+

**TEAM ANODYNE**

**SRS**

SOFTWARE REQUIREMENTS SPECIFICATION

**JANUARY 2014**

**PREPARED FOR**

Software Engineering

Module Code: MOD003263

**INSTRUCTOR**:

Dr. Cristina Luca

Bryant 110, Cambridge Campus

Cristina.luca@anglia.ac.uk

# REVISION HISTORY

|  |  |  |  |
| --- | --- | --- | --- |
| **DATE** | **DESCRIPTION** | **AUTHOR** | **COMMENTS** |
| October 2013 | 1.0 | Frances Obikaonu | Successful! |
| December 2013 | 1.1 | Thomas Finney | . |
| December 2013 | 1.2 | Harry Hall | Okay cool |
| January 2014 | 2.0 |  |  |

# DOCUMENT APPROVAL

The following Software Requirements Specification has been accepted and approved by the following:

|  |  |  |  |
| --- | --- | --- | --- |
| **SIGNATURE** | **PRINTED NAME** | **TITLE** | **DATE** |
|  | DR. CRISTINA LUCA | INSTUCTOR |  |
|  |  |  |  |
|  |  |  |  |

**TABLE OF CONTENTS**

Revision History ii

Document Approval ii

1. Introduction 1

1.1 Purpose 1

1.2 Scope 1

1.3 Definitions, Acronyms, and Abbreviations 1

1.4 References 1

1.5 Overview 1

2. General Description 2

2.1 Product Perspective 2

2.2 Product Functions 2

2.3 User Characteristics 2

2.4 General Constraints 2

2.5 Assumptions and Dependencies 2

3. Specific Requirements 2

3.1 External Interface Requirements 3

3.1.1 User Interfaces 3

3.1.2 Hardware Interfaces 3

3.1.3 Software Interfaces 3

3.1.4 Communications Interfaces 3

3.2 Functional Requirements 3

3.2.1 <Functional Requirement or Feature #1> 3

3.2.2 <Functional Requirement or Feature #2> 3

3.3 Use Cases 3

3.3.1 Use Case #1 3

3.3.2 Use Case #2 3

3.4 Classes / Objects 3

3.4.1 <Class / Object #1> 3

3.4.2 <Class / Object #2> 3

3.5 Non-Functional Requirements 4

3.5.1 Performance 4

3.5.2 Reliability 4

3.5.3 Availability 4

3.5.4 Security 4

3.5.5 Maintainability 4

3.5.6 Portability 4

3.6 Inverse Requirements 4

3.7 Design Constraints 4

3.8 Logical Database Requirements 4

3.9 Other Requirements 4

4. Analysis Models 4

4.1 Sequence Diagrams 5

4.3 Data Flow Diagrams (DFD) 5

4.2 State-Transition Diagrams (STD) 5

5. Change Management Process 5

A. Appendices 5

A.1 Appendix 1 5

A.2 Appendix 2 5

# 1. INTRODUCTION

This specification will demonstrate the requirements needed for the Over Surgery application. There will be included images and diagrams as well as associative descriptions which should dictate the way the requirements should be dealt with and implemented successfully.

## 1.1 PURPOSE

The aim of this document is to give a detailed description of the proposed system that has been designed to replace non-electronic system currently in place.

It will serve as a guideline for the engineers on what is required and how it should be implemented.

## 1.2 SCOPE

The software produced will be called Over Surgery System.

Over Surgery System will allow the user to:

1. Login into the system.
2. Check the GP or nurse’s availability on a specific day.
3. Check all the GPs’ and nurse’s on duty on a specific day.
4. Register a new patient by entering the patient’s details.
5. Find a patient that can provide either by their patient ID, name, date of birth or their address.
6. Book a patient for an appointment.
7. Change a patient’s appointment.
8. Cancel a patient’s appointment.
9. Extend the prescriptions for a patient
10. Only allow the prescriptions to be extended.
11. Check and print the results of the tests the patent did.

## 1.3 Definitions, Acronyms, and Abbreviations

|  |  |  |
| --- | --- | --- |
| **TERM** | **SECTION USED** | **MEANING** |
| GP | 1 | Software Requirement Specification |
|  |  |  |
|  |  |  |
|  |  |  |

## 1.4 REFERENCES

**[1]** Dr. Orest Pilskalns (WSU, Vancover) and Jack Hagemeister (WSU, Pullman: The SRS templates used as guides in developing the template for the WSU-TC Spring 2005 CptS 322 course that is used in this document

**[2]** Sections of this document are based upon the IEEE Guide to Software Requirements Specification (ANSI/IEEE Std. 830-1984).

