### *Mining Frequent Closed Itemsets with Charm/dCharm Algorithms*

**Test Plan**

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Version | Author | Description |
| 26/04/19 | V1.0.0 | Desislava Andreeva | Test plan is created |
| 07/05/2019 | V1.0.0 | Desislava Andreeva | Test plan revision |
| 09/05/2019 | V1.0.0 | Desislava Andreeva | Test plan review |

Table of Contents

[1. Overview](#_Toc326150504) 4

[1.1. Purpose](#_Toc326150505) 4

[1.2. Scope](#_Toc326150506) 4

2[. Definitions](#_Toc326150546) 5

3[. Analysis of Scope and Test Focus Areas](#_Toc326150509) 6

2[.1. Unit testing](#_Toc326150510) 6

2[.2. Functional testing 6](#_Toc326150511)

4[. Unit Test Objectives](#_Toc326150513) 7

5[. Functional Test Objectives](#_Toc326150514) 8

6[. Test Strategy](#_Toc326150522) 9

6[.1. Test level responsibility](#_Toc326150523) 9

6[.2. Test types and approach](#_Toc326150524) 9

6[.3. Document creation and test execution schedule](#_Toc326150526) 9

6[.4. Requirement, data, and resource provision plan](#_Toc326150527) 10

6[.5. Testing tools](#_Toc326150528) 10

7[. Entry and Exit Acceptance Criteria](#_Toc326150540) 11

7[.1. Unit testing entry criteria](#_Toc326150523) 11

7[.2. Unit testing exit criteria](#_Toc326150524) 11

7[.3. Functional testing entry criteria](#_Toc326150524) 11

7[.4. Functional testing exit criteria](#_Toc326150524) 11

8[. Administrative Plan](#_Toc326150541) 12

[8.1. Defect management](#_Toc326150545) 12

[9. Points of Contact 13](#_Toc326150548)

# Overview

## 1.1 Purpose

The purpose of this document is to define:

* The test scope and objectives
* The test responsibilities
* The test strategy for the levels and types of test for this release
* The entry and exit criteria
* The basis of the test estimates
* Any risks, issues and defects
* The test schedule

## 

## Scope

This document details the testing that will be performed for the *Mining Frequent Closed Itemsets with Charm/dCharm Algorithms*  project. It defines the overall testing requirements and provides an integrated view of the project test activities.

Its purpose is to document:

* What will be tested;
* How testing will be performed;
* What resources are needed, and when;

*This test plan describes only unit and functional testing. Tests, such as integration, regression, acceptance, etc. are out of scope for this document.*

# Definitions

The following acronyms and terms have been used through out this document

|  |  |
| --- | --- |
| **Term/Acronym** | **Definition** |
| QA | Quality assurance |
| BBT | Black-box testing |
| WBT | White-box testing |
| P | Primary |
| S | Secondary |
| STLC | Software Testing Life Cycle |

# Analysis of Scope and Test Focus Areas

## Unit testing

Unit testing is a software testing method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures, are tested to determine whether they are fit for use.

Unit testing need to be performed as part of writing the code. The developer should write test cases that will cover most of the code logic.

## Functional testing

Functional testing is a QA processand a type of BBT that bases its test cases on the specifications of the software component under test. Functions are tested by feeding them input and examining the output, and internal program structure is rarely considered (unlike WBT). Functional testing is conducted to evaluate the compliance of a system or component with specified functional requirements. Functional testing usually describes what the system does.

Functional testing does not imply that you are testing a function (method) of your module or class. Functional testing tests a slice of functionality of the whole system. It should be performed after the implementation is done.

