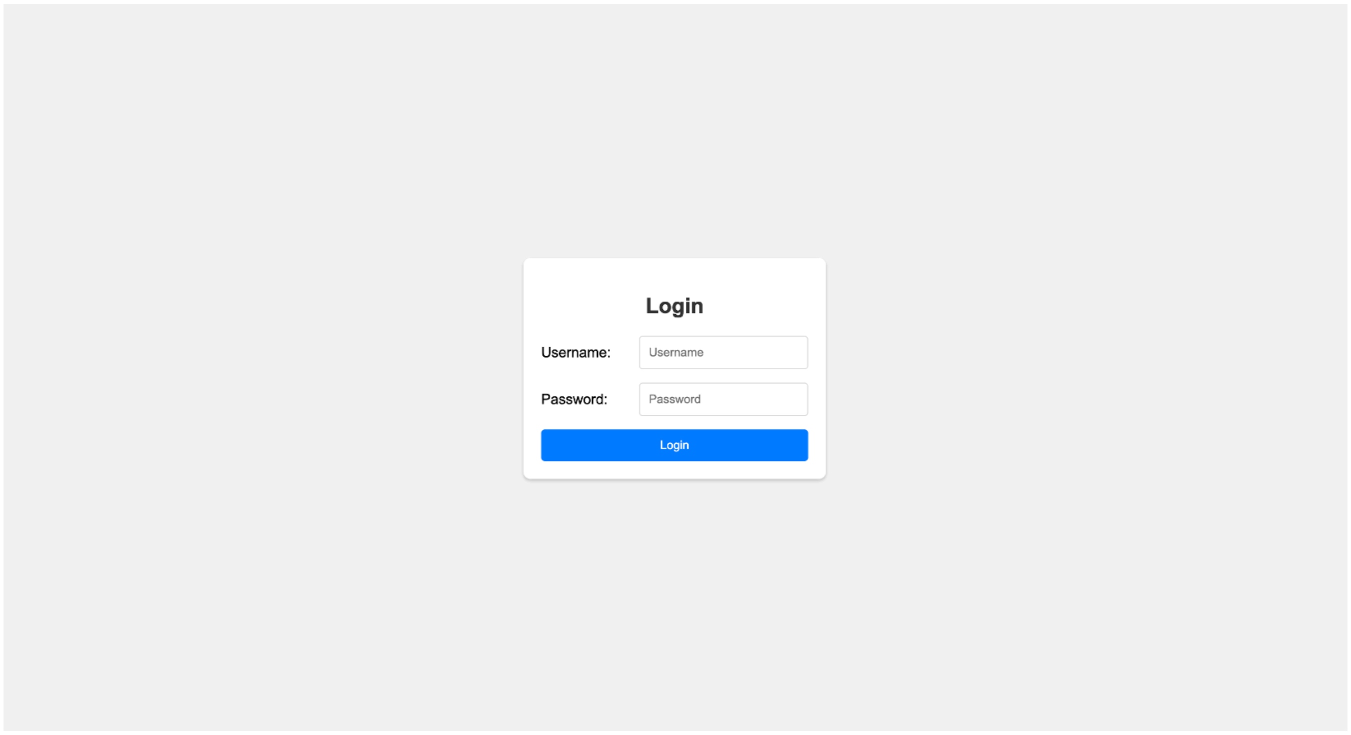
- Test the configuration and restart the Nginx service:

  ```
  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/


Login
Username:
Username
Password:
Password
Login