Requirements Document Revision 0
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)
Revision 0	October 28, 2015	Document Outline	Zachary Bazen
Revision 1	October 30, 2015	Sections 1-4	Zachary Bazen
Revision 2	October 30, 2015	Edits	Alexander Jackson
Revision 3	November 1, 2015	Sections 4-7	Radhika Sharma
Revision 4	November 2, 2015	Sections 7-9	Alexander Jackson
Revision 5	November 3, 2015	Revisions	Zachary Bazen
Revision 6	November 4, 2015	Section 3 and Edits	Radhika Sharma
Revision 7	November 4, 2015	Edits to Sections 17-22	Alexander Jackson
Revision 8	November 5, 2015	Revisions	Radhika Sharma
Revision 9	December 4, 2015	Revisions, GUI class additions	Radhika Sharma

Table 1: Revision History

2 Introduction

The Expense Tracker is an application for small business owners who wish to keep track of expenses for various projects. The application will allow the user to load existing entries from an excel file, add an entry, search for an existing entry, delete an entry and modify an existing entry. The project goal is to allow small business owners to eliminate the need for paper records and to increase efficiency.

To ensure that the application is able to handle changes, the application was designed following the MVC model. ~~An MVC model is one where the model and its classes can be categorized as either the Model of the system, the View of the system, or the Controller of the system.~~ The letters MVC represent the words Model, View and Controller respectively. The Model of the system is the classes where the application actually manipulates the information that was given. The View of the system is the classes where the information is given as output to the user. The Controller of the system is the hardware that allows the user to interact with the application. All classes of the Expense Tracker were created such that they would fit into one of the above categories.

In addition to the MVC model, it was also imperative that the Expense Tracker implemented the software principle of information hiding. Information Hiding is where all classes and their methods are unknown to the user, and other classes however how the methods are implemented are hidden. This was important during the implementation of the expense tracker because it ensured that if any of the individual classes were changed, that the system would still work in a similar way. Information hiding allowed for flexibility in the application.

This document will explore the internal workings of the modules of the Expense Tracker and how they interact with each other to ensure that the above principles are followed. Other documents detailing the Expense tracker include the Problem Statement, Requirements Document, and Management Information System (MIS). The problem statement outlined the problem to be solved and the motivations for choosing this problem. The Reuirements document detailed the requirements that must be met by the program to solve the problem. This document will explore how the modules of the system must be deconstructed such that the program follows the basic software principles. The MIS is an extension of the Design document outlining the specific methods, state variables, and classes of the program.

3 Anticipated Changes

The following list outlines what changes are anticipated for the Expense Tracker. The Expense Tracker has been designed such that if one of the below changes occur, then minimal changes must be made. The Expense Tracker has been designed specifically so that if any of the changes below occur, the application will still meet the requirements after a minimal amount of changes.

AC1: The hardware on which the software is executed.
AC2: Format of the initial input data.
AC3: Data structure storing the entries.
AC4: Format of the output data.
AC5: User commands.
AC6: Type of data that is stored.

4 Unlikely Changes

The following is a list of changes that are not expected to occur for the application, and as such if one of the following were to be changed, it would require significant changes to the application. The unlikely changes are a result of the design decisions made to ensure that the Expense Tracker would also be simplistic in design.

UC1: Existing entries are loaded into the application.
UC2: Entries are stored in a data structure.
~~**UC3:** Type of data that is stored.~~

5 Module Hierarchy

The following is a generalized list of the module design. The module hierarchy is shown in a table below.

M1: Hardware Hiding Module
~~**M2:** Main Module~~
M2: GUI Main Module
M3: File Input Module
M4: File Output Module
~~**M5:** Exception Handling Module~~
M5: Date Parser Module
M6: Entry Collection Module
M7: Entry Comparison Module
~~**M8:** Expense Entry Module~~
M8: Single Entry Module
~~**M9:** Expense Load Module~~
M9: Data Store Module
M10: GUI Add Entry Module
M11: GUI Calendar Chooser Module
M12: GUI Text Size Module
M13: GUI Modify Entry Module
M14: GUI Search1 Module
M15: GUI Search2 Module

Table 2: Module Hierarchy

Level 1	**Level 2**
Hardware-Hiding Module	
Behavior-Hiding Module	~~Main Module~~
	~~Input Module~~
	~~Output Module~~
	GUI Main Module
	GUI Calendar Chooser Module
	GUI Add Entry Module
	GUI Change Text Size Module
	GUI Modify Entry Module
	GUI Search1 Module
	GUI Search2 Module
Software Decision Module	Entry Collection Module
	Entry Comparison Module
	~~Expense Entry Module~~
	~~Expense Load Module~~
	Data store Module
	Date Parser Module
	File Input Module
	File Output Module
	Single Module

6 Connection Between Requirements and Design

During the design process , it was imperative to ensure that both requirements and the basic principles of software engineering were met. This was implemented through the modularity of classes where each class would handle a set of requirements. This followed the software

engineering principle of information hiding.

