SE 3XA3: Software Requirements Specification
QChat

Team #14, QChat
Adit Patel - patela14
Harsh Patel - patelh11
Vrushesh Patel - patelv12

November 10, 2017

Contents

List of Tables

List of Figures

1 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 used 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.

Table 1: **Revision History**

Date	**Version**	**Notes**
10/11/2017	1.0	Initial Version
25/11/2017	1.1	Updated Module Decomposition - Software Decision Module

The rest of the document is organized 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.

1.1 Introduction - QChat

This document indicates the Module Guide for the implementation of the QChat anonymous questions and answers web app. This document is intended to facilitate the design and maintenance of the project while giving and through and clear specifications of the interfacing that occurs between the different modules in the project.

One common approach to developing software that is commonly accepted is decomposing the system into modules and then interconnecting them in way that allows for the system to be easily maintained and updated. Any work assignment for a programmer or a programming team is considered to be a module. This module guide will allows for designers and maintainers to easily identify different parts of the software.

Our system design will follow the following rules:

- Each module will be designed to handle a specific task
- System details that are handled independently should be secrets of separate modules, this is where information hiding will be used.
- Other programs/modules that require information stored in a certain modules data must obtain it by calling access programs that belong to that module.

The potential readers of this document are one or more of the following:

- **New Project Members:** Any new member to the team will be expected to read this document to understand the structure of the project and how the different modules interact to make the system work. Reading this document will also answer most trivial questions a new member will have regarding the project.
- **Project Maintainers:** The software maintenance team will be expected to read this document as it will give them a good understanding of the hierarchical structure of the modules and how they are prioritized and interconnected.
- **Project Designers:** Designers can use this document to check for consistency, feasibility and flexibility of the system. Designers can use many techniques to verify the system however they must all match the expected results noted and implied in this document.

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 hardware on which the application is going to run.

AC2: The database used by the application.

AC3: The format of input data (eg. SessionID could be changed to take only numbers).

AC4: The user interface of the application.

AC5: Adding feature to trace the user from their posts while keeping it anonymous (can only be used by authorized person).

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: The format of the data stored (eg. JSON).

UC2: The goal of the application: To allow users to ask questions and reply answers anonymously.

UC3: The web technologies used to make QChat.

UC4: The structure of the data stored.

UC5: Firebase configuration data (eg. API key, database URL).

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: Hardware-Hiding Module

M2: Data Manipulation Module

M3: Initializer/Firebase Module

M4: UI Module

M5: Client/Server Communication Module

M6: Input verification Module

Level 1	**Level 2** (Model)	**Level 3** (View/Controller)
Hardware-Hiding Module		
Behaviour-Hiding Module	Data Manipulation Module	UI Module Client/Server Communication Module
Software Decision Module	Initializer/Firebase Module	Input verification Module

Table 2: Module Hierarchy

4 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 3.

5 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.

5.1 Hardware Hiding Modules (M1)

Secrets: Although there isn't any physical hardware involved for our project, the data structures and algorithms used to implement the virtual hardware are the secrets of this module.

Services: Serves as virtual hardware used by the rest of the system, this module provides the interface between the hardware and the software which will allow the system to use it to display outputs and accept inputs.

Implemented By: OS

5.2 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: –

5.2.1 Data Manipulation Module (M2)

Secrets: How to manipulate/store/access/update data and where it is placed and accessed.

Services: Updates database, fetches data and returns it to the accessing module.

Implemented By: -

5.2.2 User Interface Module (M4)

Secrets: UI features implementation, how it's updated dynamically.

Services: Updated UI as user interacts with the different fields available, shows data.

Implemented By: Html files, CS files and JavaScript files

5.2.3 Client/Server Communication Module (M5)

Secrets: How and where communication occurs

Services: Transferring data from other module to database and accessing data from database to use in other modules

Implemented By: Index.js file, Firebase

5.3 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: –

5.3.1 Client/Server Communication Module (M6)

Secrets: How inputted data is verified is hidden from other modules.

Services: Handles user inputs, checks for correct input semantics before packaging and transferring inputted data to other modules

Implemented By: Index.js file, Html files, CS files and JavaScript files, Firebase

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
Functional - 1	M1, M2, M3, M4, M5, M6
Functional - 2	M1, M2, M3, M4, M5, M6
Functional - 3	M4, M5, M6
Functional - 4	M2, M4, M6
Non-Functional-3.1	M4
Non-Functional-3.2	M4, M6
Non-Functional-3.3	M2, M3, M4, M5, M6
Non-Functional-3.4	M3, M5, M6
Non-Functional-3.5	M2, M3, M5
Non-Functional-3.6	M2, M3, M5
Non-Functional-3.7	M2, M4, M6

Table 3: Trace Between Requirements and Modules

AC	Modules
AC1	M1
AC2	M3, M5
AC3	M6
AC4	M4
AC5	M2, M3, M5, M6

Table 4: Trace Between Anticipated Changes and Modules

7 Use Hierarchy Between Modules

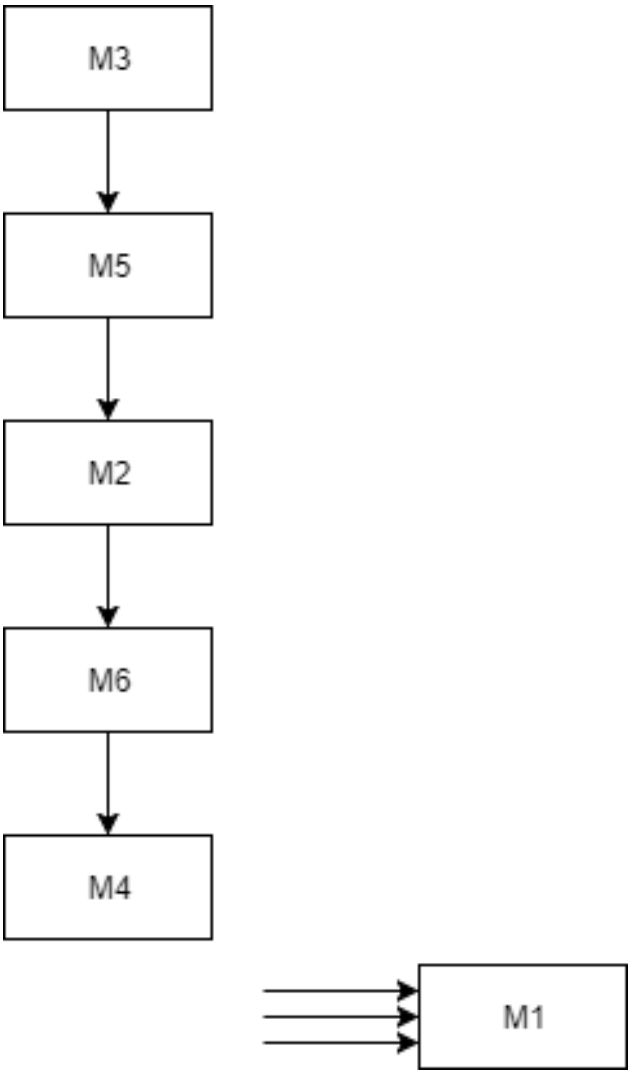

Figure 1: Use hierarchy among modules