Software requirement specification document for project ————

Ahmed Mohamed,....
Supervised by: Dr. ......

October 24, 2019

1 Introduction

1.1 Purpose of this document

Describes the purpose of the document, and the intended audience.

1.2 Scope of this document

Describes the scope of this requirements definition effort. Introduces the requirements elicitation team, including users, customers, system engineers, and developers. This section also details any constraints that were placed upon the requirements elicitation process, such as schedules, costs, or the software engineering environment used to develop requirements.

1.3 Overview

Provides a brief overview of the product defined as a result of the requirements elicitation process.

1.4 Business Context

Provides an overview of the business organization sponsoring the development of this product. This overview should include the business's mission statement and its organizational objectives or goals.

2 General Description

2.1 Product Functions

Describes the general functionality of the product, which will be discussed in more detail below.

2.2 Similar System Information

Describes the relationship of this product with any other products. Specifies if this product is intended to be stand-alone, or else used as a component of a larger product. If the latter, this section discusses the relationship of this product to the larger product. This is how you can [1] a document.

2.3 User Characteristics

Describes the features of the user community, including their expected expertise with software systems and the application domain.

2.4 User Problem Statement

This section describes the essential problem(s) currently confronted by the user community.

2.5 User Objectives

This section describes the set of objectives and requirements for the system from the user's perspective. It may include a ”wish list” of desirable characteristics, along with more feasible solutions that are in line with the business objectives.

2.6 General Constraints

Lists general constraints placed upon the design team, including speed requirements, industry protocols, hardware platforms, and so forth.

3 Functional Requirements

This section lists the functional requirements in ranked order. Functional requirements describes the possible effects of a software system, in other words, what the system must accomplish. Other kinds of requirements (such as interface requirements, performance requirements, or reliability requirements) describe how the system accomplishes its functional requirements see table 1. Each functional requirement should be specified in a format similar to the following:

Table 1: Functional Requirement XYZ

Function Name	Short, imperative sentence stating highest ranked functional requirement.
Description	A full description of the requirement.
Critically	Describes how essential this requirement is to the overall system.
Technical issues	Describes any design or implementation issues involved in satisfying this requirement.
Cost and schedule	Describes the relative or absolute costs associated with this issue.
Risks	Describes the circumstances under which this requirement might not able to be satisfied, and what actions can be taken to reduce the probability of this occurrence.
Dependencies with other requirements	Describes interactions with other requirements.
Pre-Condition	The state of the system before the running of the function
Post-Condition	The state of the system after the run of the function

4 Interface Requirements

This section describes how the software interfaces with other software products or users for input or output. Examples of such interfaces include library routines, token streams, shared memory, data streams, and so forth.

4.1 User Interfaces

Use some software for primitive plan of your project. Describes how this product interfaces with the user.

4.1.1 GUI

Describes the graphical user interface if present. This section should include a set of screen dumps or mock-ups to illustrate user interface features. If the system is menu-driven, a description of all menus and their components should be provided.

4.1.2 CLI

Describes the command-line interface if present. For each command, a description of all arguments and example values and invocations should be provided.

4.1.3 API

Describes the application programming interface, if present. For each public interface function, the name, arguments, return values, examples of invocation, and interactions with other functions should be provided.

4.1.4 Diagnostics or ROM

Describes how to obtain debugging information or other diagnostic data.

4.2 Hardware Interfaces

Describes interfaces to hardware devices.

4.3 Communications Interfaces

Describes network interfaces.

4.4 Software Interfaces

Describes any remaining software interfaces not included above.

5 Performance Requirements

Specifies speed and memory requirements.

6 Design Constraints

Specifies any constraints for the design team using this document.

6.1 Standards Compliance

6.2 Hardware Limitations

6.3 others as appropriate

7 Other non-functional attributes

Specifies any other particular non-functional attributes required by the system. Examples are provided below.

