Prepared by

Ivan White

Version 1.0

An overview of how this application is to be designed and built.

Software Design Document

Component App (Prototype)

Table of Contents

[1. Introduction 3](#_Toc15637626)

[1.1 What is being built 3](#_Toc15637627)

[1.2 Why it is being built 3](#_Toc15637628)

[1.3 How it is being built 3](#_Toc15637629)

[1.4 Development Platform 3](#_Toc15637630)

[1.5 Development Environment 3](#_Toc15637631)

[1.6 Deployment Environment 4](#_Toc15637632)

[1.7 Programming Language 4](#_Toc15637633)

[2. System Overview 5](#_Toc15637634)

[2.1 Scope and Purpose 5](#_Toc15637635)

[2.2 System Boundaries 5](#_Toc15637636)

[2.3 Functionalities 5](#_Toc15637637)

[2.4 Requirements 7](#_Toc15637638)

[2.5 Interface Design 8](#_Toc15637639)

[3. Development Methodology 8](#_Toc15637640)

[3.1 Influence on Development 8](#_Toc15637641)

[3.2 Testing 8](#_Toc15637642)

[3.2.1 Static Testing 8](#_Toc15637643)

[3.2.2 Dynamic Testing 8](#_Toc15637644)

[3.3 Frequency of software releases 8](#_Toc15637645)

[3.4 Milestones 9](#_Toc15637646)

[3.5 Prioritization of Work 9](#_Toc15637647)

[4. Detailed Design 9](#_Toc15637648)

[4.1 Modelling Techniques 9](#_Toc15637649)

[4.2 Architecture 9](#_Toc15637650)

[4.3 Presentation Layer 9](#_Toc15637651)

[Component App Version 1.0 9](#_Toc15637652)

[Component App Version 1.1 10](#_Toc15637653)

[4.4 Business Layer 13](#_Toc15637654)

[Component App Version 1.0 UML Class Diagram 13](#_Toc15637655)

[Component App Version 1.0 UMP Object Diagram 14](#_Toc15637656)

[Component App Version 1.1 UML Class Diagram 15](#_Toc15637657)

[Component App Version 1.1 UML Object Diagram 16](#_Toc15637658)

[4.5 Data Layer 16](#_Toc15637659)

[4.6 Program Structure 17](#_Toc15637660)

[Component App Version 1.0 17](#_Toc15637661)

[Component App Version 1.1 18](#_Toc15637662)

[4.7 Data Structures 18](#_Toc15637663)

[4.8 Control Process 19](#_Toc15637664)

[Component App Version 1.0 19](#_Toc15637665)

[Component App Version 1.1 20](#_Toc15637666)

[4.9 Dataflow 21](#_Toc15637667)

[Component App Version 1.0 21](#_Toc15637668)

[Component App Version 1.1 22](#_Toc15637669)

[5. Code Elaboration 23](#_Toc15637670)

# 1. Introduction

## 1.1 What is being built

We are building an inventory management CLI application to keep track of software and hardware stock for the client.

## 1.2 Why it is being built

Currently the client TC has no formal computer based inventory management system for the numerous hardware and software components which they sell to customers from their 10 nationwide stores. The prototype described in this SRS will have initial limited functionally and will be used to derive further requirements and propose different platforms, features and interfaces for which future version may be designed. The requirements in this document will be used to develop a CLI based ‘Component App’ prototype. The prototypes developed will give rise to a later SRS documents with specifications for potential GUI desktop, web and mobile interfaces incorporating a centralised database and more advanced features. Contracts have been drawn up and this will be an ongoing project until the project team and the client TC have elicited and defined all requirements of the final software solution.

## 1.3 How it is being built

It is to be built using the Java programming language and the Eclipse IDE.

## 1.4 Development Platform

|  |  |
| --- | --- |
| Machine name | Dell OptiPlex 7010 |
| CPU | Intel Core i3-2120 Dual Core Processor at 3.3GHz (LGA 1155) |
| Motherboard | Dell 0773VG A00 |
| RAM | 4GB DDR3 |
| Graphics | Intel HD Graphics 2000 |
| Storage | 160GB HDD |
| Wireless LAN | TP-LINK Wireless USB Adapter |
| Ethernet | Intel 82579LM Gigabit Network Connection |
| Peripherals | Monitor, Keyboard, Mouse |

## 1.5 Development Environment

Eclipse

## 1.6 Deployment Environment

|  |  |
| --- | --- |
| Machine name | Dell OptiPlex 7010 |
| CPU | Intel Core i3-2120 Dual Core Processor at 3.3GHz (LGA 1155) |
| Motherboard | Dell 0773VG A00 |
| RAM | 4GB DDR3 |
| Graphics | Intel HD Graphics 2000 |
| Storage | 160GB HDD |
| Wireless LAN | TP-LINK Wireless USB Adapter |
| Ethernet | Intel 82579LM Gigabit Network Connection |
| Peripherals | Monitor, Keyboard, Mouse |

## 1.7 Programming Language

Java

# 2. System Overview

## 2.1 Scope and Purpose

The main goals of the software are to provide a fast convenient way for TC employees to find and view component information. There is currently no centralized component management system in the company that can do this effectively. Authorized admins will also have the ability to update component information as required however this admin functionality may not be introduced until later versions.

The first versions of the prototype software application should store information about various computer components and allow the user to iterate (browse) through the individual components with the option to view details of each component. All functional requirements are explained in detail in the requirements sections later in this SRS.

## 2.2 System Boundaries

The prototype described in this SRS will only simulate the data in the software using a hard-coded components list, which will need to be generated as part of the prototype design. The High Level class diagram in Appendix B: Analysis Models, will provide clarification of how the system data is intended to be created in the prototype.

