Module Guide for SmartServe

Team 21, StoneCap Solutions
Max Turek *turekm*
Ryan Were *werer*
Sam Nusselder *nusselds*
Peter Minbashian *minbashp*
David Bednar *bednad*1

April 6, 2023

1 Revision History

Version	Date	Developer(s)	Change(s)
1.0	01/18/23	Max Turek Ryan Were Sam Nusselder Peter Minbashian David Bednar	Initial Draft
1.0	04/05/23	Max Turek Ryan Were Sam Nusselder Peter Minbashian David Bednar	Final Version

2 Reference Material

2.1 Abbreviations and Acronyms

symbol	description
AC	Anticipated Change
DAG	Directed Acyclic Graph
M	Module
MG	Module Guide
OS	Operating System
R	Requirement
SC	Scientific Computing
SRS	Software Requirements Specification
SmartServe	Explanation of program name
UC	Unlikely Change

Contents

List of Tables

List of Figures

3 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 (**?**). We advocate 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 in SC, where modifications are frequent, especially during initial development as the solution space is explored.

Our design follows the rules layed out by **?**, as follows:

- System details that are likely to change independently should be the secrets of separate modules.
- Each data structure is implemented 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 requirements. Section 5 summarizes the module decomposition that was constructed according to the likely changes. Section 6 specifies the connections between the software requirements and the modules. Section 7 gives a detailed description of the modules. Section 8 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 9 describes the use relation between modules.

4 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 4.1, and unlikely changes are listed in Section 4.2.

4.1 Anticipated Changes

AC1: Instead of using an API for login functionality we will use the Raspberry Pi as a host for user login functionality

Rational: This anticipated change is expected for two reasons. Firstly, it becomes easier for the design if all processes are done through the pi. Also, it is cheaper as hosting anything externally costs money.

AC2: Drink volumes may now be calculated via a number of drinks made with certain ingredients to find volumes rather than using sensors

Rational: This anticipated change is expected for to both reduce costs, as less sensor will be required for the project and to make the functionality simpler as it would be easier to design and require less processing power from the pi.

4.2 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: All drink order data will be saved in a database

Rational: Too complex to implement and the manager can supervise

UC2: Personalized drink recommendation feature is implemented

Rational: Too complex to implement and the manager can supervise usage of machine for any safety concerns around under age people using the machine

UC3: Face ID recognition is required for all users before order confirmation

Rational: Too complex to implement and the Pi would not have enough processing power

UC4: Custom drink creation feature is implemented

Rational: People could make drinks which exceed appropriate alcohol limits

UC5: Social media sharing options for all drinks

Rational: Too complex to implement and the Pi would not have enough processing power

UC6: Drink discussion forum feature

Rational: This would require a lot of data storage and far too much processing power for current supplies

UC7: Drink review feature on individual drinks

Rational: This would more data storage resulting in total usage going beyond what is currently available for the entire system

UC8: Sliced lemon or lime add-on option for all orders

Rational: Too complex to integrate into design and presents risk of poor maintenance resulting in expired food

5 Module Hierarchy

This section provides an overview of the module design. Modules are summarized in a hierarchy decomposed by secrets in Table 1. The modules listed below, which are leaves in the hierarchy tree, are the modules that will actually be implemented.

M1: Python Hardware Module

M2: Send Order Module

M3: Volume Tracker Module

M4: Drink Ready

M5: Menu Page Module

M6: Admin Ingredient Input Module

M7: Order History Module

M8: Header Module

M9: Login Page Module

Level 1	Level 2
Hardware-Hiding Module	Python Hardware Module
Behaviour-Hiding Module	Send Order
	Volume Tracker
	Drink Ready
Software Decision Module	Menu Page
	Admin Ingredient Input
	Order History
	Header
	Login Page

Table 1: Module Hierarchy

6 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 2.

7 Module Decomposition

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.

7.1 Python Hardware Module

Secrets: A boolean value corresponding to the creation of a drink

Services: Serves to give insight as to when the drink is complete after, all ingredients have been poured, and the machine is ready to begin the next drink

Implemented By: OS and Web Sockets

7.2 Send Order

Secrets: JSON Data types and NodeJS Server Environment (Hosted on Rasberry Pi)

Services: Module is used to send drink orders made by customers to Smart Serve Hardware to prompt the hardware to begin making said drink. Data consists of GPIO pin

numbers to activate along with a coupling number to correspond with how long the pin should be activated.

