

Lab9-Report

Tests List

1. In our CI/CD, test part contains Unit Tests, Integration Tests (Controller tests), javadoc test.

```
run-unit-tests:
  stage: test
  script:
    - "mvn $MAVEN_CLI_OPTS -Dmaven.main.skip=true -Dtest=!lsea/controllers/* test -DfailIfNoTests=false"
  needs:
    - validate-project

run-api-tests:
  stage: test
  script:
    - "mvn $MAVEN_CLI_OPTS -Dmaven.main.skip=true -Dtest=lsea/controllers/* test"
  needs:
    - validate-project

test-javadoc:
  stage: test
  script:
    - mvn javadoc:javadoc
    - test -d target/site/apidocs/
  needs:
    - validate-project
```

2. For tests, it contains
 1. Non-trivial tests
 2. No cookie (Controller)
 3. No content (Controller)
 4. Invalid token (Controller)
 5. Different special cases if exist

Controller tests

Business Logic	Test Name	Test Result	Test Description
Index Controller	sanityCheck	200	This test only have this response, which response to check the sanity.
Log Controller - CreateOne	testCreateOne	200	Successfully create a log.
Log Controller - CreateOne	testCreateOneWithoutCookie	403 "No cookies found"	Forbid when no cookie.
Log Controller - CreateOne	testCreateOneWithoutContent	422 "Something went wrong"	UnprocessableEntity when no content.
Log Controller - CreateOne	testCreateOneWithInvalidToken	403 "Invalid token"	Forbid when token is invalid.

Business Logic	Test Name	Test Result	Test Description
Log Controller - GenerateData	testGenerateDataIsOK	200	Successfully generate data.
Log Controller - GenerateData	testGenerateDataNoContent()	422 "Something went wrong"	UnprocessableEntity when no content.
Log Controller - GenerateData	testGenerateDataNoCookie	403 "No cookies found"	Forbid when no cookie.
Log Controller - GenerateData	testGenerateDataInvalidToken	403 "Invalid token"	Forbid when token is invalid.
Management Controller - Analysis Longest Log	testAnalysisLongestIsOK	200	Successfully analysis longest log.
Management Controller - Analysis Longest Log	testAnalysisLongestNoContent	422 "Something went wrong"	UnprocessableEntity when no content.
Management Controller - Analysis Longest Log	testAnalysisLongestNoCookie	403 "No cookies found"	Forbid when no cookie.
Management Controller - Analysis Longest Log	testAnalysisLongestInvalidToken	403 "Invalid token"	Forbid when token is invalid.
Management Controller - Analysis Shortest Log	testAnalysisShortestIsOK	200	Successfully analysis shortest log.
Management Controller - Analysis Shortest Log	testAnalysisShortestNoContent	422 "Something went wrong"	UnprocessableEntity when no content.
Management Controller - Analysis Shortest Log	testAnalysisShortestNoCookie	403 "No cookies found"	Forbid when no cookie.
Management Controller - Analysis Shortest Log	testAnalysisShortestInvalidToken	403 "Invalid token"	Forbid when token is invalid.
User Controller - ping pong	testPing	200	This test only have this response, which is used for client to test if the server is alive.

Business Logic	Test Name	Test Result	Test Description
User Controller - Create User	testCreateUserIsOK	200	Successfully create a user.
User Controller - Create User	testCreateUserWithInvalidEmail	403 "A validation error occurred: email must match \"^[a-z0-9_+.]+@[a-z0-9-]+\.\.[a-z0-9-]+\.\$\""	Forbid when email is invalid.
User Controller - Create User	testAlreadyExistingUser	409 "User with email xxxxx already exists."	Conflict when user already exists.
User Controller - Authorize User	testAuthorizeUser	200	Successfully authorize a user.
User Controller - Authorize User	testAuthorizeUserWithWrongEmailOrPassword	403 "Invalid e-mail or password"	Forbid when email or password is wrong.
User Controller - Ban User	testBanUserAdminBan	200	Successfully ban a user.
User Controller - Ban User	testBanUserNotAdminBan	403	Forbid when user is not admin.
User Controller - Unban User	testUnbanAdminUnban	200	Successfully unban a user.
User Controller - Unban User	testUnbanNotAdminUnban	403	Forbid when user is not admin.
User Controller - Update User	testUpdateUserIsOK	200 - check user updated	Successfully update a user.
User Controller - Update User	testUpdateUserInvalidToken	403 "Invalid token"	Forbid when token is invalid.
User Controller - Update User	testUpdateUserNoContent	422 "Something went wrong"	UnprocessableEntity when no content.
User Controller - Update User	testUpdateUserNoCookie	403 "No cookies found"	Forbid when no cookie.
Website Controller - Create one	testCreateOneIsOK	200	Successfully create a website.
Website Controller - Create one	testCreateOneNoContent	422 "Something went wrong"	UnprocessableEntity when no content.
Website Controller - Create one	testCreateOneNoCookie	403 "No cookies found"	Forbid when no cookie.
Website Controller - Create one	testCreateOneInvalidToken	403 "Invalid token"	Forbid when token is invalid.

