

**CAPSTONE PROJECT REPORT**

**Report 5 – Software Test Documentation**

– Hanoi, October 2020 –

**Table of Contents**

[I. Project Report 3](#_heading=h.gjdgxs)

[1. Status Report 3](#_heading=h.30j0zll)

[2. Team Involvements 3](#_heading=h.1fob9te)

[3. Issues/Suggestions 3](#_heading=h.3znysh7)

[II. Project Management Plan 4](#_heading=h.2et92p0)

[1. Overall Description 4](#_heading=h.tyjcwt)

[1.1 Test Model 4](#_heading=h.3dy6vkm)

[1.2 Testing Levels 4](#_heading=h.1t3h5sf)

[1.3 Testing Types 4](#_heading=h.4d34og8)

[2. Test Plan 4](#_heading=h.2s8eyo1)

[2.1 Test Stages 4](#_heading=h.17dp8vu)

[2.2 Resources 4](#_heading=h.3rdcrjn)

[2.3 Test Milestones 5](#_heading=h.26in1rg)

[2.4 Deliverables 5](#_heading=h.lnxbz9)

[3. Test Cases 5](#_heading=h.35nkun2)

[4. Test Reports 5](#_heading=h.1ksv4uv)

# I. Project Report

## 1. Status Report

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Work Item** | **Status** | **Notes (Work Item in Details)** |
| 1 |  | Pending |  |
| 2 |  | In Progress |  |
| 3 |  | Completed |  |

## 2. Team Involvements

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Task** | **Member** | **Notes (Task Details, etc.)** |
| 1 |  | KienNT |  |
| 2 |  | TuanTV |  |
| 3 |  | AnhLM |  |

## 3. Issues/Suggestions

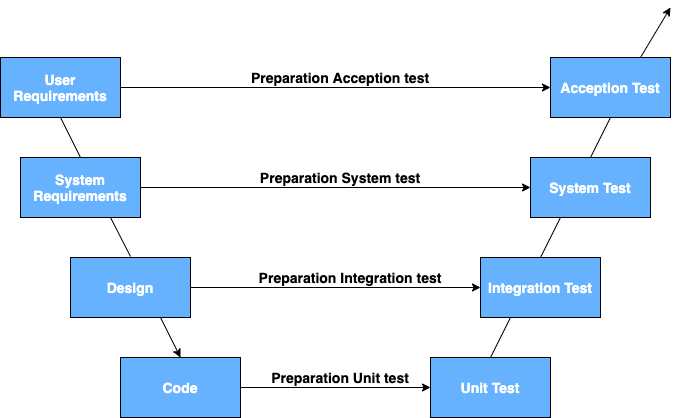
|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Issue** | **Status** | **Notes (Solution, Suggestion, etc.)** |
| 1 |  | Pending |  |
| 2 |  | In Progress |  |
| 3 |  | Completed |  |

# II. Project Management Plan

## 1. Overall Description

### 1.1 Test Model

Overall, to fit the software development process model selected Iterative and Incremental. We choose V-Model to implement the testing process.

**

*Figure 5.1- V-model*

With V-Model, software development is separated into two appropriate phase’s groups. Each stage will be a V-model. In this model, the verification and validation will be done side by side. It emphasizes the strict process flow to develop a quality product. The errors occurring in any phase will be corrected in that phase. Proactive defect tracking defects, which are found at early stages even, may be in the development phase before application being tested.

Specifically, CMA is divided into 2 sub-systems: CMA Back-end and Front- end. In each phase of the process, we use a specific process for each sub-system team to fit the requirement, the characteristic and the human resource of each team.

CMA Back-end has 2 levels of test:

● Unit testing: Automation tests that cover logic of Models and Libraries.

● API testing: Automation tests that involve testing APIs directly (in isolation) to determine whether APIs return the correct response (in the expected format) for a broad range of feasible requests, react properly to edge cases such as failures and unexpected/extreme inputs.

CMA Front-end works mostly with GUI instead of logic and it depends on CMA Back-end, so that CMA Front-end applies system testing which covers the whole CMA system.

