**Exercise 6: Test Service Exception Handling**

**Task:** Test how a service handles a missing user. Test: Write code for this.

**UserService.java :-**

@Service

public class UserService {

@Autowired

private UserRepository userRepository;

public User getUserById(Long id) {

return userRepository.findById(id)

.orElseThrow(() -> new UserNotFoundException("User with ID " + id + " not found"));

}

}

**UserNotFoundException.java:-**

public class UserNotFoundException extends RuntimeException {

public UserNotFoundException(String message) {

super(message);

}

}

**UserServiceTest.java :-**

package com.example.demo;

import com.example.demo.exception.UserNotFoundException;

import com.example.demo.model.User;

import com.example.demo.repository.UserRepository;

import com.example.demo.service.UserService;

import org.junit.jupiter.api.Test;

import org.junit.jupiter.api.extension.ExtendWith;

import org.mockito.InjectMocks;

import org.mockito.Mock;

import org.mockito.junit.jupiter.MockitoExtension;

import java.util.Optional;

import static org.junit.jupiter.api.Assertions.\*;

import static org.mockito.Mockito.\*;

@ExtendWith(MockitoExtension.class)

public class UserServiceTest {

@Mock

private UserRepository userRepository;

@InjectMocks

private UserService userService;

@Test

public void testUserNotFoundException() {

Long userId = 100L;

// Mocking repository to return empty

when(userRepository.findById(userId)).thenReturn(Optional.empty());

// Assert exception is thrown

UserNotFoundException exception = assertThrows(

UserNotFoundException.class,

() -> userService.getUserById(userId)

);

assertEquals("User with ID 100 not found", exception.getMessage());

// Verify that repository was called once

verify(userRepository, times(1)).findById(userId);

}

}

**Pom.xml :-**

<project xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0

http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>user-service</artifactId>

<version>1.0.0</version>

<packaging>jar</packaging>

<name>User Service App</name>

<properties>

<java.version>17</java.version>

<spring.boot.version>3.2.0</spring.boot.version>

</properties>

<dependencies>

<!-- Spring Boot Web -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<!-- Spring Data JPA -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>

<!-- H2 in-memory database -->

<dependency>

<groupId>com.h2database</groupId>

<artifactId>h2</artifactId>

<scope>runtime</scope>

</dependency>

<!-- Jackson (JSON mapping) -->

<dependency>

<groupId>com.fasterxml.jackson.core</groupId>

<artifactId>jackson-databind</artifactId>

</dependency>

<!-- Spring Boot Test (includes JUnit, Mockito, AssertJ) -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

<!-- Mockito Core (optional, if using manually) -->

<dependency>

<groupId>org.mockito</groupId>

<artifactId>mockito-core</artifactId>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<!-- Spring Boot Maven Plugin -->

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

<version>${spring.boot.version}</version>

</plugin>

<!-- Java Compiler Plugin with -parameters -->

<plugin>

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

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

<version>3.10.1</version>

<configuration>

<source>${java.version}</source>

<target>${java.version}</target>

<compilerArgs>

<arg>-parameters</arg>

</compilerArgs>

</configuration>

</plugin>

</plugins>

</build>

</project>

**Output :-**