## 2.3 Functionalities

|  |  |  |
| --- | --- | --- |
| **Requirement ID** | **Description** | **Priority** |
| FR1 | User can log into to the system. | Low |
| FR2 | The user can issue commands by input of specific number(s) based on presented menu options and pressing Enter on the keyboard. | **Must** |
| FR3 (Dependant on REQ2) | The PS shall display and respond to a user selection from a menu of options to include the below: | **Must** |
| FR4 | Menu option to View All Components | **Must** |
| FR5 | Menu option to View component lists by category | Medium |
| FR6 | Menu option to View component lists by sub category | Medium |
| FR7 | Menu option to View a specific component by SKU | **Must** |
| FR8 | Menu Option to Exit the system should always be available. | **Must** |
| FR9 | Menu Option to go back to the main menu should always be available (where applicable). | **Must** |
| FR10 | The PS shall allow the user (where applicable) to navigate a shown list of components using the following presented options: | Medium |
| FR11 | First (Show all details of the first component on the list including the specific properties of the selected components sub category). | Medium |
| FR12 | Next (Show all details of next component on the list including the specific properties of the selected components sub category). | Medium |
| FR13 | Previous (Show all details of previous component on the list including the specific properties of the selected components sub category). | Medium |
| FR14 | Last (Show all details of last component on the list including the specific properties of the selected components sub category). | Medium |

## 2.4 Requirements

|  |  |  |
| --- | --- | --- |
| **Requirement ID** | **Description** | **Priority** |
| REQ1 | The PS shall be represented by a Command Line Interface | **Must have** |
| REQ2 | The PS shall generate a hard coded List (or other suitable collection structure) of various hardware and software components in stock for development and testing purposes. It is suggested that a static class which has a method to return a list/collection structure of components should be used to fulfil this requirement. The components should be of different types that correspond to the component types and categories as per the high level class diagram in Appendix B, | **Must have** |
| REQ3 | Each component shall have the following properties: name, price, SKU (unique identifier) and manufacturer | **Must have** |
| REQ4 | The components shall be grouped into 2 main categories: Hardware and Software. | **Must have** |
| REQ5 | Hardware category components should have a property to hold their mass in kilograms. | **Must have** |
| REQ6 | Software category components should have a property to hold a Version number | **Must have** |
| REQ7 | Hardware and software categories are further divided in to subclass categories (As per the high Level diagram in Appendix B.) | **Must have** |
| REQ8 | Monitors must have a property for their Diagonal Size in inches. | **Must have** |
| REQ9 | Keyboards must have with a property to indicate if it is a QWERTY keyboard (displayed as Yes or No). | **Must have** |
| REQ10 | Operating Systems will require a property to hold the Platform Name (which it runs on). | **Must have** |
| REQ11 | Applications with require a property to hold the size of the application in Megabytes. | **Must have** |
| REQ12 | Games will require a property to hold their PEGY Rating. | **Must have** |

## 2.5 Interface Design

The prototype software will only involve a CLI interface for now which can run on a desktop environment as per the given client operating environment. The CLI will display menu options for available functions at different points in the flow of the program and allow the user to make menu selections via keyboard input.

# 3. Development Methodology

## 3.1 Influence on Development

The agile development methodology will be used to create the Components app. As the application is split into 3 versions (1.0, 1.1, 1.2), agile will allow us to develop the application in increments and release them to the customer when an increment is done. If the customer requires a change to be made to the application at any stage, this should be easy to accomplish.

## 3.2 Testing

Testing has been broken down into two categories – static and dynamic.

### 3.2.1 Static Testing

Informal Reviews

All documentation will be subjected to an informal review and proofread to find any potential errors in it.

Technical Reviews

The requirements specification document has already had a technical review done by the team members involved. The detailed design specification will also be reviewed when it is completed.

Walkthrough

As I am the sole developer of the product being developed, a walkthrough is not necessary.

Inspection

The tutor will carry out inspections during various stages of document completion.

Static Code Review

The tutor will carry out code inspections to ensure they comply with coding standards and that the code is well optimized.

### 3.2.2 Dynamic Testing

Functional and non-functional testing will be carried out at the end of each increment of software completion. Test plans, logs, and reports will be used.

## 3.3 Frequency of software releases

The development will be split into two 2-day sprints. At the end of each sprint, a version of the software will be released.

## 3.4 Milestones

Completion of sprint 1 will result in version 1.0 being built and tested.

Completion of sprint 2 will result in version 1.1 being built and tested.

## 3.5 Prioritization of Work

Please see the table in the System Overview Section > Functionalities sub section of this document to see what functionalities of the Component App should be prioritized.

# 4. Detailed Design

## 4.1 Modelling Techniques

We will be using UML to model the Architecture of the Component App. The modelling has been broken down into the following:

* Program Structure – see diagram below
* Control Processes - see UML Activity diagram below
* Dataflow - see UML Dataflow diagram below
* Class Diagrams: - see UML class diagram below to see the structure of the classes within the program and their relationship with each other

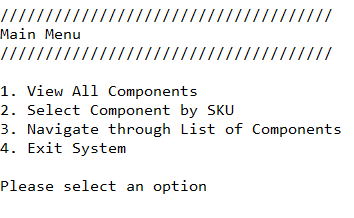
## 4.2 Architecture

The program will be built using a 3-tier model (presentation layer, business layer, and data layer. If the program ever needs to be converted from a command line interface to a GUI interface in future, only the presentation layer needs to be changed. The data layer will be hard coded in the version that is now being built, however the 3 tier model would also allow for the data layer to be switched over to a database in potential future versions without affecting the presentation and business layers.

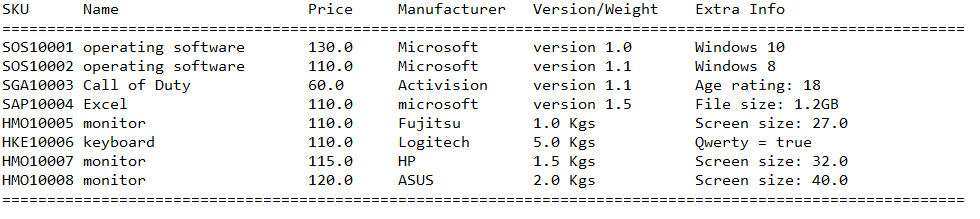
## 4.3 Presentation Layer

### Component App Version 1.0

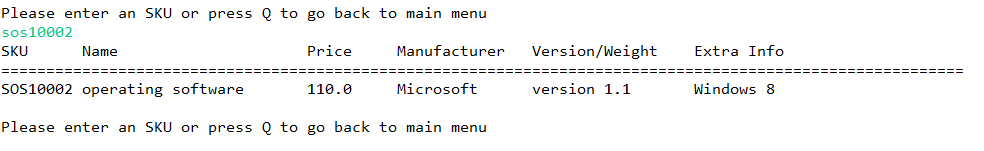
The user will be presented with the main menu when they open the program as shown below



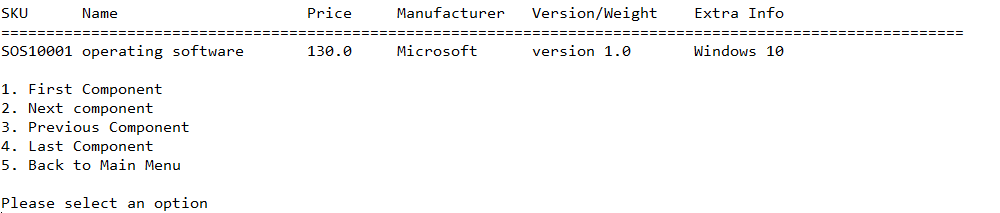
Selecting option 1 in the main menu displays the full list of components to the user in the format shown below



