

**Pom.xml use for communicate with maven and provide dependencies**

Maven project structure, we have to tell to the devloper write code in this file.(we = devops engineer)

Maven:  
maven is the the build tool, it is use to create to project folder strucutre

Why we do this? : in order to build app

Different build tool: mvn , ant, gradle

Mvn language for .java

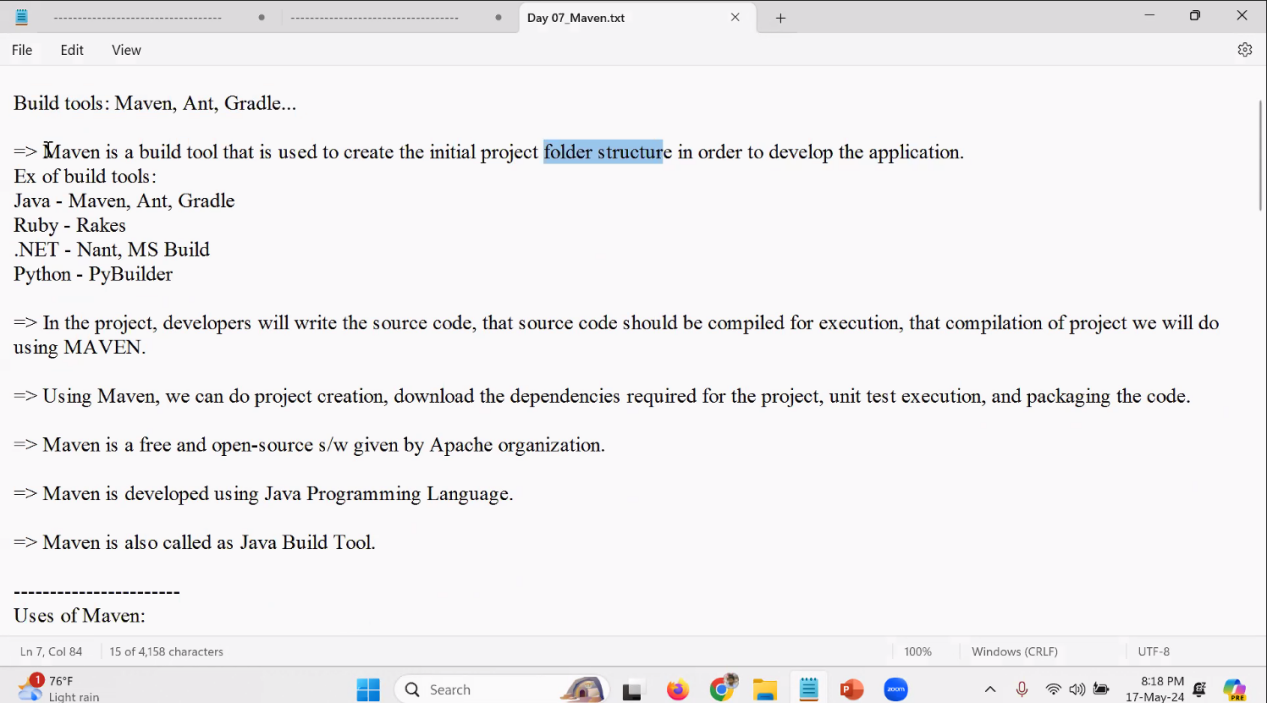
Python - pybuilder

.net- nant or MSbuild

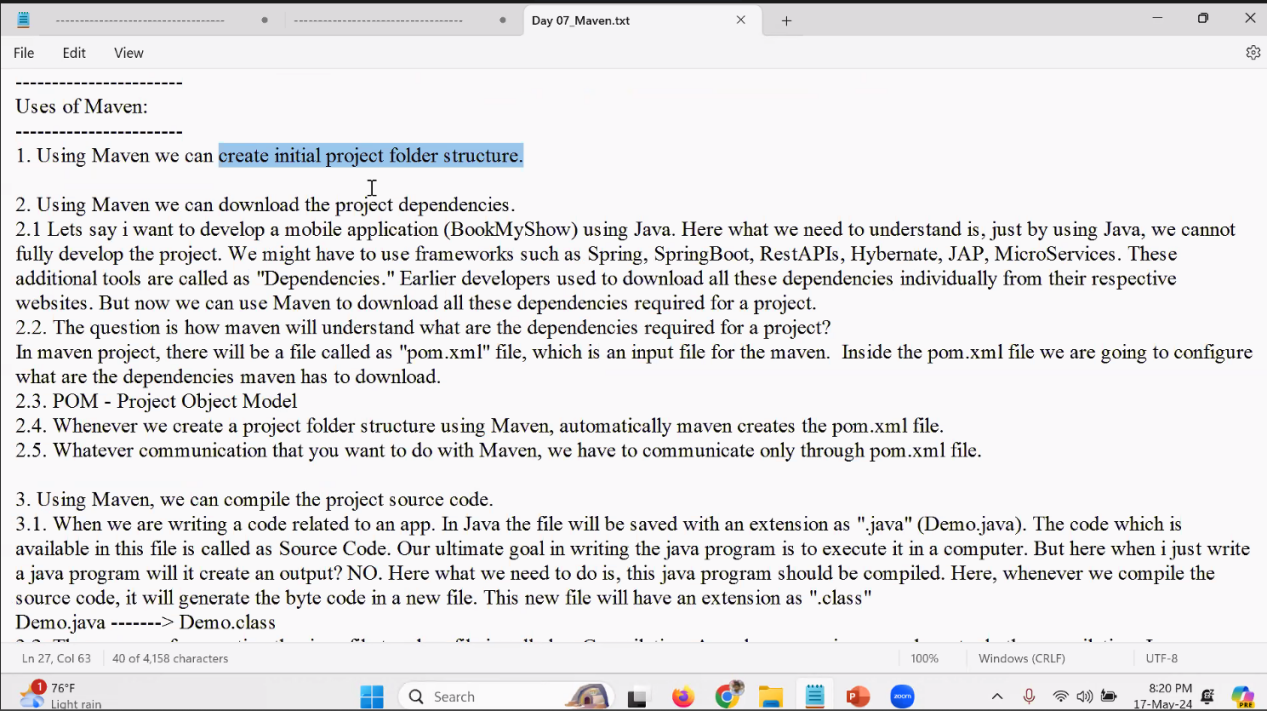
Maven was developed by using java developing language.

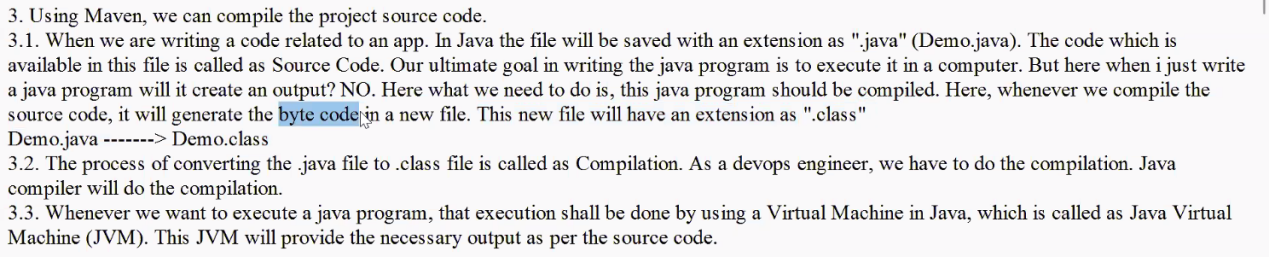
To work with mvn its compulsory to have java s/w in the system

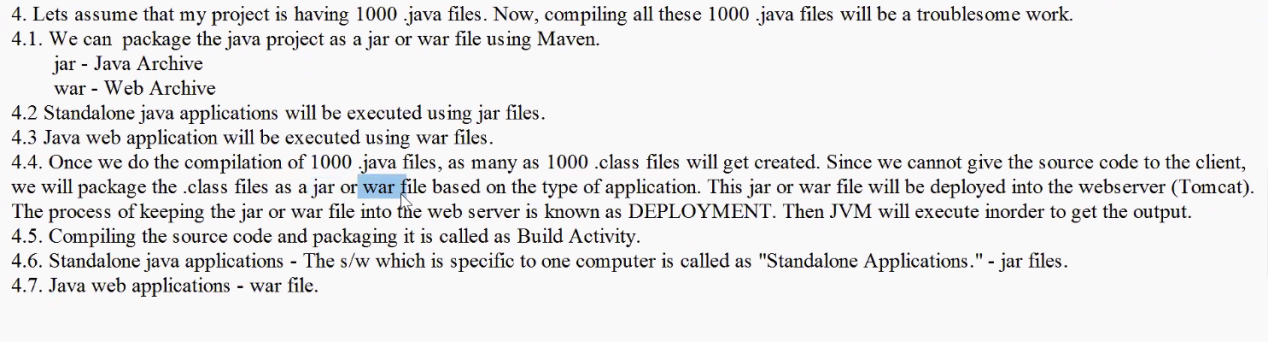
Using mvn we can do project folder structure, We can build application with maven.



