POSEiDON

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

Case Study

Developing using the POSEIDON framework by example of a Healthy Eating App

Call:	FP7-ICT-2013-10
Objective:	ICT-2013.5.3 ICT for smart and personalised inclusion
Contractual delivery date:	not applicable
Actual delivery date:	31.01.2017
Version:	v2
Editor:	Silvia Rus (FhG)
	Andreas Braun (FhG)
Contributors:	Patrick Schmitt (FhG)
Reviewers:	Terje Grimstad (Karde)
Dissemination level:	Public
Number of pages:	36

Contents

1 Introduction

This document provides a case study guiding and exploring the development process of an application using the POSEIDON development framework. The goal is to enable the reader to independently create applications that provide new functionalities to existing users of POSEIDON. We assume that the reader already knows some basic information about the project, its background, and how apps are developed. If not, the following links are a good starting point:

- POSEIDON Developer guide - http://www.poseidon-project.org/introduction/
- POSEIDON additional development documents - http://www.poseidon-project.org/developer-documentation/
- General information about the project - http://www.poseidon-project.org/

The chosen use case is a support app for carers and persons with Down Syndrome (DS) that helps in managing food intake. It lets carers set up a meal plan, whereas persons with DS get a simplified app that provides an abstract view on calories eaten and alarms when they should stop eating. This application was chosen as an important tool during workshops and other interactions with carers for persons with DS, as they often struggle with appropriate food intake.

This project was realized by a student of the Technische Universität Darmstadt in the scope of a six-month internship. In the end two different applications were created. A Healthy Eating App for persons with DS and the FoodCreator App for carers.

This case study is presented in a suspense form[1], whereas we begin with a short explanation of a mobile example app, before we follow the steps in the developer guide and present our actual application. We include references to additional documents used a developer reference, to read up on more information on specific parts. After this we give a short overview of how the POSEIDON development framework was used in the process before we outline the requirements gathering. Afterwards, the design of the apps is described, before diving into implementation details. The lessons learned chapter finally gives an overview of pitfalls that may occur during the development of a POSEIDON app. Finally, there are four appendices that show the time used for the development, an app description as result of the requirement gathering process, a detailed result of the requirements analysis performed, and a full functional view created.

[1] http://www.uefap.com/writing/genre/casestud.htm (accessed 16.01.2017)

2 Using the starter app

A common function for apps using the POSEIDON development framework and thus applicable to the Healthy Eating app, is the communication with the POSEIDON file server. This storage and repository is protected by a login system, supports upload, download, changing, and deleting files. It is therefore a quick and easy way to create an application that uses an account for the POSEIDON web, the design language of POSEIDON, and is available as source code at the following location (http://www.poseidon-project.org/getting-started-mobile-application-tutorial/). Other developers can use this demo code and include these functions into their own app, extending them with their required changes

2.1 Android application

Every Android application is able to use POSEIDON resources if it exposes the right level of APIs. The POSEIDON file server is accessible over a regular HTTP connection, therefore the Android app will need permission to access the internet and communicate over a network that has the necessary ports exposed (which is the case in the majority of cases).

If required the developers should refer to the Android documentation to learn more about general Android development[2] and using HTTP communication[3].

2.2 POSEIDON design language

While using the POSEIDON design language is not necessary in this starter app, we include general colour schemes and logos to create the look & feel of other apps for carers. The full documentation for user interfaces is included in the developers guide (http://www.poseidon-project.org/user-interface-guidelines/) and should particularly be taken into account, when designing applications for persons with DS.

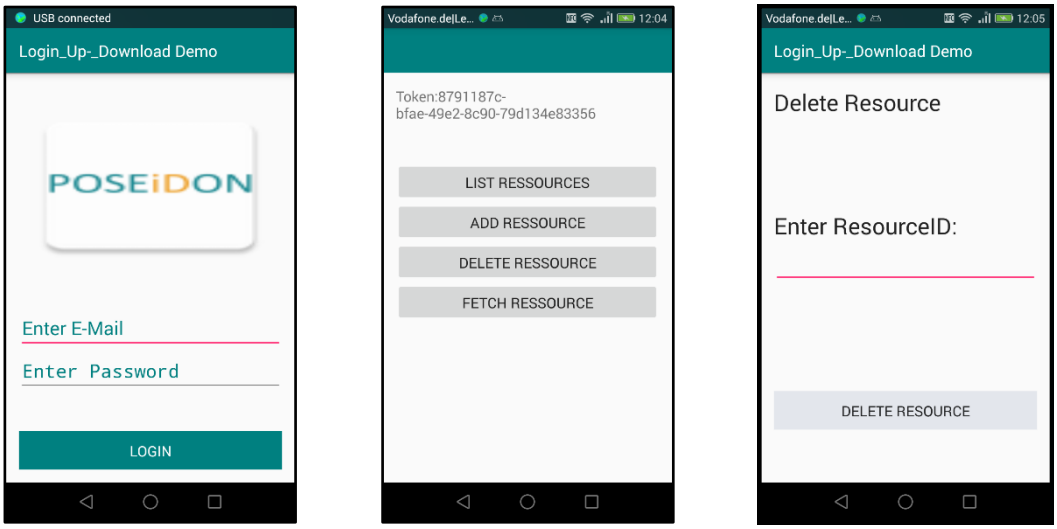


Figure 1 Screenshots of the POSEIDON starter app - left: login screen, center: main view with authentication token displayed, right: example of "Delete Resource" screen

