MrPenn

SINGLE SOFTWARE DESIGN DESCRIPTION DOCUMENT

Revision 2 – 15/02/2020

Abbo Giulio Antonio

CONTENTS

[1 Introduction 3](#_Toc33220099)

[1.1 Purpose 3](#_Toc33220100)

[1.2 Overview 3](#_Toc33220101)

[1.2.1 Goals 3](#_Toc33220102)

[1.2.2 Requirements 3](#_Toc33220103)

[1.2.3 Assumptions 4](#_Toc33220104)

[1.2.4 Other relevant data from RSD 4](#_Toc33220105)

[1.3 Definitions 4](#_Toc33220106)

[1.4 Acronyms and abbreviations 4](#_Toc33220107)

[1.5 Revision history 4](#_Toc33220108)

[2 References 5](#_Toc33220109)

[3 Context 6](#_Toc33220110)

[3.1 Use case 6](#_Toc33220111)

[4 Composition 7](#_Toc33220112)

[4.1 Functional decomposition 7](#_Toc33220113)

[4.2 Run time decomposition 7](#_Toc33220114)

[5 Logical 8](#_Toc33220115)

[6 Interactions 8](#_Toc33220116)

[7 Patterns 8](#_Toc33220117)

[8 Algorithm 9](#_Toc33220118)

[9 Resources 9](#_Toc33220119)

[10 User Interface 9](#_Toc33220120)

[11 Implementation notes 9](#_Toc33220121)

# Introduction

## Purpose

The purpose of the MrPenn system is to allow its users to store their money transactions, then they will have access to their personal balance and some personal statistics or graphs based on this data. All the data is personal and confidential, thus is not shared between users.

## Overview

### Goals

* G1: The system allows users to privately store their transactions.
* G2: Each user can see, update and delete each of his transactions.
* G3: The user can request statistics based on his transactions.
  + Transaction that have not been returned.
  + Total balances for each entity.
  + Partial monthly balances for each entity and category.

### Requirements

* R1: The user is prompted to log in if necessary
* ~~R2: The user can sign up to the system~~[[1]](#footnote-1)
* R3: The user can add a new transaction with:
  + Id
  + Positive amount
  + Origin and destination entities
  + Date
  + Optionally one or more categories
  + Optional notes
  + Whether must be returned
  + Optional returning id
* R4: The backend checks that the new transaction is correct and stores it
* R5: For each transaction, the value is subtracted to the source entity
* R6: For each transaction, the value is added to the destination entity
* R7: For each transaction, the value is added to its categories
* R8: The data is only accessible by the user that created it
* R9: The user can see, update and delete each of his transactions
* R10: The user can specify preferred entities
* R11: The user can specify preferred categories
* R12: The user can add some of the preferred entities to a “total”, that is their sum
* R13: The “total” is shown with other entities in the statistics, but cannot exist in transactions
* R14: When an entity is set as preferred, the user is asked for its initial balance
* R15: Each category allows to specify if its value is positive or negative
* R16: The user can request the monthly partial balances for each entity
* R17: The user can request the monthly partial balances for each category
* R18: The user can request the up-to-this-month balances for each entity
* R19: The user can request the up-to-this-month balances for each category
* R20: The user can request the total balances for each entity
* R21: The user can see some statistics:
  + Transaction that have not been returned
  + Graph with preferred entities balances by month
  + Partial monthly balances for the current and previous month, for preferred entities
  + Partial monthly balances for the current and previous month, for preferred categories
  + Total balance for the preferred entities
  + Graph with the monthly spending for each category

### Assumptions

* D1: The system relies on the abstraction it is built on for the security of the connection.
* D2: The user has a device and the MrPenn app.
* D3: The user is connected to the internet most of the time.
* D4: The external services behave correctly.

### Other relevant data from RSD

* ~~U1: User sign up.~~
* ~~U2: User log in.~~
* ~~U2: New transaction.~~
* ~~U3: Request transactions.~~
* ~~U4: Request statistic data.~~
* UI1: The UI adheres for the most part to the Material guidelines, especially for usability.
* UI2: A transaction that must be returned allows to open a precompiled form to insert the returning transaction.
* UI3: When adding a transaction, the user can specify which transaction is returning.
* O1: The data must be portable.
* O2: The back end is always backward-compatible (from the viewpoint of the front end).
* O3: The system will adhere to the principal GDPR guidelines.

