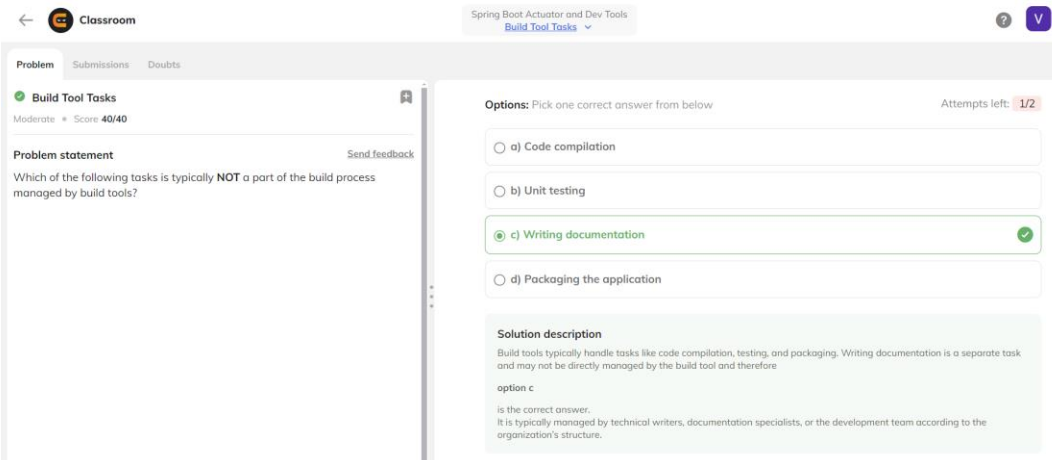
Quiz Questions

(1) Build Tool Tasks



✓ Build Tool Tasks

Moderate • Score 40/40

Problem statement

Which of the following tasks is typically **NOT** a part of the build process managed by build tools?

Options: Pick one correct answer from below Attempts left: 1/2

○ a) Code compilation

○ b) Unit testing

● c) Writing documentation ✓

○ d) Packaging the application

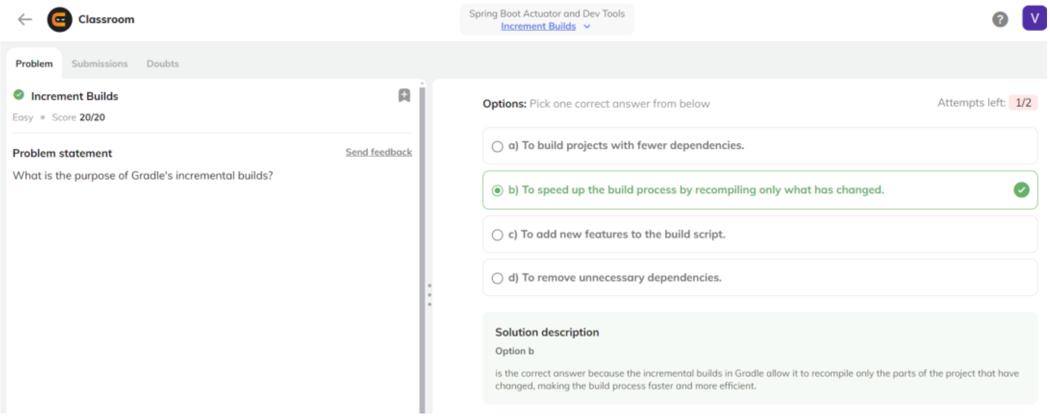
Solution description

Build tools typically handle tasks like code compilation, testing, and packaging. Writing documentation is a separate task and may not be directly managed by the build tool and therefore

option c

is the correct answer.
It is typically managed by technical writers, documentation specialists, or the development team according to the organization's structure.

(2) Increment Builds



✓ Increment Builds

Easy • Score 20/20

Problem statement

What is the purpose of Gradle's incremental builds?

Options: Pick one correct answer from below Attempts left: 1/2

○ a) To build projects with fewer dependencies.

● b) To speed up the build process by recompiling only what has changed. ✓

○ c) To add new features to the build script.