7.1 Security

7.2 Binary Compatibility

7.3 Reliability

7.4 Maintainability

7.5 Portability

7.6 Extensibility

7.7 Re-usability

7.8 Application Affinity/Compatibility

7.9 Resource Utilization

7.10 Serviceability

7.11 others as appropriate

8 Preliminary Object-Oriented Domain Analysis

This section presents a list of the fundamental objects that must be modeled within the system to satisfy its requirements. The purpose is to provide an alternative, "structural" view on the requirements stated above and how they might be satisfied in the system. A primitive class diagram to be delivered.

8.1 Inheritance Relationships

This section should contain a set of graphs that illustrate the primary inheritance hierarchy (is-kind-of) for the system. For example:

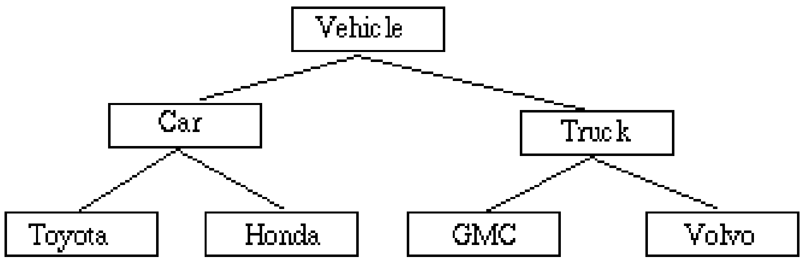

Figure 1: Inheritance Relations

8.2 Class descriptions

This section presents a more detailed description of each class identified during the OO Domain Analysis. For more details on the process giving rise to these descriptions, see Lecture 5.3: OO Domain Analysis and/or texts on object-oriented software development. Each class description should conform to the following structure:

8.2.1 Class name

Abstract or Concrete: Indicates whether this class is abstract or concrete.

8.2.2 List of Superclasses:

Names all immediate superclasses.

8.2.3 List of Subclasses:

Names all immediate subclasses.

8.2.4 Purpose:

States the basic purpose of the class.

8.2.5 Collaborations:

Names each class with which this class must interact in order to accomplish its purpose, and how.

8.2.6 Attributes:

Lists each attribute (state variable) associated with each instance of this class, and indicates examples of possible values (or a range).

8.2.7 Operations

: Lists each operation that can be invoked upon instances of this class. For each operation, the arguments (and their type), the return value (and its type), and any side effects of the operation should be specified.

8.2.8 Constraints:

Lists any restrictions upon the general state or behavior of instances of this class.

9 Operational Scenarios

This section should describe a set of scenarios that illustrate, from the user's perspective, what will be experienced when utilizing the system under various situations. In the article Inquiry-Based Requirements Analysis (IEEE Software, March 1994), scenarios are defined as follows: In the broad sense, a scenario is simply a proposed specific use of the system. More specifically, a scenario is a description of one or more end-to-end transactions involving the required system and its environment. Scenarios can be documented in different ways, depending up on the level of detail needed. The simplest form is a use case, which consists merely of a short description with a number attached. More detailed forms are called scripts. These are usually represented as tables or diagrams and involved identifying an action and the agent (doer) of the action. For this reason, a script can also be called an action table. Although scenarios are useful in acquiring and validating requirements, they are not themselves requirements, because the describe the system's behavior only in specific situations; a specification, on the other hand, describes what the system should do in general.

10 Preliminary Schedule Adjusted

This section provides an initial version of the project plan, including the major tasks to be accomplished, their interdependence's, and their tentative start/stop dates. The plan also includes information on hardware, software, and resource requirements. The project plan should be accompanied by one or more PERT or GANTT charts.

11 Preliminary Budget Adjusted

This section provides an initial budget for the project, itemized by cost factor.

12 Appendices

Specifies other useful information for understanding the requirements. All SRS documents should include at least the following two appendices:

12.1 Definitions, Acronyms, Abbreviations

Provides definitions of unfamiliar definitions, terms, and acronyms.

12.2 Collected material

13 References

References

[1] M. Rehm, N. Bee, and E. Andre, "Wave like an egyptian: accelerometer based gesture recognition for culture specific interactions," in *Proceedings of the 22nd British HCI Group Annual Conference on People and Computers: Culture, Creativity, Interaction - Volume 1*, ser. BCS-HCI '08. Swinton, UK, UK: British Computer Society, 2008, pp. 13–22.