POSEiDON

PersOnalized Smart Environments to increase Inclusion of people with DOwn's syndrome

Report 42

Healthy Eating App

Development Documentation

Call:	FP7-ICT-2013-10
Objective:	ICT-2013.5.3 ICT for smart and personalised inclusion
Contractual delivery date:	not applicable
Actual delivery date:	December 2016
Version:	v1
Editor:	Silvia Rus (FhG)
Contributors:	Patrick Schmitt (FhG)
Reviewers:	
Dissemination level:	Public
Number of pages:	31

Contents

1 Introduction

As agreed in the last review, the "Healthy Food" App was developed. This application show-cases the usage of the framework, describing how this supports possible future developers.

We start with an overview of the development process and present afterwards a task-related detailed description. Each chapter corresponds to a main task and presents a few subtasks. The total development time is 59 days. From these, roughly 40 % represent pure implementation time. The other 60% were used for requirement gathering and app design.

Task	Subtasks	Required time [days]
Requirement analysis	Ideas gathering	10
	Idea review	4
	Methodical approach	5
	Functional view	3
	Subtotal	**22**
App design	App behaviour and look (also implementation)	*9*
	Communication with file server (also implementation)	*2*
	Add data model	1
	Context awareness	10
	Subtotal	**23**
Implement ation	Login, up- and download App	*4*
	Food Creator App	*2*
	Healthy Eating App	*8*
	Subtotal	**14**
	TOTAL	**59**

The two apps, one aimed at the PU and one at the SU, were designed by a developer which worked on the project as part of an internship required at the Computer Science course of the University of Applied Sciences in Darmstadt. He previously had no experience in App development, however has a strong basis in Java development.

The requirement analysis has shown many possible ways of addressing the subject of healthy eating for persons with Down syndrome. The developer has chosen the final functionality of the apps and planned the implementation work. As shown in the functional view, see Chapter 1, the tasks are prioritized. All tasks prioritized with HIGH were developed in the scope of his internship, and are featured in this report. Medium and low priority tasks will be implemented in his bachelor thesis, along with the evaluation of the app.

In the following chapters, Chapters 2-4, the developer has kept track of the different tasks and their different sub activities. For each activity, the already available documents he used are listed. For each activity result, eventual difficulties and the required time to fulfil the activity are recorded. These chapters are edited in form of a journal, from developer perspective.

In Chapter 5 we present the next steps regarding implementation and user evaluation, while in the last chapter, Chapter 6, of the document we present a quick developer guideline. This is made up from links to used documents and used code snippets, as well as the developed Starter App.

2 Requirement analysis

The requirement analysis phase is comprised of multiple steps which include gathering the app ideas on the topic of healthy food, refining the ideas, which includes talking to the primary user group and to expert secondary users, identifying potential contexts. While finalizing the functionality of the app different app descriptions, fitting different perspectives are created, in form of an app walkthrough and a functional view of the app.

2.1 Idea gathering

At the beginning of the project I gathered ideas about the functionality of the healthy eating app. I needed to know in which life situations the app can help the primary user to be more independent. For this, I engaged in email communication with the POSEIDON Developer and Carer Community. In this way, I gathered the first ideas for the app. These ideas included a restaurant finder app, a calorie calculator or a diet creator. At first, the restaurant finder app was my favourite because this idea was easy to connect with the POSEIDON route app and the wallet app. But after I read some user stories from the Personas and scenarios.pdf, published at the POSEIDON project webpage, I got a better understanding of the PUs situations and their daily routine. After talking to some SUs, I found out that the PUs don't go out often in restaurants without family members or carers. This wrong supposition was underlined by the personal interview with a PU and her family, which struggled with weight and eating problems.

To prepare for the interview, I read the document Interviews-with-people-with-Down-syndrome.pdf from the POSEIDON project website. I found it to be very helpful. In that interview I ruled out the not so suitable ideas for app development. In the end, I had a clear view on what was needed for the Healthy Eating app.

In this development step, there were no difficulties. Formulating the questions to ask for the requirement gathering and waiting for the different answers to come in takes a lot of time. For me to get to know the target user group better, I found the documents provided on the website to be very helpful. What was extremely helpful, was to have someone which can establish the contact to a PU struggling with the topic of the app. Additionally, because the PU already knew the persons which helped from the DSA, it was easy to establish contact and helped the PU to express herself freely.

Activity	Used resources	Results	Difficulties	Required time
Email communication with the POSEIDON development community	POSEIDON UI guidelines	First ideas for the app	None	1 week
Reading user stories	Personas and scenarios.pdf [POSEIDON webpage]	Better understanding of the primary users' situations and daily routine	None	2 days
Personal interview with PU	Interviews-with-people-with-Down-syndrome.pdf [POSEIDON webpage]	Clarification of suitable ideas for app development	None	1 day

Total time: 10 days

2.2 Idea selection

The second step was the idea review. At first I wrote all ideas of the functionality of the Healthy Eating app which I gathered as feedback into an app description. I collected the information from the email communication and the personal interview with the PU. I have combined everything into a first draft. Afterwards I have sent the app description document to the POSEIDON development community in order to gather initial feedback and new suggestions. After the community answered I included the feedback and the suggestions into the app description document and created the final Description Healthy Eating App.doc.

At this step, I had no difficulties where additional documentation would have been more helpful. The difficulty was to integrate the different ideas gathered and to concentrate on one problem which I want to solve with this app. The PU interview was a totally clarifying experience.

Activity	Used resources	Results	Difficulties	Required time
Writing first app description	Input from DSAs and PU interview.	First draft of app functionality overview.	None	2 days
Email communication with POSEIDON development community about the app description	Input from DSAs.	App description document created. First feedback and new suggestions included.	None	1 day
Include feedback into the app description		App description and feedback from the communication. Improved app description.	None	1 day

Total time: 4 days

In the following I attach the created app description:

2.2.1 App description

Healthy Eating App

The Healthy Eating App helps persons with Down syndrome to become an overview of what and how much they eat. This App calculates the calories which the person with Down syndrome consumes during the day and shows him/her over a progress-bar.

Index

User types

User type	Description
Primary user	Person with Down Syndrome
Secondary user	Family member or adviser of the primary user

App Description

General App System

Before the primary user eats a meal s/he can check how much s/he should eat and drink at this meal.

For this the primary user touches the button of the meal type and after that s/he touches the information button in the top of the next window (See Figure 2). Now s/he sees suggestions of complete meals as several pictures of food. The health status of each proposal is indicated by a smiley. Very healthy meal (green and happy smiley), moderate healthy meal (yellow and normal smiley), not healthy meal (red and sad smiley). The primary user can choose between the meals over the arrow button left and right of the meal. If the primary user wants to go back to the food choices s/he touches the return button in the top of this window.

When the primary user eats something s/he can register it in the App. For this the primary user opens the Healthy Eating App. After s/he opens the App s/he sees buttons of the different types of meals breakfast, lunch and dinner and one button for snacks and one button for drinks. The primary user can select the meal-type now. When s/he selects it s/he sees pictures of choices of food which are categorized by the specific meal type.

