Software Requirements Specification

for

Project Name: XXXX

Version <1.0> approved

Prepared by: Kv-030.Java team

Organization: SoftServe

Date: XX.XX.2017

**Table of Contents**

[1. Introduction 3](#_Toc496783732)

[1.1 Purpose 4](#_Toc496783733)

[1.2 Scope 4](#_Toc496783734)

[1.3 Product overview 4](#_Toc496783735)

[1.3.1 Product perspective 4](#_Toc496783736)

[1.3.2 Product functions 5](#_Toc496783737)

[1.3.3 User characteristics 7](#_Toc496783738)

[1.3.4 Limitations 7](#_Toc496783739)

[1.4 Definitions 7](#_Toc496783740)

[2. References 8](#_Toc496783741)

[3. Specific requirements 8](#_Toc496783742)

[3.1 External interfaces 8](#_Toc496783743)

[3.2 Functions 8](#_Toc496783744)

[3.3 Usability Requirements 9](#_Toc496783745)

[3.4 Performance requirements 9](#_Toc496783746)

[3.5 Logical database requirements 10](#_Toc496783747)

[3.6 Design constraints 11](#_Toc496783748)

[3.7 Software system attributes 11](#_Toc496783749)

[Reliability 11](#_Toc496783750)

[Availability 11](#_Toc496783751)

[Security 12](#_Toc496783752)

[Maintainability 12](#_Toc496783753)

[Portability 12](#_Toc496783754)

[3.8 Supporting information 13](#_Toc496783755)

[4. Verification *(parallel to subsections in Section 3)* 13](#_Toc496783756)

[4.1 External interfaces 13](#_Toc496783757)

[4.2 Functions 13](#_Toc496783758)

[4.3 Usability Requirements 13](#_Toc496783759)

[4.4 Performance requirements 13](#_Toc496783760)

[4.5 Logical database requirements 14](#_Toc496783761)

[4.6 Design constraints 14](#_Toc496783762)

[4.7 Software system attributes 14](#_Toc496783763)

[4.8 Supporting information 14](#_Toc496783764)

[5. Appendices 14](#_Toc496783765)

[5.1 Assumptions and dependencies 14](#_Toc496783766)

[5.2 Acronyms and abbreviations 14](#_Toc496783767)

# 

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
|  |  |  |  |
|  |  |  |  |

# 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

The purpose of this document is to give a detailed description of the requirements for the <*Product name*> system. It will illustrate the purpose and complete declaration for the development of system. It will also explain system constraints, interface and interactions with other external applications.

This document is primarily intended to be proposed to a customer for its approval and a reference for developing the first version of the system for the development team.

*<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

<*Product name*> is a software product intended for secure storage of corporate documentation and for secure documents sharing within premises of the customer company. The employees should be able to select colleagues they share the document with, while others won’t be able to read this document.

This system will allow the customer company to outsource documents storage to cloud storage provider. In case of a hacker attack on the cloud storage provider, the risks of the company’s documents to be stolen by some competing company or be made public will be minimized as much as possible.

Storing documents on cloud will reduce workload on company’s own server, which in turn will reduce the cost for maintaining the company's own server.

*<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 Product overview

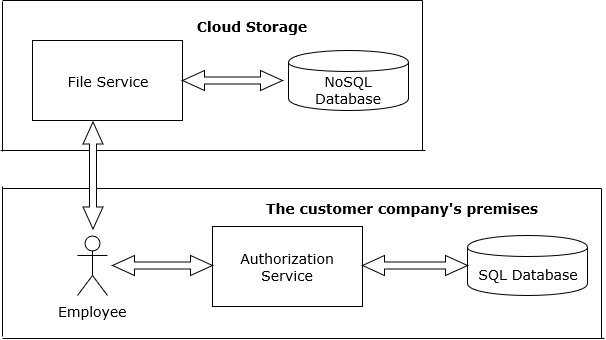
### 1.3.1 Product perspective

This system should be tailored in accordance with needs of the customer company and free of any redundancy functionality, not be publically available. Similar-functionality software available on market place is not likely to satisfy above-mentioned criteria, so new software system has to be developed.