[2] https://developer.android.com/about/start.html (accessed 16.01.2017)
[3] https://developer.android.com/training/volley/simple.html (accessed 16.01.2017)

For the development of apps the POSEIDON team is also providing some icons and other resources that are appropriate for developers. They can be found at the POSEIDON symbol repository (http://www.poseidon-project.org/product/symbols/).

2.3 Communication with POSEIDON file server

The POSEIDON file server is a centralized component that binds the different components together. It is used by the POSEIDON web and many other apps in the POSEIDON ecosystem, including navigation, home navigation training, and money-handling. The file server API supports logging in, as well as adding, changing, listing, fetching, and deleting of resources. In the following, we show how to add a resource. The full documentation with more examples can be found as part of the developer documentation (http://www.poseidon-project.org/getting-started-mobile-application-tutorial/).

To be able to use the file server the client has to authenticate with the authentication token identifying the user in SmartTracker. This token can be received either by logging in using the SmartTracker API directly or by logging in using the file server API. The following pages show the necessary HTTP request, the potential return objects that can be analysed by the app, and potential error codes that can be sent. Any appropriately designed app should catch and handle these errors. Once logged in, the server may use a cookie to maintain the X-Auth-Token. This is done because the web uses <img src="http://…"> tags that does not support a header.

Method	POST	<BASE_URL>/files/login.php
Headers	Content-Type	application/json

Payload	JSON-Object	{
		"username":"<USERNAME",
		"password":"<PASSWORD>"
		}

<USERNAME> and <PASSWORD> is you username and password of the respective SmartTracker account.

On success the request above will return a JSON-object containing the authentication token that is used for interactions with the file server.

```
{
    token: "36346d0f-6d08-423e-8059-491b1144ab6f"
}
```

The following errors may occur.

CODE	Error	Description
402	Payment Required	If authentication data is wrong
400	Bad Request	If body is missing or malformed

3 Using the POSEIDON development framework

3.1 User-centered-design as a development tool

User-centered design is a process of development with a strong emphasis on user involvement. The basic idea is to cooperatively generate ideas and refine these in iterative steps. This document will not give a detailed overview of this technique. Instead we would refer to some online resources, such as the Wikipedia page on this topic[4] that gives a good overview, or the information provided by the U.S. Department of Health & Human Services[5].

This report is the result of executing a single cycle of user-centered design, going from gathering ideas, creating requirements, development, and testing of the solution. Any market-ready product for the POSEIDON target group should complete several cycles.

3.2 Reference and support resources

The following tables are describing various resources that have been used in the process of creating this case study and are available to get more in-depth about developing for and with persons with DS. They are distinguished into resources during the requirements gathering, resources during the app design, and resources during the implementation of the app.

Table 1 Resources for requirement gathering

Resource	Where to find it
User stories with people with Down's syndrome.	http://www.poseidon-project.org/research-2/personas-and-scenarios/
POSEIDON user interface guidelines	http://www.poseidon-project.org/user-interface-guidelines/
Preparation for interview with primary user	http://www.poseidon-project.org/research-2/questionnaires/
Identifying contexts	http://www.poseidon-project.org/wp-content/uploads/4.Tutorialforcontext-awaresystems-2.pdf

Table 2 Resources for app design

Resource	Where to find it
POSEIDON GUI look and feel	http://www.poseidon-project.org/user-interface-guidelines/
Communication with file server	http://www.poseidon-project.org/getting-started-mobile-application-tutorial/
Data model specifications	http://www.poseidon-project.org/developer-documentation/

[4] https://en.wikipedia.org/wiki/User-centered_design (Accessed 20.01.2017)
[5] https://www.usability.gov/what-and-why/user-centered-design.html (Accessed 20.01.2017)

Table 3 Resources for code implementation

Resource	Where to find it
Example mobile app	http://www.poseidon-project.org/developers__trashed/code/
Additional libraries	http://www.poseidon-project.org/developers__trashed/code/

4 Requirements 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.

4.1 Idea gathering

At the beginning of the project ideas were gathered about the functionality of the healthy eating app. It was necessary to know in which life situations the app can help the primary user to be more independent. For this, we engaged in email communication with the POSEIDON Developer and Carer Community (http://www.poseidon-project.org/contact-2/). In this way, we 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 our favourite because this idea was easy to connect with the POSEIDON route app and the wallet app. But after reading some user stories (see document: Personas and scenarios.pdf[6]), published at the POSEIDON project webpage, we got a better understanding of the situations persons with DS encounter and their daily routine. After talking to some carers, we found out that the persons with DS don't usually go to restaurants without family members or carers. This wrong supposition was underlined by the personal interview with a person with DS and her family. She struggled with weight and eating problems.

