Money Tracker

Software Requirement Specification

Version: <1.0>

Date: <03.11.2015>

State: <Draft>

Created by: <Andrei Martinescu>

Created by		**Approved by**	
Name:	Andrei Martinescu	Name:	
Date:	03.11.2015	Date:	
Signature:		Signature:	

1 History

1.1 Document Versions

Version	Date	Author	Changes
<1.0>	03.11.2015	<Andrei Martinescu>	<Initial version>

2 Contents

2.1 Content

1 History.....Error! Bookmark not defined.
1.1 Releases.....**Error! Bookmark not defined.**
2 Content.....Error! Bookmark not defined.
2.1 Table of content.....**Error! Bookmark not defined.**
2.2 Figures.....**Error! Bookmark not defined.**
2.3 Tables.....**Error! Bookmark not defined.**
3 Introduction.....Error! Bookmark not defined.
3.1 Purpose of this document.....**Error! Bookmark not defined.**
3.2 Scope, area of applicability.....**Error! Bookmark not defined.**
3.3 Definitions.....**Error! Bookmark not defined.**
3.4 Abbreviations.....**Error! Bookmark not defined.**
3.5 References.....**Error! Bookmark not defined.**
4 General description.....Error! Bookmark not defined.
4.1 Capabilities and limitations.....**Error! Bookmark not defined.**
4.2 Safety & Hazard Considerations..... 8
5 Requirements.....Error! Bookmark not defined.
5.1 Requirement identifier naming convention..... 9
5.2 Parts of requirement..... 9
5.3 Functional requirements..... 10
5.4 Security requirements.....**Error! Bookmark not defined.**
5.5 Safety requirements.....**Error! Bookmark not defined.**
5.6 Interface requirements.....**Error! Bookmark not defined.**
5.6.1 User interface.....**Error! Bookmark not defined.**
5.6.2 Hardware interface.....**Error! Bookmark not defined.**
5.6.3 Software interfaces.....**Error! Bookmark not defined.**
5.6.4 Communication interfaces.....**Error! Bookmark not defined.**
5.7 Requirements on data.....**Error! Bookmark not defined.**
5.8 Requirements on performance.....**Error! Bookmark not defined.**
5.9 Other requirements.....**Error! Bookmark not defined.**

2.2 Figures

None.

2.3 Tables

None.

3 Introduction

3.1 Purpose of this document

The purpose of this Customer Requirement Specification document is to provide a detailed description of the functionalities of the Money Tracker software application. This document will cover each of the software application functionalities.

3.2 Scope, area of applicability

Money Tracker application is a C++-based computer operating systems which helps people to track their finances, in a very easy and efficient way to keep their wallet in good shapes. Application provides intuitive features, which allows user to create wallet, add transactions (income and spend), get balance.

3.3 Definitions

Money Tracker	Name of software application
Create	Command for create a wallet
Income	Command for adding transaction income in wallet
Spend	Command for adding transaction spend in wallet
Balance	Command for printing the balance of the wallet
Moneytracker.exe	Name of exe file for application
Salary	Default category for income transaction
Other	Default category for spend transaction
Transaction time	Transaction time for each transaction type printed on user output
Epoch time	Transaction time in Epoch format printed in wallet
-c	Type of command for creating a specify category for income/spend
--category	Type of command for creating a specify category for income/spend

Feature	Properties of software application
User	Someone who interacts with the software application
Administrator	System administrator who is given permission for managing and controlling the software application
Command line	Console user interface with software application
Notepad++	Code editor for c++ software application
Money tracking	Name of software application file

3.4 Abbreviations

argc	Number of arguments entered by user in command line
argv	Arguments entered by user in command line
Cmd	Command line
Mingw32-make	Minimalist GNU for Windows, compiler program for software application
Batch	Type of files used for acceptance tests
Git	GitHub, web-based repository hosting service
Ctrl + c	Quit the running application from command line

3.5 References

4 General description

Money Tracker is an application which helps you keep an eye on your personal budget.

For now it works from the command line and you can control by passing some commands and parameters to it.

4.1 Relation to other projects

This application will be independent of any other projects.

4.2 Relation to former and future projects

This will be the first version of the application, so it does not have any connection to former projects.

