Software Requirements Specification

for Project Cashr

Prepared by   
Yana Koval, Marek Szeles  
ČVUT FEL SIT,   
Enterprise architectures

Version 1.0  
27. 11. 2016

Table of Contents

1. Introduction 1

1.1 Purpose 1

1.2 Intended Audience and Reading Suggestions 1

2. Overall Description 1

2.1 Product Perspective 1

2.2 Product Functions 1

2.3 User Classes and Characteristics 2

2.4 Operating Environment 2

2.5 Design and Implementation Constraints 2

2.6 User Documentation 2

3. External Interface Requirements 3

3.1 User Interfaces 3

3.2 Software Interfaces 4

3.3 Communications Interfaces 4

4. System Features – Use Cases 5

4.1 Basic features to be implemented within project scope 5

4.2 Advanced features NOT to be implemented within project scope 7

5. Other Nonfunctional Requirements 8

5.1 Performance Requirements 8

5.2 Security Requirements 8

5.3 Software Quality Attributes 8

5.4 Business Rules 8

6. Other Requirements 9

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
| Marek Szeles | 22.11.2016 |  | 0.1 |
| Marek Szeles | 23.11.2016 | Added new chapters | 0.2 |
| Marek Szeles | 26.11.2016 | Added new chapters | 0.3 |
| Marek Szeles | 27.11.2016 | Update of object model after new specification | 0.4 |
| Marek Szeles | 27.11.2016 | First release version | 1.0 |

# Introduction

## Purpose

The product of this project will be a Java EE based Maven compilable program that allows its registered users to analyse their personal finances.

## Intended Audience and Reading Suggestions

This document is intended for anyone interested to learn about the concepts and architecture used in the Cashr project.

# Overall Description

## Product Perspective

This is a standalone product, completely independent in its basic functions. If its planned advanced functions are to be implemented, some use of external APIs is to be expected.

## Product Functions

### Basic functions to be implemented within project scope

* Adding an income/expense using a name, category and amount
* Showing an overview of the current wealth, with segmentation to different parts (accounts, cash, currencies, investments)
* Historical development overview
* Category management – creation of own categories (Such as “sport”, “food”, or “transport”]
* Option to create a financial plan, limits for spending and warnings when approaching them.

### More advanced functions NOT to be implemented within project scope

* Login and registration using external accounts (Such as Facebook, or Google)
* Connection to a bank account
  + Ability to fetch incoming and outgoing transaction information from the account
* Invoice and receipt scanning
  + The system will be able to read an electronic form (pdf/jpg) of an invoice and load relevant data from it (such as vendor, amount, category derived from keywords)
  + This information can be read from a photo taken by a smartphone, too
* Connection to investment portals
  + The system will allow to use an allocated amount of funds at stock exchange servers to gain profit over time

## User Classes and Characteristics

Apart from the state before logging in, there are only two expected user classes defined within the Cashr project – user and admin. Both have very similar functions, only overall rights are different.

### Everyone (before logging in)

Anyone accessing the server/side program through a browser will be presented a simple form to log in with or the possibility to register into the system using a username and password.

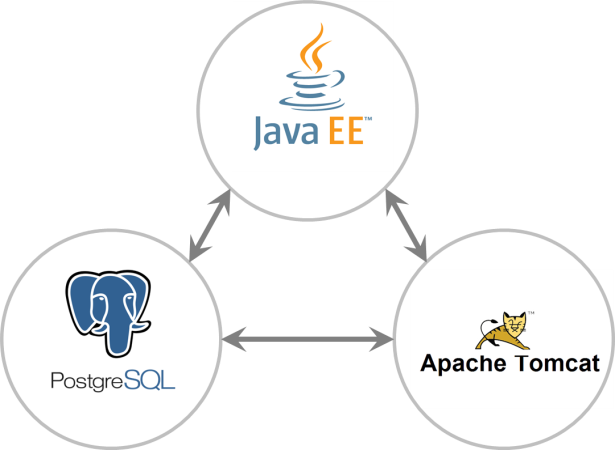
### User

After logging in, a user can review his accounts, and after selecting one of them, has the possibility to review the history of the account, to see the current balance broken down to separate categories, and to add new entries, both income and expense to the account log. The user can also select a settings screen where he can manage their own custom categories and tags for entries.

### Admin

An admin has all the rights a standard user has, but also can create and destroy all the default tags and categories visible to all users.

## Operating Environment

The source code of the program is assembled mainly using Java Enterprise Edition. It is deployed on a Tomcat server environment, using a Postgres database to store inputted information.

## Design and Implementation Constraints

If some of the advanced functions are to be implemented, the development is likely to be limited by external APIs – for example, if using an external API to recognize text, or when connecting to an external API handling the access to the stock exchange.

## User Documentation

Overview and development documentation will be provided, along with a simple user manual in the form of pdf files.

# External Interface Requirements

## User Interfaces

The GUI is going to be design with minimalism and simplicity in mind in order to allow for intuitive user flow of work. The layout is to be clearly separated into functional and passive parts, with labels describing all buttons and functions.

Here are several drafts of how the interface may look like:





Figure : Login screen

Figure : Registration screen

## 

Figure : Transaction history GUI

## Software Interfaces

The program in its basic functions is fully integrated with itself and needs no access to outside sources. If and when the advanced functions are implemented, they will by definition need interfaces to access outside systems, however these systems and the nature of such transactions are to be determined when it comes to the relevant phase of development.