## 1.5 OVERVIEW

The subsequent sections will give a description of the proposed system and the functions the product will perform. They will also contain the requirements of the system and the associated use cases as well as a class diagram. The final section will have the designed sequence diagrams which demonstrate how the system should work to complete the tasks as outlined.

# 2. GENERAL DESCRIPTION

## 2.1 Product Perspective

The system will allow the user to carry out a wide variety of tasks, which the user will use at the Over Surgery. The user uses a paper based system to carry out tasks such as booking appointments and canceling appointments and registering patients and many more. With each task been carried out using a paper system, this is very hard to manage and keep tack of, due to the size of the system. When the system uses a computer it will allow the user to carry out the exact same tasks but on an electronic system, which will be easier to use and manage. The product will also be used by itself and not run along side any other product. The function of the product will be discussed bellow in section 2.2 Product function.

## 2.2 Product Functions

The software will perform will perform a wide variety of tasks which will allow the user to carry out the same tasks that they do on the current paper based system. The system should be easier to use and quicker to use while at the same time allowing the user to keep all the functionality of the current system. The system will need to be able to:

* Handle large amounts of data
* Run without crashing
* Provide useful error messages
* Be upgradeable – Add new parts to the system
* Easy to maintain
* Scalable
* Perform quickly even when processing large amounts of data
* Have a simple easy to use GUI

To ensure that each part of the system meets the product function set out in this documentation. It will be tested to make sure that no problems arise. The product function will also be communicated to the user.

## 2.3 User Characteristics

The user is of a non-technical nature who wants to enter as little information into the system as possible. The user needs to know only about patient information and GP/nurse information to use the system effectively. The user needs aspect to all parts of the system with the expectation of that database itself, so the user will not require a high technical knowledge. The user is also slow at typing so minimum input by the user will be the best option of the system.

## 2.4 General Constraints

The computer that the system will be running on will be a standalone machine with an age of 3 years, so the system will have to use very few resources in order to function on this old computer. The system will only have to interface with other applications when printing patient prescriptions. The system will also be accessed by one user and must ask for a secure username and password to make sure that the database can only be accessed by the receptionist.

The user will have access to all patient and GP/nurse information but will have no control over the physical database therefore eliminating any threat of data been stolen by the user.

## 2.5 Assumptions and Dependencies

A number of factors that may affect the requirements specified in the SRS include:

* The operating system been changed through the project to OSX or a never Windows operating system. This intron may cause the application to crash or not work correctly, resulting in a loss of data.
* The user maybe changed which may affect how the system is intended to work as the user may have different requirements to the previous user, resulting in a change of the system which could affect the surgery due to system downtime.
* The SRS may change also if the hardware is out of date or not functioning correctly, such as faulty USB ports so not been able to connect a printer, resulting in the system not been able to print prescriptions.

# 3. SPECIFIC REQUIREMENTS

This will be the largest and most important section of the SRS. The customer requirements will be embodied within Section 2, but this section will give the D-requirements that are used to guide the project’s software design, implementation, and testing.

Each requirement in this section should be:

* Correct
* Traceable (both forward and backward to prior/future artifacts)
* Unambiguous
* Verifiable (i.e., testable)
* Prioritized (with respect to importance and/or stability)
* Complete
* Consistent
* Uniquely identifiable (usually via numbering like 3.4.5.6)

Attention should be paid to the carefuly organize the requirements presented in this section so that they may easily accessed and understood. Furthermore, this SRS is not the software design document, therefore one should avoid the tendency to over-constrain (and therefore design) the software project within this SRS.

## 3.1 External Interface Requirements

### 3.1.1 User Interfaces

### 3.1.2 Hardware Interfaces

### 3.1.3 Software Interfaces

### 3.1.4 Communications Interfaces

## 3.2 Functional Requirements

This section describes specific features of the software project. If desired, some requirements may be specified in the use-case format and listed in the Use Cases Section.

