Mobile Open Source SW Utilization

SRS

Money Planner

(Group D)

32141631 Juhwan Moon

32154019 Hansol Jang

32170854 Yujin Kim

32174035 Soyeon Jung

32179181 Davi Chalita Meneguelli

INDEX

1. Introduction
    - 1.1 Purpose
    - 1.2 Document Convention
    - 1.3 Intend Audience and Reading Suggestion
    - 1.4 Project Scope
    - 1.5 References

2. Overall Description
    - 2.1 Product feature
    - 2.2 User classes and characteristic
    - 2.3 Operating environment
    - 2.4 Assumption and dependencies

3. Usecase

4. System feature
    - 4.1 Description and Priority
    - 4.2 Expected requirement

5. External Interface Requirement
    - 5.1 User Interface
    - 5.2 Hardware Interface
    - 5.3 Software Interface
    - 5.4 Communications Interface

6. Activity Diagram

7. Functional/ Non-functional Requirements
    - 7.1 Functional Requirements
    - 7.2 Non-functional Requirements

8. References

1. Introduction

1.1 Purpose

The purpose of this document is to present a detailed description of the Money Manager Application. It will explain the purpose and features of the application, the interfaces of the application, what the application will do, the constraints under which it must operate and how the constraints will react to external input.

1.2 Document Convention

This document follows the standards of IEEE.

1.3 Intend Audience and Reading Suggestion

This report is for people who develop this application and stakeholders. This SRS's order is Introduction, Overall Description, Usecase, System Feature, External Interface Requirement, Activity Diagram, Functional/Non-functional requirements, Class Diagram, Sequence Diagram and References.

If you want to know about this application feature, then read second paragraph, Overall Description. If you want to know about this application's structure, then read third paragraph, Usecase.

1.4 Project Scope

Our goal is that people be alert about money. Using money manager, we will help those who are not sensitive about money flow and want to know about it. The application will separate the spendings on the following parts: food, transport, health, leisure, etc. It will make the user aware of his spending pattern and help on how to plan it in a more intelligent way.

2. Overall Description

2.1 Product feature

The major feature of money planner application are the following.

- Expenditure analysis

  The user can figure out how much money he spent over the month, the system will graph it by category. It can also compares with last month's spending, making it easier to see spending history.

- Alarm function

  A spending alarm can be setted and it will alert the user when the spendings reaches the value designated previously.

- Balance display

  The user can check the balance in real time.

2.2 User classes and characteristics

The money planner application can be used by people of any age. Specifically among those who are not good at managing money. It is especially recommended for students who receives pocket money and for those who have a fixed monthly fee.

- Check your balance to see how you will spend it for the rest of the month.
- Alarm function give you instant confirmation as soon as you spend.

