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 Introduction section

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

1 Revision History **i**

2 Reference Material **ii**
2.1 Abbreviations and Acronyms . . . ii

3 Introduction **1**
3.1 Purpose . . . 1
3.2 Scope . . . 1
3.3 MG Overiew . . . 1

4 Anticipated and Unlikely Changes **1**
4.1 Anticipated Changes . . . 2
4.2 Unlikely Changes . . . 2

5 Module Hierarchy **2**

6 Connection Between Requirements and Design **2**

7 Module Decomposition **3**
7.1 Hardware Hiding Modules . . . 3
7.2 Behaviour-Hiding Module . . . 3
7.2.1 Main Menu (M1) . . . 3
7.2.2 Calendar (M2) . . . 4
7.3 Software Decision Module . . . 4
7.3.1 Etc. . . . 4

8 Traceability Matrix **4**

9 Use Hierarchy Between Modules **5**

List of Tables

1 Module Hierarchy . . . 3
2 Trace Between Requirements and Modules . . . 5
3 Trace Between Anticipated Changes and Modules . . . 5

List of Figures

1 Use hierarchy among modules . . . 6

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

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.

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

Table 1: Module Hierarchy

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 dispaying 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

7.3 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 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.