To prepare for the interview, preparation material is provided on the POSEIDON project website[7]. 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. It is important to get a good understanding of the target group, particularly persons with DS that are directly affected by the challenge of the healthy eating app.

4.2 Idea selection

The second step was the idea review. All ideas for the functionality of the Healthy Eating app that are gathered as feedback, should be collected. In our case this takes the form of an app description (see Appendix B). It includes information from email communication with the POSEIDON network, personal interviews with carers, and a personal interview with a person with DS. This app description was

[6] http://www.poseidon-project.org/wp-content/uploads/2016/02/Personas-and-scenarios.pdf (accessed 16.01.2017)

[7] http://www.poseidon-project.org/wp-content/uploads/2016/02/Interviews-with-people-with-Down-syndrome.pdf (accessed 16.01.2017)

verified and commented by the POSEIDON development community. The final report can be found in Appendix B.

4.3 Methodical approach of requirement analysis

After reading the documentation provided on the topic of identifying context awareness situations, one needs 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. Breaking down this information enables one to generate code for the context implementation.

4.4 Functional View

Finally, in this requirement gathering step, a Functional View document should be created that describes the future app not as user walkthrough, but from the point of view of functionality. For creating the Functional view document a template is provided in the developer's resources[8]. In our case we gathered feedback about this document from the POSEIDON community, which was included in the final version. This version can be found in Appendix D. This concludes the final step of the requirements gathering process, leading into the design step.

Table 4 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

[8] http://www.poseidon-project.org/wp-content/uploads/Functional-view-template.docx

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	

5 Designing the application

5.1 App behaviour and look

In the design part at first ideas should be collected about the app look and behaviour. This should include how the navigation in the app should work and the look of the different windows in the app. The app description and functional view documents created can be used. For POSEIDON apps a GUI design guide is provided[9], which also shaped the design of this app. The developer should refer to general development guidelines for Android, or refer to the source code of the provided StarterApp[10]. Figure 2 shows the result of this initial development for the Healthy Eating App.

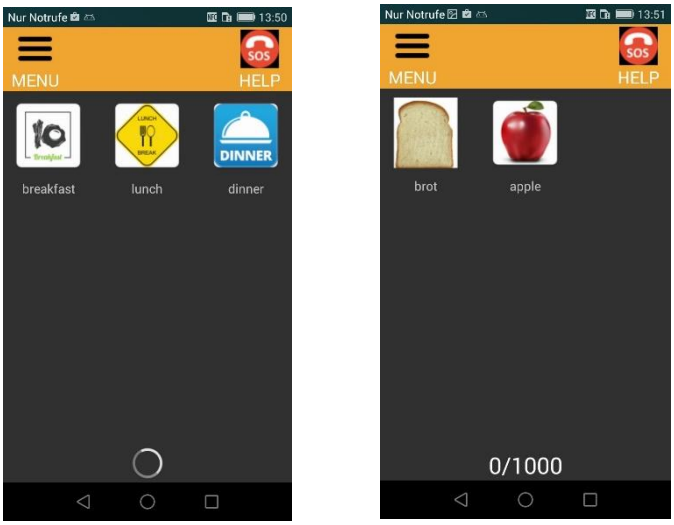


Figure 2 Early samples of live application

5.2 App communication with file server

If this basic design is finished the integration into the POSEIDON framework can begin. In this case the communication with the POSEIDON file server has to be established. For this step, both the StarterApp that establishes communication, as well as API documentation is provided[11].

5.3 App data model

The data stored has to be defined, by creating data models of the resources the app has to write and read from the POSEIDON system. These should follow a simple JSON format for data classes. In the case of the Healthy Eating App we need distinct data models for meals and single food items. Each has a name, calorie count and resourceID, whereas the food class is additionally linked to certain meals.

```
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"
}
```

[9] http://www.poseidon-project.org/introduction/
[10] http://www.poseidon-project.org/getting-started-mobile-application-tutorial/
[11] http://www.poseidon-project.org/wp-content/uploads/File-server-API.pdf

Examples are shown in the listing, with Figure 3 showcasing the data flow model of the Healthy Eating and Food Creator Apps.