4.3 Goal of the product

Money Tracker is an application which helps you keep an eye on your personal budget. Its goal is to track users spending and income in the personal wallet, which starts with an initial amount.

If you find it hard to keep track of where the money in your pocket goes, try carrying around a notebook for a month or two. Write down every purchase you pay for with cash (parking, coffee, lunch). And add in any receipts for bill payments etc. to get a full picture of what you are spending.

You can do all of these using Money Tracker application to keep track on your computer all your spending and incomes.

The goal is to develop a stable application which is able to process all you command implemented in command line.

4.4 Delimitation and integration of the product

Money Tracker application is a C++-based computer operating systems.

Delimitation of Money Tracker is based on usability only on computer operating systems.

Integration: The Money Tracker application is a C++-based, it supposed to be fully integrated in all computer operating system from command line. Will response to user inputs with minimal response time (less than second per command). For fully integration the application executable *.exe will need to have the moneytracker.config placed next to the application, its name does not change, and it is not moved to another path. For features implementation the restrictions for moneytracker.config location, name and path will be removed.

Having few delimitation the Money Tracker will be a portability application. The application is a standalone application, so it does not integrate with other products or services. It does not depend of databases, servers, etc.

4.5 Overview on the required functionality

Money Tracker application has implement 4 functionalities:

- Create: create a wallet with a default amount or with a specified amount. You can also add a name for a wallet or a path for creating a specified wallet.
- Income: add transaction income with a specified amount or with a specified category.
- Spend: add transaction spend with a specified amount or with a specified category.
- Balance: print the balance from the wallet.

4.6 General capabilities and limitations

The programming language used for the development of the application will be C++, so the application will not be dependent on the operating system.

4.7 Users of the product

No special requirements regarding targeted users.

4.8 Safety & Hazard Considerations

The application will try to avoid unwanted behaviors that can be related to the wrong format of loaded file and will try to handle properly the processing data of large size.

5 Requirements

This section of the SRS should contain all the software requirements to a level of detail sufficient to enable designers to design a system to satisfy those requirements, and testers to test that the system satisfies those requirements. Requirements are therefore to be unambiguous and testable. Furthermore requirements are to be redundancy free, consistent and without contradictions.

The requirements should include at a minimum a description of every input (stimulus) into the system, every output (response) from the system and all functions performed by the system in response to an input or in support of an output.

5.1 Requirement identifier naming convention

5.2 Parts of requirement

Identifier: a unique identifier which is created according to rules provided in the Requirement Identifier naming convention section above.

Requirement statement: detailed textual description of the requirements. Consider that based on this statement a clear, unambiguous understanding of the requirement should be guaranteed. Verification regarding the fulfillment of the requirement in the final product should be possible based on this requirement statement.

Type: R a requirement, that shall be implemented and tested
O an optional requirement, if implemented it shall be tested
L a limitation or negative requirement, which shall not be tested
E an opportunity for evolution. A requirement that the design should allow for in the future. This requirement shall not be implemented. If it is implemented, the type should be changed to “O”.

Safety: An “X” is placed in this box if the requirement is related to safety considerations.

Origin (optional): reference to other customer requirement specification or other document which provided input to this requirement. Possibly identifying and pointing to a single statement of this document.

5.3 Functional requirements

CREATE:

Identifier	*#REQ_FUN_SRS_CREATE_NO_AMOUNT_SPECIFIED*
Requirement statement	Calling a create wallet command with no amount specified will create a wallet with the default amount of +00.00
Type	R
Safety	
Origin	*#REQ_FUN_CRS_CREATE*

Identifier	*#REQ_FUN_SRS_CREATE_AMOUNT_SPECIFIED*
Requirement statement	Calling a create wallet command with amount specified will create a wallet with the specified amount by user.
Type	R
Safety	
Origin	*#REQ_FUN_CRS_CREATE*

Identifier	*#REQ_FUN_SRS_CREATE_VALID_COMMAND*
Requirement statement	Calling a create wallet command with or without amount specified will create a wallet and print a confirmation message.
Type	R
Safety	
Origin	*#REQ_FUN_CRS_CREATE*

Identifier	*#REQ_FUN_SRS_CREATE_INVALID_COMMAND*
Requirement statement	Calling an invalid create wallet command with or without amount specified print an error message.
Type	R
Safety	
Origin	*#REQ_FUN_CRS_CREATE*

