**Exercise 1 : Mocking a Service Dependency in a Controller Test**

**UserControllerTest.java**

import com.example.demo.controller.UserController;

import com.example.demo.model.User;

import com.example.demo.service.UserService;

import org.junit.jupiter.api.Test;

import org.mockito.InjectMocks;

import org.mockito.Mock;

import org.mockito.MockitoAnnotations;

import org.springframework.http.ResponseEntity;

import static org.junit.jupiter.api.Assertions.assertEquals;

import static org.mockito.Mockito.when;

public class UserControllerTest {

@Mock

private UserService userService;

@InjectMocks

private UserController userController;

public UserControllerTest() {

MockitoAnnotations.openMocks(this);

}

@Test

public void testGetUser(){

Long userId = 1L;

User mockUser = new User();

mockUser.setId(userId);

mockUser.setName("John Doe");

when(userService.getUserById(userId)).thenReturn(mockUser);

ResponseEntity<User> response = userController.getUser(userId);

assertEquals(200, response.getStatusCodeValue());

assertEquals("John Doe", response.getBody().getName());

assertEquals(userId, response.getBody().getId());

}

}

**Pom.xml**

<dependencies>

<dependency>

<groupId>org.junit.jupiter</groupId>

<artifactId>junit-jupiter</artifactId>

<version>5.10.0</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.mockito</groupId>

<artifactId>mockito-core</artifactId>

<version>5.10.0</version>

<scope>test</scope>

</dependency>

<dependency>

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

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

<version>3.2.0</version>

<scope>test</scope>

</dependency>

</dependencies>

**Exercise 2 : Mocking a Repository in Service Test**

**UserServiceTest.java**

package com.example.demo.service;

import com.example.demo.model.User;

import com.example.demo.repository.UserRepository;

import org.junit.jupiter.api.BeforeEach;

import org.junit.jupiter.api.Test;

import org.mockito.\*;

import java.util.Optional;

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

import static org.mockito.Mockito.\*;

public class UserServiceTest {

@Mock

private UserRepository userRepository;

@InjectMocks

private UserService userService;

@BeforeEach

public void setUp() {

MockitoAnnotations.openMocks(this); // Initialize @Mock and @InjectMocks

}

@Test

public void testGetUserById\_UserExists() {

Long userId = 1L;

User mockUser = new User();

mockUser.setId(userId);

mockUser.setName("Alice");

when(userRepository.findById(userId)).thenReturn(Optional.of(mockUser));

User result = userService.getUserById(userId);

assertNotNull(result);

assertEquals("Alice", result.getName());

assertEquals(userId, result.getId());

}

@Test

public void testGetUserById\_UserDoesNotExist() {

Long userId = 2L;

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

User result = userService.getUserById(userId);

assertNull(result);

}

}

**Exercise 3 : Mocking a Service Dependency in an Integration Test**

**UserIntegrationTest.java**

package com.example.demo;

import com.example.demo.model.User;

import com.example.demo.service.UserService;

import org.junit.jupiter.api.Test;

import org.mockito.Mockito;

import static org.mockito.Mockito.when;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.boot.test.autoconfigure.web.servlet.AutoConfigureMockMvc;

import org.springframework.boot.test.mock.mockito.MockBean;

import org.springframework.boot.test.context.SpringBootTest;

import org.springframework.http.MediaType;

import org.springframework.test.web.servlet.MockMvc;

import static org.springframework.test.web.servlet.request.MockMvcRequestBuilders.get;

import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.\*;

@SpringBootTest

@AutoConfigureMockMvc

public class UserIntegrationTest {

@Autowired

private MockMvc mockMvc;

@MockBean

private UserService userService;

@Test

public void testGetUserById() throws Exception {

Long userId = 1L;

User mockUser = new User();

mockUser.setId(userId);

mockUser.setName("Kavipriya");

when(userService.getUserById(userId)).thenReturn(mockUser);

mockMvc.perform(get("/users/{id}", userId)

.accept(MediaType.APPLICATION\_JSON))

.andExpect(status().isOk())

.andExpect(jsonPath("$.id").value(userId))

.andExpect(jsonPath("$.name").value("Kavipriya"));

}

}