For example: The primary user selects breakfast: s/he sees pictures of a slice of bread, slice of cheese, different slices of sausages, jam, butter and so on.

Now the primary user can touch the pictures of the food that s/he has eaten. The primary user must touch the pictures of food as often as s/he ate it.

For example: When the primary user has eaten two slices of bread one with butter and some slices of sausages and the other one only with jam. So, the user touches two times the picture with slices of bread, one time the picture with butter, one time the picture with slices of sausages and one time the picture with jam.

After that, the App calculates the calories and displays them in a daily progress-bar. This progress-bar compares the daily amount of calories and the consumed calories. The progress-bar has the colour green when the primary user has enough calories for the next meals on this day. Yellow when the primary user must eat less on the next meals because s/he has eaten too much on one meal. And red when the primary user has reached the daily calorie requirement, see Figure 1.

When the primary user has eaten too much at a meal the app shows a sad smiley and will alert the primary user that s/he should not eat more now, because s/he wants to eat later another meal.

When the primary user has eaten too much today the app shows a sad smiley and will alert the primary user that s/he should not eat more during this day or should make sport: like walking a few times around the block. This physical activity alert can relate to the POSEIDON Routes app. So, the Healthy Eating App shows a route that the primary user should walk. When the primary user walks this route, the app calculates the burned calories and reduces the progress-bar.

Once the weight-activity app from Middlesex University is finished:

The App relates to the weight-activity app from Middlesex University. So, if the primary user does any physical activity the healthy eating app can include the burned calories. Sport activities reported will reduce the calorie bar.

When the primary user registers eaten food, the app will store it in a history for 48 hours. So, it is possible that the primary user and the secondary user can hold a review about the primary users eating behaviour. The primary user has the option to turn of the function that the eating history is sent to the secondary user.

Support App System

The Healthy Eating App needs a system for the secondary user to configure the content of the PU App. Thus, the secondary user must register the meals with the food choices and drinks for the primary user.

The support app system can be a web-portal like the web-portal for the Poseidon shopping app or an external app.

In the support app the secondary user uploads pictures of food. After that s/he assigns the food to one of specific meal type breakfast, lunch, dinner, snacks and drinks, which are personalisable. As a last step, the secondary user specifies the calories of that food or drink.

The secondary user can create new meal categories or rename and change the pictures of these one which exist.

The secondary user can see the eating history of the primary user. The eating history will store for one week in the support app system.

Optional developments and extensions

- The Healthy Eating App can show how much the primary user has drunk during one day and if s/he has drunk enough.
- The Healthy Eating App shows better alternatives to a food choice.
- The secondary user can specify more information for a food choice like fat, sugar and so on.
- A calculator giving sugar cubes for showing the content of sugar in different food and drinks
- When the primary user eats 3 times too much on different days. The App asks him/her to do a Healthy Eating tutorial in the App. In this tutorial, the primary user learns what food is healthy and how much s/he should eat at the most on one day.

Pictures

This pictures are example views. They only show the functions and should give a hint how the system could look. The final version will be created in a later development phase.

Eating is good (green bar) User has eaten too much (red bar) Breakfast food choices