Selecting option 2 from the main menu allows the user to select a component by SKU through the interface shown below



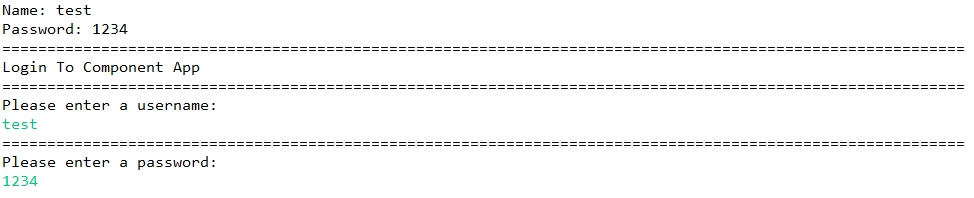
Selecting option 3 from the main menu will automatically display the first component in the list to the user. The user can then navigate their way through the list as per the options shown below. The selected component will then automatically display to the user.



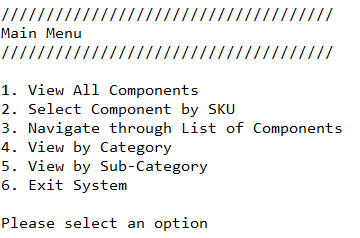
Option 4 in the main menu will immediately exit the program.

### Component App Version 1.1

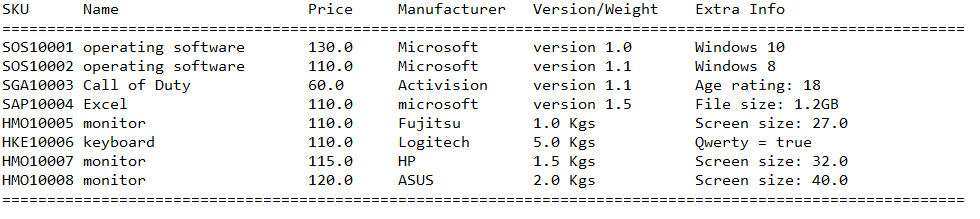
Upon opening version 1.1 of the program the user will be prompted to enter their login credentials as follows. Note that as this is just a simulated login system for this version, the user will be given hints of the correct username and password at the top of the screen for testing purposes. These hints are to be removed when the full login system is implemented in version 1.2.



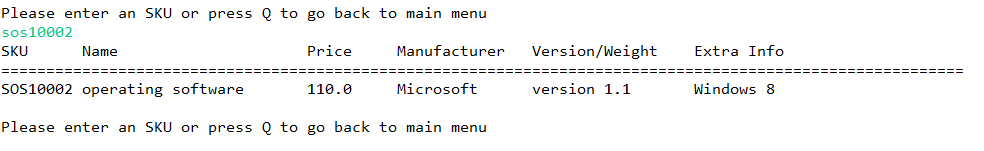
When the user logs into the system they are presented with the main menu as shown below



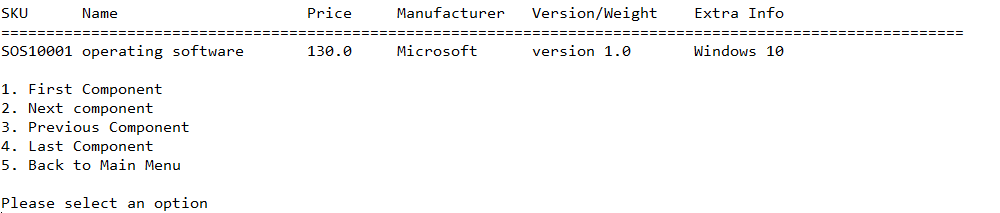
Selecting option 1 in the main menu displays the full list of components to the user in the format shown below



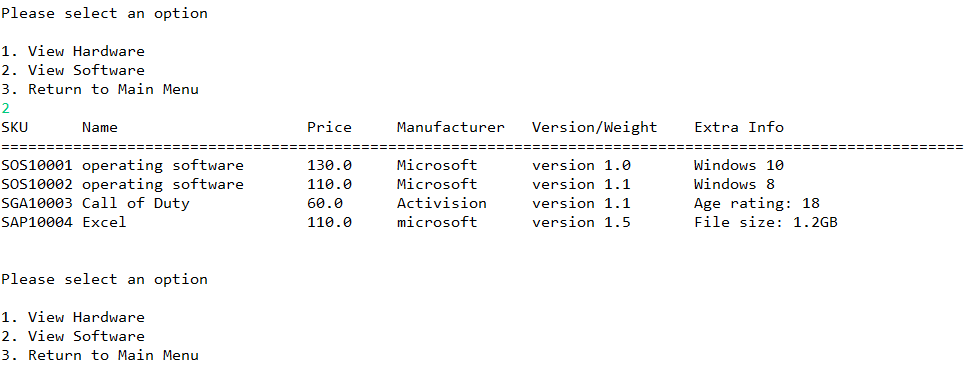
Selecting option 2 from the main menu allows the user to select a component by SKU through the interface shown below



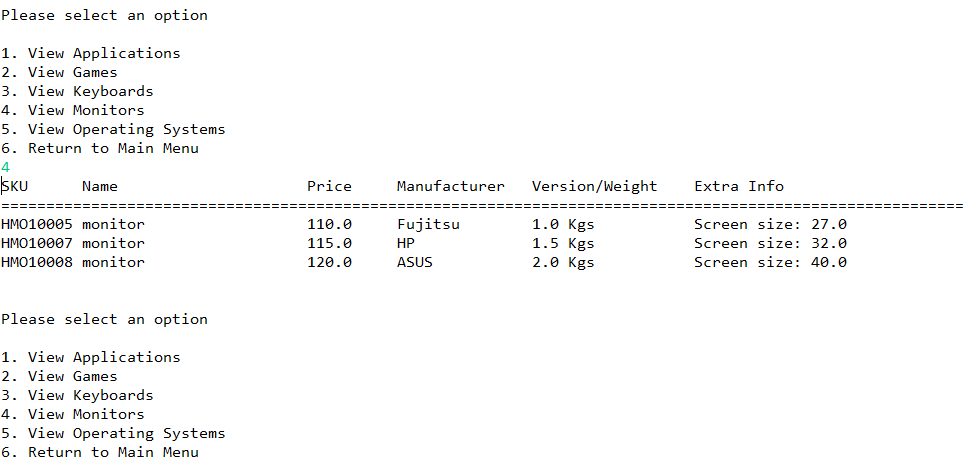
Selecting option 3 from the main menu will automatically display the first component in the list to the user. The user can then navigate their way through the list as per the options shown below. The selected component will then automatically display to the user.