Implemented By: OS

7.3 Volume Tracker

Secrets: The contents of the required behaviors.

Services: Module is used to track the amount of liquid of ingredients left within the system. Anytime a drink has been ordered, the system should use the volume tracker to update the amount of volume it has for each drink.

Implemented By : OS

7.4 Drink Ready

Secrets: A boolean value corresponding to the completion of a drink sent from the Hardware

Services: This Module serves as the purpose to notify the user once their drink has been finished. This message will consist of a pop up window for the user on their device when on the web app.

Implemented By : OS

7.5 Login Page Module

Secrets: The login module contains a database of usernames and corresponding passwords. The module also contains functions and variables used to sign in and register users.

Services: Users will be granted access to the header module when the sign-in or register processes are successful.

Implemented By: OS and Database (MySQl)

7.6 Menu Page Module

Secrets: The Menu Page Module consists of a main list containing all available drinks. Each drink contains a name, image, category, and JSON object of ingredients and ingredient sizes.

Services: Users will be able to see the list of all available drinks to select from. Users will be able to send orders to the send order module to begin the process of making their drink.

Implemented By: OS

7.7 Header Page Module

Secrets: The Header Page Module contains the current users' username and a database of administrative users.

Services: Users will be able to navigate to either the menu page, the admin ingredient input, or the order history module

Implemented By: OS

7.8 Admin Ingredient Input Module

Secrets: The Admin Ingredient Input Module contains a list of ingredients and corresponding dispenser locations.

Services: Admin users will be able to input ingredients and corresponding dispenser locations. The menu page drink database is filled with available ingredients from the current module.

Implemented By: OS and Database (MySQl)

7.9 Order History Module

Secrets: The Order History Module contains a list of all drinks ordered since the web application has been running.

Services: Users will be able to get access and view their entire order history through a vizualization methods such as histograms and pie charts.

Implemented By: Database (MySQl)

8 Traceability Matrix

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

Req.	**Modules**
ODR2	M2, M5, M8
ODR5	M3, M5, M6
ODR6	M1, M4
ODR9	M6
ODR10	M6
ODR12	M7
ODR14	M1
ODR15	M9
ODR16	M6, M9
UHR4	M5
MSR4	M6
LR3	M6
LFR1	M6, M7, M8, M9
PR2	M6, M7, M8, M9

Table 2: Trace Between Requirements and Modules

AC	**Modules**
AC1	M2
AC2	M5

Table 3: Trace Between Anticipated Changes and Modules

9 Use Hierarchy Between Modules

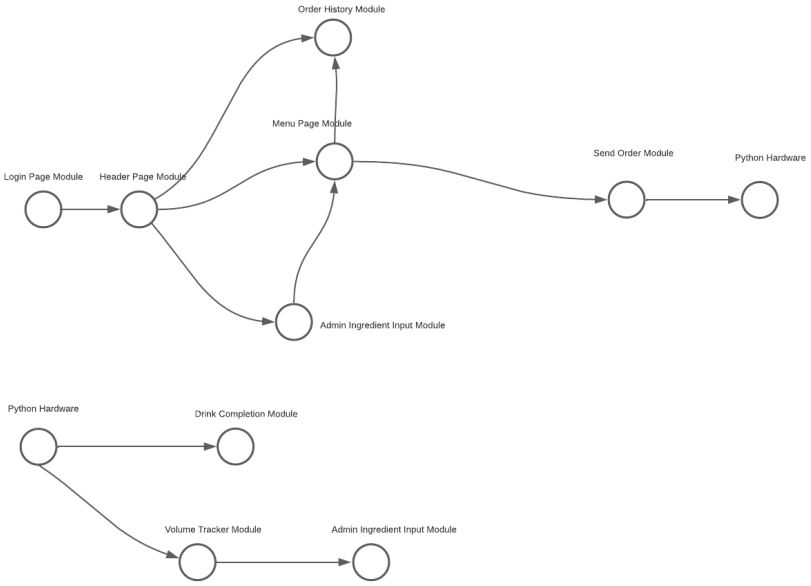

Figure 1: Use hierarchy from the web application to the hardware

Figure 2: Use hierarchy from the hardware to the web application