2.3 Operating environment

- Android Studio
- Language : JAVA

2.4 Assumption and dependencies

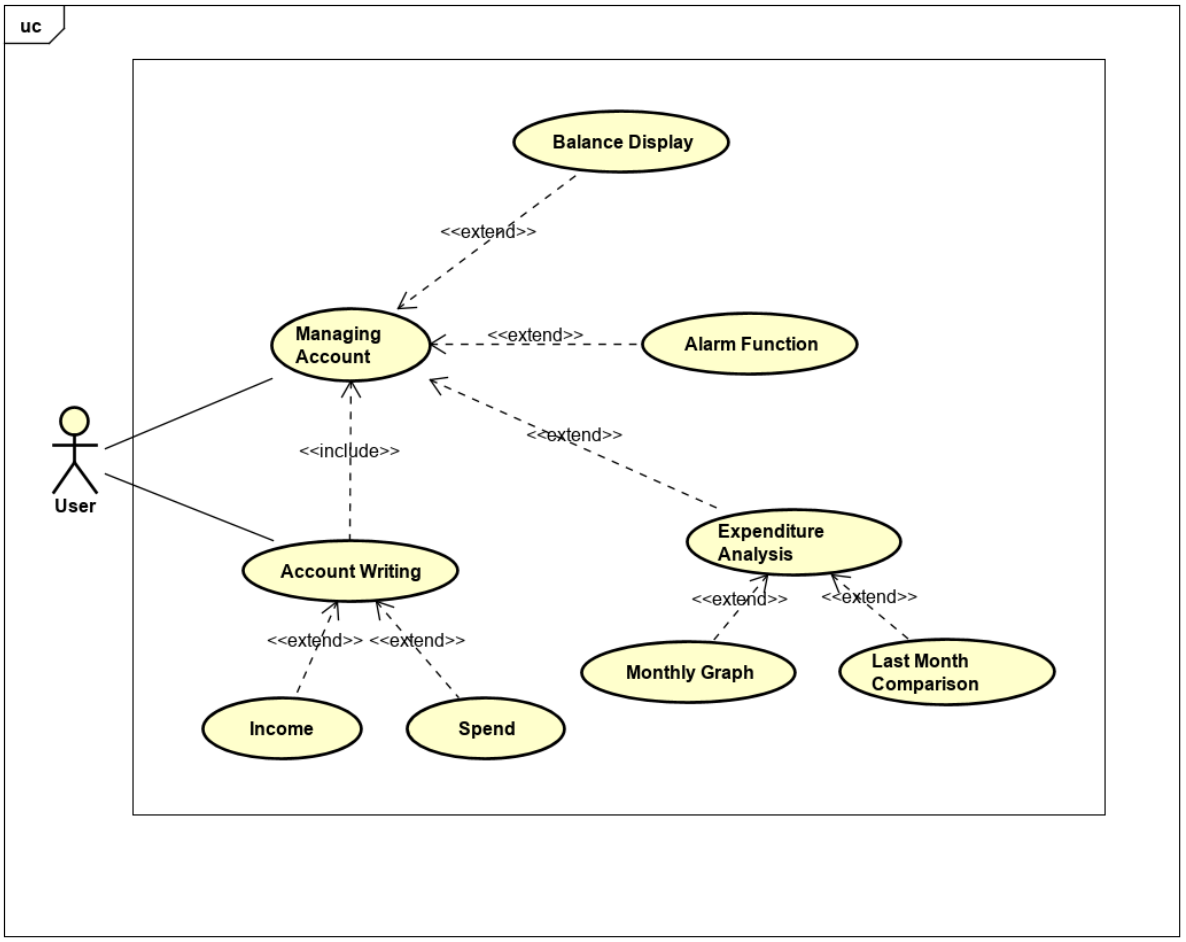
- Assumption

  Used for programs assuming good security. As it is a money planner application, security is a really important part. Therefore, the security is designed well so that there is no risk of hacking.

- Dependencies

  We need to set up a basic password entry to enable security.

3. Use case diagram



4. Use case description

Use case: Balance Display
Actor: User.
Description: The user inserts his account balance on the application and it shows the money balance to him.
Precondition: The user logs into his account.
Postcondition: The application shows the bank account balance to the user.
Main flow:

1. The system requests the user to insert his account balance.
2. The user inserts the data.
3. The user confirms the data inserted.
4. The system attributes the data to the respective user.

Use case: Alarm function
Actor: User.
Type: Primary.
Description: The user sets when the app will warn of how much money was spent.
Precondition: The user logs into his account.
Postcondition: The application has a money alarm.
Main flow:

1. The system requests the user to set the money alarm.
2. The user inserts the value.
3. The system sets the new money alarm.

Use case: Income
Actor: User.
Description: The user inserts how much money he received.
Precondition: The user receives some money.
Postcondition: The balance will have more money than before.
Main flow:

1. The system requests the user to insert how much money he received.
2. The user fills the money quantity.
3. The user confirms the quantity.
4. The system adds the value from the total balance.

Use case: Spend
Actor: User.
Description: The user inserts how much money he spent.
Precondition: The user spends the money.
Postcondition: The balance will have less money than before.
Main flow:

1. The system requests the user to insert how much money he spent.
2. The user fills the money quantity.
3. The user confirms the quantity.
4. The system subtracts the value from the total balance.

Use case: Last Month Comparison
Actor: User.
Description: The system compares the user spends from the actual month with the previous one.
Precondition: Have at least more than one month of account writing
Postcondition:
Main flow:

1. The system compares the data from the current month with the last one
2. The system generates a graph with the comparison
3. The system shows the graph to the user

Exception:

Flow 1: There's no record of the previous month

1. The system verifies that there's no record of the previous month
2. The system warns the user that at least one month needs to pass so he will be able to make the comparison

5. System Feature

4.1 Description and Priority

- Among the main functions, the easiest to implement is the 'balance display' or 'alarm function'. However, all of these are based on 'Consumption, Expenditure input system'. So the priority is as follows.

Consumption, Expenditure input system -> Balance display -> Alarm function -> Expenditure analysis

4.2 Expected requirement

As the system requires users to enter their own income and spending, they should not accept any input values other than numbers. At that point, it seems necessary to notify the user immediately that an error has occurred.

6. External Interface Requirement

5.1. User Interface

This app needs to have an easy interface so the users will be able to understand the appearance or layout of the contents. We can increase the ease of use with intuitive icons that helps to identify features and uses. Also this applications will open with a brief splash. After that, the display which shows their balance will be out. When the users click the menu, they can confirm their monthly expenditure analysis. In addition, they can set up the alarm which rings if it is over the set limit. All activities will be designed to be opened, closed or swiped by the user except the splash.

