**CS340 Software Engineering**

**Software Requirements Specification (SRS) Template**

Items that are intended to stay in as part of your document are in **bold**; explanatory comments are in *italic* text. Plain text is used where you might insert wording about your project.

The document in this file is an annotated outline for specifying software requirements, adapted from the IEEE Guide to Software Requirements Specifications (Std 830-1993).

Tailor this to your needs, removing explanatory comments as you go along. Where you decide to omit a section, keep the header, but insert a comment saying why you omit the data.

**Chatbot and Scheduler**

Team 3:

Dylan Wahlstrom, Devin Roering, Yang Toua, Bikash Timalsina, Chandler

**Software Requirements Specification**

**Document**

**Version: (1)** **Date: (10/19/2019)**

**Table of Contents**

**1. Introduction**

*1.1 Purpose*

*1.2 Scope*

*1.3 Definitions, Acronyms, and Abbreviations*

*1.4 References*

*1.5 Overview*

**2. The Overall Description**

*2.1 Product Perspective*

2.1.1 System Interfaces

2.1.2 Interfaces

2.1.3 Hardware Interfaces

2.1.4 Software Interfaces

2.1.5 Communications Interfaces

2.1.6 Memory Constraints

2.1.7 Operations

2.1.8 Site Adaptation Requirements

*2.2 Product Functions*

*2.3 User Characteristics*

*2.4 Constraints*

*2.5 Assumptions and Dependencies*

*2.6 Apportioning of Requirements*

**3. Specific Requirements**

*3.1 External interfaces*

*3.2 Functions*

*3.3 Performance Requirements*

*3.4 Logical Database Requirements*

*3.5 Design Constraints*

3.5.1 Standards Compliance

*3.6 Software System Attributes*

3.6.1 Reliability

3.6.2 Availability

3.6.3 Security

3.6.4 Maintainability

3.6.5 Portability

*3.7 Organizing the Specific Requirements*

3.7.1 System Mode

3.7.2 User Class

3.7.3 Objects

3.7.4 Feature

3.7.5 Stimulus

3.7.6 Response

3.7.7 Functional Hierarchy

*3.8 Additional Comments*

**4. Change Management Process**

**5. Document Approvals**

**6. Supporting Information**

# 1. Introduction

## 1.1 Purpose

Our goal is to create a Chatbot that utilizes cloud infrastructure and visualization services to answer FAQ, schedule appointments with physicians, and pay outstanding bills. A chatbot is an automated program that answers questions and completes similar tasks through a text box or through links in order to help move through a website and find the information you need with the help of artificial intelligence. This would be a good way for clients to get their information while keeping the physicians free. For our project, we will be focusing on setting up the project to work for clinics. This document shall give current and future designers the information they require to build, manipulate, and maintain the system as intended.

## 1.2 Scope

The scope of this product entails the utilization and integration of an AI chatbot, a visualization of a human that physically and verbally responds to input based on the AI’s response to a question, and the necessary APIs to implement a system that can schedule appointments, pay and view current bills, and answer frequently asked questions.

The goal of this process is to free up staff and allow them to more effectively do their job, while the Chatbot handles support, scheduling, and billing in a streamlined and professional way. The AI must be professional, meaning the dictionary used will omit certain forms of language that could be perceived as rude or abrupt. If patients feel uncomfortable with the system, they will not be inclined to use the service again.

This product shall not diagnose, handle the distribution of confidential information, or control prescriptions. The only user information that should be accessible to the program is the users account information on file, and any medical flags or alerts, such as heart conditions or diabetes information. This is only to assist the physicians that the program puts clients in touch with at diagnosing by allowing them to view all necessary information easily. A Link to this information would be sent to physicians with appointment times as they are scheduled to allow for prep time to be completed more efficiently. This will speed up wait times and improve the speed in which physicians can assist the public.

## 1.3 Definitions, Acronyms, and Abbreviations.

Chatbot- a program that receives user input and displays a response based on that input.

AI- Artificial intelligence, a way to create simulated intelligence in a computer program, making the

responses it gives relevant to the questions asked in the chatbot.

Visual Studio- A Microsoft program that allows code to be edited and assembled, as well as compiled.

Unity- Visualization Software used in the creation of animation. It is easily integrated into Visual Studio.

Github- An online service that will allow us to publish projects that can then be downloaded by users.

## 1.4 References

(1) Chatbot and Scheduler SDD Document (can be obtained in trello)

*In this subsection:*

*(1) Provide a complete list of all documents referenced elsewhere in the SRS*

*(2) Identify each document by title, report number (if applicable), date, and publishing organization*

1. *Specify the sources from which the references can be obtained.*

*This information can be provided by reference to an appendix or to another document. If your application uses specific protocols or RFC’s, then reference them here so designers know where to find them.*

## 1.5 Overview

This document contains descriptions on how the system is supposed to be set up. It is intended for users who wish to learn more about the system, and for developers designing the system. Section 1 contains the overall top-level description of the system and what it can and cannot do, as well as any references and definitions that the readers should be informed of. Section 2 contains the overall descriptions of the system. This includes the Perspective, System Interfaces, UI design elements, Memory Constraints, Hardware and Software Interfaces, Product Functions, User Scenarios, and Assumptions and Dependencies. Section 3 contains the specific requirements. This includes the functional and nonfunctional requirements, as well as functional hierarchy, user classes, objects, stimulus, and responses. The rest of the document includes the supporting information required to implement the other information held in the document.

# 2. The Overall Description

The overall description of this chatbot is so that clinics time can be streamlined so that doctors and staff can focus more on taking care for the patients. In order to do this, the chatbot must be able to schedule appointments, give back answers for customers(FAQs), and be able to do basic billing so patients do not have to waste more time going to the clinic but rather pay bills on the fly.

