Daniel Luján Villarreal {dluj@itu.dk}

Project Report

T16 Smartphone E-Business Application Development

Application: FastMgmt

1. App Idea - Fast Financial Management

The idea for this project comes from a night out in the city. **FastMgmt** is an application that stores and tracks your expenses in a fast and easy way, making you aware when you are over a desired amount.

App in general

- ➔ The application works fast and does not have too many clicks to capture the information from the user
- ➔ This application is not intended to calculate averages, project expenses, connect to any bank account or have sensitive personal information
- ➔ The user can change the amount to track, the tracking period, the starting time of tracking or the use of categories at any point.
- ➔ The application warns the user when adding an expense results in going over the amount to track, but it is still possible to add more expenses.
- ➔ Once the tracking period has expired it is not possible to add expenses. The user has to go into the settings and start a new tracking period.

Main Screen

In the main screen of the application it is displayed the amount and the time you selected to track, it also displays what you have spent. If categories are being used,
the user can add new categories by clicking the “New Category” button. After adding a new category it can be clicked to add new expenses in that category. If categories are not being used, then the user can just click the “Add Expense” button and add one. When the already spent amount is displayed, if it is above the amount to track its color changes to red.

Settings

In the settings the application is configured for the start of the tracking period, the time period to track, the amount to track and the use of categories. The tracking period can start at the exact moment of saving the settings or at the beginning of the selected time.

The time options to track expenses can be during the present day, week or month. Another option for track the expenses is the use of categories. When using categories it is up to the user to manage the categories and make good use of them.

Categories

Using categories means that, as previously stated, in the main screen of the application the user has the option to add categories and click one to add an expense. If no categories have been added then no expenses can be added. The use of categories helps the user to learn about spending patterns and keeping track of the big expenses.

In case the user changes the options for the use of categories once the tracking has started two things can happen:

- ➔ Use Categories to No Categories: the expenses for all the categories are mashed up in one and categories are deleted
- ➔ No Categories to Use Categories: the expenses without category are mashed up in one generic category called “No Category Expenses”

Categories can also be managed and the user has two options:

- ➔ Rename Category: the user can rename the category without modifying the already assigned expenses to that category
- ➔ Delete Category: the user can remove a category. In case the removed category has assigned expenses, this expenses are deleted and will not show up in the already spent amount.

PhoneGap

This application is built using the PhoneGap framework. PhoneGap is a web-based mobile development framework. This means that the applications are built using web technologies like HTML5, CSS and Javascript which allows cross-platform development. The framework supports development for the main mobile platforms including Android, IOS and Windows Phone.(7 or 8). Using the PhoneGap framework is a big advantage when developing applications like FastMgmt that don’t require platform specific capabilities, although the framework has several API’s to cover some of the major ones like access to accelerometer, camera and storage. The main advantage of having this cross-platform development possibility is that the developer can do a generic application and when wanted compile the application for specific platforms. This project will be available in the Android and IOS platforms.

2. Business Case/Value Proposition

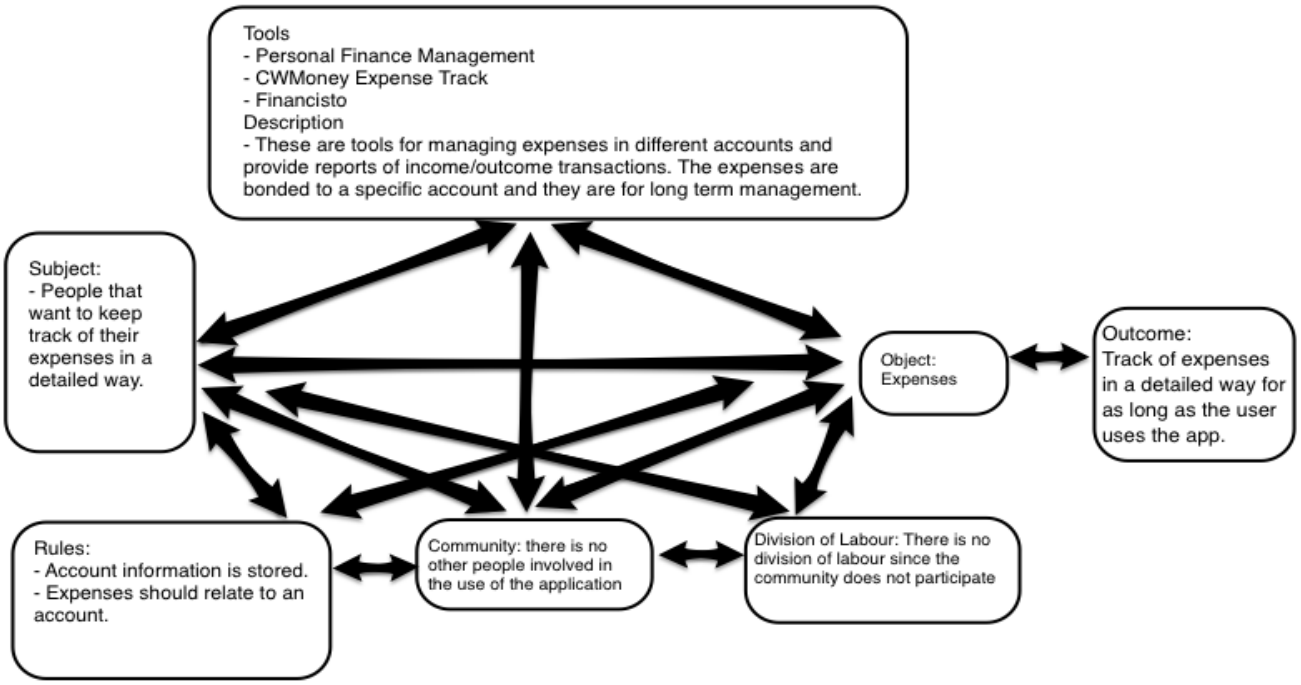
FastMgmt provides a fast way to help the user keep track of expenses over a short period of time (daily, weekly or monthly). With the development of new forms of payment, like NemPay, and quick checkouts the customer can easily lose track of money spent over a short period of time.

