|  |  |
| --- | --- |
| **Logo_FPT_University_doc** | **MINISTRY OF EDUCATION AND TRAINING** |

**FPT UNIVERSITY**

**CAPSTONE PROJECT DOCUMENT**

**Fstay**

**Report #5 – Software Test Documents**

|  |  |
| --- | --- |
| **SWP490\_G7** | |
| **Group Member** | Phạm Minh Dương – SE05785 |
| Lý Văn Thiên – SE05754 |
| Trần Đức Anh – SE06025 |
| Võ Đại Dương Phi – SE05784 |
| Phạm Mạnh Đạt – SE05386 |
| **Supervisor** | Mr. Nguyễn Văn Sang |
| **Project Code** | Fstay |

- Hanoi, 08/2020 –

**Table of Contents**

[1](#_heading=h.gjdgxs) INTRODUCTION 3

[1.1](#_heading=h.30j0zll) Purposes 3

[1.2](#_heading=h.1fob9te) Test Objectives 3

[2](#_heading=h.3znysh7) TEST PLAN 3

[2.1](#_heading=h.2et92p0) Scope of Testing 3

[2.2](#_heading=h.tyjcwt) Requirements for Testing 3

[2.2.1](#_heading=h.3dy6vkm) Test Items 3

[2.2.2](#_heading=h.1t3h5sf) Acceptance Test Criteria 5

[2.2.3](#_heading=h.4d34og8) Constraints 5

[2.2.4](#_heading=h.2s8eyo1) Completion Criteria 5

[2.2.5](#_heading=h.17dp8vu) Testing Risks 6

[2.3](#_heading=h.3rdcrjn) Test Strategies 6

[2.3.1](#_heading=h.26in1rg) Test Policies 6

[2.3.2](#_heading=h.lnxbz9) Test Model 6

[2.3.3](#_heading=h.44sinio) Type of Testing 9

[2.3.4](#_heading=h.2jxsxqh) Test Stages 10

[2.3.5](#_heading=h.z337ya) Tools/Environments 11

[2.3.6](#_heading=h.3j2qqm3) Resources 11

[2.3.7](#_heading=h.1y810tw) Test Schedules 11

[2.3.8](#_heading=h.2xcytpi) Test Deliverables 12

[3](#_heading=h.3whwml4) TEST CASES 12

[3.1](#_heading=h.2bn6wsx) Unit Testing and API Testing 12

[3.1.1](#_heading=h.qsh70q) Unit Testing 12

[3.1.2](#_heading=h.49x2ik5) API Testing 13

[3.2](#_heading=h.3o7alnk) System Testing 14

[4](#_heading=h.ihv636) CHECK LIST 14

[5](#_heading=h.1hmsyys) TEST LOGS 16

[5.1](#_heading=h.41mghml) Defect Logs 16

[5.2](#_heading=h.vx1227) Test Reports 16

# INTRODUCTION

## Purposes

The purpose of this document is to plan all the activities needed to perform testing for the system including test strategy and test control. This document also defines scope, requirements, and constraints, risks of testing stages.

## Test Objectives

The test objectives are to verify the functionality of Fstay System. It should focus on testing the main scenario that defined in SRS to guarantee all of them could work normally.

# TEST PLAN

## Scope of Testing

The scope of testing in the project include:

* **Stages of testing**: The testing of this project consists Unit Test, System Test/Integration Test and Acceptance Test
* Unit Test: performed by developers and approved by technical leader. Developers will manage sample, data, output, defect information of unit test and must show if necessary.
* System Test/ Integration Test: performed by test team with support of develop team and approved by test leader. Test team will manage all information of System/Integration Test.
* **Types of testing**: There are two types of testing: functional testing and user interface testing.
* **Range of testing**: Team performs testing all functions defined in the SRS based on the approved version. Test cases of each function are defined in the Test cases document that will be tested to determine if they meet the requirements and intermediate result will be checked according to the SRS.

## Requirements for Testing

### Test Items

This application has these functions in below table that need to test:

|  |  |  |
| --- | --- | --- |
| **Group of functions** | **Use case No.** | **Function** |
| **Common module for customer (renter, owner)** | | |
| Account Management | UC01-01 | Register new account |
| UC01-02 | Login |
| UC01-03 | Logout |
| UC01-04 | Forget password |
| UC01-05 | Change password |
| UC01-06 | View personal information |
| UC01-07 | Update personal information |
| UC01-08 | View list transaction’s history |
| UC01-09 | View notifications |
| UC01-10 | Chat |
| **Renter module** | | |
| Renter – Homestay Management | UC02-01 | View homepage |
| UC02-02 | Search homestay |
| UC02-03 | View list homestay |
| UC02-04 | View homestay details |
| UC02-05 | Review homestay |
| UC02-06 | Report homestay0 |
| UC02-07 | Add favourite homestay |
| UC02-08 | View list favourite homestay |
| UC02-09 | Delete favourite homestay |
| Renter – Order Management | UC03-01 | Book homestay |
| UC03-02 | View list order’s history |
| UC03-03 | View order details |
| UC03-04 | Cancel executing order |
| UC03-05 | Pay in |
| UC03-06 | View list pay in history |
| **Owner module** | | |
| Owner – Account Management | UC04-01 | Verify owner’ identification |
| UC04-02 | View dashboard |
| UC04-03 | Withdraw cash |
| UC04-04 | View list withdraw cash’s history |
| Owner – Homestay Management | UC05-01 | Add new homestay |
| UC05-02 | View list owner’s homestay |
| UC05-03 | View owner’s homestay details |
| UC05-04 | Update homestay’s information |
| UC05-05 | Update homestay’ status |
| UC05-06 | Add discount |
| Owner – Order Management | UC06-01 | View list order |
| UC06-02 | View order details |
| UC06-03 | Cancel executing order |
| **Admin module** | | |
| Admin – Common Management | UC07-01 | Login |
| UC07-02 | Logout |
| UC07-03 | Forget password |
| UC07-04 | View personal information |
| UC07-05 | Update personal information |
| UC07-06 | Change password |
| UC07-07 | View transaction |
| UC07-08 | View dashboard |
| UC07-09 | Change customer’s wallet |
| Admin – Renter Management | UC08-01 | View list renter |
| UC08-02 | View renter’s information details |
| UC08-03 | Change renter’s status |
| Admin – Owner Management | UC09-01 | View list owner |
| UC09-02 | View owner’s information details |
| UC09-03 | Change owner’s status |
| Admin – Homestay Management | UC10-01 | View list homestay |
| UC10-02 | View homestay details |
| UC10-03 | Approve homestay |
| UC10-04 | Change homestay’s status |
| Admin – Promotion Management | UC11-01 | View list promotion |
| UC11-02 | Add new promotion |
| UC11-03 | Change promotion’s status |
| Admin – Utility  Management | UC12-01 | View list utility |
| UC12-02 | Add new utility |
| UC12-03 | Delete utility |
| Admin – Commission Fee Management | UC13-01 | View list commission fee |
| UC13-02 | Add new commission fee |
| UC13-03 | Update commission fee |
| UC13-04 | Delete commission fee |
| Admin – Staff Management | UC14-01 | View list staff |
| UC14-02 | View staff’s information details |
| UC14-03 | Add new staff |
| UC14-04 | Change staff’s positon |
| UC14-05 | Change staff’s status |

Table 2-1: Group of functions

### Acceptance Test Criteria

In this project, we do not have enough time to send application to real customers, so acceptance testing is not performed.