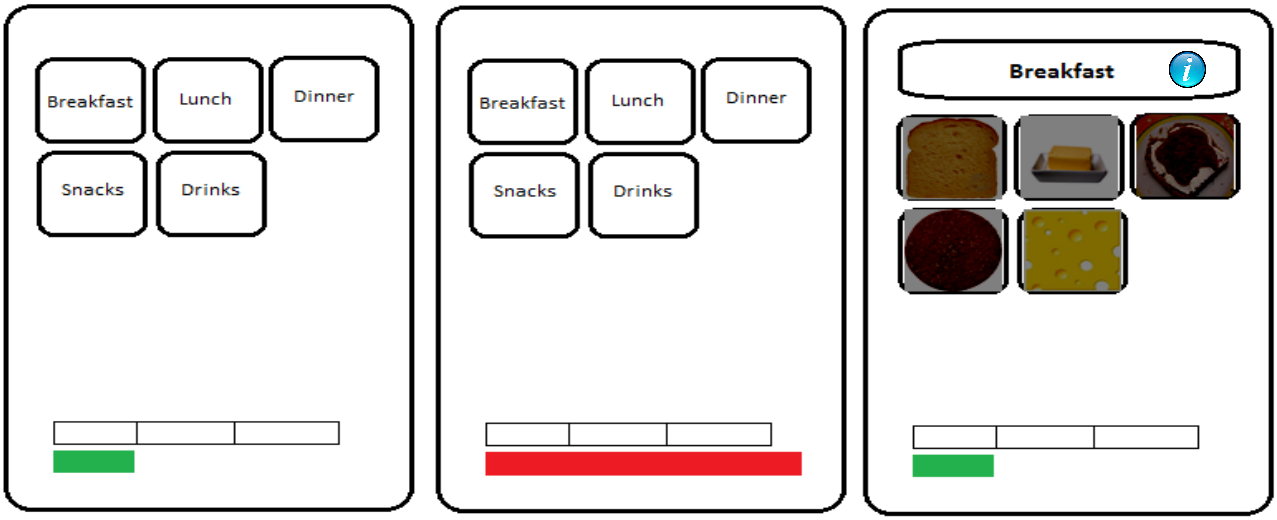


Figure 1

Breakfast meal suggestions

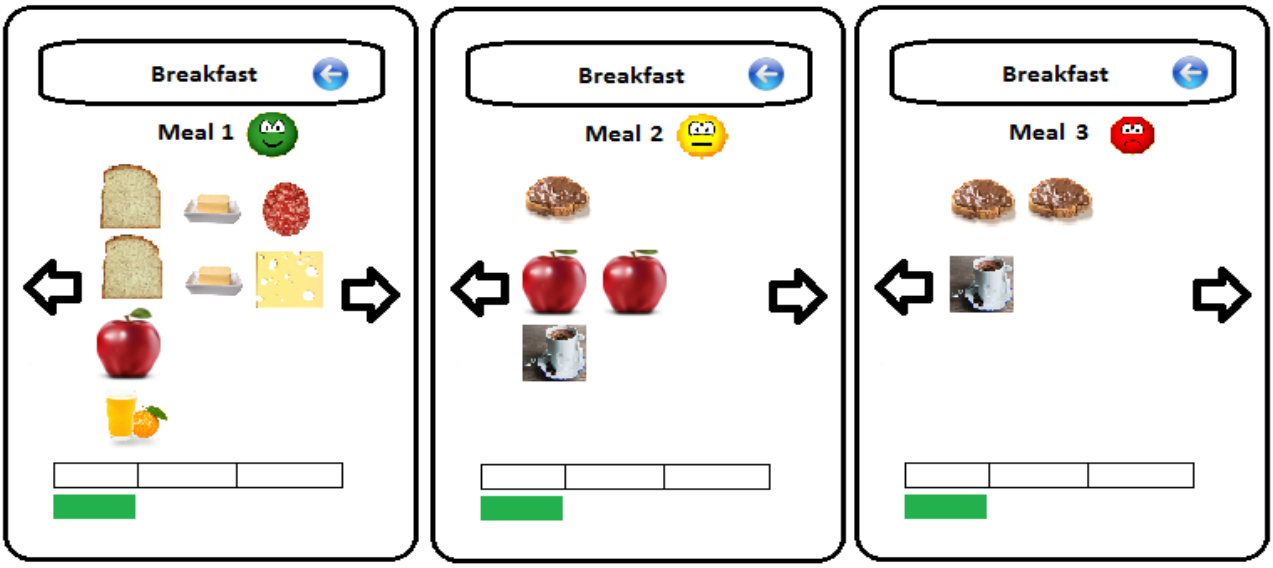


Figure 2

Functionality

Primary User

Description	Priority
PU can decide between different meals	Hi
Standard meals are Breakfast, Lunch, Dinner, Drinks	Hi
PU sees food choices when s/he selects a meal	Hi
PU can select food choices	Hi
PU see suggestions for a meal when s/he touches the information button	Me
Each meal suggestion has a smiley which signal how healthy this meal is	Me
PU see a progress-bar of absorbed calories and the daily maximum	Hi
PU gets a notification when s/he eats too much at a meal	Hi
PU gets a notification when s/he eats too much on a day	Hi
The notifications have a text message and a sad smiley	Hi
When PU walks a route or does physical activities, the burned calories will be subtracted from the absorbed calories	Hi

PU can select a route out of the POSEIDON Routes app to burn calories	Hi
PU can see the eating history for 48 hours	Hi
PU can decide if the SU can see the eating history	Hi

Secondary User

Description	Priority
SU can create food choices	Hi
SU can set calories to food choices	Hi
SU can categorize food choices to a meal	Hi
SU can set daily maximum calories	Hi
SU can create new meal categories	Me
SU can set name of a meal category	Me
SU can set the name of a meal category	Me
SU can set the picture of a meal category	Me
SU can set burned calories to a route out of the POSEIDON Routes app	Hi
SU can see the eating history of the PU for one week	Hi
SU can create meal suggestions	Me

General

Description	Priority
All notifications are simple to understand for the PU	Hi
All notifications for the eating behaviour have smileys	Hi
No numbers or text input from the PU	Hi
All buttons for food and meals have pictures	Hi
All selections from PU will be confirmed in a window	Hi
Connection to the POSEIDON Routes app	Hi
Connection between PU- and SU-system	Hi

Connection to the weight-activity app from Middlesex University	Me

2.3 Methodical approach of requirement analysis

After reading the documentation provided on the topic of identifying context awareness situations, I needed to learn to think in terms of contexts and be open for different kind of situations which could be identified as contexts. This method helps to imagine by knowing the PU target group daily routines what applicable situations of interested, where context awareness could be used, could be interesting to develop. The methodical approach is a step by step guideline on how to find situations and describe and break every possible context down until the level of input. Taking this broken-down information, I used it later on to generate code for my context implementation.

Activity	Used resources	Results	Difficulties	Required time
Learning Methodical approach requirement analyses	Tutorial questionnaire developers.pdf [Middlesex University]	First idea on how the methodical approach requirement analysis works	Understanding what is meant by the different	1 day
Methodical approach requirement analyses for the app idea	Tutorial questionnaire developers.pdf [Middlesex University], App Description.doc	First version of context awareness document	Differentiation between app functionalities and context situations, because app idea was already in place.	2 days
Review of the context awareness document	Context awareness document. [Middlesex University]	Feedback about context awareness document	None	1 day
Include feedback into the context awareness document and context situations into the app description document	Input from Middlesex, Context awareness Document, [Middlesex University] App description.	Final version of context awareness document and app description	None	1 day

Total time: 5 days

1 Methodical approach of requirement analysis

1.1 Establish Scope and High-Level Objectives

Scope	High Level Objectives
Mobile Application	Encourage people with Down syndrome to have a healthy life style.
Based on food choices.	Help people with Down syndrome to eat healthy without depending on others.
Tailored notifications to guide users depending how much they eat.	

Table 1: Scope and High-Level objectives

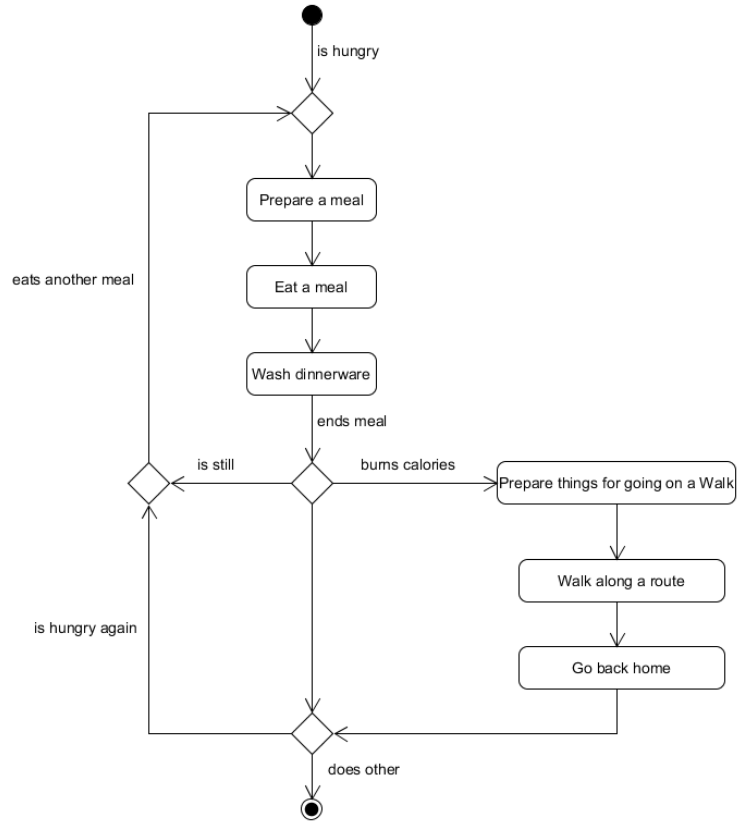
1.2 Identify Stakeholders & Profiles

Stakeholder	Description
Primary Users	People with Down syndrome.
Secondary Users	Parents or carers of people with Down syndrome.
Tertiary Users	Teachers or supervisors of people with Down syndrome.
Calls Provider	Company that provides phone calls and SMS to the mobile device.
Internet Provider	Company that provides internet to the mobile device.
Device Manufacturer	Company that manufactures the device.
Operating System Developers	Group involved in the development of the operating system of the device.

