Day-1

SOLID Principles Overview

1. Single Responsibility Principle -> A class should be responsible for handling one responsibility only, hence there should be only one reason for a class to change.
2. Open/Close Principle -> Classes/Interfaces should be open for extension but close for modification.
3. Liskov Substitution Principle -> Objects of a subclass should be replaceable with the object of its subclasses without breaking the system.
4. Interface Segregation Principle -> No client should be forced to depend on methods that it does not use.
5. Dependency Inversion -> Classes should depend on the interface rather than concrete implementation.

Day-2

Richardson Maturity Model Overview

1. Grades the API based on how close the API structure follows the REST Constraints.

2. Following are the Levels of RMM:

a. Level - 0 [POX Swamp, Single URI and Single HTTP Method]

b. Level - 1 [Multiple URIs and Single HTTP Method]

c. Level - 2 [Multiple URIs and Multiple HTTP Methods]

d. Level - 3 [Level 2 + HATEOAS]

Day-3

Testing Framework Overview

1. Unit test refers to testing every unit or part of the program.

2. Junit 5 + Mockito are mostly used for Unit test in java based applications.

3. Ways to implement Unit test are, traditional @Test methods and Parameterized @ParameterizedTest methods.

Day-4

Design Patterns Overview

1. Design patterns in Java are reusable solutions to common problems encountered during software development.
2. Design patterns help developers create code that is more maintainable, scalable, and understandable.
3. Types of Design Patterns:-
   1. Creational Design Pattern
   2. Structural Design Pattern
   3. Behavioural Design Pattern

4. Following are the design patterns that are used in FPAPI front:

a. Factory Design Pattern: Defines an interface for creating objects, but allows subclasses to alter the type of objects that will be created.

b. Builder Design Pattern: Separates the construction of a complex object from its representation, allowing the same construction process to create different representations.

c. Singleton Design Pattern: Ensures that a class has only one instance and provides a global point of access to that instance.

d. Adapter Pattern: Allows objects with incompatible interfaces to work together by providing a wrapper with a compatible interface.

e. Template Method Pattern: Defines the skeleton of an algorithm in the superclass but lets subclasses override specific steps of the algorithm without changing its structure.

Day-5

Build and release Pipeline using Jenkins

1. Jenkins is an open-source automation server written in Java. It is used primarily for continuous integration (CI) and continuous delivery (CD) pipelines in software development.
2. For FPAPI, we have two Jenkins jobs, i.e., Build and Deployment pipelines.
3. Whenever we made a code commit and push it to remote repository, the build job gets triggered.
4. Main job of the build pipeline is to create the build of the application and then create a Docker image corresponding to that build artifacts.
5. The second one is the deployment job. This is mainly responsible for deploying the newly created build(the container image) to Openshift/Mirantis based on the environment (as of now only dev and sit are in Openshift Environment).

Day-6

CDD Overview

1. The Continous Delivery Director (CDD) is mainly used for creating a continuous deployment pipeline.
2. For FPAPI, a CDD pipeline gets created automatically whenever we create a new branch in the remote repo.
3. For creating a new CDD pipeline, we must create a cdd-release-dsl file.
4. The CDD segregates all the Jenkins jobs into a single pipeline.
5. The deployment to DEV environment occurs automatically whenever a commit is made. For other deployments we have to trigger the pipeline manually.

Day-7

Security and Auth 2.0 Overview

1. OAuth 2.0 is an authorization framework that enables third-party applications to access protected resources on behalf of a user without necessarily sharing their credentials.
2. FPAPI uses Okta for API security. Okta is a popular identity and access management (IAM) platform that provides secure authentication, authorization, and user management services for web and mobile applications.
3. Based on the scope of the client, the access to the endpoints are given to the client.
4. FPAPI also has it’s credentials which it uses to call other downstream services such as Conquest and CP APIs.