Final Project – teams of two/solo

In this project you will be creating a backend web services of an ecommerce website. The project is made of four services and a database. These services communicate with each other via API calls. All these services should be containerized and run locally on your Docker engine. Below is a description of each service. You are not supposed to develop the frontend for the project. You will be testing the project using Postman and API calls.

Service 1 - Customers

This service is responsible of registering customers and managing their credits and payments. You are supposed to develop the following APIs:

- Customer registration: registering a new customer. The service should save the following information:
    - Full name
    - Username and password. The username should be unique. The service should check that the username is unique and return an error if it is taken.
    - Age
    - Address
    - Gender
    - Marital status
- Delete customer
- Update customer information. It should update one or many fields related to the customer information.
- Get all customers
- Get customer per username. The username should be unique.
- Charge customer account/wallet in dollars
- Deduct money from the wallet.

Service 2 - Inventory

This service is responsible of managing the inventory of goods. You are supposed to develop the following APIs:

- Adding goods:
    - Name
    - Category (food, clothes, accessories, electronics)
    - Price per item
    - Description
    - Count of available items in stock
- Deducting goods: removing an item from stock
- Updating goods: updating fields related to a specific item

Service 3 - Sales

This service is responsible of displaying goods to customers and managing sales. It should provide the following APIs:

- Display available goods: this API should return a list of
  - Good name
  - Price
- Get goods details: This API should return full information related to a specific good
- Sale: when a customer makes a sale, this API takes the name of good, the customer username, checks if the user has enough money to purchase and if the good is still available. If all the conditions are met, the service should deduct the money from the customer wallet and decrease the count of the purchased good from the database.
- You should find a way to save all the historical purchases made by a customer

Service 4 - Reviews

Description: The Reviews service manages product reviews and ratings submitted by customers. This service allows customers to provide feedback on products, which can help other customers make informed decisions. It also enables administrators to manage and moderate these reviews.

The Reviews service is designed to manage and facilitate customer feedback on products within the eCommerce platform. It provides a set of APIs to allow customers to submit, update, and delete reviews, while also enabling administrators to moderate and manage the reviews. Key functionalities include:

- Submit Review: Allows customers to post reviews for specific products, including a rating and comment. This API ensures that feedback is recorded and associated with both the customer and the product.
- Update Review: Enables customers to modify their existing reviews, updating the rating and comment as needed. This API supports maintaining accurate and up-to-date feedback.
- Delete Review: Provides functionality for customers and administrators to remove reviews. This helps in managing inappropriate or outdated feedback.
- Get Product Reviews: Retrieves all reviews for a particular product, offering insights into customer experiences and satisfaction levels.
- Get Customer Reviews: Lists all reviews submitted by a specific customer, which is useful for tracking individual feedback history.
- Moderate Review: Allows administrators to moderate reviews by flagging or approving them, ensuring the review system remains free from inappropriate content.
- Get Review Details: Provides detailed information about a specific review, including the username, product name, rating, and comments, for transparency and further analysis.

This service ensures that the review process is streamlined and that feedback is handled efficiently, enhancing the overall customer experience and maintaining the integrity of product evaluations.

Mandatory Work to be done(80%)

1. Name your project ecommerce_FamilynameMember1_FamilynameMember2 in case you are in a team or only ecommerce_Familyname if you are solo.
2. Develop 4 services that runs on Docker using different ports.
3. You can use the database of your choice. You can create it from scratch or you can use Dockerhub to download DB image and run it on a dedicated container.
4. Create your project on VScode and your virtual environment.
5. Start your design by writing your API calls for each service using Postman. Create a collection for the project and for each API call write an example, description and comments for the fields.
6. Create a Github repo for the project and start by the service of your choice. I need to see that you were using Github effectively by pulling/pushing your code regularly.
7. Write Pytest for all the services. These tests should insure that your code is well tested when making new changes.
8. Do Documentation using Docstings and Sphinx.
9. Do performance, memory, and code coverage profiling to your code. Include appropriate sanpshots in your report.
10. Package your applications and your dependencies into containers for consistent and efficient deployment across various environments.
11. Validation and Sanitization: Ensure that user inputs for reviews are validated and sanitized to prevent SQL injection and other security issues.
12. User Authentication: Ensure that only authenticated users can submit, update, or delete reviews. Describe how user authentication is handled and how authorization is managed.
13. Moderation: Implement moderation features to handle inappropriate reviews, which could be flagged by users or administrators.

Additional Professional tasks(20%) .Choose 2(as minimum) or more from the 25 tasks listed below. P.S. you are not required to pay for the services as some require free subscription. No paid hosting is needed:

1. Enhanced Inter-Service Communication

- **Circuit Breaker Pattern**: Implement fault-tolerance mechanisms to handle failures when one service becomes unavailable. Use libraries like `pybreaker`.
- **Rate Limiting and Throttling**: Apply rate-limiting strategies for APIs to prevent abuse or overload of individual services.

2. Advanced Security Measures

- **Auditing and Logging**: Implement detailed logging of API requests/responses, capturing timestamps, user details, and operation types for audit purposes.
- **Encryption**: Use end-to-end encryption for sensitive data transfers (e.g., customer information).
- **Secure Configuration Management**: Store sensitive data like API keys and database credentials in a secure vault (e.g., HashiCorp Vault or AWS Secrets Manager).

3. Performance Optimization

- **Caching Mechanism**: Introduce caching with tools like Redis or Memcached for repetitive database queries (e.g., fetching product details or customer data).
- **Load Balancing**: Simulate load balancing for services using tools like HAProxy or Nginx.
- **Database Indexing**: Optimize database queries by creating proper indexes and measuring query performance using `EXPLAIN` statements.