Business Logic	Test Name	Test Result	Test Description
Website Controller - Delete one	testDeleteOnelsOK	200	Successfully delete a website.
Website Controller - Delete one	testDeleteOneNoCookie	403 "No cookies found"	Forbid when no cookie.
Website Controller - Delete one	testDeleteOneInvalidToken	403 "Invalid token"	Forbid when token is invalid.
Website Controller - Delete one	testDeleteOneNoWebsiteFound	404 "Website not found for this user"	Not found when this user doesn't contains this website.
Website Controller - Delete one	testDeleteOneNotOwner	403 "User has no permission to delete this website"	Forbid when this user is not the owner of this website.

Other Business Logic Unit Tests

Business Logic	Test Name	Test Situation
Log - create log	testCreateLog	assertEquals(createLogDto.getData(), log.getData());
Log - create log	testCreateLog	assertEquals(LogType.Success, log.getLogType());
Log - create log	testCreateLog	assertEquals(user.getId(), log.getUserId());
Log - create log	testCreateLog	assertNotNull(log.getCreatedAt());
Log - create log	testCreateLog	assertEquals(user.toJson(), log.getUserCurrentState());
Log - dto - create log	testCreateLogWithInvalidLogType	assertThrows
UserGroup - create user group	testCreateUserGroup	Assertions.assertNotNull(userGroup.getId());
UserGroup - create user group	testCreateUserGroup	Assertions.assertEquals(dto.getName(), userGroup.getName());
UserGroup - create user group	testCreateUserGroup	Assertions.assertEquals(dto.getDescription(), userGroup.getDescription());
UserGroup - create user group	testCreateUserGroup	Assertions.assertEquals(GlobalPermissions.USER, userGroup.getGlobalPermission());

Business Logic	Test Name	Test Situation
UserGroup - create user group	testCreateUserGroup	Assertions.assertNotNull(userGroup.getCreatedAt());
UserGroup - dto - create user group	testCreateUserGroupThrowsGenericForbiddenError()	assertThrows
UserGroup - To Json	testToJson	Not Null
UserGroupUser - Create	testCreate	Assertions.assertNotNull(userGroupUser);
UserGroupUser - Create	testCreate	Assertions.assertNotNull(userGroupUser.getId());
UserGroupUser - Create	testCreate	Assertions.assertEquals(UUID.fromString(userGroupId), userGroupUser.getUserGroupId());
UserGroupUser - Create	testCreate	Assertions.assertEquals(UUID.fromString(userId), userGroupUser.getUserId());
UserGroupUser - Create	testCreate	Assertions.assertEquals(GroupPermissions.REGULAR, userGroupUser.getGroupPermission());
UserGroupUser - dto - Create	testCreateWithInsufficientPermissions	assertThrows
UserGroupUser - dto - Create	testCreateWithSpectatorPermissions	assertThrows
User - Create	testCreate	Assertions.assertEquals(dto.getUsername(), createdUser.getUsername());
User - Create	testCreate	Assertions.assertEquals(dto.getEmail(), createdUser.getEmail());
User - Create	testCreate	Assertions.assertNotEquals(dto.getPassword(), createdUser.getPassword());
User - Ban	testBan	Assertions.assertNotNull(user.getBannedAt());
User - Ban	testBan	Assertions.assertEquals(adminId, user.getBannedById());
User - Ban	testBan	Assertions.assertEquals(banReason, user.getBanReason());
User - UnBan	testUnBan	Assertions.assertNull(user.getBannedAt());
User - UnBan	testUnBan	Assertions.assertNull(user.getBannedById());
User - UnBan	testUnBan	Assertions.assertNull(user.getBanReason());
User - isBanned	testIsBanned	Assertions.assertTrue(user.isBanned());
User - Change Password	testChangePassword	assertThrows verifyPassword