The system will consist of two services: Authorization Service and File Service, see Figure 1.

Authorization service should be deployed on customer’s own server. It will use locally hosted database for related data storage. No documents should be stored in this database, only information sufficient to retrieve document from cloud storage. Users of this system will communicate with Authorization service via Intranet.

File service should be deployed outside customer’s premises, on cloud/third party storage. File service should use cloud/storage provider’s database for document storage. Only ciphered documents can be passed to this service. Internet connection is required to communicate with the File service.



*Figure 1 - Block diagram*

*<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.*

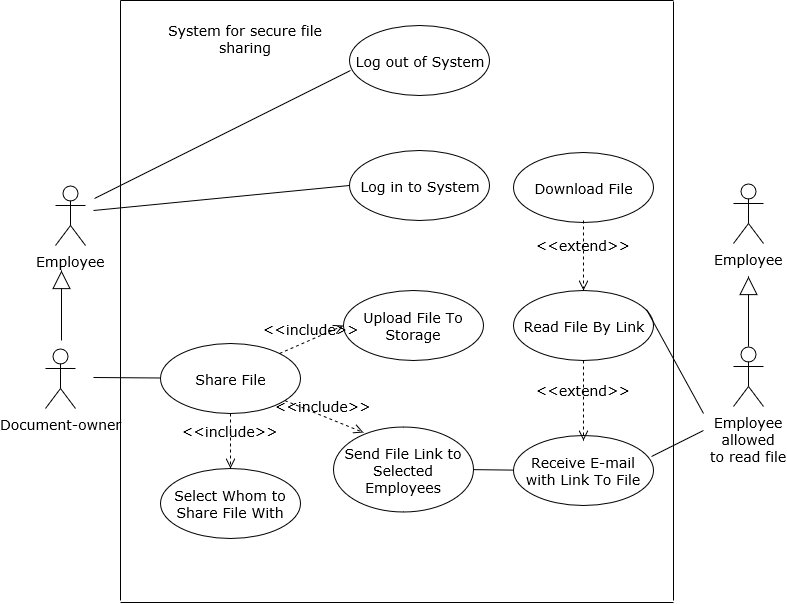
*The following subsections describe how the software operates inside various constraints*.>

### 1.3.2 Product functions

Basic features of the <*Product name*>:

* Employees should log in to this system.
* Every employee should be able to share documents with other employees of the company.
* The employee, that shares the document, should be opt to select who is able to read the document:
  + all employees of the company
  + only certain employees of the company
* Selected employees receive the e-mail notification with link to the shared document.
* Only allowed employees can open the link and download the document. In case “not-allowed to read this document” employee clicks the link, the link must not open the document for reading.

Use case diagram is displayed on Figure 2:



*Figure 2 – Use case diagram*

*<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. 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).>*

### 1.3.3 User characteristics

This system will be used by those employees of the customer company, which have corporate e-mails. Education, professional background, technical skills of employees varies, but sufficient to use corporate e-mails, web, friendly GUI applications.

*<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.>*

### 1.3.4 Limitations

This system will be used by the customer company only.

## 1.4 Definitions

**Database** – A database is an organized collection of data. Formally, a "database" refers to a set of related data and the way it is organized.

**Cloud storage** - is a model of data storage in which the digital data is stored in logical pools, the physical storage spans multiple servers (and often locations), and the physical environment is typically owned and managed by a hosting company. These cloud storage providers are responsible for keeping the data available and accessible, and the physical environment protected and running. People and organizations buy or lease storage capacity from the providers to store user, organization, or application data.

