

**American University of Beirut**

**Faculty of Engineering and Architecture**

**Department of Electrical and Computer Engineering**

A Project of Backend Web Services for an E-Commerce Website

By

**Omar Succar**

**ID: 202101374**

# (oms21@mail.aub.edu)

**Ibrahim El Khansa**

**ID:** **202151253**

# (ike06@mail.aub.edu)

A REPORT

submitted to Dr. Iman Ghalayini in partial fulfillment of the requirements of the final project for the course EECE435L – Software Tools

To be submitted by December 2024

Table of Contents

Introduction ................................................................................................................................................................................................. 4

Project Overview ..................................................................................................................................................................................... 4

Project Objective ................................................................................................................................................................................ 4

System Architecture .................................................................................................................................................................................... 4

Service 1 Description “Customers” (Ibrahim’s Work) ....................................................................................................................... 5

Service 2 Description “Inventory” (Omar’s Work) ............................................................................................................................ 5

Service 3 Description “Sales” (Omar’s Work) ................................................................................................................................... 5 Service 4 Description “Reviews” (Ibrahim’s Work) .......................................................................................................................... 5

Implementation and DetailsService 1 – Customers (Ibrahim’s Work)............................................................................................... 6

Functionality:.................................................................................................................................................................................. 6

Challenges Faced:........................................................................................................................................................................... 6

APIs: ............................................................................................................................................................................................... 6

Service 2 – Inventory (Omar’s Work) ................................................................................................................................................ 6

Functionality:.................................................................................................................................................................................. 6

Challenges Faced:........................................................................................................................................................................... 6

APIs: ............................................................................................................................................................................................... 7

Service 3 – Sales (Omar’s Work) ....................................................................................................................................................... 7

Functionality:.................................................................................................................................................................................. 7

Challenges Faced:........................................................................................................................................................................... 7

APIs: ............................................................................................................................................................................................... 7

Service 4 – Reviews (Ibrahim’s Work) .............................................................................................................................................. 7

Functionality:.................................................................................................................................................................................. 7

Challenges Faced:........................................................................................................................................................................... 8

APIs: ............................................................................................................................................................................................... 8

Service: Authentication (Ibrahim’s Work) ......................................................................................................................................... 8

Functionality ................................................................................................................................................................................... 8

Challenges Faced ............................................................................................................................................................................ 8

API Documentation ........................................................................................................................................................................ 8

API Documentation (Screenshots) .......................................................................................................................................................... 9

Customer Service: ............................................................................................................................................................................... 9

Inventory Service .............................................................................................................................................................................. 12

Sales Service ..................................................................................................................................................................................... 14

Review Service ................................................................................................................................................................................. 17

Auth Service (Available for All Ports) ............................................................................................................................................. 19

Review Moderation Service ............................................................................................................................................................. 21

Review Validation & Sanitization Service ....................................................................................................................................... 23

Database Design Error Handling and Validation ...................................................................................................................................... 25

1. Error Management ......................................................................................................................................................................... 25
2. Validation ...................................................................................................................................................................................... 25 Testing (Omar’s Work) ............................................................................................................................................................................. 25

Memory Usage .......................................................................................................................................................................................... 26

Customer Service Memory Usage ................................................................................................................................................ 26 Inventory Service Memory Usage ................................................................................................................................................ 29

...................................................................................................................................................................................................... 29

Reviews Service Memory Usage ................................................................................................................................................. 30

Sales Service Memory Usage ....................................................................................................................................................... 31

A. Coverage .................................................................................................................................................................................... 32

Customer Coverage Report .......................................................................................................................................................... 32

Inventory Coverage Report .......................................................................................................................................................... 33

Sales Coverage Report ................................................................................................................................................................. 33

Reviews Coverage Report ............................................................................................................................................................ 33

Deployment and Integration (Ibrahim’s Work) ......................................................................................................................................... 33

Documentation and Profiling (Omar’s Work) ........................................................................................................................................... 34

Github and Version Control (Ibrahim’s Work) ......................................................................................................................................... 35

Docker (Ibrahim’s Work) .......................................................................................................................................................................... 35

Packaging Process: ....................................................................................................................................................................... 35

Validation and Sanitization (Ibrahim’s Work) .......................................................................................................................................... 36

User Authentication (Ibrahim’s Work)...................................................................................................................................................... 36 Moderation (Ibrahim’s Work) ................................................................................................................................................................... 37

Addition Professional Tasks ...................................................................................................................................................................... 37

Enhanced Inter-Service Communication (Omar’s Work) ............................................................................................................ 37

Advanced Security Measures (Ibrahim’s Work) .......................................................................................................................... 37

Introduction

## Project Overview

Our project consists of backend web services of an E-commerce website, where we aim to apply all our gained knowledge from EECE 435Lab taken this semester. The project is designed to cover crucial aspects of E-commerce operations and services ranging from customer management, inventory management, sales processes, all while managing product reviews and ratings.

### Project Objective

Our aim is to build a modular, scalable, and efficient backend infrastructure for the E-commerce website while applying the knowledge we gained this semester to ensure that the backend is well-documented, tested, and follows the most recent trends and tools in the software industry, like version control and Github.

The project also focuses on the security aspect due to its importance nowadays. For that we aimed to ensure that the website abides by important security measures like authentication and authorization while ensuring that user inputs are validated and sanitized.

## System Architecture

When building the system’s architecture, we made sure that it is modular and that it allows seamless integration for any services if needed in future and allows scalability. For that, our architecture consists of a folder for routes, a folder for models, a folder for services, and a folder for testing. Each folder consists of several files, one for each service. This methodology allows easier understanding of the code structure rather than depending on singular files. It also allows easier integration of any future changes, as you can directly locate where the change falls in this architecture.

### Service 1 Description “Customers” (Ibrahim’s Work)

This service manages customer-related operations, including registration, updating profiles, managing wallet balances, and handling authentication and authorization. Key features include:

* Secure user authentication and role-based access control (e.g., distinguishing between regular users and admins).
* Wallet balance operations, such as charging or deducting amounts.
* Validation of user inputs to ensure data integrity.
* Modular design allowing seamless user profile updates and the addition of admin privileges.

### Service 2 Description “Inventory” (Omar’s Work)

The Inventory Service is responsible for managing product information and stock levels. It ensures:

* Accurate tracking of product availability and attributes.
* The ability to update product information dynamically.
* Validation of product details to avoid invalid or incomplete records. This service lays the groundwork for integrating product browsing and sales features.

### Service 3 Description “Sales” (Omar’s Work)

This service handles the core e-commerce functionality related to transactions and order processing. It provides:

* Mechanisms for placing orders, updating order statuses, and managing sales records.
* Integration with the Customer Service for linking transactions to user accounts.
* Ensures security and validation to avoid processing invalid orders.

### Service 4 Description “Reviews” (Ibrahim’s Work)

The Reviews Service allows users to submit and manage product reviews while maintaining high security standards. Key features include:

* Review moderation, allowing admins to approve or reject reviews.
* Validation and sanitization of user inputs to prevent SQL injection and cross-site scripting (XSS) attacks.
* Retrieval of reviews by product or customer, enabling insights into product feedback.
* A focus on ensuring that only authentic, relevant, and secure reviews are displayed on the platform.

Implementation and Details

### **Service 1 – Customers (Ibrahim’s Work)**

***Functionality:***

* Manages user accounts, including registration, profile updates, and wallet operations.
* Implements secure authentication and authorization using JWT tokens.
* Allows admins to oversee customer details and operations.

***Challenges Faced:***

* Designing secure authentication mechanisms to handle sensitive user data.
* Adding role-based access control (is\_admin) without breaking existing functionalities.
* Ensuring concurrent updates to customer wallets are managed correctly.

***APIs:***

1. **POST /api/customers** Create a new customer account.
2. **GET /api/customers**

Retrieve a list of all customers (admin-only).

1. **GET /api/customers/{username}**

Fetch details of a specific customer.

1. **PUT /api/customers/{username}**

Update details of an existing customer.

1. **DELETE /api/customers/{username}**

Remove a customer account.

1. **POST /api/customers/{username}/charge**

Add funds to a customer’s wallet.

1. **POST /api/customers/{username}/deduct**

Deduct funds from a customer’s wallet.

### **Service 2 – Inventory (Omar’s Work)**

***Functionality:***

* Handles product data, including addition, updates, and retrieval.
* Tracks inventory levels and ensures consistency in stock updates.
* Provides endpoints for both admin and customer access to product data.

***Challenges Faced:***

* Validating input to ensure correct product details (e.g., prices, categories).
* Preventing conflicts during stock updates in high-concurrency scenarios.
* Managing relationships between inventory and dependent services like sales.

***APIs:***

1. **POST /api/inventory/products**

Add a new product to the inventory (admin-only).

1. **GET /api/inventory/products**

Retrieve a list of all products.

1. **GET /api/inventory/products/{product\_id}**

Get details of a specific product.

1. **PUT /api/inventory/products/{product\_id}**

Update product information (admin-only).

1. **DELETE /api/inventory/products/{product\_id}**

Remove a product from inventory (admin-only).

### **Service 3 – Sales (Omar’s Work)**

***Functionality:***

* Manages sales operations, linking customers to product purchases.
* Tracks historical sales data for reporting and analytics.
* Supports endpoints for placing orders and retrieving sales information.

***Challenges Faced:***

* Linking transactions to customers and products without data inconsistencies.
* Ensuring sales operations do not disrupt inventory or customer data.
* Handling edge cases like invalid sales (e.g., insufficient stock or customer funds).

***APIs:***

1. **POST /api/sales**

Create a new sale by specifying customer and product details.

1. **GET /api/sales/customer/{customer\_id}**

Retrieve all sales linked to a specific customer.

1. **GET /api/sales/item/{product\_id}**

Get all sales for a specific product.

1. **PUT /api/sales/{sale\_id}**

Update an existing sale’s details.

1. **DELETE /api/sales/{sale\_id}**

Cancel or delete a specific sale.

### **Service 4 – Reviews (Ibrahim’s Work)**

***Functionality:***

* Allows customers to submit reviews for products, including ratings and comments.
* Supports admins in moderating reviews to maintain quality and relevance.
* Provides endpoints for retrieving reviews by product or customer.

***Challenges Faced:***

* Validating and sanitizing review inputs to prevent security vulnerabilities (e.g., XSS).
* Balancing transparency in customer feedback with moderation control.
* Managing scalability as review volume increases.

***APIs:***

1. **POST /api/reviews**

Submit a new review for a product.

1. **PUT /api/reviews/{review\_id}**

Update an existing review.

1. **DELETE /api/reviews/{review\_id}**

Remove a review.

1. **GET /api/reviews/product/{product\_id}**

Fetch all reviews for a specific product.

1. **GET /api/reviews/customer/{customer\_id}**

Retrieve all reviews submitted by a specific customer.

1. **GET /api/reviews/pending**

Get a list of reviews pending moderation (admin-only).

1. **PUT /api/reviews/{review\_id}/moderate**

Moderate a review by approving or rejecting it (admin-only).

### **Service: Authentication (Ibrahim’s Work)**

***Functionality***

The Authentication Service manages user registration, login, and token-based access control. It ensures secure communication between the client and the backend.

***Challenges Faced***

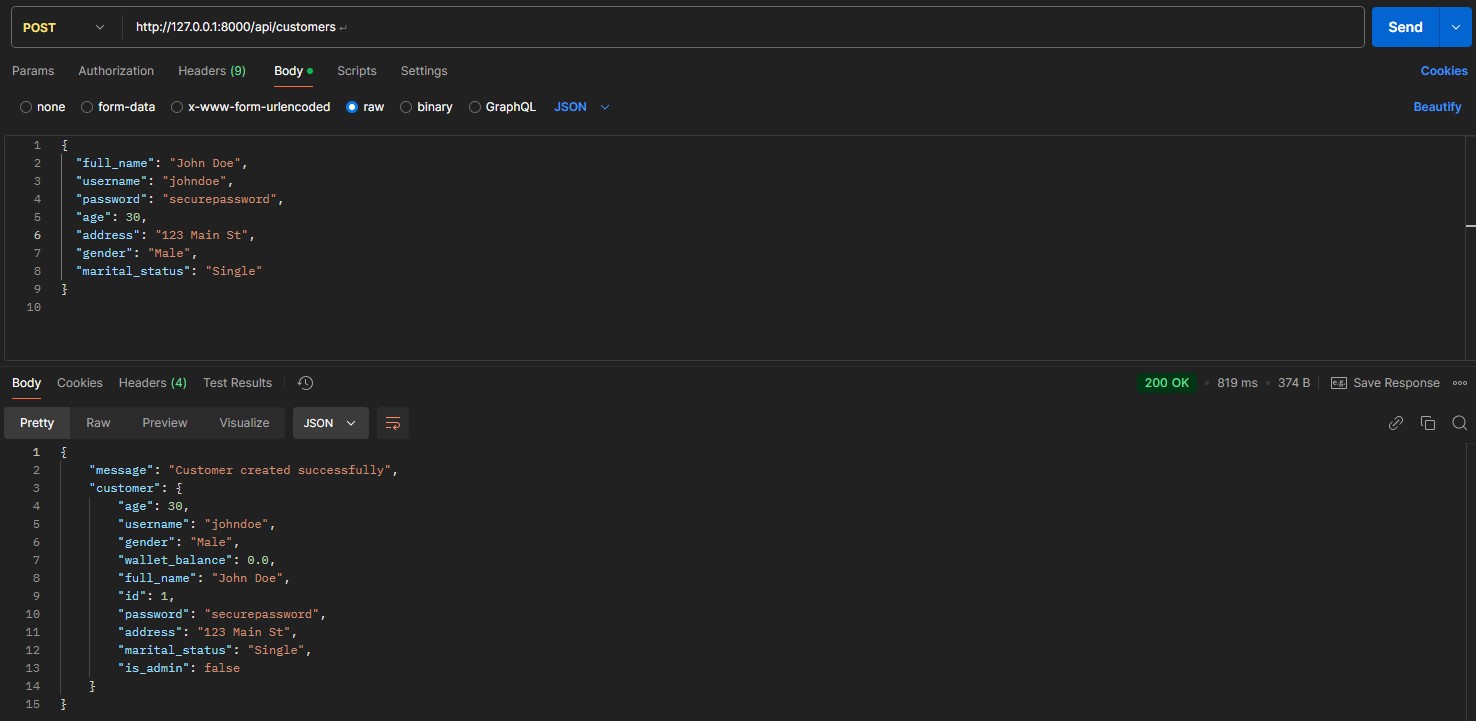
* **Token Validation**: Ensuring tokens remain secure while verifying expiration and claims.
* **Role Management**: Implementing role-based access (e.g., admin and user roles).
* **Integration**: Seamlessly integrating authentication checks across other services.

***API Documentation***

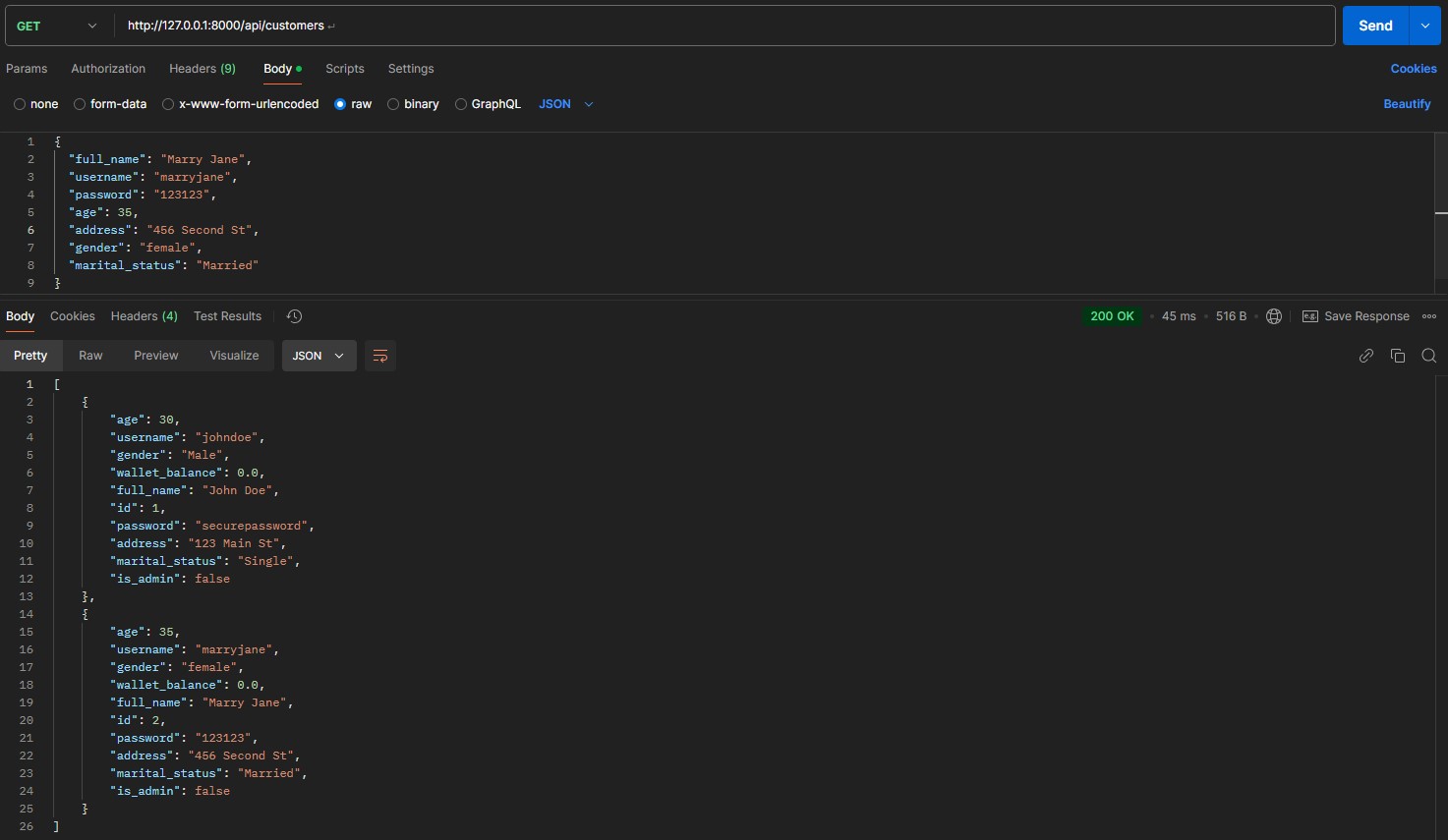
1. **Register a User** 
   1. **Endpoint**: POST /auth/register
   2. **Description**: Registers a new user in the system.
2. **Login** 
   1. **Endpoint**: POST /auth/login
   2. **Description**: Logs in a user and provides a JWT token.
3. **Access Protected Endpoints** 
   1. **Description**: Tokens are included in requests to access protected endpoints across services.
   2. **Authentication Method**: Bearer token in Authorization header.

