**Software Requirements Specification**



**Approvals Signature Block**

|  |  |  |
| --- | --- | --- |
| **Project Responsibility** | **Signature** | **Date** |
| *Project Guide (Internal)* |  |  |
| *Project Guide (External)* |  |  |
| *Documentation Leader* |  |  |

*The document in this file is adapted from the IEEE standards for Software Project Requirements Specifications, 830-1998, which conforms to the requirements of ISO standard 12207 Software Life Cycle Processes.*

*Items that are intended to stay in as part of your document are in* **bold***; blue italic text is used for explanatory information that should be removed when the template is used.*

**Table of Contents**

[**1.**](#_heading=h.gjdgxs) **INTRODUCTION 4**

[1.1](#_heading=h.30j0zll) Purpose 4

[1.2](#_heading=h.1fob9te) Scope 4

[1.3](#_heading=h.3znysh7) Definitions, acronyms, and abbreviations 4

[1.4](#_heading=h.2et92p0) References 4

[1.5](#_heading=h.tyjcwt) Overview 4

[**2.**](#_heading=h.3dy6vkm) **OVERALL DESCRIPTION 5**

[Problem Statement 5](#_heading=h.1t3h5sf)

[2.1](#_heading=h.4d34og8) Product Perspective 5

[Product Position Statement 5](#_heading=h.3l18frh)

[2.1.1](#_heading=h.17dp8vu) System Interfaces 5

[2.1.2](#_heading=h.206ipza) User Interfaces 6

[2.1.3](#_heading=h.4k668n3) Hardware Interfaces 6

[2.1.4](#_heading=h.2zbgiuw) Software Interfaces 6

[2.1.5](#_heading=h.1egqt2p) Communications Interfaces 6

[2.1.6](#_heading=h.3ygebqi) Memory Constraints 6

[2.1.7](#_heading=h.2dlolyb) Operations 6

[2.1.8](#_heading=h.sqyw64) Site Adaptation Requirements 6

[2.2](#_heading=h.z337ya) Product Functions 7

[2.3](#_heading=h.3j2qqm3) User Characteristics 7

[2.4](#_heading=h.1y810tw) Constraints 7

[2.5](#_heading=h.4i7ojhp) Assumptions and Dependencies 7

[2.6](#_heading=h.2xcytpi) Apportioning of Requirements 7

[**3.**](#_heading=h.1ci93xb) **SPECIFIC REQUIREMENTS 7**

[3.1](#_heading=h.3whwml4) External Interfaces 8

[3.2](#_heading=h.2bn6wsx) Functions 8

[3.3](#_heading=h.qsh70q) Performance Requirements 9

[3.4](#_heading=h.3as4poj) Logical Database Requirements 9

[3.5](#_heading=h.1pxezwc) Design Constraints 9

[3.5.1](#_heading=h.3cqmetx) Standards Compliance. 9

[3.6](#_heading=h.1rvwp1q) Software System Attributes 10

[3.6.1](#_heading=h.4bvk7pj) Reliability 10

[3.6.2](#_heading=h.23ckvvd) Availability 10

[3.6.3](#_heading=h.32hioqz) Security 10

[3.6.4](#_heading=h.41mghml) Portability 10

[3.7](#_heading=h.2r0uhxc) Organizing the Specific Requirements 11

[3.7.1](#_heading=h.1664s55) System Mode 11

[3.7.2](#_heading=h.3q5sasy) User Class 11

[3.7.3](#_heading=h.25b2l0r) Feature 11

[3.7.4](#_heading=h.kgcv8k) Stimulus 12

[3.7.5](#_heading=h.19c6y18) Response 12

[3.7.6](#_heading=h.34g0dwd) Functional Hierarchy 12

[3.8](#_heading=h.1jlao46) Additional Comments 12

[**4.**](#_heading=h.43ky6rz) **SUPPORTING INFORMATION. 12**

[**DOCUMENT CONTROL 13**](#_heading=h.2iq8gzs)

[Change History 13](#_heading=h.xvir7l)

[Document Storage 13](#_heading=h.3hv69ve)

