Umeå University
Department of Computing Science

Development of Mobile Appliations 7.5 p
5DV155

'Mealpricer' - Free of Choice App Project

Submitted 2017-08-21
Author: Lorenz Gerber (dv15lgr@cs.umu.se lozger03@student.umu.se)
Instructor: Johan Eliasson / Jonathan Westin

Contents

1 Introduction

The aim was to come up with a good idea for a reasonable sized mobile application that should then be implemented. Reasonable size here was that implementation should be possible in about 60 hours of work (1.5 week).

During the course, there were some propositions of possible applications where one was a cooking recipt app with shopping list functionality. I like to cook and I am usually on a tight budget. Further, I actually like to guess or calculate the price of all kind of stuff I see around me. Hence, from the cooking receipt application I got inspired to develop 'Mealpricer', an app to calculate the fractional costs of meals.

Generally, wile pondering about potential projects, I was convinced that I did not want to implement a game as this would involve mostly general java programming. My aim was to find a project that would give me the possibility to improve on my general crafts for navigating the android framework. In my opinion, the most typical mobile application is data centric. It presents some sort of data in lists. Currently, to make a professional looking android app, I think it is indispensable to master the elements of material design. Therefore, another aim for me was to incorparate as much material design as possible in my project. So far, I have not spent much time with GUI programming in general, there I wanted to develop more in this field.

2 General Description and Target Group

The envisioned and implemented android app with the name 'Mealpricer' is targeted towards indviduals that are looking for a tool to quick and easily determine the fractional cost of a specific meal.

I checked out the Google Play Store for similar apps. There were a few with a smiliar idea. There I also realized the potential behind the concept: Calculating fractional costs of a meal from bulk ingredient prices and ingredient amounts is actually a very common application for restaurants to price their meals.

So besides individuals, I think this application could make a good tool for food and catering professionals to quickly recalculate meal prices while shopping for ingredients.

3 PlayStore Description

MealPricer is an application that helps you to keep track of how much individual meals that you prepare cost. Whether you are a student on a tight budget or a restaurant owner that wants to calculate the fractional costs of his offerings from bulk products, MealPricer will help you in the most simple way to keep track of it.

4 Target Course Grade

I aim for a 'VG' and below follows some reasoning around it.

I have spent quite a bit more time on the app than the indicated target of 60 hours. I am aware that for a senior android developer, such an app could be developed in a fraction of that time. However, I think it is also understood that this is an introduction course to the android framework, hence additional time is justified.

Generally, I am pretty happy with the result of my application. I am aware that the app

has a rather simple concept with an easy to implement data model. But this was also by purpose as mentioned earlier: My aim was to spend as much time as possible with actual android framwork programming and not general Java. I think that the final product looks appealing and can be used practically. I spent quite some time on drafting the initial flow of the app and looking up which concepts of material design that would make for a good user experience.

5 Aspects on Security and Ethics

6 User Guide - How To

7 Application Architecture

8 Selected Discussion Topics

8.1 Embedding Fragments statically/dynamically in Activity

Trying to use the provided templates for google material designed activities and fragments, I run into the situation where fragments where by default embedded into activities by instantiating them in XML. This works fine when the Activity/Fragment combo in question does not need to obtain extras on creation. In many cases however, for example when the Activity/Fragment combo represents the detail view after a list activity, the ID of the chosen list item has to be obtained. Hence, the given template code had to be modified for dynamic instantiation of the fragment from within the activity. During planning for the above modification, it was noted that there are at least three single activity / single fragment views in the current application. According to the example in the course book, this could make up for an 'Abstract' 'SingleFragmentActivity' class. However, it was decided against this due to implications on the material design elements: The Appbar and Floating Action Button are kept in the activity, while the content layout is in the fragment. But the three single fragment activities don't share exactly the same setup for appbar and Floating Action Buttons. Hence another layer of abstraction would be needed that would make things overly verbose.

8.2 Deciding on the Persistance Model

It was from the beginning decided to store the data in SQLite. It was needed to decide how to store data from the UI on for example orientation changes. Currently, in the IngredientChooser, when an ingredient is not selected, the value will not persist when the device changes orientation. To come to a somehow fluent user experience, it was decided to connect the checkbox for selected with the textchange listener of the entry boxes. Hence, when text is entered, the selected textbox is activated. In the same sense, to improve the user experience, unselecting the checkbox will automatically also empty the entry fields. As such, it is no longer possible to have entered text without selection active. Hence, on orientation change, all data will be stored in the DB and read back on resume.

8.3 Deciding on Ingredient Chooser UI

Showing zero's or not. Showing a search bar. Sorting alphabetically.

8.4 In InigredientChooser, correct initial focus

On ActivityStartup, always the first field get's focus before data is loaded. The onFocus method then determines that there is no data, hence it writes/overwrites directly a zero to the database. The solution was to add a dummy focus on the title element in the XML file.