Test Report: Expense Tracker

Radhika Sharma
Alexander Jackson
Zachary Bazen

McMaster University

Revision 1 - December 8th 2015

Contents

List of Tables

List of Figures

1. Revision History

Revision #	Revision Date	Description of Change	Author(s)
0	November 23, 2015	Black Box Test Results for getters and setters	Zachary Bazen
1	November 26, 2015	Black Box Testing of GUI	Alexander Jackson
2	November 26, 2015	White Box Testing of internal classes, Edits	Radhika Sharma
3	December 7, 2015	Black Box testing of internal classes and Edits	Radhika Sharma

Table 1: Revision History

2. General Information

The following document is a report on the various methods and their results used to test the Expense Tracker application. Both the internal and external methods are tested using black box and white box testing. Usability testing is also documented.

All testing was completed on windows computers that had Java. Tests performed on the Graphical User Interface were completed by actual trial and error methods. Tests on the internal workings of the program were completed using automated testing, which in this case was JUnit.

Usability tested was completed by having arbitrary users complete a set of actions, and then rank the Expense Tracker on a scale of 1 - 5 based on set criterion which were specified to the user.

2.1 References

a. Sample Test Report - UniNotes
b. Sample Test Report - Navigable Campus
c. Sample Test Report - Integrated Stroke
d. Test Plan - Expense Tracker

3. File Input and FileOutput

These classes read in the entries from an existing csv file and write to an existing csv file.

3.1 White Box Testing: File Input

Method : openRead(String)
Input: file path of a correct input file
Expected Output : File is loaded into system
Actual Output : Expected Output: File is loaded into system
Result : Pass

Method : openRead(String)
Input: File with correct input but not CSV
Expected Output: Error Message
Actual Output: Error Message
Result: Pass

Method: openRead(String)
Input: CSV File with incorrect input
Expected Output: Error Message
Actual Output: Error Message
Result: Pass

3.2 White Box Testing: File Output

Method: fileOutput(String, Entry Collection)
Input: File path to a CSV File
Expected Output: Entry Collection written to file
Actual Output: Expected Output: Entry Collection written to file
Result: Pass

Method: fileOutput(String, Entry Collection)
Input: File path to a non CSV File
Expected Output: Error Message
Actual Output: Error Message
Result: Pass

4. Test Results and Findings

4.1 White Box Testing: ExpenseEntryTest

Class Name: SingleEntry()

This class describes what information each expense entry contains. The main function of this class is to allow each field to be edited individually through getter and setter methods. Fields can also be modified indirectly using the set attribute method. J-Unit testing framework was used to test each method individually. The results of the test can be found below.

Fields:
1. **Date** date
2. **int** price
3. **String** project
4. **String** category
5. **String** note

Trace to Requirement: Requirement # 8: The program must allow the user to modify existing transactions

J-Unit Test Name: testPrice()
Purpose: Tests the getter and setter methods of the price field.

Method: setPrice()
Input: 2000.00
Expected Output: Entry now has price 2000.00
Actual Output: Entry now has price 2000.00
Result: Pass

Method: setPrice()

Input: -44.4
Expected Output: Entry now has price -44.40
Actual Output: Entry now has price -44.40
Result: Pass

Method: setPrice()
Input: "- -"
Expected Output: Exception Handler
Actual Output: Exception Handler
Result: Pass

Method: getPrice()
Input: Entry with price equal to 2000.00
Expected Output: 2000.00
Actual Output: 2000.00
Result: Pass

Method: getPrice()
Input: Entry with price equal to -44.4
Expected Output: -44.4
Actual Output: -44.4
Result: Pass

J-Unit Test Name: testProject()

Purpose: Tests the getter and setter methods of the project field.

