**Maven:**

1. Why Maven?

Can Maintain proper structure

Can manage dependencies on our behalf

Can build the code for us

Can create documentation sites

**Terminologies**

**GroupID**: Generally it will be company name

**ArtifactID**: Project Name

**Version**: Version Number

**Example**: Google company have many products(youtube, hangout, translate) then name will be like below

com.google.youtube.110

Here com.google -> GroupID(company name)

Youtube -> ArtifactID(Project Name)

110-> Version

**Archetypes** ->Means template..based on the need(if ur automating webapp, then select particular template, then it will provide all project cutomized template relate to webapp]

**Dependencies** ->jar files [eg selenium jar, java jar, library]if we need to interact with webappl thru java code we need selenium jar & Java related jar.

**Plugins ->** similar like dependencies, but different..like some action Jar will not handle where Plugins can handle it

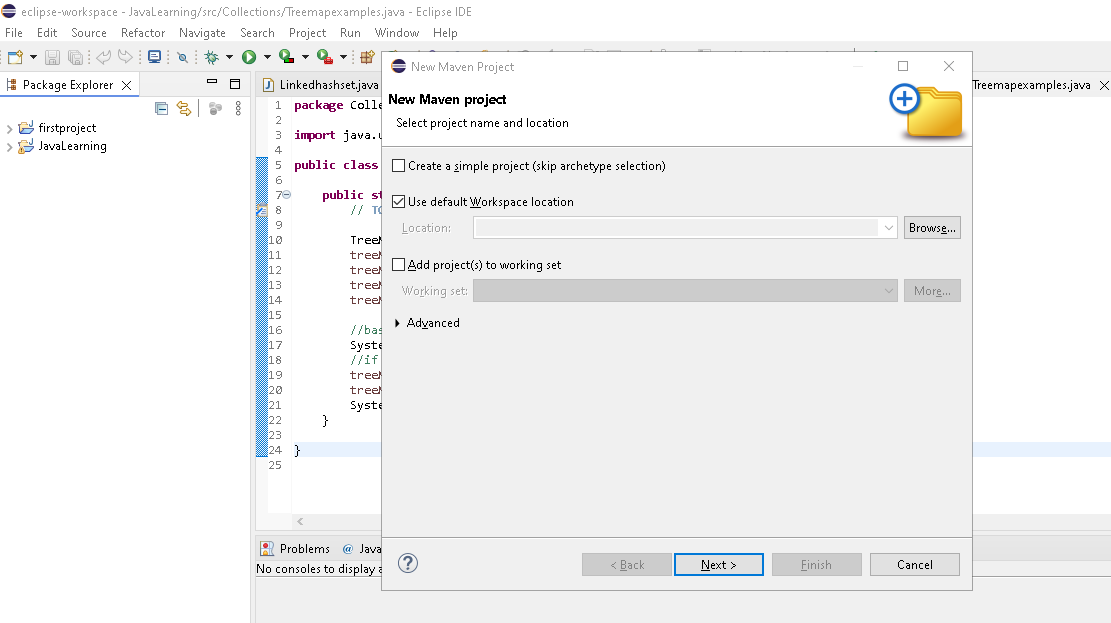
**Lifecycle->**

**Goals->**

**Maven central repository(MVN repository**)->its a website contain all java related Jar files stored in one place

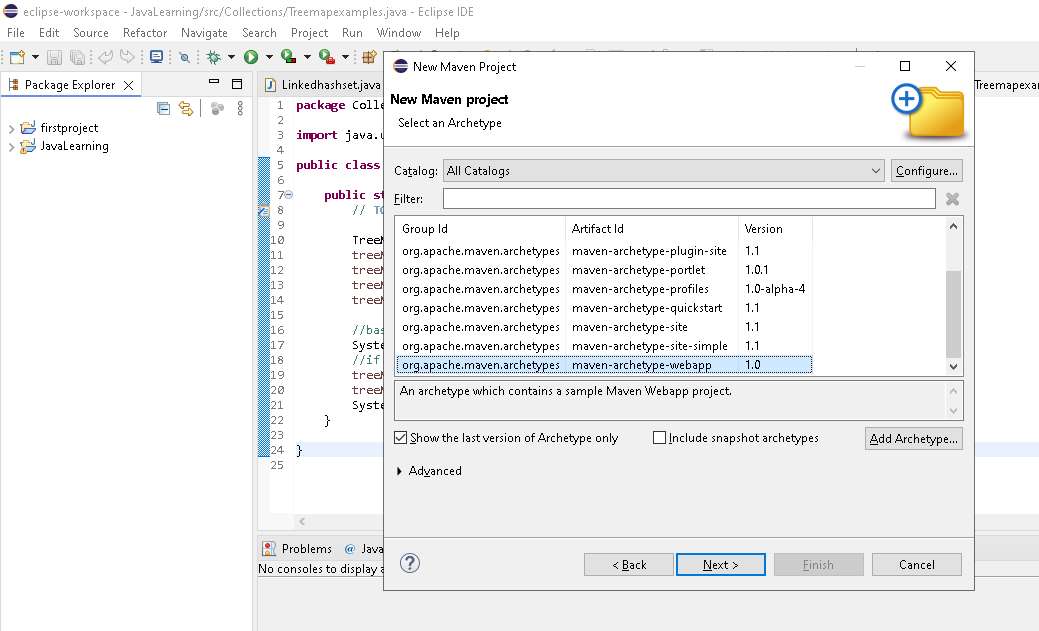
**How to create a Maven project in ellipse?**

File -> New ->Maven project



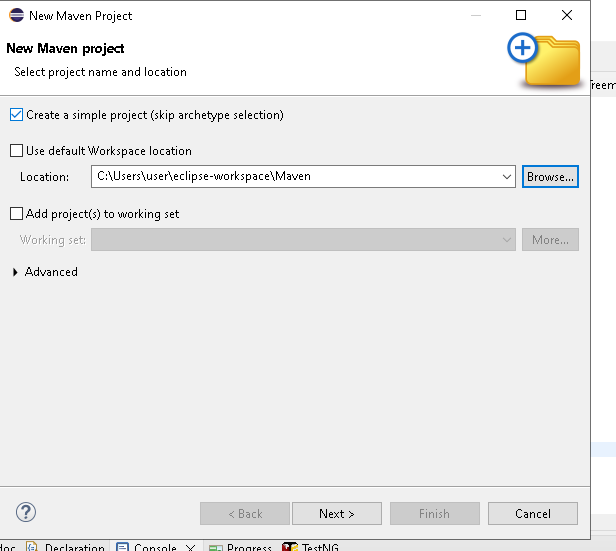
Click next, it will show different Archetypes (template)

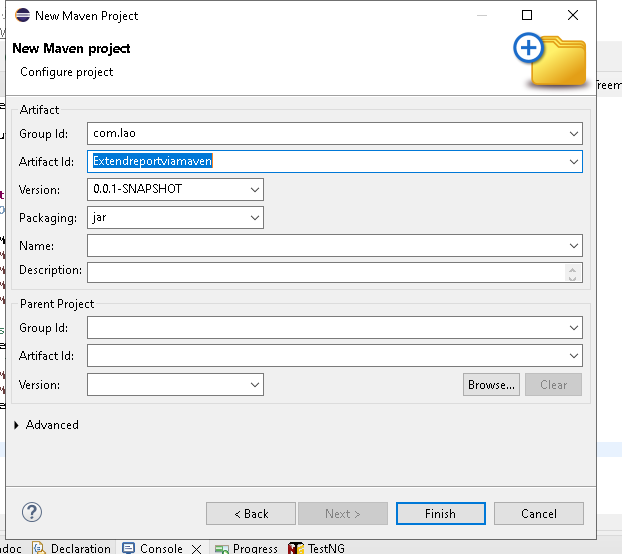
**Archetypes** ->Means template..based on the need(if ur automating webapp, then select particular template, then it will provide all project cutomized template relate to webapp]



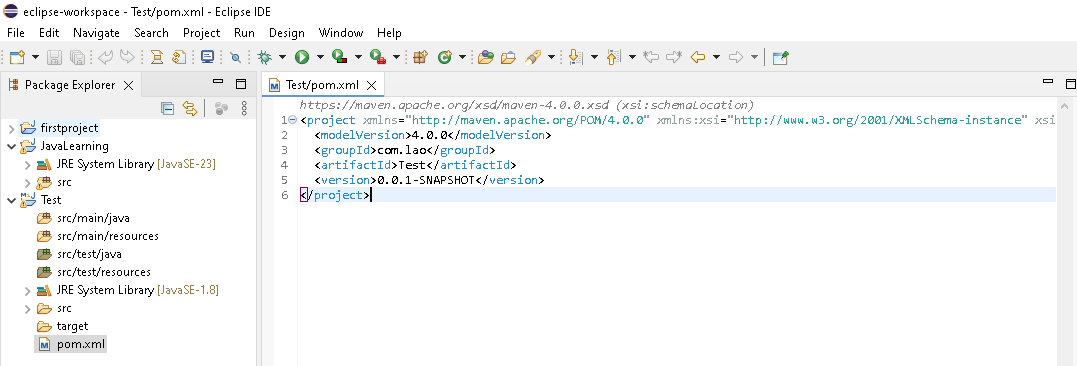
Click cancel & create simple maven project

Location – browse & give any local folder





Artifact ID name changed to Test & saved

c

If you create normal java project, then u can see only src & JRE System library

1.In Maven project, u can see structure folder as above **[ Maintain proper structure]**

Src/main/java ->all the actual codes[Eg calculator add/sub all function code] should be placed/keep in that folder

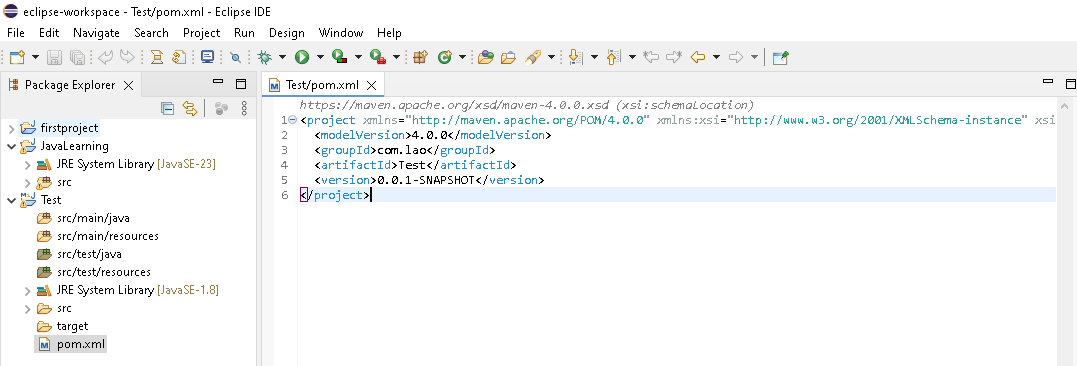
Src/main/resources ->apart from actual code, Additional resources wil be placed here[Eg Calculatore image, color, size]

Src/test/java-.>For unit testing codes, developer will keep for their testing

Src/main/resources ->Resources related to unit testing codes

**2.Can manage dependencies on our behalf**

Porn.xml ->Project object model -

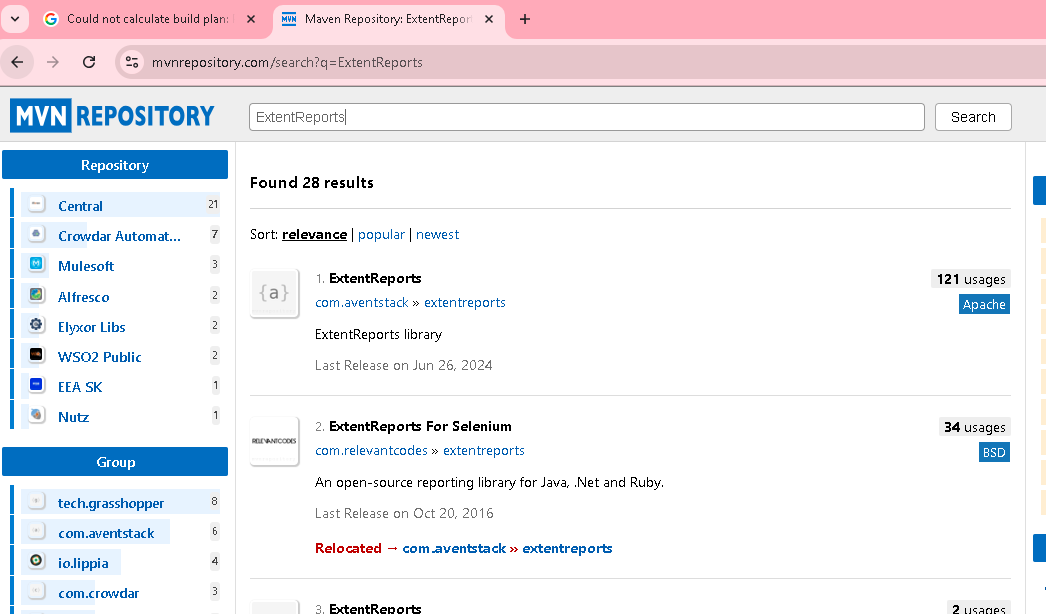


In Maven – just we have to mention the dependency(Source Name & jar version eg selenium & related jar)..on behalf of us, it will download from mvn repository & integrate in ellipse

