ky aap mjh crud operation ka junit test cases likh sakte he spring boot main

Haan, main aapko CRUD operation ke liye JUnit test cases likh sakta hu. Neeche diye gaye code snippets mein, ek sample Spring Boot project hai jo MongoDB ke saath CRUD operations karta hai. Aap is project se CRUD operation ke liye JUnit test cases ko samajh sakte hai.

1. Create Operation Test Case:

```java
@RunWith(SpringRunner.class)
@SpringBootTest
@AutoConfigureMockMvc
public class UserControllerTest {

    private MockMvc mockMvc;

    @Autowired
    private ObjectMapper objectMapper;

    @Autowired
    private UserRepository userRepository;

    @Autowired
    private WebApplicationContext webApplicationContext;

    @BeforeEach
    public void setUp() {
        mockMvc = MockMvcBuilders.webAppContextSetup(webApplicationContext).build();
    }

    @Test
    public void testCreateUser() throws Exception {
        // Create User Request
        User user = new User("testuser", "testemail@test.com", "Test User");
        String requestBody = objectMapper.writeValueAsString(user);

        // Send POST request to create user
        MvcResult result = mockMvc.perform(post("/users")
                .contentType(MediaType.APPLICATION_JSON)
                .content(requestBody))
                .andExpect(status().isCreated())
                .andReturn();

        // Verify User is Created
        String responseString = result.getResponse().getContentAsString();
        User createdUser = objectMapper.readValue(responseString, User.class);

        assertThat(createdUser.getId()).isNotEmpty();
        assertThat(createdUser.getUsername()).isEqualTo(user.getUsername());
        assertThat(createdUser.getEmail()).isEqualTo(user.getEmail());
        assertThat(createdUser.getName()).isEqualTo(user.getName());

        // Verify User is saved in Database
        Optional<User> optionalUser = userRepository.findById(createdUser.getId());
        assertThat(optionalUser).isPresent();

        User savedUser = optionalUser.get();
        assertThat(savedUser.getUsername()).isEqualTo(user.getUsername());
        assertThat(savedUser.getEmail()).isEqualTo(user.getEmail());
        assertThat(savedUser.getName()).isEqualTo(user.getName());
    }
}
```

2. Read Operation Test Case:

```java
@Test
public void testGetUserById() throws Exception {
    // Create User
    User user = new User("testuser", "testemail@test.com", "Test User");
    userRepository.save(user);

    // Send GET request to get user by Id
    MvcResult result = mockMvc.perform(get("/users/" + user.getId()))
            .andExpect(status().isOk())
            .andReturn();

    // Verify User is returned
    String responseString = result.getResponse().getContentAsString();
    User returnedUser = objectMapper.readValue(responseString, User.class);

    assertThat(returnedUser.getId()).isEqualTo(user.getId());
    assertThat(returnedUser.getUsername()).isEqualTo(user.getUsername());
    assertThat(returnedUser.getEmail()).isEqualTo(user.getEmail());
    assertThat(returnedUser.getName()).isEqualTo(user.getName());
}
```

3. Update Operation Test Case:

```java
@Test
public void testUpdateUser() throws Exception {
    // Create User
    User user = new User("testuser", "testemail@test.com", "Test User");
    userRepository.save(user);

    // Update User Request
    User updatedUser = new User("newtestuser", "newtestemail@test.com", "New Test
    String requestBody = objectMapper.writeValueAsString(updatedUser);

    // Send PUT request to update user
    MvcResult result = mockMvc.perform(put("/users/" + user.getId())
            .contentType(MediaType.APPLICATION_JSON)
            .content(requestBody))
            .andExpect(status().isOk())
            .andReturn();

    // Verify User is Updated
    String responseString = result.getResponse().getContentAsString();
    User returnedUser = objectMapper.readValue(responseString, User.class);

    assertThat(returnedUser.getId()).isEqualTo(user.getId());
    assertThat(returnedUser.getUsername()).isEqualTo(updatedUser.getUsername());
    assertThat(returnedUser.getEmail
```

continue

()).isEqualTo(updatedUser.getEmail());
assertThat(returnedUser.getName()).isEqualTo(updatedUser.getName());

```java
// Verify User is updated in Database
Optional<User> optionalUser = userRepository.findById(user.getId());
assertThat(optionalUser).isPresent();

User savedUser = optionalUser.get();
assertThat(savedUser.getUsername()).isEqualTo(updatedUser.getUsername());
assertThat(savedUser.getEmail()).isEqualTo(updatedUser.getEmail());
assertThat(savedUser.getName()).isEqualTo(updatedUser.getName());
```

}

```sql
4. Delete Operation Test Case:

```java
@Test
public void testDeleteUser() throws Exception {
    // Create User
    User user = new User("testuser", "testemail@test.com", "Test User");
    userRepository.save(user);

    // Send DELETE request to delete user
    mockMvc.perform(delete("/users/" + user.getId()))
            .andExpect(status().isOk());

    // Verify User is deleted from Database
    Optional<User> optionalUser = userRepository.findById(user.getId());
    assertThat(optionalUser).isEmpty();
}
```

Yeh code snippets sample JUnit test cases hai, jo MongoDB ke saath CRUD operations karne ke liye Spring Boot mein likhe gaye hai. Iska upyog karke aap apane project ke liye JUnit test cases likh sakte hai.
```