

# SE 3XA3: Software Requirements Specification PocketSaver

Team 12,

Mevin Mathew , mathem1 , 400012057

Shalmi Patel , patels19 , 400023762

Diya Mathews , mathewsd , 400014156

December 7, 2017

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Anticipated and Unlikely Changes</b>	<b>2</b>
2.1	Anticipated Changes . . . . .	2
2.2	Unlikely Changes . . . . .	2
<b>3</b>	<b>Module Hierarchy</b>	<b>2</b>
<b>4</b>	<b>Connection Between Requirements and Design</b>	<b>3</b>
<b>5</b>	<b>Module Decomposition</b>	<b>3</b>
5.1	Hardware Hiding Modules . . . . .	4
5.1.1	CustomImageCell Module(M1) . . . . .	4
5.1.2	TintedImage Module(M7) . . . . .	4
5.1.3	GroupedMasterPageItem Module(M13) . . . . .	4
5.1.4	MasterPageItem Module(M14) . . . . .	5
5.1.5	MainMasterDetailPage Module(M15) . . . . .	5
5.1.6	MainMenuPage Module(M16) . . . . .	5
5.2	Behaviour-Hiding Module . . . . .	5
5.2.1	ApiSV Model(M3) . . . . .	5
5.2.2	Home ViewModel Module(M4) . . . . .	5
5.2.3	Transaction ViewModel Module(M5) . . . . .	6
5.2.4	StroageSV Model(M6) . . . . .	6
5.2.5	TransactionList Cell Module (M9) . . . . .	6
5.2.6	TransactionListPage Module (M10) . . . . .	6
5.2.7	AboutPage Module(M11) . . . . .	6
5.2.8	HomePage Module (M12) . . . . .	6
<b>6</b>	<b>Traceability Matrix</b>	<b>7</b>
<b>7</b>	<b>Use Hierarchy Between Modules</b>	<b>7</b>

## List of Tables

<b>1</b>	<b>Revision History</b> . . . . .	<b>ii</b>
<b>2</b>	<b>Module Hierarchy</b> . . . . .	<b>4</b>
<b>3</b>	<b>Trace Between Requirements and Modules</b> . . . . .	<b>7</b>
<b>4</b>	<b>Trace Between Anticipated Changes and Modules</b> . . . . .	<b>7</b>

# List of Figures

1	Use hierarchy among modules . . . . .	8
---	---------------------------------------	---

Table 1: **Revision History**

Date	Version	Notes
November 9	1.0	Made an outline of all the modules in PocketSaver and started brainstorming the design
November 10	1.1	Finished the design document
December 6	1.2	Rev1 Submission

# 1 Introduction

Decomposing a system into modules is a commonly accepted approach to developing software. A module is a work assignment for a programmer or programming team (Parnas et al., 1984). We advocate a decomposition based on the principle of information hiding (Parnas, 1972). This principle supports design for change, because the “secrets” that each module hides represent likely future changes. Design for change is valuable in SC, where modifications are frequent, especially during initial development as the solution space is explored.

Our design follows the rules layed out by Parnas et al. (1984), as follows:

- System details that are likely to change independently should be the secrets of separate modules.
- Each data structure is used in only one module.
- Any other program that requires information stored in a module’s data structures must obtain it by calling access programs belonging to that module.

After completing the first stage of the design, the Software Requirements Specification (SRS), the Module Guide (MG) is developed (?). The MG specifies the modular structure of the system and is intended to allow both designers and maintainers to easily identify the parts of the software. The potential readers of this document are as follows:

- New project members: This document can be a guide for a new project member to easily understand the overall structure and quickly find the relevant modules they are searching for.
- Maintainers: The hierarchical structure of the module guide improves the maintainers’ understanding when they need to make changes to the system. It is important for a maintainer to update the relevant sections of the document after changes have been made.
- Designers: Once the module guide has been written, it can be used to check for consistency, feasibility and flexibility. Designers can verify the system in various ways, such as consistency among modules, feasibility of the decomposition, and flexibility of the design.

The rest of the document is organized as follows. Section 2 lists the anticipated and unlikely changes of the software requirements. Section 3 summarizes the module decomposition that was constructed according to the likely changes. Section 4 specifies the connections between the software requirements and the modules. Section 5 gives a detailed description of the modules. Section 6 includes two traceability matrices. One checks the completeness of the design against the requirements provided in the SRS. The other shows the relation between anticipated changes and the modules. Section 7 describes the use relation between modules.

## 2 Anticipated and Unlikely Changes

This section lists possible changes to the system. According to the likeliness of the change, the possible changes are classified into two categories. Anticipated changes are listed in Section 2.1, and unlikely changes are listed in Section 2.2.