## Definitions

* System: the software to be.
* User: an individual that uses the app.
* Transaction: an exchange of money from a certain origin to a certain destination.
* (Personal) balance: the sum of all the transactions for an entity or category.
* Entity: the source or destination of the money in a transaction.
* Category: labels for the transactions.
* To return: a sum of money is to be returned if it must be given back eventually.
* Initial balance: for an entity, the initial sum of money, used to produce statistics.
* Total: a special entity that is the sum (in the context) of other selected entities.
* Positive or negative category: whether the transactions in this category are incomes or outgoings.
* Backend: the software behind the user interface and the external services if present.

## Acronyms and abbreviations

* Gn: goal number n.
* Rn: requirement number n.
* Dn: domain assumption number n.
* Un: use case number n.
* UIn: user interface requirement number n.
* On: other requirement number n.

## Revision history

* Version 1: first version.
* Version 2: switched from client-server to client only with backup.

# References

The structure of this document is inspired on those proposed in IEEE 1016.

Material Design by Google.

GDPR regulations.

# Context

This design document focuses on the app and its characteristics. Since the architecture moved from client-server to client only, the backend now refers to the app’s core functionalities.

## Use case

Figure 1 represents the use cases of the front end and their relationship with the backend functions.



Figure 1 – UML use case diagram for the front end.

Here follows a specification of each use case, the actors are not specified since there is only the user.

#### UA: New transaction

Execution: the user inserts the data and sends it.

Postcondition: the data is checked and if correct it is stored. If not, the user receives an error message.

Notes: the backend can perform additional operations when it receives the transaction.

#### UB: Request transactions

Execution: the user requests a list of the transaction he has inserted. The list is created and displayed.

Postcondition: The user can see the requested transaction in detail and modify them.

Notes: when the user modifies or deletes a transaction, the backend could do some operations on the data.

#### UC: Request statistic data

Execution: the app fetches some data or uses the data previously cached to create the required statistics or graph.

Postcondition: the user can see the statistics or graph.

Notes: the client could keep some data cached.

#### UD: Import / Export

Execution: the app allows the user to import or export transactions.

Postcondition: the user can import transaction into the database and export it to file.

#### UE: Manage settings

Execution: the user can add or change details on transactions and the elements displayed.

Postcondition: the changes are correctly memorized by the system.

Notes: settings can also be managed through storage.

# Composition

The app is entirely on the user’s device. It will offer the possibility to automatically backup on a cloud storage. The storage of choice will be Google Drive for start, with the possibility to expand the offer in the future.

## Functional decomposition

Figure 2 represents the generic component diagram for the app. Only the main components are shown. The app will be developed in Flutter, which has a Provider package recommended by the developers. This allows to create the skeleton of the app and have a separate state that is provided to it with the data only when needed; the view can then update the state if necessary. The state will then take care of updating the database.

Intl package is used for internationalization, Sqflite package is used to manage the database.



Figure 2 – UML component diagram

## Run time decomposition

There are no relevant aspects regarding run time decomposition.

Figure 3 shows that the application, specified in the component diagram in Figure 2, is deployed on a mobile device. It is possible for it to communicate with an external backup solution (initially Google Drive).



Figure 3 – UML deployment diagram

# Logical

## Interfaces

## Model

## Database



Figure 4 – Database scheme

Figure 4 shows the database scheme: it contains three tables, one for all the transaction and one for entities and categories respectively. The primary keys are highlighted. The field categoriesIdList contains a json list of the names of the categories; dateTime is in microseconds since epoch. Some integers fields represent Boolean values.

# Interactions

Sequence diagram

# Patterns

Nothing relevant.

# Algorithm

Nothing relevant.

# Resources

Resource utilization.

# User Interface

The user interface consists in a main page called HUD where personalized widget can be put. Widgets contain the statistics requested. A second tab allows to access all the transactions. Selecting a transaction, it is possible to see its details and to interact with it (modify, delete…). A settings or options page allows to access the configuration parameters of the app and more functions. A specific button allows to add a new transaction.

The UI is not described in depth here, since needs more studies ad hoc; the design of the app codebase will allow to easily make visual changes without modifying the behavioural aspects.

# Implementation notes

The language of choice will be Dart, thanks to the Flutter framework it will be possible to write a single app for many platforms. Focus will be put on maintainability, testing, and code compartmentalization.

Bug: i/o, sliver, filtri, doppio spazio schede, categorie negative, rimuovo con id?

road map: backup, fix returns

settings (bind to tool?), other statistics: temporal graph, bar graph, to refund(page?)

(translation and versioning)

1. No longer required since there is no registration. [↑](#footnote-ref-1)