## API Documentation (Screenshots)

Customer Service: 1. Create Customer



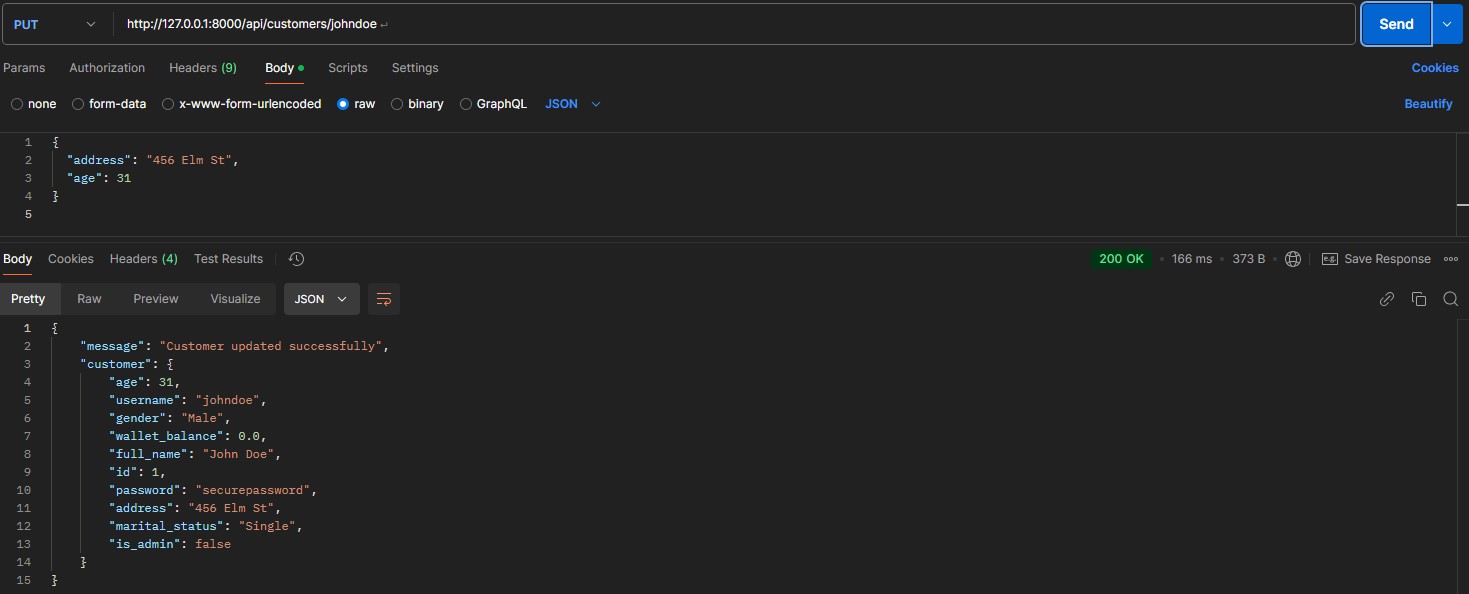
1. Get All Customers



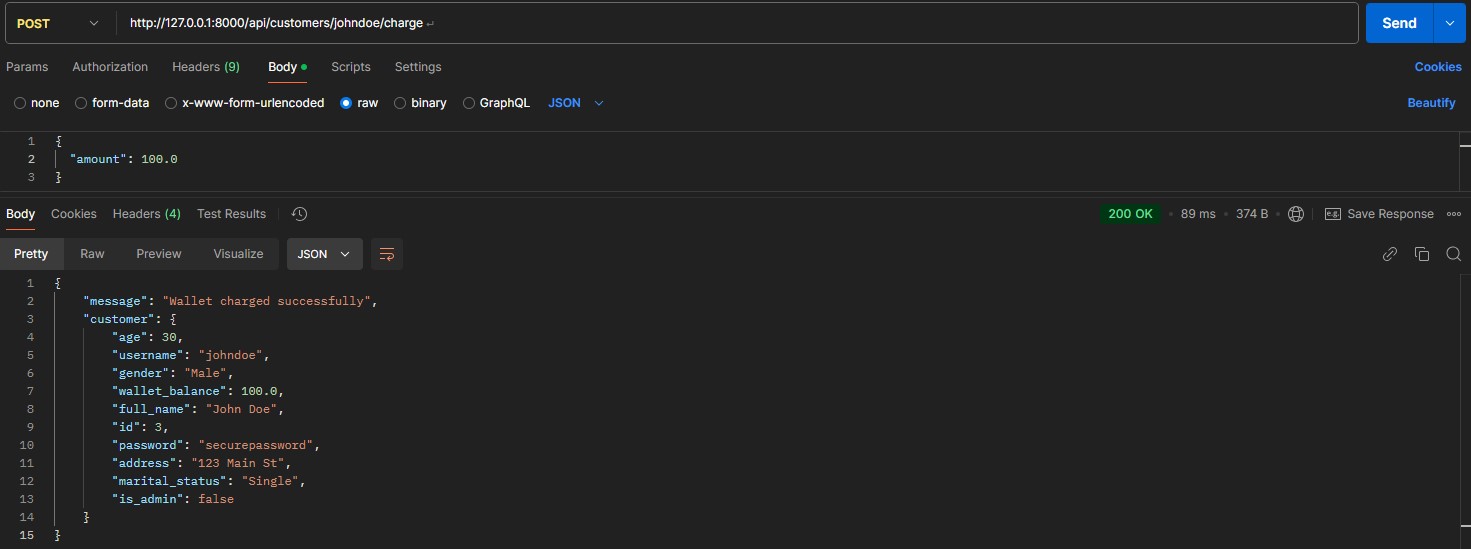
1. Get Customer by ID



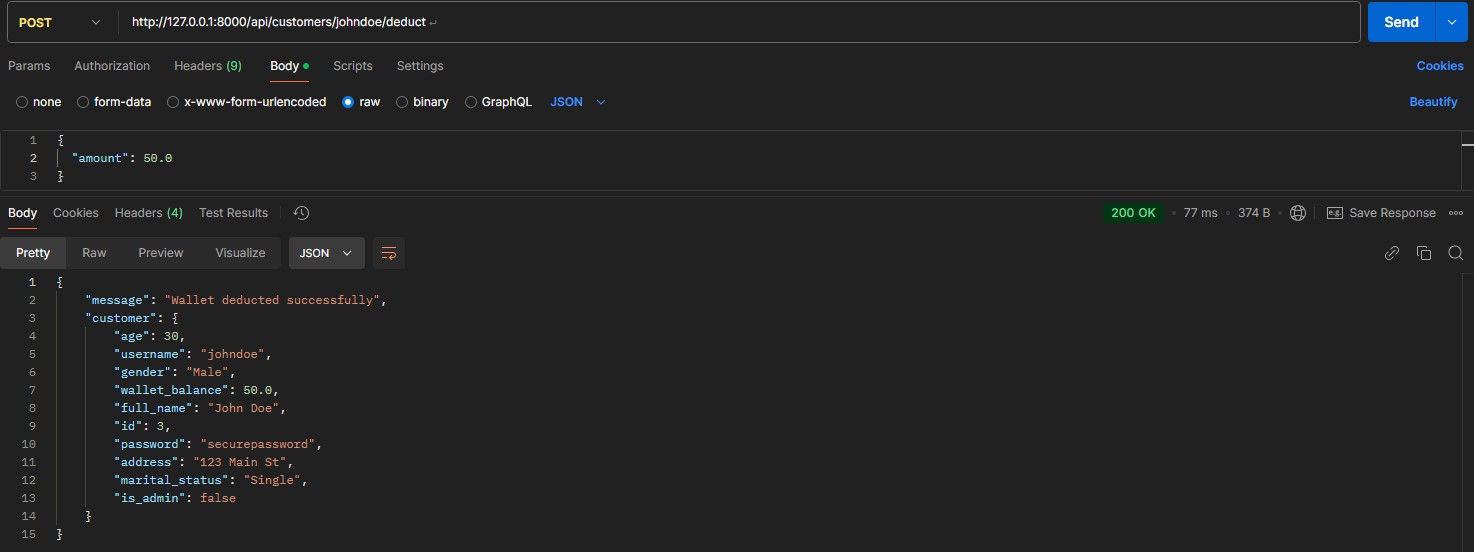
1. Update Customer Information



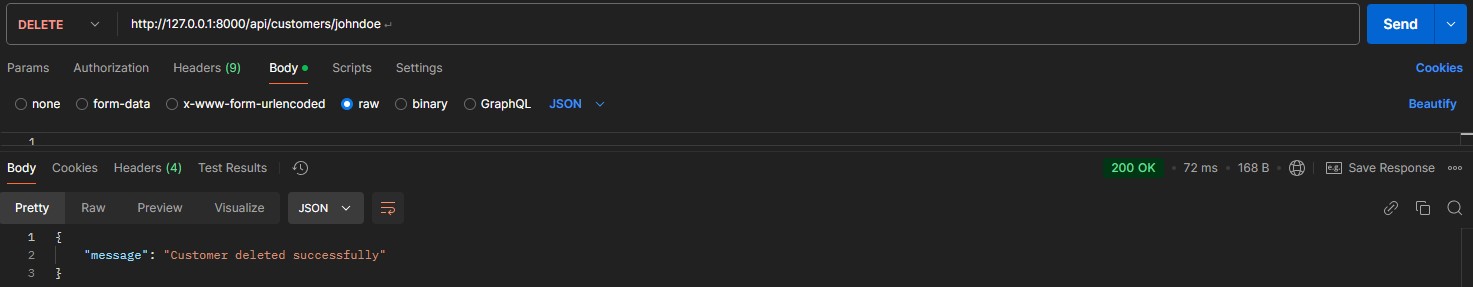
1. Charge Customer Wallet



1. Deduct Customer Wallet

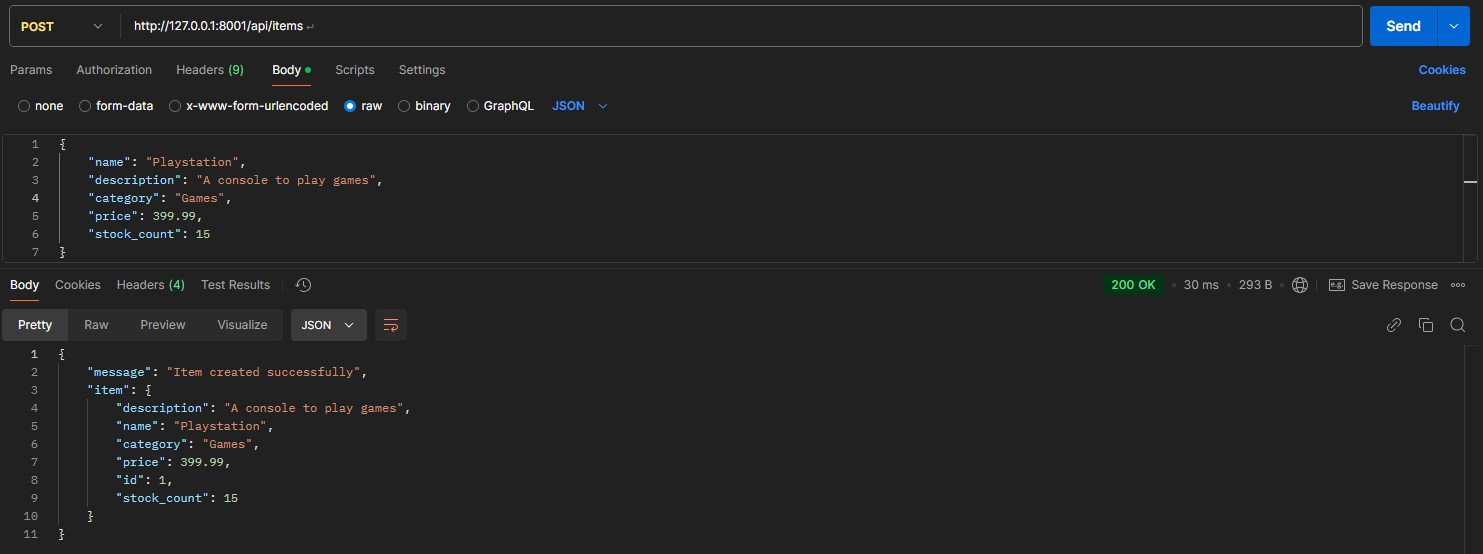


1. Delete Customer

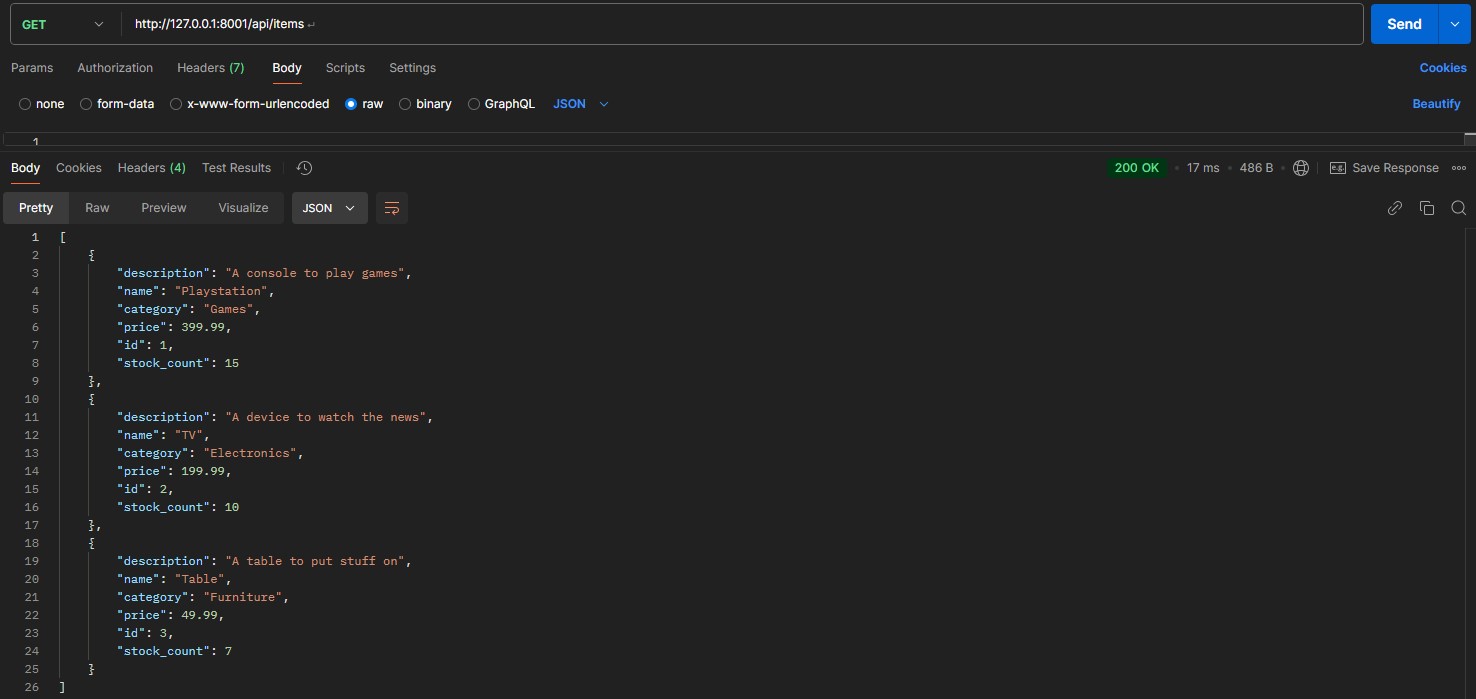


### Inventory Service

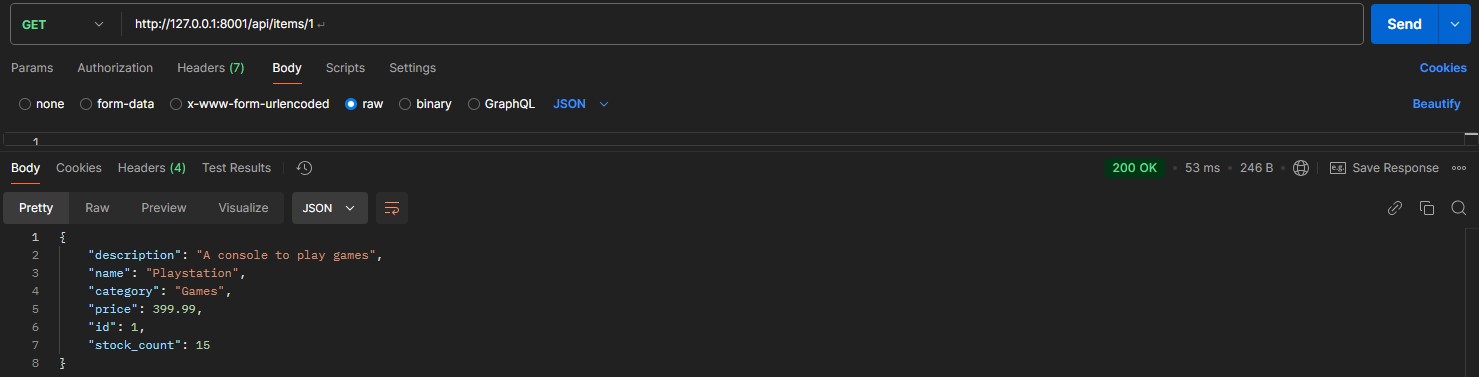
