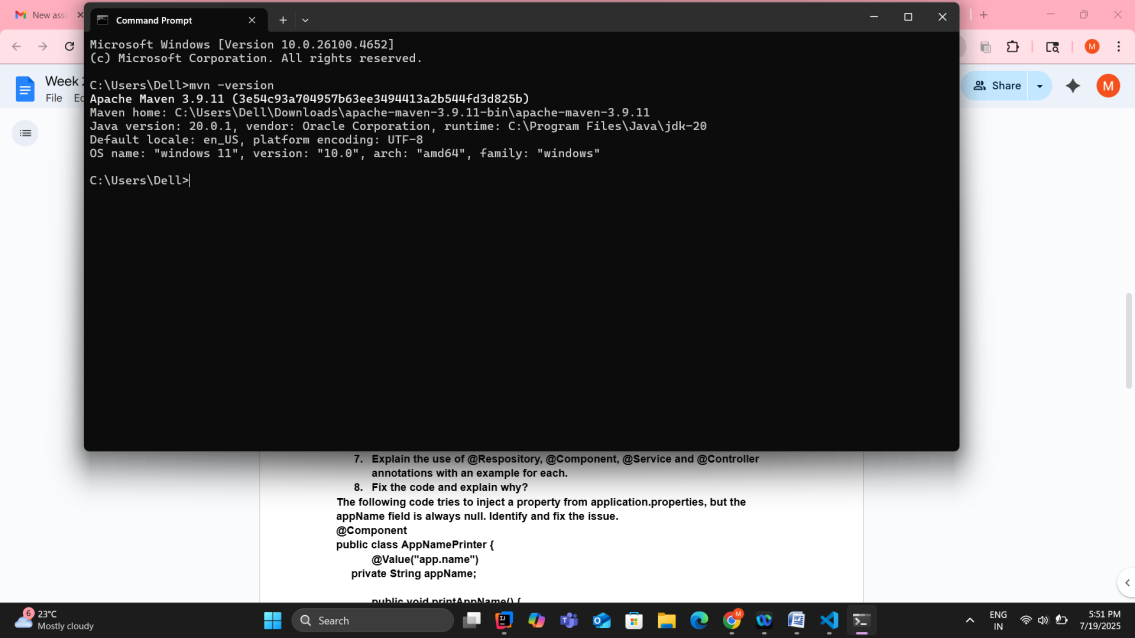
**Maven, Spring and Spring Boot**

**1. Install maven 3.6 or above. Execute mvn -v  in the local terminal/command prompt and share the screenshot**



2. **What is the difference between maven central repository and local repository?**

**Maven local repository** resides in the developer’s machine. Whenever we run Maven goals that require these dependencies, maven will download the dependencies from remote servers and store them on the developer’s machine.

By default, Maven creates the local repository inside the user’s home directory.

**Maven central repository** is located at [https://repo.maven.apache.org/maven2/](http://repo.maven.apache.org/maven2/). Whenever we run a build job, maven first tries to find dependency from the local repository. If it is not there, then, by default, maven will trigger the download from this

central repository location.

