

# Test Report

🕒 Created	@October 21, 2021 6:33 PM
🕒 Last Edited Time	@January 16, 2022 11:48 PM
📄 Type	Technical Spec
📄 Status	
👤 Created By	
👤 Last Edited By	
👥 Stakeholders	

[1. Introduction](#)

[2. Test Plan](#)

[2.1 Testing Strategy](#)

[2.2 Test Subjects](#)

[2.3 Black-Box Testing](#)

[2.4 White-Box Testing](#)

[3. Additional Tests](#)

[4. Test Results](#)

[4.1 Detailed Test Report](#)

[4.2 Detailed Test Results](#)

## 1. Introduction

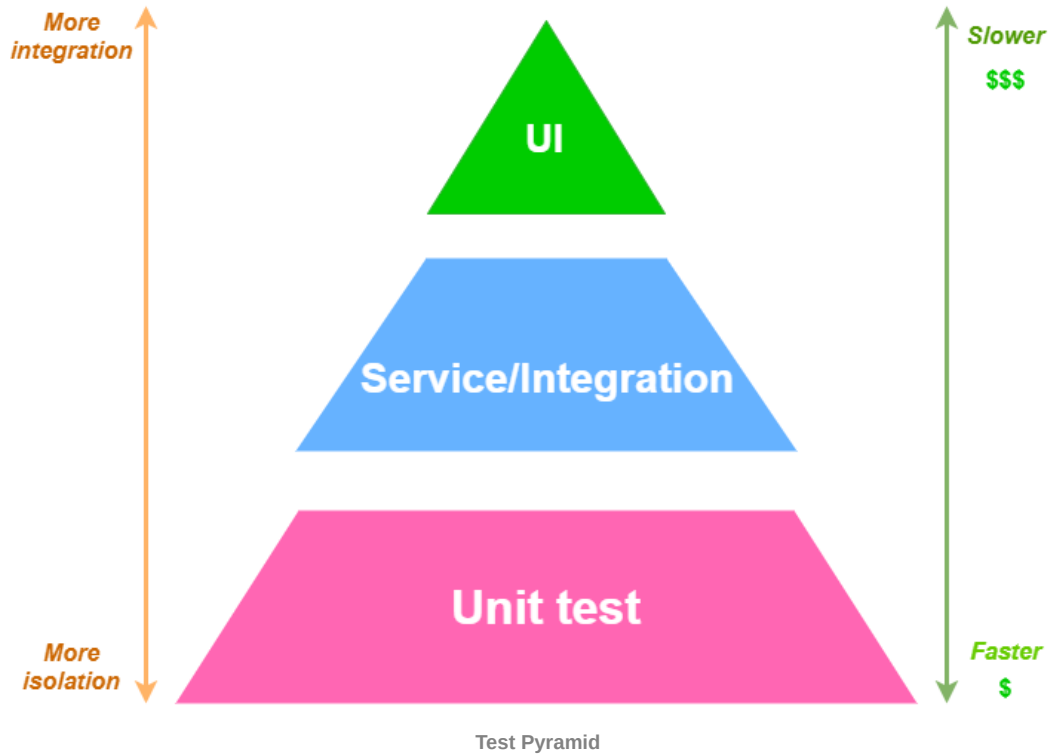
This document is dedicated to explain the project test plan and test results. In the next sections, we will mention about the testing methodologies we have implemented and the motivation for choosing them. Also, the test cases we have implemented and their results will be mentioned.

## 2. Test Plan

### 2.1 Testing Strategy

To choose the testing strategy, we based the Test Pyramid showing the relationship between different testing strategies. Our initial plan was to implement integration tests between separate micro-services. However, since the correctness of unit modules and classes according to the Test Pyramid must be ensured, we did the unit tests before the integration tests. We also thought that unit testing would increase the potential for success for other high-level testing strategies such as service/integration testing, UI testin.

In brief, for the purpose of applying integration tests, we started to implement unit tests for our service classes and methods. However, we could only implement the unit tests due to limited time.