Identifier	*#REQ_FUN_SRS_CREATE_WALLET_EXISTS*
Requirement statement	Calling a create wallet command with a wallet that already was created or exists than an error will be printed on the command line.
Type	R
Safety	
Origin	*#REQ_FUN_CRS_CREATE*

Identifier	*#REQ_FUN_SRS_CREATE_PATH_EXISTS*
Requirement statement	Calling a create wallet command with a path for wallet that already was created or exists than an error will be printed on the command line.
Type	R
Safety	
Origin	SRS

Identifier	*#REQ_FUN_SRS_CREATE_INCORECT_AMOUNT*
Requirement statement	Calling a create wallet command with an amount not a valid will print an error message on the command line informing the user that the amount is not valid.
Type	R
Safety	
Origin	*#REQ_FUN_CRS_CREATE*

Identifier	*#REQ_FUN_SRS_CREATE_WITH_PATH*
Requirement statement	Calling a create wallet command with a path for wallet specified will print a confirmation message containing the path on the command line.
Type	R
Safety	
Origin	SRS

INCOME:

Identifier	*#REQ_FUN_SRS_INCOME*
Requirement statement	Calling an income command will write the specified amount in the wallet file on new line. Default category for income is “salary”.
Type	R
Safety	
Origin	*#REQ_FUN_CRS_INCOME*

Identifier	*#REQ_FUN_SRS_INCOME_CATEGORY*
Requirement statement	Calling an income command with specified category will write the specified amount and category in the wallet file on new line. The correct command will contain income command on first parameter, a category command (specified by –c or --category) with a type of category (Ex: “salary”), and an amount. The correct command will accept both situations when category and amount positions are changed.
Type	R
Safety	
Origin	*#REQ_FUN_CRS_INCOME*

Identifier	*#REQ_FUN_SRS_INCOME_VALID_COMMAND_MESSAGE*
Requirement statement	Calling an income command will print a confirmation message on command line.
Type	R
Safety	
Origin	*#REQ_FUN_CRS_INCOME*

Identifier	*#REQ_FUN_SRS_INCOME_INVALID_COMMAND_MESSAGE*
Requirement statement	Calling an invalid income command will print an error message on command line.
Type	R
Safety	
Origin	*#REQ_FUN_CRS_INCOME*

Identifier	*#REQ_FUN_SRS_INCOME_SPECIFIED_CATEGORY_VALID_COMMAND*
Requirement statement	Calling an income command with specified category will print a confirmation message on command line.
Type	R
Safety	
Origin	*#REQ_FUN_CRS_INCOME*

Identifier	*#REQ_FUN_SRS_INCOME_SPECIFIED_CATEGORY_INVALID_COMMAND*
Requirement statement	Calling an invalid income command with specified category will print an error message on command line.
Type	R
Safety	
Origin	SRS

Identifier	*#REQ_FUN_SRS_INCOME_WALLET_CHANGES*
Requirement statement	Calling an income command will write the amount in the wallet on a new line at the end of the file and should put another empty line after the new line added.
Type	R
Safety	
Origin	*#REQ_FUN_CRS_INCOME*

Identifier	*#REQ_FUN_SRS_INCOME_WALLET_NOT_OPEN*
Requirement statement	Calling an income command will write the amount in the wallet .If the wallet couldn't be opened then an error message will be printed on the command line informing the user about the error.
Type	R
Safety	
Origin	*#REQ_FUN_CRS_INCOME*

Identifier	*#REQ_FUN_SRS_INCOME*
Requirement statement	Calling an income command with no amount specified will print an error message on command line.
Type	R
Safety	
Origin	*#REQ_FUN_CRS_INCOME*

SPENDING:

Identifier	*#REQ_FUN_SRS_SPENDING*
Requirement statement	Calling a spending command will write the specified amount in the wallet file on new line. Default category for income is "other".
Type	R
Safety	
Origin	*#REQ_FUN_CRS_SPENDING*

Identifier	*#REQ_FUN_SRS_SPENDING_CATEGORY*
Requirement statement	Calling a spending command with specified category will write the specified amount and category in the wallet file on new line. The correct command will contain spending command on first parameter, a category command (specified by –c or --category) with a type of category (Ex: “food”), and an amount. The correct command will accept both situations when category and amount positions are changed.
Type	R
Safety	
Origin	*#REQ_FUN_CRS_SPENDING*

