Project Mistakes

UI should be proper

Global Exception handling should be done—should not show the error page

Product stock should check while purchasing the product

After placing order ,product count should get reduce from the original stock

If any product get cancel, the stock count has to update

Without purchasing the product, review should not give

All process should show the completeness even if it is a mock

In ER diagram no table should be left without any relation

New category add facility should be there

Filtering the product should be based on the common usage

Once after completing each process , a message should be shown.

Registration successfully!!

Logging should be like ..its start to end ……

Already used addres should be shown while purchasing the product, from there you should be able to get the address

Once a user login , give the user name …add use session…..

Once a user order a product ---default status –dispatched, admin the status like out for delivery …..it should have the option

Filter the products status ….

P2 Project Solution architecture

Refer the github Solution architecture picture

P2 Tech stack

**Design patterns for our Microservice project:**

1. **Eureka server**
2. **Docker compose**
3. **Auth0**
4. **resilience4j**
5. **Spring Gateway**
6. **Spring cloud gateway**
7. **Prometheous**

Standard Expectations

1. User input validation at the client level – form validations should be done
   1. For email id and phone number, date (UTC) format.
   2. For mandatory fields.
2. User should get appropriate error or information message for their actions on the respective pages
3. Application should have very intuitive user interface design with professional color themes and images/icons.
4. Application should use bootstrap framework with optional material UI design for its responsiveness.
5. Application should have centralized logging through Spring AOP
6. Application should have application/domain specific custom exceptions with appropriate user messages and should be handled gracefully the errors.
7. Make sure that the user messages for the exception, is coming from the properties file.
8. Application should have proper layers such as Controllers, Business Services, Data Access Objects and other dependent layers.
9. Ensure that the application complies with W3 standards such as right meta tags and usage of alt text and or other required attributes that are compliance to accessibility requirements.
10. Use compressed images / assets to increase the page performance
11. Use the application validated using the Chrome’s Lighthouse tool and improved based on the report
12. Have the token-based authentication using JWT for user authentication.
13. Usage of Java 8 features such as Generics, Collections, Stream API and Lambda is expected.
14. Have an interceptor that validates the user’s auth token before letting them to access the system
15. Ensure the application secrets are stored in environment variables and accessed properly
16. Make sure that the routing is centrally configured for the application
17. Application should use the right design patterns for its layers/components.
18. Ensure the CORS restriction is applied, if applicable.
19. Ensure Route Guarding/Authenticated Routing is implemented
20. Ensure the REST standards are followed for API naming, HTTP Operation and Response (output definition)
21. Secure the protected APIs
22. Define a common URL pattern for public and secure APIs
23. Proper documentation of APIs with Input and Output Samples to be documented
24. Application API should have proper error codes/information codes using the HTTP status codes and relevant messages to the caller.
25. Ensure that each micro service is defined to do one job.
26. Provide / enable to API gateway to route the request through a single channel
27. Ensure the micro services are enabled for traceability and better monitoring.
28. Use the recommended design patterns to aggregate the micro services (wherever is applicable)
29. Create and manage CI/CD pipelines using GitHub Actions to automate building, testing, and deploying applications.

Non-Functional Expectations

1. Application development supposed to follow the Scrum process, having daily standup meeting for 15 mins, daily review calls evening with showcasing of the work done. Use Trello board for user story tracking.
2. Application password should be encrypted using appropriate hashing algorithm