Group 3 Team 5

DB Edit Software

Test plan

Version <1.0>

<April 2020>

Document Control

Approval

The Guidance Team and the customer shall approve this document.

Document Change Control

|  |  |
| --- | --- |
| Initial Release: | 1.0 |
| Current Release: | 1.0 |
| Indicator of Last Page in Document: | $ |
| Date of Last Review: |  |
| Date of Next Review: |  |
| Target Date for Next Update: |  |

Distribution List

This following list of people shall receive a copy of this document every time a new version of this document becomes available:

Guidance Team Members:

Dr. Steve Roach

Customer:

Software Team Members:

Nouri, Ali

Guajardo, Patricia S.

Sanchez, Briana

Change Summary

The following table details changes made between versions of this document

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Modifier | Description |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Note: The template presented in this document was taken from:

Donaldson, S., and S. Siegel, *Successful Software Development*. Upper Saddle River, NJ: Prentice Hall, 2001, pp. 321-323.

Note: The template presented in this document was taken from: Donaldson, S., and S. Siegel, *Successful Software Development*. Upper Saddle River, NJ: Prentice Hall, 2001, pp. 321-323 and modified by Humberto Mendoza and Steve Roach.

Supplementary information is from:

Pfleeger, S. *Software Engineering, Theory and Practice*. Upper Saddle River, NJ: Prentice Hall, 1998, p. 365.

Table of Contents

[Document Control ii](#_Toc37538219)

[Approval ii](#_Toc37538220)

[Document Change Control ii](#_Toc37538221)

[Distribution List ii](#_Toc37538222)

[Change Summary ii](#_Toc37538223)

[1. Introduction 1](#_Toc37538224)

[1.1. Purpose 1](#_Toc37538225)

[1.2. Scope 1](#_Toc37538226)

[1.3. System Overview 1](#_Toc37538227)

[1.4. Suspension and Exit Criteria 1](#_Toc37538228)

[1.5. Document Overview 2](#_Toc37538229)

[1.6. References 2](#_Toc37538230)

[2. Test Items and Features 3](#_Toc37538231)

[2.1. Undo 3](#_Toc37538232)

[2.2. Copy Row 3](#_Toc37538233)

[2.3. Paste Row 3](#_Toc37538234)

[2.4. Insert Row 3](#_Toc37538235)

[2.5. Delete Row 3](#_Toc37538236)

[2.6. Search 3](#_Toc37538237)

[2.6.1. Search and Replace Elements 3](#_Toc37538238)

[2.6.2. Find Match Cases 4](#_Toc37538239)

[2.6.3. Find Match Whole Words 4](#_Toc37538240)

[3. Testing Approach 5](#_Toc37538241)

[4. Test Edit Menu Functions 6](#_Toc37538242)

[4.1. Requirements 6](#_Toc37538243)

[4.2. Test Case 1 6](#_Toc37538244)

[4.3. Test Case 2 and Test Case 3 7](#_Toc37538245)

[4.4. Test Case 4 8](#_Toc37538246)

[4.5. Test Case 5 9](#_Toc37538247)

[4.6. Test Case 6 10](#_Toc37538248)

[4.7. Test Case 7 11](#_Toc37538249)

[5. Test Schedule 14](#_Toc37538250)

[6. Appendix 15](#_Toc37538251)

# Introduction

In this test plan, we try to provide a comprehensive test suite for a database editor. Database editor is a software that stores data in XML files. There is no relation between the columns. This editor allows users to access the database, read, add, and modify the data. The goal of this test plan is to cover all possible bugs or glitches in the software. Here, we focus on edit the menu functionality of the software. We do not have any access to the code of the software. Thus, our approach is black-box testing. Each test cases explain with all the necessary details. The test cases are covered undo, copy row, paste row, insert row, delete row, and search functions in the editor.

## Purpose

This test plan contains all the requirements needed to test the software, a test suite that covers all editing menu functionality, and documentation of all the steps to apply each test case. The purpose of this project is to provide a comprehensive test plan document. This document covers all the steps that needed to test each test case, elaborate on the purpose of each test case, and analyze the performance of the system after each test case.