# 

*Describe the general factors that affect the product and its requirements. This section does not state specific requirements. Instead, it provides a background for those requirements, which are defined in section 3, and makes them easier to understand*. *In a sense, this section tells the requirements in plain English for the consumption of the customer. Section3 will contain a specification written for the developers.*

## 2.1 Product Perspective

*Put the product into perspective with other related products. If the product is independent and totally self-contained, it should be so stated here. If the SRS defines a product that is a component of a larger system, as frequently occurs, then this subsection relates the requirements of the larger system to functionality of the software and identifies interfaces between that system and the software. If you are building a real system,compare its similarity and differences to other systems in the marketplace. If you are doing a research-oriented project, what related research compares to the system you are planning to build.*

*A block diagram showing the major components of the larger system, interconnections, and external interfaces can be helpful. This is not a design or architecture picture. It is more to provide context, especially if your system will interact with external actors. The system you are building should be shown as a black box. Let the design document present the internals.*

*The following subsections describe how the software operates inside various constraints*.

### 2.1.1 System Interfaces

*List each system interface and identify the functionality of the software to accomplish the system requirement and the interface description to match the system. These are external systems that you have to interact with. For instance, if you are building a business application that interfaces with the existing employee payroll system, what is the API to that system that designer’s will need to use?*

### 

### 2.1.2 Interfaces

*Specify:*

1. *The logical characteristics of each interface between the software product and its users.*
2. *All the aspects of optimizing the interface with the person who must use the system*  
     
     
     
   *This is a description of how the system will interact with its users. Is there a GUI, a command line or some other type of interface? Are there special interface requirements? If you are designing for the general student population for instance, what is the impact of ADA (American with Disabilities Act) on your interface?*

### 2.1.3 Hardware Interfaces

*Specify the logical characteristics of each interface between the software product and the hardware components of the system. This includes configuration characteristics. It also covers such matters as what devices are to be supported, how they are to be supported and protocols. This is not a description of hardware requirements in the sense that “This program must run on a Mac with 64M of RAM”. This section is for detailing the actual hardware devices your application will interact with and control. For instance, if you are controlling X10 type home devices, what is the interface to those devices? Designers should be able to look at this and know what hardware they need to worry about in the design. Many business type applications will have no hardware interfaces. If none, just state “The system has no hardware interface requirements” If you just delete sections that are not applicable, then readers do not know if: a. this does not apply or b. you forgot to include the section in the first place.*

### 

### 2.1.4 Software Interfaces

*Specify the use of other required software products and interfaces with other application systems. For each required software product, include:*

1. *Name*
2. *Mnemonic*
3. *Specification number*
4. *Version number*
5. *Source*  
     
     
     
   *For each interface, provide:*
6. *Discussion of the purpose of the interfacing software as related to this software product*
7. *Definition of the interface in terms of message content and format*  
     
     
     
   *Here we document the APIs, versions of software that we do not have to write, but that our system has to use. For instance if your customer uses SQL Server 7 and you are required to use that, then you need to specify i.e.*  
     
   *2.1.4.1 Microsoft SQL Server 7. The system must use SQL Server as its database component. Communication with the DB is through ODBC connections. The system must provide SQL data table definintions to be provided to the company DBA for setup.*  
     
     
     
   *A key point to remember is that you do NOT want to specify software here that you think would be good to use. This is only for* ***customer-specified systems*** *that you* ***have*** *to interact with. Choosing SQL Server 7 as a DB without a customer requirement is a Design choice, not a requirement. This is a subtle but important point to writing good requirements and not over-constraining the design.*

### 2.1.5 Communications Interfaces

*Specify the various interfaces to communications such as local network protocols, etc. These are protocols you will need to directly interact with. If you happen to use web services transparently to your application then do not list it here. If you are using a custom protocol to communicate between systems, then document that protocol here so designers know what to design. If it is a standard protocol, you can reference an existing document or RFC.*

### 2.1.6 Memory Constraints

*Specify any applicable characteristics and limits on primary and secondary memory*. *Don’t just make up something here. If all the customer’s machines have only 128K of RAM, then your target design has got to come in under 128K so there is an actual requirement. You could also cite market research here for shrink-wrap type applications “Focus groups have determined that our target market has between 256-512M of RAM, therefore the design footprint should not exceed 256M.” If there are no memory constraints, so state.*

### 2.1.7 Operations

*Specify the normal and special operations required by the user such as:*

1. *The various modes of operations in the user organization*
2. *Periods of interactive operations and periods of unattended operations*
3. *Data processing support functions*
4. *Backup and recovery operations*   
     
     
     
   *(Note: This is sometimes specified as part of the User Interfaces section.) If you separate this from the UI stuff earlier, then cover business process type stuff that would impact the design. For instance, if the company brings all their systems down at midnight for data backup that might impact the design. These are all the work tasks that impact the design of an application, but which might not be located in software.*

### 2.1.8 Site Adaptation Requirements

*In this section:*

1. *Define the requirements for any data or initialization sequences that are specific to a given site, mission, or operational mode*
2. *Specify the site or mission-related features that should be modified to adapt the software to a particular installation*  
     
     
     
   *If any modifications to the customer’s work area would be required by your system, then document that here. For instance, “A 100Kw backup generator and 10000 BTU air conditioning system must be installed at the user site prior to software installation”.*  
     
   *This could also be software-specific like, “New data tables created for this system must be installed on the company’s existing DB server and populated prior to system activation.” Any equipment the customer would need to buy or any software setup that needs to be done so that your system will install and operate correctly should be documented here.*

## 2.2 Product Functions

*Provide a summary of the major functions that the software will perform. Sometimes the function summary that is necessary for this part can be taken directly from the section of the higher-level specification (if one exists) that allocates particular functions to the software product.*

*For clarity:*

1. *The functions should be organized in a way that makes the list of functions understandable to the customer or to anyone else reading the document for the first time.*
2. *Textual or graphic methods can be used to show the different functions and their relationships. Such a diagram is not intended to show a design of a product but simply shows the logical relationships among variables.*  
     
     
     
   *AH, Finally the real meat of section 2. This describes the functionality of the system in the language of the customer. What specifically does the system that will be designed have to do? Drawings are good, but remember this is a description of what the system needs to do, not how you are going to build it. (That comes in the design document).*

## 2.3 User Characteristics

*Describe those general characteristics of the intended users of the product including educational level, experience, and technical expertise. Do not state specific requirements but rather provide the reasons why certain specific requirements are later specified in section 3.*

*What is it about your potential user base that will impact the design? Their experience and comfort with technology will drive UI design. Other characteristics might actually influence internal design of the system.*

## 2.4 Constraints

*Provide a general description of any other items that will limit the developer's options. These can include:*

