SE 3XA3: Software Requirements Specification
Synergy Inventory Management System (SIMS)

Team #33, 'Sick Ideas'
Nathan Coit - 400022342
Lucas Shanks - 400029943
Cameron Van Ravens - 400020215

December 5, 2017

Contents

List of Tables

List of Figures

1 Introduction

This is the software Requirement Specification guide for the Synergy Inventory Management System (SIMS) project. The purpose of this project is to create an open source Inventory Management System, giving small companies and warehouses a simple to use and cost effective inventory management platform.

Decomposing a system into modules is a commonly accepted approach to developing software. Our group advocates a decomposition based on the principle of information hiding. This principle supports design for change, because the "secrets" that each module hides represent likely future changes. Design for change is valuable when developing software, where modifications are frequent, especially during initial development as the solution space is explored.

Our design principle 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.

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 8 describes the use relation between modules.

2 Anticipated and Unlikely Changes

During the design process for SIMS, design decisions made by the team members followed the principle of information hiding and designing for change. While the application will not likely stray from the original specifications, should an improved or new change arise, it

Table 1: **Revision History**

Date	**Version**	**Notes**
11/10/17	0.0	Initial Revision
12/6/17	1.0	Revision 1

will be easily implementable. The following sections lists possible changes to the relatively finalized functional implementation classified into two categories according to the likeliness of the change with Anticipated Changes located in Section 2.1 and Unlikely Changes listed in Section 2.2.

2.1 Anticipated Changes

During the design process, the design for change paradigm was used. Changes that are likely to happen to elements hidden in modules are identified as Anticipated Changes. These changes are easy to implement and do not affect the rest of the project.

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

AC2: The specific NodeJS, AngularJS, PostgreSQL and dependency versions

AC3: The standard web security protocols

AC4: Additional functionality of the inventory management system

AC5: The front-end interface design

2.2 Unlikely Changes

Design decisions that are likely to affect the rest of the system are classified as unlikely to change. The following design decision are unlikely to change, due to the risk of having to change multiple modules.

UC1: Input/Output devices (Input: File and/or Keyboard, Output: File, Memory, and/or Screen).

UC2: There will always be a source of input data external to the software.

UC3: The database schema design

UC4: The inventory analytics algorithms

UC5: The overall purpose of the system: Providing an Inventory Management System

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: Hardware-Hiding Module*

M2: Database Module

M3: OS Validation Module

M4: Data Validation Module

M5: Data Query Module

M6: User Authentication Module

M7: GUI Data Retrieval Module

M8: GUI Display Module

Note*: Hardware Hiding is not a module implemented in SIMS since it is software based. The lowest level that the software interfaces with the OS. This module, although not in the design, was included for legacy purposes, as it is a part of the Parnas and Madey four-variable model.

Level 1	**Level 2**
Hardware-Hiding Module	
Behaviour-Hiding Module	~~OS Validation Module~~ GUI Display Module Data Validation Module Database Module
Software Decision Module	Data Query Module User Authentication Module GUI Data Retrieval Module

Table 2: Module Hierarchy

4 Connection Between Requirements and Design

The design of the application is intended to satisfy the requirements developed in the SRS, more specifically designing for requirements. One example this can be seen in the implementation of user security in the non-functional requirements. From the requirement of allowing users to only view their own inventory and data and user security, with information hiding, designing for change, as well as security in mind, solutions can be created to protect each users personal information (password encryption, checking user authorization).

Further examples of the connection between the requirements and design can be found in in Table 3 showing connections between the requirements and modules. The system is designed to satisfy the requirements set in the SRS document.

5 Module Decomposition

The *Secrets* field in 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.

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.

5.1 Behaviour-Hiding Module

5.1.1 Database Module (M2)

Secrets: The format and structure of the save data.

Services: Stores/outputs user inventory data.

Implemented By: SIMS

5.1.2 OS Validation Module (M3)

Secrets: The algorithms determining how to set the proper file path.

Services: Determines user OS and configures file paths accordingly

Implemented By: SIMS

5.1.3 Data Validation Module (M4)

Secrets: The rules for the input: accepts valid input data.

Services: Converts the input data into the data structure used by the input parameters module.

Implemented By: SIMS

5.1.4 GUI Display Module (M8)

Secrets: The rules for the display: mapping the data to the output format.

Services: Interfaces with OS Validation to display and write contents

Implemented By: SIMS

5.2 Software Decision Module

Secrets: The design decision based on mathematical theorems, physical facts, or programming considerations. The secrets of this module are *not* described in the SRS.

Services: Includes data structure and algorithms used in the system that do not provide direct interaction with the user.

Implemented By: –

5.2.1 Data Query Module (M5)

Secrets: The format and structure of the input data.

Services: Converts the input data into the data structure used by the input parameters module.

Implemented By: SIMS

5.2.2 User Authentication (M6)

Secrets: User authentication token.

Services: Checks user login status and controls accessibility between users.

Implemented By: SIMS

5.2.3 GUI Data Retrieval Module (M7)

Secrets: The data structure/format.

Services: Takes in or builds data to be displayed or formatted. input parameters module.