### 1.2 Testing Phases

There are 4 testing phases that would be performed in the project: Unit testing, integration testing, system testing and acceptance testing.

|  |  |  |
| --- | --- | --- |
| **ID** | **Phases of testing** | **Description** |
| 1 | Unit Testing | The main aim of this endeavor is to determine whether the application functions as designed. In this phase, a unit can refer to a function, individual program or even a procedure, and a White-box Testing method is usually used to get the job done |
| 2 | Integration Testing | Integration testing allows individuals the opportunity to combine all of the units within a program and test them as a group. This testing level is designed to find interface defects between the modules/functions. This is particularly beneficial because it determines how efficiently the units are running together. |
| 3 | System Testing | System testing is the first level in which the complete application is tested as a whole. The goal at this level is to evaluate whether the system has complied with all of the outlined requirements and to see that it meets Quality Standards. |
| 4 | Acceptance Testing | Acceptance testing (or User Acceptance Testing), is conducted to determine whether the system is ready for release. During this phase, the tester and some alpha test users will test the system to  find out whether the application meets their business’ needs. |

### 1.3 Testing Types

The different types of testing that will be carried out this project are:

* API testing :
  + API testing will test all of the individual implemented API of CMA. Verify it works correctly for the function it was created.
  + Basically, almost all API test cases are executed as automation tests. After that all API with standard sample datasets will be saved and confirmation tests will be executed by using Postman with the developer’s local database.
* Function testing:
  + Test case dựa trên đặc tả của phần mềm. Các chức năng được test bằng cách nhập vào các giá trị nhập và kiểm tra kết quả đầu ra, và ít quan tâm đến cấu trúc bên trong của hệ thống. It is the process of trying to find the differences between the external specification of the software and the fact that the software provides.
  + Functional tests often use a black box test technique: Equivalence Class Partitioning, Pairwise, domain analysis.
* User Interface testing:
  + Is testing the application through the graphical interface (GUI) to check if the application's interface meets the requirements of the design as well as the operations of each component on that interface (Click button, link ...):
    - GUI tests will be performed fully on all screens.
    - This test targets to cover the verification of the overall look and feel of the Family OKR system including initial position, font, text size, color, focus, initial button, tab order, label, screen sizes and sentences width.
    - Check all the GUI elements for size, position, width, length and acceptance of characters or numbers. For instance, it must be able to provide inputs to the input fields.

Check if error messages are displayed correctly.

Check if fonts used in the application are readable.

Check if the alignment of the text is proper.

* Security testing: Security Testing is a type of Software Testing that uncovers vulnerabilities of the system and determines that the data and resources of the system are protected from possible intruders. It ensures that the software system and application are free from any threats or risks that can cause a loss.
* Regression testing:
  + This test is performed to ensure that new code changes will not affect existing functions. It makes sure that the old code still works after a new code change is made.
  + Software maintenance is an activity that includes improving, fixing bugs, optimizing, and removing existing features. These modifications can cause the system to work incorrectly. Hence, Regression testing becomes necessary. Regression testing can be performed using the following techniques: Retest all, regression test selection, Prioritization of test case

## 2. Test Plan

### 2.1 Test Phases

The table below describes the stages in which common tests are executed:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Type of Test** | **Phases of Test** | | | |
| ***Unit*** | ***Integration*** | ***System*** | ***Acceptance*** |
| API test | X |  |  |  |
| Function test | X | X | X | X |
| User Interface test | X | X | X |  |
| Security test | X | X |  |  |
| Regression testing | X | X | X | X |

### 2.2 Resources

#### a. Human Resources

The details on roles and responsibilities of the project members who would involve in testing works:

|  |  |  |
| --- | --- | --- |
| **Worker/Doer** | **Role** | **Specific Responsibilities/Comments** |
| Nguyen Thi Trang | Project Manager/ Developer | * Responsible for project schedule and overall success of the project. * Create UT and UT report * Fix bugs |
| Le Thi Thu Trang | Test Leader | * Performing the actual system testing. * Manage test resources and assign test tasks. * Create test plan * Create test cases ( IT, ST) * Create test report * Execute test * Test log report |
| Do Trung Duc | Tester/Developer | * Create unit test * Fix bugs |
| Nguyen Duc Thien | Tester/Developer | * Create API test * Fix bugs. |
| Do Ngoc Khanh | Developer | * Fix bugs. |