1. Create Item



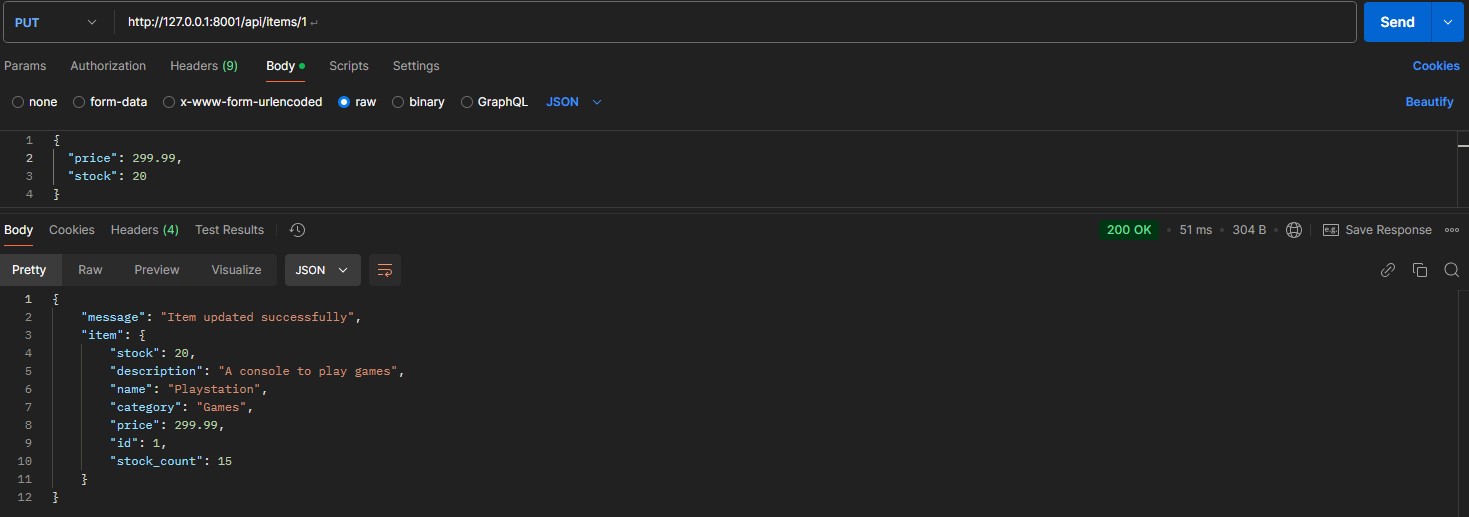
1. Get All Items



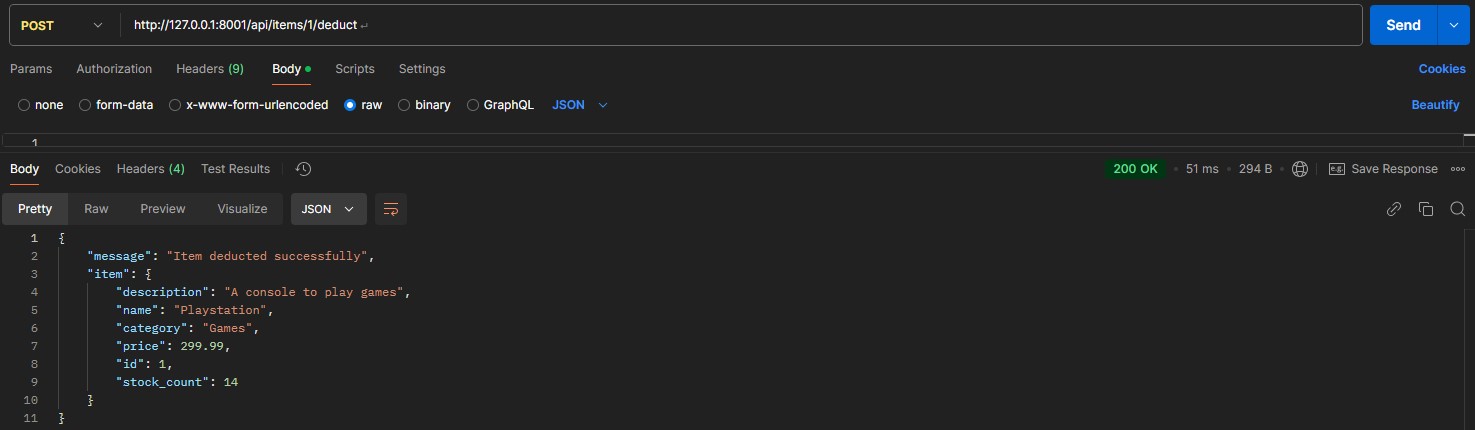
1. Get Item by ID



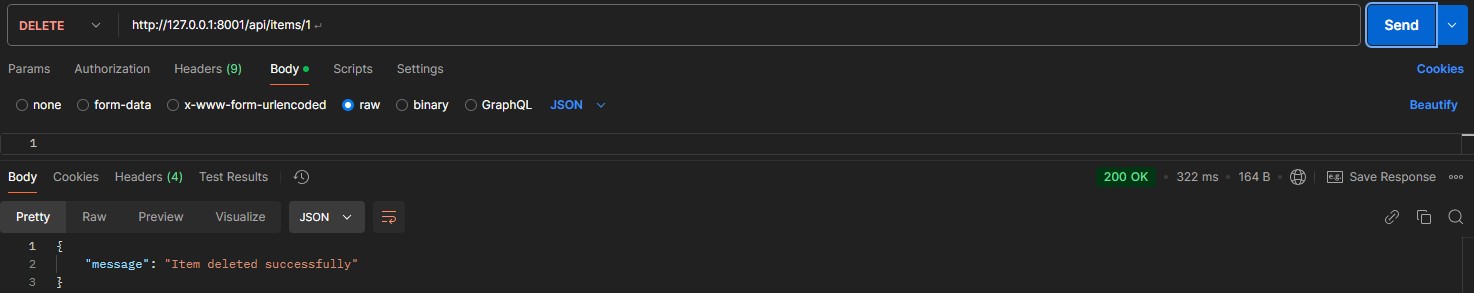
1. Update Item



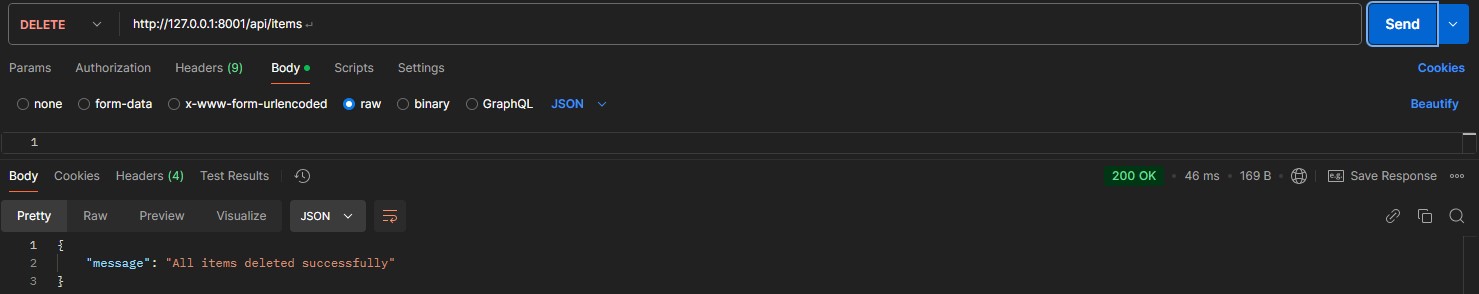
1. Deduct Item Stock



1. Delete Item

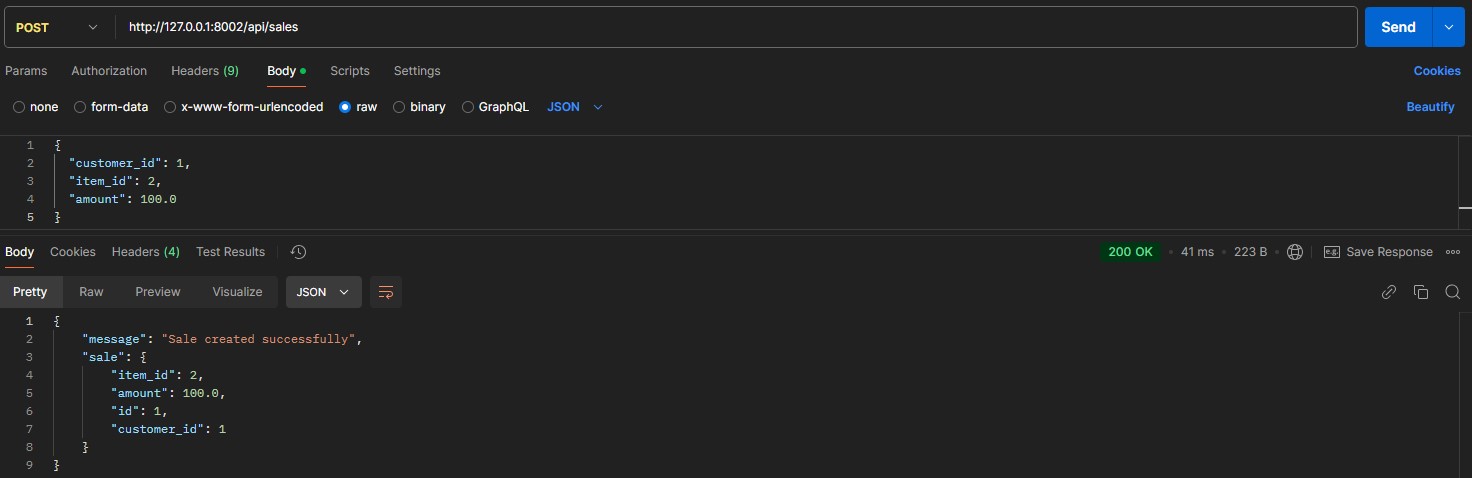


1. Delete All Items

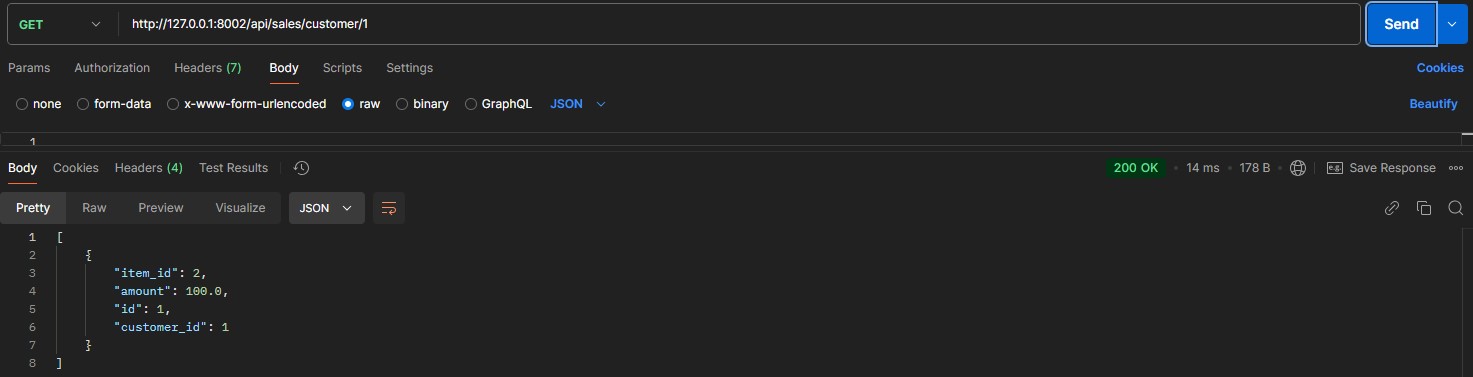


### Sales Service

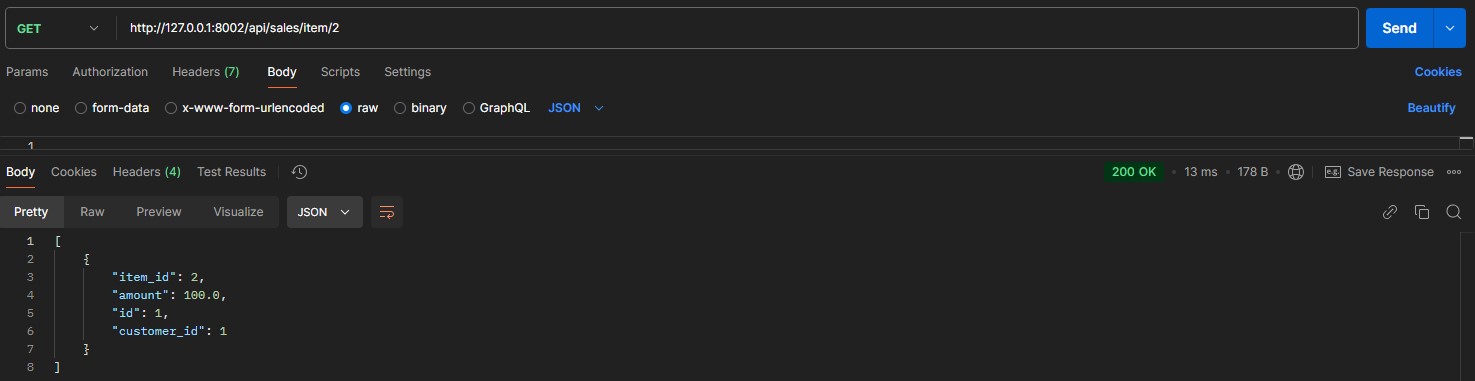
1. Create Sale



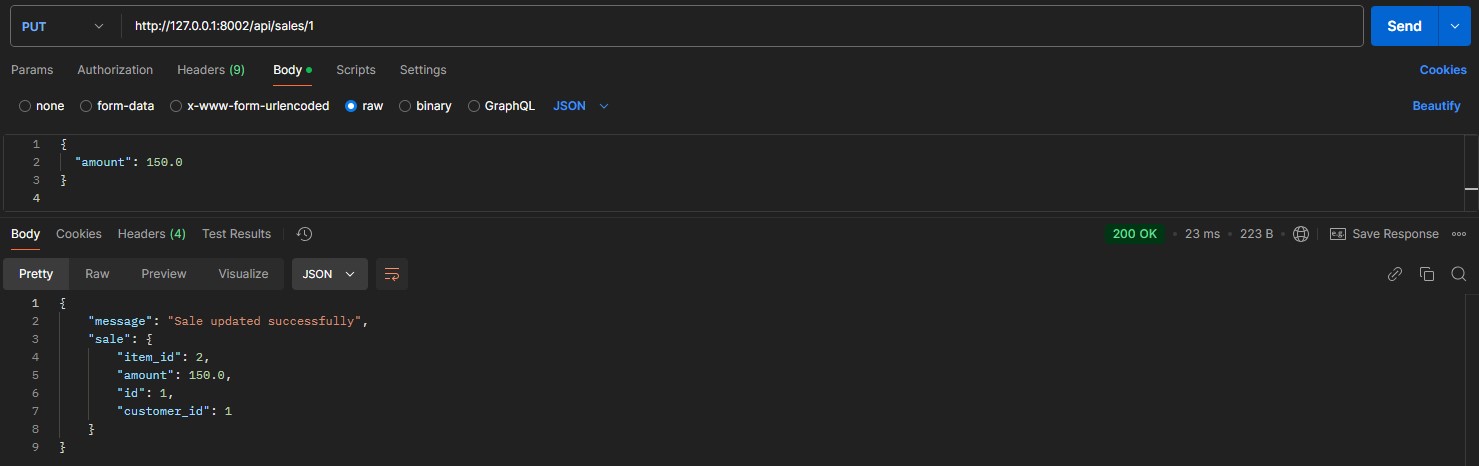
1. Get Sales by Customer



