Requirements Engineering
November 28, 2015

Andrei Fangli
Class 258

BillPath

Requirements Specification

1 Introduction

The purpose of this document is to offer both a high level view of the product in order to be able to grasp the general idea and a more detailed view to better understand the depth of its features.

The product aims to be an application that will help individuals keep track of their savings and analyze their income/expense ratios in order to help them save money. There are enterprise level applications that do this for organizations however there are not so many for the individual.

As a general overview, this documents starts by describing the need of this product and how it would fit in to the day to day routing that most if not all people are going through. After that a listing of features that will be available after the 1st release followed by the main target audience and general constraints.

Afterwards a more detailed view on the product requirements is illustrated followed by future evolution and a general release plan for the application.

2 General Description

2.1 Product Perspective

This application is for anyone who wishes to keep track of their incomes and expenses thus savings. One way of doing this is to keep all receipts and in one day simply sit at a table and sum them up, compare the total of expenses to a month's income and find out how economically money was spent. Some expenses may not happen every month (for instance one does not buy a TV every month) thus they may not be included in savings done for a month but rather in some other category of expenses. All this collecting and crunching of numbers can be automated and that is what BillPath wants to do.

In order to make it easy to keep track of income and expenses they should be memorized or saved right when they occur and not at a later date. This ensures that all transactions are stored when they occur (or at least in the same day) reducing the likelihood of miscalculations. To do this it must be made easy and part of the day to day routine. Most people in Europe have personal computers or tables that they use every day. Having an application where one can just click a few buttons and enter a small amount of details to keep track of what they bought seems more appealing than crunching the receipts.

At some point the salary is received and some calculations can kick off, however something less pleasing can happen. One can run low on money during the days before pay day, or may need to pay some loans to the bank before pay day and are wondering whether they have enough money to do all other expenses. Regardless of the reasons, the day when the salary is received is part of a short term plan to do savings when necessary. Having all expenses recorded in every day helps track the actual total balance (cash + bank accounts) one has which is useful in tight situations.

Last but not least the most important aspect when saving money to buy a new car is to establish an average of expenses over a period of time (e.g.: per month). Besides having the average there is also need for expense analysis which can lead to a more efficient way of saving money. This is where most of the usefulness of the application lies. Applications are able to store massive amounts of data, display it to the user in a fancy way that makes sense to them and most importantly do complex calculations and manipulations on that data really fast.

Any optimization plan starts with analyzing the current situation and determining what parts can be dropped to obtain the desired result faster and avoid any unwanted situations.

The aim of the product is to offer a free to use application that helps individuals keep track of their incomes and expenses.

2.2 Terminology

Amount – this is a decimal value which has an attached currency. E.g.: 4.12 EUR, -4 RON.

Transaction – this is the at the heart of the application. It represents an amount of money received or sent which is recorded within the application. A transaction has an amount, date when it was realized and

Income – this is a special kind of transaction in that it refers to amounts that have been received on a given date and have a description.

Expense – this is a special kind of transaction in that it refers to amounts that have been spent on a given date, have a description and belong to exactly one expense category.

Expense category – this is a grouping of expenses, they can be used to resemble separate accounts or just categories of expenses (e.g. food, clothing and so on). They have a name and a color. The name must be unique for each category while the color can repeat itself. The color is used for charts and in views that display expenses from multiple categories in order to help the user know to what category an expense belongs.

2.3 Product Functions

Here is a list of features that the application will primarily offer:

- Income management (viewing, adding, editing and removing incomes)
- Expense category management (viewing, adding, editing and removing expense categories)
- Expense management (viewing, adding, editing, and removing expenses)
- Creating Charts (Income vs Expense over a given period of time)

2.4 User Characteristics

In the introduction it was stated that the product targets individuals. To be more specific as not any individual is targeted but those that are employed and have at least a tablet or a personal computer.

The main age range is between 20 and 45 because people in this range often loan money from banks to buy a new car or to move to their place meaning it is a period where savings are more critical than at other ages. This does not mean that people outside this range are not to use this application, it means that they may not feed the need to use it.

2.5 General Constraints

For the 1st release the application aims to run only of personal computers and tablets running Windows 8.1 or newer or WindowsRT (e.g.: Microsoft Surface).