Method: setProject()
Input: "Canadian Tire"
Expected Output: Entry now has project as "Canadian Tire"
Actual Output: Entry with project as "Canadian Tire"
Result: Pass

Method: setProject()
Input: "Smith Auto'
Expected Output: Entry now has project as "Smith Auto"
Actual Output: Entry with project as "Smith Auto"
Result: Pass

Method: setProject()
Input: "Empty space"
Expected Output: Entry now has project as "Default Value"
Actual Output: Entry now has project as "Default Value"
Result: Pass

Method: getProject()
Input: Entry with project as “Canadian Tire”
Expected Output: “Canadian Tire”
Actual Output: “Canadian Tire”
Result: Pass

Method: getProject()
Input: Entry with project as “Smith Auto”
Expected Output: “Smith Auto”
Actual Output: “Smith Auto”
Result: Pass

Method: setProject()
Input: Expense Entry with “Default value ” as project
Expected Output: “Default Value”
Actual Output: “Default Value”
Result: Pass

J-Unit Test Name: testCategory()

Purpose: Tests the getter and setter methods of the category field.

Method: setCategory()
Input: “Test Entry A”
Expected Output: Entry now has category as“Test Entry A”
Actual Output: Entry now has category as“Test Entry A”
Result: Pass

Method: setCategory()
Input: “Test Entry B”
Expected Output: Entry now has category as“Test Entry B”
Actual Output: Entry now has category as“Test Entry B”
Result: Pass

Method: setCategory()
Input: “Empty space”
Expected Output: Entry now has category as “Default Value”
Actual Output: Entry now has category as “Default Value”
Result: Pass

Method: getCategory()
Input: Entry with category as “Test Entry A”
Expected Output: “Test Entry A”
Actual Output: “Test Entry A’
Result: Pass

Method: getCategory()
Input: Entry with category as “Test Entry B”
Expected Output: “Test Entry B”
Actual Output: “Test Entry B”
Result: Pass

Method: getCategory()
Input: Entry with category as “Default Value”
Expected Output: “Default Value”
Actual Output: “Default Value”
Result: Pass

J-Unit Test Name: testNotes()

Purpose: Tests the getter and setter methods of the note field.

Method: setNotes()
Input: “Test A”
Expected Output: Entry now has note as “Test A ”
Actual Output: Entry now has note as “Test A ”
Result: Pass

Method: setNotes()
Input: “Test B”
Expected Output: Entry now has note as “Test B ”
Actual Output: Entry now has note as “Test B ”
Result: Pass

Method: setNotes()
Input: “Empty space”
Expected Output: Entry now has note as “Default Value”
Actual Output: Entry now has note as “Default Value”
Result: Pass

Method: getNotes()
Input: Entry with category as “Test A”
Expected Output: “Test A”
Actual Output: “Test A”’
Result: Pass

Method: getNotes()
Input: Entry with category as “Test B”
Expected Output: “Test B”
Actual Output: “Test B”

Result: Pass

Method: getNotes()
Input: Entry with note as “Default Value’
Expected Output: “Default Value”
Actual Output: “Default Value”
Result: Pass

J-Unit Test Name: testDateObject()

Purpose: The date field contains Java date type which has its own getters and setters. This test is done to ensure the date type works as expected with the model.

Method: setDate()
Input: 02/03/2014
Expected Output: Entry now has date as 02/03/2014
Actual Output: Entry now has date as 02/03/2014
Result: Pass

Method: setDate()
Input: 03/03/2014
Expected Output: Entry now has date as 03/03/2014
Actual Output: Entry now has date as 03/03/2014
Result: Pass

Method: setDate()
Input: ‘‘~’’
Expected Output: Exception Handler
Actual Output: Exception Handler
Result: Pass

Method: getDate()
Input: Entry with date set to 02/23/2014
Expected Output: 02/23/2014
Actual Output: 02/23/2014
Result: Pass

Method: getDate()
Input: Entry with note as 03/03/2014
Expected Output: 03/03/2014
Actual Output: 03/03/2014
Result: Pass

J-Unit Test Name: testAttribute()

Purpose: The methods setAttribute() and get Attribute function similar to the getter and setter methods, they allow modification of the fields indirectly. This test ensures that these methods work as expected with the model.