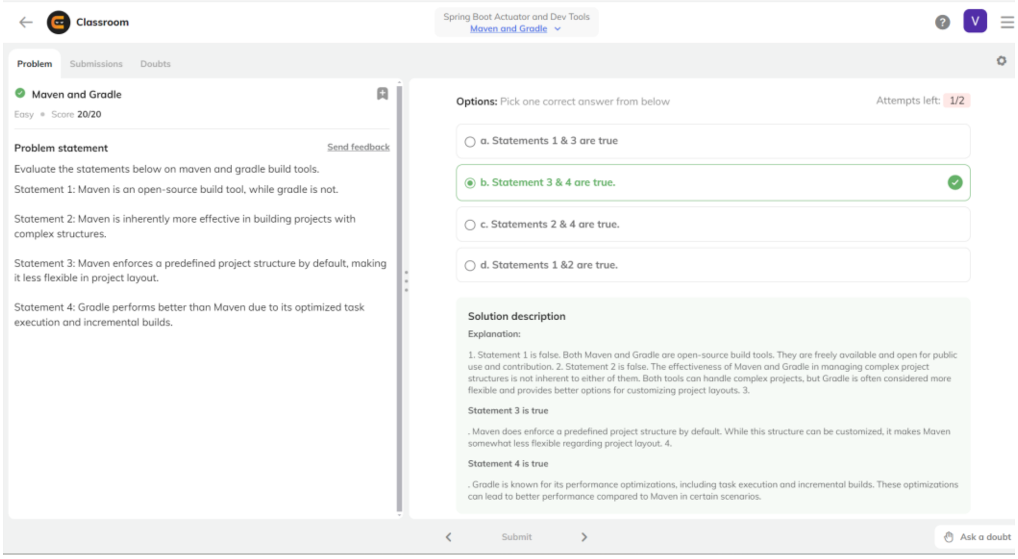
○ d) To remove unnecessary dependencies.

Solution description

Option b

is the correct answer because the incremental builds in Gradle allow it to recompile only the parts of the project that have changed, making the build process faster and more efficient.

(3) Maven and Gradle



✓ Maven and Gradle

Easy • Score 20/20

Problem statement

Evaluate the statements below on maven and gradle build tools.

Statement 1: Maven is an open-source build tool, while gradle is not.

Statement 2: Maven is inherently more effective in building projects with complex structures.

Statement 3: Maven enforces a predefined project structure by default, making it less flexible in project layout.

Statement 4: Gradle performs better than Maven due to its optimized task execution and incremental builds.

Options: Pick one correct answer from below Attempts left: 1/2

○ a. Statements 1 & 3 are true

● b. Statement 3 & 4 are true. ✓

○ c. Statements 2 & 4 are true.

○ d. Statements 1 &2 are true.

Solution description

Explanation:

1. Statement 1 is false. Both Maven and Gradle are open-source build tools. They are freely available and open for public use and contribution. 2. Statement 2 is false. The effectiveness of Maven and Gradle in managing complex project structures is not inherent to either of them. Both tools can handle complex projects, but Gradle is often considered more flexible and provides better options for customizing project layouts. 3.