The application initially targets just Europe supporting two languages: Romanian and English. More languages will be supported as the product progresses.

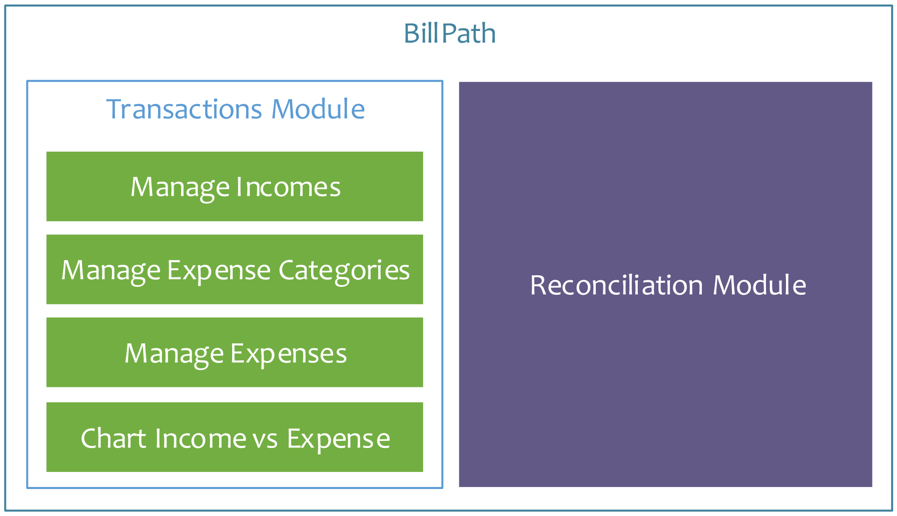
2.6 Assumptions and Dependencies

It is assumed that the readers of this document have elemental accountancy knowledge (knowing at least about currencies, transfer rates, debits, credits will be enough) as well as minimal knowledge about UML (mainly class diagrams as they are used to illustrate a high level view of the application domain model, no need in depth understanding, just the basics).

3 Product Details

3.1 System Architecture

Below is a diagram illustrating the requirements for the 1st release (Transactions Module) as well as a future module (Reconciliation Module):



3.2 System Requirements Definition

- *Infrastructure (nonfunctional requirements)*
    - *MVVM infrastructure*

      The recommended way of implementing application for Windows Store is by following the Model-View-ViewModel (MVVM) pattern. The framework has support for this which makes it faster to implement the application. This is a non-functional requirement that once implemented can be used for all other requirements.

  - *Pagination infrastructure*

    Viewing incomes and expenses is done through paged views. A more general component is required to implement pagination so it can be reused for both transaction types and even future items that will be displayed in a list.

- *Income management*
  - *View incomes*

  This will allow users to view incomes ordered by the date when they were realized in descending order in a paged view.
  - *Add income*

  This allows users to add new incomes, adding an income automatically updates the paged view that shows them.
  - *Edit income*

  This allows users to edit a previously added incomes, editing an income automatically updates the paged view that shows them.
  - *Remove income*

  This allows users to remove a previously added income, removing an income automatically updates the paged view that shows them.
- *Expense category management*

  Expenses are grouped by category. Each expense belongs to one category; this helps in grouping expenses in order to provide more context for what they were made.
  - *View categories*

  Categories can be viewed in a horizontal list each having a header part where general information is show (e.g.: total amount of contained expenses, the number of contained expenses and so on). Under the header a paged list of contained expenses is show, this is referred as the category content part.
  - *Add category*

  This allows users to add new categories that can be later used when adding or editing an expense. Adding a category automatically updates the list that shows them.
  - *Edit category*

  This allows users to edit previously added categories. Editing a category automatically updates the list that shows them.
  - *Remove category*

  This allows users to remove a previously created category. Removing a category automatically updates the list that shows them as well as removing all expenses within that category.
- *Expense management*
  - *View expenses*

  This will allow users to view expenses ordered by the date when they were realized in descending order in the content part of their corresponding category.
  - *Add expense*

  This allows users to add new expenses, adding an expense automatically updates the paged view that shows them.

  - *Edit expense*

    This allows users to edit a previously added expense, editing an expense automatically updates the paged view that shows them.