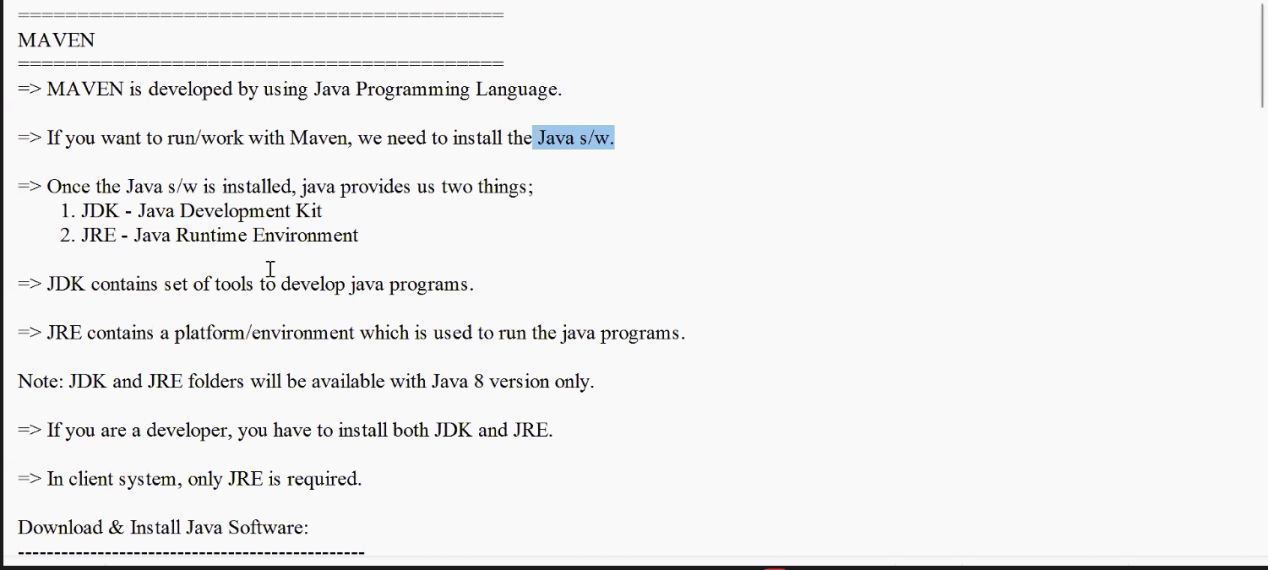
USE CASE OF MAVEN:

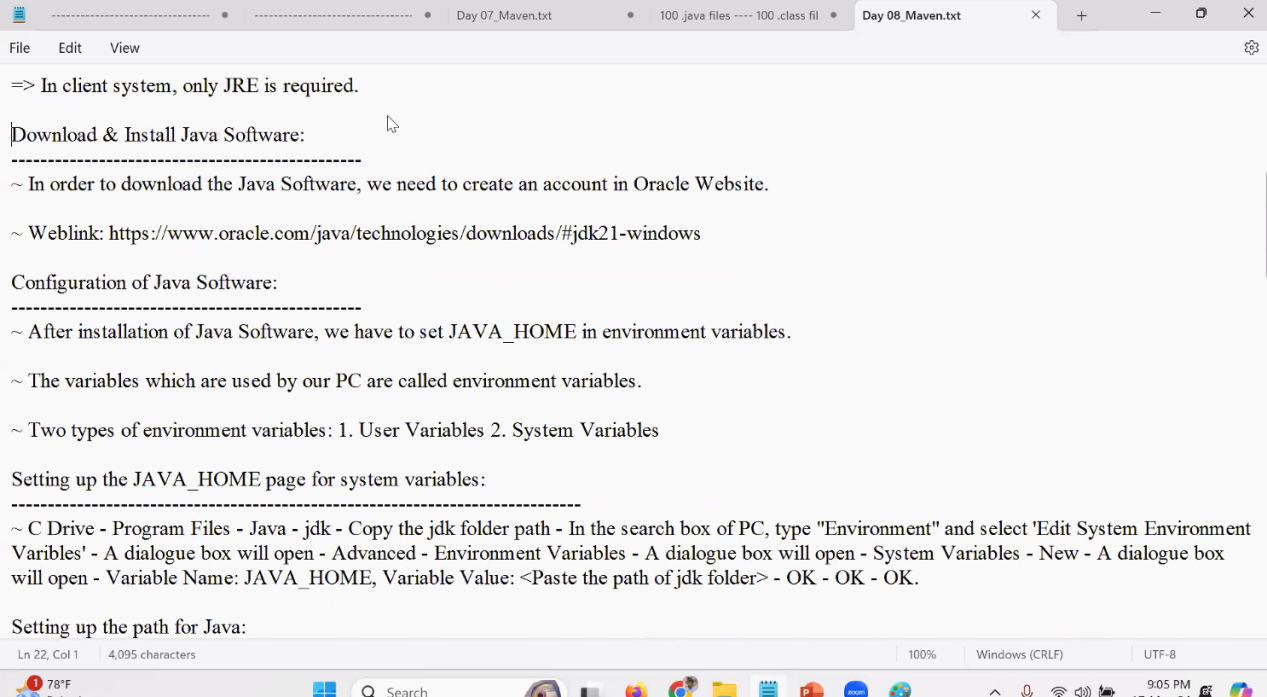


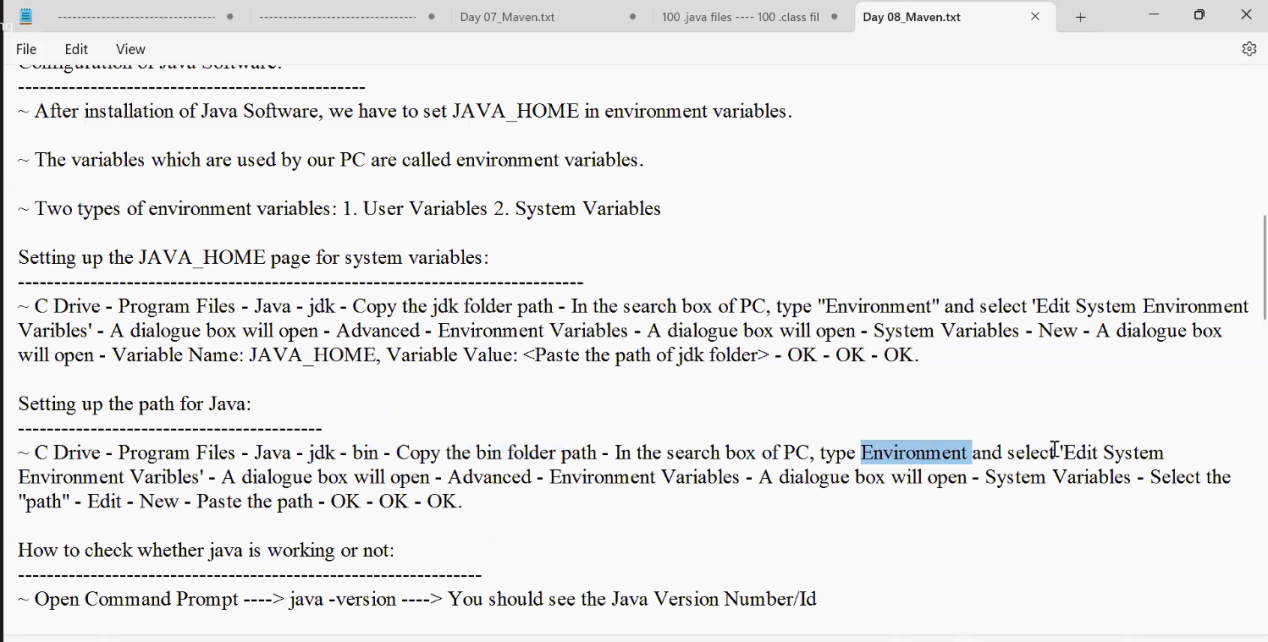


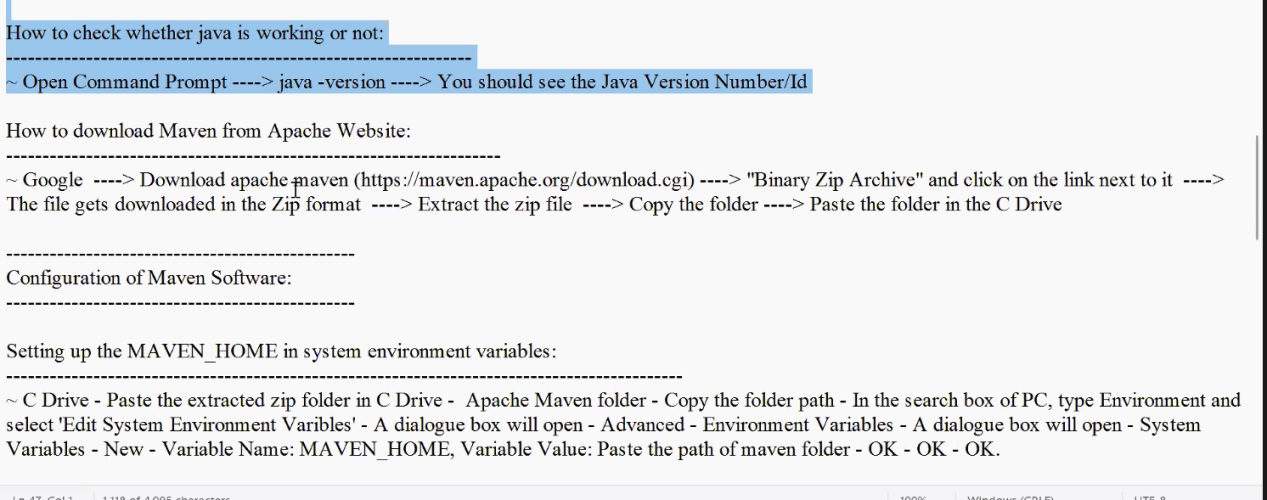


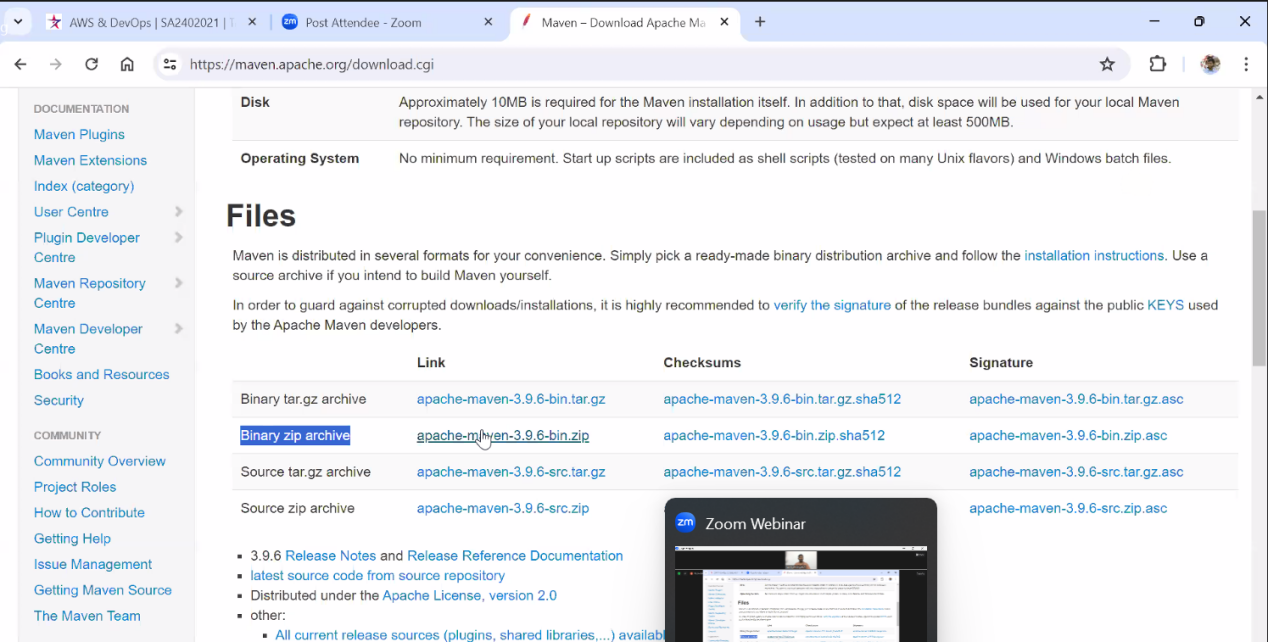
Install MAVEN

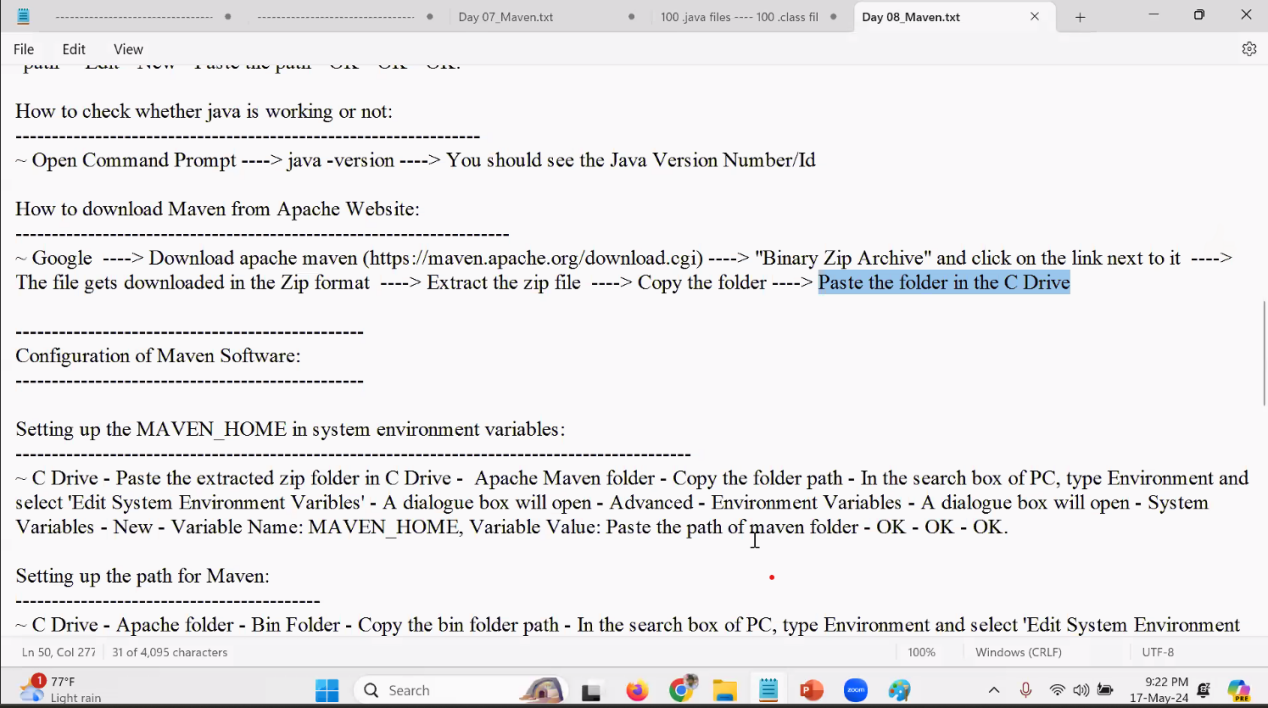


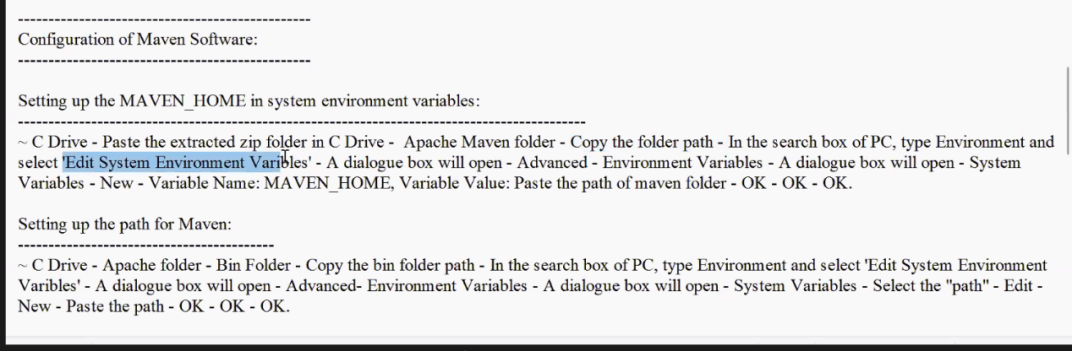






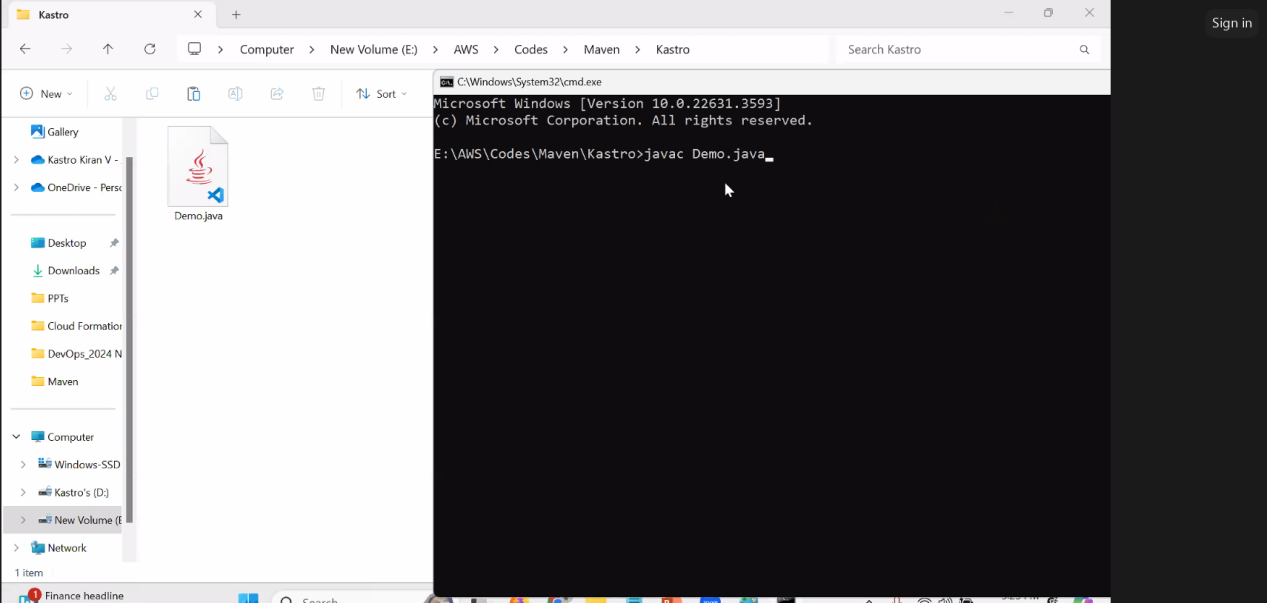


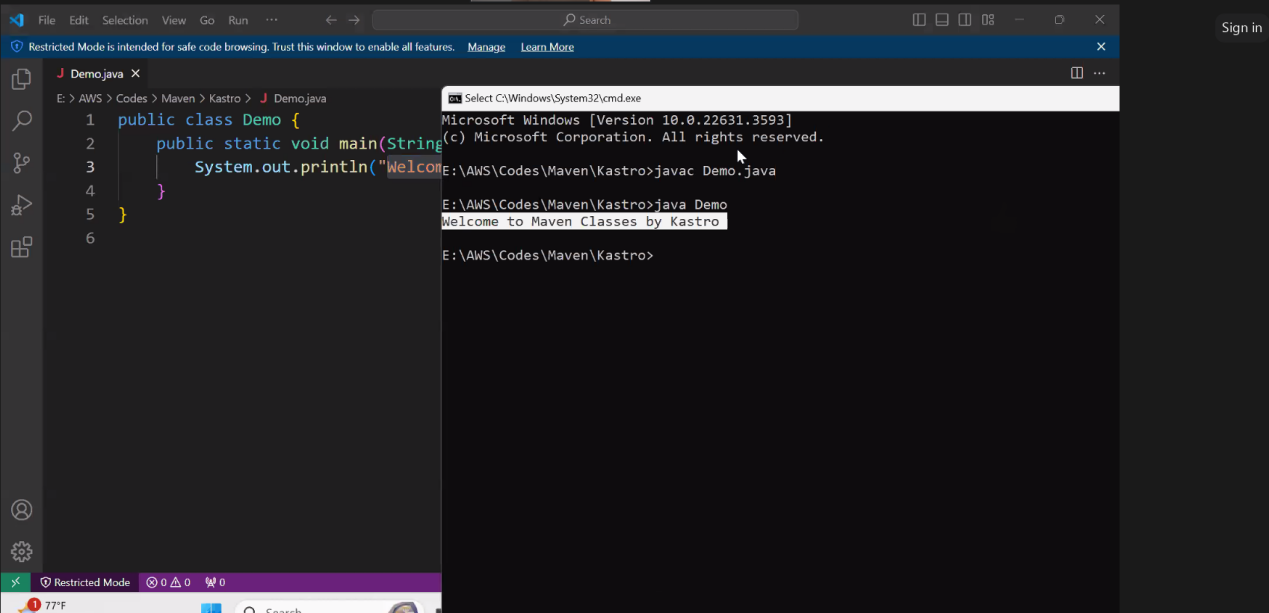




Compile file

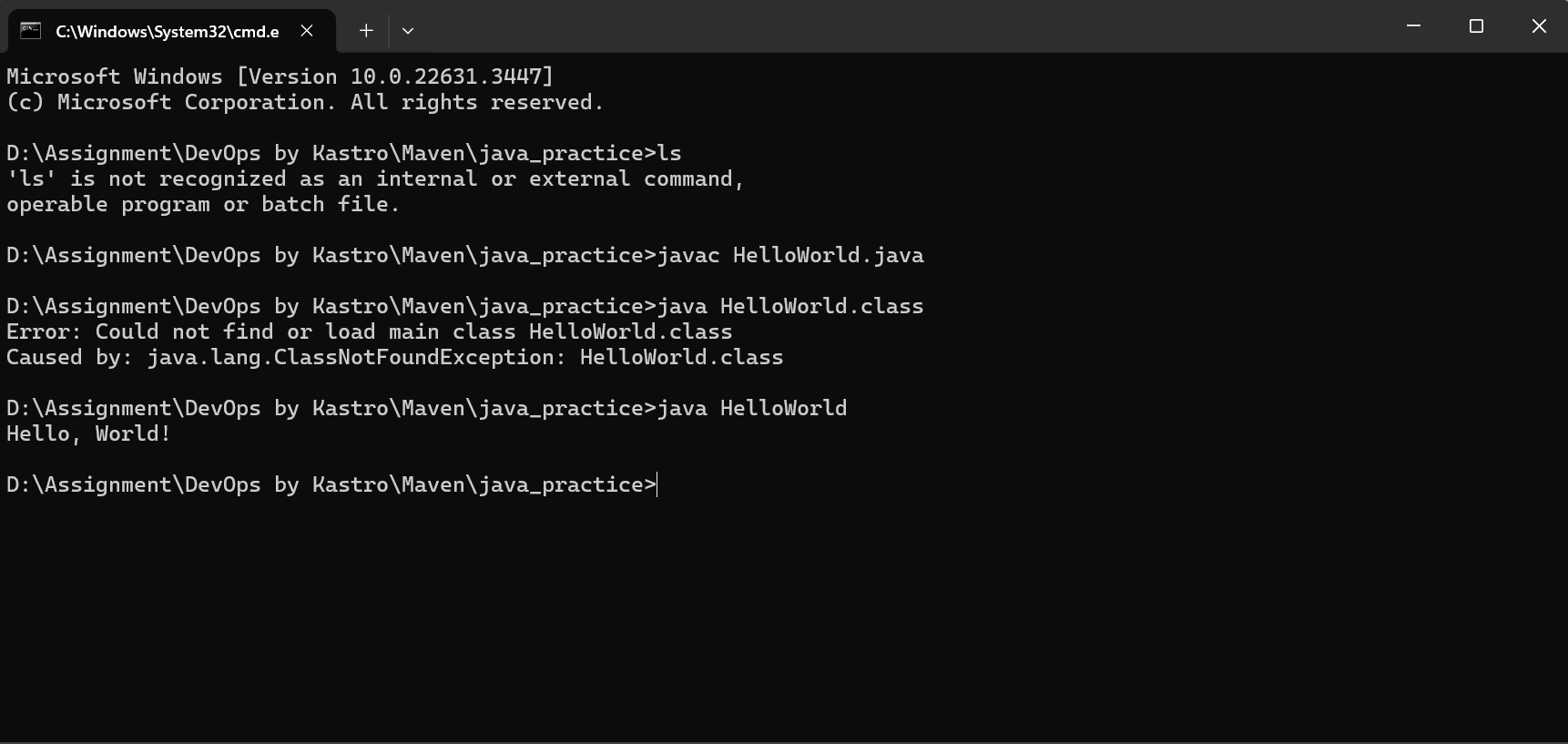
Compile of code change .java file to .class file and change humans readeble language to Binary(Bytes)





**1.First we have to compile the file with** $ **javac filename.java**

**2. To run the file $ java filename(Without extention)**



Compile of code create .class file from .javafile and change humans readable language to Binary(Bytes)

############################**COMPLETE FILE COMPILE** ################################

=========================================

MAVEN

=========================================

WITH THIS COMMAND USE WE WILL CREATE WORKING FILE STRUCTURE

Maven Terminology: -------------------------------

Ex: IBM ---> Project (Hotstar) ---> Java ---> Code ---> hotstar.java (Source Code) ---> Compilation (Java Compiler) ---> hotstar.class ---> JVM ---> Output

**1.Archetype:**

It represents what type of project we are going to create. It refers to a project template that you can use to quickly set up new projects. It essentially acts as a blueprint containing a basic structure and configuration for a specific type of project.

1. maven-archetype-quickstart ---> It represents the standalone application creation.

QUICKSTART = Standalone app

1. maven-archetype-webapp ---> It represents the web application creation. There are 100+ types of architypes

WEBAPP = Webapplication