5.2. Hardware Interface

- An android phone or tablet

5.3. Software Interface

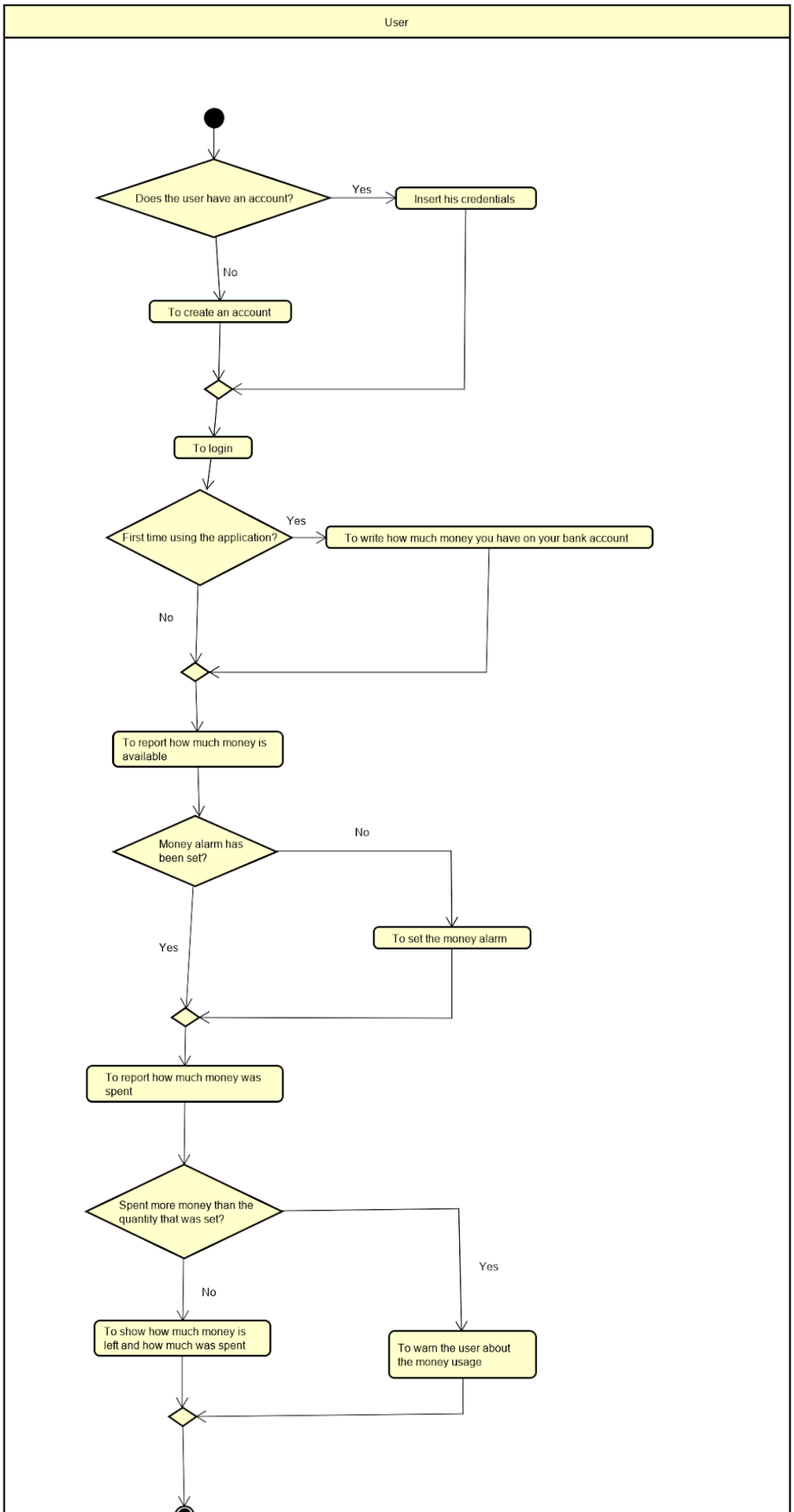
Following are the software used for the money planner application.

Software used	**Description**
Operating system	We have chosen mobiles operating system for its best supports and user-friendliness.
Database	To save the money records, we have chosen Firebase Realtime Database.
Authentication	To realize the log-in service, we have chosen Firebase auth.
Android studio	To implement the project we have chosen Java language.

5.4. Communications Interface

This project supports all types of android applications. It is communicated with the firebase so the user will be able to login.

6. Activity Diagram



7. Functional / Non-functional Requirements

7.1 Functional Requirements

FR01: The system should allow the user to create an account
FR02: The system should allow the user to log into his own account
FR03: The system should allow that the user inserts his spends
FR04: The system should allow that the user inserts his incomes
FR05: The system should allow the user to set an alarm
FR06: The system should allow the user to see how much money he spent
FR07: The system should allow the user to compare the spends from the current month with last month
FR08: The system should show the user's current balance

7.2 Non-functional Requirements

Usability:
NFR01: The system should have an interface with buttons/functions common between systems
NFR02: The system should perform the functions at least 2 seconds after the user clicks on the button
NFR03: The system should show the user balance at least 2 seconds after the login
NFR04: The system should be able to compare user's spending from the current month with their spending from last month
NFR05: The system should be able to turn on the alarm when the spending limit is trespassed
Security:
NFR06: The system should only show the information from the current user and no one else
Reliability:
NFR07: The system should be available 24 hours per day, 7 days per week
Efficiency:
NFR08: The system should store the data on a non relational database
NFR09: The system should work on a less robust device
Interoperability:
NFR10: The system should work on a big as possible variety of mobile devices
Maintainability:
NFR11: The system should provide code with good programming practices
NFR12: The system should provide ways to change the data if its format is outdated
Portability:
NFR13: The system should be used on mobile devices

8. References

https://krazytech.com/projects/sample-software-requirements-specificationsrs-report-airline-database