Nappy, the ingenious
Software Requirements Specification

Version 1.0

Revision History

Date	Version	Description	Author
12/10/15	1.0	Erste Version des Dokuments	Mehmet Ali Incekara, Marc Mahler, Marvin Zerulla

Table of Contents

Software Requirements Specification

1. Introduction

Das Projekt "Nappy, the ingenious" hat das Ziel eine Desktop-Applikation zu entwickeln. Es ist eine Art Quizspiel, in dem der Actor / der Spieler gegen den Computer (gegen Nappy) spielen muss. Es wird zwischen zwei Spielmodis unterschieden: Im ersten Spielmodus denkt der Actor an eine Figur aus der Fernsehserie „The Simpsons". Nappy versucht anhand von Fragen, welche der Actor mit ja, nein oder „ich weiß nicht" beantwortet, die Figur zu erraten. Im zweiten Spielmodus denkt Nappy an eine Figur und der Spieler muss anhand vorgegebener Fragen die Figur erraten. Derjeniger, der weniger Fragen benötigt, gewinnt das Duell.

1.1 Purpose

Das SRS soll einen Überblick über die Anforderungen an unser Projekt bieten. Außerdem soll es als Richtlinie für die Entwicklung des Spiels gelten.

1.2 Scope

Dokument zur internen Nutzung -----

1.3 Definitions, Acronyms, and Abbreviations

SRS	Software Requirements Specification
Actor	Spieler
H2	Eine Datenbank

1.4 References

Blog: https://nappydevelopment.wordpress.com/

GitHub: https://github.com/nappydevelopment/Nappy-the-ingenious

UCD: to be determined (tbd)

1.5 Overview

Im Folgenden werden das Projekt und die für es vorgesehenen Vorgaben erklärt.

2. Overall Description

*[This section of the **SRS** describes 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 detail in Section 3, and makes them easier to understand. Include such items as:*

- *product perspective*
- *product functions*
- *user characteristics*
- *constraints*
- *assumptions and dependencies*
- *requirements subsets]*

3. Specific Requirements

*[This section of the **SRS** contains all software requirements to 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. When using use-case modeling, these requirements are captured in the Use Cases and the applicable supplementary specifications. If use-case modeling is not used, the outline for supplementary specifications may be inserted directly into this section, as shown below.]*

3.1 Functionality

*[This section describes the functional requirements of the system for those requirements that are expressed in the natural language style. For many applications, this may constitute the bulk of the **SRS** package and thought should be given to the organization of this section. This section is typically organized by feature, but alternative organization methods may also be appropriate; for example, organization by user or organization by subsystem. Functional requirements may include feature sets, capabilities, and security.*

Where application development tools, such as requirements tools, modeling tools, and the like, are employed to capture the functionality, this section of the document would refer to the availability of that data, indicating the location and name of the tool used to capture the data.]

3.1.1 <Functional Requirement One>

[The requirement description.]

3.2 Usability

[This section includes all those requirements that affect usability. For example,

- *specify the required training time for a normal users and a power user to become productive at particular operations*
- *specify measurable task times for typical tasks or base the new system's usability requirements on other systems that the users know and like*
- *specify requirement to conform to common usability standards, such as IBM's CUA standards Microsoft's GUI standards]*

3.2.1 <Usability Requirement One>

[The requirement description goes here.]

3.3 Reliability

[Requirements for reliability of the system should be specified here. Some suggestions follow:

- *Availability—specify the percentage of time available (xx.xx%), hours of use, maintenance access, degraded mode operations, and so on.*
- *Mean Time Between Failures (MTBF) — this is usually specified in hours, but it could also be specified in terms of days, months or years.*
- *Mean Time To Repair (MTTR)—how long is the system allowed to be out of operation after it has failed?*
- *Accuracy—specifies precision (resolution) and accuracy (by some known standard) that is required in the system's output.*
- *Maximum Bugs or Defect Rate—usually expressed in terms of bugs per thousand lines of code (bugs/KLOC) or bugs per function-point(bugs/function-point).*
- *Bugs or Defect Rate—categorized in terms of minor, significant, and critical bugs: the requirement(s) must define what is meant by a "critical" bug; for example, complete loss of data*

or a complete inability to use certain parts of the system's functionality.]

3.3.1 <Reliability Requirement One>

[The requirement description.]

3.4 Performance

[The system's performance characteristics are outlined in this section. Include specific response times. Where applicable, reference related Use Cases by name.

- *Response time for a transaction (average, maximum)*
- *Throughput, for example, transactions per second*
- *Capacity, for example, the number of customers or transactions the system can accommodate*
- *Degradation modes (what is the acceptable mode of operation when the system has been degraded in some manner)*
- *Resource utilization, such as memory, disk, communications, and so forth.*

3.4.1 <Performance Requirement One>

[The requirement description goes here.]

3.5 Supportability

[This section indicates any requirements that will enhance the supportability or maintainability of the system being built, including coding standards, naming conventions, class libraries, maintenance access, and maintenance utilities.]

3.5.1 <Supportability Requirement One>

[The requirement description goes here.]

3.6 Design Constraints

[This section indicates any design constraints on the system being built. Design constraints represent design decisions that have been mandated and must be adhered to. Examples include software languages, software process requirements, prescribed use of developmental tools, architectural and design constraints, purchased components, class libraries, and so on.]

3.6.1 <Design Constraint One>

[The requirement description goes here.]

3.7 On-line User Documentation and Help System Requirements

[Describes the requirements, if any, for o-line user documentation, help systems, help about notices, and so forth.]

3.8 Purchased Components

[This section describes any purchased components to be used with the system, any applicable licensing or usage restrictions, and any associated compatibility and interoperability or interface standards.]

3.9 Interfaces

[This section defines the interfaces that must be supported by the application. It should contain adequate specificity, protocols, ports and logical addresses, and the like, so that the software can be developed and verified against the interface requirements.]

3.9.1 User Interfaces

[Describe the user interfaces that are to be implemented by the software.]

3.9.2 Hardware Interfaces

[This section defines any hardware interfaces that are to be supported by the software, including logical structure, physical addresses, expected behavior, and so on.]

3.9.3 Software Interfaces

*[This section describes software interfaces to other components of the software system. These may be purchased components, components reused from another application or components being developed for subsystems outside of the scope of this **SRS** but with which this software application must interact.]*

3.9.4 Communications Interfaces

[Describe any communications interfaces to other systems or devices such as local area networks, remote serial devices, and so forth.]

3.10 Licensing Requirements

[Defines any licensing enforcement requirements or other usage restriction requirements that are to be exhibited by the software.]

3.11 Legal, Copyright, and Other Notices

[This section describes any necessary legal disclaimers, warranties, copyright notices, patent notices, wordmark, trademark, or logo compliance issues for the software.]

3.12 Applicable Standards

[This section describes by reference any applicable standard and the specific sections of any such standards which apply to the system being described. For example, this could include legal, quality and regulatory standards, industry standards for usability, interoperability, internationalization, operating system compliance, and so forth.]

4. Supporting Information

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

- *Table of contents*
- *Index*
- *Appendices*

*These may include use-case storyboards or user-interface prototypes. When appendices are included, the **SRS** should explicitly state whether or not the appendices are to be considered part of the requirements.]*