With this comes the need for a personal finance manager, but the existing applications need too many clicks, they store more information than you need or you have to grant access to your bank account. Banks applications give the user an overall look of their expenses, like restaurants or entertainment, but sometimes tracking the things you spent in those places is more important, not just the overall amount.

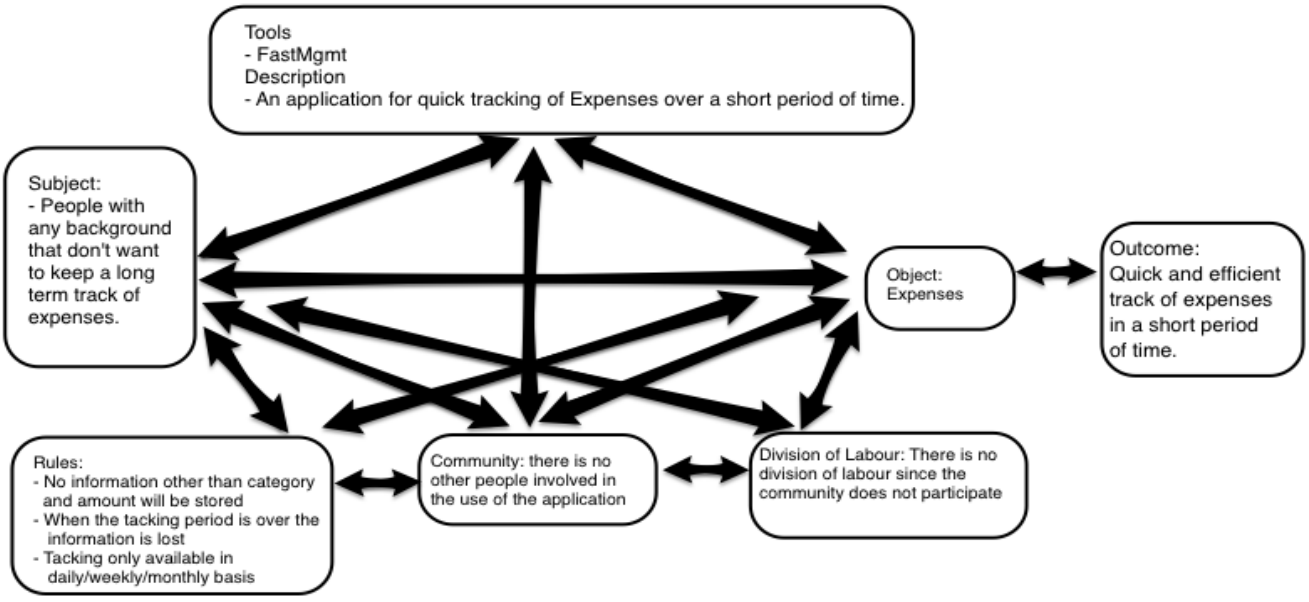
FastMgmt is for people that do not care about past or future expenses. The only thing they need is to keep track for a short period of time of how much money they have spent without storing personal/sensitive data. Also, the user has free choice of which expenses to track and which not. This application is for the user that wants an easier and faster way of tracking expenses.

