CIS-11 Project Documentation Template

Team Name: A+sian Assembly

Team Members: Joseph Sary, Chloe-Jane Cuevas, Darren Mcmillon
Project Name: Option A Bubble Sort
Date: 5/25/25

Advisor: Kasey Nguyen, PhD

Part I – Application Overview

This part of the requirements document serves to present the "big picture" of the application. Here you lay out the objectives of the application, how it fits into the business process of the company, and how it relates to other software systems. The sections listed below should be included in this part of the requirements document.

Objectives

In this section you state the commonly accepted objectives of the project.

You must determine the business objectives of the project early on; without clear objectives your project has little chance of succeeding anyway so it does not make sense to move on until the objectives are agreed upon.

Why are we doing this?

The goal of our project is to create an LC3 Program that will utilize bubble-sort to sort any set of data in order to show our understanding of the LC3 assembly language.

To elicit the objectives, ask the business expert, the development manager, and the project sponsor the following questions:

- **What business objectives of the company will this project help achieve? Possible objectives might be reducing costs, improving the customer service, simplifying the workflow, replacing obsolete technology, piloting a new technology, and many others. Also, make sure you understand exactly how the proposed project will help accomplish the stated objective.**
- **Why are we doing this project now? What will happen if we do it later? What if we do not do it at all?**
- **Who will benefit from this project? Do the people who will benefit from it consider it the most important improvement that can possibly be made at this time? Should we be doing a different project instead?**

Business Process

In this section you describe the business process and how your application will be used in this context.

In some cases you will need two sections of this sort – one describing the existing business process using existing systems and the other describing the future business process using the system you are developing. This happens any time the business process changes once your system is introduced. When this is the case, the purpose of describing the existing business process is to have a basis of reference to explain the new business process.

If the business process won't change when your application is introduced, you should be able to describe it in a single section. However, in this case be sure you understand and communicate to others what value your application brings to the customers. This question should be answered in the Objectives section.

N/A

User Roles and Responsibilities

In this section you describe who the users are and how the system fits into what they do.

You need to list all users for your system in terms of user roles*. Typically each individual performs multiple roles in the course of his work since his job involves meeting multiple business objectives. A user role is related to meeting a specific business objective. When gathering requirements it is most useful to consider roles since you will want to focus only on those business objectives that are relevant to your application.*

For each role you need to list the tasks that involve the use of your system (directly or indirectly). You also need to describe the relationships among the tasks for each individual user role and the hand-offs from one role to another. This is usually represented as a workflow diagram.

Consider time in describing tasks and their relationships – different sets of tasks may be performed at different times (daily, monthly, etc.) and several workflow diagrams may be needed.

Once you have written the Objectives, Business Process, and the User Roles and Responsibilities sections, give them to the business expert to read. If you and he agree on what's written, congratulations! You are well on your way to understanding what needs to be done.

The user of the program will be responsible for inputting 8 numbers from 1-100 that will then be sorted by the algorithm. The user will then receive said data as a console output.

Production Rollout Considerations

In this section you describe the strategy for production rollout.

In addition, either this section, or an appendix in the requirements document, or a separate document should include the discussion of populating the system data for rollout and the discussion of the expected data and transaction volume.

N/A

Terminology

In this section you define the business terms used in the requirements document.

You should include this section even if at first it seems like a waste of everyone's time. Once you show it to people you may be surprised to learn that not everyone understood the terms the same way after all!

N/A

Part II – Functional Requirements

This part of the requirements document states in a detailed and precise manner what the application will do.

Statement of Functionality

In this section you state **precisely** *what the application will do.*

This part, more than anything else in the requirements document, spells out your contract with your customers. The application will ***include all functions listed here and will not include any of the functions not listed****.*

In this section you must use as precise language as you can since the developers will use it to code the application. When reviewing this part with other people you should pay extreme attention to removing any possibility for ambiguous interpretation of any of the requirements.

If your application has several distinct categories of users, you can list the requirements by user category. User categories may be defined in terms of their job title (clerk, manager, administrator), the frequency with which they will use the system (heavy or casual), the purpose for which they will use the system (operational decisions, long-term decisions), etc. If each category of users uses the system in its own way, structuring the requirements based on user category will make sense.

If your application deals with several kinds of real-world objects, you can list the requirements by object. For example, for a reservation system a booking is an important object, and you may want to list all requirements pertaining to bookings in one sub-section.

One of the most common approaches is to list the requirements by feature. For example, features of a word processing application are file management, formatting, editing, etc.

This program will use these main functions:

- Gathering input
    - receive 8 different values from 1-100 from the user and store them into memory.
- Processing data
    - process the data stored into memory and sort the data in order from least to greatest.
- Output to user
    - read through sorted data and print out to the user.

Scope

In this section you state what functionality will be delivered and in which phase.

You should include this section if your development consists of multiple phases. As an alternative to this section, you can note the planned project phase for each feature in the functionality statement section. Usually, it is better to include a separate scope section for easy reference and communication.

Once completed, this program will be able to sort a set of numbers, given that the user provides 8 numbers that are between 1-100 respectively.

Performance

In this section you describe any specific performance requirements.