### Constraints

* Duration of testing is from 15-Jul to 25-Aug.
* Testing team has three members.
* The program has many functions so the number of line code is also, lead to have more number of test cases.

### Completion Criteria

* All test cases have been executed and approved.
* All serious defects have been corrected.
* 95% of identified defects have been corrected and closed.

### Testing Risks

There are some risks that the project team could face when performing to test this application:

* Test team members do not have many experiences in testing and do not know particularly about testing process so maybe testers cannot keep deadline or even miss bugs.
* Team do not have enough devices for testers.
* The time of testing is short, so maybe test team cannot re-test when developers finish fixing bugs.

## Test Strategies

### Test Policies

* “Test without good faith”
* “Find bugs as soon as possible and always make sure they have been fixed”
* “No outstanding high severity faults”
* “Ensuring the product covers key features and attributes requirements”
* “Find bugs as soon as possible, as much as possible”
* “No outstanding high severity faults prior to products release”

### Test Model

#### Quality assurance model

As all the characteristics and specifications of this project, our team decide to choose V-model as our quality assurance model.

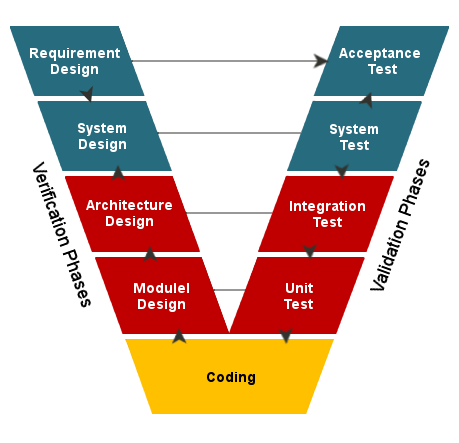


Figure 2-1: V-Model to implement testing process

V-Model evolved from waterfall Model. Each phase must be completed before the next phase begins. Instead of moving down in a linear way, the process steps are bent upwards after the coding phase, to form the typical V shape. Testing is emphasized in this model more than in the waterfall model. It is a structured approach to testing. It brings high quality into the development of our products. The V-model is also called as Verification and Validation model.

QA activities can also be classified by the binary grouping of verification vs. validation activities. Validation checks the conformance to quality expectations of customers and users in the form of whether the expected functions or features are present or not. On the other hand, verification checks the conformance of software product implementation against its specification to see the if it is implemented correctly. Therefore, validation deals directly with users and their requirements; while verification deals with internal product specifications. In the software development process perspective, different processes may involve customers and users in different ways. Therefore, verification and validation activities may be distributed in these different processes differently.

In the waterfall process, direct involvement of users and user requirement is at the very beginning and the very end of the development process. These phases include project planning, market analysis, requirement analysis, specification, acceptance testing, product release, and post-release product support and maintenance. Therefore, these are the phases where validation activities may be the focus.

These verification and validation activities can be best illustrated by the V-model in Figure 5-1, a variation of the waterfall process model where the different development phases are presented in a V-shaped graph, relating specific verification or validation activities to their corresponding requirements or specifications. For example, customer requirements are validated by operational use; while product specification, high-level design, and low- level design are verified by system test, integration test, and component test, respectively. In addition, system test also validates the product by focusing on how the overall operations under an environment that resembles that for target customers. In a sense, the users' operational environment is captured as part of the product specification or as part of the testing model. At the bottom, coding and unit testing are typically grouped in a single phase, where the code itself specifies the expected behavior and needs to be verified through unit test. Sometimes, various other QA activities, such as inspections, reviews, walkthroughs, analyses, formal verification, etc., are also associated with the left arm of the V-model and illustrated by additional dotted lines pointed to the specific phases.

Similar to the mapping of QA activities to other process models above, validation and verification activities can be mapped into non-sequential processes such as incremental, iterative, spiral, and extreme programming processes. Typically, there is some level of user involvement in each part or iteration. Therefore, validation plays a more important role in these processes than in the waterfall process or the V-model. (Jeff Tian, 2005, ‘Verification and Validation Perspectives’ in Software Quality Engineering, John Wiley and Sons Inc., New Jersey, pp. 48-49)

In this project, we do not have enough time to implement application to the real customers, so we do not run Acceptance Test. We only perform Unit Test, Integration Test and System Test. There are the technique used for these test above:

* Unit Test: This test is performed by developers. The developers use white box testing technique to do it. It will test the internal structure of application. The developers chooses inputs to exercise paths through the code and determine the appropriate outputs.
* Integration Test: Base on requirement in SRS and validation in database, test team will test as black box testing method to check if the input and output display as expected and does that flow work fluently.
* System Test: Based on requirements in SRS, test team will create scenarios for system testing (ST) to ensure project’s business. In ST, tester will create test cases based on these scenarios and run to verify functional and non-functional requirements.

**Completion criteria:**

* All planned test cases have been executed compared with expected result in test cases.
* System test cases must be green (passed) at least 97%
* All defects must be logged into test documents and re-tested after it is fixed.
* All defects, which are accepted by customer or project technical leader and project manager, will be marked as accepted and not be retested.
* All defects must be resolved or accepted.
* These criteria will be applied for all test types.

**Special considerations:**

* Test databases will be required.
* Testing may be stopped when :
* Time runs out.
* A certain number of defects found.
* Test coverage > 97%.
* Stop when testing becomes unproductive.

#### Testing process

As our team uses V- model as quality assurance model, the figure below is the testing process we will use to perform testing. It has three main phases: Planning & Preparation, Execution and Analysis & Follow-up:

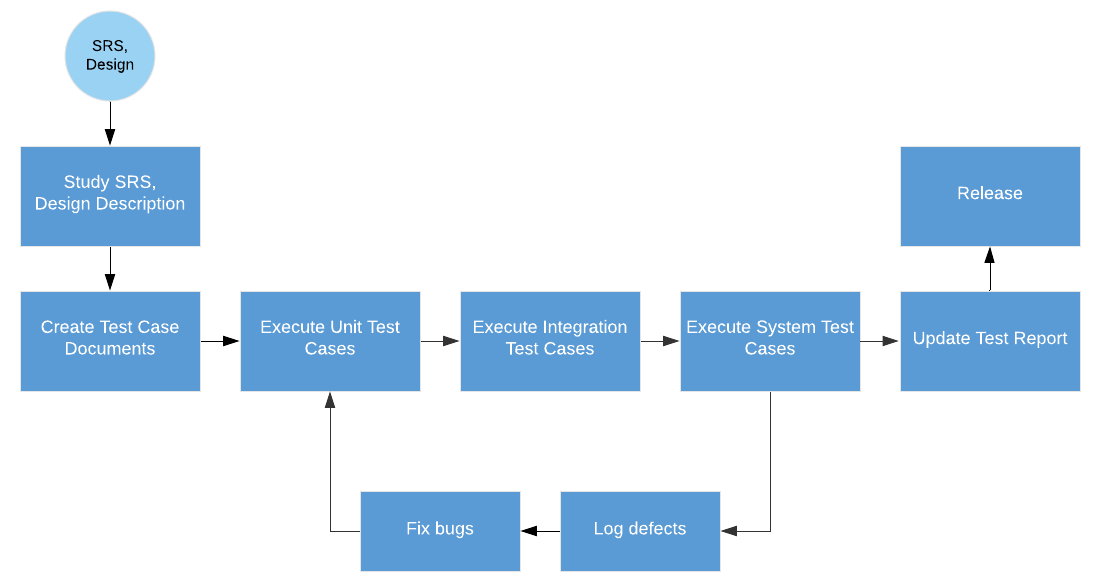


Figure 2-2: Testing process

* Test planning and preparation, which set the goals for testing, select an overall testing strategy, and prepare specific test cases and the general test procedure.
* Test execution and related activities, which also include related observation and measurement of product behavior.
* Analysis and follow-up, which include result checking and analysis to determine if a failure has been observed, and if so, follow-up activities are initiated and monitored to ensure removal of the underlying causes, or faults, that led to the observed failures in the first place.

The major test activities are centered on test execution, or performing the actual tests. At a minimum, testing involves executing the software and communicating the related observations. In fact, many forms of informal testing include just this middle group of activities related to test execution, with some informal ways to communicate the results and fix the defects, but without much planning and preparation. The execution of a specific test case, or a sub-division of the overall test execution sequence for some systems that require continuous operation, is often referred to as a “test run”. One of the key component to effective test execution is the handling of problems to ensure that failed runs will not block the executions of other test cases. This is particularly important for systems that require continuous operation. To many people, defect fixing is not considered to be a part of testing, but rather a part of the development activities. However, re-verification of problem fixes is considered as a part of testing. In this book, we consider all of these activities as a part of testing.

Data captured during execution and other related measurements can be used to locate and fix the underlying faults that led to the observed failures. After we have determined if a test run is a success or failure, appropriate actions can be initiated for failed runs to locate and fix the underlying faults. In addition, further analyses can be performed to provide valuable feedback to the testing process and to the overall development process in general. These analysis results provide us with assessments of the current status with respect to progress, effort, defect, and product quality, so that decisions, such as when to stop testing, can be made based on facts instead of on people’s gut feelings. In addition, some analyses can also help us identify opportunities for long-term product quality improvement. Therefore, various other activities, such as measurement, analysis, and follow-up activities, also need to be supported (Jeff Tian, 2005, ‘Major activities and the generic testing process’ in Software Quality Engineering, John Wiley and Sons Inc., New Jersey, pp. 68-69).

### Type of Testing

#### Functional Testing

Function testing of the target-of-test should focus on any requirements for test that can be traced directly to use cases or business functions and business rules.  The goals of these tests are to verify proper data acceptance, processing, and retrieval, and the appropriate implementation of the business rules.  Functional Testing focuses on testing software against design document, use cases and requirements document. It is black box type of testing and does not require internal working of the software. Typically, functional testing involves evaluating and comparing each software function with the business requirements.

|  |  |
| --- | --- |
| Testing Objective | Ensuring all functions of software meet the requirements as mentioned in the Software Requirement Specification and Software Design Specification, including user interaction, all function defined in specification document implemented correctly |
| Testing Progress | * Testers execute all test cases, use cases, work flows to verify: * When using right use case, right flow, does the application return expected result? * When using wrong flow or inputting invalid data, does the application display warning message or error message? * If any defect has been found, testers will notice the developer who takes care of the appropriated function. The developer will fix it and return to testers to re-test. * If the defect has not been corrected, testers will return it back to developer to fix. If the defect has been corrected, testers will close it. |
| Technique | The test team will use white-box testing, black-box testing, test scenarios in order to 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. |
| Completion Criteria | All test cases have been executed and approved  All serious defects have been corrected  95% identified defects have been corrected and closed |
| Special Considerations | Testing may be stopped when:   * Time runs out. * A certain number of defects found. * Test coverage > 97%. * Testing becomes unproductive. |

Table 2-2: Functional Testing

#### User Interface Testing

User Interface Testing is defined as to test the interface between the application and the end user, testing application elements like fonts, layouts, buttons, images, colors etc. During this testing, certain application is being tested to know how effective it is when it comes to user interaction.

|  |  |
| --- | --- |
| Testing Objective | Ensuring all components in each screen have the best interaction with user and all screens have the same design with the screen described in Software Design Document. |
| Testing Progress | * Tester will review all screens of application by using check lists. * If any defect has been found, testers will notice the developer who takes care of the appropriated screen. The developer will fix it and return to testers to re-test. * If the defect has not been corrected, testers will return it back to developer to fix. If the defect has been corrected, testers will close it. |
| Technique | Checklists, black-box testing |
| Completion Criteria | * All screens have been reviewed through check lists * All identified defects have been corrected and closed |

Table 2-3: User Interface Testing

### Test Stages

The table below identifies the stages in which common tests are executed.

|  |  |  |  |
| --- | --- | --- | --- |
| Type of test | Stage of test | | |
| Unit | Integration | System |
| Functional Test | x | x | x |
| User Interface Test | x |  | x |

Table 2-4: Test stages

### Tools/Environments

#### Hardware

|  |  |  |
| --- | --- | --- |
| Name | Purpose | Detail |
| Laptop DELL | Device for executing test | Ram: 16GB  Hard disk: 1000GB  SSD: 128 Gb  Processor: Core i5 2.50 Hz |
| Laptop ASUS K501L | Device for executing test | Ram: 8GB  Hard disk: 500GB  Processor: Core i5 2.50 Hz |

Table 2-5: Hardware for testing

#### Software

|  |  |  |
| --- | --- | --- |
| Purpose | Name | Version |
| Keep tracking defects | Microsoft Excel | Office 2016 |
| Test plan | Microsoft Excel, Microsoft Word | Office 2016 |
| Test case, Test report | Microsoft Excel | Office 2016 |
| Check lists | Microsoft Excel | Office 2016 |

Table 2-6: Software for testing

### Resources

This table shows the staffing assumptions for the project.

|  |  |  |
| --- | --- | --- |
| Member | Position | Responsibilities |
| Trần Đức Anh | Test Leader | * Assign test tasks * Manage test resources * Create test plan * Review test cases * Execute test * Review test report |
| Võ Đại Dương Phi | Tester | * Create test cases * Execute test |
| Phạm Mạnh Đạt | Tester | * Create test cases * Execute test |

Table 2-7: Resources for testing

### Test Schedules

The table below describes the test schedules for the Fstay Project:

|  |  |  |
| --- | --- | --- |
| Name | Start date | End date |
| Unit Testing and API Testing | July 27, 2020 | July 31, 2020 |
| System Testing | July 20, 2020 | August 4, 2020 |
| Integration Testing | July 27, 2020 | August 1, 2020 |
| User Interface Testing | July 31, 2020 | August 3, 2020 |