### 3.2.1 <Functional Requirement or Feature #1>

3.2.1.1 Introduction

3.2.1.2 Inputs

3.2.1.3 Processing

3.2.1.4 Outputs

3.2.1.5 Error Handling

### 3.2.2 <Functional Requirement or Feature #2>

…

## 3.3 Use Cases

### 3.3.1 Use Case Diagram



Figure 1: The use case diagram

The use case diagram displays all the scenarios the user would typically encounter when using the system. They also model the proposed requirements as the use cases are limited to what the system can actually do.

### 3.3.2 Use Case Descriptions

**Use case:** Login

**Actors:** Receptionist

**Goal:**  To log into the system

**Overview:**

When the user launches the application the system, the system should go to the login screen, requesting the user to enter their username and password. The system recognises the user’s details and allows them to login to their account.

**Typical course of events**

|  |  |
| --- | --- |
| **Actor action:** | **System response:** |
| 1. User open the login screen |  |
|  | 1. Display the login screen |
| 1. User enter their user and password and click login |  |
|  | 1. Check the user’s Username and password |
|  | 1. Logs the user on to their user account. |

**Alternative courses**

Step 2: The system encounters and error and doesn’t display the login screen correctly.

Step 4: The login details are not recognised by the system, the system doesn’t allow the user to login.

**Use case:** Check individual nurse/GP availability

**Actors:**  Receptionist

**Goal:**  To check individual nurse and GP availability on specific days

**Overview:**

When the user clicks the button ‘GP/nurse availability’, the system should then bring up the correct screen asking the user to select the GP or nurse name from a drop down list. Then ask the user what day they want to search if the GP or nurse is available on. The system should search for the GP or nurse straight away and display the correct information on the screen.

**Typical course of events**

|  |  |
| --- | --- |
| **Actor action:** | **System response:** |
| 1. User clicks GP/nurse availability |  |
|  | 1. Displays the input page for nurse/GP availability |
| 1. User selects nurse or GP’s last name from a drop down box |  |
| 1. Users selects the date they want to see the availability of GP or nurse |  |
|  | 1. Checks if all input fields have been filled in |
|  | 1. Checks if the GP or nurse is in the selected day |
|  | 1. Checks if the Dr surgery is open that day |
|  | 1. Displays the Nurse or GP availability details on screen |

**Alternative course**

Step 1: The system doesn’t respond when the GP/nurse availability button is clicked, this may terminate the application.

Step 3: The Dr or nurse might not be on the system, resulting in the user not been able to check their availability.

Step 8: The system encounters an error when displaying the availability of a GP or nurse, thus not allowing the user to see the GP or nurse availability.

**Use case:** GP/Nurse timetable

**Actors:**  Receptionist

**Goal:**  To check nurse and GP availability in the surgery

**Overview:**

When the user clicks the button ‘GP/nurse timetable’, the correct screen will be displayed showing the user the timetable for all GP and nurses who are in the surgery that week. The user should also be able to view the timetable for the following week.

**Typical course of events**

|  |  |
| --- | --- |
| **Actor action:** | **System response:** |
| 1. User clicks GP/nurse timetable |  |
|  | 1. Search all staff who are in the surgery that week |
|  | 1. Displays the timetable for all nurse and GPs in the surgery that week |
| 1. User clicks to view the next following week for the availability of GP and nurses |  |
|  | 1. Search all staff who are in the surgery the following week |
|  | 1. Displays the timetable for all nurse and GPs in the surgery for the following week |

**Alternative course**

Step 2, 5: The system doesn’t find any members of staff in for the current week or next week, thus not allowing the user to view the staff timetable.

Step 3, 6: The system encounters an error and doesn’t allow the user to view the timetable due to it not been displayed correctly.

**Use case:** Register patient

**Actors:**  Receptionist

**Goal:**  To add a new patient to the system

**Overview:**

When the user clicks the ‘register a new patient’ button, the system will display input fields and drop down menus for the patients details. The system will give the new patient a unique ID so the data can be saved. The system will then check if the user has entered all the details in every mandatory field before saving the data.

**Typical course of events**

|  |  |
| --- | --- |
| **Actor action:** | **System response:** |
| 1. The user clicks ‘register a new patient’ |  |
|  | 1. Displays the register patient screen and input boxes and drop down menu’s |
|  | 1. Give the patient a unique ID |
|  | 1. Asks user to enter the patient details in the correct boxes |
| 1. The user enters all the relevant patient details into the fields. |  |
| 1. The user clicks the save button to save the patients details |  |
|  | 1. Checks if any of the mandatory fields are empty, displays error message if they are |
|  | 1. Saves the details of the patient in the database |

**Alternative course**

Step 2: The system may not display all the input fields and drop down menus in the correct format resulting in the user not been able to enter the new patient details.

