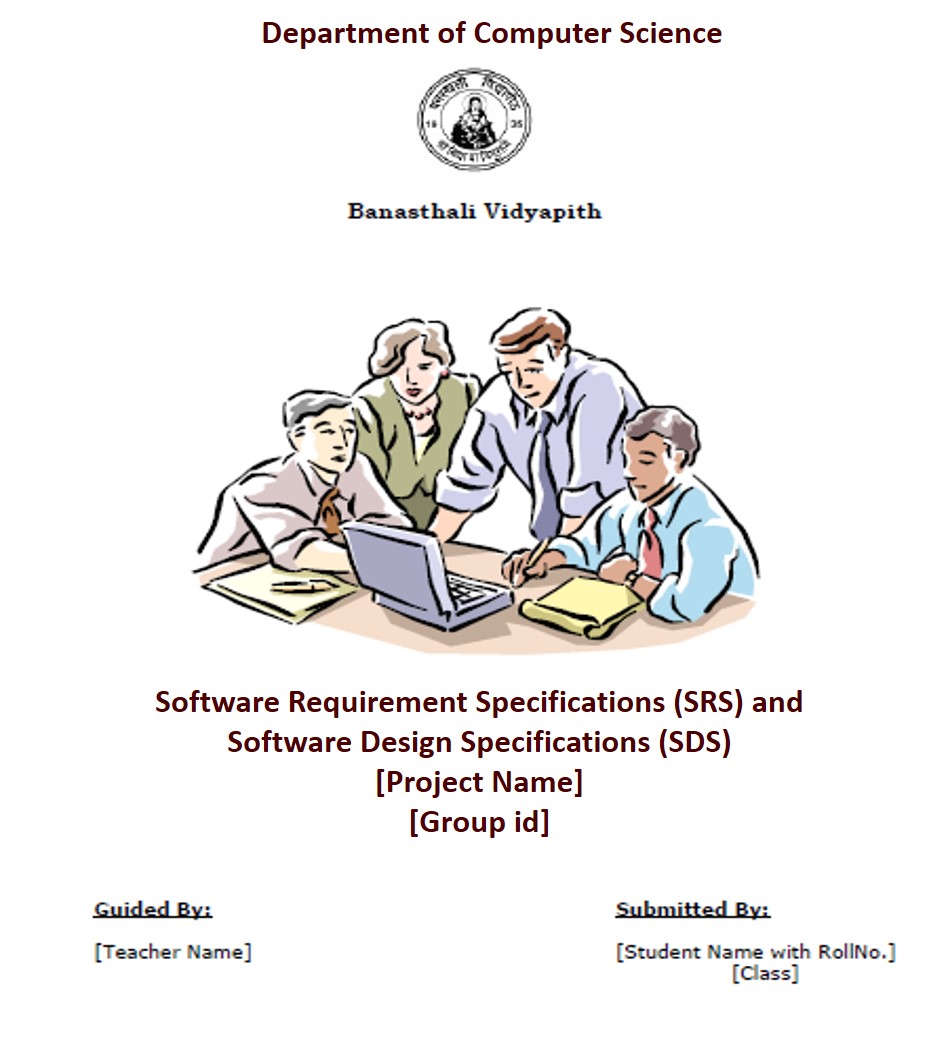
**

**Chapter 1 SRS**

**TABLE OF CONTENTS**

**1**. **Introduction**

1.1 Purpose …………………………………………………… Page No

1.2 Scope……………………………………………………… Page No

1.3 Definitions, Acronyms and Abbreviations……………….. Page No

1.4 Overview………………………………………………….. Page No

**2**. **General Description**

2.1 Product Perspective……………………………………….. Page No

2.1.1 Product Function …………………………………. Page No

2.1.2 Hardware Interface………………………………… Page No

2.1.3 Software Interface…………………………………..Page No

2.1.4 Communication Interface…………………………...Page No

2.2 User Characteristics…………………………………………Page No

2.3 General Constraints………………………………………. ...Page No

2.4 Technologies used………………………………………....... Page No

**3. Specific Requirements**

3.1 Functional Requirements……………………………………..Page No

3.2 Non-Functional Requirements ………………………………..Page No

3.2.1 AVAILABILITY……………………………………..Page No

3.2.2 SECURITY…………………………………………...Page No

3.2.3 RELIABILITY……….………………………………..Page No

3.2.4 PORTABILITY………………………………………. Page No

3.2.5 MAINTAINABILITY……………………………….... Page No

# *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. Do not use this document for design!!!*

## *1.1 Purpose*

*Identify the purpose of this SRS and its intended audience. In this subsection, describe the purpose of the particular SRS and specify the intended audience for the SRS.*

