## Requirements for Test　テスト要求

### Test items　テスト頭目

|  |  |  |  |
| --- | --- | --- | --- |
| No | Name of features and functions | Outline of features and functions | Note |
|  | Login | Sign in on PSM with admin or manager account. |  |
|  | Forgot Password to send request forgot password. |  |
|  |
|  |
|  |
|  |
|  | User Management  ( Admin Role ) | Search Manager of any Studio |  |
|  | Add Manager to any Studio |  |
|  | Edit Manager |  |
|  | Delete Manager |  |
|  | Studio Management  ( Admin Role ) | Search Studio |  |
|  | Add Studio |  |
|  | Edit Studio |  |
|  | Delete Studio |  |
|  | District Management  ( Admin Role ) | Search District |  |
|  | Add District |  |
|  | Edit District |  |
|  | Delete District |  |
|  | Studio Information  ( Manager Role ) | Edit Studio Infomation |  |
|  | Package Management  ( Manager Role ) | Search Package |  |
|  | Add Package |  |
|  | Edit Package |  |
|  | Delete Package |  |
|  | Picture Management  ( Manager Role ) | Search Picture |  |
|  | Add Picture |  |
|  | Edit Picture |  |
|  | Delete Picture |  |
|  | Upload Picture |  |
|  | Contract Management  ( Manager Role ) | Search Contract |  |
|  | Add Contract |  |
|  | Edit Contract |  |
|  | Delete Contract |  |
|  | Edit Contract Detail |  |
|  | Delete Contract Detail |  |
|  | View Picture |  |
|  | Add Contract Detail |  |
|  | Edit Customer Infomation |  |
|  | Home Page  ( Customer ) | View Sample Picture |  |
|  | View About US |  |
|  | Studio List Page  ( Customer ) | View Information of Studio |  |
|  | Studio Detail Page  ( Customer ) | View Sample Picture |  |
|  | View Package |  |
|  | Add Package to Cart |  |
|  | Cart Page  ( Customer ) | Remove Package on Cart |  |
|  | Buy Package |  |
|  | Show Contract Code |  |
|  | Album Page  ( Customer ) | Show Sample Picture |  |
|  | Contract Page  ( Customer ) | Enter Contract Code |  |
|  | Show Contract |  |
|  | View Picture |  |

Table 5.3.1

## Test strategy　テスト戦略

### Test types テストタ;イプ

PSM will be tested by 3 level

**Level 1 : Unit testing**

- Unit Testing will be done by developers.

- Testing all individual implemented methods, functions of XXXService.js files.

- Implemented function’s error message and database error message will be included in this test.

- When executing Unit Testing, if any bugs are finded, developers have to log bug on “Defect Log” file and fix it until it is correct.

Rule for filling test result:

|  |  |
| --- | --- |
| Test result pass | Pass |
| Test result fail | Fail |
| Do not test | Untested |
| Cannot test | N/A (Not available) |

**Level 2 : Integration testing**

* After finishing component testing, integration testing will be performed by testers.
* Material are integration test cases, high- level design and test tools.
* Integration test focuses on specific areas of use cases when all requirements are completed.
* Integration test should be performed to ensure all components incorporate well.

*Rule for filling test result:*

|  |  |
| --- | --- |
| Test result pass | Pass |
| Test result fail | Fail |
| Do not test | Untested |
| Cannot test | N/A (Not available) |

**Level 3: System testing**

* After finishing integration testing and developers collect all functions and items, testers will be performed system testing, it means doing test whole system.
* If any bugs are found, developers have to fix and testers will verify them. System test is ended only when test cases are passed and no bug is found.