Interview qution: How you idenfify application

Ans : Quickstart = Standalone app

WEBAPP = Webapplication

1. **Group ID**: It represents the company name or a project name.

Ex: IBM-Hotstar

**3.Artifact ID**: It represents the project name or project module name. (WE CAN GIVE BOTH FOR BETTER UNDERSTAND)

Hotstar (movies, cricket,...) ---> Movies, Cricket...

DevOps ---> Introduction to DevOps, Linux, Maven…

-An artifact refers to any file that can be downloaded, installed, or deployed during the build process. It's essentially a piece of code or resource that your project either uses or produces.

PhonePe ---> Transactions, Recharge, Tickets, Bills...

Ex: PhonePe(Group name) - Transactions(Module name)

**4. Packaging:** It represents how you want to package your application i.e either war file or jar file.

-Standalone App ---> Jar file (multiple .class files will be available)

-Web App ---> War file (multiple .class files will be available)

.java file ----> .class file ----> Execute (JVM) ----> Output

1000 .java files ----> Package (jar or war) ----> Execute (JVM) ----> Output(Example will be show in TOMCAT lecture)

WITH THIS FILE STRUCTURE WILL AUTOMATICALLY GENRATE

#############How to create a standalone application using Maven############

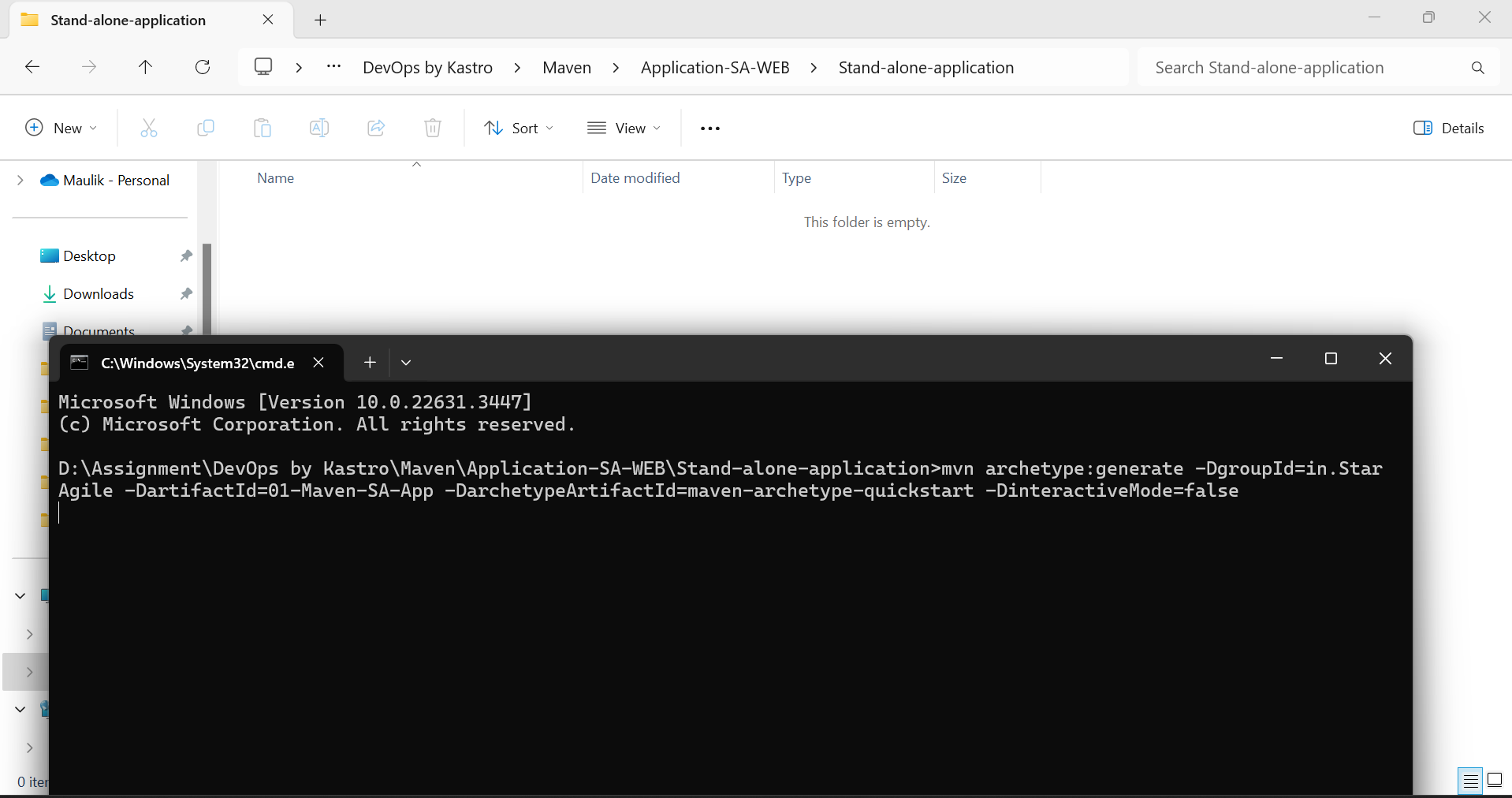
mvn archetype:generate -DgroupId=in.StarAgile -DartifactId=01-Maven-SA-App - DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false

This syntex provide by devloper

DevOps no need to remember

DinteractiveMode=false means DO NOT BOTHER ABOTE ALL MENTIONED INFO IS AVAILABE, JUST CREATE

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*STANDALONE APPLICATION\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*