7 Module Decomposition

7.1 Hardware Hiding Modules (M1)

7.2 GUI Main Module (M2)

- **Secret:** Control Flow of the Application
- **Services:** Allows the user to specify what action to take by clicking on buttons/menus on a GUI.
- **State Variables:**
  - serialVersionUID : long
  - expenseTracker_Main : JPanel
  - systemEntries : ET_Entry_Collection
  - searchEntries : ET_Entry_Collection
  - modelUpdate : ET_Data_Store
  - userInputFileName : String
  - windowActionChoice : int
  - testSize : int
  - savedAlready : boolean
  - openOrNot : Boolean
  - fileChooser : JFileChooser
  - tableSystemEntries : JTable
  - tableResults : JTable
  - tableHeading : String []
  - dataEntries : String[][]
  - searchResults : String [][]
  - currentEntries : DefaultTableModel
  - currentSearchResults : DefaultTableModel
  - priceFormat : DecimalFormat
  - btnClearResults : JButton
  - popupMenu : JPopupMenu
  - mntmModify : JMenuItem
  - mntmDelete : JMenuItem
  - menuBar : JMenuBar
  - mnFile : JMenu
  - mntmOpen : JMenuItem
  - mntmSave : JMenuItem
  - mntmExit1 : JMenuItem
  - mnEdit : JMenu
  - mntmAdd : JMenuItem

  - mnSearch : JMenu
  - mntmOneElementSearch : JMenuItem
  - mntmTwoElementSearch : JMenuItem
  - mnWindow : JMenu
  - mntmChangeTextSize : JMenuItem
  - mntmScaleResultsTo : JMenuItem
  - mntmScaleResultsTo1 : JMenuItem
  - mntmScaleElementsTo : JMenuItem
  - mntmScaleElementsTo1 : JMenuItem
- **Environment Variables:** Requires the use of a Computer(Mouse, Keyboard, Computer Screen) so that user can specify actions and inputs.

Table 3: Access Program

Name	Input	Output	Exception
resizeColumnWidth	JTable		
updateView	int		
addPopup	Component JPopupMenu		
save	int		
~~InputParse~~	~~String~~	~~String~~	

7.3 File Input Module (M3)

- **Secrets:** File Reading
- **Services:** Allows the user to specify a CSV file containing existing entries. The file is then read and sent to Data Store Module.
- **State Variables:**
  - notFound : int
- **Environment Variables:** Requires the use of a Computer(Mouse, Keyboard, Computer Screen) so that user can input file name of existing entries.

Table 4: Input Module

Name	Input	Output	Exception
openRead	string ET_Entry_Collection		

getNotFound	int	
setNotFound	int	

7.4 File Output Module (M4)

- **Secret:** File Writing
- **Services:** Writes the entries that the user has performed actions on back to the file that the user initially specified.
- **State Variables**
- **Environment Variables** Requires the use of a Computer(Mouse, Keyboard, Computer Screen) so that user can specify file to write to.

Table 5: Output Module

Name	Input	Output	Exception
fileOutput	string ET_Entry_Collection	int	

7.5 Date Parser Module (M5)

- **Secret:**Date Parsing from GUI
- **Services:** Parses the date that user selects from the GUI
- **State Variables:**
  - columnNo : int
- **Environment Variables**Requires the use of a Computer(Mouse, Keyboard, Computer Screen) so that user can specify the date to parse.