*(1) Regulatory policies*

*(2) Hardware limitations (for example, signal timing requirements)*

*(3) Interface to other applications*

*(4) Parallel operation*

*(5) Audit functions*

*(6) Control functions*

*(7) Higher-order language requirements*

1. *Signal handshake protocols (for example, XON-XOFF, ACK-NACK)*
2. *Reliability requirements*  
     
   *(10) Criticality of the application*  
     
   *(11) Safety and security considerations*

*This section captures non-functional requirements in the customers language. A more formal presentation of these will occur in section 3.*

## 2.5 Assumptions and Dependencies

*List each of the factors that affect the requirements stated in the SRS. These factors are not design constraints on the software but are, rather, any changes to them that can affect the requirements in the SRS. For example, an assumption might be that a specific operating system would be available on the hardware designated for the software product. If, in fact, the operating system were not available, the SRS would then have to change accordingly.*

*This section is catch-all for everything else that might influence the design of the system and that did not fit in any of the categories above.*

## 2.6 Apportioning of Requirements.

*Identify requirements that may be delayed until future versions of the system. After you look at the project plan and hours available, you may realize that you just cannot get everything done. This section divides the requirements into different sections for development and delivery. Remember to check with the customer – they should prioritize the requirements and decide what does and does not get done. This can also be useful if you are using an iterative life cycle model to specify which requirements will map to which interation.*

# 3. Specific Requirements

*This section contains all the software requirements at a level of detail sufficient to enable designers to design a system to satisfy those requirements, and testers to test that the system satisfies those requirements. Throughout this section, every stated requirement should be externally perceivable by users, operators, or other external systems. These requirements should include at a minimum a description of every input (stimulus) into the system, every output (response) from the system and all functions performed by the system in response to an input or in support of an output. The following principles apply:*

1. *Specific requirements should be stated with all the characteristics of a good SRS*

* *correct*
* *unambiguous*
* *complete*
* *consistent*
* *ranked for importance and/or stability*
* *verifiable*
* *modifiable*
* *traceable*

1. *Specific requirements should be cross-referenced to earlier documents that relate*
2. *All requirements should be uniquely identifiable (usually via numbering like 3.1.2.3)*
3. *Careful attention should be given to organizing the requirements to maximize readability (Several alternative organizations are given at end of document)*  
     
     
     
   *Before examining specific ways of organizing the requirements it is helpful to understand the various items that comprise requirements as described in the following subclasses. This section reiterates section 2, but is for developers not the customer. The customer buys in with section 2, the designers use section 3 to design and build the actual application.*  
     
     
     
   *Remember this is not design. Do not require specific software packages, etc unless the customer specifically requires them. Avoid over-constraining your design. Use proper terminology:*  
     
   *The system shall… A required, must have feature*  
     
   *The system should… A desired feature, but may be deferred til later*  
     
   *The system may… An optional, nice-to-have feature that may never make it to implementation.*  
     
     
     
   *Each requirement should be uniquely identified for traceability. Usually, they are numbered 3.1, 3.1.1, 3.1.2.1 etc. Each requirement should also be testable. Avoid imprecise statements like, “The system shall be easy to use” Well no kidding, what does that mean? Avoid “motherhood and apple pie” type statements, “The system shall be developed using good software engineering practice”*  
     
     
     
   *Avoid examples, This is a specification, a designer should be able to read this spec and build the system without bothering the customer again. Don’t say things like, “The system shall accept configuration information such as name and address.” The designer doesn’t know if that is the only two data elements or if there are 200. List every piece of information that is required so the designers can build the right UI and data tables.*

## 

## 3.1 External Interfaces

*This contains a detailed description of all inputs into and outputs from the software system. It complements the interface descriptions in section 2 but does not repeat information there. Remember section 2 presents information oriented to the customer/user while section 3 is oriented to the developer.*

*It contains both content and format as follows:*

* *Name of item*
* *Description of purpose*
* *Source of input or destination of output*
* *Valid range, accuracy and/or tolerance*
* *Units of measure*
* *Timing*
* *Relationships to other inputs/outputs*
* *Screen formats/organization*
* *Window formats/organization*
* *Data formats*
* *Command formats*
* *End messages*

## 3.2 Functions

*Functional requirements define the fundamental actions that must take place in the software in accepting and processing the inputs and in processing and generating the outputs. These are generally listed as “shall” statements starting with "The system shall…*

*These include:*

* *Validity checks on the inputs*
* *Exact sequence of operations*
* *Responses to abnormal situation, including*
* *Overflow*
* *Communication facilities*
* *Error handling and recovery*
* *Effect of parameters*
* *Relationship of outputs to inputs, including*
* *Input/Output sequences*
* *Formulas for input to output conversion*

*It may be appropriate to partition the functional requirements into sub-functions or sub-processes. This does not imply that the software design will also be partitioned that way.*

**Functional Requirement:**

|  |
| --- |
| **ID: 01 Use Case Name**: Interact with chatbot  **Trigger**: User wants to use the chatbot  **Precondition**: User is on the chatbot interface  **Actor:** User  **1.** System provides the means to interact by searching  **2.** User submits a question to chatbot  **3.** System displays the relevant answers to the question  **Outcome**: Answer relevant to the search are displayed |

**FR 1.1**

**Description:** The system should have a search option.

**Rationale:** To allow users to easily input specific questions

**Fit Criterion:** A field exists that accepts input which returns answer according to question.

**Originator:** Toua Yang

**FR 1.2**

**Description:** Users can search using any available characters

**Rationale:** Reduce the limits on the search schemes and search freedom

**Fit Criterion:** Any character inputted into the search field will return relevant answers or no answer if none exist matching the search criteria.

**Originator:** Toua Yang

**FR 1.3**

**Description:** Submitting a search request with no input doesn't really submit a search request to the database

**Rationale:** Search submissions that were unintentional should not cause confusion

**Fit Criterion:** A search submission with no input returns a notification explaining the lack of input

**Originator:** Toua Yang

**FR 1.4**

**Description:** System should display answers relevant to the search criteria

**Rationale:** Provide a meaningful search function that returns expected results

**Fit Criterion:** Any question displayed from a search submission must contain a matching segment of characters to what was entered in the search field

**Originator:** Toua Yang

**NFR 1.1**

**Description:** The system should have a user friendly and intuitive interface.

**Rationale:** Users need to be able to quickly learn and understand how to use the chatbot.

**Fit Criterion:** 80% of the user interface design team must agree that the product adheres to modern usability standards.

