Valedictorian
Unit Test Plan
Version 1.0.0

Project team
J.M.A. Boender | 0978526
R. Coonen | 0902230
R.A.T. van Dijk | 0864724
H.R. Galioulline | 0927184
B.A.M. van Geffen | 0892070
A.A.W.M. de Kroon | 0905382
R. Morel | 0905326
W.M.W.R. Verlaek | 0908937
C.C. Weibel | 0883114

Project managers
J. Ubbink
R. Wouters

Project supervisor
dr. N. Zannone

Customer
prof.dr.ir. M.W.J. Prins
WW.C.C. Brekelmans
Abstract

This document describes the unit test plan for the SensUs Digital Platform. The tests described by this document are based on the design specified in the Detailed Design Document [2]. This document also complies with the ESA software standard.
# Contents

1 Introduction 7  
1.1 Purpose .......................................................... 7  
1.2 Overview ......................................................... 7  
1.3 List of definitions ............................................. 7  
1.4 List of abbreviations .......................................... 7  
1.5 List of references .............................................. 8  

2 Test plan 9  
2.1 Test items ........................................................ 9  
2.2 Features to be tested ......................................... 9  
2.3 Test deliverables ................................................. 9  
2.4 Testing tasks .................................................... 9  
2.5 Environment needs ........................................... 9  
2.6 Test case pass/fail criteria ................................. 10  

3 Test case specifications 11  
3.1 Server application tests ...................................... 11  
    3.1.1 Answer Module ........................................... 11  
    3.1.2 Measurement module methods ...................... 14  
    3.1.3 Livestream module methods ........................ 16  
    3.1.4 Question module methods ............................ 19  
    3.1.5 Tag module methods ................................ 25  
    3.1.6 Team module methods ................................ 26  
    3.1.7 User module methods ................................ 28  
    3.1.8 Video module methods ............................... 29  
    3.1.9 Picture module methods ............................... 32  
    3.1.10 Vote module methods ................................ 35  
    3.1.11 View module methods ................................ 44  
    3.1.12 Authorization module methods ................. 45  
3.2 Main web application tests ................................ 50  
    3.2.1 App module .............................................. 50  
    3.2.2 Explore module .......................................... 52  
    3.2.3 Live stream module ..................................... 55  
    3.2.4 Picture module ........................................... 60  
    3.2.5 QA module ................................................. 61  
    3.2.6 Sensor data module ..................................... 63  
    3.2.7 Team module .............................................. 63
3.2.8 Video module ................................................. 66
3.2.9 Vote module .................................................. 67

4 Test procedures ................................................. 73
  4.1 Running tests with coverage and detailed results ....................... 73

5 Test Coverage .................................................. 74
  5.1 Server code .................................................. 74
  5.2 Client code .................................................. 77

6 Test Reports .................................................. 79
  6.1 Server tests .................................................. 79
  6.2 Client tests .................................................. 82
DOCUMENT STATUS SHEET

GENERAL

Document title: Unit Test Plan v1.0.0
Identification: UTP/1.0.0
Authors: R. Coonen, J.M.A. Boender, B.A.M van Geffen, R. van Dijk, R. Morel, A.A.W.M. de Kroon
Document status: Final 1.0.0

DOCUMENT HISTORY

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Author(s)</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0.1</td>
<td>26-6-2016</td>
<td>J.M.A Boender</td>
<td>Document setup</td>
</tr>
<tr>
<td>1.0.0</td>
<td>6-7-2016</td>
<td>R. Coonen, J.M.A. Boender, B.A.M van Geffen, R. van Dijk, R. Morel, A.A.W.M. de Kroon</td>
<td>Completion of the document</td>
</tr>
</tbody>
</table>
### DOCUMENT CHANGE RECORDS

<table>
<thead>
<tr>
<th>Section</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>Valedictorian</td>
<td>Unit Test Plan</td>
</tr>
</tbody>
</table>
1 INTRODUCTION

1.1 PURPOSE

The purpose of this Unit Test Plan (UTP) is to specify how the SensUs Digital Platform should be tested so that all parts of the software work as intended. The SensUs Digital Platform has been split up into different units (as specified in the Detailed Design Document [2]) and this unit structure will dictate the structure of the UTP tests. Tests will be organized based on which unit they will be testing and will be called Unit Tests (UT). The results will be described in this document, and they will be checked with the DDD.

1.2 OVERVIEW

This document consists of five chapters after this introductory chapter. Chapter 2 will introduce all the features that need to be tested, as well as the criteria to pass each Unit Test (UT). Then, chapter 3 will list all the tests that make up this test plan. This list will define each test, along with the necessary inputs and expected outputs. Chapter 4 will use the tests described in chapter 3 to define the testing procedure that should be used to correctly execute all tests. Chapter 5 shows the coverage results for the tests described in chapter 3 and lastly the test reports are given in chapter 6.

1.3 LIST OF DEFINITIONS

<table>
<thead>
<tr>
<th>Award</th>
<th>A prize that can be won by a team participating in the SensUs competition.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Biomarker</td>
<td>A biological substance that acts as an indicator of a particular physiological or pathological state of an organism. The biomarker for SensUs 2017 is NT-proBNP. This is a protein that is a biomarker for heart failure.</td>
</tr>
<tr>
<td>Biosensor</td>
<td>A device of a team that measures the concentration or presence of a biomarker in a sample.</td>
</tr>
<tr>
<td>SensUs</td>
<td>An organization responsible for organizing a biosensor design competition between universities around the world.</td>
</tr>
<tr>
<td>SensUs Digital Platform</td>
<td>The product being developed by Valedictorian.</td>
</tr>
<tr>
<td>Valedictorian</td>
<td>The team that developed the SensUs Digital Platform.</td>
</tr>
</tbody>
</table>

1.4 LIST OF ABBREVIATIONS

Valedictorian | Unit Test Plan | 7
For any definitions specific to this product, the list of definitions in the URD [1] can be referenced.

### 1.5 LIST OF REFERENCES


2 TEST PLAN

2.1 TEST ITEMS

The software to be tested is the SensUs Digital Platform as specified in the DDD[2]. This includes the main SensUs Digital website that regular users will see, as well as the dashboard accessible only to the different levels of SensUs administration. Since the SensUs Digital mobile application was not implemented, we have no tests for that proposed part of the SensUs Digital Platform. More information on both the main SensUs Digital website, the dashboard and the SensUs Digital mobile application can be found in the URD [1].

2.2 FEATURES TO BE TESTED

The items to be tested are all functional units of the SensUs Digital Platform as specified in the DDD[2].

2.3 TEST DELIVERABLES

Before performing any of the tests, the following documents and products need to be finished:

- The DDD [2]
- UTP (this document)

After the the test plan is successfully executed, the following documents need to be delivered:

- The results of the UTP
- Problem reports (if any)

2.4 TESTING TASKS

Before starting on execution of the test plan the following tasks need to be done:

- Design of tests for each of the units specified in the DDD[2]
- For each test, input data will need to be created (seeding database, etc)

2.5 ENVIRONMENT NEEDS

To perform the unit tests in this document, the following resources are required:

- A desktop or laptop computer running Chrome version 57 or later
• A server running the SensUs Digital Platform back end

• Connection to the internet

2.6 TEST CASE PASS/FAIL CRITERIA

Every test describes its own pass/fail criteria - so depending on that criteria the test will either pass or fail. The complete unit test plan passes if all individual unit tests, as specified in chapter 3, also pass.
TEST CASE SPECIFICATIONS

The following sections describe all unit tests needed to be executed on the SensUs Digital Platform. The tests will follow the structure described below:

- **Test case identifier**
  This will be the unique identifier for the test case.

- **Test items**
  This will be the description of the function or method that is tested (as specified in the DDD [2]).

- **Input specification**
  This will be the description of the input necessary for the functions or methods being tested.

- **Output specifications**
  This will be the output expected from the functions or methods.

- **Environmental needs**
  This will be any other necessary constraints needed to perform the test in question.

3.1 SERVER APPLICATION TESTS

3.1.1 ANSWER MODULE

UT1 testLegalPost()

Test items: ‘POST’ request to api/question/.$questionId./answer’ where $questionId is the id of the question in the database.

Here we test the api endpoints for the "create" method of the answer module. This is used when a user wants to submit an answer to a question.

**Input specifications:** ['title' => 'answer title','description' => 'answer description']

**Output specifications:** 200 HTTP response indicating success - the answer is in the database.

**Environmental Needs:**

- The test case is stored in file 'AnswerControllerTest.php'.
- The question the answer is answering has to exist in the database.
- The user has a role of SensUs Personnel or higher.
UT2  
**testLegalPostWithLink()**

**Test items:** 'POST' request to api/question/.$questionId./answer' where $questionId is the id of the question in the database

Here we test the api endpoints for the "create" method of the answer module with a link as part of the answer. This is when a user wants to submit an answer with a link to a question.

**Input specifications:** [['title' => 'answer title','description' => 'answer description'], ['link' => ['source' => $this->legalLink['source_type'], 'id' => $this->legalLink['content_id']]]

**Output specifications:** 200 HTTP response indicating success - the answer is in the database.

**Environmental Needs:**
- The test case is stored in file 'AnswerControllerTest.php'.
- The question the answer is answering has to exist in the database.
- The user has a role of SensUs Personnel or higher.

UT3  
**testUnauthorizedPost()**

**Test items:** 'POST' request to api/question/.$questionId./answer' where $questionId is the id of the question in the database

Here we test the api endpoints for the "create" method of the answer module with a user that doesn't have proper authorization.

**Input specifications:** [['title' => 'answer title','description' => 'answer description']

**Output specifications:** 403 HTTP response indicating failure - the user doesn't have permission to do perform this action

**Environmental Needs:**
- The test case is stored in file 'AnswerControllerTest.php'.
- The question the answer is answering has to exist in the database.
- The user has a role of authorized user or lower.

UT4  
**testLegalAnswerApprove()**

**Test items:** 'PUT' request to api/question/.$questionId./answer' where $questionId is the id of the question in the database

Here we test the api endpoints for the 'update" method of the answer module. This is when a user wants to update the state of an answer.

**Input specifications:** [['title' => 'answer title','description' => 'answer description'],
array('status' => AnswerStates::Approved['value'])

**Output specifications:** 200 HTTP response indicating success - The Answer in the database now as "Approved" as it's status.

**Environmental Needs:**

- The test case is stored in file 'AnswerControllerTest.php'.
- The answer getting approved has to exist in the database.
- The user has a role of SensUs Personnel or higher.

### UT5 testIllegalAnswerApprove()

**Test items:** 'PUT' request to api/question/.$questionId/answer' where $questionId is the id of the question in the database

Here we test the api endpoints for the 'update' method of the answer module without proper authorization. This is when a user wants to update the state of an answer.

**Input specifications:** [['title' => 'answer title','description' => 'answer description'], array('status' => AnswerStates::Approved['value'])]

**Output specifications:** 403 HTTP response indicating failure - Unauthorized action.

**Environmental Needs:**

- The test case is stored in file 'AnswerControllerTest.php'.
- The answer getting approved has to exist in the database.
- The user has a role of team member or lower.

### UT6 testDeleteAnswer()

**Test items:** 'POST', 'request to api/answer/' . $answer['id']

Here we test the api endpoint for the 'delete' method of the answer module. We use this method when the user wants to delete an answer.

**Input specifications:** answerId

**Output specifications:** 200 HTTP response indicating success - the answer is no longer in the database.

**Environmental Needs:**

- The test case is stored in file 'AnswerControllerTest.php'.
- The answer getting deleted has to exist in the database.
- The user has a role of SensUs Personnel or higher.
3.1.2 MEASUREMENT MODULE METHODS

**UT7**  
**testGetJSONstructure()**

**Test items:** ‘GET’, ‘/api/sensordata’

Here we test the api endpoint for the ‘retrieve’ method of the Data Setting module. Specifically we check if the response is of the right structure.

**Input specifications:** No input needed.

**Output specifications:** 200 HTTP response indicating success - that we actually get something back and it is of the right structure.

**Environmental Needs:**

- The test case is stored in file ‘SensorDataControllerTest.php’.
- The data being retrieved has to exist in the database.

**UT8**  
**testGetAdminJSONstructure()**

**Test items:** ‘GET’, ‘/api/sensordata/admin’

Here we test the api endpoint for the ‘retrieve’ method of the Data Setting module. Specifically we check if the response is of the right structure.

**Input specifications:** No input needed.

**Output specifications:** 200 HTTP response indicating success - that we get a response and it is has the correct structure.

**Environmental Needs:**

- The test case is stored in file ‘SensorDataControllerTest.php’.
- The data being retrieved has to exist in the database.
- The user has permissions of a web administrator.

**UT9**  
**testGetAdminFalseIfNotAdmin()**

**Test items:** ‘GET’, ‘/api/sensordata/admin’

Here we test the api endpoint for the ‘retrieve’ method of the Data Setting module. Specifically we check the authorization for this request.

**Input specifications:** No input needed.

**Output specifications:** 403 HTTP response indicating forbidden - that the request is not executed.

**Environmental Needs:**

- The test case is stored in file ‘SensorDataControllerTest.php’.
- The user has permissions of SensUs Personnel (or lower).
UT10 testRetrieveDoesReturnAssignedConcentrationWhenAdmin()

Test items: 'GET', '/api/sensordata/admin'
Here we test the api endpoint for the 'retrieve' method of the Data Setting module. Specifically we check if the data returned is the same after it's inserted.
Input specifications: No input needed.
Output specifications: 200 HTTP response indicating forbidden - that the response contains the data we inserted.
Environmental Needs:
- The test case is stored in file 'SensorDataControllerTest.php'.
- The user has permissions of a web administrator.

