Module Guide for REVITALIZE

Team 13, REVITALIZE
Bill Nguyen
Syed Bokhari
Hasan Kibria
Youssef Dahab
Logan Brown
Mahmoud Anklis

January 18, 2023

1 Revision History

Date	Version	Notes
January 17th, 2023	Bill Nguyen	Added MG for Main Menu and Calendar
January 18th, 2023	Youssef Dahab	Added Intro, MG for Login, Container, Label, & Circular Slider
January 18th, 2023	Hasan Kibria	Added to SRS Connection, Module Decomposition

2 Reference Material

This section records information for easy reference.

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
REVITALIZE	Explanation of program name
UC	Unlikely Change
[etc. —SS]	[... —SS]

Contents

List of Tables

List of Figures

3 Introduction

REVITALIZE is an app designed to supply users with the means to improve their health by providing them with meal recipe's based on their nutritional preferences, a personalized workouts planner and a sleep tracker.

3.1 Purpose

The purpose of this document is to outline REVITALIZE's modular structure using decomposition based on the principle of information hiding to allow project members to easily identify parts within the app (Parnas, 1972).

3.2 Scope

This document outlines the modules which are based off the requirements specified in the **Software Requirements Specification**. In addition, the external behavior of those modules is specified in the **Module Interface Specification**.

3.3 MG Overiew

- Section 4 lists the anticipated and unlikely changes of the software requirements.
- Section 5 summarizes the module decomposition that was constructed according to the likely changes.
- Section 6 specifies the connection between the software requirements and the modules.
- Section 7 is a 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 hierarchy 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

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 specific hardware on which the software is running.

AC2: The format of the initial input data.

...

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

...

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: Main Menu

M2: Calendar

M3: Login

M4: Container

M5: Label

M6: Circular Slider

M7: Diet Log

M8: Search or Add Food

M9: Custom Meal

M10: Search Recipe

M11: Recipe Results

M12: Recipe Details

M13: FoodT

Level 1	**Level 2**
Hardware-Hiding Module	
Behaviour-Hiding Module	Main Menu
	Calendar
	?
	?
	?
	?
	?
	?
Software Decision Module	?
	?
	?

Table 1: Module Hierarchy

6 Connection Between Requirements and Design

The system design of REVITALIZE is founded upon the requirements previously set in the SRS. It is designed to cater for every single requirement attributed to this project. Hence, the design is decomposed into modular pieces of functionality which each help serve the fulfillment of aforementioned requirements. Specifications on how each module relates to (a) requirement(s) can be found in Section 8.
The functional outlook of each module can be understood by its name and, if needed, its access programs detailed in the MIS. After realizing the functionality of each module, the connections outlined in the Traceability Matrix in Section 8 should be clear and easily comprehended.

7 Module Decomposition

Modules are decomposed according to the principle of "information hiding" proposed by Parnas et al. (1984). 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. *REVITALIZE* means the module will be implemented by the REVITALIZE 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.

7.1 Hardware Hiding Modules

N/A

7.2 Behaviour-Hiding Module

7.2.1 Main Menu (M1)

Secrets: Method to visualize main menu

Services: Visualizes main menu by displaying interactive buttons to navigate to Diet, Exercise and/or Sleep section. Also shows current date in the top-center of screen which is clickable to display calendar screen. Finally a backward and forward button to display previous and next day.

Implemented By: REVITALIZE

7.2.2 Calendar (M2)

Secrets: Method to visualize calendar screen

Services: Visualizes calendar by displaying interactive screen that shows current month and the respective days of the month, where user can click on desired day. Also has a backward and forward button to display previous and next month.

Implemented By: REVITALIZE

7.2.3 Login (M3)

Secrets: Method to visualize login screen

Services: Visualizes login screen by displaying interactive screen that shows name or email and password input text boxes for user to enter. Also has a login button for user to

login. Moreover, has forgot password and sign up links for user to reset password or sign up respectively.