Table 2-8: Test schedules

### Test Deliverables

The table below describes the test deliverables for the Fstay Project:

|  |  |  |
| --- | --- | --- |
| Deliverables | Responsibilities | Completed date |
| Test plan | Tester | July 7, 2020 |
| Test cases | Tester | August 1, 2020 |
| Test case review | Tester + PM | August 6, 2020 |
| Defect report | All member | August 25, 2020 |
| Final test summary report | PM | August 25, 2020 |

Table 2-9: Test deliverables

# TEST CASES

## Unit Testing and API Testing

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

The Fstay development team embraces this feature to gain the following advantages:

* Detect bugs as soon as possible
* Make it easier to change and refactor code by improving the design of code.
* Save development time.
* Reduce the level of bugs in production code.
* Automation tests can be run as frequently as required.
* Can easily form a document from the tests.
* Easier to maintain than [GUI tests](https://en.wikipedia.org/wiki/Graphical_user_interface_testing) which are difficult to maintain with the short release cycles and frequent changes and with a complex system.
* Reduce the cost of resource to corresponding GUI testing.

### Unit Testing

Unit testing was performed by using Spring Boot Famework, Mockito 2 and Junit 5 to test.

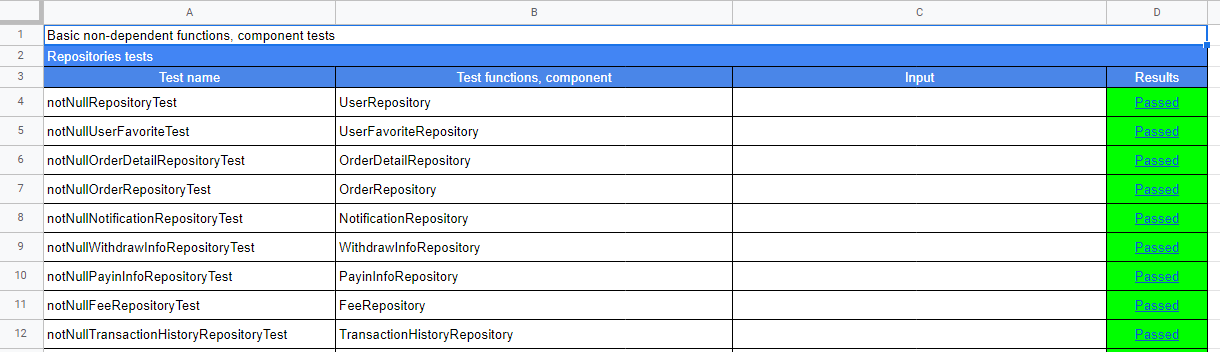


Figure 3-1: Unit testing sample

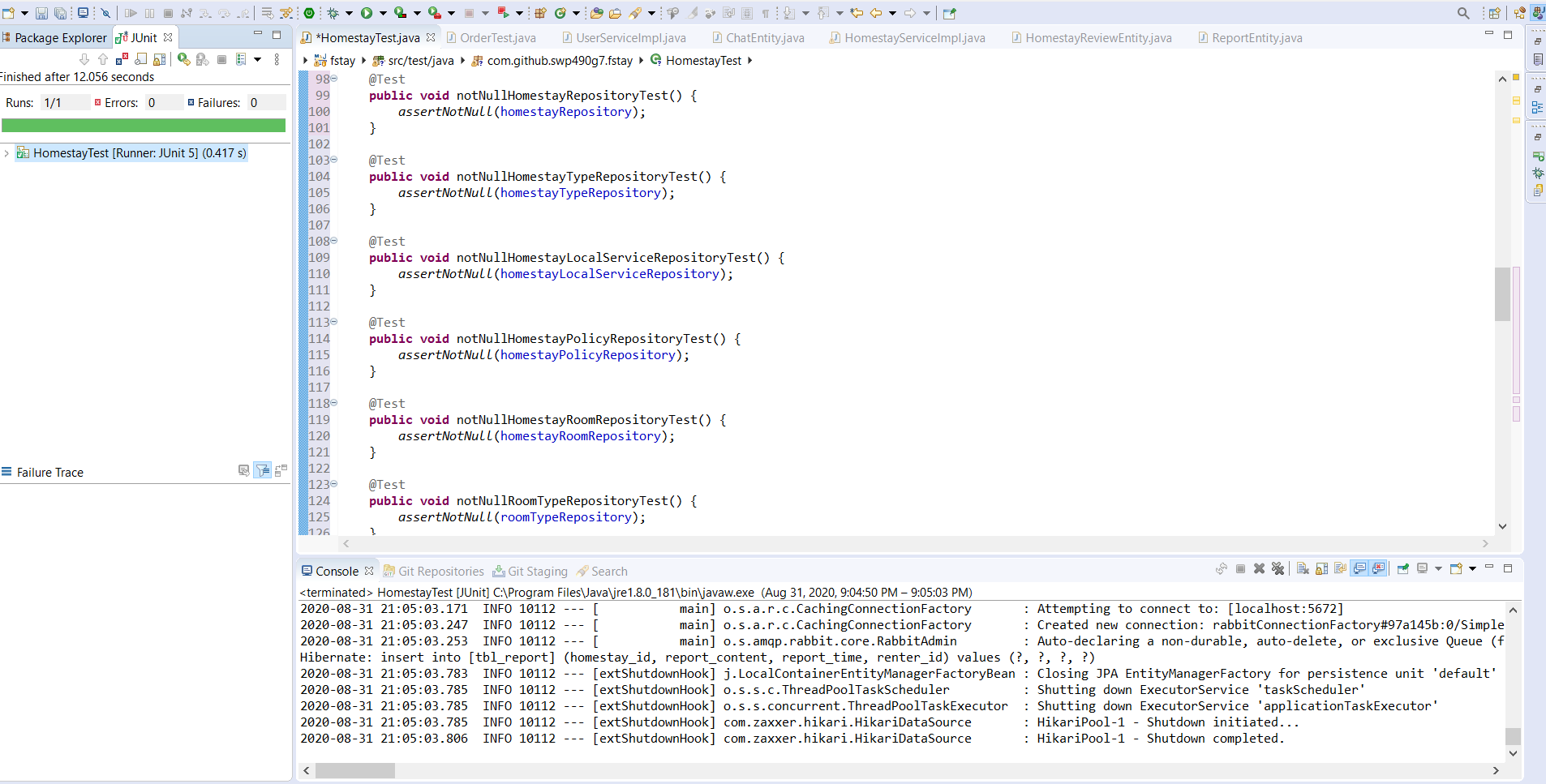


Figure 3-2: Using Spring Boot + Mockito + JUnit for Unit testing sample

### API Testing

API testing was performed by using Postman to test.