UT11 testUpdateSingleValueAdmin()

Test items: 'PUT', '/api/sensordata/
Here we test the api endpoint for the 'update' method of the Data Setting module. Specifically we check if we can update data in the database.
Input specifications:
- ['sensordata' => $currentval, 'sensor_data',
  'settings' => $settings,
  'teamNameToYAxisScaleMap' => $teamNameToYAxisScaleMap,
  ]
Output specifications: 200 HTTP response indicating success - that the response contains the data we inserted.
Environmental Needs:
- The test case is stored in file 'SensorDataControllerTest.php'.
- The user has permissions of a web administrator.
- The data is already in the database.

UT12 testAdminRequiredForUpdate()

Test items: 'PUT', '/api/sensordata/
Here we test the api endpoint for the 'update' method of the Data Setting module. Specifically we check if the authorization check works.
Input specifications: [] - an empty array
Output specifications: 403 HTTP response indicating forbidden - that the request is forbidden.
Environmental Needs:
- The test case is stored in file 'SensorDataControllerTest.php'.
- The user has permissions of a SensUs Personnel or lower.

**UT13**

`testRetrieveDoesNotReturnRowWhenNotPublic()`

**Test items:** ‘GET’, ‘/api/sensordata/’

Here we test the api endpoint for the ‘retrieve’ method of the Data Setting module. Specifically we check if a set of data is set to hidden, that we do not return it.

**Input specifications:** No input needed.

**Output specifications:** 200 HTTP response indicating success - that the response we receive does not contain the data item was set to hidden.

**Environmental Needs:**
- The test case is stored in file 'SensorDataControllerTest.php'.
- The data is already in the database and set to hidden.

**UT14**

`testRetrieveReturnsRowWhenPublic()`

**Test items:** ‘GET’, ‘/api/sensordata/’

Here we test the api endpoint for the ‘retrieve’ method of the Data Setting module. Specifically we check if a set of data is not set to hidden, that we in fact return it.

**Input specifications:** No input needed.

**Output specifications:** 200 HTTP response indicating success - that the response we receive contains the data item was not set to hidden.

**Environmental Needs:**
- The test case is stored in file 'SensorDataControllerTest.php'.
- The data is already in the database and is not set to hidden.

### 3.1.3 LIVESTREAM MODULE METHODS

**UT15**

`testUpdateLivestreamSuccess()`

**Test items:** ‘PATCH’, ‘api/livestreams/$team_id/$liveStream_id’ where $team_id is the id of the team we are uploading to and the $liveStream_id is the id number of the livestream (youtube link in this case) to be set as the livestream.

Here we test the api endpoint for the ‘update’ method of the Livestream module. Specifically we check if we can successfully update a livestream.

**Input specifications:** No input needed.

**Output specifications:** 200 HTTP response indicating success - that the response we receive when we successfully update a livestream.

**Environmental Needs:**
• The test case is stored in file 'LivestreamControllerTest.php'.
• The user has permissions of SensUs Personnel or higher.
• The database has the livestream in the database

UT16 testCreateLivestreamSuccess()

Test items: ‘PATCH’, ‘api/livestreams/$team_id/$liveStream_id’ where $team_id is the id of the team we are uploading to and the $liveStream_id is the id number of the livestream (youtube link in this case) to be set as the livestream.

Here we test the api endpoint for the ‘update’ method of the Livestream module. Specifically we check if we can successfully create a livestream that is not already in the database.

Input specifications: No input needed.

Output specifications: 200 HTTP response indicating success - that the response we receive when we successfully update a livestream.

Environmental Needs:

• The test case is stored in file ‘LivestreamControllerTest.php’.
• The user has permissions of SensUs Personnel or higher.
• livestream we want to add is not already in the database.

UT17 testUnauthorizedUpdate()

Test items: ‘PATCH’, ‘api/livestreams/$team_id/$liveStream_id’ where $team_id is the id of the team we are uploading to and the $liveStream_id is the id number of the livestream (youtube link in this case) to be set as the livestream.

Here we test the api endpoint for the ‘update’ method of the Livestream module. Specifically we check if we dont have the right role, if the api in fact rejects our update.

Input specifications: No input needed.

Output specifications: 403 HTTP response indicating forbidden - that the update didn’t get executed.

Environmental Needs:

• The test case is stored in file ‘LivestreamControllerTest.php’.
• The user has permissions of team member or lower.

UT18 testInvalidContent()

Test items: ‘PATCH’, ‘api/livestreams/$team_id/$liveStream_id’ where $team_id is the id of the team we are uploading to and the $liveStream_id is the id number of the livestream (youtube
link in this case) to be set as the livestream.
Here we test the api endpoint for the ‘update’ method of the Livestream module. Specifically we check if we give bad input, if the api in fact rejects our update.

**Input specifications:** No input needed.

**Output specifications:** 400 HTTP response indicating bad request - that the update didn’t get executed.

**Environmental Needs:**

- The test case is stored in file ‘LivestreamControllerTest.php’.
- The user has permissions of team member or lower.
- The $livestream_id does not adhere to the required length.

## UT19 testNonExistingTeam()

**Test items:** ‘PATCH’, ‘api/livestreams/$team_id/$liveStream_id’ where $team_id is the id of the team we are uploading to and the $liveStream_id is the id number of the livestream (youtube link in this case) to be set as the livestream.

Here we test the api endpoint for the ‘update’ method of the Livestream module. Specifically we check if we give bad input, if the api in fact rejects our update.

**Input specifications:** No input needed.

**Output specifications:** 400 HTTP response indicating bad request - that the update didn’t get executed.

**Environmental Needs:**

- The test case is stored in file ‘LivestreamControllerTest.php’.
- The user has permissions of team member or lower.
- The $team_id isn’t an existing team.

## UT20 testGetStream()

**Test items:** ‘GET’, ‘api/livestreams/$team_id/$liveStream_id’ where $team_id is the id of the team we are uploading to and the $liveStream_id is the id number of the livestream (youtube link in this case) to be set as the livestream.

Here we test the api endpoint for the ‘retrieve’ method of the Livestream module. Specifically we check if the retrieve request works as intended.

**Input specifications:** No input needed.

**Output specifications:** 200 HTTP response indicating request success - that the request has the livestream we want.

**Environmental Needs:**
• The test case is stored in file 'LivestreamControllerTest.php'.

• The livestream requested is actually in the database

3.1.4 QUESTION MODULE METHODS

UT21 testLegalPost()

Test items: ‘POST’, ‘api/question’.
Here we test the api endpoint for the ‘create’ method of the Question module. Specifically we check if a question without a link can be added.

Input specifications:
[
  'title' => "Test Title",
  'description' => 'Test Description goes here. ',
  'team_id' => 1,
]

Output specifications: 200 HTTP response indicating request success- that the question has been successfully submitted.

Environmental Needs:
• The test case is stored in file 'QuestionControllerTest.php'.
• The user has permissions of at least an authenticated user.
• The question being submitted has no link.

UT22 testLegalPostWithVideo()

Test items: ‘POST’, ‘api/question’.
Here we test the api endpoint for the ‘create’ method of the Question module. Specifically we check if a question with a link can be added.

Input specifications:
[
  'title' => 'Test Title",
  'description' => 'Test Description goes here. ',
  "team_id" => 1,
  'link' => [
    'source' => $link['source_type'],
    'id' => $link['content_id']
  ]
]

Output specifications: 200 HTTP response indicating request success- that the question has been successfully submitted.

Environmental Needs:
• The test case is stored in file 'QuestionControllerTest.php'.

• The user has permissions of at least an authenticated user.

**UT23**

**testIllegalPost()**

**Test items:** ‘POST’, ‘api/question’.

Here we test the api endpoint for the ‘create’ method of the Question module. Specifically we check if the authorization check for submitting a question is checked.

**Input specifications:**

```php
['title' => 'Test Title',
'description' => 'Test Description goes here',
'team_id' => 1,
]
```

**Output specifications:** 403 HTTP response indicating forbidden- that the action is not authorized.

**Environmental Needs:**

• The test case is stored in file 'QuestionControllerTest.php'.

• The user has permissions of at least an authenticated user.

**UT24**

**testPostToInvalidTeam()**

**Test items:** ‘POST’, ‘api/question’.

Here we test the api endpoint for the ‘create’ method of the Question module. Specifically we check if submitting a question to a non-existent team is possible.

**Input specifications:**

```php
['title' => 'Test Title',
'description' => 'Test Description goes here',
'team_id' => 1987,
]
```

**Output specifications:** 422 HTTP response indicating unprocessable entity- that the question needs to be submitted to a valid team.

**Environmental Needs:**

• The test case is stored in file 'QuestionControllerTest.php'.

• The user has permissions of at least an authenticated user.

**UT25**

**testPostWithoutDescription()**

**Test items:** ‘POST’, ‘api/question’.

Here we test the api endpoint for the ‘create’ method of the Question module. Specifically we check if submitting a question without a description is possible. Note: For brevity, this example omits detailed input and output specifications.

**Environmental Needs:**

• The test case is stored in file 'QuestionControllerTest.php'.

• The user has permissions of at least an authenticated user.
check that submitting a question without a description is possible.

**Input specifications:**
```php
[  
    'title' => 'Test Title',
    'description' => '',
    'team_id' => 1,
]
```

**Output specifications:** 200 HTTP response indicating success - that the question has successfully been submitted without a description.

**Environmental Needs:**
- The test case is stored in file 'QuestionControllerTest.php'.
- The user has permissions of at least an authenticated user.

**UT26 testPostWithoutTitle()**

**Test items:** 'POST', 'api/question'.
Here we test the api endpoint for the 'create' method of the Question module. Specifically we check that submitting a question without a title is not possible.

**Input specifications:**
```php
[  
    'title' => '',
    'description' => 'Test Description goes here,'
    'team_id' => 1,
]
```

**Output specifications:** 422 HTTP response indicating unprocessable entity - that the given question needs a title.

**Environmental Needs:**
- The test case is stored in file 'QuestionControllerTest.php'.
- The user has permissions of at least an authenticated user.

**UT27 testLegalApprove()**

**Test items:** 'PUT', 'api/question/$questionId' where $questionId is the id of a question that is in the database.
Here we test the api endpoint for the 'update' method of the Question module. Specifically we check that approving a question is possible by SensUs Personnel.

**Input specifications:**
```
[  
    'status' => QuestionStates::Approved
]
```

**Output specifications:** 200 HTTP response indicating success - that the given question got approved

**Environmental Needs:**
- The test case is stored in file 'QuestionControllerTest.php'.
- The user has permissions of at least an authenticated user.
• The test case is stored in file 'QuestionControllerTest.php'.
• The user has permissions of at least SensUs Personnel.

**UT28 testIllegalApprove()**

**Test items:** ‘PUT’, ‘api/question/$questionId’ where $questionId is the id of a question that is in the database.

Here we test the api endpoint for the ‘update’ method of the Question module. Specifically we check that approving a question is not possible by roles with less permissions than SensUs Personnel.

**Input specifications:** ['status' => QuestionStates::Approved]

**Output specifications:** 403 HTTP response indicating forbidden- that the given question did not get approved

**Environmental Needs:**

• The test case is stored in file 'QuestionControllerTest.php'.
• The user has permissions of an authenticated user.

**UT29 testIllegalApproveByTeamMember()**

**Test items:** ‘PUT’, ‘api/question/$questionId’ where $questionId is the id of a question that is in the database.

Here we test the api endpoint for the ‘update’ method of the Question module. Specifically we check that approving a question is not possible by roles with less permissions than SensUs Personnel.

**Input specifications:** ['status' => QuestionStates::Approved]

**Output specifications:** 403 HTTP response indicating forbidden- that the given question did not get approved

**Environmental Needs:**

• The test case is stored in file 'QuestionControllerTest.php'.
• The user has permissions of a team member.

**UT30 testLegalClaim()**

**Test items:** ‘PUT’, ‘api/question/$questionId’ where $questionId is the id of a question that is in the database.

Here we test the api endpoint for the ‘update’ method of the Question module. Specifically we check that claiming a question is possible by roles with SensUs Personnel and higher.

**Input specifications:** ['status' => QuestionStates::BeingAnswered]
Output specifications: 200 HTTP response indicating success- that the given question got claimed

Environmental Needs:

- The test case is stored in file 'QuestionControllerTest.php'.
- The user has permissions of a SensUs Personnel or higher.

**UT31**

testChangeToPendingDelete()

Test items: ‘PUT’, ‘api/question/$questionId’ where $questionId is the id of a question that is in the database.
Here we test an api endpoint for the ‘update’ method of the Question module. Specifically we check that claiming a question after it has been answer, in fact deletes the answer.

Input specifications: ['status' => QuestionStates::Pending]

Output specifications: 200 HTTP response indicating success- that the given question got set back to pending and the answer was deleted

Environmental Needs:

- The test case is stored in file 'QuestionControllerTest.php'.
- The user has permissions of a SensUs Personnel or higher.

**UT32**

testLegalDelete()

Test items: ‘PUT’, ‘api/question/$questionId’ where $questionId is the id of a question that is in the database.
Here we test the api endpoint for the ‘update’ method of the Question module. Specifically we check that deleting a question is possible for SensUs Personnel and higher roles.

Input specifications: No input needed.

Output specifications: 200 HTTP response indicating success- that the given question has been deleted from the database

Environmental Needs:

- The test case is stored in file 'QuestionControllerTest.php'.
- The user has permissions of a SensUs Personnel or higher.
- The question to delete has to be in the database.

