"My Money" Software Requirements Specification

Team PB-PK

11 February 2018

Member Name	Member ID
Noemi Lemonnier	40001085
Genevieve Plante-Brisebois	40003112
Han Gao	40053734
Theo Grimond	27276044
Real Nguyen	27566263
Ornela Bregu	26898580
William Prioriello	27080956
Tiantian Ji	27781083
Dong-Son Nguyen-Huu	40014054
Ashesh Patel	40018519
Sabrina Rieck	40032864

Table 1: Team PB-PK

Revision History

Date	Version	Description	Author
21/01/2018	1.0	Document Start	Ornela Bregu
28/01/2018	1.1	Added comments describing what needs to be done for each section.	Ornela Bregu
30/01/2018	1.2	Added the document purpose, business goals and Use Case Diagram.	Ornela Bregu
04/02/2018	1.3	Added the remaining use cases. Fixed formatting. Added figure descriptions for the domain model and the system overview	Dong-Son Nguyen-Huu
07/02/2018	1.4	Edited introduction, use goals and added Functional Requirements.	Ornela Bregu
08/02/2018	1.5	Added domain model diagram.	Ornela Bregu
08/02/2018	1.6	Added Data Dictionary and References.	Ashesh Patel
09/02/2018	1.7	Added Non-Functional Requirements and fixed formatting.	Ornela Bregu
11/02/2018	1.8	Final quality checking for content and formatting.	Real Nguyen

Table 2: Revision History

List of Figures

List of Tables

Contents

1 Introduction

The purpose of this document is to define the user requirements for the desktop application "My Money" and to ensure that we deliver what the user really needs by mitigating the risks as much as possible. We start by specifying the user goals and the reasons why we are developing this application. Then, we continue explaining the domain concepts, their meaning in the context of our application and an overview of the program we are planning to build. In the last part, we define functional and non-functional requirements for this project.
The intended audience of this document is described in the table below.

Group of readers	**Reasons for reading**
Users	To give feedback about the requirements
System developers	To understand what functions and properties the system must contain
Testers	To test the system against the requirements
Project organization team	To follow-up the status of the project against the requirements
Project coordinator	To follow-up the status of the project

Table 3: Intended Audience

2 User Goals

This section describes the reasons why we are developing the "My Money" application.

The application is a tool to help any user manage their finances, keep track of their cash spending, and discover where they are

spending the most money to then make necessary adjustments. This desktop application allows the user to quickly refer to their accounts, modify their budget categories, and input new transactions.

3 System Overview

The use of "My Money" revolves around the user, who can use the application to update their cash spending by listing the transactions that they have entered and specify which categories they fall into; manage their available cards, see the type and balance of each card, and add or remove cards; or set a budget, which can either automatically allocate resources among various categories or allow the user to set them manually.

4 Domain Concepts

The use of "My Money" starts with the user authenticating their identity by entering a username and a password. If the information entered is correct, the user can access the application. From there, the user can either track their income and expenses incurred through transactions, or calculate their budget according to their available income. The transactions are then associated with their respective cards. Cards may be either credit or debit, and have a balance and a card number. Transactions are also used for cash spending to be used with the budget set by the user. Alternatively, the budget can be paid for through the use of the user's available