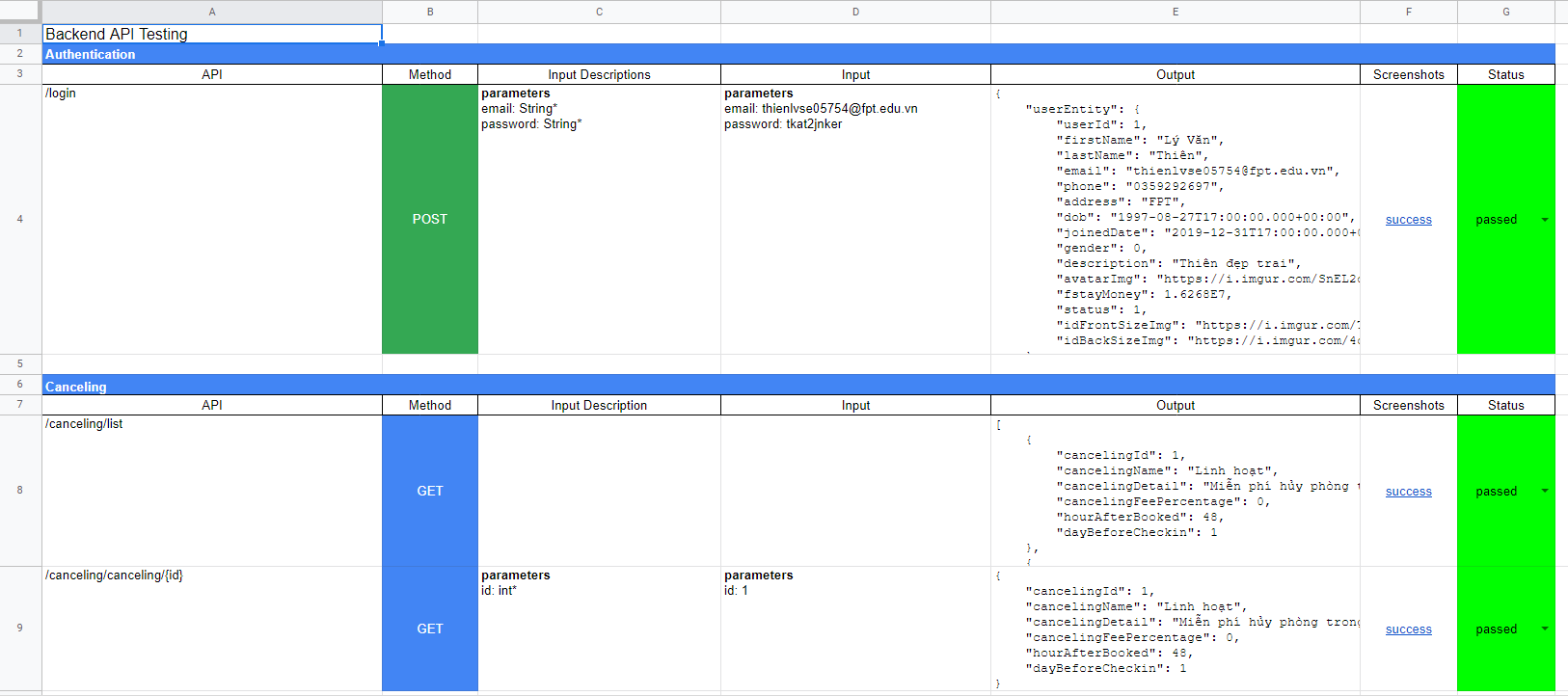


Figure 3-3: API testing sample

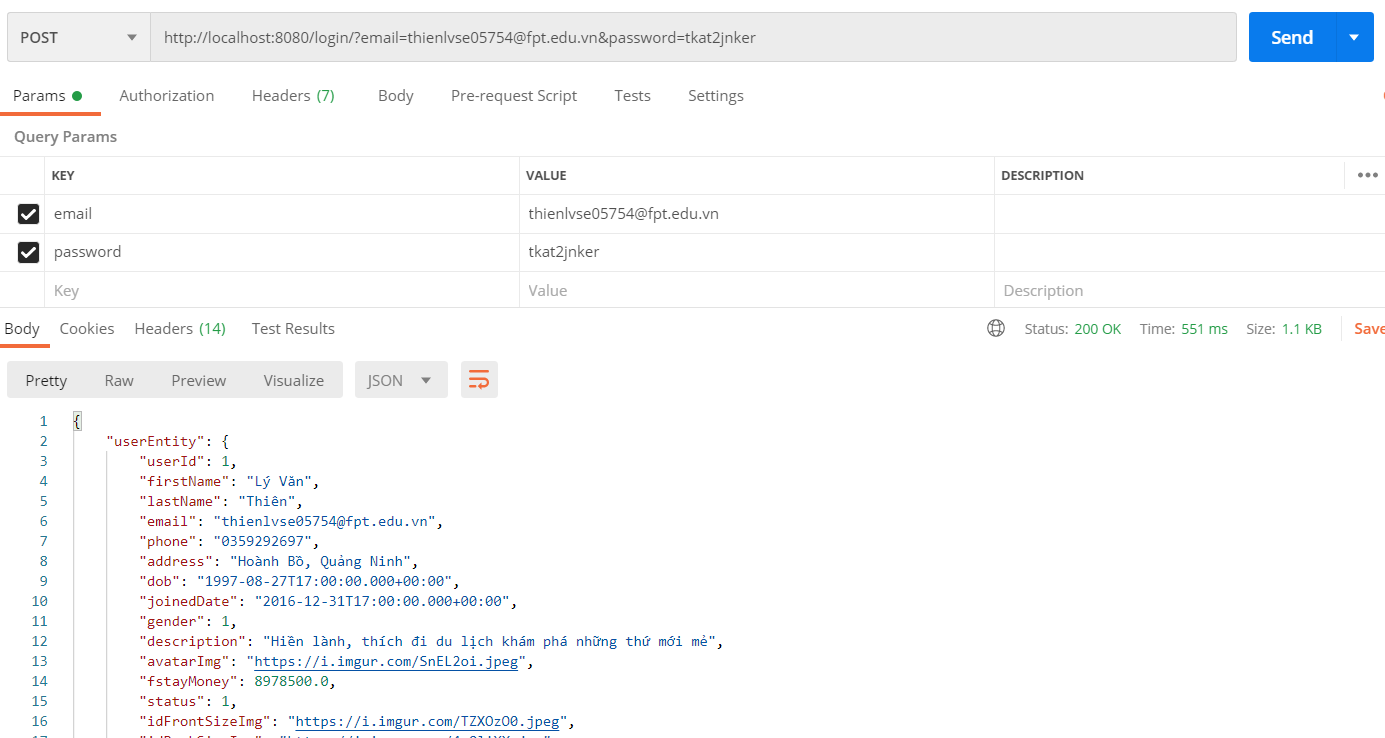


Figure 3-4: Using Postman for API testing sample

## System Testing

Detailed test cases will be described in ***Fstay***\_***SystemTestcases\_FinalReport.xlsx*** file.

