Software Requirements Specification

Vincent Boivin, ID 27419694 Kisife Giles, ID 40001926
Marc-Antoine Jette-Leger, ID 27895038
Jean-Loup Johnston, ID 40006259
Fabian Vergara, ID 40006706 Michael Xu, ID 27206356

March 3, 2018

Software Requirements Specification
Prepared by Group PA-PJ

Contents

1 Revision History

Tuesday February 6th, 2018 Version 1.0

Tuesday February 7th, 2018 Version 1.1 Reworked context, added Business rules, Non-functional requirements

Tuesday February 9th, 2018 Version 1.2 Described Use Case 3, added Domain Model, added the domain model entries to the glossary, formatted tables and lists

Saturday March 3rd, 2018 Version 1.3 Described Use Case 4, Use Case 5 and the MVC model.

2 Project Description

2.1 Purpose

Our team has been given a mandate to design and implement a money budgeting application for use by the clients of our customer. The application aims to provide a representation of a user's current spending based on type and to offer him/her ways in which that spending can be changed. The purpose of this document is to provide the Software Requirement Specification (SRS) report for the the said application. The purpose of it is to give a high level overview and a full description of the functional and nonfunctional requirements of the system. It will also cover the system design and implementation constraints, and different external interfaces with which the system shall interface.

2.2 Introduction

Currently, many users have trouble discerning the way in which they spend their money unless they track it themselves, however, that task is arduous and time consuming. In order to offer customers with a clear representation of their finances, companies have developed complicated software to solve this task, however, often times, users hesitate to use it due to the aforementioned complexity. As a solution, we have been tasked with presenting customers with a simple, yet efficient and intuitive application.

2.3 Context

In the context of our application, the main way users will be able to interact with our application will be mainly through a desktop application. The application will use user provided user information (bank statement) in order to provide an assessment of the current spending.

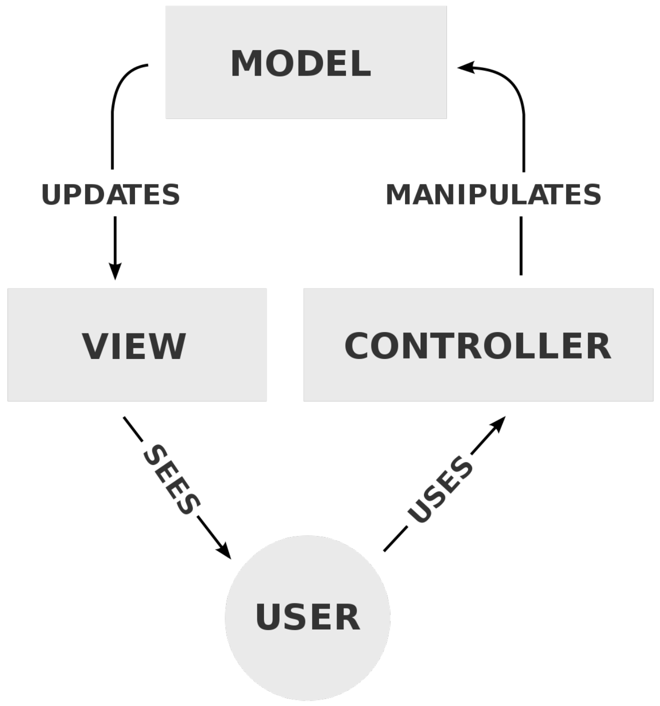
2.4 Business goals

The primary business goal is to make our customers save more and to make them adopt our software applications, which will allow us to provide a better service as a more well rounded financial company. This in turn allows us to increase efficiency (needs less employees to serve customers on the same topic), increase our market share, as we did not have this service beforehand which will in turn increase our profit margin because we have access to more customers than before.