Step 3: The system may not give the user a unique ID if it encounters an error, which could lead to the patient details not been saved correctly.

Step 7: The system may not display an error message if any fields are empty which may result in important patient details not been saved correctly.

Step 8: The system may not save the patient details in the database, which will result in the patient not been on the system.

**Use case:** Search for patient

**Actors:**  Receptionist

**Goal:**  To search for a patient on the system

**Overview:**

When the user clicks ‘search for a patient’, the system will display the correct screen asking the user to enter the patients: last name, DOB and post code. The system will then search for the user based on these three details, and then display all the users’ details on the screen.

**Typical course of events**

|  |  |
| --- | --- |
| **Actor action:** | **System response:** |
| 1. User clicks ‘search for a patient’ |  |
|  | 1. Displays the search for a patient screen and the input fields for patient details |
| 1. The user enter the patients: last name, DOB, post code |  |
|  | 1. Checks if all details have been entered into the fields |
|  | 1. Searches for the patient in the database |
|  | 1. Displays the patient details on screen |

**Alternative course**

Step 2, 6: The system may not display the pages correctly resulting in the user not been able to enter the patient’s details or been able to view the patient’s details.

Step 3: The user may forget to enter the details or may not enter the details correctly which will lead to the system not been able to find the patient.

Step 5: The system may not be able to find the patient resulting in the user having to recheck the details that they have entered.

**Use case:** Extend the patient medication

**Actors:**  Receptionist <<extends>>

**Goal:**  Allow the receptionist to extend the patients medication

**Overview:**

The user will click ‘extend patients medication’, the system will display all the patients’ medication and when they finish their medication. The user can then extend a particular medication to a specific day, then save the new medication details into the system.

**Typical course of events**

|  |  |
| --- | --- |
| **Actor action:** | **System response:** |
| 1. User clicks ‘extend patient medication’ |  |
|  | 1. Displays the patients medication and length of time till the finish the medication |
| 1. User selects the medication they wish to extend |  |
| 1. User selects the amount of time they wish to extend the medication |  |
| 1. User saves the data by clicking the ‘save’ button |  |
|  | 1. Checks the appropriate fields have been filled in |
|  | 1. The data is saved to the database |

**Alternative course**

Step 2: The page may not be displayed correctly due to an error, which may not allow the user to see the information on the page.

Step 6: The system may encounter an error when checking all the fields have filled in, which could result in the medication not been extended for the patient or the patient been on the medication for too long.

Step 7: The data may not save on the system due to an error, resulting in the relevant medication extensions not been changed.

**Use case:** Print the results of patients test

**Actors:**  Receptionist <<extends>>

**Goal:**  To allow the receptionist to print the patients test results

**Overview:**

The user clicks the ‘print patient test results’, the test results for the patient are then displayed on the screen. The user can then select the test result they want to print, and then print the test result.

**Typical course of events**

|  |  |
| --- | --- |
| **Actor action:** | **System response:** |
| 1. User clicks ‘print patients test results’ |  |
|  | 1. Displays the patients test results on screen |
| 1. User selects the test results they wish to be printed |  |
| 1. User clicks ‘print’ |  |
|  | 1. The results are put into a standard printing format and then it prints the results |

**Alternative course**

Step 2: The page may not be displayed correctly due to an error, which may not allow the user to see the information on the page.

Step 5: The system may not print the page according to the page template which may the illegibility of the information.

**Use case:** Delete a patient

**Actors:**  Receptionist <<extends>>

**Goal:**  To allow the receptionist to delete a patient from the system

**Overview:**

The user selects ‘delete a patient’; all the patients’ details are then displayed on the screen, the user the clicks ‘delete a patient’, the system asks if the user is sure they want to delete the patient. The patient’s deleted details are then sorted in a separate database and will be deleted from the system after a certain amount of time.

**Typical course of events**

|  |  |
| --- | --- |
| **Actor action:** | **System response:** |
| 1. User clicks ‘delete a patient’ |  |
|  | 1. Displays all the patients details on screen |
| 1. The user clicks ‘delete the patient’ |  |
|  | 1. The system asks if the user is sure they want to delete a patient |
| 1. The user can click ‘yes’ or ‘cancel’ |  |
|  | 1. The patient’s details are then deleted from the system and stored in a separate part of the database for a certain amount of time. |

**Alternative course**

Step 2: The page may not be displayed correctly due to an error, which may not allow the user to see the information on the page.