Option 4 in the main menu will allow the user to choose whether would like to view all hardware or all software components. In the example below, the user has chosen to view all software components.



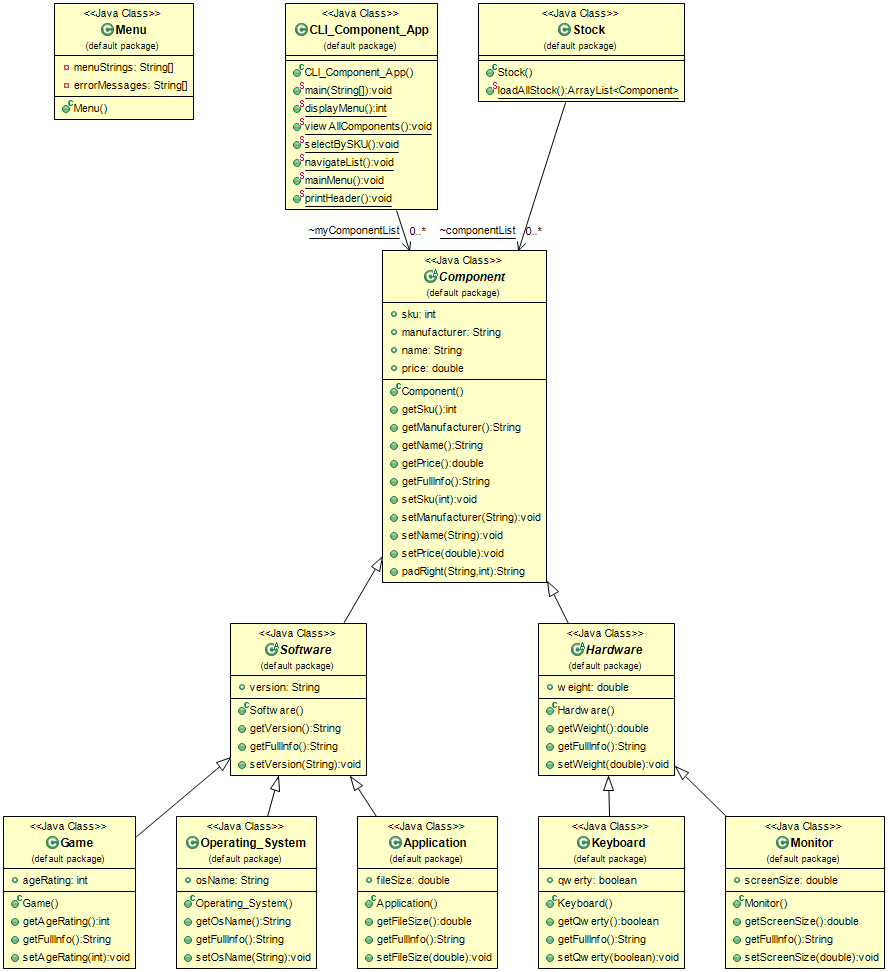
Option 5 in the main menu will allow the user to view components via sub-category. In the example below the user has chosen to view all monitors.



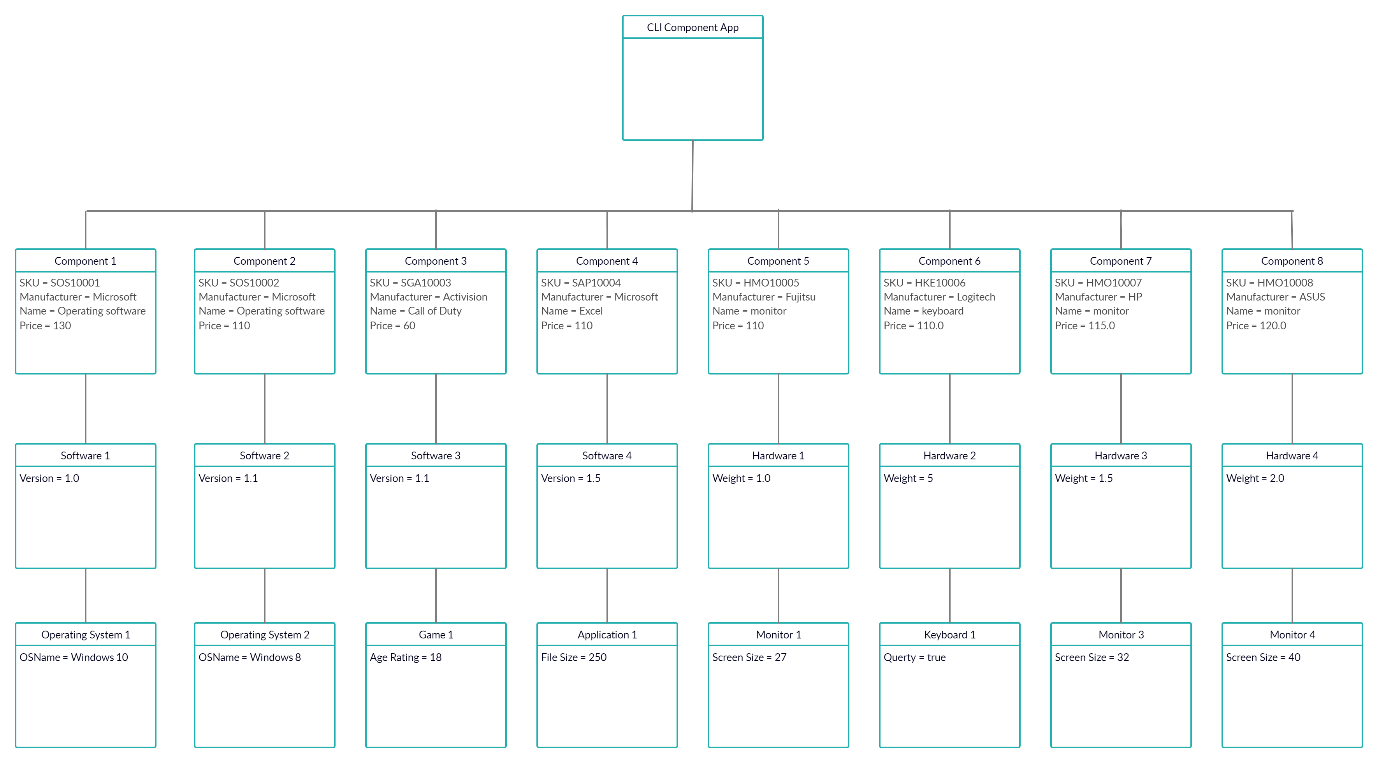
Option 6 in the main menu will immediately exit the program.

