**Exercise 5.4: Integration Testing with SpringBoot**

**Code:**

**User.java Class(Entity): -**

package org.example;  
import jakarta.persistence.\*;  
  
@Entity  
@Table(name = "users")  
public class User {  
  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private Long id;  
  
 private String name;  
 private String email;  
  
 public User() {}  
 public User(String name, String email) {  
 this.name = name;  
 this.email = email;  
 }  
  
 public Long getId() { return id; }  
 public void setId(Long id) { this.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; }  
}

**UserService Class: -**

package org.example;  
  
import org.springframework.stereotype.Service;  
  
import java.util.Optional;  
  
@Service  
public class UserService {  
  
 private final UserRepository userRepository;  
 public UserService(UserRepository userRepository) {  
 this.userRepository = userRepository;  
 }  
  
 public User createUser(User user) {  
 return userRepository.save(user);  
 }  
  
 public Optional<User> getUserById(Long id) {  
 return userRepository.findById(id);  
 }  
}

**UserRepository Interface: -**

package org.example;  
import org.springframework.data.jpa.repository.JpaRepository;  
  
public interface UserRepository extends JpaRepository<User, Long> {  
}

**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;  
 }  
  
 @PostMapping  
 public ResponseEntity<User> createUser(@RequestBody User user) {  
 return ResponseEntity.*ok*(userService.createUser(user));  
 }  
  
 @GetMapping("/{id}")  
 public ResponseEntity<User> getUser(@PathVariable Long id) {  
 return userService.getUserById(id)  
 .map(ResponseEntity::*ok*)  
 .orElse(ResponseEntity.*notFound*().build());  
 }  
}

**UserIntegrationTest Class: -**

import com.fasterxml.jackson.databind.ObjectMapper;  
import org.example.User;  
import org.example.UserRepository;  
import org.junit.jupiter.api.Test;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.boot.test.autoconfigure.web.servlet.AutoConfigureMockMvc;  
import org.springframework.boot.test.context.SpringBootTest;  
import org.springframework.http.MediaType;  
import org.springframework.test.context.TestPropertySource;  
import org.springframework.test.web.servlet.MockMvc;  
  
import static org.springframework.test.web.servlet.request.MockMvcRequestBuilders.\*;  
import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.\*;  
  
@SpringBootTest  
@AutoConfigureMockMvc  
@TestPropertySource(locations = "classpath:application-test.properties")  
public class UserIntegrationTest {  
  
 @Autowired  
 private MockMvc mockMvc;  
  
 @Autowired  
 private UserRepository userRepository;  
  
 @Autowired  
 private ObjectMapper objectMapper;  
  
 @Test  
 public void testCreateAndFetchUser() throws Exception {  
 User user=new User("Alice Reddy", "alice@example.com");  
  
 //Creating user(POST)  
 String response=mockMvc.perform(*post*("/users")  
 .contentType(MediaType.*APPLICATION\_JSON*)  
 .content(objectMapper.writeValueAsString(user)))  
 .andExpect(*status*().isOk())  
 .andExpect(*jsonPath*("$.name").value("Alice Reddy"))  
 .andExpect(*jsonPath*("$.email").value("alice@example.com"))  
 .andReturn().getResponse().getContentAsString();  
 System.*out*.println("User created successfully.Raw JSON response: "+response);  
 User createdUser = objectMapper.readValue(response, User.class);  
  
 //Fetching user(GET)  
 mockMvc.perform(*get*("/users/" + createdUser.getId()))  
 .andExpect(*status*().isOk())  
 .andExpect(*jsonPath*("$.id").value(createdUser.getId()))  
 .andExpect(*jsonPath*("$.name").value("Alice Reddy"))  
 .andExpect(*jsonPath*("$.email").value("alice@example.com"));  
 System.*out*.println("User fetched and validated successfully!");  
 }  
}

**OUTPUT: -**