Step 4: The system may not display this message due to an error, which may result in the patients been deleted from the system.

Step 6: The system may encounter an error when storing the deleted patient’s details in the system, and as a result the details may be permanently deleted from the system.

**Use case:** Change patient details

**Actors:**  Receptionist <<extends>>

**Goal:**  To allow the receptionist to change patients details in the system

**Overview:**

The user clicks ‘change patient details’, the system brings up the change details page. The user then can change any details of the patients, and then save the changed details. The system should overwrite the old patient’s details in the system with the new ones.

**Typical course of events**

|  |  |
| --- | --- |
| **Actor action:** | **System response:** |
| 1. User clicks ‘change patient details’ |  |
|  | 1. Displays all the patients details on screen |
| 1. The user edits the patients details |  |
| 1. The user clicks ‘save patient details’ |  |
|  | 1. Checks all the fields have been filled in |
|  | 1. The patients details are saved in the database, overwriting the old patient details in the system |

**Alternative course**

Step 2: The page may not be displayed correctly due to an error, which may not allow the user to see the information on the page.

Step 6: The system may encounter an error when storing the patient’s new details into the system, or may not overwrite the existing patient’s details resulting in the database not been correct and the patient having incorrect details.

**Use case:** Cancel patient appointment

**Actors:**  Receptionist <<extends>>

**Goal:**  To allow the receptionist to cancel patient appointments

**Overview:**

The user clicks ‘cancel patient appointment’, the system then displays the appointments that patient for that week. The user can the select the patients appoint they wish to cancel, then cancel it. The system should then update itself and display the cancelled appointment as now available.

**Typical course of events**

|  |  |
| --- | --- |
| **Actor action:** | **System response:** |
| 1. User clicks ‘cancel patient appointments’ |  |
|  | 1. Displays the patients appointments for that Week |
| 1. The user selects the appointment they wish to cancel |  |
| 1. The user then clicks ‘cancel appointment’ |  |
|  | 1. Asks if the user is sure they want to cancel the appointment |
| 1. The user can click ‘yes’ or ‘cancel’ |  |
|  | 1. Cancels the appointment from the system |
|  | 1. Displays the time as available on the ‘Booking page’ |

**Alternative course**

Step 2: The page may not display correctly resulting in the user not been able to view the page.

Step 3: The appointment may not be there, which would not allow the user to cancel the patient’s appointment.

Step 5: The system may not display this message due to an error, which may result in the patients appointment been cancelled.

Step 8: The system may not reallocate the deleted time back into the system, which in turn would not allow patients to book that appointment again.

**Use case:** Change appointment

**Actors:**  Receptionist <<extends>>

**Goal:**  To allow the receptionist to change patient appointment

**Overview:**

The user clicks ‘change patient appointment’, the system then displays the available appointments for the week. The user then selects the appointment they wish to change and a new appointment the patient wants. The user then saves the database. The system should delete the old time and replace it with the new time, then reallocate that old time back into the system.

**Typical course of events**

|  |  |
| --- | --- |
| **Actor action:** | **System response:** |
| 1. User clicks ‘change patients appointment’ |  |
|  | 1. Displays the patients available appointments for that week |
| 1. The user select what appointment they wish to change |  |
| 1. The user selects the new appointment the patient wishes to have |  |
| 1. The user selects the Dr/nurse the patient wants |  |
| 1. User clicks ‘save appointment’ |  |
|  | 1. Checks all the fields have been filled in |
|  | 1. Deletes the old appointment and save the new one |
|  | 1. Puts the old appointment time back on the ‘Booking page’ |

**Alternative course**

Step 2: The page may not display correctly resulting in the user not been able to view the page.

Step 8: The system may encounter an error when changing the patient’s appointment, this could result in the old patient’s time not been overwritten by the system.

Step 9: The system may not reallocate the changed time back into the system, which in turn would not allow patients to book that appointment again.