As a standard definition, Fstay Project defines that a test case is:

* A set of test data and test programs (test scripts) and their expected results. A test case validates one or more system requirements and generates a pass or fail.
* A good test case should follow two basic aspects, the Contents and the Style. 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 system testing, ATSS 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.

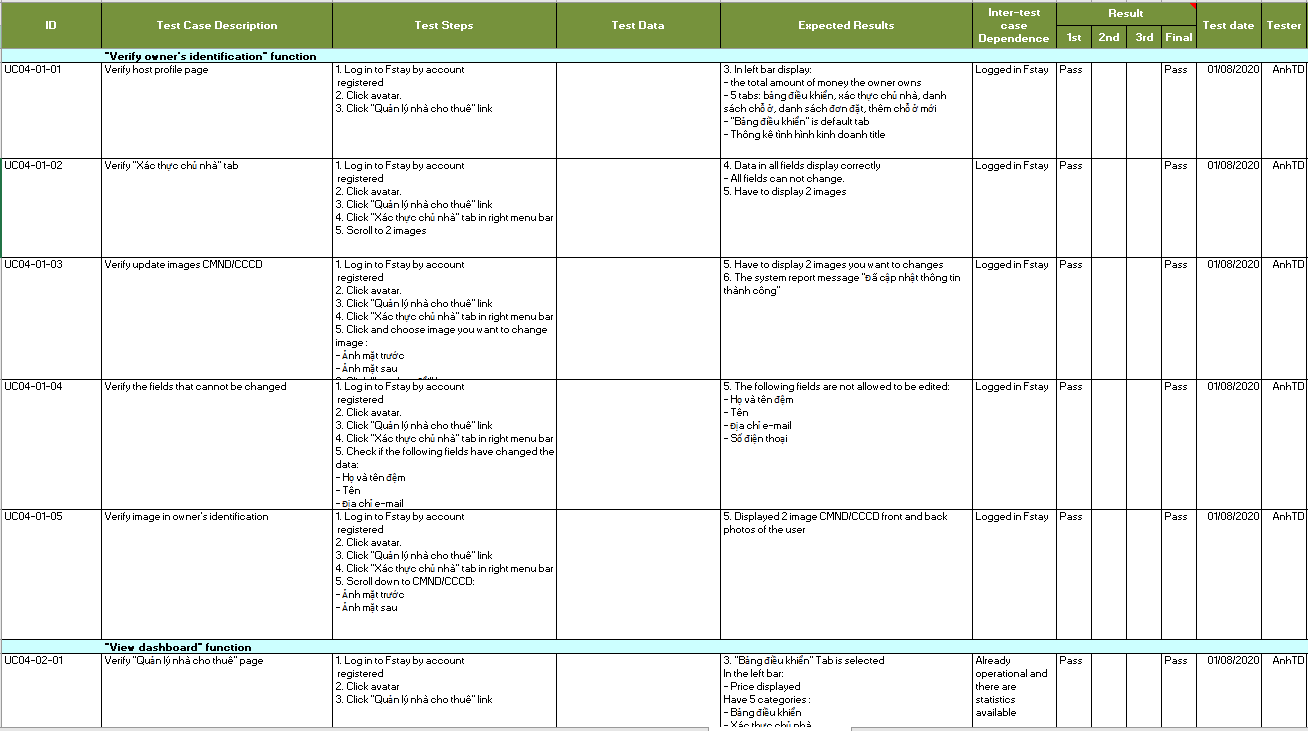


Figure 3-.5: Test cases list sample

# CHECK LIST

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Check List | Yes | No |
| **General** | | | |
| CL-001 | Text on all pages for spelling and grammatical errors. | x |  |
| CL-002 | Functionality of buttons available on all pages. | x |  |
| CL-003 | All mandatory fields are validated. | x |  |
| CL-004 | Validation error messages are displayed properly below the field. | x |  |
| CL-005 | All error messages are displayed in red color. | x |  |
| CL-006 | Delete functionality for any record on page are asked for confirmation. | x |  |
| CL-007 | All numeric values are formatted properly. | x |  |
| CL-008 | Application crash or unavailable pages are redirected to an error page. | x |  |
| **GUI and Usability** | | | |
| CL-009 | The screen well organized and easy to use. | x |  |
| CL-010 | All fields on page (e.g. text box, radio options, dropdown lists) should be aligned properly. | x |  |
| CL-011 | Information is arranged symmetrically with adequate spacing between components. | x |  |
| CL-012 | The most important fields are located where they are easy to see. | x |  |
| CL-013 | Information is presented in the order that the user needs it. | x |  |
| CL-014 | The screen designed to fit the requirements for international use. | x |  |
| CL-015 | The text easy to translate. Do not use slang, acronyms, and abbreviations. | x |  |
| CL-016 | Icons and images are designed impression and copyright. | x |  |
| CL-017 | Font size, style and color for headline, description test, labels, infield data, and grid information standard as specified in SRS. | x |  |
| CL-018 | Page Text must be standing by language locale. | x |  |
| CL-019 | Numeric values must be standing by language locale. | x |  |
| CL-020 | The static text is clear, concise, and meaningful. | x |  |
| CL-021 | Buttons follow the project standards for size and position (e.g., Negative button is correct size and to the right of the Positive button). | x |  |
| CL-022 | Graphical objects are used appropriately and according to the guidelines specified in Graphical User Interface Design. | x |  |
| CL-023 | A list view is used to allow a collection of items that are on a single hierarchical level. | x |  |
| CL-024 | A tree view is used to allow a collection of items to be displayed and manipulated within varying hierarchical levels. | x |  |
| CL-025 | Pop-up menus are provided for the user to access information about an object's properties or perform specific tasks on the object. | x |  |
| CL-026 | Command buttons are used to trigger application processes. | x |  |
| CL-027 | Toggle buttons are used to show independent on/off choices. Can change status for different answer. | x |  |
| CL-028 | System display notification message when meet trouble, error. | x |  |
| **Database** | | | |
| CL-029 | Correct data is getting saved in the database upon successful page submit. | x |  |
| CL-030 | Values columns are not accepting null values. | x |  |
| CL-031 | Data should be stored in single or multiple tables based on design. | x |  |
| CL-032 | Input data is not truncated. Field length shown to user on page and in database schema should be same. | x |  |
| CL-033 | Input numeric fields with minimum, maximum, and float values. | x |  |
| CL-034 | Input numeric fields with negative values (for both acceptance and non-acceptance). | x |  |
| CL-035 | Database fields are designed with correct data type and data length. | x |  |
| CL-036 | Input field leading and trailing spaces are truncated before committing data to database. | x |  |
| **Security** | | | |
| CL-037 | Test cookie allowed (disabled or allowed to be edited) | x |  |
| CL-038 | Test the form giving away security information if the source is viewed. | x |  |
| CL-039 | Test password security and password policy enforcement. | x |  |
| CL-040 | Check application logout functionality. | x |  |
| CL-041 | Cookie information should be stored in encrypted format only. | x |  |

Table 4-1: Check list

# TEST LOGS

## Defect Logs

Fstay project uses Google Sheet to log bugs.