**UT33**

testLegalDelete()

Test items: ‘PUT’, ‘api/question/$questionId’ where $questionId is the id of a question that is in the database.
Here we test the api endpoint for the ‘update’ method of the Question module. Specifically we check that deleting a question is not possible for Team members.

**Input specifications:** No input needed.

**Output specifications:** 403 HTTP response indicating forbidden- that the given question has not been deleted from the database

**Environmental Needs:**

- The test case is stored in file 'QuestionControllerTest.php'.
- The user has permissions of a Team member.
- The question to delete has to be in the database.

**UT34 testGetApproved()**

**Test items:** ‘GET’, ‘api/questions’

Here we test the api endpoint for the ‘retrieve’ method of the Question module. Specifically we check that we are able to retrieve all approved questions as an authenticated user.

**Input specifications:** No input needed.

**Output specifications:** 200 HTTP response indicating success- that all approved questions have been returned.

**Environmental Needs:**

- The test case is stored in file 'QuestionControllerTest.php'.
- The user has permissions of an unauthenticated user.

**UT35 testGetAll()**

**Test items:** ‘GET’, ‘api/questions/all’

Here we test the api endpoint for the ‘retrieveAll’ method of the Question module. Specifically we check that we are able to retrieve all questions in the databases.

**Input specifications:** No input needed.

**Output specifications:** 200 HTTP response indicating success- that all questions have been returned.

**Environmental Needs:**

- The test case is stored in file 'QuestionControllerTest.php'.
- The user has permissions of a SensUs Personnel or higher.
UT36  
**testGetFromTeam()**

**Test items:** ‘GET’, ‘api/team/$team_id/questions’ where $team_id is the id of one of the teams in the database.

Here we test the api endpoint for the ‘retrieve’ method of the Question module. Specifically we check that we are able to retrieve all questions for a specific team in the database.

**Input specifications:** No input needed.

**Output specifications:** 200 HTTP response indicating success- that all questions for that team have been returned.

**Environmental Needs:**
- The test case is stored in file 'QuestionControllerTest.php'.
- The user has permissions of a SensUs Personnel or higher.

UT37  
**testUnauthorizedGetAll()**

**Test items:** ‘GET’, ‘api/questions/all’

Here we test the api endpoint for the ‘retrieveAll’ method of the Question module. Specifically we check that we are not able to retrieve all questions in the databases when the user role is less then SensUs Personnel.

**Input specifications:** No input needed.

**Output specifications:** 403 HTTP response indicating forbidden- that all questions have not been returned.

**Environmental Needs:**
- The test case is stored in file 'QuestionControllerTest.php'.
- The user has permissions of a team member.

3.1.5 **TAG MODULE METHODS**

UT38  
**testGetTeams()**

**Test items:** ‘GET’ request to /api/tags/

Tests the retrieveAll method of the Tag module as an unauthenticated user.

**Input specifications:** None

**Output specifications:** 200 HTTP response indicating success. Each tag in the response should be contained in the database.

**Environmental Needs:**
- The test case is stored in file 'TagControllerTest.php'.
- Request as an unauthenticated user.
- Tags must be seeded in the database.
3.1.6 TEAM MODULE METHODS

UT39 testAuthenticatedPrivilege()

Test items: 'PUT' request to /api/team/$i where $i takes the value of every team id in the database
Tests the update method of the Team module as an authenticated user.

Input specifications: ['description' => 'fakeDescription']

Output specifications: 403 HTTP response indicating failure for each $i. Description should be unchanged in the database.

Environmental Needs:
- The test case is stored in file 'TeamControllerTest.php'.
- Teams should be seeded in the database.
- Request as an authenticated user.

UT40 testRegularPrivilege()

Test items: 'PUT' request to /api/team/$i where $i takes the value of every team id in the database
Tests the update method of the Team module as an unauthenticated user.

Input specifications: ['description' => 'fakeDescription']

Output specifications: 403 HTTP response indicating failure for each $i. Description should be unchanged in the database.

Environmental Needs:
- The test case is stored in file 'TeamControllerTest.php'.
- Teams should be seeded in the database.
- Request as an unauthenticated user.

UT41 testEmptyDescription()

Test items: 'PUT' request to /api/team/$i where $i takes the value of every team id in the database
Tests the update method of the Team module as an administrator with empty description as input.

Input specifications: ['description' => '']

Output specifications: 422 HTTP response indicating failure for each $i. Description should be unchanged in the database.

Environmental Needs:
• The test case is stored in file 'TeamControllerTest.php'.

• Teams should be seeded in the database.

• Request as an administrator.

**UT42**

testTextMatch()

**Test items:** 'PUT' request to /api/team/$i where $i takes the value of every team id in the database
Tests the update method of the Team module as an administrator.

**Input specifications:** ['description' => 'Test description']

**Output specifications:** 200 HTTP response indicating success for each $i. Description should be changed in the database to 'Test description'.

**Environmental Needs:**

• The test case is stored in file 'TeamControllerTest.php'.

• Teams should be seeded in the database.

• Request as an administrator.

**UT43**

testInvalidTeam()

**Test items:** 'PUT' request to /api/team/0
Tests the update method of the Team module as an administrator with an invalid team id.

**Input specifications:** ['description' => 'Test description']

**Output specifications:** 404 HTTP response indicating failure. Description should be unchanged in the database.

**Environmental Needs:**

• The test case is stored in file 'TeamControllerTest.php'.

• Request as an administrator.

• There should be no team with id 0 in the database.

**UT44**

testGetTeams()

**Test items:** 'GET' request to /api/teams/
Tests the retrieveAll method of the Team module as an unauthenticated user.

**Input specifications:** None

**Output specifications:** 200 HTTP response indicating success. The response should have a key 'teams'.

**Environmental Needs:**
• The test case is stored in file 'TeamControllerTest.php'.

• Request as an unauthenticated user.

**UT45**

**testGetTeamInfo()**

**Test items:** ‘GET’ request to /api/teams/$i where $i is the id of the team in the database with the lowest id

Tests the retrieve method of the Team module as an unauthenticated user.

**Input specifications:** None

**Output specifications:** 200 HTTP response indicating success. The response should have keys 'id', 'name', 'country' and 'description'.

**Environmental Needs:**

• The test case is stored in file 'TeamControllerTest.php'.

• Request as an unauthenticated user.

### 3.1.7 USER MODULE METHODS

**UT46**

**testAllCombinations()**

**Test items:** ‘POST’ request to /api/users/edit

Tests whether a web administrator can assign any team id and user role combination to another user.

**Input specifications:** ['user_id' => user_id(), 'user_role' => role, 'team_id' => team id]

**Output specifications:** 200 HTTP response indicating success, and a JSON with structure ['userRole', 'teamId', 'userId']

**Environmental Needs:**

• The test case is stored in file 'UserManagementTest.php'.

• Request is made as an web administrator user.

**UT47**

**testSelectNoTeam()**

**Test items:** ‘POST’ request to /api/users/edit

Tests whether a super admin can unassign a team id from another (team member) user.

**Input specifications:** ['user_id' => user_id(), 'user_role' => 10, 'team_id' => -1]

**Output specifications:** 200 HTTP response indicating success, and JSON ['userRole' => 10, 'teamId' => -1, 'userId' => user_id()]

**Environmental Needs:**

• The test case is stored in file 'UserManagementTest.php'.

---

Valedictorian | Unit Test Plan 28
• Request is made as an web administrator user.
• The user that is changed is assigned to a team.

UT48  
**testUnauthorizedEdit()**

**Test items:** 'POST' request to /api/users/edit
Tests whether an administrator (SensUs personnel, unauthorized) can edit another user.

**Input specifications:** ['user_id' => user_id(), 'user_role' => 20, 'team_id' => team_id()]

**Output specifications:** 403 HTTP response indicating an unauthorized request.

**Environmental Needs:**
• The test case is stored in file 'UserManagementTest.php'.
• Request is made as an administrator (SensUs personnel) user.
• The user that is changed is assigned to a team.

UT49  
**testNonExistingUser()**

**Test items:** 'POST' request to /api/users/edit
Tests whether editing a non-existing user is handled properly.

**Input specifications:** ['user_id' => user_id(), 'user_role' => 20, 'team_id' => team_id()]

**Output specifications:** 403 HTTP response indicating an unauthorized request.

**Environmental Needs:**
• The test case is stored in file 'UserManagementTest.php'.
• Request is made as an administrator (SensUs personnel) user.
• The user that is changed is assigned to a team.

3.1.8 VIDEO MODULE METHODS

UT50  
**testGetAllFoundMediaEqualsDatabaseMedia()**

**Test items:** 'GET' request to /api/videos
Tests the RetrieveAll method of the Video module as an unauthenticated user.

**Input specifications:** None

**Output specifications:** 200 HTTP response indicating success. The returned Video items should equal the videos present in the database.

**Environmental Needs:**
• The test case is stored in file 'MediaControllerTest.php'.
• The database has videos seeded in the database.
• Request as unauthenticated user.
UT51 testSingleMediaSuccess()

Test items: 'GET' request to /api/video/$i for each $i such that $i is the id of a video in the database
Tests the Retrieve method of the Video module as an unauthenticated user.
Input specifications: None
Output specifications: 200 HTTP response indicating success. The returned Video item should equal the video present in the database
Environmental Needs:

- The test case is stored in file 'MediaControllerTest.php'.
- The database has videos seeded in the database.
- Request as unauthenticated user.

UT52 testStoreInvalidInput()

Test items: 'POST' request to /api/video
Tests the Create method of the Video module as an authenticated user with invalid input.
Input specifications: Several inputs are tried, where for each a valid video input has one of its fields removed. Input is also attempted where the 'source_id' field is set to 33 which is an invalid source id.
Output specifications: 422 HTTP response indicating failure for each attempted input.
Environmental Needs:

- The test case is stored in file 'MediaControllerTest.php'.
- Request as authenticated user.

UT53 testProperPostMediaInput()

Test items: 'POST' request to /api/video
Tests the Create method of the Video module as an administrator.
Input specifications: Valid video input
Output specifications: 200 HTTP response indicating success. After the request the video should be in the database
Environmental Needs:

- The test case is stored in file 'MediaControllerTest.php'.
- Request as authenticated user.
**UT54  testDelete()**

**Test items:** ‘DELETE’ request to /api/video/$id where $id is the id of a video in the database
Tests the Delete method of the Video module as an administrator.

**Input specifications:** None

**Output specifications:** 200 HTTP response indicating success. After the request the video with id $id should no longer be in the database.

**Environmental Needs:**
- The test case is stored in file 'MediaControllerTest.php'.
- Request as authenticated user.
- Video with id $id must be in the database.

**UT55  testAlterVideo()**

**Test items:** ‘PUT’ request to /api/video/$id where $id is the id of a video in the database
Tests the Update method of the Video module as an administrator.

**Input specifications:** ['content_id' => unique_content_id(), 'source_type' => 1, 'title'=> 'test3', 'tags' => [2,1,3], 'teams' => [1,2], 'description' => 'blabla']

**Output specifications:** 200 HTTP response indicating success. After the request the video with id $id should have fields matching the input in the database.

**Environmental Needs:**
- The test case is stored in file 'MediaControllerTest.php'.
- Request as authenticated user.
- Video with id $id must be in the database and must have fields different from the input.

**UT56  testGetVideoInfo()**

**Test items:** ‘GET’ request to /api/video/info/$id where $id is the id of a video in the database
Tests the GetInfo method of the Video module as an unauthenticated user.

**Input specifications:** None

**Output specifications:** 200 HTTP response indicating success. HTTP body should contain ‘kink’, ‘etag’, ‘items’ and ‘pageInfo’ fields

**Environmental Needs:**
- The test case is stored in file 'MediaControllerTest.php'.
- Request as unauthenticated user.
- Picture with id $id must be in the database.
- Valid YouTube API key must be in .env file.
UT57  testInvalidVideo()

**Test items:** ‘GET’ request to /api/video/info/0
Tests the GetInfo method of the Video module as an unauthenticated user on an invalid video id.

**Input specifications:** None

**Output specifications:** 404 HTTP response indicating failure.

**Environmental Needs:**

- The test case is stored in file ‘MediaControllerTest.php’.
- Request as unauthenticated user.
- Picture with id 0 must not be in the database.
- Valid YouTube API key must be in .env file.

UT58  testMissingConfig()

**Test items:** ‘GET’ request to /api/video/info/$id where $id is the id of a video in the database
Tests the GetInfo method of the Video module as an unauthenticated user when no YouTube API key is configured.

**Input specifications:** None

**Output specifications:** 500 HTTP response indicating failure

**Environmental Needs:**

- The test case is stored in file ‘MediaControllerTest.php’.
- Request as unauthenticated user.
- Picture with id $id must be in the database.
- No YouTube API key must be in .env file.

3.1.9  PICTURE MODULE METHODS

UT59  testGetAllFoundMediaEqualsDatabaseMedia()

**Test items:** ‘GET’ request to /api/pictures
Tests the RetrieveAll method of the picture module as an unauthenticated user.

**Input specifications:** None

**Output specifications:** 200 HTTP response indicating success. The returned Picture items should equal the pictures present in the database

**Environmental Needs:**

- The test case is stored in file ‘MediaControllerTest.php’.
- The database has pictures seeded in the database.
- Request as unauthenticated user.

**UT60**

**testSingleMediaSuccess()**

**Test items:** ‘GET’ request to /api/picture/$i for each $i such that $i is the id of a picture in the database

Tests the Retrieve method of the Picture module as an unauthenticated user.

