NATIONAL INFORMATICS CENTRE SERVICES INC. (NICSI)
(A GOVERNMENT OF INDIA ENTERPRISE UNDER NIC)
MINISTRY OF ELECTRONICS & INFORMATION TECHNOLOGY

Software Requirements Specification Template

In coherence with Standard 830-1998 - IEEE Recommended Practice for Software Requirements Specifications

NICSI-PBD-SRS Template 830-1998 -13082019-V1.0

Product Business Division NICSI

PRODUCT BUSINESS DIVISION
NATIONAL INFORMATICS CENTRE NICSI
SERVICES INCORPORATED

Date	**Document Version**	**Description**	**Author**
June,2019	0.1	First Draft	Product Business Division NICSI
June,2019	0.2	Subsection details updated & Example given for each section for easy understanding.	Product Business Division NICSI
June, 2019	0.3	1. Appendix section updated. 2. Use Case specifications template updated.	Product Business Division NICSI

Table of Contents

1 Introduction

The following subsections of the Software Requirements Specifications (SRS) document should provide an overview of the entire SRS. The thing to keep in mind as you write this document is that you are telling what the system must do – so that designers can ultimately build it.

1.1 Purpose

This subsection should:

a) Delineate the purpose of the SRS
b) Specify the intended audience for the SRS

1.2 Scope

This subsection should:

a) Identify the software product(s) to be produced by name *(e.g. host DBMS, report generator, etc.)*
b) Explain what the software product(s) will, and, if necessary, will not do
c) Describe the application of the software being specified, including relevant benefits, objectives, and goals
d) Be consistent with similar statements in higher-level specifications *(e.g. the system specification)*, if they exist.

1.3 Definitions and Acronyms

This subsection should provide the definitions of all terms and acronyms required for properly interpreting the SRS. This information may be provided by reference to one or more appendixes in the SRS or by reference to other documents.

Example:

Term	***Definition***
OS	*Operating System*
GUI	*Graphical User Interface*
SRS	*Software Requirement Specification*
GPS	*Global Positioning System*
IEEE	*Institute of Electrical And Electronic Engineers*
User	*Anyone who uses the application.*
IDE	*Integrated Development Environment*
SDK	*Standard development kit*
DBMS	*Data Base Management System*
API	*Application Processing Interface*

1.4 References

This subsection should

a) Provide a complete list of all documents referenced elsewhere in the SRS
b) Identify each document by title, report number (if applicable), date, and publishing organization
c) Specify the sources from which the references can be obtained.

This information may be provided by reference to an appendix or to another document.

Reference	Version
Document Being referred	Version of the document

Example

IEEE. IEEE Std. 830-1998 IEEE Recommended Practice for Software Requirements Specifications. IEEE Computer Society, 1998.

1.5 Overview

This subsection should

a) Describe what the rest of the SRS contains
b) Explain how the SRS is organized

Example

This document is prepared to explain all detailed information about overall system description, functional, non-functional and specific requirements, data and behavioural model description of the system. This document basically consists of three parts: the first part includes introduction and overall description of the application/system/product/software/etc.

The second part contains specific requirements, data and behavioural model description of the system which are section 3, 4 and 5 in the document. Last part gives planning, conclusion and supporting information about the system

2 Overall Description

This section of the SRS should describe 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 of the SRS, and makes them easier to understand.

This section usually consists of six subsections, as follows:

a) Product perspective (Section 2.1)
b) Product functions (Section 2.2)
c) User characteristics (Section 2.3)
d) Constraints (Section 2.4)
e) Assumptions and dependencies (Section 2.5)
f) Apportioning of requirements (Section 2.6)

2.1 Product Perspective

This subsection of the SRS should put the product into perspective with other related products. If the product is independent and totally self-contained, it should be so stated here. If the SRS defines a product that is a component of a larger system, as frequently occurs, then this subsection should relate the requirements of that larger system to functionality of the software and should identify interfaces between that system and the software.

A block diagram showing the major components of the larger system, interconnections, and external interfaces can be helpful. This is not a design or architecture picture. It is more to provide context, especially if your system will interact with external actors. The system you are building should be shown as a black box. Let the design document present the internals.

This subsection should also describe how the software operates inside various constraints. For example, these constraints could include

a) System interfaces (section 2.1.1)
b) User interfaces (section 2.1.2)
c) Hardware interfaces (section 2.1.3)
d) Software interfaces (section 2.1.4)
e) Communications interfaces (section 2.1.5)
f) Memory (section 2.1.6)
g) Operations (section 2.1.7)
h) Site adaptation requirements (section 2.1.8)

2.1.1 System Interfaces

This should list each system interface and identify the functionality of the software to accomplish the system requirement and the interface description to match the system. These are external systems that you have to interact with.