## *1.2 Scope*

*In this subsection:*

1. *Identify the software product(s) to be produced by name*
2. *Explain what the software product(s) will, and, if necessary, will not do*
3. *Describe the application of the software being specified, including relevant benefits, objectives, and goals*
4. *Be consistent with similar statements in higher-level specifications if they exist*

*This should be an executive-level summary. Do not enumerate the whole requirements list here.*

## *1.3 Definitions, Acronyms, and Abbreviations.*

*Provide the definitions of all terms, acronyms, and abbreviations required to properly interpret the SRS. This information may be provided by reference to one or more appendices in the SRS or by reference to documents. This information may be provided by reference to an Appendix.*

## *1.4 Overview*

*In this subsection:*

1. *Describe what the rest of the SRS contains*
2. *Explain how the SRS is organized*

*Don’t rehash the table of contents here. Point people to the parts of the document they are most concerned with. Customers/potential users care about section 2, developers care about section 3.*

# *2. The Overall Description*

*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 section 3, and makes them easier to understand. In a sense, this section tells the requirements in plain English for the consumption of the customer. Section3 will contain a specification written for the developers.*

## *2.1 Product Perspective*

*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 relates the requirements of the larger system to functionality of the software and identifies interfaces between that system and the software. If you are building a real system, compare its similarity and differences to other systems in the marketplace. If you are doing a research-oriented project, what related research compares to the system you are planning to build.*

*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.*

*The following subsections describe how the software operates inside various constraints.*

### *2.1.1 Product Function*

*Provide a summary of the major functions that the software will perform. 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.*

*For clarity:*

1. *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.*
2. *Textual or graphic 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.*

*AH, finally the real meat of section 2. This describes the functionality of the system in the language of the customer. What specifically does the system that will be designed have to do? Drawings are good, but remember this is a description of what the system needs to do, not how you are going to build it. (That comes in the design document).*

### 

### *2.1.2 Hardware Interfaces*

*Specify the logical characteristics of each interface between the software product and the hardware components of the system. This includes configuration characteristics. It also covers such matters as what devices are to be supported, how they are to be supported and protocols. This is not a description of hardware requirements in the sense that “This program must run on a Mac with 64M of RAM”. This section is for detailing the actual hardware devices your application will interact with and control. For instance, if you are controlling X10 type home devices, what is the interface to those devices? Designers should be able to look at this and know what hardware they need to worry about in the design. Many business type applications will have no hardware interfaces. If none, just state “The system has no hardware interface requirements” If you just delete sections that are not applicable, then readers do not know if: a. this does not apply or b. you forgot to include the section in the first place.*

### 

### *2.1.3 Software Interfaces*

*Specify the use of other required software products and interfaces with other application systems. For each required software product, include:*

1. *Name*
2. *Mnemonic*
3. *Specification number*
4. *Version number*
5. *Source*

*For each interface, provide:*

1. *Discussion of the purpose of the interfacing software as related to this software product*
2. *Definition of the interface in terms of message content and format*

*Here we document the APIs, versions of software that we do not have to write, but that our system has to use. For instance if your customer uses SQL Server 7 and you are required to use that, then you need to specify i.e.*

*2.1.4.1 Microsoft SQL Server 7. The system must use SQL Server as its database component. Communication with the DB is through ODBC connections. The system must provide SQL data table definitions to be provided to the company DBA for setup.*

*A key point to remember is that you do NOT want to specify software here that you think would be good to use. This is only for* ***customer-specified systems*** *that you* ***have*** *to interact with. Choosing SQL Server 7 as a DB without a customer requirement is a Design choice, not a requirement. This is a subtle but important point to writing good requirements and not over-constraining the design.*

