**Spring Testing  
  
 1.Basic unit test for a Service Method**

**CalculatorService.java**

@Service

public class CalculatorService {

public int add(int a, int b) {

return a + b;

}

}

**CalculatorServiceTest.java**

public class CalculatorServiceTest {

@Test

public void testAdd() {

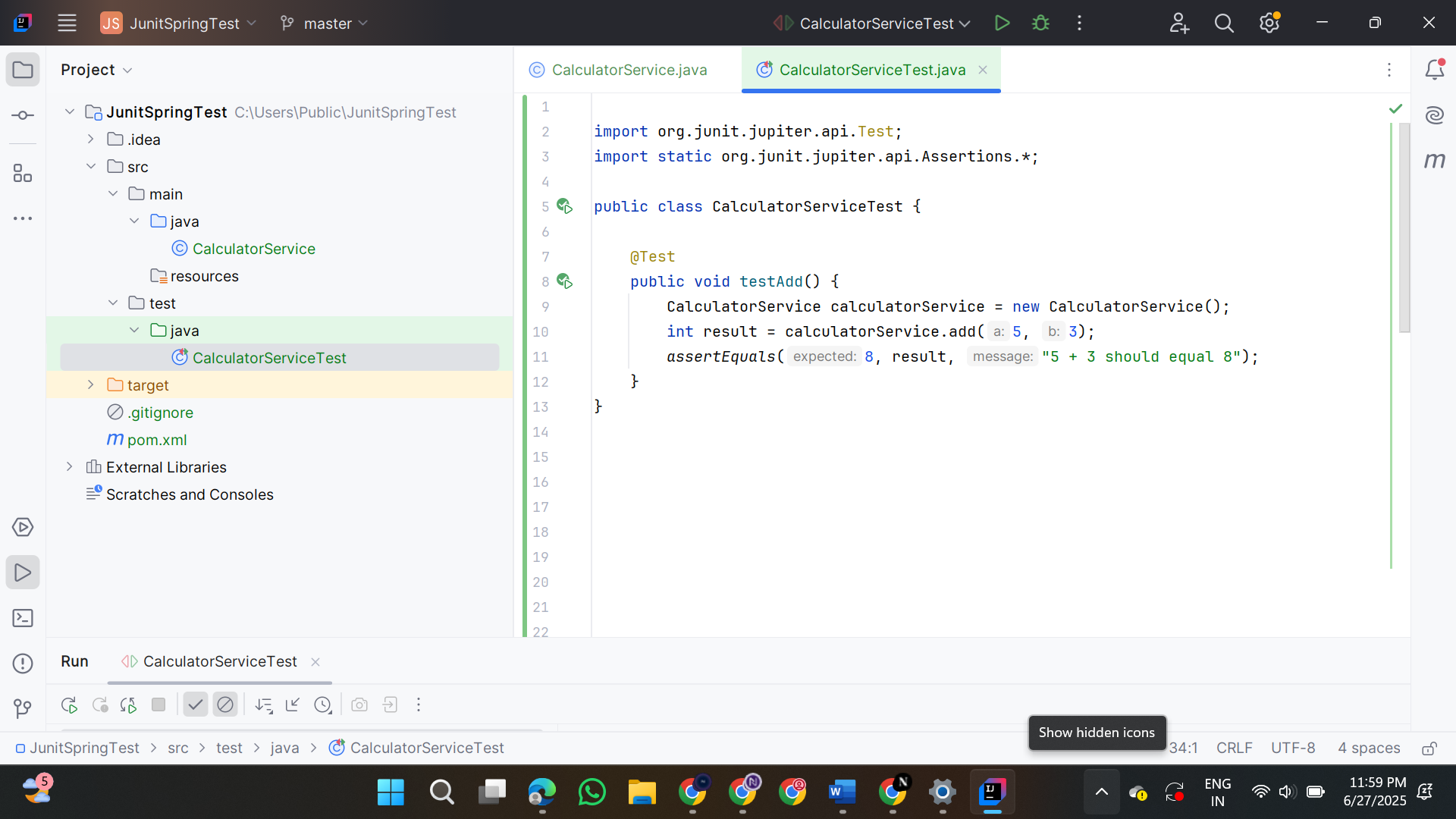
CalculatorService calculatorService = new CalculatorService();