## 2.2 Test Subjects

We did not apply integration testing.

## 2.3 Black-Box Testing

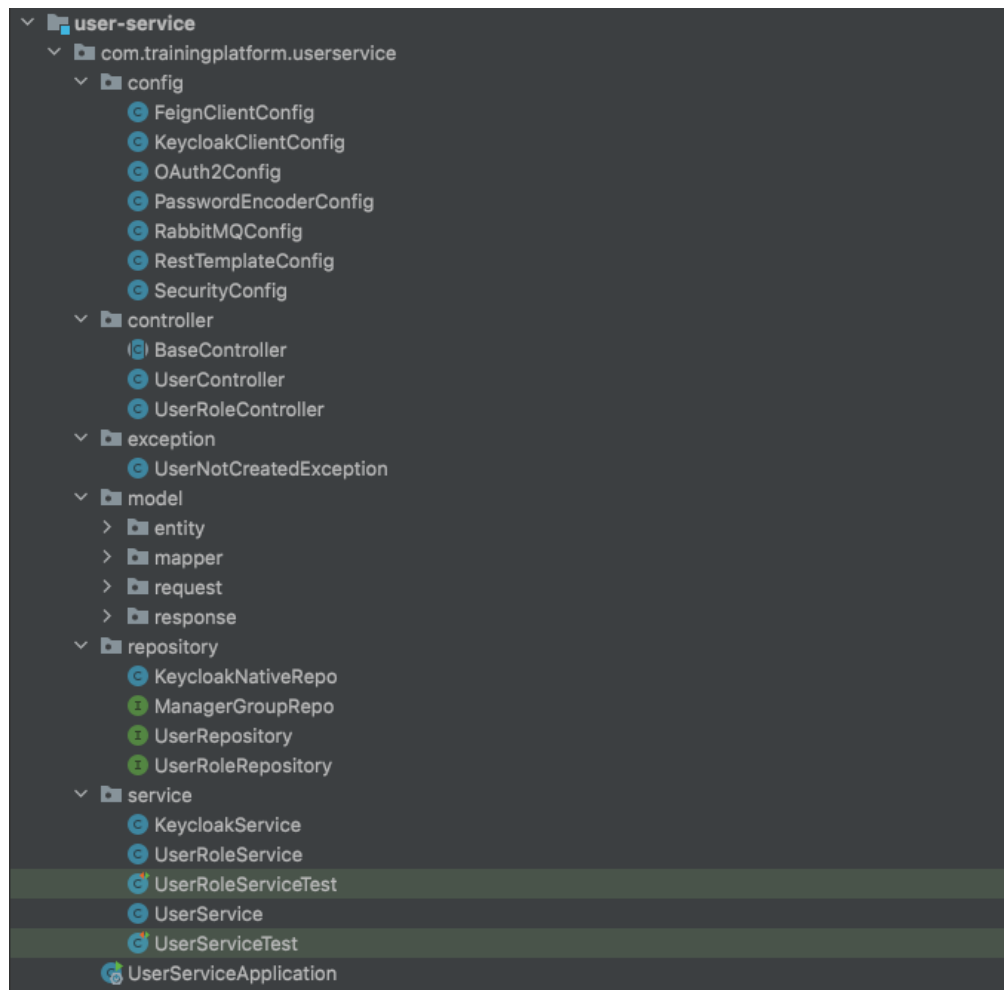
# Test Number	Aa Description	≡ Test Input	≡ Expected Output	≡ Related Use Case
1	<u>Return token when user is created</u>	User profile model	Access token	Create new user
2	<u>Throw 403 when user cannot be created</u>	User profile model	403 error	Create new user
3	<u>Throw 409 when user cannot be created</u>	User profile model	409 error	Created new user
4	<u>Return user when user is found</u>	User profile model	User Model	Get user
5	<u>Throw exception when user is not found</u>	User profile model	Exception	Get user
6	<u>Return user when user is found by username</u>	Username	User model	Get user
7	<u>Throw exception when user is not found by manager group id</u>	Manager group id of user	Exception	Get user
8	<u>Throw exception when user is not found by username</u>	Username	Exception	Get user
9	<u>Return users when role given</u>	User role	User models	Get all users