### *2.1.4 Communications Interfaces*

*Specify the various interfaces to communications such as local network protocols, etc. These are protocols you will need to directly interact with. If you happen to use web services transparently to your application then do not list it here. If you are using a custom protocol to communicate between systems, then document that protocol here so designers know what to design. If it is a standard protocol, you can reference an existing document or RFC.*

## *2.2 User Characteristics*

*Describe those general characteristics of the intended users of the product including educational level, experience, and technical expertise. Do not state specific requirements but rather provide the reasons why certain specific requirements are later specified in section 3.*

*What is it about your potential user base that will impact the design? Their experience and comfort with technology will drive UI design. Other characteristics might actually influence internal design of the system.*

## *2.3 Constraints*

*Provide a general description of any other items that will limit the developer's options. These can include:*

*(1) Regulatory policies*

*(2) Hardware limitations (for example, signal timing requirements)*

*(3) Interface to other applications*

*(4) Parallel operation*

*(5) Audit functions*

*(6) Control functions*

*(7) Higher-order language requirements*

1. *Signal handshake protocols (for example, XON-XOFF, ACK-NACK)*
2. *Reliability requirements*

*(10) Criticality of the application*

*(11) Safety and security considerations*

*This section captures non-functional requirements in the customers language. A more formal presentation of these will occur in section 3.*

**2.4** **Technologies Used**

# *3. Specific Requirements*

*This section contains all the software requirements at 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. The following principles apply:*

1. *Specific requirements should be stated with all the characteristics of a good SRS*
2. *correct*
3. *unambiguous*
4. *complete*
5. *consistent*
6. *ranked for importance and/or stability*
7. *verifiable*
8. *modifiable*
9. *traceable*
10. *Specific requirements should be cross-referenced to earlier documents that relate*
11. *All requirements should be uniquely identifiable (usually via numbering like 3.1.2.3)*
12. *Careful attention should be given to organizing the requirements to maximize readability (Several alternative organizations are given at end of document)*

*Before examining specific ways of organizing the requirements it is helpful to understand the various items that comprise requirements as described in the following subclasses. This section reiterates section 2, but is for developers not the customer. The customer buys in with section 2, the designers use section 3 to design and build the actual application.*

*Remember this is not design. Do not require specific software packages, etc unless the customer specifically requires them. Avoid over-constraining your design. Use proper terminology:*

*The system shall… A required, must have feature*

*The system should… A desired feature, but may be deferred till later*

*The system may… An optional, nice-to-have feature that may never make it to implementation.*

*Each requirement should be uniquely identified for traceability. Usually, they are numbered 3.1, 3.1.1, 3.1.2.1 etc. Each requirement should also be testable. Avoid imprecise statements like, “The system shall be easy to use” Well no kidding, what does that mean? Avoid “motherhood and apple pie” type statements, “The system shall be developed using good software engineering practice”*

*Avoid examples, This is a specification, a designer should be able to read this spec and build the system without bothering the customer again. Don’t say things like, “The system shall accept configuration information such as name and address.” The designer doesn’t know if that is the only two data elements or if there are 200. List every piece of information that is required so the designers can build the right UI and data tables.*

## *3.1 Functional*

*Functional requirements define the fundamental actions that must take place in the software in accepting and processing the inputs and in processing and generating the outputs. These are generally listed as “shall” statements starting with "The system shall…*

*These include:*

1. *Validity checks on the inputs*
2. *Exact sequence of operations*
3. *Responses to abnormal situation, including*
4. *Overflow*
5. *Communication facilities*
6. *Error handling and recovery*
7. *Effect of parameters*
8. *Relationship of outputs to inputs, including*
9. *Input/output sequences*
10. *Formulas for input to output conversion*

*It may be appropriate to partition the functional requirements into sub-functions or sub-processes. This does not imply that the software design will also be partitioned that way.*

* 1. ***Non functional Requirements(Software System Attributes)***

***3.2.1 AVAILABILITY***

*The availability of this web-site is up to the Internet connection of the client. Since this is client-server related web-site shall be attainable all the time. User should have an account to enter the system; if user does not have an account then user can only see the information which will be displayed on the homepage of the web-site.*