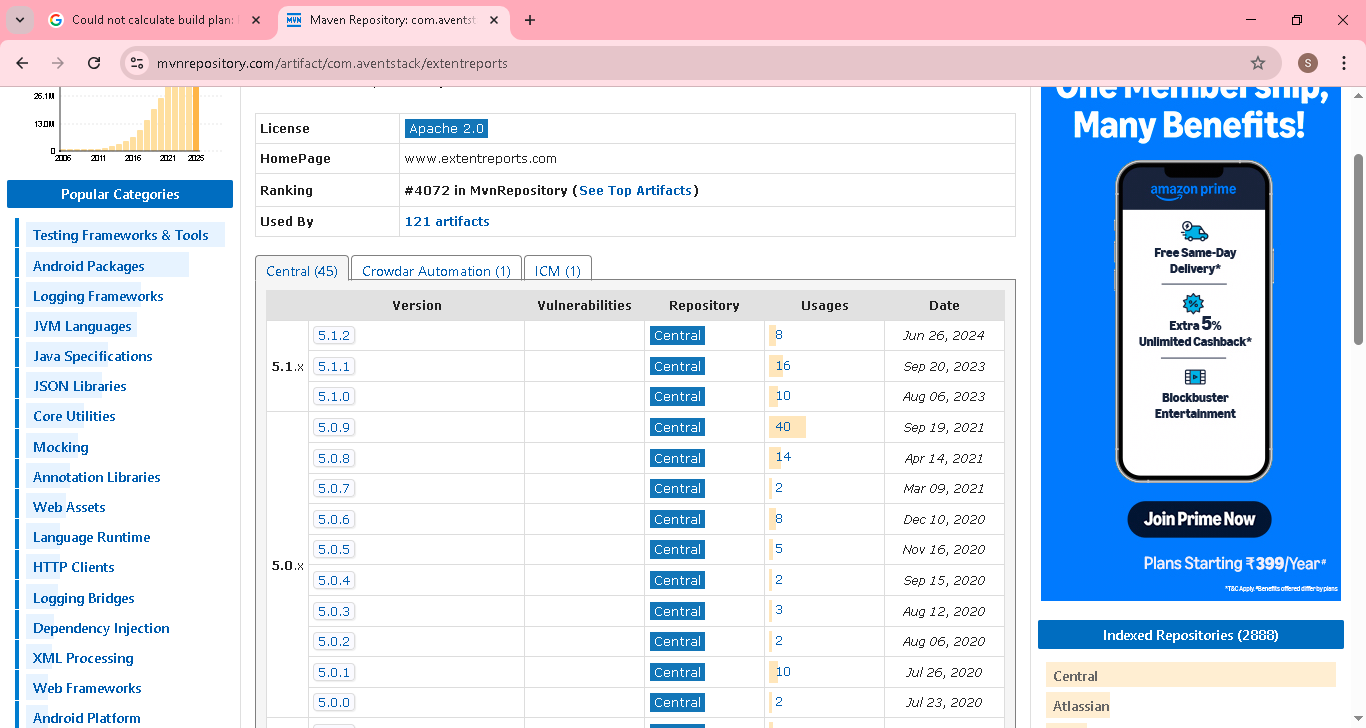
Here[ExtentReports jar we are downloading]

Go to MVN repository(<https://mvnrepository.com/>)