# Unit Tests Objectives

*This section details the unit test objectives that will be covered by the project team.*

| Ref | Function from code | Test Objective | Evaluation Criteria | Priority |
| --- | --- | --- | --- | --- |
| 2.1 | getMapValueNumber(EVENT) | To demonstrate that the function maps the specific *event* item to the correct number. | The returned number should be equal to the expected one. | S |
| 2.2 | getMapValueNumber(COMPONENT) | To demonstrate that the function maps the specific *component* item to the correct number. | The number returned from the function should be equal to the expected one. | S |
| 2.3 | getMappedValue(EVENT) | To demonstrate that the function maps the given number to its specific *event* item value. | The string returned from the function  should be equal to the expected one. | S |
| 2.4 | getMappedValue(COMPONENT) | To demonstrate that the function maps the given number to its specific *component* item value. | The string returned from the function  should be equal to the expected one. | S |
| 2.5 | convertToTransactionDataset() | To demonstrate that the function will convert the given database file (with the specific structure) to transaction dataset and save it to the file path which the user specified. If this function passes without error we will be sure that 80% of the code logic will be covered (tested). | The function should return CSVC\_OK  and the converted file should appear in the file path specified at the beginning.  On error CSVC\_FILE\_NOT\_FOUND and CSVC\_OI\_ERR are returned and proper message appears. | P |
| 2.6 | convertResultsForUsers() | To demonstrate that the function will translate the output result (result from the mining algorithm) for the user. Normally, the output from charm/dCharm algorithms is not perfectly understandable for the user, so this function takes it and converts it to more readable format. | The function should return CSVC\_OK and the result should be written in the specified file. A message with the location and the name of the file should appear when the function finish. On error CSVC\_FILE\_NOT\_FOUND and CSVC\_OI\_ERR are returned and proper message appears. | P |

*Note: Ref column contains the numbers of the test cases from the test report document!*

# Functional Tests Objectives

*This section details the functional test objectives that will be covered.*

| Ref | Function | Test Objective | Evaluation Criteria | Priority |
| --- | --- | --- | --- | --- |
| 2.7 | Invalid data handling | Test the behavior of the software when files with extensions different that .txt (.txt or .csv for database file) are entered as user input. | The test should be marked as PASS when you are unable continue until entering file with .txt (.csv) extension. | P |
| 2.8 | Invalid data handling | Test the behavior of the software when not existing input (database) file is entered. | Proper messages should be printed, explaining the reason for the failure. | P |
| 2.9 | Proper conversion of the database | Test the behavior of the software when all input data is valid. | All statistics should be printed if test finishes successfully. All files that are generated from the software should be found at the specified paths. | P |

*Note: Ref column contains the numbers of the test cases from the test report document!*

# Test Strategy

## Test level responsibility

|  |  |  |  |
| --- | --- | --- | --- |
| Test Level | External Party | Project Team | User |
| Unit Testing | S | P |  |
| Functional Testing | S | P |  |
| User Acceptance Testing |  |  | P |

## 

## Test types and approach

This table provides details about the types of testing covered by the project team and their standard objectives.

|  |  |
| --- | --- |
| Test Type | Objectives |
| Unit testing | The objective are to verify:   * Implementation logic; * If the functions return expected results; * Code work with invalid data;   Unit testing is performed as part of the development process. |
| Functional testing | The objectives are to verify that the application:   * Meets the defined requirements; * Performs and functions accurately; * Correctly handles error conditions; * Data load is successful. * Data is validated;   Functional testing will occur in an iterative and controlled manner, ensuring the solution matches the defined requirements. |

## Documents creation and test execution schedule

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Number | Activity/Test | Priority | Start date | End date |
| 1 | Test plan creation | High | 26.04.2019 | 27.04.2019 |
| 2 | Unit tests implementation | High | 27.04.2019 | 28.04.2019 |
| 3 | Unit testing and report generation | High | 28.04.2019 | 29.04.2019 |
| 4 | Functional test cases design | High | 29.04.2019 | 29.04.2019 |
| 5 | Functional testing | High | 30.04.2019 | 30.04.2019 |
| 6 | Generating test report for functional test results | High | 01.05.2019 | 02.05.2019 |

## 

## Requirements, data and resource provision plan

### Testing Requirements

Each person involved in Unit testing will need the following:

* The application with all the libraries (smpf and etc.);
* The proper input file with the structure Time;Event context;Component;Event name;Description;Origin;IP address;
* Java JDK version (>=1.8) to compile the application;
* To download JUnit JAR and set JUNIT\_HOME and CLASSPATH environment variables;
* Eclipse to execute the JUnit tests;
* Access to Microsoft Word to generate test reports;

Each person involved in Functional testing will need the following:

* The application with all the libraries;
* The proper input file with the structure Time;Event context;Component;Event name;Description;Origin;IP address;
* Access to Microsoft Word to generate test reports;

### Data Requirements

As we mentioned above the data needed for testing should be with the following structure: Time;Event context;Component;Event name;Description;Origin;IP address;

The rest of the data needed is generated from the code.

### Resources and skills

Each person involved in Unit testing should have the following skills:

* Understanding of Java programming language to resolve any defects;
* Knowledge of working with JUnit framework;
* Should be familiar with the implementation logic;

Each person involved in Function testing should have the following skills:

* Basic understanding what the software do without any implementation logic knowledge;
* Knowledge what is the expected results when specific input is given to the software;

## Testing Tools

This section provides details about the tools used in the testing process.

| Process | Tool |
| --- | --- |
| Test plan creation | Microsoft Word |
| Test report creation | Microsoft Word |
| Coding convention document | Microsoft Word |
| Code Review Checklist | Microsoft Word |
| Unit tests execution | JUnit framework |
| Functional tests execution | Manual |

# Entry and Exit Criteria

Entry Criteria for STLC phases are specific conditions or all those documents which are required to start a particular phase of STLC should be present before entering any of the STLC phase. Entry criteria is a set of conditions that permits a task to perform, or in absence of any of these conditions, the task cannot be performed.

Exit Criteria for STLC phases can be defined as items/documents/actions/tasks that must be completed before concluding the current phase and moving on to the next phase. Exit criteria is a set of expectations; this should be met before concluding the STLC phase.

**7.1 Unit testing entry criteria**

To start Unit Test Cases development phase, the following expectations should be met:

* The test plan document should be ready;
* Complete understanding of the application flow is required;
* The code implementation should be done;

**7.2 Unit testing exit criteria**

To conclude Unit Test Cases development phase, the following expectations should be met:

* The test cases should be written and reviewed;
* The test data should be identified and ready;
* Result reports should be written;
* 100% of the test cases should be with status pass;

**7.3 Functional testing entry criteria**

To start Functional Test Cases execution phase, the following expectations should be met:

* The test plan should be ready;
* The test report with results of Unit test should be ready;
* All Unit test with High priority should be with status PASS;

**7.2 Functional testing exit criteria**

To conclude Functional Test Cases execution phase, the following expectations should be met:

* The test report with results should be written;
* All defects need to be described and their severity should be defined;
* All documented bugs with status “Critical” should be fixed;
* All documented bugs with priority “High” should be fixed;

# Administrative Plan

## Defect Management

When any defect is found during testing, the following expectations should be met:

* The defect should be described in the test report (section 5);
* If the defect is fixed it should be described in resolved test incidents;
* If the defect is not fixed it should be described in unresolved test incidents;
* If there is a doubt about a defect ( no test case for it and not tested ) it should be described as unresolved test incidents;
* The tool that is used is Microsoft Word;

Optionally, if the defects are too many, a separate document (Microsoft Excel table) should be created, meeting the following requirements:

* Every defect should have the severity and priority specified, to help the developer to know which one should be fixed first;
* The defects that breaks the basic functionality of the applications should be marked with severity “Critical” and priority “High”;
* The defects related to returned errors or how adequate the printed messages are, could be with severity from “Major” to “Minor” and priority from “Medium” to “Low”;

# Points of contact

The following people can be contacted in reference to this document

|  |  |
| --- | --- |
| Primary Contact | |
| **Name** | Desislava Petrova Andreeva |
| **Title/Organisation** | Developer/Tester |
| **Phone** | 0886505218 |
| **Email** | dessiandreeva10@gmail.com |
|  | |