Example:

For instance, if you are building a business application that interfaces with the existing employee payroll system, what is the API to that system that designer's will need to use?

The application will be implemented in java and it will use Android SDK, Facebook SDK and Google maps SDK. Eclipse will be used as IDE while implementing. MYSQL workbench will also be used as DBMS. Database of the system will be temporarily held in local servers.

2.1.2 User Interfaces

This should specify the following:

a) The logical characteristics of each interface between the software product and its users. This includes those configuration characteristics *(e.g., required screen formats, page or window layouts, content of any reports or menus, or availability of programmable function keys)* necessary to accomplish the software requirements.
b) All the aspects of optimizing the interface with the person who must use the system. This may simply comprise a list of do's and don'ts on how the system will appear to the user.
Note: special interface requirements may be addressed in this section *(e.g. Special Screen Reader facility for users with visual disability)*

2.1.3 Hardware Interfaces

This should specify the logical characteristics of each interface between the software product and the hardware components of the system. This includes configuration characteristics *(e.g. number of ports, instruction sets, etc.)*. It also covers such matters as what devices are to be supported, how they are to be supported, and protocols. This section is for detailing the actual hardware devices your application will interact with and control. *(For instance, if you are controlling a hardware device, what is the interface to that device?)*

2.1.4 Software Interfaces

This should specify the use of other required software products *(e.g., a data management system, an operating system, or a mathematical package)*, and interfaces with other application systems *(e.g., the linkage between an accounts receivable system and a general ledger system)*. For each required software product, the following should be provided:

a) Name (Software interface used identified by its name)
b) Specification number (Software id Specification)
c) Version number (version of the software interface used)
d) Source (source from where we can get the same)

For each interface, the following should be provided:

a) Discussion of the purpose of the interfacing software as related to this software product.
b) Definition of the interface in terms of message content and format. It is not necessary to detail any well-documented interface, but a reference to the document defining the interface is required.

Note: This subsection can be skipped if the interaction with other application is not required.

2.1.5 Communications Interfaces

This should specify the various interfaces to communications. *(e.g., Local network protocols, HTTP, TCP, etc.)*

Note: if you happen to use web services to your application then do not list it here.

2.1.6 Memory Constraints

This should specify any applicable characteristics and limits on primary and secondary memory. *(e.g., system shall use no more than 50 Mb of storage and 100 MB of RAM.)*

2.1.7 Operations

This should specify the normal and special operations required by the user such as:

a) The various modes of operations in the user organization *(e.g. user-initiated operations)*
b) Periods of interactive operations and periods of unattended operations
c) Data processing support functions
d) Backup and recovery operations

Note: This is sometimes specific as part of the User Interfaces (Section 2.1.2). If this separate from User Interface Section, then cover business process type stuff that would impact the design.

For instance,
If the company brings all their systems down at midnight for data backup that might impact the design. These are all the work tasks that impact the design of an application, but which might not be located in software.

2.1.8 Site Adaptation Requirement

This should

a) Define the requirements for any data or initialization sequences that are specific to a given site, mission, or operational mode *(e.g., grid values, safety limits, etc.)*

b) Specify the site or mission-related features that should be modified to adapt the software to a particular installation.

Example

If any modifications to the customer's work area would be required by your system, then document that here. For instance, "A 100Kw backup generator and 10000 BTU air conditioning systems must be installed at the user site prior to software installation".

This could also be software-specific like, "New data tables created for this system must be installed on the company's existing DB server and populated prior to system activation." Any equipment the customer would need to buy or any software setup that needs to be done so that your system will install and operate correctly should be documented here.

2.2 Product Functions

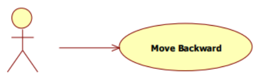
This subsection of the SRS should provide a summary of the major functions that the software will perform. *(e.g., an SRS for an accounting program may use this part to address customer account maintenance, customer statement, and invoice preparation without mentioning the vast amount of detail that each of those functions requires.).*

Sometimes the function summary that is necessary for this part can be taken directly from the section of the higher-level specification (if one exists) that allocates particular functions to the software product. Note that for the sake of clarity

a) 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.
b) Textual or graphical methods can be used to show the different functions and their relationships. Such a diagram is not intended to show a design of a product, but simply shows the logical relationships among variables*. (e.g. Use Case diagrams and description shown below)*

Use Case ID.	**Use Case Name**	**Diagram**	**Brief Description**
This section will contain unique id of the use case	The name of the use case. It usually starts with a verb	Section will be a diagram of the use case	Summarizes the use case in a short paragraph.