*Rule for filling test result:*

|  |  |
| --- | --- |
| Test result pass | Pass |
| Test result fail | Fail |
| Do not test | Untested |
| Cannot test | N/A (Not available) |

### Function Testing　機能テスト

* Function Testing is a type of software testing whereby the system is tested against the functional requirements/ specifications.
* Functions are tested by feeding them input and examining the output. Functional testing ensures that the requirements are properly satisfied by application. This type of testing is not concerned with how processing occurs, but rather, with results of processing.

|  |  |
| --- | --- |
| Test Objective: | Ensure proper target-of-test functionality, including navigation, data entry, processing, and retrieval. |
| Technique: | Execute each use case, use-case flow, or function, using valid and invalid data, to verify the following:  -    The expected results occur when valid data is used.  -    The appropriate error or warning messages are displayed when invalid data is used.  -    Each business rule is properly applied.  - Use Test tool |
| Completion Criteria: | -   All planned tests have been executed.  -     All identified defects have been addressed and closed |
| Special Considerations: | Identify or describe those items or issues (internal or external) that impact the implementation and execution of function test |

Table 5.4.3 Function Testing

### User Interface Testing　ユーザーインターフェーステスト

* User interface testing, a testing technique used to identify the presence of defects is a product/software under test by using GUI.
* GUI testing evaluates design elements such as layout, colors, [fonts](http://whatis.techtarget.com/definition/font), font sizes, labels, text boxes, text formatting, captions, buttons, lists, icons, links and content.

|  |  |
| --- | --- |
| Test Objective: | Verify the following:  Navigation through the target-of-test properly reflects business functions and requirements, including window-to-window, field-to-field, and use of access methods (tab keys, mouse movements, accelerator keys)  Window objects and characteristics, such as menus, size, position, state, and focus conform to standards. |
| Technique: | Create or modify tests for each window to verify proper navigation and object states for each application window and objects. |
| Completion Criteria: | Each window successfully verified to remain consistent with benchmark version or within acceptable standard. |
| Special Considerations: | Not all properties for custom and third party objects can be accessed. |

Table 5.4.4 User Interface Testing

### API Testing

* Involves testing APIs directly to determine if they meet expectations for functionality, reliability, performance, and security. API testing will test all of individual implemented API of PSM API.
* Testing will mock all possible datasets and call to corresponding API with separate testing database by using Chaijs, SuperTest and Mocha.

### Test stages　テストステージ

| Type of Tests | Stage of Test | | |
| --- | --- | --- | --- |
| Unit Test | Integration Test | System Test |
| Function Test | X | X | X |
| User Interface Test | X |  | X |
| API Test |  | X |  |

Table 5.4.6 Test Stages

## Resource　リソース

### Human Resource　人的リソース

|  |  |  |  |
| --- | --- | --- | --- |
| Worker/Doer | Role | Specific Responsibilities/Comments | Location |
| DungPT | Test leader | Write Test Case, Test Report , Check bug | FPT University Vietnam |
| TrungNH | Tester | Fix Bug | FPT University Vietnam |
| VuCL | Tester | Fix Bug | FPT University Vietnam |

Table 5.5.1 Human Resource

### Test management　テスト管理

Test management is practice of organizing and controlling the process and artifacts required for testing effort.

The general goal of test management is to allow teams to plan, develop, execute, and assess all testing activities within the overall software development effort. This includes coordinating efforts of all those involved in the testing effort, tracking dependencies and relationships among test assets and, most importantly, defining, measuring, and tracking quality goals.

* **Test planning** is the overall set of tasks that address the questions of why, what, where, and when to test. The reason why a given test is created is called a test motivator (for example, a specific requirement must be validated). What should be tested is broken down into many test cases for a project. Where to test is answered by determining and documenting the needed software and hardware configurations. When to test is resolved by tracking iterations (or cycles, or time period) to the testing.
* **Test authoring** is a process of capturing the specific steps required to complete a given test. This addresses the question of how something will be tested. This is where somewhat abstract test cases are developed into more detailed test steps, which in turn will become test scripts (either manual or automated).
* **Test execution** entails running the tests by assembling sequences of test scripts into a suite of tests. This is a continuation of answering the question of how something will be tested (more specifically, how the testing will be conducted).
* **Test reporting** is how the various results of the testing effort are analyzed and communicated. This is used to determine the current status of project testing, as well as the overall level of quality of the application or system.