### 2.1 Anticipated Changes

Anticipated changes are the source of the information that is to be hidden inside the modules. Ideally, changing one of the anticipated changes will only require changing the one module that hides the associated decision. The approach adapted here is called design for change.

**AC1:** The hardware on which the software is running.

**AC2:** Application user interface layout on various devices (Responsivity).

**AC3:** User interface appearance and colour scheme.

### 2.2 Unlikely Changes

The module design should be as general as possible. However, a general system is more complex. Sometimes this complexity is not necessary. Fixing some design decisions at the system architecture stage can simplify the software design. If these decision should later need to be changed, then many parts of the design will potentially need to be modified. Hence, it is not intended that these decisions will be changed.

**UC1:** Input device (On screen keyboard).

**UC2:** Output device (device screen).

**UC3:** Transaction entries updating and new entries logic.

**UC4:** Total expense calculation logic.

## 3 Module Hierarchy

This section provides an overview of the module design. Modules are summarized in a hierarchy decomposed by secrets in Table 2. The modules listed below, which are leaves in the hierarchy tree, are the modules that will actually be implemented.

**M1:** CustomImageCell Module

**M2:** Transaction Model Module

**M3:** ApiSV ~~Module~~ model

M4: HomePage ViewModel Module  
M5: TransactionPage ViewModel Module  
M6: ~~SettingsPage ViewModel Module~~  
M7: ~~PageActivityIndicator Module~~  
M8: TintedImage Module  
M9: TransactionListCell Module  
M10: TransactionListPage  
M11: AboutPage Module  
M12: HomePage Module  
M13: GroupedMasterPageItem Module  
M14: MasterPageItem Module  
M15: MainMasterDetailPage Module  
M16: MainMenuPage Module  
M17: [StorageSV model](#)

## 4 Connection Between Requirements and Design

The design of the system is intended to satisfy the requirements developed in the SRS. In this stage, the system is decomposed into modules. The connection between requirements and modules is listed in Table 3.

## 5 Module Decomposition

Modules are decomposed according to the principle of “information hiding” proposed by ?. The *Secrets* field in a module decomposition is a brief statement of the design decision hidden by the module. The *Services* field specifies *what* the module will do without documenting *how* to do it. For each module, a suggestion for the implementing software is given under the *Implemented By* title. If the entry is *OS*, this means that the module is provided by the operating system or by standard programming language libraries. Also indicate if the module will be implemented specifically for the software.

Only the leaf modules in the hierarchy have to be implemented. If a dash (–) is shown, this means that the module is not a leaf and will not have to be implemented. Whether or not this module is implemented depends on the programming language selected.

Level 1	Level 2
Hardware-Hiding Module	CustomImageCell Module TintedImage Module GroupedMasterPageItem Module MasterPageItem Module MainMasterDetailPage Module MainMenuPage Module
Behaviour-Hiding Module	Transaction Model Module ApiSV Module Home ViewModel Module Transaction ViewModel Module Settings ViewModel Module PageActivityIndicator Module TransactionList Cell Module TransactionListPage Module AboutPage Module HomePage Module
Software Decision Module	

Table 2: Module Hierarchy

## 5.1 Hardware Hiding Modules

### 5.1.1 CustomImageCell Module(M1)

**Secrets:** [Get](#) Colours from module

**Services:** Allows custom cells to have tinted images

**Implemented By:** CustomImageCell.cs

### 5.1.2 TintedImage Module(M7)

**Secrets:** [Get](#) Colours from module

**Services:** A custom image that allows for image colour tinting for the main menu images and various other pages

**Implemented By:** TintedImage.cs

### 5.1.3 GroupedMasterPageItem Module(M13)

**Secrets:** [How the MasterPageItem is grouped together](#) MasterPageItem grouped layout

**Services:** Groups the items in the master page into Modules for the MainMenuPage list

**Implemented By:** GroupedMasterPageItem.cs

#### 5.1.4 MasterPageItem Module(M14)

**Secrets:** How each page is created for each view

**Services:** Determines the instances of pages to be loaded upon user selection

**Implemented By:** MasterPageItem.cs

#### 5.1.5 MainMasterDetailPage Module(M15)

**Secrets:** ~~How the~~ Navigation flows from page to page using MasterDetail layout

**Services:** Displays the master detail format for the pages of the application Module that creates an instance of a master detail format for the pages

**Implemented By:** MainMasterDetailPage.xaml, MainMasterDetailPage.cs

#### 5.1.6 MainMenuPage Module(M16)

**Secrets:** ~~How the MainMenu is laid out~~ Layout of the MainMenu

**Services:** Module that displays the list of all pages within the application

**Implemented By:** MainMenuPage.xaml, MainMenuPage.cs

### 5.2 Behaviour-Hiding Module

#### 5.2.1 ApiSV Model(M3)

**Secrets:** ~~How the GET and POST requests are made~~ GET and POST requests

**Services:** API Service that allows the application to use GET and POST requests to retrieve and add data to the online database