Trace to Requirement: Requirement # 8: The program must allow the user to modify existing transactions

Method: setAttribute()
Input: 0, 12/03/2014
Expected Output: Entry now has date set to 12/03/2014
Actual Output: Entry now has date set to 12/03/2014
Result: Pass

Method: setAttribute()
Input: 0, 03/03/2014
Expected Output: Entry with date set to 03/03/2014
Actual Output: Entry with date set to 03/03/2014
Result: Pass

Method: setAttribute()
Input: 1, "30.0"
Expected Output: Entry with price set to "30.0"
Actual Output: Entry with price set to "30.0"
Result: Pass

Method: setAttribute()
Input: 1, "-50.01"
Expected Output: Entry with price set to "-50.01"
Actual Output: Entry with price set to "-50.01"
Result: Pass

Method: setAttribute()
Input: 3, "Bob"
Expected Output: Entry with project set to "Bob"
Actual Output: Entry with project set to "Bob"
Result: Pass

Method: setAttribute()
Input: 3, "Smith Auto"
Expected Output: Entry with project set to "Smith Auto"
Actual Output: Entry with project set to "Smith Auto"
Result: Pass

Method: setAttribute()
Input: 4, "A"

Expected Output: Entry with category set to “A”
Actual Output: Entry with category set to “A”
Result: Pass

Method: setAttribute()
Input: 4, “~”
Expected Output: Entry with category set to “Default Value”
Actual Output: Entry with category set to “Default Value”
Result: Pass

Boundary Cases
Method: setAttribute()
Input: -5, “0”
Expected Output: Null
Actual Output: Null
Result: Pass

Method: setAttribute()
Input: 88, “0”
Expected Output: Null
Actual Output: Null
Result: Pass

Method: setAttribute()
Input: 0, “~”
Expected Output: Exception Handler
Actual Output: Exception Handler
Result: Pass

Method: setAttribute()
Input: 1, “~”
Expected Output: Exception Handler
Actual Output: Exception Handler
Result: Pass

Method: getAttribute()
Input: 0
Expected Output: “12/03/2014”
Actual Output: “12/03/2014”
Result: Pass

Method: getAttribute()
Input: 0
Expected Output: “03/03/2014”
Actual Output: “03/03/2014”

Result: Pass

Method: getAttribute()
Input: 1
Expected Output: “30.0”
Actual Output: “30.0”
Result: Pass

Method: getAttribute()
Input: 1
Expected Output: “-50.01”
Actual Output: “-50.01”
Result: Pass

Method: getAttribute()
Input: 3
Expected Output: “Bob”
Actual Output: “Bob”
Result: Pass

Method: getAttribute()
Input: 3
Expected Output: “Smith Auto”
Actual Output: “Smith Auto”
Result: Pass

Method: getAttribute()
Input: 4
Expected Output: “A”
Actual Output: “A”
Result: Pass

Method: getAttribute()
Input: 4 **Expected Output:** “~”
Actual Output: “~”
Result: Pass

Boundary Cases

Method: getAttribute()
Input: -5
Expected Output: null
Actual Output: null
Result: Pass

Method: getAttribute()
Input: 88
Expected Output: null
Actual Output: null
Result: Pass

Method: getAttribute()
Input: 0
Expected Output: null
Actual Output: null
Result: Pass

Method: getAttribute()
Input: 1
Expected Output: NumberFormatException
Actual Output: NumberFormatException
Result: Pass

J-Unit Test Name: testCheck()

Purpose: The method is a custom comparator. The purpose of this test is to ensure the comparator orders strings in the correct order.