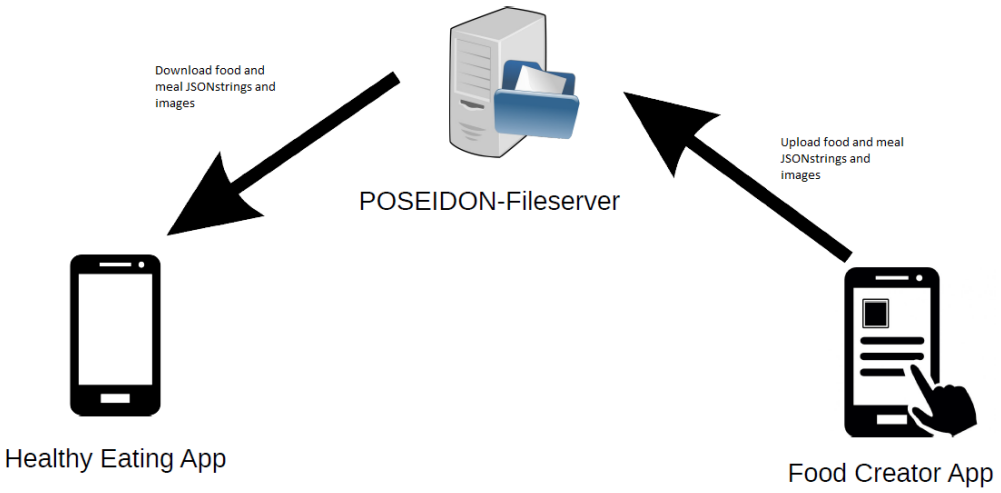


Figure 3 Data flow model of Healthy Eating and Food Creator Apps

5.4 Analysis of detectable contexts - Context Awareness

To show the different app functionalities connected to the situations of interest it is useful to create diagrams that clearly show the different use cases. This can help in establishing any links to the Context Awareness provided by POSEIDON. These contexts can be used by multiple apps in the POSEIDON ecosystem, yet can also be created by those.

An example for the first case is the learning of typical dinner times. Whenever the Healthy Eating app is used the times associated to meals are saved. Any app can use this information, e.g. to include events in calendars or optimize shopping and navigation experiences. This can e.g. be used by location services to trigger eating reminders, when at a certain location.

An example for the second case is a trigger that is enabled when the user exceeds a certain caloric input. In our case it is only used for displaying warning messages, but it could also lead to notifications to carers by other apps of POSEIDON.