**Input specifications:** None

**Output specifications:** 200 HTTP response indicating success. The returned Picture item should equal the video present in the database

**Environmental Needs:**

- The test case is stored in file ‘MediaControllerTest.php’.
- The database has pictures seeded in the database.
- Request as unauthenticated user.

**UT61**

**testStoreInvalidInput()**

**Test items:** ‘POST’ request to /api/picture

Tests the Create method of the Picture module as an authenticated user with invalid input.

**Input specifications:** Several inputs are tried, where for each a valid picture input has one of its fields removed. Input is also attempted where the ‘source_id’ field is set to 33 which is an invalid source id.

**Output specifications:** 422 HTTP response indicating failure for each attempted input.

**Environmental Needs:**

- The test case is stored in file ‘MediaControllerTest.php’.
- Request as authenticated user.

**UT62**

**testProperPostMediaInput()**

**Test items:** ‘POST’ request to /api/picture

Tests the Create method of the Picture module as an administrator.

**Input specifications:** Valid picture input

**Output specifications:** 200 HTTP response indicating success. After the request the picture should be in the database

**Environmental Needs:**

- The test case is stored in file ‘MediaControllerTest.php’.
- Request as authenticated user.
Test items: ‘DELETE’ request to /api/picture/$id where $id is the id of a picture in the database
Tests the Delete method of the Picture module as an administrator.
Input specifications: None
Output specifications: 200 HTTP response indicating success. After the request the picture
with id $id should no longer be in the database.
Environmental Needs:

- The test case is stored in file 'MediaControllerTest.php'.
- Request as authenticated user.
- Picture with id $id must be in the database.

Test items: ‘PUT’ request to /api/picture/$id where $id is the id of a picture in the database
Tests the Update method of the Picture module as an administrator.
Input specifications: ['content_id' => unique_content_id(), 'source_type' => 2, 'title'=> 'test3',
'tags' => [2,1,3], 'teams' => [1,2], 'description' => 'blabla']
Output specifications: 200 HTTP response indicating success. After the request the picture
with id $id should have fields matching the input in the database.
Environmental Needs:

- The test case is stored in file 'MediaControllerTest.php'.
- Request as authenticated user.
- Picture with id $id must be in the database and must have fields different from the input.

Test items: ‘GET’ request to /api/picture/info/$id where $id is the id of a picture in the database
Tests the GetInfo method of the Picture module as an unauthenticated user.
Input specifications: None
Output specifications: 200 HTTP response indicating success. HTTP body should contain ‘reason’ fields
Environmental Needs:

- The test case is stored in file 'PictureControllerTest.php'.
- Request as unauthenticated user.
- Picture with id $id must be in the database.
3.1.10 VOTE MODULE METHODS

**UT66 testCastUnauthenticated()**

**Test items:** ‘POST’ request to /api/votes/cast
Tests the cast method of the Vote module as an unauthenticated user.

**Input specifications:** ['team_a' => 1, 'team_b' => 2]

**Output specifications:** 401 HTTP response indicating failure. Vote count should be unchanged in the database.

**Environmental Needs:**
- The test case is stored in file 'VoteControllerTest.php'.
- Request as unauthenticated user.

**UT67 testCastAuthenticatedNewVote()**

**Test items:** ‘POST’ request to /api/votes/cast
Tests the cast method of the Vote module as an authenticated user with no registered votes.

**Input specifications:** ['team_a' => 1, 'team_b' => 2]

**Output specifications:** 200 HTTP response indicating success. Vote count should have increased for team 1 and team 2 by one each in the database.

**Environmental Needs:**
- The test case is stored in file 'VoteControllerTest.php'.
- Request as authenticated user with no votes registered.
- Teams with id 1 and 2 must be seeded in the database.

**UT68 testCastAuthenticatedUpdateVote()**

**Test items:** ‘POST’ request to /api/votes/cast
Tests the cast method of the Vote module as an authenticated user to update registered votes.

**Input specifications:** ['team_a' => 3, 'team_b' => 4]

**Output specifications:** 200 HTTP response indicating success. Vote count should have increased for team 3 and team 4 by one each in the database and decreased by one each for team 1 and 2.

**Environmental Needs:**
- The test case is stored in file 'VoteControllerTest.php'.
- Request as authenticated user with votes registered for team 1 and 2.
- Teams with id 1, 2, 3 and 4 must be seeded in the database.
**UT69 testCastMissingTeamNew()**

**Test items:** ‘POST’ request to /api/votes/cast
Tests the cast method of the Vote module as an authenticated user with no registered vote with only one team as input instead of two.

**Input specifications:**
- first test: ['team_a' => 1]
- second test: ['team_b' => 1]

**Output specifications:** For both tests: 400 HTTP response indicating failure. Vote count should be unchanged.

**Environmental Needs:**
- The test case is stored in file ‘VoteControllerTest.php’.
- Request as authenticated user with no votes registered.
- Teams with id 1 must be seeded in the database.

**UT70 testCastMissingTeamUpdate()**

**Test items:** ‘POST’ request to /api/votes/cast
Tests the cast method of the Vote module as an authenticated user to update registered votes with only one team as input instead of two.

**Input specifications:**
- First test: ['team_a' => 1]
- Second test: ['team_b' => 4]

**Output specifications:** For both tests: 400 HTTP response indicating failure. Vote count should be unchanged.

**Environmental Needs:**
- The test case is stored in file ‘VoteControllerTest.php’.
- Request as authenticated user with votes registered for team 1 and 4.
- Teams with id 1 and 4 must be seeded in the database.

**UT71 testCastNoDataNew()**

**Test items:** ‘POST’ request to /api/votes/cast
Tests the cast method of the Vote module as an authenticated user with no registered votes with no data as input.

**Input specifications:** Empty

**Output specifications:** 400 HTTP response indicating failure. Vote count should be unchanged.

**Environmental Needs:**
- The test case is stored in file ‘VoteControllerTest.php’.
- Request as authenticated user with no votes registered.
UT72 testCastNoDataUpdate()

Test items: ‘POST’ request to /api/votes/cast
Tests the cast method of the Vote module as an authenticated user to update registered votes with no data as input.

Input specifications: Empty

Output specifications: 400 HTTP response indicating failure. Vote count should be unchanged.

Environmental Needs:

- The test case is stored in file ‘VoteControllerTest.php’.
- Request as authenticated user with votes registered for team 3 and 5.
- Teams with id 3 and 5 should be seeded in the database.

UT73 testCastSameTeamsNew()

Test items: ‘POST’ request to /api/votes/cast
Tests the cast method of the Vote module as an authenticated user with no registered votes with the same team on both input fields.

Input specifications: ['team_a' => 8, 'team_b' => 8]

Output specifications: 400 HTTP response indicating failure. Vote count should be unchanged.

Environmental Needs:

- The test case is stored in file ‘VoteControllerTest.php’.
- Request as authenticated user no votes registered.
- Team with id 8 should be seeded in the database.

UT74 testCastSame_teamsUpdate()

Test items: ‘POST’ request to /api/votes/cast
Tests the cast method of the Vote module as an authenticated user to update registered votes with the same team on both input fields.

Input specifications: ['team_a' => 6, 'team_b' => 6]

Output specifications: 400 HTTP response indicating failure. Vote count should be unchanged.

Environmental Needs:

- The test case is stored in file ‘VoteControllerTest.php’.
- Request as authenticated user with votes registered for team 1 and 8.
- Teams with id 1, 6 and 8 should be seeded in the database.
UT75  testCastInvalidTeamNew()

Test items: ‘POST’ request to /api/votes/cast
Tests the cast method of the Vote module as an authenticated user with no registered votes
with invalid team id as one of the inputs where the other input is valid.

Input specifications: First test: ['team_a' => -1, 'team_b' => 1]
Second test: ['team_a' => null, 'team_b' => 1]
Third test: ['team_a' => 'this is a team', 'team_b' => 1]
Fourth test: ['team_a' => 1, 'team_b' => -1]
Fifth test: ['team_a' => 1, 'team_b' => null]
Sixth test: ['team_a' => 1, 'team_b' => 'this is a team']

Output specifications: For each test: 400 HTTP response indicating failure. Vote count should
be unchanged.

Environmental Needs:

- The test case is stored in file ‘VoteControllerTest.php’.
- Request as authenticated user with no votes registered.
- Teams with id 1 should be seeded in the database.

UT76  testCastInvalidTeamUpdate()

Test items: ‘POST’ request to /api/votes/cast
Tests the cast method of the Vote module as an authenticated user to update registered votes
with invalid team id as one of the inputs where the other input is valid.

Input specifications: First test: ['team_a' => -1, 'team_b' => 1]
Second test: ['team_a' => null, 'team_b' => 1]
Third test: ['team_a' => 'this is a team', 'team_b' => 1]
Fourth test: ['team_a' => 1, 'team_b' => -1]
Fifth test: ['team_a' => 1, 'team_b' => null]
Sixth test: ['team_a' => 1, 'team_b' => 'this is a team']

Output specifications: For each test: 400 HTTP response indicating failure. Vote count should
be unchanged.

Environmental Needs:

- The test case is stored in file ‘VoteControllerTest.php’.
- Request as authenticated user with votes registered for team 1 and 9.
- Teams with id 1 and 9 should be seeded in the database.
July 6, 2017

UT77

testCastInvalidTeamsNew()

Test items: ‘POST’ request to /api/votes/cast
Tests the cast method of the Vote module as an authenticated user with no registered votes with invalid team id’s as both inputs.

Input specifications: First test: ['team_a' => -1, 'team_b' => -2]
Second test: ['team_a' => -1, 'team_b' => -1]
Third test: ['team_a' => 'this is a team', 'team_b' => null]
Fourth test: ['team_a' => null, 'team_b' => null]
Fifth test: ['team_a' => -23945723455463463463456356, 'team_b' => ';;;;']
Sixth test: ['team_a' => 3.0000001, 'team_b' => 5.00000001]

Output specifications: For each test: 400 HTTP response indicating failure. Vote count should be unchanged.

Environmental Needs:
- The test case is stored in file ‘VoteControllerTest.php’. 
- Request as authenticated user with no votes registered.

UT78

testCastInvalidTeamsUpdate()

Test items: ‘POST’ request to /api/votes/cast
Tests the cast method of the Vote module as an authenticated user to update registered votes with invalid team id’s as both inputs.

Input specifications: First test: ['team_a' => -1, 'team_b' => -2]
Second test: ['team_a' => -1, 'team_b' => -1]
Third test: ['team_a' => 'this is a team', 'team_b' => null]
Fourth test: ['team_a' => null, 'team_b' => null]
Fifth test: ['team_a' => -23945723455463463463456356, 'team_b' => ';;;;']
Sixth test: ['team_a' => 3.0000001, 'team_b' => 5.00000001]

Output specifications: For each test: 400 HTTP response indicating failure. Vote count should be unchanged.

Environmental Needs:
- The test case is stored in file ‘VoteControllerTest.php’. 
- Request as authenticated user with votes registered for team 1 and 8.
- Teams with id 1 and 8 should be seeded in the database.

UT79

testCastGetUnauthenticated()

Test items: ‘GET’ request to /api/votes/cast
Tests the cast method of the Vote module as an unauthenticated user using the wrong method.
Input specifications: None
Output specifications: 404 HTTP response indicating failure. Vote count should be unchanged.
Environmental Needs:

- The test case is stored in file 'VoteControllerTest.php'.
- Request as unauthenticated user.

**UT80**

**testCastGetUnauthenticated()**

Test items: 'GET' request to /api/votes/cast
Tests the cast method of the Vote module as an authenticated user using the wrong method.
Input specifications: None
Output specifications: 404 HTTP response indicating failure. Vote count should be unchanged.
Environmental Needs:

- The test case is stored in file 'VoteControllerTest.php'.
- Request as authenticated user.

**UT81**

**testCastTeamIDsStringNew()**

Test items: 'POST' request to /api/votes/cast
Tests the cast method of the Vote module as an authenticated user with no registered votes, where the team id's are given as strings instead of as integers.
Input specifications: ['team_a' => '1', 'team_b' => '2']
Output specifications: 200 HTTP response indicating success. Vote count should have increased for team 1 and team 2 by one each in the database.
Environmental Needs:

- The test case is stored in file 'VoteControllerTest.php'.
- Request as authenticated user with no votes registered.
- Teams with id 1 and 2 must be seeded in the database.

**UT82**

**testCastTeamIDsStringUpdate()**

Test items: 'POST' request to /api/votes/cast
Tests the cast method of the Vote module as an authenticated user to update registered votes, where the team id's are given as strings instead of as integers.
Input specifications: ['team_a' => '6', 'team_b' => '4']
Output specifications: 200 HTTP response indicating success. Vote count should have increased for team 6 and team 4 by one each, and decreased by one each for teams 3 and 8 in
the database.

Environmental Needs:

- The test case is stored in file ‘VoteControllerTest.php’.
- Request as authenticated user with votes registered for team 3 and 8.
- Teams with id 3, 4, 6 and 8 must be seeded in the database.

**UT83**  
**testCasInvalidJSONUnauthenticated()**

**Test items:** ‘POST’ request to /api/votes/cast

Tests the cast method of the Vote module as an unauthenticated user with no registered votes, where the body of the request consists of invalid JSON.

**Input specifications:** '{msg: "invalid json"}'

**Output specifications:** 400 HTTP response indicating failure. Vote count should have remained unchanged.

**Environmental Needs:**

- The test case is stored in file ‘VoteControllerTest.php’.
- Request as unauthenticated user with no votes registered.

**UT84**  
**testCasInvalidJSONAuthenticated()**

**Test items:** ‘POST’ request to /api/votes/cast

Tests the cast method of the Vote module as an authenticated user with no registered votes, where the body of the request consists of invalid JSON.