Trace to Requirement(s): Requirement # 7: The program must allow the user to search on a second field

Method: getCheck()
Input: 0, "12/03/2014", Entry with date set to "12/03/2014"
Expected Output: true
Actual Output: true
Result: Pass

Method: getCheck()
Input: 1, "2000.00", Entry with price set to "2000.00"
Expected Output: true
Actual Output: true
Result: Pass

Method: getCheck()
Input: 2, "Canadian Tire", Entry with project set to "Canadian Tire"
Expected Output: true
Actual Output: true
Result: Pass

Method: getCheck()

Input: 3, “Test Entry A”, Entry with category set to “Test Entry A”
Expected Output: true
Actual Output: true
Result: Pass

Method: getCheck()
Input: 4, “Test A”, Entry with note set to “Test Entry A”
Expected Output: true
Actual Output: true
Result: Pass

Method: getCheck()
Input: 0, “12/03/2015”, Entry with date set to “03/03/2014”
Expected Output: false
Actual Output: false
Result: Pass

Method: getCheck()
Input: 1, “10.0”, Entry with price set to “2000.00”
Expected Output: false
Actual Output: false
Result: Pass

Method: getCheck()
Input: 2, “Smith Auto”, Entry with project set to “Canadian Tire”
Expected Output: false
Actual Output: false
Result: Pass

Method: getCheck()
Input: 3, “TestEntry A”, Entry with category set to “Test Entry B”
Expected Output: false
Actual Output: false
Result: Pass

Method: getCheck()
Input: 4, “Test B”, Entry with note set to “Test Entry A”
Expected Output: false
Actual Output: false
Result: Pass

Method: getCheck()
Input: 1, ”- - - -”
Expected Output: NumberFormatException
Actual Output: NumberFormatException

Result: Pass

4.2 White Box Testing: DataStore()

Class Name: DataStore()

This class is an interface between the model implementation and the rest of the program. The main functionality is to provide a getter and setter for the model. J-Unit Testing framework was used to test each method. The results of the can be found in the following text.

Fields: EntryCollection expenses

J-Unit Test Name: testExpenses()

Purpose: Tests the getter and setter methods of expenses field.

Trace to Requirement(s): Requirement # 5: The program must allow the user to view all transaction in the file

Method: setExpenses()
Input: ExpenseEntry(01/20/2015, 20001.01, "Test Project"i, "Test Category"i,"Test Note i") (i is an integer in range from 1-10)
Expected Output: Entry created with following attributes:01/20/2015, 20001.01, "Test Project"i, "Test Category"i,"Test Note i"
Actual Output: Entry created with following attributes:01/20/2015, 20001.01, "Test Project"i, "Test Category"i,"Test Note i"
Result: Pass

Method: getExpenses()
Input: ExpenseEntry with following attributes(01/20/2015, 20001.01, "Test Project"i, "Test Category"i,"Test Note"i)
Expected Output: ExpenseEntry(01/20/2015, 20001.01, "Test Project"i, "Test Category"i,"Test Note"i) (i is an integer in range from 1-10)
Actual Output: ExpenseEntry(01/20/2015, 20001.01, "Test Project"i, "Test Category"i,"Test Note"i)
(i is an integer in range from 1-10)
Result: Pass

4.3 Black Box Testing: GUI

Class Name: N/A
Fields: N/A
Trace to Requirement: Appearance Requirements: The interface of the program must be simple and intuitive for all users who have a basic knowledge of computers.

Function: File → Open

Test: Opening exising CSV file.
Input: expense_tracker/Data/expense_report.csv
Expected Output: All entries in expense_report.csv listed in table "Elements in the System." Message saying "The file loaded successfully."
Actual Output: All entries in expense_report.csv listed in table "Elements in the System." Message saying "The file loaded successfully."
Result: Pass

Test: Attempting to open text file.
Input: expense_tracker/README.txt
Expected Output: Error message saying "Incorrect file format."
Actual Output: Error message saying "Incorrect file format."
Result: Pass

Test: Attempting to open invalid file name.
Input: expense_tracker/asdf
Expected Output: Error message saying "Incorrect file format."
Actual Output: Error message saying "Incorrect file format."
Result: Pass