## Scope

This test plan started on April 8th and finished on April 21st. The test plan improved by the review of other team members and the writer used their feedback to provide a comprehensive test plan. GitHub was the platform to communicate between the team members.

## System Overview

The system is a database editor that uses XML to store the data. This system is an interface for users to manipulate the data in the database. User can add, remove, change, or search the data easier by using this system. When the user wants to start working with the editor, he/she must open a database edit window. This window displays the tables in a database. Initially, no database is loaded. There is a menu with File and Search. File has the options to open, close, and print tables. A user can open either a database description file or a data file. If a data file is opened, the corresponding database description is also opened. The database description window displays all the database tables. Selecting a table from this display opens the data table display in a new window. It is possible to open more than one database at a time. Opening a database table file results in the database description also being opened. It is possible to open more than one database at a time. All the windows associated with a specific database have the same color outline. Opening an already-opened table should result in the table editor window for that table being raised to the top of the screen and being visible.

## Suspension and Exit Criteria

The exit criteria for this test plan is all the high critical tests must be passed and more than 60% of the medium critical test must be passed. Otherwise, we consider the system as a failed project and the test must be repeated after debugging the problems.

## Document Overview

In chapter one, the system introduced, the scope of the project explained, the suspension and exit criteria specified, and an overview of the system provided. In chapter two, the test items and features explained. Chapter three is about the test approach and listed the test cases, the critical status of each test case, and demonstrated each test case objective. Chapter four each test case explained, the operation actions and the purpose of them provided. The expected results and the actual result can be found for each test case in this chapter. Finally, in Chapter five the test schedule is listed. The appendix is available in chapter six of this document.

## References

[1] Spec.docx

# Test Items and Features

In this test, our focus will be on editing menu operations. All the possible functions and operations must be tested. There are some common user actions that need to be tested. The functions that are provided on the menu are inserting, deleting, copying, pasting, Search, and undo. We will go through each of them and provide appropriate tests for each of these functions.

## Undo

This function reverses the last command or deletes the last entry the user typed. This function can be active by pointer or just press its shortcut (^z). This function works by storing the last changes made. By calling this function, the last changes will be overwritten on what we have.

## Copy Row

This function copies the selection from the table and stores it in the clipboard of the O.S. It can be paste in another cell or can be paste in space in another software or document. The shortcut to copy a row is (^c). The clipboard of the O.S. just can store one content. That means by copy two different objects at a different time, the second one will overwrite on the first one.

## Paste Row

This function pastes the content in the clipboard of the O.S. into the selected cell. This function can apply for multiple times. So, the users can paste the content into multiple cells without losing the content. This function can be called by using the pointer or using the (^p) shortcut.

## Insert Row

This function helps the users to add a new entity to the dataset. Technically it creates a new row in the table. This function stores the new entity by adding a new node to the XML file too. This function can be active by using a pointer or press the shortcut (^i).

## Delete Row

It deletes selected row from the table. This function removes an entity from the database. It can be called by pointer or pressing the shortcut (^d).

## Search

This function helps users to find a specific string they are looking for. This function doesn’t have any shortcut and must be called by the pointer. The GUI could provide a shortcut for this action, but it didn’t.

### Search and Replace Elements

Users can use the find function or they can use find and replace action. It helps the users to find specific data and replace the cell by another value. The ReplaceAll function is already implemented in the GUI. Thus, the users don’t need to replace the previous value by a new value one by one.

### Find Match Cases

Find Match Case option helps users to find the exact match they are looking for. This option helps them to find the result they want more accurate. It shows the cells that have the word inside them.

### Find Match Whole Words

This option helps the user to find the exact word they want on the table. This option just returns the cells that have the exact word the users searched.

# Testing Approach

In this test, we try to test the functionalities of the editing menu in the database editor software. The purpose of this testing is to check the functionality of the edit menu and find any possible scenarios that cause any error. We will use black-box testing as our approach. We do not have any access to the software codes, and this is the reason we chose black box testing. We will try different scenarios, inputs, and check the boundaries to be sure the edit menu works and all the functions inside the edit menu are working without any problem. We will categories the test cases by low, medium, and high critical condition. These labels help debuggers to address the problems based on the priority of fixing the issues.

