SOFTWARE DESIGN DOCUMENT

For

Expense Tracker

Prepared By: -
Abhishek Dutta
Department of Software Engineering
Arizona State University

Academic Year: 2016 - 2017

Version: 1.0

Revision History

Date	Revision	Description	Author
11/26/16	1.0	Initial Version	Abhishek Dutta

Table of Contents

1. Introduction

1.1 Purpose

The purpose of this document is to illustrate on the architecture and system design of the Expense Tracker. It is directed towards the development team to guide and walk through the complete architecture of the application. It describes in detail about the design architecture used in this system and its components and sub components.

1.2 Scope

Expense Tracker basically is maintaining the user expense information with details as such category, date of entry and the amount spent on the category. A user can go back and update an existing entry or delete an existing entry. Users can also add/delete different members or check other user's expenses. The application takes input using command line console and accepts only a valid entry for an existing user.

The project is mainly designed for the use of students, who want to track their monthly expenses or a family person who wants to keep track of the expenses made by the family members. It can be also useful to a small group of people or a small organization/association who wants to track their expenses made from a common fund. The user interface is a simple command line interface and all the options are clear to understand for a person. The programming language used for building this application is Java and data is stored using Json.

1.3 Definitions, Acronyms, and Abbreviations

ANT	Java library.
IDE	Integrated Development Environment
JAVA	Platform Independent Programming Language
JSON	JavaScript Object Notation
JRE	Java Runtime Environment
SRS	Software Requirements Specification

Table 1. Acronyms and Terminology

1.4 References

- Books
  - Software Engineering: Ninth Edition by Ian Sommerville
  - The Personal Software Process(PSPSM), November 2000 by Watts S. Humphrey

1.5 Overview

Rest of the document is mainly focused on the architecture and design of the system using block diagrams and pictorial representations.

2. System Architecture Design

2.1 Overview of modules/components

The Expense Tracker application uses limited number of subsystems and components. For interacting with users and doing the I/O operations, it uses system command line interfaces. It uses different modules to add/delete/update expense. Similarly, it uses different modules for add/delete Member and display expenses and members. Application uses JSON to store the data which is the backend part. Separate modules of code are reading and writing from the JSON.

2.2 Structure and Relationship

The structure of this application is like a two-layer architecture with most of the components having one to one relationship. Please refer the use case diagram below for understanding: -

