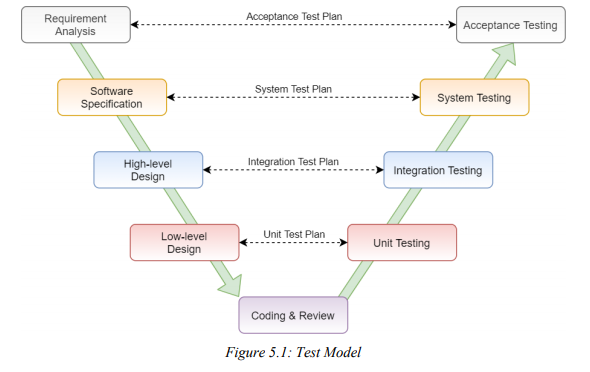
**V. Software Testing Documentation**

**1. Overall Description**

**1.1 Test Model**

****

**1.2 Testing Levels**

|  |  |  |
| --- | --- | --- |
| **No** | **Level** | **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 Whitebox Testing method is usually used to get the job done. |
| 2 | Integration 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 Whitebox Testing method is usually used to get the job done. |
| 3 | System 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 Whitebox Testing method is usually used to get the job done. |
| 4 | Acceptance Testing | User Acceptance Testing is conducted to determine whether the system is ready for release. During this phase, the tester and some alpha test user will test the system to find out whether the application meets their business’ needs. |

Table 5.1: Testing Levels

**1.3 Testing Types**

|  |  |  |
| --- | --- | --- |
| **No** | **Testing Types** | **Description** |
| 1 | Functional Testing | - Testing all individual implemented methods, functions of model class or library class.  - Test case will have to cover all logic branch that function or method could execute with difference data input |
| 2 | 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 Back-end Service.  - Test case will verify constraint of data which be mention in Business rule.  - All API will be tested with Swagger and Postman tools. |
| 3 | Algorithm Testing | -Algorithm testing is used to test about the speed and accuracy of algorithm with different similarity measures and difference input data to evaluate the performance of algorithm. |
| 4 | UI/UX Testing | - UI/UX testing verifies a user’s interaction with the software. The goal of UI/UX testing is to ensure that the UI/UX provides the user with an appropriate access and navigation through the functions of the target-of-test. In addition, UI/UX testing ensures that the objects within the UI/UX function as expected and conform to requirement.  - UI/UX test will be performed fully on all screens.  - UI/UX test targets to cover the verification of the overall look and feel of the system including initial position, font, text size, color, focus, initial button, tab order, label, screen sizes, sentences width and animation.  - Check all the UI elements for color, size, position, width, length and acceptance of characters or numbers. |
| 5 | Regression Testing | Regression Testing confirms that the bug was removed including the extent of the impact, when developers fix bug, developers and testers will confirm with each other what is the impacts of fix bug modification, after that all impact unit or function will be retested by developers then testers after that |
| 6 | Acception Testing | UAT takes place at close relation user’s site and are free test to detect bug and strange behavior. By that, development team will improve the system |

Table 5.2: Testing Types

**2. Test Plan**

**2.1 Test Stages**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Type of Test** | **Level of Test** | | | |
| **Unit Testing** | **Integration Testing** | **System Testing** | **Acceptance Testing** |
| Acceptance Testing | x | x | x |  |
| API Testing | x | x |  |  |
| Algorithm Testing | x | x |  |  |
| UI/UX Testing |  | x | x | x |
| Regression Testing | x | x | x | x |
| Acception Testing |  |  |  | x |

Table 5.3: Test Stages

**2.2 Resources**

**2.2.1 Human Resources**

|  |  |  |
| --- | --- | --- |
| **No** | **Resources** | **Responsibilities** |
| 1 | PM | - Responsible for project schedule and overall success of the project.  - Review test cases and reports. |
| 2 | Tester | - Preforming the actual system testing.  - Manage test resource and assign test tasks.  - Create test plan.  - Create test cases.  - Create test report.  - Execute test.  - Test log report. |
| 3 | Developer | - Create unit test and integration test scripts.  - Fix bugs. |