  - *Remove expense*

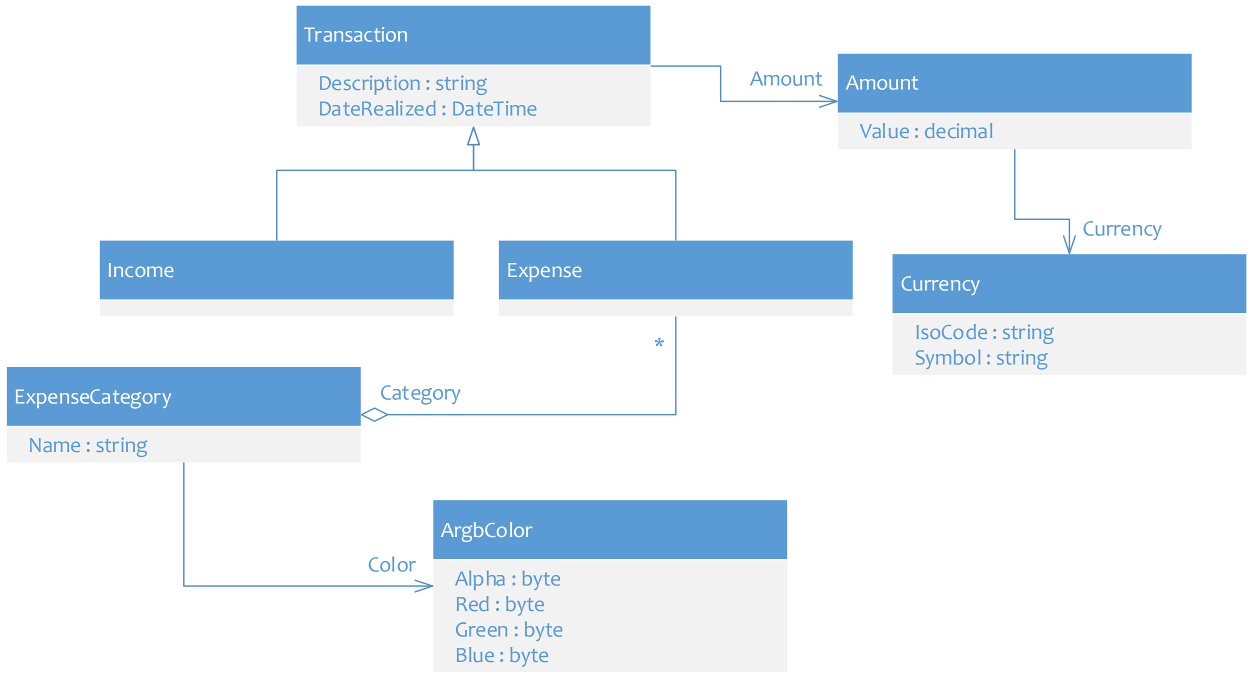
    This allows users to remove a previously added expense, removing an expense automatically updates the paged view that shows them.

- *Creating Charts (Income vs Expense over a given period of time)*

  This allows the user to create a chart where he/she can analyze the income vs expense for a period of time. The user also has the option of selecting what expense categories to include.

3.3 System Models

Below is an UML diagram illustrating the domain model used for the 1st release.