2.5 Scope

The MyMoneyApp software is a financial management application, developed in Java as a standalone desktop application targeting young consumers. This system's aim is to help users to make wise and accurate decisions when they have a target amount of savings they want to have for a month. It performs this operation by first getting the user's bank statement, then displaying them on-screen to differentiate what amount of money has went into what type of service, and to then be able to change those amounts in the coming months. The main qualities of MyMoneyApp, is that it is easy-of-use, user-friendly and efficient, which will allow the purchasing company to satisfy its customers' needs. All in all, this software will help the company to hold its market leader position in the financial domain.

2.6 Model-View-Controller(MVC) model



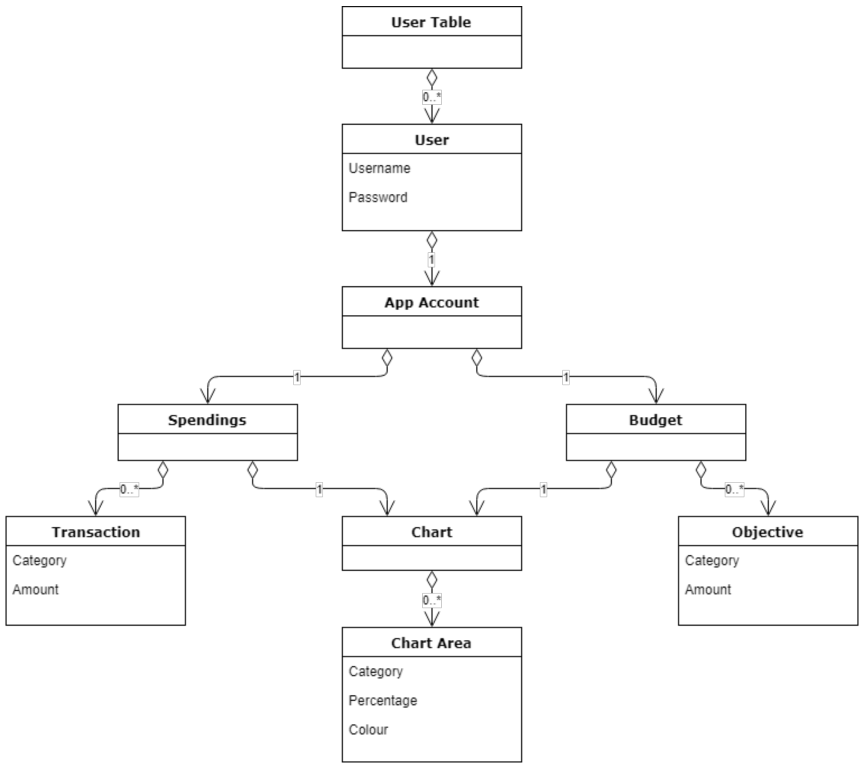
The MVC pattern is commonly used in computer applications which requires interaction with the user. There is three components to the model-view-controller architectural pattern.

First, the model is the backbone of the application. The model contains the functions and the data used by the application.