***3.2.2******SECURITY***

*The authorization mechanism of the system will block the unwanted attempts to the server and also let the system decide on which privileges may the user have. The system has different types of users so there are different levels of authorization.*

***3.2.3 RELIABILITY***

*A backup file is maintained so that in case of system crash, the data will not be affected.*

***3.2.4******PORTABILITY***

*The system is developed using ASP.Net which provides a framework for developing web based applications.*

***3.2.5******MAINTAINABILITY***

*This website will follow the modular structure so it will be easy to maintain.*

***Chapter 2 SDS***

**TABLE OF CONTENTS**

1. **Introduction**

1.1 Purpose ……………………………………………………Page No

1.2 Scope………………………………………………………Page No

1.3 Terms, Definitions, Acronyms and Abbreviations………..Page No

1.4 Overview of Document ……..……………………………..Page No

2. **System Architectural Design**

2.1 High Level Design Overview…………………………….. Page No

2.2) Detailed Description of Components

2.2.1 Structure Chart …………………………………... Page No

2.2.2 Data Flow Diagram………………………………………. Page No

3. **Data Design**

3.1 Database Description…………………………………Page No

3.2 E-R Diagram…………………………. ………………… Page No

4 **User Interface Design**

4.1Detailed Description of Components……………………… Page No

4.2Screen images

**5. TYPES OF TESTS (With Implementation)**................................................. Page No

**6. Appendices**

## 7. References

# *1. Introduction*

*The following subsections of the Software Design Specifications (SDS) document should provide an overview of the entire SDS. 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. Do not use this document for design!!!*

## *1.1 Purpose*

*Identify the purpose of this SDS and its intended audience. In this subsection, describe the purpose of the particular SDS and specify the intended audience for the SDS.*

## *1.2 Scope*

*In this subsection:*

1. *Identify the software product(s) to be produced by name*
2. *Explain what the software product(s) will, and, if necessary, will not do*
3. *Describe the application of the software being specified, including relevant benefits, objectives, and goals*
4. *Be consistent with similar statements in higher-level specifications if they exist*

*This should be an executive-level summary. Do not enumerate the whole requirements list here.*

## *1.3 Definitions, Acronyms, and Abbreviations.*

*Provide the definitions of all terms, acronyms, and abbreviations required to properly interpret the SDS. This information may be provided by reference to one or more appendices in the SDS or by reference to documents. This information may be provided by reference to an Appendix.*

## *1.4 Overview*

*In this subsection:*

1. *Describe what the rest of the SDS contains*
2. *Explain how the SDS is organized*

***2. System Architectural Design***

*NOTE: This section is the main focus of the high-level conceptual design. The reader should come away with a good view of exactly how your solution is to be organized.*

***2.1 High-level Design Overview***

*Introduce the various components and systems at a high conceptual level. A*

## *Pictorial representation of the system architecture is presented.*

SQL Server

Web Server

(IIS with ASP.NET run time)

INTERNET

User

1-TIER

3-TIER

2-TIER

***2.2) Detailed Description of Components***

*NOTE: This section is the main focus of the technical design portion of the SDS, the detailed design. This section will provide most of the basis for implementing the product.*

***Description for Component n***

*A detailed description of each software component contained within the architecture is presented. A description of each major software function along with:*

***Examples:-***

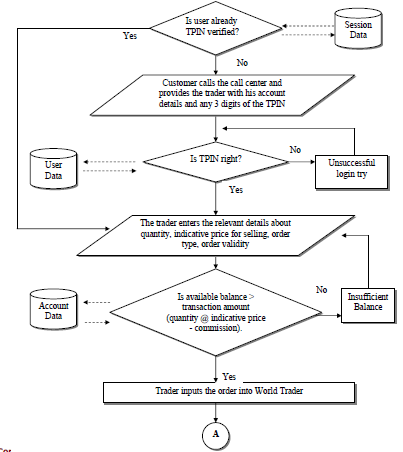
***Data flow diagram (DFD) and Structure chart, Use Case diagram, Entity relationship diagram, Class diagram, Activity diagram and Sequence chart.***

***Component and interface diagram***

*A detailed description of the input and output interfaces for the component is presented.*

***Component and processing detail***

*A detailed algorithmic description for each component is presented. A diagram showing the flow of information through the function and the transformation it undergoes is presented. For Example –*

