Requirements Analysis Document

SKAJ Corp - KvittoSkanner

Version: 1.9

Date: 2017-05-26

Author: Sanjin Slavnic, Joakim Mattsson, Kevin Brunström & Anne Keller

This version overrides all previous versions

Table of contents

1. Introduction

This application is sought after in the business industry, primarily to make life easier handling expenses. As of today there is no obviously easy way to take care of all your receipts. In order to do your bookkeeping you need to archive the receipts and manually register all the data (expenses on the company) in the end of each month. This is time consuming and inefficient.

Our goal is to develop an Android application that allows users to take a photo of a receipt and and get all the data organised in the system directly. This application is wanted on the market by mostly traveling businessmen and women to take care of a receipt straight away.

The application will be used every time a purchase has been made that needs to be registered. The user should be able to effortlessly take a picture of their receipt and document it in the app, not being forced to bookkeep all expenses at once by the end of the month.

1.1. Definitions, acronyms and abbreviation

A.

API

- Application Programming Interfaces is a set of tools and protocols used for building software application, specifying how software components should interact.

Apache License

- The Apache License is a free software license that allows the user of the software the freedom to use the software for any purpose.

B.

BottomBar

- A custom view component that mimics Material Design[1] Bottom Navigation pattern.

E.

Enumerated type (Enum)

- A data type built up with a set of named values.

G.

Gradle

- Open source build automation system designed for multi-project builds.

GUI

- Graphical user interface.

[1] https://material.io/

L.

Library

- A library is a collection of resources, including pre-written code, classes, values, specifications and documentation used by programs to develop software.

M.

MVC

- Model-View-Controller is a way to organize the structure of an application. The Model containing the data and logic that determines how the application will operate. The View responsible for everything visible (the GUI) and Controller which handles user input and controls the model and view.

T.

TextRecognizer

- Object from Google library that collects text information from images.

2. Requirements

2.1. User interface

As for the user interface we decided to have a Bottom Bar. It should provide three different options:

1. Browse in the receipt archive.
    a. The archive should provide different ways to filter the list of all receipts. One should be able to filter list by dates, categories and price.
    The user should also be able to add, edit and delete his/hers receipts.
2. Manage companies.
    a. User should be able to manage his or hers companies, add or delete a company, add new employees and comment on all of the mentioned. The comments will be shown when you select an employee or a company.
3. Manage suppliers.
    a. User should be able to manage all his or hers suppliers, add, edit or delete a supplier.

See appendix 5.1 for early mock-ups of the GUI and for the final version.

2.2. Functional requirements

2.2.1. Overview

The user should be able to take a picture of a receipt, the data found on the receipt will then will be collected and shown in applications wizard guide. User will then input additional data or missing data the application couldn't collect from the picture. All information will be saved as a receipt in a list and available for the user to manipulate if needed to.

The user should also be able to get an overview of the receipts, visually shown in a the archive. There is a list of receipts and user are able to mark receipts for deletion, pick one to amend it or simply sort the whole list by price or date or possibly filter the list of receipts by category. The picture of the "real" receipts will also be shown in the archive when choosing to view a particular receipt.
Lastly there is a possibility to view the companies under the company tab. There is information displayed about the company, list of employees and cards for the company, also there is the ability to delete or amend users and cards and also add comments to your company.

2.2.2. Table of application content

1. **Company**
    a. List of companies
        i. Add new company
        ii. Edit selected company
            1. Edit name
            2. Edit employees
            3. Edit suppliers
        iii. Delete company
2. **Archive**
    a. List of receipts
        i. Add new receipt
        ii. Edit selected receipt
            1. Edit data
        iii. Delete selected receipt
        iv. Sort receipts
        v. Filter receipts
3. **Suppliers**
    a. List of suppliers
        i. Add new supplier
        ii. Edit selected supplier
        iii. Delete selected supplier

2.3. Non-functional requirements

2.3.1. Usability

The application will provides easy usability and run with swedish as primary language. The wizard should lead user effortlessly through the process of adding a new receipt. The introduction when you start the application will provide all information needed so no prior knowledge is needed for usage.