**Originator:** Chandler Mattila

**NFR 1.2**

**Description:** The product shall not use offensive language.

**Rationale:** With the diverse customer base that will be using the software, there cannot be language used that could be deemed offensive.

**Fit Criterion:** 80% of the human resources team must agree that the language used by chatbot is not offensive.

**Originator:** Chandler Mattila

**NFR 1.3**

**Description:** The system shall provide a quick response.

**Rationale:** The chatbot needs to quickly come up with a response to the users input in order to improve customer satisfaction.

**Fit Criterion:** The chatbot shall reply within 3 seconds of the users input.

**Originator:** Chandler Mattila

**NFR 1.4**

**Description:** The system must be able to manage multiple people simultaneously.

**Rationale:** Many people will be using the chatbot so it needs to be able to support concurrent users.

**Fit Criterion:** 100 people must be able to use the chatbot simultaneously.

**Originator:** Chandler Mattila

|  |
| --- |
| **ID: 02 Use Case Name**: Pay or view current bills  **Trigger**: User wants to pay or view their current bills  **Precondition**: Chatbot confirmed user credentials  **Actor:** User  **1**. User requests to view their bills account.  **2**. System displays user account information.  **3**. User requests to pay their bills account.  **4**. System displays payment methods.  **5.** User inputs necessary information.  **6.** System confirms payment completed.  **Outcome**: Bills is paid successfully. |

**FR 2.1**

**Description:** The product shall allow users to view / pay current bills.

**Rationale:** Users need an automated process to pay and view bills, which can be done

with chatbot.

**Fit Criterion:** The user must be able to successfully pay the bill by using chatbot.

**Originator:** Chandler Mattila

**FR 2.2**

**Description:** The product shall keep users’ payment credentials secure.

**Rationale:** The payment system must be secure to prevent customers from getting their information stolen.

**Fit Criterion:**  80% of the security testing team must agree that the payment system is secure.

**Originator:** Chandler Mattila

**FR 2.3**

**Description:** The system must require identity confirmation before paying the bill.

**Rationale:** Users need to confirm identity before viewing bills, as this information may be confidential.

**Fit Criterion:** Users must not be able to pay bills without confirming identity.

**Originator:** Chandler Mattila

**FR 2.4**

**Description:** A confirmation notice must be sent to the user.

**Rationale:** The user needs to know that the payment successfully went through.

**Fit Criterion:** The user must receive a confirmation notice for each bill they pay.

**Originator:** Chandler Mattila

**NFR 2.1**

**Description:** The format for entering payment information must be clearly displayed and understood.

**Rationale:** The user needs to know exactly where to enter each piece of information for payment.

**Fit Criterion:** 80% of the design team must agree that the payment system is clearly designed so that the user understands where to enter all the information.

**Originator:** Chandler Mattila

**NFR 2.2**

**Description:** There must be an option to enter a precise amount to be paid if the customer cannot pay the full amount.

**Rationale:** Users will not always be able to pay the full amount, so there should be an option to pay a specified amount.

**Fit Criterion:** There should be an option to enter a specific amount to be paid.

**Originator:** Chandler Mattila

**NFR 2.3**

**Description:** The system shall specify which types of payment are accepted.

**Rationale:** Users may use different payment method, so the accepted forms of payment (Paypal, bank transfer, credit card, etc) need to be specified.

**Fit Criterion:** The accepted forms of payment must be specified.

**Originator:** Chandler Mattila

**NFR 2.4**

**Description:** The payment system needs to follow all tax and legal regulations.

**Rationale:** The payment system is dealing with customer payment and will need to follow the respective laws in this regard in order to avoid getting in legal trouble.

**Fit Criterion:** 80% of the legal team must agree that the payment system follows all the necessary laws and regulations.

**Originator:** Chandler Mattila

|  |
| --- |
| **ID: 03 Use Case Name**: Schedule an appointment  **Trigger**: User asks chatbot to schedule an appointment.  **Precondition**: User has opened the chatbot interface.  **Actor:** User  **1.** User asks chatbot to schedule an appointment.  **2.** Chatbot asks user for date and time of desired appointment.  **3.** Chatbot displays the available dates and times that match closest to user’s desired date and time.  **4.**  User selects their preferred appointment time from the list of available times.  **5.** Confirmation email gets sent to the user.  **Outcome**: Appointment is scheduled for the user at the desired date and time. |
|  |

**FR 3.1**

**Description:** The product shall provide a visible feedback when user asks to make a schedule.

**Rationale:** To provide visibility of feedback when user navigates through scheduling.

**Fit Criterion:** Each time a user schedules an appointment there must be a visible feedback to help the user understand the action was completed successfully.

**Originator:** Chandler Mattila

**FR 3.2**

**Description:** The system shall not schedule two appointments at the same time.

**Rationale:** There needs to be protective measures to prevent two people from getting scheduled at the same time as only one person can be scheduled at a time.

**Fit Criterion:** There can be no more than one appointment at a given time.

**Originator:** Chandler Mattila

**FR 3.3**

**Description:** The appointment must be scheduled during the working hours of the clinic.

**Rationale:** There needs to be measures to protect from appointments being scheduled before or after working hours to prevent people from coming when the clinic is not open.

**Fit Criterion:** Each appointment must be scheduled within the working hours of the clinic.

**Originator:** Chandler Mattila

**FR 3.4**

**Description:** If an appointment cannot be made at the desired time, the system must suggest the closest available time.

**Rationale:** Not all appointments will be made at the desired time, so a close alternative must be provided.

**Fit Criterion:** Alternative times must be suggested when a time slot is not available for an appointment.

**Originator:** Chandler Mattila

**NFR 3.1**

**Description:** A confirmation email must be sent to confirm the appointment time for the user.

**Rationale:** Users need to be sure that their appointment was successfully scheduled.

**Fit Criterion:** For each appointment scheduled, there needs to be a confirmation email.

**Originator:** Chandler Mattila

**NFR 3.2**

**Description:** Multiple people must be able to schedule at the same time.

**Rationale:** There will many simultaneous users, so the system must support multiple users scheduling appointments at the same time.