**Input specifications:** '{msg: "invalid json"}'

**Output specifications:** 400 HTTP response indicating failure. Vote count should have remained unchanged.

**Environmental Needs:**

- The test case is stored in file ‘VoteControllerTest.php’.
- Request as authenticated user with no votes registered.

**UT85**  
**testCasSameAsPreviousCast()**

**Test items:** ‘POST’ request to /api/votes/cast

Tests the cast method of the Vote module as an unauthenticated user with no registered votes, where a request is sent to the API end point twice with the same teams in each of the requests.

**Input specifications:** ['First request: ['team_a' => 6, 'team_b' => 1]
Second request: ['team_a' => 6, 'team_b' => 1]
Output specifications: Two 200 HTTP responses indicating success for each request. Vote count for team 1 and 6 should be increased by one each in the database after the first request. The second request should have no effect on vote counts.

Environmental Needs:

- The test case is stored in file 'VoteControllerTest.php'.
- Request as unauthenticated user with no votes registered.
- Teams with id 1 and 6 should be seeded in the database.

UT86 testCastPartySameAsPreviousCastA()

Test items: ‘POST’ request to /api/votes/cast
Tests the cast method of the Vote module as an unauthenticated user with no registered votes, where a request is sent to the API end point twice where request 1 and 2 share a team.

Input specifications: ['First request: ['team_a' => 6, 'team_b' => 1]
Second request: ['team_a' => 6, 'team_b' => 3]

Output specifications: Two 200 HTTP responses indicating success for each request. Vote count for team 1 and 6 should be increased by one each in the database after the first request. The second request should have no effect on the vote counts of team 6, however team 1 should lose one vote and team 3 should gain one vote after the second request.

Environmental Needs:

- The test case is stored in file 'VoteControllerTest.php'.
- Request as unauthenticated user with no votes registered.
- Teams with id 1, 3 and 6 should be seeded in the database.

UT87 testCastPartySameAsPreviousCastB()

Test items: ‘POST’ request to /api/votes/cast
Tests the cast method of the Vote module as an unauthenticated user with no registered votes, where a request is sent to the API end point twice where request 1 and 2 share a team.

Input specifications: ['First request: ['team_a' => 6, 'team_b' => 3]
Second request: ['team_a' => 1, 'team_b' => 3]

Output specifications: Two 200 HTTP responses indicating success for each request. Vote count for team 3 and 6 should be increased by one each in the database after the first request. The second request should have no effect on the vote counts of team 6, however team 1 should lose one vote and team 1 should gain one vote after the second request.

Environmental Needs:

- The test case is stored in file 'VoteControllerTest.php'.
Request as unauthenticated user with no votes registered.

Teams with id 1, 3 and 6 should be seeded in the database.

**UT88 testCastPartySameAsPreviousCastACross()**

**Test items:** 'POST' request to /api/votes/cast

Tests the cast method of the Vote module as an unauthenticated user with no registered votes, where a request is sent to the API end point twice where request 1 and 2 share a team.

**Input specifications:** ['First request: ['team_a' => 6, 'team_b' => 3]
Second request: ['team_a' => 1, 'team_b' => 6]

**Output specifications:** Two 200 HTTP responses indicating success for each request. Vote count for team 3 and 6 should be increased by one each in the database after the first request. The second request should have no effect on the vote counts of team 6, however team 3 should lose one vote and team 1 should gain one vote after the second request.

**Environmental Needs:**

- The test case is stored in file 'VoteControllerTest.php'.
- Request as unauthenticated user with no votes registered.
- Teams with id 1, 3 and 6 should be seeded in the database.

**UT89 testCastPartySameAsPreviousCastBCross()**

**Test items:** 'POST' request to /api/votes/cast

Tests the cast method of the Vote module as an unauthenticated user with no registered votes, where a request is sent to the API end point twice where request 1 and 2 share a team.

**Input specifications:** First request: ['team_a' => 6, 'team_b' => 3]
Second request: ['team_a' => 3, 'team_b' => 1]

**Output specifications:** Two 200 HTTP responses indicating success for each request. Vote count for team 3 and 6 should be increased by one each in the database after the first request. The second request should have no effect on the vote counts of team 6, however team 3 should lose one vote and team 1 should gain one vote after the second request.

**Environmental Needs:**

- The test case is stored in file 'VoteControllerTest.php'.
- Request as unauthenticated user with no votes registered.
- Teams with id 1, 3 and 6 should be seeded in the database.
UT90  testLegalPost()

**Test items:** 'GET' request to /api/votes/counts
Tests the retrieve method of the vote module as an authenticated user.

**Input specifications:** None

**Output specifications:** Total vote count should be integer. All teams should be present in the returned data. Total votes cast should be 40.

**Environmental Needs:**

- The test case is stored in file 'VoteControllerTest.php'.
- Twenty users must be seeded into the database with 2 votes cast randomly each, such that the total vote count in the database is 40.

### 3.1.11 VIEW MODULE METHODS

**UT91**  testGetIndex()

**Test items:** 'GET' request to /
Tests the RetrieveIndex method of the View module as an unauthenticated user.

**Input specifications:** None

**Output specifications:** 200 HTTP response indicating success.

**Environmental Needs:**

- The test case is stored in file 'ViewControllerTest.php'.
- Request as an unauthenticated user.

**UT92**  testGetDashboard()

**Test items:** 'GET' request to /dashboard
Tests the RetrieveDashboard method of the View module as an administrator.

**Input specifications:** None

**Output specifications:** 200 HTTP response indicating success.

**Environmental Needs:**

- The test case is stored in file 'ViewControllerTest.php'.
- Request as an administrator.

**UT93**  testUnauthorizedDashboard()

**Test items:** 'GET' request to /dashboard
Tests the RetrieveDashboard method of the View module as an authenticated user.

**Input specifications:** None
Output specifications: 403 HTTP response indicating failure.

Environmental Needs:

- The test case is stored in file 'ViewControllerTest.php'.
- Request as an authenticated user.

### 3.1.12 AUTHORIZATION MODULE METHODS

**UT94 testAllCombinations()**

Test items: 'POST' request to /api/users/edit
Tests whether a web administrator can assign any team id and user role combination to another user.

**Input specifications:** 
'\text{['user\_id' => user\_id(), 'user\_role' => role, 'team\_id' => team\_id]}'

**Output specifications:** 200 HTTP response indicating success, and a JSON with structure ['userRole', 'teamId', 'userId']

Environmental Needs:

- The test case is stored in file 'UserManagementTest.php'.
- Request is made as an web administrator user.

**UT95 testSelectNoTeam()**

Test items: 'POST' request to /api/users/edit
Tests whether a super admin can unassign a team id from another (team member) user.

**Input specifications:** 
'\text{['user\_id' => user\_id(), 'user\_role' => 10, 'team\_id' => -1]}'

**Output specifications:** 200 HTTP response indicating success, and JSON ['userRole' => 10, 'teamId' => -1, 'userId' => user\_id()]

Environmental Needs:

- The test case is stored in file 'UserManagementTest.php'.
- Request is made as an web administrator user.
- The user that is changed is assigned to a team.

**UT96 testUnauthorizedEdit()**

Test items: 'POST' request to /api/users/edit
Tests whether an administrator (SensUs personnel, unauthorized) can edit another user.

**Input specifications:** 
'\text{['user\_id' => user\_id(), 'user\_role' => 20, 'team\_id' => team\_id()]}'

**Output specifications:** 403 HTTP response indicating an unauthorized request.

Environmental Needs:
- The test case is stored in file 'UserManagementTest.php'.
- Request is made as an administrator (SensUs personnel) user.
- The user that is changed is assigned to a team.

**UT97**

**testNonExistingUser()**

**Test items:** ‘POST’ request to /api/users/edit
Tests whether editing a non-existing user is handled properly.

**Input specifications:** ['user_id' => -1, 'user_role' => 10, 'team_id' => team_id()]

**Output specifications:** 404 HTTP response indicating the user is not found.

**Environmental Needs:**
- The test case is stored in file 'UserManagementTest.php'.
- Request is made as a web administrator user.

**UT98**

**testNonExistingUser()**

**Test items:** ‘POST’ request to /api/users/edit
Tests whether editing an unspecified user (in the request) is handled properly.

**Input specifications:** ['user_role' => 10, 'team_id' => team_id()]

**Output specifications:** 400 HTTP response indicating the request is invalid.

**Environmental Needs:**
- The test case is stored in file 'UserManagementTest.php'.
- Request is made as a web administrator user.

**UT99**

**testNonExistingRole()**

**Test items:** ‘POST’ request to /api/users/edit
Tests whether changing the role of an user to an non existing user role is handled properly.

**Input specifications:** ['user_id' => user_id(), 'user_role' => 77, 'team_id' => team_id()]

**Output specifications:** 404 HTTP response indicating the user role does not exist.

**Environmental Needs:**
- The test case is stored in file 'UserManagementTest.php'.
- Request is made as a web administrator user.
UT100  testNonExistingRole()

Test items: ‘POST’ request to /api/users/edit
Tests whether changing an user to an unspecified role (in the request) is handled properly.
Input specifications: ['user_id' => user_id(), 'team_id' => team_id()]
Output specifications: 400 HTTP response indicating the request is invalid.

Environmental Needs:
- The test case is stored in file 'UserManagementTest.php'.
- Request is made as a web administrator user.

UT101  testNonExistingTeam()

Test items: ‘POST’ request to /api/users/edit
Tests whether changing the team of an user to an non existing team is handled properly.
Input specifications: ['user_id' => user_id(), 'team_id' => -2]
Output specifications: 404 HTTP response indicating the team does not exist.

Environmental Needs:
- The test case is stored in file 'UserManagementTest.php'.
- Request is made as a web administrator user.

UT102  testNonExistingTeam()

Test items: ‘POST’ request to /api/users/edit
Tests whether changing an user to an unspecified team (in the request) is handled properly.
Input specifications: ['user_id' => user_id(), 'user_role' => 10]
Output specifications: 400 HTTP response indicating the request is invalid.

Environmental Needs:
- The test case is stored in file 'UserManagementTest.php'.
- Request is made as a web administrator user.

UT103  testBan()

Test items: ‘POST’ request to /api/users/ban
Tests whether banning a (unbanned) user succeeds.
Input specifications: ['user_id' => user_id(), 'ban_status' => true]
Output specifications: 200 HTTP response indicating the request is successful.

Environmental Needs:
- The test case is stored in file 'UserManagementTest.php'.
- Request is made as a web administrator user.
- The user to be changes is unbanned.

**UT104 testBan()**

**Test items:** 'POST' request to /api/users/ban
Tests whether unbanning a (banned) user succeeds.

**Input specifications:** ['user_id' => user_id(), 'ban_status' => false]

**Output specifications:** 200 HTTP response indicating the request is successful.

**Environmental Needs:**
- The test case is stored in file 'UserManagementTest.php'.
- Request is made as a web administrator user.
- The user to be changes is banned.

**UT105 testMissingArgumentsBan()**

**Test items:** 'POST' request to /api/users/ban
Tests whether banning an unspecified user (in the request) is handled properly.

**Input specifications:** ['ban_status' => true]

**Output specifications:** 404 HTTP response indicating the team does not exist.

**Environmental Needs:**
- The test case is stored in file 'UserManagementTest.php'.
- Request is made as a web administrator user.

**UT106 testMissingArgumentsBan()**

**Test items:** 'POST' request to /api/users/ban
Tests whether setting the ban status of an user to an unspecified value (in the request) is handled properly.

**Input specifications:** ['user_id' => user_id()]

**Output specifications:** 400 HTTP response indicating the request is invalid.

**Environmental Needs:**
- The test case is stored in file 'UserManagementTest.php'.
- Request is made as a web administrator user.
UT107 testMissingArgumentsBan()

**Test items: 'POST' request to /api/users/ban**
Tests whether setting the ban status of an user to an invalid (non-boolean) value is handled properly.

**Input specifications:** ['user_id' => user_id(), 'ban_status' => 'banned']

**Output specifications:** 400 HTTP response indicating the request is invalid.

**Environmental Needs:**
- The test case is stored in file 'UserManagementTest.php'.
- Request is made as a web administrator user.

UT108 testMissingArgumentsBan()

**Test items: 'POST' request to /api/users/ban**
Tests whether setting the ban status of a non-existing user is handled properly.

**Input specifications:** ['user_id' => -1, 'ban_status' => true]

**Output specifications:** 404 HTTP response indicating the user is not found.

**Environmental Needs:**
- The test case is stored in file 'UserManagementTest.php'.
- Request is made as a web administrator user.

UT109 testSuperAdminBan()

**Test items: 'POST' request to /api/users/ban**
Tests banning a web administrator.

**Input specifications:** ['user_id' => user_id(), 'ban_status' => true]

**Output specifications:** 400 HTTP response indicating the request is invalid.

**Environmental Needs:**
- The test case is stored in file 'UserManagementTest.php'.
- Request is made as a web administrator user.
- The user to be changes is a web administrator user.

UT110 testGetUsers()

**Test items: 'GET' request to /api/users/all**
Tests banning a web administrator.

**Input specifications:** ['user_id' => user_id(), 'ban_status' => true]

**Output specifications:** 200 HTTP response with JSON structure ['users']

**Environmental Needs:**
The test case is stored in file 'UserManagementTest.php'.

Request is made as a web administrator user.

**UT111**

-testUnauthorizedGetUsers()

Test items: ‘GET’ request to /api/users/all
Tests banning a web administrator as a team member user (unauthorized).