[Document Owner 13](#_heading=h.2lwamvv)

[**APPENDICES 14**](#_heading=h.111kx3o)

# INTRODUCTION

*DATAHANKS is a web based social media application for education which will be useful for the students and teachers to interact well and get the study related materials for everyone to use them offline as well.*

## Purpose

*The purpose of this document is to capture, in natural language and at a functional level, the description and requirements of an educational website for any students and teachers. The focus here is the student’s educational needs. This is a functional description of those features required to address current educational requirements. A short discussion accompanies each requirement, to add the background and framework necessary to explain the functionality. It also describes nonfunctional requirements and other factors necessary to provide a complete and comprehensive description of the requirements for the software.*

## Scope

1. *The primary purpose of the student course management system is to upload the study related materials and get that study material downloaded for offline use. The system is based on a MongoDB database which will store the information of courses, corresponding materials and user’s details. We will have a database server which will manage all the information related to courses and users. Above all, we hope to provide a comfortable user experience along with the free access to all the courses.*
2. *User friendly interface so that anyone can get benefited from this website.*

## Definitions, acronyms, and abbreviations

|  |  |
| --- | --- |
| **Term or Acronym** | **Definition** |
| Educational Portal | Any user who has created account can upload and download the course material. Also, user can discuss in with the help of comment section. |
| Course Administrator | Only admin is allowed to modify or update the content and have the rights to delete the irrelevant comments. |

Table **x**. Definitions and Acronyms

## References

1. *Learnt Node JS from* [*https://www.w3schools.com/nodejs/*](https://www.w3schools.com/nodejs/)
2. *Learnt how to use MongoDB from* [*https://www.tutorialspoint.com/mongodb/index.htm*](https://www.tutorialspoint.com/mongodb/index.htm)
3. *Learnt Express JS* [*https://www.tutorialspoint.com/expressjs/index.htm*](https://www.tutorialspoint.com/expressjs/index.htm)

## Overview

*This is a working document and, as such, is subject to change. In its initial form, it is incomplete by definition, and will require continuing refinement. Requirements may be modified and additional requirements may be added as development progresses and the system description becomes more refined. This information will serve as a framework for the current definition and future evolution of the course management system’s portal.*

# Overall Description

## Problem Statement

|  |  |
| --- | --- |
| The problem of | *Lack of educational material which is free of cost.* |
| Affects | *Students and teachers.* |
| The impact of which is | *This will be a social media platform for the education which is free of cost to use.* |
| A successful solution would | *Give everyone the opportunity to use the study related materials without any cost and use them offline as well.* |

## Product Perspective

*DATAHANKS is meant to serve a common social media platform for everyone to share their thoughts and study related posts on it. Our goal is to develop a web -based application which will help most of the students to study using the materials available on the website and posted by teachers all over the world.*

*Database of DATAHANKS will store the following data:*

1. *User’s details like email-id, name, etc.*
2. *Courses which are uploaded by the students and teachers all over the world.*
3. *Comments which are commented by anyone on the particular course.*

## Product Position Statement

[Provide an overall statement summarizing at the highest level, the unique position the product intends to fill in the marketplace. The following format may be used:]

|  |  |
| --- | --- |
| For | *Students and teachers* |
| Who | *Opportunity to the people who can’t afford the costly materials available online and the needy students who have lack of internet connectivity.* |
| The (product name) | *DATAHANKS – A WEB BASED STUDENT COURSE MANAGEMENT SYSTEM.* |
| That | *Users of this website can freely download the material and can use them offline and can discuss their doubts through the comments section.* |
| Unlike | *(primary competitive alternative)* |
| Our product | *(statement of primary differentiation)* |

[A product position statement communicates the intent of the application and the importance of the project to all concerned personnel.]

### System Interfaces

*We are creating our own server using express.js with node.js runtime. Also, we are using live server extension from VS Code which will be used as a web server. The user inputs data via the web server using HTML forms. The actual program that will perform the operations is written in JavaScript. MongoDB will be used as a database management system to store the information.*

### User Interfaces

Our system will provide a very useful simple interface for both students and the teachers, so that the student can easily deal with the subjects they want to learn, ask their doubts directly through comment which will be displayed to everyone.