Table 5.4: Human Resources

**2.2.2 Tools and Environment**

**2.2.2.1 Tools**

**• Jira**: Manage issues

**• Google Sheets, Microsoft Excel , Microsoft Word**: Manage test cases

**• Postman, Swagger**: Manage and test API

**• Chrome DevTools**: View logs, inspect element front-end

**• Vue.js DevTools**: Inspect Vue component and Vuex state tree

**• Mockito**: Generate unit tests in back-end

**• DBeaver, Navicat , SQLyog** : Verify data has updated in database

**2.2.2.2 Environment**

|  |  |  |
| --- | --- | --- |
| **Testing Types** | **Software** | **Hardware** |
| Unit Testing, API Testing & Algorithm Testing | - Mockito v3.1.0  - IntelliJ IDEA  - Postman  - Swagger  - DBeaver  - Navicat | Personal computer for testing with the minimum configuration: Windows 10 Professional 64 bit, Intel® Core™ i5, RAM 8.00GB |
| Integration Testing, System Testing & Acceptance Testing | - Google Sheets  - Microsoft Excel 2016  - Chrome - Jira | Personal computer for testing with the minimum configuration: Windows 10 Professional 64 bit, Intel® Core™ i5, RAM 8.00GB |

Table 5.5: Testing Environment

**2.3 Test Milestones**

|  |  |  |
| --- | --- | --- |
| **Milestone Task** | **Start Date** | **End Date** |
| **Iteration 1: Anest Web Application** |  |  |
| Unit Testing, API Testing, Algorithm Testing |  |  |
| Integration Testing |  |  |
| System Testing |  |  |
| Regression Testing |  |  |
| UAT |  |  |
| **Iteration 2: Admin Web Application** |  |  |
| Unit Testing, API Testing |  |  |
| Integration Testing |  |  |
| System Testing |  |  |
| Regression Testing |  |  |
| UAT |  |  |

Table 5.6: Test Milestones

**2.4 Deliverables**

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Deliverables** | **Due Date** | |
| **Iteration 1** | **Iteration 2** |
| 1 | Test plan |  |  |
| 2 | Test cases |  |  |
| 3 | Test case review |  |  |
| 4 | Defect Report |  |  |
| 5 | Final Test Summary Report |  | |

Table 5.7: Testing Deliverables

**3. Test Cases**

**3.1 Unit Testing**

All unit tests are located in the folder src/test in source code. Here is an example of unit test cases:



Figure 5.2: Unit Test Example

**3.2 Integration Testing & System Testing**

Detailed test cases of these types of testing are described in 2 files: Fusu\_IntegrationTest.xlsx (for Integration Testing) and Fusu \_SystemTest.xlsx (for System Testing).

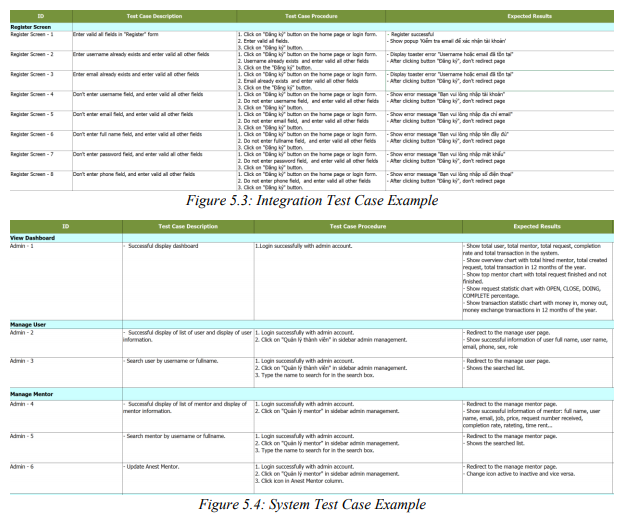
As a standard definition, Fusu Project defines that a test case is:

• 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 content 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.

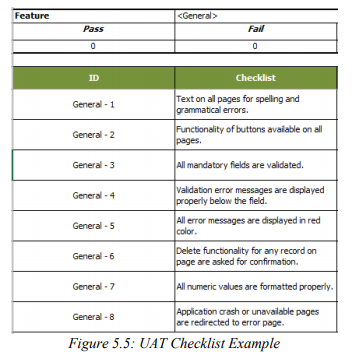
Integration testing and system testing are done by testers to ensure that each high-level module and the whole system work properly. In integration testing, testers test each module function follow each screen in the system and in system testing, testers follow the use case list to do testing process. If having any bugs or defects, testers are responsible to create issue in Jira and report to developers to fix as soon as possible.

**Test case examples:**

****

**3.3 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 checklist as a substitute for acceptance testing. Detailed checklists are described in Fusu\_UAT.xlsx file.