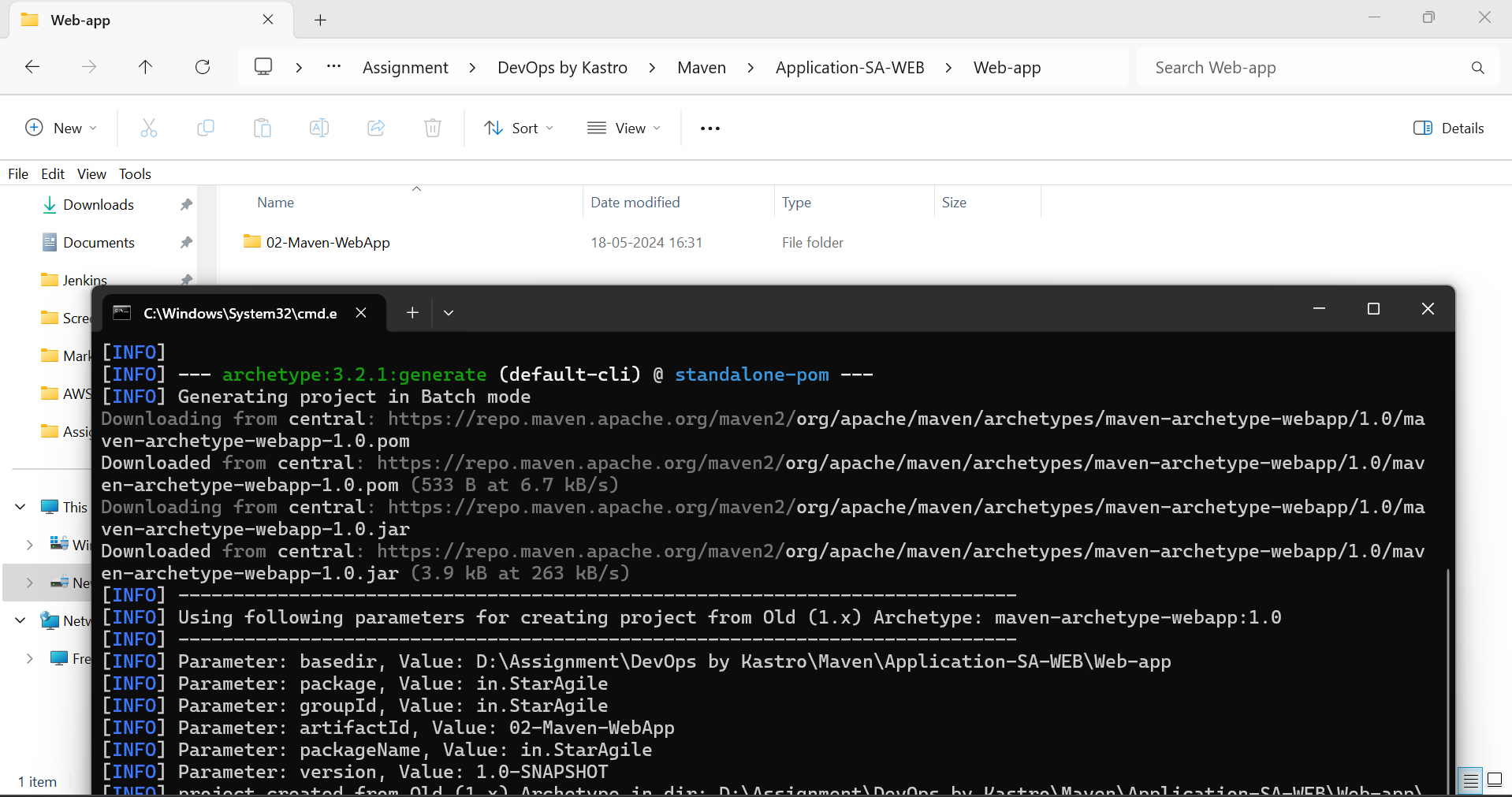
#############How to create a web app using Maven############

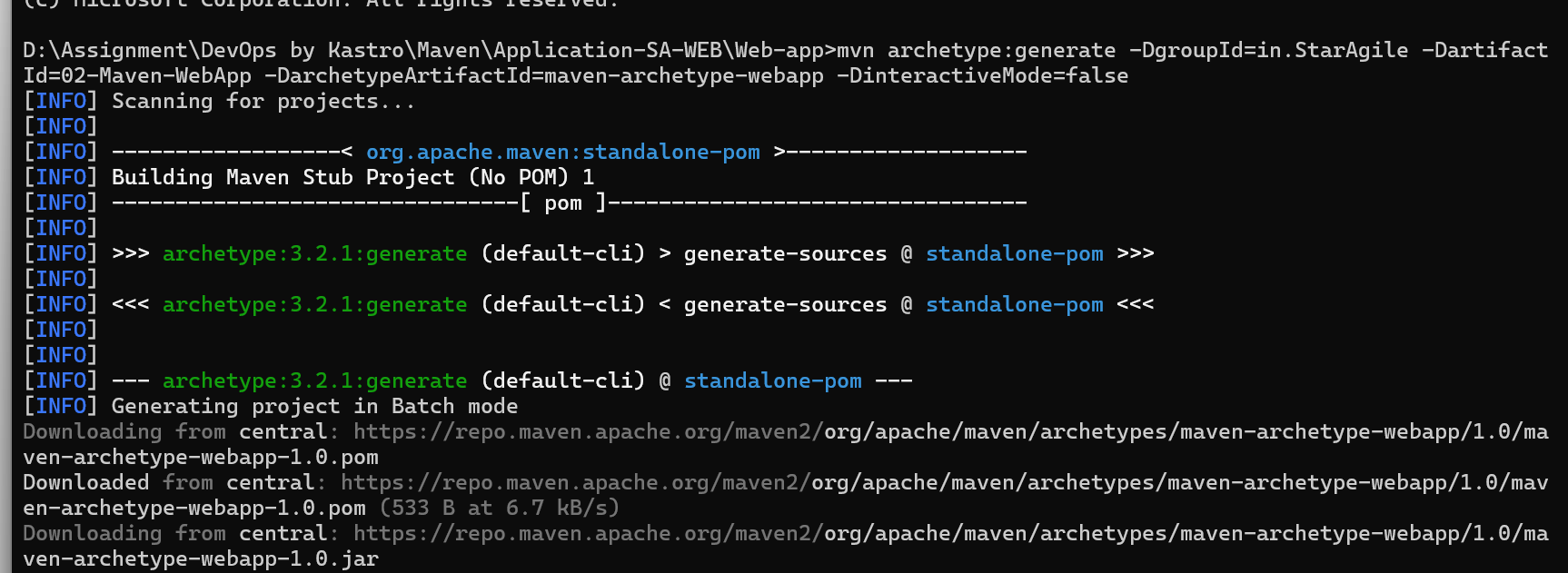
mvn archetype:generate -DarchetypeArtifactId=maven-archetype-webapp -DgroupId=in.StarAgile - DartifactId=02-Maven-WebApp -DinteractiveMode=false

This syntex provide by devloper

DevOps no need to remember

DinteractiveMode=false means DO NOT BOTHER ABOTE ALL MENTIONED INFO IS AVAILABE, JUST CREATE





**How to create initial project folder structure for standalone application(COMPLETED UPPER STEPS)**

D Drive ---> Create a folder (01-Standalone-App) ---> Open Command Prompt in the created folder path ---> Execute the below command

mvn archetype:generate -DgroupId=in.StarAgile -DartifactId=01-Maven-App - DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false

=> Once the project folder structure is created, we will see two things

1. SRC Folder
   1. Main Folder - java\in\StarAgile\App.java ---> In ".java" file app source code will be written.
   2. Test Folder - java\in\StarAgile\AppTest.java ---> In "AppTest.java" file, code testing (J-Unit Test) related info.
2. pom.xml File Without pom.xml file maven will not work. Whatever work we are going to do in maven, everything will be communicated using pom.xml file only.

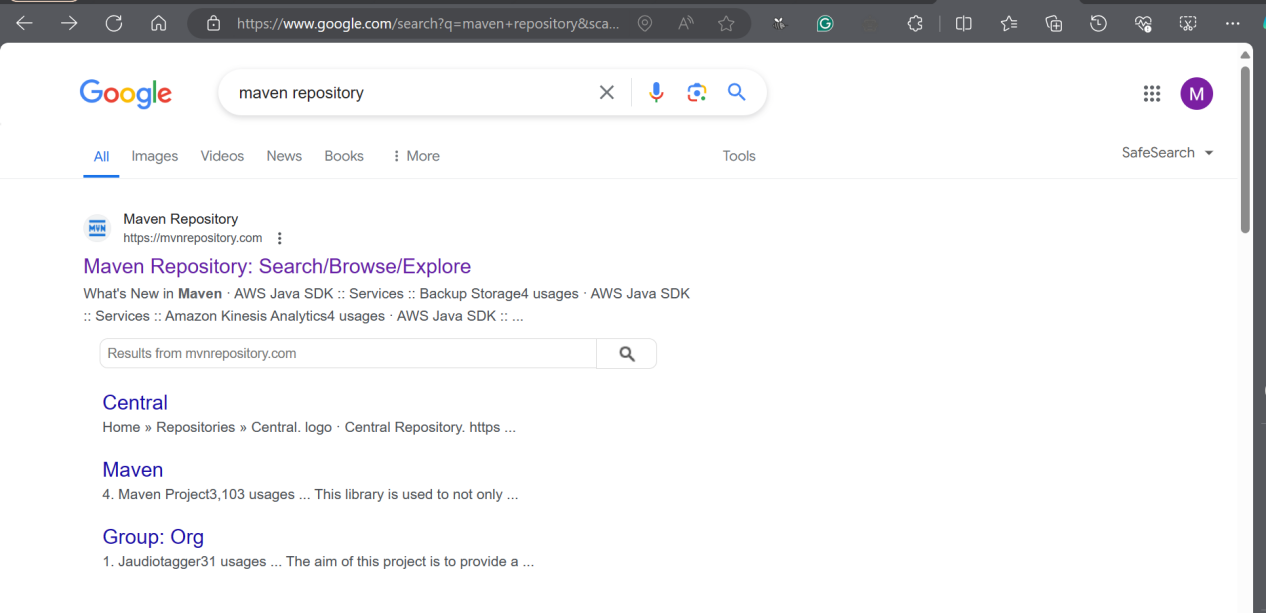
**‘Note: Whenever we are executing maven goals we should execute all the maven goals in the pom.xml file path only.’**