**Use case:** Book a patient

**Actors:**  Receptionist <<extends>>

**Goal:**  To allow the receptionist to book a patient in for an appointment

**Overview:**

The user clicks ‘book patient appointment’, the system then displays the available appointments for the week. The user then selects the date, time and Dr/ nurse the patient requires and clicks save appointment. The system should then save the appointment into the database and not allow that time, date and Dr/nurse to be booked again.

**Typical course of events**

|  |  |
| --- | --- |
| **Actor action:** | **System response:** |
| 1. User clicks ‘Book patient appointment’ |  |
|  | 1. Displays the patients available appointments for that week |
| 1. The user selects the Date and time the user desires |  |
| 1. The user selects the Dr/nurse the patient wants |  |
| 1. The user then clicks ‘save appointment’ |  |
|  | 1. Checks all the fields have been filled in |
|  | 1. Saves the appointment in the database |

**Alternative course**

Step 2: The page may not display correctly resulting in the user not been able to view the page.

Step 3, 4: The date and time or Dr or nurse may not be available when the patient wants them, which could mean the patient isn’t seen that day.

Step 7: The system may not save the patients appointment, resulting in the patient not been in the database and not been seen by the Dr or nurse. This will also allow that time to be rebooked by another patient.

## 3.4 Classes / Objects

### 3.4.1 Class Diagram



Figure 2: The class diagram

Figure 2 is the class diagram that has been designed. It has all of the appropriate attributes and the operations within each class.

The basic thought behind it was that there would be individual classes, like patient and staff, with base attributes that would be assigned a value when an object of that class of that is made. Next, there would be an operation that would pass the object to the database class which in turn would interact with the database as it seemed best to only have one class that was directly accessing the database.

There was also a sufficient attempt at modelling the forms that would be required, LoginScreen and MainScreen, though it appears the attempt was futile as it couldn’t be divined how many buttons or interactive things there would be or ow many forms there would actually be, hence the discrepancy (if the discrepancy can’t be seen, it will become apparent later).

### 3.4.2 Class List

Receptionist:

Attributes:

userName,

A private string that will hold the Receptionist’s username for logging in.

password

A private string that will hold the Receptionist’s password for logging in

Functions:

Login(),

The method that should call the database class and then will check if the supplied password and username match what is in the database and then either redirect the user to the main menu or display an error message.

LoginScreen:

Attributes:

Functions:

Button\_Login(),

The method that will be activated when the button on the form is clicked, the method will call the login method from the receptionist. Kind of a redundant operation in hindsight.

MainScreen:

Functions:

Button\_Search(),

Presumably the method that will somehow search for something.

Date\_Picker(),

A method which chooses a date for an unspecified reason.

Save\_Details\_Button(),

The method invoked when a the “Save\_Details” button is clicked. It should save some details.

SearchPrescription(),

The method which searches for a prescription. Whose prescription? A mystery.

## 3.5 Non-Functional Requirements

Non-functional requirements may exist for the following attributes. Often these requirements must be achieved at a system-wide level rather than at a unit level. State the requirements in the following sections in measurable terms (e.g., 95% of transaction shall be processed in less than a second, system downtime may not exceed 1 minute per day, > 30 day MTBF value, etc).

### 3.5.1 Performance

### 3.5.2 Reliability

### 3.5.3 Availability

### 3.5.4 Security

### 3.5.5 Maintainability

### 3.5.6 Portability

## 3.6 Inverse Requirements

State any \*useful\* inverse requirements.

## 3.7 Design Constraints

Specify design constrains imposed by other standards, company policies, hardware limitation, etc. that will impact this software project.

## 3.8 Logical Database Requirements

Will a database be used? If so, what logical requirements exist for data formats, storage capabilities, data retention, data integrity, etc.

## 3.9 Other Requirements

Catchall section for any additional requirements.

# 4. ANALYSIS MODELS

## 4.1 Sequence Diagrams

## 4.3 Data Flow Diagrams (DFD)

## 4.2 State-Transition Diagrams (STD)

# 5. Change Management Process

Identify and describe the process that will be used to update the SRS, as needed, when project scope or requirements change. Who can submit changes and by what means, and how will these changes be approved.

# A. Appendices

Appendices may be used to provide additional (and hopefully helpful) information. If present, the SRS should explicitly state whether the information contained within an appendix is to be considered as a part of the SRS’s overall set of requirements.

*Example Appendices could include (initial) conceptual documents for the software project, marketing materials, minutes of meetings with the customer(s), etc.*

## A.1 Appendix 1

## A.2 Appendix 2