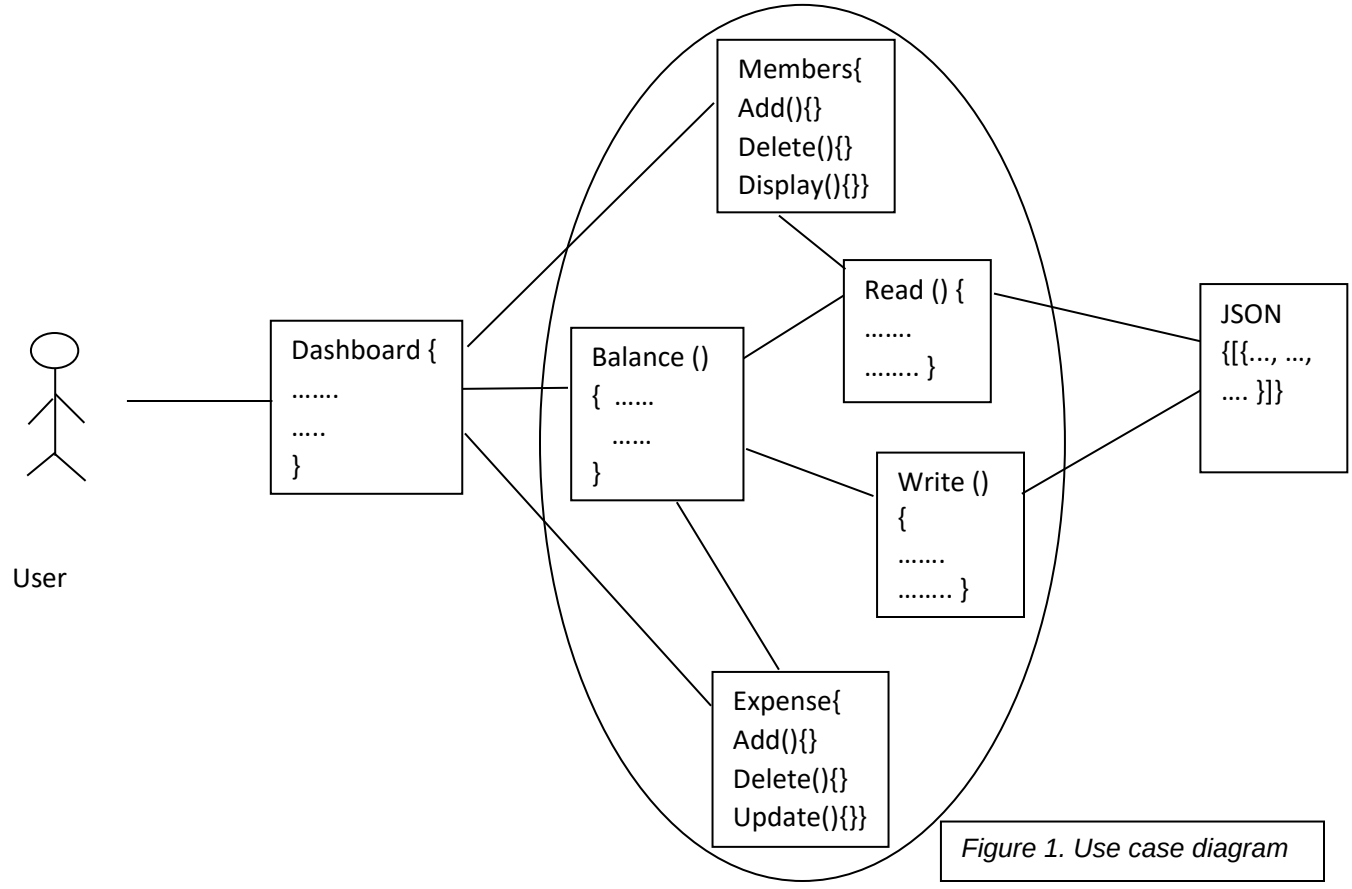

Figure 1. Use case diagram

As per the use case diagram in fig. 1, user interacts with the dashboard class which further internally connects to the other classes which will connect to read/write operation. That will further connect to JSON and flow will be back to user through the same path in the opposite direction.

3. DATA DESIGN

3.1 Data Description

The data in the application is accepted in the form of “Name”, “Category”, “Amount” and “Date” fields from the user. The output data is also provided to the user in these fields itself along with some information messages that user follows. All these data are String types except the “Date” which is date type. A “name” can have multiple entries containing “Category”,” Amount” and” Date”. There can be multiple such users. Data is stored as an object of array of objects format in the JSON file. Whenever a data is fetched from backend or forwarded to backend, it is done as an object.

4. Detailed Description of Components

This section describes about the classes and methods used in the application and their functionality.

1)Dashboard – This class contains the main method that provides a list of options to choose for user and will continuously check for users for any input.

2)Members – This class contains methods such as Add_Members() to add new members to the application. Remove_Members() to remove an existing member from the application and Display_Members() to display the list of members present in the application.

3)Expense – This class contains methods such as Add_Expense() to add new expenses to an existing user, Delete_Expense() to delete an existing expense for an existing user and Update_Expense() to update and existing expense of an existing user.

4)Balance – This class is used to display the balance of any existing user in the application by using Display_Balance() method.

5)Read_File – It is used to read the data from the JSON file. It uses read() method.

6)Write_File – It is used to write the data to the JSON file. It uses write() method.

5. Human Interface Design

5.1 Overview of User Interface

A user can perform all such operations listed in the fig. 2 below. It's a command line user interface where user needs to select the option number designated to each of the options. A user will be able to add/remove a member, list all the members, add/delete/update an expense for a user, list all the expenses made by the user.

5.2 Screen Images

The screenshot below is an instant from the command line user interface. User can select any option they want.