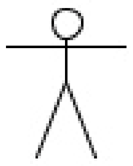
Illustration

Use Case ID.	Use Case Name	Diagram	Brief Description
UC001	Example: Move Backward		This function becomes activate when the user press "backward key" button on the keyboard. This function makes the user move backward.

2.3 User Characteristics

This subsection of the SRS should describe those general characteristics of the intended users of the product including educational level, experience, and technical expertise. It should not be used to state specific requirements, but rather should provide the reasons why certain specific requirements are later specified in Section 3 of the SRS.

For the use case approach, this section will contain the list of the actors that perform the use cases along with their characteristics.
List Use Case Actors

Actor ID	Actor/Roll Name	Role Description and objectives
Unique ID number assigned to various actors	Actor Name 	Briefly describe the role of the actor to the system and how the actor will use the system What is the actor's goal? What does the actor need from the system? What is the expected outcome from the system?

Illustrations

Actor ID	Actor/Roll Name	Role Description and objectives
AC001	Prospect	A prospect of our organization, someone who is interested in becoming a Publisher
AC002	Publisher	A customer who is publishing a magazine or website
AC003	Advertising Manager	The people who sell advertising for the Publisher

2.4 Constraints

This subsection of the SRS should provide a general description of any other items that will limit the developer's options. These include

a) Regulatory policies *(e.g. GIGW for website)*
b) Hardware limitations *(e.g., signal timing requirements)*
c) Interfaces to other applications *(e.g. API or XML data sharing)*

d) Parallel operation *(e.g. what are the operations that need to be supported for parallel operations without breaking the integrity of the data stored.)*
e) Audit functions *(e.g. Ability to choose and filter the user's desired dataset)*
f) Control functions
g) Higher-order language requirements *(e.g. Java shall be the implementation language with Android, Facebook, Google SDK's.)*
h) Signal handshake protocols *(e.g., XON-XOFF, ACK-NACK)*
i) Reliability requirements *(e.g. shall operate on Android v4.0 or higher operating system)*
j) Criticality of the application *(e.g. How critical the application is and what kind of availability & security measures need to be taken care of should be stated e.g. Aadhaar database is highly critical)*
k) Safety and security considerations *(e.g. Information of the users shall be encrypted before transferred to database in order to maintain the security of the crucial information about users.)*

2.5 Assumptions and Dependencies

This subsection of the SRS should list each of the factors that affect the requirements stated in the SRS. These factors are not design constraints on the software but are, rather, any changes to them that can affect the requirements in the SRS.

Example

An assumption may be that a specific operating system will be available on the hardware designated for the software product. If, in fact, the operating system is not available, the SRS would then have to change accordingly.

2.6 Apportioning of Requirements

This subsection of the SRS should identify requirements that may be delayed until future versions of the system. This section divides the requirements into different sections for development and delivery

Use Case ID	Use Case Name	Primary Actor	Priority
Use Case identified by its unique id	The Name of the Use Case. It Usually Starts With a Verb	Mention the name of the primary actor who is responsible for the use case	priority can be identified as 1,2,3,4,5….etc 1 being the top most priority.

Illustration

Use Case ID	Use Case Name	Primary Actor	Priority
UC001	Place a Bid	Buyer	1
UC004	Ship an Item	Seller	2
UC182	Add a Warehouse	Admin	-

3 Specific Requirements

This section of the SRS should contain all of the 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. Throughout this section, every stated requirement should be externally perceivable by users, operators, or other external systems. These requirements should include at a minimum, a description of every input (stimulus) into the system, every output (response) from the system, and all functions performed by the system in response to an input or in support of an output. As this is often the largest and most important part of the SRS, the following principles apply:

a) Specific requirements should be stated in conformance with all the characteristics given below:
    a. Correct
    b. Unambiguous
    c. Complete
    d. Consistent
    e. Ranked for importance and/or stability
    f. Verifiable
    g. Modifiable
    h. Traceable
b) Specific requirements should be cross-referenced to earlier documents that relate.
c) All requirements should be uniquely identifiable.
d) Careful attention should be given to organizing the requirements to maximize readability.