**Fit Criterion:** Fifty people must be able to simultaneously schedule appointments.

**Originator:** Chandler Mattila

**NFR 3.3**

**Description:** The system must be able to store the appointment information for all scheduled appointments.

**Rationale:** The appointments cannot be lost so the database needs to hold all scheduled appointments.

**Fit Criterion:** Each scheduled appointment needs to be stored in a database securely.

**Originator:** Chandler Mattila

**NFR 3.4**

**Description:** Scheduling an appointment must be quick and efficient.

**Rationale:** Users do not want to spend a long time scheduling an appointment.

**Fit Criterion:** It must take 5 minutes or less for 80% of users to schedule an appointment.

**Originator:** Chandler Mattila

|  |
| --- |
| **ID: 04 Use Case Name**: answer FAQ  **Trigger**: User has opened the chatbot  **Precondition**: User has a question in mind to ask the chatbot.  **Actor:** User  **1.** User asks chatbot a frequently asked question.  **2.** Chatbot responds to the question.  **3.** User views the answer.  **4.** User may continue asking questions.  **Outcome**: User gets the answer to their question. |

**FR 4.1**

**Description:** The user must be able to get an answer for a frequently asked question.

**Rationale:** Users are going to have many questions in common so the bot can answer the questions saving time and money.

**Fit Criterion:** If a question qualifies as a FAQ, the chatbot must be able to answer it.

**Originator:** Chandler Mattila

**FR 4.2**

**Description:** The chatbot must have the answer to the most commonly asked questions.

**Rationale:** The system needs to cover a wide enough range of questions to be useful.

**Fit Criterion:** The system must know the answer to at least the top 10 most commonly asked questions.

**Originator:** Chandler Mattila

**FR 4.3**

**Description:** The system utliize amazon web services to answer questions.

**Rationale:** AWS is a suitable technology that can implement the chatbot.

**Fit Criterion:** The cloud computing service used to make chatbot must be AWS.

**Originator:** Chandler Mattila

**FR 4.4**

**Description:** The chatbot must suggest answers to related questions if the system doesn’t know the answer to the question.

**Rationale:** Not all questions will have an answer, so if the question seems like a different question the chatbot knows the answer to, the chatbot should suggest that.

**Fit Criterion:** Questions that are similar to a FAQ but chatbot doesn’t have the answer to should be suggested to the user.

**Originator:** Chandler Mattila

**NFR 4.1**

**Description:** The answers to chatbot must be in English.

**Rationale:** The most common language in the area of the clinic is English.

**Fit Criterion:** The language of the answers must be in English.

**Originator:** Chandler Mattila

**NFR 4.2**

**Description:** The font and text size should be visually appealing.

**Rationale:** Formatting and design plays a role in customer satisfaction.

**Fit Criterion:** 80% of the design team must agree that the formatting and text is visually appealing.

**Originator:** Chandler Mattila

**NFR 4.3**

**Description:** The answers from the chatbot should be reliable.

**Rationale:** The answers need to be reliable so that the user can trust the chatbot’s answers.

**Fit Criterion:** 90% of answers given from the chatbot must be accurate answers to the questions.

**Originator:** Chandler Mattila

**NFR 4.4**

**Description:**

**Rationale:**

**Fit Criterion:**

**Originator:**

## 3.3 Performance Requirements

*This subsection specifies both the static and the dynamic numerical requirements placed on the software or on human interaction with the software, as a whole. Static numerical requirements may include:*

*(a) The number of terminals to be supported*

*(b) The number of simultaneous users to be supported*

*(c) Amount and type of information to be handled*

*Static numerical requirements are sometimes identified under a separate section entitled capacity.*

*Dynamic numerical requirements may include, for example, the numbers of transactions and tasks and the amount of data to be processed within certain time periods for both normal and peak workload conditions.*

*All of these requirements should be stated in measurable terms.*

*For example,*

*95% of the transactions shall be processed in less than 1 second*

*rather than,*

*An operator shall not have to wait for the transaction to complete.*

*(Note: Numerical limits applied to one specific function are normally specified as part of the processing subparagraph description of that function.)*

**NFRx.y.z**

**Description:** The system should respond within 1-3 seconds

**Rationale:** Users should not be inconvenienced by the system

**Fit Criterion:** At least95% of the results falls within the range of 1-3 seconds

**Originator:** Toua Yang

**Customer Satisfaction:** 5 **Customer Dissatisfaction:** 4

**Priority:** 3 **Conflicts:**

**Supporting Materials:** None

## 3.4 Logical Database Requirements

*This section specifies the logical requirements for any information that is to be placed into a database. This may include:*

* *Types of information used by various functions*
* *Frequency of use*
* *Accessing capabilities*
* *Data entities and their relationships*
* *Integrity constraints*
* *Data retention requirements*

*If the customer provided you with data models, those can be presented here. ER diagrams (or static class diagrams) can be useful here to show complex data relationships. Remember a diagram is worth a thousand words of confusing text.*

## 3.5 Design Constraints

*Specify design constraints that can be imposed by other standards, hardware limitations, etc.*

### 3.5.1 Standards Compliance

*Specify the requirements derived from existing standards or regulations. They might include:*

*(1) Report format*

*(2) Data naming*

*(3) Accounting procedures*

*(4) Audit Tracing*

*For example, this could specify the requirement for software to trace processing activity. Such traces are needed for some applications to meet minimum regulatory or financial standards. An audit trace requirement may, for example, state that all changes to a payroll database must be recorded in a trace file with before and after values.*

## 3.6 Software System Attributes

*There are a number of attributes of software that can serve as requirements. It is important that required attributes by specified so that their achievement can be objectively verified. The following items provide a partial list of examples. These are also known as non-functional requirements or quality attributes.*

*These are characteristics the system must possess, but that pervade (or cross-cut) the design. These requirements have to be testable just like the functional requirements. Its easy to start philosophizing here, but keep it specific.*

### 3.6.1 Reliability

*Specify the factors required to establish the required reliability of the software system at time of delivery. If you have MTBF requirements, express them here. This doesn’t refer to just having a program that does not crash. This has a specific engineering meaning.*