### Hardware Interfaces

1. ***Server side:***

*The web application will be hosted on a web server which is listening on the web*

*standard port.*

1. ***Client side:***

*Monitor screen – the software shall display information to the user via the monitor*

*screen.*

*Mouse – the software shall interact with the movement of the mouse and the mouse*

*buttons. The mouse shall activate areas for data input, command buttons and select*

*options from menus.*

*Keyboard – the software shall interact with the keystrokes of the keyboard. The*

*keyboard will input data into the active area of the database.*

### Software Interfaces

1. **Server side:**

*We are creating our own server using express.js with node.js runtime. Also, we are using live server extension from VS Code which will be used as a web server.*

A database will be hosted centrally using MongoDB.

1. **Client side:**

An OS which is capable of running a modern web browser which supports

JavaScript, HTML5, and bootstrap.

### Communications Interfaces

*The HTTP or HTTPS protocol(s) will be used to facilitate communication between the client and server.*

### Memory Constraints

*As MongoDB is a NoSQL based database it will expand dynamically, so there are no memory constraints.*

### Operations

*The product shall have operations to protect the database from being corrupted or accidentally altered during a system failure.*

### Site Adaptation Requirements

*Not Applicable.*

## Product Functions

* + 1. **Context Diagram:**

Students

Teachers

Search, upload/share Content. Able to add/remove/update comments.

Search, upload/share Content. Able to add comments.

Information about uploaded courses and user’s info.

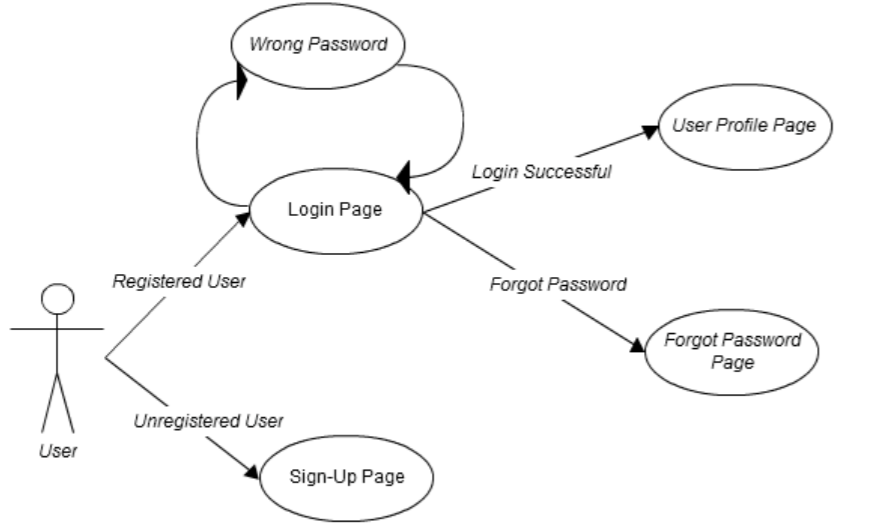
Only the users who have valid email id can login into the system.