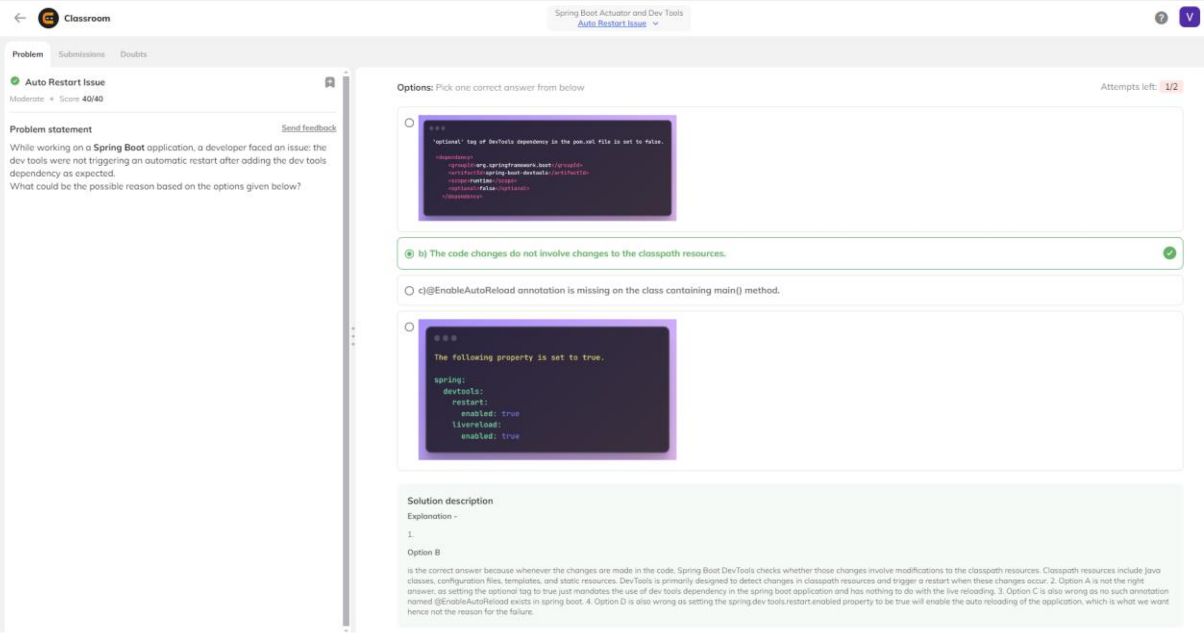
Statement 3 is true

. Maven does enforce a predefined project structure by default. While this structure can be customized, it makes Maven somewhat less flexible regarding project layout. 4.

Statement 4 is true

. Gradle is known for its performance optimizations, including task execution and incremental builds. These optimizations can lead to better performance compared to Maven in certain scenarios.

(4) Auto Restart Issue



Auto Restart Issue

Moderate • Score 40/40

Problem statement

While working on a **Spring Boot** application, a developer faced an issue: the dev tools were not triggering an automatic restart after adding the dev tools dependency as expected.
What could be the possible reason based on the options given below?

Options: Pick one correct answer from below

Attempts left: 1/2

○

```
'optional' tag of DevTools dependency in the pom.xml file is set to false.

<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-devtools</artifactId>
    <scope>runtime</scope>
    <optional>false</optional>
</dependency>
```

● b) The code changes do not involve changes to the classpath resources. ✓

○ c)@EnableAutoReload annotation is missing on the class containing main() method.

○

```
The following property is set to true.

spring:
  devtools:
    restart:
      enabled: true
    livereload:
      enabled: true
```

Solution description

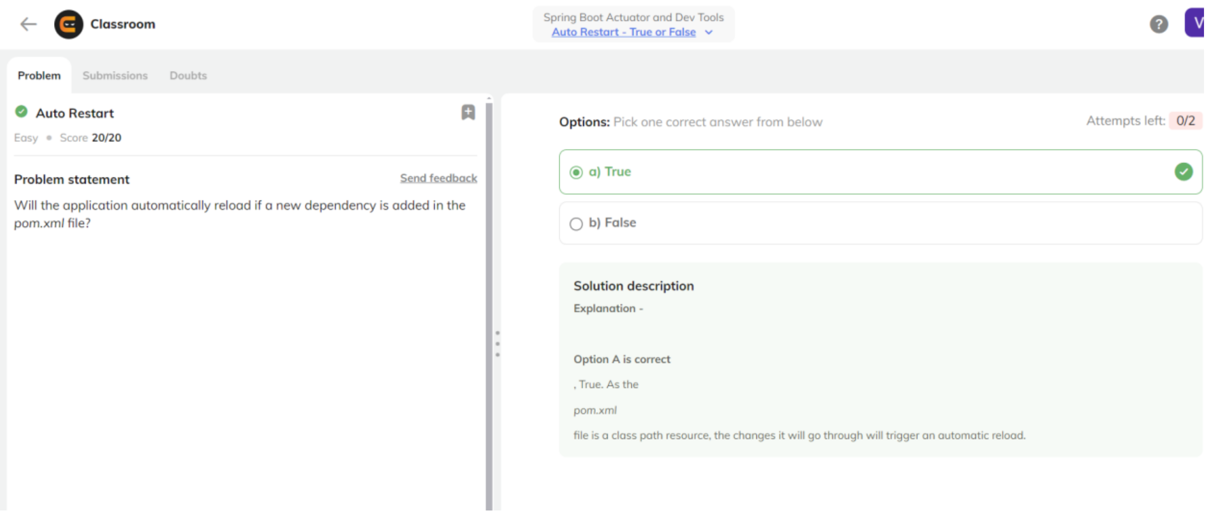
Explanation -

1.