3. Activity Diagram

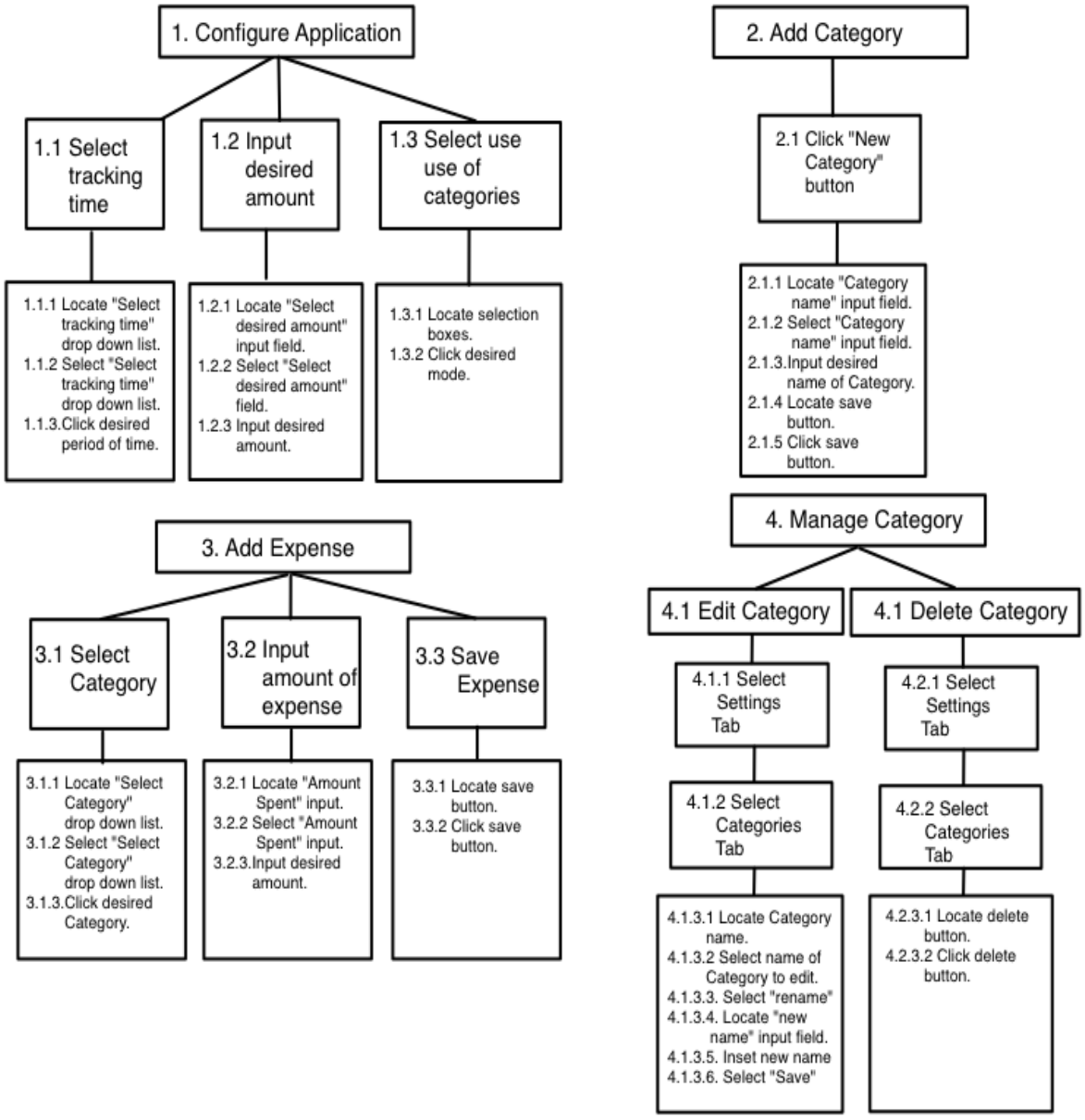
"As Is" Activity Diagram



"Will be" Activity Diagram



4. User Tasks Definition

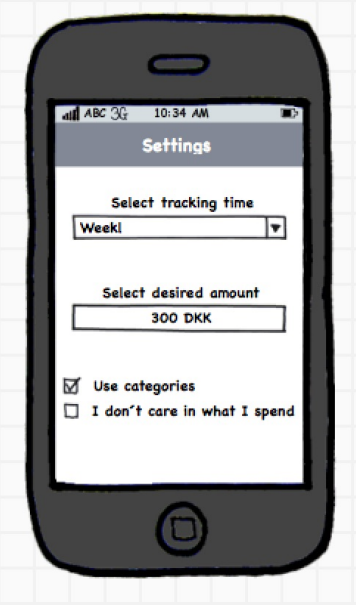


5. Design Sketches of Look & Feel and Page Navigation

User Interface

Home

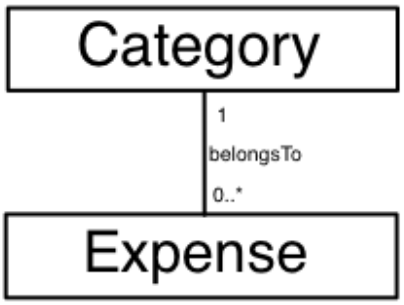
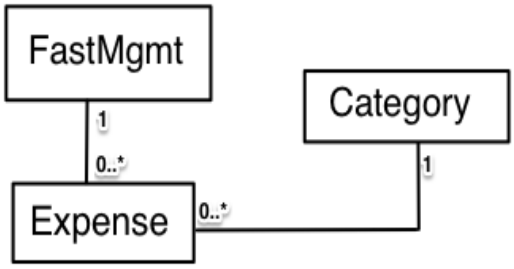
Settings