Function: File → Save As

Test: Saving a new CSV file.
Input: expense_tracker/Data/save_test.csv
Expected Output: Message saying "File saved successfully." File named save_test.csv saved to expense_tracker/Data/.
Actual Output: Message saying "File saved successfully." File named save_test.csv saved to expense_tracker/Data/.
Result: Pass

Test: Saving a new file without the CSV file extension.
Input: expense_tracker/Data/save_test
Expected Output: Message saying "File saved successfully." File named save_test.csv saved to expense_tracker/Data/.
Actual Output: Message saying "File saved successfully." File named save_test.csv saved to expense_tracker/Data/.
Result: Pass

Test: Attempting to save as a new file without opening a file first.
Input: N/A
Expected Output: Error message saying "Nothing to save."
Actual Output: Error message saying "Nothing to save.'
Result: Pass

Function: File → Save

Test: Saving an open file.
Input: N/A
Expected Output: Message saying "File saved successfully."
Actual Output: Message saying "File saved successfully."
Result: Pass

Test: Attempting to save without opening a file first.
Input: N/A
Expected Output: Error message saying "Nothing to save."
Actual Output: Error message saying "Nothing to save.'
Result: Pass

Function: File → Exit

Test: Exiting with a file open.
Input: N/A
Expected Output: Message saying "File saved successfully." Application closes.
Actual Output: Message saying "File saved successfully." Application closes.
Result: Pass

Test: Exiting without opening a file first.
Input: N/A
Expected Output: Application closes.
Actual Output: Application closes.
Result: Pass

Function: Edit → Add

Test: Add a new entry to an open file.
Input: 11/25/2015, 99.99, "3XA3", "Test", "Misc"
Expected Output: Message saying "Element Added." New entry displayed in table "Elements in the System".
Actual Output: Message saying "Element Added." New entry displayed in table "Elements in the System".
Result: Pass

Test: Attempt to add an entry without opening a file first.
Input: N/A
Expected Output: Error message saying "No open file."
Actual Output: Error message saying "No open file."
Result: Pass

Test: Add an entry with blank String fields.
Input: 11/26/2015, 123.45, “”, “”, “”
Expected Output: Message saying “Element Added.” New entry displayed in table “Elements in the System”. New element has “Default Value” in columns Project, Category, and Note.
Actual Output: Message saying “Element Added.” New entry displayed in table “Elements in the System”. New element has “Default Value” in columns Project, Category, and Note.
Result: Pass

Test: Add an entry with String fields containing commas.
Input: 11/26/2015, 11.11, “Sfwr, 3XA3”, “,Test”, “Misc, Notes,”
Expected Output: Message saying “Element Added.” New entry displayed in table “Elements in the System”. New element has “Sfwr 3XA3”, “Test”, “Misc Notes” in columns Project, Category, and Note, respectively.
Actual Output: Message saying “Element Added.” New entry displayed in table “Elements in the System”. New element has “Sfwr 3XA3”, “Test”, “Misc Notes” in columns Project, Category, and Note, respectively.
Result: Pass

Test: Add an entry with blank Double fields.
Input: 11/26/2015, “”, “SE 3XA3”, “Test”, “Text”
Expected Output: Message saying “Element Added.” New entry displayed in table “Elements in the System”. New element has 0.00 in column Price.
Actual Output: Message saying “Element Added.” New entry displayed in table “Elements in the System”. New element has 0.00 in column Price.
Result: Pass

Test: Attempt to add an entry with Double fields containing non-numeric characters.
Input: 11/26/2015, “123.abc”, “def”, “ghi”, “jkl”
Expected Output: Error message saying “Incorrect or missing input. Check your date and price.”
Actual Output: Error message saying “Incorrect or missing input. Check your date and price.”
Result: Pass

Test: Attempt to add an entry with blank Date fields.
Input: “”, “20.00”, “Group 9”, “Test”, “Misc Text”
Expected Output: Error message saying “Incorrect or missing input. Check your date and price.”
Actual Output: Error message saying “Incorrect or missing input. Check your date and price.”
Result: Pass

Test: Attempt to add an entry with Date fields containing an invalid date.