# Test Number	Aa Description	≡ Test Input	≡ Expected Output	≡ Related Use Case
10	<u>Return updated user when user profile updated</u>	Updated user profile	Updated user model	Update user
11	<u>Return access token when user logged-in</u>	User credentials	Access token	Login
12	<u>Throw exception when user credentials are not correct</u>	User credentials	Exception	Login
13	<u>Return all instructors</u>	-	User models	Get instructors of training
14	<u>Create new user role</u>	Role name and manager group id	-	Create new user role
15	<u>Assign user to given role</u>	Username and role name	-	Assign user to a role
16	<u>Return all trainings</u>	-	Training models	Get all trainings
17	<u>Return training when training is found</u>	Training id	Training model	Get training
18	<u>Throw exception when training is not found</u>	Training id	Exception	Get training
19	<u>Return training when training is created</u>	Training model	Training model	Create training
20	<u>Throw exception when training is not found</u>	Training model	Exception	Create training
21	<u>Update training when training is found</u>	Training model	Updated training model	Update training
22	<u>Throw exception when training is not found</u>	Training model	Exception	Update training
23	<u>Delete training when training is found</u>	Training id	-	Delete training
24	<u>Throw exception when training is not found</u>	Training id	Exception	Delete training
25	<u>Return user progress when training is found</u>	User id	User progress model	Get training progress of user
26	<u>Return participants when training is found</u>	Training id	User models	Get training participants
27	<u>Update lesson progress when lesson is found</u>	Lesson id	Updated lesson models	Track lesson progress
28	<u>Throw exception when lesson is not found</u>	Lesson id	Exception	Track lesson progress
29	<u>Return lesson when lesson is found</u>	Lesson id	Lesson model	Get lesson
30	<u>Throw exception when lesson is not found</u>	Lesson id	Exception	Get lesson
31	<u>Return lesson when lesson is created</u>	Lesson model	Lesson model	Create lesson
32	<u>Throw exception when lesson is not found</u>	Lesson model	Exception	Create lesson
33	<u>Update lesson when lesson is found</u>	Updated lesson model	-	Update lesson
34	<u>Throw exception when lesson is not found</u>	Updated lesson model	Exception	Update lesson
35	<u>Delete lesson when lesson is found</u>	Lesson id	-	Delete lesson
36	<u>Throw exception when lesson is not found</u>	Lesson id	Exception	Delete lesson
37	<u>Return success when participation request is sent</u>	Training id and user id	Success status	Send participation request to a training
38	<u>Return success when a participant is added to a training</u>	Training id and user ids	Success status	Add participants to a training
39	<u>Return pending participation requests</u>	Manager group id	Participation request models	List participation requests
40	<u>Return approved request ids when participations are approved</u>	Participation request ids	Participation request model	Approve participation requests

