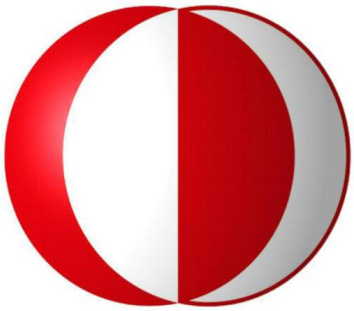
Middle East Technical University

Department of Computer Engineering



Music Mentor

Software Test Document

MemoryLeak

Bahtiyar Onur Geyik, 1745975

Emre Ercan, 1745918

Gökçe Aksu, 1745694

Ege Bozkurt, 1745819

Preface

This document contains the test specifications, test types and test cases for Music Mentor project. The document is prepared according to the “IEEE Standard for Software and System Test Documentation - IEEE STD 829-2008”

This test plan provides a complete description of all the test cases of Music Mentor. The first section of this document includes scope, level in overall sequence, test classes and overall test conditions and references subsections.

The second section contains details for system test of Music Mentor and there are features to be tested and features not to be tested, test items and their identifiers, test traceability matrix, approach, feature pass/fail criteria and test deliverables subsections in this section.

The third section is test management section. It includes planned activities and tasks; test progression, and environment/infrastructure.

Fourth section of this document includes system test cases in terms of test case name, objective, inputs and expected outputs.

Finally fifth section of the document contains system test report details in which we have test results, rationale for decisions, and a conclusion.

Table of Contents

1. Introduction

1.1 Document Identifier

This document is prepared by MemoryLeak team on May 25, 2014. This is the Version 1.0 of Software Testing Documentation.

1.2 Scope

This project is a music recommendation system that suggests new music to the user based on his/her previous music preferences. Using this application each user may discover new music based on the music one previously listened to. The project may be used for military purposes.

The purpose of the document is to explain the system testing of Music Mentor.
The document contains purposes, features and expected outputs of the tests which are run on the system. This document may be used by stakeholders. Stakeholders may use this document to see which parts of the project are ready to use and error-free.

1.3 References

[1] IEEE STD 829-2008, IEEE Standard for Software and System Test Documentation
[2] SDD of Music Mentor, Music Recommender System Software Design Description v1.1

1.4. Level in the Overall Sequence

1) Integration testing: The next level of testing is often called integration testing. In this level, modules in Music Mentor are combined together into different subsystems, which are then tested. The goal here is to see if the modules can be integrated properly. Hence, the emphasis

is on testing interfaces between modules. This testing activity can be considered testing the design.

2) System testing: Here the entire software system is tested. The reference document for this process is the requirements document, and the goal is to see if the software meets its requirements. This is essentially a validation exercise.

3) Acceptance testing: Acceptance Tests are the tests will be performed by the stakeholders to validate the system to see if the system meet their needs or not. Acceptance Tests will be performed after the system test and performance tests.

In this document we have only done system testing as single level testing was the only required testing process for this project.

2. Details for System Test Plan

2.1 Features to be Tested

Firstly in Music Mentor the functional properties are to be tested. While testing functional properties we use specific parameters and inputs in order to see the software's behavior and see the related errors. After the test of functional properties the non-functional features are to be tested. The non-functional features include properties like performance, reliability and storage requirements.

2.2 Features not to be Tested

During testing process, some efficiency properties are not to be tested. Since the log data cannot be increased dramatically, efficiency will not be tested.

2.3 Approach

The approaches used in testing of Music Mentor are black box method and analysis method.

2.4 Item Pass/Fail Criteria

In this testing report we may consider faults in two categories; deficiency and defect. Deficiencies are faults that do not block the software from its functionality. Defects are faults that stop or break down the software from running and they do not meet the requirements.

The result is evaluated in three criteria which are pass, conditional pass and fail. Pass means the test case is run and no defects or deficiencies are observed. Conditional pass means when the test case is applied there is at least one deficiency observed but there are no defects. Fail means the test procedure includes a defect so it does not meet requirements.

After the tests are run, the failed ones and conditionally passed ones will be analyzed and required changes will be done accordingly. After the changes are done then we will evaluate the situation and select a case in terms of regression. There may be no regression which means there is no need for regression testing because nothing on the general program structure is affected by the change. There may be regression which means the segment of the code where the change is made should be evaluated and some test cases may be run again. Lastly there may be full regression which means the changes affected general program structure so all changes should run again.