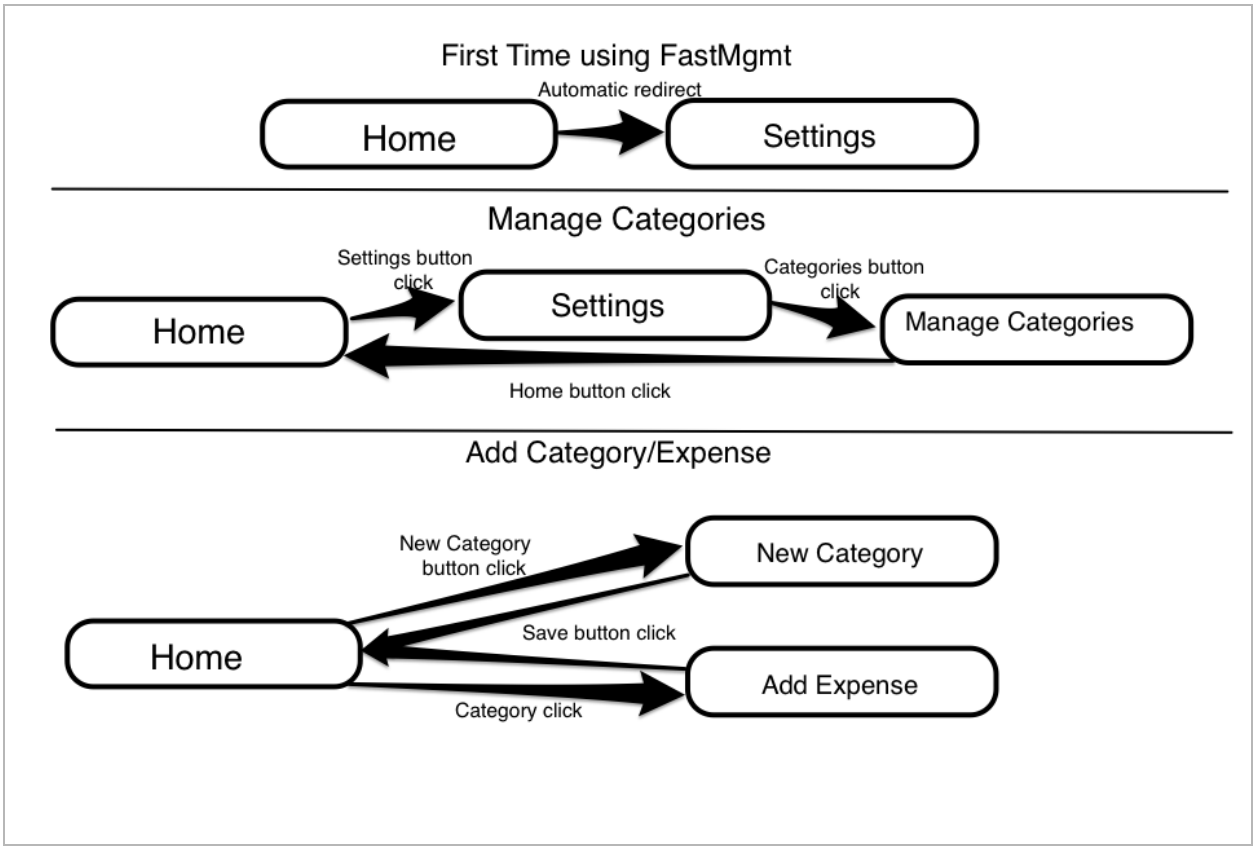
Add Expense

Add Category

Class Diagram and Domain Model

Class Diagram	Domain Model
	

Page Navigation



6. User Interface and User Experience

Discussion of Key Code Snippets

Color of Spent amount

A very simple code snippet but still important for the user interface is the one that changes the color of the “spent” amount in the main screen. This gives the user the possibility to, without giving a very detailed look at the number, realize that the spent amount is over the desired tracking amount and either stop spending or slow down.

```
function loadAmounts(permanentStorage){
    var amount = permanentStorage.getItem("amount");
    var spent = permanentStorage.getItem("spent");
    if(amount != "" && amount != null){
        $("#ipt_selectedAmount").val( amount +" DKK");
    }else{
        $("#ipt_selectedAmount").val("0 DKK");
    }
    if(spent != "" && spent != null){
        $("#ipt_leftAmount").val(spent + " DKK");
    }else{
        $("#ipt_leftAmount").val("0 DKK");
    }
    if(parseInt(spent) > parseInt(amount)){
        $("#ipt_leftAmount").css("color","#ff4400");
    }
}
```

“Div” attribute for dialog display

Thanks to the flexibility of JqueryMobile it is very easy to display complete pages in different ways. One of those ways is the dialog format. Just by adding the attribute “data-role=’dialog’” to the main div JqueryMobile displays this file as a pop-up dialog. After adding this simple attribute, the only thing left to do is to call it as if it was another file with the anchor (<a href=”path/to/file.html”>) element.

```
<div data-role="dialog" id="settings" data-close-btn="right">
<div data-role="header" data-position="fixed">
        <h1>Add Expense</h1>
    </div><!-- /header -->
```