Input specifications: ['user_id' => user_id(), 'ban_status' => true]

Output specifications: 403 HTTP response indicating that the request was unauthorized.

Environmental Needs:

- The test case is stored in file 'UserManagementTest.php'.
- Request is made as a team member user.

### 3.2 MAIN WEB APPLICATION TESTS

#### 3.2.1 APP MODULE

**UT112**

Test items: Login Action
Test a successful login action

Input specifications: ['token' => 'this is a test']

Output specifications:
LOGIN_REQUEST
LOGIN_SUCCESS

with ['name' => 'Test name', 'avatar' => 'Best movie ever.avi', 'email' => 'abc@xyz.alphabet']

Environmental Needs:

- The api route '/social/handleStateless/google' is mocked to respond with the desired name, avatar and email, given the input token.
- The test case is stored in file 'app.actions.test.js'.

**UT113**

Test items: Login Action
Test a failed login action

Input specifications: ['token' => 'this is a test']

Output specifications:
LOGIN_REQUEST
LOGIN_FAILURE

with ['error' => 'Request failed with status code 401']

Environmental Needs:
• The api route '/social/handleStateless/google' is mocked to respond with a 401 status code.

• The test case is stored in file 'app.actions.test.js'.

UT114

Test items: Login Request
Test whether the correct login request action is generated
Input specifications: ['token' => 'this is a test']
Output specifications: LOGIN_REQUEST

UT115

Test items: Login Request
Test whether an error is thrown if no token is given
Input specifications: No token given
Output specifications: Error thrown

UT116

Test items: Login Success
Test whether the correct login success action is created
Input specifications: ['name' => 'Test name', 'avatar' => 'Best movie ever.avi', 'email' => 'abc@xyz.alphabet']
Output specifications: LOGIN_SUCCESS
with ['name' => 'Test name', 'avatar' => 'Best movie ever.avi', 'email' => 'abc@xyz.alphabet']

UT117

Test items: Login Failure
Test whether the correct login failure action is created
Input specifications: ['token' => 'this is a test', 'error' => 'test error']
Output specifications: LOGIN_FAILURE
with ['authCode' => 'this is a test', 'error' => 'test error']

Environmental Needs:

- The test case is stored in file 'app.actions.test.js'.

### 3.2.2 EXPLORE MODULE

**UT118**

**Test items:** Reset Explore Action
Test whether the corrects action is created for resetting the explore view

**Input specifications:**

**Output specifications:** RESET_EXPLORE

**Environmental Needs:**

- The test case is stored in file 'explore.actions.test.js'.

**UT119**

**Test items:** Expand Viewable Tiles Action
Test whether increasing the size of the explore view generates the correct action

**Input specifications:** ['to' => 5]

**Output specifications:** EXPAND_EXPLORE_TILES_VIEWABLE with ['to' => 5]

**Environmental Needs:**

- The test case is stored in file 'explore.actions.test.js'.

**UT120**

**Test items:** Expand Viewable Tiles Action
Test that an error is thrown if infinity is given

**Input specifications:** ['to' => \(\infty\)]

**Output specifications:** Throw 'invalid argument'

**Environmental Needs:**

- The test case is stored in file 'explore.actions.test.js'.

**UT121**

**Test items:** Expand Viewable Tiles Action
Test whether an error is thrown if a non integer is given

**Input specifications:** ['to' => Object]

**Output specifications:** Throw 'invalid argument'

**Environmental Needs:**
• The test case is stored in file 'explore.actions.test.js'.

**UT122**

**Test items:** Set Seed Action  
Test if an action is created to set the correct seed  
**Input specifications:**  
['seed' => 0.5]  
**Output specifications:** SET_SEED  
with ['seed' => 0.5]  
**Environmental Needs:**

• The test case is stored in file 'explore.actions.test.js'.

**UT123**

**Test items:** Set Seed Action  
Test if an error is thrown if infinity is given as a seed  
**Input specifications:**  
['seed' => \(\infty\)]  
**Output specifications:** Throw 'invalid argument'  
**Environmental Needs:**

• The test case is stored in file 'explore.actions.test.js'.

**UT124**

**Test items:** Set Seed Action  
Test whether an error is thrown if a non integer is given  
**Input specifications:**  
['seed' => Object]  
**Output specifications:** Throw 'invalid argument'  
**Environmental Needs:**

• The test case is stored in file 'explore.actions.test.js'.

**UT125**

**Test items:** Set Sort Action  
Test whether the right sorting action is created.  
**Input specifications:**  
['sortType' => 'banaan']  
**Output specifications:** SET_SORT  
with ['sortType' => 'banaan']  
**Environmental Needs:**

• The test case is stored in file 'explore.actions.test.js'.

The test case is stored in file 'explore.actions.test.js'.

Valedictorian | Unit Test Plan
UT126

Test items: Set Filter Action
Test whether the right filter action is created
Input specifications: ['activeFilters' => 'banaan']
Output specifications: SET_FILTER
with ['activeFilters' => 'banaan']
Environmental Needs:

- The test case is stored in file 'explore.actions.test.js'.

UT127

Test items: Reducer Initial State
Test whether the initial state is correct
Input specifications: -
Output specifications: [{'tiles' => []}, 'from' => 0, 'to' => 0, 'mediaTypes' => [mediatypes.video, mediatypes.picture], 'activeFilters' => {'tags' => new Set(), 'mediaTypes' => new Set()}, 'sort-Type' => 'random', 'seed' => 0.12234 ]
Environmental Needs:

- The test case is stored in file 'explore.reducers.test.js'.

UT128

Test items: Reducer Expand Viewable Tiles
Test whether the reducer correctly expands the viewable tiles
Input specifications: EXPAND_EXPLORE_TILES_VIEWABLE
with ['to' => 10]
Output specifications: initialState with ['to' => initialState.to + 10]
Environmental Needs:

- The test case is stored in file 'explore.reducers.test.js'.

UT129

Test items: Reducer Set Seed
Test whether the reducer correctly sets the seed
Input specifications: SET_SEED
with ['seed' => 10]
Output specifications: initialState with ['seed' => 10]
Environmental Needs:

- The test case is stored in file 'explore.reducers.test.js'.


**UT130**

**Test items:** Reducer Set Sort  
Test whether the reducer correctly sets the sort setting  
**Input specifications:** SET_SORT with ['sortType' => 'some string']  
**Output specifications:** initialState with ['sortType' => 'some string']  
**Environmental Needs:**  
- The test case is stored in file 'explore.reducers.test.js'.

**UT131**

**Test items:** Reducer Set Filter  
Test whether the reducer correctly set the filter setting  
**Input specifications:** SET_FILTER with ['activeFilters' => 'some string']  
**Output specifications:** initialState with ['activeFilters' => 'some string']  
**Environmental Needs:**  
- The test case is stored in file 'explore.reducers.test.js'.

### 3.2.3 LIVE STREAM MODULE

**UT132**

**Test items:** Get Stream Action  
Test successful retrieval of a stream  
**Input specifications:** None  
**Output specifications:** GET_STREAM_REQUEST with ['id' => 9999]  
GET_STREAM_SUCCESS with ['id' => 9999, 'streamData' => mockedStreamData]  
**Environmental Needs:**  
- The test case is stored in file 'livestream.actions.test.js'.  
- The api route 'api/livestreams/9999' is mocked to respond with the expected stream data.

**UT133**

**Test items:** Get Stream Action  
Test unsuccessful retrieval of a stream  
**Input specifications:** None  
**Output specifications:** GET_STREAM_REQUEST with ['id' => 9999]
GET_STREAM_FAILURE
with ['id' => 9999, 'error' => 'Request failed with status code 404']

Environmental Needs:

- The test case is stored in file 'livestream.actions.test.js'.
- The api route 'api/livestreams/9999' is mocked to respond with a 404 error

UT134

Test items: Get Stream Request
Test whether a correctly stated input results in the correct output

Input specifications: None

Output specifications: GET_STREAM_REQUEST
with ['id' => 9999]

Environmental Needs:

- The test case is stored in file 'livestream.actions.test.js'.

UT135

Test items: Get Stream Request
Test whether a incorrectly stated input results in an exception being thrown

Input specifications: Four tests with input ['id' => [id]] with [id] where in the four tests [id] is replaced by nothing, 'a', '∞', NaN.

Output specifications: The action creator should throw an exception

Environmental Needs:

- The test case is stored in file 'livestream.actions.test.js'.

UT136

Test items: Get Stream Success
Test whether a correctly stated input results in the correct output

Input specifications: None

Output specifications: GET_STREAM_SUCCESS
with ['id' => 9999, 'streamData' => passedData]

Environmental Needs:

- The test case is stored in file 'livestream.actions.test.js'.

Valedictorian | Unit Test Plan
**UT137**

**Test items:** Get Stream Success
Test whether an incorrectly stated input results in an exception being thrown

**Input specifications:** Four tests with input `['id' => id, 'data' => anyStreamData]` with `id` where in the four tests `id` is replaced by nothing, `a`, `'\infty'`, NaN.

**Output specifications:** The action creator should throw an exception

**Environmental Needs:**
- The test case is stored in file 'livestream.actions.test.js'.

**UT138**

**Test items:** Get Stream Failure
Test whether a correctly stated input results in the correct output

**Input specifications:** None

**Output specifications:** GET_STREAM_FAILURE
with `['id' => 9999, 'error' => givenError]`

**Environmental Needs:**
- The test case is stored in file 'livestream.actions.test.js'.

**UT139**

**Test items:** Get Stream Failure
Test whether an incorrectly stated input results in an exception being thrown

**Input specifications:** Four tests with input `['id' => id]` with `id` where in the four tests `id` is replaced by nothing, `a`, `'\infty'`, NaN.

**Output specifications:** The action creator should throw an exception

**Environmental Needs:**
- The test case is stored in file 'livestream.actions.test.js'.

**UT140**

**Test items:** Save Link Action

**Input specifications:** None

**Output specifications:** SAVE_LINK_REQUEST
with `['teamId' => 9999, 'contentId' => '123456789-A']`
SAVE_LINK_SUCCESS
with `['teamId' => 9999, 'contentId' => '123456789-A', 'streamData' => mockedStreamData]`

**Environmental Needs:**
- The test case is stored in file 'livestream.actions.test.js'.
The api route `/api/livestreams/9999/123456789-A` is mocked to respond with the posted stream data

**UT141**

**Test items:** Save Link Action  
**Input specifications:** None  
**Output specifications:**  
- `SAVE_LINK_REQUEST` with `{'teamId' => 9999, 'contentId' => '123456789-A'}`  
- `SAVE_LINK_FAILURE` with `{'teamId' => 9999, 'contentId' => '123456789-A', 'error' => 'Request failed with status code 404'}`  

**Environmental Needs:**  
- The test case is stored in file `livestream.actions.test.js`.  
- The api route `/api/livestreams/9999/123456789-A` is mocked to respond with a 404 error

**UT142**

**Test items:** Save Link Request  
should create an action to notify the save link request has started  
**Input specifications:** `saveLinkRequest()`  
**Output specifications:** `SAVE_LINK_REQUEST`  
**Environmental Needs:**  
- The test case is stored in file `livestream.actions.test.js`.

**UT143**

**Test items:** Save Link Request  
Too few arguments.  
**Input specifications:** `saveLinkRequest(streamId)`  
**Output specifications:** Throws error  
**Environmental Needs:**  
- The test case is stored in file `livestream.actions.test.js`.

**UT144**

**Test items:** Save Link Request  
No arguments.  
**Input specifications:** `saveLinkRequest()`
Output specifications: Throws error
Environmental Needs:

- The test case is stored in file 'livestream.actions.test.js'.

**UT145**

Test items: Save Link Request
illegal argument type
Input specifications: saveLinkRequest(IllegalArgumentException), IllegalArgumentException is either: string, int, infinite, NaN
Output specifications: Throws error
Environmental Needs:

- The test case is stored in file 'livestream.actions.test.js'.

**UT146**

Test items: Save Link Success
should create an action to notify the save livestream link request was successful
Input specifications: saveLinkSuccess()
Output specifications: SAVE_LINK_SUCCESS
Environmental Needs:

- The test case is stored in file 'livestream.actions.test.js'.

**UT147**

Test items: Save Link Success
illegal argument type
Input specifications: saveLinkSuccess(IllegalArgumentException), IllegalArgumentException is either: string, int, infinite, NaN
Output specifications: Throws error
Environmental Needs:

- The test case is stored in file 'livestream.actions.test.js'.

**UT148**

Test items: Save Link Failure
Test whether a correctly stated input results in the correct output
Input specifications: ['streamId' => 9999, 'contentId' => '123456789-A', 'error' => ['error' => ['error' => test error']]]
Output specifications: SAVE_LINK_FAILURE
Environmental Needs:

- The test case is stored in file 'livestream.actions.test.js'.

**UT149**

**Test items:** Save Link Failure

Test whether a incorrectly stated input results in the correct output

**Input specifications:** Four tests with input `['streamId' => 9999, 'contentId' => '123456789-A', 'error' => ['error' => 'test error']]` where for each of the four tests the value of 'contentId' is replaced by 'a', true, Infinity and NaN

**Output specifications:** The action creator should throw an exception

Environmental Needs:

- The test case is stored in file 'livestream.actions.test.js'.

**3.2.4 PICTURE MODULE**

**UT150**

**Test items:** Get Pictures Action

Test being able to successfully receive pictures.

**Input specifications:**

**Output specifications:**

- GET_ALL_PICTURES_REQUEST
- GET_ALL_PICTURES_SUCCESS
with `['success' => true]`

Environmental Needs:

- The api route '/api/pictures' is mocked to respond to a post request with a 200 status code with `['success' => true]` as response body.