Option B

is the correct answer because whenever the changes are made in the code, Spring Boot DevTools checks whether those changes involve modifications to the classpath resources. Classpath resources include Java classes, configuration files, templates, and static resources. DevTools is primarily designed to detect changes in classpath resources and trigger a restart when these changes occur. 2. Option A is not the right answer, as setting the optional tag to true just mandates the use of dev tools dependency in the spring boot application and has nothing to do with the live reloading. 3. Option C is also wrong as no such annotation named @EnableAutoReload exists in spring boot. 4. Option D is also wrong as setting the spring.dev tools.restart.enabled property to be true will enable the auto reloading of the application, which is what we want hence not the reason for the failure.

(5) Auto Restart – True or False


Classroom
Spring Boot Actuator and Dev Tools
Auto Restart - True or False
Problem
Submissions
Doubts

Auto Restart

Easy • Score 20/20

Problem statement

Will the application automatically reload if a new dependency is added in the pom.xml file?

Options: Pick one correct answer from below

Attempts left: 0/2

● a) True ✓

○ b) False

Solution description

Explanation -

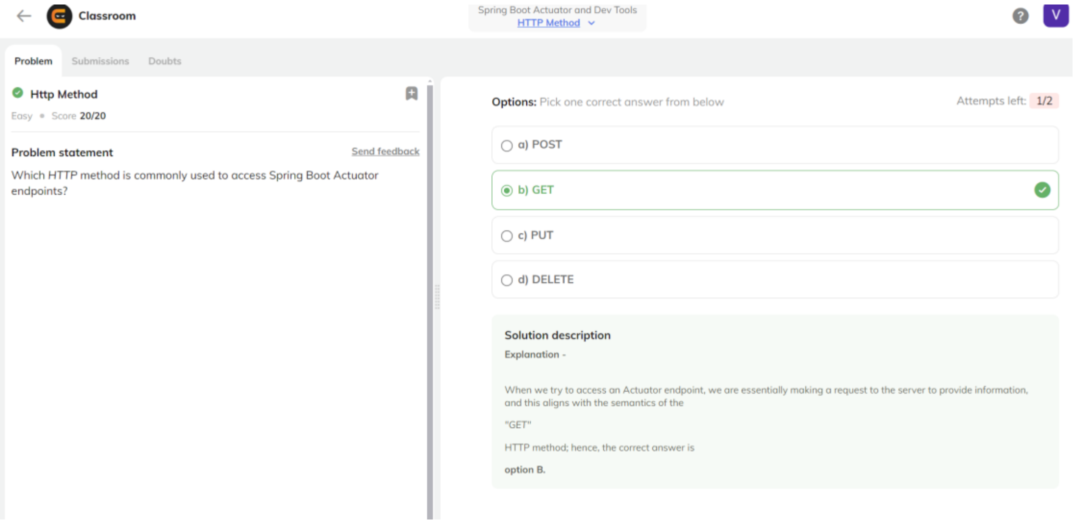
Option A is correct

, True. As the

pom.xml

file is a class path resource, the changes it will go through will trigger an automatic reload.

(6) Http Method



Http Method

Easy • Score 20/20

Problem statement

Which HTTP method is commonly used to access Spring Boot Actuator endpoints?