Identifier	*#REQ_FUN_SRS_SPENDING_VALID_COMMAND_MESSAGE*
Requirement statement	Calling a spending command will print a confirmation message on command line.
Type	R
Safety	
Origin	*#REQ_FUN_CRS_ SPENDING*

Identifier	*#REQ_FUN_SRS_SPENDING_INVALID_COMMAND_MESSAGE*
Requirement statement	Calling an invalid spending command will print an error message on command line.
Type	R
Safety	
Origin	*#REQ_FUN_CRS_SPENDING*

Identifier	*#REQ_FUN_SRS_SPENDING_SPECIFIED_CATEGORY_VALID_COMMAND*
Requirement statement	Calling a spending command with specified category will print a confirmation message on command line.
Type	R
Safety	
Origin	*#REQ_FUN_CRS_SPENDING*

Identifier	*#REQ_FUN_SRS_SPENDING_SPECIFIED_CATEGORY_INVALID_COMMAND*
Requirement statement	Calling a spending income command with specified category will print an error message on command line.
Type	R
Safety	
Origin	*#REQ_FUN_CRS_SPENDING*

Identifier	*#REQ_FUN_SRS_SPENDING_WALLET_CHANGES*
Requirement statement	Calling a spending command will write the amount in the wallet on a new line at the end of the file and should put another empty line after the new line added.
Type	R
Safety	
Origin	*#REQ_FUN_CRS_SPENDING*

Identifier	*#REQ_FUN_SRS_SPENDING_WALLET_NOT_OPEN*
Requirement statement	Calling a spending command will write the amount in the wallet .If the wallet couldn't be opened then an error message will be printed on the command line informing the user about the error.
Type	R
Safety	
Origin	*#REQ_FUN_CRS_SPENDING*

Identifier	*#REQ_FUN_SRS_SPENDING*
Requirement statement	Calling a spending command with no amount specified will print an error message on command line.
Type	R
Safety	
Origin	*#REQ_FUN_CRS_SPENDING*

BALANCE:

Identifier	*#REQ_FUN_SRS_BALANCE*
Requirement statement	Calling a balance command will print the balance of all transactions founded in the wallet.
Type	R
Safety	
Origin	*#REQ_FUN_CRS_BALANCE*

Identifier	*#REQ_FUN_SRS_BALANCE_VALID_COMMAND*
Requirement statement	Calling a balance command will print a confirmation message on command line.
Type	R
Safety	
Origin	*#REQ_FUN_CRS_BALANCE*

Identifier	*#REQ_FUN_SRS_BALANCE_INVALID_COMMAND*
Requirement statement	Calling an invalid balance command will print an error message on command line.
Type	R
Safety	
Origin	*#REQ_FUN_CRS_BALANCE*

5.4 Security requirements

Identifier	*#REQ_SEC_SRS_ACCESS*
Requirement statement	Calling a login command will display a message for entering the username and the password.
Type	E
Safety	
Origin	*#REQ_SEC_CRS_ACCESS*

Identifier	*#REQ_SEC_SRS_ACCESS_WRONG_USERNAME*
Requirement statement	Calling a login command will display a message for entering the username and the password. Entering a wrong username will display an error message.
Type	E
Safety	
Origin	*#REQ_SEC_CRS_ACCESS*

Identifier	*#REQ_SEC_SRS_ACCESS_WRONG_PASSWORD*
Requirement statement	Calling a login command will display a message for entering the username and the password. Entering a wrong password will display an error message.
Type	E
Safety	
Origin	*#REQ_SEC_CRS_ACCESS*

Identifier	*#REQ_SEC_SRS_ACCESS_VALID*
Requirement statement	Calling a login command will display a message for entering the username and the password. Entering correct username and password will display a confirmation message.
Type	E
Safety	
Origin	*#REQ_SEC_CRS_ACCESS*

Identifier	*# REQ_SEC_CRS_USER_CREATE*
Requirement statement	Calling a create user command will display a message for entering the username and the password.
Type	E
Safety	
Origin	*#REQ_SEC_CRS_USER_CREATE*