Input: “01/01/Foo”, “20.00”, “Group 9”, “Test”, “Misc Text”
Expected Output: Error message saying “Incorrect or missing input. Check your date and price.”
Actual Output: Error message saying “Incorrect or missing input. Check your date and price.”
Result: Pass

Function: Right Click → Modify

Test: Modify an entry in an open file.
Input: 01/13/2015, 4.24, “Group 9”, “Test”, “Misc Text”
Expected Output: Message saying “Element Added.” New entry displayed in table “Elements in the System”.
Actual Output: Message saying “Element Added.” New entry displayed in table “Elements in the System”.
Result: Pass

Test: Modify an entry to have blank String fields.
Input: 04/19/2015, 50.99, “”, “”, “”
Expected Output: Message saying “Element Added.” New entry displayed in table “Elements in the System”. New element has “Default Value” in columns Project, Category, and Note.
Actual Output: Message saying “Element Added.” New entry displayed in table “Elements in the System”. New element has “Default Value” in columns Project, Category, and Note.
Result: Pass

Test: Modify an entry to have String fields containing commas.
Input: 12/12/1995, 18.87, “Group, 9”, “,Test”, “Misc, Text,”
Expected Output: Message saying “Element Added.” New entry displayed in table “Elements in the System”. New element has “Group 9”, “Test”, “Misc Text” in columns Project, Category, and Note, respectively.
Actual Output: Message saying “Element Added.” New entry displayed in table “Elements in the System”. New element has “Group 9”, “Test”, “Misc Text” in columns Project, Category, and Note, respectively.
Result: Pass

Test: Modify an entry to have blank Double fields.
Input: 10/10/2015, “”, “Group 9”, “Test”, “Misc Text”
Expected Output: Message saying “Element Added.” New entry displayed in table “Elements in the System”. New element has 0.00 in column Price.
Actual Output: Message saying “Element Added.” New entry displayed in table “Elements in the System”. New element has 0.00 in column Price.
Result: Pass

Test: Attempt to modify an entry to have Double fields containing non-numeric characters.
Input: 11/26/2015, "one hundred", "Group 9", "Test", "Misc Notes"
Expected Output: Error message saying "Incorrect or missing input. Check your date and price."
Actual Output: Error message saying "Incorrect or missing input. Check your date and price."
Result: Pass

Test: Attempt to modify an entry to have blank Date fields.
Input: "", "44.44", "Group 9", "Test", "Misc Text"
Expected Output: Error message saying "Incorrect or missing input. Check your date and price."
Actual Output: Error message saying "Incorrect or missing input. Check your date and price."
Result: Pass

Test: Attempt to modify an entry to have Date fields containing an invalid date.
Input: "01/01/Foo", "44.44", "Group 9", "Test", "Misc Text"
Expected Output: Error message saying "Incorrect or missing input. Check your date and price."
Actual Output: Error message saying "Incorrect or missing input. Check your date and price."
Result: Pass

Function: Right Click → Delete

Test: Delete an entry from an open file.
Input: N/A
Expected Output: Message saying "Delete Complete" Entry removed from table.
Actual Output: Message saying "Delete Complete" Entry removed from table.
Result: Pass

Function: Search → One Element Search

Test: Attempt to search without opening a file first.
Input: N/A
Expected Output: Error message saying "No open file".
Actual Output: Error message saying "No open file'.
Result: Pass

Test: Search an open file by Date.
Input: Date, 03/31/2014
Expected Output: Entries with the Date 03/31/2014 listed in the table "Search Results".
Actual Output: Entries with the Date 03/31/2014 listed in the table "Search Results".