Options: Pick one correct answer from below

Attempts left: 1/2

○ a) POST

● b) GET ✓

○ c) PUT

○ d) DELETE

Solution description

Explanation -

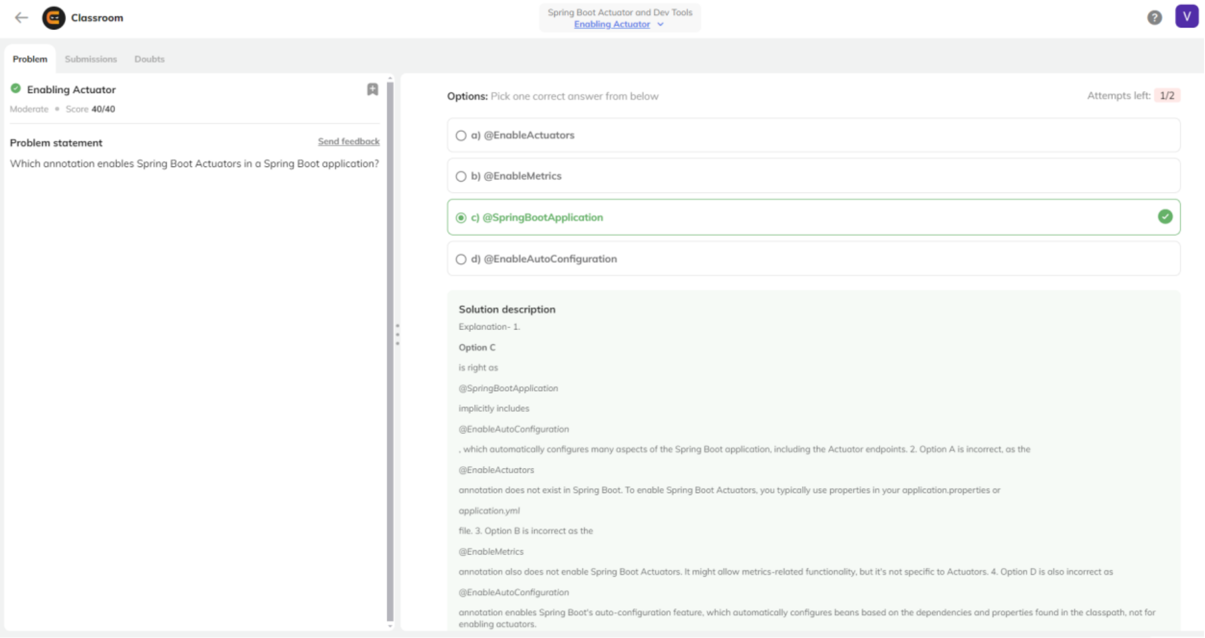
When we try to access an Actuator endpoint, we are essentially making a request to the server to provide information, and this aligns with the semantics of the

"GET"

HTTP method; hence, the correct answer is

option B.

(7) Enabling Actuators



Enabling Actuator

Moderate • Score 40/40

Problem statement

Which annotation enables Spring Boot Actuators in a Spring Boot application?

Options: Pick one correct answer from below

Attempts left: 1/2

○ a) @EnableActuators

○ b) @EnableMetrics

● c) @SpringBootApplication ✓

○ d) @EnableAutoConfiguration

Solution description

Explanation- 1.

Option C

is right as

@SpringBootApplication

implicitly includes

@EnableAutoConfiguration

, which automatically configures many aspects of the Spring Boot application, including the Actuator endpoints. 2. Option A is incorrect, as the

@EnableActuators

annotation does not exist in Spring Boot. To enable Spring Boot Actuators, you typically use properties in your application.properties or