Before examining specific ways of organizing the requirements it is helpful to understand the various items that comprise requirements as described in Section 3.1 through Section 3.7

3.1 External interfaces Requirements

This should be a detailed description of all inputs into and outputs from the software system. It should complement the interface descriptions in Section 2 and should not repeat information there.

This subsection should also describe how the software operates inside various constraints. For example, these constraints could include

a) User interface requirements (section 3.1.1.)
b) Hardware interface requirements (section 3.1.2)
c) Software interface requirements (section 3.1.3)
d) Communications interface requirements (section 3.1.4)

3.1.1 User Interface requirements

This should specify the following:

c) The logical characteristics of each interface between the software product and its users. This includes those configuration characteristics *(e.g., required screen formats, page or window layouts, content of any reports or menus, or availability of programmable function keys)* necessary to accomplish the software requirements.
d) All the aspects of optimizing the interface with the person who must use the system. This may simply comprise a list of do's and don'ts on how the system will appear to the user.
Note: special interface requirements may be addressed in this section *(e.g. Special Screen Reader facility for users with visual disability)*

3.1.2 Hardware Interface requirements

This should specify the logical characteristics of each interface between the software product and the hardware components of the system. This includes configuration characteristics *(e.g. number of ports, instruction sets, etc.)*. It also covers such matters as what devices are to be supported, how they are to be supported, and protocols. This section is for detailing the actual hardware devices your application will interact with and control. *(For instance, if you are controlling a hardware device, what is the interface to that device?)*

3.1.3 Software Interface requirements

This should specify the use of other required software products *(e.g., a data management system, an operating system, or a mathematical package)*, and interfaces with other application systems *(e.g., the linkage between an accounts receivable system and a general ledger system)*. For each required software product, the following should be provided:

e) Name (Software interface used identified by its name)
f) Specification number (Software id Specification)
g) Version number (version of the software interface used)
h) Source (source from where we can get the same)

For each interface, the following should be provided:

c) Discussion of the purpose of the interfacing software as related to this software product.
d) Definition of the interface in terms of message content and format. It is not necessary to detail any well-documented interface, but a reference to the document defining the interface is required.

Note: This subsection can be skipped if the interaction with other application is not required.

3.1.4 Communications Interface requirements

This should specify the various interfaces to communications. *(e.g., Local network protocols, HTTP, TCP, etc.)*

Note: if you happen to use web services to your application then do not list it here.

3.2 Functional Requirement

Functional requirements should define the fundamental actions that must take place in the software in accepting and processing the inputs and generating the outputs.
Requirements can be static or dynamic as per the user operations, and organized on the basis of use cases. It can also be mentioned in use cases narratives based on the product functions in sec. 2.2

3.2.1 Use Case Description

Use Case Specification Template.

Use Case ID	Use Case identified by its unique id		
Use Case Name	The Name of the Use Case. It Usually Starts with a Verb.		
Created By	Initial Author Name	**Last Updated By:**	Updated By
Date Created	Creation Date	**Date Last Updated:**	Updating Date
Use Case Description	This section describes the use case in a few sentences that summarizes the interaction between the actor and the system.		
Actors	Primary: The actor which initiates the use case. Secondary: Other actors the system relies on to accomplish the services of the use case.		

Triggers	Any Stimuli or action by the actor that initiates the use case. A use case may be triggered by one or more than one actor.
Preconditions	What Should be true before the use case is initiated.
Basic Flow	This section gives a textual description of a use case's sequence of actions that has been identified as part of the normal flow. Generally, simple deviations will be defined "inline" within the normal flow itself. **Steps(Numbered)** **Flow of Events** Step 1 Step 2 Step n
Alternative Flow	If there are major deviations from the normal flow of events, then they are described here as alternate flows. **RFS - A Reference flow step number where flow branches from.** **Steps(Numbered)** **Flow of Events** Step 1 Step 2 Step n
Post-condition	This is a description of the state of the system after the successful completion of the use case
Business Rules/Validations	Any Business rules that need to be incorporated as part of the use case at the time of implementation.
Exceptions	This section Describes all error conditions that can arise in the use case. Only exceptions that are unique to the use case are detailed here.
Relationships	This section defines the relationships that the use case has with other use cases. The relationships have been defined in terms of three types: **Extends**-This relationship means that the use case that is being described extends the functionality of the use cases that are listed under this heading. Invariably, the extended functionality is optional and the user may or may not use it. **Is Extended By**- Under this heading are listed the use cases that extend the functionality of this use case, implying that these use cases may be optionally executed while executing the current use case. **Uses**- This relationship means that the use case being described always

	uses the use case listed under this heading to complete its functionality.
Special Requirement	Any special hardware or software requirement for use case initiation.
Assumptions	Any assumptions that are made about the environment such as hardware, software, network connectivity, user capabilities, etc. in which the software is expected to operate and which are necessary for the successful execution of the use case