### 3.6.2 Availability

*Specify the factors required to guarantee a defined availability level for the entire system such as checkpoint, recovery, and restart. This is somewhat related to reliability. Some systems run only infrequently on-demand (like MS Word). Some systems have to run 24/7 (like an e-commerce web site). The required availability will greatly impact the design. What are the requirements for system recovery from a failure? “The system shall allow users to restart the application after failure with the loss of at most 12 characters of input”.*

**NFRx.y.z**

**Description:**The system shouldappear professional

**Rationale:** To appeal to users and provide a sense of officiality

**Fit Criterion:** 75% of users continue using the system after 3 months

**Originator:** Toua Yang

**Customer Satisfaction: 4 Customer Dissatisfaction: 4**

**Priority: 4 Conflicts:**

**Supporting Materials:** None

### 3.6.3 Security

*Specify the factors that would protect the software from accidental or malicious access, use, modification, destruction, or disclosure. Specific requirements in this area could include the need to:*

* *Utilize certain cryptographic techniques*
* *Keep specific log or history data sets*
* *Assign certain functions to different modules*
* *Restrict communications between some areas of the program*
* *Check data integrity for critical variables*

**NFRx.y.z**

**Description:** The system should protect any data lost

**Rationale:** Users should not fear data lost

**Fit Criterion:** System backs up any lost data

**Originator:** Toua Yang

**Customer Satisfaction:** 5 **Customer Dissatisfaction:** 4

**Priority:** 3 **Conflicts:none**

**Supporting Materials:** None

**NFRx.y.z**

**Description:** The system should protect any user from any cyber attacks

**Rationale:** Users should not fear of confidential

**Fit Criterion:** System is current maintaining high security

**Originator:** Toua Yang

**Customer Satisfaction:** 5 **Customer Dissatisfaction:** 4

**Priority:** 3 **Conflicts:none**

**Supporting Materials:** None

### 3.6.4 Maintainability

*Specify attributes of software that relate to the ease of maintenance of the software itself. There may be some requirement for certain modularity, interfaces, complexity, etc. Requirements should not be placed here just because they are thought to be good design practices. If someone else will maintain the system*

### 3.6.5 Portability

*Specify attributes of software that relate to the ease of porting the software to other host machines and/or operating systems. This may include:*

* *Percentage of components with host-dependent code*
* *Percentage of code that is host dependent*
* *Use of a proven portable language*
* *Use of a particular compiler or language subset*
* *Use of a particular operating system*

*Once the relevant characteristics are selected, a subsection should be written for each, explaining the rationale for including this characteristic and how it will be tested and measured. A chart like this might be used to identify the key characteristics (rating them High or Medium), then identifying which are preferred when trading off design or implementation decisions (with the ID of the preferred one indicated in the chart to the right). The chart below is optional (it can be confusing) and is for demonstrating tradeoff analysis between different non-functional requirements. H/M/L is the relative priority of that non-functional requirement.*

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **ID** | **Characteristic** | **H/M/L** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** |
| 1 | Correctness |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 2 | Efficiency |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 3 | Flexibility |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 4 | Integrity/Security |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 5 | Interoperability |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 6 | Maintainability |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 7 | Portability |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 8 | Reliability |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 9 | Reusability |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 10 | Testability |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 11 | Usability |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 12 | Availability |  |  |  |  |  |  |  |  |  |  |  |  |  |

*Definitions of the quality characteristics not defined in the paragraphs above follow.*

*• Correctness - extent to which program satisfies specifications, fulfills user’s mission objectives*

*• Efficiency - amount of computing resources and code required to perform function*

*• Flexibility - effort needed to modify operational program*

*• Interoperability - effort needed to couple one system with another*

*• Reliability - extent to which program performs with required precision*

*• Reusability - extent to which it can be reused in another application*

*• Testability - effort needed to test to ensure performs as intended*

*• Usability - effort required to learn, operate, prepare input, and interpret output*

*THE FOLLOWING (3.7) is not really a section, it is talking about how to organize requirements you write in section 3.2. At the end of this template there are a bunch of alternative organizations for section 3.2. Choose the ONE best for the system you are writing the requirements for.*

## 3.7 Organizing the Specific Requirements

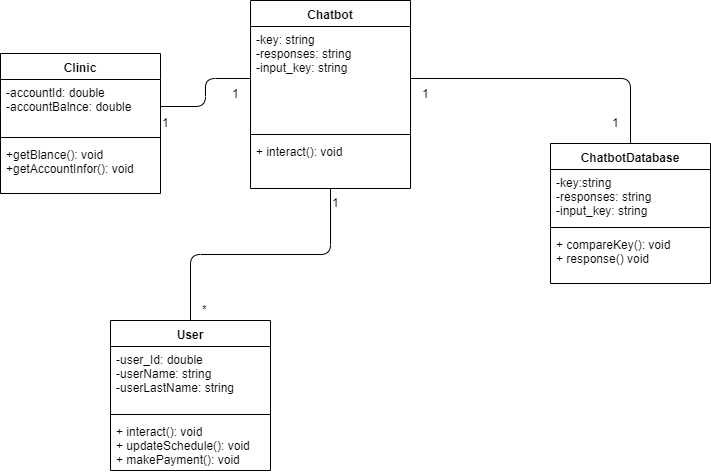
*For anything but trivial systems the detailed requirements tend to be extensive. For this reason, it is recommended that careful consideration be given to organizing these in a manner optimal for understanding. There is no one optimal organization for all systems. Different classes of systems lend themselves to different organizations of requirements in section 3. Some of these organizations are described in the following subclasses.*

### 3.7.1 System Mode

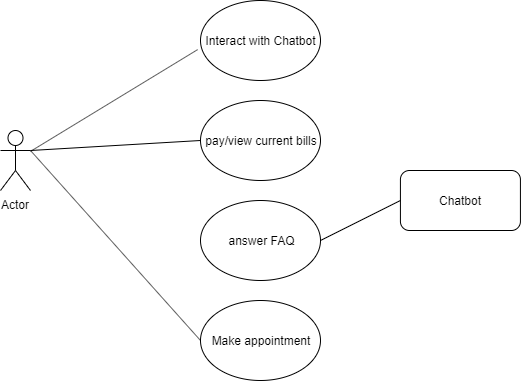
*Some systems behave quite differently depending on the mode of operation. When organizing by mode there are two possible outlines. The choice depends on whether interfaces and performance are dependent on mode.*