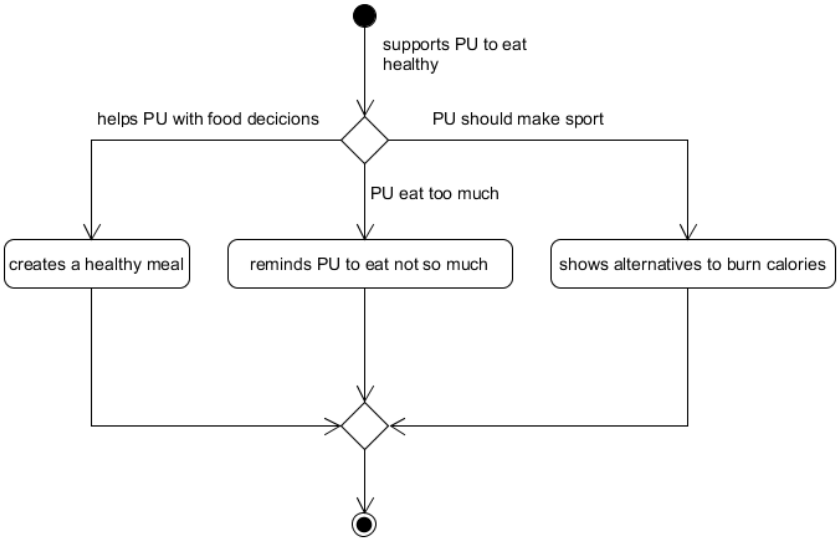
Table 2: Stakeholders & Profiles

1.3 Identify Activities

Primary User Activities



Secondary User Activities



Identifying System Performance Qualities

Stakeholder	Goals	Sub-Goals	Requirements
Primary User (PU)	Improve their Eating behaviour	Remind PU that s/he don't eat too much	Receive notifications that the PU can under-stand, taking into account possible visual and auditory impairments
		PU learn healthy food alternatives	The PU can decide between different foods
			PU see what food belong to which meal
			PU see food alternatives
	Improve their overview how much they eat	A basis for a review between PU and SU exists	The eaten food will be register in a history
		PU see how much they should eat and drink each meal	PU should check it before they start to eat a meal
			SU can register how much PU should eat and drink each meal
	Increase their sport activities	PU make more sport and burn more calories	The System reminds PU to make sport
		PU know what they have to do to burn calories	The System enables a connection to a route or weight management app
		Reduce their weight	The PU become an overview about the daily absorbed calories as a simply to understand information for them.

			PU can register physical activities done the current day

Stakeholder	Goals	Sub-Goals	Requirements
Secondary User (SU)	Reduce the support and attention that PU require on them when they eat something.	SU can create healthy meals for PU.	The system enables a creation of meals.
			The SU system and the PU system are connected.
	Reduce the support and attention that PU require on them when they make a sport activity	SU can create sport activity for the PU	The system enables a creation of sport activities.
			The SU system and the PU system are connected.
		PU can make sport without SU's help	The System guide PU by sport activities like SU.
			PU receive instruction which they understand

2. Identifying situations of interest, situational parameters and situational services

Identifying Situations of Interest & Identifying Situational Services

Primary User

Activity	Situation of interest	Situational Need	Situational Service	Interaction Type
Eat a meal	PU eats something	Know what and how much PU eats	Calculates the calories which PU absorbed per meal and per day	Passive

	PU eats too much today	Know how much calories PU absorbed today	Notify the PU when s/he eat too much	Active

Activity	**Situation of interest**	**Situational Need**	**Situational Service**	**Interaction Type**
Walk a route	PU walks a route to burn calories	Know the route which PU is walking	Guide PU along a route	Active
		Know how much calories PU burns on this route	Subtract the burning calories from the daily absorbed calories	Active

Secondary User

Activity	**Situation of interest**	**Situational Need**	**Situational Service**	**Interaction Type**
Create a healthy meal	SU wants to create a food choice for the PU.	Picture of the food choice	Uploads the picture	Passive
		Know how much calories have the food choice	Set calories to the food choice	Passive
		Know which meal type the food choice is	Categorize the food choice	Passive

Activity	**Situation of interest**	**Situational Need**	**Situational Service**	**Interaction Type**
Shows alternatives to burn calories	SU sets a route to burn calories	Know how much calories are burned on this route	Set calories to a route	Passive
		A Route	Set route to burn calories	Passive

Identifying Situational Parameters

Primary User

Activity	**Situation of interest**	**Identification Description**	**Situational Parameter**	**Source**

Eat a meal	PU eats something	PU selects a food choice	Food choice	Created by SU
			Amount of calories of the food choice	Set by SU
	PU eats too much today	PU's daily absorbed calories are more than his/her daily maximum calories	Today absorbed calories	Sum of calories of the today eaten food
			Daily maximum calories	Set daily maximum calories by SU

Activity	**Situation of interest**	**Identification Description**	**Situational Parameter**	**Source**
Walk a route	PU walk a route to burn calories	PU get notification that s/he has eaten too much.	Today absorbed calories	Sum of calories of the today eaten food
			Daily maximum calories	Set daily maximum calories by SU
		PU starts walking a route	PU Location	PU Mobile Device's
			Route coordinates	Registered route on the POSEIDON Routes app
		PU burns calories	Burned calories	To the map registered calories of route
			Daily absorbed calories	Sum of calories of the today eaten food

Secondary User

Activity	Situation of interest	Identification Description	Situational Parameter	Source
Create a healthy meal	The SU wants to create a food choice for the PU	SU uploads a picture	Picture of food	Filesystem of SU / SU's Camera
			Meal type	Meal types of the System
			Amount of calories	SU input

Activity	Situation of interest	Identification Description	Situational Parameter	Source
Shows alternatives to burn calories	SU sets a route to burn calorie	SU creates an activity for burning calories	A route / sports activity	POSEIDON route app.
			Amount of burned calories after walking the route / sports activity	SU input

2.4 Functional View

Finally, in this requirement gathering step, I created the Functional View document, describing the future app not as user walkthrough, but from the point of view of functionality. For creating the Functional view document, I used the Functional view template.docx, created especially for me, this was helpful. After I had the first version I sent it around for comments for the POSEIDON team to get the chance to check with the app description. With this feedback, I created the final version of the functional view document. With this I finished the requirement analysis of the development process. I had gathered enough information for the design step.

There are different functionalities which I categorized according to how needed and easily implementable this functionality is. Thus, I will implement the functionality categorized as HIGH until the Third POSEIDON User Workshop. This document covers the development status until then. Further on, I will continue to implement most of the medium and some of the low priority functionality.