## Communications Interfaces

Communication between the server and the user is going to take place through a web browser, using the HTTP protocol. All other communication is translated directly into code and objects within the program itself.

# System Features – Use Cases

This section describes the features to be implemented in the project and features planned for future development.

## Basic features to be implemented within project scope

### Registration

#### Description and Priority

The user will be able to register using a username and a password of his choosing

High priority

#### Stimulus/Response Sequences

The user selects register > Fills out their username and password, which he inputs twice for confirmation > Registration complete

#### Functional Requirements

The software is required to assign unique Ids to all new users, check for database consistency (duplicate usernames) and handle security (hash passwords)

### Login

#### Description and Priority

The user will be able to login using a combination of username and password

High priority

#### Stimulus/Response Sequences

The user selects login > Fills out their username and password > System checks if an entry in database equals to input > Registration complete

#### Functional Requirements

The software is required to keep track of each login session and not mix up requests.

### Add/remove account

#### Description and Priority

The user will be able to add new account to his profile

High priority

#### Stimulus/Response Sequences

The user selects to add account > Fills out information about the account (currency) > confirms > New account added

#### Functional Requirements

The software is required to keep and store all accounts and show them only to their respective owners

### Add/remove income/expense

#### Description and Priority

The user will be able to add new entries to the income/expense table

High priority

#### Stimulus/Response Sequences

The user selects to add entry > Fills out information about the transaction > confirms > New entry added

#### Functional Requirements

The software is required to keep and store all entries until deleted

### Add/remove custom categories and tags

#### Description and Priority

The user will be able to add new categories and tags to mark entries with

Medium priority

#### Stimulus/Response Sequences

The user selects to manage categories and tags > Fills out information about the category/tag > confirms > New category/tag added

#### Functional Requirements

The software is required to keep and store all categories and tags until deleted and differentiate between users (admin/user) and show them only their respective tags/categories as well as the default ones.

### Show history of transactions

#### Description and Priority

The user will be able to view his account history

Medium priority

#### Stimulus/Response Sequences

The user selects history > Account history is shown

#### Functional Requirements

The software is required to show all transactions stored relevant to particular account

### Show account overview

#### Description and Priority

The user will be able to view his account overview – how much funds he has in which accounts

Medium priority

#### Stimulus/Response Sequences

The user selects overview > User accounts overview is shown

#### Functional Requirements

The software is required to show all transactions stored relevant to particular account

### Make limit to spending

#### Description and Priority

The user will be able to create a financial plan and set up warnings when approaching limit

Low priority

#### Stimulus/Response Sequences

The user selects Planning > Enters limits and warnings > system periodically checks whether user isn’t overdrafting

#### Functional Requirements

The software is required to regularly assess users financial performance and give warnings when approaching overdraft

## Advanced features NOT to be implemented within project scope

### Registration using Google or Facebook

#### Description and Priority

The user will be able to register using Google or Facebook for login credentials,

#### Stimulus/Response Sequences

The user selects register > Register using Google/Facebook > Grants permission to Cashr to fetch email > Registration complete

#### Functional Requirements

The software is required to repeatedly and correctly handle authorization communication with external servers.

### Login using Google or Facebook

#### Description and Priority

The user will be able to login using Google or Facebook for login credentials,

#### Stimulus/Response Sequences

The user selects login > Login using Google/Facebook > System checks if user is registered > login complete

#### Functional Requirements

The software is required to repeatedly and correctly handle authorization communication with external servers.

### Connection to a bank account

#### Description and Priority

The user will be able to connect to a bank account and automatically synchronise his transactions

#### Stimulus/Response Sequences

The user connects bank account > system periodically refreshes account movements

#### Functional Requirements

The software is required to repeatedly and correctly refresh connection to bank account and download relevant data

### Scanning an invoice

#### Description and Priority

The system will be able to fetch relevant information from a digital copy of an invoice or receipt, even a photo of it

#### Stimulus/Response Sequences

The user selects scan invoice > uploads picture > system fetches information from data and pre-fills form > asks user to confirm > new entry created

#### Functional Requirements

The software is required to efficiently and correctly fetch data from different digital sources

### Connect to investment portals

#### Description and Priority

The system will allow users to dedicate funds for investments and manage their portfolios

#### Stimulus/Response Sequences

The user selects stock market > selects amount of funds dedicated and risk to be taken > computer monitors movements and informs user of development

#### Functional Requirements

The software is required to efficiently and correctly fetch data from different stock market servers

# Other Nonfunctional Requirements

## Performance Requirements

The program is expected to handle all logged users with no noticeable delays. To assure this, sufficient hardware is needed for possible service overload.

## Security Requirements

Due to the nature of the stored data, the database and user login data is going to be secured using different methods, such as hashing passwords and passing data through secure connections, to which a separate security layer will be committed.

## Software Quality Attributes

The resulting software should be flexible and adaptable to different uses – it should be simple and intuitive to add new categories or tags for example. Also, if, in the future, a new world currency comes to exist which is going to be widely used, its retrospective implementation into the project should be simple as well.

## Business Rules

While all users can see the default tags, only admins can see who (which admin) had created them.

# Other Requirements

The project is required to have at least five database tables, have at least one M:N relation and use one dependency. strategy.

Appendix A: Glossary

Java EE – Java Enterprise Edition

GUI – Graphical User Interface

Appendix B: Analysis Models

**Object model**