### 3.7.2 User Class

<https://drive.google.com/file/d/1_PFulkOCnTRiHT-p3gjGbKaKHWjNED1p/view?usp=sharing>



EDITABLE - https://drive.google.com/file/d/1kIq6AENcGfc8-uu9fwqmqKiqqIpItF7K/view?usp=sharing



*Some systems provide different sets of functions to different classes of users.*

### 

### 3.7.3 Objects

*Objects are real-world entities that have a counterpart within the system. Associated with each object is a set of attributes and functions. These functions are also called services, methods, or processes. Note that sets of objects may share attributes and services. These are grouped together as classes.*

### 3.7.4 Feature

*A feature is an externally desired service by the system that may require a sequence of inputs to effect the desired result. Each feature is generally described in as sequence eof stimulus-response pairs.*

### 3.7.5 Stimulus

*Some systems can be best organized by describing their functions in terms of stimuli.*

### 3. 7.6 Response

*Some systems can be best organized by describing their functions in support of the generation of a response.*

### 3.7.7 Functional Hierarchy

*When none of he above organizational schemes prove helpful, the overall functionality can be organized into a hierarchy of functions organized by either common inputs, common outputs, or common internal data access. Data flow diagrams and data dictionaries can be use dot show the relationships between and among the functions and data.*

## 3.8 Additional Comments

*Whenever a new SRS is contemplated, more than one of the organizational techniques given in 3.7 may be appropriate. In such cases, organize the specific requirements for multiple hierarchies tailored to the specific needs of the system under specification.*

*Three are many notations, methods, and automated support tools available to aid in the documentation of requirements. For the most part, their usefulness is a function of organization. For example, when organizing by mode, finite state machines or state charts may prove helpful; when organizing by object, object-oriented analysis may prove helpful; when organizing by feature, stimulus-response sequences may prove helpful; when organizing by functional hierarchy, data flow diagrams and data dictionaries may prove helpful.*

*In any of the outlines below, those sections called “Functional Requirement i” may be described in native language, in pseudocode, in a system definition language, or in four subsections titled: Introduction, Inputs, Processing, Outputs.*

# 4. Change Management Process

*Identify the change management process to be used to identify, log, evaluate, and update the SRS to reflect changes in project scope and requirements. How are you going to control changes to the requirements. Can the customer just call up and ask for something new? Does your team have to reach consensus? How do changes to requirements get submitted to the team? Formally in writing, email or phone call?*

# 5. Document Approvals

*Identify the approvers of the SRS document. Approver name, signature, and date should be used.*

# 6. Supporting Information

*The supporting information makes the SRS easier to use. It includes:*

* *Table of Contents*
* *Index*
* *Appendices*

*The Appendices are not always considered part of the actual requirements specification and are not always necessary. They may include:*

*(a) Sample I/O formats, descriptions of cost analysis studies, results of user surveys*

*(b) Supporting or background information that can help the readers of the SRS*

*(c) A description of the problems to be solved by the software*

*(d) Special packaging instructions for the code and the media to meet security, export, initial loading, or other requirements*

*When Appendices are included, the SRS should explicitly state whether or not the Appendices are to be considered part of the requirements.*

Tables on the following pages provide alternate ways to structure section 3 on the specific requirements. You should pick the best one of these to organize section 3 requirements.

**Outline for SRS Section 3**

**Organized by mode: Version 1**

3. Specific Requirements

3.1 External interface requirements

1. User interfaces
2. Hardware interfaces
3. Software interfaces
4. Communications interfaces
5. Functional requirements

3.2.1 Mode 1

3.2.1.1 Functional requirement 1.1

.....

3.2.1.*n* Functional requirement 1.*n*

1. Mode 2

.....

3.2.*m* Mode *m*

3.2.*m*.1 Functional requirement *m*.1

.....

3.2.*m.n* Functional requirement *m.n*

3.3 Performance Requirements

3.4 Design Constraints

3.5 Software system attributes

3.6 Other requirements

**Outline for SRS Section 3**

**Organized by mode: Version 2**

3. Specific Requirements

3.1 Functional Requirements

1. Mode 1

3.1.1.1 External interfaces

3.1.1.1 User interfaces

3.1.1.2 Hardware interfaces

3.1.1.3 Software interfaces

3.1.1.4 Communications interfaces

3.1.1.2 Functional Requirement

3.1.1.2.1 Functional requirement 1

.....

3.1.1.2.*n* Functional requirement *n*

3.1.1.3 Performance

3.1.2 Mode 2

.....

3.1.*m* Mode *m*

1. Design constraints
2. Software system attributes
3. Other requirements

**Outline for SRS Section 3**

**Organized by user class (i.e. different types of users ->System Adminstrators, Managers, Clerks, etc.)**

3. Specific Requirements

3.1 External interface requirements

1. User interfaces
2. Hardware interfaces
3. Software interfaces
4. Communications interfaces
5. Functional requirements

3.2.1 User class 1

3.2.1.1 Functional requirement 1.1

.....

3.2.1.*n* Functional requirement 1.*n*

1. User class 2

.....

3.2.*m* User class *m*

3.2.*m*.1 Functional requirement *m*.1

.....

3.2.*m.n* Functional requirement *m.n*

3.3 Performance Requirements

3.4 Design Constraints

3.5 Software system attributes

3.6 Other requirements

**Outline for SRS Section 3**

**Organized by object (Good if you did an object-oriented analysis as part of your requirements)**

3 Specific Requirements

3.1 External interface requirements

1. User interfaces
2. Hardware interfaces
3. Software interfaces
4. Communications interfaces
5. Classes/Objects

3.2.1 Class/Object 1

3.2.1.1 Attributes (direct or inherited)

1. Attribute 1

.....

3.2.1.1.*n* Attribute *n*

1. Functions (services, methods, direct or inherited)

3.2.1.2.1 Functional requirement 1.1

.....

3.2.1.2.*m* Functional requirement 1.*m*

3.2.1.3 Messages (communications received or sent)

3.2.2 Class/Object 2

.....

