POS-Plus Co.

GUI Design and Database Module

Software Requirements Specification Document

Team Angstrom

Version: (1) Date: 9-21-2018

Table of Contents

1. **Introduction**
   1. Purpose
   2. Scope
   3. Definitions, Acronyms, and Abbreviations
   4. References
   5. Overview
2. **Overall Description**
   1. Product Perspective
      1. System Interfaces
      2. User Interfaces
      3. Hardware Interfaces
      4. Software Interfaces
      5. Communication Interfaces
      6. Memory Constraints
      7. Operations
      8. Site Adaption Requirements
   2. Product Functions
   3. User Characteristics
   4. Constraints
   5. Assumption and Dependencies
   6. Apportioning of Requirements
3. **Specific Requirements**
   1. External Interfaces
   2. Functions
      1. List of Requirements
      2. Use-Cases
      3. Domain Model
   3. Performance Requirements/Non-functional Requirements
   4. Logical Database Requirements
   5. Design Constraints
      1. Standards Compliance
   6. System Software Attributes
      1. Reliability
      2. Availability
      3. Security
      4. Maintainability
      5. Portability
   7. Organizing the Specific Requirements
      1. System Mode
      2. User Class
      3. Objects
      4. Features
      5. Stimulus
      6. Response
      7. Functional Hierarchy
4. Project Changes Management Process
5. Document Approvals
6. Supporting Information
7. **Introduction**

This section explains the purpose, scope, and overview of Team Angstrom’s product POS-Plus. Below, we will detail the specifications objectives, and goals of POS-Plus as well as the features the client can expect to be represented in the final iteration of our product. The aim of this Software Requirements Specification (SRS) document is to serve as both a roadmap into the design process of POS-Plus and a contract between the client and our company that represents that the client understands the full-breadth of functionality that our product provides. Thereby, the contents of this document should not be glanced over… because by agreeing with the contents of the document, the client is also agreeing to accept the current iteration of POS-Plus upon its purchase.

* 1. Purpose

POS-Plus includes a fully customizable and visual user-interface through the prudent use of modules. It aims to provide for the user one less thing to worry about as a start-up business because it is so affordable and because our graphical user interface is so versatile and easy to use. Our project will provide affordability without sacrificing ease-of-use or quality. Our target customer base is startup businesses because of our product’s low cost and its low barrier-to-entry.

* 1. Scope

The aforementioned name of our product is POS-Plus. Our product is at its core a POS system and a database module. As a POS system, our product provides the functionality of a sales and transaction tool and a shift management system. As a database module, our product provides the functionality of an inventory management system. The visual modules incorporated within POS-Plus’s user-interface are all customizable cosmetically and a selection of tools are available for outfitting and modifying the any of the existing visual components in place within the module with other visual components.

We emphasize that all the modifications to the core system modules are purely cosmetic. Because the system’s core modules are integral to maintaining the basic functionality of POS product, our product necessitates that we limit the degree of modification that can be performed on them. Still the client has access to a wide range of visual tools that can be utilized to contribute to a high degree of personalization.

The restrictions that POS-Plus enforces on module modifications are really only meaningful from a software developer’s viewpoint, and typically go unnoticed be the vast majority of our company’s client-base. In fact, these limitations actually stand to ensure that POS-Plus can continue to provide its full-range of functions throughout duration of the company-clientele contract life-cycle.

POS-Plus also aims to provide extensions to its functionality by supporting the operation of third-party, functional widgets that upon incorporation superimpose an additional layer of functionality upon its core functionality. The realization of this widget concept will be pursued through the use of some “app-store” model.

Since there are concerns about its implementation interfering with security of the system and additional concerns about its implementation burdening the development process with additional time constraints, this concept is chiefly confined to the domain of theory and will not be a primary focus throughout the scope of the development process.

1.3. Definitions, Acronyms, and Abbreviations

**POS system** – stands for point-of-sale system, and as James Dyson Foundation1 specifies, “the point of sale (POS) or point of purchase (POP) is the time and place where a retail transaction is completed.”

The point of sale system is the system in which the merchant carries out the process of calculating the amount owed by the customer, indicating that amount, may prepare an invoice for the customer (which may be a cash register printout), and indicating the options for the customer to make payment at the point of sale.

**GUI** – stands for graphical user interface, and as Software Terms2 specifies, “it is a user interface that includes graphical elements, such as windows, icons and buttons.”

**Addendum –** as our Software Requirements Specification document calls for the use of more nebulous acronyms, abbreviations, and technical jargon, we will update this document accordingly with the appropriate definitions for those terms.

* 1. References

1. "Paperless Receipt Solution (PRS) System". James Dyson Foundation. Retrieved August 9, 2015.
2. "GUI Definition". TechTerms. Archived from the original on 11 May 2009. Retrieved 19 February 2010.
   1. Overview

This SRS document attempts to organize its contents through the use of a functional hierarchy. Components that share common inputs, common outputs, and common internal access data of essential functions are grouped and prioritize first in the hierarchy. While components that share characteristics of less-essential functions are prioritized lower within the hierarchy. The remainder of this document will detail the specifics of the system requirements that our product, POS-Plus, necessitates.