3.3 Performance Requirements

This subsection should specify both the static and the dynamic numerical requirements placed on the software or on human interaction with the software as a whole. Static numerical requirements may include the following:

a) The number of terminals to be supported
b) The number of simultaneous users to be supported
c) Amount and type of information to be handled

Static numerical requirements are sometimes identified under a separate section entitled Capacity

Dynamic numerical requirements may include, for example, the numbers of transactions and tasks and the amount of data to be processed within certain time periods for both normal and peak workload conditions.

Note: All of these requirements should be stated in measurable terms.

Example

95% of the transactions shall be processed in less than 1 s.
rather than,
An operator shall not have to wait for the transaction to complete.

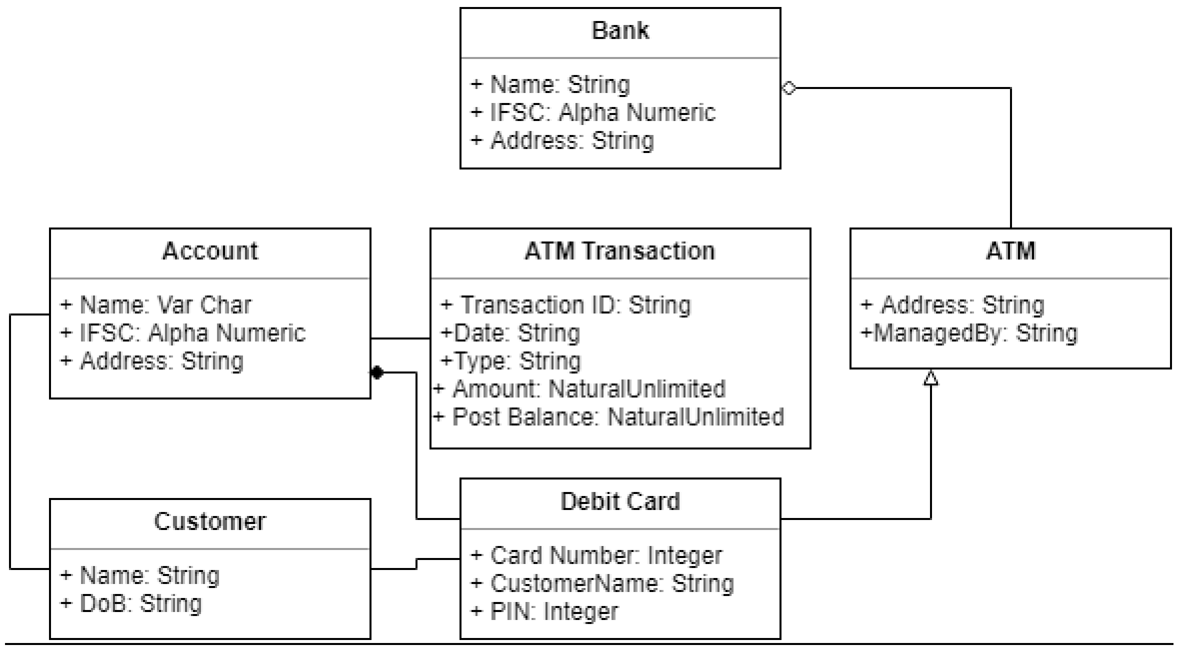
NOTE: Numerical limits applied to one specific function are normally specified as part of the processing subparagraph description of that function.

3.4 Logical Database Requirements

This should specify the logical requirements for any information that is to be placed into a database. This may include the following:

a) Types of information used by various functions *(e.g. integer, varchar. etc.)*
b) Frequency of use *(e.g. Stored information is used how many time- for posts, messages, check-in information etc.)*
c) Accessing capabilities
d) Data entities and their relationships*(e.g. ER Diagram given below shows the relationship)*
e) Integrity constraints: Integrity constraints ensure that the data insertion, updating, and other processes have to be performed in such a way that data integrity is not affected, so should the diagram represent
f) Data Retention Requirement: Describe the length of time the data must be retained *(e.g. Information about an application for naturalization shall be retained in immediately accessible form for 3 years after receipt of the application.)*

Example



3.5 Design Constraints

This should specify design constraints that can be imposed by other standards, hardware limitations, etc.