Activity	Used resources	Results	Difficulties	Required time

Creation of the functional view document for POSEIDON mobile apps	Functional view template.docx [created by POSEIDON team]	Functional view document	None	1 day
Review functional view document		Feedback of functional view document	None	1 day
Include feedback into functional view document		Final version of functional view document	None	1 day

Total time: 3 days

Functional view of Heathy Eating App

	Description	Priority	Status
Healthy Eating App access	The Healthy Eating app can only be accessed from the main POSEIDON App.	MED	
Healthy Eating App access	The Healthy Eating app can be accessed by logging in with the Poseidon account.	HIGH	DONE
Download meal categories	The meal categories are downloaded from the Poseidon account.	HIGH	DONE
Download food	The food with the amount of calories and pictures are downloaded from the Poseidon account.	HIGH	DONE
Download meal suggestions	The meal suggestions are downloaded from the Poseidon account.	LOW	
Show meal categories	The meal categories are displayed.	HIGH	DONE
Show food choices	The food choices are displayed.	HIGH	DONE
Shows meal suggestions	The meal suggestions are displayed.	LOW	
View meal categories	Images of all meal categories present in the Healthy Eating app are shown.	HIGH	DONE
View food	Images of all food choices present in the Healthy Eating app are shown.	HIGH	DONE
View meal suggestions	Images of all food choices of a meal suggestion present in the Healthy Eating app are shown.	LOW	
Select food	Food from the Healthy Eating app can be selected as eaten.	HIGH	DONE

Total sum of daily absorbed calories	The sum of calories is calculated as daily absorbed calories.	HIGH	DONE
Show progress-bar of absorbed calories	The amount of absorbed calories is shown as a progress-bar.	HIGH	DONE
Meal suggestions	Show food choices collected as a meal.	LOW	
Healthy status of meal suggestion	A smiley signals the healthy status of a meal suggestion.	LOW	
Show eating history	Show the eating history of eaten food.	HIGH	DONE
Send eating history	The eating history can be sent to the SU-system.	MED	
Connection to POSEIDON Routes app	Have a connection to the POSEIDON Routes app.	MED	
Connection to weight-activity app	Have a connection to the weight-activity app from Middlesex University.	MED	
Notification «eat too much» at meal	Show notification when user ate too much at a meal.	HIGH	DONE
Notification «eat too much» at day	Show notification when user ate too much during a day.	HIGH	DONE
Subtract burned calories from absorbed	Subtract the burned calories by a physical activity from the absorbed calories.	MED	
Privacy	PU can decide if eating history is sent to the SU-system.	MED	
Simley signalisation	All notifications for the eating behaviour have smileys.	MED	
No input by typing	No numbers or text input from the PU.	HIGH	DONE
Few text	All buttons for food and meals have pictures.	HIGH	DONE
Input control	All selections from PU will be confirmed in a window.	MED	

Functional view of Food Creator App

	Description	Priority	Status
Create Food	Food can be created with picture and amount of calories.	HIGH	DONE
Set calories to food choice	The amount of calories of a food choice can be set.	HIGH	DONE
Categorize food	Food can be added to multiple meal categories.	HIGH	DONE
Set daily maximum calories	The daily maximum amount of calories can be set.	MED	
Set burned calories to activity	The amount of burned calories can be set to a route / physical activity.	LOW	
Store eating history	Eating history received from Healthy Eating app is stored for one week.	LOW	
Create meal categories	New meal categories can be created.	HIGH	DONE
Set name of meal category	Meal categories can be renamed.	HIGH	DONE
Set picture to meal category	The picture of a meal category can be changed.	HIGH	DONE
Set picture to food	The picture of a food can be changed.	HIGH	DONE
Create meal suggestion	Meal suggestions can be created.	LOW	

3 App design

3.1 App behaviour and look

In the design part at first I collected ideas about the app behaviour and look to get ideas how the navigation in the app should work and how the different windows in the app should look. For this I used the app description and functional view document which I previously created during the Requirement analysis part. Taking the POSEIDON GUI design guidelines into account I have developed drafts for the app windows and the navigation behaviour. Out of this I created a first app prototype with the different windows the navigation behaviour operating with dummy data. The most difficult thing at this point was to understand how app development worked. I had to understand how app windows (activities) are created and how to call different app screens. Having the ShoppingApp code available, was a great help because I directly learned on an existing example and could reuse many things. Through that I had a first success and was motivated to continue.

Activity	Used resources	Results	Difficulties	Required time
Collect Ideas	App description document, functional view document	Ideas how the app navigation should work, app drafts	None	2 days
Creating design prototype	POSEIDON interface design document for tests and pilots.pdf [POSEIDON website] Shopping App code [from Fraunhofer, POSEIDON website]	First app prototype	Understanding basics of how app development works. Using dummy data, was cumbersome since I had not yet configured the server side.	1 week

Total time: 9 days

3.2 App communication with file server

After I finished the first prototype the next step was to design the communication with the file server to fill the app with real data. For this I had to learn how the POSEIDON file server works. To learn this the File-server-API.pdf document was very helpful. From this document, I learned how the server works has and how the API works. After this, I created drafts for the communication between the Healthy Eating App and the FoodCreator App with the POSEIDON file server. However, the difficulty in this process was to understand how to implement the really well explained documentation into real java code. At this point, there was no exemplary app developed. Later on, this difficulty will be overcome by future developers by having a basic app which exemplifies up and download from the file server, our Starter App.

Activity	Used resources	Results	Difficulties	Required time
Learning POSEIDON file	File-server-API.pdf [POSEIDON website]	Knowledge on how the	None	1 day

server communication		POSEIDON file server works		
Creating drafts and data up- and download for communication between 2 apps over the POSEIDON file-server	File-server-API.pdf, App description document	Ideas for the data model of the app	Implementing the file server API in Java code.	1 day

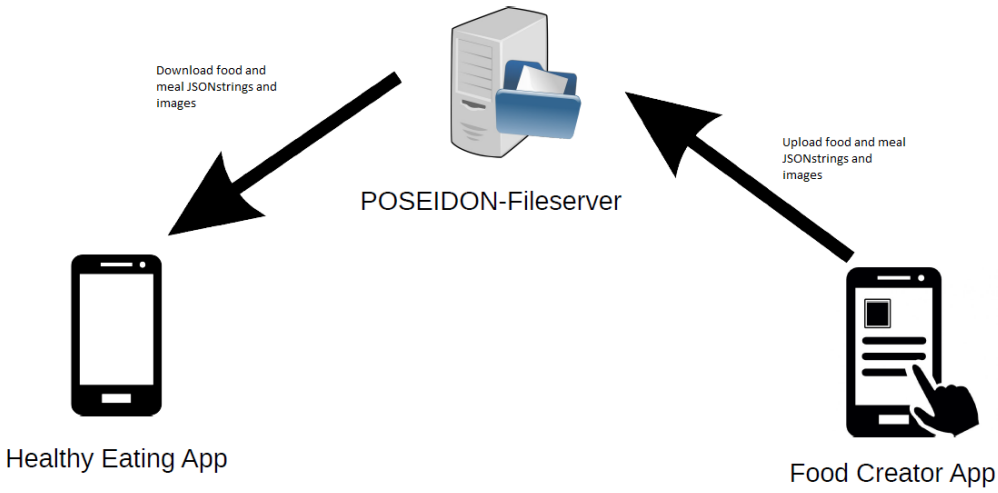
Total time: 2 days

3.3 App data model

With the ideas on how to create the app data model I created drafts of the data types and their attributes. After this, I had enough information to create data classes. So I implemented this classes into the app prototype. After this step, the app prototype reached a useful level comprised of a skeleton for implementation purposes.

```
meal:
{
        "Name":"Dinner",
        "Calories":"500",
        "ResourceID":"8cb49cfd-
e129-497a-96a7-012a5be6f8ec"
}
```

```
food:
{
   "Name":"Schnitzel",
   "Meals":"Lunch,Dinner,",
   "Calories":"500",
   "ResourceID":"8cb49cfd-e129-497a-96a7-012a5be6f8ec"
}
```