2.3.2. Reliability

Users will rely on the applications ability to guarantee all images are saved as they contain necessary data for end-of-month bookkeeping. Images of receipt will be saved locally on the mobile device providing the user to access them if needed from outside the application.

2.3.3. Performance

Users interaction with the application should not result in noticeable stuttering or delays. There will be a loading bar when the application reads saved data. There will also be a loading bar when TextRecognizer is collection data from an image. This is necessary to allow the application to load all accessible data.

2.3.4. Supportability

This application is supported by Android devices running Nougat 7.1 with API 25 or higher. It should hopefully be available for download on Google Play.

The source code will provide tests for the model. GUI should be tested manually while running the application.

2.3.5. Implementation

The application is written in Java using Android Studio IDE. Gradle build system will build the application in order to run it.

2.3.6. Packaging and installation

The application will run on in Android Studio and an Android device or emulator. See 2.3.4. Supportability for detailed information.

2.3.7. Legal

All libraries in use are distributed with the Apache 2.0 license. There will exist documentation in the code to inform what library are in use.

3. Use cases

See appendix 5.2 for Use Cases.

3.1. Use case priority

Below are Use Cases listed in order to priority.

High priority Use Cases:

1. Company
2. Archive

Medium priority Use Cases:

1. Supplier

Low priority Use Cases:

1. Sort
2. Filter

4. Domain model

See appendix 5.3 for Domain model.

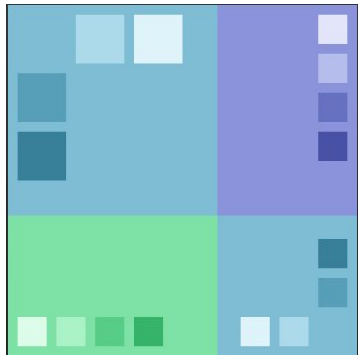
4.1 Class responsibilities

Category: Enum class. Holds categories for products.
Product: Holds information about the purchased item.
Comment: Holds information about users comments. Gets and sets a comment
Supplier: Holds information about users supplier.
Receipt: Holds products, date and the total amount of a receipt which has been collected
Employee: Holds the name, comments and purchases an employee has made.
Purchase: Sets what type of purchase have been made.
Company: Holds a company's employees, cards and comments.
User: The user of the application, holds its companies and suppliers.
Card: Holds information about the card used for the purchase.