3. **Maven commands**

**a. To build the maven project**

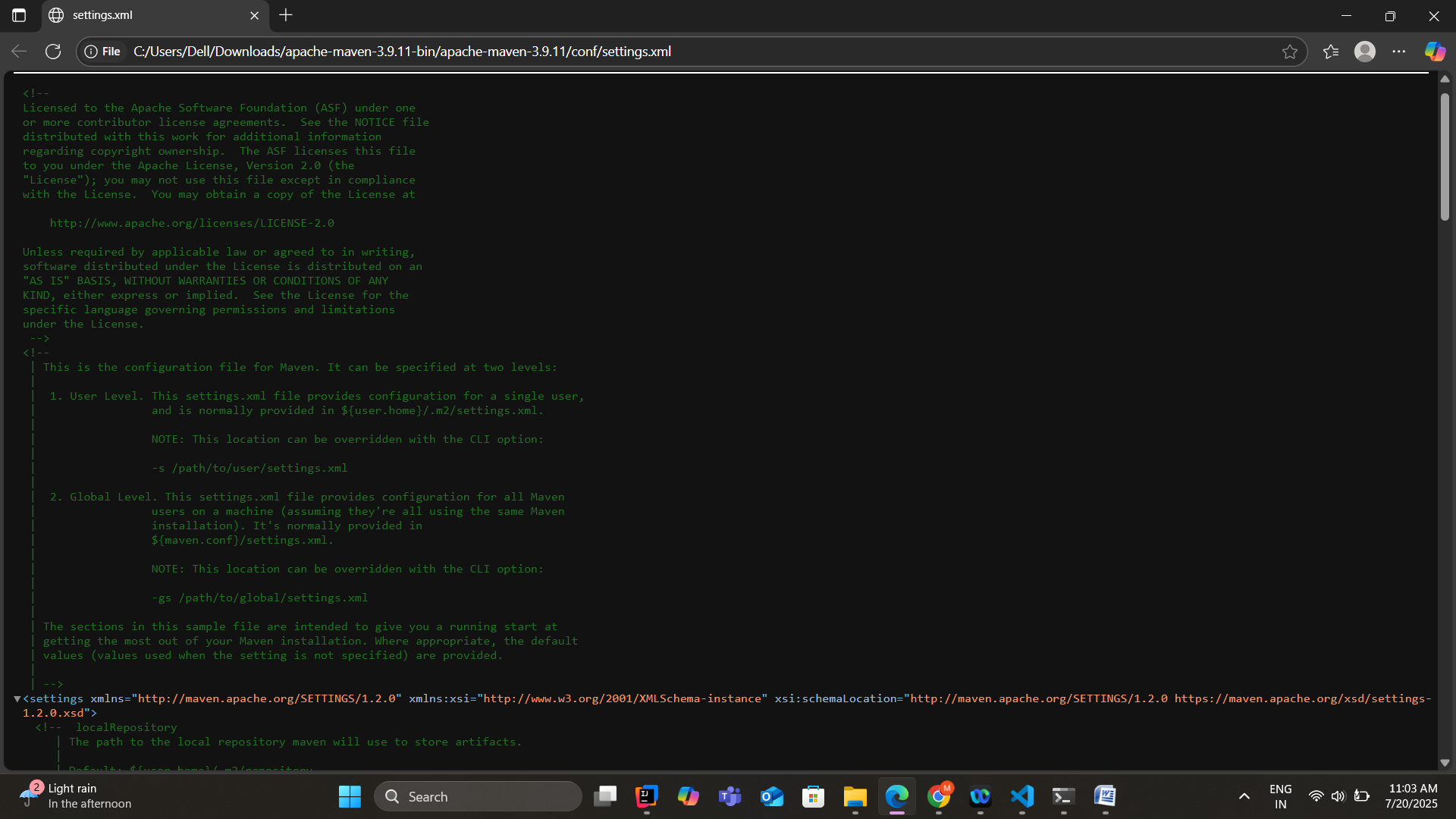
* mvn clean install
* **clean –** deletes the target/ directory or cleans previous builds
* **install –** compiles code, run tests, and install package into local maven repository

**b. To run the maven tests**

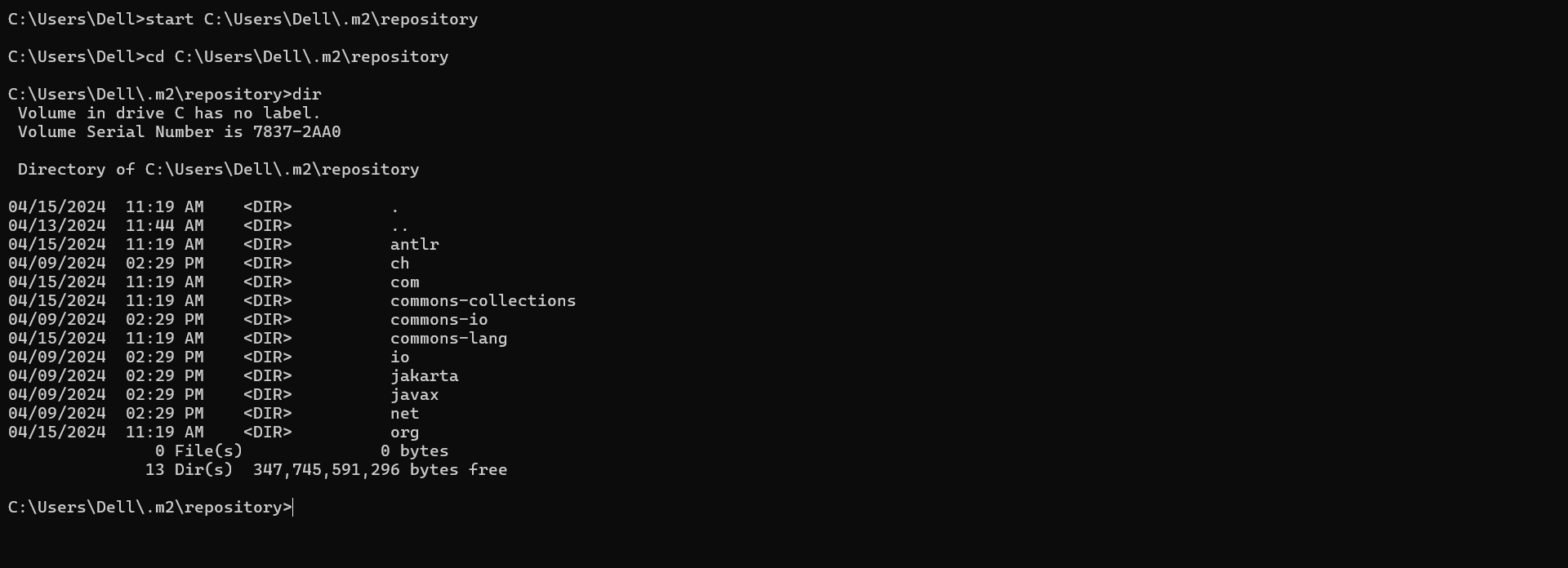
* mvn test

4. **Please locate the maven settings.xml file and local maven repository in your machine and share the screenshot**