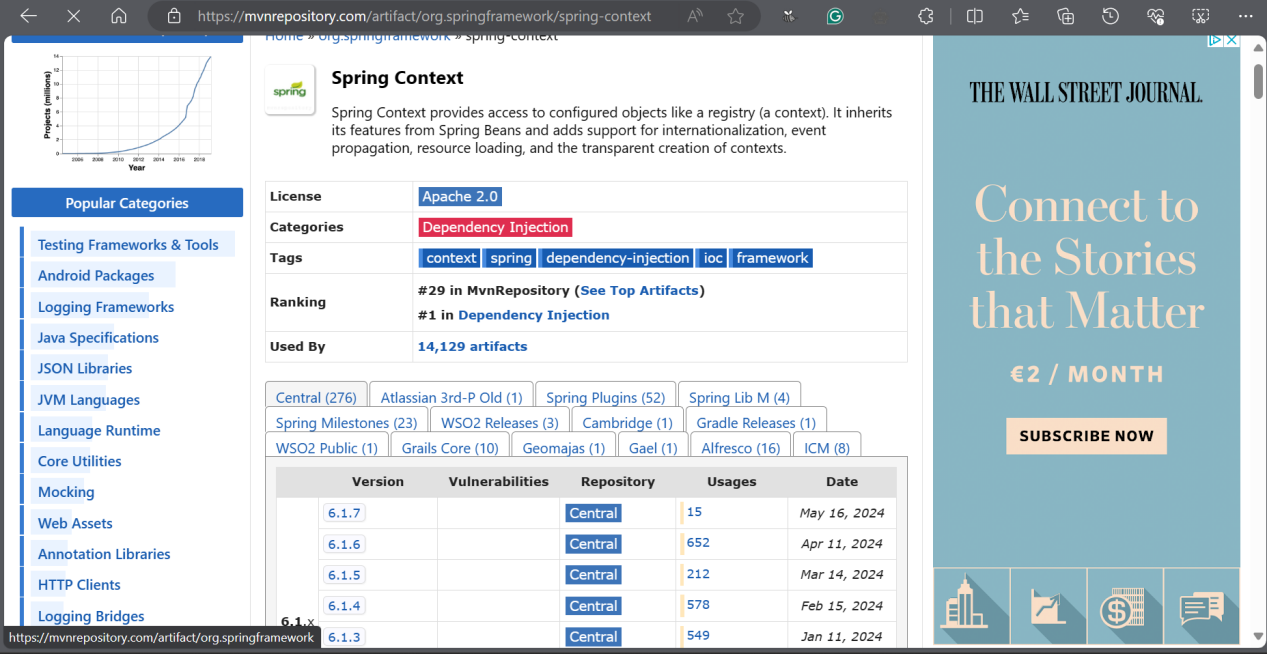
**How to add dependencies in the pom.xml file**

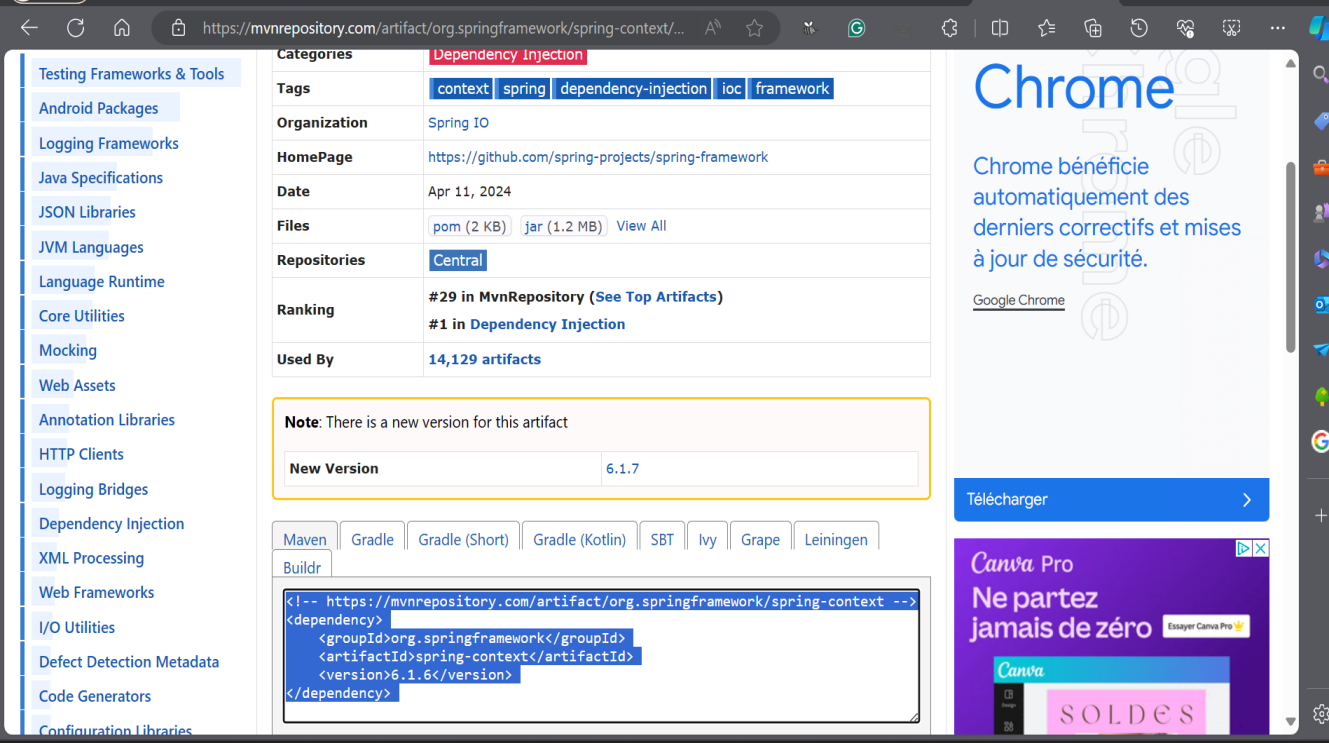
**Normally Devloper provide pom.xml with all dependencies if there is a missing then DevOps engineer add**

**\*Why we add dependencies?  
-**Dependencies are added to a Maven project to specify external libraries or components that the project needs in order to compile, run, or test correctly.

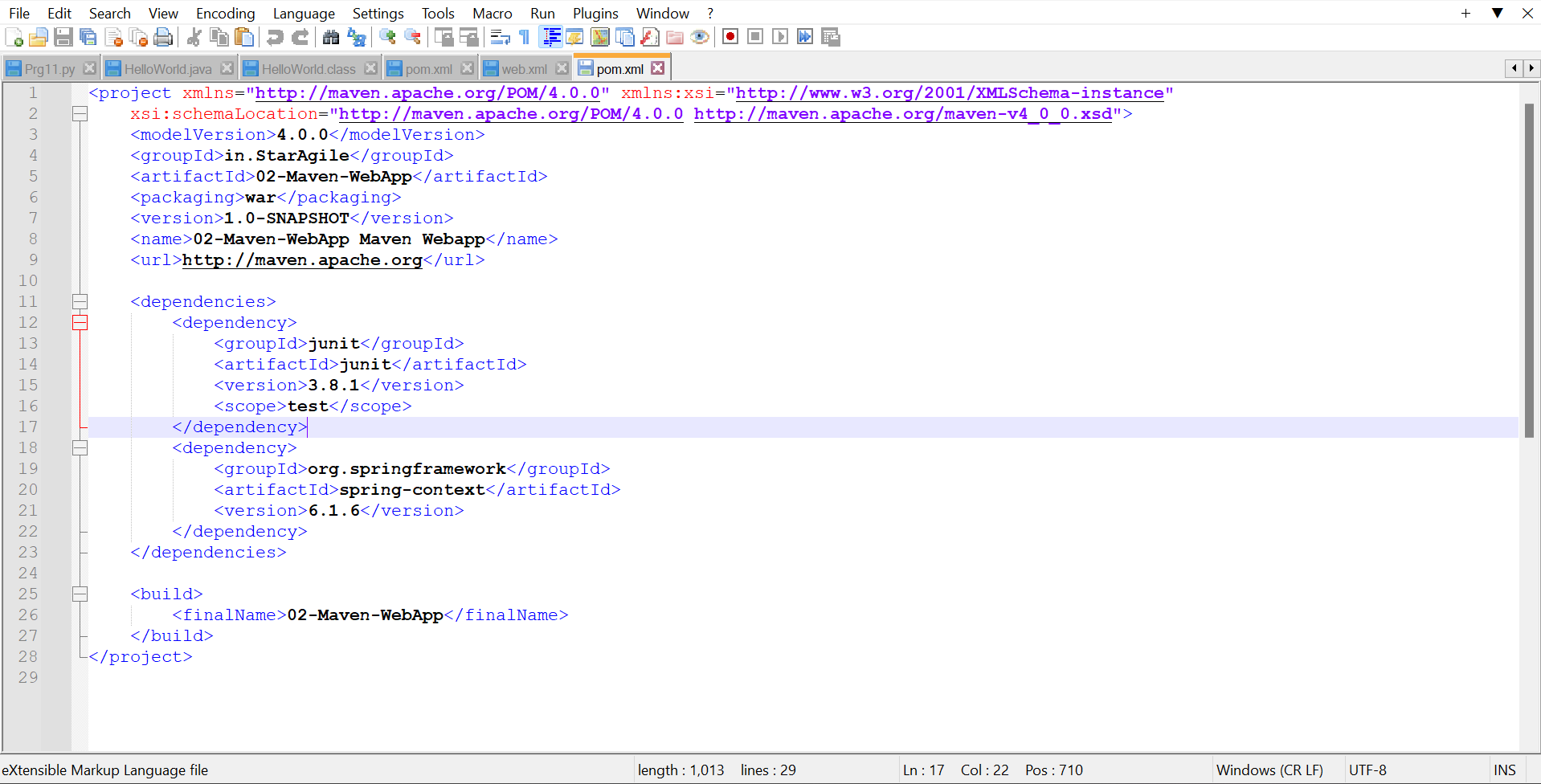
**-With only java run time environment app will not work that’s why we add dependencies.**







Copy Dependencies and past at pom.xml



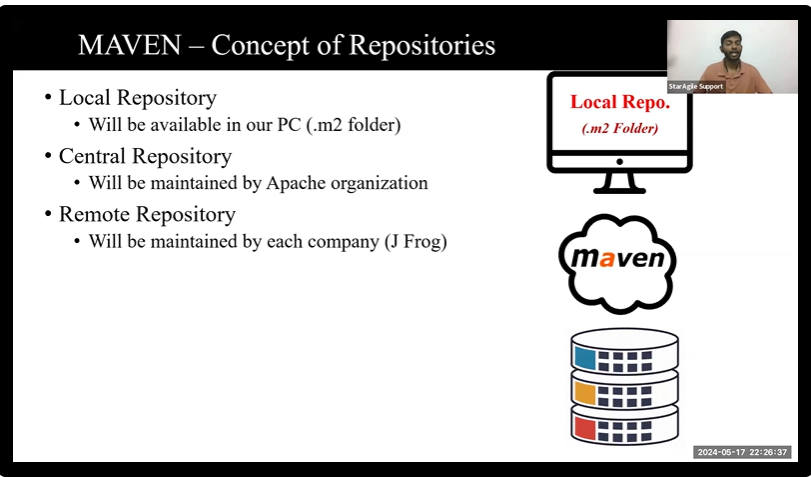
“That’s how we can add dependencies to the project structure pom.xml file”

#######################MAVEN REPOSITORIES########################

=> Maven will download any dependency using a concept called repository.