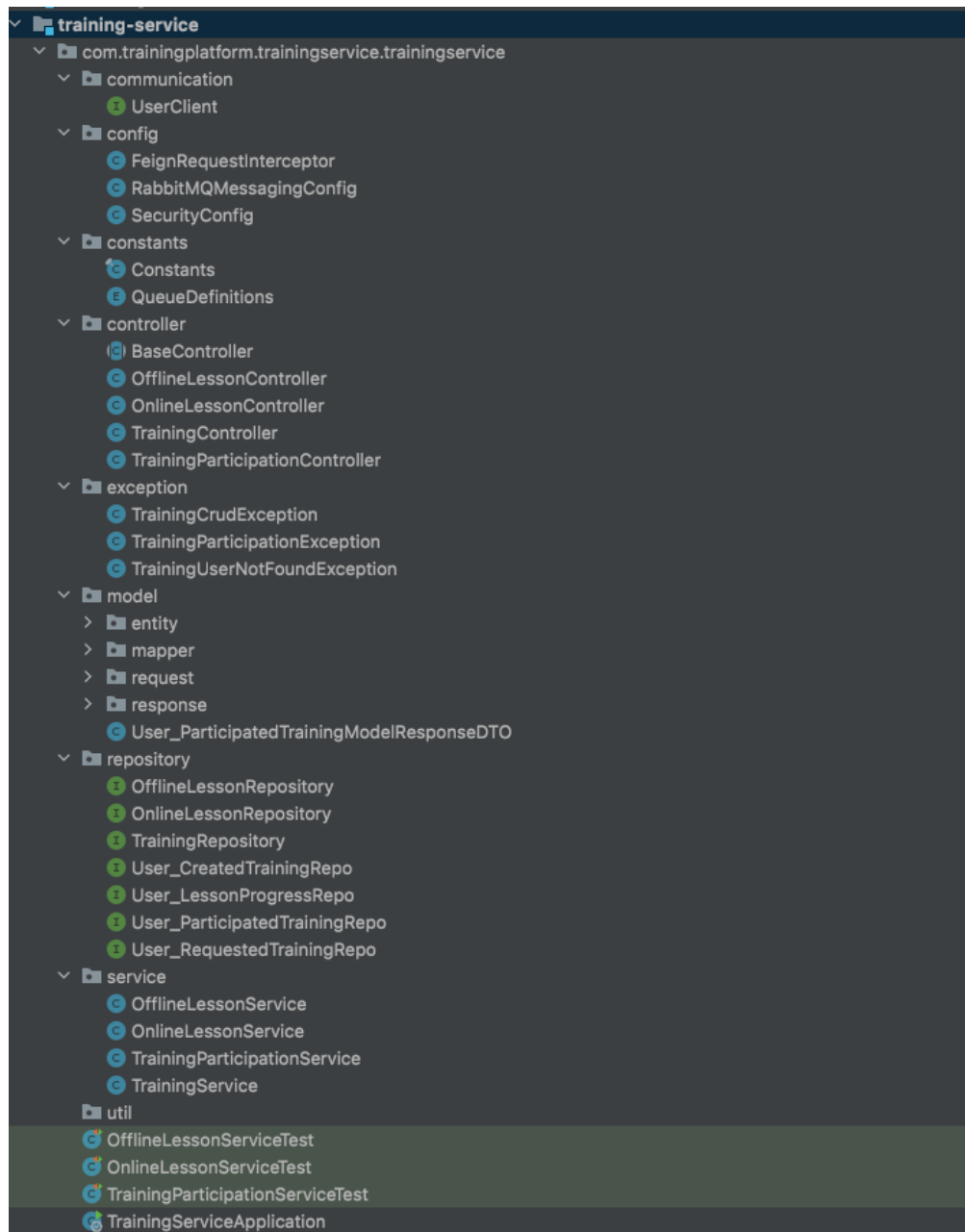
# Test Number	Aa Description	≡ Test Input	≡ Expected Output	≡ Related Use Case
41	<u>Return rejected request ids when participations are rejected</u>	Participation request ids	Participation request ids	Reject participation requests
42	<u>Upload video when lesson is given</u>	Lesson id and video file	Video file url	Upload training lesson video
43	<u>Delete file when file is found</u>	File id	-	Delete training lesson video
44	<u>Create user notification</u>	User notification model	-	Send notification
45	<u>Update notifications as read</u>	Updated user notification model	-	Read notifications
46	<u>Return notifications when user is given</u>	User id	User notification model	Get all notifications