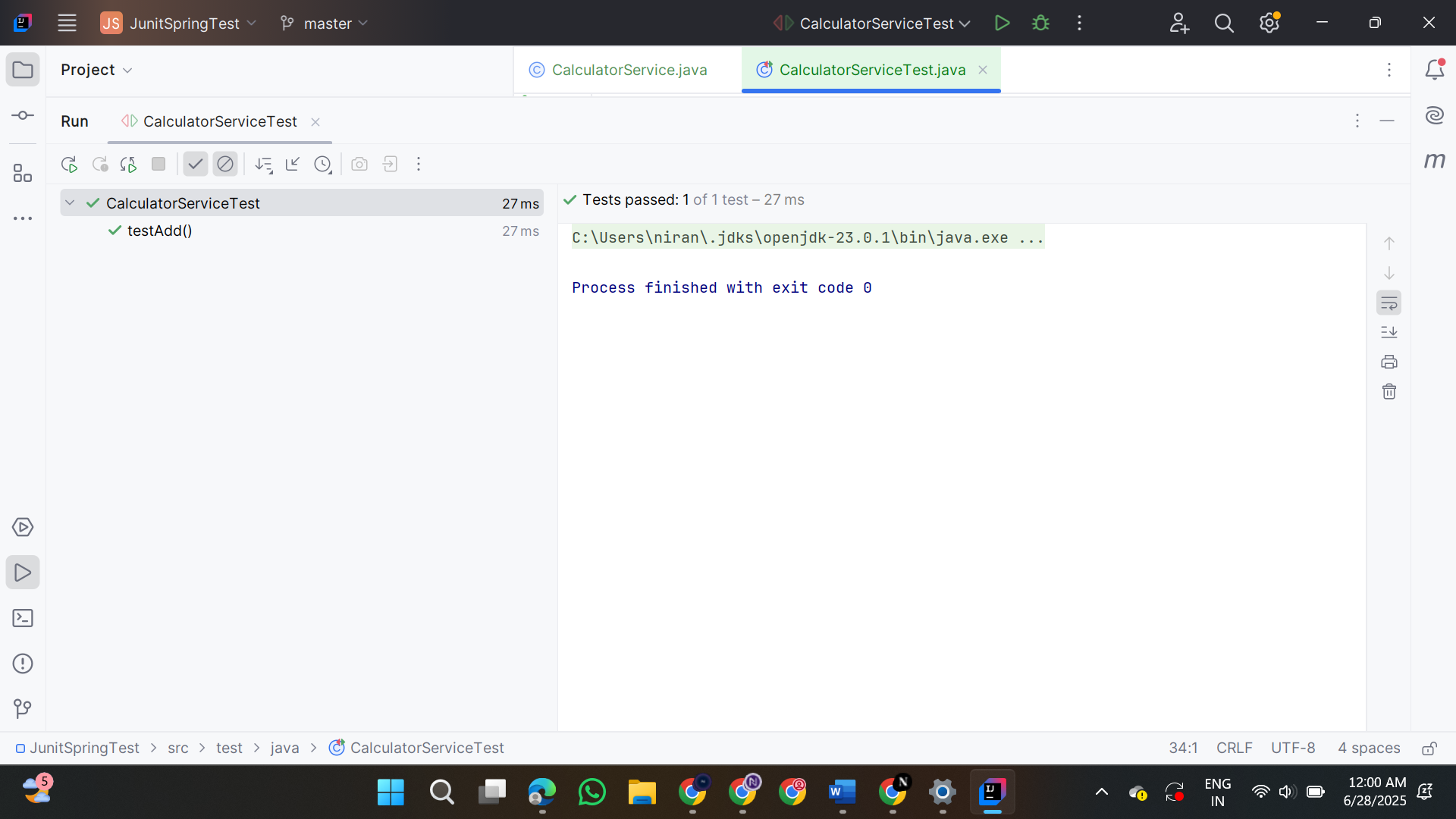
int result = calculatorService.add(5, 3);

assertEquals(8, result, "5 + 3 should equal 8");

}

}





**2.Basic unit test for a Service Method**

**User.java**

@Entity

public class User {

@Id

private Long id;

private String name;

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;

}

}

**UserRepository.java**

package com.calculator.repository;

import com.calculator.model.User;

import org.springframework.data.jpa.repository.JpaRepository;

public interface UserRepository extends JpaRepository<User, Long> {

}

**UserService.java**

import com.calculator.model.User;

import com.calculator.repository.UserRepository;

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

import org.springframework.stereotype.Service;