- The test case is stored in file 'picture.actions.test.js'.

**UT151**

**Test items:** Get Pictures Action

Test actions being generated upon an server error when trying to receive pictures.

**Input specifications:**

**Output specifications:**

- GET_ALL_PICTURES_REQUEST
- GET_ALL_PICTURES_FAILURE

Environmental Needs:

- The api route '/api/pictures' is mocked to respond to a post request with a 422 status code with `['success' => false]` as response body.
The test case is stored in file ‘picture.actions.test.js’.

**UT152**

**Test items:** Get Picture Ids Action  
Test being able to successfully receive the picture ids.  

**Input specifications:** -  

**Output specifications:**  
- GET_ALL_PICTURES_IDS_REQUEST  
- GET_ALL_PICTURES_IDS_SUCCESS  
with ['success' => true]  

**Environmental Needs:**  
- The api route '/api/pictures/id' is mocked to respond to a post request with a 200 status code with ['success' => true] as response body.  
- The test case is stored in file ‘picture.actions.test.js’.

**UT153**

**Test items:** Get Picture Ids Action  
Test actions being generated upon an server error when trying to receive pictures.  

**Input specifications:** -  

**Output specifications:**  
- GET_ALL_PICTURES_REQUEST  
- GET_ALL_PICTURES_FAILURE  

**Environmental Needs:**  
- The api route '/api/pictures/id' is mocked to respond to a post request with a 422 status code with ['success' => false] as response body.  
- The test case is stored in file ‘picture.actions.test.js’.

3.2.5 **QA MODULE**

**UT154**

**Test items:** Get Questions Action  
Test a successful receive of questions  

**Input specifications:** -  

**Output specifications:**  
- QUESTIONS_RECEIVED  
with a list of the mocked questions and answers  

**Environmental Needs:**  
- The test case is stored in file ‘qa.actions.test.js’.  
- The api route 'api/questions' is mocked to return a list of questions and answers
**UT155**

**Test items:** Get Questions Action  
Test an unsuccessful receive of questions  
**Input specifications:** -  
**Output specifications:** QUESTION_RECEIVED_FAIL  
**Environmental Needs:**
- The test case is stored in file 'qa.actions.test.js'.  
- The api route 'api/questions' is mocked to return with status code 422

**UT156**

**Test items:** Initial State  
Check whether the initial state is correct  
**Input specifications:** -  
**Output specifications:** ['questions' => [], 'answers' => []]  
**Environmental Needs:**
- The test case is stored in file 'qa.reducer.test.js'.

**UT157**

**Test items:** Receive Questions  
Test whether received questions are processed properly  
**Input specifications:** QUESTIONS_RECEIVED  
with ['questions' => newQuestion, 'answers' => []]  
**Output specifications:** initialState with ['questions' => newQuestion]  
**Environmental Needs:**
- The test case is stored in file 'qa.reducer.test.js'.

**UT158**

**Test items:** Receive Team Answers  
Test whether received team answer are processed properly  
**Input specifications:** GET_TEAM_ANSWERS_SUCCESS  
with ['answers' => theAnswer]  
**Output specifications:** initialState with ['answers' => theAnswer]  
**Environmental Needs:**
- The test case is stored in file 'qa.reducer.test.js'.
**UT159**

**Test items:** Receive Team Answers  
Test whether received multiple team answers are processed  
**Input specifications:** GET_TEAM_ANSWERS_SUCCESS  
with ['answers' => allTheAnswers]  
**Output specifications:** initialState with ['answers' => allTheAnswers]  
**Environmental Needs:**

- The test case is stored in file 'qa.reducer.test.js'.

---

**3.2.6 SENSORDATA MODULE**

**UT160**

**Test items:** Get Sensor Data Action  
creates SENSOR_DATA_RETRIEVED when fetching sensordata has been done  
**Input specifications:** retrieveSensorData()  
**Output specifications:** SENSOR_DATA_RETRIEVED  
**Environmental Needs:**

- The test case is stored in file 'sensordata.actions.test.js'.

---

**UT161**

**Test items:** Get Sensor Data Action  
creates SENSOR_DATA_ERROR when fetching sensordata fails  
**Input specifications:** retrieveSensorData() with a false response.  
**Output specifications:** SENSOR_DATA_ERROR  
**Environmental Needs:**

- The test case is stored in file 'sensordata.actions.test.js'.

---

**3.2.7 TEAM MODULE**

**UT162**

**Test items:** Get Teams Action  
Test being able to successfully receive teams.  
**Input specifications:** -  
**Output specifications:** GET_TEAMS_REQUEST  
GET_TEAMS_SUCCESS  
with ['teams' => true, 'sensus' => false]  
**Environmental Needs:**
• The api route '/api/teams' is mocked to respond to a post request with a 200 status code with ['teams' => true, 'sensus' => false] as response body.

• The test case is stored in file 'team.actions.test.js'.

**UT163**

**Test items:** Get Teams Action
Test actions being generated upon an server error when trying to receive teams.

**Input specifications:** -

**Output specifications:** GET_TEAMS_REQUEST
GET_TEAMS_FAILURE

**Environmental Needs:**

• The api route '/api/teams' is mocked to respond to a post request with a 422 status code with ['success' => false] as response body.

• The test case is stored in file 'team.actions.test.js'.

**UT164**

**Test items:** Get Teams Questions and Answers Action
Test being able to successfully receive questions and answers for teams.

**Input specifications:** [teamId => $id] where $id from 1 .. 10.

**Output specifications:** GET_TEAM_QUESTIONS_REQUEST
GET_TEAM_QUESTIONS_SUCCESS
GET_TEAM_QUESTIONS_REQUEST
GET_TEAM_QUESTIONS_SUCCESS
GET_TEAM_ANSWERS_REQUEST
GET_TEAM_ANSWERS_SUCCESS
GET_TEAM_ANSWERS_REQUEST
GET_TEAM_ANSWERS_SUCCESS

with ['teamId' => $id, 'questions' => true]

with ['teamId' => $id, 'answers' => false]

**Environmental Needs:**

• The api route '/api/team/$id/questions' (where $id from 1 .. 10) is mocked to respond to a post request with a 200 status code with ['questions' => true, 'answers' => false] as response body.

• The test case is stored in file 'team.actions.test.js'.

**UT165**

**Test items:** Get Teams Questions and Answers Action
Test actions being generated upon an server error when trying to receive teams.

**Input specifications:** [teamId => 1]
Output specifications: GET_TEAM_QUESTIONS_REQUEST
GET_TEAM_ANSWERS_REQUEST
GET_TEAM_QUESTIONS_FAILURE
GET_TEAM_ANSWERS_FAILURE

Environmental Needs:

- The api route '/api/team/1/questions' is mocked to respond to a post request with a 422 status code with ['success' => false] as response body.
- The test case is stored in file 'team.actions.test.js'.

**UT166**

Test items: Update Description Action
Test being able to successfully update description for team.

Input specifications: [teamId => $id, 'description' => “abc”] where $id from 1 .. 10.

Output specifications: UPDATED_DESCRIPTION
with ['teamId' => $id, 'description' => "abc"]

Environmental Needs:

- The test case is stored in file 'team.actions.test.js'.

**UT167**

Test items: Save Description Action
Test being able to successfully save a description for a team.

Input specifications: [teamId => $id, description => “abc”] where $id from 1 .. 10.

Output specifications: SAVE_DESCRIPTION_REQUEST
SAVE_DESCRIPTION_SUCCESS
with ['teamId' => $id, 'description' => "abc"]

Environmental Needs:

- The api route '/api/teams/$id' (where $id from 1 .. 10) is mocked to respond to a put request with a 200 status code with [] as response body.
- The test case is stored in file 'team.actions.test.js'.

**UT168**

Test items: Save Description Action
Test actions being generated upon an server error when trying to receive teams.

Input specifications: [teamId => 1]

Output specifications: SAVE_DESCRIPTION_REQUEST
SAVE_DESCRIPTION_FAILURE

Environmental Needs:

- The api route '/api/teams/1' is mocked to respond to a post request with a 422 status code with ['success' => false] as response body.
- The test case is stored in file 'team.actions.test.js'.

3.2.8 VIDEO MODULE

UT169

Test items: Get Videos Action
Test being able to successfully receive videos.

Input specifications: -
Output specifications: GET_ALL_VIDEOS_REQUEST
GET_ALL_VIDEOS_SUCCESS
with ['success' => true]

Environmental Needs:

- The api route '/api/videos' is mocked to respond to a post request with a 200 status code with ['success' => true] as response body.
- The test case is stored in file 'video.actions.test.js'.

UT170

Test items: Get Videos Action
Test actions being generated upon an server error when trying to receive videos.

Input specifications: -
Output specifications: GET_ALL_VIDEOS_REQUEST
GET_ALL_VIDEOS_FAILURE

Environmental Needs:

- The api route '/api/videos' is mocked to respond to a post request with a 422 status code with ['success' => false] as response body.
- The test case is stored in file 'video.actions.test.js'.

UT171

Test items: Get Picture Ids Action
Test being able to successfully receive the picture ids.

Input specifications: -
Output specifications: GET_ALL_VIDEOS_IDS_REQUEST
GET_ALL_VIDEOS_IDS_SUCCESS with ['success' => true]

Environmental Needs:

- The api route '/api/videos/id' is mocked to respond to a post request with a 200 status code with ['success' => true] as response body.

- The test case is stored in file 'video.actions.test.js'.

UT172

Test items: Get Picture Ids Action
Test actions being generated upon an server error when trying to receive pictures.

Input specifications: -

Output specifications:
GET_ALL_PICTURES_REQUEST
GET_ALL_PICTURES_FAILURE

Environmental Needs:

- The api route '/api/pictures/id' is mocked to respond to a post request with a 422 status code with ['success' => false] as response body.

- The test case is stored in file 'picture.actions.test.js'.

3.2.9 VOTE MODULE

UT173

Test items: Cast Vote Action
Test whether a correct response from server results in the correct actions being dispatched.

Input specifications: ['team_a' => ['id' => 2], 'team_b' => ['id' => 7]]

Output specifications:
CAST VOTE REQUEST
with ['teamA' => ['id' => 2], 'teamB' => ['id' => 7]]

CAST VOTE SUCCESS
with ['idTeamA' => 2, 'idTeamB' => 7]

Environmental Needs:

- The api route '/api/votes/cast' is mocked to respond to a post request with a 200 status code with ['team_a' => 2, 'team_b' => 7] as response body.

- The test case is stored in file 'vote.actions.test.js'.

UT174

Test items: Cast Vote Action
Test whether a failed response from server results in the correct actions being dispatched.

Input specifications: ['team_a' => ['id' => 2], 'team_b' => ['id' => 7]]
Output specifications: CAST_VOTE_REQUEST
with ['teamA' => ['id' => 2], 'teamB' => ['id' => 7]]
CAST_VOTE_FAILURE
with ['idTeamA' => 2, 'idTeamB' => 7, 'error' => 'Request failed with status code 401']

Environmental Needs:
- The api route '/api/votes/cast' is mocked to respond to a post request with a 401 status code.
- The test case is stored in file 'vote.actions.test.js'.

UT175

Test items: Cast Vote Request
Test whether a correctly stated input results in the correct output
Input specifications: ['team_a' => ['id' => 3], 'team_b' => ['id' => 6]]
Output specifications: CAST_VOTE_SUCCESS
with ['idTeamA' => 3, 'idTeamB' => 6]
Environmental Needs:
- The test case is stored in file 'vote.actions.test.js'.

UT176

Test items: Cast Vote Request
Test whether a incorrectly stated input results in the correct output
Input specifications: Four tests with input ['team_a' => ['id' => 3], 'team_b' => ['id' => 6]] where for each of the four tests one of the values of 'team_a' or 'team_b' is replaced by 3, undefined, Infinity or NaN
Output specifications: The action creator should throw an exception
Environmental Needs:
- The test case is stored in file 'vote.actions.test.js'.

UT177

Test items: Cast Vote Success
Test whether a correctly stated input results in the correct output
Input specifications: ['idTeamA' => 3, 'idTeamB' => 6]
Output specifications: CAST_VOTE_SUCCESS
with ['idTeamA' => 3, 'idTeamB' => 6]
Environmental Needs:
- The test case is stored in file 'vote.actions.test.js'.
UT178

Test items: Cast Vote Success
Test whether a incorrectly stated input results in the correct output
Input specifications: Four tests with input ['idTeamA' => 3, 'idTeamB' => 6] where for each of the four tests one of the values of ‘team_a’ or ‘team_b’ is replaced by 3, undefined, Infinity or NaN
Output specifications: The action creator should throw an exception
Environmental Needs:

- The test case is stored in file ’vote.actions.test.js’.

UT179

Test items: Cast Vote Failure
Test whether a correctly stated input results in the correct output
Input specifications: ['team_a' => ['id' => 3], 'team_b' => ['id' => 6], 'error' => ‘test error’]
Output specifications: CAST_VOTE_FAILURE with ['teamA' => ['id' => 3], 'teamB' => ['id' => 6], 'error' => ‘test error’]
Environmental Needs:

- The test case is stored in file ’vote.actions.test.js’.

UT180

Test items: Cast Vote Failure
Test whether a incorrectly stated input results in the correct output
Input specifications: Four tests with input ['team_a' => ['id' => 3], 'team_b' => ['id' => 6], 'error' => ‘test error’] where for each of the four tests one of the values of ‘team_a’ or ‘team_b’ is replaced by 3, undefined, Infinity or NaN
Output specifications: The action creator should throw an exception
Environmental Needs:

- The test case is stored in file ’vote.actions.test.js’.