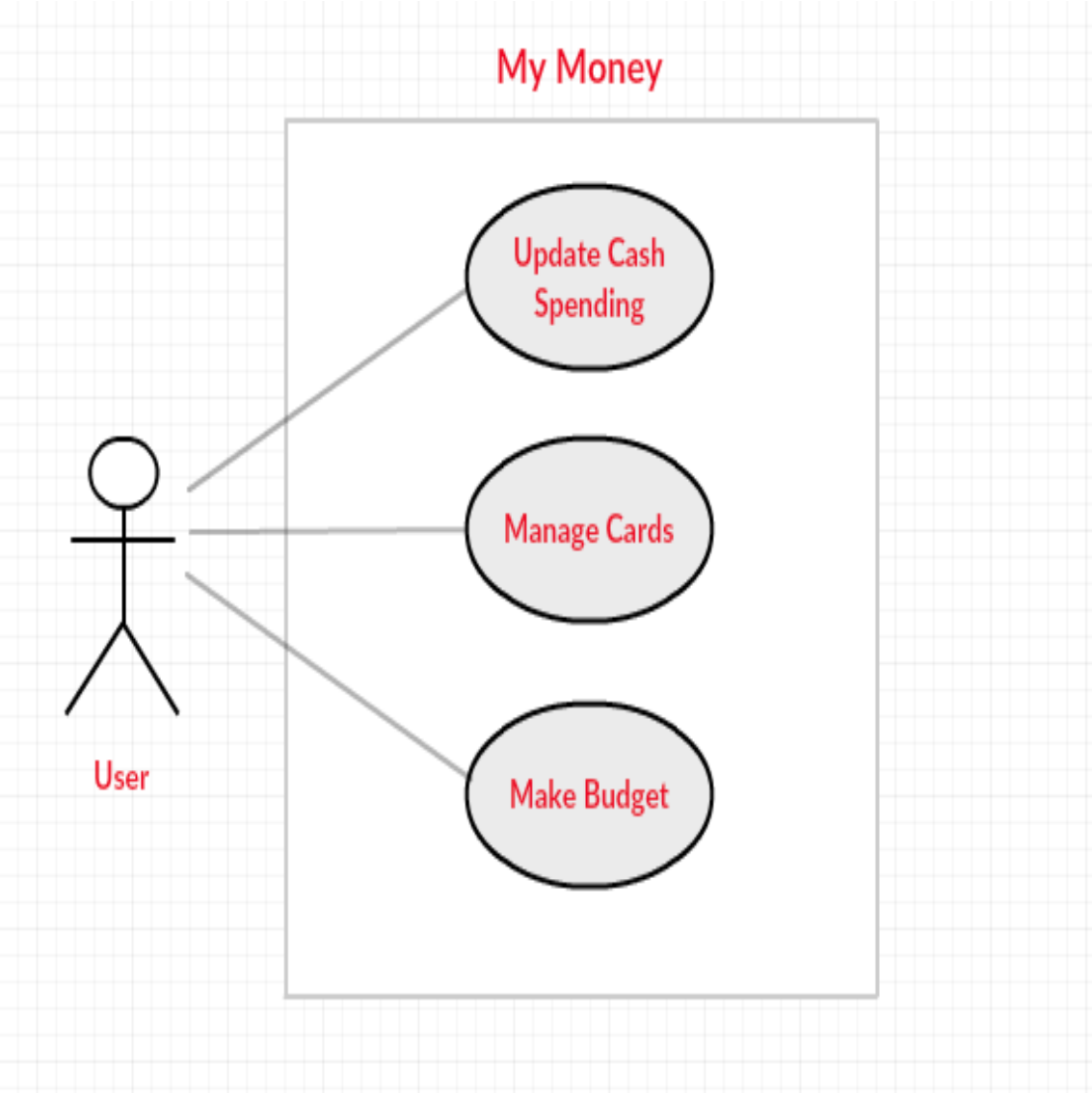

Figure 1: Use Case Diagram

cards.