1. **Overall Description**

There are factors that influence the outcome of POS-Plus’s software development process. Factors including dependencies, constraints, and assumptions, among many others, that this section seeks to account for and then chart their effects in enacting the final version of our product in the effort of making POS-Plus’s software development process transparent to its client-base.

* 1. Product Perspective

The realization POS-Plus entails the collaboration between two sub-components, a POS system and a database module. In this regard, given that we view the database module as an accessory component, our project is relatively self-contained and aligned with our goal of implementing a functional and customizable point-of-sale software. The POS system and the database module will require a Graphical User Interface (GUI) to fetch and display I/O data from user interactions, and some kind of controller structure to handle and process interactions between the database module, point-of-sale system, and the user.

2.1.1. System Interfaces

(List each system interface and identify the functionality that it

employs to handle the system requirement and then its description

relative to the system’s depiction/utilization of it.)

2.1.2. User Interfaces

(Detail the characteristics of each user interface that your software

employs and detail the aspects of optimizing the layout of information

that is displayed directly to user)

2.1.3. Hardware Interfaces

(Does our project require the implementation/use of any hardware

accessory [e.g. some biometric scanner, ‘raspberry pie’, ‘quiz bowl’

button device, etc.] If so, enumerate all such hardware components, and

describe how your software and the hardware component interact while

detailing the specifics of the hardware configuration characteristics,

protocols for handling I/O from said components, and then the support

for representing/interpreting data from the components.)

2.1.4. Software Interfaces

(I am assuming that this section is for enumerating all third-party software

that works in tandem with ours)

**Accessory Software Component 1:**

* Name
* Mnemonic
* Specification
* Version number
* Source
* [Discussion of the purpose of the interfacing software component]
* [Definition of the interface in terms of message content and format]

2.1.5 Communication Interfaces

(Why do local networks need interfaces for communicating?

The communication is between machines and not humans and

machines)

2.1.6. Memory Constraints

(We don’t know how to do this yet)

2.1.7. Operations

(Specify the normal and special operations required by the user [e.g.

various modes of operations in the user organization (?), periods of

interactive operations and periods of unattended operations, data pro-

cessing support functions, backup and recovery operations]. I suppose

this is just a mapping out of how each individual client operates their

business. What makes a business unique, what kind of special operations

would our POS software than need in order to process/satisfy a business’s

unique requirements?)

2.1.8. Site Adaption Requirements

(If your software utilizes any website communication, detail the requirements… [“f.y.i. guys, I think the terms ‘requirement’ and

‘functionality’ are being thrown around within the SRS template guide without a solid understanding of what the terms actually mean. Whoever wrote this guide, I feel, to be curt, doesn’t really know what the hell they’re talking about. At least, that’s what I’m interpreting from all of this.”] …for any data initialization sequences that are specific to given site, mission, or operational mode… [“What the actual hell?”] Also, specify the site or mission-related features that should be modified to

adapt the software to a particular instillation.)

* 1. Product Functions

(Provide a summary of the major functions that the software will perform…allocate particular functions to a software product. 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. Textual or graphic methods CAN (**not mandatory**) be used to show the different function and their relationships. Such a diagram is not intended to show a design of a product but simply shows the logical relationships among variables. [“I think we have this already guys”])

* 1. User Characteristics

(Describe the characteristics of the intended users of the product including educational level, experience, and technical expertise. Do not state specific requirements [here we go again with this word being thrown around. What exactly does ‘requirements’ mean in this context?] but rather provide the reasons why certain requirements are later specified in later sections…]

* 1. Constraints

(Any other items that will limit the developer’s options. The list of constraints may be expanded upon at any time, and relevant examples of such constraints may include.)

* Regulatory policies
* Hardware limitations [for example ‘signal-timing’ constraints (how do we measure this?)]
* Limitations on interfacing to other applications
* Limitations on parallel operation
* Limitations on audit functions
* Limitations on control functions
* Limitations on the utilization of higher-order languages
* The limitations of signal handshake protocols [(?) I take it this is for web communication… for example XON-XOFF, ACK-NACK)]
* Reliability (?) limitations [for what? What would be unreliable?]
* Criticality (?) of the application [what actually are you talking about?]
* Safety and security considerations [okay, an actual intelligible example]
  1. Assumptions and Dependencies

(List each of the factors that affect the requirements stated in the SRS. These factors are not congruent to the design constraints on the software that were previously explored, but are, rather, any changes to the constraints (?) [is this what ‘them’ is referring to?] that can affect the requirements in the SRS.

Considering the degree of access to different affiliate software or hardware components that are not provided by your software you assume your clients have available to them, you must detail each corresponding prediction/assumption that you believe correlates to your expectations for each client. [“If that makes any sense, guys”])

* 1. Apportioning of Requirements

Identify any features and their requirements that may be delayed until future versions of your software).