Identifier	*# REQ_SEC_CRS_USER_CREATE_VALID*
Requirement statement	Calling a create user command will display a message for entering the username and the password. Entering a username not occupied by other account will display a confirmation message.
Type	E
Safety	
Origin	*#REQ_SEC_CRS_USER_CREATE*

Identifier	*# REQ_SEC_CRS_USER_CREATE_INVALID*
Requirement statement	Calling a create user command will display a message for entering the username and the password. Entering a username occupied by other account will display an error message.
Type	E
Safety	
Origin	*#REQ_SEC_CRS_USER_CREATE*

Identifier	*# REQ_SEC_SRS_FILE_ACCESS*
Requirement statement	Trying to open a file created by application will not be possible. The files created from application will be protected.
Type	E
Safety	
Origin	*# REQ_SEC_CRS_FILE_ACCESS*

5.5 Safety requirements

Identifier	*#REQ_SEC_CRS_PERSONAL_SAFETY*
Requirement statement	Application is stored on the personal computer. Any application crash will not affect the operating system and application data. The application data can be affected only by an operating system crash.
Type	R
Safety	
Origin	*#REQ_SEC_CRS_PERSONAL_SAFETY*

5.6 Interface requirements

Identifier	*# REQ_INT_CRS_CMD*
Requirement statement	CMD– Command line user interface, interacts with a computer program where the user (or client) types commands to the program in the form of successive lines of test (command lines). A valid command line will need to have specified name of *.exe and a name of a feature.
Type	R
Safety	
Origin	*# REQ_INT_CRS_CMD*

5.7 Data requirements

CREATE:

Identifier	*#REQ_DAT_SRS_CREATE_STRUCT*
Requirement statement	Calling a create wallet command with or without amount specified will create a wallet and print a confirmation message. The structure of create command is: name of application, command create, name of wallet (or path), an amount if the user want's to specify an amount.
Type	R
Safety	
Origin	*#REQ_FUN_CRS_CREATE*

Identifier	*#REQ_DAT_SRS_CREATE_WALLET_FILE_STRUCT*
Requirement statement	Calling a create wallet command will create a wallet file with the amount (specified or default) in the wallet file on the first line. The file should contain a new line after the last line configured by user.
Type	R
Safety	
Origin	*#REQ_FUN_CRS_CREATE*

Identifier	*#REQ_DAT_SRS_CREATE_NAME_OF_WALLET*
Requirement statement	Calling a create wallet command with a wallet specified in command line, the application will create the wallet with the name specified.
Type	*R*
Safety	
Origin	*#REQ_FUN_CRS_CREATE*

Identifier	*#REQ_DAT_SRS_CREATE_LOCATION_NO_PATH*
Requirement statement	Calling a create wallet command with no path specify for wallet, the application will create the wallet relative to where the command was called from command line.
Type	R
Safety	
Origin	*#REQ_FUN_CRS_CREATE*

Identifier	*#REQ_DAT_SRS_CREATE_LOCATION_PATH_SPECIFIED_LOCATION*
Requirement statement	Calling a create wallet command with a path specified for wallet, the application will create the wallet in the location specified by user.
Type	R
Safety	
Origin	*#REQ_FUN_CRS_CREATE*

Identifier	*#REQ_DAT_SRS_CREATE_DEFAULT_CURRENCY*
Requirement statement	Calling a create wallet command will create the wallet with the amount (specified or initial) on the first line in the wallet with the default currency RON.
Type	R
Safety	
Origin	*#REQ_FUN_CRS_CREATE*

Identifier	*#REQ_DAT_SRS_CREATE_AMOUNT_LIMITATION*
Requirement statement	A valid amount for create wallet file is a decimal number, positive or negative. The application will validate the amount and if the amount has only one or more than two decimals will convert the amount in an amount with two decimals by rounding up.
Type	R
Safety	
Origin	*#REQ_FUN_CRS_CREATE*

Identifier	*#REQ_DAT_SRS_CREATE_AMOUNT_LIMITATION_DOT*
Requirement statement	A valid amount for create wallet file is a decimal number, positive or negative. If the amount is entered with decimals the decimals will be separated by dot.
Type	R
Safety	
Origin	*#REQ_FUN_CRS_CREATE*