2.5 Test Deliverables

In Music Mentor, there will have two different deliverables. The first one is STD (Software Test Description). STD also consists of system level test procedures/cases which are given in Section

4 of this document. The last deliverable for testing activities will be Test Report which will be prepared according to IEEE STD 829-2008. In test report, the results of the designed testing activities will be summarized and will provide evaluations based on these results.

3. Test Management

3.1 Planned Activities and Tasks; Test Progression

The test process will start with analysis. The studies will be done about all the classes in our software: Recommender class and other private classes. Then there will be some computational studies about the subject like similarity algorithms and neighborhood algorithms. After analysis phase the inspection will start. Reading SSD of the project in order to understand the requirements and design test cases accordingly. Then the code will be read in order to understand it and find the error prone parts of the code in testing purposes. In the next step we will choose necessary test cases. For better understanding then we will divide the test cases into groups according to their objectives. Then the expected outputs for each test case will be decided and descripted. Finally the inputs/parameters for the test cases will be decided. The inputs/parameters will be decided according to what we expect to test in that test case.

In the final step the results of each test will be gathered in a test results table. This will enable us to see what should be done in further steps of the software's development. The testing of this project will be done manually, not by test automation tools.

3.2 Environment/Infrastructure

Test process has following hardware and software needs:

Hardware Needs: A java-supported operating system installed computer and peripheral devices.
Software Needs: Java Virtual Machine (JVM), Java Runtime Environment (JRE), Java Development Kit (JDK).

4. Test Case Details

4.1 Introduction

This section embodies the detailed explanation for each test case accompanied by the inputs, outcomes, environmental and procedural requirements along with the dependencies among test cases. Environmental requirements require that in order to apply all test cases, whole system must be implemented and Java Environment must be installed to computer.

This section includes the information for all test cases we run on the project. For each test case there are 4 fields; test case id, test case description, inputs/parameters, expected output. Test case id is unique for each test case and is used for identifying test cases.

Test case description explains why the test is run on the project. Inputs are what inputs are to be given to the program in order to run that test and parameters are changes in source code in order to see the behavior of the program. Expected output is what we desire to see after the test is applied. There are no specific environmental needs, special procedural requirements and inter-case dependencies. However, to apply all test cases, Java Environment must be installed to the computer.

4.2 Cross Platform Testing

This test allows us to see if the project runs on different operating systems.

Test Case Id	Test Case Description	Tools	Expected Output
CP_LINUX	Checks if the application works properly on Linux based operating system driven computers.	UBUNTU operating system installed computer.	Application works properly on JVM.
CP_MAC	Checks if the application works properly on Mac OS based operating system driven computers.	Mac OS installed computer.	Application works properly on JVM.
CP_WINDOWS	Checks if the application works properly on Windows based operating system driven computers.	Windows operating system installed computer.	Application works properly on JVM.

4.3 Constraint Testing

This testing is done in order to see whether the program runs within the given boundaries.

Test Case Id	**Test Case** **Description**	**Parameters**	**Expected Output**
CT_LSTNSNG_5	Checks if the system recommends songs to a user who listened to a given number of songs in the parameter.	A user which listened to 5 songs.	System recommends as expected
CT_LSTSNG_20	Checks if the system recommends songs to a user who listened to a given number of songs in the parameter.	A user which listened to 20 songs.	System recommends as expected
CT_LSTSNG_100	Checks if the system recommends songs to a user who	A user which listened to 100 songs.	System recommends as expected

	listened to a given number of songs in the parameter.		
CT_LSTSNG_250	Checks if the system recommends songs to a user who listened to a given number of songs in the parameter.	A user which listened to 250 songs.	System recommends as expected
CT_LSTSNG_500	Checks if the system recommends songs to a user who listened to a given number of songs in the parameter.	A user which listened to 500 songs.	System recommends as expected
CT_LSTSNG_1000	Checks if the system recommends songs to a user who listened to a given number of songs in the parameter.	A user which listened to 1000 songs.	System recommends as expected

4.4 Performance Testing

This test is done in order to see if the program can run on extreme situations.

