SE 3XA3: Module Guide
Wordle 2.0

Team 8,
Richard Fan, fanr13
Noel Zacharia, zacharin
Biranugan Pirabaharan, pirabahb

March 18, 2022

Contents

List of Tables

List of Figures

Table 1: **Revision History**

Date	**Version**	**Notes**
March 12, 2022	0.1	Initial document
March 18, 2022	0.2	Finished MG

1 Introduction

1.1 Overview

Wordle 2.0 is a re-implementation of the game Not Wordle. The original game made players guess a 5 letter word in 6 tries. The purpose of this project is to recreate the functionality of the original game while also adding additional features to create extra value. Additionally, we have followed the software development process to ensure that our system is well built and understood by others.

1.2 Purpose

The Module Guide serves as a guide to the modular structure of our system. The system is decomposed into several modules based on the design principle of information hiding (Parnas, 1972). This method allows maintainers, new project members, and designers to quickly understand different parts of the system, allowing for high maintainability and readability. Additionally, this design principle also supports the idea of design for change and information hiding since likely changes can be hidden as "secrets" within individual modules.

1.3 Scope

This Module Guide describes the modules that are based upon the requirements found within the Software Requirements Specification. External behaviors of these modules will be described in the Module Interface Specification (MIS) document. Likely and unlikely changes are documented. A use hierarchy is provided to visualize use relations between modules specified in this document.

1.4 Outline

The rest of the document is structured 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 7 describes the use relation between modules.

2 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 2.1, and unlikely changes are listed in Section 2.2.

2.1 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 graphical elements used in the game will be improved and revised over time. The web page will feature a more polished appearance for the final deliverable.

AC2: The theme options the player has to choose from will have options and be refined. The current dark mode will be completely reworked as its current implementation was just a placeholder.

AC3: The algorithm used to find the accuracy of the guesses. Currently, any occurrences of letters that are present in the target word are shown as present even when the target word has fewer occurrences than the guess itself.

AC4: The appearance of the game's alert system for invalid guesses and guesses that are too short. Currently uses the browser's default alert.

AC5: Additional functionality that has yet to be added: various word lengths support.

2.2 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 methods of the user (Input: mouse and/or keyboard).

UC2 The word lists used for determining the target word and accepting the guesses. These are static lists with over 2000 possible words.

UC3 The keyboard and its layout. The game will not support unconventional layouts. It will stick to QWERTY. Additionally, only modern English letters will be supported, with no accents or other language options.

UC4 The core rules of the game.

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: Valid Guess

M2: Word Operations

M3: Create Gameboard

M4: Dark Mode

M5: Main

M6: Keyboard Colours

M7: Tile Colours

M8: Keyboard

M9: Stats

M10: Instructions

M11: Stats View

M12: Constants

M13: Update Gameboard

M14: Style

M15: Alert

M16: Click

M17: Index

M18: Share

4 Connection Between Requirements and Design

The system encompassing Wordle 2.0 is designed to satisfy all of the functional and non-functional requirements that were laid out in the SRS. The system is decomposed into modules and the connections between the requirements and modules are shown via traceability matrices in Table 3.

Modules are decomposed such that every requirement we outlined can be traced to at least one module. The module name is indicative of the module functioning.

Level 1	**Level 2**
Hardware-Hiding Module	
Behaviour-Hiding Module	Stats View Keyboard Dark Mode Instructions Keyboard Colours Tile Colours Create Gameboard Style Alert Click Index Share Update Gameboard
Software Decision Module	Word Operations Stats Valid Guess Main Constants

Table 2: Module Hierarchy

5 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. Also indicate if the module will be implemented specifically for the 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. Whether or not this module is implemented depends on the programming language selected.

5.1 Hardware Hiding Modules

N/A

5.2 Behaviour Hiding Modules

5.2.1 Create Gameboard (M3)

Secrets: The method of drawing the gameboard to the screen.

Services: Draws the gameboard to the screen.

Implemented By: Wordle 2.0

5.2.2 Dark Mode (M4)

Secrets: The method of changing the background to dark mode.

Services: Changes the the background to dark mode.

Implemented By: Wordle 2.0

5.2.3 Keyboard Colours (M6)

Secrets: The method for changing key colours to green, yellow, and grey.

Services: Changes key colours.