**3.4 Issue Log**

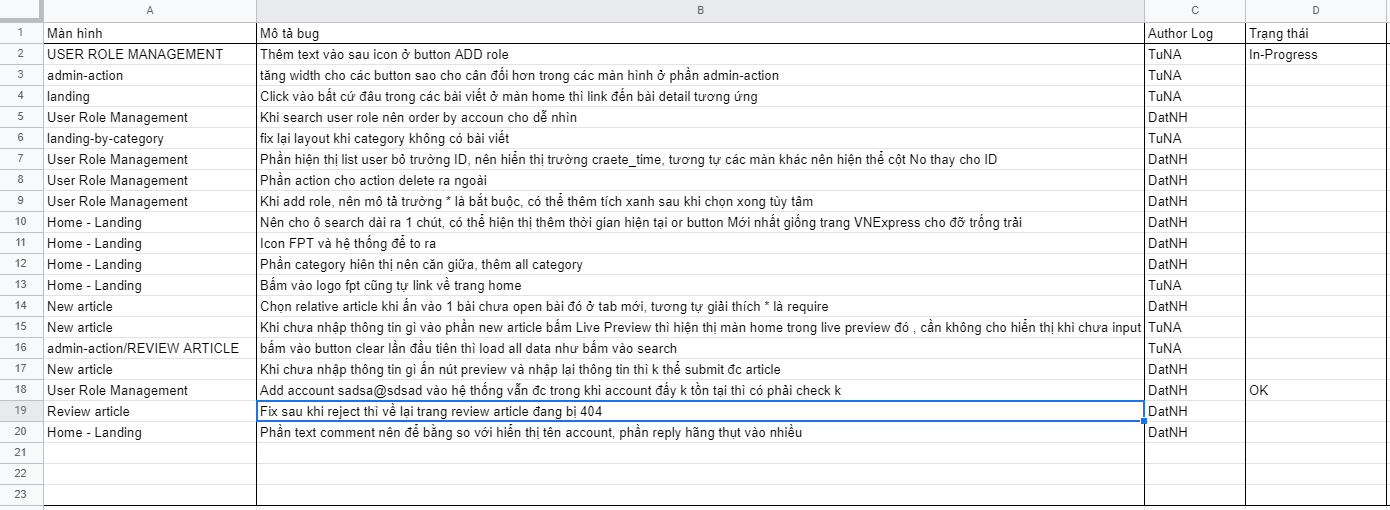
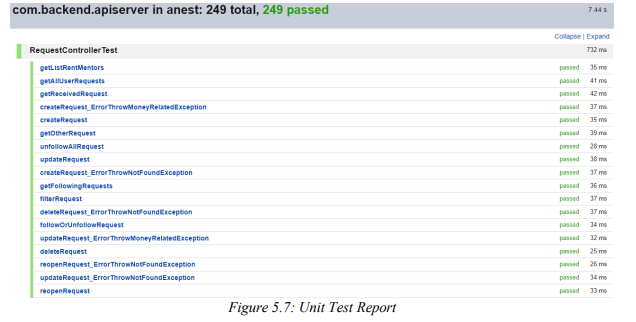
****