### Defect management　欠陥管理

Defect management is crucial to closing the loop between requirements, implementation and verification and validation. Traditional defect tracking management, implemented in a standalone fashion, can no longer address the complexity and pace of change in modern software development. Defect management processes must be tightly interlinked with all of the other software development processes. The defect management process contains the following elements:

* **Defect Discovery** – Identification and reporting of potential defects. The defect tracking software must be simple enough so that people will use it, but ensure that the minimum necessary information is captured. The information captured here should be enough to reproduce the defect and allow development to determine root cause and impact.
* **Defect Analysis & Prioritization** – The development team determines if the defect report corresponds to an actual defect, if the defect has already been reported, and what the impact and priority of the defect is. Prioritization and scheduling of the defect resolution is often part of the overall change management process for the software development organization.
* **Defect Resolution –** Here the development team determines the root cause, implements the changes needed to fix the defect, and documents the details of the resolution in the defect management software, including suggestions on how to verify the defect is fixed. In organizations using [Software Product Lines](http://www.mks.com/challenges/product-lines) approaches, or other shared component approaches, defect resolution may need to be coordinated across multiple branches of development.
* **Defect Verification** – The build containing the resolution to the defect is identified, and testing of the build is performed to ensure the defect truly has been resolved, and that the resolution has not introduced side effects or regressions. Once all affected branches of development have been verified as resolved, the defect can be closed.
* **Defect Communication** – This encompasses automatic generation of defect metrics for management reporting and process improvement purposes, as well as visibility into the presence and status of defects across all disciplines of the software development team.

## Test environment　たテスト環境

|  |  |  |
| --- | --- | --- |
| **Type of testing** | **Software** | **Hardware** |
| System test | * Microsoft Office Excel 2007. * Chrome Version 59.0.3071.115 | Personal computer for developing with the minimum configuration:   * Windows 7 Professional 32-bit * Intel® Core™ i5 * Installed memory (RAM): 4.00GB |
| Unit testing and API testing | * Chaijs v3.5.0 * Mocha v2.4.5 * SuperTest v1.2.0 | Personal computer for developing with the minimum configuration:   * MacOS High Sierra * 2.5GHz Intel Core i5 * Installed memory (RAM): 4.00GB |

Table 5.6: Test enviroment

## Test Milestones　テストマイルストーン

|  |  |  |
| --- | --- | --- |
| Milestone Task | Start Date | End Date |
| Create Test Plan | 10/02/2018 | 02/04/2018 |
| Review & update TP | 05/03/2018 | 02/04/2018 |
| Create Unit Test | 07/03/2018 | 02/04/2018 |
| Review & Update Unit Test | 09/03/2018 | 02/04/2018 |
| Create Integration Test case | 14/03/2018 | 02/04/2018 |
| Review & Update Integration TC | 20/03/2018 | 02/04/2018 |
| Create System Test case | 27/03/2018 | 02/04/2018 |
| Review & Update System TC | 01/04/2018 | 02/04/2018 |

Table 5.7 Test Milestones

## Deliverables　納品物

| No | Deliverables | Responsibilities | Delivered Date |
| --- | --- | --- | --- |
|  | Test Plan | DungPT,HuyNM | 10/02/2018 |
|  | Unit Test Cases | HuyNM | 07/03/2018 |
|  | Integration Test Cases | DungPT,HuyNM | 14/03/2018 |
|  | System Test cases | DungPT | 30/03/2018 |
|  | Test reports | DungPT,HuyNM | 02/04/2018 |

Table 5.8 Deliverables

## Test Case　テストケース

Refer to:

* PSM\_Unit TestCase\_v1.0 xlxs
  + PSM\_IntegrationTestCase\_v1.0.xlsx
  + PSM \_SystemTestCase\_v1.0.xlsx

Sample Test Cases :