Table 1: Test Plan

|  |  |  |
| --- | --- | --- |
| **TEST SUITE <Identifier>** | | |
| **Description of Test Suite** | **In this test suite, the focus is to test edit menu operations on the software. This test suite is included testing for inserting, deleting, copying, pasting, Search, and undo** | |
| **Test Case Identifier** | **Objective** | **Criticality** |
| Test Case 1 | **Test the functionality of UNDO** | **High** |
| Test Case 2 | **Test the functionality of COPY ROW** | **Medium** |
| Test Case 3 | **Test the functionality of PASTE ROW** | **Medium** |
| Test Case 4 | **Test the functionality of INSERT ROW** | **High** |
| Test Case 5 | **Test the functionality of DELETE ROW** | **High** |
| Test Case 6 | **Test the functionality of SEARCH** | **High** |
| Test Case 7 | **Test the functionality of REPLACE** | **Medium** |

We provide 10 different test cases in our proposed Test Suite. We labeled them based on the critical status they have. We labeled high critical those functions which are the main editing functions of a table. The medium labels belong to those that help users to work faster or improving the user-friendly interface functions.

# Test Edit Menu Functions

In this section, we provide the requirements of each test case, the steps to apply those test cases, the inputs for each test case, and the expected output. We try to cover all the possible scenarios that cause any bugs or glitch in the software. In this section, each test case documented well, and we provided details and information to repeat the test later if it is necessary or analyze the results of any cases.

## Requirements

Before starting the test, please be sure that you have these following files in a same directory.

DbEdit.jar

TEST\_DB.xml

TEST\_DB\_\_BIG\_TABLE.XML

TEST\_DB\_\_CONSTRAINT\_TABLE.XML

TEST\_DB\_\_TYPE\_TABLE.XML

DB/ TEST\_DB\_\_TYPE\_TABLE.XML

You need to be sure that Java Compiler already is installed on your machine.

## Test Case 1

**Objective:** This test checks the functionality of UNDO in the edit menu

**Notes:** In this test, Undo button will be checked, the functionality of it, and the shortcut to reach it. This test will apply this function repeatedly and after some other actions and before any actions too.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test No.: 1\_1 | | | | Current Status: Pending | | |
| Test title: Test functionality of Undo in edit menu | | | | | | |
| Testing approach: To apply this test please follow steps below.  Run the program  Click Open from File menu  Choose Test\_DB.xml from current directory  Click CONSTRAINT\_TABLE  In the new windows, from Edit menu choose Delete Row. Be sure you deleted the selected row.  From the Edit menu, choose Undo and be sure you are returning the selected row in the previous location that it was. | | | | | | |
| STEP  1  2  3  4  5  6  7 | OPERATOR ACTION  -Run the program  Click Open from File menu  -Choose *Test\_DB.xml* from current directory  Click *CONSTRAINT\_TABLE*  -In the new windows, from Edit menu choose Delete Row. Be sure you deleted the selected row. Repeat this action more than once. Use shortcut for some cases.  -From the Edit menu, choose Undo and be sure you are returning the selected row in the previous location that it was. | PURPOSE  Navigate program to reach a data base.  Simulate an action  Try to undo the action | | | EXEPCTED RESULTS  The expected result in this test is to navigate an xml file as a data base, remove an entity by calling delete function, then return the row by using undo | COMMENTS  The test went correct when user tried to return the row. The undo function buffer the actions. So, user can return as many actions as he did. But it returns the row at the end of the table, not at the place that it was. |
| Concluding Remarks:  The Undo action works correct without any problem. It uses a buffer to store all the actions. It returns changes properly. The only problem it has is it returns the deleted rows at the end of the table not at the location they were. It can cause some problem, specifically in the sorted tables. Our suggestion is to keep the location of the entity as same as it stores the actions. | | | | | | |
| Testing Team:  Ali Nouri | | | Date Completed:  11/04 | | | |