Table 6: Date Parser Module

Name	Input	Output	Exception
ET_Entry_Comparator	int		
compare	ET_Single_Entry ET_Single_Entry	int	

7.6 Entry Collection Module (M6)

- **Secret:** Entry Data Structure
- **Services:** Stores the entries in a data structure that the user can manipulate. Data is loaded to the data structure, the user manipulates the data in the data collection, and when the user is done the data from the data structure is written back to the file.
- **State Variables:**
  - fileEntries : Arraylist¡ET_Single_Entry¿
- **Environment Variables:**

Table 7: Entry Collection Module

Name	Input	Output	Exception
ET_Entry_Collection	ET_Single_Entry		
AddEntry	ET_Single_Entry		
GetEntry	int	ET_Single_Entry	
SetEntry	int ET_Single_Entry		
DelEntry	int		
GetSize		int	
Sort	int		
SingleSearch	int Comparable	ET_Entry_Collection	
MultiSearch	int[] Comparable[]	ET_Entry_Collection	

7.7 Entry Comparison Module (M7)

- **Secret:** Entry Comparison
- **Services:** Determines whether or not the specified column of two entries are the same.
- **State Variables:**
  - int columnNo : Stores the column number to be compared. In the Expense Tracker application, each column number refers to a criterion of an ExpenseEntry.
- **Environment Variables:**

Table 8: EntryComparator

Name	Input	Output	Exception
ET_Entry_Comparator	int		
Compare	ET_Single_Entry ET_Single_Entry	int	

7.8 Single Entry Module (M8)

- **Secret:** Expense Storage
- **Services:** Creates an Expense Entry to be stored in the Entry Collection.
- **State Variables:**
  - String date : stores the date of the ExpenseEntry
  - Double price : stores the price of the ExpenseEntry
  - String project : stores the project name ExpenseEntry
  - String category: stores the category of the ExpenseEntry
  - String notes : stores the notes of the ExpenseEntry
  - ~~int id : stores the ID of the ExpenseEntry~~
- **Environment Variables:**

Table 9: Entry Module

Name	Input	Output	Exception
ET_Single_Entry	Date Double String String String ~~int~~		
getDate		Date	
getPrice		Double	
getProject		String	
getCategory		String	
getNotes		String	

getAttribute	int	Comparable
~~getID~~		~~int~~
setDate	Date	
setPrice	Double	
setProject	String	
setCategory	String	
setNotes	String	
~~setID~~	~~int~~	
setAttribute	int Comparable	
~~toString~~		~~String~~
check	int Comparable	boolean

7.9 Data Store Module (M9)

- **Secret:** EntryCollection Population
- **Services:** Loads the information from the file into the EntryCollection.
- **State Variables:**
  - currentEntries : ET_Entry_Collection
- **Environment Variables:**

Table 10: Load Module

Name	Input	Output	Exception
getExpenses		ET_Entry_Collection	
setExpenses	ET_Entry_Collection		

7.10 GUI Add Entry Module (M10)

- **Secret:**The GUI for when the user wishes to add an entry.
- **Services:** Creates a window that takes in user input corresponding to a new entry.
- **State Variables:**
  - serialVersionUID : long
  - add : JPanel
  - addUserInput : String
  - chosenDate : Date
  - textField1 : JTextField
  - textField2 : JTextField
  - textField3 : JTextField
  - textField4 : JTextField
- **Environment Variables**Requires the use of a Computer(Mouse, Keyboard, Computer Screen) so that user can specify the fields of the entry.

Table 11: GUI Add Entry Module

Name	Input	Output	Exception
getDate		Date	
getAddUserInput		String	
GUI_Add_Entry			

7.11 Calendar Chooser Module (M11)

- **Secret:**The GUI for when the user wishes to specify a date.
- **Services:** Creates a window that allows the user to click on a date from a calendar GUI.
- **State Variables:**
  - serialVersionUID : long
  - calendarChooser : JPanel
  - dateChooser : JDateChooser
  - chosenDate : Date
- **Environment Variables**Requires the use of a Computer(Mouse, Keyboard, Computer Screen) so that user can specify the fields of the entry.

Table 12: Calendar Chooser Module

Name	Input	Output	Exception

getChoosenDate	Date
GUI_Calendar_Chooser	

7.12 Text Size Module (M12)

- **Secret:**The GUI for when the user wishes to specify a text size.
- **Services:** Creates a drop down meny that allows the user to specify a font size.
- **State Variables:**
  - serialVersionUID : long
  - textSize : String
  - changeTextSize : JPanel
- **Environment Variables**Requires the use of a Computer(Mouse, Keyboard, Computer Screen) so that user can specify the fields of the entry.