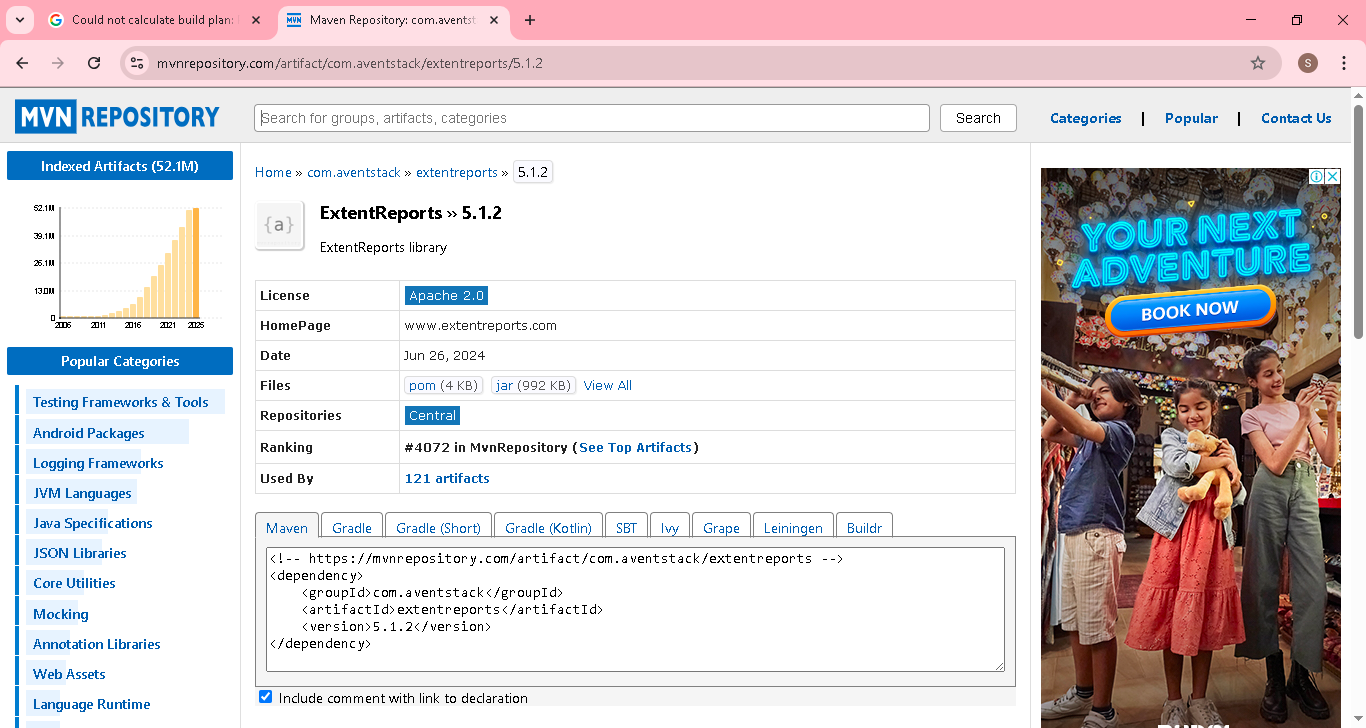
& search ExtentReports



Click the latest version 5.1.2

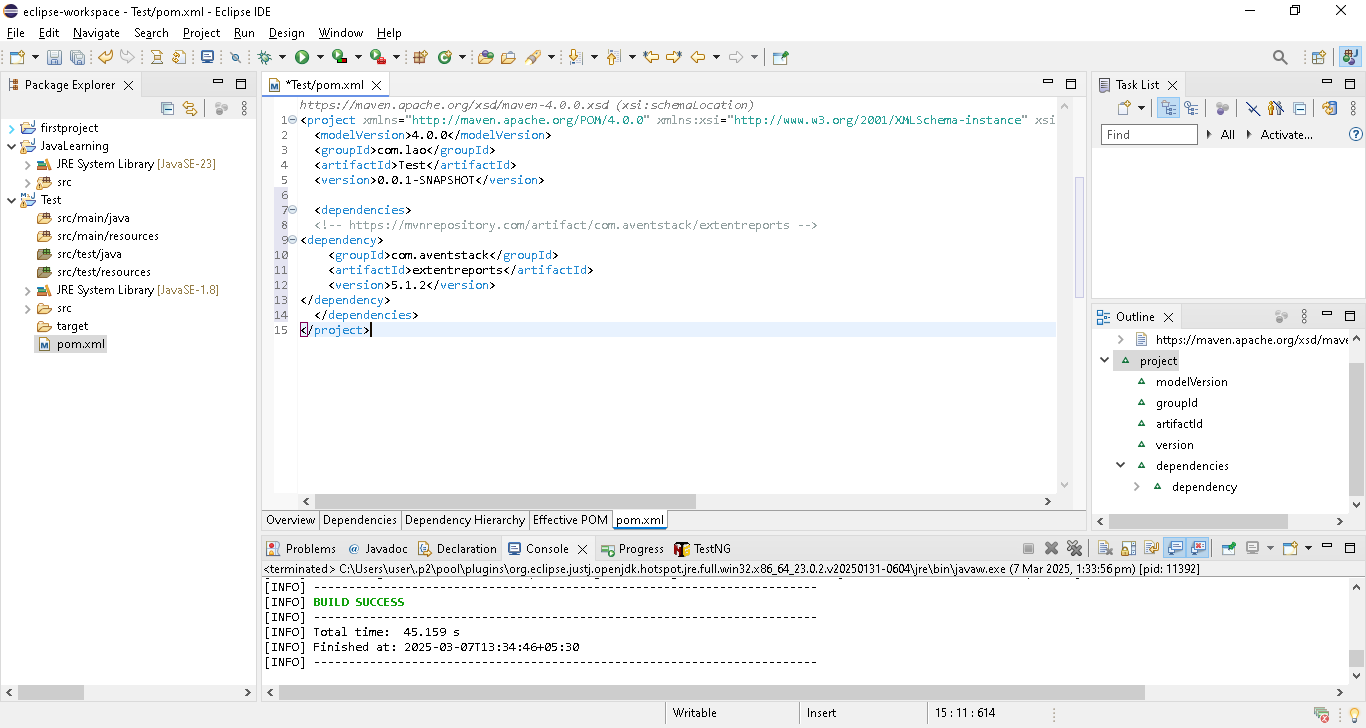


Copy the maven code



Come back to porn.xml

Paste the code inbetween the tag <dependencies> & </dependencies> & just save

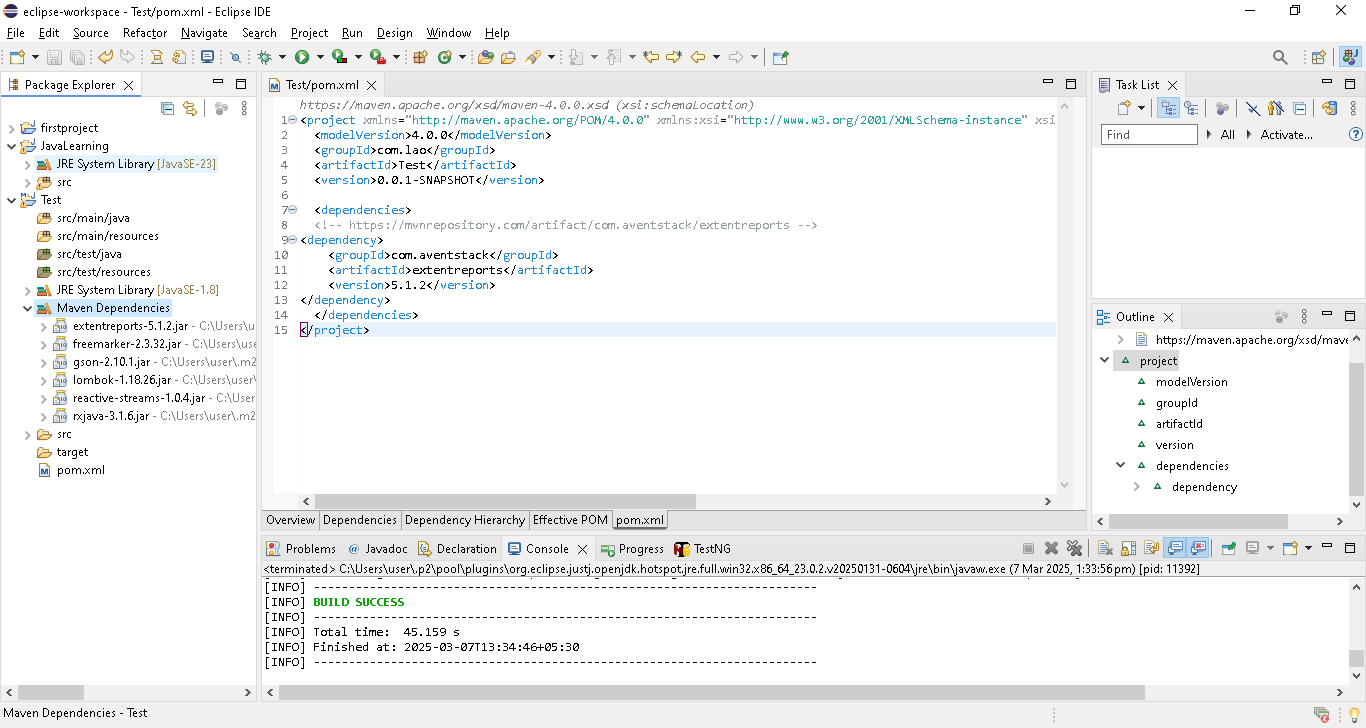


Check the folder structure after save

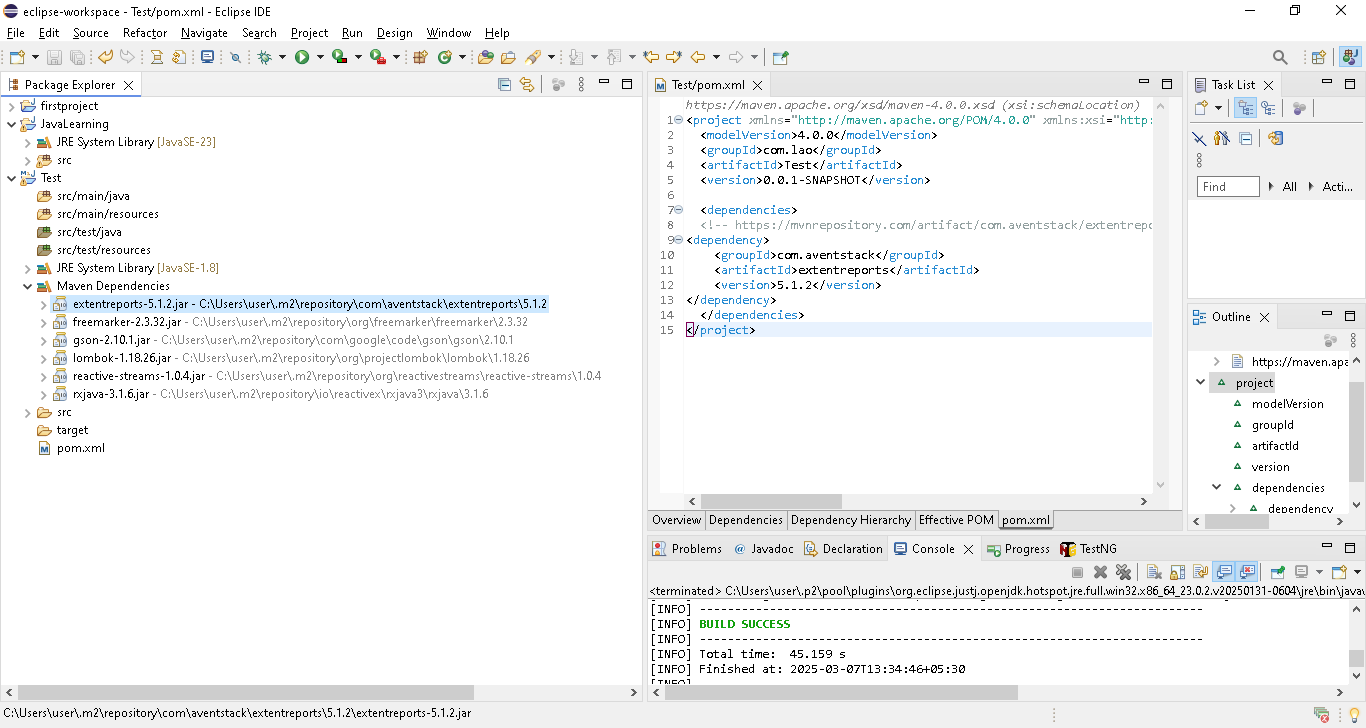
Maven dependencies’ folder created & contains all jar files related extendsreport

Note: we downloaded extendsreport jar only. Extendsreport depends on other jar like freemaker,gson jar etc..so maven will download dependcy also on behalf of us

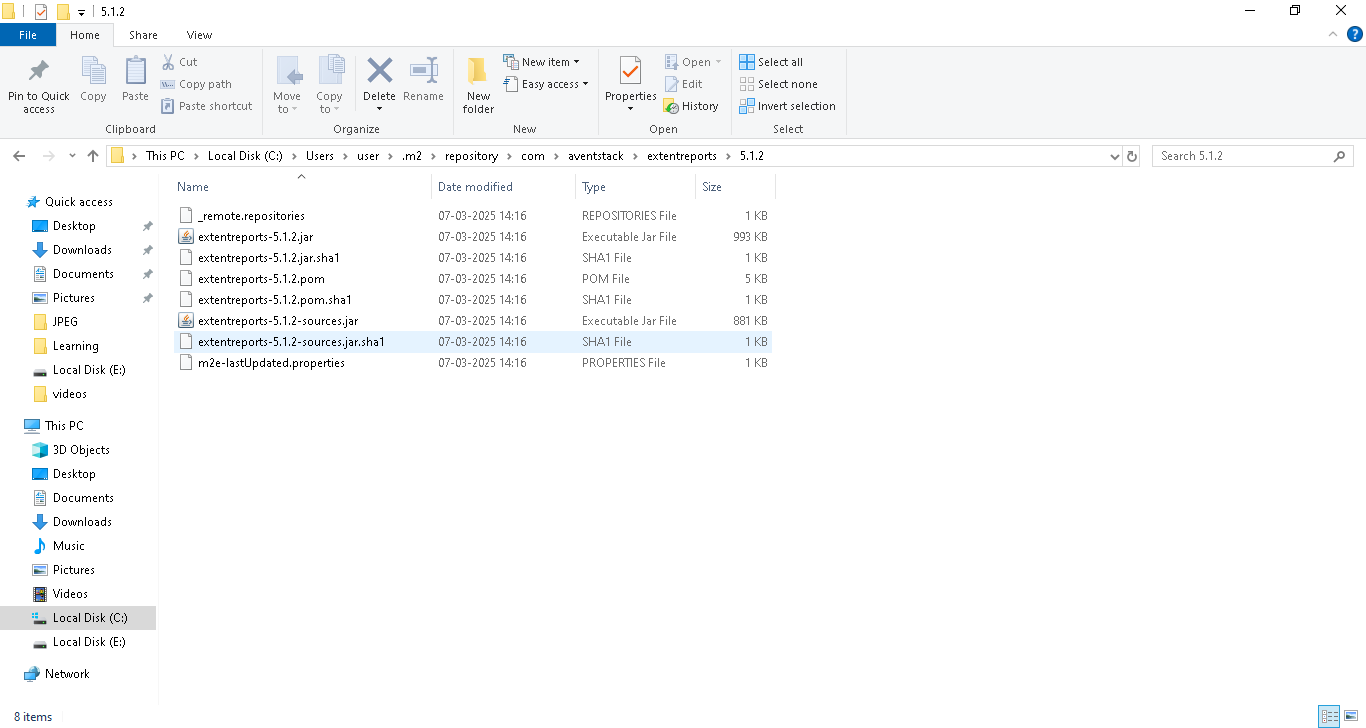
If we download using java project, then we have download one by one based on error..we download next jar



We need internet connection then only maven will download from mvn repository site.



In ur computer local also, downloaded jar will be present in below path



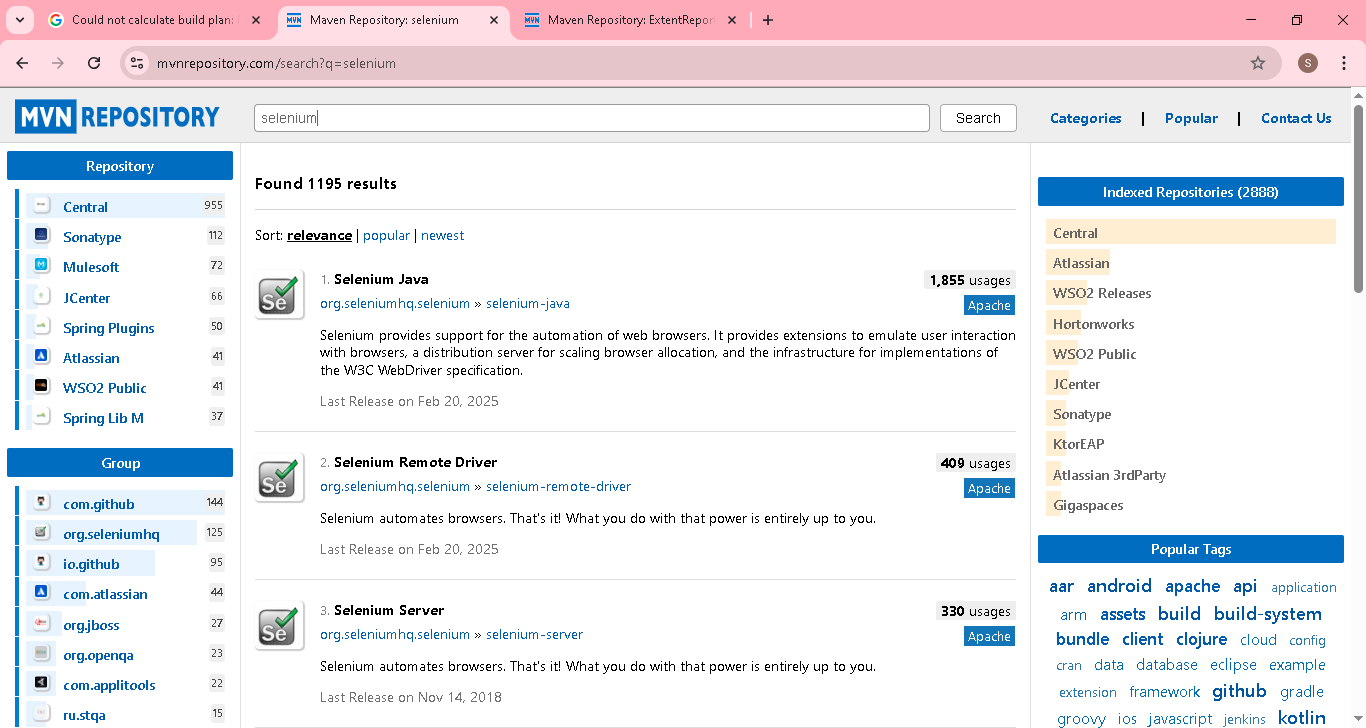
Then if we download another jar example(selenium java) using dependencies