Device width

One of the biggest problems when working with mobile devices and web sites is to control the size of the elements. As with web browsers, the web-based mobile application can also have design flaws and display the wrong data at the wrong place, things can be out of proportion or the screen size too small for the design of the application. By just using the 5th line in the header of the application a lot of troubles go away. Adding that to using the JqueryMobile Framework everything virtually disappears. When using JqueryMobile widgets and elements they are automatically resized to fit perfectly in the screen, so you have a very smooth looking application that looks just the same as in the web browser. The developer just has to pick the colors and the styles for the application, but not the sizes.

```
3  <head>
4      <title>My Page</title>
5      <meta name="viewport" content="width=device-width, initial-scale=1">
6      <!-- jquerymobile -->
7      <link rel="stylesheet" href="../css/jquery.mobile-1.3.2.min.css" />
```

Home button on Settings screen

The application has a hidden setting that determines if this is the first time of the user with the application, the same setting applies for when the tracking time has expired, and it is useful for several things and one of them is the ability of the user to get out of the Settings screen. To have a better control of the flow of the application, and to lower the possibility of unhandled errors, the home button is displayed in the Settings screen if the present time is not the first time of the user, and also if the time to track has expired and the user has not set the application to reuse it. This way, the user has to prepare for another use before going back to the main screen and don't get a weird behaviour from the application.

```
6   if(!first){
7       var endOfTracking = new Date(permanentStorage.getItem("endOfTracking"));
8       var dateNow = new Date();
9       if(dateNow > endOfTracking){
10          window.localStorage.clear();
11          permanentStorage.setItem("spent",0);
12          permanentStorage.setItem("expenses-expenses",0);
13          permanentStorage.setItem("numberCategories",0);
14          window.location.replace('settings.html');
15      }
16      $(".ui-header").append("<a href='../index.html' class='ui-btn-right' data-ajax='false' " +
17              "data-corners='false' data-theme='c' data-role='button' data-icon='home'>Home</a>").trigger("create");
```

7. Data Handling

Discussion of Key Code Snippets

Get Settings

The most important code snippet of all is the “getSettings” functions and all the modules run this function before everything else when called. The function can be divided in three parts.

Loading data. At the beginning of the function it can be seen that the local storage is acquired. The interface “window.localStorage” implements the W3C’s Web Storage which can save persistent data using key-value pairs. This is used to save all the information relevant for the use of the application like tracking times, amount to track and expenses for every category.

First time use or Expired tracking time. As seen in the code snippet, if the “firstTime” flag is null, meaning it hasn’t been set, the application stores the values that always have to be present, without this values the application will not behave in the correct way, and it also redirects the user to the settings module. This flag is null in two occasions, when the user opens the application after a fresh install or when the user goes into the Settings module after the tracking time has expired. In both occasions the “Home” button is hidden in the Settings module forcing the user to start another tracking period.

Load Data. If the present time is not the first time of the user it means that the user is has set the application for a tracking period and the settings are loaded. The three called methods are very descriptive in their names and as seen in the code snippet the categories, the tracking amount, spent amount and the tracking time are displayed in the screen.

```
14 function getSettings(){
15     var permanentStorage = window.localStorage;
16     var firstTime = permanentStorage.getItem("firstTime");
17     if(firstTime == null){
18         permanentStorage.setItem("spent",0);
19         permanentStorage.setItem("expenses-expenses",0);
20         permanentStorage.setItem("numberCategories",0);
21         window.location.replace("settings/settings.html");
22     }else{
23         permanentStorage.setItem("firstTime", false);
24         loadCategories(permanentStorage);
25         loadAmounts(permanentStorage);
26         loadTrackingTime(permanentStorage);
27     }
28 }
```

Loading Categories

As seen in the previous code snippet, if the user has set the application for a tracking period the function “loadCategories” is called. The important thing about this code snippet is the first 6 lines of the function. If the tracking period has expired, instead of automatically redirect the user to the Settings module to set another tracking period, the containers that let you add

expenses are deleted. It does not matter if the user is using categories or not, when this containers are deleted there is no way for the user to keep adding expenses and the only available button is the “Settings” button. This allows the user to see the outcome of the last tracking period, and allows the application to have tighter control of its own flow. It is important to repeat that if the user enters the Settings module once the tracking period has expired, the only thing that can be done is set another tracking period.

```
30 function loadCategories(permanentStorage){
31     var endOfTracking = new Date(permanentStorage.getItem("endOfTracking"));
32     var dateNow = new Date();
33     if(dateNow > endOfTracking){
34         alert("Date Overdue. Go to settings and start tracking again!");
35         $("#fc_categories").remove();
36         $("#btn_addCategory").remove();
37     }else{
```

Add Category

The next code snippet is run when the “Save” button in the “Add Category” pop-up is clicked. The importance of this snippet is in the lines 183, 184 and 186. This lines show how the expenses and the categories are handled. Since the Web Storage works with key-value pairs a string structure is applied to manage this values.

Category Name. The names of the categories are stored with the key “category_numberOfCategory” and the value is the name of the category. This allows the application to iterate through all the names of the categories and display them in screen.

Category Expenses. The expenses for each category are stored with the key “expenses-category_numberOfCategory” and the value is the assigned spent amount in the category. As with the category name, this string structure allows the application to iterate through all the categories’ expenses to display them in screen.