**SQL** - Structured Query Language, is a domain-specific language used in programming and designed for managing data held in a relational database management system (RDBMS), or for stream processing in a relational data stream management system (RDSMS).

*<Provide the definitions of all terms, acronyms, and abbreviations required to properly interpret the SRS. >*

# 2. References

This document does not reference any other document.

.

*<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.* >

# 3. Specific requirements

## 3.1 External interfaces

*<This contains a detailed description of all inputs into and outputs from the software system. It complements the interface descriptions in section 2 but does not repeat information there. Remember section 2 presents information oriented to the customer/user while section 3 is oriented to the developer.*

*It contains both content and format as follows:*

1. *Name of item*
2. *Description of purpose*
3. *Source of input or destination of output*
4. *Valid range, accuracy and/or tolerance*
5. *Units of measure*
6. *Timing*
7. *Relationships to other inputs/outputs*
8. *Screen formats/organization*
9. *Window formats/organization*
10. *Data formats*
11. *Command formats*
12. *End messages*

*>*

## 3.2 Functions

*<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:*

* *Validity checks on the inputs*
* *Exact sequence of operations*
* *Responses to abnormal situation, including*
* *Overflow*
* *Communication facilities*
* *Error handling and recovery*
* *Effect of parameters*
* *Relationship of outputs to inputs, including*
* *Input/Output sequences*
* *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.>*

## 3.3 Usability Requirements

*<Define usability (quality in use) requirements. Usability requirements and objectives for the software system include measurable effectiveness, efficiency, and satisfaction criteria in specific contexts of use.>*

## 3.4 Performance requirements

*<This subsection specifies 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:*

*(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.*

*All of these requirements should be stated in measurable terms.*

*For example,*

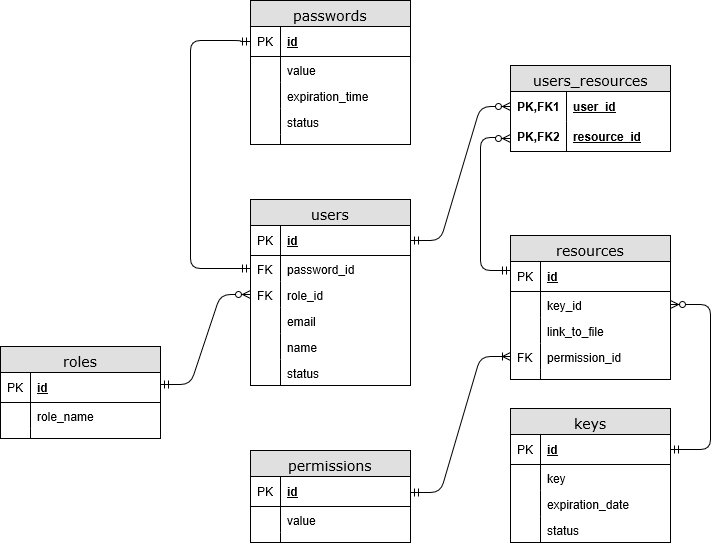
*95% of the transactions shall be processed in less than 1 second*

*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.)>*

## 3.5 Logical database requirements



*Figure ? – ER diagram*

*<This section specifies the logical requirements for any information that is to be placed into a database. This may include:*

* *Types of information used by various functions*
* *Frequency of use*
* *Accessing capabilities*
* *Data entities and their relationships*
* *Integrity constraints*
* *Data retention requirements*

*If the customer provided you with data models, those can be presented here. ER diagrams (or static class diagrams) can be useful here to show complex data relationships. Remember a diagram is worth a thousand words of confusing text.>*

## 3.6 Design constraints

*<Specify design constraints that can be imposed by other standards, hardware limitations, etc.>*

## 3.7 Software system attributes

*<There are a number of attributes of software that can serve as requirements. It is important that required attributes by specified so that their achievement can be objectively verified. The following items provide a partial list of examples. These are also known as non-functional requirements or quality attributes.*