Login Authentication

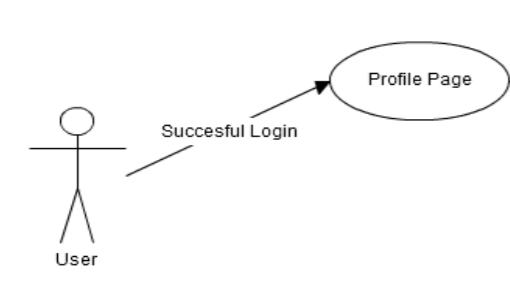
Database

**2.2.2 User Case Diagram:**

**2.2.2.1 User Login:**

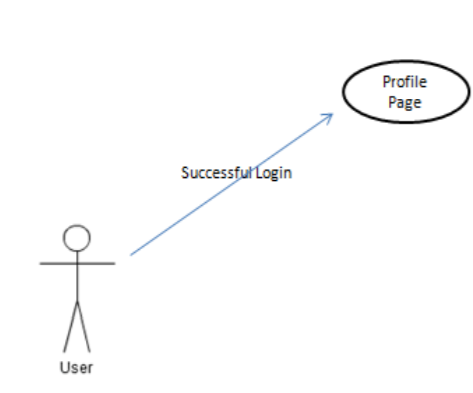


**2.2.2.2 Content Sharing:**



Available to all the users

**2.2.2.3 Search:**



Search Box

All files and comments uploaded

Retrieves data present in the database

User Database

## User Characteristics

***2.3.1 Students:***

*Students are the primary consumers of an academic portal. They are accessing information posted by professors, also they can upload the information related to that course and everyone can discuss the concepts through the comment section.*

***2.3.2 Professors:***

*Professors are the primary content administrators of the course. They can upload files, links and can discuss the student’s doubts with them.*

***2.3.3 Admin:***

*Admin are allowed to delete and update the courses as well as the comments on the courses which are commented by various users.*

## Constraints

***2.4.1 User Interface Constraints:*** *Using this system is fairly simple and intuitive. A user familiar with basic browser navigation skills should be able to understand all functionality provided by the system.*

***2.4.2 Hardware Constraints****: The system should work on most home desktop and laptop computers which support required frontend.*

***2.4.3 Software Constraints:*** *The system will be intended to run on Firefox, Google Chrome and Internet Explorer.*

***2.4.4 Design Standards:*** *Compliance The system shall be implemented in JavaScript, Node JS and Express JS.*

## Assumptions and Dependencies

*Most of the educational portals have a lot of redundant features which are rarely used in educational sessions. Our new system focuses on the features which are most important to the users of an educational institute along with the introduction of some new features which other educational portals lack.*

## Apportioning of Requirements

*Integration with the google authentication might be apportioned in future versions.*

# Specific Requirements

## External Interfaces

*3.1.1* ***Web Server:*** *The user inputs data via the web server using HTML forms. The web server executes the JavaScript as a module and Node JS retrieves the post data if available. The web server receives information back from the Node JS. The web server displays a HTML page as a result to the end-user.*

*3.1.2* ***JavaScript and Node JS Application:*** *The actual program that will perform the operations is written in JavaScript and Node JS. All data will be stored in a database.*

3.1.3 **MongoDB Database:** It’s an open-source NoSQL database to store all data which communicates with the application on the server.

## Functions

* + 1. ***Use Case scenario (User Login)***

|  |  |
| --- | --- |
| Purpose | User logs into the system using existing profile. |
| User | A user with an existing email id. |
| Input Data | Profile username and password. |
| Output Data | Corresponding page data. |
| Invariants | Profile table data and user information. |
| Pre-conditions | User is not logged in to a profile, input profile exists in data base, user password matches profile. |
| Post-conditions | User's computer has been supplied with appropriate cookie; page data is appropriate for selected profile |
| Basic Flow: | Invalid password, invalid username, or mismatched username and password redirect to try to login again page. |
| Business Rules: | This allows users to log in to their profile from anywhere. |

* + 1. ***Use Case scenario (Content Sharing)***

A user logs into the system and is on any page and wants to share some content to everyone.

|  |  |
| --- | --- |
| Purpose | A user wants to share some study related material. |
| User | A legitimate user logged into the system. |
| Input Data | The URL of file to be shared. |
| Output Data | File ready to download by other users. |
| Invariants | The File |
| Pre-conditions | User is Logged in; file exists on user's computer. |
| Post-conditions | Everyone to whom the content was made available is able to download it. |
| Basic Flow: | The user uploads a course or file to be shared using the upload box. The file then gets uploaded to the server and desired users are able to download it after logging in. |

* + 1. ***Use Case scenario (Comment section)***

A user starts a discussion through comment section in a particular which, to which the all the members who are visiting the course can contribute/respond.