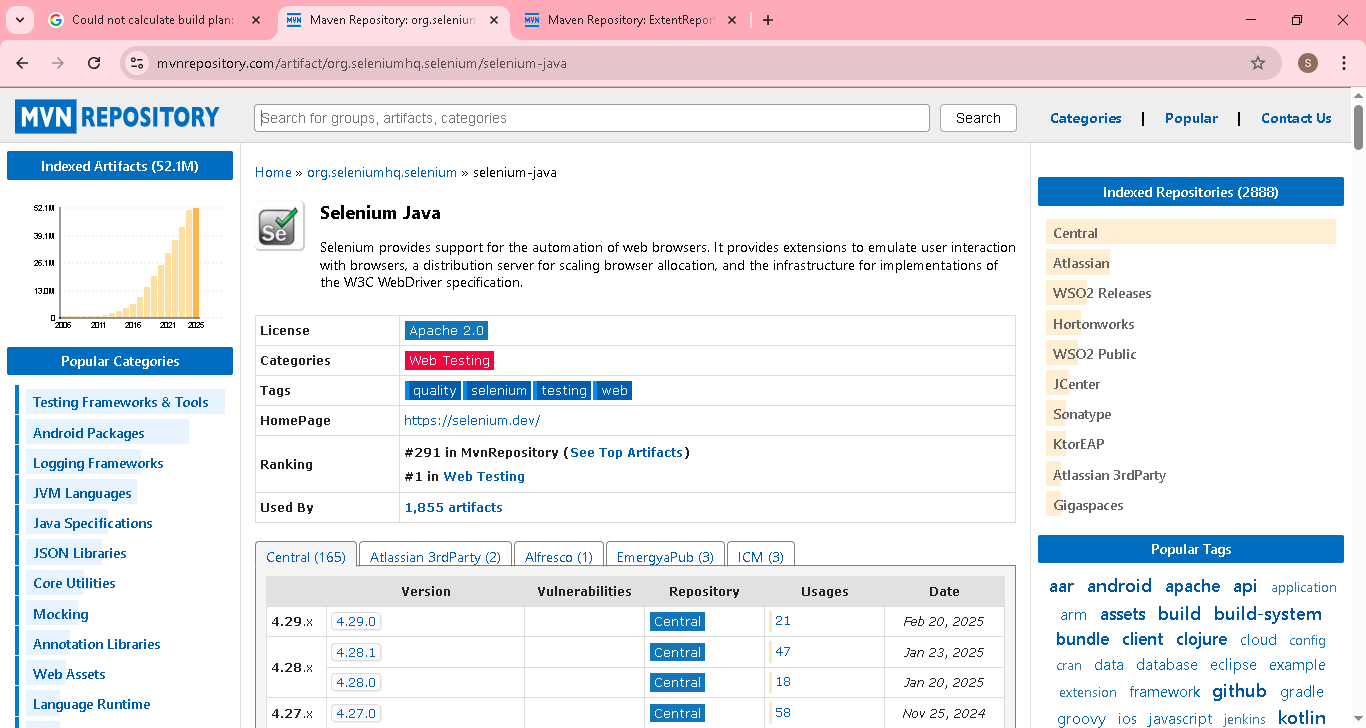
Add new dependencies in pornxml to download selenium java

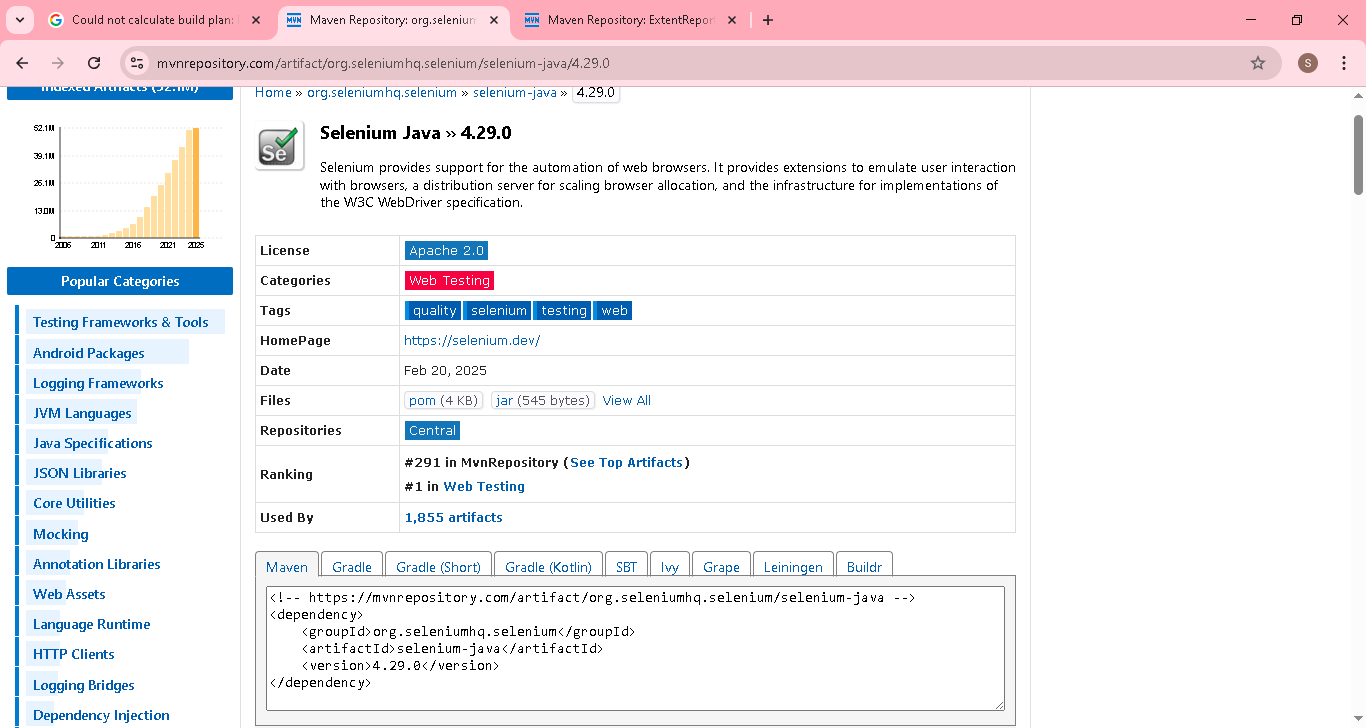
In that time, first poenxml will go and check in local path where extendsreport jar present or not.

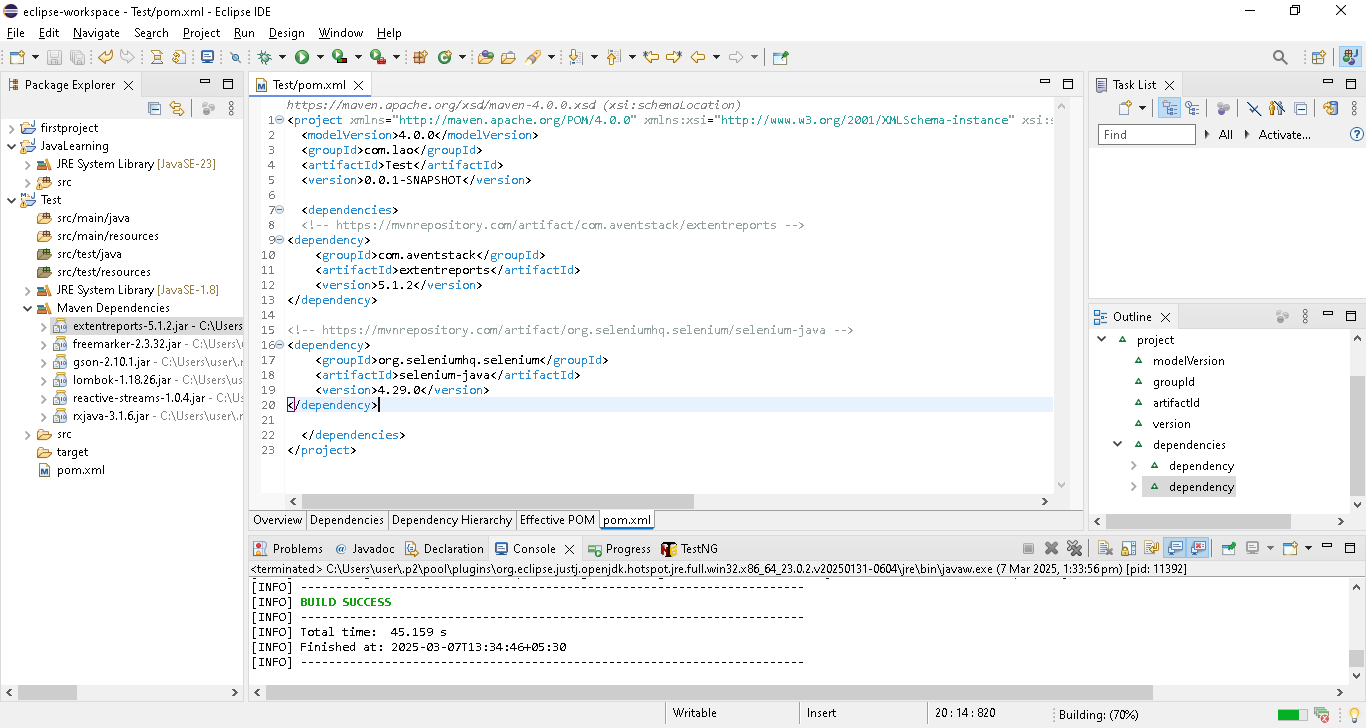
If present then it will download only selenium java(new added in pornxml)

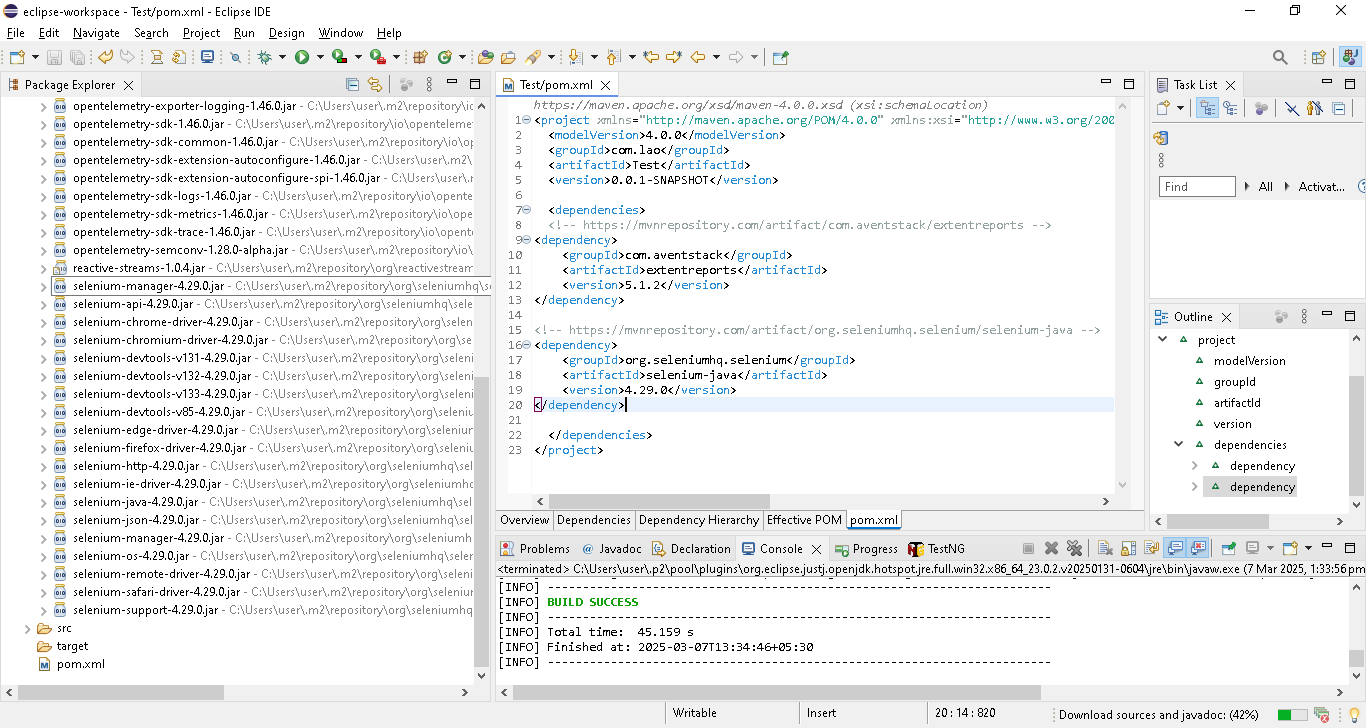
Each time it will download new jar based on jar available in local place.