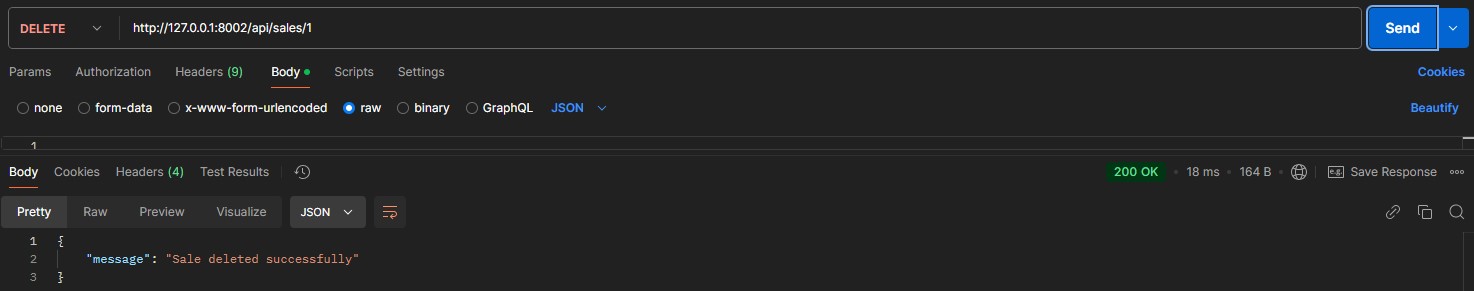
1. Get Sales by Item



1. Update Sale

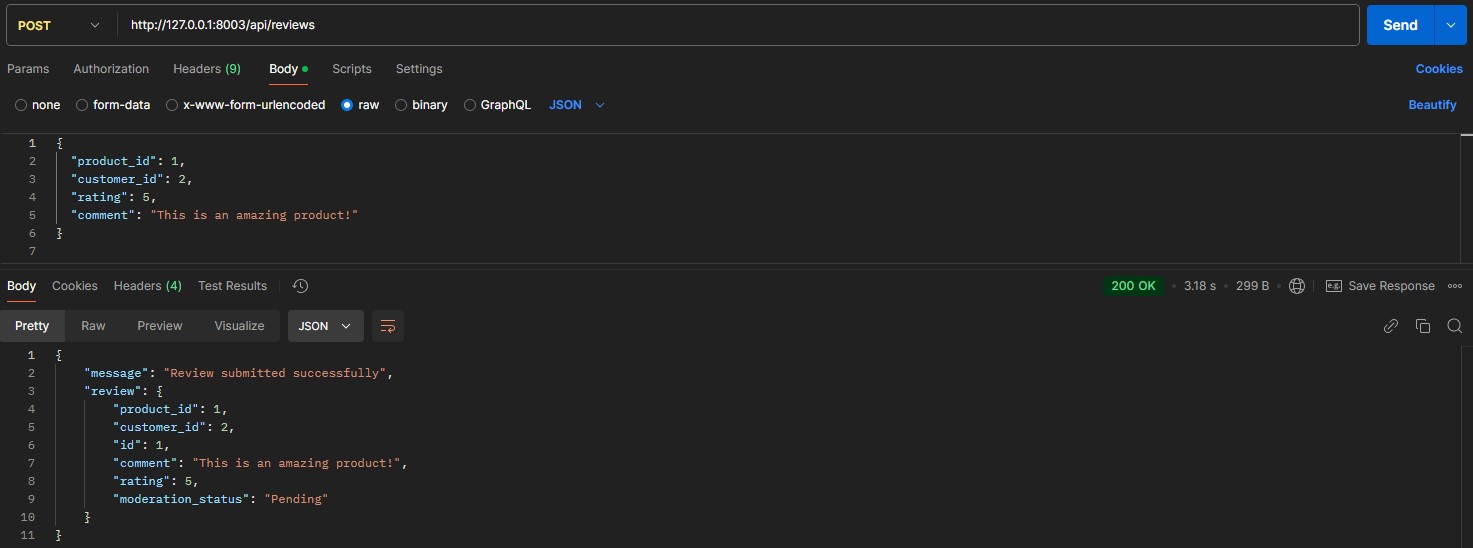


1. Delete Sale

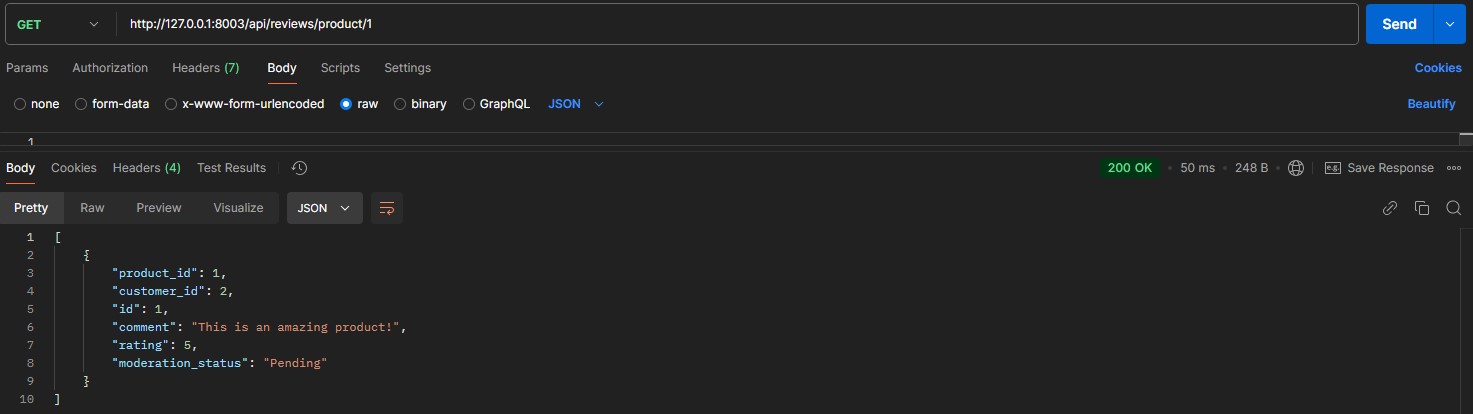


### Review Service

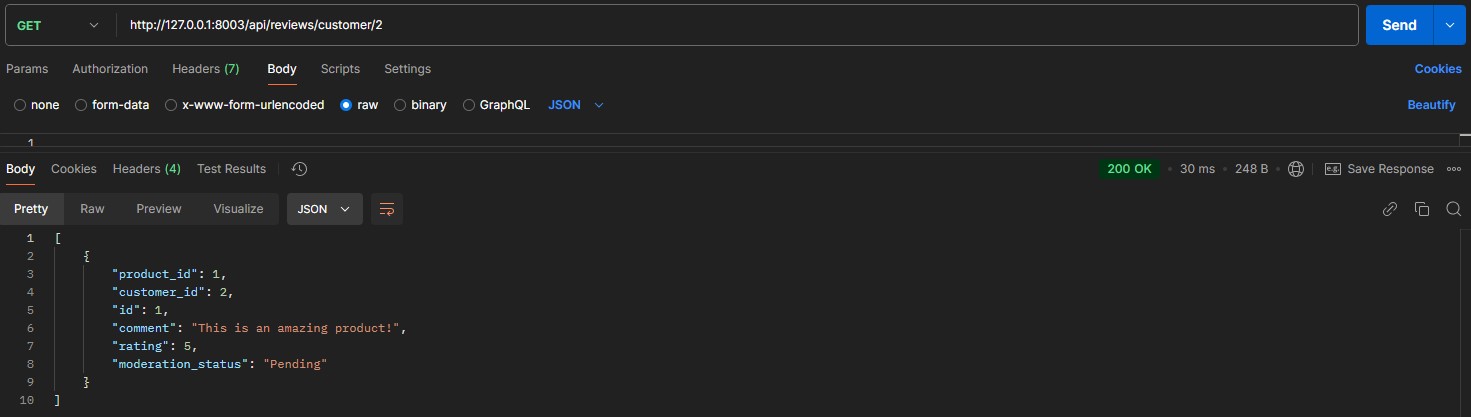
1. Create Review



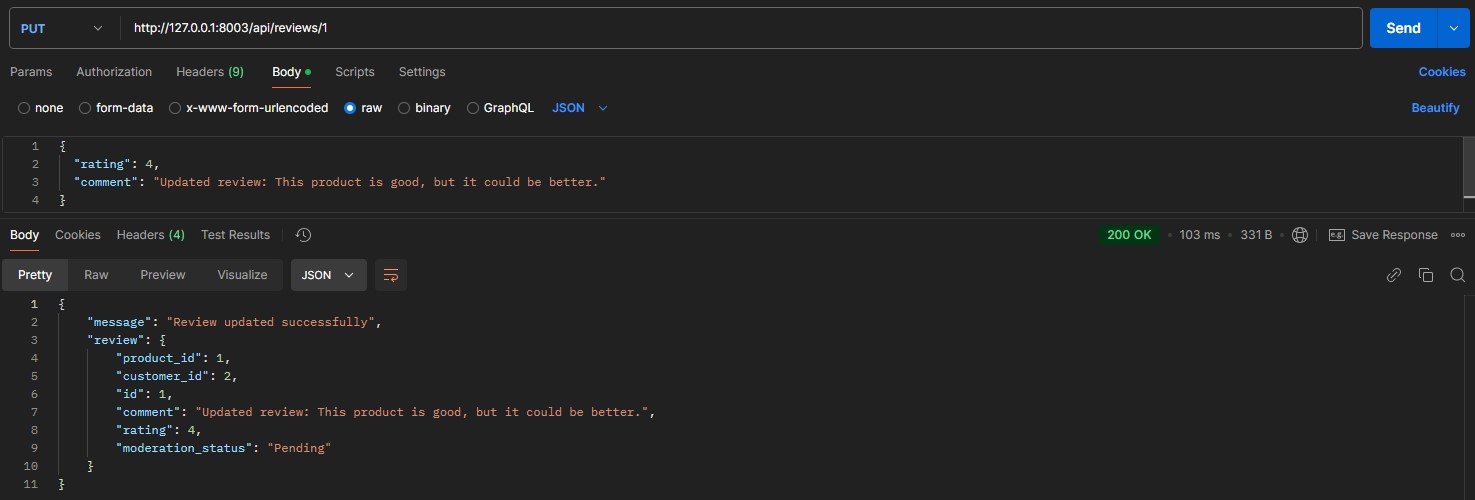
1. Get Reviews by Product



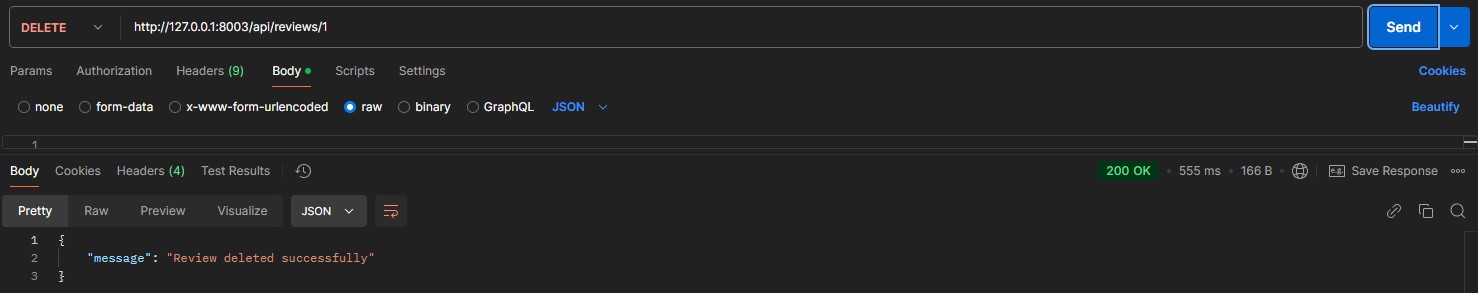
1. Get Reviews by Customer



1. Update Review

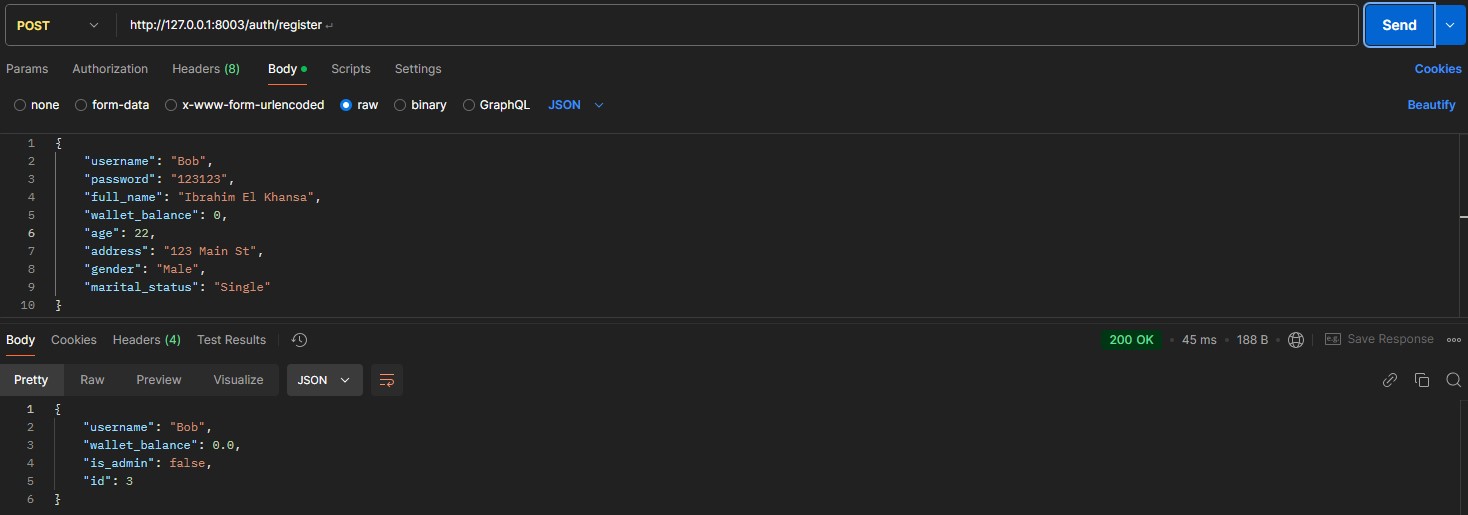


1. Delete Review

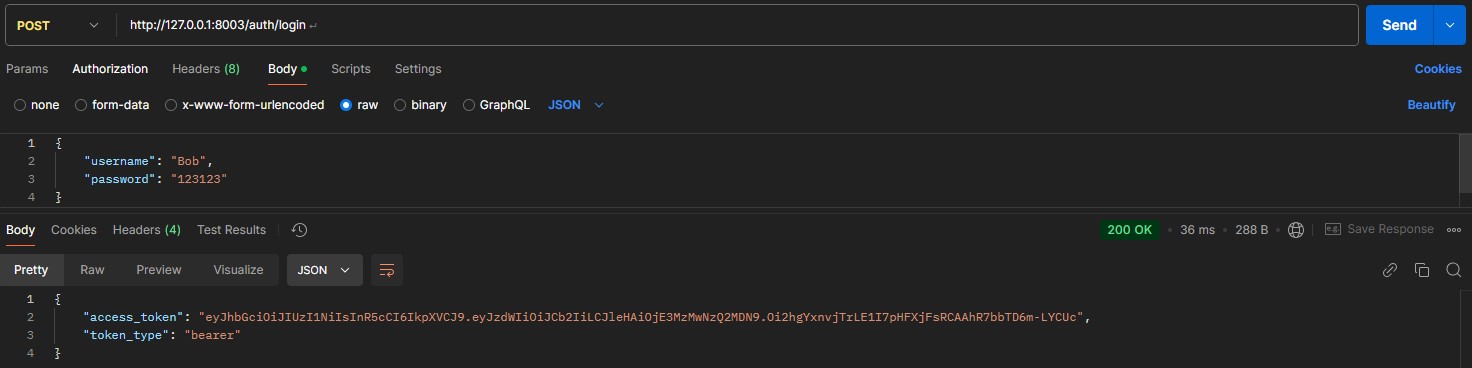