## 4.4 Business Layer

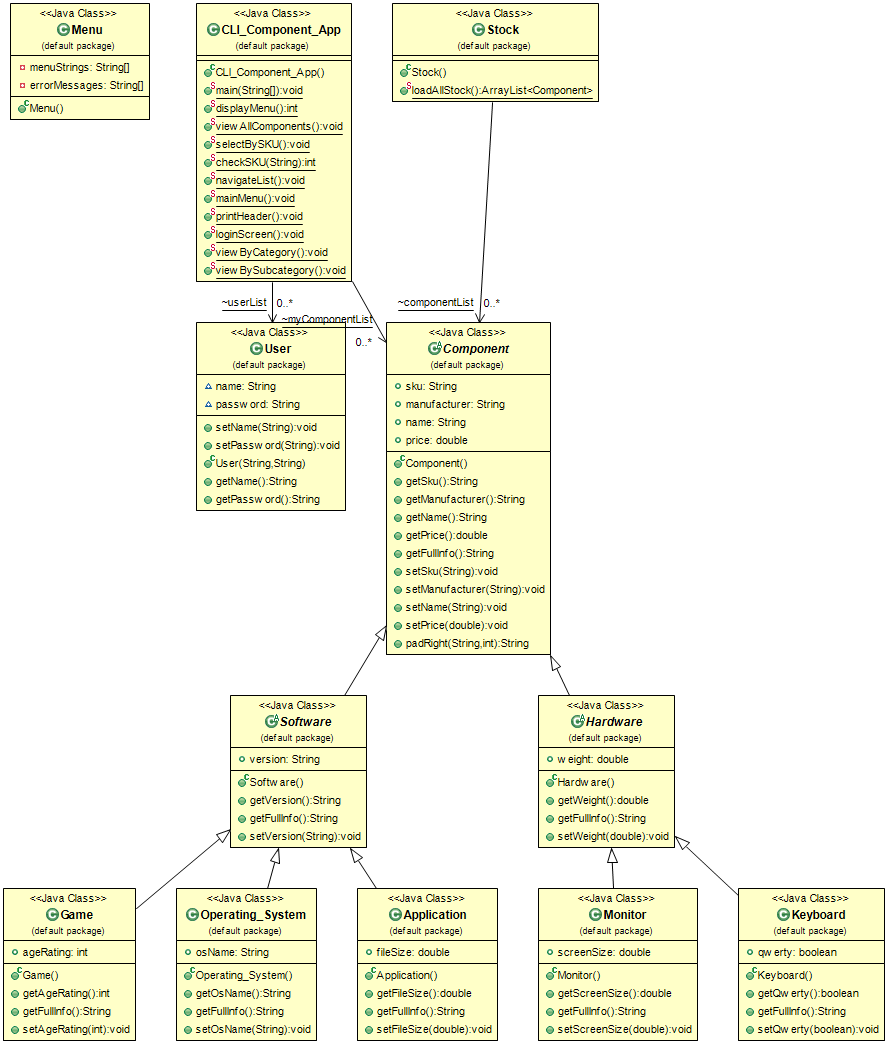
### Component App Version 1.0 UML Class Diagram



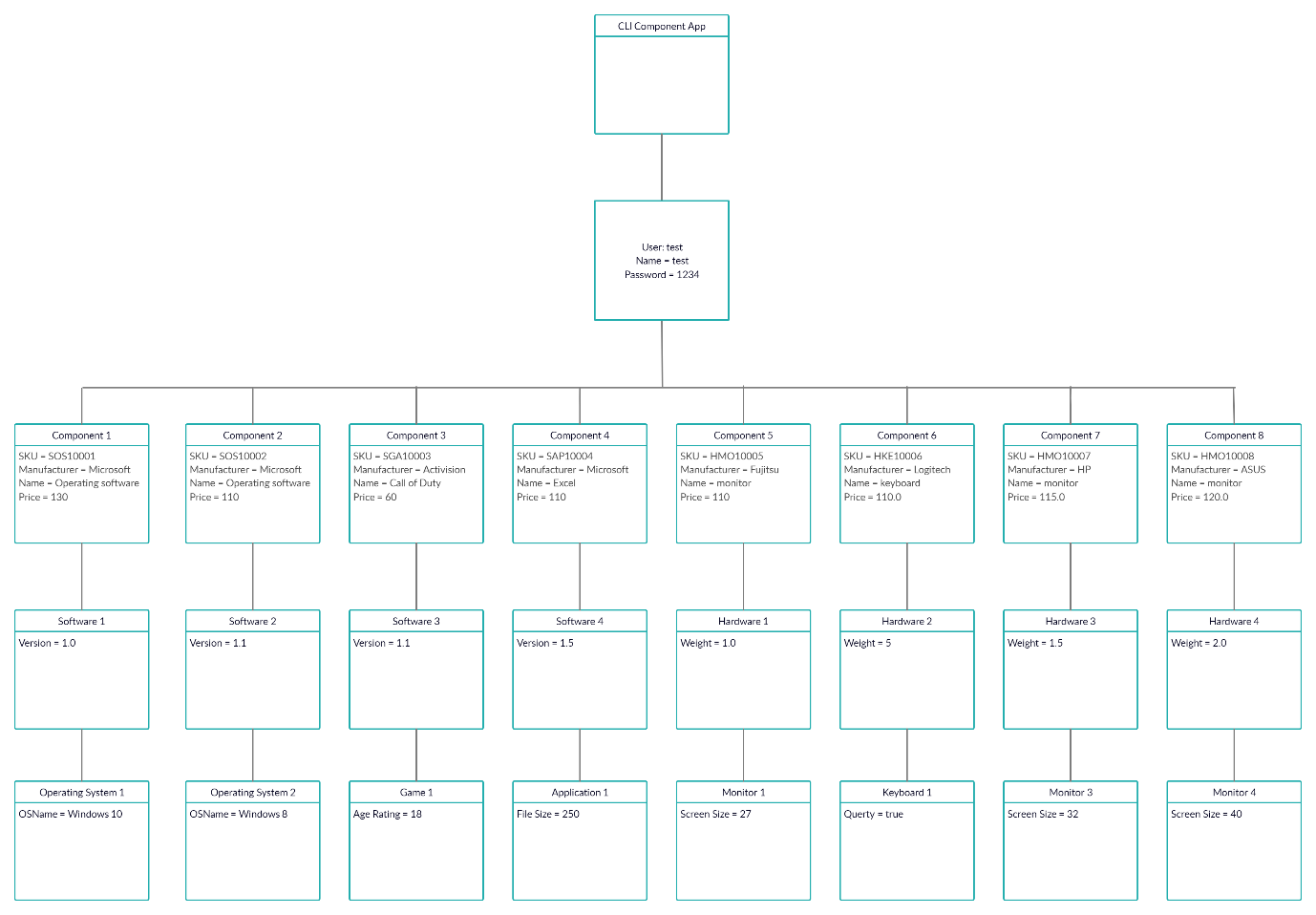
### Component App Version 1.0 UMP Object Diagram