**Note: if new member joining in ur team, just u need to share the porn.xml file he can download all project related jar at one go..saving lot of time & efforts**

3. **Can build the code for us**

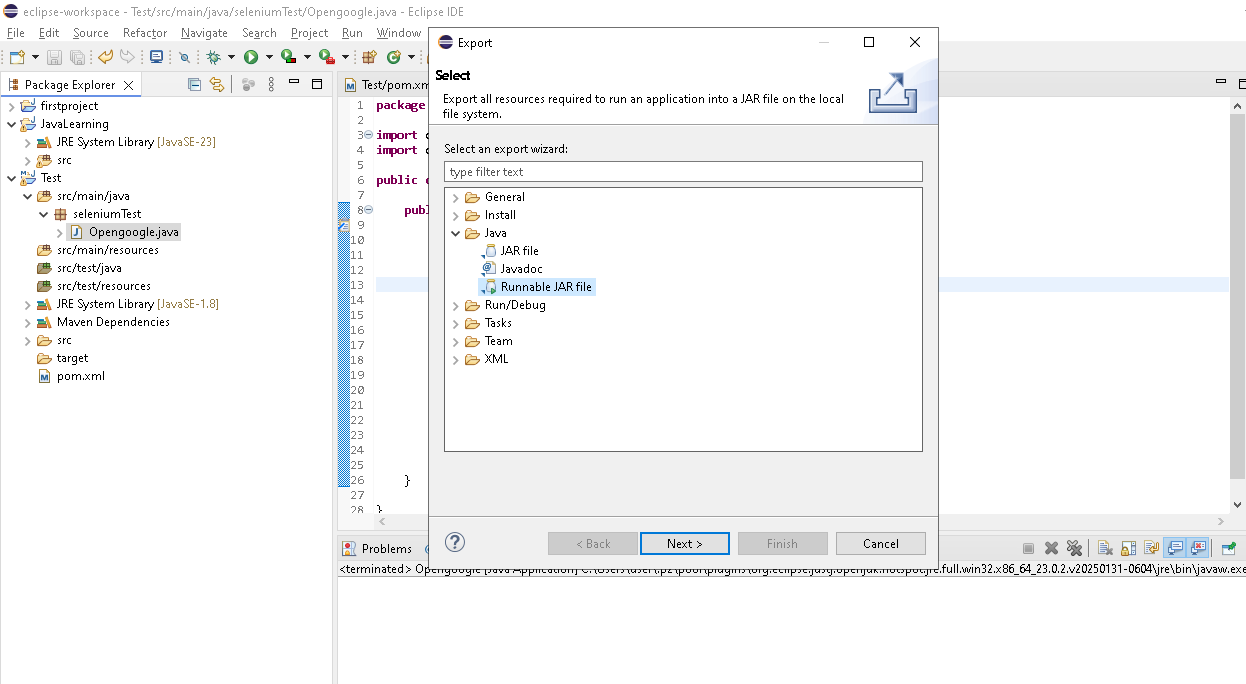
**Built process**->source code converted into executable file , called as built process[code converted into executable file & run in another device]

**So , In Maven making this built process easy..**

In project, once test case automation(code) completed, then we can convert the code in jar file using maven & provide to client for execution.

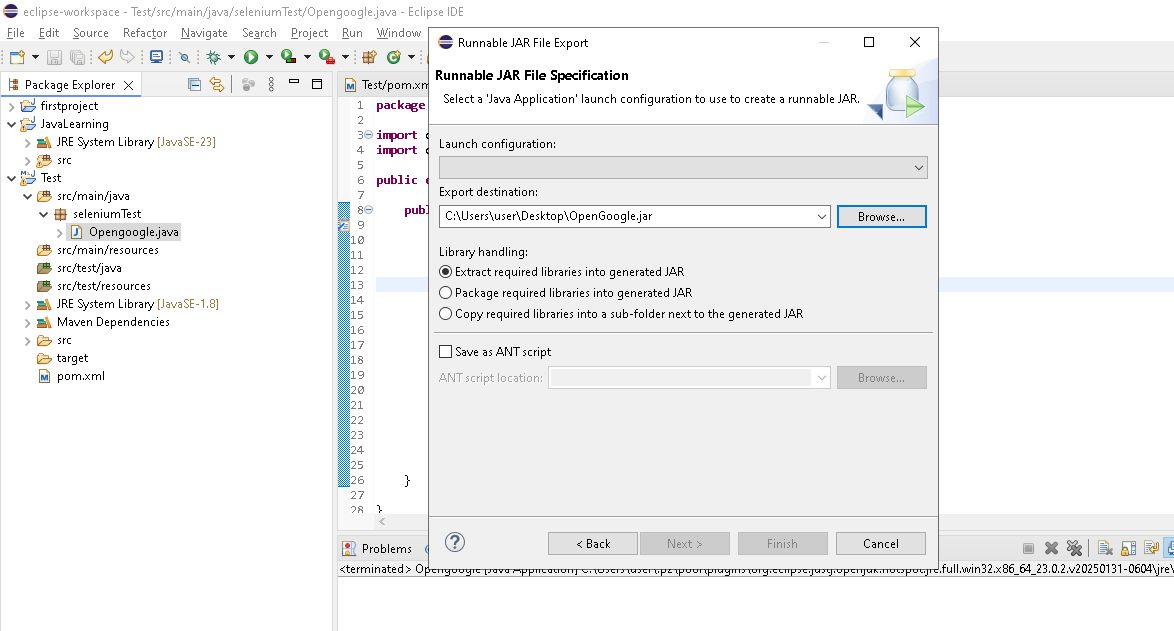
**First we will see normal process to convert the code into executable jar file**

Right click the Opengoogle class & export



& select Runnable JAR file & next

In the export destination : provide the place to store with any name(provided->OpenGoogle.ajar)



In launch configuration, we need to select the class name..in the dropdown it will show all the class across different package

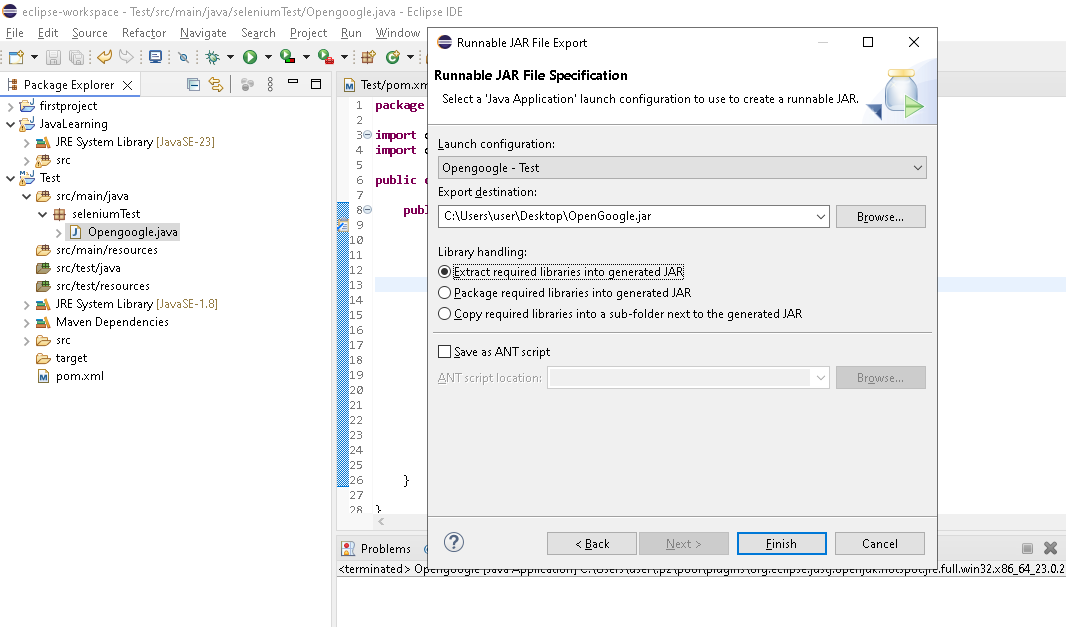


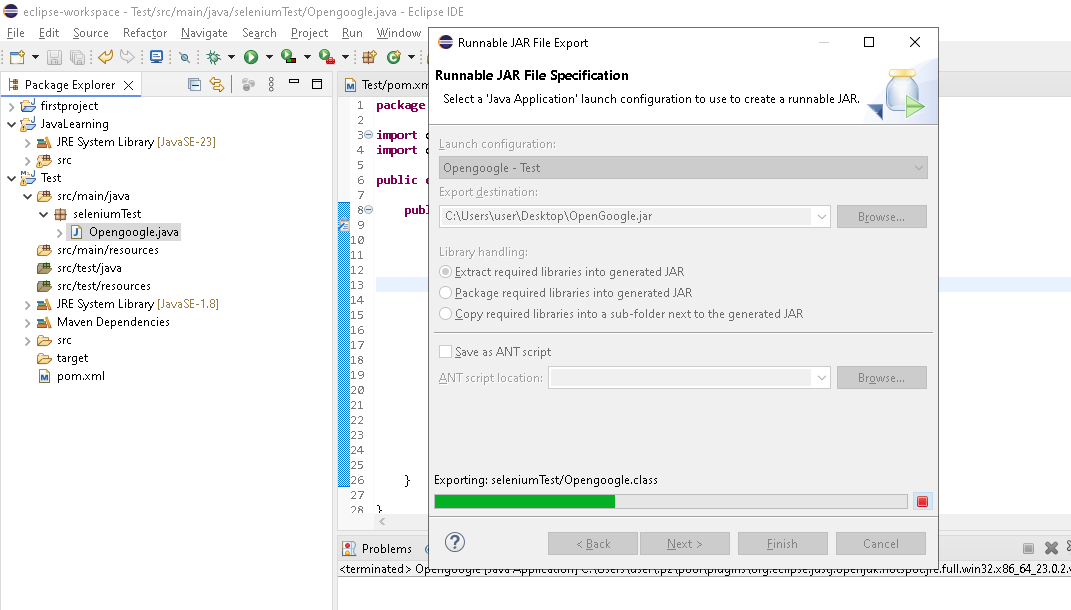
Under Library handing 3 option available..we can select any one based on need

Extract required lib into generated jar ->only required lib as part of opengoogle class code & combine with generated jar

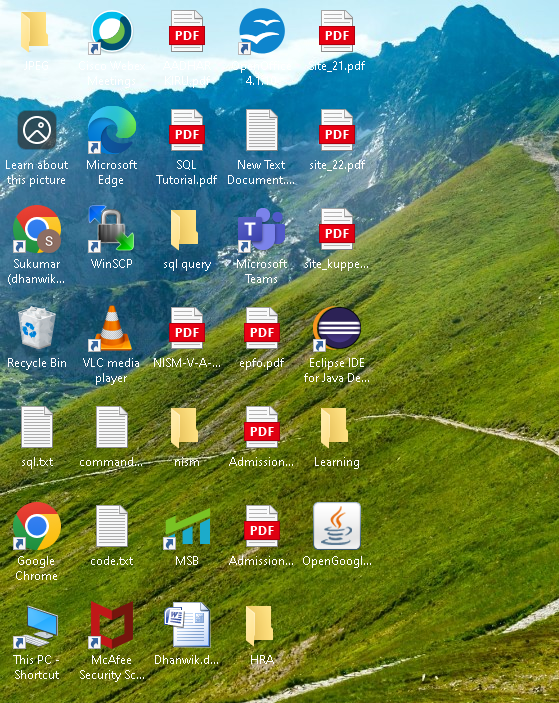
Package required lib into generated jar ->all lib & combine with generated jar

Copy required lib into a sub folder next to the generated jar ->it will create sub folder next to jar & contain req lib as part of opengoogle class code [lib separate from generated jar]

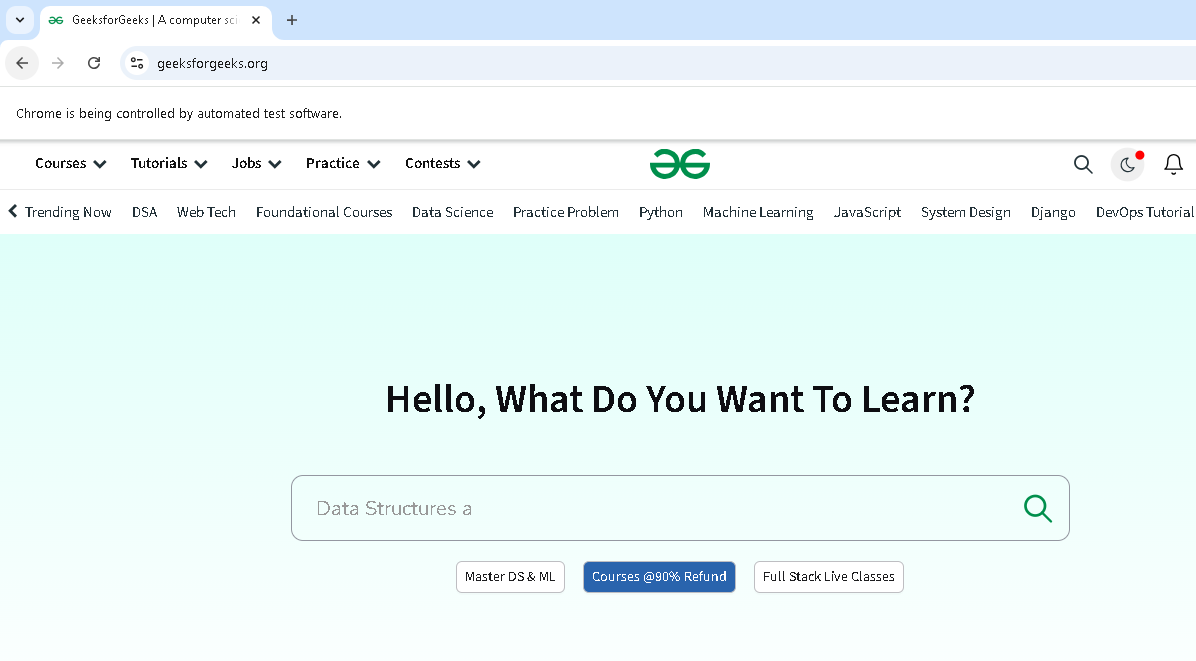




In desktop, executable jar present



Double click the jar..it will execute the opengoogle operation



**Second we will see how maven process to convert the code into executable jar file**

**Need to add the below plungins in order to convert the code into executable jar file**

**All plugin need to write inside <build> and followed by <plugins> & then <plugin?**

<build>

<plugins>

**First Plugin ->Telling run the compiler by using java JRE 1.8 version instead of other version(1.5)**

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.5</version>

<configuration>

<source>1.8</source>

<target>1.8</target>

</configuration>

</plugin>

**second Plugin (convert the code into executable jar file & tell the jar file to execute from mainclass) [**seleniumTest.Opengoogle][Package.class]