**Login Page Test Cases :**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Test case id** | **Test cases** | **Priority** | **Preconditions** | **Input test data** | **Steps to be executed** | **Expected results** | **Pass / Fail /  Untested / Cannot test** | **Comments** |
| 1 | Test if user is able to login successfully. | A | User must be registered already | Correct ID Correct Password | 1. Enter correct ID and Password  2. Click Submit button | User must successfully login to the web page | Pass |  |
| 2 | Test if unregistered users is not able to login to the site | A |  | Incorrect ID Incorrect Password | 1. Enter incorrect ID and Password  2. Click Submit button | Proper error must be displayed and prompt to enter login again | Pass |  |
| 3 | Test with valid ID and empty Password field | B | User must be registered already | Valid ID Empty Password field | 1. Enter the valid ID and empty Password field   2. Click Submit button | Proper error must be displayed and prompt to enter login again | Pass |  |
| 4 | Test with empty ID field and valid Password field | B | User must be registered already | Empty ID field Valid Password | 1. Empty ID field.  2. Enter valid Password.  3. Click Submit button. | Proper error must be displayed and prompt to enter login again | Pass |  |
| 5 | Test with empty ID field and empty Password field | A |  |  | 1. Empty ID and Password field.  2. Click Submit button | Proper error must be displayed and prompt to enter login again | Pass |  |
| 6 | Check of the Password is masked on the screen i.e., Password must be in bullets or asterisks | B |  |  | 1. Enter anything in Password field. | The Password field should display the characters in asterisks or bullets such that the Password is not visible | Pass |  |
| 7 | After logging in try to copy/cut the Password and paste it on another screen(Passwords are usually in \* such that its not visible on the screen) | B |  |  | 1. Enter ID and Password in the respective fields. Copy the Password field's content (which is in \*s)  2. Paste the content on another screen | Password shouldn’t get pasted / Password should not be visible on the screen | Pass |  |
| 9 | Test if user is able to send forgot password request. | A | Exist user name and Studio mail | Enter exist user name and Studio mail | 1. Enter exist user name and Studio mail  2. Click Forgot Password button | User request must be send and successful notification  must be displayed | Pass |  |
| 10 | Test with empty User Name and empty Studio mail | A |  |  | 1. Empty user name and Studio mail  2. Click Forgot Password button | Proper error must be displayed and prompt to input again | Pass |  |
| 11 | Test with empty User Name and Correct Studio mail | A |  |  | 1. Empty user name  2. Enter Correct Studio mail   3. Click Forgot Password button | Proper error must be displayed and prompt to input again | Pass |  |
| 12 | Test with correct User Name and empty Studio mail | A |  |  | 1. Enter Correct user name  2. Empty Studio mail field  3. Click Forgot Password button | Proper error must be displayed and prompt to input again | Pass |  |