### Component App Version 1.1 UML Class Diagram



### Component App Version 1.1 UML Object Diagram

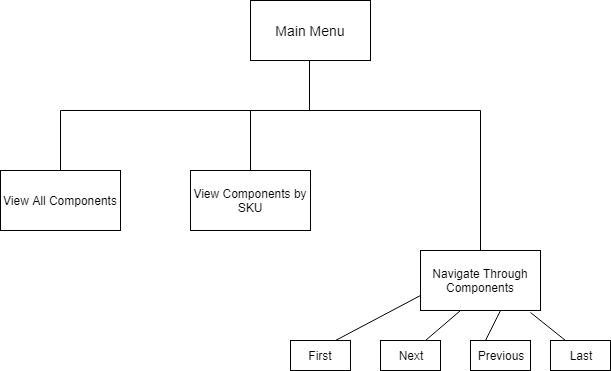


## 4.5 Data Layer

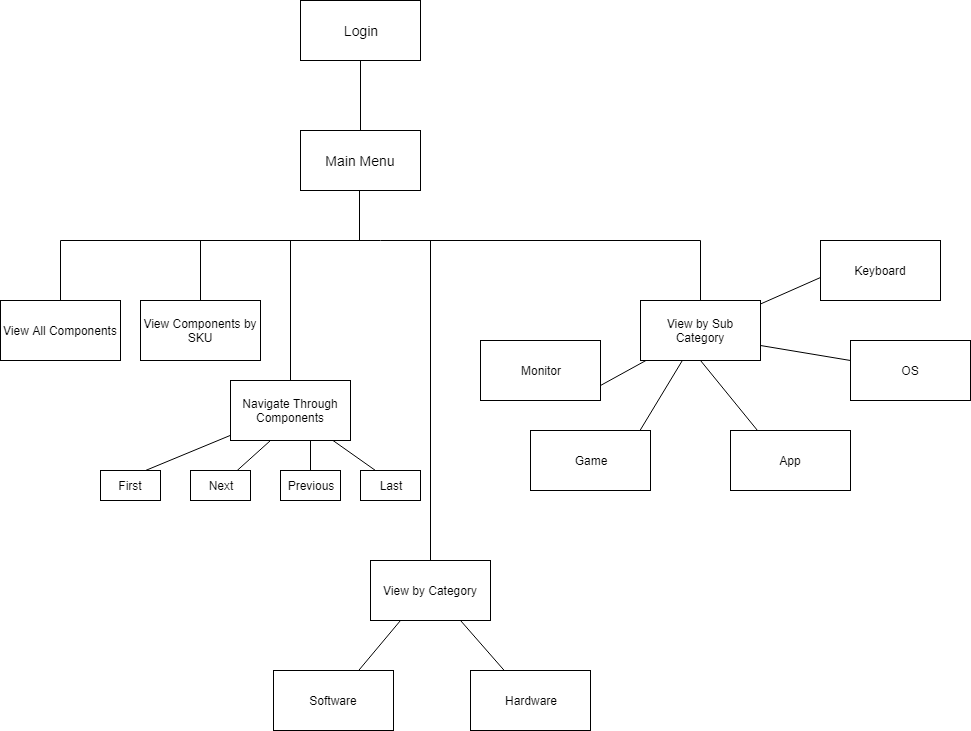
The component app does not have a database. Instead, the data is hard coded into the application and is contained within the stock class in the UML diagram above.

## 4.6 Program Structure

### Component App Version 1.0



### Component App Version 1.1



## 4.7 Data Structures

The data structure of the program will consist of an arraylist of component objects. These objects will be derived from their parent classes (either hardware or software), which in turn are derived from the Component parent class. These are subdivided into other classes – keyboard and monitor for hardware, and application, game, and operating system for software.

The component list will be generated in the stock class and will be used in the Main method in the CLI component class. Each component class will have its own unique fields as follows:

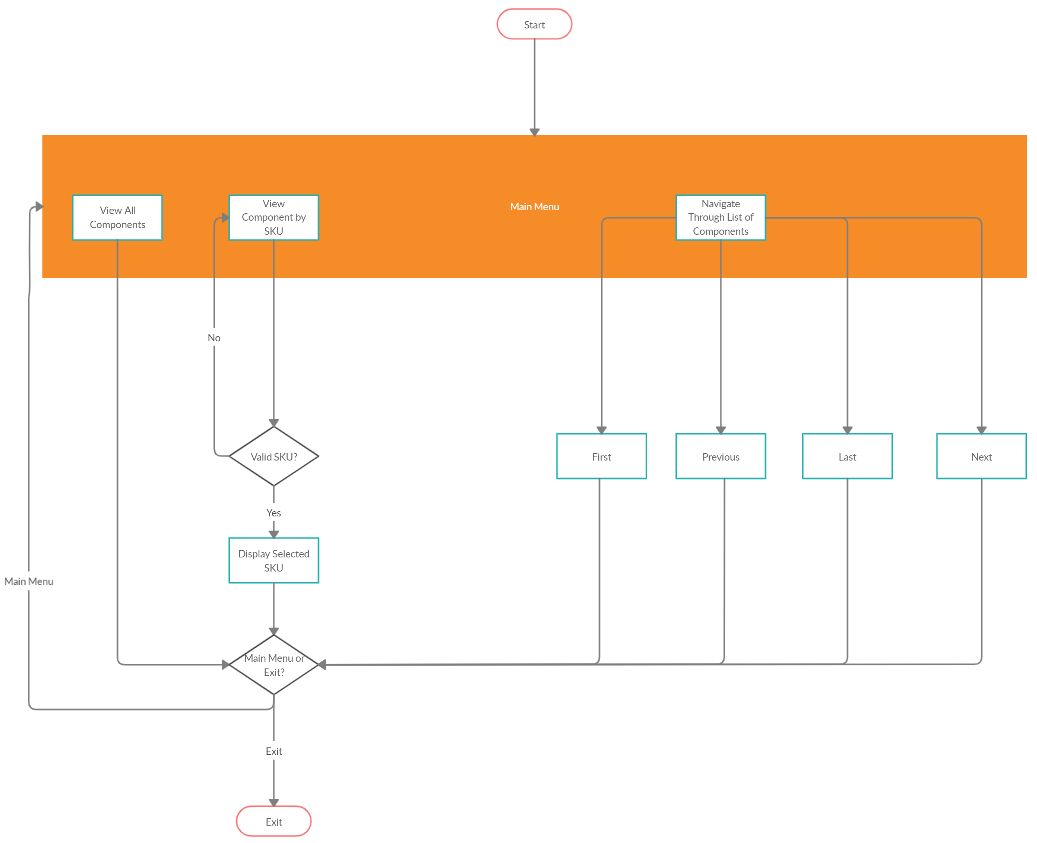
* Keyboards will have a field to indicate if they are QWERTY or not
* Monitors will have a field for screen size
* Games will have a field for their PEGI age rating
* Software will have a field for its software platform
* Applications will file a field for file size (MBs)