Implemented By: Wordle 2.0

5.2.4 Tile Colours (M7)

Secrets: The method for changing gameboard tile colours to green, yellow, and grey.

Services: Changes tile colours within the gameboard.

Implemented By: Wordle 2.0

5.2.5 Keyboard (M8)

Secrets: The method for drawing the keyboard to the screen.

Services: Draws the keyboard to the screen.

Implemented By: Wordle 2.0

5.2.6 Instructions (M10)

Secrets: The method to visualize the instructions screen.

Services: Draws the instructions menu with a GUI.

Implemented By: Wordle 2.0

5.2.7 Stats View (M11)

Secrets: The method to visualize the statistics screen.

Services: Draws the statistics menu with a GUI.

Implemented By: Wordle 2.0

5.2.8 Update Gameboard (M13)

Secrets: The method to update the gameboard with the provided letter.

Services: Updates the gameboard with the provided letter

Implemented By: Wordle 2.0

5.2.9 Style (M14)

Secrets: The styling of elements within the game.

Services: Styles elements within the game.

Implemented By: Wordle 2.0

5.2.10 Alert (M15)

Secrets: The method to visualize alerts.

Services: Draws alerts to the screen with a GUI.

Implemented By: Wordle 2.0

5.2.11 Click (M16)

Secrets: Actions that will be done when game keys are pressed.

Services: When keys are pressed, the corresponding letter is displayed on the screen.

Implemented By: Wordle 2.0

5.2.12 Index (M17)

Secrets: The content and structure of the game.

Services: Structures the content of the game.

Implemented By: Wordle 2.0

5.2.13 Share (M18)

Secrets: The method to share the user's game results.

Services: Copies final gameboard's colours as emojis to the user's clipboard.

Implemented By: Not yet.

5.3 Software Decision Module

5.3.1 Valid Guess (M1)

Secrets: The algorithm for checking if the given word is a valid word.

Services: Provides a method to check if the given word is valid.

Implemented By: Wordle 2.0

5.3.2 Word Operations (M2)

Secrets: The algorithm for finding a random word.

Services: Provides methods to generate a random word and check if it matches the user guess.

Implemented By: Wordle 2.0

5.3.3 Main (M5)

Secrets: Calls to other modules.

Services: Initializes modules and controllers.

Implemented By: Wordle 2.0

5.3.4 Stats (M9)

Secrets: The algorithms required to calculate player statistics.

Services: Provides methods to calculate player statistics.

Implemented By: Wordle 2.0

5.3.5 Constants (M12)

Secrets: The data structures which store the keyboard keys, gameboard and word lists.

Services: None

Implemented By: Wordle 2.0

6 Traceability Matrix

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

Req.	**Modules**
FR1	M8, M12, M14
FR2	M3, M12, M14
FR3	M4
FR4	M10
FR5	M9, M11
FR6	M9, M11
FR7	M9, M11
FR8	M9, M11
FR9	M9, M11
FR10	M9, M11
FR11	M16, M18
FR12	M8, M1, M5, M16
FR13	M13, M7, M6, M2
FR14	M13, M7, M6
FR15	M13, M7, M6
FR16	M1, M15
FR17	M1, M15
FR18	M13, M6, M17
FR19	M5
FR20	M5
FR21	M5
FR22	M5, M15, M2

Table 3: Trace Between Requirements and Modules

AC	**Modules**
AC1	M14
AC2	M4
AC3	M13
AC4	M14
AC5	M3

Table 4: Trace Between Anticipated Changes and Modules

7 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.

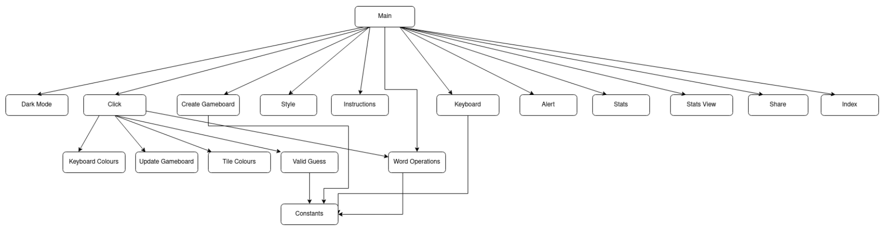

Figure 1: Use hierarchy among modules

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.