**User Management Test Cases :**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Test case id** | **Test cases** | **Priority** | **Preconditions** | **Input test data** | **Steps to be executed** | **Expected results** | **Pass/Pass** | **Comments** |
| 13 | Test if user is able to search successfully. | A |  |  | 1. Click search button | All User Management  must be display | Pass |  |
| 14 | Test with a part of  correct User Name. | B | User must have user account  already | A part of user  name | 1. Enter a part of user  name on search field.  2. Click search button | Just display user  have User Name like  search value | Pass |  |
| 15 | Test with correct user name. | A | User must have user account  already | Correct User Name | 1. Enter correct user  name on search field.  2. Click search button | Just display user  have User Name like  search value | Pass |  |
| 16 | Test with character  not match with any  User Name. | A |  | Character  not match with any User Name | 1. Enter character  not match with any  User Name on search field.  2. Click search button | Proper error must be displayed | Pass |  |
| 17 | Test Add User hyperlink. | A |  |  | 1. Click on Add User hyperlink. | Add User form must be display. | Pass |  |
| 18 | Test Show All hyperlink. | B |  |  | 1. Click one Show All hyperlink. |  | Pass |  |
| 19 | Test if user is able to add user successfully. | A |  | Enter Valid data all field and  Select one Studio | 1. Enter Valid data on all field.  2. Choose one Studio.  3. Click add button | User must be added  and successful notification  must be displayed | Pass | Valid data is : User Name : Not special character , Not empty, In range 6 ~ 32 character Password : In range 6 ~ 32 character, Not empty. Studio : Must be select |
| 20 | Test if empty User Name and valid all other field. | B |  | Empty User Name, valid all other field Select one Studio | 1. Empty User Name.  2. Enter valid Password.  3. Click add button. | Proper error must be displayed and prompt to enter add again | Pass | Valid data is : User Name : Not special character , Not empty, In range 6 ~ 32 character Password : In range 6 ~ 32 character, Not empty. Studio : Must be select |
| 21 | Test if User Name have special character and valid all other field. | B |  | Enter User Name have special character. Valid all other field. Select one Studio | 1. Enter User Name have special character.  2. Enter valid all other field.  3. Select one user role.  4. Click add button. | Proper error must be displayed and prompt to enter add again | Pass | Valid data is : User Name : Not special character , Not empty, In range 6 ~ 32 character Password : In range 6 ~ 32 character, Not empty. Studio : Must be select |
| 22 | Test if User Name less than 6 character and valid all other field. | B |  | Enter User Name less than 6 character. Valid all other field. Select one Studio Select one user role. | 1. Enter User Name less than 6 character.  2. Enter valid all other field.  3. Click add button. | Proper error must be displayed and prompt to enter add again | Pass | Valid data is : User Name : Not special character , Not empty, In range 6 ~ 32 character Password : In range 6 ~ 32 character, Not empty. Studio : Must be select |
| 23 | Test if User Name more than 32 character and valid all other field. | B |  | Enter User Name more than 32 character. Valid all other field. Select one Studio | 1. Enter User Name more than 32 character.  2. Enter valid all other field.  3. Click add button. | Proper error must be displayed and prompt to enter add again | Pass | Valid data is : User Name : Not special character , Not empty, In range 6 ~ 32 character Password : In range 6 ~ 32 character, Not empty. Studio : Must be select |
| 24 | Test if empty Password and valid all other field. | B |  | Empty Password , valid User Name Select one Studio Select one user role | 1. Empty Password.  2. Enter valid User Name.  3. Click add button. | Proper error must be displayed and prompt to enter add again | Pass | Valid data is : User Name : Not special character , Not empty, In range 6 ~ 32 character Password : In range 6 ~ 32 character, Not empty. Studio : Must be select |
| 25 | Test if Password less than 6 character and valid all other field. | B |  | Enter Password less than 6 character. Valid all other field. Select one Studio | 1. Enter Password less than 6 character.   2. Enter valid all other field.  3. Click add button. | Proper error must be displayed and prompt to enter add again | Pass | Valid data is : User Name : Not special character , Not empty, In range 6 ~ 32 character Password : In range 6 ~ 32 character, Not empty. Studio : Must be select |
| 26 | Test if Password more than 32 character and valid all other field. | B |  | Enter Password more than 32 character.  Valid all other field. Select one Studio | 1. Enter Password more than 32 character.  2. Enter valid all other field.  3. Select one user role.  4. Click add button. | Proper error must be displayed and prompt to enter add again | Pass | Valid data is : User Name : Not special character , Not empty, In range 6 ~ 32 character Password : In range 6 ~ 32 character, Not empty. Studio : Must be select |
| 27 | Test if not selected Studio and valid all other field | B |  | Valid all field but not selected Studio | 1. Enter Valid data on all field but not selected Studio.  2. Click add button. | Proper error must be displayed and prompt to enter add again | Pass | Valid data is : User Name : Not special character , Not empty, In range 6 ~ 32 character Password : In range 6 ~ 32 character, Not empty. Studio : Must be select |
| 28 | Test reset button | A |  | Enter value in any field | 1. Enter value in any field.  2. Click reset button. | All field should be clear | Pass |  |
| 29 | Test delete button | A |  |  | 1. Click delete button. | Confirm notification must be displayed. If admin click OK button that user will be delete | Pass |  |
| 30 | Test edit button | A |  | Click edit button | 1. Click Edit button. | Edit user form should be visible, all field should have old data, User Name must can not change | Pass | All field have old data User Name is read only |
| 31 | Test if empty Password field. Studio is selected. | B |  | Empty Password , Select one Studio | 1. Empty Password.  2. Select one Studio.  3. Click edit button. | Proper error must be displayed and prompt to enter edit again | Pass |  |
| 32 | Test if Password less than 6 character. Studio is selected. | B |  | Enter Password less than 6 character, Select one Studio | 1. Enter Password less than 6 character.  2. Select one Studio.  3. Click edit button. | Proper error must be displayed and prompt to enter edit again | Pass | Valid data is : User Name : Not special character , Not empty, In range 6 ~ 32 character Password : In range 6 ~ 32 character, Not empty. Studio : Must be select |
| 33 | Test if Password more than 32 character. Studio is selected. | B |  | Enter Password more than 32 character, Select one Studio | 1. Enter Password more than 32 character.  2. Select one Studio.  3. Click edit button. | Proper error must be displayed and prompt to enter edit again | Pass | Valid data is : User Name : Not special character , Not empty, In range 6 ~ 32 character Password : In range 6 ~ 32 character, Not empty. Studio : Must be select |
| 34 | Test if user is able to edit successfully. | A |  | Valid data, Select Studio. | 1. Enter valid in all field and select Studio.  2. Click edit button. | User must be edited  and successful notification  must be displayed | Pass | Valid data is : User Name : Not special character , Not empty, In range 6 ~ 32 character Password : In range 6 ~ 32 character, Not empty. Studio : Must be select |