#### b. Environment

The details about the tools (software, hardware, infrastructure) which the project would use for testing:

|  |  |  |  |
| --- | --- | --- | --- |
| **Purpose** | **Tool** | **Version** | **Provider** |
| Use to view the web server | Google chrome | v86.0 | - Support with minimal resolution: 1366 x 768  - Personal computer for developing:   * Dell Latitude E6430s * Intel(R) Core(™) i5-3320M CPU 2.60GHz * Ram 6614MB   - Server   * Intel(R) Celeron(R) CPU G550 @ 2.60GHz * RAM 4GB * HDD 500GB |
| Use to management test case | Microsoft Excel | Microsoft Office 2019 |
| Use to management test plan | Microsoft Word | Microsoft Office 2019 |
| Use to bug logging | Github | v2.27.0 |
| Use to management the list of all API and manually test API | Postman | v7.28.0 |

### 2.3 Test Milestones

The table below describes the test milestones for the CMA Project:

|  |  |  |
| --- | --- | --- |
| **Milestone Task** | **Start Date** | **End Date** |
| **Test preparation** | **19/10/2020** | **22/11/2020** |
| Test plan completion | 19/10/2020 | 19/10/2020 |
| Acceptance test cases completion | 20/10/2020 | 22/10/2020 |
| System test cases completion | 23/10/2020 | 30/10/2020 |
| Integration test cases completion | 31/10.2020 | 14/11/2020 |
| Unit test cases completion | 15/11/2020 | 22/11/2020 |
| **Test execution** | **23/11/2020** | **20/12/2020** |
| Unit test completion | 23/11/2020 | 29/11/2020 |
| Integration test completion | 25/11/2020 | 10/12/2020 |
| System test completion | 11/12/2020 | 18/12/2020 |
| Acceptance test completion | 19/12/2020 | 20/12/2020 |

### 2.4 Deliverables

The table below is the Deliverables for CMA Project:

|  |  |  |
| --- | --- | --- |
| **No** | **Deliverables** | **Due Date** |
| 1 | Test plan | 10/11/2020 |
| 2 | Test case | 22/11/2020 |
| 3 | Test case review | 24/11/2020 |
| 4 | Defect report | 21/12/2020 |
| 5 | Test report | 23/12/2020 |

## 

## 3. Test Cases

### 3.1 Unit Testing and API Testing

Unit testing and API testing will be done by the developers and approved by the team leader.

The Clinic Management Application development team embrace this feature to gain the following advantages:

* Reduce the level of bugs in production code.
* Save development time.
* Automation tests can be run as frequently as required.
* Make it easier to change and refactor code by improving the design of code especially with
* Test-Driven Development.
* Can easily form a document from the tests.
* Easier to maintain than GUI tests which are difficult to maintain with the short release cycles

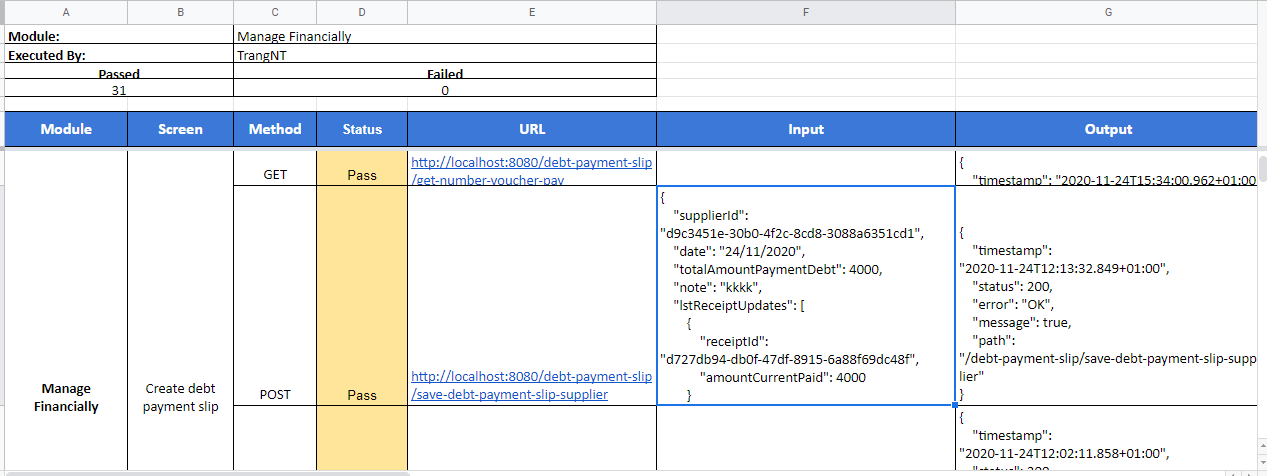
and frequent changes and with a complex system

* Reduce cost of resource to corresponding GUI testing

#### 3.1.1 API Testing

Unit testing was performed by using Postman to test

Detailed Test cases will be described in **Clinic Management Application\_Testcase\_API Test.xlsx** file.