3.5.1 Standards Compliance

This subsection should specify the requirements derived from existing standards or regulations. They may include the following

a) Report format
b) Data naming convention
c) Accounting procedures
d) Audit tracing

Example
This could specify the requirement for software to trace processing activity. Such traces are needed for some applications to meet minimum regulatory or financial standards. An audit trace requirement may, for example, state that all changes to a payroll database must be recorded in a trace file with before and after values.

3.6 Software System Attributes

There are a number of attributes of software that can serve as requirements. It is important that required attributes be specified so that their achievement can be objectively verified. Sub-clauses 3.6.1 through 3.6.5 provide a partial list of examples.

3.6.1 Reliability

This should specify the factors required to establish the required reliability of the software system at time of delivery

Example

- *This system should keep the database information consistently.*
- *The application part of the system should never fail. In the database side, failures should be minimal and there should be crash recovery systems in order not to lose information in a potential database failure.*
- *System should display informative messages when its components don't work properly.*

3.6.2 Availability

This should specify the factors required to guarantee a defined availability level for the entire system such as checkpoint, recovery, and restart.

Example

- *System should be available for 7 days and 24 hours.*
- *In the application side, system should be tested against probable failures before publishing the first version or updated versions of application. Published version should be error free*
- *In database side, in case of a failure, system should recover any information for user and system.*

3.6.3 Security

This should specify the factors that protect the software from accidental or malicious access, use, modification, destruction, or disclosure. Specific requirements in this area could include the need to

a) Utilize certain crypto-graphical techniques
b) Keep specific log or history data sets
c) Assign certain functions to different modules
d) Restrict communications between some areas of the program
e) Check data integrity for critical variables

Example

- *The system must not request unnecessary permissions from the user in order to prevent unwanted attacks.*
- *Stored data of the application should not be reached by other applications that are installed in the user's mobile device.*
- *Stored data in the mobile device and sent data via internet should be encrypted. Sent and received data should be transferred via HTTPS connection. And also authenticated and encrypted socket-level communication should be implemented.*

3.6.4 Maintainability

This should specify attributes of software that relate to the ease of maintenance of the software itself. There may be some requirement for certain modularity, interfaces, complexity, etc. Requirements should not be placed here just because they are thought to be good design practices.

Example

- *A SVN software should be used in development phase in order to reduce complexity, make the system traceable and recover the code from an unwanted crash while more than one developer are dealing with the code.*
- *Design elements should be documented well.*

- *Since programming language is object-oriented, program tasks are independent of each other and therefore easier to maintain.*
- *All parts of the code should be easy to read.*

3.6.5 Portability

This should specify attributes of software that relate to the ease of porting the software to other host machines and/or operating systems. This may include the following:

a) Percentage of components with host-dependent code
b) Percentage of code that is host dependent
c) Use of a proven portable language
d) Use of a particular compiler or language subset
e) Use of a particular operating system

Example

- *Since the program is for Android devices, all of the Android devices which meet the requirements explained in 2nd section of this document can operate the application.*
- *Application won't work on any OS except Android.*
- *%100 percent of the program depends on host. In order to change the host, all the components of database should be transferred.*

3.7 Business Rules

In this section list all the operating principles about the product, such as which individuals or roles can perform which functions under specific circumstances. These are not functional requirements in themselves, but they may imply certain functional requirements to enforce the rules.

Example

In an Hospital Management system

- *A Patient can stay in one or many Rooms during a stay in the Hospital.*
- *Each Patient Bill can have one or many Bill Items.*

4 Appendix A: Index

This section includes list of words or phrases ('headings') and associated pointers ('locators') where useful material relating to that heading can be found in a document.

5 Appendix B: Analysis Models

This section can be used optionally, it include any pertinent analysis models, such as data flow diagrams, class diagrams, state-transition diagrams, or entity-relationship diagrams.

6 Appendix D: To Be Determined List

Collect a numbered list of the TBD (to be determined) references that remain in the SRS so they can be tracked to closure.

7 Appendix E: Other Information

This Section may include:

a) Sample I/O formats, descriptions of cost analysis studies, results of user surveys
b) Supporting or background information that can help the readers of the SRS
c) A description of the problems to be solved by the software
d) Special packaging instructions for the code and the media to meet security, export, initial loading, or other requirements