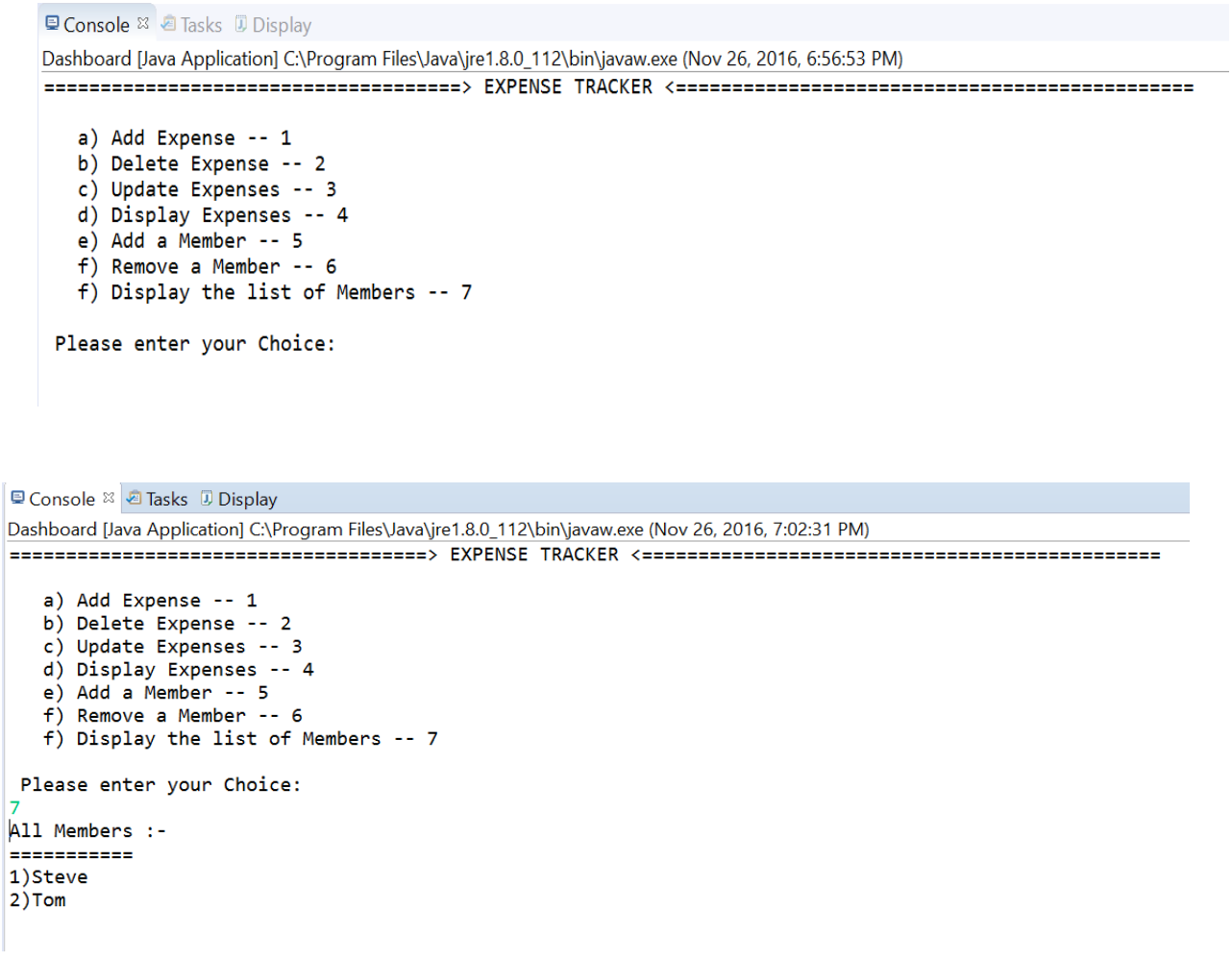

Figure 2. Console Output

6. Requirements Metrics.

The below table describes on which components satisfies the requirements specified in the SRS document.

Requirements	*Components*
Add a Member	*Members*
Display Members	*Members*
Add expense for a valid member	*Expense*
Add expense for an invalid username	*Expense*
Add expense for a valid user with an invalid entry	*Expense*
Update expense for a valid user	*Expense*
Update expense for an invalid user	*Expense*
Update expense for a valid user with an invalid entry.	*Expense*
Display the expenses	*Balance*
Delete expenses for a valid user	*Expense*
Delete expenses for an invalid user	*Expense*
Delete expenses for a valid user with invalid entry	*Expense*
Delete a member for a valid user	*Member*
Delete a member for an invalid user	*Member*

Table 2. Requirements Metrics

5. Bibliography

1) Software Engineering: Ninth Edition by Ian Sommerville

2) The Personal Software Process(PSPSM), November 2000 by Watts S. Humphrey