*These are characteristics the system must possess, but that pervade (or cross-cut) the design. These requirements have to be testable just like the functional requirements. Its easy to start philosophizing here, but keep it specific.*

### 

### Reliability

*Specify the factors required to establish the required reliability of the software system at time of delivery. If you have MTBF requirements, express them here. This doesn’t refer to just having a program that does not crash. This has a specific engineering meaning.*

### Availability

*Specify the factors required to guarantee a defined availability level for the entire system such as checkpoint, recovery, and restart. This is somewhat related to reliability. Some systems run only infrequently on-demand (like MS Word). Some systems have to run 24/7 (like an e-commerce web site). The required availability will greatly impact the design. What are the requirements for system recovery from a failure? “The system shall allow users to restart the application after failure with the loss of at most 12 characters of input”.*

### Security

*Specify the factors that would protect the software from accidental or malicious access, use, modification, destruction, or disclosure. Specific requirements in this area could include the need to:*

1. *Utilize certain cryptographic techniques*
2. *Keep specific log or history data sets*
3. *Assign certain functions to different modules*
4. *Restrict communications between some areas of the program*
5. *Check data integrity for critical variables*

### Maintainability

*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. If someone else will maintain the system*

### Portability

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

1. *Percentage of components with host-dependent code*
2. *Percentage of code that is host dependent*
3. *Use of a proven portable language*
4. *Use of a particular compiler or language subset*
5. *Use of a particular operating system*

*Once the relevant characteristics are selected, a subsection should be written for each, explaining the rationale for including this characteristic and how it will be tested and measured. A chart like this might be used to identify the key characteristics (rating them High or Medium), then identifying which are preferred when trading off design or implementation decisions (with the ID of the preferred one indicated in the chart to the right). The chart below is optional (it can be confusing) and is for demonstrating tradeoff analysis between different non-functional requirements. H/M/L is the relative priority of that non-functional requirement.*

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **ID** | **Characteristic** | **H/M/L** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** |
| 1 | Correctness |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 2 | Efficiency |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 3 | Flexibility |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 4 | Integrity/Security |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 5 | Interoperability |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 6 | Maintainability |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 7 | Portability |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 8 | Reliability |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 9 | Reusability |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 10 | Testability |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 11 | Usability |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 12 | Availability |  |  |  |  |  |  |  |  |  |  |  |  |  |

*Definitions of the quality characteristics not defined in the paragraphs above follow.*

*• Correctness - extent to which program satisfies specifications, fulfills user’s mission objectives*

*• Efficiency - amount of computing resources and code required to perform function*

*• Flexibility - effort needed to modify operational program*

*• Interoperability - effort needed to couple one system with another*

*• Reliability - extent to which program performs with required precision*

*• Reusability - extent to which it can be reused in another application*

*• Testability - effort needed to test to ensure performs as intended*

*• Usability - effort required to learn, operate, prepare input, and interpret output*

*>*

## 3.8 Supporting information

# 4. Verification *(parallel to subsections in Section 3)*

*<Provide the verification approaches and methods planned to qualify the software. The information items for verification are recommended to be given in a parallel manner with the information items in clause 3.>*

## 4.1 External interfaces

## 4.2 Functions

## 4.3 Usability Requirements

## 4.4 Performance requirements

## 4.5 Logical database requirements

## 4.6 Design constraints

## 4.7 Software system attributes

## 4.8 Supporting information

# 5. Appendices

## 5.1 Assumptions and dependencies

*<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. For example, an assumption might be that a specific operating system would be available on the hardware designated for the software product. If, in fact, the operating system were not available, the SRS would then have to change accordingly.*

*This section is catch-all for everything else that might influence the design of the system and that did not fit in any of the categories above.>*

## 5.2 Acronyms and abbreviations

**DB** – Database

**SQL** – Structured Query Language

**NoSQL** - not only SQL (Structured Query Language)