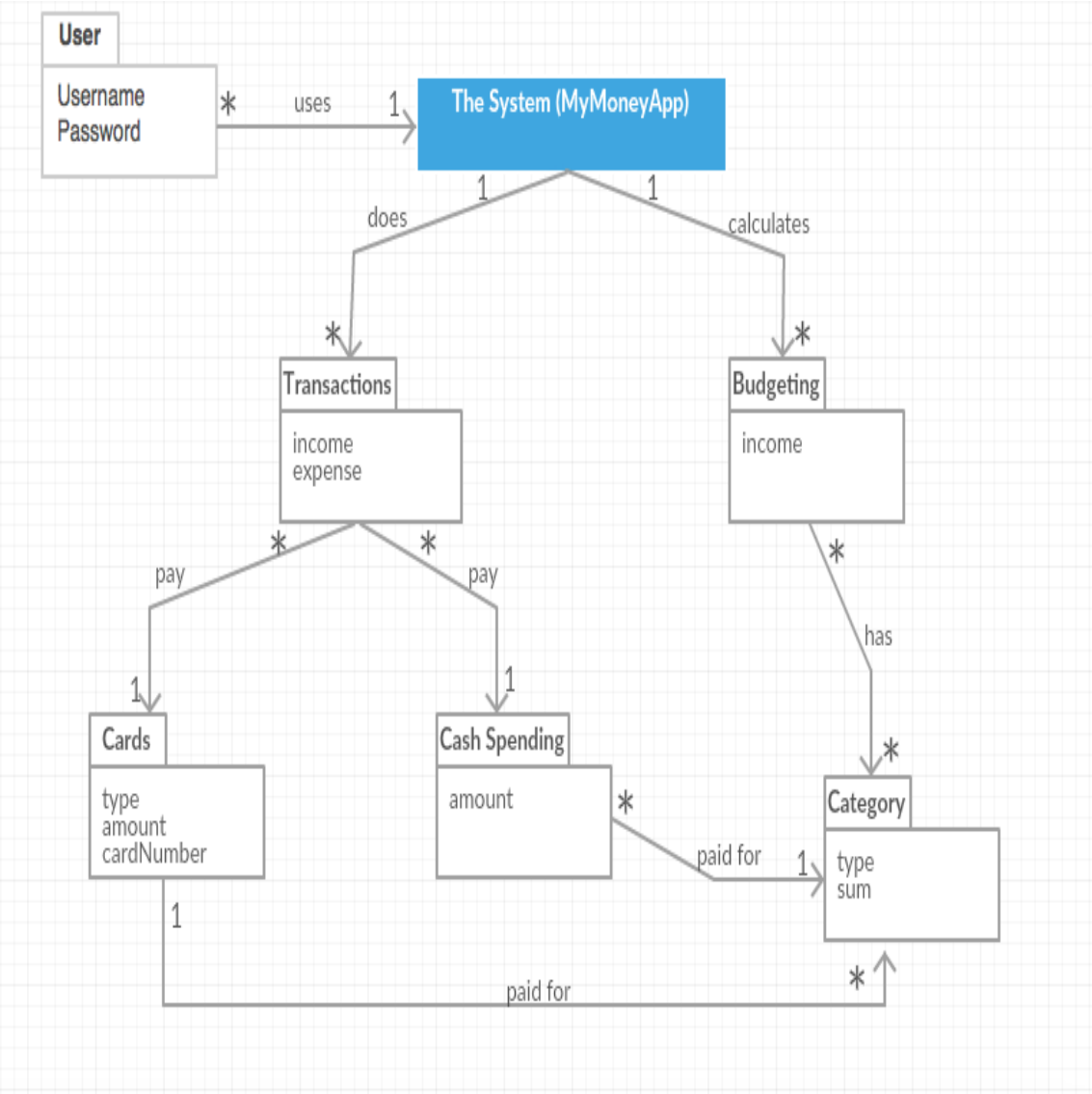

Figure 2: Domain Model Diagram

5 Functional Requirements

The Functional Requirements section details the capabilities and functions that the "My Money" application must be capable of performing. These requirements will assure that the application will correctly and reliably perform its intended functions. This section will provide general, as well as specific requirements to be used in the design, testing, and validation of the application. The focus is on what the program must do. Details on how the components will be developed and how the program will operate will be defined later in the Software Design Specification. Functional Requirements in this document are organized by use case. There are three basic use cases:

1. Manage Cards
2. Update Cash Spending
3. Budgeting

As our project follows an Agile approach to Unified Process, we are planning to improve the basic use cases with other important features on other iterations. The following subsection will describe all the basic use cases.

5.1 Basic Use Cases

This subsection describes the three main use cases of the application. They are described using the casual degree of formality.

Manage Cards

Use Case ID	CardManagementA
Use Case Name	Manage Cards
Summary	Allow a user to view the cards associated with their profile, including viewing the cards' type and balance. The user can add cards and specify the card type.
Actors	Any private individual
Pre-Condition(s)	User has installed "My Money" on their system
Main Flow	User wants to view, remove, or add cards: 1. User wants to view active cards associated with their account 2. User launches application 3. User views the details of the cards attached to their account, including their type and balance 4. User can add or remove cards
Alternate Flow(s)	1. User adds a card and choose its type and starting balance 2. User removes an existing card from their profile
Exception Flow(s)	User inputs invalid value (e.g. non-numeric or negative value) as the starting balance of an added card: 1. Application will prompt user to enter a non-negative numeric value