## Test Case 2 and Test Case 3

**Objective:** This test checks the functionality of Copy Row and Past in the edit menu

**Notes:** In this test case we want to check the functionality of Copy a row and Paste a Row. Our approach is to copy a row, switch the app and in another app, copy some text then, going back and paste in the table.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test No.: 2\_1 | | | | Current Status: Failed | | |
| Test title: Test functionality of Copy Row and Paste Row from edit menu | | | | | | |
| Testing approach: For this test, we need other software that runs simultaneously. Microsoft word or notepad can be a good option. We will copy a row from a table, then try to overwrite the O.S. clipboard by copy from another software and paste the content in the table. | | | | | | |
| STEP  1  2  3  4  5  6 | OPERATOR ACTION  -Run the program  Click Open from File menu  -Choose *Test\_DB.xml* from current directory  Click *CONSTRAINT\_TABLE*  -In the new windows, from Edit menu choose Copy Row.  -Use the shortcut for This action in the next test.  -Open Notepad and type a few words. Select those words and copy them.  -From the Edit menu, choose Paste and be sure you are pasting the same content you copied from the table in the previous action. | PURPOSE  Navigate program to reach a data base.  Copy a content from the table  Copy another content from another software  Paste the content in the table | | | EXEPCTED RESULTS  The expected result in this test is to paste the same content we copied from the table | COMMENTS  The test failed. It pasted the content from other program instead of the row that we already copied from the software. |
| Concluding Remarks:  The Copy Row and Paste Row function works well until the user tries to switch to other software, copy a new content and return to the software. In this situation, the content will be overwritten, and the main content will vanish. The test failed in this test case. It needs to re-write the code to fix this issue. | | | | | | |
| Testing Team:  Ali Nouri | | | Date Completed:  11/04 | | | |

## Test Case 4

**Objective:** This test checks the functionality of Insert Row in the edit menu

**Notes:** In this test, we want to check the functionality of Insert Row from the edit menu. This option helps the user to add a new row of data in the table.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test No.: 3\_1 | | | | Current Status: Pending | | |
| Test title: Test functionality of Insert Row from edit menu | | | | | | |
| Testing approach: For this test, we want to check if the program can add a new row properly or not. We will add a new row in the existing table, add a row in the empty table. We will apply these changes, save the table and load it again to be sure any changes you did, stored correctly. | | | | | | |
| STEP  1  2  3  4  5  6 | OPERATOR ACTION  -Run the program  Click Open from File menu  -Choose *Test\_DB.xml* from current directory  Click *CONSTRAINT\_TABLE*  -In the new windows, from Edit menu choose Insert Row. Be sure you already added a new row. Add some data in the row. Then, save the table  -Load the table again and be sure your data still exist there.  -This time, Use Delete option from the edit menu to delete all the rows in the table  -Use shortcut to add a new row, add some content there, then store the table. Try to load the table and check if the row is there or not | PURPOSE  Navigate program to reach a data base.  Simulate an action  Try to check your data inserted correctly  Try to add a new row in an empty table | | | EXEPCTED RESULTS  The expected result in this test is to check the inserted row and data stored correctly and accessible after reset the program. | COMMENTS  The shortcut worked as well as using the pointer to click the Insert Row in the edit menu.  Inserting a new ow in existing table and a new table accomplished without any problem |
| Concluding Remarks:  The Insert Row functionality in the edit menu works well without any problem. | | | | | | |
| Testing Team:  Ali Nouri | | | Date Completed:  11/04 | | | |

## Test Case 5

**Objective:** In this test case, we try to check the functionality of the Delete Row option in the edit menu.

**Notes:** This action is one of the essential actions to manipulate the data in the table. Users need this function to remove a whole row of data.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test No.: 5\_1 | | | | Current Status: Pending | | |
| Test title: Test functionality of Delete Row from edit menu | | | | | | |
| Testing approach: For this test, we need to check different scenarios that are possible to happen to the users. Delete from an empty table, and delete a whole row are the targets for us in this test case. | | | | | | |
| STEP  1  2  3 | OPERATOR ACTION  -Run the program  Click Open from File menu  -Choose *Test\_DB.xml* from current directory  Click *CONSTRAINT\_TABLE*  -In the new windows, from Edit menu choose Delete Row. Be sure you deleted whole table. Use shortcut of delete too. | PURPOSE  Navigate program to reach a data base.  Try to delete rows in the table | | | EXEPCTED RESULTS  The expected result in this test is to navigate an xml file as a data base, remove an entity by calling delete function | COMMENTS  The test went correct when user tried to return the row. The undo function buffer the actions. So, user can return as many actions as he did. But it returns the row at the end of the table, not at the place that it was. |
| Concluding Remarks:  The Delete action works correctly without any problem. Users can delete a row by just using the shortcut or calling the function from the edit menu. | | | | | | |
| Testing Team:  Ali Nouri | | | Date Completed:  11/04 | | | |

