**Exercise 5.6: Test Service Exception Handling**

**CODE:**

**User Class(Entity): -**

package org.example;  
import jakarta.persistence.Entity;  
  
@Entity  
public class User {  
 private Long Id;  
 private String name;  
 private String email;  
  
 public User() {  
 }  
 public User(Long id, String name, String email) {  
 Id = id;  
 this.name = name;  
 this.email = email;  
 }  
 public Long getId() {  
 return Id;  
 }  
 public void setId(Long id) {  
 Id = id;  
 }  
 public String getName() {  
 return name;  
 }  
 public void setName(String name) {  
 this.name = name;  
 }  
 public String getEmail() {  
 return email;  
 }  
 public void setEmail(String email) {  
 this.email = email;  
 }  
}

**UserRepository Interface: -**

package org.example;  
import java.util.Optional;  
  
public interface UserRepository {  
 Optional<User> findById(Long id);  
}

**UserController Class: -**

package org.example;  
  
import org.springframework.http.ResponseEntity;  
import org.springframework.web.bind.annotation.\*;  
  
@RestController  
@RequestMapping("/users")  
public class UserController {  
  
 private final UserService userService;  
  
 public UserController(UserService userService) {  
 this.userService = userService;  
 }  
  
 @GetMapping("/{id}")  
 public ResponseEntity<User> getUser(@PathVariable Long id) {  
 User user = userService.getUserById(id);  
 return ResponseEntity.*ok*(user);  
 }  
}

**UserService Interface: -**

package org.example;  
  
public interface UserService {  
 User getUserById(Long id);  
}

**UserServiceImpl Class: -**

package org.example;  
public class UserServiceImpl implements UserService {  
  
 private final UserRepository userRepository;  
  
 public UserServiceImpl(UserRepository userRepository) {  
 this.userRepository = userRepository;  
 }  
  
 @Override  
 public User getUserById(Long id) {  
 return userRepository.findById(id)  
 .orElseThrow(() -> new UserNotFoundException("User not found with ID: " + id));  
 }  
}

**UserServiceTest Class: -**

package org.example;  
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.*assertEquals*;  
import static org.junit.jupiter.api.Assertions.*assertThrows*;  
import static org.mockito.Mockito.*when*;  
  
@ExtendWith(MockitoExtension.class)  
public class UserServiceTest {  
  
 @Mock  
 private UserRepository userRepository;  
  
 @InjectMocks  
 private UserServiceImpl userService;  
  
 @Test  
 public void testGetUserById\_UserNotFound\_ShouldThrowException() {  
 Long missingUserId = 99L;  
  
 *when*(userRepository.findById(missingUserId)).thenReturn(Optional.*empty*());  
  
 UserNotFoundException exception = *assertThrows*(UserNotFoundException.class, () -> {  
 userService.getUserById(missingUserId);  
 });  
  
 *assertEquals*("User not found with ID: 99", exception.getMessage());  
 System.*out*.println("Test passed - exception thrown as expected.");  
 }  
}

**OUTPUT:**