### Auth Service (Available for All Ports)

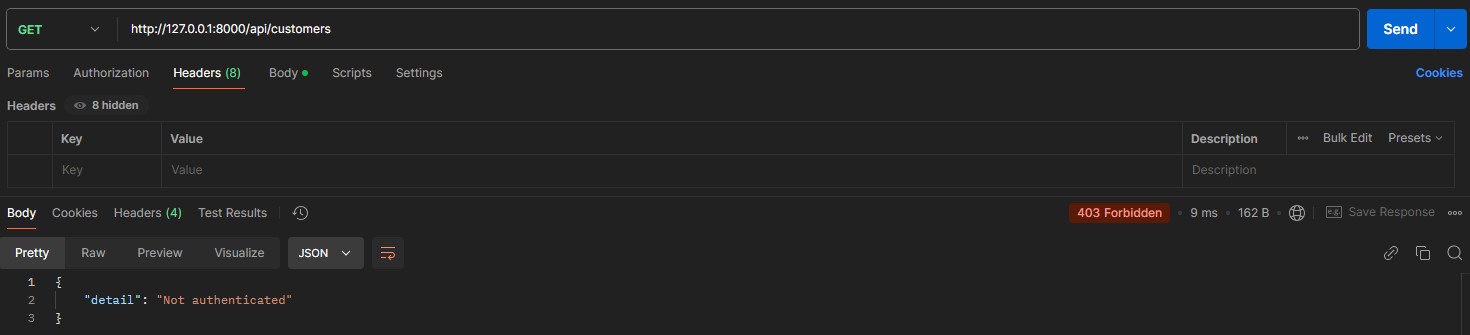
1. Register New User



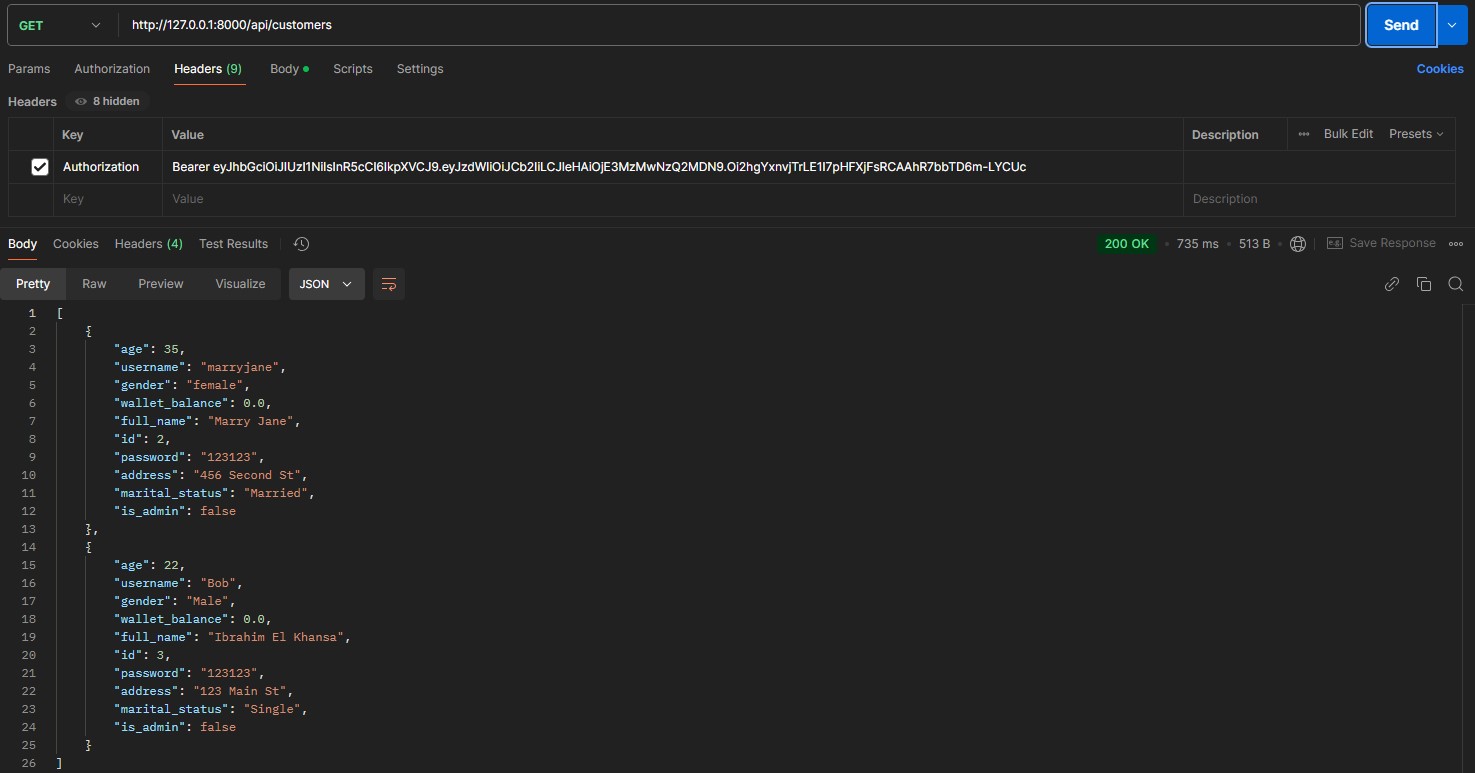
1. Login



1. Sample Request (Not Authenticated)

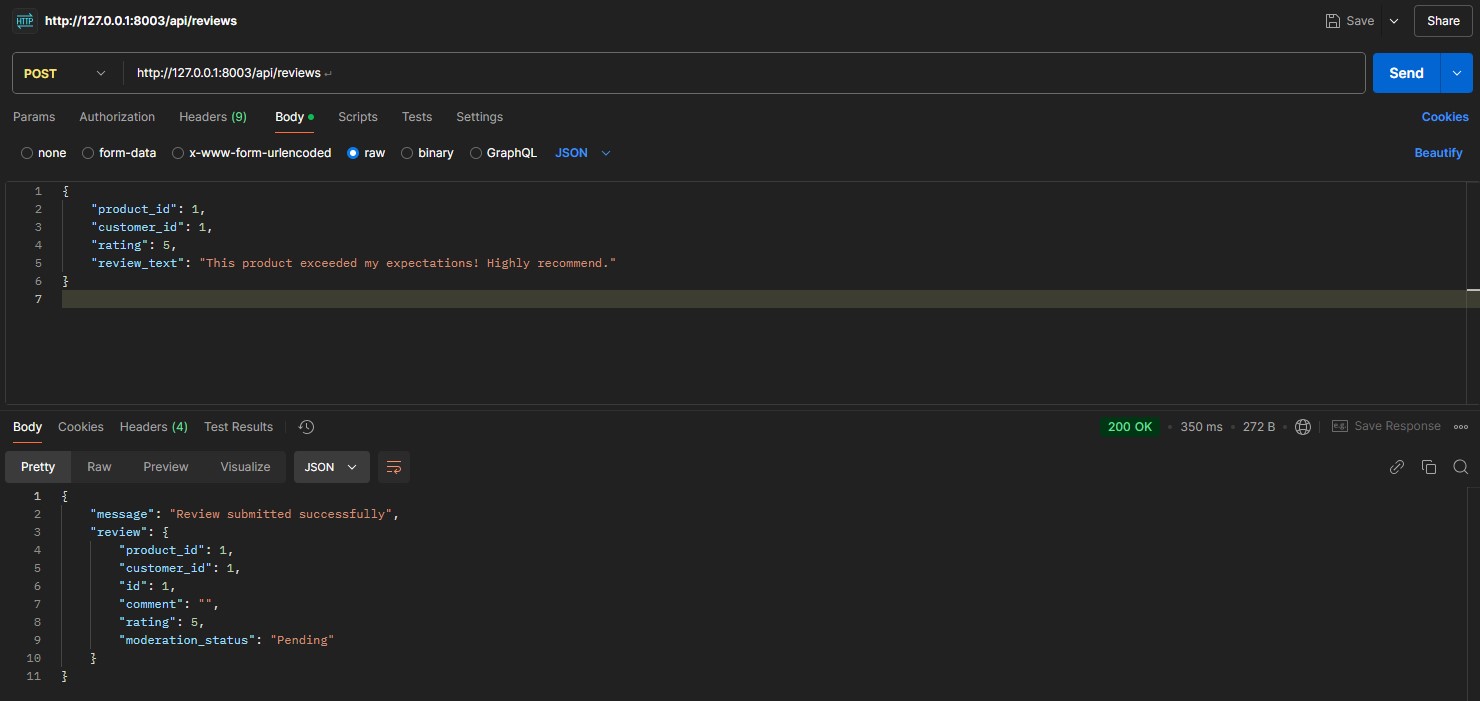


1. Sample Request (Authenticated)

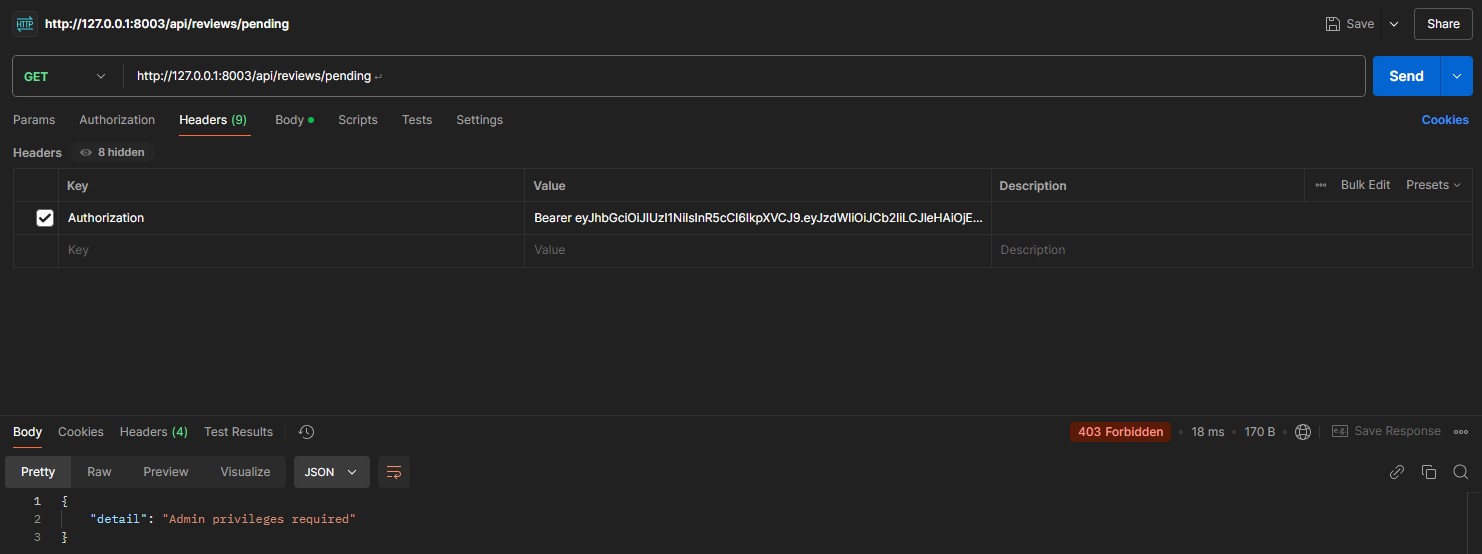


### Review Moderation Service

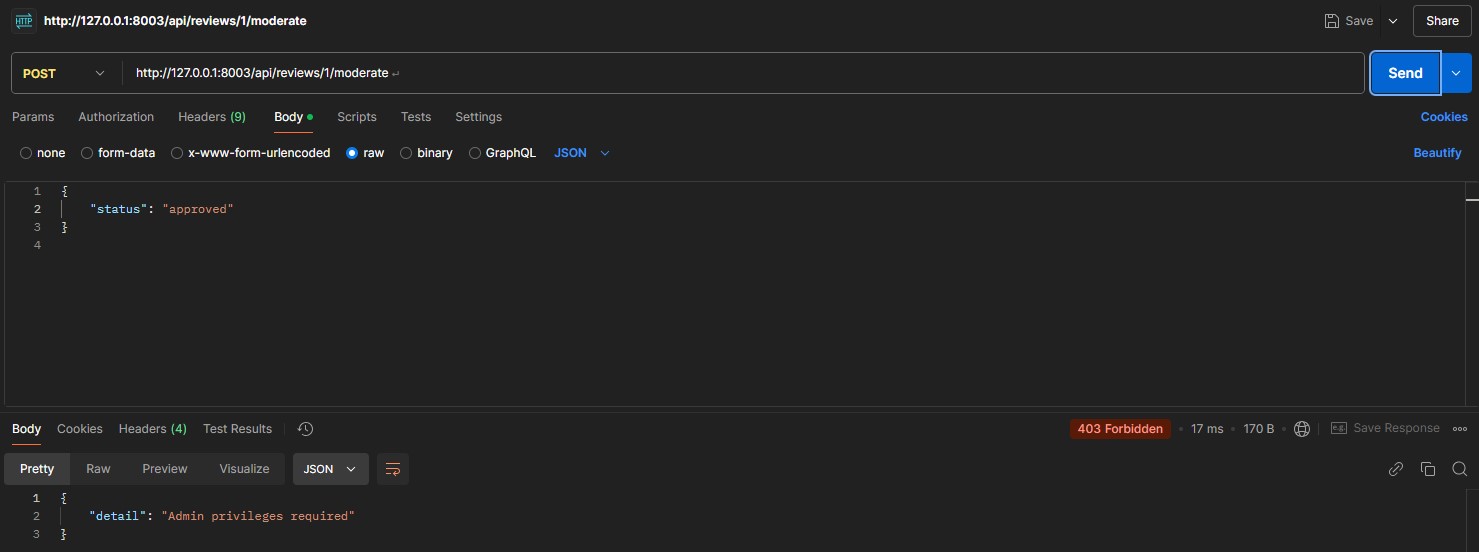
1. Create Review



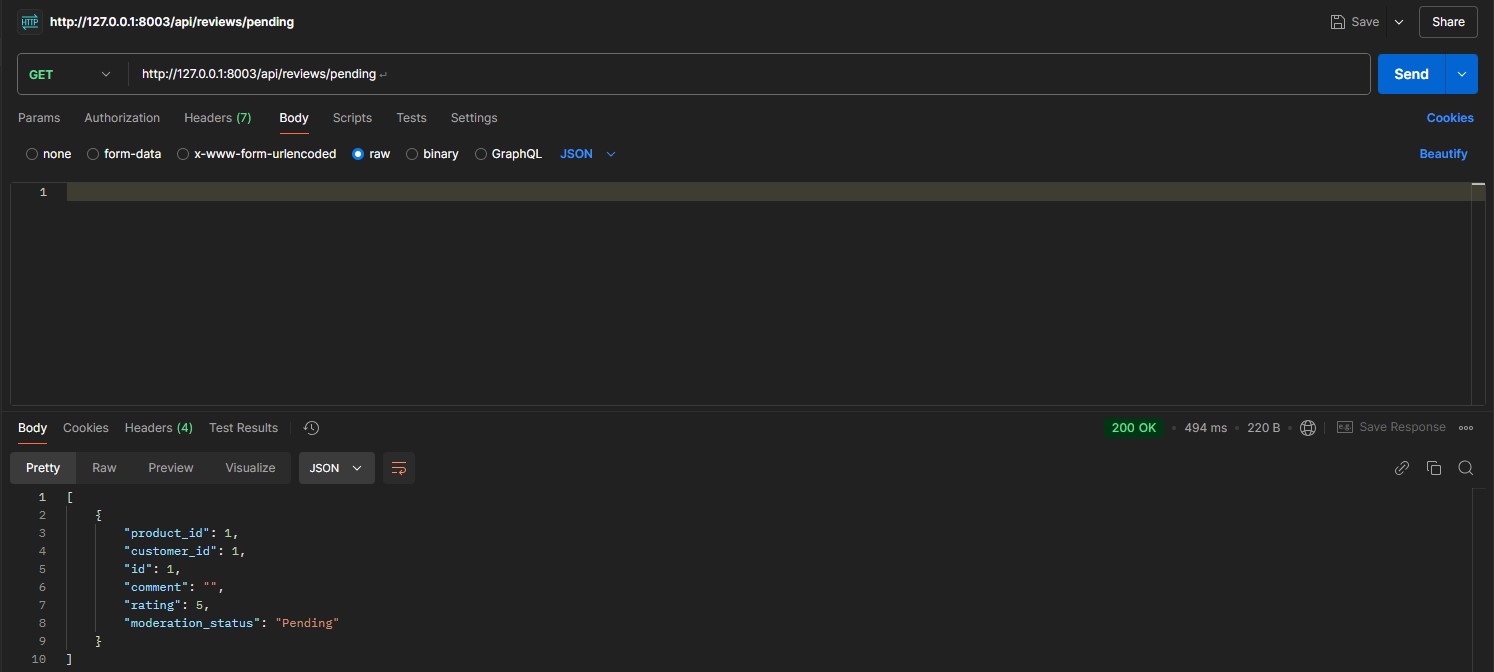
1. View All Pending Reviews (Non-Admin)



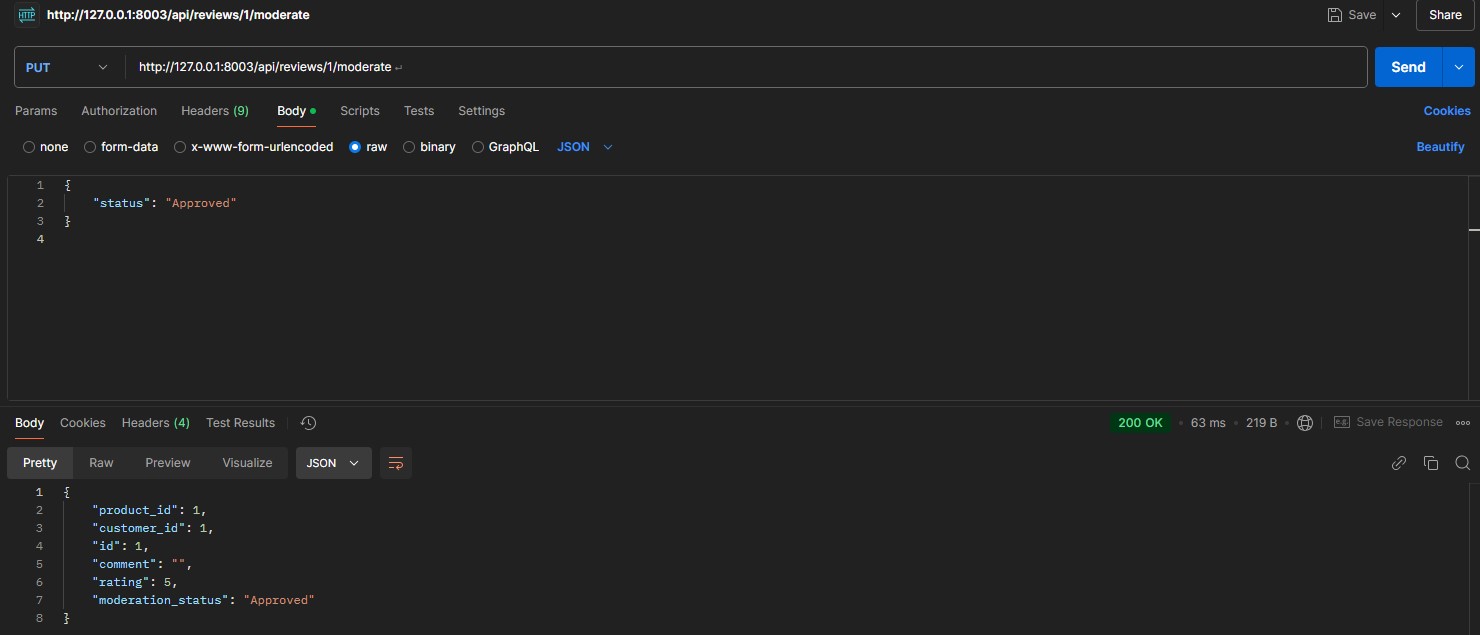
1. Moderate A Review (Non-Admin)