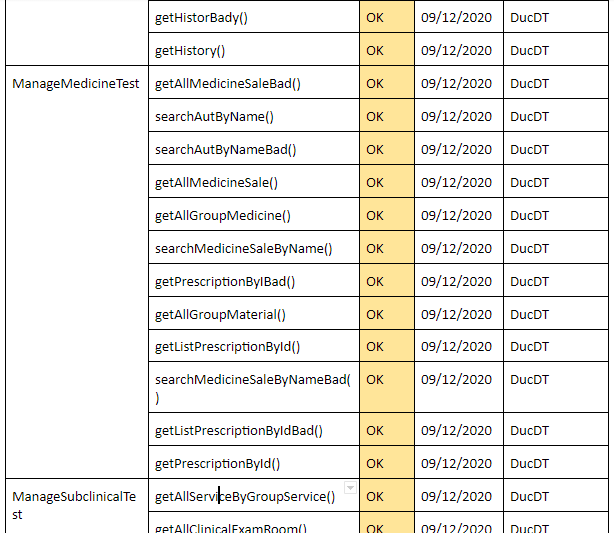


*Figure 5.2- API Testing sample*

#### 3.1.2 Unit Testing

Unit testing was performed by using JUnit to test

Detailed Test cases will be described in **Clinic Management Application\_Testcase\_Unit Test.xlsx** file.



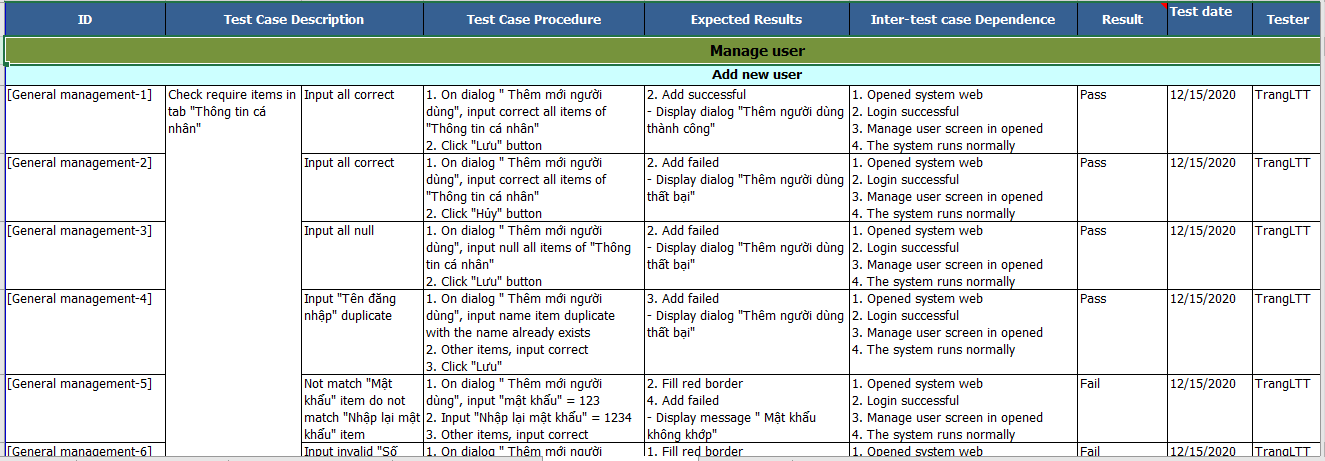
*Figure 5.3- Unit Testing sample*

### 3.2 Integration Testing

Detailed Test cases will be described in **Clinic Management Application\_Testcase\_System Test.xlsx** file.

Integration and system tests are done by testers to ensure that combined units work correctly and that the system as a whole functions as intended. Each test case is tested using Google Chrome v84.0

GUI testing is also done during this process to ensure that elements load and function correctly, text is readable.