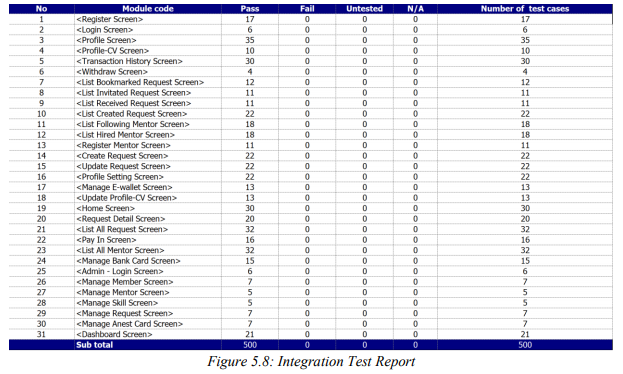
Figure 5.6: Issue Log in google sheet

**4. Test Reports**

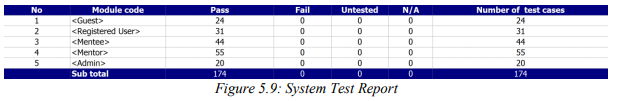
**4.1 Unit Test Report**

****

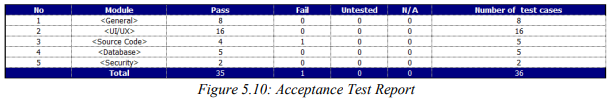
**4.2 Integration Test Report**

****

**4.3 System Test Report**

****

**4.4 Acceptance Test Report**

****

**5. Algorithm Evaluation**

**5.1 Data Test**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Dataset** | **Total User** | **Total Item** | **Total Rating** |
| 1 | MovieLens 100K | 943 | 1,682 | 100,000 |
| 2 | MovieLens 1M | 6,040 | 3,706 | 1,000,209 |
| 3 | MovieLens 10M | 69,878 | 10,677 | 10,000,054 |

Table 5.8: Data Test for Algorithm Evaluation

**5.2 Item-based Similarity Functions Speed Test**

The most important task to do in Item-Item Collaborative Filtering is to determine the similarity between the two items. There are different types of Item-based similarity functions:

• Tanimoto Coefficient Similarity

• Euclidean Distance Similarity

• Log Likelihood Similarity

• Pearson Correlation Similarity

• Uncentered Cosine Similarity

Fusu – Knowledge Connection System focuses on the realtime recommendation to improve the user experience. In this testing, we will test the recommendation time (ms) of the above similarity functions and it will be repeated 10 times with 10 suggestion items for a selected user.

**\* Dataset: MovieLens 100K**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Tanimoto Coefficient** | **Euclidean Distance** | **Log Likelihood** | **Pearson Correlation** | **Uncentered Cosine** |
| 1 | 1093 | 1277 | 1140 | 1279 | 1268 |
| 2 | 1032 | 1245 | 1185 | 1283 | 1378 |
| 3 | 1091 | 1269 | 1158 | 1235 | 1274 |
| 4 | 1112 | 1256 | 1161 | 1320 | 1250 |
| 5 | 1027 | 1246 | 1145 | 1276 | 1306 |
| 6 | 1033 | 1271 | 1174 | 1243 | 1247 |
| 7 | 1095 | 1316 | 1172 | 1274 | 1266 |
| 8 | 1092 | 1245 | 1153 | 1226 | 1238 |
| 9 | 1025 | 1235 | 1181 | 1282 | 1238 |
| 10 | 1039 | 1243 | 1130 | 1283 | 1253 |
| **Avg** | **1063.9** | **1260.3** | **1159.9** | **1270.1** | **1271.8** |

Table 5.9: Speed test report with MovieLens 100K dataset

Table 5.9: Speed test report with MovieLens 100K dataset

**\* Dataset: MovieLens 1M**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Tanimoto Coefficient** | **Euclidean Distance** | **Log Likelihood** | **Pearson Correlation** | **Uncentered Cosine** |
| 1 | 3140 | 4418 | 3216 | 4784 | 3995 |
| 2 | 3341 | 4195 | 3274 | 4256 | 4060 |
| 3 | 3082 | 3878 | 3099 | 3872 | 3828 |
| 4 | 3169 | 4249 | 3386 | 3942 | 4320 |
| 5 | 3134 | 3828 | 3201 | 4060 | 4162 |
| 6 | 3074 | 3904 | 3095 | 3865 | 3821 |
| 7 | 3369 | 4150 | 3207 | 4111 | 4079 |
| 8 | 3267 | 4109 | 3229 | 4212 | 4190 |
| 9 | 3165 | 4251 | 3308 | 4494 | 4163 |
| 10 | 3115 | 3816 | 3110 | 3774 | 3860 |
| **Avg** | **3185.6** | **4079.8** | **3212.5** | **4137** | **4047.8** |

Table 5.10: Speed test report with MovieLens 1M dataset

After testing recommendation time, we found that Tanimoto Coefficient Similarity is the Item-based similarity function that has the fastest computation speed. In the next section, we will evaluate the performance of this similarity function with different large datasets.

**5.3 Tanimoto Coefficient Similarity Evaluation**

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Dataset** | **Speed (ms)** | **Score (%)** |
| 1 | MovieLens 100K | 1018 | 81.79 |
| 2 | MovieLens 1M | 3201 | 81.79 |
| 3 | MovieLens 10M | 28617 | 73.41 |

Table 5.11: Tanimoto Coefficient Similarity Evaluation Report

Because the more data, the slower the calculation speed, in the future, when there is a large amount of data, we will refine the algorithm including rating time considerations, as well as using caching model to improve the performance of system.