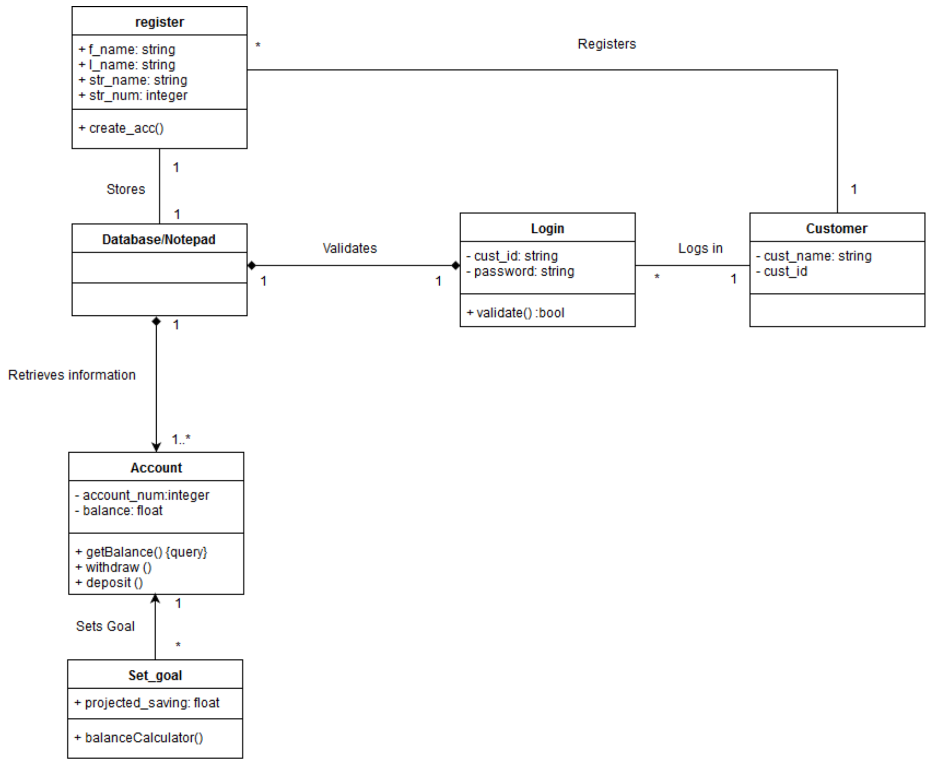
Secondly, the view is the component which will be shown to the user. In our case, the view is a graphical user interface(GUI) which will create interactions with the user through various buttons.

Finally, by clicking buttons the user will use the controller. The controller is the component connecting the view to the model and the model to the view. By pressing a button, the user will update the model through the controller and the update will be shown through the view. The most important aspect of the MVC pattern is the separation of the source code in three different components, thus enabling three engineers to work on the code at the same time.

2.7 Domain model



2.8 Class Diagram



2.9 Actors

This document is intended to be read by:

- Users of the software: this document allows them to have a more complete idea about the system and its functionality.
- Team developers: they can use this document as a primary resource for all subsequent project development phases (design, coding, testing and maintenance phases).
- Testers: To be able to test the system in accordance to the specified requirements.
- CEO of the company that has hired us: this document will allow them to deeper understand and have a more comprehensive idea about the requirements of the project.

3 Functional requirements

3.1 Overview

This section includes all the details regarding the use cases and features afforded to the user of the MyMoney application. Those features include creating and logging into an account, accessing various information about the transactions made with a bank account, and creating, inspecting and deleting budgetary goals to be stored in a list.

3.2 Use cases

3.2.1 Create user accounts

Action	Account management
Case ID	1.1
Summary	User provides the necessary information for the creation of an account.
Scope	Budget management application
Trigger	Registration button
Precondition	None
Postcondition	Account is created
Primary Actor	User
Secondary Actor	Filesystem

Main Scenario-Step	**Action**
1	User Clicks on Register.
2	User enters a username, password, and password confirmation, and click “Register Account” button.
3	System verifies if the username is already taken.
4	System gets login credentials from File System.
5	System verifies in login file if the password and password confirmation are the same.
6	System creates account object.
7	System saves the username and password combination to the database/textpad.
8	System displays login menu, with account creation confirmation, and asks user to login.
9	System goes idle.

3.2.2 Access user accounts

Action	Account management
Case ID	1.2
Summary	User provides the necessary information to login.
Scope	Budget management application
Trigger	Login button
Precondition	Have a registered account
Postcondition	Account is accessed
Primary Actor	User
Secondary Actor	Filesystem

Main Scenario-Step	**Action**
1	User inputs username and password.
2	User clicks on the “login” button.
3	System verifies if the username, password pair exists.
4	System displays the user’s transaction logs.
5	System goes idle.