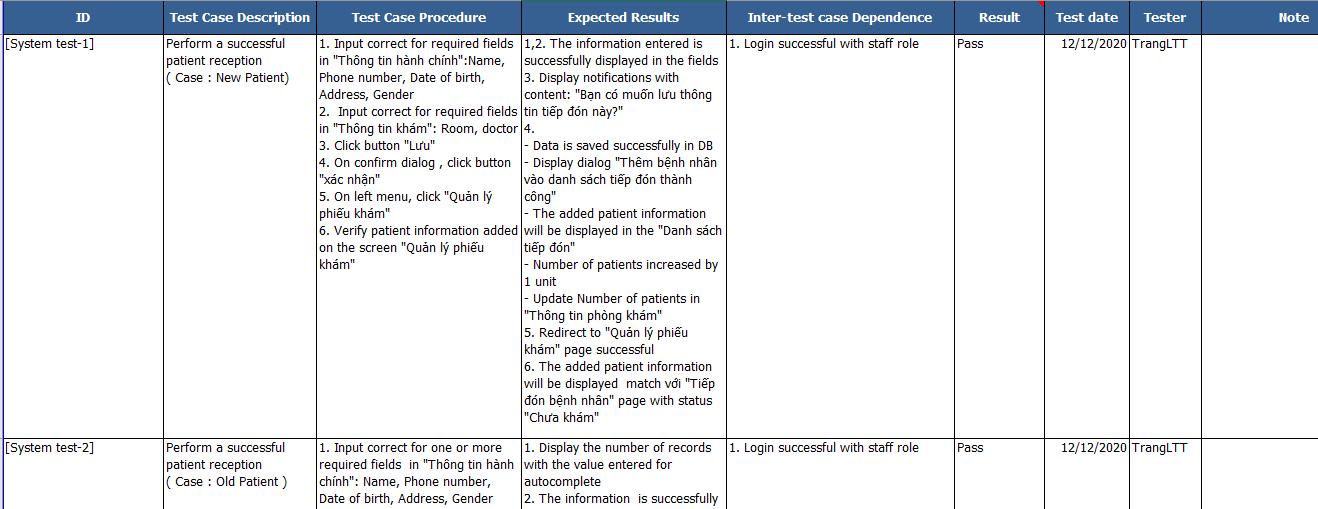
*Figure 5.4- Integration Test sample*

### 3.3 System Testing

Detailed Test cases will be described in **Clinic Management Application\_Testcase\_Integration Test.xlsx** file.

As a standard definition, Clinic Management Application development team define that case is:

* Test cases for functional testing are derived from the target of test's use cases Test cases should be developed for each use case scenario. The use case scenarios are identified by describing the paths through the use case that traverse the basic flow and alternate flows start to finish through the use case.
* By using good automation test and using, FamilyOKR Project system testing will not focus on common logic of system like length of text but focus on behavior of website and aims to validate that all software module dependencies are functionally correct and that data integrity is maintained between separate modules for the entire solution



*Figure 5.4- System Test sample*

### 3.4 Acceptance Testing

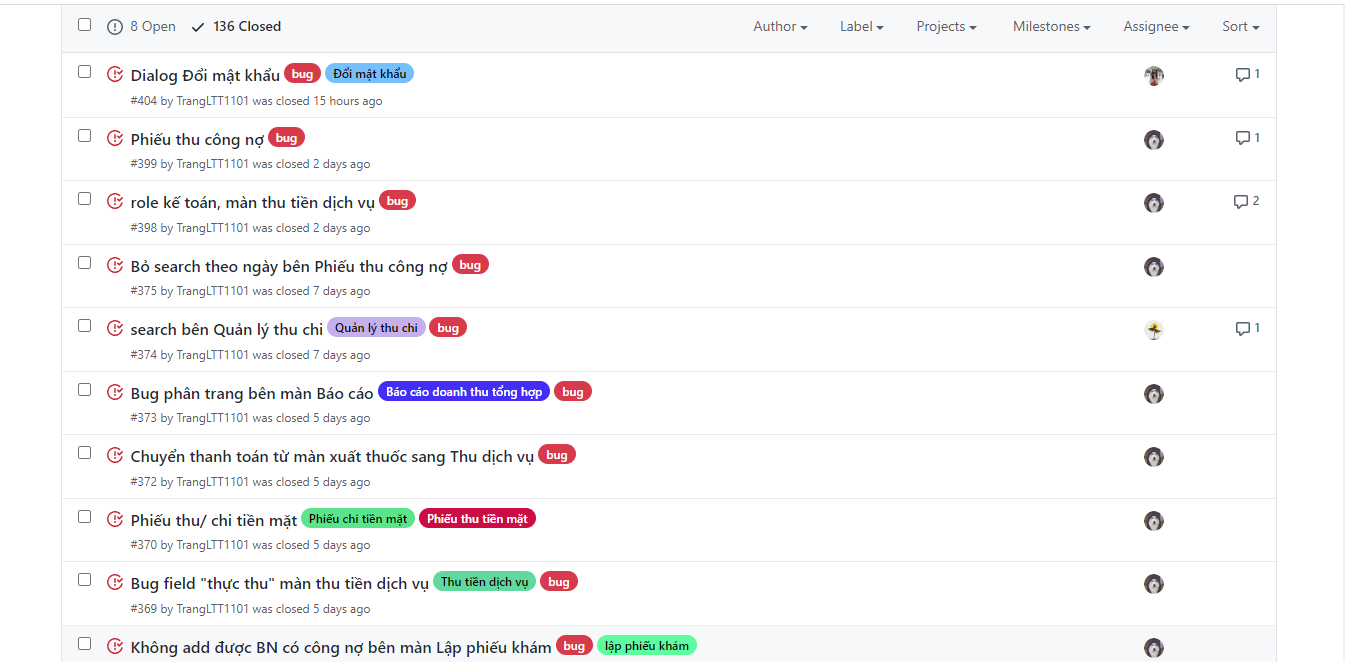
Acceptance Testing is a level of the software testing process where a system is tested for acceptability. The purpose of this test is to evaluate the system’s compliance with the business requirements and assess whether it is acceptable for delivery. But our project will use the Checklists as a substitute for Acceptance testing.