Activity	Used resources	Results	Difficulties	Required time
Drafting data model of the app	Ideas for the data model of the app	Draft of data model of the app	None	1 day

Total time: 1 day

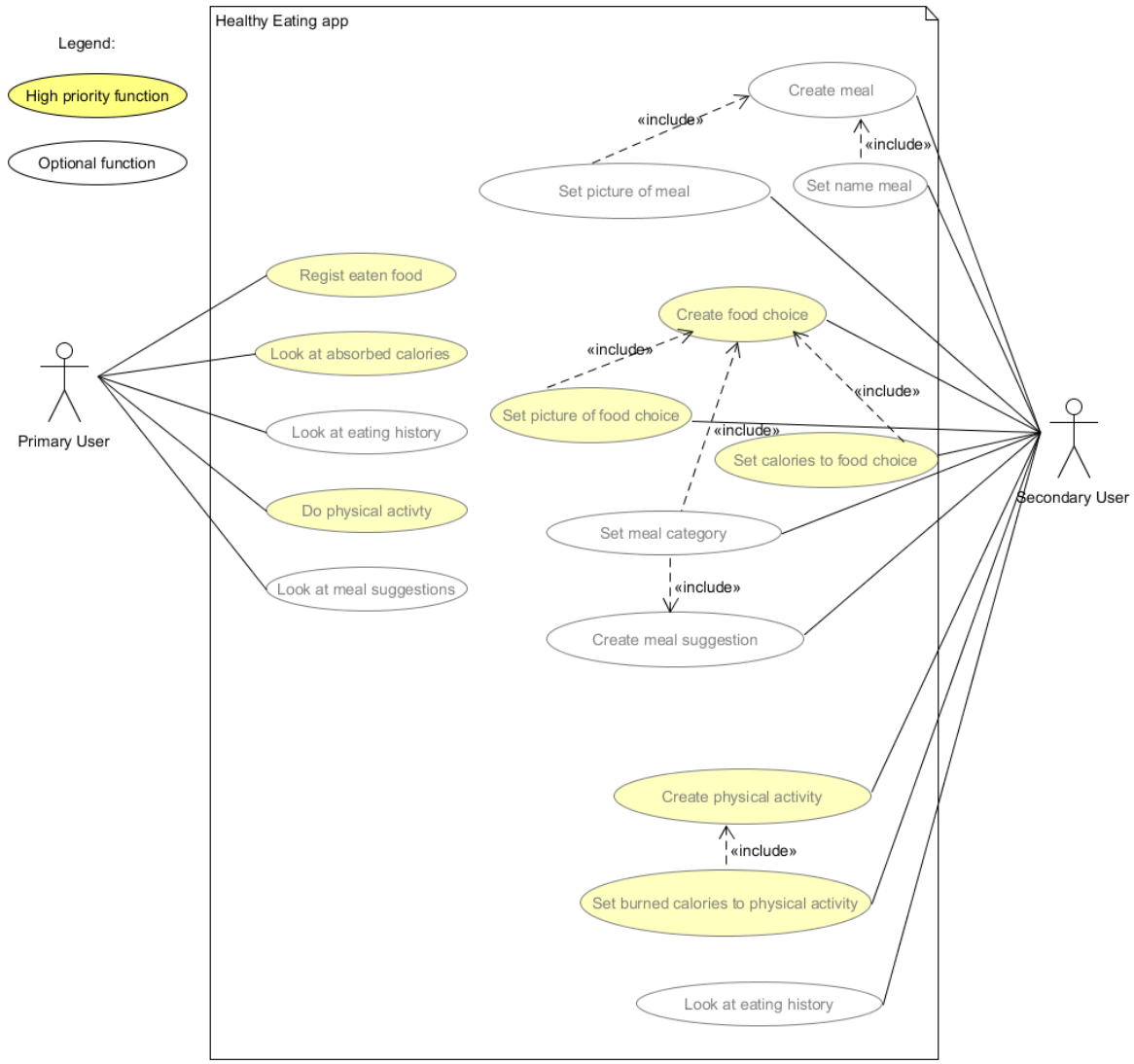
3.4 Analysis of detectable contexts - Context Awareness

To show the different app functionalities connected to the situations of interest I have created different diagrams. The use case diagram is included here.

The contexts I have identified using this method are two. The first alarms the user when he has passed the foreseen number of calories per day and meal. The second is more complex, since the system learns when the PU usually eats. If the person forgets to input data by that time, a notification of registering food or a reminder to eat is triggered whenever the person is indoors, at home or at school/work/shop.

Activity	Used resources	Results	Difficulties	Required time
Install Modelio and Eclipse Mars Studio with packages from Middlesex University	Source code from Middlesex University.	Necessary programs installed.	….	2 days
Create Use Case Diagram, Requirement Diagram, Context Dependencies Diagram	Context awareness document, App description	Use Case Diagram, Requirement Diagram, Context Dependencies Diagram	...	3 days
Review Diagrams	Context awareness Document, App description, Use Case Diagram, Requirement Diagram, Context Dependencies Diagram	Feedback on what was wrong		2 days
Include Feedback	Context awareness Document, App description, Use Case Diagram, Requirement Diagram, Context Dependencies Diagram, Feedback	New version of all diagrams		3 days

Total time: 10 days



4 Implementation

4.1 Starter App: Login, Up- and download to POSEIDON file server

The Healthy Eating app and all other apps which communicate with the POSEIDON file server need the function to login onto the file server, upload, download, change and delete files. I searched example code for this because that eases the understanding of the function and the implementation easier and faster. But I couldn't find something like this. Thus, I implemented an example app with the functions to login onto the file server, up- and download files, change files and delete files fro0m the file server. For all implementations, I used the File-server-API.pdf document [POSEIDON webpage] and code snippets from the POSEIDON MoneyHandling App.

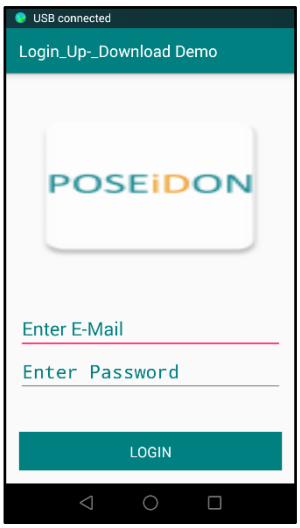
Because I had poor previous knowledge about html-connection communication, on which the file server API is based, I had a few problems with the functions upload, download and change file. For better understanding of this functions I contacted the file server administrator. He sent me example code for these functions and explained me the functionality so I could implement these functions. Because the example code was implemented with private libraries I had to change this code. As result of my work we have an example app with Java standard code for further POSEIDON app development.