Number of Categories. The string structure of the keys’ in this pairs allows to iterate over them, but it is important to store the number of categories in use. Without this number there is no easy way to keep tight control of them and the logic behind the actions involving categories, basically all, would be far more complex.

It is important to mention that if the user has selected the option to not use categories, all the expenses are stored in one generic category but the “Add Category” button in the main screen is hidden so there is no possibility to add categories.

```
function onClickAddCategory(){
    var permanentStorage = window.localStorage;
    var category = $("#name").val();
    category = category.replace(" ","_");
    if(category == "" || category == "Name..." ){
        alert("Enter name of category first!");
    }else{
        var numberCategories = parseInt(permanentStorage.getItem("numberCategories"));
        permanentStorage.setItem("category_"+numberCategories,category);
        permanentStorage.setItem("expenses-category_"+numberCategories,"0");
        var newNumber = numberCategories + 1;
        permanentStorage.setItem("numberCategories",(numberCategories+1));
        alert("Category "+ category.replace("_"," ") +" added!");
        window.location.replace("../index.html");
    }
}
```

Switch from Use Categories to No Categories, or viceversa

This code snippet is important for the correctness of the displayed data during the tracking period. The applications allows the user to switch from using categories to not use categories, or the other way around. This presented the challenge of how to keep the spent amount equal at all times while switching the use of categories back and forth. To solve this challenge the next two scenarios where applied.

Use of categories to No categories. If the user changes the settings from using categories to not use categories the change is fairly easy. As mentioned before, even when the user selects to not use categories all the expenses are stored in one generic category called “expenses-expenses” so an iteration through all the categories is done and the spent amount for each category is summed up and stored in this generic category. After the expenses for each category is summed, the category is deleted.

No categories to Use Categories. This case can’t be seen in the code snippet for space purposes, but it is very similar to the previous scenario. When the user changes the settings from not using categories to use categories the application simply takes the expenses for the generic category “expenses-expenses” and adds a new generic category named “No Category Expenses” and stores the amount there. After this is done the “expenses-expenses” category amount is set to zero. With this new generic category the user can still add expenses in it or add new categories while the data still makes sense and remains the same.

```
}else{ // switched from yes categories -> no categories
    var numberCategories = parseInt(permanentStorage.getItem("numberCategories"));
    var expenses = 0;
    for(var i=0; i< numberCategories; i++){
        var category_expenses = parseInt(permanentStorage.getItem("expenses-category_"+i));
        expenses = expenses+ category_expenses;
        permanentStorage.removeItem("expenses-category_"+i);
    }
    permanentStorage.setItem("numberCategories",0);
    var expenses_expenses = parseInt(permanentStorage.getItem("expenses-expenses")) + expenses;
    permanentStorage.setItem("expenses-expenses",expenses_expenses);
}
```

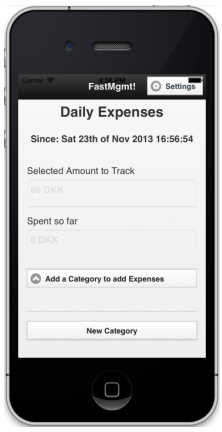
8. Evaluation

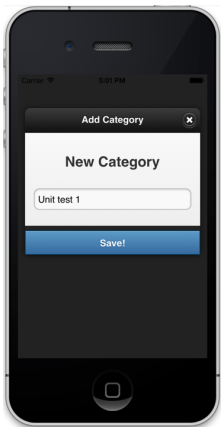
Software Testing

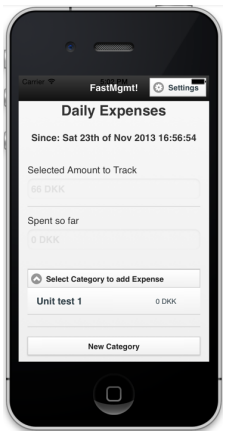
For space purposes only four unit tests will be shown for the software testing. All screenshots have been made with an iOS simulator.

Add Category

This test is for the assessment of adding a category. The first screen is the application right after saving the settings. The user clicks the button “New Category” to add a category and a popup window shows to enter the name of the category appears as seen in the second screen. After clicking the “Save!” button the category appears in the main screen, in this case “Unit test 1”.

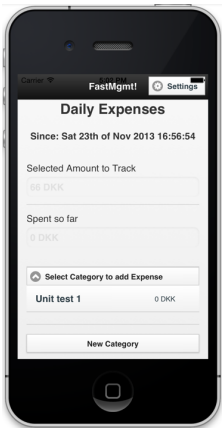


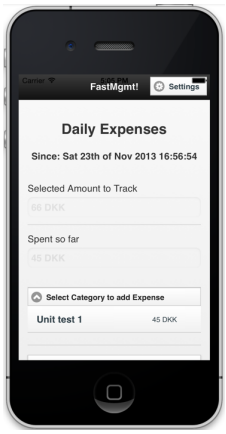




Add Expense in Category

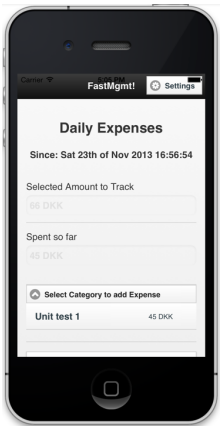
This test is for the assessment of adding an expense in a category. The first screen shows a category without expenses. To add an expense the user clicks on the name of the category and a popup window shows to enter the amount of the expense as seen in screen 2. After clicking the “Save!” button the expense appears besides the name of the category and in the “Spent so far” field, in this case 45.