Every member of Fstay Project use Google Drive: Google Sheet to take part in activities like control bugs, fix bugs, re-test bugs and close bug. Bug will be logged by tester and/or developer during the development progress.

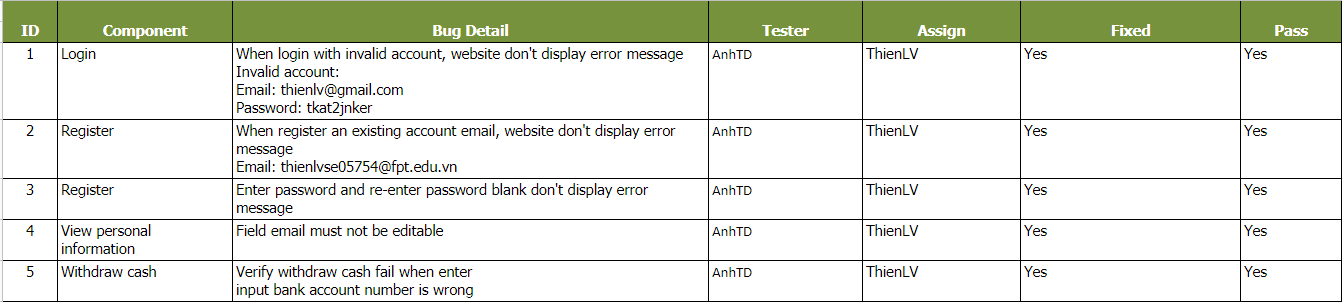


Figure 5-1: Defect Log Sample

## Test Reports

#### API Testing Report

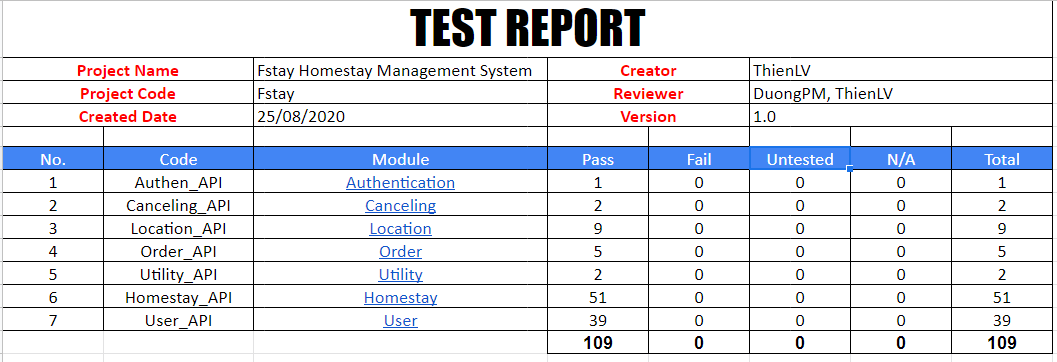


Figure 5-2: API testing report

#### Unit Testing Report

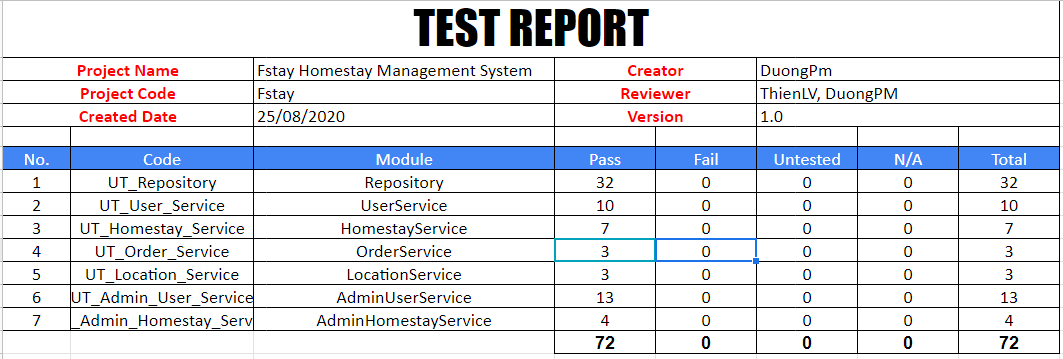


Figure 5-3: Unit testing report

#### System Testing Report

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| No | Module code | Pass | Fail | Untested | N/A | Number of test cases |
| **Common module for customer (renter, owner)** | | | | | | |
| 1 | Register | 41 | 0 | 0 | 0 | 41 |
| 2 | Login | 7 | 0 | 0 | 2 | 9 |
| 3 | Logout | 2 | 0 | 0 | 0 | 2 |
| 4 | Forget password | 4 | 0 | 0 | 0 | 4 |
| 5 | Change password | 14 | 0 | 0 | 0 | 14 |
| 6 | View personal information | 13 | 0 | 0 | 0 | 13 |
| 7 | Update personal information | 30 | 0 | 0 | 0 | 30 |
| 8 | View list transaction’s history | 23 | 0 | 0 | 16 | 39 |
| 9 | View notifications | 9 | 0 | 0 | 14 | 23 |
| 10 | Chat | 10 | 0 | 0 | 0 | 10 |
| **Renter module** | | | | | | |
| 11 | View homepage | 29 | 0 | 0 | 0 | 29 |
| 12 | Search homestay | 30 | 0 | 0 | 0 | 30 |
| 13 | View list homestay | 23 | 0 | 0 | 7 | 30 |
| 14 | View homestay details | 8 | 0 | 0 | 0 | 8 |
| 15 | Review homestay | 33 | 0 | 0 | 0 | 33 |
| 16 | Report homestay | 13 | 0 | 0 | 0 | 13 |
| 17 | Add favorite homestay | 3 | 0 | 0 | 1 | 4 |
| 18 | View list favorite homestay | 3 | 0 | 0 | 0 | 3 |
| 19 | Delete favorite homestay | 2 | 0 | 0 | 0 | 2 |
| 20 | Book homestay | 27 | 0 | 0 | 0 | 27 |
| 21 | View list order’s history | 3 | 0 | 0 | 0 | 3 |
| 22 | View order details | 3 | 0 | 0 | 0 | 3 |
| 23 | Cancel executing order | 3 | 0 | 0 | 1 | 4 |
| 24 | Pay in | 2 | 0 | 0 | 0 | 2 |
| 25 | View list pay in history | 4 | 0 | 0 | 9 | 13 |
| **Owner module** | | | | | | |
| 26 | Verify owner’ identification | 5 | 0 | 0 | 0 | 5 |
| 27 | View dashboard | 13 | 0 | 0 | 0 | 13 |
| 28 | Withdraw cash | 4 | 0 | 0 | 0 | 4 |
| 29 | View list withdraw cash’s history | 3 | 0 | 0 | 0 | 3 |
| 30 | Add new homestay | 142 | 0 | 0 | 14 | 156 |
| 31 | View list owner’s homestay | 2 | 0 | 0 | 0 | 5 |
| 32 | View owner’s homestay details | 1 | 0 | 0 | 1 | 2 |
| 33 | Update homestay’s information | 10 | 0 | 0 | 2 | 12 |
| 34 | Update homestay’ status | 1 | 0 | 0 | 1 | 1 |
| 35 | Add discount | 0 | 0 | 0 | 1 | 1 |