application.yml

file. 3. Option B is incorrect as the

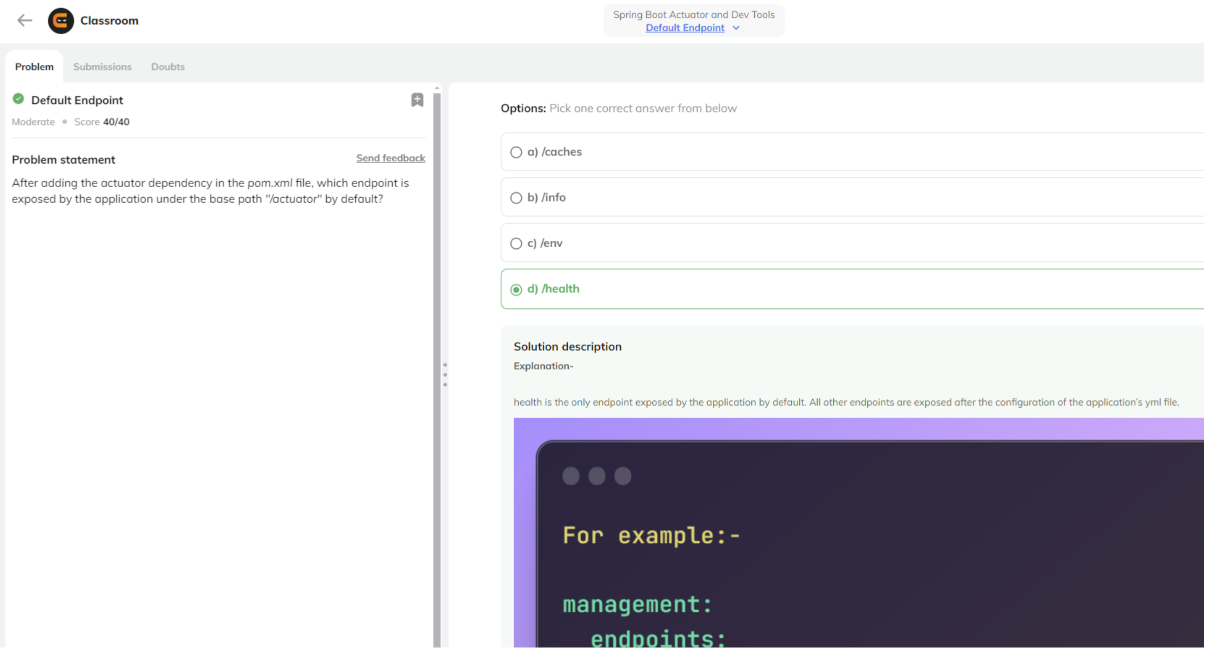
@EnableMetrics

annotation also does not enable Spring Boot Actuators. It might allow metrics-related functionality, but it's not specific to Actuators. 4. Option D is also incorrect as

@EnableAutoConfiguration

annotation enables Spring Boot's auto-configuration feature, which automatically configures beans based on the dependencies and properties found in the classpath, not for enabling actuators.

(8) Default Endpoint



Default Endpoint

Moderate • Score 40/40

Problem statement

After adding the actuator dependency in the pom.xml file, which endpoint is exposed by the application under the base path "/actuator" by default?

Options: Pick one correct answer from below

○ a) /caches

○ b) /info

○ c) /env

● d) /health

Solution description

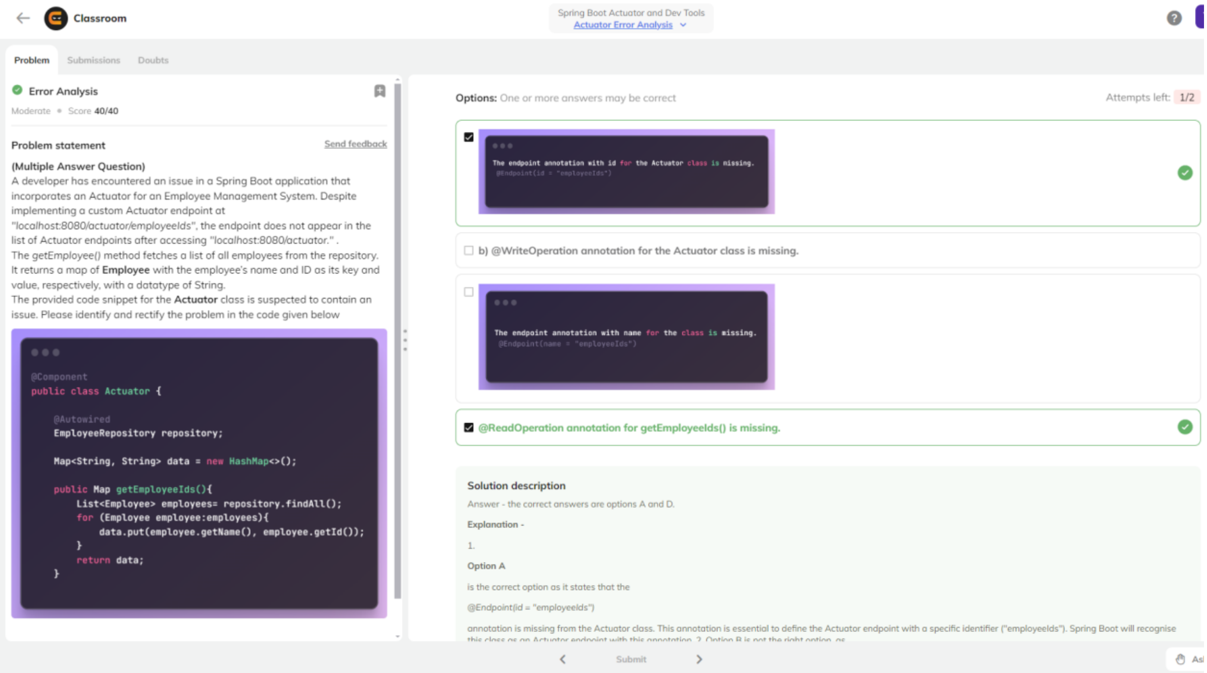
Explanation-

health is the only endpoint exposed by the application by default. All other endpoints are exposed after the configuration of the application's yml file.

```
For example:-

management:
  endpoints:
```