|  |  |
| --- | --- |
| Purpose | A user wants to start a discussion on a particular topic or on particular doubt from that course. |
| User | Any user who has logged into the website. |
| Input Data | Text of the doubt or a discussion. |
| Output Data | The comments to which other users are able to view and reply. |
| Invariants | The discussion/doubt or comment data. |
| Pre-conditions | User is logged into the system. |
| Post-conditions | A comment section to which other users are able to view and reply. |
| Basic Flow: | User logs in, selects enters the text of discussion or doubt and submits it comment section. The data is then stored on the server and other members will be able to view and respond. |

* + 1. ***Use Case scenario (Download File)***

A user wants to download a particular course related material.

|  |  |
| --- | --- |
| Purpose | A user wants download a particular file. |
| User | Any user who has logged into the system. |
| Input Data | Request to download a particular file. |
| Output Data | File gets saved in the google drive of the user’s account. |
| Invariants | The user and the file. |
| Pre-conditions | User is logged in and the file must be shared with him. |
| Post-conditions | The user has downloaded the file successfully. |
| Basic Flow: | User logs in, selects the file which he wants to download. The file is then transferred from the server to the user’s goggle drive. |

* + 1. ***Use Case scenario (Search Results)***

|  |  |
| --- | --- |
| Purpose | A user wants to search for a particular course. |
| User | Any user of the system. |
| Input Data | The Keywords. |
| Output Data | Search Results. |
| Invariants | The user and the portal itself. |
| Pre-conditions | User is on the webpage. |
| Post-conditions | Search Results. |
| Basic Flow: | User visits the website, Enters the keyword in the search box, clicks the search button and gets the search results. |

## Performance Requirements

*This system should support most of the concurrent users.* This statement provides a general sense of reliability when the system is under load. It is important that a substantial number of users be able to access the system at the same time, since an academic portal is important to the courses that employ it. The times when the system will be under the most stress is likely during lot of users visits at a time. Therefore, it must be able to handle at least most of the concurrent users

## Logical Database Requirements

*All data will be saved in the database: user accounts and profiles, discussion data, etc. The database allows concurrent access and will be kept consistent at all times, requiring a good database design.*

## Design Constraints

*1. The communication between the portal software and the database will be in NoSQL.*

*2. The portal layout will be produced with HTML/CSS and JavaScript.*

*3. The product will be written in JavaScript.*

*4. The source code must follow the coding conventions of JavaScript. System administrators must have access to comprehensive documentation.*

### Standards Compliance.

*Specify the requirements derived from existing standards or regulations. They might include:*

1. *Report format*
2. *Data naming*
3. *Accounting procedures*
4. *Audit Tracing*

*For 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.*

## Software System Attributes

The software consists of the following elements:

1. VS Code

2. Goggle Chrome or any other web browser

3. The MongoDB compass and atlas

4. The database should remain consistent at all times in case of an error.

### Reliability

The reliability of the overall program depends on the reliability of the separate components.

### Availability

The system should be available at all times, meaning the user can access it using a web browser, only restricted by the down time of the server on which the system runs. In case of a of a hardware failure or database corruption, a flash message will be shown. Also, in case of a hardware failure or database corruption, backups of the database should be retrieved with the MongoDB server and saved by the administrator.

### Security

1. Passwords will be saved encrypted in the database in order to ensure the user's privacy.

2. The user's IP will be logged.

3. The system will be protected against vulnerabilities.

### Portability

The application should be compatible with most of the operating systems. Apache, JavaScript and MongoDB programs are practically independent of the OS-system which they communicate with. The end-user part is fully portable and any system using any web browser should be able to use the features of the application.

|  |  |  |
| --- | --- | --- |
| **ID** | **Characteristic** | **Rank** |
| 1 | Correctness | 3 |
| 2 | Efficiency | 2 |
| 3 | Flexibility | 1 |

## Organizing the Specific Requirements

*For anything but trivial systems the detailed requirements tend to be extensive. For this reason, it is recommended that careful consideration be given to organizing these in a manner optimal for understanding. There is no one optimal organization for all systems. Different classes of systems lend themselves to different organizations of requirements in section 3. Some of these organizations are described in the following subsections.*