## 2.4 White-Box Testing

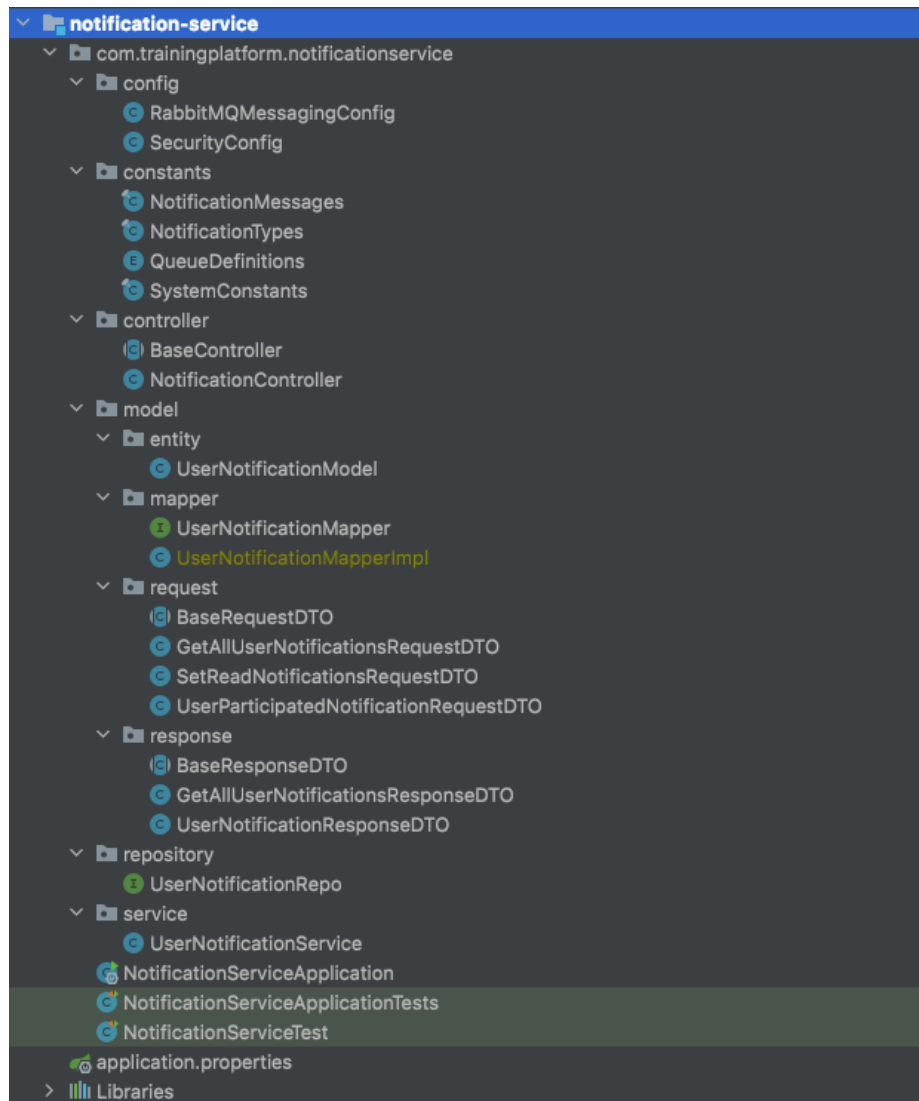
User Service Structure:



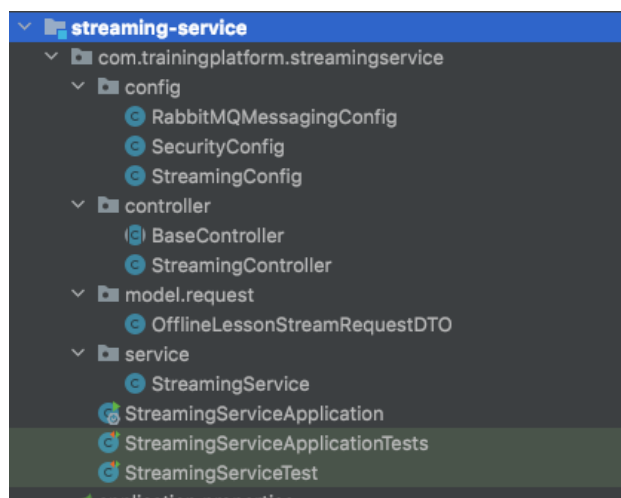
Training Service Structure:



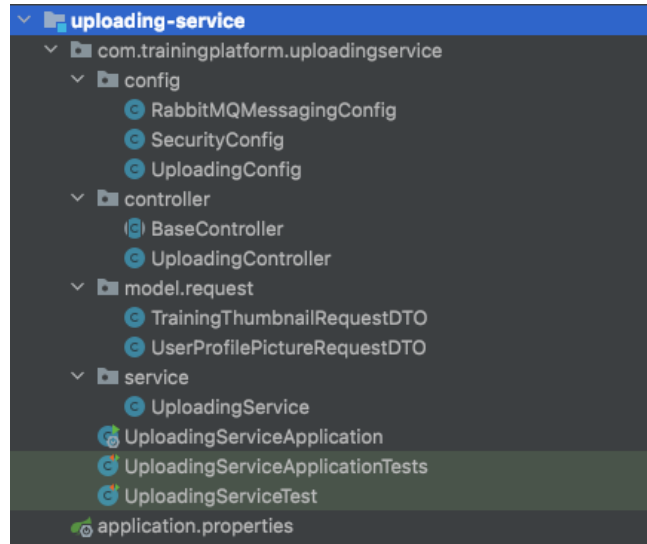
Notification Service Structure:



Streaming Service Structure:



## Uploading Service Structure:



# Test Number	Parent Service	Aa Title	Status
1	Notification Service	<u>it should return notification count when user given()</u>	passed
2	Notification Service	<u>it should return unread notifications when user given()</u>	passed
3	Notification Service	<u>it should return unread notification count when user given()</u>	passed
4	Notification Service	<u>it should return all notifications when user given()</u>	passed
5	Notification Service	<u>it should make notifs read()</u>	passed
6	Notification Service	<u>it should create notif()</u>	passed
7	Reporting Service	<u>it should return training when id given()</u>	passed
8	Reporting Service	<u>it should return participant list when training found()</u>	passed
9	Streaming Service	<u>it should upload video when lesson given()</u>	passed
10	Streaming Service	<u>it should delete file when file found()</u>	passed
11	Training (Participation) Service	<u>test_requestParticipation()</u>	passed
12	Training (Participation) Service	<u>it should return participation approved list when participation approved for offline training()</u>	passed

# Test Number	Parent Service	Aa Title	Status
13	Training (Participation) Service	<u>it should return participation approved list when participation approved for online training()</u>	passed
14	Training (Participation) Service	<u>it should return map when participants added()</u>	passed
15	Training (Participation) Service	<u>it should return participation reject list when participation approved for online training()</u>	passed
16	Training (Participation) Service	<u>it should return all participants when training found()</u>	passed
17	Training (Participation) Service	<u>it should return pending participation list()</u>	passed
18	Training (Participation) Service	<u>it should catch feignexception when user not found()</u>	passed
19	Training (Participation) Service	<u>it should addsingleparticipanttotraining()</u>	passed
20	Training (Online Lesson) Service	<u>test_updateOnlineLesson()</u>	passed
21	Training (Online Lesson) Service	<u>it should return onlinelessonrespondedto list when training found()</u>	passed
22	Training (Online Lesson) Service	<u>it should return onlinelesson when lesson found()</u>	passed
23	Training (Online Lesson) Service	<u>it should return onlinelessonrespondedto when lesson created()</u>	passed
24	Training (Online Lesson) Service	<u>it should delete lesson when lesson found()</u>	passed
25	Training (Offline Lesson) Service	<u>it should return emptylist when training not found()</u>	passed