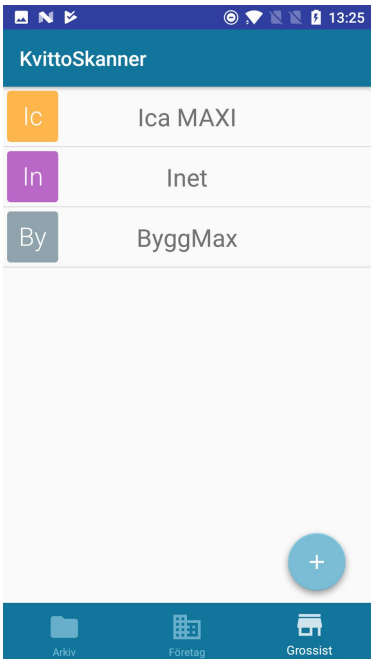
5. Appendix

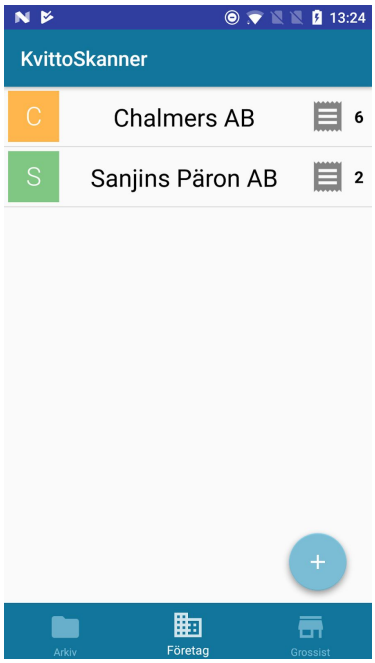
5.1 Graphical User Interface

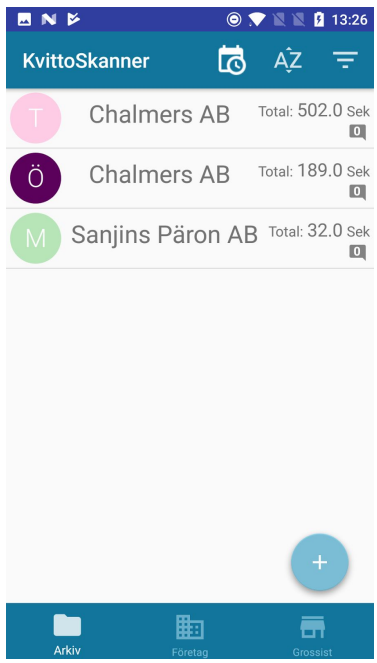
Below is colorscheme and first mock-ups of our GUI.



Below is the final picture of how the GUI looks as of today. It ended up with only three choices in the bottom bar instead of five as was the plan to start with.







5.2 Use-Cases

Please be aware that for the sequence maps coming up under two of the use cases there is bigger versions in section 5.3

UC: Archive

Summary:

Overview of all your registered receipts. Includes a floating action button for adding new receipts.

Priority:

High

Extends:

Bottom bar.

Includes:

- List of receipts
- Single GUI representation of a receipt.

Participators:

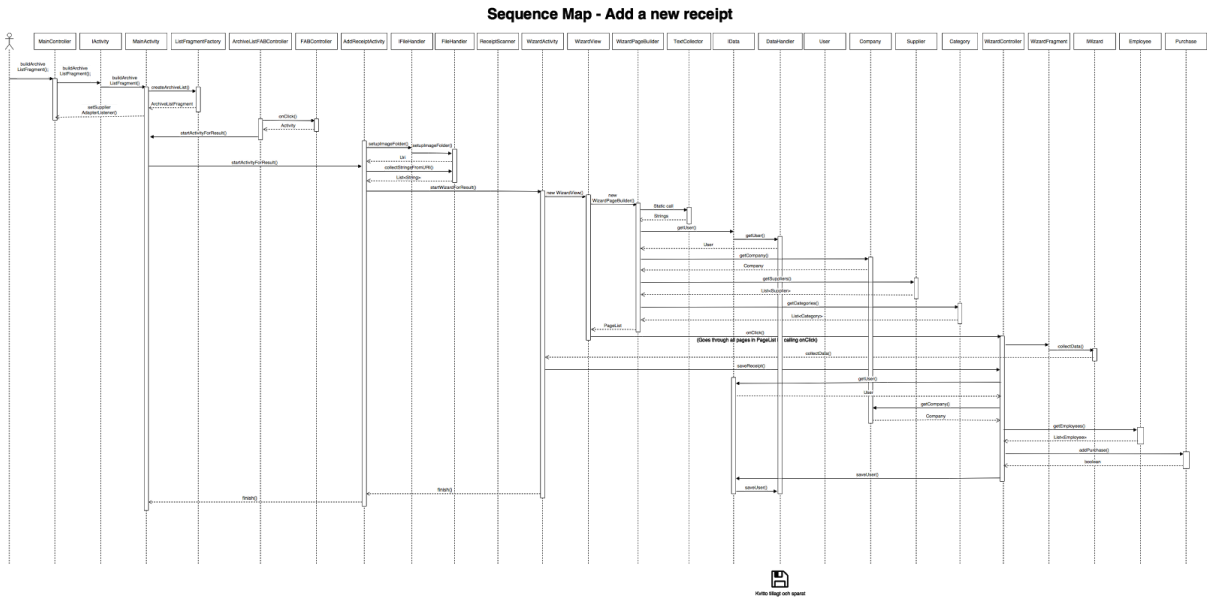
App user

Normal flow - Add a new receipt

	Actor	**System**
1	Press archive	
2		Display: - List of receipt. - Floating button
3	User presses button	
4		Display: 1. Button expands and shows 3 button options: a. Take new picture b. Choose image from gallery c. No image
5	User selects one of three options, a, b or c.	
6		App initiates wizard no matter previous choice. Display: - Type of cards.
7	User selects type card. Next button is pressed or swipe is used to continue [1].	
8		Display: - List of companies.
9	User selects company. [1]	
10		Display:

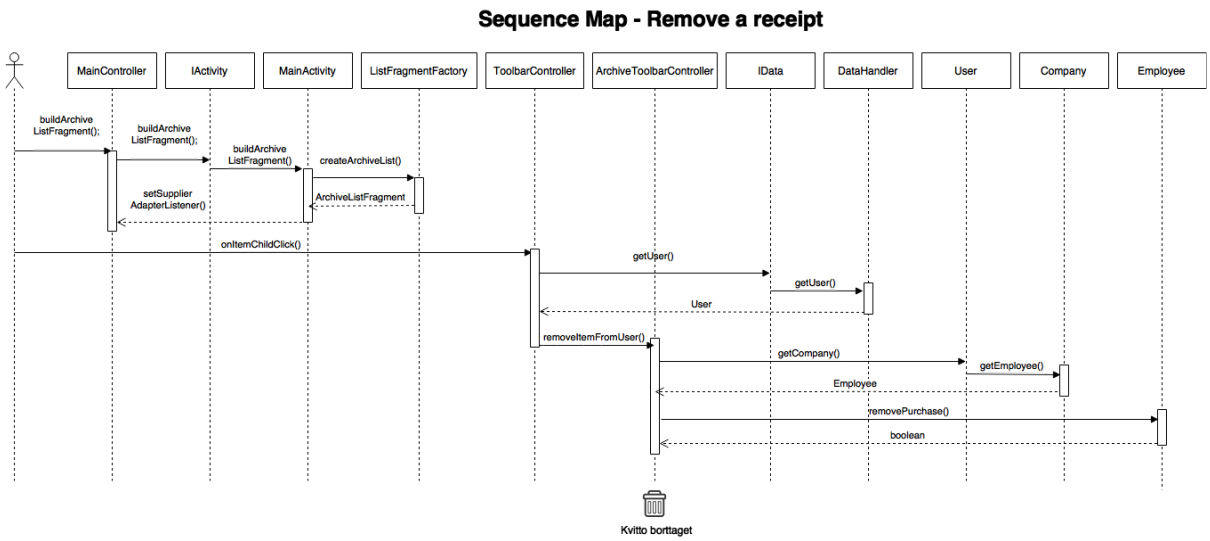
		- List of suppliers.
11	User selects supplier. [1]	
12		Display: - Date
13	User validates or edits date. [1]	
14		Display: - Total price
15	User validates or edits price. [1]	
16		Display: - Tax
17	User validates or edits tax. [1]	
18		Display: - List of categories.
19	User picks one category. [1]	
20		Display: - Textfield for comment.
21	User inputs optional comment. [1]	
22		Display: - Summary of all choices is shown.