**Settings.xml:**

****

**Local maven repository:**

****

**5. The basic principle behind Dependency Injection(DI) is that the objects define their dependencies .What are the different ways in which an object can define its   
dependency ?**

**There are different ways an object can define its dependency:**

**1.Constructor Injection:**

Dependencies are provided through the constructor

public class Car {

private Engine engine;

public Car(Engine engine) {

this.engine = engine;

}

}

**2.Setter Injection:**

Dependencies are set using **public setter methods** after object creation.

public class Car {

private Engine engine;

public void setEngine(Engine engine) {

this.engine = engine;

}

}

**3.Direct Injection(Field Injection):**

Dependencies are injected directly into the fields using annotations(@Autowired)

public class Car {

@Autowired

private Engine engine;

}

**4.Interface Injection:**

interface Drivable {

void drive();

}

class Car implements Drivable {

private Drivable engine;

public Car(Drivable engine) {

this.engine = engine;

}

@Override

public void drive() {

engine.drive();

}

}

6. **What is the difference between the @Autowired and @Inject annotation?**

We can annotate the fields and constructor using @Autowired to tell spring framework to find the dependencies. The @Inject annotation also serves the same purpose but the main difference is that @Inject is standard annotation for dependency injection and @Autowired is spring specific.

7. **Explain the use of @Respository, @Component, @Service and @Controller annotations with an example for each.**

**@Repository**

It indicates that the class interacts with the database.

@Repository

public class UserRepository {

public User findById(int id) {

return new User(id, "Alice");

}

}

**@Component**

@Component signifies that the spring framework will automatically detect and register this class as a spring bean within the application.

@Component

public class AgeValidator {

public boolean isValid(int age) {

if(age>100)

return false;

else

return true;

}

}

**@Service**

Marks a class as a service provider

@Service

public class UserService {

@Autowired

private UserRepository userRepository;

public User getUserDetails(int id) {

return userRepository.findById(id);

}

}

**@Controller**

It marks the class as a spring MVC controller. Handles the web requests using @RequestMapping

@Controller

public class UserController {

@Autowired

private UserService userService;

@GetMapping("/user/{id}")

@ResponseBody

public User getUser(@PathVariable int id) {

return userService.getUserDetails(id);

}

}

**8.Fix the code and explain why?**

**The following code tries to inject a property from application.properties, but the appName field is always null. Identify and fix the issue.**

**@Component**

**public class AppNamePrinter {**

**@Value("app.name")**

**private String appName;**

**public void printAppName() {  
     System.out.println("Application Name: " + appName);  
 }  
 }**

**Here app.name is treated as a string not as a spring placeholder. So the appName remains null.**

@Component

public class AppNamePrinter {

@Value("${app.name}")

private String appName;

public void printAppName() {

System.out.println("Application Name: " + appName);

}

}

9. **What does the @SpringBootApplication annotation do?**

The @SpringBootApplication annotation is a **convenience annotation** that combines several key Spring annotations to quickly configure and bootstrap a Spring Boot application.

It combines  
@SpringBootConfiguration

@EnableAutoConfiguration

@ComponentScan

10. **What is the maven command to start the SpringBootApplication?**

**mvn spring-boot:run**

**11. Implement EmployeeCRUD using Spring and JDBC with the below Employee class. In the branch feature-spring, create a folder Employee-Spring. Push the solution to the branch and share the link.**

**class Employee{**

**private int id;**

**private String name;**

**private String department;**

**}**

<https://github.com/Madhuri0987/rg_assignment/tree/feature-spring/Employee-spring>

12. **Implement EmployeeCRUD using SpringBoot and Spring Data JPA with the below Employee class. In the branch feature-spring, create a folder Employee-SpringBoot-JPA. Push the solution to the branch and share the link.**

**class Employee{**

**private int id;**

**private String name;**

**private String department;**

**}**

<https://github.com/Madhuri0987/rg_assignment/tree/feature-spring/SpringBoot-JPA>

13. **Follow the demo in the pre-work link** [**https://www.youtube.com/watch?v=hr2XTbKSdAQ&t=18s**](https://www.youtube.com/watch?v=hr2XTbKSdAQ&t=18s) **and create a Spring Batch application that processes customer data.  In the branch feature-spring, create a folder Customer-SpringBatch. Push the solution to the branch and share the link.**

[**https://github.com/Madhuri0987/rg\_assignment/tree/feature-spring/Customer\_SpringBatch**](https://github.com/Madhuri0987/rg_assignment/tree/feature-spring/Customer_SpringBatch)