@Service

public class UserService {

@Autowired

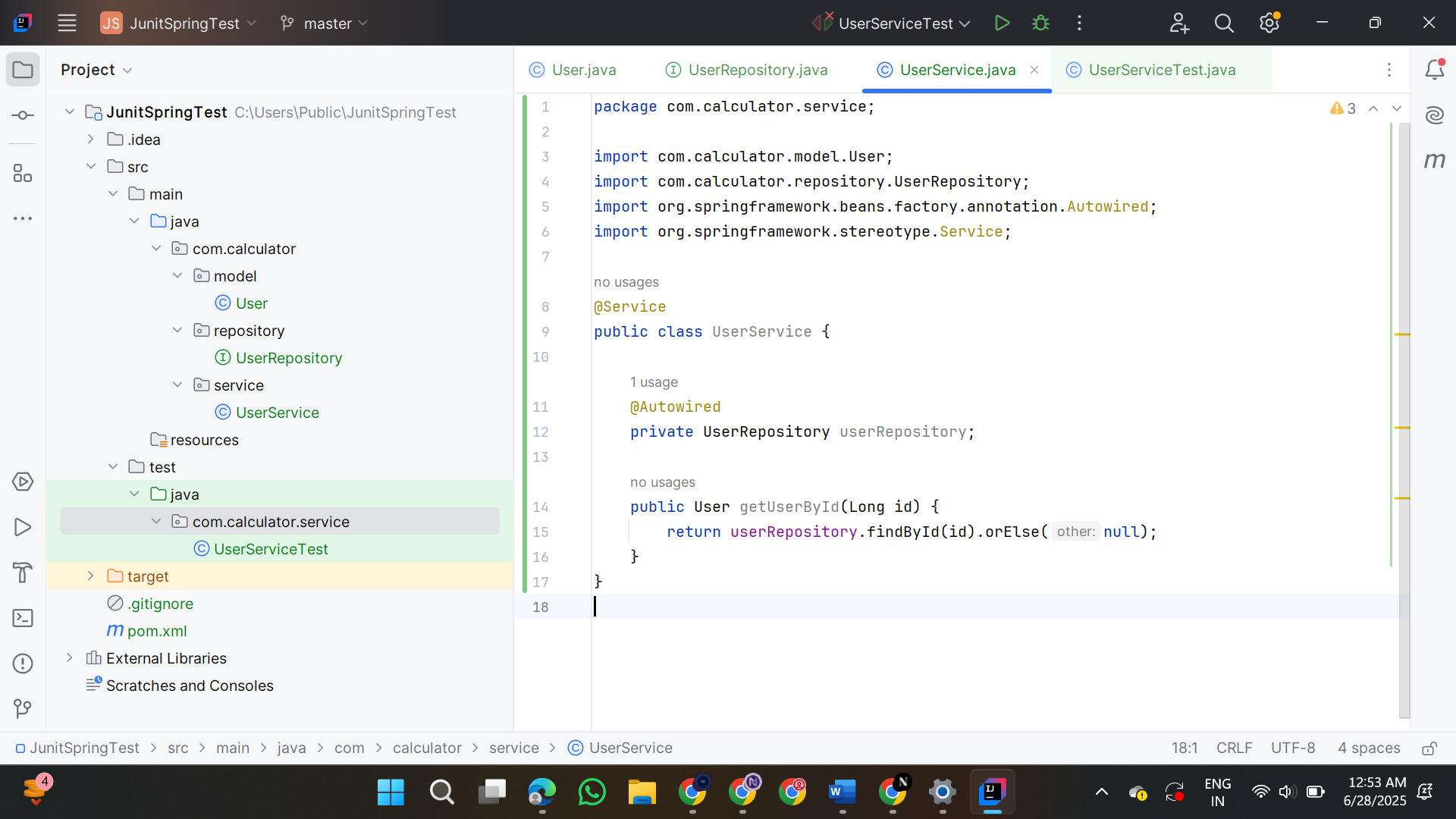
private UserRepository userRepository;