UT181

Test items: Get Votes Action
Test whether a correctly stated input results in correct output when the server responds with a response conform with a success.
Input specifications: None
Output specifications: GET_VOTES_REQUEST
GET_VOTES_SUCCESS
with ['totalVoteCount' => 10, 'teamVotes' => ['1' => 4, '2' => 40, '3' => 0]]

**Environmental Needs:**

- The test case is stored in file 'vote.actions.test.js'.

- The api route '/api/votes/counts' is mocked to respond to a get request with a 200 status code and with ['totalVoteCount' => 10, 'teamVotes' => ['1' => 4, '2' => 40, '3' => 0]] as response body.

### UT182

**Test items:** Get Votes Action  
Test whether a correctly stated input results in correct output when the server responds with a response conform with a failure.  
**Input specifications:** None  
**Output specifications:** GET_VOTES_REQUEST  
GET_VOTES_FAILURE  
with ['error' => 'Request failed with error code 400']

**Environmental Needs:**

- The test case is stored in file 'vote.actions.test.js'.

- The api route '/api/votes/counts' is mocked to respond to a get request with a 400 status code.

### UT183

**Test items:** Cast Vote Request  
Test whether a correctly stated input results in the correct output  
**Input specifications:** None  
**Output specifications:** CAST_VOTE_REQUEST  
CAST_VOTE_FAILURE  
with ['totalVoteCount' => 100, 'teamVotes' => ['0' => 1, '1' => 5, '2' => 0]]

**Environmental Needs:**

- The test case is stored in file 'vote.actions.test.js'.

### UT184

**Test items:** Cast Vote Success  
Test whether a correctly stated input results in the correct output  
**Input specifications:** ['totalVoteCount' => 100, 'teamVotes' => ['0' => 1, '1' => 5, '2' => 0]]  
**Output specifications:** CAST_VOTE_SUCCESS  
with ['totalVoteCount' => 100, 'teamVotes' => ['0' => 1, '1' => 5, '2' => 0]]

**Environmental Needs:**

- The test case is stored in file 'vote.actions.test.js'.
UT185

Test items: Cast Vote Failure
Test whether a correctly stated input results in the correct output
Input specifications: ['error' => 'test error']
Output specifications: CAST_VOTE_FAILURE with ['error' => 'test error']
Environmental Needs:
- The test case is stored in file ‘vote.actions.test.js’.

UT186

Test items: Add To Team Votes
Test the addToTeamVotes function.
Input specifications: ['votes' => [0 => 100, 1 => 6, 2 => 3, 3 => -1, 7 => 22374823, 10 => 52734895793], 'ids' => [0, 1, 2, 3, 7, 10], 'values' => [1, 1, 1, 1, 1, 1]]
Output specifications: [0 => 101, 1 => 7, 2 => 4, 3 => 0, 7 => 22374824, 10 => 52734895794]
Environmental Needs:
- The test case is stored in file ‘vote.actions.test.js’.

UT187

Test items: Add To Team Votes
Test whether the addToTeamVotes function does not mutate its 'votes' argument.
Input specifications: ['votes' => [0 => 100, 1 => 6, 2 => 3, 3 => -1, 7 => 22374823, 10 => 52734895793], 'ids' => [0, 1, 2, 3, 7, 10], 'values' => [1, 1, 1, 1, 1, 1]]
Output specifications: The 'Votes' argument is not mutated
Environmental Needs:
- The test case is stored in file ‘vote.actions.test.js’.

UT188

Test items: Add To Team Votes
Test whether the addToTeamVotes.
Input specifications: ['votes' => [0 => 100, 1 => 6, 2 => 3, 3 => -1, 7 => 22374823, 10 => 52734895793], 'ids' => [0, 1, 2, 3, 7, 10], 'values' => [-1, 0, 1, -201, 1, 1]]
Output specifications: [0 => 99, 1 => 6, 2 => 4, 3 => -202, 7 => 22374824, 10 => 52734895794]
Environmental Needs:
- The test case is stored in file ‘vote.actions.test.js’.
Test items: Add To Team Votes
Test whether the addToTeamVotes.

Input specifications: ['votes' => [0 => 100, 1 => 6, 2 => 3, 3 => -1, 7 => 22374823, 10 => 52734895793], 'ids' => [7], 'values' => [1501023]]

Output specifications: [0 => 100, 1 => 6, 2 => 3, 3 => -1, 7 => 23875846, 10 => 52734895793]

Environmental Needs:
- The test case is stored in file ‘vote.actions.test.js’.
4 TEST PROCEDURES

To run the unit tests, the following procedure should be followed.

1. Open a terminal in the application’s root folder.
2. To run the client tests, run the command “npm run test”.
3. To run the server tests, run the command “vendor/bin/phpunit”.
4. The results of all tests are shown in the command line.

4.1 RUNNING TESTS WITH COVERAGE AND DETAILED RESULTS

To run the client tests with coverage, run the following command:

```
node_modules/.bin/jest --coverage --coverageDirectory <directory/for/coverage/results>
```

The coverage results will then be available in the given folder. Specifically, they can be viewed by opening the index.html in the “Icov-report” subfolder.

To run the server tests with coverage, run the following command:

```
vendor/bin/phpunit --coverage-html <coverage/report/folder> --testdox-html <path/to/testreport.html>
```

Replace the path to the coverage results folder and the testreport file to the desired output locations.

After the tests finish running, there should be a test report at the given file path. There should also be a more comprehensive coverage report in the provided output folder for that report.
5 TEST COVERAGE

This chapter provides screenshots of the test coverage results that have been acquired using the PHPUnit and Jest testing frameworks. To get the full results in interactive form, the steps described in the previous chapter should be followed. However, the screenshots below should provide a complete overview.

5.1 SERVER CODE

For the server code, phpunit is used to generate coverage information. The coverage of all the server code tests are shown below.

APP FOLDER

The App folder contains all the PHP code for the application. The Globals folder contains only Enums and an abstract class `BasicEnum` which is used to produce enum-like classes, since PHP lacks a native implementation for Enums. This explains the relatively low coverage. Events, Providers and HTTP contain the core classes of our server application, which is why we ensured a high coverage for these folders. Finally, note that Models contains only definitions for relations in each model (to other models), and the low coverage is a result of many relations existing but being unused in the controllers (or elsewhere in the back-end). Important to note is that no "real" logic is written in any of the models.

<table>
<thead>
<tr>
<th></th>
<th>Lines</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total</td>
<td>632 / 713</td>
</tr>
<tr>
<td>Events</td>
<td>10 / 10</td>
</tr>
<tr>
<td>Exceptions</td>
<td>13 / 13</td>
</tr>
<tr>
<td>Globals</td>
<td>10 / 15</td>
</tr>
<tr>
<td>Http</td>
<td>533 / 592</td>
</tr>
<tr>
<td>Models</td>
<td>22 / 39</td>
</tr>
<tr>
<td>Providers</td>
<td>44 / 44</td>
</tr>
</tbody>
</table>

CONTROLLERS

The controller classes contain the logic of the application and are therefore the most important to test. The figure below shows the coverage results for all the controllers. Note that
Controller is the parent class of all other controllers, and contains no testable code as it only defines traits. For AuthController, the relatively low coverage is a result of two methods that require communication with the Google OAuth API (or a highly advanced method to mock this interaction). Nevertheless, the login functionality was extensively tested manually.

### COVERAGE STATISTICS

The figures below shows the distribution of method and class coverage for the server code tests respectively.
### 5.2 CLIENT CODE

For the client code, Jest is used to generate coverage information. The coverage of all the client code tests are shown below. It should be noted that the client code tests do not cover all the Javascript code. It has been our conscious choice to focus testing the server-side code, for security reasons. Furthermore, it is also much more tedious to test all the user interface code and it only provides so much value. For that reason, and considering the limited amount of time in the project, the unit tests mostly only cover the actions and reducers of the main website and some important other functions. The coverage for this functionality is shown below.

#### MAIN/COMMON/ACTIONS

<table>
<thead>
<tr>
<th>File</th>
<th>Statements</th>
<th>Branches</th>
<th>Functions</th>
<th>Lines</th>
</tr>
</thead>
<tbody>
<tr>
<td>app_actions.js</td>
<td>100%</td>
<td>15/15</td>
<td>100%</td>
<td>15/15</td>
</tr>
<tr>
<td>explore_actions.js</td>
<td>83.33%</td>
<td>15/18</td>
<td>66.67%</td>
<td>15/18</td>
</tr>
</tbody>
</table>

#### MAIN/COMMON/REDUCERS

<table>
<thead>
<tr>
<th>File</th>
<th>Statements</th>
<th>Branches</th>
<th>Functions</th>
<th>Lines</th>
</tr>
</thead>
<tbody>
<tr>
<td>explore_reducer.js</td>
<td>77.78%</td>
<td>7/9</td>
<td>75%</td>
<td>7/9</td>
</tr>
</tbody>
</table>

#### SHARED/UTIL

<table>
<thead>
<tr>
<th>File</th>
<th>Statements</th>
<th>Branches</th>
<th>Functions</th>
<th>Lines</th>
</tr>
</thead>
<tbody>
<tr>
<td>globals.js</td>
<td>100%</td>
<td>5/5</td>
<td>100%</td>
<td>1/1</td>
</tr>
<tr>
<td>mediatypes.js</td>
<td>100%</td>
<td>2/2</td>
<td>100%</td>
<td>0/0</td>
</tr>
</tbody>
</table>
# SHARED/UTIL/REDUX/ACTIONS

<table>
<thead>
<tr>
<th>File</th>
<th>Statements</th>
<th>Branches</th>
<th>Functions</th>
<th>Lines</th>
</tr>
</thead>
<tbody>
<tr>
<td>livstream actions.js</td>
<td>94%</td>
<td>50%</td>
<td>94.74%</td>
<td>94%</td>
</tr>
<tr>
<td>picture actions.js</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>qa actions.js</td>
<td>20.41%</td>
<td>100%</td>
<td>15.91%</td>
<td>15.13%</td>
</tr>
<tr>
<td>sensordata actions.js</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>team actions.js</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>video actions.js</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>vote actions.js</td>
<td>67.31%</td>
<td>72.22%</td>
<td>72.73%</td>
<td>72.22%</td>
</tr>
</tbody>
</table>

# SHARED/UTIL/REDUX/REDUCERS

<table>
<thead>
<tr>
<th>File</th>
<th>Statements</th>
<th>Branches</th>
<th>Functions</th>
<th>Lines</th>
</tr>
</thead>
<tbody>
<tr>
<td>qa reducer.js</td>
<td>27.03%</td>
<td>20%</td>
<td>25%</td>
<td>25.71%</td>
</tr>
<tr>
<td>vote reducer.js</td>
<td>38.89%</td>
<td>8.33%</td>
<td>100%</td>
<td>38.89%</td>
</tr>
</tbody>
</table>
6 TEST REPORTS

6.1 SERVER TESTS

The figures below show all the server test and whether they passed (they all passed).

**Unit\AnswerController**

- ✓ Legal post
- ✓ Legal post with link
- ✓ Unauthorized post
- ✓ Legal answer approve
- ✓ Illegal answer approve
- ✓ Delete answer

**s\Feature\Authentication**

- ✓ Authenticated role
- ✓ Team member role
- ✓ Team member role fail
- ✓ Admin role
- ✓ Super admin
- ✓ Super admin fail
- ✓ Unauthorized
- ✓ Unauthorized fail
- ✓ User role hierarchy
- ✓ Banned
- ✓ Authenticated
- ✓ Unauthenticated
- ✓ Logout
- ✓ Redirect

**Unit\LivestreamController**

- ✓ Update livestream success
- ✓ Create livestream success
- ✓ Unauthorized update
- ✓ Invalid content
- ✓ Non existing team
- ✓ Get stream
Unit\MediaController

✓ Get all media found media subset database
✓ Single media success
✓ Get all database media subset found media
✓ Store invalid input
✓ Proper post media input
✓ Delete
✓ Alter video
✓ Alter picture

s\Feature\PictureController

✓ Get picture info

Unit\QuestionController

✓ Legal post
✓ Legal post with video
✓ Illegal post
✓ Post to invalid team
✓ Post without description
✓ Post without title
✓ Legal approve
✓ Illegal approve
✓ Illegal approve by team member
✓ Legal claim
✓ Change to pending delete
✓ Legal delete
✓ Illegal delete
✓ Get approved
✓ Get all
✓ Get from team
✓ Unauthorized get all

s\Unit\SensorDataController

✓ Get j s o n structure
✓ Get admin j s o n structure
✓ Get admin false if not admin
✓ Retrieve does return assigned concentration when admin
✓ Update single value admin
✓ Admin required for update
✓ Retrieve does not return row when not public
✓ Retrieve returns row when public

s\Feature\TagController

✓ Get tags
s\Unit\TeamController

✓ Authenticated privilege
✓ Regular privilege
✓ Empty description
✓ Text match
✓ Invalid team
✓ Get teams
✓ Get team info

s\Feature\UserManagement

✓ All combinations
✓ Select no team
✓ Unauthorized edit
✓ Non existing user
✓ Non existing role
✓ Non existing team
✓ Ban
✓ Missing arguments ban
✓ Invalid ban
✓ Super admin ban
✓ Get users
✓ Unauthorized get users

s\Feature\VideoController

✓ Get video info
✓ Invalid video
✓ Missing config

s\Feature\ViewController

✓ Get index
✓ Get dashboard
✓ Unauthorized dashboard
6.2 CLIENT TESTS

As shown in the figure below, all the client tests passed.