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-jar-plugin</artifactId>

<version>2.4</version>

<configuration>

<archive>

<manifest>

<addClasspath>true</addClasspath>

<mainClass>seleniumTest.Opengoogle</mainClass>

<classpathPrefix>lib/</classpathPrefix>

</manifest>

</archive>

</configuration>

</plugin>

**Third Plugin** -> to copy the all dependency lib associated with mainclass in local folder(lib)

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-dependency-plugin</artifactId>

<version>3.8.1</version>

<executions>

<execution>

<id>copy-dependencies</id>

<phase>package</phase>

<goals>

<goal>copy-dependencies</goal>

</goals>

<configuration>

<outputDirectory>${project.build.directory}/lib</outputDirectory>

<includeScope>runtime</includeScope>

</configuration>

</execution>

</executions>

</plugin>

</plugins>

</build>

----------------------------------------------------------

**Explanation below**

Need to add the plugin which include mainclass in Porn xml..then only jar file will run the particular mainclass

<mainClass>seleniumTest.Opengoogle</mainClass>

Need to add the plugin which include library in Porn xml..all dependency library associated with code will mapped

<classpathPrefix>lib/</classpathPrefix>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-jar-plugin</artifactId>

<version>2.4</version>

<configuration>

<archive>

<manifest>

<addClasspath>true</addClasspath>

<mainClass>seleniumTest.Opengoogle</mainClass>

<classpathPrefix>lib/</classpathPrefix>

</manifest>

</archive>

</configuration>

</plugin>

**Below plugin code copy the dependency library associated with code(Opengoogle class) & stored in lib folder locally**

For below code we need webdriver jar file..

Eg

System.*setProperty*(

"webdriver.chrome.driver",

"C:\\Users\\user\\Downloads\\chromedriver.exe");

// Instantiate a ChromeDriver class.

WebDriver driver = **new** ChromeDriver();

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-dependency-plugin</artifactId>

<version>3.8.1</version>

<executions>

<execution>

<id>copy-dependencies</id>

<phase>package</phase>

<goals>

<goal>copy-dependencies</goal>

</goals>

<configuration>

<outputDirectory>${project.build.directory}/lib</outputDirectory>

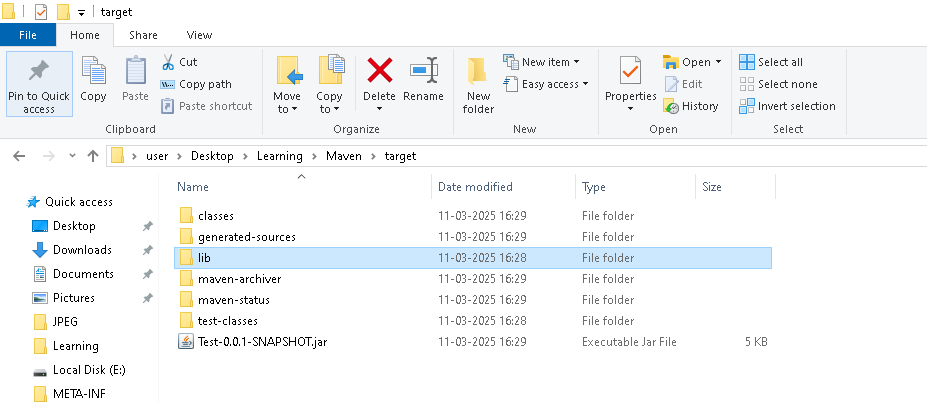
<includeScope>runtime</includeScope>

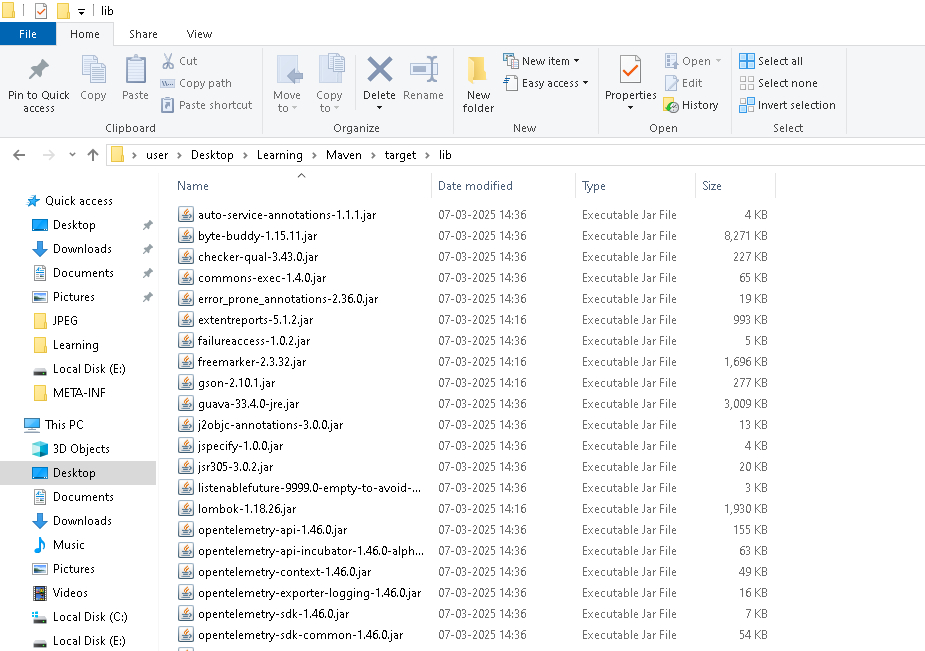
</configuration>

</execution>

</executions>

</plugin>





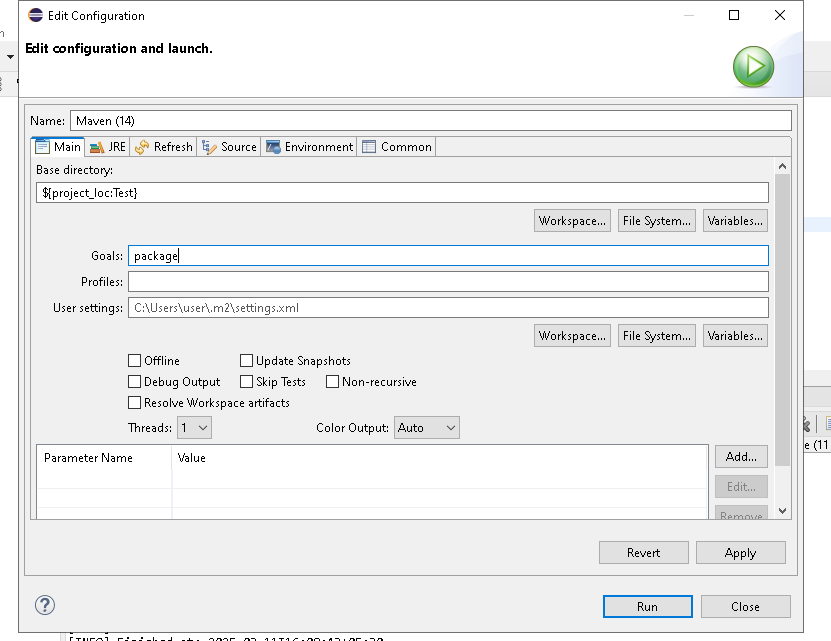
**After entering the above plunging code in pron.xml..save it**

Right click the package ->Maven->Update project..then only we can able to proceed with maven built [Note: whenever plunging updated then need to perform package ->Maven->Update project]

Right click the package ->Run as ->Maven build

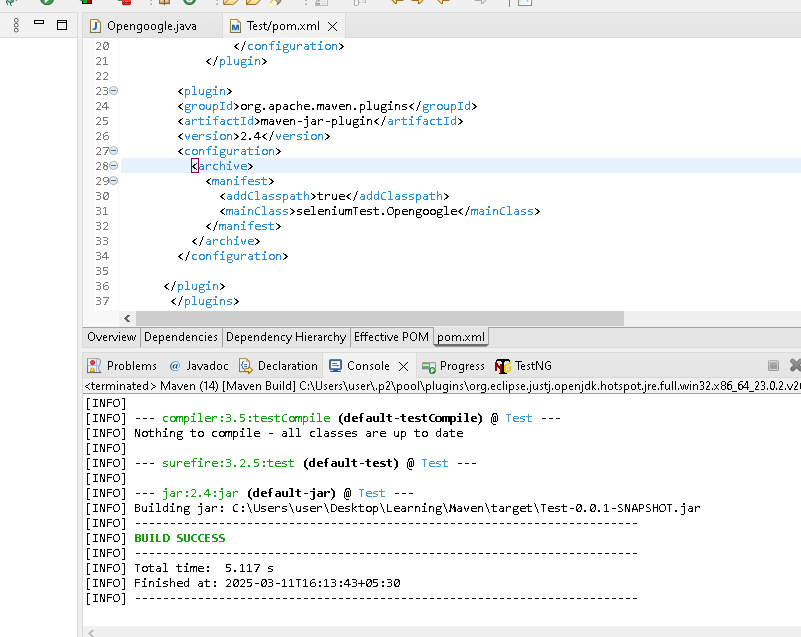
Enter “package” in goals [package keyword will create execute jar file]

Click Apply & then click run

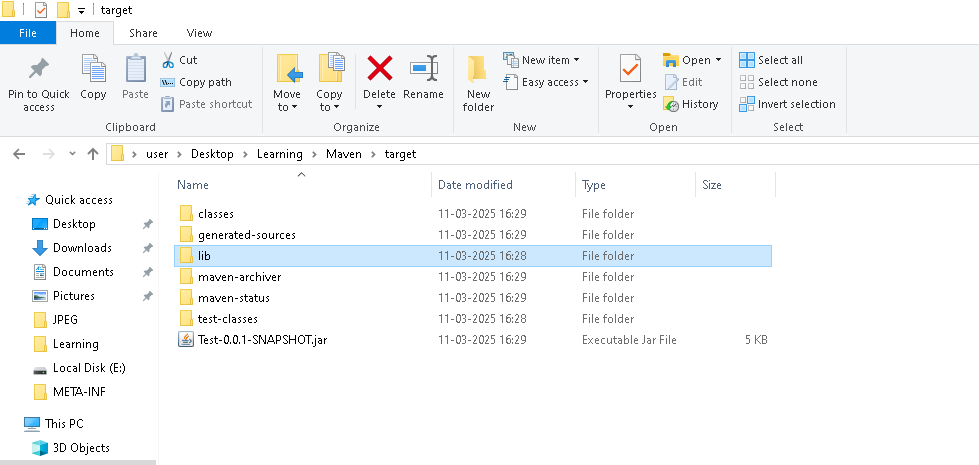


Navigate to console window. U can see the build success & path

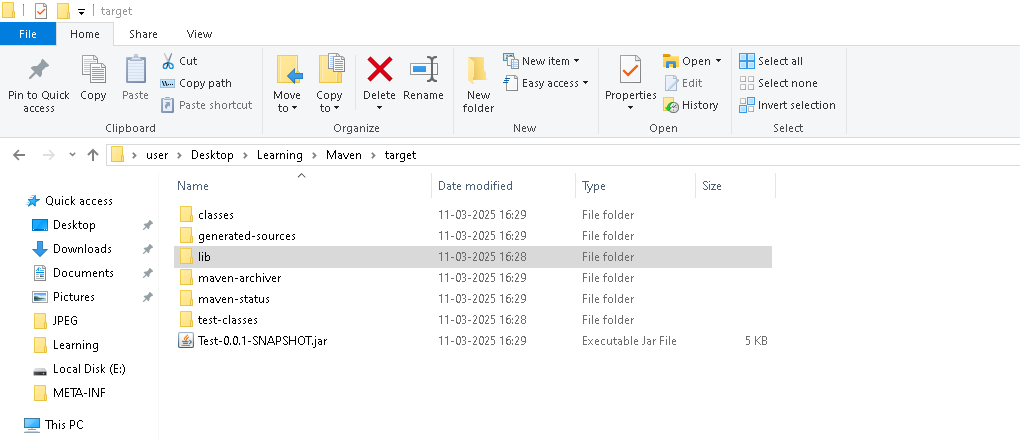
Building jar: C:\Users\user\Desktop\Learning\Maven\target\Test-0.0.1-SNAPSHOT.jar