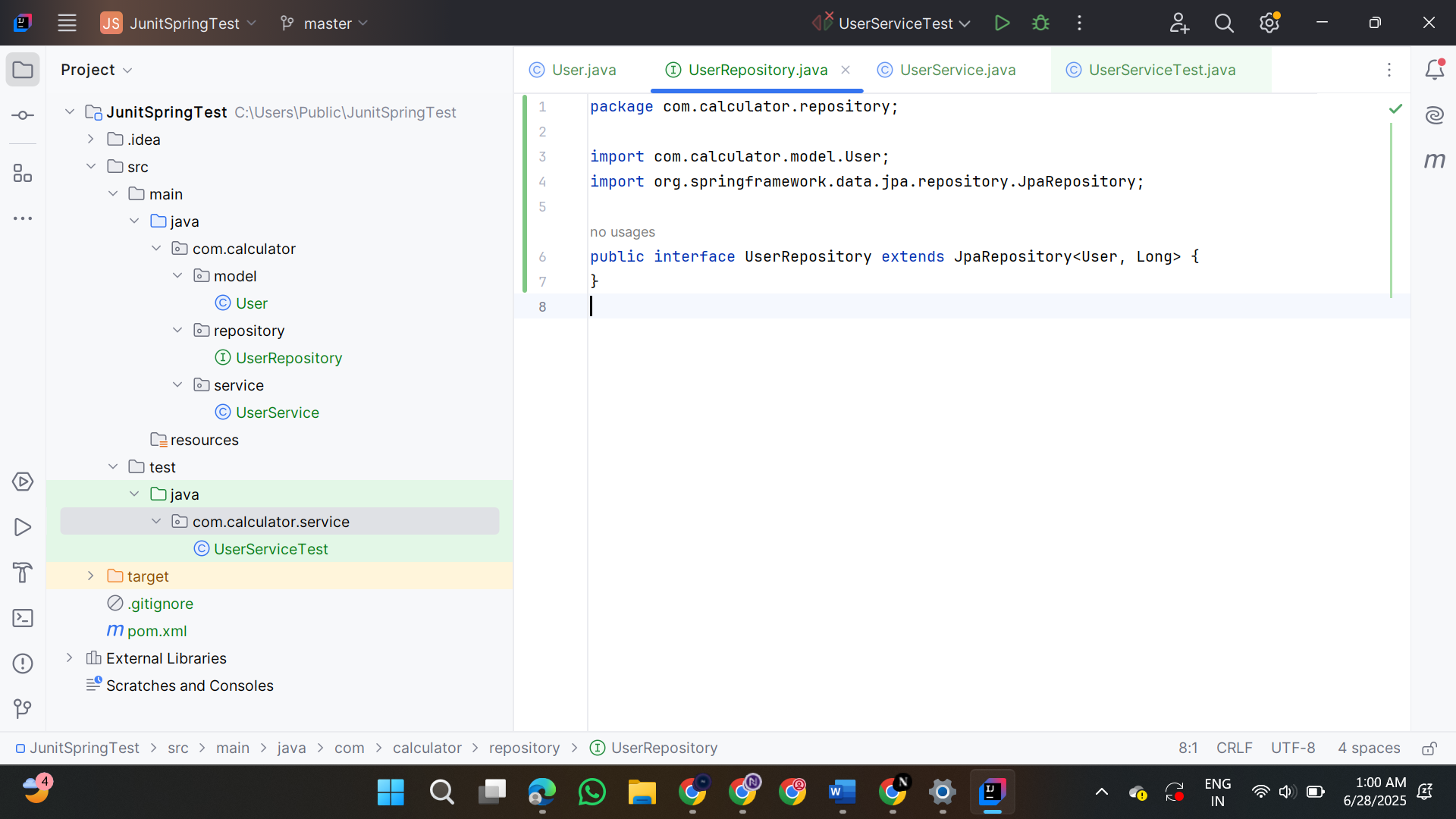
public User getUserById(Long id) {

return userRepository.findById(id).orElse(null);

}

}





**3.Testing a REST controller with Mockmvc**

**UserServiceTest.java**

import com.calculator.model.User;

import com.calculator.repository.UserRepository;

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

void testGetUserById\_ReturnsUser() {

Long id = 1L;

User user = new User();

user.setId(id);

user.setName("Alice");

when(userRepository.findById(id)).thenReturn(Optional.of(user));

User result = userService.getUserById(id);

assertNotNull(result);

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

verify(userRepository).findById(id);

}

@Test

void testGetUserById\_ReturnsNullWhenNotFound() {

Long id = 2L;

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

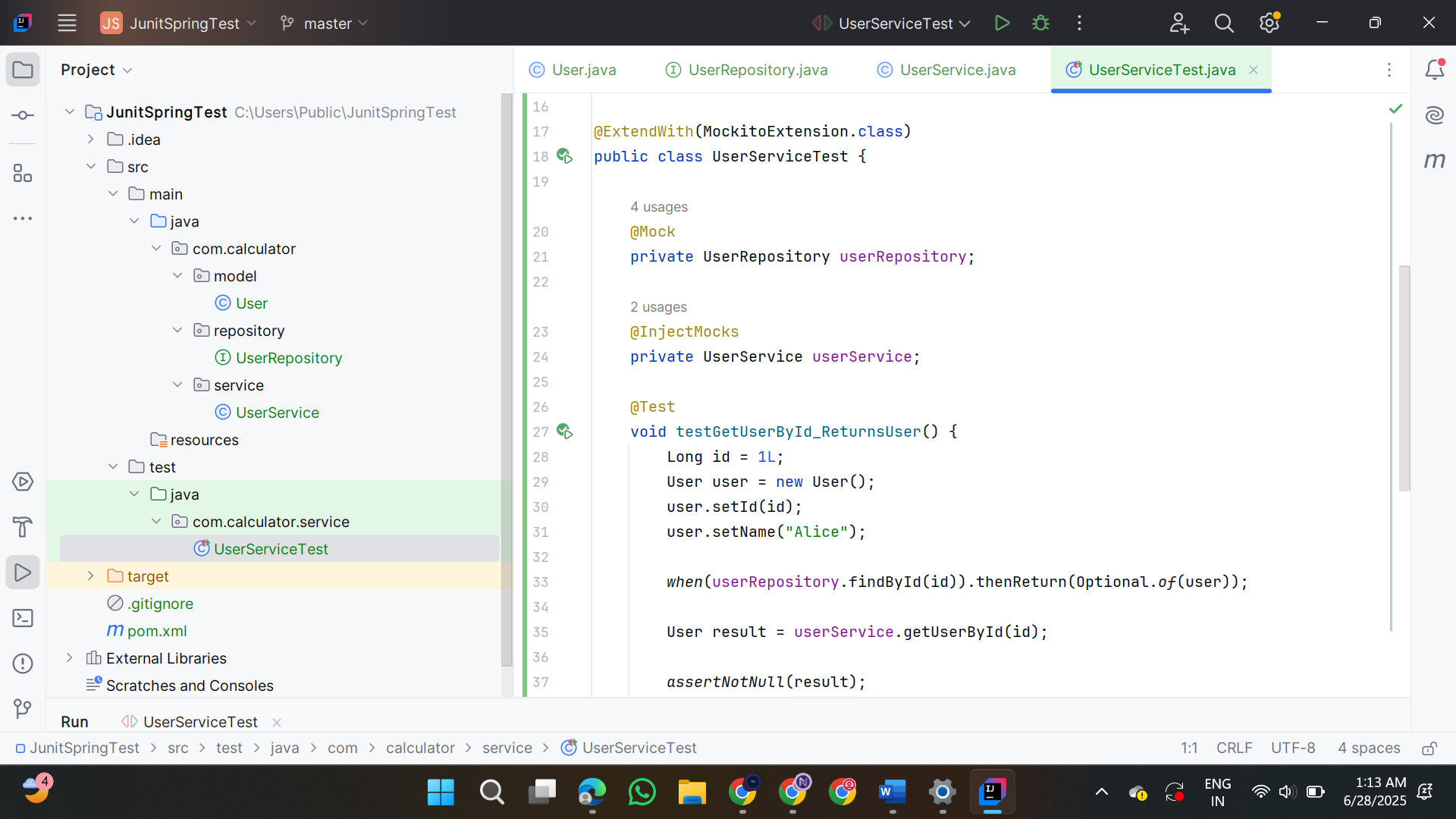
User result = userService.getUserById(id);