### System Mode

*System behaves different based on the mode of operation. Organizing by mode there are two possible outlines. The choice depends on whether interfaces and performance are dependent on which mode.*

### User Class

*There will be a class of users who have uploaded the course and they have the rights to:*

* + 1. *Change or update the course.*
    2. *Delete the course.*
    3. *Remove the irrelevant comments.*

### Feature

*DATAHANKS provides features like add/update/delete the course. Also, any user is able to comment on the course but only admin of that course is allowed to remove the comment.*

### Stimulus

1. User request to create a new account.

2. User request to add a course.

3. User request to search for particular course.

4. User request to add comment on the course.

5. User request to update course or comment.

### Response

1. System provides Sign up form to create an account.

2. System provides a form for the user to enter the course data.

3. System provides the all the available results related to the search.

4. System provides the text view to enter the comment.

5. The course data is presented in an editable format. When finished, changes may be saved or discarded.

### Functional Hierarchy

*When none of the above organizational schemes prove helpful, the overall functionality can be organized into a hierarchy of functions organized by either common inputs, common outputs, or common internal data access. Data flow diagrams and data dictionaries can be used to show the relationships between and among the functions and data.*

## Additional Comments

*Whenever a new SRS is contemplated, more than one of the organizational techniques given in 3.7 may be appropriate. In such cases, organize the specific requirements for multiple hierarchies tailored to the specific needs of the system under specification. Any additional requirements may be put in a separate section at the end of the SRS.*

*There are many notations, methods, and automated support tools available to aid in the documentation of requirements. For the most part, their usefulness is a function of organization. For example, when organizing by mode, finite state machines or state charts may prove helpful; when organizing by object, object-oriented analysis may prove helpful; when organizing by feature, stimulus-response sequences may prove helpful; when organizing by functional hierarchy, data flow diagrams and data dictionaries may prove helpful.*

*In any of the outlines below, those sections called “Functional Requirement i” may be described in native language, in pseudocode, in a system definition language, or in four subsections titled: Introduction, Inputs, Processing, Outputs.*

# Supporting Information.

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

1. Table of Contents at the front of the document
2. Index
3. Appendices

*The Appendices are not always considered part of the actual requirements specification and are not always necessary. They may include:*

1. *Sample I/O formats, descriptions of cost analysis studies, results of user surveys;*
2. *Supporting or background information that can help the readers of the SRS;*
3. *A description of the problems to be solved by the software;*
4. *Special packaging instructions for the code and the media to meet security, export, initial loading, or other requirements.*

*When Appendices are included, the SRS should explicitly state whether or not the Appendices are to be considered part of the requirements.*

Tables on the following pages provide alternate ways to structure section 3 on the specific requirements.

# Document Control

**Change History**

|  |  |  |
| --- | --- | --- |
| **Revision** | **Release Date** | **Description [list of changed pages and reason for change]** |
|  |  |  |
|  |  |  |

**Document Storage**

This document was created using Microsoft Word. The file is stored in a computer.

**Document Owner**

Tejas Kachare, Rushikesh Chounde and Naman Chandak are responsible for developing and maintaining this document.

**Appendices**

**A.1 Outline for SRS Section 3: Organized by mode: Version 1**

3 Specific Requirements

3.1 External interface requirements

1. User interfaces
2. Hardware interfaces
3. Software interfaces
4. Communications interfaces
5. Functional requirements

3.2.1 Mode 1

3.2.1.1 Functional requirement 1.1

.....

3.2.1.*n* Functional requirement 1.*n*

1. Mode 2

.....

3.2.*m* Mode *m*

3.2.*m*.1 Functional requirement *m*.1

.....

3.2.*m.n* Functional requirement *m.n*

3.3 Performance Requirements

3.4 Design Constraints

3.5 Software system attributes

3.6 Other requirements

**Outline for SRS Section 3: Organized by mode: Version 2**

3 Specific Requirements

3.1 Functional Requirements

1. Mode 1

3.1.1.1 External interfaces

3.1.1.1 User interfaces

3.1.1.2 Hardware interfaces

3.1.1.3 Software interfaces

3.1.1.4 Communications interfaces

3.1.1.2 Functional Requirement

3.1.1.2.1 Functional requirement 1

.....