An array list of components is to be hardcoded in the stock class.

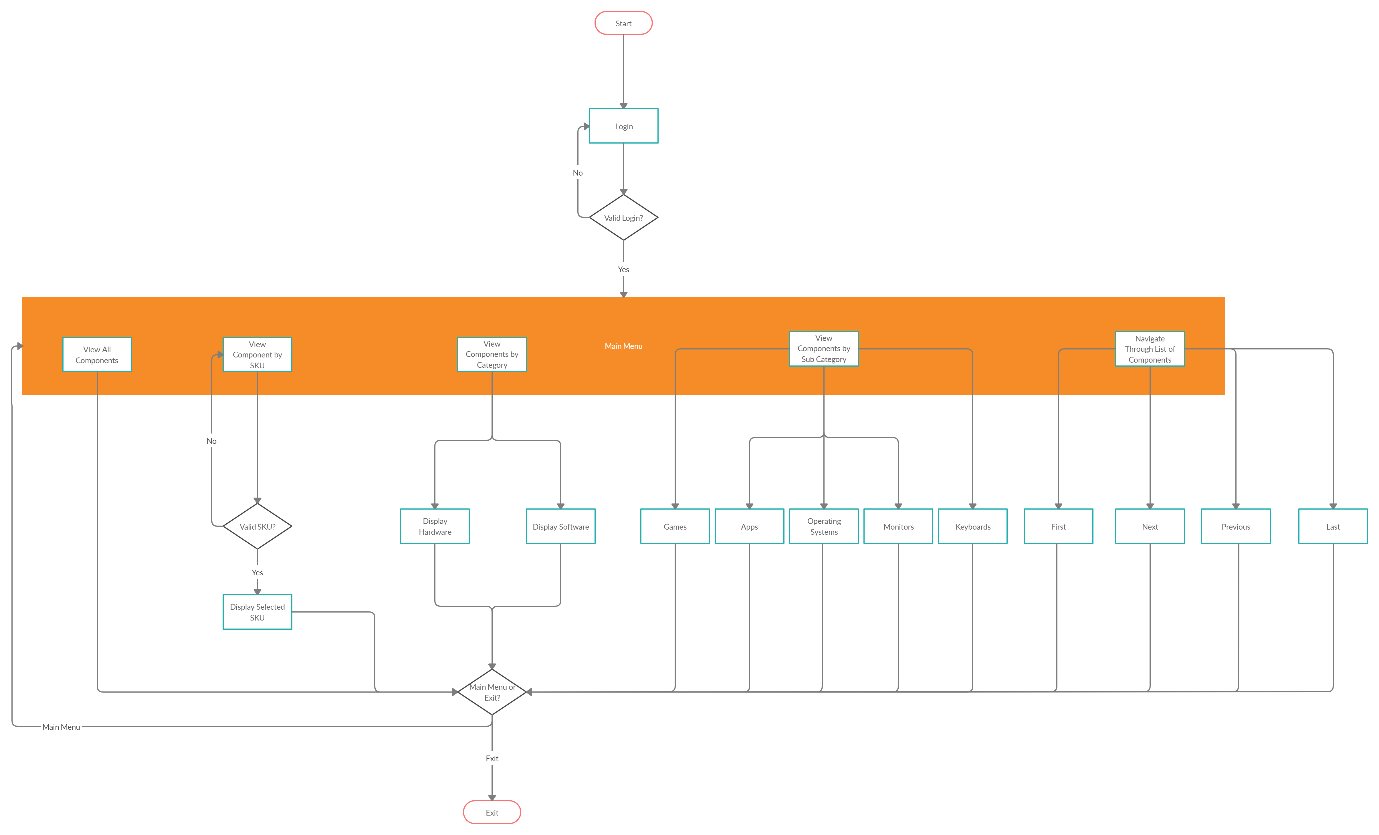
In version 1.1 of the software a list of users will be generated from a user class. The user class will contain a name and password field and will be populated when the program launches.