Cash Spending

Use Case ID	CashSpendingA
Use Case Name	Update Cash Spending
Goal/Purpose	Allow a user to manage their cash spending on different categories
Actors	Any private individual
Pre-Conditions	User has installed "My Money" desktop application
Post-Conditions	Cash expenses are updated
Main Flow	1. User wants to update their expenses 2. User starts application 3. User enters amount they spent on any selected category 4. Application stores value and in the selected category of expenses
Alternate Flow(s)	At any time, Application fails: - User restarts Application.
Exception Flow(s)	User inputs invalid value (e.g. non-numeric or negative value) as the expense: 1. Application will prompt user to enter a non-negative numeric value

Budgeting

Use Case ID	BudgetManagementA
Use Case Name	Budgeting
Goal/Purpose	Allow a user to manage their budget
Actors	Any private individual
Pre-Conditions	User has installed "My Money" on their system
Post-Conditions	Available amount is calculated and displayed for each category
Main Flow	User wants details on the usage of their funds: 1. User opens the "My Money" application and presses on "Budgeting" to calculate their budget 2. The user enters their available funds 3. The application returns the amount for each category, calculated according to the default recommended percentages and the input
Alternate Flow(s)	At any time, Application fails: - User restarts Application.
Exception Flow(s)	User inputs invalid value (e.g. non-numeric or negative value) as the expense: 1. Application will prompt user to enter a non-negative numeric value

5.2 Other Use Cases

For the following iterations, we are planning to make the whole application more user friendly. The user interface of the application will be more attractive, how the data is presented will be more interesting by giving the user more options, etc. Listed below are added functionalities that will be added in future iterations. Please note that as we are following an Agile methodology, the use cases below are subject to change or may require more information before adding details.

Update Card-Payments

1. User wants to link their expenses with the card they used for the payment.
2. User opens the application and goes to the cash spending feature.
3. User enters the amount spent and selects the card they used for the payment.
4. Application validates and records the amount to a user-specified card and updates the total amount spent for that card.

Control Expenses

1. User wants to link their expenses with the budget and get notified if they go over budget.
2. User opens the application and goes to the cash spending feature.

3. User enters the amount spent and other related information.
4. System validates and records the amount to specified category, checks the planned budget and generates a notification with amount left to spend or a budget warning.

Control Budgeting

1. The user wants to change the percentages of each category in their budget.
2. User opens the application and goes to the budgeting feature.
3. User changes the percentage of each category.
4. System validates and records each percentage to each specified category.
5. System updates the budget.

Authenticate User

User wants to authenticate themselves with a username and a password.

Manage Budget

User wants to add/delete categories and put a fixed amount for any of them.

Real-time Transactions

User wants to link the application with real-time transactions.

6 Non-Functional Requirements

In this section, we describe the quality attributes and characteristics of the software. Most of the quality attributes come from the FURPS+ model.

ID	Vers	Description	Priority	Traces to use cases
1	1	Only registered users are allowed to use the application.	Must-have	Authenticate Users
2	1	The source code shall be self-documented by placing the design description in a Javadoc-readable method header.	Must-have	All use cases
3	1	If the input data format changes, the developer will be able to make the required changes in less then 24 person-hours.	Must-have	All use cases
4	1	There can be no unhandled exceptions from incorrect user input.	Must-have	All use cases
5	1	All menus must have a consistent format.	Must-have	All use cases
6	1	A user can install and operate the program without assistance of any kind.	Must-have	All use cases
7	1	During a system restart, the system will return to a functioning state.	Must-have	All use cases
8	1	90% of novice users can learn to operate major use cases without outside assistance.	Must-have	All use cases
9	1	The main use cases must be accessible from the top screen.	Must-have	All use cases
10	1	The program should be available 24/7.	Must-have	All use cases
11	1	The finished software must support new users without needing to be rewritten or recompiled.	Must-have	All use cases

Table 4: Non-Functional Requirements

7 Data Dictionary

Term	Definition
User	A person who uses the program for their own personal use
The System	The application
Transactions	Cash flow (income and expense)
Cards	Virtual currency object linked to bank account
Cash Spending	Expenditure in cash
Category	Predefined budget options
FURPS+	Model for classifying software quality attributes
Domain Model	Depiction of relationship between entities (person, concept, object or event)
Use Case Diagram	Depiction of association between actor and use cases

Table 5: Data Dictionary

8 References

Leave Debt Behind. (2010). 10 Recommended Category Percentages for Your Family Budget. [online] Available at: http://www.leavedebtbehind
living/budgeting/10-recommended-category-percentages-for-your-family-budget/ [Accessed 9 Feb. 2018].