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Check lists** | **Yes** | **No** |
| **General** | | | |
| CL-001 | Text on all screens and pages for spelling and grammatical errors | √ |  |
| CL-002 | Functionality of buttons available on all screens and pages | √ |  |
| CL-003 | Validation error messages are displayed properly below the field | √ |  |
| CL-004 | All error messages are displayed in border red color | √ |  |
| CL-005 | All inactive items are gray out | √ |  |
| CL-006 | Delete, cancel functionality for any record on screen are asked for confirmation | √ |  |
| CL-007 | All numeric values are formatted properly | √ |  |
| **GUI and Usability** | | | |
| CL-008 | Screens are designed follow project standards | √ |  |
| CL-009 | The screen well organized and easy to use | √ |  |
| CL-010 | All fields on screen and page (e.g. text box, radio options, dropdown lists) should be aligned properly | √ |  |
| CL-011 | Information is arranged symmetrically with adequate spacing between components | √ |  |
| CL-012 | The most important fields are located where they are easy to see | √ |  |
| CL-013 | Information is presented in the order that the user needs it | √ |  |
| CL-014 | The screen designed to fit the requirements for international use | √ |  |
| CL-015 | The text is easy to translate. Don't use slang, acronyms, and abbreviations | √ |  |
| CL-016 | Icons and images are designed impression and copyright | √ |  |
| CL-017 | Font size, style and color for headline, description test, labels, infield data, and grid information standard as specified in SRS | √ |  |
| CL-018 | The static text is clear, concise, and meaningful | √ |  |
| CL-019 | Buttons follow the project standards for size and position (e.g., Negative button is correct size and to the right of the Positive button) | √ |  |
| CL-020 | Graphical objects are used appropriately and according to the guidelines specified in Graphical User Interface Design | √ |  |
| CL-021 | A list view is used to allow a collection of items that are on a single hierarchical level | √ |  |
| CL-022 | A tree view is used to allow a collection of items to be displayed and manipulated within varying hierarchical levels | √ |  |
| CL-023 | Pop-up menus are provided for the user to access information about an object's properties or perform specific tasks on the object | √ |  |
| CL-024 | Command button are used to trigger application processes | √ |  |
| CL-025 | Switch buttons are used to change status on/off. | √ |  |
| CL-026 | System display notification message when meet trouble, error | √ |  |
| **Database** | | | |
| CL-027 | Correct data is getting saved in database upon successful backend submit | √ |  |
| CL-028 | Values columns are not accepting null values | √ |  |
| CL-029 | Data should be stored in single multiple tables based on design | √ |  |
| CL-030 | Input data is not truncated. Field length shown to user on application, web and in database schema should be same | √ |  |
| CL-031 | Input numeric fields with minimum, maximum, and float values | √ |  |
| CL-032 | Input numeric fields with negative values (for both acceptance and non-acceptance) | √ |  |
| CL-033 | Drop down list are saved correctly in database | √ |  |
| CL-034 | Database fields are designed with correct data type and data length | √ |  |
| CL-035 | All collections constraints like Primary key, etc. are implemented correctly | √ |  |
| CL-036 | Input field leading and trailing spaces are truncated before committing data to database | √ |  |
| **Performance** | | | |
| CL-037 | Real time connection | √ |  |
| CL-038 | Web page load time is within acceptable range | √ |  |
| CL-039 | Web page load on slow connections | √ |  |
| CL-040 | Response time for any action under light, normal, moderate and heavy load conditions | √ |  |
| CL-041 | Database query execution time | √ |  |
| CL-042 | Stress testing of application |  | √ |
| **Security** | | | |
| CL-043 | Test cookie allowed (disabled or allowed to be edited) | √ |  |
| CL-044 | Test the form giving away security information if the source is viewed | √ |  |
| CL-045 | Test password security and password policy enforcement | √ |  |
| CL-046 | Check application logout functionality | √ |  |
| CL-047 | Cookie information should be stored in encrypted format only | √ |  |