## 4.8 Control Process

### Component App Version 1.0

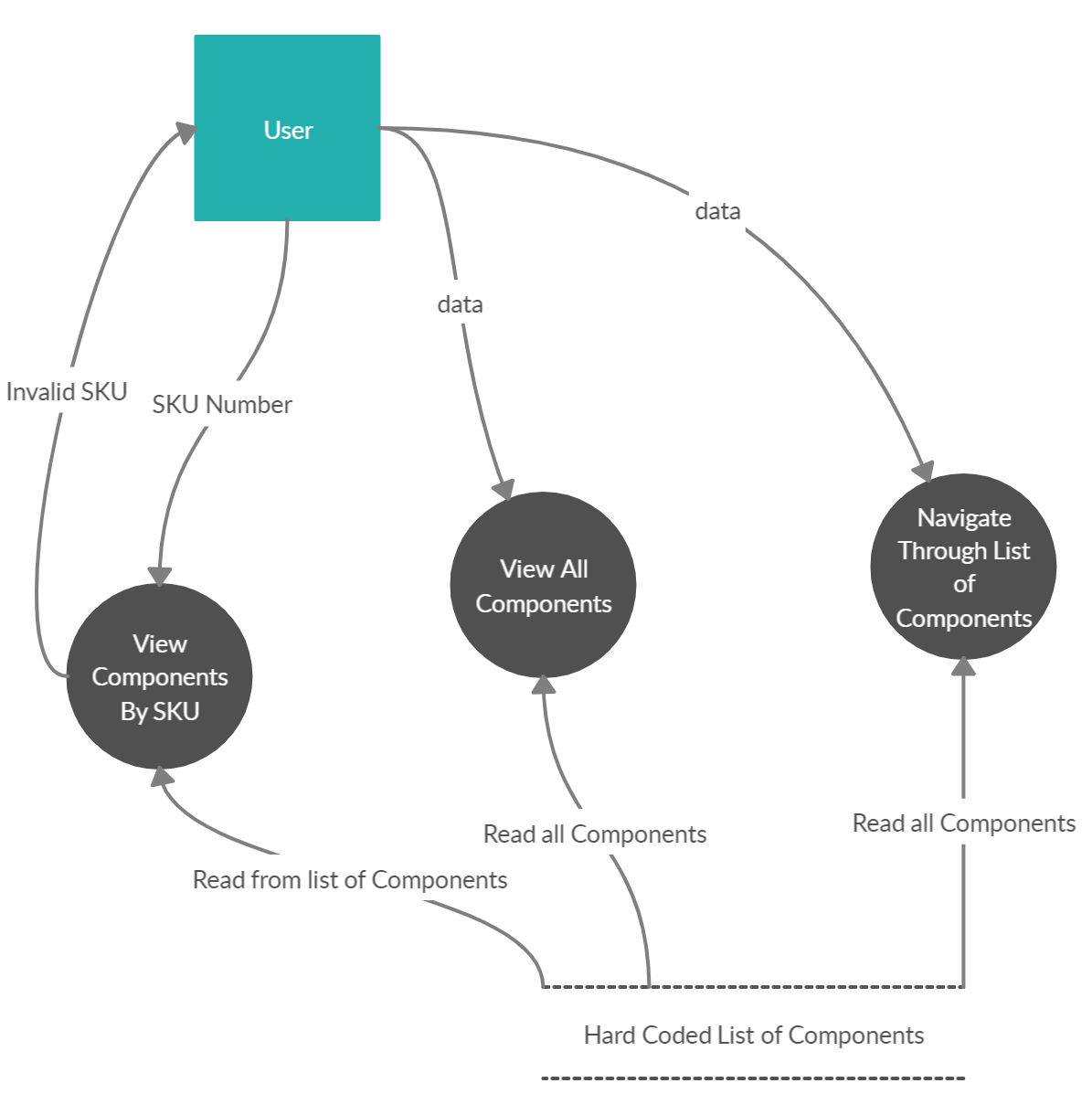


### Component App Version 1.1

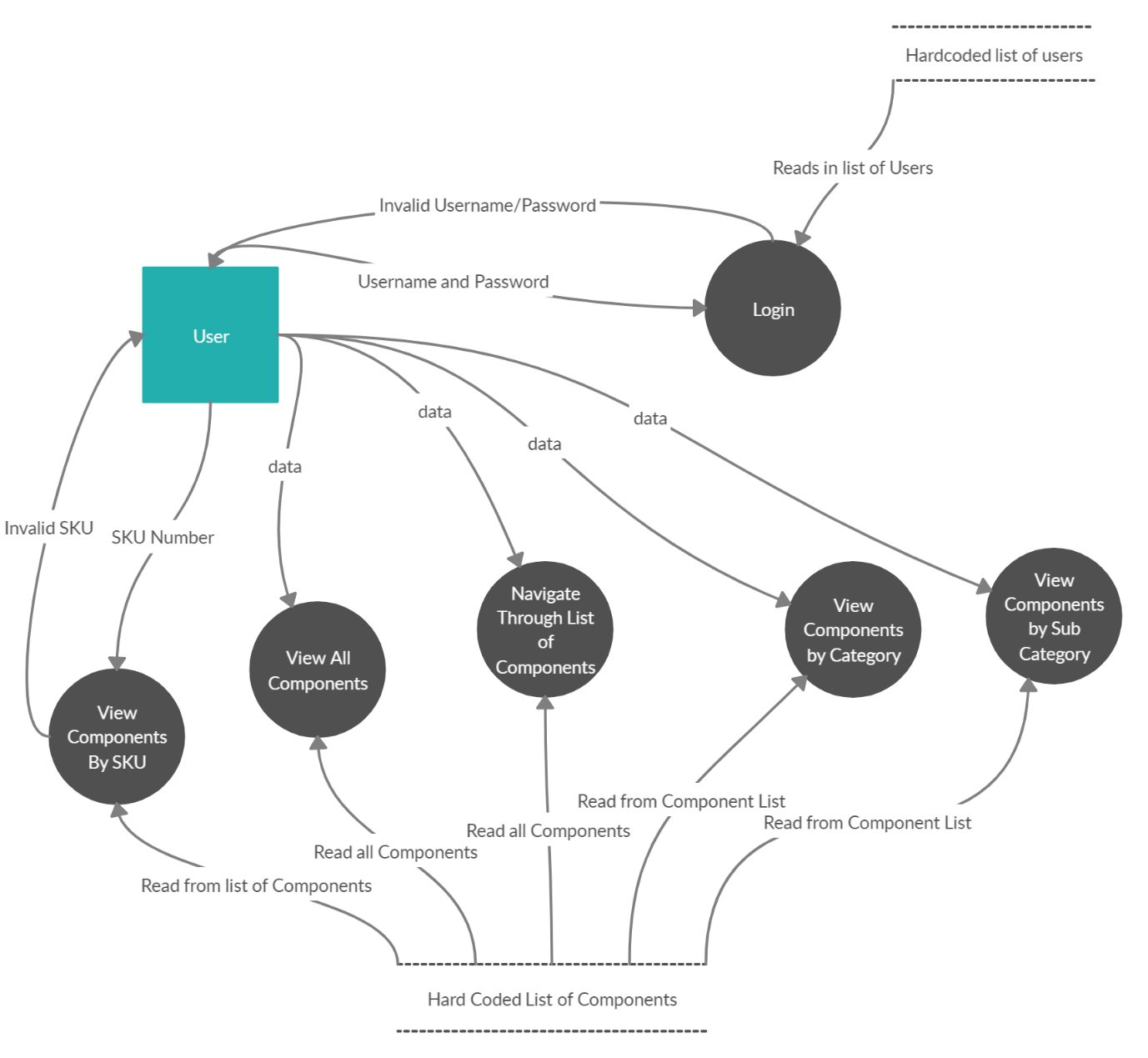


## 4.9 Dataflow

### Component App Version 1.0



### Component App Version 1.1



# 5. Code Elaboration

Below is a code extract for the Navigate method which is contained in the CLI Component App class. This method displays options to the user in order for them to navigate through the list of components, and then prompts them to select an option between 1 and 5. An index variable is used to keep track of what component is currently selected. Selecting option 5 will return to the main menu.