Result: Pass

Test: Attempt to search with a blank Date.
Input: Date, “”
Expected Output: Error message saying “Incorrect date input. Use the date selection”.
Actual Output: Error message saying “Incorrect date input. Use the date selection”.
Result: Pass

Test: Attempt to search with an invalid Date.
Input: Date, “asdf”
Expected Output: Error message saying “Incorrect date input. Use the date selection”.
Actual Output: Error message saying “Incorrect date input. Use the date selection”.
Result: Pass

Test: Search an open file by Price.
Input: Price, 157.89
Expected Output: Entry with the Price 157.89 listed in the table “Search Results”.
Actual Output: Entry with the Price 157.89 listed in the table “Search Results”.
Result: Pass

Test: Attempt to search with a blank Price.
Input: Price, “”
Expected Output: Error message saying “Incorrect Price format.”
Actual Output: Error message saying “Incorrect Price format.”
Result: Pass

Test: Attempt to search with a Price containing non-numeric characters.
Input: Price, “123ff4”
Expected Output: Error message saying “Incorrect Price format.”
Actual Output: Error message saying “Incorrect Price format.”
Result: Pass

Test: Search an open file by Price.
Input: Price, 157.89
Expected Output: Entry with the Price 157.89 listed in the table “Search Results”.
Actual Output: Entry with the Price 157.89 listed in the table “Search Results”.
Result: Pass

Test: Attempt to search with a blank Price.
Input: Price, “”
Expected Output: Error message saying “Incorrect Price format.”
Actual Output: Error message saying “Incorrect Price format.”
Result: Pass

Test: Attempt to search with a Price containing non-numeric characters.

Input: Price, “123ff4”
Expected Output: Error message saying “Incorrect Price format.”
Actual Output: Error message saying “Incorrect Price format.”
Result: Pass

Test: Search an open file by Project.
Input: Project, “Staples”
Expected Output: Entries with the Project “Staples” listed in the table “Search Results”.
Actual Output: Entries with the Project “Staples” listed in the table “Search Results”.
Result: Pass

Test: Attempt to search with a blank Project.
Input: Project, “”
Expected Output: All entries listed in the table “Search Results”.
Actual Output: All entries listed in the table “Search Results”.
Result: Pass

Test: Search with an incomplete Project name.
Input: Project, “Bob”
Expected Output: Entries with Projects that contain “Bob” listed in the table “Search Results”.
Actual Output: Entries with Projects that contain “Bob” listed in the table “Search Results”.
Result: Pass

Test: Search an open file by Category.
Input: Category, “A”
Expected Output: Entries with the Category “A” listed in the table “Search Results”.
Actual Output: Entries with the Category “A” listed in the table “Search Results”.
Result: Pass

Test: Attempt to search with a blank Category.
Input: Category, “”
Expected Output: All entries listed in the table “Search Results”.
Actual Output: All entries listed in the table “Search Results”.
Result: Pass

Test: Search with an incomplete Category name.
Input: Category, “D”
Expected Output: Entries with Categories that contain “D” listed in the table “Search Results”.
Actual Output: Entries with Categories that contain “D” listed in the table “Search Results”.
Result: Pass

Test: Search an open file by Note.
Input: Note, "Note 100"
Expected Output: Entries with the Note "Note 100" listed in the table "Search Results"..
Actual Output: Entries with the Note "Note 100" listed in the table "Search Results".
Result: Pass

Test: Attempt to search with a blank Note.
Input: Note, ""
Expected Output: All entries listed in the table "Search Results".
Actual Output: All entries listed in the table "Search Results".
Result: Pass

Test: Search with incomplete Note text.
Input: Note, "2"
Expected Output: Entries with Notes that contain "2" listed in the table "Search Results".
Actual Output: Entries with Notes that contain "2" listed in the table "Search Results".
Result: Pass

Test: Search with no results.
Input: Project, "xyz"
Expected Output: Message saying "Nothing Found".
Actual Output: Message saying "Nothing Found".
Result: Pass