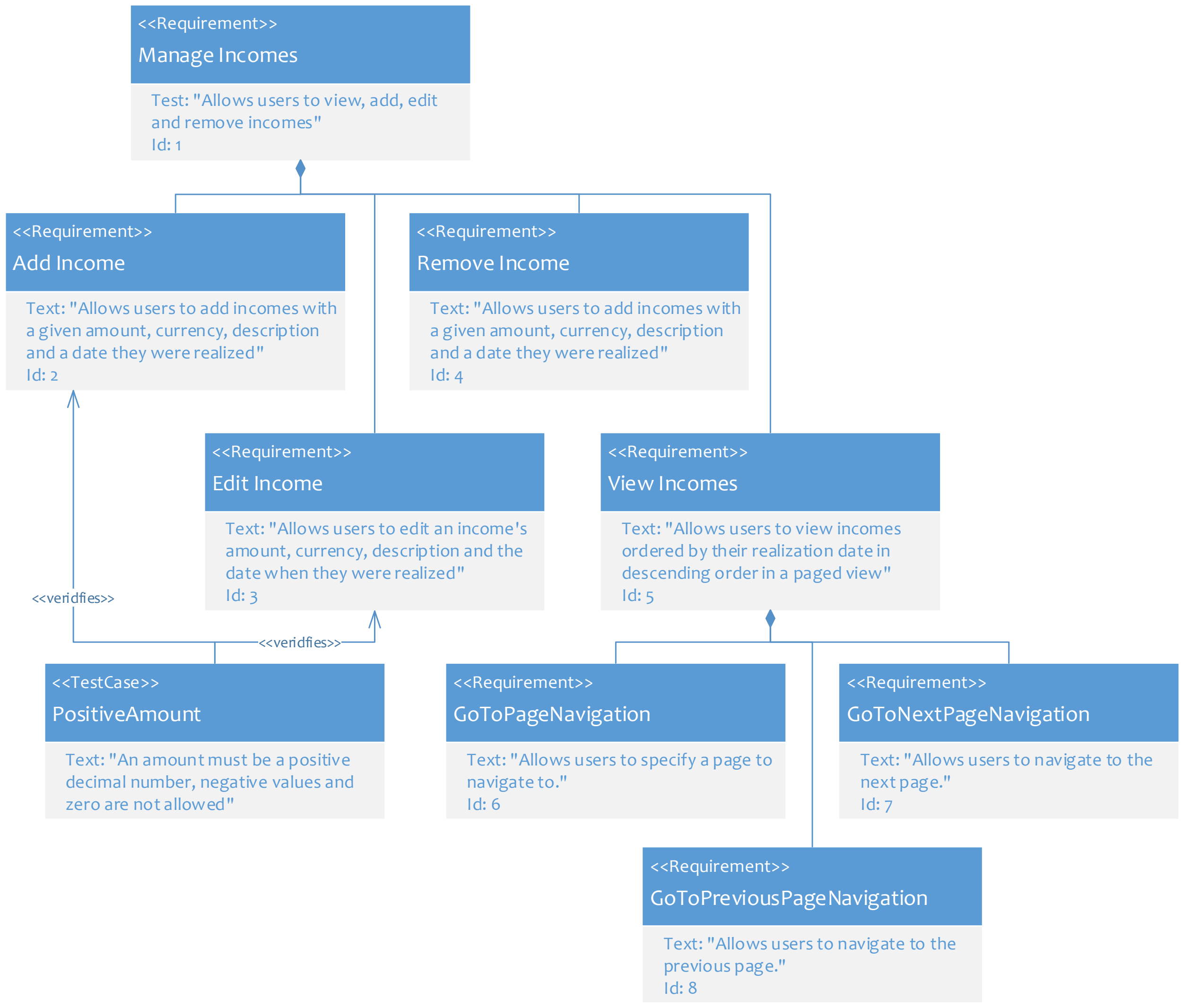
4 System Evolution

The application is being built using a "free to use" approach. The term is actually taken from the gaming industry where "free to play" games represent games that can be downloaded and installed without any charge, however if the user desires more content that is available in game he/she must pay for it. The same concept is applied here. The application will be available for download and installation for free however as modules are developed they are most likely going to require an in-app purchase to have access to them. This benefits the user as he/she will not require to pay a greater sum to get the features that they desire besides a few others that they will not be using. They will only pay for those exact features that they want resulting in a lesser expense for them.

The 1st release will include the mentioned features in this document for free. Users will be able to keep track of their incomes and expenses for free, even draw a chart to analyze their savings over a period of time.

The next module is the reconciliation module where users can verify whether they have been tricked by a seller. Reconciliations are a standard procedure in accounting as a means to verify whether the expected expense is the actual expense by comparing different sources that can claim the same expense (e.g.: shop receipt versus account extract from a bank). This can be helpful as a means to verify that the saved transactions within the application are valid, mistakes can happen at any time.