Implemented By: SIMS

5.2.4 GUI Data Retrieval Module (M7)

~~[Secrets:]The data structure/format.~~

~~[Services:]Takes in or builds data to be displayed or formatted. input parameters module.~~

~~[Implemented By:] SIMS~~

6 Traceability Matrix

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

Trace Between Requirements and Modules

Req.	Modules
R1	M1, M2, M3
R2	M2, M4, M6
R3	M6, M8
R4	M2, M4, M7
R5	M4
R6	M2, M5, M6, M7, M8
R7	M3
R8	M6, M8
R9	M2

Table 3: Trace Between Requirements and Modules

7 Trace Between Anticipated Changes and Modules

AC	Modules
AC1	M1
AC2	M2, M5
AC3	M6, M7
AC4	M2, M4, M5, M6, M7, M8
AC5	M7, M8

Table 4: Trace Between Anticipated Changes and Modules

8 Use Hierarchy Between Modules

In this section, the uses hierarchy between modules is provided. That is, A *uses* B if there exists 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.

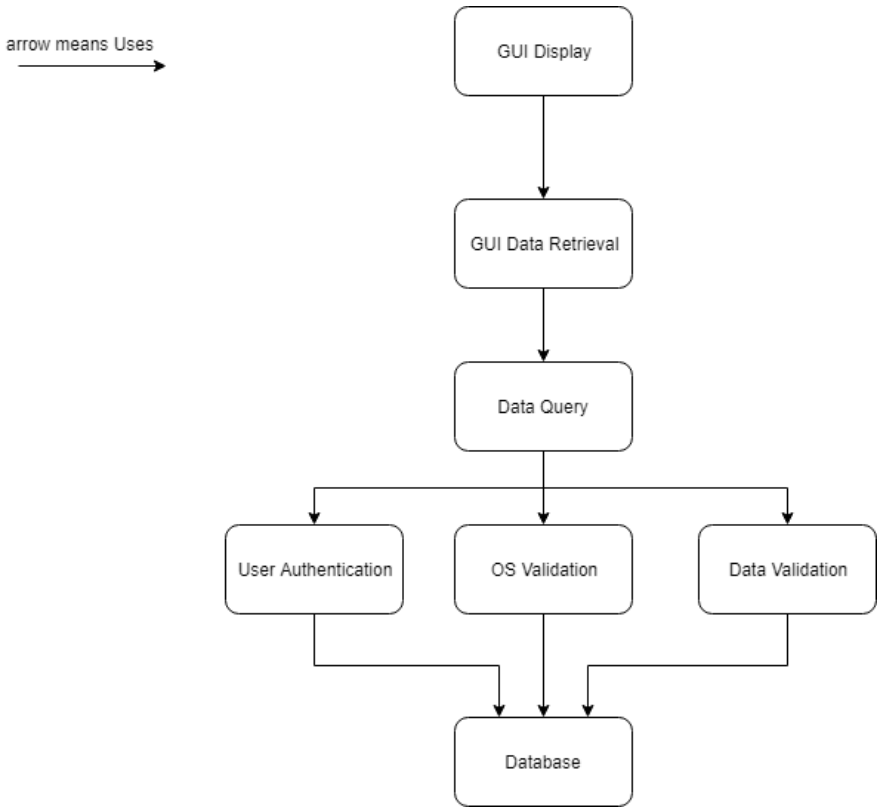
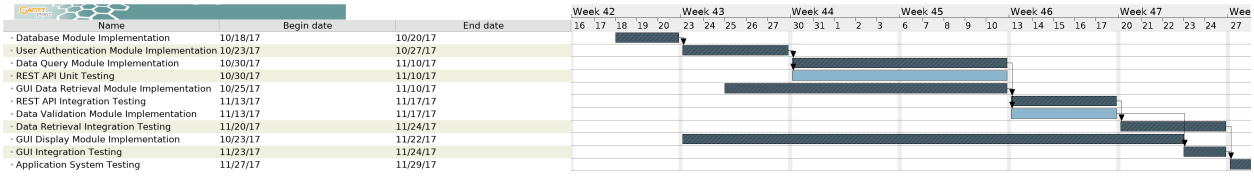


Figure 1: Use hierarchy among modules

The Gantt Chart located on the project page contains individual test role assignments while the Gantt chart below breaks the testing into a set of tasks.



Name	Begin date	End date
Database Module Implementation	10/18/17	10/20/17
User Authentication Module Implementation	10/23/17	10/27/17
Data Query Module Implementation	10/30/17	11/10/17
REST API Unit Testing	10/30/17	11/10/17
GUI Data Retrieval Module Implementation	10/25/17	11/10/17
REST API Integration Testing	11/13/17	11/17/17
Data Validation Module Implementation	11/13/17	11/17/17
Data Retrieval Integration Testing	11/20/17	11/24/17
GUI Display Module Implementation	10/23/17	11/22/17
GUI Integration Testing	11/23/17	11/24/17
Application System Testing	11/27/17	11/29/17

Figure 2: Updated Gantt Chart With Testing

References