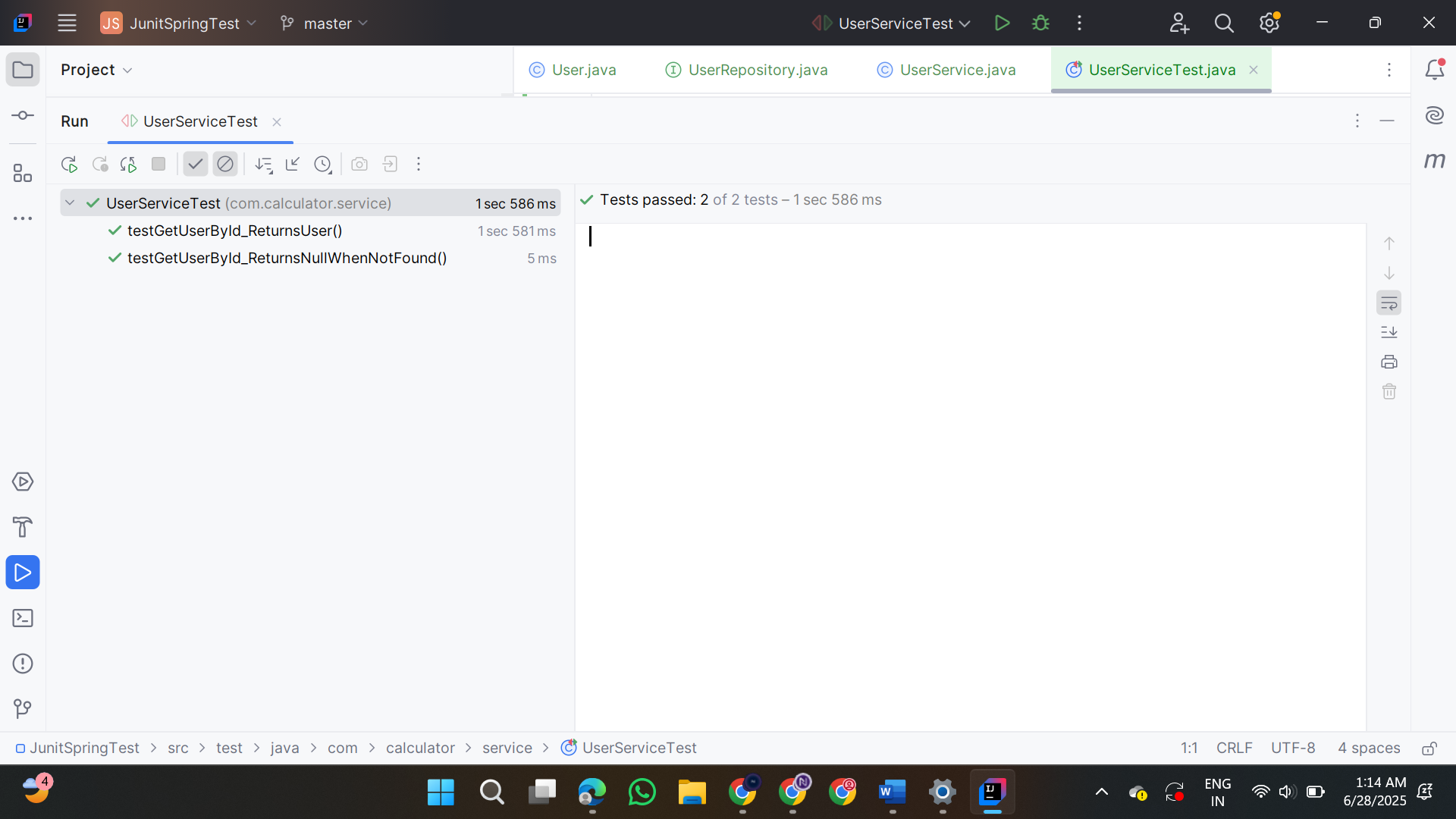
assertNull(result);

verify(userRepository).findById(id);

}

}





**4.Integration test with Spring boot**

**UserIntegrationTest.java**

@SpringBootTest(webEnvironment =

SpringBootTest.WebEnvironment.RANDOM\_PORT)

public class UserIntegrationTest {

@LocalServerPort

private int port;

@Autowired

private TestRestTemplate restTemplate;

@Autowired

private UserRepository userRepository;

@Test

public void testSaveAndGetUser() {

User user = new User();

user.setName("Test User");

HttpHeaders headers = new HttpHeaders();

headers.setContentType(MediaType.APPLICATION\_JSON);

HttpEntity<User> request = new HttpEntity<>(user, headers);

ResponseEntity<User> response = restTemplate.postForEntity("http://localhost:" + port + "/api/users", request, User.class);

assertThat(response.getStatusCode()).isEqualTo(HttpStatus.OK);

assertThat(response.getBody()).isNotNull();

assertThat(response.getBody().getName()).isEqualTo("Test User");

Long id = response.getBody().getId();