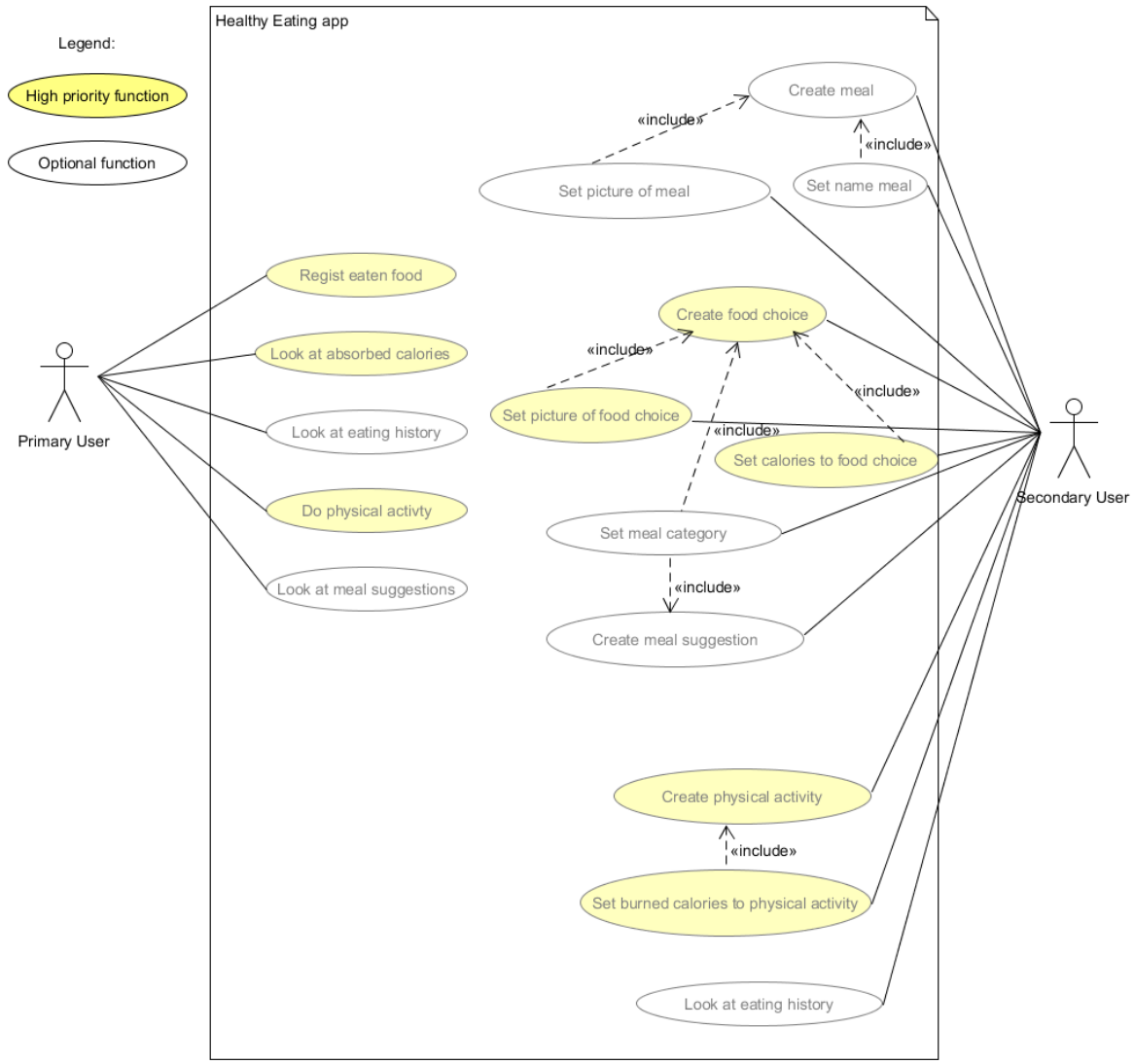


Figure 4Use Case Diagram of Healthy Eating and Food Creator App

6 Implementation

6.1 FoodCreator App

The implementation of the FoodCreator App was based on the Starter App provided, using the app data model out of the design part, as well as the app description from the requirement analysis. Code can be reused, e.g. for the login screen. For the functionalities Create Food and Create Meal that were established from the functional view, the app data model from the design prototype and the code snippet for uploading files were used. For the functionalities delete and change food and delete and change meal the data has to be downloaded first, extending the code snippets from the Starter App.

The FoodCreator App is used by carers, in order to create meals and foods. This app creates the content for the Healthy Eating App, where eaten foods have to be marked. The daily food intake is thus controlled. Figure 5 on the left shows the buttons to create food, edit and delete it, as well as meals. Figure 5 on the right shows how a food can be created and uploaded.

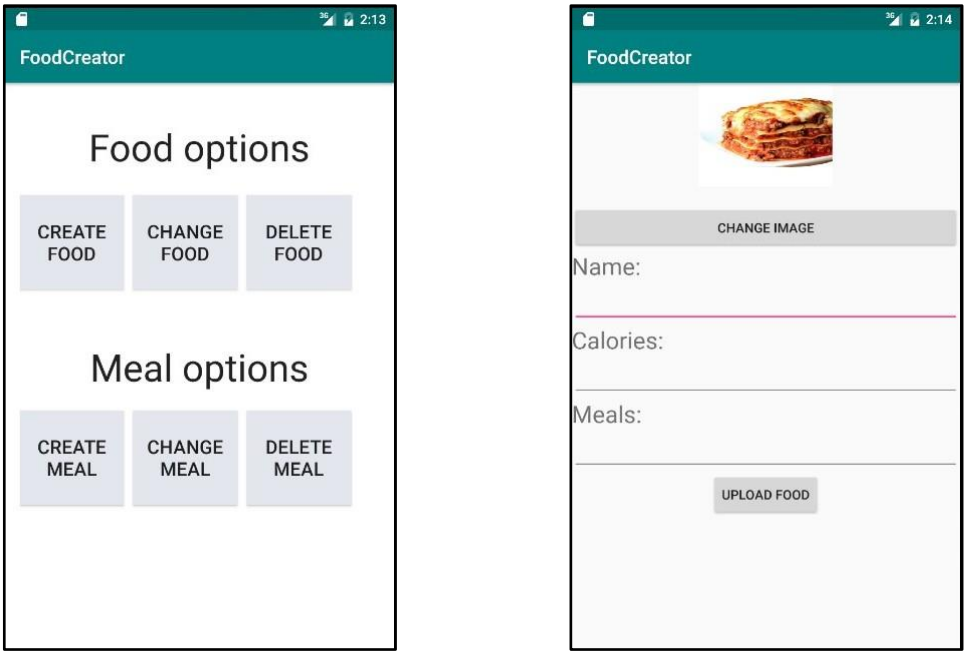
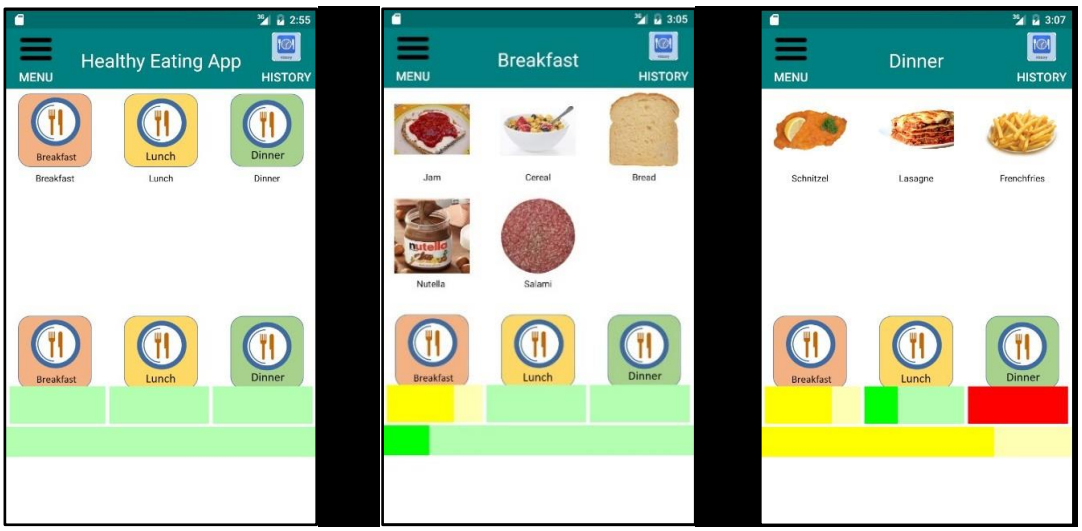