5 SysML Diagram


<<Requirement>>
Manage Incomes
Test: "Allows users to view, add, edit and remove incomes"
Id: 1
<<Requirement>>
Add Income
Text: "Allows users to add incomes with a given amount, currency, description and a date they were realized"
Id: 2
<<Requirement>>
Remove Income
Text: "Allows users to add incomes with a given amount, currency, description and a date they were realized"
Id: 4
<<Requirement>>
Edit Income
Text: "Allows users to edit an income's amount, currency, description and the date when they were realized"
Id: 3
<<Requirement>>
View Incomes
Text: "Allows users to view incomes ordered by their realization date in descending order in a paged view"
Id: 5
<<veridfies>>
<<veridfies>>
<<TestCase>>
PositiveAmount
Text: "An amount must be a positive decimal number, negative values and zero are not allowed"
<<Requirement>>
GoToPageNavigation
Text: "Allows users to specify a page to navigate to."
Id: 6
<<Requirement>>
GoToNextPageNavigation
Text: "Allows users to navigate to the next page."
Id: 7
<<Requirement>>
GoToPreviousPageNavigation
Text: "Allows users to navigate to the previous page."
Id: 8

6 SpecFlow Features

Feature: Add Expense

Scenario: Add first expense
  Given I have an empty Test category
  When I add an expense with 10 USD on 11/12/2015 for testing purposes in Test category
  Then I have 1 expenses in Test category
  And I have a total of 10 USD in Test category

Scenario: Add an expense to non-empty category
  Given I have a Test category
  And Test category contains a 10 USD expense
  And Test category contains a 20 USD expense
  And Test category contains a 30 USD expense
  When I add an expense with 22.2 USD on 11/12/2015 for testing purposes in Test category
  Then I have 4 expenses in Test category
  And I have a total of 82.2 USD in Test category

Scenario: Add an expense of different currency to non-empty category
  Given I have a Test category
  And Test category contains a 10 USD expense
  And Test category contains a 20 USD expense
  And Test category contains a 30 USD expense
  When I add an expense with 1.5 RON on 11/12/2015 for testing purposes in Test category
  Then I have 4 expenses in Test category
  And I have 3 expenses in USD in Test category
  And I have 1 expenses in RON in Test category
  And I have a total of 60 USD in Test category
  And I have a total of 1.5 RON in Test category

Feature: Edit Expense

Scenario: Edit an expense
  Given I have a Test category
  And Test category contains a 10 USD expense
  And Test category contains a 20 USD expense
  And Test category contains a 30 USD expense
  When I edit expense number 1 and set amount to 50 USD
  Then I have 3 expenses in Test category
  And I have a total of 100 USD in Test category

Scenario: Edit an expense of different currency
  Given I have a Test category
  And Test category contains a 10 USD expense
  And Test category contains a 20 USD expense
  And Test category contains a 30 USD expense

  And Test category contains a 40 USD expense
  When I edit expense number 3 and set amount to 30 RON
  Then I have 4 expenses in Test category
  And I have 3 expenses in USD in Test category
  And I have 1 expenses in RON in Test category
  And I have a total of 70 USD in Test category
  And I have a total of 30 RON in Test category

Feature: Remove Expense

Scenario: Remove an expense
  Given I have a Test category
  And Test category contains a 10 USD expense
  And Test category contains a 20 USD expense
  And Test category contains a 30 USD expense
  When I remove expense number 1
  Then I have 2 expenses in Test category
  And I have a total of 50 USD in Test category

Scenario: Remove an expense of different currency
  Given I have a Test category
  And Test category contains a 10 USD expense
  And Test category contains a 20 USD expense
  And Test category contains a 30 RON expense
  And Test category contains a 40 USD expense
  When I remove expense number 3
  Then I have 3 expenses in Test category
  And I have 3 expenses in USD in Test category
  And I have 9 expenses in RON in Test category
  And I have a total of 70 USD in Test category
  And I have a total of 0 RON in Test category

Feature: Add Expense Category

Scenario: Add first category
  Given I have no categories
  When I add an expense category named Test and color Red
  Then I have 1 expense categories
  And I have a total of 0 USD in Test category

Scenario: Add an expense category while others exist
  Given I have a FirstTest category
  And FirstTest category contains a 10 USD expense
  And FirstTest category contains a 20 USD expense
  And FirstTest category contains a 30 USD expense
  When I add an expense category named SecondTest and color Red

Then I have 2 expense categories
And I have a total of 60 USD in FirstTest category
And I have a total of 0 USD in SecondTest category
And I have a total of 60 USD across all categories