Business Logic	Test Name	Test Situation
User - Verify Password	testVerifyPassword	Assertions.assertDoesNotThrow() -> user.verifyPassword(correctPassword);
User - Verify Password	testVerifyPassword	Assertions.assertThrows(GenericForbiddenError.class, () -> user.verifyPassword(incorrectPassword));
User - jwt token	testGetJwtToken	Assertions.assertNotNull(jwtToken);
User - jwt token	testVerifyToken	Assertions.assertEquals(user.getId(), userId);
User - jwt token	testVerifyToken	Assertions.assertThrows(GenericForbiddenError.class, () -> User.verifyToken("invalid_token"));
User - to json	testToJson	Assertions.assertNotNull(user.toJson());
Website - Create	testCreateWebsite	assertNotNull(website.getId());
Website - Create	testCreateWebsite	assertEquals(createWebsiteDto.getDisplayName(), website.getDisplayName());
Website - Create	testCreateWebsite	assertEquals(user, website.getUser());
Website - Create	testCreateWebsite	assertNotNull(website.getCreatedAt());
Website - Create	testCreateWebsite	assertEquals(createWebsiteDto.getRedirectUrl(), website.getRedirectUrl());
Website - Create	testCreateWebsite	assertNotNull(website.getPrivateKey());
Website - Create	testCreateWebsite	assertFalse(website.getIsActive());
Website - Create	testCreateWebsite	assertEquals(website.getCreatedAt(), website.getUpdatedAt());
Website - deep clone	testDeepClone	assert for each field

Special statement

Test should avoid loops instructions. We have two places have loops, but

- one is used for deep clone test to check each needed field has been deeply cloned.
- another one is used for generate different lengths of data to check the logic of analyzing longest/shortest logs

Other notes

Mock

Use mock when testing with business components.

Using mocks in Java testing is a common practice when testing business components, particularly in unit testing scenarios. A mock object is a simulated object that mimics the behavior of a real object, allowing you to isolate the component under test and verify its interactions with dependencies. - Isolation - Control and Predictability - Speed and Efficiency - Reproducible Tests - Test Independence and Stability

Arrange-Act-Assert

Tests follow **Arrange-Act-Assert** pattern which is commented as `// Arrange`, `// Act`, `//Assert` in the code.

Example code:

Here is part of codes from `ManagementControllerTest.java`:

```
@RunWith(SpringRunner.class)
@SpringBootTest(classes = {LaboratoryApplication.class}, webEnvironment =
SpringBootTest.WebEnvironment.RANDOM_PORT)
@AutoConfigureMockMvc
@Transactional
public class ManagementControllerTest {

    /**
     * This is the MockMvc object that is used to send requests to the endpoints.
     */
    @Resource
    private MockMvc mockMvc;

    /**
     * This test method sends a Get request to the "/api/v1/management/analysis-longest-five"
     *
     * @throws Exception Exception of mockMvc.perform
     */
    @Test
    @DisplayName("Test of ManagementController")
    @Rollback
    public void testAnalysisLongestIsOK() throws Exception {
        // Arrange
        String userRequest = "{\n" +
            "    \"username\": \"test\",\n" +
            "    \"password\": \"12345678\",\n" +
            "    \"email\": \"123@11.com\"\n" +
            "}";

        String userAuthRequest = "{\n" +
            "    \"password\": \"12345678\",\n" +
            "    \"email\": \"123@11.com\"\n" +
            "}";

        mockMvc
            .perform(
                MockMvcRequestBuilders
                    .post("/api/v1/users")
                    .contentType("application/json;charset=UTF-8")
                    .content(userRequest))
            .andReturn();

        MvcResult result = mockMvc
            .perform(
                MockMvcRequestBuilders
                    .post("/api/v1/users/authorize")
                    .contentType("application/json;charset=UTF-8")
                    .content(userAuthRequest))
            .andReturn();

        Cookie cookie = result.getResponse().getCookie("token");

        for (int i = 0; i < 10; i++) {
```

```

        StringBuilder logData = new StringBuilder("test data");
        for (int j = 0; j < i; j++) {
            logData.append(i);
        }
        String logRequest = "{\n" +
            "    \"data\": \"\" +
            logData +
            "\",\n" +
            "    \"logType\": 0\n" +
            "}";
        MockHttpServletRequestBuilder requestBuilder = MockMvcRequestBuilders
            .post("/api/v1/logs")
            .cookie(cookie)
            .contentType("application/json;charset=UTF-8")
            .content(logRequest);

        mockMvc.perform(requestBuilder)
            .andExpect(MockMvcResultMatchers.print())
            .andReturn();
    }

    userAuthRequest = "{\n" +
        "    \"password\": \"test_admin\",\n" +
        "    \"email\": \"test_admin@example.com\"\n" +
        "}";

    result = mockMvc.perform(
        MockMvcRequestBuilders
            .post("/api/v1/users/authorize")
            .contentType("application/json;charset=UTF-8")
            .content(userAuthRequest)
            .andReturn());

    cookie = result.getResponse().getCookie("token");

    MockHttpServletRequestBuilder requestBuilder = MockMvcRequestBuilders
        .get("/api/v1/management/analysis-longest-five")
        .param("numThreads", "5")
        .cookie(cookie)
        .contentType("application/json;charset=UTF-8");

    /* Requirement 9 */
    // isOK
    // Act
    mockMvc.perform(requestBuilder)
        // Assert
        .andExpect(MockMvcResultMatchers.status().isOk())
        .andExpect(MockMvcResultMatchers
            .jsonPath("$.data[0].data")
            .value("test data5555"))
        .andExpect(MockMvcResultMatchers
            .jsonPath("$.data[4].data")
            .value("test data999999999"))
        .andExpect(MockMvcResultMatchers.print());
    }
}