Figure 5 Screen shots of Food Creator App

6.2 Healthy Eating App

The Healthy Eating App was created in a similar fashion. Step by step the functionalities for the app as denoted in the functional view are implemented, cross-checking with the app description, and if necessary modifying the developed data models.

Figure 6 presents a few screenshots of the state of the app, with all high priority functions integrated. The top left shows the main screen of the app, where the Meals are shown, as well as all progress bars in a neutral state. Selecting one of the meals from the tops opens the food selection attributed to that meal, shown in the top middle. Clicking on the food, the calorie bar is updated, shown also in the top right. If for one specific meal the calorie limit is passed the person with DS gets a message to stop eating, as shown on the bottom left. In the history view the person with DS can together with carers check what he has eaten in the last days, as shown on the bottom right.



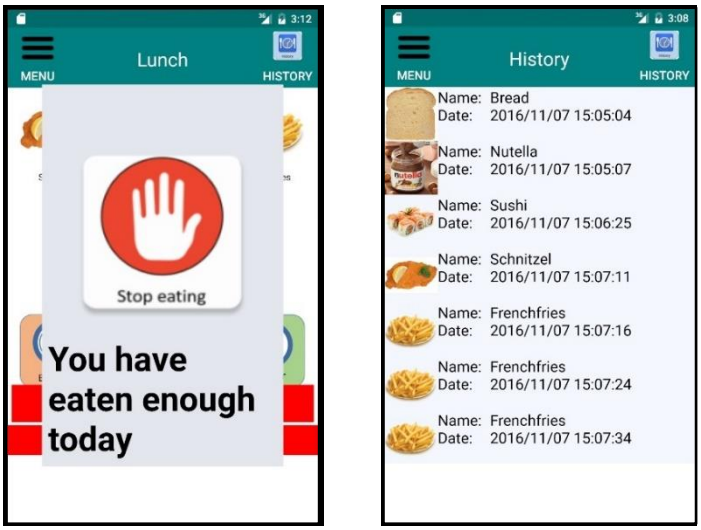


Figure 6 Screenshots of Healthy Eating App

7 Lessons learned

In this chapter we outline some of the lessons that have been learned by the developer in the course of this project.

7.1 Understanding the needs of persons with DS

It was important to get a good idea about the functionality of the app in the initial state. When developing for users with DS, it is essential to understand their point of view, as well as that of the carers. There should be interviews with persons with DS and interaction with the developer community.

7.2 Formalizing requirements gathering

Using the formal requirements gathering methods can improve the documentation of these features and lead to a more efficient development process. Thinking about situations instead of functions can greatly improve this process. There are various resources already available, but interaction with the POSEIDON community proved essential.

7.3 Understanding the basics of app development

At the beginning of the project only limited example code was available. Creating the Starter App was a measure taken to improve this situation for future developers of POSEIDON.

8 Additional development steps

8.1 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 person with DS 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 person with DS's app, not also on the carers 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.

8.2 App evaluation

The Healthy Eating App and the FoodCreator App were evaluated with the POSEIDON community during the creation process. However, this should be formalized and happen on a larger scale. In addition, tests can be performed with the general populace, in order to assure the basic usability and functionality of the app. After that a person with DS with healthy eating problems and their family could try the apps and use them for a certain amount of time. Feedback can be gathered to improve the app, before conducting a larger and longer pilot study.