## Test Case 6

**Objective:** Check the functionality of search option in the edit menu

**Notes:** This Function used to find an entity from the entire table. We want to check the different scenarios in searching for an entity. Looking for an empty string, look for a string and looking for a non-ascii code are the test targets.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test No.: 6\_1 | | | | Current Status: Pending | | |
| Test title: Test functionality of Search function from edit menu | | | | | | |
| Testing approach: For this test, we want to search for different inputs. We will search for a different type of inputs, empty string, non ascii input, and a string doesn’t exist in the table. | | | | | | |
| STEP  1  2  3  4  5 | OPERATOR ACTION  -Run the program  Click Open from File menu  -Choose *Test\_DB.xml* from current directory  Click *CONSTRAINT\_TABLE*  -In the new windows, from Edit menu choose Search.  -In a new box try these values: “CORRECT”, “Bj��rk����oacute�”, “ “  -Click the Find button and repeat it to move all possible values | PURPOSE  Navigate program to reach a data base.  Simulate an action  Search for three different values  Call the function a few times to highlight all possible values | | | EXEPCTED RESULTS  The expected result in this test is to find search values in the table. It must be robust by type of the inputs. It must accept and search for ascii or non ascii values | COMMENTS  Be sure before searching for “Bj��rk����oacute�”, you already added it in the file. |
| Concluding Remarks:  The Search function works well. Users can search for any type of string. It iterates through all rows and it highlights all possible match values. | | | | | | |
| Testing Team:  Ali Nouri | | | Date Completed:  11/04 | | | |

## Test Case 7

**Objective:** In this test case, we want to check the functionality of the Replace function.

**Notes:** In this test case, you need to search for the same value to be sure all the previous values are replaced by the new one.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test No.: 7\_1 | | | | Current Status: Pending | | |
| Test title: Test functionality of Replace from Search function | | | | | | |
| Testing approach: For this test, we need to replace a value by another one in the table. We will test in a table that has a multi number of that value and we want to be sure we are replacing all of them. | | | | | | |
| STEP  1  2  3  4 | OPERATOR ACTION  -Run the program  Click Open from File menu  -Choose *Test\_DB.xml* from current directory  Click *CONSTRAINT\_TABLE*  -In the new windows, from Edit menu choose Search. In the new window, put the word “CORRECT” in front of search box, and put the word “WRONG” in front of replace box. Press button Replace ALL.  -Import the table into EXCEL, and try to search for the word “CORRECT” | PURPOSE  Navigate program to reach a data base.  Simulate an action  Try to check the correctness of Replace function in another software | | | EXEPCTED RESULTS  The expected result in this test is to replace all previous values with the new one. There shouldn’t be any previous value in the table after replacing them all. | COMMENTS  We could test the functionality of the replacement by the same software, but we preferred to use another software. In this way we can claim we test functions in this software independently. |
| Concluding Remarks:  The Replace All function worked well, and all possible values replaced with the new one. But consider this function is case sensitive. So, there is a difference between “A” and “a”. We didn’t test this aspect of the function, because we don’t have any idea that it is implemented by purpose or not. | | | | | | |
| Testing Team:  Ali Nouri | | | Date Completed:  11/04 | | | |

# Test Schedule

|  |  |  |
| --- | --- | --- |
| **Task and date** | **People** | **Description** |
| 11/ April | Alireza Nouri | Write the first edition of the test plan document. |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

# Appendix

<< possibly more readable to put the expected output here and refer to it in the previous sections. Might also provide explicit directions for analysis of output, if it’s easier to read as an appendix or if analysis is post execution. >>

$