Identifier	*#REQ_DAT_SRS_CREATE_WALLET_FILE_STRUCT*
Requirement statement	Calling a create wallet command will create a wallet file with the amount (specified or default) in the wallet file on the first line. The file should contain a new line after the last line configured by user.
Type	R
Safety	
Origin	*#REQ_FUN_CRS_CREATE*

Identifier	*#REQ_DAT_SRS_CREATE_NO_WALLET_SPECIFIED_IN_MONEYTRACKER.CONFIG*
Requirement statement	Calling a create wallet command will check if a wallet is specified in default moneytracker.config file and if not should set the default wallet with the wallet entered by user.
Type	E
Safety	
Origin	*#REQ_FUN_CRS_CREATE*

INCOME:

Identifier	*#REQ_ DAT_SRS_INCOME_VALID_COMMAND_STRUCTURE*
Requirement statement	The correct command for income an amount in the wallet must have the income command and an amount specified. The command has the options to specify a category for income. Default category for income is “salary”.
Type	R
Safety	
Origin	*#REQ_FUN_CRS_INCOME*

Identifier	*#REQ_ DAT_SRS_INCOME_AMOUNT_VALID*
Requirement statement	For income command, a valid amount should be a positive number.
Type	R
Safety	
Origin	*#REQ_FUN_CRS_INCOME*

Identifier	*#REQ_ DAT_SRS_INCOME_AMOUNT_VALID_DECIMALS*
Requirement statement	For income command a valid amount should be a double number specified with 2 decimals. The decimals are separated by dot from the number.
Type	R
Safety	
Origin	*#REQ_FUN_CRS_INCOME*

Identifier	*#REQ_DAT_SRS_INCOME_DEFAULT_CURRENCY*
Requirement statement	Calling an income command will write the amount in the wallet with the default currency RON.
Type	R
Safety	
Origin	*#REQ_FUN_CRS_INCOME*

Identifier	*#REQ_DAT_SRS_INCOME_TIME*
Requirement statement	Calling an income command will write the amount in the wallet, and will specified the correct time of the transaction.
Type	R
Safety	
Origin	*#REQ_FUN_CRS_INCOME*

Identifier	*#REQ_DAT_SRS_INCOME_DEFAULT_SIGN*
Requirement statement	Calling an income command will write the amount in the wallet, and will specified the sign of the amount with default sign for income “+”.
Type	R
Safety	
Origin	*#REQ_FUN_CRS_INCOME*

Identifier	*#REQ_DAT_SRS_INCOME_DEFAULT_CATEGORY*
Requirement statement	Calling an income command will write the amount in the wallet, and will specified default category “salary” if the user didn’t specified an category.
Type	R
Safety	
Origin	*#REQ_FUN_CRS_INCOME*

Identifier	*#REQ_FUN_SRS_INCOME_SPECIFIED_CATEGORY*
Requirement statement	Calling an income command, the user can specified a category for income. The correct command should contain income command on first parameter, a category command (specified by –c or --category) with a type of category (Ex: “food”), and an amount. The correct command will accept both situations when category and amount positions are changed.
Type	R
Safety	
Origin	*#REQ_FUN_CRS_INCOME*

SPENDING:

Identifier	*#REQ_ DAT_SRS_SPENDING_VALID_COMMAND_STRUCTURE*
Requirement statement	The correct command for spending an amount in the wallet must have the spending command and an amount specified. The command has the options to specify a category for spending. Default category for spending is “other”.
Type	R
Safety	
Origin	*#REQ_FUN_CRS_SPENDING*

Identifier	*#REQ_ DAT_SRS_SPENDING_AMOUNT_VALID*
Requirement statement	For spending command, a valid amount should be a positive number.
Type	R
Safety	
Origin	*#REQ_FUN_CRS_SPENDING*

Identifier	*#REQ_ DAT_SRS_SPENDING_AMOUNT_VALID_DECIMALS*
Requirement statement	For spending command a valid amount should be a double number specified with 2 decimals. The decimals are separated by dot from the number.
Type	R
Safety	
Origin	*#REQ_FUN_CRS_SPENDING*

Identifier	*#REQ_DAT_SRS_SPENDING_DEFAULT_CURRENCY*
Requirement statement	Calling a spending command will write the amount in the wallet with the default currency RON.
Type	R
Safety	
Origin	*#REQ_FUN_CRS_SPENDING*