## Test Report　テストレポート

### Unit Test Report

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **No** | **Module code** | **Pass** | **Fail** | **Untested** | **N/A** | **Number of test cases** |
|  | Login Page | 1 | 0 | 0 | 0 | 1 |
|  | Forgot Password | 1 | 0 | 0 | 0 | 1 |
|  | User Management | 1 | 0 | 0 | 0 | 1 |
|  | Studio Management | 1 | 0 | 0 | 0 | 1 |
|  | District Management | 1 | 0 | 0 | 0 | 1 |
|  | Studio Profile | 1 | 0 | 0 | 0 | 1 |
|  | Package Management | 1 | 0 | 0 | 0 | 1 |
|  | Picture Management | 1 | 0 | 0 | 0 | 1 |
|  | Contract Management | 1 | 0 | 0 | 0 | 1 |
|  | Studio Page | 1 | 0 | 0 | 0 | 1 |
|  | **Sub total** | 10 | 0 | 0 | 0 | 10 |

Table 5.10.4

### Integration Test Report　統合テストレポート

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **No** | **Module code** | **Pass** | **Fail** | **Untested** | **N/A** | **Number of test cases** |
|  | Login Page | 7 | 0 | 0 | 0 | 7 |
|  | Forgot Password | 7 | 0 | 0 | 0 | 7 |
|  | User Management | 11 | 0 | 0 | 0 | 11 |
|  | Studio Management | 12 | 0 | 0 | 0 | 12 |
|  | District Management | 6 | 0 | 0 | 0 | 6 |
|  | Studio Profile | 3 | 0 | 0 | 0 | 3 |
|  | Package Management | 11 | 0 | 0 | 0 | 11 |
|  | Picture Management | 10 | 0 | 0 | 0 | 10 |
|  | Contract Management | 19 | 0 | 0 | 0 | 19 |
|  | Studio Page | 14 | 0 | 0 | 0 | 14 |
|  | **Sub total** | 96 | 0 | 0 | 0 | 96 |

Table 5.10.5

### System Test Report　システムテストレポート

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **No** | **Module code** | **Pass** | **Fail** | **Untested** | **N/A** | **Number of**  **test cases** |
|  | Login Page | 24 | 0 | 0 | 0 | 12 |
|  | User Management | 22 | 0 | 0 | 0 | 22 |
|  | Studio Management | 46 | 0 | 0 | 0 | 46 |
|  | District Management | 9 | 0 | 0 | 0 | 9 |
|  | Manager Login | 5 | 0 | 0 | 0 | 5 |
|  | Studio Profile | 28 | 0 | 0 | 0 | 28 |
|  | Package Management | 26 | 0 | 0 | 0 | 26 |
|  | Contract Management | 94 | 0 | 0 | 0 | 94 |
|  | Picture Management | 16 | 0 | 0 | 0 | 16 |
|  | Home Page | 15 | 0 | 0 | 0 | 15 |
|  | Studio List Page | 5 | 0 | 0 | 0 | 5 |
|  | Studio Detail Page | 8 | 0 | 0 | 0 | 8 |
|  | Cart Page | 12 | 0 | 0 | 0 | 12 |
|  | Contract Page | 4 | 0 | 0 | 0 | 4 |
|  | **Sub total** | 307 | 0 | 0 | 0 | 307 |

Table 5.10.6

### Test Report　テストレポート

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **P** | Pass | Fail | Untested | N/A |
| **Unit Test** | 10 | 0 | 0 | 0 |
| **Integration Test** | 96 | 0 | 0 | 0 |
| **System Test** | 307 | 0 | 0 | 0 |
| **Total** | **403** | **0** | **0** | **0** |

Table 5.10.7