(9) Error Analysis



Error Analysis

Moderate • Score **40/40**

Problem statement Send feedback

(Multiple Answer Question)
A developer has encountered an issue in a Spring Boot application that incorporates an Actuator for an Employee Management System. Despite implementing a custom Actuator endpoint at "localhost:8080/actuator/employeeIds", the endpoint does not appear in the list of Actuator endpoints after accessing "localhost:8080/actuator." .
The getEmployee() method fetches a list of all employees from the repository. It returns a map of **Employee** with the employee's name and ID as its key and value, respectively, with a datatype of String.
The provided code snippet for the **Actuator** class is suspected to contain an issue. Please identify and rectify the problem in the code given below

```
@Component
public class Actuator {

    @Autowired
    EmployeeRepository repository;

    Map<String, String> data = new HashMap<>();

    public Map getEmployeeIds(){
        List<Employee> employees= repository.findAll();
        for (Employee employee:employees){
            data.put(employee.getName(), employee.getId());
        }
        return data;
    }
```

Options: One or more answers may be correct   Attempts left: 1/2

☑
```
The endpoint annotation with id for the Actuator class is missing.
 @Endpoint(id = "employeeIds")
```
✓

☐ b) @WriteOperation annotation for the Actuator class is missing.

☐
```
The endpoint annotation with name for the class is missing.
 @Endpoint(name = "employeeIds")
```

☑ @ReadOperation annotation for getEmployeeIds() is missing. ✓

Solution description

Answer - the correct answers are options A and D.

Explanation -

1.

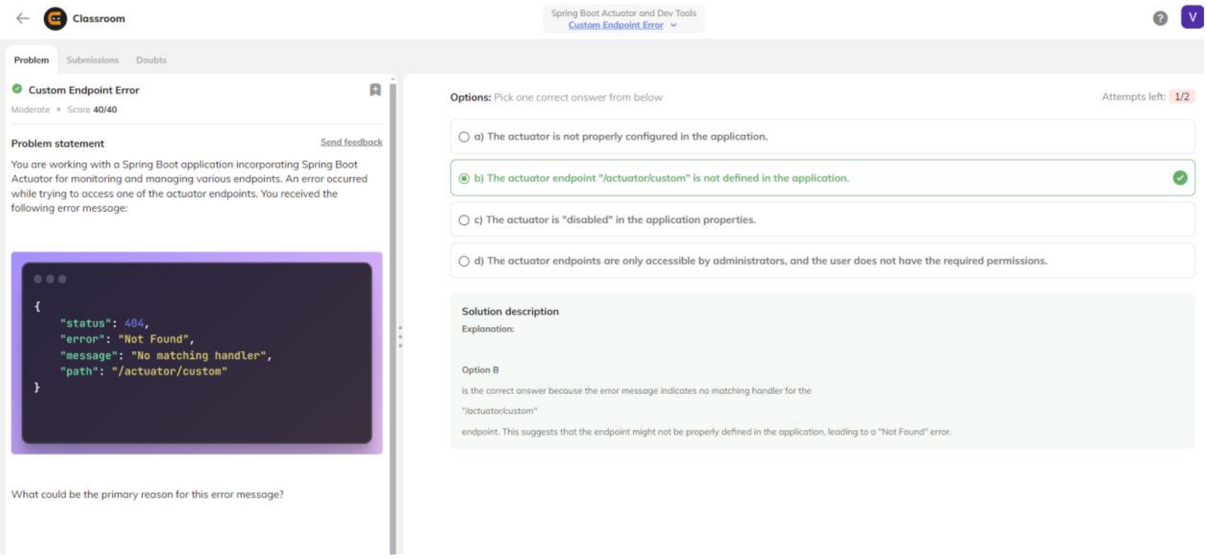
Option A

is the correct option as it states that the

@Endpoint(id = "employeeIds")

annotation is missing from the Actuator class. This annotation is essential to define the Actuator endpoint with a specific identifier ("employeeIds"). Spring Boot will recognise this class as an Actuator endpoint with this annotation. 2. Option B is not the right option, as

(10) Custom Endpoint Error



Custom Endpoint Error

Moderate • Score **40/40**

Problem statement Send feedback

You are working with a Spring Boot application incorporating Spring Boot Actuator for monitoring and managing various endpoints. An error occurred while trying to access one of the actuator endpoints. You received the following error message:

```
{
    "status": 404,
    "error": "Not Found",
    "message": "No matching handler",
    "path": "/actuator/custom"
}
```

What could be the primary reason for this error message?