Identifier	*#REQ_DAT_SRS_SPENDING_TIME*
Requirement statement	Calling a spending command will write the amount in the wallet, and will specified the correct time of the transaction.
Type	R
Safety	
Origin	*#REQ_FUN_CRS_SPENDING*

Identifier	*#REQ_DAT_SRS_SPENDING_DEFAULT_SIGN*
Requirement statement	Calling a spending command will write the amount in the wallet, and will specified the sign of the amount with default sign for income “+”.
Type	R
Safety	
Origin	*#REQ_FUN_CRS_SPENDING*

Identifier	*#REQ_DAT_SRS_SPENDING_DEFAULT_CATEGORY*
Requirement statement	Calling a spending command will write the amount in the wallet, and will specified default category “salary” if the user didn’t specified an category.
Type	R
Safety	
Origin	*#REQ_FUN_CRS_SPENDING*

Identifier	*#REQ_FUN_SRS_SPENDING_SPECIFIED_CATEGORY*
Requirement statement	Calling a spending command, then the user can specified a category for spending. The correct command should contain spending command on first parameter, a category command (specified by –c or --category) with a type of category (Ex: “food”), and an amount. The correct command will accept both situations when category and amount positions are changed.
Type	R
Safety	
Origin	*#REQ_FUN_CRS_SPENDING*

BALANCE:

Identifier	*#REQ_DAT_SRS_BALANCE_COMMAND*
Requirement statement	The correct command for balance must contain the balance word in command after name of application.
Type	R
Safety	
Origin	*#REQ_FUN_CRS_BALANCE*

Identifier	*#REQ_DAT_SRS_BALANCE_WALLET*
Requirement statement	Balance- The user will obtain a balance from the wallet. The balance is calculated from all transactions found in the wallet. The wallet for creating the balance is obtained from the configuration file, reading the default wallet set in config.
Type	R
Safety	
Origin	*#REQ_FUN_CRS_BALANCE*

Identifier	*REQ_DAT_SRS_ADD_DECIMALS*
Requirement statement	Add Decimals – User must enter a double format amount (positive or negative) for create/income/spend commands. Amount should be entered with specified sign for positive or negative. If the amount is not entered with two decimals than the application will automatically add the necessary decimals until amount will have two decimals, or if the amount is entered with more than two decimals the application will automatically remove unnecessary decimals.
Type	R
Safety	
Origin	*#REQ_DAT _CRS_ADD_DECIMALS*

Identifier	*#REQ_DAT_SRS_REMOVE_ZEROES*
Requirement statement	Remove Zeroes – User must enter a double format amount (positive or negative) for create/income/spend commands. Amount should be entered with specified sign for positive or negative. If the amount is entered and has more than one zero at the beginning of the amount then the application automatically removes the unnecessary zeroes.
Type	R
Safety	
Origin	*# REQ_DAT_CRS_REMOVE_ZEROES*

Identifier	*#REQ_DAT_SRS_ONE_CONFIG*
Requirement statement	One Configuration File– The application use only one configuration file. The administrator will add in future the option to use more than one configuration file. The application use the configuration file to read information about how to configure the default wallet.
Type	R
Safety	
Origin	*#REQ_DAT_CRS_ONE_CONFIG*

Identifier	*# REQ_DAT_CRS_WALLET*
Requirement statement	Wallet- Each transaction operated from user will be stored in a wallet file, created by user. In wallet file first line will contain initial amount. For each transaction a new line will be implemented in wallet and has information about: transaction time, transaction type (income, spend), amount, transaction category, currency. The wallet is stored on user' PC
Type	R
Safety	
Origin	*#REQ_DAT_CRS_WALLET*

Identifier	*# REQ_DAT_SRS_MONEYTRACKER.CONFIG*

Requirement statement	MONEYTRACKER.CONFIG– All data contain in a moneytracker.config are store on a file. The purpose of monetracker.config file is to store default information needed for functional requirements.
	
