**SME (Subject Matter Expert) walkthrough:**

### 1. src/main/java – Folder with application code

📁 This is where our **main source code** lives.

Example path:

swift

CopyEdit

src/main/java/com/cognizant/springlearn/SpringLearnApplication.java

We can say:

"This folder contains the Java classes of our Spring Boot application. Currently, it includes the main class SpringLearnApplication, and later we can add controllers, services, or entities here."

### 2. src/main/resources – Folder for application configuration

📁 This contains:

application.properties or application.yml

We can say:

"This folder is used for configuration files like application.properties. It's where we define things like server ports, database URLs, and other application settings."

### 3. src/test/java – Folder with code for testing the application

📁 This is used for writing unit and integration tests.

We can say:

"This folder is structured similarly to main/java but is used for test classes. By default, Spring Boot generates a test for the main application class."

### 4. SpringLearnApplication.java – Walkthrough the main() method

Show this file and explain:

java

CopyEdit

@SpringBootApplicationpublic class SpringLearnApplication {

public static void main(String[] args) {

System.out.println("SpringLearnApplication started!");

SpringApplication.run(SpringLearnApplication.class, args);

}

}

Explain:

"This is the entry point of the Spring Boot application. The main() method launches the app using SpringApplication.run(), and we added a log to confirm startup."

### 5. Purpose of @SpringBootApplication annotation

Explain:

"@SpringBootApplication is a combination of three annotations:

@Configuration: marks the class as a source of bean definitions

@EnableAutoConfiguration: enables Spring Boot’s auto-configuration

@ComponentScan: tells Spring to scan the package for components like controllers and services."

So, it's a shortcut that tells Spring Boot to **auto-configure the application** and scan for components.

### 6. pom.xml – Walkthrough

Open the pom.xml file and show:

Project metadata: groupId, artifactId, version

Dependencies: Spring Web, Spring Boot DevTools

Plugins (optional, for build and run config)

You can say:

"The pom.xml is the build configuration file for Maven. It contains metadata and lists all dependencies and plugins required by the project."

### 6.2. Open 'Dependency Hierarchy' and show the tree

In **VS Code**, go to:

**Explorer > MAVEN** (on the left sidebar) → expand your project → expand **Dependencies**

Explain:

"This shows all the libraries our project depends on — including transitive dependencies pulled in by Spring Web and Spring Boot DevTools."