You should be very specific and use numeric measures of performance. Stating that the application should open files quickly is not a performance requirement since it is ambiguous and cannot be verified. Stating that opening a file should take less than 3 seconds for 90% of the files and less than 10 seconds for every file is a requirement.

Instead of providing a special section on performance requirements, you may include the relevant information for each feature in the statement of functionality.

The program must require that input inserted by the user will not damage the program and instead, simply reject the user's attempt for invalid input. The program must also always sort the data as long as the data is within the proper constraints.

Usability

In this section you describe any specific usability requirements.

You need to include this section only if there are any "overarching" usability goals and considerations. For example, the speed of navigation of the UI may be such a goal. As in the previous section, use numeric measures of usability whenever possible.

Usability requirements include an input checking system that will reject improper inputs such as character inputs, or numerical inputs that are out of scope.

Documenting Requests for Enhancements

There does come a time when the requirements for the initial release of your application

are frozen. Usually, it happens after the system acceptance test which is the last chance for the users to lobby for some changes to be introduced in the upcoming release.

Currently, you need to begin maintaining the list of requested enhancements. Below is a template for tracking requests for enhancements.

Date	Enhancement	Requested by	Notes	Priority	Release No/ Status

Part III – Appendices

Appendices are used to capture any information that does not fit naturally anywhere else in the requirements document yet is important. Here are some examples of appendices.

Supporting and background information may be appropriate to include as an appendix – things like results of user surveys, examples of problems to be solved by the applications, etc. Some of the supporting information may be graphical – remember all those charts you drew trying to explain your document to others?

Appendices can be used to address a specialized audience. For example, some information in the requirements document may be more important to the developers than to the users. Sometimes this information can be put into an appendix.

LC3 Documentation Manual - link

Flow chart or pseudo-code.

Include branching, iteration, subroutines/functions in flowchart or pseudocode'..

```
MAIN
        init
        num1            .fill x3150-x3157
        num2
        num3
        num4
        num5
        num6
        num7
        num8
        init_list
        count = 8
        base_addr = num1
        init_loop
        if count < 8
                store INPUT at base_addr + count        (essentially num[count] = input)
                count -= 1
                jmp init_loop
        SORT            ;sort memory locations x3150 - x3157
        OUTPUT


INPUT ;function to find single entry for sorted list. will happen 8 times to find unordered set
        digit1 = 0
```

```
digit2 = 0
digit3 = 0
sum = 0
print(type entry)
READFIRST
        if input1 = enter
                jmp DONE
        convert from number to ascii
        subtract 30 and 39 to error check invalid
        if invalid
                go to INPUTERROR_1
        digit1 = input1
READSECOND          ; check second digit input
        if input = enter
                jmp DONE
        convert from number to ascii
        subtract 30 and 39 (ascii)
        if invalid
                go to INPUTERROR_2
        digit1 = 10 * digit1        ; gets moved one place over
        digit2 = input2
READTHIRD
        if input = enter
                jmp DONE
        convert from num to ascii
        subtract 30 and 39 (ascii)
        if invalid
                go to INPUTERROR_3
        digit1 = 10 * digit1        ;gets moved to hundreds place
        digit2 = 10 * digit2        ;gets moved to tens place
        digit3 = input3



INPUTERROR_1
        print input is not a num
        go to READFIRST to try again
INPUTERROR_2
        print input is not a num
        go to READSECOND to try again
INPUTERROR_3
        print input is not a num
        go to READTHIRD to try again
INPUT_INVALID
        print input is too large try again
        jmp to READFIRST
DONE
        sum += digit1, digit2, digit3
        if sum > 100
                INPUT_INVALID
        if sum = 0
                print user quit (if they dont put anything, they shouldn't be reprompted)
        return sum        ; main function responsible for placing the number into memory
```

```
SORT
        base_addr = num1 addr
        count = 0
        swapped = 0

        bubble_loop
        num1= num[base_addr + count]
        num1= num[base_addr + count + 1]
        i = 0
        sort_i
        if i < 7
                swapped = false          ;clear swapped
                j = 0
                if j < 7 - i
                        if num[j] > num[j+1]
                                swap(count)
                                swapped = true
                        j += 1
                        jmp sort_j
        if swapped = 0  ; if it goes through whole loop without swapping, then in order
                ret
        i += 1
        jmp sort_i

OUTPUT                  ; prints memory locations x3150-x3157
        base_addr = num1 addr;
        count = 0
        output_loop
        if count < 7
                print "num [count]: "
                convert num[base_addr + 0] to ascii
                print num
                jmp output_loop

MULT
        num1;
        num2;
        sign = 1        ; 1 for pos, -1 for neg
        if num1 < 0
                sign = -sign
                num1 = - num1           ;flip to non neg
        if num2 < 0
                sign = -sign
                num2 = -num2            ;flip to non neg
        count = num2
        sum = 0
        mult_loop
        if count > 0
                sum += num1
                num2 -= 1
                jmp mult_loop
        if sign = -1
```

```
        sum = -sum

SWAP
        baseaddr
        count
        num1 = baseaddr + count;        ;load
        num2 = baseaddr + count + 1;    ;load
        tmp = num1                      ;init
        num1 = num2                     ;store
        num2 = tmp                      ;store
```