23	User confirms choices with “Slutför” button.	
24		Display: - Popup, confirm “Slutför” button.
25	User confirms.	
26		New receipt is added. Display: - “Kvitto sparat”



Alternate flow - Remove receipt

	Actor	**System**
1	User selects one or several receipt from list of receipts.	
2		Action bar at the top toggles appearance. A backwards, trashcan and share button appears.
3	User selects trashcan.	
4		Display: - Popup shows: “Är du säker på att du vill radera valda kvitton?”
5	User confims,	
6		Display: - “Kvitton borttagna”



Alternate flow - Edit receipt

	Actor	**System**
1	User selects one or several receipt from list of receipts.	
2		Display: - GUI representation of a receipt with editable textfields. Also a button to view image of receipt and save changes.
3	User edits some information and presses save button.	
4		Display: - “Dina ändringar har sparats” App return to list of receipts.

Alternate flow - Sort list of receipts

	Actor	**System**
1		Display: 1. Filter options: a. Company b. Employee c. Category 2. Sorting options: a. Date b. Price (Ascending/descending sorting option)
2	User selects filter and sorting option.	
3		Display: - Filtered and sorted list of receipts by users choice

UC: Company

Summary:
Shows the different companies you can interact with

Priority:
High.

Extends:
Bottom bar.

Includes:
List of companies.

Participators:
App user

Normal flow - Add a new company

	Actor	**System**
1	Press Company	
2		Display: - List of companies. - Floating button
3	User presses button	
4		Display: - Button takes user to add new company page
5	User inserts the demanded information. User presses save.	
6		Display: - “Ditt nya företag har skapats”

Alternate flow - Delete company

	Actor	**System**
1	Press company	
2		Display: - List of companies
3	User chooses company/companies	
4		Action bar at the top toggles appearance. A backwards and trashcan button appears.
5	User selects trashcan	
6		Display: - Popup shows: “Är du säker på att du vill radera valda företag?”
7	User confirms	
8		Display: - “Valda företag har raderats”

Alternate flow - Edit company

	Actor	**System**
1	Press company	
2		Display: - GUI representation of a company with two spinners for selecting Employees or Cards. Also a textfield for comment.

3	User edits information by choosing employee or card and pressing “edit button”.	
4		Display: - “Dina ändringar har sparats”

Exceptional flow - Delete the only existing company

	Actor	**System**
1	Press company	
2		Display: - List of employee's companies shown
3	User chooses company	
5		Action bar at the top toggles appearance. A backwards, trashcan and share button appears.
6	User selects trashcan	
		Display: - "Du kan inte radera ditt enda företag"

UC: Supplier

Summary:
Shows the different suppliers

Priority:
Medium

Extends:
Bottom bar (App menu at the bottom of screen. Always visible)

Includes:
All suppliers for the users companies

Participators:
App user

Normal flow - Add a new supplier

	Actor	**System**
1	Press Supplier	
2		Display: - List of companies. - Floating button
3	User presses button	
4		Display: 1. Button takes user to add new supplier page
5	User inserts the demanded information. User presses save.	
6		Display: - “Din nya grossist har skapats”

Alternate flow - Delete supplier

	Actor	**System**
1	Press supplier	
2		Display: - List of suppliers
3	User chooses supplier/suppliers	
4		Action bar at the top toggles appearance. A backwards, trashcan and share button appears.
5	User selects trashcan	
6		Display: - Popup shows: “Är du säker på att du vill radera valda grossister?”
7	User confirms	
8		Display: - “Valda grossister har raderats”

Alternate flow - Edit supplier

	Actor	**System**
1	Press supplier	
2		Display: - Pop-up of supplier with editable textfield. Also a button to save changes.
3	User edits some information and presses save button.	
4		Display: - “Dina ändringar har sparats” App return to list of suppliers.