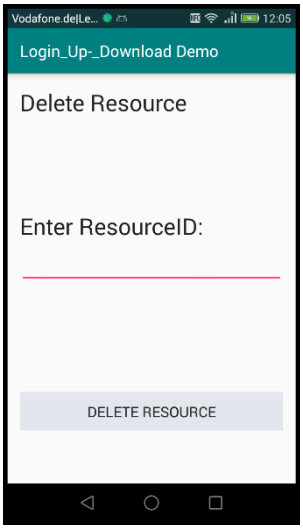
Other developers can use this demo code and include these functions into their own app, extending them with their required changes.

Activity	Used resources	Results	Difficulties	Required time
Implement login functionality	File-server-API.pdf [POSEIDON webpage], source code from POSEIDON MoneyHandling App	Example-code for login functionality	None	1 day
Implement upload and download files functionality	File-server-API.pdf [POSEIDON webpage], source code from POSEIDON MoneyHandling App, example code upload and download files	Example-code for up- and download files functionalities	Poor previous knowledge about html-connection communication.	2 days
Implement delete and change file functionalities	File-server-API.pdf [POSEIDON webpage], source code from POSEIDON MoneyHandling App, example code upload and download files	Example-code for delete and change file functionalities	None	1 day

Total time: 4 days

After logging into the app, the main functions of the file server are shown. These are showing the available resources, adding resources, deleting them and uploading them. The token for a specific POSEIDON account is also displayed. The token is the result of the authentication process and can be further used to implement functionality or to call other apps.





4.2 FoodCreator App

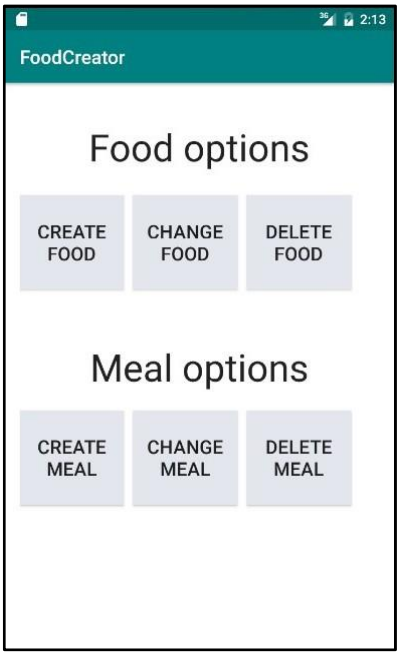
The implementation of the FoodCreator App was a very fast and easy process after I had the Starter App from the step before and the app data model out of the design part. To find all functionalities of the app I used the app description which I created in the Requirement analysis part. For the login screen, I used the code snippet for the login function from the Starter App because we need to login

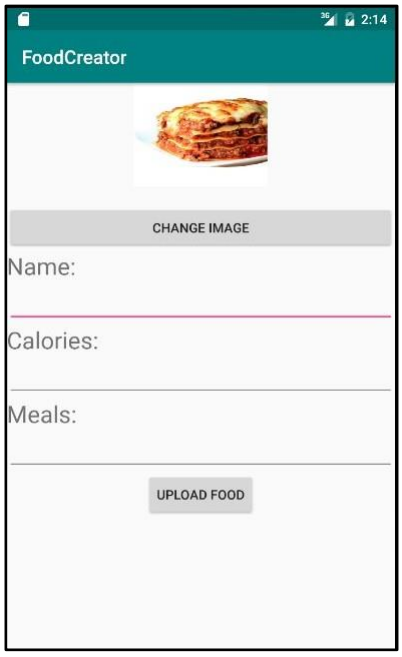
on the file server where we want to upload, download, change or delete files. For the functionalities Create Food and Create Meal, from the functional view, I used the app data model from the design prototype and the code snippet for upload files. For the functionalities delete and change food and delete and change meal I had to download the data first. For this I used the code snippet for download.

Activity	Used resources	Results	Difficulties	Required time
Implement Create Food and Create Meal functionality	Draft of data model of the app, App description, Starter App	App has functionalities to create new food and meal data types	None	1 day
Implement Delete/Change Food and Delete/Change Meal functionality	Draft of data model of the app, App description, Starter App	App has functionality to delete or change already added food and meals.	None	1 day

Total time: 2 days

The FoodCreator App is used by the SU in order to create meals and foods. This app is used in order to create the content of the Healthy Eating App, where eaten foods have to be marked. The daily food intake is thus controlled. The two screenshots show the buttons to create food, edit and delete it, as well as meals. The second screenshot shows how a food can be created and uploaded.





4.3 Healthy Eating App

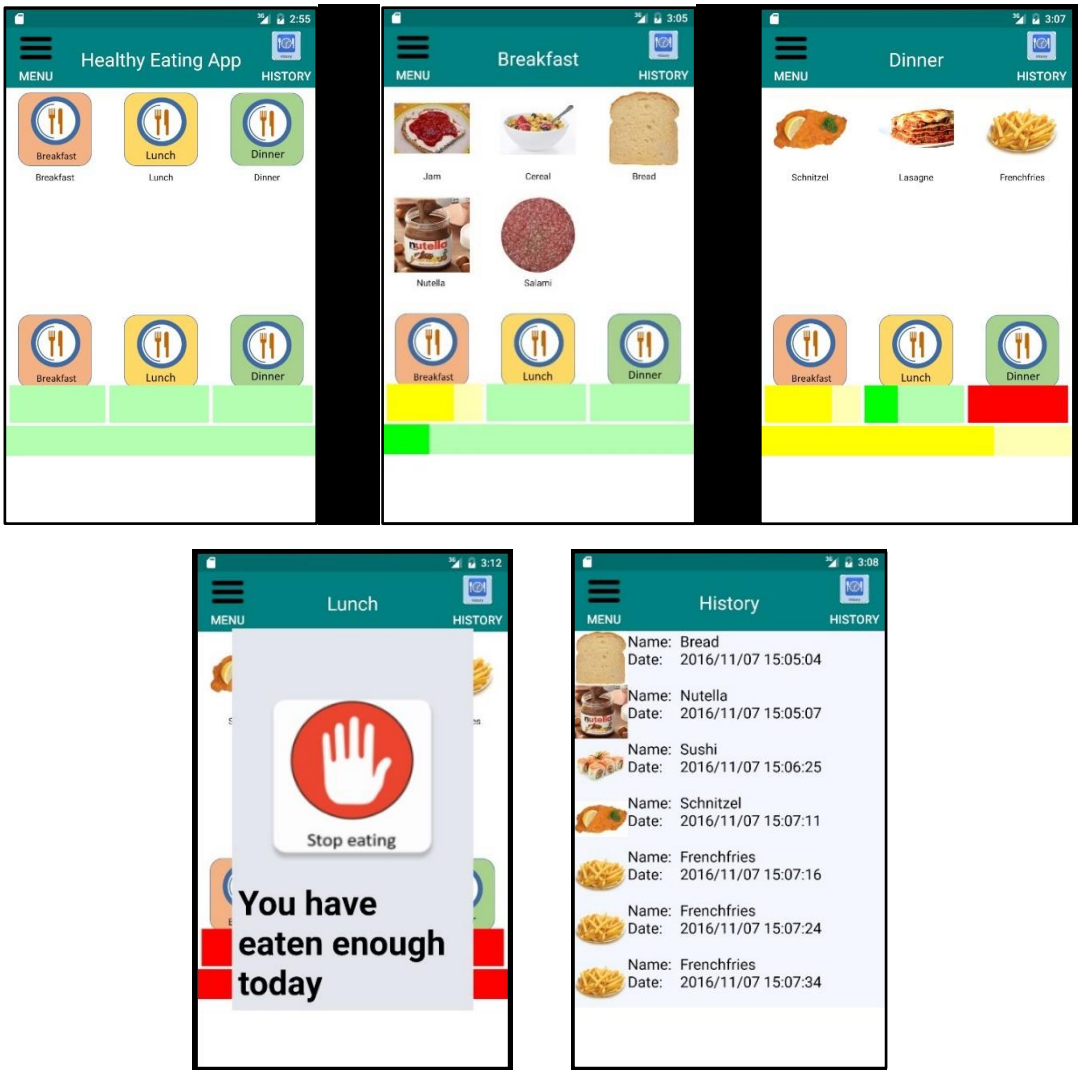
At this point I had learned how the communication with the file server works and how images can be uploaded. Step by step I implemented the functionality of the app, checking the app description, the functional view and creating or adjusting the data models. The Android developer community has been