Table 13: Text Size Module

Name	Input	Output	Exception
getTextSize		String	
GUI_Change _Text_Size			

7.13 GUI Modify Entry Module (M13)

- **Secret:**The GUI to modify an entry.
- **Services:** Creates a window for the user to enter new information for an existing entry.
- **State Variables:**
  - serialVersionUID : long
  - modify : JPanel
  - modifyInput : String
  - chosenDate : Date
  - textField1 : JTextField
  - textField2 : JTextField
  - textField3 : JTextField
  - textField4 : JTextField
- **Environment Variables**Requires the use of a Computer(Mouse, Keyboard, Computer Screen) so that user can specify the fields of the entry.

Table 14: GUI Modify Entry Module

Name	Input	Output	Exception
getModifyInput		String	
GUI_Modify _Entry	Date String String String String		

7.14 GUI Search1 Module (M14)

- **Secret:**The GUI to search by 1 parameter.
- **Services:** Creates a window for the to specify a field and a search term to search the file for.
- **State Variables:**
  - serialVersionUID : long
  - search1 : JPanel
  - chosenDate : Date
  - uesrSearchFieldSelect : int
  - saerch1Input : String
  - textField : JTextField
  - btnCancel : JButton
  - buttonGroupSearch : ButtonGroup
- **Environment Variables**Requires the use of a Computer(Mouse, Keyboard, Computer Screen) so that user can specify the fields of the entry.

Table 15: GUI Search1 Module

Name	Input	Output	Exception
getSearch1Input		String	
GUI_Modify _Entry			

7.15 GUI Search2 Module (M15)

- **Secret:**The GUI to search by 2 parameters.
- **Services:** Creates a window for the user to specify 2 fields and obtains the terms on which the user wishes to search by.
- **State Variables:**

- serialVersionUID : long
- search2 : JPanel
- search2Input : String
- userSearchFieldLeftSelect : int
- userSearchFieldRightSelect : int
- chosenDate : Date
- textFieldLeft : JTextField
- textFieldRight : JTextField
- buttonGroupSearchLeft : ButtonGroup
- buttonGroupSearchRight : ButtonGroup

- **Environment Variables**Requires the use of a Computer(Mouse, Keyboard, Computer Screen) so that user can specify the fields of the entry.

Table 16: GUI Search2 Module

Name	**Input**	**Output**	**Exception**
getSearch2Input		String	
GUI_Search2 _Entry			

8 Traceability

Table 17: Traceability to Requirements

Module	**Requirements**
M2	R1,R2, R3, R4, R5, R6, R7, R8, R12, R13
M3	R1, R9, R10
M4	R2, R3, R11
M5	~~R5~~, R6, R7, R8
M6	~~R6~~, R7
M7	R6,R7
M8	~~R1~~,R8

M9	R1,
M10	R5
M11	R5
M12	R5
M13	R8, R13
M14	R6,R14
M15	R7,R15

Table 18: Traceability to Anticipated Changes

Module	**Requirements**
M2	AC5, AC2, AC6
M3	AC2
M4	AC4
M5	~~AC3~~
M6	AC3
M7	
M8	AC2, AC6
M9	
M10	AC2
M11	
M12	
M13	
M14	