3.2.3 Load and display transactions data

Action	Load and display transaction data
Case ID	2
Summary	User provides the the number of his credit card number for the reviewing of his spending.
Scope	Budget management application
Trigger	Get my info button
Precondition	To be logged in as a user
Post condition	Transaction data is displayed
Primary Actor	User
Secondary Actor	Filesystem

Main Scenario-Step	**Action**
1	User Enters his credit card number.
2	User clicks on the get transaction data button.
3	System pulls the information from the textpad/database.
4	Transaction data is displayed on the screen.
5	System goes idle.

3.2.4 Create and visualize budget

Action	Account customization
Case ID	3
Summary	The user is able to create a budget and visualize it with a chart
Scope	Budget management application
Trigger	Budget button
Precondition	To be logged in as a user
Postcondition	Budget information displayed and editable, chart is accessible
Primary Actor	User

Main Scenario-Step	**Action**
1	User clicks the Budget button
2	User chooses a category
3	User enters an amount
4	User clicks the Add to budget button
5	User's budget is updated in real time
6	User clicks Create chart button
7	User's budget is instantly computed and illustrated by a chart in a new window
8	User clicks the Reset budget button
9	User's budget values are all set to 0
10	User's budget is updated in real time

3.2.5 Automate the spendings in the budgeting section

Action	Automation of spendings
Case ID	4
Summary	Once the user logs in, he is able to look at his spendings and use it to display charts
Scope	Budget management application
Trigger	Budget button
Precondition	To be logged in as a user
Postcondition	Budget information displayed and editable, chart is accessible
Primary Actor	User

Main Scenario-Step	**Action**
1	User logs in
2	User chooses the budgeting window
3	User create charts with the automated spendings

3.2.6 Give advice based on the budgeting section

Action	Advice giving from budgeting section
Case ID	5
Summary	After the use of the budgeting section, the user is able to receive advice on his spendings.
Scope	Budget management application
Trigger	Budget button
Precondition	To be logged in as a user
Postcondition	Budget information displayed and editable, chart is accessible
Primary Actor	User

Main Scenario-Step	**Action**
1	User logs in
2	User chooses the budgeting window
3	User creates charts with the automated spendings
4	User press yes when he is given the choice for financial advice
5	User financial advice appears on the screen

3.3 Business Rules

- The customer must not be able to alter his balance that he entered (read only)
- The customer must have a credit card with a balance on it (textpad in this case)

4 Non-functional requirements

- The login information (password/username) is encrypted
- The application is intuitive to use and requires no computer knowledge

5 Design Constraints

The programming language used in this software is Java. The main feature is giving a clear representation of the user's spending over the course of a bank statement. The representation will take the form of graphs and charts defining where their money went. Users can only view their past data and cannot alter it. The maintenance and feature upgrades are handled by us, the developers of MyMoneyApp.

6 Glossary

User table The user table is the collection of registered users, along with their login information

User A user is the virtual representation of a person using the application

App account An account is the model used by the application to store and use a user's information and interact with all the different features present in the application

Spending A user's spending is the agglomeration of every transaction done and entered by the user

Budget A user's budget is the collection of the user's projected spending

Chart A chart is the graphic illustration of the user's spending or budget information

Chart area A chart area represents a category of the user's projected or actual spending, represented by a percentage of the total spending or budget and unique color in the chart, indicating which category the area relates to

Transaction A transaction summarizes a positive or negative money transaction made by the user in real life and is composed of the amount of money exchanged during the transaction and the category which the transaction relates to (e.g. food, home, transportation, salary, ...)

Objective An objective is an entry reflecting a projected amount spent in a given category. It is composed of a category and an amount. These values are used by the system to produce a budget for the user, illustrated by a chart if the user so chooses)

7 References

Larman, C. (2016). Applying UML and patterns. Chennai: Pearson.
Sommerville, I. and Sommerville, I. (2006). Software engineering 7.5. Harlow: Addison-Wesley.