Appendix A: Analysis of time required

Table 5 Overview of required time for designing app

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**
Implementation	Login, up- and download App	*4*
	Food Creator App	*2*
	Healthy Eating App	*8*
	Subtotal	**14**
	TOTAL	**59**

Table 6 Time required for starter app

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

Table 7 Idea gathering

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

Table 8 Idea selection

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

Table 9 Methodical requirement analysis

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

Table 10 Creating functional view

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

Table 11 Designing app behavior and look

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

Table 12 App communication with file server

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

Table 13 App Data Model

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

Table 14 Designing context awareness

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

Table 15 Implementing Food Creator App

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

Table 16 Implementing Healthy Eating App

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

Appendix B: 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

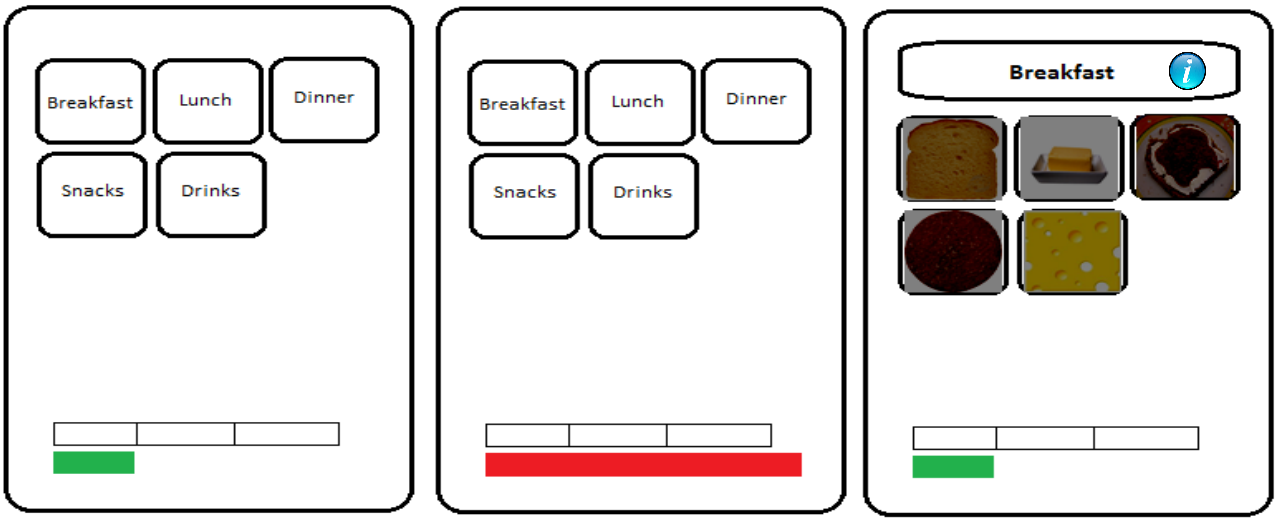


Figure 1

Breakfast meal suggestions

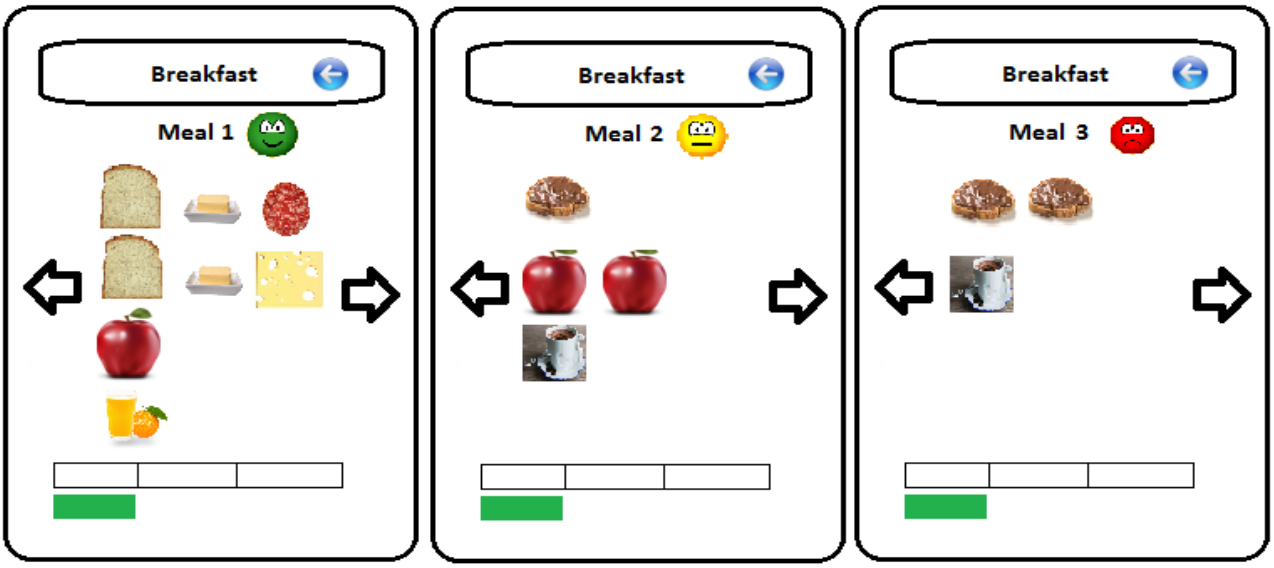


Figure 2

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

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.