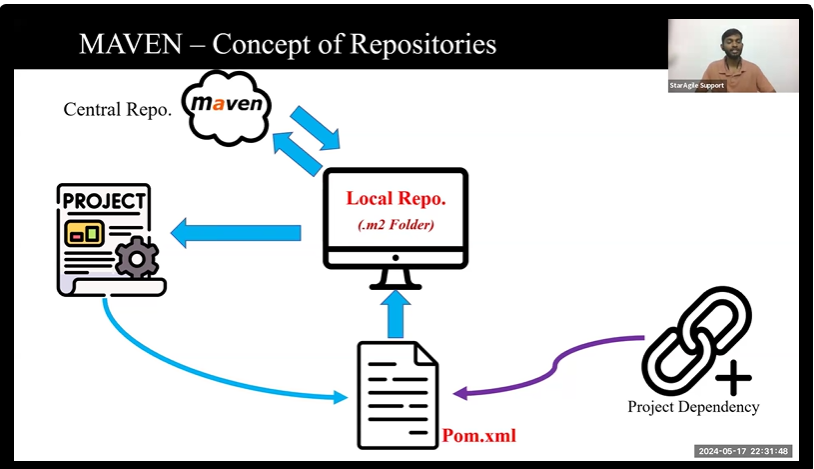
=> In Maven, we have 3 types of repositories:

1. Central Repo. - Apache Organisation ([Maven Repository: Search/Browse/Explore (mvnrepository.com)](https://mvnrepository.com/))
2. Remote Repo. - Individual company (JFROG)(OWN REPO)
3. Local Repo. - Our PC (.m2 folderin C:\Users\mauli)



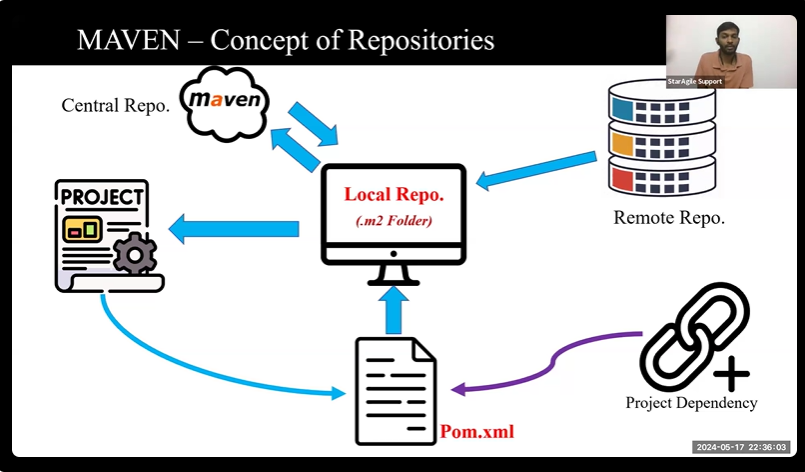
Concept of back-end work of adding dependency:

First it will check in local repo, if not available it will go to centrel repo(mvn reoo)



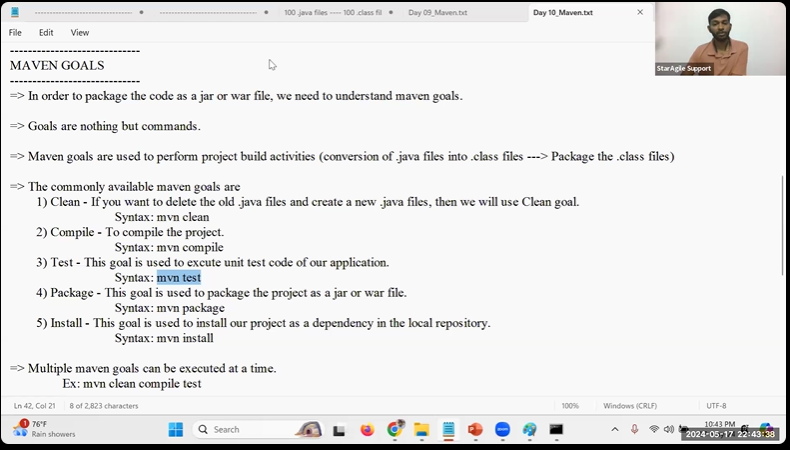
\***WITH REMOTE REPO(Which is hold by company by company)**

**That secure projects to affect from getting Vulnerabilities(bug).**



NEXT LECTURE

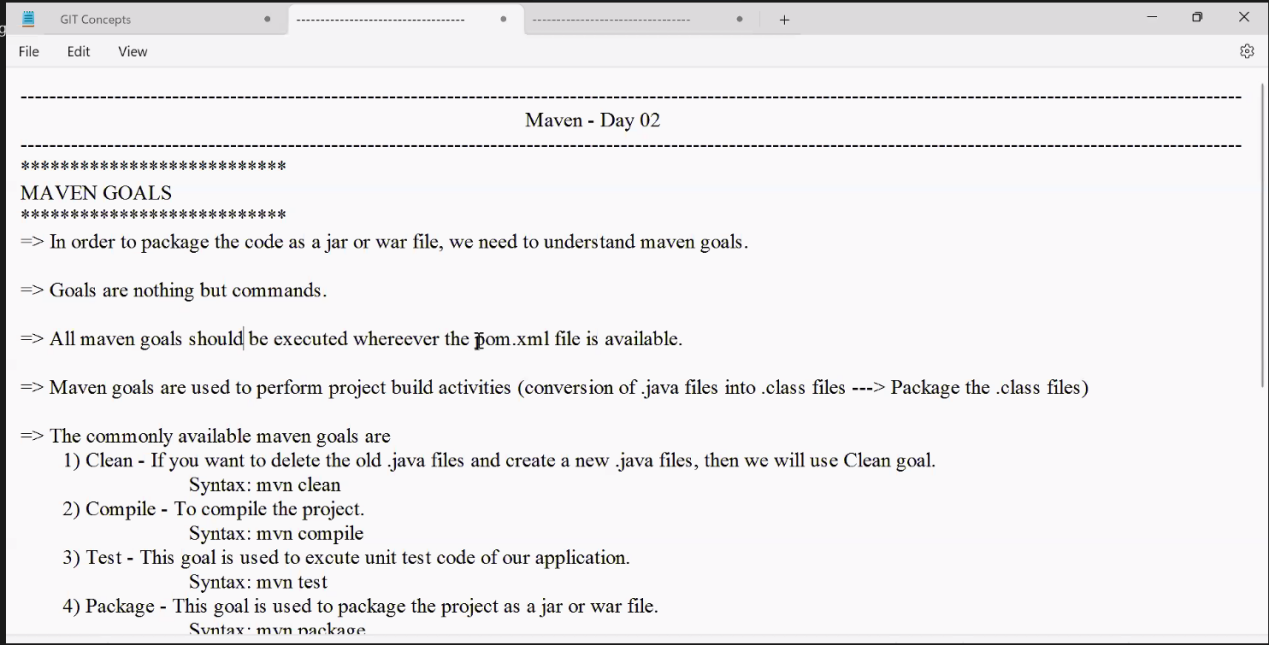
REST

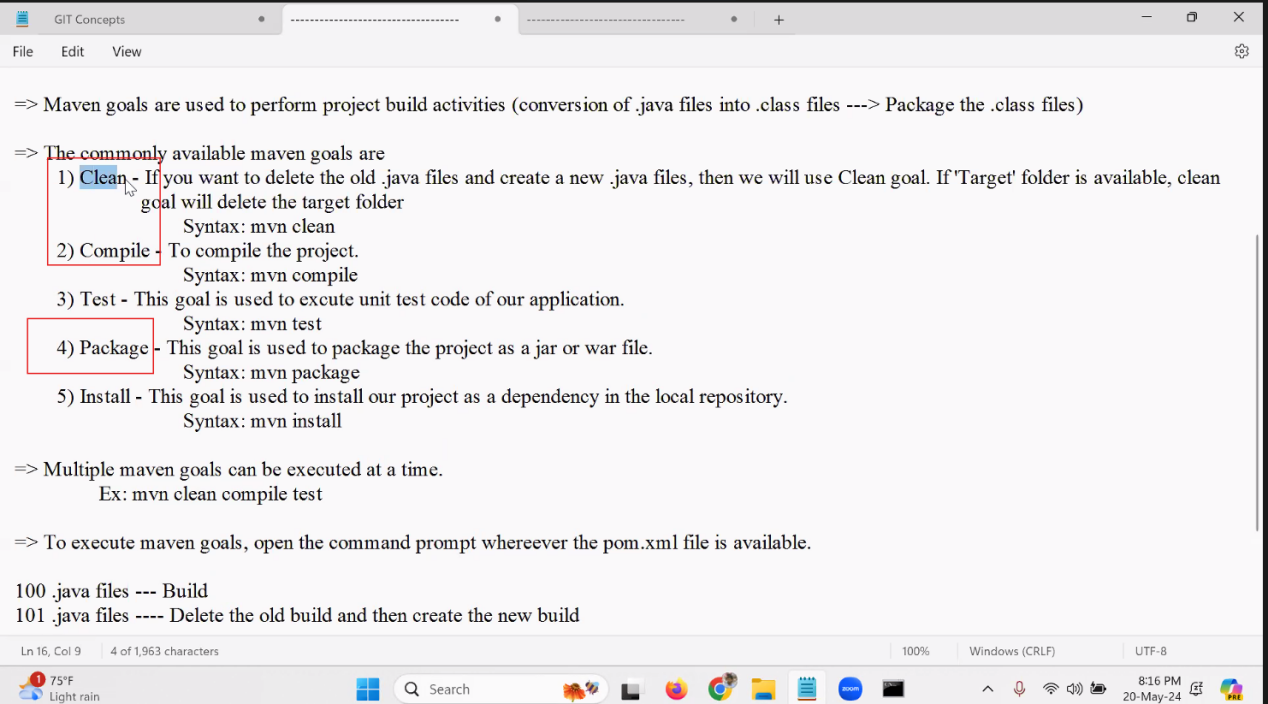


20-05-2024

MAVEN GOALS is called for maven actually its command:

LIST OF COMMAND:





DevOps engineer work with 3 command

Goal perform one by one like

You can not pacakge the code without compile it

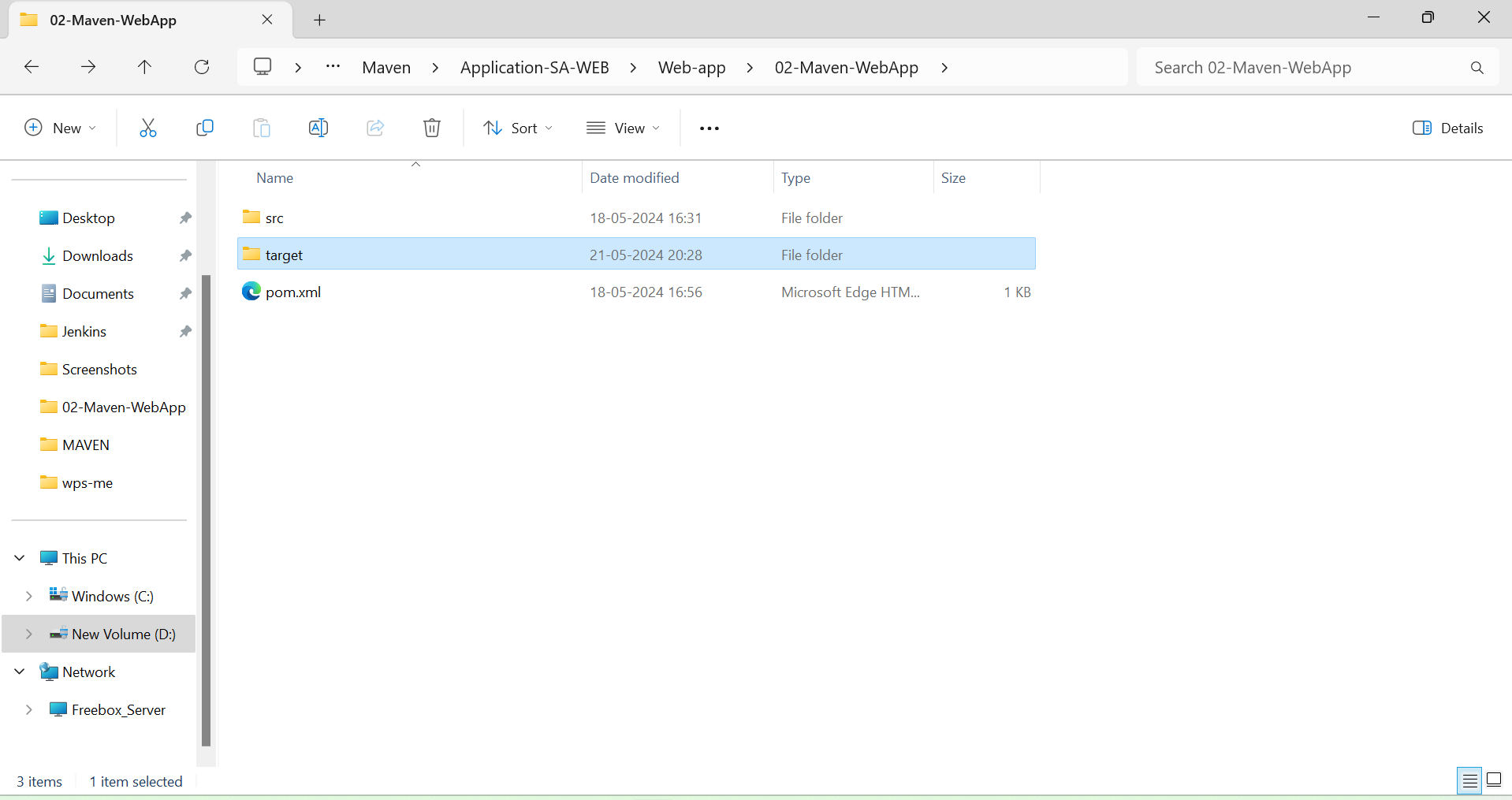
**mvn clean** goal clear all **.java** file from filr strucutre include **target** folder

**mvn compile** goal will create **target** folder in the same file strucure

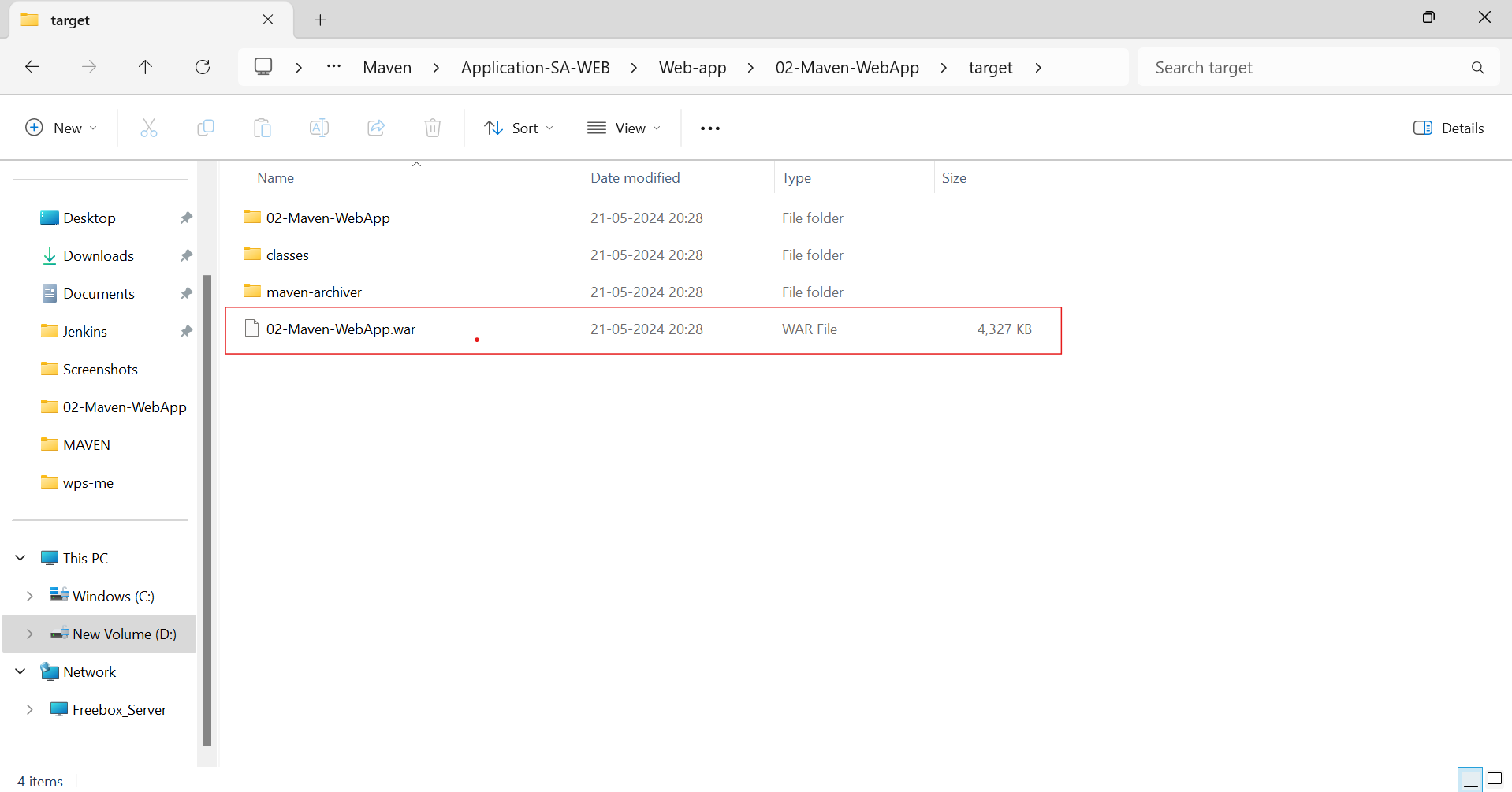
**mvn package** goal, package the all .class file and pacage it, also create .war or .jar file that will be uploaded in the main server

Output from:

mvn package(Directly used with out following complie goal)



Box is for main package file, this uploade in server



MAVEN FINISHED HERE FOR ME:

Used for compile and package the code

**MAVEN FILE SRUCTURE**

**: Maven uses a standard directory structure.**

**Main have main source code and test have after testing code that’s how it will seprate from each other**

SRC - **MAIN**

Briefly, **src** holds your project's source code and resources, with

**src/main** containing the core code **(src/main/java**) and non-code resources (**src/main/resources**) for your application.

SRC - **TEST**

In Maven, the **src/test** directory is dedicated to test code. It usually mirrors the structure of src/main but holds unit tests for your main code. This separation keeps your main application code clean and separates testing concerns.

TARGET:

In Maven, the target directory is where the build process stores compiled class files, packaged applications (JARs, WARs, etc.), and other temporary files generated during the build. It's generally not part of your source code and isn't included when you share your project with others.