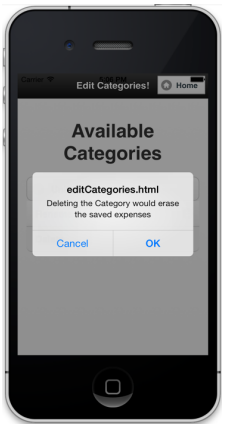


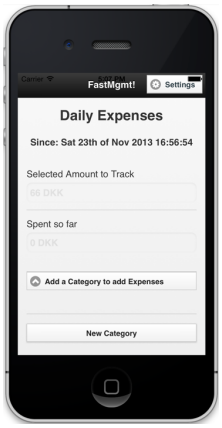


Delete Category

This test is to assess the deleting of a category. The first screen shows the main screen with the category created for the first test. The user goes to the Categories module and clicks on “Delete” category. a popup window is shown to let the user know that the expenses related to this category will be deleted. The third screen shows the main screen without the previously deleted category and no expenses.

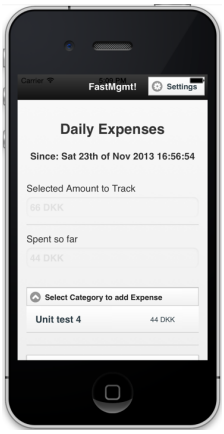


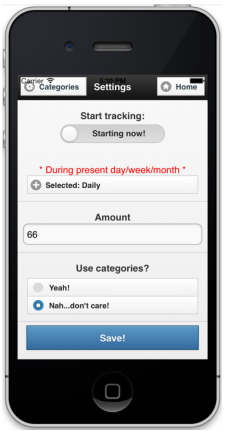


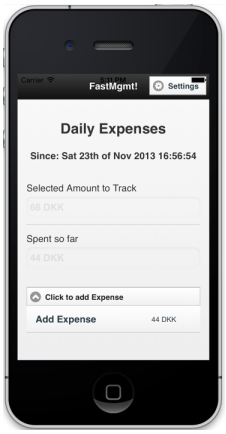


Switch from Use Categories to Not use Categories

This test is for the assessment of switching from using categories to not use categories. The first screen shows a category with an amount. The user goes to the Settings module and changes from use categories to not use categories. After clicking the “Save!” button the main screen shows the button “Add Expense” with the same expenses as with using categories and no “New Category” button.







User Testing

For the user testing of the application the final prototype was given to 8 people (friends and colleagues) and they were asked to do some tasks in the following order:

1. Set a new tracking period using categories
2. Add new category
3. Add expenses in that category until going over the amount to track
4. Go to Settings, insert a higher amount to track and change settings to not using categories.
5. Add a final expense to exceeded the amount to track.
6. Rename the category
7. Delete the category

Overall the tests went smoothly and the users gave a positive feedback about the application, some even started using it to track expenses over the weekend. The negative feedback was in small quantity but still very useful because it was mainly to clarify things that were not clearly defined in the user interface. This changes are explained next.

Add pop up when going over amount. Before the user testing, when the user inserted an expense that made the “Spent” amount to go over the tracking amount the only thing that happened was the change of font color for the “Spent” amount in the main screen. Some users suggested to alert them in a clearer way that they had gone over the tracking amount. This was done by adding a pop-up dialog when the “Spent” amount went over the tracking amount. The advantage to have this pop-up dialog is that the application stops and waits until the user clicks “ok” in such dialog, making the user more aware of the expenses.

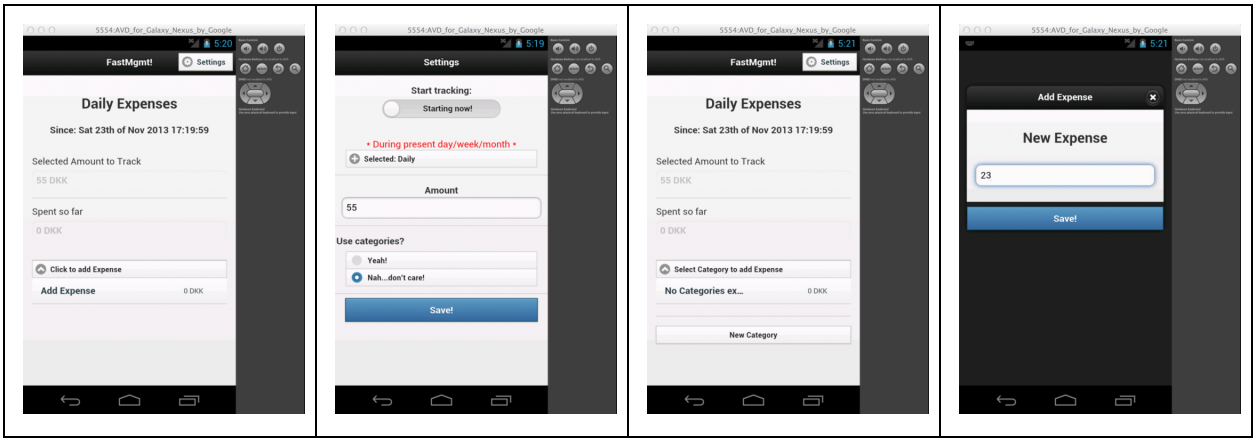
Add date in main screen. Before the user testing, the user had no way of checking the start of the tracking period, the only thing presented in screen was the selecter period to track (day/week/month). In response to some suggestions from the users the date of the start of the tracking period is being displayed in the main screen.