| 36 | View list order | 4 | 0 | 0 | 0 | 4 |
| 37 | View order details | 4 | 0 | 0 | 0 | 4 |
| 38 | Cancel executing order | 0 | 0 | 0 | 4 | 4 |
| **Admin module** | | | | | | |
| 39 | Login | 7 | 0 | 0 | 2 | 9 |
| 40 | Logout | 2 | 0 | 0 | 0 | 2 |
| 41 | Forget password | 3 | 0 | 0 | 0 | 3 |
| 42 | View personal information | 13 | 0 | 0 | 0 | 13 |
| 43 | Update personal information | 6 | 0 | 0 | 0 | 6 |
| 44 | Change password | 7 | 0 | 0 | 0 | 7 |
| 45 | View transaction | 39 | 0 | 0 | 0 | 39 |
| 46 | View dashboard | 6 | 0 | 0 | 0 | 6 |
| 47 | Change customer’s wallet | 6 | 0 | 0 | 0 | 6 |
| 48 | View list renter | 2 | 0 | 0 | 0 | 2 |
| 49 | View renter’s information details | 4 | 0 | 0 | 0 | 4 |
| 50 | Change renter’s status | 2 | 0 | 0 | 0 | 2 |
| 51 | View list owner | 1 | 0 | 0 | 0 | 1 |
| 52 | View owner’s information details | 7 | 0 | 0 | 0 | 7 |
| 53 | Change owner’s status | 1 | 0 | 0 | 0 | 1 |
| 54 | View list homestay | 3 | 0 | 0 | 0 | 3 |
| 55 | View homestay details | 5 | 0 | 0 | 0 | 5 |
| 56 | Approve homestay | 3 | 0 | 0 | 0 | 3 |
| 57 | Change homestay’s status | 3 | 0 | 0 | 0 | 3 |
| 58 | View list promotion | 1 | 0 | 0 | 0 | 1 |
| 59 | Add new promotion | 6 | 0 | 0 | 0 | 6 |
| 60 | Change promotion’s status | 1 | 0 | 0 | 0 | 1 |
| 61 | View list utility | 2 | 0 | 0 | 0 | 2 |
| 62 | Add new utility | 2 | 0 | 0 | 0 | 2 |
| 63 | Delete utility | 1 | 0 | 0 | 0 | 1 |
| 64 | View list commission fee | 1 | 0 | 0 | 0 | 1 |
| 65 | Add new commission fee | 10 | 0 | 0 | 0 | 10 |
| 66 | Update commission fee | 2 | 0 | 0 | 0 | 2 |
| 67 | Delete commission fee | 1 | 0 | 0 | 0 | 1 |
| 68 | View list staff | 1 | 0 | 0 | 0 | 1 |
| 69 | View staff’s information details | 1 | 0 | 0 | 0 | 1 |
| 70 | Add new staff | 23 | 0 | 0 | 6 | 29 |
| 71 | Change staff’s positon | 2 | 0 | 0 | 0 | 2 |
| 72 | Change staff’s status | 2 | 0 | 0 | 0 | 2 |
|  | Total | 717 | 0 | 0 | 81 | 798 |

Table 5-1: System test cases report

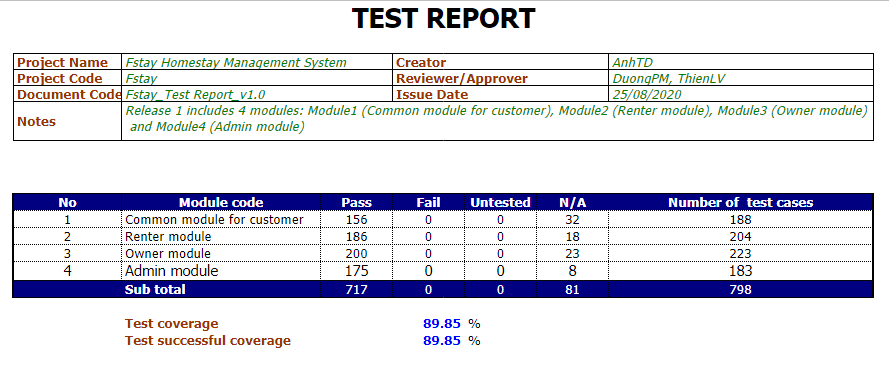


Figure 5-4: System testing report

#### GUI Testing Report

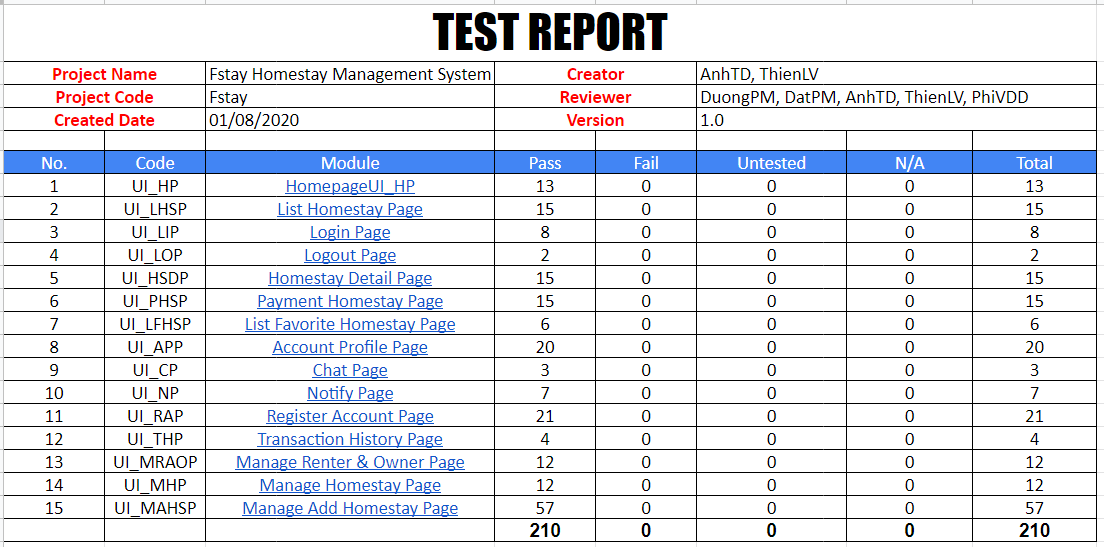


Figure 5-5: GUI testing report

#### GUI Testing Report

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| No. | Type of Testing | Pass | Failed | Untested | N/A | Total |
| 1 | Unit Test | 72 | 0 | 0 | 0 | **72** |
| 2 | Integration Test | 109 | 0 | 0 | 0 | **109** |
| 3 | System Test | 717 | 0 | 0 | 81 | **798** |
| 4 | GUI Test | 210 | 0 | 0 | 0 | **210** |
| Total | | **1108** | **0** | **0** | **0** | **1189** |

Table 5.5.2: Summary testing report