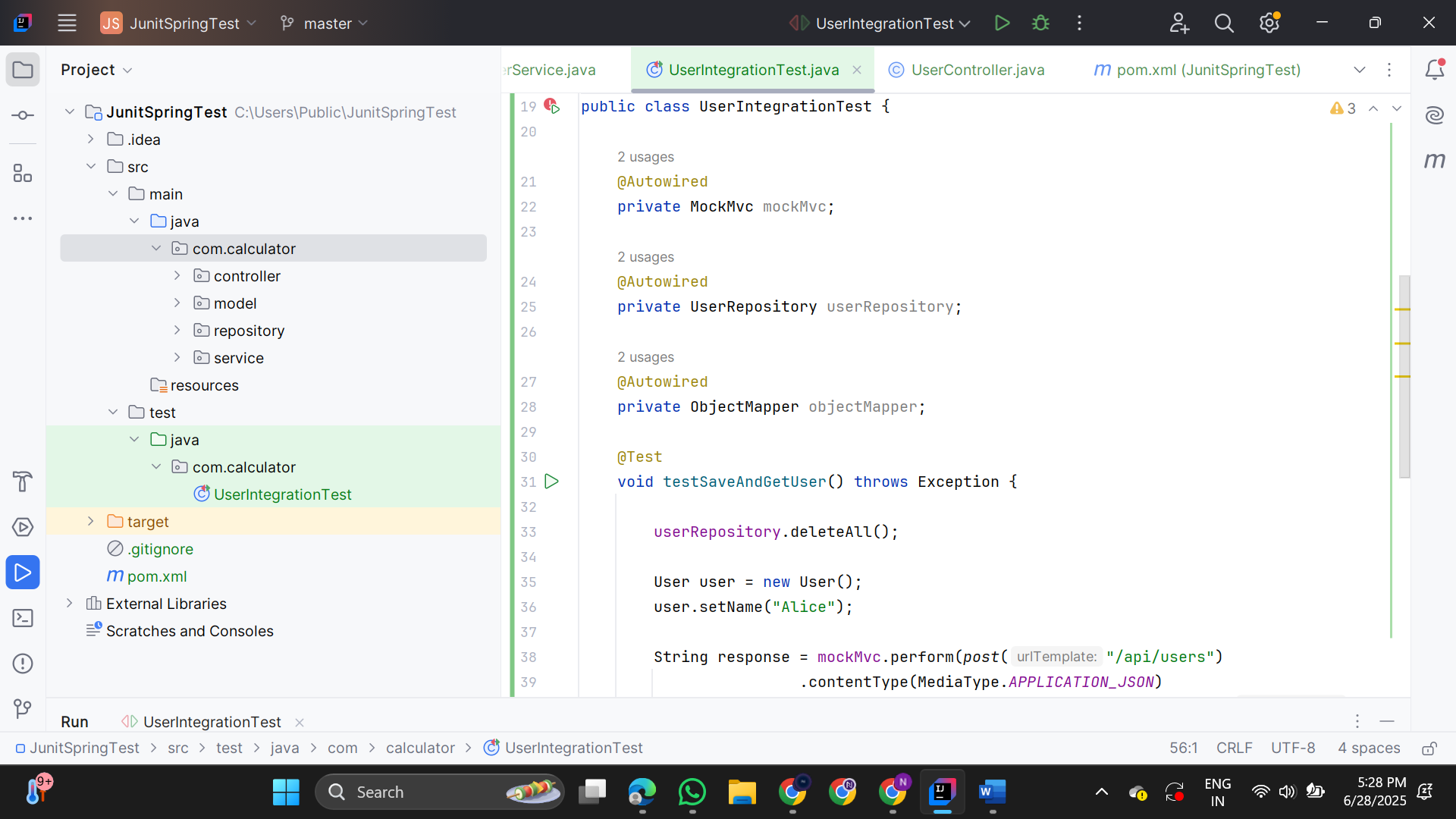
User retrieved = restTemplate.getForObject("http://localhost:" + port + "/api/users/" + id, User.class);

assertThat(retrieved).isNotNull();

assertThat(retrieved.getName()).isEqualTo("Test User");