**

***3 Structure and relationships***

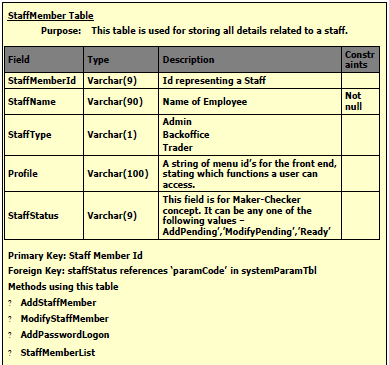
*This subsection will make clear the interrelationships and dependencies between modules/layers/components. Structure charts can be useful here. Another good idea is to use a simple finite state machine that demonstrates the operation of the product you are designing. There should always be explanatory text to help the reader understand any charts.*

***3.1) Data design***

*A description of all data structures including internal, global, and temporary data structures.*

***Temporary data structure***

*Files created for interim use are described.*

**

***3.2 Database description***

*Database(s) created as part of the application is (are) described. E-R Model should clearly specify the data entities and relationship between them. After normalization, each table must be described with purpose, methods using it, all its fields description, their data type, and integrity constraints.*

**

***4) User Interface Design***

*A description of the user interface design of the software is presented.*

***4.1 Description of the user interface***

*A detailed description of user interface including screen images or prototype is presented.*

***4.2 Screen images***

*Representation of the interface form from the user's point of view.*

***4.3 Objects and actions***

*All screen objects and actions are identified. Describe each object’s purpose, data type, properties and responses for different events.*

*For Example –*

***The Execution Order Function***

******

***5. TYPES OF TESTS (With Implementation)***

***Unit Tests***

*Unit tests are most commonly done by developers on their own machines or on a common server that is very volatile. It is not necessary that the unit test machines be the same platform and operating system as the target deployment environment, but the movement from the unit test environment to other testing environments should not require material code changes by developers. A plan for one machine per developer plus one small server should be included in the overall system architecture.*

***System Tests***

*The system test environment allows multiple modules to be connected together and executed as in a typical use-case scenario. The choice as to whether this is done on a separate machine from unit testing is up to the implementation and test team. If the target deployment environment is different from the unit test environment, the system test environment should contain a machine that matches the target environment. Although the system test machine need not match the size of the deployment box, it should have the same platform and operating system. A good rule of thumb is to prepare to add one more box for system tests of a smaller size, but the same operating system as the target environment. Again, this will be a relatively volatile environment, so it should not be viewed as a place to do industrial-strength testing by a large team.*

***Integration or Regression Tests***

*To perform integration and regression tests, it is advisable to have a separate environment that is similar to the target environment. Generally, one server will be enough at this point. However, the contents of this server should be strictly controlled. Either the test coordinator or his or her designate should make all software changes to this environment. Stability and auditability are essential to ensuring the accuracy of test results. Plan for at least one more servers at this stage in testing.*

***Stress Tests***

*Stress tests should be done in an environment identical to the target hosting environment. In the*

*first development cycle, this can be done in the production site, before the cutover to production.*

*For subsequent development cycles, a separate environment will have to be maintained for stress*

*testing.*

*Plan to replicate the deployment environment as part of the test bed for at least the second development cycle of the site. It has also been observed that the most common problem after performance in a high stress environment is database deadlock due to improper programming.*

*Deadlocks are typically difficult to detect and fix and may not show up until the site is highly stressed in production. So it is important that these conditions be stringently tested during the*

*stress test phase.*

***Acceptance Tests and Staging***

*Acceptance tests are generally performed in the same environment as the stress tests, so additional hardware is not needed to support this phase of testing. Again, during the initial development cycle, the production environment can be used to perform both acceptance testing and staging, but a new environment should be created for subsequent development cycles.*

***6. Appendices***

## *7. References*

*In this subsection:*

*(1) Provide a complete list of all documents referenced elsewhere in the SRS*

*(2) Identify each document by title, report number (if applicable), date, and publishing organization*

1. *Specify the sources from which the references can be obtained.*

*This information can be provided by reference to an appendix or to another document. If your application uses specific protocols or RFC’s, then reference them here so designers know where to find them.*