Function: Search → Two Element Search

Test: Attempt to search without opening a file first.
Input: N/A
Expected Output: Error message saying "No open file".
Actual Output: Error message saying "No open file'.
Result: Pass

Test: Search an open file by two parameters.
Input: Date, 03/23/2014, Project "Engineer"
Expected Output: Entries with the Date 03/23/2014 and Project containing "Engineer" listed in the table "Search Results".
Actual Output: Entries with the Date 03/23/2014 and Project containing "Engineer" listed in the table "Search Results".
Result: Pass

Test: Search with no results.
Input: Project, "xyz", Date, 01/01/2015
Expected Output: Message saying "Nothing Found".
Actual Output: Message saying "Nothing Found".

Result: Pass

Function: Window → Scale Elements to Window

Test: Scale main table to window size.
Input: N/A
Expected Output: Table "Elements in the System" is scaled to window size.
Actual Output: Table "Elements in the System" is scaled to window size.
Result: Pass

Function: Window → Scale Elements to Text

Test: Scale main table to the size of its contents.
Input: N/A
Expected Output: Table "Elements in the System" is scaled to the size of its contents.
Actual Output: Table "Elements in the System" is scaled to the size of its contents.
Result: Pass

Function: Window → Scale Results to Window

Test: Scale search results table to window size.
Input: N/A
Expected Output: Table "Search Results" is scaled to window size.
Actual Output: Table "Search Results" is scaled to window size.
Result: Pass

Function: Window → Scale Results to Text

Test: Scale search results table to the size of its contents.
Input: N/A
Expected Output: Table "Search Results" is scaled to the size of its contents.
Actual Output: Table "Search Results" is scaled to the size of its contents.
Result: Pass

Function: Window → Change Text Size

Test: Increase text size.
Input: 20
Expected Output: Text size increases to 20 point font.
Actual Output: Text size increases to 20 point font.
Result: Pass

Test: Decrease text size.
Input: 10
Expected Output: Text size decreases to 10 point font.

Actual Output: Text size decreases to 10 point font.
Result: Pass

Function: Clear Results

Test: Clear the search results table.
Input: N/A
Expected Output: The table "Search Results" becomes empty.
Actual Output: The table "Search Results" becomes empty.
Result: Pass

4.4 Usability Testing

The motivation behind the implementation of the Expense Tracker application was to allow for users to have a user freindly application that would enable them to track expenses related to projects. Therefore it was essential that the application was tested among intended users. Usability tested was completed by having arbitrary users complete a set of actions, and then rank the Expense Tracker on a scale of 1 - 5 based on set criterion which were specified to the user. The following is the results from the usability testing. Ten people were asked to evaluate the Expense Tracker application. The were given no instructions or user guides explaining to the user how to user the application. Instructions on how to use the application were intentionally left out to test whether or not the application was intuitive to use. The following is the average score given for each criteria.

Look / Aesthetic : 4.2
Organization of Actions: 4.0
Usefulness of Application: 4.1
Correctness of Actions: 4.7
User Friendliness: 4.6

5. Analysis Summary

5.1 Deficiencies

Through extensive testing, deficiencies were found in the Expense Tracker. Large strings used for the fields project, category and note result in unresponsiveness from the program. Large numbers used for price result in the same issue as mentioned previously.

When more than 500 entries are stored in a batch file, this resulted in the program also becoming unresponsive. This particular bug was attributed to the use of Java Windows Builder for the Graphical User Interface.

5.2 Risks

Due to the nature of the data being handled by the application, any bugs could potentially lead to an error. Financial errors can easily become catastrophic to any company for various

reasons.

5.3 Changes

The bugs mentioned above did not alter the source data, however they did impede the effectiveness of the program. Although these bugs were embedded in the program, they were found to only be true in extreme cases. Nonetheless these issues were addressed by implementing a limit on the amount of characters that a user could enter.