3.1.1.2.*n* Functional requirement *n*

3.1.1.3 Performance

3.1.2 Mode 2

.....

3.1.*m* Mode *m*

1. Design constraints
2. Software system attributes
3. Other requirements

**Outline for SRS Section 3: Organized by user class**

3 Specific Requirements

3.1 External interface requirements

1. User interfaces
2. Hardware interfaces
3. Software interfaces
4. Communications interfaces
5. Functional requirements

3.2.1 User class 1

3.2.1.1 Functional requirement 1.1

.....

3.2.1.*n* Functional requirement 1.*n*

1. User class 2

.....

3.2.*m* User class *m*

3.2.*m*.1 Functional requirement *m*.1

.....

3.2.*m.n* Functional requirement *m.n*

3.3 Performance Requirements

3.4 Design Constraints

3.5 Software system attributes

3.6 Other requirements

**Outline for SRS Section 3: Organized by object**

3 Specific Requirements

3.1 External interface requirements

1. User interfaces
2. Hardware interfaces
3. Software interfaces
4. Communications interfaces
5. Classes/Objects

3.2.1 Class/Object 1

3.2.1.1 Attributes (direct or inherited)

1. Attribute 1

.....

3.2.1.1.*n* Attribute *n*

1. Functions (services, methods, direct or inherited)

3.2.1.2.1 Functional requirement 1.1

.....

3.2.1.2.*m* Functional requirement 1.*m*

3.2.1.3 Messages (communications received or sent)

3.2.2 Class/Object 2

.....

3.2.*p* Class/Object *p*

3.3 Performance Requirements

3.4 Design Constraints

3.5 Software system attributes

3.6 Other requirements

**Outline for SRS Section 3: Organized by use case**

An alternative to embedding the use cases in line with the functional requirements (as shown below) is to provide the use cases separately from the SRS and refer to the use case name or ID for each functional requirement in the SRS.

3 Specific Requirements

3.1 External interface requirements

1. User interfaces
2. Hardware interfaces
3. Software interfaces
4. Communications interfaces
5. Functional Requirements

3.1.1.2 Functional Requirement

3.1.1.2.1 Functional requirement 1

Use case for functional requirement 1

.....

3.1.1.2.*n* Functional requirement *n*

Use case for functional requirement *n*

3.3 Performance Requirements

3.4 Design Constraints

3.5 Software system attributes

3.6 Other requirements

**Outline for SRS Section 3: Organized by feature**

3 Specific Requirements

3.1 External interface requirements

1. User interfaces
2. Hardware interfaces
3. Software interfaces
4. Communications interfaces
5. System features

3.2.1 System Feature 1

3.2.1.1 Introduction/Purpose of feature

3.2.1.2 Stimulus/Response sequence

3.2.1.3 Associated functional requirements

3.2.1.3.1 Functional requirement 1

.....

3.2.1.3.*n* Functional requirement *n*

3.2.2 System Feature 2

.....

3.2.*m* System Feature *m*

.....

3.3 Performance Requirements

3.4 Design Constraints

3.5 Software system attributes

3.6 Other requirements

**Outline for SRS Section 3: Organized by stimulus**

3 Specific Requirements

3.1 External interface requirements

1. User interfaces
2. Hardware interfaces
3. Software interfaces
4. Communications interfaces
5. Functional requirements

3.2.1 Stimulus 1

3.2.1.1 Functional requirement 1.1

.....

3.2.1.*n* Functional requirement 1.*n*

3.2.2 Stimulus 2

.....

3.2.*m* Stimulus *m*

3.2.*m*.1 Functional requirement *m*.1

.....

3.2.*m.n* Functional requirement *m.n*

3.3 Performance Requirements

3.4 Design Constraints

3.5 Software system attributes

3.6 Other requirements

**Outline for SRS Section 3: Organized by response**

3 Specific Requirements

3.1 External interface requirements

1. User interfaces
2. Hardware interfaces
3. Software interfaces
4. Communications interfaces
5. Functional requirements

3.2.1 Response 1

3.2.1.1 Functional requirement 1.1

.....