# Test Number	≡ Parent Service	Aa Title	≡ Status
26	Training (Offline Lesson) Service	<u>it_should_delete_lesson()</u>	passed
27	Training (Offline Lesson) Service	<u>it_should_return_offlinelesson_when_lesson_found()</u>	passed
28	Training (Offline Lesson) Service	<u>it_should_return_offlinelessonrespondedtlist_when_training_found()</u>	passed
29	Training (Offline Lesson) Service	<u>it_should_return_offlinelessonrespondesto_when_lesson_created()</u>	passed
30	Training (Offline Lesson) Service	<u>it_should_update_lesson()</u>	passed
31	Training Service	<u>it_should_delete_training_when_training_found()</u>	passed
32	Training Service	<u>it_should_return_empty_list_when_no_trainings_exist()</u>	passed
33	Training Service	<u>test_update_successful()</u>	passed
34	Training Service	<u>it_should_return_participants_when_training_given()</u>	passed
35	Training Service	<u>it_should_throw_exception_when_training_not_found()</u>	passed
36	Training Service	<u>it_should_return_userrespondesto_when_user_given()</u>	passed
37	Training Service	<u>it_should_return_user_progress_when_offlinetraining_user_found()</u>	passed
38	Training Service	<u>it_should_return_trainingrespondesto_when_training_created()</u>	passed
39	Training Service	<u>it_should_return_all_trainings_when_trainings_exist()</u>	passed
40	Training Service	<u>it_should_return_user_progress_when_onlinetraining_user_found()</u>	passed
41	Training Service	<u>test_updatelessonprogress_successful()</u>	passed
42	Training Service	<u>it_should_return_true_if_user_participated_training()</u>	passed
43	Training Service	<u>it_should_return_trainingmodellist_when_user_participated()</u>	passed

# Test Number	≡ Parent Service	Aa Title	≡ Status
44	Training Service	<u>it should return training when training found()</u>	passed
45	User (Role) Service	<u>test_assignUserToUserRole()</u>	passed
46	User (Role) Service	<u>void create_new_user_role()</u>	passed
47	User (Role) Service	<u>it should return user role name when()</u>	passed
48	User (Role) Service	<u>it should return all user roles when manager group given()</u>	passed
49	User (Role) Service	<u>it should return manager group name when user role given()</u>	passed
50	User Service	<u>it should return updated user when update()</u>	passed
51	User Service	<u>it should throw when user cannot created 403()</u>	passed
52	User Service	<u>it should throw when user cannot created 409()</u>	passed
53	User Service	<u>it should return user when user found()</u>	passed
54	User Service	<u>it should throw exception when user not found()</u>	passed
55	User Service	<u>it should return all users when role given()</u>	passed
56	User Service	<u>it should return user when user found_getUserByUsername()</u>	passed
57	User Service	<u>it should return manager group id when user found()</u>	passed
58	User Service	<u>it should return token user when credentials correct()</u>	passed
59	User Service	<u>it should throw exception when credentials wrong()</u>	passed
60	User Service	<u>it should throw exception when user not found_getUserByUsername()</u>	passed
61	User Service	<u>it should throw exception when user not found_getManagerGroupId()</u>	passed
62	User Service	<u>it should return true when user found()</u>	passed
63	User Service	<u>it should return false when user not found()</u>	passed
64	User Service	<u>it should return token when user created()</u>	passed
65	User Service	<u>it should return all instructors list()</u>	passed
66	User Service	<u>it should return map()</u>	passed

## 3. Additional Tests

We did not perform additional tests.

## 4. Test Results

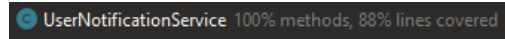
### 4.1 Detailed Test Report

We employed microservice architecture. Therefore, we have several services. Each service has classes that are named as \*Service. Business logic lies in these classes, hence we created unit tests to cover them.

Each test case we employed reflects the correct usage of methods. The number of test cases can be found in 4.2 test results.

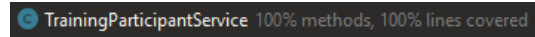
For services that has \*Service classes, method and line coverages can be seen via the images below. We used IntelliJIdea to calculate coverage.