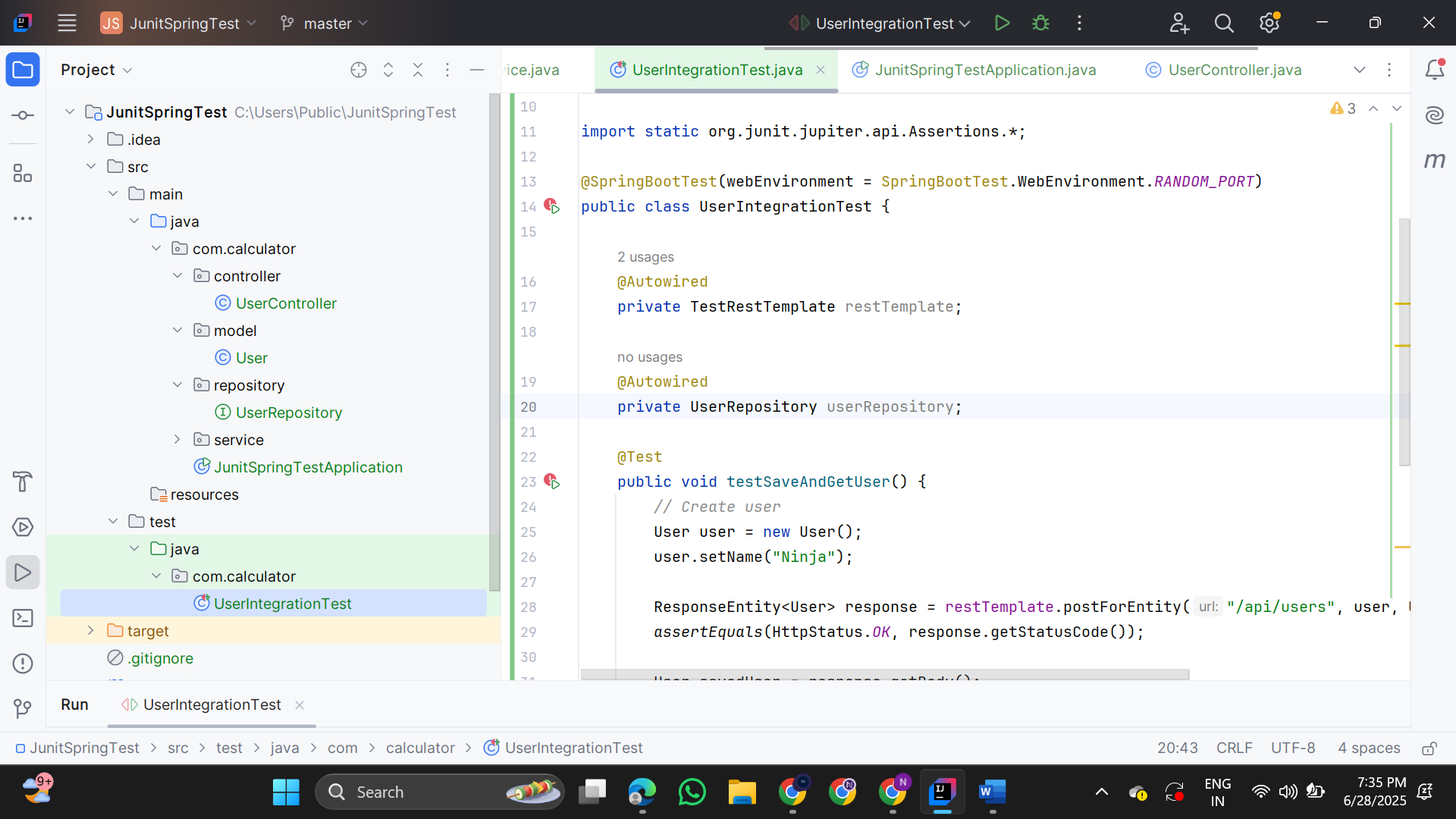
}

}



**5.Test Controller POST Endpoint**

**UserIntegrationTest.java**  
import com.calculator.model.User;  
import com.calculator.repository.UserRepository;  
import org.junit.jupiter.api.Test;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.boot.test.context.SpringBootTest;  
import org.springframework.boot.test.web.client.TestRestTemplate;  
import org.springframework.http.\*;  
  
import static org.junit.jupiter.api.Assertions.\*;  
  
@SpringBootTest(webEnvironment = SpringBootTest.WebEnvironment.RANDOM\_PORT)  
public class UserIntegrationTest {  
  
 @Autowired  
 private TestRestTemplate restTemplate;  
  
 @Autowired  
 private UserRepository userRepository;  
  
 @Test  
 public void testSaveAndGetUser() {  
   
 User user = new User();  
 user.setName("Ninja");  
  
 ResponseEntity<User> response = restTemplate.postForEntity("/api/users", user, User.class);  
 assertEquals(HttpStatus.OK, response.getStatusCode());  
  
 User savedUser = response.getBody();  
 assertNotNull(savedUser);  
 assertNotNull(savedUser.getId());  
 assertEquals("Ninja", savedUser.getName());  
  
   
 ResponseEntity<User> fetched = restTemplate.getForEntity("/api/users/" + savedUser.getId(), User.class);  
 assertEquals(HttpStatus.OK, fetched.getStatusCode());  
 assertEquals("Ninja", fetched.getBody().getName());  
 }  
}



**6.Test Service Exception Handling**

**UserIntegrationTest.java**

@SpringBootTest  
@AutoConfigureMockMvc  
public class UserIntegrationTest {

@Autowired

private MockMvc mockMvc;

@Autowired

private UserRepository userRepository;

@Test

public void testSaveAndGetUser() throws Exception {

String json = "{\"name\": \"John\"}";

mockMvc.perform(post("/api/users")

.contentType(MediaType.APPLICATION\_JSON)

.content(json))

.andExpect(status().isOk())

.andExpect(jsonPath("$.id").exists())

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

}

}

**UserControllerTest.java**

@WebMvcTest(UserController.class)  
public class UserControllerTest {

@Autowired

private MockMvc mockMvc;

@MockBean

private UserService userService;

@Autowired

private ObjectMapper objectMapper;

@Test

public void createUser\_returnsOk() throws Exception {

User user = new User("TestUser");

user.setId(1L);

when(userService.saveUser(user)).thenReturn(user);

mockMvc.perform(post("/api/users")

.contentType(MediaType.APPLICATION\_JSON)

.content(objectMapper.writeValueAsString(user)))

.andExpect(status().isOk())

.andExpect(jsonPath("$.id").value(1L))

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

}

}

**UserServiceExceptionTest.java**

public class UserServiceExceptionTest {

csharp

CopyEdit

@Mock

private UserRepository userRepository;