a great support throughout the app development progress. One needs to understand how the components interact and how

Activity	Used resources	Results	Difficulties	Required time
Create app skeleton	App description, Android developer community	Healthy Eating App has all necessary windows	None	2 days
Implement functionality of main window	Draft of data model of the app, App description, Android developer community	Main window of the app has all necessary functionalities	None	1 day
Implement functionality of meal windows	Draft of data model of the app, App description, Android developer community	Meal window has all necessary functionalities	None	1 day
Implement progress-bar	Draft of data model of the app, App description, Android developer community	Main window and meal window have a working progress-bar	None	1 day
Implement history	Draft of data model of the app, App description, Android developer community	Healthy Eating App has a history function	None	1 day
Implement login screen	Example-code for login functionality, Android developer community	Healthy Eating App has a login screen	None	1 day
Implement Download data functionality	Example-code for Download file functionality, Draft of data model of the app, App description, Android developer community	Food and meals data is downloaded from the file-server	None	1 day

Total time: 8 days

In the following we present a few screenshots of the state of the app where all as HIGH prioritized functionalities are implemented. First, we show the main screen of the app where the Meals are shown in the upper row and below the progress-bar and the according meals are depicted. Selecting one of the meals from the tops opens the food selection attributed to that meal. Clicking on the food, the calorie bar is updated. If for one specific meal the calorie limit is passed the PU gets a message to stop eating. In the history view the PU can check what he has eaten.



5 Next steps

5.1 Next implementation steps

After the first prototype has been shown at the 3rd POSEIDON workshop, feedback has been gathered. This feedback mostly addressed the GUI. During this more intense use, some bugs have been discovered and will be addressed shortly.

Further on, as described in the functional view, there are many possibilities to extend the Healthy Eating App further than the existing functionality. One aspect is to integrate the app into the POSEIDON main app. An aspect regarding functionality extension is for the SU to be able to add meal suggestions which the PU sees using the app. The healthiness status of the meal is indicated by a smiley. Currently the history of the consumed food is shown only on the PUs app, not also on the SUs app. Finally, the app needs to be connected to another app, which monitors activity and weight management, using this as input which contributes to the calorie status of the progress bar.

5.2 App evaluation

The Healthy Eating App and the FoodCreator App need to be evaluated both. First, tests with general users, not restricted to persons with Down's syndrome, will be done in order to assure the basic usability and functionality of the app.

In a second step a PU and SU pair, struggling with healthy eating problems, are going to try out the apps and use them for a week. Intense communication with the PU and SU pair will assure gathering of all feedback. Only after implementing this feedback, the App could be distributed and input gathered from PU and SU pairs in a longer pilot study.

6 Quick developer guide

6.1 Requirement gathering

Resource	Name of document	Where to find it
User stories with people with Down's syndrome.	Personas and scenarios.pdf	http://www.poseidon-project.org/research-2/personas-and-scenarios/
POSEIDON user interface guidelines	D2.3 v2 Report on Design of HW Interfaces and Software Interim report [Chapters 2-5]	http://www.poseidon-project.org/research-2/deliverables/
Prepare for interview with primary user.	Interviews-with-people-with-Down-syndrome.pdf	http://www.poseidon-project.org/research-2/questionnaires/
Identifying contexts	Tutorial questionnaire developers.pdf Context awareness document	http://www.poseidon-project.org/developers/developer-documentation/
Prepare functionality list	FunctionalViewTemplate.docx	http://www.poseidon-project.org/developers/developer-documentation/

6.2 App design

Resource	Name of document	Where to find it
POSEIDON family GUI look and feel	POSEIDON interface design document for tests and pilots.pdf	http://www.poseidon-project.org/
Communication with file server	File-server-API.pdf	http://www.poseidon-project.org/developers/developer-documentation/
Data model specifications	Shopping-list-data-specification.pdf Video-list-data-specification.pdf Route-data-specification.pdf Calendar-data-specification.pdf	http://www.poseidon-project.org/developers/developer-documentation/

6.3 Start implementing

Resource	Name of resource	Where to find it
Example app: POSEIDON file server communication app [code]	Starter App	http://www.poseidon-project.org/developers/code/

Libraries	rest-lib-java.zip edge-lib-java.zip smartPlatform-libs-android.zip	http://www.poseidon-project.org/developers/code/

7 Conclusion

This document provides a case study guiding and exploring the development process of a Healthy Eating App for the PUs and the FoodCreator App for SUs. With this work, we highlight the use of the POSEIDON framework and the available POSEIDON guidelines and documentation. Especially the contact to the PU was highlighted as a decisive point, which contributed to selecting the most needed functionality from the many proposed app ideas as well as showing the need for utmost possibilities for personalisation.

The developer journal helps to quantify the efforts needed in creating the app. The fact that 25% of the total time was used for the actual implementation and 75% of the time was needed in order to get the idea and design the app in such a way that development can get started shows that creating the concept of the app is the most time-consuming part. Using the POSEIDON framework, the app was easily integrated into the POSEIDON system. Describing the development steps shows how the occurring difficulties were addressed by creating aiding documents or software. In order to ease the access for future developers which would like to contribute to the POSEIDON app family, a small overview of the documentation is provided with links to POSEIDON resources.