Test Case Id	**Test Case** **Description**	**Parameters**	**Expected Output**
PT_LOGSIZE_1M	Checks if the system recommends songs to a user within feasible time with a log file size of given number of logs.	A generated log file with approximately 1 million logs.	System recommends as expected in a feasible time.
PT_LOGSIZE_5M	Checks if the system recommends songs to a user within feasible time with a log file size of given number of logs.	A generated log file with approximately 5 million logs.	System recommends as expected in a feasible time.
PT_LOGSIZE_15M	Checks if the system recommends songs to a user within feasible time with a log file size of given number of logs.	A generated log file with approximately 15 million logs.	System recommends as expected in a feasible time.
PT_LOGSIZE_30M	Checks if the system recommends songs to a user within	A generated log file with approximately 30 million logs.	System recommends as expected in a feasible time.

	feasible time with a log file size of given number of logs.		
PT_LOGSIZE_47M	Checks if the system recommends songs to a user within feasible time with a log file size of given number of logs.	A generated log file with approximately 47 million logs.	System recommends as expected in a feasible time.
PT_GEN_LOGS1	Checks if the system can generate a log file properly with given number of parameters.	Dataset with 144,292 different users and 109,147 different songs.	System generates log files without causing any error.
PT_GEN_LOGS2	Checks if the system can generate a log file properly with given number of parameters.	Dataset with 408,489 different users and 257,964 different songs.	System generates log files without causing any error.

4.5 Scenario Testing

The general purpose of this type of testing is to see correctness of software with different scenarios.

Test Case Id	Test Case Description	Parameters	Expected Output
SC_RCM_FNC	The behavior of the software is checked when a wrong userId is given to public List<Long> recommend(Long userId)	userId	Software throws an exception: User does not exist
SC_RCM_FNC2	The behavior of the software is checked when a wrong userId is given to public List<Long> recommend(Long userId, Long songId)	userId	Software throws an exception: User does not exist
SC_LIST_FNC_SNG	The behavior of the software is checked when a wrong songId is given to public List<Long> recommend(Long userId, Long songId)	songId	Software throws an exception: Song does not exist
SC_TRN_FP	The behavior of the	folderPath	Software throws an

	software is checked when a wrong folderPath is given to public void train(String folderPath, String workspacePath)		appropriate exception
SC_TRN_INVLD	The behavior of the software is checked when the folder in the given folderPath to the function public void train(String folderPath, String workspacePath) contains any documents other than user logs, directories which contain the user logs and rating.csv,	folderPath	Software throws an appropriate exception
SC_TRN_WP	The behavior of the software is checked when the given workspacePath to the function public void	workspacePath	Software throws an appropriate exception and does not create a folder

	train(String folderPath, String workspacePath) is not valid		
SC_LWS	The behavior of the software is checked when the given workspacePath to the function public void loadWorkspace(String workspacePath) is not valid.	workspacePath	Software throws an appropriate exception

5. System Test Report Details

5.1. Overview of Test Results

The testing procedure of Music Mentor included cross-platform testing, constraint testing, performance testing and scenario testing. The documentation contains detailed explanation of all test cases that are run on the software. The test cases are tested by trying the inputs in different situations and changing the parameters in the code.

5.2. Detailed Test Results

No	Test Case Id	Result
1	CP_LINUX	PASS
2	CP_MAC	PASS
3	CP_WINDOWS	PASS
4	CT_LSTSNG_5	PASS
5	CT_LSTSNG_20	PASS
6	CT_LSTSNG_100	PASS
7	CT_LSTSNG_250	PASS
8	CT_LSTSNG_500	PASS
9	CT_LSTSNG_1000	PASS
10	PT_LOGSIZE_1M	PASS
11	PT_LOGSIZE_5M	PASS
12	PT_LOGSIZE_15M	PASS
13	PT_LOGSIZE_30M	PASS
14	PT_LOGSIZE_47M	PASS
15	PT_GEN_LOGS1	PASS
16	PT_GEN_LOGS2	PASS
17	SC_RCM_FNC	PASS
18	SC_RCM_FNC2	PASS
19	SC_LIST_FNC_SNG	PASS
20	SC_TRN_FP	PASS
21	SC_TRN_INVLD	PASS
22	SC_TRN_WP	PASS
23	SC_LWS	PASS

5.3. Rationale for Decisions

We have chosen the test cases to see the behavior of the code in error prone parameters and inputs. The test cases were tested number of times and the behaviors are observed using our evaluator.

5.4. Conclusions and Recommendations

In our testing process we have tested 23 test cases. The cases included performance testing, cross-platform testing, constraint testing and scenario testing.