- notification service



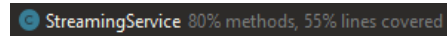
IntelliJ coverage snippet for UserNotificationService: 100% methods, 88% lines covered

- reporting service



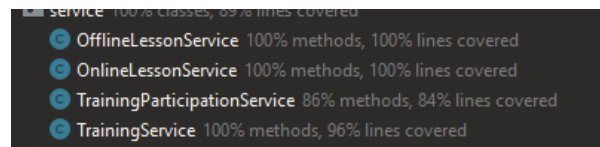
IntelliJ coverage snippet for TrainingParticipantService: 100% methods, 100% lines covered

- streaming service



IntelliJ coverage snippet for StreamingService: 80% methods, 55% lines covered

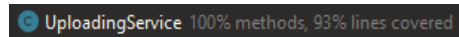
- training service



IntelliJ coverage snippet for training service:

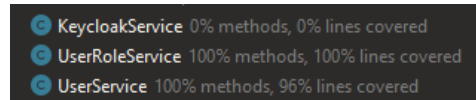
- OfflineLessonService 100% methods, 100% lines covered
- OnlineLessonService 100% methods, 100% lines covered
- TrainingParticipationService 86% methods, 84% lines covered
- TrainingService 100% methods, 96% lines covered

- uploading service



IntelliJ coverage snippet for UploadingService: 100% methods, 93% lines covered

- user service



IntelliJ coverage snippet for user service:

- KeycloakService 0% methods, 0% lines covered
- UserRoleService 100% methods, 100% lines covered
- UserService 100% methods, 96% lines covered

## 4.2 Detailed Test Results

- notification service

[https://s3-us-west-2.amazonaws.com/secure.notion-static.com/d6c14415-79bd-446c-89bc-10d5f63c9820/Test\\_Results\\_-\\_NotificationServiceTest.html](https://s3-us-west-2.amazonaws.com/secure.notion-static.com/d6c14415-79bd-446c-89bc-10d5f63c9820/Test_Results_-_NotificationServiceTest.html)

- reporting service

[https://s3-us-west-2.amazonaws.com/secure.notion-static.com/809ffdc9-79c4-49c4-9a2b-f6626936b32e/Test\\_Results\\_-\\_TrainingParticipantServiceTest.html](https://s3-us-west-2.amazonaws.com/secure.notion-static.com/809ffdc9-79c4-49c4-9a2b-f6626936b32e/Test_Results_-_TrainingParticipantServiceTest.html)

- streaming service

[https://s3-us-west-2.amazonaws.com/secure.notion-static.com/11156c26-5ba8-4e34-b811-f2a39457ebe5/Test\\_Results\\_-\\_StreamingServiceTest.html](https://s3-us-west-2.amazonaws.com/secure.notion-static.com/11156c26-5ba8-4e34-b811-f2a39457ebe5/Test_Results_-_StreamingServiceTest.html)

- training service

[https://s3-us-west-2.amazonaws.com/secure.notion-static.com/72e0e369-139d-4c22-aa09-710c2404c298/Test\\_Results\\_-\\_com\\_trainingplatform\\_trainingservice\\_trainingservice\\_in\\_training-service.html](https://s3-us-west-2.amazonaws.com/secure.notion-static.com/72e0e369-139d-4c22-aa09-710c2404c298/Test_Results_-_com_trainingplatform_trainingservice_trainingservice_in_training-service.html)

- uploading service

[https://s3-us-west-2.amazonaws.com/secure.notion-static.com/2a50ddc0-c795-4dce-83af-ecf7718cb129/Test\\_Results\\_-\\_UploadingServiceTest.html](https://s3-us-west-2.amazonaws.com/secure.notion-static.com/2a50ddc0-c795-4dce-83af-ecf7718cb129/Test_Results_-_UploadingServiceTest.html)

- user service

[https://s3-us-west-2.amazonaws.com/secure.notion-static.com/baff7fba-f274-40bb-af27-1dae552a22e2/Test\\_Results\\_-\\_com\\_trainingplatform\\_userservice\\_service\\_in\\_user-service.html](https://s3-us-west-2.amazonaws.com/secure.notion-static.com/baff7fba-f274-40bb-af27-1dae552a22e2/Test_Results_-_com_trainingplatform_userservice_service_in_user-service.html)