	Example:
	`default_wallet = my.wallet`
	`default_currency = RON`
	`default_income_category = salary`
	`default_spending_category = other`
	`currencies = RON, EUR, USD`
	`rate_EUR_RON = 4.42`
	`rate_RON_EUR = 0.23`
	`rate_USD_RON = 3.92`
	`rate_EUR_USD = 1.13`

Type	R
Safety	
Origin	*# REQ_DAT_CRS_MONEYTRACKER.CONFIG*

5.8 Performance requirements

Identifier	*#REQ_PER _SRS_SYSTEM_DEPENDABILITY*
Requirement statement	System Dependability– The system dependability is given by the fault tolerance of the computer operating systems.
Type	R
Safety	
Origin	*#REQ_PER _CRS_SYSTEM_DEPENDABILITY*

Identifier	*# REQ_PER _SRS_HARD_DRIVE_SPACE*
Requirement statement	Hard Drive Space– The application's needs of hard drive space is meter in MB and must use no more than 10 MB. Administrator wish is that application's needs of hard drive space use in no more than 3 MB.
Type	R
Safety	
Origin	*# REQ_PER _CRS_HARD_DRIVE_SPACE*

Identifier	*# REQ_PER _SRS_RELIABILITY*
Requirement statement	Reliability – The reliability of the application is measured in the right result of a correct command line. Measurements obtained from 50 different valid commands. The application has a reliability of 100%.
Type	R
Safety	
Origin	*#REQ_PER _CRS_RELIABILITY*
Identifier	*#REQ_PER _SRS_AVAILABILITY*
Requirement statement	Availability– The System Availability of the application is given by the computer operating system. The application has no dependencies of internet connections. Measurements obtained from 100 hours of usage during testing. The application has a availability 100% of the time.

Type	*R*
Safety	
Origin	*# REQ_PER _CRS_AVAILABILITY*

Identifier	*#REQ_PER _SRS_RESPONSE_TIME*
Requirement statement	Response Time– The Application should response to any command in less than 2 seconds.
Type	R
Safety	
Origin	*#REQ_PER _CRS_RESPONSE_TIME*

5.9 Other requirements

Identifier	*#REQ_OTH _SRS_PORTABILITY*
Requirement statement	Portability– The System Portability of the application is given by the programming language used for the development of the application. The application is implemented in C++. The application will be adaptable on every operating systems platform.
Type	R
Safety	
Origin	*#REQ_OTH _CRS_PORTABILITY*

Identifier	*# REQ_OTH _SRS_USABILITY*
Requirement statement	Usability– The System Usability of the application is given by the easy to use property. A user should be able to learn the application in less than 12 hours.
Type	R
Safety	
Origin	*# REQ_OTH _CRS_USABILITY*

Identifier	*#REQ_OTH _SRS_EXTENDIBILITY*
Requirement statement	Extendibility– The application should be easy to extend. The code should be written in a way that favors implementation of new functions.
Type	R
Safety	
Origin	*#REQ_OTH _CRS_EXTENDIBILITY*

Identifier	*# REQ_OTH _SRS_TESTABILITY*
Requirement statement	Testability– The application should support testing. The code should be written in a way that allows testing, and has a high testability coverage.
Type	R
Safety	
Origin	*#REQ_OTH _CRS_TESTABILITY*

5.10 Appendix

To track your spending and income, you use a wallet, which starts with an initial amount. Each spending and income is recorded in this wallet, and one will be able to get some really smart aggregated data out of this, like:

- current balance
- balance at a specific time (in the past or future)
- totals for incomes and/or spending (for a period of time)
- list of operations in the wallet, filtered for a period of time

The following recording features are available:

- record spending or income
- recording might be done in a default currency or one specified in the command
- recording might be done with the current time or one specified in the command (it is possible to make a record in the past or future)
- each spending or income has a category. When recording this might be a default one, or one specified in the command line
- default values for currency and category for spending and income are specified in a configuration file.
- Conversion rates for currencies are specified in the configuration file

There are commands to change the configuration (including the file where the wallet is stored.)

The wallet file looks like this:

```
+1000.00 RON
1444216713;+;200.00;"salary advance";RON
1444218713;-;10.00;food;RON
1444296713;-;200.00;travel;EUR
```

The configuration file looks like this:

```
default_wallet = my.wallet
default_currency = RON
default_income_category = salary
default_spending_category = other
currencies = RON, EUR, USD
rate_EUR_RON = 4.42
rate_RON_EUR = 0.23
rate_USD_RON = 3.92
rate_EUR_USD = 1.13
```