Implemented By: REVITALIZE

7.2.4 Container (M4)

Secrets: Method to visualize sleep screen

Services: Visualizes sleep screen by displaying interactive screen that shows labels and circular slider in sleep screen.

Implemented By: REVITALIZE

7.2.5 Label (M5)

Secrets: Method to visualize certain components in sleep screen

Services: Visualizes bed icon, "BEDTIME" text, ring icon, "WAKE UP" text, user set bed-time and wake-up time in sleep screen.

Implemented By: REVITALIZE

7.2.6 Circular Slider (M6)

Secrets: Method to visualize circular slider and total sleep time in sleep screen

Services: Visualizes circular slider by displaying interactive screen that shows circular slider that user can slide to set bed-time and wake-up times. Also shows user their total sleep time.

Implemented By: REVITALIZE

7.2.7 Diet Log Module (M7)

Secrets: Method to visualize daily diet log screen

Services: Visualizes diet log by displaying interactive table with adherent food entries. Also shows net nutritional intake for the day.

Implemented By: REVITALIZE

7.2.8 Search or Add Food Module (M8)

Secrets: Method to visualize decision making screen between adding a custom food adn searching an online recipe

Services: Visualizes decision making screen by displaying two buttons, each representative of the possible decision made, which leads to another screen to help user complete their preferred decision.

Implemented By: REVITALIZE

7.2.9 Custom Meal Module (M9)

Secrets: Method to visualize custom meal logger

Services: Visualizes custom meal logger by displaying input fields which take information relative to custom meal.

Implemented By: REVITALIZE

7.2.10 Search Recipe Module (M10)

Secrets: Method to visualize app-internal search engine for recipes with custom filterization

Services: Visualizes app-internal search engine for recipes by displaying multiple filtering inputs and a search button which retreives filtered recipe data.

Implemented By: REVITALIZE

7.2.11 Recipe Results Module (M11)

Secrets: Method to visualize search results from search query produced through Search Recipe Module

Services: Visualizes search results by displaying a list of scrollable search results with a small description for each result.

Implemented By: REVITALIZE

7.2.12 Recipe Details Module (M12)

Secrets: Method to visualize detailed information about each recipe

Services: Visualizes detailed information about each recipe using a paragraph description and an image

Implemented By: REVITALIZE

7.3 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: –

7.3.1 FoodT Module (M13)

Secrets: Data structure to hold information about each distinct food

Services:

Implemented By: REVITALIZE

7.3.2 Etc.

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
R1	M??, M??, M??, M??
R2	M??, M??
R3	M??
R4	M??, M??
R5	M??, M??, M??, M??, M??, M??
R6	M??, M??, M??, M??, M??, M??
R7	M??, M??, M??, M??, M??
R8	M??, M??, M??, M??, M??
R9	M??
R10	M??, M??, M??
R11	M??, M??, M??, M??

Table 2: Trace Between Requirements and Modules

AC	Modules
AC1	M??
AC2	M??
AC??	M??
AC??	M??
AC??	M??
AC??	M??
AC??	M??
AC??	M??
AC??	M??
AC??	M??
AC??	M??
AC??	M??

Table 3: Trace Between Anticipated Changes and Modules

9 Use Hierarchy Between Modules

In this section, the uses hierarchy between modules is provided. Parnas (1978) 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

References

David L. Parnas. On the criteria to be used in decomposing systems into modules. *Comm. ACM*, 15(2):1053–1058, December 1972.

David L. Parnas. Designing software for ease of extension and contraction. In *ICSE '78: Proceedings of the 3rd international conference on Software engineering*, pages 264–277, Piscataway, NJ, USA, 1978. IEEE Press. ISBN none.

D.L. Parnas, P.C. Clement, and D. M. Weiss. The modular structure of complex systems. In *International Conference on Software Engineering*, pages 408–419, 1984.