**Software Requirements Specification (SRS) Review**

Reviewed by Frazer Bayley

March 8, 2017

This review the mirrors the layout of the actual SRS with suggested changes/additions/deletions mapped to each of their corresponding sections. Action items are colored red.

1. Introduction

* 1. Intended Audience and Purpose
* This section successfully and clearly specifies who the document is for.
* This section does not clearly define purpose of each user
  + Needs to go further and more specific in how each member of the team will use the document (ie. Product Manager, QA Personnel, Developer…)
  + Look at this example <https://app.assembla.com/spaces/cis422f12_team1/wiki/Software_Requirements>
  1. How to use this Document
* Needs to describe the document organization so that the document is navigable.
  + Combine <https://app.assembla.com/wiki/show/cis422w12_team3/Software_Requirements>

With

<https://app.assembla.com/spaces/cis422f12_team1/wiki/Software_Requirements>

so that there are links to each section of the SRA with brief descriptions of each.

2. Concept of Operations

2.1 Proposed System + 2.2 Scope

* Good background information, to reduce confusion remove section titles and just place it under concept of operations. Proposed system and scope is not a required part of the SRA.

2.3 System Context + 2.4 System Capabilities

* Great description of user interaction.

2.5 Process Descriptions

* Move somewhere else or is unnecessary. This section isn’t required by the SRA specifications

2.6 Use Cases

1. Loader 🡨 rename this, too technical.

2. Actors

* + - Just use User, debugger is assumed (change for all use cases)

4. Basic Flow

* + - Break thse into separately titled Alternate flows:
      * Direct input
      * Copy/Paste

5. Explicitly Name these as Alternate Flows

* + - ie. Alternate Flow: Direct Input

6. Post Conditions

* + - avoid looking into the future

2. Assemble Function

3. Precondition

* + - Mention use case for loading

4. Basic Flow

* + - Assume code compiles
    - Have failure be an alternate flow

5. Alternate Flows

* + - Break into separate alternate flows
    - Remover user closes debugger (for all use cases)

3. Step Functions

3. Preconditions

* + - Mention Assemble use case, with all post conditions true

4. Basic Flow

* + - Describe what happens after the user pushes button

5. Alternate Flow

* + - Place case when user keeps stepping for n iterations

4. Step Over

* + Same changes as Step

5. Execute

5. Alternate Flows

* + - No alternate flows for this

6. Memory Dump Function

* + No changes

7. ADD MORE USE CASES:

* + Save Use cases
  + Edit code use case
  + Close program use case

3. Detailed Requirements

* Break down into inputs and output sections

4. Quality Requirements

* Good

5. Expected Subsets

* Define the increments of the projects (L0, L1,…)
  + Specifying how they link up with requirements

6. Fundamental Assumptions

* Good

7. Expected Changes

* Need to describe future and expected changes