*Table 5-1: Check list*

### 3.5 Defect log

Clinic management application project used Github to manage bugs



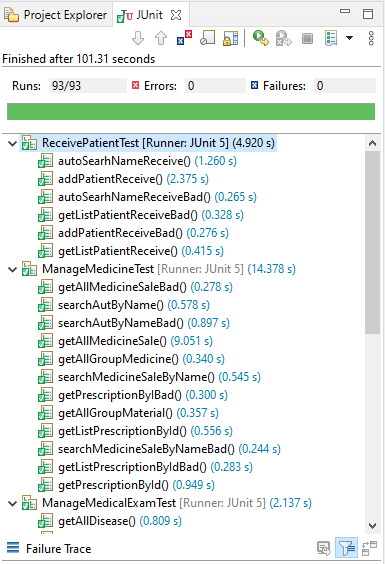
*Figure 5.5- Defect log sample*

Every member of the CMA project created an account in Github to take part in activities: control bugs, fix bugs, re-test bugs and close bugs. Bugs will be log by tester or developer in develop progress:

* Tester detect defect: tester perform testing system using integration test and system testing to find the bug
* Tester log bug on gitlab: tester create a issue in Github, at the same time the tester assigns the developer the message that needs to be fixed
* Developer fixed bug: the developer is assigned fix the bug, the stature on gitlab change to doing and when fixed developer change status to close and message to tester
* Tester test again: if bug have been fixed tester close the bug else if bug not fixed tester reopen bug