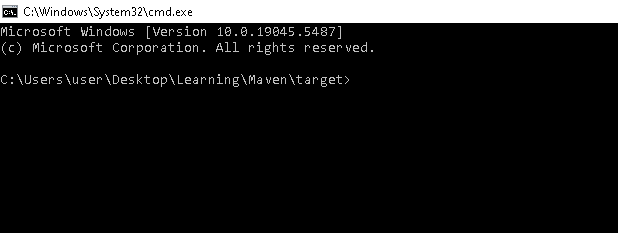
Go to local & check the path [lib created & Test-0.0.1 – snapshot.jar created]



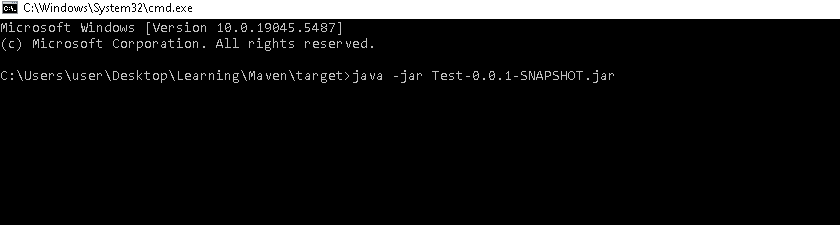
Go the local location where jar file present, type cmd in place of user->desktop->etc & enter



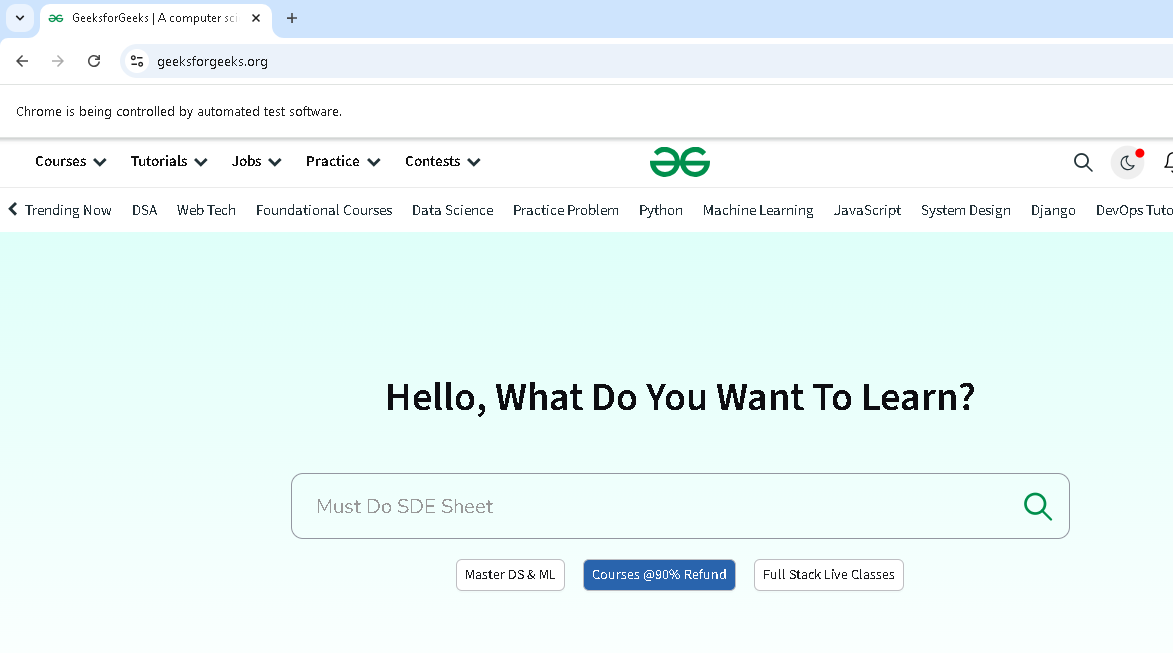
**Command prompt open like below**



Type java –jar Test-0.0.1 – snapshot.jar & enter



Prompt will execute the “Test-0.0.1 – snapshot.jar” & respective code, browser open successfully

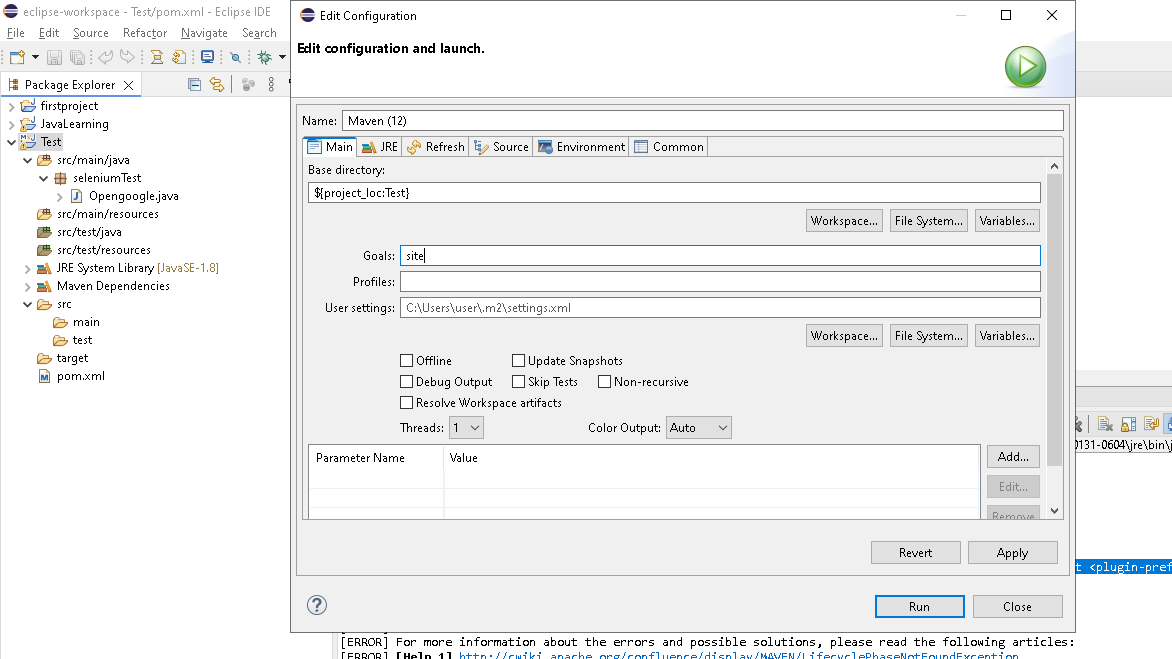


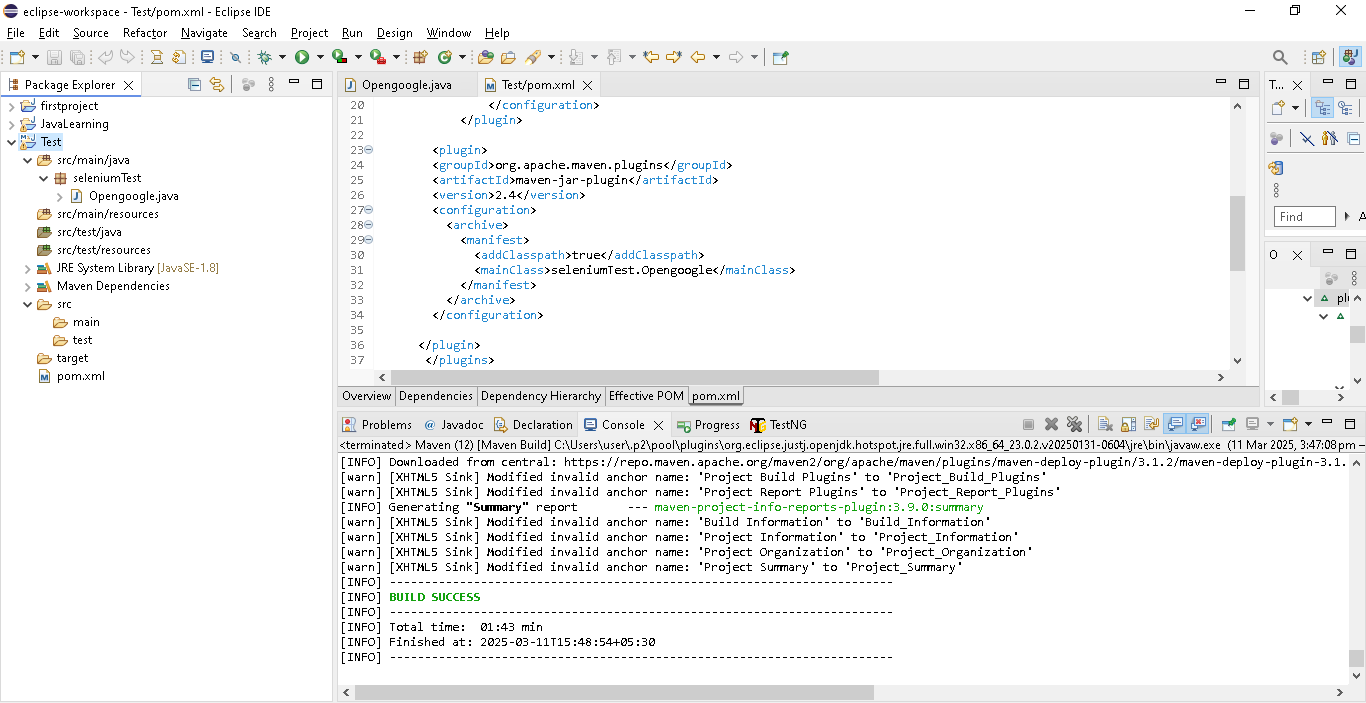
**4.Can create documentation sites->It contain dependency details, java detail, execution detail etc**

Right click the package ->Run as ->Maven build

Enter “site” in goals [site keyword will create documentation]