3.2.*p* Class/Object *p*

3.3 Performance Requirements

3.4 Design Constraints

3.5 Software system attributes

3.6 Other requirements

**Outline for SRS Section 3**

**Organized by feature (Good when there are clearly delimited feature sets.**

3 Specific Requirements

3.1 External interface requirements

1. User interfaces
2. Hardware interfaces
3. Software interfaces
4. Communications interfaces
5. System features

3.2.1 System Feature 1

3.2.1.1 Introduction/Purpose of feature

3.2.1.2 Stimulus/Response sequence

3.2.1.3 Associated functional requirements

3.2.1.3.1 Functional requirement 1

.....

3.2.1.3.*n* Functional requirement *n*

3.2.2 System Feature 2

.....

3.2.*m* System Feature *m*

.....

3.3 Performance Requirements

3.4 Design Constraints

3.5 Software system attributes

3.6 Other requirements

**Outline for SRS Section 3**

**Organized by stimulus (Good for event driven systems where the events form logical groupings)**

3 Specific Requirements

3.1 External interface requirements

1. User interfaces
2. Hardware interfaces
3. Software interfaces
4. Communications interfaces
5. Functional requirements

3.2.1 Stimulus 1

3.2.1.1 Functional requirement 1.1

.....

3.2.1.*n* Functional requirement 1.*n*

3.2.2 Stimulus 2

.....

3.2.*m* Stimulus *m*

3.2.*m*.1 Functional requirement *m*.1

.....

3.2.*m.n* Functional requirement *m.n*

3.3 Performance Requirements

3.4 Design Constraints

3.5 Software system attributes

3.6 Other requirements

**Outline for SRS Section 3**

**Organized by response (Good for event driven systems where the responses form logical groupings)**

3 Specific Requirements

3.1 External interface requirements

1. User interfaces
2. Hardware interfaces
3. Software interfaces
4. Communications interfaces
5. Functional requirements

3.2.1 Response 1

3.2.1.1 Functional requirement 1.1

.....

3.2.1.*n* Functional requirement 1.*n*

3.2.2 Response 2

.....

3.2.*m* Response *m*

3.2.*m*.1 Functional requirement *m*.1

.....

3.2.*m.n* Functional requirement *m.n*

3.3 Performance Requirements

3.4 Design Constraints

3.5 Software system attributes

3.6 Other requirements

**Outline for SRS Section 3**

**Organized by functional hierarchy (Good if you have done structured analysis as part of your design.)**

3 Specific Requirements

3.1 External interface requirements

1. User interfaces
2. Hardware interfaces
3. Software interfaces
4. Communications interfaces
5. Functional requirements

3.2.1 Information flows

3.2.1.1 Data flow diagram 1

1. Data entities
2. Pertinent processes
3. Topology

3.2.1.2 Data flow diagram 2

1. Data entities
2. Pertinent processes
3. Topology

.....

3.2.1.*n* Data flow diagram *n*

3.2.1.*n*.1 Data entities

3.2.1.*n*.2 Pertinent processes

3.2.1.*n*.3 Topology

3.2.2 Process descriptions

1. Process 1
2. Input data entities
3. Algorithm or formula of process
4. Affected data entities

3.2.2.2 Process 2

3.2.2.2.1 Input data entities

3.2.2.2.2 Algorithm or formula of process

3.2.2.2.3 Affected data entities

.….

3.2.2.*m* Process *m*

3.2.2.*m*.1 Input data entities

3.2.2.*m*.2 Algorithm or formula of process

3.2.2.*m*.3 Affected data entities

3.2.3 Data construct specifications

3.2.3.1 Construct 1

3.2.3.1.1 Record type

3.2.3.1.2 Constituent fields

3.2.3.2 Construct 2

3.2.3.2.1 Record type

3.2.3.2.2 Constituent fields

…..

3.2.3.*p* Construct *p*

3.2.3.*p*.1 Record type

3.2.3.*p*.2 Constituent fields

3.2.4 Data dictionary

3.2.4.1 Data element 1

3.2.4.1.1 Name

3.2.4.1.2 Representation

3.2.4.1.3 Units/Format

3.2.4.1.4 Precision/Accuracy

3.2.4.1.5 Range

3.2.4.2 Data element 2

3.2.4.2.1 Name

3.2.4.2.2 Representation

3.2.4.2.3 Units/Format

3.2.4.2.4 Precision/Accuracy

3.2.4.2.5 Range

…..

3.2.4.*q* Data element *q*

3.2.4.*q*.1 Name

3.2.4.*q*.2 Representation

3.2.4.*q*.3 Units/Format

3.2.4.*q*.4 Precision/Accuracy

3.2.4.*q*.5 Range

3.3 Performance Requirements

3.4 Design Constraints

3.5 Software system attributes

3.6 Other requirements

**Outline for SRS Section 3**

**Showing multiple organizations (Can’t decide? Then glob it all together)**

3 Specific Requirements

3.1 External interface requirements

1. User interfaces
2. Hardware interfaces
3. Software interfaces
4. Communications interfaces
5. Functional requirements

3.2.1 User class 1

3.2.1.1 Feature 1.1

3.2.1.1.1 Introduction/Purpose of feature

3.2.1.1.2 Stimulus/Response sequence

3.2.1.1.3 Associated functional requirements

3.2.1.2 Feature 1.2

3.2.1.2.1 Introduction/Purpose of feature

3.2.1.2.2 Stimulus/Response sequence

3.2.1.2.3 Associated functional requirements

…..

3.2.1.*m* Feature 1.*m*

3.2.1.*m*.1 Introduction/Purpose of feature

3.2.1.*m*.2 Stimulus/Response sequence

3.2.1.*m*.3 Associated functional requirements

3.2.2 User class 2

.....

3.2.*n* User class *n*

.....

3.3 Performance Requirements

3.4 Design Constraints

3.5 Software system attributes

3.6 Other requirements

**Outline for SRS Section 3**

**Organized by Use Case (Good when following UML development)**

3. Specific Requirements

3.1 External Actor Descriptions

3.1.1 Human Actors

3.1.2 Hardware Actors

3.1.3 Software System Actors

3.2 Use Case Descriptions

3.2.1 Use Case 1

3.2.2 Use Case 2

3.2.n Use Case n

3.3 Performance Requirements

3.4 Design Constraints

3.5 Software system attributes

3.6 Other requirements