@InjectMocks

private UserService userService;

public UserServiceExceptionTest() {

MockitoAnnotations.openMocks(this);

}

@Test

public void getUserById\_userNotFound\_throwsException() {

Long userId = 1L;

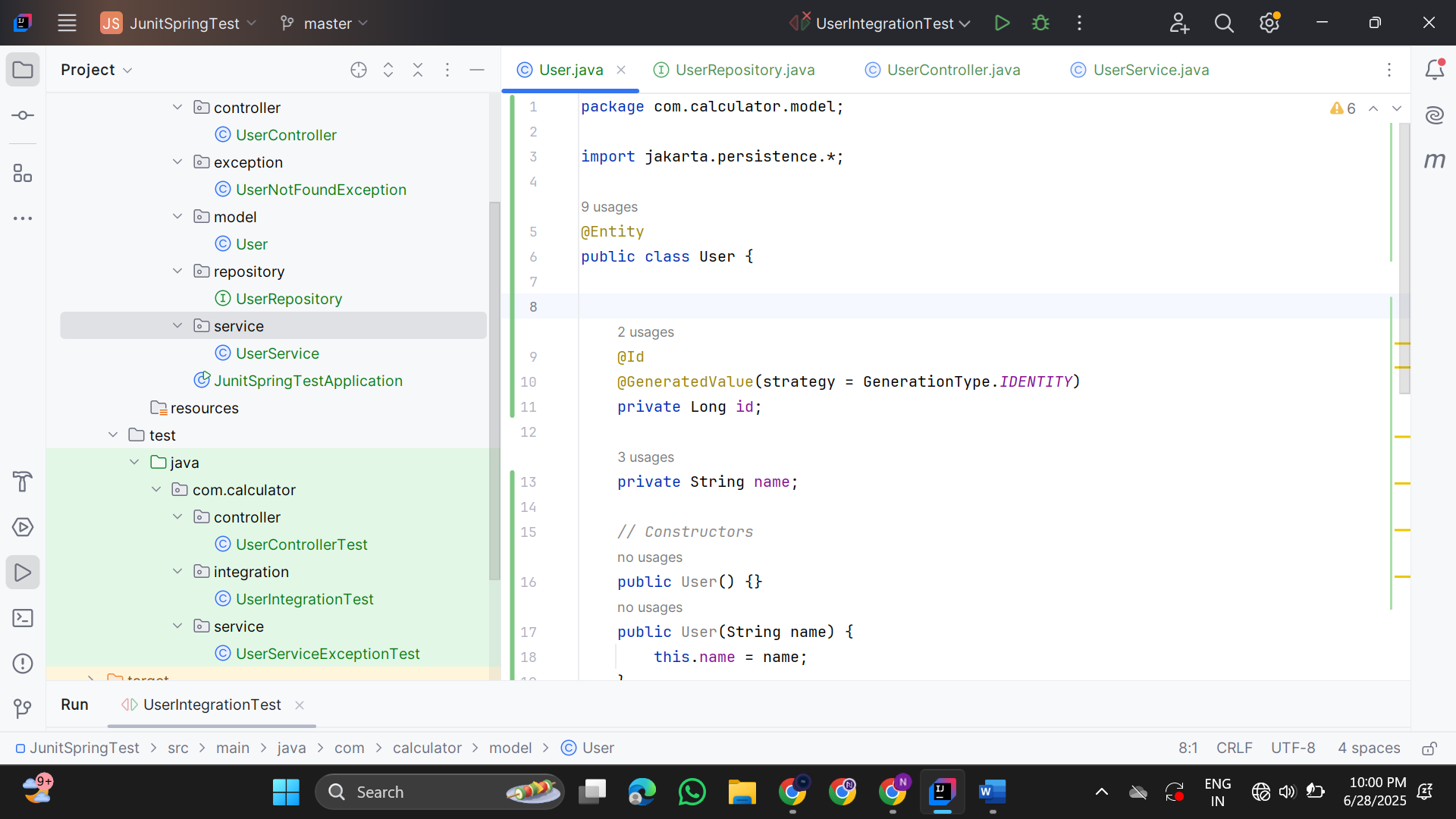
when(userRepository.findById(userId)).thenReturn(Optional.empty());

assertThrows(UserNotFoundException.class, () -> {

userService.getUserById(userId);

});

}



**6.Test Custom Repository Query**

**UserRepositoryTest.java**

import com.example.demo.model.User;

import org.junit.jupiter.api.BeforeEach;

import org.junit.jupiter.api.Test;

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

import org.springframework.boot.test.autoconfigure.orm.jpa.DataJpaTest;

import java.util.List;

import static org.assertj.core.api.Assertions.assertThat;

@DataJpaTest

public class UserRepositoryTest {

@Autowired

private UserRepository userRepository;

@BeforeEach

public void setUp() {

userRepository.save(new User(null, "Alice"));

userRepository.save(new User(null, "Bob"));

userRepository.save(new User(null, "Alice"));

}

@Test

public void testFindByName() {

List<User> usersNamedAlice = userRepository.findByName("Alice");

assertThat(usersNamedAlice).hasSize(2);

assertThat(usersNamedAlice).allMatch(user -> user.getName().equals("Alice"));

}

}