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 4.4, 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 **Fehler! Verweisquelle konnte nicht gefunden werden.**-**Fehler! Verweisquelle konnte nicht gefunden werden.**, 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 **Fehler! Verweisquelle konnte nicht gefunden werden.** we present the next steps regarding implementation and user evaluation, while in the last chapter, Chapter **Fehler! Verweisquelle konnte nicht gefunden werden.**, 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.

Appendix C: Detailed results of requirement analysis

1 Methodical approach of requirement analysis

1.1 Establish Scope and High-Level Objectives

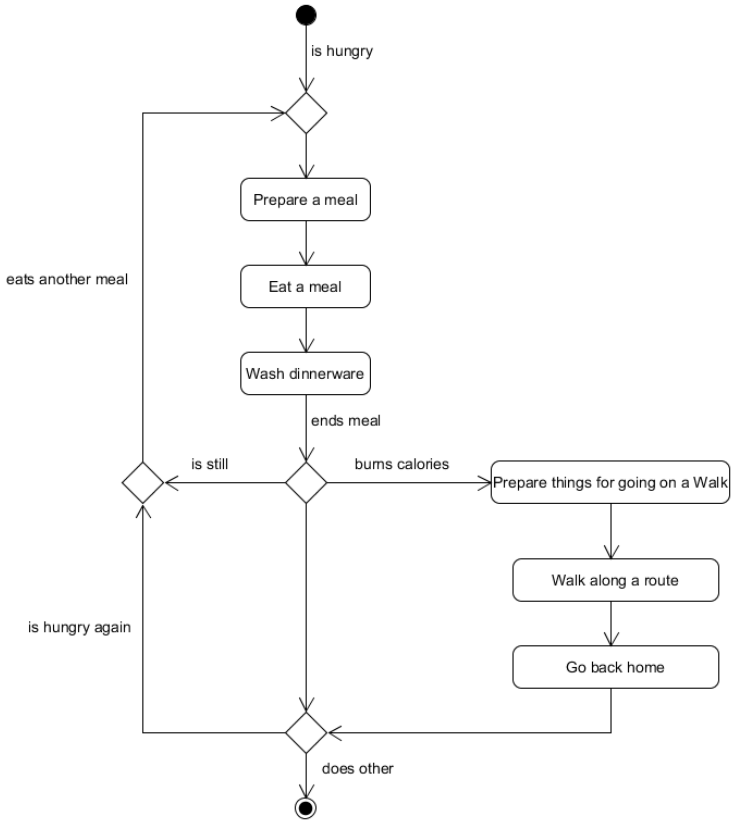
Scope	High Level Objectives
Mobile Application	Encourage people with Down syndrome to have a healthy life style.
Based on food choices.	Help people with Down syndrome to eat healthy without depending on others.
Tailored notifications to guide users depending how much they eat.	

1.2 Identify Stakeholders & Profiles

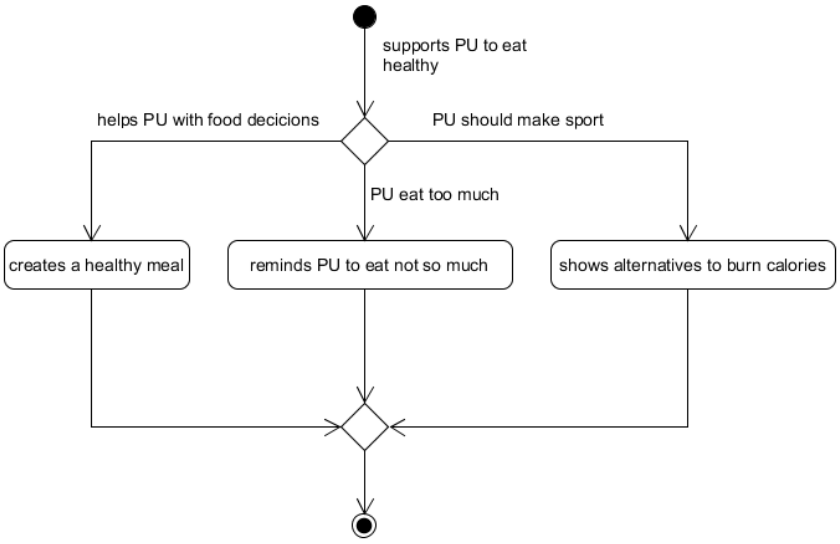
Stakeholder	Description
Primary Users	People with Down syndrome.
Secondary Users	Parents or carers of people with Down syndrome.
Tertiary Users	Teachers or supervisors of people with Down syndrome.
Calls Provider	Company that provides phone calls and SMS to the mobile device.
Internet Provider	Company 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.

1.3 Identify Activities

Primary User Activities



Secondary User Activities



1.4 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

2.1 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	Passive

			per meal and per day	
	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

2.2

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

Appendix D: Functional view of Healthy Eating App

The different functionalities are categorized according to how necessary they are according to potential users and the complexity of implementation. In the case of this app functionalities categorized as HIGH were to be implemented until the Third POSEIDON User Workshop.

Table 17 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	