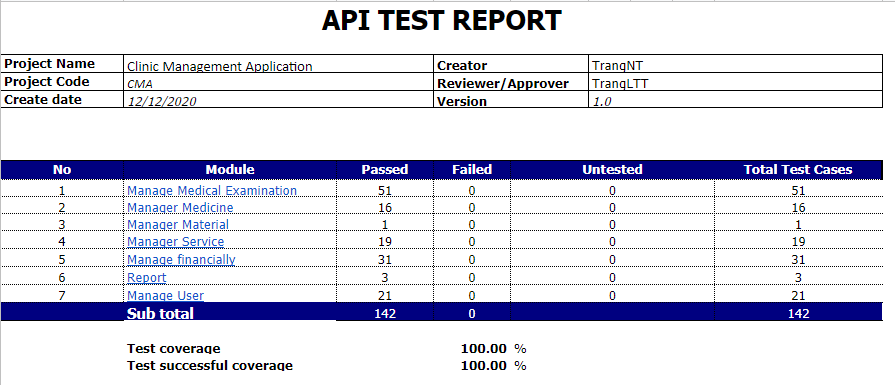
## 4. Test Reports

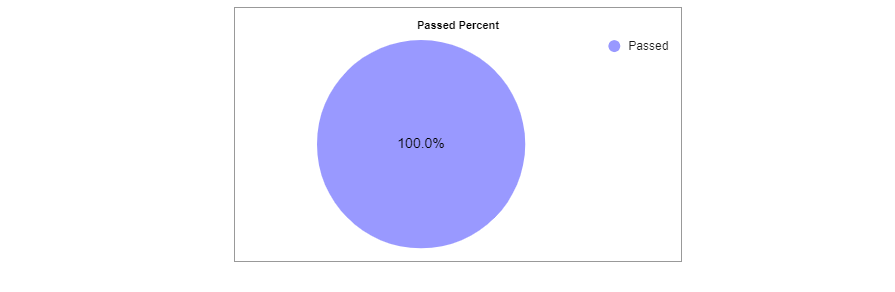
### 4.1 Unit Testing



*Figure 5.6- Unit test report*

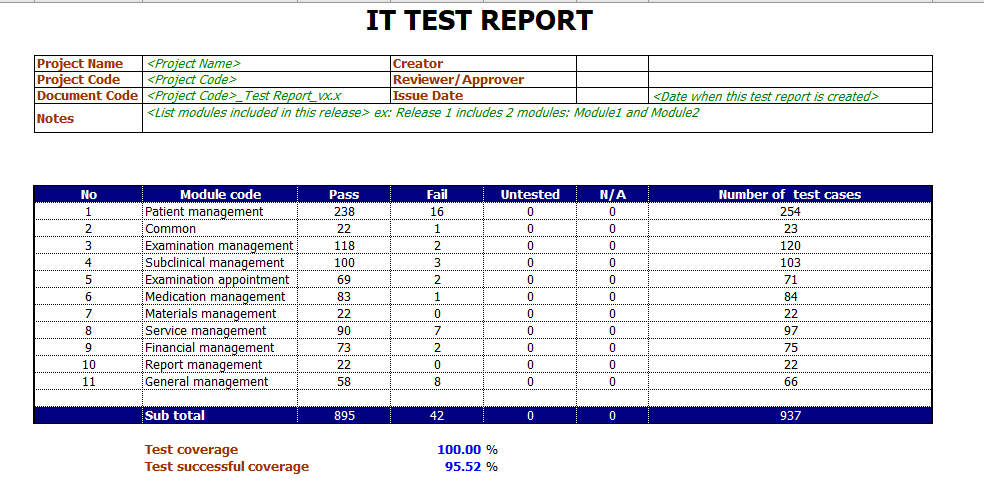
### 4.2 API Testing





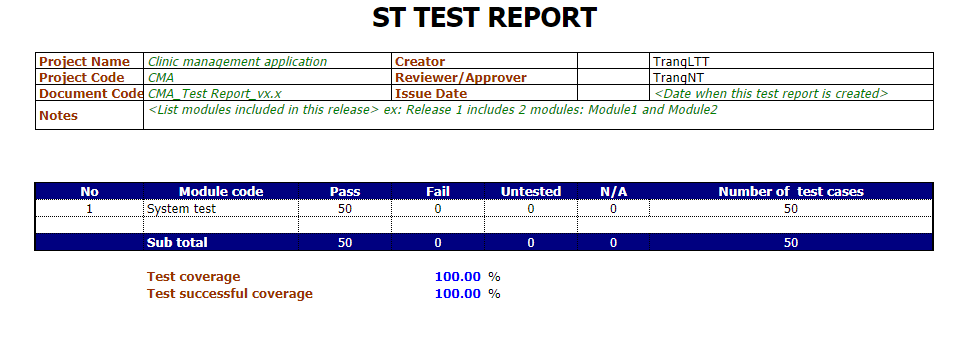
*Figure 5.7- API testing report*

### 4.3 Integration Testing



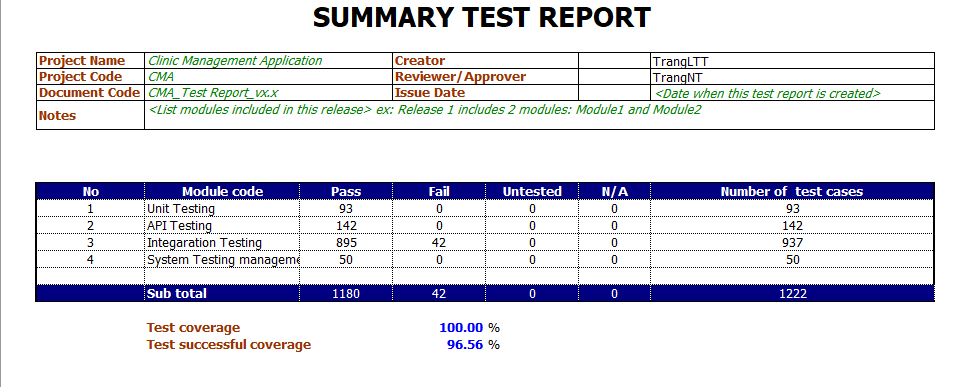
*Figure 5.8- Integration testing report*

### 4.4 System Testing



*Figure 5.9- System testing report*

### 4.5 Summary Testing Report



*Figure 5.10- Summary testing report*