# SE 3XA3: Software Requirements Specification
# Title of Project

Team #24, Node Messenger
Tasin Ahmed - ahmedm31
Shardool Patel - pates25
Omar Elemary - elemaryo

November 9, 2018

# Contents



# List of Tables



# List of Figures

# 1 Major Revision History

| Revision | Date | Change |
|---|---|---|
| Revision 0 | Date 1 | Notes |
| Revision 1 | Date 2 | Notes |

Table 1: My caption

# 2 Introduction

## 2.1 Overview

As the world becomes increasingly more connected through the internet, many common internet users are looking for an easy way to communicate with each other or reach out to distant loved ones. The market for messengers has become saturated with products that put the goal of earning maximum revenue over the needs of the consumer. Node Messenger's focus is to capture consumer interest by implementing a free and accessible web application messenger that allows them to personalize their experience and chat with other users in a simple, clean and non-intrusive way. Node messenger will become a haven for users searching for a consumer-friendly product with great functionality and cross-platform support.

## 2.2 Context

This document is the Module Guide (MG), created after the completion of the Software Requirements Specifications (SRS. The SRS lists the functional and non-functional requirements for the development of the project. The MG shows how the project meets the functional and non-functional requirements mentioned in the SRS, as well as showing how the modules are broken up in the project.

Following the creation of the MG, comes the creation of the Module Interface Specification (MIS). The MIS specifies the functions of each of the modules mentioned in the MG. It does so by documenting the variables, inputs, outputs, and exceptions for each module.

## 2.3 Design Principles

## 2.4 Document Structure

The rest of the document is organized as follows:

- Section 3 lists the anticipated and unlikely changes of the software requirements.
- Section 4 summarizes the module decomposition that was constructed according to the likely changes.

- Section 5 specifies the connections between the software requirements and the modules.
- Section 6 gives a detailed description of the modules.
- Section 7 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 8 describes the use relation between modules.

# 3 Anticipated and Unlikely Changes

## 3.1 Anticipated Changes

**AC1:** The specific hardware on which the software is running.

**AC2:** The format of the HTML, CSS, and Javascript files.

**AC3:** The color theme of the messaging app.

**AC4:** The format of the message box.

**AC5:** The format of the Login/Register screen.

**AC6:** Domain name.

**AC7:** Addition or removal of Buttons, and their functions.

**AC8:** The Graphical User Interference for sending and receiving messages.

**AC9:** The settings available to every user.

## 3.2 Unlikely Changes

**UC1:** Input/Output devices (Input: File and/or Keyboard, Output: File, Memory, and/or Screen).

**UC2:** There will always be a source of input data external to the software.

# 4 Module Hierarchy

This section provides an overview of the module design. Modules are summarized in a hierarchy decomposed by secrets in Table 3. The modules listed below in Table 2, which are leaves in the hierarchy tree, are the modules that will actually be implemented.

| Module Name | Module Number |
|---|---|
| Hardware Hiding Module | M1 |
| Sign up Module | M2 |
| Login Module | M3 |
| Sign in Module | M4 |
| Chat Module | M5 |
| App Module | M6 |
| Index Module | M7 |

Table 2: Module Number Format

| **Level 1** | **Level 2** |
|---|---|
| Hardware-Hiding Module | |
| Behaviour-Hiding Module | Index Module<br>App Module<br>Login Module |
| Software Decision Module | Sign up Module<br>Sign in Module<br>Chat Module |

Table 3: Module Hierarchy

# 5 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 4.

# 6 Module Decomposition

Modules are decomposed according to the principle of "information hiding" proposed by **?**. 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.

## 6.1 Hardware Hiding Modules (M??)

**Secrets:** The data structure and algorithm used to implement the virtual hardware.

**Services:** Serves as a virtual hardware used by the rest of the system. This module provides the interface between the hardware and the software. So, the system can use it to display outputs or to accept inputs.

**Implemented By:** OS

## 6.2 Behaviour-Hiding Module

**Secrets:** The contents of the required behaviours.

**Services:** Includes programs that provide externally visible behaviour of the system as specified in the software requirements specification (SRS) documents. This module serves as a communication layer between the hardware-hiding module and the software decision module. The programs in this module will need to change if there are changes in the SRS.

**Implemented By:** –

### 6.2.1 Input Format Module (M??)

**Secrets:** The format and structure of the input data.

**Services:** Converts the input data into the data structure used by the input parameters module.

**Implemented By:** [Your Program Name Here]

### 6.2.2 Etc.

## 6.3 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:** –

### 6.3.1 Etc.

# 7 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 4: Trace Between Requirements and Modules

| AC | Modules |
|---|---|
| AC1 | M?? |
| AC9 | M?? |
| AC?? | M?? |
| AC?? | M?? |
| AC?? | M?? |
| AC?? | M?? |
| AC?? | M?? |
| AC?? | M?? |
| AC?? | M?? |
| AC?? | M?? |
| AC?? | M?? |
| AC?? | M?? |

Table 5: Trace Between Anticipated Changes and Modules

# 8 Use Hierarchy Between Modules

In this section, the uses hierarchy between modules is provided. **?** 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