**Implemented By:** ApiSV.cs

#### 5.2.2 Home ViewModel Module(M4)

**Secrets:** Data Retrieval from the ApiSV module

**Services:** Does all back end data retrieval for the Home page to be used within the Home-Page view

**Implemented By:** HomeViewModel.cs



### 5.2.3 Transaction ViewModel Module(M5)

**Secrets:** Data Retrieval from the ApiSV module

**Services:** Does all of the back end data retrieval for the TransactionPage view

**Implemented By:** TransactionViewModel.cs

### 5.2.4 StroageSV Model(M6)

**Secrets:** Data Retrieval from the ApiSV

**Services:** Saves any local settings created by the user to be used for the SettingsPage view

**Implemented By:** SettingsViewModel.cs

### 5.2.5 TransactionList Cell Module (M9)

**Secrets:** How the ListCell is formatted for the ListView in the TransactionListPage

**Services:** Displays the data for each transaction as a specific format determined by the developer

**Implemented By:** TransactionListCell.cs

### 5.2.6 TransactionListPage Module (M10)

**Secrets:** The data ~~is displayed~~ received from the Transaction ViewModel

**Services:** Displaying the data from the TransactionViewModel onto the UI View for the user

**Implemented By:** TransactionListPage.xaml, TransactionListPage.cs

### 5.2.7 AboutPage Module(M11)

**Secrets:** N/A

**Services:** Displays the title, version and brief description of the PocketSaver Application

**Implemented By:** AboutPage.xaml, AboutPage.cs

### 5.2.8 HomePage Module (M12)

**Secrets:** Data ~~is displayed~~ received from the HomePageViewModel to the UI view

**Services:** Displays the data from the HomePageViewModel to the UI View for the user

**Implemented By:** HomePage.xaml, HomePage.cs

## 6 Traceability Matrix

This section shows two traceability matrices: between the modules and the requirements and between the modules and the anticipated changes.

Req.	Modules
FR1	M9, M7
FR2	M9, M7, M11, M12
FR3	M7, M11, M12
FR4	M7

Table 3: Trace Between Requirements and Modules

AC	Modules
AC1	M1, M2, M4, M5, M6, M10, M16
AC2	M4, M5, M10, M11, M16
AC3	M4, M5, M10, M11, M16

Table 4: Trace Between Anticipated Changes and Modules

## 7 Use Hierarchy Between Modules

In this section, the uses hierarchy between modules is provided. Parnas1978 said of two programs A and B that A *uses* B if correct execution of B may be necessary for A to complete the task described in its specification. That is, A *uses* B if there exist situations in which the correct functioning of A depends upon the availability of a correct implementation of B. Figure 1 illustrates the use relation between the modules. It can be seen that the graph is a directed acyclic graph (DAG). Each level of the hierarchy offers a testable and usable subset of the system, and modules in the higher level of the hierarchy are essentially simpler because they use modules from the lower levels.

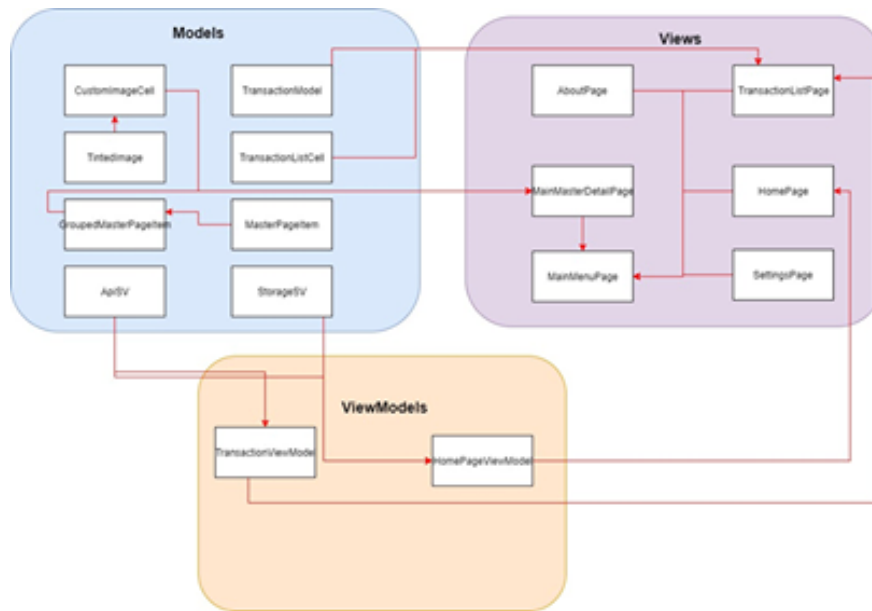


Figure 1: Use hierarchy among modules