Options: Pick one correct answer from below   Attempts left: 1/2

○ a) The actuator is not properly configured in the application.

● b) The actuator endpoint "/actuator/custom" is not defined in the application. ✓

○ c) The actuator is "disabled" in the application properties.

○ d) The actuator endpoints are only accessible by administrators, and the user does not have the required permissions.

Solution description

Explanation:

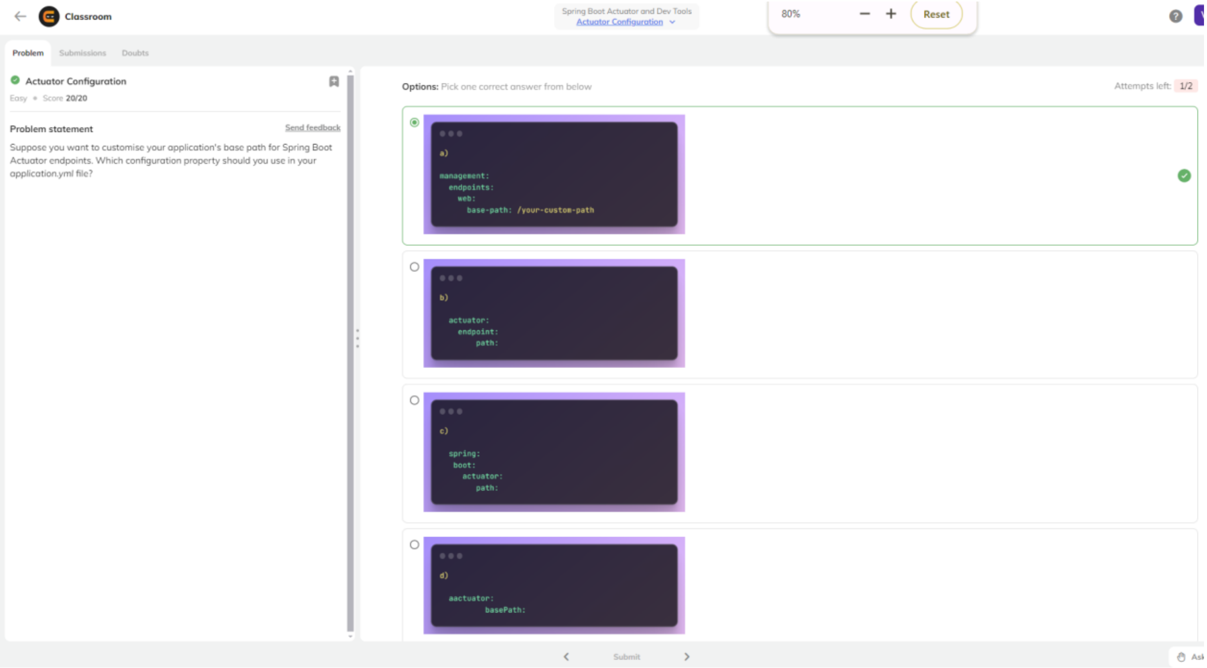
Option B

is the correct answer because the error message indicates no matching handler for the

"/actuator/custom"

endpoint. This suggests that the endpoint might not be properly defined in the application, leading to a "Not Found" error.

(11) Actuator Configuration



Actuator Configuration

Easy • Score 20/20

Problem statement

Suppose you want to customise your application's base path for Spring Boot Actuator endpoints. Which configuration property should you use in your application.yml file?

Options: Pick one correct answer from below

Attempts left: 1/2

● ✓

```
a)

management:
  endpoints:
    web:
      base-path: /your-custom-path
```

○

```
b)

  actuator:
    endpoint:
        path:
```

○

```
c)

  spring:
   boot:
     actuator:
        path:
```

○

```
d)

  aactuator:
          basePath:
```

(12)