1. View All Pending Reviews (Admin)

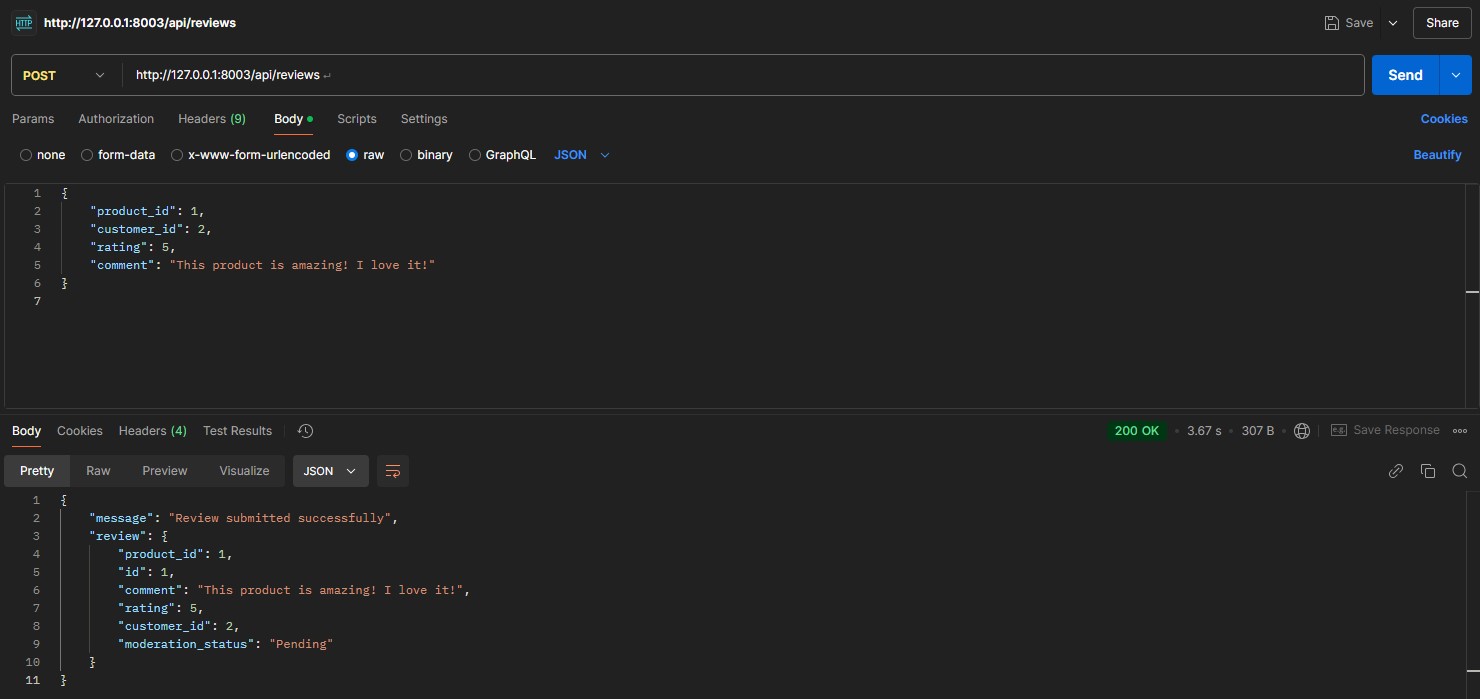


1. Moderate A Review (Admin)

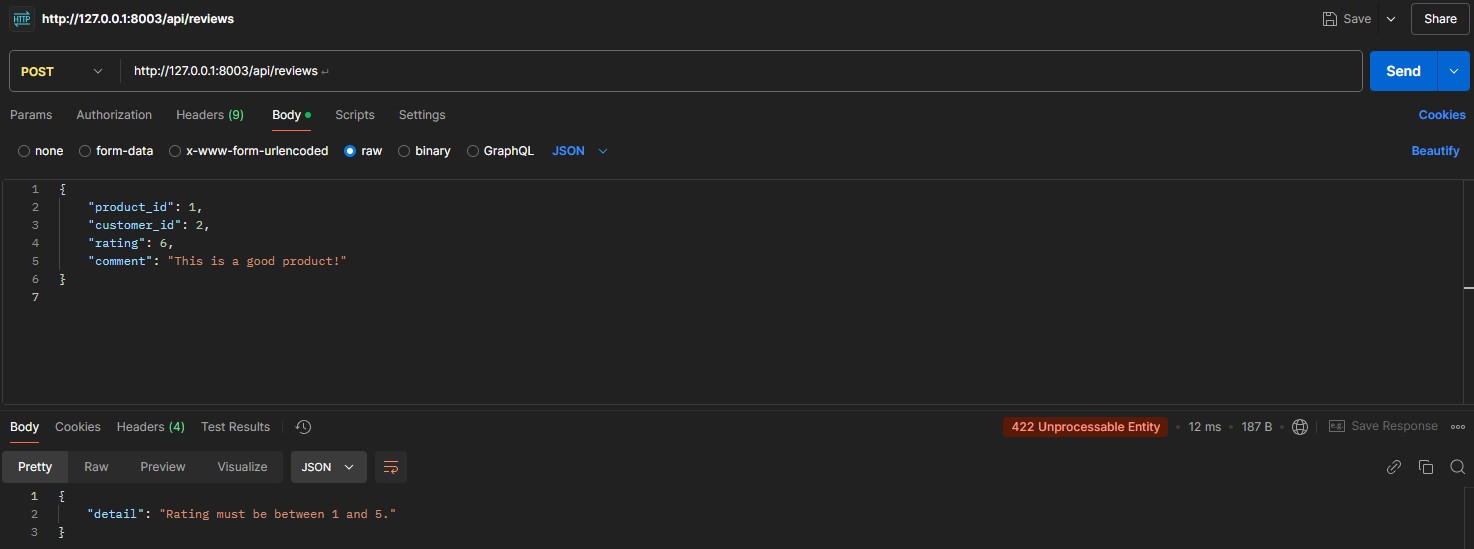


### Review Validation & Sanitization Service

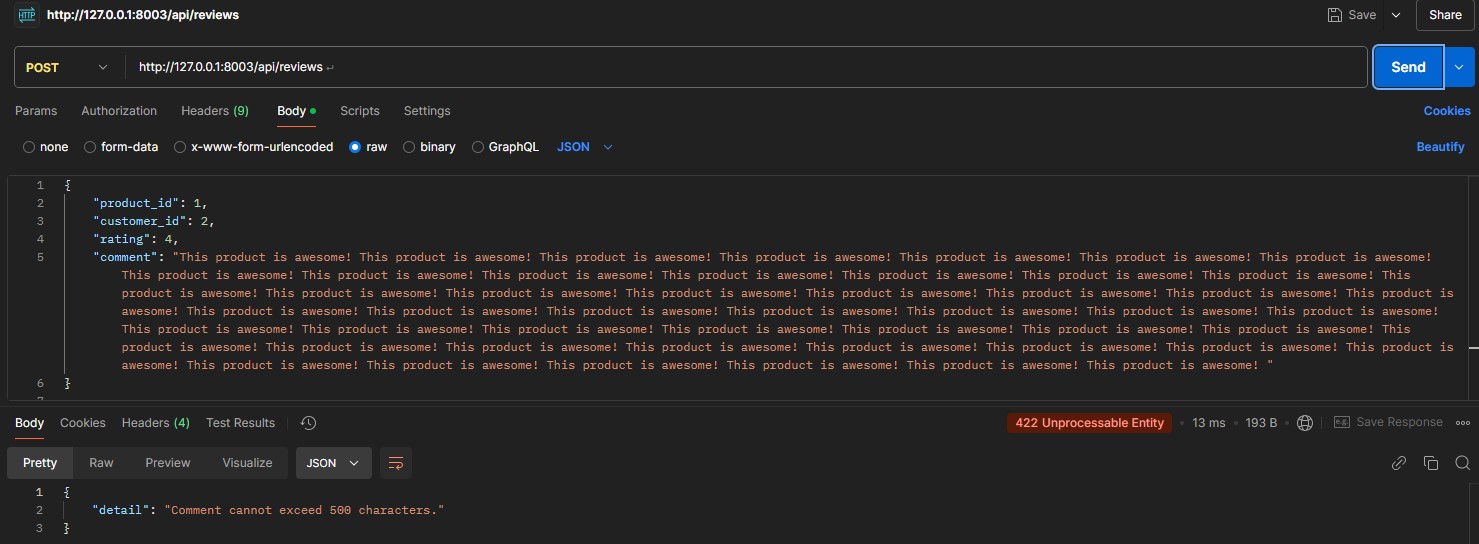
1. Valid Review



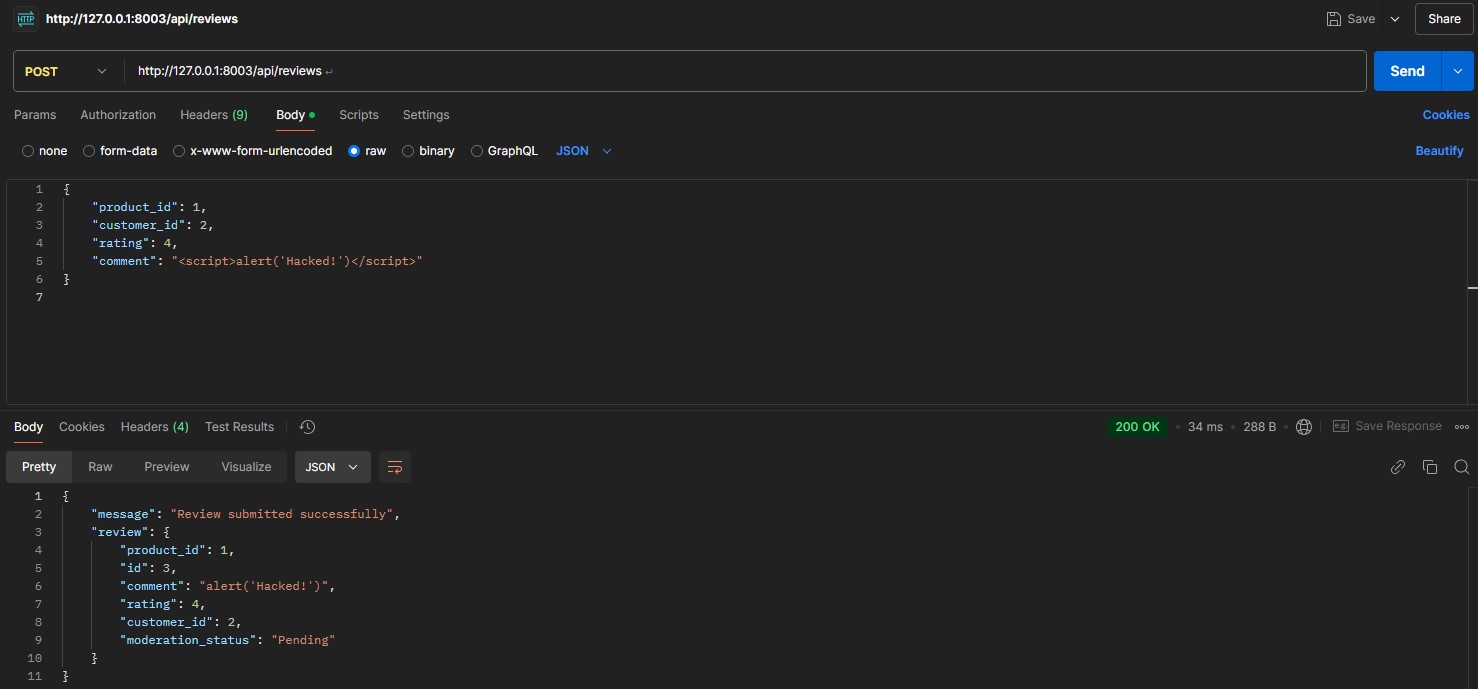
1. Invalid Review Rating



1. Invalid Review Comment



1. Review with Potentially Malicious Input



What happens is that the request with the malicious review is cleaned and then added to the DB rather than returned with a 422-status code

## Database Design



## Error Handling and Validation

1. Error Management
   1. **Global Exception Handling**:
      1. The system uses structured error handling through FastAPI's HTTPException to raise meaningful errors with proper HTTP status codes. ii. Critical failures such as database errors or missing data are caught, and appropriate error messages are logged for debugging purposes.
   2. **Circuit Breakers**:
      1. To handle external API failures, a pybreaker.CircuitBreaker is integrated, ensuring resilience by halting requests after repeated failures and resetting after a cooldown period. c. **Logging**:

i. Detailed error logs are maintained in respective log files (e.g., log\_customer.log) for each service to track issues systematically.

d. **Rollback on Failure**:

i. Database transactions are rolled back using SQLAlchemy to maintain data integrity when exceptions occur.

e. **User-Friendly Responses**:

i. Error messages are standardized to provide concise and helpful information to the end-user without exposing internal details.

1. Validation
   1. **Data Validation on Input**:
      1. All incoming data is validated using Pydantic models for structure, types, and constraints (e.g., mandatory fields, data ranges).
   2. **Field-Level Validation**:
      1. For example, customer data requires mandatory fields like username, age, and address. Specific constraints, such as age being a positive integer, are enforced at the model level.
      2. Reviews ensure rating is between 1 and 5, and comments do not exceed 500 characters. c. **Sanitization**:

i. Inputs such as review comments are sanitized using the bleach library to prevent XSS attacks.

d. **Custom Business Rules**:

i. Services implement custom validation for logical consistency. For example:

* + - 1. Deducting a wallet balance ensures sufficient funds exist.
      2. Creating a sale ensures the referenced customer and item exist. e. **Error Feedback**:

i. Validation errors return detailed responses with error locations and descriptions, helping users correct their requests.

Testing (Omar’s Work) Pytest Customer Service



Pytest Inventory Service



Pytest Sales Service



Pytest Review Service

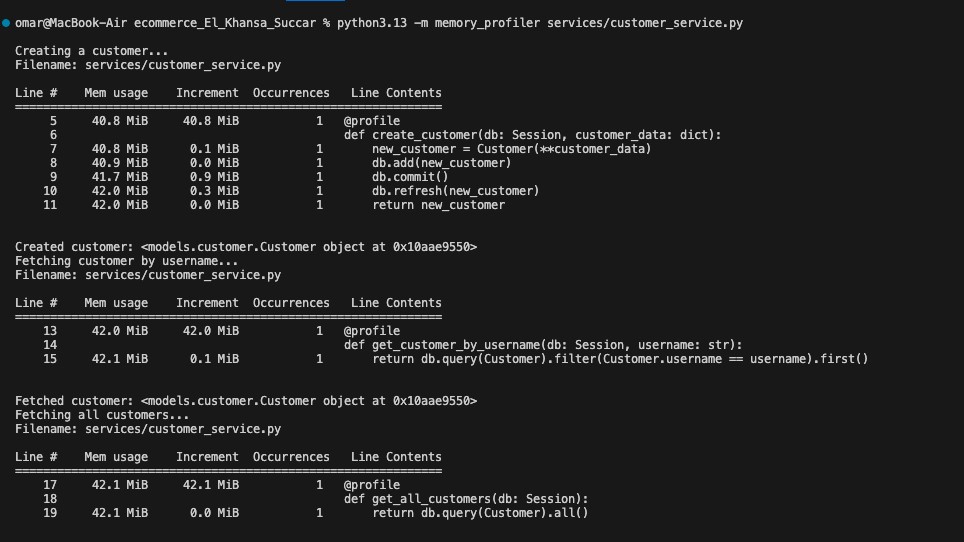


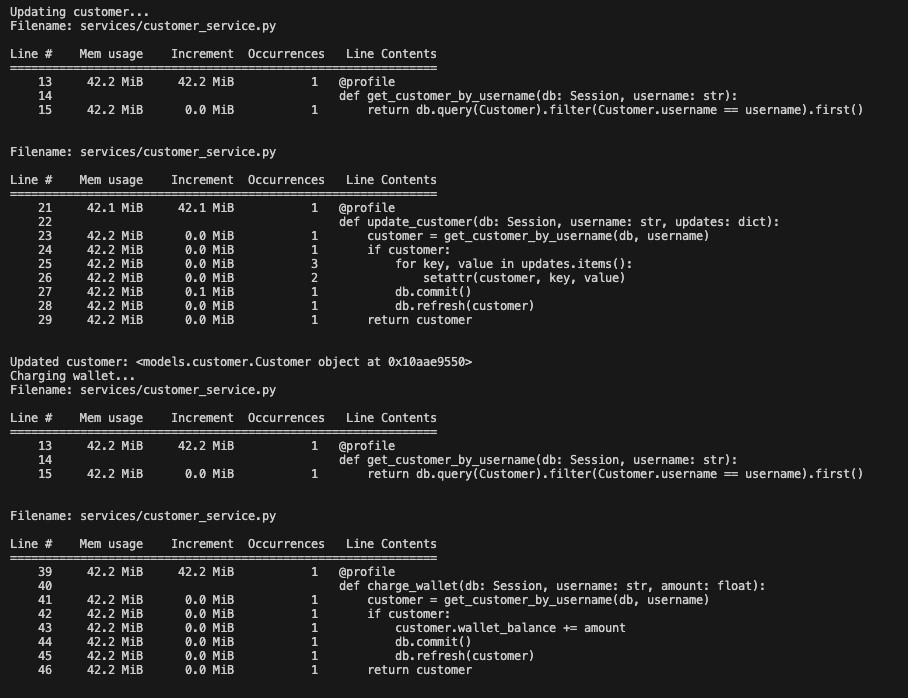
## Memory Usage

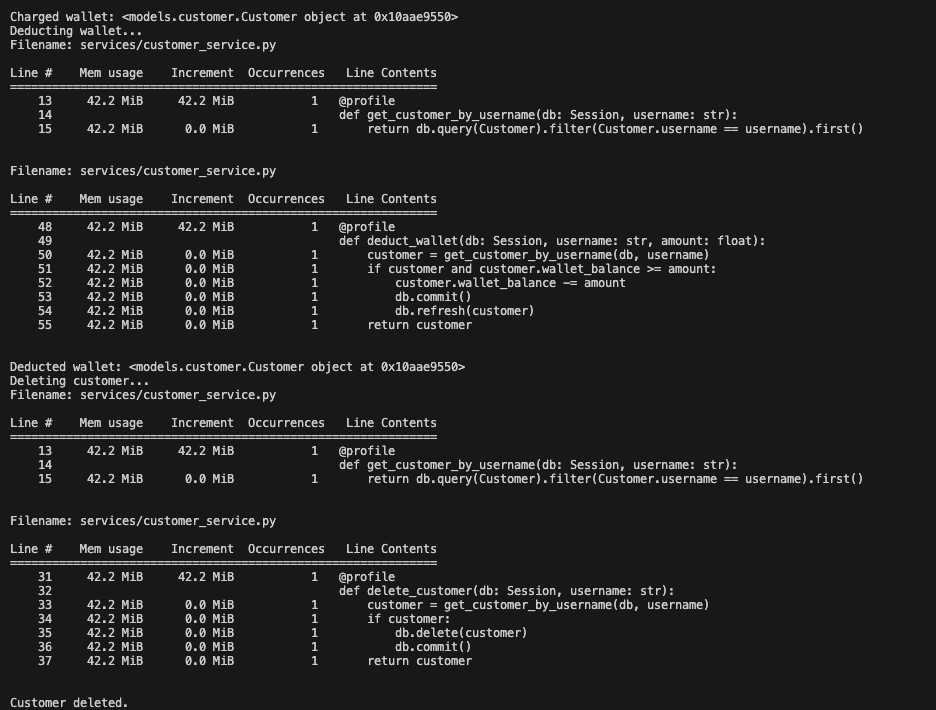
While testing our services for memory usage, the memory profiling output showed that the processes have a similar baseline memory usage of approximately 40–42 MiB. This baseline is reasonable, considering tge contribution of the imported libraries (e.g.,

SQLAlchemy, FastAPI), ORM, and database session management, due to the need to allocate memory resources for data retrieval. We can also notice the small increments between lines, which would represent the additional memory consumed by the operations. For example, the largest observed increment is +1.2 MiB, occurring during a database commit (db.commit()), which is expected as it involves allocating memory for the transaction and refreshing the object in memory. Most other operations, such as creating objects or fetching database results, contribute minimal increments (typically +0.1 MiB or less), highlighting the efficiency of the implementation and the extremely minimal memory usage. The overall peak memory usage of 42.2 MiB, even after performing multiple operations like fetching, updating, and deleting, indicates a relatively small and efficient memory footprint. This pattern suggests that the memory usage of the service is well-optimized and suitable for the workload.