Scenario: Add an expense category while others exist in different currencies
Given I have a FirstTest category
And FirstTest category contains a 10 USD expense
And FirstTest category contains a 20 USD expense
And FirstTest category contains a 30 USD expense
When I add an expense category named SecondTest and color Red
And I add an expense with 3 RON on 10/2/2013 for testing purposes in SecondTest category
Then I have 2 expense categories
And I have a total of 60 USD in FirstTest category
And I have a total of 0 RON in FirstTest category
And I have a total of 0 USD in SecondTest category
And I have a total of 3 RON in SecondTest category
And I have a total of 60 USD across all categories
And I have a total of 3 RON across all categories

Feature: Edit Expense Category

Scenario: Edit an expense category
Given I have a FirstTest category
And FirstTest category contains a 10 USD expense
And FirstTest category contains a 20 USD expense
And FirstTest category contains a 30 USD expense
And I have a SecondTest category
And SecondTest category contains a 1 USD expense
And SecondTest category contains a 2 USD expense
And SecondTest category contains a 3 USD expense
When I edit SecondTest category and rename to RenameTest
Then I have 2 expense categories
And SecondTest category does not exist
And I have a total of 60 USD in FirstTest category
And I have a total of 6 USD in RenameTest category
And I have a total of 66 USD across all categories

Scenario: Edit an expense category containing expenses in different currency
Given I have a FirstTest category
And FirstTest category contains a 10 USD expense
And FirstTest category contains a 20 RON expense
And FirstTest category contains a 30 USD expense
And I have a SecondTest category
And SecondTest category contains a 1 RON expense
And SecondTest category contains a 2 USD expense
And SecondTest category contains a 3 RON expense

When I edit SecondTest category and rename to RenameTest
Then I have 2 expense categories
And SecondTest category does not exist
And I have a total of 40 USD in FirstTest category
And I have a total of 20 RON in FirstTest category
And I have a total of 4 USD in RenameTest category
And I have a total of 2 RON in RenameTest category
And I have a total of 44 USD across all categories
And I have a total of 22 RON across all categories

Feature: Remove Expense Category

Scenario: Remove an expense category
Given I have a FirstTest category
And FirstTest category contains a 10 USD expense
And FirstTest category contains a 20 USD expense
And FirstTest category contains a 30 USD expense
And I have a SecondTest category
And SecondTest category contains a 1 USD expense
And SecondTest category contains a 2 USD expense
And SecondTest category contains a 3 USD expense
When I remove SecondTest category
Then I have 1 expense categories
And SecondTest category does not exist
And I have a total of 60 USD in FirstTest category
And I have a total of 60 USD across all categories

Scenario: Remove an expense category containing expenses in different currency
Given I have a FirstTest category
And FirstTest category contains a 10 USD expense
And FirstTest category contains a 20 RON expense
And FirstTest category contains a 30 USD expense
And I have a SecondTest category
And SecondTest category contains a 1 RON expense
And SecondTest category contains a 2 USD expense
And SecondTest category contains a 3 RON expense
When I remove SecondTest
Then I have 1 expense categories
And SecondTest category does not exist
And I have a total of 40 USD in FirstTest category
And I have a total of 20 RON in FirstTest category
And I have a total of 40 USD across all categories
And I have a total of 20 RON across all categories

7 Requirements prioritization

At the moment there is only one stakeholder for the project making the ranking prioritization technique appropriate as the project itself is not very complex and does not require thorough analysis.

Below is a table illustrating the functional requirements with their rank and cost:

Requirement	*Rank*	*Time (days)*	*Budget cost*
Income management (add)	1	10	19%
Income management (update)	2	3	8%
Income management (remove)	3	3	8%
Expense category management (add)	4	3	10%
Expense category management (update)	5	3	8%
Expense category management (remove)	6	3	8%
Expense management (add)	7	3	10%
Expense management (update)	8	3	8%
Expense management (delete)	9	3	8%
Income vs Expense Chart	10	5	13%

Based on the table above the iterations are as follows:

1. *Income management*: add and update (cost 27%).
2. *Income management*: remove, *expense category management*: add, update and remove (cost34%).
3. *Expense management*: add, update and delete, *income vs expense chart* (cost 39%).