*** During present day/week/month *.** When the users had little information about the application some things were not fully intuitive. One of this being the option for selecting the tracking time when selecting the settings. Almost every time this option had to be explained to the users. In response to this behaviour the banner * During present day/week/month * was added above the options to select. This banner helps this option to be more intuitive helping the user to understand it better. To select this banner different options were presented to the users during the ongoing tests and it was the most popular.

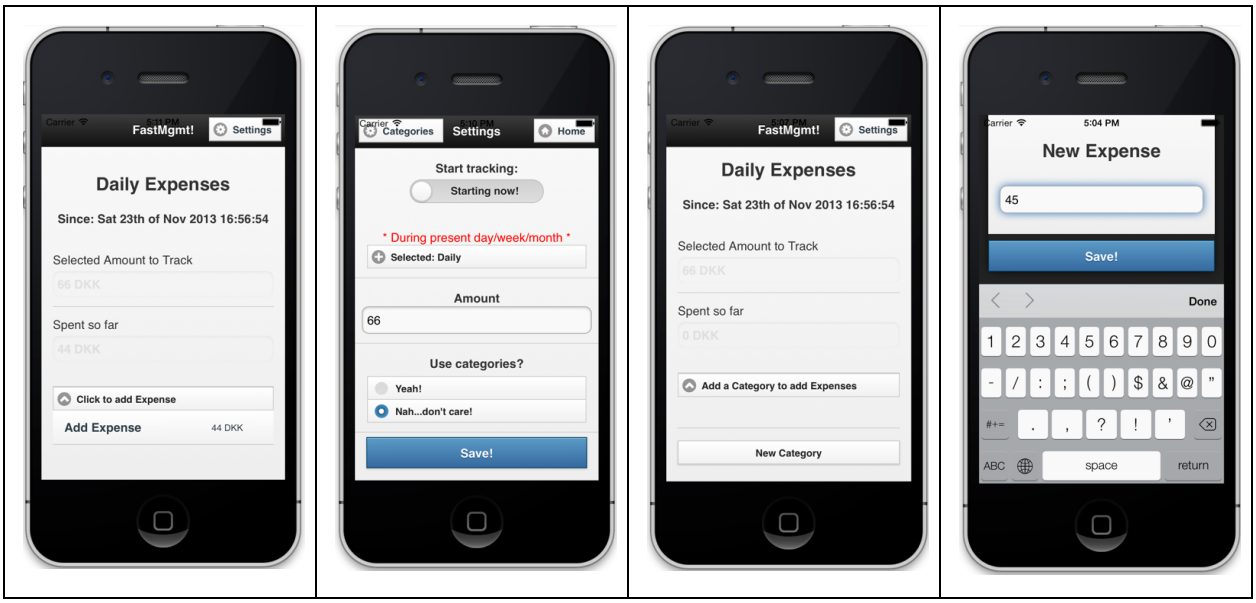
Automatic numeric keyboard. Before the tests the inputs for the amounts were set as text inputs. Some users asked if it was possible to make the numeric keyboard automatically appear. After doing this change on the application one things was noted. When setting an input to be numeric instead of text the field does not accept default values, that is, if the user has already selected an amount to track in the settings, goes out of the Settings module and returns, this field will be empty again. After noticing this behaviour the users that suggested this change were contacted and asked if they considered more important to have the numeric keyboard than the previously inserted amount and the responses were positive so the change prevailed.

9. Screenshots

Android



iOS



10. Project code base

https://github.com/dluj/fastMgmtAll.git

See steps to run the application on each available platform in the appendix.

Appendix

Steps to run application

1. Download the code from Github in https://github.com/dluj/fastMgmtAll.git
2. Platforms:
    a. Android
        i. Open Eclipse ADT Bundle.
        ii. Go to File->Import->Existing Android Project Into Workspace
        iii. Select root directory of the project. This can be the main folder or the folder path/to/app/fastMgmtAll/FastMgmt/platforms/android
        iv. Click Finish
    b. IOS
        i. Open Xcode
        ii. Go to File->Open
        iii. Select file path/to/app/fastMgmtAll/FastMgmt/platforms/ios/FastMgmt.xcodeproj