4. Scalability and Reliability

- **Asynchronous Messaging**: Implement a message broker like RabbitMQ or Kafka for asynchronous communication between services (e.g., notifications after a sale).
- **Health Checks**: Add health-check APIs for all services to monitor their availability and performance.
- **Horizontal Scaling**: Configure Docker Compose or Kubernetes to scale specific services based on demand.

5. Analytics and Insights

- **Dashboards for Monitoring**: Create real-time dashboards using Grafana or Kibana to monitor service performance (e.g., API latency, error rates).
- **Data Analytics Service**: Introduce a dedicated analytics service to aggregate and analyze data, like total revenue trends or customer demographics.

6. User and Access Management

- **Multi-Factor Authentication (MFA)**: Add MFA for sensitive operations like payments or account updates.
- **Role-Based Access Control (RBAC)**: Implement granular permissions for admin and customer roles within the API.

7. Advanced Development Practices

- **Custom Exception Handling**: Build a standardized exception-handling framework across services for clear error messaging.
- **Versioning APIs**: Implement API versioning to ensure backward compatibility as services evolve.

8. Continuous Monitoring and Alerts

- **Prometheus Integration**: Set up Prometheus to monitor service health and configure alerting for anomalies (e.g., CPU spikes or downtime).

- **Error Tracking**: Use tools like Sentry to track and analyze runtime errors in the services.

9. Extending the Project Scope

- **Recommendation System**: Add a recommendation engine for customers, suggesting products based on purchase history (e.g., using collaborative filtering or simple rules).
- **Wishlist or Favorites**: Enable customers to save products they're interested in purchasing later.
- **Abandoned Cart Feature**: Track and notify customers about unpurchased items they left in their cart.

10. Integration with Third-Party Services

- **Payment Gateway Integration**: Simulate integration with a payment gateway (e.g., Stripe or PayPal API) for external payment handling.
- **Shipping API**: Integrate with a mock or real shipping API to calculate delivery costs and update order statuses.

11. Security Measures

- **Security Practices:** Explain measures taken to prevent common vulnerabilities (e.g., SQL injection).

Report:

P. S.
For each of the below titles in your project you are supposed to mention the name of the team member who did it if you are in a duo team. Please note that there will be no grade for the entire project but each student will be graded separately based on what he/she did, so divide the services and subservices evenly between you.
Each team should choose minimum of two (or more as optional) from the additional tasks as a whole for the whole project,i.e. one additional task per student (or more as optional).
Your project shouldn't be the same(I.e. copy paste) as any other team /solo member or this will be considered as plagiarism.
For each of the sections in your project, include the necessary snapshots to prove your work.

Project Report Structure
1. Title Page

- **Project Title**
- **Student Name(s)**
- **Date of Submission**

2. Table of Contents

- List of sections with page numbers and team member name who worked on each.

3. Introduction

- **Project Overview:** Briefly describe the project, its purpose, and its scope.
- **Objectives:** What were the main goals of your project?

4. System Architecture

- **Service Description:** Describe each of the services (Customers, Inventory, Sales, and Reviews) and their responsibilities.

5. Implementation Details

- **Service-Specific Implementation:**
  - **Service 1 - Customers:** Explain the functionality, key APIs, and any challenges faced.
  - **Service 2 - Inventory:** Describe how goods are managed, added, and updated.
  - **Service 3 - Sales:** Outline how sales are processed, and historical data is handled.
  - **Service 4 - Reviews:** Detail the review submission, updating, moderation, and retrieval processes.
- **API Documentation:** Include descriptions of API endpoints, request/response formats, and all calls. Provide screenshots or snippets from Postman collections.

6. Database Design

- **Schema Diagram:** Provide a diagram of the database schema showing tables and corresponding fields.

7. Error Handling and Validation

- **Error Management:** Describe how errors are handled in the system.
- **Validation:** Explain what types of validation are implemented and where.

8. Testing

- **Testing Strategy:** Outline the approach to testing, including unit tests, integration tests, and any other relevant testing.

- **Test Cases:** Provide examples of all test cases and corresponding results. Include screenshots or summaries of Pytest results.

9. Deployment and Integration

- **Docker Setup:** Describe how Docker was used to containerize the services. Include Dockerfile and docker-compose.yml configurations.

10. Documentation and Profiling

- **Documentation:** Provide links or references to your project documentation created with Sphinx or other tools.
- **Performance Profiling:** Do performance, memory, and code coverage profiling to your code.Include appropriate snapshots in your report.

11. GitHub and Version Control

- **Repository Links:** Provide links to the GitHub repository and highlight important branches or commits.

12.Docker and images:
Describe how you packaged your applications and your dependencies into containers for consistent and efficient deployment across various environments. Provide appropriate screenshots.

13.Validation and sanitization:

Describe how and where you did validation and sanitization to ensure that user inputs for reviews are validated and sanitized to prevent SQL injection and other security issues.Include appropriate screenshots

14. User Authentication:

Describe how you ensure that only authenticated users can submit, update, or delete reviews. Describe how user authentication is handled and how authorization is managed. Include appropriate screenshots.

15. Moderation:

Describe how you Implemented moderation features to handle inappropriate reviews, which could be flagged by users or administrators. Include appropriate screenshots.

16. Additional Professional tasks:

Describe how you Implemented additional features from the available list and what did you handle and how. Include appropriate screenshots.

References

- Include any references or resources used during the project.

Appendices

- **Appendix A:** Screenshots, code snippets, or additional details that support the report.

Submissions to Moodle:
You are supposed to upload to Moodle your full project zipped in addition to the report and database file.
Good luck!