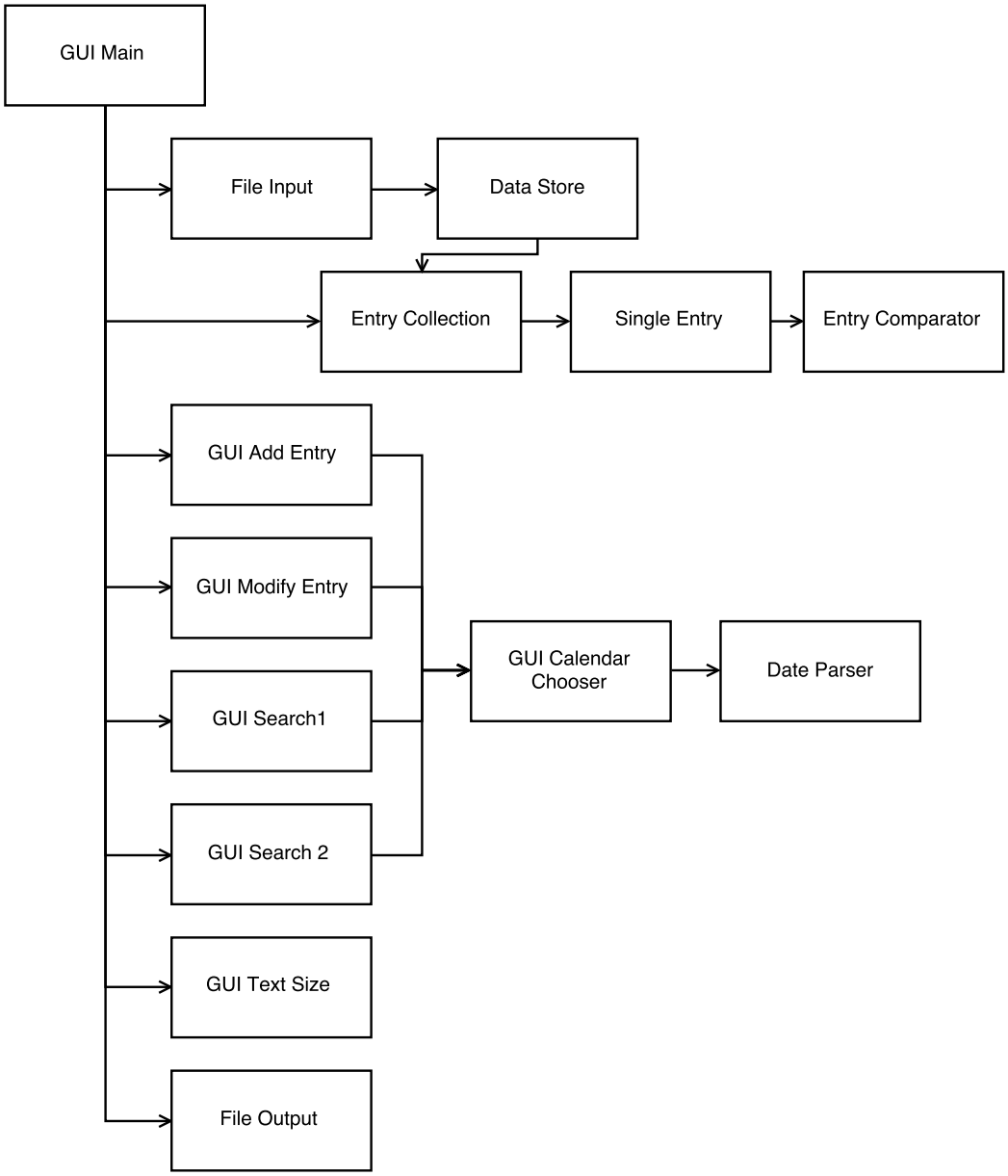
M15

Table 19: Traceability to Unlikely Changes

Module	**Requirements**
M2	UC1, UC2 ~~UC3~~
M3	UC1, UC2 ~~UC3~~
M4	UC1, UC2 ~~UC3~~
M5	UC1, UC2 ~~UC3~~
M6	~~UC3~~
M7	
M8	UC1, UC2 ~~UC3~~
M9	UC1, UC2
M10	UC1, UC2
M11	
M12	
M13	UC1, UC2
M14	UC1, UC2
M15	UC1, UC2

9 Uses Hierarchy

Figure 1: Uses Hierachy



10 Project Schedule

The gantt chart is included in the repository as a pdf **(project_schedule.pdf)** in a directory Schedule. Also screenshot images of the PERT chart are included in this directory as well **(pert_1.png, pert_2.png, pert_3.png)**.

Figure 2: Project Gantt Chart

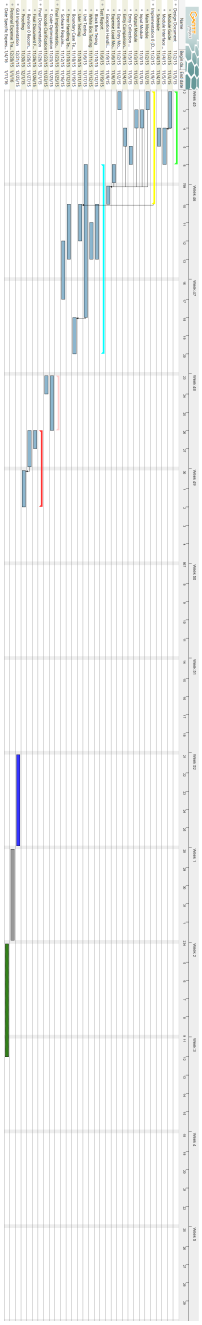


Figure 3: PERT Chart