UC: Filter

Summary:
Puts a filter on list of receipts

Priority:
Medium

Extends:
Toolbar

Includes:
List of receipts in archive

Participators:
App user

Normal flow - Filter a list

	Actor	**System**
1	Press archive	
2		Display: - List of receipts is shown - Button for filteroptions is shown in toolbar
3	User chooses what to filter by	
4		Display: - List filtered by user's choice is shown

UC: Sort

Summary:
Sorts list of receipts (ascending/descending)

Priority:
Medium

Extends:
Toolbar

Includes:
List of receipts in archive

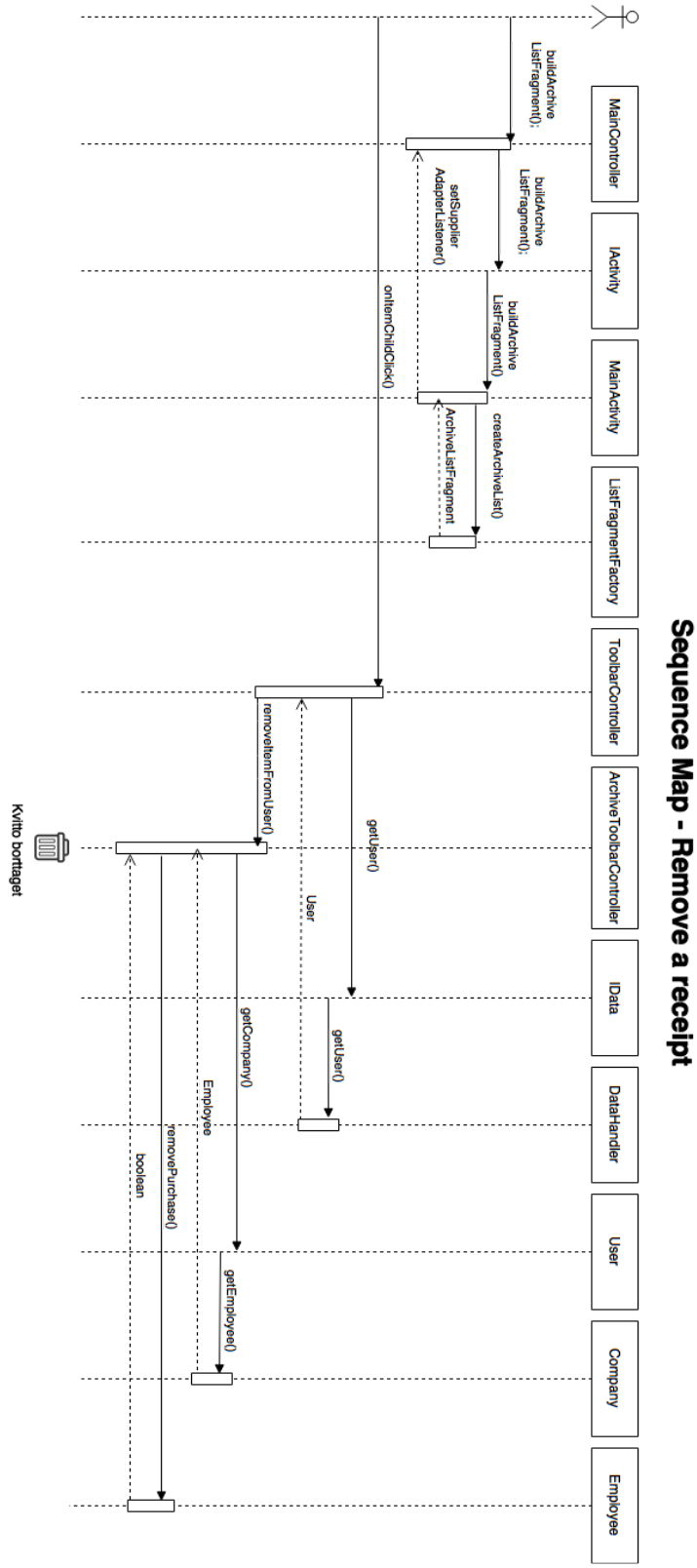
Participators:
App user

Normal flow - Sort a list

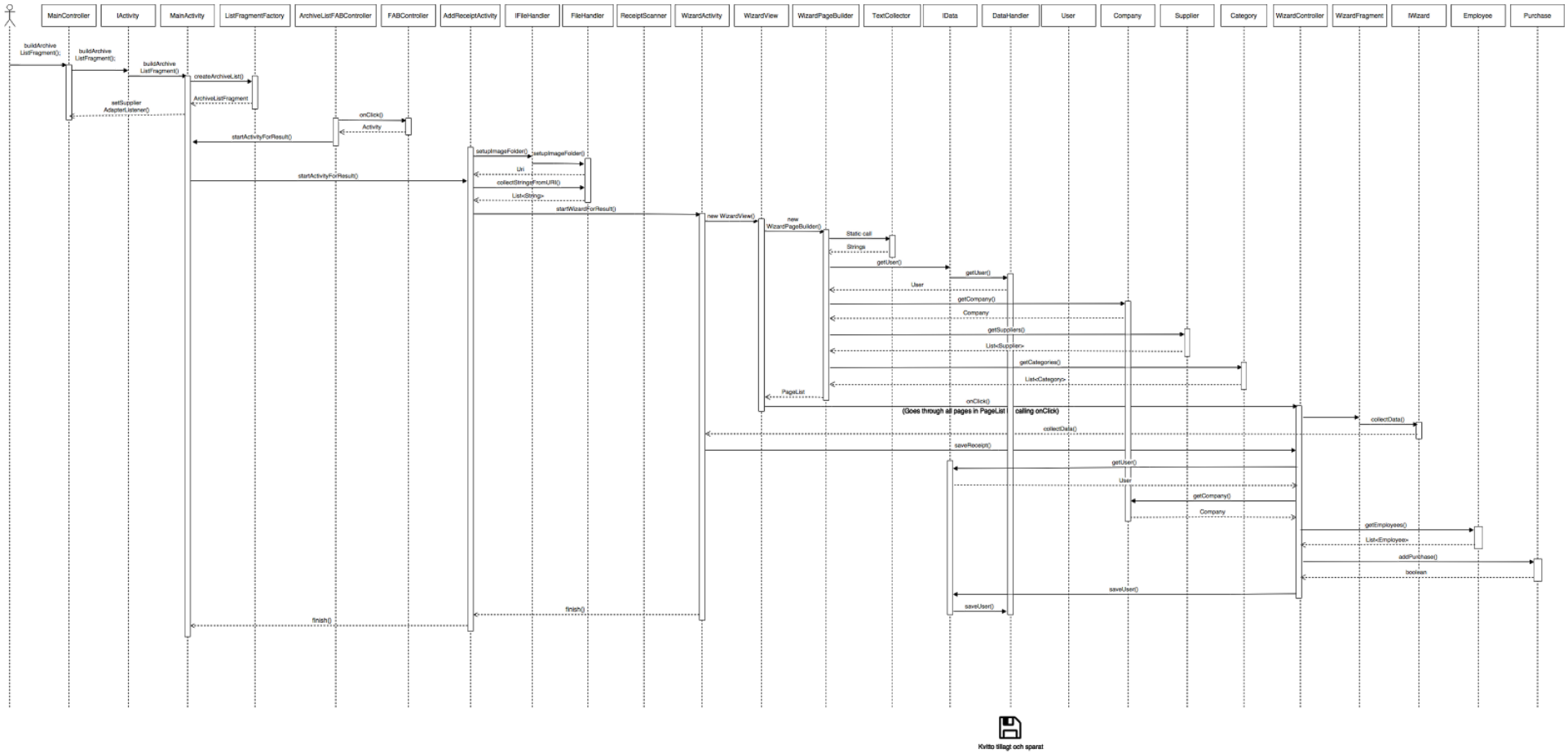
	Actor	**System**
1	Presses archive	
2		Display: - Shows list of receipts - Button in Toolbar to sort by date or by price
3	User chooses what to sort by	
4		Display: - List sorted by user choices is shown

5.3 Full Size Sequence Maps

Please see below for bigger versions of the sequence maps included in section 5.2



Sequence Map - Add a new receipt



5.4 Domain-Model