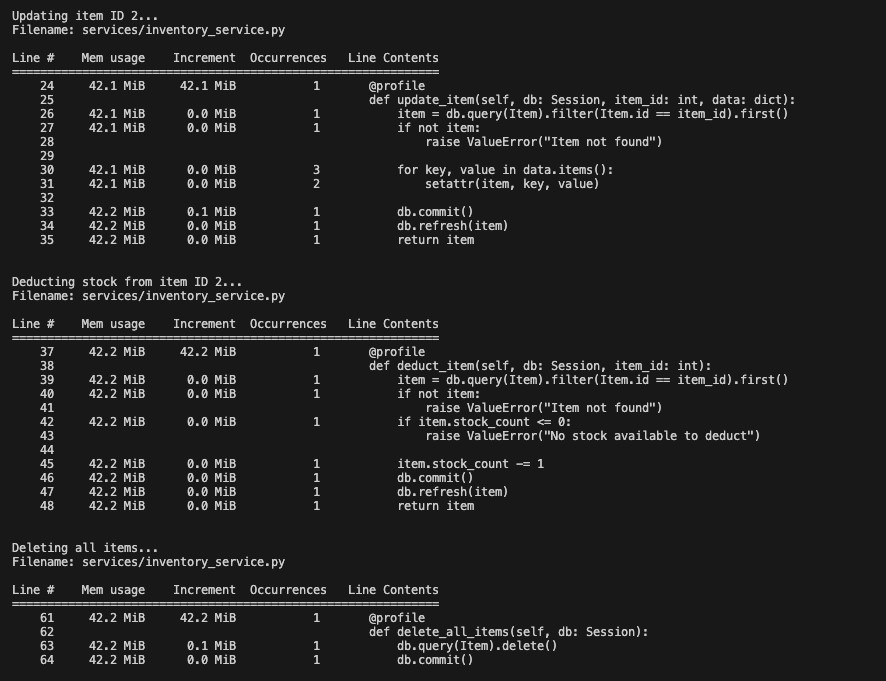
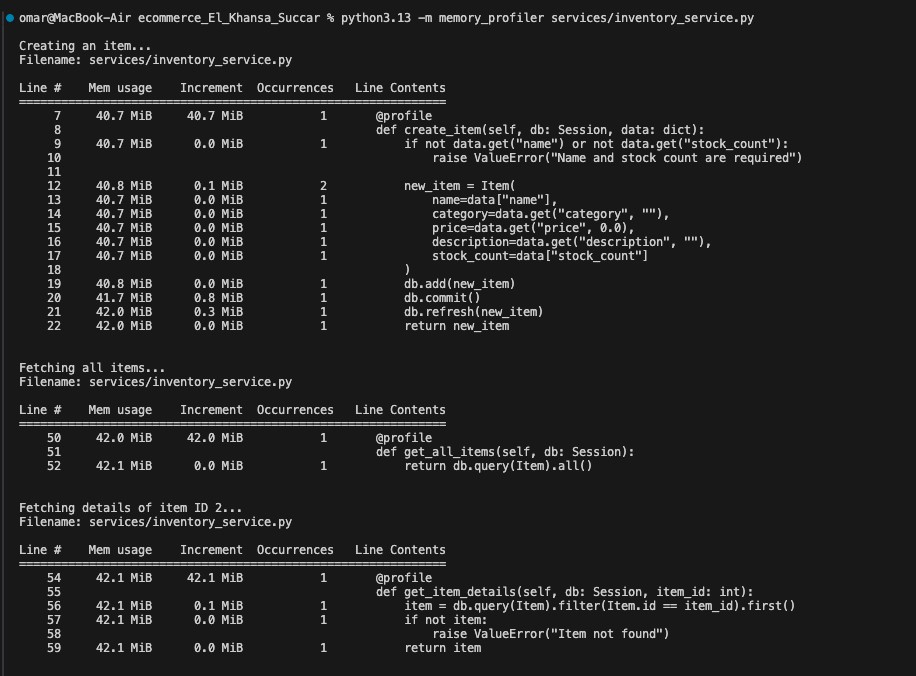
*Customer Service Memory Usage*



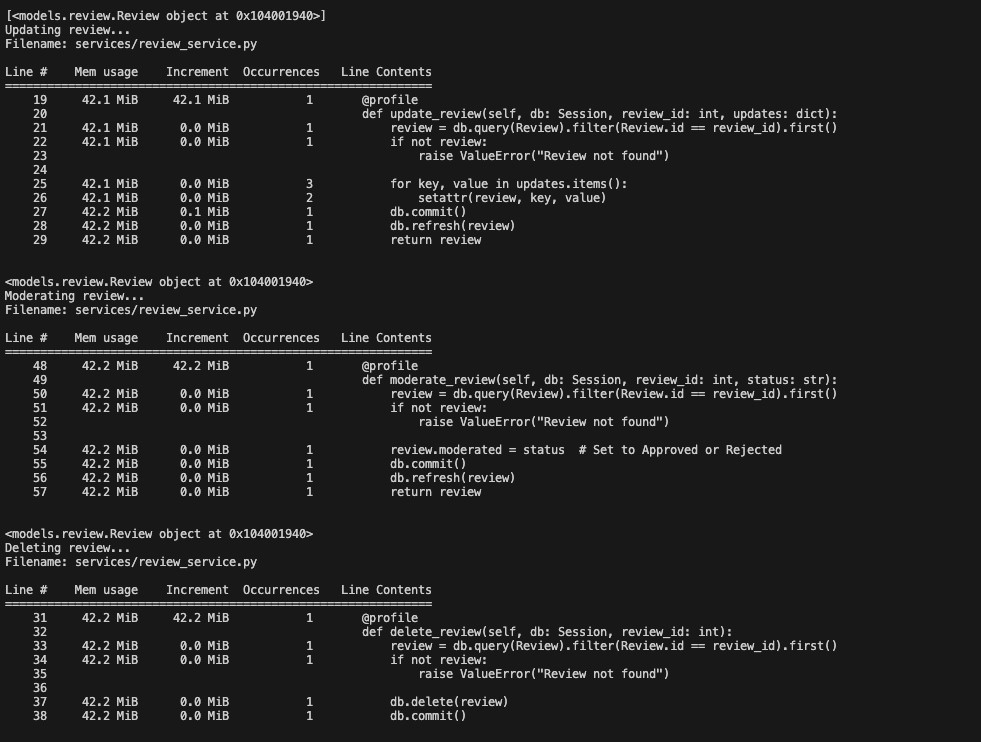
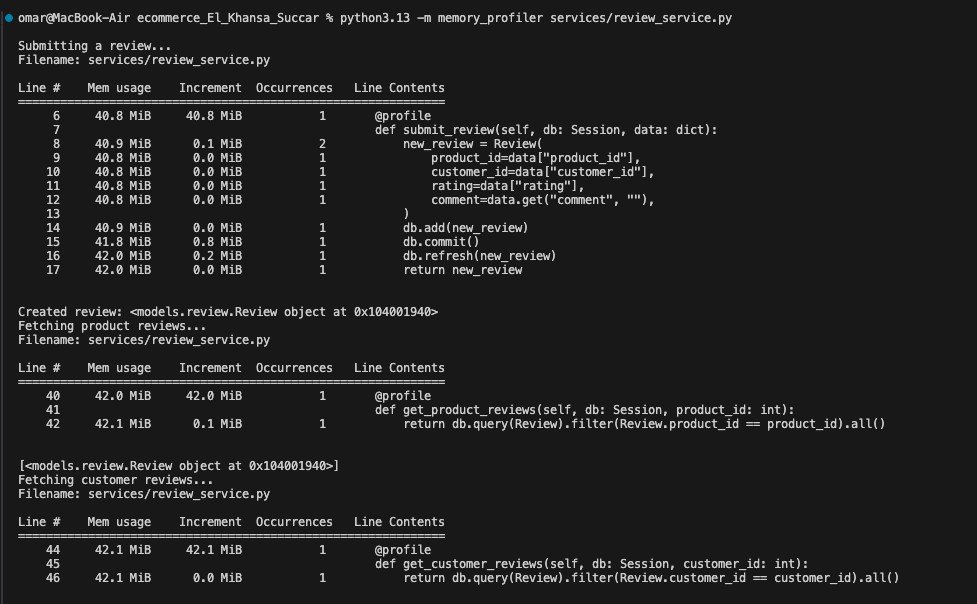


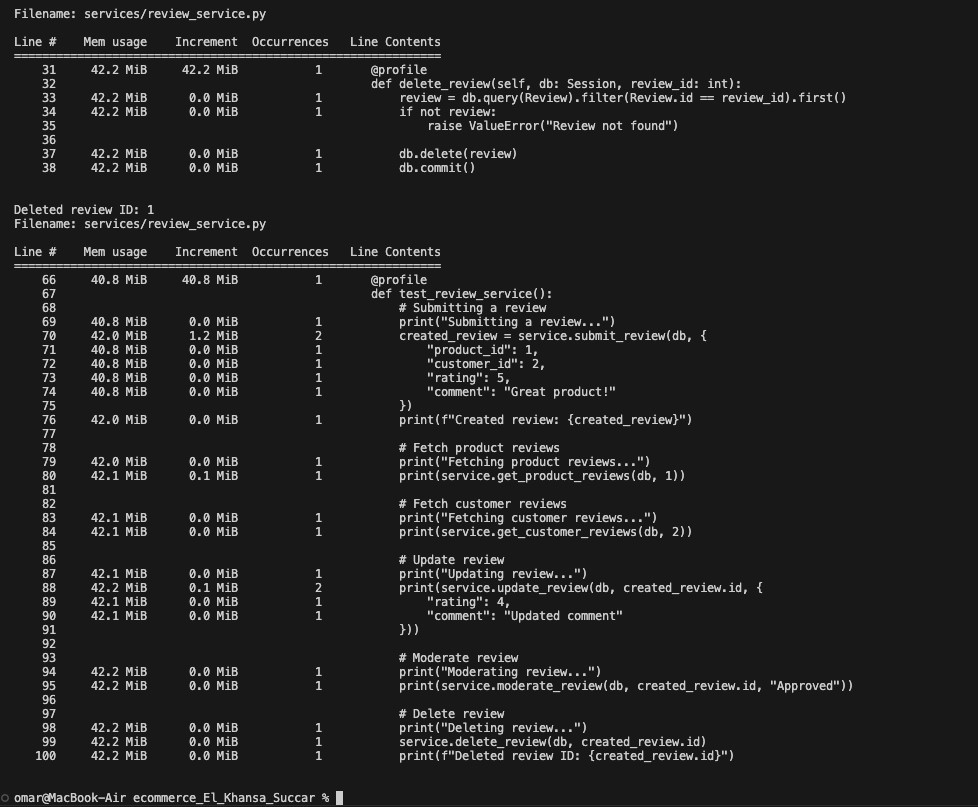


*Inventory Service Memory Usage*



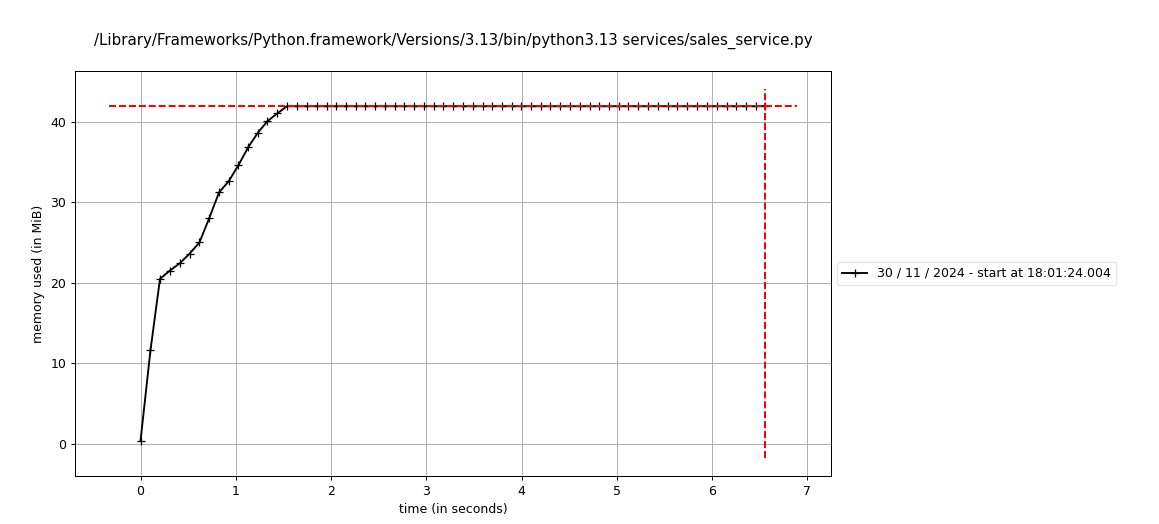
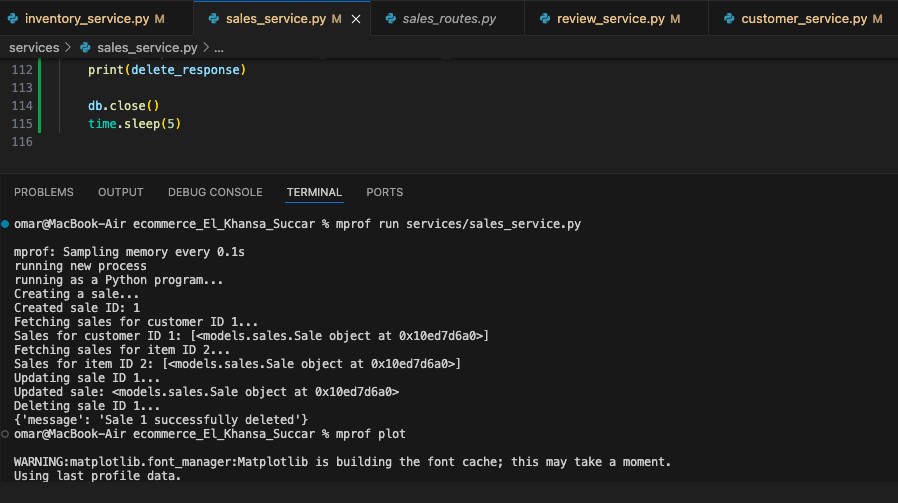
*Reviews Service Memory Usage*





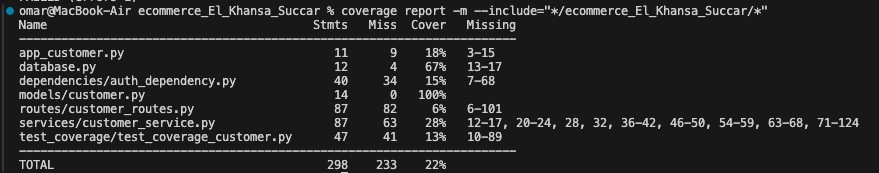
*Sales Service Memory Usage*

Trying to see memory usage for the sales service, the output only showed the printed statements without the memory usage table. Searching for the problem, I found that I can check the memory usage by plotting it using matplotlib in case our usual method does not work. I introduced a time delay, and the plot was generated and is attached below, showing results similar to the above with 41 MiB at the beginning with 0 increments after that.