```

Why we still have @Transactional and @Rollback here:

It is because in this test class, there are two tests which are `testAnalysisLongestIsOK` and `testAnalysisShortestIsOK`, they will share the same `mockMvc` bean instance. They can pass the test independently, but if we run them together in the CI/CD, the

second test will fail because the data in the mock instance has been changed by the first test. So we need to rollback the data after each test. Or we can arrange the tests before running them in the other method.

Appendix - Test Result From CI/CD

run-api-tests

```
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 37, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 01:11 min
[INFO] Finished at: 2023-06-04T13:33:26Z
[INFO] -----
section_end:1685885606:step_script
[0Ksection_start:1685885606:archive_cache
[0K[0K[36;1mSaving cache for successful job[0;m[0;m
[32;1mCreating cache VERY_COOL_KEY-7...[0;m
.m2/repository: found 5033 matching artifact files and directories[0;m
Archive is up to date! [0;m
[32;1mCreated cache[0;m
section_end:1685885607:archive_cache
[0Ksection_start:1685885607:cleanup_file_variables
[0K[0K[36;1mCleaning up project directory and file based variables[0;m[0;m
section_end:1685885608:cleanup_file_variables
[0K[32;1mJob succeeded[0;m
```

run-unit-tests

```
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 21, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 50.931 s
[INFO] Finished at: 2023-06-04T13:31:54Z
[INFO] -----
section_end:1685885514:step_script
[0Ksection_start:1685885514:archive_cache
[0K[0K[36;1mSaving cache for successful job[0;m[0;m
[32;1mCreating cache VERY_COOL_KEY-7...[0;m
.m2/repository: found 5033 matching artifact files and directories[0;m
Archive is up to date! [0;m
[32;1mCreated cache[0;m
section_end:1685885515:archive_cache
[0Ksection_start:1685885515:cleanup_file_variables
[0K[0K[36;1mCleaning up project directory and file based variables[0;m[0;m
section_end:1685885515:cleanup_file_variables
[0K[32;1mJob succeeded[0;m
```

test-javadoc

```
[INFO] -----  
[INFO] BUILD SUCCESS  
[INFO] -----  
[INFO] Total time: 17.058 s  
[INFO] Finished at: 2023-06-04T13:34:04Z  
[INFO] -----  
$ test -d target/site/apidocs/  
Saving cache for successful job  
00:01  
Creating cache VERY_COOL_KEY-7...  
.m2/repository: found 5033 matching artifact files and directories  
Archive is up to date!  
Created cache  
Cleaning up project directory and file based variables  
00:01  
Job succeeded
```