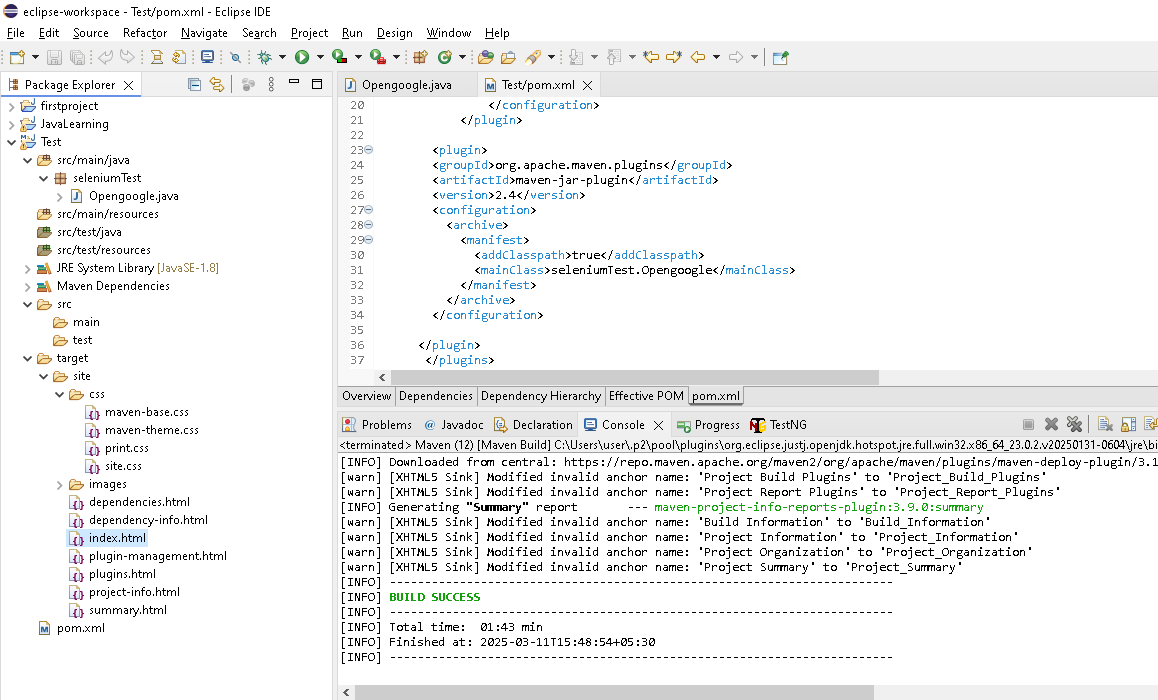
Click Apply & then click run

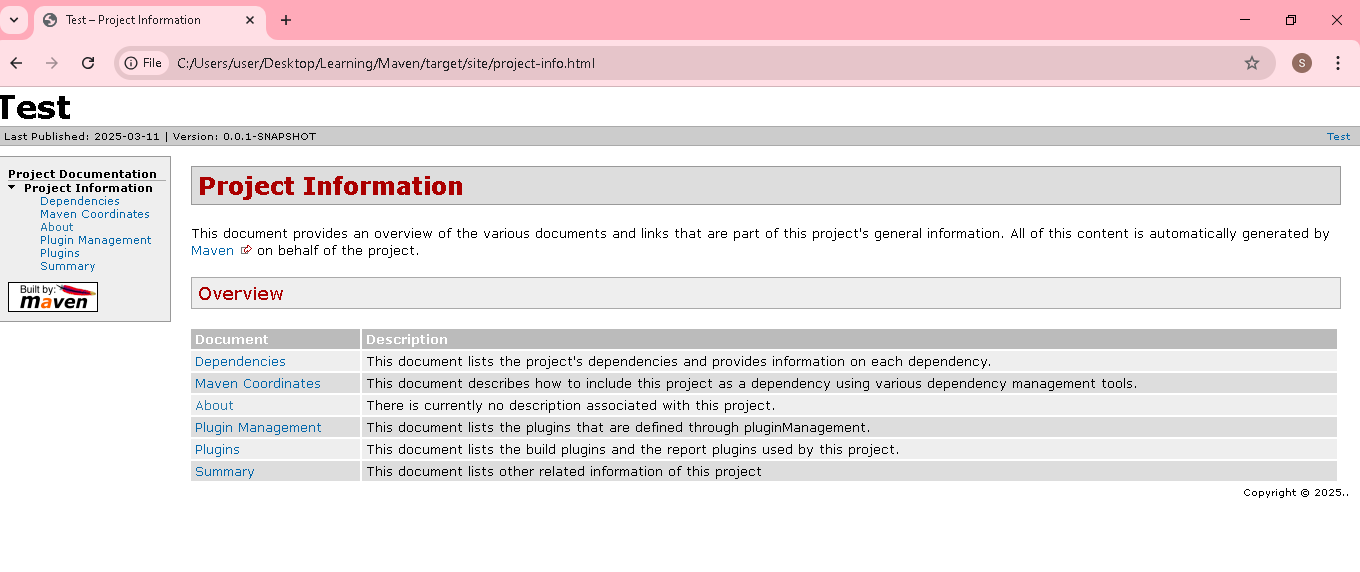




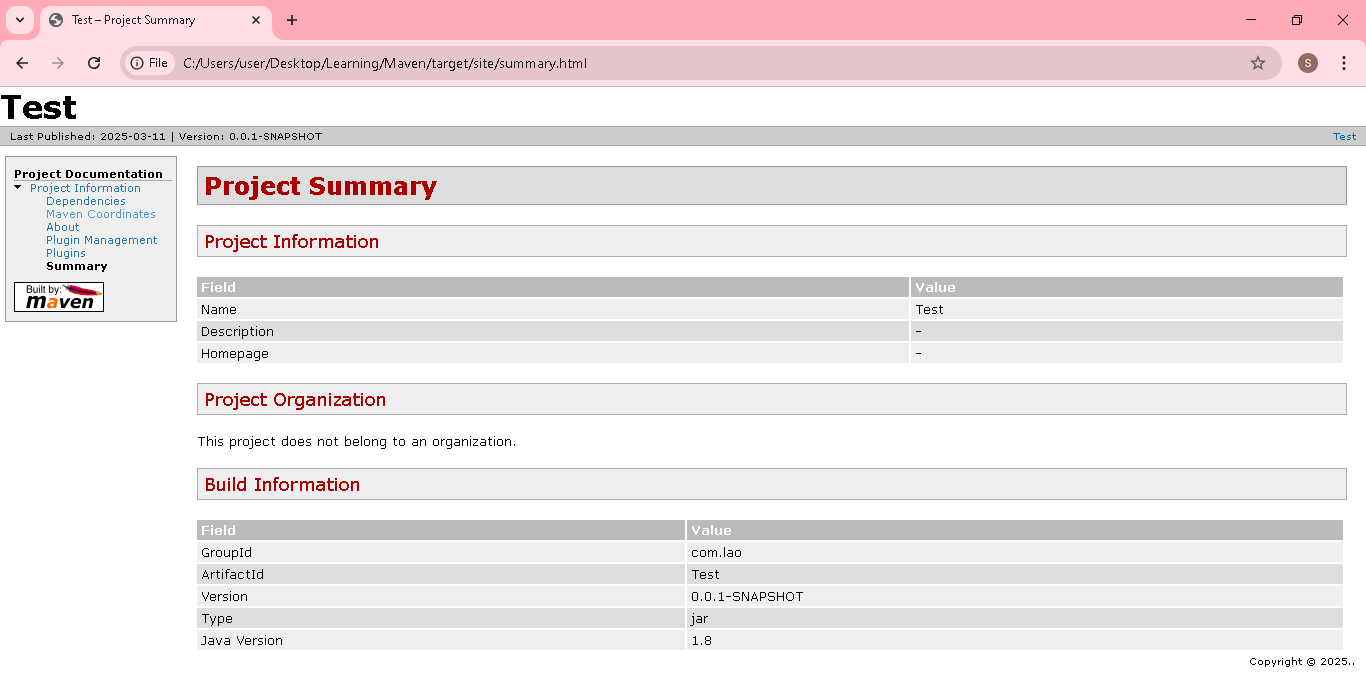
Refresh the package(test)

Nvaigate to target ->site ->image ->index.html





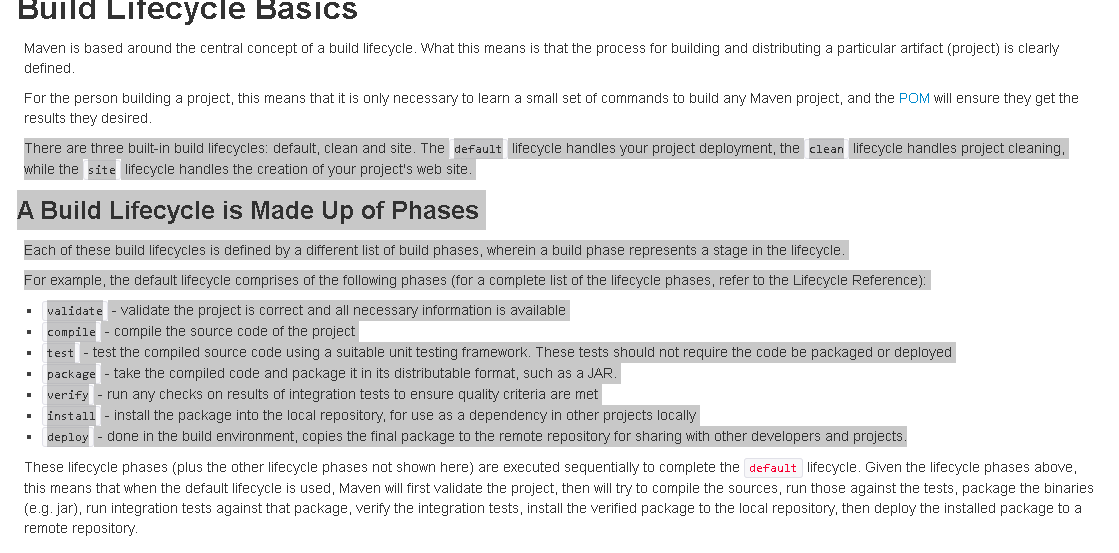




**Built life cycle of Maven**[life cycle -> comprises of several phases]

1. **Default** ->handles your project deployment in below 7 phases(validate, compile,test,package,verify,install, deploy)

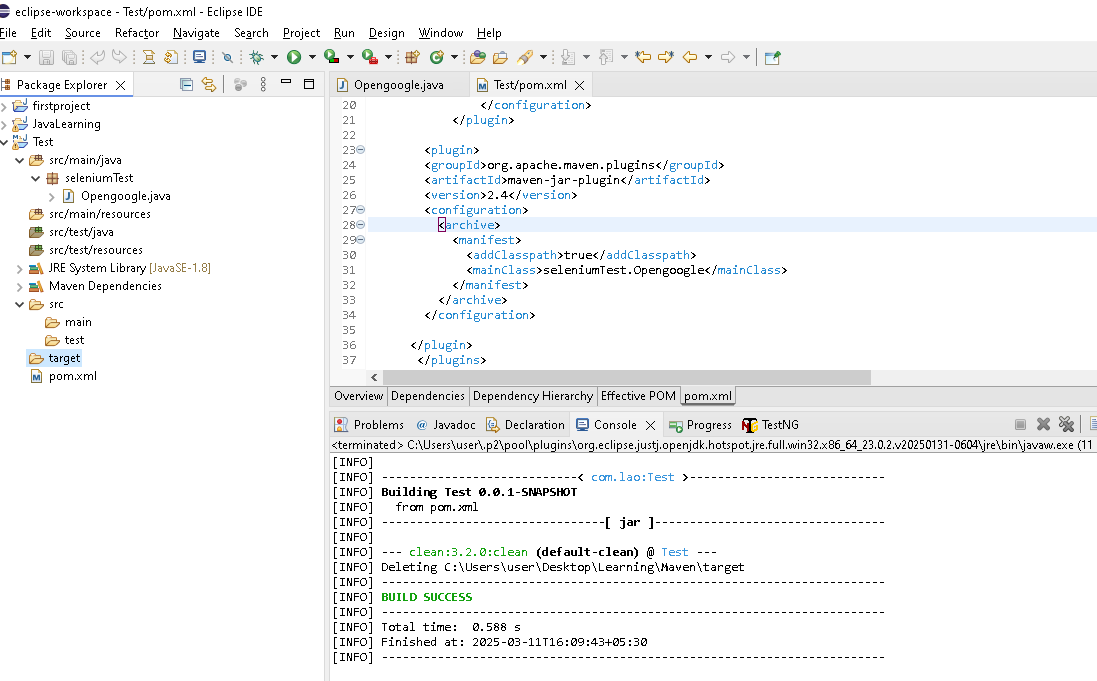
<https://maven.apache.org/guides/introduction/introduction-to-the-lifecycle.html>



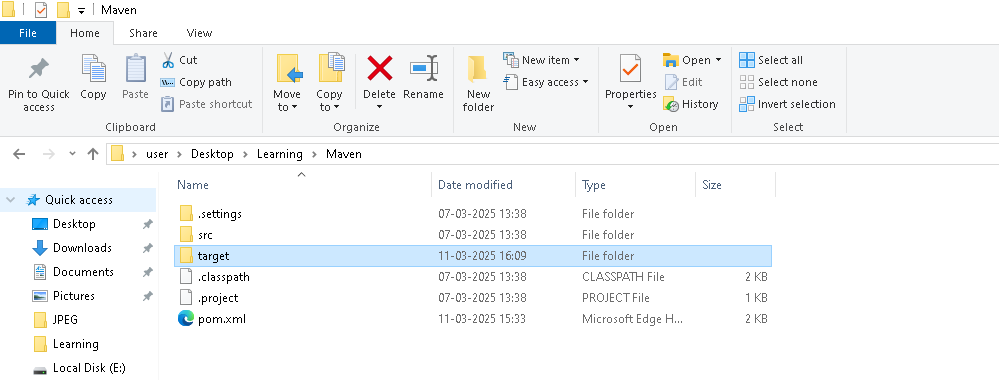
1. **Clean** ->Handles project cleaning->

Right click the package ->Run as ->Maven clean

It will delete all the files stored in target folder



**Its deleted the site documentation details which created previously & all stored lib, jar file etc**



1. **Site** ->handles the creation of project documentation report(as explained above)

-------------------------------------------------