3.2.1.*n* Functional requirement 1.*n*

3.2.2 Response 2

.....

3.2.*m* Response *m*

3.2.*m*.1 Functional requirement *m*.1

.....

3.2.*m.n* Functional requirement *m.n*

3.3 Performance Requirements

3.4 Design Constraints

3.5 Software system attributes

3.6 Other requirements

**Outline for SRS Section 3: Organized by functional hierarchy**

3 Specific Requirements

3.1 External interface requirements

1. User interfaces
2. Hardware interfaces
3. Software interfaces
4. Communications interfaces
5. Functional requirements

3.2.1 Information flows

3.2.1.1 Data flow diagram 1

1. Data entities
2. Pertinent processes
3. Topology

3.2.1.2 Data flow diagram 2

1. Data entities
2. Pertinent processes
3. Topology

.....

3.2.1.*n* Data flow diagram *n*

3.2.1.*n*.1 Data entities

3.2.1.*n*.2 Pertinent processes

3.2.1.*n*.3 Topology

3.2.2 Process descriptions

1. Process 1
2. Input data entities
3. Algorithm or formula of process
4. Affected data entities

3.2.2.2 Process 2

3.2.2.2.1 Input data entities

3.2.2.2.2 Algorithm or formula of process

3.2.2.2.3 Affected data entities

.….

3.2.2.*m* Process *m*

3.2.2.*m*.1 Input data entities

3.2.2.*m*.2 Algorithm or formula of process

3.2.2.*m*.3 Affected data entities

3.2.3 Data construct specifications

3.2.3.1 Construct 1

3.2.3.1.1 Record type

3.2.3.1.2 Constituent fields

3.2.3.2 Construct 2

3.2.3.2.1 Record type

3.2.3.2.2 Constituent fields

…..

3.2.3.*p* Construct *p*

3.2.3.*p*.1 Record type

3.2.3.*p*.2 Constituent fields

3.2.4 Data dictionary

3.2.4.1 Data element 1

3.2.4.1.1 Name

3.2.4.1.2 Representation

3.2.4.1.3 Units/Format

3.2.4.1.4 Precision/Accuracy

3.2.4.1.5 Range

3.2.4.2 Data element 2

3.2.4.2.1 Name

3.2.4.2.2 Representation

3.2.4.2.3 Units/Format

3.2.4.2.4 Precision/Accuracy

3.2.4.2.5 Range

…..

3.2.4.*q* Data element *q*

3.2.4.*q*.1 Name

3.2.4.*q*.2 Representation

3.2.4.*q*.3 Units/Format

3.2.4.*q*.4 Precision/Accuracy

3.2.4.*q*.5 Range

3.3 Performance Requirements

3.4 Design Constraints

3.5 Software system attributes

3.6 Other requirements

**Outline for SRS Section 3: Showing multiple organizations**

3 Specific Requirements

3.1 External interface requirements

1. User interfaces
2. Hardware interfaces
3. Software interfaces
4. Communications interfaces
5. Functional requirements

3.2.1 User class 1

3.2.1.1 Feature 1.1

3.2.1.1.1 Introduction/Purpose of feature

3.2.1.1.2 Stimulus/Response sequence

3.2.1.1.3 Associated functional requirements

3.2.1.2 Feature 1.2

3.2.1.2.1 Introduction/Purpose of feature

3.2.1.2.2 Stimulus/Response sequence

3.2.1.2.3 Associated functional requirements

…..

3.2.1.*m* Feature 1.*m*

3.2.1.*m*.1 Introduction/Purpose of feature

3.2.1.*m*.2 Stimulus/Response sequence

3.2.1.*m*.3 Associated functional requirements

3.2.2 User class 2

.....

3.2.*n* User class *n*

.....

3.3 Performance Requirements

3.4 Design Constraints

3.5 Software system attributes

3.6 Other requirements