AEM Assignment

Maven Life Cycle
Maven follows a structured build process consisting of key phases:
1. Clean – Deletes old build files.
2. Compile – Compiles the source code.
3. Test – Runs unit tests.
4. Package – Bundles the compiled code into JAR/WAR.
5. Install – Saves the package in the local repository.
6. Deploy – Sends the package to a remote repository.

Each phase builds upon the previous one, meaning running mvn package will automatically compile and test before packaging.

What is pom.xml and why do we use it?
pom.xml (Project Object Model) is the core of a Maven project. It defines dependencies, plugins, build instructions, and configurations. Instead of manually managing libraries, Maven reads pom.xml, fetches required dependencies, and sets up the project efficiently.

How Dependencies Work?
Dependencies are external libraries needed by the project. In pom.xml, a dependency is added like this:
```
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-core</artifactId>
  <version>5.3.20</version>
</dependency>
```

Maven then downloads it from the repository and stores it in the .m2 folder. It also handles version conflicts and transitive dependencies (dependencies of dependencies).

How Are All Modules Built Using Maven?
In multi-module projects, a parent POM manages linked modules. Running mvn install from the parent directory ensures proper build order, automatically handling dependencies between modules.

Can We Build a Specific Module?
Yes, you can build a specific module without rebuilding the entire project using:
```
mvn install -pl module-name -am
```
This ensures only the specified module is built along with its dependencies.

Role of ui.apps, ui.content, and ui.frontend Folders
- ui.apps – Contains all AEM-related code (components, templates, configurations).
- ui.content – Stores content packages such as pages and assets.
- ui.frontend – Manages frontend elements like CSS, JS, and client libraries.

Each folder serves a unique purpose, contributing to the AEM project structure.

Why Use Run Modes?
Run modes enable AEM to behave differently based on the environment (development, staging, production). They allow specific configurations for logging, users, and features.

Example: author.dev, author.prod – same AEM instance with different configurations.

What is the Publish Environment?
The publish environment is where content is live for users. Unlike the author environment, where content is created and edited, the publish instance is optimized for performance and security.

Why Use Dispatcher?
The Dispatcher acts as a security and performance layer for AEM. It:
- Caches content for faster page loads.
- Filters requests to prevent unauthorized access.

It operates in front of the AEM publish instance to optimize traffic flow.

How to Access crx/de?
You can access CRXDE Lite by visiting:
http://localhost:4502/crx/de
It allows managing JCR content, nodes, and configurations. On a publish instance, replace 4502 with the relevant port (e.g., 4503).