**3.** **Specific Requirements**

This section is to explicitly detail the requirements that the POS-Plus software needs to operate. This information is made available to the advantage of our software designers and testers as criteria to measure the correctness of our company’s final product and to the advantage of our users, operators, and external systems so that they may have access to an exhaustive list of the features that POS-Plus provides and then the operational requirements of those features. This section will, at a minimum, include the range of data POS-Plus considers valid input, and then the system’s output response to those inputs. The sequence of functions that constitute each output response will also be enumerated and detailed.

3.1. External Interfaces

(All the inputs into and out of the software system)

**Item 1:**

* 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 input/output
* Screen formats/organization
* Data formats
* Command formats
* End messages

3.2. Functions

(Fundamental actions that must take place in software in accepting and processing the inputs and in processing and generating outputs… “The system shall” … partition the functional requirements into sub-functions and sub-processes)

* Validity check on the inputs
* Exact sequence of operations
* Responses to abnormal situations (e.g. overflow, communication \*facilities\*, error handling and recovery)
* Effect of parameters
* Relationship of outputs to inputs (e.g. input/output sequences, formulas for input to output conversion)
  + 1. List of Requirements

(Has a list subsystem functions/big functions and corresponding requirements)

* + 1. Use Cases

(Has an actor-goal list and use-case diagrams for each actor-goal pair, which will model the functionality of the software. The use cases that exist between each actor-goal pair will be enumerated and described)

* + 1. Domain Model Diagram

(No description available. Who knows what this might be…)

* 1. Performance Requirements

(Define the number of terminals to be supported, the number of simultaneous users supported, and the amount and type of information to be handled [at the terminals (?)]—static/unchanging conditions

The number of transactions and tasks and the amount of data to be process within certain time periods (for both normal and peak workload conditions)—dynamic/changing conditions

All of these conditions should be described in measurable terms (e.g. ‘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.)

* 1. Logical Database Requirements

(Logical requirements (?) [maybe he is referring to database logic or organization (e.g. choice of foreign keys, forms, etc.)] for any information that is placed into the database.

This may include types of information used by various functions, frequency of use (of what?), accessing capabilities, data entities and their relationships, integrity constraints, data retention requirements, etc.)

* 1. Design Constraints

(Specify the design constraints that can be imposed by other standards, hardware limitations, etc.)

* + 1. Requirements of Standards Compliance

(Specify the requirements derived from existing standards or regulations. They might include requirements of report formats, requirements of data naming, requirements of accounting procedures, requirements of audit tracing. Basically, I guess, the restrictions that our software places in items that constitute a specific standard.)

* 1. Software System Attributes

(Demands placed on the aspects of our software’s attributes that constitute requirements in their own right. And then we have, demands on the availability of our software attributes, demands on the security of our software attributes, demands on the maintainability of our software attributes, and demands on the portability of our software attributes.)

* 1. Organizing Specific Requirements

(This section is dedicated to the details of the requirements of our software, which are to be extensive. That’s why this section should break each aspect of those requirements into subcategories as to accomplish the task of effectively explaining them divisibly.

Apparently, there are classes of systems [“ha, yet another term that is thrown around arbitrarily and without mindfulness. I wonder what the author really meant to say when he was using ‘systems’ all those many times. I guess we may never know…”] that lend themselves go different organizations in the requirements section… [“whatever that means…I bet the author doesn’t even know….”]

Anyways… the subcategories are User Class, Objects, Feature, Stimulus, Response, Functional Hierarchy. Since we can’t even interpret what the author wants us to do for this section, I vouch that we avoid implementing it in the final version of this SRS document.)

* 1. Additional Comments

(What actually, maybe this section is for the instructor’s additional comments. It is unclear what he expects us to actually do here. He goes on to mention some nebulous notions of organization of our documents and of “requirements” [“biggest buzzword of 2018”] and then other archetypes for organization, etc. However, like I said, what he expects us to do with this section is unclear. We might choose to interpret this section as a place for us to express any of our own additional comments for the “client” for our own “software engineers” and “software developers”, or any else that may look to this document for more clarification of things and more detail on complicated/nebulous topics. Or we can always omit this section [lol].)

1. **Project Changes Management Process**

(This is a place for our logs to go (the logs that detail expansions that we make to our project, changes to the SRS document, and logs alternative ideas/viewpoints for our project’s design that we chose not to follow up on but still need to document.)

1. **Document Approvals**

(Who the hell approves our document? Us? Trivial section, perhaps? Waste of time? Probably?)

1. **Supporting Information**

(Section for our table of contents, index, and appendices (appendices include sample I/O formats, descriptions of cost analysis studies, and results of user survey; supporting background information that can help the readers interpret the SRS; a description of the problems to be solved by the software; special packaging instructions for the code and the media to meet security, export, initial loading, or other requirements.)

Also, when appendices are included, the SRS should explicitly state whether the appendices are to be considered part of the requirements or not.)