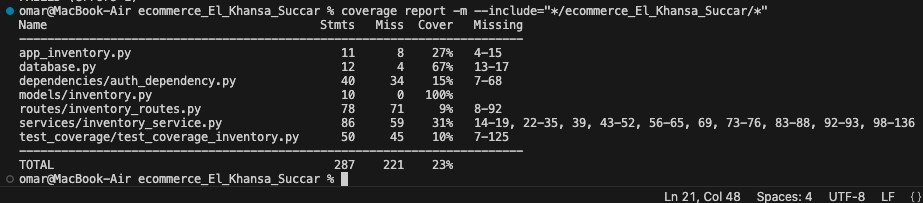


### A. Coverage

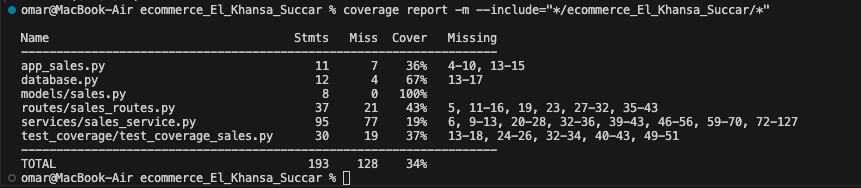
*Customer Coverage Report*



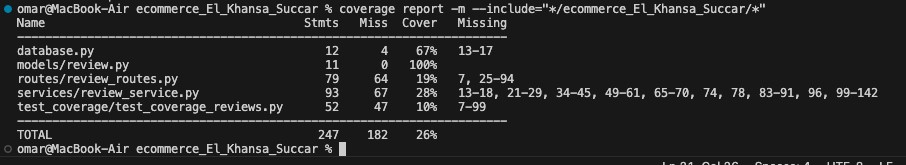
*Inventory Coverage Report*



*Sales Coverage Report*



*Reviews Coverage Report*



We can notice that there is a pattern in missing lines. In all the app\_ “service x”.py files, the uncovered lines are since the lines include initialization code or configuration that isn’t invoked during tests. The coverage test will also not cover error handling or fallback logic scenarios.

In the database.py, the uncovered lines are due to the fact that they are un-invoked during testing, as these are primarily one-time setup lines or conditionally executed blocks.

In the auth dependencies files, the uncovered lines are the dependency functions (decode\_jwt, get\_current\_user, require\_admin).The reason is due to the fact that tests are not explicitly checking for authentication or authorization. Some of the uncovered lines handle authorization scenarios and are thus not triggered by the tests.

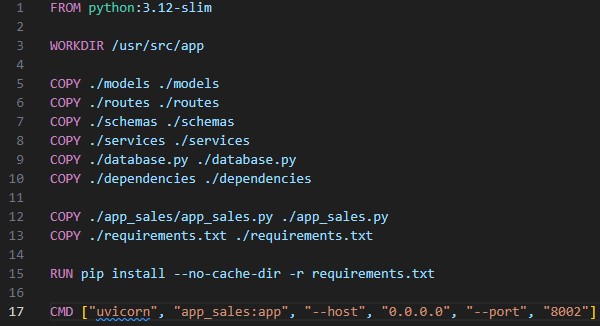
Dependency functions like decode\_jwt and require\_admin are only invoked if endpoints requiring these dependencies are tested.

Services files tests do not directly invoke service methods; instead, they test endpoints in routes, which depend on these methods indirectly. Several handling and validation logic (exception handling, validations in create\_item or deduct\_item) are not covered in the tests as well, which explains the coverage pattern we noticed above.

## Deployment and Integration (Ibrahim’s Work)

The services in this application were containerized using Docker for seamless deployment and integration. Below are the configurations and steps:

1. **Dockerfiles**: Each microservice has its own Dockerfile to build its respective image. For example, the Dockerfile for the app\_sales service is structured as follows:

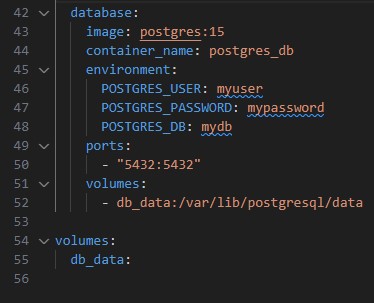
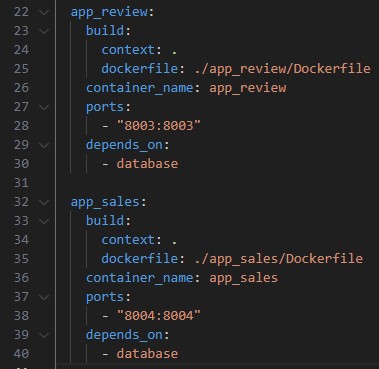
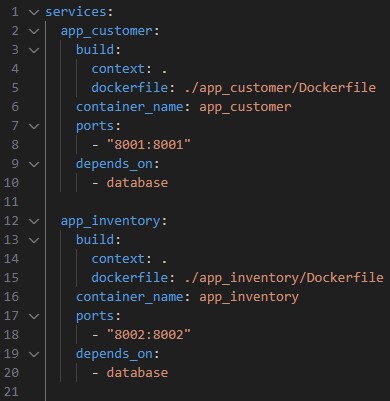


This Dockerfile:

* + Uses the python:3.12-slim base image.
  + Sets the working directory to /usr/src/app.
  + Copies necessary code files and dependencies.
  + Installs Python packages from requirements.txt.
  + Defines the command to run the service with uvicorn.

1. **docker-compose.yml**: The docker-compose.yml file orchestrates multiple services. Below is a breakdown of its structure:

o The docker-compose.yml: o Defines a PostgreSQL database service named postgres\_db with required credentials. o Builds and runs the app\_sales service, linking it to the database.



o Maps ports for external access (e.g., 5432 for the database, 8003 for the sales app).

1. **Integration**: o Each service runs in its own container but communicates via a shared network defined in docker-compose.yml. o Dependencies like the postgres\_db ensure the database is ready before the application services start.

o Environment variables provide database connection details dynamically.

1. **Execution Steps**:
   * 1. Build the Docker images: docker-compose build
     2. Start the containers: docker-compose up
2. **Benefits**:
   * **Isolation**: Each service is containerized, ensuring minimal interference.
   * **Reproducibility**: The Dockerfile and docker-compose.yml configurations enable consistent environments across systems.
   * **Scalability**: Additional services or replicas can be added easily.

Documentation and Profiling (Omar’s Work)

All our code is documented using Docstrings and Sphinx. The report is in the repository.

Github and Version Control (Ibrahim’s Work)

Link: <https://github.com/Ibrahim-elKhansa/ecommerce_El_Khansa_Succar>

## Docker (Ibrahim’s Work)

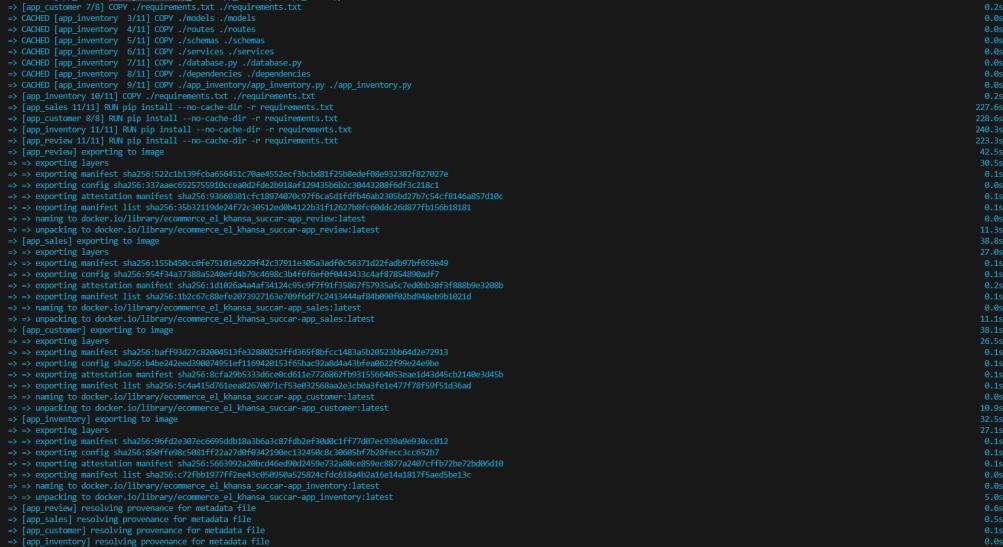
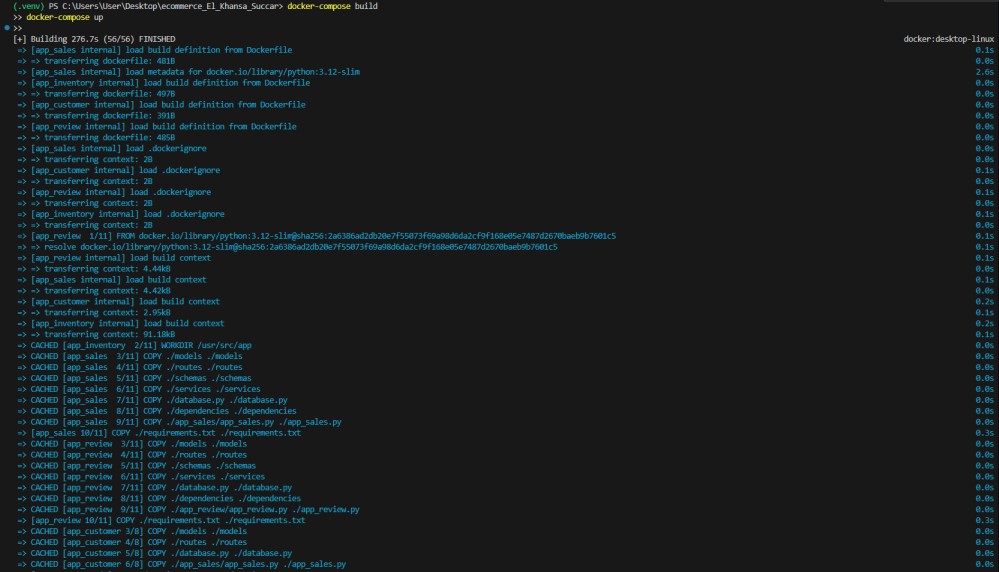
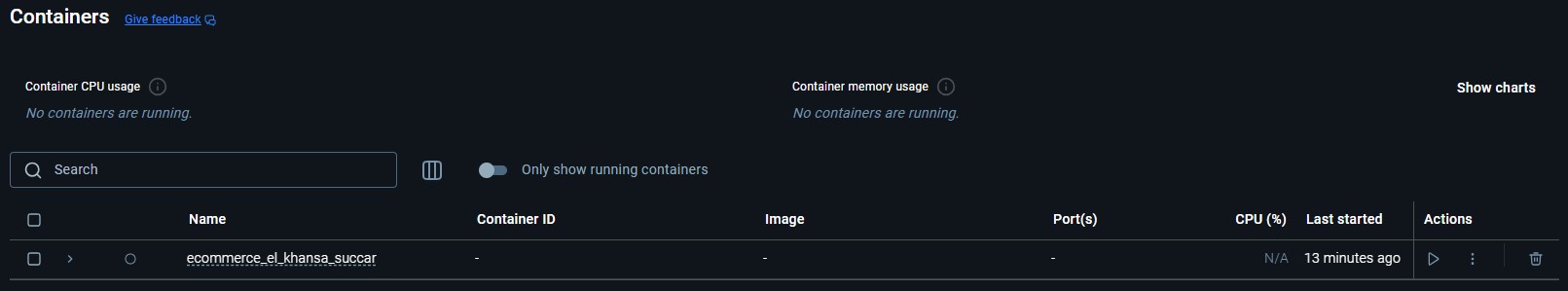
To ensure consistent and efficient deployment across various environments, the application and its dependencies were packaged into Docker containers. Docker allows for an isolated runtime environment, guaranteeing that the application behaves the same way irrespective of where it is deployed.

***Packaging Process:***

**Dockerfile**:

a. Each microservice has a dedicated Dockerfile to package the service. These Dockerfiles:

1. Use lightweight base images (e.g., python:3.12-slim).
2. Set up the required working directory and copy the service-specific files. iii. Install dependencies from requirements.txt. iv. Define the service's runtime behavior using CMD.

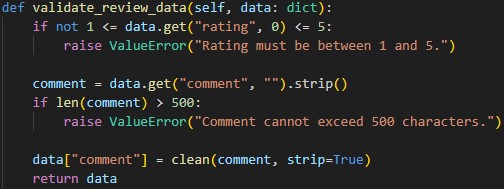


## Validation and Sanitization (Ibrahim’s Work)

Validation and sanitization processes are implemented to ensure user inputs for reviews are secure and prevent vulnerabilities like SQL injection or cross-site scripting (XSS).

**Implementation Details:**

1. **Validation:** 
   1. Ratings are validated to ensure they fall between 1 and 5. Any input outside this range is rejected.
   2. Comments are checked for length, ensuring they do not exceed 500 characters.
   3. Mandatory fields like product ID, customer ID, and rating are verified for presence.
2. **Sanitization:** 
   1. The bleach library is used to sanitize comments, stripping out any potentially malicious scripts or HTML tags.
   2. Inputs are cleaned before they are stored in the database, ensuring data integrity and security.



## User Authentication (Ibrahim’s Work)

Authentication ensures that only authorized users can access, modify, or delete resources such as customer or review data.

**Implementation Details:**

1. **Authentication:** 
   1. Each API endpoint is secured with a token-based authentication system.
   2. The user's token is validated using the Authorization header in each request.
2. **Authorization:** 
   1. Role-based access control (RBAC) is implemented. For example:

i. Admin users can moderate reviews. ii. Regular users can submit and update reviews but cannot moderate.

## Moderation (Ibrahim’s Work)

Moderation features allow admins to approve or reject reviews flagged as inappropriate by users or detected automatically.

**Implementation Details:**

1. **Approval or Rejection:** 
   1. Admins can update the moderation status of reviews (Approved or Rejected).
   2. Only admins are authorized to perform this action, ensured through role-based authentication.
2. **Pending Reviews:** 
   1. A dedicated endpoint fetches reviews marked as "Pending" for admin review.

## Addition Professional Tasks

***Enhanced Inter-Service Communication (Omar’s Work)***

1. **Circuit Breaker Pattern**

The circuit breaker pattern was implemented using the pybreaker library to ensure fault tolerance across interservice communication. For instance, when the Sales API is down, the circuit breaker temporarily halts requests to the service and switches to a fallback mechanism, preventing the system from overloading or timing out during service failures.

* 1. **Implementation**:
     1. Added pybreaker to handle service communication failures. ii. Configured CircuitBreaker with fail\_max=5 and reset\_timeout=30 to retry after 30 seconds if 5 consecutive failures occur.
  2. **Use Case**: Protect communication with the Review and Inventory services. c. Screenshot:

1. **Rate Limiting and Throttling**

Rate limiting was implemented using the slowapi library to prevent abuse or overload of the API services. The limit was configured to allow a maximum of 10 requests per minute per IP address.

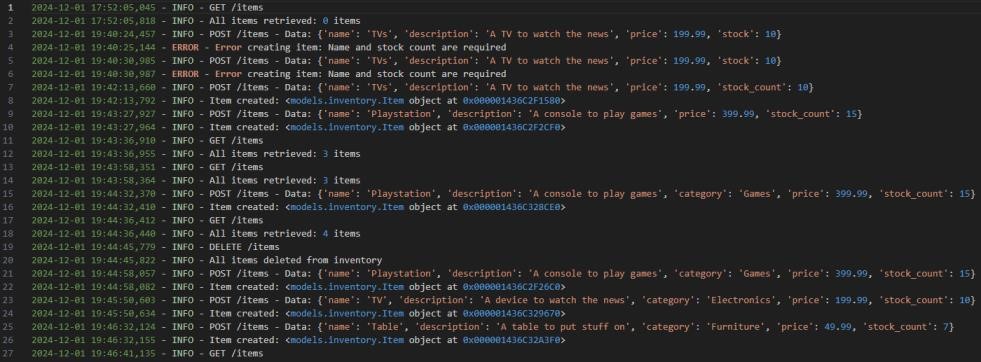
* 1. **Implementation**:
     1. Integrated Limiter using the get\_remote\_address key function.
     2. Applied limits to endpoints like /reviews/{review\_id}/update to prevent excessive update operations.
  2. **Use Case**: Safeguard against potential API misuse during high traffic events. c. Screenshot:

***Advanced Security Measures (Ibrahim’s Work)***

1. **Auditing and Logging**

Detailed auditing and logging mechanisms were added to track API usage, errors, and operation histories. Logs are stored in categorized log files, such as log\_review.log, for each service.

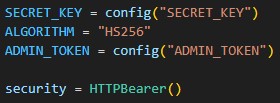
* 1. **Implementation**:
     1. Configured Python's logging module to log critical events, including errors, API requests, and user actions.
     2. Separate log files for customer, review, and inventory services.
  2. **Use Case**: Ensure audit trails for critical operations like review updates, customer creation, and item deductions. c. Screenshot:



1. **Encryption**

End-to-end encryption for sensitive data was ensured using HTTPS and encrypted database credentials.

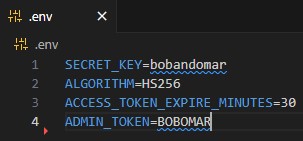
* 1. **Implementation**:
     1. Deployed FastAPI over HTTPS for encrypted data transmission.
     2. Stored sensitive database credentials in an .env file, securely accessed using python-decouple.
  2. **Use Case**: Safeguard sensitive customer and review data during API communication. **c. Screenshot**:



1. **Secure Configuration Management**

Sensitive configuration data like API keys and database credentials were stored securely using environment variables accessed through python-decouple.

* 1. **Implementation**:
     1. All sensitive keys (e.g., admin tokens) were stored in the .env file.
     2. Used config() to fetch these keys securely in code.
  2. **Use Case**: Prevent accidental exposure of sensitive credentials in the codebase. c. **Screenshot**:



These enhancements significantly improved the robustness, security, and reliability of the system.