Assignment:

**Maven life cycle:**

1.Default Life Cycle:

validate,

compile,

test,

package,

verify,

install,

deploy.

2.Clean Life Cycle:

pre-clean,

clean,

post-clean.

3.Site Life Cycle:

pre-site,

site,

post-site,

deploy-site

**What is pom.xml file and why we use it?**

pom.xml (Project Object Model) is the core configuration file in a Maven project. It defines the project structure, dependencies, build configurations, plugins, and more.

We use it because,

Dependency Management,

Build Automation,

Project Information,

Plugin Configuration,

Consistent Build Process.

**How dependencies work?**

In Maven, dependencies are external libraries or JAR files that your project needs to run. Instead of manually downloading and adding JAR files, Maven fetches them automatically from Maven repositories (like Maven Central).

**Check the maven repository:**

Cmd:mvn dependency:tree

**How all modules build using maven?**

In a multi-module Maven project, all modules can be built together using the parent POM file, which acts as a central configuration for the entire project. The parent POM defines the structure and dependencies of multiple sub-modules by listing them inside the <modules> section. Each sub-module has its own pom.xml but inherits configurations from the parent. When you run the 'mvn clean install' command from the root project directory, Maven automatically compiles, tests, and packages all modules in the correct order, considering dependencies between them.

**Can we build specific module?**

mvn clean install -pl module-name - single module.

mvn clean install -pl module-name -am - to its dependencies.

mvn clean install -pl module1,module2 -am - to build multiple modules.

**Role of ui.apps and ui.content and ui.frontend folder?**

ui.apps stores all AEM-specific content like components, templates, dialogs, and configurations and it contains the jcr\_root folder with the /apps and /etc paths for deployment in AEM.

ui.content stores content-related data like pages, assets, and content structure. It contains the jcr\_root folder with the /content path and it is used for deploying sample content (like demo pages or initial site structure).

ui.frontend contains frontend code such as CSS, JavaScript, and client libraries. It uses Webpack, Node.js, and NPM for modern frontend development.

**Why we are using run mode?**

Run Modes in Adobe Experience Manager (AEM) allow you to configure and optimize the system differently based on the environment (e.g., development, staging, production). They help in managing environment-specific configurations without modifying the core application code.

**What is publish env?**

The Publish Environment in Adobe Experience Manager (AEM) is the instance where final content is made available to end users. It is responsible for serving published pages and assets to the audience via a dispatcher or directly.

**Why we are using dispatcher?**

The Dispatcher is a caching and load-balancing tool used in Adobe Experience Manager (AEM) to optimize the delivery of content, ensure better performance, and enhance security in a production environment. It is the front-end cache for AEM's Publish Environment, acting as an intermediary between end users and AEM Publish instances.

**From where can access the crx/de?**

1) Start AEM: Make sure your AEM instance is up and running.

2) Login: You'll be prompted to log in with your AEM credentials (admin by default).

3) Explore CRX/DE: After logging in, you can browse the repository's content and nodes, create or modify data, and interact with the underlying JCR (Java Content Repository).